Un poco de Java y +

Otra forma de hablar de nuestro día a día…

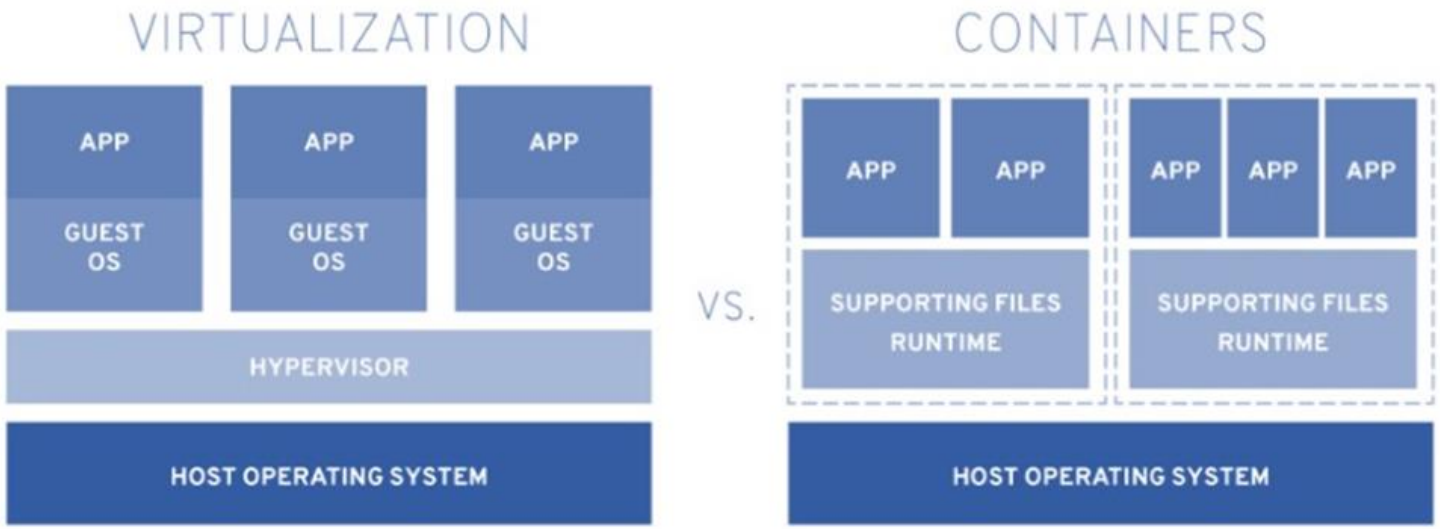
7 ENERO 2020
LUISMI GRACIA

¿Qué es Kubernetes?

(https://unpocodejava.files.wordpress.com/2020/01/image002.png)

A estas alturas, resulta difícil encontrar a alguien que no sepa lo que es un **Contenedor,** al igual que encontrar una organización que no esté ejecutando **aplicaciones contenerizadas….**(según Gartner en 2022 más del 75% de las organizaciones estarán ejecutando aplicaciones contenerizadas en Producción).

Podríamos definir un Contenedor como un conjunto de uno o más procesos aislados del resto del sistema; todos los ficheros necesarios para ejecutar un contenedor se proveen desde una imagen, de modo que estas imágenes son portables entre entornos. (podéis leer algo más sobre las ventajas de los contenedores en este post (https://unpocodejava.com/2018/05/16/beneficios-de-usar-contenedores-y-docker/))



(https://unpocodejava.files.wordpress.com/2020/01/image004.jpg)

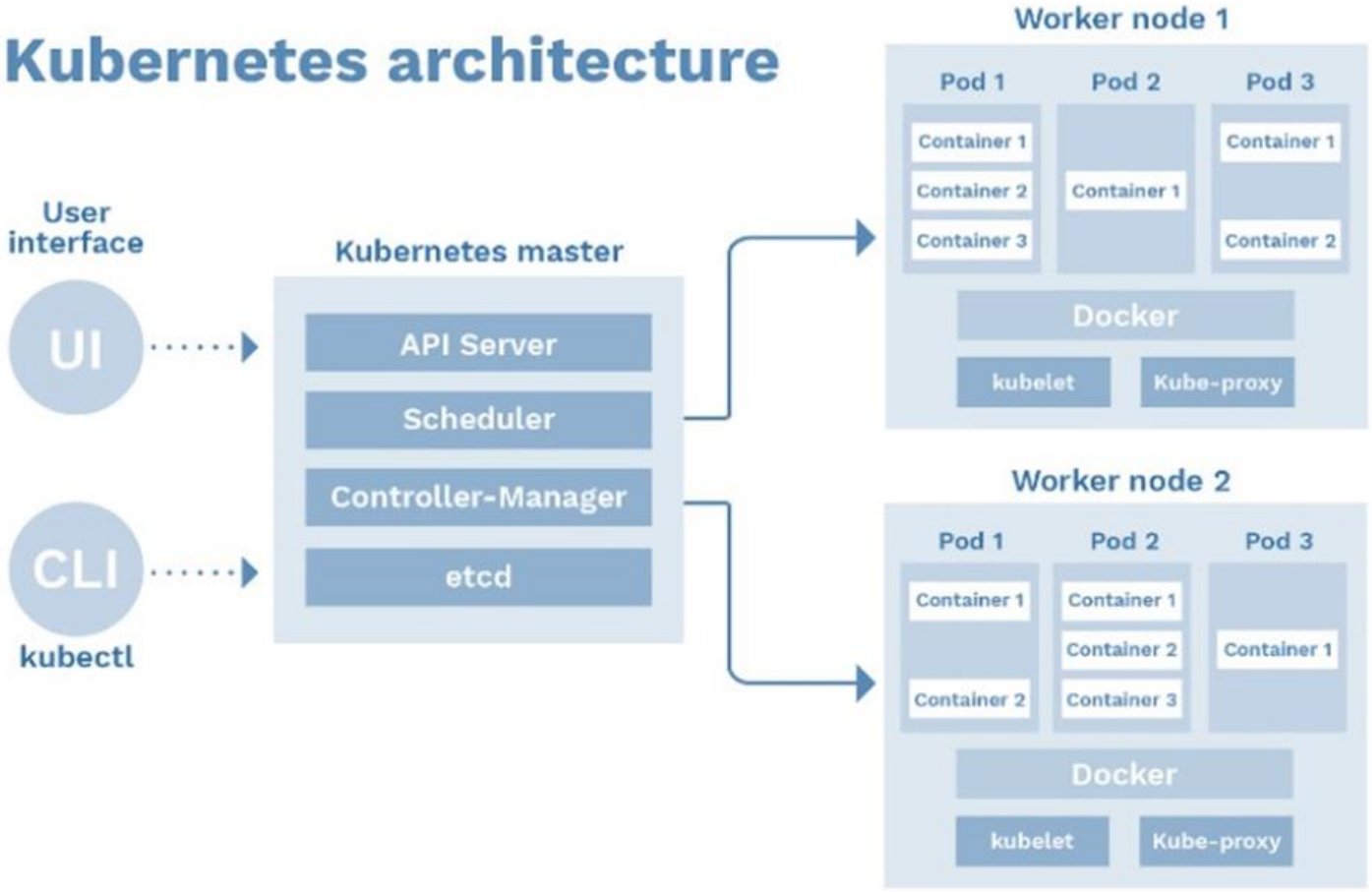
Pero una vez que una organización empieza a usar contenedores y empieza a operarlos se encuentra con que los contenedores deben comunicar entre ellos (y más en una arquitectura de microservicios)…lo que nos lleva al problema de **¿cómo gestionar todos esos contenedores? ¿cómo despliegue estos contenedores? ¿cómo gestiono la comunicación entre ellos?¿y la escalabilidad?**

Aquí aparece **Kubernetes como un sistema de orquestación de contenedores,** una herramienta para gestionar el ciclo de vida de las aplicaciones contenerizadas., un proceso que permite automatizar el despliegue y escalado de contenedores.

NOTA: Kubernetes no es la única herramienta para orquestar contenedores pero hay que reconocer que se ha convertido en el estándar de facto y ha ido eliminando a casi toda la competencia 😉.

ARQUITECTURA DE KUBERNETES

Como vais a ver Kubernetes introduce una gran cantidad de vocabulario para definir sus conceptos….veamos los principales:



(https://unpocodejava.files.wordpress.com/2020/01/image006.jpg)

TERMINOLOGÍA KUBERNETES:

PODS:

Un POD en Kubernetes es un grupo de contenedores, y además **es la unidad más pequeña que administra Kubernetes.**

- Los PODS tienen una IP única que se aplica a cada contenedores en el POD
- Los contenedores en un POD comparten los recursos (memoria y almacenamiento) lo que permite que los contenedores individuales en un POD puedan tratarse como una aplicación única (como si todos los procesos estuvieran ejecutando juntos en el mismo host)
- Es común tener un POD con un único contenedor, pero puede escalarse cuando se necesita ampliar la capacidad.

DEPLOYMENTS:

Los deployments de Kubernetes definen los detalles de despliegue de los PODS en los nodos de Kubernetes.

Describen el número de réplicas de PODS a ejecutar y la estrategia de actualización, de modo que Kubernetes se encarga de revisar la salud del POD y añadir o eliminar PODs conforme se necesiten.

SERVICES:

En Kubernetes no debemos confiar en la vida de un contenedor, todo desde su IP hasta su existencia puede cambiar. Por eso se dice que se trata los PODs como “cattle” (ganado) y no como un “pet”.

Kubernetes no trata sus PODs como instancias únicas o de larga duración y si se encuentra un problema y este mueve, Kubernetes se encarga de reemplazarlo sin que la aplicación sufra un downtime.

Un servicio es una abstracción sobre los PODS, y esencialmente es el único interface con el que los consumidores interactúan con el POD. Como hemos dicho, los PODS pueden reemplazarse y sus nombres internos e IPs pueden cambiar. Un **Service expone un único nombre e IP mapeada a los PODs.** El Service se encarga de que de cara a la red exterior todo siga igual.

NODES:

Un nodo Kubernetes gestiona y ejecuta PODS. Es la máquina (virtual o física) que ejecuta las tareas. Un node recolecta PODs enteros que funcionan juntos.

MASTER SERVER

Es el punto de entrada principal para los administradores y usuarios que gestionan los nodos. Las operaciones se lanzar a través de llamadas HTPP o bien a través de un CLI.

CLUSTER

Un cluster representa todos los componentes comentados (PODS, NODES, SERVICES; DEPLOYMENTS) juntos como una unidad.

COMPONENTES KUBERNETES

Una vez entendida la idea principal de cómo funciona Kubernetes podemos ver cómo los componentes de Kubernetes permiten que esto funcione bien.

Existen 2 tipos de componentes: **Componentes del Master Server y Componentes del Worker Node**

COMPONENTES DEL MASTER SERVER

- **API Server:** expone un interface REST al cluster Kubernetes. Todas las operaciones contra los pods, servicios,…se ejecutan programáticamente comunicando con los endpoints que da.
- **Scheduler** es el responsable de asignar los trabajos a los nodos. Revisa la capacidad de los recursos y asegura que el rendimiento de los nodos worker cumplen con lo separado.
- **Controller-Manager** es el responsable de asegurar que el estado del cluster está operativo.
- **etcd:** es un almacenamiento Key-Value distribuido que usa Kubernetes para compartir información sobre el estado de un cluster, además de para almacenar la configuración global.

COMPONENTES DEL WORKER NODE

- **Kubelet** traquea el estado de un POD para asegurar que todos los contenedores están ejecutando. Da un heartbeat al Master Server cada X segundos. Si el Replication Controller no recibe el heartbeat el nodo se marca como unhealthy.
- **Kube Proxy:** rutea el tráfico de entrada a un nodo del servicio. Rutea las peticiones a los contenedores correctos.

ARQUITECTURA **CLOUD** **CONTENEDORES** **DOCKER** **KUBERNETES** **QUÉ ES UN POCO DE**

Un comentario

1. **LuisMi Gracia** dice:
   22 ABRIL 2020 DE 11:02
   Reblogueó esto en Un poco de Java y +.

RESPONDER