CI/CD Pipelines with Jenkins

Continuous Integration and Continuous Delivery with Jenkins

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#Dev-Ops

Table of Contents

Overview

Jenkins

- Jenkins
  - Open-source automation server
- Used for facilitating CI/CD
- Supports various platforms and languages
- Large ecosystem of plugins
  - Allows users to integrate it with various tools and technologies
- Simplifies CI/CD pipeline

Key Features and Benefits

- **Web-based interface**
  - Easier configuration and management of CI/CD
  - No need for extensive scripting
- Extensible
  - Through its plugin architecture
    - Providing a wide range of options for task completion
- Supports **distributed** builds
  - Allows **multiple** build agents to work in parallel
  - Optimizes resource utilization
  - Speeds up development process

Pipeline Framework

Jenkinsfile Pipelines

Software University

- **Set of plugins** that support the integration and implementation of CD pipelines
- Provides a **domain-specific language** (DSL) for **defining steps** involved in the software delivery process
  - Automates the entire process of software delivery
- Ensures that software is **always** in a **releasable state** through its lifecycle

Pipeline Components

- **Stages**
  - High-level phases that organize the main activities in a pipeline
    - Build, test, deploy, etc.
- **Steps**
  - Concrete tasks within each stage
- **Nodes**
  - Define the system or agent where the pipeline or a specific stage will run
- **Agents**
  - Direct the pipeline where to run

Pipeline as Code

Software University

- Practice that treats the continuous integration, continuous delivery and continuous deployment as **part** of the **application code**
- Enables **collaboration** on design and changes
- Facilitates tracking **changes** and reviewing previous versions
- Improves transparency
  - All team members can see the pipeline's **logic** and understand the delivery **process**

Jenkinsfile

Software University

- Core component representing the "**Pipeline as Code**" philosophy
- Defines the **pipeline configuration as code**
- Outlines the **stages**, **steps** and **actions** that Jenkins will execute during the build, test and deploy processes
- Usually, placed at the **root** of the project repository
  - Allows revision and versioning
- Two main **types** of syntax, written in **Groovy** (optionally typed and dynamic language)
  - Choice between the two types depends on project's complexity and team's preferences

Declarative Syntax

- **Newer** and **simplified** way of defining the pipelines
- Aims to provide more readable way to define pipeline configuration
  - Easy to read and write
- **Pre-defined** structure

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
            // Commands to build
        }
    }
    post {
        always {
            // actions to perform after
the pipeline runs
        }
    }
}
```

Scripted Syntax

- Traditional way of scripting the Jenkinsfile
- Based on Groovy
  - Provides **more flexibility** and control
- Complete control over the script
- Allows **more complex logic**

```
node {
    stage('Build') {
        // Commands to build
    }
    stage(Test') {
        // Commands to build
    }
    stage('Deploy') {
        // Commands to build
    }
}
```

Key Concepts

Events

- Start a Jenkins **job** or **pipeline**
- Executed by external **triggers**
  - Source code changes
    - Commit or merge to a version control system, e.g., Git
  - Manual initiation
    - Started through the Jenkins UI
  - Upstream or downstream triggers
    - Completion of another job
  - Scheduled event

Workflows

- High-level **definition** of the **entire process** for deployment
- Described in a **Jenkinsfile**
  - Defines one or more **pipeline** jobs
  - Stored in source control
    - Enables versioning and review
- Supports complex logic
  - Conditional execution
  - Parallel steps
  - Etc.

Jobs

- Runnable **tasks** in Jenkins
  - Basic unit of functionality
  - Defined in a pipeline
  - Can include stages
- Accept various **parameters** in order to modify the build process
- Store **artifacts** (binaries, reports, etc.) and record build **results**

Steps

- **Individual tasks** within a Jenkins job
- **Command** or a series of **commands**
- In declarative syntax
  - Script commands
    - Shell scripts or batch commands
  - Tool invocation
  - File operations

Actions

- **Operations** that are performed by steps
- Actual command executions or function calls that
  - Interact with the workspace
  - Modify the build state
  - Send notifications

Jenkins Pipeline Syntax Keywords

- `pipeline`
  - Defines the block where the pipeline process is described
- `agent`
  - Specifies where the entire pipeline or a specific stage will execute in the Jenkins environment
- `stages`
  - Sequence of one or more stages that are to be executed in a defined order
- `stage`
  - Defines a conceptually distinct subset of tasks performed through the entire pipeline

Jenkins Pipeline Syntax Keywords

Software University

- `steps`
  - Defines a series of one or more steps to be executed in a given stage
- `script`
  - Allows for the inclusion of arbitrary Groovy code to be executed
- `environment`
  - Defines a set of environment variables for the steps to use
- `post`
  - Determines one or more additional steps that are run upon the completion of the pipeline's or stage's execution

Architecture

Jenkins Architecture

- Jenkins follows a distributed architecture
- **Main component → controller**
  - Responsible for scheduling jobs, dispatching builds to nodes (agents) and monitoring them
- **Distributed nature**
  - Jenkins can run jobs on different machines (**nodes** or **agents**)
    - Allows scaling as the workload increases

Controller/Agent Model

- **Controller**
  - Manages entire Jenkins environment
  - Previously known as **master**
- **Agents**
  - Machines or virtual instances that execute the jobs, dispatched by the controller
  - Allow builds and test to run in different environment
- **Distributed builds**
  - Multiple agents can run concurrently
    - Optimizes the utilization of resources

Scalability, Load Balancing and Security

- Jenkins scales **horizontally** by adding more agents
- Automatically **distributes jobs** among **available agents** based on their configurations and capabilities
- Supports various **authentication mechanisms**
- Communication between controller and agents can be **encrypted**
  - Ensures code and build results are securely transmitted

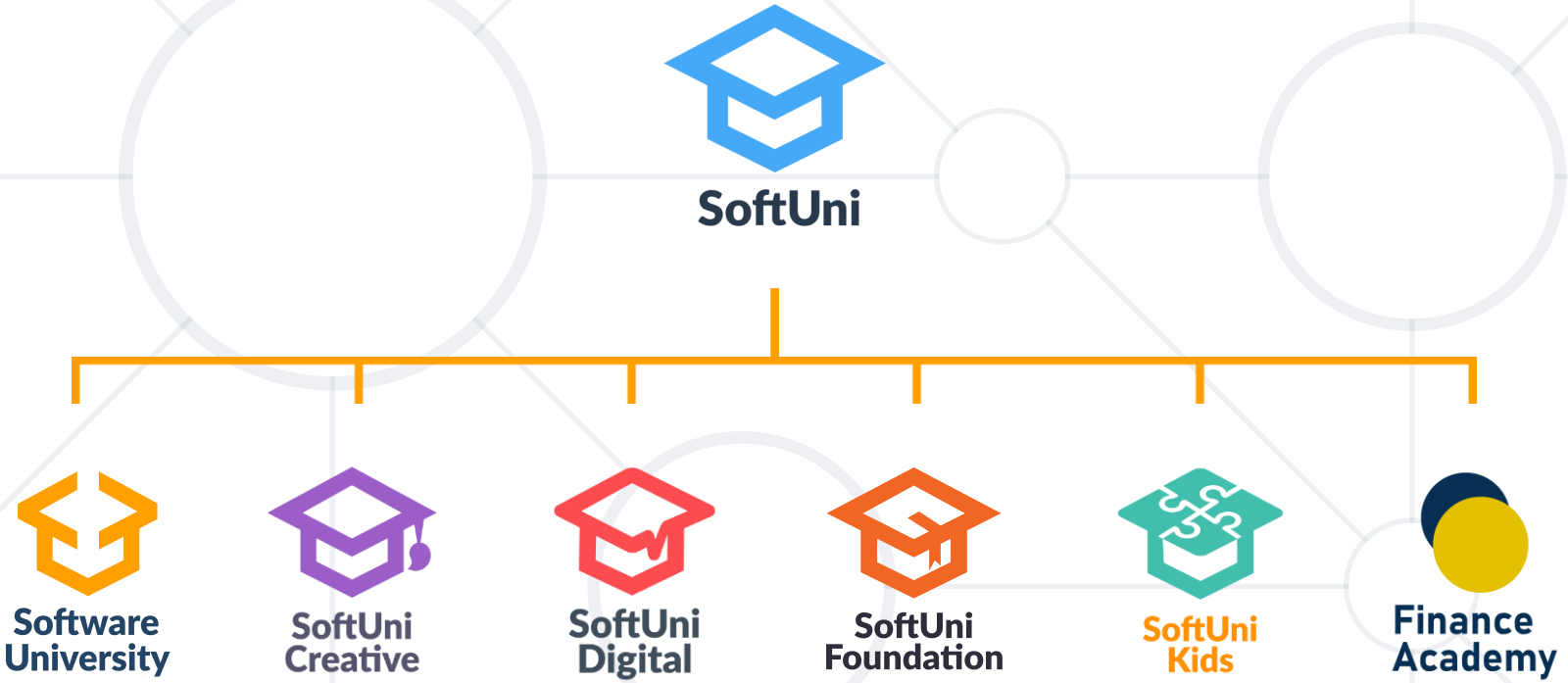
Plugin Architecture

- **Plugins** == **primary** method extending Jenkins
  - Thousands of plugins available in the **ecosystem**
- Plugin architecture makes Jenkins highly **extensible** and **customizable**
  - Plugins can be chosen based on the user's specific requirements
- Allows for a lightweight and lean core with ability to expand capabilities if needed
  - Helps Jenkins evolve with the changing technology

Summary

- **Jenkins** == open-source automation server that simplifies CI/CD
- Using the **key concepts**, a **Jenkinsfile** is written, which embodies the Pipeline as Code philosophy
- Jenkins follows **distributed architecture** and uses **plugins**, which make it more extensible and customizable

Questions?



SoftUni Diamond Partners

Software University

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg