Jenkins Basics

Jenkins is an open-source automation server that is used to automate parts of the software development process, including building, testing, and deploying applications. It is a powerful tool for continuous integration and continuous delivery (CI/CD).

Key Concepts

1. **Continuous Integration (CI):** The practice of merging all developers' working copies to a shared mainline several times a day. Jenkins helps automate this process.
2. **Continuous Delivery (CD):** An extension of CI, which ensures that the codebase is always in a deployable state. Jenkins can automate the deployment process.

Jenkins Architecture

Jenkins Master: The central server that manages the overall CI/CD pipeline, including scheduling jobs, dispatching builds to slaves, and monitoring results.

Jenkins Slave: A machine that runs builds and tests dispatched by the master. This helps distribute workloads and parallelize job execution.

Key Features

1. **Extensibility:** Jenkins has a vast library of plugins that allow it to integrate with various other tools and services.
2. **Pipeline as Code:** Jenkins pipelines can be defined as code using the Groovy-based domain-specific language (DSL).
3. **Freestyle Projects:** Basic projects that support simple build, test, and deploy workflows.
4. **Declarative Pipelines:** A more structured way to define Jenkins pipelines, which makes it easier to read and write pipeline code.

Installing Jenkins on Amazon Linux

To install Jenkins on an Amazon Linux instance, follow these steps:

Step 1: Launch an Amazon EC2 Instance with Amazon Linux

1. **Log in to AWS Management Console:** Go to the EC2 Dashboard.
2. **Launch Instance:**
    - Click on "Launch Instance".
    - Choose an Amazon Machine Image (AMI): Select "Amazon Linux 2 AMI (HVM)".
    - Choose an Instance Type: Select an appropriate instance type (e.g., `t2.micro` if you're testing).
    - Configure Instance: Accept the default settings or configure as needed.
    - Add Storage: Accept the default settings or configure as needed.
    - Add Tags: Optionally add tags to identify your instance.
    - Configure Security Group: Create a new security group or select an existing one. Ensure ports 22 (SSH) and 8080 (for Jenkins) are open.
    - Review and Launch: Review your settings and launch the instance. Make sure to create or select a key pair for SSH access.

Step 2: Connect to Your Instance

1. **Connect to the Instance:**

    Use SSH to connect to your instance. Replace `<your-key-pair.pem>` and `<ec2-instance-public-dns>` with your actual key pair file and instance public DNS:
    ```
    ssh -i <your-key-pair.pem>
    ec2-user@<ec2-instance-public-dns>
    ```

Step 3: Install Java

Jenkins requires Java to run. Install OpenJDK 11:

```
sudo amazon-linux-extras install java-openjdk11 -y
```

Step 4: Add Jenkins Repository

Add the Jenkins repository to your system:

```
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import
https://pkg.jenkins.io/redhat-stable/jenkins.io.key
```

Step 5: Install Jenkins

Install Jenkins using `yum`:

```
sudo yum install jenkins -y
```

Step 6: Start and Enable Jenkins

Start Jenkins and enable it to start on boot:

```
sudo systemctl start jenkins

sudo systemctl enable jenkins
```

Step 7: Adjust Firewall Settings

Ensure the firewall allows traffic on port 8080, which Jenkins uses by default. Amazon Linux 2 uses `firewalld`:

```
sudo firewall-cmd --permanent --zone=public
--add-port=8080/tcp

sudo firewall-cmd --reload
```

If you're using a security group instead of `firewalld`, ensure port 8080 is open:

1. Go to the EC2 Dashboard.
2. Select "Security Groups" from the sidebar.
3. Find and select your instance's security group.
4. Click on "Edit Inbound Rules".
5. Add a rule to allow traffic on port 8080.

Step 8: Access Jenkins

1. **Open Your Browser:**
   - Go to `http://<ec2-instance-public-dns>:8080`.
2. **Unlock Jenkins:**

   Retrieve the initial admin password:
   ```
   sudo cat /var/lib/jenkins/secrets/initialAdminPassword
   ```

   Copy the password and paste it into the Jenkins setup wizard.

Once the setup is complete, Jenkins will be ready to use on your Amazon Linux instance. You can start configuring it to fit your CI/CD needs, adding plugins, setting up projects, and creating pipelines.

Configuring Jenkins:

- ○ **Global Tool Configuration:** Set up JDK, Git, Maven, etc.
- ○ **Manage Plugins:** Add or update Jenkins plugins.
- ○ **Manage Nodes and Clouds:** Configure build agents (slaves) and cloud integrations.

Jenkins provides several types of jobs (also known as projects) that can be configured to automate various tasks in a CI/CD pipeline. Here are the main types of Jenkins jobs:

1. Freestyle Project

- **Description:** The most basic type of Jenkins job. It allows you to configure simple build, test, and deploy steps.
- **Use Case:** Ideal for straightforward build processes, such as compiling code and running tests.
- **Configuration:** Supports build triggers, build steps (e.g., executing shell commands), and post-build actions (e.g., archiving artifacts).

2. Pipeline

- **Description:** A powerful type of job that allows you to define a sequence of tasks, known as stages, in a script. Pipelines are defined using a domain-specific language (DSL) based on Groovy.
- **Use Case:** Suitable for complex CI/CD workflows, including branching, parallel execution, and conditional execution.
- **Configuration:** Can be defined in the Jenkins UI or in a `Jenkinsfile` stored in the source code repository.
- **Types:**
    - **Declarative Pipeline:** A simpler and more structured way to define pipelines.
    - **Scripted Pipeline:** Offers more flexibility and control but requires more complex Groovy scripting.

3. Multibranch Pipeline

- **Description:** Automatically creates a pipeline for each branch in your source control repository. It enables automatic management of branches and pipelines.
- **Use Case:** Useful for projects with multiple branches, such as feature branches, where each branch needs its own pipeline.

- **Configuration:** Jenkins scans the repository and creates pipelines for each branch with a `Jenkinsfile`.

4. Folder

- **Description:** Not a job itself, but a way to organize multiple jobs into a hierarchical structure.
- **Use Case:** Helps manage large numbers of jobs by grouping them into folders.
- **Configuration:** Allows you to apply common settings to all jobs within a folder.

5. GitHub Organization

- **Description:** Automatically discovers and manages repositories within a GitHub organization.
- **Use Case:** Suitable for organizations with multiple repositories that need to be managed under a single Jenkins job.
- **Configuration:** Jenkins scans the GitHub organization and creates Multibranch Pipelines for each repository.

6. Bitbucket Team/Project

- **Description:** Similar to the GitHub Organization job, but for Bitbucket. It automatically discovers and manages repositories within a Bitbucket team or project.
- **Use Case:** Ideal for Bitbucket users with multiple repositories that need to be managed under a single Jenkins job.
- **Configuration:** Jenkins scans the Bitbucket team or project and creates Multibranch Pipelines for each repository.

7. External Job

- **Description:** Tracks the execution of jobs that are run outside of Jenkins.
- **Use Case:** Useful for integrating external build systems or tools with Jenkins.
- **Configuration:** Configured to monitor the status of external jobs and report back to Jenkins.

8. Maven Project

- **Description:** Specialized job type for building Maven projects. It understands the Maven build lifecycle and provides specific features for Maven builds.
- **Use Case:** Best suited for projects that use Maven as their build tool.
- **Configuration:** Supports advanced Maven-specific configurations and reporting.

9. Matrix Project (Deprecated)

- **Description:** Allows the configuration of a job to run multiple configurations (e.g., different JDK versions, operating systems) in parallel.
- **Use Case:** Useful for testing applications across different environments.
- **Configuration:** Define axes (e.g., JDK version, OS type) and Jenkins will run the job with all combinations of the defined axes.

Jenkins provides several types of jobs (also known as projects) that can be configured to automate various tasks in a CI/CD pipeline. Here are the main types of Jenkins jobs:

1. Freestyle Project

- **Description:** The most basic type of Jenkins job. It allows you to configure simple build, test, and deploy steps.
- **Use Case:** Ideal for straightforward build processes, such as compiling code and running tests.
- **Configuration:** Supports build triggers, build steps (e.g., executing shell commands), and post-build actions (e.g., archiving artifacts).

2. Pipeline

- **Description:** A powerful type of job that allows you to define a sequence of tasks, known as stages, in a script. Pipelines are defined using a domain-specific language (DSL) based on Groovy.
- **Use Case:** Suitable for complex CI/CD workflows, including branching, parallel execution, and conditional execution.
- **Configuration:** Can be defined in the Jenkins UI or in a `Jenkinsfile` stored in the source code repository.
- **Types:**
  - **Declarative Pipeline:** A simpler and more structured way to define pipelines.
  - **Scripted Pipeline:** Offers more flexibility and control but requires more complex Groovy scripting.

3. Multibranch Pipeline

- **Description:** Automatically creates a pipeline for each branch in your source control repository. It enables automatic management of branches and pipelines.
- **Use Case:** Useful for projects with multiple branches, such as feature branches, where each branch needs its own pipeline.
- **Configuration:** Jenkins scans the repository and creates pipelines for each branch with a `Jenkinsfile`.

4. Folder

- **Description:** Not a job itself, but a way to organize multiple jobs into a hierarchical structure.
- **Use Case:** Helps manage large numbers of jobs by grouping them into folders.
- **Configuration:** Allows you to apply common settings to all jobs within a folder.

5. GitHub Organization

- **Description:** Automatically discovers and manages repositories within a GitHub organization.
- **Use Case:** Suitable for organizations with multiple repositories that need to be managed under a single Jenkins job.
- **Configuration:** Jenkins scans the GitHub organization and creates Multibranch Pipelines for each repository.

6. Bitbucket Team/Project

- **Description:** Similar to the GitHub Organization job, but for Bitbucket. It automatically discovers and manages repositories within a Bitbucket team or project.
- **Use Case:** Ideal for Bitbucket users with multiple repositories that need to be managed under a single Jenkins job.
- **Configuration:** Jenkins scans the Bitbucket team or project and creates Multibranch Pipelines for each repository.

7. External Job

- **Description:** Tracks the execution of jobs that are run outside of Jenkins.
- **Use Case:** Useful for integrating external build systems or tools with Jenkins.
- **Configuration:** Configured to monitor the status of external jobs and report back to Jenkins.

8. Maven Project

- **Description:** Specialized job type for building Maven projects. It understands the Maven build lifecycle and provides specific features for Maven builds.
- **Use Case:** Best suited for projects that use Maven as their build tool.
- **Configuration:** Supports advanced Maven-specific configurations and reporting.

9. Matrix Project (Deprecated)

- **Description:** Allows the configuration of a job to run multiple configurations (e.g., different JDK versions, operating systems) in parallel.
- **Use Case:** Useful for testing applications across different environments.

- **Configuration:** Define axes (e.g., JDK version, OS type) and Jenkins will run the job with all combinations of the defined axes.

Example Usage Scenarios

- **Freestyle Project:** Compiling a small Java application and running unit tests.
- **Pipeline:** Building, testing, and deploying a microservices application with multiple stages and conditional logic.
- **Multibranch Pipeline:** Automatically creating CI/CD pipelines for each feature branch in a Git repository.
- **Folder:** Organizing jobs for different teams or projects within an organization.
- **GitHub Organization:** Managing CI/CD pipelines for all repositories in a GitHub organization.
- **Bitbucket Team/Project:** Managing CI/CD pipelines for all repositories in a Bitbucket project.
- **External Job:** Integrating builds that are triggered by an external tool but tracked in Jenkins.
- **Maven Project:** Building a large enterprise Java application using Maven.

By understanding the different types of Jenkins jobs, you can choose the most appropriate one for your specific use case, ensuring efficient and effective CI/CD processes.

Creating a Jenkins Job

1. **Freestyle Project:**
   - Go to Jenkins dashboard and click on "New Item".
   - Select "Freestyle project" and give it a name.
   - Configure source code management, build triggers, and build steps.
   - Save and run the job.
2. **Pipeline Project:**
   - Go to Jenkins dashboard and click on "New Item".
   - Select "Pipeline" and give it a name.
   - Define the pipeline script in the job configuration or reference a `Jenkinsfile` from your source control repository.

Basic Pipeline Example

Declarative Pipeline:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                echo 'Building...'
                // Add build steps here
            }
        }
        stage('Test') {
            steps {
                echo 'Testing...'
                // Add test steps here
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying...'
                // Add deploy steps here
            }
        }
    }
}
```

Common Jenkins Plugins

- **Git Plugin:** Integrates Jenkins with Git repositories.
- **Pipeline Plugin:** Adds support for Jenkins pipelines.
- **Docker Plugin:** Allows Jenkins to interact with Docker containers.
- **Credentials Plugin:** Manages credentials securely within Jenkins.
- **Blue Ocean:** A modern user interface for Jenkins.

Jenkins is highly configurable and scalable, making it a critical tool in modern DevOps practices.