TALLER

Docker, Kubernetes y OpenShift

Hola!

Soy José Domingo Muñoz

@pledin_jd

www.josedomingo.org

1. Docker

Contenedores

Los **contenedores** son un tipo de partición aislada dentro de un solo sistema operativo. Ofrecen muchos de los mismos beneficios que las máquinas virtuales, como seguridad, almacenamiento y aislamiento de redes, pero requieren muchos menos recursos de hardware y son más rápidos de iniciar y finalizar. También aíslan las librerías y el entorno de tiempo de ejecución (como CPU y almacenamiento) utilizados por una aplicación para minimizar el impacto de una actualización de SO en el SO del host.

Ventajas de uso de los contenedores

- Aislamiento del entorno.
- Menor tamaño del hardware.
- Implementación rápida.
- Reutilización de componentes.
- Minimización del impacto frente a errores/cambios.

Desventajas de uso de los contenedores

- Complejidad

Tipos de contenedores

- **Contenedores de sistemas:** Son similares a las máquinas virtuales, comparten el núcleo del anfitrión. Un ejemplo: **LXC**, que forma parte del núcleo Linux y que nos aporta aislamiento y seguridad usando cgroups y namespaces.
- **Contenedores de aplicaciones:** Especializados en la ejecución de aplicaciones, normalmente cada contenedor ejecuta un sólo proceso. Contienen todas las librerías necesarias para que esa aplicación pueda funcionar. Ejemplo: **docker.**

Docker

- Virtualización ligera: aprovechamos mejor el hardware y únicamente necesitamos el sistema de archivos mínimo para que funcionen los servicios.
- Los contenedores son autosuficientes, sólo necesitamos una imagen para crear contenedores.
- Una imagen Docker podríamos entenderla como "un Sistema Operativo con aplicaciones instaladas".
- El proyecto nos ofrece es un repositorio de imágenes: Registry Docker Hub que nos permite gestionar imágenes.
- Un contenedor suele ejecutar un sólo servicio. Una aplicación suele necesitar la ejecución de varios contenedores que trabajan juntos

Componentes de Docker

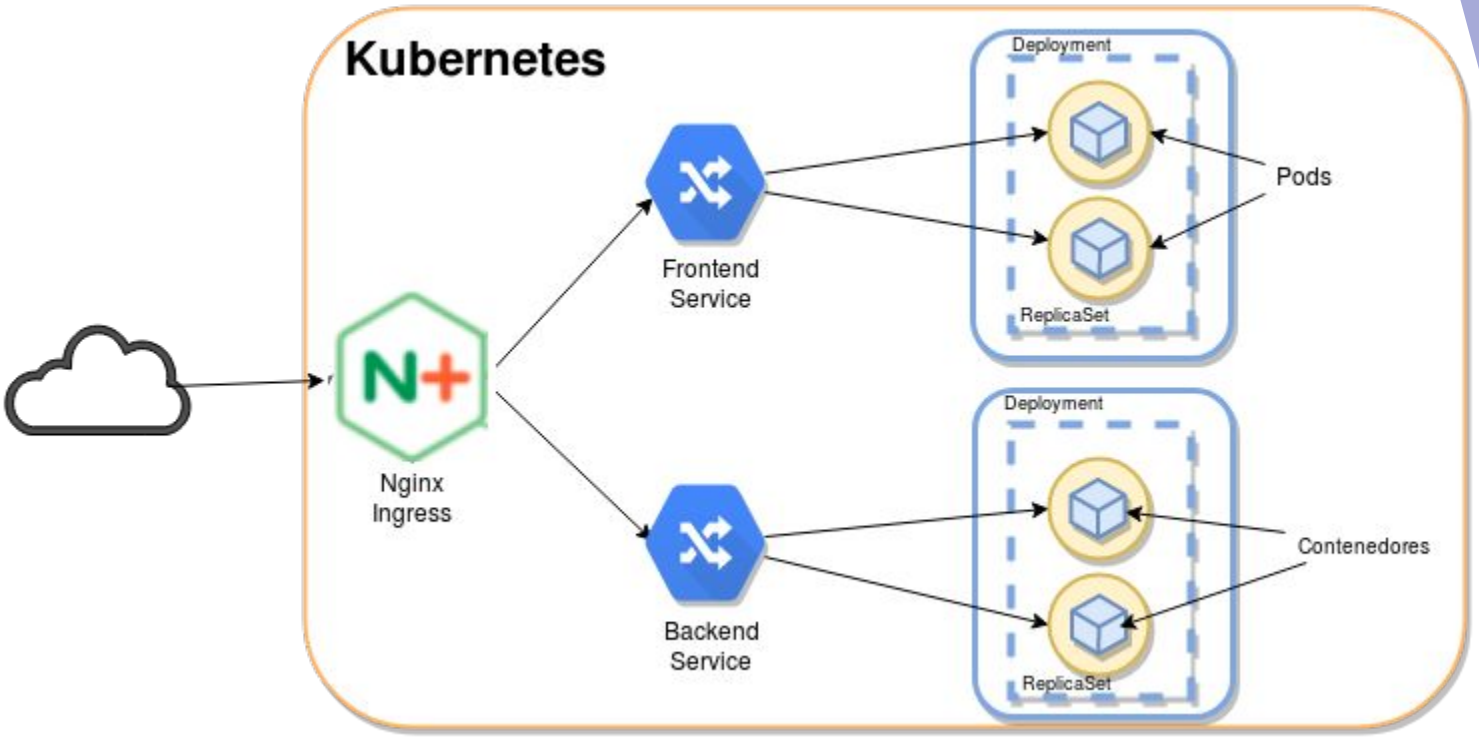
- **Docker Engine:** Es un demonio que corre sobre cualquier distribución de Linux y que expone una API externa para la gestión de imágenes y contenedores.
- **Docker Client:** Es el cliente de línea de comandos (CLI) que nos permite gestionar el Docker Engine. El cliente docker se puede configurar para trabajar con con un Docker Engine local o remoto.
- **Docker Registry:** La finalidad de este componente es almacenar las imágenes generadas por el Docker Engine. Nos permite distribuir nuestras imágenes. Podemos instalar un registro privado, o hacer uso de uno público como Docker Hub

2. Kubernetes

Kubernetes

Kubernetes es un orquestador de contenedores de aplicación escrito en Go y con licencia Apache-2.0, fue lanzado el 21 de julio de 2015 por Google que rápidamente se alió con la Linux Foundation para crear la Cloud Native Computing Foundation (CNCF) y dejó el proyecto en sus manos.

Recursos de Kubernetes



Recursos de Kubernetes

- **Pods:** La unidad mínima de computación, permite ejecutar contenedores. Representa un conjunto de contenedores y almacenamiento compartido que comparte una única IP.
- **ReplicaSet**: Asegura que siempre se ejecute un número de réplicas de un pod determinado. Nos proporciona las siguientes características:
  - **Que no haya caída del servicio**
  - **Tolerancia a errores**
  - **Escalabilidad dinámica**

Recursos de Kubernetes

- **Deployment**: Nos permite manejar los ReplicaSets. Nos proporciona las siguientes características:
  - **Actualizaciones continuas**
  - **Despliegues automáticos**
- **Service**: Nos permite el acceso a los pod.
- **Ingress:** Nos permite implementar un proxy inverso para el acceso a los distintos servicios establecidos. Estos dos elementos nos proporcionan la siguiente funcionalidad:
  - **Balanceo de carga**

3. OpenShift

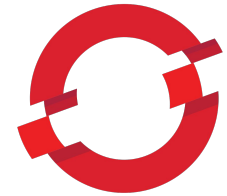
Plataforma de Desarrollo, con características de Cloud Computing (PaaS) desarrollada por Red Hat

- Nos centramos en el desarrollo de la aplicación
- Openshift utiliza internamente Docker y Kubernetes
- Nos permite desplegar aplicaciones en diferentes entornos (desarrollo, producción,...)
- Facilita la integración continúa
- Tenemos varías formas de interactuar con OpenShift: aplicación web, CLI o API REST

OPENSHIFT

Ventajas

- Simplifica el ciclo de vida de implantación de nuestras aplicaciones que nos ofrece Docker.
  - El desarrollador sólo se tiene que centrar en el desarrollo de su aplicación.
  - El proyecto se guardará en un repositorio GitHub
  - .A la hora de desplegar la aplicación, OpenShift leerá el código fuente del repositorio GitHub
  - E inyectará el código fuente en una imagen base de Docker (diferencias según el lenguaje de programación) creando una nueva imagen de forma automática.
  - source2image



OPENSHIFT

Ventajas

- Nos ofrece todas las ventajas del uso de Kubernetes:
  - Tolerancia a errores
  - Escalabilidad dinámica
  - Actualizaciones continuas
  - Despliegues automáticos
  - Enrutamiento a nuestras aplicaciones
  - Balanceo de carga
  - Volúmenes persistentes
- Pero además...

OPENSHIFT

Ventajas

- OpenShift nos ofrece una serie de extras:
  - Gestión más sencilla de proyectos y usuarios
  - Conjunto de imágenes base para distintos lenguajes de programación y bases de datos
  - Asignación automática de nombre para nuestra aplicación (recurso service y ingress controller)
  - Gestión más sencilla de los volúmenes
  - Flujos de CI/CD integradas
  - Herramientas de métrica y monitorización