May 2026 | Guide

What It Takes to Build a Production-Ready Kubernetes Platform

Learn why DIY Kubernetes breaks down in production and what platform teams need to do differently.

NUTANIX

Table of Contents

Chapter 1: The Challenges: Why Kubernetes Isn't Enough For a Production Ready Platform

The Lifecycle of a DIY Platform

Most organizations don't set out to build a complex internal platform, they fall into it one YAML at a time. In the Day 0 honeymoon phase, assembling open-source components feels like pure productivity and the focus stays on enabling delivery. Each new problem is an interesting puzzle with a novel solution, but as the platform scales, the math changes.

What began as a simple orchestrator evolves into a fragile web of interdependent tools for networking, mesh, observability, and authentication. Every Kubernetes® update triggers a dependency domino effect. Day 2 operations are dominated by testing compatibility and managing configuration drift across the fleet becomes the team's full time job.

The cognitive load that was removed from developers hasn't disappeared, it has simply been offloaded to the platform engineer. Instead of building a "Golden Path" for their users, the team is trapped in the cycle of keeping the lights on and constantly sacrificing roadmap innovation at the altar of maintenance. This gap in production readiness isn't Kubernetes orchestration itself, rather it's the managed lifecycle of everything around it.

The Functional Gap: Beyond Orchestration and Lifecycle Maintenance

Building a complete, enterprise-grade container platform requires functionality that spans far beyond core Kubernetes orchestration. To provide a stable environment for developers, platform engineers must manually integrate and manage a diverse stack of components.

With over 1,200+ projects in the CNCF landscape, organizations face significant complexity in tool selection and integration. Every tool added to the stack, whether for GitOps, observability, or security, becomes a permanent maintenance requirement that includes:

- **Advanced Networking:** L4/L7 load balancing, DNS, and network policy enforcement
- **Ingress Gateway:** HTTP routing, SSL/TLS termination, and external access management
- **Container Registry:** Private image hosting and lifecycle management
- **Data Services:** Stateful storage, storage migration, and container-native backup/restore with replication for business continuity
- **Policy Management:** Centralized RBAC with policy-as-code enforcement across clusters
- **Security:** Secrets management, supply-chain controls, and zero-trust networking patterns
- **Observability:** metrics, logging, and tracing across clusters
- **Automation and Delivery:** GitOps workflows (Argo CD / Flux) plus CI/CD pipeline integration for build/test/deploy
- **Multi-cluster Management:** Fleet provisioning, upgrades, policy rollout, and operational control across hybrid/multi-cloud
- **FinOps/Cost Tools:** Visibility into consumption and cost optimization across teams and clusters
- **GPU Access:** Managing GPU access and scheduling for AI/ML workloads
- **Service Mesh:** Traffic management, service-to-service security (mTLS), and workload-level observability

Manually managing the lifecycle of these components is resource-intensive. A production-ready platform requires the manual integration of over 20+ disparate components. As each project releases 3–4 times per year, a platform team faces a validated burden of over 100 upgrades annually, each requiring independent compatibility and regression testing. Without a single platform that upgrades the entire stack as a validated unit, engineers remain focused on underlying infrastructure management rather than delivering architectural value.

The Support Window and API Fragility

Kubernetes ships minor releases on a fast cadence and upstream patch support for a given minor version is roughly 12 months (often treated as ~14 months end-to-end), so platform teams end up in a continuous upgrade motion. Since upstream Kubernetes doesn't ship as a pre-integrated enterprise stack, each upgrade becomes a full compatibility exercise. This includes auditing deprecated and removed APIs, updating manifests and operators, and validating that critical add-ons (CNI, CSI, ingress controllers, policy and observability components) remain compatible with the target version. This often leads to upgrade debt, where teams delay critical security patches because they cannot risk the downtime or do not have the resources to properly validate all of the interdependencies.

Fragmentation Across Environments

Many organizations operate mainly on-premises, however they are increasingly being asked to support hybrid environments that also span public clouds and edge locations. Without a unified, vendor-agnostic API, these environments become isolated operational silos. This fragmentation typically forces companies to need multiple teams to support hybrid operations, including separate engineering, on-premises, and public cloud teams since management workflows and automation scripts are not portable between providers.

The primary implication of maintaining these multiple teams is a significant increase in operational overhead and technical complexity, as organizations must duplicate efforts to manage identical workloads across different infrastructures. This lack of consistency makes it difficult to enforce a unified security posture, turning a fleet of clusters into a collection of disparate environments. Teams require a solution that standardizes how clusters are built and secured across any environment with a single operating model that can reduce the need for redundant specialized teams and ensure that production readiness is no longer dependent on where a workload runs.

Stateful Workload Resilience and Data Protection

Kubernetes was born for stateless services, with state and persistent data being pushed to external infrastructure. As organizations migrate mission-critical databases, key/value stores, or other long-term storage to the fleet, storage becomes the ultimate bottleneck with platform teams stuck with manually bridging the gap between Kubernetes and legacy storage arrays. Implementing enterprise-grade data services remains a significant hurdle in cloud native architecture. While Kubernetes handles stateless scaling effectively, protecting stateful workloads requires complex integrations for multi-protocol support, including block, file, and S3-compatible object storage. Without native, application-aware storage integration that supports metro or async replication, achieving the recovery times required for mission-critical workloads remains a primary source of Day 2 burnout for most platform teams.

Escalating Costs and Operational Debt

The initial appeal of an open-source, DIY platform is often eclipsed by the maintenance tax and manual effort required to maintain them. When a platform is built piece-by-piece, engineering hours are diverted from building developer services to a perpetual cycle of patching, testing, and troubleshooting a fragmented stack. As the fleet grows, the operational burden scales non-linearly creating the trap where the team's capacity is consumed by lifecycle management and break/fix activities. Ultimately, this operational debt becomes a bottleneck, forcing the best engineers to focus on maintaining the plumbing rather than delivering the high-value services that actually drive business results and innovation.

Chapter 2: The Strategic Shift: Internal Developer Platform and the Golden Path

Many enterprises are investigating a managed and curated platform experience to bypass the ongoing burden of maintaining fragmented DIY stacks. The objective is to move the team's focus away from low-level component integration and toward the delivery of an Internal Developer Platform (IDP). The platform engineering team creates a golden path to production as the easiest way to simplify and standardize innovation.

To be effective, the foundation of an IDP should be:

- **Open and Extensible:** Built on pure, upstream Kubernetes to ensure full compatibility with the CNCF ecosystem and avoid proprietary lock-in.
- **Consistency Across Environments:** In line with the strategic intent of Kubernetes adoption, the Kubernetes platform must be consistent and extensible to provide unified visibility across a variety of environments, including on-prem, public cloud, hybrid, and edge.
- **AI-Ready Infrastructure:** Moving beyond simple containers to handle resources like GPUs and dynamic resource allocation (DRA) as a standard platform service.
- **Developer-Centric:** Built-in automation for patching, updates, upgrade, and testing for fleets of clusters for centralized observability and to enable developers to focus on shipping application code and innovation.
- **Data Services:** Data services and management stack, including object storage for images and media, SQL/NoSQL databases for structured user data, and file storage for logs. This ensures the necessary data persistence and storage resources alongside compute.

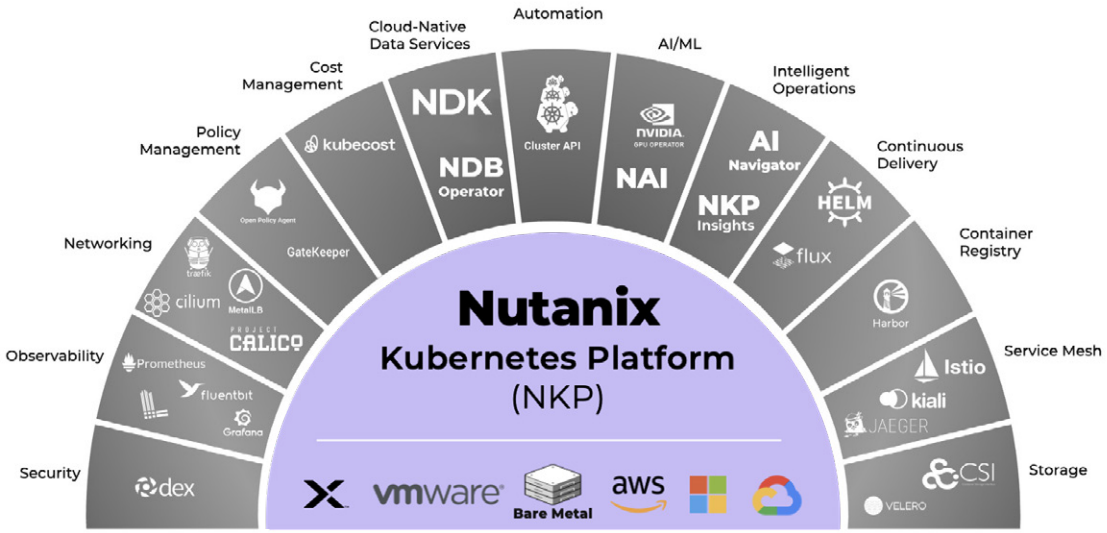
Nutanix helps platform engineering teams deploy an enterprise-ready platform built on pure upstream Kubernetes, providing a self-service environment so developers can move from code-commit to production without the manual delays of infrastructure provisioning.

Chapter 3: Nutanix Kubernetes Platform

The Nutanix Kubernetes Platform (NKP) is a complete, open, and enterprise-grade platform that brings resiliency, security, and Day 2 operations to cloud native applications. NKP is designed to remove the integration tax of DIY Kubernetes with an opinionated stack that standardizes how clusters are built, upgraded, secured, and observed, delivering fleets of clusters across on-prem, edge, and public cloud environments with one operating model.

Key Capabilities Include:

- **Enterprise-Ready Platform:** The full-stack platform provides all of the components needed to deploy and run containerized applications in production, including monitoring, security, ingress, continuous delivery, and storage.
- **Open:** Full upstream CNCF compliance that allows access to a full catalog of validated CNCF projects with freedom to integrate any alternative tools for platform needs.
- **Streamlined Lifecycle Management (LCM):** Upgrade a single platform, which automates the cluster deployment, upgrade, the OS, and the add-on components – all of which have had upgrade testing and compatibility testing.
- **Observability and Insights:** Integrated logging, monitoring, and alerting. NKP Insights and AI Navigator offer anomaly detection with root cause analysis, conversational troubleshooting interface, and recommendations to troubleshoot faster.
- **Golden Templates for Repeatable Builds:** NKP provides standardized “golden paths” and automation that provide consistent and repeatable cluster builds across any environment. This helps customers avoid the manual configuration drift that typically occurs as an organization scales its number of containers.
- **Accelerated Delivery with Native Integration:** Native CI/CD pipeline integration and GitOps-based reconciliation allow developers to move from code-commit to production without the infrastructure bottlenecks of a DIY platform.
- **Portability Across Environments:** Maintain consistent operations and run workloads anywhere.
- **Minimize Operational Cost:** By unifying the management of clusters across different environments (cloud, on-prem, edge), NKP can help reduce the overhead of managing disparate tools and redundant teams.
- **Nutanix Enterprise-Grade Support:** Reliable, ongoing support with a net promoter score of 90+.



Fleet Management

- **Declarative Fleet Orchestration:** NKP utilizes Cluster API patterns to provide a unified, declarative interface for cluster lifecycle management, leveraging Kubernetes controllers to automate the reconciliation of infrastructure to its desired state across on-premises, cloud, and edge environments.
- **Centralized fleet management:** Standardize how clusters are created, configured, upgraded, and monitored across the fleet from a single pane.
- **Consistent change rollout:** Push application and configuration updates across multiple clusters without manual intervention.
- **Integrated platform tooling:** NKP includes a validated catalog of CNCF projects and platform components like networking, observability, policy management, and security so teams spend less time stitching tools together and re-testing compatibility.
- **AI-Driven Diagnostics and Operations:** NKP Insights and AI Navigator act as automated reliability engineers, continuously scanning clusters against best practices to detect anomalies before they cause outages, while providing a conversational interface for natural language troubleshooting to reduce Mean Time to Resolution (MTTR).

Run Anywhere with One Platform

Organizations can deploy NKP across a wide range of environments. NKP works seamlessly on-prem, including on Nutanix's own Nutanix Cloud Infrastructure (NCI) platform, virtual machines, bare-metal servers, public clouds, edge locations, and even air-gapped sites. Running NKP on Nutanix infrastructure provides unique integrations that streamline deployment, accelerate operations, and unlock access to a comprehensive suite of data services. Additionally, the Nutanix platform's distributed database architecture offers an added layer of resiliency, strengthening the underlying foundation for NKP.

Security and Air-Gapped Support

NKP is designed with built-in security and capabilities that support customer compliance programs. It standardizes identity, access control, policy enforcement, network segmentation, and secure upgrade practices, including support for restricted and air-gapped environments.

- **Centralized Access**
  - **Authentication:** Support SSO and federated authentication patterns so identity and access remain consistent across clusters.
  - **RBAC and encryption:** Use Kubernetes-native RBAC and encryption to meet enterprise security requirements and reduce cluster-by-cluster access models.
  - **Compliance Support:** Provides capabilities that can help customers align with requirements such as FIPS 140-2 where applicable.
- **Policy enforcement and network controls**
  - **Policy as Code (OPA):** Use policy enforcement (OPA Gatekeeper) to apply standards consistently like admission controls and baseline security rules without slowing delivery.
  - **Network control:** Support cloud native networking options (Cilium/Calico) and Kubernetes Network Policies for pod/service-level traffic control.
  - **Service mesh for mTLS:** Support service mesh capabilities to enable mTLS for service-to-service security when required.
- **Secure lifecycle and validated operations**
  - **Lifecycle alignment:** Deploy and maintain core platform security components as validated, version-aligned platform applications to reduce version-skew and upgrade risk.

- **Air-gapped / restricted environment operations**
  - **Offline packaging and distribution:** Support offline bundle approaches to reduce manual effort moving images/charts into restricted networks.
  - **Local registry option:** Enable a local registry model so images can be stored and managed inside the environment.
  - **Proxy and offline flexibility:** Work with existing proxies or local registries so air-gapped setups don't require custom one-offs.
  - **Military-Grade DevSecOps:**
    - Pre-scanned releases and installation bundles for secure, validated deployments.
    - SSO and centralized RBAC for unified access control.
    - Multi-tenancy support for secure workload isolation.
    - VPC integration and network policies for fine-grained traffic control.
    - Encrypted communications to protect data in transit.
    - Policy enforcement and anomaly detection to support proactive security practices and help customers meet compliance requirements.

Cloud Native Technical Demo Series

Dive into Kubernetes deployment and management in this playlist, featuring topics like automation, security, and streamlined day 2 operations.

Nutanix Kubernetes Platform (NKP)

Discover how NKP simplifies platform engineering and provides consistency across any environment.

Ready to Learn More?

Explore the Cloud Native Tech Resource Center for technical blogs, how-to videos, and validated designs.

Chapter 4: Enterprise-Grade Data Services and Management for Stateful Workloads

Nutanix Data Services for Kubernetes (NDK) extends enterprise storage to Kubernetes with application-aware replication and disaster recovery for stateful Kubernetes workloads so platform teams can protect data and recover applications without integrating a separate storage or data services solution. Developers can now define application-level snapshot and replication schedules as part of their deployment pipeline.

What this delivers for platform teams:

- **Enterprise Storage for Kubernetes:** Nutanix Unified Storage (NUS) delivers integrated persistent storage for all data including support for block, file, object, and database. Its scale-out architecture aligns with how Kubernetes apps are built and run, ensuring simplified scaling across clusters
- **Database-as-a-Service:** Provides Database-as-a-Service (DBaaS) support for fast, repeatable deployments of SQL, NoSQL, and vector databases. By using the NDB Kubernetes Operator, developers can use “infrastructure as code” to automate the provisioning, patching, and backup of databases directly from within NKP—replicating the ease of public cloud database services in the data center or at the edge.
- **NDK is available through the NKP app catalog:** Platform teams can enable NDK directly through NKP, making it straightforward to adopt application-aware replication and DR as part of the Kubernetes platform standard.
- **Application- and namespace-level protection:** Policy-based backup, snapshots, replication and DR orchestration helps reduce RPO/RTO risk
- **Multi-site governance support:** Enable geographically distributed data protection with consistent policies across clusters to support compliance and governance needs.

Nutanix Data Services for Kubernetes (NDK)

Learn how Nutanix ensures application resilience and simplifies cloud-native operations.

Chapter 5: Operational Outcomes for Platform Teams

Streamlined Lifecycle Management

NKP automates Kubernetes deployments, scaling, and upgrades for various infrastructure providers. Instead of treating every Kubernetes upgrade as a custom integration exercise, platform teams can take advantage of a platform with known compatibility across core platform capabilities. This can limit version skew across clusters, simplify coordination between hardware refresh cycles, minimize upgrade debt, and makes it easier to keep fleets current on patches without introducing unnecessary operational risk.

Operational Consistency Across the Fleet

NKP enables a consistent operating model across clusters and environments by standardizing how Kubernetes is deployed, configured, and governed. Whether clusters run on-prem, in public cloud, or at the edge, platform teams can apply the same lifecycle workflows, security controls, and policy boundaries across the fleet. This consistency can reduce drift between clusters and ensure production readiness is not dependent on where a workload runs or how a cluster was originally created.

Faster Software Delivery

NKP is designed to deliver fast time-to-value with blueprinted clusters and golden images, significantly reducing deployment time for repeatable deployments. Instead of assembling and continuously re-validating dozens of independent open source components, teams operate against a complete, integrated platform layer with a consistent lifecycle. By reducing manual dependency tracking, version compatibility testing, and environment-specific scripting, NKP frees platform capacity to focus on higher-value work. NKP can help decrease time-to-market with a unified self-service experience, allowing developers to utilize critical tools and consume data services easily and without delay.

Ready to Learn More?

Explore the Cloud Native Tech Resource Center for technical blogs, how-to videos, and validated designs.