Exposición de aplicaciones fuera de Kubernetes

KEEPCODING
Tech School

Exponiendo PODs al exterior

- En esta sección:
  - Servicios - recordatorio
  - Accediendo a nuestras aplicaciones desde el exterior.
  - Ingress Controller & Ingress
  - Ejercicios

Servicios

KEEPCODING
Tech School

Recordatorio de Servicios (vistos en sección 4)

- Los servicios permiten **exponer nuestras aplicaciones (pods)** dentro del cluster y que otras aplicaciones puedan conectarse a ellas.
- Actúan a modo de **balanceador** de tráfico. Un servicio lleva asociado "siempre" una dirección IP (excepto headless services).
- Conceptos básicos
  - Nombre del servicio (DNS)
  - Tipo de servicio (ClusterIP, NodePort, LoadBalancer, …)
  - Selector de pods (para poder apuntar a los endpoints)
  - Puerto servicio
  - Puerto destino (targetPort)

Acceso a aplicaciones desde el exterior

- Normalmente, mucha de las aplicaciones que despleguemos en el clúster tendrán que ser accesibles desde el exterior.
- Con los servicios de tipo **ClusterIP** exponemos la aplicación internamente.
- **Headless** (ClusterIP=None) es un tipo especial de servicio donde Kube-proxy no interviene.
- Existen diversas formas para exponer aplicaciones al exterior:
  - Servicios NodePort / Load Balancer
  - Ingress Controllers + Ingress: muchos tipos
  - Service Meshes: Istio, Consul, Envoy, …

Ingress & Ingress Controllers

KEEPCODING
Tech School

Ingress Controller

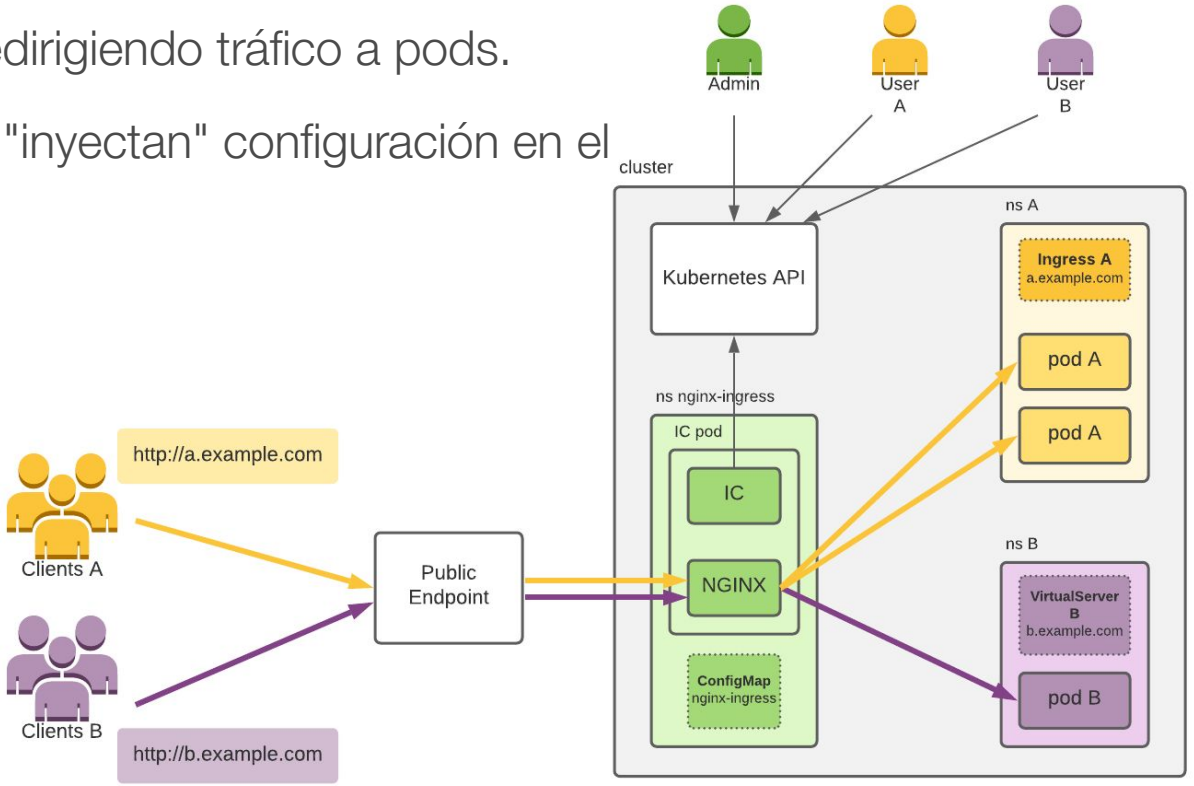
- El ingress controller es un **proxy inverso** accesible desde el exterior del cluster, que se configura a través de objetos **Ingress**.
    - Como muchas de las aplicaciones que se despliegan son HTTP/HTTPS se vio que era muy útil y necesario poder disponer de un **proxy inverso** potente y **configurable dinámicamente** para exponer nuestras aplicaciones, y así **no tener que crear NodePorts o Load Balancers dedicados para cada una de nuestras aplicaciones**.
- Para evitar que cada usuario tenga que diseñar y crear su propio proxy inverso se inventó el concepto de Ingress e Ingress Controller.
- El tipo de objeto **Ingress** no existe por defecto en Kubernetes (aunque ya es parte de la especificación oficial), es definido por el controlador via CRD (Custom Resource Definition).

Ingress Controller

- El Ingress Controller contiene:
  - **Proxy inverso** (capacidad de recibir peticiones HTTP y enviarlas a distintos "backends"). Es el controller en sí, y generalmente se expone al exterior vía LoadBalancer.
  - Definición de objetos (**CRD**) llamados `Ingress` que permitirán configurar en tiempo real el proxy inverso.
- Algunos Kubernetes as a service instalan por defecto un controlador, pero normalmente hay que elegir uno e instalarlo.
- Un `Ingress`, por lo tanto, será parte de la configuración del proxy inverso, son reglas para manejar el tráfico.

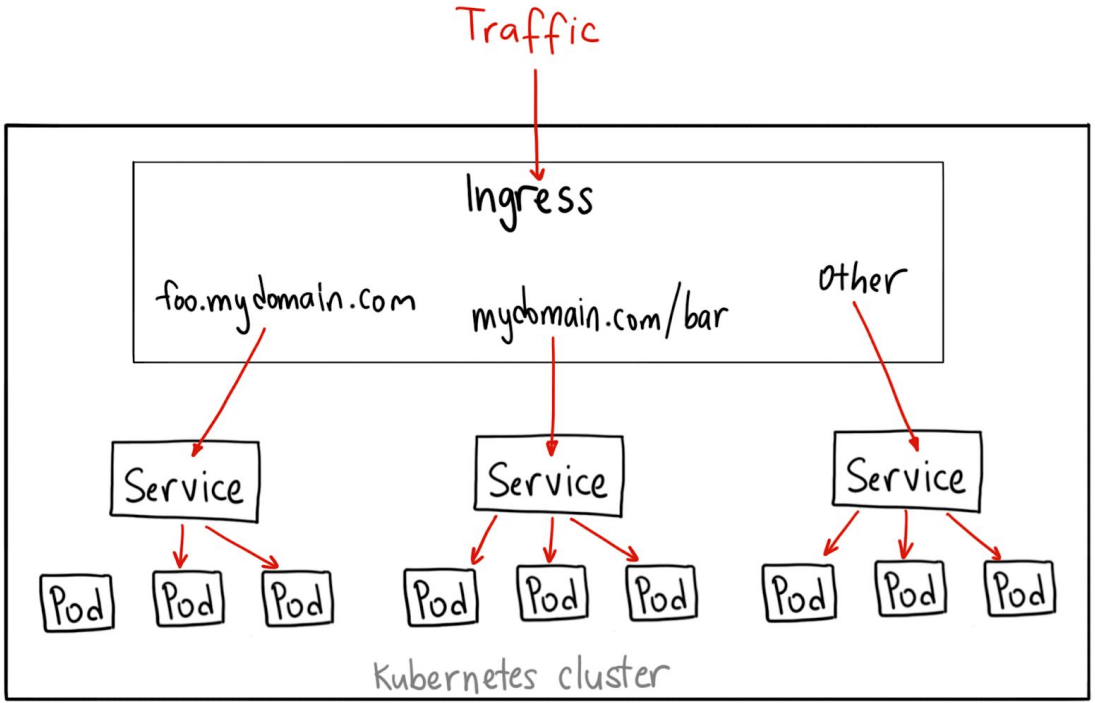
Ingress Controller

- Ingress Controller redirigiendo tráfico a pods.
- Los objetos Ingress "inyectan" configuración en el controller.



Ingress Controller

- Este diagrama muestra el concepto de Ingress, donde definiremos **reglas**, **virtual hosts,** etc.



Ingress Controller

- El único proceso que está en ejecución es el Ingress Controller. Los Ingress representan configuración que se inyecta en tiempo real al controlador, que es el proxy inverso.
- Que usemos Ingress no quiere decir:
  - Que no necesitemos crear un servicio para nuestra aplicación (**siempre será necesario al menos un "ClusterIP" para cada app**!)
  - Que no haga falta ningún NodePort o LoadBalancer (el propio Ingress Controller necesitará estar expuesto al exterior del cluster de alguna manera!).
- Si usamos Ingress no necesitaremos de servicios NodePort o LoadBalancer para nuestra aplicación (con ClusterIP será suficiente).

Ingress Controller

- Documentación oficial:
  - Ingress
  - NGINX Ingress controller (open source)
  - NGINX Ingress controller (NGINX Inc.)

Creando objetos Ingress - Demo

- Instalación Ingress Controller (NGINX open source)
- Desplegar workloads con sus servicios (varias apps)
- Crear recursos Ingress, casos de uso:
  - Todo el tráfico a un servicio
  - Simple fanout (basado en path de la URL)
  - Name based virtual hosting algo más complejo.
  - Seguridad: TLS termination & TLS pass-through
  - Exponiendo servicios no-HTTP: TCP/UDP

NGINX Ingress Controller (open source)

- Se instala por defecto en el namespace 'nginx-ingress', y se expone por defecto como LoadBalancer. Se puede instalar como Deployment o como DaemonSet.
- Podemos editar el manifiesto original para adaptar la configuración.
- Hay que considerar múltiples réplicas (HA y rendimiento), que no corran en los mismos nodos de kubernetes, etc.
- Podemos analizar los logs del controlador en tiempo real.
- La clase por defecto que lleva asociada es "nginx", que deberemos utilizar en nuestros objetos Ingress mediante la anotación `kubernetes.io/ingress.class: nginx`

Múltiples Ingress Controllers

- Se pueden tener varios controllers instalados, incluso del mismo tipo. Para ello usaremos las "Ingress Classes", y en cada objeto ingress tendremos que especificar la clase a la que pertenecen, para que sea procesado por el controlador específico.
- Con NGINX, si necesitamos un nuevo LoadBalancer para procesar nuestro tráfico (porque queremos un dominio / IP dedicado para una aplicación por ejemplo) simplemente instalaremos un nuevo controller siguiendo estas instrucciones.

Terminación TLS

- Podemos exponer servicios con HTTPs y un certificado (doc oficial).
- Para ello debemos guardar el certificado en un secreto especial de tipo TLS.
- Después ya podemos hacer uso del certificado SSL en nuestro ingress.

```
apiVersion: v1
kind: Secret
metadata:
  name: testsecret-tls
type: kubernetes.io/tls
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
```

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
  - hosts:
    - sslexample.foo.com
    secretName: testsecret-tls
  rules:
    - host: sslexample.foo.com
      http:
        paths:
        - path: /
          backend:
            serviceName: service1
              servicePort: 80
```

Más sobre Ingress Controllers

- Existen multitud de Ingress Controllers en el mercado:
  - Nginx: Ojo que hay 2 versiones:
    - Basada en Nginx open source, integrada en la doc oficial de Kubernetes (aquí).
    - La versión basada en Nginx Plus de la propia empresa (aquí).
  - HAProxy
  - Traefik
  - Envoy
  - GCLB (o ingress-gce), el de Google que viene con GKE.
    - Funciona de forma un poco especial, creando LBs para cada Ingress.

TCP Passthrough

- Los Ingress Controllers (proxies inversos) y los objetos Ingress están diseñados para trabajar con tráfico HTTP/HTTPS, por lo general no soportan tráfico TCP / UDP.
- Dependiendo del controlador igual podemos utilizar Ingress para otro tipo de protocolos TCP / UDP, pero no es lo común.
- Para ello deberemos documentarnos y configurar tanto el ingress controller como el objeto Ingress de forma adecuada.
- Ejemplo para nginx:
    - https://kubernetes.github.io/ingress-nginx/user-guide/exposing-tcp-udp-services/

Wildcard DNS

- Un DNS wildcard es un DNS que siempre resuelve al mismo sitio.
- Muy útil para no tener que estar configurando entradas DNS para los ingress.
- Podemos usar nip.io para las pruebas siempre que conozcamos la IP pública y esta no cambie.
- https://nip.io/
    - Por ejemplo **nginx.35-197-3-88.nip.io** se traducirá a "**35.197.3.88**" sin necesidad de tener un servidor DNS configurado.

https://kubernetes.io/docs/concepts/services-networking/#loadbalancer

Certificados Let’s Encrypt con Kubernetes

- cert-manager: herramienta muy potente para manejar certificados SSL en Kubernetes. Permite integración con sistemas ACME (Automatic Certificate Management Environment), como Letsencrypt.
- cert-manager ofrece muchas formas de integrar la generación de certificados SSL.
- Página oficial: https://cert-manager.io/docs/
- Durante la demo hemos visto conceptos básicos (creación de ClusterIssuer) y ejemplos de integración con Letsencrypt.

Ejercicios servicios

KEEPCODING
Tech School

Servicios - 1

- Crea un deployment y un servicio basado en la imagen 'docker.io/kennethreitz/httpbin' y exponlo fuera del clúster de diversas formas

Servicios - 2

- Crea un repositorio GIT con la aplicación Flask counter
- Asegúrate de que la imagen de docker de tu aplicación funciona y la has subido a dockerhub.
- Crea un directorio k8s
- Crea las definiciones de los objetos de Kubernetes necesarios para desplegar la aplicación.
- Despliega la aplicación
- Documenta con un README.md

Servicios - 3

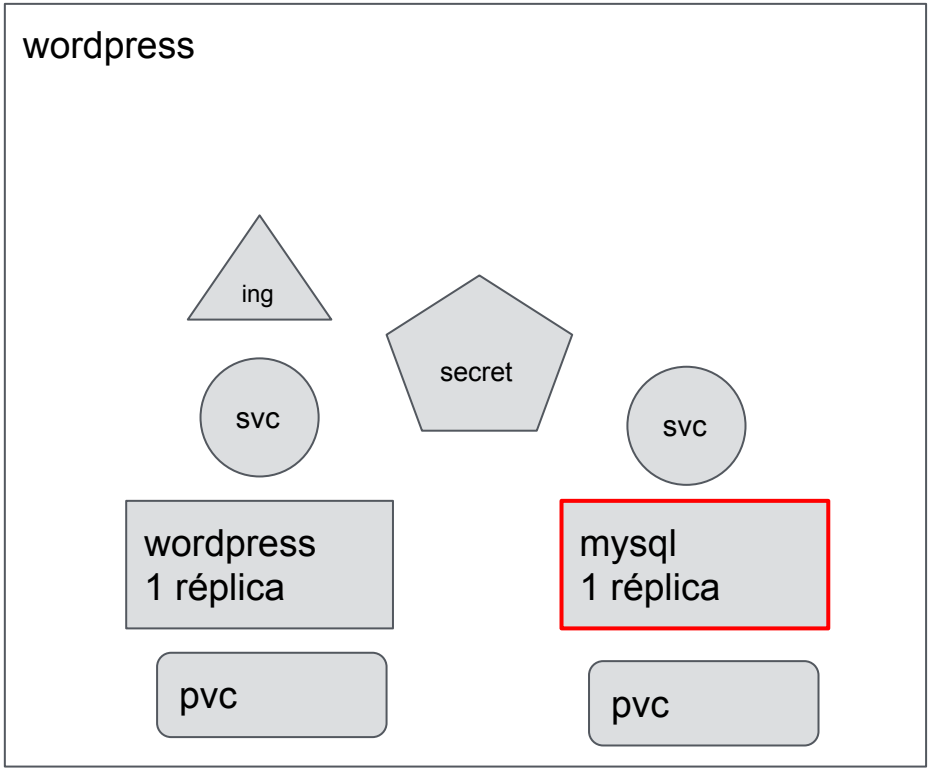
- Instala el Ingress Controller NGINX OpenSource manualmente:
  - https://kubernetes.github.io/ingress-nginx/deploy/

- Asegúrate de que el Ingress Controller se ha expuesto al exterior y toma nota de cómo se ha hecho (NodePort? LoadBalancer? qué puerto(s)?).

- Crea un par de aplicaciones y exponlas como Ingress en los paths (/app1) y (/app2) del acceso principal, apuntando a servicios web reales (como flask_counter, nginx, wordpress, apache, etc). Asegúrate de que estos Ingress están usando el controlador que has instalado y no otro que pudiera haber desplegado en el sistema!

Ejercicio completo

- Desplegar un Wordpress
  - Deployment (1 réplica)
    - + replicas > NFS (Filestore) o Bucket de GCS (Opcional)
  - MySQL (1 réplica)
    - Secretos (contraseña, etc)
    - Nota: Cloud SQL?
- Persistencia
  - PersitentVolumeClaim
    - storageclass: standard
- Ingress
  - DNS a elección (nip.io es válido)



KEEPCODING

Tech School

Madrid | Barcelona | Bogotá