**Jenkins running workers/slaves in k8s:**

**Below are the high level steps:**

1. Create role for EKS cluster creation
2. Create 2 EC2 instances

* For EKS cluster setup
* For Jenkins setup to be run using dokcer-compose

1. Create EKS cluster
2. Install docker,docker-compose and run the Jenkins master on it.
3. Install plugin Kubernetes and add Kubernetes cluster to Jenkins
4. Create the required service account in k8s and generate the toke which is required for kubeconfig creation
5. Complete the k8s and POD configuration
6. Create pipeline to use the podTemplate and run the job

# 1. Create IAM Role with Admin policy for eks-cluster and attach to ec2-instance

Install eksctl

==============

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/bin

eksctl version

**Install Kubectl**

===============

curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/amd64/kubectl

chmod +x ./kubectl

mkdir -p $HOME/bin

cp ./kubectl $HOME/bin/kubectl

export PATH=$HOME/bin:$PATH

echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

source $HOME/.bashrc

kubectl version --short –client

Cluster Creation

====================

eksctl create cluster --name=eksdemo \

--region=us-east-1 \

--zones=us-east-1a,us-east-1b \

--without-nodegroup

eksctl utils associate-iam-oidc-provider \

--region us-east-1 \

--cluster eksdemo \

--approve

**Create node-group/worker node**

====================================

eksctl create nodegroup --cluster=eksdemo \

--region=us-east-1 \

--name=eksdemo-ng-public \

--node-type=t2.micro \

--nodes=2 \

--nodes-min=2 \

--nodes-max=3 \

--node-volume-size=10 \

--ssh-access \

--ssh-public-key=eks \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access

kubectl get nodes # if we see our 3 nodes, we know we have authenticated correctly

[root@ip-172-31-87-198 ~]# kubectl get nodes

# 2. On the other EC2 instance install docker/docker-compose and run Jenkins image

Install Jenkins on docker using docker compose:

=======================================

Create the EC2 instance of minimum size t2.medium

$ sudo hostnamectl set-hostname Jenkins

$ sudo -i

[root@jenkins ~]# yum install java-1.8.0-openjdk-devel -y

# yum install docker -y

# service docker start

Redirecting to /bin/systemctl start docker.service

Download the docker compose:

# curl -L "https://github.com/docker/compose/releases/download/1.28.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# sudo chmod +x /usr/local/bin/docker-compose

# sudo ln -s /usr/local/bin/docker-compose //usr/local/sbin/docker-compose

# docker-compose --version

docker-compose version 1.28.4, build cabd5cfb

# chmod 666 /var/run/docker.sock

[root@jenkins ~]# mkdir jenkins\_home

[root@jenkins ~]# cd jenkins\_home/

[root@jenkins jenkins\_home]# pwd

/root/jenkins\_home

[root@jenkins jenkins\_home]# cat jenkins-compose.yaml

version: '3.5'

networks:

jenkins:

name: jenkins

services:

jenkins:

user: root

image: jenkins/jenkins:2.249.3

networks:

- jenkins

ports:

- '8080:8080'

- '50000:50000'

volumes:

- /root/jenkins\_home/:/var/lib/jenkins

- /var/run/docker.sock:/var/run/docker.sock

- /usr/bin/docker:/usr/bin/docker

environment:

- JENKINS\_HOME=/var/lib/jenkins

- JAVA\_OPTS=-Duser.timezone=Europe/Kiev

logging:

driver: "journald"

Run the instance :

# docker-compose -f jenkins-compose.yaml up

# docker-compose -f jenkins-compose.yaml up --d

Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/

Creating network "jenkins" with the default driver

Pulling jenkins (jenkins/jenkins:2.249.3)...

2.249.3: Pulling from jenkins/jenkins

3192219afd04: Pull complete

17c160265e75: Pull complete

cc4fe40d0e61: Pull complete

9d647f502a07: Pull complete

d108b8c498aa: Pull complete

1bfe918b8aa5: Pull complete

dafa1a7c0751: Pull complete

951c6d241d2e: Pull complete

31de6d9c9dda: Pull complete

aa795e9dd171: Pull complete

1f9217a855bf: Pull complete

680e8ed9e1f6: Pull complete

4f468d4218e8: Pull complete

c7832382c80b: Pull complete

10df404c3f3b: Pull complete

d395f3d36399: Pull complete

20e117b65303: Pull complete

ed0bbe9c461c: Pull complete

3f98bfdadcfc: Pull complete

4b88e29a38b3: Pull complete

Digest: sha256:980d55fd29a287d2d085c08c2bb6c629395ab2e3dd7547641035b4f126acc322

Status: Downloaded newer image for jenkins/jenkins:2.249.3

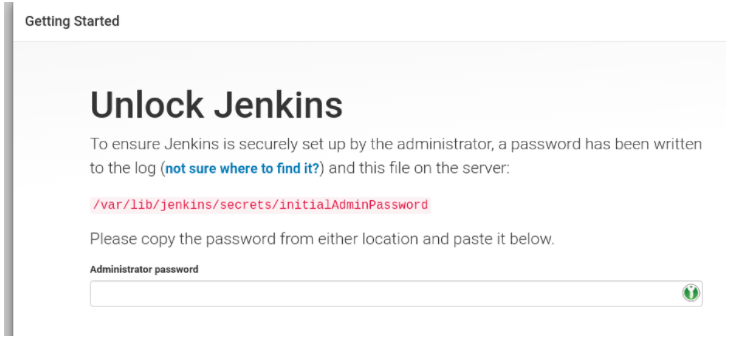
Creating jenkins\_home\_jenkins\_1 ... done

[root@jenkins jenkins\_home]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

87eeeeffb17a jenkins/jenkins:2.249.3 "/sbin/tini -- /usr/…" 19 seconds ago Up 14 seconds 0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp jenkins\_home\_jenkins\_1

Now Jenkins UI is accessible using http://host\_ip:8080



To get the initial admin password from Jenkins docker container:

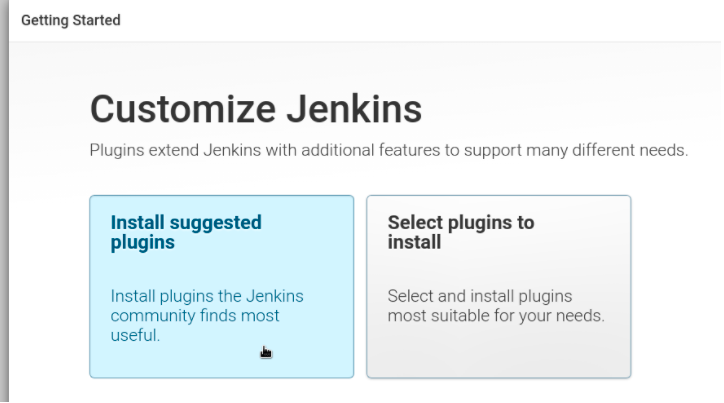
[root@jenkins jenkins\_home]# docker exec -it 87eeeeffb17a bash

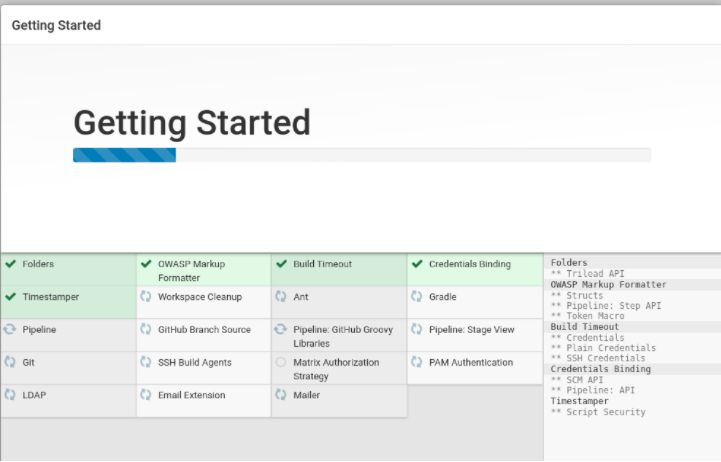
root@87eeeeffb17a:/# cat /var/lib/jenkins/secrets/initialAdminPassword

d98ae4b262434887a0ad46904ea32611

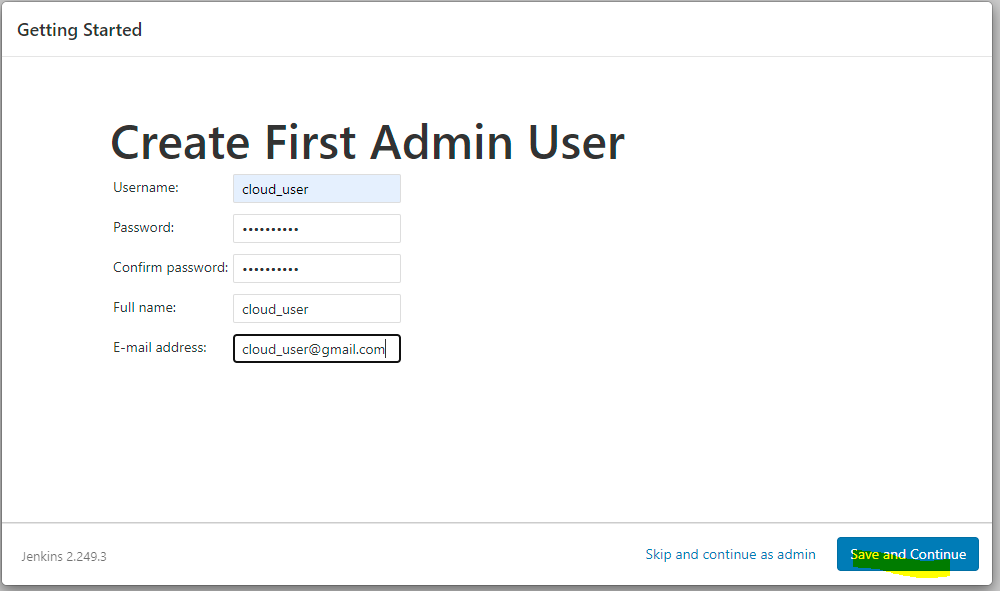
root@87eeeeffb17a:/# exit

exit



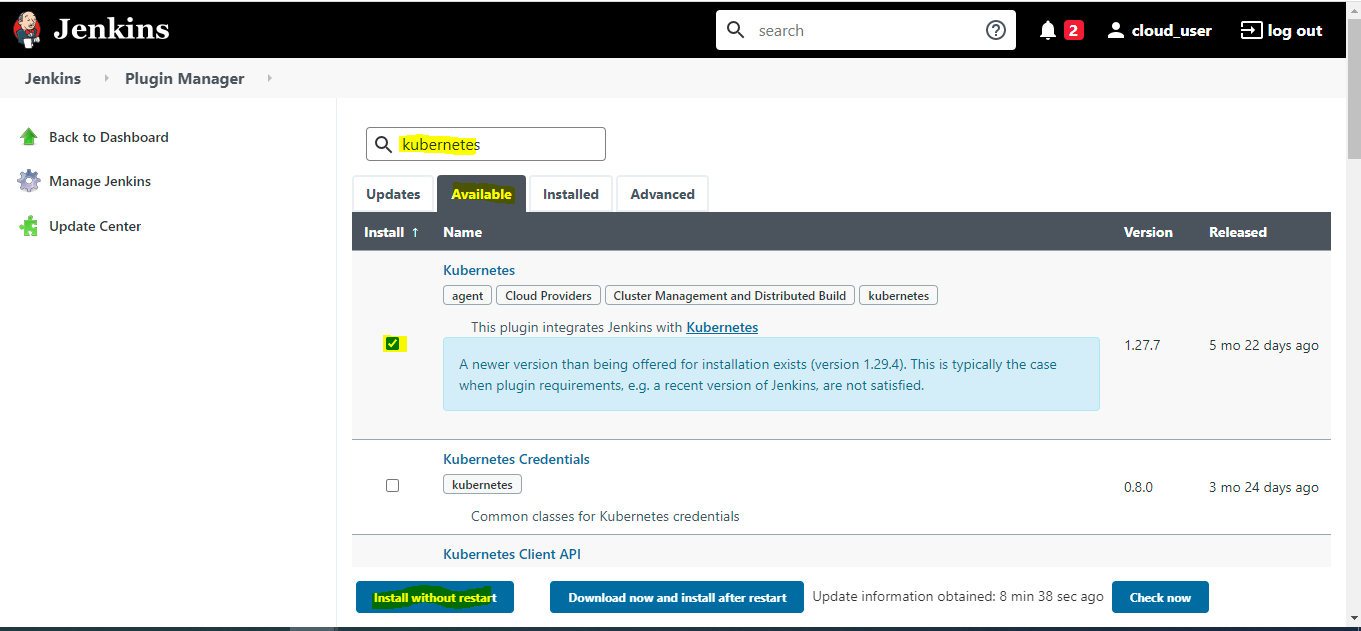


Create a user:

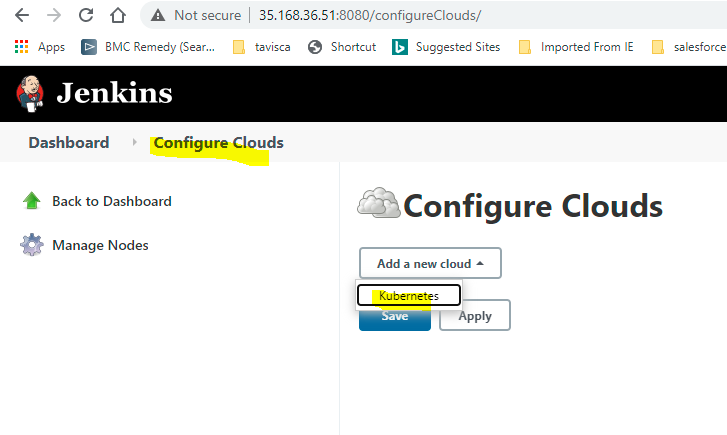


# 3. Install Kubernetes plugin

Goto manage Jenkins 🡪 manage plugins 🡪 available and install the Kubernetes plugin.



Install it and go to the Manage Nodes and Clouds > Configure Clouds:



To get the details of EKS Cluster ARN and other useful info from cli. You can get this info from AWS console as well.

Note down the below output, it will required for k8s configuration.

## # aws eks describe-cluster --name eksdemo --region us-east-1

[root@ip-172-31-23-116 opt]# aws eks describe-cluster --name eksdemo --region us-east-1

{

"cluster": {

"status": "ACTIVE",

"endpoint": "https://64044AB4BBE273C34C3CA81F379E3E64.gr7.us-east-1.eks.amazonaws.com",

"logging": {

"clusterLogging": [

{

"enabled": false,

"types": [

"api",

"audit",

"authenticator",

"controllerManager",

"scheduler"

]

}

]

},

"name": "eksdemo",

"tags": {},

"certificateAuthority": {

"data": ""

},

"roleArn": "arn:aws:iam::412675759620:role/eksctl-eksdemo-cluster-ServiceRole-1GWN09H6FT1D3",

"kubernetesNetworkConfig": {

"serviceIpv4Cidr": "10.100.0.0/16"

},

"resourcesVpcConfig": {

"vpcId": "vpc-00552bb1dac6e76ce",

"subnetIds": [

"subnet-0cb09ee980ad955d8",

"subnet-0c385043e89b58b3b",

"subnet-05c34b565c95a18e9",

"subnet-08179df70e47bb99a"

],

"securityGroupIds": [

"sg-00bedab254c058f60"

],

"clusterSecurityGroupId": "sg-015d6bab478fe7a47",

"publicAccessCidrs": [

"0.0.0.0/0"

],

"endpointPublicAccess": true,

"endpointPrivateAccess": false

},

"platformVersion": "eks.4",

"version": "1.19",

"arn": "arn:aws:eks:us-east-1:412675759620:cluster/eksdemo",

"identity": {

"oidc": {

"issuer": "https://oidc.eks.us-east-1.amazonaws.com/id/64044AB4BBE273C34C3CA81F379E3E64"

}

},

"createdAt": 1621062520.176

}

}

## Jenkins ServiceAccount

In the cluster, create a Namespace and ServiceAccount which will be used by Jenkins for authorization.For now, add a [Kubernetes RoleBinding](https://rtfm.co.ua/en/kubernetes-part-5-rbac-authorization-with-a-role-and-rolebinding-example/) mapped to the [default admin role](https://kubernetes.io/docs/reference/access-authn-authz/rbac/#user-facing-roles) (or create your own – here the admin used just for simplicity as this is a PoC setup) in our dev-1-18-devops-jenkins-slaves-ns namespace:

[root@eks opt]# cat jenkins-slaves-sa.yaml

---

apiVersion: v1

kind: Namespace

metadata:

name: dev-1-18-devops-jenkins-slaves-ns

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: jenkins-slaves-service-account

namespace: dev-1-18-devops-jenkins-slaves-ns

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: jenkins-slaves-rolebinding

namespace: dev-1-18-devops-jenkins-slaves-ns

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: admin

subjects:

- kind: ServiceAccount

name: jenkins-slaves-service-account

namespace: dev-1-18-devops-jenkins-slaves-ns

**Deploy it:**

[root@eks opt]# kubectl apply -f jenkins-slaves-sa.yaml

namespace/dev-1-18-devops-jenkins-slaves-ns unchanged

serviceaccount/jenkins-slaves-service-account unchanged

rolebinding.rbac.authorization.k8s.io/jenkins-slaves-rolebinding created

## Generate kubeconfig using service account for Jenkins

The next thing to do is to generate a [kubeconfig](https://rtfm.co.ua/kubernetes-kubectl-i-kubeconfig-obzor-fajla-dobavlenie-klastera-polzovatelya-i-konteksta/" \t "_blank), which will use this ServiceAccount.

## # kubectl get ns

NAME STATUS AGE

default Active 40m

dev-1-18-devops-jenkins-slaves-ns Active 2m5s

kube-node-lease Active 40m

kube-public Active 40m

kube-system Active 40m

## # kubectl get sa -n dev-1-18-devops-jenkins-slaves-ns

NAME SECRETS AGE

default 1 2m18s

jenkins-slaves-service-account 1 2m18s

## # kubectl -n dev-1-18-devops-jenkins-slaves-ns get sa jenkins-slaves-service-account -o jsonpath='{.secrets[0].name}'

jenkins-slaves-service-account-token-vklkd

**Get the token from the secret, decrypt it with base64:**

# kubectl -n dev-1-18-devops-jenkins-slaves-ns get secret release-with-above-service-account-token -o jsonpath='{.data.token}' | base64 --decode

eyJhbGciOiJSUzI1NiIsImtpZCI6IjV1cVV6N3N1Z21GdGVrbEI2SXlDMHFhWXgyMEdWelpIRjYyclZOc1JSTjAifQ..hdvfYxPTAufTmIJ\_AKEzjRWb4r1uEZMMTRANEmIKbGo83DF4Z-OJINU6wgHxme3ntEyN27ST5\_5\_M7nDSS7Tnq0RxEcVcEleLo6VJqiyiTE7yIG7yvUvtCr2QCt8BiumU88HnwjXLPLdpeF-ZpI8vBoR8EtFH6Ok\_2XrgRgOGOeC9dGZWzNidPSppOCDGcawxWVnGKMGyMU-4jgHT-NP3uLWZy396OLQi6v1oMTQySdsqpPYaDu0-3hImIv3XkUhvpyVcr-455pq-IvRWRQp\_4wsnMAB9itxGDDKYzX\_rWSd4uwmBdaKTqwPo7nCqX-xD\_QnEEozWwVW15SRkZOaxA

Write a [kubeconfig](https://rtfm.co.ua/kubernetes-kubectl-i-kubeconfig-obzor-fajla-dobavlenie-klastera-polzovatelya-i-konteksta/" \t "_blank), for example as a jenkins-dev-1-18-kubeconfig.yaml file:

## # cat jenkins-dev-1-18-kubeconfig.yaml

apiVersion: v1

clusters:

- cluster:

**certificate-authority-data:** 

server: https://64044AB4BBE273C34C3CA81F379E3E64.gr7.us-east-1.eks.amazonaws.com

name: arn:aws:eks:us-east-1:412675759620:cluster/eksdemo

contexts:

- context:

cluster: arn:aws:eks:us-east-1:412675759620:cluster/eksdemo

user: jenkins-slaves-service-account

namespace: dev-1-18-devops-jenkins-slaves-ns

name: jenkins-slaves-service-account@bttrm-dev-1-18

current-context: jenkins-slaves-service-account@bttrm-dev-1-18

kind: Config

users:

- name: jenkins-slaves-service-account

user:

token: eyJhbGciOiJSUzI1NiIsImtpZCI6IjV1cVV6N3N1Z21GdGVrbEI2SXlDMHFhWXgyMEdWelpIRjYyclZOc1JSTjAifQ..hdvfYxPTAufTmIJ\_AKEzjRWb4r1uEZMMTRANEmIKbGo83DF4Z-OJINU6wgHxme3ntEyN27ST5\_5\_M7nDSS7Tnq0RxEcVcEleLo6VJqiyiTE7yIG7yvUvtCr2QCt8BiumU88HnwjXLPLdpeF-ZpI8vBoR8EtFH6Ok\_2XrgRgOGOeC9dGZWzNidPSppOCDGcawxWVnGKMGyMU-4jgHT-NP3uLWZy396OLQi6v1oMTQySdsqpPYaDu0-3hImIv3XkUhvpyVcr-455pq-IvRWRQp\_4wsnMAB9itxGDDKYzX\_rWSd4uwmBdaKTqwPo7nCqX-xD\_QnEEozWwVW15SRkZOaxA

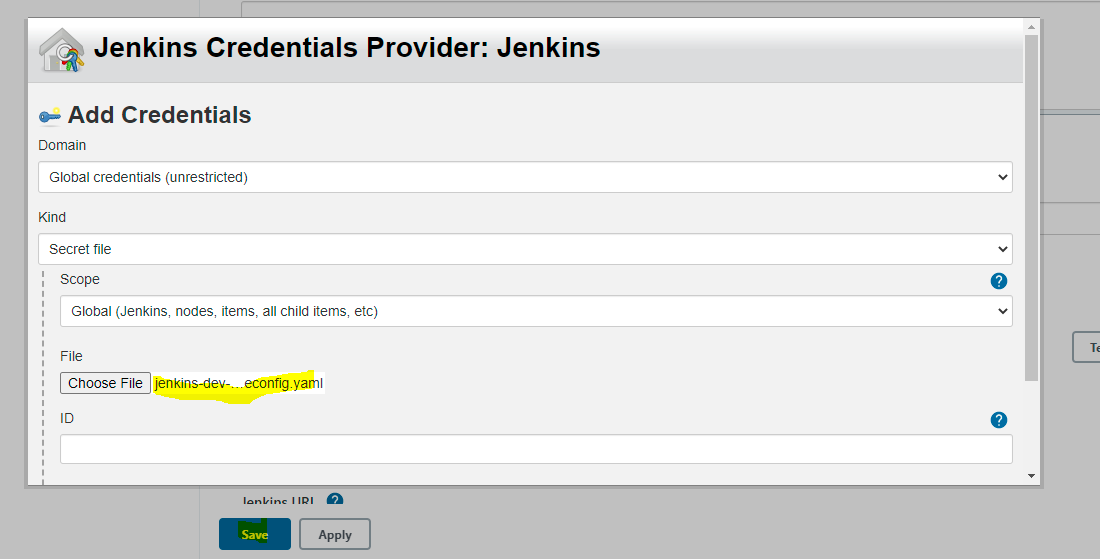
Check if its working or not :

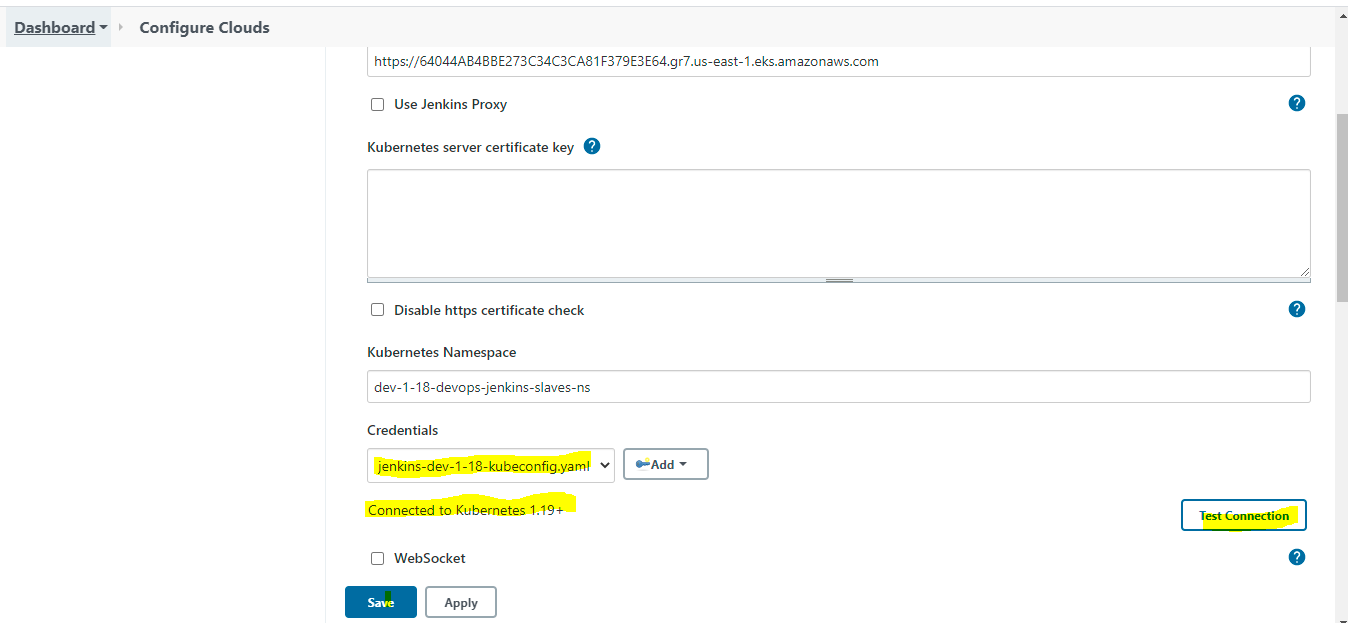
[root@eks opt]# kubectl -n dev-1-18-devops-jenkins-slaves-ns --kubeconfig jenkins-dev-1-18-kubeconfig.yaml auth can-i get pod

**Yes**

## Jenkins Kubernetes Credentials

Go back to the Jenkins, add a new Credential:

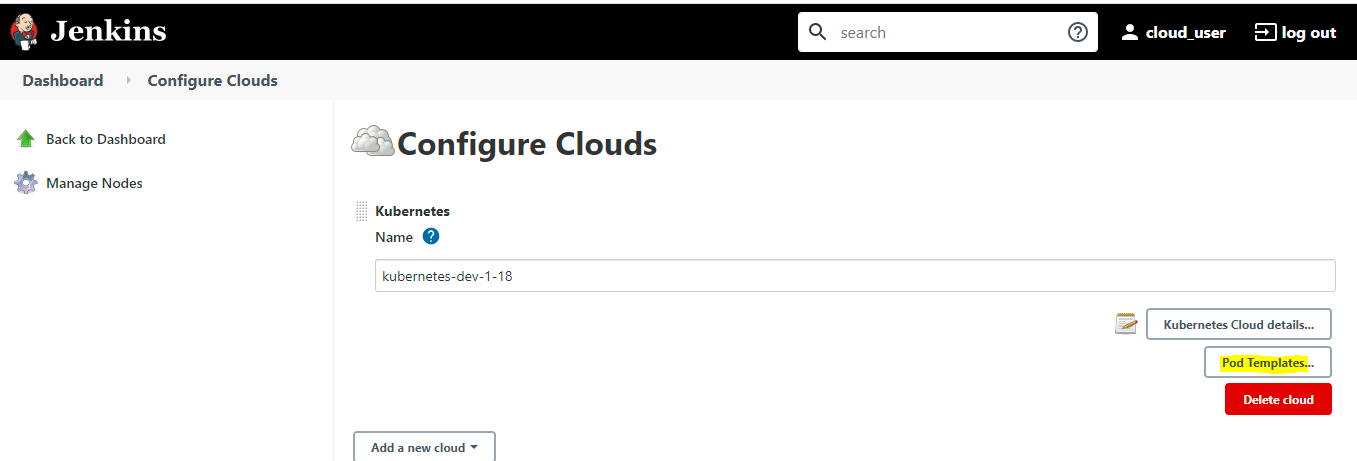




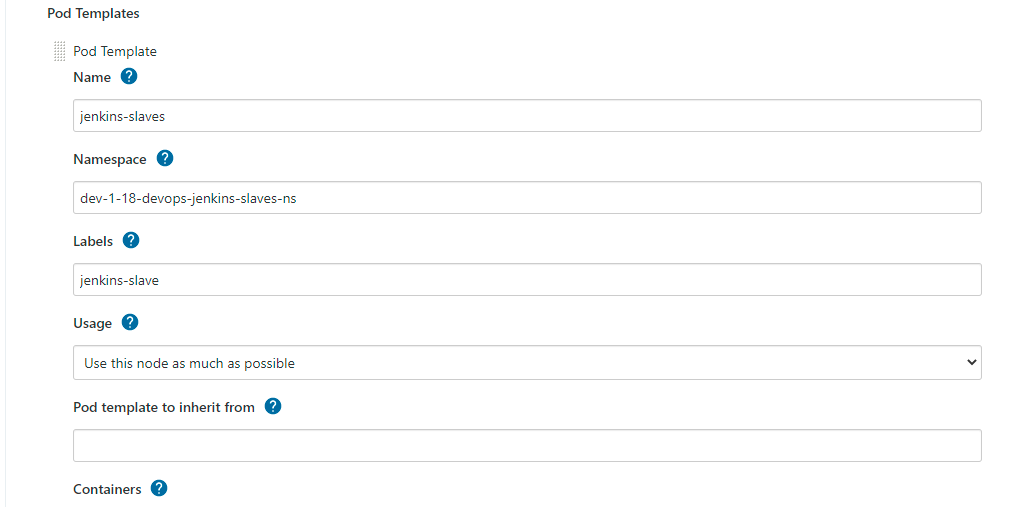
Save the settings.

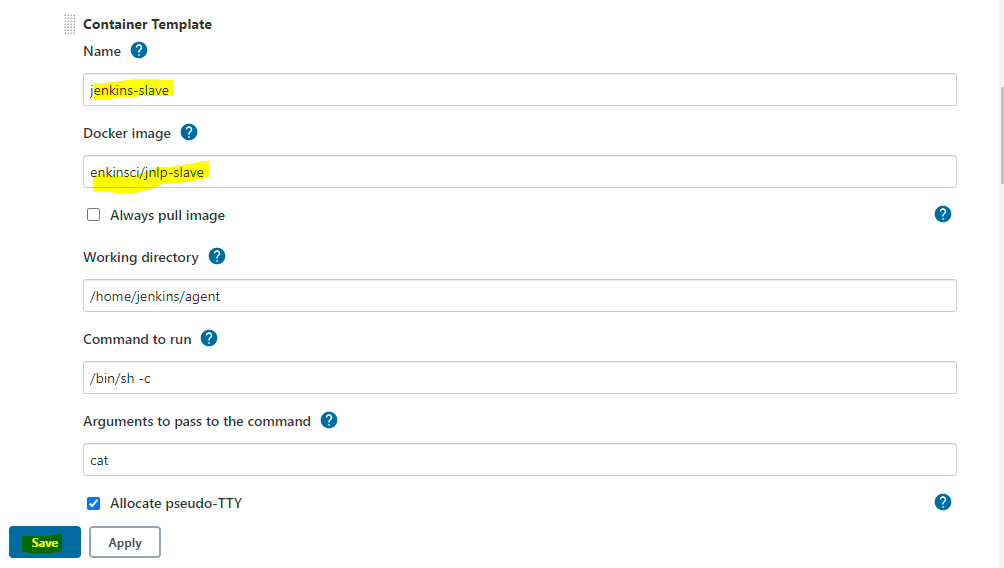
## Jenkins Slaves Pod Template

Go to the Pod templates:



Fill the fields of the pod and its default container, set the jenkinsci/jnlp-slave as the Docker image:





## Jenkins Job

Create a test job with the Pipeline type:

Write a pipeline’s script:

podTemplate {

node(POD\_LABEL) {

stage('Run shell') {

sh 'echo hello world'

}

}

}

Note:

Need to open port 50000 in SG else you will face below error during POD creation

# kubectl logs -n dev-1-18-devops-jenkins-slaves-ns k8s-test-3-c9vzp-p2h5w-lkl18

May 15, 2021 8:40:43 AM hudson.remoting.jnlp.Main createEngine

INFO: Setting up agent: k8s-test-3-c9vzp-p2h5w-lkl18

May 15, 2021 8:40:43 AM hudson.remoting.jnlp.Main$CuiListener <init>

INFO: Jenkins agent is running in headless mode.

May 15, 2021 8:40:43 AM hudson.remoting.Engine startEngine

INFO: Using Remoting version: 4.3

May 15, 2021 8:40:43 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir

INFO: Using /home/jenkins/agent/remoting as a remoting work directory

May 15, 2021 8:40:43 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging

INFO: Both error and output logs will be printed to /home/jenkins/agent/remoting

May 15, 2021 8:40:43 AM hudson.remoting.jnlp.Main$CuiListener status

INFO: Locating server among [http://35.168.36.51:8080/]

May 15, 2021 8:40:43 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver resolve

INFO: Remoting server accepts the following protocols: [JNLP4-connect, Ping]

May 15, 2021 8:40:48 AM org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver isPortVisible

WARNING: connect timed out

May 15, 2021 8:40:48 AM hudson.remoting.jnlp.Main$CuiListener error

SEVERE: http://35.168.36.51:8080/ provided port:50000 is not reachable

java.io.IOException: http://35.168.36.51:8080/ provided port:50000 is not reachable

at org.jenkinsci.remoting.engine.JnlpAgentEndpointResolver.resolve(JnlpAgentEndpointResolver.java:314)

at hudson.remoting.Engine.innerRun(Engine.java:693)

at hudson.remoting.Engine.run(Engine.java:518)

Run the job and check Jenkins’ logs:

