NEXA SITE RELIABILITY ENGINEERING | INTERNAL – OPERATIONS

Infrastructure Runbook

Nexa Payments API – Production Environment

Document ID	Version	Last Updated	Classification
RB-PAY-0142	4.3	2026-06-30	Internal – Restricted

Owner Elena Cross, SRE Lead – `elena.cross@nexa.io`

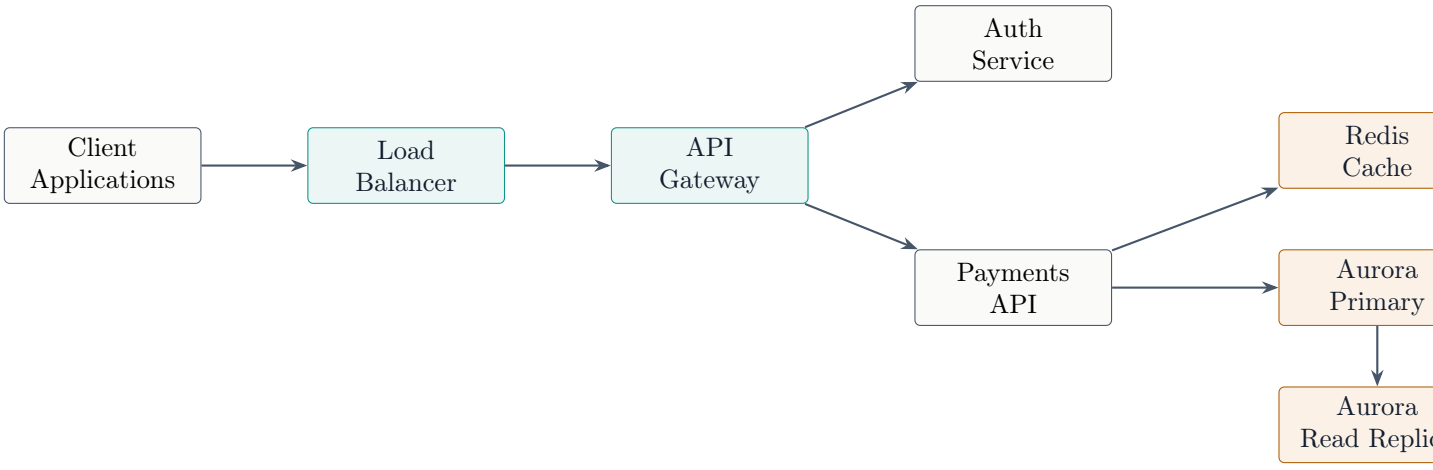
Scope Deployment, scaling, credential rotation, and incident response for the Nexa Payments API service running on cluster `nexa-prod-use1`

Audience On-call SRE, Payments Platform engineers, Incident Commanders

Version	Date	Change
4.3	2026-06-30	Added rollback fast-path for canary failures; updated escalation roster
4.2	2026-04-11	Rotated database failover commands to new Aurora-compatible tooling
4.0	2026-01-08	Full rewrite following migration to `nexa-prod-use1` cluster

1 Service Overview

The Nexa Payments API is a stateless Go service handling transaction authorization, settlement callbacks, and merchant ledger writes for the Nexa platform. It serves approximately 3,200 requests per second at peak (weekday 14:00–18:00 UTC) with a 99.95% monthly availability target.



Cluster	`nexa-prod-use1` (Kubernetes 1.29, 3 node pools)
Namespace	`payments`
Deployment	`nexa-payments-api` (min 6 / max 24 replicas, HPA on CPU + queue depth)
Database	Aurora PostgreSQL 15, cluster `nexa-pay-aurora-prod`
Cache	Redis 7, cluster `nexa-pay-cache-prod`
Dashboards	Grafana: `grafana.nexa.internal/d/payments-overview`

2 Escalation & Contacts

Role	Name	Contact	Response Window
On-call Primary	Marcus Webb	PagerDuty: `payments-primary`	5 min (SEV1/2)
On-call Secondary	Priya Anand	PagerDuty: `payments-secondary`	15 min
Engineering Manager	David Kim	`david.kim@nexa.io`	30 min (SEV1 only)
Incident Commander	Rotation – see IC calendar	Slack: `#ic-oncall`	Immediate
Database On-call	Sofia Reyes	PagerDuty: `db-oncall`	15 min

Note

All SEV1 and SEV2 incidents must be declared in Slack channel `#incidents` using the `/incident` command *before* remediation begins. This creates the incident record, timeline, and a dedicated bridge automatically.

3 Standard Operating Procedures

3.1 Deploying a New Release

Deployments to `nexa-prod-use1` run through the canary pipeline. Do not deploy directly with `kubectl apply` outside of a declared incident.

1. Confirm the release build passed staging soak (minimum 30 minutes, error rate under 0.1%).
2. Trigger the canary rollout from the deploy pipeline.
3. Watch the canary dashboard for 10 minutes; canary receives 5% of production traffic.
4. Promote to full rollout only if p99 latency and error rate stay within baseline.

```
# 1.  Verify current rollout status
kubectl -n payments rollout status deployment/nexa-payments-api

# 2.  Trigger canary (5% traffic weight)
nexactl deploy start -service payments-api -version v2.114.0 -canary 5

# 3.  Watch canary error budget in real time
nexactl deploy watch -service payments-api -window 10m
```

```
# 4.  Promote to 100% once green
nexactl deploy promote -service payments-api
```

3.2 Scaling the Service

The horizontal pod autoscaler handles routine load. Manual scaling is only for anticipated traffic spikes (e.g. merchant promotions) or HPA misbehavior.

```
# Inspect current HPA state
kubectl -n payments get hpa nexa-payments-api

# Temporarily raise the floor ahead of a known traffic event
kubectl -n payments patch hpa nexa-payments-api \
-p '{"spec":{"minReplicas":14}}'

# Revert once the event has passed
kubectl -n payments patch hpa nexa-payments-api \
-p '{"spec":{"minReplicas":6}}'
```

3.3 Rotating Database Credentials

⚠ Warning

Credential rotation triggers a rolling pod restart. Perform only during the posted maintenance window (Tuesdays 02:00–03:00 UTC) unless responding to a suspected credential leak.

```
# 1.  Issue new credentials in Vault
vault write database/rotate-root/nexa-pay-aurora-prod

# 2.  Sync the Kubernetes secret from Vault
nexactl secrets sync -cluster nexa-prod-use1 -secret pay-db-creds

# 3.  Roll the deployment to pick up the new secret
kubectl -n payments rollout restart deployment/nexa-payments-api
```

4 Incident Response

4.1 Severity Classification

Level	Impact	Example	Ack SLA
SEV1	Full outage or data loss risk	Payments API returning 5xx cluster-wide	5 min
SEV2	Major degradation	p99 latency above 3s, partial region failure	15 min
SEV3	Minor, contained impact	Single pod crash-looping, one merchant affected	1 hour
SEV4	No customer impact	Non-critical alert, capacity warning	Next business day

4.2 General Incident Workflow

1. **Acknowledge** the page in PagerDuty within the SLA for the assessed severity.
2. **Declare** the incident via `/incident` in Slack; this opens `#inc-{id}` and a Zoom bridge.
3. **Assess** severity using the table above; adjust as new information arrives.
4. **Mitigate** first, diagnose root cause second – restore service, don't debug in production under fire.
5. **Communicate** status every 15 minutes for SEV1, every 30 for SEV2, in the incident channel.
6. **Resolve** and downgrade PagerDuty once metrics are stable for 15 consecutive minutes.
7. **Postmortem** within 3 business days for SEV1/SEV2, posted to the incident wiki.

4.3 Runbook: API Latency Degradation

Trigger: Alert `payments-api-p99-latency-high` fires when p99 exceeds 1.5s for 5 consecutive minutes.

1. Check whether the Aurora primary is under connection pressure.
2. Check whether the Redis cache hit ratio has dropped.
3. If a recent deploy correlates with the onset, roll back immediately (Section 4.4).

```
# Database connection saturation
nexactl db connections -cluster nexa-pay-aurora-prod

# Cache hit ratio over the last 30 minutes
nexactl cache stats -cluster nexa-pay-cache-prod -window 30m

# Correlate with recent deploys
nexactl deploy history -service payments-api -limit 5
```

⚠ Critical – Read Before Executing

If Aurora primary CPU exceeds 90% for more than 5 minutes, do *not* restart the writer instance. Failing over under load can extend the outage. Page the Database on-call (Section 2) and force a read-replica promotion only if directed by the Incident Commander.

4.4 Rollback Procedure

```
# 1.  Identify the last known-good version
nexactl deploy history -service payments-api -limit 5

# 2.  Roll back immediately, bypassing canary
nexactl deploy rollback -service payments-api -to v2.113.2 -skip-canary

# 3.  Confirm all pods are on the rolled-back image
kubectl -n payments get pods -l app=payments-api \
-o jsonpath='{.items[*].spec.containers[0].image}'

# 4.  Watch error rate return to baseline
nexactl deploy watch -service payments-api -window 10m
```

5 Monitoring & Alert Thresholds

Metric	Warning	Critical	Source
p99 latency	> 800ms	> 1.5s	Grafana
5xx error rate	$> 0.5\%$	$> 2\%$	Grafana
Aurora CPU utilization	$> 75\%$	$> 90\%$	CloudWatch
Redis cache hit ratio	$< 92\%$	$< 80\%$	Grafana
Pod restart count (5 min)	≥ 3	≥ 8	Kubernetes

6 Appendix: Quick Reference

Task	Command
Tail live logs	`nexactl logs tail -service payments-api`
Open a shell in a pod	`kubectl -n payments exec -it {pod} - /bin/sh`
Force a config reload	`nexactl config reload -service payments-api`
Check current on-call	`nexactl oncall -team payments`
Drain a single node	`kubectl drain {node} -ignore-daemonsets`

This runbook is reviewed quarterly by the Payments SRE team. Report inaccuracies in `#sre-runbooks` or open a pull request against the `runbooks` repository.