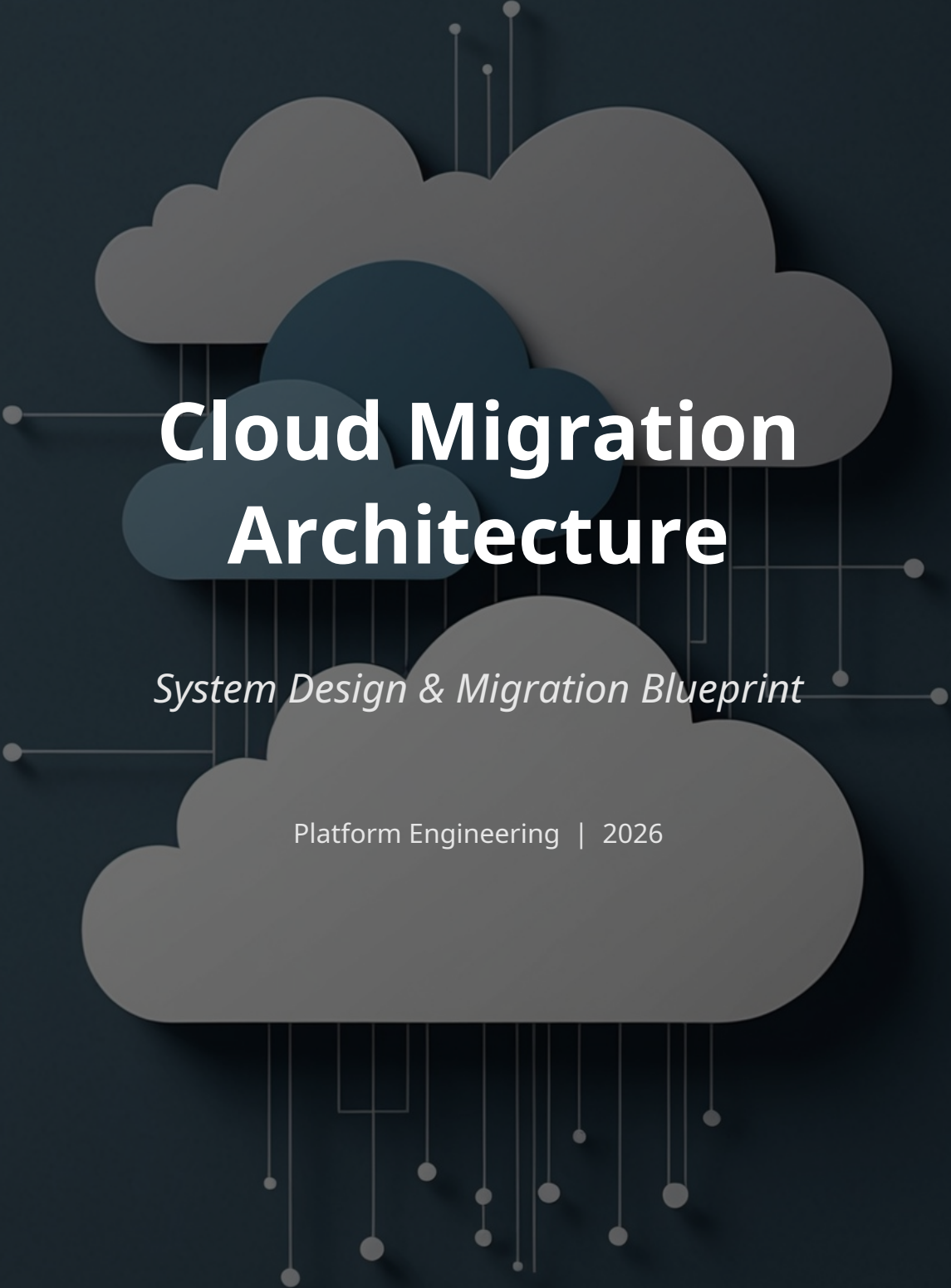




Cloud Migration Architecture

System Design & Migration Blueprint

Platform Engineering | 2026

Overview & Goals

This document defines the target system architecture and execution plan for migrating our legacy on-premise infrastructure to a modern, cloud-native platform. It is intended as a shared technical reference for engineering, security, and operations stakeholders throughout the migration program.

Primary Goals

- Eliminate single points of failure via horizontally scalable compute and managed data services
 - Reduce operational overhead through managed load-balancing, caching, and database services
 - Improve global latency and availability using a CDN edge layer
 - Establish clear security and network boundaries aligned to VPC/subnet isolation
 - Deliver the migration in incremental, low-risk phases with defined rollback points
-

Success Criteria

- Zero unplanned downtime during cutover window
- 99.9% service availability post-migration
- All workloads migrated within the 20-week program timeline
- Monthly infrastructure spend held within the estimated budget envelope

Current State Architecture

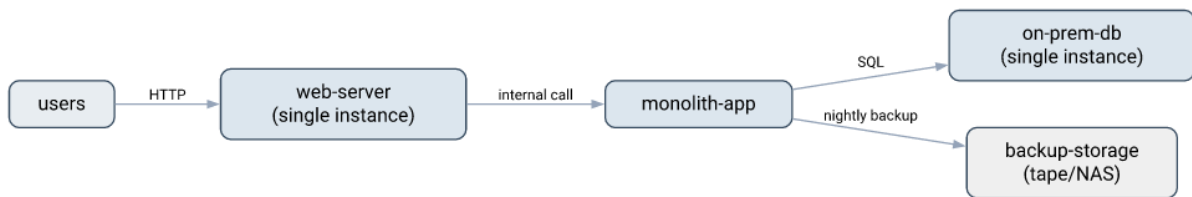


Figure 1: Current State Architecture

The existing environment is a single-region, on-premise deployment built around a monolithic application tier and a single database instance. This design carries limited horizontal scalability and several single points of failure, motivating the proposed target architecture below.

Proposed Target Architecture

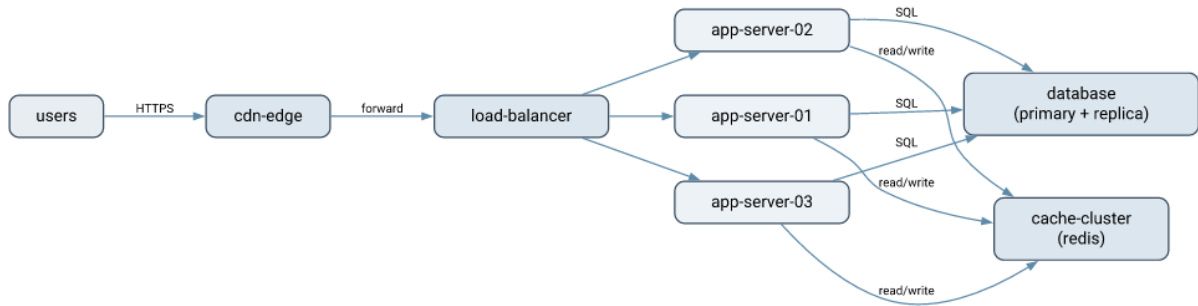


Figure 2: Proposed Target Architecture

The target architecture introduces a load-balancer fronted by a cdn-edge layer, distributing traffic across multiple stateless app-server instances. A shared cache-cluster reduces read load on the primary database, which is deployed with a replica for resilience.

Data Flow Diagram

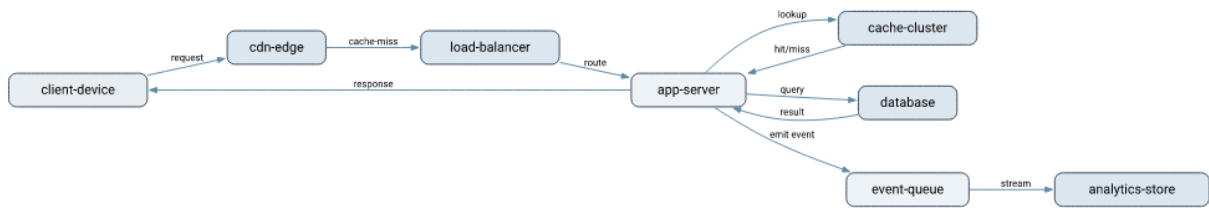


Figure 3: End-to-End Data Flow

Client requests are first evaluated at the cdn-edge layer; cache misses are forwarded to the load-balancer and routed to an available app-server. The application checks the cache-cluster before querying the database, and emits asynchronous events to an event-queue for downstream analytics.

Security & Network Boundaries

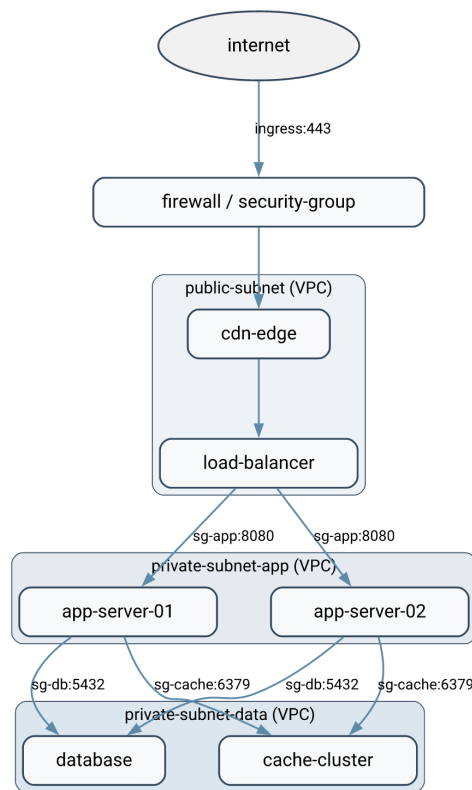


Figure 4: Security & Network Boundary Zones

Network isolation is enforced through a segmented VPC. The `public-subnet` hosts only the `cdn-edge` and `load-balancer`, while application and data tiers reside in isolated private subnets, each protected by dedicated security groups permitting only the minimum required ports.

Migration Phases Timeline

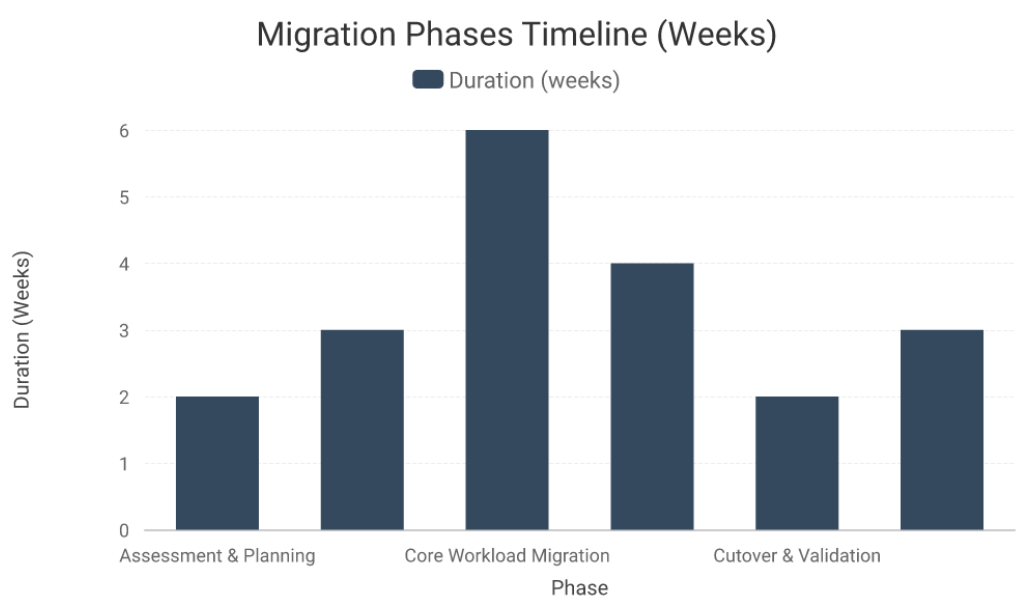


Figure 5: Migration Phases Timeline (Gantt-style, in weeks)

The migration is executed across six sequential phases spanning a total of 20 weeks. Each phase concludes with a validation checkpoint before proceeding to the next.

Phase Breakdown

Table 1: Migration Phase Durations

Phase	Duration (weeks)
Assessment & Planning	2
Pilot Migration	3
Core Workload Migration	6
Data Migration & Sync	4
Cutover & Validation	2

Phase	Duration (weeks)
Optimization & Decommission	3
Total	20

Cost Estimate

Estimated recurring monthly infrastructure cost for the target architecture, based on projected workload sizing.

Table 2: Estimated Monthly Infrastructure Cost

Component	Estimated Monthly Cost (USD)
app-server compute (3x instances)	\$4,200
database (primary + replica)	\$2,800
load-balancer	\$350
cdn-edge	\$600
cache-cluster (Redis)	\$900
Storage & Backup	\$750
Networking (VPC / Data Transfer)	\$500
Monitoring & Security	\$400
Total	\$10,500

Risks & Mitigations

Table 3: Key Migration Risks & Mitigation Strategies

Risk	Impact	Mitigation
Data loss/corruption during cutover	High	Continuous replication + verified backup snapshots before cutover window
Extended downtime during cutover	High	Blue/green cutover strategy with instant rollback to on-prem-db
Cost overrun from oversized instances	Medium	Right-sizing review after Pilot phase; auto-scaling on app-server tier
Security misconfiguration in new VPC	High	Mandatory security review of subnet/security-group rules before go-live
Team unfamiliarity with new stack	Medium	Pilot phase used as hands-on enablement; runbooks authored pre-cutover
Third-party integration incompatibility	Medium	Integration test suite run against pilot environment before core migration