Setup Guide: Demo Project: Create a CI Pipeline with Jenkinsfile

---

Install Build Tools (Maven, Node) and Docker in Jenkins

This guide provides step-by-step instructions for configuring Maven, Node.js, and Docker in Jenkins. It includes both UI-based and manual methods, along with detailed reasoning for each step.

---

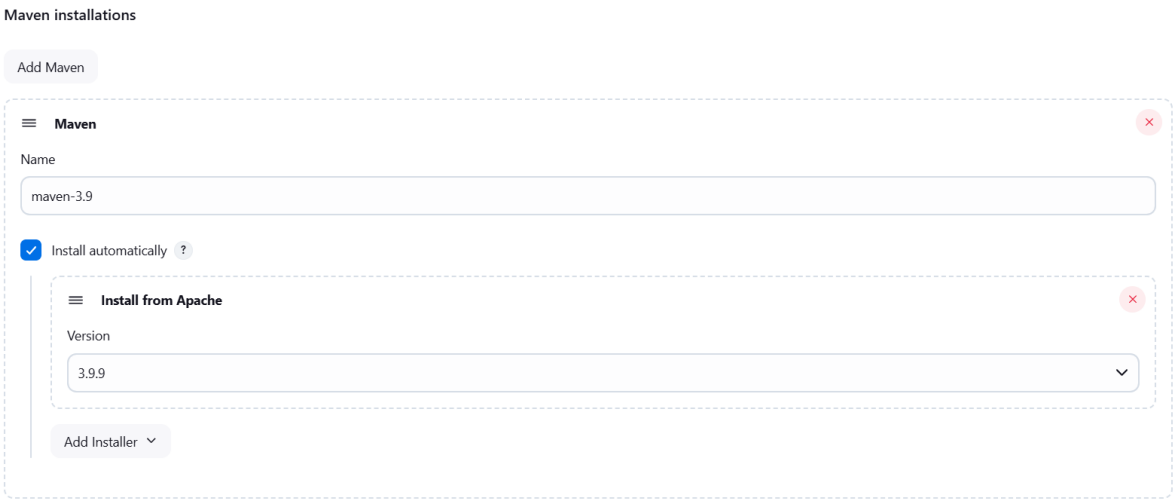
1. Install and Configure Maven (install via UI)

Maven is required to build Java applications and manage dependencies.

Steps:

1. Go to Jenkins **Manage Jenkins → System Configuration → Tools**.
2. Scroll down to **Maven**:

- **Name**: `maven-3.9`
- **Version**: Choose the latest version available (e.g., `3.9.9`).



3. Click **Save**.

Verification:

After saving the configuration, verify the Maven installation:

1. Create a Freestyle job or use the Jenkins Script Console.
2. Run the following command: `mvn -v`

   You should see the installed Maven version.

---

2. Install Node.js and npm in Jenkins container

Node.js is required for front-end builds, but Jenkins does not natively support Node.js installation via the UI.

Why Manual Installation?

The Node.js plugin for Jenkins supports installing Node.js, but npm is not included. This requires manual installation in the Jenkins container.

Steps:

1. Find the Jenkins container ID: `docker ps`

2. Enter the container as the root user: `docker exec -u 0 -it <container-id> bash`
3. Check the Linux distribution: `cat /etc/issue`
    - Example: `Debian GNU/Linux`.
4. Update the package manager and install curl: `apt update && apt install curl -y`
5. Download and execute the Node.js setup script:
    - `curl -sL https://deb.nodesource.com/setup_20.x -o nodesource_setup.sh`
    - `bash nodesource_setup.sh`
6. Install Node.js and npm: `apt install nodejs -y`
7. Verify the installation:
    - `node -v`
    - `npm -v`

---

3. Make Docker Available in Jenkins

Docker is essential for building containerized applications. Here's how to ensure Docker commands are accessible inside the Jenkins container.

Steps:

1. Stop the Jenkins container if it's already running:

- `docker ps`
- `docker stop <container id>`

2. Run Jenkins with Docker.sock mounted:

```
docker run -p 8080:8080 -p 50000:50000 -d -v jenkins_home:/var/jenkins_home -v
/var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts
```

- **Why mount `docker.sock`?**

  This allows the Jenkins container to communicate with the Docker daemon on the host.

3. Enter the container as the root user: `docker exec -u 0 -it <container id> bash`

4. Install Docker inside the container:

   `curl https://get.docker.com/ > dockerinstall && chmod 777 dockerinstall && ./dockerinstall`

5. Set permissions for the Docker socket:

   - `ls -l /var/run/docker.sock`

     ```
     srw-rw---- 1 root 112 0 Dec 14 23:18 /var/run/docker.sock
     ```

   - `chmod 666 /var/run/docker.sock`
   - `ls -l /var/run/docker.sock`

     ```
     srw-rw-rw- 1 root 112 0 Dec 14 23:18 /var/run/docker.sock
     ```

   - `exit`

6. Loggin in as Jenkins user: `docker exec -it <container id> bash` and verify Docker inside Jenkins: `docker ps`

---

4. Create Credentials in Jenkins

4.1 DockerHub Credentials

1. Go to Jenkins **Manage Jenkins → Credentials → System → Add Credentials**.
2. Enter the following details:
   - **Kind**: Username with password.
   - **Username**: Your DockerHub username.
   - **Password**: Your DockerHub password.
   - **ID**: `docker-hub-repo`.
   - **Description**: `DockerHub repository credentials`.
3. Click **Create**.

4.2 GitLab Credentials

1. Go to Jenkins **Manage Jenkins → Credentials → System → Add Credentials**.
2. Enter the following details:
   - **Kind**: Username with password.
   - **Username**: Your GitLab username.
   - **Password**: Your GitLab personal access token.
   - **ID**: `gitlab-credentials`.
   - **Description**: `GitLab repository credentials`.
3. Click **Create**.

Best Practices

- Regularly update Node.js, npm, and Docker to their latest versions for security and compatibility.
- Use secure credentials (e.g., avoid hardcoding passwords in pipeline scripts).
- Mount `docker.sock` only in secure environments.
