Arcline

November 6, 2026

SAMPLE DELIVERABLE

Reliability & Cost Audit: Findings Report

A written, prioritized summary of what the platform costs, what it risks, and what to fix first.

SAMPLE Fictional demonstration data · Prepared by Arcline

PREPARED FOR

Northwind Payments

Fintech / payments (PCI DSS scope) · 28 engineers · 3 Kubernetes clusters · Azure and AWS

Cloud spend, the Kubernetes platform, delivery pipeline, on-call practice, and the evidence behind SOC 2 and PCI DSS controls, across production and staging.

01 Executive summary

$24.6K	3	3	2
RECOVERABLE SPEND / MONTH	HIGH FINDINGS	MEDIUM FINDINGS	LOW FINDINGS

This report summarizes a reliability and cost audit of Northwind Payments, completed in November 2026. We reviewed the cloud spend across both providers, the Kubernetes platform, the delivery pipeline, the on-call practice, and the evidence behind the SOC 2 and PCI DSS controls. The goal was to find what the platform costs and what it risks, not to grade the team.

The platform is better built than most at this size: three clusters with consistent namespaces, GitOps already in place for two thirds of services, and a median deploy time of eleven minutes with a working rollback. Nothing we found suggests an active breach or an unrecoverable failure. But $24,600 a month is not buying anything, and the two controls an auditor will test first — privileged audit logging and tested restores — are not yet in place. Both are cheap to fix and expensive to explain.

WHAT'S GOING WELL

- Three clusters, one namespace convention, and real separation between staging and production
- GitOps already covers two thirds of services, and rollback is practised rather than theoretical
- Median deploy time of eleven minutes, with a change history that maps to pull requests
- Image scanning blocks critical vulnerabilities before a deploy
- No evidence of a breach, and no unrecoverable failure in the platform's history

TOP PRIORITIES

1. Agree SLOs for checkout and adopt an error-budget policy
2. Delete the idle capacity and orphaned disks this week
3. Turn on Kubernetes audit logging before the next audit cycle

02 Prioritized findings

Every finding, ranked by what it costs in dollars and in risk. Detail follows in the next section.

ID	FINDING	SEVERITY	OWNER	TIMELINE
R-01	No SLOs or error budgets on the payment path	HIGH	Platform	within 14 days
R-02	38% of production spend is not buying anything	HIGH	Platform	within 14 days

ID	FINDING	SEVERITY	OWNER	TIMELINE
R-03	Kubernetes audit logging is off, and control-plane logs last three days	HIGH	Platform	before the next audit
R-04	Terraform drift: 14 production resources are outside the code	MEDIUM	Platform	within 30 days
R-05	On-call is one engineer, with no escalation and no runbooks	MEDIUM	Platform	within 30 days
R-06	Backups have never been restored, and retention is seven days	MEDIUM	Platform	within 30 days
R-07	40% of workloads have no resource requests or limits	LOW	Platform	within 60 days
R-08	A shared cluster-admin kubeconfig lives in the team vault	LOW	Platform	within 60 days

03 Detailed findings

R-01 · No SLOs or error budgets on the payment path

HIGH

Affected: checkout-api, ledger-api, webhook-dispatcher · Owner: platform · within 14 days

WHY IT MATTERS Without a written objective, reliability is whatever the loudest incident last week made it. The team cannot say whether last month was good or bad, so every conversation about shipping faster becomes an argument rather than arithmetic.

WHAT WE FOUND Three services sit on the critical path, and none has an SLI, an SLO, or an error budget. The dashboards show CPU and memory, so nobody can answer “what share of checkouts succeeded last month”. Alerts fire on host metrics: a busy database pages someone at 2am, while a spike in failed checkouts does not page anyone at all.

EVIDENCE No SLO definitions in the repository or the monitoring configuration. 63 alert rules, 58 of them on infrastructure metrics, 5 on error rate, none tied to a user-visible objective.

RECOMMENDATION Define three SLIs (checkout success, checkout latency p95, webhook delivery), agree the SLOs with product, and adopt an error-budget policy. Retire any alert that is not tied to a symptom threatening an SLO.

R-02 · 38% of production spend is not buying anything

HIGH

Affected: Azure and AWS production accounts · Owner: platform · within 14 days

WHY IT MATTERS Every dollar of waste is a dollar not spent on reliability, and the waste compounds quietly because no single line on the invoice looks wrong.

WHAT WE FOUND Production runs at $64,200 a month across two clouds. $24,600 of that is idle or oversized: a node pool left running after a cancelled migration, a 32-vCore Postgres averaging 8% CPU, 42 unattached disks, and 90-day log retention that nobody queries. None of it appears as an anomaly on the bill.

EVIDENCE Billing export, 90 days. Idle node pool: zero pods for 61 days. Postgres average CPU 8.2%, peak 19%. Orphaned disks: 42 unattached, $2,900/month.

RECOMMENDATION Delete the idle pool and the orphaned disks this week, rightsize the database next sprint and commit the stable baseline to a savings plan. The detail is in the Cloud Cost Analysis.

R-03 · Kubernetes audit logging is off, and control-plane logs last three days

HIGH

Affected: Both production AKS clusters · Owner: platform · before the next audit

WHY IT MATTERS	An auditor will ask who changed what, and when. Today the platform cannot answer, which turns a routine audit into a reconstruction project and an incident into a guess.
WHAT WE FOUND	The clusters run the default audit policy, so most write operations are never recorded. Control-plane logs are retained for three days. There is no record of a kubectl exec into production and no way to distinguish a reviewed deploy from a manual change made during an incident.
EVIDENCE	Audit policy unset on both clusters. Log Analytics retention 3 days. No audit-log export configured. No RBAC change history beyond what Git happens to contain.
RECOMMENDATION	Apply a baseline audit policy (Metadata at minimum, RequestResponse for secrets and RBAC), ship the logs to immutable storage, and set 12-month retention. This is also the evidence for CC7.2.

R-04 · Terraform drift: 14 production resources are outside the code

MEDIUM

Affected: Production workspace · Owner: platform · within 30 days

WHY IT MATTERS	The repository says one thing and the cluster says another. The next apply either overwrites someone's hotfix or silently reverts a security setting, and nobody knows which until it happens.
WHAT WE FOUND	A plan against the production workspace shows 14 resources would change with no code change: two network security groups, a node-pool autoscaler setting, nine Kubernetes objects applied by hand during an incident, and two secrets managed outside the pipeline.
EVIDENCE	terraform plan, production: 14 to change, 0 to add, 0 to destroy. Git history contains no corresponding commit.
RECOMMENDATION	Import or codify the drifted resources, then make the pipeline the only path to production. Add a nightly drift check that opens a ticket instead of waiting to be discovered during an incident.

R-05 · On-call is one engineer, with no escalation and no runbooks

MEDIUM

Affected: All production services · Owner: platform · within 30 days

WHY IT MATTERS	A rotation of one is not a rotation. When that person is asleep, ill, or on a plane, nobody is on call, and the bus factor is exactly the same person.
WHAT WE FOUND	The same engineer takes every page, every week. There is no secondary, no escalation policy beyond a message in Slack, and no runbooks for the five alerts that page most often. Two of those alerts fired more than 40 times this quarter.
EVIDENCE	PagerDuty schedule: one member. Escalation policy: notify the same user. 213 pages in 90 days; the top two alerts account for 88 of them.
RECOMMENDATION	Design a two-person rotation with an escalation path, write runbooks for the top five alerts, and make alert quality a standing monthly review. This is on-call design, not 24/7 staffing.

R-06 · Backups have never been restored, and retention is seven days

MEDIUM

Affected: Managed Postgres, persistent volumes · Owner: platform · within 30 days

WHY IT MATTERS	An untested backup is a hope. A seven-day window also means a slow, quiet corruption is discovered after the good copies have already expired.
WHAT WE FOUND	Nightly snapshots run and report success. Nobody has restored one in the platform's history. Database retention is seven days, persistent volumes have no snapshot policy at all, and no recovery time is documented.
EVIDENCE	Backup policy: 7-day retention, no point-in-time recovery. No restore record. PVC snapshot policy absent.
RECOMMENDATION	Enable point-in-time recovery, extend retention to 35 days, configure volume snapshots, and run a quarterly restore drill that is timed and written down. That record is the evidence A1.2 asks for.

R-07 · 40% of workloads have no resource requests or limits

LOW

Affected: 11 of 27 Deployments in production · Owner: platform · within 60 days

WHY IT MATTERS	Without requests the scheduler is guessing, so one noisy workload can starve its neighbours and the node pool is sized from fiction rather than from measured demand.
WHAT WE FOUND	Eleven Deployments define no CPU or memory requests and 16 define no limits. The node pools are sized from peak observed usage, which is inflated by exactly this absence.
EVIDENCE	kubectl audit across production namespaces: 11/27 without requests, 16/27 without limits. No LimitRange in any namespace.
RECOMMENDATION	Set requests from a two-week usage baseline, add a LimitRange per namespace, and let the autoscaler reclaim the difference. Expect a smaller node pool as a result.

R-08 · A shared cluster-admin kubeconfig lives in the team vault

LOW

Affected: Both production clusters · Owner: platform · within 60 days

WHY IT MATTERS	One credential that can do anything, held by everyone, is the finding an auditor writes first — and it makes every other access control in the report unprovable.
WHAT WE FOUND	A single cluster-admin kubeconfig is stored in the shared password manager and used for break-glass access by four engineers. There is no rotation schedule and no record of who used it or when.
EVIDENCE	Vault item shared with four users. Last rotated 14 months ago. No session logging on either cluster.
RECOMMENDATION	Replace it with per-person, time-bound access through the identity provider, and log every session. Rotate the shared credential now and retire it once the replacement is in place.

04 The next 90 days

The full owner-by-owner sequence is in the 90-Day Remediation Plan. This is the order of work and what it is worth.

WHEN	WORK	MONTHLY EFFECT	EFFORT
Week 1–2	Delete the idle node pool and 42 orphaned disks	+$7,200	1 day
Week 1–2	Agree SLOs and an error-budget policy for checkout	Risk	2 days
Week 3–4	Enable audit logging and 12-month retention	Compliance	3 days
Week 3–4	Rightsize the managed database	+$5,200	1 week
Week 5–8	Codify drift and add a nightly drift check	Risk	1 week
Week 5–8	Two-person rotation, escalation, five runbooks	Risk	1 week
Week 9–12	Point-in-time recovery and the first timed restore drill	Compliance	3 days
Week 9–12	Commit the stable baseline to a savings plan	+$6,800	1 week