Universidad Mariano Gálvez de Guatemala

Ingeniería en Sistemas

Sistemas operativos 2 sección A

Proyecto – Fase 2

Integrantes:

Jonatán Rodolfo Suruy Figueroa 3190-15-2425

José Antonio Rojas Gómez 3190-17-2307

Jorge Eduardo Miguel Aguilar Oliva 3190 – 17 -11155

Santos Yonatan Zurdo Sunun 3190-14-6019

Francisco Alejandro Barrios Meléndez 3190-15-2711

Introducción

El siguiente documento comprende una guía para levantar la segunda fase del proyecto de sistemas operativos 2. El alcance fue el siguiente: Un nodo master, dos nodos esclavos, un pod para levantar el dashboard de Kubernetes y un pod para levantar una imagen de Nginx. El ambiente fue completamente virtualizado por medio de virtual box. Las tres máquinas virtuales utilizadas fueron creadas con una imagen de Centos 7.

Guía

OpenSSH

Para empezar, debe instalarse el servidor OpenSSH en todas las máquinas. Esto facilita la comunicación y el resto de comandos pueden ser ejecutados desde una misma máquina.

1) Instalamos el servidor OpenSSH con el comando

$sudo yum install openssh-server

2) Modificamos el fichero sshd_config para habilitar el puerto 22

$sudo nano /etc/ssh/sshd_config

3) Reiniciamos el servidor OpenSSH

$sudo systemctl reload sshd

4) Configuramos el firewall y habilitamos el puerto 22

$sudo firewall-cmd --permanent --add-port=2244/tcp

5) Reiniciamos el firewall

$sudo firewall-cmd –reload

Cluster Kubernetes

Luego de instalar el servidor OpenSSH se continua con la instalación del cluster de kubernetes. (Para este paso y el resto podría utilizarse el comando ssh seguido de la dirección IP para controlar todas las maquinas desde una misma)

la instalación se realiza de manera individual, ya sea para el master o para los nodos(esclavos), es importante establecer una IP estática a cada máquina que se usará.

Master

Instalación en el master node

1) se le cambia el nombre al host de la máquina que será el master

$hostnamectl set-hostname master-node

2) Se modifica el archivo del host ubicado en /etc/hosts, se puede modificar por medio de cat o nano, en nuestro caso preferimos nano, en ese archivo se pondrá las IP de cada máquina y el nombre del host según su papel a desarrollar, en nuestro caso.

$nano /etc/hosts

192.168.0.14 master-node

192.168.0.15 node-1 worker-node-1

192.168.0.16 node-2 worker-node-2

3) Desactivar SElinux y reiniciar la máquina

$setenforce 0

$sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux

$reboot

4) Establecer nuevas reglas al firewall y actualizar la misma

firewall-cmd --permanent --add-port=6443/tcp

firewall-cmd --permanent --add-port=2379-2380/tcp

firewall-cmd --permanent --add-port=10250/tcp

firewall-cmd --permanent --add-port=10251/tcp

firewall-cmd --permanent --add-port=10252/tcp

firewall-cmd --permanent --add-port=10255/tcp

firewall-cmd --reload

modprobe br_netfilter

echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

5) Se deberá agregar repositorios Kubernetes manualmente, ya que no vienen instalados de forma predeterminada en CentOS 7, de igual manera se puede hacer por cat o nano, en nuestro caso preferimos nano, se guardará los siguientes comandos:

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64

enabled=1

gpgcheck=1

repo_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

6) Una vez creado el repositorio se procederá instalar kubeadm y docker si no se ha instalado. Hoy en día Docker es una herramienta muy utilizada. La implementación de aplicaciones en forma de contenedores nos salva de problemas de compatibilidad. Además, Docker facilita la instalación y el mantenimiento de los contenedores existentes.

$yum install kubeadm docker -y

7) Cuando la instalación ya este completa se procederá a activar los servicios

$systemctl enable kubelet

$systemctl start kubelet

$systemctl enable docker

$systemctl start docker

8)Ahora estamos listos para inicializar kubernetes master, pero antes de eso debe deshabilitar el intercambio para ejecutar el comando "kubeadm init".

$swapoff -a

9) La inicialización de Kubernetes master es un proceso totalmente automatizado que se gestiona mediante el comando "kubeadm init" que ejecutará, este proceso solo se realizará en el master, se tiene que copiar la última línea, el cual es el token que se usará en los nodos.

$kubeadm init

$kubeadm join 192.168.0.14:6443 --token 6ipbu3.c6v76rmu4zy5s617 \

--discovery-token-ca-cert-hash\

sha256:604ad816ab3ee39d71d38eeb81deee5e314a8d02538e8c7df5bb83dae0ac2eb8

en algunas ocasiones el \ puede ocasionar problemas, se sugiere quitarlo

$kubeadm join 192.168.0.14:6443 --token 6ipbu3.c6v76rmu4zy5s617 --discovery-token-ca-cert-hash sha256:604ad816ab3ee39d71d38eeb81deee5e314a8d02538e8c7df5bb83dae0ac2eb8

10) Ahora se puede configurar para que el usuario o el root utilice el cluster en nuestro caso elegiremos el root

$mkdir -p $HOME/.kube

$sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

$sudo chown $(id -u):$(id -g) $HOME/.kube/config

11) Verificaremos si el comando kubectl está activado

$kubectl get nodes

este punto nos aparecerá como NotReady esto se debe a q tenemos que implementar la red pod. La red pod es una red superpuesta para el cluster que se implementa en la parte superior de la red del nodo actual.

11) La implementación del clúster de red es un proceso altamente flexible que depende de sus necesidades y hay muchas opciones disponibles.

$export kubever=$(kubectl version | base64 | tr -d '\n')

$kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"

12) A continuación checamos nuevamente los nodos y ahora aparece ready

$kubectl get nodes

Esclavos

Configuración de los nodos esclavos

Los pasos son los mismos que se usaron con el maestro exceptuando el kubeadm init en adelante, esto se realizará en todos los nodos .

1) Se le cambia el nombre al host de la máquina según el nombre del nodo esclavo en cada máquina.

$hostnamectl set-hostname node-1

2) se modifica el archivo del host ubicado en /etc/hosts, por medio de nano, en esta parte usamos la comunicación con ssh para poder copiar el archivo hosts del master y solo copiar la información.

$nano /etc/hosts

192.168.0.14 master-node

192.168.0.15 node-1 worker-node-1

192.168.0.16 node-2 worker-node-2

3) Desactivar SElinux y reiniciar la máquina

$setenforce 0

$sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux

$reboot

4) establecer nuevas reglas al firewall y actualizar la misma

firewall-cmd --permanent --add-port=6443/tcp

firewall-cmd --permanent --add-port=2379-2380/tcp

firewall-cmd --permanent --add-port=10250/tcp

firewall-cmd --permanent --add-port=10251/tcp

firewall-cmd --permanent --add-port=10252/tcp

firewall-cmd --permanent --add-port=10255/tcp

firewall-cmd --reload

modprobe br_netfilter

echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

5) Se deberá agregar repositorios Kubernetes manualmente, ya que no vienen instalados de forma predeterminada en CentOS 7, de igual manera se puede hacer por cat o nano, en nuestro caso preferimos nano, se guardará los siguientes comandos

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64

enabled=1

gpgcheck=1

repo_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

6) Una vez creado el repositorio se procederá instalar kubeadm y docker si no se ha instalado. Hoy en día Docker es una herramienta muy utilizada. La implementación de aplicaciones en forma de contenedores nos salva de problemas de compatibilidad. Además, Docker facilita la instalación y el mantenimiento de los contenedores existentes.

$yum install kubeadm docker -y

7) cuando la instalación ya esté completa se procederá a activar los servicios

$systemctl enable kubelet

$systemctl start kubelet

$systemctl enable docker

$systemctl start docker

8) Ahora estamos listos para inicializar kubernetes master, pero antes de eso debe deshabilitar el intercambio para ejecutar el comando "kubeadm init".

$swapoff -a

9) Por medio del túnel ssh se accederá al archivo donde hayamos copiado el token y lo pegaremos en nuestra terminal, al terminal vamos al master ah verificar que aparezca el nodo y este ready, es recomendable esperar unos minutos a que termine de iniciar.

$kubeadm join 192.168.0.14:6443 --token 6ipbu3.c6v76rmu4zy5s617 --discovery-token-ca-cert-hash sha256:604ad816ab3ee39d71d38eeb81deee5e314a8d02538e8c7df5bb83dae0ac2eb8

Levantar imagen de Nginx

Luego de tener los nodos listos (Master – 2 esclavos) podemos realizar el despliegue de una app por medio de deploy nginx en nuestro cluster de kubernetes

1) Verificamos que todos los nodos y maestro esten ready

$kubectl get nodes

2) Crear un deploy de nginx usando una imagen de nginx

$kubectl create deployment nginx --image=nginx

3) Ahora podemos ver el estado del deploy por medio del siguiente codigo

$kubectl get deployments

4) Si se desea ver más detalles usar el siguiente comando, es prudente esperar ah que la imagen tenga estado true

$kubectl describe deployment nginx

5) Una vez activo el deploy se procederá a exponer nuestro nginx a una network pública usaremos

La propiedad type: NodePort, expone el servicio en cada nodo del cluster de forma que serás capaz de contactar con el servicio desde cualquier ip de los nodos.

$kubectl create service nodeport nginx --tcp=80:80

6) Ahora Ejecutamos el comando get svc para ver un resumen del servicio y los puertos expuestos.

$kubectl get svc

7) Verifique que la implementación de NGINX sea exitosa utilizando curl en el nodo esclavo, La salida mostrará el mensaje "¡Bienvenido a nginx!" página HTML.

$curl node-1:puerto expuesto

8) para escalar la aplicación se puede hacer fácilmente con el siguiente comando, Teniendo en cuenta que los pods siempre estarán distribuidos de la forma más eficiente entre los nodos del cluster.

$kubectl scale deploy nginx --replicas=4

9) Para eliminar el escenario, eliminamos el deployment y el service

$ kubectl delete deploy my-nginx

$ kubectl delete svc my-nginx-service

Dashboard Kubernetes

El dashboard no se instala junto con el cluster, por lo que es necesario levantarlo en un pod. Los siguientes pasos se realizan en el master.

1) Crear el pod

$kubectl create –f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/alternative/kubernetes-dashboard.yaml

2) Verificar el estado ready del pod “kube-system”

$kubectl get pods --all-namespaces

ó

$kubectl get pods --namespace kube-system

3) Exponer el dashboard

$kubectl proxy --address 0.0.0.0 --accept-hosts '.*'

4) Acceder al dashboard, esta acción se realiza por medio del siguiente link: http://192.168.0.14:8001/api/v1/namespaces/kube-system/services/kubernetes-dashboard/proxy/#!/node?namespace=default

Nota: El dashboard solo será posible de acceder por medio de la maquina en la que se haya levantado (Master).

5) Luego de esto será necesario dar acceso para poder utilizar el dashboard. Lo cual se hacer por medio del siguiente código.

```
$ kubectl create clusterrolebinding kubernetes-dashboard \
  --clusterrole=cluster-admin \
  --serviceaccount=kube-system:kubernetes-dashboard
```

De esta manera ya se puede visualizar el dashboard por completo y utilizar todas sus funcionalidades.

Luego de concretar la creación y despliegue del dashboard junto con una imagen se pretende continuar con la adición de un nuevo maestro.

Así mismo, se continuará con el despliegue de una imagen, pero extrayéndola de un repositorio privado.