Sistema híbrido de kubernetes (VMs + Pods)

Objetivos

-Combinar el uso de pods y máquinas virtuales
-Usar máquinas virtuales dentro del sistema de kubernetes
-Usar replicaset en máquinas virtuales

¿Qué quiero conseguir?

Usar ambas tecnologías para desplegar un conjunto de aplicaciones eliminando el hándicap de la ausencia de kernel en contenedores usando kubeadm y Kubevirt

Escenario

Kubernetes

- Master
- Worker

Containerd

Instalación

Empezaremos por instalar y configurar los módulos del kernel necesarios para containerd, para ello ejecutamos las siguientes lineas

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
```

Después configuraremos los parámetros necesarios en sysctl, para ello ejecutamos las siguientes lineas

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
```

Para que los cambios sean permanentes tendremos que ejecutar `sudo sysctl --system`

A continuación descargaremos y descomprimiremos el comprimido que contiene containerd y sus herramientas de su repositorio oficial

```
wget https://github.com/containerd/containerd/releases/download/v1.4.3/cri-containerd-cni-1.4.3-linux-amd64.tar.gz
tar xvf cri-containerd-cni-1.4.3-linux-amd64.tar.gz
```

Después copiaremos las carpetas resultantes a su correspondiente ubicación

```
cp -r etc/* /etc/
cp -r usr/* /usr/
cp -r opt/* /opt/
```

Después

Lo mismo realizaremos con nuestra maquina que actuará como worker de Kubernetes

Kubernetes

Instalación

Empezaremos accediendo a nuestra maquina que actuará como nodo master.

Después añadiremos el repositorio oficial de kubernetes, para ello ejecutaremos las siguientes lineas:

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list \
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

apt-get update
```

Después instalaremos el demonio de kubernetes (kubelet) las herramientas de configuración de kubernetes (kubeadm) y el cliente de kubernetes (kubectl), también fijaremos los paquetes para evitar que se actualice en futuras posibles actualizaciones de paquetes ya que normalmente no actualizamos estos paquetes sin ser extremadamente necesario

```
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl
```

Realizaremos los mismos pasos con la maquina que actuará como nodo worker

Configuración

Kubeadm

Empezaremos con el nodo master.

Empezaremos por cambiar el cgroup driver de docker predeterminado por `systemd`, para ello nos vamos al .service de docker y añadimos la linea `--exec-opt native.cgroupdriver=systemd \` al apartado `ExecStart`

Después iniciaremos Kubernetes con un parámetro que asignará IPs a los pods en una red determinada, para ello ejecutaremos la siguiente linea

```
kubeadm init --pod-network-cidr=<ip_deseada>/<cidr>
```

```
kubeadm init --pod-network-cidr=10.7.0.0/16
```

Resultado

```
W1202 11:26:03.506880   22609 configset.go:348] WARNING: kubeadm cannot validate
component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
[init] Using Kubernetes version: v1.19.4
[preflight] Running pre-flight checks
    [WARNING SystemVerification]: missing optional cgroups: hugetlb
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your
internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config
images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes
kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local
master] and IPs [10.96.0.1 192.168.1.4]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master] and
IPs [192.168.1.4 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master] and IPs
[192.168.1.4 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as
static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 27.615078 seconds
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in
the "kube-system" Namespace
```

```
[kubelet] Creating a ConfigMap "kubelet-config-1.19" in namespace kube-system
with the configuration for the kubelets in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node master as control-plane by adding the label
"node-role.kubernetes.io/master=''"
[mark-control-plane] Marking the node master as control-plane by adding the
taints [node-role.kubernetes.io/master:NoSchedule]
[bootstrap-token] Using token: qq4zo7.l6e73wwh8an84x8s
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC
Roles
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get
nodes
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post
CSRs in order for nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller
automatically approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all
node client certificates in the cluster
[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public"
namespace
[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a
rotatable kubelet client certificate and key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as
root:

kubeadm join 192.168.1.4:6443 --token qq4zo7.l6e73wwh8an84x8s \
    --discovery-token-ca-cert-hash
sha256:e3f914f134b40d3e61c24cde28264fa424d416e20c6d3aabf9e4808d8c981ee1
```

Una vez terminado nos proporcionará una breve indicación de que tenemos que introducir en nuestro nodo worker

Después nos conectaremos a nuestra maquina que actuará como nodo worker de Kubernetes e introduciremos el comando que nos ha indicado en el resultado de la instalación de kubeadm en nuestra maquina que actuará como nodo master

```
kubeadm join 192.168.1.4:6443 --token qq4zo7.l6e73wwh8an84x8s \
    --discovery-token-ca-cert-hash
sha256:e3f914f134b40d3e61c24cde28264fa424d416e20c6d3aabf9e4808d8c981ee1
```

Resultado

```
[preflight] Running pre-flight checks
    [WARNING SystemVerification]: missing optional cgroups: hugetlb
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system
get cm kubeadm-config -oyaml'
[kubelet-start] Writing kubelet configuration to file
"/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file
"/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Kubectl

Empezaremos ejecutando las siguientes lineas para copiar el fichero de configuración del cliente de kubernetes al usuario que usemos de nuestra maquina master.

Antes de ejecutarlo asegúrate de que estas logeado con el usuario deseado ya que utiliza variables de entorno que cambian su valor según el usuario que uses

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

También podemos llevarnos el fichero de configuración de kubectl a donde prefiramos usar dicho cliente, en mi caso en mi maquina real, para ello ejecutamos las siguientes lineas

```
scp debian@192.168.1.4:.kube/config
cp config ~/.kube/config
```

Después comprobaremos que esta correctamente instalado

```
kubectl get nodes -o wide
```

Resultado

```
NAME     STATUS   ROLES    AGE    VERSION   INTERNAL-IP   [...]
master   Ready    master   3d4h   v1.19.4   192.168.1.4   [...]
worker   Ready    <none>   3d3h   v1.19.4   192.168.1.5   [...]
```

Calico CNI

Instalación

Empezaremos por instalar el operador de Calico en Kubernetes, para ello ejecutamos la siguiente linea

```
kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml
```

Configuración

Empezaremos descargando el fichero `custom-resources.yaml` de la pagina oficial y modificaremos el yaml cambiando el valor del parámetro `cidr` por nuestro direccionamiento de red que previamente configuramos en `kubeadm`

```
wget https://docs.projectcalico.org/manifests/custom-resources.yaml
```

Una vez modificado el fichero quedaría de la siguiente manera

```
# This section includes base Calico installation configuration.
# For more information, see:
https://docs.projectcalico.org/v3.17/reference/installation/api#operator.tigera.i
o/v1.Installation
apiVersion: operator.tigera.io/v1
kind: Installation
metadata:
  name: default
spec:
  # Configures Calico networking.
  calicoNetwork:
    # Note: The ipPools section cannot be modified post-install.
    ipPools:
    - blockSize: 26
      cidr: 10.7.0.0/16
      encapsulation: VXLANCrossSubnet
      natOutgoing: Enabled
      nodeSelector: all()
```

Después implantamos dicho yaml en nuestro servidor de Kubernetes, para ello ejecutaremos la siguientes linea

```
kubectl create -f custom-resources.yaml
```

Kubevirt

¿Que es Kubevirt?

KubeVirt es un proyecto open-source que ofrece la opción de virtualizar de VMs sobre un cluster Kubernetes.

Repositorio oficial
Características:

- Esta desarrollado en GO
- Permite el uso de maquinas virtuales en Kubernetes

- Usa como tarjeta de red "virtio"
- Sus interfaces pueden ser "NAT" o "bridge"
- Permite la conexión de escritorio remoto por VNC

Instalación

Kubevirt

Empezaremos por crear una variable que contendrá la ultima versión de Kubevirt

```
export KUBEVIRT_VERSION=$(curl -s
https://api.github.com/repos/kubevirt/kubevirt/releases/latest | grep tag/v | tr
'"' " " | tr "," " " | tr "/" "\n" | grep v[0-9])
```

Después implementaremos el operador, para ello ejecutamos la siguiente linea

```
kubectl apply -f https://github.com/kubevirt/kubevirt/releases/download/$(echo
$KUBEVIRT_VERSION)/kubevirt-operator.yaml
```

A continuación activaremos la emulación para Kubevirt, para ello ejecutamos la siguiente linea

```
kubectl create configmap kubevirt-config -n kubevirt --from-literal
debug.useEmulation=true
```

Después desplegaremos las definiciones de los recursos de Kubevirt

```
kubectl apply -f https://github.com/kubevirt/kubevirt/releases/download/$(echo
$KUBEVIRT_VERSION)/kubevirt-cr.yaml
```

Después comprobaremos que todos los componentes están instalados y funcionando.

Empezaremos ejecutando la siguiente linea

```
kubectl get kubevirt.kubevirt.io/kubevirt -n kubevirt -o=jsonpath="
{.status.phase}"
```

El resultado correcto sera "Deployed"

Después verificaremos el estado delos Componentes de Kubevirt, para ello ejecutamos la siguiente linea

```
kubectl get all -n kubevirt
```

Virtctl

A continuación descargaremos la herramienta para controlar las maquinas virtuales, para ello descargaremos, asignaremos permisos y lo instalaremos con los siguientes comandos

```
curl -L -o virtctl https://github.com/kubevirt/kubevirt/releases/download/$(echo
$KUBEVIRT_VERSION)/virtctl-$(echo $KUBEVIRT_VERSION)-linux-amd64
chmod +x virtctl
sudo install virtctl /usr/local/bin
```

Después comprobaremos que esta instalado para ello ejecutamos el comando `virtctl`

Resultado

```
Available Commands:
  console      Connect to a console of a virtual machine instance.
  expose       Expose a virtual machine instance, virtual machine, or virtual
machine instance replica set as a new service.
  fslist       Return full list of filesystems available on the guest machine.
  guestosinfo  Return guest agent info about operating system.
  help         Help about any command
  image-upload Upload a VM image to a DataVolume/PersistentVolumeClaim.
  migrate      Migrate a virtual machine.
  pause        Pause a virtual machine
  rename       Rename a stopped virtual machine.
  restart      Restart a virtual machine.
  start        Start a virtual machine.
  stop         Stop a virtual machine.
  unpause      Unpause a virtual machine
  userlist     Return full list of logged in users on the guest machine.
  version      Print the client and server version information.
  vnc          Open a vnc connection to a virtual machine instance.

Use "virtctl <command> --help" for more information about a given command.
Use "virtctl options" for a list of global command-line options (applies to all
commands).
```

Importación de maquinas virtuales en KubeVirt

Para usar tus propias imágenes de maquinas virtuales deberemos de crear una imagen de docker que contenga el disco que tengamos como imagen, para ello usaremos el siguiente `Dockerfile`

```
FROM scratch
ADD <imagen/disco_a_añadir>.<formato> /disk/
```

Después construiremos la imagen con el comando `docker build -t <registro>/<nombre_imagen>:<version> .`

Creación de una maquina virtual

Empezaremos por crear una instancia para nuestra maquina virtual, este es un ejemplo

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  name: debian10
spec:
  running: false
  template:
    metadata:
      labels:
        kubevirt.io/size: small
        kubevirt.io/domain: debian10
    spec:
      domain:
```

```
        devices:
          disks:
          - disk:
              bus: virtio
            name: rootfs
          interfaces:
          - name: default
            masquerade: {}
        resources:
          requests:
            memory: 1024M
      networks:
      - name: default
        pod: {}
      volumes:
        - name: rootfs
          containerDisk:
            image: machde/debian-qcow2:latest
```

Posibles errores y sus soluciones

El PATH de root no contiene la ruta `/sbin`

```
root@master:/home/debian# kubeadm init --pod-network-cidr=10.32.0.0/16
W1130 15:51:07.952852   10099 configset.go:348] WARNING: kubeadm cannot validate
component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
[init] Using Kubernetes version: v1.19.4
[preflight] Running pre-flight checks
    [WARNING FileExisting-ebtables]: ebtables not found in system path
    [WARNING FileExisting-ethtool]: ethtool not found in system path
    [WARNING FileExisting-tc]: tc not found in system path
    [WARNING SystemVerification]: missing optional cgroups: hugetlb
error execution phase preflight: [preflight] Some fatal errors occurred:
    [ERROR FileExisting-conntrack]: conntrack not found in system path
    [ERROR FileExisting-iptables]: iptables not found in system path
[preflight] If you know what you are doing, you can make a check non-fatal with
`--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher
```

Esto se debe a que se ha ejecutado como root pero sin su path "sbin"

Para ello iniciaremos la sesion como `root`, en el caso de que accedamos por ssh y tengamos bloqueado la conexion directa por ssh mediante el usuario `root` usaremos el comando `su -`, `sudo su` o `sudo login root` en la conexion que hemos establecido con el servidor

Links

Calico CNI implementation
Kubeadm install
Kubevirt install
Runtimes for Kubernetes