

## Jenkins on Kubernetes Assignment

Objective: Deploy Jenkins as a master pod on Kubernetes, configure dynamic agent pods to run pipeline jobs, and verify execution.

### Jenkins Master Deployment

Deployment YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  labels:
    app: jenkins
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      serviceAccountName: jenkins
      containers:
        - name: jenkins
          image: jenkins/jenkins:lts-jdk17
          ports:
            - containerPort: 8080
            - containerPort: 50000
          volumeMounts:
            - name: jenkins-home
              mountPath: /var/jenkins_home
      volumes:
        - name: jenkins-home
          emptyDir: {}
```

### NodePort Service YAML

Exposing port 30000 for web UI and 50000 for agent pods.

```
apiVersion: v1
kind: Service
metadata:
  name: jenkins
  labels:
    app: jenkins
```

```

spec:
  type: NodePort
  selector:
    app: jenkins
  ports:
    - name: http
      port: 8080
      targetPort: 8080
      nodePort: 30000
    - name: jnlp
      port: 50000
      targetPort: 50000

kubectl get pods -o wide
kubectl get svc jenkins

```

## ServiceAccount & RBAC

Created ServiceAccount: jenkins and ClusterRoleBinding: jenkins for permissions.

### **jenkins-rbac.yaml**

```

apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: jenkins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: jenkins
    namespace: default

```

### **jenkins-crb.yaml**

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: jenkins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole

```

```

name: cluster-admin
subjects:
- kind: ServiceAccount
  name: jenkins
  namespace: default
jenkins-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: jenkins
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30000
  selector:
    app: Jenkins

```

### **jenkins-sa.yaml**

```

apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: default

```

```

laborant@dev-machine:~$ git clone https://github.com/KMCandashekar/jenkins-on-k8s.git
Cloning into 'jenkins-on-k8s'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.
laborant@dev-machine:~$ ls
jenkins-on-k8s
laborant@dev-machine:~$ cd jenkins-on-k8s/
laborant@dev-machine:jenkins-on-k8s$ ls
README.md jenkins-deployment.yaml jenkins-rbac.yaml jenkins-service.yaml
laborant@dev-machine:jenkins-on-k8s$ kubectl apply -f jenkins-rbac.yaml
serviceaccount/jenkins created
clusterrolebinding.rbac.authorization.k8s.io/jenkins created
laborant@dev-machine:jenkins-on-k8s$ kubectl apply -f jenkins-deployment.yaml
deployment.apps/jenkins created
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME READY STATUS RESTARTS AGE
jenkins-9dc577d99-lm9wc 0/1 ContainerCreating 0 14s
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME READY STATUS RESTARTS AGE
jenkins-9dc577d99-lm9wc 1/1 Running 0 41s
laborant@dev-machine:jenkins-on-k8s$ kubectl apply -f jenkins-service.yaml
service/jenkins created
laborant@dev-machine:jenkins-on-k8s$ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
jenkins NodePort 10.103.37.239 <none> 8080:30000/TCP 10s

```

Kubernetes Cluster Playground

IDE Explorer dev-machine × cplane-01 × node-01 × node-02 × +

```

jenkins-on-k8s
laborant@dev-machine:~$ cd jenkins-on-k8s/
laborant@dev-machine:jenkins-on-k8s$ ls
README.md  jenkins-deployment.yaml  jenkins-rbac.yaml  jenkins-service.yaml
laborant@dev-machine:jenkins-on-k8s$ kubectl apply -f jenkins-rbac.yaml
serviceaccount/jenkins created
clusterrolebinding.rbac.authorization.k8s.io/jenkins created
laborant@dev-machine:jenkins-on-k8s$ kubectl apply -f jenkins-deployment.yaml
deployment.apps/jenkins created
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-lm9wc  0/1     ContainerCreating   0          14s
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-lm9wc  1/1     Running   0          41s
laborant@dev-machine:jenkins-on-k8s$ kubectl apply -f jenkins-service.yaml
service/jenkins created
laborant@dev-machine:jenkins-on-k8s$ kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
jenkins   NodePort   10.103.37.239  <none>        8080:30000/TCP  10s
kubernetes  ClusterIP   10.96.0.1    <none>        443/TCP      2m51s
laborant@dev-machine:jenkins-on-k8s$ kubectl get po
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-lm9wc  1/1     Running   0          2m10s
laborant@dev-machine:jenkins-on-k8s$ kubectl get po -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP          NODE   NOMINATED-NODE   READINESS   GATES
jenkins-9dc577d99-lm9wc  1/1     Running   0          2m18s  10.244.2.2  node-02  <none>        <none>
laborant@dev-machine:jenkins-on-k8s$ 

```

## Expose HTTP(S) Ports

Access HTTP and HTTPS services running in the playground from your browser or local terminal. Handy for exploring web apps (e.g. Prometheus UI, Kubernetes Dashboard, etc.) or exposing HTTP APIs.

Machine	Port	HTTPS <small>i</small>	<input checked="" type="checkbox"/>	<b>EXPOSE</b>
node-02	30000			

[Advanced...](#)

Only ports opened on the VM's primary network interface can be exposed. If you need to expose a port opened on a different interface (e.g., localhost), first forward it using socat or a similar tool.

You can verify the target service is exposable using the following command from inside the playground:

```
curl http://node-02:30000
```

Port	URL	Host	Path	HTTPS	Access	Actions
node-02:30000	<a href="http://68...fe931.node-ap-alde.iximiuz.com">68...fe931.node-ap-alde.iximiuz.com</a>	-	-	(No)	(private)	

## Configuring Jenkins Kubernetes Cloud

Steps: 1. Manage Jenkins → Configure System → Clouds → Add new cloud → Kubernetes  
2. Fields:  
- Kubernetes URL: `https://kubernetes.default:443`  
- Jenkins URL: `http://jenkins:8080`  
- Jenkins Tunnel: `jenkins:50000`  
- Namespace: `default`  
- Credentials: Kubernetes Service Account - Skip TLS Verify:

The image contains two screenshots of the Jenkins 'New cloud' configuration page.

**Top Screenshot (Kubernetes Configuration):**

- Name:** jenkin-k8s
- Kubernetes URL:** https://kubernetes.default:443
- Use Jenkins Proxy:**
- Kubernetes server certificate key:** (Empty text area)
- Disable https certificate check:**
- Kubernetes Namespace:** (Empty text area)

**Bottom Screenshot (Jenkins Configuration):**

- Direct Connection:**
- Jenkins URL:** http://jenkins:8080
- Jenkins tunnel:** jenkins:50000

## Adding Pod Template for Agents

- Label: jenkins-agent
- Container Name: jnlp
- Image: jenkins/inbound-agent:lts-jdk17
- Command / Args: empty
- Working Directory: /home/jenkins/agent

New pod template - Jenkins

Manage Jenkins / Clouds / jenkins-k8s / New pod template

New pod template settings

Status

Name ? jenkins-agent

Pod Templates

Configure

Delete Cloud

Namespace ?

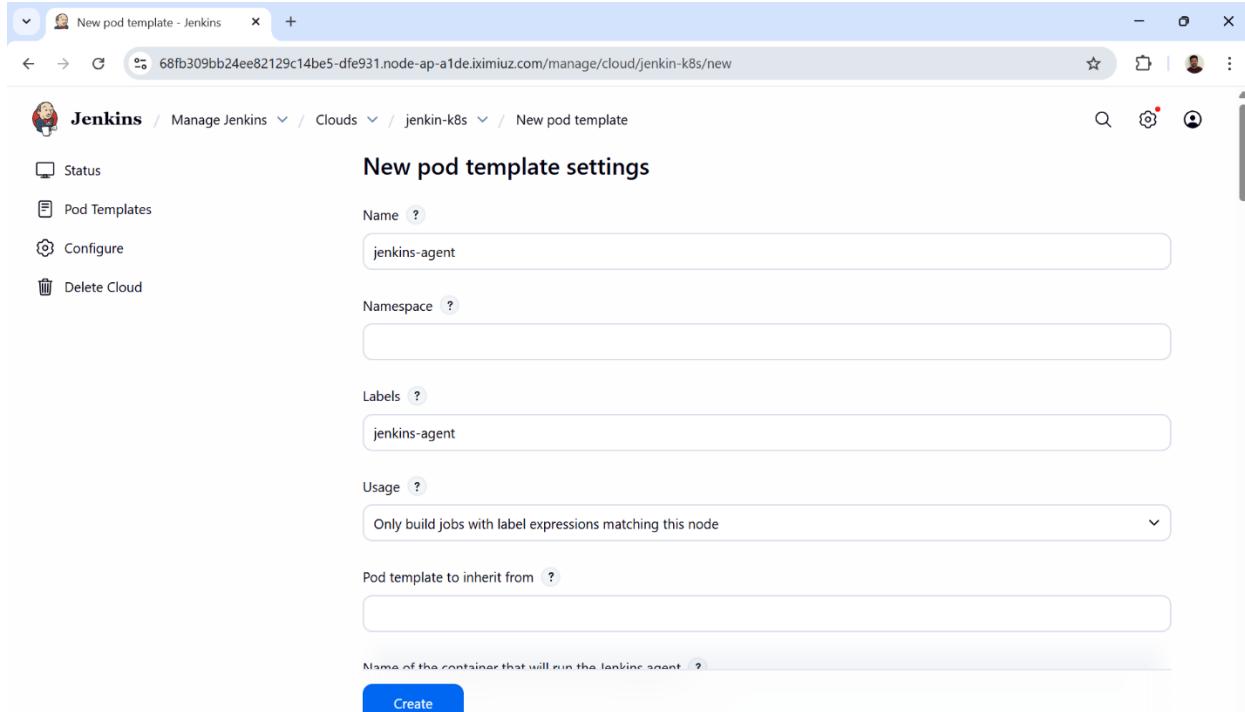
Labels ? jenkins-agent

Usage ? Only build jobs with label expressions matching this node

Pod template to inherit from ?

Name of the container that will run the Jenkins agent ?

Create



Pod template settings - Jenkins

Manage Jenkins / Clouds / jenkins-k8s / jenkins-agent

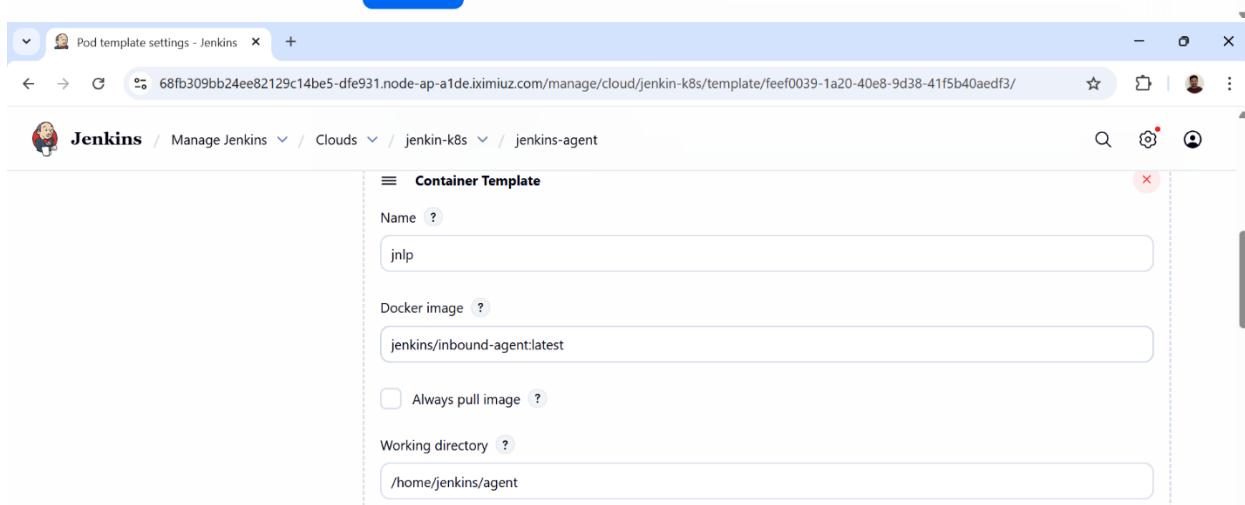
Container Template

Name ? jnlp

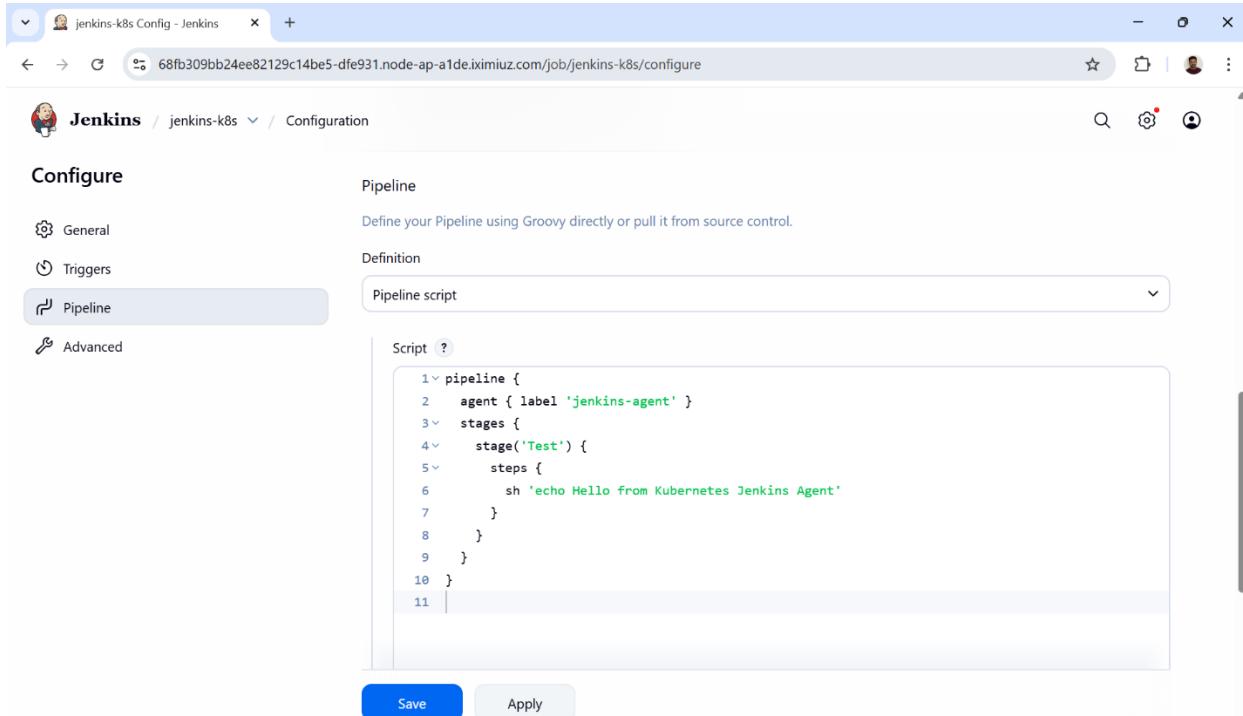
Docker image ? jenkins/inbound-agent:latest

Always pull image ?

Working directory ? /home/jenkins/agent



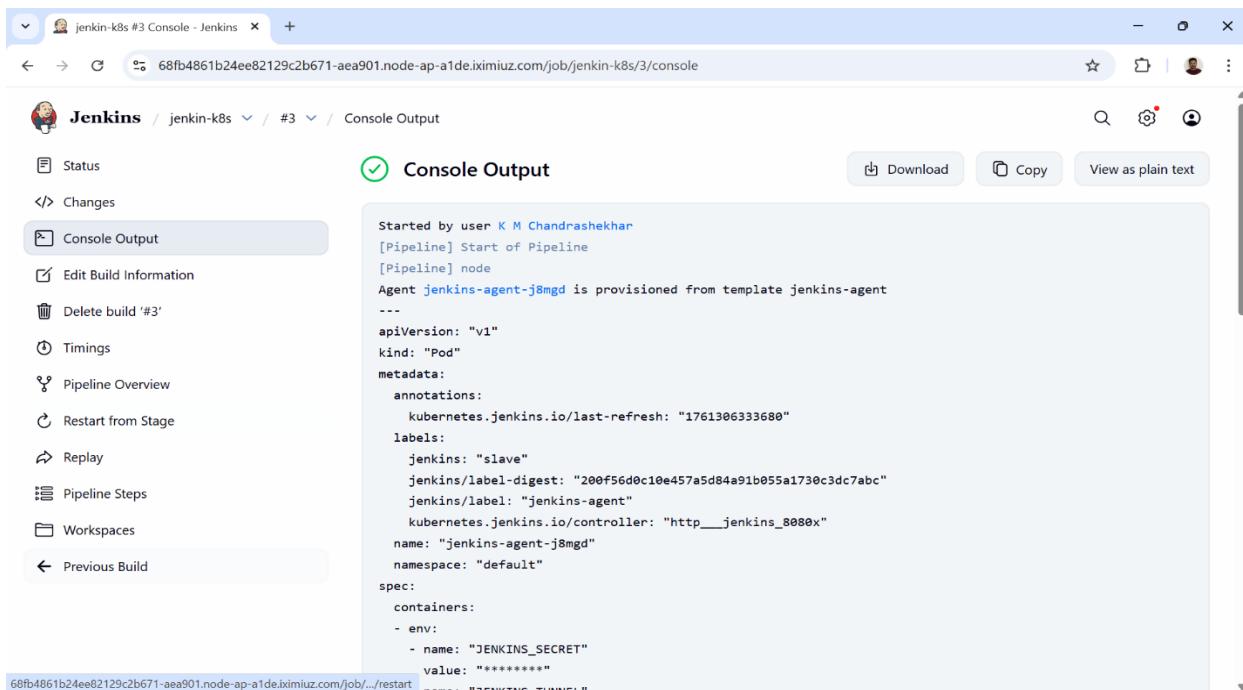
# Pipeline Job Execution



The screenshot shows the Jenkins Pipeline configuration page for a job named "jenkins-k8s". The "Pipeline" tab is selected under the "Definition" section. The pipeline script is defined as follows:

```
1v pipeline {  
2  agent { label 'jenkins-agent' }  
3v  stages {  
4v    stage('Test') {  
5v      steps {  
6        sh 'echo Hello from Kubernetes Jenkins Agent'  
7      }  
8    }  
9  }  
10 }  
11 |
```

At the bottom, there are "Save" and "Apply" buttons.



The screenshot shows the Jenkins Pipeline console output for build #3. The "Console Output" tab is selected. The output shows the start of the pipeline and the provisioning of a Kubernetes pod:

```
Started by user K M Chandrashekhar  
[Pipeline] Start of Pipeline  
[Pipeline] node  
Agent jenkins-agent-j8mgd is provisioned from template jenkins-agent  
---  
apiVersion: "v1"  
kind: "Pod"  
metadata:  
  annotations:  
    kubernetes.jenkins.io/last-refresh: "1761306333680"  
  labels:  
    jenkins: "slave"  
    jenkins/label-digest: "200f56d0c10e457a5d84a91b055a1730c3dc7abc"  
    jenkins/label: "jenkins-agent"  
    kubernetes.jenkins.io/controller: "http_jenkins_8080x"  
  name: "jenkins-agent-j8mgd"  
  namespace: "default"  
spec:  
  containers:  
    - env:  
      - name: "JENKINS_SECRET"  
        value: "*****"  
        valueFrom: "jenkins-tunnel"
```

The screenshot shows a Jenkins job named 'jenkin-k8s' with a build number of 3. The console output window displays the configuration of a Kubernetes pod template and the execution of a pipeline script. The script includes an echo command that prints 'Hello from Jenkins Agent on K8s!'. The pipeline ends with a 'Finished: SUCCESS' message.

```
- mountPath: "/home/jenkins/agent"
  name: "workspace-volume"
  readOnly: false
  workingDir: "/home/jenkins/agent"
hostNetwork: false
nodeSelector:
  kubernetes.io/os: "linux"
restartPolicy: "Never"
volumes:
- emptyDir:
    medium: ""
    name: "workspace-volume"

Running on jenkins-agent-j8mgd in /home/jenkins/agent/workspace/jenkin-k8s
[Pipeline]
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
+ echo Hello from Jenkins Agent on K8s!
Hello from Jenkins Agent on K8s!
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

The screenshot shows a terminal session within the Kubernetes Cluster Playground. The user runs several 'kubectl get pods' commands to check the status of Jenkins-related pods. The output shows three pods: 'jenkins-9dc577d99-vhqsv', 'jenkins-agent-j8mgd', and 'jenkins-9dc577d99-vhqsv'. The first two are running, while the third is in the process of being created.

```
IDE Explorer cplane-01 node-01 node-02 +
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-vhqsv  1/1    Running   0          13m
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-vhqsv  1/1    Running   0          14m
jenkins-agent-j8mgd     0/1    ContainerCreating   0          0s
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-vhqsv  1/1    Running   0          14m
jenkins-agent-j8mgd     0/1    Completed  0          4s
laborant@dev-machine:jenkins-on-k8s$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
jenkins-9dc577d99-vhqsv  1/1    Running   0          14m
laborant@dev-machine:jenkins-on-k8s$
```

- Jenkins master running on Kubernetes
- Agent pods spin up dynamically to execute jobs
- Pipeline successfully executed using an agent pod
- NodePort service allowed web UI access