JUPITER PRIME
Consulting • Training • Advisory
جوبيتر برايم للاستشارات والتدريب

COURSE BROCHURE

Continuous Integration and Delivery Pipelines Training Course

21 – 25 December 2026
New York

Continuous Integration and Delivery Pipelines Training Course

Ref: 2816_108001 **Date:** 21 – 25 December 2026 **Location:** New York **Fees:** 9500 GBP

Continuous Integration and Delivery Pipelines is a 1-week training course for software engineers, release leads and platform specialists who build and run automated release paths, and it ends with a pipeline blueprint with quality gates. Organisations now ship changes weekly or daily, so a slow or fragile release path turns directly into outages and missed commitments. Jupiter Prime Training presents Continuous Integration and Delivery Pipelines as a release discipline rather than a scripting exercise.

Why Continuous Integration and Delivery Pipelines Matter Now

Three forces shape release engineering today: product owners who expect small, frequent changes; security and audit functions that want every build traceable from commit to production; and operations staff who carry the cost when a deployment fails at night. Organisations that handle it well merge daily and roll back calmly; those that handle it badly batch changes into risky releases and long code freezes. The course rests on widely used engineering practice: trunk-based development, the test pyramid, immutable artefacts and progressive deployment.

Who Gains Most from Continuous Integration and Delivery Pipelines

- Engineers who write build definitions and merge rules
- Release coordinators who decide when a change is promoted
- Platform specialists who run build agents and artefact stores
- Quality engineers who decide which checks block a merge
- Operations staff who own rollback and recovery steps

A product group that sends a developer, a tester and an operations engineer together gains most, sharing one view of the release path.

Continuous Integration and Delivery Pipelines Capabilities

Participants move from scattered build scripts to a staged release path where every change is built once and promoted with evidence.

- Design a branching and merge policy that keeps the main line releasable — engineering leads
- Build a staged pipeline with fast feedback before slower integration suites — developers
- Specify quality gates with clear pass criteria for each promotion step — release coordinators
- Weigh blue-green, canary and rolling deployment for a given service — operations owners
- Test rollback and roll-forward paths before they are needed in an incident — on-call staff
- Communicate lead time, change failure rate and recovery time to sponsors — product owners

Key Continuous Integration and Delivery Pipelines Terms

- Trunk-based development — merging small changes into one main line at least daily, without long-lived branches.
- Build artefact — the versioned, unchangeable output of a build, promoted unchanged through every environment.
- Quality gate — a pipeline checkpoint with explicit pass criteria that blocks promotion when unmet.
- Test pyramid — many fast unit tests at the base, fewer integration tests above, few end-to-end tests at the top.
- Canary release — sending a small share of live traffic to a new version and watching health signals before widening.
- Change failure rate — the share of production deployments causing an incident, rollback or hotfix.

Continuous Integration and Delivery Pipelines Day by Day

Day 1: The Release Path and Its Forces

- Value Stream Mapping of Commit to Production — Where Waiting Time Hides
- Delivery Performance Measures — Lead Time, Failure Rate and Recovery
- Release Cadence Options — Batch Size Against Customer Expectations
- Stakeholder Map for Releases — Product, Security, Audit and Operations
- Current Pipeline Assessment — Baseline for the Blueprint

Day 2: Version Control and Integration

- Trunk-Based Development and Short-Lived Branches — Merge Conflict Exposure
- Pull Request Rules and Peer Review — Speed Against Scrutiny
- Feature Flags for Unfinished Work — Decoupling Merge From Release
- Reproducible Builds and Dependency Pinning — Trust in Every Output
- Build Caching and Parallel Stages — Feedback Time for Developers

Day 3: Automated Testing Stages and Gates

- Test Pyramid Design — Balance of Unit, Integration and End-to-End Checks
- Contract Tests Between Services — Independent Release of Components
- Static Analysis and Dependency Scanning — Security Findings That Block Merges
- Flaky Test Quarantine Policy — Confidence in Red and Green Results
- Quality Gate Criteria — Evidence Required at Each Promotion

Day 4: Artefacts, Environments and Deployment

- Immutable Artefact Versioning and Storage — Build Once, Promote Everywhere
- Environment Configuration and Secrets Handling — Drift Between Stages
- Blue-Green and Rolling Deployment — Downtime Tolerance per Service
- Canary Analysis With Health Signals — Automatic Halt or Widening
- Database Schema Migration Patterns — Backward Compatibility During Release

Day 5: Practising the Pipeline Blueprint

- Rollback and Roll-Forward Drills — Recovery Time Under Pressure
- Applying Branching Policy to a Live Codebase — Merge Rules Agreed
- Applying Gate Criteria to Sample Releases — Promotion Decisions Tested
- Applying Deployment Strategy Choices — Service-by-Service Release Plan
- Pipeline Blueprint With Quality Gates — Final Design and Owners

Applying Continuous Integration and Delivery Pipelines After the Course

- Run a commit-to-production value stream mapping session for one product and publish the three longest waiting steps with an owner for each.
- Introduce one quality gate with written pass criteria on the main line and track change failure rate before and after for six weeks.

Continuous Integration and Delivery Pipelines Questions FAQ

What are continuous integration and delivery pipelines and why do they matter now?

They are automated paths that build, test and release every code change in stages. They matter now because organisations release far more often, and a reliable pipeline keeps frequent change from turning into frequent outages, in banking apps and logistics tracking alike.

Which roles gain most from continuous integration and delivery pipelines training?

Developers, release coordinators, quality engineers, platform specialists and operations staff gain most. Each owns part of the release path and gains a shared language for branching, gates and deployment.

How does a continuous integration and delivery pipelines course differ from a general software testing course?

It covers the whole path from commit to production, not only test design. Testing is one stage among branching, artefacts, environments, deployment and rollback.

What do you bring back from continuous integration and delivery pipelines training?

You bring back a pipeline blueprint for your own release process. It sets out branching rules, test stages, gate criteria, deployment strategy and rollback steps, with owners.

Continuous Integration and Delivery Pipelines: The Way Forward

The pipeline blueprint with quality gates first informs which gate to automate next on the main line. Engineering leads and release coordinators use it to agree promotion rules, and operations staff use it to plan rollback drills. After the first quarter of use, revisit the balance between gate strictness and feedback speed, since teams often tighten checks until developers start working around them.