Admissions & Support:
support@mockattempt.com
Phone / WhatsApp: +91 95400 07917
Course Code: DEVO • **Updated:** 05 Nov 2026
https://mockattempt.com/courses/devops-training

DEVO • DEVOPS CI/CD, GIT, TERRAFORM IAC & SRE | **BEGINNER TO PRINCIPAL DEVOPS & PLATFORM ENGINEER**

100% LIVE INDUSTRY EXPERT COHORT + INTERNSHIP LETTER

DevOps & Git CI/CD Training

Master DevOps & Platform Engineering from DORA Metrics, SRE Error Budgets, Git Content-Addressable DAGs & Strict Bash Automation to Jenkins Declarative Pipelines, GitHub Actions OIDC, Idempotent Ansible, Terraform Remote State/HCL & AWS/Azure DevOps — with 200 Mock Tests & Voice MockAttempt AI Interviews at ₹4,299 (Incl. 18% GST).

Duration: 16–18 Weeks • Live + Hands-On Projects (210 Hrs) • **Modules:** 16 • **Live Topics:** 42 • **Subtopics:** 336+ • **Labs & Tasks:** 84 • **Mocks / AI Vivas:** 244 / 244 • **Projects:** 16 Module + 1 Live Client Prod

All-Inclusive Direct Fee (Incl. 18% GST • Zero Hidden Charges): Total Payable: ₹4,299 (Net Tuition ₹3,643 + Included 18% GST ₹656) • **Split Option:** 2 × ₹2,150 Split • **Effective ROI:** ~₹20/Guided Hr (~₹102/Live Topic)

Faculty: 100% Active Working IT Professionals • **Live Batch Tracks (7:00 AM – 8:00 PM IST Flexible Slots):** Weekday Live Online Batch (Mon – Fri • 1 Hr/Day • 7:00 AM – 8:00 PM) (Monday to Friday (1 Hour Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 1-Hour Slot Based on Trainer Availability) • 18–20 Weeks Fast-Track) | Weekend Live Online Batch (Sat – Sun • 2 Hrs/Day • 7:00 AM – 8:00 PM) (Saturday & Sunday (2 Hours Live Session / Day) • 07:00 AM – 08:00 PM IST (Students Can Choose Any 2-Hour Slot Based on Trainer Availability) • 20–22 Weeks Weekend Track)

Core Technologies & Engineering Stack: Git Internals, Trunk-Based & GitFlow • Strict Bash Scripting & Linux Automation • GitHub Actions OIDC & Jenkins Pipelines • Ansible Playbooks & Molecule Roles • Terraform 1.9 HCL & S3 Remote State • AWS DevOps (OIDC, CodePipeline, ECS/EKS) • Azure DevOps YAML & SRE DORA Metrics

1. PROGRAM MASTERY OUTCOMES & ELIGIBILITY

End-to-End Engineering Outcomes:

1. Measure and optimize elite DORA engineering velocity (Deployment Frequency, Lead Time, Change Failure Rate, MTTR) alongside SRE SLO Error Budgets
2. Navigate Git DAG internals (`blob`, `tree`, `commit`, `reflog`, `bisect`, `rebase`, `stash`) and author defensive `set -euo pipefail` Bash automation
3. Build secretless OIDC CI/CD pipelines across Jenkins, GitHub Actions, AWS, and Azure DevOps, plus idempotent Ansible roles and state-locked Terraform HCL modules
4. Clear 244 Timed Mock Exams (10,000 DevOps, Git & Terraform MCQs) and 200 Spoken Voice + MockAttempt AI Platform Engineering Interviews
5. Linux server administration, Bash automation, systemd services, SSH hardening, DNS/TLS networking, and Nginx reverse proxies.
6. Automated CI/CD pipelines (GitHub Actions) with build caching, automated test gates, SAST/container scans, and environments.
7. Declarative Infrastructure as Code with Terraform modules, remote S3/DynamoDB state locking, and drift detection.
8. Full-stack observability with Prometheus metrics, Grafana dashboards, centralized logs, and incident runbooks.

Eligibility & Prerequisites: Basic familiarity with Linux terminal navigation and editing text files — Git, Bash, CI/CD, and Terraform start from scratch • Any Windows (WSL 2 / Git Bash), macOS, or Linux laptop/PC with Git, Bash, OpenSSH, and Terraform CLI installed • No prior DevOps & Git CI/CD industry experience required — Module 1 starts from environment setup, core mental models, and plain-English fundamentals. • Ideal for B.Tech / BCA / MCA / B.Sc IT students, non-CS beginners, career switchers, and working developers upgrading to 2026 production standards. • Commitment to attend live Weekday or Weekend classes, complete daily hands-on lab checklists, and practice topic-wise Mock Tests & AI Interviews.

2. DETAILED ZERO-TO-EXPERT MODULE, TOPIC & SUBTOPIC SYLLABUS (16 MODULES • 42 LIVE TOPICS • 336+ SUBTOPICS)

Progressive Zero-to-Expert Mastery Stages: Modules 1–4: Stage 1: Day-1 Foundations & Core Mental Models (8 Topics / 64 Subtopics) • Modules 5–8: Stage 2: Applied Engineering, Data & API Workflows (8 Topics / 64 Subtopics) • Modules 9–12: Stage 3: Internals, Concurrency, Security & Performance (12 Topics / 96 Subtopics) • Modules 13–16: Stage 4: Production Architecture, Testing, CI/CD & Expert Mastery (14 Topics / 112 Subtopics)

MODULE 1 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 1: DevOps Culture (`CALMS`), DORA Elite Metrics & Immutable Cloud-Native Architecture

Understand how DevOps bridges development agility and operational stability using `CALMS`, the 4 DORA velocity/stability metrics, and immutable pets-vs-cattle delivery architectures.

Objectives: ✓ DevOps Foundations — `CALMS` Principles, Wall of Confusion & The 4 DORA Engineering Metrics — Why DevOps is an organizational engineering discipline (not just a job title or YAML tool!): breaking the Dev ("ship features fast") vs Ops ("never touch production") incentive conflict via `CALMS`, blameless postmortems, and Google DORA's 4 empirically validated software delivery metrics. • ✓ DevOps Architecture — Immutable Infrastructure, Pets vs Cattle & Build-Once-Deploy-Anywhere — End-to-end delivery topology: Developer Git Push -> Ephemeral CI Runner -> Static/SAST/SCA Scan -> Single Immutable Signed Artifact (OCI Image) -> Artifact Registry -> Progressive CD (`Dev -> Staging -> Canary -> Prod`) -> Unified Observability feedback loop. • ✓ Build, verify, and ship Module Capstone: DORA Engineering Telemetry & SRE Error-Budget Gate Evaluator.

1.1 DevOps Foundations — `CALMS` Principles, Wall of Confusion & The 4 DORA Engineering Metrics [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: CALMS organizational framework & DORA 4 delivery & stability metrics • Core Syntax, CLI & Production APIs: DORA Metrics (DF, LT, CFR, MTTR), CALMS Framework, SRE SLI / SLO / Error Budget • Deep-Dive Mechanics: Small batch size lowers blast radius & SLI (metric) vs SLO (target) vs SLA • Advanced Patterns: Error Budget release governance & Blameless postmortem action items • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: SRE `99.9%` Monthly Error-Budget & DORA CFR Gate in Strict Bash • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for DevOps Foundations

Purpose: To understand and apply DevOps Foundations — `CALMS` Principles, Wall of Confusion & The 4 DORA Engineering Metrics in real DevOps & Git CI/CD systems — specifically: Why...

What Topic Is Next → Next Topic 1.2: DevOps Architecture — Immutable Infrastructure, Pets vs Cattle & Build-Once-Deploy-Anywhere

1.2 DevOps Architecture — Immutable Infrastructure, Pets vs Cattle & Build-Once-Deploy-Anywhere [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Pets (mutable) vs Cattle (immutable) & Build Once Deploy Anywhere SHA-256 • Core Syntax, CLI & Production APIs: Immutable OCI Artifacts, 12-Factor Config Injection, Ephemeral CI Runners, Canary / Blue-Gre... • Deep-Dive Mechanics: 12-Factor runtime config separation & Ephemeral clean-room CI runners • Advanced Patterns: Blue-Green vs Canary traffic shifting & Closed-loop telemetry feedback • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Verifying Identical Artifact SHA-256 Digest Promotion Across Environments • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for DevOps Architecture

Purpose: To understand and apply DevOps Architecture — Immutable Infrastructure, Pets vs Cattle & Build-Once-Deploy-Anywhere in real DevOps & Git CI/CD systems — specifically: End...

What Topic Is Next → Module 1 Capstone (DORA Engineering Telemetry & SRE Error-Budget Gate Evaluator) → Next Topic 2.1: The Infinite DevOps Lifecycle — Continuous Integration vs Continuous Delivery vs Continuous Deployment

Module 1 Hands-On Capstone Project (100 Marks): DORA Engineering Telemetry & SRE Error-Budget Gate Evaluator — Build an automated Bash/jq release-readiness evaluator that calculates monthly SRE 99.9% SLO error budget burn and DORA Change Failure Rate before approving production auto-promotion.

MODULE 2 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 2: The Continuous DevOps Lifecycle (`7 Cs`) & Shift-Left DevSecOps Principles

Connect all phases of the infinite DevOps loop (`Plan -> Code -> Build -> Test -> Release -> Deploy -> Operate -> Monitor`) and apply Gene Kim's Three Ways and Shift-Left DevSecOps.

Objectives: ✓ The Infinite DevOps Lifecycle — Continuous Integration vs Continuous Delivery vs Continuous Deployment — Tracing the 8 phases (`Plan`, `Code`, `Build`, `Test`, `Release`, `Deploy`, `Operate`, `Monitor`) and mastering the critical distinction: **Continuous Delivery** guarantees every `main` build is *production-ready* behind a one-click business gate, whereas **Continuous Deployment** pushes every green `main` commit straight to production automatically!. • ✓ DevOps Workflow & Principles — The Three Ways, Value-Stream Mapping, WIP Limits & Shift-Left Security — Gene Kim's **Three Ways**: (1) Systems Thinking & left-to-right flow optimization (eliminating handoff queues & limiting Work-in-Progress `WIP`), (2) Amplify Feedback Loops right-to-left (Andon cord, actionable alerts), and (3) Culture of Continual Experimentation (GameDays, Chaos Engineering, DevSecOps). • ✓ Build, verify, and ship Module Capstone: Shift-Left Pre-Merge DevSecOps Gate (Secret Scan + Lockfile Audit + Smoke Check).

2.1 The Infinite DevOps Lifecycle — Continuous Integration vs Continuous Delivery vs Continuous Deployment [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: 8-stage infinite feedback loop & Daily mainline Continuous Integration • Core Syntax, CLI & Production APIs: CI (Automated Build + Test) -> Continuous Delivery (1-Click Gate) vs Continuous Deployment (... • Deep-Dive Mechanics: Continuous Delivery (gated prod) & Continuous Deployment (auto prod) • Advanced Patterns: Feature Flags decouple deploy & release & Shift-Left defect cost reduction • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Feature-Flag Decoupled Deploy vs Release Evaluator in Bash • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for The Infinite DevOps Lifecycle

Purpose: To understand and apply The Infinite DevOps Lifecycle — Continuous Integration vs Continuous Delivery vs Continuous Deployment in real DevOps & Git CI/CD systems — specif...

What Topic Is Next → Next Topic 2.2: DevOps Workflow & Principles — The Three Ways, Value-Stream Mapping, WIP Limits & Shift-Left Security

2.2 DevOps Workflow & Principles — The Three Ways, Value-Stream Mapping, WIP Limits & Shift-Left Security [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: First Way: Systems Flow & WIP limits & Second Way: Fast Feedback & Andon Cord • Core Syntax, CLI & Production APIs: Value-Stream Lead Time vs Process Time, Little's Law (LeadTime = WIP / Throughput), Shift-Le... • Deep-Dive Mechanics: Third Way: GameDays & Chaos resilience & Little's Law (LeadTime = WIP / Thruput) • Advanced Patterns: Theory of Constraints bottleneck focus & Shift-Left DevSecOps (SAST/SCA/Secrets) • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Shift-Left Secret Pattern Guard Blocking Leaked AWS Keys • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-

green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for DevOps Workflow

Purpose: To understand and apply DevOps Workflow & Principles — The Three Ways, Value-Stream Mapping, WIP Limits & Shift-Left Security in real DevOps & Git CI/CD systems — specifi...

What Topic Is Next → Module 2 Capstone (Shift-Left Pre-Merge DevSecOps Gate (Secret Scan + Lockfile Audit + Smoke Check)) → Next Topic 3.1: CI/CD Pipeline Methodology — Fail-Fast Stage Ordering, Fan-Out/Fan-In DAGs & The Test Pyramid

Module 2 Hands-On Capstone Project (100 Marks): Shift-Left Pre-Merge DevSecOps Gate (Secret Scan + Lockfile Audit + Smoke Check) — Create a fast `<30s` pre-merge CI gate script that blocks leaked AWS/API credentials, verifies deterministic lockfiles, and enforces Continuous Delivery vs Continuous Deployment promotion ru...

MODULE 3 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 3: CI/CD Pipeline Architecture, Testing Pyramids & The Cloud-Native DevOps Toolchain

Design fast fail-early CI/CD DAG pipelines following the Test Pyramid, and map the CNCF/DevOps landscape cleanly across SCM, CI, IaC, Configuration Management, Containers, and Observability.

Objectives: ✓ CI/CD Pipeline Methodology — Fail-Fast Stage Ordering, Fan-Out/Fan-In DAGs & The Test Pyramid — Structuring deterministic pipelines: fast static checks first (`<30s` lint, typecheck, secret scan), parallel unit/integration fan-out, hermetic container builds, SBOM + vulnerability gating, and progressive Canary/Blue-Green rollouts backed by automated rollback SLOs. • ✓ The DevOps Toolchain Matrix — Terraform vs Ansible vs Docker vs Kubernetes vs Prometheus — Understanding exact boundaries across the modern platform stack so tools never overlap awkwardly: **Git** (source of truth), **GitHub Actions/Jenkins** (CI orchestration), **Terraform/OpenTofu** (declarative cloud resource provisioning), **Ansible** (procedural/declarative OS config), **Docker** (OCI packaging), **Kubernetes + ArgoCD** (runtime orchestration), and **Prometheus/Grafana/OTel** (observability). • ✓ Build, verify, and ship Module Capstone: Fail-Fast Multi-Stage GitHub Actions OIDC CI/CD Pipeline Blueprint.

3.1 CI/CD Pipeline Methodology — Fail-Fast Stage Ordering, Fan-Out/Fan-In DAGs & The Test Pyramid

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Fail-Fast stage ordering (<30s lint) & Fan-Out / Fan-In parallel DAG • Core Syntax, CLI & Production APIs: Fan-Out / Fan-In DAG, Test Pyramid (70/20/10), Concurrency Cancel-In-Progress, Hermetic Buil... • Deep-Dive Mechanics: Test Pyramid vs Ice-Cream Cone trap & Flaky test compound probability math • Advanced Patterns: Hermetic reproducible builds & Automated SLO-triggered rollback • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: GitHub Actions Fail-Fast Parallel DAG with OIDC Least-Privilege • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for CI/CD Pipeline Methodology

Purpose: To understand and apply CI/CD Pipeline Methodology — Fail-Fast Stage Ordering, Fan-Out/Fan-In DAGs & The Test Pyramid in real DevOps & Git CI/CD systems — specifically: S...

What Topic Is Next → Next Topic 3.2: The DevOps Toolchain Matrix — Terraform vs Ansible vs Docker vs Kubernetes vs Prometheus

3.2 The DevOps Toolchain Matrix — Terraform vs Ansible vs Docker vs Kubernetes vs Prometheus

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Terraform (stateful cloud provisioning) & Ansible (agentless SSH OS config) • Core Syntax, CLI & Production APIs: Terraform (Cloud IaC) + Ansible (OS Config) + Docker (OCI Image) + K8s/ArgoCD (Orchestration... • Deep-Dive Mechanics: Avoid Terraform remote-exec anti-pattern & Packer golden AMI / OCI pre-baking • Advanced Patterns: ArgoCD GitOps workload reconciliation & OpenTelemetry + Prometheus + Grafana • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: DevOps Toolchain Responsibility Verification Script • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for The DevOps Toolchain Matrix

Purpose: To understand and apply The DevOps Toolchain Matrix — Terraform vs Ansible vs Docker vs Kubernetes vs Prometheus in real DevOps & Git CI/CD systems — specifically: Unders...

What Topic Is Next → Module 3 Capstone (Fail-Fast Multi-Stage GitHub Actions OIDC CI/CD Pipeline Blueprint) → Next Topic 4.1: Git Internals — Snapshots (Not Diffs!), `.git/objects` (`blob`, `tree`, `commit`) & Branch Pointers

Module 3 Hands-On Capstone Project (100 Marks): Fail-Fast Multi-Stage GitHub Actions OIDC CI/CD Pipeline Blueprint — Author a production GitHub Actions CI/CD workflow YAML using parallel fan-out lint/test jobs, dependency caching, and short-lived OIDC cloud authentication (zero long-lived AWS keys).

MODULE 4 OF 16 • STAGE 1 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 4: Git Content-Addressable Internals (`.git/objects` DAG) & Branching Workflows (`GitFlow` vs Trunk-Based)

Understand Git from first principles as an immutable Merkle-tree DAG (`blob`, `tree`, `commit`, `tag` + lightweight pointer branches), and compare GitFlow against modern Trunk-Based Development.

Objectives: ✓ Git Internals — Snapshots (Not Diffs!), `.git/objects` (`blob`, `tree`, `commit`) & Branch Pointers — Why Git is a content-addressable filesystem: every commit stores a full snapshot reference (`tree`) deduplicated via SHA hashes (`blob`s store pure file bytes without filenames; `tree`s map filenames & permissions to `blob`s/`tree`s; `commit`s point to root `tree` + parent `commit`s), and a branch is simply a **41-byte file containing a commit hash**!. • ✓ Git Branching Strategies — `GitFlow` (`main`/`develop`/`release`/`hotfix`) vs Trunk-Based Development — Comparing Vincent Driessen's classic **GitFlow** (`main`, `develop`, `feature/*`, `release/*`, `hotfix/*` — suited for scheduled versioned firmware/on-prem releases) against **Trunk-Based Development** (`main` + `<24h` short-lived branches + Feature Flags — the DORA Elite standard for continuous SaaS CI/CD). • ✓ Build, verify, and ship Module Capstone: Automated Branch Protection, Conventional Commit Hook & Release Tagging Workflow.

4.1 Git Internals — Snapshots (Not Diffs!), `.git/objects` (`blob`, `tree`, `commit`) & Branch Pointers

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Snapshots (Merkle DAG) not delta diffs & Blob (bytes) vs tree (dir) vs commit • Core Syntax, CLI & Production APIs: git init, git hash-object, git cat-file -p, HEAD -> refs/heads/main, Working Tree -> Index -... • Deep-Dive Mechanics: 41-byte refs/heads/<branch> pointer & Working Tree -> Staging Index -> HEAD • Advanced Patterns: Deleted files stay in .git history! & Git filter-repo + immediate key rotation • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Inspecting Git Content-Addressable `blob` & `commit` DAG Objects • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Git Internals

Purpose: To understand and apply Git Internals — Snapshots (Not Diffs!), `.git/objects` (`blob`, `tree`, `commit`) & Branch Pointers in real DevOps & Git CI/CD systems — specifica...

What Topic Is Next → Next Topic 4.2: Git Branching Strategies — `GitFlow` (`main`/`develop`/`release`/`hotfix`) vs Trunk-Based Development

4.2 Git Branching Strategies — `GitFlow` (`main`/`develop`/`release`/`hotfix`) vs Trunk-Based Development

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: GitFlow develop/release/hotfix branches & Trunk-Based Development (<24h branches) • Core Syntax, CLI & Production APIs: Trunk-Based Development (short-lived PRs -> main) vs GitFlow (develop / release/* / hotfix/*... • Deep-Dive Mechanics: GitHub Flow for SaaS microservices & GitFlow fits multi-versioned on-prem/SDKs • Advanced Patterns: Conventional Commits + SemVer tags & Merge queue & branch protection rules • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Conventional Commit & Annotated SemVer Tag Validator in Bash • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Git Branching Strategies

Purpose: To understand and apply Git Branching Strategies — `GitFlow` (`main`/`develop`/`release`/`hotfix`) vs Trunk-Based Development in real DevOps & Git CI/CD systems — specifi...

What Topic Is Next → Module 4 Capstone (Automated Branch Protection, Conventional Commit Hook & Release Tagging Workflow) → Next Topic 5.1: Production Git Commands — `revert` vs `reset`, `rebase`, `--force-with-lease`, `reflog` & `bisect`

Module 4 Hands-On Capstone Project (100 Marks): Automated Branch Protection, Conventional Commit Hook & Release Tagging Workflow — Implement a local/CI Git validation suite enforcing Conventional Commits (`feat:`, `fix:`, `chore:`), preventing direct force-pushes to `main`, and generating annotated SemVer release tags.

MODULE 5 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 5: Surgical Git Operations (`rebase`, `reflog`, `bisect`, `revert`) & Safe `git stash` Workflows

Recover from any "lost" commit using `git reflog`, hunt regressions in `O(log N)` steps with `git bisect`, understand `revert` vs `reset` & `--force-with-lease`, and master `git stash -u` / `git worktree`.

Objectives: ✓ Production Git Commands — `revert` vs `reset`, `rebase`, `--force-with-lease`, `reflog` & `bisect` — Mastering high-leverage Git commands: `git switch` / `git restore`, why `git revert` preserves shared history whereas `git reset --hard` rewrites local `HEAD`, `git rebase` Golden Rule (never rebase public shared branches!), `git push --force-with-lease`, `git reflog` 90-day safety net, and binary-searching bugs with `git bisect`. • ✓ Mastering `git stash` & `git worktree` — Untracked `-u` Trap, `apply` vs `pop` & `stash branch` — Temporarily shelving dirty working-tree edits during urgent P0 hotfix context switches: `git stash push -u -m "WIP: ..."`, why plain `git stash` **ignores untracked new files**, `git stash list`, `stash show -p`, `stash apply` (keeps stash) vs `stash pop` (drops if clean), `git stash branch`, and `git worktree` for parallel checkouts. • ✓ Build, verify, and ship Module Capstone: Automated `git bisect run` Regression Finder & Safe Hotfix Stash/Worktree Lab.

5.1 Production Git Commands — `revert` vs `reset`, `rebase`, `--force-with-lease`, `reflog` & `bisect`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Git revert (shared) vs git reset (local) & --soft vs --mixed vs --hard reset • Core Syntax, CLI & Production APIs: git switch, git restore, git revert, git reset (--soft|--mixed|--hard), git rebase -i, git p... • Deep-Dive Mechanics: Git push --force-with-lease safety & Git reflog 30-90 day recovery net • Advanced Patterns: Git bisect O(log N) bug search & Golden Rule of git rebase • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Recovering a Hard-Reset Commit Automatically Using `git reflog` • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Production Git Commands

Purpose: To understand and apply Production Git Commands — `revert` vs `reset`, `rebase`, `--force-with-lease`, `reflog` & `bisect` in real DevOps & Git CI/CD systems — specifical...

What Topic Is Next → Next Topic 5.2: Mastering `git stash` & `git worktree` — Untracked `-u` Trap, `apply` vs `pop` & `stash branch`

5.2 Mastering `git stash` & `git worktree` — Untracked `-u` Trap, `apply` vs `pop` & `stash branch`

[50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Git stash push -u (--include-untracked) & Always name stashes with -m message • Core Syntax, CLI & Production APIs: git stash push -u -m <msg>, git stash list, git stash pop, git stash apply, git stash branch... • Deep-Dive Mechanics: Stash apply (keep) vs stash pop (drop) & Pop conflict retains stash@{0} safely • Advanced Patterns: Git stash branch <new-branch> conflict-free & Git worktree parallel branch folders • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Stashing Tracked + Untracked Files Safely (`git stash push -u -m`) • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Mastering `git stash`

Purpose: To understand and apply Mastering `git stash` & `git worktree` — Untracked `-u` Trap, `apply` vs `pop` & `stash branch` in real DevOps & Git CI/CD systems — specifically:...

What Topic Is Next → Module 5 Capstone (Automated `git bisect run` Regression Finder & Safe Hotfix Stash/Worktree Lab) → Next Topic 6.1: Defensive Bash for DevOps — `set -euo pipefail`, `trap EXIT`, Quoting Rules & Retry Backoff

Module 5 Hands-On Capstone Project (100 Marks): Automated `git bisect run` Regression Finder & Safe Hotfix Stash/Worktree Lab — Write an automated non-interactive Git regression hunter (`git bisect run ./test.sh`) that pinpoints the exact offending commit across 64 commits in 6 steps, plus safe `--force-with-lease` &...

MODULE 6 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 6: Defensive Production Bash Scripting (`set -euo pipefail`) & Declarative Jenkins CI/CD

Eliminate silent shell automation bugs using strict-mode Bash (`set -euo pipefail`, `trap`, quoted expansions, `shellcheck`), and engineer scalable Declarative `Jenkinsfile` pipelines with ephemeral K8s/Docker agents.

Objectives: ✓ Defensive Bash for DevOps — `set -euo pipefail`, `trap EXIT`, Quoting Rules & Retry Backoff — Why default POSIX/Bash shells silently continue executing destructive `rm -rf /$UNSET_DIR` commands after an earlier step fails! Mastering `set -e` (errexit), `set -u` (nounset — catches `$UNSET_DIR` immediately!), `set -o pipefail` (propagates pipeline failures), `trap ... EXIT` cleanup, `"${var}"` word-splitting defense, and `ShellCheck`. • ✓ Jenkins Pipeline-as-Code — Declarative `Jenkinsfile`, Ephemeral Agents & Groovy String Interpolation Security — Controller (scheduling & UI; `0` build executors!) vs Ephemeral Agent Pods/Containers, Multibranch Pipelines reading versioned `Jenkinsfile`s, Declarative vs Scripted syntax, `credentials()` secret binding, Shared Libraries, and `post { always / failure }` cleanup. • ✓ Build, verify, and ship Module Capstone: Idempotent Zero-Leak Bash Deployment Script + Declarative Ephemeral `Jenkinsfile`.

6.1 Defensive Bash for DevOps — `set -euo pipefail`, `trap EXIT`, Quoting Rules & Retry Backoff [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Set -e (exit on error) & Set -u (error on unset variables!) • Core Syntax, CLI & Production APIs: #!/usr/bin/env bash, set -euo pipefail, trap cleanup EXIT, [[...]], "${var:?required}", mk... • Deep-Dive Mechanics: Set -o pipefail (catch broken pipes) & Trap cleanup EXIT signal safety • Advanced Patterns: "${VAR:?}" empty-variable rm -rf guard & ShellCheck static lint in CI • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Strict-Mode Bash (`set -euo pipefail` + `trap` + Parameter Guard) • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Defensive Bash for DevOps

Purpose: To understand and apply Defensive Bash for DevOps — `set -euo pipefail`, `trap EXIT`, Quoting Rules & Retry Backoff in real DevOps & Git CI/CD systems — specifically: Why...

What Topic Is Next → Next Topic 6.2: Jenkins Pipeline-as-Code — Declarative `Jenkinsfile`, Ephemeral Agents & Groovy String Interpolation Security

6.2 Jenkins Pipeline-as-Code — Declarative `Jenkinsfile`, Ephemeral Agents & Groovy String Interpolation Security [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Declarative Jenkinsfile in Git SCM & 0 executors on Jenkins Controller • Core Syntax, CLI & Production APIs: pipeline { agent ... environment { CREDS = credentials(...) } stages { ... } post { always {... • Deep-Dive Mechanics: Ephemeral Docker / K8s pod agents & Single-quoted sh '$SECRET' (no GString!) • Advanced Patterns: Options { timeout(...) disableConcurrentBuilds() } & Post { always { cleanWs() } } • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Hardened Declarative `Jenkinsfile` Using Single-Quoted `sh` Env Masking • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Jenkins Pipeline-as-Code

Purpose: To understand and apply Jenkins Pipeline-as-Code — Declarative `Jenkinsfile`, Ephemeral Agents & Groovy String Interpolation Security in real DevOps & Git CI/CD systems —...

What Topic Is Next → Module 6 Capstone (Idempotent Zero-Leak Bash Deployment Script + Declarative Ephemeral `Jenkinsfile`) → Next Topic 7.1: Ansible Automation — Agentless SSH, Idempotent FQCN Modules (`changed=0`), Handlers & Ansible Vault

Module 6 Hands-On Capstone Project (100 Marks): Idempotent Zero-Leak Bash Deployment Script + Declarative Ephemeral `Jenkinsfile` — Build a ShellCheck-clean deployment health script with `trap` temp-file cleanup and retry backoff, invoked from a Declarative `Jenkinsfile` running inside ephemeral container agents with mas...

MODULE 7 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 7: Idempotent Configuration Management (`Ansible`) & `Terraform` Remote State Internals

Automate fleet OS hardening with agentless idempotent Ansible Roles & Handlers, and master Terraform's core engine: `.tfstate` state locking, S3/Cloud backends, and zero-plaintext-secret state hygiene.

Objectives: ✓ Ansible Automation — Agentless SSH, Idempotent FQCN Modules (`changed=0`), Handlers & Ansible Vault — Why Ansible requires zero daemon agents on target Linux hosts (SSH + Python!), Inventories, Playbooks, Roles, Fully-Qualified Collection Names (`ansible.builtin.apt`, `ansible.builtin.copy`), why declarative modules guarantee **idempotence** whereas raw `ansible.builtin.shell` runs blindly every time unless guarded, `notify` Handlers, `--check --diff`, and `ansible-vault`. • ✓ Terraform State (`terraform.tfstate`) — Desired vs Real-World Graph, Remote Locking & Plaintext Secret Risk — Why Terraform needs `terraform.tfstate` (mapping HCL resource addresses like `aws_db_instance.main` to cloud provider IDs + caching dependency metadata), remote backends (`s3` with KMS encryption, versioning & locking), `terraform plan -out=tfplan`, `moved {}` / `import {}` blocks, and why `sensitive = true` does **NOT** encrypt `.tfstate`!. • ✓ Build, verify, and ship Module Capstone: Idempotent Ansible SSH Hardening Playbook + Encrypted Locked S3 Terraform Backend.

7.1 Ansible Automation — Agentless SSH, Idempotent FQCN Modules (`changed=0`), Handlers & Ansible Vault [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Agentless SSH + Python execution & Idempotence (2nd run -> changed=0) • Core Syntax, CLI & Production APIs: ansible-playbook -i inv.ini site.yml --check --diff, ansible.builtin.package / template / se... • Deep-Dive Mechanics: FQCN (ansible.builtin.*) modules & Validate: before notify Handler restart • Advanced Patterns: --check --diff dry-run verification & Ansible-vault encrypted group_vars • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Idempotent FQCN Ansible Playbook with Pre-Restart Syntax Validation • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Ansible Automation

Purpose: To understand and apply Ansible Automation — Agentless SSH, Idempotent FQCN Modules (`changed=0`), Handlers & Ansible Vault in real DevOps & Git CI/CD systems — specifica...

What Topic Is Next → Next Topic 7.2: Terraform State (`terraform.tfstate`) — Desired vs Real-World Graph, Remote Locking & Plaintext Secret Risk

7.2 Terraform State (`terraform.tfstate`) — Desired vs Real-World Graph, Remote Locking & Plaintext Secret Risk [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: HCL address <-> Cloud ID state mapping & S3 KMS encryption + state locking • Core Syntax, CLI & Production APIs: terraform { backend "s3" { encrypt = true, use_lockfile = true } }, terraform state list, im... • Deep-Dive Mechanics: Sensitive=true STILL plaintext in tfstate! & Never commit *.tfstate to Git • Advanced Patterns: Declarative moved {} & import {} blocks & Terraform plan -out=tfplan exact apply • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Encrypted & State-Locked Production Terraform `backend "s3"` Spec • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Terraform State (`terraform.tfstate`)

Purpose: To understand and apply Terraform State (`terraform.tfstate`) — Desired vs Real-World Graph, Remote Locking & Plaintext Secret Risk in real DevOps & Git CI/CD systems — s...

What Topic Is Next → Module 7 Capstone (Idempotent Ansible SSH Hardening Playbook + Encrypted Locked S3 Terraform Backend) → Next Topic 8.1: Terraform Loops — `for` Expressions vs `for_each` Meta-Argument vs The `count` Index-Shift Trap

Module 7 Hands-On Capstone Project (100 Marks): Idempotent Ansible SSH Hardening Playbook + Encrypted Locked S3 Terraform Backend — Author an idempotent FQCN Ansible Playbook validating config syntax before restarting `sshd` via `handlers`, plus a locked & KMS-encrypted Terraform S3 backend configuration.

MODULE 8 OF 16 • STAGE 2 • 12 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 8: Dynamic Terraform HCL (`for_each`, `for` Expressions, `dynamic`) & Safe `output` Contracts

Eliminate brittle index-shifting `count` bugs using keyed `for_each` + HCL `for` comprehensions, and expose clean cross-module contracts using typed, precondition-checked, `sensitive` Terraform `output`s.

Objectives: ✓ Terraform Loops — `for` Expressions vs `for_each` Meta-Argument vs The `count` Index-Shift Trap — Mastering Terraform's 3 iteration constructs: (1) **`for_each`** meta-argument creating keyed resource instances (`resource.type["key"]`), (2) **`count`** (`0..N-1` — safe ONLY for `0/1` boolean feature toggles!), and (3) HCL **`[for ...]` / `{for ...}`** data-transformation comprehensions inside `locals` / `output`s (plus `dynamic` nested blocks). • ✓ Terraform Outputs — Cross-Module Interfaces, `sensitive = true`, `ephemeral` & `terraform output -raw` — How child modules return computed resource IDs/endpoints solely through `output` blocks (`module.vpc.private_subnet_ids`), redacting console/CI logs with `sensitive = true` (and Terraform 1.10+ `ephemeral = true` values omitted from state!), enforcing output `precondition` invariants, and scripting `terraform output -raw` / `-json`. • ✓ Build, verify, and ship Module Capstone: Zero-Destruction Multi-AZ Subnet Module (`for_each` + Filtered `for` Map + `output`s).

8.1 Terraform Loops — `for` Expressions vs `for_each` Meta-Argument vs The `count` Index-Shift Trap [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: For_each keyed map/set stability & Count index-shift destruction trap! • Core Syntax, CLI & Production APIs: for_each = toset(...) / map, each.key & each.value, { for k, v in map : k => v if ... }, dyn... • Deep-Dive Mechanics: Count = var.enabled ? 1 : 0 toggle & { for k, v in ... : k => v if ... } • Advanced Patterns: For_each keys must be known at plan & Dynamic blocks for nested rule lists • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Terraform HCL `for` Map Filtering & Keyed `for_each` Blueprint • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Terraform Loops

Purpose: To understand and apply Terraform Loops — `for` Expressions vs `for_each` Meta-Argument vs The `count` Index-Shift Trap in real DevOps & Git CI/CD systems — specifically:...

What Topic Is Next → Next Topic 8.2: Terraform Outputs — Cross-Module Interfaces, `sensitive = true`, `ephemeral` & `terraform output -raw`

8.2 Terraform Outputs — Cross-Module Interfaces, `sensitive = true`, `ephemeral` & `terraform output -raw` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Module.<name>.<output> encapsulation & Terraform output -raw (no extra quotes!) • Core Syntax, CLI & Production APIs: output "<name>" { value = ..., description = ..., sensitive = true, precondition { ... } },... • Deep-Dive Mechanics: Terraform output -json | jq parsing & Mandatory sensitive = true propagation • Advanced Patterns: Output precondition {} contract checks & Ephemeral = true secret-free values • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Precondition-Guarded & Sensitive-Redacted Terraform `output` Spec • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Terraform Outputs

Purpose: To understand and apply Terraform Outputs — Cross-Module Interfaces, `sensitive = true`, `ephemeral` & `terraform output -raw` in real DevOps & Git CI/CD systems — specif...

What Topic Is Next → Module 8 Capstone (Zero-Destruction Multi-AZ Subnet Module (`for_each` + Filtered `for` Map + `output`s)) → Next Topic 9.1: Terraform Formatting & Static Quality Gates — `fmt -check`, `validate`, `check {}` Blocks & `tflint`

Module 8 Hands-On Capstone Project (100 Marks): Zero-Destruction Multi-AZ Subnet Module (`for_each` + Filtered `for` Map + `output`s) — Refactor a dangerous `count`-indexed subnet configuration into a stable `for_each` map keyed by Availability Zone so removing middle items never destroys and recreates surviving subnets!

MODULE 9 OF 16 • STAGE 3 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 9: Terraform Code Quality Gates (`fmt`, `validate`, `tflint`) & AWS Cloud DevOps

Enforce deterministic IaC pull-request quality gates (`terraform fmt -check`, `validate`, `tflint`, `trivy config`), and architect secretless AWS DevOps pipelines using OIDC Federation, ECR, and ECS/EKS Blue-Green deployments.

Objectives: ✓ Terraform Formatting & Static Quality Gates — `fmt -check`, `validate`, `check {}` Blocks & `tflint` — Keeping multi-team HCL clean and secure before `plan`: `terraform fmt -check -recursive` (canonical 2-space alignment, exits code `3` if unformatted!), `terraform validate` (offline HCL type & reference verification via `-backend=false`), `tflint` (catches invalid AWS instance types like `t3.superlarge` before `apply`!), and `trivy config` / `checkov`. • ✓ AWS DevOps Engineering — IAM OIDC Federation (Zero Static Keys!), ECR Immutable Tags & CodeDeploy Blue-Green — Modern AWS CI/CD security: replacing leaked long-lived IAM User access keys (`AKIA...`) with short-lived 1-hour STS tokens (`ASIA...`) federated via GitHub/GitLab **OIDC (`AssumeRoleWithWebIdentity`)**, ECR `IMMUTABLE` image tag mutability + scan-on-push, SSM Parameter Store / Secrets Manager, and CodeDeploy Blue-Green ALB target-group shifts with

CloudWatch auto-rollback. • ✓ Build, verify, and ship Module Capstone: Automated Terraform PR Linter + Secretless AWS OIDC AssumeRole Trust Policy.

9.1 Terraform Formatting & Static Quality Gates — `fmt -check`, `validate`, `check {}` Blocks & `tflint` [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Terraform fmt -check -recursive (exit 3) & Init -backend=false offline validate • Core Syntax, CLI & Production APIs: terraform fmt -check -recursive -diff, terraform init -backend=false && terraform validate,... • Deep-Dive Mechanics: Validate (HCL schema) vs tflint (AWS API) & Trivy config / checkov IaC security scan • Advanced Patterns: Terraform 1.5+ check {} assertions & .terraform.lock.hcl provider hashes • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: CI Pull-Request Terraform Quality Gate (`fmt -check` + Offline `validate`) • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Terraform Formatting

Purpose: To understand and apply Terraform Formatting & Static Quality Gates — `fmt -check`, `validate`, `check {}` Blocks & `tflint` in real DevOps & Git CI/CD systems — specific...

What Topic Is Next → Next Topic 9.2: AWS DevOps Engineering — IAM OIDC Federation (Zero Static Keys!), ECR Immutable Tags & CodeDeploy Blue-Green

9.2 AWS DevOps Engineering — IAM OIDC Federation (Zero Static Keys!), ECR Immutable Tags & CodeDeploy Blue-Green [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: OIDC AssumeRoleWithWebIdentity (ASIA...) & Strict token sub + aud trust condition • Core Syntax, CLI & Production APIs: sts:AssumeRoleWithWebIdentity (token.actions.githubusercontent.com:sub), aws_ecr_repository... • Deep-Dive Mechanics: Never use static AKIA IAM User keys in CI & ECR image_tag_mutability = IMMUTABLE • Advanced Patterns: ECS / EKS CodeDeploy Blue-Green shift & CloudWatch Alarm automated rollback • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Locked-Down AWS IAM OIDC Trust Policy JSON for Production CD • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for AWS DevOps Engineering

Purpose: To understand and apply AWS DevOps Engineering — IAM OIDC Federation (Zero Static Keys!), ECR Immutable Tags & CodeDeploy Blue-Green in real DevOps & Git CI/CD systems —...

What Topic Is Next → Module 9 Capstone (Automated Terraform PR Linter + Secretless AWS OIDC AssumeRole Trust Policy) → Next Topic 10.1: Azure DevOps Pipelines — Multi-Stage `azure-pipelines.yml`, `deployment` Environments, Templates & Workload Identity

Module 9 Hands-On Capstone Project (100 Marks): Automated Terraform PR Linter + Secretless AWS OIDC AssumeRole Trust Policy — Create a zero-network-credential CI check validating HCL formatting/types (`fmt -check` + `validate`) paired with a strict `repo`+`environment`-scoped AWS IAM OIDC Trust Policy.

MODULE 10 OF 16 • STAGE 3 • 11 LIVE + LAB HOURS • 2 TOPICS (16 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Module 10: Enterprise `Azure DevOps` Multi-Stage YAML Pipelines & Self-Healing SRE Automation

Build governed multi-stage `azure-pipelines.yml` workflows with Workload Identity Federation, Templates, and Environment Approvals, and eliminate operational toil through idempotent event-driven SRE automation.

Objectives: ✓ Azure DevOps Pipelines — Multi-Stage `azure-pipelines.yml`, `deployment` Environments, Templates & Workload Identity — Enterprise Azure DevOps (`Repos`, `Boards`, `Pipelines`, `Artifacts`): `stages -> jobs -> steps` hierarchy, `deployment` jobs bound to governed `Environments` (Approvals, Business-Hours Gates, `lockBehavior: sequential`), `extends` YAML governance templates, Azure Key Vault variable groups, and secretless **Workload Identity Federation**. • ✓ DevOps & SRE Automation — Eliminating Toil, Circuit-Broken Auto-Remediation, Policy-as-Code & Drift Detection — Google SRE definition of **Toil** (manual, repetitive, automatable, tactical, devoid of enduring value, scaling `O(N)` with fleet size), Policy-as-Code (`OPA` / `Rego` / `Kyverno`), automated dependency upgrades (`Renovate` / `Dependabot`), scheduled Terraform drift detection, and bounding self-healing scripts with blast-radius **circuit breakers**. • ✓ Build, verify, and ship Module Capstone: Reusable `azure-pipelines.yml` Governance Template + Bounded SRE Auto-Remediation Engine.

10.1 Azure DevOps Pipelines — Multi-Stage `azure-pipelines.yml`, `deployment` Environments, Templates & Workload Identity [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: Stages -> deployment -> environment gates & Workload Identity Federation (no secrets) • Core Syntax, CLI & Production APIs: azure-pipelines.yml (stages, deployment, environment, lockBehavior: sequential, extends: tem... • Deep-Dive Mechanics: LockBehavior: sequential race guard & Extends: YAML compliance templates • Advanced Patterns: ${{ compile }} vs $(runtime) expressions & Azure Key Vault linked variable groups • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Multi-Stage `azure-pipelines.yml` with Sequential `deployment` Lock • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Azure DevOps Pipelines

Purpose: To understand and apply Azure DevOps Pipelines — Multi-Stage `azure-pipelines.yml`, `deployment` Environments, Templates & Workload Identity in real DevOps & Git CI/CD sy...

What Topic Is Next → Next Topic 10.2: DevOps & SRE Automation — Eliminating Toil, Circuit-Broken Auto-Remediation, Policy-as-Code & Drift Detection

10.2 DevOps & SRE Automation — Eliminating Toil, Circuit-Broken Auto-Remediation, Policy-as-Code & Drift Detection [50 mins • Assignment (100M) • Hands-On Lab (50M) • 10 Mock Sets • 10 AI Vivas]

Subtopics Covered: SRE Toil (<50% engineering time cap) & Blast-radius circuit breaker (<10% fleet) • Core Syntax, CLI & Production APIs: OPA / Rego Policy-as-Code, Renovate, Scheduled Drift Detection (terraform plan -detailed-exi... • Deep-Dive Mechanics: Terraform plan -detailed-exitcode (0/1/2) & OPA / Kyverno Policy-as-Code gates • Advanced Patterns: Renovate automated lockfile PRs & Idempotent audit-logged ChatOps runbooks • Production Failure Mode & Pitfall Guard • Hands-On Live Coding Lab: Blast-Radius Circuit-Broken SRE Auto-Remediation Guard in Bash • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for DevOps

Purpose: To understand and apply DevOps & SRE Automation — Eliminating Toil, Circuit-Broken Auto-Remediation, Policy-as-Code & Drift Detection in real DevOps & Git CI/CD systems —...

What Topic Is Next → Module 10 Capstone (Reusable `azure-pipelines.yml` Governance Template + Bounded SRE Auto-Remediation Engine) → Next Topic 11.1: Reusable Workflows, Matrix Caching & Concurrency Cancel-in-Progress

Module 10 Hands-On Capstone Project (100 Marks): Reusable `azure-pipelines.yml` Governance Template + Bounded SRE Auto-Remediation Engine — Author a multi-stage `azure-pipelines.yml` enforcing gated `production` `deployment` approvals and Workload Identity Federation, paired with a circuit-broken SRE self-healing script that pre...

MODULE 11 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Reusable CI/CD Pipelines, OIDC Cloud Auth & Supply-Chain DevSecOps

Production-grade DevOps & Git CI/CD Training engineering module covering Reusable CI/CD Pipelines, OIDC Cloud Auth & Supply-Chain DevSecOps with hands-on labs and capstone build.

Objectives: ✓ Reusable Workflows, Matrix Caching & Concurrency Cancel-in-Progress — Purpose: Cut CI pipeline feedback from 18 minutes down to 3 minutes across microservices. • ✓ Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews — Purpose: Eliminate leaked long-lived cloud secrets and spin up live PR preview URLs automatically. • ✓ Shift-Left DevSecOps: Secret Scanning, SAST (Semgrep), Trivy & SBOM Signing — Purpose: Block leaked keys, vulnerable dependencies, and unsigned images before deployment. • ✓ Semantic Release Automation, Immutable Artifact Promotion & Changelogs — Purpose: Promote the exact tested SHA artifact from staging to production without rebuilding.

11.1 Reusable Workflows, Matrix Caching & Concurrency Cancel-in-Progress

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Matrix Caching • Integration with Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews • Live Debugging, Telemetry & Technical Interview Trade • What Is Reusable Workflows, Matrix Caching & Concurrency Cancel • In DevOps & Git CI/CD Training, Reusable Workflows, Matrix Caching & Concurrency Cancel • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Reusable Workflows, Matrix Caching & Concurrency Cancel-in-Progress in real DevOps & Git CI/CD systems — specifically: Cut CI pipeline feedback fr...

What Topic Is Next → Next Topic 11.2: Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews

11.2 Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Ephemeral Environment Previews • Integration with Shift • Live Debugging, Telemetry & Technical Interview Trade • What Is Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews exists for one.... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Keyless OIDC Federation to AWS/GCP & Ephemeral Environment Previews in real DevOps & Git CI/CD systems — specifically: Eliminate leaked long-lived...

What Topic Is Next → Next Topic 11.3: Shift-Left DevSecOps: Secret Scanning, SAST (Semgrep), Trivy & SBOM Signing

11.3 Shift-Left DevSecOps: Secret Scanning, SAST (Semgrep), Trivy & SBOM Signing

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of SAST (Semgrep) • Integration with Semantic Release Automation, Immutable Artifact Promotion & Changelogs • Live Debugging, Telemetry & Technical Interview Trade • What Is Shift • In DevOps & Git CI/CD Training, Shift • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Shift-Left DevSecOps: Secret Scanning, SAST (Semgrep), Trivy & SBOM Signing in real DevOps & Git CI/CD systems — specifically: Block leaked keys,...

What Topic Is Next → Next Topic 11.4: Semantic Release Automation, Immutable Artifact Promotion & Changelogs

11.4 Semantic Release Automation, Immutable Artifact Promotion & Changelogs

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Immutable Artifact Promotion • Integration with Reusable Workflows, Matrix Caching & Concurrency Cancel • Live Debugging, Telemetry & Technical Interview Trade • What Is Semantic Release Automation, Immutable Artifact Promotion & Changelogs & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Semantic Release Automation, Immutable Artifact Promotion & Changelogs exists for o... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Semantic Release Automation, Immutable Artifact Promotion & Changelogs in real DevOps & Git CI/CD systems — specifically: Promote the exact tested...

What Topic Is Next → Module 11 Capstone (Zero-Trust GitHub Actions CI/CD Pipeline with SAST/Container Gates) → Next Topic 12.1: Modular Terraform Design, Remote State Locking, Imports & Drift Checks

Module 11 Hands-On Capstone Project (100 Marks): Zero-Trust GitHub Actions CI/CD Pipeline with SAST/Container Gates — Design, implement, test, and document 'Zero-Trust GitHub Actions CI/CD Pipeline with SAST/Container Gates' applying every production concept from Module 11 (Reusable CI/CD Pipelines, OIDC CI...

MODULE 12 OF 16 • STAGE 3 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Production Terraform IaC, Ansible Hardening & Secrets Management

Production-grade DevOps & Git CI/CD Training engineering module covering Production Terraform IaC, Ansible Hardening & Secrets Management with hands-on labs and capstone build.

Objectives: ✓ Modular Terraform Design, Remote State Locking, Imports & Drift Checks — Purpose: Manage VPCs, clusters, DNS, and databases cleanly across dev, stage, and prod. • ✓ Policy-as-Code Checks (Checkov / OPA) & Automated Terraform Plan PR Reviews — Purpose: Catch accidental public S3 buckets or destructive resource replacements in PR review. • ✓ Idempotent Linux Host Hardening & Patching with Ansible Roles — Purpose: Standardize SSH, firewall (ufw/nftables), sysctl, and systemd baselines across fleets. • ✓ Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager — Purpose: Issue

short-lived leased DB credentials with instant revocation capabilities.

12.1 Modular Terraform Design, Remote State Locking, Imports & Drift Checks [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Remote State Locking • Integration with Policy • Live Debugging, Telemetry & Technical Interview Trade • What Is Modular Terraform Design, Remote State Locking, Imports & Drift Checks & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Modular Terraform Design, Remote State Locking, Imports & Drift Checks exists for o... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Modular Terraform Design, Remote State Locking, Imports & Drift Checks in real DevOps & Git CI/CD systems — specifically: Manage VPCs, clusters, D...

What Topic Is Next → Next Topic 12.2: Policy-as-Code Checks (Checkov / OPA) & Automated Terraform Plan PR Reviews

12.2 Policy-as-Code Checks (Checkov / OPA) & Automated Terraform Plan PR Reviews [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Automated Terraform Plan PR Reviews • Integration with Idempotent Linux Host Hardening & Patching with Ansible Roles • Live Debugging, Telemetry & Technical Interview Trade • What Is Policy • In DevOps & Git CI/CD Training, Policy • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Policy-as-Code Checks (Checkov / OPA) & Automated Terraform Plan PR Reviews in real DevOps & Git CI/CD systems — specifically: Catch accidental pu...

What Topic Is Next → Next Topic 12.3: Idempotent Linux Host Hardening & Patching with Ansible Roles

12.3 Idempotent Linux Host Hardening & Patching with Ansible Roles [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Patching with Ansible Roles • Integration with Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager • Live Debugging, Telemetry & Technical Interview Trade • What Is Idempotent Linux Host Hardening & Patching with Ansible Roles & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Idempotent Linux Host Hardening & Patching with Ansible Roles exists for one clear.... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Idempotent Linux Host Hardening & Patching with Ansible Roles in real DevOps & Git CI/CD systems — specifically: Standardize SSH, firewall (ufw/nf...

What Topic Is Next → Next Topic 12.4: Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager

12.4 Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Edge • Integration with Modular Terraform Design, Remote State Locking, Imports & Drift Checks • Live Debugging, Telemetry & Technical Interview Trade • What Is Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager exists for... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Centralized Dynamic Secrets with HashiCorp Vault / Cloud Secrets Manager in real DevOps & Git CI/CD systems — specifically: Issue short-lived leas...

What Topic Is Next → Module 12 Capstone (Modular Multi-AZ Cloud Stack Provisioned 100% via Terraform) → Next Topic 13.1: Declarative ArgoCD GitOps Sync, Health Waves & Drift Self-Healing

Module 12 Hands-On Capstone Project (100 Marks): Modular Multi-AZ Cloud Stack Provisioned 100% via Terraform — Design, implement, test, and document 'Modular Multi-AZ Cloud Stack Provisioned 100% via Terraform' applying every production concept from Module 12 (Production Terraform IaC, Ansible Harden...

MODULE 13 OF 16 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

GitOps Kubernetes Delivery (ArgoCD), Progressive Rollouts & Autoscaling

Production-grade DevOps & Git CI/CD Training engineering module covering GitOps Kubernetes Delivery (ArgoCD), Progressive Rollouts & Autoscaling with hands-on labs and capstone build.

Objectives: ✓ Declarative ArgoCD GitOps Sync, Health Waves & Drift Self-Healing — Purpose: Converge cluster state automatically to Git commits with full audit history. • ✓ Blue/Green & Metric-Gated Canary Rollouts with Automated Rollback — Purpose: Shift 5% -> 25% -> 100% live traffic while verifying latency and 5xx thresholds. • ✓ Ingress TLS Automation (cert-manager), ExternalDNS & Network Isolation — Purpose: Provision HTTPS certificates and DNS records automatically when services launch. • ✓ Cluster Cost & Reliability Engineering: HPA, Karpenter Spot Mix & PDBs — Purpose: Maintain 99.95% availability while cutting compute bills using diversified instances.

13.1 Declarative ArgoCD GitOps Sync, Health Waves & Drift Self-Healing [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Health Waves • Integration with Blue/Green & Metric • Live Debugging, Telemetry & Technical Interview Trade • What Is Declarative ArgoCD GitOps Sync, Health Waves & Drift Self • In DevOps & Git CI/CD Training, Declarative ArgoCD GitOps Sync, Health Waves & Drift Self • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Declarative ArgoCD GitOps Sync, Health Waves & Drift Self-Healing in real DevOps & Git CI/CD systems — specifically: Converge cluster state automa...

What Topic Is Next → Next Topic 13.2: Blue/Green & Metric-Gated Canary Rollouts with Automated Rollback

13.2 Blue/Green & Metric-Gated Canary Rollouts with Automated Rollback [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Metric • Integration with Ingress TLS Automation (cert • Live Debugging, Telemetry & Technical Interview Trade • What Is Blue/Green & Metric • In DevOps & Git CI/CD Training, Blue/Green & Metric • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Blue/Green & Metric-Gated Canary Rollouts with Automated Rollback in real DevOps & Git CI/CD systems — specifically: Shift 5% -> 25% -> 100% live...

What Topic Is Next → Next Topic 13.3: Ingress TLS Automation (cert-manager), ExternalDNS & Network Isolation

13.3 Ingress TLS Automation (cert-manager), ExternalDNS & Network Isolation

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of ExternalDNS • Integration with Cluster Cost & Reliability Engineering • Live Debugging, Telemetry & Technical Interview Trade • What Is Ingress TLS Automation (cert • In DevOps & Git CI/CD Training, Ingress TLS Automation (cert • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Ingress TLS Automation (cert-manager), ExternalDNS & Network Isolation in real DevOps & Git CI/CD systems — specifically: Provision HTTPS certific...

What Topic Is Next → Next Topic 13.4: Cluster Cost & Reliability Engineering: HPA, Karpenter Spot Mix & PDBs

13.4 Cluster Cost & Reliability Engineering: HPA, Karpenter Spot Mix & PDBs

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Reliability Engineering • Integration with Declarative ArgoCD GitOps Sync, Health Waves & Drift Self • Live Debugging, Telemetry & Technical Interview Trade • What Is Cluster Cost & Reliability Engineering • In DevOps & Git CI/CD Training, Cluster Cost & Reliability Engineering • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Cluster Cost & Reliability Engineering: HPA, Karpenter Spot Mix & PDBs in real DevOps & Git CI/CD systems — specifically: Maintain 99.95% availabi...

What Topic Is Next → Module 13 Capstone (Zero-Downtime Canary Delivery Platform on Kubernetes) → Next Topic 14.1: SLIs, SLOs & Multi-Window Error-Budget Burn-Rate Alerting

Module 13 Hands-On Capstone Project (100 Marks): Zero-Downtime Canary Delivery Platform on Kubernetes — Design, implement, test, and document 'Zero-Downtime Canary Delivery Platform on Kubernetes' applying every production concept from Module 13 (GitOps Kubernetes Delivery (ArgoCD), Progressiv...

MODULE 14 OF 16 • STAGE 4 • 16 LIVE + LAB HOURS • 4 TOPICS (32 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

SRE Observability Stack (Prometheus, Grafana, Loki, OTel) & Incident Ops

Production-grade DevOps & Git CI/CD Training engineering module covering SRE Observability Stack (Prometheus, Grafana, Loki, OTel) & Incident Ops with hands-on labs and capstone build.

Objectives: ✓ SLIs, SLOs & Multi-Window Error-Budget Burn-Rate Alerting — Purpose: Eliminate alert fatigue by paging engineers only when real user SLO budgets burn. • ✓ Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces — Purpose: Correlate metrics, structured logs, and distributed traces in one unified pane. • ✓ Load Testing (k6) Capacity Planning & Chaos Fault-Injection GameDays — Purpose: Find breaking points and verify failover runbooks weeks before peak sale traffic. • ✓ Automated Backup Restore Verification, Runbooks & Blameless Postmortems — Purpose: Turn every incident or drill into permanent automated platform resilience.

14.1 SLIs, SLOs & Multi-Window Error-Budget Burn-Rate Alerting

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of SLOs • Integration with Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces • Live Debugging, Telemetry & Technical Interview Trade • What Is SLIs, SLOs & Multi • In DevOps & Git CI/CD Training, SLIs, SLOs & Multi • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply SLIs, SLOs & Multi-Window Error-Budget Burn-Rate Alerting in real DevOps & Git CI/CD systems — specifically: Eliminate alert fatigue by paging eng...

What Topic Is Next → Next Topic 14.2: Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces

14.2 Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Grafana Dashboards • Integration with Load Testing (k6) Capacity Planning & Chaos Fault • Live Debugging, Telemetry & Technical Interview Trade • What Is Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces exists for.... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Prometheus PromQL, Grafana Dashboards, Loki Logs & OpenTelemetry Traces in real DevOps & Git CI/CD systems — specifically: Correlate metrics, stru...

What Topic Is Next → Next Topic 14.3: Load Testing (k6) Capacity Planning & Chaos Fault-Injection GameDays

14.3 Load Testing (k6) Capacity Planning & Chaos Fault-Injection GameDays

[65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Chaos Fault • Integration with Automated Backup Restore Verification, Runbooks & Blameless Postmortems • Live Debugging, Telemetry & Technical Interview Trade • What Is Load Testing (k6) Capacity Planning & Chaos Fault • In DevOps & Git CI/CD Training, Load Testing (k6) Capacity Planning & Chaos Fault • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Load Testing (k6) Capacity Planning & Chaos Fault-Injection GameDays in real DevOps & Git CI/CD systems — specifically: Find breaking points and v...

What Topic Is Next → Next Topic 14.4: Automated Backup Restore Verification, Runbooks & Blameless Postmortems

14.4 Automated Backup Restore Verification, Runbooks & Blameless Postmortems [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Plain • Production Implementation of Runbooks • Integration with SLIs, SLOs & Multi • Live Debugging, Telemetry & Technical Interview Trade • What Is Automated Backup Restore Verification, Runbooks & Blameless Postmortems & Why Industry Teams Use It • In DevOps & Git CI/CD Training, Automated Backup Restore Verification, Runbooks & Blameless Postmortems exists for.... • Even if you come from a non • Runnable Code Implementation (YAML)

Purpose: To understand and apply Automated Backup Restore Verification, Runbooks & Blameless Postmortems in real DevOps & Git CI/CD systems — specifically: Turn every incident or...

What Topic Is Next → Module 14 Capstone (Full SRE Golden-Signals, SLO Burn-Rate & Disaster Drill Suite) → Next Topic 15.1: Keyless OIDC Cloud Authentication, Least-Privilege Workflow Permissions &Pinned Actions

Module 14 Hands-On Capstone Project (100 Marks): Full SRE Golden-Signals, SLO Burn-Rate & Disaster Drill Suite — Design, implement, test, and document 'Full SRE Golden-Signals, SLO Burn-Rate & Disaster Drill Suite' applying every production concept from Module 14 (SRE Observability Stack (Prometheus, G...

MODULE 15 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

DevSecOps Supply Chain, OIDC Keyless Cloud Deploys & Terraform State Governance

Self-explanatory list-and-bullet production mastery module covering DevSecOps Supply Chain, OIDC Keyless Cloud Deploys & Terraform State Governance in DevOps & Git CI/CD Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Keyless OIDC Cloud Authentication, Least-Privilege Workflow Permissions &Pinned Actions — Purpose: Eliminate long-lived cloud secrets in CI/CD and block supply-chain workflow injection. • ✓ Automated SAST, Secret Scanning, Dependency Audits & Container CVE Gates — Purpose: Catch leaked keys and vulnerable packages on every pull request before merge. • ✓ Remote Terraform State Locking, Drift Detection & Reusable Environment Modules — Purpose: Manage dev, staging, and production infrastructure safely across multi-engineer teams. • ✓ Build, verify, and document 'Keyless GitHub Actions OIDC -> AWS/Docker/Terraform Pipeline with Policy Gates' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

15.1 Keyless OIDC Cloud Authentication, Least-Privilege Workflow Permissions &Pinned Actions [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Keyless OIDC Cloud Authentication, Least • Performance Optimization, Complexity & Latency Impact of Keyless OIDC Cloud Authentication, Least • Live Debugging, Stack • Runnable Code Implementation (YAML) • Hands • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Keyless OIDC Cloud Authentication • Production Hardening & Security: OIDC secretless CI, SBOM/Cosign image signing, Vault & policy-as-code

Purpose: To understand and apply Keyless OIDC Cloud Authentication, Least-Privilege Workflow Permissions &Pinned Actions in real DevOps & Git CI/CD systems — specifically: Elimina...

What Topic Is Next → Next Topic 15.2: Automated SAST, Secret Scanning, Dependency Audits & Container CVE Gates

15.2 Automated SAST, Secret Scanning, Dependency Audits & Container CVE Gates [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Automated SAST, Secret Scanning, Dependency Audits & Container CVE Gates • Performance Optimization, Complexity & Latency Impact of Automated SAST, Secret Scanning, Dependency Audits & Conta... • Live Debugging, Stack • Runnable Code Implementation (YAML) • Hands • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Automated SAST • Production Hardening & Security: OIDC secretless CI, SBOM/Cosign image signing, Vault & policy-as-code

Purpose: To understand and apply Automated SAST, Secret Scanning, Dependency Audits & Container CVE Gates in real DevOps & Git CI/CD systems — specifically: Catch leaked keys and...

What Topic Is Next → Next Topic 15.3: Remote Terraform State Locking, Drift Detection & Reusable Environment Modules

15.3 Remote Terraform State Locking, Drift Detection & Reusable Environment Modules [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Remote Terraform State Locking, Drift Detection & Reusable Environment Mod... • Performance Optimization, Complexity & Latency Impact of Remote Terraform State Locking, Drift Detection & Reusable... • Live Debugging, Stack • Runnable Code Implementation (YAML) • Hands • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Remote Terraform State Locking • Production Hardening & Security: OIDC secretless CI, SBOM/Cosign image signing, Vault & policy-as-code

Purpose: To understand and apply Remote Terraform State Locking, Drift Detection & Reusable Environment Modules in real DevOps & Git CI/CD systems — specifically: Manage dev, stag...

What Topic Is Next → Module 15 Capstone (Keyless GitHub Actions OIDC -> AWS/Docker/Terraform Pipeline with Policy Gates) → Next Topic 16.1: Blue-Green, Canary & Rolling Releases with Automated Health-Check Rollbacks

Module 15 Hands-On Capstone Project (100 Marks): Keyless GitHub Actions OIDC -> AWS/Docker/Terraform Pipeline with Policy Gates — Architect, implement, test, and document 'Keyless GitHub Actions OIDC -> AWS/Docker/Terraform Pipeline with Policy Gates' applying every bullet-wise engineering concept from Module 15 (DevSe...

MODULE 16 OF 16 • STAGE 4 • 14 LIVE + LAB HOURS • 3 TOPICS (24 SUBTOPICS) • 1 CAPSTONE PROJECT (100 MARKS)

Progressive Delivery (Blue-Green/Canary), SLO Observability & Incident Response

Self-explanatory list-and-bullet production mastery module covering Progressive Delivery (Blue-Green/Canary), SLO Observability & Incident Response in DevOps & Git CI/CD Training, paired with hands-on labs, capstone repository, Topic Mock Tests, and Spoken MockAttempt AI Viva.

Objectives: ✓ Blue-Green, Canary & Rolling Releases with Automated Health-Check Rollbacks — Purpose: Ship multiple production deployments per day while keeping blast radius near zero. • ✓ Four Golden Signals (Latency, Traffic, Errors, Saturation), SLIs/SLOs & Error Budgets — Purpose: Replace noisy CPU pager fatigue with actionable symptom-based SRE alerts. • ✓ Blameless Postmortems, Runbook Automation & DevOps/SRE Interview Scenarios — Purpose: Walk through broken CI/CD pipelines, DNS/TLS outages, and scaling incidents calmly. • ✓ Build, verify, and document 'Prometheus + Grafana Golden-Signals Alerting Stack & Zero-Downtime Rollback Drill' with clean Git commits and edge-case protection. • ✓ Clear 100% of Topic Timed Mock Tests and Spoken Voice + MockAttempt AI Technical Interviews.

16.1 Blue-Green, Canary & Rolling Releases with Automated Health-Check Rollbacks [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Blue • Performance Optimization, Complexity & Latency Impact of Blue • Live Debugging, Stack • Runnable Code Implementation (YAML) • Hands • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Blue-Green • Production Hardening & Security: OIDC secretless CI, SBOM/Cosign image signing, Vault & policy-as-code

Purpose: To understand and apply Blue-Green, Canary & Rolling Releases with Automated Health-Check Rollbacks in real DevOps & Git CI/CD systems — specifically: Ship multiple produ...

What Topic Is Next → Next Topic 16.2: Four Golden Signals (Latency, Traffic, Errors, Saturation), SLIs/SLOs & Error Budgets

16.2 Four Golden Signals (Latency, Traffic, Errors, Saturation), SLIs/SLOs & Error Budgets [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Four Golden Signals (Latency, Traffic, Errors, Saturation), SLIs/SLOs & Er... • Performance Optimization, Complexity & Latency Impact of Four Golden Signals (Latency, Traffic, Errors, Saturation)... • Live Debugging, Stack • Runnable Code Implementation (YAML) • Hands • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Four Golden Signals (Latency, Traffic, Errors, Satur... • Production Hardening & Security: OIDC secretless CI, SBOM/Cosign image signing, Vault & policy-as-code

Purpose: To understand and apply Four Golden Signals (Latency, Traffic, Errors, Saturation), SLIs/SLOs & Error Budgets in real DevOps & Git CI/CD systems — specifically: Replace n...

What Topic Is Next → Next Topic 16.3: Blameless Postmortems, Runbook Automation & DevOps/SRE Interview Scenarios

16.3 Blameless Postmortems, Runbook Automation & DevOps/SRE Interview Scenarios [65 mins • Assignment (100M) • Hands-On Lab (50M) • 2 Mock Sets • 2 AI Vivas]

Subtopics Covered: Security Hygiene & Data Safety Rules for Blameless Postmortems, Runbook Automation & DevOps/SRE Interview Scenarios • Performance Optimization, Complexity & Latency Impact of Blameless Postmortems, Runbook Automation & DevOps/SRE Int... • Live Debugging, Stack • Runnable Code Implementation (YAML) • Hands • Idiomatic DEVO Implementation: immutable GitOps pipelines, canary/blue-green rollouts & SLO error budgets • Linux Kernel, GitHub Actions CI/CD, Terraform, Docker/Kubernetes & Observability Execution & Memory Lifecycle for Blameless Postmortems • Production Hardening & Security: OIDC secretless CI, SBOM/Cosign image signing, Vault & policy-as-code

Purpose: To understand and apply Blameless Postmortems, Runbook Automation & DevOps/SRE Interview Scenarios in real DevOps & Git CI/CD systems — specifically: Walk through broken...

What Topic Is Next → Module 16 Capstone → End-of-Course Live Client Project: ShipFast Automated GitSecOps CI/CD Pipeline, Terraform Cloud & Prometheus Stack

Module 16 Hands-On Capstone Project (100 Marks): Prometheus + Grafana Golden-Signals Alerting Stack & Zero-Downtime Rollback Drill — Architect, implement, test, and document 'Prometheus + Grafana Golden-Signals Alerting Stack & Zero-Downtime Rollback Drill' applying every bullet-wise engineering concept from Module 16 (Pr...

3. LIVE CLIENT PRODUCTION PROJECT & CAREER SCOPE

End-of-Course Live Client Production Project + Official Internship Confirmation Letter

Project: ShipFast Automated GitSecOps CI/CD Pipeline, Terraform Cloud & Prometheus Stack • **Confirmed Role:** DevOps, CI/CD & Cloud Infrastructure Automation Intern

Domain & Production Infra: Cloud-Native SaaS Release & Reliability Engineering Client (Linux + GitHub Actions + Docker + Terraform IaC + AWS + Prometheus & Grafana)

- Built multi-stage GitHub Actions CI/CD pipelines running lint, tests, Trivy container scans, and automated rollouts.
- Provisioned reproducible AWS compute, networking, and security resources using modular Terraform state workflows.
- Configured Nginx reverse-proxy TLS, systemd services, zero-downtime blue/green deployments, and secret injection.
- Deployed Prometheus metrics, Grafana golden-signal dashboards, and Alertmanager incident hooks to production.

Target Job Roles: DevOps Engineer, CI/CD & Release Automation Engineer, Junior Site Reliability Engineer (SRE), Infrastructure & Platform Engineer

Realistic Annual CTC Scope: **Fresher / Associate DevOps Engineer (0–1 Yr):** ₹5.5 LPA – ₹10.0 LPA **Mid-Level DevOps / SRE Engineer (1–3 Yrs):** ₹11.0 LPA – ₹22.5 LPA **Lead DevSecOps / Principal Platform Architect (3–6+ Yrs):** ₹23.0 LPA – ₹45.0+ LPA