Kubernetes microk8s

Kubernetes es ahora mismo el estándar de facto para el despliegue de aplicaciones, y aquí ya hemos hablado muchas veces de él y de las necesidades de máquina que hacen que la mejor solución sea llevarlo al cloud.

Esto hace que sea muy costoso poder hacer pruebas en desarrollo de nuestros manifestos de Kubernetes y se hace prohibitivo para empresas que realmente por su negocio no necesitan un clúster de alta disponibilidad.

Entonces la solución pasa por poder ejecutar Kubernetes de forma local y esto es lo que podemos hacer con microk8s, que a diferencia de Minikube no requiere de una máquina virtual sino que podemos instalarlo directamente en Ubuntu como un paquete de snap para tener Kubernetes corriendo en nuestra máquina en segundos y consumiendo muchos menos recursos que si levantamos un clúster

Entorno

El siguiente tutorial esta diseñado para realizarse en:

- Arquitectura: Virtual Box VM 64 bits

- OS: Ubuntu mini básico de 64 bits

- Red: Adaptador de red en modo bridge (ver configuración de vm en virutal box)

Snappy

Snappy es un sistema de gestión de paquetes e implementación de software diseñado y creado originalmente por Canonical para el sistema operativo de teléfonos Ubuntu. Los paquetes, llamados 'snaps' y la herramienta para usarlos 'snapd', funcionan en un rango de distribuciones de Linux y, por lo tanto, permiten el despliegue de software en sentido ascendente. El sistema está diseñado para funcionar para el internet de las cosas, la nube y la computación de escritorio.

```
$ sudo apt update && sudo apt install snapd
```

Extras

Se recomienda instalar los siguientes paquetes

```
$ sudo apt install git vim net-tools openssh-server curl gnupg2
```

Deploy

Instalar micrk8s
MicroK8s instalará un Kubernetes mínimo y liviano que puede ejecutar y usar en prácticamente cualquier máquina.

```
$ sudo snap install microk8s --classic
```

Grupos

MicroK8s crea un grupo para permitir el uso continuo de comandos que requieren privilegios de administrador. Para agregar su usuario actual al grupo y obtener acceso al directorio de almacenamiento en caché **.kube**, ejecute los siguientes dos comandos:

```
$ sudo usermod -a -G microk8s $USER
$ sudo chown -f -R $USER ~/.kube
```

Status

MicroK8s tiene un comando incorporado para mostrar su estado. Durante la instalación, puede usar el indicador **--wait-ready** para esperar a que los servicios de Kubernetes se inicialicen:

```
$ microk8s status --wait-ready
microk8s is running
addons:
cilium: disabled
dashboard: disabled
dns: disabled
fluentd: disabled
gpu: disabled
helm: disabled
helm3: disabled
host-access: disabled
ingress: disabled
istio: disabled
jaeger: disabled
knative: disabled
kubeflow: disabled
linkerd: disabled
metallb: disabled
```

```
metrics-server: disabled
prometheus: disabled
rbac: disabled
registry: disabled
storage: disabled
```

Acceso
MicroK8s incluye su propia versión de `kubectl` para acceder a Kubernetes. Úselo para ejecutar comandos para monitorear y controlar sus Kubernetes. Por ejemplo, para ver su nodo:

```
$ microk8s kubectl get nodes
NAME     STATUS   ROLES    AGE     VERSION
ubuntu   Ready    <none>   4m58s   v1.18.4-1+6f17be3f1fd54a
```

O para listar servicios

```
$ microk8s kubectl get services
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.152.183.1   <none>        443/TCP
6m33s
```

Add Ons
MicroK8s utiliza el mínimo de componentes para un Kubernetes puro y ligero. Sin embargo, hay muchas funciones adicionales disponibles: componentes preempaquetados que proporcionarán capacidades adicionales para sus Kubernetes, desde la simple administración de DNS hasta el aprendizaje automático con Kubeflow.

```
$ microk8s enable dashboard istio
Enabling Kubernetes Dashboard
Addon metrics-server is already enabled.
Applying manifest
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard
created
```

```
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created

If RBAC is not enabled access the dashboard using the default
token retrieved with:
token=$(microk8s kubectl -n kube-system get secret | grep
default-token | cut -d " " -f1)
microk8s kubectl -n kube-system describe secret $token
In an RBAC enabled setup (microk8s enable RBAC) you need to create
a user with restricted
permissions as shown in:
https://github.com/kubernetes/dashboard/blob/master/docs/user/acce
ss-control/creating-sample-user.md
Enabling DNS
Applying manifest
serviceaccount/coredns created
configmap/coredns created
deployment.apps/coredns created
service/kube-dns created
clusterrole.rbac.authorization.k8s.io/coredns created
clusterrolebinding.rbac.authorization.k8s.io/coredns created
Restarting kubelet
DNS is enabled
```

Dashboard

Para acceder de forma externa al dashboard de kubernetes:

```
$ microk8s dashboard-proxy --address 0.0.0.0
Checking if Dashboard is running.
Dashboard will be available at https://127.0.0.1:10443
```