DevOps Process and Concept Training

Curriculum 1 Days

Day - 1

Kick-off: Understanding DevOps

- What is DevOps? A philosophy, not a product; flow + feedback + continual learning
- Vision and direction of DevOps for modern software delivery
- Myths & stereotypes (e.g., “DevOps = one team” or “only automation”) and what DevOps actually is

Consumption & Delivery Context (Past → Present)

- Software consumption trends (past): packaged releases, on-prem, long cycles
- Software development model (past): waterfall, heavy handoffs, big-bang releases
- Software consumption trends (present): SaaS, mobile, continuous change, always-on
- Software development model (present): Agile, iterative, product thinking, platform enablement
- Why this matters: faster expectations, higher complexity, constant evolution.

Challenges & Organizational Pain

- Common challenges: siloed teams, long lead times, fragile releases, “works on my machine”
- Large-org challenges: compliance/audit burden, multiple platforms, Conway's Law, legacy estates
- Integration complexity: multi-service meshes, cross-team dependencies, env drift, versioning
- Identifying waste: rework, queues, WIP thrash, manual gates, duplicated infra, context switching

The Problem DevOps Solves

- From project to product: sustained value streams, outcomes over outputs
- Flow and stability together: shorten lead time while improving reliability
- Organization characteristics: customer-centric, platform-enabled, autonomous teams with guardrails

DevOps Principles & Operating Model

- CALMS: Culture · Automation · Lean · Measurement · Sharing
- The Three Ways: Flow → Feedback → Continuous learning & experimentation
- Unifying multiple platforms: golden paths, shared services, internal developer platforms (IDP)
  Automated builds & repeatable delivery: pipeline as code, immutable artifacts, policy as code

Benefits You Can Explain Upward

- Business benefits: faster time-to-market, lower change failure rate, improved MTTR, predictable delivery
- Non-functional benefits: reliability, security, scalability, cost transparency, auditability/compliance

DevOps Automation — Tooling Landscape (Conceptual)

- Collaboration tools (knowledge sharing, ChatOps)
- Planning tools (roadmaps, backlog, product discovery)
- Issue tracking tools (work visibility, flow metrics)
- Source control tools (branching, trunk-based, code review)
- Dev environment tools (standardized, reproducible, containerized dev)
- Build automation tools (compile, test, package consistently)
- Packaging / artifact / repository management (versioned, immutable, SBOM)
- Continuous Integration tools (fast feedback, quality gates)
- Continuous Delivery / release tools (promotion, approvals, rollout strategies)
- Configuration & infrastructure management (idempotent config, IaC, drift detection)
- Monitoring & observability tools (logs/metrics/traces, SLI/SLO dashboards)
- Repository management (code + artifact lifecycle, provenance, retention)

Core Practices & “How”

- Agile development: small batches, product increments, empirical learning
- Test-Driven Development (TDD) & automation: shift-left quality, confidence to change
- Continuous Integration: merge daily, fast tests, fix-forward culture
- Continuous Delivery: deployable at any time, environment parity, quality gates
- Continuous Deployment (when appropriate): progressive delivery, feature flags

Microservices & the Container World (Conceptual)

- Why microservices? Why containers? Independent deployability, scaling, team autonomy
- Trade-offs: operational complexity; security, observability, and resilience as cross-cutting concerns
- Platform patterns: service contracts, API versioning, golden paths, paved roads

Making a DevOps Transition (Pragmatic Playbook)

- Why change? Tie to customer impact and business outcomes
- Change culture: psychological safety, blameless postmortems, shared ownership
- Change organization: Team Topologies (stream-aligned, platform, enabling, complicated-subsystem)
- Address objections: risk, compliance, separation of duties—show audit trails & policy as code
- Quick wins: trunk-based pilot, value-stream mapping, first quality gate, CI baseline
- DORA metrics (measure what matters): deploy frequency, lead time, change fail rate, MTTR
- Roadmap: from ad-hoc → repeatable → reliable → scalable → optimized

Integration & Complexity Management

- Architect for change: contract testing, backward compatibility, canary/blue-green
- Dependency & versioning strategy: mono vs multi-repo, artifact versioning, SBOM
- Legacy modernization: strangler patterns, progressive decomposition, integration adapters

Wrap-Up & Consolidation

- DevOps checklist (culture, automation, lean, measurement, sharing)
- Self-assessment & next steps: pick three improvements for the next 90 days
- Glossary (DevOps, SLI/SLO/SLA, CI/CD, trunk-based, feature flags, golden path, IDP)

Wrap-Up & Consolidation

- DevOps checklist (culture, automation, lean, measurement, sharing)
- Self-assessment & next steps: pick three improvements for the next 90 days
- Glossary (DevOps, SLI/SLO/SLA, CI/CD, trunk-based, feature flags, golden path, IDP)