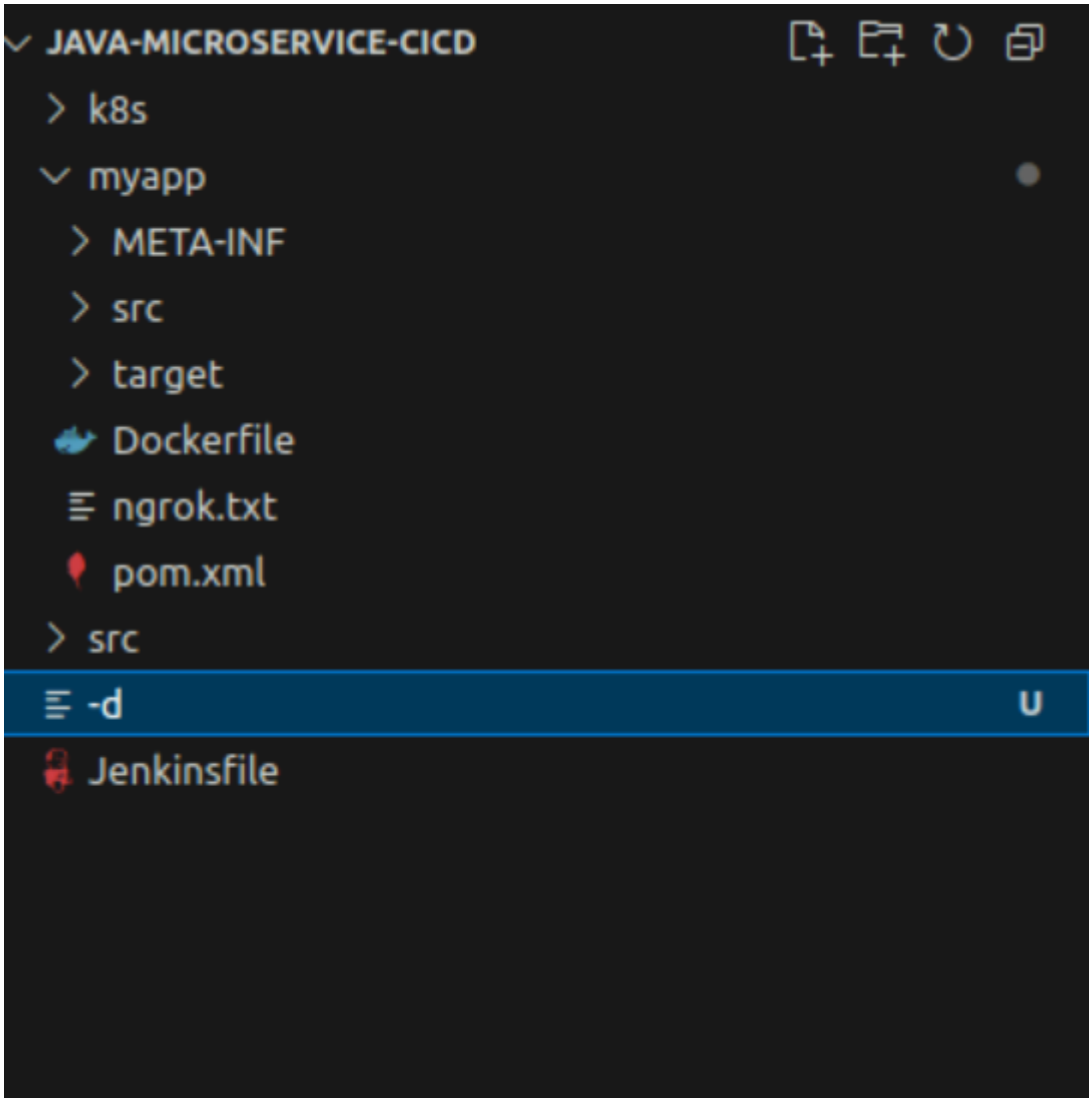
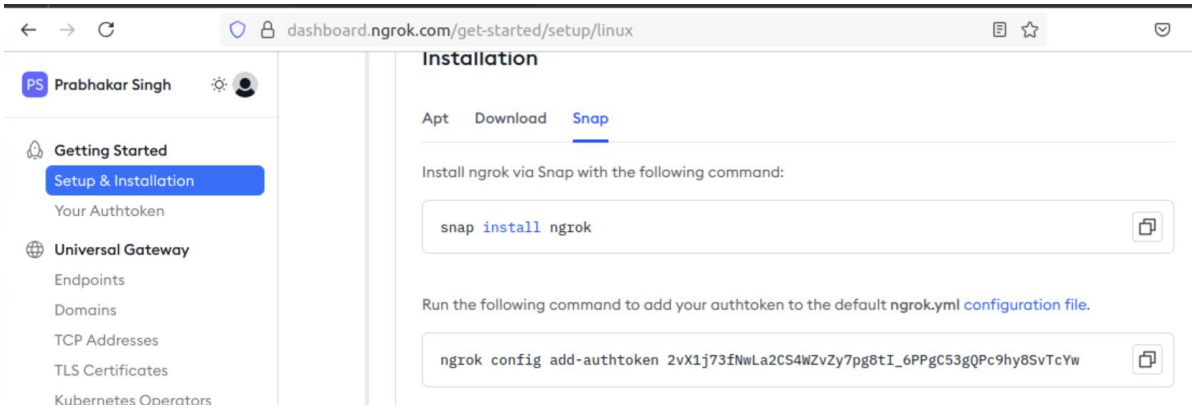
Q) Implement a CI/CD pipeline using Jenkins multi-branch pipelines , Docker for containerization , GitHub Webhooks and Kubernetes for deployment .

Step1) Create a Repository on Github and add remote origin to local folder.



Step 2) Install ngrok on ubuntu terminal

- Sign in to ngrok website and add the auth token provided there.
- Run ngrok 8080 to expose Jenkins




```
master@master-vm:~/Desktop/java-microservice-cicd$ ngrok http 8080
```

```
ngrok                                                                          (Ctrl+C to quit)

👋 Goodbye tunnels, hello Agent Endpoints: https://ngrok.com/r/aep

Session Status                online
Account                       Prabhakar Singh (Plan: Free)
Version                       3.22.0
Region                        India (in)
Latency                       87ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://35a5-103-182-177-42.ngrok-free.app -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              72      0       0.00    0.00    3.75    180.89

HTTP Requests
-------------

17:19:05.743 IST POST /job/microservices_ci-cd/widget/BuildQueueWidget/ajax 200 OK
17:19:05.811 IST POST /job/microservices_ci-cd/widget/ExecutorsWidget/ajax  200 OK
17:19:00.792 IST POST /job/microservices_ci-cd/widget/ExecutorsWidget/ajax  200 OK
17:19:00.728 IST POST /job/microservices_ci-cd/widget/BuildQueueWidget/ajax 200 OK
17:18:55.720 IST POST /job/microservices_ci-cd/widget/BuildQueueWidget/ajax 200 OK
17:18:55.799 IST POST /job/microservices_ci-cd/widget/ExecutorsWidget/ajax  200 OK
17:18:50.597 IST POST /job/microservices_ci-cd/widget/BuildQueueWidget/ajax 200 OK
17:18:50.663 IST POST /job/microservices_ci-cd/widget/ExecutorsWidget/ajax  200 OK
17:18:45.597 IST POST /job/microservices_ci-cd/widget/BuildQueueWidget/ajax 200 OK
17:18:45.661 IST POST /job/microservices_ci-cd/widget/ExecutorsWidget/ajax  200 OK
```

Step3) Create a Webhook Integration

- In Git repo go to settings and select webhooks
- Add payload URL from forward url given by ngrok followed by /github-webhooks
- Configure git webhook to trigger when a commit is pushed and pull request is created

Prsingh9/java-micro × | Webhook · https://3 × | Setup - ngrok × | 192.168.114.70:8081/ × | Branches (3) [micro ×

github.com/Prsingh9/java-microservice-cicd/settings/hooks/540212825

General

Access
- Collaborators
- Moderation options

Code and automation
- Branches
- Tags
- Rules
- Actions
- Webhooks
- Environments
- Codespaces
- Pages

Security
- Code Security
- Deploy keys
- Secrets and variables

Webhooks / Manage webhook

Settings | Recent Deliveries

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *
https://35a5-103-182-177-42.ngrok-free.app/github-webhooks

Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.

Enable SSL verification | Disable (not recommended)

Which events would you like to trigger this webhook?

github.com/Prsingh9/java-microservice-cicd/settings/hooks/540212825

- [ ] **Packages** — GitHub Packages published or updated in a repository.
- [ ] **Page builds** — Pages site built.
- [ ] **Pull request review comments** — Pull request diff comment created, edited, or deleted.
- [ ] **Pull request review threads** — A pull request review thread was resolved or unresolved.
- [ ] **Pull request reviews** — Pull request review submitted, edited, or dismissed.
- [x] **Pull requests** — Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestoned, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.
- [x] **Pushes** — Git push to a repository.
- [ ] **Registry packages** — Registry package published or updated in a repository.
- [ ] **Releases** — Release created, edited, published, unpublished, or deleted.
- [ ] **Repositories** — Repository created, deleted, archived, unarchived, publicized, privatized, edited, renamed, or transferred.
- [ ] **Repository advisories**
- [ ] **Repository imports**

Step 4) Building Java code with maven and generate .jar file

- Create a sample maven file using following command

```
master@master-vm:~/Desktop$ mvn archetype:generate -DgroupId=com.example -DartifactId=myapp -DarchetypeArtifactId=maven-archetype-qu
ickstart -DinteractiveMode=false
```

- Create Jenkins file with branch feature to build image and deploy only when branch is develop

Jenkinsfile

```
pipeline {
    agent any

    environment {
        IMAGE_NAME = "prabsin/myapp"
        K8S= credentials('config')
    }
    stages {
        stage('Build & Test') {
    steps {
        dir('myapp') {
            sh 'mvn clean package'
        }
    }
}


        stage('Docker Build & Push') {
            when {
                branch 'develop'
            }
            steps {
                script {
                    withDockerRegistry([credentialsId: 'docker-hub-credentials', url: '']) {
                         def image = docker.build("${IMAGE_NAME}:${env.BUILD_NUMBER}", "myapp/")
                        image.push()
                        image.push('latest')
                    }
                }
            }
        }

        stage('Deploy to Kubernetes') {
    when {
        branch 'develop'
    }
    steps {
```

```
    stages {
        stage('Build & Test') {
    steps {
        dir('myapp') {
            sh 'mvn clean package'
        }
    }
}


        stage('Docker Build & Push') {
            when {
                branch 'develop'
            }
            steps {
                script {
                    withDockerRegistry([credentialsId: 'docker-hub-credentials', url: '']) {
                        def image = docker.build("${IMAGE_NAME}:${env.BUILD_NUMBER}", "myapp/")
                        image.push()
                        image.push('latest')
                    }
                }
            }
        }

        stage('Deploy to Kubernetes') {
    when {
        branch 'develop'
    }
    steps {
        withCredentials([file(credentialsId: 'config', variable: 'KUBECONFIG_FILE')]) {
            sh '''
                export KUBECONFIG=$KUBECONFIG_FILE
                kubectl apply -f k8s/deployment.yaml
                kubectl apply -f k8s/service.yaml
            '''
        }
    }
```

- Write docker file and Kubernetes file
- To generate .jar file we can run locally mvn clean build to check

```
master@master-vm:~/Desktop/java-microservice-cicd/myapp$ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< com.example:myapp >------------------------
[INFO] Building myapp 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ myapp ---
[INFO] Deleting /home/master/Desktop/java-microservice-cicd/myapp/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ myapp ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/master/Desktop/java-microservice-cicd/myapp/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ myapp ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/master/Desktop/java-microservice-cicd/myapp/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ myapp ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory /home/master/Desktop/java-microservice-cicd/myapp/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ myapp ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file to /home/master/Desktop/java-microservice-cicd/myapp/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ myapp ---
[INFO] Surefire report directory: /home/master/Desktop/java-microservice-cicd/myapp/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.112 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ myapp ---
[INFO] Building jar: /home/master/Desktop/java-microservice-cicd/myapp/target/myapp-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.727 s
[INFO] Finished at: 2025-04-10T15:31:08+05:30
[INFO] ------------------------------------------------------------------------
```

Step 5) Create the Docker file

- Build the image locally to check and run

```
Dockerfile ×
myapp > Dockerfile
1  FROM openjdk:21-slim
2  COPY target/myapp-1.0-SNAPSHOT.jar /app.jar
3  EXPOSE 8081
4  ENTRYPOINT ["java", "-jar", "/app.jar"]
5
```

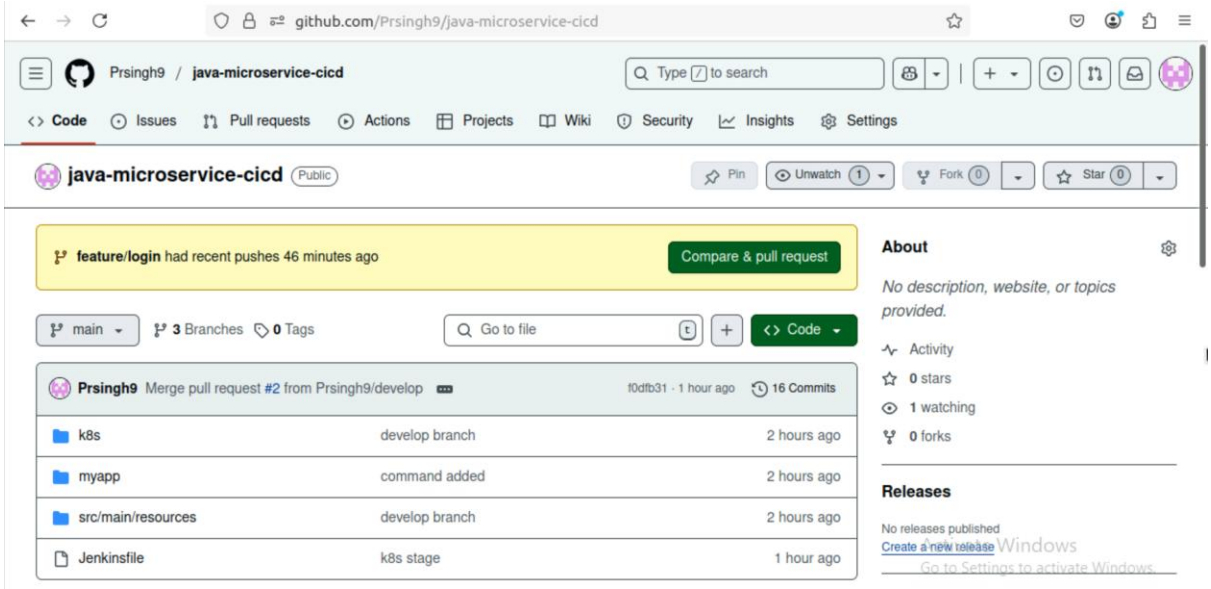
```
master@master-vm:~/Desktop/java-microservice-cicd/myapp$ docker build -t myapp:test .
[+] Building 4.9s (8/8) FINISHED                                                    docker:default
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 154B                                                          0.0s
 => [internal] load metadata for docker.io/library/openjdk:21-slim                            4.6s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                0.0s
 => [internal] load .dockerignore                                                             0.0s
 => => transferring context: 2B                                                               0.0s
 => [internal] load build context                                                             0.0s
 => => transferring context: 2.30kB                                                           0.0s
 => CACHED [1/2] FROM docker.io/library/openjdk:21-slim@sha256:7072053847a8a05d7f3a14ebc778a90b38c50ce7e8f199382128a533851606  0.0s
 => [2/2] COPY target/myapp-1.0-SNAPSHOT.jar /app.jar                                         0.1s
 => exporting to image                                                                        0.1s
 => => exporting layers                                                                       0.0s
 => => writing image sha256:2dfc4d08596371be19d89577f7f15ab26b74f1f25cb275218149f47435086f2b  0.0s
 => => naming to docker.io/library/myapp:test                                                 0.0s
master@master-vm:~/Desktop/java-microservice-cicd/myapp$ docker run --rm myapp:test
Hello World!
```

Step 6) Write Kubernetes deployment.yaml and service.yaml file

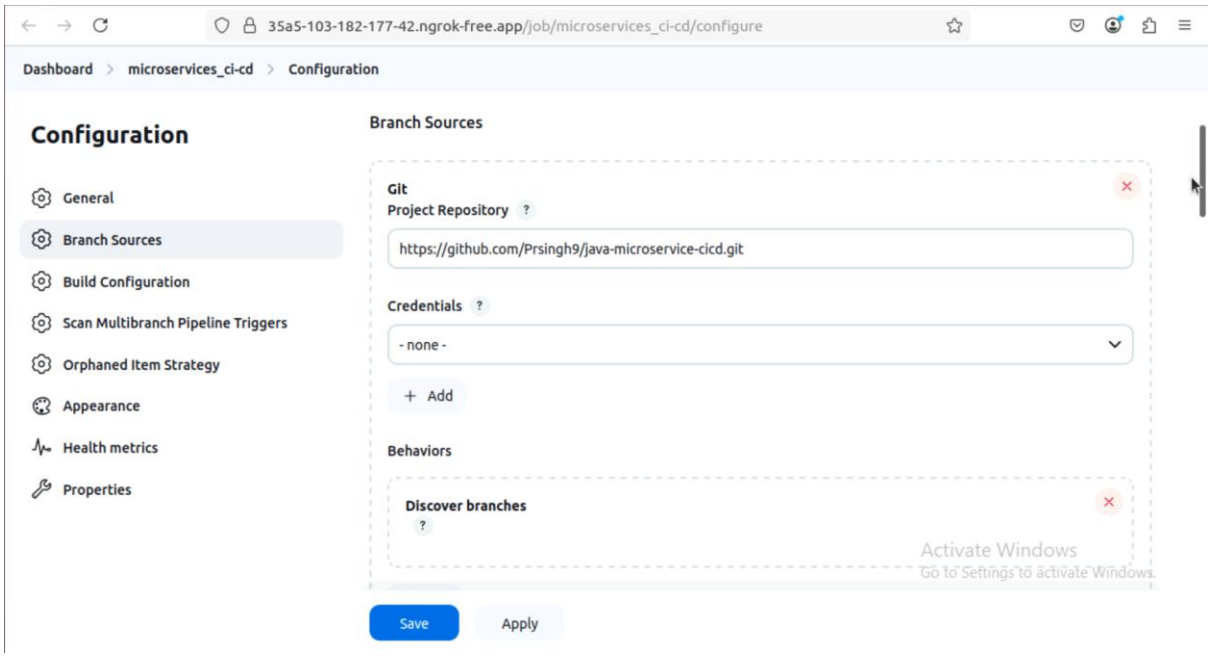
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp
        image: prabsin/myapp:latest
        ports:
        - containerPort: 8081
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  selector:
    app: myapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8081
  type: ClusterIP
```

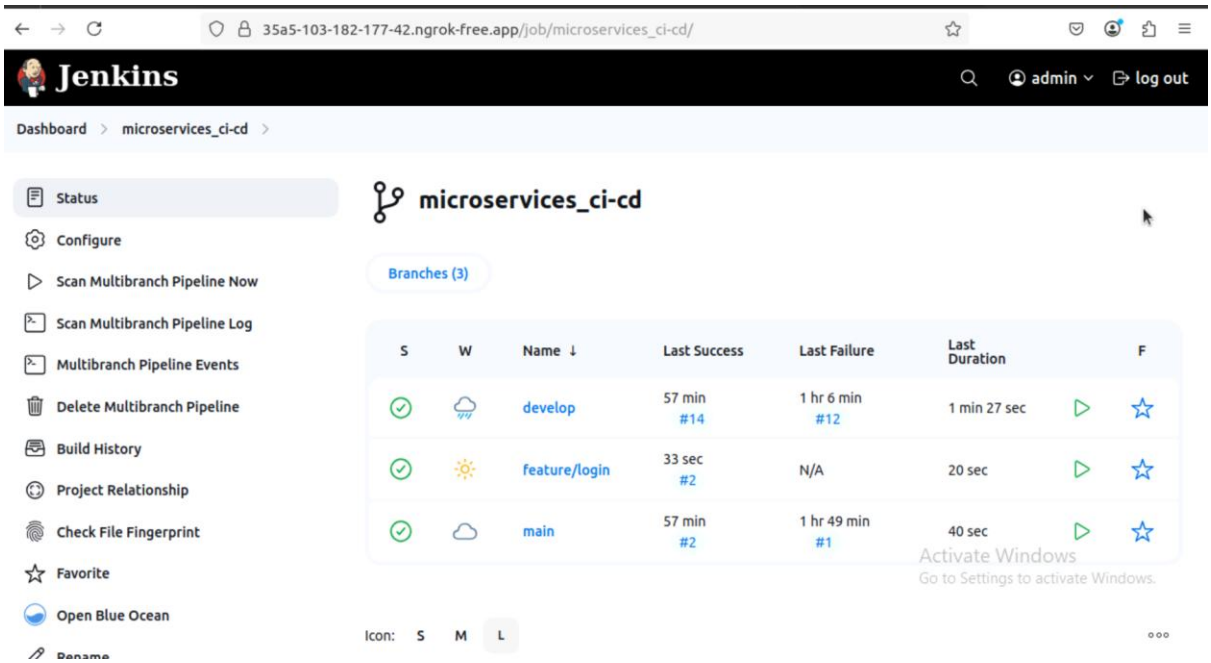
Step 7) Push code to the github and with different branches as well

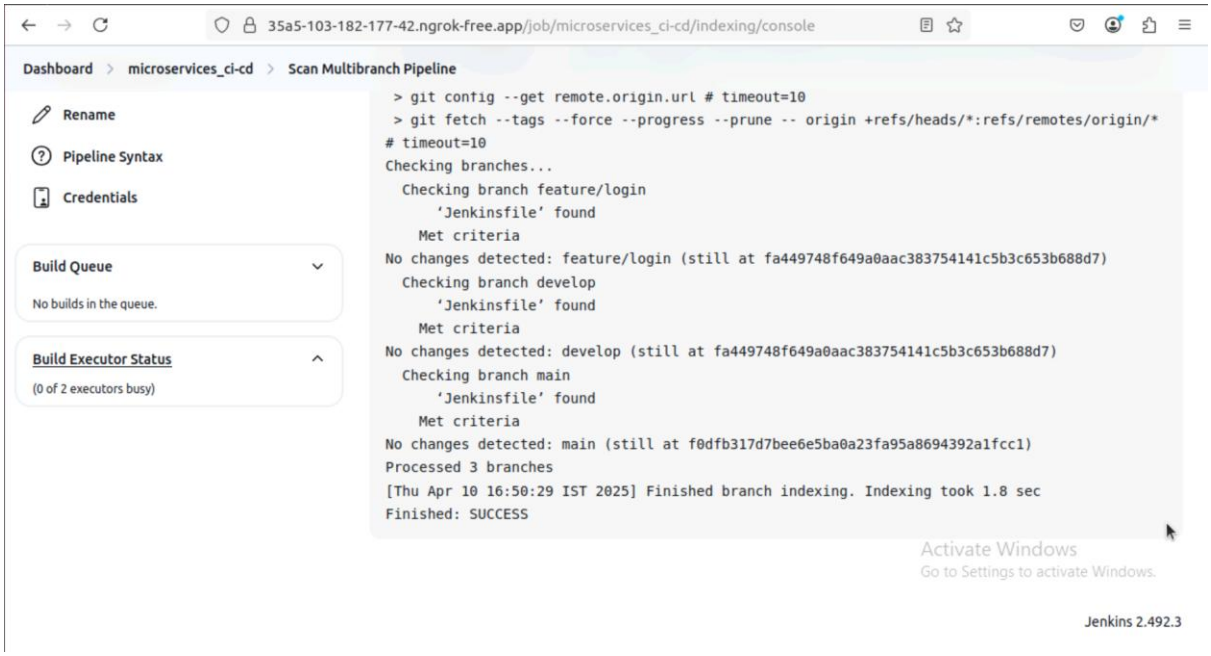


Step 8) In Jenkins select Multibranch pipeline and give the following configuration

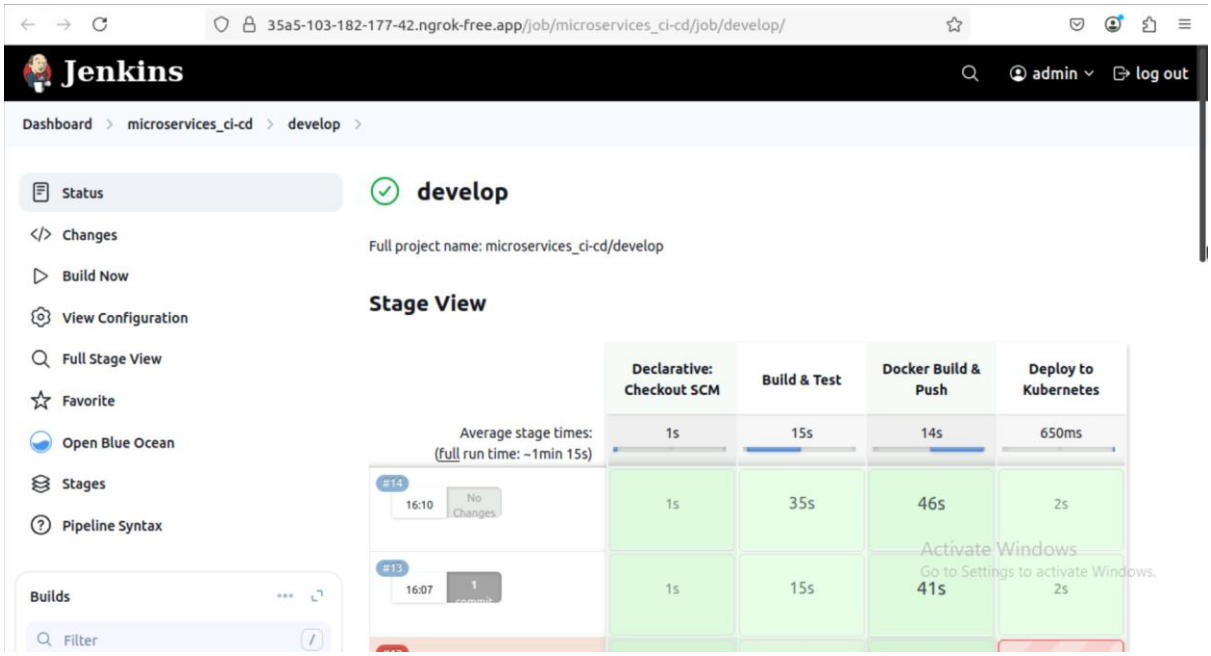


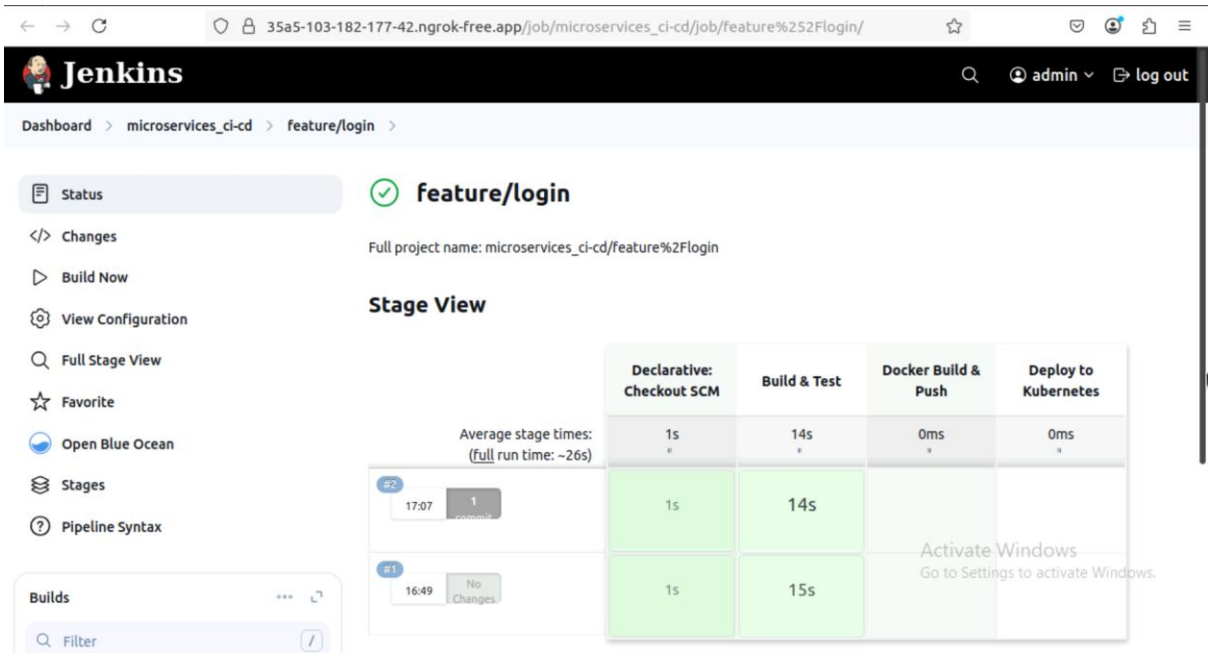
Step 9) Automatic build will be trigger when codes are pushed or pull request is made to github





Expected Output






```
master@master-vm:~/Desktop$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
myapp-deployment-77cd6464c7-4dl4n     1/1     Running   0          96m
myapp-deployment-77cd6464c7-mz77f     1/1     Running   0          96m
```

```
master@master-vm:~/Desktop$ kubectl get svc
NAME                  TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes            ClusterIP      10.96.0.1        <none>        443/TCP        2d
myapp-service         ClusterIP      10.109.255.180   <none>        80/TCP         109m
nginx                 NodePort       10.104.90.123    <none>        80:31841/TCP   41h
sample-ci-cd-service  LoadBalancer   10.104.178.151   <pending>     80:30488/TCP   31h
master@master-vm:~/Desktop$ kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   2/2     2            2           109m
```