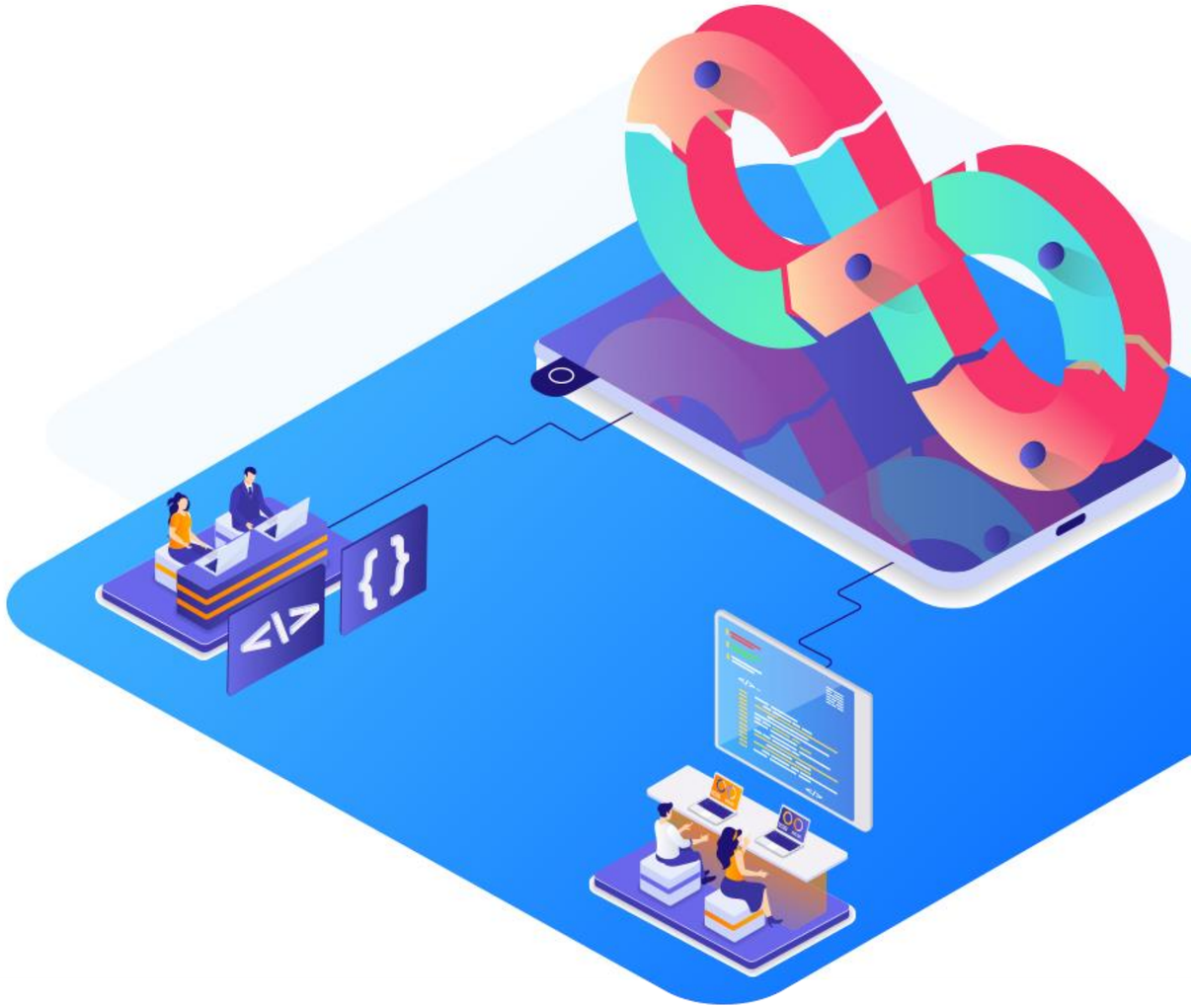
DevOps Foundations: Version Control and CI/CD with Jenkins



Jenkins Build Tools and Build Triggers

Learning Objectives

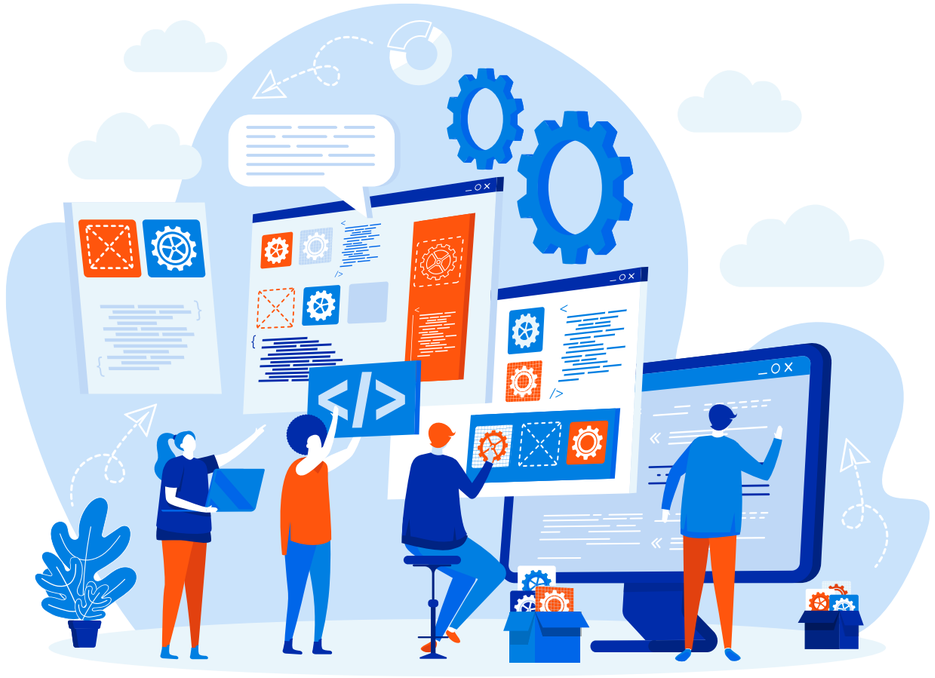
By the end of this lesson, you will be able to:

- Outline how Jenkins integrates with various build tools to automate build processes
- Utilize the Maven build tool in Jenkins to automate the compilation, testing, and packaging of the Java projects efficiently
- Apply knowledge of unit testing concepts to create effective test suites in Jenkins

Build Tools Integration

Jenkins Build Tools

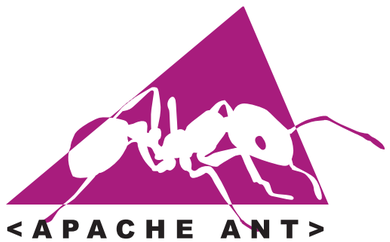
It automates the conversion of source code into deployable formats, enhancing SDLC phases and serving as a vital component in software development.



Build tools like Ant, Maven, and Gradle can be used to perform build automation and IDE integration.

Popular Build Tools

Below are some popular build tools available in Jenkins:



Apache Ant

A Java tool that automates building software, simplifying tasks like compiling, packaging, and deploying, primarily using XML configuration.

Maven

It is a popular build automation tool that has found its niche in Java development. It controls construction via XML-based Project Object Model (POM) files

Popular Build Tools

Gradle

A flexible and fast open-source tool for automating builds, it uses Groovy-based scripts to define tasks, focusing on adaptable and efficient development processes.

Travis CI

It is the preferred cloud solution for automating build and testing on GitHub, due to its easy-to-use interface and cloud-based setup.

Assisted Practice

Setting up Git configuration in Jenkins job **Duration: 15 Min.**

Problem statement:

You have been assigned a task to set up Git configuration in Jenkins for streamlined version control and automated build processes.

Outcome:

By the end of this demo, you will be able to set up Git configuration in Jenkins for efficient version control and automated build processes.

Note: Refer to the demo document for detailed steps:

ASSISTED PRACTICE

Assisted Practice: Guidelines

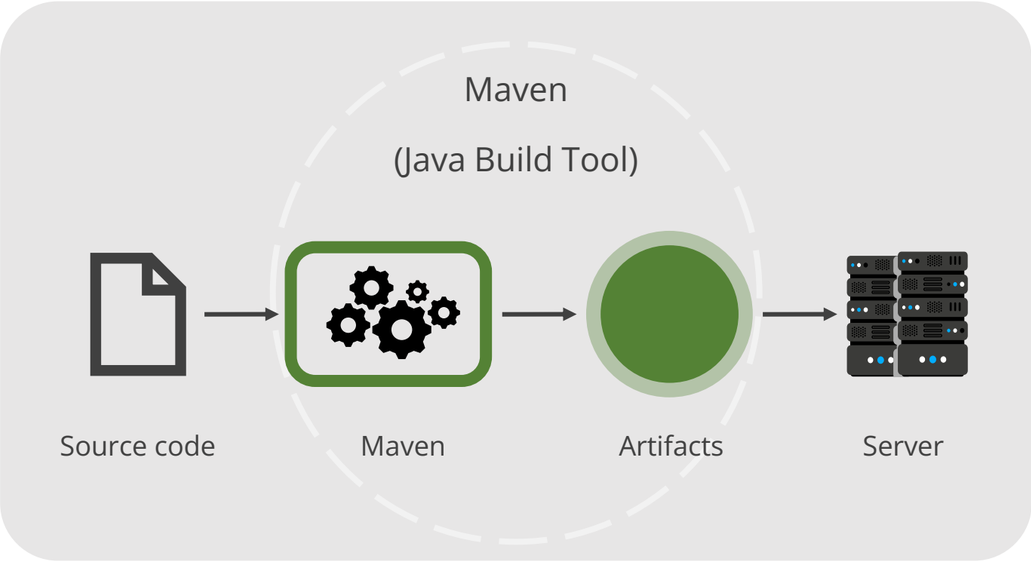
Steps to be followed:

1. Create new credentials in Jenkins for Git configuration
2. Create a Jenkins freestyle project
3. Configure Git in Jenkins

ASSISTED PRACTICE

Maven Build Tool

It is an open-source build tool for Java projects that simplifies the build process, ensuring consistency and streamlined project management.



It supports building and managing projects in various languages like C#, Ruby, and Scala. It's based on a Project Object Model (POM) for managing builds.

Maven Build Tool

Some of the key functionalities of Maven are:

Project structure and setup:
Ensures a consistent project structure and simplifies project navigation

Dependency management:
Downloads and manages external libraries, ensuring compatibility and eliminating manual downloads for project

Build automation:
Automates tasks like compiling, packaging, and testing, Maven frees developers and ensures consistent builds across environments

Maven Build Tool

Some of the key functionalities of Maven are:

Reporting and documentation:
Generates reports on project aspects and automates project documentation based on metadata

Extensibility:
Enables custom plugin creation in Java or scripting languages for specific project needs

Integration with other tools:
Integrates with CI systems for automated builds and deployments alongside other development tools.

Assisted Practice

Integrating Maven with Jenkins **Duration: 10 Min.**

Problem statement:

You have been assigned a task to install the Maven plugin in Jenkins for smooth integration and automation of Maven-based build processes within the Jenkins environment.

Outcome:

By the end of this demo, you will be able to install the Maven plugin in Jenkins for seamless integration and automation of Maven-based build processes.

Note: Refer to the demo document for detailed steps:

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Install the Maven plugin
2. Set up Global Tool Configuration
3. Fork a sample repository
4. Integrate Maven with Jenkins

ASSISTED PRACTICE

Build Automation Using Jenkins

Build automation refers to the process of automating the repetitive tasks involved in building software.

Build Automation Using Jenkins

Below are some of the tasks:

Retrieving code from VCS: Retrieves the source code from version control systems such as Git or SVN to begin the build process

Compiling code: Compiles the source code into executable files or libraries based on the project configuration and dependencies

Running unit tests: Executes unit tests to ensure the code functions correctly and meets the specified requirements

Packaging the final application: Packages the compiled code, resources, and dependencies into a distributable format like JAR or WAR for deployment

Build Automation Using Jenkins: Benefits

Some advantages of using Jenkins for build automation are:

- Frees developers from repetitive tasks
- Ensures consistent builds across environments
- Improves build quality through automated testing
- Speeds up builds through parallelization
- Enhances collaboration with a central build management hub

Assisted Practice

Setting up Maven build job in Jenkins **Duration: 10 Min.**

Problem statement:

You have been assigned a task to set up a Maven build job in Jenkins for automating the build process, enabling continuous integration to enhance the software development lifecycle.

Outcome:

By the end of this demo, you will be able to set up a Maven build job in Jenkins to automate the build process and enabling continuous integration in your software development lifecycle.

Note: Refer to the demo document for detailed steps:

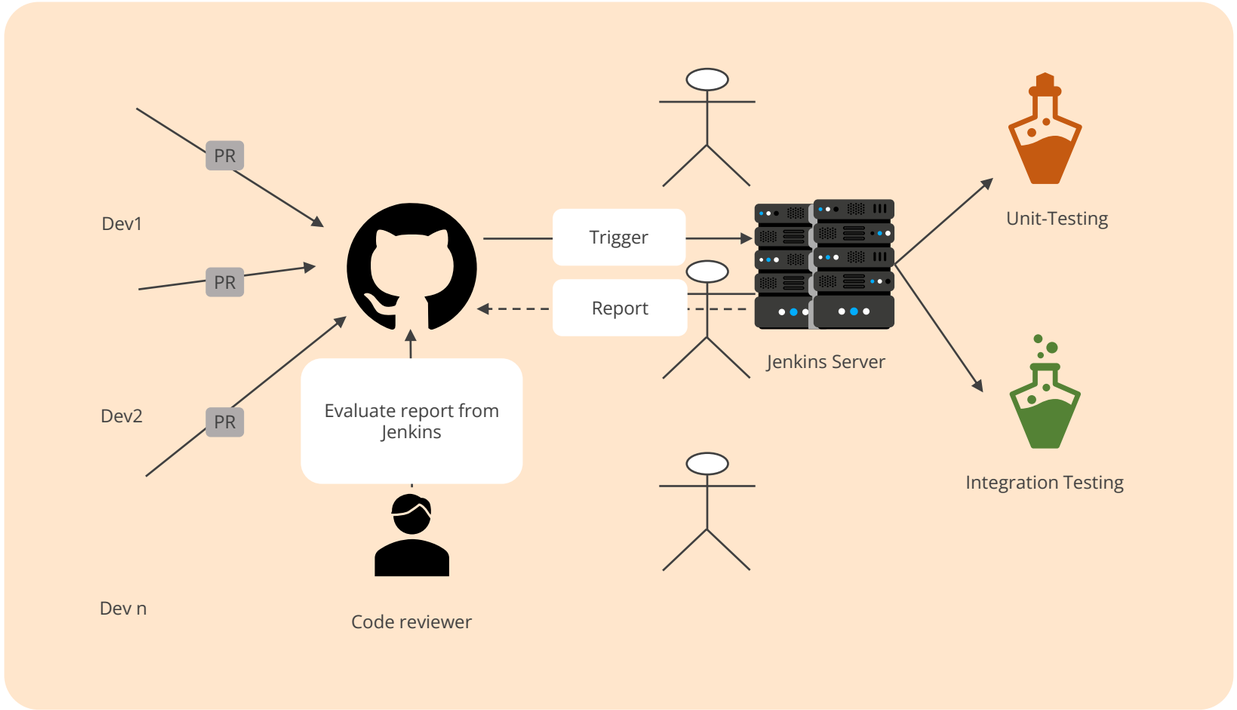
Assisted Practice: Guidelines

Steps to be followed:

1. Log in to Jenkins CI tool and configure Maven freestyle job

Configuring Unit Test Cases

Jenkins plugins automate unit testing and improve code quality with clear pass or fail results.



Configure Jenkins by installing **Testing** plugins, specifying the test file location, and enabling automatic execution of unit tests.

Configuring Unit Test Cases

Key steps to configure Jenkins for running unit tests:

1. Install necessary plugins:

- Install unit test framework plugins in Jenkins
- Choose plugins based on the test framework (for example: JUnit for Java tests, NUnit for .NET tests)
- Interpret and present the test results effectively by utilizing these plugins and allowing Jenkins to act as translators

Configuring Unit Test Cases

2. Configure build steps:

- Configure how Jenkins executes tests after installing the right plugins

- Setting up build steps within the project configuration typically involves the below processes:
  - **Invoke Ant**: For Java projects, use Ant build scripts to run tests
  - **Build Maven**: For Maven projects, leverage Maven goals to execute tests

Configuring Unit Test Cases

3. Run tests and view results:

- Execute unit tests automatically with the configured plugins and build steps
- Trigger a build and let Jenkins handle the execution of unit tests automatically
- Use plugins to show clear pass or fail results and errors, providing insights into code health and status

Generating Test Reports in Jenkins

It involves automatically creating detailed summaries and analyses of test results following the execution of automated tests as part of a build process.

These reports offer insights into test outcomes (passed or failed), indicating the number of tests run along with any encountered errors or issues during testing.

Generating Test Reports in Jenkins

Key steps in generating test reports Jenkins include:

Generating Test Reports in Jenkins

Benefits of test reports:

Detailed analysis:
Identify failing tests by analyzing individual test cases using reports

Improved collaboration:
Share testing results effectively with the team using clear and shareable reports

Tracked trends:
Monitor reports over time to track code quality trends and identify areas for improvement

Assisted Practice

Publishing test cases reports in Jenkins **Duration: 15 Min.**

Problem statement:

You have been assigned a task to publish test case reports in Jenkins to enhance transparency, accountability, and quality in the software development process

Outcome:

By the end of this demo, you will be able to configure Jenkins to publish test case reports, promoting transparency, accountability, and quality in your software development process.

Note: Refer to the demo document for detailed steps:

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed:

1. Log in to the Jenkins CI tool and publish test case reports

Quick Check

Imagine working in a company where you need to track code quality trends and identify areas for improvement over time. Which feature of Jenkins should you utilize?

A. Building automation using Jenkins

B. Configuring unit test cases

C. Generating test reports in Jenkins

D. Configuring Git in Jenkins job

Key Takeaways

- Build tools automates the conversion of source code into deployable formats enhancing SDLC phases.
- Git configurations in the Jenkins job focus on specifying how Jenkins interacts with the Git repository to retrieve code for the build process.
- Maven build tool is an open-source build tool for Java projects that centralizes and simplifies the build process.
- Configuring unit test cases involves setting up automated tests that verify the functionality of individual units of code.
- The unit tests are typically used within a continuous integration (CI) or development environment, typically using testing frameworks like JUnit for Java.

Sending Test Reports Using Email Notification

Duration: 20 Min.

Project agenda: To set up a Jenkins pipeline to compile code, run JUnit tests, and distribute test reports via email notifications

Description: Imagine a development team using Jenkins for automated email notifications of test reports, improving code quality and speed. Upon code commits, Jenkins triggers a pipeline that compiles, tests, and emails test results via an SMTP setup. This quick feedback loop is vital for agile practices, ensuring that developers immediately identify and address issues.

LESSON-END PROJECT

Sending Test Reports Using Email Notification

Duration: 20 Min.

Perform the following:

1. Create an app password for SMTP configuration
2. Configure SMTP configurations in Jenkins
3. Configure Maven in Jenkins

Expected deliverables: A fully configured Jenkins pipeline that compiles code, executes JUnit tests, and sends detailed test reports via email upon each build's completion

LESSON-END PROJECT

Thank You