Kubernetes

Víctor Iranzo

28 de julio de 2018

1. Introducción

Los contenedores son ligeros, pero no son máquinas virtuales y necesitan un gran esfuerzo para ser orquestados y ejecutarse de forma eficiente y resilente. Están diseñados para ser frágiles y tener una vida muy corta. Un sistema estable se fundamenta en el hecho de que cualquiera de sus componentes puede fallar. Los contenedores no son una excepción. El aislamiento entre estos componentes también es importante: el fallo de un contenedor no debe provocar la caída del sistema al completo.

Un servicio está diseñado para hacer un pequeño número de cosas. No tiene interfaz y es invocado a través de una API. Una aplicación tiene una interfaz de usuario y ofrece gran cantidad de funcionalidades distintas. Kubernetes administra servicios, no aplicaciones. Cuanto más pequeños sean y más claro sea que solo tienen una única responsabilidad, más fácil será su gestión.

2. Términos en Kubernetes

Cada una de las máquinas que aloja uno o más contenedores Docker se denomina nodo. Otras máquinas ejecutan un software especial para coordinar a los contenedores en los otros nodos. Estas máquinas se denominan maestros. Una colección de maestros y nodos se denomina clúster.

El maestro ejecuta 3 elementos principalmente: el servidor de la API, que gestiona las invocaciones hechas a través de la API al clúster, Etcd, un servicio que se encarga de replicar la configuración y estado del clúster, y el planificador, que administra los contenedores en los nodos y se encarga de que el número correcto de instancias se ejecuten en todo momento.

Un nodo ejecuta 3 elementos principalmente: Kubelet, un cliente que se ejecuta en el nodo y monitoriza el estado de sus contenedores y responde a los comandos del servidor, el proxy, que separa la dirección IP del contenedor del nombre provisto al servicio, y el cAdvisor, una herramienta que almacena información sobre los estados en ejecución.

Un pod es una colección de contendores y volúmenes agrupados juntos porque comparten un recurso. Mientras que Docker asigna a cada contenedor su propia dirección IP, Kubernetes asigna una dirección IP compartida en todo el pod. Se puede entender como una VM que contiene contenedores. Kubernetes orquesta las cosas al nivel de los pods y no de sus contenedores, por lo que dos contenedores en el mismo pod serán administrados de forma equitativa.

Los pods no son elementos duraderos: un maestro puede decidir eliminar un pod y volverlo a crear en cualquier momento. Preservar el estado de la aplicación se debe externalizar a través de una base de datos o un volumen.

En Docker, un volumen es un sistema de archivos virtual que el contenedor puede ver y usar. Kubernetes también tiene esta característica pero al nivel de los pods. Los volúmenes son más duraderos que la vida de los contenedores y pueden ser empleados por cualquier contenedor del pod, por lo que pueden ser un mecanismo de comunicación entre ellos.

Existen 3 tipos de volúmenes: EmptyDir, inicialmente vacío y efímero porque solo sobrevive existe durante la vida del pod, NFS, que si existen de forma persistente tras la vida de los pods, y GCP (Google Cloud Platform), también persistente, se puede entender como un NFS gestionado en el que solo nos tenemos que preocupar por los datos.

3. Etiquetas y anotaciones

Kubernetes provee de dos sistemas para documentar la infraestructura:

- Etiquetes: una tupla clave valor que se asigna a un pod en el archivo .yaml de este ("lenguaje":"java"). Las etiquetas siguen las mismas reglas que las entradas DNS, no pudiendo contener espacios ni carácteres inválidos y teniendo una longitud máxima. Se pueden hacer consultas sobre las etiquetas que contiene un pod a través de selectores de etiquetas.

  Existen dos tipos de selectores de etiquetas: los basados en igualdades (equality-based) y en conjuntos (set-based). Los primeros nos permiten

obtener los pods con una etiqueta con un determinado valor (lenguaje = java). Los segundos nos permiten obtener los pods cuyo valor para un etiqueta está dentro de un conjunto de valores especificados (lenguaje in (java, javascript)).

- Anotaciones: contienen información útil que se desea almacenar de un pod y sobre la que no se quiere hacer consultas.

4. Controlador de la repliación

Las aplicaciones complejas necesitan manejar mucha carga de trabajo y deben asegurar la disponibilidad. Es necesario por tanto tener más de un pod ejecutando al mismo tiempo. Las replicas son las diferentes copias de un pod. El proceso que gestiona las replicas es el controlador de replicación (controller replication), que se asegura de que el número correcto de replicas se ejecute en todo momentos. La manera en que hace esto es mediante reglas definidas en la plantilla del pod, un archivo .yaml donde se especifica que imagen se debe usar, el número de replicas necesario, etc.

El esquema de replicación está diseñado para ser poco acoplado. No se define que pods debe controlar, sino que el controlador maneja aquellos pods que se ajusten a un selector de etiquetas. Se debe tener en cuenta que estos controladores también son susceptibles de morir.

Los controladores son buenos para asegurar la disponiblidad de un servicio, pero garantizan la escalabilidad. Para garantizar la escalabilidad se tiene que modificar el tamaño de un clúster de forma dinámica en función de su demanda. Con este propósito, se debe situar un balanceador de carga antes del clúster que modifique el número de replicas para crear nuevos en función de la demanda que observe.

Es siempre aconsejable tener un controlador para garantizar el número correcto de replicas de un pod aunque solo exista una unidad. Otro uso conocido de los controladores es actualizar la versión de un pod a través de reglas que sustituyen progresivamente una replica antigua por una actualizada.

5. Servicios

Un balanceador de carga situado frente a nuestro clúster no puede conocer donde redirigir las peticiones que recibe por si solo porque los pods son destruídos y recreados continuamente con otras direcciones IP. Kubernetes

soluciona este problema mediante los servicios.

Los servicios son componentes duraderos que apuntan a una serie de pods dentro del cluster. Consta de tres partes: una dirección IP externa, un puerto y un selector de etiquetas. El servicio se expone a través de un proxy al que se le deben dirigir las peticiones para que sea el servicio quien redirija estas hacia el pod indicado, que conoce a través del selector de etiquetas.

Cuando un cliente realiza una petición al sistema, esta es dirigida a un balanceador de carga externo, que la dirige a un servicio a través de su IP externa, que la redirigirá al pod adecuado. Si existe más de un pod que pueda atender la petición, el servicio sigue una política de round-robin.

A la hora de diseñar el sistema hay que tener en cuenta que el pod que gestiona una petición es probable que no gestione la siguiente: puede ser destruido y regenerado en cuestión de segundos.

6. Integración de servicios

En lugar de emplear selectores de etiquetas, un servicio puede redirigir hacia una dirección IP estática. Cuando tengamos que integrar nuestros pods con un sistema externo legado, en lugar de especificar la IP del sistema legado en cada uno de los pods, podemos definir un servicio que nos redirija a él. Así, si se cambia la IP del sistema legado no se tienen que desplegar de nuevo todos los pods, solo cambiar la configuración del servicio.

Cuando un pod debe consumir un servicio se pueden emplear dos mecanismos: variables de entorno y DNSs. El primer mecanismo consiste en variables a nivel de un nodo sobre los servicios asociados a un pod del mismo nodo. El segundo mecanismo consiste en un clúster DNS donde se registran los servicios disponibles tal y como se registran en un maestro.