# Setting Up a CI/CD Workspace with Docker, Gitea, Jenkins, and Allure

This tutorial guides you through setting up a complete CI/CD environment using Docker, Gitea (a self-hosted Git service), Jenkins (automation server), and Allure (test reporting tool). By the end of this tutorial, you will have a running workspace capable of building and testing Maven projects with automated reports.

## Prerequisites

Ensure you have the following installed:

• Docker and Docker Compose (Installation depends on OS)

## Step 1: Setting Up Docker Compose

Create a new directory for your workspace:

```bash
mkdir -p ~/Desktop/vvss/workspace && cd ~/Desktop/vvss/workspace
```

Create a `docker-compose.yml` file with the following content:

```yml
services:
 jenkins:
 build:
 context: .
 dockerfile\_inline: |
 FROM jenkins/jenkins:lts
 USER root
 RUN apt-get update && apt-get install -y maven tree
 RUN jenkins-plugin-cli --plugins allure-jenkins-plugin:2.32.0 workflow-aggregator:600.vb\_57cdd26fdd7 matrix-project:845.vffd7fa\_f27555 git:5.7.0
 RUN mkdir -p /usr/share/jenkins/ref/init.groovy.d && echo 'import jenkins.model.\*' > /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'import hudson.tools.\*' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'import ru.yandex.qatools.allure.jenkins.tools.\*' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'def desc = Jenkins.instance.getDescriptor(AllureCommandlineInstallation.class)' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'def installer = new AllureCommandlineInstaller("2.32.2")' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'def installSourceProperty = new InstallSourceProperty([installer])' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'def installation = new AllureCommandlineInstallation("allure", "", [installSourceProperty])' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'desc.setInstallations(installation)' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy && echo 'desc.save()' >> /usr/share/jenkins/ref/init.groovy.d/allure\_init.groovy
 USER jenkins
 container\_name: jenkins
 restart: always
 ports:
 - "8080:8080"
 - "50000:50000"
 volumes:
 - './jenkins\_data:/var/jenkins\_home'

 gitea:
 image: gitea/gitea:latest
 container\_name: gitea
 restart: always
 ports:
 - "3000:3000"
 - "2222:22"
 volumes:
 - './gitea\_data:/data'
 environment:
 - USER\_UID=1000
 - USER\_GID=1000
 - GITEA\_\_database\_\_DB\_TYPE=sqlite3
 - GITEA\_\_server\_\_DISABLE\_REGISTRATION=true
 - GITEA\_\_security\_\_INSTALL\_LOCK=true
 - GITEA\_\_server\_\_ROOT\_URL=http://gitea:3000/
```

## Step 2: Running Docker Compose

Open a terminal in the docker-compose.yml parent folder and start the services by running:

```bash
docker compose up
```

This will create Dockerized instances of Jenkins (pre-configured with java 17, Maven and Allure) and Gitea (our self-hosted github alternative)

## Step 3: Setting Up Gitea

Access Gitea by navigating to `[http://localhost:3000](http://localhost:3000/)`.



Register an account with:

• Username: admin

• Password: adminadmin



Create a new public repository named ‘project’



 Clone the repository locally:

```bash
git clone http://localhost:3000/admin/project.git
```

Move all your project files in the empty repository

Push all the project files to the repository

```bash
git add . && git commit -m “base” && git push
```



## Step 4: Setting Up Jenkins

Access Jenkins at `[http://localhost:8080](http://localhost:8080/)`. No username required, default is `admin`.

## Step 5: Creating a Pipeline Job

Create a new Jenkins Item (job)

Name: tests & Item Type: Pipeline



Configure the pipeline definition as: Pipeline Script

 and use the following script:

```groovy
pipeline {
 agent any
 stages {
 stage('Clean Workspace') {
 steps {
 sh 'rm -rf \*'
 }
 }

 stage('Clone the repository') {
 steps {
 git branch: 'master', url: 'http://gitea:3000/admin/repo.git'
 sh 'tree'
 }
 }

 stage('Build') {
 steps {
 sh 'mvn clean install -DincludeTags=ExamplePack'
 }
 }

 stage('Publish Allure Report') {
 steps {
 allure includeProperties: false, jdk: '', results: [[path: 'target/allure-results']]
 }
 }
 }
}
```

Take some time to inspect the steps of the pipeline.

Note down in your own words what you think it’s doing.

Apply & Save

## Step 6: Running the Pipeline

Click on 'Build Now' in Jenkins to run the pipeline.



Fingers crossed for the execution to be ‘green’ (successful)

## Step 7: Viewing Allure Reports



Once the pipeline completes, navigate to the Allure Reports to view detailed test results grouped by tags.