# Kubernetes Lab

# Connect to Kubernetes Master Node using the Public IP

# Open an SSH client.

# Locate your private key file (singaporekeypair.pem)

# # chmod 400 singaporekeypair.pem

# Now you will be able to SSH using your Public DNS/Public IP

# ssh -i "singaporekeypair.pem" [ec2-user@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com](mailto:ec2-user@ec2-xx-xx-xx-xx.ap-south-1.compute.amazonaws.com)

# NB:- If SSH connectivity does not work, see changing the chmod to 600 (chmod 600 singaporekeypair.pem)

# [ec2-user@ip-xx-xx-xx-xx ~]$ sudo su

# # yum install -y git [Only If GIT is not installed in your EC2 instance]

# Steps to Check out from git repository

# # git init

# # git config --global user.email "nevin.cleetus@gmail.com"

# # git config --global user.name "nevin-cleetus"

# # mkdir gitrepo

# # cd gitrepo

# # git clone https://github.com/nevin-cleetus/k8.git

# Ensure

# 1. Internet is working

# 2. Nobody should be connected to vpn or any other proxy.

# 3. Disable firewall if enabled.

# 

# 

# 

# Lab Exercise 1

# Role Based Access control

# Verify the Kubernetes Master and Worker Node(s) are in running state

# [ec2-user@ip-xx-xx-xx-xx ~]$ sudo su

[root@ip- kubernetes] kubectl get nodes

Confirm both Master and Worker node is in Ready state.

NAME STATUS ROLES AGE VERSION

ip- xxxxxxx Ready master 15m v1.18.0

ip-xxxxxxx Ready <none> 10m v1.18.0

[root@ip- kubernetes] kubectl api-versions | grep -i 'rbac.authorization.k8s.io/v1'

Expected Result

bac.authorization.k8s.io/v1

rbac.authorization.k8s.io/v1beta1

1. Create a user name Dave
2. Give Dave access to a namespace called ‘devops’
3. Private Key (dave.key), Certificate Signing Request for Dave (dave.csr)

Requirements

1. Kubernetes Certificate Authority private key and certificate in order to sign Dave’s certificate request (/etc/Kubernetes/pki)

Dave would need his private key and certificate to login to the Kubernetes

**Authentication - Generate Certificate and Config file**

Connect to Kubernetes master

1. Create a directory in Master node to work on the security

[root@ip- kubernetes] mkdir /home/ec2-user/security

1. Create the private key

[root ip- kubernetes] cd /home/ec2-user/security

[root@ip- kubernetes] openssl genrsa -out dave.key 2048

Output: - Generating RSA private key, 2048 bit long modulus

.....+++

.....+++

e is 65537 (0x10001)

1. Create a certificate signing request

[root ip- kubernetes] cd /home/ec2-user/security

[root@ip- kubernetes] openssl req -new -key dave.key -out dave.csr -subj "/CN=dave/O=devops"

[root@ip- kubernetes] ls

dave.csr dave.key

1. Sign Dave’s certificate

We need Certificate Authorities certificate and private key

[root@ip- kubernetes] cp /etc/kubernetes/pki/ca.crt /home/ec2-user/security/

[root@ip- kubernetes] sudo cp /etc/kubernetes/pki/ca.key /home/ec2-user/security/

[root@ip- kubernetes] openssl x509 -req -in dave.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out dave.crt -days 365

Signature ok

subject=/CN=dave/O=devops

Getting CA Private Key

1. We need to create a kube-config file for Dave to access our Kubernetes cluster.

[REPLACE THE KUBE API SERVER IP BELOW ]

[root@k8masterMLB rback-dave]# kubectl --kubeconfig dave.kubeconfig config set-cluster kubernetes --server <https://172.42.42.100:6443> --certificate-authority=ca.crt

Cluster "kubernetes" set.

Add user to the kube-config file

kubectl --kubeconfig dave.kubeconfig config set-credentials dave --client-certificate dave.crt --client-key dave.key

User "dave" set.

[root@k8masterMLB rback-dave]# kubectl create namespace devops

Set the context

[root@k8masterMLB rback-dave]# kubectl --kubeconfig dave.kubeconfig config set-context dave-kubernetes --cluster kubernetes --namespace devops --user dave

Context "dave-kubernetes" created.

[root@k8masterMLB rback-dave]# vi dave.kubeconfig

Modify the current-context to dave-kubernetes

current-context: dave-kubernetes

Now Dave can copy the dave.kubeconfig file to WORKER NODE ~/.kube/config

Or Pass the kubeconfig as parameter to point to the config file as below

[root@k8masterMLB rback-dave]# kubectl --kubeconfig dave.kubeconfig get pods

Error from server (Forbidden): pods is forbidden: User "dave" cannot list resource "pods" in API group "" in the namespace "devops"

**Authorization**

We could add roles to the user ‘dave’ by running YAML or from Kubectl inline command

Below steps is to be done by administrator who has complete permission. You may use either kubectl through command line or using the YAML. Both approaches are below.

**Using kubectl inline commands**

[root@k8masterMLB rback-dave]# kubectl create role dave-devops --verb=get,list --resource=pods --namespace devops

role.rbac.authorization.k8s.io/dave-devops created

kubectl create rolebinding dave-devops-role-binding --role=dave-devops --user=dave --namespace devops

rolebinding.rbac.authorization.k8s.io/dave-devops-role-binding created

**Using YAML**

# [ec2-user@ip-xx-xx-xx-xx ~]$ cd <HOME>/gitrepo/k8//home/vagrant/gitrepo/k8/day4/

# 

# [ec2-user@ip-xx-xx-xx-xx ~]$ kubectl create -f role.yaml

# [ec2-user@ip-xx-xx-xx-xx ~]$ kubectl create -f rolebinding.yaml

[root@k8masterMLB rback-dave]# kubectl --kubeconfig dave.kubeconfig get pods

[root@k18master security]# kubectl get pods

# No resources found in devops namespace.

# 

# Dave have permission only to view a POD under the devops namespace. You may now create a POD using administrator config file and try access the pods from dave’s config file.

[root@k18master security]# kubectl --kubeconfig dave.kubeconfig get pods

NAME READY STATUS RESTARTS AGE

nginxdevops 0/1 ContainerCreating 0 6s

[root@k18master security]# kubectl --kubeconfig dave.kubeconfig get pods

Error from server (Forbidden): services is forbidden: User "dave" cannot list resource "services" namespace "devops"