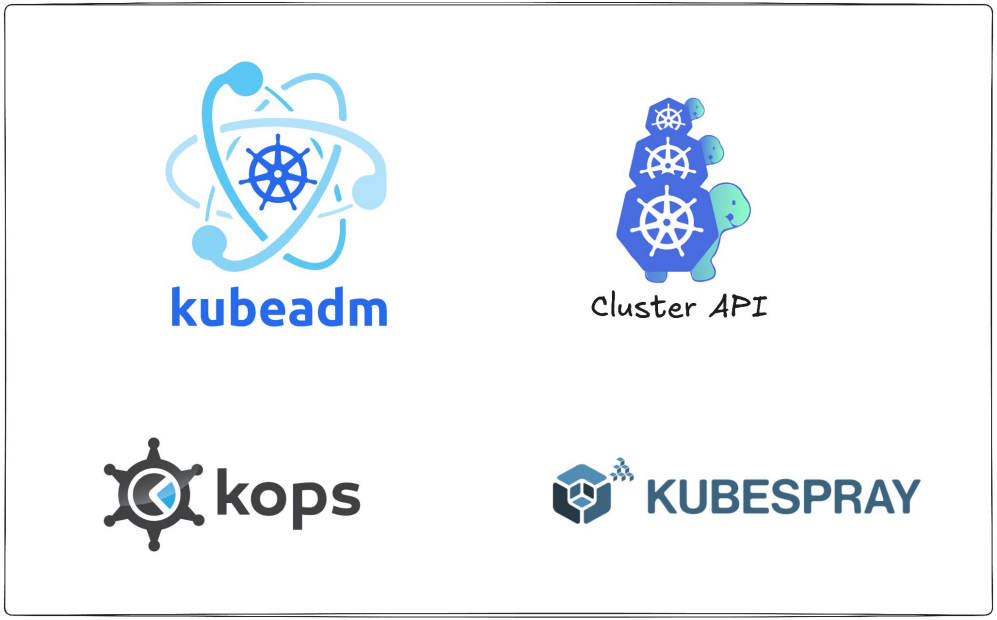
Kubernetes: primeros pasos
Servicios

KEEPCODING
Tech School

Anteriormente…

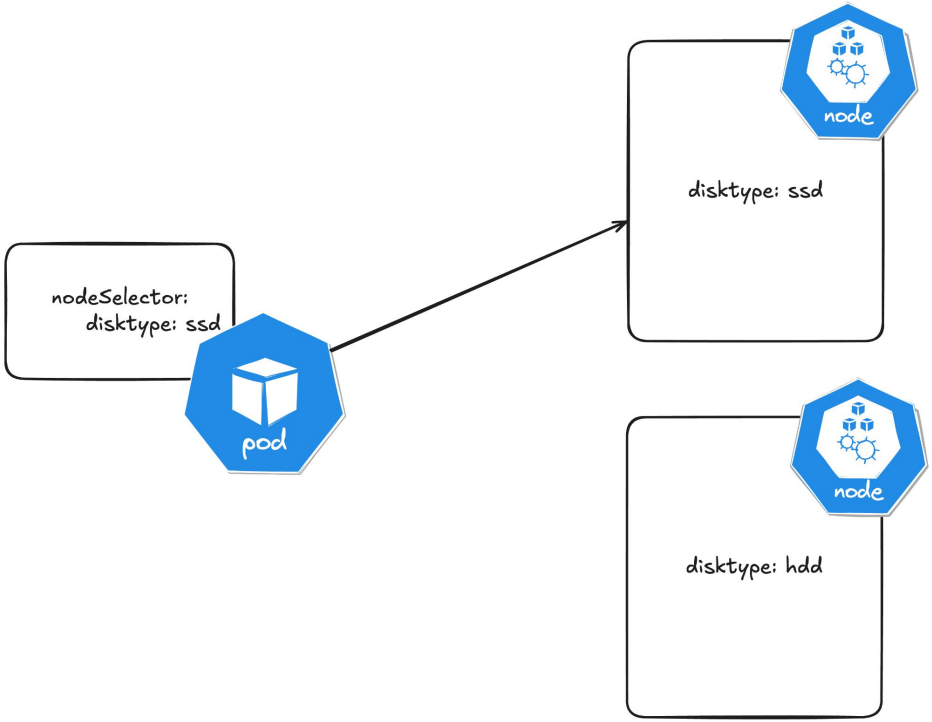


Anteriormente…

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod2
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c']
    args:
      - |
        echo 'Hello Kubernetes!'
        sleep 30
```

```
$ kubectl apply -f pod.yaml
$ kubectl get pods
$ kubectl describe pod myapp-pod
$ kubectl logs myapp-pod
```

Anteriormente…



Anteriormente…

Liveness para reiniciar un pod cuando no responde

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
```

Readiness para retirar de la red un pod cuando no responde

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: readiness
  name: readiness-http
spec:
  containers:
  - name: readiness
    image: k8s.gcr.io/liveness
    args:
    - /server
    readinessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
```

Primeros Pasos en Kubernetes

- En esta sección:
  - Servicios

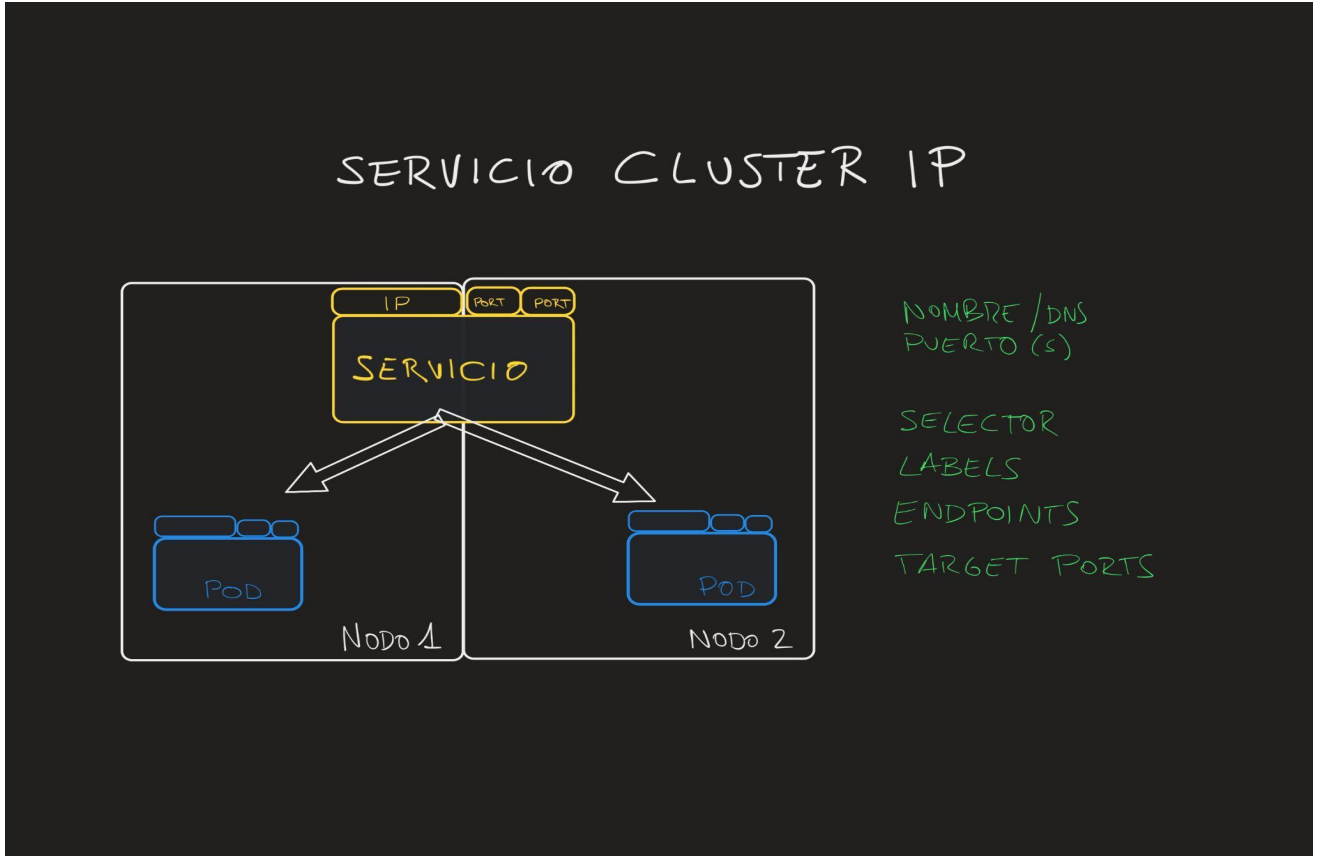
Servicios

KEEPCODING
Tech School

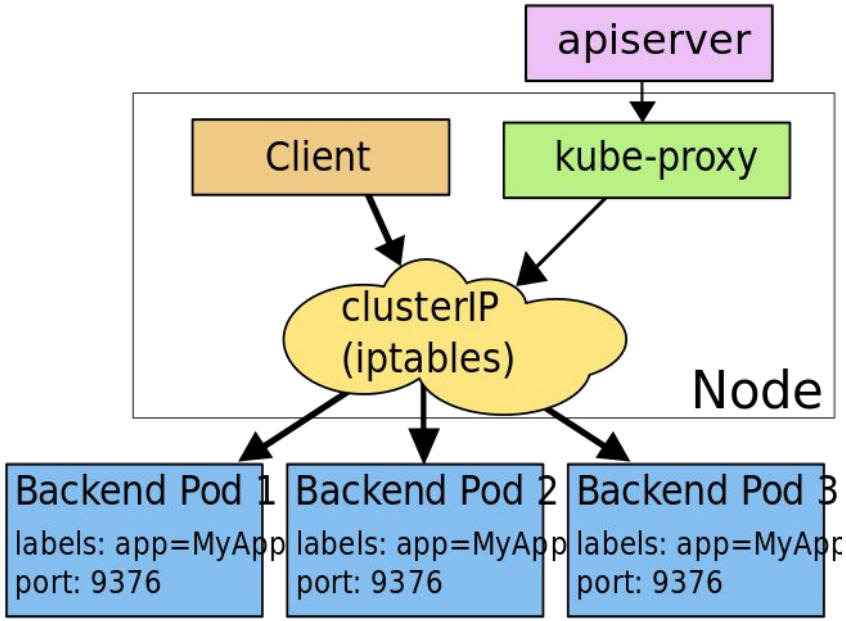
Servicios

- 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).
- Flujo de tráfico: Cliente —> Servicio (name / ip / puerto) → **Endpoints** (IPs de los pods de destino).
- Los pods de destino (target endpoints) de un servicio se obtienen a través del "**label-selector**", y dependen del estado del propio pod (si no está healthy el servicio no enviará tráfico).
- Son fáciles de entender con "kubectl describe service xxx"
- Manejados por **kube-proxy** (componente de Kubernetes)

Servicio básico (ClusterIP)



Servicio básico (ClusterIP)



Servicios

- Tipos de Servicios:
  - **ClusterIP**: Expone la aplicación en una IP interna virtual (VIP) al resto del cluster. Proporciona **acceso interno.**
  - **NodePort**: Expone la aplicación al exterior e interior (extensión de Cluster IP) a través de un puerto en los nodos de Kubernetes. Proporciona **acceso externo**.
  - **LoadBalancer**: Expone la aplicación al exterior e interior y además crea un LoadBalancer externo (extensión de NodePort). Proporciona **acceso externo**.
- Doc oficial →aquí

Servicios

- Tipos de servicios especiales:
  - **Headless Service**: Es un servicio ClusterIP pero sin dirección IP asociada. No lo gestiona Kube-proxy sino que directamente la resolución DNS devolverá IPs de los endpoints.
  - ExternalName

Service Spec

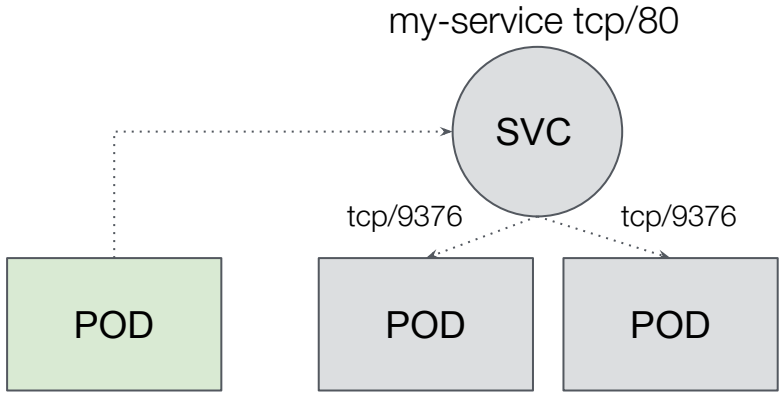
- `clusterIP` → Sólo lo usaremos para 'HeadLess' services (con valor 'None').
- `externalName`→ Sólo lo usaremos en servicios del tipo ExternalName.
- `externalTrafficPolicy`→ Cluster vs Local, implicaciones importantes en cuanto a routing y NAT. Valor por defecto y recomendado: 'Cluster'.
- **`ports`**→ Definimos los puertos donde escuchará el servicio y los de destino de los pods.
    - `name`→ Nombre descriptivo del puerto
    - `nodePort` → (sólo cuando haya que abrir un puerto a nivel de host)
    - `port` → Puerto en el que escucha el servicio
    - `targetPort` →Puerto en el que escuchan los pods (permite nombre)
- **`selector`** → Filtrará labels de los pods, tiene una sintaxis especial.
- `sessionAffinity` → Tipo de afinidad para balanceo (acepta ClientIP y None)
- **`type:`**
    - `ClusterIP`
    - `NodePort`
    - `LoadBalancer`
    - `ExternalName`
    - `"Headless" (clusterIP: none)`

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.31/#servicespec-v1-core

Servicios

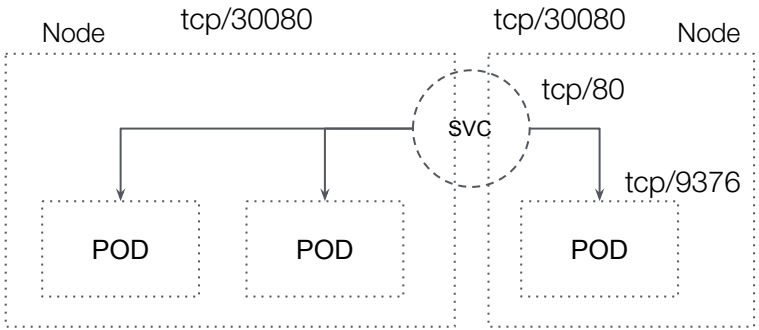
- Referencias:
  - https://kubernetes.io/docs/concepts/services-networking/service/
  - https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```



Service type NodePort

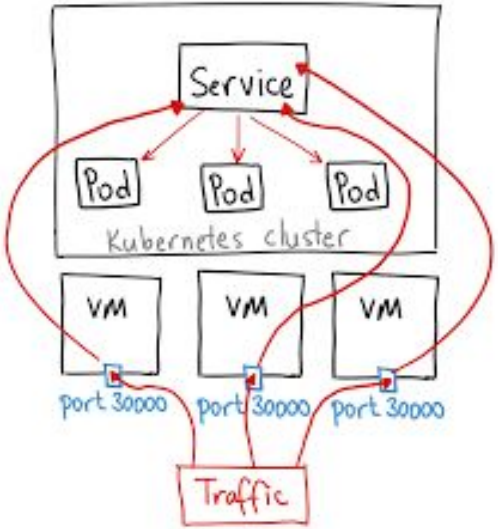
- Expone el servicio en un puerto TCP de todos los nodos.
- Mismo puerto para todos los nodos, usando la IP real de cada nodo.
- Útil si no disponemos de LoadBalancer o si lo queremos gestionar nosotros.
- Por defecto el rango posible está entre 30000 y 32767
- **Tedioso de integrar**, ya que por defecto los puertos son aleatorios y además **no se pueden reutilizar!**
- **Implicaciones de seguridad** (hay que abrir el puerto al exterior)



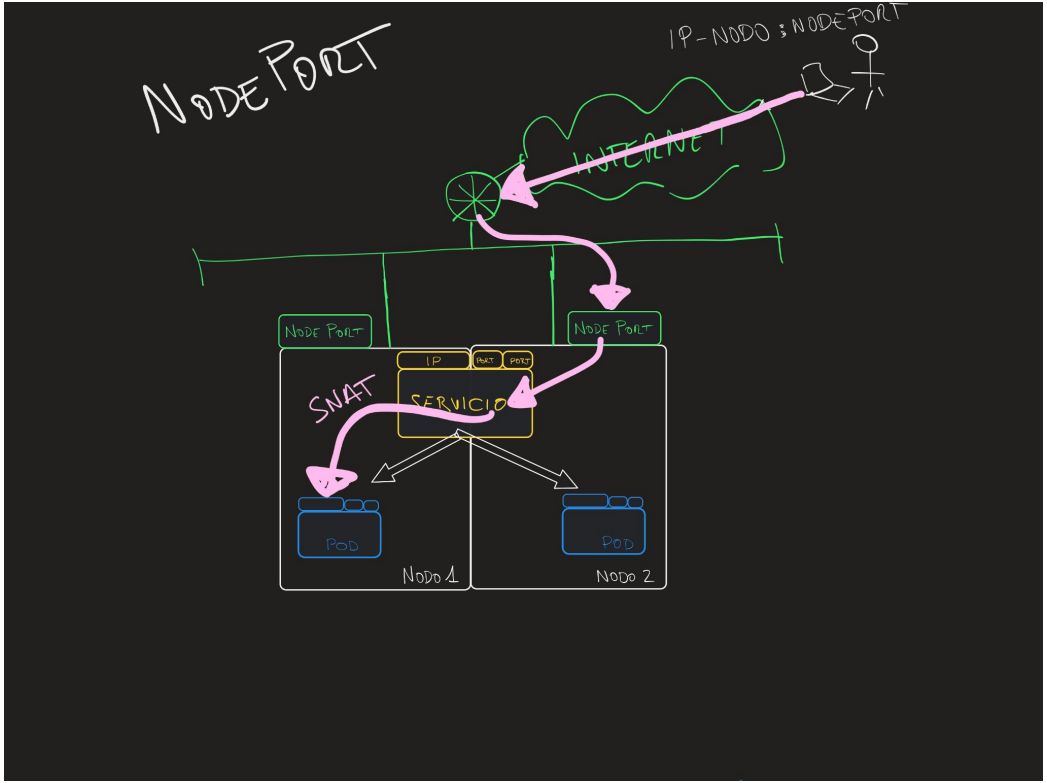

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
      nodePort: 30080
  type: NodePort
```

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

Servicio NodePort

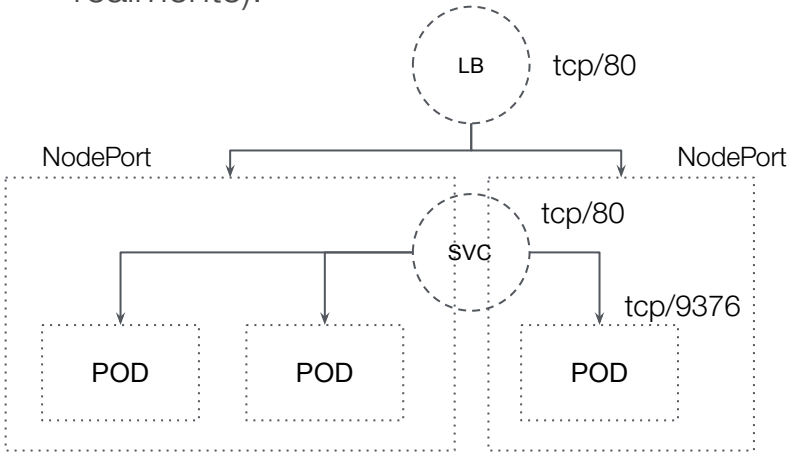


Servicio NodePort



Service type LoadBalancer

- Igual que NodePort pero además crea y configura un LoadBalancer externo en la cloud.
- Requiere integración con entidad externa.
- Generalmente el backend del Load Balancer apuntará a todos los nodos, haya o no un POD en ese nodo (apunta al NodePort realmente).



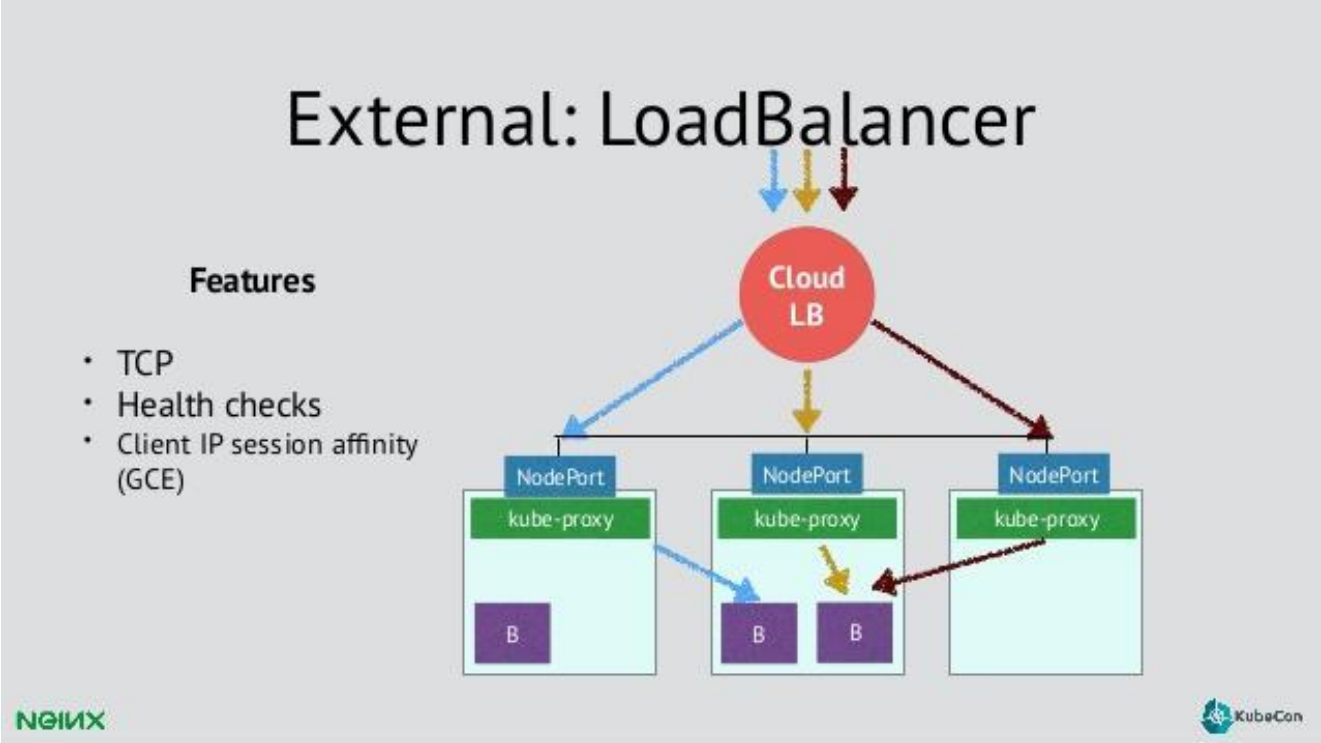

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  type: LoadBalancer
```

Mediante **annotations** podemos exponer el service en un balanceador interno (GKE)

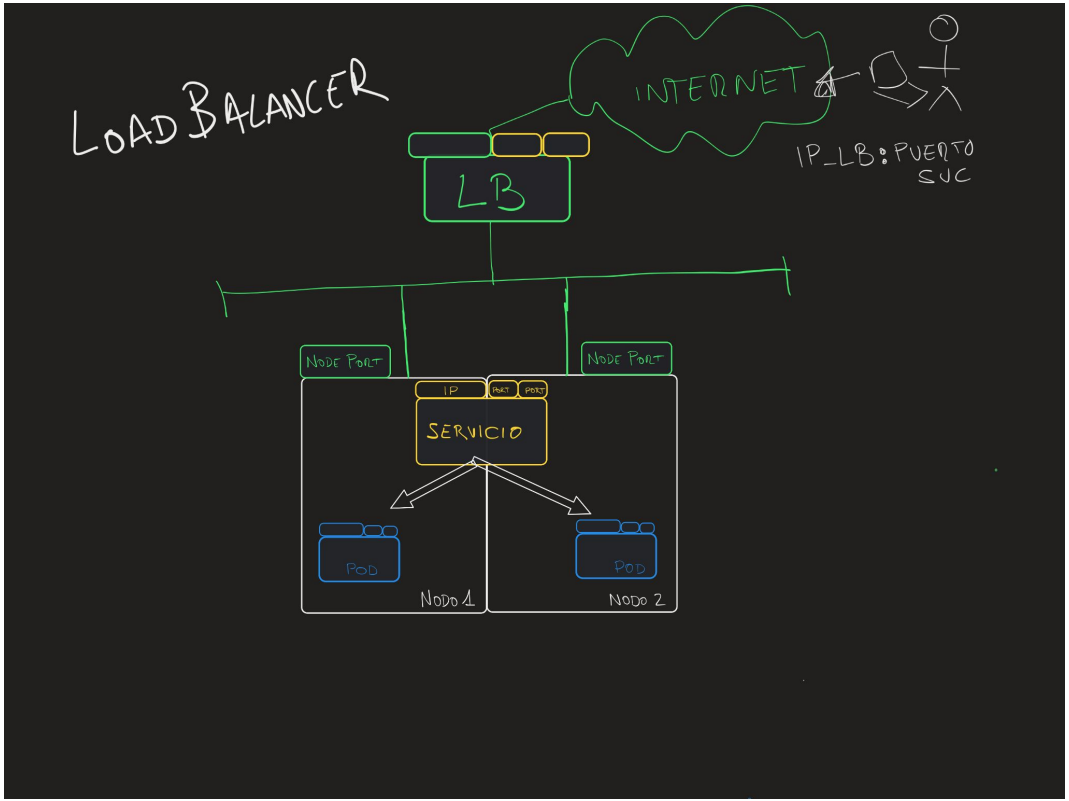
```yaml
metadata:
    name: my-service
    annotations:
      cloud.google.com/load-balancer-type: "Internal"
```

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

Servicio LoadBalancer

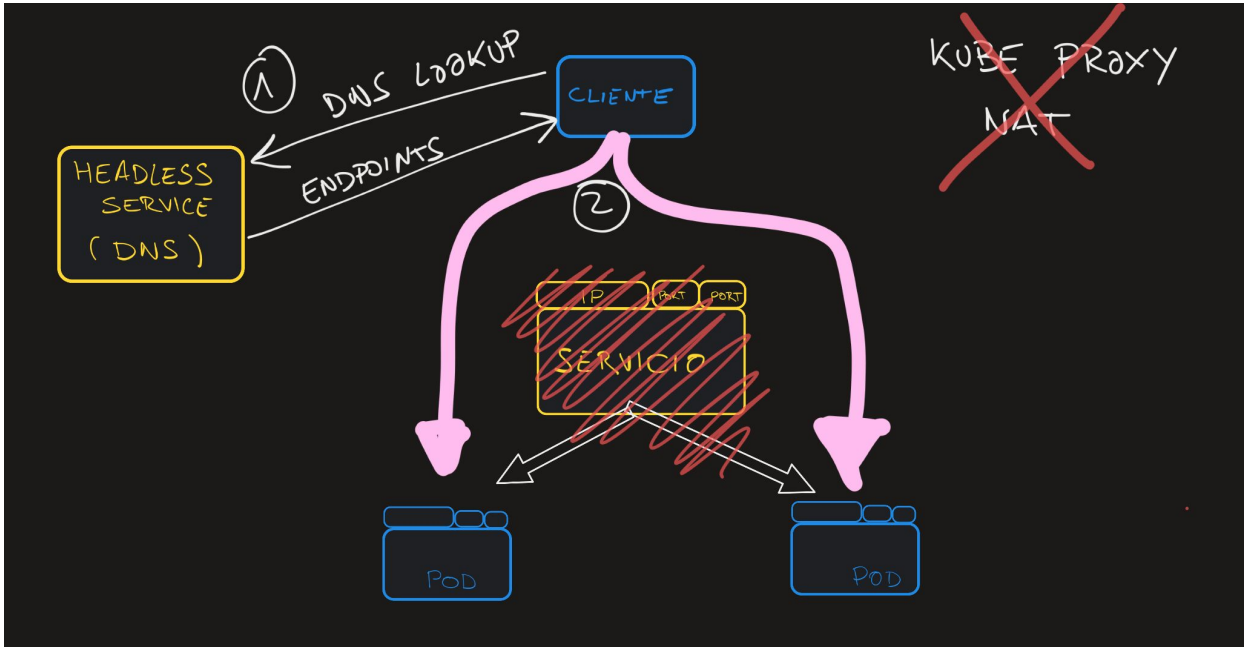


Servicio LoadBalancer



Servicios Especiales

- Headless Service (no hay enrutado inteligente y NAT, sino que se basa en DNS).



Services

- DEMO!

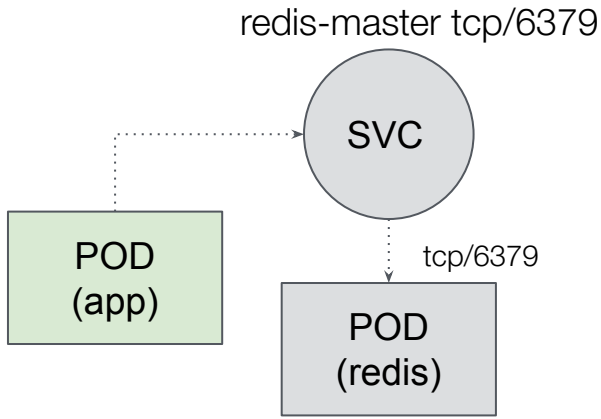
Relación ClusterIP / NodePort / LoadBalancer

- Es muy importante entender las diferencias y cómo están relacionados.
- Un servicio de tipo LoadBalancer incluye internamente un NodePort que a su vez incluye un ClusterIP.
- Un servicio del tipo NodePort incluye internamente un ClusterIP
- Un servicio del tipo ClusterIP incluye simplemente la IP interna (VIP) asignada al servicio y kube-proxy es capaz de enrutar el tráfico.
- Un servicio HeadLess es un ClusterIP sin IP (redirección basada en DNS).

- A partir de 1.20 existe una feature alpha para permitir a los LoadBalancers enviar tráfico directamente a los pods, eliminando la necesidad del NodePort. Más aquí

Descubrimiento de servicios

- Soporta 2 modos: Variables de entorno y **DNS** (recomendado!!)
- Si tenemos un servicio que se llame “redis-master”, desde un POD cualquiera dentro del mismo namespace podremos acceder al servicio "redis-master” usando las variables de entorno inyectadas o el nombre DNS



{SERVICIO}.{NAMESPACE}.svc.cluster.local

```
$ kubectl exec -it app -- sh
/ # nslookup redis-master
/ # env
```

```
REDIS_MASTER_SERVICE_HOST=10.0.0.11
REDIS_MASTER_SERVICE_PORT=6379
REDIS_MASTER_PORT=tcp://10.0.0.11:6379
REDIS_MASTER_PORT_6379_TCP=tcp://10.0.0.11:6379
REDIS_MASTER_PORT_6379_TCP_PROTO=tcp
REDIS_MASTER_PORT_6379_TCP_PORT=6379
REDIS_MASTER_PORT_6379_TCP_ADDR=10.0.0.11
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app
  labels:
    app: app
spec:
  containers:
  - name: app
    image: busybox:1.28
    command: ['sh', '-c']
    args:
      - |
        sleep 3600
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis
  labels:
    app: redis
spec:
  containers:
  - name: redis
    image: redis
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-master
spec:
  selector:
    app: redis
  ports:
  - protocol: TCP
    port: 6379
    targetPort: 6379
```

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá