

DevOps Task 5: Jenkins & Kubernetes Deployment

Step 1: Setting Up WSL and Ubuntu

1. Open **Windows Subsystem for Linux (WSL)** and start **Ubuntu**:

```
wsl.exe -d Ubuntu
```

2. Verify system information and update packages:

```
sudo apt update
```

```
sudo apt upgrade
```

```
C:\Windows\System32>wsl.exe -d Ubuntu
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sat Mar 22 03:50:55 UTC 2025

System load: 0.0          Processes:              86
Usage of /:  1.2% of 1006.85GB   Users logged in:       0
Memory usage: 16%          IPv4 address for eth0: 172.27.44.42
Swap usage:  0%
```

Step 2: Installing and Configuring Jenkins

1. Start the Jenkins service:

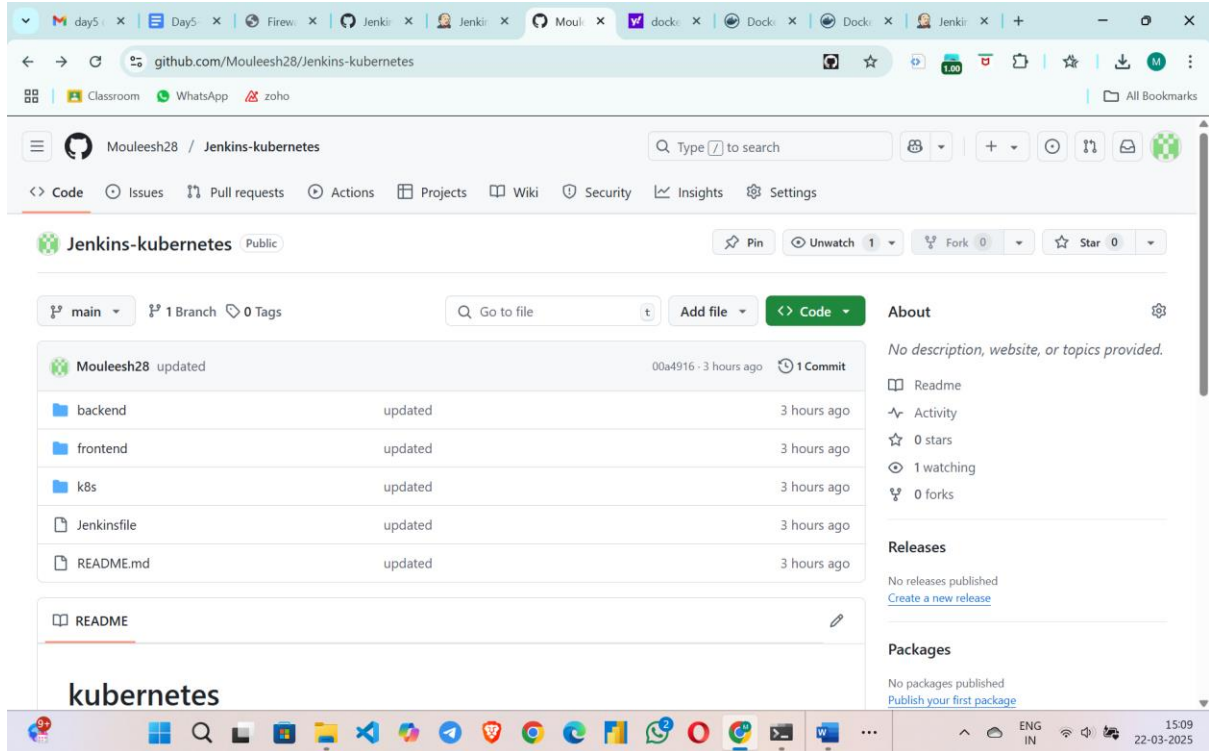
```
sudo systemctl start Jenkins
```

2. Enable Jenkins to start on boot:

```
sudo systemctl enable Jenkins
```

Step 3: Create new repository

1. Add Readme.md file



Step 4: Cloning Repositories

1. Clone the **Jenkins-Kubernetes** repository:

```
git clone https://github.com/Mouleesh28/Jenkins-Kubernetes.git  
cd Jenkins-Kubernetes
```

2. Clone the **Kubernetes** configuration repository:

```
git clone https://github.com/Mouleesh28/kubernetes.git
```

```
lsWelcome to Ubuntu 24.04.2 LTS (GNU/Linux 5.15.167.4-microsoft-standard-WSL2 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Sat Mar 22 03:41:51 UTC 2025

System load:  1.14          Processes:           48
Usage of /:   1.0% of 1006.85GB    Users logged in:    0
Memory usage: 10%          IPv4 address for eth0: 172.20.120.48
Swap usage:   0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/home/mouleesh/.hushlogin file.
mouleesh@DESKTOP-PB8PDK2:~$ ls
E-commerce  Jenkins-Docker-Demo  Jenkinsfile  docker-python-app  kubernetes  minikube-linux-amd64
mouleesh@DESKTOP-PB8PDK2:~$ git clone https://github.com/Mouleesh28/Jenkins-kubernetes.git
Cloning into 'Jenkins-kubernetes'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
mouleesh@DESKTOP-PB8PDK2:~$ cd Jenkins-kubernetes
mouleesh@DESKTOP-PB8PDK2:~/Jenkins-kubernetes$ git clone https://github.com/Mouleesh28/Kubernetes.git
Cloning into 'Kubernetes'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 17 (delta 1), reused 17 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (1/1), done.
mouleesh@DESKTOP-PB8PDK2:~/Jenkins-kubernetes$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
```

Step 5: Creating the Jenkins Pipeline

1. Open **Jenkinsfile** using a text editor:

```
nano Jenkinsfile
```

2. Add the following pipeline script:

```
pipeline {
    agent any
    environment {
        FRONTEND_IMAGE = "viratpk18/frontend-app:latest"
        BACKEND_IMAGE = "viratpk18/backend-app:latest"
        FRONTEND_CONTAINER = "frontend-container"
        BACKEND_CONTAINER = "backend-container"
        REGISTRY_CREDENTIALS = "docker-praveen"
    }
    stages {
        stage('Checkout Code') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'github-pk',
usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN')]) {
                    git url:
                    "https://$GIT_USER:$GIT_TOKEN@github.com/viratpk18/Jenkins-
Kubernetes.git", branch: 'main'
                }
            }
        }
    }
}
```

```

    stage('Build & Push Backend Image') {
        steps {
            dir('backend') {
                sh 'docker build -t $BACKEND_IMAGE .'
                withCredentials([usernamePassword(credentialsId: 'docker-praveen',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')) {
                    sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --
password-stdin'
                }
                sh 'docker push $BACKEND_IMAGE'
            }
        }
    }

    stage('Build & Push Frontend Image') {
        steps {
            dir('frontend') {
                sh 'docker build -t $FRONTEND_IMAGE .'
                withCredentials([usernamePassword(credentialsId: 'docker-praveen',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')) {
                    sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --
password-stdin'
                }
                sh 'docker push $FRONTEND_IMAGE'
            }
        }
    }

    stage('Deploy Containers') {
        steps {
            sh 'docker run -d -p 5000:5000 --name $BACKEND_CONTAINER
$BACKEND_IMAGE'
            sh 'docker run -d -p 3000:3000 --name $FRONTEND_CONTAINER
$FRONTEND_IMAGE'
        }
    }

    post {
        success {
            echo "Deployment successful!"
        }
        failure {
            echo "Deployment failed."
        }
    }
}

```

```
mouleesh@DESKTOP-PBBPDI x + v
mouleesh@DESKTOP-PBBPDI$ nano Jenkinsfile

mouleesh@DESKTOP-PBBPDI$ cat Jenkinsfile
pipeline {
  agent any

  environment {
    FRONTEND_IMAGE = "mouleeshwaran28/frontend-app:latest" // Update with your DockerHub username
    BACKEND_IMAGE = "mouleeshwaran28/backend-app:latest"
    FRONTEND_CONTAINER = "frontend-container"
    BACKEND_CONTAINER = "backend-container"
    REGISTRY_CREDENTIALS = "github-mouleesh" // Jenkins credentials ID for Docker login
  }

  stages {
    stage('Checkout Code') {
      steps {
        withCredentials([usernamePassword(credentialsId: 'github-Mouleesh28', usernameVariable: 'GIT_USER', passwordVariable: 'GIT_TOKEN
' )]) {
          git url: "https://github.com/Mouleesh28/Jenkins-kubernetes.git", branch: 'main'
        }
      }
    }

    stage('Build & Push Backend Image') {
      steps {
        dir('backend') {
          sh 'docker build -t $BACKEND_IMAGE .'
          withCredentials([usernamePassword(credentialsId: 'github-mouleesh', usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCK
ER_PASS')]) {
            sh 'echo $DOCKER_PASS | docker login -u $DOCKER_USER --password-stdin'
            sh 'docker push $BACKEND_IMAGE'
          }
        }
      }
    }

    stage('Build & Push Frontend Image') {
```

Step 6: Committing and Pushing the Jenkinsfile

1. Add and commit changes:

```
git add .  
git commit -m "Jenkinsfile added"
```

2. Push to GitHub:

```
git push https://github.com/Mouleesh28/Jenkins-Kubernetes.git
```

```
mouleesh@DESKTOP-PBBPDK2:~/Jenkins-kubernetes$ git push https://Mouleesh28:ghp_3DljDq41QvS105Jkw9Fp1fQ7GY37L084zwb9@github.com/Mouleesh28/Jenkins-kubernetes.git --force  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 920 bytes | 920.00 KiB/s, done.  
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/Mouleesh28/Jenkins-kubernetes.git  
00a4916..7cfc41c main -> main
```

Step 7: Build the Jenkins

The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `localhost:8080/job/Jenkins-kubernetes/2/console`. The Jenkins dashboard is visible, with the 'Console Output' tab selected. The console output shows the following details:

- Started by user Mouleesh
- Obtained Jenkinsfile from git `https://github.com/Mouleesh28/Jenkins-kubernetes.git`
- [Pipeline] Start of Pipeline
- [Pipeline] node
- Running on Jenkins in `/var/lib/jenkins/workspace/Jenkins-kubernetes@2`
- [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
- Cloning the remote Git repository
- Cloning repository `https://github.com/Mouleesh28/Jenkins-kubernetes.git`
- > git init /var/lib/jenkins/workspace/Jenkins-kubernetes@2 # timeout=10
- Fetching upstream changes from `https://github.com/Mouleesh28/Jenkins-kubernetes.git`
- > git --version # timeout=10
- > git --version # 'git version 2.43.0'

