

Introduction to DevOps Fundamentals

- DevOps unites development and IT operations to deliver value faster and more reliably
- Core promise: shorten time between change and impact without increasing risk
- Built on three reinforcing pillars: culture, automation, and measurement
- Culture defines what to automate; tools shape how culture operates
- Learning journey begins for developers, operations teams, technical leads, and career switchers



Why DevOps Now: From Silos to Flow



- Traditional silo model: developers write code, operations deploys it



- 'Throwing code over the fence' creates bottlenecks and misalignment



- Business drivers: faster time to market, reduced risk, better responsiveness



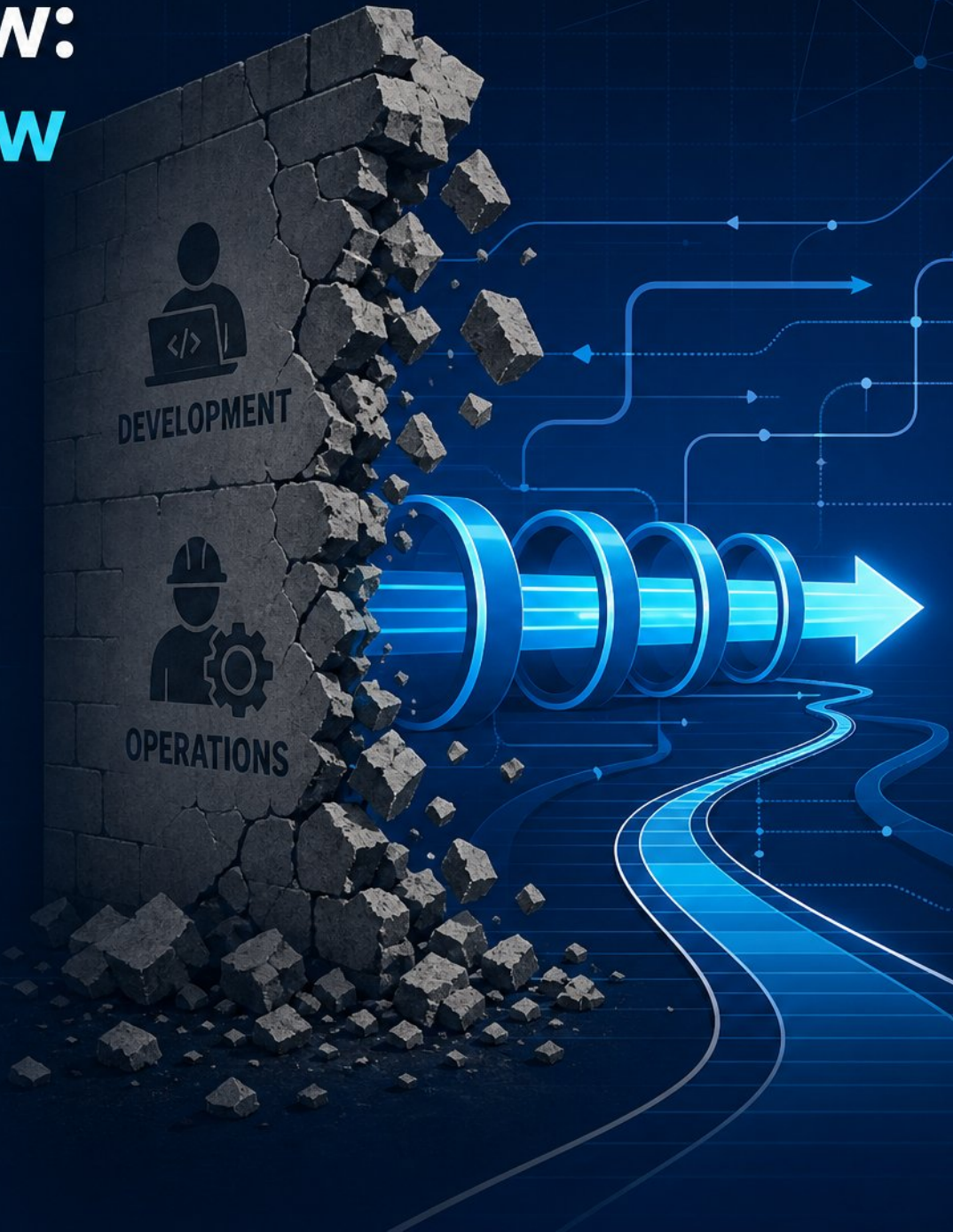
- DevOps extends Agile and Lean through full software delivery lifecycle



- 'You build it, you run it' ownership model improves troubleshooting



- Shared ownership eliminates handoff friction and accelerates flow



The Cultural Foundation: Shared Responsibility



- Shared ownership from build to run breaks down silos



- Blameless postmortems focus on system gaps, not individual fault



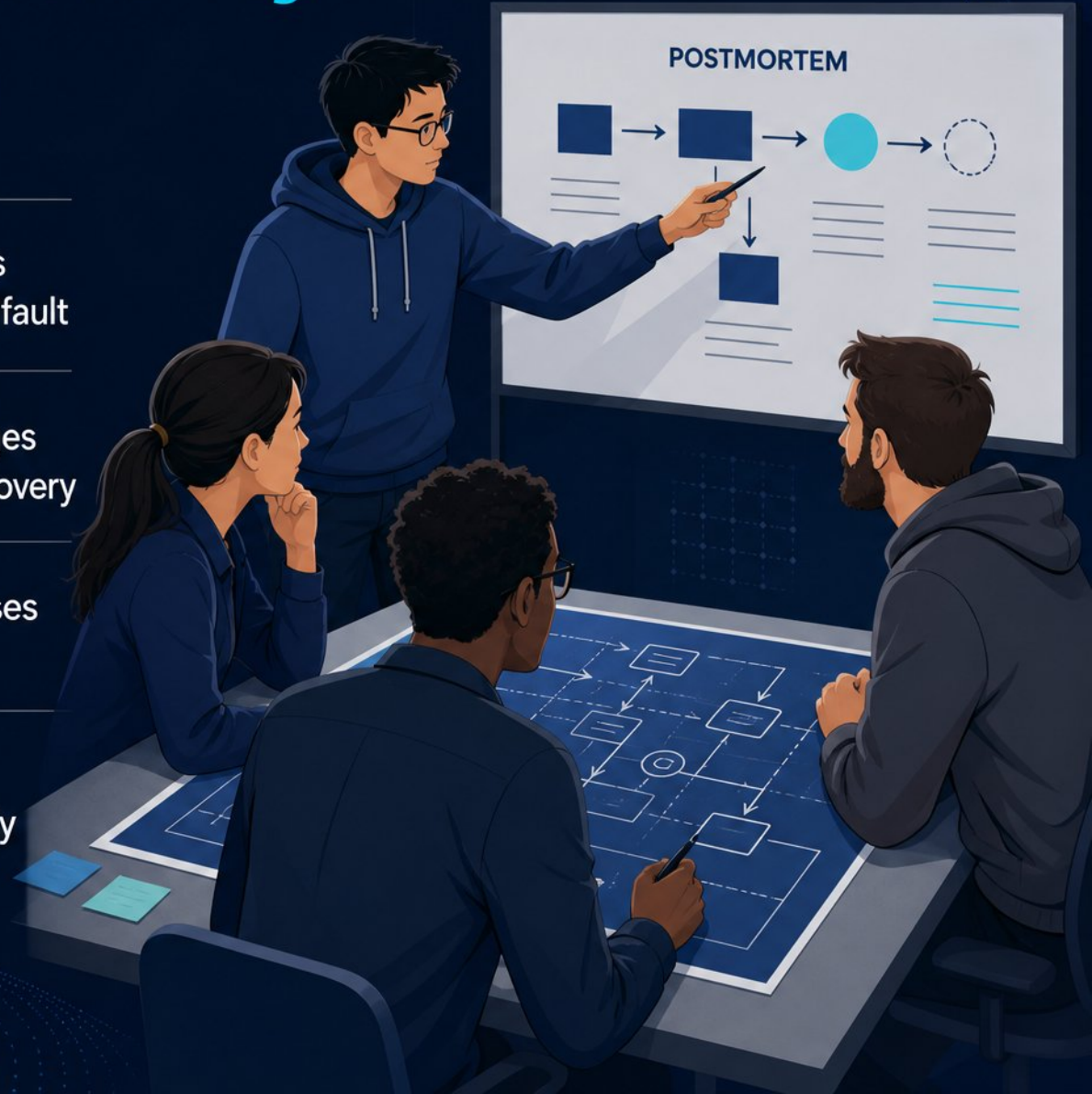
- Psychological safety encourages honest reporting and faster recovery



- Use 5 Whys to trace root causes in processes and tools



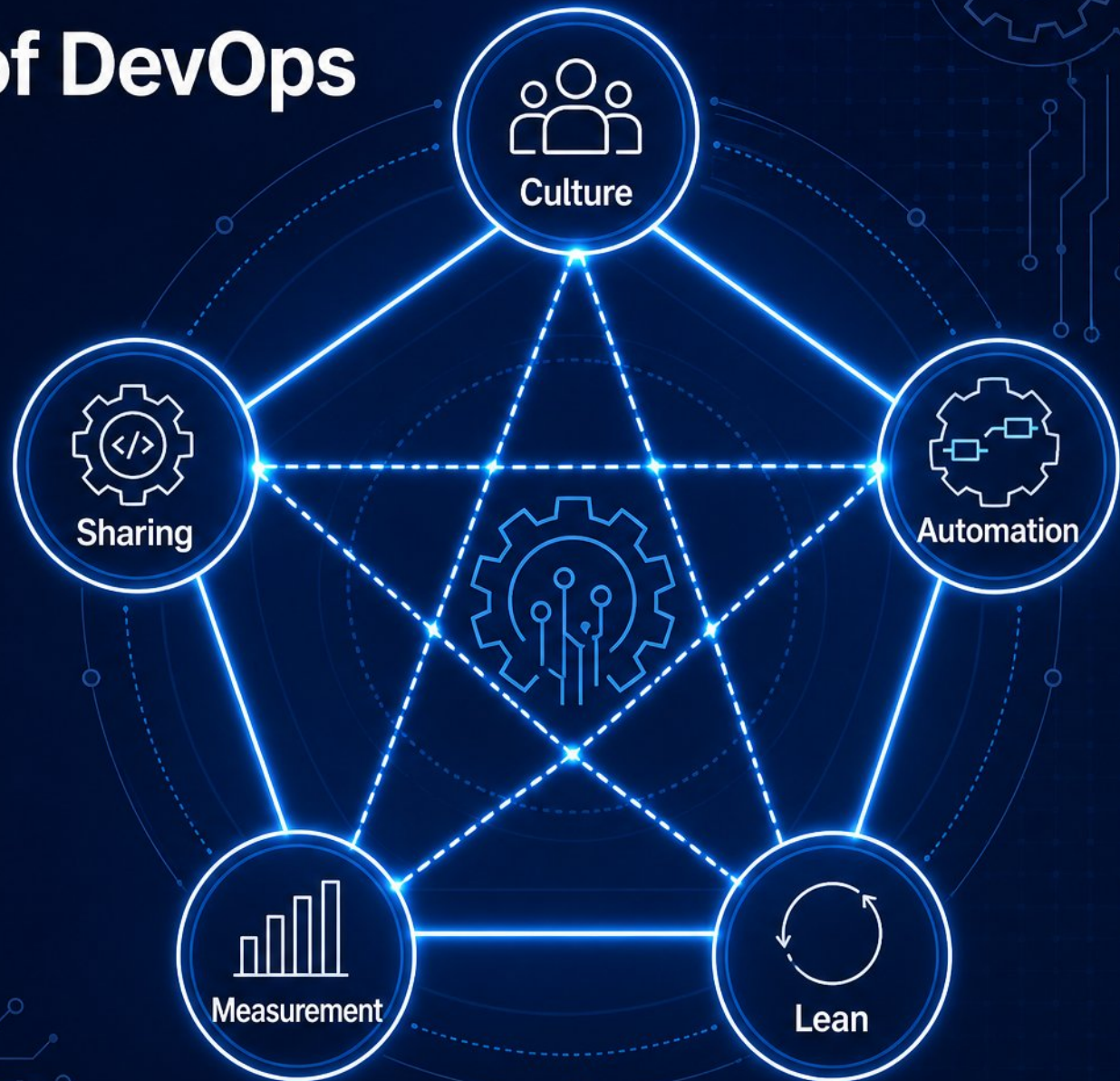
- Leadership must model blameless behavior consistently



CALMS:

The Five Pillars of DevOps

- **Culture:** blameless postmortems and shared ownership
- **Automation:** CI/CD pipelines and infrastructure as code
- **Lean:** small batch sizes and short feedback loops
- **Measurement:** data-driven decisions with DORA metrics
- **Sharing:** open knowledge flow and shared dashboards
- All five pillars must be reinforced together



Core Practices: CI/CD and Infrastructure as Code



- **Continuous Integration:** merge code often, run automated builds and tests



- **Continuous Delivery:** every CI-passing change is prepared for release



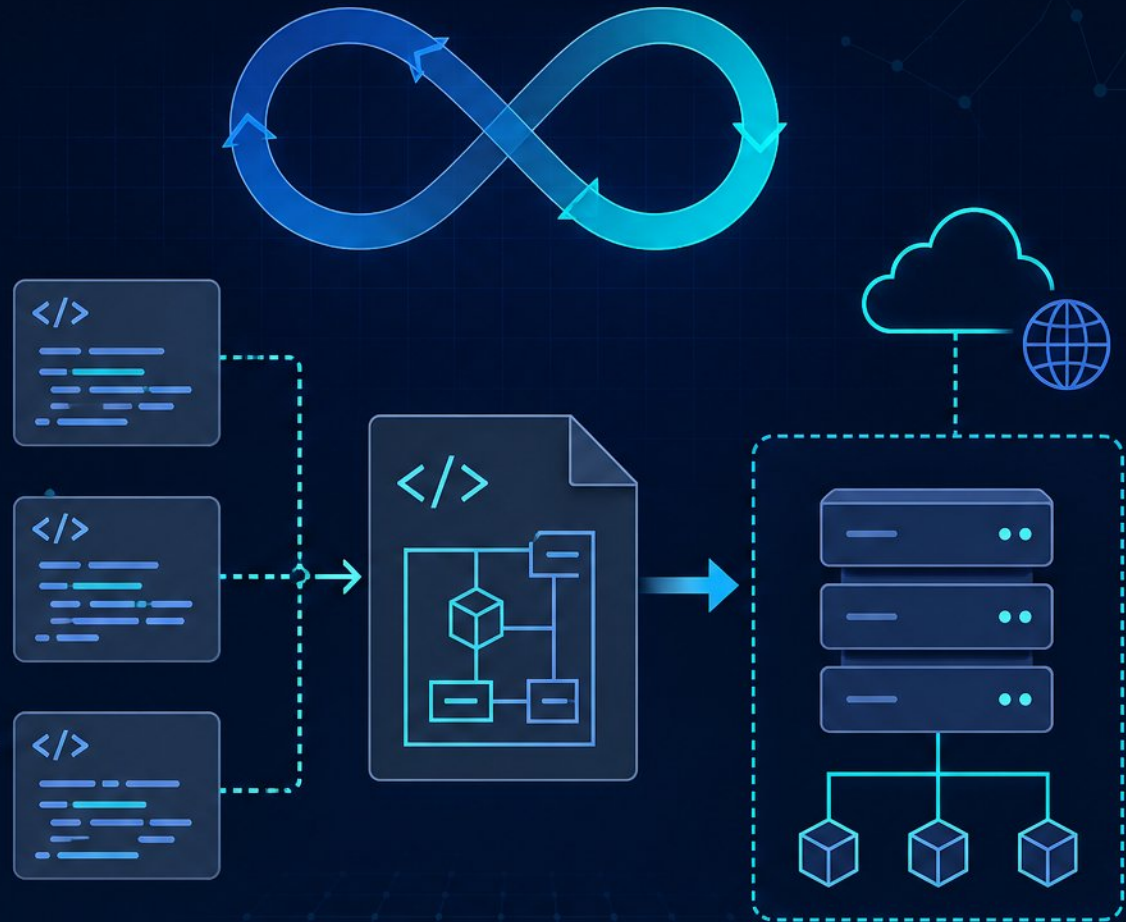
- **Continuous Deployment:** deploy every change to production automatically



- **Infrastructure as Code:** versioned, declarative provisioning of environments



- Removes manual, error-prone work and gives you a repeatable path to production



The Modern Architecture Landscape

MONOLITH



VS

MICROSERVICES



- Microservices enable frequent, independent deployments but increase operational complexity



- Containers package code and dependencies for portability



- Kubernetes is the standard for orchestrating containers at scale



- Observability means understanding system state from external outputs



- Logs, metrics, and traces are the three core signals

The Essential Toolchain: Categories and Selection



Source
Control



CI/CD



Artifact
Management



IaC



Containers



Orchestration



Observability



Security

- Source control, CI/CD, artifact management, IaC, containers, orchestration, observability, security
- Git, GitHub Actions, Jenkins, Terraform, Docker, Kubernetes, Prometheus, Grafana
- Terraform leads IaC; Ansible leads configuration; Kubernetes dominates orchestration
- ArgoCD for GitOps; OpenTelemetry for observability
- Choose tools for your needs to avoid sprawl; toolchain reflects culture

Automating Build and Test: The CI Pipeline



- Commit triggers build, unit tests, static analysis, artifact creation
- Layer tests: unit, integration, API contract, then end-to-end
- Quality gates block on coverage, critical bugs, and vulnerabilities
- Fast feedback under 10 minutes prevents context switching
- Flaky tests erode trust: quarantine and fix them immediately

Deployment Strategies and Continuous Delivery

- - Continuous delivery requires manual approval before release; continuous deployment is fully automated
- - Blue-green: instant switch and rollback, but doubles infrastructure cost
- - Canary: gradual traffic shift to limit blast radius, requires strong observability
- - Argo Rollouts automates metric-driven canary promotion and rollback on Kubernetes
- - Use backward-compatible migrations and feature flags to decouple deploy from release

DEPLOYMENT STRATEGY TIMELINE



Infrastructure as Code in Practice: Terraform



- Declarative definitions reduce infrastructure drift and technical debt



- Providers identify platforms; resources specify desired components



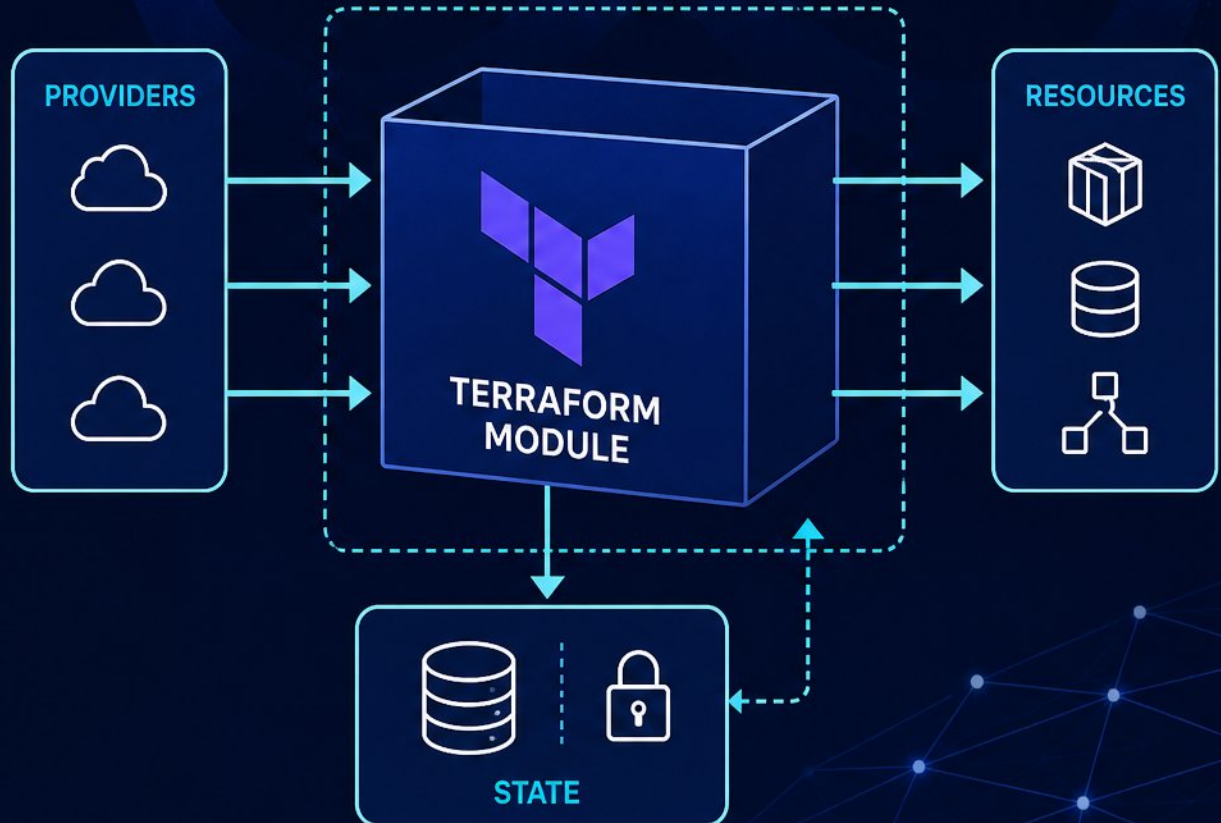
- State must be stored remotely with locking for team safety



- Pin module versions and isolate environments by directory

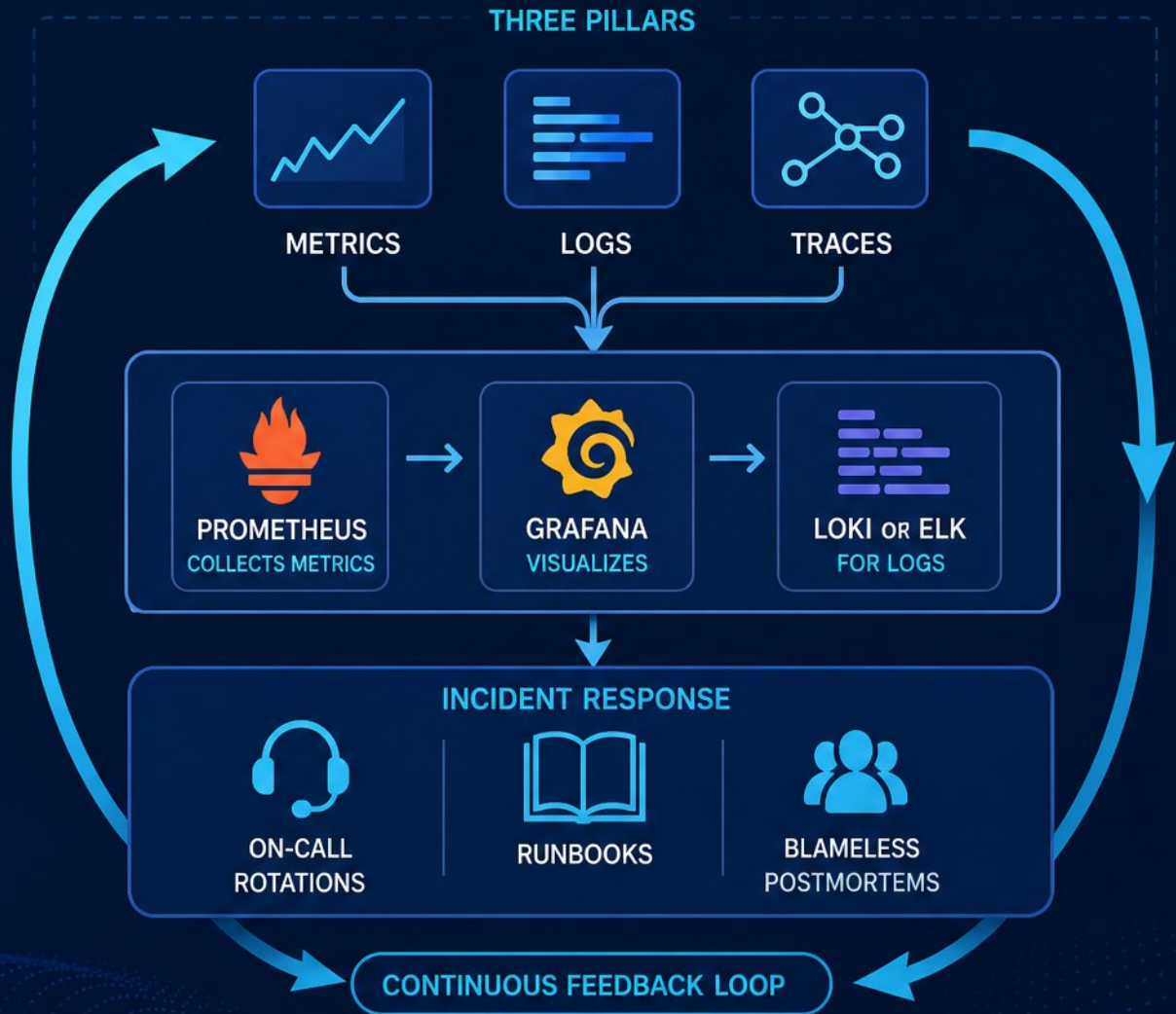


- Use CI plan/apply workflows; never hardcode secrets in state



Observability: Metrics, Logs, and Incident Response

- Logs, metrics, and traces are the three pillars
- Prometheus collects metrics; Grafana visualizes; Loki or ELK for logs
- RED for services; USE for resources; SLO-based alerts cut noise
- On-call rotations, runbooks, and blameless postmortems drive recovery



Security in the DevOps Lifecycle: DevSecOps



- Shift left to catch vulnerabilities when cheapest to fix
- Four scan types: SAST, SCA, IaC, and DAST
- SAST and SCA on every pull request
- IaC scans catch misconfigurations before provisioning
- DAST runs async against deployed builds
- Use vaults, short-lived credentials, and secret detection
- Block only on reachable, exploitable, high-severity findings

Measuring Success: DORA Metrics and Beyond



- Four DORA metrics: deployment frequency, lead time, change failure rate, recovery time



- 2024 benchmark tiers: Elite, High, Medium, and Low performance



- Elite: on-demand deploys, sub-one-day lead time, under 15% failure rate



- Use metrics as a diagnostic thermometer, not a target



- Deployment rework rate added as fifth stability metric in 2024

2024 BENCHMARK TIERS

	ELITE	HIGH	MEDIUM	LOW
 Deployment frequency				
 Lead time				
 Change failure rate				
 Recovery time				

Continuous Improvement and Your Next Steps



- **DevOps:** a continuous improvement journey, not a destination



- **Developers:** adopt CI/CD and small, frequent releases



- **Operations:** adopt Infrastructure as Code and automation



- **Leads:** drive blameless culture and shared ownership



- **Measure:** instrument DORA metrics as a diagnostic



- **Learn:** DORA research and practitioner communities



PersonWise.

PersonWise • AI digital-human course platform



Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/28e317a2/public