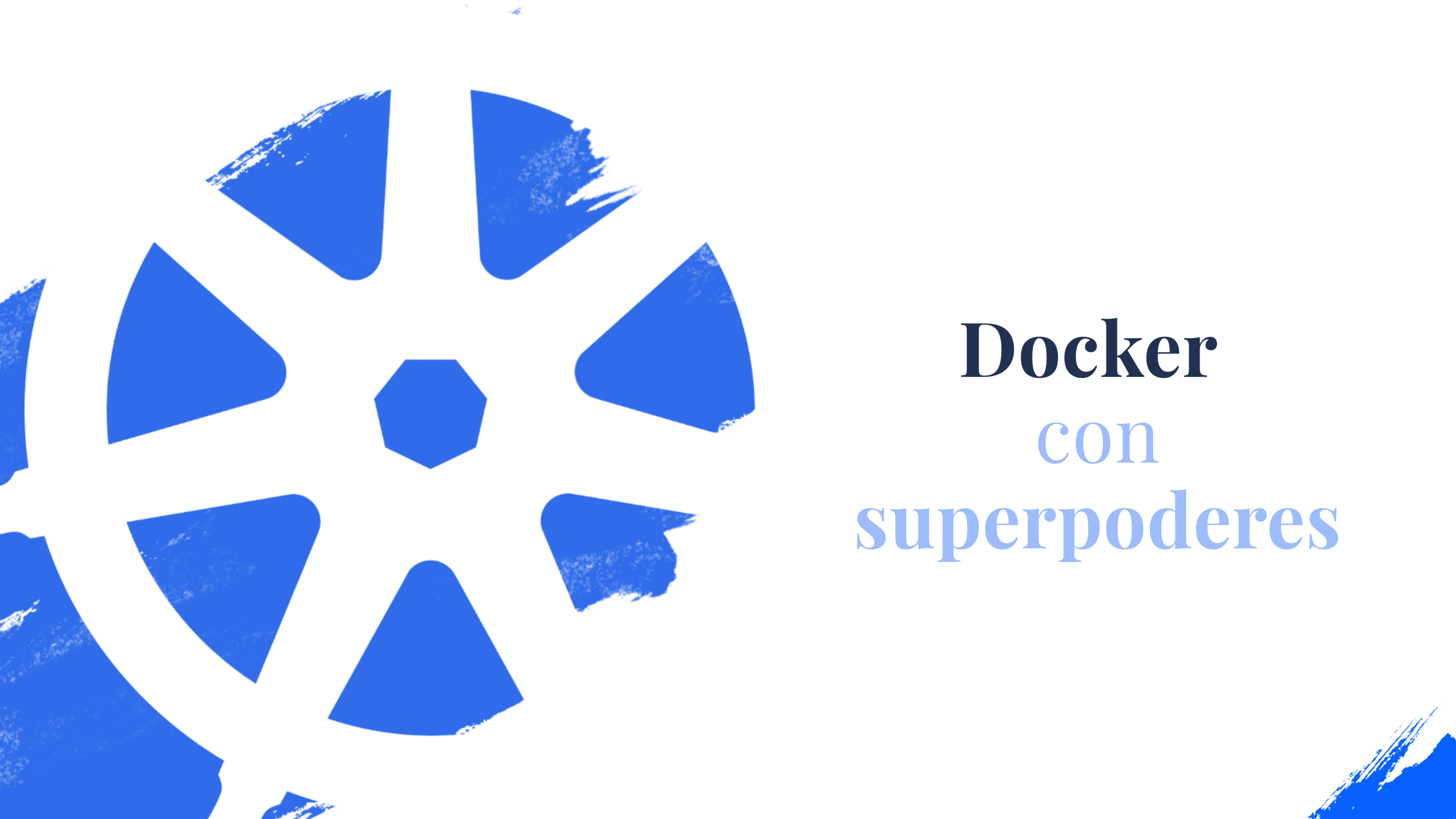


# Kubernetes

- Diego Fontán, Stéphan Jeandon, Jordan Oreiro -



Docker  
con  
superpoderes

# Enviroment Setup

```
#! /bin/bash

[ $EUID -eq 0 ] && sudo='' || sudo='sudo'

dependecies=( curl virtualbox-qt apt-transport-https kubectl )
for dep in ${dependecies[@]};do
    if [ $dep = kubectl ]; then
        if [ ! -f /etc/apt/sources.list.d/kubernetes.list ]; then
            curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | $sudo apt-key add -
            echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | $sudo tee -a /etc/apt/sources.list.d/kubernetes.list
            $sudo apt-get update -y
        fi
    fi
    if ! dpkg -l $dep >/dev/null; then
        $sudo apt-get install -y $dep
    fi
done
```

Dependencias

# Environment Setup

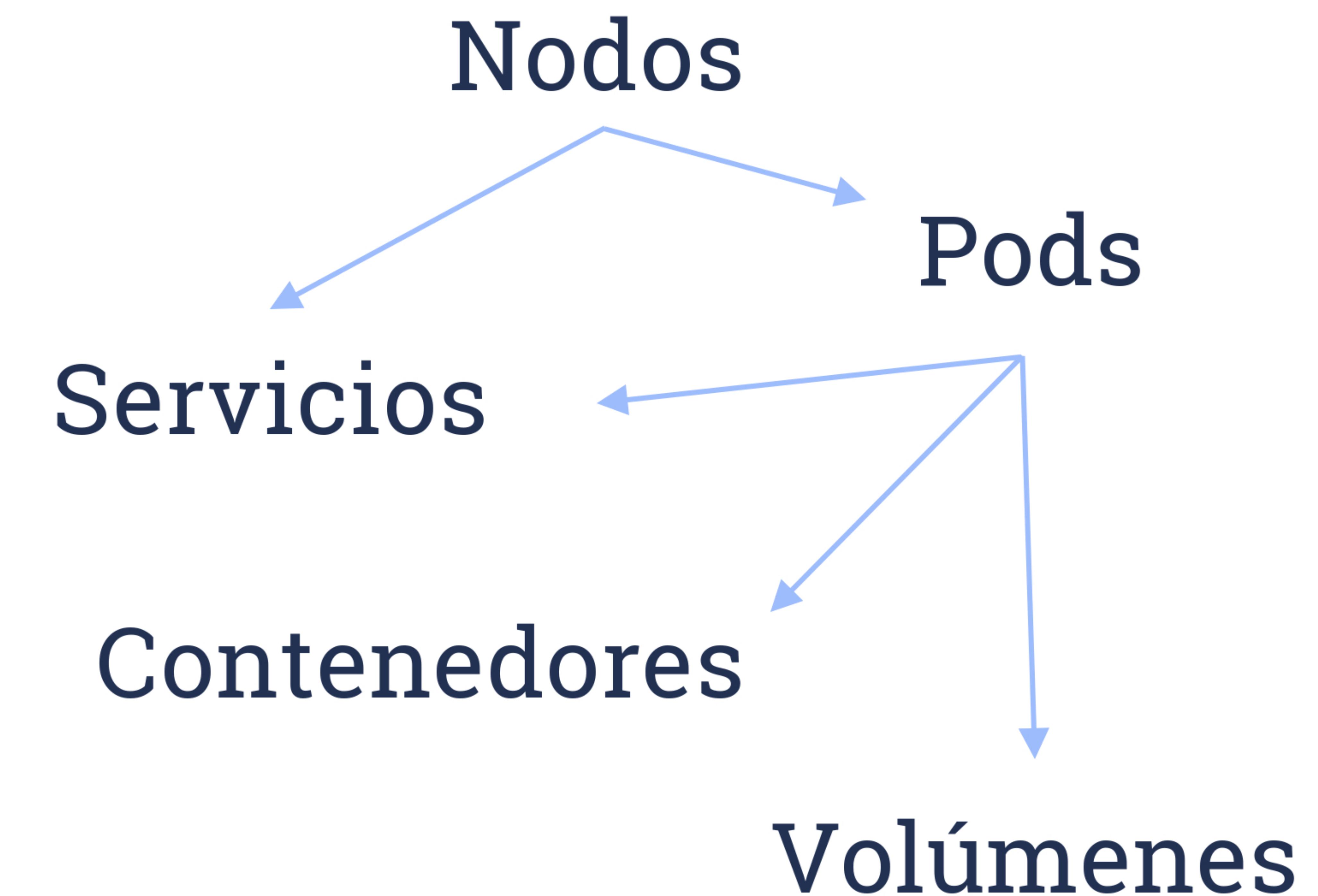
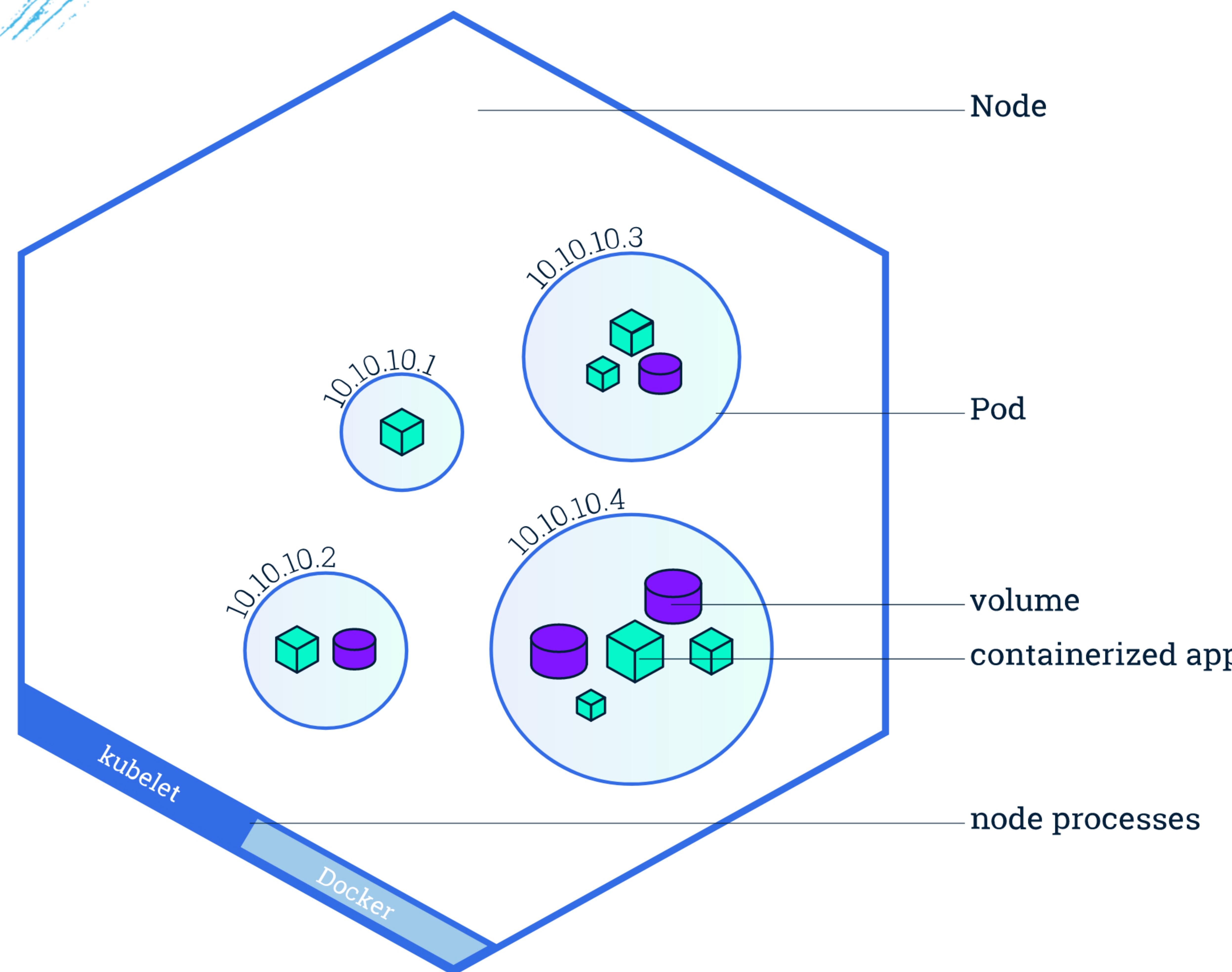
Minikube

```
MINIKUBE_VERSION() {  
    curl --silent "https://api.github.com/repos/kubernetes/minikube/releases/latest" |  
    grep '"tag_name":' | sed -E 's/.*([^"]+").*/\1/'  
}  
  
if ! which minikube &>/dev/null; then  
    curl -Lo minikube https://storage.googleapis.com/minikube/releases/$(MINIKUBE_VERSION)/minikube-linux-amd64  
    chmod +x minikube  
    $sudo mv minikube /usr/local/bin/  
fi
```

# CDA Bundle

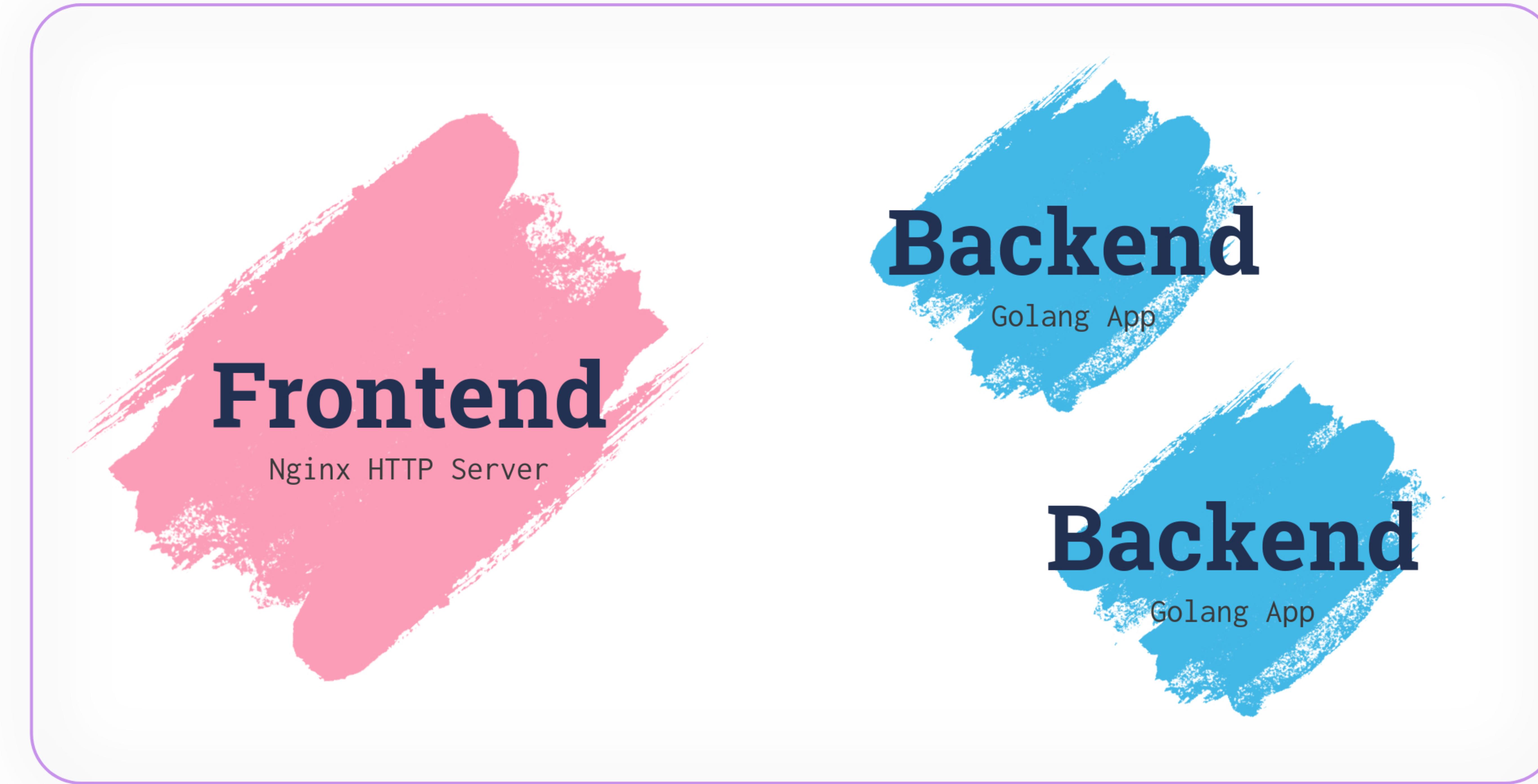
```
git clone https://github.com/Student-Puma/Homelab --branch k8s Kubernetes
```

# Infraestructura



# La parte práctica

(lo realmente divertido)



Pregunta cuál es la respuesta  
Hace una petición → Responde  
← Comunica la respuesta

```
1 apiVersion: apps/v1      # Versión YAML de K8s
2 kind: Deployment        # Táctica de despliegue
3 metadata:
4   name: hello           # Nombre de la aplicación
5 spec:
6   selector:              # Seleccionador de Pods...
7     matchLabels:
8       app: hello          # ... que se llamen 'hello'
9   replicas: 7             # Creamos 7 Pods iguales
10  template:               # Plantilla de los Pods:
11    metadata:
12      labels:
13        app: hello         # - Nombre: hello
14    spec:
15      containers:          # - Contenedor Docker usado por el Pod
16        - name: hello
17          image: "gcr.io/google-samples/hello-go-gke:1.0"
18        ports:                # Apertura de puertos
19          - containerPort: 80
```

```
1 apiVersion: v1          # Versión YAML de K8s
2 kind: Service          # Desplegamos un servicio
3 metadata:
4   name: hello           # Nombre: hello
5 spec:
6   selector:              # Seleccionador de Pods...
7     app: hello            # ... que se llamen 'hello'
8   ports:                  # Apertura de puertos:
9     - protocol: TCP      # - Protocolo TCP
10    port: 80                # - Puerto 80
11    targetPort: http      # - Servicio HTTP
```

Backend - Service

```
1 apiVersion: apps/v1          # Versión YAML de K8s
2 kind: Deployment            # Táctica de despliegue
3 metadata:
4   name: frontend             # Nombre de la aplicación
5 spec:
6   selector:                  # Seleccionador de Pods...
7     matchLabels:
8       app: hello              # ... que se llamen hello ...
9   replicas: 1                 # Creamos sólo un Pod
10  template:                  # Plantilla del Pod:
11    metadata:
12      labels:
13        app: hello            # - Nombre: hello
14    spec:
15      containers:             # - Contenedor Docker usado por el Pod
16        - name: nginx
17          image: "gcr.io/google-samples/hello-frontend:1.0"
```

```
1 apiVersion: v1          # Versión YAML de K8s
2 kind: Service          # Desplegamos un servicio
3 metadata:
4   name: frontend        # Nombre: frontend
5 spec:
6   selector:              # Seleccionador de Pods...
7     app: hello           # ... que se llamen 'hello' ...
8     tier: frontend       # ... y que sean parte del frontend
9   ports:                 # Apertura de puertos:
10    - protocol: "TCP"    # - Protocolo TCP
11      port: 80            # - Puerto 80 (escucha)
12      targetPort: 80      # - Puerto 80 (port-forward)
13    type: LoadBalancer   # Distribuye las peticiones ...
14                                # ... entre los diferentes backend
```

# Build & Run

```
# Iniciar Kubernetes
minikube start --vm-driver virtualbox
# Desplegar las aplicaciones y servicios
kubectl create --filename backend.yaml
kubectl create --filename frontend.yaml
# Configurar reenvio de puertos en local
kubectl port-forward
  $(kubectl get pods -l tier=frontend
    -o go-template --template '{{range .items}}{{.metadata.name}}{{end}}') \
  8080:80
```

# Test

```
# Mandamos una petición GET al frontend  
curl localhost:8080
```

```
# Respuesta obtenida del backend  
{"message": "Hello"}
```



Gracias  
por vuestra  
atención