Sesión 07

Clustering & Orquestación

Instructor:

ERICK ARÓSTEGUI

earostegui@galaxy.edu.pe

GALAXY
TRAINING

ÍNDICE

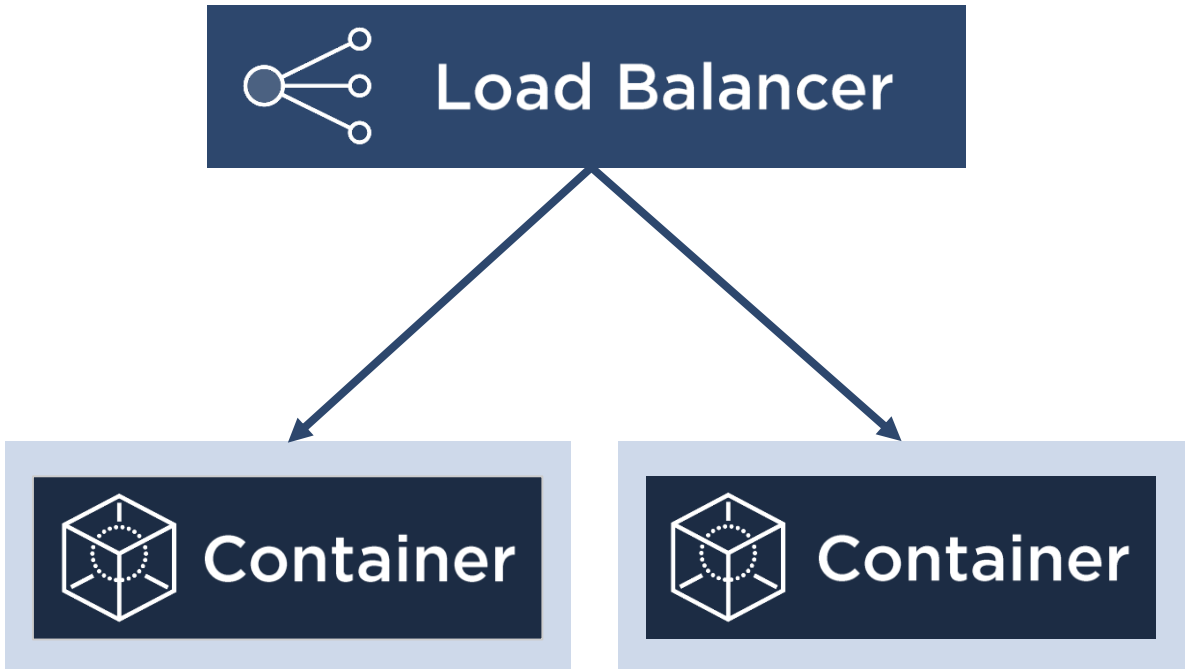
GALAXY
TRAINING

01

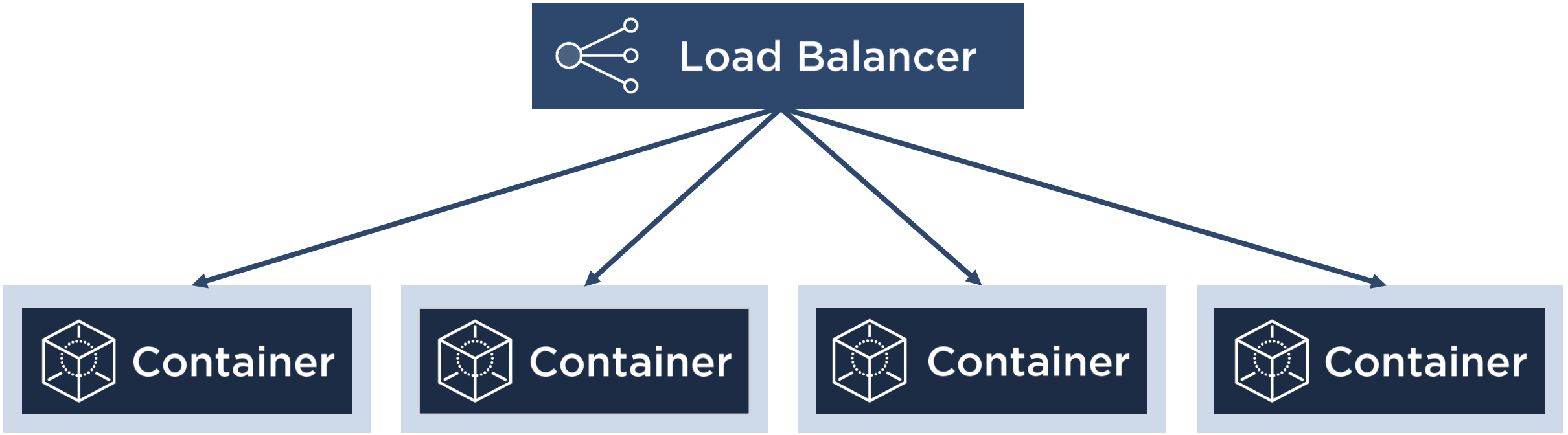
Software para clustering, orquestación y programación de contenedores.

GALAXY
TRAINING

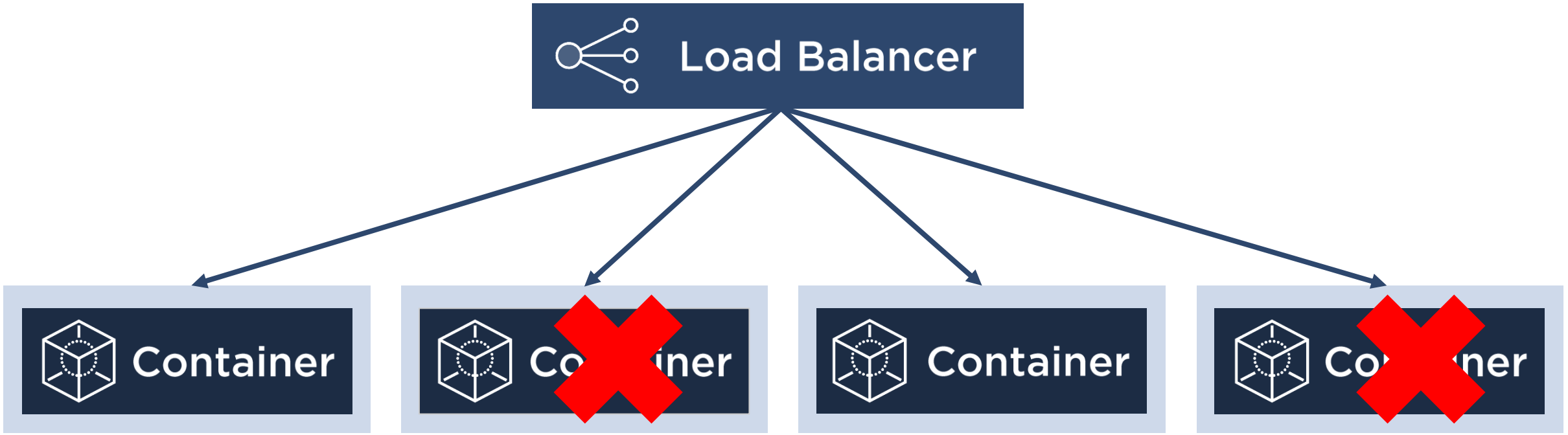
¿Cómo administrar los contenedores?



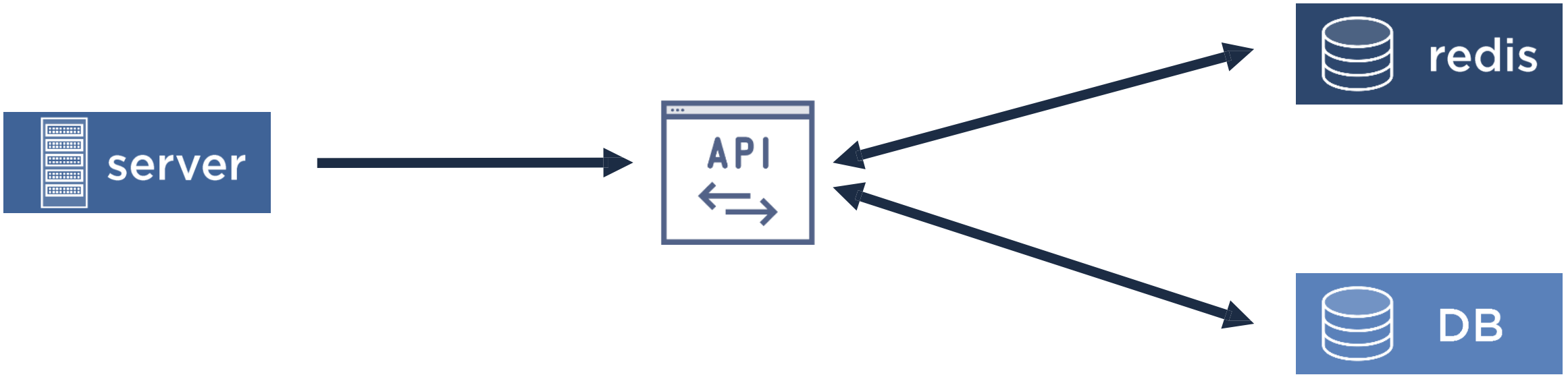
¿Cómo administrar los contenedores?



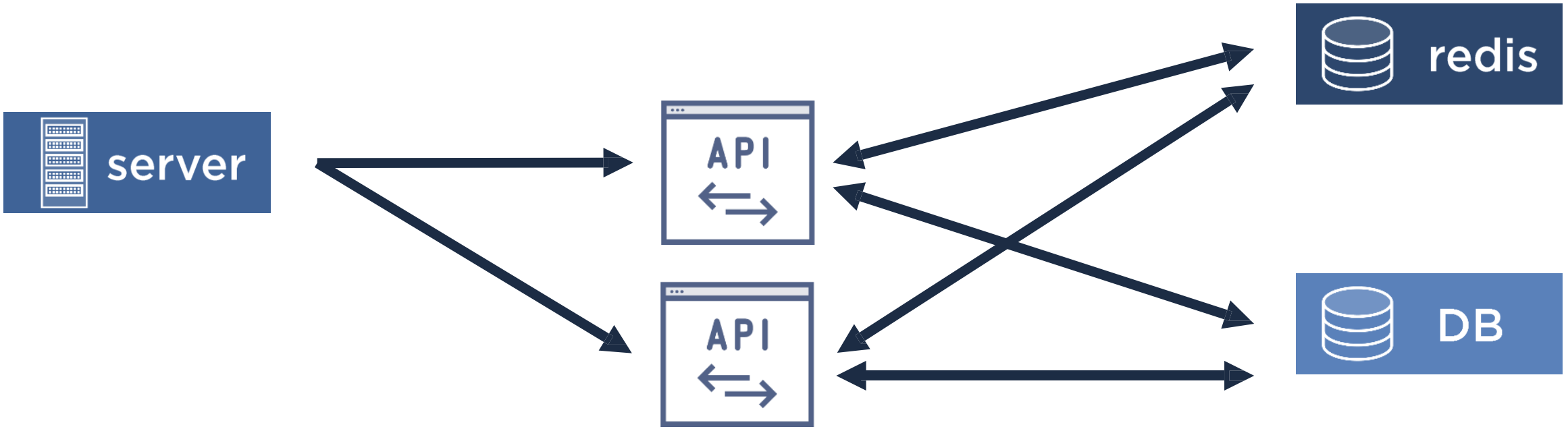
¿Cómo administrar los contenedores?



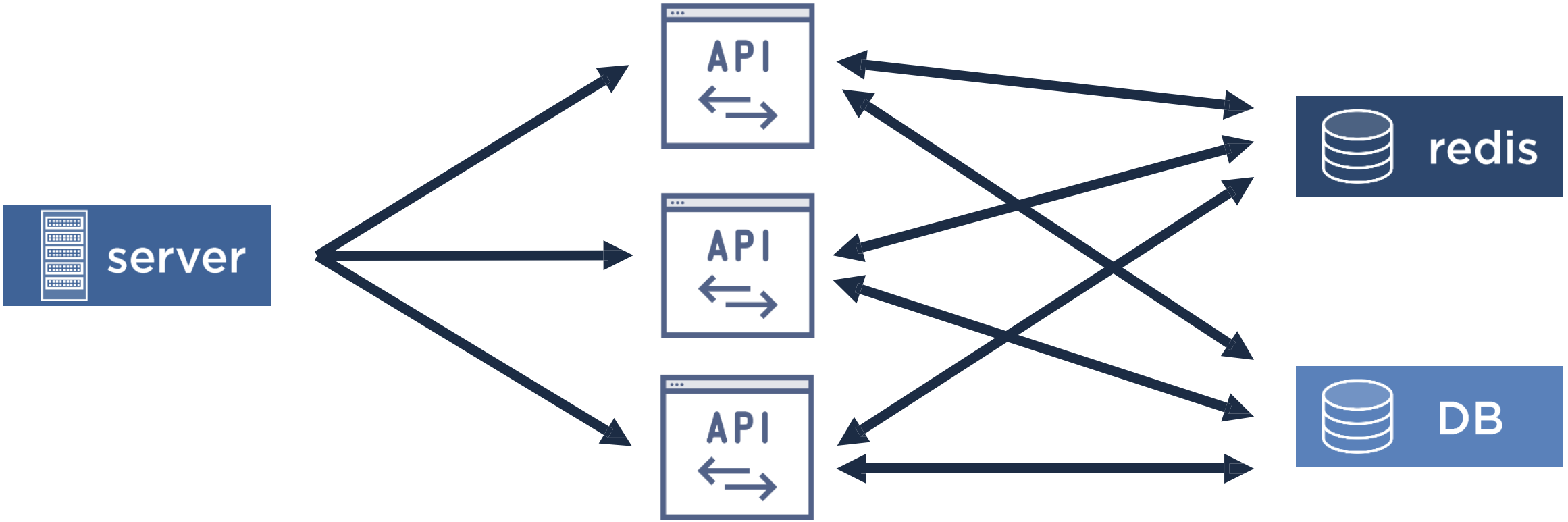
¿Cómo gestionar todos estos contenedores?



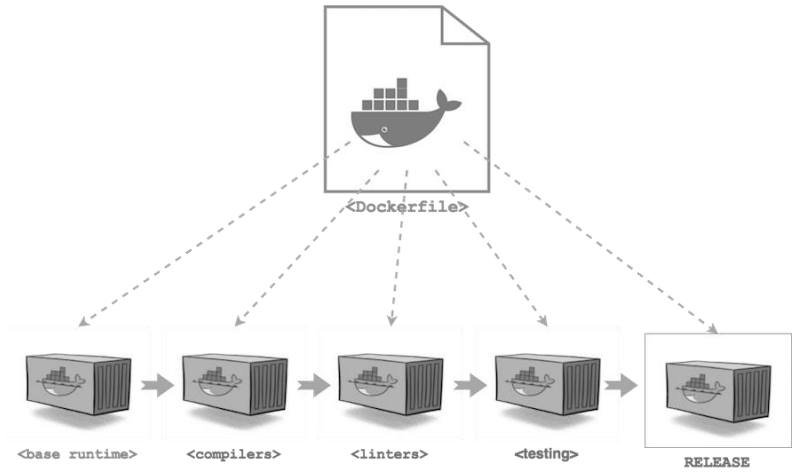
¿Cómo gestionar todos estos contenedores?



¿Cómo gestionar todos estos contenedores?



¿Que necesitamos?



- **Empaquetar una aplicación** y dejar que otra persona la administre por nosotros
- **No preocuparse por la gestión** de contenedores
- **Eliminar puntos únicos de falla**
- **Escalar** contenedores
- **Actualizar contenedores** sin cerrar la aplicación
- Tener opciones sólidas de **almacenamiento persistente y de trabajo en red**

Kubernetes

Kubernetes es el director de la orquesta de contenedores.

Kubernetes (K8s) es un sistema de código abierto para automatizar el despliegue, el escalado y la administración de aplicaciones en contenedores.

02

GALAXY
TRAINING

Kubernetes (beneficios y principios operativos)

Características clave de kubernetes

- Service Discovery / Load Balancing
- Gestión de almacenamiento
- Automatización de Despliegues y Rollbacks
- Autorrecuperación
- Gestión de configuración y secrets
- Escalamiento horizontal

Kubernetes

- **Gestión de contenedores y clústeres**
- Proyecto de **código abierto**
- Utilizado internamente por Google (**2003-2004 inicios**) durante más de 15 años y donado a Cloud Native Computing Foundation (**2014**)
- **Compatible** con todas las principales plataformas en **la nube**
- Proporciona una forma "**declarativa**" de definir el estado de un clúster.

Beneficios usando Kubernetes

Velocidad de despliegue

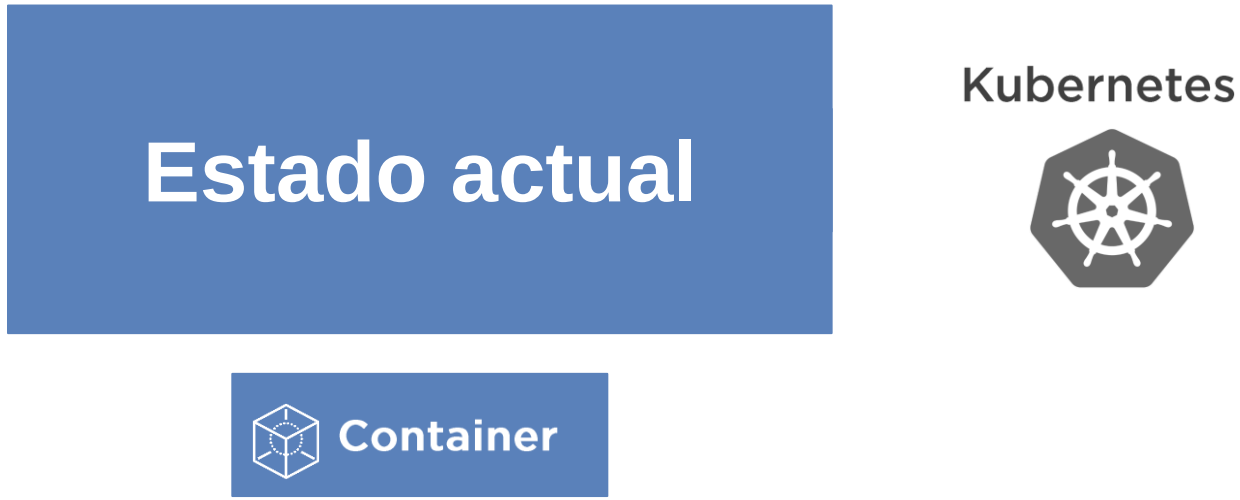
Capacidad para absorber los cambios rápidamente

Capacidad para recuperarse rápidamente

Ocultar la complejidad en el clúster

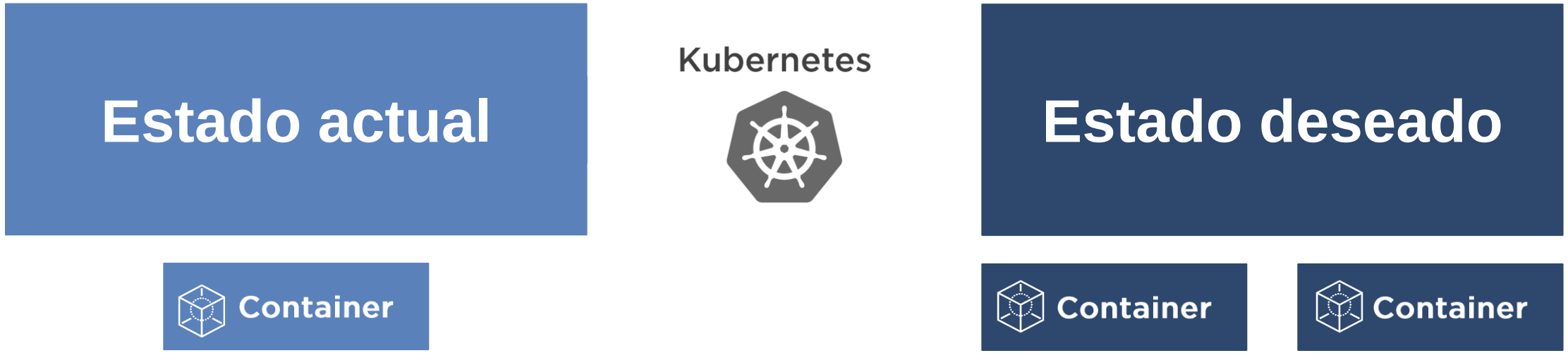
Kubernetes te lleva al estado deseado

Kubernetes te lleva al estado deseado

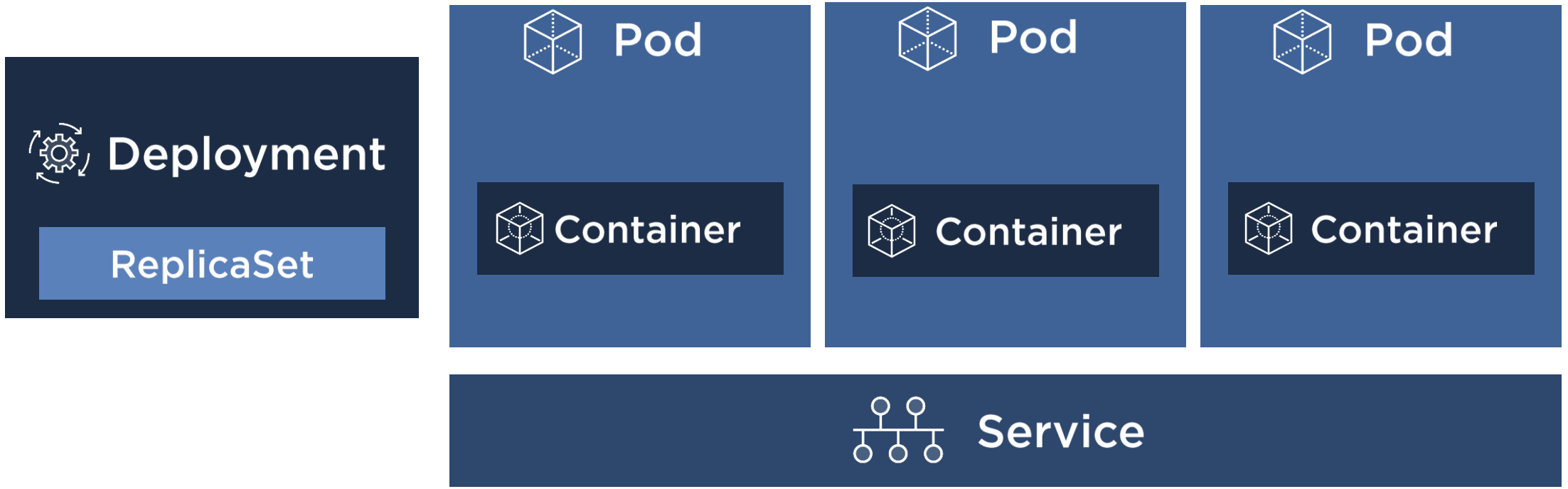


GALAXY TRAINING

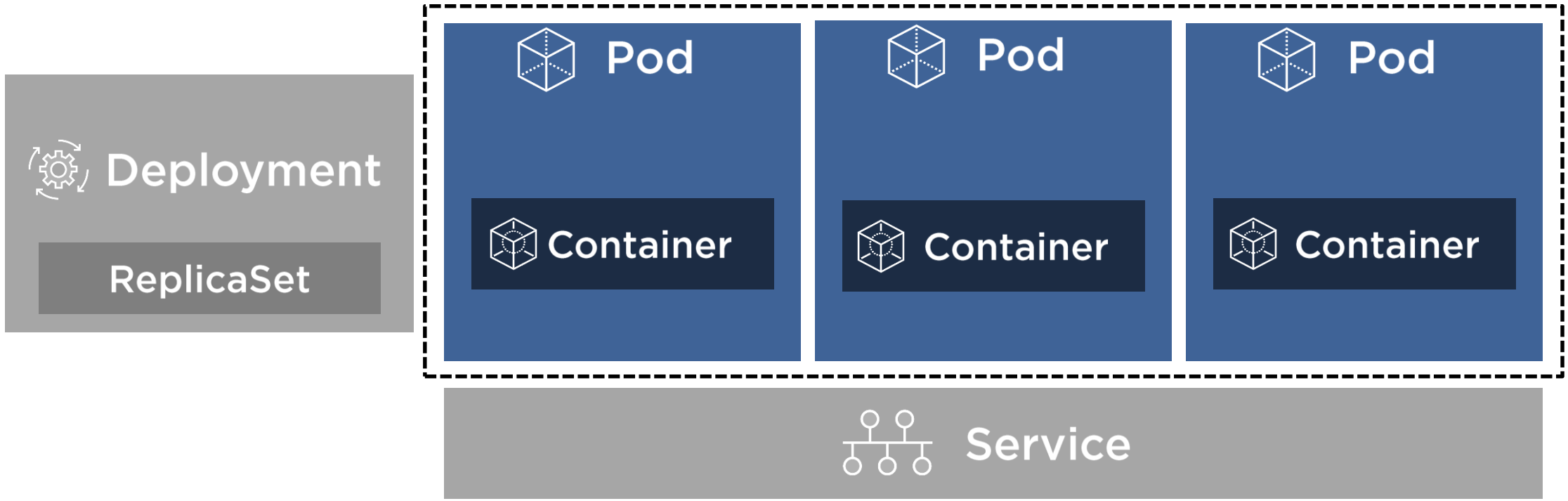
Kubernetes te lleva al estado deseado



Kubernetes - Arquitectura

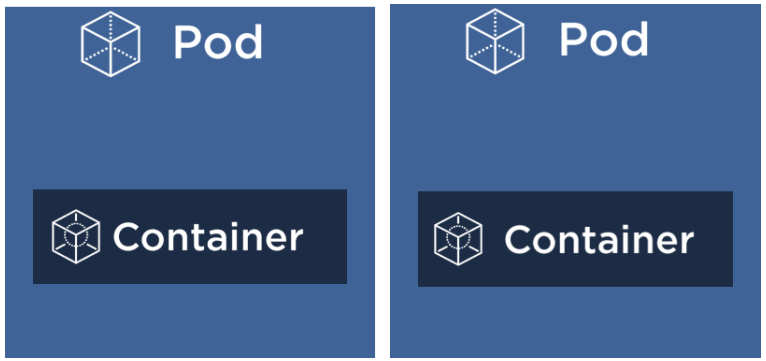


Kubernetes - Arquitectura



Un pod es la unidad de ejecución básica de las aplicaciones de Kubernetes, **la unidad más pequeña y simple** del modelo de objetos de Kubernetes que crea o implementa

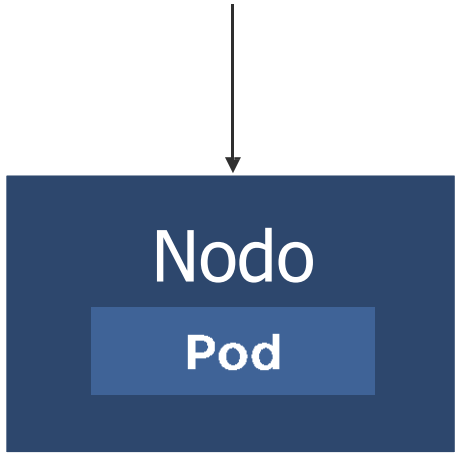
Kubernetes – Arquitectura - Pod



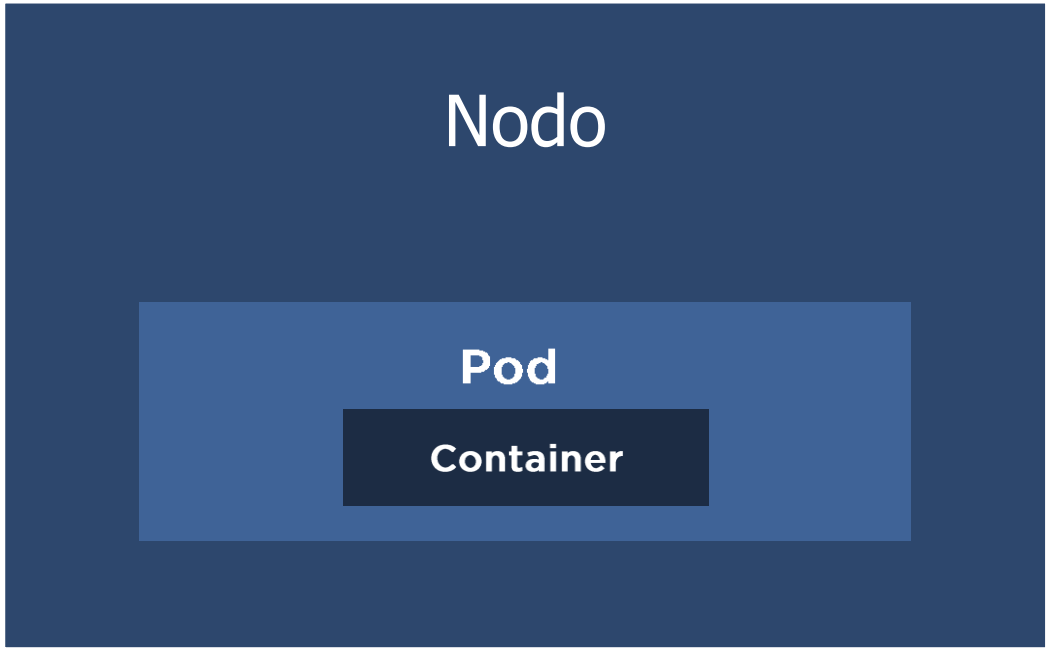
- Pequeño objeto del modelo de objetos de Kubernetes
- **Entorno de ejecución para contenedores**
- **Organizar la aplicación en "partes"** a traves de Pod's (servidor, almacenamiento en caché, API, base de datos, etc.)
- Pod IP, memoria, volúmenes, etc. compartidos entre contenedores
- **Escalamiento horizontal** agregando réplicas de pod.
- Podrá vivir y morir pero **nunca volverá a la vida**

¿Que hace un POD?

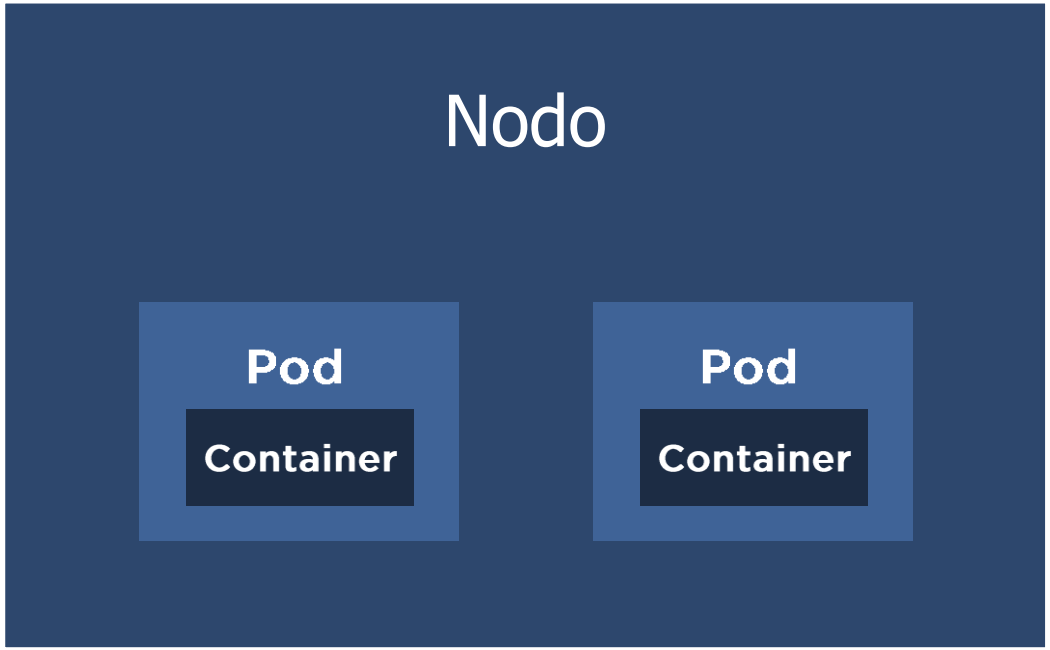
Master



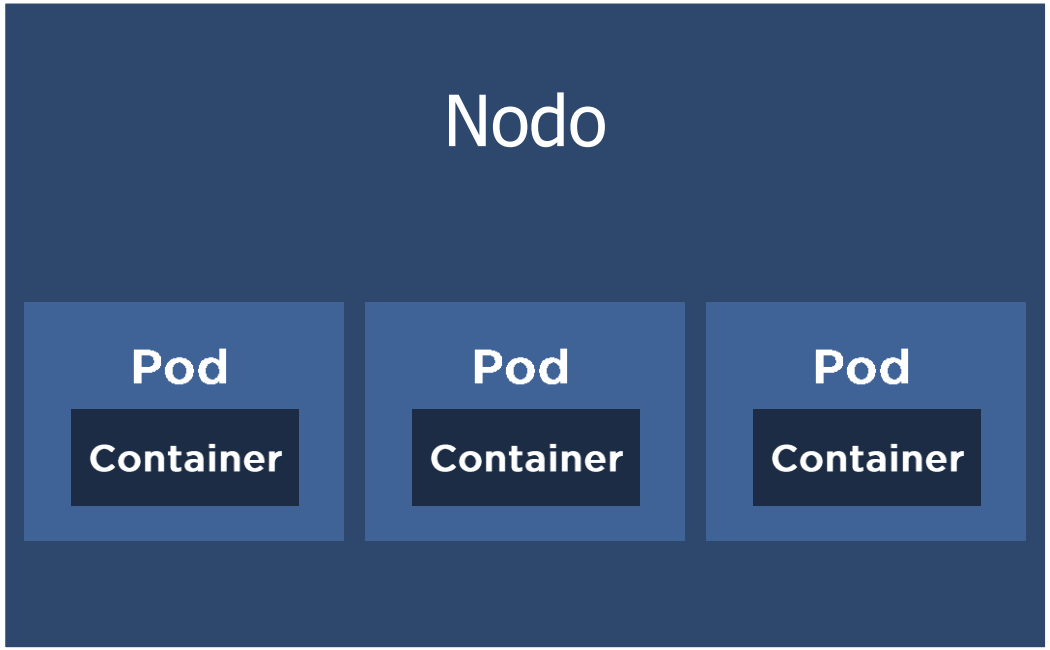
¿Que hace un POD?



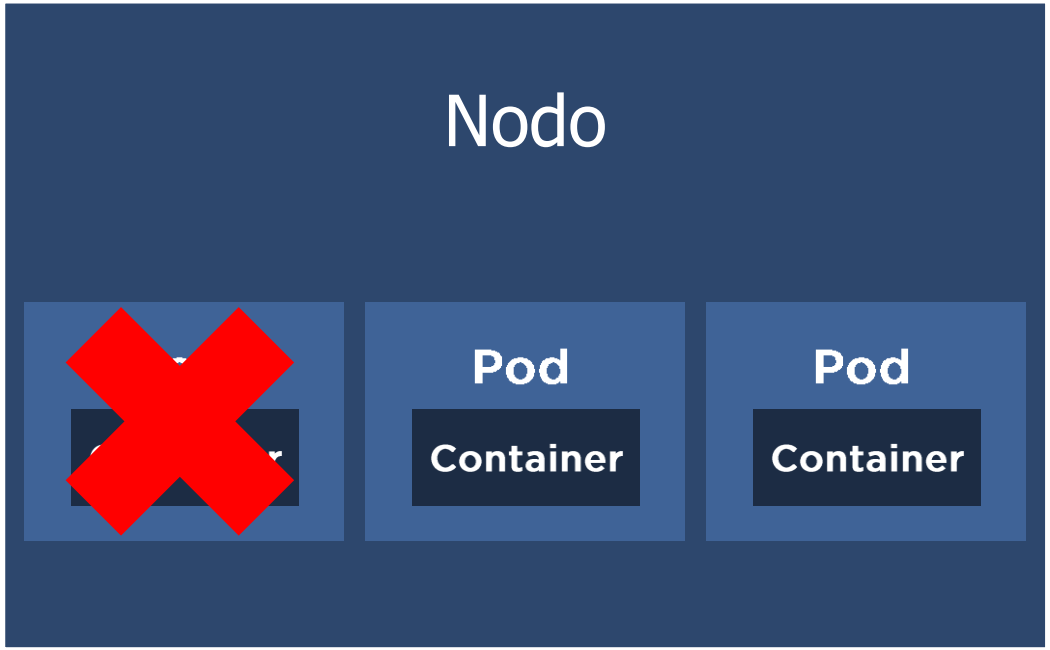
¿Que hace un POD?



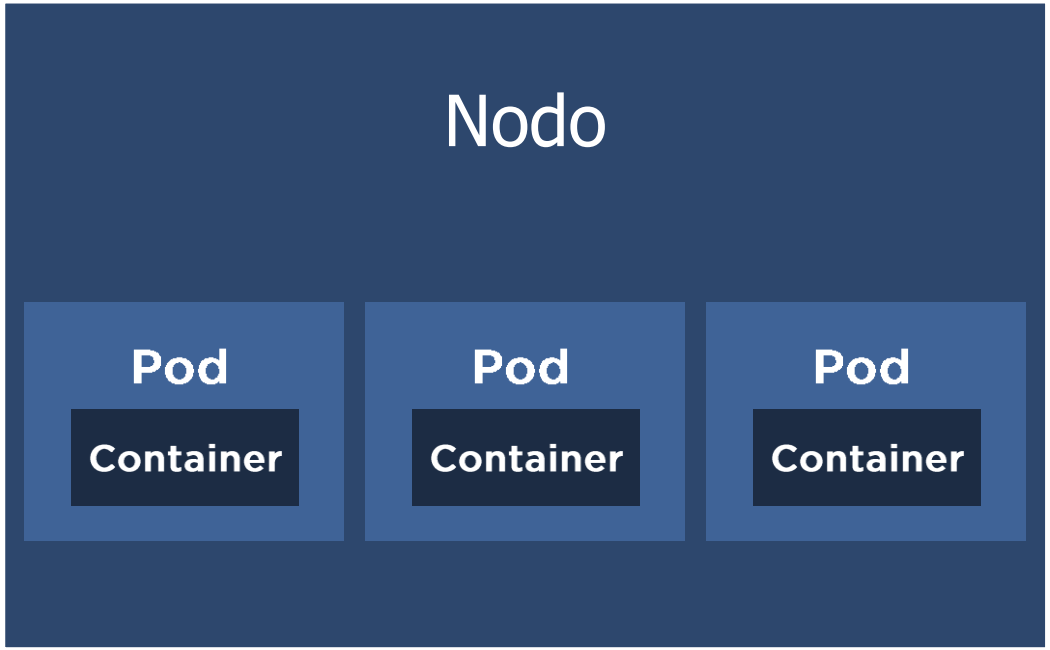
¿Que hace un POD?



¿Que hace un POD?

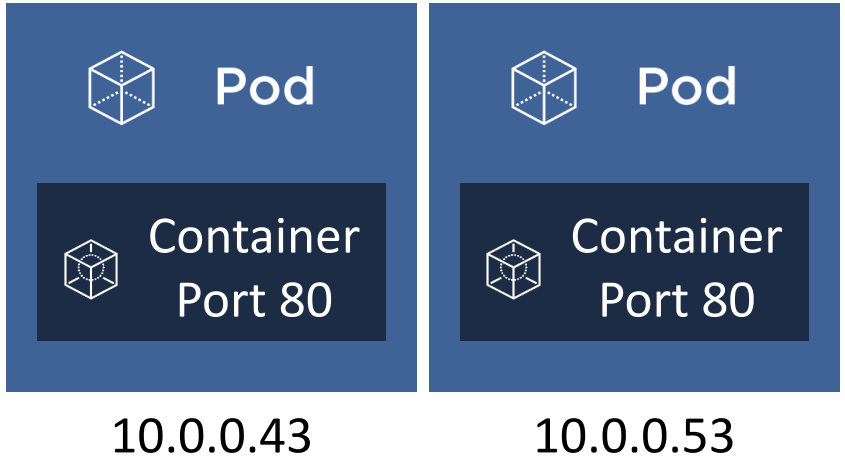


¿Que hace un POD?



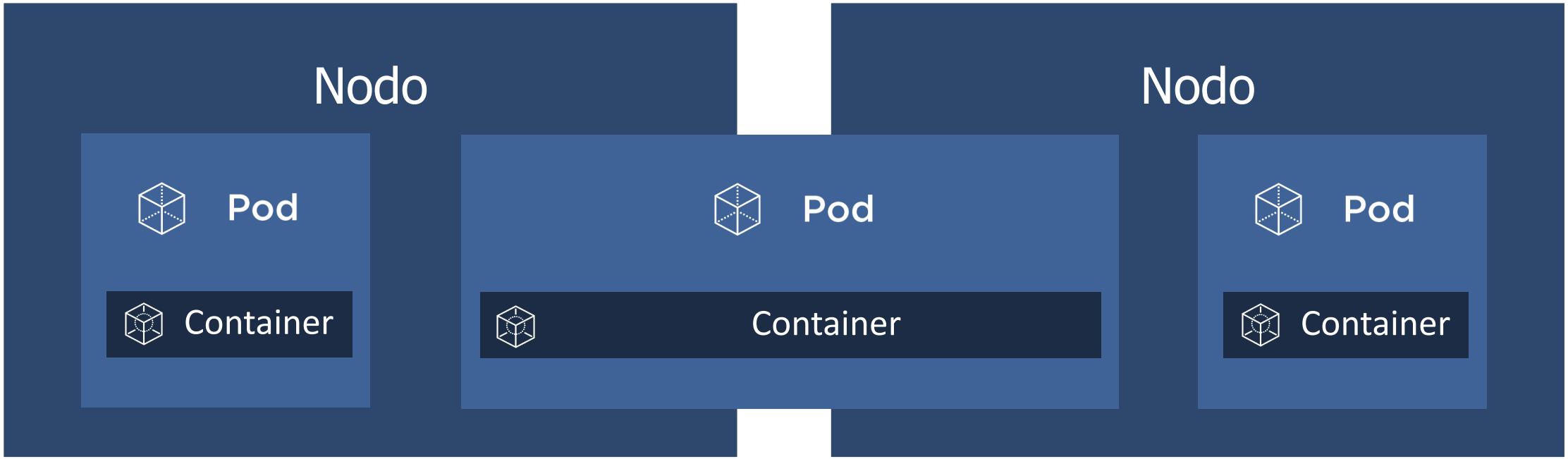
Kubernetes – Arquitectura - Pod

Pods, IPs y Puertos



- Los contenedores de pod **comparten el namespace** de red (compartir IP / puerto)
- Los contenedores del pod tienen la misma interfaz de red de bucle invertido (**localhost**)
- Los procesos de contenedor deben **vincularse a diferentes puertos** dentro de un Pod
- Los **puertos pueden ser reutilizados** por contenedores en pods separados

Kubernetes – Arquitectura - Pod



Kubernetes – Arquitectura - Pod

Kubernetes se basa en **Probes (sondas)** para determinar el estado del contenedor de un POD

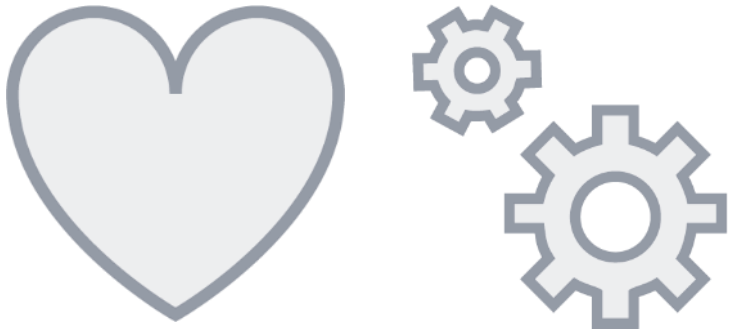
Una sonda es un **diagnóstico que se realiza periódicamente** por el kubelet a un contenedor

Kubernetes – Arquitectura - Pod

Tipos de sondeos

Kubernetes – Arquitectura - Pod

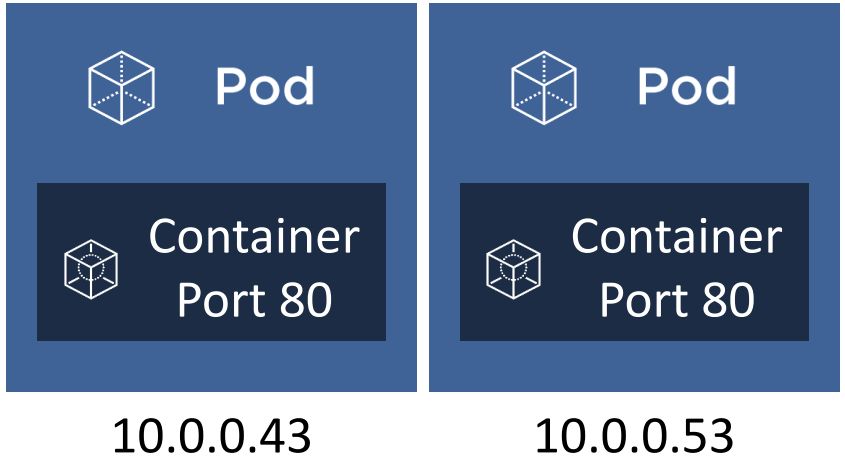
Tipos de sondeos



- **Liveness probes** determinan si un Pod está en buen estado y funcionando como se esperaba
- **Readiness probe** determina si un pod debe recibir solicitudes
- Los contenedores de **pod fallidos se vuelven a crear de forma predeterminada** (**reiniciarPolicy** tiene el valor predeterminado **Always**)

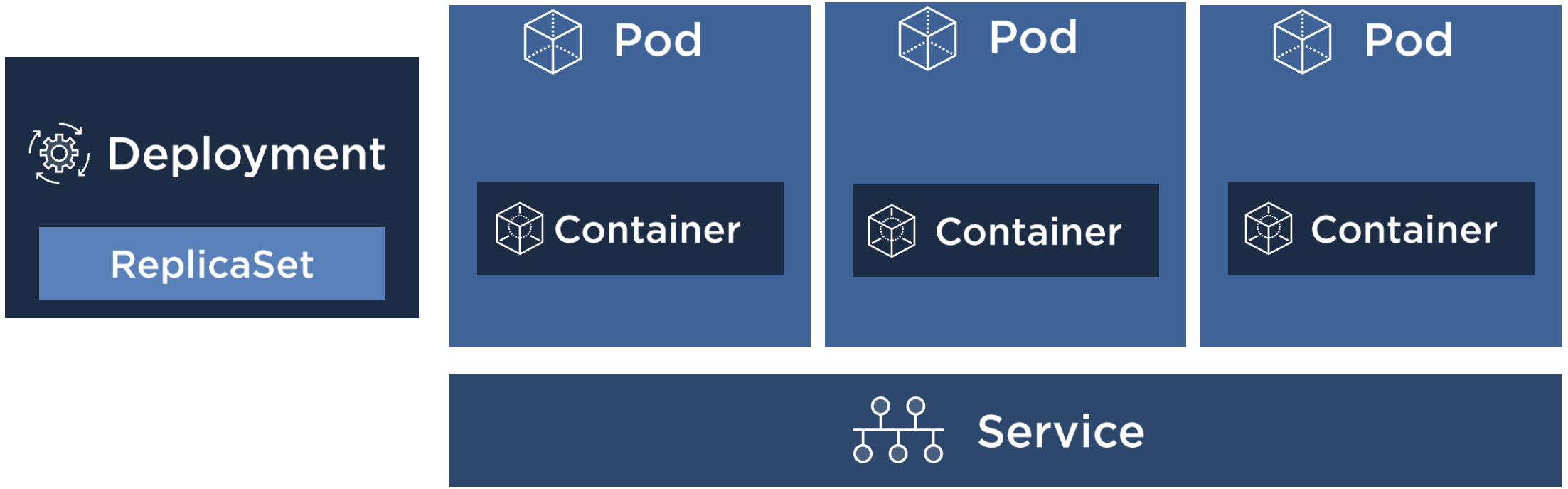
Kubernetes – Arquitectura - Pod

Tipos de sondeos

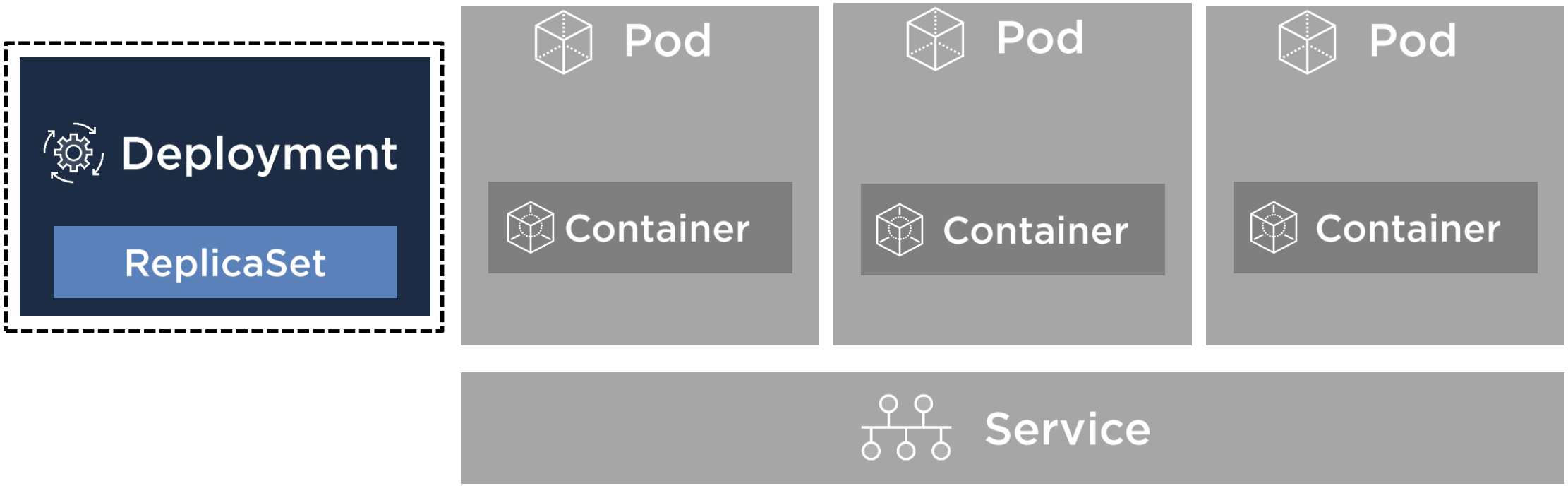


- **ExecAction** Ejecuta la acción dentro del contenedor.
- **TCPSocketAction**: Comprobación de TCP con la dirección IP del contenedor en un puerto especificado
- **HTTPGetAction**: solicitud HTTP GET contra el contenedor
- Las sondas pueden tener los siguientes resultados:
  - **Success**
  - **Failure**
  - **Unknown**

Kubernetes - Arquitectura



Kubernetes - Arquitectura



Un **ReplicaSet** es una forma declarativa de administrar pods

Un **Deployment** es una forma declarativa de administrar Pods usando un ReplicaSet

Kubernetes – Arquitectura - Deployment

Pods, Deployments y ReplicaSets

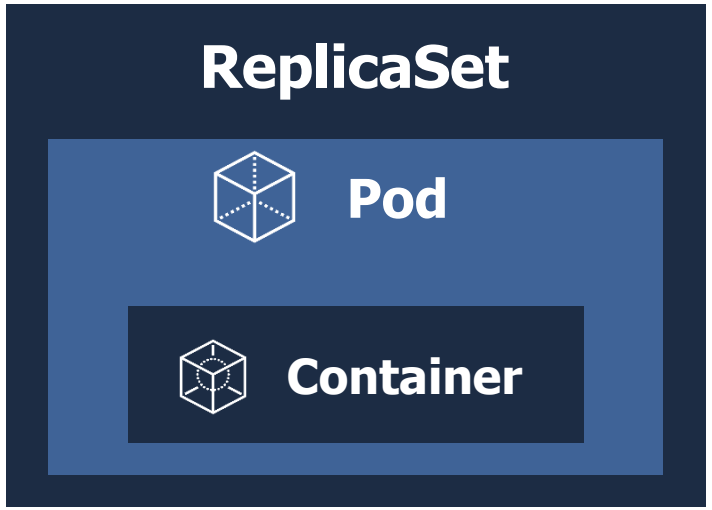
- Los pods representan el recurso más básico en kubernetes.
- Se pueden crear y destruir, pero nunca se vuelven a recrear.

¿Qué sucede si se destruye un Pod?

Los **Deployments** y **ReplicaSets** garantizan que los pods se **mantengan en ejecución** y se puedan usar para **escalar pods**

Kubernetes – Arquitectura - Deployment

¿Que hacen los ReplicaSets?

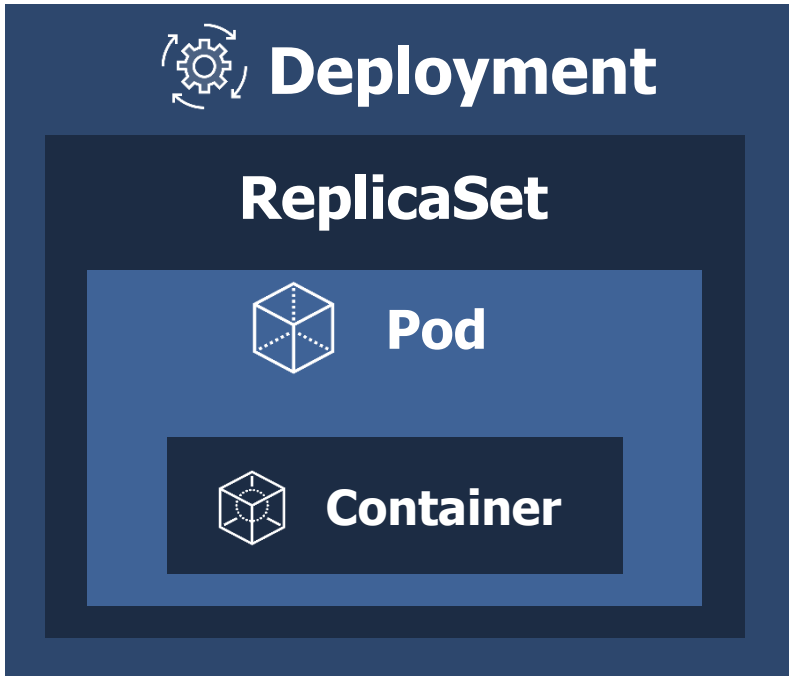


El ReplicaSet actúa como un controlador de Pod

- Mecanismos de **autorreparación**
- **Asegurar de que la cantidad solicitada** de pods esté disponible
- Proporcionar **tolerancia a fallas**
- Se puede usar para **escalar pods**
- Se basa en una **plantilla de Pod**
- **¡No es necesario crear pods directamente!**
- Utilizado por los Deployments

Kubernetes – Arquitectura - Deployment

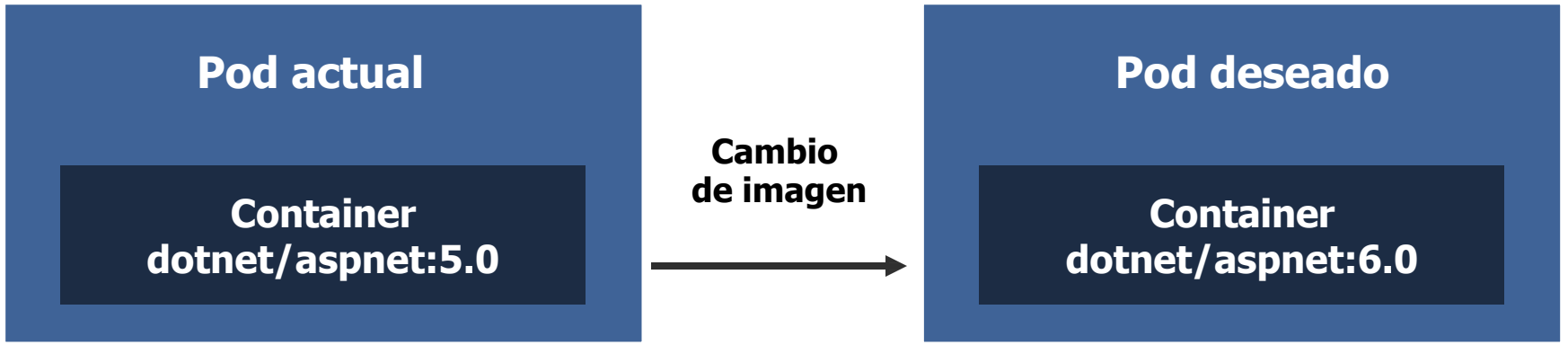
¿Que hacen los Deployments?



Una Deployment administra los pods

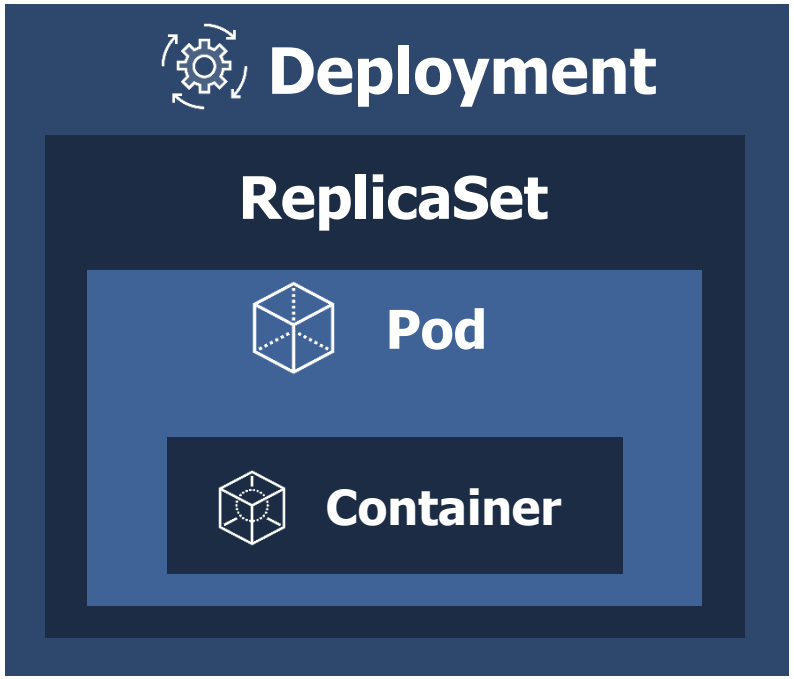
- Los pods se administran mediante ReplicaSets
- **Escale ReplicaSets**, escala pods
- **Admite actualizaciones sin tiempo de inactividad** al crear y destruir ReplicaSets
- Proporciona funcionalidad de **rollback**
- Crea una etiqueta única que se asigna al ReplicaSet y a los pods generados

Kubernetes – Arquitectura - Deployment



Kubernetes – Arquitectura - Deployment

Opciones de Deployment



Uno de los puntos fuertes de Kubernetes son las implementaciones sin tiempo de inactividad.

Actualizar pods de aplicaciones sin afectar a los usuarios finales

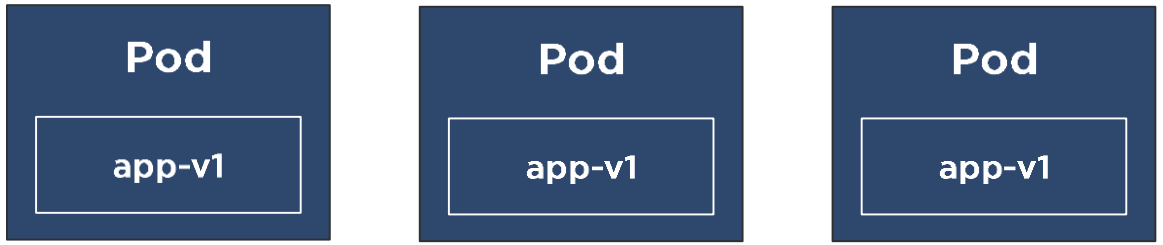
Hay varias opciones disponibles:

- **Rolling updates**
- **Blue-green deployments**
- **Canary deployments**
- **Rollbacks**

Kubernetes – Arquitectura - Deployment

Rolling Deployment

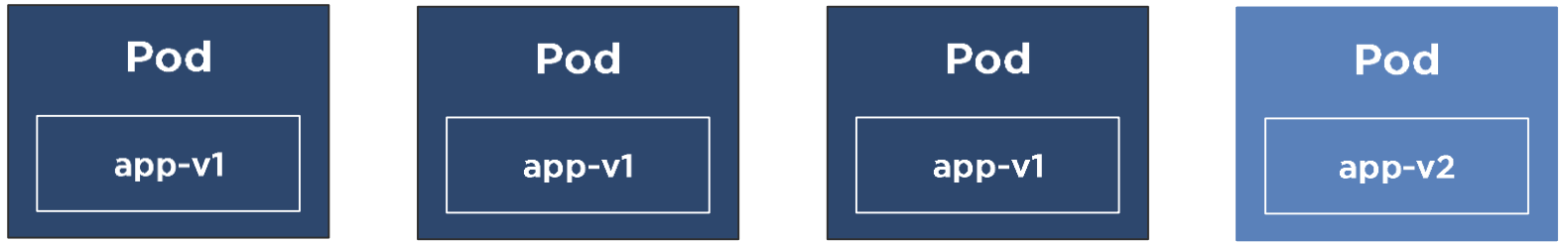
Initial Pod State



Kubernetes – Arquitectura - Deployment

Rolling Deployment

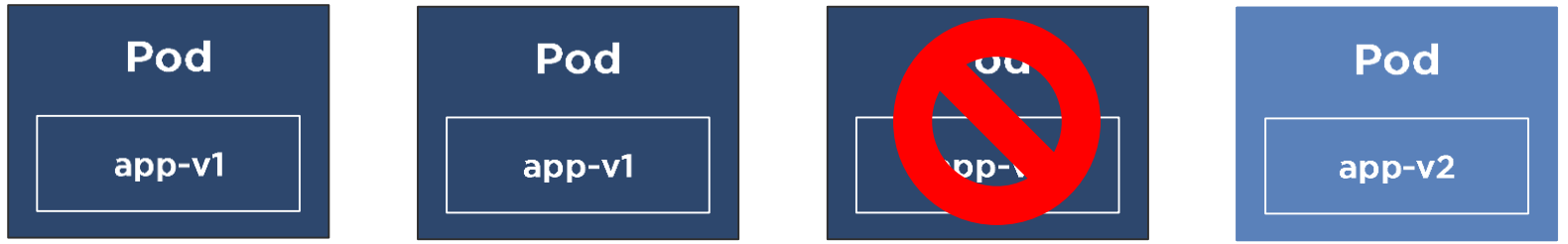
Rollout New Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

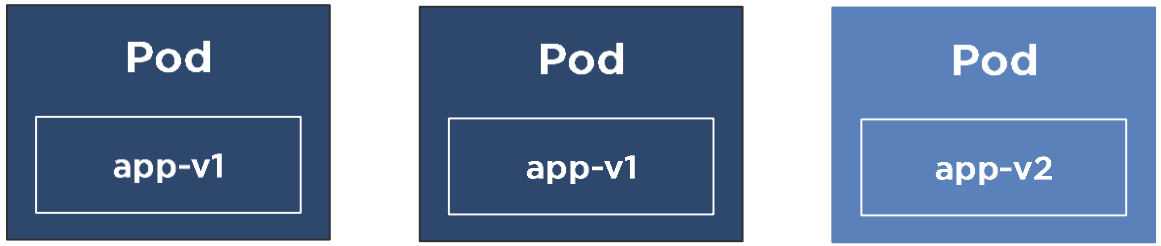
Delete Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

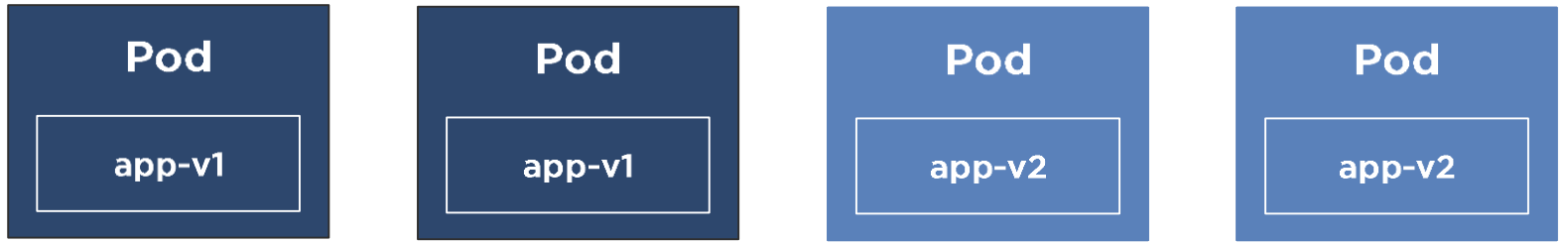
Rollout New Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

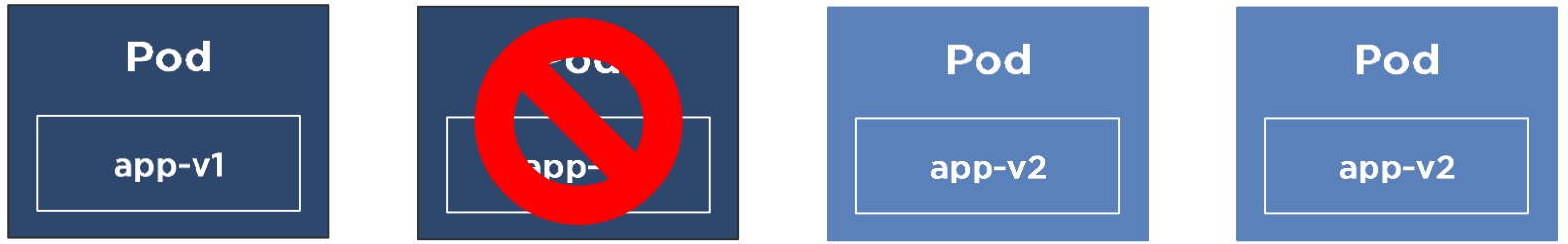
Rollout New Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

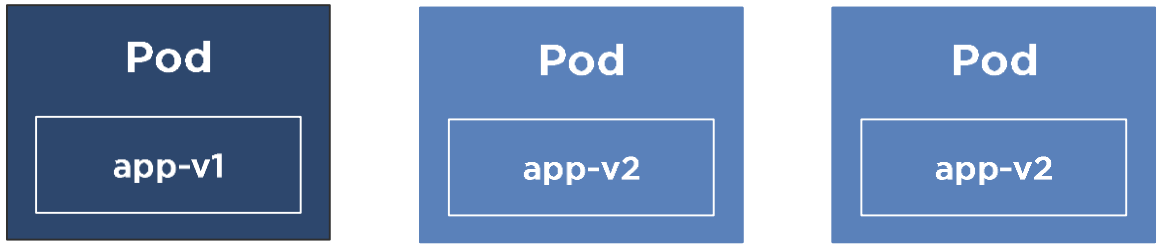
Delete Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

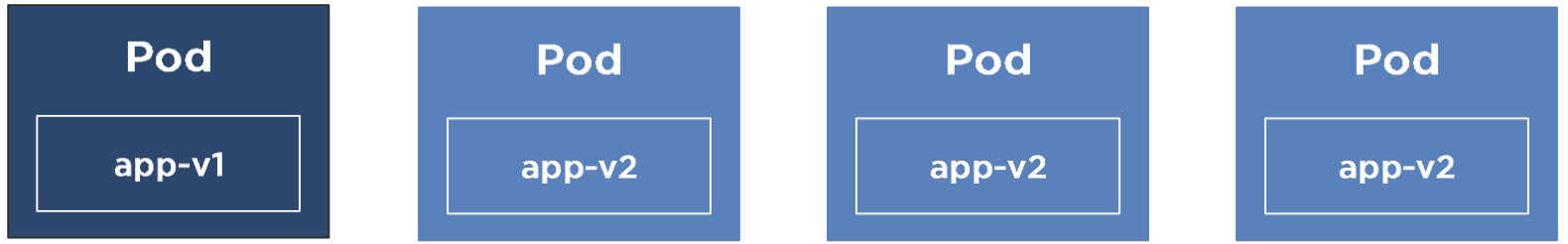
Rollout New Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

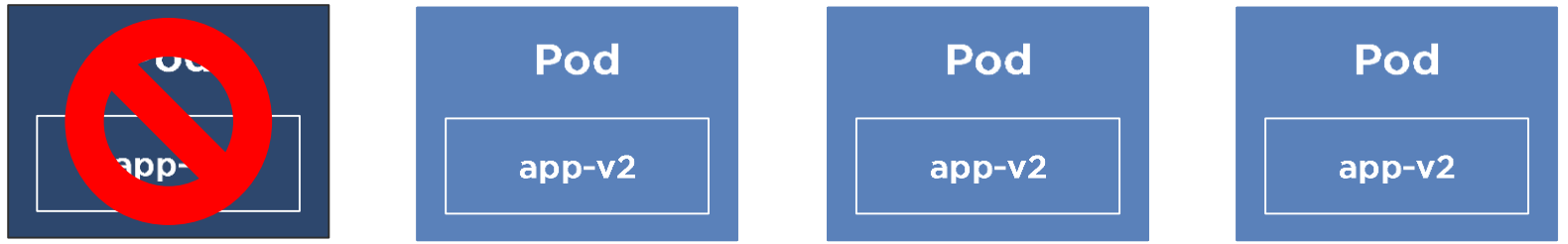
Rollout New Pod



Kubernetes – Arquitectura - Deployment

Rolling Deployment

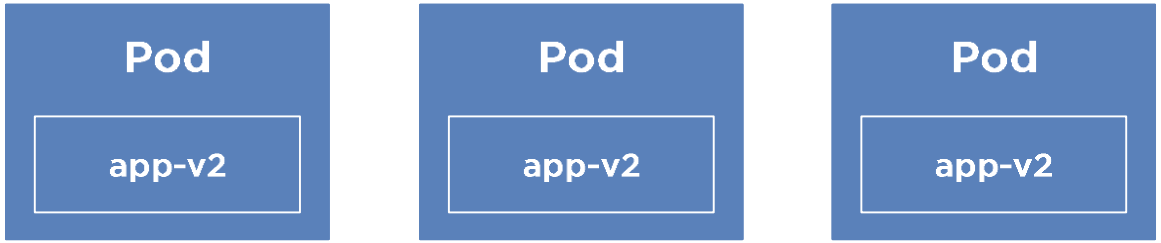
Delete Pod



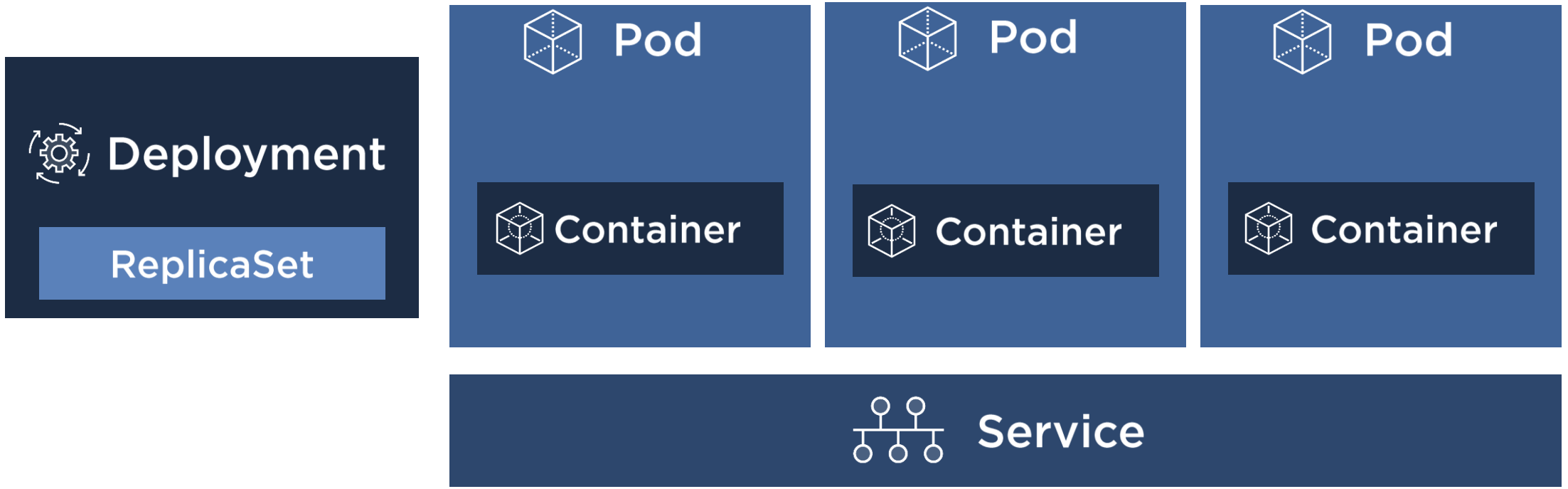
Kubernetes – Arquitectura - Deployment

Rolling Deployment

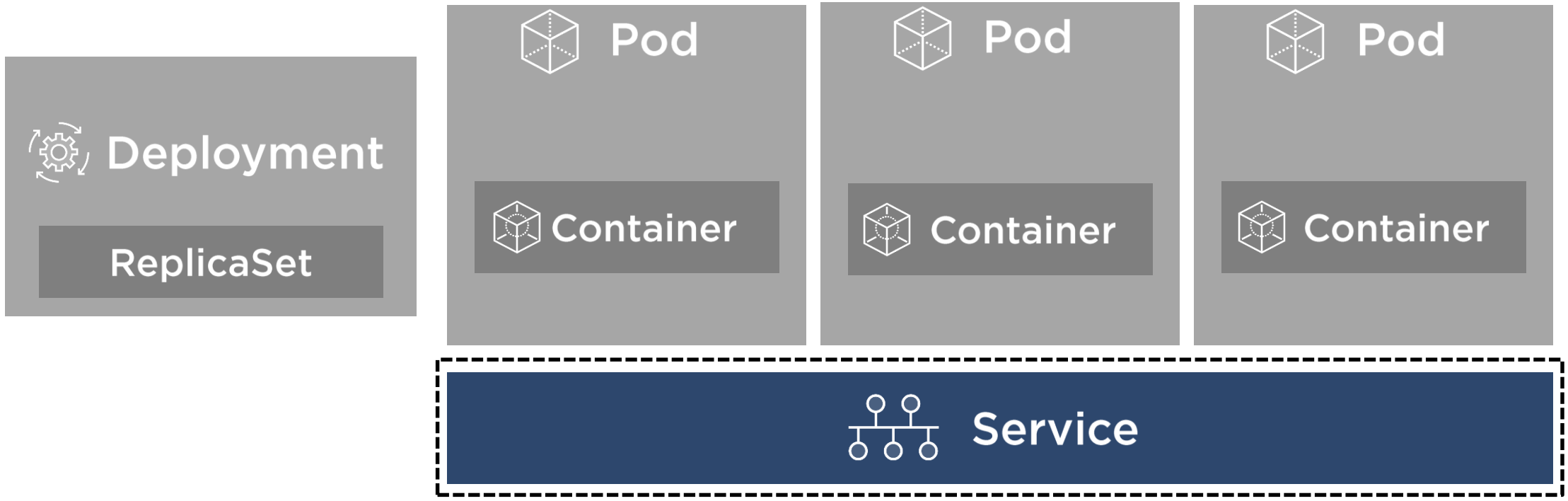
Rollout New Pod



Kubernetes - Arquitectura



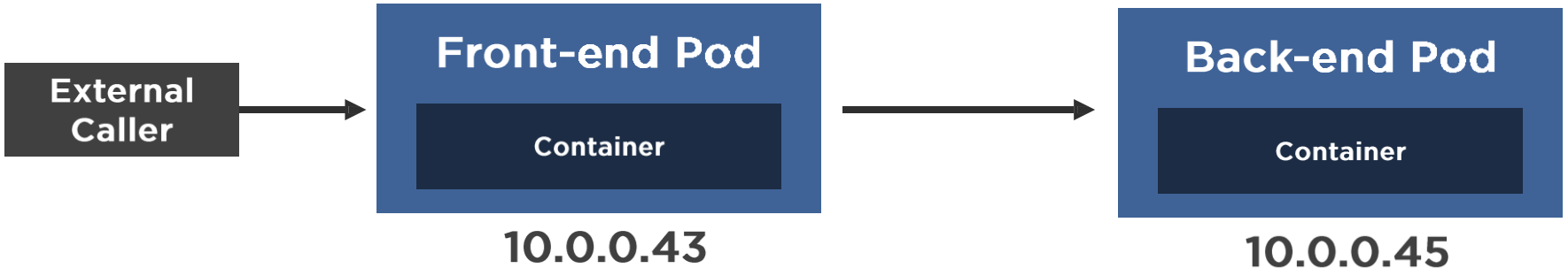
Kubernetes - Service



Un **Service** proporciona un único punto de entrada para acceder a uno o más pods

Kubernetes - Arquitectura - Service

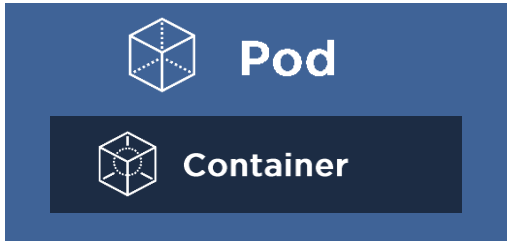
Dado que los Pods viven y mueren, ¿puedes confiar en su IP?



¡No!, Por eso necesitamos a los **Service's** : ¡las IP cambian mucho!

Kubernetes – Arquitectura - Service

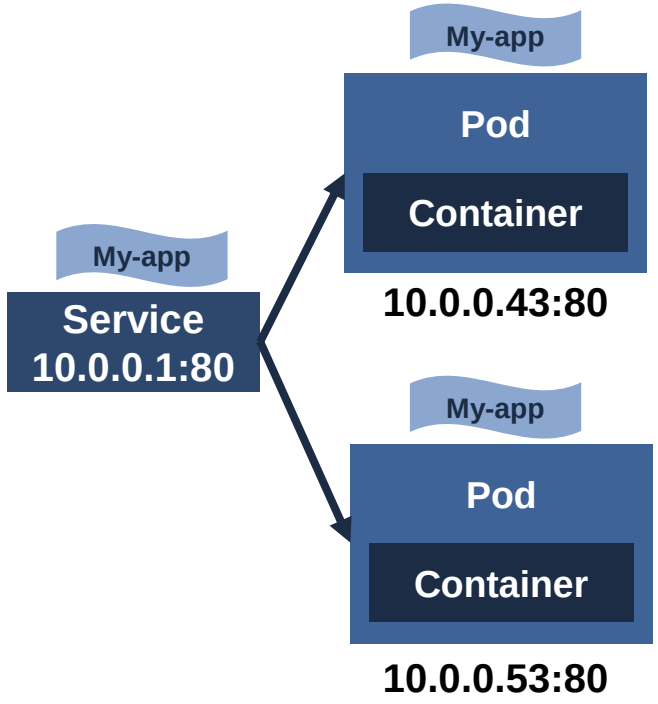
La vida de un Pod



- Los **Pod’s son "mortales"** y pueden vivir poco tiempo (efímeros)
- No puede confiar en que la dirección IP de un Pod se mantenga igual
- **Los Pod’s se pueden escalar horizontalmente**, cada Pod tiene su propia dirección IP
- **Un Pod obtiene una dirección IP después de que se ha programado** (no hay forma de que los clientes conozcan la IP antes de tiempo)

Kubernetes – Arquitectura - Service

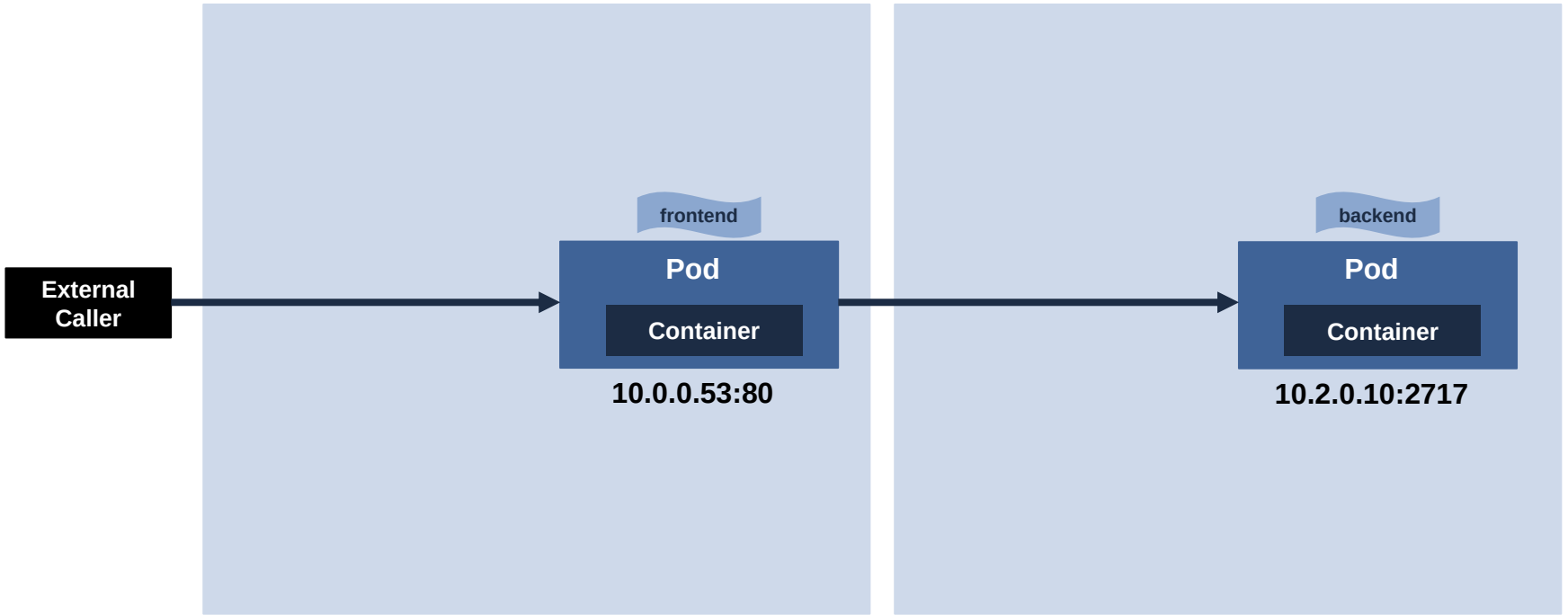
¿Qué hacen los Service's?



- **Los Service's abstraen las direcciones IP** de los pods de los consumidores.
- Load balancer entre pods.
- **Se basa en labels** para asociar un servicio con un pod
- El Kube-proxy del nodo crea una IP virtual para los Service's
- Capa 4 (TCP / UDP sobre IP)
- Los **servicios no son efímeros**
- Crea endpoint que se ubican entre un servicio y un pod

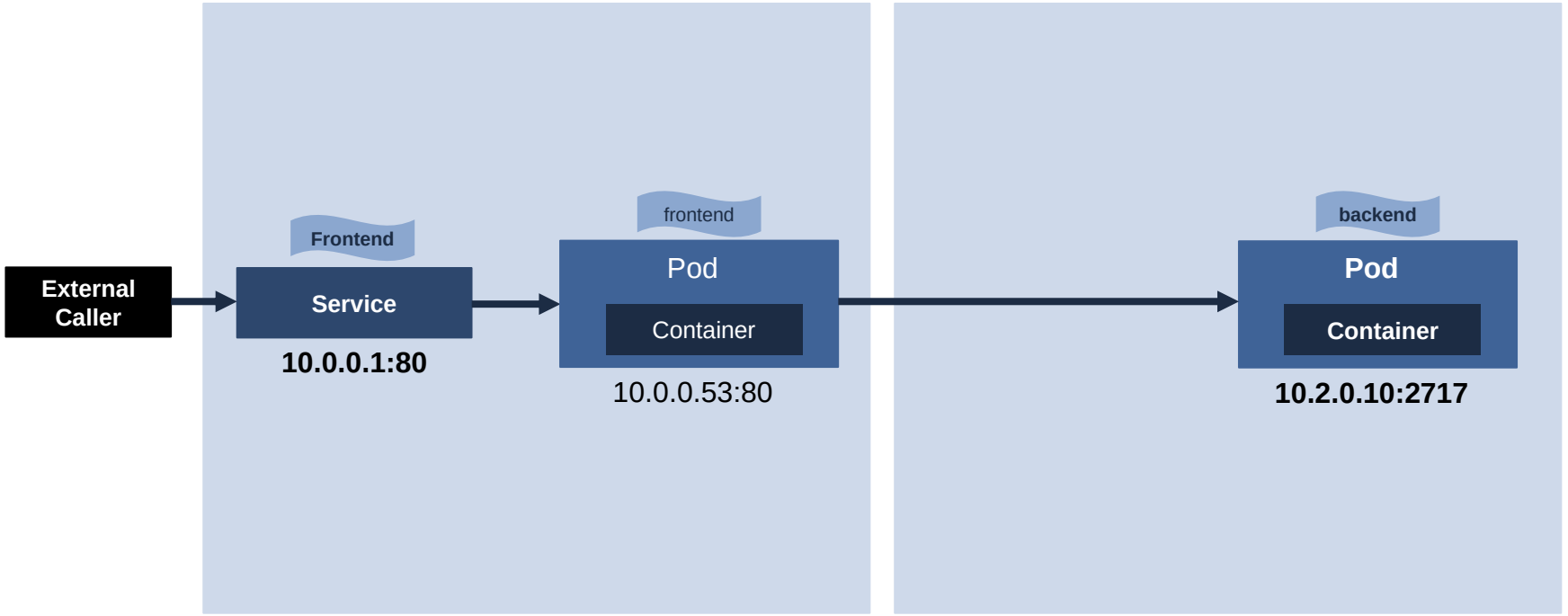
Kubernetes – Arquitectura - Service

Comunicación entre servicios



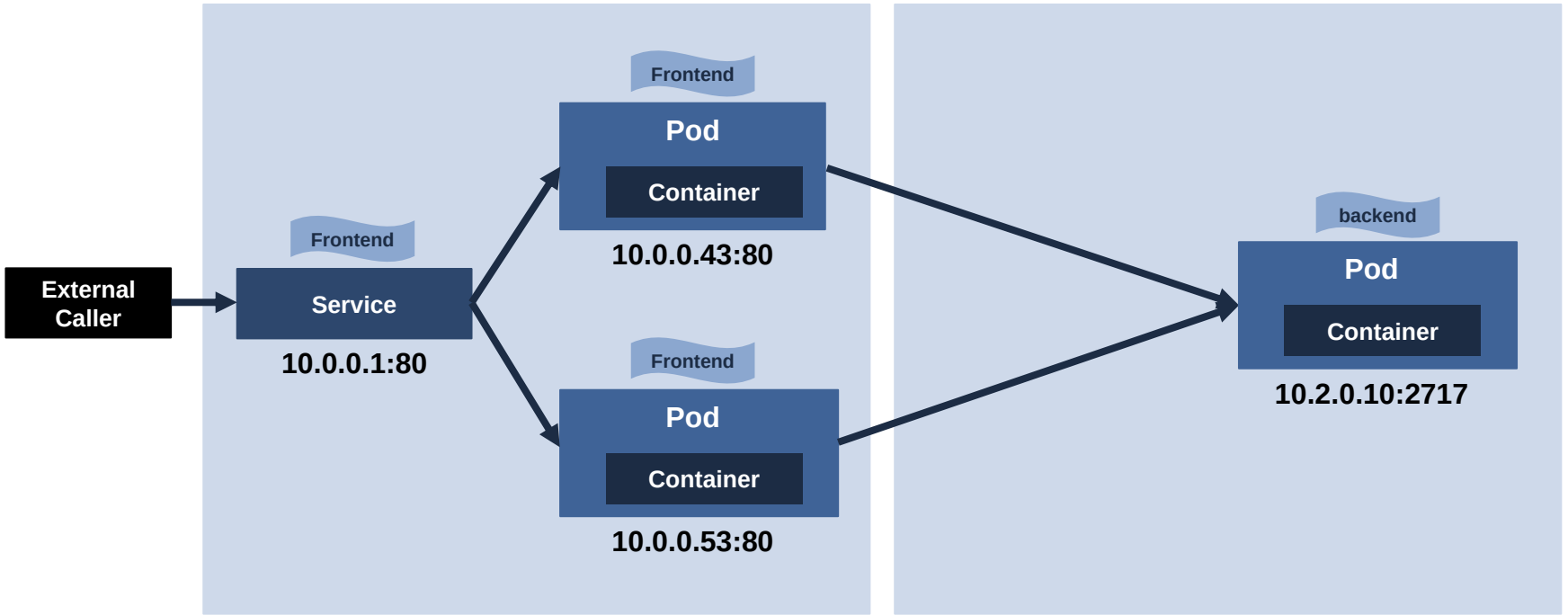
Kubernetes – Arquitectura - Service

Comunicación entre servicios



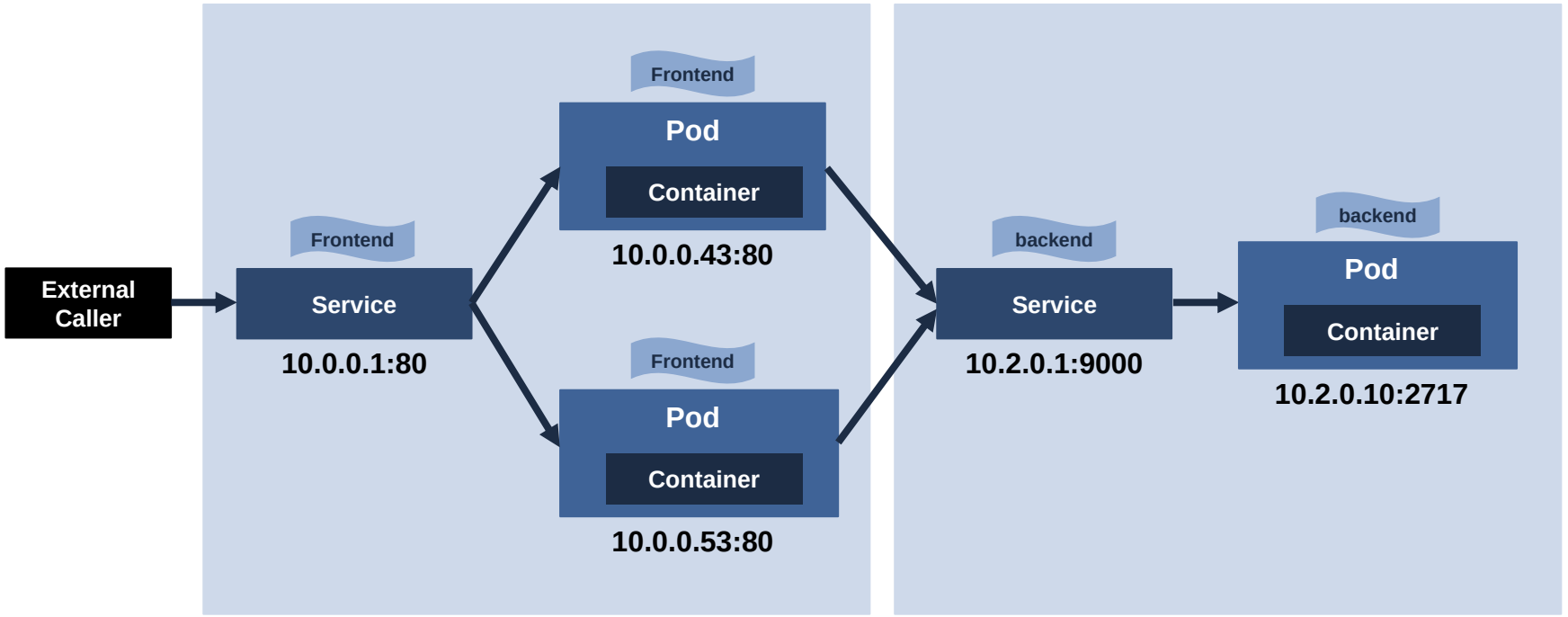
Kubernetes – Arquitectura - Service

Comunicación entre servicios



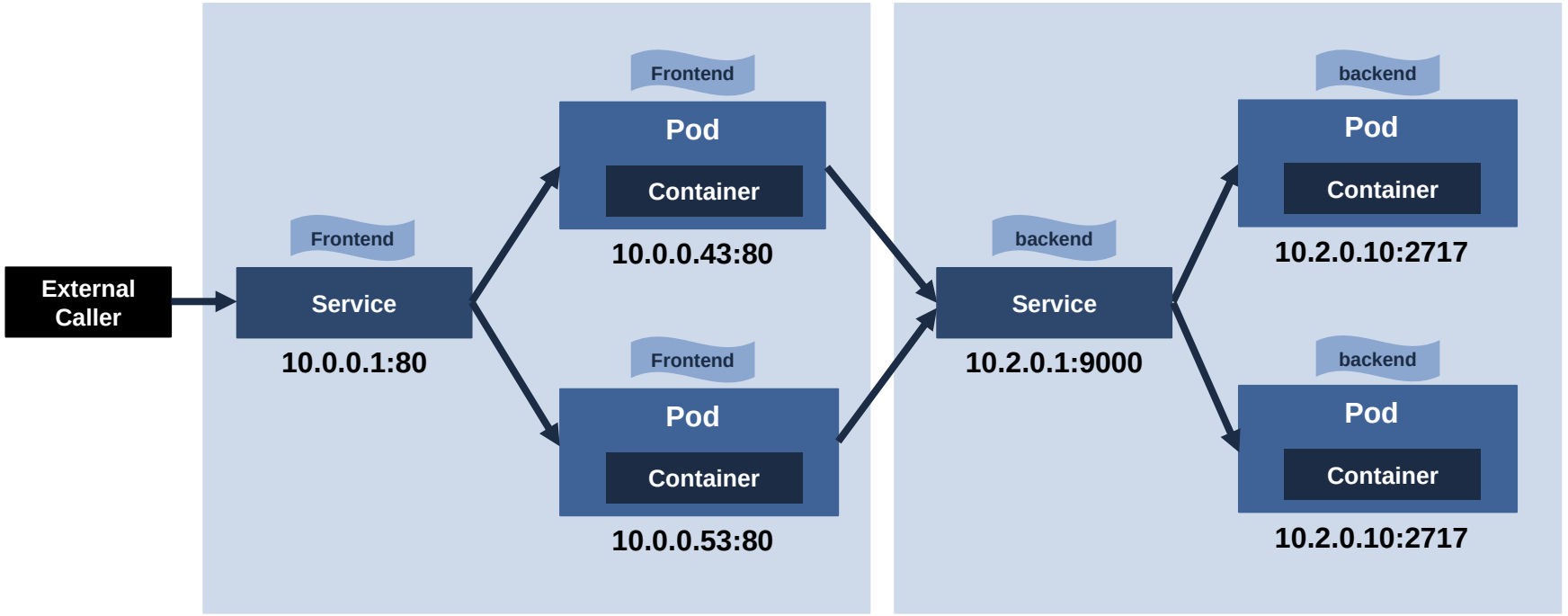
Kubernetes – Arquitectura - Service

Comunicación entre servicios



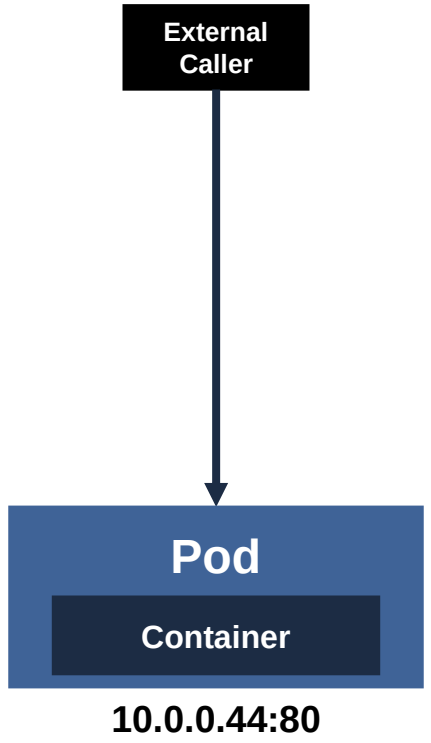
Kubernetes – Arquitectura - Service

Comunicación entre servicios



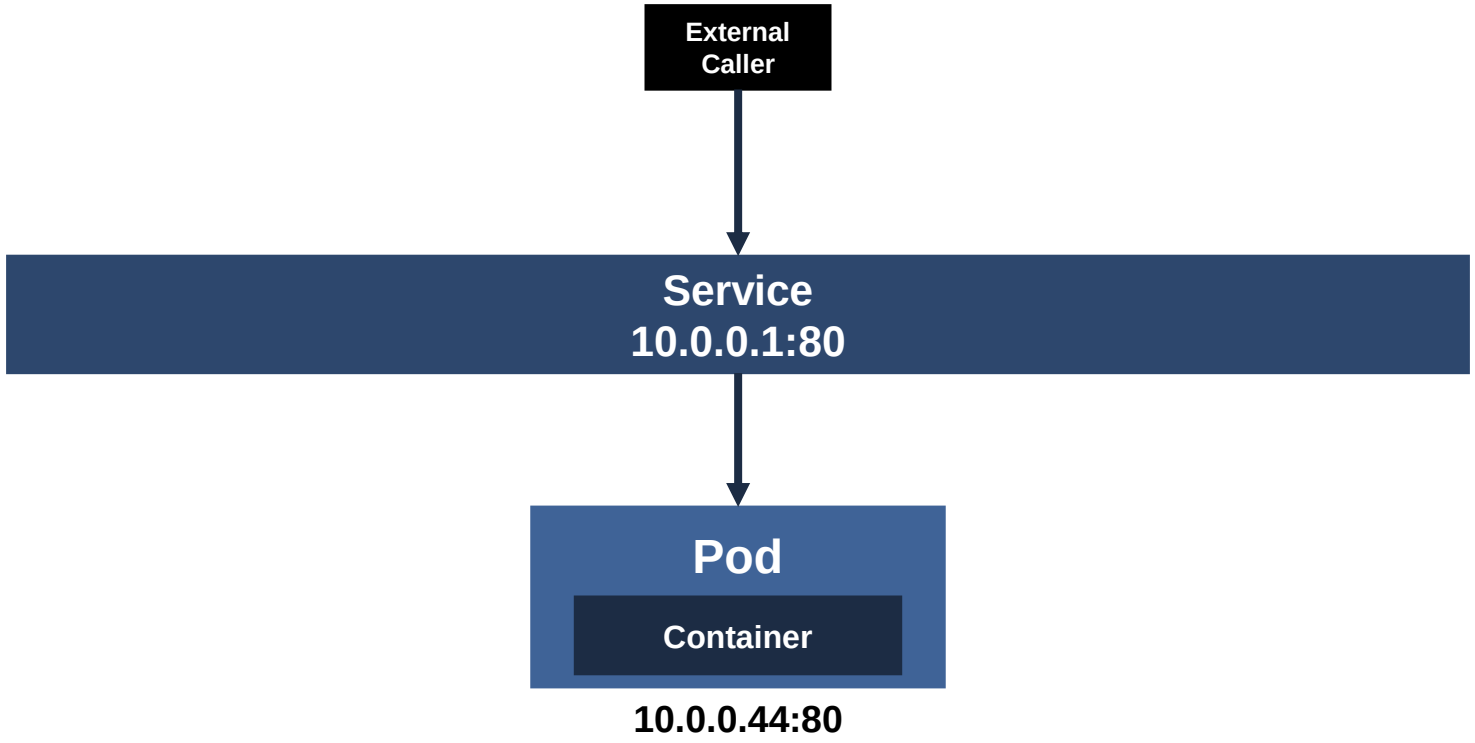
Kubernetes – Arquitectura - Service

Services y Load Balancing



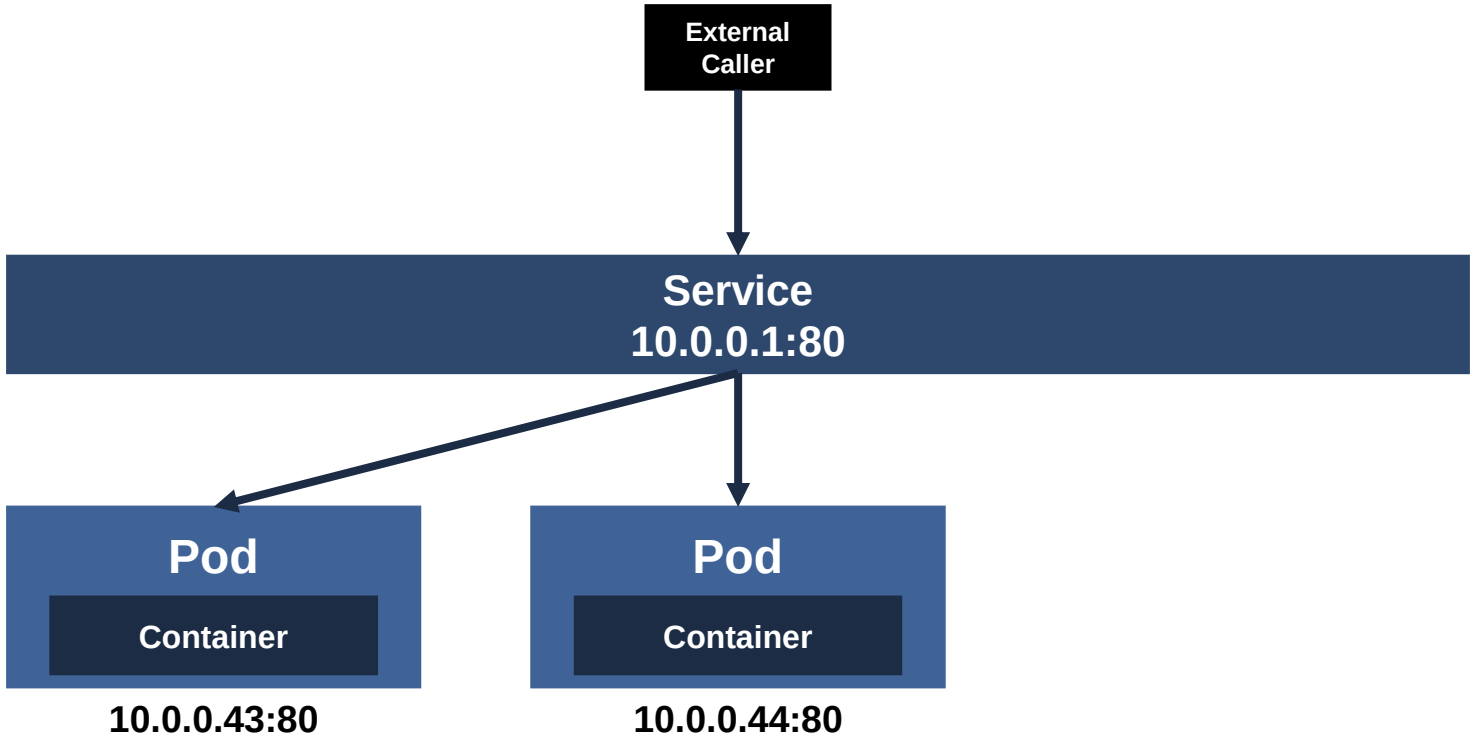
Kubernetes – Arquitectura - Service

Services y Load Balancing



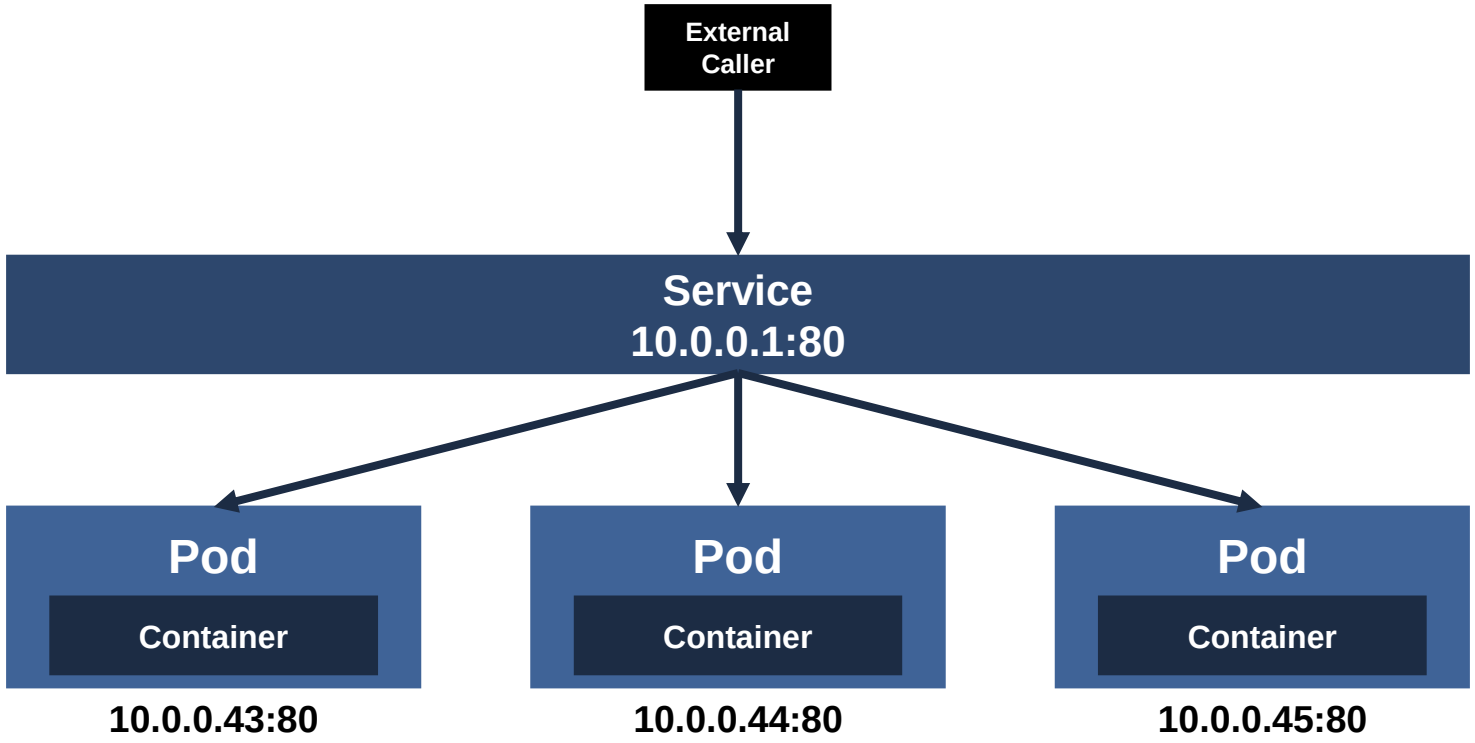
Kubernetes – Arquitectura - Service

Services y Load Balancing



Kubernetes – Arquitectura - Service

Services y Load Balancing



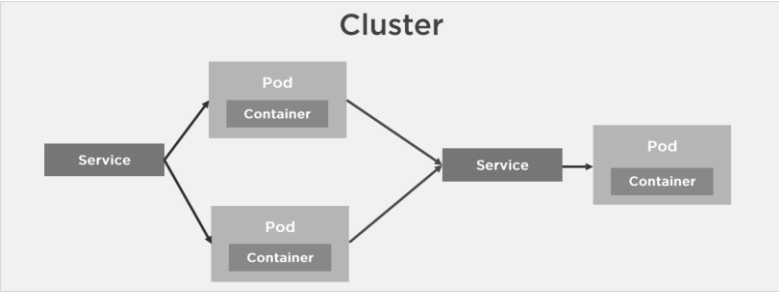
Kubernetes – Arquitectura - Service

Tipos de Service’s

Los Service’s se pueden definir de diferentes formas :

- **ClusterIP**: Expone el servicio en una IP interna del clúster (predeterminado)
- **NodePort**: Expone el servicio en la IP de cada nodo en un puerto estático
- **LoadBalancer**: Aprovisiona una IP externa para que actúe como equilibrador de carga para el servicio
- **ExternalName**: Asigna al servicio un nombre asociado al DNS

Kubernetes – Arquitectura - Service

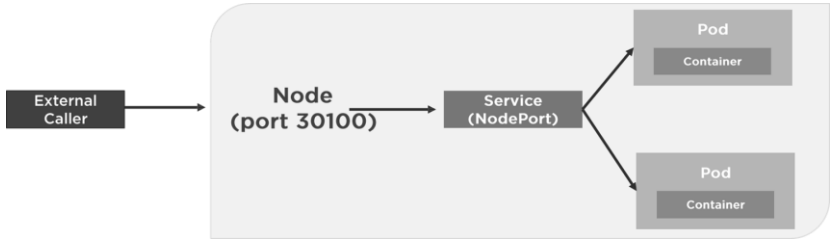


ClusterIP Service

La IP del servicio se expone internamente con el clúster, solo los pods dentro del clúster pueden comunicarse con el servicio.

Permite que los pods se comuniquen con otros pods.

Kubernetes – Arquitectura - Service



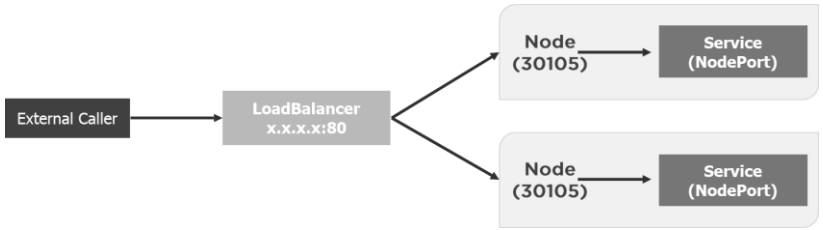
NodePort Service

Expone el servicio en la IP de cada nodo en un puerto estático.

Asigna un puerto de un rango (el valor predeterminado es 30000 - 32767).

Cada nodo hace proxy del puerto asignado.

Kubernetes – Arquitectura - Service



LoadBalancer Service

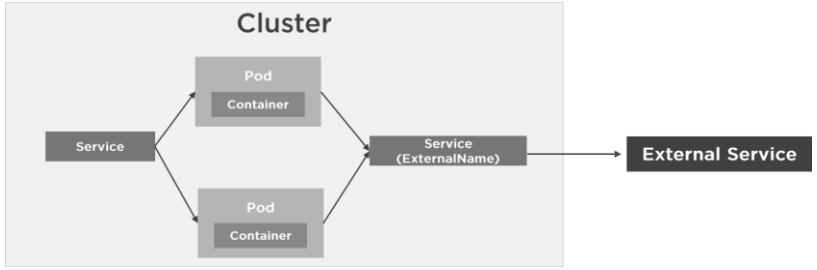
Expone un Servicio externamente.

Útil cuando se combina con un equilibrador de carga de un proveedor de nube.

Se crean los servicios NodePort y ClusterIP.

Cada nodo hace proxy del puerto asignado.

Kubernetes – Arquitectura - Service



ExternalName Service

Servicio que actúa como alias de un servicio externo.

Permite que un servicio actúe como proxy de un servicio externo.

Los detalles del servicio externo están ocultos del clúster (más fácil de cambiar)

03

GALAXY
TRAINING

Explorando la Arquitectura Local Kubernetes, Azure Kubernetes y Google Kubernetes.

DEMO

Explorando la Arquitectura

- Local Kubernetes
- Azure Kubernetes
- Google Kubernetes.

GALAXY
TRAINING

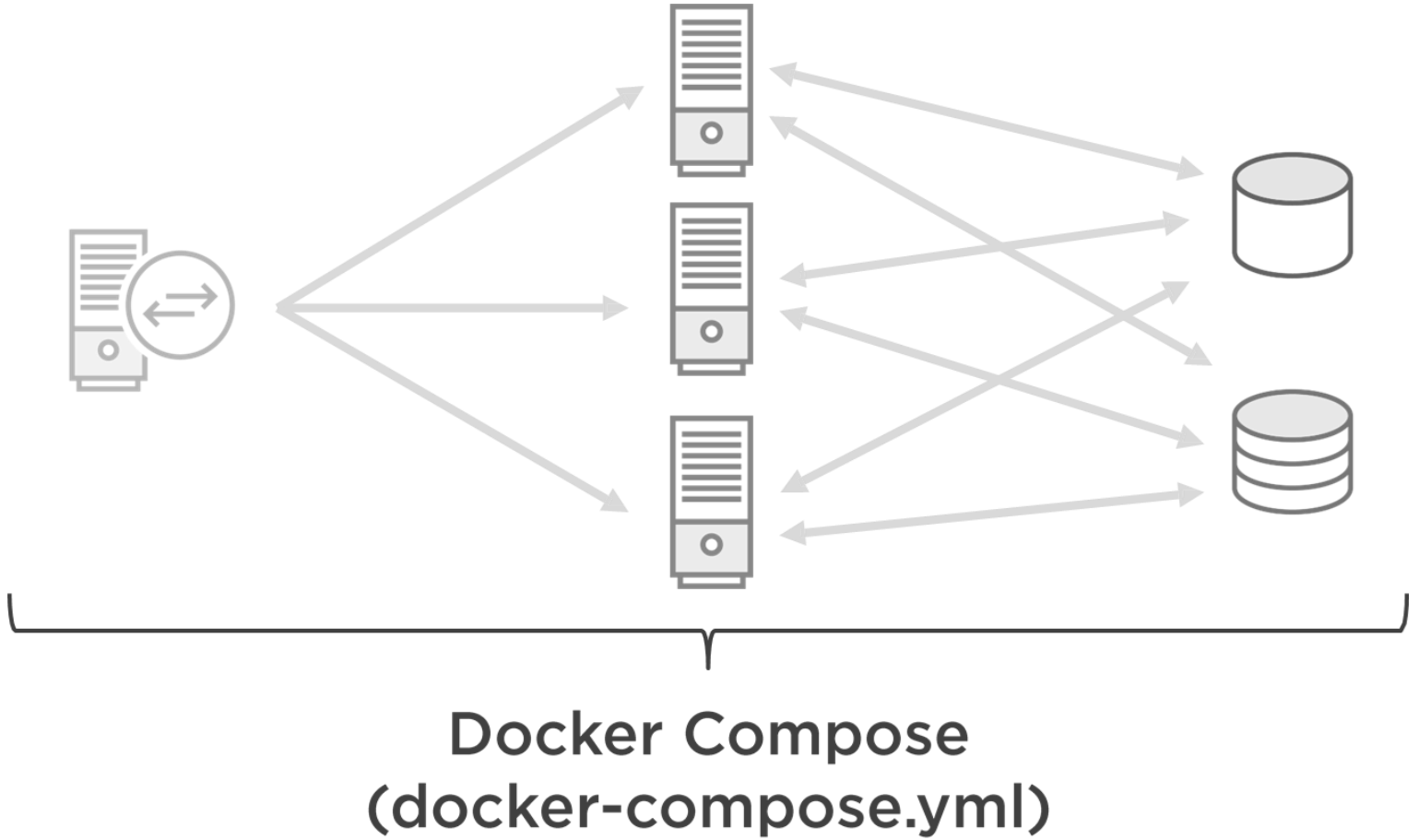
04

GALAXY TRAINING

Generando archivos YAML.

Generando archivos YAML

Usando Docker Compose

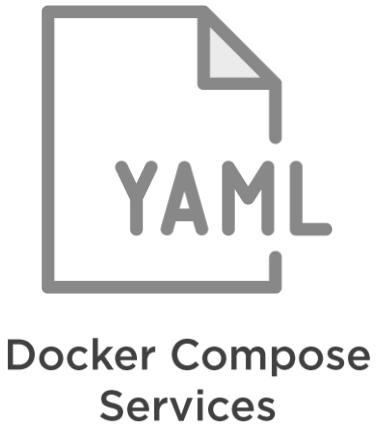


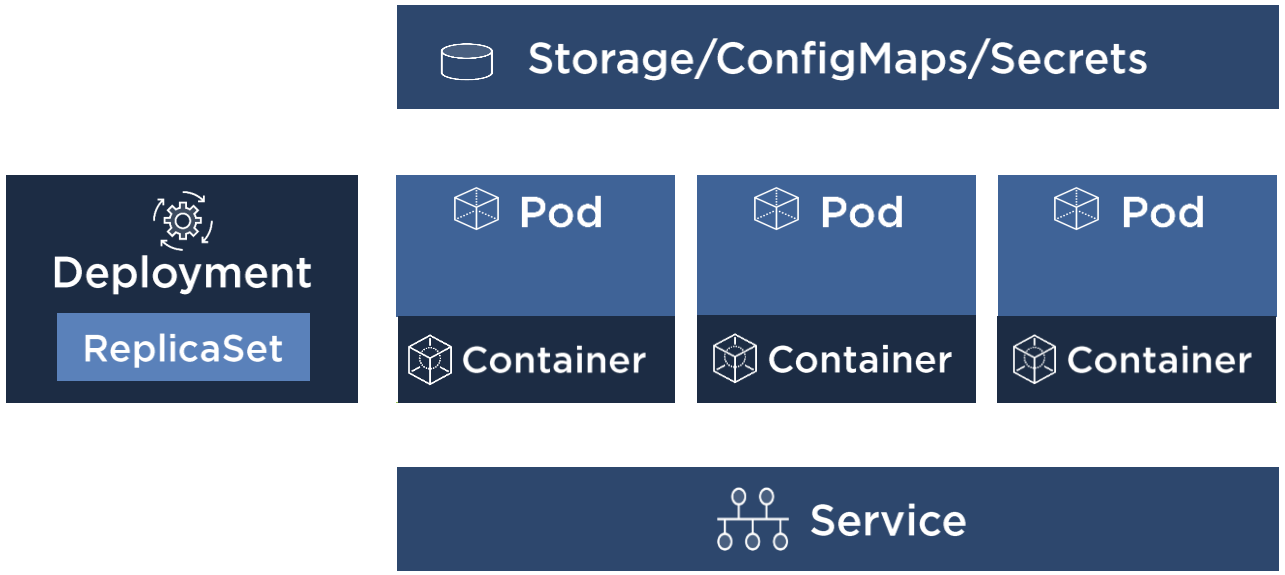
Generando archivos YAML

Docker Compose

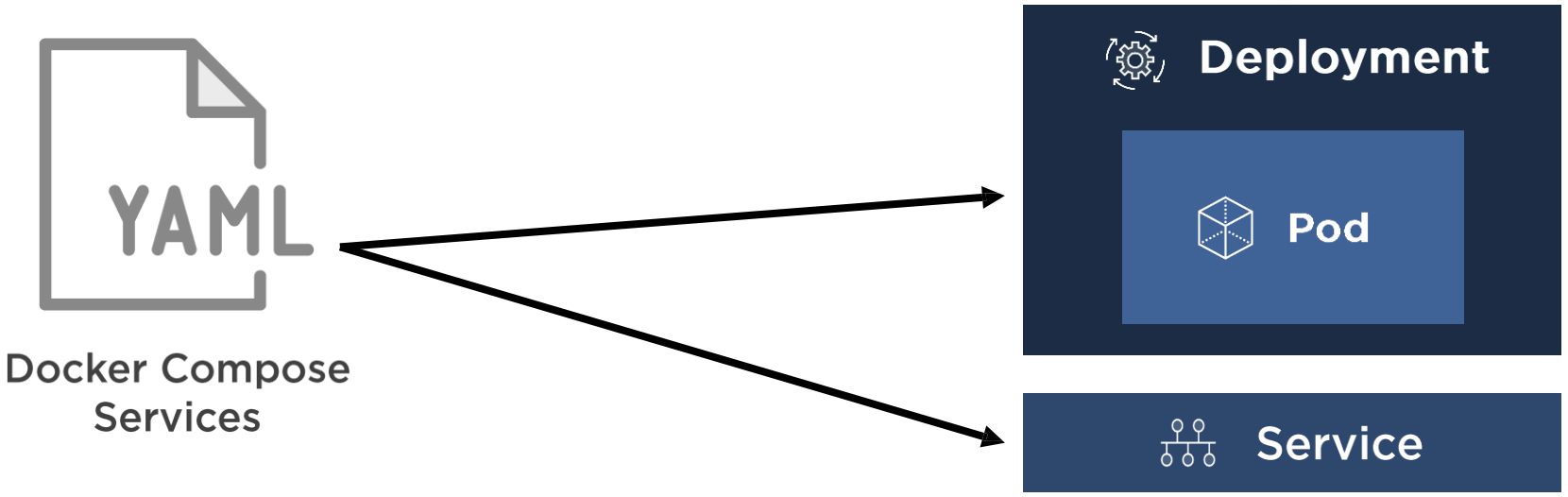
Docker Compose y Kubernetes

Docker Compose y Kubernetes





Docker Compose y Kubernetes



Mapeando Docker Compose a Kubernetes

```
version: "3.x"
services:
    node:
      container_name: nodeapp
      image: yourAccount/nodeapp
      ports:
        - "80:80"
      volumes:
        - .:/var/www/logs
      env_file:
      - ./nodeapp.env
      networks:
        - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```

Mapeando Docker Compose a Kubernetes

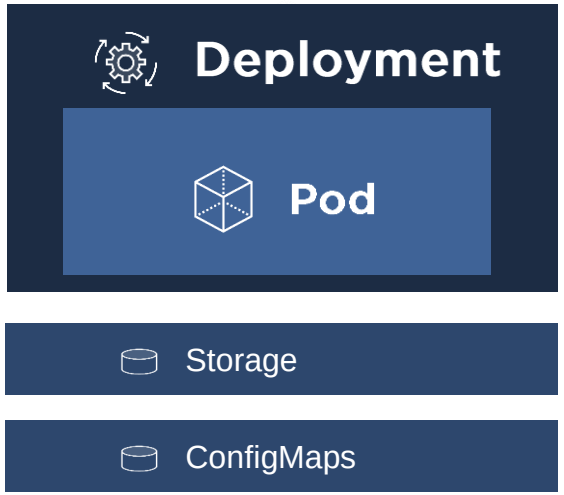
```
version: "3.x"
services:
    node:
      container_name: nodeapp
      image: yourAccount/nodeapp
      ports:
        - "80:80"
      volumes:
        - .:/var/www/logs
      env_file:
      - ./nodeapp.env
      networks:
        - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```

Mapeando Docker Compose a Kubernetes

```
version: "3.x"
services:
    node:
      container_name: nodeapp
      image: yourAccount/nodeapp
      ports:
        - "80:80"
      volumes:
        - .:/var/www/logs
      env_file:
      - ./nodeapp.env
      networks:
        - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```

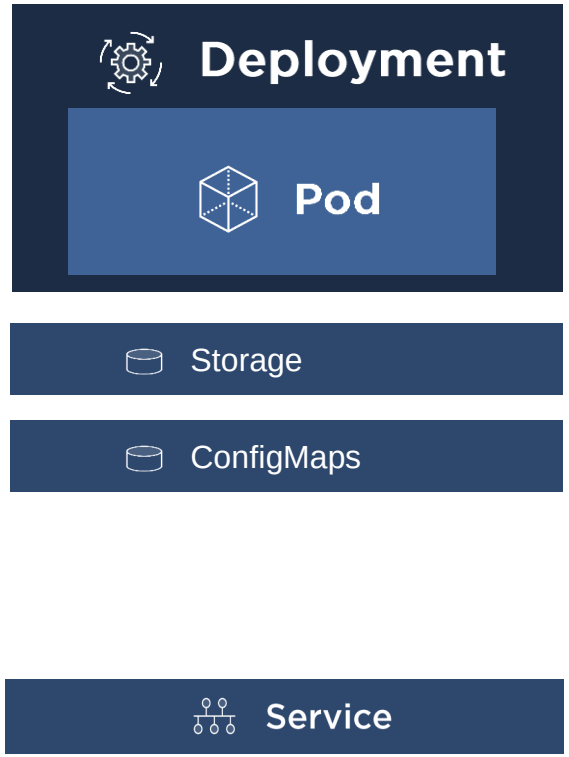
Mapeando Docker Compose a Kubernetes

```
version: "3.x"
services:
    node:
      container_name: nodeapp
      image: yourAccount/nodeapp
      ports:
        - "80:80"
      volumes:
        - .:/var/www/logs
      env_file:
      - ./nodeapp.env
      networks:
        - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```



Mapeando Docker Compose a Kubernetes

```
version: "3.x"
services:
    node:
      container_name: nodeapp
      image: yourAccount/nodeapp
      ports:
        - "80:80"
      volumes:
        - .:/var/www/logs
      env_file:
      - ./nodeapp.env
      networks:
        - nodeapp-network

networks:
  nodeapp-network
    driver: bridge
```



Generando archivos YAML

Kubernetes + Compose = **Kompose**

GRACIAS

POR SU PREFERENCIA

GALAXY
TRAINING