Networking on Containers

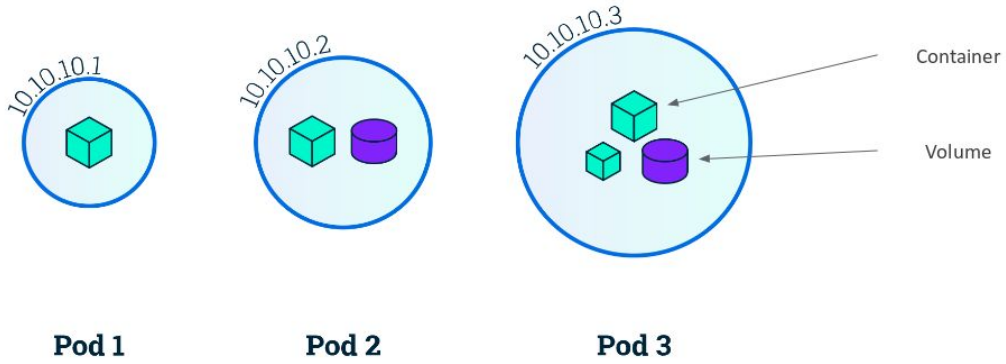
September 2018

ZEVENET
GROW WITHOUT LIMITS

High Availability
Load Balancing
Web Scale
Security
made easy

¿Qué es un pod?

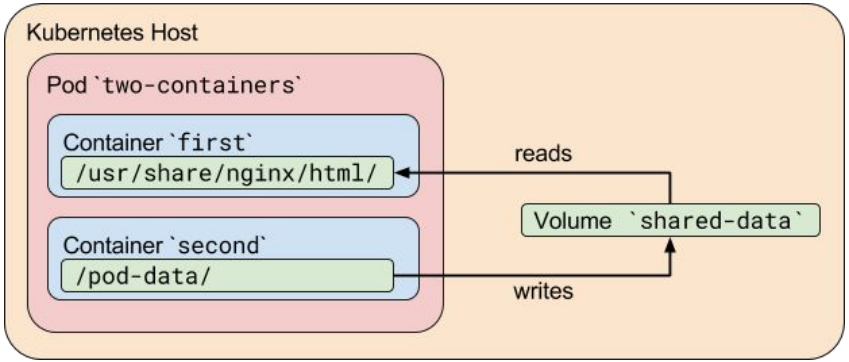
- ★ En Kubernetes, es un grupo de uno o más contenedores.
- ★ Un pod encapsula lo siguiente:
  - **Contenedor que corre una aplicación (o más de una con más contenedores).**
  - **Recursos de red/almacenamiento.**
  - **Reglas de funcionamiento.**



¿Qué es un pod?

- ★ Un Pod equivale a una instancia de una aplicación dada; si se necesitan múltiples instancias de una aplicación, será necesario **replicar** el Pod.
- ★ Estos múltiples Pods pueden ser manejados en grupo por una capa de abstracción llamada **Controller**, que no solo puede llegar a controlar la replicación de los Pods, también se puede encargar de mantener los Pods en continuo funcionamiento dentro de un cluster aún cuando falle un Node (un Node es una máquina de trabajo, ya sea física o VM). → ***Concepto self-healing***

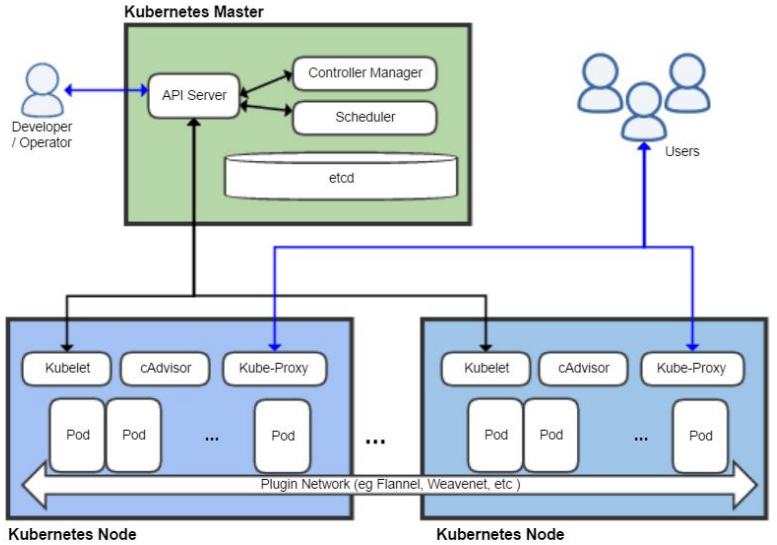
Esquema de un Node que contiene un Pod



¿Qué es un pod?

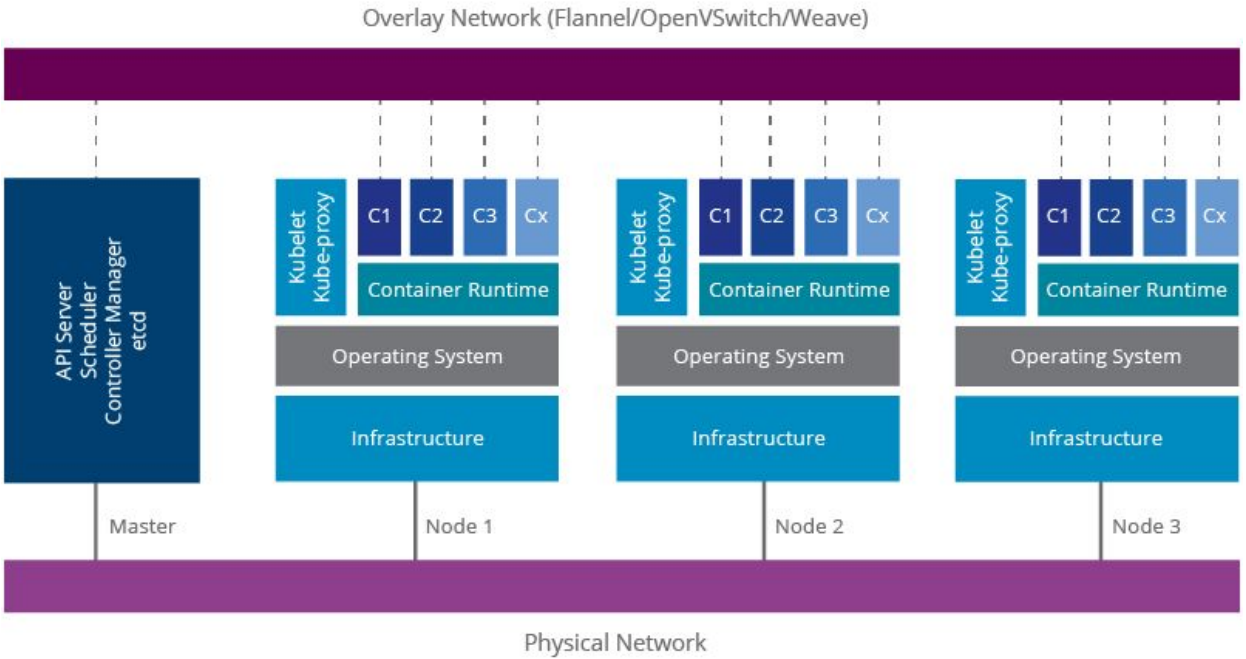
- La intención al crear un Pod no es que deba ser durable, al contrario: **un Pod debe ser efímero**, de poca duración.
- *Más detalles sobre Controller*: este se basa en un **Pod Template** del cual va creando Pods, y este Pod Template puede ser editado “en vivo” sin afectar a los Pods ya creados: los Pod Templates definen el estado de Pods futuros.

Esquema general de cómo funciona Kubernetes, donde se aprecia la capa Controller



¿Qué es un pod?

Otro esquema general de Kubernetes donde se aprecian sus elementos individuales, con otro tipo de detalles



¿Qué es un pod?

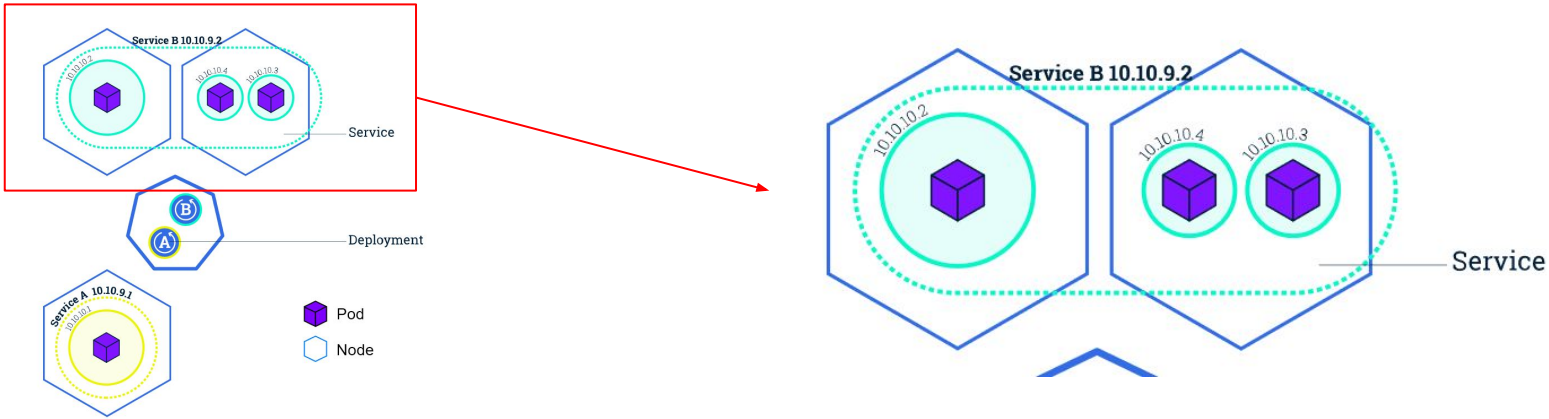
- En definitiva, un Pod equivale a un **entorno donde se ejecutan contenedores y donde se establecen las reglas de funcionamiento, no es un proceso en sí**, y Kubernetes se encarga de que los Pods alcancen el estado definido por el desarrollador.
- *Resumen de las diapositivas*:
  - Kubernetes se ejecuta en uno o varios…
  - **Nodes** → que forman un cluster y que contienen una capa de abstracción llamada…
  - **Controller** → que usa unas plantillas para crear Pods llamadas…
  - **Pod Templates** → de las cuales se crean…
  - **Pods** → que encapsulan contenedores, recursos y reglas.
- *Otras definiciones simplificadas*:
  - **Node** → máquina de trabajo (física o VM) que se comunica con la interfaz de Kubernetes.
  - **Controller** → controla grupos de Pods, y puede replicarlos y mantenerlos en funcionamiento.

¿Qué es un pod?

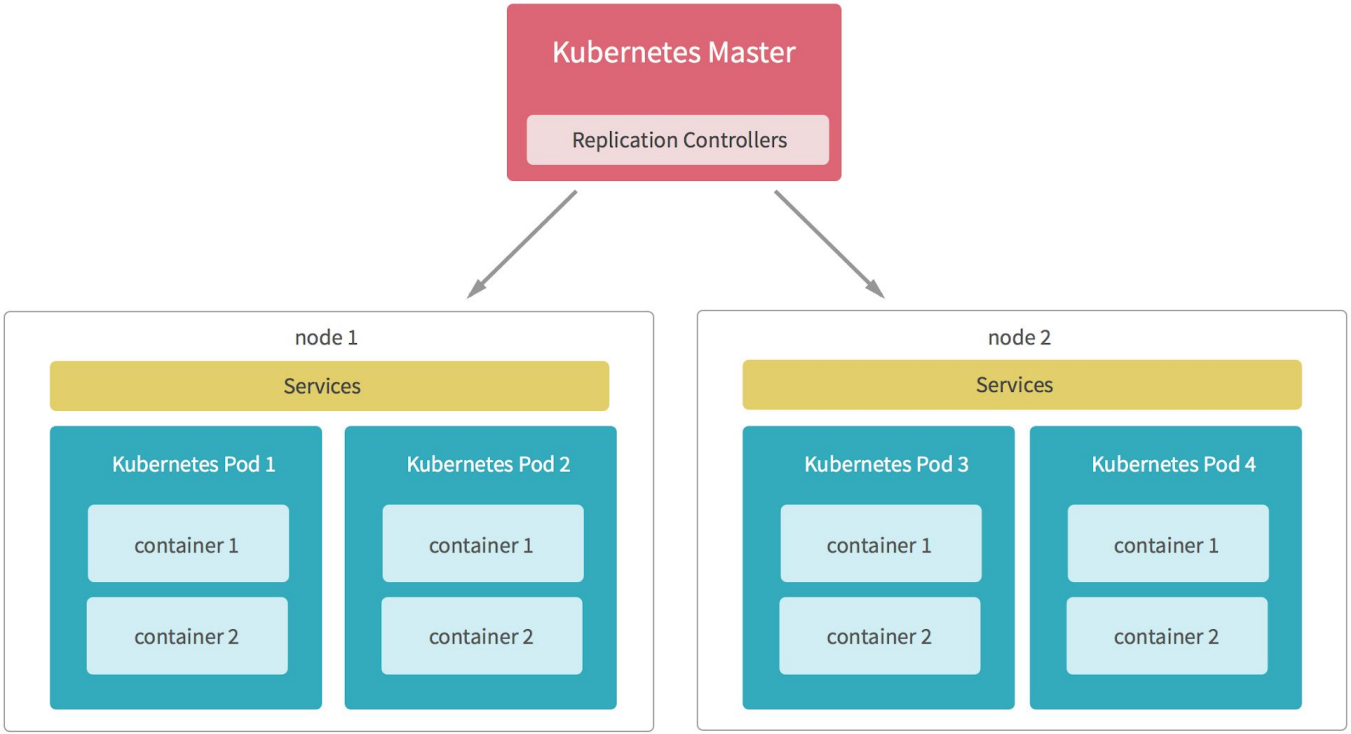
- ★ *Fuentes:*
  - https://kubernetes.io/docs/concepts/workloads/pods/pod/#what-is-a-pod
  - https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/
  - https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/
  - https://www.digitalocean.com/community/tutorials/an-introduction-to-kubernetes
  - https://kubernetes.io/docs/concepts/architecture/nodes/
  - https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16
  - https://linuxhint.com/kubernetes-nodes-and-pods/
  - https://medium.com/google-cloud/understanding-kubernetes-networking-pods-7117dd28727

¿Qué son servicios en kubernetes?

- ★ Un servicio en Kubernetes es **un recurso que permite acceder a un grupo de Pods y define cómo se acceden a ellos**, independientemente del Node en el que se localicen.
- ★ Cada Pod individual lleva una IP propia asignada dentro del Node donde se encuentren; un Service (el objeto instanciado por Kubernetes) **abstrae el set de Pods que se le indique y permite que los clientes frontend del cluster se desentiendan de dónde localizar los Pods que componen el backend**. Esto significa que al ser los Pods efímeros, las IPs que tienen no son estables y van cambiando; Service se encarga de solucionar este problema.



¿Qué son servicios en kubernetes?



¿Qué son servicios en kubernetes?

- Existen diferentes tipos de Services:
  - **ClusterIP:** expone dentro del cluster (no tiene contacto directo con el exterior) la IP del servicio. Es el modo por defecto u omisión.
  - **NodePort:** expone el servicio en cada IP del Node en un puerto estático. Permite contacto con el exterior del cluster.
  - **LoadBalancer:** expone el servicio al exterior usando un balanceador de carga de un cloud provider. Necesita Services de tipo NodePort y ClusterIP a los cuales le enrutará las conexiones.
  - **ExternalName:** mapea el servicio a lo indicado por el campo “externalName”. Como nota, requiere que la versión de kube-dns sea igual o superior a 1.7.
- Los 3 primeros tipos usan kube-proxy como responsable de implementar una forma de IPs virtuales.
- El recurso Service es un constructo de “capa 4” (TCP/UDP sobre IP).

¿Qué son servicios en kubernetes?

- El problema que se plantea con los Services es el siguiente: ¿cómo sabe Kubernetes el registro de Services que hay? ¿Qué elemento lleva ese registro? Hay dos formas distintas:
  - **Variables de entorno:** kubelet añade un set de variables de entorno por cada servicio activo. Esto implica que primero ha de crearse el servicio para que se pueda propagar correctamente las variables de entorno por los Pods.
  - **DNS:** a diferencia de las variables del entorno, no tiene la restricción de “crear el servicio antes que el Pod”. El servidor DNS observa la creación de Services a través de la API Server, y guarda el registro por cada servicio creado. Si se activa por todo el cluster, el servidor DNS puede resolver automáticamente cualquier búsqueda de nombre de los Services registrados. Esta forma es necesaria para usar ExternalName como tipo de Service, ya que depende de kube-dns.

¿Qué son servicios en kubernetes?

- *Fuentes:*
  - https://kubernetes.io/docs/concepts/services-networking/service/
  - http://enmilocalfunciona.io/introduccion-a-kubernetes-i/
  - https://medium.com/google-cloud/understanding-kubernetes-networking-services-f0cb48e4cc82
  - https://blog.codeship.com/getting-started-with-kubernetes/
  - https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/
  - https://wso2.com/whitepapers/a-reference-architecture-for-deploying-wso2-middleware-on-kubernetes/

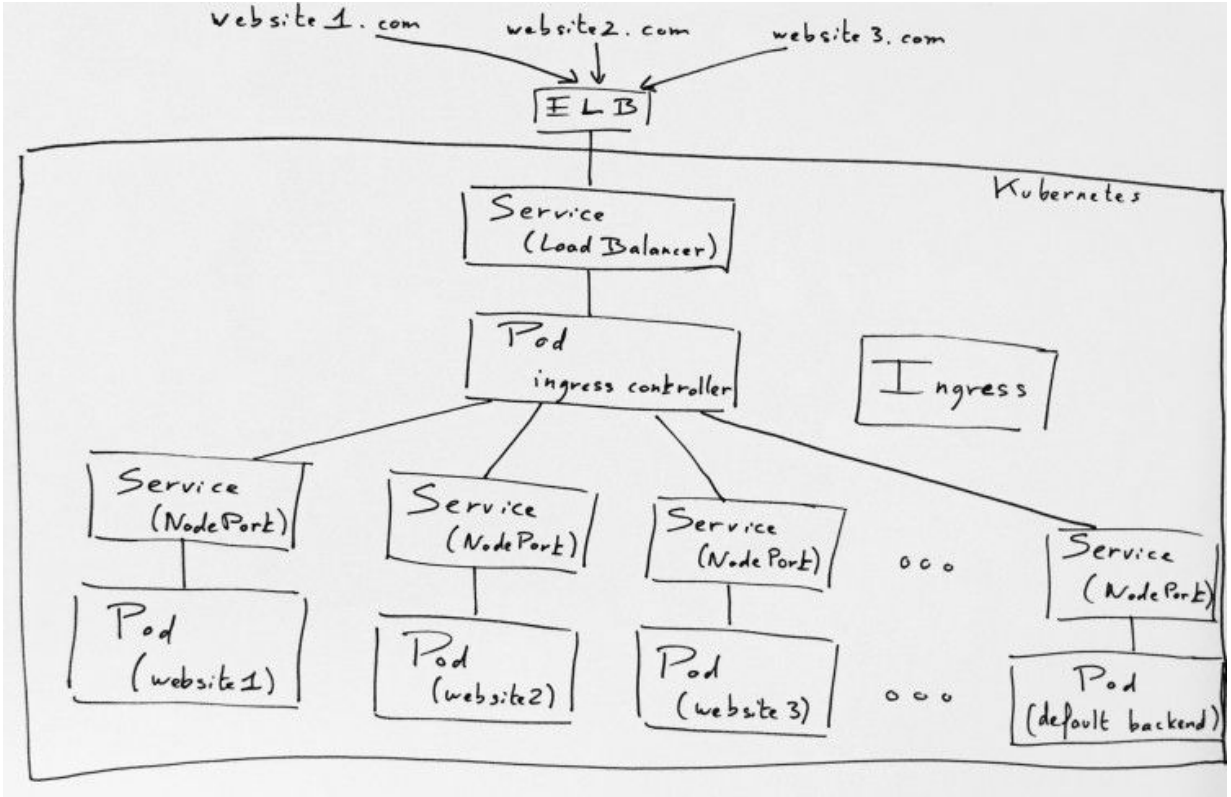
¿Qué es y cómo se usa Ingress en kubernetes?

- ★ Ingress en Kubernetes es el recurso que permite **enrutar conexiones desde el exterior del cluster a servicios dentro de éste**, y que contiene un conjunto de reglas para ello.
- ★ Se divide en dos componentes:
  - **Ingress Controller:** resuelve la petición a cualquier usuario que pida acceso a un recurso Ingress; es decir, sin un Ingress Controller, los Ingresses Resources no servirían para nada, ya que es el Controller quien dirige el tráfico.
  - **Ingress Resource:** indica a qué Services debe ser dirigido el tráfico según el hostname o ruta.
- ★ Un Ingress puede montar fácilmente un balanceador de carga, además de configurar un servidor proxy para múltiples Services.
- ★ Los Services a los que debe enrutar Ingress han de ser de tipo NodePort o ClusterIP.
- ★ Ingress es un recurso de “capa 7” (Aplicación).

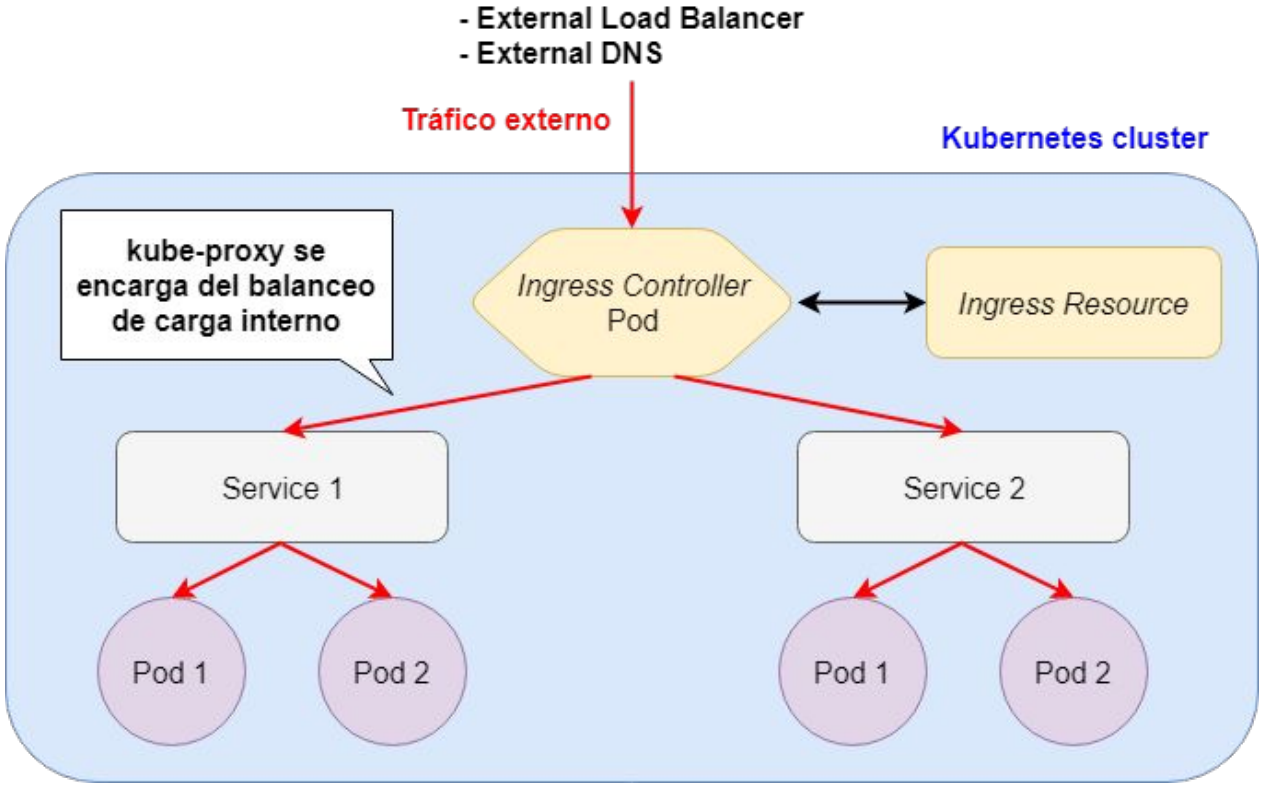
¿Qué es y cómo se usa Ingress en kubernetes?

- ★ Diferencias entre Ingress y LoadBalancer:
  - El motivo de un Service de tipo LoadBalancer es ofrecer acceso exterior a un único servicio del cluster. Ingress, en contraste, **permite acceder a múltiples servicios**.
  - Ingress soporta **terminación TLS**, LoadBalancer no.
  - Ingress soporta **"path-based routing"**, LoadBalancer no (recordemos que LoadBalancer da acceso exterior a un solo servicio, por lo que este punto es algo casi evidente).
  - Ingress soporta **"name-based virtual hosts"** (que usa múltiples hostnames para una sola IP), LoadBalancer no.
- ★ Con algunos Ingress Controllers (como NGINX, usado como referencia por Kubernetes), **el mismo Controller configura el servidor proxy (que circunvala kube-proxy)** usando Endpoints API.
- ★ A veces se utiliza un Service de tipo LoadBalancer para no tener una única réplica del Ingress Controller, de modo que diferentes réplicas se distribuyen en diferentes nodos.

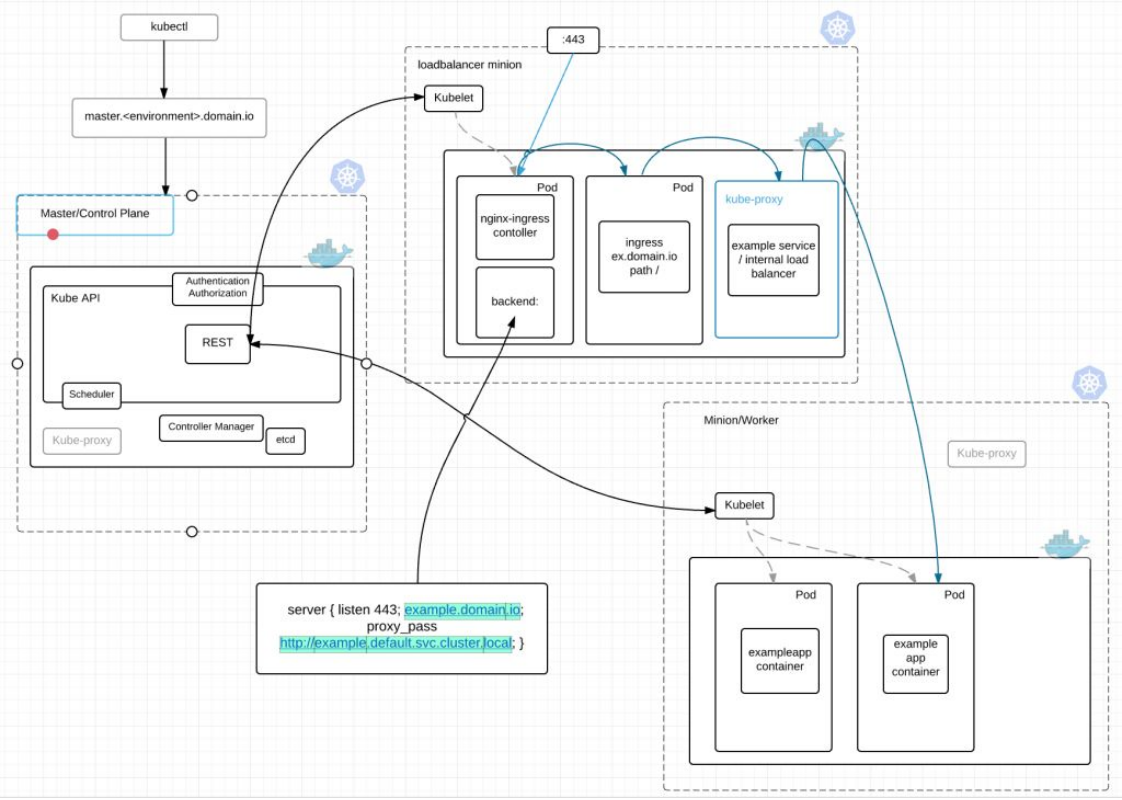
¿Qué es y cómo se usa Ingress en kubernetes?



¿Qué es y cómo se usa Ingress en kubernetes?



¿Qué es y cómo se usa Ingress en kubernetes?



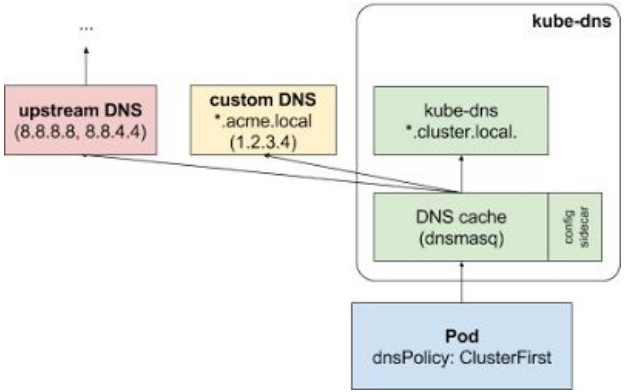
¿Qué es y cómo se usa Ingress en kubernetes?

★ *Fuentes:*

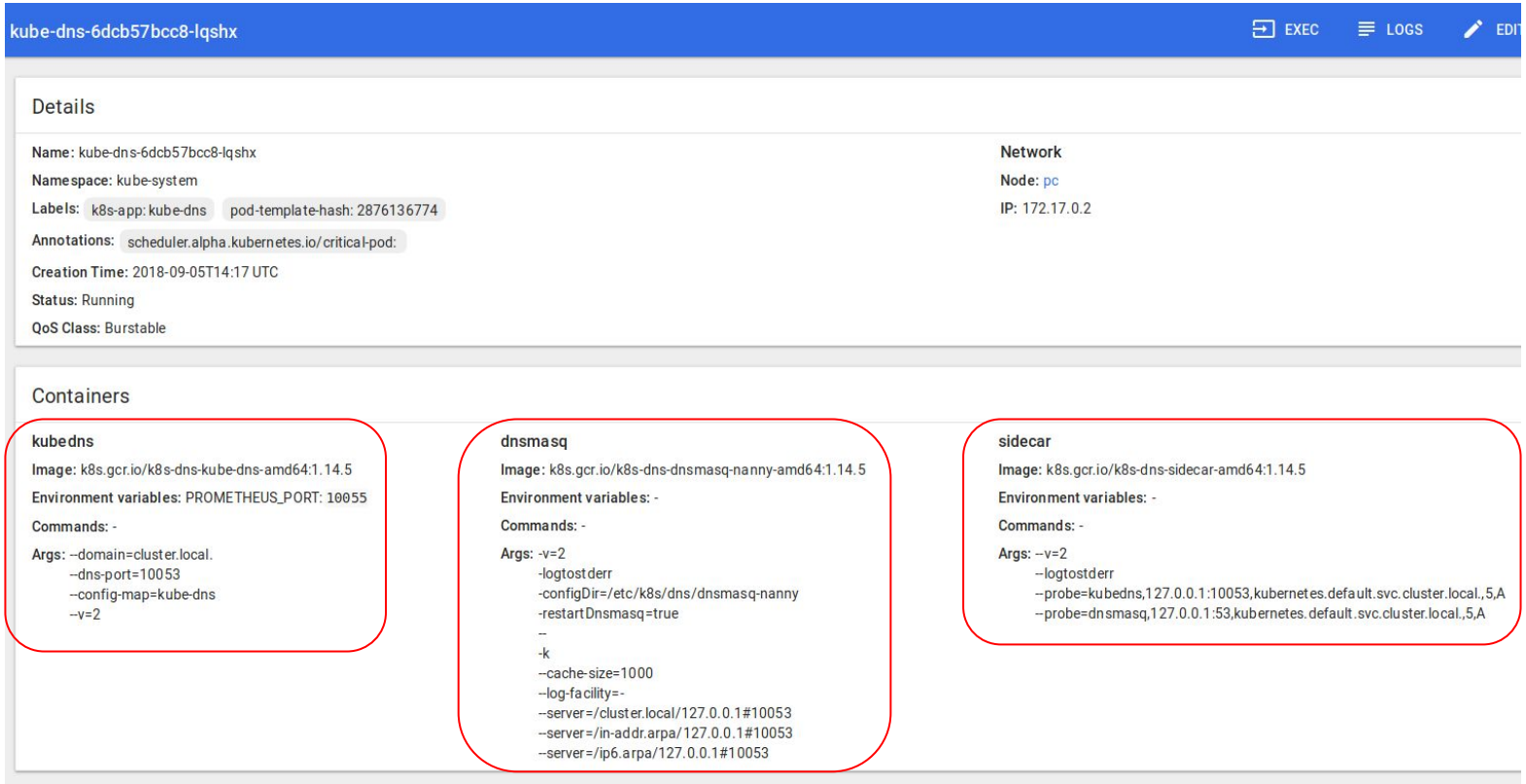
- https://kubernetes.io/docs/concepts/services-networking/ingress/
- https://medium.com/google-cloud/understanding-kubernetes-networking-ingress-1bc341c84078
- https://medium.com/@Oskarr3/setting-up-ingress-on-minikube-6ae825e98f82
- https://www.sandtable.com/a-single-aws-elastic-load-balancer-for-several-kubernetes-services-using-kubernetes-ingress/
- https://codeblog.dotsandbrackets.com/kubernetes-example/
- https://twitter.com/rothgar/status/898220641954013184
- https://www.slideshare.net/imesh/deep-dive-into-kubernetes-part-1
- https://stackoverflow.com/questions/50966300/whats-the-difference-between-exposing-nginx-as-load-balancer-vs-ingress-control
- https://medium.com/@awkwardferny/getting-started-with-kubernetes-ingress-nginx-on-minikube-d75e58f52b6c
- https://github.com/kubernetes/ingress-nginx/issues/257

¿Qué son kube-dns y external-dns?

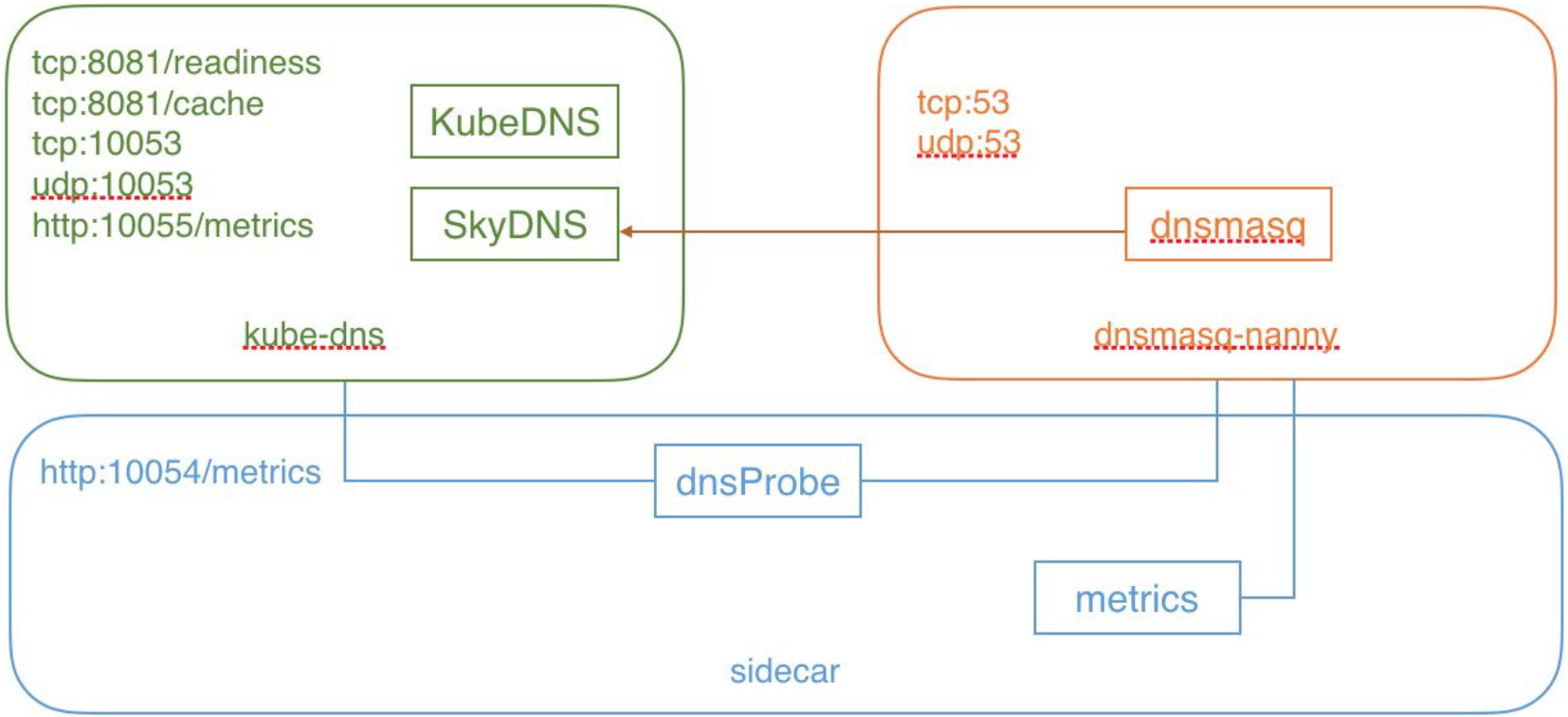
- ★ kube-dns es el componente **(Service y Pod)** instanciado por Kubernetes DNS para la **resolución DNS del cluster**. Actúa como servidor DNS y observa constantemente qué Services y Pods se crean y se destruyen para mantener el registro actualizado, además de resolver búsquedas de nombres.
- ★ El Pod de kube-dns se divide en 3 contenedores que realizan una funcionalidad diferente:
  - **kubedns:** contenedor principal, observa los cambios en Services y Endpoints y resuelve peticiones DNS.
  - **dnsmasq:** proxy de kubedns que actúa como caché DNS para mejorar el rendimiento.
  - **sidecar:** exporta métricas y realiza chequeos de estado en sistemas DNS.



¿Qué son kube-dns y external-dns?



¿Qué son kube-dns y external-dns?



¿Qué son kube-dns y external-dns?

- external-dns es un add-on similar a kube-dns, que **hace visible a servidores DNS públicos los recursos del cluster (Services, Ingresses, etc)**. Recibe una lista de recursos a través de la API de Kubernetes para determinar una lista deseada de registros DNS.
- A diferencia de kube-dns, **no es un servidor DNS en sí**; configura otros proveedores de DNS.
- Los servicios que permite tener expuestos son de tipo LoadBalancer (servicios de tipo NodePort todavía no, pero está en desarrollo).

¿Qué son kube-dns y external-dns?

- ★ *Fuentes:*
  - https://kubernetes.io/docs/tasks/administer-cluster/dns-custom-nameservers/
  - https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/
  - https://github.com/kubernetes/dns
  - https://github.com/kubernetes/dns/blob/master/docs/sidecar/README.md
  - https://github.com/kubernetes/minikube/tree/master/deploy/addons/kube-dns
  - https://blog.sophaskins.net/blog/misadventures-with-kube-dns/
  - https://github.com/kubernetes-incubator/external-dns

¿Qué es NetworkPolicy vs PodReady++?

- ★ NetworkPolicy es un recurso de Kubernetes que **controla el tráfico de un grupo determinado de Pods**.
- ★ El tráfico se controla mediante una especificación de reglas que reúne un recurso NetworkPolicy. La existencia de este recurso indica que **por defecto, no se regula qué tráfico circula por los Pods**.
- ★ NetworkPolicy no es lo mismo que el recurso Ingress:
  - La diferencia fundamental es que **Ingress dirige** el tráfico del exterior del cluster al interior, mientras que **NetworkPolicy regula** si este tráfico debe entrar en los Pods o bloquea su salida.

Ingress:
- une
- posibilita
- comunica

NetworkPolicy:
- controla
- permite
- deja paso

¿Qué es NetworkPolicy vs PodReady++?

- ★ Para entender qué es PodReady++, primero se ha de entender cuál es el problema que intenta solucionar este recurso.
- ★ **Un Pod lleva asociado un objeto PodStatus** con varios atributos, **que indican el estado de un Pod** según comprobaciones distintas realizadas del Pod.
- ★ El objetivo de lo anterior es indicar **si un Pod está preparado (ready) o no**, y si no lo está, que indique a qué se debe el error.
- ★ Entonces, el problema se simplifica del siguiente modo:
  - Concepto “Pod readiness” → Indica cuándo está listo el Pod para servir tráfico.
  - La definición de “Pod readiness” actual → Si los contenedores están listos, el Pod está listo.
  - **¿Qué ocurre cuando el Pod está listo, pero los servicios con los que se comunica no, ni los NetworkPolicy que le afectan?**

¿Qué es NetworkPolicy vs PodReady++?

- Si se cambia la definición de “Pod readiness” teniendo en cuenta elementos externos al Pod, quedaría así:
  - Si los contenedores están listos y se cumplen las condiciones indicadas en ReadinessGates, entonces el **Pod está listo++ (status ready++)**.
- ReadinessGates es el elemento de un pod spec donde se indican las condiciones extra que se han de tener en cuenta para decidir si un Pod está listo o no.
- En conclusión: **PodReady++ es una extensión del actual “Pod readiness” que permite retroalimentación externa**.
- La cuestión “NetworkPolicy vs PodReady++” surge de cómo debe afectar NetworkPolicy a PodReady++ (diferentes debates y soluciones), como por ejemplo:
  - Una solución dinámica como tener en cuenta todos los NetworkPolicy que se refieren a un set de Pods no es escalable.
  - Se considera como parte de una solución mejor el tenerlos en cuenta sólo si los recursos están disponibles.
- El debate sigue abierto, por lo que no hay solución definida a día de hoy.

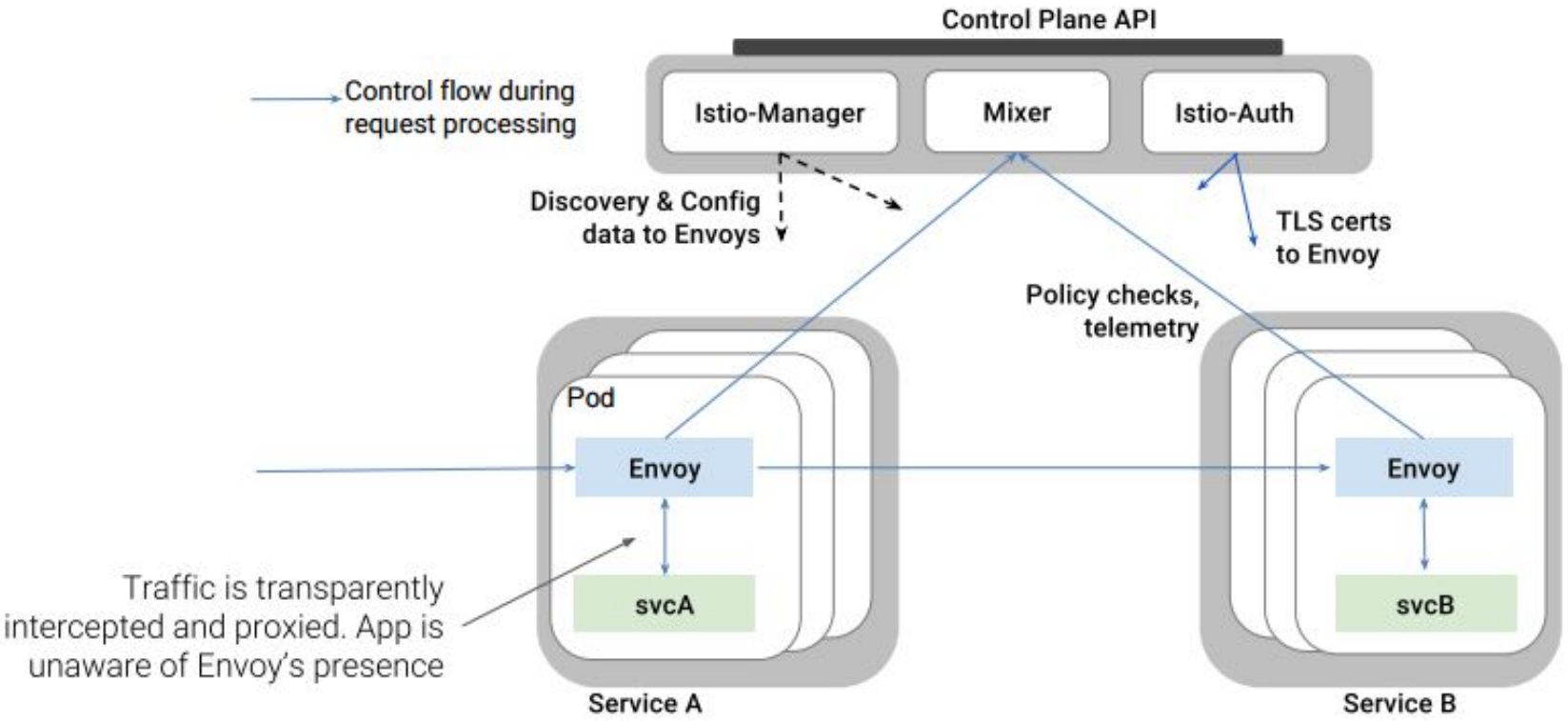
¿Qué es NetworkPolicy vs PodReady++?

- *Fuentes:*
  - https://kubernetes.io/docs/concepts/services-networking/network-policies/
  - https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/
  - https://cloud.google.com/kubernetes-engine/docs/tutorials/network-policy
  - https://groups.google.com/forum/#!topic/kubernetes-sig-network/A8oyMrWpJ8Q
  - https://github.com/kubernetes/community/blob/master/keps/sig-network/0007-pod-ready%2B%2B.md
  - https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/
  - https://docs.google.com/document/d/1pe-s1NHNkzsKYD-CNMnT0AQaa7je3_B0B4ZjJP8kI9o

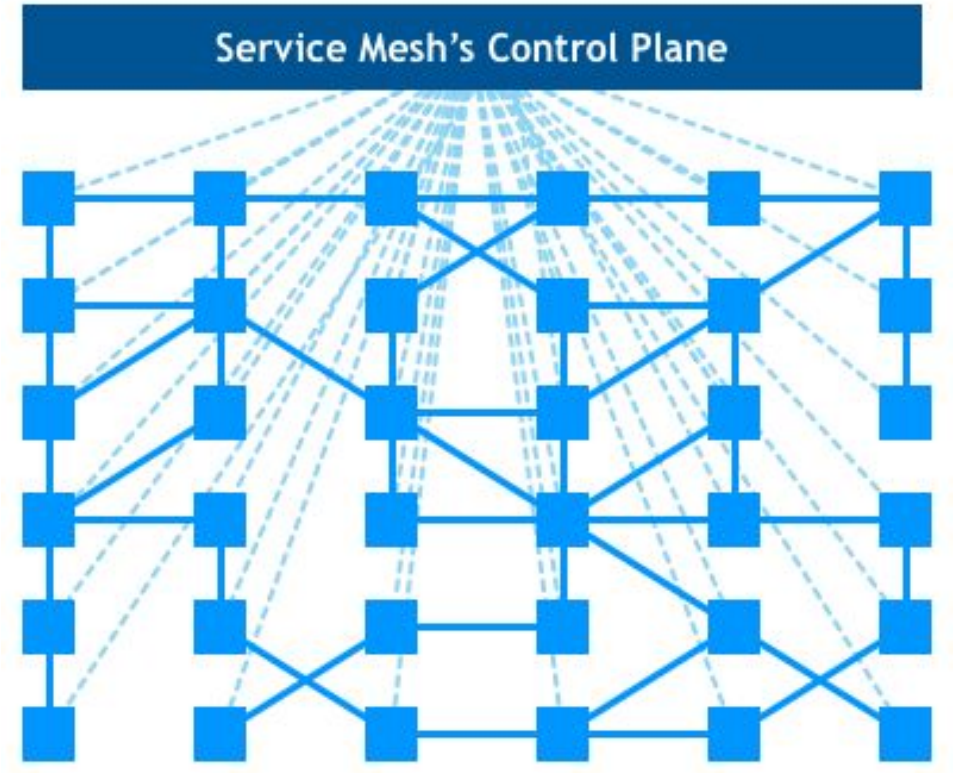
¿Qué es Network Service Mesh?

- Para comprender primero qué es un Network Service Mesh, primero se debe entender de dónde surge el término Service Mesh: **un Service Mesh es una capa de abstracción de la comunicación servicio-a-servicio**, de forma que la vuelve una entidad consolidada y que se puede controlar y observar con mayor detalle..
- Un Service Mesh se divide en dos componentes:
  - **Control Plane:** elemento que controla/vigila la comunicación en Data Plane, externo a los propios Pods y contenedores.
  - **Data Plane:** elemento donde ocurre la comunicación entre servicios.
- Dentro del Data Plane, donde ocurre la comunicación, **hay proxies que remiten la información hacia el Control Plane**, donde se decide cómo dirigir el tráfico (configura los proxies). Estos proxies normalmente se encuentran dentro del Pod como un contenedor “inyectado” **(sidecar container/proxy)** o como un Pod aparte, dependiendo de la estructura del Service Mesh.
- Se puede considerar como **un modelo de networking que se asienta sobre TCP/IP como una abstracción de la misma**.

¿Qué es Network Service Mesh?



¿Qué es Network Service Mesh?



¿Qué es Network Service Mesh?

- Propiedades principales que puede tener un Service Mesh:
  - **Balanceo de carga.**
  - **Routing.**
  - **Health checking.**
  - **Autenticación y autorización.**
  - **Observación y medición de la red.**
  - **Service discovery.**
- Como detalle, **todo lo anterior lo realiza el Data Plane**. Por defecto, cuando Kubernetes usa NGINX como Ingress Controller, realmente se está usando Data Plane dentro del cluster sin Control Plane. **Control Plane es un componente separado de Data Plane**, que realiza las funciones de control, configuración y recolección de métricas por lo general.

¿Qué es Network Service Mesh?

Ejemplo de Service Mesh en funcionamiento:
- Istio (Control Plane)
- Envoy (Data Plane)

Instalo Istio y veo que todo funciona.

```
root@pc:~/istio-1.0.2# kubectl get svc -n istio-system
NAME                       TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
     AGE
grafana                    ClusterIP      10.96.213.117    <none>        3000/TCP
     17s
istio-citadel              ClusterIP      10.99.51.100     <none>        8060/TCP,9093/TCP
     16s
istio-egressgateway        ClusterIP      10.99.168.116    <none>        80/TCP,443/TCP
     17s
istio-galley               ClusterIP      10.109.208.112   <none>        443/TCP,9093/TCP
     17s
istio-ingressgateway       LoadBalancer   10.98.41.161     <pending>     80:31380/TCP,443:31390/TCP,31400:31400/TCP,15011:32592/TCP,
CP   17s
istio-pilot                ClusterIP      10.96.177.10     <none>        15010/TCP,15011/TCP,8080/TCP,9093/TCP
     16s
istio-policy               ClusterIP      10.99.122.110    <none>        9091/TCP,15004/TCP,9093/TCP
     17s
istio-sidecar-injector     ClusterIP      10.100.198.147   <none>        443/TCP
     16s
istio-statsd-prom-bridge   ClusterIP      10.98.70.64      <none>        9102/TCP,9125/UDP
     17s
istio-telemetry            ClusterIP      10.111.69.32     <none>        9091/TCP,15004/TCP,9093/TCP,42422/TCP
     17s
jaeger-agent               ClusterIP      None             <none>        5775/UDP,6831/UDP,6832/UDP
     14s
jaeger-collector           ClusterIP      10.102.239.120   <none>        14267/TCP,14268/TCP
     14s
jaeger-query               ClusterIP      10.110.38.89     <none>        16686/TCP
     14s
prometheus                 ClusterIP      10.105.29.148    <none>        9090/TCP
     16s
servicegraph               ClusterIP      10.104.172.154   <none>        8088/TCP
     16s
tracing                    ClusterIP      10.110.85.183    <none>        80/TCP
     14s
zipkin                     ClusterIP      10.96.238.55     <none>        9411/TCP
     14s
```

ZEVENET

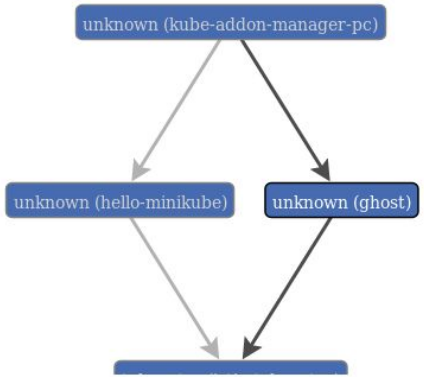
¿Qué es Network Service Mesh?

```
root@pc:~/istio-1.0.2# kubectl get pods -n istio-system
NAME                                      READY     STATUS              RESTARTS   AGE
grafana-6cbdcfb45-4z95k                   0/1       ContainerCreating   0          1m
istio-citadel-6b6fdfdd6f-82dcc            0/1       ContainerCreating   0          1m
istio-cleanup-secrets-hxtth               0/1       ContainerCreating   0          1m
istio-egressgateway-56bdd5fcfb-tc9gl      0/1       ContainerCreating   0          1m
istio-galley-96464ff6-qjtfm               0/1       ContainerCreating   0          1m
istio-grafana-post-install-dvw28          0/1       ContainerCreating   0          1m
istio-ingressgateway-7f4dd7d699-lv7j6     0/1       ContainerCreating   0          1m
istio-pilot-6f8d49d4c4-m9fkn              0/2       ContainerCreating   0          1m
istio-policy-67f4d49564-jvw2q             0/2       ContainerCreating   0          1m
istio-sidecar-injector-69c4bc7974-4l4w2   0/1       ContainerCreating   0          1m
istio-statsd-prom-bridge-7f44bb5ddb-rkln5 0/1       ContainerCreating   0          1m
istio-telemetry-76869cd64f-jv2qq          0/2       ContainerCreating   0          1m
istio-tracing-ff94688bb-wk7nr             0/1       ContainerCreating   0          1m
prometheus-84bd4b9796-79vxr               0/1       ContainerCreating   0          1m
servicegraph-c6456d6f5-pmc68              0/1       ContainerCreating   0          1m
```

¿Qué es Network Service Mesh?

Aquí se ve después de usar Apache Bench para provocar tráfico en hello-minikube y ghost.

El diagrama indica el tráfico a nivel de servicios
(como se indicaba, Service Mesh abstrae comunicación servicio-a-servicio).



Close

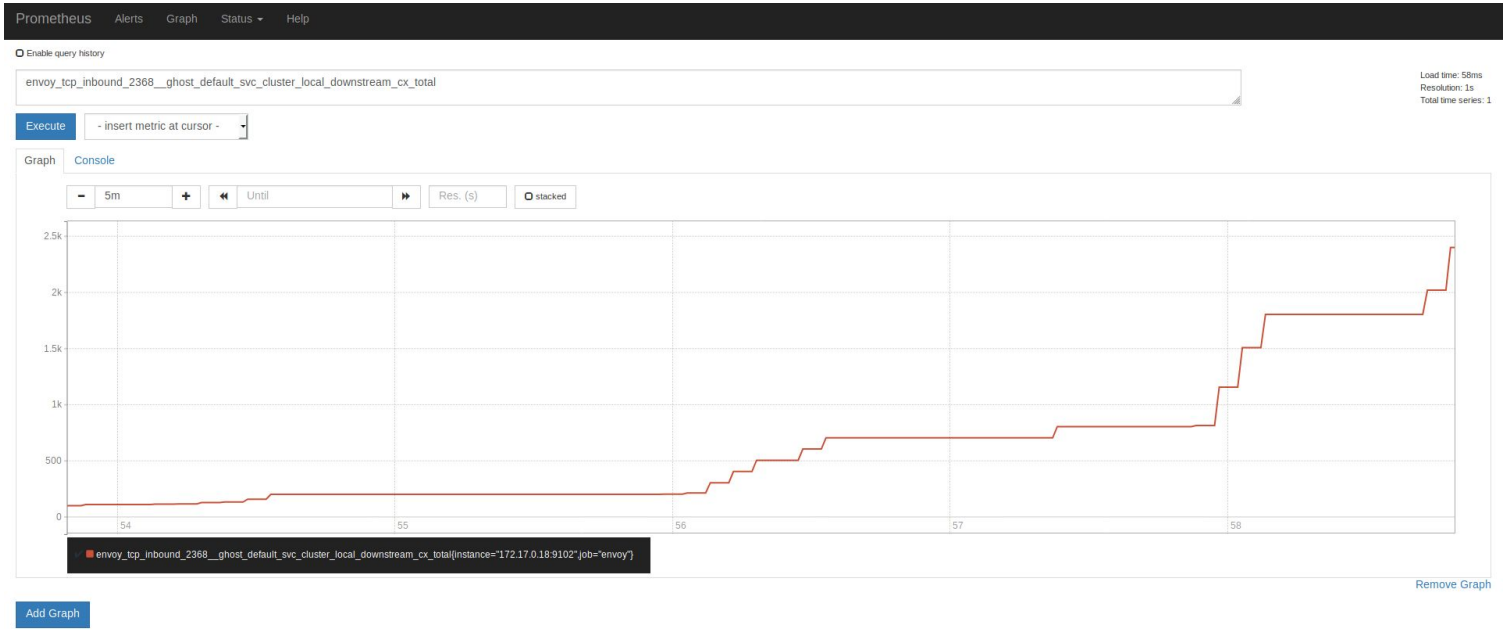
unknown (ghost)

Incoming Connections	Reqs/sec
unknown (kube-addon-manager-pc)	

Outgoing Connections	Reqs/sec
telemetry (istio-telemetry)	0.264880

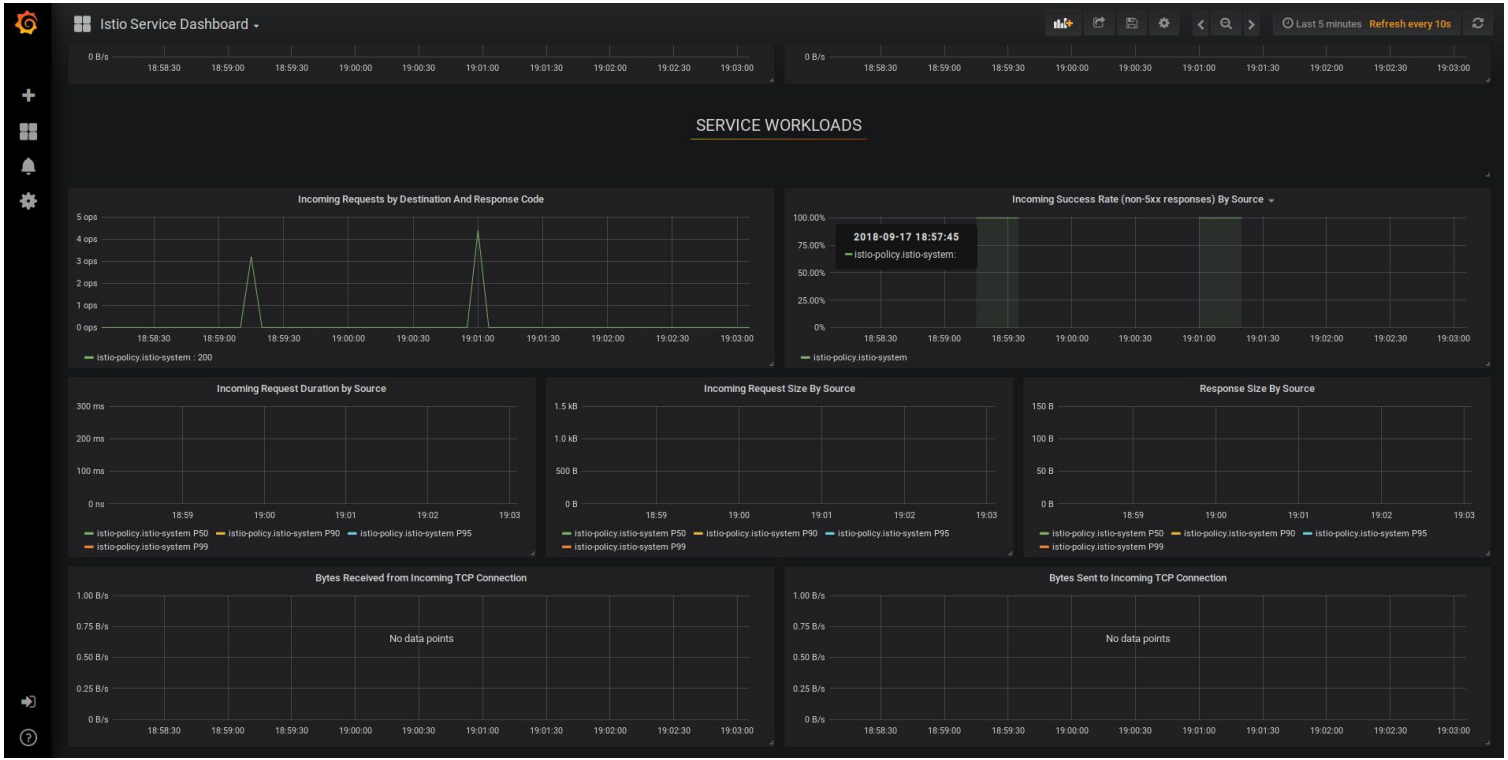
¿Qué es Network Service Mesh?

Prometheus: sistema básico de monitorización.



¿Qué es Network Service Mesh?

Graphana: otra herramienta de monitorización, que se apoya en Prometheus.



¿Qué es Network Service Mesh?

- ★ Una vez entendido qué es un Service Mesh, se puede definir qué es un Network Service Mesh: **es una extensión del concepto del Service Mesh aplicado a cargas útiles (payloads) de "capas 2/3"**, de forma que abarca casos más específicos que son complicados de estructurar en el modelo actual del networking de Kubernetes.
- ★ Network Service Mesh añade las siguientes propiedades al networking de Kubernetes:
  - **Configuraciones heterogéneas de la red.**
  - **Soporte para protocolos raros/exóticos.**
  - **Conexiones dinámicas, negociadas, "En-Demanda".**
  - **Tunneling como "first-class citizen".**
  - **Networking context como "first-class citizen".**
  - **"Policy-driven Service Funtion Chaining" (SFC).**

¿Qué es Network Service Mesh?

- ★ *Fuentes:*
  - https://github.com/ligato/networkservicemesh/blob/master/docs/What-is-nsm.md
  - http://codeblog.dotsandbrackets.com/service-mesh/
  - https://docs.google.com/presentation/d/1vmN5EevNccel6Wt8KgmkXhAfnjlli4lbjskezQjyfUE/edit#slide=id.p
  - https://drive.google.com/drive/folders/1f5fek-PLvoycMTCp6c-Dn_d9_sBNTfag
  - https://www.networkservicemesh.io/
  - https://www.nginx.com/blog/what-is-a-service-mesh/
  - https://www.youtube.com/watch?v=dJyPbBRoEE0

ZEVENET
GROW WITHOUT LIMITS

Zevenet Containers Project