Containers

Camila Ferreyra, Gastón Martínez Castro,
Sebastián Rodríguez

Instituto Politécnico Superior Gral. San Martín

Junio 2019

Índice I

Índice II

Sección I

Sección II

5 Kubernetes

Aclaración

A lo largo de esta presentación haremos referencia a características específicas a Docker, porque este es el software más popular en cuanto a manipulación de Containers.

Qué es un Container?

- Es una unidad de software que contiene un programa y todas las dependencias necesarias para su funcionamiento.
- Introduce una abstracción que garantiza el funcionamiento del programa independientemente de las características del ambiente en el que se encuentre el container.
- Aisla al programa del entorno en que se ejecuta.

Qué es Docker?

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de containers.

Utiliza características de aislamiento de recursos del kernel Linux, como `cgroups` y espacios de nombres (`namespaces`) para permitir que containers independientes se ejecuten en una sola instancia de Linux, evitando el coste de iniciar y mantener máquinas virtuales.

Container vs Máquina Virtual

Maquina Virtual

- Para aislar n procesos se necesitan n SO operativos virtualizados.
- Virtualizaciones controladas por `Hypervisor`.
- Gran consumo de recursos.
- SO virtualizado puede contener bibliotecas innecesarias.

Container

- Cada container se ejecuta sobre un único Kernel.
- Ahorra recursos
- Containers controlados, en el caso de Docker, por `Docker Engine`.
- Cada containers contiene las dependencias minimas y necesarias

Sección I

Qué es una Imagen de Container?

Básicamente un Container es la instancia de su Imagen.
En la imagen del Container se almacena la configuración del mismo.
En esta configuración se establece el proceso a ejecutar y sus dependencias necesarias, debiéndose especificar las versiones de las mismas.
Esta configuración se almacena en una estructura de árbol.
A partir de una Imagen se puede crear un Contenedor y viceversa, podemos crear y configurar un Contenedor manualmente y luego generar una Imagen.

Que es un DockerFile?

Se denomina DockerFile al archivo en que se almacena la configuracion de una Imagen de Container en Docker.
Luego este archivo se construye en una imagen y se agrega a la lista de imagenes disponibles de Docker para luego poder instanciar un container.

Sección I

Sección II

5 Kubernetes

Un volumen es una manera de perpetuar los cambios realizados tanto desde el lado del host como del container.

Esto quiere decir que los cambios efectuados en un volumen desde el host se ven reflejados en la imagen del container sin necesidad de volverla a crear y permitiendo al container modificar los archivos en un directorio de manera directa

Un Bind Mount monta una carpeta desde el sistema de archivos local directamente al container.
De esta forma el motor de Docker no puede afectarlo ni reconocerlo directamente.

Ademas los cambios que se realicen a la carpeta desde el container no se van a ver reflejados en el sistema de archivos del Host ya que los Bind Mount solo comparten los contenidos de una carpeta.

Volume

- Posibilidad del container de modificar directamente los contenidos del volumen.
- El volumen es creado dentro de una carpeta manejada por el Docker Engine.
- Accesible desde consola.

Bind Mount

- Consiste en montar un archivo o directorio perteneciente al host, dentro de un container. Este archivo se referencia por su path.
- Buen rendimiento.
- Crea dependencias por fuera del Container y del Docker Engine, que no es lo ideal.
- No puede utilizarse desde la consola de Docker.
- Rapido despliegue.

Sección I

Sección II

- Red None

5 Kubernetes

Introducción

Como ya sabemos, la idea fundamental de los Containers, es aislar software. Sin embargo, en muchas aplicaciones es necesaria una forma de transmitir información entre Containers, para esto, se crean mecanismos de conexión entre Containers.
Cabe aclarar que en las siguientes diapositivas se haga referencia a mecanicas específicas a Docker, al ser este el software de Containers en el que nos concentramos.

Como se conectan los Containers?

En la comunicacion entre Containers lo que se hace básicamente es crear una red en la que se expone cada Container como un host. Esto implica que a cada container se le asigna su propia direccion IP y MAC, entre otros parametros.

Redes: Bridge

Por defecto todos los containers se conectan a un bridge virtual de Ethernet, en `Docker` esta representado por la red `docker0`. Existen otros 2 tipos de redes por defecto en `Docker`, la red `host` y la red `none`

La red `host` añade el Container a la pila de protocolos del Host en el que se ejecuta. Esto significa que no existe ningun tipo de aislamiento entre el Container y el Host, en cuanto a redes se refiere.

Esto implica una correspondencia entre los puertos del Container y el Host.

Por ejemplo, cualquier servicio ejecutándose en un puerto X del Container, es accesible desde el puerto X del Host.

La red `none` agrega al Container a una pila de Protocolos específica a dicho Container. En este caso, el Container no tiene una interfaz de red.

Sección I

Sección II

- Confiabilidad
- Balance de Carga
- Escalado

Que es Kubernetes?

Kubernetes es una aplicacion que automatiza las tareas manuales relacionadas al despliegue y escabilidad de una aplicacion.

A demas Kubernetes puede administrar multiples host balanceando los niveles de carga entre los hosts y los niveles de demanda sobre los containers lanzados en estos.

Como funciona Kubernetes?

Para funcionar Kubernetes cuenta con una unidad minima llamada Pod.

Un Pod es una abstraccion que permite incluir varios contenedores dentro del mismo elemento y administrar sus requerimientos.

De esta forma, se generan conjuntos de containers autonomos que comparten un Namespace, espacio de IP, etc.

Escabilidad y Confiabilidad

Una de las características estrella de Kubernetes es su concepto de replicas.

Precisamente a esto apunta la abstraccion de Containers en Pods; que luego se convertiran en multiples instancias o `replicas` de una misma aplicacion o servicio.
La utilidad de las replicas esta en su aplicacion para Confiabilidad, Balance de Carga y Escalado.

Confiabilidad

Al tener varias replicas de una aplicacion o servicio, se garantiza el producto final con mayor seguridad, aun si uno o mas de las replicas falla.

Esta es una características muy util en aplicaciones como cloud computing, que estan basadas en la teoria de que uno o mas componentes pueden fallar.

Balance de Carga

Al tener multiples replicas de un container podemos facilmente distribuir la carga de trabajo efectuada por los usuarios entre las distintas replicas.

Esta características es acompañada a la vez con la posibilidad de utilizar multiples hosts, haciendo asi un mejor uso del hardware disponible.

Escalado

Cuando los niveles de carga se elevan demasiado para la cantidad de replicas existentes, Kubernetes permite facilmente escalar la aplicacion añadiendo tantas replicas como se requieran para el trabajo.