

Jenkins + Docker Pipeline Project Documentation

Objective

Automate the build, test, and deployment of a Dockerized web application using Jenkins Pipeline.

Prerequisites

Ensure the following are installed on your Jenkins server:

- ✓ Jenkins (Latest version)
 - ✓ Docker (For containerization)
 - ✓ Git (For version control)
 - ✓ SSH (For remote deployments)
 - ✓ Jenkins Plugins:
 - Pipeline Plugin
 - Docker Pipeline Plugin
 - Git Plugin
-

Step 1: Install Docker on Jenkins Server

1. **Update system packages and install Docker:**

```
sudo apt update  
sudo apt install docker.io -y
```

2. **Start and enable Docker:**

```
sudo systemctl start docker  
sudo systemctl enable docker
```

3. **Add Jenkins user to Docker group (to allow Jenkins to run Docker commands):**

```
sudo usermod -aG docker jenkins
```

4. **Restart Jenkins to apply changes:**

```
sudo systemctl restart jenkins
```

5. **Verify Docker installation:**

```
docker --version
```

Step 2: Enable Password Authentication (If Needed)

If SSH key authentication is not set up, enable password login:

1. **Connect to the remote server and edit the SSH configuration file:**

```
sudo nano /etc/ssh/sshd_config
```

2. **Modify these lines:**

```
PasswordAuthentication yes
```

```
PermitRootLogin yes
```

3. **Save the file and restart SSH:**

```
sudo systemctl restart ssh
```

4. **Test SSH login:**

```
ssh master@192.168.203.128
```

Step 3: Create a Simple Web Application

1. **Clone the sample app repository:**

```
git clone https://github.com/KyathamRohith/jenkins-docker.git
```

```
cd jenkins-docker
```

2. **Create a Dockerfile in the project directory:**

```
FROM nginx:latest – to run static web application
```

```
COPY index.html /usr/share/nginx/html/
```

```
EXPOSE 80
```

```
CMD ["nginx", "-g", "daemon off;"]
```

3. **Create an index.html file:**

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Jenkins + Docker Pipeline</title>
```

```
</head>
```

```
<body>
```

```
<h1>Deployment Successful with Jenkins and Docker!</h1>
```

```
</body>
```

```
</html>
```

Step 4: Create a Jenkins Pipeline

1. **Open Jenkins → New Item → Pipeline → OK**

2. **Copy and paste the following Jenkinsfile:**

```
pipeline {
  agent any
  environment {
    DOCKER_IMAGE = "app-image"
    DOCKER_TAG = "latest"
    DOCKER_REPO = "default-repo/app-image"
    DOCKER_CREDENTIALS_ID = "docker-credentials-id"
    CONTAINER_NAME = "local-container"
    CONTAINER_NAME1 = "server-container"
  }
  stages {
    stage('Clone Repository') {
      steps {
        git 'https://github.com/default-user/repo.git'
      }
    }
    stage('Docker Login') {
      steps {
        script {
          docker.withRegistry('https://index.docker.io/v1/', DOCKER_CREDENTIALS_ID) {
            echo "Logged into Docker Hub"
          }
        }
      }
    }
    stage('Build Docker Image') {
      steps {
        script {
          sh "docker build -t ${DOCKER_IMAGE}:${DOCKER_TAG} ."
        }
      }
    }
  }
}
```

```

    }
  }
}
stage('Run Container Locally') {
  steps {
    script {
      sh """
        docker ps -a -q --filter name=${CONTAINER_NAME} | xargs -r docker stop || true
        docker ps -a -q --filter name=${CONTAINER_NAME} | xargs -r docker rm || true
        docker run -d -p 8093:80 --name ${CONTAINER_NAME}
        ${DOCKER_IMAGE}:${DOCKER_TAG}
      """
    }
  }
}
stage('Push to Docker Hub') {
  steps {
    script {
      docker.withRegistry('https://index.docker.io/v1/', DOCKER_CREDENTIALS_ID) {
        sh "docker tag ${DOCKER_IMAGE}:${DOCKER_TAG}
        ${DOCKER_REPO}:${DOCKER_TAG}"
        sh "docker push ${DOCKER_REPO}:${DOCKER_TAG}"
      }
    }
  }
}
stage('Deploy to Server') {
  steps {
    script {
      sh """
        sshpass -p "password" ssh -o StrictHostKeyChecking=no user@server-ip '
        docker pull ${DOCKER_REPO}:${DOCKER_TAG} &&

```

```

        docker ps -a -q --filter name=${CONTAINER_NAME1} | xargs -r docker stop || true &&

        docker ps -a -q --filter name=${CONTAINER_NAME1} | xargs -r docker rm || true &&

        docker run -d -p 80:80 --name ${CONTAINER_NAME1}
        ${DOCKER_REPO}:${DOCKER_TAG}'

        """"

    }

}

}

}

}

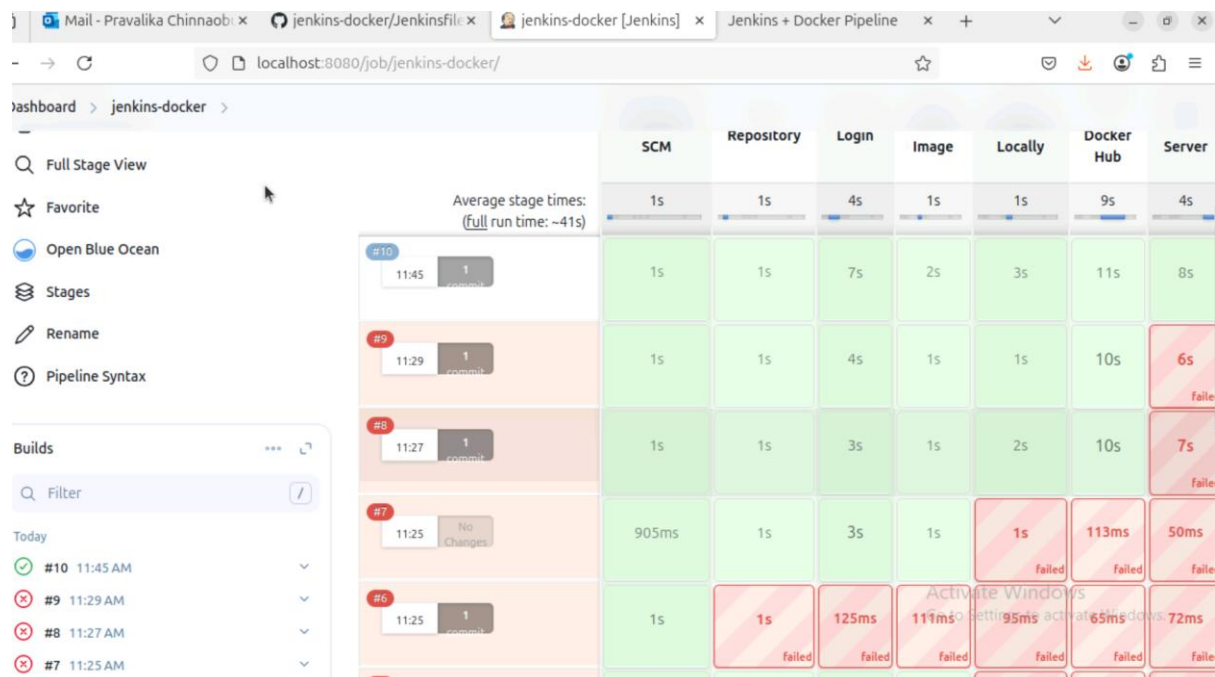
}

```

Step 5: Run the Jenkins Pipeline

1. Go to Jenkins Dashboard → Click on your pipeline job
2. Click "Build Now"
3. Check console output to verify the build process

OUTPUT



localhost:8080/job/jenkins-docker/10/console

Jenkins

Dashboard > jenkins-docker > #10

- Status
- Changes
- Console Output**
- Edit Build Information
- Delete build '#10'
- Timings
- Git Build Data
- Git Build Data
- Open Blue Ocean
- Pipeline Overview
- Pipeline Console

Console Output

Started by user admin

Obtained Jenkinsfile from git <https://github.com/Pravalikaa18/jenkins-docker.git>

[Pipeline] Start of Pipeline

[Pipeline] node

Running on Jenkins in /var/lib/jenkins/workspace/jenkins-docker

[Pipeline] {

[Pipeline] stage

[Pipeline] { (Declarative: Checkout SCM)

[Pipeline] checkout

Selected Git installation does not exist. Using Default

The recommended git tool is: NONE

No credentials specified

> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/jenkins-docker/.git #

timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url <https://github.com/Pravalikaa18/jenkins-docker.git> #

timeout=10

Localhost with portn0.

File Edit View VM Tabs Help

Mail - Pravalika Chinnab x jenkins-docker/Jenkinsfile x jenkins-docker [Jenkins] x Jenkins + Docker Pipeline x

localhost:909

Deployment Successful with Jenkins and Docker!