# UNIDAD 3
# ACCESO A LAS APLICACIONES EN KUBERNETES

**Noviembre 2020**

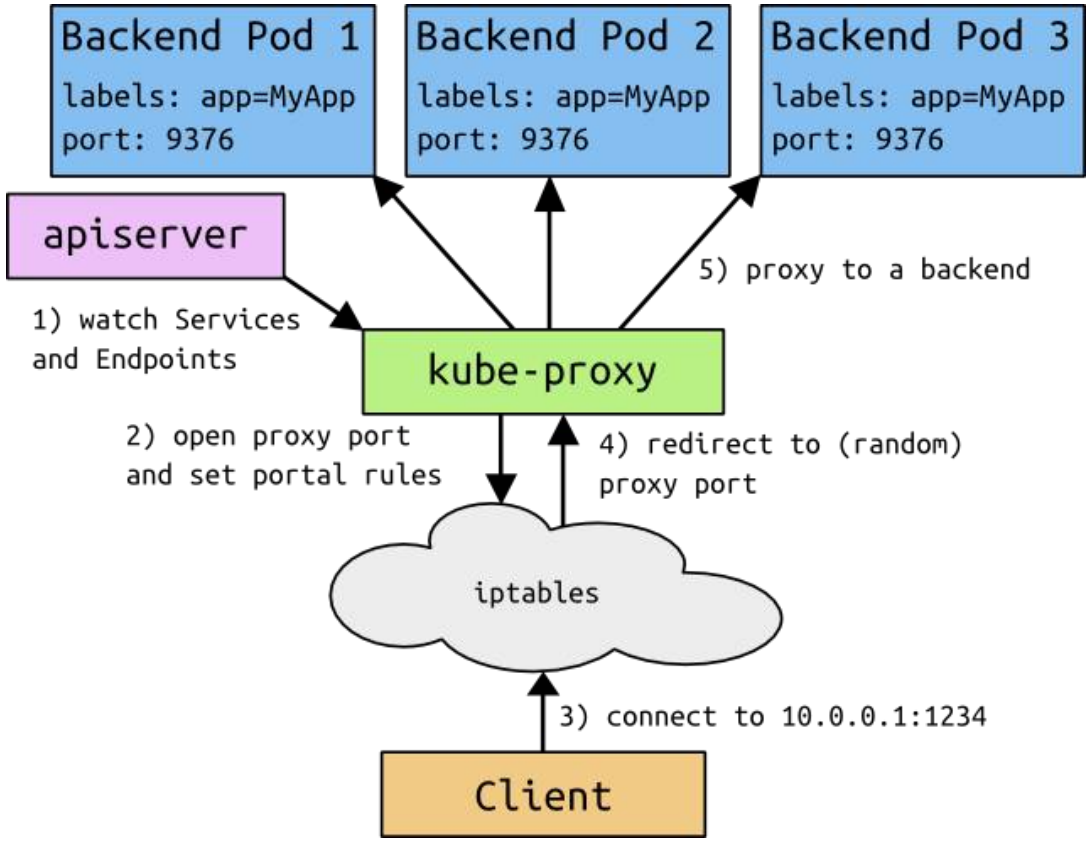
## Services

Los servicios (services) nos permiten acceder a nuestra aplicaciones.

- Un servicio es una abstracción que define un conjunto de pods que implementan un micro-servicio. (Por ejemplo el *servicio frontend*).
- Ofrecen una dirección virtual (CLUSTER-IP) y un nombre que identifica al conjunto de pods que representa, al cual nos podemos conectar.
- La conexión al servicio se puede realizar desde otros pods o desde el exterior (mediante la generación aleatoria de un puerto).

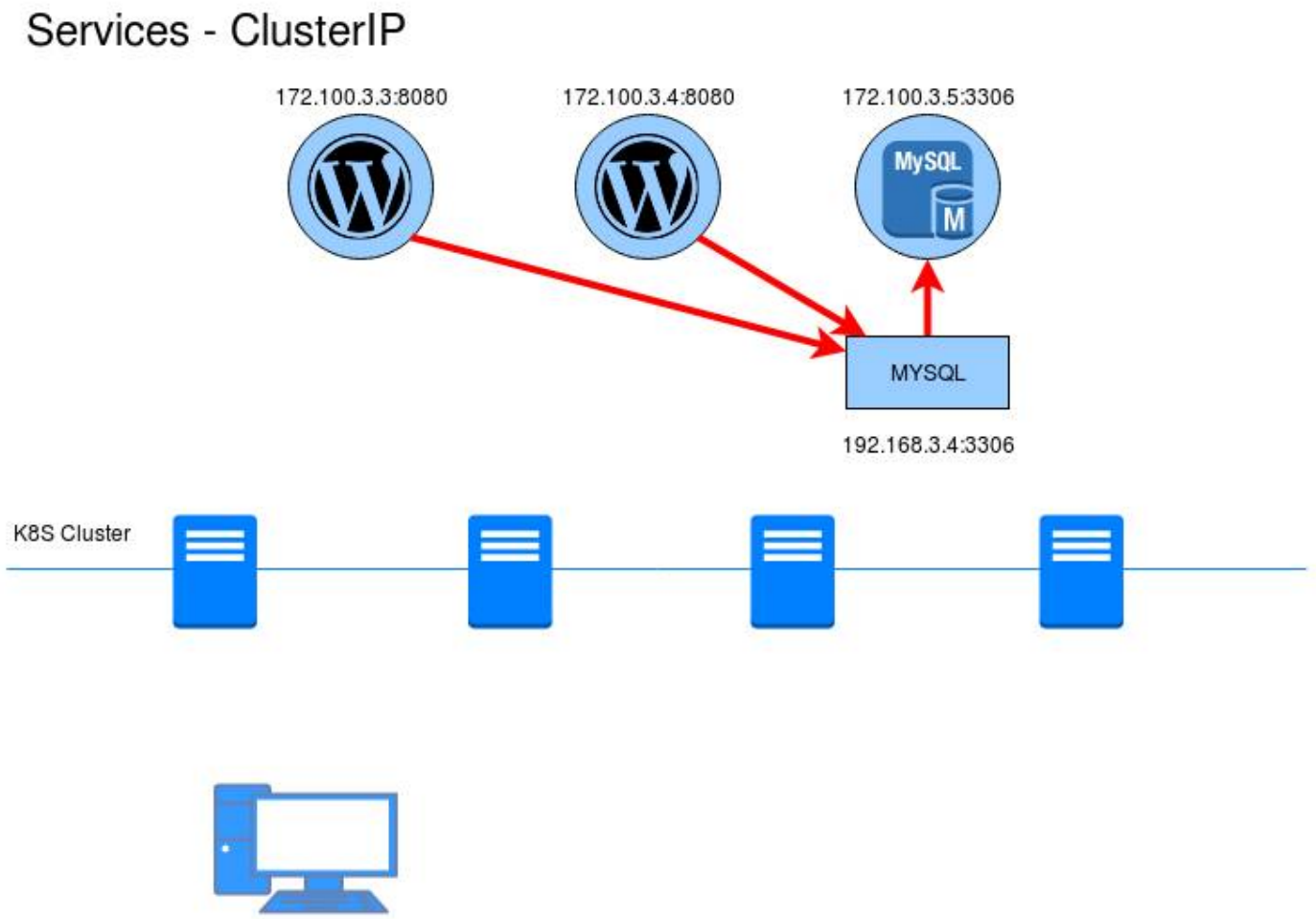
# Services

- Los servicios se implementan con iptables.
- El componente kube-proxy de Kubernetes se comunica con el servidor de API para comprobar si se han creado nuevos servicios.
- Cuando se crea un nuevo servicio, se le asigna una nueva ip interna virtual (IP-CLUSTER) que permite conexiones desde otros pods.
- Además podemos habilitar el acceso desde el exterior, se abre un puerto aleatorio que permite que accediendo a la IP del cluster y a ese puerto se acceda al conjunto de pods.
- Si tenemos más de un pod el acceso se hará siguiendo una política round-robin.



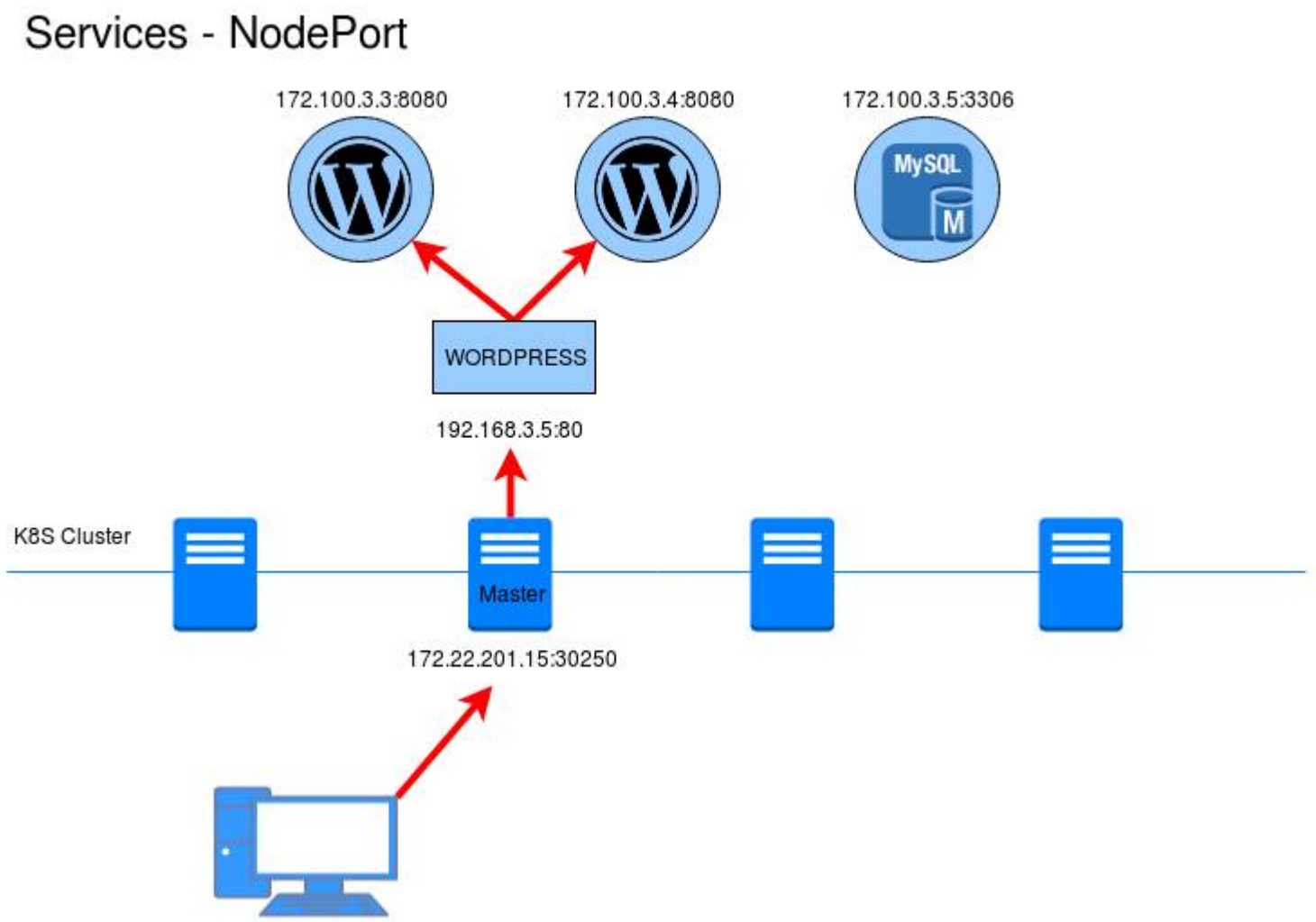
## Services ClusterIP

- **ClusterIP:** Solo permite el acceso interno entre distintos servicios. Es el tipo por defecto. Podemos acceder desde el exterior con la instrucción kubectl proxy, puede de ser gran ayuda para los desarrolladores.



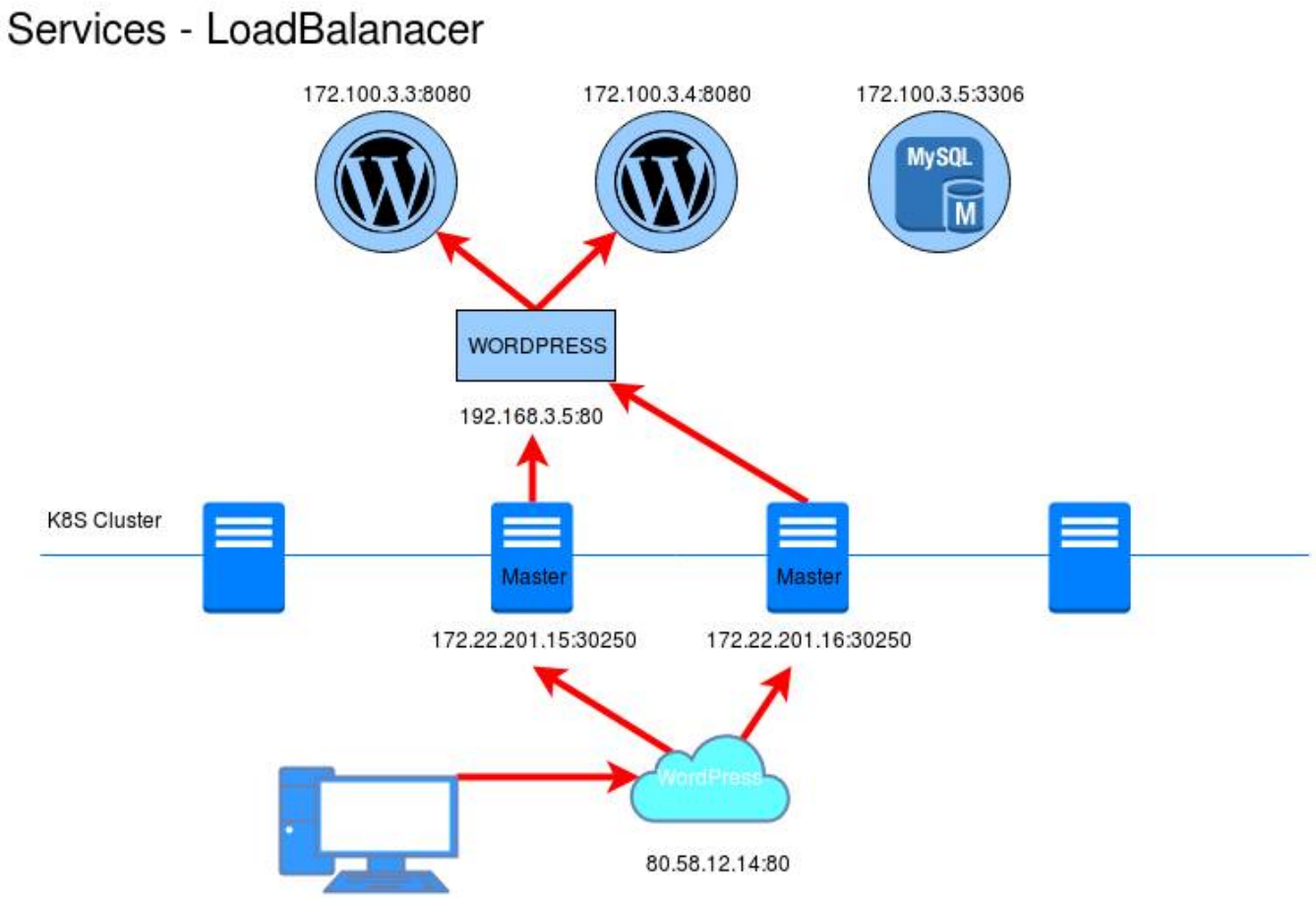
## Services NodePort

- **NodePort:** Abre un puerto, para que el servicio sea accesible desde el exterior. Por defecto el puerto generado está en el rango de 30000:40000. Para acceder usamos la ip del servidor master del cluster y el puerto asignado.



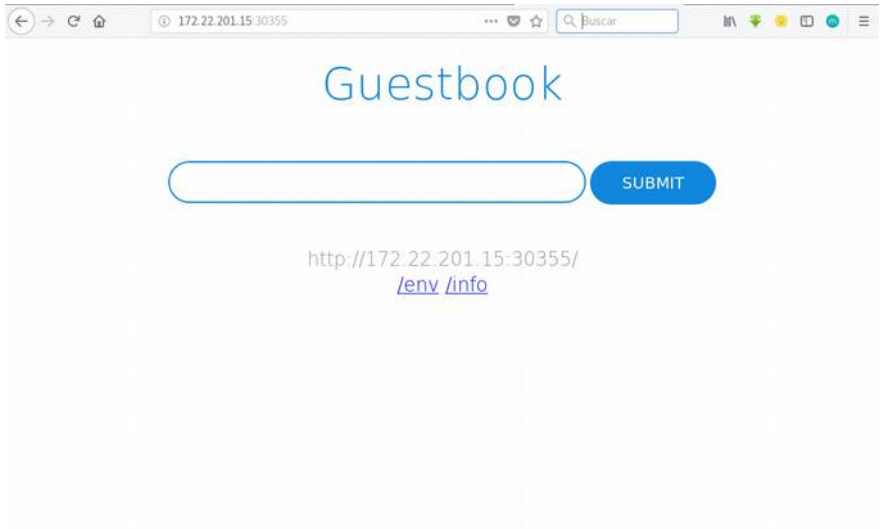
## Services LoadBalancer

- **LoadBalancer**: Este tipo sólo está soportado en servicios de cloud público (GKE, AKS o AWS). El proveedor asignara un recurso de balanceo de carga para el acceso a los servicios. si usamos un cloud privado, como OpenSatck necesitaremos un plugin para configurar el funcionamiento.



## Ejemplo 1: Describiendo objetos k8s: Services ClusterIP

## Ejemplo 2: Desplegando la aplicación GuestBook (Parte 2)



## DNS

Existe un componente de Kubernetes llamado CoreDNS, que ofrece un servidor DNS para que los pods puedan resolver diferentes nombres de recursos (servicios, pods, ...) a direcciones IP.

- Cada vez que se crea un nuevo servicio se crea un registro de tipo A con el nombre `servicio.namespace.svc.cluster.local.`

## EJEMPLO 3: Servicio para acceder a servidor remoto

Vamos a crear un servicio de tipo ClusterIP que apunte (endpoint) a un servidor remoto, para ello:

service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
 name: mariadb
spec:
 type: ClusterIP
 ports:
 - port: 3306
   TargetPort: 3306
```

endpoint.yaml

```yaml
apiVersion: v1
kind: Endpoints
metadata:
 name: mariadb
subsets:
 - addresses:
     - ip: 1.1.1.1
   ports:
     - port: 3306
```

- El nombre del servicio y del endpoint deben coincidir.
- Comprobamos que el servicio apunta al endpoint:

```
kubectl describe service mariadb
```

- Creamos  un pod con el cliente de maridb y accedemos usando el nombre del servicio:

```
kubectl apply -f mariadb-deployment.yaml

kubectl exec -it pod/mariadb -- mysql -u prueba -p -h mariadb.default.svc.cluster.local
```

## Ejemplo 3: DNS

## Ejemplo 4: Balanceo de carga

## Ejemplo5 : Servicio para acceder a servidor remoto

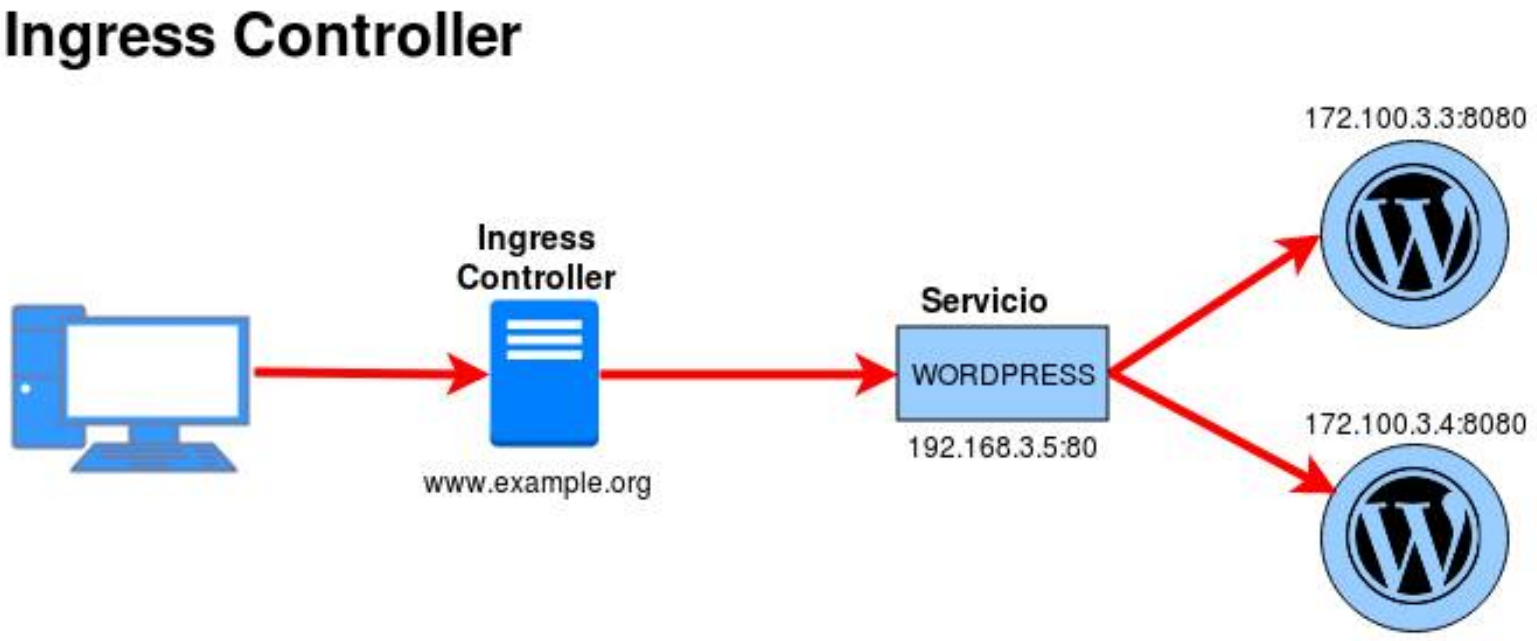
## Ejemplo 6: Despliegue canary

# Ingress Controller

Hasta ahora tenemos dos opciones principales para acceder a nuestras aplicaciones desde el exterior:

1. Utilizando servicios del tipo *NodePort*: Esta opción no es muy viable para entornos de producción ya que tenemos que utilizar puertos aleatorios desde 30000-40000.
2. Utilizando servicios del tipo *LoadBalancer*: Esta opción sólo es válida si trabajamos en un proveedor Cloud que nos cree un balanceador de carga para cada una de las aplicaciones, en cloud público puede ser una opción muy cara.

La solución puede ser utilizar un Ingress controller que nos permite utilizar un proxy inverso (HAproxy, nginx, traefik,...) que por medio de reglas de encaminamiento que obtiene de la API de Kubernetes nos permite el acceso a nuestras aplicaciones por medio de nombres.



## Ejemplo 7: Trabajando con  Ingress

## Ejemplo 8: Desplegando la aplicación LetsChat