Date: 14.07.2025

# CI/CD pipeline with Jenkins

## Contents

1. Pre-requisites ..... 2

2. Script to install nodejs ..... 2

3. Create a pipeline to pull code from GitHub, build image with Docker and push image to Docker Hub ..... 2

4. Edit the pipeline to Build, Test, and Deploy a Dockerized Application. ..... 4

## 1. Pre-requisites

## 2. Script to install nodejs

```
#!usr/bin/env bash
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash
source ~/.bashrc
nvm -v
nvm install --lts
nvm alias default node
nvm use --lts
node -v
npm -v
npx -v
```

## 3. Create a pipeline to pull code from GitHub, build image with Docker and push image to Docker Hub

1. Create a simple NodeJS app with `npm create vite@latest`
2. Dockerfile

```
FROM node:20

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN npm run build

RUN npm install -g serve
EXPOSE 3000

CMD ["serve", "-s", "dist", "-l", "3000"]
```

3. Jenkinsfile

```
pipeline {

  agent any

  environment {
    DOCKERHUB_CRED = 'dockerhub-creds-id'
    IMAGE = 'aryanfafo/jenkins-nodejs-app'
  }

  stages {

    stage('Install Dependencies') {
```

```
      steps {
        script {
          echo 'Installing Dependencies…'
          sh 'npm install'

        }
      }
    }

    stage('Build Docker Image') {
      steps {
        script {
          echo 'Building Docker Image…'
          sh "docker build -t ${env.IMAGE}:${env.BUILD_NUMBER} ."
          sh "docker tag ${env.IMAGE}:${env.BUILD_NUMBER}
${env.IMAGE}:latest"
        }
      }
    }

    stage('Push to DockerHub') {
      steps {
        script {
          echo 'Pushing image to Docker Hub…'
          withCredentials([usernamePassword(credentialsId:
env.DOCKERHUB_CRED, usernameVariable: 'DOCKER_USER', passwordVariable:
'DOCKER_PASS')]) {
            sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --
password-stdin'
            sh "docker push ${env.IMAGE}:${env.BUILD_NUMBER}"
            sh "docker push ${env.IMAGE}:latest"
          }
        }
      }
    }
  }

  post {
    always {
      cleanWs()
    }
  }
}
```

4. jenkins_dind_setup.sh

```
#!/usr/bin/env bash

echo "[INFO] Starting Jenkins container with Docker socket access..."
docker run -d --name jenkins-dind -u root \
  -p 8080:8080 -p 50000:50000 \
  -v jenkins_home:/var/jenkins_home \
```

```
  -v /var/run/docker.sock:/var/run/docker.sock \
  jenkins/jenkins:lts

echo "[INFO] Installing Docker CLI in Jenkins container..."
docker exec -u root jenkins-dind bash -c "
  apt-get update && \
  apt-get install -y docker.io
"

echo "[INFO] Jenkins is running on http://localhost:8080"
echo "[INFO] Admin password:"
docker exec jenkins-dind cat /var/jenkins_home/secrets/initialAdminPassword
```

chmod +x jenkins_dind_setup.sh

5. Initialize git and push to github
6. Run jenkins_dind_setup.sh and copy the intialAdminPassword
7. Access the Jenkins container using local browser: `http://<EC2_IP>:8080`
8. Paste the initialAdminPassword
9. Install the suggested plugins
10. Create an admin user
11. Goto Dashboard > Manage Jenkins > Credentials > System > Global Credentials
Add a credential with following -
Username: aryanfafo, Password: (DockerHub's PAT), Id: dockerhub-creds-id
12. Goto Dashboard, then create a new job
Job name: vite-react-app, Type: Multibranch Pipeline
13. Configure Branch Source
Add Source > Select Git > Paste Repository's URL
14. Click Save. First Build will be triggered automatically. Next builds must be triggered manually.
15. Monitor by clicking on the Build Number on the left under Build Queue section and check the Pipeline Overview.

## 4. Edit the pipeline to Build, Test, and Deploy a Dockerized Application.

1. Update jenkins_dind_setup.sh

```
#!/usr/bin/env bash

echo "[INFO] Checking for existing Jenkins container..."
if [ "$(docker ps -aq -f name=jenkins-dind)" ]; then
  echo "[INFO] Stopping and removing existing jenkins-dind container..."
  docker stop jenkins-dind
  docker rm jenkins-dind
fi

echo "[INFO] Starting Jenkins container with Docker socket access..."
docker run -d --name jenkins-dind -u root \
  -p 8080:8080 -p 50000:50000 \
```

```
  -v jenkins_home:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock \
  jenkins/jenkins:lts

echo "[INFO] Installing Docker CLI, curl, unzip, Node.js 21, npm in Jenkins container..."
docker exec -u root jenkins-dind bash -c "
  apt-get update && \
  apt-get install -y docker.io curl unzip && \
  curl -fsSL https://deb.nodesource.com/setup_21.x | bash - && \
  apt-get install -y nodejs
"

echo "[INFO] Installing AWS CLI v2 in Jenkins container..."
docker exec -u root jenkins-dind bash -c "
  curl 'https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip' -o 'awscliv2.zip' && \
  unzip awscliv2.zip && \
  ./aws/install && \
  rm -rf aws awscliv2.zip
"

echo "[INFO] Jenkins is running on http://localhost:8080"
echo "[INFO] Admin password:"
docker exec jenkins-dind cat /var/jenkins_home/secrets/initialAdminPassword
```

2. Update Jenkinsfile

```
pipeline {

  agent any

  environment {
    DOCKERHUB_CRED = 'dockerhub-creds-id'
    IMAGE = 'aryanfafo/jenkins-nodejs-app'
    INSTANCE_ID = 'i-007144a114a8151ac'
    REGION = 'ap-south-1'
    CONTAINER = 'jenkins-nodejs-app'
  }

  stages {

    stage('Run Tests') {
      steps {
        script {
          echo 'Running Tests...'
          sh 'npm install'
          sh 'npm test'
        }
      }
    }
```

```
    stage('Build Docker Image') {
      steps {
        script {
          echo 'Building Docker Image...'
          sh "docker build -t ${env.IMAGE}:${env.BUILD_NUMBER} ."
          sh "docker tag ${env.IMAGE}:${env.BUILD_NUMBER} ${env.IMAGE}:latest"
        }
      }
    }

    stage('Push to DockerHub') {
      steps {
        script {
          echo 'Pushing image to Docker Hub...'
          withCredentials([usernamePassword(credentialsId: env.DOCKERHUB_CRED,
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
            sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
            sh "docker push ${env.IMAGE}:${env.BUILD_NUMBER}"
            sh "docker push ${env.IMAGE}:latest"
          }
        }
      }
    }

    stage('Deploy to EC2 via SSM') {
      steps {
        script {
          echo 'Deploying Docker container on EC2 using SSM...'
             withCredentials([usernamePassword(credentialsId: DOCKERHUB_CRED,
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
          def deployCommand = "docker pull ${IMAGE}:latest && docker stop
${CONTAINER} || true && docker rm ${CONTAINER} || true && docker run -d --name
${CONTAINER} -p 3000:3000 ${IMAGE}:latest"

            sh """
              aws ssm send-command \
                --document-name "AWS-RunShellScript" \
                --region ${REGION} \
                --instance-ids "${INSTANCE_ID}" \
                --parameters 'commands=["${deployCommand}"]' \
                --output text
            """
          }
        }
      }
    }
  }
```

```
  post {
    always {
      cleanWs()
    }
  }
}
```

3. Add a Test file

```
import { render, screen, fireEvent } from '@testing-library/react'
import { describe, it, expect } from 'vitest'
import App from './App.jsx'

describe('App Component', () => {
  it('should render and increment count on button click', () => {
    render(<App />)

    const button = screen.getByRole('button', { name: /count is/i })

    expect(button.textContent).toBe('count is 0')

    fireEvent.click(button)
    expect(button.textContent).toBe('count is 1')

    fireEvent.click(button)
    expect(button.textContent).toBe('count is 2')
  })
})
```

4. Update following (Add the following wherever specified)

| package.json | vite.config.js |
|---|---|
| "scripts": {<br>"test": "vitest run"<br>},<br>"devDependencies": {<br>"@testing-library/react": "^16.3.0",<br>"jsdom": "^26.1.0",<br>"vite": "^7.0.3",<br>"vitest": "^3.2.4"<br>} | import { defineConfig } from 'vite'<br>import react from '@vitejs/plugin-react'<br>// https://vite.dev/config/<br>export default defineConfig({<br>plugins: [react()],<br>test: {<br>environment: 'jsdom',<br>}<br>}) |

Run npm install

5. Push to GitHub
6. Create a new IAM role in AWS
Trusted Entity: EC2 and Policies: AmazonSSMFullAccess
Attach this new policy to EC2 instance
7. Setup and configure SSM Agent on EC2 instance

```
#!/usr/bin/env bash

echo "Installing Amazon SSM Agent using snap..."

sudo snap install amazon-ssm-agent --classic

sudo systemctl enable snap.amazon-ssm-agent.amazon-ssm-agent.service
sudo systemctl start snap.amazon-ssm-agent.amazon-ssm-agent.service

echo "SSM Agent status:"
sudo systemctl status snap.amazon-ssm-agent.amazon-ssm-agent.service --no-pager
```

8. Run the jenkins_dind_setup.sh script
9. Follow steps from previous section to setup Jenkins once again and create a multibranch pipeline
10. Monitor the pipeline