

Self-Service Administration and Operations Guide

Self-Service 3.8.1.1

May 26, 2025

Contents

Introduction to Nutanix Cloud Manager Self-Service.....	8
Self-Service Key Capabilities.....	8
Where to Access App Management Capabilities.....	9
Self-Service Prerequisites and Deployment.....	10
Self-Service Benchmarks.....	10
Port Information in Self-Service.....	12
Self-Service Deployment.....	13
Prerequisites to Deploy Self-Service.....	13
Getting Started with Self-Service.....	14
Pre-configuration for Using Self-Service.....	14
Checking Self-Service Version.....	14
Self-Service Overview.....	15
Exploring Self-Service.....	16
Accessing Self-Service REST API Explorer.....	17
Self-Service Role-Based Access Control.....	18
Self-Service VM Deployment.....	22
Deploying Self-Service VM on AHV.....	22
Launching Self-Service VM with a Static IP Address on AHV.....	25
Setting Up Self-Service VM.....	27
Setting up Scale-Out Self-Service VM.....	28
Enabling Microservices Infrastructure on Self-Service VM.....	30
Enabling Policy Engine for Self-Service VM.....	32
Hardening Self-Service VM.....	33
Self-Service DSL Overview.....	37
Quick Start with Self-Service.....	40
Launching a Blueprint Instantaneously.....	40
Provisioning a Linux or Windows Infrastructure as a Service (IaaS).....	40
Self-Service Settings.....	42
General Settings.....	42
Showback.....	42
App Protection Status Overview.....	43
Policy Settings.....	44
Policy Engine and Quotas Overview.....	44
Setting up Quota defaults.....	45
Approval Settings.....	45
Credential Settings.....	48
Credentials Overview.....	48

Configuring a Credential Provider.....	49
Provider Account Settings in Self-Service.....	52
Nutanix Account Configuration.....	52
Support for the Multi-Prism Central Setup.....	53
Tunnels for Orchestration within a VPC.....	54
Configuring a VMware Account.....	58
Permission Required in vCenter.....	59
Supported vSphere Versions.....	61
Configuring an AWS Account.....	61
Configuring AWS C2S Provider on Self-Service.....	62
Configuring AWS User Account with Minimum Privilege.....	63
AWS Policy Privileges.....	65
Configuring a GCP Account.....	66
Configuring an Azure Account.....	67
Configuring Azure User Account with Minimum Privilege.....	68
Configuring a Kubernetes Account.....	69
Configuring Amazon EKS, Azure Kubernetes Service, Anthos, or Red Hat OpenShift.....	72
Configuring a Nutanix Cloud Account.....	72
Platform Sync for Provider Accounts.....	73
Synchronizing Platform Configuration Changes.....	73
Allocating Resource Quota to an Account.....	74
Viewing Quota Utilization Report.....	75
Connecting Nutanix Database Service (NDB) to a Nutanix Account.....	76
Projects Overview.....	78
Environments in Self-Service.....	80
Environment Patching Behavior.....	81
Patching for Clusters and Subnets.....	85
Self-Service Blueprints Overview.....	87
Services Overview.....	90
Macros Overview.....	91
Built-in Macros.....	93
Platform Macros.....	95
Endpoint Macros.....	96
Runbook Macros.....	98
Virtual Machine Common Properties.....	98
Variables Overview.....	99
Nutanix Variables.....	101
VMware Variables.....	101
AWS Variables.....	101
GCP Variables.....	102
Azure Variables.....	103
Kubernetes Variables.....	103
Runtime Variables Overview.....	104
Categories Overview.....	104
Single-VM Blueprints in Self-Service.....	106
Creating a Single-VM Blueprint.....	106

Setting up a Single-VM Blueprint.....	106
Adding VM Details to a Blueprint.....	107
VM Configuration.....	107
Configuring VM for Nutanix Account.....	107
Configuring VM for VMware Account.....	110
Configuring VM for GCP Account.....	113
Configuring VM for AWS Account.....	116
Configuring VM for Azure Account.....	118
Configuring VM for Nutanix Cloud Account.....	121
Configuring Advanced Options for a Blueprint.....	123
Configuring Tasks or Packages in a Blueprint.....	124
Configuring App Variables in a Blueprint.....	126
 Multi-VM Blueprints in Self-Service.....	 130
Creating a Multi-VM Blueprint.....	130
Adding a Service.....	130
Configure Multi-VM, Package, and Service.....	131
Configuring Nutanix and Existing Machine VM, Package, and Service.....	132
Configuring VMware VM, Package, and Service.....	138
Configuring AWS VM, Package, and Service.....	145
Configuring GCP VM, Package, and Service.....	149
Configuring Azure VM, Package, and Service.....	153
Configuring Nutanix Cloud VM, Package, and Service.....	159
Configuring Kubernetes Deployment, Containers, and Service.....	162
Setting up the Service Dependencies.....	165
Adding and Configuring an App Profile.....	166
 Brownfield App Overview.....	 170
Importing Brownfield App.....	170
Launching Brownfield App.....	171
 Blueprint Configurations in Self-Service.....	 173
Configuring a Blueprint.....	173
Adding Credentials.....	173
Configuring Check Log-In.....	174
Tasks Overview.....	176
Adding an Execute Task.....	176
Adding a Set Variable Task.....	179
Adding an HTTP Task.....	185
Adding a Delay Task.....	186
Adding a Pre-create, Post-create, or Post-delete Task.....	187
Pre-create Task Workflow.....	189
Actions Overview.....	192
Adding an Action to a Single-VM Blueprint.....	194
Adding an Action to a Multi-VM Blueprint.....	196
Adding and Configuring Scale Out and Scale In.....	198
Blueprint Configuration for Snapshots and Restore.....	199
Configuring Single-VM Blueprints on Nutanix for Snapshots.....	199
Configuring Multi-VM Blueprints on Nutanix for Snapshots.....	201
Configuring Single-VM Blueprints on VMware for Snapshots.....	203
Configuring Multi-VM Blueprints on VMware for Snapshots.....	204
Update Configuration for VM.....	205
Adding Single-VM Update Configuration on Nutanix.....	206

Adding Multi-VM Update Configuration on Nutanix.....	210
Adding Single-VM Update Configuration on VMware.....	212
Adding Multi-VM Update Configuration on VMware.....	216
Blueprints Management in Self-Service.....	221
Submitting a Blueprint for Approval.....	222
Launching a Blueprint.....	224
Platform Validation Errors.....	226
Uploading a Blueprint.....	226
Downloading a Blueprint.....	227
Viewing a Blueprint.....	227
Editing a Blueprint.....	228
Deleting a Blueprint.....	228
Viewing Blueprint Error.....	228
Recovering Deleted Blueprints.....	229
Nutanix Marketplace Overview.....	230
Marketplace Manager in Self-Service.....	231
Marketplace Manager Overview.....	231
Marketplace Version.....	232
Approving and Publishing a Blueprint or Runbook.....	232
Unpublishing a Blueprint or Runbook.....	233
Deploying a Pre-Seeded App from Self-Service.....	234
Deleting an Unpublished Blueprint or Runbook.....	235
Applications in Self-Service.....	236
Policies in Self-Service.....	238
Scheduler Overview.....	238
Creating a Scheduler Job.....	238
Viewing and Updating Scheduler Jobs.....	240
Deleting a Scheduler Job.....	241
Approval Policy Overview.....	242
Creating an Approval Policy.....	244
Cloning an Approval Policy.....	255
Enabling or Disabling an Approval Policy.....	255
Deleting an Approval Policy.....	256
Viewing an Approval Policy Details.....	256
Approving or Rejecting an Approval Request.....	257
Library in Self-Service.....	259
Library Overview.....	259
Variable Types Overview.....	259
Creating Variable Types.....	259
Task Library Overview.....	263
Adding a Task to a Project.....	263
Deleting a Task from the Task Library.....	263

Runbooks in Self-Service.....	265
Runbooks Overview.....	265
Runbook Sharing across Projects.....	265
Creating a Runbook.....	266
Creating a Runbook with an Execute or Set Variable Task.....	270
Creating a Runbook with a Delay Task.....	272
Creating a Runbook with an HTTP Task.....	272
Creating a Runbook with a While Loop Task.....	273
Nutanix Database Service (NDB) Integration with Self-Service Runbooks.....	274
Configuring a Runbook for Database Management.....	274
Configuring the Clone Action for Database Management Runbook.....	277
Configuring Snapshot Action for Database Management Runbook.....	278
Configuring Create Action for Database Management Runbook.....	279
Configuring Restore Action for Database Management Runbook.....	282
Configuring Delete Action for Database Management Runbook.....	282
Submitting a Runbook for Publishing.....	283
Executing a Runbook.....	285
Runbook Execution using a Playbook.....	286
Deleting a Runbook.....	286
 Multitenant Capability for Service Providers.....	 288
Tenant Onboarding Runbook.....	289
External Subnet Management Runbook.....	300
Virtual Private Cloud Management Runbook.....	306
Overlay Subnet Management Runbook.....	309
Users and Groups Management Runbook.....	319
VPC Static Routing Runbook.....	330
VLAN Subnet Management Runbook.....	334
Disaster Recovery Runbook.....	342
Network Configuration Runbook.....	355
Policy-Based Routing Runbook.....	362
Backup and Restore Runbook.....	373
Floating IP Assignment Runbook.....	377
Single-VM Blueprint Runbook.....	379
Test Failover Runbook.....	383
 Endpoints in Self-Service.....	 386
Endpoints Overview.....	386
Creating an Endpoint.....	386
Deleting an Endpoint.....	388
Configuring a Remote Machine to Run Self-Service Python Scripts	389
 Backup and Restore in Self-Service.....	 391
Self-Service Data Backup and Restore.....	391
Backing up Self-Service Data.....	393
Restoring Self-Service Data.....	395
Flag Options for Backup.....	397
Backing Up and Restoring Policy Engine Database.....	399
 Python 2 Deprecation.....	 401

EScripts Update in Blueprints, Runbooks, Tasks, and Marketplace Items.....	402
Updating eScripts When You Have a Development Instance of Self-Service VM.....	403
Updating eScripts When You Do Not Have a Development Instance of Self-Service VM....	405
Updating eScripts When You Have a Development Instance of Self-Service in Prism Central.....	406
Updating eScripts When You Have a Production Instance of Self-Service in Prism Central.....	408
Updating eScripts in Applications.....	410
Updating Custom Action Scripts of a Multi-VM Application - Example.....	411
Self-Service Scripts.....	413
Sample Scripts for Installing and Uninstalling Services.....	413
Sample Scripts to Configure Non-Managed AHV Network.....	413
Supported eScript Modules and Functions.....	416
EScript Sample Script.....	421
JWT Usage Sample Script.....	421
Sample Powershell Script.....	422
Sample Auto Logon and First Logon Scripts.....	422
Sample Guest Customization Scripts for VMware and GCP Services.....	422
Self-Service Blueprints Public Repository.....	423
Seeding Scripts to the Self-Service Task Library.....	423
Licensing and Upgrades in Self-Service.....	424
Self-Service Licensing.....	424
Self-Service Upgrades.....	424
Life Cycle Manager.....	424
Self-Service VM Upgrades.....	427
Upgrading Calm VM from Version 3.5.2 or 3.6 to Version 3.6.2.....	428
Upgrading Calm VM from Version 3.5.2 or 3.6 to Version 3.6.2 at a Dark Site.....	429
Upgrading Self-Service VM from Version 3.6.2 to Version 3.7.2.1.....	432
Upgrading Self-Service VM from Version 3.7 to Version 3.7.2.1.....	433
Upgrading Self-Service VM from Version 3.7.1, 3.7.2, 3.7.2.1, or 3.7.2.2 to Version 3.8.1....	435
Upgrading Self-Service VM from Version 3.8 to Version 3.8.1.....	437
Upgrade the Policy Engine VM on an Upgraded Self-Service VM.....	439
Additional Information.....	443
Syslog for Calm and Epsilon Services.....	443
Credential Security Support Provider.....	444
Enabling CredSSP.....	444
Generating SSH Key on a Linux VM.....	444
Generating SSH Key on a Windows VM.....	445
Integrated Linux Based PowerShell Gateway Errors.....	445
Localization.....	446
Copyright.....	447

INTRODUCTION TO NUTANIX CLOUD MANAGER SELF-SERVICE

Nutanix Cloud Manager (NCM) Self-Service allows you to seamlessly select, provision, and manage your business apps across your infrastructure for both the private and public clouds. Self-Service provides app automation, life-cycle management, monitoring, and remediation to manage your heterogeneous infrastructure, for example, VMs or bare-metal servers.

Self-Service supports multiple platforms so that you can use the single self-service and automation interface to manage all your infrastructure. Self-Service provides an interactive and user-friendly graphical user interface (GUI) to manage your infrastructure.

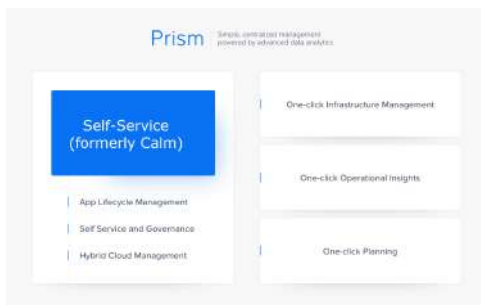


Figure 1: Self-Service Model

Self-Service Key Capabilities

Self-Service is a multi-cloud app management framework that offers the following key benefits:

- **IT Agility and Human Error Elimination**

Self-Service simplifies the setup and management of custom enterprise apps by incorporating all important elements, such as the relevant VMs, configurations, and related binaries into an easy-to-use blueprint. These blueprints make the deployment and life-cycle management of common apps repeatable and help infrastructure teams eliminate extensive and complex routine app management.

- **Unified Multi-Cloud Orchestration**

Self-Service unifies the management of all your clouds into a single-pane-of-glass, removing the need to switch between portals. Self-Service automates the provisioning of multi-cloud architectures, scaling both multi-tiered and distributed apps across different cloud environments, including AWS, GCP, Azure, and VMware (on both Nutanix and non-Nutanix platforms).

- **Nutanix Marketplace and App Management**

The marketplace offers preconfigured app blueprints that infrastructure teams can instantly consume to provision apps. The marketplace also provides the option to publish shareable runbooks. A runbook is a collection of tasks that are run sequentially at different endpoints. Infrastructure teams can define endpoints and use runbooks to automate routine tasks and procedures that span across multiple apps without the involvement of a blueprint or an app.

App Management (My Apps) empowers different groups in the organization to manage their own apps, giving app owners and developers an attractive alternative to public cloud services.

Nutanix Marketplace and App management are now accessed through Prism Central Admin Center. For more information, see [My Apps](#) and [Marketplace](#) in the *Prism Central Admin Center Guide*.

- **Cost Governance**

With native integration into Cost Governance (formerly Beam), Self-Service also shows the overall utilization and true cost of consumption to help you make deployment decisions with confidence.

- **Application Development and Modernization**

Combined with Nutanix Kubernetes Engine or your choice of certified Kubernetes, Self-Service provides the tools required to modernize apps without losing control of policy. Additionally, Self-Service natively integrates with Jenkins to empower CI/CD pipelines with automatic infrastructure provisioning or upgrades for all apps.

- **Self-Service DSL - Infrastructure-as-Code (IaC)**

Self-Service DSL describes a simpler Python3-based Domain Specific Language (DSL) for writing Self-Service blueprints. DSL offers all the richness of the Self-Service user interface along with additional benefits of being human readable and version controllable code that can handle even the most complex app scenario. DSL can be also used to operate Self-Service from a CLI. For more information, see [Self-Service DSL Overview](#) on page 37.

Where to Access App Management Capabilities

With the new Prism Central experience (pc.2023.1.0.1 and later), the capabilities of Prism Central and Self-Service are organized to offer easy product discovery and seamless cross product transition from a single control plane.

How these Capabilities are Distributed

With the new Prism Central experience, you can now access Nutanix Marketplace and app management (My Apps) through the Prism Central Admin Center. Prism Central Admin Center provides a common workspace for different users and administrators to manage projects, deploy different apps, and manage the apps at one place. For more information, see the [Prism Central Admin Center Guide](#).

You, however, use Self-Service to carry out all configuration aspects related to app management, such as:

- Configuring provider accounts - Nutanix, VMware, Azure, GCP, and AWS.
- Managing credential providers.
- Blueprint configuration, approval, and publishing to the Marketplace.
- Managing quota policies and snapshot policies.
- Managing availability of apps in the Marketplace.

SELF-SERVICE PREREQUISITES AND DEPLOYMENT

The topics in this section cover aspects such as Self-Service deployment benchmark guidelines, port information, and deployment prerequisites.

Self-Service Benchmarks

Nutanix certifies the following benchmarks for single-node deployment profiles (non-scale-out) and three-node deployment profiles (scale-out). Each benchmark contains scale numbers across different entities of Self-Service. Because the scaling properties of these entities often depend on each other, changes to one entity might affect the scale of other entities. For example, if your deployment has smaller number of VMs than the benchmarked number, you can have a higher number of blueprints, projects, runbooks, and so on.

Use these benchmark guidelines as a good starting point for your Self-Service installation. You might have to allocate more resources over time as your infrastructure grows.

Self-Service Single-Node Profile

The following table shows the Self-Service benchmarks for a single-node Prism Central profile.

Table 1: Single-Node Profile Benchmarks

Prism Central size	Prism Central configuration	Number of VMs	Number of single-VM blueprints	Number of single-VM apps	Number of projects	Number of runbooks
Small (1 node)	6 vCPUs and 30 GB of memory for each node. 8 vCPUs and 32 GB of memory for each node (on a new Prism Central 2022.9 or later deployment).	3000	600	3000	50	250

Prism Central size	Prism Central configuration	Number of VMs	Number of single-VM blueprints	Number of single-VM apps	Number of projects	Number of runbooks
Large (1 node)	10 vCPUs and 52 GB of memory for each node. 14 vCPUs and 54 GB of memory for each node (on a new Prism Central 2022.9 or later deployment).	12500	2500	12500	250	500

Self-Service Three-Node (Scale-Out) Profile

The following table shows the Self-Service benchmarks for a three-node Prism Central profile. If high-availability is preferred, it is recommended to use the scale-out deployment.

Table 2: Scale-Out Profile Benchmarks

Prism Central size	Prism Central configuration	Number of VMs	Number of single-VM blueprints	Number of single-VM apps	Number of projects	Number of runbooks
Small (3 nodes, scale out)	6 vCPUs and 30 GB of memory for each node. 8 vCPUs and 32 GB of memory for each node (on a new Prism Central 2022.9 or later deployment).	6000	1200	6000	100	500

Prism Central size	Prism Central configuration	Number of VMs	Number of single-VM blueprints	Number of single-VM apps	Number of projects	Number of runbooks
Large (3 nodes, scale out)	10 vCPUs and 52 GB of memory for each node. 14 vCPUs and 54 GB of memory for each node (on a new Prism Central 2022.9 or later deployment).	25000	5000	25000	500	1000

Benchmark Considerations

The following considerations are applicable for both Self-Service single-node and three-node (scale-out) profiles:

- When you enable Self-Service on a new deployment of Prism Central version 2022.9 or later:
 - An additional 2 vCPUs and 4 GB of memory per node is added to the Prism Central small deployment profile.
 - An additional 4 vCPUs and 8 GB of memory is added to the Prism Central large deployment profile.

Note: No additional vCPUs are added to the Prism Central deployment profile if you upgrade from Prism Central version 2022.6 or earlier.

- The listed app and blueprint numbers include both running and deleted apps and blueprints. Data related to deleted apps and blueprints is cleaned up after 3 months.
- All the listed VM numbers are for the VMs that are managed by Self-Service and not Prism Central overall.
- These performance tests are done by using single-service blueprints and apps. Results might be lower when blueprints with multiple services are deployed.
- Self-Service automatically archives the logs every 3 months to clean up the database and to free up the resources. You can also increase the memory (RAM) of your deployed Prism Central VM to increase the Self-Service capacity.

Self-Service Throughput

The maximum throughput on a large three-node (scale-out) deployment profile is 400 VMs per hour.

Note: For customized higher capacity Self-Service configurations, work with your Nutanix sales representative.

Port Information in Self-Service

For a list of required Self-Service ports, see [Port Reference](#). The Port Reference section provides detailed port information for Nutanix products and services, including port sources and destinations, service descriptions, directionality, and protocol requirements.

Self-Service Deployment

Self-Service is integrated into Prism Central and does not require you to deploy any additional VMs. To start using Self-Service, you only have to deploy Self-Service from Prism Central.

As a Prism Central Administrator, you can deploy Self-Service from the Nutanix Marketplace. You can access Nutanix Marketplace in the Prism Central Admin Center. For more information on the deployment, see the [Deploying Self-Service](#) section in the *Prism Central Admin Center Guide*.

After deployment, you can select **Self-Service** in the Prism Central Application Switcher to explore and use Self-Service. For more information on Application Switcher, see [Application Switcher Function](#) in the *Prism Central Infrastructure Guide*.

Prerequisites to Deploy Self-Service

Before you deploy the Self-Service Applications from the Marketplace, ensure that you have met the following prerequisites.

- The Prism Central version is compatible with Self-Service.

You can go to the [Software Product Interoperability](#) page to verify the compatible versions of Self-Service and Prism Central.

- The Prism Central in which Self-Service is running is registered with the same cluster.

Note:

- Self-Service is supported only on AHV and ESXi hypervisors on a Nutanix cluster.
- Self-Service is not supported on Hyper-V cluster.
- Self-Service does not support vSphere Essentials edition because hot-pluggable virtual hardware is not supported by vSphere Essentials edition.

- A unique data service IP address is configured in the Prism web console cluster that is running on Prism Central. For more information on configuring data service IP address, see the [Modifying Cluster Details](#) in the *Prism Web Console Guide*.

Note: Do not change the data service IP address after Self-Service enablement.

- A minimum allocation of 4 GB of memory for a small Prism Central and 8 GB of memory for large Prism Central. For more information, see [Self-Service Benchmarks](#) on page 10.
- All the required ports are open to communicate between a Prism web console and Prism Central. For more information on ports, see [Port Reference](#).
- The DNS server is reachable from Prism Central. Unreachable DNS server from Prism Central can cause slow deployment of Self-Service.

GETTING STARTED WITH SELF-SERVICE

This section covers pre-configuration requirements of Self-Service, an overview of different components of Self-Service, and the Role-Based Access Control in Self-Service.

Pre-configuration for Using Self-Service

You must configure the following components before you start using Self-Service.

- Image configuration for VM: To use Nutanix as your provider, you must add and configure the image for your VMs. Images are provider-specific. You can upload images to Prism Central or Prism web console and use those images when you configure your blueprints. For more information, see [Image Management](#) in the *Prism Central Infrastructure Guide*.
- SSH key (optional): Generate SSH keys so that you can use them for authentication when you configure or launch your blueprints. For more information, see [Generating SSH Key on a Linux VM](#) on page 444 or [Generating SSH Key on a Windows VM](#) on page 445.
- Directory Service: Configure Lightweight Directory Access Protocol (LDAP) or OpenLDAP to authenticate users or group of users to use Self-Service. For more information, see [Configuring Authentication](#) in the *Security Guide*.

Checking Self-Service Version

You can check the version of your Self-Service instance from the Self-Service user interface.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.

- Click **About Self-Service** in the navigation bar.
The **About Nutanix Self-Service** page appears displaying the version number of Self-Service. For example:

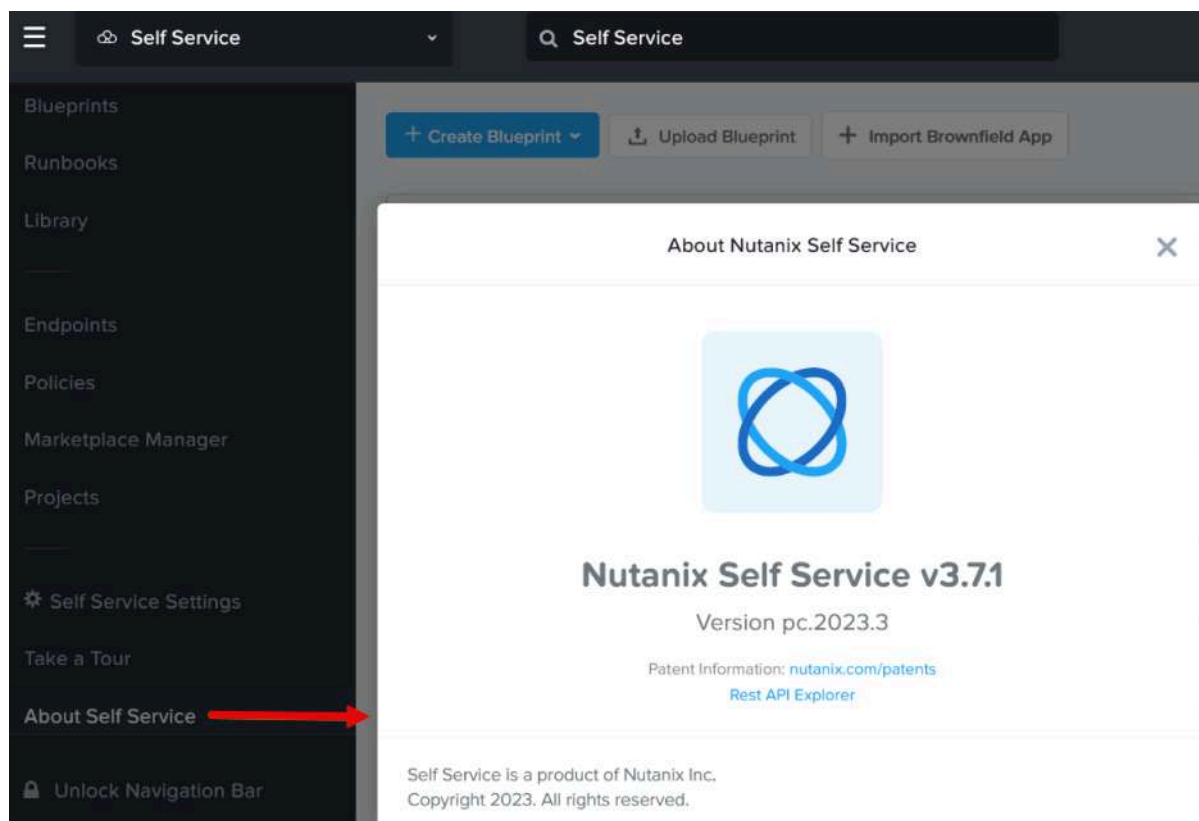


Figure 2: Version Number

Self-Service Overview

The following table lists the different components in Self-Service and their usage:

Table 3: Self-Service Entities

Component	Usage
Applications	To view and manage apps that are launched from blueprints. You can access this capability in the Prism Central Admin Center. For more information, see My Apps in the <i>Prism Central Admin Center Guide</i> .
Marketplace	To instantly consume app blueprints to provision apps. You can access this capability in the Prism Central Admin Center. For more information, see Marketplace in the <i>Prism Central Admin Center Guide</i> .

Component	Usage
Blueprints	To create, configure, publish, and launch single-VM or multi-VM blueprints. For more information, see Self-Service Blueprints Overview on page 87.
Runbooks	To automate routine tasks and procedures that span across multiple apps without involving any blueprints or apps. For more information, see Runbooks Overview on page 265.
Library	To create and use variable types and tasks. You use variables and tasks while configuring a blueprint. See Library Overview on page 259.
Endpoints	To create and manage target resources where the tasks defined in a runbook or in a blueprint can run. For more information, see Endpoints Overview on page 386.
Policies	<p>To schedule app actions and runbook executions. For more information, see Scheduler Overview on page 238.</p> <p>To configure approval policy to manage infrastructure resources. See Approval Policy Overview on page 242.</p>
Marketplace Manager	To manage approval and publishing of app blueprints. For more information, see Marketplace Manager Overview on page 231.
Projects	To create users or groups and assign permissions. You can access this capability in the Prism Central Admin Center. For more information, see Project Management in the <i>Prism Central Admin Center Guide</i> .
Self-Service Settings	<p>To manage general settings, such as Showback and Marketplace Branding. For more information, see Self-Service Settings on page 42.</p> <p>To configure and manage provider accounts. For more information, see Provider Account Settings in Self-Service on page 52.</p> <p>To configure and manage credential provider. For more information, see Configuring a Credential Provider on page 49.</p> <p>To configure and manage approvals and quota policies. For more information, see Policy Engine and Quotas Overview on page 44.</p>

Exploring Self-Service

You can use the following procedure to explore Self-Service user interface and get an overview of the Self-Service components.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Take a Tour** in the navigation bar.
4. Click **Explore Self-Service**.
5. Do one of the following.
 - a. To navigate through the Self-Service components, click the right or left arrow.
 - b. To skip the tour, click **Skip tour**.

Accessing Self-Service REST API Explorer

Use the following procedure to access the Self-Service REST API explorer console from the Self-Service user interface.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **About Self-Service** in the navigation bar.

The **About Nutanix Self-Service** page appears displaying the version number of Self-Service. For example:

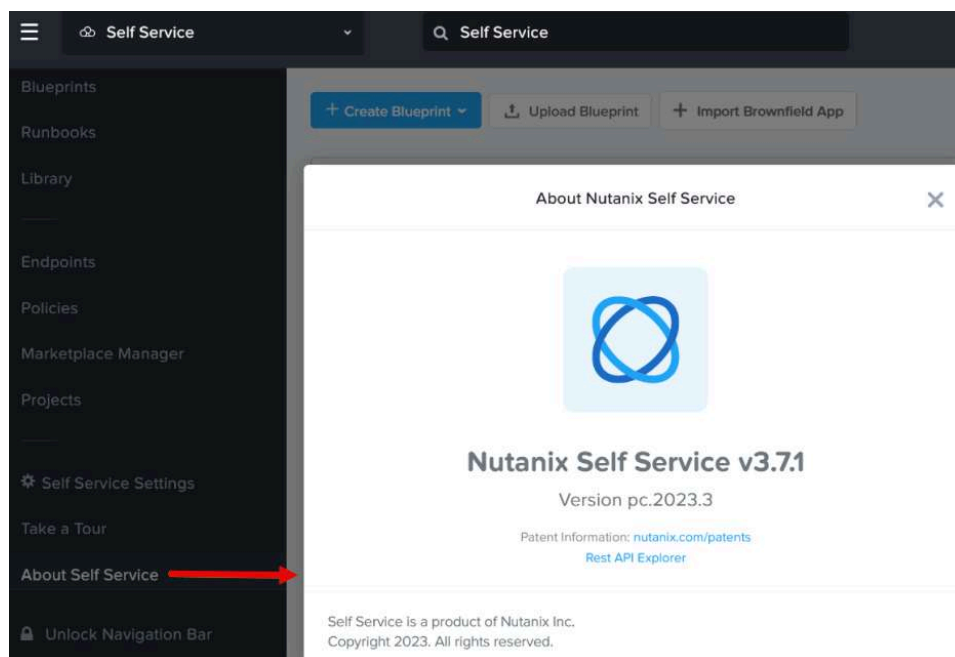


Figure 3: Version Number

4. Click the **Rest API Explorer** link.

The Self-Service REST API explorer interface appears.

Self-Service Role-Based Access Control

Self-Service manages the role-based access control using projects. Projects are logical groupings of user roles, accounts, VM templates, and credentials that are used to manage and launch blueprints and apps within your organization. For more information, see [Projects Overview](#) on page 78.

Users or groups are allowed to view, launch, or manage apps based on the roles that are assigned within the projects. Self-Service has the following roles for users or groups:

- **Project Admin**

Project admins have full control of the project. They can perform reporting and user management, create blueprints, launch blueprints, and run actions on the apps.

- **Developer**

Developers can create blueprints, launch blueprints, and run actions on the apps. They are, however, not allowed to perform reporting and user management.

- **Consumer**

Consumers can launch new blueprints from the marketplace and run actions on the apps. They are, however, not allowed to create their own blueprints.

- **Operator**

Operators have minimum access and are allowed only to run actions against existing apps. They are not allowed to launch new blueprints or edit any existing blueprints.

Note: A Prism Admin is a super user within Self-Service and within the rest of Prism Central who has full access to all the features and functionalities of Self-Service.

The following table details the operations for different components and the users who have permissions to perform those operations.

Table 4: Roles and Responsibilities Matrix

Component	Operation	Prism Admin	Project Admin	Developer	Consumer	Operator
Marketplace	Enable	Yes	No	No	No	No
	Manage	Yes	No	No	No	No
	App publishing request	Yes	Yes	Yes	No	No
	Send app publishing request to the Administrator	Yes	Yes	No	No	No
	Clone and edit app blueprint	Yes	Yes	Yes	No	No

Component	Operation	Prism Admin	Project Admin	Developer	Consumer	Operator
Blueprint	Create, update, delete, and duplicate	Yes	Yes	Yes	No	No
	Read-only	Yes	Yes	Yes	Yes	No
	Launch	Yes	Yes	Yes	Yes	No
Applications	Complete app summary	Yes	Yes	Yes	Yes	Yes
	Run application actions	Yes	Yes	Yes	Yes	Yes
	App debug mode	Yes	Yes	Yes	Yes	Yes
	Action edit (applicable only for single-VM apps)	Yes	Yes	Yes	No	No
	Create App (brownfield import)	Yes	Yes	Yes	No	No
	Delete app	Yes	Yes	Yes	Yes	No
Settings	CRUD	Yes	No	No	No	No
Task Library	View	Yes	Yes	Yes	Yes	Yes
	Create and update	Yes	Yes	Yes	No	No
	Delete	Yes	No	No	No	No
	Sharing with projects	Yes	No	No	No	No
Projects	Add project	Yes	No	No	No	No
	Update project	Yes	Yes	No	No	No
	Add VMs to projects	Yes	No	No	No	No
	Custom roles	No	No	No	No	No
Users	Add users to the system and change roles	Yes	No	No	No	No

Component	Operation	Prism Admin	Project Admin	Developer	Consumer	Operator
	Add and remove users to or from a project	Yes	Yes	No	No	No
	Change user roles in a project	Yes	Yes	No	No	No
	Create Administrator	Yes	No	No	No	No
	Create Project Administrator	Yes	Yes	No	No	No
Runbooks	Create and update	Yes	Yes	Yes	No	No
	View	Yes	Yes	Yes	Yes	Yes
	Delete	Yes	Yes	Yes	No	No
	Execute	Yes	Yes	Yes	Yes	Yes
Endpoints	Create and update	Yes	Yes	Yes	No	No
	View	Yes	Yes	Yes	Yes	Yes
	Delete	Yes	Yes	Yes	No	No
Scheduler	Create, delete, and clone jobs	Yes	Yes	Yes	Yes	No
	Read job and view execution status	Yes	Yes	Yes	Yes	Yes
	Update job name, schedule, executable, and app action	Yes	Yes	Yes	Yes	No
	Edit operations on a blueprint launch	Yes	Yes	Yes	Yes	No
	Edit operations on runbook executions	Yes	Yes	Yes	Yes	No

Component	Operation	Prism Admin	Project Admin	Developer	Consumer	Operator
	Edit operations on app actions	Yes	Yes	Yes	Yes	No
	Edit operations on Marketplace launch	Yes	Yes	Yes	Yes	No

Note: Scheduler does not support custom roles in this release.

SELF-SERVICE VM DEPLOYMENT

Self-Service VM (formerly Calm VM) is a standalone VM that you can deploy on AHV hypervisor and leverage Self-Service functionality without the Nutanix infrastructure.

Note: As of this release, Nutanix supports Self-Service VM on AOS only on AHV and does not support on ESXi.

You can deploy Self-Service using the image at the [Nutanix Support Portal - Downloads page](#) and manage your apps across a variety of cloud platforms. Self-Service VM deployment eliminates the need of the complete Nutanix infrastructure to use Self-Service features.

Note:

- Self-Service VM currently supports Self-Service version 3.8.1.
- Self-Service VM supports scale-out deployment. For more information, see [Setting up Scale-Out Self-Service VM](#) on page 28.
- For a list of required ports for Self-Service VM with and without VPC, see [Port Reference for Self-Service VM with VPC](#) and [Port Reference for Self-Service VM without VPC](#).
- Due to Microservices Infrastructure (MSP) limitations, only asynchronous disaster recovery (Async DR) of Self-Service VM 3.8.1 is supported.

For information on Self-Service VM deployment, see [Deploying Self-Service VM on AHV](#) on page 22 and [Setting Up Self-Service VM](#) on page 27.

Deploying Self-Service VM on AHV

This section describes the steps to deploy Self-Service VM (formerly Calm VM) version 3.8.1 on AHV.

Before you begin

Ensure that you have downloaded the version 3.8.1 OVA file to your computer from the [Nutanix Support Portal - Downloads page](#).

Procedure

1. Log in to your Prism Central instance.
2. Select **Infrastructure** in the Application Switcher.
3. Navigate to **Compute & Storage > OVAs**.
4. Click **Upload OVA**.
The Upload OVA page appears.

5. Under OVA Source, select **OVA File** and do the following.

The screenshot shows the 'OVA Source' section of a web form. It has two radio buttons: 'OVA File' (selected) and 'URL'. Below is a 'Select AHV Cluster' dropdown menu. A note states: 'You can upload the OVA to multiple clusters together using URL upload'. There is a 'Name' text input field. A 'Checksum' section has a dropdown set to 'Optional' and a 'SHA-256' button. The 'Choose OVA File' section shows 'No file selected' and a 'Select File' link. At the bottom are 'Cancel' and 'Upload' buttons.

Figure 4: Upload OVA

- a. Provide a name in the **Name** field.
 - b. Under Choose OVA File, click the **Select File** option to navigate to the location of the OVA file and select it.
 - c. Click **Upload**.
6. On the OVAs page, select the uploaded OVA from the list.
7. From the **Actions** dropdown menu, select **Deploy as VM**.

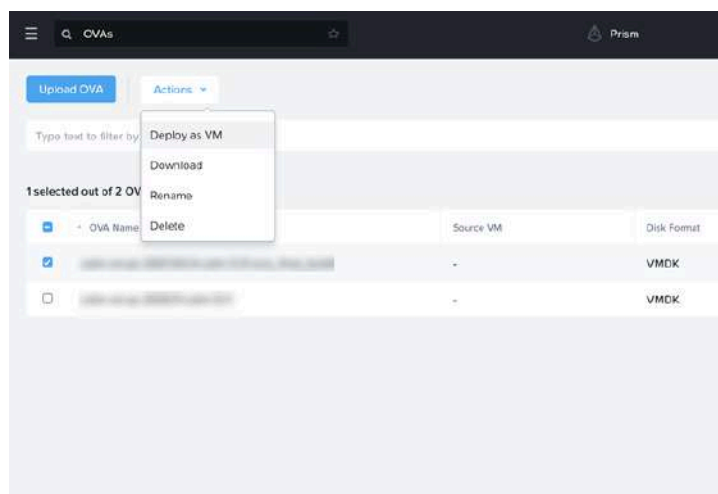


Figure 5: Deploy VM

8. On the Deploy as VM page, do the following:

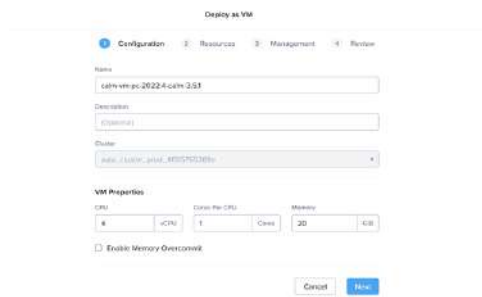


Figure 6: Deploy as VM - Configuration

- a. On the **Configuration** tab, specify the values for **CPU**, **Cores Per CPU**, and **Memory** in the VM Properties section.
Provide appropriate values for the large or small configuration of your Self-Service VM. The minimum resources required are 8 vCPUs and 32 GB of memory for each node.
The CPU and Memory requirements of the Self-Service VM Deployment is equivalent to the Self-Service single-node profile. For detailed benchmark values, see [Self-Service Benchmarks](#) on page 10.
- b. Click **Next**.
- c. On the **Resources** tab, associate the required subnets for your Self-Service VM instance to configure the networks.
- d. Ensure that the **Legacy BIOS Mode** is selected in the Boot Configuration section.
- e. Click **Next**.
- f. Click **Next**.
- g. Click **Create VM** to start the deployment of the Self-Service VM instance.

9. Navigate to **Compute & Storage > VMs**.

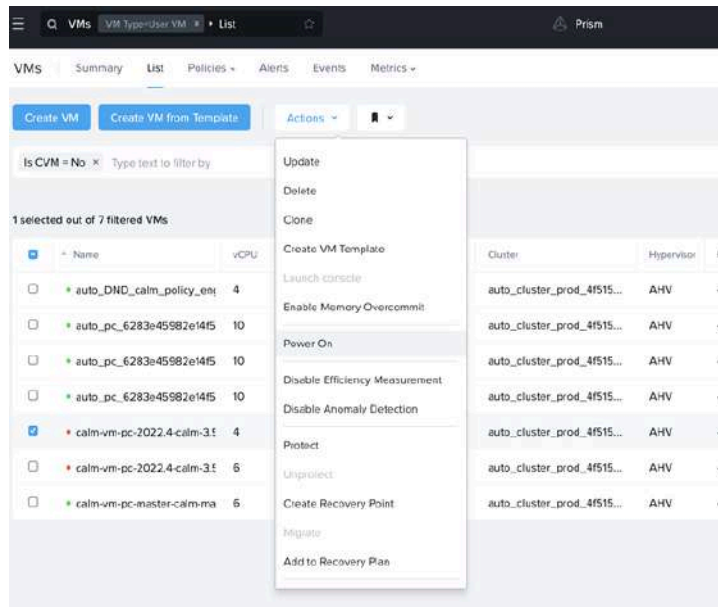


Figure 7: VM Power On

- Select the Self-Service VM instance that you deployed.
- Click **Actions** and then click **Power On** to power on the Self-Service VM instance.
- Wait for the Self-Service services to be up and running.

What to do next

Set up Self-Service VM. For more information, see [Setting Up Self-Service VM](#) on page 27.

Launching Self-Service VM with a Static IP Address on AHV

This section describes the steps to launch Self-Service VM (formerly Calm VM) with a static IP address on AHV.

About this task

Note: For scale-out Self-Service VM, create three instances of Self-Service VM with static IP address using this procedure. For more information on scale-out Self-Service VM, see [Setting up Scale-Out Self-Service VM](#) on page 28.

Before you begin

Ensure that you have configured a static subnet from which you can use the static IP for Self-Service VM.

Procedure

- Log in to your Prism Central instance.
- Select **Infrastructure** in the Application Switcher.
- Navigate to **Compute & Storage > OVAs**.

4. Click **Upload OVA**.
The Upload OVA page appears.
5. Under OVA Source, select **URL** and do the following.

The screenshot shows the 'Upload OVA' page. At the top, under 'OVA Source', the 'OVA File' radio button is selected, while the 'URL' radio button is unselected. Below this is a 'Select AHV Cluster' dropdown menu. A note states: 'You can upload the OVA to multiple clusters together using URL upload'. There is a 'Name' text input field. Below that is a 'Checksum' section with a dropdown menu currently set to 'Optional' and a 'SHA-256' button. The 'Choose OVA File' section shows 'No file selected' and a 'Select File' link. At the bottom right are 'Cancel' and 'Upload' buttons.

Figure 8: Upload OVA

- a. Provide a name in the **Name** field.
 - b. Provide the URL of the OVA file of the Self-Service VM in the **OVA URL** field.
 - c. Click **Upload**.
6. On the OVAs page, select the uploaded OVA from the list.
 7. From the **Actions** dropdown menu, select **Deploy as VM**.

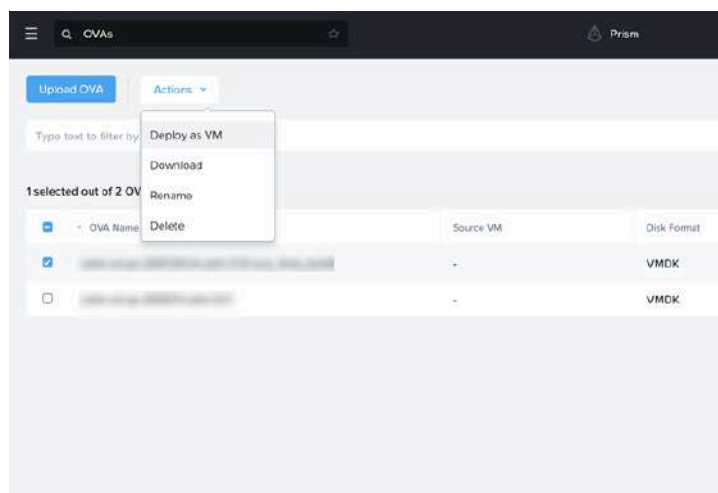


Figure 9: Deploy VM

8. On the Deploy as VM page, do the following:
 - a. Under the VM Properties section, specify the values for **CPU**, **Cores Per CPU**, and **Memory**.
The CPU and Memory requirements of the Self-Service VM Deployment is equivalent to the Self-Service single-node profile. For the benchmark values, see [Self-Service Benchmarks](#) on page 10.
 - b. Click **Next**.
 - c. To configure the networks, select the static VLAN that you want to place your Self-Service VM in.
 - d. From the **Assignment Type** dropdown menu, select **Assign Static IP**.
The **Assignment Type** dropdown menu appears when you select a managed static VLAN (VLAN managed by IPAM).
 - e. In the **IP Address** field, specify the static IP address you want to assign to your Self-Service VM instance.
 - f. Click **Save**.
 - g. Click **Next**.
 - h. Click **Create VM** to start the deployment of the Self-Service VM instance.
9. Navigate to **Compute & Storage > VMs** and do the following:
 - a. Select the Self-Service VM instance that you deployed.
 - b. Click **Actions** and then click **Power On** to power on the Self-Service VM instance.
 - c. Wait for the Self-Service services to be up and running.

Setting Up Self-Service VM

Perform the following steps to set up Self-Service VM (formerly Calm VM) version 3.8.1 (pc.2024.2).

Before you begin

Ensure that you followed the steps provided in the [Deploying Self-Service VM on AHV](#) on page 22 topic to upload the OVA file, deploy, and power on the VM.

Procedure

1. Wait until the Self-Service VM is up and the Nutalm, Epsilon, and Domain Manager containers are in the healthy state.

Note: You cannot log in using the console or any remote access until all services (Nutalm, Epsilon, and Domain Manager) are up.

2. Log on to the Self-Service VM with SSH.

- Username: nutanix
- Password: nutanix/4u

If you want to reset the admin password, then use the `ncli user reset-password` command.

```
nutanix@pcvm$ ncli user reset-password user-name=xxxxx password=yyyyy
```

where `user-name=xxxxx` is the name of the user whose password is to be reset and `password=yyyyy` is the new password.

3. If your Self-Service VM configuration is Large, use the following commands to increase the IDF resident limit to 21 GB.

Note: The CPU and Memory requirements of the Self-Service VM deployment is equivalent to the Self-Service single-node profile. To identify the size of your Self-Service VM deployment, see [Self-Service Benchmarks](#) on page 10.

```
nutanix@pcvm$ allssh genesis stop insights_server
```

```
nutanix@pcvm$ allssh echo --insights_rss_share_mb=21000 >> ~/config/insights_server.gflags
```

```
nutanix@pcvm$ allssh echo --insights_multiclustercgroup_limit_factor_pct=500 >> ~/config/genesis.gflags
```

```
nutanix@pcvm$ allssh genesis restart
```

```
nutanix@pcvm$ cluster start
```

4. Enable MSP on Self-Service VM. For more information, see [Enabling Microservices Infrastructure on Self-Service VM](#) on page 30.

Enabling MSP requires downloading packages from the Nutanix website. For dark sites, a dark site bundle server has to be configured.

What to do next

To manage and administer your apps, use the Self-Service VM IP address and the following default credentials to log in to the Self-Service VM user interface for the first time:

- Username: admin
- Password: Nutanix/4u

You can change the default credentials after you log in to the Self-Service VM user interface.

Setting up Scale-Out Self-Service VM

Use the following procedure to set up scale-out version of Self-Service VM (formerly Calm VM).

About this task

Note: Scaling out of an existing single-node Self-Service VM is possible by performing the backup and restore. For more information on the backup and restore, see [Self-Service Data Backup and Restore](#) on page 391.

Before you begin

Ensure that:

- You downloaded the Self-Service VM OVA file from the [Nutanix Support Portal - Downloads page](#).
- You uploaded the Self-Service VM OVA file on AHV using the Prism Central User Interface. For more information, see [Deploying Self-Service VM on AHV](#) on page 22.
- You have taken a backup of Self-Service data in case you are planning to extend an existing Self-Service VM. For more information, see [Self-Service Data Backup and Restore](#) on page 391.

Procedure

1. Create three Self-Service VMs using the templates you uploaded.

2. Do one of the following:

- » To assign static IP addresses to the VMs, see [Launching Self-Service VM with a Static IP Address on AHV](#) on page 25.
- » If DHCP is enabled, continue to step 3.

3. SSH to the three Self-Service VMs and wait until all the cluster services, including Calm and Epsilon, are up and running.

4. Run the following commands on all three VMs.

```
nutanix@pcvm$ cluster stop
nutanix@pcvm$ cluster destroy
```

Note: Run these commands only after taking a backup of the Self-Service data in case you are extending an existing Self-Service VM.

In addition, run the following command to prevent any duplicate entities of the Infrastructure and Self-Service apps in the Self-Service VM.

```
nutanix@pcvm$ sudo rm -rf /home/docker/epsilon/elasticsearch/*
```

5. Run the cluster create command on one of the three VMs.

- » To create a simple cluster, run the following command:

```
nutanix@pcvm$ #cluster --cluster_function_list="multicluster" -s <ip1>,<ip2>,<ip3>
create
```

For example:

```
nutanix@pcvm$ cluster --cluster_function_list="multicluster" -s
10.46.141.71,10.46.138.20,10.46.138.26 create
```

- » To create an advanced cluster with cluster name and virtual IP, run the following command:

```
nutanix@pcvm$ cluster --cluster_function_list="multicluster" --cluster_name
"<Cluster Name>" -s <ip1>,<ip2>,<ip3> --cluster_external_ip=<vip> create
```

For example:

```
nutanix@pcvm$ cluster --cluster_function_list="multicluster" --cluster_name "Demo"
-s 10.46.141.71,10.46.138.20,10.46.138.26 --cluster_external_ip=10.46.141.70 --
dns_servers 10.40.64.15,10.40.64.16 create
```

6. Run the following command on one of the three VMs.

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ python create_nucalm_zk_node.py
```

```
nutanix@pcvm$ cd /home/nutanix/bin/ && python create_calm_dep_zk_node.py
```

```
nutanix@pcvm$ python enable_calm.py
```

7. Run the following command to verify Epsilon and Self-Service services status:

```
nutanix@pcvm$ cluster status
```

8. Enable MSP on Self-Service VMs. For more information, see [Enabling Microservices Infrastructure on Self-Service VM](#) on page 30.

What to do next

- Enable policy engine for Self-Service VM. For more information, see [Enabling Policy Engine for Self-Service VM](#) on page 32.
- To manage and administer your apps, use the Self-Service VM IP address and the following default credentials to log in to the Self-Service VM user interface for the first time:
 - Username: admin
 - Password: Nutanix/4u

You can change the default credentials after you log in to the Self-Service VM user interface.

Enabling Microservices Infrastructure on Self-Service VM

Microservices Infrastructure (sometimes referred to as MSP) provides a common framework and services to deploy the container-based services associated with Prism Central based components. It deploys services such as Identity and Access Management (IAM), Load Balancing (LB), and Virtual Private Networking (VPN).

Before you begin

- Ensure that you have configured virtual IP address and iSCSI data services IP (DSIP) address on the Prism Element that hosts Prism Central. For additional prerequisites and considerations, see [Microservices Infrastructure Prerequisites and Considerations](#) in the *Prism Central Infrastructure Guide*.
- Ensure that you configured a minimum of one name server and one NTP server in both the host Prism Element cluster and Calm VM. For more information on the name server and NTP server configurations, see the [Admin Center Settings Options](#) section in the *Prism Central Admin Center Guide*.

Note: For dark sites:

Enabling MSP requires downloading packages from the Nutanix website. For dark sites, you must add an LCM dark site bundle server to the LCM configuration in the Calm VM before enabling MSP.

Procedure

Perform the following steps to enable MSP on Self-Service VM (formerly Calm VM).

1. Run the following commands to download the `enable_cmsp.py` and `update_factory_config_json.py` scripts into the `/home/nutanix/bin` directory.

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
enable_cmsp.py
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
update_factory_config_json.py
```

2. Depending on your Self-Service VM configuration, run the followings commands:

For single-node Self-Service VM:

```
nutanix@pcvm$ cd /home/nutanix/bin  
nutanix@pcvm$ python enable_cmsp.py <PEIP> '<PE_UI_Username>' '<PE_UI_Password>'  
'<CalmVM_UI_Username>' '<CalmVM_UI_Password>' <Calm VM1_Name>
```

For scale-out Self-Service VM, run the following commands on the node on which you downloaded the `enable_cmsp.py` script:

```
nutanix@pcvm$ cd /home/nutanix/bin  
nutanix@pcvm$ python enable_cmsp.py <PEIP> '<PE_UI_Username>' '<PE_UI_Password>'  
'<CalmVM_UI_Username>' '<CalmVM_UI_Password>' <Calm VM1_Name> <Calm VM2_Name> <Calm  
VM3_Name>
```

In these commands:

- `<PEIP>` is the IP address of the host Prism Element cluster.
- `<PE_UI_Username>` is the username of the host Prism Element cluster.
- `<PE_UI_Password>` is the password of the host Prism Element cluster.
- `<CalmVM_UI_Username>` is the Self-Service VM username.
- `<CalmVM_UI_Password>` is the Self-Service VM password.
- `<Calm VM1_Name>`, `<Calm VM2_Name>`, and `<Calm VM3_Name>` are names of the Self-Service VM instances that can be found on the VM page of the Prism Central that you used to launch them.

Note: Do not use PC IP instead of PE IP for the enablement. For more information, see [KB-15305](#).

3. SSH to Self-Service VM and run `mspctl task list` to view the IAM enablement status. Wait until all the tasks move to the SUCCEEDED state.

Note: After MSP enablement, you must wait for a few minutes for the MSP to be ready and then run the `mspctl task list` command. Self-Service might display the "Error listing tasks" or "failed to get task list" errors if you run the `mspctl task list` immediately after triggering the MSP enablement.

4. After enabling MSP, if you experience multiple restarts in the Redis, PostgreSQL and CAPE pods, you can increase the ReadinessProbe timeouts to reduce the restarts. To do that:

- a. Run the following commands to download the `increase_readiness_probe_timeouts.sh` script from the [Downloads](#) page into the `/home/nutanix` directory of the Self-Service VM.

The following commands do not work at the dark site. Therefore for the dark site, create the `increase_readiness_probe_timeouts.sh` file in the `/home/nutanix` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/nutanix  
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
increase_readiness_probe_timeouts.sh
```

- b. Run the following commands:

```
nutanix@pcvm$ chmod +x increase_readiness_probe_timeouts.sh  
nutanix@pcvm$ ./increase_readiness_probe_timeouts.sh
```

Enabling Policy Engine for Self-Service VM

Use the following steps to enable policy engine for Self-Service VM (formerly Calm VM).

Before you begin

- Ensure that you have the accounts configured. For more information, see [Provider Account Settings in Self-Service](#) on page 52.
- Ensure that you have created a project for the blueprint. For more information, see [Projects Overview](#) on page 78.
- The project to which you upload the blueprint must have the account in which you want to create the policy engine VM.

About this task

Do the following to enable the policy engine on a new deployment of Self-Service VM:

Procedure

1. Download the blueprint from one of the following locations.

- » For a single-node Self-Service VM, download the blueprint from this [Downloads](#) page.
- » For a scale-out Self-Service VM, download the blueprint from this [Downloads](#) page.

2. Upload the downloaded blueprint to a project.

For more information on how to upload the blueprint, see [Uploading a Blueprint](#) on page 226.

Note: You can add any string in the credential password and save the blueprint to avoid any blueprint errors after the upload.

3. Set values for the following variables in the blueprint.

- Desired policy engine IP address. This IP address must be in the same network as that of the Self-Service VM.
- VIP of the Self-Service VM
- Netmask of the Self-Service VM network
- Gateway of the Self-Service VM network
- DNS IP of the Self-Service VM
- NTP IP of the Self-Service VM
- Public key of Self-Service VM. For scale-out Self-Service VM, provide the public key of all VMs. You can run the following command to get the public keys of your VMs.

```
allssh cat .ssh/id_rsa.pub
```

The scale-out Self-Service VM blueprint has additional variables for 3 public keys. You do not need to create additional variables for the public keys.

- Disable check-login in case the check-login is enabled by default.

4. Launch the blueprint.

5. Wait for the app that you created by launching the blueprint to get into the running state.

6. SSH into the Self-Service VM as a Nutanix user and run the following commands after making the required changes.

Note: On a scale-out Self-Service VM, run the following command on any one of the VMs.

```
nutanix@pcvm$ sh /home/nutanix/bin/set_policy_calvm.sh <POLICY_VM_IP>  
<POLICY_VM_UUID>
```

Where, *<POLICY_VM_UUID>* is the UUID of the policy engine VM that you can get either from the Services tab or from the audit logs on the Audit tab after launching the blueprint.

Hardening Self-Service VM

You can use Nutanix Command Line Interface (nCLI) in order to customize the various configuration settings related to hardening Self-Service VM (formerly Calm VM).

Note: In a clustered environment, you only have to run the following commands on a single machine. The configuration changes are synced automatically to the remaining machines.

- Run the following command to support cluster-wide configuration of the SCMA policy.

```
nutanix@cvm$ ncli cluster get-pcvm-security-config
```

The current cluster configuration is displayed.

```
Enable Aide          : false  
  Enable Core        : false  
  Enable High Strength P... : false  
  Enable Banner      : false  
  Enable SNMPv3 Only : false  
  Schedule           : DAILY  
  Enable Kernel Mitigations : false  
  SSH Security Level : DEFAULT  
  Enable Lock Status : false  
  Enable Kernel Core  : false
```

- Run the following command to schedule weekly execution of Advanced Intrusion Detection Environment (AIDE).

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-aide=true
```

The following output is displayed.

```
Enable Aide          : true  
  Enable Core        : false  
  Enable High Strength P... : false  
  Enable Banner      : false  
  Enable SNMPv3 Only : false  
  Schedule           : DAILY  
  Enable Kernel Mitigations : false  
  SSH Security Level : DEFAULT  
  Enable Lock Status : false  
  Enable Kernel Core  : false
```

- Run the following command to enable the settings to generate stack traces for any cluster issue.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-core=true
```

The following output is displayed.

```
Enable Aide          : true  
  Enable Core        : true  
  Enable High Strength P... : false  
  Enable Banner      : false
```

```
Enable SNMPv3 Only      : false
Schedule                : DAILY
Enable Kernel Mitigations : false
SSH Security Level      : DEFAULT
Enable Lock Status      : false
Enable Kernel Core      : false
```

Note: Nutanix recommends that Core should not be set to true unless instructed by the Nutanix support team.

- Run the following command to enable the strong password policy.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-high-strength-  
password=true
```

The following output is displayed.

```
Enable Aide              : true
  Enable Core             : true
  Enable High Strength P... : true
  Enable Banner           : false
  Enable SNMPv3 Only      : false
  Schedule                : DAILY
  Enable Kernel Mitigations : false
  SSH Security Level      : DEFAULT
  Enable Lock Status      : false
  Enable Kernel Core      : false
```

- Run the following command to enable the defense knowledge consent banner of the US department.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-banner=true
```

The following output is displayed.

```
Enable Aide              : true
  Enable Core             : true
  Enable High Strength P... : true
  Enable Banner           : true
  Enable SNMPv3 Only      : false
  Schedule                : DAILY
  Enable Kernel Mitigations : false
  SSH Security Level      : DEFAULT
  Enable Lock Status      : false
  Enable Kernel Core      : false
```

- Run the following command to enable the settings to allow only SNMP version 3.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-snmpv3-only=true
```

The following output is displayed.

```
Enable Aide              : true
  Enable Core             : true
  Enable High Strength P... : true
  Enable Banner           : true
  Enable SNMPv3 Only      : true
  Schedule                : DAILY
  Enable Kernel Mitigations : false
  SSH Security Level      : DEFAULT
  Enable Lock Status      : false
  Enable Kernel Core      : false
```

- Run the following command to change the default schedule of running the SCMA. The schedule can be hourly, daily, weekly, and monthly.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params schedule=hourly
```

The following output is displayed.

```
Enable Aide           : true
  Enable Core         : true
  Enable High Strength P... : true
  Enable Banner       : true
  Enable SNMPv3 Only  : true
  Schedule            : HOURLY
  Enable Kernel Mitigations : false
  SSH Security Level  : DEFAULT
  Enable Lock Status  : false
  Enable Kernel Core  : false
```

- Run the following command to configure security levels for the nutanix user for ssh login to the Nutanix Cluster.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params ssh-security-level=limited
```

The following output is displayed.

```
Enable Aide           : true
  Enable Core         : true
  Enable High Strength P... : true
  Enable Banner       : true
  Enable SNMPv3 Only  : true
  Schedule            : HOURLY
  Enable Kernel Mitigations : false
  SSH Security Level  : LIMITED
  Enable Lock Status  : false
  Enable Kernel Core  : false
```

- Run the following command to enable to locking of the security configuration.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-lock-status=true
```

The following output is displayed.

```
Enable Aide           : true
  Enable Core         : true
  Enable High Strength P... : true
  Enable Banner       : true
  Enable SNMPv3 Only  : true
  Schedule            : HOURLY
  Enable Kernel Mitigations : false
  SSH Security Level  : LIMITED
  Enable Lock Status  : true
  Enable Kernel Core  : false
```

Note: If set true, the configuration settings can not be edited by the user and a support call will need to be made to unlock this configuration.

- Run the following command to enable kernel core dumps.

```
nutanix@cvm$ ncli cluster edit-pcvm-security-params enable-kernel-core=true
```

The following output is displayed.

```
Enable Aide           : true
  Enable Core         : true
  Enable High Strength P... : true
```

```

Enable Banner           : true
Enable SNMPv3 Only     : true
Schedule                : HOURLY
Enable Kernel Mitigations : false
SSH Security Level      : LIMITED
Enable Lock Status      : true
Enable Kernel Core      : true

```

Scenario-Based Hardening

- When a high governance official needs to run the hardened configuration then the settings should be as follows.

```

Enable Aide             : true
  Enable Core           : false
  Enable High Strength P... : true
  Enable Banner         : false
  Enable SNMPv3 Only    : true
  Schedule              : HOURLY
  Enable Kernel Mitigations : false
  SSH Security Level     : LIMITED
  Enable Lock Status     : true
  Enable Kernel Core     : true

```

- When a federal official needs to run the hardened configuration then the settings should be as follows.

```

Enable Aide             : true
  Enable Core           : false
  Enable High Strength P... : true
  Enable Banner         : true
  Enable SNMPv3 Only    : true
  Schedule              : HOURLY
  Enable Kernel Mitigations : false
  SSH Security Level     : LIMITED
  Enable Lock Status     : true
  Enable Kernel Core     : true

```

SELF-SERVICE DSL OVERVIEW

NCM Self-Service DSL, which refers to the Domain-Specific Language (DSL) used in NCM Self-Service (formerly Calm), is a specialized open-source, Python-based programming language that allows you to define and automate tasks and application workflow within your infrastructure as code (IaC).

NCM Self-Service DSL also supports executing CLI commands so you can interact with and use Self-Service features and functionalities conveniently, efficiently, and in an automated manner.

Key Advantages of Using Self-Service DSL

Here are a few key advantages of using Self-Service DSL:

Table 5: Self-Service DSL - Advantages

Advantage	Description
Ease of use	Self-Service DSL provides a straightforward interface for you to interact with your project. You can run key actions and operations directly from the command line using short, easy-to-remember commands. You do not have to navigate through complex graphical user interfaces (GUIs) or write extensive codes.
Efficient Automation	With Self-Service DSL, you can automate tasks and integrate them into scripts or workflow. By running CLI commands, you can trigger specific actions programmatically in Python and easily incorporate Self-Service automation into larger automation pipelines or processes. The DSL scripting capability empowers you to automate complex sequences of tasks or run repetitive operations with minimal manual intervention, saving time and effort.
Accessibility	DSL is a platform-agnostic means of interacting with Self-Service. You can access and run commands from various operating systems and environments, including local machines, servers, or even within containerized or virtualized environments. This flexibility makes your project more accessible to a wider range of users.
Portability	A CLI interface is more portable than a graphic user interface. The CLI interface does not depend on any specific operating system or graphical environment and, therefore, you can use Self-Service automation in a wide range of environments.

Advantage	Description
Version Control and Reproducibility	You can easily capture and store DSL modules in version control systems (such as Git or GitLab). This capability facilitates better tracking of changes and ensures that specific actions can be consistently executed across different environments or by different team members, promoting collaboration and maintaining workflow integrity.

When to use Self-Service DSL

Here are a few examples for you to understand how to use DSL to run certain key actions within Self-Service:

Table 6: Self-Service DSL Usage

Usage	Description
Create App Blueprints	Use Self-Service DSL to create a blueprint that defines a web server application. The blueprint defines the VM on which an application runs, the operating system that the VM uses, the software (packages) installed on the VM, and other advanced capabilities. For more information about blueprints, see Self-Service Blueprints Overview on page 87.
Manage Applications	Use Self-Service DSL to deploy and manage applications through the actions such as starting, stopping, restarting, or performing any custom Day-2 action. For more information on app management, see My Apps in the <i>Prism Central Admin Center Guide</i> .
Automate tasks with Runbooks	Use Self-Service DSL to automate tasks through scripts in Python to create a runbook. A runbook is an automation workflow with a sequence of tasks that you can run on different endpoints. For more information about runbooks, see Runbooks Overview on page 265.

Note:

- Self-Service uses Services, Packages, Substrates, Deployments, and Application Profiles as building blocks for a blueprint, and therefore, these entities can be defined as Python classes. You can specify their attributes as class attributes and define actions on those entities (procedural runbooks) as class methods.
- Self-Service DSL also accepts appropriate native data formats such as YAML and JSON that allow reuse into the larger app life cycle context of a Self-Service blueprint.

Apart from these, you can use Self-Service DSL with other Self-Service features such as Approvals, Job Scheduling, Launching Marketplace Items, and so on, to carry out crucial application management tasks.

Self-Service DSL - Resources

For technical documentation on Self-Service DSL, see [Nutanix Cloud Manager \(NCM\) Self-Service DSL](#) on *Nutanix.dev*.

For videos, labs, and resources on Self-Service DSL, see [Nutanix Self-Service DSL](#) on *Nutanix.dev*.

You can use our mailing lists or create a support ticket in GitHub to get help or additional information. You have the following resources:

- [GitHub Repository](#)
- [Nutanix Community](#)

QUICK START WITH SELF-SERVICE

The topics in this section cover the procedure to launch a blueprint instantaneously in Self-Service and to provision a Linux or Windows infrastructure as a Service (IaaS).

Launching a Blueprint Instantaneously

When you enable Self-Service, you get an out-of-the-box blueprint with your Nutanix account. You can use the blueprint with your configured project and app profile to instantaneously launch your first app.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Take a Tour** in the navigation bar.
4. Click **Launch Blueprint**.
The **Quick Launch A Blueprint** page appears.
5. On the **Select Blueprint** tab, click **Next**.
The default selection for launch is ExpressLaunch.
6. On the **Select Project** tab, select a project and click **Next**.
Projects are logical groupings of user roles, providers, VM templates, and credentials used to manage and launch blueprints within your organization. For more information, see [Projects Overview](#) on page 78.
7. On the **Select App Profile** tab, click **Next**.
The default selection for launch is an app profile that is configured with your Nutanix account.
Application profiles are profiles for different datacenters or cloud services where you want to run your app. For more information, see [Self-Service Blueprints Overview](#) on page 87.
8. On the **Preview and Launch** tab, type a name for your app in the **Application Name** field and click **Launch and Create App**.
The **Preview and Launch** tab displays the VM details of your app.
The app is created and provisioned.

What to do next

You can view the details of the app in My Apps. For more information on app management, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Provisioning a Linux or Windows Infrastructure as a Service (IaaS)

To quickly provision a Linux or Windows Infrastructure as a Service (IaaS) for your end users, you can configure and launch a single-VM blueprint in Self-Service.

Before you begin

- Ensure that you have deployed Self-Service from your Prism Central instance. For more information, see [Deploying Self-Service](#) in the *Prism Central Admin Center Guide*.
- Ensure that you have configured the projects that you want to use to provision your IaaS. For more information, see [Project Management](#) in the *Prism Central Admin Center Guide*.

About this task

Provisioning a Linux or Windows IaaS involves configuring the single-VM blueprint VM specifications and launching the blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. Click **+ Create Blueprint > Single VM Blueprint**.
5. On the **Blueprint Settings** tab, enter a name and description for your blueprint, and select a project and an environment.
6. On the **VM Details** tab, enter a VM name, and then select an account and an operating system.
If you have not configured any account, then keep the default Nutanix account selected. For more information on configuring provider accounts, see [Provider Account Settings in Self-Service](#) on page 52.
You can select **Linux** or **Windows** as the operating system of your IaaS.
7. On the **VM Configuration** tab, do the following:
 - a. Enter the number of vCPU, cores of each vCPU, and total memory to configure the processing unit of the VM.
 - b. Provide the guest customization, if required.
 - c. Based on the operating system you selected, select a Linux image or a Windows image from the image service under the **Disks** section.
 - d. Select a network configuration under the **NICs** section.For more information on the VM configuration for different provider accounts, see [VM Configuration](#) on page 107.
8. Click **Save** to save the blueprint.
9. Click **Launch** to launch the blueprint.
10. Provide an app name, description, environment, and app profile, and then click **Deploy**.
If you have not configured any environment or app profile, keep the default environment and app profile selected.

What to do next

- You can view and access the app in the **Admin Center > My Apps**. For more information on app management, see [My Apps](#) in the *Prism Central Admin Center Guide*.
- You can publish the blueprint to the Marketplace so that other project users can also launch the blueprint and use the IaaS. For more information, see [Submitting a Blueprint for Approval](#) on page 222.
- You can create single-VM blueprints with different provider accounts and launch them. For more information, see [Creating a Single-VM Blueprint](#) on page 106.

SELF-SERVICE SETTINGS

Self-Service Settings allow you to configure general administrative functionalities, provider accounts, credential provider accounts, quota policies, and approval policies. You must be a Prism Central administrator to access Self-Service Settings. You can configure the following:

- Enable showback to estimate the overall service cost of the apps running on your on-prem cloud. For more information, see [Showback](#) on page 42.
- Enable the option to show app protection status. For more information, see [App Protection Status Overview](#) on page 43.
- Configure provider accounts (by using the provider credentials) to enable Self-Service to manage apps by using your virtualization resources. For more information, see [Provider Account Settings in Self-Service](#) on page 52.
- Enable the policy engine to enforce resource quota policy for the infrastructure resources, approval policies, and to schedule app actions. For more information, see [Policy Engine and Quotas Overview](#) on page 44.
- Configure credential provider accounts. For more information, see [Credentials Overview](#) on page 48.

General Settings

The Self-Service General Settings facilitate you to configure Showback, configure app protection status, and manage your Marketplace branding experience.

Showback

Showback allows you to estimate the overall service cost of the apps running on your on-prem cloud. You can also view the graphical representation of the cost of the apps.

Self-Service supports showback for the following platforms.

- Nutanix
- VMware through vCenter

To enable and configure showback, see [Enabling Showback](#) on page 42.

Note: Starting with AOS 5.10 or NCC 3.6.3, Prism Central generates the following NCC alerts for showback:

- Cost Governance (formerly Beam) connectivity with Prism Central
- Prism Central and Prism web console (also known as Prism Element) registration or de-registration

Enabling Showback

Enable Showback to configure the resource cost of your apps and monitor them while you configure a blueprint or manage an app. Showback is applicable only for the Nutanix platform and the VMware through vCenter platform.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.

3. Click **Self-Service Settings** in the navigation bar.
4. On the **General** tab, turn on **Enable Showback**.
The **Enable Showback** window is displayed.
5. Click the supported provider for which you want to define the cost.
6. To configure the resource usage cost, click the vertical ellipsis next to the provider, and then click **Edit**.
7. Configure the cost of the following resources.
 - a. In the **vCPU** field, enter the cost of vCPU consumption for each hour in dollars.
The default value is \$0.01 for each vCPU for each hour.
 - b. In the **Memory** field, enter the cost of memory consumption for each hour in dollars.
The default value is \$0.01 for each GB of usage for each hour.
 - c. In the **Storage** field, enter the cost of storage consumption for each hour in dollars.
The default value is \$0.0003 for each GB of usage for each hour.
8. Click **Enable Showback**.

Disabling Showback

Disable showback to stop monitoring the resources cost of your app blueprints.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **General** tab, turn off **Enable Showback**.
The **Disable Showback** window is displayed.
5. To disable Showback, click **Disable Showback**.

App Protection Status Overview

You can view the protection and recovery status of a Self-Service app when:

- The VMs of the app running on a Nutanix platform are protected by a protection policy in Prism Central.
- You enabled the option to show app protection status in Self-Service.

You can view the protection and recovery status of the app on the My Apps page. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Note:

- The option to show app protection status is available only when at least one VM of the app is protected by a protection policy in Prism Central.
- If the target recovery location is set to another Prism Central, Self-Service still displays the correct protection status. However, the recovery is not tracked, and there is no recovery status available for the app. Self-Service app still points to the old VMs.

To enable the option to show app protection status, see [Enabling App Protection Status View](#) on page 44.

Enabling App Protection Status View

Turn on **Show App Protection Status** to view the protection and recovery status of a Self-Service app that is deployed on a Nutanix platform. You must be a Prism Central administrator to turn on or turn off the toggle button.

Before you begin

- Ensure that the versions of Prism Central and Prism Element are 5.17 or above.
- Ensure that at least one VM of the app is protected by a protection policy in Prism Central.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **General** tab, turn on **Show App Protection Status**.

What to do next

You can view the protection and recovery status of the app on the app details page. For more information, see [Overview Tab](#) in the *Prism Central Admin Center Guide*.

Policy Settings

The Self-Service Policy Settings facilitate you to set quota defaults for your policy engine after you enable the policy engine from the Prism Central Admin Center Settings and manage your approval settings.

For more information on how to set quota defaults, see [Setting up Quota defaults](#) on page 45.

For more information on how to enable policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.

For more information on approval settings, see [Approval Policy Overview](#) on page 242.

Policy Engine and Quotas Overview

The policy engine is a single-VM setup for the single or scale-out Prism Central. You enable and use policy engine to do the following:

- Enforce resource quota policy for the infrastructure resources (compute, memory, and storage) on Nutanix and VMware. For more information, see [Quota Policy Overview](#) in the *Prism Central Admin Center Guide*.
- Orchestrate apps through tunnels on a virtual private network (VPC) on Nutanix accounts. For more information, see [Tunnels for Orchestration within a VPC](#) on page 54.
- Enforce approval policies to manage resources and control actions in your environment. For more information, see [Approval Policy Overview](#) on page 242.
- Schedule app actions and runbook executions. For more information, see [Scheduler Overview](#) on page 238.

When you enable the policy engine for your Prism Central instance, a new VM is created and deployed for the policy engine. To deploy the policy engine VM, you require an available IP address that belongs to the same network as that of your Prism Central VM during the policy engine enablement.

Note: Starting from version pc.2024.1, you can enable policy engine or disable policy enforcement from the Admin Center Settings of your Prism Central instance. For information on enabling policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.

Setting up Quota defaults

After you enable the policy engine, you can set up the default quota values for vCPU, memory, and disk. This step is optional.

About this task

Setting up quota defaults saves you from repeatedly entering vCPU, memory, and disk quota values for each cluster. After you set the quota defaults, the default quota values populate automatically when you allocate quotas to your provider accounts.

Note: The quota defaults are visible in the accounts only after the next platform sync. You can also run the platform sync manually. For information on how to run platform sync, see [Synchronizing Platform Configuration Changes](#) on page 73.

Before you begin

Ensure that you enabled the policy engine for your Prism Central instance. For information on enabling the policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **Policies** tab, click **Quotas** in the left pane.
5. Select the **Set Quota Defaults** checkbox.
6. Specify the values for **vCPU**, **Memory**, and **Disk**.
7. Click the **Save** icon next to the fields.

Approval Settings

Enabling Approvals

You can enable approvals for your Self-Service instance from the Self-Service Settings page.

About this task

When you enable approvals, events such as runbook executions, app launch, and app day-2 operations that match the conditions defined in the approval policy go through the approval process.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.

3. Click **Self-Service Settings** in the navigation bar.
4. On the **Policies** tab, click **Approvals** in the left pane.
5. Turn on **Approvals** to enable approvals.

Disabling Approvals

You can disable approvals for your Self-Service instance from the Self-Service Settings page.

About this task

When you enable approvals, events such as runbook executions, app launch, and app day-2 operations do not go through the approval process even when they match the conditions defined in the approval policy.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **Policies** tab, click **Approvals** in the left pane.
5. Turn off **Approvals** to disable approvals.

Viewing Approval Email Templates

You can view the configuration details and email template on the **Policies** tab of the Self-Service Settings page.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **Policies** tab, click **Approvals** in the left pane.
5. Under the General Configuration section, view details such as the number of days after which a request expires and the frequency of the email sent to the approver and requester.

- Under the Email Content section, click **For Approver** or **For Requester** to view the template of the emails that are sent with each request.

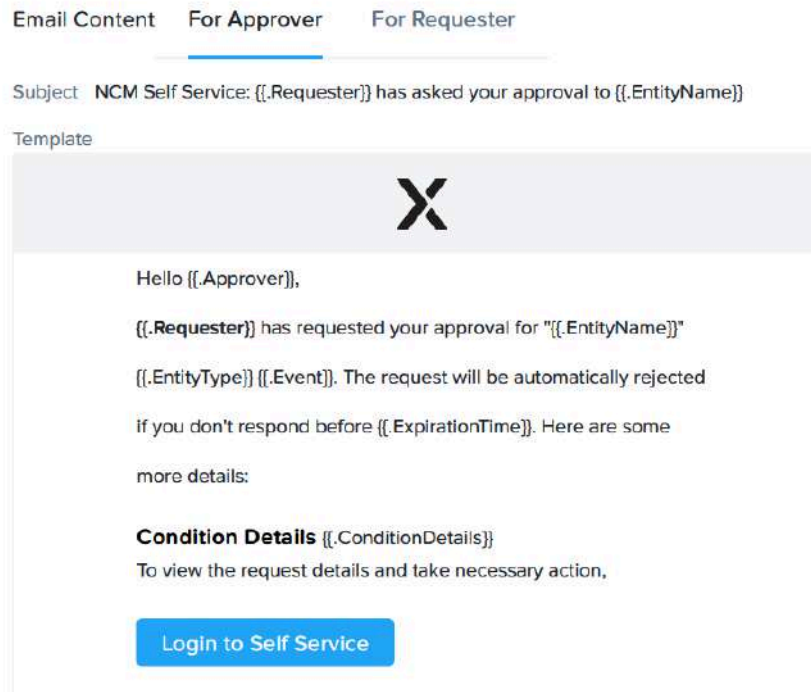


Figure 10: Email Template

The content of the email templates for approver or requester can be modified only using the APIs. You can use the following supported email template variables.

- Approver
- Requester
- ConditionDetails
- Event
- EntityType
- EntityName
- State
- PCIP
- CreationTime
- ExpirationTime
- NutanixLogo

You can use these variables with the `{{ }}` syntax. For example, `{{.PCIP}}`.

Credential Settings

Credentials help in abstracting identity settings while connecting to an external system. Credentials are used to authenticate a user to access various services in Self-Service. Self-Service supports key-based and password-based authentication method.

Credentials Overview

Credentials are used in multiple Self-Service entities and workflow.

- **Environment**

Environment allows a Project Admin to add multiple credentials and configure VM default specifications for each of the selected providers as a part of project and environment configurations.

Project admins must configure an environment before launching an app from the marketplace. The recommendation is to have at least one credential of each secret type (SSH or password) to be defined under each environment in the project. These values get patched wherever the credential values are empty when you launch your marketplace items.

- **Blueprints and runbooks**

Developers can add credentials to a blueprint. These credentials are referenced after the VM is provisioned. Credentials defined within an environment of a project have no significance or impact on the credentials you define within the blueprint.

Self-Service supports export and import of blueprints across different Prism Central or Self-Service instances along with the secrets. The developer uses a passphrase to encrypt credentials and then decrypts credentials in a different instance using the same passphrase to create a blueprint copy.

- **Marketplace**

All global marketplace items have empty credentials values. However, locally published blueprints can have the credential values if the developer published the blueprint with the **Publish with Secrets** option enabled.

When you launch a marketplace item, credentials are patched wherever the value is empty. In case there are multiple credentials of a particular type configured within the environment of a project, you get the option to select a credential for the launch.

- **Applications**

Owners can change the credential value of an app multiple times until the app is deleted. The latest value of a credential that is available at that point in the app instance is used when an action is triggered.

Any change in the credential value at the app level does not impact the credential value at the corresponding blueprint level.

Self-Service allows managing the following types of credentials:

- **Static Credentials**

Static credentials in Self-Service are modelled to store secrets (password or SSH private key) in the credential objects that are contained in the blueprints that the apps copy.

- **Dynamic Credentials**

Self-Service supports external credential store integration for dynamic credentials. A credential store holds username and password or key certificate combinations and enables apps to retrieve and use credentials for authentication to external services whenever required. As a developer, you can:

- Define credential attributes that you want to pass on to the credential provider from the blueprint during execution.
- Define variables that the credential provider must use. By default, Self-Service defines the secret variable when you configure your credential provider.

Note: To use the credential in the blueprint, the variables in the credential and the blueprint must match.

- Define a runbook with eScript tasks in the dynamic credential provider definition. The tasks you define in the runbook can set the username, password, private key, or passphrase values for the credential.

Note: Nutanix is going to end support for Python 2 on June 30, 2024. Therefore, format all your new eScripts in Python 3 and update any existing Python 2 eScripts to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

For more information on configuring a credential provider, see [Configuring a Credential Provider](#) on page 49.

When a blueprint uses a dynamic credential, the secret (password or SSH private key) is not stored in the credential objects within the blueprint. The secret values are fetched on demand by executing the runbook within the credential provider that you configure in Self-Service and associate with the blueprint.

Note:

- You cannot add a dynamic credential when you configure an environment in our project. You can, however, allow the credential provider in the environment.
- You cannot use dynamic credentials for HTTP variables, such as profile variables, service variables, runbooks, and so on.
- You cannot use dynamic credentials for HTTP endpoints and Open Terminal in apps.
- For ready-to-use blueprints or blueprints that are published without secrets, the empty credential values are patched with the credential along with its associated runbook and variable values.

Configuring a Credential Provider

Self-Service supports external credential store integration for dynamic credentials.

About this task

As a developer, you can define variable, runbook, and attributes in a dynamic credential provider definition.

Note: HashiCorp Vault Sample is available as a preseeded credential provider for reference implementation. The sample has basic variable and runbook configurations and can be changed according to your specific requirements.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the Credential Providers tab, click **Add Credential Provider**.
5. On the Credentials Provider Account Settings page, enter a name for the credential provider in the **Name** field.
6. Enter the server address of the credential provider in the **Provider Server Address** field.
7. Enter the secret for the account in the **Provider Secret** field. The secret for the provider can be a key, token, or password.
8. To add the credential attributes, click the **+** icon next to **Credential Attributes** and do the following:
 - a. Enter a name for the credential attribute in the **Name** field.
 - b. Select a data type for the credential attribute in the **Data Type** field.
 - c. Provide a value for the data type you selected in the **Value** field. You can select the **Secret** checkbox to hide the value of the credential attribute.

Credential attributes are the variables that you pass on to the credential provider from the blueprint during execution. Developers require these attributes in blueprints and runbooks to use the credential provider.
9. To add variables, click the **+** icon in the **Variables** section and do the following:
 - a. Enter a name for the variable in the **Name** field.
 - b. Select a data type for the variable in the **Data Type** field.
 - c. Provide a value for the data type you selected in the **Value** field. You can select the **Secret** checkbox to hide the value of the variable.

The variables that you add during the credential provider configuration are only used in the runbooks that you define for the credential provider.

To use the credential in the blueprint, the variables in the credential and the blueprint must match.
10. Configure a runbook for the credential provider. For information on runbook configuration, see [Runbooks Overview](#) on page 265.

You can define a runbook with an eScript task. The tasks are used to set the username, password, private key, or passphrase values for the credential.

Note: When the runbook uses the eScript task, then do the following in the Set Variable eScript task to fetch SSH keys or multi-line secrets from the credential provider.

- Encode the secrets.
- Set the `is_secret_encoded` variable to True.

Ensure that you encode in the Python 3 format.

The runbook uses the variables you defined during the credential provider configuration. You can click **Variable/Attributes** within the runbook to view, edit, or add variables. After your runbook is defined, you can click **Test** to test the runbook.

11. Click **Save**.

What to do next

You can add the credential provider to a project. For more information, see [Infrastructure in Projects](#) in the *Prism Central Admin Center Guide*.

PROVIDER ACCOUNT SETTINGS IN SELF-SERVICE

Provider accounts are cloud services, baremetals, or existing machines that you can use to deploy, monitor, and govern your apps. You can configure multiple accounts of the same provider.

Use the **Self-Service Settings > Accounts** tab to configure provider accounts. You configure provider accounts (by using the provider credentials) to enable Self-Service to manage apps by using your virtualization resources.

Self-Service supports the following provider accounts:

Table 7: Provider Accounts

Provider Accounts	Description
Nutanix	All the AHV clusters that are registered to the Prism Central instance are automatically added as providers. Note: If you want to add a remote Prism Central (PC) instance as a provider in a multi-PC setup, you must add the remote PC instance as an account in Self-Service. For more information, see Configuring a Remote Prism Central Account on page 53.
VMware	To configure a VMware account, see Configuring a VMware Account on page 58.
AWS	To configure an AWS account, see Configuring an AWS Account on page 61.
Azure	To configure an Azure account, see Configuring an Azure Account on page 67.
GCP	To configure a GCP account, see Configuring a GCP Account on page 66.
Kubernetes	To configure a Kubernetes account, see Configuring a Kubernetes Account on page 69.
Nutanix Cloud	To configure Nutanix Cloud as a provider, see Configuring a Nutanix Cloud Account on page 72.

Nutanix Account Configuration

All AHV clusters that are registered to your Prism Central instance are automatically added as provider accounts to Self-Service.

You can also configure any remote Prism Central (PC) as an account in Self-Service to deploy apps on the remote PC. For more information, see [Support for the Multi-Prism Central Setup](#) on page 53.

Support for the Multi-Prism Central Setup

In a multi-Prism Central setup, a central Self-Service instance (called global Self-Service instance) runs only on one of the Prism Centrals (called host or parent Prism Central) and all the other Prism Centrals are connected to the central Self-Service instance as remote Prism Centrals.

The global Self-Service instance can now manage the apps deployed on the geographically distributed Prism Centrals (also called remote Prism Centrals) without the need of separate Self-Service instances for every Prism Central. A remote Prism Central is only used to provision the tasks for the deployed apps.

In a multi-Prism Central environment, every remote Prism Central is added as an account to the host Prism Central, and you can add the account to your project before creating and launching a blueprint.

Before you launch the first blueprint to the destination Remote Prism Central site on a multi-Prism Central setup, you must perform the image checkout on the destination Prism Element cluster.

For more information on adding a remote Prism Central as an account, see [Configuring a Remote Prism Central Account](#) on page 53.

For more information on adding the account to a project, see [Infrastructure in Projects](#) in the *Prism Central Admin Center Guide*.

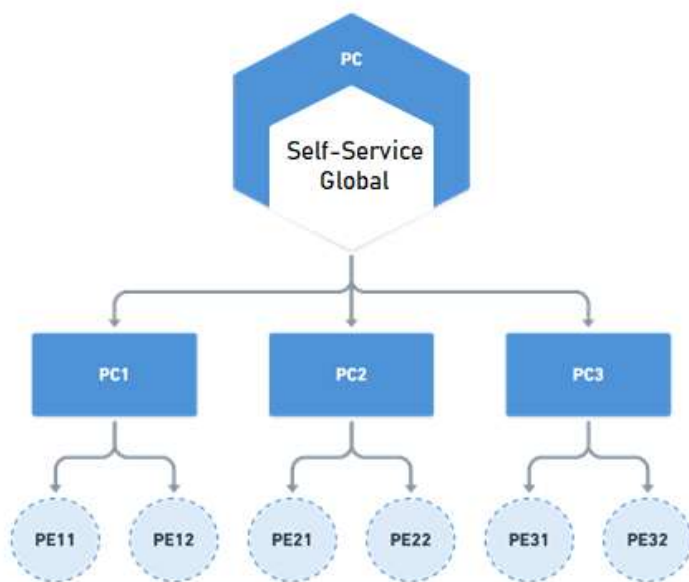


Figure 11: Multi-Prism Central Setup

Configuring a Remote Prism Central Account

To deploy an app on a remote PC, you must configure the remote PC as an account in Self-Service.

About this task

You require the role of a Prism Admin to configure a remote PC account.

For more information on multiple Prism Central setup support, see [Support for the Multi-Prism Central Setup](#) on page 53.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.

3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Account** tab.
The account inspector panel appears.
5. Click **+Add Account**.
The **Account settings** page appears.
6. In the **Name** field, type a name for the PC account.
7. From the **Provider** dropdown menu, select **Nutanix**.
8. In the **PC IP** field, type the IP address of the remote PC.
The app is provisioned in the remote PC IP address.
9. In the **PC Port** field, type the port number for the IP address.
10. In the **User name** field, type the administrator username of the remote PC.
11. In the **Password** field, type the administrator password of the remote PC.
12. In the **Account Sync Interval** field, specify the interval after which the platform sync must run for a cluster.
Self-Service uses platform sync to synchronize any configuration changes occur in Self-Service-managed resources, such as IP Address changes, disk resizing, and so on. Platform sync enables Self-Service to maintain accurate quota and Showback information.
13. Click **Save**.
The account list displays the account that you created.
14. To verify the credentials of the account, click **Verify**.
Self-Service adds the remote PC as a Nutanix account after credential authentication and account verification.

Tunnels for Orchestration within a VPC

Self-Service lets you use Virtual Private Clouds within the Flow Virtual Networking framework to network the VMs using overlay networks. A VPC is an independent and isolated IP address space that functions as a logically isolated virtual network. VMs that you create with VPC Subnets cannot communicate with a VM that is outside the VPC. Even the VMs outside the VPC cannot reach the VMs within the VPC.

In the absence of this direct communication, you can set up tunnels to communicate with the VMs within the VPC for orchestration activities and to run script-based tasks. You can set up the tunnel VM in any one of the subnets within the VPC. For more information on VPCs, see [Virtual Private Cloud](#).

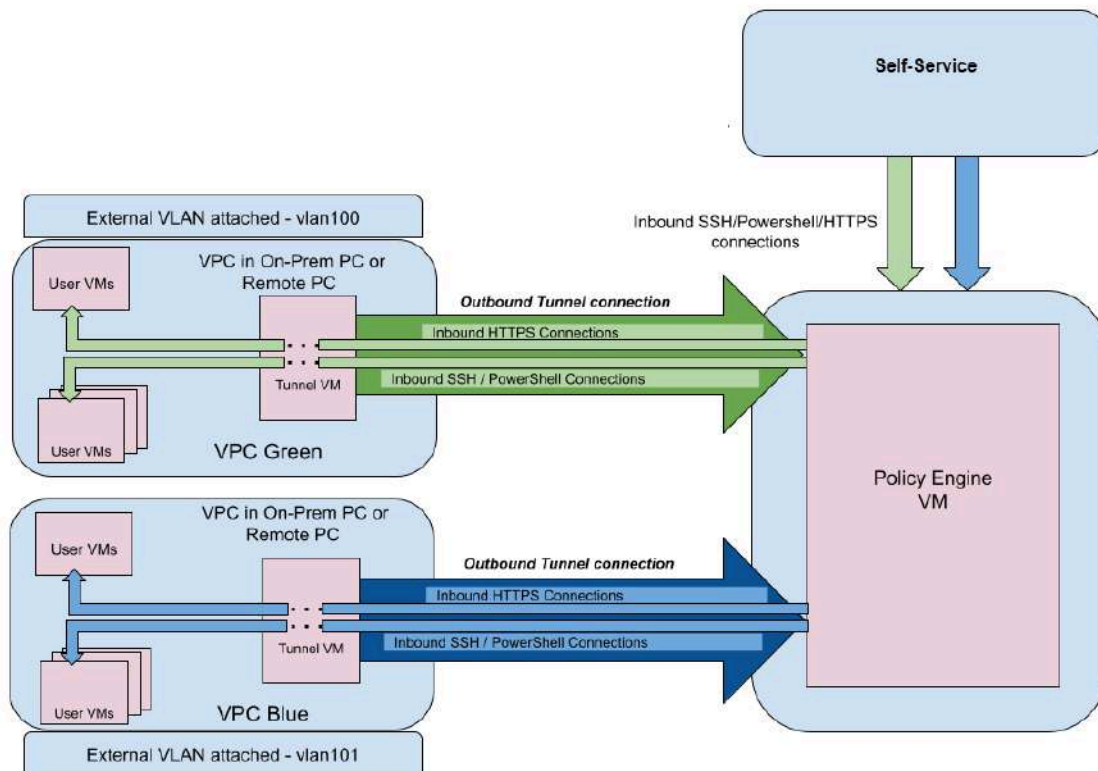


Figure 12: Diagrammatic Representation of the VPC Tunnel Configuration

To set up tunnels for your VPCs, you must:

- Have Flow Virtual Networking enabled in the Prism Central that hosts the Self-Service instance. For more information, see [Enabling Flow Networking](#).
- Enable the Network Controller.
- Attach an external subnet (VLAN) to the VPC to enable the tunnel VM to reach the policy engine VM and establish a tunnel connection. An external subnet allows VMs inside the VPC to reach the VMs that are outside the VPC network.
- Ensure that the VMs inside the VPC is able to ping the policy engine VM and Prism Central.
- Permit traffic from the tunnel VM to the Policy Engine VM on port 2222 using TCP connections.
- Allow connections on default 22 port for SSH script execution or default 5985 for Powershell script execution on the target VM.
- Enable the policy engine in Self-Service to allow the tunnel VM to communicate with Self-Service.
- Have 2 vCPUs, 2 GiB memory and 10 GiB disk space for the tunnel VM.

For more information on creating VPC tunnels, see [Creating VPC Tunnels](#) on page 55.

Creating VPC Tunnels

In your Nutanix account, you set up tunnels to get access to the VMs that are created within the VPCs.

About this task

The tunnels that you create enables you to perform check log-in and run script-based execution tasks on the VMs that use the overlay subnets of the VPC.

Note: In a Nutanix VPC tunnel, using subnet value as a macro for overlay subnets of the VPC is not supported.

If tunnel is not configured for the selected VPC, you can only perform basic operations (such as VM provisioning) on the VPC.

Before you begin

Ensure that you have enabled the policy engine on the **Settings** page. For information on enabling the policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*. For information on the other requirements to set up the tunnel, see [Tunnels for Orchestration within a VPC](#) on page 54.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
5. Select the Nutanix account in the left pane.
The **Account Settings** page appears.

6. In the **VPC Tunnels** section, do the following to set up the tunnel.

- a. Click **Create Tunnel**.
The Create Tunnel for VPCs window appears.

The screenshot shows the 'Create Tunnel for VPCs' window. It includes a title bar, a 'Select a VPC to connect to' section with two dropdowns, a 'Tunnel Configuration' section with an information box and three dropdowns, and two text input fields at the bottom. The 'Create' button is highlighted in blue.

Figure 13: Create Tunnel

- b. From the **Select VPC** dropdown menu, select the VPC on which you want to set up the tunnel.
- c. From the **Select VPC Subnet** dropdown menu, select the subnet that must be used for the NIC of the tunnel VM.
- d. Under Tunnel Configuration section, select the cluster on which you want to place the VM from the **Select Cluster** dropdown menu.
- e. From the **Select Categories** dropdown menu, select the categories that you want to associate with the tunnel.
Categories are grouping of entities into key-value pairs and are used for organizing, reporting, and filtering resources and applying policies. For more information on categories, see [Category Management](#) in the *Prism Central Infrastructure Guide*.
- f. In the **Tunnel Name** field, edit the name of the tunnel. This step is optional.

The tunnel name is auto-generated to ensure uniqueness. You can only edit or append based on your requirement.

- g. In the **Tunnel VM Name** field, edit the tunnel VM name. This step is optional.

The tunnel VM name is auto-generated to ensure uniqueness. You can only edit or append based on your requirement.

- h. Click **Create**.

Configuring a VMware Account

Configure your VMware account in Self-Service to manage apps on the VMware platform.

About this task

Note:

- If you do not have an administrator user account in vCenter while configuring the account, then you can also use a user account with required permissions. For more information, see [Permission Required in vCenter](#) on page 59.
- You cannot enable Self-Service with vSphere Essentials edition license because vSphere Essentials edition does not support hot-pluggable virtual hardware.
- With VMware accounts, Self-Service also supports virtual switch (vSwitch) networks and VMware NSX-based networks.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
The account inspector panel is displayed.
5. Click **+Add Account**.
The **Account Settings** page appears.
6. In the **Name** field, type a name for the account.

Note: The name you specify appears as the account name when you add the account to a project.

7. From the **Provider** dropdown menu, select **VMware**.
8. In the **Server** field, type the server IP address of the vCenter server.
9. In the **Username** field, type the user name of the vCenter account.
If a domain is part of the username, then the username syntax must be <username>@<domain>.
10. In the **Password** field, type the password of the account.
11. In the **Port** field, type the port number as 443.
12. Click **Save**.

13. From the **Datacenter** dropdown menu, select the datacenter.
A VMware datacenter is the grouping of servers, storage networks, IP networks, and arrays. All the datacenters that are assigned to your vCenter account are available for your selection.
14. In the **Account Sync Interval** field, specify the interval after which the platform sync must run for a cluster.
Self-Service uses platform sync to synchronize any configuration changes occur in Self-Service-managed VMware resources, such as IP Address changes, disk resizing, and so on. Platform sync enables Self-Service to maintain accurate quota and Showback information.
15. To monitor the operating cost of your apps, configure the cost of the following resources. This step is optional.

Note: Ensure that you have enabled showback. For more information on enabling showback, see [Enabling Showback](#) on page 42.

 - In the **vCPU** field, type the cost of vCPU consumption for each hour in dollars. The default value is \$0.01 for each vCPU for each hour.
 - In the **Memory** field, type the cost of memory consumption for each hour in dollars. The default value is \$0.01 for each GB of usage for each hour.
 - In the **Storage** field, type the cost of storage consumption for each hour in dollars. The default value is \$0.0003 for each GB of usage for each hour.
16. Click **Save**.
17. To verify the credentials of the account, click **Verify**.
Self-Service lists the account as an active account after successful credential authentication and account verification.

What to do next

Create a project and configure a VMware environment. For more information, see [Project Management](#) in the *Prism Central Admin Center Guide*.

Permission Required in vCenter

The following table provides the complete list of permissions that you need to enable in vCenter before you configure your VMware account in Self-Service.

Table 8: Permissions required in vCenter

Entity	Permission
Datastore	<ul style="list-style-type: none">• Allocate space• Browse datastore• Low level file operation• Update virtual machine files

Entity	Permission
Network	<ul style="list-style-type: none"> • Assign Network • Configure • Move Network
Resource	<ul style="list-style-type: none"> • Assign virtual machine to resource pool
vSphere Tagging	<ul style="list-style-type: none"> • All
Virtual Machine > Change Configuration	<ul style="list-style-type: none"> • Add existing disk • Add new disk • Add or remove device • Change CPU count • Change memory • Modify device settings • Configure raw device • Rename • Set annotation • Change settings • Upgrade virtual machine compatibility
Virtual Machine > Interaction	<ul style="list-style-type: none"> • Configure CD media • Connect devices • Power On • Power off • Reset • Install VMware tools
Virtual Machine > Edit Inventory	<ul style="list-style-type: none"> • Create from existing • Remove
Virtual Machine > Provisioning	<ul style="list-style-type: none"> • Clone template • Customize guest • Deploy template • Read customization specifications

You must define the custom role at the vCenter level instead of the Datacenter level. For information on how to enable permissions in vCenter, see the *vSphere Users and Permissions* section in the *VMware Documentation*.

Supported vSphere Versions

Self-Service supports the following versions of vSphere.

- 7.0
- 6.7
- 6.5
- 6.0

Configuring an AWS Account

Configure your AWS account in Self-Service to manage apps on the AWS platform.

Before you begin

Ensure that you have the following accounts and details.

- An AWS account with valid credentials.
- An IAM user account. For information on how to create an IAM user account, refer to *AWS Documentation*.
- A user account with full EC2 access and IAM read-only access.
- The access key ID and the secret access key for the IAM user account.

Note:

- Ensure that you have configured the domain name server (DNS). For more information, see [Configuring Name Servers](#) in the *Prism Central Admin Center Guide*.
- If you are configuring DNS now, then you must restart the Prism Central VM.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
The account inspector panel appears.
5. Click **+Add Account**.
The **Account Settings** page appears.
6. In the **Name** field, type a name for the account.
7. From the **Provider** dropdown menu, select **AWS**.
8. In the **Access Key ID** field, type the access key ID of your AWS account.
9. In the **Secret Access Key** field, type the secret access key of your AWS account.

10. From the **Regions** dropdown menu, select the geographical regions.

By default, Self-Service includes all regions except China and GovCloud in the account. You can remove a region from the account. You can also clear the **All Regions** checkbox and select regions from the **Regions** dropdown menu.

Warning: Removing a region from an existing AWS account impacts the deployed VMs in that region.

11. In the **Search Public Image** field, search the public image applicable to your region, and select the public image. This step is optional.

You must authenticate the credentials before searching. You can select multiple public images and use any of the selected public images when you create a blueprint for AWS.

12. Click **Save**.

13. To verify the credentials of the account, click **Verify**.

Self-Service lists the account as an active account after successful credential authentication and account verification.

What to do next

Create a project and configure an AWS environment. For more information, see [Project Management](#) in the *Prism Central Admin Center Guide*.

Configuring AWS C2S Provider on Self-Service

GovCloud (US) is an isolated AWS region to help the United States government agencies and federal IT contractors host sensitive workloads into the cloud by addressing their specific regulatory and compliance requirements.

About this task

With AWS C2S support in Self-Service, you can configure your GovCloud authentication, and then create or manage your workload instances on AWS GovCloud region as done for other AWS regions. The AWS GovCloud provides the same high-level security as other AWS regions, however, the Commercial Cloud Services (C2S) and the C2S Access Portal (CAP) are used to grant controlled access to the C2S Management Console and C2S APIs for Government users and apps.

Note:

The AWS GovCloud (US) region supports the management of regulated data by restricting physical and logical administrative access to U.S. citizens only.

Before you begin

Ensure that you have the following accounts.

- An AWS GovCloud (US) account with valid credentials.
- A C2S account configured in AWS.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.

4. Click the **Accounts** tab.
The inspector panel appears.
5. In the **Name** field, type the name of the account.
6. From the **Type** dropdown menu, select **AWS C2S**.
7. In the **C2S account address** field, type the C2S account IP address.
8. In the **Client Certificate** field, type or upload the client certificate.
9. In the **Client Key** field, type or upload the client key.
10. In the **Role** field, type the required IAM role.
11. In the **Mission** field, type the mission.
12. In the **Agency** field, type the agency.
13. Select **All GovCloud Regions** checkbox to select all the GovCloud regions.
14. Click **Save**.
15. To verify the credentials of the account, click **Verify**.
Self-Service lists the account as an active account after successful credential authentication and account verification.

What to do next

You can use the configured AWS C2S provider while you create a blueprint.

Configuring AWS User Account with Minimum Privilege

To manage apps on the AWS platform using Self-Service, you must have a privileged AWS user account with an appropriate policy.

Before you begin

Ensure that you have an AWS administrator user account.

Procedure

1. Log on to the AWS console with your AWS administrator account.
2. Click **Services > IAM**.
3. To add a user, click **Users > Add User**.

4. On the Add User page, do the following.

- a. In the **User name** field, type a user name.
- b. In the **Access Type** area, select the checkboxes next to the **Programmatic access** and **AWS Management Console access** fields, and then click **Next: Permission**.

Note: Do not configure any fields on the Set Permission page.

- c. Click **Next: Tags**.
- d. To add a tag to a user, type the key and value pair in the **Key** and **Value** fields.
For more information on IAM tags, see *AWS Documents*.
- e. Click **Next: Review**.
- f. Click **Create User**.
An IAM user is created.
- g. To display the credential of the user, click **Show** in the **Access key ID**, **Secret access Key**, and **Password** fields.

Note: Copy the credentials in a file and save the file on to your local machine. You need the credentials when you configure AWS as an account in Self-Service to manage apps.

5. To assign permission to the user, click the user you created on the Users page.
The Summary page appears.

6. On the **Permissions** tab, click **+ Add inline policy**.

7. On the Create Policy page, click the **JSON** tab and use the following JSON code in the code editor area.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListRoles",
        "iam:ListSSHPublicKeys",
        "iam:GetSSHPublicKey",
        "iam:GetAccountPasswordPolicy",
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:RebootInstances",
        "ec2:CreateTags",
        "ec2:CreateVolume",
        "ec2:CreateSnapshot",
        "ec2:CreateImage",
        "ec2:ModifyImageAttribute",
        "ec2:ModifyInstanceAttribute",
        "ec2:AttachVolume",
        "ec2:DetachVolume",
        "ec2:ModifyVolume",
        "ec2:AssociateIamInstanceProfile",
        "ec2:ReplaceIamInstanceProfileAssociation",
        "ec2:DisassociateIamInstanceProfile",
        "ec2:RegisterImage",

```



```

        "ec2:DeregisterImage",
        "ec2:DeleteSnapshot",
        "ec2:GetConsoleOutput",
        "ec2:Describe*",
        "ec2:DeleteTags",
        "ec2:TerminateInstances"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [ "iam:ListUserPolicies" ],
    "Resource": [ "arn:aws:iam::*:user/${aws:username}" ]
  },
  {
    "Effect": "Allow",
    "Action": [ "iam:PassRole" ],
    "Resource": [ "arn:aws:iam::*:role/*" ]
  }
]
}

```

8. Click **Review Policy**.

9. On the Review Policy page, in the **Name** field, type a name for the policy and click **Create policy**.

What to do next

You can configure AWS as a provider on the Settings page. For more information, see [Configuring an AWS Account](#) on page 61. You can also assign different policy privileges to the user. For more information, see [AWS Policy Privileges](#) on page 65.

AWS Policy Privileges

The following table displays the list of user policy privileges and the corresponding JSON attributes that you can add in the JSON syntax to assign different privileges to a user.

Table 9: User Privileges and the JSON attributes

To create	JSON attributes
EC2 Instances	<code>ec2:RunInstances</code>
Volumes	<code>ec2:CreateVolume</code>
Snapshot	<code>ec2:CreateSnapshot</code>
Image(AMI)	<code>ec2:CreateImage</code>
To list or get	JSON attributes
SSH Public Keys for all users	<code>iam:ListSSHPublicKeys</code>
List IAM Roles	<code>iam:ListRoles</code>
EC2 attributes	<code>ec2:Describe*</code>
EC2 instance console output	<code>ec2:GetConsoleOutput</code>
IAM user policies for the user	<code>iam:ListUserPolicies</code>

To update	JSON attributes
Image(AMI) attributes	<code>ec2:ModifyImageAttribute</code>
To delete	JSON attributes
EC2 Instances	<code>ec2:TerminateInstances</code>
Instance Tags	<code>ec2:DeleteTags</code>
Snapshot	<code>ec2:DeleteSnapshot</code>
Images(deregister images)	<code>ec2:DeregisterImage</code>
Others	JSON attributes
Start/Stop/Restart Instances	<code>ec2:RunInstances</code> , <code>ec2:StartInstances</code> , <code>ec2:StopInstances</code> , <code>ec2:RebootInstances</code>
Pass and IAM role to service	<code>iam:PassRole</code>

Configuring a GCP Account

Configure your GCP account in Self-Service to manage apps on the GCP platform.

Before you begin

Ensure that you have the service account file of your GCP account in a JSON format saved on your local machine. To create a GCP service account file, see the *GCP documentation*.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
The account inspector panel is displayed.
5. Click **+Add Account**.
The **Account Settings** page appears.
6. In the **Name** field, type a name for the account.
7. From the **Provider** dropdown menu, select **GCP**.
8. To import the service account file from your local machine, click **Service Account File**.
A service account file is a special Google account file that you can use to upload the details of your GCP account.
The values in the **Project ID**, **Private Key**, **Client Email**, and **Token URI** fields are auto-filled after you upload the file.
9. From the **Regions** dropdown menu, select the geographical regions.
By default, Self-Service includes all regions in the account. You can remove a region from the account. You can also clear the **All Regions** checkbox and select regions from the **Regions** dropdown menu.

Warning: Removing a region from an existing GCP account impacts the deployed VMs in that region.

10. Select the **Enable GKE** checkbox to enable Google Kubernetes Engine (GKE). This step is optional.
11. In the **Server IP** field, type the GKE leader IP address.
12. In the **Port** field, type the port number.
13. Click **Save**.
14. To verify the credentials of the account, click **Verify**.
Self-Service lists the account as an active account after successful credential authentication and account verification.

What to do next

To troubleshoot some common issues, see [KB-5616](#). You can create a project and configure a GCP environment. For more information, see [Project Management](#) in the *Prism Central Admin Center Guide*.

Configuring an Azure Account

Configure your Azure account in Self-Service to manage apps on the Azure platform.

About this task

Note:

- For detailed description of the required fields, refer to the *Azure documentation*.
- Only authorized organizations can use restricted regions like Australia Central through Self-Service. For more information, refer to the *Microsoft documentation*.

Before you begin

- Assign appropriate role to your app. For detailed information, refer to the *Microsoft documentation*.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
The account inspector panel appears.
5. In the **Name** field, type a name for the account.
6. From the **Provider** dropdown menu, select **Azure**.
7. Do the following under the Service Principal Credentials section.
 - a. In the **Directory/Tenant ID** field, type the directory/tenant ID of your Azure app.
 - b. In the **Application/Client ID** field, type the app ID or client ID.
 - c. In the **Client Key/Secret** field, type the client key or secret.

8. From the **Subscriptions** dropdown menu, select your Azure subscriptions.
The subscriptions you select provide access to the associated resource groups during blueprint configuration. The subscriptions allow you to launch VMs in the associated resource groups with a single account. When you do not select any subscriptions, Self-Service provides access to all the subscriptions available in the Azure service principal.
9. From the **Default Subscription** dropdown menu, select a default subscription. This step is optional.
Specify the default subscription if the Azure VM configurations such as blueprints and marketplace items created in any earlier versions of Self-Service require any backward compatibility.
10. From the **Cloud Environment** dropdown menu, select a cloud environment.
You can select **Public Cloud**, **US Government Cloud**, **China Cloud**, or **German Cloud**.
11. Click **Save**.
12. To verify the credentials of the account, click **Verify**.
Self-Service lists the account as an active account after successful credential authentication and account verification.

What to do next

Create a project and configure an Azure environment, see [Project Management](#) in the *Prism Central Admin Center Guide*.

Configuring Azure User Account with Minimum Privilege

You must have a privileged Azure user account to manage apps on an Azure platform using Self-Service.

About this task

For information on assigning minimum privilege to configure Azure account to work with Self-Service, see the [Azure Setup and App Deployment](#) video.

Procedure

1. Log on to Azure portal with your administrator account.
2. Open <https://shell.azure.com> and select bash.
3. Create a .json file with the following content.

```
{
  "Name": "Calm Admin",
  "IsCustom": true,
  "Description": "For calm to manage VMs on azure provisioned from calm applications",
  "Actions": [
    "Microsoft.Storage/storageAccounts/read",
    "Microsoft.Storage/storageAccounts/write",
    "Microsoft.Storage/checknameavailability/read",
    "Microsoft.Storage/skus/read",
    "Microsoft.Network/virtualNetworks/subnets/*",
    "Microsoft.Network/virtualNetworks/read",
    "Microsoft.Network/networkSecurityGroups/*",
    "Microsoft.Network/networkInterfaces/*",
    "Microsoft.Network/publicIPAddresses/*",
    "Microsoft.Network/publicIPPrefixes/*",
    "Microsoft.Compute/availabilitySets/vmSizes/read",
    "Microsoft.Compute/availabilitySets/read",
    "Microsoft.Compute/availabilitySets/write",
  ]
}
```

```

"Microsoft.Compute/disks/*",
"Microsoft.Compute/images/read",
"Microsoft.Compute/images/write",
"Microsoft.Compute/locations/publishers/read",
"Microsoft.Compute/locations/publishers/artifacttypes/offers/read",
"Microsoft.Compute/locations/publishers/artifacttypes/offers/skus/read",
"Microsoft.Compute/locations/publishers/artifacttypes/offers/skus/versions/read",
"Microsoft.Compute/skus/read",
"Microsoft.Compute/snapshots/*",
"Microsoft.Compute/locations/vmSizes/read",
"Microsoft.Compute/virtualMachines/*",
"Microsoft.Resources/subscriptions/resourceGroups/read",
"Microsoft.Resources/subscriptions/resourceGroups/write",
"Microsoft.Resources/subscriptions/resourceGroups/delete",
"Microsoft.GuestConfiguration/*/read",
"Microsoft.GuestConfiguration/*/write",
"Microsoft.GuestConfiguration/*/action",
"Microsoft.Compute/galleries/read",
"Microsoft.Compute/galleries/images/read",
"Microsoft.Compute/galleries/images/versions/read",
"Microsoft.KeyVault/vaults/read",
"Microsoft.KeyVault/vaults/deploy/action"
],
"NotActions": [],
"AssignableScopes": [
  "/subscriptions/<subscription id>"
]
}

```

4. In the Azure cloud shell, run the following command.

```
az role definition create --role-definition <file>.json
```

Use the file you created in step 4 in place of *<file>.json*.

Calm Admin user role is created.

5. In the Azure cloud shell, run the following command to create an Azure Service Principal. The command returns all the information required to add the Azure account in Self-Service.

```
az ad sp create-for-rbac -n "CalmAccount" --role "Calm Admin"
```

6. Copy the values for *appId*, *password*, and *tenant*. You need these values to add the Azure account in Self-Service.

Configuring a Kubernetes Account

Configure your Kubernetes account in Self-Service to manage apps on the Kubernetes platform.

Before you begin

Ensure that you meet the following requirements.

- You have a compatible version of Kubernetes. Self-Service is compatible with Kubernetes 1.16 and 1.17.
- You have necessary RBAC permissions on the Kubernetes server.
- You have the authentication mechanism enabled on the Kubernetes cluster.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
The account inspector panel is displayed.
5. Click **+Add Account**.
The **Account Settings** page appears.
6. In the **Name** field, type a name for the account.
7. From the **Provider** dropdown menu, select **Kubernetes**.
8. From the **Type** dropdown menu, select one of the following.
 - » Select **Vanilla** to self deploy the kubernetes clusters.
 - » Select **Karbon** to add Nutanix Kubernetes Engine as the provider type. Nutanix Kubernetes Engine is a curated turnkey offering that provides simplified provisioning and operations of Kubernetes clusters.

9. If you have selected the kubernetes type as **Vanilla**, then do the following.
 - a. In the **Server IP** field, type the Kubernetes leader IP address.
 - b. In the **Port** field, type the port number of the Kubernetes server.
 - c. From the **Auth Type** dropdown menu, select the authentication type.

You can select one of the following authentication types.

 - **Basic Auth**: Basic authentication is a method for an HTTP user agent, for example, a web browser, to provide a user name and password when making a request.
 - **Client Certificate**: A client certificate is a digital certificate protected with a key for authentication.
 - **CA Certificate**: A client authentication certificate is a certificate that is used to authenticate clients during an SSL handshake. The certificate authenticates users who access a server by exchanging the client authentication certificate.
 - **Service Account**: A service account is an automatically enabled authenticator that uses signed bearer tokens to verify requests.
 - d. If you have selected **Basic Auth**, then do one of the following.
 - In the **Username** field, type the username. If the domain is a part of the username, then the username syntax should be <username>@<domain>.
 - In the **Password** field, type the password.
 - e. If you have selected **Client Certificate**, then do one of the following.
 - Under **Client Certificate**, upload the client certificate.
 - Under **Client Key**, upload the private key.
 - f. If you have selected **CA Certificate**, then do one of the following.
 - Under **CA Certificate**, upload the CA certificate.
 - Under **Client Certificate**, upload the client certificate.
 - Under **Client Key**, upload the private key.
 - g. If you selected **Service Account**, then do one of the following.
 - Under **Token**, upload the service account authentication token.
 - Under **CA Certificate**, upload the CA certificate.
10. If you have selected the kubernetes type as **Karbon**, then do the following.
 - a. In the **Cluster** dropdown menu, select the respective kubernetes cluster that you want to add.
11. Click **Save**.
12. To verify the credentials of the account, click **Verify**.

Self-Service lists the account as an active account after successful credential authentication and account verification.

Configuring Amazon EKS, Azure Kubernetes Service, Anthos, or Red Hat OpenShift

For Self-Service to manage workloads on Amazon EKS, Azure Kubernetes Service (AKS), Anthos or Red Hat OpenShift, enable the generic authentication mechanism and create a service account on the Kubernetes cluster. You can then use the service account to communicate with the cluster.

Procedure

1. Create a service account by running the following Kubernetes command:

```
$ kubectl create serviceaccount ntnx-calm
```

A service account is a user that the Kubernetes API manages. A service account is used to provide an identity for the processes that run in a pod.

2. Bind the cluster admin role to the Self-Service service account using the following command:

```
$ kubectl create clusterrolebinding ntnx-calm-admin --clusterrole cluster-admin --serviceaccount default:ntnx-calm
```

3. Get the service account secret name using the following command:

```
SECRET_NAME=$(kubectl get serviceaccount ntnx-calm -o jsonpath='{$.secrets[0].name}')
```

Secrets are objects that contain sensitive data such as a key, token, or password. Placing such information in a Secret allows better control and reduces the risk of exposure.

4. Get the service account token using the following command:

```
$ kubectl get secret ${SECRET_NAME} -o jsonpath='{$.data.token}' | base64 -decode
```

5. Get the CA certificate using the following command:

```
$ kubectl config view --minify --raw -o jsonpath='{.clusters[*].cluster.certificate-authority-data}' | base64 -decode
```

What to do next

After receiving the service token, you can add the account in Self-Service and use the token to communicate with the cluster. For more information on adding the account, see [Configuring a Kubernetes Account](#) on page 69.

Configuring a Nutanix Cloud Account

To manage workloads on Nutanix Cloud, add your Nutanix Cloud as an account in Self-Service if your Prism Central is paired with a Nutanix Cloud. Self-Service automatically discovers the availability zones of the Nutanix Cloud and allows you to add the Nutanix Cloud account as a provider account.

Before you begin

Ensure that you meet the following conditions.

- You enabled Nutanix DRaaS in Prism Central.
- Your Prism Central is paired with Nutanix Cloud.
- Your Prism Central and Nutanix Cloud are connected to a VPN.
- You added the routes to Nutanix Cloud gateway in Prism Central.

Procedure

1. Log in to your Prism Central instance as an administrator.

2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. Click the **Accounts** tab.
The account inspector panel appears.
5. Click **+Add Account**.
The **Account Settings** page appears.
6. In the **Name** field, type a name for the Nutanix Cloud.
7. From the **Provider** dropdown menu, select **Xi**.
Self-Service automatically add the paired availability zones in the **Availability Zones** field.
8. Click **Save**.
9. To verify the credentials of the account, click **Verify**.
Self-Service lists the account as an active account after successful credential authentication and account verification.

What to do next

You can use the configured Nutanix Cloud to host blueprints and app by using Self-Service. For more information, see [Self-Service Blueprints Overview](#) on page 87.

Platform Sync for Provider Accounts

Self-Service automates the provisioning and management of infrastructure resources for both private and public clouds. When any configuration changes are made directly to the Self-Service-managed resources, Self-Service needs to sync up the changes to accurately calculate and display quotas and Showback information.

Platform sync enables Self-Service to synchronize any changes in the clusters that are managed by Self-Service on connected providers. These changes can be any IP Address changes, disk resizing, unavailability of VMs, and so on.

For example, when a VM is powered off externally or deleted, platform sync updates the VM status in Self-Service. Self-Service then adds the infrastructure resources consumed by the VM (memory and vCPU) to the total available quota.

You can specify an interval after which the platform sync must run for a cluster. For more information, see [Configuring a Remote Prism Central Account](#) on page 53 and [Configuring a VMware Account](#) on page 58.

Note: Platform sync is supported for Nutanix, VMware, and AWS. Self-Service provides automatic platform sync for AWS with a predefined sync interval of 20 minutes. You can, however, sync up the configuration changes instantly for your Nutanix or VMware account. For more information, see [Synchronizing Platform Configuration Changes](#) on page 73.

Synchronizing Platform Configuration Changes

Platform sync enables Self-Service to synchronize any changes in the clusters that are managed by Self-Service on connected providers. These changes can be any IP Address changes, disk resizing, unavailability of VMs, and so on. You can sync up the configuration changes instantly for your accounts.

About this task

Note: Platform sync is supported for Nutanix, VMware, and AWS. Self-Service provides automatic platform sync for AWS with a predefined sync interval of 20 minutes. The following steps are applicable only to the Nutanix and VMware accounts.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **Accounts** tab, select the Nutanix or VMware account for which you want to sync up configuration changes in the left pane.
5. On the Account Settings page, click **Sync Now**.

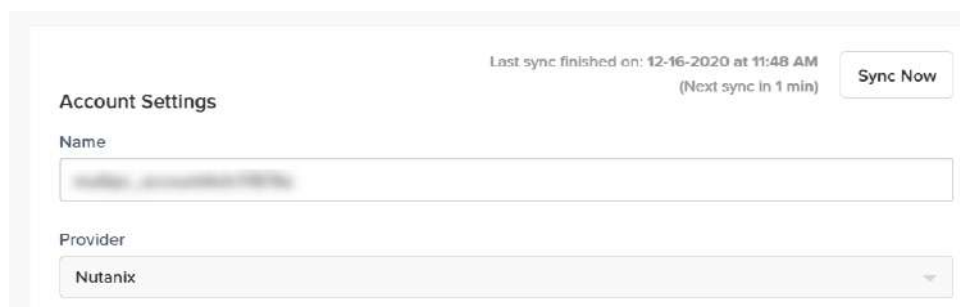


Figure 14: Sync Now

Allocating Resource Quota to an Account

Allocate resource quotas to your accounts to have a better control over the infrastructure resources (computer, memory, and storage) that are provisioned through Self-Service. Based on the resource quota you allocate, the policy engine enforces quota checks when apps are launched, scaled-out, or updated.

About this task

Note: You can allocate resource quotas to Nutanix and VMware accounts only.

Before you begin

- Ensure that you configured your Nutanix or VMware account. For more information on configuring an account, see [Provider Account Settings in Self-Service](#) on page 52.
- Ensure that you enabled the policy engine in Self-Service Settings. For more information on enabling the policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.

4. On the **Accounts** tab, select the Nutanix or VMware account in the left pane.
5. On the Account Settings page, select the **Quotas** checkbox.
If you have set up the quota defaults on the **Policies** tab (for information, see [Setting up Quota defaults](#) on page 45), the default values populate automatically in the **vCPU**, **Memory**, and **Disk** fields of the discovered clusters.

☒ **Quotas**

Set resource quotas for the selected clusters. (Empty field means no quota set.)

Cluster Name	vCPU	Memory (GiB)	Disk (GiB)
CLUSTER-NAME-1	<input type="text" value="200"/>	<input type="text" value="1000"/>	<input type="text" value="2000"/>
Quota Utilization View Report	<input type="range" value="0%"/> 0%	<input type="range" value="0%"/> 0%	<input type="range" value="0%"/> 0%
Physical Resources ⓘ	1715/33.58 GHz	76/126 GiB	148/3616 GiB

Figure 15: Quota Definition

6. Allocate required quota values to the discovered clusters.
The **Physical Resources** row below the quota fields shows the physical resource already used and the total physical resource of the cluster. You can use this information when you allocate resource quotas to the account.
7. Click **Save**.

Viewing Quota Utilization Report

Use the utilization report to analyze how the projects to which the cluster is assigned consumed the allocated resources of the cluster. For example, if a Nutanix cluster is assigned to three different projects, you can analyze how the assigned projects consumed the allocated resources of that cluster.

Before you begin

- Ensure that you have enabled the policy engine in Self-Service Settings. For more information on enabling the policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.
- Ensure that you have allocated resource quotas to the provider. For more information, see [Allocating Resource Quota to an Account](#) on page 74.
- Ensure that you have allocated resource quotas to projects. For more information, see [Adding Accounts to a Project](#).

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Self-Service Settings** in the navigation bar.
4. On the **Accounts** tab, select the Nutanix or VMware account in the left pane.
The **Account Settings** page appears.

5. In the **Quotas** section, do the following to view the resources consumed by each cluster.
 - a. In the **Quota Utilization | View Report** row, hover your mouse over the status bar of a resource. The ToolTip displays the resources consumed and the resources allocated to the cluster.



Figure 16: Quota Utilization Status Bar

Note: You can also use the status bar to view the overall status and percentage of resources consumed.

- b. Click **View Report**. The **Utilization Report** window appears.

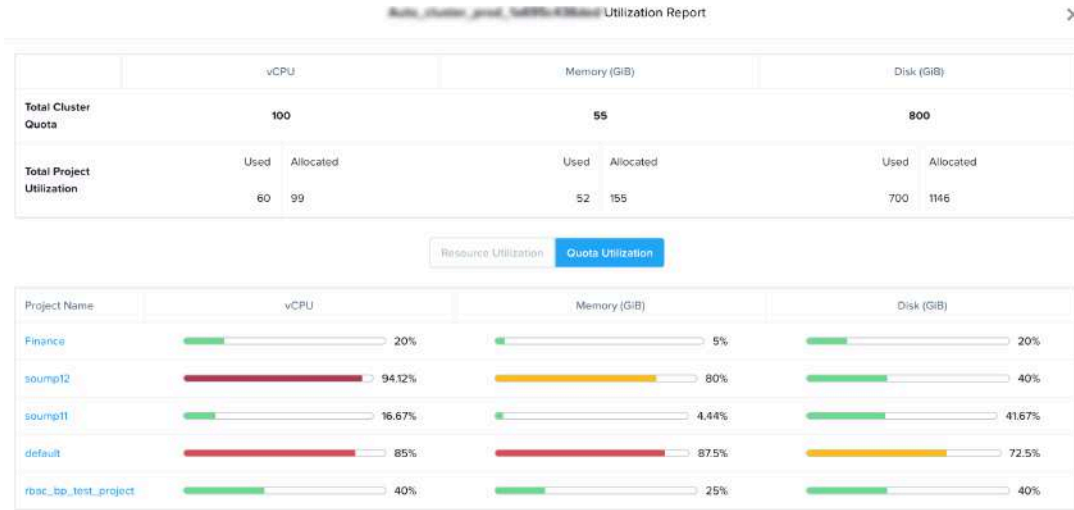


Figure 17: Utilization Report

- c. View project-wise consumption of resources along with the amount of compute, memory, and storage that the projects used.

Connecting Nutanix Database Service (NDB) to a Nutanix Account

A Nutanix service refers to a specific software component or functionality that Nutanix provides to enhance capabilities, performance, and management of the Nutanix infrastructure and software stack. For example, Nutanix Database Service (NDB).

About this task

Connect your NDB accounts to the Nutanix accounts in Self-Service to facilitate integration of NDB accounts with Self-Service runbooks.

Note:

- You can connect NDB accounts to the Nutanix accounts only.
- Self-Service supports integration of NDB version 2.5.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. In the Application Switcher, select **Self-Service**.
3. In the navigation bar, click **Self-Service Settings**.
4. On the **Accounts** tab, select the Nutanix account that you want to connect to your NDB accounts.
5. On the Account Settings page, scroll down to the Nutanix Service section, and then click **+ Connect a New Account**.
6. In the Connect NDB Account window, do the following to configure the connection:
 - a. In the **Name** field, specify a name for the NDB account that you want to connect to the Nutanix account.
 - b. In the **Server IP** field, type the IP address of your NDB account.
 - c. In the **Username** field, type the username that you use to access your NDB account.
 - d. In the **Password** field, type the password that you use to access your NDB account.
 - e. Click **Connect**.
7. On the Account Settings page, click **Verify** to verify the account connection.

What to do next

You can add the NDB accounts you connected to your Nutanix account in your project so that you can use the NDB accounts in runbooks. For more information, see [Adding Infrastructure in a Prism Central](#) in the *Prism Central Admin Center Guide*.

PROJECTS OVERVIEW

A project defines Active Directory users or groups to manage common set of requirements or functions. For example, a project can define a team collaborating on an engineering project. A project specifies roles to associate its members, select existing networks that the deployed VMs can use, and (optionally) set usage limits on infrastructure resources.

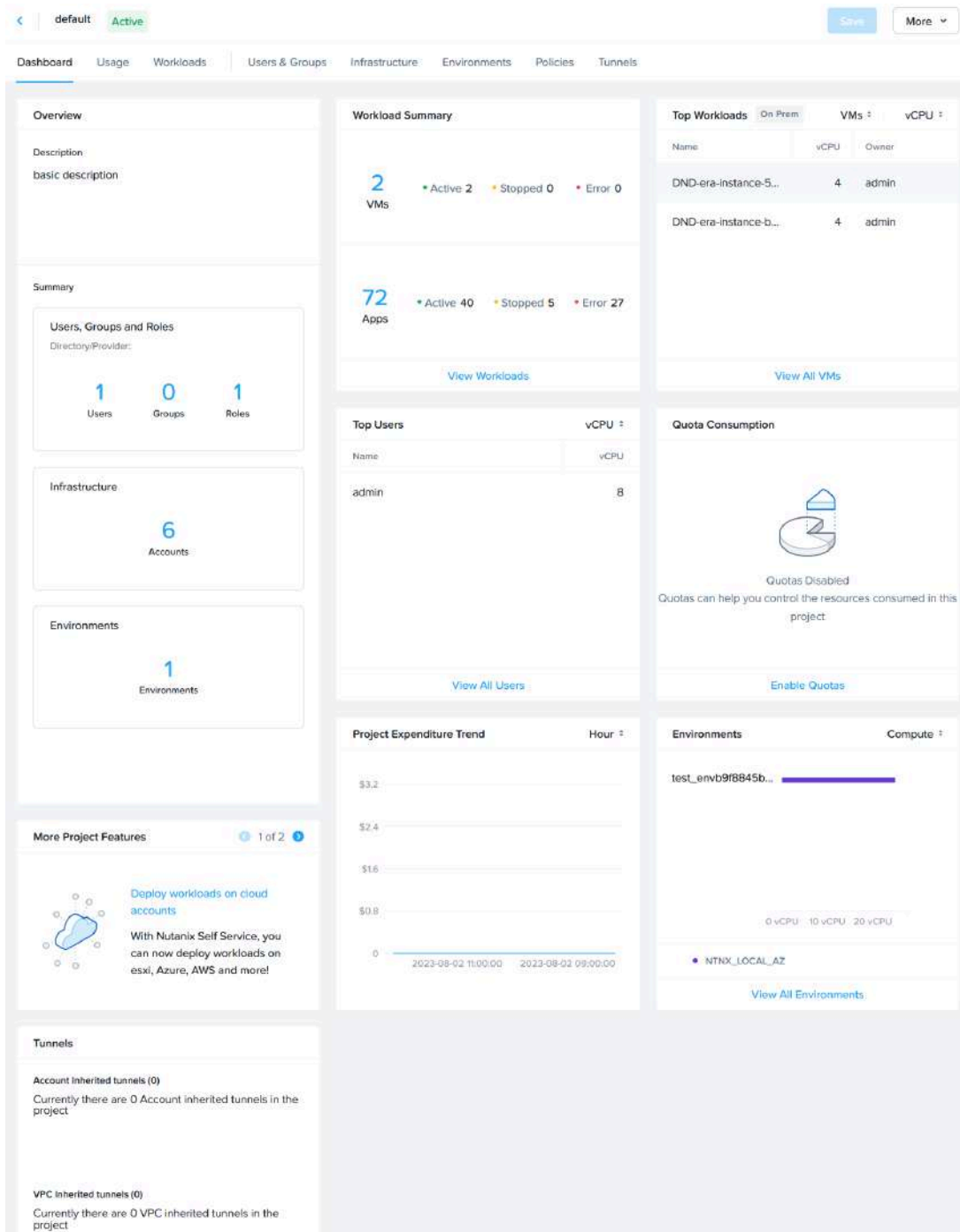


Figure 18: Projects

The refactored project provides a consistent experience when you access it from Prism Central or from Self-Service. However when Self-Service is enabled, you can also configure app management specific features in your projects.

For more information on how to create a project, add users, add infrastructure, configure environments, and managing quota and snapshot policies, see the [Project Management](#) section in the *Prism Central Admin Center Guide*.

ENVIRONMENTS IN SELF-SERVICE

An environment is a specific context or target infrastructure where you can deploy and manage applications and services. Environments encapsulate details about the infrastructure, configurations, and settings necessary for successful application deployment and management.

Here is how environments work in Self-Service.

- **Deployment Targets:** You configure environments to define platforms or infrastructure that you require to deploy your applications. Your environment can include public clouds such as AWS, Azure, GCP, private clouds built on Nutanix infrastructure, and so on. Environment configuration requires details such as credentials, network settings, security groups, and infrastructure-specific information that applications require for interaction.

Blueprint Adaptation: You can create a blueprint that includes environment-agnostic settings and then customize the blueprint for each specific environment.

Environment Variables: You define environment-specific variables such as URLs, credentials, API keys, and other dynamic values that the blueprints can use to adapt to different situations.

Application Deployment: When you launch a blueprint, you can select the target environment whose configuration and settings can be used to provision and manage applications on that specific infrastructure.

Life-cycle Management: Environments are central to the life-cycle management of applications. Self-Service uses environment-specific information to handle provisioning, scaling, updating, and eventually decommissioning applications.

Environment Configuration

When Self-Service is enabled in your Prism Central instance, you can configure environments as the subset of your project. To configure environments, you must add the accounts that you configured in Self-Service to your project and then configure environments using those accounts. You can configure environments for the following accounts:

- Nutanix. For more information, see [Configuring Nutanix Environment](#).
- VMware. For more information, see [Configuring VMware Environment](#).
- AWS. For more information, see [Configuring AWS Environment](#).
- Azure. For more information, see [Configuring Azure Environment](#).
- GCP. For more information, see [Configuring GCP Environment](#).

Environment Usage

You use a configured environment either during your blueprint creation or during an application launch.

When you launch a blueprint, you can select the target environment whose configuration and settings can be used to provision and manage applications on that specific infrastructure. When you select an environment that is different from the account that you used for blueprint configuration, Self-Service updates all platform-dependent fields to match with the selected environment configuration. For more information, see [Environment Patching Behavior](#) on page 81 and [Patching for Clusters and Subnets](#) on page 85.

You can deploy a blueprint from the Blueprints page. For more information, see [Launching a Blueprint](#) on page 224). You can also deploy a blueprint from the Marketplace. For more information, see [Deploying Custom Apps from the Marketplace](#) in the *Prism Central Admin Center Guide*).

Environment Patching Behavior

VM configurations in blueprints and environments are associated with accounts. The environment patching depends on the account that you associate with the app blueprint and the account in the environment you configured.

To patch a cloud provider VM that has a specific OS type, Prism Central finds the corresponding match in the environment. In case there are no matches available, Prism Central displays a notification.

The following table lists the environment patching behavior for platform-dependent and platform-independent fields:

Table 10: Environment Patching

Fields	Condition	Patching Behavior
Platform-Dependent Fields	When different accounts are associated with the blueprint and environment	Values from the environment get preference for patching, irrespective of the values in the blueprint. Note: Self-Service does not patch downloadable images from the environment.
	When the blueprint and the environment have the same account	Values from the environment are patched only when the fields do not have any value in the blueprint.
Platform-Independent Fields	When different accounts are associated with the blueprint and environment	Values from the environment are patched only when the fields do not have any value in the blueprint.
	When the blueprint and the environment have the same account	Values from the environment are patched only when the fields do not have any value in the blueprint.

Note: These behaviors do not apply to credentials. For more information, see the *Credential Patching* section in this topic.

The following table lists the platform-dependent fields for different platforms.

Table 11: Platform-Dependent Fields

Platform	Platform-Dependent Fields
Nutanix	Image, Categories, Cluster, and NIC
AWS	Machine Image, Key, Instance Profile Name, VPC ID, Subnet ID, and Security Group List

Platform	Platform-Dependent Fields
GCP	Machine Type, Zone, Network, Disk Type, Source Image, and Email
VMware	Host, Template, Datastore, Cluster, Storage Pod, Network Name, NIC Type, Disk Location, Disk ISO Path, Folder, and Tag List
Azure	Resource Group, Location, Availability Set ID, Resource Group Details, Resource Group Operation, Network Security Group Name, Network Name, Subnet Name, Network Security Group ID, Virtual Network ID, Subnet ID, Publisher, Offer, SKU, Version, Source Image Type, and Source Image ID

Environment Patching Behavior with Nutanix – Example-1

Assume that you have two Nutanix Prism Central accounts PC1 and PC2, and you added these accounts to your project (Project1). You then create two environments in the project with the following VM configuration:

Table 12: Environments

ENV1	ENV2
<ul style="list-style-type: none"> Account: PC1 NIC: PC1_Net1 Image: PC1_Image1 Categories: PC1_category1 and PC1_category2 Cluster: PC1_Cluster1 Operating System: Linux 	<ul style="list-style-type: none"> Account: PC2 NIC: PC2_Net1 Image: PC2_Image1 Categories: PC2_category1 and PC2_category2 Cluster: PC2_Cluster1 Operating System: Linux

You then create a blueprint with a Nutanix service under Project1 having the following configuration:

- Account: PC1
- Image: PC1_Image2
- Categories: PC1_category3
- Cluster: PC1_Cluster2
- NIC: PC1_Net2

When you publish this blueprint in the Marketplace and launch the blueprint with a different environment, the environment patching happens as follows:

- When you select Project1 and ENV2 for launching, the account in the blueprint is PC1, and the account in ENV2 is PC2.

Because different accounts are associated with the blueprint and environment, all platform-dependent field values are patched from the environment to the blueprint, irrespective of the values already available in the blueprint. The blueprint is launched with the following configuration.

- Image: PC2_Image1
 - Categories: PC2_category1 and PC2_category2
 - Cluster: PC2_Cluster1
 - NIC: PC2_Net1
- When you select Project1 and ENV1 for launching, the account in both the blueprint and ENV1 is PC1.
- Because the account is same for both blueprint and environment and all the platform-dependent fields already have values, the patching does not happen. The blueprint is launched with the following configuration.
- Image: PC1_Image2
 - Categories: PC1_category3
 - Cluster: PC1_Cluster2
 - NIC: PC1_Net2

Environment Patching Behavior with Nutanix – Example-2

Assume that you have a Prism Central account PC1 that is associated with two Prism Elements PE1 and PE2, and you add PC1 to your project (Project1).

Assume that the associated Prism Elements have the following networks.

- PE1: PE1_Net1 and PE1_Net2
- PE2: PE2_Net1 and PE2_Net2

You then create two environments with the following VM configuration:

Table 13: Environments

ENV1	ENV2
<ul style="list-style-type: none"> • NIC: PE1_Net1 • Image: PC1_Image1 • Categories: PC1_category1 and PC1_category2 • Operating System: Linux 	<ul style="list-style-type: none"> • NIC: PE2_Net1 • Image: PC1_Image2 • Categories: PC1_category3 and PC1_category4 • Operating System: Linux

You then create a blueprint with a Nutanix service under Project1 having the following configuration:

- NIC: PE1_Net2
- Image: PC1_Image3
- Categories: PC1_category5 and PC1_category6

When you publish this blueprint in the Marketplace and launch the blueprint with a different environment, the environment patching happens as follows:

- When you select Project1 and ENV2 for launching:

Prism Element accounts are derived from the NIC or subnet. The PE1_Net2 network used in the blueprint associates the blueprint to Prism Element PE1, and the PE2_Net1 network used in ENV2 associates the environment to Prism Element PE2.

Because these two networks are connected to two different Prism Element `account_uuid`, Prism Central considers this case as two different accounts associated with the blueprint and environment. All platform-dependent field values are, therefore, patched from the environment to the blueprint, irrespective of the values already available in the blueprint. The blueprint is launched with the following configuration.

- NIC: PE2_Net1
- Image: PC1_Image2
- Categories: PC1_category3 and PC1_category4

- When you select Project1 and ENV1 for launching:

The PE1_Net2 network used in the blueprint and the PE1_Net1 network used in ENV belong to the same Prism Element account.

Because these two networks share the same Prism Element `account_uuid`, Prism Central considers this case as the same account associated with both the blueprint and environment. Platform-dependent fields in this case already have values, and the patching does not happen. The blueprint is launched with the following configuration.

- NIC: PE1_Net2
- Image: PC1_Image3
- Categories: PC1_category5 and PC1_category6

Credentials Patching

Patching of credentials involve the following considerations:

- During the blueprint launch, when you select an environment, the credentials are patched from that environment irrespective of whether the account in that environment is different or the same as that of the blueprint.
- During the blueprint launch, if the environment has **All Project Accounts** selected, the credentials are patched from the blueprint.

For patching, the credentials of the Marketplace blueprint are mapped with the environment using the associated provider account and operating system type. The password or the key value of the corresponding environment is then patched to the blueprint. The credential name and the credential username are never patched from the environment.

For example, if the blueprint and the environment have the following configurations:

Table 14: VM Configuration

Blueprint	Environment
<ul style="list-style-type: none"> Credential Name: BP_Credentials Username: BP_User1 Password: BP_Password Provider Account: Nutanix Operating System: Linux 	<ul style="list-style-type: none"> Credential Name: ENV_Credentials Username: ENV_User1 Password: ENV_Password Provider Account: Nutanix Operating System: Linux

The credentials patching in the blueprint happens as follows:

Table 15: Credential Patching

When Blueprint is Published with Secrets	When Blueprint is Published without Secrets
<ul style="list-style-type: none"> Credential Name: BP_Credentials Username: BP_User1 Password: BP_Password 	<ul style="list-style-type: none"> Credential Name: BP_Credentials Username: BP_User1 Password: ENV_Password

Patching for Clusters and Subnets

The Cluster field is platform dependent. The environment patching logic of a platform-dependent field depends on the account that you associate with the Marketplace item and the VM configuration of the environment.

Table 16: Conditions for Cluster Patching

Condition	Patching Behavior
When the cluster reference in the blueprint and in the environment VM configuration is the same.	No patching happens. The cluster reference from the blueprint is used for the launch.
When the cluster reference in the blueprint and in the environment VM configuration is different.	Patching happens. The cluster value is patched from the environment for the launch.
When the cluster reference in the blueprint is a macro.	No patching happens. The cluster value will remain as a macro.
Note: Cluster reference can be a macro only when all the subnets are overlay subnets or all the subnets are macros.	When the reference is a macro, it is independent of the environment or the account that is being used for launch.

VLAN subnets are platform dependent. The environment patching logic of VLAN subnets depends on the cluster reference of the blueprint and the cluster reference of the associated environment VM configuration.

Overlay subnets are VPC dependent. The environment patching logic of these subnets depends on the VPC reference in the blueprint and the VPC reference of the associated environment VM configuration.

All subnets in the substrate of a blueprint can either have overlay subnets or VLAN subnets. If subnets are overlay subnets, then all the subnets in the substrate must belong to the same VPC.

Table 17: Conditions for Subnet Patching

Condition	Patching Behavior
When the VLAN subnets in the blueprint and in the environment VM configuration is the same.	No patching happens. VLAN subnets are platform dependent. The VLAN subnet values referred in the blueprint are used.
When the VLAN subnets in the blueprint and in the environment VM configuration is different.	Patching happens. VLAN subnets are platform dependent. The VLAN subnet values are patched from the environment.
When the VPC reference of the subnets (overlay subnets) in the blueprint and the environment VM configuration is the same.	No patching happens. The subnet values of the blueprint are used for the launch. Values from the environment is patched only if it is empty in the blueprint or not allowed in the destination environment.
When the VPC reference of the subnets (overlay subnets) in the blueprint and the environment VM configuration is different.	Patching happens. The subnet values are patched directly from the environment.
When the network type in the blueprint and the environment VM configuration are different (for example, overlay subnets in the blueprint and VLAN subnets in the environment).	Patching happens. The subnet values are patched directly from the environment.
When the subnet reference of the any of the NICs in the blueprint is a macro.	Patching follows the usual conditions. However, the macros are never patched.

SELF-SERVICE BLUEPRINTS OVERVIEW

A blueprint is the framework for every app that you model by using Self-Service. Blueprints are templates that describe all the steps that are required to provision, configure, and execute tasks on the services and apps that you create.

You create a blueprint to represent the architecture of your app and then run the blueprint repeatedly to create an instance, provision, and launch apps.

A blueprint also defines the life cycle of an app and its underlying infrastructure; starting from the creation of the app to the actions that are carried out on a blueprint until the termination of the app.

You can use blueprints to model the apps of various complexities; from simply provisioning a single virtual machine to provisioning and managing a multi-node, multi-tier app.

Building Blocks of a Blueprint

Self-Service uses services, app profiles, packages, substrates, and actions as building blocks for a blueprint to define apps.

- **Services**

An app is made up of multiple components (or services) working together. The architecture of an app is composed of compute, storage, network, and their connections and dependencies. Services are logical entities that are exposed by an IP address. End users and services communicate with each other over a network through their exposed IP addresses and ports. For more information, see [Services Overview](#) on page 90.

- **Application Profiles**

Any useful blueprint requires infrastructure for instantiation. A blueprint can specify the exact infrastructure or can be completely left to the blueprint user to specify at the time of instantiation.

An app profile provides different combinations of the service, package, and VM (infrastructure choices) while configuring a blueprint. The app profile allows you to use the same set of services and packages on the different platforms. You select an app profile while launching your blueprint.

Application profiles determine where an app should run, for example, on a Nutanix provider account or on an Azure account. Application profiles also control the T-shirt sizing of an app. T-shirt sizing means that the value of a variable might change based on the selection of a small or a large instance of an app.

If Showback feature is enabled, the app profile also displays service cost of the resources used for an app.

- **Package (Install and Uninstall)**

Package Install and Uninstall are operations that are run when you first launch a blueprint or when you finally delete the entire app. In other words, these operations are run during the Create or Delete profile actions. Package Install and Uninstall are unique to each app profile, which means that the tasks or the task contents can vary depending upon the underlying cloud or the size.

Package install is commonly used for installing software packages. For example, installing PostgreSQL with `sudo yum -y install postgresql-server postgresql-contrib`.

- **Substrates**

Substrates are a combination of the underlying cloud and the virtual machine instance. When you select the desired cloud, Self-Service displays all of the fields required for creating a virtual machine instance on that particular cloud. The combination of all these fields constitutes a substrate. Substrates are the infrastructure abstraction layer for Self-Service. Self-Service can quickly change where or how apps are deployed by simply changing the substrate.

- **Actions**

Actions are runbooks to accomplish a particular task on your app. You can use actions to automate any process such as backup, upgrade, new user creation, or clean-up, and enforce an order of operations across services. For more information, see [Actions Overview](#) on page 192.

Other Configuration-Specific Components

Self-Service also has a few other components that you can use while configuring your blueprints.

- **Macros**

Self-Service macros are part of a templating language for Self-Service scripts. These are evaluated by Self-Service's execution engine before the script is run. Macros help in making scripts generic and creating reusable workflows. For more information, see [Macros Overview](#) on page 91.

- **Variables**

Variables are either user defined or added to the entities by Self-Service. Variables are always present within the context of a Self-Service entity and are accessible directly in scripts running on that entity or any of its child entities. For more information, see [Variables Overview](#) on page 99.

- **Categories**

Categories (or tags) are metadata labels that you assign to your cloud resources to categorize them for cost allocation, reporting, compliance, security, and so on. Each category is a combination of key and values. For more information, see [Categories Overview](#) on page 104.

- **Dependencies**

Dependencies are used to define the dependence of one service in your app on another service or multiple other services for properties such as IP addresses and DNS names. For example, if service 2 is dependent on service 1, then service 1 starts first and stops after service 2.

For information on how to define dependencies between services, see [Setting up the Service Dependencies](#) on page 165.

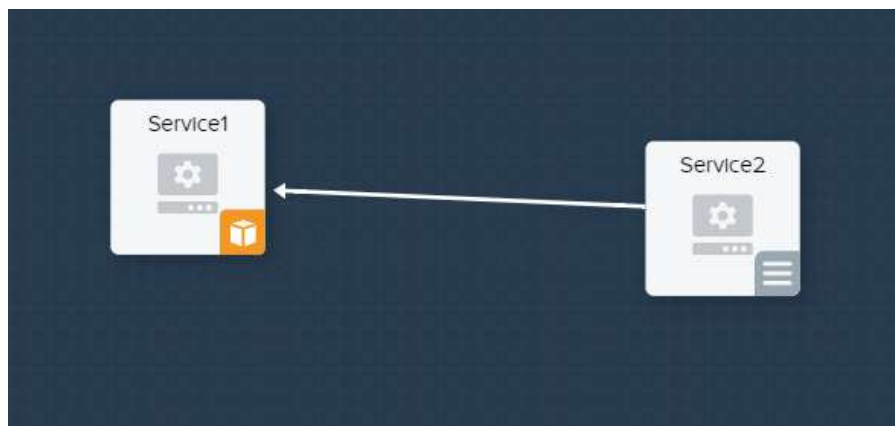


Figure 19: Dependencies

Note: If there are no dependencies between tasks in a service, Self-Service runs the tasks in any order or even in parallel.

Blueprint Types

You can configure the following blueprint types in Self-Service.

- **Single-VM Blueprint**

A single-VM blueprint is a framework that you can use to create and provision an instance and launch apps that require only one virtual machine. Single-VM blueprints enable you to quickly provide Infrastructure-as-a-Service (IaaS) to your end users. For more information, see [Creating a Single-VM Blueprint](#) on page 106.

- **Multi-VM Blueprint**

A multi-VM blueprint is a framework that you can use to create an instance, provision, and launch apps requiring multiple VMs. You can define the underlying infrastructure of the VMs, app details, and actions that are carried out on a blueprint until the termination of the app. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.

Blueprint Editor

The blueprint editor provides a graphical representation of various components that allow you to visualize and configure the components and their dependencies in your environment.

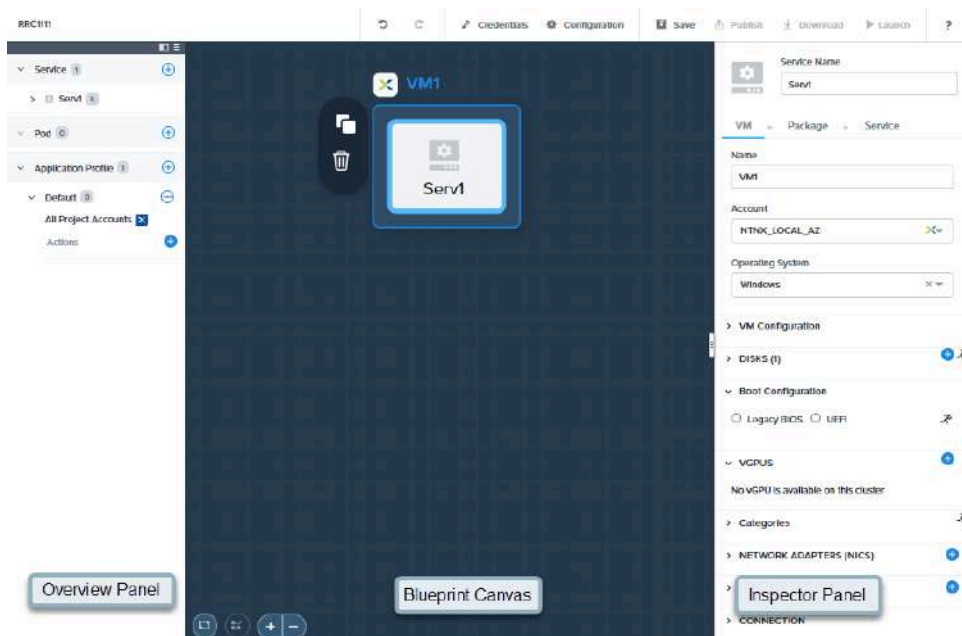


Figure 20: Blueprint Editor

Use the Blueprints tab to perform actions, such as:

- Create app blueprints for single-VM or multi-VM architectures. For more information, see [Creating a Single-VM Blueprint](#) on page 106 and [Creating a Multi-VM Blueprint](#) on page 130.
- Add update configuration. For more information, see [Update Configuration for VM](#) on page 205.
- Add configuration for snapshots and restore. For more information, see [Blueprint Configuration for Snapshots and Restore](#) on page 199.
- Publish blueprints. For more information, see [Submitting a Blueprint for Approval](#) on page 222.
- Launch blueprints. For more information, see [Launching a Blueprint](#) on page 224.
- Upload existing blueprints from your local machine. For more information, see [Uploading a Blueprint](#) on page 226.

- View details of your blueprints. For more information, see [Viewing a Blueprint](#) on page 227.
- Edit details of an existing blueprint. For more information, see [Editing a Blueprint](#) on page 228.

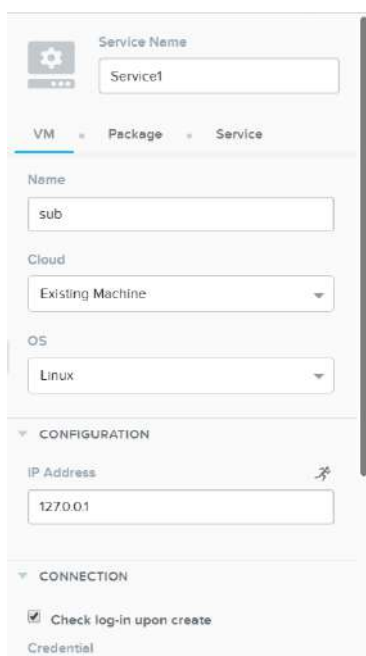
Services Overview

Services are the virtual machine instances, existing machines or bare-metal machines, that you can provision and configure by using Self-Service. You can either provision a single service instance or multiple services based on the topology of your app. A service can only expose an IP address and ports on which the request is received. After a service is configured, you can clone or edit the service as required.

A service includes the following entities:

VM

A VM defines the configuration of the virtual machine instance, the platform on which the VM will be installed, and the connection information of the machine. For example, as shown in the following figure, you need to define the name, cloud, operating system, IP address, and the connection information for an existing machine.



The screenshot shows a web interface for configuring a service. At the top, there's a 'Service Name' field with the value 'Service1'. Below this are three tabs: 'VM' (selected), 'Package', and 'Service'. The 'VM' tab contains several configuration fields: 'Name' with the value 'sub', 'Cloud' with a dropdown menu showing 'Existing Machine', and 'OS' with a dropdown menu showing 'Linux'. Below these is a section titled 'CONFIGURATION' with an 'IP Address' field containing '127.0.0.1'. At the bottom is a section titled 'CONNECTION' with a checkbox labeled 'Check log-in upon create' which is checked, and a 'Credential' field.

Figure 21: VM Tab

Package

A package enables you to install and uninstall software on an existing machine or bare metal machine by using a script. You need to provide the credentials of the VM on which you need to run the script. A sample script is shown in the following figure. Package also defines the port number and the protocol that is used to access the service.

The screenshot shows the 'Package' tab of a configuration interface. At the top, there is a 'Service Name' field with the value 'Service1'. Below this, there are three tabs: 'VM', 'Package', and 'Service', with 'Package' being the active tab. Under the 'Package' tab, there is a 'Package Name' field with the value 'Package1'. At the bottom of the tab, there are two buttons: 'Configure install' and 'Configure uninstall'.

Figure 22: Package Tab

Service

A service enables you to create the variables that are used to define the service-level tasks and service-level actions. As part of the service, you can also define the number of replicas that you want to create of a service. The maximum number of replicas allowed is 300.

The screenshot shows the 'Service' tab of a configuration interface. At the top, there is a 'Service Name' field with the value 'Service1'. Below this, there are three tabs: 'VM', 'Package', and 'Service', with 'Service' being the active tab. Under the 'Service' tab, there is a section titled 'DEPLOYMENT CONFIG' which contains a text box stating 'Replicas will be named as "Service_name[0]", "Service_name[1]" and so on.' Below this, there is a 'Number of replicas' field with the value '1'. Below the 'DEPLOYMENT CONFIG' section, there is a section titled 'VARIABLES' which contains a table with columns 'Name' and 'Value'. The table has one row with 'gv' in the 'Name' column and 'Secret Value' in the 'Value' column. Below the table, there are checkboxes for 'Secret' and 'Runtime', both of which are checked. Below the checkboxes, there are two empty input fields. At the bottom of the tab, there is a section titled 'PORT LIST' which contains a table with columns 'Name' and 'Value'. The table is currently empty.

Figure 23: Service Tab

For information on how to configure a service, see [Configuring Nutanix and Existing Machine VM, Package, and Service](#) on page 132.

Macros Overview

Self-Service macros are part of a templating language for Self-Service scripts. These are evaluated by Self-Service's execution engine before the script is run.

Macros enable you to access the value of variables and properties that are set on entities. The variables can be user defined or system generated. For more information, see [Variables Overview](#) on page 99.

Macro Usage

Macros help in making scripts generic and creating reusable workflow. You can use macros in tasks within the blueprints or in the configuration of Self-Service entities, such as the VM name.

Macro Syntax

Macros require a set of delimiters for evaluation. These are `@@{` and `}@@`. Everything within these delimiters is parsed and evaluated. For example,

- To concatenate the value of a path and a string variable, you can use `cd "@@{path + '/data'}@@"` in your script.
- To access credentials, you can use the `@@{cred_name.username}@@` and `@@{cred_name.secret}@@` formats, where `cred_name` is the name of the credential with which the credential is created.

Supported Entities

Macros support the following entities.

- Application
- Deployment
- Service
- Package
- Virtual machine
- Runbooks

Note: The scoping of macros across entities follows the property inheritance rules similar to variables. For more information on the variable value inheritance, see [Variables Overview](#) on page 99.

Supported Data Types

Macros support the following data types.

Table 18: Supported Data Types

Data Type	Usage
String	<code>@@{"some string"}@@</code> or <code>@@{'some string'}@@</code> Note: Newline or other such special characters are not supported. You can use <code>\</code> to escape quotes.
Numbers	Supports integer and float. For example, <code>@@{ 10 + 20.63 }@@</code> Note: All variables are treated as strings.

Supported Operations

Macros support the following operations.

- Supports basic binary operations or numbers. For example, `@@{(2 * calm_int(variable1) + 10) / 32 }@@`.
- Supports string concatenation. For example, `@@{ foo + bar }@@`.
- Supports slicing for strings. For example, `@@{foo[3:6]}@@`.

Note: For a comma separated value, slicing splits the string on comma (.). For example, `@@{"x,y,z"[1]}@@` results in y.

Macros of an Array Service

Self-Service allows you to access macros of an array service using a special macro which starts with `calm_array`. You can configure a VM with replicas and access the common macros of all the replicas. For example, you can:

- Use the following syntax to retrieve the name of all the instances of VM separated by commas.
`@@ { calm_array_name } @@`
- Use the following syntax to retrieve the IP address of all the instances of VM separated by commas.
`@@{calm_array_address}@@`
- Use the following syntax to retrieve the ID of all the instances of VM separated by commas.
`@@{calm_array_id}@@`

Built-in Macros

The following table lists the built-in macros that you can use to retrieve and display the entities.

Table 19: Built-in Macros

Macro	Usage
<code>@@ { calm_array_index } @@</code>	Index of the entity within an array.
<code>@@ { calm_blueprint_name } @@</code>	Name of the blueprint from which the app was created.
<code>@@ { calm_blueprint_uuid } @@</code>	Universally unique identifier (UUID) of the blueprint from which the app was created.
<code>@@ { calm_application_name } @@</code>	Name of the app.
<code>@@ { calm_application_uuid } @@</code>	UUID of the app.
<code>@@ { calm_uuid } @@</code>	UUID of the entity within the app on which the current task is running.
<code>@@ { calm_random } @@</code>	A random number is generated each time this is used. This will be evaluated each time and should not be used in fields such as VM name.
<code>@@ { calm_unique } @@</code>	A random number that is unique to this replica. This will be evaluated to the same value across runs.
<code>@@ { calm_jwt } @@</code>	JWT for the currently logged in user for API authentication.

Macro	Usage
<code>@@{calm_now}@@</code>	The current time stamp.
<code>@@{calm_today}@@</code>	
<code>@@{calm_time("<format>")}@@</code>	The current time in the specified format.
<code>@@{calm_year("YYYY")}@@</code>	The current year in YYYY or YY format.
<code>@@{calm_year("YY")}@@</code>	
<code>@@{calm_month("short")}@@</code>	Name of the current month in long or short format.
<code>@@{calm_month("long")}@@</code>	
<code>@@{calm_day("month")}@@</code>	Numeric day of the month or year.
<code>@@{calm_day("year")}@@</code>	
<code>@@{calm_weeknumber}@@</code>	ISO Numeric week of the year.
<code>@@{calm_weeknumber("iso")}@@</code>	
<code>@@{calm_weekday("number")}@@</code>	Day of the week in numeric or short name or long name.
<code>@@{calm_weekday("name_short")}@@</code>	
<code>@@{calm_weekday("name_long")}@@</code>	
<code>@@{calm_hour("12")}@@</code>	Numeric hour of the day in 12:00-hour or 24:00-hour format along with AM or PM.
<code>@@{calm_hour("24")}@@</code>	
<code>@@{calm_hour("am_pm")}@@</code>	
<code>@@{calm_minute}@@</code>	Numeric minute.
<code>@@{calm_second}@@</code>	Numeric second.
<code>@@{calm_is_weekday}@@</code>	Displays 1 if the current day is a weekday.
<code>@@{calm_is_long_weekday}@@</code>	Displays 1 if the current day is a weekday from Monday to Saturday.
<code>@@{calm_is_within("time1", "time2")}@@</code>	Displays 1 if the current time is within the time1 and time2 range.
<code>@@{calm_project_name}@@</code>	Displays the project name.
<code>@@{calm_username + @nutanix.com}@@</code>	Displays the username.
<code>@@{calm_float("32.65") * 2}@@</code>	Typecast to integer. This is useful for binary operations.
<code>@@{calm_int(calm_array_index) + 1}@@</code>	

Macro	Usage
<code>@@{calm_string(256) + "-bit"}@@</code>	Typecast to string. This is useful for string concatenation.
<code>@@{"xyz" + calm_string(42)}@@</code>	
<code>@@{calm_b64encode(api_response)}@@</code>	Base64 encode the data passed to this macro.
<code>@@{calm_b64encode("a,b,c")}@@</code>	
<code>@@{calm_b64decode(b64_encoded_data)}@@</code>	Base64 decode the data passed to this macro.
<code>@@{calm_b64decode("YSxiLGM=")}@@</code>	

Note: In day-2 action dynamic variables:

- Service variables as macros are not supported but profile variables as macros are supported.
- Only the following built-in macros (in the form of `calm_*`) are supported:
 - `@@{calm_application_name}@@`
 - `@@{calm_blueprint_name}@@` (in this case, the blueprint corresponds to the source blueprint of the application)
 - `@@{calm_blueprint_uuid}@@` (in this case, the blueprint corresponds to the source blueprint of the application)
 - `@@{calm_project_name}@@`
 - `@@{calm_username}@@`
 - `@@{calm_jwt}@@`

Platform Macros

You can access the properties of a VM by using the platform macros. The following section describes the macros to access the VM properties for different providers.

Table 20: AHV platform Macros

Macro	Usage
<code>@@{platform}@@</code>	To access all the properties of a VM.
<code>@@{platform.status.cluster_reference.uuid}@@</code>	To access the uuid of the cluster or the Prism element.

Macro	Usage
<code>@@{platform.status.resources.nic_list[0].mac_address}@@</code>	To access mac the address.
Note: Use the <code>nic_list</code> index to access the mac address of a specific nic.	
<code>@@{platform.status.resources.nic_list[0].subnet_reference.name}@@</code>	To access the NIC name.
<code>@@{platform.status.resources.power_state}@@</code>	To get the state of the VM.
<code>@@{platform.status.num_sockets}@@</code>	To access number of sockets of the VM.
Note: The <code>@@{platform}@@</code> macro stores the GET response of the VM. You can access any VM information that is available through the GET API response.	

Table 21: VMware platform Macros

Macro	Usage
<code>@@{platform}@@</code>	To access all the properties of a VM.
<code>@@{platform.datastore[0].Name}@@</code>	To access the datastore name.
<code>@@{platform.num_sockets}@@</code>	To access number of sockets of the VM.
Note: The <code>@@{platform}@@</code> macro stores the GET response of the VM. You can access any VM information that is available through the GET API response.	

Table 22: GCP platform Macros

Macro	Usage
<code>@@{platform}@@</code>	To access all the properties of a VM.
<code>@@{platform.creationTimestamp}@@</code>	To get the VM creation time stamp.
<code>@@{platform.selfLink}@@</code>	To access the self link of the VM.
<code>@@{platform.networkInterfaces[0].subnetwork}@@</code>	To access the network details of the VM.
Note: The <code>@@{platform}@@</code> macro stores the GET response of the VM. You can access any VM information that is available through the GET API response.	

Endpoint Macros

The following table lists the endpoint macros for HTTP, Linux, and Windows endpoint types.

Table 23: HTTP

Macro	Usage
@@{endpoint.name}@@	Name of the endpoint
@@{endpoint.type}@@	Type of the endpoint
@@{endpoint.length}@@	Number of IP Addresses in the endpoint
@@{endpoint.index}@@	Index of the IP address or VM in a given endpoint
@@{endpoint.base_url}@@	Base URL of the HTTP endpoint
@@{endpoint.connection_timeout}@@	Time interval in seconds after which the connection attempt to the endpoint stops
@@{endpoint.retry_count}@@	Number of attempts the system performs to create a task after each failure
@@{endpoint.retry_interval}@@	Time interval in seconds for each retry if the task fails
@@{endpoint.tls_verify}@@	Verification for the URL of the HTTP endpoint with a TLS certificate
@@{endpoint.proxy_type}@@	HTTP(s) proxy/SOCKS5 proxy to use
@@{endpoint.base_urls}@@	Base URLs of HTTP endpoints
@@{endpoint.authentication_type}@@	Authentication method to connect to an HTTP endpoint: Basic or None
@@{endpoint.credential.username}@@	User name in the credential to access the endpoint
@@{endpoint.credential.secret}@@	Credential secret type to access the endpoint: Passphrase or SSH Private Key

Table 24: Linux

Macro	Usage
@@{endpoint.name}@@	Name of the endpoint
@@{endpoint.type}@@	Type of the endpoint
@@{endpoint.length}@@	Number of IP Addresses in the endpoint
@@{endpoint.index}@@	Index of the IP address or VM in a given endpoint
@@{endpoint.address}@@	IP address to access the endpoint device
@@{endpoint.port}@@	Port number to access the endpoint
@@{endpoint.value_type}@@	Target type of the endpoint: IP address or VM
@@{endpoint.addresses}@@	IP addresses to access endpoint devices
@@{endpoint.credential.secret}@@	Credential secret type to access the endpoint: Passphrase or SSH Private Key
@@{endpoint.credential.username}@@	User name in the credential to access the endpoint

Table 25: Windows

Macro	Usage
<code>@@{endpoint.name}@@</code>	Name of the endpoint
<code>@@{endpoint.type}@@</code>	Type of the endpoint
<code>@@{endpoint.length}@@</code>	Number of IP Addresses in the endpoint
<code>@@{endpoint.index}@@</code>	Index of the IP address or VM in a given endpoint
<code>@@{endpoint.address}@@</code>	IP address to access the endpoint device
<code>@@{endpoint.port}@@</code>	Port number to access the endpoint
<code>@@{endpoint.value_type}@@</code>	Target type of the endpoint: IP address or VM
<code>@@{endpoint.connection_protocol}@@</code>	Connection protocol to access the endpoint: HTTP or HTTPS
<code>@@{endpoint.addresses}@@</code>	IP addresses to access endpoint devices
<code>@@{endpoint.credential.secret}@@</code>	Credential secret type to access the endpoint: Passphrase or SSH Private Key
<code>@@{endpoint.credential.username}@@</code>	User name in the credential to access the endpoint

Note: To call an endpoint variable from another object, replace `endpoint` with the other endpoint name.

Runbook Macros

The following table lists the runbook macros.

Table 26: Runbook Macros

Macro	Usage
<code>@@{calm_runbook_name}@@</code>	Name of the runbook
<code>@@{calm_runbook_uuid}@@</code>	Universally unique identifier (UUID) of the runbook

Virtual Machine Common Properties

The following table lists the common properties of the virtual machine that are available for usage.

Table 27: Virtual Machine Common Properties

Properties	Usage
<code>@@{address}@@</code>	IP address of the instance that is used by Self-Service to access the VM
<code>@@{id}@@</code>	ID of the platform identifier
<code>@@{name}@@</code>	Name of the VM or container

Properties	Usage
<code>@@{mac_address}@@</code>	Mac address of the VM
<code>@@{platform}@@</code>	Platform response for a GET query. This is the response in JSON format from provider.

Note: For an existing machine, only the address property is applicable.

Variables Overview

Macros provide a way to access the values of variables that you set on entities. Variables are either user defined or added to the entities by Self-Service. Variables are always present within the context of a Self-Service entity and are accessible directly in scripts running on that entity or any of its child entities.

Variable Value Inheritance

The variable value of a parent entity can be accessed by the child entity unless the properties or the variables are overridden by another entity.

For example, if Variable1 is a variable that you defined on the app profile, then all child entity of the app profile can directly access the value of Variable1 in any task or script running on it as `@@{variable1}@@` unless overridden by another entity.

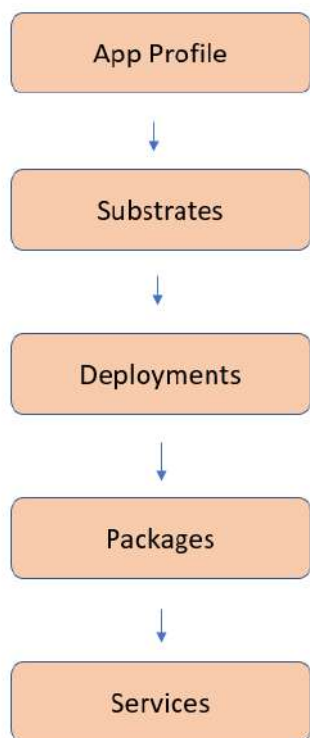


Figure 24: Variable Value Inheritance

Variable Access

Variables are directly accessed as `@@{variable_name}@@` within any task on an entity where the variable is defined and all child entity that inherit this variable. This syntax only delivers the value for the

corresponding replica in which the task is running. To get comma-separated values across replicas, you can use `@@{calm_array_variable_name}@@`.

For example, on a service with 2 replicas, if you set a `backup_dir` variable through a set variable Escript task such as:

```
print "backup_dir=/tmp/backup_@@{calm_array_index}@"
```

You get `/tmp/backup_0` and `/tmp/backup_1` values for replica 0 and 1 respectively.

When a task runs on this service with the `echo "@@{backup_dir}@"` script, the script evaluates the following values in each replica of the service:

- Replica 0
`/tmp/backup_0`
- Replica 1
`/tmp/backup_1`

When you change the script to `echo "@@{calm_array_backup_dir}@"`, the script evaluates to the following values in each replica of the service:

- Replica 0
`/tmp/backup_0,/tmp/backup_1`
- Replica 1
`/tmp/backup_0,/tmp/backup_1`

The syntax to access the value of variables or properties of other entities or dependencies is `@@{<entity name>.<variable/attribute name>}@@` where entity name, is the name of the other entity or dependency and variable/attribute name is the name of the variable or attribute. For example:

- Example 1: If a blueprint contains a service by the name of `app_container`, you can access the IP address of the `app_container` service in any other service using `@@{app_container.address}@@` syntax.
- Example 2: If you need addresses of a service (S1) in a task on another service (S2), you can use `@@{S1.address}@@` in the script for the task running on S2. The script will evaluate to the value, such as `10.0.0.3,10.0.0.4,10.0.0.5` in case S1 has 3 replicas.

Action-Level Variables

Action-level variables are variables that are associated to an action and passed as an argument to the runlog when you run the action. Service action variables are unique for each service while the profile action variables are unique for each profile across all services and replicas. If you deploy five replicas, the service action variables will be the same across all replicas.

Action variables are used in the context of running an action and are defined at the action level. For example, if you have an action to install or uninstall a package on a particular VM, you can have the following action variables.

- Type of action (in this case install or uninstall)
- Name of the package

With multiple runs of this action, you can then install or uninstall multiple packages on the VM.

Variable Conditions

- User-defined variables in Self-Service cannot have macros in their values.

- The values of a variable cannot be resolved if it has a dependency on another variable that is marked as Secret.
- If the same variable name with different values is defined across services, packages, actions, and profiles then the value of service variable is fetched for all the other variables.

Nutanix Variables

The following table lists the Nutanix variables that are available for usage.

Table 28: Nutanix Variables

Variables	Usage
@@{address}@@	IP address of the instance that is used by Self-Service to access the VM
@@{id}@@	ID of the platform identifier
@@{name}@@	Name of the VM or container
@@{mac_address}@@	Mac address of the VM
@@{platform}@@	Platform response for a GET query. This is the response in JSON format from provider.

Note: For an existing machine, only the address property is applicable.

VMware Variables

The following table lists the built-in VMware macros that you can use to retrieve and display the entities.

Table 29: VMware Macros

Properties	Usage
@@{address}@@	IP address of the instance that is used by Self-Service to access the VM
@@{id}@@	ID of the platform identifier
@@{name}@@	Name of the VM or container
@@{mac_address}@@	Mac address of the VM
@@{platform}@@	Platform response for a GET query. This is the response in JSON format from provider.

Note: For an existing machine, only the address property is applicable.

AWS Variables

The following table lists the built-in AWS macros that you can use to retrieve and display the entities.

Table 30: AWS Macros

Macros	Usage
<code>@@{address}@@</code>	IP address of the instance that is used by Self-Service to access the VM. Note: The VM Name field does not support this macro.
<code>@@{id}@@</code>	Internal ID of the instance that is used within the Prism. Note: The VM Name field does not support this macro.
<code>@@{name}@@</code>	Name of the VM. Note: The VM Name field does not support this macro.
<code>@@{aws_instance_id}@@</code>	Instance ID of AWS
<code>@@{private_ip_address}@@</code>	Private IP address
<code>@@{private_dns_name}@@</code>	Private DNS name
<code>@@{public_ip_address}@@</code>	Public IP address
<code>@@{public_dns_name}@@</code>	Public DNS name
<code>@@{vm_zone}@@</code>	AWS zone of instance
<code>@@{platform}@@</code>	Platform response for a GET query. This is the response in JSON format from provider.

GCP Variables

The following table lists the built-in GCP macros that you can use to retrieve and display the entities.

Table 31: GCP Macros

Macros	Usage
<code>@@{address}@@</code>	IP address of the instance that is used by Self-Service to access the VM. Note: The VM Name field does not support this macro.
<code>@@{ip_address}@@</code>	
<code>@@{public_ip_address}@@</code>	
<code>@@{id}@@</code>	Internal ID of the instance that is used within the Prism. Note: The VM Name field does not support this macro.

Macros	Usage
<code>@@{name}@@</code>	Name of the VM. Note: The VM Name field does not support this macro.
<code>@@{zone}@@</code>	Zone in which the VM instance is created.
<code>@@{platform_data}@@</code>	Platform response for a GET query. This is the response in JSON format from provider.
<code>@@{internal_ips}@@</code>	List of all the private IP addresses.
<code>@@{external_ips}@@</code>	List of all the public IP addresses.

Azure Variables

The following table lists the built-in Azure macros that you can use to retrieve and display the entities.

Table 32: Azure Macros

Macros	Usage
<code>@@{address}@@</code>	IP address of the instance that is used by Self-Service to access the VM. Note: The VM Name field does not support this macro.
<code>@@{id}@@</code>	Internal ID of the instance that is used within the Prism. Note: The VM Name field does not support this macro.
<code>@@{name}@@</code>	Name of the VM. Note: The VM Name field does not support this macro.
<code>@@{private_ip_address}@@</code>	Private IP address
<code>@@{public_ip_address}@@</code>	Public IP address
<code>@@{resource_group}@@</code>	Resource group name in which the VM instance is created.
<code>@@{platform_data}@@</code>	Platform response for a GET query. This is the response in JSON format from provider.

Kubernetes Variables

The following table lists the Kubernetes variables that are available for usage.

Table 33: Kubernetes Variables

Properties	Usage
@@ {K8sPublishedService.address} @@	IP address of the service.
@@ {K8sPublishedService.name} @@	Name of the service.
@@ {K8sPublishedService.ingress} @@	Load balancer IP for public service.
@@ {K8sPublishedService.platform} @@	Platform data for the service.
@@ {K8sDeployment.name} @@	Name of the deployment.
@@ {K8sDeployment.platform} @@	Platform data for the deployment.

Note: Do not use deployment macros in publish service specs or publish macros in deployment service specs in the same Self-Service deployment.

Runtime Variables Overview

Runtime variables are used to mark the attributes while creating the blueprint so that those attributes can be modified at the time of launching the app blueprint. This is useful for the users who cannot edit or create a blueprint such as consumers. For example, while creating a blueprint, if memory attribute is marked as a runtime variable then you can change its value before launching the app blueprint.

Note: Ensure that the attributes marked as runtime variable are not null or empty and an initial value is configured.

The screenshot shows a 'VM CONFIGURATION' form. The 'VM Name' field contains 'test_ahv'. Under the 'IMAGES (1)' section, the 'Image' dropdown is set to 'Centos7HadoopMaster'. The 'Device Type' is 'DISK' and the 'Device Bus' is 'SCSI'. The 'Bootable' checkbox is checked. At the bottom, the 'vCPUs' field is '1' and 'Cores per vCPU' is '1'. The 'Memory (GiB)' field contains '2' and has a red box around its runtime variable icon (a blue square with a white lightning bolt).

Figure 25: Runtime Variable

Categories Overview

Categories (or tags) are metadata labels that you assign to your cloud resources to categorize them for cost allocation, reporting, compliance, security, and so on. Each category is a combination of key and values.

Your providers impose a limit to the number of tags that you can use for cloud governance. The following table lists the category or tag limit imposed by each provider:

Table 34: Tag or Category Limit

Providers	Category or Tag Limit
Nutanix	30
AWS	50
VMware	No limit
GCP	15
Azure	15

Self-Service reserves 6 tags out of the total tags allowed by your provider and populates them automatically when you provision your VMs using Self-Service. For example, AWS allows a limit of 50 tags. When you provision your VM on AWS using Self-Service, 6 out of 50 tags are automatically populated with keys and values specific to Self-Service VM provisioning. You can use the remaining 44 tags to define other key-value pairs.

The following table lists the Self-Service-specific categories or tags and their availability for different providers:

Table 35: Self-Service-Specific Categories or Tags

Categories or Tags	Nutanix	AWS	VMware	GCP	Azure
account_uuid		X	X	X	X
CalmApplication	X	X	X	X	X
CalmService	X	X	X	X	X
CalmUsername	X	X	X	X	X
Calm Project		X	X	X	X
OSType	X	X	X	X	X

SINGLE-VM BLUEPRINTS IN SELF-SERVICE

A single-VM blueprint is a framework that you can use to create and provision an instance and launch apps that require only one virtual machine.

Creating a Single-VM Blueprint

Single-VM blueprints enable you to quickly provide Infrastructure-as-a-Service (IaaS) to your end users.

About this task

You can create single-VM blueprints with your Nutanix, VMware, AWS, GCP, or Azure accounts. Use these steps to create a single-VM blueprint with any of your provider accounts.

Before you begin

Ensure that the Prism web console (also known as Prism Element) is registered with your Prism Central.

Procedure

1. Set up your single-VM blueprint. In this step, you provide the name and description for the blueprint and select the project and environment for the blueprint. This step is common for all provider accounts. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
2. Add VM details to your blueprint. In this step, you provide a VM name and associate a provider account and an operating system to the blueprint. This step is also common for all provider accounts. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
3. Configure the VM of your blueprint. This options available for VM configuration are derived from either the project or the environment that you selected while setting up the blueprint. For more information, see [VM Configuration](#) on page 107.
4. Configure advanced options. In this optional step, you add credentials, configure options to check the logon status of the VM after blueprint provisioning, add pre-create and post-delete tasks, or add packages. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.

Setting up a Single-VM Blueprint

Perform the following steps to do the preliminary setup of your single-VM blueprint.

Before you begin

Ensure that you created a project and configured an environment for the provider account that you want to associate to your blueprint. For more information, see [Creating a Project](#) and [Configuring Environments in a Project](#) in the *Prism Central Admin Center Guide*.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. Click **+ Create Blueprint > Single VM Blueprint**.

5. On the **Blueprint Settings** tab, enter a name and a description for your blueprint.
6. From the **Project** dropdown menu, select a project.
7. From the **Environment** dropdown menu, select an environment to configure your blueprint.
8. To save your blueprint setup, click **Save**.

What to do next

Click **VM Details** to provide a VM name and associate a provider account and an operating system to the blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Adding VM Details to a Blueprint

Perform the following steps to add VM details to your blueprint.

Before you begin

Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.

Procedure

1. On the **VM Details** tab, enter a name for the VM.
2. From the **Account** dropdown menu, select the provider account that you want to associate to your blueprint.
If your provider account does not appear in the **Account** dropdown menu, ensure that you selected the correct project on the **Blueprint Settings** tab. The project must have the required provider account configured.
3. From the **Operating System** dropdown menu, select either **Linux** or **Windows** as the operating system for the VM.
4. To save the configurations, click **Save**.

What to do next

Click **VM Configuration** and configure the VM in your single-VM blueprint. For more information, see [VM Configuration](#) on page 107.

VM Configuration

Configuring the VM in your blueprint is specific to the provider account and the operating system you select for your blueprint. You can configure the VM in a blueprint with Nutanix, VMware, AWS, GCP, or Azure accounts.

Configuring VM for Nutanix Account

Perform the following steps to configure the VM in a single-VM blueprint for your Nutanix account.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Procedure

1. If you have configured an environment during the project creation, then on the **VM Configuration** tab, click **Clone from environment** to autofill the VM configuration details. This step is optional.
The **Clone from environment** button appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu on the **Blueprint Settings** tab. The option does not appear if you select **All Project Accounts** as your environment.
You can also click the **View Configuration** option to review the configuration details before cloning the environment.
2. In the **Cluster** dropdown menu, select the cluster that you want to associate to the blueprint.
The **Cluster** dropdown menu displays the clusters that you allowed in the project.
The VLAN subnets have direct association with the cluster. When you select a VLAN subnet under the **Network Adapters (NICs)** section, the associated cluster is auto-populated in the **Cluster** dropdown menu. However, if you intend to use overlay subnets, you must select the cluster in list.
If you mark the cluster as runtime editable, the selected subnets also become runtime editable.
3. Edit the VM name in the **VM Name** field.
You can use Self-Service macros to provide a unique name to the VM. For example, `vm-@@{calm_time}@@`. For more information on Self-Service macros, see [Macros Overview](#) on page 91.
4. Configure the processing unit of the VM by entering the number of vCPU, cores of each vCPU, and total memory in GB of the VM in the **vCPU**, **cores per vCPU**, and **Memory (GiB)** fields.
5. To configure post-create tasks or actions, select the **Off** radio button next to the **VM Power State** to keep the VM in the powered off state after creation.
You can configure post-create tasks or actions on the **Advanced Options** tab. For more information on post-create options, see [Configuring Tasks or Packages in a Blueprint](#) on page 124.
6. (Optional) If you want to customize the default OS properties of the VM, select the **Guest Customization** checkbox.
Guest customization allows you to modify the properties of the VM operating system. You can prevent conflicts that might result due to the deployment of virtual machines with identical settings, such as duplicate VM names or same SID. You can also change the computer name or network settings by using a custom script.
 - a. Select **Cloud-init** for Linux or **SysPrep** for Windows, and enter or upload the script in the **Script** panel.
For Sysprep, you must use double back slash for all escape characters . For example, `\\v.`
 - b. For Sysprep script, click **Join a Domain** checkbox and configure the following fields.
 - Enter the domain name of the Windows server in the **Domain Name** field.
 - Select a credential for the Windows VM in the **Credentials** dropdown menu. You can also add new credentials.
 - Enter the IP address of the DNS server in the **DNS IP** field.
 - Enter the DNS search path for the domain in the **DNS Search Path** field.

7. Under the **Disks** section, do the following:
 - a. To add a disk, click the **+** icon next to **Disks**.
 - b. Select the device from the **Device Type** dropdown menu.
You can select **CD-ROM** or **DISK**.
 - c. Select the device bus from the **Device Bus** dropdown menu.
You can select **IDE** or **SATA** for CD-ROM and **SCSI, IDE, PCI**, or **SATA** for DISK.
 - d. From the **Operation** dropdown menu, select one of the following:
 - » To allocate the disk memory from the storage container, select **Allocate on Storage Container**.
 - » To clone an image from the disk, select **Clone from Image Service**.
 - e. If you selected **Allocate on Storage Container**, enter the disk size in GB in the **Size (GiB)** field.
 - f. If you selected **Clone from Image Service**, select the image you want to add to the disk in the **Image** field.
All the images that you uploaded to Prism Central are available for selection. For more information on image configuration, see [Image Management](#) section in the *Prism Central Infrastructure Guide*.
 - g. To expand the size of a boot disk for the image, specify the size in the **Size (GiB)** field.
 - h. Select the **Bootable** checkbox for the image that you want to use to start the VM.

Note: You can add more than one disk and select the disk with which you want to boot up the VM.

8. Under the **Boot Configuration** section, select a firmware type to boot the VM.
 - » To boot the VM with legacy BIOS firmware, select **Legacy BIOS**.
 - » To boot the VM with UEFI firmware, select **UEFI**. UEFI firmware supports larger hard drives, faster boot time, and provides more security features.
 - » To boot the VM with the Secure Boot feature of UEFI, select **Secure Boot**. Secure Boot ensures a safe and secure start by preventing unauthorized software such as a malware to take control during the VM bootup.
9. (For GPU-enabled clusters only) To configure a vGPU, click the **+** icon under the **vGPUs** section and do the following:
 - a. From the **Vendor** dropdown menu, select the GPU vendor.
 - b. From the **Device ID** dropdown menu, select the device ID of the GPU.
 - c. From the **Mode** dropdown menu, select the GPU mode.
10. Under the **Categories** section, select a category in the **Key: Value** dropdown menu.
Use this option to tag your VM to a defined category in Prism Central. The list options are available based on your Prism Central configuration. If you want to protect your app by a protection policy, select the category defined for the policy in your Prism Central.

11. To add a network adapter, click the + icon next to the **Network Adapters (NICs)** field.
The **NIC** dropdown menu shows all the VLAN and overlay subnets. The VLAN subnets have direct association with the cluster. Therefore, when you select a VLAN subnet, the associated cluster is auto-populated in the **Cluster** dropdown menu.
The NICs of a VM can either use VLAN subnets or overlay subnets. For example, if you select an overlay subnet in NIC 1 and then add NIC 2, the NIC 2 list displays only the overlay subnets.
If you select a VLAN subnet in NIC 1, all subsequent VLAN subnets belong to the same cluster. Similarly, if you select an overlay subnet, all subsequent overlay subnets belong to the same VPC.
12. To add a serial port to the VM, click the + icon next to the **Serial Ports** field.
You can use serial ports to connect a physical port or a file on the VM.
13. To save the blueprint, click **Save**.

What to do next

- You can optionally configure the advanced options in the blueprint. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring VM for VMware Account

Perform the following steps to configure the VM in a single-VM blueprint for your VMware account.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Procedure

1. (Optional) If you have configured an environment during the project creation, then on the **VM Configuration** tab, click **Clone from environment** to autofill the VM configuration details.
The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu on the **Blueprint Settings** tab. The option does not appear if you select **All Project Accounts** as your environment.
You can also click the **View Configuration** option to review the configuration details before cloning the environment.
2. Select the **Compute DRS Mode** checkbox to enable load sharing and automatic VM placement.
Distributed Resource Scheduler (DRS) is a utility that balances computing workloads with available resources in a virtualized environment. For more information on DRS mode, see the *VMware documentation*.
 - » If you selected **Compute DRS Mode**, then select the cluster where you want to host your VM from the **Cluster** dropdown menu.
 - » If you have not selected **Compute DRS Mode**, then select the host name of the VM from the **Host** dropdown menu.

3. Do one of the following:

- » Select **VM Templates** and then select a template from the **Template** dropdown menu.

Templates allow you to create multiple virtual machines with the same characteristics, such as resources allocated to CPU and memory or the type of virtual hardware. Templates save time and avoid errors when configuring settings and other parameters to create VMs. The VM template retrieves the list options from the configured vCenter.

Note:

- Install the VMware Tools on the Windows templates. For Linux VMs, install *Open-vm-tools* or *VMware-tools* and configure the *Vmtoolsd* service for automatic start-up.
- Support for *Open-vm-tools* is available. When using *Open-vm-tools*, install Perl for the template.
- Do not use SysPrepped as the Windows template image.
- If you select a template that has unsupported version of VMware Tools, then a warning appears stating VMware tool or version is unsupported and could lead to VM issues.
- You can also edit the NIC type when you use a template.

For more information, refer to VMware KB articles.

- » Select **Content Library**, a content library in the **Content Library** dropdown menu, and then select an OVF template or VM template from the content library.

A content library stores and manages content (VMs, vApp templates, and other types of files) in the form of library items. A single library item can consist of one file or multiple files. For more information on the vCenter content library, see the *VMware Documentation*.

Caution: Content Library support is currently a technical preview feature in Self-Service. Do not use any technical preview features in a production environment.

4. If you want to use the storage DRS mode, then select the **Storage DRS Mode** checkbox and a datastore cluster from the **Datastore Cluster** dropdown menu.

The datastore clusters are referred as storage pod in vCenter. A datastore cluster is a collection of datastores with shared resources and a shared management interface.

5. If you do not want to use storage DRS mode, then do not select the **Storage DRS Mode** checkbox, and select a datastore from the **Datastore** dropdown menu.

6. In the **VM Location** field, specify the location of the folder in which the VM must be created when you deploy the blueprint. Ensure that you specify a valid folder name already created in your VMware account.

To create a subfolder in the location you specified, select the **Create a folder/directory structure here** checkbox and specify a folder name in the **Folder/Directory Name** field.

Note: Self-Service gives preference to the VM location specified in the environment you select while launching an app. For example, you specify a subfolder structure as the VM location in the blueprint and the top-level folder in the environment. When you select this environment while launching your app, Self-Service considers the VM location you specified in the environment and creates the VM at the top-level folder.

Select the **Delete empty folder** checkbox to delete the subfolder created within the specified location, in case the folder does not contain any VM resources. This option helps you to keep a clean folder structure.

7. Enter the instance name of the VM in the **Instance Name** field.
This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
8. If you want to configure post-create tasks or actions, then select the **Off** radio button next to the **VM Power State** to keep the VM in the powered off state after creation.
You can configure post-create tasks or actions on the **Advanced Options** tab. For more information on post-create options, see [Configuring Tasks or Packages in a Blueprint](#) on page 124.
9. Under **VM Config**, do the following:
 - a. Select the **CPU Hot Add** checkbox if you want to increase the VCPU count of a running VM.
Support for CPU Hot Add depends on the Guest OS of the VM.
 - b. Update the **vCPUs** and **Core Per Socket** count.
The number of sockets is calculated by dividing the total number of vCPUs by Cores Per Socket. For example, if the number of vCPUs is 4 and the Cores Per Socket is 2, then the number of sockets will be 2.
 - c. Select the **Memory Hot Plug** checkbox if you want to increase the memory of a running VM.
Support for Memory Hot Plug depends on the Guest OS of the VM.
 - d. Update the memory in the **Memory** field.
10. Under **Controllers**, click the **+** icon to add the type of controller.
You can select either SCSI or SATA controller. You can add up to three SCSI and four SATA controllers.
11. Under the **Disks** section, click the **+** icon to add vDisks and do the following:
 - a. Select the device type from the **Device Type** dropdown menu.
You can either select **CD-ROM** or **DISK**.
 - b. Select the adapter type from the **Adapter Type** dropdown menu.
You can select **IDE** for CD-ROM.
You can select **SCSI**, **IDE**, or **SATA** for DISK.
 - c. Enter the size of the disk in GiB.
 - d. In the **Location** field, select the disk location.
 - e. If you want to add a controller to the vDisk, select the type of controller in the **Controller** dropdown menu to attach to the disk.

Note: You can add either SCSI or SATA controllers. The available options depend on the adapter type.
 - f. In the **Disk mode** dropdown menu, select the type of the disk mode. Your options are:
 - » **Dependent:** Dependent disk mode is the default disk mode for the vDisk.
 - » **Independent - Persistent:** Disks in persistent mode behave like conventional disks on your physical computer. All data written to a disk in persistent mode are written permanently to the disk.
 - » **Independent - Nonpersistent:** Changes to disks in nonpersistent mode are discarded when you shut down or reset the virtual machine. With nonpersistent mode, you can restart the virtual

machine with a virtual disk in the same state every time. Changes to the disk are written to and read from a redo log file that is deleted when you shut down or reset.

You can also mark the vDisks runtime editable so you can add, delete, or edit the vDisks while launching the blueprint. For more information on runtime editable attributes, see [Runtime Variables Overview](#) on page 104.

12. Under the **Tags** section, select tags from the **Category: Tag pairs** field.
You can assign tags to your VMs so you can view the objects associated with your VMs in your VMware account. For example, you can create a tag for a specific environment and assign the tag to multiple VMs. You can then view all the VMs that are associated with the tag.
13. (Optional) If you want to customize the default OS properties of the VM, then click the **Enable** checkbox under **VM Guest Customization** and select a customization from the **Predefined Guest Customization** dropdown menu.
14. If you do not have any predefined customization available, select **None**.
15. Select **Cloud-init** or **Custom Spec**.
16. If you selected **Cloud-init**, enter or upload the script in the **Script** field.
17. If you have selected **Custom Spec**, enter the details for the VM in the following fields:
 - a. Enter the hostname in the **Hostname** field.
 - b. Enter the domain in the **Domain** field.
 - c. Select timezone from the **Timezone** dropdown menu.
 - d. Select **Hardware clock UTC** checkbox to enable hardware clock UTC.
 - e. Click the **+** icon to add network settings.
To automatically configure DHCP server, enable the **Use DHCP** checkbox and then skip to the **DNS Setting** section.
 - f. Enter a name for the network configuration you are adding to the VM in the **Setting name** field.
Settings name is the saved configuration of your network that you want to connect to your VM.
 - g. Enter values in the **IP Address**, **Subnet Mask**, **Default Gateway**, and **Alternative Gateway** fields.
 - h. Under the **DNS Settings** section, enter the DNS primary, DNS secondary, DNS tertiary, and DNS search path name.

Note: You can launch a single-VM blueprint without a NIC or network adapter with your VMware account.

18. To save the blueprint, click **Save**.

What to do next

- You can optionally configure the advanced options in the blueprint. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring VM for GCP Account

Perform the following steps to configure the VM in a single-VM blueprint for your GCP account.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Procedure

1. (Optional) If you have configured an environment during the project creation, then on the **VM Configuration** tab, click **Clone from environment** to autofill the VM configuration details.
The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu on the **Blueprint Settings** tab. The option does not appear if you select **All Project Accounts** as your environment.
You can also click the **View Configuration** option to review the configuration details before cloning the environment.
2. (Optional) Edit the VM name in the **Instance Name** field.
This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
3. Select a zone from the **Zone** dropdown menu.
A zone is a physical location where you can host the VM.
4. Select the type of machine from the **Machine type** dropdown menu.
The machine types are available based on your zone. A machine type is a set of virtualized hardware resources available to a virtual machine (VM) instance, including the system memory size, virtual CPU (vCPU) count, and persistent disk limits. In Compute Engine, machine types are grouped and curated by families for different workloads.
5. Under the **DISKS** section, click the + icon to add a disk.
You can also mark the added vDisks runtime editable so you can add, delete, or edit the vDisks while launching the blueprint. For more information on runtime editable attributes, see [Runtime Variables Overview](#) on page 104.
6. To use an existing disk configuration, select the **Use existing disk** checkbox, and then select the persistent disk from the **Disk** dropdown menu.

7. If you have not selected the **Use existing disk** checkbox, then do the following:
 - a. Select the type of storage from the **Storage Type** dropdown menu. The available options are as follows.
 - » **pd-balanced**: Use this option as an alternative to SSD persistent disks with a balanced performance and cost.
 - » **pd-extreme**: Use this option to use SSD drives for high-end database workloads. This option has higher maximum IOPS and throughput and allows you to provision IOPS and capacity separately.
 - » **pd-ssd**: Use this option to use SSD drives as your persistent disk.
 - » **pd-standard**: Use this option to use HDD drives as your persistent disk.

The persistent disk types are durable network storage devices that your instances can access like physical disks in a desktop or a server. The data on each disk is distributed across several physical disks.
 - b. Select the image source from the **Source Image** dropdown menu.

The images available for your selection are based on the selected zone.
 - c. Enter the size of the disk in GB in the **Size in GB** field.
 - d. To delete the disk configuration after the instance is deleted, select the **Delete when instance is deleted** checkbox under the **Disks** section.
8. To add a blank disk, click the **+** icon under the **Blank Disks** section and configure the blank disk.
9. To add networking details to the VM, click the **+** icon under the **Networking** section.
10. To configure a public IP address, select the **Associate Public IP address** checkbox and configure the following fields.
 - a. Select the network from the **Network** dropdown menu and the sub network from the **Subnetwork** dropdown menu.
 - b. Enter a name of the network in the **Access configuration Name** field and select the access configuration type from the **Access configuration type** dropdown menu.

These fields appear when you select the **Associate public IP Address** checkbox.
11. To configure a private IP address, clear the **Associate Public IP address** checkbox and select the network and sub network.
12. Under the **SSH Key** section, click the **+** icon and enter or upload the username key data in the **Username** field.
13. Select **Block project-wide SSH Keys** to enable blocking project-wide SSH keys.
14. Under the **Management** section, do the following:
 - a. Enter the metadata in the **Metadata** field.
 - b. Select the security group from the **Network Tags** dropdown menu.

Network tags are text attributes you can add to VM instances. These tags allow you to make firewall rules and routes applicable to specific VM instances.
 - c. Enter the key-value pair in the **Labels** field.

A label is a key-value pair that helps you organize the VMs created with GCP as the provider. You can attach a label to each resource, then filter the resources based on their labels.

15. Under the **API Access** section, do the following:
 - a. Specify the service account in the **Service Account** field.
 - b. Under Scopes, select **Default Access** or **Full Access**.
16. To save the blueprint, click **Save**.

What to do next

- You can optionally configure the advanced options in the blueprint. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring VM for AWS Account

Perform the following steps to configure the VM in a single-VM blueprint for your AWS account.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Procedure

1. (Optional) If you have configured an environment during the project creation, then on the **VM Configuration** tab, click **Clone from environment** to autofill the VM configuration details.

The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu on the **Blueprint Settings** tab. The option does not appear if you select **All Project Accounts** as your environment.

You can also click the **View Configuration** option to review the configuration details before cloning the environment.
2. Edit the VM name in the **Instance Name** field.

This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
3. Select the **Associate Public IP Address** checkbox to associate a public IP address with your AWS instance.

If you do not select the **Associate Public IP Address** checkbox, ensure that the AWS account and Self-Service are on the same network for the scripts to run.
4. Select an AWS instance type from the **Instance Type** dropdown menu.

Instance types include varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to select the appropriate mix of resources for your apps. Each instance type includes one or more instance sizes that allows you to scale your resources to the requirements of your target workload.

The list displays the instances that are available in the AWS account. For more information, see AWS documentation.

5. Select a region from the **Region** dropdown menu and do the following:

Note: The list displays the regions that are selected while configuring the AWS setting.

a. Select an availability zone from the **Availability Zone** dropdown menu.

An availability zone is one or more discrete datacenters with redundant power, networking, and connectivity in an AWS region. Availability zones allow you to operate production apps and databases that are more highly available, fault tolerant, and scalable than would be possible from a single datacenter.

b. Select a machine image from the **Machine Image** dropdown menu.

An Amazon Machine Image is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud. It serves as the basic unit of deployment for services delivered using EC2.

c. Select an IAM role from the **IAM Role** dropdown menu.

An IAM role is an AWS Identity and Access Management entity with permissions to make AWS service requests.

d. Select a key pair from the **Key Pairs** dropdown menu.

A key pair (consisting of a private key and a public key) is a set of security credentials that you use to prove your identity when connecting to an instance.

e. Select the VPC from the **VPC** dropdown menu and do the following:

Amazon Virtual Private Cloud (Amazon VPC) allows you to provision a logically isolated section of the AWS cloud where you can launch AWS resources in your defined virtual network.

- Select the **Include Classic Security Group** checkbox to enable security group rules.
- Select security groups from the **Security Groups** dropdown menu.
- Select a subnet from the **Subnet** dropdown menu.

6. Enter or upload the AWS user data in the **User Data** field.

7. Enter AWS tags in the **AWS Tags** field.

AWS tags are key and value pair to manage, identify, organize, search for, and filter resources. You can create tags to categorize resources by purpose, owner, environment, or other criteria.

8. Under the **Storage** section, configure the following to boot the AWS instance with the selected image.

a. From the **Device** dropdown menu, select the device to boot the AWS instance.

The available options are based on the image you have selected.

b. In the **Size(GiB)** field, enter the required size for the bootable device.

c. From the **Volume Type** dropdown menu, select the volume type. You can select either **General Purpose SSD**, **Provisioned IOPS SSD**, and **EBS Magnetic HDD**.

For more information on the volume types, see AWS documentation.

d. Optionally, select the **Delete on termination** checkbox to delete the storage when the instance is terminated.

You can also add more secondary storages by clicking the **+** icon next to the **Storage** section.

9. To save the blueprint, click **Save**.

What to do next

- You can optionally configure the advanced options in the blueprint. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring VM for Azure Account

Perform the following steps to configure the VM in a single-VM blueprint for your Azure account.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Procedure

1. (Optional) If you have configured an environment during the project creation, then on the **VM Configuration** tab, click **Clone from environment** to autofill the VM configuration details.
The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu on the **Blueprint Settings** tab. The option does not appear if you select **All Project Accounts** as your environment.
You can also click the **View Configuration** option to review the configuration details before cloning the environment.
2. Edit the VM name in the **Instance Name** field.
This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
3. Select a resource group from the **Resource Group** dropdown menu or select the **Create Resource Group** checkbox to create a resource group.
Each resource in Azure must belong to a resource group. A resource group is simply a logical construct that groups multiple resources together so you can manage the resources as a single entity. For example, you can create or delete resources as a group that share a similar life cycle, such as the resources for an n-tier app.
The **Resource Group** dropdown menu displays the resource groups that are associated with the subscriptions you selected in your Azure account. In case you have not selected any subscriptions, Self-Service considers all the subscriptions that are available in the Azure service principal to display the resource groups. Each resource group in the list also displays the associated subscription.
4. If you selected a resource group from the **Resource Group** dropdown menu, then do the following:
 - a. Select the geographical location of the datacenter from the **Location** dropdown menu.
 - b. Select **Availability Sets** or **Availability Zones** from the **Availability Option** dropdown menu.
You can then select an availability set or availability zone. An availability set is a logical grouping capability to ensure that the VM resources are isolated from each other to provide High Availability

if deployed within an Azure datacenter. An availability zone allows you to deploy your VM into different datacenters within the same region.

- c. Select the hardware profile as per your hardware requirements from the **Hardware Profile** dropdown menu.

The number of data disks and NICs depends upon the selected hardware profile. For information on the sizes of Windows and Linux VMs, see *Windows and Linux Documentation*.

5. If you selected the **Create Resource Group** checkbox to create a resource group, then do the following:
 - a. Select a subscription associated to your Azure account in the **Subscription** field.
 - b. Enter a unique name for the resource group in the **Name** field.
 - c. Select the geographical location of the datacenter that you want to add to the resource group in the **Location** dropdown menu.
 - d. Under **Tags**, enter a key and value pair in the **Key** and **Value** fields respectively.
Tags are key and value pairs that enable you to categorize resources. You can apply a tag to multiple resource groups.
 - e. If you want to automatically delete a resource group that has empty resources while deleting an app, click the **Delete Empty Resource Group** checkbox.
 - f. Specify the location and hardware profile.
6. Under the **Secrets** section, click the **+** icon and do the following:
 - a. Enter a unique vault ID in the **Vault ID** field.
 - b. Under **Certificates**, click the **+** icon.
 - c. Enter the URL of the configuration certificate in the **URL** field.
The URL of the certificate is uploaded to the key vault as a secret.
7. Under the **Admin Credentials** section, do the following:
 - a. Enter the username in the **Username** field.
 - b. Select a secret type from the **Secret Type** dropdown menu.
You can either select Password or SSH Private Key.
 - c. Do one of the following.
 - » If you selected password, then enter the password in the **Password** field.
 - » If you selected SSH Private Key, then enter or upload the SSH Private Key in the **SSH Private Key** field.
 - You can use the selected or default credential as the default credential for the VM.
 - You cannot use key-based credential for Windows VMs.
 - Username and password must adhere to the complexity requirements of Azure.
8. (For Windows) Select the **Provision Windows Guest Agent** checkbox.
This option indicates whether or not to provision the virtual machine agent on the virtual machine. When this property is not specified in the request body, the default behavior is to set it to true. This ensures that the VM Agent is installed on the VM, and the extensions can be added to the VM later.

9. (For Windows) To indicate that the VM is enabled for automatic updates, select the **Automatic OS Upgrades** checkbox.

10. Under the **Additional Unattended Content** section, click the + icon and do the following:

a. Select a setting from the **Setting Name** dropdown menu.

You can select **Auto Logon** or **First Logon Commands**.

Note: Guest customization is applicable only on images that allow or support guest customization.

b. Enter or upload the xml content. For more information, see [Sample Auto Logon and First Logon Scripts](#) on page 422.

11. Under the **WinRM Listeners** section, click the + icon and do the following:

a. Select the protocol from the **Protocol** dropdown menu.

You can select **HTTP** or **HTTPS**.

b. If you selected HTTPS, then select the certificate URL from the **Certificate URL** dropdown menu.

12. Under the **Storage Profile** section, select the **Use Custom Image** checkbox to use a custom VM image created in your subscription.

You can then select a custom image or publisher-offer-SKU-version from the **Custom Image** dropdown menu.

13. Under the **VM Image Details** section, select an image type in the **Source Image Type** dropdown menu.

You can select **Marketplace**, **Subscription**, or **Shared Image Gallery**.

» If you selected **Marketplace**, then specify the publisher, offer, SKU, and version for the image.

» If you selected **Subscription**, then select the custom image.

» If you selected **Shared Image Gallery**, then select the gallery and the image.

14. Under the **OS Disk Details** section, do the following:

a. Select the storage type from the **Storage Type** dropdown menu.

You can select **Standard HDD**, **Standard SSD**, or **Premium SSD**.

b. Select a disk storage account from the **Disk Storage** dropdown menu.

This field is available only when the **Use Custom Image** checkbox is enabled.

c. Select disk caching type from the **Disk Caching Type** dropdown menu.

You can select **None**, **Read-only**, or **Read write**.

d. Select disk create option from the **Disk Create Option** dropdown menu.

You can select **Attach**, **Empty**, or **From Image**.

15. Under the **Data Disk** section, do the following:

- a. Select the storage type from the **Storage Type** dropdown menu.
You can select **Standard HDD**, **Standard SSD**, or **Premium SSD**.
- b. Select disk caching type from the **Disk Caching Type** dropdown menu.
You can select **None**, **Read-only**, or **Read write**.
- c. Enter the size in GB in the **Size** field.
- d. Enter disk logical unit number (LUN) in the **Disk LUN** field.

Note: The LUN value should be unique across data disk list.

16. Under the **Network Profile** section, add NICs as per your requirement and do the following for each NIC:

- a. Select a security group from the **Security Group** dropdown menu.
- b. Select application security groups from the **ASGs** dropdown menu.

Unlike Network Security Groups that are defined at the network level, Application Security Groups (ASGs) are logical entities that are defined at the app level. ASGs help to manage the VM security by grouping the VMs according to the apps that run on them and allow app-centric use of Network Security Groups. You can select multiple ASGs in a NIC.

Note:

- ASGs are supported when you clone your apps and are also supported in snapshot and restore.
- ASGs are not supported in VM configuration updates.

- c. Select a virtual network from the **Virtual Network** dropdown menu.
- d. Under **Public IP Config**, enter a name and select an allocation method.
- e. Under **Private IP Config**, select an allocation method.
If you selected **Static** as the allocation method, then enter the private IP address in the **IP Address** field.

17. Enter tags in the **Tags** field.

18. To save the blueprint, click **Save**.

What to do next

- You can optionally configure the advanced options in the blueprint. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring VM for Nutanix Cloud Account

Perform the following steps to configure the VM in a single-VM blueprint for your Nutanix Cloud account.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.

Procedure

1. On the **VM Configuration** tab, edit the VM name in the **VM Name** field.
You can use Self-Service macros to provide a unique name to the VM. For example, `vm-@@{calm_time}@@`. For more information on Self-Service macros, see [Macros Overview](#) on page 91.
2. Configure the processing unit of the VM by entering the number of vCPU, cores of each vCPU, and total memory in GB of the VM in the **vCPU**, **cores per vCPU**, and **Memory** fields.
3. If you want to customize the default OS properties of the VM, select the **Guest Customization** checkbox.
Guest customization allows you to modify the properties of the VM operating system. You can prevent conflicts that might result due to the deployment of virtual machines with identical settings, such as duplicate VM names or same SID. You can also change the computer name or network settings by using a custom script.
4. Select **Cloud-init** for Linux or **SysPrep** for Windows, and enter or upload the script in the **Script** panel.
For Sysprep, you must use double back slash for all escape characters . For example, `\\v.`
For Sysprep script, click **Join a Domain** checkbox and configure the following fields.
 - Enter the domain name of the Windows server in the **Domain Name** field.
 - Select a credential for the Windows VM in the **Credentials** dropdown menu. You can also add new credentials.
 - Enter the IP address of the DNS server in the **DNS IP** field.
 - Enter the DNS search path for the domain in the **DNS Search Path** field.
5. Select the image from the **Image** dropdown menu.
The list displays the images that are available in the cluster. You can add more than one image by clicking the + icon.
All the images that you uploaded to Prism Central are available for selection. For more information on image configuration, see [Image Management](#) section in the *Prism Central Infrastructure Guide*.
6. Select the device from the **Device Type** dropdown menu.
You can select **CD-ROM** or **Disk**.
7. Select the device bus from the **Device Bus** dropdown menu.
You can select **IDE** or **SATA** for CD-ROM and **SCSI**, **IDE**, **PCI**, or **SATA** for DISK.
8. Select the **Bootable** checkbox for the image that you want to use to start the VM.
9. To add a vDisk, click the + icon and specify the device type, device bus, and disk size.
You can also mark the vDisks runtime editable so you can add, delete, or edit the vDisks while launching the blueprint. For more information on runtime editable attributes, see [Runtime Variables Overview](#) on page 104.

10. Under **Categories**, select a category in the list.
Use this option to tag your VM to a defined category in Prism Central. The list options are available based on your Prism Central configuration. If you want to protect your app by a protection policy, select the category defined for the policy in your Prism Central.
11. Under the **Network** section, select the VPC from the **VPC** dropdown menu. For more information on VPC, see [Nutanix Cloud Infrastructure Service Administration Guide](#).
12. To save the blueprint, click **Save**.

What to do next

- You can optionally configure the advanced options in the blueprint. For more information, see [Configuring Advanced Options for a Blueprint](#) on page 123.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring Advanced Options for a Blueprint

Perform the following steps to configure advanced options such as credentials, packages, pre-create and post-delete tasks. Configuring advanced options is optional for a blueprint.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.

Procedure

1. Add credentials to enable packages and actions. For more information, see [Adding Credentials](#) on page 173.
2. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.
3. Configure pre-create, post-create, or post-delete tasks and packages in the blueprint. For more information, see [Configuring Tasks or Packages in a Blueprint](#) on page 124.
4. Add an action. For more information, see [Adding an Action to a Single-VM Blueprint](#) on page 194.
5. Click **Save**.

What to do next

- You can configure app variables in the blueprint. For more information, see [Configuring App Variables in a Blueprint](#) on page 126.
- You can view the blueprint on the Blueprints page. You can use the blueprint to model your app. For more information, see [Blueprints Management in Self-Service](#) on page 221.

Configuring Tasks or Packages in a Blueprint

Perform the following steps to configure a pre-create task, post-create task, post-delete task, install package, or uninstall package in a single-VM blueprint.

About this task

You have the following Advanced Options for your single-VM blueprint:

- Pre VM create tasks

Pre VM create tasks are actions that are performed before the blueprint or application is launched. These tasks help you to prepare the environment or set up prerequisites required for the successful deployment of the application or blueprint. Pre VM create tasks enable administrators or developers to automate and streamline the preparation steps thereby reducing manual efforts and potential errors during application deployments.

- Post VM create tasks

The Post VM create stage is a stage between the creation of VM and package installation. The Post VM create tasks allow addition of automated tasks and actions that run on substrate without any manual intervention before the VM is powered on. To configure these automated tasks or actions, you have to keep the VM in the powered off state after creation. You can use Post VM create configuration to carry out tasks such as:

- To use the Mac address of the VM for further orchestration in the pre-boot stage.
- To place the provisioned VMs in a security group before they are powered on.

The post-create tasks or actions are applicable to all providers.

- Post VM delete tasks

Post VM delete tasks are actions that are performed after a blueprint or application is deleted or terminated. These tasks are configured to clean up any remaining resources, configurations, or dependencies that were created during the deployment or execution of the blueprint or application. Post VM delete tasks promote efficient resource utilization and help maintain a clean infrastructure state.

- Package install

Package install defines the software packages that need to be installed on the target machines as part of the blueprint deployment. You can specify the package names, versions, repositories, and any additional configuration parameters required for the installation process.

- Package uninstall

Package uninstall defines the software packages that should be uninstalled from the target machines when the blueprint is terminated or deleted. Similar to package install, you can specify the package names and any additional parameters required for the uninstallation process.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.

Procedure

Perform the following steps to configure the Advanced options of your blueprint.

1. Add a VM credential to enable packages and actions. For more information, see [Adding Credentials](#) on page 173.
2. Do one of the following:
 - » To configure a pre-create, post-create, or post-delete task, click **Edit** next to the **Pre VM create tasks**, **Post VM create tasks**, or **Post VM delete tasks** field under the **PreCreate**, **PostCreate** & **PostDelete** section.
 - » To configure an install or uninstall package, click **Edit** next to the **Package Install** or **Package Uninstall** field under the **Packages** section.

For **Post VM create tasks** in Nutanix or VMware, ensure that the VM is configured to stay in the powered off state after it is created. For more information, see [Configuring VM for Nutanix Account](#) on page 107 or [Configuring VM for VMware Account](#) on page 110.

3. Click **+ Add Task**.
For **Post VM create tasks**, you can also click **+ Add Action** to add a Service or VM Action such as **VM Power On**, **VM Power Off**, **VM Restart**, or **VM Check Login**.
4. Click the task and enter the task name in the **Task Name** field.
5. Select the type of tasks from the **Type** dropdown menu.
The available options are:
 - **Execute**: Use this task type to run eScripts on the VM. For more information, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: Use this task to change variables in a blueprint. For more information, see [Adding a Set Variable Task](#) on page 179.
 - **Delay** : Use this task type to set a time interval between two tasks or actions. For more information, see [Adding a Delay Task](#) on page 186.
 - **HTTP Task**: Use this task type to query REST calls from a URL. An HTTP task supports GET, PUT, POST, and DELETE methods. For more information, see [Adding an HTTP Task](#) on page 185.
6. To establish a connection between tasks, click **Add Connector** and use the arrow to create the connection.
7. To delete a task, click **Delete** next to the task.
8. To add variables, do one of the following:
 - » To add a pre-create, post-create, or post-delete variable, click the **Pre create Variables**, **Post Create Variables** or **Post Delete Variables** tab.
 - » To add a package install or uninstall variable, click the **Package Install Variables** or **Package Uninstall Variables** tab.
9. Click the **+** icon next to **Variables**.
10. In the **Name** field, enter a name for the variable.

11. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
If you selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259.
If you imported a custom variable type, all the variable parameters are auto filled.
12. If you want to hide the variable value, select the **Secret** checkbox.
13. Click **Done**.

Configuring App Variables in a Blueprint

Perform the following steps to configure app variables in your blueprint.

Procedure

1. On the blueprint page, click **App variables**.
2. Click **+ Add Variable**.
3. In the **Name** field, enter a name for the variable.
4. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type. Your options are:
 - » String
 - » Integer
 - » Multi-line string
 - » Date
 - » Time
 - » Date Time

If you selected a base type variable, then configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259.

If you have imported a custom variable type, all the variable parameters are auto filled.
5. Enter a value for the selected data type in the **Value** field.
You can select the **Secret** checkbox to hide the variable value.
6. Click **Show Additional Options**.
7. In the **Input Type** field, select one of the following input types:
 - » **Simple**: Use this option for default value.
 - » **Predefined**: Use this option to assign static values.
 - » **eScript**: Use this option to attach a script that you run to retrieve values dynamically at runtime. Script can return single or multiple values depending on the selected base data type.
 - » **HTTP**: Use this option to retrieve values dynamically from the defined HTTP end point. Result is processed and assigned to the variable based on the selected base data type.
8. If you selected **Simple**, then enter the value for the variable in the **Value** field.

9. If you selected **Predefined**, then enter the value for the variable in the **Option** field.
To add multiple values for the variable, click **+ Add Option**, and enter values in the **Option** field.

Note: To make any value as default, select **Default** for the option.

10. If you selected **eScript**, do the following:

- a. Select the Python version from the version drop-down menu. By default, Python 3 is selected as the version for you to enter the script.

Note: Based on the decision that the Python Software Foundation (PSF) took to discontinue support for Python 2, Nutanix has decided to end support for Python 2 on June 30, 2024. Nutanix recommends that you format all your new eScripts in Python 3 and update any existing eScripts from Python 2 to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

- b. Enter the script or use the upload icon to upload the script in the **Script** field.

You can upload the script from the library or from your computer by clicking the upload icon.

You can also publish the script to the library by clicking the publish button.

Note:

- You cannot add macros to eScripts.
- If you have selected **Multiple Input (Array)** checkbox with input type as eScript, then ensure that the script returns a list of values separated by comma. For example, CentOS, Ubuntu, Windows.

11. If you selected **HTTP**, then configure the following fields.

- a. In the **Request URL** field, enter the URL of the server that you want to run the methods on.
- b. In the **Request Method** dropdown menu, select one of the following request methods.

- Use the **GET** method to retrieve data from a specified resource.
- Use the **PUT** method to send data to a server to update a resource.
- Use the **POST** method to send data to a server to create a resource.
- Use the **DELETE** method to send data to a server to delete a resource.

In the **Request Body** field, enter the PUT, POST, or DELETE request. You can also upload the request by clicking the upload icon.

- c. In the **Content Type** dropdown menu, select the type of the output format.
The available options are **XML**, **JSON**, and **HTML**.
- d. In the **Connection Timeout (sec)** field, enter the timeout interval in seconds.
- e. (Optional) In the **Authentication** field, select **Basic** and do the following:
 - In the **Username** field, enter the user name.
 - In the **Password** field, enter the password.
- f. If you want to verify the TLS certificate for the task, select the **Verify TLS Certificate** checkbox.
- g. If you want to use a proxy server that you configured in Prism Central, select the **Use PC Proxy configuration** checkbox.

Note: Ensure that the Prism Central has the appropriate HTTP proxy configuration.

- h. In the **Retry Count** field, enter the number of attempts the system must perform to create a task after each failure.
By default, the retry count is zero. It implies that the task creation procedure stops after the first attempt.
- i. In the **Retry Interval** field, enter the time interval in seconds for each retry if the task fails.
- j. Under the **Headers** section, enter the HTTP header key and value in the **Key** and **Value** fields respectively.
If you want to publish the HTTP header key and value pair as secret, select the **Secrets** checkbox.
- k. Under the **Expected Response Options** section, enter the details for the following fields:
 - In the **Response Code** field, enter the response code.
 - From the **Response Status** dropdown menu, select either **Success** or **Failure** as the response status for the task.
- l. In the **Set Response Path for Variable** field, enter the variables from the specified response path.
The example of json format is \$.x.y and xml format is //x/y. For example, if the response path for variable is \$.[*].display for response.

```
[
  {
    "display": "HTML Tutorial",
```



```

    "url": "https://www.w3schools.com/html/default.asp"
  },
  {
    "display": "CSS Tutorial",
    "url": "https://www.w3schools.com/css/default.asp"
  },
  {
    "display": "JavaScript Tutorial",
    "url": "https://www.w3schools.com/js/default.asp"
  },
  {
    "display": "jQuery Tutorial",
    "url": "https://www.w3schools.com/jquery/default.asp"
  },
  {
    "display": "SQL Tutorial",
    "url": "https://www.w3schools.com/sql/default.asp"
  },
  {
    "display": "PHP Tutorial",
    "url": "https://www.w3schools.com/php/default.asp"
  },
  {
    "display": "XML Tutorial",
    "url": "https://www.w3schools.com/xml/default.asp"
  }
]

```

Then, during the launch time the list options are ["HTML Tutorial","CSS Tutorial","JavaScript Tutorial","jQuery Tutorial","SQL Tutorial","PHP Tutorial","XML Tutorial"].

12. (Optional) Enter a label and description for the variable.

13. (Optional) Set variable options.

- Select the **Mark this variable private** checkbox to make the variable private. Private variables are not shown during the blueprint launch or in the app.
- Select the **Mark this variable mandatory** checkbox to make the variable a requisite for app launch.
- Select the **Validate with Regular Expression** checkbox if you want to test the Regex values. Click **Test Regex**, provide the value for the Regex, and test or save the Regex. You can enter regex values in PCRE format. For more details, see from <http://pcre.org/>.

14. Click **Done**.

MULTI-VM BLUEPRINTS IN SELF-SERVICE

A multi-VM blueprint is a framework that you can use to create an instance, provision, and launch apps that require multiple VMs.

Creating a Multi-VM Blueprint

In a Multi-VM blueprint, you can define the underlying infrastructure of the VMs, app details, and actions that are carried out on a blueprint until the termination of the app.

About this task

You can create and configure multi-VM blueprints with your Nutanix, VMware, AWS, GCP, or Azure accounts.

Before you begin

Ensure that you have configured an account and a project for your blueprint.

Procedure

1. Add a service. For more information, see [Adding a Service](#) on page 130.
2. Configure VM, package, and service for your provider account. For more information, see [Configure Multi-VM, Package, and Service](#) on page 131.
3. Set the service dependencies. For more information, see [Setting up the Service Dependencies](#) on page 165.
4. Add and configure an app profile. For more information, see [Adding and Configuring an App Profile](#) on page 166.
5. (Optional) Add and configure Scale Out and Scale In. For more information, see [Adding and Configuring Scale Out and Scale In](#) on page 198.
6. Create an action. For more information, see [Adding an Action to a Multi-VM Blueprint](#) on page 196.

Adding a Service

Services are the virtual machine instances, existing machines or bare-metal machines, that you can provision and configure by using Self-Service. A service exposes the IP address and ports on which the request is received. You can either provision a single-service instance or multiple services based on the topology of your app.

About this task

For more information on services in Self-Service, see [Services Overview](#) on page 90.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. Click **+ Create Blueprint > Multi VM/Pod VM Blueprint**.
5. Enter the name of the blueprint in the **Name** field.

6. Optionally, provide a description about the blueprint in the **Description** field.
7. Select a project from the **Project** dropdown menu.

Note: The available account options depend on the selected project.

8. Click **Proceed**.
The Multi-VM Blueprint Editor appears.

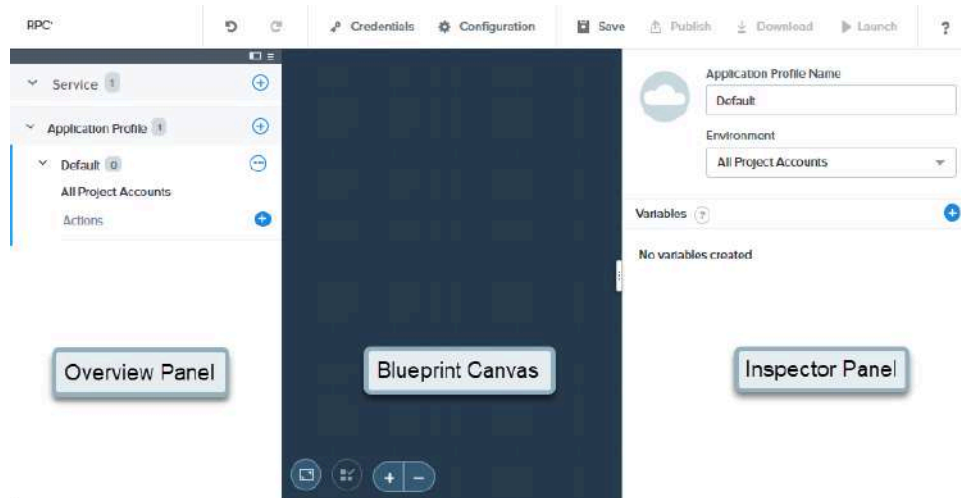


Figure 26: Multi-VM Blueprint Editor

9. To add a service, click the **+** icon next to **Service** in the Overview Panel.
The service inspector appears on the blueprint canvas.

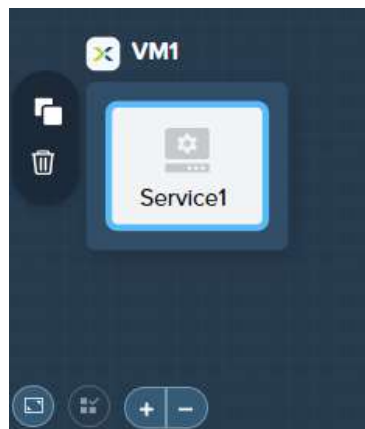


Figure 27: Service Inspector

What to do next

Configure the VM, package, and service. For more information, see [Configure Multi-VM, Package, and Service](#) on page 131.

Configure Multi-VM, Package, and Service

You can define and configure the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app for a service provider.

Configuring Nutanix and Existing Machine VM, Package, and Service

You can define the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app on a Nutanix platform.

Before you begin

- Ensure that you have completed the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have created a project and configured an environment for Nutanix. For more information, see [Creating a Project](#) and [Configuring Nutanix Environment](#).

Procedure

1. Perform the basic blueprint setup, and add a service. For more information, see [Adding a Service](#) on page 130.

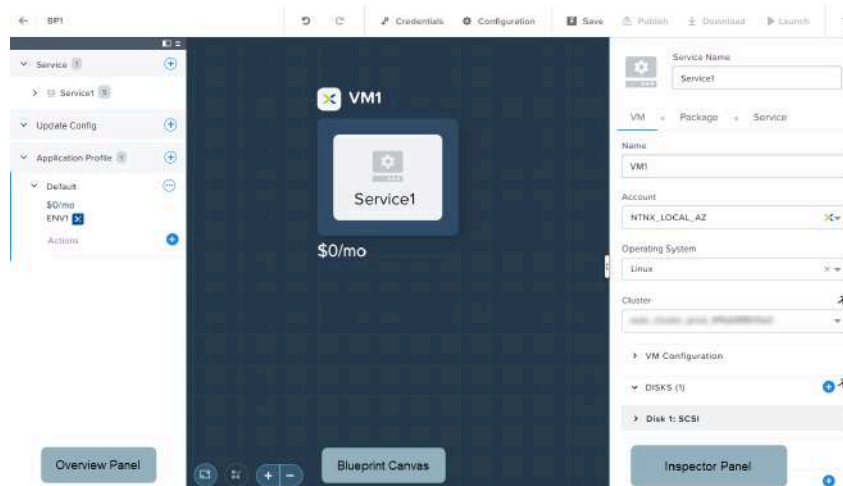


Figure 28: Blueprint Configuration

2. Enter a name of the service in the **Service Name** field.
3. On the **VM** tab, in the **Name** field, enter a name for the VM.
4. Select the provider account from the **Account** dropdown menu. You can select **Existing Machine** or a Nutanix account.

Note: The account options depend on the project you selected while setting up your blueprint.

5. If you selected **Existing Machine**, then do the following:
 - a. Select **Windows** or **Linux** from the **Operating System** dropdown menu.
 - b. In the **Configuration** section, enter the IP address of the existing machine in the **IP Address** field
 - c. In the **Select tunnel to connect with** dropdown menu, select a tunnel that you want to use to connect with this VM if the VM is within the VPC. This step is optional.
 - d. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.

The screenshot shows a web form for configuring a VM. At the top, there are tabs for 'VM', 'Package', and 'Service'. The 'VM' tab is active. Below the tabs, there are several input fields and dropdown menus. The 'Name' field contains 'VM1'. The 'Account' dropdown is set to 'Existing Machine'. Below this, a blue banner states 'Showback is only available for Nutanix and VMware.' The 'Operating System' dropdown is set to 'Linux'. Under a 'Configuration' section, the 'IP Address' field is empty. At the bottom, the 'Select tunnel to connect with (optional)' dropdown is set to 'vpc_name_1_10_tunnel'.

Figure 29: Existing Machine

6. If you selected a Nutanix account, then select **Windows** or **Linux** from the **Operating System** dropdown menu.
7. If you have configured an environment during the project creation, then under the Preset VM Config section, click **Clone from environment** to autofill the VM configuration details. This step is optional.
 The **Clone from environment** button appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu in the app profile. The option does not appear if you select **All Project Accounts** as your environment.
 You can also click the **View Configuration** option to review the configuration details before cloning the environment.
8. In the **Cluster** dropdown menu, select the cluster you want to associate to the service.
 The **Cluster** dropdown menu displays the clusters that you allowed in the project.
 The VLAN subnets have direct association with the cluster. When you select a VLAN subnet under the **Network Adapters (NICs)** section, the associated cluster is auto-populated in the **Cluster** dropdown menu. However, if you intend to use overlay subnets, you must select the cluster in list.
 If you mark the cluster as runtime editable, the selected subnets also become runtime editable.
9. Under the **VM Configuration** section, enter the name of the VM in the **VM Name** field.
 You can use Self-Service macros to provide a unique name to the VM. For example, `vm-@@{calm_array_index}@@-@@{calm_time}@@`. For more information on Self-Service macros, see [Macros Overview](#) on page 91.
10. Configure the processing unit of the VM by entering the number of vCPU, cores of each vCPU, and total memory in GB of the VM in the **vCPU**, **cores per vCPU**, and **Memory (GiB)** fields.

11. To create automated tasks or actions that you want to run before the VM is powered on, do the following:

The Post-create stage is a stage between the creation of VM and package installation. Post-create allows addition of automated tasks and actions that run on substrate without any manual intervention before the VM is powered on.

You can use Post-create to carry out tasks, such as:

- To use the Mac address of the VM for further orchestration in the pre-boot stage.
 - To place the provisioned VMs in a security group before they are powered on.
- a. In the Inspector Panel, select the **Off** radio button next to the **VM Power State** to keep the VM in the powered off state after creation.



Figure 30: VM Power State

Note: If you do not select the **Off** option, the blueprint skips any post-create tasks or actions you configured.

- b. In the Overview panel, click **Post-create** for the service you selected.

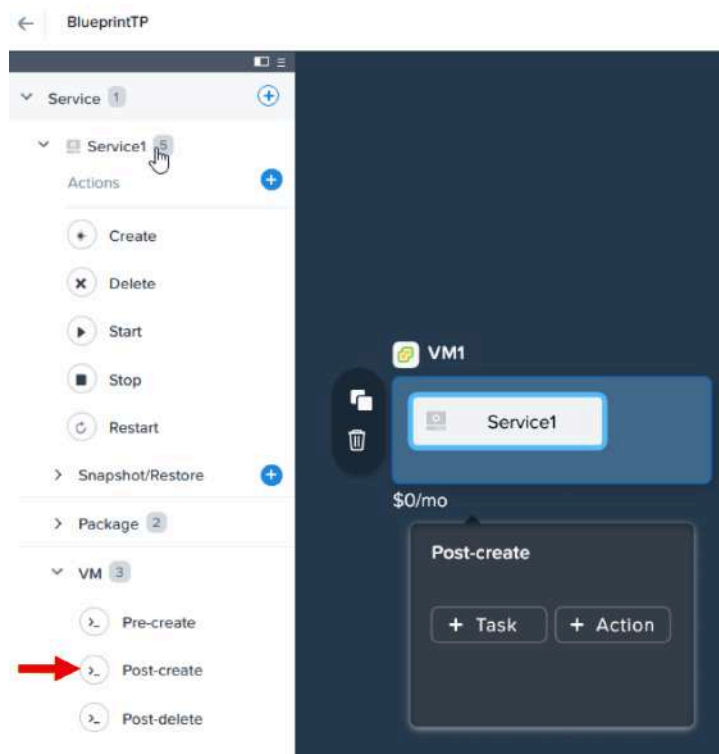


Figure 31: Post-create

- c. Add and configure task or actions that you want to run before the VM is powered on and establish the connection between the tasks or actions.

- d. Add a task to power on the VM.

The task to power on the VM must be the last task for post-create so that the VM can be powered on after all other tasks or actions are complete. In a multi-VM setup, ensure that you have at least one VM in the power on state for the App Status to be **On**.

12. (Optional) If you want to customize the default OS properties of the VM, select the **Guest Customization** checkbox.

Guest customization allows you to modify the properties of the VM operating system. You can prevent conflicts that might result due to the deployment of virtual machines with identical settings, such as duplicate VM names or same SID. You can also change the computer name or network settings by using a custom script.

- a. Select **Cloud-init** for Linux or **SysPrep** for Windows, and enter or upload the script in the **Script** panel.

For Sysprep, you must use double back slash for all escape characters . For example, \\v.

- b. For Sysprep script, click **Join a Domain** checkbox and configure the following fields.

- Enter the domain name of the Windows server in the **Domain Name** field.
- Select a credential for the Windows VM in the **Credentials** dropdown menu. You can also add new credentials.
- Enter the IP address of the DNS server in the **DNS IP** field.
- Enter the DNS search path for the domain in the **DNS Search Path** field.

13. Under the **DISKS** section, do the following:

- a. To add a disk, click the **+** icon next to **DISKS**.

- b. Select the device from the **Device Type** dropdown menu.
You can select **CD-ROM** or **DISK**.

- c. Select the device bus from the **Device Bus** dropdown menu.
You can select **IDE** or **SATA** for CD-ROM and **SCSI**, **IDE**, **PCI**, or **SATA** for DISK.

- d. From the **Operation** dropdown menu, select one of the following:

- » To allocate the disk memory from the storage container, select **Allocate on Storage Container**.
- » To clone an image from the disk, select **Clone from Image Service**.

- e. If you selected **Allocate on Storage Container**, enter the disk size in GB in the **Size (GiB)** field.

- f. If you selected **Clone from Image Service**, select the image you want to add to the disk in the **Image** field.

All the images that you uploaded to Prism Central are available for selection. For more information on image configuration, see [Image Management](#) section in the *Prism Central Infrastructure Guide*.

- g. To expand the size of a boot disk for the image, specify the size in the **Size (GiB)** field.

- h. Select **Bootable** for the image that you want to use to start the VM.

Note: You can add more than one disk and select the disk with which you want to boot up the VM.

14. Select one of the following firmwares to boot the VM.
 - » **Legacy BIOS:** Select legacy BIOS to boot the VM with legacy BIOS firmware.
 - » **UEFI:** Select UEFI to boot the VM with UEFI firmware. UEFI firmware supports larger hard drives, faster boot time, and provides more security features.
 - » To boot the VM with the Secure Boot feature of UEFI, select **Secure Boot**. Secure Boot ensures a safe and secure start by preventing unauthorized software such as a malware to take control during the VM bootup.
15. Under the **Categories** section, select a category in the **Key: Value** dropdown menu.

Use this option to tag your VM to a defined category in Prism Central. The list options are available based on your Prism Central configuration. If you want to protect your app by a protection policy, select the category defined for the policy in your Prism Central.
16. To add a network adapter, click the **+** icon next to the **Network Adapters (NICs)** field and select the subnet from the **NIC** dropdown menu.

The **NIC** dropdown menu shows all the VLAN and overlay subnets. The VLAN subnets have direct association with the cluster. Therefore, when you select a VLAN subnet, the associated cluster is auto-populated in the **Cluster** dropdown menu.

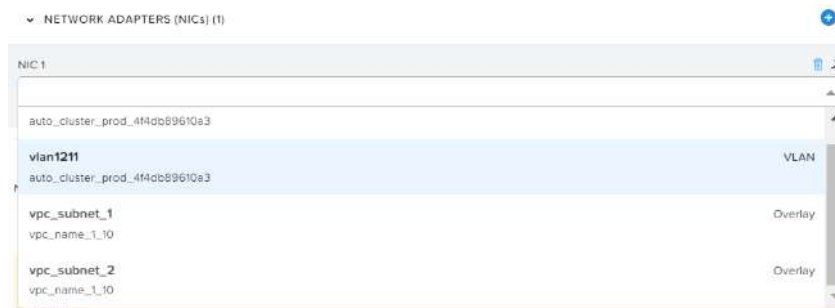


Figure 32: Network Adapter

- The NICs of a VM can either use VLAN subnets or overlay subnets. For example, if you select an overlay subnet in NIC 1 and then add NIC 2, the NIC 2 list displays only the overlay subnets. If you select a VLAN subnet in NIC 1, all subsequent VLAN subnets belong to the same cluster. Similarly, if you select an overlay subnet, all subsequent overlay subnets belong to the same VPC.
17. To add a serial port to the VM, click the **+** icon next to the **Serial Ports** field.

You can use serial ports to connect a physical port or a file on the VM.
 18. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.

19. On the **Package** tab, enter the package name in the **Package Name** field.
 - a. Click one of the following:
 - » To create a task to install a package, click **Configure install**.
 - » To create a task to uninstall a package, click **Configure uninstall**.
 - b. On the Blueprint Canvas, click **+ Task**.

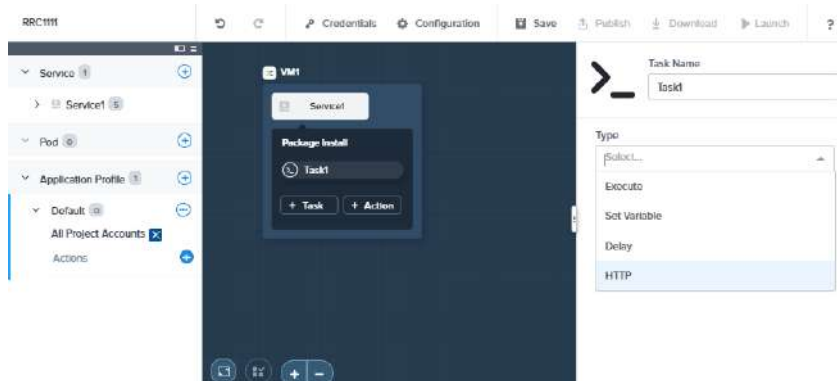


Figure 33: Package

- c. Enter the task name in the **Task Name** field.
20. To create a task, select the type of task from the **Type** dropdown menu. The available options are:
 - **Execute**: To create the **Execute** task type, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: To create the **Set Variable** task type, see [Adding a Set Variable Task](#) on page 179.
 - **HTTP**: To create the **HTTP** type, see [Adding an HTTP Task](#) on page 185.
 - **Delay** : To create the **Delay** task type, see [Adding a Delay Task](#) on page 186.

For **Execute** and **Set Variable** tasks, you can use endpoints as targets for script execution. For more information, see [Endpoints Overview](#) on page 386.

21. To reuse a task from the task library, do the following.
 - a. Click **Browse Library**.
 - b. Select the task from the task library.

When you select a task, the task inspector panel displays the selected task details.
 - c. Click **Select**.
 - d. Optionally, edit the variable or macro names as per your blueprint.

The variable or macro names used in the task can be generic, you can update them with corresponding variable or macro names as per your blueprint.
 - e. To update the variable or macro names, click **Apply**.
 - f. To copy the task, click **Copy**.

22. On the **Service** tab, under **Deployment Config**, enter the number of default, minimum and maximum service replicas that you want to create in the **Default**, **Min**, and **Max** fields respectively.

The **Min** and **Max** fields define the scale-in and scale-out actions. The scale-in and scale-out actions cannot scale beyond the defined minimum and maximum numbers. The default field defines the number of default replicas the service creates.

If there is an array of three VMs, define the minimum number as three.

Note: Min and Max fields require numeric values. Macros are not supported for these fields.

23. On the **Service** tab, do the following to define the variable for the service.
- In the **Variables** section, click the **+** icon.
 - In the **Name** field, enter the name of the variable you defined on the **Package** tab.
 - From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
 - If you have selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
 - Select the **Secret** checkbox to hide the value of the variable. This step is optional.
24. Add credentials to the blueprint. For more information, see [Adding Credentials](#) on page 173.
25. Click **Save** on the Blueprint Editor page.
The blueprint is saved and listed on the Blueprints page.

What to do next

Define the service dependencies. See [Setting up the Service Dependencies](#) on page 165.

Configuring VMware VM, Package, and Service

You can define the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app on a VMware platform.

Before you begin

- Ensure that you complete the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have created a project and configured an environment for VMware. For more information, see [Creating a Project](#) and [Configuring VMware Environment](#).
- You need licenses for both *Compute* and *Storage* distributed resource scheduler (DRS) in order to use the VMware DRS mode.
- Ensure that storage DRS is enabled and set to fully automated in vCenter.

Procedure

1. Perform the basic blueprint setup, and add a service. For more information, see [Adding a Service](#) on page 130.

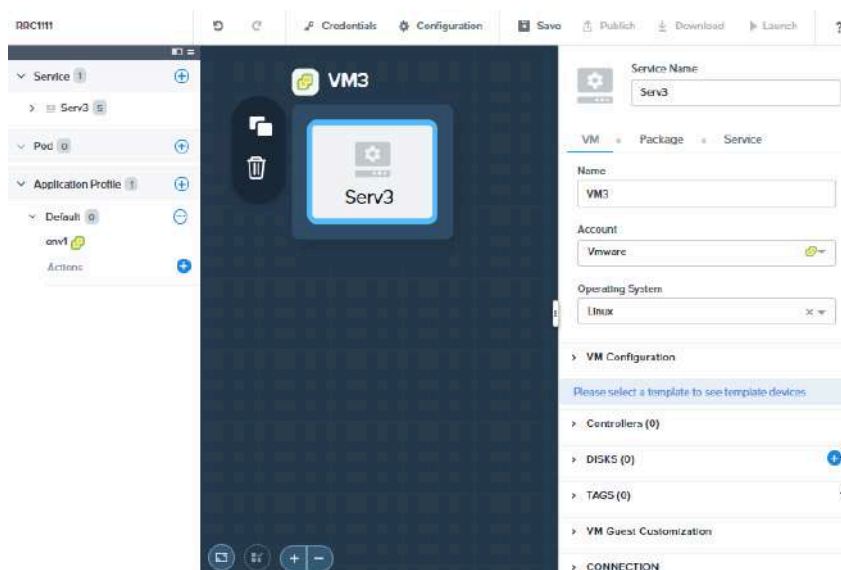


Figure 34: Blueprint Configuration

2. Enter a name of the service in the **Service Name** field.
3. On the **VM** tab, enter the name of the VM in the **Name** field.
4. Select **VMware** from the **Account** dropdown menu.

Note: The account options depend on the project you selected while setting up the blueprint.

5. Select **Windows** or **Linux** from the **Operating System** dropdown menu.
6. (Optional) If you have configured an environment during the project creation, then click **Clone from environment** to autofill the VM configuration details.
The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu in the app profile. The option does not appear if you select **All Project Accounts** as your environment.
You can also click the **View Configuration** option to review the configuration details before cloning the environment.
7. Select the **Compute DRS Mode** checkbox to enable load sharing and automatic VM placement. Distributed Resource Scheduler (DRS) is a utility that balances computing workloads with available resources in a virtualized environment. For more information on DRS mode, see the *VMware documentation*.
 - » If you selected **Compute DRS Mode**, then select the cluster where you want to host your VM from the **Cluster** dropdown menu.
 - » If you have not selected **Compute DRS Mode**, then select the host name of the VM from the **Host** dropdown menu.

8. Do one of the following:

- » Select **VM Templates** and then select a template from the **Template** dropdown menu.

Templates allow you to create multiple virtual machines with the same characteristics, such as resources allocated to CPU and memory or the type of virtual hardware. Templates save time and avoid errors when configuring settings and other parameters to create VMs. The VM template retrieves the list options from the configured vCenter.

Note:

- Install the VMware Tools on the Windows templates. For Linux VMs, install *Open-vm-tools* or *VMware-tools* and configure the *Vmtoolsd* service for automatic start-up.
- Support for *Open-vm-tools* is available. When using *Open-vm-tools*, install Perl for the template.
- Do not use SysPrepped as the Windows template image.
- If you select a template that has unsupported version of VMware Tools, then a warning appears stating VMware tool or version is unsupported and could lead to VM issues.
- You can also edit the NIC type when you use a template.

For more information, refer to VMware KB articles.

- » Select **Content Library**, a content library in the **Content Library** dropdown menu, and then select an OVF template or VM template from the content library.

A content library stores and manages content (VMs, vApp templates, and other types of files) in the form of library items. A single library item can consist of one file or multiple files. For more information on the vCenter content library, see the *VMware Documentation*.

Caution: Content Library support is currently a technical preview feature in Self-Service. Do not use any technical preview features in a production environment.

9. If you want to use the storage DRS mode, then select the **Storage DRS Mode** checkbox and a datastore cluster from the **Datastore Cluster** dropdown menu.

The datastore clusters are referred as storage pod in vCenter. A datastore cluster is a collection of datastores with shared resources and a shared management interface.

10. If you do not want to use storage DRS mode, then do not select the **Storage DRS Mode** checkbox, and select a datastore from the **Datastore** dropdown menu.

11. In the **VM Location** field, specify the location of the folder in which the VM must be created when you deploy the blueprint. Ensure that you specify a valid folder name already created in your VMware account.

To create a subfolder in the location you specified, select the **Create a folder/directory structure here** checkbox and specify a folder name in the **Folder/Directory Name** field.

Note: Self-Service gives preference to the VM location specified in the environment you select while launching an app. For example, you specify a subfolder structure as the VM location in the blueprint and the top-level folder in the environment. When you select this environment while launching your app, Self-Service considers the VM location you specified in the environment and creates the VM at the top-level folder.

Select the **Delete empty folder** checkbox to delete the subfolder created within the specified location, in case the folder does not contain any VM resources. This option helps you to keep a clean folder structure.

12. Enter the instance name of the VM in the **Instance Name** field.

This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.

13. To create automated tasks or actions that you want to run before the VM is powered on, do the following:

The Post-create stage is a stage between the creation of VM and package installation. Post-create allows addition of automated tasks and actions that run on substrate without any manual intervention before the VM is powered on.

You can use Post-create to carry out tasks, such as:

- To use the Mac address of the VM for further orchestration in the pre-boot stage.
 - To place the provisioned VMs in a security group before they are powered on.
- a. In the Inspector Panel, select the **Off** radio button next to the **VM Power State** to keep the VM in the powered off state after creation.

Note: If you do not select the **Off** option, the blueprint skips any post-create tasks or actions you configured.

- b. In the Overview panel, click **Post-create** for the service you selected.

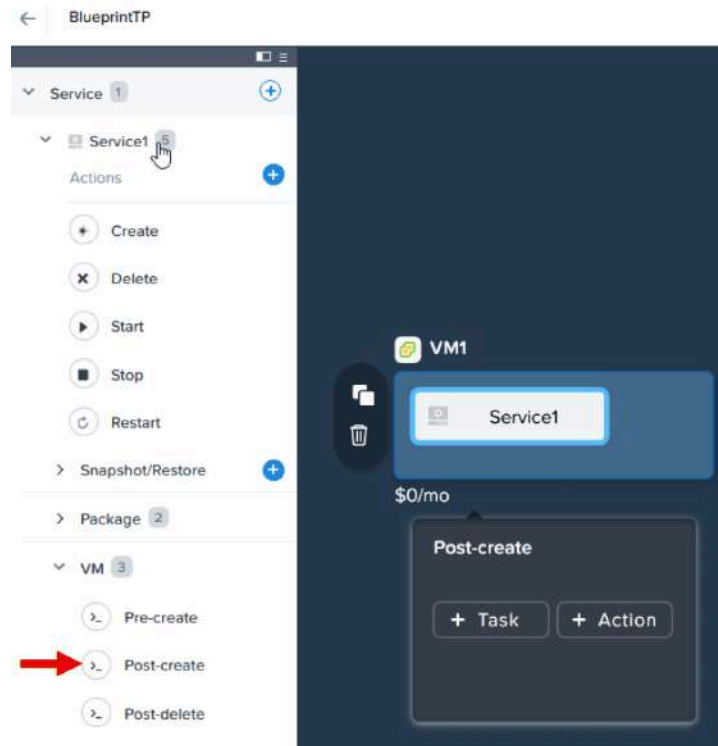


Figure 35: Post-create

- c. Add and configure task or actions that you want to run before the VM is powered on and establish the connection between the tasks or actions.
- d. Add a task to power on the VM.
- The task to power on the VM must be the last task for post-create so that the VM can be powered on after all other tasks or actions are complete. In a multi-VM setup, ensure that you have at least one VM in the power on state for the App Status to be **On**.

14. Under **VM Config**, do the following:

- a. Select the **CPU Hot Add** checkbox if you want to increase the VCPU count of a running VM. Support for CPU Hot Add depends on the Guest OS of the VM.
- b. Update the **vCPUs** and **Core Per Socket** count.
The number of sockets is calculated by dividing the total number of vCPUs by Cores Per Socket. For example, if the number of vCPUs is 4 and the Cores Per Socket is 2, then the number of sockets will be 2.
- c. Select the **Memory Hot Plug** checkbox if you want to increase the memory of a running VM. Support for Memory Hot Plug depends on the Guest OS of the VM.
- d. Update the memory in the **Memory** field.

15. Under **Controllers**, click the **+** icon to add the type of controller.

You can select either SCSI or SATA controller. You can add up to three SCSI and four SATA controllers.

16. Under the **Disks** section, click the **+** icon to add vDisks and do the following:

- a. Select the device type from the **Device Type** dropdown menu.
You can either select **CD-ROM** or **DISK**.
- b. Select the adapter type from the **Adapter Type** dropdown menu.
You can select **IDE** for CD-ROM.
You can select **SCSI**, **IDE**, or **SATA** for DISK.
- c. Enter the size of the disk in GiB.
- d. In the **Location** field, select the disk location.
- e. If you want to add a controller to the vDisk, select the type of controller in the **Controller** dropdown menu to attach to the disk.

Note: You can add either SCSI or SATA controllers. The available options depend on the adapter type.

f. In the **Disk mode** dropdown menu, select the type of the disk mode. Your options are:

- » **Dependent:** Dependent disk mode is the default disk mode for the vDisk.
- » **Independent - Persistent:** Disks in persistent mode behave like conventional disks on your physical computer. All data written to a disk in persistent mode are written permanently to the disk.
- » **Independent - Nonpersistent:** Changes to disks in nonpersistent mode are discarded when you shut down or reset the virtual machine. With nonpersistent mode, you can restart the virtual machine with a virtual disk in the same state every time. Changes to the disk are written to and read from a redo log file that is deleted when you shut down or reset.

You can also mark the vDisks runtime editable so you can add, delete, or edit the vDisks while launching the blueprint. For more information on runtime editable attributes, see [Runtime Variables Overview](#) on page 104.

17. Under the **Tags** section, select tags from the **Category: Tag pairs** field.

You can assign tags to your VMs so you can view the objects associated with your VMs in your VMware account. For example, you can create a tag for a specific environment and assign the tag to multiple VMs. You can then view all the VMs that are associated with the tag.

18. (Optional) If you want to customize the default OS properties of the VM, then click the **Enable** checkbox under **VM Guest Customization** and select a customization from the **Predefined Guest Customization** dropdown menu.
19. If you do not have any predefined customization available, select **None**.
20. Select **Cloud-init** or **Custom Spec**.
21. If you selected **Cloud-init**, enter or upload the script in the **Script** field.
22. If you have selected **Custom Spec**, enter the details for the VM in the following fields:
 - a. Enter the hostname in the **Hostname** field.
 - b. Enter the domain in the **Domain** field.
 - c. Select timezone from the **Timezone** dropdown menu.
 - d. Select **Hardware clock UTC** checkbox to enable hardware clock UTC.
 - e. Click the **+** icon to add network settings.
To automatically configure DHCP server, enable the **Use DHCP** checkbox and then skip to the **DNS Setting** section.
 - f. Enter a name for the network configuration you are adding to the VM in the **Setting name** field.
Settings name is the saved configuration of your network that you want to connect to your VM.
 - g. Enter values in the **IP Address**, **Subnet Mask**, **Default Gateway**, and **Alternative Gateway** fields.
 - h. Under the **DNS Settings** section, enter the DNS primary, DNS secondary, DNS tertiary, and DNS search path name.
23. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.
24. On the **Package** tab, enter the package name in the **Package Name** field.
 - a. Click one of the following:
 - **Configure install**: To create a task to install a package.
 - **Configure uninstall**: To create a task to uninstall a package.
 - b. On the Blueprint Canvas, click **+ Task**.
 - c. Enter the task name in the **Task Name** field.
25. To create a task, select the type of task from the **Type** dropdown menu.
The available options are:
 - **Execute**: To create the **Execute** type of task, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: To create the set variable type of task, see [Adding a Set Variable Task](#) on page 179.
 - **HTTP**: To create the HTTP task type, see [Adding an HTTP Task](#) on page 185.
 - **Delay**: To create the Delay task type, see [Adding a Delay Task](#) on page 186.

For **Execute** and **Set Variable** tasks, you can use endpoints as targets for script execution. For more information, see [Endpoints Overview](#) on page 386.

26. To reuse a task from the task library, do the following.

- a. Click **Browse Library**.
- b. Select the task from the task library.
When you select a task, the task inspector panel displays the selected task details.
- c. Click **Select**.
- d. Optionally, edit the variable or macro names as per your blueprint.
The variable or macro names used in the task can be generic, you can update them with corresponding variable or macro names as per your blueprint.
- e. To update the variable or macro names, click **Apply**.
- f. To copy the task, click **Copy**.

27. On the **Service** tab, under **Deployment Config**, enter the number of default, minimum and maximum service replicas that you want to create in the **Default**, **Min**, and **Max** fields respectively.

The **Min** and **Max** fields define the scale-in and scale-out actions. The scale-in and scale-out actions cannot scale beyond the defined minimum and maximum numbers. The default field defines the number of default replicas the service creates.

If there is an array of three VMs, define the minimum number as three.

Note: Min and Max fields require numeric values. Macros are not supported for these fields.

28. On the **Service** tab, do the following to define the variable for the service.

- a. In the **Variables** section, click the **+** icon.
- b. In the **Name** field, enter the name of the variable you defined on the **Package** tab.
- c. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
- d. If you have selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
- e. Select the **Secret** checkbox to hide the value of the variable. This step is optional.

29. Add credentials to the blueprint. For more information, see [Adding Credentials](#) on page 173.

30. Click **Save** on the Blueprint Editor page.

What to do next

Define the service dependencies. For more information, see [Setting up the Service Dependencies](#) on page 165.

Supported VMware Guest Tools Versions

To know the supported VMware guest tools versions, see the *VMware Product Interoperability Matrices*.

Configuring AWS VM, Package, and Service

You can define the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app on an AWS platform.

Before you begin

- Ensure that you have completed the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have created a project and configured an environment for AWS. For more information, See [Creating a Project](#) and [Configuring AWS Environment](#).

Procedure

1. Perform the basic blueprint setup, and add a service. For more information, see [Adding a Service](#) on page 130.

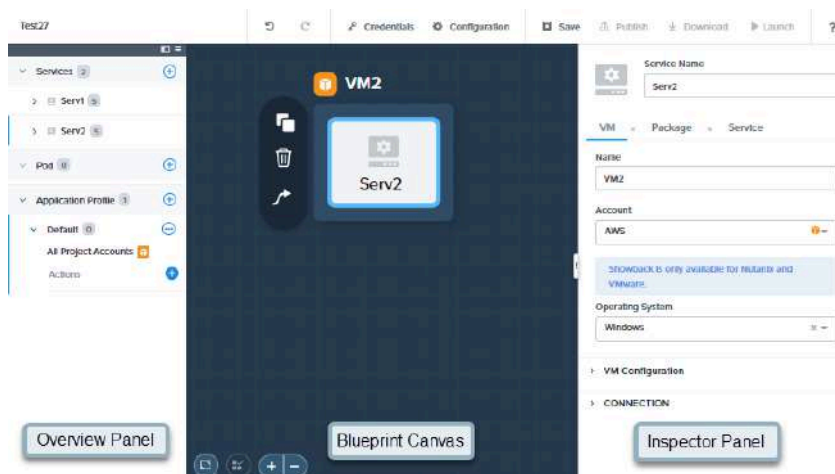


Figure 36: Blueprint Configuration

2. Enter a name of the service in the **Service Name** field.
3. On the **VM** tab, enter a name for the VM in the **Name** field.
4. Select the AWS account from the **Account** dropdown menu.

Note: The account options depend on the project you selected while setting up your blueprint.

5. Select **Windows** or **Linux** from the **Operating System** dropdown menu.
6. (Optional) If you have configured an environment during the project creation, then click **Clone from environment** to autofill the VM configuration details.

The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu in the app profile. The option does not appear if you select **All Project Accounts** as your environment.

You can also click the **View Configuration** option to review the configuration details before cloning the environment.

7. Edit the VM name in the **Instance Name** field.
This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
8. Select the **Associate Public IP Address** checkbox to associate a public IP address with your AWS instance.

If you do not select the **Associate Public IP Address** checkbox, ensure that the AWS account and Self-Service are on the same network for the scripts to run.

9. Select an AWS instance type from the **Instance Type** dropdown menu.

Instance types include varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to select the appropriate mix of resources for your apps. Each instance type includes one or more instance sizes that allows you to scale your resources to the requirements of your target workload.

The list displays the instances that are available in the AWS account. For more information, see AWS documentation.

10. Select a region from the **Region** dropdown menu and do the following:

Note: The list displays the regions that are selected while configuring the AWS setting.

- a. Select an availability zone from the **Availability Zone** dropdown menu.

An availability zone is one or more discrete datacenters with redundant power, networking, and connectivity in an AWS region. Availability zones allow you to operate production apps and databases that are more highly available, fault tolerant, and scalable than would be possible from a single datacenter.

- b. Select a machine image from the **Machine Image** dropdown menu.

An Amazon Machine Image is a special type of virtual appliance that is used to create a virtual machine within the Amazon Elastic Compute Cloud. It serves as the basic unit of deployment for services delivered using EC2.

- c. Select an IAM role from the **IAM Role** dropdown menu.

An IAM role is an AWS Identity and Access Management entity with permissions to make AWS service requests.

- d. Select a key pair from the **Key Pairs** dropdown menu.

A key pair (consisting of a private key and a public key) is a set of security credentials that you use to prove your identity when connecting to an instance.

- e. Select the VPC from the **VPC** dropdown menu and do the following:

Amazon Virtual Private Cloud (Amazon VPC) allows you to provision a logically isolated section of the AWS cloud where you can launch AWS resources in your defined virtual network.

11. Enter AWS tags in the **AWS Tags** field.

AWS tags are key and value pair to manage, identify, organize, search for, and filter resources. You can create tags to categorize resources by purpose, owner, environment, or other criteria.

12. Under the **Storage** section, configure the following to boot the AWS instance with the selected image.

- a. From the **Device** dropdown menu, select the device to boot the AWS instance.

The available options are based on the image you have selected.

- b. In the **Size(GiB)** field, enter the required size for the bootable device.

- c. From the **Volume Type** dropdown menu, select the volume type. You can select either **General Purpose SSD**, **Provisioned IOPS SSD**, and **EBS Magnetic HDD**.

For more information on the volume types, see AWS documentation.

- d. Optionally, select the **Delete on termination** checkbox to delete the storage when the instance is terminated.

You can also add more secondary storages by clicking the + icon next to the **Storage** section.

13. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.

14. On the **Package** tab, enter the package name in the **Package Name** field.

a. Click one of the following:

- **Configure install**: To create a task to install a package.
- **Configure uninstall**: To create a task to uninstall a package.

b. On the Blueprint Canvas, click **+ Task**.

c. Enter the task name in the **Task Name** field.

15. To create a task, select the type of task from the **Type** dropdown menu.

The available options are:

- **Execute**: To create the execute type of task, see [Adding an Execute Task](#) on page 176.
- **Set Variable**: To create the set variable type of task, see [Adding a Set Variable Task](#) on page 179.
- **HTTP**: To create the HTTP Task type, see [Adding an HTTP Task](#) on page 185.
- **Delay**: To create the Delay task type, see [Adding a Delay Task](#) on page 186.

For **Execute** and **Set Variable** tasks, you can use endpoints as targets for script execution. For more information, see [Endpoints Overview](#) on page 386.

16. To reuse a task from the task library, do the following.

a. Click **Browse Library**.

b. Select the task from the task library.

When you select a task, the task inspector panel displays the selected task details.

c. Click **Select**.

d. Optionally, edit the variable or macro names as per your blueprint.

The variable or macro names used in the task can be generic, you can update them with corresponding variable or macro names as per your blueprint.

e. To update the variable or macro names, click **Apply**.

f. To copy the task, click **Copy**.

17. On the **Service** tab, under **Deployment Config**, enter the number of default, minimum and maximum service replicas that you want to create in the **Default**, **Min**, and **Max** fields respectively.

The **Min** and **Max** fields define the scale-in and scale-out actions. The scale-in and scale-out actions cannot scale beyond the defined minimum and maximum numbers. The default field defines the number of default replicas the service creates.

If there is an array of three VMs, define the minimum number as three.

Note: Min and Max fields require numeric values. Macros are not supported for these fields.

18. On the **Service** tab, do the following to define the variable for the service.
 - a. In the **Variables** section, click the **+** icon.
 - b. In the **Name** field, enter the name of the variable you defined on the **Package** tab.
 - c. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
 - d. If you have selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
 - e. Select the **Secret** checkbox to hide the value of the variable. This step is optional.
19. Add credentials to the blueprint. For more information, see [Adding Credentials](#) on page 173.
20. Click **Save** on the Blueprint Editor page.
The blueprint is saved and listed on the Blueprints page.

What to do next

You can define the service dependencies. For more information, see [Setting up the Service Dependencies](#) on page 165. You can also add pre-create, post-create, or post-delete tasks. For more information, see [Adding a Pre-create, Post-create, or Post-delete Task](#) on page 187.

Configuring GCP VM, Package, and Service

You can define the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app on a GCP platform.

Before you begin

- Ensure that you complete the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have created a project and configured an environment for AWS. For more information, see [Creating a Project](#) and [Configuring GCP Environment](#).

Procedure

1. Perform the basic blueprint setup, and add a service. For more information, see [Adding a Service](#) on page 130.

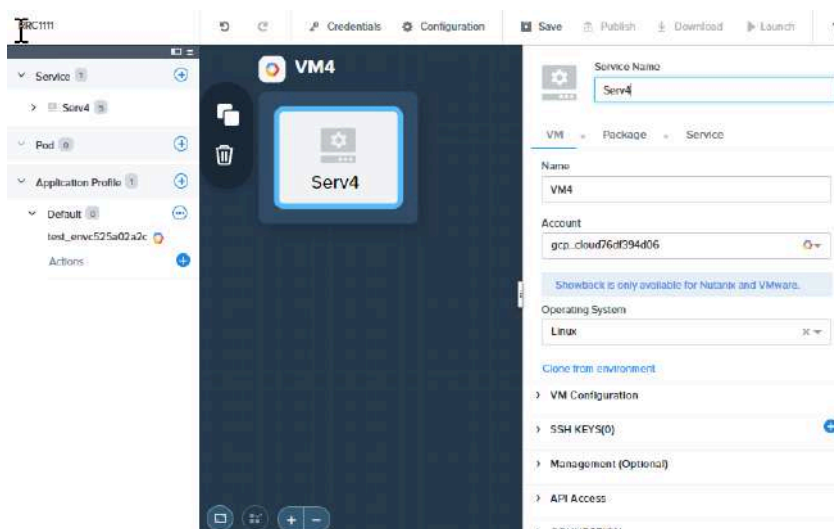


Figure 37: Blueprint Configuration

2. Enter a name of the service in the **Service Name** field.
3. On the **VM** tab, enter the name of the VM in the **Name** field.
4. Select a GCP account from the **Account** dropdown menu.

Note: The account options depend on the selected project while setting up the blueprint.

5. Select **Windows** or **Linux** from the **Operating System** dropdown menu.
6. (Optional) If you have configured an environment during the project creation, then click **Clone from environment** to autofill the VM configuration details.

The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu in the app profile. The option does not appear if you select **All Project Accounts** as your environment.

You can also click the **View Configuration** option to review the configuration details before cloning the environment.

7. (Optional) Edit the VM name in the **Instance Name** field.
This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
8. Select a zone from the **Zone** dropdown menu.
A zone is a physical location where you can host the VM.
9. Select the type of machine from the **Machine type** dropdown menu.
The machine types are available based on your zone. A machine type is a set of virtualized hardware resources available to a virtual machine (VM) instance, including the system memory size, virtual CPU (vCPU) count, and persistent disk limits. In Compute Engine, machine types are grouped and curated by families for different workloads.

10. Under the **DISKS** section, click the **+** icon to add a disk.
You can also mark the added vDisks runtime editable so you can add, delete, or edit the vDisks while launching the blueprint. For more information on runtime editable attributes, see [Runtime Variables Overview](#) on page 104.
11. To use an existing disk configuration, select the **Use existing disk** checkbox, and then select the persistent disk from the **Disk** dropdown menu.
12. If you have not selected the **Use existing disk** checkbox, then do the following:
 - a. Select the type of storage from the **Storage Type** dropdown menu. The available options are as follows.
 - » **pd-balanced**: Use this option as an alternative to SSD persistent disks with a balanced performance and cost.
 - » **pd-extreme**: Use this option to use SSD drives for high-end database workloads. This option has higher maximum IOPS and throughput and allows you to provision IOPS and capacity separately.
 - » **pd-ssd**: Use this option to use SSD drives as your persistent disk.
 - » **pd-standard**: Use this option to use HDD drives as your persistent disk.The persistent disk types are durable network storage devices that your instances can access like physical disks in a desktop or a server. The data on each disk is distributed across several physical disks.
 - b. Select the image source from the **Source Image** dropdown menu.
The images available for your selection are based on the selected zone.
 - c. Enter the size of the disk in GB in the **Size in GB** field.
 - d. To delete the disk configuration after the instance is deleted, select the **Delete when instance is deleted** checkbox under the **Disks** section.
13. To add a blank disk, click the **+** icon under the **Blank Disks** section and configure the blank disk.
14. To add networking details to the VM, click the **+** icon under the **Networking** section.
15. To configure a public IP address, select the **Associate Public IP address** checkbox and configure the following fields.
 - a. Select the network from the **Network** dropdown menu and the sub network from the **Subnetwork** dropdown menu.
 - b. Enter a name of the network in the **Access configuration Name** field and select the access configuration type from the **Access configuration type** dropdown menu.
These fields appear when you select the **Associate public IP Address** checkbox.
16. To configure a private IP address, clear the **Associate Public IP address** checkbox and select the network and sub network.
17. Under the **SSH Key** section, click the **+** icon and enter or upload the username key data in the **Username** field.
18. Select **Block project-wide SSH Keys** to enable blocking project-wide SSH keys.

19. Under the **Management** section, do the following:
 - a. Enter the metadata in the **Metadata** field.
 - b. Select the security group from the **Network Tags** dropdown menu.
Network tags are text attributes you can add to VM instances. These tags allow you to make firewall rules and routes applicable to specific VM instances.
 - c. Enter the key-value pair in the **Labels** field.
A label is a key-value pair that helps you organize the VMs created with GCP as the provider. You can attach a label to each resource, then filter the resources based on their labels.
20. Under the **API Access** section, do the following:
 - a. Specify the service account in the **Service Account** field.
 - b. Under Scopes, select **Default Access** or **Full Access**.
21. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.
22. On the **Package** tab, enter the package name in the **Name** field.
 - a. Click one of the following:
 - **Configure install**: To create a task to install a package.
 - **Configure uninstall**: To create a task to uninstall a package.
 - b. On the Blueprint Canvas, click **+ Task**.

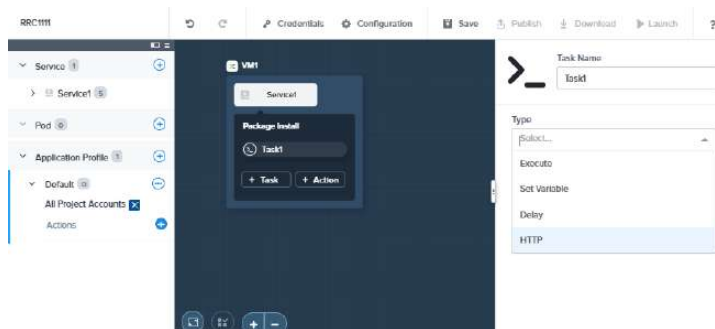


Figure 38: Package

- c. Enter the task name in the **Task Name** field.
23. To create a task, select the type of task from the **Type** dropdown menu.
The available options are:
 - **Execute**: To create the **Execute** task type, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: To create the **Set Variable** task type, see [Adding a Set Variable Task](#) on page 179.
 - **HTTP**: To create the **HTTP Task** type, see [Adding an HTTP Task](#) on page 185.
 - **Delay**: To create the **Delay** task type, see [Adding a Delay Task](#) on page 186.

For **Execute** and **Set Variable** tasks, you can use endpoints as targets for script execution. For more information, see [Endpoints Overview](#) on page 386.

24. To reuse a task from the task library, do the following.

- a. Click **Browse Library**.
- b. Select the task from the task library.
When you select a task, the task inspector panel displays the selected task details.
- c. Click **Select**.
- d. Optionally, edit the variable or macro names as per your blueprint.
The variable or macro names used in the task can be generic, you can update them with corresponding variable or macro names as per your blueprint.
- e. To update the variable or macro names, click **Apply**.
- f. To copy the task, click **Copy**.

25. On the **Service** tab, under **Deployment Config**, enter the number of default, minimum and maximum service replicas that you want to create in the **Default**, **Min**, and **Max** fields respectively.

The **Min** and **Max** fields define the scale-in and scale-out actions. The scale-in and scale-out actions cannot scale beyond the defined minimum and maximum numbers. The default field defines the number of default replicas the service creates.

If there is an array of three VMs, define the minimum number as three.

Note: Min and Max fields require numeric values. Macros are not supported for these fields.

26. On the **Service** tab, do the following to define the variable for the service.

- a. In the **Variables** section, click the **+** icon.
- b. In the **Name** field, enter the name of the variable you defined on the **Package** tab.
- c. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
- d. If you have selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
- e. Select the **Secret** checkbox to hide the value of the variable. This step is optional.

27. Add credentials to the blueprint. For more information, see [Adding Credentials](#) on page 173.

28. Click **Save** on the Blueprint Editor page.
The blueprint is saved and listed on the Blueprints page.

What to do next

You can define the service dependencies. For more information, see [Setting up the Service Dependencies](#) on page 165. You can also add pre-create, post-create, or post-delete tasks. For more information, see [Adding a Pre-create, Post-create, or Post-delete Task](#) on page 187.

Configuring Azure VM, Package, and Service

You can define the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app on an Azure platform.

Before you begin

- Ensure that you have configured the following entities in the Azure account.
 - Resource Group
 - Availability set
 - Network Security Group
 - Application Security Group
 - Virtual Network
 - Vault Certificates
- Ensure that you complete the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have created a project and configured an environment for Azure. For more information, see [Creating a Project](#) and [Configuring Azure Environment](#).

Procedure

1. Perform the basic blueprint setup, and add a service. For more information, see [Adding a Service](#) on page 130.
2. Enter a name of the service in the **Service Name** field.
3. Under the **VM** tab, enter the name of the VM in the **Name** field.
4. Select **Azure** from the **Account** dropdown menu.

Note: The account options depend on the selected project while creating the blueprint.

5. Select **Windows** or **Linux** from the **Operating System** dropdown menu.
6. (Optional) If you have configured an environment during the project creation, then click **Clone from environment** to autofill the VM configuration details.

The **Clone from environment** option appears only when you select a specific environment you configured for the account from the **Environment** dropdown menu in the app profile. The option does not appear if you select **All Project Accounts** as your environment.

You can also click the **View Configuration** option to review the configuration details before cloning the environment.
7. Edit the VM name in the **Instance Name** field.

This field is pre-populated with a macro as suffix to ensure name uniqueness. The service provider uses this name as the VM name.
8. Select a resource group from the **Resource Group** dropdown menu or select the **Create Resource Group** checkbox to create a resource group.

Each resource in Azure must belong to a resource group. A resource group is simply a logical construct that groups multiple resources together so you can manage the resources as a single entity. For example, you can create or delete resources as a group that share a similar life cycle, such as the resources for an n-tier app.

The **Resource Group** dropdown menu displays the resource groups that are associated with the subscriptions you selected in your Azure account. In case you have not selected any subscriptions, Self-Service considers all the subscriptions that are available in the Azure service principal to display the resource groups. Each resource group in the list also displays the associated subscription.

9. If you selected a resource group from the **Resource Group** dropdown menu, then do the following:
 - a. Select the geographical location of the datacenter from the **Location** dropdown menu.
 - b. Select **Availability Sets** or **Availability Zones** from the **Availability Option** dropdown menu.

You can then select an availability set or availability zone. An availability set is a logical grouping capability to ensure that the VM resources are isolated from each other to provide High Availability if deployed within an Azure datacenter. An availability zone allows you to deploy your VM into different datacenters within the same region.
 - c. Select the hardware profile as per your hardware requirements from the **Hardware Profile** dropdown menu.

The number of data disks and NICs depends upon the selected hardware profile. For information on the sizes of Windows and Linux VMs, see *Windows and Linux Documentation*.
10. If you selected the **Create Resource Group** checkbox to create a resource group, then do the following:
 - a. Select a subscription associated to your Azure account in the **Subscription** field.
 - b. Enter a unique name for the resource group in the **Name** field.
 - c. Select the geographical location of the datacenter that you want to add to the resource group in the **Location** dropdown menu.
 - d. Under **Tags**, enter a key and value pair in the **Key** and **Value** fields respectively.

Tags are key and value pairs that enable you to categorize resources. You can apply a tag to multiple resource groups.
 - e. If you want to automatically delete a resource group that has empty resources while deleting an app, click the **Delete Empty Resource Group** checkbox.
 - f. Specify the location and hardware profile.
11. Under the **Secrets** section, click the **+** icon and do the following:
 - a. Enter a unique vault ID in the **Vault ID** field.
 - b. Under **Certificates**, click the **+** icon.
 - c. Enter the URL of the configuration certificate in the **URL** field.

The URL of the certificate is uploaded to the key vault as a secret.
 - d. Enter store in the **Store** field.
 - For Windows VMs, the Store field specifies the certificate store on the virtual machine to which the certificate is added. The specified certificate store is implicitly created in the LocalMachine account.
 - For Linux VMs, the certificate file is placed under the /var/lib/waagent directory. The format of the file name is <UppercaseThumbprint>.crt for the X509 certificate and <UppercaseThumbprint>.prv for private key. Both of these files are .pem formatted.
12. (For Windows) Select the **Provision Windows Guest Agent** checkbox.

This option indicates whether or not to provision the virtual machine agent on the virtual machine. When this property is not specified in the request body, the default behavior is to set it to true. This ensures that the VM Agent is installed on the VM, and the extensions can be added to the VM later.
13. (For Windows) To indicate that the VM is enabled for automatic updates, select the **Automatic OS Upgrades** checkbox.

14. Under the **Additional Unattended Content** section, click the + icon and do the following:

- a. Select a setting from the **Setting Name** dropdown menu.
You can select **Auto Logon** or **First Logon Commands**.

Note: Guest customization is applicable only on images that allows or support guest customization.

- b. Enter or upload the xml content. For more information, see [Sample Auto Logon and First Logon Scripts](#) on page 422.

15. Under the **WinRM Listeners** section, click the + icon and do the following:

- a. Select the protocol from the **Protocol** dropdown menu.
You can select **HTTP** or **HTTPS**.
- b. If you selected HTTPS, then select the certificate URL from the **Certificate URL** dropdown menu.

16. Under the **Storage Profile** section, select the **Use Custom Image** checkbox to use a custom VM image created in your subscription.

You can then select a custom image or publisher-offer-SKU-version from the **Custom Image** dropdown menu.

17. Under the **VM Image Details** section, select an image type in the **Source Image Type** dropdown menu.

You can select **Marketplace**, **Subscription**, or **Shared Image Gallery**.

- » If you selected **Marketplace**, then specify the publisher, offer, SKU, and version for the image.
- » If you selected **Subscription**, then select the custom image.
- » If you selected **Shared Image Gallery**, then select the gallery and the image.

18. Under the **OS Disk Details** section, do the following:

- a. Select the storage type from the **Storage Type** dropdown menu.
You can select **Standard HDD**, **Standard SSD**, or **Premium SSD**.
- b. Select a disk storage account from the **Disk Storage** dropdown menu.
This field is available only when the **Use Custom Image** checkbox is enabled.
- c. Select disk caching type from the **Disk Caching Type** dropdown menu.
You can select **None**, **Read-only**, or **Read write**.
- d. Select disk create option from the **Disk Create Option** dropdown menu.
You can select **Attach**, **Empty**, or **From Image**.

19. Under the **Data Disk** section, do the following:

- a. Select the storage type from the **Storage Type** dropdown menu.
You can select **Standard HDD**, **Standard SSD**, or **Premium SSD**.
- b. Select disk caching type from the **Disk Caching Type** dropdown menu.
You can select **None**, **Read-only**, or **Read write**.
- c. Enter the size in GB in the **Size** field.
- d. Enter disk logical unit number (LUN) in the **Disk LUN** field.

Note: The LUN value should be unique across data disk list.

20. Under the **Network Profile** section, add NICs as per your requirement and do the following for each NIC:

a. Select a security group from the **Security Group** dropdown menu.

b. Select app security groups from the **ASGs** dropdown menu.

Unlike Network Security Groups that are defined at the network level, Application Security Groups (ASGs) are logical entities that are defined at the app level. ASGs help to manage the VM security by grouping the VMs according to the apps that run on them and allow app-centric use of Network Security Groups. You can select multiple ASGs in a NIC.

Note:

- ASGs are supported when you clone your apps and are also supported in snapshot and restore.
- ASGs are not supported in VM configuration updates.

c. Select a virtual network from the **Virtual Network** dropdown menu.

d. Under **Public IP Config**, enter a name and select an allocation method.

e. Under **Private IP Config**, select an allocation method.

If you selected **Static** as the allocation method, then enter the private IP address in the **IP Address** field.

21. Optionally, enter tags in the **Tags** field.

22. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.

23. On the **Package** tab, enter the package name in the **Name** field.

a. Click one of the following:

- **Configure install:** To create a task to install a package.
- **Configure uninstall:** To create a task to uninstall a package.

b. On the Blueprint Canvas, click **+ Task**.

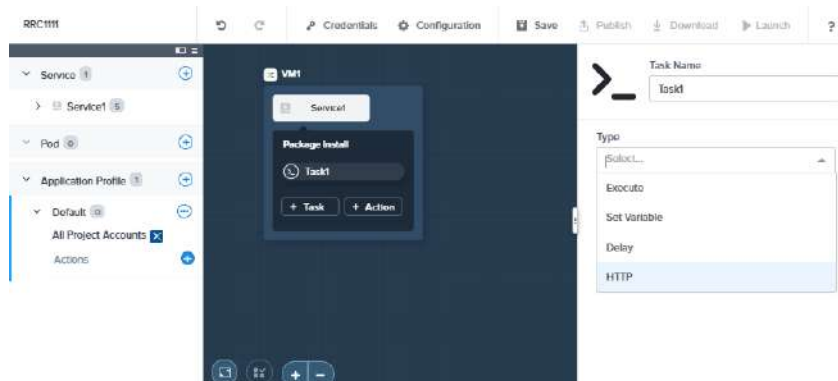


Figure 39: Package

c. Enter the task name in the **Task Name** field.

24. To create a task, select the type of task from the **Type** dropdown menu.

The available options are:

- **Execute**: To create the execute type of task, see [Adding an Execute Task](#) on page 176.
- **Set Variable**: To create the set variable type of task, see [Adding a Set Variable Task](#) on page 179.
- **HTTP**: To create the HTTP Task type, see [Adding an HTTP Task](#) on page 185.
- **Delay**: To create the Delay task type, see [Adding a Delay Task](#) on page 186.

For **Execute** and **Set Variable** tasks, you can use endpoints as targets for script execution. For more information, see [Endpoints Overview](#) on page 386.

25. To reuse a task from the library do the following.

- a. Click **Browse Library**.
- b. Select the task from the library.
When you select a task, the task inspector panel displays the selected task details.
- c. Click **Select**.
- d. Optionally, edit the variable or macro names as per your blueprint.
The variable or macro names used in the task can be generic, you can update them with corresponding variable or macro names as per your blueprint.
- e. Click **Apply** to update the variable or macro names.
- f. Click **Copy** to copy the task.

26. On the **Service** tab, under **Deployment Config**, enter the number of default, minimum and maximum service replicas that you want to create in the **Default**, **Min**, and **Max** fields respectively.

The **Min** and **Max** fields define the scale-in and scale-out actions. The scale-in and scale-out actions cannot scale beyond the defined minimum and maximum numbers. The default field defines the number of default replicas the service creates.

If there is an array of three VMs, define the minimum number as three.

Note: Min and Max fields require numeric values. Macros are not supported for these fields.

27. On the **Service** tab, do the following to define the variable for the service.

- a. In the **Variables** section, click the **+** icon.
- b. In the **Name** field, enter the name of the variable you defined on the **Package** tab.
- c. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
- d. If you have selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
- e. Select the **Secret** checkbox to hide the value of the variable. This step is optional.

28. Add credentials to the blueprint. For more information, see [Adding Credentials](#) on page 173.

29. Click **Save** on the Blueprint Editor page.

The blueprint is saved and listed under blueprints tab.

What to do next

You can define the service dependencies. For more information, see [Setting up the Service Dependencies](#) on page 165. You can also add pre-create, post-create, or post-delete tasks. For more information, see [Adding a Pre-create, Post-create, or Post-delete Task](#) on page 187.

Azure Troubleshooting

The following section describes Azure troubleshooting.

- For settings save or verification failure, you can check the logs at the following location.
`/home/calm/log/styx.log`
- For app blueprints save failure, you can check the logs at the following locations.
 - `/home/calm/log/hercules_*.log`
 - `/home/calm/log/styx.log`
- For provisioning failure, you can check the logs at the following locations.
 - Task logs on UI
 - `/home/epsilon/log/indra_*.log` [Signature: Encountered ERROR]
 - `/home/epsilon/log/durga_*.log`
 - `/home/epsilon/log/arjun_*.log`
 - `/home/calm/log/hercules_*.log`
 - `/home/calm/log/styx.log`

Configuring Nutanix Cloud VM, Package, and Service

You can define the underlying infrastructure of the VM, app details, and actions that are carried out on a blueprint until the termination of the app on Nutanix Cloud provider.

Before you begin

Ensure that you have configured DNS in the VPC section in the Nutanix Cloud dashboard in the Prism Central.

Procedure

1. Perform the basic blueprint setup, and add a service. For more information, see [Adding a Service](#) on page 130.
2. Enter a name of the service in the **Service Name** field.
3. On the **VM** tab, enter the name of the VM in the **Name** field.
4. Select **Xi** from the **Account** dropdown menu.

Note: The account options depend on the project you selected while setting up the blueprint.

The **Availability Zone** field is automatically filled.

5. Select **Windows** or **Linux** from the **Operating System** dropdown menu.

6. Under **VM Configuration**, enter the instance name of the VM in the **VM Name** field. This field displays the macro as suffix to ensure name uniqueness.
The service provider uses this name as the VM name.
7. In the **vCPUs** field, enter the required number of vCPUs for the VM.
8. In the **Cores per vCPU** field, enter the number of cores per vCPU for the VM.
9. In the **Memory** field, enter the required memory in GiB for the VM.
10. (Optional) If you want to customize the default OS properties of the VM, select the **Guest Customization** checkbox and do the following.
Guest customization allows you to upload custom scripts to modify the properties of the OS of the VM.
 - a. Select **Cloud-init** or **SysPrep** type and enter the script in the **Script** panel.

Note:

- Select Cloud-init for Linux and Sysprep for Windows. For Sysprep, you must use double back slash for all escape characters . For example, \\v.
- You can also upload the script by clicking the upload icon.

- b. For Sysprep script, click **Join a Domain** checkbox and configure the following fields.
 - **Domain Name:** Enter the domain name of the Windows server.
 - **Credentials:** From the **Credentials** dropdown menu, enter a credential for the Windows VM. You can also create new credentials. For more information, see step 22.
 - **DNS IP:** Enter the IP address of the DNS server.
 - **DNS Search Path:** Enter the DNS search path for the domain.
11. To add a vDisk, click **+ vDisks** and do the following.
 - a. Select the device type from the **Device Type** dropdown menu.
You can select **CD-ROM** or **Disk**.
 - b. Select the device bus from the **Device Bus** dropdown menu.
You can select **IDE** or **SATA** for **CD-ROM**.
You can select **SCSI**, **IDE**, **PCI**, or **SATA** for **Disk**.
 - c. Enter the size of the vDisk in GiB.
You can also make the vDisks as runtime editable. If you have marked the vDisk attribute as runtime editable, you can add, delete, or edit vDisks while launching the blueprint. For more information on runtime editable attributes, see [Runtime Variables Overview](#) on page 104.
12. Select categories from the **Categories** dropdown menu.

Note: Categories field allows you to tag your VM to a defined category in the Prism Central. Based on the Prism Central configuration, the list options are available.
13. Under **Network**, select the VPC from the **VPC** dropdown menu. For more information on VPC, see [Nutanix Cloud Infrastructure Service Administration Guide](#).
14. Configure the connection in your blueprint. For more information, see [Configuring Check Log-In](#) on page 174.

15. Under the **Package** tab, enter the package name in the **Name** field.

a. Click one of the following:

- **Configure install**: To create a task to install a package.
- **Configure uninstall**: To create a task to uninstall a package.

b. On the Blueprint Canvas, click **+ Task**.

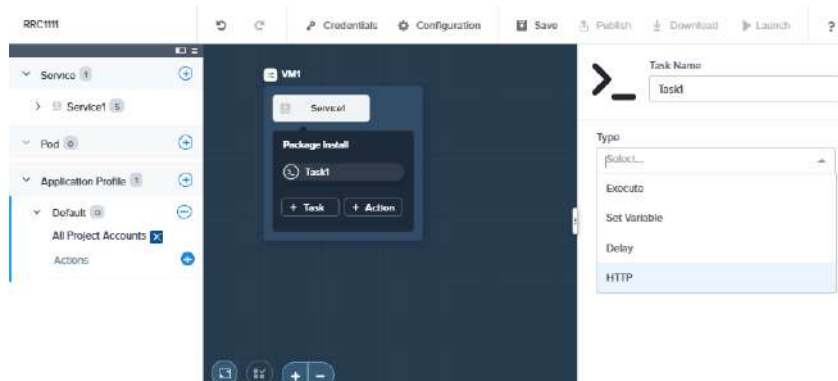


Figure 40: Package

c. Enter the task name in the **Task Name** field.

16. If you want to create a task, select the type of task from the **Type** dropdown menu.

The available options are:

- **Execute**: To create the **Execute** task type, see [Adding an Execute Task](#) on page 176.
- **Set Variable**: To create the **Set Variable** task type, see [Adding a Set Variable Task](#) on page 179.
- **HTTP**: To create the **HTTP Task** type, see [Adding an HTTP Task](#) on page 185.
- **Delay** : To create the **Delay** task type, see [Adding a Delay Task](#) on page 186.

17. (Optional) To reuse a task from the task library, do the following.

a. Click **Browse Library**.

b. Select the task from the task library.

When you select a task, the task inspector panel displays the selected task details.

c. Click **Select**.

d. (Optional) Edit the variable or macro names as per your blueprint.

The variable or macro names used in the task can be generic, you can update them with corresponding variable or macro names as per your blueprint.

e. To update the variable or macro names, click **Apply**.

f. To copy the task, click **Copy**.

18. On the **Service** tab, under **Deployment Config**, enter the number of default, minimum and maximum service replicas that you want to create in the **Default**, **Min**, and **Max** fields respectively.

The **Min** and **Max** fields define the scale-in and scale-out actions. The scale-in and scale-out actions cannot scale beyond the defined minimum and maximum numbers. The default field defines the number of default replicas the service creates.

If there is an array of three VMs, define the minimum number as three.

Note: Min and Max fields require numeric values. Macros are not supported for these fields.

19. On the **Service** tab, do the following to define the variable for the service.
 - a. In the **Variables** section, click the **+** icon.
 - b. In the **Name** field, enter the name of the variable you defined on the **Package** tab.
 - c. From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
 - d. If you have selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
 - e. Select the **Secret** checkbox to hide the value of the variable. This step is optional.
20. Click **Save** on the Blueprint Editor page.
The blueprint is saved and listed on the Blueprints page.

Configuring Kubernetes Deployment, Containers, and Service

Perform the following procedure to configure Kubernetes Deployment, Containers, and Service.

Before you begin

- Ensure that you have completed the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that the selected project has Kubernetes or GCP with GKE enabled or both as part of it.
- Refer to the [Kubernetes Documentation](#) to get information about the Kubernetes attributes and configuration.

Procedure


1. Add a service to the blueprint. For more information, see [Adding a Service](#) on page 130.
2. To add a Pod, click **+** against the **Pod**.

A Pod is the basic execution unit of a Kubernetes app and the smallest and simplest unit in the Kubernetes object model that you create or deploy. A Pod represents processes running on your cluster.

The Pod service inspector panel appears.
3. Enter a name of the pod in the **Pod Name** field.
4. Under the **Deployment** tab, select the account from the **Account** dropdown menu. All the accounts added to the project are available for selection.
5. Optionally, edit the Self-Service deployment name in the **Self-Service Deployment Name** field.
This field is auto-populated.

6. Optionally, edit the K8s deployment name in the **K8s Deployment Name** field.
This field is automatically populated.
7. Enter the namespace in the **Namespace** field.
A namespace is a Kubernetes field to use in environments with many users spread across multiple teams, or projects.
8. Enter the number of replicas in the **Replica** field.
9. Optionally, enter annotations in the **Annotations** field.
You can use Kubernetes annotations to attach arbitrary non-identifying metadata to objects.
10. Enter selector in the **Selectors** field.
The selector field defines how the Deployment finds which pods to manage.
11. Enter label in the **Label** field.
Labels are key/value pairs that are attached to objects, such as pods. You can use Labels to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. You can also use Labels to organize and to select subsets of objects. You can attach Labels to objects either at the creation time or later. Each object can have a set of key/value labels defined. Each key must be unique for a given object.
12. Optionally, you can edit the pod name in the **K8s Pod Name** field.
This field is auto-populated.
13. Enter value of image pull secrets in the **Image Pull Secrets** field.
You can provide the list of secret names (pre-configured in a Kubernetes cluster by using Kubernetes Docker secret object) to be used by Kubernetes cluster to pull the container images from registries that require authentication.
14. Select DNS policy from the **DNS Policy** dropdown menu.
15. Under **Containers** tab, optionally edit the Self-Service service name in the **Self-Service Service Name** field.
16. Optionally, you can edit the K8s service name in the **K8s Service Name** field.
This field is auto-populated.
17. Enter arguments for the container in the **Args** field.
18. Enter Docker image in the **Image** field.
19. Select a value from the **Image Pull Policy**.
You can either select **Never** or **Always** or **IfNotPresent** (default).
20. Under **Pre Stop Lifecycle**, select an action.
You can select **None** (default) or **Exec** or **HTTP Get Action** or **TCP Socket**.
This hook is called immediately before a container is terminated. It is blocking, meaning it is synchronous, so it must complete before the call to delete the container can be sent. No parameters are passed to the handler.
21. Under **Post Stop Lifecycle**, select an action.
You can select **None** (default) or **Exec** or **HTTP Get Action** or **TCP Socket**.
This hook runs immediately after a container is created. However, there is no guarantee that the hook runs before the container ENTRYPOINT. No parameters are passed to the handler.

22. Under **Container Port**, enter the port number in the **Port** field.
- Enter name of the port in the **Name** field.
 - Select protocol from the **Protocol** dropdown menu.
You can either select **TCP** or **UDP**.
23. Under **Readiness Probe**, enter command in the **Command** field.
The kubelet uses readiness probes to know when a Container is ready to start accepting traffic. A pod is ready after all of its Containers are ready. One use of this signal is to control the pods used as backends for Services. When a pod is not ready, it is removed from Service load balancers.
24. Under **Resource Limit**, enter the cores per CPU in the **CPU** field.
When you specify a pod, you can optionally specify how much CPU and memory (RAM) each Container needs. CPU and memory are each a resource type. A resource type has a base unit. You can specify CPU in units of cores and memory in bytes.
- Enter the bytes of memory in the **Memory** field.
25. Under **Resource Request**, enter the termination message path in the **Termination Message Path** field.
Termination messages provide a way for containers to write information about fatal events to a location where you can easily retrieve and surface these events by tools like dashboards and monitoring software.
26. Under **Service** tab, optionally you can edit the **Self-Service Published Service Name** field.
27. Optionally, you can edit the **K8s Service Name** field.
28. Select a service type from the **Service Type** dropdown menu. You can select one of the following.
- ClusterIP**: Exposes the service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster.
 - NodePort**: Exposes the service on each node's IP at a static port (the **NodePort**). A **ClusterIP** Service, to which the **NodePort** Service routes, is automatically created. You'll be able to contact the **NodePort** Service, from outside the cluster, by requesting **<NodeIP>:<NodePort>**.
 - LoadBalancer**: Exposes the service externally using a cloud provider's load balancer. **NodePort** and **ClusterIP** Services, to which the external load balancer routes, are automatically created.
29. Enter namespace in the **Namespace** field.
You can use Namespaces in environments with many users spread across multiple teams, or projects.
30. Enter label in the **Label** field.
Labels are key/value pairs that are attached to objects, such as pods. You can use Labels to specify identifying attributes of objects that are meaningful and relevant, but do not directly imply semantics to the core system. You can also use Labels to organize and select subsets of objects. You can attach Labels to objects at creation time and add or modify at any time. Each object can have a set of key/value labels defined. Each key must be unique for a given object.
31. Enter selector in the **Selectors** field.
The selector field defines how the Deployment finds which pods to manage.

32. Under **Port List**, enter the port name in the **Port Name** field.
 - a. Enter node port in the **Node Port** field.
 - b. Enter port number in the **Port** field.
 - c. Select protocol from the **Protocol** dropdown menu.
You can either select **TCP** or **UDP**.
 - d. Enter the target port in the **Target Port** field.
33. To upload the POD specification file in .JSON or .YAML format from your local machine, click against the  icon next to the **Pod Name** field and upload the specification file.
You can also download the POD specification file in .JSON or .YAML format.
34. To edit the uploaded POD specification, turn on **Spec Editor** and click **Edit**.
The **Script Editor** page is displayed. You can edit the specification file in .YAML or .JSON format.
35. To save the edited POD specification file, click **Done**.
36. Click **Save**.
The blueprint is saved and listed on the Blueprints page.

What to do next

Define the service dependencies. For more information, see [Setting up the Service Dependencies](#) on page 165.

Setting up the Service Dependencies

Dependencies are used to define the order in which tasks must get executed. Perform the following procedure to set up the service dependency.

Before you begin

- Ensure that at least more than one service must be available. See [Adding a Service](#) on page 130.
- Ensure that you have completed the pre-configuration requirements. See [Creating a Multi-VM Blueprint](#) on page 130.

Procedure

1. Select the service.

2. Select the dependency icon and drag to the service on which you want to create the dependency.

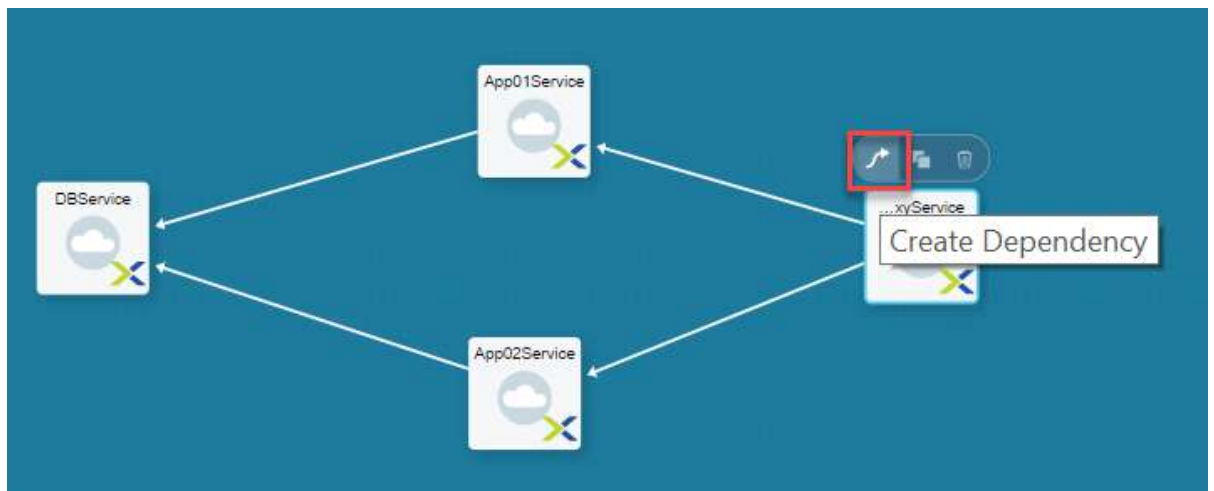


Figure 41: Create Dependency

What to do next

Configure the app profile. See [Adding and Configuring an App Profile](#) on page 166.

Adding and Configuring an App Profile

An app profile provides different combinations of the service, package, and VM while configuring a blueprint. You configure app profiles and use them while launching a blueprint.

Before you begin

Ensure that you have completed the pre-configuration requirements. See [Creating a Multi-VM Blueprint](#) on page 130.

Procedure

1. To create an app profile, click the **+** icon next to **Application Profile** in the Overview Panel.

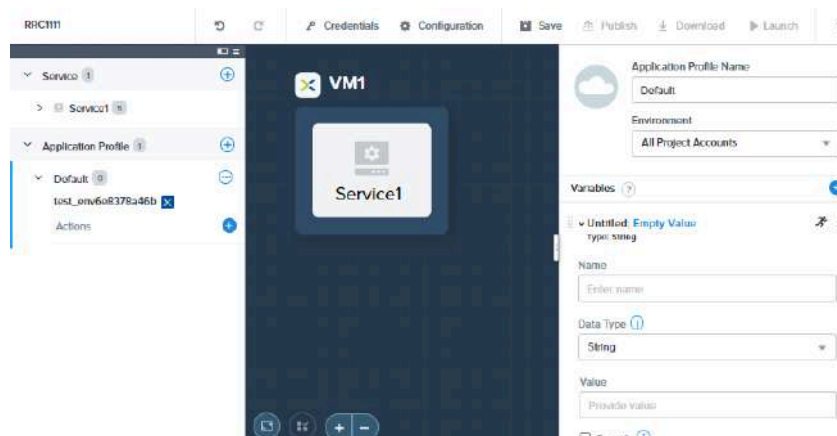


Figure 42: App Profile

2. In the Inspector Panel, enter the name of the app profile in the **Application Profile Name** field.

3. Optionally, select an environment for the app profile from the **Environment** dropdown menu.
The environment available for the selection in the **Environment** dropdown menu depends on the project you selected in the Blueprint Setup window. If you selected a default environment while configuring your environments for the project, the default environment automatically appears in the **Environment** dropdown menu. You can select a different environment if required.
4. Click the **+** icon next to **Variables**.
5. Enter a name for the variable in the **Name** field.
6. Select a data type from the **Data Type** dropdown menu. You can select one of the following data type:
 - » String
 - » Integer
 - » Multi-line string
 - » Date
 - » Time
 - » Date Time
7. Enter a value for the selected data type in the **Value** field.
You can select the **Secret** checkbox to hide the variable value.
8. Click **Show Additional Options**.
9. In the **Input Type** field, select one of the following input type:
 - Simple: Use this option for default value.
 - Predefined: Use this option to assign static values.
 - eScript: Use this option to attach a script that is run to retrieve values dynamically at runtime. Script can return single or multiple values depending on the selected base data type.
 - HTTP: Use this option to retrieve values dynamically from the defined HTTP end point. Result is processed and assigned to the variable based on the selected base data type.
10. If you have selected **Simple**, enter the value for the variable in the **Value** field.
11. If you have selected **Predefined**, enter the value for the variable in the **Option** field.
 - a. To add multiple values for the variable, click **+ Add Option**, and enter values in the **Option** field.

Note: To make any value as default, select **Default** for the option.

12. If you have selected **eScript**, do the following:
 - a. Select the Python version from the version drop-down menu. By default, Python 3 is selected as the version for you to enter the script.

Note: Based on the decision that the Python Software Foundation (PSF) took to discontinue support for Python 2, Nutanix has decided to end support for Python 2 on June 30, 2024. Nutanix recommends that you format all your new eScripts in Python 3 and update any existing eScripts from Python 2 to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

b. Enter the script or use the upload icon to upload the script in the **Script** field.
You can also upload the script from the library or from the computer by clicking the upload icon.
You can also publish the script to the library by clicking the publish button.

Note:

- You cannot add macros to eScripts.
- If you have selected Multiple Input (Array) checkbox with input type as eScript, then ensure that the script returns a list of values separated by comma. For example, CentOS, Ubuntu, Windows.

13. If you have selected **HTTP**, configure the following fields.

- **Request URL:** In the **Request URL** field, enter the URL of the server that you want to run the methods on.
- **Request Method:** In the **Request Method** dropdown menu, select one of the following request methods. The available options are GET, PUT, POST, and DELETE.
- **Request Body:** In the **Request Body** field, enter the PUT request. You can also upload the PUT request by clicking the upload icon.
- **Content Type:** In the **Content Type** dropdown menu, select the type of the output format. The available options are XML , JSON, and HTML.
- **Connection Timeout (sec):** In the **Connection Timeout (sec)** field, enter the timeout interval in seconds.
- **Authentication:** Optionally, if you have selected authentication type as BASIC, enter the user name and the password in the **User name** and **Password** fields respectively.
- **SSL Certificate Verification:** If you want to verify SSL certificate for the task, click the **SSL Certificate Verification** field.
- **Retry Count:** Enter the number of attempts the system performs to create a task after each failure. By default, the retry count is zero. It implies that the task creation procedure stops after the first attempt.
- **Retry Interval:** Enter the time interval in seconds for each retry if the task fails.
- **Headers:** In the Header area, enter the HTTP header key and value in the Key and Value fields respectively. If you want to publish the HTTP header key and value pair as secret, click the **Secrets** fields.
- **Response Code:** Enter the response code for the selected response status.
- **Response Status:** Select either Success or Failure as the response status for the task.
- **Set Response Path for Variable:** Enter the variables from the specified response path. The example of json format is \$.x.y and xml format is //x/y. For example, if the response path for variable is \$.[*].display for response.

```
[
  {
    "display": "HTML Tutorial",
    "url": "https://www.w3schools.com/html/default.asp"
  },
  {
    "display": "CSS Tutorial",
    "url": "https://www.w3schools.com/css/default.asp"
  }
]
```



```

    },
    {
      "display": "JavaScript Tutorial",
      "url": "https://www.w3schools.com/js/default.asp"
    },
    {
      "display": "jQuery Tutorial",
      "url": "https://www.w3schools.com/jquery/default.asp"
    },
    {
      "display": "SQL Tutorial",
      "url": "https://www.w3schools.com/sql/default.asp"
    },
    {
      "display": "PHP Tutorial",
      "url": "https://www.w3schools.com/php/default.asp"
    },
    {
      "display": "XML Tutorial",
      "url": "https://www.w3schools.com/xml/default.asp"
    }
  ]

```

Then, during the launch time the list options are ["HTML Tutorial", "CSS Tutorial", "JavaScript Tutorial", "jQuery Tutorial", "SQL Tutorial", "PHP Tutorial", "XML Tutorial"].

14. Optionally, enter a label and description for the variable.

15. Optionally, do the following:

- **Mark this variable private:** Select this to make the variable private. Private variables are not shown at launch or in the app.
- **Mark this variable mandatory:** Select this to make the variable a requisite for app launch.
- **Validate with Regular Expression:** Select this if you want to test the Regex values. Click **Test Regex**, provide the value for the Regex, and test or save the Regex. You can enter Regex values in PCRE format. For more details, see from <http://pcre.org/>.

16. Click **Save** on the Blueprint Editor page.

What to do next

After you added the app profile and its variables, you can create actions in the profile. For more information, see [Adding an Action to a Multi-VM Blueprint](#) on page 196.

BROWNFIELD APP OVERVIEW

Brownfield apps are imported to manage existing VMs that are currently not managed by Self-Service. To import a brownfield app, Self-Service must communicate with the VMs that are not managed by Self-Service. The app runs like any other Self-Service app after it is imported.

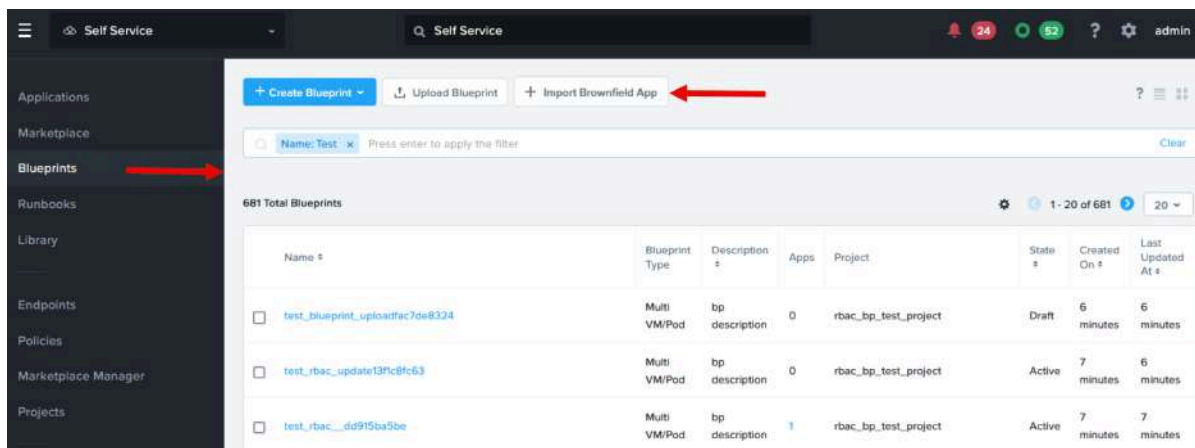


Figure 43: Brownfield Applications Page

Brownfield Applications - Key Points

The following are the key points you must consider before you import a brownfield app.

- You need administrator privileges to import a brownfield app.
- Self-Service supports brownfield import of only those Remote Prism Central account VMs that are associated with the **_internal** project.
- For the quota utilization check and VM update configuration to work accurately, you must either select a single VM per service or the VMs that have the same configuration.

In Self-Service, the update configuration is stored as a single element per service and applicable from the first VM instance. When you select multiple VMs with different configurations in a service and update the configuration, the update configuration applies to the first VM instance. The same configuration is then followed for all the remaining VM instances.

Let's say you selected VM1 and VM2 for the service with a RAM of 4 GB and 8 GB respectively. If you define the update configuration to increase the RAM by 1 GB and run the action, the update applies to VM1 to increase the RAM to 5 GB. The same configuration is then followed for VM2 to change the RAM from 8 GB to 5 GB causing undesirable results in both the update configuration and quota utilization checks.

- When you add credentials for the VMs, ensure that the credentials are same for all the VMs.
- After a VM is created, the VM takes some time to be listed for brownfield import.
- Brownfield apps do not support snapshot and restore.

For information on how to import a brownfield app, see [Importing Brownfield App](#) on page 170.

Importing Brownfield App

Brownfield apps are imported to manage existing VMs that are currently not managed by Self-Service. Perform the following procedure to import brownfield app.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. Click **Import Brownfield App**.
The **Brownfield Import** window is displayed.
5. Enter an app name in the **Name** field.
6. Optionally, enter a description about the app in the **Description** field.
7. Select a project from the **Project** dropdown menu.
8. Click **Proceed**.
The brownfield app editor page is displayed.
9. To add a service, click **+** next to the service.
10. Enter the service name in the **Service Name** field.
11. Select one of the following type of deployment.
 - Select **Greenfield** if all the existing VMs are managed by Self-Service.
 - Select **Brownfield** if the existing VMs are currently not managed by Self-Service.
12. If you have selected **Brownfield**, then do the following.
 - a. Click **Select Machines**, select the VMs from the Select The Machines window, and then click **Done**.

Note: For the quota utilization check and VM update configuration to work accurately, ensure that you either select a single VM or the VMs that have the same configuration.
 - b. Configure the connection. For more information, see [Configuring Check Log-In](#) on page 174.
 - c. Add credentials. For more information, see [Adding Credentials](#) on page 173.
13. If you have selected **Greenfield**, then configure the VM, package, and service. To configure VM, package, and service, see [Configure Multi-VM, Package, and Service](#) on page 131.
14. Click **Save**.
The brownfield app is imported and listed on the Blueprints page.

What to do next

Launch the brownfield app from the Blueprints page. For more information, see [Launching Brownfield App](#) on page 171.

Launching Brownfield App

You must launch the configured brownfield apps to be managed by Self-Service.

Before you begin

Ensure that you have imported the brownfield app. For more information, see [Importing Brownfield App](#) on page 170.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. On the Blueprints page, click the brownfield app that you want to launch.
The brownfield app details page is displayed.
5. Click **Launch**.
The brownfield app is launched and is listed on the My Apps page in Admin Center. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

BLUEPRINT CONFIGURATIONS IN SELF-SERVICE

Blueprint configuration involves adding tasks, actions, snapshot and restore configurations, and VM update configurations.

Configuring a Blueprint

Perform the following procedure to configure a blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Click the blueprint that you want to configure.
The blueprint editor page is displayed.
5. Click **Configuration**.
The **Blueprint Configuration** window is displayed.
6. In the **Blueprint Name** field, enter a name of the blueprint.
7. In the **Blueprint Description** field, enter a brief description about the blueprint.
8. Click **+** against the **Downloadable Image Configuration** field and configure the following:
 - a. In the **Package Name** field, enter the name of the package.
 - b. In the **Description** field, enter a brief description about the package.
 - c. In the **Image Name** field, enter the name of the image.
 - d. In the **Image Type** dropdown menu, select the type of image.
 - e. In the **Architecture** dropdown menu, select the architecture.
 - f. In the **Source URI** field, enter the source URI to download the image.
9. In the **Product Name** field, enter the name of the product.
10. In the **Product Version** field, enter the version of the product.
11. Click one of the following:
 - To save the configuration, click **Save**.
 - To go back to the previous screen, click **Back**.

Adding Credentials

Credentials are used to authenticate a user to access various services in Self-Service. Self-Service supports static and dynamic credentials with key-based and password-based authentication methods.

Procedure

1. To add a credential, do one of the following:
 - » To add a credential in a single-VM blueprint, click **Add/Edit Credentials** on the **Advanced Options** tab and then click **+ Add Credentials**.
 - » To add a credential in a multi-VM blueprint or a brownfield app, click **Credentials** on the Blueprint Editor page and then click **Credentials +**.
 - » To add a credential in a task, click **Add New Credential** in the **Credential** dropdown menu.
2. In the **Name** field, type a name for the credential.
3. Under the **Type** section, select the type of credential that you want to add.
 - » **Static**: Credentials store keys and passwords in the credential objects that are contained in the blueprints.
 - » **Dynamic**: Credentials fetch keys and passwords from an external credential store that you integrate with Self-Service as the credential provider.
4. In the **Username** field, type the user name.

For dynamic credentials, specify the @@(username)@@ that you defined while configuring the credential provider.

Note: A dynamic credential provider definition requires username and secret. The secret variable is defined by default when you configure your credential provider. However, you must configure a runbook in the dynamic credential provider definition for the username variable before you use the variable in different entities.
5. Select either **Password** or **SSH Private Key** as the secret type.
6. Do one of the following to configure the secret type.
 - » If you selected **Static** as the credential type and **Password** as the secret type, then type the password in the **Password** field.
 - » If you selected **Static** as the credential type and **SSH Private Key** as the secret type, then enter or upload the key in the **SSH Private Key** field.
 - » If you selected **Dynamic** as the credential type and **Password** or **SSH Private Key** as the secret type, then select a credential provider in the **Provider** field. After you select the provider, verify or edit the attributes defined for the credential provider.

If the private key is password protected, click **+Add Passphrase** to provide the passphrase. For dynamic credentials, you must configure a runbook in the dynamic credential provider definition for the passphrase variable and then use the @@{passphrase}@@ variable.

The type of SSH key supported is RSA. For information on how to generate a private key, see [Generating SSH Key on a Linux VM](#) on page 444 or [Generating SSH Key on a Windows VM](#) on page 445.
7. If you want this credential as your default credential, select the **Use as default** checkbox.
8. Click **Done** to add the credential.

Configuring Check Log-In

You configure a check log-in task to check whether you are able to SSH into the VM you create. Perform the following steps to configure check log-in.

Procedure

1. Under **Connection**, select the **Check log-in upon create & every subsequent power on** checkbox to check the log on status after creating the VM.
2. In the **Credential** dropdown menu, select **Add New Credential** to add a new credential and do the following:

- a. Enter a name for the credential in the **Name** field.
- b. Select the type of credential you want to add under the **Type** section. Your options are:
 - » **Static**: Credentials store keys and passwords in the credential objects that are contained in the blueprints.
 - » **Dynamic**: Credentials fetch keys and passwords from an external credential store that you integrate with Self-Service as the credential provider.
- c. Enter the user name in the **Username** field.
For dynamic credentials, specify the @@(username)@@ that you defined while configuring the credential provider.

Note: A dynamic credential provider definition requires username and secret. The secret variable is defined by default when you configure your credential provider. However, you must configure a runbook in the dynamic credential provider definition for the username variable to use in different entities.

- d. Select either **Password** or **SSH Private Key** as the secret type.
- e. Do one of the following to configure the secret type.
 - » If you selected **Static** as the credential type and **Password** as the secret type, then type the password in the **Password** field.
 - » If you selected **Static** as the credential type and **SSH Private Key** as the secret type, then enter or upload the key in the **SSH Private Key** field.
 - » If you selected **Dynamic** as the credential type and **Password** or **SSH Private Key** as the secret type, then select a credential provider in the **Provider** field. After you select the provider, verify or edit the attributes defined for the credential provider.

If the private key is password protected, click **+Add Passphrase** to provide the passphrase. For dynamic credentials, you must configure a runbook in the dynamic credential provider definition for the passphrase variable and then use the @@{passphrase}@@ variable.

The type of SSH key supported is RSA. For information on how to generate a private key, see [Generating SSH Key on a Linux VM](#) on page 444 or [Generating SSH Key on a Windows VM](#) on page 445.

- f. If you want this credential as your default credential, select the **Use as default** checkbox.
 - g. Click **Done**.
3. Select address from the **Address** dropdown menu.

You can either select the public IP address or private IP address of a NIC.

4. Select the connection from the **Connection Type** dropdown menu.

Select **SSH** for Linux or **Windows (Powershell)** for Windows.

If you selected **Windows (Powershell)**, then select the protocol from the **Connection Protocol** dropdown menu. You can select **HTTP** or **HTTPS**.

The **Connection Port** field is automatically populated depending upon the selected **Connection Type**. For SSH, the connection port is 22 and for PowerShell the connection port is 5985 for HTTP and 5986 for HTTPS.

5. Enter the delay in seconds in the **Delay** field.

Delay timer defines the time period when the check login script is run after the VM starts. It allows you to configure the delay time to allow guest customization script, IP, and all other services to come up before running the check login script.

6. In the **Retries** field, enter the number of log-on attempts the system must perform after each log on failure.

7. To save the blueprint, click **Save**.

Tasks Overview

Tasks are part of your deployment creation process and are run one after the other. The tasks are used to perform a variety of operations such as setting up your environment, installing a set of software on your service, and so on.

You have the following basic types of tasks.

- **Execute:** Used to run eScripts on a VM. For more information, see [Adding an Execute Task](#) on page 176.
- **Set variable:** Used to change variables in a task. For more information, see [Adding a Set Variable Task](#) on page 179.
- **HTTP:** Used to query REST calls from a URL. For more information, see [Adding an HTTP Task](#) on page 185.
- **Delay:** Used to set a time interval between two tasks or actions. For more information, see [Adding a Delay Task](#) on page 186.

Pre-create and Post-delete Tasks

Pre-create tasks are actions that are performed before a service is provisioned in a blueprint. For example, if you want to assign static IP addresses to your VMs by using IPAM service, you can create and run a pre-create task to receive the IP addresses before the service is provisioned. The pre-create task helps to restrict the broadcast traffic to receive the IP addresses for those VMs during the service provision.

Post-delete tasks are actions that are performed after you delete a service in a blueprint. For example, if you want to delete the assigned IP addresses from your VMs, you can add a post-delete task to delete the IP addresses after the service is deleted. The post-delete task helps to restrict the broadcast traffic to delete the IP addresses for those VMs during the service provision.

Adding an Execute Task

You can add the Execute task type to run scripts on the VM.

About this task

Use this procedure to add an Execute task.

Procedure

1. In the **Script Type** dropdown menu, select one of the following:
 - **Shell**
 - **EScript**
 - **Powershell**
 - **Python**
2. If you have selected the script type as **Shell** or **Powershell**, do the following:
 - a. In the **Endpoint (Optional)** dropdown menu, select an endpoint for the task or click **Add New Endpoint** to create an endpoint. For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386.
 - b. In the **Credential** dropdown menu, select an existing credential or click **Add New Credential** to add a credential. For more information on adding a credential, see [Adding Credentials](#) on page 173.
 - c. Enter the script or use the upload icon to upload the script in the **Script** field.
You can access the available list of macros by using @@{ in the **Script** field.
For sample **Powershell** scripts, see [Sample Powershell Script](#) on page 422.
 - d. If you want to test the script in Self-Service playground, click **Test Script**.
Self-Service playground allows you to test a script by running and reviewing the output and making required changes.
The **Test Script** page is displayed.
 - e. On the **Authorization** tab, enter the following fields:
 - **IP Address** : Enter the IP address of the test machine.
 - **Port** : Enter the port number of the test machine.
 - **Select tunnel to connect with (Optional)** : Select the tunnel to get access to the VMs within the VPC.
 - **Credential** : Select the credential from the list.
 - **User name** : Enter a user name.
 - **Password** : Enter a password.
 - f. Click **Login and Test**.
The **Test script** page is displayed.
You can also view your script in the **Source Script** field.
 - g. (Optional) You can edit your script in the **Source Script** field.
 - h. If you are using macros in your script, provide the variable values in the macro inspector panel and click **Assign and Test**.
 - i. Click **Test**.
The test result is displayed in the **Output** field.
 - j. To go back to the previous screen, click **Done**.

3. If you have selected the script type as **EScript**, do the following:

- a. In the **Select tunnel to connect with** dropdown menu, select a tunnel to access VMs in the VPC. This step is optional.
- b. In the **Script** field, select the Python version from the version drop-down menu.
By default, Python 3 is selected as the version for you to enter the script.

Note: Based on the decision that the Python Software Foundation (PSF) took to discontinue support for Python 2, Nutanix has decided to end support for Python 2 on June 30, 2024. Nutanix recommends that you format all your new eScripts in Python 3 and update any existing eScripts from Python 2 to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

- c. Enter the script or use the upload icon to upload the script in the **Script** field.
You can access the available list of macros by using @@{ in the **Script** field.
For sample `Escripts`, see [Supported eScript Modules and Functions](#) on page 416.

Note: You can use macro expansions for variables used for eScripts.

- d. To test the script, save the blueprint, and click **Test Script**.
The Test EScript page is displayed.
You can also view your script in the **Source Script** field.
- e. If you are using macros in your script, provide the variable values in the macro inspector panel and click **Assign and Test**.
- f. To test the script, click **Test**.
The test result is displayed in the **Output** field.
- g. To go back to the previous screen, click **Done**.

4. If you have selected the script type as **Python**, do the following:

Use Python script type when you want your scripts to run in a predefined Python environment. This option enables you to run Python scripts in your own customized endpoint wherein you can add custom modules, any version of Python, and their corresponding binaries.

- a. In the **Endpoint** dropdown menu, select the endpoint that you created to run your Python scripts or click **Add New Endpoint** to create an endpoint.

For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386. For information on how to configure your endpoint to run Self-Service Python scripts, see [Configuring a Remote Machine to Run Self-Service Python Scripts](#) on page 389.

Note:

- Providing an endpoint is mandatory when you select the **Python** script type.
- The Python script type does not support tunnels and endpoints that are created within a VPC.
- The Python script type is supported only with Linux endpoints.

- b. In the **Credential** dropdown menu, select an existing credential or click **Add New Credential** to add a credential.

For more information on adding a credential, see [Adding Credentials](#) on page 173.

- c. Enter the script or use the upload icon to upload the script in the **Script** field.

You can access the available list of macros by using `@@{` in the **Script** field.

5. To publish this task to the task library, click **Publish to Library**.

The task is published to the Library and you can browse and use the task while creating a blueprint.

Adding a Set Variable Task

Use the Set Variable task to define and assign values to variables for customizing or adding flexibility to your blueprints and Deployments.

Before you begin

Ensure that you have the basic configuration ready for the entities where you want to add the Set Variable task.

- For more information on the Single-VM blueprint configuration, see [Configuring Tasks or Packages in a Blueprint](#) on page 124.
- For more information on the Multi-VM blueprint configuration, see [Configure Multi-VM, Package, and Service](#) on page 131.

About this task

Perform the following steps to add a Set Variable task.

Procedure

1. In the **Script Type** dropdown menu, select one of the following:
 - **Shell**
 - **Powershell**
 - **EScript**
 - **Python**
2. If you selected the script type as **Shell** or **Powershell**, do the following:
 - a. In the **Endpoint** dropdown menu, select an endpoint for the task or click **Add New Endpoint** to create an endpoint. For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386.
 - b. In the **Credential** dropdown menu, select an existing credential or click **Add New Credential** to add a credential. For more information on adding a credential, see [Adding Credentials](#) on page 173.
3. If you selected the script type as **EScript**, then select a tunnel to access VMs in the VPC in the **Select tunnel to connect with** dropdown menu. This step is optional.
4. If you have selected the script type as **Python**, do the following:

Use Python script type when you want your scripts to run in a predefined Python environment. This option enables you to run Python scripts in your own customized endpoint wherein you can add custom modules, any version of Python, and their corresponding binaries.

 - a. In the **Endpoint** dropdown menu, select the endpoint that you created to run your Python scripts or click **Add New Endpoint** to create an endpoint.

For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386. For information on configuring your endpoint to run Self-Service Python scripts, see [Configuring a Remote Machine to Run Self-Service Python Scripts](#) on page 389.

Note:

 - Providing an endpoint is mandatory when you select the **Python** script type.
 - The Python script type does not support tunnels and endpoints that are created within a VPC.
 - The Python script type is supported only with Linux endpoints.
 - b. In the **Credential** dropdown menu, select an existing credential or click **Add New Credential** to add a credential.

For more information on adding a credential, see [Adding Credentials](#) on page 173.

5. To assign the value of the variable, do the following in the **Script** field.

The screenshot displays the 'Configuration' tab of the Nutanix Self-Service Blueprint Configuration interface. The 'Task Name' is 'Task 1'. The 'Type' is 'Set Variable'. The 'Script Type' is 'EScript'. The 'Select tunnel to connect with (optional)' dropdown is set to 'Select a tunnel'. The 'Script' field contains the code: `1 print("<var-name> = <var-value>")`. The 'Version' dropdown is set to 'Python 3'. The 'Output' field is set to 'variable1'. The 'Publish To Library' button is visible at the bottom.

Figure 44: Set Output Variable Definition

- » If you selected **EScript** as the script type, select the Python version from the version drop-down menu, and use the print statement. For example,

```
print("<var-name> = <var-value>")
```

Note: Based on the decision that the Python Software Foundation (PSF) took to discontinue support for Python 2, Nutanix has decided to end support for Python 2 on June 30, 2024. Nutanix recommends that you format all your new eScripts in Python 3 and update any existing eScripts

from Python 2 to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

- » If you selected **Shell** or **Powershell** as the script type, use the echo statement. For example,

```
echo "<var-name> = <var-value>"
```

Note:

- The only way to set the value to the variable or expose the variable is through the print or echo statement.
- To mask the value of sensitive variables and the value in audit logs, use the following format to print output.

```
print/echo "<var-name>,secret=<var-value>"
```

For examples, see [Sample Scripts for Installing and Uninstalling Services](#) on page 413.

You can access the available list of macros by using @@{ in the **Script** field.

For sample **Escripts**, see [Supported eScript Modules and Functions](#) on page 416. For sample **Powershell** scripts, see [Sample Powershell Script](#) on page 422.

6. In the **Output** field, enter the name of the variable that you have defined through the Set Variable task. If you are setting multiple variables, enter the name for each of the variable in the **Output** field.
7. [Optional] To publish this task to the task library so you can browse and use the task while creating a blueprint in the future, click **Publish to Library**.

8. [For Multi-VM blueprints only] Do the following to define the variable for the service on the **Service** tab to close the mapping.

The screenshot displays the 'Service' tab in a configuration interface. On the left sidebar, 'VM1' is selected, and under it, 'Service1' is highlighted. The main area has three tabs: 'VM', 'Package', and 'Service'. The 'Service' tab is active, showing a 'Description' field with a 'View' link. Below this is a 'Deployment Config' section with a note: 'Replicas will be named as 'Service_name[0]', 'Service_name[1]' and so on.' Underneath is a 'Number of replicas' section with 'Default', 'Min', and 'Max' input fields, all containing the value '1'. The 'Variables' section at the bottom shows a variable named 'variable1' of type 'String' with a value of 'value1'.

Figure 45: Service Tab - Service Variable Definition

- In the **Variables** section, click the **+** icon.
- In the **Name** field, enter the name of the variable you defined on the **Package** tab.
- From the **Data Type** dropdown menu, select one of the base type variables or import a custom library variable type.
- If you selected a base type variable, configure all the variable parameters. For more information on configuring variable parameters, see [Creating Variable Types](#) on page 259. If you have imported a custom variable type, all the variable parameters are auto filled.
- Select the **Secret** checkbox to hide the value of the variable. This step is optional.
- Click **Save** to save the blueprint configuration changes.

9. To consume the variable you defined in a subsequent task inside the same service, do the following:

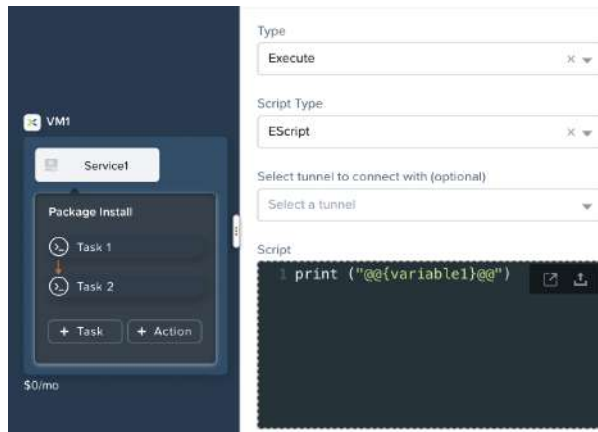


Figure 46: Service Variable Reference

- On the Blueprint Canvas, click **+ Task** for the service.
- In the Inspector panel, enter a name for the task in the **Name** field.
- In the **Type** dropdown menu, select **Execute**.
- In the **Script Type** dropdown menu, select the script type.
- In the **Script** field, print out the value of the variable.

For example, `print ("@{<var-name>}@@")`.

This configuration ensures that after the first task runs and sets the variable value in the service, the subsequent task prints out the variable value.

- Click **Save** to save the blueprint configuration changes.

10. [For Multi-VM blueprints only] To consume the variable you defined in another service (for example, Service2) in the same blueprint, do the following:

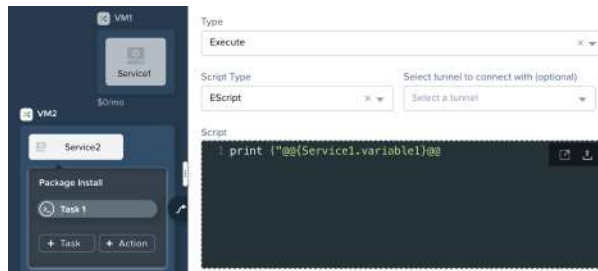


Figure 47: Service Variable Reference in Another Service

- a. Ensure that you select **Execute** as the task type on the **Package** tab of the other service (Service2).
- b. In the **Script Type** dropdown menu, select the script type.
- c. In the **Script** field, print out the value of the variable.

Because this variable exists in the first service (for example, Service1), ensure that you use the `<service-name>.<var-name>` format. For example, `print ("@@{Service1.<var-name>}@@")`.

- d. Click **Save** to save the blueprint configuration changes.

Adding an HTTP Task

You can add an HTTP task type to query REST calls from a URL. An HTTP task supports GET, PUT, POST, and DELETE methods.

About this task

Note: You can use macro expansions for variables used in an HTTP task.

Procedure

1. In the **Request URL** field, enter the URL of the server that you want to run the methods on.
2. In the **Select tunnel to connect with** dropdown menu, select a tunnel to access VMs in the Virtual Private Cloud (VPC). This step is optional.
3. In the **Request Method** dropdown menu, select one of the following request methods.
 - **GET:** Use this method to retrieve data from a specified resource.
 - **PUT:** Use this method to send data to a server to update a resource. In the **Request Body** field, enter the PUT request. You can also upload the put request by clicking the upload icon.
 - **POST:** Use this method to send data to a server to create a resource. In the **Request Body** field, enter the POST request. You can also upload the post request by clicking the upload icon.
 - **DELETE:** Use this method to send data to a server to delete a resource. In the **Request Body** field, enter the DELETE request. You can also upload the delete request by clicking the upload icon.
4. In the **Content Type** dropdown menu, select the type of the output format.
The available options are **XML**, **JSON**, and **HTML**.

5. In the **Header** area, enter the HTTP header key and value in the **Key** and **Value** fields respectively.
6. If you want to publish the HTTP header key and value pair as secret, click the **Secrets** fields.
7. In the **Connection Time Out** field, enter the timeout interval in seconds.
8. Optionally, in the **Authentication** field, select **Basic** and do the following:
 - a. In the **Username** field, enter the user name.
 - b. In the **Password** field, enter the password.
9. If you want to verify SSL certificate for the task, click the **SSL Certificate Verification** field.
10. If you want to use a proxy server as configured in the Prism Central, click the **Use PC Proxy configuration**.

Note: Ensure that the Prism Central has the appropriate HTTP proxy configuration.

11. In the **Retry Count** field, enter the number of attempts the system performs to create a task after each failure.
By default, the retry count is zero. It implies that the task creation procedure stops after the first attempt.
12. In the **Retry Interval** field, enter the time interval in seconds for each retry if the task fails.
13. In the **Expected Response Options** area, enter the details for the following fields:
 - **Response Status:** Select either **Success** or **Failure** as the response status for the task.
 - **Response Code:** Enter the response code for the selected response status.

Note: If the response code is not defined, then by default all the 2xx response codes are marked as success and any other response codes are marked as failure.

- **Set Variables from response:** Enter the variables from the specified response path. The example of json format is \$.x.y and xml format is //x/y. For more information on json path syntax, see <http://jsonpath.com>.

Note: To retrieve the output format in HTML format, add a * in the syntax.

14. If you want to test the script in Self-Service playground, click **Test script**.
Self-Service playground allows you to test a script by running and reviewing the output and making required changes.
The **Test Script** page is displayed. You can also edit the fields described from step 1–11.
15. Click **Test**.
The test result is displayed in the **Output field**.
16. To publish this task to the task library, click **Publish to Library**.
The task is published to the Library and you can browse and use the task while creating a blueprint.

Adding a Delay Task

You can add a Delay task type to set a time interval between two tasks or actions.

Procedure

In the **Sleep Interval** field, enter the sleep time interval in seconds for the task.

The delay task type is created. You can use the task type to set a time interval between two tasks or actions.

Adding a Pre-create, Post-create, or Post-delete Task

Perform the following steps to configure a pre-create task, post-create task, or post-delete task in a multi-VM blueprint.

About this task

You have the following tasks for a multi-VM blueprint:

- **Pre-create tasks**

Pre-create tasks are actions that are performed before the blueprint or application is launched. These tasks help you to prepare the environment or set up prerequisites required for the successful deployment of the application or blueprint. Pre-create tasks enable administrators or developers to automate and streamline the preparation steps thereby reducing manual efforts and potential errors during application deployments.

- **Post-create tasks**

The post-create stage is a stage between the creation of VM and package installation. Post-create tasks allow addition of automated tasks and actions that run on substrate without any manual intervention before the VM is powered on. To configure these automated tasks or actions, keep the VM in the powered off state after creation. You can use the Post-create configuration to carry out tasks such as:

- To use the Mac address of the VM for further orchestration in the pre-boot stage.
- To place the provisioned VMs in a security group before they are powered on.

The post-create tasks or actions are applicable to all providers.

- **Post-delete tasks**

Post-delete tasks are actions that are performed after a blueprint or application is deleted or terminated. These tasks are configured to clean up any remaining resources, configurations, or dependencies that were created during the deployment or execution of the blueprint or application. Post-delete tasks promote efficient resource utilization and help maintain a clean infrastructure state.

Procedure

1. In the Overview Panel, under the service in which you want to add the task, expand **VM**, and then click **Pre-create**, **Post-create**, or **Post-delete**.

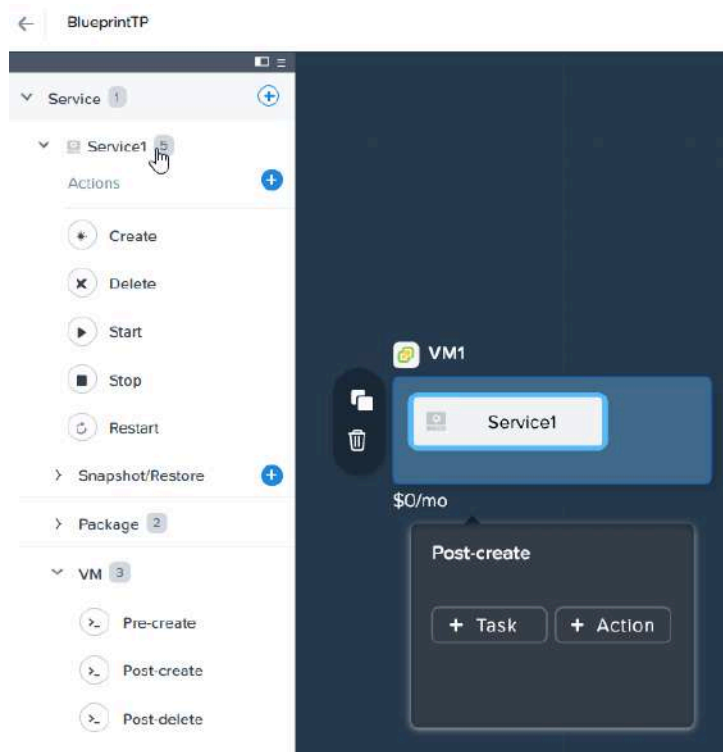


Figure 48: Pre-create, Post-create, or Post-delete Task

For **Post-create** in Nutanix and VMware, ensure that the VM is configured to stay in the powered off state after it is created. For more information, see [Configuring Nutanix and Existing Machine VM, Package, and Service](#) on page 132 or [Configuring VMware VM, Package, and Service](#) on page 138.

2. On the Blueprint Canvas, click **+ Task** for the Pre-create, Post-create, or Post-delete.
For **Post-create**, you can also click **+ Action** to add a Service or VM Action such as **VM Power On**, **VM Power Off**, **VM Restart**, or **VM Check Login**.

Note: The check log-in task uses the credential that you selected in the VM Configuration. If you have not provided the credential in VM configuration check log-in, the check log-in task uses the default credential.

3. In the Inspector Panel, do the following:

- a. Enter the task name in the **Task Name** field.
- b. Select the type of tasks from the **Type** dropdown menu.
 - **Execute**: Use this task type to run eScripts on the VM. To create the **Execute** task type, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: Use this task to change variables in a blueprint. To create the **Set Variable** task type, see [Adding a Set Variable Task](#) on page 179.
 - **HTTP**: Use this task type to query REST calls from a URL. An HTTP task supports GET, PUT, POST, and DELETE methods. To create the **HTTP** type, see [Adding an HTTP Task](#) on page 185.
 - **Delay**: Use this task type to set a time interval between two tasks or actions. To create the **Delay** task type, see [Adding a Delay Task](#) on page 186.
- c. To use tasks from the library, click **Browse Library**, select the task in the **Browse Task from Library** page, and click **Select**. This step is optional.
- d. Click **Publish to Library** to publish the task you configured to your task library. This step is optional.

4. Click **Save**.

Pre-create Task Workflow

Self-Service allows you to use the pre-create task to run actions before your blueprint is launched. The following workflow shows how you can dynamically assign a network or a cluster using variables in a pre-create task.

1. Add a Pre-create Task

In the blueprint editor, navigate to the service that requires the dynamic network or cluster assignment, and add a pre-create task. For more information on adding a pre-create task, see [Adding a Pre-create, Post-create, or Post-delete Task](#) on page 187.

2. Define Variables

In Self-Service, variables are used to store and pass values dynamically during the blueprint launch. Inside the pre-create task, create the set variable task or HTTP task that emits an output and map it to the variable.

To assign network or cluster, ensure that you define the variable in the correct format. For example, if you want to use a variable in your network specifications, you must provide the JSON value in the following format:

```
{"name": "<subnet_name>", "uuid": "<subnet_uuid>"}
```

3. Use Variables as Macros

Use the variables that you defined in the set variable task or HTTP task as macros within your VM configuration fields, such as

For Network:

- Subnet: @@{subnet}@@
- Gateway: @@{gateway}@@
- DNS: @@{dns}@@

For a Cluster:

- Cluster Name: @@{cluster_name}@@
- Cluster Address: @@{cluster_address}@@
- Cluster User: @@{cluster_user}@@
- Cluster Password: @@{cluster_password}@@

4. Configure the Blueprint and Launch

Once the pre-create task is configured with dynamic assignment logic, continue configuring the rest of the blueprint with other tasks and settings as required. Save your configurations and launch the blueprint.

Network Assignment - Example

With Self-Service integration with Infoblox IPAM, you can configure your blueprint to make REST API calls to Infoblox to find the next available IP and reserve it. The IP address supplied by Infoblox is then used to create the VM.

Suppose you create a blueprint with a single service to represent the VM, one input variable for Infoblox Grid Master IP, and the other input variable to select the network ID. To query the network ID from Infoblox, you add a pre-create action to the service with the following tasks:

- **A task to reserve IP Address**

To reserve the next available IP address, you use the fixed address Infoblox API. For example:

```
https://<infobloxGrid_IP>/wapi/v2.7/fixedaddress
```

You then update the hostname using the Self-Service application name macro in the task Request Body field. For example,

```
{
  "ipv4addr": "func:nextavailableip:@@{network}@@",
  "name": "@@{calm_application name}@@.nutanix.com",
  "match_client": "RESERVED"
}
```

- **A task to get the IP address**

Using the IP reference obtained in the previous task, you can get the `ipv4addr` (`static_ip`) and later reference the same in the **Static IP** field in the VM configuration.

Set variables from response ?

Variable	Path
static_ip	\$.ipv4addr

Figure 49: IP Reference -Variable and Path

A sample Request URL that you can use this case is:

```
https://<infobloxGrid_IP>/wapi/v2.7/@@{ipref}@@
```

- **A task to get the subnet reference**

You need to have the subnet reference in the VM configuration. To obtain the subnet reference, save the subnet reference expected by Self-Service as an extensible attribute in the subnet and query the same.

Set variables from response ?

Variable	Path
subnet_ref	\$.extattrs.CalmRef.value

Figure 50: Subnet Reference - Variable and Path

For the subnet reference to work properly, the value fetched for the subnet reference must be in the following format:

```
{"name": "<subnet_name>", "uuid": "<subnet_uuid>"}
```

In case the value is not in the correct format, you can add another set variable task to change the value to the correct format before using the variable as macro in your VM configuration fields.

Use the subnet reference as a macro (`subnet_ref`) in the **NIC** dropdown in the VM configuration.

NETWORK ADAPTERS (NICs) (1)

NIC 1

Subnet: subnet_ref

Private IP: ☐ Dynamic ☒ Static

Static IP: @@{static_ip}@@

Figure 51: Network Adapter - Using Subnet Reference

After the pre-create task is configured, launch the blueprint. During the launch, provide the app name, verify the already-populated Grid Master IP address, and select the network from which an IP address has to be obtained.

On the **Audit** tab of the application, you can verify each of the API call in the pre-create task. Once the VM is running, you can log in to the Infoblox Grid Master, navigate to the network, and verify the IP address and the associated hostname.

Actions Overview

Actions are flows to accomplish a particular task on your app. You can use actions to automate any process such as backup, upgrade, new user creation, or clean-up and enforce an order of operations across services.

You can categorize actions into the following types.

Table 36: Action Types

Type	Description
Profile Actions	<p>Application Profile Actions are a set of operations that you can run on your app. For example, when you launch a blueprint, the Create action is run. When you do not need the app for a period of time, you can run the Stop action to gracefully stop your app. When you are ready to resume your work, you can run Start action to bring the app back to the running state.</p> <p>You have the following types of profile actions.</p> <ul style="list-style-type: none">• System-defined Profile Actions These actions are automatically created by Self-Service in every blueprint and the underlying app. Because these actions are system-defined, a blueprint developer cannot directly edit the tasks or the order of tasks within the action.• Custom Profile Actions These actions are created by the blueprint developer and are added whenever the developer needs to expose a set of operations to the app user. Common custom profile actions are Upgrade, Scale In, and Scale Out. In these actions, individual tasks can be manually added in the desired order by the developer.

Type	Description
Service Actions	<p>Service Actions are a set of operations that are run on an individual service. These actions cannot be run directly by the app user but can be run indirectly using either a profile actions or a package install or uninstall operation.</p> <p>Services span app profiles. For example, if you create a service action in the AHV profile, the same service action is available in the AWS profile as well.</p> <p>You have the following types of service actions.</p> <ul style="list-style-type: none"> • System-defined Service Actions <p>These actions are automatically created by Self-Service in every blueprint and the underlying app. These actions cannot be run individually and are run only when the corresponding profile action is run. For example, any operations within the Stop service action are run when an app user runs the Stop profile action.</p> • Custom Service Actions <p>These actions are created by the blueprint developer for any repeatable operations within the blueprint. For example, if the App service should fetch new code from git during both the Create and Upgrade profile actions, the blueprint developer can create a single custom service action. The developer can then reference the action in both the Create and Upgrade actions rather than maintaining two separate tasks that perform the same set of operations.</p>

Custom Actions

The following are the most common custom actions that developers add to their blueprints:

Table 37: Custom Actions

Custom Action	Description
Scale In	<p>The scale-in functionality enables you to decrease the number of replicas of a service deployment. The number of instances to be removed from a service for each scale-in action is defined in the blueprint while configuring the task in the profile level action.</p> <p>The scale count number must be less than or equals to the minimum number of replicas defined for the service. The VM that is created last is deleted first.</p> <p>For information on configuring a scale-in action, see Adding and Configuring Scale Out and Scale In on page 198.</p>
Scale Out	<p>The scale out functionality enables you to increase the number of replicas of a service deployment. The number of instances to be added to a service for each scale-out action is defined in the blueprint while configuring the task in the profile level action.</p> <p>The scale count number must be less than or equals to the maximum number of replicas defined for the service.</p> <p>For information on configuring a scale-out action, see Adding and Configuring Scale Out and Scale In on page 198.</p>

For information on how to create an action, see [Adding an Action to a Multi-VM Blueprint](#) on page 196 and [Adding an Action to a Single-VM Blueprint](#) on page 194.

Adding an Action to a Single-VM Blueprint

An action is a set of operations that you can run on your app that are created as a result of running a blueprint.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.
- Ensure that you added credentials to enable packages and actions. For more information, see [Adding Credentials](#) on page 173.

Procedure

1. To add a custom task or action, click the **+ Add action** next to **Actions**.

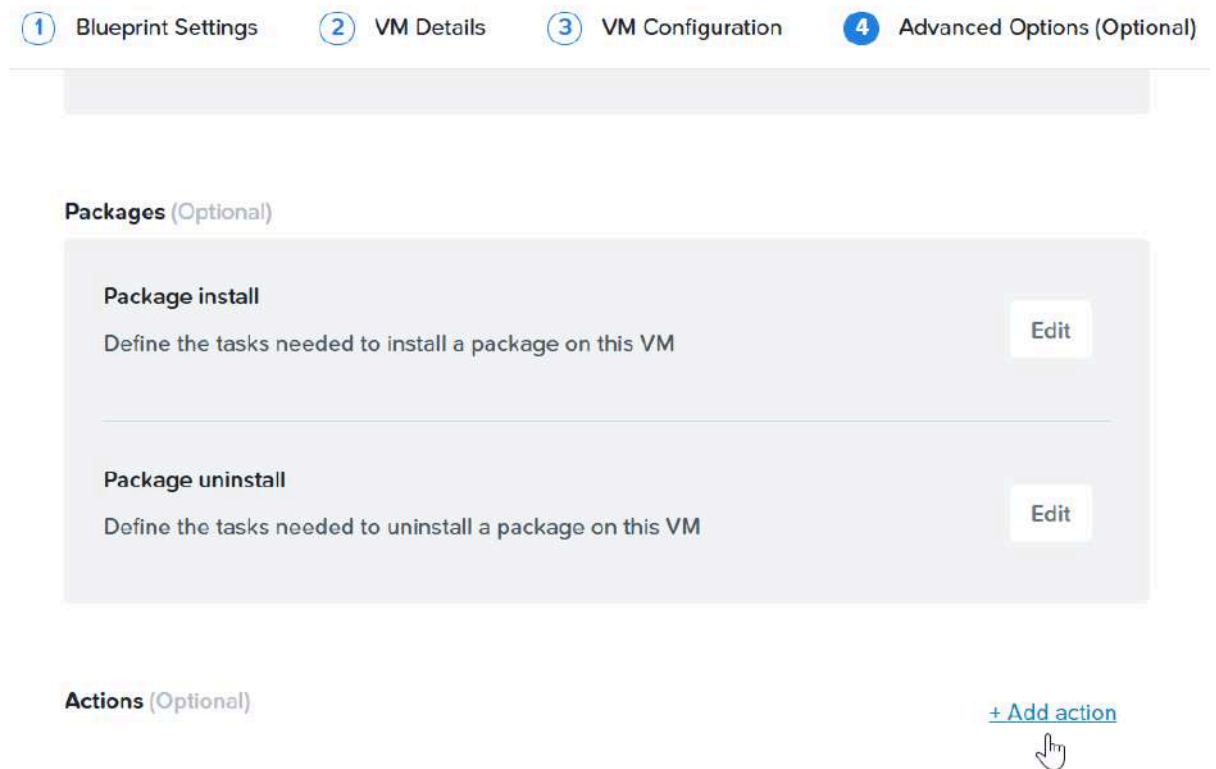


Figure 52: Blueprint Action

2. To change the action name, click the edit icon on the **Tasks** tab.

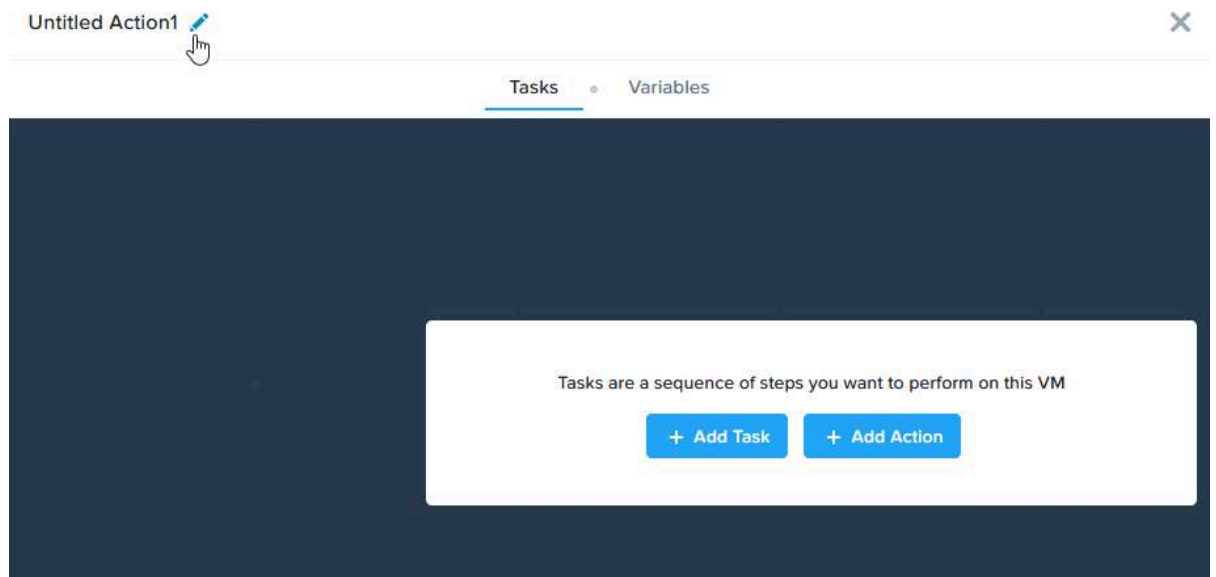


Figure 53: Add Action

3. To add a custom task, click **+ Add Task** and do the following.
 - a. On the Blueprint Canvas, select the task.
 - b. Enter the name of the task in the **Task Name** field.
 - c. Select the type of the task from the **Type** dropdown menu.
 - **Execute**: Use this task type to run eScripts on the VM. To create the **Execute** task type, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: Use this task to change variables in a blueprint. To create the **Set Variable** task type, see [Adding a Set Variable Task](#) on page 179.
 - **HTTP**: Use this task type to query REST calls from a URL. An HTTP task supports GET, PUT, POST, and DELETE methods. To create the **HTTP** type, see [Adding an HTTP Task](#) on page 185.
 - **Delay**: Use this task type to set a time interval between two tasks or actions. To create the **Delay** task type, see [Adding a Delay Task](#) on page 186.
 - d. (Optional) To use tasks from the library, click **Browse Library**, select the task in the **Browse Task from Library** page, and click **Select**.
 - e. (Optional) Click **Publish to Library** to publish the task you configured to your task library.
 - f. Click **Done**.
4. To add an custom action, click **+ Action** and do the following: **VM Power On**, **VM Power Off**, **VM Restart**, or **VM Check Login**.
 - a. On the Blueprint Canvas, select the task.
 - b. Enter the name of the task in the **Task Name** field.
 - c. Select **VM Power On**, **VM Power Off**, **VM Restart**, or **VM Check Login** in the **Service/VM Actions** dropdown menu.

Note: The check log-in task uses the credential that you selected in the VM Configuration. If you have not provided the credential in VM configuration check log-in, the check log-in task uses the default credential.

- d. Click **Done**.

Adding an Action to a Multi-VM Blueprint

An action is a set of operations that you can run on your app that are created as a result of running a blueprint.

Before you begin

- Ensure that you have at least one service available. For more information, see [Adding a Service](#) on page 130.
- Ensure that you have completed the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.

Procedure

1. To add an action, click the **+** icon next to **Actions** in the Overview Panel.

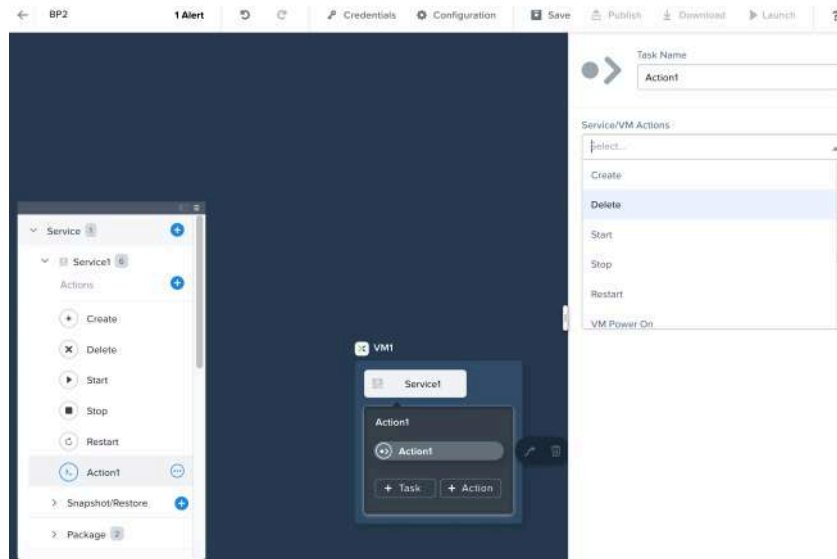


Figure 54: Blueprint Action

2. On the Blueprint Canvas, select **+ Action** for the service, and do the following in the Inspector Panel.
 - a. Enter the task name in the **Task Name** field.
 - b. Select the action from the **Service Actions** dropdown menu.
 - c. Click **Save**.

3. On the Blueprint Canvas, select **+ Task** and do the following in the Inspector Panel.

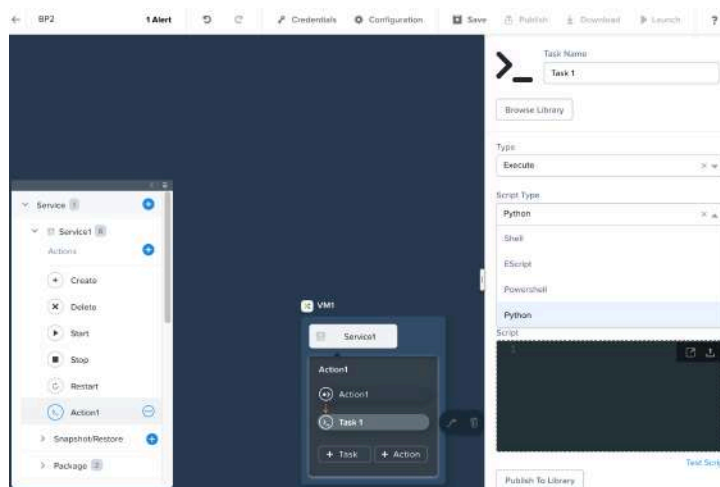


Figure 55: Task

- a. Enter the name of the task in the **Task Name** field.
- b. Select the type of the task from the **Type** dropdown menu.
 - **Execute**: Use this task type to run eScripts on the VM. To create the **Execute** task type, see [Adding an Execute Task](#) on page 176.
 - **Set Variable**: Use this task to change variables in a blueprint. To create the **Set Variable** task type, see [Adding a Set Variable Task](#) on page 179.
 - **HTTP**: Use this task type to query REST calls from a URL. An HTTP task supports GET, PUT, POST, and DELETE methods. To create the **HTTP** type, see [Adding an HTTP Task](#) on page 185.
 - **Delay**: Use this task type to set a time interval between two tasks or actions. To create the **Delay** task type, see [Adding a Delay Task](#) on page 186.
- c. Click **Save** on the Blueprint Editor page.

Adding and Configuring Scale Out and Scale In

Perform the following procedure to add and configure the Scale Out and Scale In task.

Procedure

1. Add a service. For more information, see [Adding a Service](#) on page 130.
2. Configure VM, Package and Service. For more information, see [Configuring Nutanix and Existing Machine VM, Package, and Service](#) on page 132.
3. Add an Application profile. For more information, see [Adding and Configuring an App Profile](#) on page 166.

4. In the Overview Panel, under **Application Profile**, click the **+** icon next to Actions.

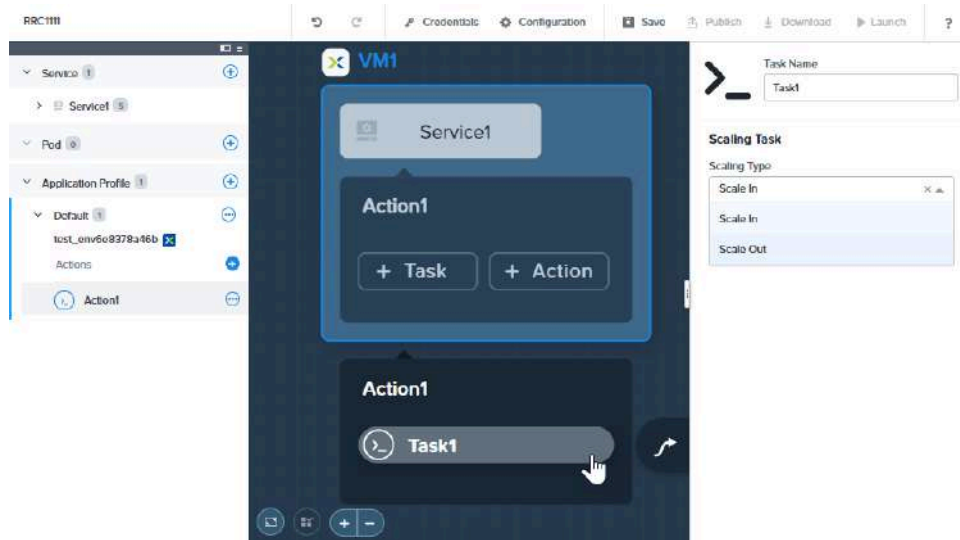


Figure 56: Scale In and Scale Out

5. On the Blueprint Canvas, below the service inspector, click **+ Task**.
6. In the Inspector Panel, enter the name of the task in the **Task Name** field.
7. Select **Scale In** or **Scale Out** from the **Scaling Type** dropdown menu.
8. Enter the number in the **Scaling Count** field.
The Scaling out and Scaling in number should be less than the minimum and maximum number of replicas defined for the service.
9. Click **Save**.

What to do next

You can run the scale-in or scale-out tasks on the My Apps page. To do that, you first have to launch the blueprint and then go to My Apps in Admin Center to view the created app. You can run the scale in or scale out on the **Manage** tab of the app.

Blueprint Configuration for Snapshots and Restore

The snapshot and restore feature allows you to create a snapshot of a virtual machine at a particular point in time and restore from the snapshot to recreate the app VM from that time. You can configure snapshot and restore for both single-VM and multi-VM apps on a Nutanix or VMware platform. All you need to do is to add the snapshot/restore configuration to the blueprint. Adding the configuration generates separate profile actions for snapshot and restore to which you can add further tasks and actions.

For AWS and Azure platforms, the snapshot and restore feature is available by default only to the single-VM apps.

Configuring Single-VM Blueprints on Nutanix for Snapshots

The snapshot/restore action for single-VM apps on a Nutanix account is no longer available by default. To enable snapshot, you must add a snapshot/restore configuration to the single-VM blueprint. You can configure to create snapshots locally or on a remote cluster. Snapshot and restore is a paired action in a blueprint and are always managed together.

About this task

The snapshot/restore configuration generates separate app profile actions for snapshot and restore. These actions also allow you to add more tasks and actions as part of the snapshot and restore configuration. For example, shutting down the app and the VM before creating the snapshot or restarting the VM before a restore. You can access these actions from the **Manage** tab of the Applications page.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.
- Ensure that you added credentials to enable packages and actions. For more information, see [Adding Credentials](#) on page 173.
- Ensure that you created the required snapshot policy. You associate snapshot policies when you launch the blueprint configured for snapshot and restore. For more information on creating snapshot policy, see [Creating a Snapshot Policy](#).

Procedure

1. On the **Advanced Options** tab, next to Snapshot/Restore, click the **+ Add Snapshot/ Restore Config** option.

1 Blueprint Settings 2 VM Details 3 VM Configuration 4 Advanced Options (O)

Add Snapshot And Restore

Snapshot/Restore action suffix

Snapshot

Snapshot name

snapshot-@[calm_array_index]@-@[calm_time]@

VM Name and Timestamp are used as the default snapshot name

Snapshot location

☒ Local ☐ Remote Cluster

☒ Delete older VM after restore

This will generate separate application profile actions for Snapshot and Restore

Cancel Save

Snapshot action for AHV VMs is no longer available by default. Add a config here for customizable snapshot/restore actions.

+ Add Snapshot/Restore Config

Figure 57: Snapshot Config

2. In the Add Snapshot and Restore window, do the following:
 - a. Enter the suffix that you want to associate to the snapshot/restore profile action.
For example, if you enter Profile1 as the suffix, the profile actions will show up as **Snapshot_Profile1** and **Restore_Profile1**.
 - b. If required, enter a name for the snapshot in the **Snapshot Name** field.
The VM name and timestamp are used by default to generate a snapshot name.
 - c. Under the Snapshot Location section, select **Local** or **Remote Cluster** to specify whether this configuration should manage your snapshots locally or on a remote cluster.
 - d. Select the **Delete older VM after restore** checkbox to delete the older VM after the service is restored from the snapshot.
 - e. Click **Save**.
Saving the configuration generates separate profile actions for snapshot and restore.
3. On the **Advanced Options** tab, click **Edit** next to the snapshot or restore action to edit the configuration or add tasks. For more information on adding a task, see [Configuring Tasks or Packages in a Blueprint](#) on page 124.

What to do next

- You can launch the blueprint after associating snapshot policies and rules. For more information, see [Launching a Blueprint](#) on page 224.
- You can access the create snapshots from the **Manage** tab of the app. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Configuring Multi-VM Blueprints on Nutanix for Snapshots

You can configure the snapshot/restore action in a blueprint on a Nutanix account to create snapshots of the application locally or on a remote cluster. Snapshot/restore is a paired action for a particular service in a blueprint and are always managed together.

About this task

The snapshot/restore definition of a service generates snapshot configuration and its corresponding restore configuration. You can use these configurations to modify your snapshot and restore setup.

The snapshot/restore configuration generates separate app profile actions for snapshot and restore. These actions allow you to add more tasks and actions as part of the snapshot and restore configuration. For example, shutting down the app and the VM before creating the snapshot or restarting the VM or services before a restore. You can access these actions from the **Manage** tab of the Applications page to create or restore snapshots.

Before you begin

- Ensure that you have completed the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have at least one service available. For more information, see [Adding a Service](#) on page 130.
- Ensure that you create the required snapshot policy. You associate snapshot policies when you launch the blueprint configured for snapshot and restore. For more information on creating snapshot policy, see [Creating a Snapshot Policy](#).

Procedure

1. In the Overview Panel, expand the service to which you want to add the snapshot and restore action.
2. Click the + icon next to **Snapshot/Restore**.

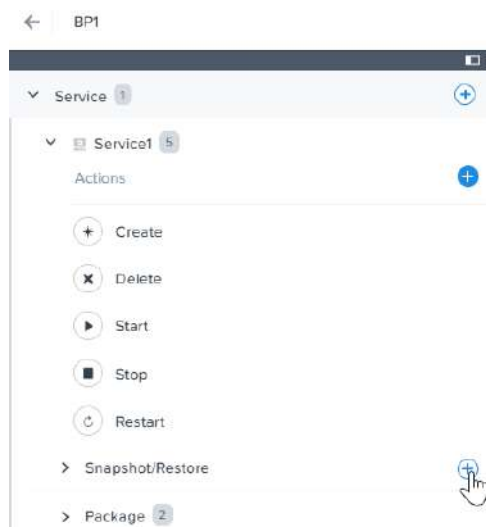


Figure 58: Multi-VM Snapshot

3. In the Add Snapshot and Restore window, do the following:
 - a. Enter the suffix that you want to associate to the snapshot/restore profile action.
 - b. Enter a name for the snapshot in the **Snapshot Name** field.

You can use Self-Service macros to provide a unique name to the snapshot whenever they are generated. For example, `snapshot-@@{calm_array_index}@@-@@{calm_time}@@`.
 - c. Under the Snapshot Location section, select **Local** or **Remote Cluster** to specify whether this configuration should manage your snapshots locally or on a remote cluster.
 - d. In case of multiple replicas of the service, do one of the following:
 - » Select **Take snapshot of the first replica only** to take snapshot of only the first replica.
 - » Select **Take snapshot of the entire replica set** to take snapshot of the entire replica set.
 - e. Select the **Delete older VM after restore** checkbox to delete the older VM after the service is restored from the snapshot.
 - f. Click **Save**.

Saving the configuration generates separate profile actions for snapshot and restore.
4. To view and edit the snapshot and restore configurations, expand the corresponding **Snapshot/Restore** under the service.

You can click the configuration to view the details in the Inspector Panel or make any changes to the configuration.
5. To view the profile actions for snapshot and restore or add more tasks and actions as part of snapshot and restore configuration, expand the **Application Profile** section.

What to do next

- You can add more tasks and actions to the snapshot and restore app profiles. For more information, see [Adding an Action to a Multi-VM Blueprint](#) on page 196.
- You can launch the blueprint after associating snapshot policies and rules. For more information, see [Launching a Blueprint](#) on page 224.
- You can create snapshots from the **Manage** tab of the app. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Configuring Single-VM Blueprints on VMware for Snapshots

The snapshot/restore action for single-VM apps on a VMware account is no longer available by default. To enable snapshot, you must add a snapshot/restore configuration to the single-VM blueprint. Snapshot and restore is a paired action in a blueprint and are always managed together.

About this task

The snapshot/restore configuration generates separate app profile actions for snapshot and restore. These actions also allow you to add more tasks and actions as part of the snapshot and restore configuration. For example, shutting down the app and the VM before creating the snapshot or restarting the VM before a restore. You can access these actions from the **Manage** tab of the Applications page.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.
- Ensure that you added credentials to enable packages and actions. For more information, see [Adding Credentials](#) on page 173.
- Ensure that you created the required snapshot policy. You associate snapshot policies when you launch the blueprint configured for snapshot and restore. For more information on creating snapshot policy, see [Creating a Snapshot Policy](#).

Procedure

1. On the **Advanced Options** tab, next to Snapshot/Restore, click the **+ Add Snapshot/ Restore Config** option.
2. In the Add Snapshot and Restore window, do the following:
 - a. Enter the suffix that you want to associate to the snapshot/restore profile action.
For example, if you enter Profile1 as the suffix, the profile actions will show up as **Snapshot_Profile1** and **Restore_Profile1**.
 - b. If required, enter a name for the snapshot in the **Snapshot Name** field.
The VM name and timestamp are used by default to generate a snapshot name.
 - c. Provide a description for the snapshot configuration in the **Description** field.
 - d. Click **Save**.
Saving the configuration generates separate profile actions for snapshot and restore.

3. On the **Advanced Options** tab, click **Edit** next to the snapshot or restore action to edit the configuration or add tasks. For more information on adding a task, see [Configuring Tasks or Packages in a Blueprint](#) on page 124.

Configuring Multi-VM Blueprints on VMware for Snapshots

You can configure the snapshot/restore action in a multi-VM blueprint on a VMware account to create snapshots of the application. Snapshot/restore is a paired action for a particular service in a blueprint and are always managed together.

About this task

The snapshot/restore definition of a service generates snapshot configuration and its corresponding restore configuration. You can use these configurations to modify your snapshot and restore setup.

The snapshot/restore configuration generates separate app profile actions for snapshot and restore. These actions allow you to add more tasks and actions as part of the snapshot and restore configuration. For example, shutting down the app and the VM before creating the snapshot or restarting the VM or services before a restore. You can access these actions from the **Manage** tab of the Applications page to create or restore snapshots.

Before you begin

- Ensure that you have completed the pre-configuration requirements. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
- Ensure that you have at least one service available. For more information, see [Adding a Service](#) on page 130.
- Ensure that you create the required snapshot policy. You associate snapshot policies when you launch the blueprint configured for snapshot and restore. For more information on creating snapshot policy, see [Creating a Snapshot Policy](#).

Procedure

1. In the Overview Panel, expand the service to which you want to add the snapshot and restore action.
2. Click the **+** icon next to **Snapshot/Restore**.

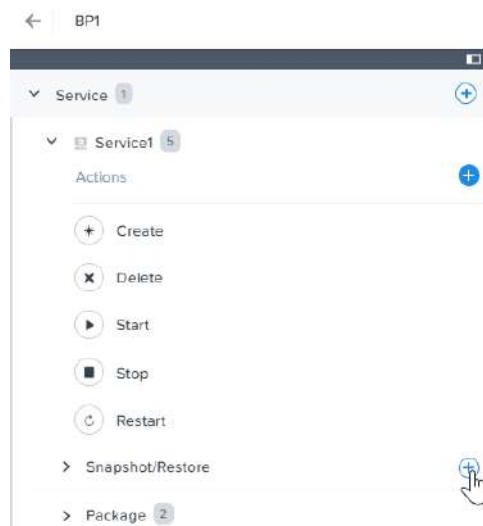


Figure 59: Multi-VM Snapshot

3. In the Add Snapshot and Restore window, do the following:
 - a. Enter the suffix that you want to associate to the snapshot/restore profile action.
For example, if you enter Profile1 as the suffix, the profile actions will show up as **Snapshot_Profile1** and **Restore_Profile1**.
 - b. Enter a name for the snapshot in the **Snapshot Name** field.
The VM name and timestamp are used by default to generate a snapshot name.
 - c. Provide a description for the snapshot configuration in the **Description** field.
 - d. In case of multiple replicas of the service, do one of the following:
 - » Select **Take snapshot of the first replica only** to take snapshot of only the first replica.
 - » Select **Take snapshot of the entire replica set** to take snapshot of the entire replica set.
 - e. Click **Save**.
Saving the configuration generates separate profile actions for snapshot and restore.
4. To view and edit the snapshot and restore configurations, expand the corresponding **Snapshot/Restore** under the service.
You can click the configuration to view the details in the Inspector Panel or make any changes to the configuration.
5. To view the profile actions for snapshot and restore or add more tasks and actions as part of snapshot and restore configuration, expand the **Application Profile** section.

What to do next

- You can add more tasks and actions to the snapshot and restore app profiles. For more information, see [Adding an Action to a Multi-VM Blueprint](#) on page 196.
- You can launch the blueprint after associating snapshot policies. For more information, see [Launching a Blueprint](#) on page 224.
- You can create snapshots from the **Manage** tab of the app. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Update Configuration for VM

The update configuration feature allows you to update virtual machines of running apps on Nutanix and VMware to a higher or lower configuration. Using this feature, you can modify VM specifications such as the vCPU, memory, disks, networking, or categories (tags) of a running app with minimal downtime. You no longer have to create new blueprints or approach your IT administrator to modify VM resources.

To update configurations of a running app VM, you need to perform the following actions:

- Add an update configuration to the app blueprint.
- Launch the update configuration

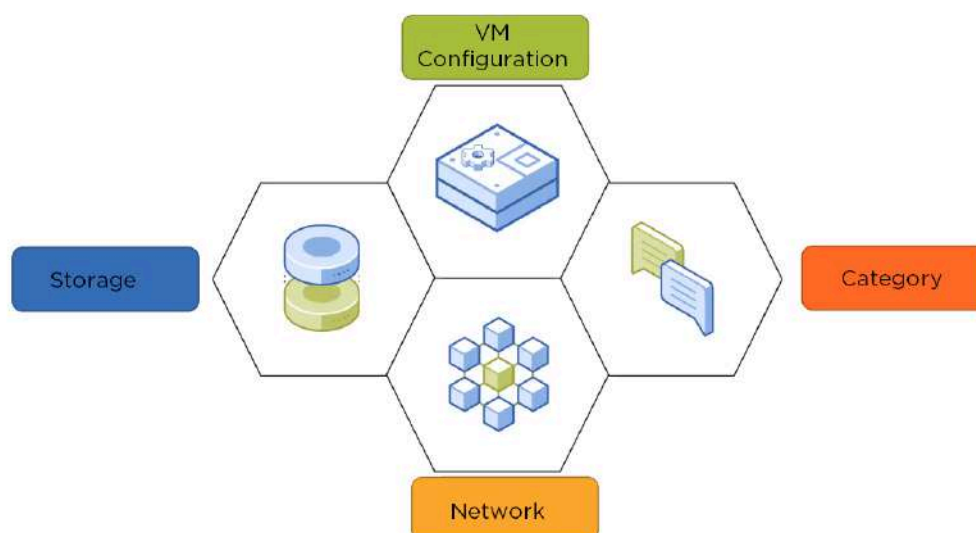


Figure 60: Update Configurations

Add Update Configuration in the Blueprint

As a blueprint developer, you can add update configurations for a service in the blueprint. These update configurations are at the parallel level of app profile actions and can be executed individually for a particular service. As part of the configuration, you can do the following:

- Specify the change factor (increase, decrease, or provide a definitive value) for VM configurations (vCPU, cores per vCPU, and memory) on Nutanix. You can provide a minimum or maximum limit for each component based on the change factor you select and allow blueprint consumers to edit components during updates. For example, consider a case where the original vCPU value in the blueprint is 4. You then add a change factor to the update configuration to increase the vCPU by 1 with a maximum limit of 5. When this update is launched, you can run the action only once to increase the vCPU to 5. Once the VM is upgraded to 5 vCPU, you cannot add any more vCPUs to the VM.
- Specify the change factor (increase, decrease, or provide a definitive value) for VM configurations (vCPU, cores per Socket, and memory) on VMware. You can select CPU Hot Add or Memory Hot Plug options to add vCPUs or memory resources while the virtual machine is turned on.
- Add disks with a minimum and maximum limit for each disk. You can allow blueprint users to edit the disk size during updates until the value reaches the maximum or minimum limit. You can also allow blueprint users to remove existing vdisks from the VM.
- Add categories (tags) to the running VM.
- Add NICs to the VM or allow blueprint consumers to remove NICs. You can only add those NICs that belong to the same cluster and remove only those NICs that are not used to provide the address. You can also allow consumers to choose the desired subnet during updates.

The update configuration generates the corresponding action where you can add tasks to define how you want to execute the update.

Launch Update Configuration

You can update VM specifications from the **Manage** tab of apps. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Adding Single-VM Update Configuration on Nutanix

As a blueprint developer, you can add an update configuration to a single-VM app blueprint on Nutanix.

About this task

The update configuration feature allows you to update the virtual machine of a running single-VM app to a higher or lower configuration. For more information, see [Update Configuration for VM](#) on page 205.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.

Procedure

1. On the **Advanced Options** tab, next to Update Configs, click the **+ Add Config** option.
The **+ Add Config** option is available only after you add VM credentials for packages and actions enablement. For more information, see [Adding Credentials](#) on page 173.
2. On the Update Configs page, click the edit icon next to the update config name to change the name of the configuration.

- View the current value of vCPUs, No. of Cores, and Memory (GiB) in the Current Value column.
- Click **Update** for the attribute that you want to update.

Update Configs

Variables

Name the update configuration

Config1

VM Configuration

Update the VM specifications as needed. Marking an attribute editable will allow the end-user to change its value within the limits specified via Min / Max. If the operation selected is 'Increase' or 'Decrease', the Update value should be relative. If 'Equal' is selected, the Update value should be absolute.

	Current Value	Operation	Update	Editable	Min Value	Max Value	
vCPUs	1	Equal	4	<input checked="" type="checkbox"/>	1	4	Cancel
Cores per vCPU	1	Equal	4	<input checked="" type="checkbox"/>	1	4	Cancel
Memory (GiB)	1	Equal	10	<input checked="" type="checkbox"/>	1	10	Cancel

Disks

Add / Edit vDisks to this VM

Disk 2: SCSI

New

DISK

×

SCSI

×

Operation

Allocate on Storage Container

	Value	Editable	Min Value	Max Value
Size (GiB)	1	<input checked="" type="checkbox"/>	1	4

Advanced settings

☐ Allow users to remove existing vDisks

Categories

Ensure that SSH port (default 22) is open in the security policies of the selected categories.

Key: Value

Advanced settings

☒ Allow users to remove existing categories
 ☒ Allow users to add new categories

Network Adapters

Add / Edit NICs to this VM

Advanced settings

☐ Allow users to remove existing NICs

Figure 61: Single-VM Update Config Options

- c. Select a value in the **Operation** dropdown menu for the attribute. You can select **Increase**, **Decrease**, or **Equal**.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then select **Increase** in the **Operation** dropdown menu.

Note: The update value is relative to the current value when you select **Increase** or **Decrease**. The update value is absolute when you select **Equal**.

- d. Specify an update value in the **Update** column based on the **Operation** value you selected.
For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then enter 1 in the **Update** field.
- e. Specify the limit value to which the configuration of an attribute can be updated in the **Min Value** or **Max Value** column.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU to a maximum limit of 6, then specify 2 in the **Max Value** column. The vCPU of the VM can be updated until its value reaches 6vCPU. After the VM reaches 6 vCPU, more vCPUs cannot be added to the VM.

You can also turn on **Editable** of an attribute to allow your users to change its value within the limits you specify in the **Min Value** or **Max Value** column during the launch of the update.

4. Under the **Disks** section, do the following:

- a. To add vdisk to the update configuration, click the + icon next to **Add/Edit vDisks to this VM**.
- b. Select the device type and the device bus.
- c. To define the disk size, specify the value for the vdisk size in the **Value** field.
You can turn on **Editable** and specify the **Min Value** and **Max Value** for the vdisk.
- d. To allow your users to remove existing vdisks from the VM during the launch of the update, click **Advanced Settings** and select the **Allow users to remove existing vDisks** checkbox.

5. Under the **Categories** section, do the following:

- a. To add to the update configuration, select the categories in the **Key: Value** dropdown menu.

Note: The categories you select must have the default SSH port (port 22) open in the security policies.

- b. To allow your users to remove existing categories during the launch of the update, click **Advanced Settings** and select the **Allow users to remove existing categories** checkbox.
- c. To allow your users to add new categories during the launch of the update, select the **Allow users to add new categories** checkbox.

6. Under the **Network Adapters** section, do the following:
 - a. To add more NICs to the update configuration, click the **+** icon next to **Add/Edit NICs to this VM** and select the NIC from the list.

The NICs of a VM can either use VLAN subnets or overlay subnets. For example, if an overlay subnet is selected for NIC 1 and you want to add NIC 2, the NIC 2 list displays only the overlay subnets.

If you selected a VLAN subnet in NIC 1, any subsequent VLAN subnets belong to the same cluster. Similarly, if you select an overlay subnet, all subsequent overlay subnets belong to the same VPC.

You can turn on **Editable** to allow your users to choose the desired subnet during the launch of the update.
 - b. To allow your users to remove existing NICs during the launch of the update, click **Advanced Settings** and select the **Allow users to remove existing NICs** checkbox.
7. To add variables to the update configuration, click the **Variables** tab and then the **+** icon next to **Variables**.
8. Click **Done** to save the update configuration.

Saving the update config generates the **Config** component. The Config component lets you open the Update Config window to edit the update configuration.
9. On the **Advanced Options** tab, click **Save** to save the blueprint and to generate the corresponding action for the update configuration.
10. Click **Edit** next to the configuration to add more tasks to the update configuration.

Adding Multi-VM Update Configuration on Nutanix

As a blueprint developer, you can add an update configuration for a service to a multi-VM app blueprint on Nutanix.

About this task

The update configuration feature allows you to update virtual machines of running multi-VM apps to a higher or lower configuration. For more information, see [Update Configuration for VM](#) on page 205.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. Do one of the following:
 - » To add an update configuration to a new blueprint, select **Multi VM/Pod Blueprint** from the **+** **Create Blueprint** dropdown menu, and create a blueprint. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
 - » To add an update configuration to an existing blueprint, click the blueprint name to open the blueprint editor.
5. Ensure that you have added the service to which you want to add the update configuration. For more information on adding a service, see [Adding a Service](#) on page 130.
6. In the Overview Panel, click the **+** icon next to **Update Config**.

The **Update Config** window appears.

7. From the **Select Service to Update** dropdown menu, select the service to which you want to add the update configuration.
8. In the **Name the update configuration** field, specify a name for the configuration.
9. Under the **VM Configuration** section, select a change factor for the attributes and specify the value for the selected factor. To do that:
 - a. View the current value of vCPUs, No. of Cores, and Memory (GiB) in the Current Value column.
 - b. Click **Update** for the attribute that you want to update.
 - c. Select a value in the **Operation** dropdown menu for the attribute. You can select **Increase**, **Decrease**, or **Equal**.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then select **Increase** in the **Operation** dropdown menu.

Note: The update value is relative to the current value when you select **Increase** or **Decrease**. The update value is absolute when you select **Equal**.

- d. Specify an update value in the **Update** column based on the **Operation** value you selected.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then enter 1 in the **Update** field.
 - e. Specify the limit value to which the configuration of an attribute can be updated in the **Min Value** or **Max Value** column.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU to a maximum limit of 6, then specify 2 in the **Max Value** column. The vCPU of the VM can be updated until its value reaches 6vCPU. After the VM reaches 6 vCPU, more vCPUs cannot be added to the VM.

You can also turn on **Editable** of an attribute to allow your users to change its value within the limits you specify in the **Min Value** or **Max Value** column during the launch of the update.
10. Under the **Disks** section, do the following:
 - a. To add vdisk to the update configuration, click the + icon next to **Add/Edit vDisks to this VM**.
 - b. Select the device type and the device bus.
 - c. To define the disk size, specify the value for the vdisk size in the **Value** field.

You can turn on **Editable** and specify the **Min Value** and **Max Value** for the vdisk.
 - d. To allow your users to remove existing vdisks from the VM during the launch of the update, click **Advanced Settings** and select the **Allow users to remove existing vDisks** checkbox.
11. Under the **Categories** section, do the following:
 - a. To add to the update configuration, select the categories in the **Key: Value** dropdown menu.

Note: The categories you select must have the default SSH port (port 22) open in the security policies.

- b. To allow your users to remove existing categories during the launch of the update, click **Advanced Settings** and select the **Allow users to remove existing categories** checkbox.
 - c. To allow your users to add new categories during the launch of the update, select the **Allow users to add new categories** checkbox.

12. Under the **Network Adapters** section, do the following:
 - a. To add more NICs to the update configuration, click the **+** icon next to **Add/Edit NICs to this VM** and select the NIC from the list.
You can turn on **Editable** to allow your users to choose the desired subnet during the launch of the update.
 - b. To allow your users to remove existing NICs during the launch of the update, click **Advanced Settings** and select the **Allow users to remove existing NICs** checkbox.
13. Click **Done** to save the update configuration.
Saving the update config generates the **Config** component. The Config component lets you open the **Update Config** window to edit the update configuration.
14. On the Blueprint Editor page, click **Save** to save the blueprint and generate the corresponding action for the update configuration.
Saving the blueprint generates the **Action** component. The auto-generated **Action** component performs the start and stop of the service. You can also add tasks and actions to the component to define how you want your users to launch the update.

Adding Single-VM Update Configuration on VMware

As a blueprint developer, you can add an update configuration to a single-VM app blueprint on VMware.

About this task

The update configuration feature allows you to update the virtual machine of a running single-VM app to a higher or lower configuration. For more information, see [Update Configuration for VM](#) on page 205.

Before you begin

- Ensure that you set up your single-VM blueprint. For more information, see [Setting up a Single-VM Blueprint](#) on page 106.
- Ensure that you added the VM details to your blueprint. For more information, see [Adding VM Details to a Blueprint](#) on page 107.
- Ensure that you configured the VM in your blueprint. For more information, see [VM Configuration](#) on page 107.

Procedure

1. On the **Advanced Options** tab, next to Update Configs, click the **+ Add Config** option.
The **+ Add Config** option is available only after you add VM credentials for packages and actions enablement. For more information, see [Adding Credentials](#) on page 173.
2. On the **Update Configs** tab, type a name in the **Name the update configuration** field to change the name of the configuration.

3. Under the **VM Configuration** section, select a change factor for the attributes and specify the value for the selected factor. To do that:

- a. Select **CPU Hot Add** to allow addition of vCPUs while the virtual machine is turned on.

CPU resources cannot be added to an ESXi VM when it is powered on. The **CPU Hot Add** option lets you add CPU resources to a running VM.

Note: Before you use this feature, ensure that:

- You have the latest version of VMware Tools installed.
- You have the guest operating system that supports the CPU hot add functionality.
- Your virtual machine is compatible with ESXi 4.x or later.

- b. Select **Memory Hot Plug** to allow addition of memory resources while the virtual machine is turned on.

Note: Before you use this feature, ensure that:

- You have the latest version of VMware Tools installed.
- You have the guest operating system that supports the memory hot plug functionality.
- Your virtual machine is compatible with ESXi 4.x or later.

- c. View the current value of the vCPUs, Cores per Socket, and Memory (GiB) attributes in the Current Value column.

- d. Click **Update** for the attribute that you want to update.

Update Configs Variables

Name the update configuration

Config1

VM Configuration

Update the VM specifications as needed. Marking an attribute editable will allow the end-user to change its value within the limits specified via Min / Max. If the operation selected is 'Increase' or 'Decrease', the Update value should be relative. If 'Equal' is selected, the Update value should be absolute.

☒ CPU Hot Add ☒ Memory Hot Plug

	Current Value	Operation	Update	Editable	Min Value	Max Value	
vCPUs	4	Equal	6	<input checked="" type="checkbox"/>	1	6	Cancel
Cores per Socket	1	Equal	2	<input type="checkbox"/>	1	6	Cancel
Memory (GiB)	4	Equal	6	<input checked="" type="checkbox"/>	1	6	Cancel

Figure 62: Single-VM Update Config Options

- e. Select a value in the **Operation** dropdown menu for the attribute. You can select **Increase**, **Decrease**, or **Equal**.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then select **Increase** in the **Operation** dropdown menu.

Note: The update value is relative to the current value when you select **Increase** or **Decrease**. The update value is absolute when you select **Equal**.

- f. Specify an update value in the **Update** column based on the **Operation** value you selected.
For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then enter 1 in the **Update** field.
- g. Specify the limit value to which the configuration of an attribute can be updated in the **Min Value** or **Max Value** column.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU to a maximum limit of 6, then specify 2 in the **Max Value** column. The vCPU of the VM can be updated until its value reaches 6vCPU. After the VM reaches 6 vCPU, more vCPUs cannot be added to the VM.

You can also turn on **Editable** of an attribute to allow your users to change its value within the limits you specify in the **Min Value** or **Max Value** column during the launch of the update.

- 4. In the **VM Location** section, specify the location of the folder in which the VM must be created when you deploy the application blueprint. Ensure that you specify a valid folder name already created in your VMware account.

To create a subfolder in the location you specified, select the **Create a folder/directory structure here** checkbox and specify a folder name in the **Folder/Directory Name** field.

Note: Self-Service gives preference to the VM location specified in the environment you select while launching an app. For example, you specify a subfolder structure as the VM location in the blueprint and the top-level folder in the environment. When you select this environment while launching your app, Self-Service considers the VM location you specified in the environment and creates the VM at the top-level folder.

- 5. Under **Controllers**, click the **+** icon to add the type of controller.

You can select either SCSI or SATA controller. You can add up to three SCSI and four SATA controllers.

Select the **Allow users to remove existing vControllers** checkbox to allow the removal of existing controllers during the launch of the update.

6. Under the **Disks** section, do the following:

- a. To add vdisk to the update configuration, click the **+** icon next to **Add/Edit vDisks to this VM**.
- b. To allow removal of existing vdisks from the VM during the launch of the update, select the **Allow users to remove existing vDisks** checkbox.
- c. Expand the **VDISK** that you want to configure.
- d. Select the device type from the **Device Type** dropdown menu.
You can either select **CD-ROM** or **DISK**.
- e. Select one of the following adapter type from the **Adapter Type** dropdown menu.
 - Select **IDE** for CD-ROM.
 - Select **SCSI, IDE, or SATA** for DISK.
- f. In the **Location** or **Datastore** field, select the VM datastore location.
- g. If you want to add a controller to the vDisk, select the type of controller in the **Controller** dropdown menu to attach to the disk.

Note: You can add either SCSI or SATA controllers. The available options depend on the adapter type.

- h. In the **Disk mode** dropdown menu, select the type of the disk mode. Your options are:
 - » **Dependent:** Dependent disk mode is the default disk mode for the vDisk.
 - » **Independent - Persistent:** Disks in persistent mode behave like conventional disks on your physical computer. All data written to a disk in persistent mode are written permanently to the disk.
 - » **Independent - Nonpersistent:** Changes to disks in nonpersistent mode are discarded when you shut down or reset the virtual machine. With nonpersistent mode, you can restart the virtual machine with a virtual disk in the same state every time. Changes to the disk are written to and read from a redo log file that is deleted when you shut down or reset.
- i. To define the disk size, specify the value for the vdisk size in the **Value** field.
You can turn on **Editable** and specify the **Min Value** and **Max Value** for the vdisk.

7. Under the **Tags** section, do the following:

- a. To allow your users to add new tags during the launch of the update, select the **Allow users to add new Tags** checkbox.
- b. To allow your users to remove existing tags during the launch of the update, select the **Allow users to remove existing Tags** checkbox.
- c. To add tags to the update configuration, select the tags in the **Category: Tag Pairs** dropdown menu.

8. Under the **Network Adapters** section, do the following:

- a. To allow removal of existing NICs during the launch of the update, select the **Allow users to remove existing NICs** checkbox.
- b. To add more NICs to the update configuration, click the **+** icon next to **Add/Edit NICs to this VM** and select the NIC from the list.

9. To add variables to the update configuration, click the **Variables** tab at the top of the page and add variables.
The steps to add variables to the update configuration is similar to adding app variables. See [Configuring App Variables in a Blueprint](#) on page 126 and perform the steps to add variables in your update configuration.
10. Click **Done** to save the update configuration.
Saving the update config generates the Config component. The **Edit** button next to the Config component lets you open the **Update Config** window to edit the update configuration.
11. On the **Advanced Options** tab, click **Save** to save the blueprint.

Adding Multi-VM Update Configuration on VMware

As a blueprint developer, you can add an update configuration for a service to a multi-VM app blueprint on VMware.

About this task

The update configuration feature allows you to update the virtual machine of a running multi-VM app to a higher or lower configuration. For more information, see [Update Configuration for VM](#) on page 205.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
4. Do one of the following:
 - » To add an update configuration to a new blueprint, select **Multi VM/Pod Blueprint** from the **+ Create Blueprint** dropdown menu, and create a blueprint. For more information, see [Creating a Multi-VM Blueprint](#) on page 130.
 - » To add an update configuration to an existing blueprint, click the blueprint name to open the blueprint editor.
5. Ensure that you have added the service to which you want to add the update configuration. For more information on adding a service, see [Adding a Service](#) on page 130.
6. In the Overview Panel, click the **+** icon next to **Update Config**.
The **Update Config** window appears.
7. From the **Select Service to Update** dropdown menu, select the service to which you want to add the update configuration.
8. Type a name in the **Name the update configuration** field to change the name of the configuration.

9. Under the **VM Configuration** section, select a change factor for the attributes and specify the value for the selected factor. To do that:

- a. Select **CPU Hot Add** to allow addition of vCPUs while the virtual machine is turned on.

CPU resources cannot be added to an ESXi VM when it is powered on. The **CPU Hot Add** option lets you add CPU resources to a running VM.

Note: Before you use this feature, ensure that:

- You have the latest version of VMware Tools installed.
- You have the guest operating system that supports the CPU hot add functionality.
- Your virtual machine is compatible with ESXi 4.x or later.

- b. Select **Memory Hot Plug** to allow addition of memory resources while the virtual machine is turned on.

Note: Before you use this feature, ensure that:

- You have the latest version of VMware Tools installed.
- You have the guest operating system that supports the memory hot plug functionality.
- Your virtual machine is compatible with ESXi 4.x or later.

- c. View the current value of the vCPUs, Cores per Socket, and Memory (GiB) attributes in the Current Value column.

- d. Click **Update** for the attribute that you want to update.

Update Configs Variables

Name the update configuration

Config1

VM Configuration

Update the VM specifications as needed. Marking an attribute editable will allow the end-user to change its value within the limits specified via Min / Max. If the operation selected is 'Increase' or 'Decrease', the Update value should be relative. If 'Equal' is selected, the Update value should be absolute.

☒ CPU Hot Add ☒ Memory Hot Plug

	Current Value	Operation	Update	Editable	Min Value	Max Value	
vCPUs	4	Equal	6	<input checked="" type="checkbox"/>	1	6	Cancel
Cores per Socket	1	Equal	2	<input type="checkbox"/>	1	6	Cancel
Memory (GiB)	4	Equal	6	<input checked="" type="checkbox"/>	1	6	Cancel

Figure 63: Single-VM Update Config Options

- e. Select a value in the **Operation** dropdown menu for the attribute. You can select **Increase**, **Decrease**, or **Equal**.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then select **Increase** in the **Operation** dropdown menu.

Note: The update value is relative to the current value when you select **Increase** or **Decrease**. The update value is absolute when you select **Equal**.

- f. Specify an update value in the **Update** column based on the **Operation** value you selected.
For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU by 1, then enter 1 in the **Update** field.
- g. Specify the limit value to which the configuration of an attribute can be updated in the **Min Value** or **Max Value** column.

For example, if the original vCPU value in the blueprint is 4 and you want to increase the vCPU to a maximum limit of 6, then specify 2 in the **Max Value** column. The vCPU of the VM can be updated until its value reaches 6vCPU. After the VM reaches 6 vCPU, more vCPUs cannot be added to the VM.

You can also turn on **Editable** of an attribute to allow your users to change its value within the limits you specify in the **Min Value** or **Max Value** column during the launch of the update.

- 10. In the **VM Location** section, specify the location of the folder in which the VM must be created when you deploy the application blueprint. Ensure that you specify a valid folder name already created in your VMware account.

To create a subfolder in the location you specified, select the **Create a folder/directory structure here** checkbox and specify a folder name in the **Folder/Directory Name** field.

Note: Self-Service gives preference to the VM location specified in the environment you select while launching an app. For example, you specify a subfolder structure as the VM location in the blueprint and the top-level folder in the environment. When you select this environment while launching your app, Self-Service considers the VM location you specified in the environment and creates the VM at the top-level folder.

- 11. Under **Controllers**, click the **+** icon to add the type of controller.

You can select either SCSI or SATA controller. You can add up to three SCSI and four SATA controllers.

Select the **Allow users to remove existing vControllers** checkbox to allow the removal of existing controllers during the launch of the update.

12. Under the **Disks** section, do the following:

- a. To add vdisk to the update configuration, click the **+** icon next to **Add/Edit vDisks to this VM**.
- b. To allow removal of existing vdisks from the VM during the launch of the update, select the **Allow users to remove existing vDisks** checkbox.
- c. Expand the **VDISK** that you want to configure.
- d. Select the device type from the **Device Type** dropdown menu.
You can either select **CD-ROM** or **DISK**.
- e. Select the adapter type from the **Adapter Type** dropdown menu.
 - Select **IDE** for CD-ROM.
 - Select **SCSI, IDE, or SATA** for DISK.
- f. In the **Location** or **Datastore** field, select the VM datastore location.
- g. If you want to add a controller to the vDisk, select the type of controller in the **Controller** dropdown menu to attach to the disk.

Note: You can add either SCSI or SATA controllers. The available options depend on the adapter type.

- h. In the **Disk mode** dropdown menu, select the type of the disk mode. Your options are:
 - » **Dependent:** Dependent disk mode is the default disk mode for the vDisk.
 - » **Independent - Persistent:** Disks in persistent mode behave like conventional disks on your physical computer. All data written to a disk in persistent mode are written permanently to the disk.
 - » **Independent - Nonpersistent:** Changes to disks in nonpersistent mode are discarded when you shut down or reset the virtual machine. With nonpersistent mode, you can restart the virtual machine with a virtual disk in the same state every time. Changes to the disk are written to and read from a redo log file that is deleted when you shut down or reset.
- i. To define the disk size, specify the value for the vdisk size in the **Value** field.
You can turn on **Editable** and specify the **Min Value** and **Max Value** for the vdisk.

13. Under the **Tags** section, do the following:

- a. To allow your users to add new tags during the launch of the update, select the **Allow users to add new Tags** checkbox.
- b. To allow your users to remove existing tags during the launch of the update, select the **Allow users to remove existing Tags** checkbox.
- c. To add tags to the update configuration, select the tags in the **Category: Tag Pairs** dropdown menu.

14. Under the **Network Adapters** section, do the following:

- a. To allow removal of existing NICs during the launch of the update, select the **Allow users to remove existing NICs** checkbox.
- b. To add more NICs to the update configuration, click the **+** icon next to **Add/Edit NICs to this VM** and select the NIC from the list.

15. Click **Done** to save the update configuration.

Saving the update config generates the **Config** component. The Config component lets you open the **Update Config** window to edit the update configuration.

16. To add variables to the update configuration, click the update configuration and then add variables in the inspector panel.

The steps to add variables to the update configuration is similar to adding app variables. See [Configuring App Variables in a Blueprint](#) on page 126 and perform the steps to add variables in your update configuration.

17. On the Blueprint Editor page, click **Save** to save the blueprint and generate the corresponding action for the update configuration.

Saving the blueprint generates the **Action** component. The auto-generated **Action** component performs the start and stop of the service. You can also add tasks and actions to the component to define how you want your users to launch the update.

BLUEPRINTS MANAGEMENT IN SELF-SERVICE

After you configure a blueprint, you can publish, unpublish, launch, or delete a blueprint.

Blueprint Publishing

Publishing a blueprint allows you to make the blueprint available at Marketplace, so that other users can use the published blueprint. Unpublishing a blueprint allows you to remove the blueprint from the Marketplace. For more information, see [Submitting a Blueprint for Approval](#) on page 222.

Blueprint Launching

Launching a blueprint allows you to deploy your app on the blueprint and start using it.

The blueprint launch page provides the following views:

← Launch test_project_2_bp

View as Consumer (Editable fields only) ▲

View as Consumer (Editable fields only)

View as Developer (All fields)

Application Name

TestAppLaunch

Application Description

Application Description

Project

Total Cost : 0\$/month

Cancel Deploy

Figure 64: Blueprint Launch Views

- **View as Consumer:** This view of the blueprint launch page displays only the editable fields that consumers require to launch a blueprint. When you design your blueprint for consumers with minimum configurations at runtime, use this view to get an idea about the blueprint launching experience of your consumers.

Blueprints that are launched from the marketplace display only the fields that require inputs from consumers. Displaying only editable fields offers a simpler and easy launching experience for your consumers.

- **View as Developer:** This view of the blueprint launch page displays all editable and non-editable fields that you configure for the blueprint. As a blueprint developer, you can switch between View as Consumer and View as Developer on the blueprint launch page.

You can switch to View as Developer after you develop your blueprints to verify how you configured different fields and the launching experience the configuration will provide to your consumers.

For more information, see [Launching a Blueprint](#) on page 224.

Submitting a Blueprint for Approval

After you configure a blueprint, you can submit the blueprint to get an approval from the administrator. The administrator approves the blueprint and then publishes the blueprint to the marketplace for consumption.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Click the blueprint that you want to publish.
The blueprint editor page is displayed.
5. Click **Publish**.

Figure 65: Publish Blueprint window

The **Publish Blueprint** window is displayed.

6. If the blueprint is getting published for the first time, select **New Marketplace blueprint** and do the following.
 - a. Specify a name for the Marketplace blueprint in the **Name** field.
 - b. To publish the blueprint with secret variables, turn on **Publish with Secrets**.
 - By default, Self-Service does not preserve the secret values from the blueprint while publishing. The values are either patched from the environment or require your inputs. Turn on this option to preserve the secret values as is. Credential passwords or keys and secret variables are considered secret values. The values are encrypted when you publish with secrets.
 - Nutanix recommends that you turn on this option if your blueprint contains any task type as HTTP because the secrets from HTTP tasks once dropped cannot be patched or edited at runtime.
 - Endpoints with secrets (encrypted) are always preserved.
 - c. To clear all the platform-dependent fields and macros used in the Blueprint VM configuration before publishing the blueprint to the marketplace, turn on **Publish without platform-dependent configuration**.

Clearing the platform-dependent fields allows Self-Service to populate VM configuration fields from the VM configurations of the environment you select during the launch of the Marketplace blueprint. To view the list of platform-dependent fields, see [Environment Patching Behavior](#) in the *Prism Central Admin Center Guide*.
 - d. Enter the version number in the **Initial Version** field.

Note: Ensure that the version number is in the x.x.x format.
 - e. Enter the blueprint description in the **Description** field. This step is optional.
7. If you want to revise a published blueprint version, select **New version of an existing marketplace blueprint** and do the following.
 - a. To publish the blueprint with secret variables, enable **Publish with Secrets**.
 - b. To clear all the platform-dependent fields and macros used in the Blueprint VM configuration before publishing the blueprint to the marketplace, turn on **Publish without platform-dependent configuration**.
 - c. Select the already published blueprint from the **Marketplace Item** dropdown menu.
 - d. Enter the version number in the **Version** field.

Note: Ensure that the version number is in the x.x.x format.
 - e. Enter the blueprint description in the **Description** field. This step is optional.
 - f. Enter the log changes on the **Change Log** tab.

8. If you want to upload an icon for the blueprint, click **Change**.
 - a. Click **Upload from computer** to browse and select an image from your local machine.
 - b. Click **Open**.
 - c. Provide a name to the image in the **Name of the Icon** field.
 - d. Click the right icon.
 - e. Click **Select & Continue**.

Note: User with administrator role can only upload an icon.

9. Optionally, if you want to select an icon, already available in a blueprint, click the right icon.
10. Optionally, to delete an icon, click the delete icon.
11. Click **Submit for Approval**.

The blueprint is submitted to the marketplace manager for approval. Your administrator can find the submitted blueprint on the **Approval Pending** tab of the Marketplace Manager page.

What to do next

You can request your administrator to approve and publish the blueprint to the marketplace. For more information on blueprint approval and publishing, see [Approving and Publishing a Blueprint or Runbook](#) on page 232.

Launching a Blueprint

You launch a blueprint to deploy an app on the blueprint and start using the app.

Before you begin

For blueprints on a Nutanix platform, ensure that you have created the snapshot policy. For more information on snapshot policy creation, see [Creating a Snapshot Policy](#).

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.

The **Blueprints** page displays the list of all the available blueprints.
4. Click the blueprint that you want to launch.

The blueprint details page is displayed.
5. Click **Launch**.

The blueprint launch page is displayed.
6. Enter a name for the app in the **Application Name** field.
7. Enter a description for the app in the **Application Description** field.

8. Select the environment from the **Environment** dropdown menu.
If you select an environment that is different from the account that you used for blueprint configuration, Self-Service updates all platform-dependent fields to match with the selected environment configuration.
For example, you created the app blueprint using an account with an environment (ENV1) so that the platform-dependent fields are similar to ENV1. While launching the app blueprint, if you select a different environment (ENV2), Self-Service updates all platform-dependent fields to match with the ENV2 configuration. For more information, see [Environment Patching Behavior](#) on page 81 and [Patching for Clusters and Subnets](#) on page 85.
9. Select the app profile from the **App Profile** field.
In case, any of the fields are marked runtime while creating the blueprint, those fields are editable and displayed here. To view the runtime variables, expand the service under **VM Configurations**.
10. In the sections for the service configuration and credentials configuration, verify and edit the configuration requirements for your app services and credentials.

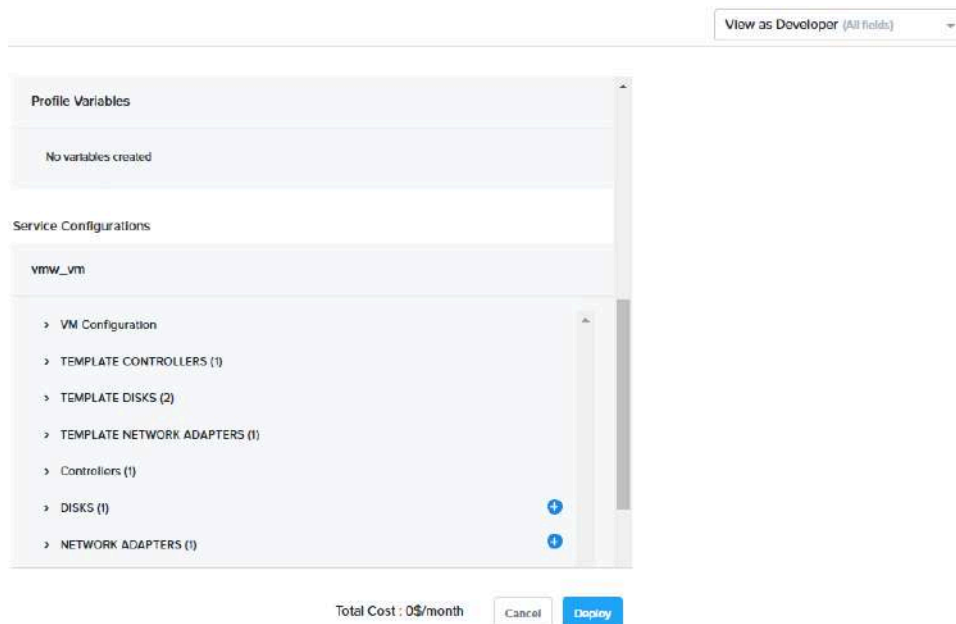


Figure 66: Blueprint Launch - Service Configuration

Use the **View as Developer** option at the top of the blueprint launch page to view all configuration fields.

Note: The **View as Consumer** view displays only the editable fields while the **View as Developer** view displays all configuration fields for your services and credentials. As a developer, you can select the **View as Developer** to view the configuration details of all fields.

11. If the blueprint is configured with a Nutanix or VMware account, do the following:
 - a. Under Snapshot Configurations, select a snapshot policy in the **Select Policy** dropdown menu.
 - b. For the blueprint with a Nutanix account, based on the policy you select, select a rule in the **Select Local Rule** or **Select Remote Rule** dropdown menu.
The **Select Local Rule** or **Select Remote Rule** dropdown menu appears based on the **Snapshot Location** you defined in your blueprint. For more information, see [Blueprint Configuration for Snapshots and Restore](#) on page 199. The values in the list appear based on the snapshot

policy you defined in the project and selected in the Snapshot Policy dropdown menu. For more information, see [Creating a Snapshot Policy](#). The values also depend on the VM categories you configured in your blueprint.

Note: Ensure that you have a valid NIC in the blueprint.

For the blueprint with a VMware account, only the local rule is applicable for the policy. By default, the **Select Local Rule** dropdown menu displays the **Snapshots have no expiry** option. You do not have to select any option for the rule.

The Snapshot Configuration section appears depending on the environment you select while launching the blueprint. If you select a specific environment, you must provide the snapshot policy and snapshot rule to launch the blueprint. The Snapshot Configuration section does not appear in case you select the environment with all project accounts for the launch.

12. Click **Deploy.**

The system validates the provided platform-specific data against the selected provider and if the validation fails, an error message appears. For more information on the validation error, see [Platform Validation Errors](#) on page 226.

If the validation is successful, the app is available under My Apps in Admin Center. For more information, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Platform Validation Errors

When you enter the platform data that is invalid for a provider while creating a blueprint, you get a validation error. The following table details the invalid platform data for each provider.

Providers	Invalid Platform Data
Nutanix	Image, NIC List, and Categories.
GCP	Machine Type, Disk Type, Network, SubNetwork, Source, Image, Zone, and Blank Disk.
AWS	VPC, Security Groups, and Subnets.
VMware	Network name, NIC Type, NIC settings mismatch, Host, Template, Datastore, Datacenter, Storage Pod, and cluster.
Azure	Image details (publisher, offer, SKU, version), Custom image, Resource group, Availability Set Id, NIC List, Network Security group, Virtual Network Name, and Subnet Name.

Uploading a Blueprint

You can also upload configured blueprints to the Blueprints tab. Perform the following procedure to upload a blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.

4. Click **Upload Blueprint**.
The browser window is displayed.
5. Navigate to the location of the saved blueprint and select the blueprint.
6. Click **Open**.
The **Upload Blueprint** window is displayed.
7. Enter the name of the blueprint in the **Blueprint Name** field.
8. Select the project from the **Project** dropdown menu.
9. Click **Upload**.
The blueprint is uploaded and available for use.

Note: You must provide the credentials password or key of the blueprint.

Downloading a Blueprint

You can also download a configured blueprint to your local machine and use it later. Perform the following procedure to download a blueprint.

Before you begin

Ensure that at least one blueprint must be available.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Do one of the following.
 - » Click the blueprint that you want to download and click **Download**.
 - » Select the blueprint that you want to download and **Action > Download**.
The **Download Blueprint** window appears.
5. Optionally, if you want to download the blueprint with the credentials and secrets used in the blueprint, click the checkbox in the **Download Blueprint** window.
6. In the **Enter Passphrase** field, type a password.
The **Enter Passphrase** field is a mandatory field and is activated only after you have clicked the checkbox to download the blueprint with credentials and secrets.
7. Click **Continue**.
The blueprint is downloaded to your local machine.

Viewing a Blueprint

Perform the following procedure to view a blueprint.

Procedure

1. Log in to your Prism Central instance.

2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Click the blueprint that you want to view the details of.
The selected blueprints details are displayed.

Editing a Blueprint

You can edit a configured blueprint from the blueprints tab. Perform the following procedure to edit a blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Click the blueprint that you want to edit.
The blueprint details page is displayed.
5. Make the necessary edits in the layers (**Services**, **Actions**, and **Application Profiles**).

Note: You cannot delete System level actions.

6. Click **Save**.

Deleting a Blueprint

Perform the following procedure to delete a blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Select the listed blueprint that you want to delete.
5. Click **Actions > Delete**.
6. Click **Yes** to confirm.
The blueprint is deleted.

Viewing Blueprint Error

If you have configured wrong details in your blueprint, you can view the error message while saving or publishing a blueprint. Perform the following procedure to view blueprint error message.

Procedure

1. Log in to your Prism Central instance.

2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. Click the blueprint that you want to view the details.
The selected blueprint details are displayed. If there is any error in the blueprint, then the error is denoted by !.
5. Click !.

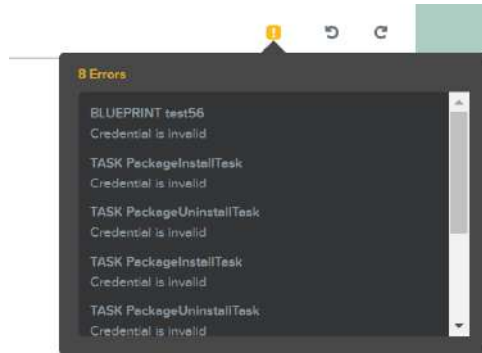


Figure 67: Blueprint Error

The blueprint errors are displayed.

Recovering Deleted Blueprints

You can recover the deleted app blueprints within a time period of 90 days after you delete an app blueprint. This chapter describes the procedure to recover a deleted blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Blueprints** in the navigation bar.
The **Blueprints** page displays the list of all the available blueprints.
4. In the search filter field, enter `State:Deleted` and press Enter.
You can view the list of all deleted blueprints based on the 90 days retention period.
5. Click the blueprint that you want to recover.
6. From the **Action** dropdown menu, select **Clone**.
The **Clone Blueprint** page appears.
7. In the **Blueprint Name** field, enter a name for the blueprint and click **Clone**.
The name is used as the blueprint name after recovery.
A clone of the deleted blueprint is created and you can view the recovered blueprint on the Blueprints page with the new name.

NUTANIX MARKETPLACE OVERVIEW

Nutanix Marketplace provides consumers and publishers a common platform to instantly consume the required app resources with a variety of provisioning options based on roles.

The Marketplace contains pre-seeded and validated blueprints to enable IT teams quickly introduce new technologies into their environments. IT teams can use these pre-seeded blueprints to quickly pilot new tools with developers and collaboratively decide their adoption. You can also create custom blueprints and publish them to the Marketplace to have your own private marketplace of apps in addition to the pre-seeded blueprints.

You can execute application delivery in a repeatable way to eliminate extensive and routine app provisioning and management. You can deploy on-premises, in public clouds, or both to implement a multi-cloud strategy. You can also have complete visibility into the consumption and costs of resources across clouds.

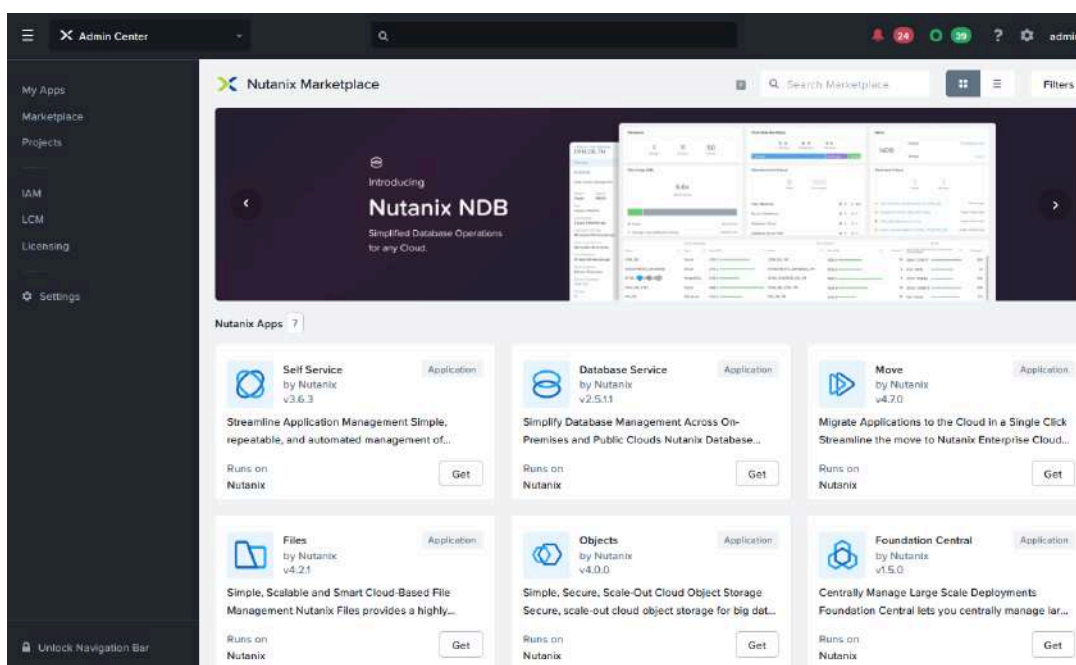


Figure 68: Nutanix Marketplace

Note:

- Nutanix recommends you to use the pre-seeded Marketplace apps without any customization.
- Customizing a pre-seeded application voids the Self-Service support for the application.

You can view and launch the Marketplace apps after you enable Marketplace in the Admin Center. For more information on enabling Marketplace, Marketplace app category, and app deployment, see [Marketplace](#) in the *Prism Central Admin Center Guide*.

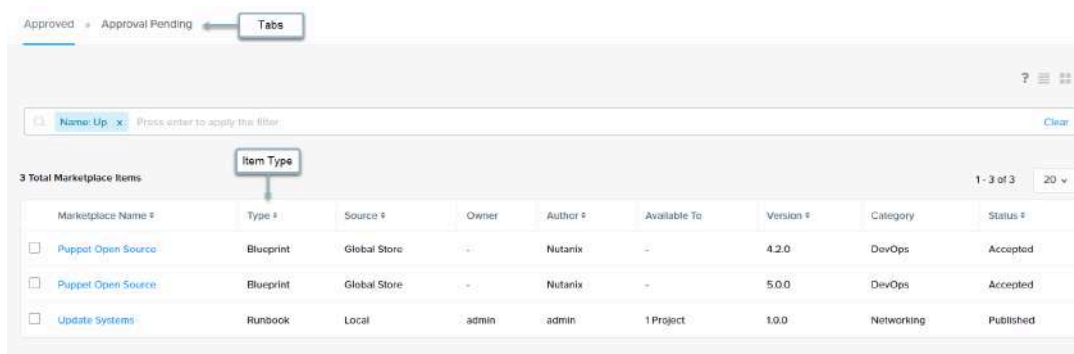
MARKETPLACE MANAGER IN SELF-SERVICE

The topics in this section cover the usage of the Marketplace Manager functionality in Self-Service.

Marketplace Manager Overview

Use Marketplace Manager to manage the list of custom blueprints, ready-to-use marketplace app blueprints, and runbooks. You can approve, reject, launch, publish, unpublish, assign a category, and select projects for a blueprint. You can also approve, reject, publish, unpublish, and execute runbooks.


The **Approved** tab on the Marketplace Manager page provide you a list of ready-to-use app blueprints and the custom blueprints or runbooks you approved. The **Approval Pending** tab provides a list of custom blueprints and runbooks that require your approval to be available in the Marketplace for consumption.



Marketplace Name #	Type #	Source #	Owner	Author #	Available To	Version #	Category	Status #
<input type="checkbox"/> Puppet Open Source	Blueprint	Global Store	-	Nutanix	-	4.2.0	DevOps	Accepted
<input type="checkbox"/> Puppet Open Source	Blueprint	Global Store	-	Nutanix	-	5.0.0	DevOps	Accepted
<input type="checkbox"/> Update Systems	Runbook	Local	admin	admin	1 Project	1.0.0	Networking	Published

Figure 69: Marketplace Manager

When you select a blueprint or runbook from the list on any tab, the inspector panel displays the operations you can perform on the selected blueprint or runbook. The inspector panel also displays a brief overview of the blueprint or runbook and allows you to assign projects to blueprint or runbook.



test_service_level_user_a...

Description

test_publish

Category

Select...

Projects Shared With

test_rbac

Figure 70: Inspector Panel

You can perform the following actions on blueprints or runbooks.

- You can approve blueprints or runbooks and publish them to the marketplace for consumption. You can also publish the ready-to-use app blueprints to the marketplace. For more information, see [Approving and Publishing a Blueprint or Runbook](#) on page 232.
- You can unpublish blueprints or runbooks to remove them from the marketplace. For more information, see [Unpublishing a Blueprint or Runbook](#) on page 233.
- You can also delete an unpublished blueprint or runbook. For more information, see [Deleting an Unpublished Blueprint or Runbook](#) on page 235.

Marketplace Version

Marketplace version enables you to define the initial version number of the blueprint or runbook that is getting published to the marketplace. Marketplace version also enables you to revise the version of a blueprint or runbook that is already published to the marketplace. For information on how to define marketplace version, see [Submitting a Blueprint for Approval](#) on page 222 or [Submitting a Runbook for Publishing](#) on page 283.

Approving and Publishing a Blueprint or Runbook

You can approve custom blueprints or runbooks that are submitted for approval on the **Approval Pending** tab. You can also publish the approved blueprints or runbooks to the marketplace after associating them with a project on the **Approved** tab.

About this task

The **Approved** tab also displays the ready-to-use app blueprints that are available after enabling Nutanix Marketplace in the Admin Center. These app blueprints do not require approval and can be published directly to the marketplace after associating them with a project.

Before you begin

- To publish a blueprint, ensure that you have configured a blueprint and submitted the blueprint for approval. For more information, see [Self-Service Blueprints Overview](#) on page 87 and [Submitting a Blueprint for Approval](#) on page 222.
- To publish a runbook, ensure that you have submitted a runbook for publishing. For more information, see [Submitting a Runbook for Publishing](#) on page 283.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Marketplace Manager** in the navigation bar.
The **Marketplace Manager** page is displayed.

- Click the **Approval Pending** tab to get the list of all unpublished blueprints and runbook requests. A list of all the unpublished blueprint and runbook requests is displayed.

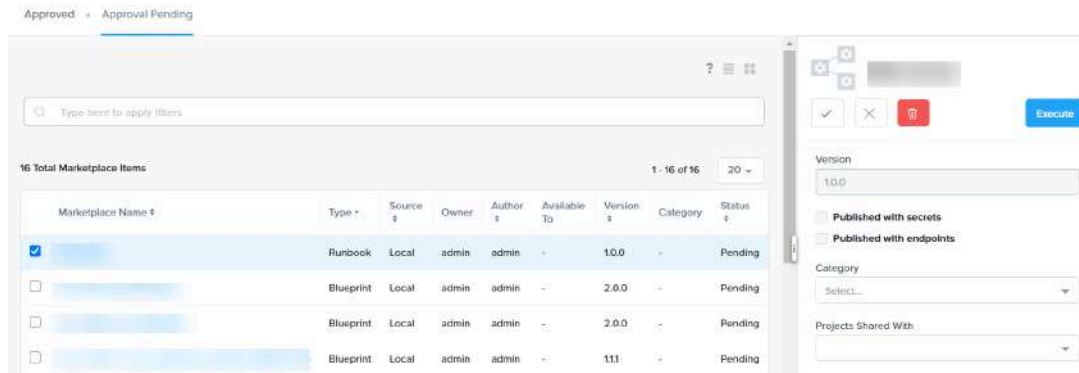


Figure 71: Marketplace Manager Approval Pending

- Select the blueprint or runbook that you want to approve and publish. The inspector panel appears.
- Click the check mark button to approve.
- Click the **Approved** tab to get the list of all approved blueprints and runbooks. A list of all the approved blueprints and runbooks is displayed.
- Select the approved blueprint or runbook that you want to publish.

Note: You can also select a ready-to-use marketplace app blueprint on the **Approved** tab for publishing.

- In the Inspector Panel, select the category from the **Category** dropdown menu. You can also add a new app category value and select the value for publishing. To add a new category value for apps, you need to add the value to the AppFamily category. To know how to add a value to a category, see [Category Management](#) in the *Prism Central Infrastructure Guide*.
- Select one or more projects from the **Projects Shared With** dropdown menu.
- Click **Apply**.
- Click **Publish**. The blueprint or runbook will be published in the marketplace.

What to do next

You can launch the published blueprint or execute the published runbook from the Marketplace in Admin Center. For more information, see [Marketplace](#) in the *Prism Central Admin Center Guide*.

Unpublishing a Blueprint or Runbook

You can unpublish a blueprint or runbook if you do not want to list it in the Marketplace. You can publish the blueprint or runbook again if required.

Procedure

- Log in to your Prism Central instance as an administrator.
- Select **Self-Service** in the Application Switcher.

3. Click **Marketplace Manager** in the navigation bar.
The **Approved** tab lists all the published blueprints and runbooks.
4. Select the blueprint or runbook that you want to unpublish.
The inspector panel is displayed.
5. Click **Unpublish**.
The blueprint or runbook is unpublished and does not appear in the marketplace.

Deploying a Pre-Seeded App from Self-Service

The Other Apps category includes third-party or open-source apps (pre-seeded apps) and custom apps that are available for deployment only when Self-Service is deployed in your Prism Central instance.

About this task

As a Prism Admin, you can deploy a pre-seeded app from the Marketplace Manager in Self-Service.

Before you begin

Ensure that you have a project with an environment configured to deploy the app from the Marketplace. For more information, see the [Project Management](#) section in the *Prism Central Admin Center Guide*.

Procedure

1. Log in to your Prism Central instance as Prism Admin.
2. Select **Self-Service** in the Application Switcher.
3. Click **Marketplace Manager** in the navigation bar.
4. Type the name of the app that you want to deploy in the **Search** field.
5. Click the app name to open the app details in the inspector panel.
6. Do the following to assign a project to the application.
 - a. Select a project from the **Project Shared With** dropdown menu.
 - b. Click **Apply**.
You can assign multiple projects to the application.
7. Click **Deploy**.
The Deploy page is displayed.
8. Enter a name for the application in the **Application Name** field.
Following are the rules for naming convention.
 - The name of the application can start with an alphanumeric character or an underscore.
 - The name must have at least one character.
 - Use only space, underscore, and dash as special characters.
 - Do not end the name with a dash.
9. Enter a description for the application in the **Application Description** field.
10. Select the project from the **Project** dropdown menu.

11. Select the environment from the **Environment** dropdown menu.
If you select an environment that is different from the account that you used for blueprint configuration, Self-Service updates all platform-dependent fields to match with the selected environment configuration.
For example, you created the application blueprint using an account with an environment (ENV1) so that the platform-dependent fields are similar to ENV1. While launching the application blueprint, if you select a different environment (ENV2), Self-Service updates all platform-dependent fields to match with the ENV2 configuration. For more information, see the [Environment Patching Behavior](#) section in the *Prism Central Admin Center Guide*.
12. Select an application profile in the **App Profile** field.
Application profile provides different combinations of the service, package, and VM while configuring a blueprint.
13. In the section for the service configuration, verify the VM, disk, boot configuration, and network configuration. You can edit the fields based on your application-specific requirements.
14. Click **Deploy**.
The deployed app appears under **My Apps** in the Admin Center. For more information on My Apps, see [My Apps](#) in the *Prism Central Admin Center Guide*.

Deleting an Unpublished Blueprint or Runbook

You can delete a blueprint or runbook that is not published in the marketplace. If you want to delete a published blueprint or runbook, you first have to unpublish it and then delete it.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Self-Service** in the Application Switcher.
3. Click **Marketplace Manager** in the navigation bar.
The **Marketplace Manager** page is displayed.
4. Do one of the following:
 - » To delete a blueprint or runbook that is not yet approved, click the **Approval Pending** tab.
 - » To delete a blueprint or runbook that is approved and unpublished, click the **Approved** tab.
5. Select the blueprint or runbook that you want to delete.
The inspector panel appears.
6. Click the **Delete** icon.
The blueprint or runbook is deleted from the marketplace manager.

APPLICATIONS IN SELF-SERVICE

The application management functionality of Self-Service allows you to standardize and automate the entire application life cycle to ensure consistency, reduce errors, and speed up deployment times.

The following table lists the applications that you design or enable in Self-Service and are available for deployment only when Self-Service is enabled and licensed in your Prism Central instance.

Table 38: Applications in Self-Service

Application Type	Description
Hybrid Cloud Apps	<p>The hybrid cloud apps category in Self-Service includes applications that Nutanix has certified to operate across multiple computing environments, typically a combination of on-premises datacenters (private clouds) and public cloud services. These applications leverage the benefits of both private and public clouds, such as enhanced flexibility, scalability, cost efficiency, and improved disaster recovery capabilities.</p> <p>For example, you can host database and critical business logic on-premises to ensure data security and compliance and web front-end and mobile application interfaces in the public cloud for scalability and efficient user traffic handling.</p> <p>For information on how to deploy the pre-seeded hybrid cloud apps, such as LAMP, Icinga, CouchDB, Docker Swarm, and so on, see the Self-Service Marketplace Items Guide.</p>
Custom Apps	<p>The custom apps category in Self-Service includes applications that you can uniquely design and configure to meet your specific business needs and requirements. You define these applications through custom blueprints that provide a detailed template to outline the application architecture, components, configurations, and dependencies.</p> <p>Use of blueprints in custom apps ensures consistent deployments across different environments (on-premises, in the cloud, or in a hybrid setup). The RBAC capabilities provide control over who can access or modify applications, ensuring compliance with organizational policies. For information on custom app configuration, see Self-Service Blueprints Overview on page 87.</p> <p>You can publish custom applications in the Nutanix Marketplace for end-user consumption after they are approved by an administrator. For information on approval and publishing of custom apps, see Marketplace Manager Overview on page 231.</p>

For information on how you can manage the deployed hybrid cloud apps and custom apps, see [My Apps](#) in the *Prism Central Admin Center Guide*.

My Apps

My Apps provides a single workspace to manage all the applications that you deployed from the Marketplace or Self-Service. You can access and manage apps in My Apps only after your Prism Central administrator enables the Marketplace. For more information, see [Enabling Marketplace](#) in the *Prism Central Admin Center Guide*.

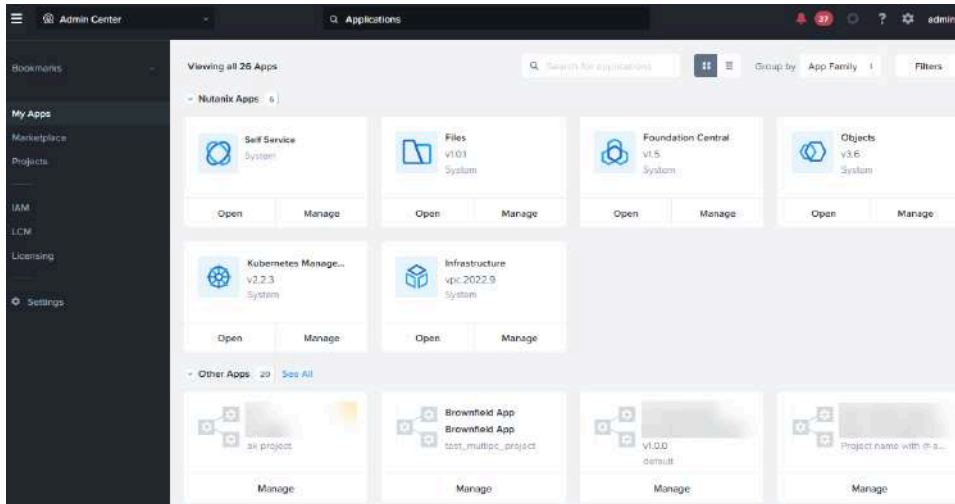


Figure 72: My Apps

For more information on app management in My Apps, see [My Apps](#) in the *Prism Central Admin Center Guide*.

POLICIES IN SELF-SERVICE

The topics in this section cover the configuration and usage of scheduler policies and approval policies in Self-Service.

Scheduler Overview

Scheduler allows you to schedule app action and runbook executions. You can schedule recurring jobs and one-time jobs for critical operations throughout the app life cycle.

You can schedule any user-defined app actions, create or restore app snapshots (only AHV), or any pre-defined system actions such as Start, Stop, Restart, Delete, and Soft Delete. For example, you can schedule a Stop action and a Start action on a single-VM Self-Service app to run at a particular date and time.

Scheduler supports two types of entities.

- **Application action.** You can use scheduler to schedule app actions, such as Start and Stop, to run at a particular date and time. You can also schedule any custom-defined actions and snapshot create and restore action for AHV.
- **Runbook execution.** You can schedule runbooks to run on a particular date and time.

Scheduler jobs have a role ownership. A user can modify the job that you created if the user has access to the entity and **Allow Collaboration** is enabled in the associated project. For example, if you create a scheduler job for an app action as a developer, a consumer that has access to the same app can modify the job. If **Allow Collaboration** is disabled in the project, then only the creator of the scheduler job can modify the job. For information on the role required to schedule app action and runbook execution, see [Self-Service Role-Based Access Control](#) on page 18.

Creating a Scheduler Job

Create a scheduler job to perform an app action or runbook execution.

Before you begin

Ensure that you have enabled the policy engine in Self-Service Settings. For information on enabling the policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Scheduler** tab, click **+Create Job** to create a job. The **Create Job** page appears.
5. In the **Job Name** field, type a name for the job.
6. Enter a description for the job. This step is optional.

7. From the **Select Action** dropdown menu, select an entity type that you want the scheduler job to run on. Your options are:
 - » Select **Application Action** to schedule app actions such as start and stop, schedule any user-defined actions, or snapshot create and restore action for AHV.
 - » Select **Execute Runbook** to schedule the execution of a runbook.
8. From the **Project** dropdown menu, select the project associated with your app or runbook.
9. Click **Action Details**.
10. If you have selected **Application Action** as the action type, then do the following:
 - a. From the **Select Application** dropdown menu, select the app for which you want to schedule an action.

The **Select Application** dropdown menu displays only those apps that are associated with the project you selected on the **Job Details** tab.
 - b. From the **Select Application Action** dropdown menu, select an app-level action or a user-defined action.
 - c. If you select any user-defined action, then review or edit the variables for the selected action.
11. If you have selected **Execute Runbook** as the action type, then do the following:
 - a. From the **Runbook** dropdown menu, select the runbook for which you want to schedule an action.

The **Runbook** dropdown menu displays only those runbooks that are associated with the project you selected on the **Job Details** tab.
 - b. From the **Default Endpoint** dropdown menu, select a default endpoint for the runbook execution.
 - c. Verify the variables and endpoint data (such as base URLs, IP addresses, and VMs) defined for the runbook.
12. Click **Set Schedule**.
13. Under **Schedule Type**, select **Recurring Job** to run the job at regular intervals or **One-Time Job** to run the job only once.
14. If you have selected **Recurring Job**, then do the following:
 - a. Under **Starts**, define the start date and time in the **Start on** and **Start at** fields if you want to start the job at a particular date and time. This step is optional.
 - b. Under **Ends**, select **Never** to run the job indefinitely or **On** to define the ending date and time for the job.
 - c. In the **Select Timezone** field, select a location to define the time zone for the schedule.
 - d. Under **How often does the job occur** section, select an option in the **Every** field to define the frequency of the job schedule.

You can also enter a cron expression to define the frequency of the job schedule.

Depending on the option you select, you have to define specific criteria for the job frequency. For example, when you select **Year**, you also have to define the months, days, day of the week, hours, and minutes.

15. If you have selected **One-Time Job**, then do the following:
 - a. In the **Executes on** field, specify the execution date.
 - b. In the **Executes at** field, specify the execution time.
 - c. In the **Select Timezone** field, select a location to define the time zone.
16. Click **Save**.

Viewing and Updating Scheduler Jobs

You can view or update a scheduler job on the Scheduler tab of the Policies page.

About this task

Scheduler jobs have a role ownership. You can update a job that a different user has created only when you have access to the entity and collaboration is allowed in the associated project.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Scheduler** tab, click the job that you want to view or update.

5. On the job details page, do the following:

- On the **Job Info** tab, view the action type, runbook or app name, the next scheduled date, execution time or recurrence, and the state of the job. A job can show one of the following states.
 - Active: When the job is created or updated without any errors and the job has not completed the last execution and has not crossed the last execution date.
 - Inactive: When the job is created or updated with errors.
 - Expired: When the last execution of the job is complete or the job has already crossed the last execution date.

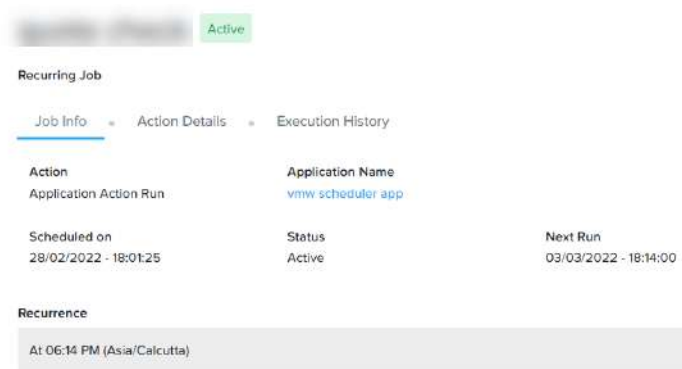


Figure 73: Job Info

- On the **Action Details** tab, view the following action details:
 - For an app action job, view the associated app and app action.
 - For a runbook job, view the associated runbook, default endpoint, variable details, and endpoint date.
- On the **Execution History** tab, view the scheduled time, execution status, and execution time. A job can have one of the following execution statuses.
 - Executed: When the job is already executed as scheduled.
 - Running: When the job is running as scheduled.
 - Success: When a job run is completed successfully.
 - Aborted: When you manually cancel a running or scheduled job.
 - Failed: When the job failed to execute because of some errors.

You can also click **View Logs** for any executed job to go to the **Audit** tab and view the logs.

6. To edit the job, click **Update** and edit the details of the job. For more information on the fields, see [Creating a Scheduler Job](#) on page 238.

Deleting a Scheduler Job

You can delete a scheduler job on the Scheduler tab of the Policies page.

Procedure

1. Log in to your Prism Central instance.

2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Scheduler** tab, click the job that you want to delete.
5. From the **Action** dropdown menu, click **Delete**.
6. In the Confirm Delete window, click **Delete**.

Approval Policy Overview

An approval policy adds a level of governance to determine which app deployment requests or actions require approvals before they are initiated. You can use approval policies to manage your infrastructure resources, their associated costs, and compliance more effectively.

For example, consider a marketplace item that consumes a significant part of your available resources. You can use an approval policy to enable your IT administrator to review all deployment requests for that marketplace item and ensure that all requests are justified.

You can also use approval policies to enable a project administrator to review all the changes that are done as part of orchestration to a critical app instance.

Approval Policy Creation and Management

- The Approvals feature is disabled by default. You must enable the feature from the Settings page before creating your approval policies. For more information, see [Enabling Approvals](#) on page 45.
- You must enable the policy engine to create and manage approval policies.
- Each approval policy is a defined set of conditions that you configure for specific entities in Self-Service. For more information, see [Conditions in Approval Policies](#) on page 246.
- A policy can have multiple conditions. An approval request is generated when an event meets all the conditions defined in the policy.
- As a Prism Central Admin or Project Admin, you can create approval policies for runbook execution, app launch, and app day-2 operations (system-defined or user-defined actions).
- A policy can have more than one set of approvers, and the approvals are done sequentially during the policy enforcement. For example, if the policy has Set 1 and Set 2 approvers, then during the policy enforcement, Set 1 approvers have to approve the request before Set 2.
- The default expiration time of an approval request is 30 days. You can modify the expiration time using the associated API. For information on the API, see [API Reference](#).
- The approver must be an existing user of Prism Central.
- You can enable a policy to enforce the policy on an event that matches the entity, action, and conditions of the policy or disable the policy to skip policy enforcement.
- You can enable or disable approvals from the Settings page to enforce all enabled approval policies in Self-Service or disable all approval policy enforcement.
- When the policy engine VM does not respond, the runbook execution, app launch, or app day 2 operations that match the approval policy conditions fail to process completely. To process those events, you must disable policy enforcement. For more information, see [Disabling Policy Checks](#) in the *Prism Central Admin Center Guide*.
- You can clone an existing policy and edit its information to quickly create a new policy.
- You can delete an approval policy.

- The approval feature is currently not supported for VMware update config, Azure update config, or AWS update config.

Approval Process

- An approver receives an email notification when an approval action is required for a request.
 - For email notifications, the SMTP server must be configured in Prism Central. For more information, see [Configuring an SMTP Server](#) in the *Prism Central Admin Center Guide*.
 - Notifications are sent based on the value of the **E-mail** field in your Active Directory. To receive approval notifications, ensure that the value is specified in the **E-mail** field of the Active Directory.

Figure 74: Active Directory Configuration

- The email notification contains the approval request name, and each approval request name is appended with the approval request creation time stamp for better identification.
- As an approver, you can view a list of all pending approval policies and can approve or reject the request with a reason. When you approve a request, the event moves to the next task. When you reject a request, the requester is notified about the rejection of the request.
- As an approver, you cannot make any updates to the original approval request.
- If an Active Directory group is added as an approver, any user from the group can approve the request to move the event to the next task.
- A Prism Central admin cannot override and approve pending requests. All pending requests have to be approved by the approvers assigned in the enforced policy.
- The **Audit** tab of an app displays the confirmation of the enforced policy. The Policy Execute - Approval task is added on the **Audit** tab, and the task remains in the POLICY_EXEC status until the request is approved or rejected.

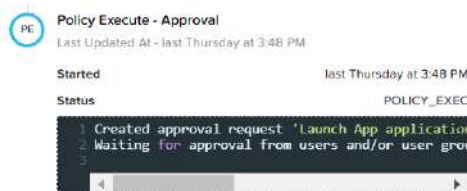


Figure 75: Policy Execute - Approval

Note: Before you upgrade Self-Service (through LCM or Prism Central upgrade), it is recommended that you do not leave any executions (blueprint launch, runbook execute, or action run) in the pending approval state.

Approvals Page

- The **Policy Configurations** tab provides the option to create an approval policy and lists all the approval policies you created as an admin for management.
- The **Approval Requests** tab displays all requests that you need to approve on the **Pending on me** tab and all requests generated on the **All requests** tab.
- The **My Requests** tab displays all the requests that you created. The **Pending** tab displays all pending requests, and the **Reviewed** tab displays all requests that the approvers reviewed. When you click a request on the **Pending** tab, you can view the approvers who are required to approve your request. If you are an admin, you can also view the details of the enforced policy.

Creating an Approval Policy

As a Prism Central Admin or Project Admin, you can create approval policies for runbook executions, app launch, and app day-2 operations (system-defined or user-defined actions).

About this task

Each approval policy is a defined set of conditions that you apply to specific entities in Self-Service. An approval request is generated when an associated event meets all the conditions defined in the policy.

Before you begin

Ensure that you have enabled the policy engine on the **Settings** page. For information on enabling the policy engine, see [Enabling Policy Engine](#) in the *Prism Central Admin Center Guide*.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Approvals** tab, click **+ Create Approval Policy** to create a policy.
5. On the **Basic Information** tab, provide the basic information such as the name, project, and the entity with its associated action. To do that:
 - a. In the **Name** field, provide a name for the approval policy.
 - b. In the **Description** field, provide a description for the policy. This step is optional.
 - c. From the **Select the project this policy is applicable to** dropdown menu, select a project with which you want to associate the approval policy.
 - d. From the **Entity Type** dropdown menu, select the entity to which you want to apply the approval policy.
You can select **Runbook** or **Application**.
 - e. From the **Action** dropdown menu, select the action during which the approval policy must be enforced.
The options in the **Action** dropdown menu appear based on the entity you selected.
 - f. Click **Next**.

6. On the **Set Conditions** tab, specify the attribute, its associated operator, and the value for the approval policy. To do that:
 - a. From the **Attribute** dropdown menu, search the attribute for the policy enforcement.

The options in the **Attribute** dropdown menu change based on the entity and the associated action you selected on the **Basic Information** tab.

The **Attribute** dropdown menu does not display all available options when you click the dropdown arrow. You must type the attribute name in the field to view and select the desired attribute. See [Conditions in Approval Policies](#) on page 246 for a list of available attributes.
 - b. From the **Operator** dropdown menu, select the operator for the policy attribute.

The options in the **Operator** dropdown menu change based on the attribute you selected as the condition.
 - c. In the **Value** field, specify the value for the attribute-operator condition.

With some attribute-operator combinations, an information icon appears that displays the supported units or values for the **Value** field. You can use the information to specify the appropriate values in the field.

For system actions, you must specify the name in the action_<system action> format. For example, for the Restart action, you must specify the value as action_restart. For the list of supported system action names for approval policies, see [Conditions in Approval Policies](#) on page 246.

For Azure locations, you must specify the Azure location name instead of the Azure location displayName. For example, instead of using Central US (the Azure location displayName) in the **Value** field, use centralus (the Azure location name).
 - d. Click **Done** next to the condition name.
 - e. To add another condition to the policy, click **+ Add Condition** and then specify the attribute, operator, and value.

You can also click the **Copy** icon next to the condition name of an existing condition to quickly create a new condition and edit its details.

You can add multiple conditions in the policy. An approval request is generated when an event meets all the conditions defined in a policy. To view the list of conditions, see [Conditions in Approval Policies](#) on page 246.
7. Click **Next**.

8. On the **Select Approvers** tab, specify the set name, approver rule, and approvers for the policy. To do that:
 - a. In the **Set Name** field, specify the name of the policy set.
 - b. Under approval rule, select one of the following:
 - » **Any one can approve**: Select this option if you want any approvers selected for the set to approve the action.
 - » **All need to approve**: Select this option if you want all approvers in the set to approve the action.
 - c. From the **Approvers** dropdown menu, select the approvers you want to include in the set. You can select and add multiple approvers in a set.
 - d. Click **Done** next to the set name.
 - e. To create another set, click **+ Add Approver Set** and specify the set name, approver rule, and approvers.

You can also click the **Copy** icon next to the set name of an existing set to quickly create a new set and edit its details.

You can add multiple sets of approvers in the policy. The approver sets are applied sequentially during the policy enforcement. For example, if you have configured Set 1 and Set 2 approvers in your policy, then during policy enforcement, Set 1 approvers have to approve the request before Set 2.
9. Click **Save**.
10. In the Policy Saved confirmation window, select **Yes, enable this policy** to enable the policy. You can click **No, keep it disabled** if you want to create the policy in the disabled state.

Conditions in Approval Policies

You can configure approval policies for specific events with different set of conditions. For example, to configure an approval policy for a marketplace item, you can use the following values:

- **Entity Type**: Application
- **Action**: Launch
- **Attribute**: Blueprint Name
- **Operator**: Contains
- **Value**: <Name of the Marketplace Item for which you want to create an approval policy>

The following table lists the different conditions that you can define for different events in approval policies. To search for a provider-specific attribute, type the provider name in the **Attribute** field.

Table 39: Conditions in Approval Policies

Entity Type and Action	Provider	Attribute	Operator	Supported Values
Entity Type: Runbook Action: Execute	All	Runbook Name	Equals, Contains, Like	Generic
		Task Name	Equals, Contains, Like	Generic
		Endpoint Name	Equals, Contains, Like	Generic
Entity Type: Application Action: Launch	All	Substrate Type	Equals, Contains, Like	AWS_VM AHV_VM VMWARE_VM GCP_VM AZURE_VM EXISTING_VM
		Blueprint Name	Equals, Contains, Like	Generic
		Application Name	Equals, Contains, Like	Generic
		Application Profile Name	Equals, Contains, Like	Generic
		Estimated Application Profile Cost	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Estimated cost of app when launched in (\$)USD per month
		Account Name	Equals, Contains, Like	Generic
		VM Name	Equals, Contains, Like	Generic
		Service Name	Equals, Contains, Like	Generic
		App Replicas Count	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer Value
		OS Type	Equals, Contains, Like	Linux Windows

Entity Type and Action	Provider	Attribute	Operator	Supported Values
	Azure Specific Attributes	Azure Tag	Equals, Contains, Like	Key and value pair in the key:value format
		Azure Location	Equals, Contains, Like	User should specify Azure location name and not the Azure location displayName. For example, "Central US" is the location displayName and "centralus" is its location name. For more details on location displayName and location name, refer to the Azure documentation.
		Azure Instance Name	Equals, Contains, Like	Generic
		Azure Resource Group	Equals, Contains, Like	Resource Group Name
		Azure Availability Zone	Equals, Contains, Like	Availability Zone Name
		Azure Availability Set	Equals, Contains, Like	Availability Set Name
		Azure Hardware Profile	Equals, Contains, Like	Hardware Profile Name
		Azure Data Disk Name	Equals, Contains, Like	Generic
		Azure Data Disk Type	Equals, Contains, Like	Allowed values are Standard_LRS, StandardSSD_LRS, and Premium_LRS.
		Azure Data Disk Size	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer Value
		Azure Network Profile Subnet	Equals, Contains, Like	Subnet Name
		Azure Network Profile NIC Name	Equals, Contains, Like	Generic
		Azure Network Profile Virtual Network	Equals, Contains, Like	Virtual Network Name
		Azure Network Profile Network Security Group	Equals, Contains, Like	Security Group Name

Entity Type and Action	Provider	Attribute	Operator	Supported Values
	VMware Specific Attributes	VMware Instance Name	Equals, Contains, Like	VM Name
		VMware Datastore Cluster	Equals, Contains, Like	Datastore Cluster Name
		VMware Datastore	Equals, Contains, Like	Datastore Name
		VMware Cluster	Equals, Contains, Like	Cluster Name
		VMware Host	Equals, Contains, Like	Host ipv4 address
		VMware Sockets	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer Value
		VMware Cores Per Socket	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer Value
		VMware Memory	Equals, Contains, Like	Size of memory in GiB
		VMware Adapter Type	Equals, Contains, Like	Currently allowed values are: <ul style="list-style-type: none"> • e1000 • e1000e • pcnet32 • vmxnet • vmxnet2 • vmxnet3
		VMware Network	Equals, Contains, Like	Network Name
		VMware Disk Type	Equals, Contains, Like	Allowed values are: <ul style="list-style-type: none"> • Disk • CD-ROM
		VMware Tag	Equals, Contains, Like	Key and value pair in the key:value format

Entity Type and Action	Provider	Attribute	Operator	Supported Values
		VMware Disk Size	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of disk in GiB
		VMware Template Name	Equals, Contains, Like	Template Name
	AHV Specific Attributes	AHV vCPU	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer
		AHV Cores Per vCPU	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer
		AHV Memory	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of memory in GiB
		AHV Category	Equals, Contains, Like	Key and value pair in the key:value format
		AHV VPC Name	Equals, Contains, Like	VPC Name
		AHV vLAN Name	Equals, Contains, Like	vLAN Name
		AHV Disk Type	Equals, Contains, Like	Allowed values are: <ul style="list-style-type: none"> Disk CD-ROM
		AHV Disk Image Name	Equals, Contains, Like	Image Name
		AHV Disk Size	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of disk in GiB
		AHV Boot Configuration Type	Equals, Contains, Like	Allowed values are: <ul style="list-style-type: none"> LEGACY UEFI

Entity Type and Action	Provider	Attribute	Operator	Supported Values
	AWS Specific Attributes	AWS Instance Type	Equals, Contains, Like	Instance type. For example, t2.small
		AWS Region	Equals, Contains, Like	Region. For example, us-east-1
		AWS Tag	Equals, Contains, Like	Key and value pair in the key:value format
		AWS Root Volume Type	Equals, Contains, Like	Allowed values are GP2, IO1, ST1, SC1, STANDARD. Mappings: <ul style="list-style-type: none"> GP2: General Purpose SSD IO1: Provisioned IOPS SSD ST1: Throughput Optimized HDD SC1: Cold HDD STANDARD: EBS Magnetic HDD
		AWS Data Volume Type	Equals, Contains, Like	Allowed values are GP2, IO1, ST1, SC1, STANDARD. Mappings: <ul style="list-style-type: none"> GP2: General Purpose SSD IO1: Provisioned IOPS SSD ST1: Throughput Optimized HDD SC1: Cold HDD STANDARD: EBS Magnetic HDD
		AWS Root Disk Size	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of disk in GiB
		AWS Data Disk Size	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of disk in GiB

Entity Type and Action	Provider	Attribute	Operator	Supported Values
		AWS IAM Role	Equals, Contains, Like	Role name. For example, aws-controltower-AdministratorExecutionRole
		AWS VPC ID	Equals, Contains, Like	VPC ID. For example, vpc-ffd543a1
		AWS Security Group ID	Equals, Contains, Like	Security Group ID. For example,, sg-127ead61
		AWS Subnet ID	Equals, Contains, Like	Subnet ID. For example, subnet-c512a5ea
		AWS Machine Image ID	Equals, Contains, Like	AMI ID. For example, ami-c1632ad5
	GCP Specific Attributes	GCP Instance Name	Equals, Contains, Like	VM Name
		GCP Machine Type	Equals, Contains, Like	Machine Type. For example, f1-micro
		GCP Zone	Equals, Contains, Like	Zone. For example, us-central1-c
		GCP Boot Disk Storage Type	Equals, Contains, Like	Storage Type. For example, pd-standard
		GCP Boot Disk Source Image	Equals, Contains, Like	Source Image Name. For example, centos-7
		GCP Labels	Equals, Contains, Like	Key and value pair in the key:value format
Entity Type: Application Action: Day 2 Operation	All	Application Name	Equals, Contains, Like	Generic
		Application Profile Cost	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Estimated cost of app when launched in (\$)USD per month
		App Replicas Count	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer

Entity Type and Action	Provider	Attribute	Operator	Supported Values
		Action Name	Equals, Contains, Like	System actions: <ul style="list-style-type: none"> action_start action_restart action_stop action_delete action_soft_delete action_snapshot_create action_restore action_update Custom or user-defined action: Generic name
	AHV Specific Attributes (for Update Config Only)	AHV vCPU	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer
		AHV Cores Per vCPU	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Integer
		AHV Memory	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of Memory in GiB
		AHV Category	Equals, Contains, Like	Key and value pair in the key:value format
		AHV vLAN Name	Equals, Contains, Like	vLAN Name
		AHV VPC Name	Equals, Contains, Like	VPC Name
		AHV Device Type	Equals, Contains, Like	Allowed values are: <ul style="list-style-type: none"> Disk CD-ROM

Entity Type and Action	Provider	Attribute	Operator	Supported Values
		AHV Disk Size	Equals, Less than, Greater than, Greater than or Equals, Less than or Equals	Size of disk in GiB
	AHV (for Snapshots)	AHV Snapshot Location	Equals, Contains, Like	Allowed values are: <ul style="list-style-type: none"> • LOCAL • REMOTE
		AHV Snapshot Replica	Equals, Contains, Like	Allowed values are: <ul style="list-style-type: none"> • ONE • ALL
		AHV Snapshot Name	Equals, Contains, Like	Snapshot Name

Day 2 operations are combination of multiple actions. Ensure that you use the supported attributes for different day 2 operations to enforce the policy appropriately. For example, when you configure a policy with scale in or scale out task, the supported attributes can be App Replicas Count and Application Profile Cost.

The following table provides the day 2 operation with the supported attributes.

Table 40: List of Supported Attributes for Day 2 Operations

Day 2 Operation	Supported Attributes
AHV Update Config	Estimated Application Profile Cost, AHV vCPU, AHV Cores Per vCPU, AHV Memory, AHV Category, AHV VPC Name, AHV vLAN Name, AHV Disk Size, AHV Device Type, and Substrate Type
Scale-in or Scale-out task	App Replicas Count and Application Profile Cost
AHV Snapshot Config	AHV Snapshot Name, AHV Snapshot Replica, and AHV Snapshot Location
Supported Attributes for All Day 2 Operations	Application Name and Action Name

For system actions, you must specify the name in the `action_<system action>` format. The following table lists the system action names supported for approval policies.

Note: Approval policy does not apply to a day 2 operation for an Action and Credentials update.

Table 41: Supported System Action Names

System Action	Names
Start	action_start
Restart	action_restart
Stop	action_stop
Delete	action_delete
Soft Delete	action_soft_delete
Snapshot Create	action_snapshot_create
Restore	action_restore
Update	action_update

Cloning an Approval Policy

To quickly create a new policy, you can clone an existing policy and edit its basic information, conditions, and approvers.

About this task

You cannot clone an approval policy that is in the Draft state.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Approvals** tab, click the vertical ellipsis next to the policy that you want to clone and then click **Clone**.
5. In the Clone Approval Policy window, provide a name for your new policy in the **Approval policy name** field, and then click **Clone**.
6. On the **Basic Information**, **Set Conditions**, and **Select Approvers** tab, edit the fields that you need to change in your new policy. For information on the fields, see [Creating an Approval Policy](#) on page 244.
7. Click **Save** to save the approval policy.
You can also clone a policy from the policy details page.

Enabling or Disabling an Approval Policy

You can enable a policy to enforce the policy on an event that matches the entity, action, and conditions of the policy or disable the policy to skip policy enforcement.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.

4. To enable or disable a policy, do one of the following on the **Approvals** tab.
 - » To enable a disabled policy, click the vertical ellipsis next to the policy that you want to enable and then click **Enable**.
 - » To disable an enabled policy, click the vertical ellipsis next to the policy that you want to disable and then click **Disable**.

You can also enable or disable a policy from the policy details page.

Deleting an Approval Policy

As a Prism Central Administrator or Project Administrator, you can delete an approval policy if the policy is no longer required for the event.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Approvals** tab, click the vertical ellipsis next to the policy that you want to delete and then click **Delete**.
5. In the Confirm Delete window, click **Delete**.

Viewing an Approval Policy Details

After you have created a policy, you can view the details of the policy on the policy details page.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Approvals** tab, click the policy that you want to view.

The policy details page has the following tabs:

 - **Basic Information:** Use this tab to view the associated project, event, and the details related to the policy creation and updates. You can also use the **Enable Policy** or **Disable Policy** option on this tab to enable or disable the policy.
 - **Conditions:** Use this tab to view all the conditions that are associated with the policy.
 - **Approvers:** Use this tab to view the approvers sets that are associated with the policy.
 - **Execution History:** Use this tab to view the execution history of policies.
5. To clone the policy, click **Clone Policy**. For more information, see [Cloning an Approval Policy](#) on page 255.
6. To edit the policy, click **Edit** and update the required fields on the **Basic Information**, **Set Conditions**, and **Select Approvers** tab. For information on the fields, see [Creating an Approval Policy](#) on page 244.
7. To delete the policy, click **Delete Policy**.

Approving or Rejecting an Approval Request

As an approver, you can view a list of all pending approval policies on the **Approval Requests** tab and can either approve or reject the request with a reason.

About this task

When you approve a request, the event moves to the next task. When you reject a request, the requester is notified about the rejection of the request. If you are the requester, you can view your pending requests and the status of your reviewed request on the **My Requests** tab.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Policies** in the navigation bar.
4. On the **Approvals** tab, click the **Approval Requests** tab in the left pane.

5. On the **Pending on me** tab, do the following:
 - a. Click the request that is pending for approval.
 - b. View the Basic Information and Condition Details of the applicable policy.

← Search App (100) Pending

View Policy Go to Application

Basic Information

Project	default
Initiated	Apr 11th, 2022 at 12:28 PM
Expires	May 11th, 2022 at 12:28 PM
Requested by	admin

Condition Details

Add Comment (Optional)

Enter a comment about why you are approving / rejecting the request

Reject Approve

Approver Sets

1. Set 1 ANY

admin (You)	Pending
-------------	---------

Figure 76: Pending Request for Approval

If you are an admin, you can click **Go to Application** to go to the Overview tab of the app and view the details. You can also click the **View Policy** tab to view the enforced policy.

- c. (Optional) In the **Add Comment** field, provide a reason for the approval or rejection of the request.
- d. To approve the request, click **Approve**.
- e. To reject the request, click **Reject**.
- f. Click **Yes** to confirm approval or rejection.

You can also view the details of the request such as the requester, date of initiation, and conditions on the **Pending on me** tab.

LIBRARY IN SELF-SERVICE

The topics in this section cover the configuration and usage of task libraries and variables in Self-Service.

Library Overview

Library allows you to save user-defined tasks (scripts) and variables that you can use persistently for other app blueprints. You do not have to define the same tasks and variables for each blueprint.

You can also share tasks and variables listed as part of library across different projects. You can also customise an existing task or variable.

The Library tab lists all the published user-defined tasks and the created variable types to be used across multiple blueprints.

Note:

- To list a task in the Library, you must publish the task by using **Publish to Library** functionality under service package while configuring your blueprints.
- To view the list of variables, you must create and save the variables in the Library. For more information, see [Variable Types Overview](#) on page 259.

Variable Types Overview

You create custom variable types for added flexibility and utility. Beyond just string and integer data types, you can create more data types such as Date/Time, list, and multi-line string. You can define list values as a static list of values or can attach a script (eScript or HTTP task) to retrieve the values dynamically at runtime.

While creating a custom variable type, you associate a project to the variable type. You can also share the variable type with multiple other projects using the "Share" option on the same page.

Creating Variable Types

Create variable types so that you can use the variables during blueprint creation. You can also share the created variable types across multiple projects.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Library** in the navigation bar.

4. Click the **Variable Types** tab.

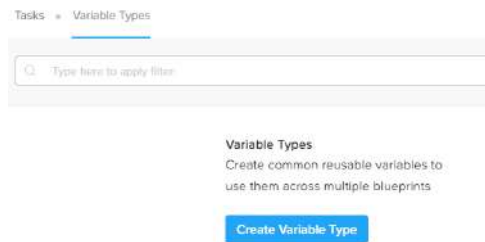


Figure 77: Variable Type

5. Click **Create Variable Type** if you are creating the first variable or **+ Add Variable Types** if you are adding a new variable to the list of variables.
The **Create Variable Type** window appears.

6. From the **Projects** dropdown menu, select the project and click **Done**.

Note: You associate a project when you create a custom variable type. You can also share the variable type with other projects using the **Share** option. Members of the same project can use the variable types while creating a blueprint.

7. In the **Name** field, type a name for the variable type.
8. In the **Description** field, type a brief description about the variable type.
9. From the **Data Type** dropdown menu, select the base type for the variables.
The base type defines the type of variable you use while configuring a blueprint. You can select one of the following data types.
 - String
 - Integer
 - Multi-line string
 - Date
 - Time
 - Date Time
10. Select one of the following input type.
 - Use **Simple** to add a default value.
 - Use **Predefined** to assign static values.
 - Use **eScript** to attach a script that is run to retrieve values dynamically at runtime. Script can return single or multiple values depending on the selected base data type.
 - Use **HTTP** to retrieve values dynamically from the defined HTTP end point. Result is processed and assigned to the variable based on the selected base data type.
11. If you have selected **Simple**, then type the value for the variable in the **Value** field.

12. If you have selected **Predefined**, then type the value for the variable in the **Option** field. To add multiple values for the variable, do the following.

- a. Click **+ Add Option**.
- b. In the **Option** field, type the value.
- c. To make any value as default, select **Default** next to the value.

13. If you have selected **eScript**, do the following:

- a. Select the Python version from the version drop-down menu. By default, Python 3 is selected as the version for you to enter the script.

Note: Based on the decision that the Python Software Foundation (PSF) took to discontinue support for Python 2, Nutanix has decided to end support for Python 2 on June 30, 2024. Nutanix recommends that you format all your new eScripts in Python 3 and update any existing eScripts from Python 2 to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

- b. Enter the script or use the upload icon to upload the script in the **Script** field.

You can click **Publish** to publish the script to the library.

Note:

- You cannot add macros to eScripts.
- If you have selected Multiple Input (Array) checkbox with input type as eScript, then ensure that the script returns a list of values separated by comma. For example, CentOS, Ubuntu, Windows.

14. If you have selected **HTTP**, then configure the following fields.

- a. In the **Request URL** field, specify the URL of the server that you want to run the methods on.
 - b. In the **Request Method** dropdown menu, select a request method. The available options are GET, PUT, POST, and DELETE.
 - c. In the **Request Body** field, enter or upload the request.
 - d. In the **Content Type** dropdown menu, select a type of the output format. The available options are XML , JSON, and HTML.
 - e. In the **Connection Timeout (sec)** field, type the timeout interval in seconds.
 - f. Select the authentication type.
 - » If you select **Basic**, then specify the **User name** and **Password**.
 - » If you select **Basic (With Credentials)**, then you can set the credentials in the blueprint after copying the task.
- By default, **Authentication** is set to **None**. This step is optional.
- g. To verify the URL of the HTTP endpoint with a TLS certificate, select the **Verify TLS Certificate** checkbox. This step is optional.
 - h. To use a proxy server that you configured in Prism Central, select the **Use PC Proxy configuration** checkbox. This step is optional.
 - i. In the **Retry Count** field, type the number of attempts the system must perform to create a task after each failure. By default, the retry count is one, which indicates that the task creation procedure stops after the first attempt.
 - j. In the **Retry Interval** field, type the time interval in seconds for each retry if the task fails. By default, the **Retry Interval** value is set to one second.
 - k. Under **Headers**, enter the HTTP header key and value in the **Key** and **Value** fields.
 - l. To publish the HTTP header key and value pair as secret, select the **Secrets** checkbox.
 - m. Under **Expected Response Options**, type the **Response Code** for the **Response Status** you select. You can select **Success** or **Failure** as the response status for the task.

15. To check the Regex, do the following.

- a. Select the **Validate with Regular Expression** checkbox.
- b. Click **Test Regex**.
- c. Provide the value for the Regex in the **Value** field.

Note: You can enter Regex values in PCRE format. For more details, see from <http://pcre.org/>.

- d. To test the expression, click **Test Regex**.

16. Click **Save**.

The Variable is saved to the Library.

17. To share the variable type with other projects, do the following.
 - a. Click **Share**.
The **Share Variable Type** screen appears.
 - b. From the **Select projects to share with** dropdown menu, add the projects with which you want to share the saved variable.
 - c. Click **Done**.

What to do next

Use this variable type while define variables in a blueprint. For more information, see [Self-Service Blueprints Overview](#) on page 87.

Task Library Overview

You can create tasks while configuring a blueprint and publish these tasks to the library. Self-Service allows you to import these published tasks while configuring other blueprints across multiple projects.

Adding a Task to a Project

Add a task to one or more projects so that you can use the task while configuring blueprints for the selected projects.

Procedure

1. Log in to your Prism Central instance.
2. From the Application Switcher, select **Self-Service**.
3. In the navigation bar, click **Library**.
The **Tasks** tab displays the list of all published tasks.
4. Select the task that you need to assign to projects.
The task inspector panel appears.
5. From the **Project Shared With** dropdown menu, select one or more projects to which you need to assign the task.
6. Click **Save**.
The task is added to the selected projects.

Deleting a Task from the Task Library

Delete unwanted tasks from the Library. The deleted tasks can no longer be used in any project while configuring a blueprint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Library** in the navigation bar.
The **Tasks** tab displays the list of all published tasks.
4. Select the task that you want to delete.
The task inspector panel appears.
5. Click **Delete**.

6. In the confirmation window, click **Delete**.
The task is deleted from the Library.

RUNBOOKS IN SELF-SERVICE

The topics in this section cover the configuration and usage of runbooks in Self-Service.

Runbooks Overview

A runbook is a framework to automate routine tasks and procedures that span across multiple apps without the involvement of a blueprint or an app.

A runbook is a collection of tasks that you can define to run sequentially at different endpoints. For more information on endpoints, see [Endpoints Overview](#) on page 386.

You can define the following types of tasks in a runbook.

Table 42: Tasks in a Runbook

Task	Description
Execute	To run Shell, PowerShell, and eScript (custom python) scripts.
Set Variable	To run a script and create variables.
Delay	To set a delay interval between two tasks or actions.
HTTP	To perform REST calls to an HTTP endpoint.
While Loop	To iterate over multiple tasks until the defined condition is met.
Decision	To define different flows or paths based on the exit condition.
VM Power On	To power on the VMs that are present in the VM endpoint type.
VM Power Off	To power off the VMs present in the VM endpoint type.
VM Restart	To restart the VMs present in the VM endpoint type.
Database Operation	To configure database management related tasks.

For more information on creating a runbook, see [Creating a Runbook](#) on page 266.

Runbook Sharing across Projects

To share an active runbook across different projects, you can submit the runbook to be published as a Marketplace item. When the runbook is available at the marketplace, members from different projects to which the runbook is assigned can view and execute it.

When you submit a runbook for publishing, your administrator approves and publishes the runbook at the Marketplace. While publishing, your administrator selects the projects that can view and execute the runbook. You can publish runbooks with or without endpoints and with or without secret values (credential passwords or keys and secret variables). For more information, see [Submitting a Runbook for Publishing](#) on page 283.

You can select endpoints with virtual machines as the target type to execute power operation tasks such as power off, power on, or restart. Executing these tasks on Virtual machines is particularly helpful in cases where you need to run a set of scripts on multiple VMs and then restart the VMs. For example, when you want to upgrade a software on your VMs. For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386.

You cannot modify the runbook after it is published. You can either execute the runbook or clone the runbook within your project from the marketplace.

Creating a Runbook

A runbook is a collection of tasks that you can define to run sequentially at different endpoints.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Runbooks** in the navigation bar.
4. Click **Create Runbook**.

5. Configure the following in the Create Runbook window.

The screenshot shows a 'Create Runbook' dialog box. It has a title bar with the text 'Create Runbook' and a close button 'X'. The dialog contains the following fields:

- Name:** A text input field containing 'Runbook1'.
- Description:** A text area with a placeholder 'Description' and a 'View' link to its right.
- Project:** A dropdown menu with 'Self-Service' selected.
- Default Endpoint (Optional):** A dropdown menu with 'Endpoint' selected, accompanied by a help icon (i).

At the bottom right of the dialog are two buttons: 'Cancel' and 'Proceed'.

Figure 78: Create Runbook

- In the **Name** field, type a name for the runbook.
- In the **Description** field, type a brief description about the runbook. This step is optional.
- From the **Project** dropdown menu, select a project to which you want to add the runbook.
- From the **Default Endpoint** dropdown menu, select a default endpoint. This step is optional.

Self-Service uses the default endpoint only when you do not configure any endpoint at the task level.

6. Click **Proceed**.

7. On the **Editor** tab, click **+Add Task** and do the following.

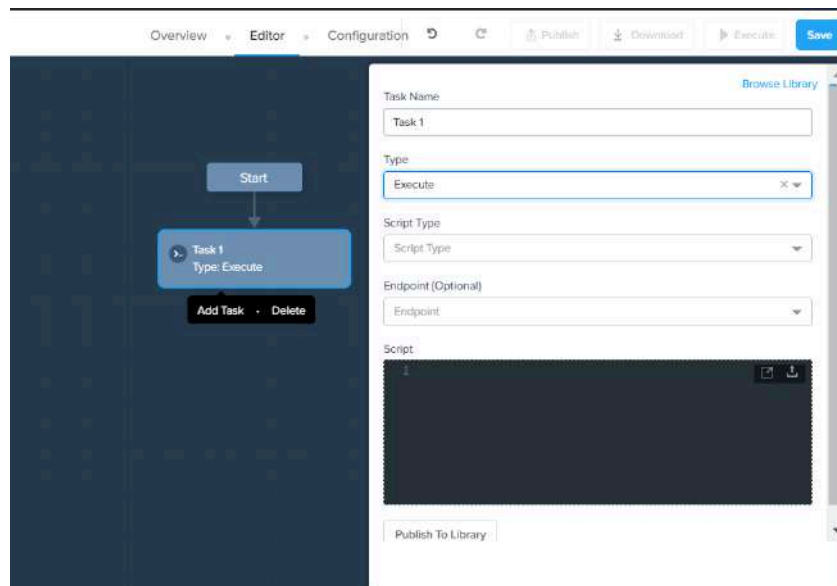


Figure 79: Runbook Editor

- a. In the **Task Name** field, type a name for the task.
- b. From the **Type** dropdown menu, select the task type.
 - » Select the **Execute** task to run Shell, PowerShell, and eScript (custom python) scripts or **Set Variable** task to run a script and create variables. For more information on how to configure the Execute or Set Variable task type, see [Creating a Runbook with an Execute or Set Variable Task](#) on page 270.
 - » Select the **Delay** task to set a delay interval between two tasks or actions. For more information on how to configure the Delay task type, see [Creating a Runbook with a Delay Task](#) on page 272.
 - » Select the **HTTP** task to perform REST calls to an HTTP endpoint. For more information on how to configure an HTTP task type, see [Creating a Runbook with an HTTP Task](#) on page 272.
 - » Select the **While Loop** task to iterate over multiple tasks until the defined condition is met. For more information on how to configure the While Loop task type, see [Creating a Runbook with a While Loop Task](#) on page 273.
 - » Select the **Decision** task to define different flows or paths based on the exit conditions.

The task is further subdivided into **True** and **False** condition. You must repeat the steps to add the tasks and configure the task type.
 - » Select the **VM Power Off**, **VM Power On**, or **VM Restart** task to power off, power on, or restart the VMs that are present in the VM endpoint type. You must select the target VM endpoint for these task types.

8. To add a credential, do the following on the **Configuration** tab.

- a. Click **Add/Edit Credentials**.
- b. Click **+ Add Credential**.
- c. In the **Name** field, type a name for the credential.
- d. Under the **Type** section, select the type of credential that you want to add.
 - » **Static**: Credentials store keys and passwords in the credential objects that are contained in the blueprints.
 - » **Dynamic**: Credentials fetch keys and passwords from an external credential store that you integrate with Self-Service as the credential provider.
- e. In the **Username** field, type the user name.
For dynamic credentials, specify the @@(username)@@ that you defined while configuring the credential provider.

Note: A dynamic credential provider definition requires username and secret. The secret variable is defined by default when you configure your credential provider. However, you must configure a runbook in the dynamic credential provider definition for the username variable before you use the variable in different entities.

- f. Select either **Password** or **SSH Private Key** as the secret type.
- g. Do one of the following to configure the secret type.
 - » If you have selected **Static** as the credential type and **Password** as the secret type, then type the password in the **Password** field.
 - » If you have selected **Static** as the credential type and **SSH Private Key** as the secret type, then enter or upload the key in the **SSH Private Key** field.
 - » If you have selected **Dynamic** as the credential type and **Password** or **SSH Private Key** as the secret type, then select a credential provider in the **Provider** field. After you select the provider, verify or edit the attributes defined for the credential provider.

If the private key is password protected, click **+Add Passphrase** to provide the passphrase. For dynamic credentials, you must configure a runbook in the dynamic credential provider definition for the passphrase variable and then use the @@{passphrase}@@ variable.

The type of SSH key supported is RSA. For information on how to generate a private key, see [Generating SSH Key on a Linux VM](#) on page 444 or [Generating SSH Key on a Windows VM](#) on page 445.

- h. Click **Done**.

Note: The credential that you add on the **Configuration** tab overrides the credential you added in the endpoint.

9. To add a variable, do the following on the **Configuration** tab. This step is optional.

- a. Click **Add/Edit Variable**.
- b. If you want to use an existing variable, select the variable that you want to use for the runbook.
- c. If you want to create a new variable, click **Add Variable**. For more information on how to create variables, see [Creating Variable Types](#) on page 259.

10. To add a default endpoint, select the endpoint from the **Default Endpoint** dropdown menu. This step is optional.

Note: The endpoint you select on the **Configuration** tab supersedes the endpoint you add on the **Editor** tab.

11. To add endpoint information, select the endpoint from the **Endpoint** dropdown menu and enter a description for the endpoint in the **Description** field. This step is optional.
12. Click **Save**

What to do next

- You can execute the runbook. For more information, see [Executing a Runbook](#) on page 285.
- You can submit the runbook for approval and publishing. For more information, see [Submitting a Runbook for Publishing](#) on page 283.

Creating a Runbook with an Execute or Set Variable Task

Create a runbook with the Execute task to run Shell, PowerShell, eScript (custom python), or Python scripts. Create a runbook with the Set Variable task to run a script and create variables.

Before you begin

Ensure that you have created a runbook and have added a task for the runbook. For more information, see [Creating a Runbook](#) on page 266.

Procedure

1. On the runbook **Editor** tab, click **+Add Task**.
2. In the **Task Name** field, type a name for the task.
3. From the **Type** dropdown menu, select the **Execute** or **Set Variable** task.
4. From the **Script Type** dropdown menu, select **Shell**, **Powershell**, **eScript** or **Python**.
You can access the available list of macros by using @@{ in the **Script** field for any script type.
5. If you have selected **EScript**, do the following:
 - a. From the **Endpoint (Optional)** dropdown menu, select an endpoint on which you want to run the task. This step is optional.
You can also select **Add New Endpoint** from the list and create an endpoint. For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386.
If you do not select an endpoint, then Self-Service uses the default endpoint that you defined while creating the runbook.
 - b. Select the tunnel that you can use to get access to the endpoint within the VPC in the **Select tunnel to connect with (optional)** dropdown menu. This step is optional.
The **Select tunnel to connect with (optional)** dropdown menu shows only those tunnels that are allowed in the project you selected for the endpoint.
 - c. Select the Python version from the version drop-down menu. By default, Python 3 is selected as the version for you to enter the script.

Note: Based on the decision that the Python Software Foundation (PSF) took to discontinue support for Python 2, Nutanix has decided to end support for Python 2 on June 30, 2024. Nutanix recommends that you format all your new eScripts in Python 3 and update any existing eScripts

from Python 2 to Python 3 as early as possible. For more information, see [Python 2 Deprecation](#) on page 401.

d. Enter the script or use the upload icon to upload the script in the **Script** field.

6. If you have selected **Shell** or **Powershell**, do the following:

a. From the **Endpoint (Optional)** dropdown menu, select an endpoint on which you want to run the task. This step is optional.

You can also select **Add New Endpoint** from the list and create an endpoint. For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386.

If you do not select an endpoint, then Self-Service uses the default endpoint that you defined while creating the runbook.

b. In the **Credential (Optional)** dropdown menu, select an existing credential or click **Add New Credential** to add a credential.

For more information on adding a credential, see [Adding Credentials](#) on page 173.

7. If you have selected the script type as **Python**, do the following:

Use Python script type when you want your scripts to run in a predefined Python environment. This option enables you to run Python scripts in your own customized endpoint wherein you can add custom modules, any version of Python, and their corresponding binaries.

Note: The Python script type is not supported in the runbooks that are used in Dynamic credentials.

a. In the **Endpoint** dropdown menu, select the endpoint that you created to run your Python scripts or click **Add New Endpoint** to create an endpoint.

For more information on creating an endpoint, see [Creating an Endpoint](#) on page 386. For information on how to configure your endpoint to run Self-Service Python scripts, see [Configuring a Remote Machine to Run Self-Service Python Scripts](#) on page 389.

Note:

- Providing an endpoint is mandatory when you select the **Python** script type.
- The Python script type does not support tunnels and endpoints that are created within a VPC.
- The Python script type is supported only with Linux endpoints.

b. In the **Credential** dropdown menu, select an existing credential or click **Add New Credential** to add a credential.

For more information on adding a credential, see [Adding Credentials](#) on page 173.

8. In the **Script** panel, enter or upload the script that you want to run.

9. To test the script in the Self-Service playground, click **Test script** and do the following.
You can use the Self-Service playground to run the script, review the output, and make any required changes.
 - a. On the **Authorization** tab, specify the **IP Address** and **Port**.
 - b. In the **Select tunnel to connect with** dropdown menu, select the tunnel that you can use to get access to the VM within the VPC. This step is optional.
The **Select tunnel to connect with** dropdown menu shows only those tunnels that are allowed in the project you selected for the endpoint.
 - c. Specify the **Credential** for the test machine.
You can also specify the **Username** and **Password** for the test machine instead of the credential.
 - d. Click **Login and Test**.
 - e. On the **Test Script** tab, view or edit your script in the **Script** field.
 - f. For macros in your script, provide the values in the macro inspector panel.
 - g. Click **Assign and Test**.
The **Output** field displays the test result.
 - h. To go back to the Runbook Editor page, click **Done**.
10. To publish this task to the task library, click **Publish to Library** and then click **Publish**.
11. Click **Save** to save the runbook.

Creating a Runbook with a Delay Task

Create a runbook with the Delay task to set a delay interval between two tasks or actions.

Before you begin

Ensure that you have created a runbook and have added a task for the runbook. For more information, see [Creating a Runbook](#) on page 266.

Procedure

1. On the runbook **Editor** tab, click **+Add Task**.
2. In the **Task Name** field, type a name for the task.
3. From the **Type** dropdown menu, select the **Delay** task.
4. In the **Sleep Interval** field, type the sleep time interval in seconds for the task.
5. Click **Save** to save the runbook.

Creating a Runbook with an HTTP Task

Create a runbook with the HTTP task to perform REST calls to an HTTP endpoint.

Before you begin

Ensure that you have created a runbook and have added a task for the runbook. For more information, see [Creating a Runbook](#) on page 266.

Procedure

1. On the runbook **Editor** tab, click **+Add Task**.
2. In the **Task Name** field, type a name for the task.
3. From the **Type** dropdown menu, select the **HTTP** task.
4. From the **Endpoint** dropdown menu, select the endpoint where you want to execute the HTTP task. This step is optional.
5. In the **Request Method** dropdown menu, select one of the following request methods.
 - Select **GET** to retrieve data from a specified resource.
 - Select **POST** to send data to a server to create a resource, and enter or upload the POST request in the **Request Body** field.
 - Select **DELETE** to send data to a server to delete a resource, and enter or upload the DELETE request in the **Request Body** field.
 - Select **PUT** to send data to a server to update a resource, and enter or upload the PUT request in the **Request Body** field.
6. In the **Relative URL** field, enter the URL of the server on which you want to run the methods.
7. In the **Content Type** dropdown menu, select the type of the output format. The available options are **HTML**, **JSON**, and **XML**.
8. In the **Headers** section, type the HTTP header key and value in the **Key** and **Value** fields respectively.
9. If you want to publish the HTTP header key and value pair as secrets, select the **Secret** checkbox.
10. In the **Expected Response Options** area, do the following configurations.
 - a. Select **Success** or **Failure** as the **Response Status**, and type the **Response Code** for the status.

Note: If the response code is not defined, then by default all the 2xx response codes are marked as success, and any other response codes are marked as failure.
 - b. Under **Set Variables from response**, type the variables for the specified response path. The example of json format is \$.x.y and xml format is //x/y. For more information on json path syntax, see <http://jsonpath.com>.

Note: To retrieve the output format in HTML format, add a * in the syntax.
11. If you want to test the script in the Self-Service playground, click **Test Request**. The test result appears in the **Output field**.
12. Click **Save** to save the runbook.

Creating a Runbook with a While Loop Task

Create a runbook with the While Loop task to iterate over multiple tasks until the defined condition is met.

Before you begin

Ensure that you have created a runbook and have added a task for the runbook. For more information, see [Creating a Runbook](#) on page 266.

Procedure

1. On the runbook **Editor** tab, click **+Add Task**.
2. In the **Task Name** field, type a name for the task.
3. From the **Type** dropdown menu, select the **While Loop** task.
4. In the **Iterations** field, type the number of times you want to iterate the task till the task meet a condition. The default value is 1.
5. From the **Exit Condition** dropdown menu, select a condition after which the task iteration must stop. The available options are as follows.
 - Select **Success** if you want to stop the task iteration after the status of the task is success.
 - Select **Failure** if you want stop the task iteration after the status of the task is failure.
 - Select **Don't care** if you want to continue the task iteration irrespective of the status of the task.
6. Click **Save** to save the runbook.

Nutanix Database Service (NDB) Integration with Self-Service Runbooks

NDB (formerly Nutanix Era) automates and simplifies database administration across multiple locations, both on-premises and in the cloud, with Nutanix clusters.

By integrating NDB with Self-Service runbooks, you leverage the database management capabilities of NDB and the application management workflow of Self-Service using a common management plane. Self-Service uses runbooks to provide easy access to NDB actions.

To leverage NDB capabilities, you create runbook workflow and automate routine activities, such as provisioning new databases and performing maintenance tasks. This integration specifically helps address some important actions that are carried out before and after the creation of the database instance. For example, the integration provides the ability to reserve an IP address from an IPAM for the database server VM and then use that IP address to provision the database server VM.

You can also configure runbooks to perform potential automation tasks after the database server VM is in operation to make those tasks efficient and error free.

Note: Self-Service supports integration of NDB version 2.5.

The integration involves the following steps:

1. Connecting NDB under Nutanix Service within the Nutanix Account.

You can do this on the Accounts tab in the Self-Service Settings. For more information, see [Connecting Nutanix Database Service \(NDB\) to a Nutanix Account](#) on page 76.

2. Configuring a runbook for NDB. NDB entities are currently only available within runbooks.

For more information, see [Configuring a Runbook for Database Management](#) on page 274.

Configuring a Runbook for Database Management

You can configure the Database Operation type runbooks to address your database requirements using the integrated Nutanix Database Services (NDB) accounts.

Before you begin

Ensure that you have connected your NDB accounts to the Nutanix accounts in Self-Service. For more information, see [Connecting Nutanix Database Service \(NDB\) to a Nutanix Account](#) on page 76.

Procedure

1. Log in to your Prism Central instance.
2. In the Application Switcher, select **Self-Service**.
3. In the navigation bar, click **Runbooks**.
4. On the **Runbooks** tab, click **Create Runbook**.
5. Specify the following basic details for the runbook in the Create Runbook window.

The screenshot shows a 'Create Runbook' window. It has a title bar with the text 'Create Runbook' and a close button (X). The window contains the following fields:

- Name:** A text input field containing 'Runbook1'.
- Description:** A text area with a placeholder 'Description' and a 'View' link to its right.
- Project:** A dropdown menu with 'Self-Service' selected.
- Default Endpoint (Optional):** A dropdown menu with 'Endpoint' selected. There is a help icon (i) next to the label.

At the bottom right of the window are two buttons: 'Cancel' and 'Proceed'.

Figure 80: Create Runbook

- In the **Name** field, type a name for the runbook.
- In the **Description** field, type a brief description about the runbook. This step is optional.
- From the **Project** dropdown menu, select a project to which you want to add the runbook.
- From the **Default Endpoint** dropdown menu, select a default endpoint. This step is optional.

Self-Service uses the default endpoint only when you do not configure any endpoint at the task level.

6. Click **Proceed**.

7. On the **Editor** tab, click **+ Add Task**.

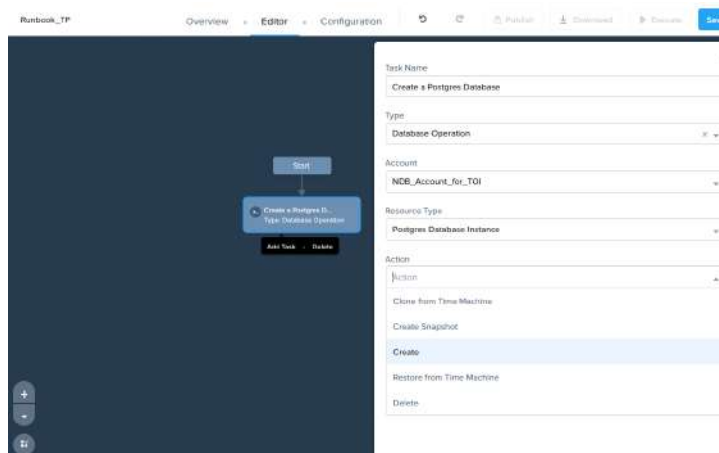


Figure 81: Runbook Editor

8. In the **Task Name** field, type a name for the task.
9. From the **Type** dropdown menu, select **Database Operation** as the task type.
10. From the Account dropdown menu, select the NDB account that you connected to your Nutanix account in Self-Service.
11. In the **Resource Type** dropdown menu, select **Postgres Database Instance**.
12. In the **Action** dropdown menu, select one of the following actions and configure the related fields.
 - » **Clone from Time Machine:** You can configure the database clone operation to create clones either to a point in time (by using transaction logs) or by using snapshots. For more information, see [Configuring the Clone Action for Database Management Runbook](#) on page 277.
 - » **Create Snapshot:** You can configure the database snapshot operation to capture and store the state of a database at various points in time (as you specify in the schedule). For more information, see [Configuring Snapshot Action for Database Management Runbook](#) on page 278.
 - » **Create:** You can configure the database create operation to create a database server VM on the Nutanix cluster on which you provision the database. For more information, see [Configuring Create Action for Database Management Runbook](#) on page 279.
 - » **Restore from Time Machine:** You can configure the database restore operation to replace the current instance with data as of the specified snapshot or point in time. For more information, see [Configuring Restore Action for Database Management Runbook](#) on page 282.
 - » **Delete:** You can configure the database delete operation to delete the database and the time machine that is associated with the database. For more information, see [Configuring Delete Action for Database Management Runbook](#) on page 282.
13. Click **Save** to save the runbook configuration.

What to do next

You can execute the runbook. For more information, see [Executing a Runbook](#) on page 285. You can also publish the runbook so that your admin can approve and make the runbook available at the marketplace. For more information, see [Submitting a Runbook for Publishing](#) on page 283.

Configuring the Clone Action for Database Management Runbook

The time machine entity captures essential data in the form of snapshots (synthetic full backups) and transaction logs of source databases as defined in the Data Access Management policies (DAM policies) and SLA schedules. These snapshots provide the required data and metadata to create a clone (a copy of the database instance or database at the point of the snapshot).

About this task

Perform the following steps to configure the Clone from Time Machine action in your database management runbook.

Before you begin

Ensure that you have the basic configuration ready for your database management runbook. For more information, see [Configuring a Runbook for Database Management](#) on page 274.

Procedure

1. On the runbook **Editor** tab, select **Clone from Time Machine** in the **Action** dropdown menu in the inspector panel.
2. Select the time machine from which the clone has to be performed in the **Target Time Machine** dropdown menu.
3. Select the **Nutanix cluster** on which you want to clone the instance.
4. Do one of the following:
 - » Select **Point in time** if you want to clone the source instance to a point in time. If you select this option, the instance is cloned by using the transaction logs.
 - » Select **Snapshot** if you want to clone the source instance by using the available snapshots.
5. In the **Database Server VM** section, configure the following:
 - a. Type a name for the database server VM in the **Database Server VM Name** field.
 - b. Type a description for the database server VM in the **Description** field.
 - c. Specify a password for the Database Server VM in the **Password** field.
 - d. Select a compute profile from the **Compute Profile** dropdown menu.

A compute profile is a configuration template that defines the resources and settings for deploying and managing database instances.
 - e. Select a network profile from the **Network Profile** dropdown menu.

A network profile is a configuration template that defines the network settings and connectivity options for database instances.
 - f. **SSH Public Key for Node Access.** Do one of the following to use SSH public keys to access the database server VM:
 - » Click the Upload icon inside the text box to upload a file that includes the public key.
 - » Type or copy and paste the public key in the text box.

6. In the **Instance** section, configure the following:
 - a. Type a name for the cloned instance in the **Instance Name** field.
 - b. Type a description for the cloned instance in the **Description** field.
 - c. Type the password of the Postgres superuser in the **Password** field.
 - d. Select a database parameter profile from the **Database Parameter Profile** dropdown menu.
A database parameter profile is a template of custom database parameters that you want to apply to your database.
 - e. In the **Pre/Post Commands** section, do the following
 - Type a complete operating system command that you want to run before the instance is created in the **Pre Create Command** field.
 - Type a complete operating system command that you want to run after the instance is created in the **Post Create Command** field.
 - f. **Schedule Data Refresh**. You can schedule data refresh to refresh the clone automatically. Select this option and define the frequency and time slots to refresh the clone.
 - g. **Removal Schedule**. You can schedule the removal of the instance clone by specifying the number of days after which the clone must be deleted automatically.
7. In the **Tags** section, type a tag value for each tag.
A tag allows you to assign metadata labels to databases, database instances, or workflows and provides a way to categorize and organize resources based on specific attributes or characteristics.
8. In the **Output** section, use the variable names and output parameters to quickly consume data in the subsequent tasks.
The output parameters are referenced as macros in the subsequent task. For few NDB tasks, the output parameters are set automatically. You can also provide an alias of your choice for the output parameters so that you can use them in the subsequent task.
You can also pass the runbook variables available on the Configuration tab to the fields in the NDB tasks.
9. Use the runbook variables in the On the configuration task,
10. Click **Save** to save the runbook.

Configuring Snapshot Action for Database Management Runbook

Snapshots store the state of a database at various points in time.

About this task

Perform the following steps to configure the Create Snapshot action in your database management runbook.

Before you begin

Ensure that you have the basic configuration ready for your database management runbook. For more information, see [Configuring a Runbook for Database Management](#) on page 274.

Procedure

1. On the runbook **Editor** tab, select **Create Snapshot** in the **Action** dropdown menu in the inspector panel.

2. Select the time machine of the database for which the snapshot has to be created in the **Target Time Machine** dropdown menu.
3. Specify a name for the snapshot in the **Snapshot Name** field.
4. Select the **Removal Schedule** checkbox to define a snapshot removal schedule and then enter the number of days after which the snapshot should be removed.
5. In the Output section, use the variable names and output parameters to quickly consume data in the subsequent tasks.

The output parameters are referenced as macros in the subsequent task. For few NDB tasks, the output parameters are set automatically. You can also provide an alias of your choice for the output parameters so that you can use them in the subsequent task.

You can also pass the runbook variables available on the Configuration tab to the fields in the NDB tasks.
6. Click **Save**.

Configuring Create Action for Database Management Runbook

Use the database Create operation to create a database server VM on the Nutanix cluster on which you provision the database. You can also create the database instance. Perform the following steps to configure the Create action in your database management runbook.

Before you begin

Ensure that you have completed the basic configuration for your database management runbook. For more information, see [Configuring a Runbook for Database Management](#) on page 274.

Procedure

1. On the runbook **Editor** tab, in the inspector panel, select **Create** in the **Action** dropdown menu.

2. In the **Database Server VM** section, do the following:

- a. In the **Database Server VM Name** field, type a name for the database server VM.
Self-Service uses the name that you provide in this field for the VM name of the PostgreSQL instance.
- b. In the **Database Server Description** field, type a description for the database server VM.
- c. From the **Cluster** dropdown menu, select the Nutanix cluster on which you want to provision the instance.
- d. From the **Software Profile** dropdown menu, select a software profile.
A software profile is a configuration template that defines the settings and specifications for installing and managing a specific database software.

Note: A software profile appears in the **Software Profile** dropdown menu only if you create or replicate the profile on the selected Nutanix cluster.

- e. From the **Software Profile Version** dropdown menu, select a software profile version.
- f. From the **Compute Profile** dropdown menu, select a compute profile.
A compute profile is a configuration template that defines the resources and settings for deploying and managing database instances.
- g. From the **Network Profile** dropdown menu, select a network profile.
A network profile is a configuration template that defines the network settings and connectivity options for database instances.
- h. In the **SSH Public Key for Node Access** field, select one of the following options to use SSH public keys to access the database server:
 - » Click the Upload icon inside the text box to upload a file that includes the public key.
 - » Type or copy and paste the public key in the text box.

3. In the **Instance** section, do the following:

- a. In the **Instance Name** field, type a name for the instance.
- b. In the **Description** field, type a description for the instance. This step is optional.
- c. Ensure that the **Port** field is populated with the required port number.
The default value of the **Port** field is the port number 5432 of a PostgreSQL instance.
- d. In the **Size (GiB)** field, type the size of the node in GB.
The default value of the **Size (GiB)** field is 200 GiB.
- e. In the **Name of Initial Database** field, type the name of the initial database that is created in the PostgreSQL instance.
- f. From the **Database Parameter Profile** dropdown menu, select a database parameter profile.
A database parameter profile is a template of custom database parameters that you can apply to your database.
- g. In the **Password** field, type the password of the PostgreSQL superuser account.
- h. Under the **Pre/Post Commands** section, do the following.
 - In the **Pre Create Command** field, type a complete operating system command that you want to run before the instance is created.
 - in the **Post Create Command** field, type a complete operating system command that you want to run after the instance is created.

4. In the **Time Machine** section, do the following:

- a. In the **Name** field, type a name for the time machine.
- b. In the **Description** field, type a description for the time machine. This step is optional.
- c. From the **SLA** dropdown menu, select an SLA.
An SLA is a snapshot retention policy that indicates how long snapshots are retained. For more information, see [SLA Management](#) in the *Nutanix Database Service Administration Guide*.

5. In the **Schedule** section, specify a schedule to take and retain the snapshots. To specify a schedule, do the following:

- a. In the **Initial Daily Snapshot at** field, specify the daily snapshot time.
The snapshot taken at this time every day is retained as the daily snapshot.
- b. In the **Snapshots Per Day** field, specify the number of snapshots to take every day.
- c. In the **Log Catch Up Every** dropdown menu, select the frequency of log catch-ups in minutes.
The log catch-up operation copies transaction logs from your source database.
- d. From the **Weekly Snapshot on** dropdown menu, select the weekly snapshot day.
The snapshot taken on this day of every week is retained as the weekly snapshot.
- e. From the **Monthly Snapshot on the** dropdown menu, select the monthly snapshot day.
The snapshot taken on this day of every month is retained as a monthly snapshot.
- f. From the **Quarterly Snapshot in** dropdown menu, select an option for the quarterly snapshot.
The snapshot taken on the first day of the first month of the quarter is retained as a quarterly snapshot. For example, if you select **Jan, Apr, Jul, Oct**, snapshots taken on January 1, April 1, July 1, and October 1 are retained as quarterly snapshots.

6. In the **Tags** section, type a tag value for each tag.

A tag allows you to assign metadata labels to databases, database instances, or workflows and provides a way to categorize and organize resources based on specific attributes or characteristics.

7. In the Output section, use the variable names and output parameters to quickly consume data in the subsequent tasks.

The output parameters are referenced as macros in the subsequent task. For few NDB tasks, the output parameters are set automatically. You can also provide an alias of your choice for the output parameters so that you can use them in the subsequent task.

You can also pass the runbook variables available on the Configuration tab to the fields in the NDB tasks.

8. Click **Save** to save the runbook.

Configuring Restore Action for Database Management Runbook

A database restore operation replaces the current instance with data as of the specified snapshot or point in time.

About this task

Perform the following steps to configure the Restore from Time Machine action in your database management runbook.

Before you begin

Ensure that you have the basic configuration ready for your database management runbook. For more information, see [Configuring a Runbook for Database Management](#) on page 274.

Procedure

1. On the runbook **Editor** tab, select **Restore from Time Machine** in the **Action** dropdown menu in the inspector panel.
2. In the **Target Database** field, select the database.
3. Under the Select Restore Source section, do one of the following:
 - » Select **Point in Time** and enter the time to which you want to restore your instance.
 - » Select **Snapshot** and then select the snapshot from the dropdown menu that you want to use for restoring the instance.
4. In the Output section, use the variable names and output parameters to quickly consume data in the subsequent tasks.

The output parameters are referenced as macros in the subsequent task. For few NDB tasks, the output parameters are set automatically. You can also provide an alias of your choice for the output parameters so that you can use them in the subsequent task.

You can also pass the runbook variables available on the Configuration tab to the fields in the NDB tasks.
5. Click **Save** to save the runbook.

Configuring Delete Action for Database Management Runbook

The Delete action deletes the database and the time machine that is associated with the database.

About this task

Perform the following steps to configure the Delete action in your database management runbook.

Before you begin

Ensure that you have the basic configuration ready for your database management runbook. For more information, see [Configuring a Runbook for Database Management](#) on page 274.

Procedure

1. On the runbook **Editor** tab, select **Delete** in the **Action** dropdown menu in the inspector panel.
2. Select the target database that you want to delete in the **Target Database** dropdown menu.
3. Click **Save** to save the runbook.

Submitting a Runbook for Publishing

Submit a runbook for publishing so that your admin can approve and publish it at the marketplace. Members from the associated projects can view and execute the runbooks that are published at the marketplace.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Runbooks** in the navigation bar.
4. Do one of the following:
 - » To publish an existing runbook, click to open a runbook from the list.
 - » To publish a new runbook, click **Create Runbook** and create a new runbook. For information on how to create a runbook, see [Creating a Runbook](#) on page 266.

5. On the Runbook Editor page, click **Publish**.
The **Publish Runbook** window appears.


Publish Runbook

1 General 2 Change Log

To publish to the Marketplace, the runbook will be submitted for approval to your administrator.

Publish as a

- ☒ New Marketplace Runbook
- ☐ New version of an existing Marketplace Runbook

 [Change](#)

Name

☐ Publish with secrets [i](#)

☐ Publish with endpoints, accounts and resource types [i](#)

Initial Version

Description [View](#)

[Submit for Approval](#) [Cancel](#)

Figure 82: Publish Runbook

6. To publish a new runbook, do the following.
- Select **New Marketplace Runbook**.
 - Provide a name for the runbook to be used at the marketplace.
 - Turn on the **Publish with secrets** to encrypt secret values such as the credential passwords, keys, and secret variables.
By default, the secret values from the runbook are not preserved when you publish them. Turn on this option if you do not want to fill the secret values or to be patched from the environment when you execute the runbook.
 - Turn on **Publish with endpoints, accounts, and resource types** to preserve the endpoints, accounts, and resource types values.
By default, the endpoints, accounts, and resource types from the runbook are not preserved when you publish them. Turn on this option if you do not want to fill these values when you execute the runbook.
 - In the **Initial Version** field, type a new version number.
 - Provide a description for the runbook. This step is optional.

7. To publish a newer version of an already published runbook, do the following:

- a. Select **New version of an existing Marketplace Runbook**.
- b. From the **Marketplace Item** dropdown menu, select the existing runbook.
- c. Turn on **Publish with secrets** and **Publish with endpoints**.
- d. Specify the version for the runbook for the existing runbook.
- e. Provide a description for the runbook. This step is optional.

8. Click **Submit for Approval**.

Self-Service submits the runbook to the Marketplace Manager for your admin to approve and publish the runbook to the Marketplace.

What to do next

If you are the admin, you can approve and publish the runbook from the Marketplace Manager. For information on approving and publishing the runbook, see [Approving and Publishing a Blueprint or Runbook](#) on page 232. If you are a project admin or a developer, you can request your admin to approve and publish the runbook at the Marketplace.

Executing a Runbook

You can execute a runbook to run the tasks sequentially on an endpoint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Runbooks** in the navigation bar.
4. Select the runbook that you want to execute.

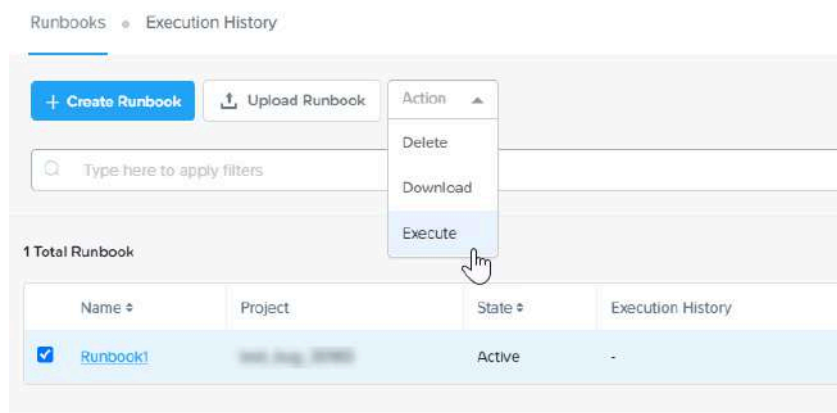


Figure 83: Execute Runbook

5. From the **Action** dropdown menu, select **Execute**.
The **Execute Runbook** page appears.
6. To change the default endpoint for the execution, select an endpoint from the **Endpoints** dropdown menu. This step is optional.

7. To update the added variable to the Runbook, click the respective variable field and edit the variable. This step is optional.

Note: You can update the variable only if you mark the variable as runtime editable while adding it in the Runbook.

8. Click **Execute**.

The runbook execution starts and you are directed to the Runbook execution page.

What to do next

You can view all the executions in the **Execution History** tab.

Runbook Execution using a Playbook

A playbook allows you to define a trigger that results in the execution of an action or a series of actions. A trigger might be an event that occurs in the system, such as an alert or a request made by you.

With the X-Play and Self-Service integration in pc.2024.1 and later versions, you can use the automation capabilities of playbooks to execute a runbook you created in Self-Service. You can configure a playbook to execute a marketplace runbook or a non-marketplace runbook.

Prerequisites

- Ensure that you have created a runbook in Self-Service with a project assigned to the runbook. For information on creating a runbook, see [Creating a Runbook](#) on page 266.
- Ensure that you have Intelligent Operations enabled in your Prism Central instance. For more information, see the [Enabling Intelligent Operations](#) section in the *Intelligent Operations Guide*.

Playbook Configuration

The action that you use to configure a playbook for runbook execution is **Execute a Self-Service Runbook**. You use this action while creating a playbook for your runbook automation.

Use Intelligent Operations to create a playbook with any of the available triggers and **Execute a Self-Service Runbook** as the action. For information on creating a playbook with different triggers, see [Creating a Playbook](#) in the *Intelligent Operations Guide*.

To configure the action, you need to set the action attributes based on your requirements. For information on the specific attribute requirement for the **Execute a Self-Service Playbook** action, see the [Playbook Actions](#) table in the *Intelligent Operations Guide*.

For information on how to run a playbook that you configured with a Manual trigger, see the [Running a Playbook \(Manual Trigger\)](#) section in the *Intelligent Operations Guide*.

Deleting a Runbook

Perform the following procedure to delete a runbook.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Runbooks** in the navigation bar.
4. Select the runbook that you want to delete.

5. From the **Action** dropdown menu, select **Delete**.
6. In the Confirm Delete window, click **Delete**.

MULTITENANT CAPABILITY FOR SERVICE PROVIDERS

Tenancy in Self-Service offers the ability to partition resources and workflows between different groups of users or organizations. Self-Service provides a tenant management platform for service providers with multitenant capability.

The multitenant capability of Self-Service enables service providers to offer Infrastructure as a Service (IaaS) to tenants without any constraints of assigning one cluster for each tenant. With multitenancy, service providers can allow multiple tenants to consume resources of one cluster or allow one tenant to consume resources from multiple clusters. Self-Service uses projects as a tenant construct to isolate resources at an account level as well as provide a management interface for tenants.

Self-Service provides the Service Provider Pack to automate the entire process of onboarding tenants and achieve multitenancy. The Service Provider Pack has a collection of runbooks that help service providers automate workflows such as tenant onboarding and Disaster Recovery as a Service (DRaaS) with just one-click operations.

Ensure that you meet the following requirements to use the Service Provider Pack:

- A management cluster or equivalent to host Self-Service VM.
- Self-Service VM (formerly Calm VM) version 3.6.2 or later.
- Prism Central version pc.2023.1.0.1 or later.
- AOS 6.5.2.X or latest AOS STS release.

You have the following runbooks in Service Provider Pack:

- Tenant Onboarding Runbook Variables. For more information, see [Tenant Onboarding Runbook](#) on page 289.
- External Subnet Management Runbook Variables. For more information, see [External Subnet Management Runbook](#) on page 300.
- Virtual Private Cloud Management Runbook Variables. For more information, see [Virtual Private Cloud Management Runbook](#) on page 306.
- Overlay Subnet Management Runbook Variables. For more information, see [Overlay Subnet Management Runbook](#) on page 309.
- Users and Groups Management Runbook Variables. For more information, see [Users and Groups Management Runbook](#) on page 319.
- VPC Static Routing Runbook Variables. For more information, see [VPC Static Routing Runbook](#) on page 330.
- VLAN Subnet Management Runbook Variables. For more information, see [VLAN Subnet Management Runbook](#) on page 334.
- Disaster Recovery Runbook Variables. For more information, see [Disaster Recovery Runbook](#) on page 342.
- Network Configuration Runbook Variables. For more information, see [Network Configuration Runbook](#) on page 355.
- Policy-Based Routing Runbook Variables. For more information, see [Policy-Based Routing Runbook](#) on page 362.

- Backup and Restore Runbook Variables. For more information, see [Backup and Restore Runbook](#) on page 373.
- Floating IP Assignment Runbook Variables. For more information, see [Floating IP Assignment Runbook](#) on page 377.
- Single-VM Blueprint Runbook Variables. For more information, see [Single-VM Blueprint Runbook](#) on page 379.
- Test Failover Runbook Variables. For more information, see [Test Failover Runbook](#) on page 383.

Accessing the Service Provider Pack Runbooks

You can use the following link to get access to the Service Provider Pack runbooks:

https://github.com/nutanix/calm_runbooks

Tenant Onboarding Runbook

As a service provider, you can execute the Tenant Onboarding runbook from a management cluster to onboard a new tenant. The runbook then creates all the required virtual resources on a Workload cluster for the tenant to run their applications.

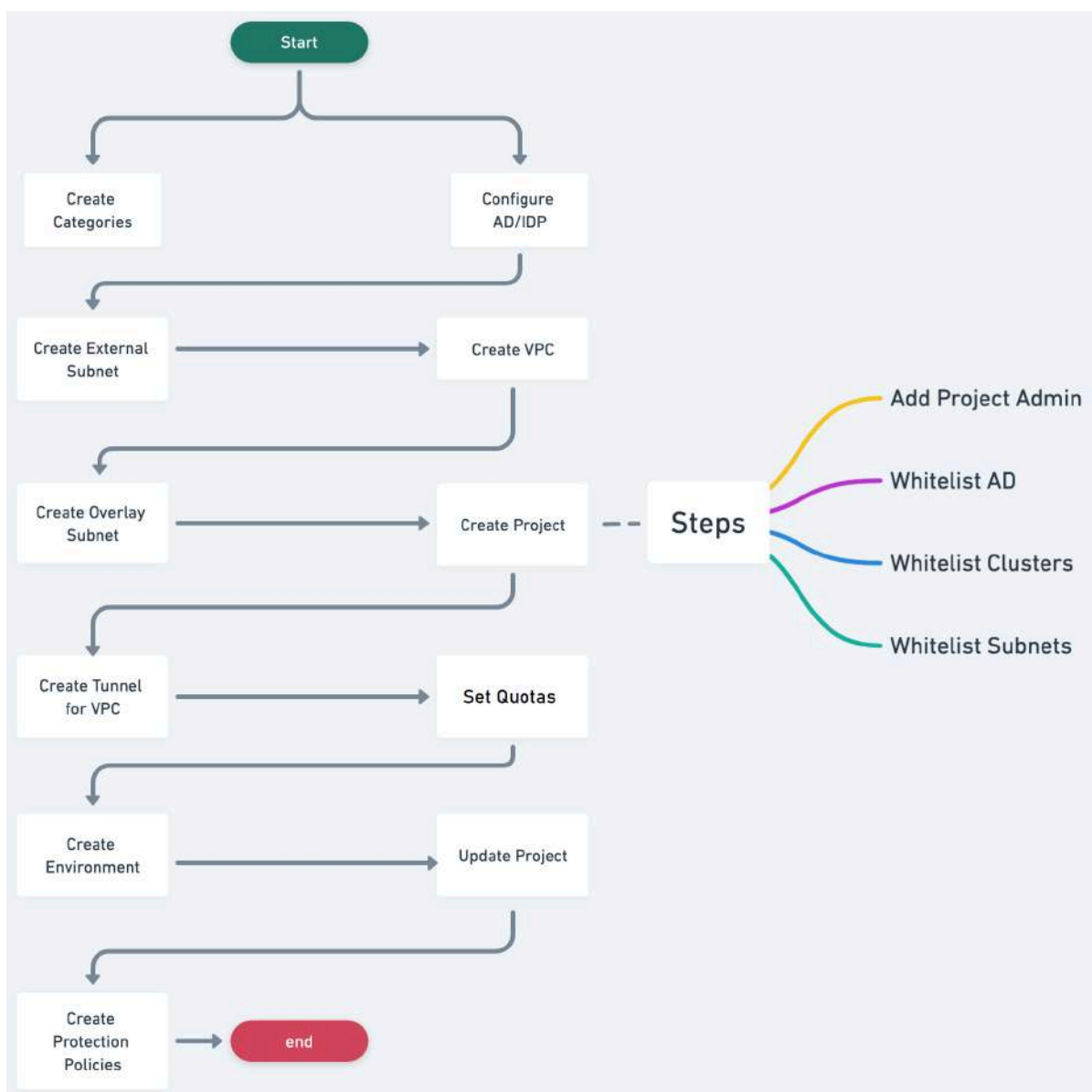


Figure 84: Tenant Onboarding Workflow

The Tenant Onboarding runbook has the following variables:

Table 43: Tenant Onboarding Variables

Variable	Description	Condition	Type
Management PC IP	<p>Provided as input to specify the IP address or hostname of the Nutanix Prism Central management interface. The Prism Central instance acts as a centralized management interface for multiple Nutanix clusters, allowing administrators to manage and monitor workloads across different clusters from a single interface.</p> <p>Ensure that you keep the Management Prism Central IP secure, as it provides access to the management interface and the Nutanix clusters it manages.</p>	Required	String
Management PC Username	<p>Provided as input to specify the username used for authentication with the Nutanix Prism Central management interface. The management username is required for performing administrative tasks on the Nutanix cluster through the Prism Central management interface.</p> <p>The Nutanix Prism Central management interface provides various features such as health monitoring, capacity planning, resource management, and user management.</p>	Required	String
Management PC Password	<p>Provided as input to specify the password used for authentication with the Nutanix Prism Central management interface. The management password is required for performing administrative tasks on the Nutanix cluster through the Prism Central management interface.</p> <p>The password is used in combination with the management username to authenticate and access features such as health monitoring, capacity planning, resource management, and user management.</p>	Required	String

Variable	Description	Condition	Type
Tenant Name	<p>Provided as input to specify the name of the tenant or organization associated with the Nutanix cluster.</p> <p>The tenant name is useful for large enterprises or service providers that manage multiple Nutanix clusters for different departments or customers. If not provided, the default tenant name is used.</p> <div> Note: When using a runbook, resources such as subnets, projects, and VPCs are created under a specific tenant. The tenant name is used as a prefix to organize the resources accordingly. </div>	Required	String
Delete Existing Setup	Used to define the clean-up process for obsolete or pre-existing resources such as projects, VPCs, tunnels, and subnets that are prefixed with a tenant name.	Required	String
Workload PC IP	<p>Provided as input to specify the IP address or hostname of the Nutanix Prism Central instance that is used for workload management.</p> <p>This variable is required for any operations that involve workload management through Prism Central, such as creating or managing virtual machines, configuring storage policies, and setting up data protection policies.</p>	Required	String
Workload PC Username	<p>Provided as input to specify the username used for authentication with the Nutanix Prism Central instance for workload management.</p> <p>The management username is required for performing administrative tasks related to workload management on the Nutanix clusters through the Prism Central management interface.</p>	Required	String

Variable	Description	Condition	Type
Workload PC Password	<p>Provided as input to specify the password used for authentication with the Nutanix Prism Central instance for workload management.</p> <p>The password is used in combination with the management username to authenticate and access features such as creating and managing virtual machines, configuring storage policies, and setting up data protection policies.</p>	Required	String
Active Directory URL	<p>Provided as input to specify the URL of the Active Directory (AD) server that is used for user authentication and authorization on the Nutanix cluster.</p>	Required	String
Active Directory Domain Name	<p>Provided as input to specify the name of the Active Directory (AD) domain that is used for user authentication and authorization on the Nutanix cluster.</p>	Required	String
Active Directory Username	<p>Provided as input to specify the username used for authentication with the Nutanix cluster using the Active Directory (AD) server.</p>	Required	String
Active Directory Password	<p>Used to set the password for the Nutanix Active Directory (AD) service account used for user authentication and authorization on the Nutanix cluster.</p>	Required	String
Project Admin	<p>Used to specify the username of the Nutanix project administrator who has permission to create, manage and delete projects on the Nutanix cluster.</p> <p>This variable is required for any operations that require creating, managing, or deleting projects on the Nutanix cluster using APIs or CLI.</p>	Required	String

Variable	Description	Condition	Type
Cluster Name	<p>Used to specify the name of the Nutanix cluster that is used for the deployment.</p> <p>The cluster name is a unique identifier for a Nutanix cluster and is used for managing resources such as virtual machines, networks, storage, and user accounts.</p> <p>This variable is required for any operations that interact with the Nutanix cluster using API or CLI.</p>	Required	String
Virtual Switch Name	<p>Used to specify the name of the virtual switch on the Nutanix cluster that is used for networking operations.</p> <p>A virtual switch is a software-based network switch that connects virtual machines to each other and to the physical network.</p> <p>This variable is required for any operations that involve configuring or managing virtual networks on the Nutanix cluster using API or CLI.</p>	Required	String
External VLAN ID	<p>Used to specify the Virtual Local Area Network (VLAN) ID for the external network connection on the Nutanix cluster.</p> <p>If you have an external network connection, you can use this variable to specify the VLAN ID to be used for communication with the external network.</p> <p>This variable is required only if you have an external network connection configured on the Nutanix cluster.</p>	Required	Integer
External Subnet IP with Prefix	<p>Used to specify the IP address and prefix length for the external subnet on the Nutanix cluster.</p> <p>The IP address and prefix length together define the range of IP addresses that can be used for the external network connection.</p> <p>This variable is required only if you have an external network connection configured on the Nutanix cluster, and you need to specify the IP address and prefix for the external subnet.</p>	Required	String

Variable	Description	Condition	Type
External Subnet IP Pool Range	<p>Used to specify the IP address pool range for the external subnet on the Nutanix cluster.</p> <p>The IP address pool range defines the range of IP addresses that can be used for the external network connection.</p> <p>This variable is required only if you have an external network connection configured on the Nutanix cluster, and you need to specify the IP address pool range for the external subnet.</p>	Required	String
External Subnet Gateway IP	<p>Used to specify the IP address of the default gateway for the external subnet on the Nutanix cluster.</p> <p>The default gateway is the IP address of the router that connects the external subnet to other networks.</p> <p>This variable is required only if you have an external network connection configured on the Nutanix cluster, and you need to specify the IP address of the default gateway for the external subnet.</p>	Required	String
External Subnet NAT	<p>Used to specify the Network Address Translation (NAT) IP address for the external subnet on the Nutanix cluster.</p> <p>NAT is used to map one IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device.</p> <p>This variable is required only if you have an external network connection configured on the Nutanix cluster and need to specify the NAT IP address for the external subnet.</p>	Required	String
Overlay Subnet IP With Prefix	<p>Used to specify the IP address and prefix length for the overlay subnet on the Nutanix cluster.</p> <p>The overlay subnet is used for virtualized workloads and is isolated from the external networks. The IP address and prefix length together define the range of IP addresses available for use within the overlay subnet.</p>	Required	String

Variable	Description	Condition	Type
Overlay Subnet Gateway IP	<p>Used to specify the IP address of the gateway for the overlay network subnet used for communication between virtual machines within the Nutanix cluster.</p> <p>The gateway IP address is used as the default gateway for the virtual machines within the overlay network.</p> <p>This variable is required for any operations that involve configuring or managing the overlay network on the Nutanix cluster, such as creating virtual machines, configuring network security policies, and setting up load balancing.</p>	Required	String
Account Name	<p>Used to specify the name of the Nutanix storage account used for provisioning and managing storage resources on the Nutanix cluster.</p> <p>Storage accounts provide a logical grouping of storage resources, such as containers and datastores, and are used to allocate and manage storage capacity for various applications and workloads.</p> <p>The Account Name variable is used for the purpose of adding infrastructure to projects. You get the information about this variable on the Accounts tab of the Self-Service Settings.</p> <p>This variable is required for any operations that involve storage provisioning or management on the Nutanix cluster using the Prism Central management interface or APIs.</p>	Required	String
Quota : vCPUs	<p>Used to specify the maximum number of virtual CPUs (vCPUs) that can be allocated to a project on the Nutanix cluster.</p> <p>The vCPU quota determines the number of virtual processors that can be used by virtual machines (VMs) running within the project.</p>	Required	Integer

Variable	Description	Condition	Type
Quota : Memory in GB	<p>Used to specify the maximum amount of memory in gigabytes that can be consumed by virtual machines running in a Nutanix project.</p> <p>The memory quota is a limit set on the amount of memory that can be allocated to virtual machines, and it ensures that the project does not exceed its resource allocation.</p> <p>This quota can be adjusted as needed to optimize resource usage and meet changing workload demands.</p>	Required	Integer
Quota : Disk Size in GB	<p>Used to specify the disk size quota in GB for a Nutanix project.</p> <p>The disk size quota refers to the maximum disk space that can be used by virtual machines created within the project.</p> <p>This variable is required for any operation that involves creating virtual machines within the project.</p>	Required	Integer
Allow Project Collaboration	<p>Used to allow multiple projects to collaborate and share resources within a single Nutanix cluster.</p> <p>When this feature is enabled, users can be added to multiple projects and can access resources such as virtual machines, networks, and storage across all projects.</p>	Required	String
Create Project Environment with Default Values	<p>Refers to the process of setting up an isolated environment for a specific project on the Nutanix cluster. This involves configuring resources such as virtual machines, networks, storage, and other services required for the project.</p> <p>This refers to the creation of a Self-Service environment within a Nutanix project, which allows for the deployment and management of applications and services.</p>	Required	String

Variable	Description	Condition	Type
Environment operating system	<p>Refers to the type of operating system that is installed on the virtual machines within the project.</p> <p>It is usually specified during the creation of a project environment and can be set to a variety of operating systems, including Windows, Linux, and others.</p> <p>Note: When you set the Create Project Environment with Default Values variable as Yes, then it is mandatory to select a proper environment operating system.</p>	Required	String
Image Name	<p>Used to specify the name of the Nutanix image that is used to create a new VM or update an existing one.</p> <p>An image is a pre-configured and optimized template that contains the operating system, applications, and other necessary components for a particular workload.</p> <p>The image name must be unique within the Nutanix cluster and match the name of an existing image in the image repository.</p> <p>Note: When you set the Create Project Environment with Default Values variable as Yes, then it is mandatory to pass the proper image name.</p>	Required	String

Variable	Description	Condition	Type
Guest Customization Script	<p>Used to automate the configuration of virtual machines running on the Nutanix AHV hypervisor.</p> <p>This script is used to install software, configure settings, and perform other tasks on the virtual machine.</p> <p>The script can be written in any scripting language that is supported by the guest operating system, such as PowerShell or Bash, and can be executed during the virtual machine provisioning process.</p> <p>This variable should be provided as input to specify the path to the guest customization script that should be executed on the virtual machine during provisioning.</p> <div> <p>Note: The customization script for Linux environments should be encoded in the Ansible Base64 format. For Windows environments, the script should be in the XML format.</p> </div>	Optional	Multi-line String
Environment Credential Username	<p>Refers to the username required to authenticate and access the environment in the Nutanix project.</p> <p>Used for purposes such as deploying and configuring applications, managing virtual machines, and performing other operations within the project environment.</p>	Required	String
Credential Type	<p>Refers to the type of credential (such as username and password, SSH key, certificate, and so on) used to authenticate and authorize access to resources in the Nutanix environment.</p> <p>The credential type may vary based on the type of resource being accessed and the security requirements of the organization.</p> <p>The credential type can be specified as input when creating or configuring resources in the Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Password Or Key	<p>Indicates the authentication method required to access the target machine during deployment.</p> <p>When using a password for authentication, the value must be provided as plain text. For an SSH key, the value must be provided in the PEM format.</p> <p>The actual value entered is determined by the specific deployment criteria and security guidelines in place.</p>	Required	Multi-line String

External Subnet Management Runbook

To provide connectivity to the VMs in a VPC, each VPC requires connectivity to the external environment. This connectivity can be either NAT (Network Address Translation) or No-NAT and is VLAN based. The network used for external connectivity is called an external subnet. You can use this runbook to create the additional external subnet for this connectivity.

In a NAT scenario, the IP address of the VM is translated when it exits the logical router of the VPC. However, in the No-NAT scenario, the IP address of the VM is not translated when it exits the logical router of the VPC.

The external subnet management runbook has the following variables.

Table 44: External Subnet Management Runbook Variables

Variable	Description	Condition	Type
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String

Variable	Description	Condition	Type
Subnet Name	<p>Refers to the name of the subnet that enables external connectivity for a Nutanix cluster. The purpose of this variable is to assign external IP addresses to the virtual machines and networking resources such as load balancers within the cluster.</p> <p>The external subnet must have an adequate number of available IP addresses to handle the anticipated workload of the cluster.</p>	Required	String
Operation	<p>Used to set the type of operation to be performed on the external subnet in the Nutanix environment. You have the following options for this variable.</p> <ul style="list-style-type: none"> • Create: Used to create a new external subnet in the Nutanix environment. • Update: Used to modify an existing external subnet in the Nutanix environment. • Delete: Used to remove an existing external subnet from the Nutanix environment. <p>The value of this variable determines which specific API call is made to the Nutanix management platform. Certain operations may require additional parameters or configuration options depending on the specific resource being created, updated, or deleted. Refer to the Nutanix documentation and API references for more information on the requirements for each operation.</p> <div> <p>Note: The external subnet is used to provide external connectivity to a Nutanix cluster. Therefore, any changes made to it can have an impact on the overall functionality of the cluster. Ensure that you follow best practices and test any changes in a non-production environment before implementing them in a production environment.</p> </div>	Required	String

Variable	Description	Condition	Type
VLAN ID	<p>Refers to the Virtual Local Area Network (VLAN) ID associated with the external subnet in a Nutanix cluster. It is used to identify the external subnet within the network infrastructure and enable the traffic to flow to and from the external network.</p> <p>The VLAN ID must be unique within the network infrastructure and configured appropriately to allow for communication between the external network and the Nutanix cluster.</p> <p>Ensure that the External Subnet VLAN ID is correctly configured and maintained to avoid any disruption to network connectivity for the Nutanix cluster.</p>	Required	Integer
External VLAN UUID	<p>Used to set the UUID of the VLAN that is associated with the external subnet in the Nutanix cluster. The UUID is used to uniquely identify the VLAN within the network infrastructure and enable the traffic to flow to and from the external network.</p> <p>This variable is not required for the Create operation as Nutanix automatically creates a new VLAN. However, this variable is mandatory for the Update and Delete Operations.</p> <p>You can obtain this UUID from the network infrastructure administrator or through network management tools.</p> <p>Ensure that this variable is correctly configured and maintained to avoid any disruption to network connectivity for the Nutanix cluster.</p> <div> <p>Note: This variable is typically used in conjunction with the External Subnet VLAN ID variable to configure external network connectivity for the Nutanix cluster.</p> </div>	Required	String

Variable	Description	Condition	Type
Cluster Name	<p>Used to store the name of the Nutanix cluster that is being managed. It is a critical identifier for the cluster within the Nutanix environment and in external systems that interact with the cluster.</p> <p>This variable must be set to a unique and descriptive name that accurately reflects the purpose and function of the Nutanix cluster.</p> <p>Maintaining this variable is crucial to ensure that the cluster remains identifiable and correctly configured.</p>	Required	String
Enable NAT	<p>Used to specify whether or not the Network Address Translation (NAT) is enabled on the external subnet of a Nutanix cluster.</p> <p>In the context of a Nutanix cluster, enabling NAT on the external subnet allows virtual machines running on the cluster to communicate with external networks using IP addresses that are translated by the Nutanix cluster.</p> <p>This variable can be set to either True or False to enable or disable NAT on the external subnet, respectively. Enabling NAT can be useful in situations where there is a shortage of public IP addresses, or where security policies require that the internal IP addresses are not exposed to the public internet.</p> <div> <p>Note: Carefully consider the implications of enabling or disabling NAT on the external subnet of a Nutanix cluster. This variable must be configured according to the specific requirements of the environment and in line with best practices for networking and security.</p> </div>	Required	String

Variable	Description	Condition	Type
Network IP with Prefix	<p>Used to specify the network IP address and prefix length for the external subnet of a Nutanix cluster.</p> <p>The Nutanix External Subnet Network IP with Prefix variable is typically set to an IP address and prefix length in the CIDR notation format, such as 192.168.0.0/24. This indicates that the network IP address is 192.168.0.0, and that the prefix length is 24 bits.</p> <p>Ensure that the Nutanix External Subnet Network IP with Prefix variable is correctly configured to avoid any conflicts or issues with IP address allocation on the external subnet.</p> <p>The network IP address and prefix length must be chosen based on the size of the external subnet and the number of IP addresses required to support the expected workload of the Nutanix cluster.</p>	Required	String
Gateway IP	<p>Used to specify the IP address of the default gateway for the external subnet of a Nutanix cluster.</p> <p>The Nutanix External Subnet Gateway IP variable is typically set to the IP address of the router that is connected to the external subnet.</p> <p>Ensure that the Nutanix External Subnet Gateway IP variable is correctly configured so that the traffic can be routed to and from the external subnet. The IP address specified in this variable must be reachable from the external subnet and should be valid for the network configuration of the external subnet.</p> <p>If the Nutanix cluster is configured to use multiple external subnets, each subnet must have its own gateway IP address specified in this variable.</p>	Required	String

Variable	Description	Condition	Type
IP Pool	<p>Used to specify a range of IP addresses that can be assigned to virtual machines running on a Nutanix cluster.</p> <p>When a virtual machine is created, it can be assigned an IP address from the IP pool specified in this variable. The Nutanix cluster then reserves the assigned IP address and ensures that it is not assigned to any other virtual machine.</p> <p>Ensure that the Nutanix External Subnet IP Pool variable is correctly configured to avoid any conflicts or issues with IP address allocation on the external subnet.</p> <p>The IP pool must be large enough to support the expected workload of the Nutanix cluster and must not overlap with any other IP address ranges in the environment.</p> <p>The IP Pool variable can be specified as a range of IP addresses in CIDR notation, such as 192.168.0.10-192.168.0.20 or 10.0.0.0/24. Alternatively, it can be specified as a list of individual IP addresses, such as 192.168.0.10, 192.168.0.11, 192.168.0.12.</p>	Required	String
PC Username	<p>Used to set the username that has been granted access to the Prism Central management interface.</p> <p>Ensure that the Nutanix Prism Central Username variable is correctly configured and kept up-to-date to manage the Nutanix clusters effectively.</p> <p>The username specified in this variable must have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

Variable	Description	Condition	Type
PC Password	<p>Used to set the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password variable is kept secure and protected. Nutanix also recommends you to periodically change the password for security reasons.</p>	Required	String

Virtual Private Cloud Management Runbook

Use this runbook to create additional Virtual Private Clouds (VPCs).

Flow Virtual Networking allows you to create completely isolated virtual networks that are separated from the physical network. You can do multitenant network provisioning with overlapping IP addresses, self-service network provisioning, and IP address preservation. Moreover, these isolated virtual networks provide security by default. The VPC is the basic unit of Flow Virtual Networking.

Each VPC is an isolated network namespace with a virtual router instance to connect all of the subnets inside the VPC. This setup allows the IP addresses inside of one VPC to overlap with any other VPC, or even with the physical network.

The VPC Management runbook has the following variables:

Table 45: VPC Management Runbook Variables

Variable	Description	Condition	Type
VPC Name	Represents the name of a Nutanix Virtual Private Cloud (VPC) and is used to identify and reference a specific VPC within a Nutanix environment.	Required	String
Operation	<p>Used to specify the type of operation to perform on a Nutanix VPC.</p> <p>Options include:</p> <ul style="list-style-type: none"> • Create: Set this option to create a new VPC. • Update: Set this option to modify an existing VPC. • Delete: Set this option to remove an existing VPC. 	Required	String

Variable	Description	Condition	Type
VPC UUID	Refers to the unique identifier (UUID) assigned to a Virtual Private Cloud (VPC) within the Nutanix infrastructure. This UUID is a string of characters that serves as a globally unique identifier for the VPC.	Required	String
External Subnet UUID	Refers to the unique identifier assigned to an external subnet within the Nutanix infrastructure. This UUID is a string of characters that serves as a globally unique identifier for the external subnet.	Required	String
Externally Routable IP with prefix	<p>Refers to an IP address range that can be used to route traffic to and from the internet or external networks. It is typically associated with a Virtual Private Cloud (VPC) to enable communication between the VPC and external entities.</p> <p>The externally routable IP with prefix consists of an IP address and a subnet prefix length, usually represented as CIDR notation (e.g., 192.168.0.0/24). The IP address range specified within the prefix is routable on the internet, allowing traffic to flow between the VPC and external networks.</p> <p>When configuring a Nutanix VPC, you can define an externally routable IP range to allocate to the VPC. This IP range is used for assigning IP addresses to the resources within the VPC, such as virtual machines or containers, that require connectivity to external networks.</p> <p>The specific procedure for configuring the externally routable IP with prefix in Nutanix may vary depending on the Nutanix software version and management interface being used. It is typically done during the creation or configuration of the VPC, where you can specify the desired IP range and prefix length for external connectivity.</p>	Required	String

Variable	Description	Condition	Type
DNS Server	<p>Virtual Private Cloud (VPC) can have its own DNS (Domain Name System) servers configured. When setting up a Nutanix VPC, you can configure DNS server IP addresses specific to the VPC to allow resources within the VPC to resolve domain names and access resources using host names.</p> <p>Typically, you provide the IP addresses of the DNS servers during the VPC creation or configuration process. The exact procedure may vary depending on the Nutanix software version and management interface you use.</p> <p>Ensure that the DNS servers configured for the Nutanix VPC are operational and accessible from the VPC resources. You can verify this by performing DNS resolution tests from within the VPC to ensure that domain names are resolved correctly.</p> <div> <p>Note: The actual DNS server IP addresses may vary based on your specific network configuration and requirements within the Nutanix VPC. Nutanix recommends you to refer to the network infrastructure documentation or contact your network administrator to get the correct DNS server IP addresses for your environment.</p> </div>	Required	String
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks. It is important to keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String

Variable	Description	Condition	Type
PC Username	<p>Refers to the username that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>The Nutanix Prism Central Username variable must be set to the username that has been granted access to the Prism Central management interface.</p> <p>Ensure that the Nutanix Prism Central Username variable is correctly configured and kept up-to-date to ensure that the Nutanix clusters can be managed effectively. The username specified in this variable must have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String
PC Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>The Nutanix Prism Central Password variable should be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password variable is kept secure and protected.</p>	Required	String

Overlay Subnet Management Runbook

Use this runbook to create additional overlay subnets in a VPC. The overlay subnet enables VMs to communicate with each other regardless of their physical location or the underlying network infrastructure. It abstracts the underlying network details and provides a virtualized network space for VMs to interact.

Each VPC can have one or more subnets, and these subnets are connected to the same VPC virtual router. In the background, a VPC leverages Geneve encapsulation to tunnel traffic between AHV hosts as required. This means that the subnets inside the VPC do not need to be created or even present on the top-of-rack switches for VMs on different hosts to communicate. When two VMs in a VPC on two different hosts send traffic to each other, packets are encapsulated in Geneve on the first host and sent to the other host where they are decapsulated and sent to the destination VM.

The overlay subnet management runbook has the following variables.

Table 46: Overlay Subnet Management Runbook Variables

Variable	Description	Condition	Type
Management PC Username	Used to specify the username required for authentication with the Nutanix Prism Central management interface. The management username is required to perform administrative tasks on the Nutanix cluster through the Prism Central management interface. It allows access to features such as health monitoring, capacity planning, resource management, and user management.	Required	String
Management PC Password	Used to specify the password used for authentication with the Nutanix Prism Central management interface. The management password is required to perform administrative tasks on the Nutanix cluster through the Prism Central management interface. The variable is used in combination with the management username to authenticate and access features such as health monitoring, capacity planning, resource management, and user management.	Required	String
Subnet Name	Represents the name assigned to a specific overlay subnet within the Nutanix environment. It provides a unique identifier for the overlay subnet.	Required	String

Variable	Description	Condition	Type
Operation	<p>Used to determine the type of operation to perform on a Nutanix overlay subnet. You have the following options:</p> <p>Create: Use this option to create a new Nutanix overlay subnet.</p> <p>Update: Use this option to modify an existing Nutanix overlay subnet.</p> <p>Delete: Use this option to remove an existing Nutanix overlay subnet.</p> <p>Setting the correct value for this variable is crucial as it determines the specific API call to be made to the Nutanix management platform.</p> <div> <p>Note: Certain operations might require additional parameters or configuration options, depending on the specific resource being created, updated, or deleted. Refer to the Nutanix documentation and API reference for detailed requirements of each operation.</p> <p>Overlay subnets play a vital role in managing virtual networks within a Nutanix cluster. Therefore, any modifications made to these subnets can significantly impact the overall cluster performance. To mitigate any potential issues, Nutanix recommends you to follow best practices and thoroughly test any changes in a non-production environment before deploying them in a production environment.</p> </div>	Required	String
VPC Name	Represents the name of a Nutanix Virtual Private Cloud (VPC) and is used to identify and reference a specific VPC within a Nutanix environment.	Required	String

Variable	Description	Condition	Type
Network IP with Prefix	<p>Refers to the IP address range allocated to the subnet along with the associated subnet mask or prefix length. The network IP represents the base address of the subnet while the prefix indicates the number of bits used to define the subnet.</p> <p>For example, if the network IP is 192.168.0.0 and the prefix is /24, it means that the subnet includes IP addresses ranging from 192.168.0.0 to 192.168.0.255 with a subnet mask of 255.255.255.0.</p> <p>The prefix length is represented as the number of consecutive bits set to 1 in the subnet mask. In the above example, /24 indicates that the first 24 bits of the IP address are used to identify the network portion, while the remaining 8 bits are available for host addresses.</p> <p>The network IP with prefix is essential for defining the address space and subnet boundaries within an overlay subnet. It helps in determining the range of available IP addresses and configuring the appropriate network settings for virtual machines, routing, and connectivity within the subnet.</p>	Required	String

Variable	Description	Condition	Type
Gateway IP	<p>Refers to the IP address assigned to the gateway device within the overlay network. The gateway acts as an intermediary between the overlay network and external networks or other subnets.</p> <p>The specific IP address of the gateway depends on the configuration of the overlay network. Typically, the gateway IP address is chosen from within the range of IP addresses assigned to the subnet. It serves as the default gateway for devices within the subnet to communicate with devices outside the subnet or in other subnets.</p> <p>For example, if you have an overlay network with a subnet using the IP address range 192.168.0.0/24, the gateway IP address might be assigned as 192.168.0.1. This means that any device within the subnet would use 192.168.0.1 as the gateway IP to send traffic outside the subnet.</p> <div> <p>Note: The specific configuration of a subnet overlay network, including the choice of gateway IP, can vary depending on the network infrastructure and the technology being used for overlay networking, such as Virtual Extensible LAN (VXLAN) or Generic Routing Encapsulation (GRE).</p> </div>	Required	String

Variable	Description	Condition	Type
IP Pools	<p>Refers to the range of IP addresses used for overlay networking within the Nutanix environment. These IP addresses are assigned to virtual machines (VMs) and services running on the Nutanix cluster.</p> <p>To configure overlay subnet IP pools in Nutanix, follow the following steps.</p> <ol style="list-style-type: none"> 1. Log on to the Nutanix Prism Central web interface. 2. Navigate to the Networking section and locate the ADNE settings. 3. Create a new overlay subnet IP pool or modify an existing one. 4. Specify the IP address range for the pool ensuring that it does not overlap with other networks or subnets in your environment. 5. Define any additional settings or options, such as the subnet mask, gateway IP, DNS servers, etc. 6. Save the configuration. <p>Once the overlay subnet IP pool is configured, Nutanix automatically allocates IP addresses from the pool to the virtual machines and services as they are created within the Nutanix cluster. This allocation allows seamless communication and networking between the VMs and services, while maintaining security and isolation through the overlay network.</p> <div> <p>Note: The exact steps and terminologies might vary depending on the specific version of Nutanix software you are using. Nutanix recommends you to refer to the Nutanix documentation or contact Nutanix support for detailed instructions based on your environment.</p> </div>	Required	String

Variable	Description	Condition	Type
DNS Servers List	<p>Refers to the DNS servers that are configured specifically for the overlay network subnet in a Nutanix environment. These DNS servers are responsible for resolving domain names to IP addresses within the overlay subnet.</p> <p>The exact configuration and IP addresses of the Nutanix overlay subnet DNS servers might vary depending on your specific Nutanix deployment and network setup. To determine the DNS servers configured for the overlay subnet in your Nutanix environment, refer to your network configuration or consult your Nutanix administrator or network team.</p> <p>Typically, the Nutanix overlay subnet DNS servers are set to the IP addresses of the DNS servers provided by your network infrastructure or DNS service provider. These DNS servers enable proper name resolution within the overlay subnet, allowing network resources to be accessed using hostnames.</p> <p>Ensure that the Nutanix overlay subnet DNS servers are properly configured and operational to facilitate smooth network communication and access to resources within the overlay network.</p>	Required	String

Variable	Description	Condition	Type
Domain Name	<p>Refers to the domain names that are typically associated with DNS (Domain Name System) and are used to translate human-readable domain names (such as example.com) into IP addresses. While the Nutanix overlay subnet may utilize DNS servers for name resolution, the subnet itself does not have a dedicated domain name.</p> <p>The domain name configuration within the Nutanix environment is typically managed at a higher level, such as the DNS configuration for the overall network or within specific virtual machines or services running on the Nutanix platform.</p> <p>If you require a domain name for resources within the Nutanix overlay subnet, you need to configure the appropriate DNS settings and assign domain names to individual virtual machines or services within that subnet.</p>	Required	String
Domain Search List	<p>Refers to the configuration setting used by DNS resolvers to search for domain names when a hostname is entered without a fully qualified domain name (FQDN). It allows the resolver to append domain suffixes to the hostname and attempt to resolve the name using different domain names.</p> <p>The domain search list is typically configured at the client level or within the DNS resolver settings of the operating system or network infrastructure. It is not directly tied to the Nutanix overlay subnet itself.</p>	Required	String

Variable	Description	Condition	Type
Boot File Name	<p>Used to configure the boot file name. The boot file name is usually specific to the operating system or boot loader being used and is typically configured within the DHCP server settings. It specifies the file name or path that the client should retrieve from a TFTP server to start the boot process.</p> <p>In a Nutanix environment, the configuration of boot file names is typically handled at the DHCP server level or within the virtual machine configurations. The overlay subnet do not have a specific boot file name associated with it as it is a networking construct rather than a boot configuration.</p> <p>To configure boot file names for virtual machines or services within your Nutanix environment, you need to configure the appropriate DHCP server settings or virtual machine configurations as per your specific requirements.</p>	Required	String
TFTP Server	<p>Used to configure the TFTP server. The configuration and setup of a TFTP server in a Nutanix environment is similar to setting up a TFTP server in any other network environment. Once the TFTP server is set up, you can configure the appropriate network boot settings or firmware update procedures within your Nutanix environment to utilize the TFTP server as needed.</p> <div> <p>Note: Specific steps for setting up a TFTP server might vary depending on your network infrastructure, operating system, and specific requirements. Nutanix recommends you to refer to the documentation of your desired TFTP server software and consider best practices for secure and reliable file transfers within your network environment.</p> </div>	Required	String
Add Subnet to Project	Defines whether the subnet should be added to the project or not.	Required	String

Variable	Description	Condition	Type
Project Name	Used to specify the project name. Projects in Nutanix offer a structured and controlled approach to managing and organizing resources, promoting efficient resource utilization, better security, and simplified administration within your Nutanix environment.	Optional	String
Workload PC IP	Used to specify the IP address or hostname of the Nutanix Prism Central instance that is used for workload management. This variable is required for any operations that involve workload management through Prism Central, such as creating or managing virtual machines, configuring storage policies, and setting up data protection policies.	Required	String
Workload PC Username	Used to specify the username used for authentication with the Nutanix Prism Central instance for workload management. The management username is required for performing administrative tasks related to workload management on the Nutanix clusters through the Prism Central management interface. Once authenticated, administrators can create and manage virtual machines, configure storage policies, and set up data protection policies for their workloads.	Required	String
Workload PC Password	Used to specify the password used for authentication with the Nutanix Prism Central instance for workload management. The management password is required for performing administrative tasks related to workload management on the Nutanix clusters through the Prism Central management interface. This variable is used in combination with the management username to authenticate and access features such as creating and managing virtual machines, configuring storage policies, and setting up data protection policies.	Required	String

Users and Groups Management Runbook

The users and groups management runbook helps in managing users in a project. With this runbook, you can add, update, or remove users from a project.

The users and groups management runbook has the following variables.

Table 47: Users and Groups Management Runbook Variables

Variable	Description	Condition	Type
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources. You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Ensure that you keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String
Project Name	<p>Refers to the name given to a specific project or initiative for creating an application within the Nutanix environment. The project name represents a unique identifier or label for the application development project.</p> <p>The specific Project Name depends on the nature of the application being developed, the organization's naming conventions, and any relevant naming guidelines or standards in place.</p>	Required	String

Variable	Description	Condition	Type
Authentication Type [Used in Project]	<p>Refers to the authentication type used for project user accounts. This authentication type depends on the specific authentication mechanisms and settings configured within the environment. Nutanix supports multiple authentication types that can be used for project users, such as:</p> <ul style="list-style-type: none"> • Local Authentication: Nutanix Prism provides a built-in local authentication mechanism where project users can be created and authenticated directly within the Nutanix cluster. With local authentication, user accounts and passwords are managed within the Nutanix environment itself. • Active Directory (AD) Authentication: Nutanix also supports integrating with an external Active Directory (AD) domain for user authentication. This integration allows project users to authenticate using their AD credentials, providing a centralized and unified authentication mechanism. • Lightweight Directory Access Protocol (LDAP) Authentication: Nutanix can also be configured to use LDAP servers for project user authentication. LDAP authentication enables authentication against a directory service that follows the LDAP protocol, such as Microsoft Active Directory or OpenLDAP. • Security Assertion Markup Language (SAML) Authentication: SAML authentication enables integration with identity providers (IDPs) that support the SAML protocol. Project users can authenticate using their credentials from the configured IDP, which can be an external SAML-based identity provider. 	Required	String

Variable	Description	Condition	Type
IDP Name	Refers to the name of an Identity Provider in the context of authentication and user management. This variable is used to represent the specific name or identifier associated with the IDP used for authentication in your system.	Required when you do not use the Active Directory Domain Name variable	String
Active Directory Domain Name	Refers to the fully qualified domain name (FQDN) of an Active Directory domain in a Windows environment. It is the unique name that identifies the specific Active Directory domain within a network.	Required when you do not use the IDP Name variable	String
Operation	Used to represent the type of operation being performed on user accounts, such as adding, updating, or deleting users. This variable helps determine the specific action to be taken when managing user accounts within a system.	Required	Integer

Variable	Description	Condition	Type
Admin Users	<p>Refers to the specific user accounts with administrative privileges that are associated with a particular project or tenant within the Nutanix environment. These users have elevated permissions within the scope of the assigned project and can perform administrative tasks related to that specific project.</p> <p>Key points about Nutanix Project Admin Users are:</p> <ul style="list-style-type: none"> • Scope: Project Admin Users have administrative privileges limited to the specific project or tenant they are associated with. Their administrative actions and settings affect only the resources within that project. • Project-Level Control: These users can create, modify, and delete resources within their project, such as virtual machines, storage containers, and network configurations. • User Management: These users have the authority to manage user accounts and assign roles and permissions to other users within their project. They can control the level of access and actions that other users can perform within the project. • Collaboration: These users can collaborate with other users within the project, sharing resources, configuring access controls, and working together to achieve project goals. <p>Note: The exact configuration and management of these users may vary depending on the version of Nutanix software being used and the specific setup of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Developer Users	<p>Refers to the user accounts with specific permissions and access rights configured towards development and application-related tasks. These users can create, deploy, and manage applications within the Nutanix environment.</p> <p>Key points about Developer Users in Nutanix are:</p> <ul style="list-style-type: none"> • Application Deployment: These users can deploy applications within the Nutanix infrastructure. They can create and manage virtual machines, containers, and other resources required for application deployment. • Resource Management: They can manage the resources allocated to their applications, including storage, networking, and compute resources. They can scale their applications based on demand and optimize resource utilization. • Monitoring and Troubleshooting: They can monitor the performance and health of their applications, leveraging Nutanix monitoring and logging features. • Collaboration: They can collaborate with other team members to coordinate application deployments, troubleshoot issues, and share resources and knowledge. <p>Note: The exact user roles, permissions, and capabilities may vary depending on the version of Nutanix software being used and the specific configuration of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Consumer Users	<p>Refers to the user accounts that have limited access and permissions within the Nutanix environment. These users are typically granted access to specific resources or services based on their role or business needs.</p> <p>Key points about Consumer Users in Nutanix are:</p> <ul style="list-style-type: none"> • Limited Access: Consumer Users have restricted access to the Nutanix infrastructure and are only granted access to the resources or services that are necessary for their specific role or requirements. • Resource Consumption: Consumer Users primarily consume the services and resources provided by Nutanix, such as virtual machines, applications, or data stored within the environment. • Role-Based Access: The permissions and access rights of Consumer Users are typically defined through role-based access control (RBAC). Their level of access and the specific actions they can perform are determined by their assigned role or user profile. • Self-Service Provisioning: In some cases, Consumer Users may have the ability to self-provision or request specific resources within predefined limits. <p>Note: The exact configuration and management of Consumer Users may vary depending on the version of Nutanix software being used and the specific setup of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Operator Users	<p>Refers to the user accounts with specific permissions and access rights that are focused on performing day-to-day administrative tasks, monitoring system health, and ensuring the smooth operation of the Nutanix cluster.</p> <p>Key points about these users in Nutanix are:</p> <ul style="list-style-type: none"> • Infrastructure Management: These users have the necessary privileges to manage and operate the Nutanix infrastructure. They can perform tasks such as configuring storage, managing networking, and ensuring high availability. • Cluster Monitoring: These users are responsible for monitoring the performance and health of the Nutanix cluster. • Troubleshooting and Maintenance: These users have the ability to troubleshoot and resolve issues related to the Nutanix infrastructure. • Resource Allocation and Optimization: These users can manage the allocation and optimization of resources within the Nutanix cluster. • Collaboration with Administrators: These users work closely with administrators and system engineers to coordinate infrastructure management activities. <p>Note: The exact user roles, permissions, and capabilities may vary depending on the version of Nutanix software being used and the specific configuration of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Admin Groups	<p>Refers to the groups of users with administrative privileges that are associated with a specific project or tenant within the Nutanix environment. These groups provide a convenient way to manage and assign administrative roles and permissions to multiple users within a project.</p> <p>Key points about these groups in Nutanix are:</p> <ul style="list-style-type: none"> • Scope: These groups have administrative privileges limited to the specific project or tenant they are associated with. The users in these groups have administrative control over the resources and settings within their assigned project. • Role-Based Access Control: Nutanix uses role-based access control (RBAC) to define and manage user roles and permissions. • Group Management: Project Admin Groups can be created, modified, and deleted within the Nutanix environment. • Role Assignment: Once a Project Admin Group is created, the desired administrative role can be assigned to the group. This role determines the level of access and the specific actions that the users within the group can perform within the project. • Collaboration: Project Admin Groups enable effective collaboration among project administrators. <p>Note: The exact configuration and management of Project Admin Groups may vary depending on the version of Nutanix software being used and the specific setup of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Developer Groups	<p>Refers to the groups of users that are created and managed to group developers together and provide them with the necessary permissions and resources to carry out their development tasks effectively.</p> <p>Key points about these groups in Nutanix are:</p> <ul style="list-style-type: none"> • Collaboration and Teamwork: Developer Groups allow developers to be organized and collaborate more efficiently. By grouping developers together, it becomes easier to assign permissions and coordinate development efforts within the Nutanix environment. • Resource Allocation: Developer Groups can be granted access to specific resources and services within the Nutanix infrastructure. This includes virtual machines, containers, storage, and other resources necessary for development purposes. • Simplified Access Management: Instead of individually managing permissions for each developer, Developer Groups allow for easier management of access control. Permissions can be assigned or revoked at the group level. • Project-specific Groups: Developer Groups can be created and associated with specific projects or development initiatives. This allows developers to work within the context of their projects. <p>Note: The exact configuration and management of Developer Groups may vary depending on the version of Nutanix software being used and the specific setup of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Operator Groups	<p>Refers to the groups of users that are created and managed to group operators together and provide them with the necessary permissions and resources to carry out their operational tasks effectively.</p> <p>Key points about Operator Groups in Nutanix are:</p> <ul style="list-style-type: none"> • Collaboration and Teamwork: Operator Groups allow operators to be organized and collaborate more efficiently. By grouping operators together, it becomes easier to assign permissions and coordinate operational tasks within the Nutanix environment. • Infrastructure Management: Operator Groups are granted permissions to manage and operate the Nutanix infrastructure. This includes tasks such as configuring storage, managing networking, and ensuring high availability. • Resource Allocation: Operator Groups can be granted access to specific resources and services within the Nutanix infrastructure. This may include managing virtual machines, storage containers, and other resources necessary for operational tasks. • Simplified Access Management: Instead of individually managing permissions for each operator, Operator Groups allow for easier management of access control. <p>Note: The exact configuration and management of Operator Groups may vary depending on the version of Nutanix software being used and the specific setup of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Consumer Groups	<p>Refers to the groups of users that are created and managed to group consumers together and provide them with the necessary permissions and resources to consume services and resources effectively.</p> <p>Key points about these groups in Nutanix are:</p> <ul style="list-style-type: none"> • Collaboration and Teamwork: Consumer Groups allow consumers to be organized and collaborate more efficiently. By grouping consumers together, it becomes easier to assign permissions and coordinate consumption of services within the Nutanix environment. • Resource Consumption: Consumer Groups are granted permissions to consume services and resources within the Nutanix infrastructure. This includes accessing virtual machines, applications, data, and other resources necessary for their specific requirements. • Simplified Access Management: Instead of individually managing permissions for each consumer, Consumer Groups allow for easier management of access control. • Project-specific Groups: Consumer Groups can be created and associated with specific projects or initiatives. This allows consumers to work within the context of their projects, focusing on their specific consumption requirements while collaborating with other team members. <p>Note: The exact configuration and management of Consumer Groups may vary depending on the version of Nutanix software being used and the specific setup of your Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Project User Username	<p>Refers to the username or login name associated with a user account that has administrative privileges within a specific project or tenant. This username is used to authenticate and identify the user when performing administrative tasks and managing resources within the project.</p> <p>This username is typically unique to each administrator and is chosen during the user account creation process. It is used in combination with a corresponding password or other authentication credentials to authenticate the user and grant them administrative access to the resources and services within the Nutanix project.</p>	Required	String
Project User Password	<p>Refers to the password associated with a user account that has administrative privileges within a specific project or tenant. This password is used to authenticate and identify the user when performing administrative tasks and managing resources within the project.</p>	Required	String

VPC Static Routing Runbook

Use this runbook to create static routes for VPC. Every VPC contains a single virtual router, and there are a few types of routes.

- External networks are the default destination of the 0.0.0.0/0 network prefix for the whole VPC. You can choose an alternate network prefix route for each external network in use. For completely isolated VPCs, you may choose not to set a default route.
- Directly connected routes are created for each subnet inside the VPC. Flow Virtual Networking assigns the first IP address of each subnet as the default gateway for that subnet. The default gateway and network prefix are determined by the subnet configuration and cannot be altered directly. Traffic between two VMs on the same host and same VPC, but in two different subnets, will be routed locally in that host.
- Remote connections such as VPN connections and External Networks can be set as the next hop destination for a network prefix.

The VPC static routing runbook has the following variables:

Table 48: VPC Static Routing Runbook Variables

Variable	Description	Condition	Type
Operation	<p>Refers to the action that you want to perform on a static route, such as updating or deleting it.</p> <p>Update: Indicates that you want to modify the configuration of an existing static route. The update can involve changing the next hop IP address, modifying the destination IP range, adjusting the metric or priority, or making any other necessary changes to the static route configuration.</p> <p>Delete: Indicates that you want to remove or delete an existing static route from the routing table. This action removes the entry for the static route, effectively stopping the traffic from being routed according to that specific route.</p>	Required	String
VPC Name	<p>Refers to the name or identifier of the Virtual Private Cloud (VPC) in which the static routing configuration is being applied.</p> <p>When configuring static routes in Nutanix, you associate the static routes with a specific VPC by specifying the VPC Name. This ensures that the static routes are applied within the context of the designated VPC.</p> <p>By providing the VPC Name in Nutanix static routing, you can define the scope of the routing configuration and ensure that the static routes are applied to the correct VPC within your cloud environment. This allows you to establish routing between different subnets or networks within the specified VPC, enabling seamless communication between resources in your network infrastructure</p>	Required	String

Variable	Description	Condition	Type
IP with Prefix [Destination]	<p>Refers to the specific IP address prefix or subnet range that is used to define the destination IP addresses for the static route.</p> <p>When configuring a static route, you can specify an IP address prefix or a subnet range (in CIDR notation) that represents the destination network or destination IP addresses for the route. This allows you to define the target network or specific range of destination IP addresses to which the static route applies.</p> <p>For example, if you define the IP with Prefix [Destination] as 192.168.0.0/24, it implies that the static route is targeting the subnet range from 192.168.0.0 to 192.168.0.255. This indicates that any traffic with a destination IP address within this range must follow the configured static route.</p> <p>By specifying the IP with Prefix [Destination] in Nutanix static routing, you can control the flow of network traffic based on the destination IP address. This allows you to direct traffic to specific destination networks or IP address ranges, ensuring that it reaches the intended destination through the defined static route.</p>	Required	String

Variable	Description	Condition	Type
External Subnet Name	<p>Refers to the name or identifier of the external subnet that is used in the static routing configuration.</p> <p>An external subnet typically represents a subnet or network segment that is outside of your Nutanix infrastructure. It could be a subnet in a different cloud provider network, a remote on-premises network, or any other external network that you need to establish connectivity with.</p> <p>When configuring static routing in Nutanix, you can specify this variable to identify the specific subnet or network segment that the static route is targeting for communication. This helps in defining the routing path and establishing connectivity between your Nutanix infrastructure and the external subnet.</p> <p>By specifying this variable in Nutanix static routing, you can ensure that the static routes are correctly associated with the desired external subnet, allowing traffic to flow between your Nutanix environment and the specified external network segment.</p>	Required	Integer
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>You must keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String

Variable	Description	Condition	Type
Prism Central Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password variable is kept secure and protected.</p>	Required	String
Prism Central Username	<p>Used to set the username that has been granted access to the Prism Central management interface.</p> <p>Ensure that the Nutanix Prism Central Username variable is correctly configured and kept up-to-date so that the Nutanix clusters can be managed effectively.</p> <p>The username specified in this variable should have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

VLAN Subnet Management Runbook

Use this runbook to create additional VLAN subnets. The VLAN subnet management runbook has the following variables.

Table 49: VLAN Subnet Management Runbook Variables

Variable	Description	Condition	Type
Subnet Name	<p>Refers to a virtual local area network (VLAN) subnet that is created and managed within a Nutanix cluster. It allows multiple networks to coexist on the same physical network, and can be used to segregate traffic based on specific requirements, such as security or performance.</p>	Required	String

Variable	Description	Condition	Type
VLAN ID	Refers to the numerical identifier assigned to a virtual local area network (VLAN) subnet within a larger network. It is used to differentiate traffic on the network and enable better network management, particularly in larger environments where multiple VLANs are in use.	Required	Integer
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String
Operation	<p>Used to determine the type of operation to perform on a Nutanix VLAN subnet. You have the following options for this variable.</p> <ul style="list-style-type: none"> • Create: Use this option to create a new Nutanix VLAN subnet. • Update: Use this option to modify an existing Nutanix VLAN subnet. • Delete: Use this option to remove an existing Nutanix VLAN subnet. <p>Note: Nutanix VLAN subnets play a crucial role in managing virtual networks within a Nutanix cluster. Therefore, any modifications made to these subnets can have an impact on the overall performance of the cluster. To avoid any potential issues, Nutanix recommends you to adhere to best practices and thoroughly test any changes in a non-production environment before deploying them in a production environment.</p>	Required	String

Variable	Description	Condition	Type
Cluster Name	<p>Used to store the name of the Nutanix cluster that is being managed. It is a critical identifier for the cluster within the Nutanix environment and in external systems that interact with the cluster.</p> <p>This variable must be set to a unique and descriptive name that accurately reflects the purpose and function of the Nutanix cluster.</p> <p>Maintaining this variable is crucial to ensure that the cluster remains identifiable and correctly configured.</p>	Required	String
Virtual Switch Name	Used to store the name of the virtual switch associated with the VLAN subnet in the Nutanix environment. This virtual switch is used to provide connectivity between virtual machines and other resources within the Nutanix cluster.	Required	String
VLAN UUID	<p>Used to store the unique identifier (UUID) of the VLAN associated with the subnet being managed in a Nutanix cluster. The VLAN UUID is used to uniquely identify the VLAN and associate it with the corresponding subnet.</p> <p>Ensure that the correct VLAN UUID is set when creating, updating, or deleting a VLAN subnet to avoid any conflicts or unintended changes to the Nutanix environment.</p>	Required	String

Variable	Description	Condition	Type
Network IP with Prefix	<p>Used to identify and segment the network traffic within the Nutanix environment. It consists of the network IP address and the corresponding prefix.</p> <p>For example, Network IP: 10.0.0.0 and Prefix: 24. The network IP and prefix together define the range of IP addresses available within the Nutanix VLAN network. In this case, the network IP is 10.0.0.0 and the prefix length is 24, which means that the network has `256` available IP addresses.</p> <p>Ensure that you configure your Nutanix VLAN network devices and systems with the appropriate IP addresses and subnet masks based on the network IP and prefix specified above.</p>	Required	String
Gateway IP	<p>Used to specify the IP address assigned to the default gateway within the VLAN network. It serves as the entry point for network traffic between the Nutanix cluster and external networks. For example, Gateway IP: 10.0.0.1. The gateway IP in this case must be configured as the default gateway for devices within the Nutanix VLAN network. This ensures proper routing of network traffic to and from the Nutanix cluster.</p> <p>You must configure the network devices and systems within the Nutanix VLAN network to use the correct gateway IP address. This allows for seamless communication between the Nutanix infrastructure and other networks or devices.</p>	Required	String

Variable	Description	Condition	Type
IP Pools Range	<p>Used to define the range of IP addresses available for assignment to virtual machines (VMs) and other network entities within the VLAN network. It helps in managing and allocating IP addresses efficiently. For example, Start IP: 10.0.0.10 and End IP: 10.0.0.50. The IP pools range in this case encompasses a total of 41 IP addresses. These addresses can be dynamically assigned to VMs and other network resources within the Nutanix VLAN network.</p> <p>When provisioning new VMs or allocating IP addresses for other network entities, ensure that the assigned IP addresses fall within this specified IP pools range. This helps prevent conflicts and ensures proper IP address management within the Nutanix environment.</p> <p>Nutanix recommends you to regularly monitor and update the IP pools range as per the network requirements to accommodate the growing needs of the Nutanix VLAN network.</p>	Required	String

Variable	Description	Condition	Type
TFTP Server	<p>Refers to the Nutanix VLAN TFTP server that is used for network booting and firmware upgrades within the VLAN network. It allows for easy file transfer and deployment of operating system images, configuration files, and firmware updates to network devices.</p> <p>Ensure that the TFTP server IP address and port are properly configured on the target devices to enable successful file transfers and network booting operations.</p> <p>Additionally, ensure that the necessary files, such as operating system images or firmware updates, are available on the TFTP server and accessible to the devices in the VLAN network.</p> <div> <p>Note: TFTP is a lightweight file transfer protocol commonly used for network booting and firmware updates. It operates over UDP (User Datagram Protocol) and does not provide encryption or authentication. Exercise caution when using TFTP for transferring sensitive or critical files.</p> </div>	Required	String

Variable	Description	Condition	Type
Boot File Name	<p>Used to specify the file that network devices within the VLAN network must retrieve and execute during the boot process. It is used for network booting and provisioning operating systems or other bootable images.</p> <p>The specified boot file name must be accessible on the designated TFTP server or boot server within the VLAN network. It contains the necessary instructions and configuration for network booting and subsequent system provisioning.</p> <p>Ensure that the boot file is correctly configured and available on the TFTP server, and that network devices are configured to request and retrieve the correct boot file during the boot process.</p> <div> <p>Note: The boot file name may vary depending on the specific network boot infrastructure and configuration used within the Nutanix VLAN network.</p> </div>	Required	String

Variable	Description	Condition	Type
Domain Search List	<p>Used to specify the domain names that network devices within the VLAN subnet must search when resolving hostnames. It helps streamline DNS resolution and simplifies the process of accessing resources within the VLAN network.</p> <p>The specified domain search list must be configured on the DNS settings of the network devices within the VLAN subnet. It allows these devices to search the specified domain names when attempting to resolve hostnames.</p> <p>Ensure that the domain names listed in the search list are valid and correspond to the appropriate DNS infrastructure within the Nutanix VLAN subnet.</p> <p>By configuring the domain search list, users and applications within the VLAN subnet can conveniently access resources using simplified hostnames without specifying the full domain name.</p> <p>Note: The actual domain names in the search list may vary based on your specific network configuration and requirements within the Nutanix VLAN subnet.</p>	Required	String
Domain Name	<p>Used to define the domain for the network devices within the VLAN subnet. It allows for logical grouping of devices and simplifies the management of DNS records and hostnames within the subnet.</p> <p>Ensure that the specified domain name accurately represents the intended domain for the VLAN subnet. This domain name is used for DNS resolution and hostname assignment within the subnet.</p> <p>Note: The actual domain name may vary based on your specific network configuration and requirements within the Nutanix VLAN subnet.</p>	Required	String

Variable	Description	Condition	Type
DNS Servers IP	<p>Refers to the Nutanix VLAN subnet DNS servers that are responsible for resolving domain names to IP addresses within the VLAN subnet. They play a crucial role in enabling proper network communication and access to resources.</p> <p>Ensure that you enter the correct IP addresses of the DNS servers provided by your network infrastructure. These DNS servers handle the resolution of domain names and allow network devices within the VLAN subnet to access resources using hostnames.</p> <p>Verify that the DNS servers are operational and accessible from devices within the VLAN subnet to ensure reliable DNS resolution.</p> <div> Note: The actual IP addresses of the DNS servers may vary based on your specific network configuration and requirements within the Nutanix VLAN subnet. </div>	Required	String
PC Username	<p>Used to set the username that has been granted access to the Prism Central management interface.</p> <p>Ensure that the Nutanix Prism Central Username variable is correctly configured and kept up-to-date to manage the Nutanix clusters effectively.</p> <p>The username specified in this variable must have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String
PC Password	<p>Used to set the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password variable is kept secure and protected. Nutanix also recommends you to periodically change the password for security reasons.</p>	Required	String

Disaster Recovery Runbook

Use this runbook to create a policy-based DR. The policy-based DR and runbooks extend upon the capabilities defined in a VM-based DR and abstracts things into a policy-driven model. The policy-based

DR simplifies configuration by focusing on the items of interest (for example recovery point objective (RPO), retention, and so on.) and assigning to categories instead of the VMs directly. The policy-based DR also allows for a default policy that can apply to all VMs.

The Disaster Recovery (formerly Leap) runbook has the following variables:

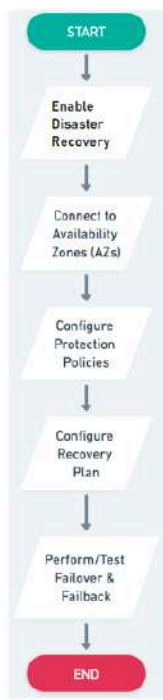


Figure 85: Disaster Recovery (formerly Leap) runbook

Table 50: Disaster Recovery Runbook Variables

Variable	Description	Condition	Type
Protection Policy Name	<p>Refers to the user-defined name given to a policy to determine how often and for how long backups or snapshots are taken of a specific object in the Nutanix cluster. This object can be a virtual machine, a container, or any other resource that requires protection against data loss.</p> <p>The policy can be configured with various options, such as backup frequency, retention period, backup schedule, compression, and encryption settings.</p> <p>This parameter is used to specify which protection policy to apply when creating or managing data protection policies for objects in the Nutanix cluster. The policy helps manage backup storage space, based on pre-defined recovery window goals.</p>	Required	String
Recovery Plan Name	<p>Refers to a user-defined name given to a recovery plan to specify the steps required to recover a particular object or service in the event of a disaster or system failure.</p> <p>The recovery plan can be set up with various options, such as the order in which services or virtual machines should be recovered, the specific recovery point to use, and the notification settings for administrators.</p> <p>The Nutanix Recovery Plan Name is used as a parameter to specify which recovery plan to apply when recovering objects or services in the Nutanix cluster.</p>	Required	String

Variable	Description	Condition	Type
Custom RPO Interval for Replication in Hours	<p>Refers to the Recovery Point Objective (RPO) interval in hours for replication of data between Nutanix clusters. The RPO interval determines how frequently the replicated data is synchronized between the source and target clusters. A smaller RPO interval means that the data is synchronized more frequently, resulting in less data loss in case of a disaster.</p> <p>The value of this parameter must be chosen based on factors such as the criticality of the data, the frequency of changes, and the available network bandwidth for replication.</p>	Required	Integer
Local Schedule	<p>Refers to the schedule for taking local backups or snapshots of a specified object, such as a virtual machine or container, within a Nutanix cluster.</p> <p>The local schedule includes settings such as the frequency of backups or snapshots, the time of day when the backups or snapshots are taken, and any additional options such as retention policies or data compression settings.</p> <p>The Local RPO setting in the schedule specifies the Recovery Point Objective, which is the maximum amount of data loss that is acceptable in the event of a disaster or outage.</p> <p>This schedule is used to keep snapshots of the VM locally, allowing for quick recovery in case of data loss or corruption.</p>	Required	String

Variable	Description	Condition	Type
Custom RPO Interval for Local Snapshot in Hours	<p>Used to define the custom RPO (Recovery Point Objective) interval in hours for local snapshots of virtual machines in a Nutanix cluster. RPO is the amount of data that an organization is willing to lose in the event of a disaster or data loss.</p> <p>The custom RPO interval determines how frequently local snapshots should be taken for the specified virtual machines, ensuring that the data is protected and the RPO objective is met.</p> <p>Note: The default value for this parameter is set to 1, and it should be greater than or equal to 1 when the local schedule is enabled.</p>	Required	Integer
Retention on Primary and Remote	<p>Refers to the duration for which backup data is kept on the primary and remote sites. In a Nutanix cluster, backups can be stored locally on the primary site or remotely on a secondary site or cloud. The retention period for each backup copy determines how long the backup data will be kept and available for restore operations.</p> <p>The retention period can be set as a fixed number of days or based on the number of available snapshots. The retention on primary and remote can be set differently depending on the organizational data protection and compliance requirements.</p>	Required	Integer
Protection Start Time	<p>Refers to the time at which the initial snapshot for the protected object is taken. This parameter can be set to start immediately or scheduled to start at a specific time. If scheduled, the protection start time can be set according to the organizational requirements for data protection and recovery objectives.</p> <p>The start time can be specified using a 24-hour clock format and must be set to a time that is in the future.</p> <p>Ensure that the protection start time is set appropriately to avoid any gaps in data protection or recovery.</p>	Required	String

Variable	Description	Condition	Type
Primary Account Cluster Name	<p>Refers to the name of the primary Nutanix cluster that is used for disaster recovery and backup purposes. This cluster serves as the primary destination for backups and replication of data to ensure business continuity in case of any disaster or data loss event.</p> <p>The primary account cluster name is an essential parameter that must be specified when configuring data protection policies, disaster recovery plans, and backup schedules for virtual machines and other objects in the Nutanix cluster.</p>	Required	String
Primary PC IP	<p>Refers to the IP address of the primary Prism Central instance that is used for managing backup and disaster recovery operations in a Nutanix cluster. The primary Prism Central instance serves as the central point of control for data protection policies, disaster recovery plans, and other management tasks related to backup and recovery.</p> <p>This parameter is used to specify the IP address of the primary instance, which allows the backup and disaster recovery tools to connect to the instance and perform the necessary operations.</p>	Required	String
Primary PC Username	<p>Refers to the username of the account that is used to authenticate with the primary Prism Central instance in a Nutanix cluster during disaster recovery operations. This account must have sufficient privileges to perform the required disaster recovery tasks, such as restoring data from backups or initiating failover operations to a remote site.</p> <p>The username is typically an administrative account that is created specifically for disaster recovery purposes and is separate from the regular user accounts used for day-to-day operations in the Nutanix cluster. The Primary Prism Central Username for Disaster Recovery parameter is used by the deployment tool to authenticate with the primary instance during disaster recovery operations.</p>	Required	String

Variable	Description	Condition	Type
Primary PC Password	Refers to the password used to authenticate with the primary Prism Central instance in a Nutanix cluster for disaster recovery and backup purposes. The primary Prism Central instance is the main instance that is used for managing and monitoring the Nutanix cluster, and it serves as the central point of control for data protection policies, disaster recovery, and other management tasks.	Required	String
DR Account Cluster Name	Refers to the name of the Nutanix cluster that is designated as the Disaster Recovery (DR) site. In a Nutanix environment, organizations can set up a secondary cluster at a separate physical location to serve as a DR site, which can be used for data protection and disaster recovery purposes. The DR Account Cluster Name parameter is used to specify the name of the DR cluster, which allows the deployment tool to connect to the DR site and perform management tasks as required.	Required	String
DR PC IP	Refers to the IP address of the disaster recovery (DR) Prism Central instance in a Nutanix cluster. In the event of a disaster or outage, the DR instance can be used to failover critical workloads and data to a secondary site, ensuring business continuity and minimizing downtime. The DR Prism Central instance is typically located at a secondary site and is configured to replicate data from the primary site on a regular basis, allowing for quick and seamless failover in the event of an outage.	Required	String
DR PC Username	Refers to the username used to authenticate with the disaster recovery Prism Central instance in a Nutanix cluster. This username is typically associated with an account that has administrative privileges, allowing the deployment tool to perform management tasks as needed on the disaster recovery site.	Required	String

Variable	Description	Condition	Type
DR PC Password	<p>Refers to the password used to authenticate with the disaster recovery Prism Central instance in a Nutanix cluster. This password is used in conjunction with the associated username to authenticate and gain access to the disaster recovery site for management tasks, data protection policies, and other administrative functions.</p> <p>It is important to ensure that this password is securely managed and kept confidential to maintain the security and integrity of the Nutanix cluster.</p>	Required	String
VM Category for Protection Policy and Recovery Plan	<p>Refers to a category or group of virtual machines within a Nutanix cluster that share a common set of data protection and recovery requirements. The category is defined by the administrator based on factors such as the criticality of the virtual machines, the type of data they contain, and the recovery objectives for the organization.</p> <p>Protection policies are applied to VM categories to define the data protection requirements for the virtual machines in the category. These policies typically include settings such as the frequency of backups, retention policies, and any specific backup options such as compression or encryption.</p> <p>Recovery plans are also associated with VM categories, and they define the recovery objectives and procedures for the virtual machines in the category in case of a disaster or data loss event. The recovery plan typically includes steps for restoring the virtual machines from backups, testing the recovery process, and verifying the recoverability of the data.</p> <p>By categorizing virtual machines based on their data protection and recovery requirements, administrators can easily manage and apply consistent policies and procedures to ensure that critical data is protected and recoverable in case of any data loss events.</p>	Required	String

Variable	Description	Condition	Type
Recovery Plan Network Type	<p>Refers to the type of network used for replication between the primary and recovery sites in a disaster recovery scenario. You have the following options for this parameter.</p> <ul style="list-style-type: none"> • Stretched: In a stretched network, the primary and recovery sites are in the same Layer 2 network domain, which allows for seamless failover and failback operations. This network type is typically used when the primary and recovery sites are in close proximity to each other. • Non-stretched: In a non-stretched network, the primary and recovery sites are in separate Layer 2 network domains, which requires additional configuration for replication and failover. This network type is typically used when the primary and recovery sites are geographically separated and cannot be in the same Layer 2 network domain. 	Required	String
Stage Delay [In Seconds]	<p>Refers to the amount of time that is added to the recovery plan execution time for each stage in the plan. The stage delay can be used to introduce a delay between stages of the recovery plan, allowing administrators to verify that each stage has completed successfully before proceeding to the next stage.</p> <p>This delay can be used to ensure that each stage has completed successfully and to provide time for any necessary troubleshooting or remediation.</p> <p>The stage delay parameter can be configured according to the organizational recovery objectives and requirements for disaster recovery.</p>	Required	Integer

Variable	Description	Condition	Type
Enable Boot Script	<p>Refers to the boolean variable that specifies whether or not to enable the execution of a boot script during the disaster recovery process. Enabling the boot script during the disaster recovery process can help to ensure that the virtual machine is configured correctly and that it is ready to run the necessary applications and services after the recovery process is complete. If the parameter is set to true, the boot script is executed during the recovery process. If it is set to false, the boot script is not executed.</p> <p>The Nutanix cluster has the following boot scripts available for virtual machine recovery:</p> <ul style="list-style-type: none"> For Linux: <p>Production: /usr/local/sbin/ production_vm_recovery</p> <p>Test: /usr/local/sbin/ test_vm_recovery</p> For Windows: <p>Production: (Relative to Nutanix directory in Program Files)/scripts/ production/vm_recovery.bat</p> <p>Test: (Relative to Nutanix directory in Program Files)/scripts/test/ vm_recovery.bat</p> <p>The specific location of the Nutanix directory may vary depending on the installation configuration. These boot scripts can be used to automate the recovery process for virtual machines during disaster recovery scenarios.</p>	Required	String

Variable	Description	Condition	Type
Primary Network Name - Production Subnet	<p>Refers to the name of the primary network used for production workloads in a Nutanix cluster. This parameter is used to identify the network that is used by virtual machines running production workloads, such as web servers, databases, and other critical applications. By specifying the name of the primary network, administrators can ensure that data protection and disaster recovery policies are applied to the correct network and workloads.</p> <p>The primary network is typically configured to provide high-speed connectivity and low latency for production workloads, and it may be segregated from other networks used for backup or management purposes.</p>	Required	String
Primary Network Name - Test Subnet	<p>Refers to the name of the network that is associated with the test subnet in a Nutanix cluster.</p> <p>The test subnet is a separate network segment that is used for testing and development purposes, and it is typically isolated from the production network to prevent any interference or impact on live systems.</p> <p>The primary network name parameter is used to specify the name of the test subnet network, which allows the deployment tool to configure network settings and policies as required.</p>	Required	String

Variable	Description	Condition	Type
DR Network Name - Production Subnet	<p>Refers to the name of the production subnet in the disaster recovery (DR) site of a Nutanix cluster. The DR site is a secondary site where data and applications can be replicated and recovered in case of a disaster or disruption at the primary site. The production subnet is the network segment in the DR site where the recovered virtual machines (VMs) are deployed and run after a failover.</p> <p>The DR Network Name - Production Subnet parameter is used to specify the name of this subnet, which is needed for configuring the network settings of the recovered VMs and ensuring they are properly connected to the network in the DR site.</p>	Required	String
DR Network Name - Test Subnet	<p>Refers to the name of the test subnet in the disaster recovery site of a Nutanix cluster. This parameter is used in the recovery plan to specify the network configuration for the test environment. The test subnet is a separate network segment that is used for testing purposes and is isolated from the production environment. It allows organizations to test their disaster recovery procedures without impacting their production environment.</p> <p>By specifying the DR Network Name - Test Subnet parameter, administrators can configure the network settings for the test environment to ensure that it is properly connected and can communicate with the necessary resources.</p>	Required	String

Variable	Description	Condition	Type
Enable Static IP Mapping	<p>Refers to a feature that enables administrators to map the IP addresses of protected virtual machines in the primary site to specific IP addresses in the disaster recovery site. This feature is useful when the IP addresses of virtual machines in the primary site are different from the IP addresses in the disaster recovery site, which can cause issues with connectivity and application functionality after a failover event.</p> <p>When this parameter is enabled, administrators can create static IP mappings for protected virtual machines in the primary site, which is used to assign specific IP addresses to the virtual machines in the disaster recovery site during a failover event. This ensures that applications and services continue to function as expected after a failover, even if the IP addresses of the virtual machines have changed.</p> <div> Note: The Enable Static IP Mapping feature is optional and may not be required in all scenarios. This feature may not be necessary if the IP addresses of virtual machines in the primary and disaster recovery sites are the same. </div>	Required	String
VM Name	To enable static IP mapping for a VM in Nutanix, you need to provide the name of the VM for which you want to enable the static IP mapping in the list of VMs.	Optional	String
Primary Network Prod Static IP	<p>Used to specify a static IP address for a protected VM in the primary site's production network in Nutanix. When disaster recovery is activated and the VM is failed over to the secondary site, the static IP mapping ensures that the VM retains the same IP address, making it easier to manage and access.</p> <p>By setting this variable, you can ensure that the protected VM has a consistent IP address across both the primary and secondary sites, even if the network configurations are different.</p>	Optional	String

Variable	Description	Condition	Type
Primary Network Test Static IP	<p>Used to specify a static IP address for a protected VM in the test network of the primary site.</p> <p>In Nutanix, test networks are typically used for non-production workloads or for testing purposes. By specifying a static IP address for a protected VM in the test network, you can ensure that the VM retains the same IP address even after a failover event. This is particularly useful for applications that are dependent on specific IP addresses.</p>	Optional	String
DR Network Prod Static IP	<p>Used to define a fixed IP address for a protected VM in the production network of the disaster recovery (DR) site.</p> <p>This configuration ensures that the network configurations remain consistent during failover events, allowing the protected VMs to be accessed and communicate with other resources in the network. In addition, the VMs will be recovered with these defined IPs after the failover.</p>	Optional	String
DR Network Test Static IP	<p>Used to set a static IP address for a protected virtual machine in the test network of the disaster recovery (DR) site.</p> <p>This configuration ensures that the virtual machines maintain consistent network configurations during failover events and are able to communicate with other resources in the network.</p> <p>Additionally, it enables the virtual machines to be recovered with the pre-defined IPs after the failover.</p>	Optional	String

Network Configuration Runbook

Use this runbook to create an external network with or without NAT, VPC, overlay subnet, static route, policy-based routing and floating IP assignment. The network configuration runbook has the following variables.

Table 51: Network Configuration Runbook Variables

Variable	Description	Condition	Type
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String
Cluster Name	<p>Used to store the name of the Nutanix cluster that is being managed. It is a critical identifier for the cluster within the Nutanix environment and in external systems that interact with the cluster.</p> <p>This variable must be set to a unique and descriptive name that accurately reflects the purpose and function of the Nutanix cluster.</p>	Required	String
Virtual Switch Name	<p>Used to store the name of the virtual switch associated with the VLAN subnet in the Nutanix environment. This virtual switch is used to provide connectivity between virtual machines and other resources within the Nutanix cluster.</p>	Required	String
Create External Subnet	<p>Used to determine whether or not an external subnet has to be created.</p>	Required	String
External Subnet Name	<p>Used to represent the name of the subnet that enables external connectivity for a Nutanix cluster. Its purpose is to assign external IP addresses to the virtual machines and networking resources such as load balancers within the cluster.</p> <p>The external subnet must have an adequate number of available IP addresses to handle the anticipated workload of the cluster.</p>	Required	String

Variable	Description	Condition	Type
External VLAN ID	<p>Refers to the virtual local area network (VLAN) ID associated with the external subnet in a Nutanix cluster. It is used to identify the external subnet within the network infrastructure and enable traffic to flow to and from the external network.</p> <p>The VLAN ID must be unique within the network infrastructure and configured appropriately to allow for communication between the external network and the Nutanix cluster.</p> <p>Ensure that this variable is correctly configured and maintained to avoid any disruption to network connectivity for the Nutanix cluster.</p>	Required	Integer
External Subnet IP with Prefix	<p>Used to specify the network IP address and prefix length for the external subnet of a Nutanix cluster.</p> <p>The network IP address and prefix length define the range of IP addresses that are available for use in the external subnet. The prefix length is a number that indicates the number of bits used for the network part of the IP address. For example, a prefix length of 24 indicates that the first 24 bits of the IP address are used for the network, leaving 8 bits for the host address.</p> <p>This variable is typically set to an IP address and prefix length in the CIDR notation format, such as 192.168.0.0/24. This indicates that the network IP address is 192.168.0.0, and that the prefix length is 24 bits.</p> <p>Ensure that the Nutanix External Subnet Network IP with Prefix variable is correctly configured to avoid any conflicts or issues with IP address allocation on the external subnet. The network IP address and prefix length should be chosen based on the size of the external subnet and the number of IP addresses required to support the expected workload of the Nutanix cluster.</p>	Optional	String

Variable	Description	Condition	Type
External Subnet IP Pool Range	<p>Used to specify a range of IP addresses that can be assigned to virtual machines running in a Nutanix cluster.</p> <p>When a virtual machine is created, it can be assigned an IP address from the IP pool specified in this variable. The Nutanix cluster reserves the assigned IP address and ensure that it is not assigned to any other virtual machine.</p> <p>Ensure that this variable is correctly configured to avoid any conflicts or issues with IP address allocation on the external subnet. The IP pool should be large enough to support the expected workload of the Nutanix cluster, and must not overlap with any other IP address ranges in the environment.</p> <p>The variable can be specified as a range of IP addresses in CIDR notation, such as 192.168.0.10-192.168.0.20 or 10.0.0.0/24. Alternatively, it can be specified as a list of individual IP addresses, such as 192.168.0.10, 192.168.0.11, 192.168.0.12.</p>	Optional	String
External Subnet Gateway IP	<p>Used to specify the IP address of the default gateway for the external subnet of a Nutanix cluster.</p> <p>This variable is typically set to the IP address of the router that is connected to the external subnet.</p> <p>Ensure that the Nutanix External Subnet Gateway IP variable is correctly configured to ensure that traffic can be routed to and from the external subnet. The IP address specified in this variable must be reachable from the external subnet and must be valid for the network configuration of the external subnet.</p> <p>If the Nutanix cluster is configured to use multiple external subnets, each subnet must have its own gateway IP address specified in this variable.</p>	Optional	String

Variable	Description	Condition	Type
External Subnet NAT	<p>Used to specify whether or not Network Address Translation (NAT) is enabled on the external subnet of a Nutanix cluster.</p> <p>In the context of a Nutanix cluster, enabling NAT on the external subnet allows virtual machines running on the cluster to communicate with external networks using IP addresses that are translated by the Nutanix cluster.</p> <p>This variable can be set to either True or False to enable or disable NAT on the external subnet, respectively. Enabling NAT can be useful in situations where there is a shortage of public IP addresses, or where security policies require that internal IP addresses are not exposed to the public internet.</p> <div> Note: Carefully consider the implications of enabling or disabling NAT on the external subnet of a Nutanix cluster. This variable should be configured according to the specific requirements of the environment and in line with best practices for networking and security. </div>	Optional	String
Create VPC	Used to determine whether a VPC has to be created or not.	Required	String
VPC Name	Represents the name of a Nutanix Virtual Private Cloud (VPC) and is used to identify and reference a specific VPC within a Nutanix environment.	Required	String
Create Overlay Subnet	Used to determine whether an overlay subnet should be created or not.	Required	String

Variable	Description	Condition	Type
Overlay Subnet Name	<p>Used to represent the name assigned to a specific overlay subnet within the Nutanix environment. It provides a unique identifier for the overlay subnet.</p> <p>The overlay subnet utilizes various technologies, such as VXLAN (Virtual Extensible LAN), to encapsulate and tunnel network traffic between VMs. This enables VMs to communicate as if they are on the same local network, even if they are distributed across different physical hosts or datacenters.</p>	Required	String
Overlay Subnet IP With Prefix	<p>Refers to the IP address range allocated to the subnet, along with the associated subnet mask or prefix length. The network IP represents the base address of the subnet, while the prefix indicates the number of bits used to define the subnet.</p> <p>For example, if the network IP is 192.168.0.0 and the prefix is /24, it means that the subnet includes IP addresses ranging from 192.168.0.0 to 192.168.0.255, with a subnet mask of 255.255.255.0.</p> <p>The prefix length is represented as the number of consecutive bits set to 1 in the subnet mask. In the example above, /24 indicates that the first 24 bits of the IP address are used to identify the network portion, while the remaining 8 bits are available for host addresses.</p> <p>The network IP with prefix is essential for defining the address space and subnet boundaries within a Nutanix overlay subnet. It helps in determining the range of available IP addresses and configuring the appropriate network settings for virtual machines, routing, and connectivity within the subnet.</p>	Optional	String

Variable	Description	Condition	Type
Overlay Subnet Gateway IP	<p>Refers to the IP address assigned to the gateway device within the overlay network. The gateway acts as an intermediary between the overlay network and external networks or other subnets.</p> <p>The specific IP address of the gateway depends on the configuration of the overlay network. Typically, the gateway IP address is chosen from within the range of IP addresses assigned to the subnet. It serves as the default gateway for devices within the subnet to communicate with devices outside the subnet or in other subnets.</p> <p>For example, if you have an overlay network with a subnet using the IP address range 192.168.0.0/24, the gateway IP address might be assigned as 192.168.0.1. This means that any device within the subnet can use 192.168.0.1 as the gateway IP to send traffic outside the subnet.</p> <div> <p>Note: The specific configuration of a subnet overlay network, including the choice of gateway IP, can vary depending on the network infrastructure and the technology being used for overlay networking, such as Virtual Extensible LAN (VXLAN) or Generic Routing Encapsulation (GRE).</p> </div>	Required	String
Prism Central Username	<p>Used to specify the username that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the username that has been granted access to the Prism Central management interface.</p> <p>The username specified in this variable should have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

Variable	Description	Condition	Type
Prism Central Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password is kept secure and protected. Additionally, Nutanix recommends you to periodically change the password for security reasons.</p>	Required	String

Policy-Based Routing Runbook

Use this runbook to create additional policy-based routing in the VPC. The policy-based routing runbook has the following variables.

Table 52: Policy-Based Routing Runbook Variables

Variable	Description	Condition	Type
Operation	Used to determine whether to create or delete Nutanix policy-based routing.	Required	String
Priority of route	Refers to the order in which routes are evaluated and applied based on their defined priorities. The route with the highest priority is processed first, followed by routes with lower priorities.	Required	Integer

Variable	Description	Condition	Type
Protocol Type	<p>Refers to the specific network protocol that is used for routing traffic between different virtual private clouds (VPCs) or subnets. The protocol type determines the type of traffic that is allowed or restricted based on the defined policies.</p> <p>Common protocol types used in Nutanix policy-based VPC routing include TCP, UDP, and ICMP.</p> <div> <p>Note: Nutanix policy-based VPC routing can also support other protocols such as ICMPv6 (ICMP for IPv6), IGMP for multicast traffic, or specific application-layer protocols based on your network requirements.</p> </div> <p>By specifying the protocol type in policy-based VPC routing, you can define rules and policies that govern the routing of traffic based on the specific protocol being used. This allows you to control and optimize the flow of network traffic within your Nutanix environment based on your application and security requirements.</p>	Required	String
Protocol Number	<p>Refers to the numerical identifier assigned to different network protocols. These protocol numbers are defined by the Internet Assigned Numbers Authority (IANA) and are used to identify and differentiate various protocols when configuring static routes.</p> <p>When configuring static routes in Nutanix, you may need to specify the appropriate protocol number based on the protocol being used. This ensures that the routing table correctly identifies and handles the traffic associated with that protocol.</p>	Required	Integer

Variable	Description	Condition	Type
ICMP Protocol Parameter Type	<p>Refers to the specific type of ICMP message being used. The ICMP Protocol Parameter Types in Nutanix VPC policy-based routing are the same as those used in standard ICMP. The commonly used ICMP Protocol Parameter Types in Nutanix VPC policy-based routing are:</p> <ul style="list-style-type: none"> • Type 0: Echo Reply • Type 3: Destination Unreachable • Type 8: Echo Request • Type 11: Time Exceeded • Type 12: Parameter Problem <p>These ICMP Protocol Parameter Types help classify and handle different types of ICMP messages within Nutanix VPC policy-based routing. By configuring policies based on the specific ICMP message types, you can control the routing behavior and apply actions accordingly.</p> <p>When defining policy-based routing rules in Nutanix VPC, you must consider the ICMP Protocol Parameter Types that are relevant to your network requirements. This allows you to create rules and actions that effectively manage ICMP traffic and meet your specific routing needs.</p>	Optional	String
ICMP Protocol Parameter Code	<p>Refers to the specific code associated with an ICMP message type. ICMP messages are used for various purposes in network communication, such as error reporting, troubleshooting, and network management.</p> <p>The ICMP Protocol Parameter Codes in Nutanix policy-based routing are the same as those used in standard ICMP.</p>	Optional	String

Variable	Description	Condition	Type
Source Port Range List	<p>Refers to a list of source port ranges used for defining policies and routing decisions based on the source port of network traffic.</p> <p>When configuring policy-based routing rules in Nutanix VPC, you can specify a range of source ports or multiple source port ranges to define the criteria for routing decisions. Each entry in the Source Port Range List typically consists of a starting port number and an ending port number, indicating the range of source ports to be matched.</p> <p>For example, a Source Port Range List can have the following entries.</p> <ul style="list-style-type: none"> • Range 1: Start Port 1024, End Port 2048 • Range 2: Start Port 5000, End Port 6000 <p>These ranges can be used to match incoming traffic based on the source port number. If the source port of a packet falls within any of the defined ranges, the corresponding routing action associated with that range applies.</p> <p>By configuring the Source Port Range List in Nutanix VPC policy-based routing, you can have granular control over routing decisions based on the source ports of network traffic. This allows you to define specific policies and actions based on the source port ranges, helping you manage and route traffic according to your network requirements.</p>	Optional	String

Variable	Description	Condition	Type
Destination Port Range List	<p>Refers to a list of destination port ranges used for defining policies and routing decisions based on the destination port of network traffic.</p> <p>When configuring policy-based routing rules in Nutanix VPC, you can specify a range of destination ports or multiple destination port ranges to define the criteria for routing decisions. Each entry in the Destination Port Range List typically consists of a starting port number and an ending port number, indicating the range of destination ports to be matched.</p> <p>For example, a Destination Port Range List can have the following entries.</p> <ul style="list-style-type: none"> • Range 1: Start Port 80, End Port 443 • Range 2: Start Port 8080, End Port 8090 <p>These ranges can be used to match incoming traffic based on the destination port number. If the destination port of a packet falls within any of the defined ranges, the corresponding routing action associated with that range applies.</p> <p>By configuring the Destination Port Range List in Nutanix VPC policy-based routing, you can have granular control over routing decisions based on the destination ports of network traffic. This allows you to define specific policies and actions based on the destination port ranges, helping you manage and route traffic according to your network requirements.</p>	Optional	String

Variable	Description	Condition	Type
Source Address Type	<p>Refers to the Source Address Type, which can be categorized into the following options.</p> <ul style="list-style-type: none"> • External: Refers to external or public IP addresses. It allows you to define routing policies based on traffic originating from external sources, such as the internet or other external networks. Using the External source address type, you can apply specific routing rules to handle traffic coming from outside of your VPC. • All: Represents all possible source IP addresses. When you choose the All source address type, it means that the routing policies is applied to all source IP addresses within your VPC, regardless of their specific ranges or subnets. This can be useful when you want the policy to be applied universally to all traffic within your VPC. • Custom: Used to specify a custom source address or a range of source addresses for policy-based routing. You can define specific IP addresses, subnet ranges (CIDR notation), or address groups as the source address criteria. Custom source address type gives you flexibility in defining granular routing policies based on specific source addresses or address ranges that suit your network requirements. <p>By selecting the appropriate Source Address Type in Nutanix VPC policy-based routing, you can define the scope of your routing policies and specify the sources from which the policies should be applied. Whether you want to target external sources, all sources within your VPC, or specific custom-defined sources, the Source Address Type provides the flexibility to tailor your routing rules accordingly.</p>	Required	String

Variable	Description	Condition	Type
Source IP Prefix	<p>Refers to the specific IP address prefix or subnet range used to define the source IP addresses for routing policies.</p> <p>When configuring policy-based routing rules, you can specify a source IP prefix that represents a range of IP addresses or a subnet in CIDR notation. This allows you to define the source addresses from which the routing policy should be applied.</p> <p>For example, if you define the Source IP Prefix as 192.168.0.0/24, it means that the routing policy will be applied to all IP addresses within the subnet range of 192.168.0.0 to 192.168.0.255. This allows you to target a specific range of source IP addresses for routing decisions.</p> <p>By specifying the Source IP Prefix in Nutanix policy-based routing, you can control which source IP addresses have to be subjected to the routing policy. This allows you to define routing rules based on specific subnets or IP address ranges, providing granular control over the flow of network traffic based on its source.</p>	Optional	String

Variable	Description	Condition	Type
Destination Address Type	<p>Refers to the Destination Address Type, which can be categorized into the following options:</p> <ul style="list-style-type: none"> • All: Represents all possible destination IP addresses. When you choose the All destination address type, it means that the routing policies will be applied to all destination IP addresses, regardless of their specific ranges or subnets. This can be useful when you want the policy to be applied universally to all traffic regardless of its destination. • Custom: Used to specify a custom destination address or a range of destination addresses for policy-based routing. You can define specific IP addresses, subnet ranges (CIDR notation), or address groups as the destination address criteria. Custom destination address type gives you flexibility in defining granular routing policies based on specific destination addresses or address ranges that suit your network requirements. • External: Refers to external or public IP addresses. It allows you to define routing policies based on traffic destined for external sources, such as the internet or other external networks. Using the External destination address type, you can apply specific routing rules to handle traffic going outside of your network. <p>By selecting the appropriate Destination Address Type in Nutanix policy-based routing, you can define the scope of your routing policies and specify the destinations to which the policies apply. Whether you want to target all destinations, custom-defined destinations, or external destinations, the Destination Address Type provides the flexibility to configure your routing rules accordingly.</p>	Required	String

Variable	Description	Condition	Type
Destination IP with Prefix	<p>Refers to the specific IP address prefix or subnet range used to define the destination IP addresses for routing policies.</p> <p>When configuring policy-based routing rules, you can specify a destination IP prefix that represents a range of IP addresses or a subnet in CIDR notation. This allows you to define the destination addresses to which the routing policy should be applied.</p> <p>For example, if you define the Destination IP with Prefix as 10.0.0.0/24, it means that the routing policy will be applied to all IP addresses within the subnet range of 10.0.0.0 to 10.0.0.255. This allows you to target a specific range of destination IP addresses for routing decisions.</p> <p>By specifying the Destination IP with Prefix in Nutanix VPC policy-based routing, you can control which destination IP addresses should be subject to the routing policy. This allows you to define routing rules based on specific subnets or IP address ranges, providing granular control over the flow of network traffic based on its destination.</p>	Optional	String

Variable	Description	Condition	Type
Action	<p>Refers to the action taken for the matched static route. You have the following actions that can be associated with a static route:</p> <ul style="list-style-type: none"> • PERMIT: Indicates that the matched traffic is allowed or permitted to proceed. This action allows the traffic to follow the configured static route and reach its intended destination. • DENY: Indicates that the matched traffic is explicitly denied or blocked from proceeding. This action is used to prevent specific traffic from following the configured static route, effectively blocking access to the destination. • REROUTE: Indicates that the matched traffic is redirected or rerouted to a different next hop or destination. This action is useful when you want to redirect traffic from the original static route to an alternative path or destination. <p>By selecting the appropriate action in Nutanix VPC static routing, you can control the behavior of the matched traffic. Whether you want to permit the traffic to follow the static route, deny the traffic to block access, or reroute the traffic to an alternative path, the action associated with the static route allows you to define the desired behavior for the traffic flow.</p>	Required	String

Variable	Description	Condition	Type
VPC Name	<p>Refers to the name or identifier of the Virtual Private Cloud (VPC) in which the static routing configuration is being applied.</p> <p>When configuring static routes in Nutanix VPC, you associate the static routes with a specific VPC by specifying the VPC name. This ensures that the static routes are applied within the context of the designated VPC.</p> <p>By providing the VPC name in Nutanix VPC static routing, you can define the scope of the routing configuration and ensure that the static routes are applied to the correct VPC within your cloud environment.</p>	Required	String
Bi-Directional	<p>Refers to the capability of routing traffic in both directions, allowing communication between source and destination devices in both outbound and inbound directions. This enables you to control and shape network traffic not only when it originates from your network (outbound) but also when it is destined for your network (inbound).</p> <p>By enabling bi-directional policy-based routing, you can implement advanced routing policies and traffic management techniques that consider the characteristics and requirements of traffic in both directions. This can be useful in scenarios where you want to apply specific routing rules, perform traffic shaping, or enforce security measures for bidirectional communication between different network segments or across different networks.</p>	Required	String

Variable	Description	Condition	Type
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String
Prism Central Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password is kept secure and protected. Additionally, Nutanix recommends you to periodically change the password for security reasons.</p>	Required	String
Prism Central Username	<p>Used to specify the username that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the username that has been granted access to the Prism Central management interface.</p> <p>The username specified in this variable should have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

Backup and Restore Runbook

Use this runbook to create or restore snapshots. The backup and restore runbook has the following variables.

Table 53: Backup and Restore Runbook Variables

Variable	Description	Condition	Type
App Name	<p>Refers to the app name. The app name for a Nutanix blueprint depends on the specific application or infrastructure being deployed using that blueprint.</p> <p>For example, if you deploy a blueprint to create a web application stack, you can assign names such as Web Application Deployment or Web App Blueprint.</p>	Required	String
Blueprint Name	<p>Refers to the name of a predefined set of configurations and specifications that describe the infrastructure or application stack to be deployed on the Nutanix platform. It serves as a template or blueprint for creating and provisioning resources within a Nutanix environment.</p>	Required	String
Account Name	<p>Represents the name of the Nutanix storage account used for provisioning and managing storage resources on the Nutanix cluster. Storage accounts provide a logical grouping of storage resources, such as containers and datastore, and are used to allocate and manage storage capacity for various applications and workloads.</p>	Required	String
Project Name for App Creation	<p>Refers to a unique identifier or label for the application development project.</p> <p>The project name is used to track and manage the progress, resources, and activities related to the application creation process on the Nutanix platform.</p> <p>The Project Name depends on the nature of the application being developed, the organizational naming conventions, and any relevant naming guidelines or standards in place.</p>	Required	String

Variable	Description	Condition	Type
Create Protection Policy	<p>Refers to the user-defined name given to a policy that determines how often and for how long backups or snapshots of a specific object are taken in the Nutanix cluster. This object can be a virtual machine, a container, or any other resource that requires protection against data loss.</p> <p>The policy can be configured with various options, such as backup frequency, retention period, backup schedule, compression, and encryption settings.</p> <p>This parameter is used to specify which protection policy to apply when creating or managing data protection policies for objects in the Nutanix cluster. The policy helps manage backup storage space, based on pre-defined recovery window goals.</p>	Required	String
App Protection Policy Name	Refers to a set of rules and configurations designed to protect and safeguard applications running within the Nutanix environment. The primary goal of an App Protection Policy is to ensure the availability, integrity, and recoverability of critical applications and their associated data.	Required	String
Environment Name	<p>Refers to the creation of an environment within a Nutanix project that allows for the deployment and management of applications and services.</p> <p>The creation of an environment typically involves specifying the infrastructure resources required for the environment, such as virtual machines, networks, and storage, as well as the applications and services that will be deployed within the environment.</p>	Required	String
Cluster Name	Used to specify the name of the Nutanix cluster that is used for the deployment. The cluster name is a unique identifier for a Nutanix cluster and is used for managing resources such as virtual machines, networks, storage, and user accounts.	Required	String

Variable	Description	Condition	Type
Subnet Name for APP VM	<p>Refers to the network subnet on which the application VMs or virtual resources are deployed within the Nutanix environment.</p> <p>The subnet name is typically chosen based on your organizational naming conventions or network segmentation strategy. It is independent of the backup and restore operations but plays a role in providing connectivity and defining network boundaries for the application.</p>	Required	String
Image Name for APP Creation	<p>Represents the name or identifier assigned to the virtual machine (VM) image used to create instances of the application within the Nutanix environment when creating an image for application deployment in Nutanix.</p>	Required	String
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Ensure that you keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String
Prism Central Username	<p>Used to specify the username that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>The Nutanix Prism Central Username variable should be set to the username that has been granted access to the Prism Central management interface.</p> <p>The username specified in this variable must have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

Variable	Description	Condition	Type
Prism Central Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password is kept secure and protected. Additionally, Nutanix recommends you to periodically change the password for security reasons.</p>	Required	String

Floating IP Assignment Runbook

Use this runbook to assign a floating IP to a specific VM. The floating IP assignment runbook has the following variables.

Table 54: Floating IP Assignment Runbook Variables

Variable	Description	Condition	Type
Floating IP Assignment Type	Refers to a network configuration in which an IP address is not permanently tied to a specific device or network interface. Instead, the IP address can be dynamically assigned to different devices or interfaces as needed.	Required	String
VM IP	Refers to the IP address assigned to a specific virtual machine (VM). In a virtualized environment, each VM is typically allocated its own unique IP address, allowing it to communicate with other devices on the network	Required	String
VPC Name	Represents the name of a Nutanix Virtual Private Cloud (VPC) and is used to identify and reference a specific VPC within a Nutanix environment.	Required	String

Variable	Description	Condition	Type
External Subnet Name	<p>Represents the name of the subnet that enables external connectivity for a Nutanix cluster. Its purpose is to assign external IP addresses to the virtual machines and networking resources such as load balancers within the cluster.</p> <p>The external subnet must have an adequate number of available IP addresses to handle the anticipated workload of the cluster.</p>	Required	String
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources.</p> <p>You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Ensure that you keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String
Prism Central Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password is kept secure and protected. Additionally, Nutanix recommends you to periodically change the password for security reasons.</p>	Required	String

Variable	Description	Condition	Type
Prism Central Username	<p>Used to specify the username that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the username that has been granted access to the Prism Central management interface.</p> <p>Ensure that the Nutanix Prism Central Username is correctly configured and kept up-to-date to ensure that the Nutanix clusters can be managed effectively. The username specified in this variable should have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

Single-VM Blueprint Runbook

Use this runbook to create a single-VM blueprint. The single-VM blueprint runbook has the following variables.

Table 55: Single-VM Blueprint Runbook Variables

Variable	Description	Condition	Type
Management PC Username	Used to specify the username used for authentication with the Nutanix Prism Central management interface. The management username is required for performing administrative tasks on the Nutanix cluster through the Prism Central management interface. It allows access to features such as health monitoring, capacity planning, resource management, and user management.	Required	String
Management PC Password	Used to specify the password that you use for authentication with the Nutanix Prism Central management interface. The management password is required for performing administrative tasks on the Nutanix cluster through the Prism Central management interface. It is used in combination with the management username to authenticate and access features such as health monitoring, capacity planning, resource management, and user management.	Required	String

Variable	Description	Condition	Type
APP Name	The app name for a Nutanix blueprint depends on the specific application or infrastructure being deployed using that blueprint.	Required	String
Blueprint Name	Refers to the name of the blueprint. A blueprint is a predefined set of configurations and specifications that describe the infrastructure or application stack to be deployed on the Nutanix platform. It serves as a template for creating and provisioning resources within a Nutanix environment.	Required	String
Project Name for App Creation	Refers to the name given to a specific project or initiative for creating an application within the Nutanix environment. The project name represents a unique identifier or label for the application development project.	Required	String
Account Name	Represents the name of the Nutanix storage account used for provisioning and managing storage resources on the Nutanix cluster. Storage accounts provide a logical grouping of storage resources, such as containers and datastores, and are used to allocate and manage storage capacity for various applications and workloads.	Required	String
Environment Name	Refers to the creation of an environment within a Nutanix project, which allows for the deployment and management of applications and services. An environment provides a platform for developing and publishing applications, and includes a range of tools and services for orchestration, automation, and monitoring. The creation of an environment typically involves specifying the infrastructure resources required for the environment, such as virtual machines, networks, and storage, as well as the applications and services that will be deployed within the environment.	Required	String

Variable	Description	Condition	Type
Tenant Category for APP VM	Refers to the category that you can assign to your application VMs using the VM Custom Tags feature. The category allows you to categorize and organize your VMs based on different criteria, such as application type, department, or environment.	Required	String
Cluster Name	Used to specify the name of the Nutanix cluster that is used for the deployment. The cluster name is a unique identifier for a Nutanix cluster and is used for managing resources such as virtual machines, networks, storage, and user accounts.	Required	String
App VM OS Type	Refers to the operating system installed on the app VM. When creating an application VM in Nutanix, you select the desired operating system from a list of supported OS options. This selection can be made based on the specific OS distribution and version that you intend to install on the VM.	Required	String
Image Name for APP Creation	Represents the name or identifier assigned to the virtual machine (VM) image used to create instances of the application within the Nutanix environment.	Required	String
App VM Memory in GB	<p>Refers to the amount of memory (RAM) that you can allocate to the VM in gigabytes (GB) when creating an application VM. The memory allocation determines the available RAM resources for the VM to run applications and perform its tasks.</p> <p>Ensure that you consider the requirements of your applications and workloads when determining the appropriate amount of memory to allocate to your VMs. Insufficient memory can result in performance issues, while excessive memory allocation can lead to resource wastage.</p>	Required	Integer

Variable	Description	Condition	Type
Number of App VM vCPUs	<p>Refers to the number of virtual CPUs (vCPUs) allocated to an application VM. This parameter determines the computing resources available to the VM for executing tasks and running applications.</p> <p>The specific number of vCPUs assigned to an app VM can vary based on the requirements of the workload and the capabilities of the underlying hardware and hypervisor.</p>	Required	Integer
Subnet Name for APP VM	<p>Refers to the network subnet on which the application VMs or virtual resources are deployed within the Nutanix environment.</p> <p>The subnet name is typically chosen based on your organization's naming conventions or network segmentation strategy. It helps identify and differentiate the network segment associated with the application and its VMs.</p>	Required	String
Custom IP for VM	<p>Refers to the custom IP address that you can assign to a virtual machine (VM) by leveraging the network configuration options available in your environment.</p>	Optional	String
App Credential User	<p>The management of application credentials and users is typically handled within the operating system and applications themselves. Nutanix provides various tools and features to manage the VMs and the underlying infrastructure, but it does not directly manage the application-level user credentials.</p> <p>To manage application credentials and users within a VM running on Nutanix, follow the standard practices specific to the operating system and applications installed on the VM.</p>	Required	String

Variable	Description	Condition	Type
App Credential Password	<p>The management of application credentials and users is typically handled within the operating system and applications themselves. Nutanix provides various tools and features to manage the VMs and the underlying infrastructure, but it does not directly manage the application-level user credentials.</p> <p>To manage application credentials and users within a VM running on Nutanix, follow the standard practices specific to the operating system and applications installed on the VM.</p>	Required	String
PC IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources. You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Ensure that you keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	Integer
Project User Username	Used to represent the specific username that you choose for a project user account within your Nutanix environment.	Required	String
Project User Password	Used to represent the specific password that you choose for a project user account within your Nutanix environment.	Required	String

Test Failover Runbook

Use this runbook to perform test failover or failback of a Disaster Recovery (DR) setup. The test failover runbook has the following variables.

Table 56: Test Failover Runbook Variables

Variable	Description	Condition	Type
Recovery Plan Name	<p>Refers to a user-defined name given to a recovery plan that specifies the steps required to recover a particular object or service in the event of a disaster or system failure. The recovery plan can be set up with various options, such as the order in which services or virtual machines should be recovered, the specific recovery point to use, and the notification settings for administrators.</p> <p>This variable is used as a parameter to specify which recovery plan to apply when recovering objects or services in the Nutanix cluster.</p>	Required	String
Account Name of Entity Failing Over From	<p>Refers to the name of the entity or organization that is failing over from one environment to another within the Nutanix infrastructure. It typically represents the account or organization associated with the source environment that is being transitioned or failed over to a target environment.</p>	Required	String
Account Name of Entity Failing Over To	<p>Represents the name of the target account or organization that is receiving the failover or migration from another environment. This refers to the account or organization associated with the destination or target Nutanix environment.</p>	Required	String
Prism Central IP	<p>Refers to the network address or IP address of the Nutanix Prism Central management platform. It is the location where you can access the central management console for managing Nutanix clusters, including virtualization, storage, and networking resources. You can use this IP address to connect to the Prism Central instance from a web browser or through API calls to automate management tasks.</p> <p>Keep the Nutanix Prism Central IP secure, as it provides access to the management platform and the Nutanix clusters it manages.</p>	Required	String

Variable	Description	Condition	Type
Prism Central Password	<p>Used to store the password that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the password that corresponds to the username specified in the Nutanix Prism Central Username variable.</p> <p>Ensure that the Nutanix Prism Central Password is kept secure and protected. Additionally, Nutanix recommends you to periodically change the password for security reasons.</p>	Required	String
Prism Central Username	<p>Used to specify the username that is used to authenticate with the Nutanix Prism Central management interface.</p> <p>This variable must be set to the username that has been granted access to the Prism Central management interface.</p> <p>Ensure that the Nutanix Prism Central Username is correctly configured and kept up-to-date to ensure that the Nutanix clusters can be managed effectively. The username specified in this variable should have the appropriate level of permissions to perform the required management tasks in Prism Central.</p>	Required	String

ENDPOINTS IN SELF-SERVICE

The topics in this section cover the configuration and usage of endpoints in Self-Service.

Endpoints Overview

Endpoints are the target resources where the tasks defined in a runbook or blueprint are run.

The endpoints are collection of IP addresses or VMs. The collection of VMs can be a static selection or can be dynamic with filter rules applied.

You have the following types of endpoints.

- A Windows machine
- A Linux machine
- An HTTP service endpoint

For information on how to create an endpoint, see [Creating an Endpoint](#) on page 386.

Endpoints with Virtual Machines

For Windows or Linux endpoint type, you can select virtual machines as the target type. Selecting VMs as target type is useful in cases where you run a set of scripts on multiple VMs and then restart the VMs. For example, you can select VMs as target type to upgrade a software on your VMs.

After you select VMs as the target type, you must select the provider account to list all the associated VMs. You can filter the list of VMs. You can either select the VMs manually or enable the option to automatically select the filtered VMs for your endpoint.

Creating an Endpoint

Create an endpoint to run the tasks that you define in a runbook or blueprint.

About this task

You must have the role of an administrator or a developer to create an endpoint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Endpoints** in the navigation bar.
4. Click **Create Endpoint**.
5. In the Create Endpoint window, type a name and description for the endpoint.
6. From the **Project** dropdown menu, select the project to which you want to assign the endpoint.
7. Select the type of the endpoint. You can select **Windows**, **Linux**, or **HTTP** as the endpoint type.

8. If you have selected **HTTP** as the endpoint type, do the following.

- a. In the **Select tunnel to connect with** dropdown menu, select the tunnel that you can use to get access to the endpoint within the VPC. This step is optional.

The **Select tunnel to connect with** dropdown menu shows only those VPC tunnels that are allowed in the project that you selected for the endpoint.

- b. In the **Base URL** field, enter the base URL of the HTTP endpoint. A base URL is the consistent part of the endpoint URL.
- c. To verify the URL of the HTTP endpoint with a TLS certificate, click the **Verify TLS Certificate** checkbox. Use this option to securely access the endpoint. This step is optional.
- d. To use a proxy server that you configured in Prism Central, select the **Use PC Proxy configuration** checkbox.

Note: Ensure that Prism Central has the appropriate HTTP proxy configuration.

- e. In the **Retry Count** field, type the number of attempts the system must perform to create a task after each failure.
The default value is 1, which implies that the task creation process stops after the first attempt.
- f. In the **Retry Interval** field, type the time interval in seconds for each retry if the task fails. The default value is 10 seconds.
- g. In the **Connection Timeout** field, type the time interval in seconds after which the connection attempt to the endpoint stops. The default value is 120 seconds.
- h. To add an authentication method to connect to an HTTP endpoint, click **Authentication** and select **Basic** from the **Type** field. Type a username and password to authenticate the endpoint. This step is optional.
By default, the authentication type is set to **None**.

9. If you have selected **Windows** or **Linux** as the endpoint type, then select a target type. The target type can be **IP Addresses** or **VMs**.

10. If you have selected **IP Addresses** as the target type, then do the following:

- a. In the **Select tunnel to connect with** dropdown menu, select the tunnel that you can use to get access to the endpoint within the VPC. This step is optional.

The **Select tunnel to connect with** dropdown menu shows only those VPC tunnels that are allowed in the project that you selected for the endpoint.

- b. In the **IP Addresses** field, type the IP address to access the endpoint device.

11. If you have selected **VMs** as the target type, do the following:

- a. In the **Account** dropdown menu, select an account.

The **Account** dropdown menu displays all the provider accounts that you configured in the project. After you select the provider account, the window displays the VMs associated with the provider account.

- b. To filter VMs, use the **Filter By** options.

You can use different attribute, operator, and value criteria to get accurate results.

- c. Select the VMs that you want to add to your endpoint.

You can also use **Auto Select VMs** to automatically add the filtered VMs to your endpoint.

Note: The resolution of the VMs from the filters happens at the runbook execution.

12. In the **Connection Protocol** field, select the connection protocol to access the endpoint. You can select either **HTTP** or **HTTPS**. This field appears only for the **Windows** endpoint type.
13. In the **Port** field, type the port number to access the endpoint.
14. Add a credential for Windows or Linux endpoint type to access the endpoint.
 - a. Click **Credential**.
 - b. Select the type of credential that you want to add under the **Type** section.
 - » **Static**: Credentials store keys and passwords in the credential objects that are contained in the blueprints.
 - » **Dynamic**: Credentials fetch keys and passwords from an external credential store that you integrate with Self-Service as the credential provider.
 - c. In the **Username** field, type a username for the endpoint credential.

For dynamic credentials, specify the @@(username)@@ that you defined while configuring the credential provider.

Note: A dynamic credential provider definition requires username and secret. The secret variable is defined by default when you configure your credential provider. However, you must configure a runbook in the dynamic credential provider definition for the username variable before you use the variable in different entities.

- d. Select either **Password** or **SSH Private Key** as the secret type.
- e. Do one of the following to configure the secret type.
 - » If you have selected **Static** as the credential type and **Password** as the secret type, then type the password in the **Password** field.
 - » If you have selected **Static** as the credential type and **SSH Private Key** as the secret type, then enter or upload the key in the **SSH Private Key** field.
 - » If you have selected **Dynamic** as the credential type and **Password** or **SSH Private Key** as the secret type, then select a credential provider in the **Provider** field. After you select the provider, verify or edit the attributes defined for the credential provider.

If the private key is password protected, click **+Add Passphrase** to provide the passphrase. For dynamic credentials, you must configure a runbook in the dynamic credential provider definition for the passphrase variable and then use the @@{passphrase}@@ variable.

The type of SSH key supported is RSA. For information on how to generate a private key, see [Generating SSH Key on a Linux VM](#) on page 444 or [Generating SSH Key on a Windows VM](#) on page 445.

15. Click **Save**.

What to do next

You can add the endpoint to a runbook. For more information, see [Creating a Runbook](#) on page 266.

Deleting an Endpoint

Perform the following procedure to delete an endpoint.

About this task

You must have the role of an administrator or a developer to delete an endpoint.

Procedure

1. Log in to your Prism Central instance.
2. Select **Self-Service** in the Application Switcher.
3. Click **Endpoints** in the navigation bar.
4. Select the endpoint that you want to delete.
5. From the **Action** dropdown menu, select **Delete**.
6. In the Confirm Delete window, click **Delete**.

Configuring a Remote Machine to Run Self-Service Python Scripts

You can configure your customized remote machine to run Self-Service Python scripts. You can add custom modules, any version of Python, and their corresponding binaries to the remote machine.

Procedure

1. SSH to the remote machine.
2. Create a virtual environment in the remote machine. For example, you can run the following commands:

```
cd ~
```

```
mkdir .calm && cd .calm
```

```
python3 -m venv venv
```

The directory structure in this example is `/home/<user>/.calm/venv`. It is recommended to use the same folder name (for example, `.calm`) under the path as mentioned in the scripts.

Note:

- When you define your own custom virtual environment path (`venv`), you must update the Epsilon container environment to point to your custom virtual environment path. You need to do this update every time you start the Epsilon container or update the Self-Service inside the Prism Central. For example, run the following command inside the Epsilon container.

```
export GADARZ3_REMOTE_INTERPRETER=CUSTOM_PYTHON_PATH[1]
```

- In case of scale-out setup, you must update all the nodes.
- After the environment variables are updated, restart the Arjun services.

```
activate
```

```
status
```

The `status` command shows the services. For example,

```
supervisor> status
db:postgresql          RUNNING    pid 98, uptime 1 day, 22:55:51
db:redis               RUNNING    pid 99, uptime 1 day, 22:55:51
epsilon-app:gozaffi_0   RUNNING    pid 7621, uptime 1 day, 15:33:06
epsilon-app:proxy_service RUNNING    pid 7620, uptime 1 day, 15:33:06
epsilon-engine:arjun_0  RUNNING    pid 43468, uptime 0:05:59
epsilon-engine:arjun_1  RUNNING    pid 43479, uptime 0:05:48
epsilon-engine:durga_0  RUNNING    pid 7376, uptime 1 day, 15:33:37
epsilon-engine:durga_1  RUNNING    pid 7377, uptime 1 day, 15:33:37
epsilon-engine:elastic_search RUNNING    pid 7379, uptime 1 day, 15:33:37
```

```
epsilon-engine:indra_0      RUNNING   pid 7678, uptime 1 day, 15:33:03
epsilon-engine:indra_1      RUNNING   pid 7688, uptime 1 day, 15:33:02
epsilon-engine:jove         RUNNING   pid 7380, uptime 1 day, 15:33:37
epsilon-engine:karan_0      RUNNING   pid 7648, uptime 1 day, 15:33:04
epsilon-engine:karan_1      RUNNING   pid 7641, uptime 1 day, 15:33:04
epsilon-engine:narad        RUNNING   pid 7373, uptime 1 day, 15:33:37
epsilon-engine:vajra_0      RUNNING   pid 7378, uptime 1 day, 15:33:
```

You can then run the following command to restart the Arjun services.

```
restart epsilon-engine:arjun_N
```

where *N* is the worker number.

3. You can install the desired Python packages in your virtual environment.

The packages that are commonly used in automation scenarios are:

- requests
- azure
- kubernetes
- boto3
- simplejson
- ujson
- pformat
- telnetlib3
- pymssql

BACKUP AND RESTORE IN SELF-SERVICE

The topics in this section elaborate how to back up and restore Self-Service data.

Self-Service Data Backup and Restore

You can take a backup of the Self-Service data to a specified location on your machine and restore the data to a new Prism Central.

Note: Restore of backed up data taken from an on-premise Self-Service to Self-Service VM or Self-Service VM to on-premise Self-Service is not supported.

You back up the following data:

- **Zookeeper Data**
 - Calm instance data
- **Elastic Search Data**
 - Task Run Logs
 - App Icons
 - Marketplace branding Logos
- **IDF PC Tables**
 - project
 - Entity Capabilities

- **IDF Calm Tables**

- "management_server_account"
- "marketplace_item"
- "nucalm_action"
- "nucalm_action_run"
- "nucalm_app_beam_status"
- "nucalm_app_blueprint"
- "nucalm_app_failover_status"
- "nucalm_application"
- "nucalm_application_cfg"
- "nucalm_app_protection_status"
- "nucalm_budget"
- "nucalm_consumption"
- "nucalm_cost"
- "nucalm_credential"
- "nucalm_deployment"
- "nucalm_deployment_cfg"
- "nucalm_deployment_element"
- "nucalm_environment"
- "nucalm_library_task"
- "nucalm_library_variable"
- "nucalm_lifecycle"
- "nucalm_loadbalancer"
- "nucalm_loadbalancer_cfg"
- "nucalm_package"
- "nucalm_package_cfg"
- "nucalm_package_element"
- "nucalm_platform_instance_element"
- "nucalm_policy_rule"
- "nucalm_price_item"
- "nucalm_price_item_status"
- "nucalm_published_service"
- "nucalm_published_service_cfg"
- "nucalm_recovery_plan_job_sync_status"

- "nucalm_runbook"
- "nucalm_run_log"
- "nucalm_secret"
- "nucalm_service"
- "nucalm_service_cfg"
- "nucalm_service_element"
- "nucalm_service_upgrade_history"
- "nucalm_service_version"
- "nucalm_substrate"
- "nucalm_substrate_cfg"
- "nucalm_substrate_element"
- "nucalm_sync_status"
- "nucalm_task"
- "nucalm_user_file"
- "nucalm_variable"
- "nucalm_worker_state"

Note: The default batch query size for the IDF entities is 100, except for the following entities where the default batch query size is 500:

- "nucalm_task"
- "nucalm_action"
- "nucalm_run_log"
- "nucalm_variable"
- "nucalm_substrate_element"

When the `override-default-batch-size` flag is `True`, the batch query size of all IDF entities honour the `fetch-limit`.

Backing up Self-Service Data

You can take a backup of the entire Self-Service data to a specified location on your machine.

About this task

Note: If the policy engine on your old setup was on a different network than that of your new Prism Central, you must back up and restore the policy engine database as well, along with the Self-Service data. For more information on backing up and restoring the policy engine database, see [Backing Up and Restoring Policy Engine Database](#) on page 399.

Procedure

1. Log on to Prism Central using the SSH session as a `nutanix` user.

2. Run the following command to determine the Prism Central leader node in a scale-out setup.

```
sudo kubectl -n ntnx-base get pods
```

In case of scale-out setup, the command returns an output on two out of three nodes. Use any of the two nodes that give an output as the leader node.

The Prism Central leader node returns the output based on the point of deliveries running on that Prism Central. If you do not see any output, log on to a different Prism Central IP address and run the command again.

```
$ ssh <Prism Central IP address>
```

You can run the following command to get the Prism Central IP addresses:

```
nutanix@pcvm$ svmips
```

Note: To complete the backup successfully, ensure that all point of deliveries are in the Running or Completed state. You can contact Nutanix Support if you have point of deliveries that are not in the Running or Completed state in the output.

3. Run the following backup command.

```
nutanix@pcvm$ docker exec -it nucalm sh -c "/home/calm/bin/calmdata backup"
```

The backup file is stored in the default folder located at /tmp/default.

Note: Ensure that the backup folder has only the calmdata tar file dump.

4. Run the IAM backup script to take a backup of the ACPs.

- a. Run the following commands on the leader node.

```
nutanix@pcvm$ cd ~/cluster/bin/
```

- b. Do one of the following:

- » Download the script from the [Nutanix Downloads](#) page.

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/backup-scripts/3.8.0/iam_backup_script.sh
```

- » If your Prism Central does not have an internet connection, you can copy and paste the content of this file into a file on your Prism Central using an editor such as the vi editor.

- c. Run the following script.

```
nutanix@pcvm$ sh iam_backup_script.sh
```

The backup zipped file will be saved on this PC at /usr/local/nutanix/iam-backup.

5. Use the following command to find the name of the backup file.

```
nutanix@pcvm$ docker exec nucalm ls tmp/default
```

You can then use the following command to remove any files other than the backup file from the backup folder.

```
nutanix@pcvm$ docker exec nucalm rm tmp/default/<file name>
```

6. Run the following command to get the nucalm container ID.

```
nutanix@pcvm$ docker ps | grep nucalm
```

The first field farthest to the left displays the container ID as shown below:

```
<container id> nucalm:latest
```

7. Run the following command to copy the file from the container.

```
nutanix@pcvm$ docker cp <container id>:tmp/default/<backup file name> ~/tmp/
```

Where the `<container id>` is the nucalm container ID you got in the previous step and the `<backup file name>` is the backup file name you got in step 5.

8. Copy the file from the old Prism Central to the new Prism Central.

- » If you can connect the new and old Prism Central through networking, then run the following command on the old Prism Central to transfer the data.

```
nutanix@pcvm$ scp ~/tmp/<backup file name> <New Prism Central IP>:~/tmp/  
&& scp /usr/local/nutanix/iam-backup <New Prism Central IP>:/usr/local/nutanix/  
iam-backup
```

- » If you cannot connect the new and old Prism Central through networking, then use a tool such as WinSCP to transfer data between the old and new Prism Central.

9. Log on to the new Prism Central through SSH.

10. Run the following command to get the nucalm container ID.

```
nutanix@pcvm$ docker ps | grep nucalm
```

The first field farthest to the left displays the container ID as shown below:

```
<container id> nucalm:latest
```

11. Run the following command to copy the calmdata backup tar file into the nucalm container of the new Prism Central.

```
nutanix@pcvm$ docker cp ~/tmp/<Backup File Name> <nucalm_container_id>:tmp/default/
```

The command assumes that the backup file is located in the `/home/nutanix/tmp` directory. You can change the directory if you have stored the backup file in a different location.

What to do next

For information on restoring Self-Service data, see [Restoring Self-Service Data](#) on page 395.

Restoring Self-Service Data

You can restore the Self-Service data to a new Prism Central using a backup you took earlier.

About this task

For more information on backing up the Self-Service data, see [Backing up Self-Service Data](#) on page 393.

Note: When you restore your Self-Service data, ensure that:

- You do not have any running apps or blueprints on the Prism Central on which you restore the Self-Service data. Any apps or blueprints available on your Prism Central might not work properly after you restore the data.

- The Prism Central on which you restore the data has the same Prism Elements as that of the backed-up Prism Central. In case of any variations, the accounts or projects associated with your local Prism Element through the Prism Central might not work properly.
- The restored Prism Central is in the same network as that of the backed-up Prism Central for the Policy Engine flows to work properly. If the policy engine on your old setup was on a different network than that of your new Prism Central, you must back up and restore the policy engine database as well, along with the Self-Service data. For more information on backing up and restoring the policy engine database, see [Backing Up and Restoring Policy Engine Database](#) on page 399.
- The restored version of the Self-Service is the same as that of the backed-up version. For example, if you have taken a backup of version 3.0, you must restore using version 3.0. You cannot use version 3.1 or 3.2 to restore the Self-Service data.
- You have enabled Self-Service on the destination Prism Central, and the Self-Service instance is new.

For new Prism Central, you must consider the following points:

- Unregister your cluster with the old Prism Central and register it with your new Prism Central. For more information, see [Registering or Unregistering a Cluster with Prism Central](#) in the *Prism Central Infrastructure Guide*.
- Enable Self-Service on your Prism Central before restoring your Self-Service backup. For more information on enabling Self-Service, see [Marketplace](#) in the *Prism Center Admin Center Guide*.
- The policy engine is automatically enabled if the cluster already has the policy engine VM. The policy engine needs to be restored only if the old policy engine VM is not on the same network as that of the new Prism Central.

Procedure

1. Log on to the new Prism Central using the SSH session as a `nutanix` user.
2. Run the following command to determine the Prism Central leader node in a scale-out setup.

```
sudo kubectl -n ntnx-base get pods
```

In case of scale-out setup, the command returns an output on two out of three nodes. Use any of the two nodes that give an output as the leader node.

The Prism Central leader node returns the output based on the point of deliveries running on that Prism Central. If you do not see any output, log on to a different Prism Central IP address and run the command again.

```
$ ssh <Prism Central IP address>
```

You can run the following command to get the Prism Central IP addresses:

```
nutanix@pcvm$ svmips
```

Note: To complete the restore successfully, ensure that all point of deliveries are in the Running or Completed state. You can contact Nutanix Support if you have point of deliveries that are not in the Running or Completed state in the output.

3. To restore the backed up Calm data, do the following:
 - a. With the session, run the following command on the new Prism Central.

```
nutanix@pcvm$ docker exec -ti nucalm bash
activate;
```

```
cd /home/calm/bin/  
./calmdata restore --dump-folder <folder>
```

The default folder is located in the /tmp/default path. Replace the folder with the new folder.

- b. If the policy engine was enabled in the old Prism Central (or the Self-Service VM) from which you took the backup, then run the following command.

```
nutanix@pcvm$ docker exec -ti nucalm bash  
activate;  
code ;  
python scripts/update_policy_vm_host_data.pyc
```

4. Additionally, run the IAM restore script if you have an IAM-enabled setup.

- a. Log on to the new Prism Central using the SSH session as a `nutanix` user.
- b. Run the following commands to change to the correct directory.

```
nutanix@pcvm$ cd ~/cluster/bin/
```

- c. Do one of the following:

- » Download the script from the [Nutanix Downloads](#) page.

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/381-Files/  
iam_restore_script.sh
```

- » If your Prism Central does not have an internet connection, you can copy and paste the content of this file into a file on your Prism Central using an editor such as the vi editor.

- d. Run the following script.

```
nutanix@pcvm$ sh iam_restore_script.sh
```

5. If the policy engine was enabled in the old Prism Central (or the Self-Service VM) from which you took the backup, SSH to the new Prism Central and run the following commands to regenerate policy certificates:

```
nutanix@pcvm$ source .venvs/bin/bin/activate
```

```
nutanix@pcvm$ cd bin
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/backup-scripts/3.8.0/  
policy_rotate_cert.py
```

```
nutanix@pcvm$ python policy_rotate_cert.py
```

```
nutanix@pcvm$ rm -f policy_rotate_cert.py
```

Flag Options for Backup

Use the following flag options for your Self-Service data backup:

Table 57: Flag Options

Options	Description
dump-folder	<p>The folder where you want to place the backup data. The default folder is located at /tmp/default. Use this flag if you need to change the location of the stored data.</p> <p>Note: Create this folder before taking the backup. When you restore, the restore binary must be present at this location.</p> <p>Example:</p> <pre>nutanix@pcvm\$./calldata backup --dump-folder="/tmp/new"</pre>
max-threads	<p>The maximum number of threads to use to take the backup. The default value is 5.</p> <p>Example:</p> <pre>nutanix@pcvm\$./calldata backup --max-thread=5</pre>
fetch-limit	<p>The maximum number of entries to fetch in batches of 100 per call. The default and the maximum value is 100, except for the following entities.</p> <ul style="list-style-type: none">• "nucalm_task"• "nucalm_action"• "nucalm_run_log"• "nucalm_variable"• "nucalm_substrate_element" <p>Decreasing the value of fetch-limit increases the time taken to back up Self-Service data.</p> <p>Example:</p> <pre>nutanix@pcvm\$./calldata backup --fetch-limit=100</pre>
idf-timeout	<p>The timeout for IDF (database). Increase the value of IDF timeout if you encounter backup failure due to timeout. The default value is 60.</p> <p>Example:</p> <pre>nutanix@pcvm\$./calldata backup --idf-timeout=120</pre>
backup-deleted-entities	<p>The flag to include deleted entities in the backup. The backup does not include deleted entities when the value is False. The default value is True.</p> <p>Example:</p> <pre>nutanix@pcvm\$./calldata backup --backup-deleted-entities=false</pre>

Options	Description
dump-max-threads	<p>The flag is used when the files are written concurrently during the backup. You can limit the concurrency by specifying the value for this flag. The default value is 20.</p> <p>Example:</p> <pre>nutanix@pcvm\$./calldata backup --dump-max-threads=20</pre>
override-default-batch-size	<p>The flag is used to override the default batch size of all IDF entities to a specific batch size as stated by the fetch-limit flag. The batch size is 100 when the fetch-limit flag is not set.</p> <p>Example:</p> <pre>nutanix@pcvm\$# ./calldata backup --override-default-batch-size=true</pre> <p>The default value is False.</p>

Backing Up and Restoring Policy Engine Database

When you enable the policy engine for your Self-Service instance, Self-Service creates and deploys a new VM for the policy engine in your Prism Central network. After the policy engine VM deployment, you can anytime create a backup of your policy engine database. You can use the backup to restore the policy engine to the earlier state on your existing policy engine VM or on a new policy engine VM.

About this task

You must run the backup and restore commands from your Prism Central instance.

Procedure

1. Get the IP address of the Policy Engine VM.

For information on how to get the Policy Engine IP address, see the [Viewing Policy Engine VM Details](#) section in the *Prism Central Admin Center Guide*.

2. To create a backup of your policy engine, run the following command:

```
$ ssh nutanix@<policy_vm_ip> /home/nutanix/scripts/backup.sh
```

Where `<policy_vm_ip>` is the IP address of the policy engine VM.

This command creates a local backup on the policy engine VM at `/home/nutanix/data/backups/`.

Note: When you run the command to create the backup, the policy engine remains unavailable until the backup is created.

3. To view all the backups that are available on your policy engine VM, use the following command:

```
$ ssh nutanix@<policy_vm_ip> /home/nutanix/scripts/restore.sh --list
```

4. To restore the policy engine to a new policy engine VM, copy the backup to Prism Central using the `scp` command and then to the new policy engine VM.

```
nutanix@pcvm$ scp nutanix@<policy_vm_ip>:/home/nutanix/data/backups/<Backup File Name> ~/tmp/
```

```
nutanix@pcvm$ scp ~/tmp/<Backup File Name> nutanix@<new_policy_vm_ip>:/home/nutanix/data/backups/<Backup File Name>
```

5. To restore policy engine to its earlier state, run the following command:

```
$ ssh nutanix@<policy_vm_ip> /home/nutanix/scripts/restore.sh -f=<backup_name>
```

Where `<policy_vm_ip>` is the IP address of the policy engine VM and `<backup_name>` is the local backup file available on the policy engine VM.

Note: When you run the command to restore your policy engine on the existing policy engine VM, the policy engine remains unavailable until it is restored.

6. Run the following commands to restore the tunnel firewall rules for existing tunnel VMs on the Prism Central VM.

```
$ ssh <Prism Central IP address>
nutanix@pcvm docker exec -ti domain_manager bash
activate
update_policy_vm_host_data.py
```


PYTHON 2 DEPRECATION

Because the Python Software Foundation (PSF) discontinued support for Python 2, Nutanix is ending support for Python 2 on June 30, 2024. For more information on Python 2 deprecation by PSF and the need to move to Python 3, see the *Sunsetting Python 2* section in the *Python Documentation*.

How This Change Affects NCM Self-Service

- Until June 30, 2024, Self-Service will continue to support all Python 2 eScripts and all previous Python 2 modules.
- From version 3.7.2.1, Self-Service also supports Python 3 eScripts.
- To support Python 2 and Python 3 eScripts, Self-Service provides a version switcher to select the Python version for your scripts. The version switcher is available with the **Script** fields where you select **EScript** as the script type to enter or upload scripts. For example, you can use the version switcher while creating or updating blueprints or runbooks, updating tasks in the library, defining runbook references in credential providers, configuring dynamic variables, and so on.

Note: You can also use the version switcher in the Self-Service Playground while testing your scripts.

Recommendations

- Ensure that you update all eScript-based Set Variable tasks and Execute tasks, variables in day-2 actions, runbooks, blueprints, and task libraries.
- Ensure that all newer eScripts and the modules used in Set Variable tasks, Execute tasks, and eScript dynamic variables are compatible with Python 3.

You can manually update Python 2 syntax to Python 3 for your scripts.

Warning: Failure to update existing eScripts to Python 3 might result in errors in the entities involving those scripts after June 30, 2024.

How to Use the Python Version Switcher

- To use the version switcher, select **EScript** as the script type when you add or edit the script. You can then select the Python version for the Script field. By default, Python 3 is selected for the version switcher.

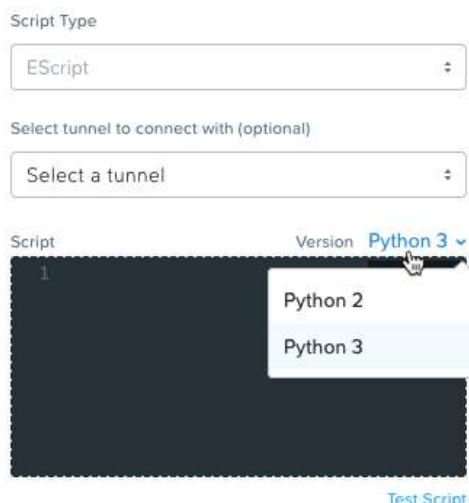


Figure 86: Script field - Version Switcher

How to Update Deployed Applications

- Single-VM Applications

For your deployed single-VM applications, use the app edit option to update Python 2 scripts to Python 3. For more information on app edit, see [Update Configuration for VM](#).

Note:

- You can update a single-VM application with custom actions having dynamic variables with secrets using the Self-Service DSL only. For more information on using Self-Service DSL to update scripts, see [Updating eScripts in Applications](#) on page 410.
- You can update snapshot configuration tasks or variables using Self-Service DSL only.
- Self-Service does not support patch configuration action updates in Single-VM and Multi-VM applications either through the user interface or the Self-Service DSL.

- Multi-VM Applications

For your multi-VM applications, use the Self-Service DSL to update your scripts to Python 3. For more information on using Self-Service DSL to update the scripts, see [Updating eScripts in Applications](#) on page 410.

EScripts Update in Blueprints, Runbooks, Tasks, and Marketplace Items

Nutanix recommends you to use a development instance of Self-Service for updating eScripts from Python 2 to Python 3.

To update eScripts from Python 2 to Python 3 in blueprints, runbooks, tasks, and marketplace, follow the steps from one of the following scenarios that matches your existing setup.

Table 58: Specific Scenarios

Scenarios	Information
When you have a development instance of Self-Service VM	For information, see Updating eScripts When You Have a Development Instance of Self-Service VM on page 403.
When you do not have a development instance of Self-Service VM	For information, see Updating eScripts When You Do Not Have a Development Instance of Self-Service VM on page 405.
When you have a development instance of Self-Service in Prism Central	For information, see Updating eScripts When You Have a Development Instance of Self-Service in Prism Central on page 406.
When you only have a production instance of Self-Service in Prism Central	For information, see Updating eScripts When You Have a Production Instance of Self-Service in Prism Central on page 408.

Basic Pointers for Python 3

You can use the following basic pointers when you are updating your scripts to Python 3.

- Print is now a function and must be used as `print("text")`.
- Unlike Python 2, bytes are not treated as strings in Python 3. So whenever there is encoding, during `json.dumps` use `reject_bytes=False` param. For example,

```
json.dumps(data, reject_bytes=False)
```
- `base64.urlsafe_b64encode` supports bytes as input. Therefore, you must encode the string before passing. For example,

```
base64.urlsafe_b64encode("sample_string".encode())
```
- In Python 2, unicode strings are prefixed with a `u'...'` because the strings are stored as ASCII. However, in Python 3, all strings are unicode by default.
- In Python 3, division operator `/` is a floating-point division.

For more information on how to write Python 2 to Python 3 compatible codes, see the *Writing Python 2-3 Compatible Code* section in the *Python Documentation*.

Updating eScripts When You Have a Development Instance of Self-Service VM

About this task

Use the following steps to update eScripts in NCM Self-Service blueprints, runbooks, tasks, and marketplace items when you have a development instance of Self-Service VM.

Procedure

1. Upgrade your development instance to Self-Service VM version 3.7.2.1.
For more information, see [Self-Service VM Upgrades](#) on page 427.
2. Connect your development instance to the [Self-Service DSL environment](#).

3. Decompile the blueprint using the following command:

```
calm decompile bp <blueprint_name>
```

For example, if you have a blueprint, BasicLinuxVM, with Python 2 eScript tasks in it, then use the following command:

```
calm decompile bp BasicLinuxVM
```

The blueprint is decompiled to the scripts folder, specs folder, and the blueprint.py script file.

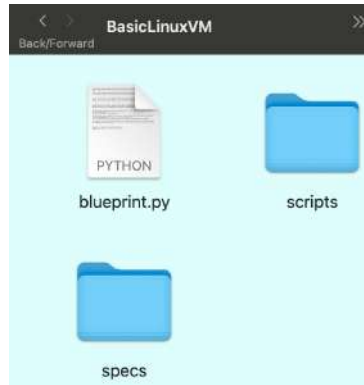


Figure 87: Decompile Blueprint

4. Navigate to the Python 2 eScript that needs to be updated.
5. Update the Python 2 eScript task type and the corresponding script in the scripts folder to Python 3.
6. In the blueprint.py file, replace all occurrences of `CalmTask.Setvariable.escript.py2` to `CalmTask.Setvariable.escript.py3` Or `CalmTask.Exec.escript.py2` to `CalmTask.Exec.escript.py3`.

```
)
CalmTask.SetVariable.escript.py2(
    name="Task1",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["appVar"],
)

CalmTask.SetVariable.escript.py2(
    name="Task2",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["var2"],
)
```

Figure 88: Script Update

Note: You must also update the syntax of the script referenced in these functions to Python 3.

For basic pointers that you can use while updating your scripts to Python 3, see [EScripts Update in Blueprints, Runbooks, Tasks, and Marketplace Items](#) on page 402.

7. Compile and recreate the blueprint with the updated Python 3 eScript code using the following command:

```
calm compile bp --file ./BasicLinuxVM/blueprint.py
calm create bp --file ./BasicLinuxVM/blueprint.py --name BasicLinuxVM_Py3Update
```

What to do next

Note: After you successfully update your blueprints from Python 2 to Python 3, ensure that you retain all necessary backups before you upgrade your production instance to pc.2023.4 (Self-Service to version 3.7.2.1) or later versions.

Updating eScripts When You Do Not Have a Development Instance of Self-Service VM

Use the following steps to update eScripts in NCM Self-Service blueprints, runbooks, tasks, and marketplace items when you do not have a development instance of Self-Service VM.

Procedure

1. Backup the existing production instance of your Self-Service VM.
For more information, see [Backing up Self-Service Data](#) on page 393.
2. Deploy a new standalone development instance of Self-Service VM of the same version as production.
3. Using the restore functionality, import the entities into the development instance you created in the previous step.
For more information, see [Restoring Self-Service Data](#) on page 395.
4. Upgrade the development instance to Self-Service VM 3.7.2.1.
5. Connect your development instance to the [Self-Service DSL environment](#).
6. Decompile the blueprint using the following command:

```
calm decompile bp <blueprint_name>
```

For example, if you have a blueprint, BasicLinuxVM, with Python 2 eScript tasks in it, then use the following command:

```
calm decompile bp BasicLinuxVM
```

The blueprint is decompiled to the scripts folder, specs folder, and the blueprint.py script file.

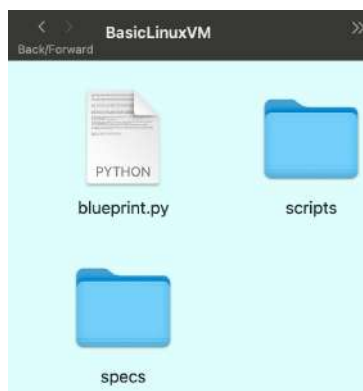


Figure 89: Decompiled Blueprint

7. Navigate to the Python 2 eScript that needs to be updated.
8. Update the Python 2 eScript task type and the corresponding script in the scripts folder to Python 3.
9. In the blueprint.py file, replace all occurrences of `CalmTask.Setvariable.escript.py2` to `CalmTask.Setvariable.escript.py3` or `CalmTask.Exec.escript.py2` to `CalmTask.Exec.escript.py3`.

```

)
CalmTask.SetVariable.escript.py2(
    name="Task1",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["appVar"],
)

CalmTask.SetVariable.escript.py2(
    name="Task2",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["var2"],
)

```

Figure 90: Script Update

Note: You must also update the syntax of the script referenced in these functions to Python 3.

For basic pointers that you can use while updating your scripts to Python 3, see [EScripts Update in Blueprints, Runbooks, Tasks, and Marketplace Items](#) on page 402.

10. Compile and recreate the blueprint with the updated Python 3 eScript code using the following command:

```

calm compile bp --file ./BasicLinuxVM/blueprint.py
calm create bp --file ./BasicLinuxVM/blueprint.py --name BasicLinuxVM_Py3Update

```

What to do next

Note: After you successfully update your blueprints from Python 2 to Python 3, ensure that you retain all necessary backups before you upgrade your production instance to pc.2023.4 (Self-Service to version 3.7.2.1) or later versions.

Updating eScripts When You Have a Development Instance of Self-Service in Prism Central

Use the following steps to update eScripts in NCM Self-Service blueprints, runbooks, tasks, and marketplace items when you have a development instance of Self-Service running within the Prism Central.

Procedure

1. Upgrade the development instance to Self-Service 3.7.2.1.
2. Connect your development instance to the [Self-Service DSL environment](#).

3. Decompile the blueprint using the following command:

```
calm decompile bp <blueprint_name>
```

For example, if you have a blueprint, BasicLinuxVM, with Python 2 eScript tasks in it, then use the following command:

```
calm decompile bp BasicLinuxVM
```

The blueprint is decompiled to the scripts folder, specs folder, and the blueprint.py script file.

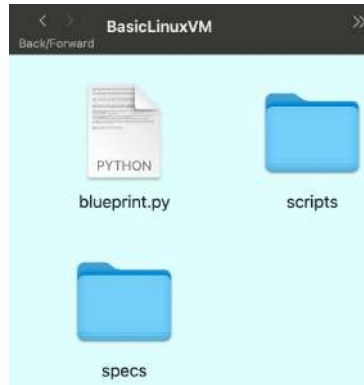


Figure 91: Decomplied Blueprint

4. Navigate to the Python 2 eScript that needs to be updated.
5. Update the Python 2 eScript task type and the corresponding script in the scripts folder to Python 3.
6. In the blueprint.py file, replace all occurrences of `CalmTask.Setvariable.escript.py2` to `CalmTask.Setvariable.escript.py3` Or `CalmTask.Exec.escript.py2` to `CalmTask.Exec.escript.py3`.

```
)
CalmTask.SetVariable.escript.py2(
    name="Task1",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["appVar"],
)

CalmTask.SetVariable.escript.py2(
    name="Task2",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["var2"],
)
```

Figure 92: Script Update

Note: You must also update the syntax of the script referenced in these functions to Python 3.

For basic pointers that you can use while updating your scripts to Python 3, see [EScripts Update in Blueprints, Runbooks, Tasks, and Marketplace Items](#) on page 402.

7. Compile and recreate the blueprint with the updated Python 3 eScript code using the following command:

```
calm compile bp --file ./BasicLinuxVM/blueprint.py
calm create bp --file ./BasicLinuxVM/blueprint.py --name BasicLinuxVM_Py3Update
```

What to do next

Note: After you successfully update your blueprints from Python 2 to Python 3, ensure that you retain all necessary backups before you upgrade your production instance to pc.2023.4 (Self-Service to version 3.7.2.1) or later versions.

Updating eScripts When You Have a Production Instance of Self-Service in Prism Central

Use the following steps to update eScripts in NCM Self-Service blueprints, runbooks, tasks, and marketplace items when you have a production instance of Self-Service running within the Prism Central.

Procedure

1. Backup the existing production instance of your Prism Central.
For more information, see [Backing Up Prism Central](#).
2. Deploy a development instance of Prism Central of the same version as production with the Self-Service enabled.
3. Using the restore functionality, restore the backed up Prism Central into the development instance you created in the previous step.
For more information, see [Restoring Prism Central](#).
4. Upgrade the development instance to Prism Central version pc.2023.4 (Self-Service to version 3.7.2.1).
5. Connect your development instance to the [Self-Service DSL environment](#).

6. Decompile the blueprint using the following command:

```
calm decompile bp <blueprint_name>
```

For example, if you have a blueprint, BasicLinuxVM, with Python 2 eScript tasks in it, then use the following command:

```
calm decompile bp BasicLinuxVM
```

The blueprint is decompiled to the scripts folder, specs folder, and the blueprint.py script file.

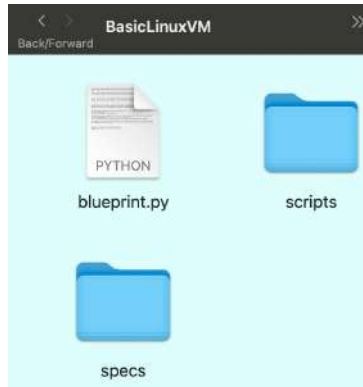


Figure 93: Decomplied Blueprint

7. Navigate to the Python 2 eScript that needs to be updated.
8. Update the Python 2 eScript task type and the corresponding script in the scripts folder to Python 3.
9. In the blueprint.py file, replace all occurrences of `CalmTask.Setvariable.escript.py2` to `CalmTask.Setvariable.escript.py3` or `CalmTask.Exec.escript.py2` to `CalmTask.Exec.escript.py3`.

```
)
CalmTask.SetVariable.escript.py2(
    name="Task1",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["appVar"],
)

CalmTask.SetVariable.escript.py2(
    name="Task2",
    filename=os.path.join(
        "scripts", "Substrate_VM1_Action__pr
    ),
    target=ref(VM1),
    variables=["var2"],
)
```

Figure 94: Script Update

Note: You must also update the syntax of the script referenced in these functions to Python 3.

For basic pointers that you can use while updating your scripts to Python 3, see [EScripts Update in Blueprints, Runbooks, Tasks, and Marketplace Items](#) on page 402.

10. Compile and recreate the blueprint with the updated Python 3 eScript code using the following command:

```
calm compile bp --file ./BasicLinuxVM/blueprint.py
calm create bp --file ./BasicLinuxVM/blueprint.py --name BasicLinuxVM_Py3Update
```

11. When you upgrade your production instance of Prism Central, ensure that you upgrade your development instance to the same version as that of your production instance.
12. Backup the development instance of your Prism Central.
13. Using the restore functionality, restore the development instance into the production instance.

What to do next

Note: After you successfully update your blueprints from Python 2 to Python 3, ensure that you retain all necessary backups before you upgrade your production instance to pc.2023.4 (Self-Service to version 3.7.2.1) or later versions.

Updating eScripts in Applications

You can use the following steps to update eScripts in the tasks of your deployed applications from Python 2 to Python 3 using Self-ServiceDSL.

About this task

Use the following Self-Service DSL steps to update application scripts to Python 3 for multi-VM applications. For Single-VM applications, use the app edit option to update application scripts to Python 3.

For more information on Self-Service DSL, see [Self-Service DSL Overview](#) on page 37 or [Self-Service DSL](#) on [Nutanix.dev](#).

Note:

- Single-VM applications with custom actions having dynamic variables with secrets can be updated using Self-Service DSL only.
- Snapshot configuration tasks or variables can be updated using Self-Service DSL only.
- Patch configuration action updates are not currently supported in either Single-VM or Multi-VM applications. As a workaround, you can soft delete and import (brownfield import) applications.

Procedure

1. Log in to your Self-Service DSL environment.
2. Describe the application that needs to be modified using the following command:

```
calm describe app-migratable-entities <app_name>
```

Where `<app_name>` is the name of the deployed application in which you can update the script from Python 2 to Python 3.

3. Decompile the application blueprint using the following command:

```
calm decompile app-migratable-bp <app_name> --dir <dir_name>
```

Where `<dir_name>` is an optional argument for the directory name to which you want to decompile the application blueprint. If the directory does not exist yet, the "calm decompile" command will create it automatically.

4. Edit the blueprint to update the task to use Python 3 scripts. For more information on this step, see [Updating Custom Action Scripts of a Multi-VM Application - Example](#) on page 411.
5. Update the application using the following command:

```
calm update app-migratable-bp <app_name> -f <blueprint_file_location_path>
```

Where `<blueprint_file_location_path>` is the location of the decompiled blueprint file (the blueprint file in which you updated tasks to use Python 3 scripts).

For example,

```
calm update app-migratable-bp <app_name> -f <dir_name>/blueprint.py
```

Updating Custom Action Scripts of a Multi-VM Application - Example

The following steps provide an example of how you can update custom action scripts of a deployed multi-VM application.

About this task

Suppose you have a deployed Multi-VM app with a custom action with the following details:

- Application name: **azure-postgres-server**
- Custom action name: **CustomAction1**
- An Execute task of type Python 2 eScript within the custom action:

```
#py2
text = "Sample Runbook with Escript Task"
encoded_text = base64.b64encode(text)
print "encoded string is: ", encoded_text
```



Figure 95: Multi-VM App with Custom Action

Do the following to update the script in the custom action to Python 3 using Self-Service DSL.

Procedure

1. Use the following command to display the eScript tasks or variables that you can migrate.

```
calm describe app-migratable-entities azure-postgres-server
```

Where `azure-postgres-server` is the application name.

2. Decompile the application blueprint using the following command.

```
calm decompile app-migratable-bp azure-postgres-server --dir appeditescrypt
```

Where `appeditescrypt` is the directory to which the application blueprint is decompiled. The decompiled blueprint only contains eScript tasks and variables.

3. Navigate to your working directory using the following command:

```
cd appeditescrypt/
```

4. To update the function reference, open the file `blueprint.py` and update the reference from "CalmTask.Exec.escript.py2" to "CalmTask.Exec.escript.py3".

```
CalmTask.Exec.escript.py3(  
    name="B64 usage task",  
    filename=os.path.join(  
        "scripts", "Profile_Default_Action_CustomAction1_Task_B64usagetask.py"  
    ),  
    target=ref(Service1),
```

5. Navigate to the scripts subdirectory using the following command:

```
cd scripts/
```

6. Open the file "Profile_Default_Action_CustomAction1_Task_B64usagetask.py" to update the existing Python 2 script to Python 3.

```
#py3  
text = "Sample Runbook with Escript Task"  
encoded_text = base64.b64encode(text.encode())  
print("encoded string is: ", encoded_text.decode())
```

7. Navigate to the directory from where you started.

```
cd ../../
```

8. Run the following command to update the application.

```
calm update app-migratable-bp azure-postgres-server -f appeditescrypt/blueprint.py
```

The following output indicates that the task has been updated in the application:

```
Application update is successful. Got Action Runlog uuid:  
40ba0cd0-60cc-4d01-97ff-3ad3516e3b7b  
Update messages:  
[App_profile.Default.Action.Custom Action1] Task 'B64  
usage task' updated in Application
```

This command updates only the scripts of the tasks or variables in the application.

9. Open the application and verify the updates.

SELF-SERVICE SCRIPTS

Self-Service scripts refer to the automation scripts that are used to define or customize the behavior of application blueprints. Self-Service scripts serve several purposes, such as:

- **Infrastructure Provisioning:** You can use scripts to define the tasks that are required to provision the underlying infrastructure resources needed for applications, such as creating virtual machines, configuring networking, or setting up storage.
- **Application Configuration:** You can use scripts to install and configure software components within the application environment.
- **Customization and Integration:** You can use scripts to customize the behavior of an application blueprint to match your specific requirements. You can incorporate custom logic, workflows, or integration with external systems using APIs or other protocols.
- **Life cycle Management:** You can use scripts to define the actions to be performed during the life cycle of an application, such as starting, stopping, scaling, or updating the application.

Note: For successful execution of scripts on a Linux based endpoint or VM, ensure that the target endpoint or VM has SFTP enabled.

Sample Scripts for Installing and Uninstalling Services

Self-Service task library public repository contains scripts for installing and uninstalling different services. Go to the [Nutanix Task Library Github location](#) to access the repository.

Sample Scripts to Configure Non-Managed AHV Network

The following sections provide the sample scripts of Cloud-init and SysPrep to configure the static IP address range for non-managed AHV network.

Cloud-init Script for Linux

Note: You can assign a static IP to a non-managed network only when the disk image contains a network card set for the static IP. You cannot assign the static IP if the NIC is configured for DHCP in the disk image.

```
#cloud-config
cloud_config_modules:
  - resolv_conf
  - runcmd
write_files:
  - path: /etc/sysconfig/network-scripts/ifcfg-eth0
    content: |
      IPADDR=10.136.103.226
      NETMASK=255.255.255.0
      GATEWAY=10.136.103.1
      BOOTPROTO=none
      ONBOOT=yes
      DEVICE=eth0
runcmd:
  - [ifdown, eth0]
  - [ifup, eth0]
manage_resolv_conf: true
resolv_conf:
  nameservers: ['8.8.4.4', '8.8.8.8']
```

```
searchdomains:
  - foo.example.com
  - bar.example.com
domain: example.com
options:
  rotate: true
  timeout: 1
```

SysPrep Script for Windows

```
<?xml version="1.0" encoding="UTF-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="specialize">
    <component xmlns:wcm="http://schemas.microsoft.com/WMICConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS">
      <ComputerName>Windows2016</ComputerName>
      <RegisteredOrganization>Nutanix</RegisteredOrganization>
      <RegisteredOwner>Acropolis</RegisteredOwner>
      <TimeZone>UTC</TimeZone>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMICConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      UnattendedJoin" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
      language="neutral" versionScope="nonSxS">
      <Identification>
        <Credentials>
          <Domain>contoso.com</Domain>
          <Password>secret</Password>
          <Username>Administrator</Username>
        </Credentials>
        <JoinDomain>contoso.com</JoinDomain>
        <UnsecureJoin>false</UnsecureJoin>
      </Identification>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMICConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-TCPIP"
      processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral"
      versionScope="nonSxS">
      <Interfaces>
        <Interface wcm:action="add">
          <Identifier>Ethernet</Identifier>
          <Ipv4Settings>
            <DhcpEnabled>false</DhcpEnabled>
            <RouterDiscoveryEnabled>true</RouterDiscoveryEnabled>
            <Metric>30</Metric>
          </Ipv4Settings>
          <UnicastIpAddresses>
            <IpAddress wcm:action="add" wcm:keyValue="1">10.0.0.2/24</IpAddress>
          </UnicastIpAddresses>
          <Routes>
            <Route wcm:action="add">
              <Identifier>10</Identifier>
              <Metric>20</Metric>
              <NextHopAddress>10.0.0.1</NextHopAddress>
              <Prefix>0.0.0.0/0</Prefix>
            </Route>
          </Routes>
        </Interface>
      </Interfaces>
    </component>
```

```

    <component xmlns:wcm="http://schemas.microsoft.com/WMICConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
DNS-Client" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
    <UseDomainNameDevolution>true</UseDomainNameDevolution>
    <DNSDomain>contoso.com</DNSDomain>
    <Interfaces>
        <Interface wcm:action="add">
            <Identifier>Ethernet</Identifier>
            <DNSDomain>contoso.com</DNSDomain>
            <DNSServerSearchOrder>
                <IpAddress wcm:action="add" wcm:keyValue="1">10.0.0.254</IpAddress>
            </DNSServerSearchOrder>
            <EnableAdapterDomainNameRegistration>true</
EnableAdapterDomainNameRegistration>
            <DisableDynamicUpdate>true</DisableDynamicUpdate>
        </Interface>
    </Interfaces>
</component>
    <component xmlns="" name="Microsoft-Windows-TerminalServices-LocalSessionManager"
publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS"
processorArchitecture="amd64">
        <fDenyTSConnections>>false</fDenyTSConnections>
    </component>
    <component xmlns="" name="Microsoft-Windows-TerminalServices-RDP-
WinStationExtensions" publicKeyToken="31bf3856ad364e35" language="neutral"
versionScope="nonSxS" processorArchitecture="amd64">
        <UserAuthentication>0</UserAuthentication>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMICConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Networking-MPSSVC-Svc"
processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral"
versionScope="nonSxS">
        <FirewallGroups>
            <FirewallGroup wcm:action="add" wcm:keyValue="RemoteDesktop">
                <Active>true</Active>
                <Profile>all</Profile>
                <Group>@FirewallAPI.dll,-28752</Group>
            </FirewallGroup>
        </FirewallGroups>
    </component>
</settings>
    <settings pass="oobeSystem">
        <component xmlns:wcm="http://schemas.microsoft.com/WMICConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
            <UserAccounts>
                <AdministratorPassword>
                    <Value>secret</Value>
                    <PlainText>true</PlainText>
                </AdministratorPassword>
            </UserAccounts>
            <AutoLogon>
                <Password>
                    <Value>secret</Value>
                    <PlainText>true</PlainText>
                </Password>
                <Enabled>true</Enabled>
                <Username>Administrator</Username>
            </AutoLogon>
            <FirstLogonCommands>

```

```

        <SynchronousCommand wcm:action="add">
            <CommandLine>cmd.exe /c netsh firewall add portopening TCP 5985 "Port
5985"</CommandLine>
            <Description>Win RM port open</Description>
            <Order>1</Order>
            <RequiresUserInput>true</RequiresUserInput>
        </SynchronousCommand>
    </FirstLogonCommands>
    <OOBE>
        <HideEULAPage>true</HideEULAPage>
        <SkipMachineOOBE>true</SkipMachineOOBE>
    </OOBE>
</component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIconfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
International-Core" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
        <InputLocale>en-US</InputLocale>
        <SystemLocale>en-US</SystemLocale>
        <UILanguageFallback>en-us</UILanguageFallback>
        <UILanguage>en-US</UILanguage>
        <UserLocale>en-US</UserLocale>
    </component>
</settings>
</unattend>

```

Supported eScript Modules and Functions

Self-Service supports the following eScript modules.

Table 59: Supported eScript Modules with Python 3

Module	Module supported as (with Python 3)	Module supported as (with Python 2)
datetime	_datetime	_datetime
re	re	re
difflib	difflib	difflib
base64	base64	base64
pprint	pprint	Not supported
simplejson	json	Not supported
ujson	ujson	Not supported
yaml	yaml	Not supported
uuid	uuid	Not supported
requests	requests	requests
boto3	boto3	boto3
azure	azure	azure
googleapiclient	google	google

Module	Module supported as (with Python 3)	Module supported as (with Python 2)
kubernetes	kubernetes	kubernetes

The following example displays the usage of the boto3 module.

```
import boto3
ec2 = boto3.client('ec2', aws_access_key_id='{}', aws_secret_access_key='{}',
    region_name='us-east-1')
print(ec2.describe_regions())
```

The following example displays the usage of the Azure module with Python 2.

```
# subscription_id macro contains your Azure Subscription ID
# client_id macro contains your Client ID
# tenant macro contains you Tenant ID
from azure.common.credentials import ServicePrincipalCredentials
from azure.mgmt.resource import ResourceManagementClient
credentials = ServicePrincipalCredentials(
    client_id=@@{client_id}@@,
    secret='secret',
    tenant=@@{tenant}@@
)
client = ResourceManagementClient(credentials, @@{subscription_id}@@)
for item in client.resource_groups.list():
    print(item)
```

The following example displays the usage of the Azure module with Python 3.

```
# subscription_id macro contains your Azure Subscription ID
# client_id macro contains your Client ID
# tenant macro contains you Tenant ID
from azure.identity import ClientSecretCredential
from azure.mgmt.resource import ResourceManagementClient
credentials = ClientSecretCredential(client_id=@@{client_id}@@,
    client_secret='secret', tenant_id=@@{tenant_id}@@)
client = ResourceManagementClient(credentials,
    subscription_id=@@{subscription_id}@@)
for item in client.resource_groups.list():
    print(item)
```

The following example displays the usage of the GCP module with Python 2 or Python 3.

```
from google.oauth2 import service_account
svc_account_info = {
    "client_email": "@@{cred_gcp.client_email}@@",
    "private_key": "@@{cred_gcp.private_key}@@", # JSON keyfile - Use SSH Key
    credential to paste JSON keyfile
    "project_id": "@@{cred_gcp.project_id}@@",
    "token_uri": "@@{cred_gcp.token_uri}@@",
}
# Authentication
credentials = service_account.Credentials.from_service_account_info(svc_account_info)

# Create client - compute API https://cloud.google.com/compute/docs/reference/rest/v1
from googleapiclient import discovery
client = discovery.build('compute', 'v1', credentials=credentials)
# Compute machineImages API https://cloud.google.com/compute/docs/reference/rest/v1/
machineImages/list
request = client.machineImages().list(project=svc_account_info['project_id'])
while request is not None:
    response = request.execute()
    for image in response['items']:
```

```

    print(image)
    request = client.machineImages().list_next(previous_request=request,
previous_response=response)

```

The following example displays the usage of the Kubernetes module.

```

from kubernetes import client as k8client
#Define the bearer token to authenticate
#See Kubernetes Documentation for information on how to create the token
configuration=k8client.Configuration()
configuration.host="https://<hostip>:6443"
configuration.verify_ssl=False
configuration.debug=True
configuration.api_key={"authorization":"Bearer "+ aToken}
k8client.Configuration.set_default(configuration)
v1=k8client.CoreV1Api()
nodes=v1.list_node(watch=False)
print(nodes.items[0].metadata.name)

```

Self-Service supports the following eScript functions.

urlreq

The API exposes REST interface as a set of objects. This action is implemented using python requests module.

```

urlreq(url, verb='GET', auth=None, c=None, user=None, passwd=None, params=None,
headers=None, timeout=None, send_form_encoded_data=True, allow_redirects=True,
cookies=None, verify=True, proxies=None)

```

requests.Response object is returned.

Table 60: Arguments

Arguments	Description
url	string, url to request
verb	string, verb is GET by default. POST, HEAD, PUT, PATCH, and DELETE are other valid entries.
auth	<p>string (optional), BASIC and DIGEST are the valid entries.</p> <p>For authentication purposes, the order is as follows.</p> <ul style="list-style-type: none"> • username and password is authenticated by using user and passwd fields. • username and password is authenticated by using name of credential supplied using c field. • username and password is authenticated by using credential attached to the task.
user	string (optional), username used for authentication.
passwd	string (optional), password used for authentication.

Arguments	Description
params	dict (optional), if verb is GET, HEAD or DELETE, parameters are sent in the query string for the request otherwise they are sent in the body of the request.
headers	dict (optional), Dictionary of HTTP headers needs to be send along with the request.
timeout	integer (optional), you can configure requests to stop waiting for a response after a given number of seconds with the timeout parameter. timeout only elects the connection process itself, not the downloading of the response body.
send_form_encoded_data	boolean (optional), = True by default. If False, parameters dict is first dumped using simplejson.dumps() and then passed as a string.
allow_redirects	boolean (optional), = True by default. Specifies whether redirects should be allowed or not.
cookies	dict (optional), cookies dict to be sent along with the request.
verify	boolean (optional), = True by default. Specifies whether SSL certificates should be verified or not.
proxies	dict (optional), Dictionary mapping protocol to the URL of the proxy

Rules for authentication in the order of priority.

- If they are not **None**, use **user** and **passwd** fields.
- If c is not **None**, authenticate username and password from the credential name supplied.
- If the above two criteria does not match, username and password are authenticated by using the credential attached to the task.

For example

```
params = {'limit': 1}
headers = {'content-type': 'application/octet-stream'}
r = urlreq(url, verb="GET", auth="BASIC", c='somecred', params=params, headers=headers)
r = urlreq(url, verb="POST", auth="BASIC", user="user", passwd="pass", params=params)
```

exit

The exit function is an alias for sys.exit of python standard library.

```
exit(exitcode)
```

For example

```
exit(0)
```

sleep

The sleep function is an alias for time.sleep.

```
sleep(num_of_secs)
```

For example

```
sleep(10)
```

get_sql_handle

The get_sql_handle function enables you to remotely connect and manage SQL Servers. It is implemented by using python pymssql module.

```
get_sql_handle(server, username, password, database='', timeout=0, login_timeout=60,
charset='UTF-8', as_dict=False, host='', appname=None, port='1433',
conn_properties=None, autocommit=False, tds_version=None)
```

Returns pymssql.Connection object

Table 61: Arguments

Argument	Description
server (str)	database host
user (str)	database user to connect as
password (str)	user's password
database (str)	The database to initialize the connection with. By default SQL Server selects the database which is set as default for specific user
timeout (int)	query timeout in seconds, default 0 (no timeout)
login_timeout (int)	timeout for connection and login in seconds, default is 60 seconds
charset (str)	character set with which to connect to the database

For example

```
username="dbuser"
password="myP@ssworD"
server="10.10.10.10"
port="1433"

cnxn = get_sql_handle(server, username, password, port=port, autocommit=True)
cursor = cnxn.cursor()

# List all databases
cursor.execute("""
    SELECT Name from sys.Databases;
""")
for row in cursor:
    print(row[0])

cnxn.close()
```

EScript Sample Script

The following script is an EScript sample script compatible with Python3.

Note: Ensure that your script starts with `#script`.

```
#script
account_name = "@@{ACCOUNT_NAME}@"
aviatrix_ip = "@@{address}@"
new_test_password = "@@{NEW_TEST_PASSWORD}@"
vpc_name = "Test"

api_url = 'https://{0}/v1/api'.format(aviatrix_ip)
#print(api_url)

def setconfig(api_url, payload):
    r = urlreq(api_url, verb='POST', auth="BASIC", user='admin', passwd='passwd',
    params=payload, verify=False)
    resp = json.loads(r.content)
    if resp['return']:
        return resp
    else:
        print("Post request failed", r.content)
        exit(1)

print("Get the session ID for making API operations")
payload = {'action': 'login', 'username': 'admin', 'password': new_test_password}
api_url1 = api_url + "?action=login&username=admin&password="+ new_aviatrix_password
cid = setconfig(api_url=api_url1, payload=payload)
cid = cid['CID']
print cid

print("Delete the gateway")
payload = {'CID': cid,
    'action': 'delete_container',
    'account_name': account_name,
    'cloud_type': 1,
    'gw_name': vpc_name
}
api_url1 = api_url + "?CID="+cid+"&action=delete_container&account_name="+account_name
+"&cloud_type=1&gw_name="+vpc_name
print(setconfig(api_url=api_url1,payload=payload))

print("Delete the aws account")

payload = {'CID': cid,
    'action': 'delete_account_profile',
    'account_name': account_name
}
api_url1 = api_url + "?CID="+cid
+"&action=delete_account_profile&account_name="+account_name
print(setconfig(api_url=api_url1,payload=payload))
```

JWT Usage Sample Script

The following script is a jwt usage sample script compatible with Python 3.

Note: Ensure that your script starts with `#script`.

```
#script
jwt = '@@{calm_jwt}@'
```

```

payload = {}
api_url = 'https://@{pc_ip}@:9440/api/nutanix/v3/apps/list'
headers = {'Content-Type': 'application/json', 'Accept': 'application/json',
  'Authorization': 'Bearer {}'.format(jwt)}
r = urlreq(api_url, verb='POST', params=json.dumps(payload), headers=headers,
  verify=False)
if r.ok:
    resp = json.loads(r.content)
    print(resp)
    exit(0)
else:
    print("Post request failed", r.content)
    exit(1)

```

Sample Powershell Script

The following script is a powershell sample script.

```

Install-PackageProvider -Name NuGet -Force
Install-Module DockerMsftProvider -Force
Install-Package Docker -ProviderName DockerMsftProvider -Force

```

Sample Auto Logon and First Logon Scripts

Sample Auto Logon Script

The following script is a guest customization sample script for the Azure service.

```

<AutoLogon>
  <Password>
    <Value>@{user.secret}@</Value>
    <PlainText>true</PlainText>
  </Password>
  <Enabled>true</Enabled>
  <Username>@{user.username}@</Username>
</AutoLogon>

```

Sample First Logon Script

The following script is a guest customization sample script for the Azure service.

```

<FirstLogonCommands>
  <SynchronousCommand>
    <CommandLine>cmd.exe /c powershell -Command get-host</CommandLine>
    <Order>1</Order>
  </SynchronousCommand>
</FirstLogonCommands>

```

Sample Guest Customization Scripts for VMware and GCP Services

The following script is a guest customization sample script for the VMware service.

```

cmd.exe /c winrm quickconfig -q
cmd.exe /c winrm set winrm/config/service/auth @{Basic="true"}
powershell -Command "enable-psremoting -Force"
powershell -Command "Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Force"

```

The following script is a guest customization sample script for the GCP service.

```

#!/bin/bash\napt-get update\napt-get install -y apache2\ncat <<EOF > /var/www/html/
index.html\n<html><body><h1>Hello World</h1>\n<p>This page was created from a simple
startup script!</p>\n</body></html>\nEOF

```

Self-Service Blueprints Public Repository

Self-Service blueprints public repository contains custom blueprints and custom scripts that are created and published by community members. Self-Service also publishes official blueprints and tasks to the github public repository. You can clone the published blueprints and scripts and use from the repository. Go to the [Nutanix Blueprint Github](#) location to access the repository.

Seeding Scripts to the Self-Service Task Library

The blueprints repository of Self-Service contains script that can be seeded into task library and published to projects. You can use these tasks for blueprint configuration.

Procedure

1. Clone the blueprint repository from the Github. Go to the [Nutanix Blueprint Github](#) location to access the repository.
2. Change the directory to `calm-integrations/generate_task_library_items`.
3. To execute the script to seed, run the following command in bash.

```
bash generate_task_library_items.sh
```

4. Enter the following information.

- **Prism Central IP:** Enter the Prism Central IP address to which the task library items are to be seeded.
- **Prism Central User:** Enter the user name with the access to create task library scripts.
- **Prism Central Password:** Enter the password of the Prism Central user.
- **Prism Central Project:** Enter the Project name to which the task library items can be published.

5. (Optional) To avoid giving inputs multiple times, run the following command to export environment variables before running the script.

```
export PC_IP=<prism central IP>
export PC_USER=<prism central user>
export PC_PASSWORD=<prism central password>
export PC_PROJECT=<prism central project>
```

6. Run the following command to seed individual files into Self-Service.

```
python generate_task_library.py --pc $PC_IP--user $PC_USER --password $PC_PASSWORD --project $PC_PROJECT --script <path of script>
```

LICENSING AND UPGRADES IN SELF-SERVICE

The topics in this section cover the Self-Service licensing and Self-Service upgrade-related information.

Self-Service Licensing

The Self-Service license for Prism Central enables you to manage VMs that are provisioned or managed by Self-Service. Nutanix provides a free trial period of 60 days to try out Self-Service.

The Prism web console and Nutanix Support portal provide the most current information about your licenses. For more information on licenses, see the [Nutanix License Manager Guide](#).

Self-Service Upgrades

Upgrade Self-Service or Epsilon using the Life Cycle Manager (LCM) from Prism Central. Epsilon is the orchestration engine for Self-Service. For more information, see [Life Cycle Manager](#) on page 424.

Note: Before you upgrade Self-Service (through LCM or Prism Central upgrade), it is recommended that you do not leave any executions (blueprint launch, runbook execute, or action run) in the pending approval state.

Life Cycle Manager

The Life Cycle Manager (LCM) allows you to track and upgrade the Self-Service and Epsilon versions in Prism Central.

Note: LCM 2.1 and above support Self-Service and Epsilon upgrades.

Performing Inventory with the Life Cycle Manager

Use LCM to display the software and firmware versions of the entities in the cluster.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Admin Center** in the Application Switcher.
3. Click **LCM** in the navigation bar.
4. On the **Inventory** tab, click **Perform Inventory**.
The LCM shows a warning message if you have not enabled the auto-update, and a new version of the LCM framework is available.
5. Click **Proceed**.
The Perform Inventory process can take several minutes depending on your cluster size. Once completed, you can view the available updates on the **Updates** tab.

Upgrading Self-Service with the Life Cycle Manager

Use LCM to upgrade Self-Service and Epsilon to the latest available versions.

Before you begin

- Configure rules in your external firewall to allow LCM updates. For more information, see [Firewall Requirements](#) in the *Security Guide*.

- Run a successful inventory operation before upgrading Calm or Epsilon.

Procedure

1. Log in to your Prism Central instance as an administrator.
2. Select **Admin Center** in the Application Switcher.
3. Click **LCM** in the navigation bar.
4. Click **Edit** and select **Nutanix Calm**.
By default, Epsilon is selected.

Note: Do not select only Epsilon to update.

5. Click **Change**.
6. Select the checkbox next to the version that you want to upgrade.
7. Click **Save**.
8. You can also update the services from the **Options** dropdown menu.
 - » To perform all available updates, select **Update All**.
 - » To perform only required updates, select **Update Required**.
 - » To perform only updates you have selected, select **Update Selected**.

If you do not select any specific updates, the LCM performs all available updates.

Upgrading Self-Service at a Dark Site

By default, LCM automatically fetches updates from a pre-configured URL. If LCM fails to access the configured URL to fetch updates, you can configure the LCM to fetch updates locally to upgrade Self-Service and Epsilon.

About this task

Perform the following procedure to upgrade Self-Service and Epsilon at a dark site.

Note:

- When you upgrade Self-Service to the latest version as part of the Prism Central upgrade and if Policy Engine is enabled, then ensure to upgrade your policy engine as well.
- When you upgrade the policy engine at a dark site from a version earlier than 3.8 to version 3.8 or later, ensure that the policy engine file is uploaded as an image. For more information, see the [Enabling Policy Engine at a Dark Site](#) topic in the *Prism Central Admin Center Guide*.

Before you begin

Ensure that LCM version is 2.3 or above.

Procedure

1. Set up a local web server that is reachable by all your Nutanix clusters.

For more information on setting up a local web server, see the [Setting Up a Local Web Server](#) section in the *Life Cycle Manager Dark Site Guide*.

Note:

- You must use this server to host the LCM repository.
- From Calm 3.0.0.2 release onwards, when you set up a Windows local web server for LCM dark site upgrade, create a MIME type called '.xz' with the type set as text/plain.

2. From a device that has public Internet access, go to the Nutanix portal.
3. Click **Downloads > Self-Service (formerly Calm)**.
4. Select the required version and download Nucalm-X.X.X.X.ZIP, Epsilon-X.X.X.X.Zip, and Policy-X.X.X.X.ZIP tar files.
X.X.X.X represents the Self-Service, Epsilon, or Policy engine versions.
5. Transfer Nucalm-X.X.X.X.ZIP, Epsilon-X.X.X.X.Zip, and Policy-X.X.X.X.ZIP tar files to your local web server.
6. Extract the files into the local release directory.
You get the following files in the release directory.
 - Nucalm-X.X.X.X.Zip
 - release/builds/calm-builds/x.x.x.x/metadata.json
 - release/builds/calm-builds/x.x.x.x/metadata.sign
 - release/builds/calm-builds/x.x.x.x/nucalm.tar.xz
 - Epsilon-X.X.X.X.Zip
 - release/builds/epsilon-builds/x.x.x.x/metadata.json
 - release/builds/epsilon-builds/x.x.x.x/metadata.sign
 - release/builds/epsilon-builds/x.x.x.x/epsilon.tar.xz
 - Policy-X.X.X.X.ZIP
 - release/builds/policy_engine-builds/x.x.x.x/metadata.json
 - release/builds/policy_engine-builds/x.x.x.x/metadata.sign
 - release/builds/policy_engine-builds/x.x.x.x/policy-engine.tar.gz
7. From a device that has public Internet access, go to the Nutanix portal.
8. Select **Downloads > Self-Service (formerly Calm)**.
9. Download nutanix_compatibility.tgz and nutanix_compatibility.tgz.sign tar files.
10. Transfer the compatibility tar files to your local web server and replace the files in the /release directory.
11. Log in to your Prism Central instance.

12. Select **Admin Center** in the Application Switcher.
13. Click **LCM** in the navigation bar.
14. On the LCM page, click the **Settings** tab.
15. In the **Source** field, select **Dark Site (Local Web Server)**.
16. In the **URL** field, enter the path to the directory where you extracted the tar file on your local server .
Use the format `http://webserver_IP_address/release`.
17. Click **Save**.
18. On the **Inventory** tab, click **Perform Inventory**.
19. Update the LCM framework before trying to update any other component.
The LCM now shows the LCM framework with the updated version.

Self-Service VM Upgrades

Refer to this section to upgrade Self-Service to the latest available version after you deploy the Self-Service VM.

Note: The LCM framework for Self-Service VM does not support upgrade of Calm, Epsilon, and Policy when you perform an inventory.

Use the following upgrade path while upgrading Self-Service VM.

Table 62: Self-Service VM Upgrade Path

Starting Version	End Version	Upgrade Path
Version 3.5.2	Self-Service VM 3.8.1	Calm VM 3.5.2 (pc.2022.6) to Calm VM 3.6.2 (pc.2023.1.0.1) to Self-Service VM 3.7.2.1 (pc.2023.4) to Self-Service VM 3.8.1 (pc.2024.2)
Version 3.6	Self-Service VM 3.8.1	Calm VM 3.6 (pc.2022.6.0.1) to Calm VM 3.6.2 (pc.2023.1.0.1) to Self-Service VM 3.7.2.1 (pc.2023.4) to Self-Service VM 3.8.1 (pc.2024.2)
Version 3.6.2 or Version 3.7	Self-Service VM 3.8.1	Calm VM 3.6.2 (pc.2023.1.0.1) or Self-Service VM 3.7 (pc.2023.1.0.2) to Self-Service VM 3.7.2.1 (pc.2023.4) to Self-Service VM 3.8.1 (pc.2024.2)
Version 3.7.1, Version 3.7.2, Version 3.7.2.1, or Version 3.7.2.2	Self-Service VM 3.8.1	Self-Service VM 3.7.1 (pc.2023.3), Self-Service VM 3.7.2 (pc.2023.4), Self-Service VM 3.7.2.1 (pc.2023.4), or Self-Service VM 3.7.2.2 (pc.2023.4.0.2) to Self-Service VM 3.8.1 (pc.2024.2)

Starting Version	End Version	Upgrade Path
Version 3.8	Self-Service VM 3.8.1	Self-Service VM 3.8 (pc.2024.1) to Self-Service VM 3.8.1 (pc.2024.2)

Note: Upgrading from version 3.7.2.2 to Self-Service VM 3.8 is not supported.

To view the upgrade path for any Self-Service VM version, go to [Upgrade Paths](#) on the Nutanix Portal page and select the relevant fields to filter information.

Upgrading Calm VM from Version 3.5.2 or 3.6 to Version 3.6.2

Perform the following steps to upgrade Calm VM 3.5.2 (pc.2022.6) or Calm VM 3.6 (pc.2022.6.0.1) to Calm VM 3.6.2 (pc.2023.1.0.1).

Before you begin

Ensure that you have enabled MSP on Calm VM before you upgrade. For more information, see [Enabling Microservices Infrastructure on Self-Service VM](#) on page 30.

Procedure

1. Upgrade the Prism Central of your Calm VM to version pc.2023.1.0.1.

Note: In this step, you upgrade the Prism Central of Calm VM and not the host Prism Central. Before you upgrade, see the [Prism Central Upgrade Requirements](#) and [Prism Central Upgrade Limitations](#) topics in the *Prism Central Infrastructure Guide*.

For additional information on Prism Central upgrade, see the [Prism Central Installation or Upgrade](#) topic in the *Prism Central Infrastructure Guide*.

- a. Go to the [Nutanix Portal Downloads](#) page.
- b. Click **Download** and **Metadata** for **Prism Central Upgrade (Version: pc.2023.1.0.1)** to save the binary (.tar.gz) and metadata (.json) files to your local media.
- c. Log in to your Calm VM.
- d. Click **Prism Central Settings** and then click **Upgrade Prism Central**.
- e. On the Upgrade Software page, click **upload the Prism Central binary**.
- f. Browse to the location to select the Prism Central Metadata and Binary files, and then click **Upload Now**.
- g. Once the upload is complete, click **Upgrade** and then click **Upgrade Now**.
- h. Once the Prism Central upgrade is complete, SSH to the Calm VM and run the `docker ps` command.
- i. Wait until the Nucalm, Epsilon, and Domain Manager containers are in the healthy state.
- j. To verify if the Prism services are up, run the `cluster status` command.

Note: Upgrading takes approximately 1.5 hrs for a single-node setup and approximately 3 hrs for a scale-out setup.

2. Run the following commands to create a Nutanix Calm zookeeper node.

```
nutanix@pcvm$ cd /home/nutanix/bin  
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/  
create_nucalm_zk_node.py  
nutanix@pcvm$ python create_nucalm_zk_node.py
```

3. Provide appropriate values for the large or small configuration of your Calm VM.

- For Small node, increase the existing vCPU count by 2.
- For Large node, increase the existing vCPU count by 4.

The CPU and Memory requirements of the Calm VM Deployment is equivalent to the Calm single-node profile. For the benchmark values, see [Self-Service Benchmarks](#) on page 10.

4. If your Calm VM configuration is Large, use the following commands to increase the IDF resident limit to 21 GB.

```
nutanix@pcvm$ allssh genesis stop insights_server  
nutanix@pcvm$ allssh echo --insights_rss_share_mb=21000 >> ~/config/  
insights_server.gflags  
nutanix@pcvm$ allssh echo --insights_multicluster_cgroup_limit_factor_pct=500 >> ~/  
config/genesis.gflags  
nutanix@pcvm$ allssh genesis restart  
nutanix@pcvm$ cluster start
```

5. Run the following commands on any one of the nodes to reseed Calm app in the Marketplace.

```
nutanix@pcvm$ cd /home/nutanix/bin  
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/reseed_calm_mpi.pyc  
nutanix@pcvm$ python reseed_calm_mpi.pyc
```

Note: After running this command, wait for a few minutes for the App Discovery to run and Calm app to be reseeded. You can track the progress of the operations on the Task List page.

6. Run the following commands on any one of the nodes of Calm VM to address any migration issues.

```
nutanix@pcvm$ cd /home/nutanix/bin  
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/  
run_iam_bootstrap_migration.sh  
nutanix@pcvm$ sh run_iam_bootstrap_migration.sh
```

Wait for completion and then run the following command to check the status (to be run on the leader node if you have a scale-out setup).

```
nutanix@pcvm$ sudo kubectl -n ntnx-base get pods | grep iam-bootstrap  
iam-bootstrap-njbr7 0/1 Completed 0 3d5h
```

These commands might take approximately 1 hour or more depending on the number of users (ACPs).

Upgrading Calm VM from Version 3.5.2 or 3.6 to Version 3.6.2 at a Dark Site

Perform the following steps to upgrade Calm VM 3.5.2 (pc.2022.6) or Calm VM 3.6 (pc.2022.6.0.1) to Calm VM 3.6.2 (pc.2023.1.0.1) at a dark site.

Before you begin

Ensure that you have enabled MSP on Calm VM before you upgrade. For more information, see [Enabling Microservices Infrastructure on Self-Service VM](#) on page 30.

Procedure

1. Upgrade your Prism Central to version pc.2023.1.0.1.

Note: Before you upgrade, see the [Prerequisites](#) and [Prism Central Upgrade Limitations](#) topics in the *Acropolis Upgrade Guide*.

For additional information on Prism Central upgrade, see the [Prism Central Installation or Upgrade](#) topic in the *Prism Central Infrastructure Guide*.

- a. Go to the [Nutanix Portal Downloads](#) page.
- b. Click **Download** and **Metadata** for **Prism Central Upgrade (Version: pc.2023.1.0.1)** to save the binary (.tar.gz) and metadata (.json) files to your local media.
- c. Log in to your Calm VM.
- d. Click **Prism Central Settings** and then click **Upgrade Prism Central**.
- e. On the Upgrade Software page, click **upload the Prism Central binary**.
- f. Browse to the location to select the Prism Central Metadata and Binary files, and then click **Upload Now**.
- g. Once the upload is complete, click **Upgrade** and then click **Upgrade Now**.
- h. Once the Prism Central upgrade is complete, SSH to the Calm VM and run the `docker ps` command.
- i. Wait until the Nucalm, Epsilon, and Domain Manager containers are in the healthy state.
- j. To verify if the Prism services are up, run the `cluster status` command.

Note: Upgrading takes approximately 1.5 hrs for a single-node setup and approximately 3 hrs for a scale-out setup.

2. Do the following to create a Nucalm zookeeper node.

- a. Download the `create_nucalm_zk_node.py` file from the [Nutanix Downloads](#) page to a local server that the dark site can access.
- b. Upload the downloaded `create_nucalm_zk_node.py` file to the `/home/nutanix/bin` directory of the Calm VM or create a file at the location with the same name and copy the content of the `create_nucalm_zk_node.py` file.
- c. Run the following command to create the Nucalm zookeeper node:

```
nutanix@pcvm$ python create_nucalm_zk_node.py
```

3. Provide appropriate values for the large or small configuration of your Calm VM.

- For Small node, increase the existing vCPU count by 2.
- For Large node, increase the existing vCPU count by 4.

The CPU and Memory requirements of the Calm VM Deployment is equivalent to the Calm single-node profile. For the benchmark values, see [Self-Service Benchmarks](#) on page 10.

4. If your Calm VM configuration is Large, use the following commands to increase the IDF resident limit to 21 GB.

```
nutanix@pcvm$ allssh genesis stop insights_server
```

```
nutanix@pcvm$ allssh echo --insights_rss_share_mb=21000 >> ~/config/  
insights_server.gflags
```

```
nutanix@pcvm$ allssh echo --insights_multiclustercgroup_limit_factor_pct=500 >> ~/  
config/genesis.gflags
```

```
nutanix@pcvm$ allssh genesis restart
```

```
nutanix@pcvm$ cluster start
```

5. Download the delete_ntnx_app_calmvm_362.py file from the [Nutanix Downloads](#) page to a local server that the dark site can access.
6. Upload the downloaded delete_ntnx_app_calmvm_362.py file to the /home/nutanix directory of the Calm VM or create a file at the location with the same name and copy the content of the delete_ntnx_app_calmvm_362.py file.
7. Download the mp_calm_3.6.2.json file from the [Nutanix Downloads](#) page to a local server that the dark site can access.
8. Upload the downloaded mp_calm_3.6.2.json file to the /home/nutanix directory of the Calm VM or create a file at the location with the same name and copy the content of the mp_calm_3.6.2.json file.
9. Do the following to reseed Calm app in the Marketplace.
 - a. Download the darksite_reseed_calm_mpi.py file from the [Nutanix Downloads](#) page to a local server that the dark site can access.
 - b. Upload the downloaded darksite_reseed_calm_mpi.py file to the /home/nutanix/bin directory of the Calm VM or create a file at the location with the same name and copy the content of the darksite_reseed_calm_mpi.py file.
 - c. Run the following command on any one of the nodes of Calm VM to reseed Calm app in the Marketplace.

```
nutanix@pcvm$ python darksite_reseed_calm_mpi.py
```

Note: After running this command, wait for a few minutes for the App Discovery to run and Calm app to be reseeded. You can track the progress of the operations on the Task List page.

10. Do the following to address any migration issues.
 - a. Download the run_iam_bootstrap_migration.sh file from the [Nutanix Downloads](#) page to a local server that the dark site can access.
 - b. Upload the downloaded run_iam_bootstrap_migration.sh file to the /home/nutanix/bin directory of the Calm VM or create a file at the location with the same name and copy the content of the run_iam_bootstrap_migration.sh file.
 - c. Run the following command on any one of the nodes of Calm VM to address any migration issues.

```
nutanix@pcvm$ sh run_iam_bootstrap_migration.sh
```

Wait for completion and then run the following command to check the status (to be run on the leader node if you have a scale-out setup).

```
nutanix@pcvm$ sudo kubectl -n ntnx-base get pods | grep iam-bootstrap
```

These commands might take approximately 1 hour or more depending on the number of users (ACPs).

Upgrading Self-Service VM from Version 3.6.2 to Version 3.7.2.1

About this task

Perform the following steps to upgrade Self-Service VM (formerly Calm VM) from version 3.6.2 (pc.2023.1.0.1) to version 3.7.2.1 (pc.2023.4).

Note: If you have a version earlier than 3.6.2 (version 3.5.2 or 3.6), see [Upgrading Calm VM from Version 3.5.2 or 3.6 to Version 3.6.2](#) on page 428 to upgrade your Self-Service VM.

Procedure

1. Upgrade the Prism Central of your Self-Service VM to version pc.2023.4.

Note: In this step, you upgrade the Prism Central of Self-Service VM and not the host Prism Central. Before you upgrade, see the [Prism Central Upgrade Requirements](#) and [Prism Central Upgrade Limitations](#) topics in the *Prism Central Infrastructure Guide*. Ensure that you understand all the limitations in the context of Self-Service VM.

For additional information on Prism Central upgrade, see the [Prism Central Installation or Upgrade](#) topic in the *Prism Central Infrastructure Guide*.

- a. Go to the [Nutanix Portal Downloads](#) page.
- b. Click **Download** and **Metadata** for **Prism Central Upgrade (Version: pc.2023.4)** to save the binary (.tar.gz) and metadata (.json) files to your local media.
- c. Log in to your Self-Service VM user interface.
- d. Click **Prism Central Settings**, and click **Upgrade Prism Central**.
- e. On the Upgrade Software page, click **upload the Prism Central binary**.
- f. Browse to the location to select the Prism Central metadata (.tar.gz) and binary (.json) files, and click **Upload Now**.
- g. Once the upload is complete, click **Upgrade**, and click **Upgrade Now**.
- h. Once the Prism Central upgrade is complete, SSH to the Self-Service VM and run the `docker ps` command.
- i. Wait until the Nucalm, Epsilon, and Domain Manager containers are in the healthy state.
- j. To verify if the Prism services are up, run the `cluster status` command.

Note: Upgrading takes approximately 1.5 hrs for a single-node setup and approximately 3 hrs for a scale-out setup.

2. Perform the following steps to upgrade your Self-Service VM from version 3.6.2 to version 3.7.2.1.

Note: This step is not required when you upgrade from version 3.7, 3.7.1, or 3.7.2 to version 3.7.2.1 as the zookeeper is created by default during Self-Service VM deployment.

- a. Run the following commands on any one of the nodes of your Self-Service VM to create a Nucalm zookeeper node.

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/370-Files/create_nucalm_zk_node.py
```

```
nutanix@pcvm$ python create_nucalm_zk_node.py
```

- b. Verify that the Nucalm, Epsilon, and Domain Manager containers are healthy.

3. Do the following for each node of Self-Service VM.

- a. SSH to the Self-Service VM node.
- b. Run the following command to stop the Nucalm and Epsilon containers.

```
genesis stop nucalm epsilon
```

- c. Navigate to the Calm directory (/usr/local/nutanix/nucalm/) and remove the tar file.
 - d. Wget nucalm.tar.xz from the [Nutanix Downloads](#) location.
 - e. Navigate to the Epsilon directory (/usr/local/nutanix/epsilon/) and remove the tar file.
 - f. Wget epsilon.tar.xz from the [Nutanix Downloads](#) location.
4. Run the following command to start Nucalm and Epsilon services and wait for the services to get healthy.

```
cluster start
```

5. Run the `docker ps` command to verify that the Nucalm, Epsilon, and Domain Manager containers are healthy for all nodes.

What to do next

You can upgrade from version 3.7.2.1 to version 3.8. For more information, see [Upgrading Self-Service VM from Version 3.7.1, 3.7.2, 3.7.2.1, or 3.7.2.2 to Version 3.8.1](#) on page 435.

Upgrading Self-Service VM from Version 3.7 to Version 3.7.2.1

Perform the following steps to upgrade Self-Service VM from version 3.7 (pc.2023.1.0.2) to version 3.7.2.1 (pc.2023.4).

About this task

Note: If you have a version earlier than 3.7 (version 3.6.2 or earlier), see [Upgrading Self-Service VM from Version 3.6.2 to Version 3.7.2.1](#) on page 432 to upgrade your Self-Service VM.

Procedure

1. Upgrade the Prism Central of your Self-Service VM to version pc.2023.4.

Note: In this step, you upgrade the Prism Central of Self-Service VM and not the host Prism Central. Before you upgrade, see the [Prism Central Upgrade Requirements](#) and [Prism Central Upgrade Limitations](#) topics in the *Prism Central Infrastructure Guide*. Ensure that you understand all the limitations in the context of Self-Service VM.

For additional information on Prism Central upgrade, see the [Prism Central Installation or Upgrade](#) topic in the *Prism Central Infrastructure Guide*.

- a. Go to the [Nutanix Portal Downloads](#) page.
- b. Click **Download** and **Metadata** for **Prism Central Upgrade (Version: pc.2023.4)** to save the binary (.tar.gz) and metadata (.json) files to your local media.
- c. Log in to your Self-Service VM user interface.
- d. Click **Prism Central Settings**, and click **Upgrade Prism Central**.
- e. On the Upgrade Software page, click **upload the Prism Central binary**.
- f. Browse to the location to select the Prism Central metadata (.tar.gz) and binary (.json) files, and click **Upload Now**.
- g. Once the upload is complete, click **Upgrade**, and click **Upgrade Now**.
- h. Once the Prism Central upgrade is complete, SSH to the Self-Service VM and run the `docker ps` command.
- i. Wait until the Nucalm, Epsilon, and Domain Manager containers are in the healthy state.
- j. To verify if the Prism services are up, run the `cluster status` command.

Note: Upgrading takes approximately 1.5 hrs for a single-node setup and approximately 3 hrs for a scale-out setup.

2. Run the `docker ps` command to verify that the Nucalm, Epsilon, and Domain Manager containers are healthy for all nodes.
3. Do the following for each node of Self-Service VM.

- a. SSH to the Self-Service VM node.
- b. Run the following command to stop the Nucalm and Epsilon containers.

```
genesis stop nucalm epsilon
```

- c. Navigate to the Calm directory (/usr/local/nutanix/nucalm/) and remove the tar file.
 - d. Wget nucalm.tar.xz from the [Nutanix Downloads](#) location.
 - e. Navigate to the Epsilon directory (/usr/local/nutanix/epsilon/) and remove the tar file.
 - f. Wget epsilon.tar.xz from the [Nutanix Downloads](#) location.
4. Run the following command to start Nucalm and Epsilon services and wait for the services to get healthy.

```
cluster start
```

5. Run the `docker ps` command to verify that the Nucalm, Epsilon, and Domain Manager containers are healthy for all nodes.

What to do next

You can upgrade from version 3.7.2.1 to version 3.8. For more information, see [Upgrading Self-Service VM from Version 3.7.1, 3.7.2, 3.7.2.1, or 3.7.2.2 to Version 3.8.1](#) on page 435.

Upgrading Self-Service VM from Version 3.7.1, 3.7.2, 3.7.2.1, or 3.7.2.2 to Version 3.8.1

Perform the following steps to upgrade Self-Service VM from version 3.7.1 (pc.2023.3), version 3.7.2 (pc.2023.4), version 3.7.2.1 (pc.2023.4), or version 3.7.2.2 (pc.2023.4.0.2) to version 3.8.1 (pc.2024.2).

About this task

Note:

- If you have version 3.7, you can upgrade to version 3.7.2.1 and then to version 3.8.1. For more information, see [Upgrading Self-Service VM from Version 3.7 to Version 3.7.2.1](#) on page 433.
- If you have version 3.6.2 or earlier, see [Upgrading Self-Service VM from Version 3.6.2 to Version 3.7.2.1](#) on page 432 to upgrade your Self-Service VM.

Procedure

1. Upgrade the Prism Central of your Self-Service VM to version pc.2024.2.

Note: In this step, you upgrade the Prism Central of Self-Service VM and not the host Prism Central. see the [Prism Central Upgrade Requirements](#) and [Prism Central Upgrade Limitations](#) topics in the

Prism Central Infrastructure Guide. Ensure that you understand all the limitations in the context of Self-Service VM.

For additional information on Prism Central upgrade, see the [Prism Central Installation or Upgrade](#) topic in the *Prism Central Infrastructure Guide*.

- a. Go to the [Nutanix Portal Downloads](#) page.
- b. Click **Download** and **Metadata** for **Prism Central Upgrade (Version: pc.2024.2)** to save the binary (.tar) and metadata (.json) files to your local media.
- c. Log in to your Self-Service VM user interface.
- d. Click **Prism Central Settings**, and click **Upgrade Prism Central**.
- e. On the Upgrade Software page, click **upload the Prism Central binary**.
- f. Browse to the location to select the Prism Central metadata (.tar) and binary (.json) files, and click **Upload Now**.
- g. Once the upload is complete, click **Upgrade**, and click **Upgrade Now**.
- h. Once the Prism Central upgrade is complete, SSH to the Self-Service VM and run the `docker ps` command.
- i. Wait until the Nucalm, Epsilon, and Domain Manager containers are in the healthy state.
- j. To verify if the Prism services are up, run the `cluster status` command.
- k. Run the `docker ps` command to verify that the Nucalm, Epsilon, and Domain Manager containers are healthy for all nodes.

Note: Upgrading takes approximately 1.5 hrs for a single-node setup and approximately 3 hrs for a scale-out setup.

2. Download the `create_nucalm_zk_node.py` script from the [Nutanix Downloads](#) page into the `/home/nutanix/` directory of the Self-Service VM.

For brownfield deployments of Self-Service VM, the zookeeper node does not get created automatically. To create the node, run the following commands:

The following commands do not work at the dark site. Therefore for the dark site, create the `create_nucalm_zk_node.py` file in the `/home/nutanix/` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/nutanix/
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
create_nucalm_zk_node.py
```

3. Run the following commands to download the `ssvm_380_postupgrade.py` post upgrade script file from the [Nutanix Downloads](#) page into the `/home/nutanix/bin` directory of the Self-Service VM.

The following commands do not work at the dark site. Therefore for the dark site, create the `ssvm_380_postupgrade.py` file in the `/home/nutanix/bin` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
ssvm_380_postupgrade.py
```

4. Run the following commands:

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ python ssvm_380_postupgrade.py <self-servicevm_username> <self-servicevm_password>
```

Where `<self-servicevm_username>` is the username of the Self-Service VM and `<self-servicevm_password>` is the password of the Self-Service VM.

5. After upgrading to Self-Service VM 3.8.1, if you experience multiple restarts in the Redis, PostgreSQL and CAPE pods, you can increase the ReadinessProbe timeouts to reduce the restarts. To do that:

- a. Run the following commands to download the `increase_readiness_probe_timeouts.sh` script from the [Nutanix Downloads](#) page into the `/home/nutanix` directory of the Self-Service VM.

The following commands do not work at the dark site. Therefore for the dark site, create the `increase_readiness_probe_timeouts.sh` file in the `/home/nutanix` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/nutanix
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/increase_readiness_probe_timeouts.sh
```

- b. Run the following commands:

```
nutanix@pcvm$ chmod +x increase_readiness_probe_timeouts.sh
```

```
nutanix@pcvm$ ./increase_readiness_probe_timeouts.sh
```

Upgrading Self-Service VM from Version 3.8 to Version 3.8.1

Perform the following steps to upgrade Self-Service VM from version 3.8 (pc.2024.1) to version 3.8.1 (pc.2024.2).

About this task

Note:

- If you have version 3.6.2 or earlier, see [Upgrading Self-Service VM from Version 3.6.2 to Version 3.7.2.1](#) on page 432 to upgrade your Self-Service VM.
- If you have version 3.7, you can upgrade to version 3.7.2.1 and then to version 3.8.1. For more information, see [Upgrading Self-Service VM from Version 3.7 to Version 3.7.2.1](#) on page 433.
- If you have version 3.7.1, 3.7.2, 3.7.2.1, or 3.7.2.2, see [Upgrading Self-Service VM from Version 3.7.1, 3.7.2, 3.7.2.1, or 3.7.2.2 to Version 3.8.1](#) on page 435 to upgrade your Self-Service VM to version 3.8.1.
- An upgrade from Self-Service VM version 3.8 to version 3.8.1 does not require the policy engine upgrade.

Procedure

1. Upgrade the Prism Central of your Self-Service VM to version pc.2024.2.

Note: In this step, you upgrade the Prism Central of Self-Service VM and not the host Prism Central. see the [Prism Central Upgrade Requirements](#) and [Prism Central Upgrade Limitations](#) topics in the

Prism Central Infrastructure Guide. Ensure that you understand all the limitations in the context of Self-Service VM.

For additional information on Prism Central upgrade, see the [Prism Central Installation or Upgrade](#) topic in the *Prism Central Infrastructure Guide*.

- a. Go to the [Nutanix Portal Downloads](#) page.
- b. Click **Download** and **Metadata** for **Prism Central Upgrade (Version: pc.2024.2)** to save the binary (.tar) and metadata (.json) files to your local media.
- c. Log in to your Self-Service VM user interface.
- d. Click **Prism Central Settings**, and click **Upgrade Prism Central**.
- e. On the Upgrade Software page, click **upload the Prism Central binary**.
- f. Browse to the location to select the Prism Central metadata (.tar) and binary (.json) files, and click **Upload Now**.
- g. Once the upload is complete, click **Upgrade**, and click **Upgrade Now**.
- h. Once the Prism Central upgrade is complete, SSH to the Self-Service VM and run the `docker ps` command.
- i. Wait until the Nucalm, Epsilon, and Domain Manager containers are in the healthy state.
- j. To verify if the Prism services are up, run the `cluster status` command.
- k. Run the `docker ps` command to verify that the Nucalm, Epsilon, and Domain Manager containers are healthy for all nodes.

Note: Upgrading takes approximately 1.5 hrs for a single-node setup and approximately 3 hrs for a scale-out setup.

2. Download the `create_nucalm_zk_node.py` script from the [Nutanix Downloads](#) page into the `/home/nutanix/` directory of the Self-Service VM.

For brownfield deployments of Self-Service VM, the zookeeper node does not get created automatically. To create the node, run the following commands:

The following commands do not work at the dark site. Therefore for the dark site, create the `create_nucalm_zk_node.py` file in the `/home/nutanix/` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/nutanix/
```

```
wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
create_nucalm_zk_node.py
```

3. Run the following commands to download the `ssvm_380_postupgrade.py` post upgrade script file from the [Nutanix Downloads](#) page into the `/home/nutanix/bin` directory of the Self-Service VM.

The following commands do not work at the dark site. Therefore for the dark site, create the `ssvm_380_postupgrade.py` file in the `/home/nutanix/bin` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/  
ssvm_380_postupgrade.py
```

4. Run the following commands:

```
nutanix@pcvm$ cd /home/nutanix/bin
```

```
nutanix@pcvm$ python ssvm_380_postupgrade.py '<self-servicevm_username>' '<self-servicevm_password>'
```

Where `<self-servicevm_username>` is the username of the Self-Service VM and `<self-servicevm_password>` is the password of the Self-Service VM.

Upgrade the Policy Engine VM on an Upgraded Self-Service VM

Use the following steps to upgrade the policy engine VM on an upgraded Self-Service VM (formerly Calm VM).

Procedure

1. Backup the policy Engine database. For more information, see [Backing Up and Restoring Policy Engine Database](#) on page 399.

The recommended path for the backup is `/home/nutanix/tmp/`.

2. Reset the policy engine feature in Nutalm using the following steps:

- a. Run the following commands to download the `clear_policy_feature.py` script from the [Nutanix Downloads](#) page into the `/home/docker/nucalm/log/` directory of the Self-Service VM.

The following commands do not work at the dark site. Therefore for the dark site, create the `clear_policy_feature.py` file in the `/home/docker/nucalm/log/` directory and copy and paste the content to the file.

```
nutanix@pcvm$ cd /home/docker/nucalm/log/
```

```
nutanix@pcvm$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/clear_policy_feature.py
```

- b. Run the following commands:

```
nutanix@pcvm$ zkrm /appliance/logical/policy_engine/status
```

```
nutanix@pcvm$ zkrm /appliance/logical/policy_engine/transient_data
```

```
nutanix@pcvm$ docker exec -i nucalm bash -c source /home/calm/venv/bin/activate && python /home/calm/log/clear_policy_feature.py
```

3. Power off the current policy engine VM to free up the associated IP address.
4. Download the blueprint from one of the following locations.
 - » From the [Nutanix Downloads](#) page for single-node Self-Service VM, download the blueprint.
 - » From the [Nutanix Downloads](#) page for scale-out Self-Service VM, download the blueprint.
5. Upload the downloaded blueprint to the Self-Service VM.

6. Set values for the following variables in the blueprint.

- Policy engine IP address. This IP address is the same as that of the old policy engine VM and must be in the same network as that of the Self-Service VM.
- VIP of the Self-Service VM.
- Netmask of the Self-Service VM network.
- Gateway of the Self-Service VM network.
- DNS IP of the Self-Service VM.
- NTP IP of the Self-Service VM.
- Public key of Self-Service VM. For scale-out Self-Service VM, provide the public key of all VMs. You can run the following command to get the public keys of your VMs.

```
allssh cat .ssh/id_rsa.pub
```

- Disable check-login in case the check-login is enabled by default.

7. Launch the blueprint.

8. Wait for the app that you created by launching the blueprint to get into the running state.

9. To enable the new policy engine VM on the upgraded Self-Service VM, do the following:

- a. Use the following commands to wget the `set_policy_calvmvm.py` script from the [Nutanix Downloads](#) page into the `/home/nutanix/bin/` directory of your Self-Service VM and provide the run permission. The following commands do not work at the dark site. Therefore for the dark site, create the `set_policy_calvmvm.py` file in the `/home/nutanix/bin/` directory and copy and paste the content to the file.

```
nutanix@PCVM-CalmVM:~$ cd /home/nutanix/bin/
```

```
nutanix@PCVM-CalmVM:~$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/set_policy_calvmvm.py
```

```
nutanix@PCVM-CalmVM:~$ chmod +x set_policy_calvmvm.py
```

- b. Use the following commands to wget the `set_policy_calvmvm.sh` script from the [Nutanix Downloads](#) page into the `/home/nutanix/bin/` directory of your Self-Service VM and provide the run permission. The following commands do not work at the dark site. Therefore for the dark site, create the `set_policy_calvmvm.sh` file in the `/home/nutanix/bin/` directory and copy and paste the content to the file.

```
nutanix@PCVM-CalmVM:~$ cd /home/nutanix/bin/
```

```
nutanix@PCVM-CalmVM:~$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/set_policy_calvmvm.sh
```

```
nutanix@PCVM-CalmVM:~$ chmod +x set_policy_calvmvm.sh
```

- c. Use the following commands to wget the `generate_policy_cert.py` script from the [Nutanix Downloads](#) page into the `/home/nutanix/bin/` directory of your Self-Service VM and provide the run permission. The following commands do not work at the dark site. Therefore for the dark site, create the `generate_policy_cert.py` file in the `/home/nutanix/bin/` directory and copy and paste the content to the file.

```
nutanix@PCVM-CalmVM:~$ cd /home/nutanix/bin/
```

```
nutanix@PCVM-CalmVM:~$ wget https://download.nutanix.com/Calm/CalmVM-Files/380-Files/generate_policy_cert.py
```

```
nutanix@PCVM-CalmVM:~$ chmod +x generate_policy_cert.py
```

10. SSH into the Self-Service VM as a Nutanix user and run the following commands after making the required changes.

Note: On a scale-out Self-Service VM, run the following command on any one of the VMs.

```
nutanix@pcvm$ sh /home/nutanix/bin/set_policy_calvmvm.sh <POLICY_VM_IP>  
<POLICY_VM_UUID>
```

Where `<POLICY_VM_IP>` is the IP address of the policy engine VM and `<POLICY_VM_UUID>` is the policy engine VM unique identifier.

11. Restore the policy engine database.

After the new policy engine VM is enabled and running, copy the backup using the `scp` command from the Self-Service VM to the new policy engine VM and then perform the restore operation on the new policy engine VM. For more information, see [Backing Up and Restoring Policy Engine Database](#) on page 399.

12. Verify that the new policy engine VM works properly and then clear the old policy engine VM. To do that:
 - a. Delete the temporary files such as the backup file from the Self-Service VM.
 - b. Delete the old policy engine VM.

ADDITIONAL INFORMATION

The topics in this section help you perform additional configuration for different components of Self-Service, such as enabling microservices infrastructure (MSP), generating SSH keys, and so on.

Syslog for Calm and Epsilon Services

In Prism Central, you can configure syslog monitoring to forward system logs of the registered clusters to an external syslog server. For information on how to configure syslog monitoring in Prism Central, see [Configuring Syslog Monitoring](#) in the *Prism Central Admin Center Guide*.

You can select different log modules to be sent to the syslog servers, including the log modules of Calm and Epsilon services. For more information on log modules and their associated severity levels, see [Syslog Modules](#) in the *Prism Central Admin Center Guide*.

The following table provides information about the Self-Service specific log level and Self-Service resource log type.

Table 63: Log-Level

Log-level	Self-Service resource log type
Error	Any error that is fatal to the operation. For example, any validation errors or errors during blueprint, runbook, or application creation.
Info	Information about successful operations, such as service start, service stop, configuration settings, application actions, API call request responses, and so on.
Debug	Typically contains information that are useful during the debug phase.
Warning	Warning conditions that are slightly less severe than errors. Error conditions happen when an operation fails. Warning indicates that an operation might fail in the future if corrective actions are not taken immediately.

The following table provides information on the Self-Service specific syslog severity levels.

Table 64: Syslog Severity Levels

Severity	Description
Emergency	Self-Service does not log anything with the Emergency level.
Alert	Self-Service does not log anything with the Alert level.
Critical	Self-Service does not log anything with the Critical level.

Severity	Description
Error	Self-Service logs with the Error tags are copied.
Warning	Self-Service logs with the Error and Warning tags are copied.
Notice	Self-Service logs with the Error and Warning tags are copied.
Info	Self-Service logs with the Error, Warning, and Info tags are copied.
Debug	Self-Service logs with the Error, Warning, Info, and Debug tags are copied (primarily all logs of Self-Service). When you use the Debug level, you get all Self-Servicelogs.

Credential Security Support Provider

The Credential Security Support Provider (CredSSP) protocol is a security support provider that you implement using the Security Support Provider Interface (SSPI). CredSSP allows an app to delegate credentials of a user from the client to the target server for remote authentication. CredSSP provides an encrypted transport layer security protocol channel. The client is authenticated over the encrypted channel by using the Simple and Protected Negotiate (SPNEGO) protocol with either Microsoft Kerberos or Microsoft NTLM.

For more information, refer to the *Microsoft Documentation*.

Enabling CredSSP

Perform the following procedure to enable CredSSP.

Procedure

Run the following command to enable CredSSP on the target machine.

```
> Enable-WSManCredSSP -Role Server -Force
```

Generating SSH Key on a Linux VM

Perform the following procedure to generate an SSH key pair on a Linux VM.

About this task

Note: Avoid generating the RSA key pair on your Prism Central VM or CVM.

Procedure

1. Run the following shell command to generate an RSA key pair on your local machine.

```
$ ssh-keygen -t rsa
```

2. Accept the default file location as `~/.ssh/id_rsa`.

You can find your public key at `~/.ssh/id_rsa.pub` and the private key at `~/.ssh/id_rsa`.

Note: Do not share your private key with anyone.

Generating SSH Key on a Windows VM

Perform the following procedure to generate an SSH key pair on Windows.

About this task

Note: Avoid generating the RSA key pair on your Prism Central VM or CVM.

Procedure

1. Launch PuTTYgen.
2. Move the mouse cursor in the blank area and click **Generate**.
3. To convert the private key into an OpenSSH format, select **Conversions > Export OpenSSH key**. PuTTYgen warning message appears.
4. Click **Yes** to save the key without a passphrase.
5. Navigate to a location on your local system to save the key.
6. Type a name for the key.
7. Click **Save**.
8. Copy the public key (highlighted in the following image) into a plain text file and save the key at the same location as that of the private key.

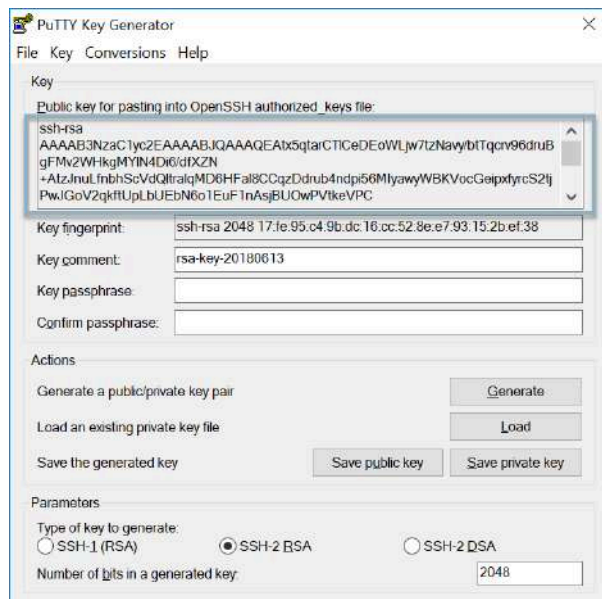


Figure 96: Public Key

Integrated Linux Based PowerShell Gateway Errors

You might encounter the following errors when you run the PowerShell scripts using the integrated Linux based PowerShell gateway.

Table 65: Integrated Linux Based PowerShell Gateway Errors

Error	Description
Access denied	<p>If the VM and the WinRM services are started but the specified credential is wrong. You encounter the error in the following cases.</p> <ul style="list-style-type: none">• When you use the local account credential of a domain member machine. You can resolve the issue by using the domain credentials.• When the script requires elevated privileges and the CredSSP is not enabled on the target machine. You can resolve the issue by enabling the CredSSP on the target machine. For more information on enabling CredSSP, see Credential Security Support Provider on page 444.
Connection refused	<p>You encounter the connection refusal error in the following cases.</p> <ul style="list-style-type: none">• When a VM is not started but Self-Service tries to do a check-login or runs a PowerShell task. You can resolve the issue by adding a delay time task. For more information, see Adding a Delay Task on page 186.• When Self-Service is not able to communicate with the target machine. You can resolve the issue by allowing the connection to the port that is used to contact the target machine. Ensure that all the firewalls between Prism Central and the target machine allow connections to the port.

Localization

Nutanix localizes the user interface in simplified Chinese and Japanese. All the static screens are translated to the selected language. You can change the language settings of the cluster from English (default) to simplified Chinese or Japanese. For more information, see [Changing the Prism Central Language Settings](#) in the *Prism Central Admin Center Guide*.

COPYRIGHT

Copyright 2025 Nutanix, Inc.

Nutanix, Inc.

1740 Technology Drive, Suite 150

San Jose, CA 95110

All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. Nutanix and the Nutanix logo are registered trademarks of Nutanix, Inc. in the United States and/or other jurisdictions. All other brand and product names mentioned herein are for identification purposes only and may be trademarks of their respective holders.