DEVOPS CI/CD PIPELINE

As steps given in the question the screenshots are placed accordingly

1.Implement a CI/CD pipeline using Jenkins multi-branch pipelines

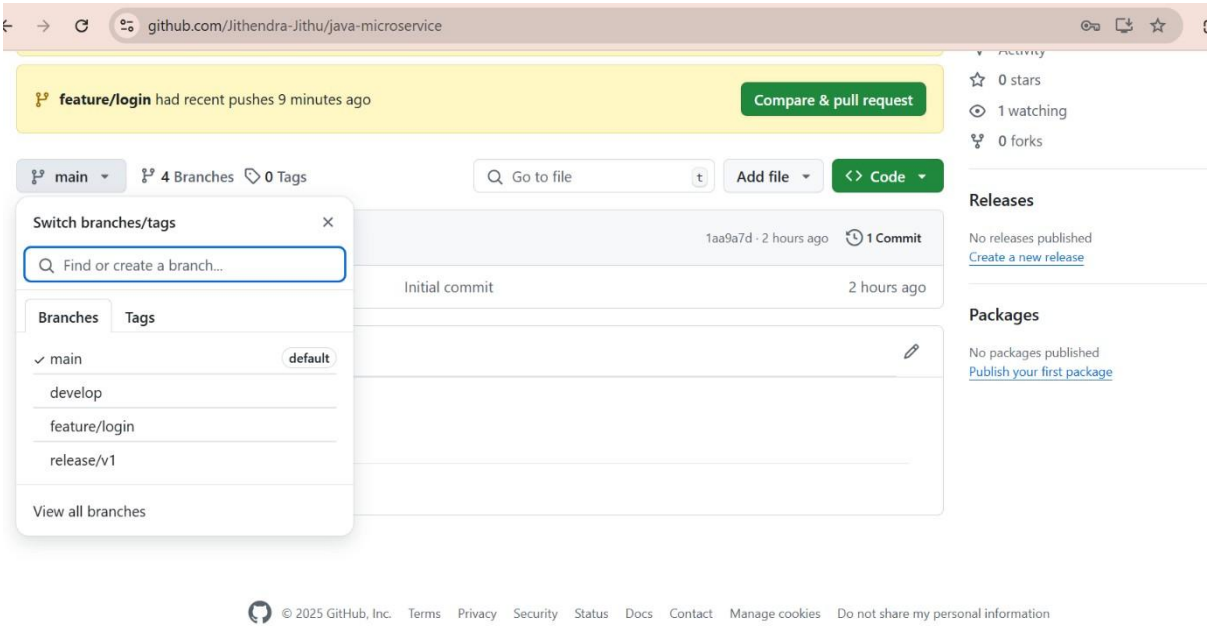
N.Jithendra
289238

This github contains three branches:

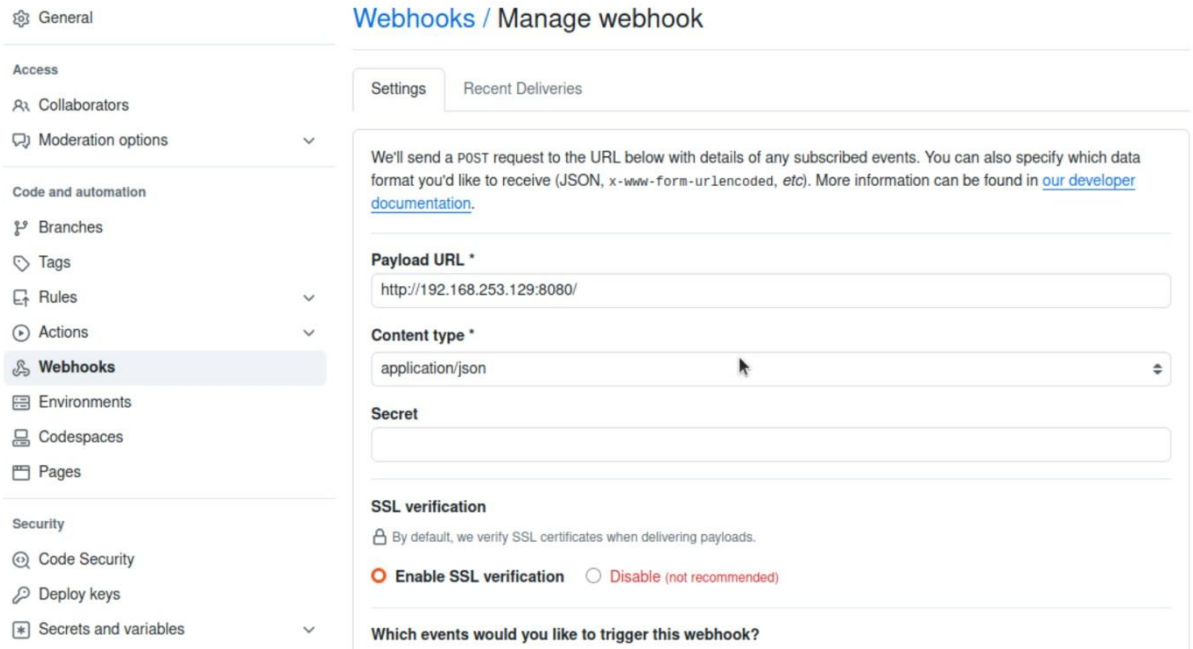
1.Develop

2.Feature/login

3.Release/v1

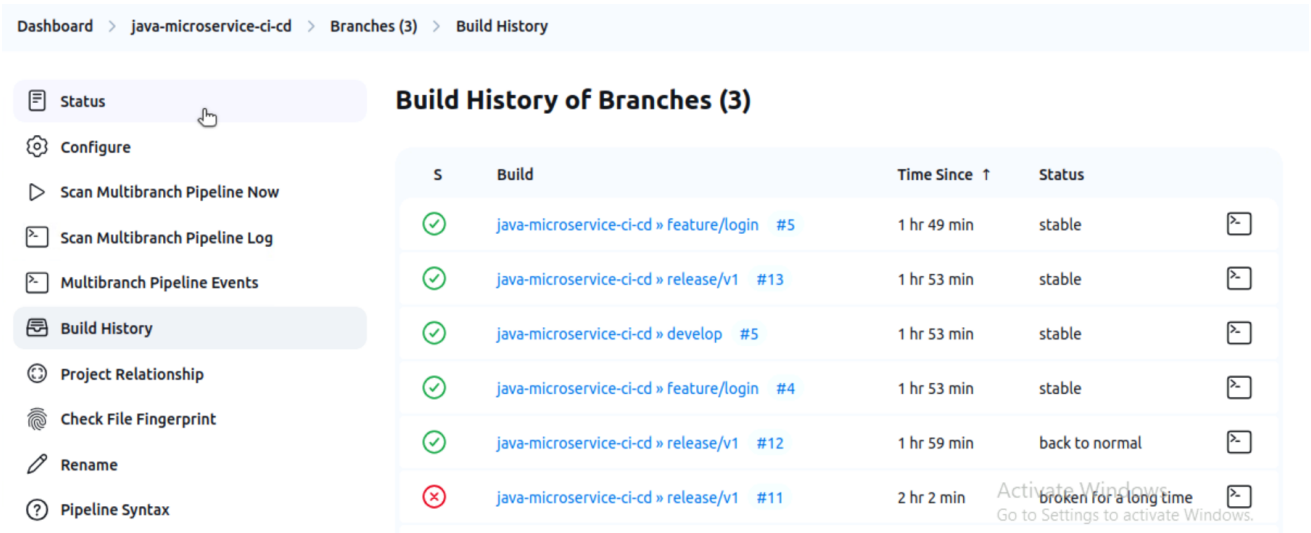


2.Triggers automatically using a GitHub webhook.



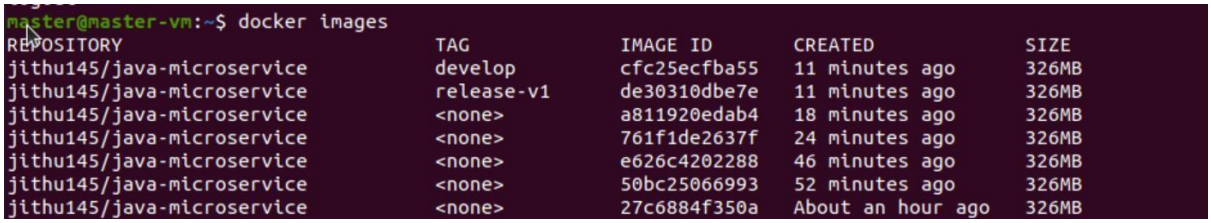
3.Builds and tests the Java application using maven.

Build history:

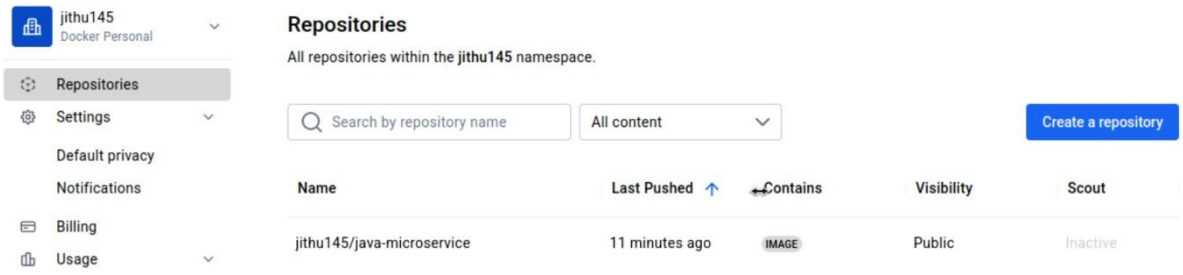


Output of maven:

4. Packages the application into a Docker container and pushes it to an image registry.



5.Docker Containerization and Image Push



Docker file:

Jithendra-Jithu Initial commit for multi-branch Java microservice

Code Blame 3 lines (3 loc) · 93 Bytes

```
1  FROM openjdk:17-jdk-alpine
2  COPY target/*.jar app.jar
3  ENTRYPOINT ["java", "-jar", "/app.jar"]
```

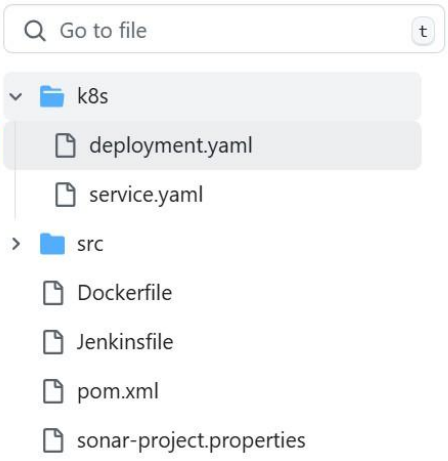
And exposed to 8080

Kubernetes Deployment

i. deployment.yaml (Defines the app deployment and replicas).

ii. service.yaml (Exposes the application internally).

b. Add Kubernetes apply command to Jenkinsfile to enable deployment.



Code Blame 19 lines (19 loc) · 360 Bytes

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: java-microservice
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: java-microservice
10   template:
11     metadata:
12       labels:
13         app: java-microservice
14     spec:
15       containers:
16       - name: java-microservice
17         image: IMAGE_PLACEHOLDER
18         ports:
19         - containerPort: 8080
```

java-microservice / k8s / **service.yaml**

Jithendra-Jithu Initial commit for multi-branch Java microservice

Code | Blame | 12 lines (12 loc) · 194 Bytes

```yaml
apiVersion: v1
kind: Service
metadata:
  name: java-microservice
spec:
  selector:
    app: java-microservice
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: NodePort
```

Jenkins file that is written in the code:

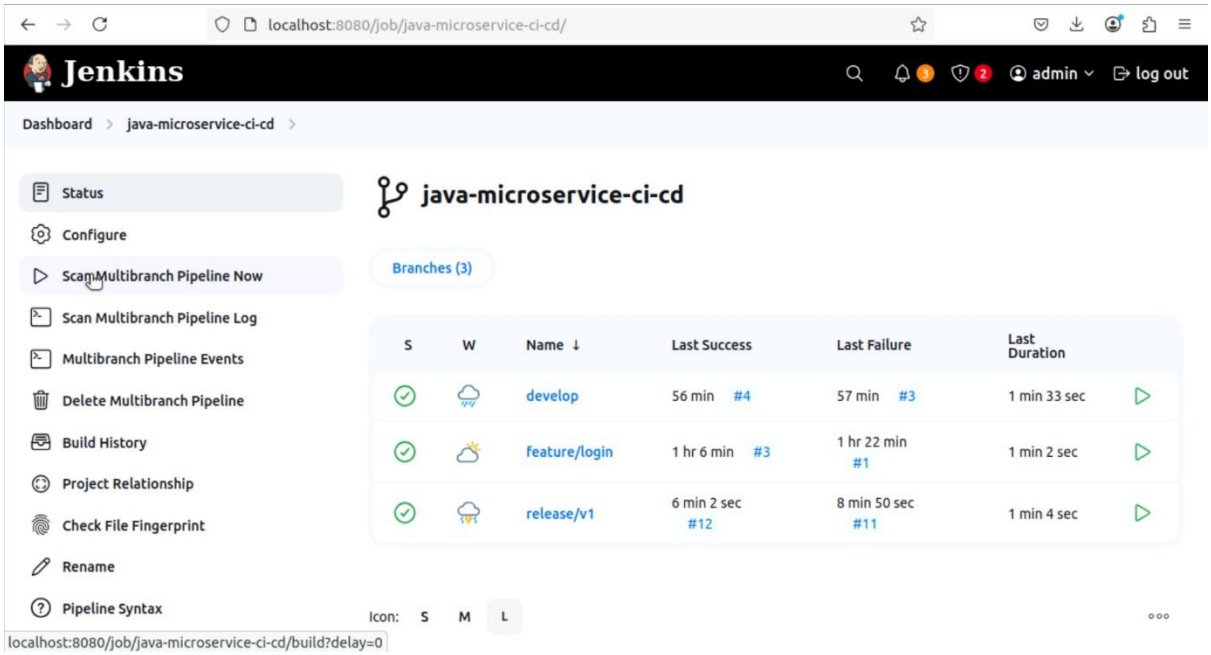
```
pipeline {
    agent any
    environment {
    IMAGE = "docker.io/jithu145/java-microservice:${env.BRANCH_NAME.replaceAll('/', '-')}"
    KUBECONFIG_PATH = "/var/lib/jenkins/.kube/config"
}


    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

        stage('Build') {
            steps {
                sh 'mvn clean package'
            }
        }

        stage('Test') {
            steps {
                sh 'mvn test'
            }
        }
    }

stage('Docker Build & Push') {
    when {
        anyOf {
            branch "develop"
            branch pattern: "release/.*", comparator: "REGEXP"
            branch "main"
        }
    }
    steps {
        withCredentials([usernamePassword(credentialsId: 'fb16b1ba-d2e9-41bb-8654-d00d3b5b61e6', usernameVariable:
            sh '''
                docker build -t ${IMAGE} .
                echo "${PASS}" | docker login -u "${USER}" --password-stdin
                docker push ${IMAGE}
            '''
        }
    }
```

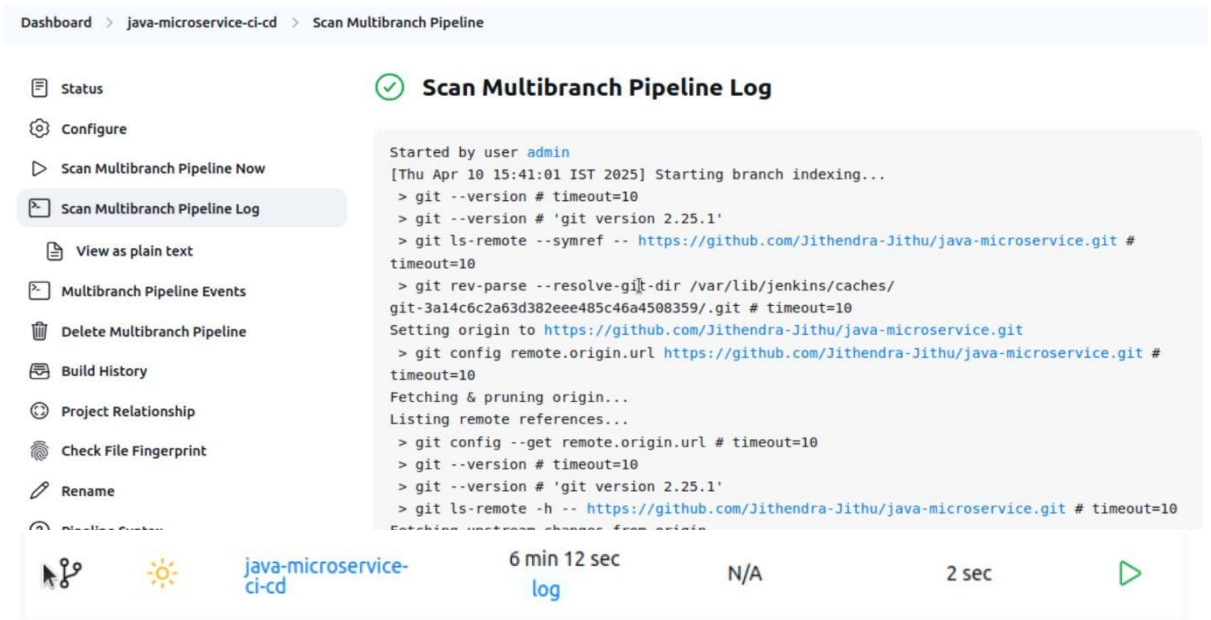
```
stage('Deploy to Staging') {
    when {
        branch pattern: "release/.*", comparator: "REGEXP"
    }
    steps {
        sh '''
            sed -i 's|IMAGE_PLACEHOLDER|${IMAGE}|' k8s/deployment.yaml
            kubectl --kubeconfig=${KUBECONFIG_PATH} apply -f k8s/deployment.yaml
            kubectl --kubeconfig=${KUBECONFIG_PATH} apply -f k8s/service.yaml
        '''
    }
}

stage('Approval & Deploy to Prod') {
    when {
        branch 'main'
    }
    steps {
        input message: "Deploy to Production?"
        sh '''
            sed -i 's|IMAGE_PLACEHOLDER|${IMAGE}|' k8s/deployment.yaml
            kubectl --kubeconfig=${KUBECONFIG_PATH} apply -f k8s/deployment.yaml
            kubectl --kubeconfig=${KUBECONFIG_PATH} apply -f k8s/service.yaml
        '''
```

After executing the Jenkins multibranch pipeline:

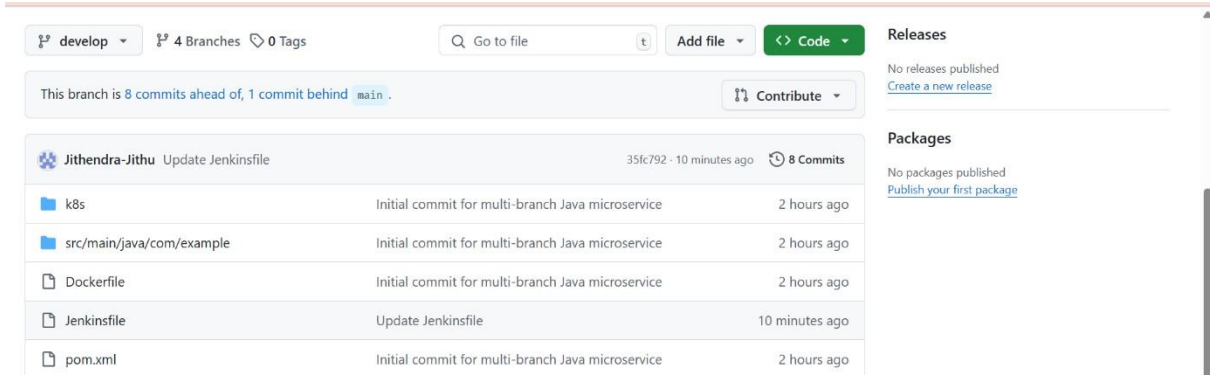
Output:

Total pipeline output:

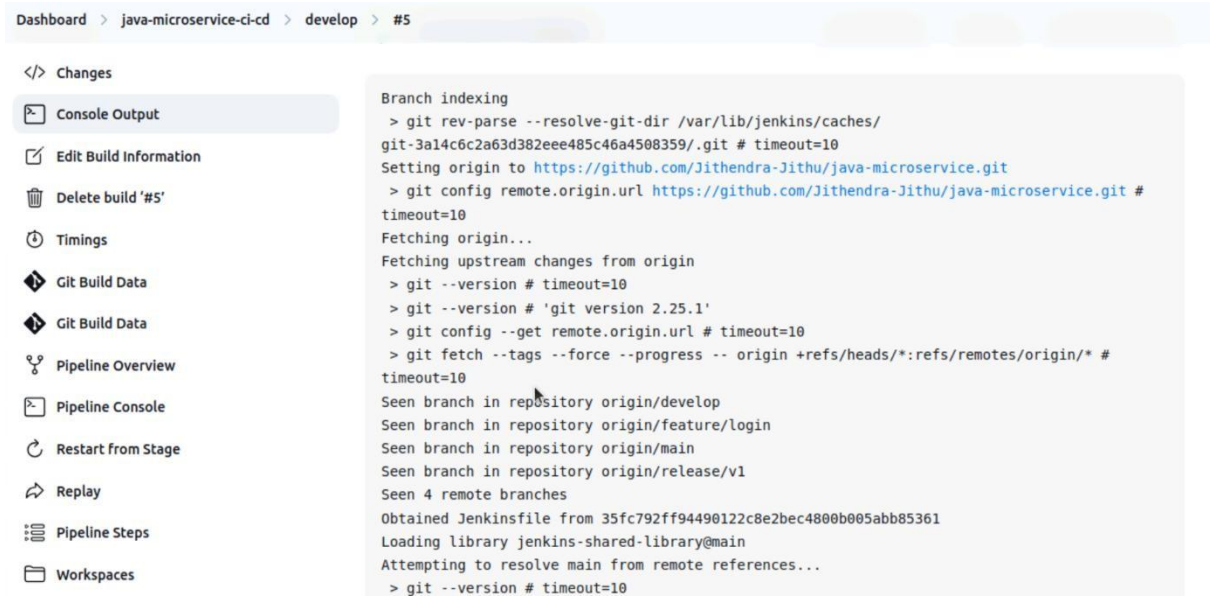




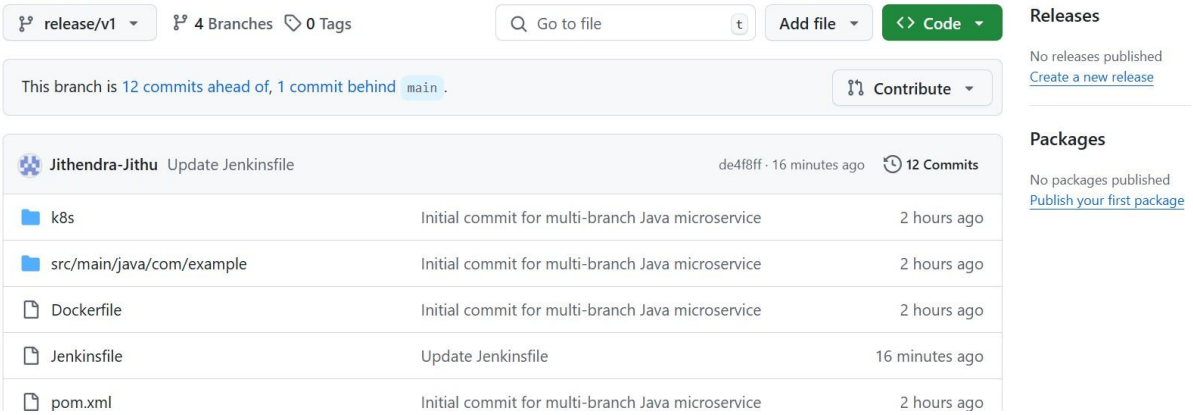
Develop branch:



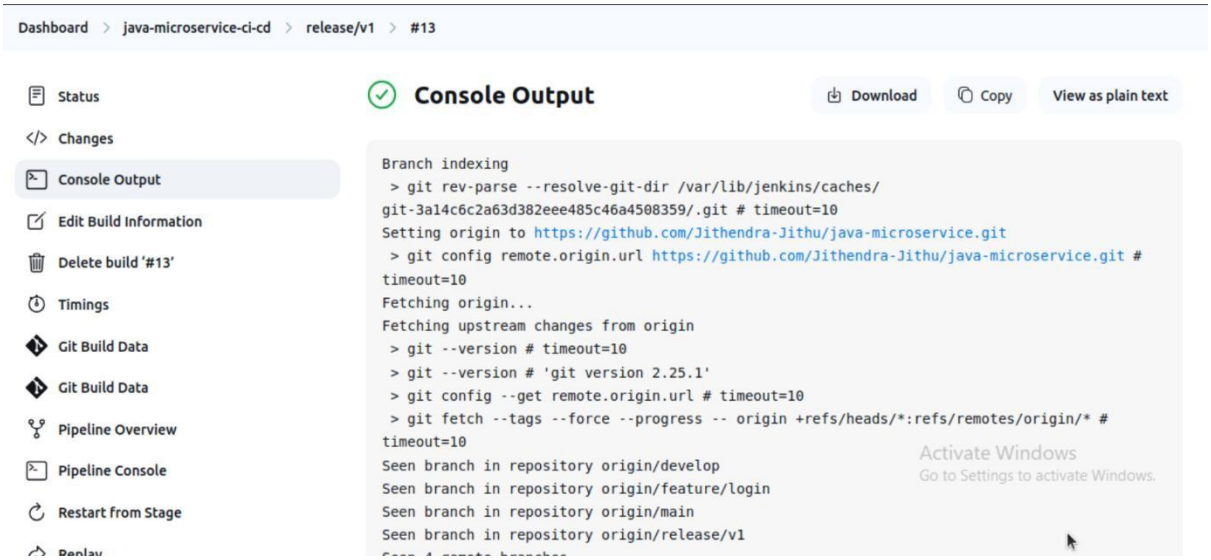
Output:



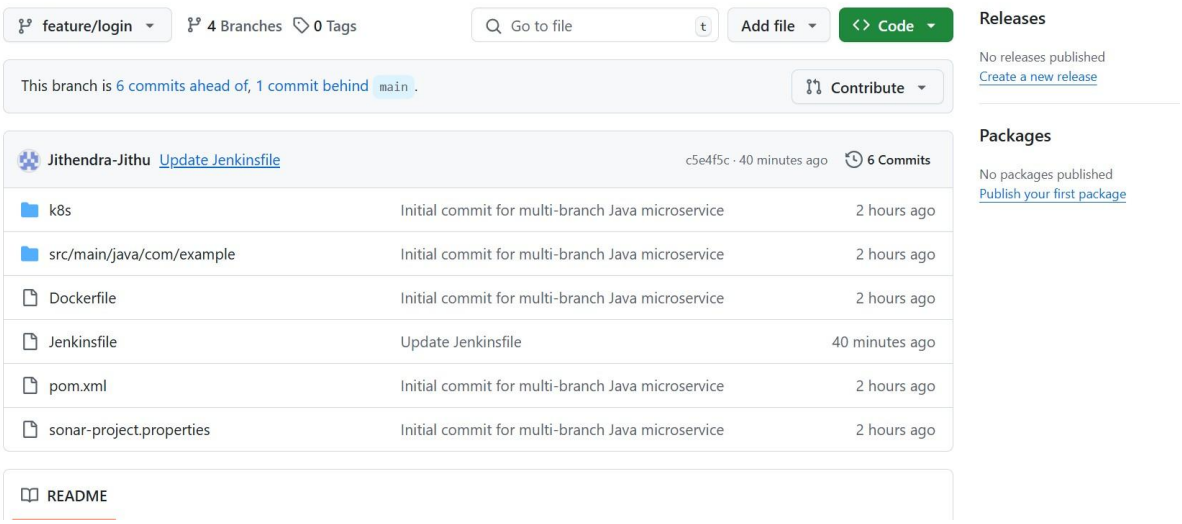
Release branch:



Output:



Feature/login:



Output: