Kubernetes

Grupo 1

¡Hola!

kubernetes

Somos el grupo 1

- Agustín Lavarello
- Sebastián Favaron
- Julián Palacci

Que es Kubernetes?

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios.

Facilita la automatización y la configuración declarativa.

Es un orquestador de contenedores

Documentación: https://kubernetes.io/docs/home/

K8s as a Service (KaaS)

Google Kubernetes Engine

Amazon EKS

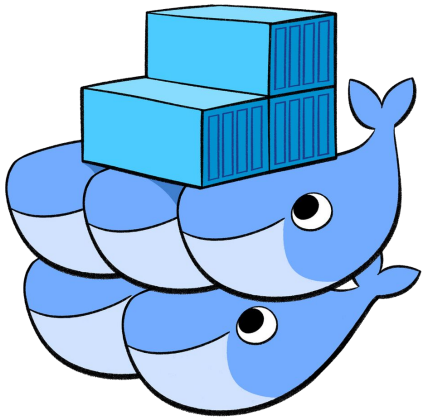
AWS Elastic Container Service for Kubernetes

Azure Kubernetes Service

Azure

Competencia

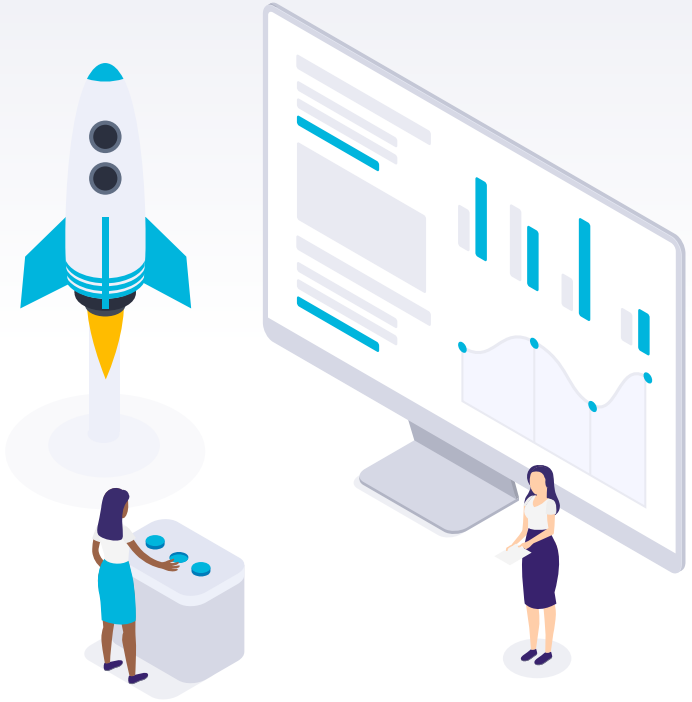
Docker Swarm



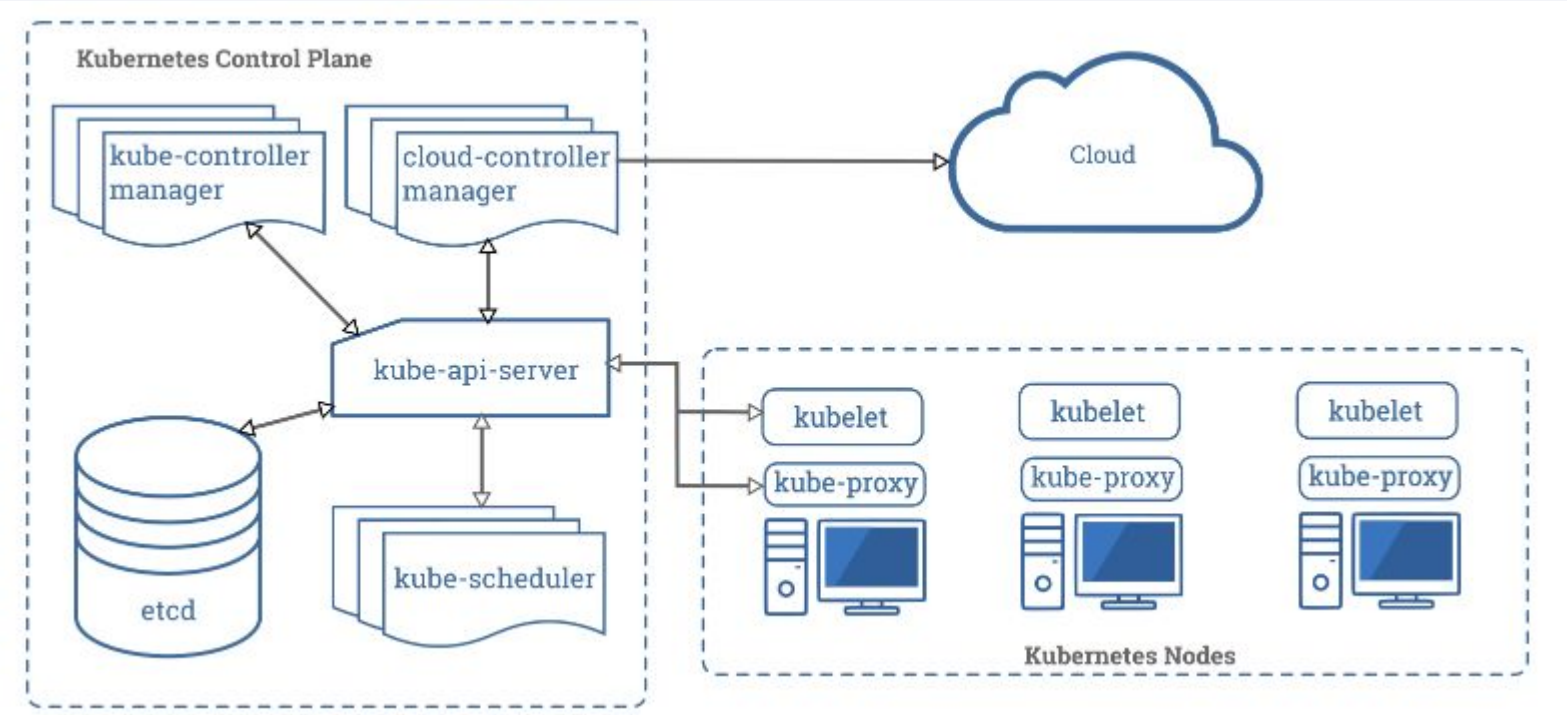
Diferencias

- Más fácil instalación
- Corre de docker-cli
- Menos complejo
  - Menos resistente (fault tolerance)
  - Mejor escalabilidad (pero no mejor automatización de la escalabilidad)
- Sin herramienta propia de logging y monitoreo
- UI solo disponible para la versión paga

1 Arquitectura



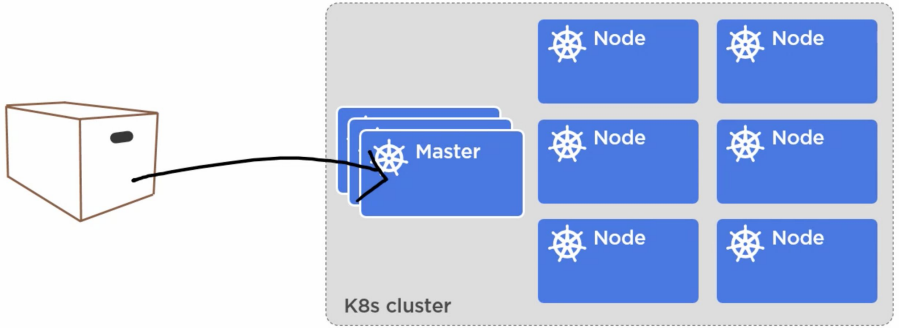
Arquitectura



Cluster

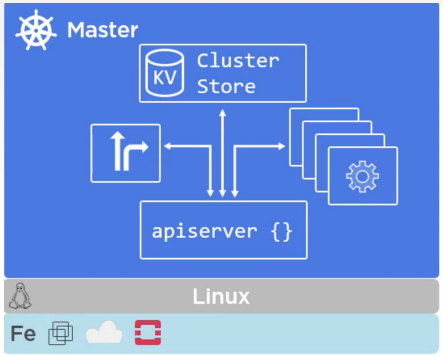
Un cluster está compuesto por varios nodos. Los roles dentro del cluster pueden ser:

- Master
- Node



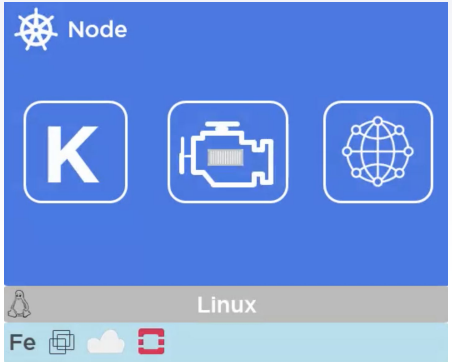
Master

- El Master de Kubernetes es el responsable de mantener el estado deseado del cluster
- Es un conjunto de tres procesos: kube-apiserver, kube-controller-manager y kube-scheduler.
- El master expone una API REST a traves del apiserver
- La API consume JSON y permite especificar el estado deseado del cluster

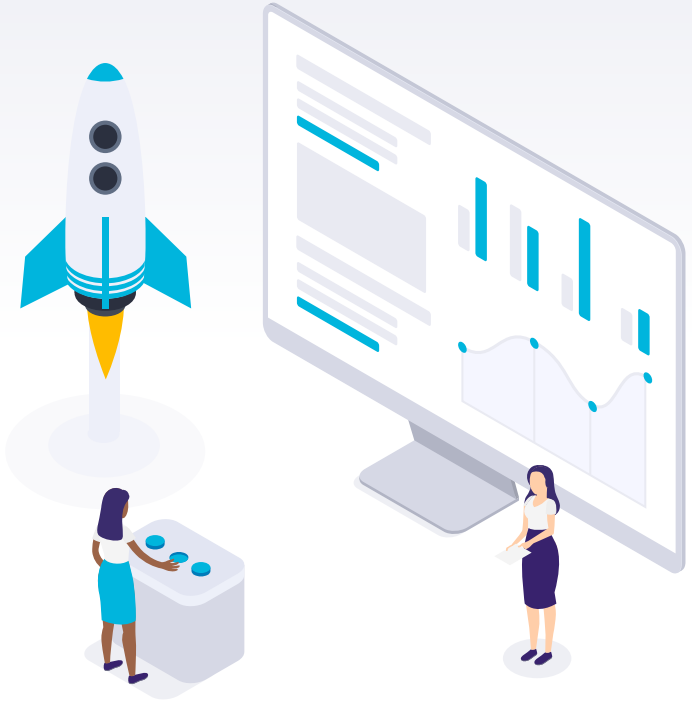


Node

- También llamados Minions
- Compuesto por Kubelet, Container Engine y kube-proxy
- El **Kubelet** se encarga de registrar al nodo con el apiserver, instanciar los Pods y reportar al Master
- El **Container Engine** administra los contenedores (usualmente es Docker)
- **Kube-proxy** se encarga de administrar la red de los Pods

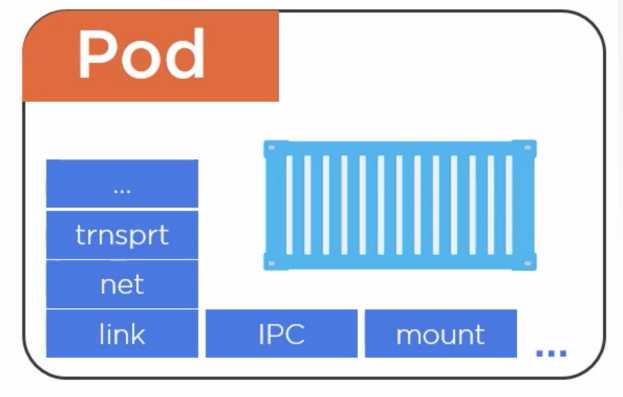


2 Objetos



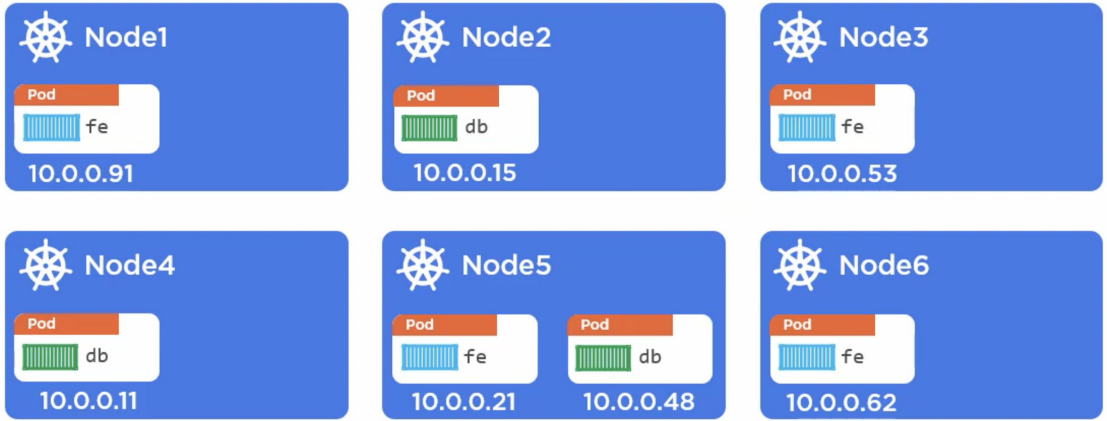
Pods

- Es la unidad atómica: lo más pequeño que se puede deployar en un cluster K8s
- Los contenedores siempre corren dentro de Pods
- Los Pods pueden tener múltiples contenedores
- Todos los contenedores dentro de un Pod comparten el mismo ambiente del Pod



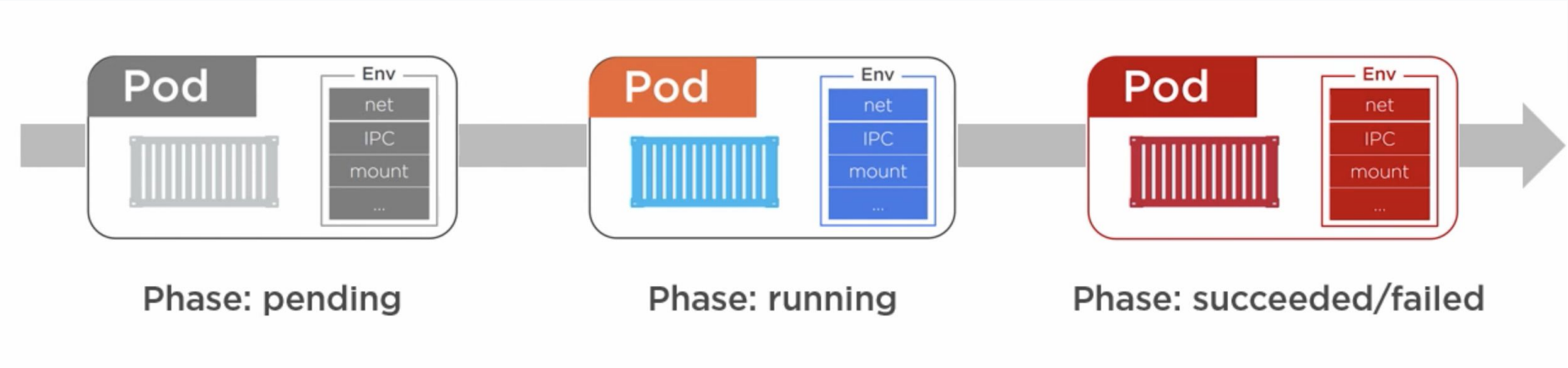
Pods

- Un nodo contiene uno o más Pods
- Cada Pod nuevo obtiene una dirección IP nueva accesible solo desde el cluster

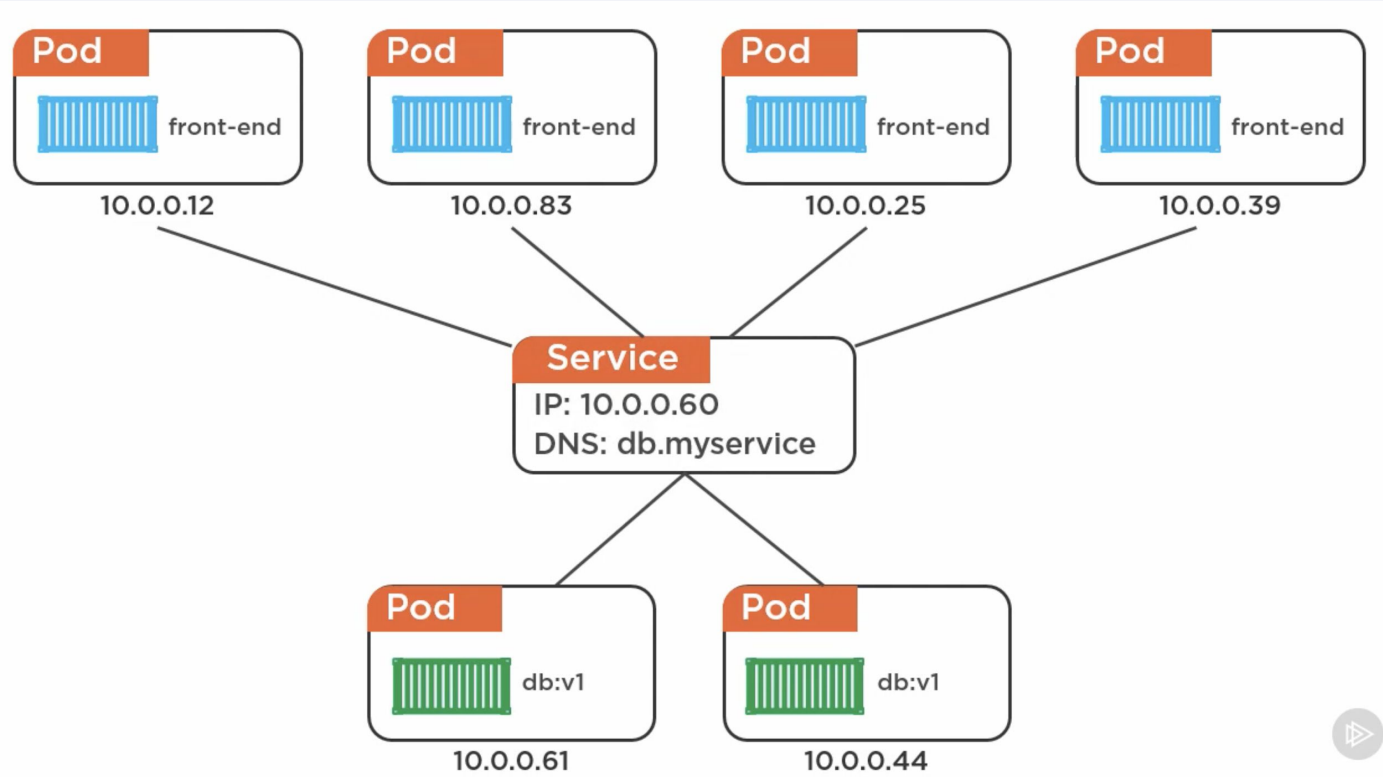


Pod Lifecycle

- Los Pods son mortales: nacen y cuando mueren no son resucitados



Services


Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.61
Pod
db:v1
10.0.0.44

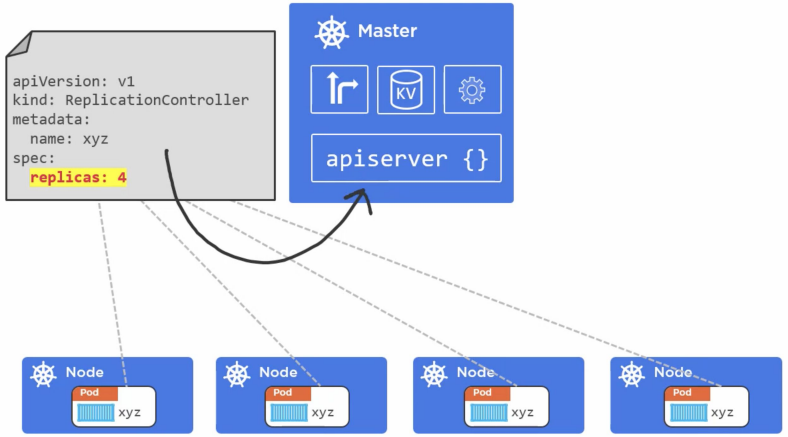
Tipos de Servicio

- **ClusterIP**: Expone el servicio con una IP dentro del cluster.
- **NodePort**: Expone el servicio en todos los nodos en un puerto
- **LoadBalancer**: Expone un servicio usando un load balancer provisto por la nube
- **ExternalName**: Mapea el servicio a lo que contiene el campo de externalName

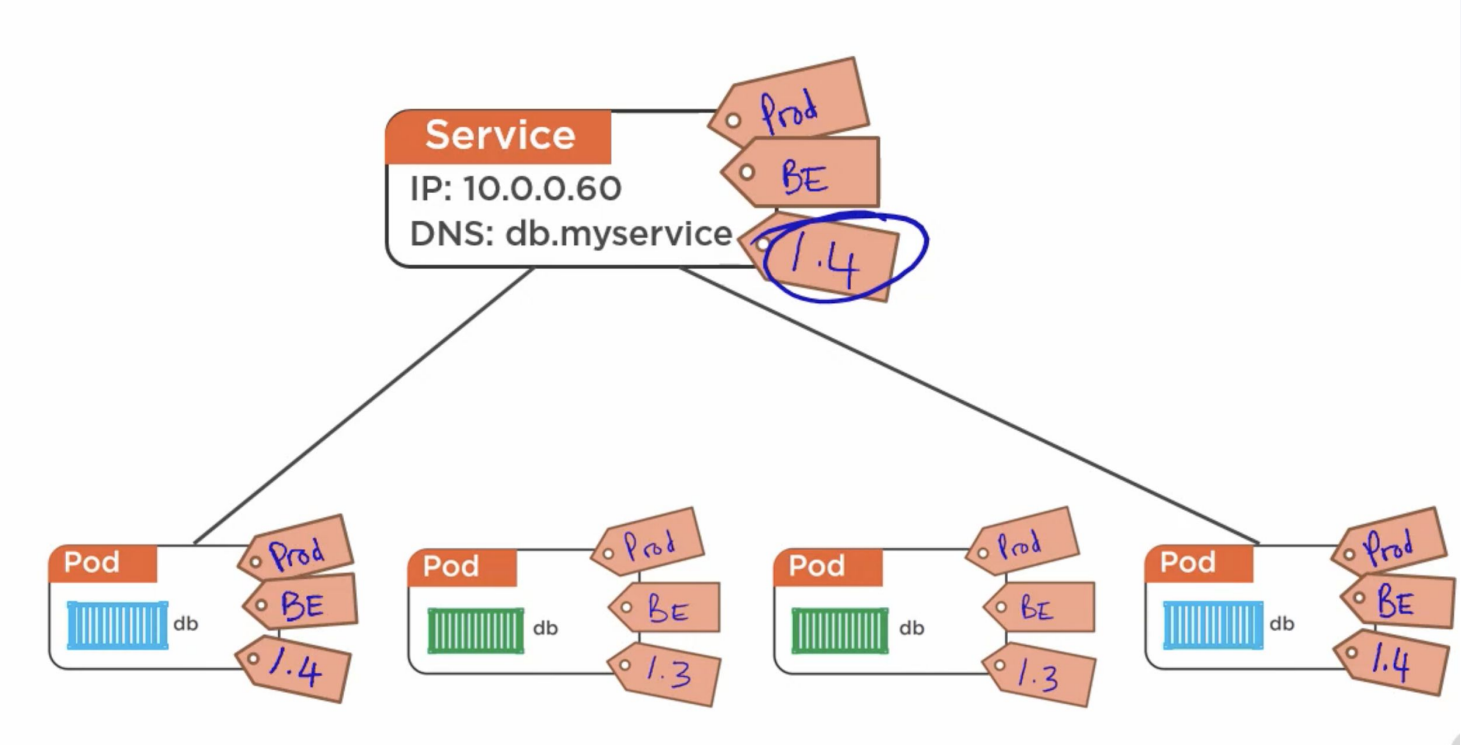
Deployments

Los deployments son una manera declarativa de hacer cambios a los Pods y a los ReplicaSets

- Canary deployments
- Blue and Green deployments
- Rollouts



Labels


Service
IP: 10.0.0.60
DNS: db.myservice
Prod
BE
1.4
Pod
db
Prod
BE
1.4
Pod
db
Prod
BE
1.3
Pod
db
Prod
BE
1.3
Pod
db
Prod
BE
1.4

3

Demo

Nuestra arquitectura

- Creación de un cluster utilizando **kind**. Cada nodo es un contenedor Docker que tiene instalado Kubernetes.

Ventajas

- Permite probar un cluster multi-nodo de Kubernetes de forma rápida y fácil
- Cada nodo es un contenedor Docker, son más livianos que levantar máquinas virtuales

Contras

- No se pueden agregar nodos una vez que se creó el cluster
- No es adecuado para ambientes de producción

Nuestra arquitectura

- Servidor con un nginx que escucha en el puerto 80
- Cluster
  - Exponemos un servicio NodePort con Nginx para redirigir a los distintos servicios dentro del cluster. El puerto expuesto es el 32333
  - Servicio ClusterIP para comunicarse con la API
  - Pods con las API que exponen el puerto 8888
  - Servicio de tipo ExternalName para acceder a la base de datos fuera del cluster
- Base de datos local Postgres