Teams' Self Assesment

# 🧩 Heureka Tech Health Framework

**Engineering Self-Assessment v1.0**

---

## 🧭 Purpose & Instructions

The **Tech Health Self-Assessment** gives every team a consistent way to evaluate their technical health across **five key areas**.

- 📅 Run **once per quarter**.
- 👥 Led by **Tech Lead / EM** with 1–2 engineers.
- 🎯 Goal: **reflection & improvement**, not judgment or ranking.
- 🗂️ Scale: **1 = chaotic / ad-hoc**, **4 = optimized & automated**.
- ✍️ Add short comments (“why this score”, “next step”).

---

## ⚙️ Scoring Levels (applies to all axes)

| Level | Description | Observable Example |
|---|---|---|
| **1 – Chaotic** | Ad-hoc, manual, unpredictable. | “Only one person can deploy; tests fail randomly; no docs.” |
| **2 – Emerging** | Partial coverage or informal discipline. | “Basic tests exist; manual review; occasional postmortems.” |
| **3 – Defined** | Stable processes & metrics for critical flows. | “Reliable CI; alerting in place; ADRs used for key changes.” |

| | | |
|---|---|---|
| **4 – Optimized** | Automated, measurable, continuously improved. | “Pipelines self-validate; incidents auto-tracked; SLO-based capacity steering.” |

---

## 🧱 A. TECH DEBT (5 axes)

| Sub-Axis | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| **Code Quality Debt** | No consistent reviews, duplication everywhere. Code inconsistent, high complexity, no shared standards or ownership. | PR reviews exist but no static checks. Manual reviews done inconsistentl y; no linting or static analysis; quality depends on individuals. | Linting & SonarQube in CI; code-owner rules. Consistent PR reviews; automated style and quality checks in CI; ownership defined. | Automated checks + refactoring backlog; clean metrics trend down. Static analysis, code smells tracked; refactoring backlog prioritized; technical debt trend improving (Sonar metrics down). |
| **Architecture / Domain Debt** | Monolith / tight coupling. Code and domains are tangled; any change affects | Some domains defined; unclear API contracts. Partial separation, but | Clear bounded contexts; service boundaries reviewed. Domains explicitly | Architecture validated by Fitness Functions; ADR history maintained. Teams own independent |

| | | | | |
|---|---|---|---|---|
| | multiple areas. No clear ownership. | boundaries fuzzy; APIs leak internal logic; unclear data ownership. | defined; APIs stable and versioned; cross-domain dependencies reviewed regularly. | bounded contexts; minimal coupling; contracts tested automatically and documented. |
| **Infrastructure Debt** | Fully manual process, inconsistent environments. | Some automation exists, but still requires manual steps or supervision | Reliable CI/CD with rollback. | Full infra-as-code; zero-touch deploy; blue-green or canary releases. |
| **Process / Delivery Debt** | Deploys happen anytime, nobody tracks what went out. | Some rules exist (code review, tagging), but people often skip them. | Releases follow a defined flow; everyone knows what's deployed and when. | Deploys fully automated; system tracks DORA metrics automatically. |
| **Knowledge Debt** | Context in people's heads. | README outdated; no ADRs. | Docs & ADRs for major services. | Continuous documentation; onboarding < 1 day. |

---

### 🧪 B. TESTING & AUTOMATION

| Sub-Axis | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| **Unit Testing Coverage** | No meaningful | Basic unit tests on key | Solid coverage of | High-confidence |

| | | | | |
|---|---|---|---|---|
| | tests; manual validation only | logic; not enforced in CI. | critical code; tests run automatically in CI. | unit suite; fast, stable, and used as living spec. |
| **Integration Testing** | No automated integration testing. | Few manual or partial API tests. | Contract tests between main services; checked in CI. | Full API contract validation in pipeline; auto-generated mocks and alerts on contract breaks. |
| **E2E Flow Coverage** | Only manual smoke tests after deploy. | Some automated E2E tests on main flows; run nightly. | Key business flows automated and part of CI. | Critical journeys fully automated; regression E2E suite runs on every deploy. |
| **Pipeline Reliability** | Builds often fail or give false results. | Pipeline sometimes flaky; unclear root cause. | Reliable CI/CD; success > 95 %, flaky < 5 %. | Self-healing pipelines; > 98 % success; issues visible and auto-reported. |
| **Deployment Automation** | Deploys done by hand (SSH, scripts, manual config). No | Basic scripts or CI jobs exist, but require manual input or approvals. Rollback | One-click deploy from CI/CD; rollback tested and documented. No direct | Fully automated delivery pipeline with health checks, canary or |

| | | | | |
|---|---|---|---|---|
| | rollback, high risk of error. | unclear or manual. | server access needed. | blue-green releases, and automatic rollback. |

---

## 🔍 C. OBSERVABILITY & STABILITY

| Sub-Axis | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| **Monitoring Coverage** | No metrics or alerts; rely on user reports. | Basic infra metrics (CPU/RAM); missing ownership. | Key services monitored with clear owners; SLOs for core SLIs. | Full observability stack (SLI/SLO/Error Budget) across services; live dashboards used daily by teams. |
| **Alert Hygiene & On-Call Discipline** | Alerts noisy or ignored; no on-call setup. | Some alerts tuned; on-call rotation informal or inconsistent. | Alerts actionable (>80%); OpsGenie used for ack/close tracking; P1/P2 severity respected. | On-call fully operational; all alerts actionable; MTTA/MTTR auto-tracked; no ignored pages. |
| **Incident Response (MTTR)** | Incidents ad-hoc; no tracking or ownership. | Incidents logged manually; response unstructured. | MTTR measured; Severity levels defined; RCA (Root Cause | MTTR <1h; incident auto-created in OpsGenie; RCA auto-linked; lessons |

| | | | | |
|---|---|---|---|---|
| | | | Analyses)required for P1s. | shared org-wide. |
| **Logging / Tracing** | Scattered logs; no correlation. | Structured logs; tracing partial on key flows. | Centralized logs; tracing active on top paths; searchable incidents. | Full distributed tracing (100%); correlation IDs from edge to DB; logs auto-correlated with incidents. |
| **Postmortems & Learning** | None or blame-based. | Written ad-hoc; no follow-up. | Formal RCAs for major incidents; improvement items tracked. | RCAs for all SEVerity 1–2s; learnings feed Tech Health backlog; trends reviewed in Tech Health Review. |

---

## ⚡ D. DELIVERY PERFORMANCE (DORA)

| Metric | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| **Deployment Frequency (DF)** | < 1 deploy / month. | Weekly deploys. | Daily or per feature branch. | On-demand / continuous delivery. |
| **Lead Time for Changes (LTC)** | 7 days. | 2–7 days. | 1–2 days. | < 24 h from commit to prod. |

| | | | | |
|---|---|---|---|---|
| **Change Failure Rate (CFR)** | 25 %. | 15–25 %. | 5–15 %. | < 5 %. |
| **Mean Time to Recovery (MTTR)** | 6 h. | 2–6 h. | 1–2 h. | < 1 h. |
| **Rollback Frequency** | Frequent rollbacks. | Occasional. | Rare (< 10 %). | Exceptional (< 3 %). |

---

## 🧠 E. GOVERNANCE & KNOWLEDGE

| Sub-Axis | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| **ADR Discipline** | No record of decisions; context lost. | A few ADRs written reactively, not shared. | ADRs created for all significant technical or architectural changes. | ADRs standardized via template; reviewed quarterly; both team-level and global ADRs maintained. |
| **Decision Traceability** | No linkage to delivery work (Jira, Epics). | Partial linkage; some ADRs referenced manually. | Decisions traceable to delivery items in ≥80 % of cases where applicable. | Full traceability: each Epic or initiative links to its ADR; org-wide ADRs referenced in standards and playbooks. |

| | | | | |
|---|---|---|---|---|
| **Architecture Docs** | None or outdated diagrams. | Partial or inconsistent C4 documentation. | Up-to-date diagrams for core domains; shared in team space. | Comprehensive Arc42-style documentation; automated generation from code/config. |
| **Ownership Clarity** | Undefined; “who owns this?” is unclear. | Some ownership lists exist, not maintained. | Clear ownership in Backstage; on-call mapping exists. | Ownership isn’t just written down — it’s proven by operations. The team that has it on the dashboard and gets the alerts owns it. |
| **Transparency** | Context closed within teams; decisions siloed. | Shared internally but not visible org-wide. | Visible across engineering; ADRs and docs accessible to all. | “Public by default” culture: decisions and lessons reviewed quarterly in Tech Health Review. |

---

### 💬 Pulse Survey (Soft Signals – Monthly, 0-10 Scale)

| Question | Purpose |
|---|---|

| | |
|---|---|
| 1️⃣ How much does tech debt slow you down? | Detects friction sources. |
| 2️⃣ How smooth is your release process? | Checks delivery flow confidence. |
| 3️⃣ Do you have time for stability / engineering work? | Validates 25 % allocation reality. |
| 4️⃣ Do you feel leadership supports technical investment? | Measures cultural alignment. |
| 5️⃣ Are your tools / environments effective? | Surfaces DevEx blockers. |

---

## 📈 Interpreting Scores

| Average Score | Meaning | Action |
|---|---|---|
| **1.0 – 1.9** | Unstable, reactive. | Create recovery plan; add to Tech Big Rocks. |
| **2.0 – 2.9** | Emerging discipline. | Prioritize automation & documentation. |
| **3.0 – 3.4** | Stable baseline. | Sustain & optimize critical paths. |
| **3.5 – 4.0** | Optimized, data-driven. | Share practices; help mentor others. |

---

## 🔄 Example Scenarios

- 🧱 **Architecture Debt = Level 2** → legacy monolith; plan for domain decomposition in Q2.
- 🧪 **Testing = Level 3** → solid CI; next step: flaky-test dashboard.
- 🔍 **Observability = Level 1** → no SLOs; urgent stability investment.
- ⚡ **Delivery = Level 4** → fast CI/CD; monitor CFR trend to keep risk low.
- 🧠 **Governance = Level 2** → ADR discipline inconsistent; add template & quarterly check.

---

### 🧩 Outcome

Teams own their improvement roadmap,
leadership sees comparable, trendable data,
and **tech health becomes a conversation, not a surprise**.

---