Cloud Migration Playbook for SaaS Companies

Jared Knedler & Kyrylo Maznik April 21, 2026

Executive Summary

Most cloud migrations fail not because the technology is hard, but because the migration is treated as an IT project rather than an engineering program. The patterns that succeed — **infrastructure as code from day one, parallel-run validation, explicit rollback paths, FinOps discipline applied during migration** — are well-known but rarely all applied together. This playbook describes the operational discipline of running cloud migrations for SaaS companies in 6-12 weeks, with zero customer-

visible downtime and a 25-35% cost reduction versus naïve lift-and-shift.

Why Migration Discipline Matters

Naïve lift-and-shift migrations land 30-50% more expensive than the source environment they replaced — capacity over-provisioned, no reserved instance modeling, no right-sizing. The savings story arrives 6-12 months later, after a separate "cloud cost optimization project." A disciplined migration applies right-sizing, Reserved Instance / Savings Plan modeling, and FinOps tagging during the migration, capturing the savings on day one.

The bigger risk is downtime. Customers who experience multi-hour planned outages during a migration remember it; some churn. Modern cloud migration patterns — DNS-weighted shifts, dual-write databases, blue-green deployments — make zero customer-visible downtime achievable for nearly any SaaS workload.

The 6Rs Framework

AWS originally codified **six migration patterns** (the "6Rs"); they apply equally on Azure and GCP:

> **Rehost (lift-and-shift):** Move VMs to EC2 with minimal change. Fast but expensive long-term. Use as a temporary

measure when deadlines are tight; replatform afterward.

Replatform: Containerize the application, move to managed services (RDS, ElastiCache, S3). The most common pattern for on-prem to cloud — operational savings justify the engineering effort.

Refactor (re-architect): Restructure the application for cloud-native patterns (serverless, managed Kafka, event-driven). Reserved for the differentiated workloads where managed services unlock real capability.

Repurchase: Replace with a SaaS equivalent. Common for email, ticketing, CRM, internal tools.

Retire: Decommission rather than migrate. Most environments have 10-20% of workloads that are never accessed.

Retain: Keep on-prem. Hybrid is sometimes the right answer (latency-sensitive, regulatory, hardware-bound).

Choosing the right R

The choice depends on three variables: *"how much engineering time can we commit"*, *"how much runway / capital is available"*, and *"what are the exit conditions from the current environment"*. Most customers migrating from on-prem or co-lo land on a replatform pattern; customers already on one cloud moving to another typically refactor the differentiated workloads and lift-and-shift the rest.

Practical Implementation

Phase 1 — Discovery and dependency mapping (2 weeks)

Map every service, dependency, network flow, and data store. Tools that help: AWS Application Discovery Service, AzureMigrate, manual interview-driven dependency mapping. Output: an architecture diagram and a migration wave plan that sequences workloads by risk and dependency.

Phase 2 — Target architecture design (2 weeks)

Design with AWS Well-Architected pillars: Operational Excellence, Security, Reliability, Performance, Cost, Sustainability. Networking (VPC, Transit Gateway, PrivateLink), data tier (RDS, Aurora, DocumentDB), compute (EKS or ECS Fargate), observability (Datadog, CloudWatch), security (CIS Benchmarks, GuardDuty).

Phase 3 — Infrastructure as code (2 weeks)

Terraform modules for the target environment. Modules versioned, reviewed in PRs, tested in a non-prod replica before any production traffic shifts. Module annotations track which SOC 2 / HIPAA / PCI controls each module satisfies.

Phase 4 — Migration execution (3 weeks)

Workload-by-workload cutover with parallel-run validation:

Database migration: AWS DMS for most relational databases, with full + ongoing-replication mode allowing the source to keep operating. Native logical replication for very large databases (10TB+).

Stateless services: DNS-weighted shifts via Route 53 weighted records. Start at 1% traffic, validate, ramp.

Stateful application services: Dual-write pattern — write to both old and new, read from old, then flip the read flag.

Object storage: S3 Cross-Region Replication or AWS DataSync for bulk; ongoing application-level dual-write for new objects.

Phase 5 — Validation and optimization (2-4 weeks)

Post-cutover monitoring, right-sizing review against actual traffic patterns, Savings Plan / Reserved Instance modeling, final compliance evidence collection. Old environment retired with documented evidence.

Common Pitfalls

"Lift-and-shift then optimize": The optimization phase rarely happens. Right-size during migration, not afterward.

Database cutover panic: Migration teams often defer database cutover to the very end, then panic when DMS lag is unexpected. Migrate the database first (with dual-write) so you have weeks of operational practice before the cutover.

No rollback plan: Each migration wave needs a documented rollback that can complete in under one hour. Skip this and pressure to "just push through" leads to production incidents.

Compliance as afterthought: Migration completes, then the SOC 2 auditor asks for evidence of every control change. Six-month follow-on. Solution: annotate Terraform modules with control responsibilities from week one.

Skipping the discovery phase: Teams underestimate dependency complexity. The hidden integrations (vendor APIs, internal services, batch jobs running on cron servers) are what break the cutover.

Measured Outcomes

6-week median migration window for a mid-stage SaaS with single primary database and <50 microservices

Zero customer-visible downtime through parallel-run validation patterns

25-35% cost reduction versus naïve lift-and-shift, captured on day one

SOC 2 / HIPAA evidence package in place at migration completion, no follow-on compliance project

Day-1 operations inheriting Datadog SLOs, IaC-managed infrastructure, runbooks, and on-call rotation from the migration team

How Jacobian Helps

We run cloud migrations as engineering programs, not IT projects. Detailed dependency mapping, infrastructure-as-code from day one, parallel-run validation against production traffic, explicit rollback paths at every stage, and FinOps tagging applied during migration rather than retrofitted afterward. Because our team operates the post-migration environment, day 1 of the new cloud is day 1 of operations — not the start of a separate "ops handoff" project.

Resource Details

Author: Jared Knedler & Kyrylo Maznik

Published: April 21, 2026

Categories:

Cloud Infrastructure | Migration | SaaS

Download

Full Document

About This Resource

A practical cloud migration playbook covering the 6Rs framework, parallel-run validation patterns, FinOps discipline during migration, and how to land compliance posture intact rather than as a six-month follow-on project.

Categories:

Cloud Infrastructure **Migration** **SaaS**