

CI/CD with Jenkins & GitHub Actions

Course Duration ~60 Hours · 8 Weeks · Live Online · Lab-Heavy · Real Pipelines, Real Deployments, Real Migrations

Curriculum reviewed: July 2026 — the delivery tooling landscape moves fast; this syllabus is checked against official documentation

[GitHub Actions](#)
[Jenkins](#)
[Supply-Chain Security](#)
[Deployment Strategies](#)
[FinOps for CI/CD](#)
[Migration](#)

In short: Own how software ships. Build pipelines that are fast, secure and affordable on the platform enterprises still run — and the one everything new is built on. This is a course about the delivery *platform itself*: pipeline architecture at scale, self-hosted runner infrastructure, supply-chain security, cost control, deployment strategies, and migrating between tools — distinct from our Git & GitHub course, which teaches collaboration and repository practice with automation as one component. Why teach Jenkins in 2026? Because the money is honest: greenfield projects don't choose Jenkins, but banks, telecoms, manufacturers, government and large services organisations run tens of thousands of Jenkins pipelines that aren't going anywhere soon, and somebody has to keep them running, secure them, and eventually move them — that somebody is well-rewarded. GitHub Actions is the default for new work on GitHub. Knowing both, and moving work between them, is what enterprise employers pay a premium for — and **migration is the flagship**: a six-to-twelve-month funded programme, where conversion tooling translates syntax but not plugin behaviour, shared libraries or custom scripting logic — exactly where the expertise lives. Across 9 modules you build full pipelines on both platforms, harden the supply chain (keyless cloud auth, pinned actions, SBOMs, signing, provenance), run canary deploys with real rollback, cut compute cost with FinOps, and convert a realistic Jenkins pipeline to GitHub Actions documenting what the tooling couldn't carry — closing with a delivery-platform capstone plus a migration plan. GitLab CI and Kubernetes-native options are covered honestly as the right answer in some contexts. The AI question: AI writes pipeline YAML fluently and produces the failure modes that cost real money — leaked secrets, over-permissioned tokens, unpinned actions, poisoned caches, pipelines that pass while testing nothing. In CI/CD the blast radius is production.

Duration	8 weeks · ~60 hours · live + hands-on	Format	Live online · lab-heavy · real pipelines, real deployments, real migrations
Prerequisite	Comfortable with Git and the command line, and able to read code in some language. Container basics help but are taught as needed.	Curriculum	9 modules, from delivery foundations to migration, modernisation and capstone
Certification	Aligned to the GitHub Actions certification; Jenkins and migration skills transfer beyond any single credential	Ideal for	Fresh IT graduates, developers owning their team's pipeline, and engineers maintaining large Jenkins estates

Not the Git & GitHub Course — the Delivery Platform

- **Git & GitHub for Developers** teaches collaboration and repository practice, with automation as one component — for developers who want to work well in a team.
- **CI/CD with Jenkins & GitHub Actions** (this course) teaches the delivery platform itself, on both an enterprise incumbent and the modern default — pipeline architecture at scale, runner infrastructure, supply-chain security, cost control, deployment strategies, and migrating between the two.
- Clear of its DevOps siblings too: GitOps deployment mechanics belong to the Docker & Kubernetes course, infrastructure provisioning to the Terraform course. This course owns the pipeline.

Jenkins or GitHub Actions? Which Should You Choose?

An honest framing. **Jenkins** stays entrenched where there are complex legacy integrations, air-gapped or on-premises requirements, multiple version control systems, heavy investment in shared libraries, or regulatory constraints — highly flexible, and correspondingly expensive to own, its real cost being engineering time rather than infrastructure. **GitHub Actions** is the default for new work on GitHub, with nothing to run and modern permission primitives, though its limits are ecosystem lock-in and cost at scale. And they are not the only options: **GitLab CI and Kubernetes-native pipelines exist and are the right answer in some contexts**. The course teaches the decision, not a dogma.

Migration Is the Flagship

Moving an estate from one platform to the other is a six-to-twelve-month funded programme in a mid-sized organisation, not a weekend task. Conversion tooling handles the mechanical translation of pipeline syntax — but it does **not** carry over plugin behaviour, shared libraries or custom scripting logic, which is precisely where the work and the expertise live. You'll learn assessment, pilot, phased cutover, and running both platforms in parallel during transition — and convert a realistic Jenkins pipeline with shared-library dependencies to GitHub Actions, documenting exactly what the tooling could not translate.

Why Learn This When AI Writes Pipeline YAML?

AI writes pipeline YAML fluently — and produces the failure modes that cost real money: secrets exposed in logs, over-permissioned tokens, unpinned third-party actions pulling in whatever ships next, caches that silently poison builds, and pipelines that pass while testing nothing. In CI/CD the blast radius is production. The value is judgment about security, cost and change safety. You'll use AI tooling as a reviewed accelerator and learn the failure patterns generated pipeline code repeats.

Course Highlights

- Every module leads with what you build, operate or migrate — not what you "learn about"
- Both platforms in depth — GitHub Actions and Jenkins, taught as production systems you operate
- Migration as the flagship — the six-to-twelve-month programme, taught as assessment, pilot and phased cutover
- Supply-chain security: keyless cloud auth, pinned actions, SBOMs, signing, provenance
- Deployment strategies that hold under pressure — canary, blue/green, feature flags, real rollback
- FinOps for CI/CD — where compute minutes actually go and how to cut them without slowing teams
- Self-hosted autoscaling runner fleets and the reliability of the delivery system itself
- AI tooling as an accelerator you review — because in CI/CD the blast radius is production

Hands-on Projects & Portfolio

- **Full CI Pipeline on Actions** — build, test matrix, caching, quality gates and artefact publishing
- **Equivalent Jenkins Pipeline** — a declarative Jenkinsfile backed by a shared library
- **Multi-Environment Deployment** — approval gates, canary release and automated rollback
- **Keyless Cloud Authentication** — replace long-lived credentials with short-lived identity in both platforms
- **Supply-Chain Hardening** — pinned dependencies, scanning, SBOM, signed artefacts and provenance
- **Autoscaling Runner Fleet** — self-hosted, with cost and queue-time measurement before and after
- **Pipeline Performance Rescue** — cut a slow build's duration substantially, with evidence
- **Migration Exercise** — convert a realistic Jenkins pipeline with shared-library dependencies to GitHub Actions, and document what the tooling could not translate
- **Capstone** — complete delivery platform plus a written migration plan for a legacy estate

Tools & Stack

Jenkins · GitHub Actions · Groovy & Jenkinsfile · Jenkins Configuration as Code · Docker · Kubernetes · Helm · Argo CD · Terraform · SonarQube · Trivy · Syft · Cosign · OIDC · HashiCorp Vault · Nexus or Artifactory · GitHub Packages · OpenTelemetry · Prometheus · Grafana · AWS · Azure · Git

Certification Alignment

- Modules 2, 4–8 map to the **GitHub Actions certification** scope — authoring and maintaining workflows and actions, consuming workflows, and managing Actions for the enterprise
- The Jenkins, deployment, supply-chain and migration skills transfer beyond any single credential — this course is deliberately broader than one exam
- Presented as preparation and alignment — no official partnership or accreditation implied, and no exam outcome guaranteed

Career Outcomes

Delivery engineering is one of the most consistently demanded DevOps skills, because every organisation ships software and almost none of them ship it as well as they'd like. The dual-platform framing is deliberate: enterprise environments — particularly large services organisations and global capability centres — run substantial Jenkins estates alongside newer GitHub-based workflows, so engineers who can operate both and move work between them are unusually valuable. Pipeline security and cost control are the fastest-growing parts of the role. This course targets:

- DevOps Engineer · CI/CD Engineer · Build & Release Engineer · Platform Engineer
- Site Reliability Engineer · Automation Engineer · DevSecOps Engineer · Developer who owns their team's delivery process

Who This Course Is For

- Final-year students and fresh IT graduates targeting DevOps or build/release roles
- Developers who want to own their team's pipeline rather than wait on someone else's
- **Engineers maintaining large Jenkins estates who need modern practice and a migration path** — funded, urgent work
- QA and automation engineers moving into delivery engineering
- System administrators moving into DevOps
- Platform and SRE teams responsible for delivery reliability, security and cost

Module 1: Delivery Engineering Foundations

6 Hours

YOU BUILD

- A diagnosis of a deliberately slow release process — measured against the delivery metrics that matter — with the bottleneck identified before a single tool is touched

TOPICS

- What continuous integration, delivery and deployment actually mean and how they differ; trunk-based development and its relationship to pipeline design
- The delivery metrics that matter — lead time, deployment frequency, change failure rate, restore time
- Pipeline anatomy; artefacts and promotion; diagnosing a slow release process before touching a tool

Assessment: Quiz + Lab

Module 2: GitHub Actions in Depth

7 Hours

YOU SHIP

- A reusable, organisation-wide workflow architecture — composite and reusable workflows, environments and approval gates, concurrency control — that scales past a single repository

TOPICS

- Workflow and job architecture; events and triggers; runners; matrix strategies; caching that actually helps; artefacts
- Reusable and composite workflows; environments and approval gates; concurrency control; service containers
- Debugging failing workflows; structuring workflows across a whole organisation rather than one repository

Assessment: Quiz + Lab

Module 3: Jenkins in Depth

8 Hours

YOU OPERATE

- A Jenkins controller run as production infrastructure — configuration as code, a shared library powering a declarative Jenkinsfile, and dynamic Kubernetes-backed agents provisioned on demand

TOPICS

- Controller and agent architecture; declarative and scripted pipelines; the Jenkinsfile; shared libraries as the mechanism serious Jenkins shops depend on
- The plugin ecosystem and its maintenance burden; credentials management; configuration as code

- Distributed builds and dynamic agents including containerised and Kubernetes-backed ones; operating the controller itself as production infrastructure

Assessment: Quiz + Lab

Module 4: Build, Test & Quality Gates

7 Hours

YOU SHIP

- A parallelised, cache-accelerated build with coverage and static-analysis gates that block a bad change — and a monorepo selective-build strategy that only tests what changed

TOPICS

- Building and testing across common language ecosystems; dependency caching and build reproducibility; parallelisation and test splitting
- Flaky test management; coverage and static analysis gates; quality thresholds that block rather than inform
- Monorepo and selective build strategies; keeping pipeline feedback fast enough that people actually wait for it

Assessment: Quiz + Lab

Module 5: Artefacts, Registries & Release Automation

6 Hours

YOU SHIP

- A build-once-promote-everywhere release flow — immutable versioned artefacts, automated semantic versioning and generated release notes — no rebuild per environment

TOPICS

- Versioning strategy; build once and promote rather than rebuild per environment; container image and package registries
- Artefact retention and cleanup; changelog and release note generation; semantic versioning automation
- Immutable release artefacts

Assessment: Quiz + Lab

Module 6: Deployment Strategies

7 Hours

YOU SHIP

- A multi-target deployment with a canary release, a feature flag decoupling deploy from release, a database migration in the pipeline, and a rollback that works under pressure

TOPICS

- Deploying to virtual machines, containers, Kubernetes and serverless targets; blue/green, canary and rolling releases
- Feature flags decoupling deployment from release; database migrations in a pipeline; approval workflows and change management for regulated environments
- Rollback that works under pressure; smoke tests and automated verification after deploy

Assessment: Quiz + Lab

Module 7: Pipeline Security & Software Supply Chain

8 Hours

YOU SHIP

- A hardened pipeline: keyless cloud authentication with short-lived identity, least-privilege job permissions, pinned and vetted actions, scanning, an SBOM, and signed artefacts with provenance

TOPICS

- Secrets management and why they leak; keyless authentication to cloud providers using short-lived identity rather than stored credentials; least-privilege job permissions
- Pinning and vetting third-party actions and plugins; protecting self-hosted runners from untrusted code; dependency and container scanning in the pipeline
- SBOM generation; artefact signing and provenance attestation; audit trails for compliance

Assessment: Quiz + Lab

Module 8: Scale, Reliability & Cost

6 Hours

YOU SHIP

- An autoscaling ephemeral runner fleet with build-cache infrastructure and pipeline observability — plus a FinOps exercise cutting compute minutes with measured before/after

TOPICS

- Self-hosted and autoscaling runner fleets; ephemeral agents; queue times and capacity planning; build cache infrastructure
- Observability for pipelines including tracing a slow build; reliability of the delivery system itself
- FinOps for CI/CD — where compute minutes actually go and how to cut them without slowing teams down

Assessment: Quiz + Lab**Module 9: Migration, Modernisation & Capstone**

5 Hours

YOU MIGRATE

- The flagship: a realistic Jenkins pipeline with shared-library dependencies converted to GitHub Actions — with a written record of what conversion tooling could not translate — plus your capstone delivery platform and a migration plan for a legacy estate

TOPICS

- Assessing an existing pipeline estate; conversion tooling and what it genuinely cannot translate; handling shared libraries and custom scripting logic
- Phased cutover and running both platforms in parallel; decommissioning safely; honest comparison with GitLab CI and Kubernetes-native alternatives
- Capstone design, build, secure, operate and defense

Assessment: Capstone Project + Migration Plan