portworx
by Everpure™

The Migration Journey

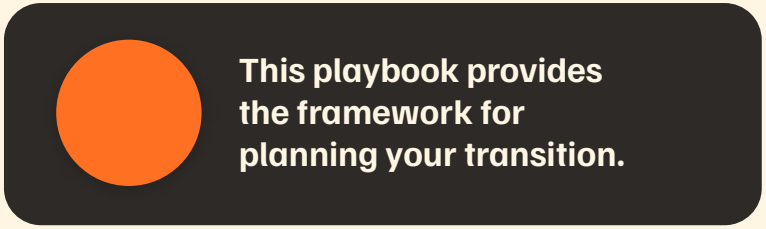
Your VM Migration Playbook

You understand the Kubernetes stack. The previous eight chapters mapped VMware concepts to their Kubernetes equivalents across compute, storage, networking, security, and Day 2 operations. You know how KubeVirt bridges the gap between VMs and containers. You have the technical vocabulary.

Now comes the harder question.

How do you move a production VMware estate to Kubernetes without disrupting the business?

Migration is a project management challenge as much as a technical one. Success depends on accurate assessment, workload categorization, team readiness, and phased execution. Organizations attempting to move everything at once fail. Organizations refusing to start at all fall behind. The right approach sits between those extremes.



Assess Your VMware Estate

Every migration begins with inventory. You need a complete picture of your current environment before making any decisions about the target state.

Start with vCenter. Export your VM inventory, including resource allocation, storage consumption, network dependencies, and performance baselines. VMware Aria Operations (formerly vRealize Operations) provides historical utilization data for CPU, memory, storage IOPS, and network throughput. These baselines become your success criteria after migration.

Document application dependencies. A single VM rarely operates in isolation. Web servers connect to application servers. Application servers connect to databases. Databases replicate to standby instances. Map these relationships explicitly. Tools like VMware Aria Operations for Networks (formerly vRealize Network Insight) reveal east-west traffic patterns between VMs. These dependency maps determine which workloads migrate together and which require special handling.

Catalog every VM by business function, owner, SLA requirement, and compliance classification. A development workstation and a production database server belong in different migration waves. The assessment should answer four questions for each workload. What does the application do? Who owns the application? What are the uptime requirements? What are the data protection requirements?

Organizations commonly find that a significant portion of their VMs serve no active purpose during this assessment. Decommissioning unused VMs before migration immediately reduces scope and cost.

Categorize Your Workloads

The industry-standard framework for migration categorization uses the "6 Rs" model. AWS developed this framework by building on an earlier Gartner migration model, adding strategies like Retain and Retire to the original set. For a VMware-to-Kubernetes transition, four of these Rs apply directly.

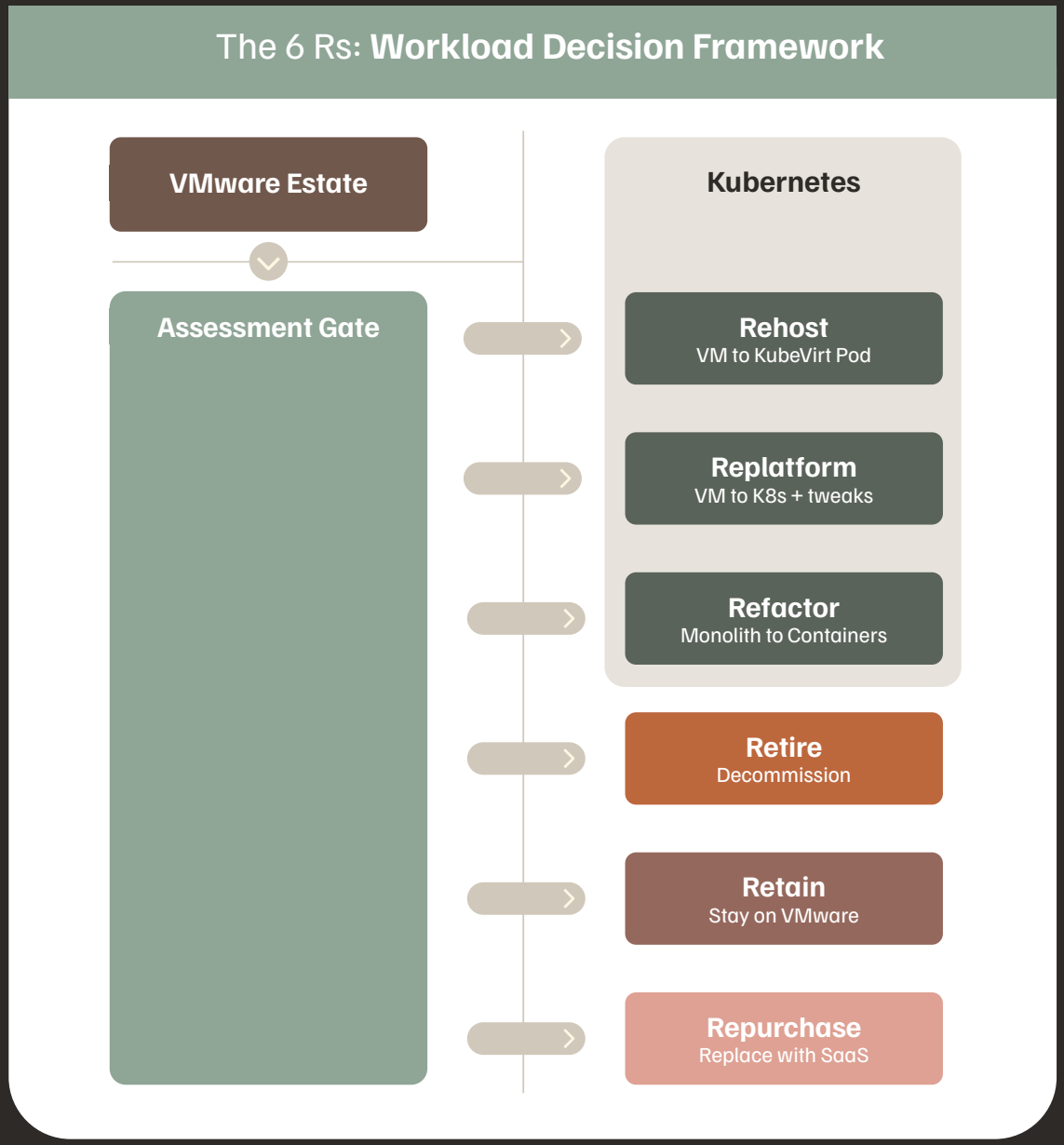


Figure 1: Workload categorization paths from VMware to Kubernetes using the 6 Rs framework.

Categorize Your Workloads

Rehost (Lift and Shift)

Move VMs to Kubernetes using KubeVirt with minimal changes. The VM runs inside a Kubernetes pod, managed by the KubeVirt operator. Operating systems, applications, and configurations remain unchanged. This approach works for legacy applications where source code is unavailable or where refactoring costs exceed the benefit. Portworx provides the storage layer for KubeVirt workloads, delivering Storage vMotion-equivalent capabilities through its Enhanced Storage Migration feature and automated storage pool rebalancing via Autopilot policies. These features preserve familiar operational workflows on the new platform.

Replatform (Lift and Reshape)

Make targeted modifications during migration to take advantage of Kubernetes-native features without rewriting the application. Examples include replacing a VM-based load balancer with a Kubernetes Service, moving from local storage to persistent volumes backed by Portworx, or switching from cron-based scheduling to Kubernetes CronJobs. The core application architecture stays intact.

Refactor (Re-architect)

Redesign the application to run as containers. Break monolithic applications into microservices. Replace stateful session management with external state stores. Adopt 12-factor application principles. This approach delivers the greatest long-term benefit but requires the highest investment in development time and testing.

Retire

Decommission applications no longer needed. The assessment phase often reveals redundant services, abandoned projects, and legacy systems with no active users. Retiring these workloads before migration reduces complexity and licensing costs.

Two additional categories, Retain and Repurchase, complete the framework. Retain keeps certain workloads on the existing infrastructure when migration risk outweighs the benefit. Repurchase replaces on-premises applications with SaaS equivalents.

Most VMware estates follow a similar pattern. The largest share of workloads is rehost candidates. A smaller group benefits from replatforming. Only a fraction justifies full refactoring. The remainder splits between retire, retain, and repurchase. Your exact ratios depend on application age, business criticality, and available development resources.

Build Your Platform Team

Kubernetes requires a different operational model than VMware. In the VMware world, a small team of vSphere administrators manages the entire stack through vCenter. In Kubernetes, responsibilities are split across a platform team and application teams.

Platform Team Responsibility Model

Application Teams	Platform Team
Deployments	**Cluster Lifecycle**
Configurations	**Storage Classes (Portworx)**
App Monitoring	**Network Policies**
CI / CD Pipelines	**RBAC Templates**
	Observability
Teams deploy and manage their own apps within namespaces and policy guardrails	Team owns clusters, security policies, storage, networking, and monitoring infrastructure

SELF - SERVICE API BOUNDARY

Figure 2: Ownership split between platform team and application teams in the Kubernetes operating model.

The platform team owns the Kubernetes clusters, including provisioning, upgrades, security policies, storage configuration, networking, and observability.

This team builds the internal platform on top of raw Kubernetes, providing guardrails and self-service capabilities for application teams.

Start with your existing VMware administrators. Their infrastructure knowledge transfers directly. They understand resource management, high availability, disaster recovery, and storage operations. These skills apply to Kubernetes. The tools change, but the operational discipline remains the same.

A small initial platform team (often 3-5 engineers in early stages) with overlapping skills provides a starting point. At least one member should have deep Kubernetes experience. Others bring expertise in VMware operations, networking, storage, and security. Cross-training fills gaps over time. Larger organizations scale this team as migration phases progress.

Define clear ownership boundaries from the start. The platform team manages cluster lifecycle, base images, storage classes, network policies, and RBAC templates. Application teams manage their deployments, configurations, and application-level monitoring within the namespaces and policies provided by the platform team.

Platform Team Responsibility Model

Application Teams	Platform Team

Upskill Your Staff

The skills gap is the most underestimated risk in VMware-to-Kubernetes migrations. Technical professionals with years of VMware experience often resist change or doubt their ability to learn a new platform. Both reactions are addressable.

Start with fundamentals. Every team member should understand pods, deployments, services, namespaces, and persistent volumes. The Certified Kubernetes Administrator (CKA) exam provides a structured learning path and an industry-recognized credential. Pair formal training with hands-on lab environments where engineers practice real-world scenarios.

Create internal learning paths tailored to your environment. Storage administrators learn Portworx and CSI drivers. Network engineers learn CNI plugins and network policies. Security engineers learn RBAC, Pod Security Standards, and policy engines like Kyverno or OPA Gatekeeper. Each role maps existing expertise to Kubernetes-specific implementations.

Avoid the mistake of sending one person to a training course and expecting them to teach everyone else. Distributed learning across the team builds collective capability faster. Pair experienced Kubernetes practitioners with VMware veterans in daily work. The VMware engineer contributes operational rigor and production awareness. The Kubernetes engineer contributes platform-specific knowledge.

Upskilling timelines vary widely by team size, existing skill depth, and target complexity. Some organizations reach production readiness in a few months. Others need six months or longer, especially when teams are learning Kubernetes fundamentals alongside platform-specific tooling like Portworx, Kyverno, and GitOps workflows. Start structured training early and treat the pilot phase as a hands-on learning accelerator. This investment pays back through fewer misconfigurations, faster troubleshooting, and higher confidence during migration waves.

Execute in Phases

Phased migration reduces risk by limiting blast radius and building organizational confidence incrementally. Each wave teaches lessons applied to subsequent waves.

Phased Migration TImeline

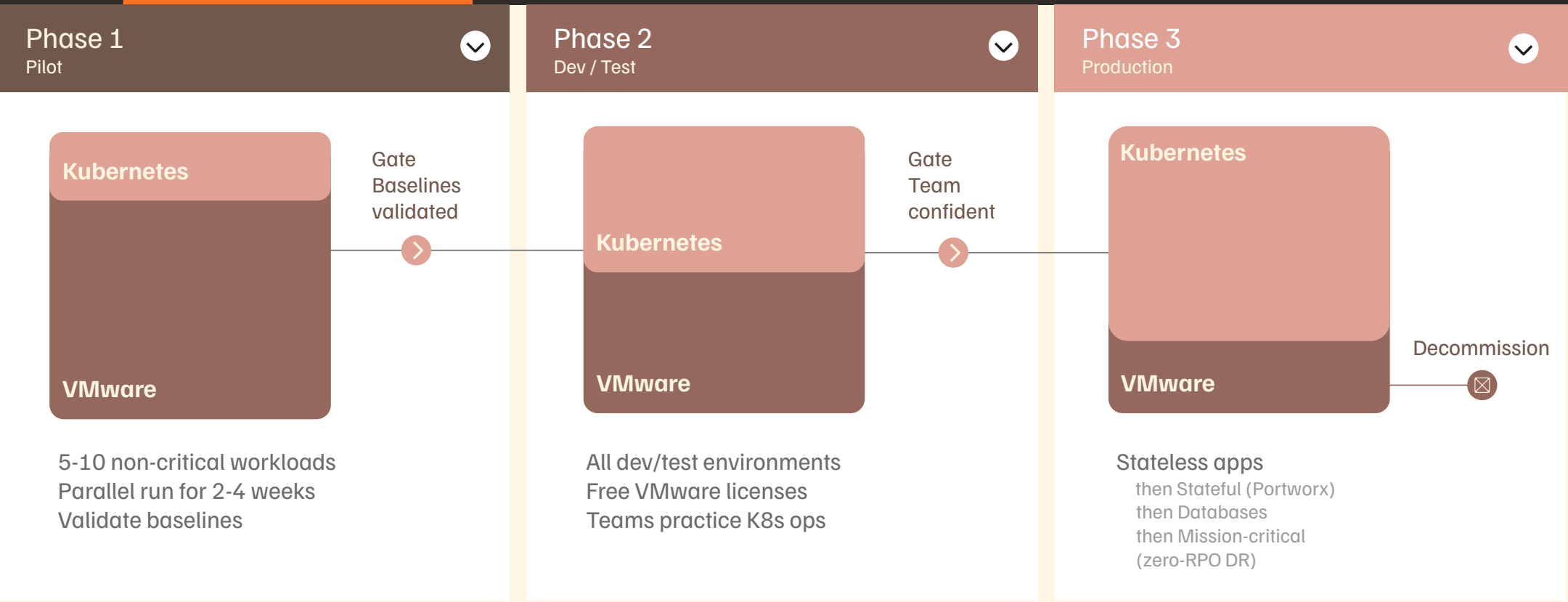


Figure 3: Three-phase migration timeline showing VMware footprint shrinking as Kubernetes adoption grows.

Each phase should include explicit success criteria. Define acceptable latency thresholds, error rates, recovery time objectives, and team confidence levels before promoting workloads and moving to the next wave.

Phase 1 is the pilot.

Select 5-10 non-critical workloads representing different application types. Include at least one KubeVirt VM workload to validate the lift-and-shift path alongside containerized workloads. Deploy Portworx for persistent storage. Configure monitoring, logging, and alerting. Run these workloads in parallel with their VMware counterparts for 2-4 weeks. Compare performance baselines. Document gaps and operational procedures.

Phase 1
Pilot

Phase 2 targets development and testing environments.

Use Portworx's open-source virtbench toolkit to benchmark KubeVirt VM provisioning. These workloads have lower SLA requirements and higher tolerance for disruption. Moving dev/test first frees VMware licenses and capacity while giving teams more practice with Kubernetes operations. Application teams begin deploying to Kubernetes as their default target.

Phase 2
Dev / Test

Phase 3 addresses production workloads in waves of increasing criticality.

Start with stateless web applications and API services. Progress to stateful workloads running on Portworx persistent volumes. Follow with database workloads using Portworx with features like synchronous replication and automated failover and follow careful validation of replication, failover, and consistency requirements balanced with any downtime tolerance. Complete with mission-critical systems requiring zero-RPO disaster recovery.

Phase 3
Production

Measure Success

Define metrics before the first workload moves. Track both technical and organizational indicators.

Technical metrics include application response time before and after migration, storage IOPS and latency compared to VMware baselines, pod scheduling time versus VM boot time, recovery time during failure scenarios, and resource utilization efficiency.

To help identify performance issues, Portworx have developed an open-source benchmarking toolkit called "virtbench.". Virtbench provides automated testing routines to measure and validate KubeVirt VM provisioning, boot times, network readiness, and failure recovery scenarios. It's designed for production environments running either KubeVirt or OpenShift Virtualization.

Organizational metrics include time-to-deploy for new applications, number of manual interventions required per week, mean time to resolution for incidents, team confidence scores through periodic surveys, and percentage of workloads migrated against the timeline.

Report progress to stakeholders regularly. Migration projects lose executive support when leadership sees no measurable outcomes. Weekly or biweekly dashboards showing workload counts, performance comparisons, and cost savings maintain visibility and funding.

Avoid Common Pitfalls

Organizations repeating the mistakes of earlier migrations lose time and credibility.

Your VMware operational discipline transfers directly to migration planning. You understand change management, maintenance windows, capacity planning, and phased rollouts. These skills apply to every stage of a Kubernetes transition. Assess the estate with the same rigor you apply to VMware upgrades. Categorize workloads using the 6 Rs framework. Build a platform team from your existing administrators. Upskill through structured learning paths and hands-on practice. Execute in phases, validate at every gate, and measure both technical and organizational outcomes. The tools change. The operational rigor stays the same.

Do not skip the assessment. Moving workloads without understanding dependencies creates cascading failures when source systems go offline.

Do not attempt a single migration event. "Big bang" migrations fail at enterprise scale. Phase the work.

Do not underestimate storage complexity. Kubernetes persistent storage requires careful planning around storage classes, access modes, backup policies, and disaster recovery. Portworx addresses these requirements through a unified platform, but the configuration still demands attention and testing.

Do not ignore organizational change. Technical migration without team enablement creates a platform nobody knows how to operate. Invest in people alongside infrastructure.

Do not replicate VMware patterns on Kubernetes. Kubernetes operates differently by design. Teams that create one-namespace-per-VM or treat pods as permanent fixtures miss the benefits of the platform. Embrace the declarative model.

Do not delay decommissioning the old environment. Organizations running both platforms indefinitely pay twice the infrastructure cost and split their team's attention. Set firm dates for VMware decommission after each phase validates successfully.

Continue Your Journey.

Learn more at **portworx.com**

Scan the QR code for a self-paced lab

portworx.com

800.379.PURE