Minikube Deployment Guide

1. Granting Jenkins User Passwordless sudo Access

```
bash
CopyEdit
jenkins ALL=(ALL) NOPASSWD: ALL
```

- This line is added to the `/etc/sudoers` file using `visudo`.
- Allows the **Jenkins** user to execute commands as root without requiring a password.
- Useful for **CI/CD automation**, enabling Jenkins to restart services or execute privileged commands.

2. Restarting SSH Service

```
bash
CopyEdit
sudo systemctl restart ssh.service
sudo systemctl restart sshd.service
```

- Restarts the **SSH server** to apply changes.
- If `ssh.service` is not found, `sshd.service` might be used instead.
- Some systems (like Ubuntu) manage SSH under `ssh` instead of `sshd`.

3. Updating Package Lists & Installing OpenSSH

```
bash
CopyEdit
sudo apt update
sudo apt install openssh-server
```

- `sudo apt update`: Updates the package list to ensure the latest versions are installed.
- `sudo apt install openssh-server`: Installs the SSH server, allowing **remote access** to the system.

4. Checking SSH Service Status

```
bash
CopyEdit
sudo systemctl restart ssh
sudo systemctl status ssh
```

- `restart ssh`: Restarts the SSH service.
- `status ssh`: Displays the current state of SSH (active, inactive, or failed).

5. Retrieving Minikube Certificate

```
bash
CopyEdit
cat /home/david/.minikube/ca.crt | base64 -w 0; echo
```

- Reads the **Minikube CA certificate**.
- Encodes the certificate into **Base64 format** for use in Kubernetes configurations.
- Ensures correct output formatting.

📌 Summary of Commands

Command	Purpose
`jenkins ALL=(ALL) NOPASSWD: ALL`	Allows Jenkins to execute sudo commands without a password.
`sudo systemctl restart ssh.service`	Restarts SSH service if available.
`sudo systemctl restart sshd.service`	Restarts SSH daemon (if service name is `sshd`).
`sudo apt update && sudo apt install openssh-server`	Updates package lists and installs OpenSSH.

`sudo systemctl restart ssh`	Ensures SSH service is restarted.
`sudo systemctl status ssh`	Checks if SSH is running.
`` `cat /home/david/.minikube/ca.crt ``	base64 -w 0; echo `` ` ``

Jenkins Pipeline for Minikube Deployment

```
groovy
CopyEdit
pipeline {
    agent any
    stages {
        stage('SCM') {
            steps {
                git branch: 'main'
            }
        }
        stage('Build - Clean') {
            steps {
                sh "mvn clean"
            }
        }
        stage('Build - Validate') {
            steps {
                sh "mvn validate"
            }
        }
        stage('Build - Compile') {
            steps {
                sh "mvn compile"
            }
        }
        stage('Build - Test') {
            steps {
                sh "mvn test"
            }
        }
        stage('Build - Package') {
```

```
            steps {
                sh "mvn package"
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    sh 'docker build -t my-app:latest .'
                }
            }
        }
        stage('Push to Docker Hub') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'Docker_cred',
url: 'https://index.docker.io/v1/') {
                        sh 'docker push my-app:latest'
                    }
                }
            }
        }
        stage('Deploy Application') {
            steps {
                withKubeConfig(
                    caCertificate: '',
                    clusterName: 'minikube',
                    contextName: 'minikube',
                    credentialsId: 'mukubeconfig_011',
                    namespace: '',
                    restrictKubeConfigAccess: false,
                    serverUrl: 'https://192.168.49.2:8443'
                ) {
                    sh 'kubectl apply -f deployment.yml --
validate=false'
                }
            }
        }
        stage('Test Deployment') {
            steps {
```

```
                withKubeConfig(
                    caCertificate: '',
                    clusterName: 'minikube',
                    contextName: 'minikube',
                    credentialsId: 'mukubeconfig_011',
                    namespace: '',
                    restrictKubeConfigAccess: false,
                    serverUrl: 'https://192.168.49.2:8443'
                ) {
                    sh 'minikube service my-service --url | xargs
curl'
                }
            }
        }
    }
}
```

Pipeline Breakdown

1. Agent Definition

```
groovy
CopyEdit
agent any
```

- Runs the pipeline on any available agent.

2. Stages Breakdown

SCM (Source Code Management)

```
groovy
CopyEdit
stage('SCM') {
    steps {
        git branch: 'main'
```

```
    }
}
```

- Pulls the **source code** from the Git repository.
- ✅ Fixed: Added **branch: 'main'** to specify the branch.

Build Stages

1. **Clean**

```
groovy
CopyEdit
stage('Build - Clean') {
    steps {
        sh "mvn clean"
    }
}
```

   a. Cleans previous build artifacts.

2. **Validate**

```
groovy
CopyEdit
stage('Build - Validate') {
    steps {
        sh "mvn validate"
    }
}
```

   a. Validates project structure and configurations.

3. **Compile**

```
groovy
CopyEdit
stage('Build - Compile') {
    steps {
        sh "mvn compile"
```

```
    }
}
```

a. Compiles the source code.

4. **Test**

```
groovy
CopyEdit
stage('Build - Test') {
    steps {
        sh "mvn test"
    }
}
```

a. Runs **unit tests**.

5. **Package**

```
groovy
CopyEdit
stage('Build - Package') {
    steps {
        sh "mvn package"
    }
}
```

a. Generates the **final application package** (JAR/WAR).

Docker Build and Push

1. **Build Docker Image**

```
groovy
CopyEdit
stage('Build Docker Image') {
    steps {
        script {
            sh 'docker build -t my-app:latest .'
        }
```

```
    }
}
```

a. ✅ Fixed: Specified image name (`my-app:latest`).

2. **Push Image to Docker Hub**

```
groovy
CopyEdit
stage('Push to Docker Hub') {
    steps {
        script {
            withDockerRegistry(credentialsId: 'Docker_cred', url:
'https://index.docker.io/v1/') {
                sh 'docker push my-app:latest'
            }
        }
    }
}
```

a. ✅ Fixed: Specified the **image name** for pushing.

Deploy and Test

1. **Deploy to Kubernetes (Minikube)**

```
groovy
CopyEdit
stage('Deploy Application') {
    steps {
        withKubeConfig(
            caCertificate: '',
            clusterName: 'minikube',
            contextName: 'minikube',
            credentialsId: 'mukubeconfig_011',
            namespace: '',
            restrictKubeConfigAccess: false,
            serverUrl: 'https://192.168.49.2:8443'
        ) {
```

```
            sh 'kubectl apply -f deployment.yml --validate=false'
        }
    }
}
```

a. Deploys the **application** to Minikube.

2. **Test Deployment**

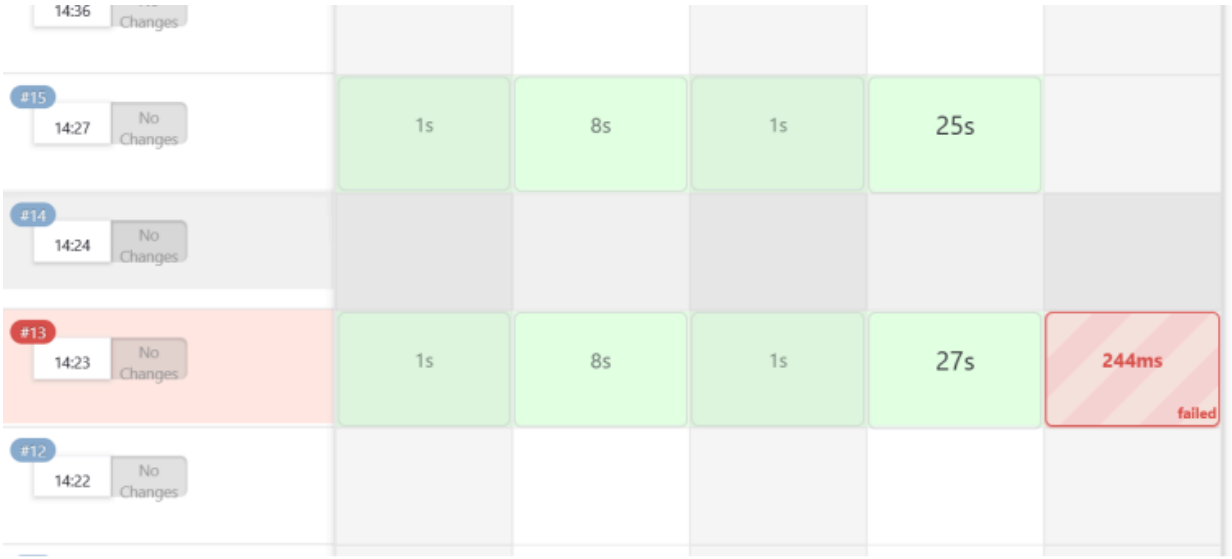
```
groovy
CopyEdit
stage('Test Deployment') {
    steps {
        withKubeConfig(
            caCertificate: '',
            clusterName: 'minikube',
            contextName: 'minikube',
            credentialsId: 'mukubeconfig_011',
            namespace: '',
            restrictKubeConfigAccess: false,
            serverUrl: 'https://192.168.49.2:8443'
        ) {
            sh 'minikube service my-service --url | xargs curl'
        }
    }
}
```

a. Retrieves the **service URL** and tests the deployment.

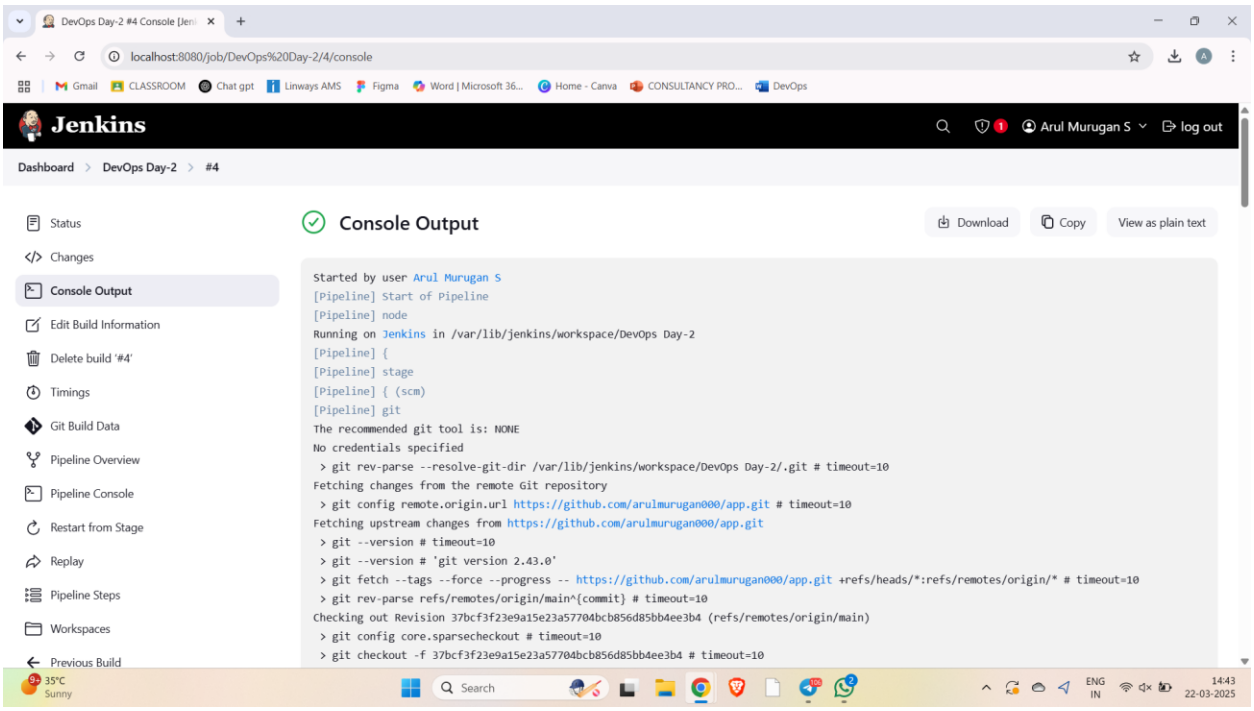
📌 Summary of Pipeline Execution

1. **Clone** repository from Git.
2. Run **Maven build** steps: clean, validate, compile, test, package.
3. **Build Docker image** for the application.
4. **Push Docker image** to Docker Hub.
5. **Deploy application** to Minikube.

6. **Test the deployed application**.


14:36
Changes
#15
14:27
No Changes
1s
8s
1s
25s
#14
14:24
No Changes
#13
14:23
No Changes
1s
8s
1s
27s
244ms
failed
#12
14:22
No Changes


Dashboard
java application
#15
5f70bf18a086: Layer already exists
43c9f8a1dd61: Layer already exists
bc05267c613b: Layer already exists
4e5b554b7345: Layer already exists
4b7c01ed0534: Layer already exists
3359bc3d7a6a: Layer already exists
39cf0ac89a5a: Layer already exists
f844dcf94898: Layer already exists
1c296840251f: Pushed
latest: digest: sha256:a9ee2a9cc1e3c8aef97f5df0ec4a945d53acf6f579dd95258fef3000bae1f798 size: 2409
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS


DevOps Day-2 #4 Console [Jen
localhost:8080/job/DevOps%20Day-2/4/console
Gmail
CLASSROOM
Chat gpt
Linways AMS
Figma
Word | Microsoft 36...
Home - Canva
CONSULTANCY PRO...
DevOps
Jenkins
Arul Murugan S
log out
Dashboard
DevOps Day-2
#4
Status
Changes
Console Output
Edit Build Information
Delete build '#4'
Timings
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build
Console Output
Download
Copy
View as plain text
Started by user Arul Murugan S
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/DevOps Day-2
[Pipeline] {
[Pipeline] stage
[Pipeline] { (scm)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/DevOps Day-2/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/arulmurugan000/app.git # timeout=10
Fetching upstream changes from https://github.com/arulmurugan000/app.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/arulmurugan000/app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 37bcf3f23e9a15e23a57704bcb856d85bb4ee3b4 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 37bcf3f23e9a15e23a57704bcb856d85bb4ee3b4 # timeout=10
35°C
Sunny
Search
ENG
IN
14:43
22-03-2025