TRABAJANDO CON KUBERNETES AVANZADO

MÓDULO 1

High Performance
Consulting Group

Módulo 1: Deploy de Kubernetes mediante KinD

Introducción a los **componentes y objetos** de Kubernetes

Uso de **clústeres** de desarrollo

Instalación de **KinD**

Creación de un clúster KinD

Revisando su clúster KinD

Agregar un **balanceador de carga** personalizado para **Ingress**

Introducción a los componentes y objetos de Kubernetes

Componente	Descripción
Control Plane	**API-Server**: Acepta las peticiones de los clientes **Kube-Scheduler**: Asigna los Workloads a los nodos **Etcd**: Base de datos con todos los datos del clúster **Kube-Controller-Manager**: Observa los nodos, replicas, endpoints, servicios, cuentas y tokens
Nodo	**Kubelet**: El agente que ejecuta los Pods basado en las instrucciones del Control Plane **Kube-Proxy**: Crea y elimina reglas de red para la comunicación entre Pods **Container-Runtime**: Es el responsable de ejecutar los contenedores

Introducción a los componentes y objetos de Kubernetes

Objeto	Descripción
Container	Es una **imagen inmutable** que contiene todo lo necesario para ejecutar una aplicación
Pod	Es el **objeto más pequeño** que puede controlarse en Kubernetes. Este mantiene uno o múltiples contenedores. Todos los contenedores en el Pod son programados en el mismo servidor con un contexto compartido (el *nodo*)
Deployment	Despliega una aplicación en un **estado deseado**, incluyendo *replicas* del Pod y configuración sobre el *rolling update*

Introducción a los componentes y objetos de Kubernetes

Objeto	Descripción
Storage Class	Define un **proveedor de almacenamiento** y se lo presenta al clúster
Persistent Volume (PV)	Provee un **objetivo de almacenamiento** que puede ser reclamado por un *Persisten Volume Request*
Persistent Volume Claim (PVC)	**Conecta (reclama)** un *Persistent Volume* que pueda ser usado dentro del Pod

Introducción a los componentes y objetos de Kubernetes

Objeto	Descripción
Container Network Interface (CNI)	Provee las **conexiones de red** para los Pods, por ejemplo, Flannel y Calico
Container Storage Interface (CSI)	Provee la conexión entre los Pods y los **sistemas de almacenamiento**

Uso de clústeres de desarrollo

Cluster	Descripción
Docker Swarm	Está **integrado a Docker** y permite administrar nodos del tipo Manager-Worker que pueden ser distribuidos en diferentes Hosts
Minikube	Consiste en un **administrador de Kubernetes** de un **único Nodo** y sirve para probar el funcionamiento y ejemplos limitados
Kubeadm	Consiste en un **administrador de Kubernetes** con **nodos del tipo Master-Worker** que pueden ser distribuidos en diferentes Hosts
KinD	"Kubernetes in Docker" está basado en crear un **ecosistema completo de Kubernetes** usando **contenedores de Docker como base** y así poder diseñar múltiples nodos en un único Host o PC personal para pruebas avanzadas **sin consumir muchos recursos** o servidores

Instalación de KinD (1)

```
%%% Install Docker %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

$ sudo snap install docker

$ sudo addgroup --system docker
$ sudo usermod -aG docker $USER

$ sudo snap disable docker
$ sudo snap enable docker

[LOGOUT]

$ docker version

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Instalación de KinD (2)

```
%%% Install Kubectl %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

$ sudo snap install kubectl --classic

$ kubectl version --output yaml

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Instalación de KinD (3)

```
%%% Install Go %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

$ sudo snap install go --classic

$ echo "GOPATH=$HOME/go" >> ~/.bashrc
$ echo "export GOPATH" >> ~/.bashrc
$ echo "PATH=\$PATH:\$GOPATH/bin" >> ~/.bashrc

$ source ~/.bashrc

$ go version

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Instalación de KinD (4)

%%% Install KinD %%%

$ go install **sigs.k8s.io/kind@latest**

$ **kind version**

%%

Creación de un clúster KinD (1)

```
%%% Create KinD cluster %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

$ kind create cluster --name cluster101

$ kubectl cluster-info

--- Inspect KinD cluster -------------------------------------

$ kind get clusters

$ kubectl get nodes

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Creación de un clúster KinD (2)

```
%%% Open cluster in Docker %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

$ docker exec -it cluster101-control-plane bash

--- Inspect Pods ----------------------------------------

cluster101-control-plane# kubectl get pod --output wide

--- Create Nginx Pod ------------------------------------

cluster101-control-plane# kubectl run nginx-pod --image=nginx

--- Test Nginx Pod Works --------------------------------

curl <pod-ip>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Creación de un clúster KinD (3)

```
--- List Resources -----------------------------------------

$ kubectl api-resources

--- Inspect Pods -------------------------------------------

$ kubectl get pods --all-namespaces

--- Delete KinD cluster ------------------------------------

$ kind delete cluster -n <name>

------------------------------------------------------------
```

Revisando su clúster KinD

```
--- List Nodes ----------------------------------------

$ kubectl get nodes

$ kubectl get csinodes

--- Describe Nodes ------------------------------------

$ kubectl describe node <name>

$ kubectl describe csinode <name>

-------------------------------------------------------
```

Revisando su clúster KinD

```
--- List Storage Drivers ------------------------------------

$ kubectl get csidrivers

--- List Storage Classes ------------------------------------

$ kubectl get storageclasses

~ kubectl get sc

---------------------------------------------------------------
```

Revisando su clúster KinD

```
--- List Persisten Volume & Claim ----------------------------

$ kubectl get pv

$ kubectl get pvc

---------------------------------------------------------------
```

Trabajo 301: Crear un PVC de prueba (1)

```
=== test-pvc.yaml ==========================================

apiVersion: v1
kind: PersistentVolumeClaim

metadata:
  name: test-claim

spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Mi

============================================================
```

Trabajo 301: Crear un PVC de prueba (2)

```
=== test-pvc-pod.yaml ========================================

apiVersion: v1
kind: Pod

metadata:
  name: test-pvc-pod

spec:
  containers:
    ...
  volumes:
    ...
  restartPolicy: "Never"

========================================
```

```
////////////////////////////////////////////////////////////

  containers:
    - name: test-pod
      image: busybox
      command:
        - "/bin/sh"
      args:
        - "-c"
        - "touch /mnt/test && exit 0 || exit 1"
      volumeMounts:
        - name: test-pvc
          mountPath: "/mnt"
  volumes:
    - name: test-pvc
      persistentVolumeClaim:
        claimName: test-pvc

////////////////////////////////////////////////////////////
```

Trabajo 301: Crear un PVC de prueba (3)

```
--- Run PVC ------------------------------------------------

$ kubectl create -f test-pvc.yaml

--- Inspect PVC --------------------------------------------

$ kubectl get pvc

$ kubectl describe pvc test-pvc

------------------------------------------------------------
```

Trabajo 301: Crear un PVC de prueba (4)

```
--- Run Pod claims PVC ---------------------------------------

$ kubectl create -f test-pvc-pod.yaml

--- Inspect Pod ----------------------------------------------

$ kubectl get pods

$ kubectl describe pod test-pvc-pod

--------------------------------------------------------------
```

Trabajo 301: Crear un PVC de prueba (5)

```
--- Inspect Pod and PV/C ------------------------------------

$ kubectl get po,pv,pvc

$ kubectl describe persistentvolumeclaim/test-pvc

$ kubectl describe persistentvolume/pvc-<<xxxx>>

$ kubectl describe pod/test-pvc-pod

--- Inspect Node Worker -------------------------------------

$ kubectl get pod test-pvc-pod --output wide

$ docker exec -it <<worker-name>> ls -l \
  /var/local-path-provisioner/pvc-<<xxxx>>_default_test-pvc

-------------------------------------------------------------
```

Agregar un balanceador de carga personalizado para Ingress

... Manos a la obra ...

https://github.com/dragonnomada/kubelab/blob/main/d301-kind-docker-cluster.txt