DevOps Policies & Standards (“How we work”)

Topic Policy (What) Standard (How)

Branching Main branch = production truth Trunk-based; feature branches < 48 h

CI Checks Code must compile & test before merge 80 %+ unit coverage; lint clean

Secrets No plaintext secrets in Git AWS Secrets Manager only; GitHub secret-scan enabled

Infrastructure IaC for everything Terraform in infra/ repos, reviewed via PR

Deployments Fully automated GitHub Actions + environment gates

Rollbacks MTTR target < 30 min One-click “redeploy previous” job

Change Records Every prod change logged Pipeline updates Jira “Change” issue automatically

Tagging Cost & owner tags mandatory owner, team, env, service

DevOps Roles & Culture (“Who owns what”)

Role Key Responsibilities Typical Tools

Product Team (“you build it”) Feature code, unit tests, on-call VS Code, Jest, Cypress

Platform/DevOps CI/CD templates, IaC modules, observability, cost guard-rails GitHub Actions, Terraform, Datadog

SRE (shared) Reliability SLOs, incident commander, PIR facilitation PagerDuty, Datadog, SharePoint PIR

Security Engineer Threat modelling, pipeline scanners, IAM reviews Snyk, Trivy, OPA/Conftest

FinOps Analyst Monthly cost reviews, tagging audits AWS Cost Explorer, Datadog Cost widgets

Cultural Principles

Blameless post-incident reviews (focus on learning).

Continuous feedback – dashboards & alerts visible to everyone.

Automation first – manual steps are tech debt.

Shared ownership – “Code, runtime, cost” belong to the same team.

DevOps Training Path (“What to learn & when”)

Audience Module Format Duration

All engineers DevOps fundamentals (CI/CD, IaC, GitOps) Recorded lunch-and-learn + quiz 2 h

Back-end devs Laravel Vapor deployment deep-dive Hands-on workshop 90 min

Front-end devs Vue + Cypress E2E pipeline Live demo 1 h

Mobile devs React-Native CI/CD with EAS & Detox Doc + screencast 1 h

SRE / DevOps Advanced Terraform & OPA Virtual lab ½ day

Leads / PMs Reading DORA metrics 30-min walk-through 30 min

DevOps Maturity Model & Self-Assessment

Area Crawl Walk Run Fly

CI Manual builds Automated builds Parallel tests + coverage < 5 min feedback on every PR

CD Manual production deploy Scripted deploy Blue/green, canary Fully automated prod with auto-rollback

IaC Infra by hand IaC checked-in PR-driven Terraform Policy-guarded GitOps

Observability Basic logs Dashboards SLO alerts AIOps correlation, self-healing

Security Annual scans Dependency scanning SAST/SCA in CI Real-time policy-as-code gates

Cost Monthly bill review Tagging Cost dashboards Automated rightsizing & anomaly alerts

Self-Assessment Checklist

Pick latest production service (e.g. Vapor API).

Score 0-3 per area (Crawl=0 … Fly=3).

Highest gaps → next quarter OKRs.

Re-score every 6 months; display trend on SharePoint chart.

DevOps Tools & Metrics (“What we measure & with what”)

Category Primary Tool Key Metric Target / SLA

Build GitHub Actions Build time ≤ 5 min

Deploy GitHub Actions Environments Deployment frequency ≥ daily (non-prod)

Runtime Datadog APM P95 latency < 300 ms

Reliability Datadog Synthetics Uptime 99.9 % API

Code Quality Jest / phpunit Coverage ≥ 80 %

Security Snyk / Dependabot Critical vulns open 0

Cost AWS Cost Explorer Untagged resources 0

Culture DORA metrics Lead-time ≤ 24 h

Incidents PagerDuty MTTR < 30 min

🔍 ITIL vs DevOps: Core Differences

Feature ITIL (IT Infrastructure Library) DevOps

Origin Traditional IT Service Management (ITSM) Agile / Lean development + operations

Focus Stability, compliance, and service delivery Speed, automation, and continuous delivery

Approach Process-oriented, structured roles and approvals Culture of collaboration, automation, CI/CD

Cycle Structured service lifecycles (e.g., Incident, Change, Release Management) Short, iterative cycles with fast feedback

Tooling ITSM platforms like ServiceNow, BMC Remedy DevOps toolchain: GitHub Actions, Jenkins, Terraform, etc.

Governance Strong governance and risk management Built-in quality via testing, monitoring, shift-left practices

⸻

✅ Where ITIL can be relevant to DevOps

There are a few intersection points — modern ITIL (v4) tries to adapt to agile and DevOps practices:

• Change Management → DevOps uses automated change approvals (e.g., pipelines), while ITIL traditionally requires manual CABs. ITIL v4 now encourages Change Enablement rather than rigid approvals.

• Incident Management → Can benefit from DevOps observability tools (e.g., Datadog, Sentry, etc.) and on-call rotation automation.

• Service Continuity → Can be enhanced via DevOps infrastructure-as-code and failover automation.

But these are more about integration points, not alignment of philosophies.

⸻

🚫 Where ITIL and DevOps conflict

• ITIL is hierarchical and slow — not well suited to continuous delivery.

• DevOps values autonomy and decentralization, which can clash with ITIL’s prescriptive processes.

• In DevOps, everyone owns the product — ITIL often separates roles too strictly (e.g., change manager, release manager, etc.)

⸻

🧠 Summary

ITIL is not DevOps and was not made for DevOps.

However, some ITIL practices can be modernized or integrated into DevOps workflows (especially in enterprise environments), but they need to be adapted, not blindly applied.

DevOps teams generally don’t follow ITIL unless forced by enterprise IT governance — and even then, it’s usually adapted or minimized.