FlaskShop

1. Current Infrastructure

a) **Cluster Composition**

Three basic Droplet nodes (smallest recommended size) for high availability and redundancy.
No attached block storage; the app is stateless, minimizing storage costs.
One LoadBalancer service fronts the cluster, providing a public endpoint for users.

b) **Deployment Characteristics**

Application runs as stateless pods, managed by a Kubernetes Deployment.
Default replica count is three, distributed across nodes.
Cluster autoscaling is enabled: nodes scale up to five if average CPU usage exceeds 70%.
Resource requests and limits are set to ensure efficient pod scheduling and avoid over-provisioning.
Docker image is optimized for size and speed (Python slim base).

2. Recommendations for Future Scaling & Cost Optimization

a) **Review Node Sizing**

Periodically assess Droplet sizes and consider resizing if workloads increase or decrease.

Enable HPA to automatically adjust pod replicas based on CPU/memory usage, further optimizing resource utilization.

b) **Optimize Resource Requests/Limits**

Continuously tune pod resource requests and limits to match actual usage, preventing wasted capacity.

c) **Monitor Load Balancer Usage**

Ensure only necessary LoadBalancer services are provisioned; consider using Ingress for multiple apps to reduce networking costs.

d) **Leverage Reserved/Committed Resources**

If usage stabilizes, consider DigitalOcean’s reserved or committed Droplet options for cost savings.

e) **Automate Idle Resource Cleanup**

Use scripts or policies to automatically detect and remove unused nodes, services, or deployments.

3. Risks & Challenges, Mitigation Strategies

a) **Unexpected Traffic Spikes**

Risk: Sudden increases in traffic may exceed autoscaling limits.

Mitigation: Set up proactive monitoring and alerts; consider increasing max node count or pre-scaling during known busy periods.

b) **Resource Over-Provisioning**

Risk: Overestimating resource needs can lead to unnecessary costs.

Mitigation: Regularly review usage metrics and adjust requests/limits and node sizes.

c) **Security & Compliance**

Risk: Exposed endpoints may be vulnerable.

Mitigation: Use firewalls, restrict access, and keep dependencies up to date.

d) **Scaling Limits**

Risk: Cluster autoscaling may not react quickly enough to rapid demand changes.

Mitigation: Test autoscaling response; set conservative thresholds and ensure sufficient max node count.

Summary

The current infrastructure is well-optimized for cost and reliability. Continued monitoring, periodic review of resource allocation, and proactive scaling strategies will ensure the Flask SaaS app remains performant and cost-effective as usage grows.