DevOps Day-5

Granting Jenkins Sudo Privileges – The jenkins ALL=(ALL) NOPASSWD: ALL entry in the sudoers file allows the Jenkins user to run any command without a password prompt.

Restarting SSH Services – Commands like sudo systemctl restart ssh.service and sudo systemctl restart sshd.service restart the SSH service, ensuring remote login functionality.

Installing OpenSSH Server – The commands sudo apt update and sudo apt install openssh-server update package lists and install the OpenSSH server for secure remote access.

Checking SSH Service Status – sudo systemctl status ssh checks if the SSH service is running and displays its current status.

Systemd Service File Lookup – ls /etc/systemd/system/sshd.service or ls /usr/lib/systemd/system/sshd.service helps locate the SSH daemon's systemd service file.

Reloading Systemd Daemon – sudo systemctl daemon-reload ensures that systemd picks up changes in service configurations without requiring a reboot.

Encoding Minikube Certificate – cat /home/david/.minikube/ca.crt | base64 -w 0; echo encodes the Minikube CA certificate in base64 format, likely for authentication.

Changing Docker Socket Permissions – sudo chmod 666 /var/run/docker.sock grants read and write access to all users for Docker's Unix socket, allowing non-root users to interact with Docker.

Deploying Kubernetes Resources – sh 'kubectl apply -f deployment.yml --validate=false' applies a Kubernetes deployment file, ignoring validation errors.

Accessing Minikube Service – minikube service my-service --url | xargs curl retrieves the Minikube service URL and sends an HTTP request to test its accessibility.

Commands:

jenkins ALL=(ALL) NOPASSWD: ALL

sudo systemctl restart ssh.service

sudo systemctl restart sshd.service

sudo apt update

sudo apt install openssh-server

sudo systemctl restart ssh

sudo systemctl status ssh

ls /etc/systemd/system/sshd.service or ls /usr/lib/systemd/system/sshd.service

sudo systemctl daemon-reload

sudo systemctl status ssh

sudo systemctl restart ssh.service

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

sudo chmod 666 /var/run/docker.sock

https://192.168.39.226:8443

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

minikube service my-service --url | xargs curl

Pipeline codes:

pipeline {

agent any

environment {

DOCKER_CREDENTIALS = credentials('docker-hub-cred') // Docker Hub Credentials ID

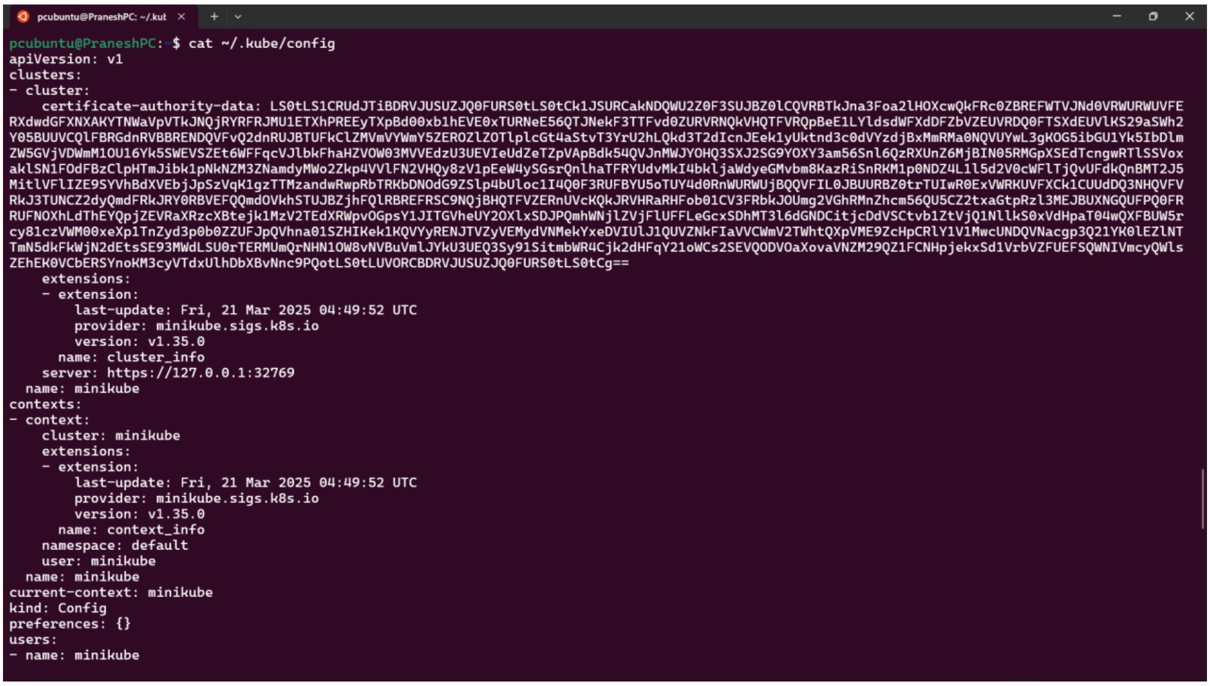
}

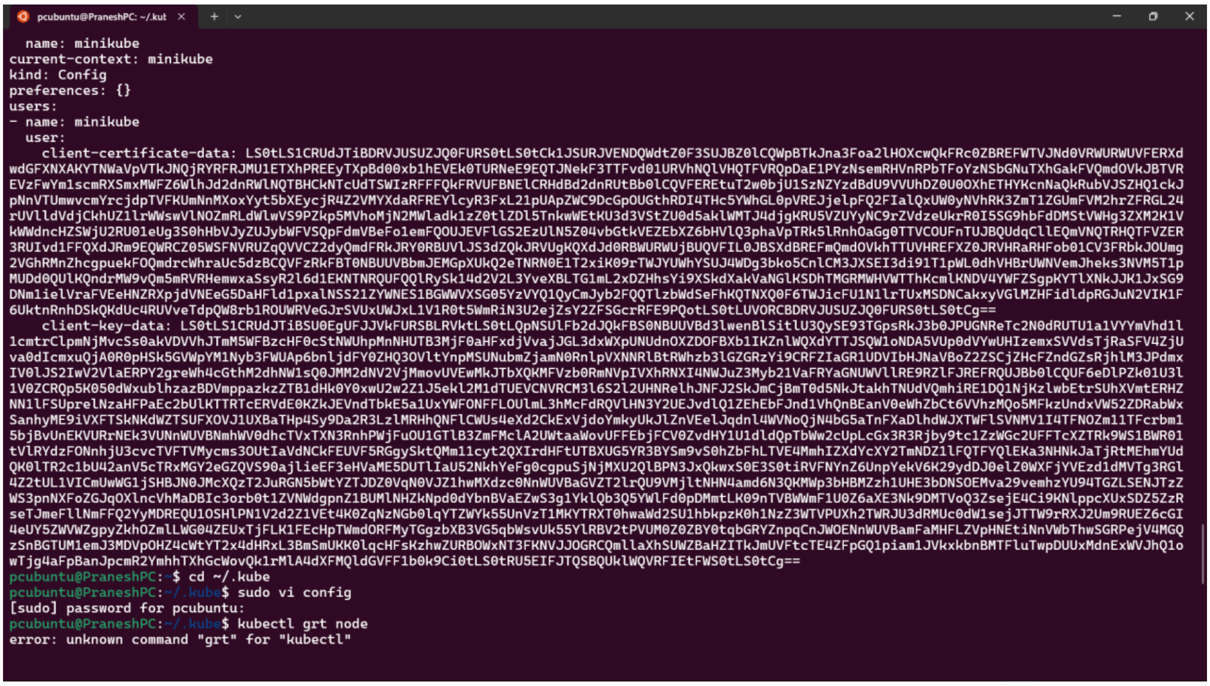
```
stages {
    stage('SCM') {
        steps {
            git branch: 'main', url: '<https://github.com/MugeshS-04/guvidevopsday1.git>'
        }
    }

    stage('Build') {
        steps {
            sh "mvn clean"
            sh "mvn install"
        }
    }

    stage('Build Docker Image') {
        steps {
            script {
                sh 'docker build -t mugeshs04/guvidevopsday1 .'
            }
        }
    }

    stage('Push to Docker Hub') {
        steps {
            script {
                docker.withRegistry('<https://index.docker.io/v1/>', 'docker-hub-cred') {
                    sh 'docker push mugeshs04/guvidevopsday1'
                }
```

```
          }
        }
    }
}

}

pipeline {
agent any

stages {
    stage('SCM') {
        steps {
            git branch: 'main', url: '<https://github.com/PraneshC2005/DevOps_simple-web-app.git>'
        }
    }
    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 to images') {
        steps {
        script{
            sh "docker build -t praneshc/webapplication ."
        }
        }
    }
    stage('docker push hub') {
        steps {
        script{
           withDockerRegistry(credentialsId: 'cred-2', url: '<https://index.docker.io/v1/>') {
            sh 'docker push praneshc/webapplication'
        }
        }
        }
    }
}}
```

Edit Config File:





Development.yml:

apiVersion: apps/v1

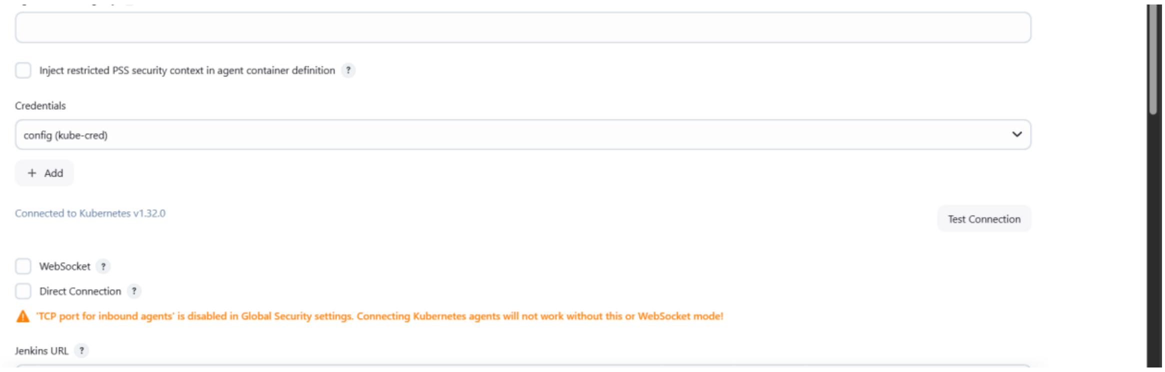
kind: Deployment

metadata:

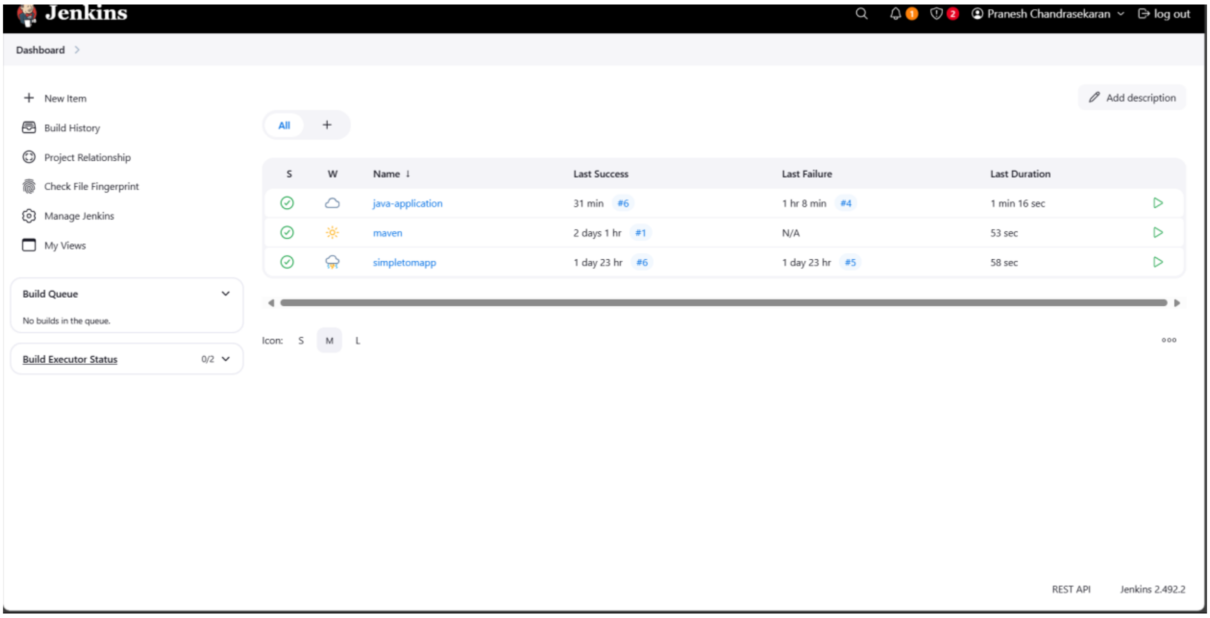
```yaml
  name: my-deploy
  labels:
    name: my-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      apptype: web-backend
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        apptype: web-backend
    spec:
      containers:
      - name: my-app
        image: praneshc/webapplication
        ports:
        - containerPort: 9000

---

apiVersion: v1
kind: Service
metadata:
```

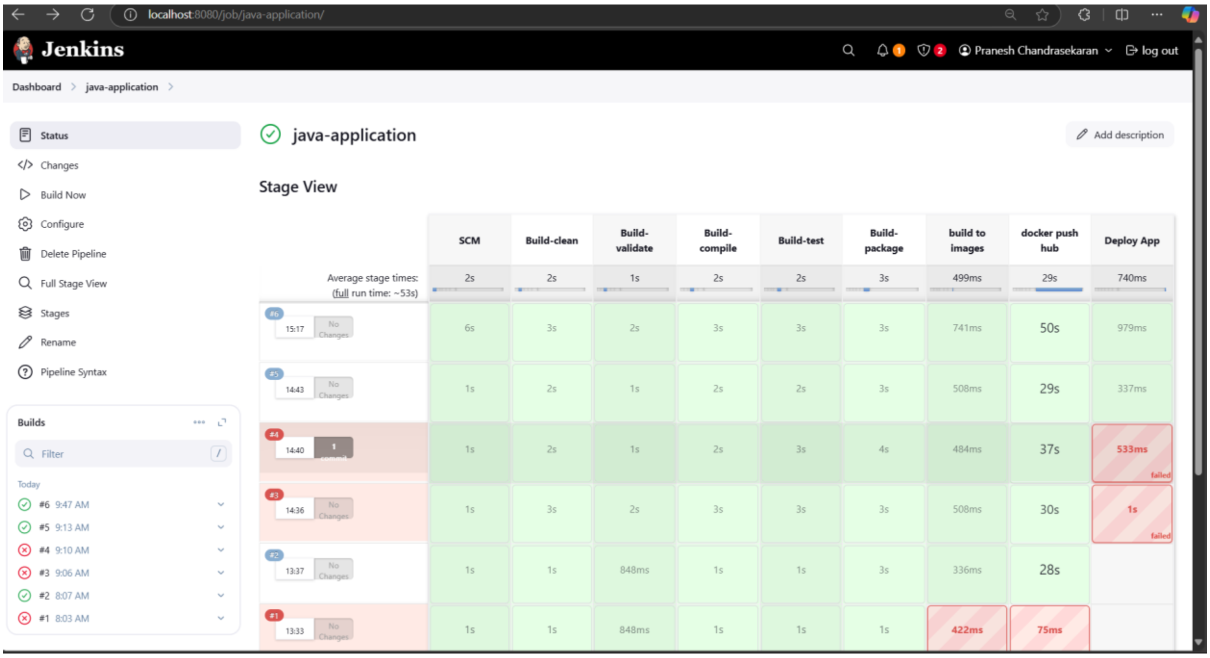
```
  name: my-service
  labels:
    app: my-service
spec:
  type: NodePort
  ports:
    - port: 9000
      targetPort: 8080
      nodePort: 30002
  selector:
    apptype: web-backend
```

Cloud Testing:



Create a new application in java ,a pipeline project





Hello World Output:

pcubuntu@PraneshPC: ~/.kut

```
🚜  Pulling base image v0.0.46 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
pcubuntu@PraneshPC:~/.kube$ kubectl get pod
NAME                          READY   STATUS    RESTARTS       AGE
my-app                        1/1     Running   1 (127m ago)   5h8m
my-deploy-795d5b5d8c-tpm7h    1/1     Running   0              58m
my-pod                        1/1     Running   1 (127m ago)   5h19m
pcubuntu@PraneshPC:~/.kube$ minikube service my-service
|-----------|------------|-------------|---------------------------|
| NAMESPACE |    NAME    | TARGET PORT |            URL            |
|-----------|------------|-------------|---------------------------|
| default   | my-service |        9000 | http://192.168.49.2:30002 |
|-----------|------------|-------------|---------------------------|
🏃  Starting tunnel for service my-service.
|-----------|------------|-------------|------------------------|
| NAMESPACE |    NAME    | TARGET PORT |          URL           |
|-----------|------------|-------------|------------------------|
| default   | my-service |             | http://127.0.0.1:34781 |
|-----------|------------|-------------|------------------------|
🎉  Opening service default/my-service in default browser...
👉  http://127.0.0.1:34781
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
^C✋  Stopping tunnel for service my-service.
pcubuntu@PraneshPC:~/.kube$ curl  http://192.168.49.2:30002
<!doctype html><html lang="en"><head><title>HTTP Status 404 – Not Found</title><style type="text/css">body {font-family:Tahoma,Arial,sans-serif;
} h1, h2, h3, b {color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:16px;} h3 {font-size:14px;} p {font-size:12px;} a {co
lor:black;} .line {height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 404 – Not Found</h1><hr class="line" />
<p><b>Type</b> Status Report</p><p><b>Description</b> The origin server did not find a current representation for the target resource or is not
willing to disclose that one exists.</p><hr class="line" /><h3>Apache Tomcat/9.0.102</h3></body></html>pcubuntu@PraneshPC:~/.kube$ ^C
pcubuntu@PraneshPC:~/.kube$ curl  http://192.168.49.2:30002/my-app/
<html>
<body>
<h2>Hello World!</h2>
</body>
</html>
```