Capítulo 3
Panel de control y línea de comandos

1. Objetivos del capítulo y requisitos previos

En primer lugar, este capítulo se ocupará de abordar brevemente la historia del origen del proyecto y las nociones relacionadas con los contenedores.

Posteriormente, se presentará una introducción al uso de Kubernetes y se descubrirán algunos elementos:

– El panel de control (dashboard) de Kubernetes.

– El comando kubectl.

– El motor Docker de Minikube.

– La consulta de algunos objetos básicos (pods y nodos).

Al final de este capítulo, dispondrá de los siguientes elementos:

– Un clúster de Kubernetes en funcionamiento.

– Un contexto kubectl coherente.

Para poder seguir todos los ejercicios, es imprescindible disponer de una conexión a Internet de banda ancha (ADSL o superior).

2. Preámbulo

2.1 Origen del nombre y del logo

Kubernetes es una palabra de origen griego (κυβερνήτης) que significa timonel (o piloto). En la navegación, es la persona que se encarga de sujetar el timón.

En Google, el nombre interno de este proyecto fue Project Seven, en referencia a un personaje del universo de Star Trek (Seven of Nine o su nombre real, Annika Hansen). Los siete radios del logotipo del timón son un guiño a este antiguo nombre.

2.2 ¿Por qué utilizar Kubernetes?

El lanzamiento de Docker en los últimos años ha supuesto un gran cambio en la forma de gestionar aplicaciones dentro de un sistema de información. De este modo, ahora es posible realizar operaciones que antes eran complicadas, de una manera más rápida y sencilla; tareas como:

– iniciar entornos de prueba bajo demanda,
– deshacerse de los problemas de dependencias de las librerías,
– estandarizar los entregables de todos los entornos,
– aislar los recursos en el mismo servidor.

Como cualquier herramienta nueva, algunas preguntas tenían forzosamente una solución sencilla y se introdujeron nuevos problemas. Entre estos problemas, se encuentran los siguientes:

– Persistencia de datos.
– Seguimiento de aplicaciones en los contenedores.
– Actualizaciones automáticas de aplicaciones.
– Escalabilidad del motor Docker.
– Escalabilidad de la carga de trabajo de la aplicación.
– Publicación de entornos al exterior.

Estas y otras fueron las razones por las que se introdujo Kubernetes en 2014, para abordar los problemas de escalabilidad de Docker.

2.3 Origen de Kubernetes

Las primeras versiones de Kubernetes provienen de Google. La empresa detrás del motor de búsqueda ha estado trabajando en los conceptos de «contenedorización» durante más de diez años. Este trabajo ha llevado a Google a hacer muchas contribuciones a la comunidad sobre el kernel de Linux (Google es uno de los cinco contribuidores más importantes).

Parte de este trabajo también ha ayudado a la aparición de Docker, como Cgroups, por ejemplo. Este mecanismo, originalmente desarrollado por ingenieros de Google, permite establecer un límite de consumo de CPU o memoria, así como aislar procesos.

En Google, estos antepasados de los contenedores contemporáneos, fueron impulsados internamente por un producto de la casa: Borg. Este producto sigue activo en Google y gestiona a diario miles de máquinas en todo el mundo.

Observe que algunos de los desarrolladores de Kubernetes son antiguos colaboradores del proyecto Borg. Estos últimos se han apropiado de gran parte de los principios presentes en Borg para aplicarlos en Kubernetes.

2.4 Fundación CNCF

Google donó el código del producto a la Linux Foundation, lo que sirvió para crear la CNCF (*Cloud Native Computing Foundation*). Esta organización está detrás de la promoción de productos open source en la nube, como Prometheus, OpenTracing o Containerd.

La idea de Kubernetes es mantener lo que funciona bien en Docker (el principio de los contenedores Dockers) y extender el resto con una capa de API, además de un mecanismo de extensión y orquestación.

2.5 Las orquestadores del mercado

La empresa detrás de Docker ha desarrollado su propio motor de orquestación: Docker Swarm. Este libro no hablará de ello, pero este producto también da respuesta a algunos de los problemas que se han menciona anteriormente. Cabe señalar que las últimas evoluciones de esta herramienta permiten comunicarse con Kubernetes.

Además de Docker Swarm, existen otros orquestadores, como Nomad o Mesos, que utilizan mecanismos completamente diferentes a los de Kubernetes.

Observación

La integración de Docker Shim que permite que Kubernetes se comunique con Docker es difícil de mantener, aunque sea posible comunicarse directamente con el motor del contenedor subyacente (Containerd). La pérdida de valor proviene directamente de esta razón.

También hay superposiciones para Kubernetes, que añaden funcionalidad o soporte comercial. En particular, encontramos los siguientes productos:

– Openshift, de RedHat
– Tectonic, de CoreOS
– Rancher, de Rancher Labs

Todos estos productos se basan en una versión de Kubernetes, en la que el editor proporcionará los siguientes servicios:

– Preconfigurar servicios listos para usar.
– Congelar un conjunto de versiones para ofrecer soporte a largo plazo.

Afortunadamente, estas diferencias suelen ser marginales. No debería tener ningún problema con ellas respecto a las instrucciones que aparecen en este libro.

3. El panel de control de Kubernetes (dashboard)

3.1 Presentación

El panel de control de Kubernetes es una forma de consultar el estado de los diferentes elementos de un clúster. Para un usuario que comienza en esta tecnología, se trata un buen punto de partida, ya que esta aplicación permite consultar gráficamente los diferentes componentes de un clúster de Kubernetes.

Sin embargo, aunque esto último es posible en una máquina Minikube, por razones de seguridad puede no ser necesariamente buena idea desplegarlo en plataformas de producción.

3.2 Panel de control Kubernetes en servicio gestionado

Si utiliza un servicio gestionado, es posible que sea necesario instalar el dashboard antes de continuar.

Para instalarlo, consulte la sección Desplegar el panel de control de Kubernetes, en los anexos.

3.3 Desplegar el dashboard en Minikube

En Minikube, la activación del panel de control se realiza con la instrucción `minikube`, seguida de estas opciones:

– La opción `addons`, seguida de la palabra clave `enable`.
– El nombre del plugin que se desea activar (aquí, `dashboard`).

A continuación, se muestra el comando que se debe ejecutar:

```
$ minikube addons enable dashboard
```

Seguidamente, se muestra el resultado de este comando:

```
    Using image kubernetesui/dashboard:v2.3.1
    Using image kubernetesui/metrics-scraper:v1.0.7
    Some dashboard features require the metrics-server addon. To
enable all features please run:
        minikube addons enable metrics-server

    The 'dashboard' addon is enabled
```

Como indica el comando anterior, active el servidor de métricas (`metrics-server`) para poder utilizar todas las funcionalidades del panel de control:

```
$ minikube addons enable metrics-server
```

A continuación, se muestran los mensajes que devuelve esta operación:

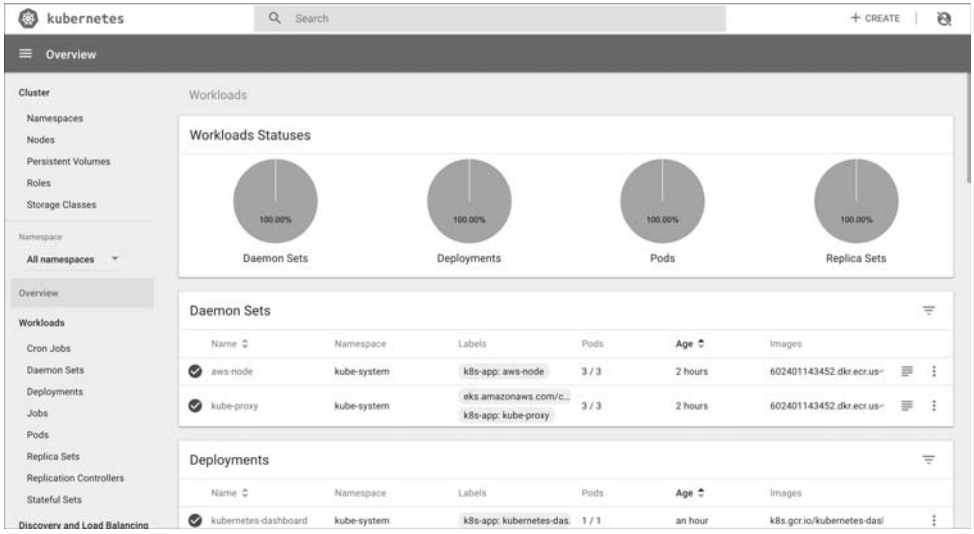
```
    Using image k8s.gcr.io/metrics-server/metrics-server:v0.4.2
    The 'metrics-server' addon is enabled
```

3.4 Acceder al dashboard en Minikube

El acceso al panel de control se puede realizar con el comando `minikube`, seguido de la opción `dashboard`.

A continuación, se muestra el comando correspondiente:

```
$ minikube dashboard
```



Página de bienvenida del panel de control de Kubernetes

El comando se encargará de publicar el dashboard en un pipeline seguro e iniciar el navegador predeterminado.

3.5 Estructura del panel de control

La parte izquierda contiene enlaces a los siguientes elementos:

- Elementos desplegados (**Workloads**).
- Las entradas de servicio interno y externo (**Services**, **Ingresses**).
- Los elementos de configuración (**Config and Storage**).
- Los elementos del clúster (**Namespaces**, **Nodes**, **Persistent Volumes**, etc.).
- Las definiciones de objetos personalizados (**Custom Resource Definitions**).