Integrating Jenkins with Docker

Integrating Jenkins with Docker allows you to automate the build, test, and deployment of Docker containers directly from Jenkins. This integration is particularly useful for DevOps pipelines, as it facilitates continuous integration and continuous deployment (CI/CD) workflows. Below is an overview and step-by-step guide for setting up Jenkins with Docker on Ubuntu.

Overview

Jenkins: Jenkins is an open-source automation server that helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and delivery.

Docker: Docker is a platform that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their software, libraries, and configuration files.

By integrating Jenkins with Docker, you can:

- **Build Docker images** using Jenkins.
- **Run Docker containers** from Jenkins.
- Use Docker containers as Jenkins agents for builds and tests.

Step-by-Step Setup

1. Prerequisites

- An Ubuntu machine (either a server or a virtual machine).
- Jenkins installed and running on Ubuntu. If Jenkins is not installed, you can follow the instructions below.

2. Install Jenkins on Ubuntu

1. **Update System Packages**:

```
sudo apt update
sudo apt upgrade -y
```

2. **Install Java**:
   Jenkins requires Java to run. Install OpenJDK:
   ```
   sudo apt install openjdk-11-jdk -y
   ```
3. **Add Jenkins Repository and Install Jenkins**:
   ```
   wget -q -O -
   https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo
   apt-key add -

   sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable
binary/ > /etc/apt/sources.list.d/jenkins.list'

   sudo apt update
   sudo apt install jenkins -y
   ```
4. **Start and Enable Jenkins Service**:
   ```
   sudo systemctl start jenkins

   sudo systemctl enable jenkins
   ```
5. **Access Jenkins**:
   Open your web browser and go to `http://your_server_ip_or_domain:8080`. You will see the Jenkins setup wizard.

   Unlock Jenkins: Obtain the initial admin password with:
   ```
   sudo cat /var/lib/jenkins/secrets/initialAdminPassword
   ```

Install Suggested Plugins: Proceed with the suggested plugins and create your admin user.

3. Install Docker on Ubuntu

1. **Remove Old Docker Versions**:
   ```
   sudo apt remove docker docker-engine docker.io containerd
   runc
   ```

2. **Set Up the Docker Repository**:
   ```
   sudo apt update
   ```

```
    sudo apt install apt-transport-https ca-certificates curl
software-properties-common -y

    curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -

    sudo add-apt-repository "deb [arch=amd64]

    https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable"

    sudo apt update
```

3. **Install Docker**:
   ```
   sudo apt install docker-ce -y
   ```

4. **Add Jenkins User to Docker Group**:
   To allow Jenkins to use Docker commands, add the Jenkins user to the Docker group:
   ```
   sudo usermod -aG docker jenkins
   ```

5. **Restart Jenkins**:
   ```
   sudo systemctl restart jenkins
   ```

4. Configure Jenkins for Docker

1. **Install Docker Plugin in Jenkins**:
   - Go to Jenkins Dashboard → Manage Jenkins → Manage Plugins.
   - In the "Available" tab, search for "Docker" and install the "Docker" and "Docker Pipeline" plugins.
2. **Configure Docker in Jenkins**:
   - Go to Jenkins Dashboard → Manage Jenkins → Configure System.
   - Scroll down to the "Cloud" section and click "Add a new cloud" → "Docker".

    - Configure Docker settings according to your setup (you can keep it simple initially and refine as needed).
3. **Test Docker Integration**:
    - Create a new Jenkins job (Pipeline or Freestyle).
    - Add a build step to execute Docker commands or create a simple pipeline script like the following:

groovy

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                script {
                    sh 'docker --version'
                    sh 'docker run hello-world'
                }
            }
        }
    }
}
```

5. Verify Jenkins and Docker Integration

- **Run the Job**: Trigger the job and check the console output to verify that Docker commands are executed successfully.
- **Troubleshoot**: If you encounter permissions errors, ensure Jenkins has the necessary permissions to execute Docker commands (Docker group membership, etc.).

By integrating Jenkins with Docker, you can automate Docker-based workflows, enabling a streamlined CI/CD process. This setup allows you to build Docker images, run containers, and perform containerized builds and tests within Jenkins, leveraging Docker's portability and isolation features.