ÆNIX

Platform Engineering Maturity Assessment

Score your organization across 8 dimensions and 5 maturity stages to find where platform investment pays back fastest.

For Heads of Platform, Infrastructure & Engineering · aenix.io

How to use this assessment

This self-assessment scores your platform engineering practice on a **5-stage maturity scale**: Pre-platform → Shared infrastructure → Self-service primitives → Internal Developer Platform → Mature. It is the same baseline framework Aenix uses during Platform Readiness Assessments before recommending Phase 2 investment.

Use it in three ways:

- [ ] **Self-assessment (recommended first pass).** One person — typically a platform lead or VP Engineering — scores in 30–45 minutes.
- [ ] **Workshop assessment (recommended for investment decisions).** Bring 4–8 people from platform engineering, product engineering, security, and ops, and score each dimension together. Disagreements about stage are themselves diagnostic.
- [ ] **Pre-engagement baseline.** Score before engaging Aenix or any other consultancy; have the assessment in hand at the first call.

The 5 maturity stages

The same five stages apply consistently across all eight dimensions.

Stage	Name	Summary
Stage 1	Pre-platform	Each team for itself.
Stage 2	Shared infrastructure	Central team manages shared infrastructure.
Stage 3	Self-service primitives	Teams self-serve common operations.
Stage 4	Internal Developer Platform	Platform-as-product, full IDP.
Stage 5	Mature	Compounding returns.

Scoring rules

- [ ] For each dimension, choose the stage that **best matches your current state** (not your target).
- [ ] If you fall **between** two stages, use the lower number unless the transition is well underway (then use the upper).
- [ ] Half-scores are allowed only for the average — individual dimensions are integer 1–5.
- [ ] Score **honestly** — the value is in identifying where to invest, not in confirming a comfortable picture.

> Score in pencil, then bring the sheet to a workshop. The conversation about why someone scored a dimension 2 rather than 3 is more valuable than the number itself.

---

1. Workload portability

How readily can a workload be moved between substrates (cloud regions, datacenters, providers, Kubernetes clusters) without rewrite?

Stage	Name	Description
Stage 1	Pre-platform	Each team configures its own infrastructure. No IaC. Bare-metal scripts, click-ops. Workloads embed substrate assumptions deep in code. Vendor lock-in everywhere.
Stage 2	Shared infrastructure	Central team manages shared infrastructure. Partial IaC (Terraform / CloudFormation in patches). Some workloads portable across teams; substrate moves still require redesign.
Stage 3	Self-service primitives	IaC for most provisioning. Teams can self-serve common operations. Workloads usually expressible as Kubernetes manifests plus Helm. Substrate-specific assumptions isolated.
Stage 4	Internal Developer Platform	Workload portability is structural. Multi-substrate-aware (region failover, cluster migration). Vendor-locked services explicitly tagged and quarantined.
Stage 5	Mature	Workloads use platform abstractions only. Portable across substrates with predictable effort. Drift detection automated. New substrates onboarded as pluggable backends.

Your score (1–5): _____ Why? Evidence?

_____

2. GitOps adoption

To what extent are infrastructure and platform changes declarative, version-controlled, PR-reviewed, and deployed via continuous reconciliation?

Stage	Name	Description
Stage 1	Pre-platform	Manual changes. SSH, click-ops, kubectl by hand. No source of truth for environment state.
Stage 2	Shared infrastructure	Some IaC committed to git, but applied manually. PRs are aspirational. Drift accumulates.
Stage 3	Self-service primitives	IaC plus GitOps for some workloads (Argo CD / Flux for app deploys). Infra changes still mixed manual and automated. PR review on platform changes.
Stage 4	Internal Developer Platform	GitOps for app and infra. Continuous reconciliation. Drift surfaced and remediated. PRs are the change-management channel. Audit trail complete.
Stage 5	Mature	Full declarative model — clusters, networks, IAM, observability, all reconciled from git. Drift impossible without an alert. PRs gated by automated policy (OPA / Gatekeeper, Conftest).

Your score (1–5): _____ Why? Evidence?

_____

3. Observability unification

Do engineers have a single, reliable way to see metrics, logs, traces, and SLOs across all environments and tenants?

Stage	Name	Description
Stage 1	Pre-platform	Each team picks its own tools. Tribal knowledge of where to look. Outage post-mortems reveal fragmented data.
Stage 2	Shared infrastructure	Centralised metrics and logs (e.g., Prometheus + Loki, or VictoriaMetrics + VictoriaLogs). Coverage incomplete. SLO discipline patchy.
Stage 3	Self-service primitives	Single pane of glass for most workloads. SLOs defined for tier-1 services. Alert hygiene improving. Tracing emerging.
Stage 4	Internal Developer Platform	Unified metrics, logs, and traces. SLO discipline universal for tier-1 and tier-2. Alert hygiene strong (low alert noise, runbooks linked). On-call dashboards standardised.
Stage 5	Mature	Full observability mesh. SLOs auto-instrumented from platform contracts. Anomaly detection feeds alerting. Cost of observability tracked and optimised. Tenants get scoped views.

Your score (1–5): _____ Why? Evidence?

_____

4. Secrets handling

How are secrets created, stored, distributed, rotated, and audited?

Stage	Name	Description
Stage 1	Pre-platform	Secrets in env files, code, chat messages. No central management. Rotation manual or never.
Stage 2	Shared infrastructure	Vault or similar deployed for some workloads. Rotation aspirational. Audit trail patchy.
Stage 3	Self-service primitives	Central secret store (Vault / External Secrets Operator). Most workloads consume from it. Rotation policy defined. Some auditability.
Stage 4	Internal Developer Platform	All secrets centralised. Rotation enforced (policy-based). Workload identity federated (no embedded credentials). Audit trail complete.
Stage 5	Mature	Workload-identity-only — no static secrets in workloads. Rotation continuous and automated. Customer-controlled keys for sovereign workloads. Periodic chaos-test of rotation.

Your score (1–5): _____ Why? Evidence?

_____

5. Identity model

Are workforce identity (engineers) and workload identity (services) federated to a single trustworthy source, or fragmented across systems?

Stage	Name	Description
Stage 1	Pre-platform	Local accounts everywhere. Shared credentials. Joiner-mover-leaver (JML) handled manually.
Stage 2	Shared infrastructure	SSO / OIDC for some systems. Workload identity ad-hoc (static API keys). JML mostly manual.
Stage 3	Self-service primitives	SSO for engineer access. Workload identity emerging (e.g., service accounts with OIDC, IAM roles for service accounts). JML semi-automated.
Stage 4	Internal Developer Platform	Universal SSO plus MFA for engineers. Workforce-to-workload identity federation (e.g., SPIFFE / SPIRE, OIDC-based). JML fully automated. RBAC reviewed quarterly.
Stage 5	Mature	Zero-trust identity. Per-action authorisation (not just per-system). Customer-side identity federation supported (BYOidp). Continuous attestation.

Your score (1–5): _____ Why? Evidence?

_____

6. Multi-tenancy

How is isolation between teams, products, customers, or environments achieved? At what layer? How robust is the boundary?

Stage	Name	Description
Stage 1	Pre-platform	No formal isolation. "Test" and "prod" share clusters, VLANs, or databases. Cross-tenant noisy-neighbour issues common.
Stage 2	Shared infrastructure	Namespace-per-team in Kubernetes. Network isolation patchy (no NetworkPolicy / Cilium). Resource quotas patchy.
Stage 3	Self-service primitives	Namespace plus quota plus NetworkPolicy. Some tenants get a cluster-per-team for stricter isolation. Identity scoped per tenant.
Stage 4	Internal Developer Platform	Tenant CRD or equivalent — the tenant is a first-class platform concept. Isolation at the network, identity, storage, and observability layers. Quotas and SLOs per tenant.
Stage 5	Mature	Multi-tier tenancy (org → project → workload). Cross-tenant guarantees provable. Customer-isolation-grade (suitable for sovereign / regulated workloads). Tenant onboarding fully automated.

Your score (1–5): _____ Why? Evidence?

_____

7. Disaster-recovery posture

Is backup, restore, and disaster recovery routinely tested and proven, or is it a paper plan?

Stage	Name	Description
Stage 1	Pre-platform	Backup partial and untested. No DR plan, or a plan untested for years. RTO / RPO not documented.
Stage 2	Shared infrastructure	Backup automated for most workloads. Restore tested ad-hoc. DR plan exists but has not been exercised in the past 12 months.
Stage 3	Self-service primitives	Backup and restore tested annually for tier-1 workloads. DR site exists; failover tested partially. RTO / RPO documented.
Stage 4	Internal Developer Platform	Backup, restore, and failover tested quarterly for tier-1 and annually for tier-2. RTO / RPO measured (not just documented). DR-tested results retained for audit.
Stage 5	Mature	Continuous DR (game-day cadence). Tier-1 workloads multi-region active-active or active-warm. Customer isolation respected during DR. Compliance evidence (DORA Art. 25, NIS2 BCP) auto-generated.

Your score (1–5): _____ Why? Evidence?

_____

8. Self-service depth

How quickly can a product engineer go from "I need an environment" to "I have one running my code", without ticket-driven hand-offs?

Stage	Name	Description
Stage 1	Pre-platform	Ticket-driven. Days to weeks. Engineers wait on ops. No standard path.
Stage 2	Shared infrastructure	Some self-service for dev environments. Production still ticket-driven. Standard paths exist informally; documentation patchy.
Stage 3	Self-service primitives	Self-service for non-prod and many prod operations. Golden path documented for common workload types. Ticket queue shrinking.
Stage 4	Internal Developer Platform	Self-service end-to-end (environment, deploy, observability, scaling, rollback). Time-to-environment in minutes. Golden paths first-class — divergence requires a PR plus review.
Stage 5	Mature	Platform-as-product. Engineers describe outcomes (workload plus SLO plus tenancy); the platform realises them. Internal NPS measured. Continuous improvement loop.

Your score (1–5): _____ Why? Evidence?

_____

Your scores

Transfer each dimension score below, then compute the average.

Dimension	Score (1–5)	Stage
1. Workload portability	_____	_____
2. GitOps adoption	_____	_____
3. Observability unification	_____	_____
4. Secrets handling	_____	_____
5. Identity model	_____	_____
6. Multi-tenancy	_____	_____
7. Disaster-recovery posture	_____	_____
8. Self-service depth	_____	_____
Average	**_____**	**_____**

Lowest 2 dimensions: _____

Highest 2 dimensions: _____

> A jagged profile — very high on some dimensions, very low on others — usually indicates uneven investment. Lifting your lowest dimensions typically has higher leverage than improving the ones that are already strong.

What does my average score mean?

Score range	Stage	Interpretation	What to do next
1.0 – 1.5	Stage 1 — Pre-platform	Mostly fragmented infrastructure. Heroics keep things running. Investment in any platform foundation pays back immediately.	Stand up a platform team (3–5 engineers). Adopt IaC plus GitOps for new workloads. Centralise secrets.
1.5 – 2.5	Stage 2 — Shared infrastructure	Core stack exists but is ticket-driven. Engineers wait on ops. A visible bottleneck.	Define golden paths for the top-3 workload types. Move provisioning to self-service. Enforce GitOps.
2.5 – 3.5	Stage 3 — Self-service	Approaching IDP. Self-service partial. Golden paths emerging.	Invest in an IDP product mindset. Treat developers as customers. Measure time-to-environment and internal NPS.
3.5 – 4.5	Stage 4 — IDP	Internal developer platform operational. Most teams self-serve.	Deepen tenancy, DR, and observability. Productize the platform internally. Expand to underserved workload types (data, AI, edge).

Score range	Stage	Interpretation	What to do next
4.5 – 5.0	Stage 5 — Mature	Compounding returns. The platform team is leveraged across the org.	Continuous improvement. Benchmark externally. Consider sharing tooling externally (open-source contribution, community presence).

What should we invest in first?

If your weakest dimension is...

Weakest dimension	Most-leveraged investment
1. Workload portability	Adopt IaC universally; quarantine vendor-locked services; pilot a Kubernetes-based substrate.
2. GitOps adoption	Deploy Argo CD or Flux; move app deploys to a PR-driven flow; layer infra GitOps next.
3. Observability unification	Stand up unified metrics plus logs (e.g., VictoriaMetrics + VictoriaLogs); define SLOs for tier-1; rationalise alerts.
4. Secrets handling	Deploy a central secret store plus External Secrets Operator; eliminate static secrets in workloads.
5. Identity model	Universal SSO plus MFA; workload identity via OIDC / SPIFFE; automate JML.
6. Multi-tenancy	Adopt a tenant-CRD pattern (e.g., Cozystack Tenant CRD); enforce NetworkPolicy / Cilium; quotas per tenant.
7. Disaster-recovery posture	Test backup restore plus failover for tier-1 within 30 days; document RTO / RPO; quarterly cadence after that.
8. Self-service depth	Define the top-3 golden paths; build a portal or CLI wrapper; measure time-to-environment.

Common patterns

> **"Strong on tooling, weak on tenancy / DR" pattern.** Often seen at organisations 2–3 years into platform investment. Tooling fluency is high (GitOps, observability, secrets), but cross-team isolation and tested DR lag behind. Investment in tenancy-as-product unlocks regulated workloads and customer-facing platform offerings.

> **"High self-service, low maturity" pattern.** Engineers can self-serve a lot, but the underlying primitives are fragile. The risk is a sudden plateau when that underlying weakness surfaces. Invest in the lowest 2 dimensions before deepening self-service further.

Next steps

Option 1 — Self-remediate using this assessment. Share the assessment with engineering leadership. Workshop the lowest 2 dimensions. Define a 90-day plan.

Option 2 — Platform Readiness Assessment with Aenix (14 days). We score with you, sanity-check the results, and produce a roadmap for Phase 2 platform engineering work. Fixed fee.

Option 3 — Platform engineering managed engagement. A multi-month build covering architecture, IDP design, golden paths, observability, DR, and tenancy. Scope is set after Phase 1.

> **Schedule a discovery call** → aenix.io/services/platform-readiness-assessment/

Aenix is the team behind Cozystack (CNCF Project) and offers Ænix Platform — our commercial productized offering based on Cozystack. aenix.io

---