Complete Guide: Jenkins Master-Slave & Maven Setup

This guide documents the end-to-end process of setting up a distributed Jenkins architecture on AWS, including manual Maven installation and Java version troubleshooting.

Phase 1: Jenkins Master Setup

Instance: Ubuntu 24.04 (`t2.medium` recommended).

1. Install Java 21: Master and Slave must have matching Java versions for stable communication.

```
sudo apt update && sudo apt install -y openjdk-21-jre
```

2. Install Jenkins:

- Add Jenkins repository keys and install the package.
- Start and enable Jenkins: `sudo systemctl start jenkins`.

3. Security Group: Open Port **8080** for UI and Port **22** for SSH.

Phase 2: Slave Node Preparation

Instance: Ubuntu 24.04 (`t2.micro` is sufficient).

1. **Install Java 21:** Use the same version as the Master.

    ```
    sudo apt update && sudo apt install -y openjdk-21-jdk
    ```

2.Create Jenkins Directory:

```
mkdir -p /home/ubuntu/jenkins

sudo chown -R ubuntu:ubuntu /home/ubuntu/jenkins
```

Phase 3: Manual Maven Installation (On Slave)

Since we want a specific version (3.9.12) and full control, we install it manually.

1. **Download and Extract:**

```
cd /opt

sudo wget https://dlcdn.apache.org/maven/maven-3/3.9.12/binaries/apache-maven-3.9.12-bin.tar.gz

sudo tar -xvzf apache-maven-3.9.12-bin.tar.gz

sudo mv apache-maven-3.9.12 maven

sudo chown -R ubuntu:ubuntu /opt/maven
```

Create Global Symbolic Link: This allows you to run `mvn` from anywhere in the terminal.

```
sudo ln -s /opt/maven/bin/mvn /usr/bin/mvn
```

Phase 4: Jenkins-Slave Integration (Master UI)

1. **SSH Credentials:**
   - **Kind:** SSH Username with private key.
   - **ID:** `slave-ssh-key`.
   - **Username:** `ubuntu`.
   - **Private Key:** Paste the content of your `.pem` file.
2. **Node Configuration:**
   - **Remote Root Directory:** `/home/ubuntu/jenkins`.
   - **Labels:** `maven-build`.
   - **Host:** Slave's **Private IP**.
   - **Verification:** Select **Non-verifying Verification Strategy**.
3. **Launch:** Check logs for `Agent successfully connected and online`.

Phase 5: Troubleshooting & The "Build Solution"

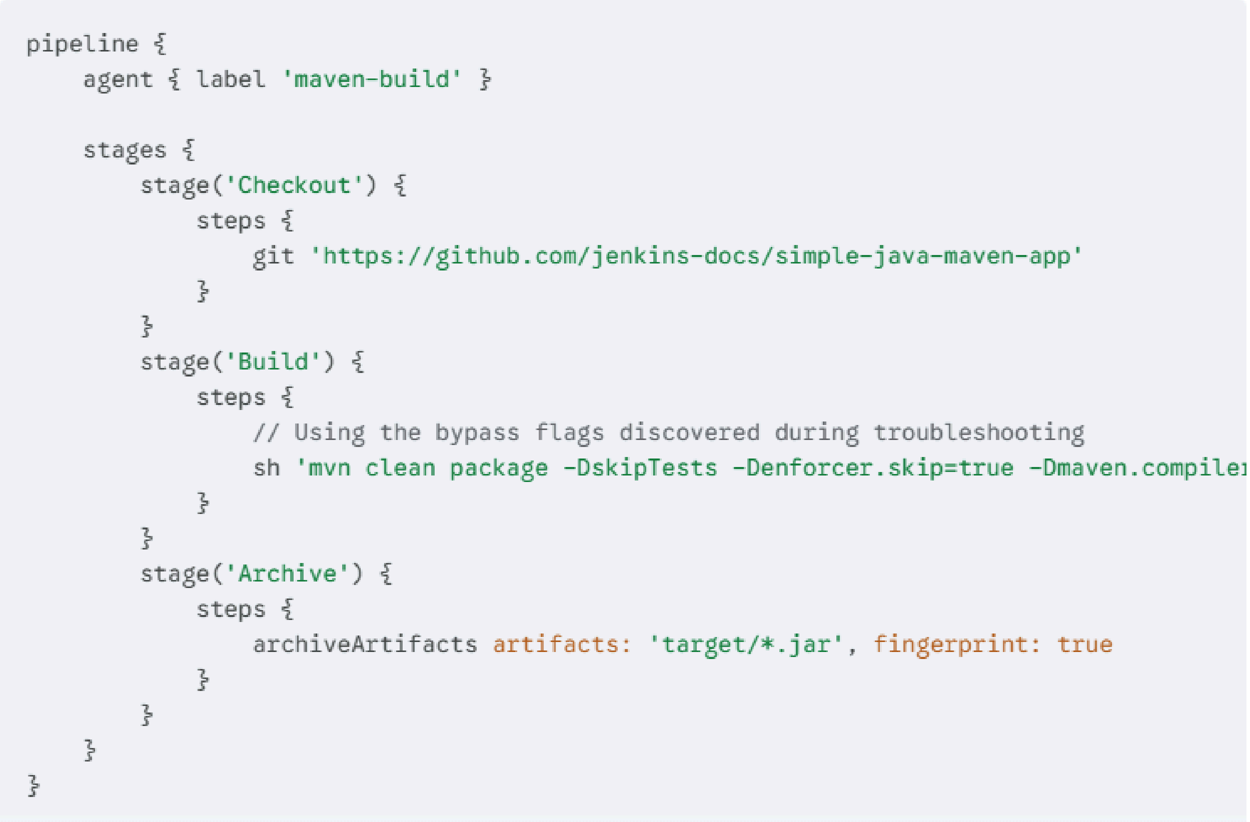
During the build, we encountered "Java version mismatches" because the project `pom.xml` had strict rules (Enforcer Plugin and Compiler Plugin).

The Final Success Command: Use these flags to bypass strict version checks and skip tests to ensure a fast build:

mvn clean package -DskipTests -Denforcer.skip=true -Dmaven.compiler.source=21 -Dmaven.compiler.target=21

Phase 6: The Pipeline Script (Declarative)

Create a **Pipeline Job** and use this script to automate the entire process.



```
pipeline {
    agent { label 'maven-build' }

    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/jenkins-docs/simple-java-maven-app'
            }
        }
        stage('Build') {
            steps {
                // Using the bypass flags discovered during troubleshooting
                sh 'mvn clean package -DskipTests -Denforcer.skip=true -Dmaven.compiler
            }
        }
        stage('Archive') {
            steps {
                archiveArtifacts artifacts: 'target/*.jar', fingerprint: true
            }
        }
    }
}
```

Code :-

```
pipeline {
    agent { label 'maven-build' }

    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/jenkins-docs/simple-java-maven-app'
            }
        }
        stage('Build') {
            steps {
                // Using the bypass flags discovered during troubleshooting
                sh 'mvn clean package -DskipTests -Denforcer.skip=true
-Dmaven.compiler.source=21 -Dmaven.compiler.target=21'
            }
        }
        stage('Archive') {
            steps {
                archiveArtifacts artifacts: 'target/*.jar', fingerprint: true
            }
        }
    }
}
```

Important Files & Paths to Remember:

- **Maven Home:** `/opt/maven`.
- **Java Executable:** `/usr/lib/jvm/java-21-openjdk-amd64/bin/java`.
- **Workspace:** `/home/ubuntu/jenkins/workspace`.
- **Build Artifact:** Found in the `target/` folder after success.