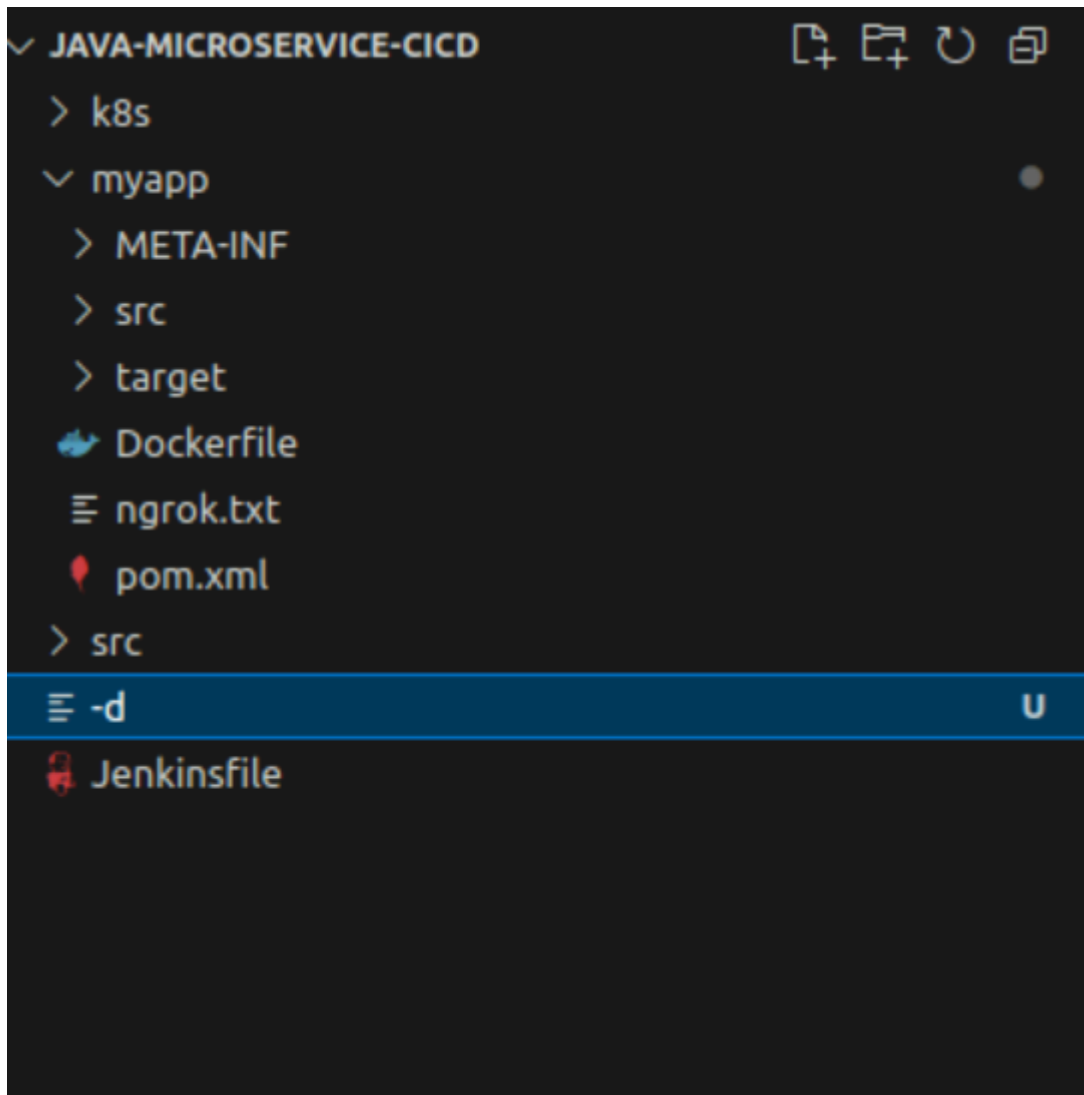


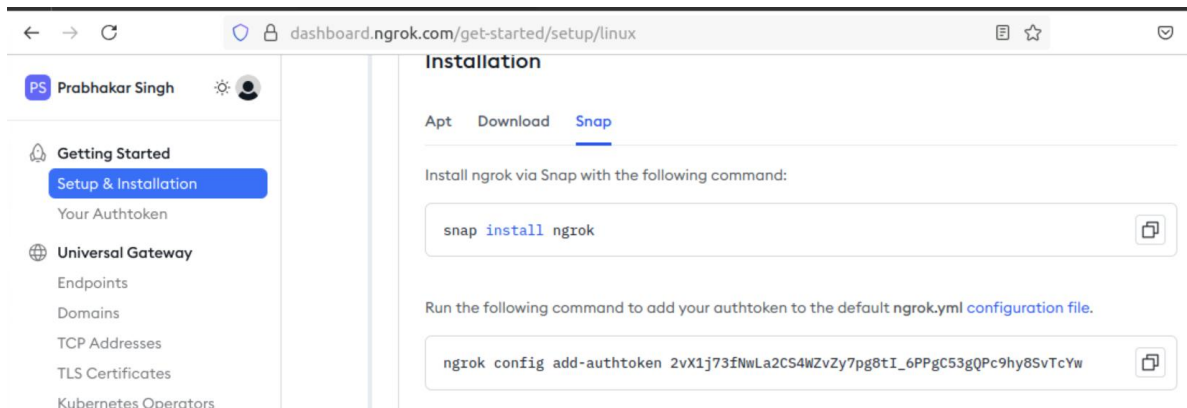
Q) Implement a CI/CD pipeline using Jenkins multi-branch pipelines , Docker for containerization , GitHub Webhooks and Kubernetes for deployment .

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

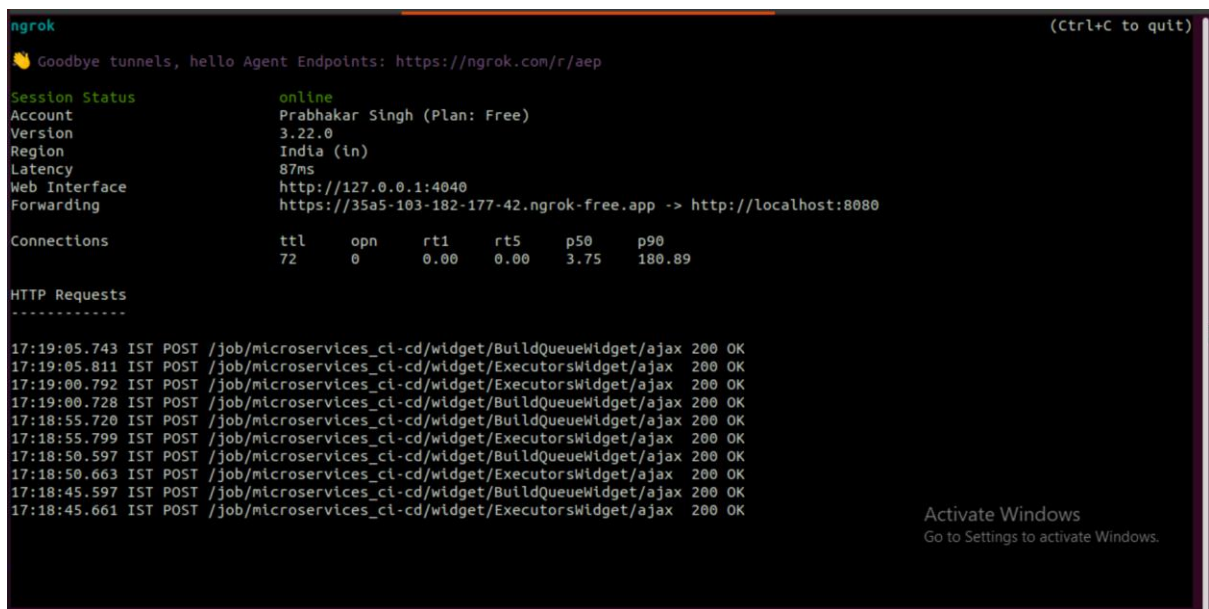


Step 2) Install ngrok on ubuntu terminal

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



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



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-microservice-cicd

Webhook - https://

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

SettingsRecent 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.

☒ **Pull requests**
Pull request assigned, auto merge disabled, auto merge enabled, closed, converted to draft, demilestoned, dequeued, edited, enqueued, labeled, locked, milestone, opened, ready for review, reopened, review request removed, review requested, synchronized, unassigned, unlabeled, or unlocked.

☒ **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-vn:~/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
1 pipeline {
2     agent any
3
4     environment {
5         IMAGE_NAME = "prabsin/myapp"
6         K8S= credentials('config')
7     }
8
9     stages {
10         stage('Build & Test') {
11             steps {
12                 dir('myapp') {
13                     sh 'mvn clean package'
14                 }
15             }
16         }
17
18         stage('Docker Build & Push') {
19             when {
20                 branch 'develop'
21             }
22             steps {
23                 script {
24                     withDockerRegistry([credentialsId: 'docker-hub-credentials', url: '']) {
25                         def image = docker.build("${IMAGE_NAME}:${env.BUILD_NUMBER}", "myapp/")
26                         image.push()
27                         image.push('latest')
28                     }
29                 }
30             }
31         }
32
33         stage('Deploy to Kubernetes') {
34             when {
35                 branch 'develop'
36             }
37             steps {
38                 sh 'kubectl apply -f deployment.yaml'

Activated  
Go to Settings


```

```

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] -----
[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 x
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

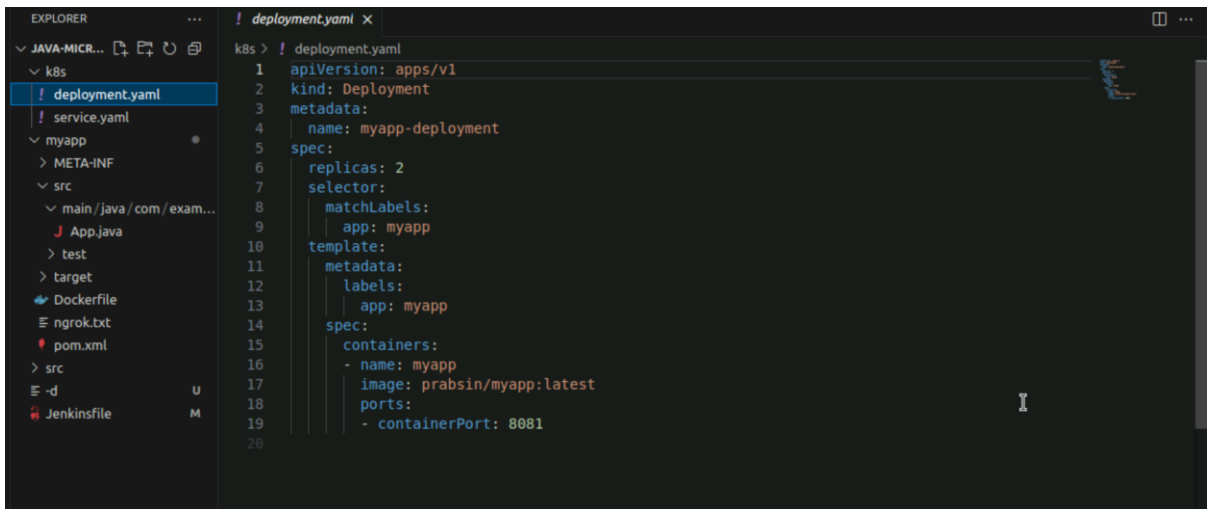
```

```

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@sha256:7072653847a8a05d7f3a14ebc778a90b38c50ce7e8f199382128a533851606 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

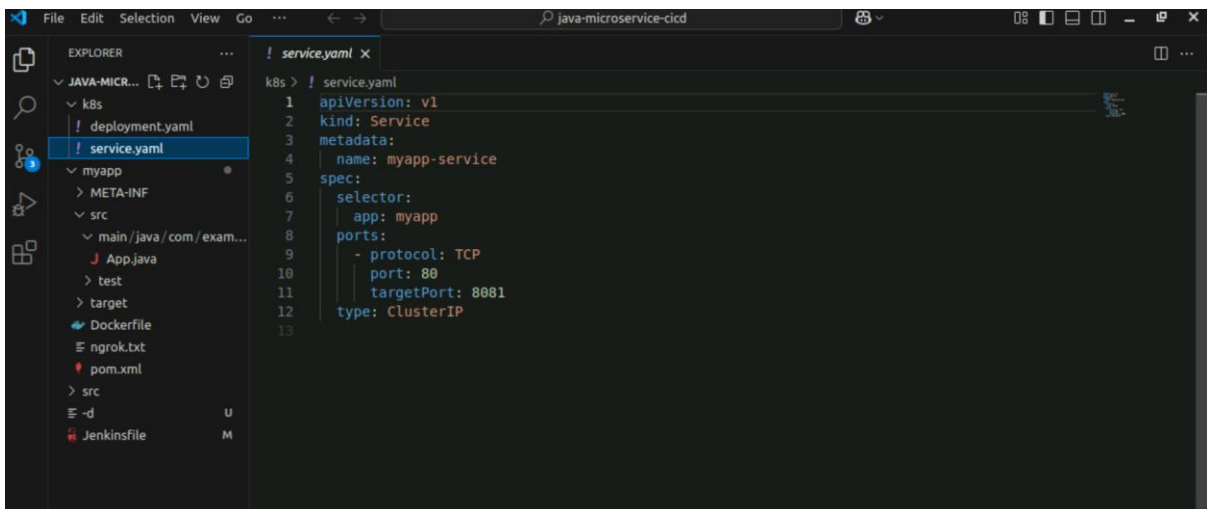


```

EXPLORER
  JAVA-MICR...
  k8s
    ! deployment.yaml
    ! service.yaml
  myapp
    META-INF
    src
      main/java/com/exam...
        App.java
      test
      target
      Dockerfile
      ngrok.txt
      pom.xml
    src
    -d
    Jenkinsfile

k8s > ! deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: myapp-deployment
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: myapp
10   template:
11     metadata:
12       labels:
13         app: myapp
14     spec:
15       containers:
16       - name: myapp
17         image: prabsin/myapp:latest
18         ports:
19         - containerPort: 8081
20

```



```

EXPLORER
  JAVA-MICR...
  k8s
    ! deployment.yaml
    ! service.yaml
  myapp
    META-INF
    src
      main/java/com/exam...
        App.java
      test
      target
      Dockerfile
      ngrok.txt
      pom.xml
    src
    -d
    Jenkinsfile

k8s > ! service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: myapp-service
5  spec:
6    selector:
7      app: myapp
8    ports:
9      - protocol: TCP
10        port: 80
11        targetPort: 8081
12    type: ClusterIP
13

```


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

The screenshot shows the GitHub repository page for 'java-microservice-cicd' by user 'Prsingh9'. The repository is public and has 0 stars, 0 forks, and 1 watching. The main content area shows a merge pull request #2 from 'Prsingh9/develop' to 'develop' branch, with 16 commits. Below this, a table lists recent commits:

| Commit | Branch | Time |
|--------------------|----------------|-------------|
| k8s | develop branch | 2 hours ago |
| myapp | command added | 2 hours ago |
| src/main/resources | develop branch | 2 hours ago |
| Jenkinsfile | k8s stage | 1 hour ago |

On the right, the 'About' section is empty, and the 'Releases' section shows 'No releases published' with a link to 'Create a new release'.

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

The screenshot shows the Jenkins Configuration page for a Multibranch pipeline. The left sidebar contains a list of configuration options: General, Branch Sources, Build Configuration, Scan Multibranch Pipeline Triggers, Orphaned Item Strategy, Appearance, Health metrics, and Properties. The 'Branch Sources' section is selected and expanded, showing the following configuration:

- Git** (Project Repository ?)
- Project Repository**: `https://github.com/Prsingh9/java-microservice-cicd.git`
- Credentials** (?): `- none -`
- Behaviors**: `Discover branches` (?)

At the bottom, there are 'Save' and 'Apply' buttons. A watermark 'Activate Windows Go to Settings to activate Windows.' is visible in the bottom right corner.

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

The screenshot displays the Jenkins web interface for a job named 'microservices_ci-cd'. The top navigation bar includes the Jenkins logo, a search icon, and user information (admin) with a 'log out' link. The breadcrumb trail shows 'Dashboard > microservices_ci-cd >'. The left sidebar contains a list of actions: Status, Configure, Scan Multibranch Pipeline Now, Scan Multibranch Pipeline Log, Multibranch Pipeline Events, Delete Multibranch Pipeline, Build History, Project Relationship, Check File Fingerprint, Favorite, Open Blue Ocean, and Rename. The main content area shows the 'microservices_ci-cd' job details, including a 'Branches (3)' section with a table of branch build history.

| S | W | Name ↓ | Last Success | Last Failure | Last Duration | | F |
|---|---|---------------|--------------|----------------|---------------|---|---|
| ✓ | ☁ | develop | 57 min #14 | 1 hr 6 min #12 | 1 min 27 sec | ▶ | ☆ |
| ✓ | ☀ | feature/login | 33 sec #2 | N/A | 20 sec | ▶ | ☆ |
| ✓ | ☁ | main | 57 min #2 | 1 hr 49 min #1 | 40 sec | ▶ | ☆ |

Below the table, there is a section for 'Icon: S M L' and a 'Rename' button. The console log is visible, showing the output of a 'Scan Multibranch Pipeline' job. The log indicates that the pipeline successfully indexed three branches (develop, feature/login, and main) and that no changes were detected for any of them. The indexing process took 1.8 seconds and finished successfully.

```
> git config --get remote.origin.url # timeout=10
> git fetch --tags --force --progress --prune -- origin +refs/heads/*:refs/remotes/origin/*
# timeout=10
Checking branches...
Checking branch feature/login
'Jenkinsfile' found
Met criteria
No changes detected: feature/login (still at fa449748f649a0aac383754141c5b3c653b688d7)
Checking branch develop
'Jenkinsfile' found
Met criteria
No changes detected: develop (still at fa449748f649a0aac383754141c5b3c653b688d7)
Checking branch main
'Jenkinsfile' found
Met criteria
No changes detected: main (still at f0dfb317d7bee6e5ba0a23fa95a8694392a1fcc1)
Processed 3 branches
[Thu Apr 10 16:50:29 IST 2025] Finished branch indexing. Indexing took 1.8 sec
Finished: SUCCESS
```

The bottom right corner of the interface shows the Jenkins version: 2.492.3.

Expected Output

← → ↻

35a5-103-182-177-42.ngrok-free.app/job/microservices_ci-cd/job/develop/


☆

🔒

👤

🔖

☰

 **Jenkins**

🔍

admin ▾

🔗 log out

Dashboard > microservices_ci-cd > develop >

Status

</> Changes

▶ Build Now

⚙️ View Configuration

🔍 Full Stage View

☆ Favorite

🌊 Open Blue Ocean

📁 Stages

🔍 Pipeline Syntax

Builds

Filter

✓ **develop**

Full project name: microservices_ci-cd/develop

Stage View

Average stage times:
(full run time: ~1min 15s)

| | Declarative: Checkout SCM | Build & Test | Docker Build & Push | Deploy to Kubernetes |
|----------------------------|------------------------------|--------------|------------------------|-------------------------|
| #14 16:10 No Changes | 1s | 35s | 46s | 2s |
| #13 16:07 1 | 1s | 15s | 41s | 2s |

← → ↻

35a5-103-182-177-42.ngrok-free.app/job/microservices_ci-cd/job/feature%252Flogin/


☆

🔒

👤

🔖

☰

 **Jenkins**

🔍

admin ▾

🔗 log out

Dashboard > microservices_ci-cd > feature/login >

Status

</> Changes

▶ Build Now

⚙️ View Configuration

🔍 Full Stage View

☆ Favorite

🌊 Open Blue Ocean

📁 Stages

🔍 Pipeline Syntax

Builds

Filter

✓ **feature/login**

Full project name: microservices_ci-cd/feature%2Flogin

Stage View

Average stage times:
(full run time: ~26s)

| | Declarative: Checkout SCM | Build & Test | Docker Build & Push | Deploy to Kubernetes |
|---------------------------|------------------------------|--------------|------------------------|-------------------------|
| #2 17:07 1 | 1s | 14s | 0ms | 0ms |
| #1 16:49 No Changes | 1s | 15s | | |

```
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
```

