Apuntes - 14/02/23

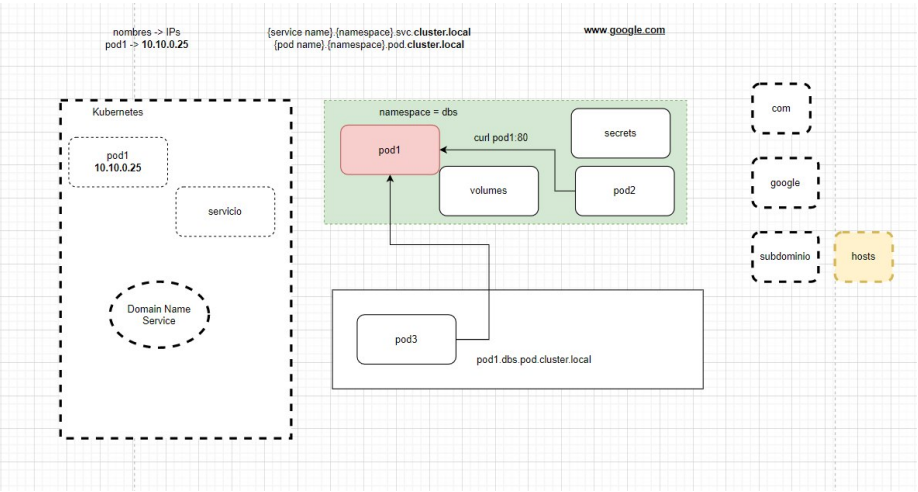
Esteban Ignacio Durán Vargas

2020388144

IC - 7602 - 2023 II Semestre

Componentes de Kubernetes

A continuación se definen múltiples componentes de Kubernetes.



Namespace Contenedor de recursos. En una misma instalación hay n namespaces. Contiene objetos (pods, volúmenes, secrets…) de Kubernetes que van a vivir dentro de él y serán exclusivos de este. Un pod en namespace diferente no puede acceder a un volumen de otro namespace. En la imagen se ve como el namespace dbs.

Domain Name Service

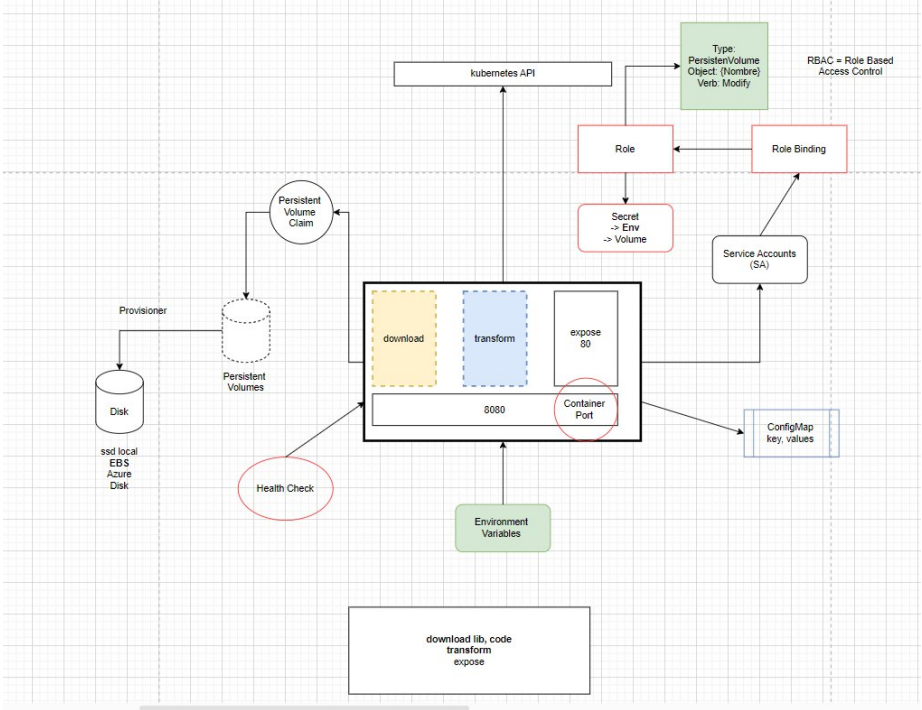
Pods y servicios tienen una dirección ip; una identificación prestada, no hay garantía que sean iguales cada vez que se levantan. El DNS mapea nombres a ips. Esto permite acceder a pods y servicios de manera no calificada. Si un pod se quiere comunicar con otro dentro del mismo namespace puede utilizar el nombre del host, sin calificar. Para comunicar entre namespaces es a través de pods o servicios por el nombre calificado.

El nombre calificado por defecto tiene la forma:

- **{service name}.{namespace}.svc.cluster.local** (para servicios).
- **{pod name}.{namespace}.pod.cluster.local** (para pods).

Pods

A continuación se presentan múltiples componentes de los pods en Kubernetes:



Múltiples contenedores, pero no todos estos contenedores correrán para siempre. En el ejemplo mostrado, download y transform solo se ejecutan una sola vez, tienen un ciclo de vida corto. A diferencia de estos, expose siempre corre y su contenedor está separado para optimizar los recursos. Si existiese un solo contenedor el cual contenga a los 3, se necesitan de libraries y código, para ejecutarse esa última vez. El propósito de los contenedores es que tengan solo lo necesario.

Persistent volumes, que son una abstracción que se mapean a un disco (partición local, nube, etc.). Para esto existe un **Provisioner** que genera la conexión entre el disco y el pod. Dentro de Kubernetes se le conoce como **Storage Classes** y básicamente son discos locales. El disco obtenido por el provisioner puede ser utilizado por un pod a la vez y por todos los pods (aún con más de uno corriendo). También existe el **Persistent Volume Claim** que es la configuración del disco generada por un provisioner y se comparte con todos los contenedores dentro de un pod.

Environment Variables, variables de entorno definidas dentro de Kubernetes con información conocida, utilizada por la aplicación. Pueden indicar información de configuración.

Service Accounts (SA) permiten que objetos como los pods hagan llamadas directas al API de Kubernetes para realizar configuraciones que necesiten (como aumentar el tamaño de un persistent volume), pero el permiso debe ser dado por un Service Account conectado al pod. **Roles** especifican permisos que tiene un rol según un Tipo, Objeto y Verbo; en el ejemplo de la imagen se muestra uno con tipo Persisten Volume y verbo Modify. **Role Binding** especifica la relación entre un Service Account y los roles. También sirven para acceder a **Secrets**. Los **Cluster Role Bindings** se utilizan para dar permisos globales. De esta manera, un pod puede hablar con el API de Kubernetes y no solo la porción del namespace.

Secrets, que son un tipo de objeto encriptado y tiene alta protección gracias a Roles y Role Bindings. Esto forma parte de RBAC (Role Based Access Control), para definir roles y regular acceso. Un secret puede exponerse a un puerto como variable de entorno o volumen persistente. Los secrets se pueden generar dinámicamente.

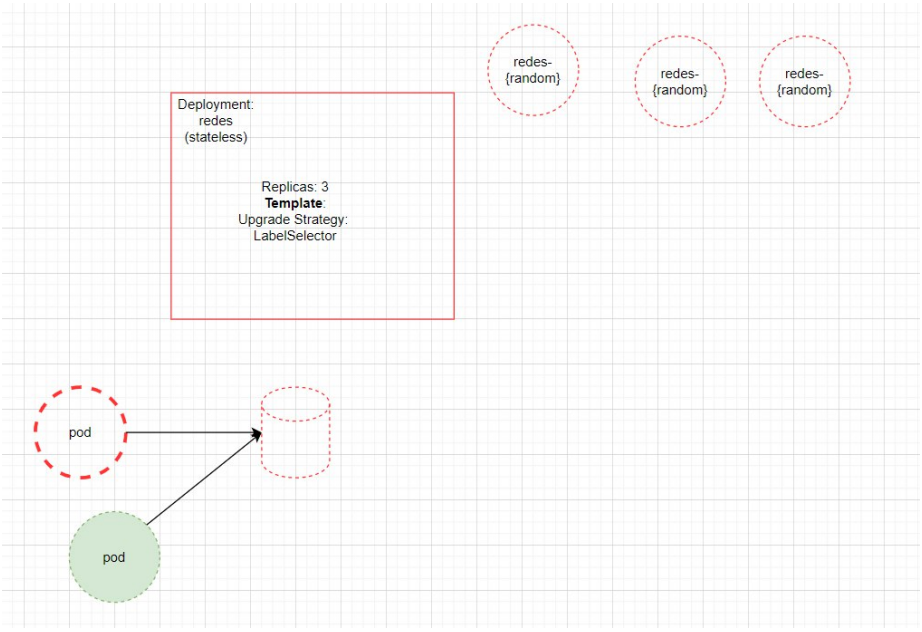
ConfigMap que es un almacenamiento de **key values**. Su acceso puede ser regulado por el API de Kubernetes o montarlo como volumen persistente.

Health Checks para los pods, para que Kubernetes esté al tanto de si un pod está funcionando correcto, sino levanta un pod nuevo.

Container Port que, cuando uno o más puertos se exponen en el pod, informa a Kubernetes que se exponen esos puertos en el pod.

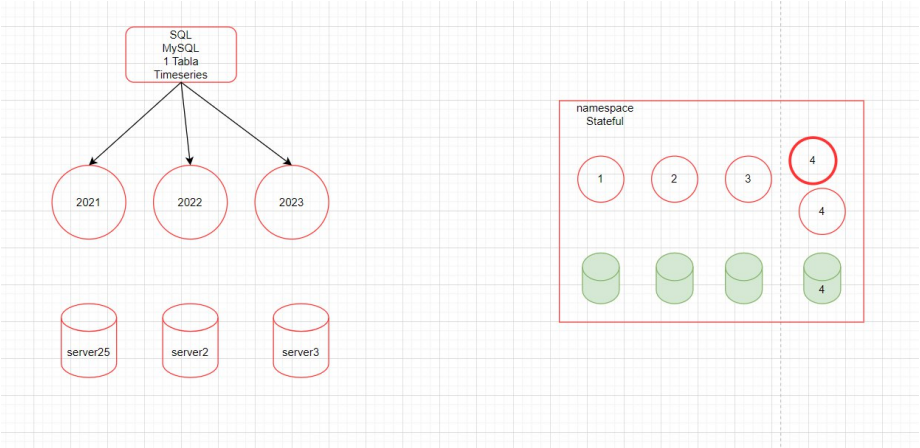
Deployment

Definen aspectos importantes para una aplicación, como:



- **Replicas**: define el número de réplicas o pods que tiene el deployment.
- **Template**: definición de alto nivel del pod.
- **Upgrade Strategy**: cómo actualizar la versión de software. Estas se pueden ver de los siguientes tipos:
  - Matar todos los pods y levantar los pods nuevos.
  - **Rolling update**: matar cierto número de pods y conservar los demás, esperar a que levante el nuevo y seguir con el proceso hasta que todos los pods estén actualizados.
- **Label Selector**: Indica al deployment cuáles pods le pertenecen. Por lo general son un string seguido de un número aleatorio. En el ejemplo podemos observar que se llaman redes, seguidos por un random.

Stateful Set



Los deployments por lo general son recursos **stateless**, que siginifica que no utilizan un disco. Si un pod se cae, otro con estos nombres se levanta; pero en este caso para utilizar un persistent volumen se debe tener el mismo nombre. También tienen sensibilidad al disco que estaban utilizando, aunque tengan el mismo nombre. Esto explica por qué no es buena práctica realizar comunicación entre 2 pods en diferentes namespaces, porque su nombre puede cambiar y mataría la conexión entre ambos. Esto se considera una mala práctica.

El Stateful Set, es un objeto de Kubernetes que corre dentro de un namespace que se asegura que los pods siempre reciban el mismo nombre. En el ejemplo se ve como cae el 4 y se levanta uno con el mismo nombre. Siempre los pods también utilizarán el mismo volumen, en el ejemplo también utiliza el volumen 4. No permite utilizar el mismo ip.

Replica Set

Evolución de los Deployments y Stateful Sets, donde se unen un solo.

Cron jobs

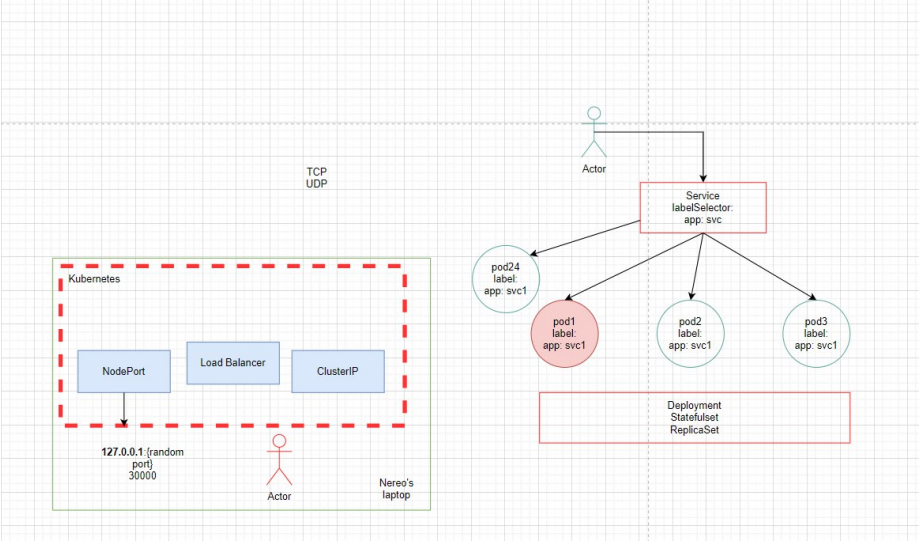
Tareas programadas que ejecuta un pod en un tiempo determinado. Estas tareas se les conocen como **jobs** que permite definir un pod que se ejecutan pero tiene una vida programada, y con el Cron Job se programa para ejecutarse según el usuario prefiera.

Resource Quotas

Define cuántos recursos (cpu, gpu, memoria, red, etc) va a usar un pod específico. Cuando se sobrepasa este límite, Kubernetes mata el pod, y si este corría con

un Deployment, Stateful Set o Replica Set; Kubernetes se encarga de levantar uno nuevo.

Services



Los objetos que se definen dentro de Kubernetes, no se pueden acceder dentro de la computadora Host (Nereo's Laptop). Es por eso que se usan Servicios para manejar el acceso a estos objetos. Para este se utiliza un labelselector que hace que el servicio se apodere de pods que compartan este label. Detrás de esos pods, existe un Deployment, Stateful Set o Replica Set que garantiza el número de pods corriendo a la vez. El servicio garantiza en incluír a los pods nuevos y excluír a los pods caídos. Esto puede ser visto en el ejemplo con el gráfico de la derecha donde el pod1 cayó y el pod24 se levantó. Existen los siguientes tipos:

- **ClusterIP:** Mantiene acceso dentro de Kubernetes. Puede ser usado para definir la comunicación entre 2 pods. Es la configuración por defecto.
- **NodePort:** Expone un puerto aleatorio de la máquina host (arriba del puerto 30000). Así un usuario desde la computadora host puede acceder al servicio. Se puede utilizar cualquier ip de la computadora.
- **Load Balancer:** Se comporta como nodeport, pero utiliza un puerto en específico y utiliza Load Balancers de Cloud Providers (solo se puede usar con estos). ### Puertos:
- **TCP:** Transmission Control Protocol, que se usa para conexiones persistentes queduran cierta cantidad de tiempo. En este importa saber que el mensaje llegó y lo espera.
- **UDP:** User Datagram Protocol, que se usa para conexiones stateless. En este no vigila si el mensaje llegó, no lo espera.

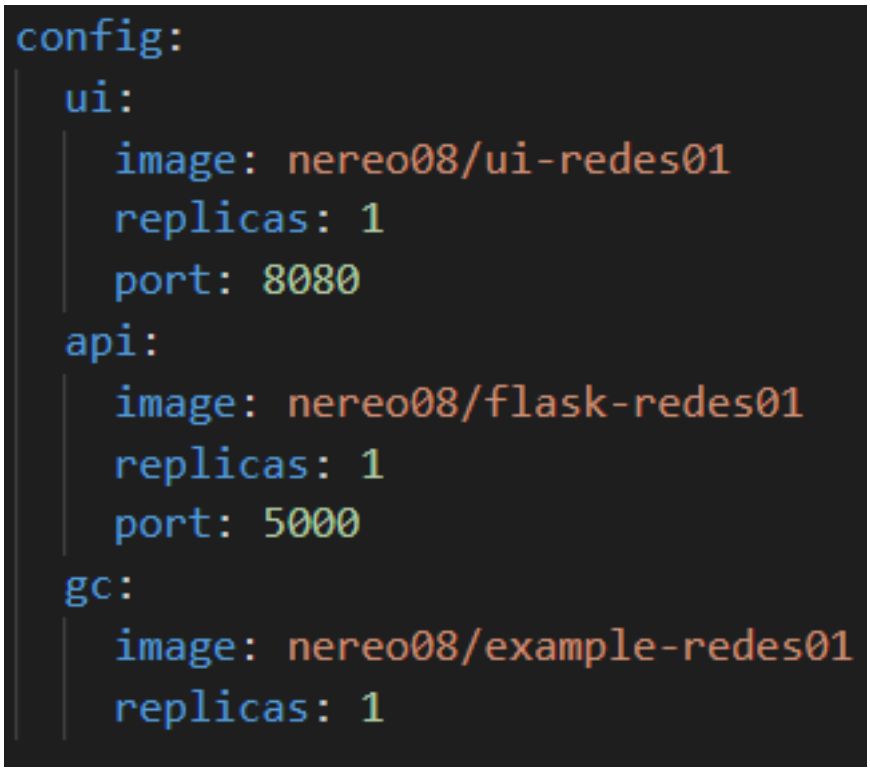
Ingresses

Proxy de capa 7. Se verá más adelante en el curso.

Custom Resource Definitions

Extiende el API de Kubernetes y añade un ciudadano de primer nivel que sea equivalente a un pod. De esta manera, se asegura que siempre se instalen los Deployment con todas sus características según se definieron.

Helm Charts



Templates de Lenguaje GoLand. Con esto se parametrizan configuraciones de Kubernetes. En este ejemplo mostrado por el profesor, se definen varios valores para los Kubernetes Manifest Files. Estos valores son accesibles en estos archivos de la siguiente manera, por ejemplo:

```
{{.Values.config.gc.replicas}}
```

Este se usa para acceder al número de réplicas definidas para el GameController. ## Creación de un Helm Chart Para crear un Helm Chart solo se necesita el comando helm y correr este comando:

```
helm create nombre
```

Instalación de un Helm Chart

Para instalar un Helm Chart solo se necesita el comando helm y correr este comando, donde nombre es un helm chart creado:

```
helm install nombre nombre
```

Instalación de un Helm Chart

Para actualizar un Helm Chart solo se necesita el comando helm y correr este comando, donde nombre es un helm chart previamente instalado:

```
helm upgrade nombre nombre
```