+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 26/06/2025

Sign up

LitmusChaos training

3 days (21 hours)

Presentation

Our LitmusChaos training course will enable you to introduce failures into a system in a controlled way, in order to assess its resilience and improve its robustness. With LitmusChaos, DevOps teams can validate that their cloud-native applications are properly resilient to failures, network disruptions or system stresses.

Our LitmusChaos training course will teach you how to inject, schedule and analyze chaos tests in Kubernetes. You'll discover how to use key components like the ChaosEngine, ChaosHub, and ChaosCenter portal, and how to orchestrate simple or advanced disruption scenarios.

By the end of this course, you'll be able to design and run chaos scenarios on your Kubernetes applications, derive actionable metrics from them, and automate your resilience testing.

Like all our training courses, it will run on the latest version of the tool: Litmus 3.19

Objectives

- Understand the fundamental principles of chaos engineering and their relevance to application resilience
- Install and configure LitmusChaos on a single-cluster Kubernetes cluster (local or cloud)
- Deploy targeted chaos experiments using Kubernetes resources (ChaosEngine, ChaosExperiment, ChaosResult)
- Design, run and analyze disruption scenarios (CPU, memory, network, pod restart, etc.)
- Orchestrate multiple experiments via Chaos Workflows
- Integrate chaos tests into a CI/CD pipeline and initiate a Continuous Chaos approach

Target audience

- DevOps engineers
- SRE
- Back-end or full-stack developers

Prerequisites

- Master the fundamentals of Kubernetes (application deployment, pod/service management, use of kubectl).
- Basic experience in DevOps or cloud-native systems administration.

Technical prerequisites

- A workstation capable of running a local Kubernetes cluster, or access to a remote cluster.

Our LitmusChaos training program

Introduction to chaos engineering

- Definition and objectives of chaos engineering
- Application resilience and fault tolerance
- History and fundamental principles of chaos (steady-state, hypothesis, validation)
- Best practices and security framework for chaos testing
- Market tools (Gremlin, Chaos Mesh, LitmusChaos)
- Positioning LitmusChaos in the CNCF ecosystem

LitmusChaos architecture and components

- LitmusChaos overview (open source, CNCF, mono-cluster)
- Control plan vs. execution plan
- Custom Resources (ChaosEngine, ChaosExperiment, ChaosResult)
- Chaos Operator and ChaosCenter
- Introducing ChaosHub: the experience library
- Operation and life cycle of a chaos experiment

Setting up the LitmusChaos environment

- Preparing the Kubernetes environment
- Installing LitmusChaos (via Helm or YAML manifests)
- Verification of installed components
- Deploy a test application (e.g. Podtato-head, Nginx, etc.)
- Configure necessary permissions and annotations
- Access to the ChaosCenter portal (if used)

- Practical workshop: Installation of LitmusChaos and validation via a first simple test (pod-delete)

First chaos test with ChaosEngine

- Structure and role of a ChaosEngine
- Referencing a ChaosExperiment in a ChaosEngine
- Target definition via labels/selectors
- Test parameters (duration, strength, randomization, etc.)
- Run experiment and read ChaosResult
- Observe effects on the application

Classic chaos experiments

- Types of chaos supported by Litmus (CPU Hog, Memory Hog, Network Delay...)
- Pod, container or node failures
- Network degradation: latency, packet loss
- Resource stress: CPU, memory, disk overloads
- Dependent service failures (e.g. database, DNS)
- Choosing the right test according to the nature of the target application
- Practical workshop: Creating and running a customized ChaosEngine (e.g. CPU Hog on a microservice)

Observation and interpretation of results

- Analysis of execution logs
- Using kubectl to diagnose effects
- Consultation of ChaosResult objects
- Application monitoring during the experiment (Prometheus, Grafana, logs)
- Key resilience metrics to observe
- Debriefing an experiment to learn from it

Orchestration with Chaos Workflows

- Introduction to Chaos Workflows (test chaining)
- Visual construction via ChaosCenter or YAML definition
- Dependencies and conditional chains
- Example of a two-step workflow (pod-delete + memory-stress)
- Workflow execution and monitoring
- Overall workflow resilience score
- Practical workshop: Designing and executing a multi-stage Chaos Workflow

Case studies and chaos planning

- Case study: chaos engineering on an e-commerce application
- Identifying weak points for testing
- Progressive selection of chaos experiments

- Organization of test campaigns (manual, planned, automatic)
- Test ramp-up strategy in a real-life context
- Establishing a team chaos engineering plan

Integrating LitmusChaos into CI/CD

- Objectives of chaos in the CI/CD pipeline
- LitmusChaos and GitHub Actions, GitLab CI, Jenkins
- Automatic triggering of chaos tests
- Validation and resilience scoring in the pipeline
- YAML configuration examples for CI/CD
- Towards continuous chaos with GitOps and ArgoCD

Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced IT technology, or to acquire specific business knowledge or modern methods.

Positioning on entry to training

Positioning at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, the learner receives a self-assessment questionnaire enabling us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the forthcoming course, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Certification

A certificate will be awarded to each trainee who has completed the entire course.