KUBERNETES

trentialearning.net

“Tecnología y Talento para lograr cosas extraordinarias”

trentia learning

ÍNDICE

01 Sobre nosotros

02 Sobre el formador

03 Alta disponibilidad

04 Comprendiendo Kubernetes

05 Manifiestos

06 Workloads (pods, deployments, etc)

07 Networking (services, ingress, etc)

08 Helm

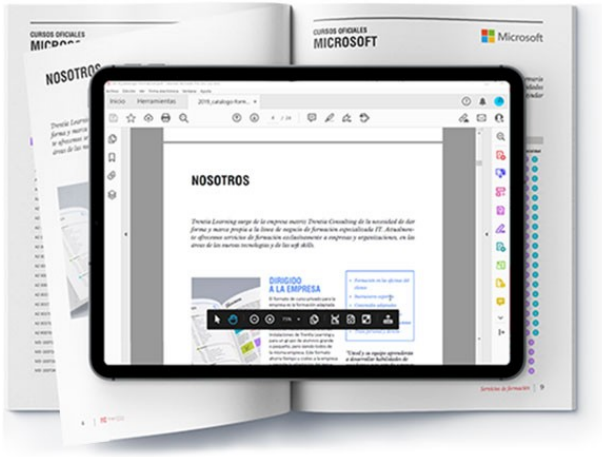
09 Operators

¿Quiénes somos?

Trentia comenzó siendo un proyecto ilusionante de unos treintañeros que queríamos hacer Tecnología porque creemos en ella.

Actualmente tenemos 3 líneas de negocio:

1. **Trentia Projects & Services:** Desarrollo de Proyectos IT/Servicios IT y Consultoría especializada.
2. **Trentia Tech Talent**: Departamento de búsqueda, atracción y contratación del mejor Talento IT especializado.
3. **Trentia Learning:** Formación IT especializada para impulsar tu Estrategia Digital de Negocio.



Trentia Learning

Creemos en el poder transformador que ejerce la Formación dentro de las organizaciones.

Contamos con un excelente equipo de formadores expertos en distintas tecnologías, una metodología docente experiencial y colaborativa, y diferentes tipos de cursos, desde oficiales a talleres prácticos de especialización. Todo para ayudaros a llevar a cabo vuestra **Estrategia Digital**.

David Pestana

Sobre mi

Ingeniero de telecomunicaciones de estudios, emprendedor en serie de espíritu, “teckie” de vocación, mentor y formador de afición, aportar valor es mi profesión.

david.pestana@trentia.net

https://www.linkedin.com/in/david-pestana-perdomo-3669b58b/

Principios de Alta disponibilidad (HA)

Alta Disponibilidad

En esta unidad vamos a trabajar con diferentes técnicas para lograr **Alta Disponibilidad** (*HA*) en nuestros entornos de producción. Teniendo en cuenta que HA no solo es poder ofrecer un servicio continuado y sin cortes el 100% del tiempo si no que también es garantizar una **Calidad de Servicio** (*QoS*) dentro de unos parámetros durante todo este tiempo, sea cual sea el escenario.

indice de disponibilidad

se mide dividiendo el tiempo durante el cual el servicio está disponible por el tiempo total en un periodo determinado, por ejemplo un año.

Indice	Tiempo de inactividad
97 %	11 días
99 %	3 días y 15 horas
99'9999 %	32 segundos

> ¿ es el indice de disponibilidad 100% un valor utópico ?

Principios de Alta disponibilidad (HA)

como lograr alta disponibilidad

redundancia

al menos dos sistemas para resolver una sola función.

balanceo

un reparto equitativo de la carga de trabajo

tolerancia

los sistemas deben ser tolerantes a fallos

resurrección

los sistemas deben tener mecanismos para volver a la vida sin intervención humana si llegan a morir

Alta disponibilidad durante el despliegue

Ya estudiamos diferentes estrategias de despliegue que aseguraban un tiempo de *downtime* cero, y cualquiera de estas nos va a servir en nuestro protocolo de Alta Disponibilidad, pero ademas de esto tenemos que garantizar la posibilidad de *RollBack* en caso de fallo durante el despliegue. Disponer de un entorno de producción en un ambiente contenerizado nos va a facilitar mucho esta tarea aunque no es imprescindible. Capistrano, Capifony, Jenkins... son algunas herramientas que nos permiten lograr Alta disponibilidad durante el despliegue sin trabajar con contenedores.

Principios de Alta disponibilidad (HA)

Calidad de servicio (QoS)

Según el servicio que estemos ofreciendo los parámetros de Calidad de Servicio podrán ser diversos, pero en habitual caso de ofrecer servicios web basados en las diferentes variantes del protocolo HTTP(s), el parámetro a medir para determinar la calidad del servicio es el tiempo de respuesta. Dejando de lado la tasa de bit que depende de la velocidad de conexión, generalmente la velocidad de conexión del cliente sera muy inferior a la que pueda ofrecer nuestra infraestructura tanto en upstream como en downstream. Consideraremos el tiempo de respuesta como el tiempo que tarda el sistema en entregar la respuesta una vez recibida la petición. Así como en la cantidad de respuestas por segundo que estamos siendo capaces de entregar. En otros servicios tcp como por ejemplo WebSocket, la calidad de servicio se apalanca mas en la estabilidad de la conexión, ping y tasa de bit, asi como en la concurrencia de clientes suscritos.

En servicios HTTP (request/response) ... consideraremos calidad de servicio mínima respuestas del orden de 200 a 500 milisegundos y una calidad de servicio optima tiempos del orden de 20 a 50 milisegundos, tiempos por encima de 500 milisegundos son considerados inaceptables, y por tanto estar fuera de estos parámetros es igual a no estar ofreciendo alta disponibilidad.

Alta disponibilidad frente a alta concurrencia

Los escenarios de producción pueden ser muy diferentes, pero existen escenarios en los que tenemos que garantizar alta disponibilidad cuando la concurrencia de conexiones es extremadamente elevada. Vease el caso de las redes sociales, periódicos digitales, servicios de streaming, etc.

escalado vertical

> consiste en dotar de mas recursos a los sistemas que conforman la infraestructura, memoria, cpu... etc, el escalado vertical es limitado.

escalado horizontal

> consiste en dotar de mas sistemas a la infraestructura, lo cual aumenta la redundancia y la capacidad de computación de manera infinita.

Principios de Alta disponibilidad (HA)

¿Cuanto cuesta la Alta disponibilidad?

El uso de sistemas escalados tanto vertical como horizontalmente aumenta los costes para poder garantizar la alta disponibilidad en casi cualquier escenario, sin embargo este posible escenario no se va a dar el 100% del tiempo.

autoescalado

> consiste en disponer de mecanismos que aumenten o disminuyan de manera automática la dotación de recursos a un determinado servicio en función de la demanda, ahorrando muchísimos costes, pero a su vez aumenta la complejidad de implantación, gestión, despliegue... etc.

Alerta Tsunami !!!

En ocasiones también en algunos contextos o escenarios, podemos encontrarnos con crecimientos abismales en la concurrencia, tanto de manera prevista como imprevista, aumentos del 1000 o 10mil por ciento de la concurrencia en muy pocos segundos, el efecto de esto es similar al de un ataque por denegación de servicio (DDoS) y estar preparados para ello requiere mucho análisis.

Introducción

Kubernetes (K8s) es una plataforma de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones en contenedores.

También se le denomina k8s, por las 8 letras que hay entre la "k" y la "S", y el proyecto se inició en Julio de 2015 y fue donado posteriormente a Cloud Native Computing Foundation CNCF

Introducción

Es un proyecto **OpenSource** creado por Google, el cual lo utiliza en la mayoría de sus proyectos como son Gmail, Maps, Drive, etc.

Además expone de una forma programática para que podamos acceder a la gestión de nuestro cluster. Como orquestados permite crear un cluster de nodos que implementa determinadas funciones sobre los contenedores:

- Permiten alta disponibilidad.
- Es tolerante a fallos.
- Permite escalar cuando se queda corto de recursos.
- Trabaja de forma eficiente.
- Nos permite realizar cambios y operaciones en caliente.

Introducción

Existen otros orquestadores:

- Swarm mode overview
- Marathon: A container orchestration platform for Mesos and DC/OS
- Nomad by HashiCorp
- shipyard project

Comprendiendo el fudamento Kubernetes

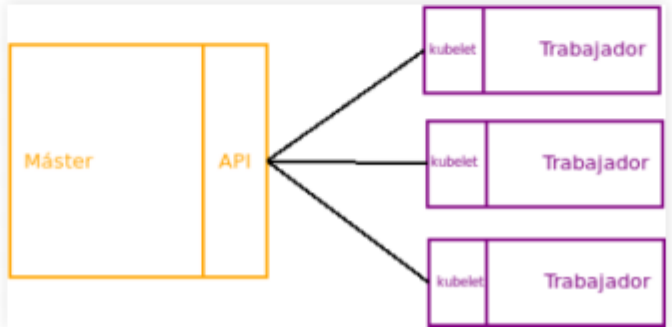
¿Que es un contenedor desde un punto de vista computaciónal?

Conceptos: Cluster

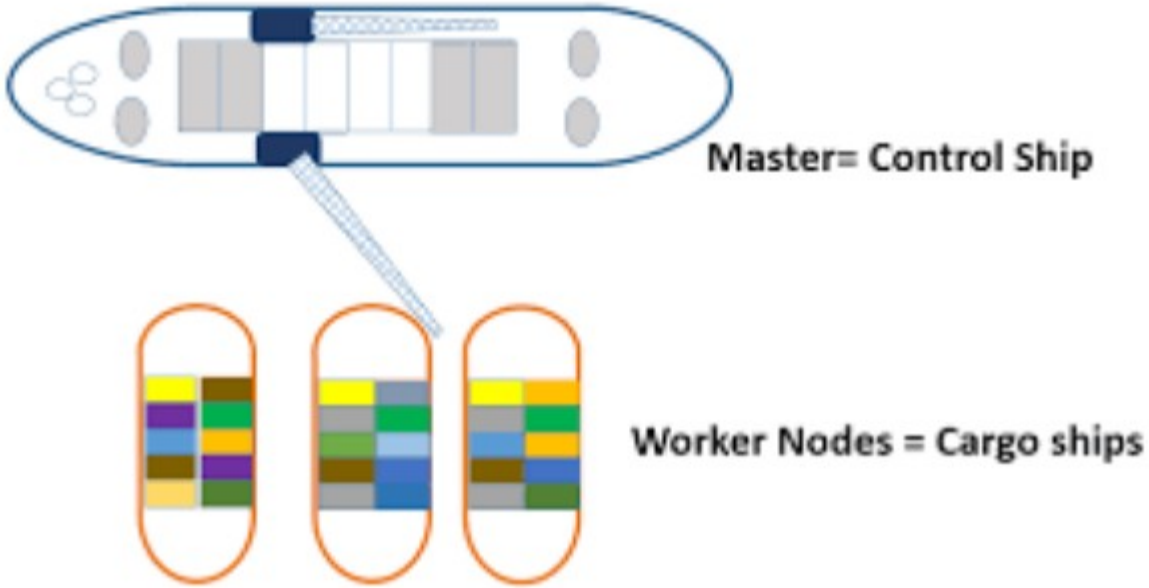
Un clúster es un conjunto de máquinas, donde cada máquina tiene un rol que se dividen en dos:

Máster: sólo hay uno en el cluster y es el encargado de repartir la carga de trabajo, entre los demás nodos, mantener el estado deseable de trabajo, escalar las aplicaciones o actualizarlas.

Workers: los demás nodos que nos Máster. Son los que ejecutan las aplicaciones por medio de contenedores:



Conceptos: Cluster



Conceptos: Manifiestos

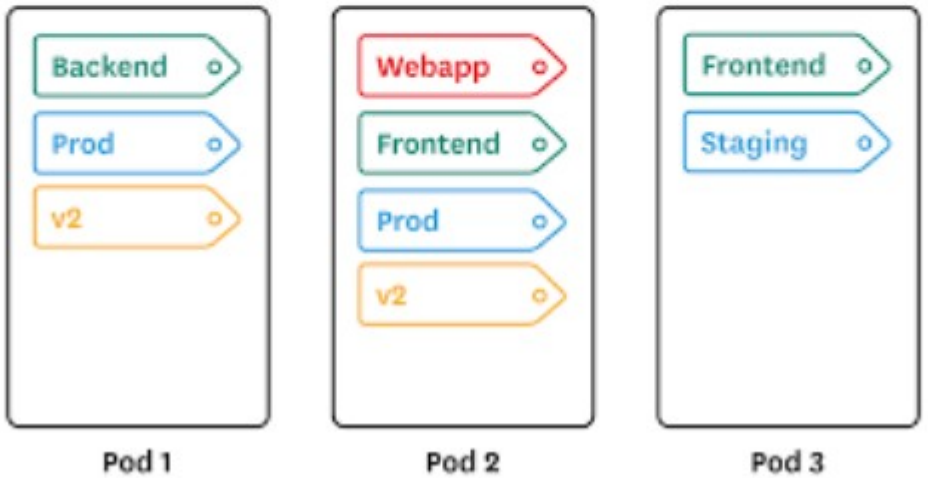
kubernetes

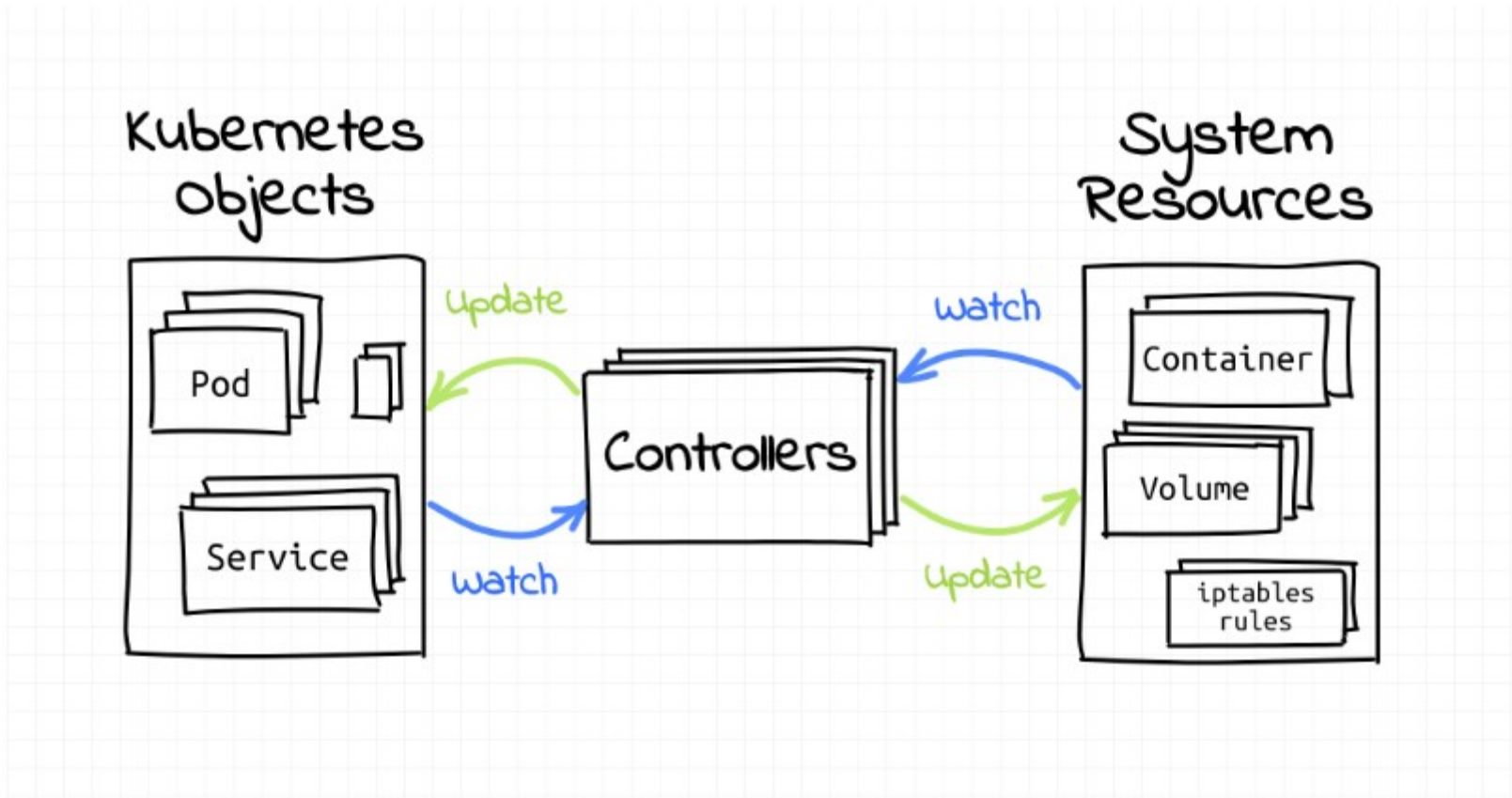
Control de Versiones

Creamos un repo !!

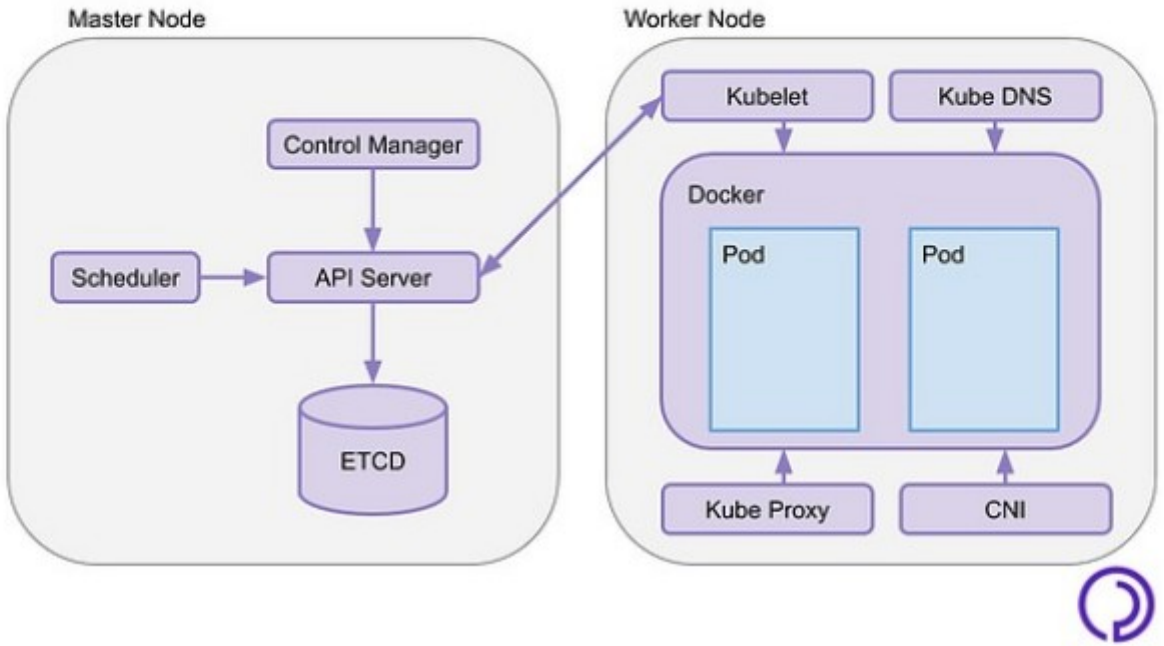
Conceptos: Labels y selectors

Son pares de claves y valores, las cuales se puede aplicar a pods, services, etc. y con
ellos podemos identificarlos para poder gestionarlos.



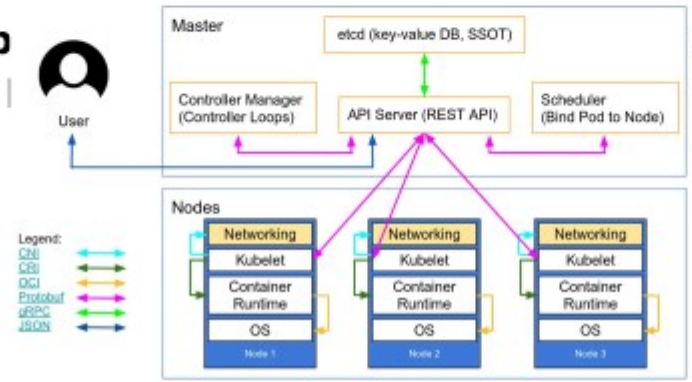

Kubernetes objects
Pod
Service
Update
watch
Controllers
watch
Update
System Resources
Container
Volume
iptables rules

Arquitectura de kubernetes.

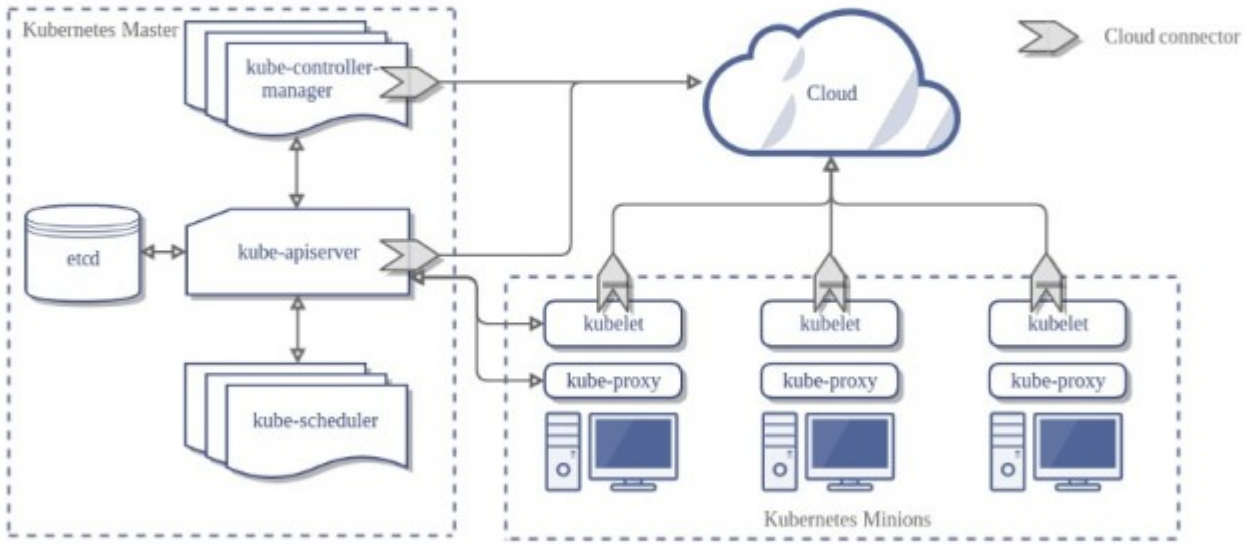


Arquitectura de kubernetes.

En el nodo Worker dispone de un Container Runtime, teniendo en cuenta que no sólo
ejecuta docker, sino cualquier otro contenedor, al nal dispone de componentes como
los pod, que son el elemento básico que utilizamos en el cluster. Y después dispone de
un agente que se llama **kubelet** que se comunica con el maestro. Por último se
encuentra en **Kub** ar por los pods, y se utiliza para
gestionar y como de los pods.



Arquitectura de kubernetes.



Arquitectura de kubernetes.

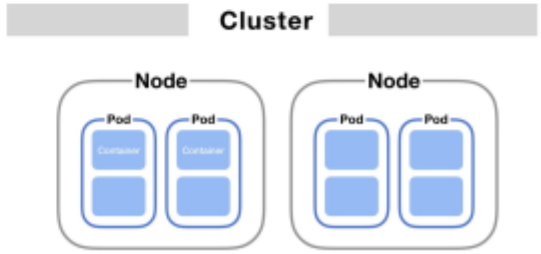
- **Etcd**: Almacena los datos de configuración a los que puede acceder el Servidor API de Kubernetes Master utilizando http simple o API JSON.
- **Kube-Apiserver**: es el centro de gestión para el nodo Master, facilita la comunicación entre los diversos componentes manteniendo con ello la salud del clúster.
- **Kube-Controller-Manager**: es el encargado de asegurar la coincidencia entre el estado deseado del clúster y el estado actual, esto lo consigue esclando cargas de trabajo hacia arriba y hacia abajo
- **Kube-Scheduler**: coloca la carga de trabajo en el nodo que corresponde; en este diagrama particular, todas las cargas de trabajo se ubican localmente en su host.
- **Kubelet**: recibe las especificaciones del pod del servidor API y administra los pods que se ejecutan en el host.

Conceptos: Node

Un nodo **es una máquina de trabajo en Kubernetes**, previamente conocida como minion.
Un nodo puede ser una máquina virtual o física, dependiendo del tipo de clúster.
Cada nodo está gestionado por el componente máster y contiene los servicios necesarios para ejecutar pods.

Los servicios en un nodo incluyen el container runtime, kubelet y el kube-proxy.
Accede a la sección The Kubernetes Node en el documento de diseño de arquitectura para más detalle.

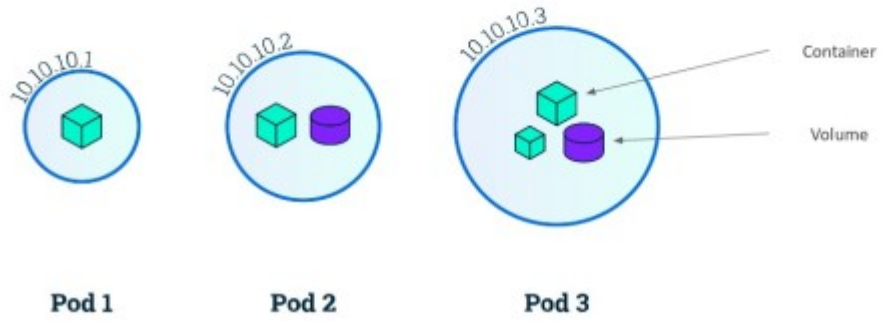
Controlador de Nodos



Conceptos: Pod

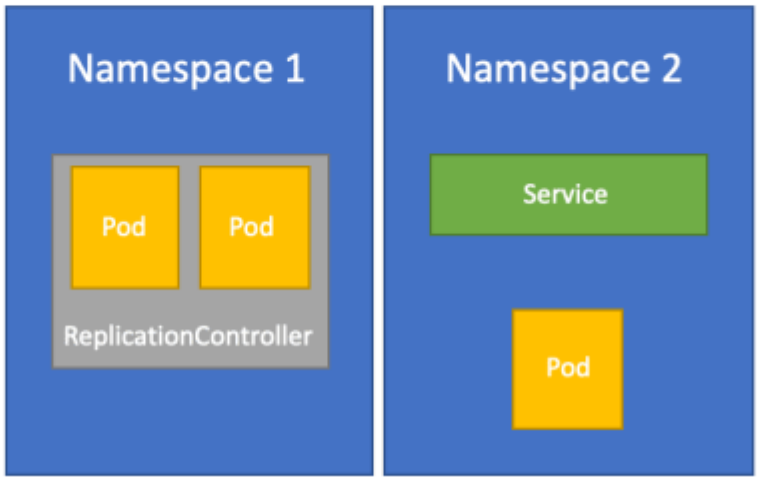
Es la unidad mínima de Kubernetes. Representa en ejecución en el clúster. Un Pod contiene uno o varios contenedores (por ejemplo Docker), y administrar los recursos de la aplicación, dirección IP y distintas opciones que determina cómo funcionan los contenedores.

https://kubernetes.io/es/docs/concepts/workloads/pods/pod/



Conceptos: Namespaces

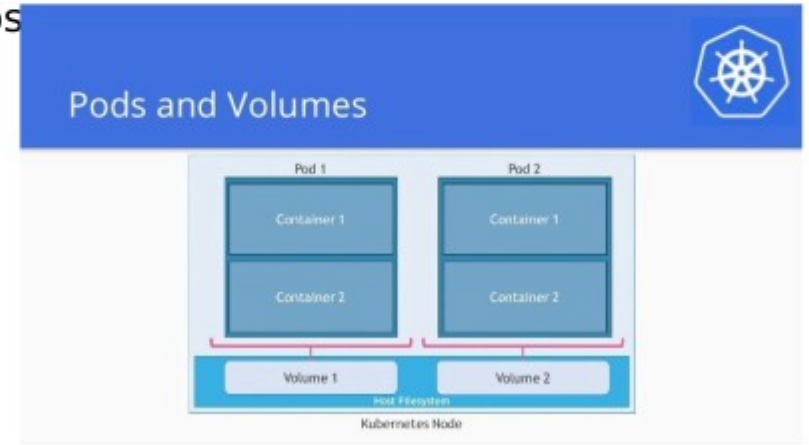
Son agrupaciones de la infraestructura que nos permite diferenciar espacios de trabajo, por ejemplo desarrollo, producción, etc



Conceptos: Volumes y Secrets

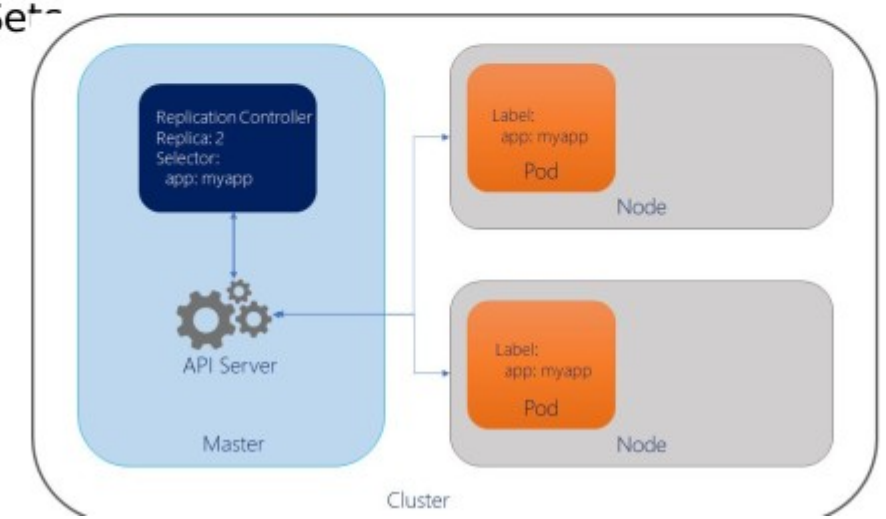
Volumes
La gestión del sistema de almacenamiento de nuestros pods.

Secrets
Es donde se guarda la información confidencial como usuario y passwords para acceder a los recursos



Conceptos: Replication Controller.

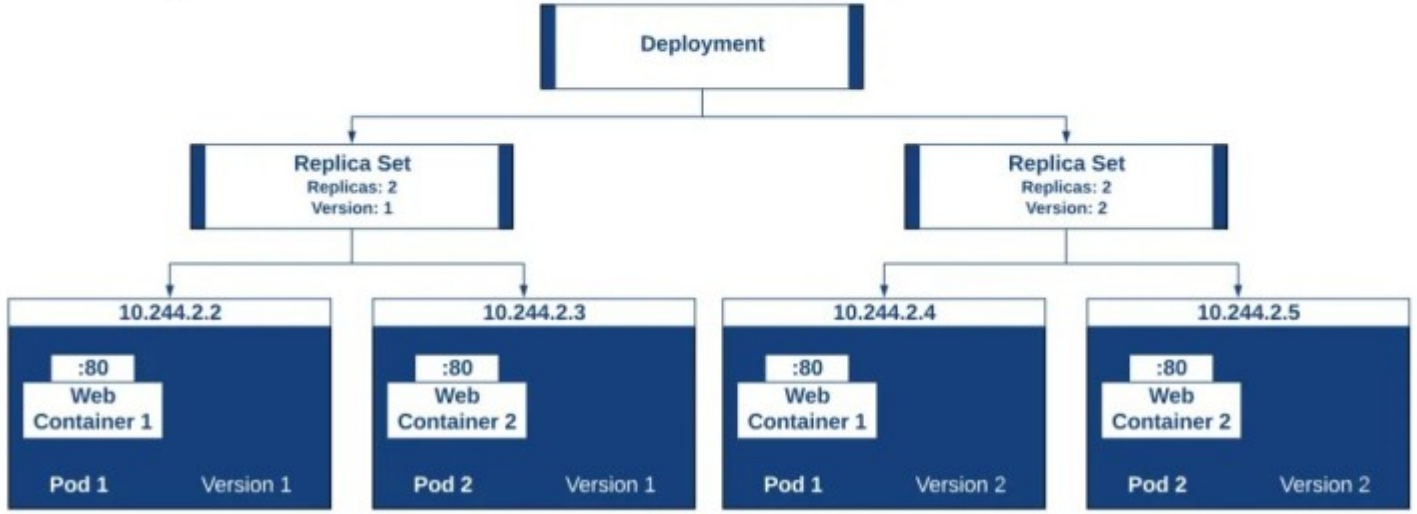
Es el responsable de gestionar la vida de los pods y el encargado de mantener arrancados los pods que se hayan indicado en la configuración.
Permite escalar de forma muy sencilla los sistemas y maneja la recreación de un pod cuando ocurre algún tipo de fallo, en las últimas versiones nos recomienda utilizar replica Se[illegible]



LABORATORIO: REPLICATION CONTROLER

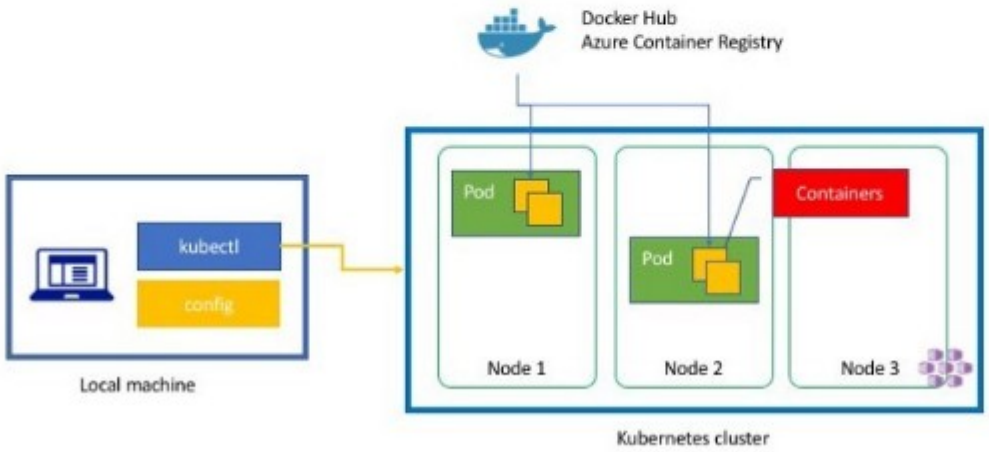
Conceptos: Replica Sets

Es la nueva generación de Replication Controller, con nuevas funcionalidades. Una de las funcionalidades destacadas que nos permite desplegar pods en función de los **labels y selectors.**



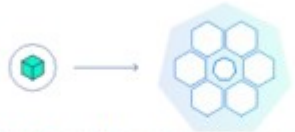
Conceptos: Deployments

Es una funcionalidad más avanzada que los Replication Controller y nos permite especificar la cantidad de réplicas de pods que tendremos en el sistema.



PRACTICANDO LO BÁSICO

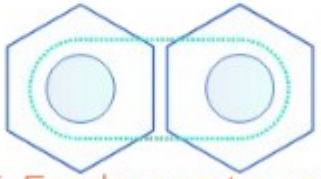
1. Crear un cluster de K8S



2. Desplegar una app

3. Explorar nuestra app

4. Publicar nuestra app

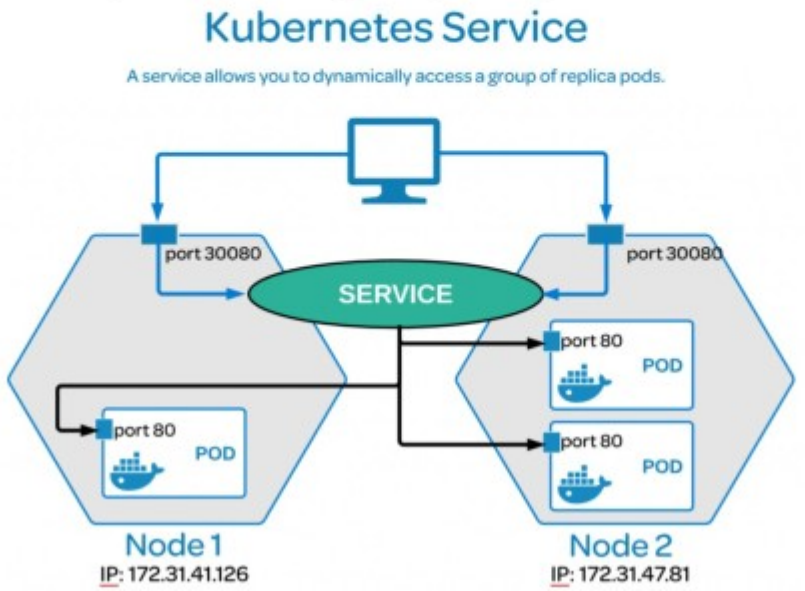


5. Escalar nuestra app

6. Actualizar nuestra app

Conceptos: Service

Es la política de acceso a los pods. Lo podríamos definir como la abstracción que dene un conjunto de pods y la lógica para poder acceder a ellos.



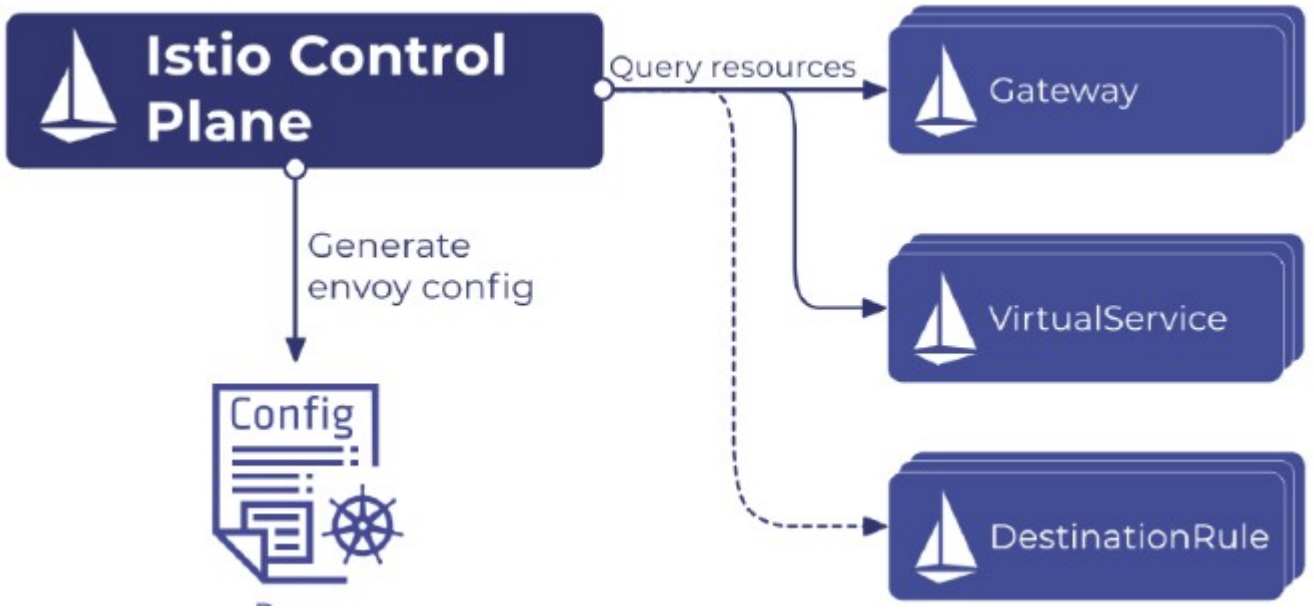
Laboratorio: Exponiendo un servicio

¿Qué es el ingress?

Ingress expone las rutas HTTP y HTTPS desde fuera del clúster a los servicios dentro del clúster. El enrutamiento del tráfico está controlado por reglas definidas en el recurso Ingress.

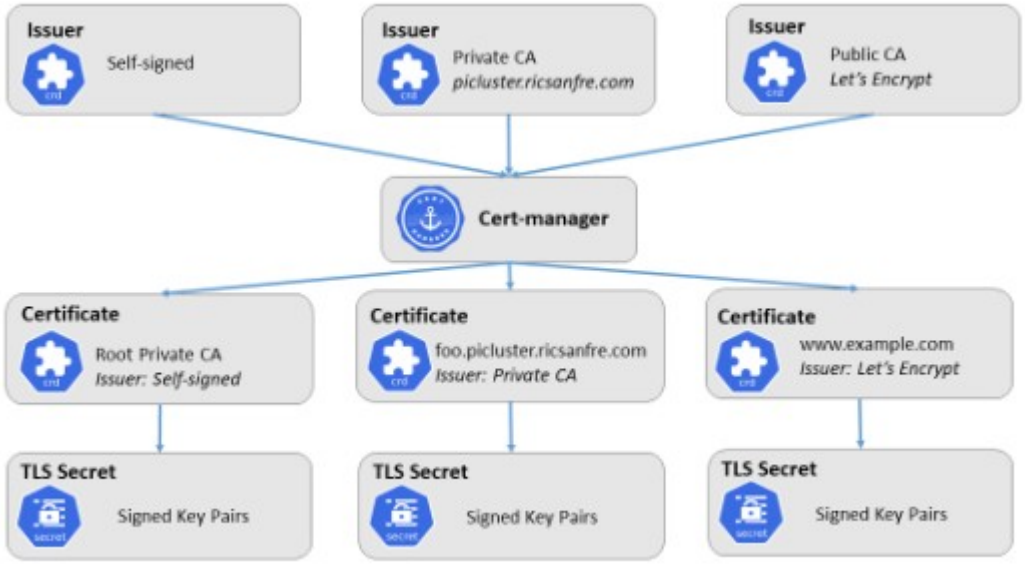
Aquí hay un ejemplo simple donde un Ingress envía todo su tráfico a un Servicio:

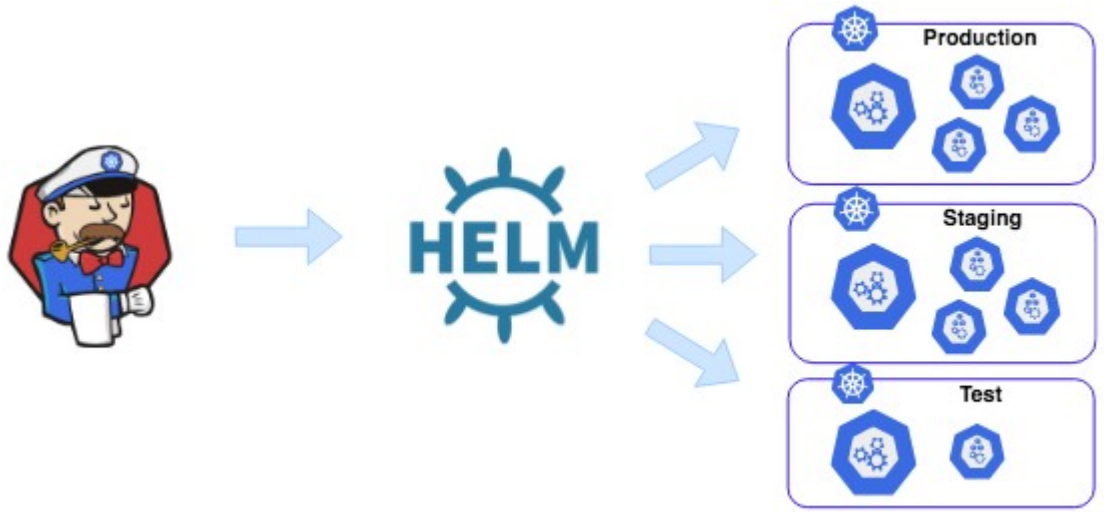
Ingress Controller - Istio



Ingress Controller - Nginx

CERT MANAGER

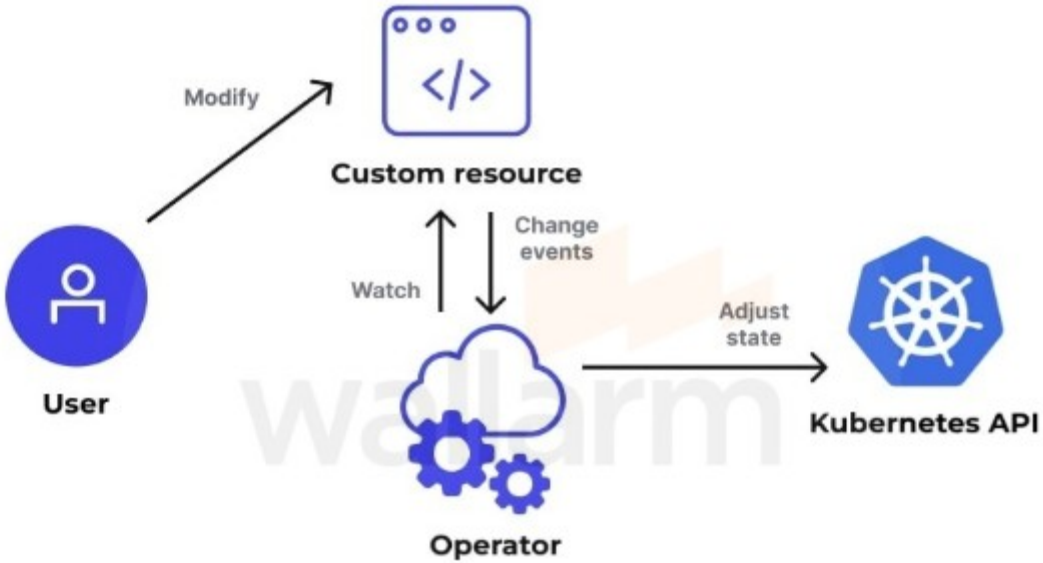



HELM
Production
Staging
Test

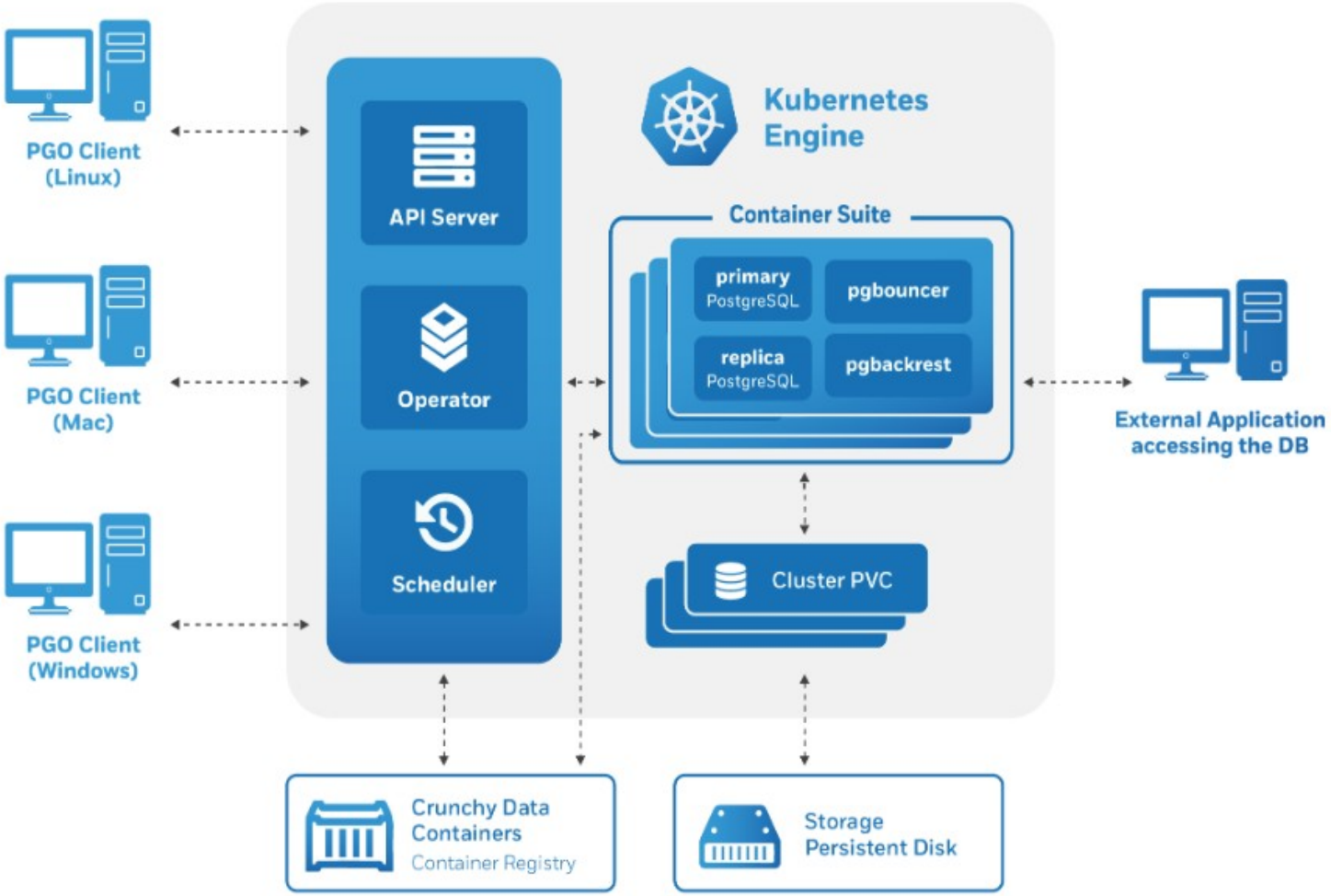
Laboratorio: Desplegando nuestro Ingress Controller

Laboratorio: Desplegando Cert Manager

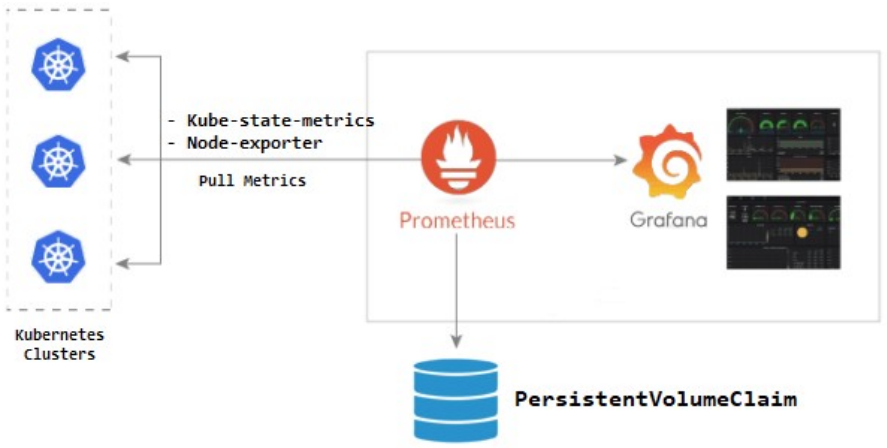
What is Kubernetes Operator?

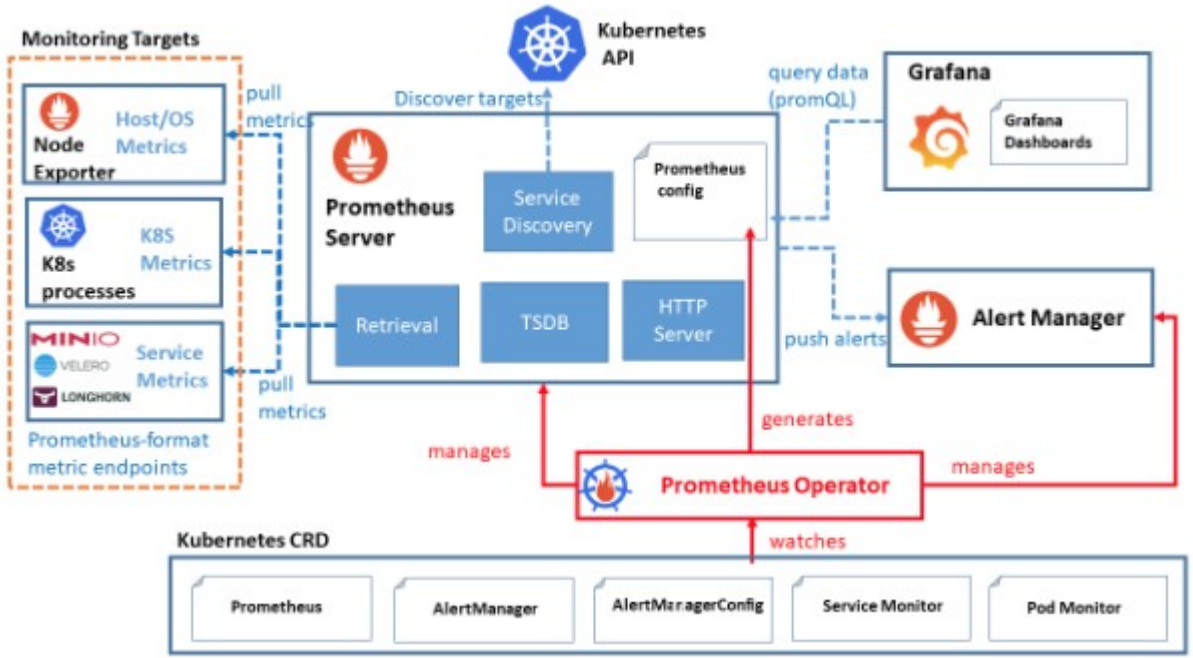


Laboratorio: Desplegar Postgres Crunchy Operator

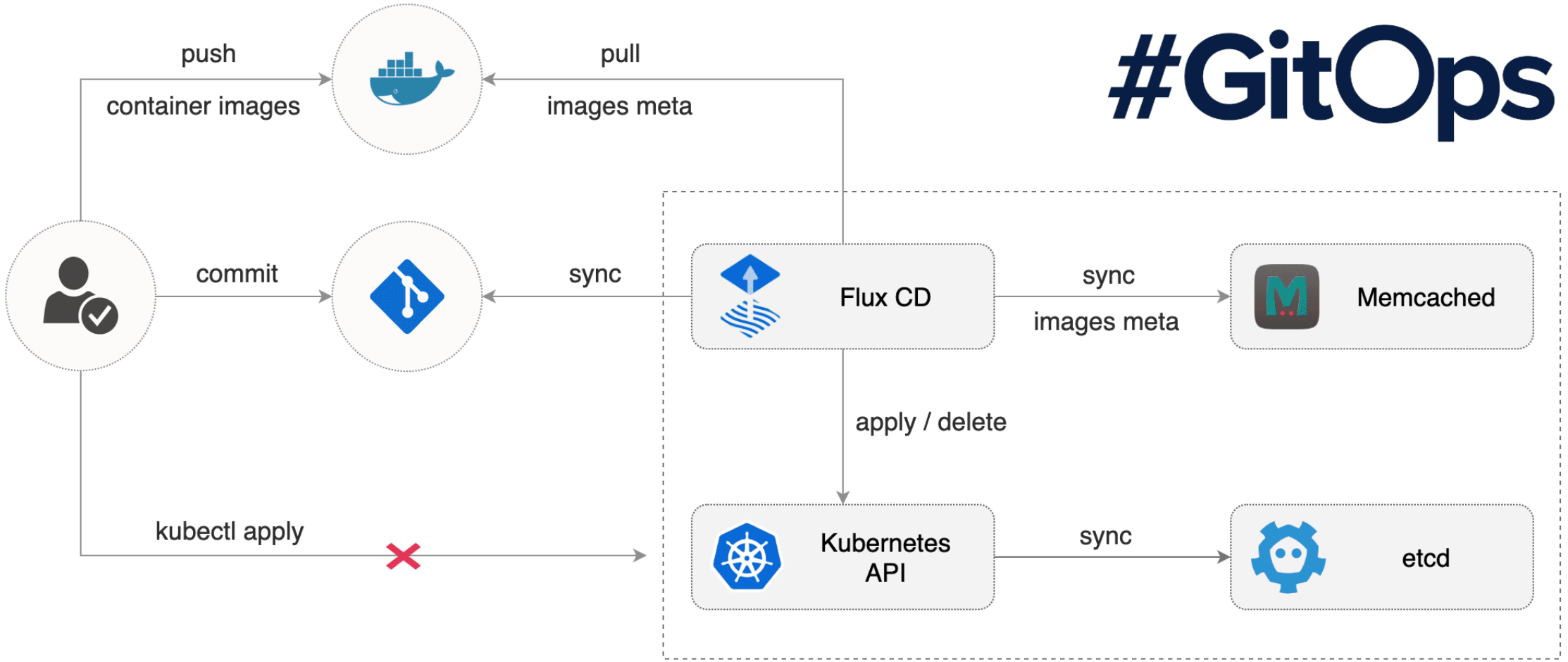


STACK DE MONITORIZACION



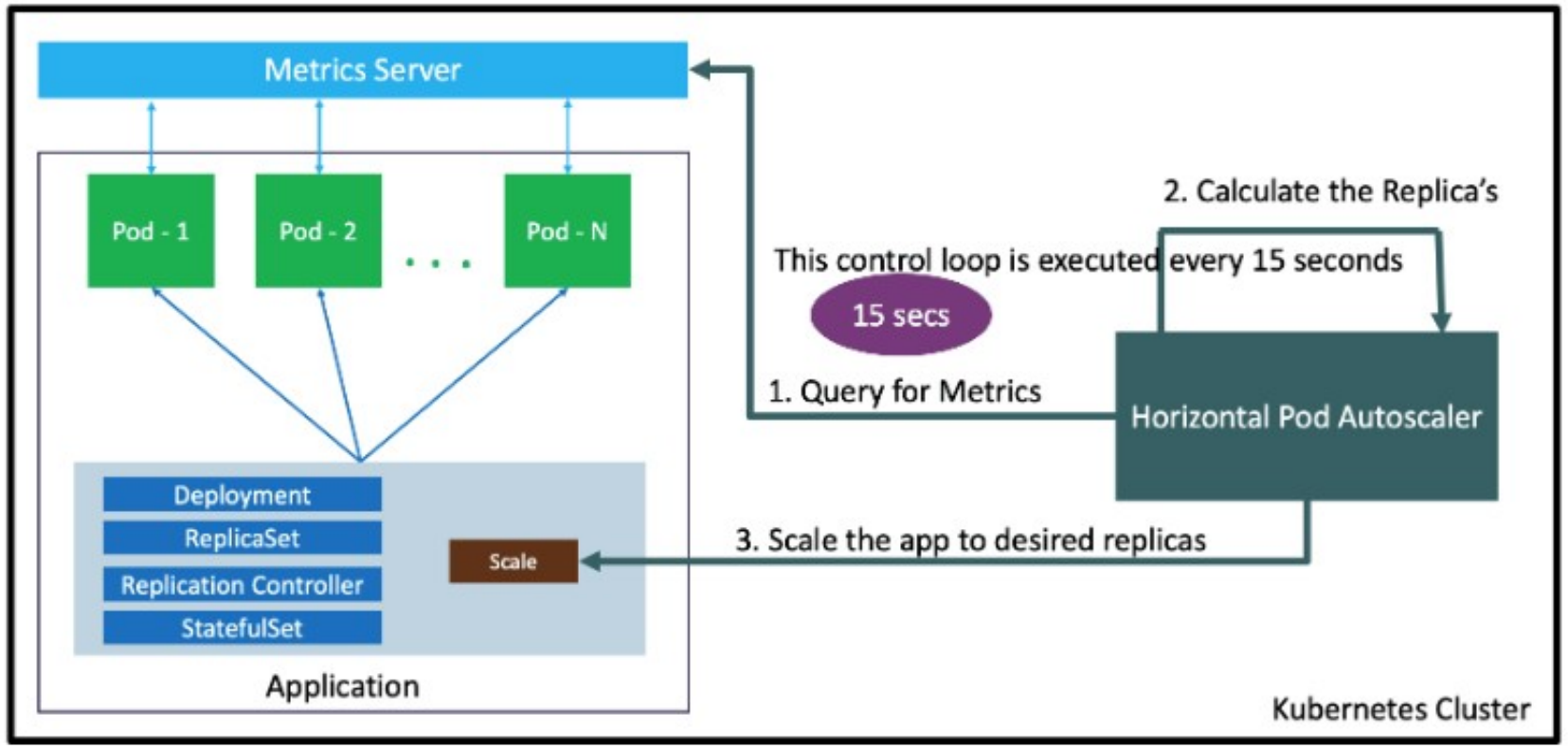


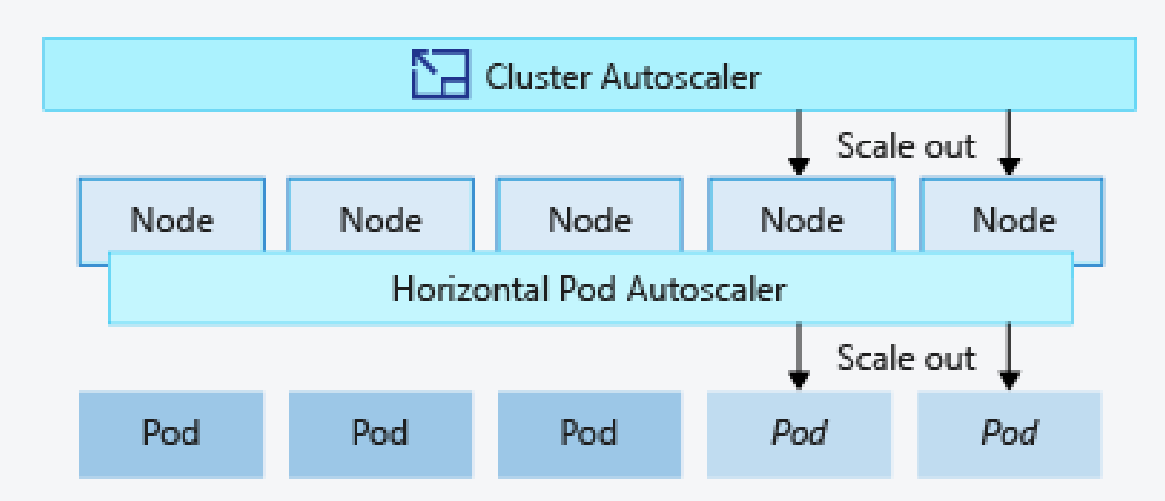
Laboratorio: Desplegar Stack de monitorización


push
container images
pull
images meta
#GitOps
commit
sync
Flux CD
sync
images meta
Memcached
apply / delete
kubectl apply
Kubernetes
API
sync
etcd

LABORATORIO: DESPLEGAR FLUX

How HPA works?




Cluster Autoscaler
Scale out
Node
Node
Node
Node
Node
Horizontal Pod Autoscaler
Scale out
Pod
Pod
Pod
Pod
Pod

¡Muchas gracias!

Oficina Barcelona

Rocafort 241-243, 4º
5ª 08029 Barcelona

Teléfono: (+34) 934 198 864
Email: hello@trentia.net

Oficina Madrid

Príncipe de Vergara 112
28002 Madrid

Teléfono: (+34) 934 198 864 Email: hello@trentia.net

Nuestras webs

Trentia: https://www.trentia.net
Trentia Learning: https://www.trentialearning.net
Trabaja con nosotros: https://empleoit.trentia.net

“NO ES LO QUE HACEMOS, ES COMO LO HACEMOS”