Quarterly Business Review (QBR)

Project: Scalable Web Application Deployment on DigitalOcean Kubernetes (DOKS)
Client: [SaaS Company]
Prepared By: Sai Hari Krishnan
Date: [06/17/2025]

Summary

We have successfully deployed a scalable, containerized JavaScript-based web application on DigitalOcean Kubernetes (DOKS). The goal was to deliver a solution optimized for performance, reliability, and cost-efficiency. The architecture includes automated scaling, a managed load balancer, and resilient infrastructure, all within the DigitalOcean ecosystem.

Current Infrastructure Overview

Component	Description
Platform	DigitalOcean Kubernetes (DOKS)
App Runtime	Node.js (Dockerized container)
Deployment	Kubernetes Deployment with 2 replicas
Scaling	Horizontal Pod Autoscaler (CPU > 50%, scales to 5 pods)
Load Balancer	DigitalOcean LB (Service type: LoadBalancer)
Node Pool	2 × s-2vcpu-4gb droplets (shared CPU), autoscaling ON

Performance & Reliability

Capability	Implementation & Benefit
High Availability	Multi-replica pods across nodes, served by LoadBalancer
Autoscaling	HPA increases pod count on CPU demand
Self-Healing	Kubernetes restarts unhealthy pods automatically
Optimized Runtime	Dockerized app ensures fast startup and minimal latency

---

Cost Summary & Optimization

Component	Quantity	Monthly Estimate
DOKS Nodes	2	$36
Load Balancer	1	$10
Block Storage	Minimal	~$1
Total	—	**~$47/month**

Cost Optimization Opportunities :

- **Enable Node Autoscaler**: Save by scaling down to 1 node during low traffic hours
- **Use `s-1vcpu-2gb` nodes**: Replace with lower-cost droplets for dev or staging
- **Clean up unused DO resources**: Remove idle LoadBalancers, volumes, and services
- **Use metrics**: Monitor via DO's Kubernetes dashboard to right-size resource limits
- **Consolidate apps**: Host multiple lightweight services on shared nodes

---

Risks & Mitigation

Risk	Mitigation Strategy
Node failure	DOKS automatically spreads pods across healthy nodes
Traffic spikes (e.g., promo)	HPA handles surge with pod autoscaling
Image pull errors	Use stable tags + readiness probes for failover
Unexpected billing spikes	Enable DigitalOcean alerts + regular resource audits

---

Recommendations (Next Steps)

- **CI/CD Pipeline**: Automate with GitHub Actions for smoother deployments
- **Monitoring**: Add Prometheus + Grafana, or use DigitalOcean Monitoring out-of-the-box to observe usage patterns and adjust infrastructure.
- **Managed DB**: Transition to Digital Ocean Managed PostgreSQL as data needs grow.
- **CDN Integration**: Consider Digital Ocean Spaces CDN for better global delivery

---

Final Notes

The current deployment offers a production-grade SaaS foundation that scales with demand while staying within predictable cost boundaries. The team is now equipped with a tested reference architecture and documented deployment guide to manage, monitor, and grow their application independently.