Industry grade project 1 – ABC technologies

CI/CD Pipeline for XYZ technologies

Introduction

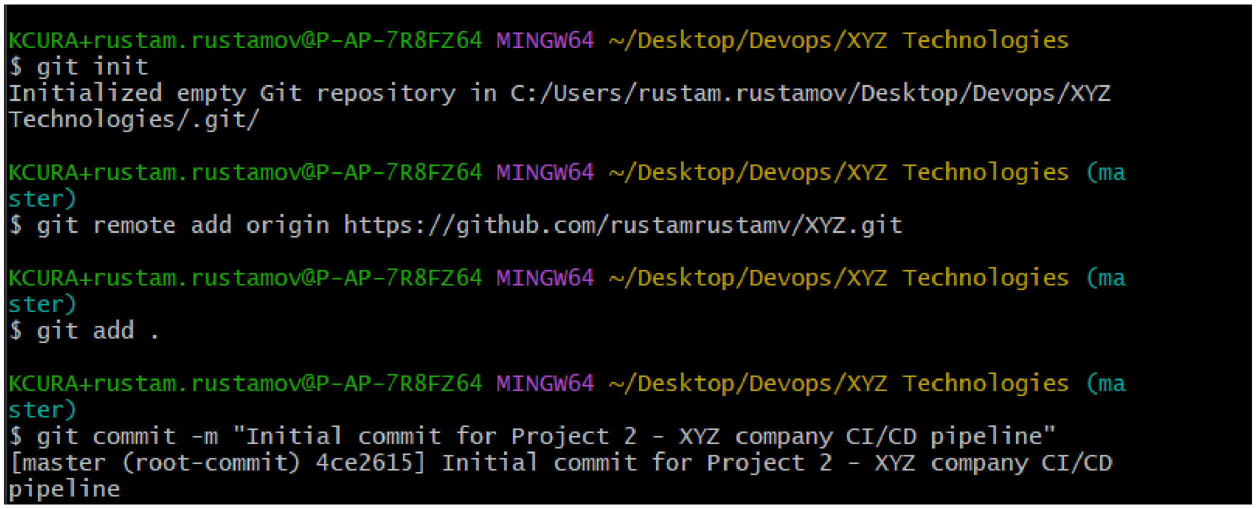
Project Goal

Build a fully automated CI/CD pipeline for the XYZ Technologies webapp—starting from code commit, through build/test, to Docker packaging, Kubernetes deployment, and public access.

1. Source Code Repository

I have started the pulling the resources which has been provided edureka and then pushed it to the my github repository. - https://github.com/rustamrustamv/XYZ.git

```
git init
git add .
git commit -m "Initial commit for Project 2 - XYZ Company CI/CD pipeline"
git push -u origin master
```



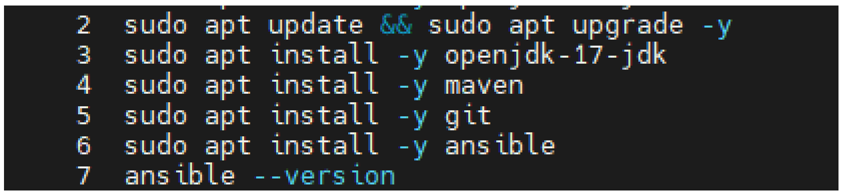
```
KCURA+rustam.rustamov@P-AP-7R8FZ64 MINGW64 ~/Desktop/Devops/XYZ Technologies
$ git init
Initialized empty Git repository in C:/Users/rustam.rustamov/Desktop/Devops/XYZ
Technologies/.git/

KCURA+rustam.rustamov@P-AP-7R8FZ64 MINGW64 ~/Desktop/Devops/XYZ Technologies (ma
ster)
$ git remote add origin https://github.com/rustamrustamv/XYZ.git

KCURA+rustam.rustamov@P-AP-7R8FZ64 MINGW64 ~/Desktop/Devops/XYZ Technologies (ma
ster)
$ git add .

KCURA+rustam.rustamov@P-AP-7R8FZ64 MINGW64 ~/Desktop/Devops/XYZ Technologies (ma
ster)
$ git commit -m "Initial commit for Project 2 - XYZ company CI/CD pipeline"
[master (root-commit) 4ce2615] Initial commit for Project 2 - XYZ company CI/CD
pipeline
```

2. Installation of tools and configuration.

During that stage I have set up an AWS Ubuntu VM for our Project

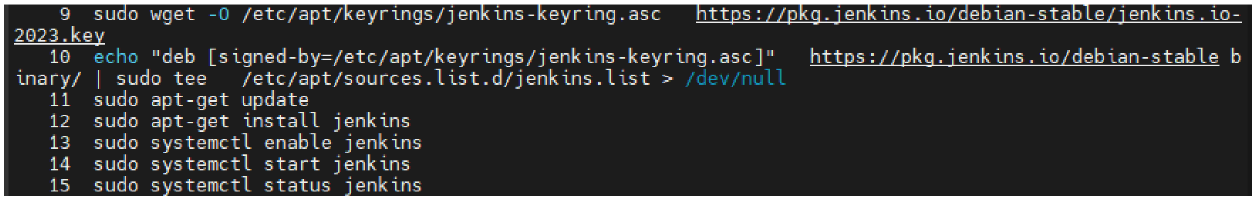
And we run the queries below to install the necessary tools for our project.

Installing Java, Maven, Git, Ansible

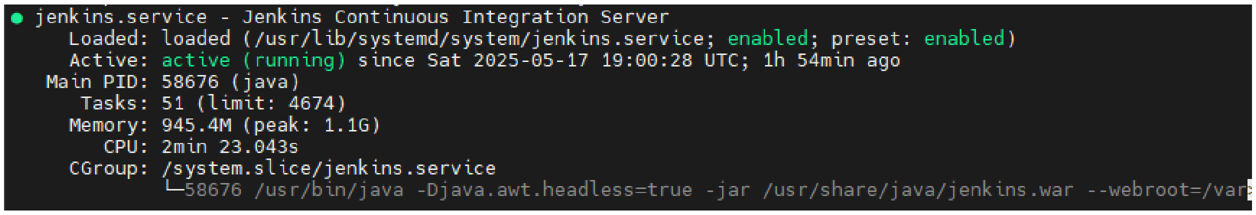


```
sudo apt update && sudo apt upgrade -y
sudo apt install -y openjdk-17-jdk
sudo apt install -y maven
sudo apt install -y git
sudo apt install -y ansible
ansible --version
```

Later we continue with Jenkins installation



```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc   https://pkg.jenkins.io/debian-stable/jenkins.io-
2023.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]"   https://pkg.jenkins.io/debian-stable b
inary/ | sudo tee   /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```



```
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-05-17 19:00:28 UTC; 1h 54min ago
   Main PID: 58676 (java)
      Tasks: 51 (limit: 4674)
     Memory: 945.4M (peak: 1.1G)
        CPU: 2min 23.043s
     CGroup: /system.slice/jenkins.service
             └─58676 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var
```

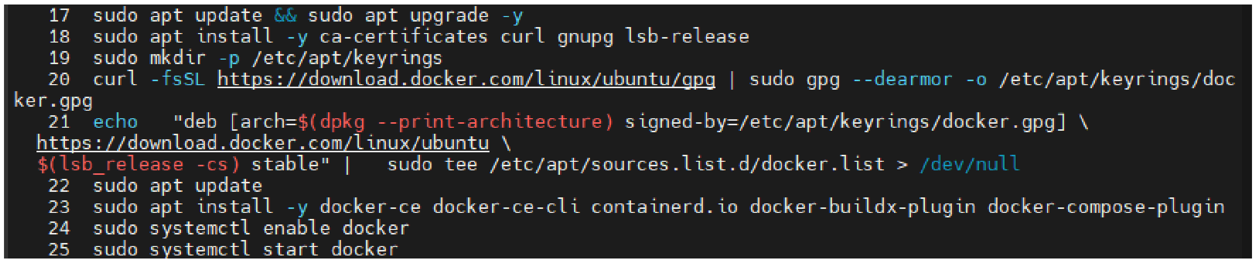
And continue with installing the necessary plugins we will need

Docker Pipeline

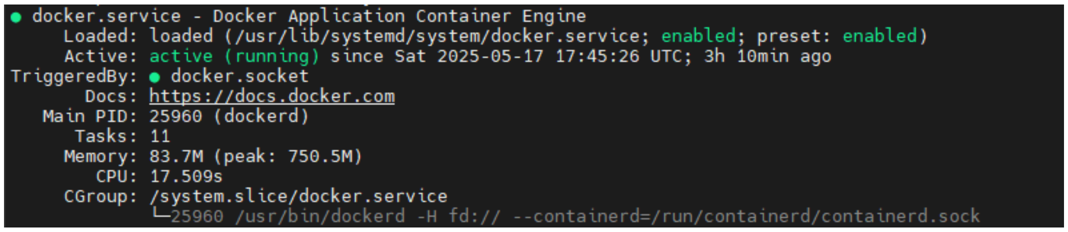
Kubernetes CLI

GitHub Integration

Next step is docker installation

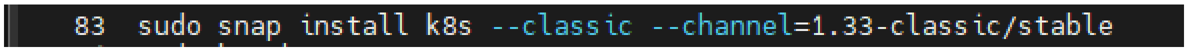


```
   17  sudo apt update && sudo apt upgrade -y
   18  sudo apt install -y ca-certificates curl gnupg lsb-release
   19  sudo mkdir -p /etc/apt/keyrings
   20  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/doc
ker.gpg
   21  echo   "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" |   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
   22  sudo apt update
   23  sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
   24  sudo systemctl enable docker
   25  sudo systemctl start docker
```



```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-05-17 17:45:26 UTC; 3h 10min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 25960 (dockerd)
      Tasks: 11
     Memory: 83.7M (peak: 750.5M)
        CPU: 17.509s
     CGroup: /system.slice/docker.service
             └─25960 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

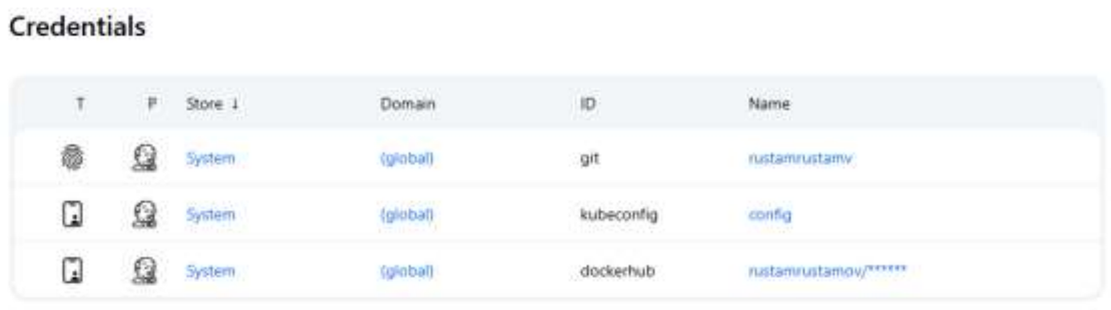
At lastly we installed Kubernetes using snap



```
   83  sudo snap install k8s --classic --channel=1.33-classic/stable
```

```
microk8s enable dns
microk8s enable dashboard
microk8s enable storage
microk8s kubectl get nodes
```

We also added the necessary credentials for ssh, git access, ansible access, Docker, Kubernetes to the Jenkins credentials.

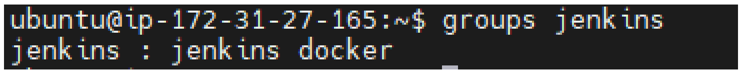


Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	git	rustamrustamv
		System	(global)	kubeconfig	config
		System	(global)	dockerhub	rustamrustamov/******

And we should not forget to add **Jenkins User to Docker Group**:

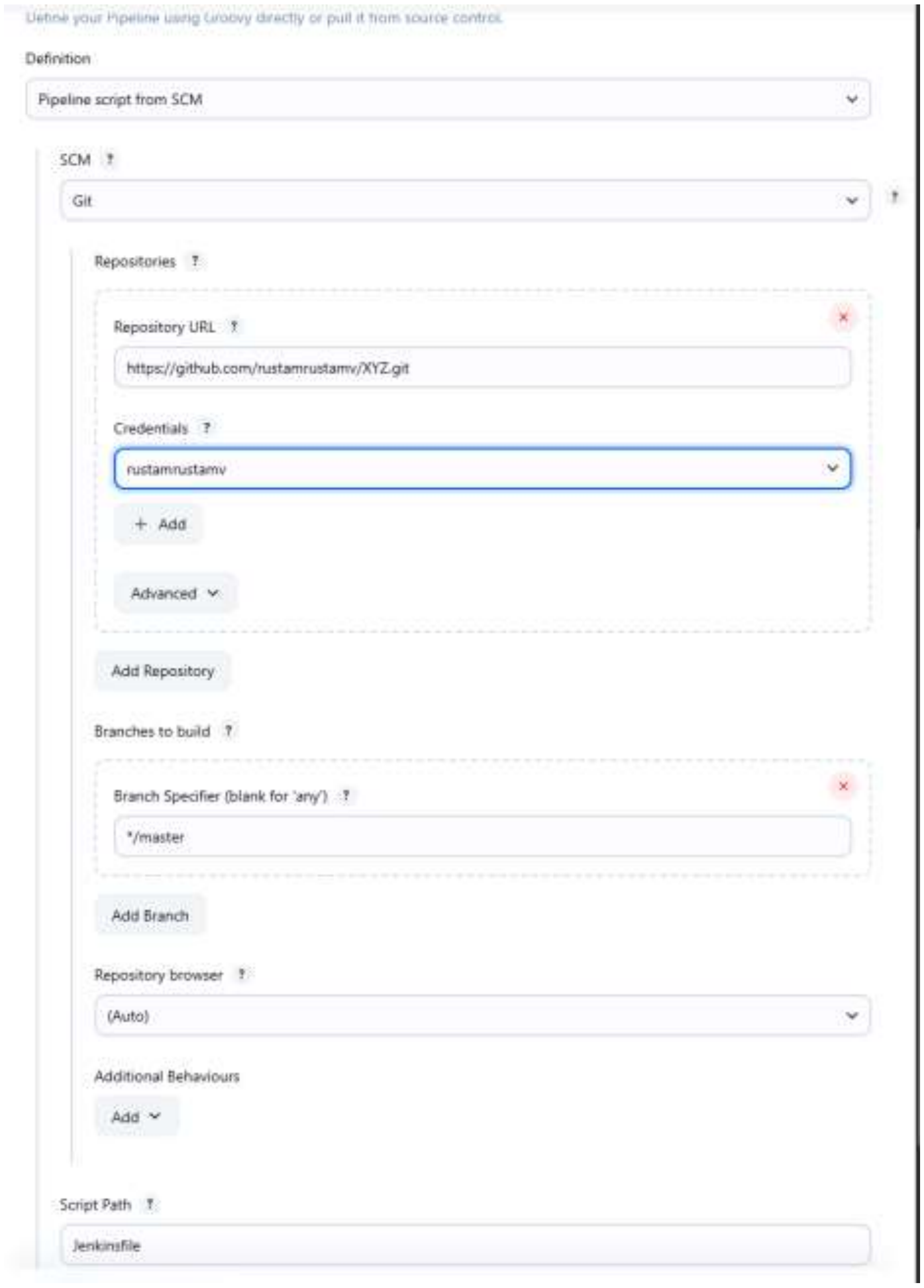
```
sudo usermod -aG docker jenkins
```



```
ubuntu@ip-172-31-27-165:~$ groups jenkins
jenkins : jenkins docker
```

3. Running the pipeline through jenkins

jenkins job configuration



Define your Pipeline using Groovy directly or pull it from source control.

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/rustamrustamv/XYZ.git

Credentials: rustamrustamv

Branches to build

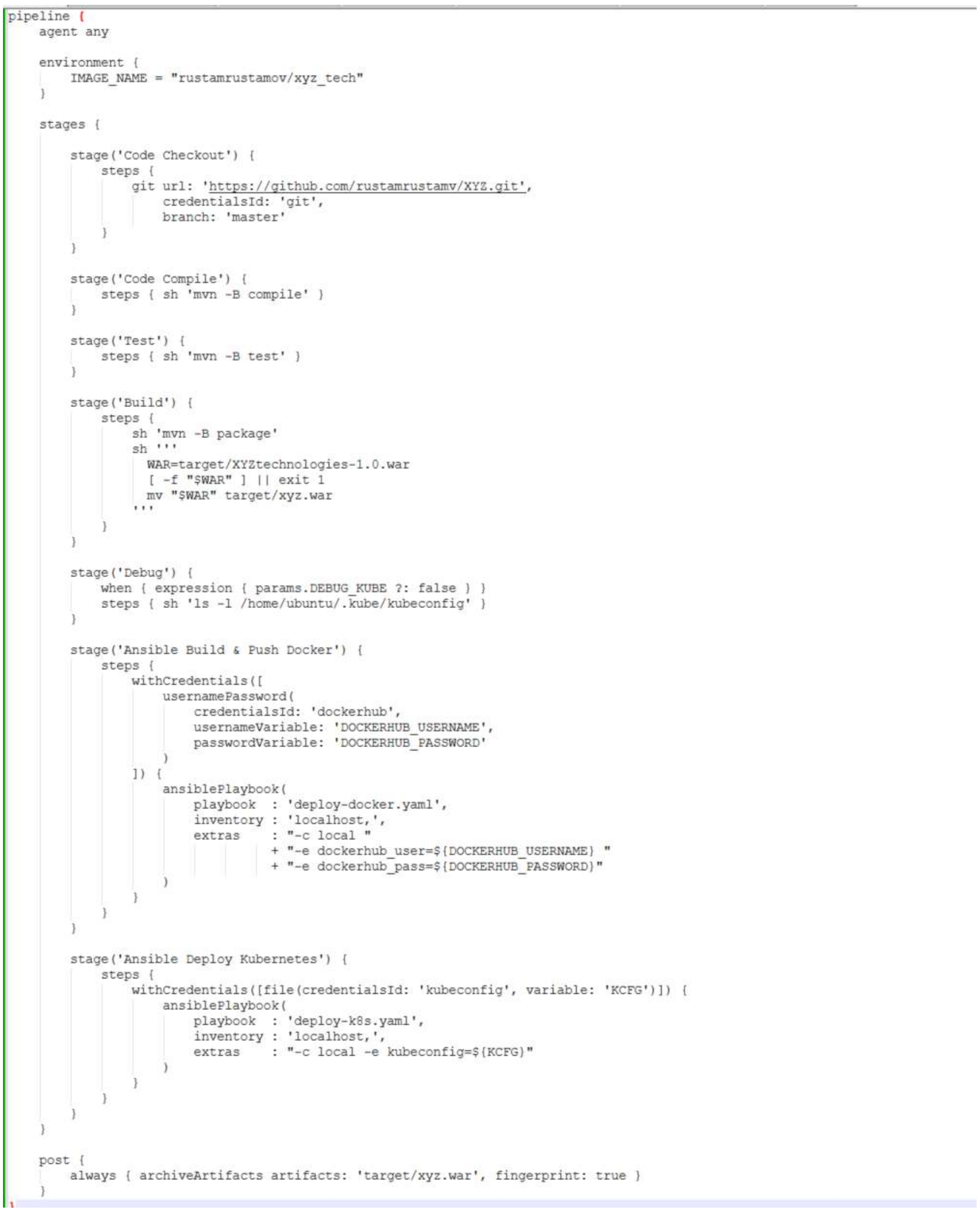
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

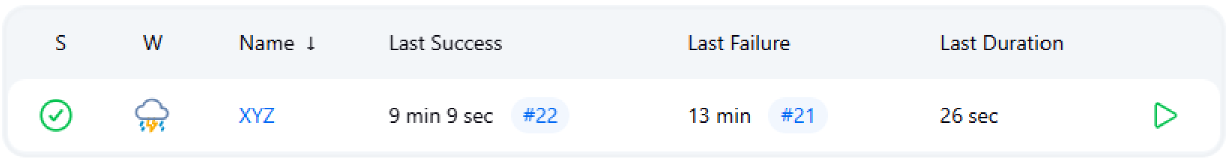
Script Path: Jenkinsfile

Jenkinsfile



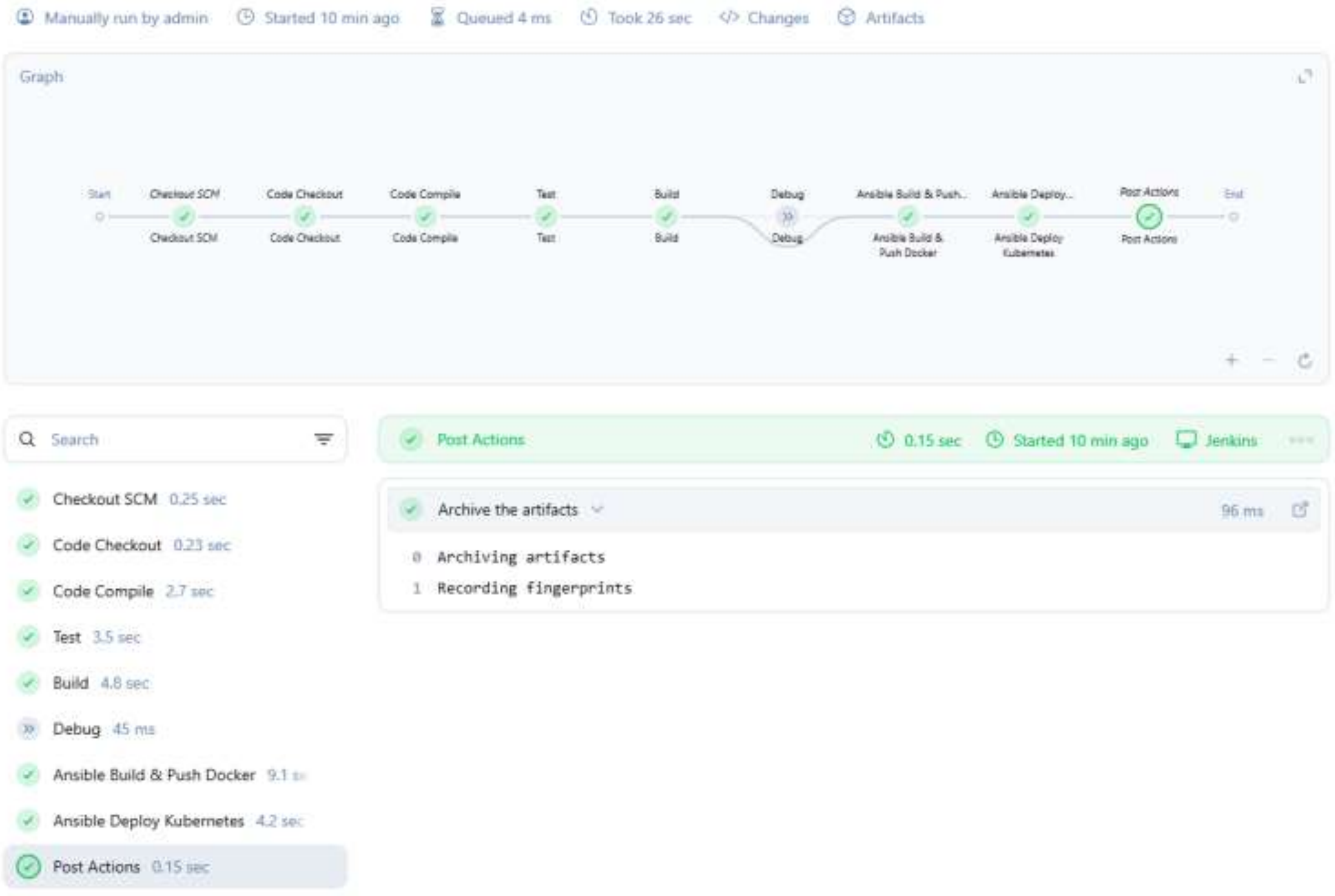
```groovy
pipeline {
    agent any

    environment {
        IMAGE_NAME = "rustamrustamov/xyz_tech"
    }

    stages {

        stage('Code Checkout') {
            steps {
                git url: 'https://github.com/rustamrustamv/XYZ.git',
                    credentialsId: 'git',
                    branch: 'master'
            }
        }

        stage('Code Compile') {
            steps { sh 'mvn -B compile' }
        }

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

        stage('Build') {
            steps {
                sh 'mvn -B package'
                sh '''
                  WAR=target/XYZtechnologies-1.0.war
                  [ -f "$WAR" ] || exit 1
                  mv "$WAR" target/xyz.war
                '''
            }
        }

        stage('Debug') {
            when { expression { params.DEBUG_KUBE ?: false } }
            steps { sh 'ls -l /home/ubuntu/.kube/kubeconfig' }
        }

        stage('Ansible Build & Push Docker') {
            steps {
                withCredentials([
                    usernamePassword(
                        credentialsId: 'dockerhub',
                        usernameVariable: 'DOCKERHUB_USERNAME',
                        passwordVariable: 'DOCKERHUB_PASSWORD'
                    )
                ]) {
                    ansiblePlaybook(
                        playbook  : 'deploy-docker.yaml',
                        inventory : 'localhost,',
                        extras    : "-c local "
                                  + "-e dockerhub_user=${DOCKERHUB_USERNAME} "
                                  + "-e dockerhub_pass=${DOCKERHUB_PASSWORD}"
                    )
                }
            }
        }

        stage('Ansible Deploy Kubernetes') {
            steps {
                withCredentials([file(credentialsId: 'kubeconfig', variable: 'KCFG')]) {
                    ansiblePlaybook(
                        playbook  : 'deploy-k8s.yaml',
                        inventory : 'localhost,',
                        extras    : "-c local -e kubeconfig=${KCFG}"
                    )
                }
            }
        }
    }

    post {
        always { archiveArtifacts artifacts: 'target/xyz.war', fingerprint: true }
    }
}
```

Job Builds



S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓		XYZ	9 min 9 sec #22	13 min #21	26 sec	▷

Pipeline overview



Manually run by admin · Started 10 min ago · Queued 4 ms · Took 26 sec · Changes · Artifacts

✓ Checkout SCM	0.25 sec
✓ Code Checkout	0.23 sec
✓ Code Compile	2.7 sec
✓ Test	3.5 sec
✓ Build	4.8 sec
» Debug	45 ms
✓ Ansible Build & Push Docker	9.1 s
✓ Ansible Deploy Kubernetes	4.2 sec
✓ Post Actions	0.15 sec

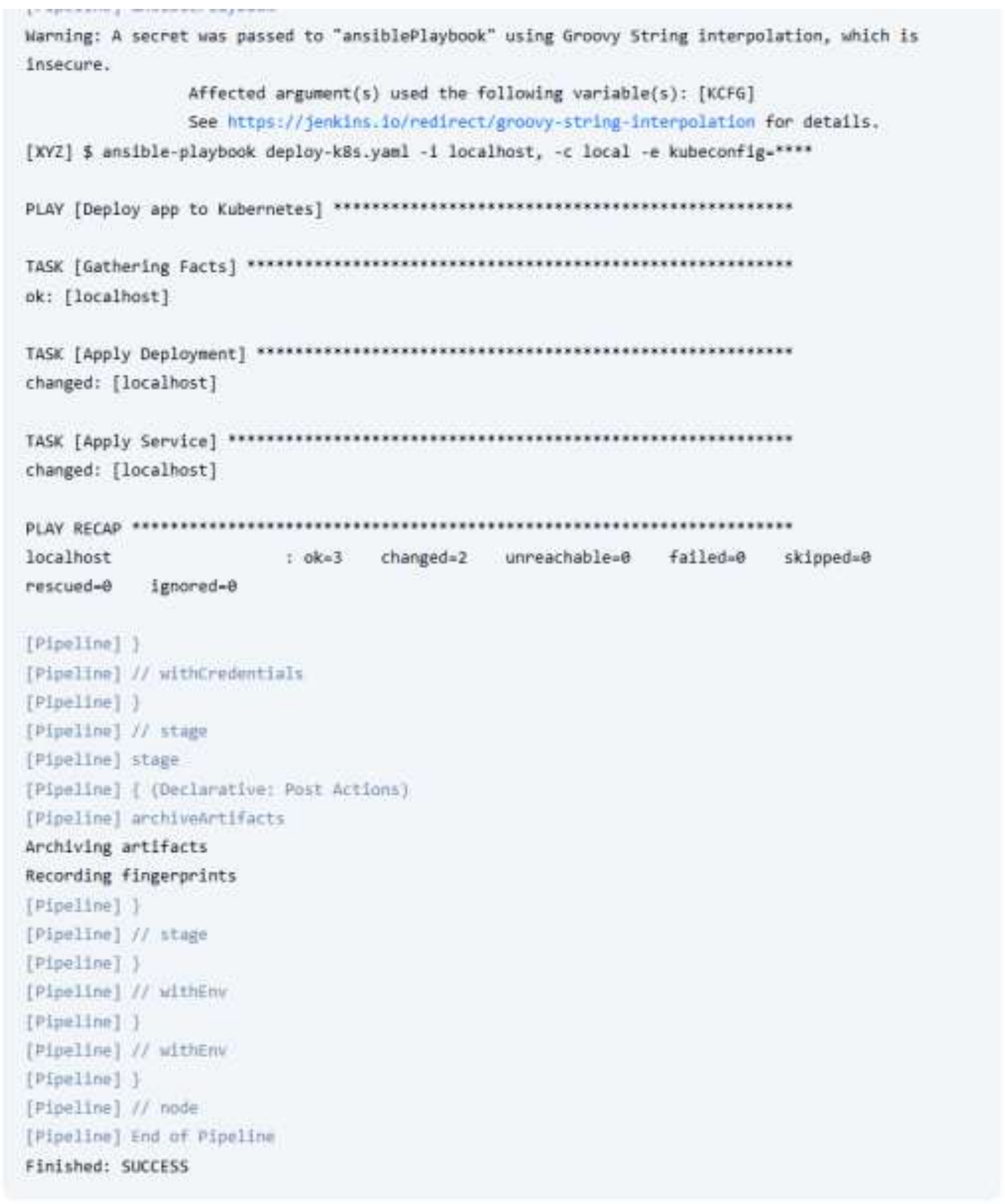
Post Actions · 0.15 sec · Started 10 min ago · Jenkins

Archive the artifacts · 96 ms

```
Archiving artifacts
Recording fingerprints
```

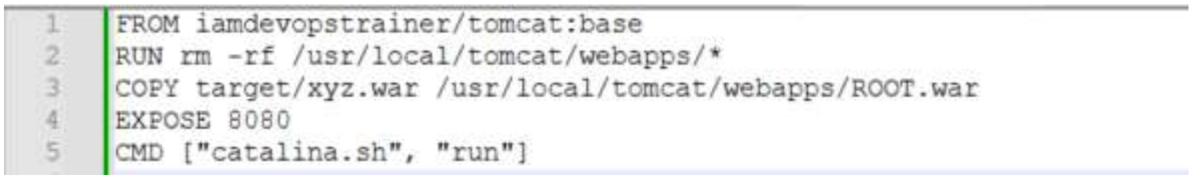
Console output

https://docs.google.com/document/d/1iuSOxI4SsT1sLJ5NMtF6DPxEdw23a5UCZNafxraXB-s/edit?tab=t.0



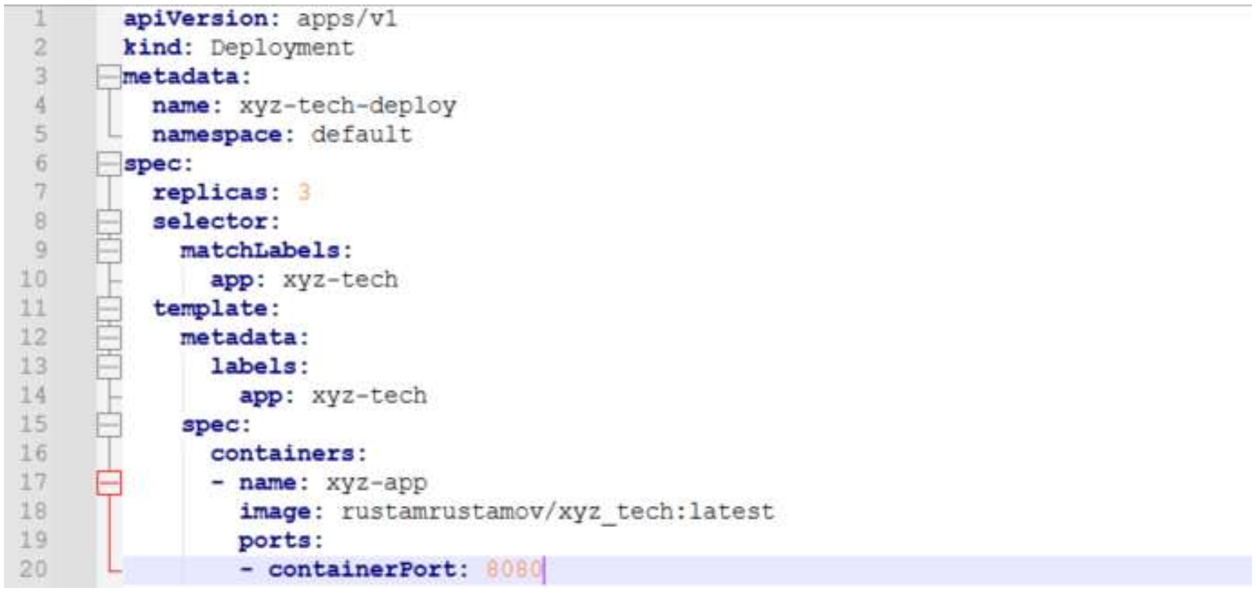
```
Warning: A secret was passed to "ansiblePlaybook" using Groovy String interpolation, which is
insecure.
                 Affected argument(s) used the following variable(s): [KCFG]
                 See https://jenkins.io/redirect/groovy-string-interpolation for details.
[XYZ] $ ansible-playbook deploy-k8s.yaml -i localhost, -c local -e kubeconfig=****

PLAY [Deploy app to Kubernetes] ************************************************

TASK [Gathering Facts] *********************************************************
ok: [localhost]

TASK [Apply Deployment] ********************************************************
changed: [localhost]

TASK [Apply Service] ***********************************************************
changed: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] archiveArtifacts
Archiving artifacts
Recording fingerprints
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Docker file



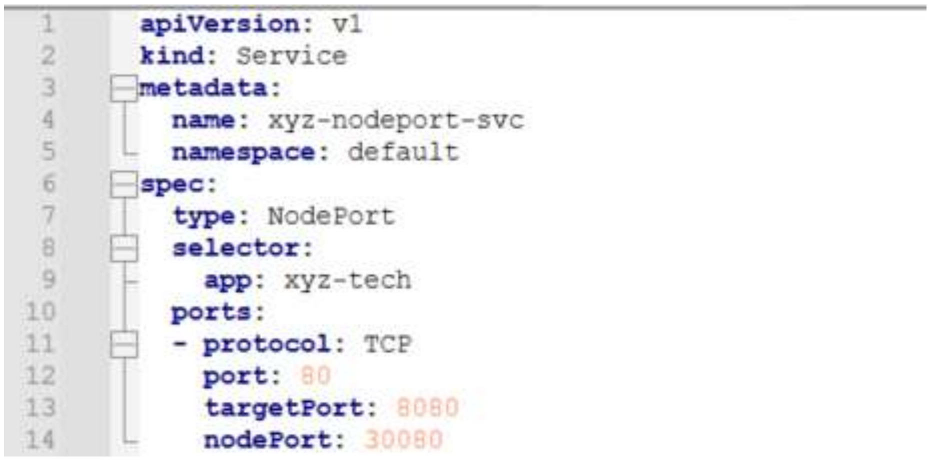
```
FROM iamdevopstrainer/tomcat:base
RUN rm -rf /usr/local/tomcat/webapps/*
COPY target/xyz.war /usr/local/tomcat/webapps/ROOT.war
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

Deployment.yaml (Kubernetes)



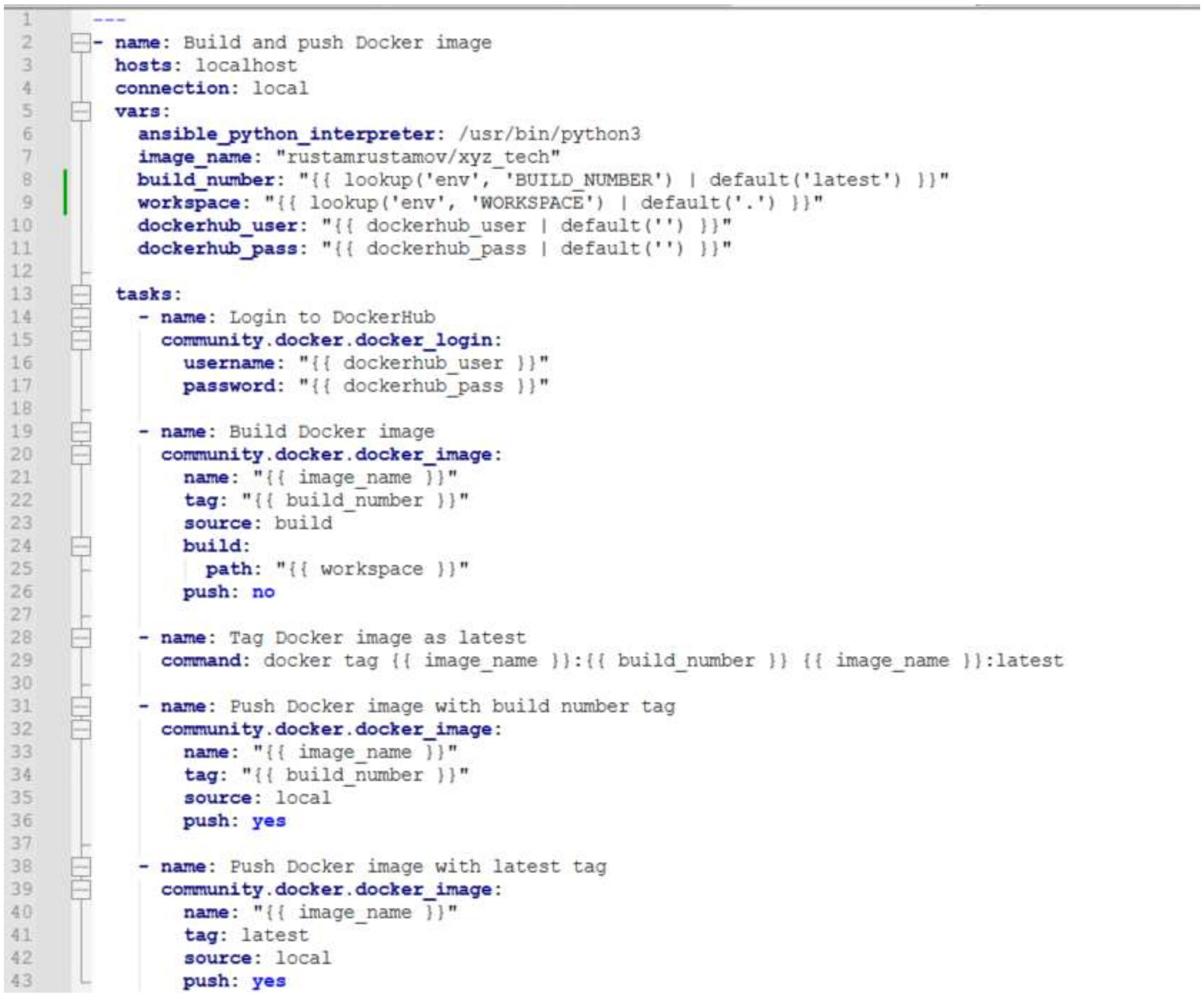
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: xyz-tech-deploy
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: xyz-tech
  template:
    metadata:
      labels:
        app: xyz-tech
    spec:
      containers:
      - name: xyz-app
        image: rustamrustamov/xyz_tech:latest
        ports:
        - containerPort: 8080
```

Service.yaml (Kubernetes)



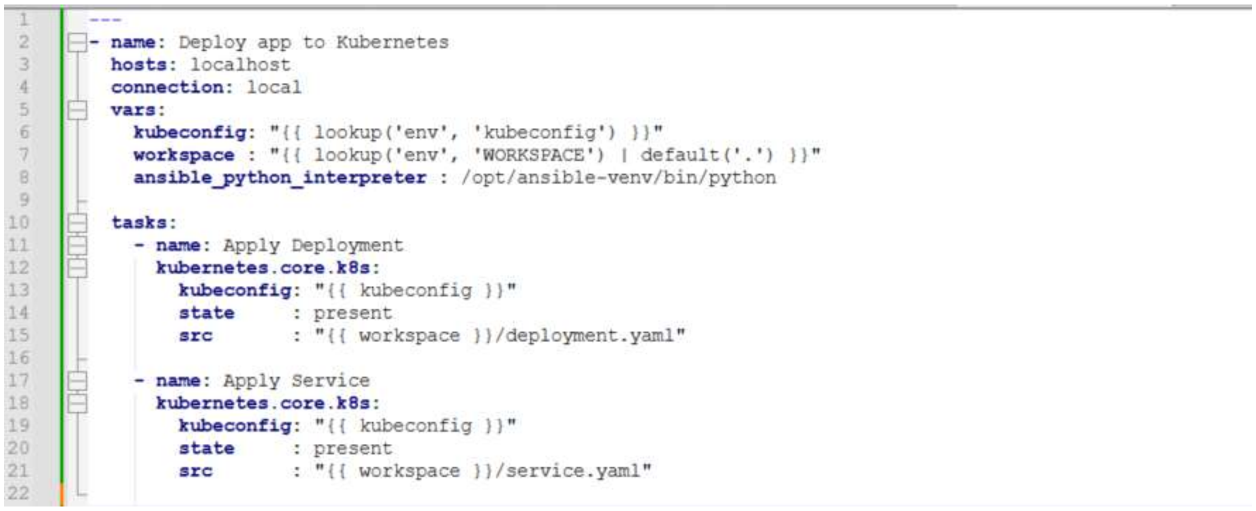
```
apiVersion: v1
kind: Service
metadata:
  name: xyz-nodeport-svc
  namespace: default
spec:
  type: NodePort
  selector:
    app: xyz-tech
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
    nodePort: 30080
```

Deploy-dcoker.yaml (ansible playbook)



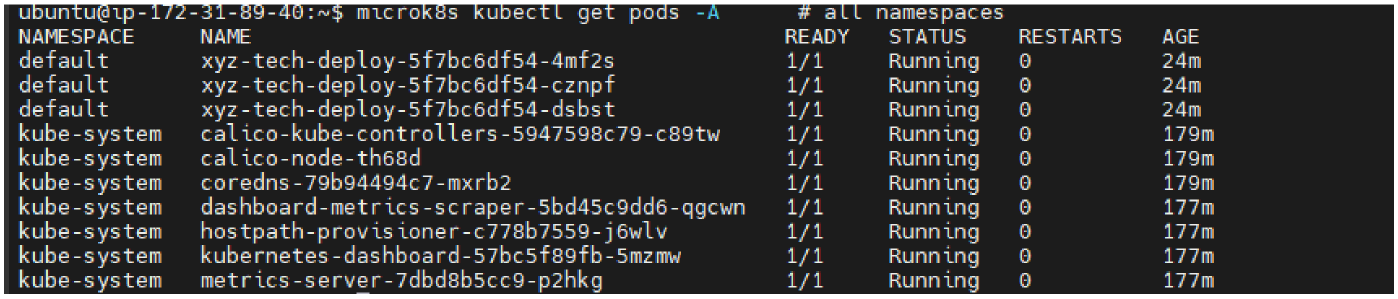
```
---
- name: Build and push Docker image
  hosts: localhost
  connection: local
  vars:
    ansible_python_interpreter: /usr/bin/python3
    image_name: "rustamrustamov/xyz_tech"
    build_number: "{{ lookup('env', 'BUILD_NUMBER') | default('latest') }}"
    workspace: "{{ lookup('env', 'WORKSPACE') | default('.') }}"
    dockerhub_user: "{{ dockerhub_user | default('') }}"
    dockerhub_pass: "{{ dockerhub_pass | default('') }}"

  tasks:
    - name: Login to DockerHub
      community.docker.docker_login:
        username: "{{ dockerhub_user }}"
        password: "{{ dockerhub_pass }}"

    - name: Build Docker image
      community.docker.docker_image:
        name: "{{ image_name }}"
        tag: "{{ build_number }}"
        source: build
        build:
          path: "{{ workspace }}"
        push: no

    - name: Tag Docker image as latest
      command: docker tag {{ image_name }}:{{ build_number }} {{ image_name }}:latest

    - name: Push Docker image with build number tag
      community.docker.docker_image:
        name: "{{ image_name }}"
        tag: "{{ build_number }}"
        source: local
        push: yes

    - name: Push Docker image with latest tag
      community.docker.docker_image:
        name: "{{ image_name }}"
        tag: latest
        source: local
        push: yes
```

Deploy-k8s.yaml (ansible playbook)



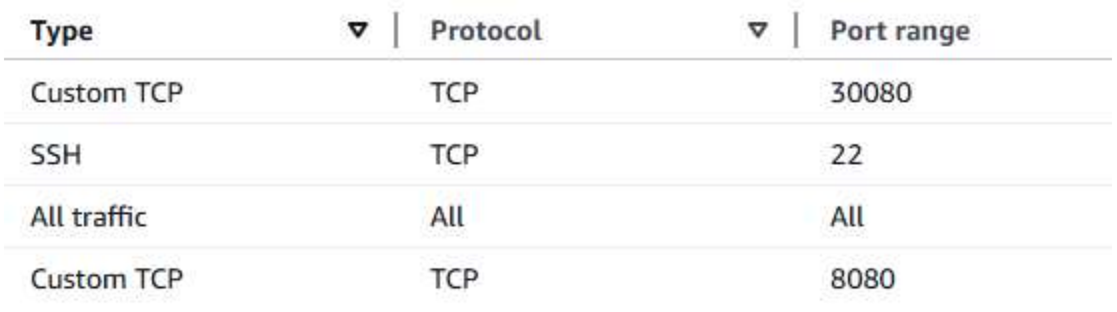
```
---
- name: Deploy app to Kubernetes
  hosts: localhost
  connection: local
  vars:
    kubeconfig: "{{ lookup('env', 'kubeconfig') }}"
    workspace : "{{ lookup('env', 'WORKSPACE') | default('.') }}"
    ansible_python_interpreter : /opt/ansible-venv/bin/python

  tasks:
    - name: Apply Deployment
      kubernetes.core.k8s:
        kubeconfig: "{{ kubeconfig }}"
        state      : present
        src        : "{{ workspace }}/deployment.yaml"

    - name: Apply Service
      kubernetes.core.k8s:
        kubeconfig: "{{ kubeconfig }}"
        state      : present
        src        : "{{ workspace }}/service.yaml"
```

Kubernetes pods and services



```
ubuntu@ip-172-31-89-40:~$ microk8s kubectl get pods -A      # all namespaces
NAMESPACE     NAME                                        READY   STATUS    RESTARTS   AGE
default       xyz-tech-deploy-5f7bc6df54-4mf2s            1/1     Running   0          24m
default       xyz-tech-deploy-5f7bc6df54-cznpf            1/1     Running   0          24m
default       xyz-tech-deploy-5f7bc6df54-dsbst            1/1     Running   0          24m
kube-system   calico-kube-controllers-5947598c79-c89tw    1/1     Running   0          179m
kube-system   calico-node-th68d                           1/1     Running   0          179m
kube-system   coredns-79b94494c7-mxrb2                    1/1     Running   0          179m
kube-system   dashboard-metrics-scraper-5bd45c9dd6-qgcwn  1/1     Running   0          177m
kube-system   hostpath-provisioner-c778b7559-j6wlv        1/1     Running   0          177m
kube-system   kubernetes-dashboard-57bc5f89fb-5mzmw       1/1     Running   0          177m
kube-system   metrics-server-7dbd8b5cc9-p2hkg             1/1     Running   0          177m
```

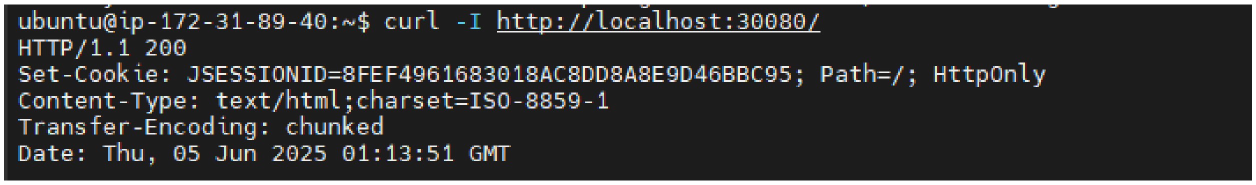
Network setup



Type	Protocol	Port range
Custom TCP	TCP	30080
SSH	TCP	22
All traffic	All	All
Custom TCP	TCP	8080

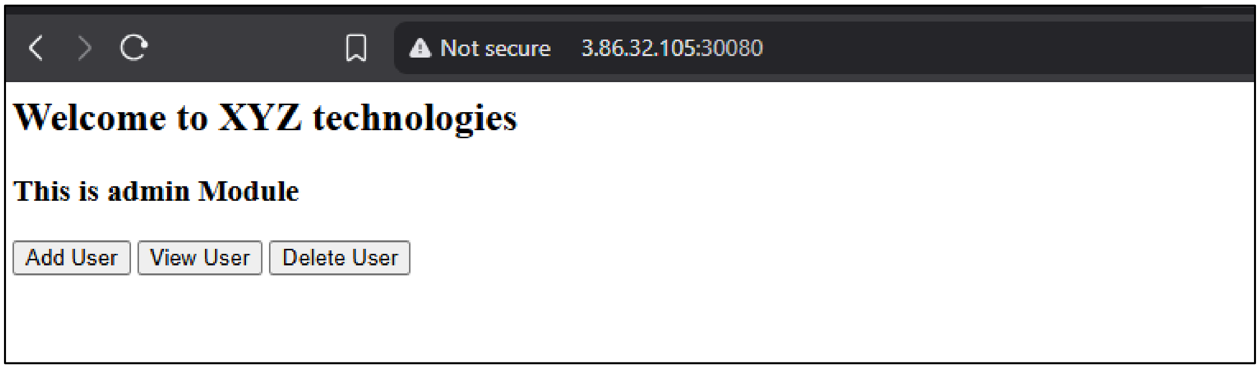
4.Validation

Now we will test if our application is deployed successfully using locally and through external browser.



```
ubuntu@ip-172-31-89-40:~$ curl -I http://localhost:30080/
HTTP/1.1 200
Set-Cookie: JSESSIONID=8FEF4961683018AC8DD8A8E9D46BBC95; Path=/; HttpOnly
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Thu, 05 Jun 2025 01:13:51 GMT
```

Throught the public internet using AWS VM public IP



Welcome to XYZ technologies

This is admin Module

Conclusion

We now have a fully automated pipeline that meets the following criteria:

- **Source control** – watchs github for commits.
- **Jenkins CI** – Maven compiles the code, runs unit tests, and packages a xyz.war artifact on every build.
- **Ansible-driven Docker build** – An Ansible playbook (deploy-docker.yaml) builds a Tomcat-based image, tags it with the build number + latest, and logs in to Docker Hub using Jenkins-managed secrets.
- **Image publication** – The freshly built image is pushed to Docker Hub under rustamrustamov/xyz_tech for easy pull from any environment.
- **Kubernetes deployment** – A second playbook (deploy-k8s.yaml) applies/updates the Kubernetes Deployment and Service objects via the kubernetes.core.k8s module, using the kube-config injected from Jenkins credentials.
- **Highly-available release** – Kubernetes rolls out **3 replicas** with defined CPU / memory requests & limits to guarantee resilience and resource governance.
- **External access** – A NodePort service publishes the application on **port 30080**, so the WAR is reachable at http://<public ip of vm>:30080/.

- **Observability ready** – Prometheus/Node-Exporter targets can be added later to the cluster to monitor pod CPU, memory and network in real time.
- **Fully automated & reproducible** – No manual SSH or click-ops; credentials (Git SSH key, Docker Hub creds, kube-config) live in Jenkins' credential store, and the same pipeline can promote future modules (admin, recommendations, etc.) with zero changes.