Kubernetes

Implantación de Aplicaciones Web

José Juan Sánchez Hernández

Curso 2023/2024

Índice

Índice de figuras

Índice de cuadros

1 Kubernetes

NOTA: El contenido de este sitio está en desarrollo.

1.1 ¿Qué es Kubernetes?

Kubernetes (K8s) es una plataforma de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones en contenedores.

1.2 ¿Qué es un clúster de Kubernetes?

Un **clúster** de Kubernetes es un grupo de máquinas físicas o virtuales donde se ha desplegado el software de Kubernetes, para que trabajen juntas y ejecuten aplicaciones en contenedores.

Dentro de un clúster de Kubernetes existen dos tipos de nodos:

- **Nodos Master** (*Control Plane*)
- **Nodos Workers**

1.3 Componentes de Kubernetes

Vamos a estudiar cuáles son los componentes que utilizan cada uno de los nodos de un clúster de Kubernetes:

1. **Componentes de los nodos Master** (*Control Plane*)

Componente	Descripción
`kube-apiserver`	Publica la API de Kubernetes y actualiza la base de datos `etcd`.
`etcd`	Es una base de datos distribuidad de tipo clave-valor donde se almacena la configuración y el estado del clúster.

Componente	Descripción
`kube-controller-manager`	Ejecuta los controladores que gestionan los recursos del clúster. Gestiona la creación de las réplicas para mantener el estado deseado en el clúster.
`kube-scheduler`	Decide en qué nodo del clúster se ejecutará cada Pod.
`cloud-controller-manager`	Interactua con proveedores externos de servicios en cloud computing como AWS, Azure o GCE. Este componente es opcional

2. **Componentes de los nodos Workers**

Componente	Descripción
`kubelet`	Ejecuta y gestiona los Pods en los nodos del clúster.
`kube-proxy`	Implementa un proxy de red para los Pods para facilitar la comunicación entre ellos.
`container-runtime`	Runtime que ejecuta los contenedores (Docker, containerd, CRI-O, etc.)

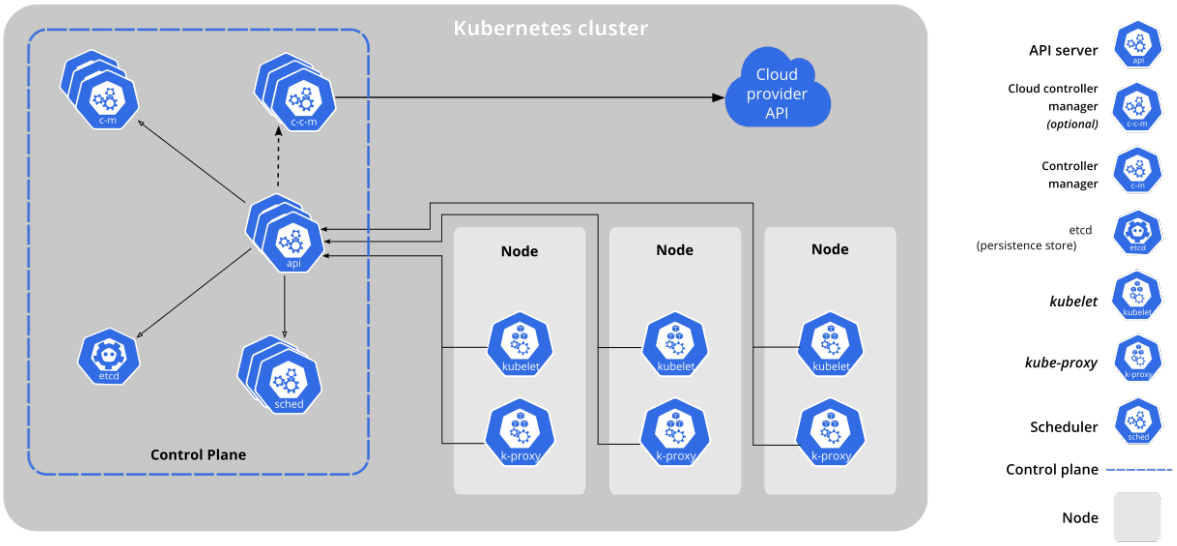

Imagen 1: Componenetes de un clúster de Kubernetes. Imagen obtenida de la página oficial de Kubernetes

1.4 ¿Cómo puedo crear un clúster de Kubernetes?

Para crear un clúster de Kubernetes existen diferentes soluciones:

1. **Utilizar Kubernetes como servicio en un proveedor de Cloud Computing.**

Una solución sencilla donde no es necesario realizar ninguna instalación, es hacer uso de los servicios de Kubernetes que ofrecen algunos proveedores de Cloud Computing como:

- Amazon Elastic Kubernetes Service (EKS)
- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)

2. **Realizar la instalación Kubernetes sobre una infraestructura determinada**

Existen diferentes proyectos open source que nos permiten realizar la instalación de un clúster de Kubernetes sobre la infraestructura que nosotros queramos. Algunas soluciones son:

- Kubeadm
- kOps
- Kubespray
- Rancher

También existen otros proyectos que nos permiten crear un clúster en un entorno de desarrollo:

- Minikube
- Kind
- K3s
- MicroK8s

1.5 Creación de un entorno de desarrollo con Minikube

En esta práctica vamos a utilizar minikube, que es una herramienta de código abierto que permite ejecutar un clúster de Kubernetes de forma local.

Dependiendo del driver utilizado, minikube puede ser desplegado en:

- **Máquina virtual**
- **Contenedor**
- **Bare-metal**

Puede encontrar más información sobre los **drivers disponibles en minikube** en la documentación oficial.

1.6 Instalación de `kubectl`

Para gestionar nuestro clúster de Kubernetes vamos a utilizar la herramienta de línea de comandos `kubectl`. Esta herramienta nos permite interaccionar con la API de Kubernetes.

1.7 Actividad 1. Instalación de `minikube` y `kubectl`

La primera actividad que vamos a realizar será la instalación de `minikube` y `kubectl` sobre una instancia de **OpenStack** o **AWS**.

El sistema operativo de la instancia tendrá que ser la última versión disponible de **Ubuntu Server** o **Debian**.

Tendrá que crear un clúster de Kubernetes con `minikube` con un solo nodo (Control Plane).

Para realizar la instalación de `minikube` y `kubectl` se recomienda utilizar los scripts del siguiente repositorio:

- https://github.com/josejuansanchez/minikube

En este repositorio puede encontrar los scripts para deplegar minikube en sistemas Debian y Ubuntu, utilizando los siguientes drivers:

- KVM2 - VM
- VirtualBox - VM
- Docker - contenedor.
- None - bare-metal

Se recomienda utilizar el **driver none** para realizar una instalación en **bare metal**.

1.8 Actividad 2. Creación de un clúster con varios nodos

En esta actividad tendrá que crear un clúster de Kubernetes con minikube con tres nodos:

- 1 Control Plane.
- 2 Nodos Workers.

1.9 Objetos de Kubernetes

En esta práctica vamos a trabajar con los siguientes objetos de Kubernetes:

- **Pod**
- **ReplicaSet**
- **Deployment**
- **Service**
- **Ingress**
- **Volumes**
- **ConfigMaps**
- **Secrets**
- **Namespace**

1.9.1 Pods

Un **Pod** es la unidad base de Kubernetes, es el recurso más pequeño que se puede desplegar en el clúster.

Un **Pod** puede tener un contenedor (principal) y varios contenedores auxiliares (*init containers*, *ephemeral containers* o *sidecar containers*).

- Contenedores de inicialización
- Contenedores efímeros
- Sidecars containers

Los contenedores que están dentro del mismo pod comparten la misma dirección IP y se comunican por `localhost`. Por este motivo, no pueden usar los mismos puertos.

En un cluster de Kubernetes podemos tener muchos **Pods** y cada uno tiene su propia dirección IP.

1.9.1.1 Definición de un Pod

En este ejemplo vamos a definir un objeto de tipo **Pod** en un archivo llamado `mi-pod.yaml` con el siguiente contenido.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mi-pod
  labels:
    mi-clave: mi-valor
spec:
  containers:
  - name: mi-contenedor
    image: nginx
    ports:
    - containerPort: 80
  restartPolicy: Always
```

1.9.1.2 Creación de un Pod

```
kubectl apply -f mi-pod.yaml
```

1.9.1.3 Obtener el listado de Pods

Para obtener el listado de Pods podemos utilizar cualquier de estas tres opciones:

```
kubectl get pods
kubectl get pod
kubectl get po
```

Podemos utilizar el flag `-o wide` para obtener información ampliada:

```
kubectl get pods -o wide
```

Para obtener un listado de todos los recursos podemos utilizar el comando:

```
kubectl get all
```

1.9.1.4 Obtener información detallada de un Pod

Podemos utilizar cualquiera de estas dos opciones.

```
kubectl describe pods mi-pod
kubectl describe pods/mi-pod
```

1.9.1.5 Mostrar los logs del contenedor del Pod

Si el Pod sólo tiene un contenedor no es necesario indicar el nombre, es opcional.

```
kubectl logs mi-pod
```

Con el flag `-f` podemos indicar que la información del log se mueste en tiempo real.

```
kubectl logs -f mi-pod
```

1.9.1.6 Ejectar un comando dentro del contenedor del Pod

El siguiente comando ejecuta el comando `/bin/bash` dentro del conenedor del pod `mi-pod`.

```
kubectl exec -it mi-pod -- /bin/bash
```

1.9.1.7 Cómo acceder al contenedor del Pod

Para acceder a contenedor del **Pod** podemos hacemos hacer una redirección de puertos desde la máquina donde estamos ejecutando el comando `kubectl` hasta el contenedor del Pod.

En este ejemplo vamos a hacer una redirección entre el puerto `8000` de la máquina donde ejecutamos `kubectl` y el puerto `80` del Pod.

```
kubectl port-forward mi-pod 8000:80
```

Al ejecutar el comando veremos que la redirección está activa y finalizará cuando terminemos la ejección del comando con `Control + C`.

```
Forwarding from 127.0.0.1:8000 -> 80
Forwarding from [::1]:8000 -> 80

^C
```

Para acceder al Pod lo haremos accediendo a través de la dirección de `localhost` de la máquina donde hemos ejecutado el comando.

```
curl http://localhost:8000
```

Si queremos hacer la redirección de puertos a una dirección IP que no sea la de `localhost` podemos utilizar el flag `--address`.

```
kubectl port-forward mi-pod 8000:80 --address 0.0.0.0
```

1.9.1.8 Eliminar un Pod

```
kubectl delete pods mi-pod
```

1.9.2 ReplicaSet

Un **ReplicaSet** es un objeto que se encarga de garantizar que siempre haya en ejecución un número específico de réplicas de un Pod dentro del clúster.

1.9.2.1 Definición de un ReplicaSet

En este ejemplo vamos a definir un objeto de tipo **ReplicaSet** en un archivo llamado `mi-replicaset.yaml` con el siguiente contenido.

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: mi-replicaset
spec:
  # Número de réplicas que queremos tener en ejecución
  replicas: 2
  # Selecciona todos los Pods que tengan la etiqueta 'app: mi-app'
  selector:
    matchLabels:
      app: mi-app
  # Define la plantilla del pod que se utilizará para crear las réplicas
  template:
    metadata:
      # Añade la etiqueta 'app: mi-app' al pod para que coincida con el selector
      labels:
        app: mi-app
    spec:
      containers:
      - name: mi-contenedor
        image: nginx
        ports:
        - containerPort: 80
      restartPolicy: Always
```

1.9.2.2 Creación de un ReplicaSet

```
kubectl apply -f mi-replicaset.yaml
```

1.9.2.3 Obtener el listado de ReplicasSet

Para obtener el listado de **ReplicaSet** podemos utilizar cualquier de estas opciones:

```
kubectl get replicasets.apps
kubectl get replicasets
kubectl get rs
```

Podemos utilizar el flag `-o wide` para obtener información ampliada:

```
kubectl get rs -o wide
```

Para obtener un listado de todos los recursos podemos utilizar el comando:

```
kubectl get all
```

1.9.2.4 Obtener información detallada de un ReplicaSet

Podemos utilizar cualquiera de estas opciones.

```
kubectl describe replicasets.apps mi-replicaset
kubectl describe replicasets.apps/mi-replicaset

kubectl describe replicasets mi-replicaset
kubectl describe replicasets/mi-replicaset

kubectl describe rs/mi-replicaset
kubectl describe rs mi-replicaset
```

1.9.2.5 Mostrar los logs del Deployment

```
kubectl logs replicasets mi-replicaset
```

Con el flag `-f` podemos indicar que la información del log se mueste en tiempo real.

```
kubectl logs -f replicasets mi-replicaset
```

1.9.2.6 Eliminar un ReplicaSet

```
kubectl delete replicasets mi-replicaset
```

1.9.3 Deployment

El **Deployment** es un objeto que nos permite automatizar el depspliegue y la actualización de una aplicación dentro del clúster. Este objeto utiliza en su definición los objetos **Pod** y **ReplicaSet**.

Las buenas prácticas recomiendan trabajar con **Deployments** en lugar de **Pods**.

1.9.3.1 Definición de un Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mi-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: mi-app
  # Estrategia de actualización: RollingUpdate
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mi-app
    spec:
      containers:
      - name: mi-contenedor
        image: nginx
        ports:
        - containerPort: 80
      restartPolicy: Always
```

1.9.3.2 Estrategia de actualización

Cuando se realiza una actualización sobre un **Deployment** que está en ejecución, es posible indicar cuál será la estrategia que se va a utilizar para reemplazar los Pods de la versión antigua por los Pods de la nueva versión.

Esta estrategia se indica en el parámetro `.spec.strategy` y se pueden indicar dos tipos:

- `Recreate`. Esta estrategia elimina todos los Pods existentes y una vez eliminados crea los Pods con la nueva versión. Tiene el inconveniente que durante la actualización hay un tiemo donde se deja de dar servicio hasta que se hayan creado los nuevos Pods.
- `RollingUpdate`. Es la opción por defecto. En este caso, se van creando los Pods con la nueva versión y de forma progresiva se van eliminando los Pods con la versión antigua.

1.9.3.3 Creación de un Deployment

```
kubectl apply -f mi-deployment.yaml
```

1.9.3.4 Obtener el listado de Deployments

Para obtener el listado de **Deployments** podemos utilizar cualquier de estas opciones:

```
kubectl get deployments.apps
kubectl get deployments
kubectl get deploy
```

Podemos utilizar el flag `-o wide` para obtener información ampliada:

```
kubectl get deploy -o wide
```

Para obtener un listado de todos los recursos podemos utilizar el comando:

```
kubectl get all
```

1.9.3.5 Obtener información detallada de un Deployment

Podemos utilizar cualquiera de estas opciones.

```
kubectl describe deployment.apps mi-deployment
kubectl describe deployment.apps/mi-deployment

kubectl describe deployment mi-deployment
kubectl describe deployment/mi-deployment

kubectl describe deploy/mi-deployment
kubectl describe deploy mi-deployment
```

1.9.3.6 Mostrar los logs del Deployment

```
kubectl logs deployments/mi-deployment
```

Con el flag `-f` podemos indicar que la información del log se mueste en tiempo real.

```
kubectl logs -f deployments/mi-deployment
```

1.9.3.7 Eliminar un Deployment

```
kubectl delete deployment mi-deployment
```

2 Referencias

- Documentación oficial de Kubernetes.
- Introducción a Kubernetes. José Domingo Muñoz Rodríguez.
- Introducción a Kubernetes. Alberto Molina Coballes y José Domingo Muñoz Rodríguez.
- Vídeos del curso de Introducción a Kubernetes. Alberto Molina Coballes y José Domingo Muñoz Rodríguez.
- Kubernetes 101. José Joaquín Cañadas y Manuel Torres.
- Kubernetes. Un orquestador de contenedores que debes poner en tu vida. Manuel Torres.
- Vídeos de Kubernetes en español. Bitnami.
- Vídeos. AWS Academy - Docker and K8s. Santos Pardos.
- Certified Kubernetes Administrator - CKA Course
- A visual guide on troubleshooting Kubernetes deployments. Daniele Polencic - Learnk8s.
- A visual guide on troubleshooting Kubernetes deployments (PDF)

3 Licencia