

# Certified Kubernetes Administrator



# Temario



¿QUE SON LOS  
KUBERNETES?



DESPLIEGUE DE APLICACIONES  
SIMPLES Y MULTI-CONTAINERS



KUBERNETES  
SERVICES



ADMINISTRACIÓN DE  
APLICACIONES  
DEPLOYMENTS Y ROLLOUTS



ASEGURAMIENTO DE CONDICIONES  
PARA CONTENEDORES



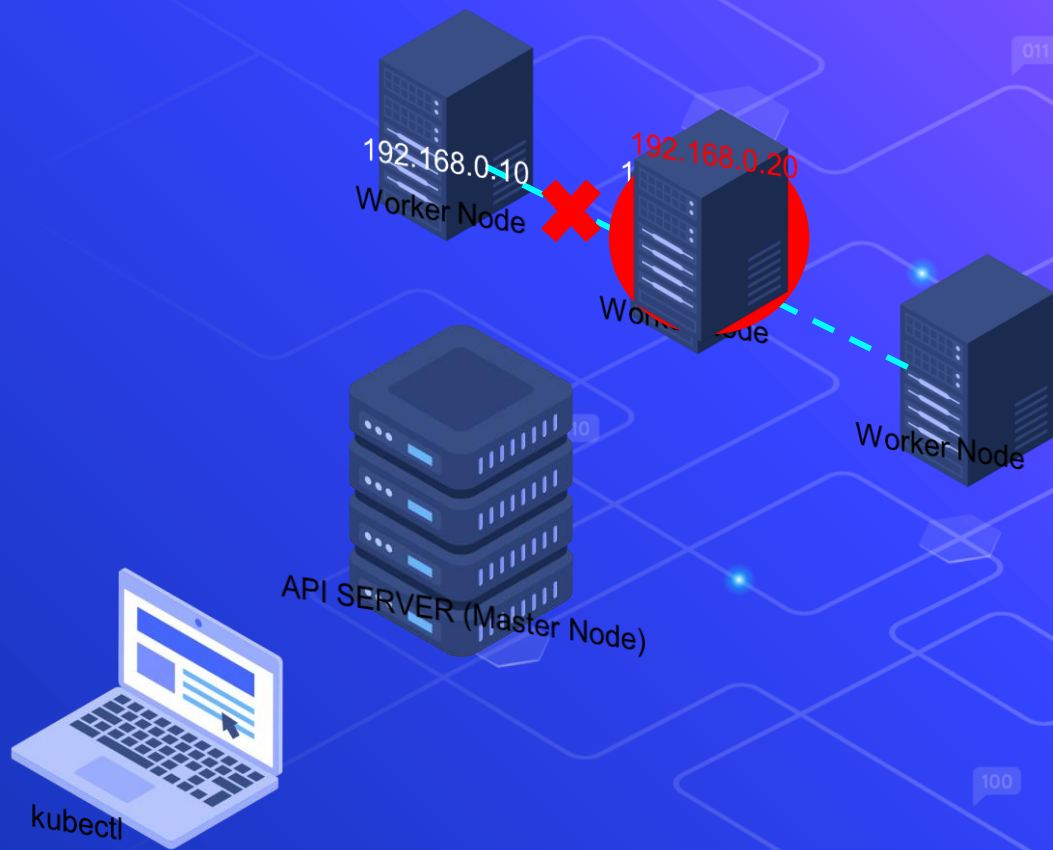
ADMINISTRACIÓN Y  
CONFIGURACIÓN DE  
MAPS, SECRETS Y  
PERSISTENCIA DE  
DATOS



# Comencemos!



“



# 1. Services



# Services

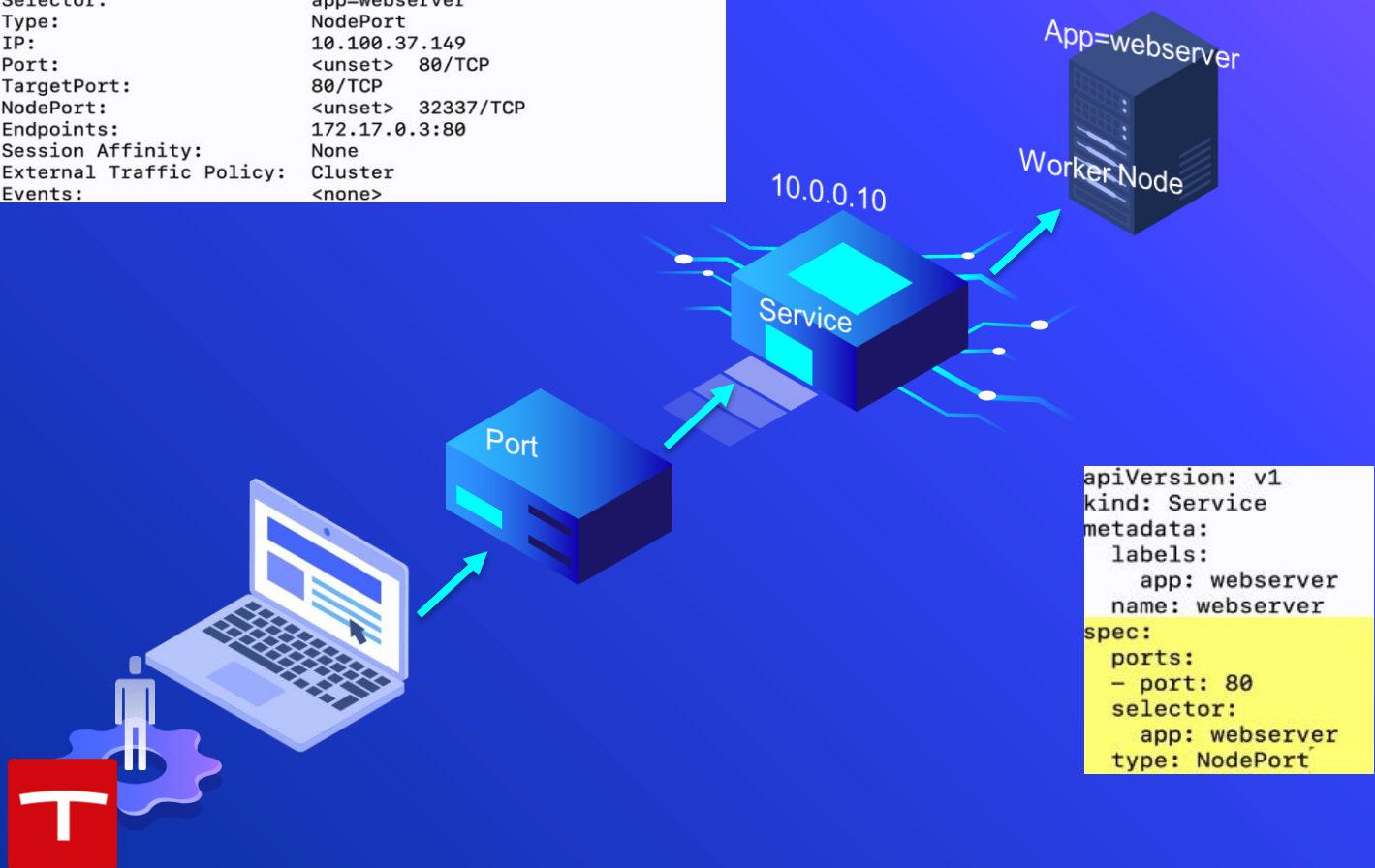
- ⬡ Los servicios definen las reglas de network mediante las cuales se accederá a los pods del clúster y a internet.
- ⬡ Usan Labels para identificar los grupos de pods.
- ⬡ Los servicios tienen una IP fija.
- ⬡ Distribuye las solicitudes a través de los pods en los grupos.



# Big concept



```
admin@Jamess-MBP src % kubectl describe service webserver
Name:         webserver
Namespace:    default
Labels:       app=webserver
Annotations:  <none>
Selector:     app=webserver
Type:         NodePort
IP:           10.100.37.149
Port:         <unset> 80/TCP
TargetPort:   80/TCP
NodePort:     <unset> 32337/TCP
Endpoints:    172.17.0.3:80
Session Affinity: None
External Traffic Policy: Cluster
Events:       <none>
```



```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: webserver
    name: webserver
spec:
  ports:
    - port: 80
  selector:
    app: webserver
  type: NodePort
```

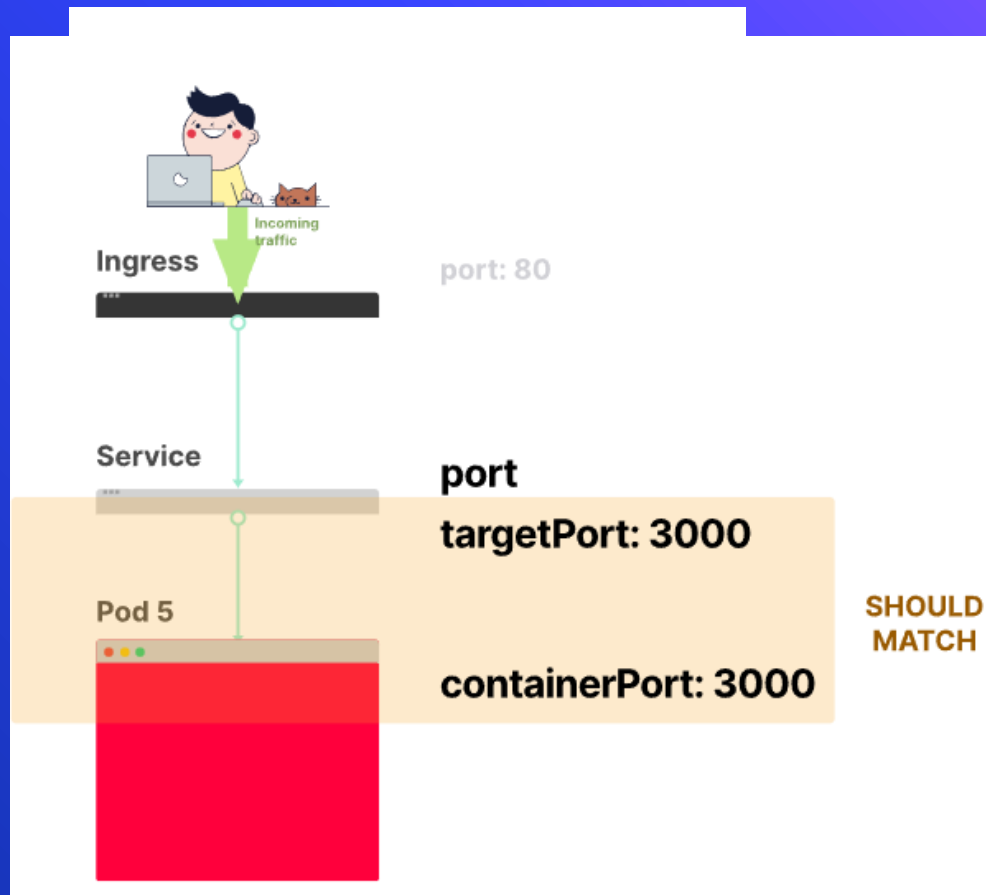




```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  labels:
    track: canary
spec:
  selector:
    matchLabels:
      any-name: my-app
  template:
    metadata:
      labels:
        any-name: my-app
    spec:
      containers:
        - name: cont1
          image: learnk8s/app:1.0.0
          ports:
            - containerPort: 8080

```



```

apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - port: 80
      targetPort: 8080
  selector:
    any-name: my-app

```



apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: my-ingress

spec:

rules:

- http:

paths:

- backend:

service:

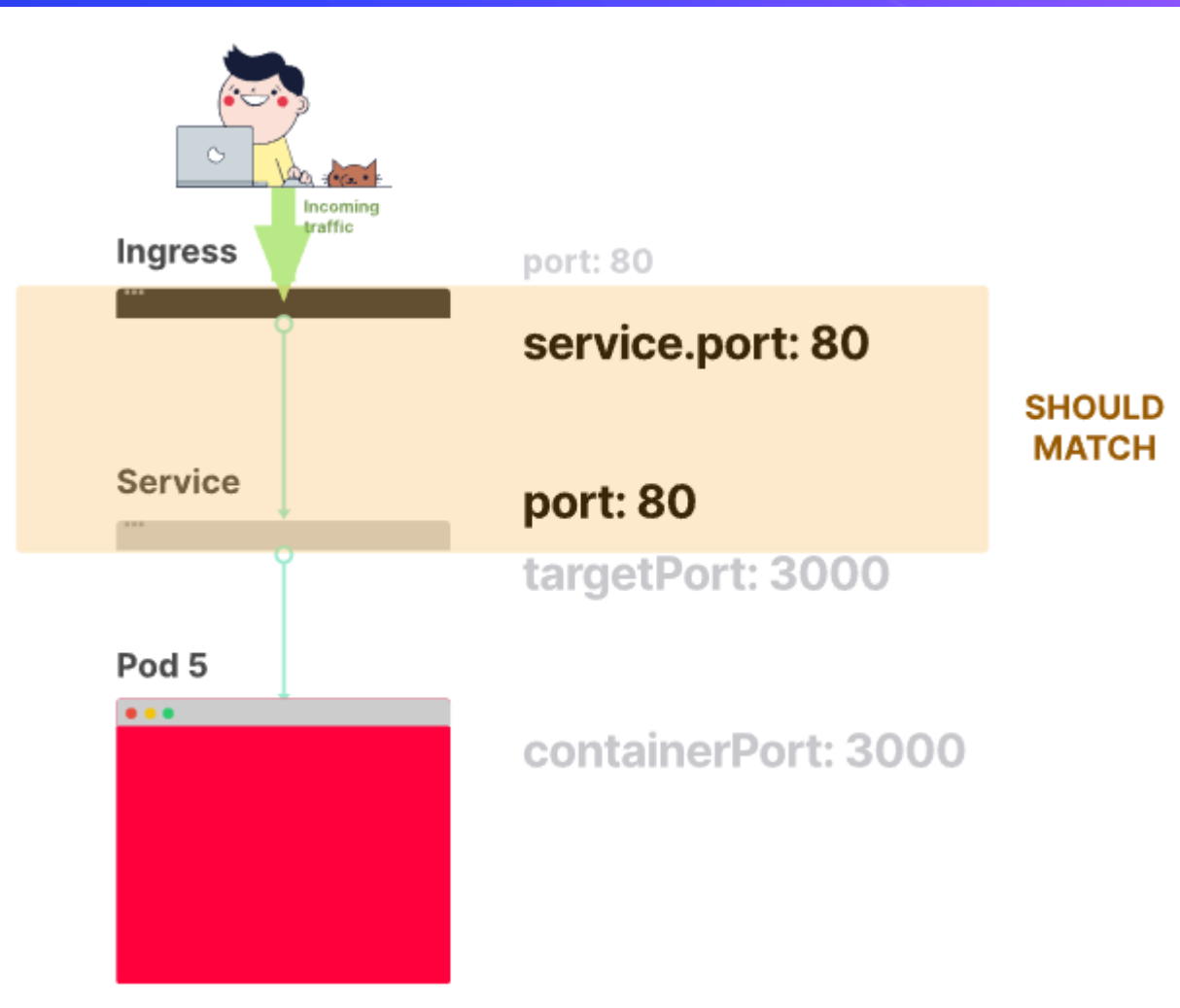
name: my-service

port:

number: 80

path: /

pathType: Prefix



# Tipos de Servicios

⬡ Cluster IP

⬡ NodePort

⬡ LoadBalancer

⬡ ExternalName

## Otros conceptos (No servicios)

⬡ Namespace

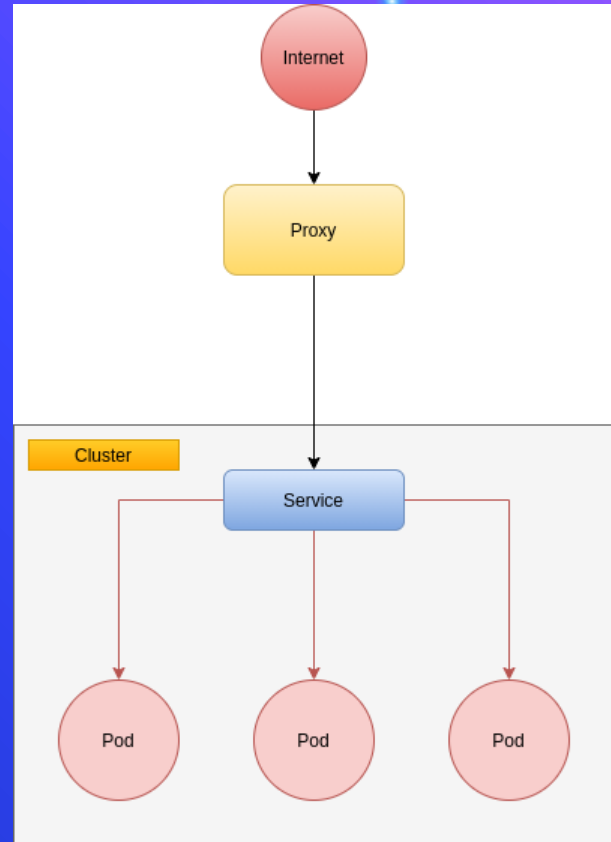
⬡ Ingress  
Controller

⬡ Headless



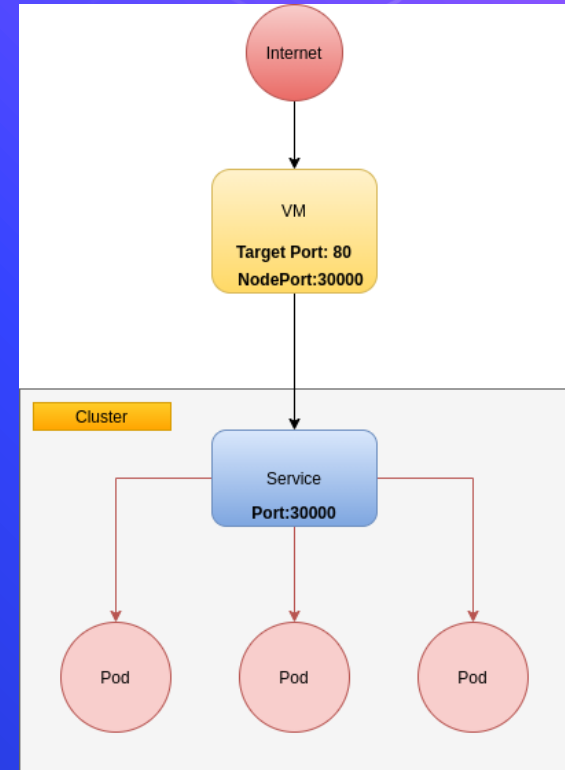
# Cluster IP

ClusterIP es el servicio que establece por defecto Kubernetes. Este tipo de servicio únicamente va a ser accesible dentro de tu clúster por otros servicios o aplicaciones, es decir, no va a tener acceso externo.



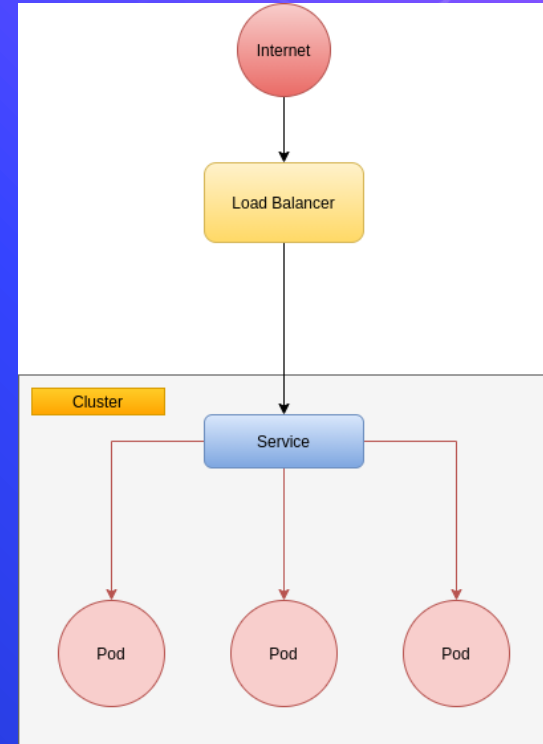
# NodePort

NodePort es una de las primeras formas que se tienen de dirigir tráfico externo de internet a tu servicio. Lo que hace NodePort, es abrir un puerto en los nodos, y todo el tráfico que llegue a ese port se dirige hacia el servicio definido.



# LoadBalance

Un LoadBalancer en Kubernetes expone externamente el servicio usando por lo general un Load Balancer del proveedor del cloud en el que te encuentres. La creación de un load balancer sucede de manera asíncrona y la información sobre el balanceador se publica en el campo *status.loadBalancer*.



# ExternalName

Los servicios externalName mapean un servicio DNS, no suelen ser un selector típico dado que el servicio DNS devuelve un registro CNAME con una URL.

apiVersion: v1

kind: Service

metadata:

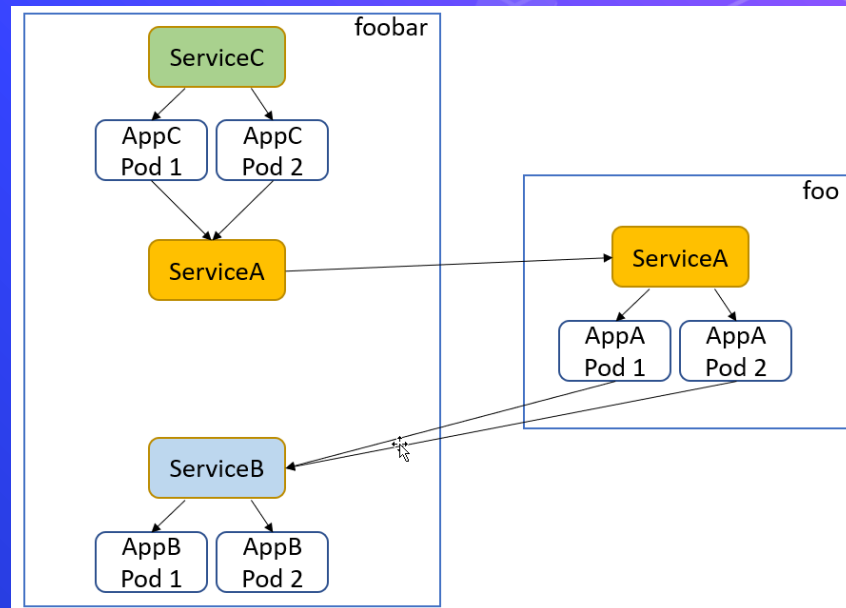
name: mi-servicio

namespace: prod

spec:

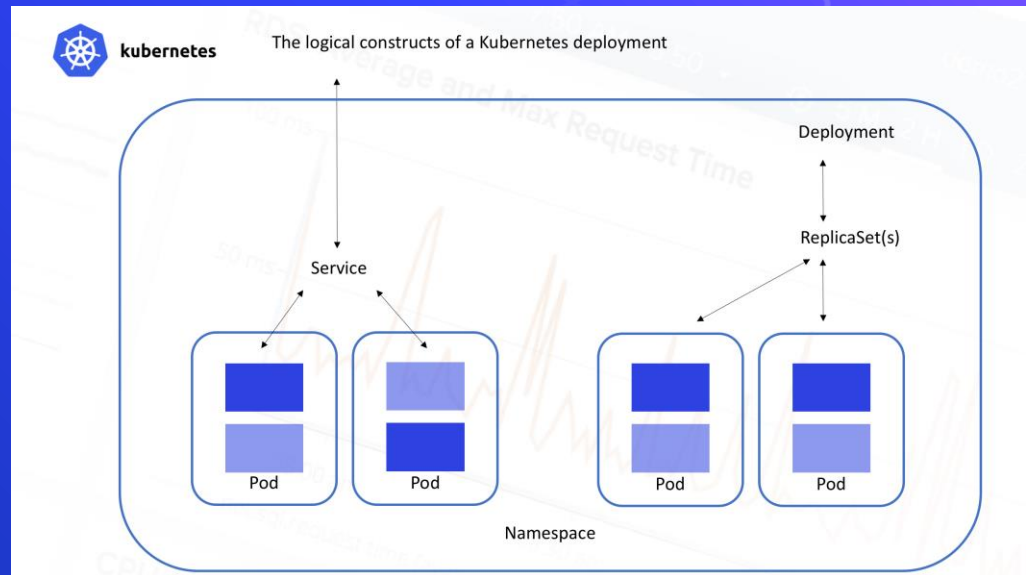
type: ExternalName

externalName: my.database.example.com



# Namespace

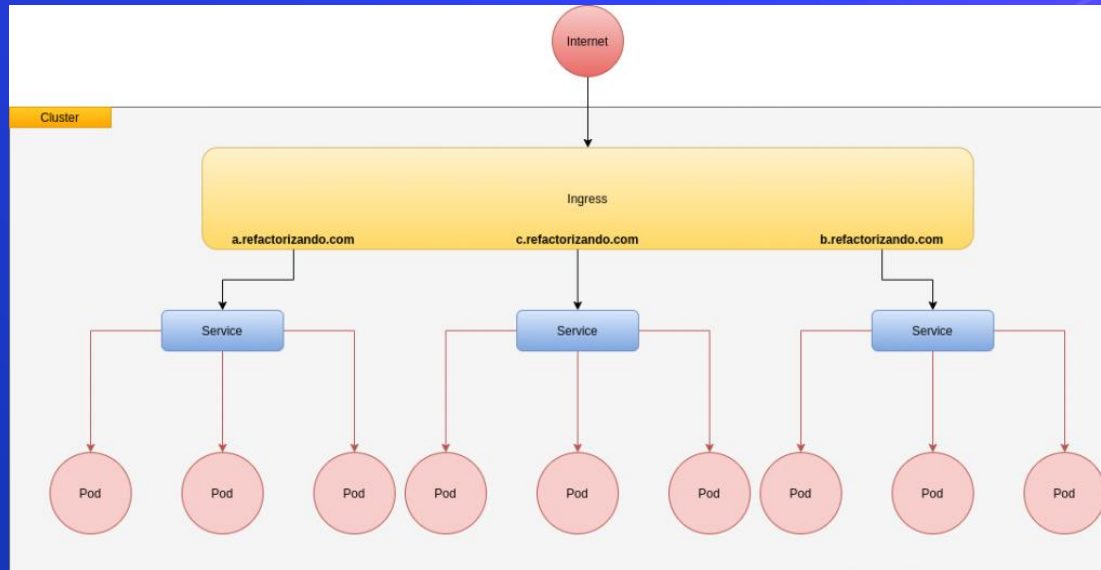
Los espacios de nombres están pensados para utilizarse en entornos con muchos usuarios distribuidos entre múltiples equipos, o proyectos. Para aquellos clústeres con unas pocas decenas de usuarios, no deberías necesitar crear o pensar en espacios de nombres en absoluto. Empieza a usarlos solamente si necesitas las características que proporcionan.







# Ingress

A diferencia de los tipos de servicio comentados anteriormente, un **Ingress** no es realmente un tipo de servicio, sino que es una «pieza» que se encuentra en frente de muchos servicios y actúa como un punto de entrada a nuestro clúster.



# Ingress

## Controller

-  Ingress es la herramienta más utilizada cuando se quiere exponer múltiples servicios a través de la misma IP, además los servicios harán uso del **protocolo L7**, el cual opera en el más alto nivel de la capa de aplicación, la cual trata con los mensajes.
-  También el uso de un Ingress, obviamente reduce el coste en un proveedor de cloud, ya que pagarás por un Load Balancer únicamente, además podrás añadir diferentes funcionalidades como SSL, Auth, etc...



# Headless

## Services

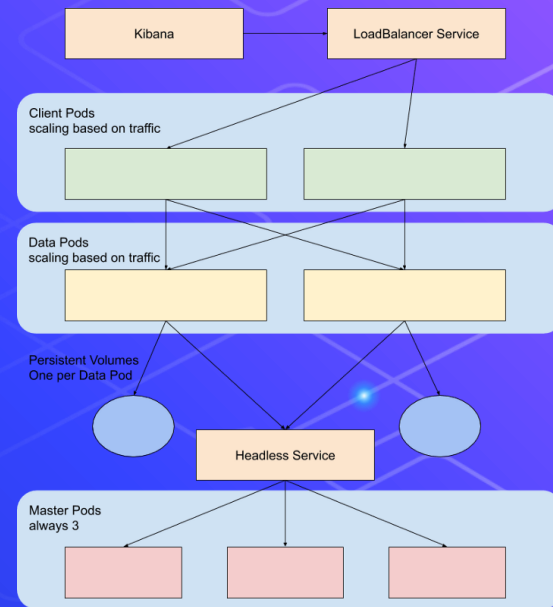
Algunas veces no necesitas balancear cargas y una IP única. En este caso, puedes crear lo que llamamos Services "headless", especificando "None" para el Cluster IP.

Puedes usar un Service headless para hacer una interfaz con otros mecanismos de descubrimiento de servicios, sin estar atado a la implementación de Kubernetes.

Para los services headless, no se asigna una clúster IP, kube-proxy no maneja estos Services, y no hay balanceo de cargas o redirección por la plataforma para ellos. Cómo se configura el DNS automáticamente depende de si el Service tiene selectores definido:

Con Selector

Sin Selector



# Recapitulamos?







```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: microservice-one
  ...
```

```
spec:
```

```
  replicas: 2
```

```
  ...
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: microservice-one
```

```
    spec:
```

```
      containers:
```

```
        - name: ms-one
```

```
          imagen: myprivaterepo
```

```
          ports:
```

```
            - containerPort: 3000
```

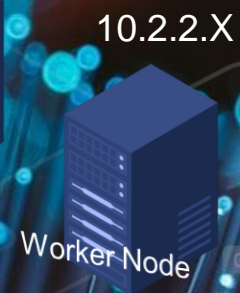
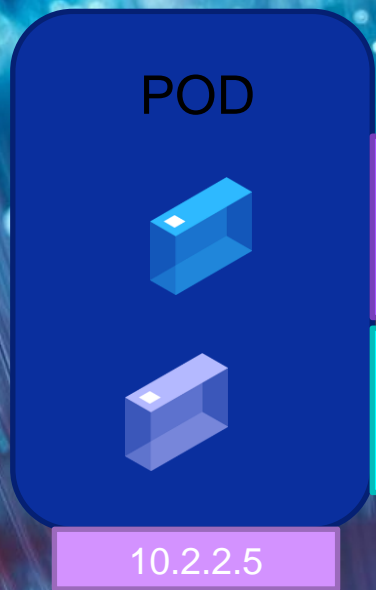
```
            - name: log-collector
```

```
              image: myprivaterepo
```

```
              ports:
```

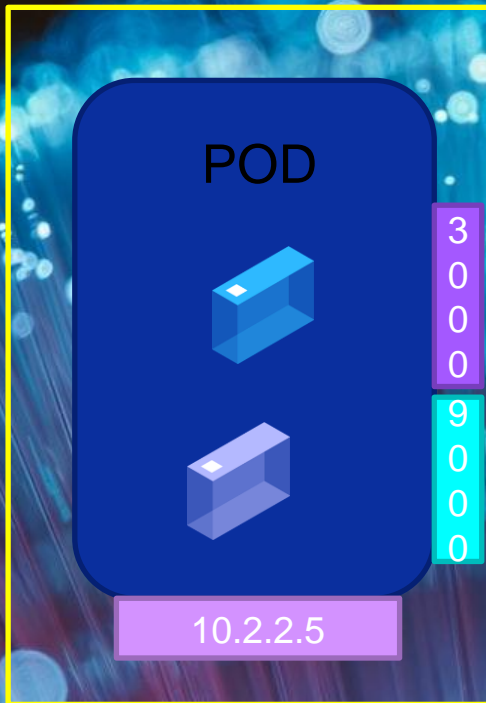
```
                - containerPort: 9000
```







NODO2



NODO1



NODO2

NODO1

POD

POD

3  
0  
0  
0  
0

9  
0  
0  
0  
0

3  
0  
0  
0  
0

9  
0  
0  
0  
0

10.2.2.5

10.2.1.4

SERVICE

INGRESS





```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ms-one-ingress
  annotations:
    kubernetes.io/ingress.class:
      "nginx"
spec:
  rules:
    - host: microservice-one.com
      http:
        paths:
          - path: /
            backend:
              serviceName: microservice-one-service
              servicePort: 3200
```



apiVersion: V1  
kind: Service  
metadata:

name: microservice-one-service

spec:

selector:

app: microservice-one

ports:

-protocol: TCP

port: 3200

targetPort: 3000

D

3  
0  
0  
0

9  
0  
0  
0

1.4

3200

SERVICE

10.128.8.64

INGRESS

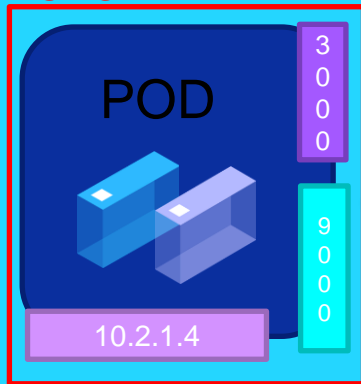
001

011

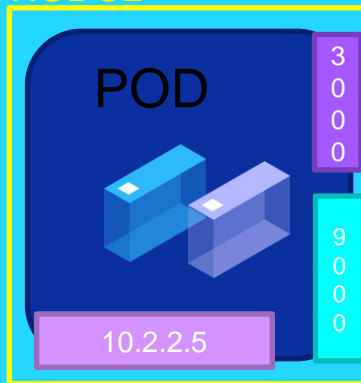
010



NODO1



NODO2



3200

SERVICE

10.128.8.64

INGRESS

apiVersion: V1  
kind: Service  
metadata:

name: microservice-one-

service

spec:

selector:

app: microservice-one

ports:

-protocol: TCP

port: 3200

targetPort: 3000

nodePort: 30008

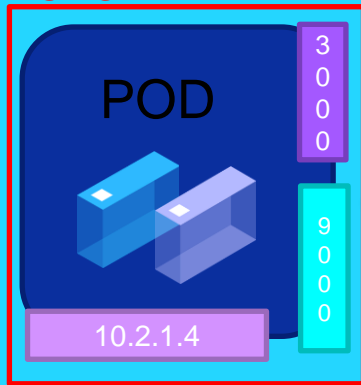
3  
0  
0  
0  
0  
8

NODEPORT



Esto permite establecer una conexión directa mediante el nodeport, sin embargo esta no es segura.

NODO1



NODO2



3200

SERVICE

10.128.8.64

INGRESS

```
apiVersion: V1
kind: LoadBalancer
metadata:
  name: microservice-one-
service
spec:
  selector:
    app: microservice-one
  ports:
    -protocol: TCP
      port: 3200
      targetPort: 3000
      nodePort: 30008
```

3  
0  
0  
0  
0  
8

NODEPORT



# Resumen

Los servicios de Loadbalancer son una extensión del servicio de NodePort.

Los servicios de NodePort son una extensión del servicio de ClusterIP.

Nodeport NO debe usarse para establecer conexiones a externas.





# Temario



¿QUE SON LOS  
KUBERNETES?



DESPLIEGUE DE APLICACIONES  
SIMPLES Y MULTI-CONTAINERS



KUBERNETES  
SERVICES



ADMINISTRACIÓN DE  
APLICACIONES  
DEPLOYMENTS Y ROLLOUTS



ASEGURAMIENTO DE CONDICIONES  
PARA CONTENEDORES



ADMINISTRACIÓN Y  
CONFIGURACIÓN DE  
MAPS, SECRETS Y  
PERSISTENCIA DE  
DATOS



## 2. ADMINISTRACIÓN DE APLICACIONES DEPLOYMENTS Y ROLLOUTS



# Deployments





# ¿Qué son?



Representan múltiples instancias de un pod.

Describen el estado DESEADO que kubernetes requieren lograr.

El controlador maestro de los deployments converge entre el estado actual y el estado deseado.



# Casos de uso:

- ❖ Crear un Deployment para desplegar un ReplicaSet. El ReplicaSet crea los Pods en segundo plano. Comprueba el estado del despliegue para comprobar si es satisfactorio o no.
- ❖ Declarar el nuevo estado de los Pods actualizando el PodTemplateSpec del Deployment. Ello crea un nuevo ReplicaSet y el Deployment gestiona el cambio de los Pods del viejo ReplicaSet al nuevo de forma controlada. Cada nuevo ReplicaSet actualiza la revisión del Deployment.
- ❖ Retroceder a una revisión anterior del Deployment si el estado actual de un Deployment no es estable. Cada retroceso actualiza la revisión del Deployment.
- ❖ Escalar horizontalmente el Deployment para soportar más carga.
- ❖ Pausar el Deployment para aplicar múltiples arreglos a su PodTemplateSpec y, a continuación, reanúdalo para que comience un nuevo despliegue.
- ❖ Usar el estado del Deployment como un indicador de que el despliegue se ha atascado.
- ❖ Limpiar los viejos ReplicaSets que no necesites más.



```

apiVersion: apps/v1 # apps API group
kind: Deployment
metadata:
  name: data-tier
  labels:
    app: microservices
    tier: data
spec:
  replicas: 1
  selector:
    matchLabels:
      tier: data
  template:
    metadata:
      labels:
        app: microservices
        tier: data
    spec: # Pod spec
      containers:
        - name: redis
          image: redis:latest
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 6379

```

```
kubectl get -n deployments deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
app-tier	1/1	1	1	76s
data-tier	1/1	1	1	76s
support-tier	1/1	1	1	76s

```
kubectl -n deployments get pods
```

NAME	READY	STATUS	RESTARTS	AGE
app-tier-74cd4c68c9-z99h4	1/1	Running	0	2m24s
data-tier-8646dd765b-k6h98	1/1	Running	0	2m24s
support-tier-997bc57fb-c579t	2/2	Running	0	2m24s



# ¿Como escala?

```
kubectl scale -n deployments deployments support-tier --replicas=5
```

NAME	READY	STATUS	RESTARTS	AGE
app-tier-74cd4c68c9-z99h4	1/1	Running	0	4m49s
data-tier-8646dd765b-k6h98	1/1	Running	0	4m49s
support-tier-997bc57fb-42s98	2/2	Running	0	54s
support-tier-997bc57fb-c579t	2/2	Running	0	4m49s
support-tier-997bc57fb-l9lkc	2/2	Running	0	54s
support-tier-997bc57fb-lp56v	2/2	Running	0	54s
support-tier-997bc57fb-mtfrf	2/2	Running	0	54s

```
kubectl delete -n deployments pods pod1 pod2 pod3
```

NAME	READY	STATUS	RESTARTS	AGE
app-tier-74cd4c68c9-h7hvf	1/1	Running	0	9m49s
data-tier-8646dd765b-wjmd	1/1	Running	0	9m49s
support-tier-997bc57fb-7597k	2/2	Running	0	3m20s
support-tier-997bc57fb-79gsl	2/2	Running	0	8m11s
support-tier-997bc57fb-8wjnf	2/2	Terminating	0	8m11s
support-tier-997bc57fb-b2f9f	2/2	Terminating	0	3m20s
support-tier-997bc57fb-b8c4q	2/2	Running	0	31s
support-tier-997bc57fb-fggcx	2/2	Running	0	31s
support-tier-997bc57fb-hprh6	2/2	Terminating	0	3m20s
support-tier-997bc57fb-k8kfd	2/2	Running	0	31s



# ¿Como escala?

kubectl describe -n deployments service app-tier

```
Name:                app-tier
Namespace:           deployments
Labels:              app=microservices
Annotations:         <none>
Selector:            tier=app
Type:                ClusterIP
IP:                  10.111.185.244
Port:                <unset> 8080/TCP
TargetPort:          8080/TCP
Endpoints:           172.17.0.10:8080,172.17.0.11:8080,172.17.0.12:8080 + 2
Session Affinity:    None
```

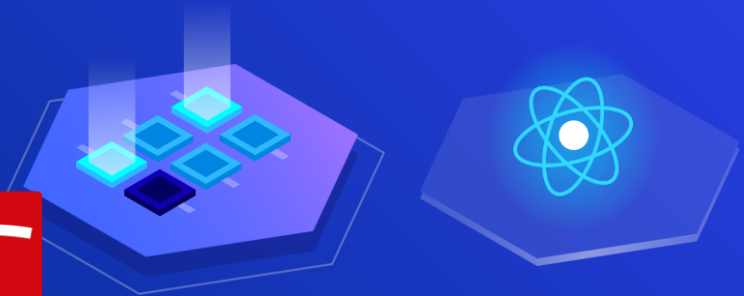


# ¿Como se define el autoescalado?

- Se define en base a métricas de uso del CPU.
- Se configura el número mínimo y máximo de replicas.
- El porcentaje de uso del CPU se expresa como porcentaje solicitado por el pod.

```
spec:  
  replicas: 5  
  selector:  
    matchLabels:  
      tier: app  
  template:  
    metadata:  
      labels:  
        app: microservices  
        tier: app  
    spec:  
      containers:  
        - name: server  
          image: lrakai/microservices:server-v1  
          ports:  
            - containerPort: 8080  
      resources:  
        requests:  
          cpu: 20m # 20 milliCPU / 0.02 CPU  
      env:
```

<https://github.com/kubernetes-sigs/metrics-server>



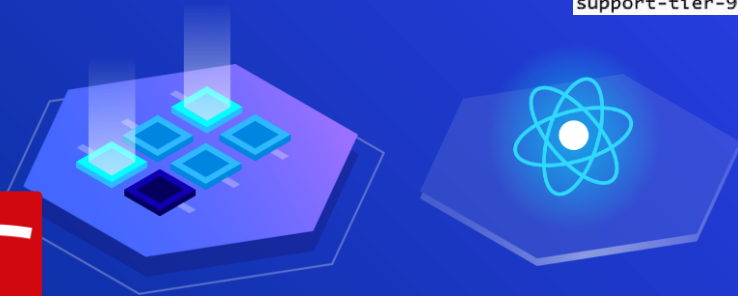
# ¿Como se define el autoescalado?

- `Kubectrl get -n deployments hpa`

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
app-tier	Deployment/app-tier	40%/70%	1	5	1	7m47s

- `Kubectrl top pods -n deployments`

NAME	CPU(cores)	MEMORY(bytes)
app-tier-74cd4c68c9-h74qz	2m	45Mi
app-tier-74cd4c68c9-hcg26	1m	45Mi
app-tier-74cd4c68c9-p24sr	2m	45Mi
app-tier-74cd4c68c9-pstcr	2m	45Mi
app-tier-74cd4c68c9-sfxgm	2m	46Mi
data-tier-8646dd765b-7f457	2m	2Mi
support-tier-997bc57fb-2dtq2	10m	2Mi
support-tier-997bc57fb-dvlcv	11m	2Mi
support-tier-997bc57fb-gtd2d	12m	2Mi
support-tier-997bc57fb-kvvtr	11m	2Mi
support-tier-997bc57fb-mdlrp	12m	2Mi





# Rollouts

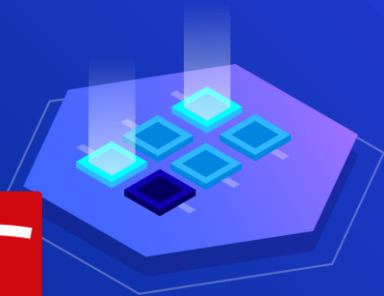
Es necesario definir una estrategia previa ejecución

```
matchLabels:  
  tier: app  
strategy:  
  rollingUpdate:  
    maxSurge: 25%  
    maxUnavailable: 25%  
  type: RollingUpdate  
template:  
  metadata:  
    creationTimestamp: null  
  labels:  
    app: microservices  
    tier: app  
spec:  
  containers:
```

Kubectrl rollout -n deployment status deployment-app-tier

Kubectrl rollout -n deployment pause deployment-app-tier

Kubectrl rollout -n deployment resume deployment-app-tier





# Rollouts

¿Necesitas ir atrás?

`Kubectl rollout -n deployment undo deployment-app-tier`

¿Necesitas seleccionar una versión del rollout?

`Kubectl rollout history -n deployment resume deployment-app-tier`



# Thanks!



# Información adicional

<https://learnk8s.io/troubleshooting-deployments>

[https://learnk8s.io/a/a-visual-guide-on-troubleshooting-kubernetes-deployments/troubleshooting-kubernetes.en\\_en.v2.pdf](https://learnk8s.io/a/a-visual-guide-on-troubleshooting-kubernetes-deployments/troubleshooting-kubernetes.en_en.v2.pdf)

<https://kubernetes.io/es/docs/concepts/workloads/controllers/deployment/>

## 2. Extra Resources

For Business Plans, Marketing Plans,  
Project Proposals, Lessons, etc



# Extra resources

