

Advanced Apache NiFi

Clustering, performance, extension development and operational security

FA-0580 · 2 days

Data & Analytics · DevOps, Cloud & Infrastructure · Software Development

Review advanced NiFi architecture and build selected clustering, scripting and extension examples in a prepared lab environment.

Introduction

This two-day advanced workshop is intended for experienced NiFi users. It examines architecture, clustering, performance, scripting, custom extensions, security and troubleshooting through selected exercises and prepared deployment examples.

Participants practise configuration and review patterns rather than constructing and certifying a complete highly available production deployment. Labs use a chosen supported NiFi release with matching Java and extension dependencies; capabilities and configuration details are checked against that release.

Learning outcomes

The course teaches participants to:

- Explain NiFi architecture and the role of clustering, load balancing and node recovery.
- Investigate performance bottlenecks and improve a selected dataflow.
- Use a supported scripting processor for a bounded transformation.
- Build and test a small custom Java processor and describe other extension points.
- Review authentication, authorisation, secure transfers and sensitive-property handling.
- Diagnose selected operational problems and document deployment or upgrade checks.

Prerequisites

- Hands-on NiFi experience, including FlowFiles, Expression Language, processors and controller services.
- Working Java knowledge for custom processor exercises.
- Linux and networking fundamentals.
- A prepared NiFi lab environment with compatible Java and build dependencies; cluster demonstrations need sufficient lab resources.
- Review the chosen release's requirements before class.

Training outline

1. Day 1 — Architecture, Clustering and Performance

Module 1 — Internal Architecture

- Overview of NiFi's internal components.
- Understanding the NiFi dataflow hierarchy.

Module 2 — Clustering and Availability

- Principles of NiFi clustering.
- Review and configure a prepared NiFi cluster in the learning environment.
- Load balancing in NiFi.
- Handling node failures and recovery strategies.
- Review a prepared cluster and a controlled node-failure scenario.
- Distinguish clustering configuration from an application-specific availability guarantee.

Module 3 — Performance Investigation

- Profiling and monitoring NiFi performance.
- Best practices for optimizing dataflows.
- Efficient resource utilization.
- Profile a selected dataflow, compare bottlenecks and record the effect of a bounded change.

2. Day 2 — Extensions, Security and Troubleshooting

Module 4 — Supported Scripting Patterns

- Use ExecuteScript with a supported engine, such as Groovy, for FlowFile transformations.
- Compare ExecuteGroovyScript where its language-specific integration is useful.
- Verify processor and engine availability for the selected release instead of assuming legacy Python or JavaScript engines are present.
- Introduce the separate Python Processor API and its current beta status; do not present it as a production-ready substitute for every script.

Module 5 — Custom Extensions

- Introduction to NiFi's processor development framework.
- Build, test and demonstrate deployment of a selected custom processor.
- Developing custom Controller Services and Reporting Tasks.
- Build and test a small prepared Java processor example; review packaging and deployment requirements.
- Treat controller services and reporting tasks as extension overviews rather than additional full builds.

Module 6 — Security Configuration

- Deep dive into multi-tenant authorization.
- Configuring and managing secure data transfers.
- Handling sensitive properties and encrypted content.
- Setting up NiFi with external authentication providers.

Module 7 — Troubleshooting and Operational Review

- Diagnosing common issues in advanced NiFi deployments.
- Advanced logging and monitoring techniques.
- Maintaining NiFi in production environments.
- Diagnose selected lab failures using logs and performance evidence.
- Review compatibility, deployment and upgrade checks for scripts and custom extensions.
- Summarise remaining production validation and support requirements.

