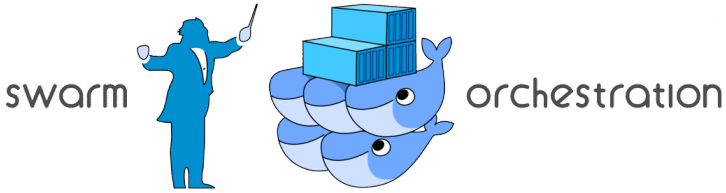
Orquestación de Contenedores

Computación en la Nube

Oscar H. Mondragón

Google Container Engine
(GKE)

Google Container Registry



kubernetes
by Google

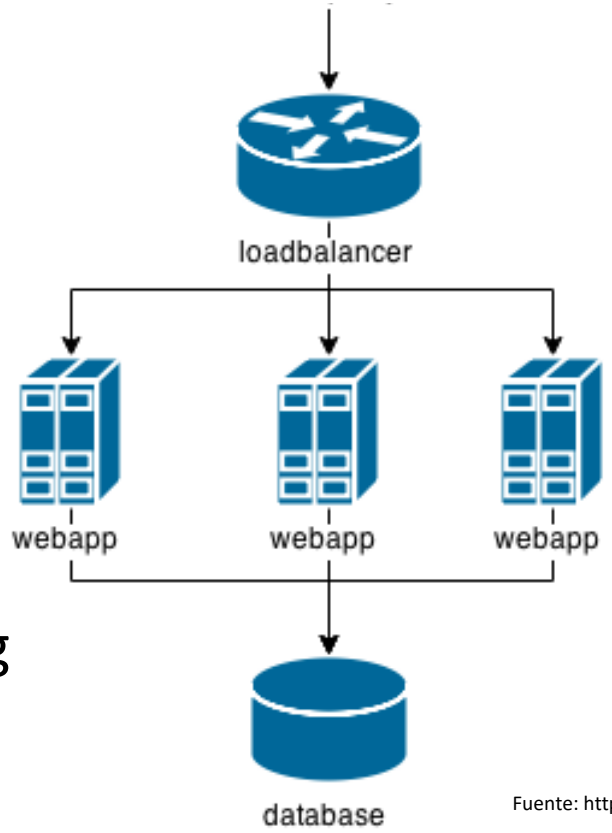
Contexto

Scheduling

Load Balancing

Autoscaling

Health Monitoring



Cluster Management

Service Discovery

Rules and Constraints

Networking

...

Fuente: https://eyenx.ch/2015/04/18/loadbalancing-containers-with-docker-compose

Blog Interesante sobre Service Discovery

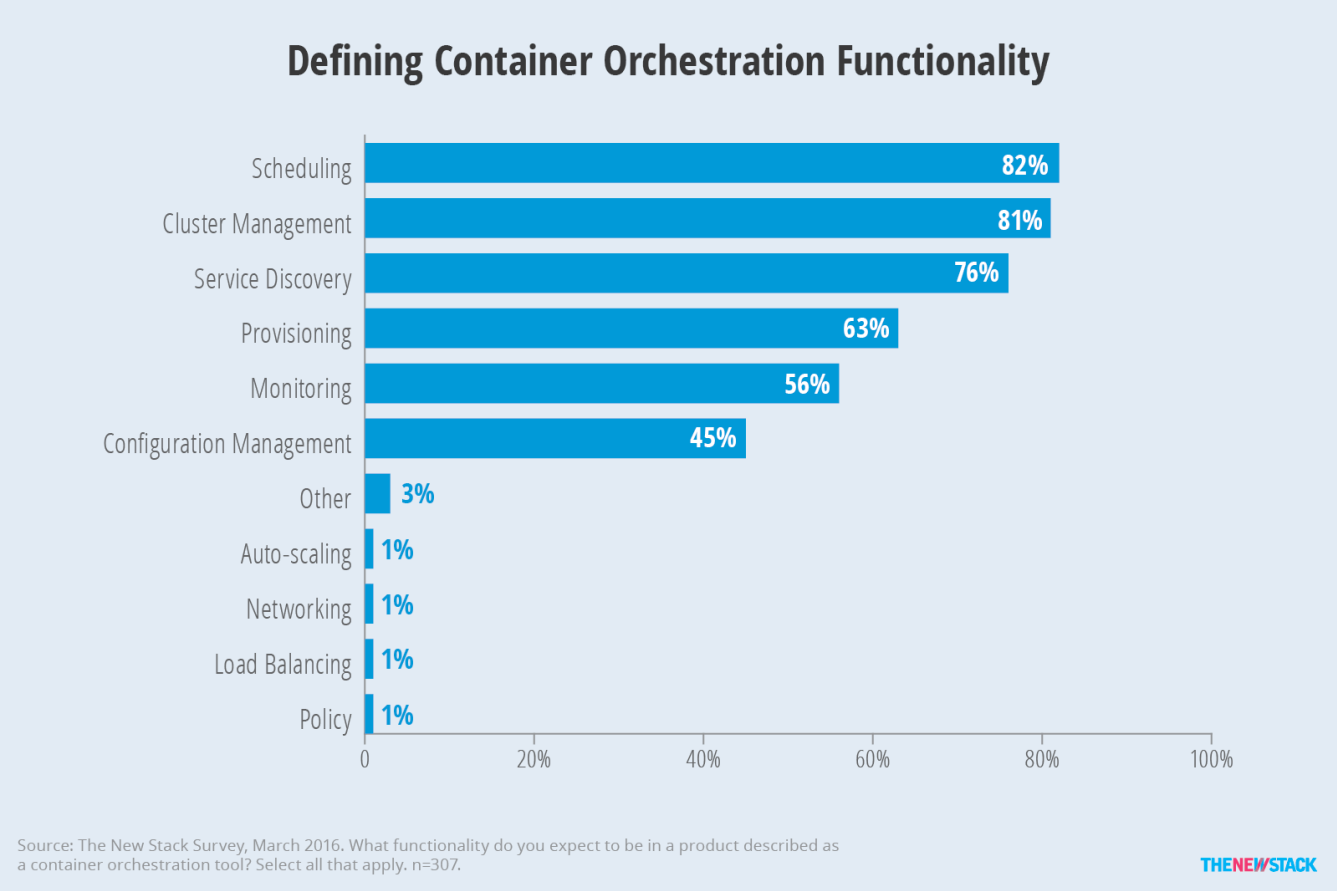
https://www.oscarblancarteblog.com/2018/09/24/service-discovery-pattern-para-microservicios/

Orquestación de Contenedores

- Las herramientas de orquestación extienden las capacidades de administración del ciclo de vida de contenedores a escenarios complejos de aplicaciones desplegadas en clústeres de maquinas

- Abstraen la infraestructura de hosts y permiten a los usuarios tratar un clúster completo como si fuera una sola maquina

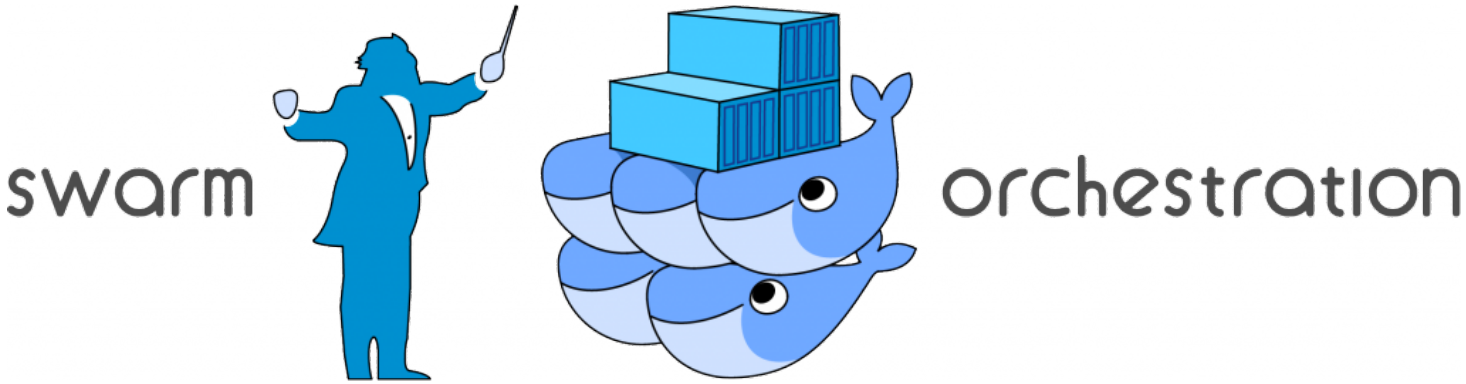
Funciones de una plataforma de orquestación



Fuente: https://thenewstack.io/containers-container-orchestration/

Plataformas de Orquestación de Contenedores

Docker Swarm

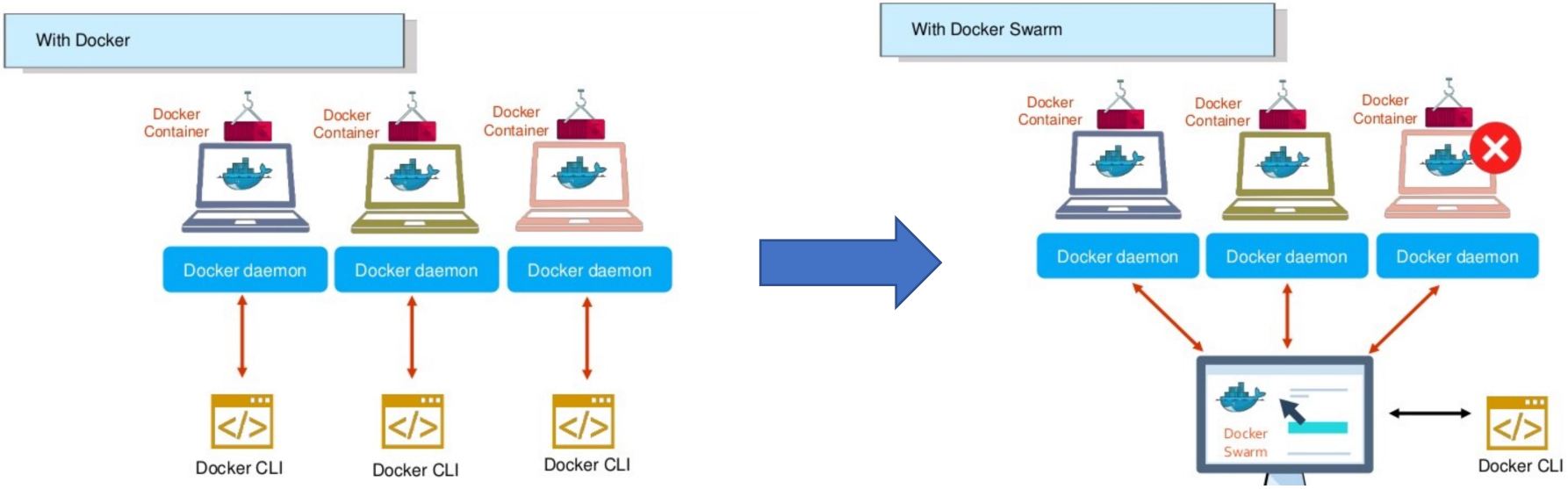


Docker Swarm

- Docker Swarm es un servicio que permite a los usuarios crear y administrar un cluster de nodos de Docker y orquestar contenedores

- Cada nodo de Docker Swarm es un Docker daemon y todos los docker daemos interactuan usando el Docker API

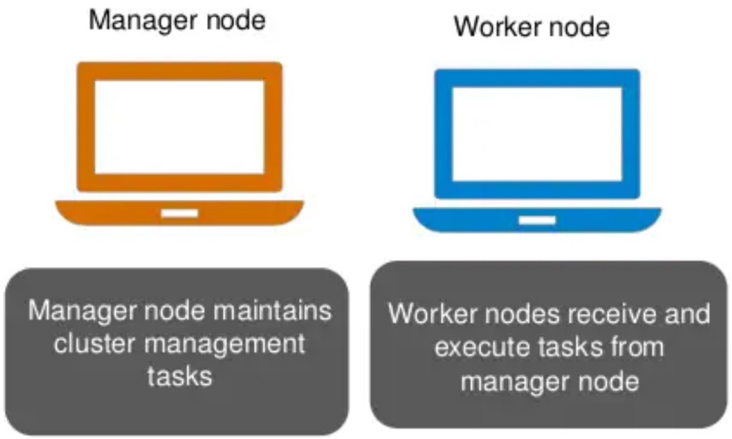
Docker vs. Docker Swarm



Servicios en Docker Swarm

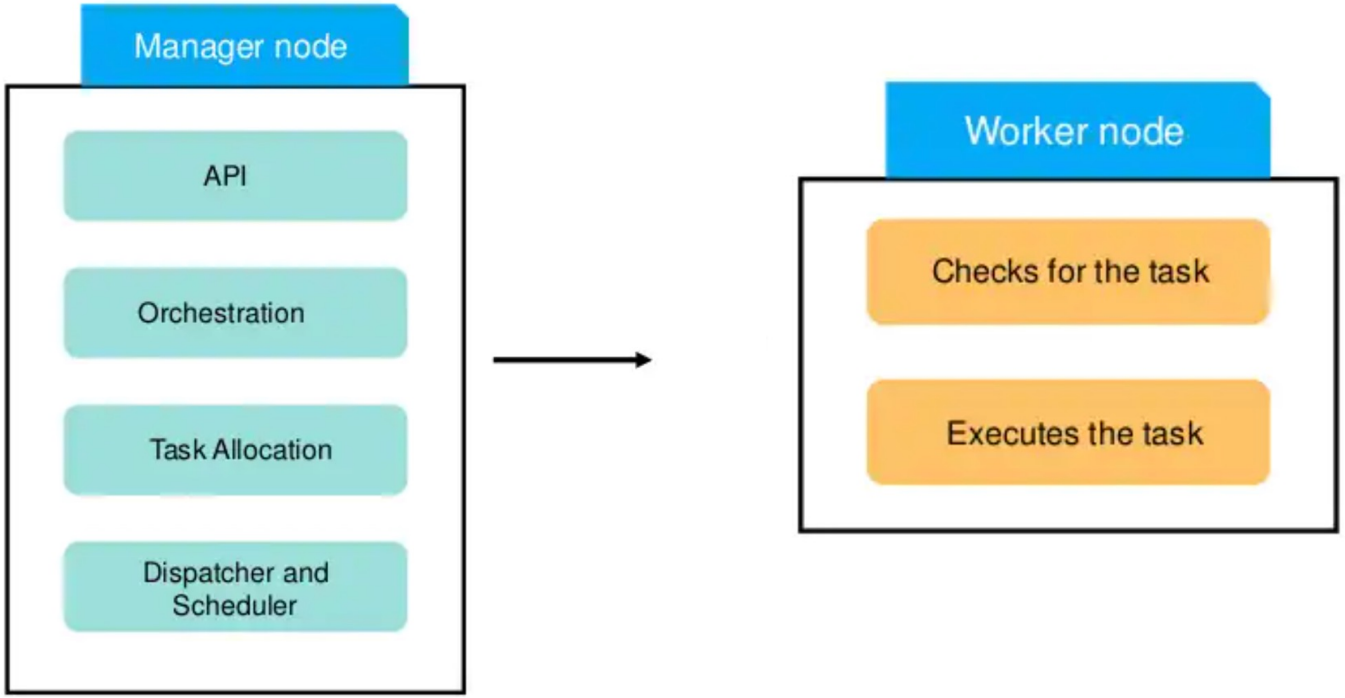
- En Docker Swarm los contenedores son lanzados usando servicios
- Un servicio es un grupo de containers de la misma imagen
- Los servicios permiten escalar la aplicación
- Antes de desplegar un servicio en Docker Swarm debe tener al menos un nodo desplegado

Tipos de Nodos



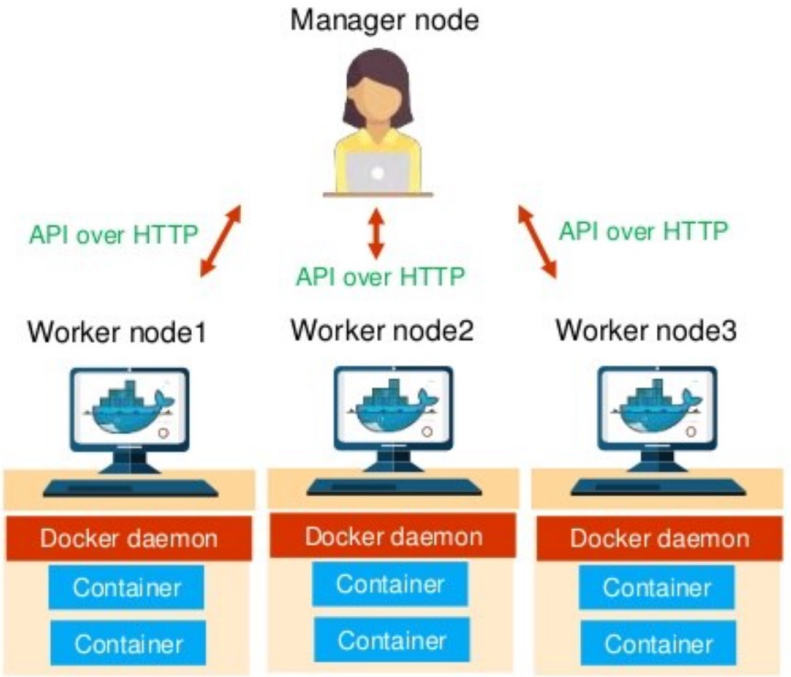
- El nodo manager conoce el estado de todos los workers en el cluster
- Cada worker tiene un agente que reporta el estado del las tareas del nodo al manager
- Los nodos workers aceptan tareas desde el manager

Tipos de Nodos



Comunicación Manager - Workers

Los workers se comunican con el manager usando un API HTTP



Tipos de Servicios

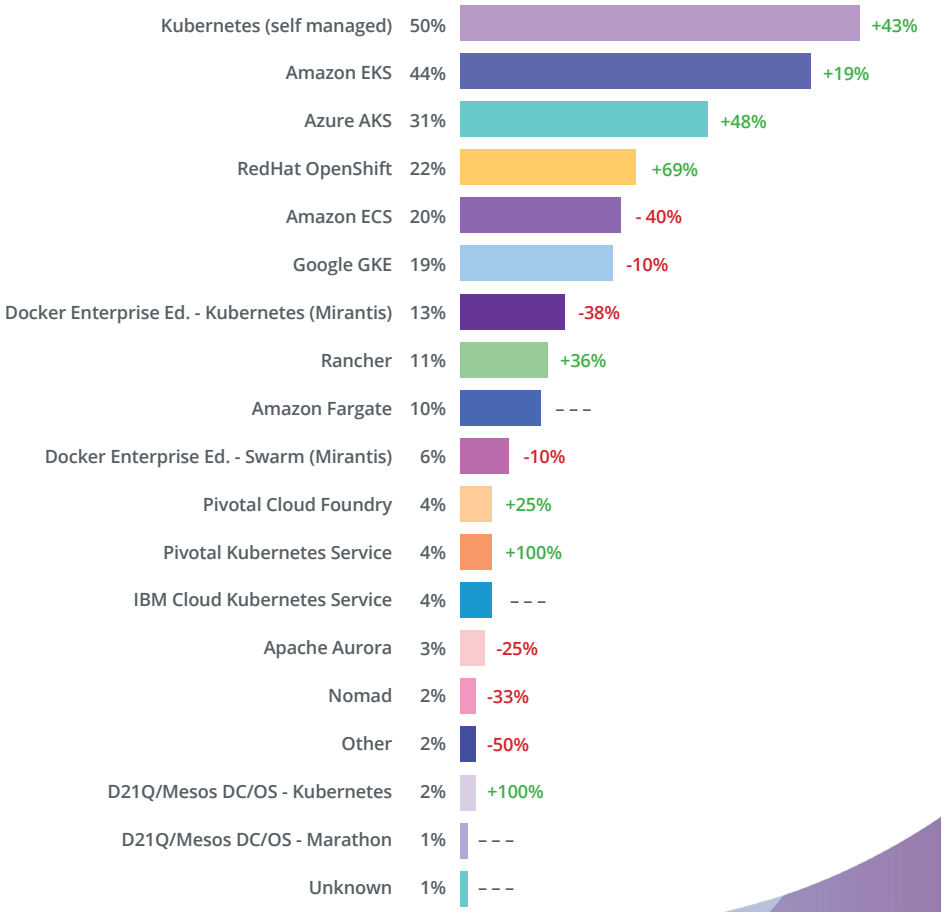
- Los servicios pueden ser desplegados y accedidos por cualquier node del mismo cluster
- Al crear un servicio, el usuario debe especificar que imagen de container usar
- Los servicios pueden ser globales o replicados
- Lo servicios globales corren en cada nodo Swarm
- En un servicio replicado, el manager distribuye las tareas entre los workers

kubernetes

by Google

Por que Kubernetes (2020)?

What do you use to orchestrate your containers? (pick all that apply)



Fuente: https://www.stackrox.com/kubernetes-adoption-security-and-market-share-for-containers/

Autoscaling Load Balancing Remote Storage Service Discovery

kubernetes by Google

Bare Metal

Fuente: https://kubernetes.io/

Overview

- Plataforma abierta para despliegue, escalamiento y administración automática para clusters de contenedores de aplicaciones
- Diseñada originalmente por Google y donada a Cloud Native Computing Foundation
- Kubernetes v1.0 liberada en Julio 21 de 2015
- Version actual: Kubernetes v1.19

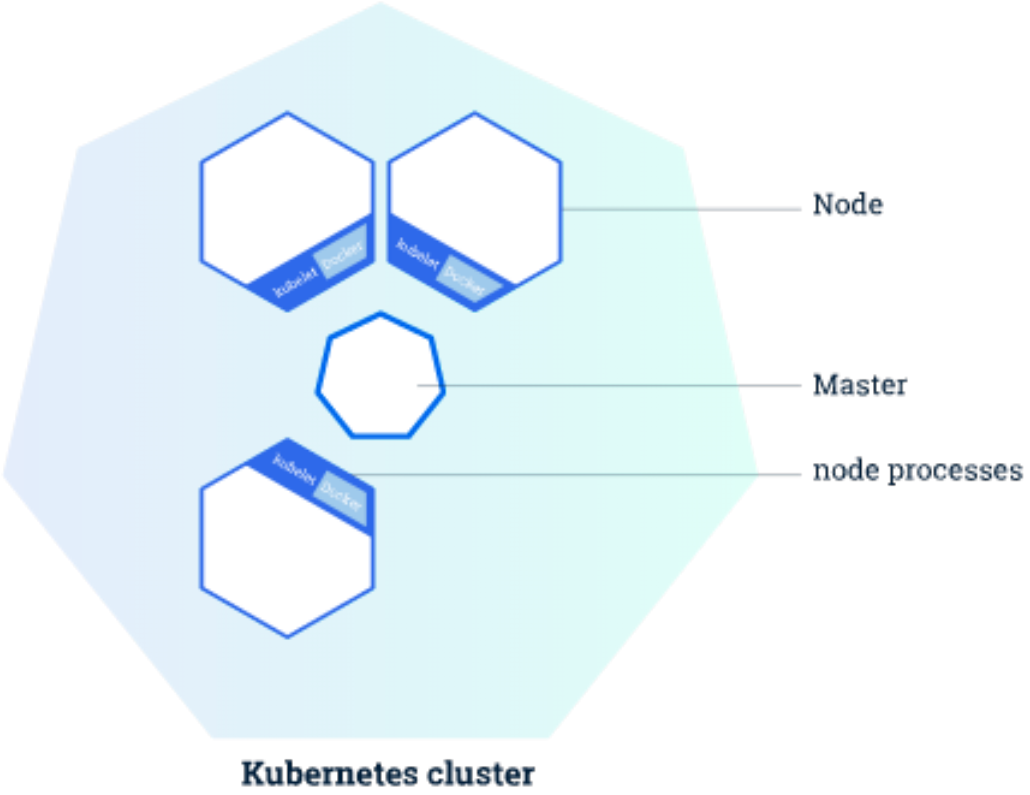
Clusters

- Kubernetes permite las coordinación de clusters de computadores altamente disponibles que funcionan como una unidad

- Automatiza la distribución y scheduling de contenedores de aplicación a través de clusters de una manera eficiente.

Dos tipos de recursos en clusters

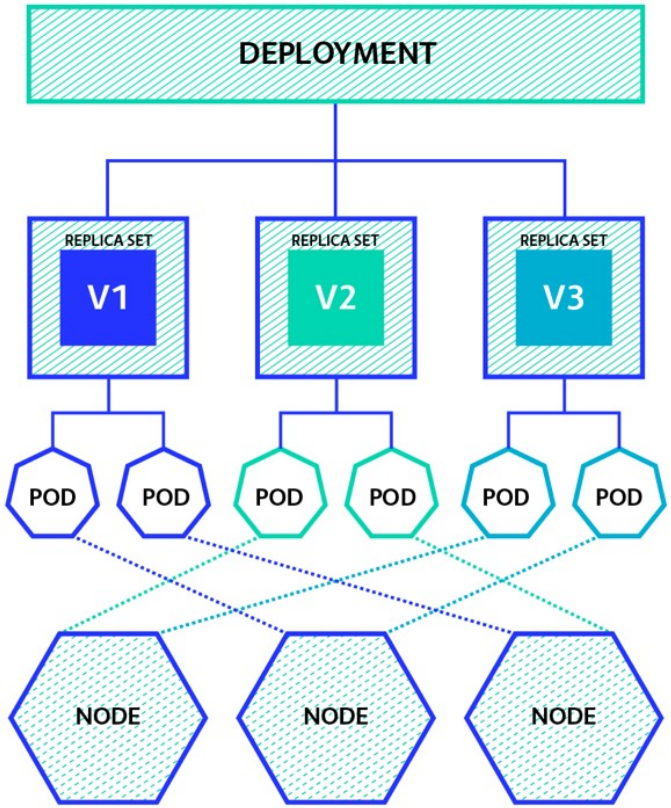
- Master: coordina el cluster
- Nodos: “workers” que corren aplicaciones
- Los nodos se comunican con el master usando el API de Kubernetes



Deployments (Despliegues)

- Una vez se tenga un cluster kubernetes funcional se pueden desplegar aplicaciones containerizadas en él
- Para esto, un Kubernetes Deployment (KD) permite crear y actualizar instancias de la aplicación
- Después de crear el Deployment, el Kubernetes Master (KM) realiza tareas de asignación de tareas de las aplicaciones a los nodos en el clúster
- El Kubernetes Deployment Controller (KDC) monitorea continuamente las instancias, si un nodo se cae, reemplaza la instancia que corría en él.

Deployments (Despliegues)



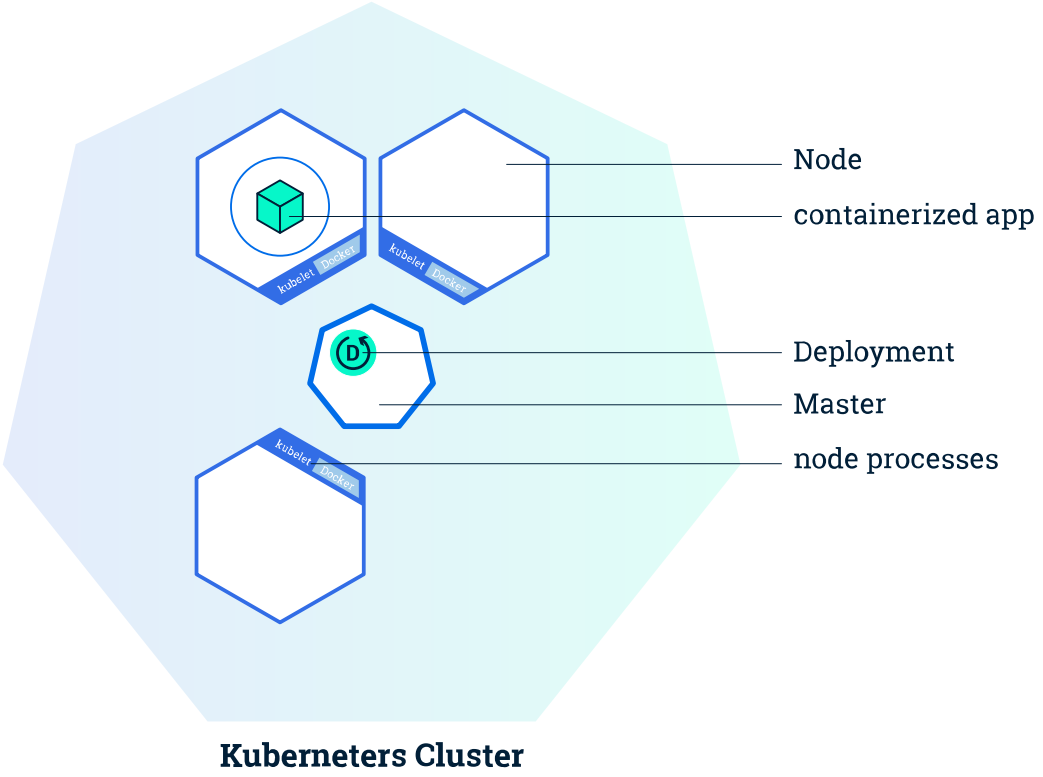
Un Kubernetes Deployment
Maneja ReplicaSets. Cada uno
Representa una versión diferente
De la aplicación desplegada.

Cada ReplicaSet maneja versiones
Identicas de Pods

Fuente: https://thenewstack.io/kubernetes-deployments-work/

Kubectl para administrar deployments

- Los deployments se pueden crear y administrar usando la interfaz de línea de comandos de Kubernetes llamada kubectl
- Kubectl interactúa con el cluster a través de API de Kubernetes



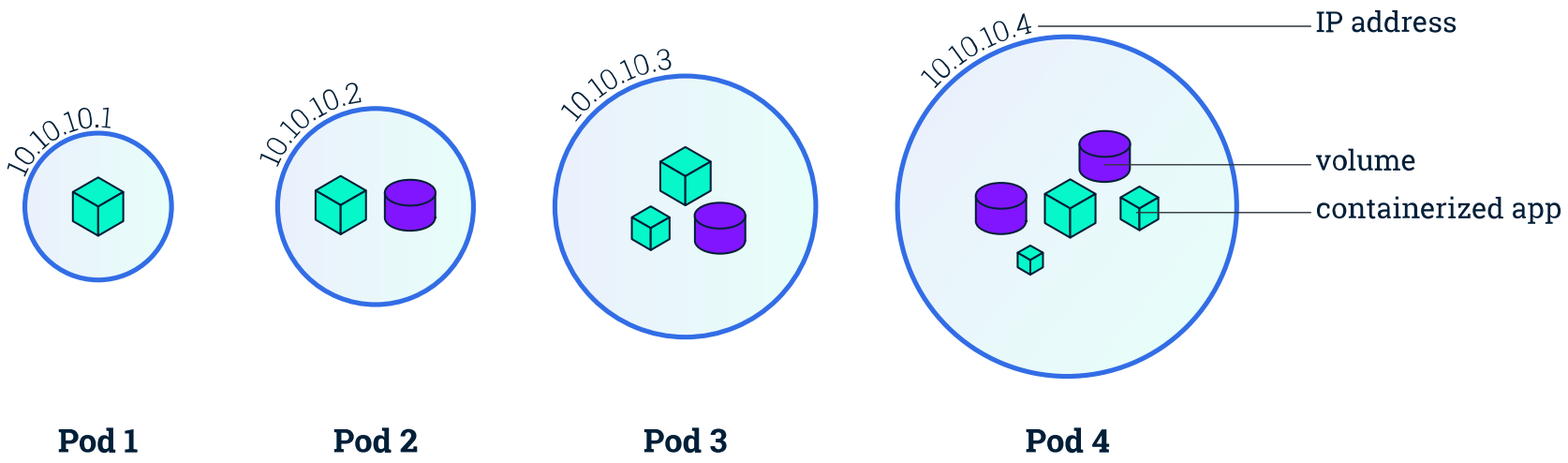
Pods

- Al crear un deployment, Kubernetes crea un Pod que contiene la instancia de la aplicación creada
- Un Pod es una abstracción que representa un grupo de uno o mas contenedores de aplicación (ejemplo Docker o rkt) y algunos recursos compartidos para esos contenedores
- Los recursos compartidos incluyen:
  - **Almacenamiento compartido**. Ejemplo: Volúmenes
  - **Networking**. Ejemplo: dirección IP única para el cluster
  - **Información de ejecución**. Ejemplo: versión de la imagen, puertos específicos a usar

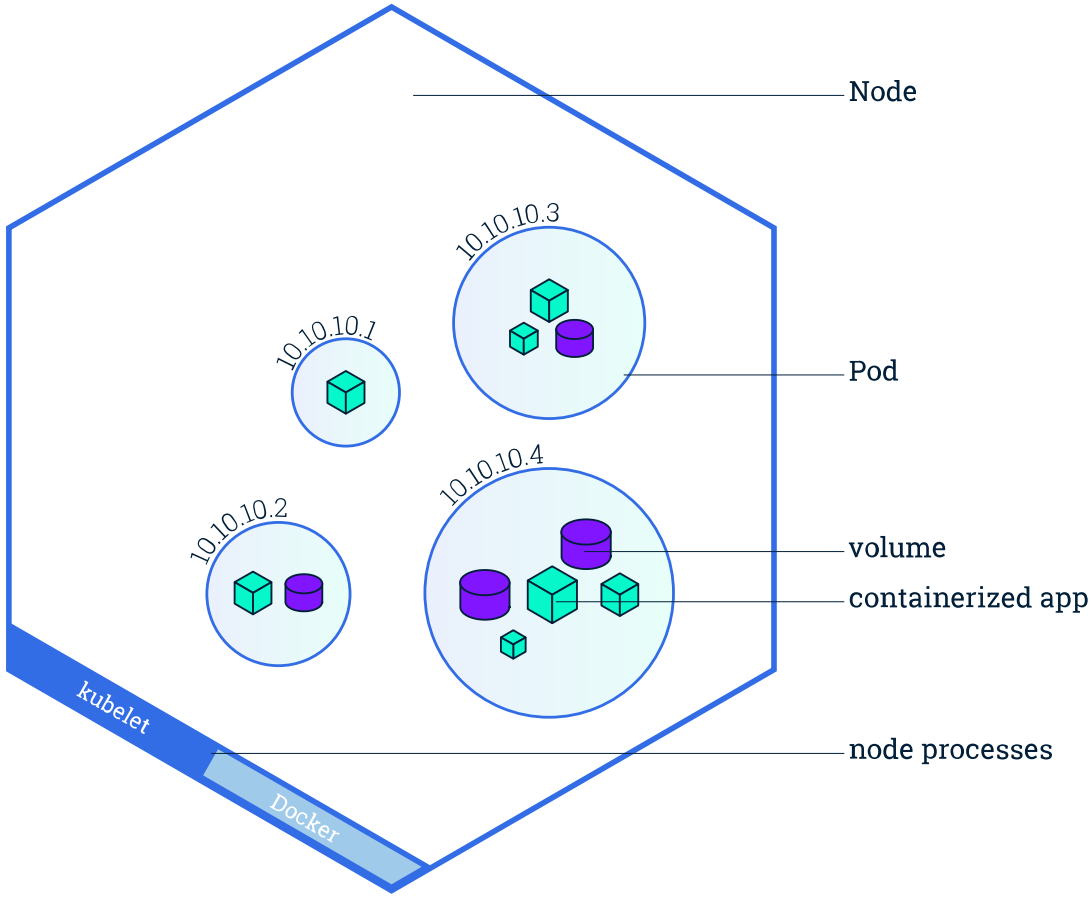
Los pods agrupan contenedores fuertemente acoplados

- Pod = “host lógico”
- Agrupan contendores fuertemente acoplados
- Los contenedores en un pod comparten una dirección IP y un espacio de puertos
- Están siempre co-locados y corren simultáneamente
- Cada pod esta ligado al nodo donde es mapeado y se mantiene ahí hasta el final (reinicio o eliminación)
- En caso de falla, pods idénticos son lanzados en otros nodos disponibes en el cluster

Ejemplo Pods



Ejemplo Nodo



Nodos

- Un pod siempre corre en un nodo
- Cada nodo corre al menos:
  - Kubelet. Un proceso responsable de la comunicación entre el master y los nodos
  - Un runtime de contenedores (ejemplo: Docker, rkt). responsable de traer la imagen desde un registry (eg. dockerhub), extraer el container y correr la aplicacion.

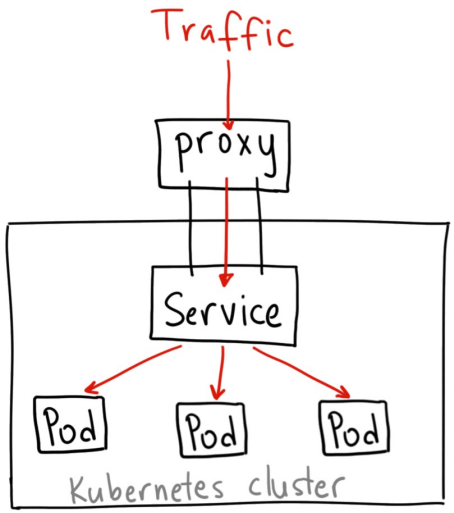
Exponer Apps Publicamente

- Los Pods son accesibles por fuera del cluster a través de servicios. Estos permiten a la aplicación recibir trafico

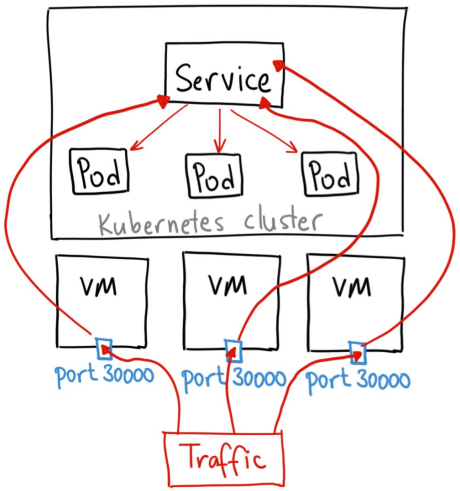
- Los servicios pueden ser publicados en formas diferentes a través del parámetro type en la especificación del servicio:

  - ClusterIP: expone el servicio en una IP interna en el cluster
  - NodePort: expone el servicio en el mismo puerto en cada nodo y lo hace accesible desde fuera del cluster
  - LoadBalancer: crea un balanceadodr de carga externo (si lo soporta) y asigna una IP externa al servicio
  - ExternalName: expone el servicio usando un nombre arbitrario (require kube-dns)

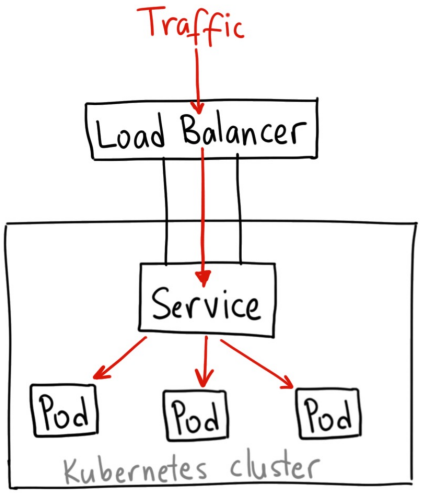
ClusterIP VS. NodePort vs. LoadBalancer



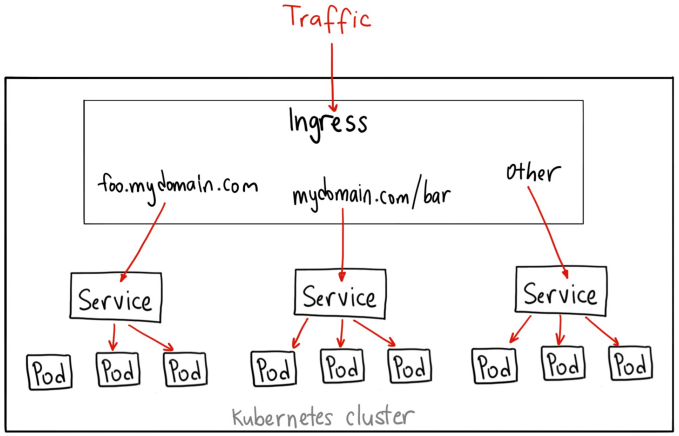
ClusterIP



NodePort

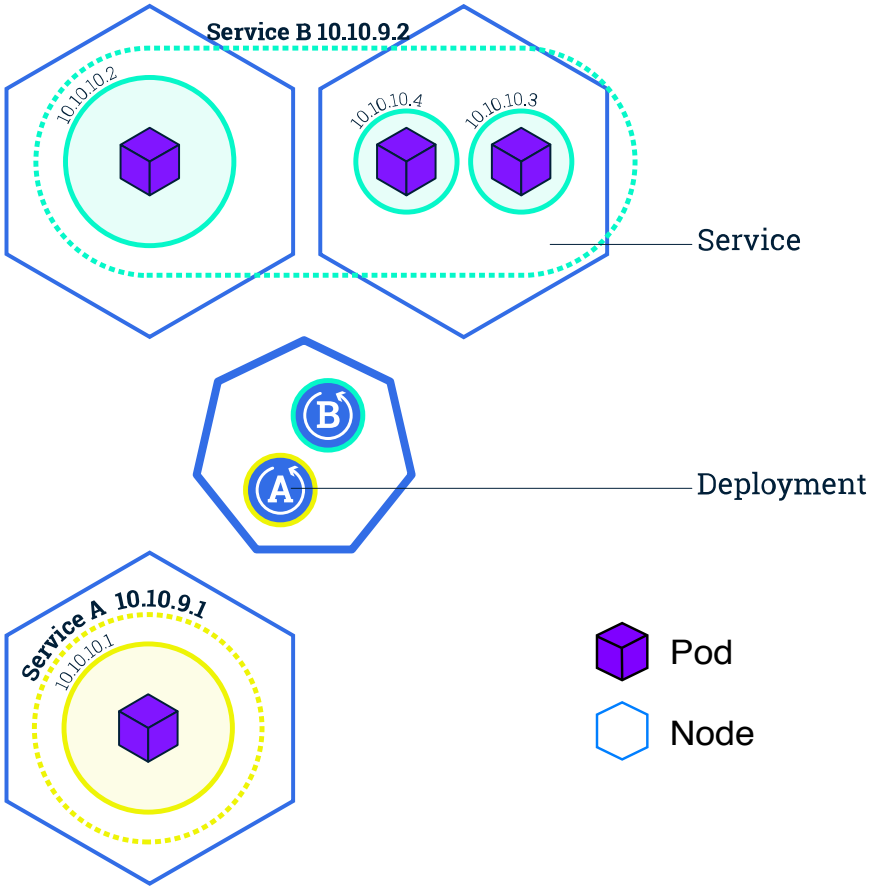


LoadBalancer



Ingress

Servicios

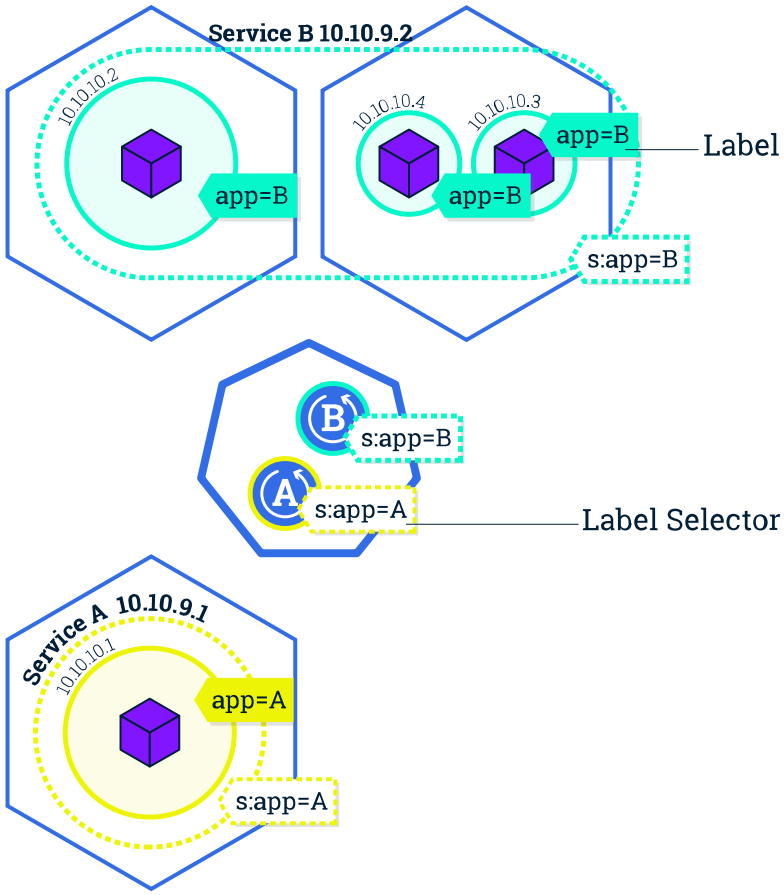

Service B 10.10.9.2
10.10.10.2
10.10.10.4
10.10.10.3
Service
B
A
Deployment
Service A 10.10.9.1
10.10.10.1
Pod
Node

Los servicios hacen transparente el ciclo de vida de los pods

- Los servicios enrutan tráfico a través de un conjunto de pods
- Los pods tienen un ciclo de vida
  - Cuando un nodo muere, los pods que corren en dicho nodo también se pierden
  - El Replication Controller puede dinámicamente recuperar el cluster a un estado deseado mediante la creación de un nuevo pod para mantener la app corriendo
- Los servicios permiten a los pods morir y replicarse en Kubernetes sin impactar la aplicacion

Labels

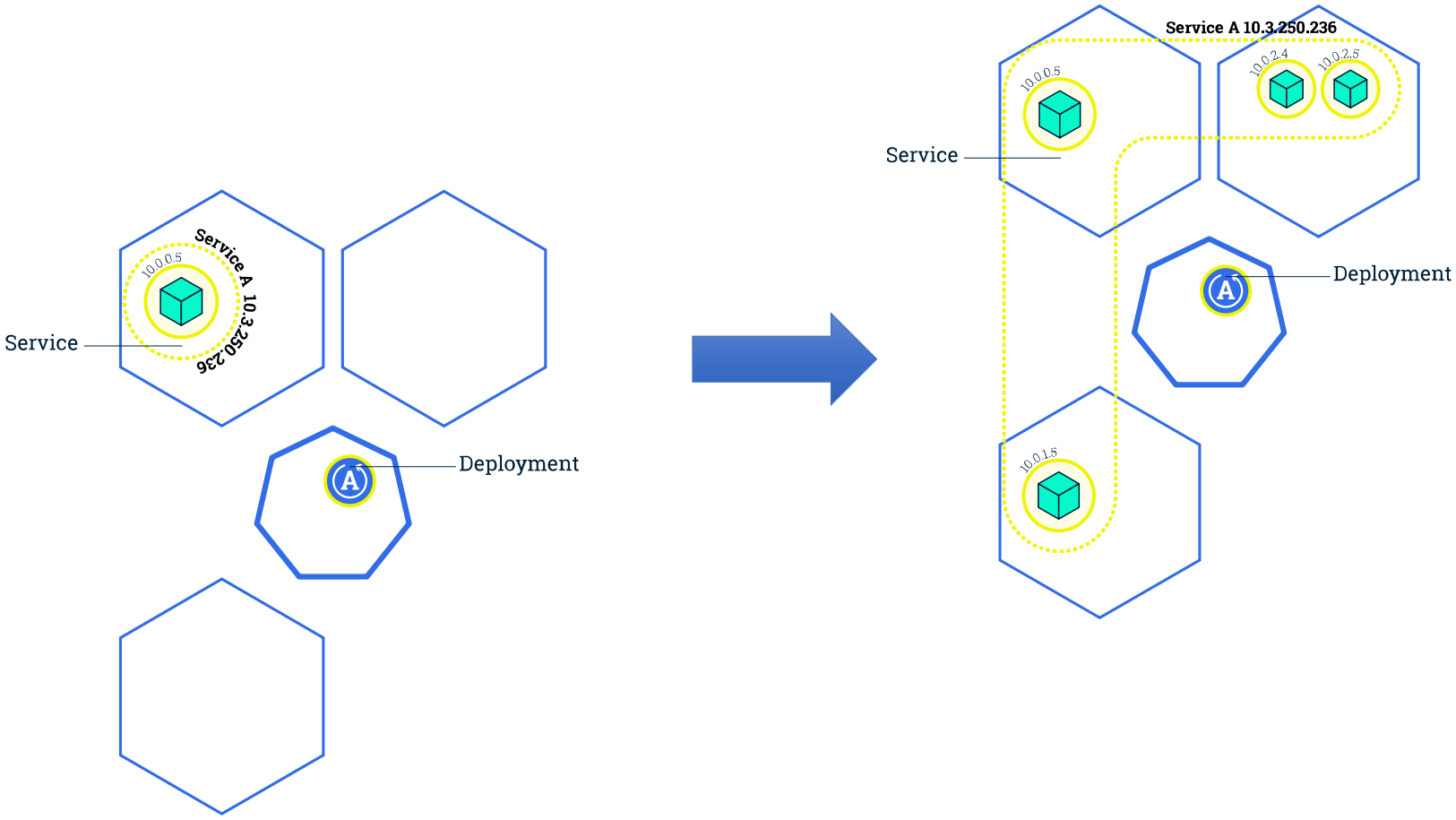
- Pares key/value que permiten ser usados con pods para fines tales como clasificar un objeto usando tags, designar objetos para desarrollo, etc.
- Asignados a objetos durante tiempo de creación o después, pueden ser modificados



Escalar Aplicaciones

- Cuando el trafico aumenta, es posible que se necesite escalar la aplicación para cumplir con la demanda de usuarios
- Esto se logra cambiando el numero de réplicas en un Deployment
- Escalar un deployment garantiza que nuevos pods son creados y asignados a nodos con recursos disponibles
- Kubernetes tambien soporta auto-scaling

Ejemplo Escalamiento



Balanceador de carga integrado

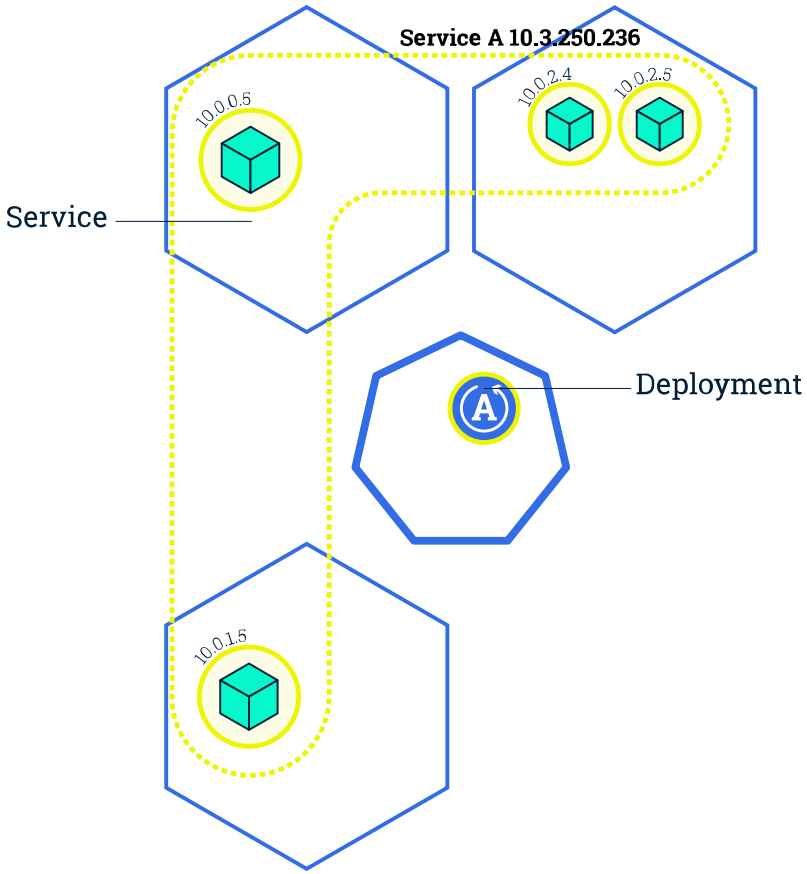
- Los servicios tienen un load-balancer integrado para distribuir el trafico de red hacia todos los pods disponibles de un deployment

- Los servicios monitorean continuamente los pods que están corriendo para garantizar que el tráfico se envié a los pods disponibles (health monitoring).

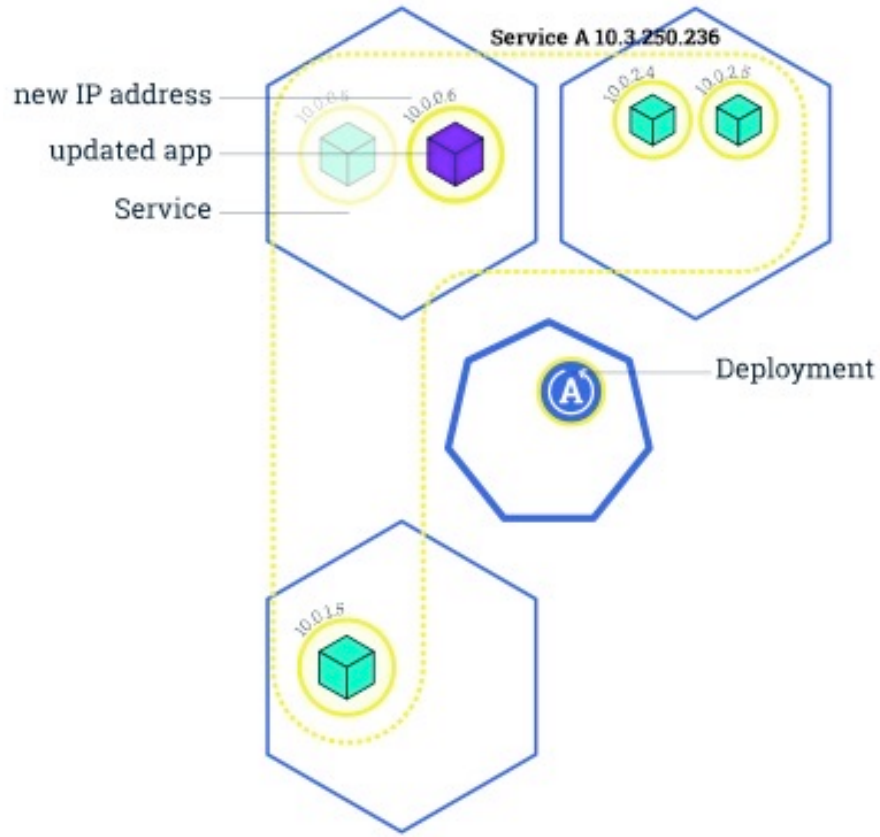
Rolling Updates

- Permiten a los desarrolladores actualizar las aplicaciones sin interrumpir el servicio
- Esto se realiza actualizando gradualmente pods existentes con las nuevas funcionalidades
- Los nuevos pods son mapeados a nodos con recursos disponibles

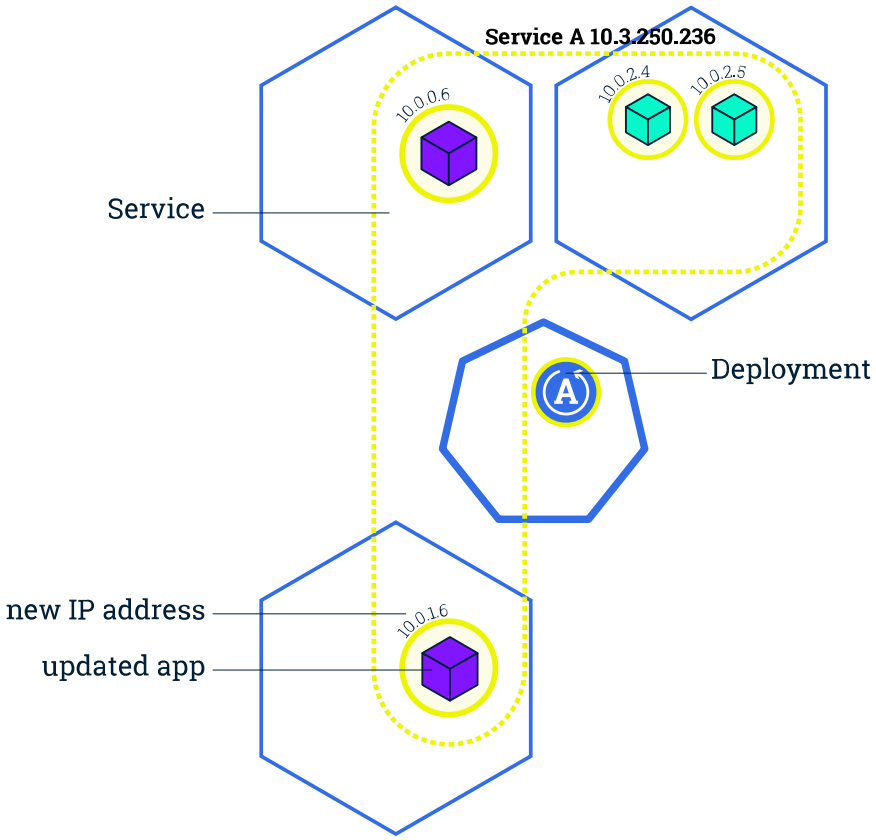
Ejemplo Rolling Updates



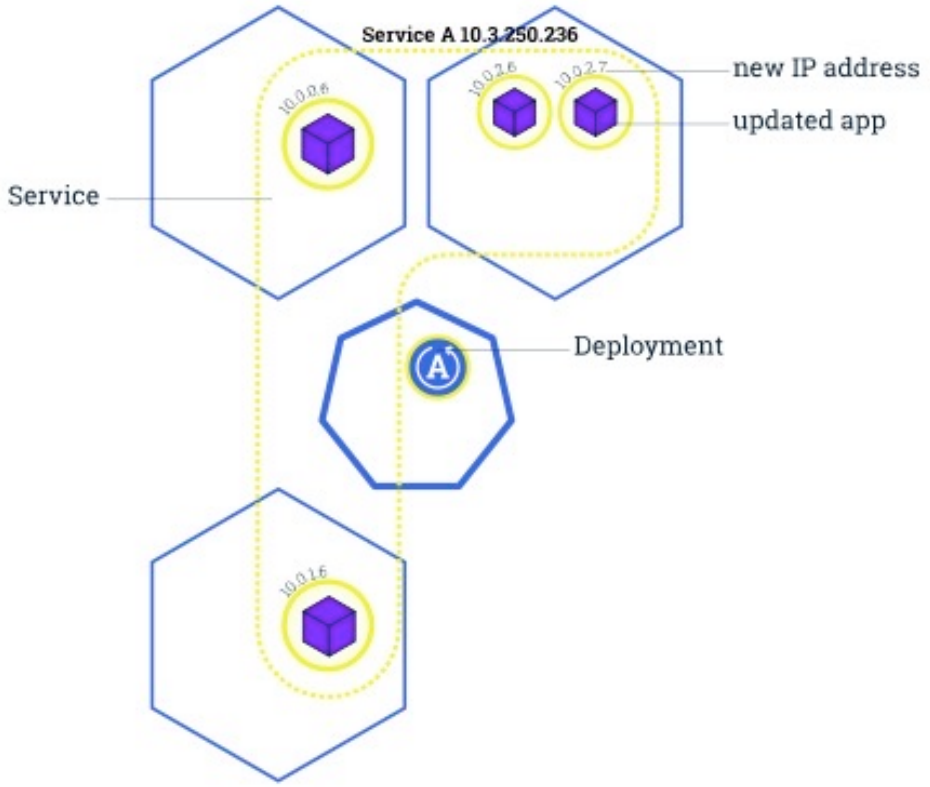
Ejemplo Rolling Updates



Ejemplo Rolling Updates



Ejemplo Rolling Updates



Referencias

- From Containers to Containers Orchestation, 2016. https://thenewstack.io/containers-container-orchestration/
- Kubernetes: Production-grade container orchestration: https://kubernetes.io/
- Docker Swarm vs. Kubernetes. https://www.upcloud.com/blog/docker-swarm-vs-kubernetes/