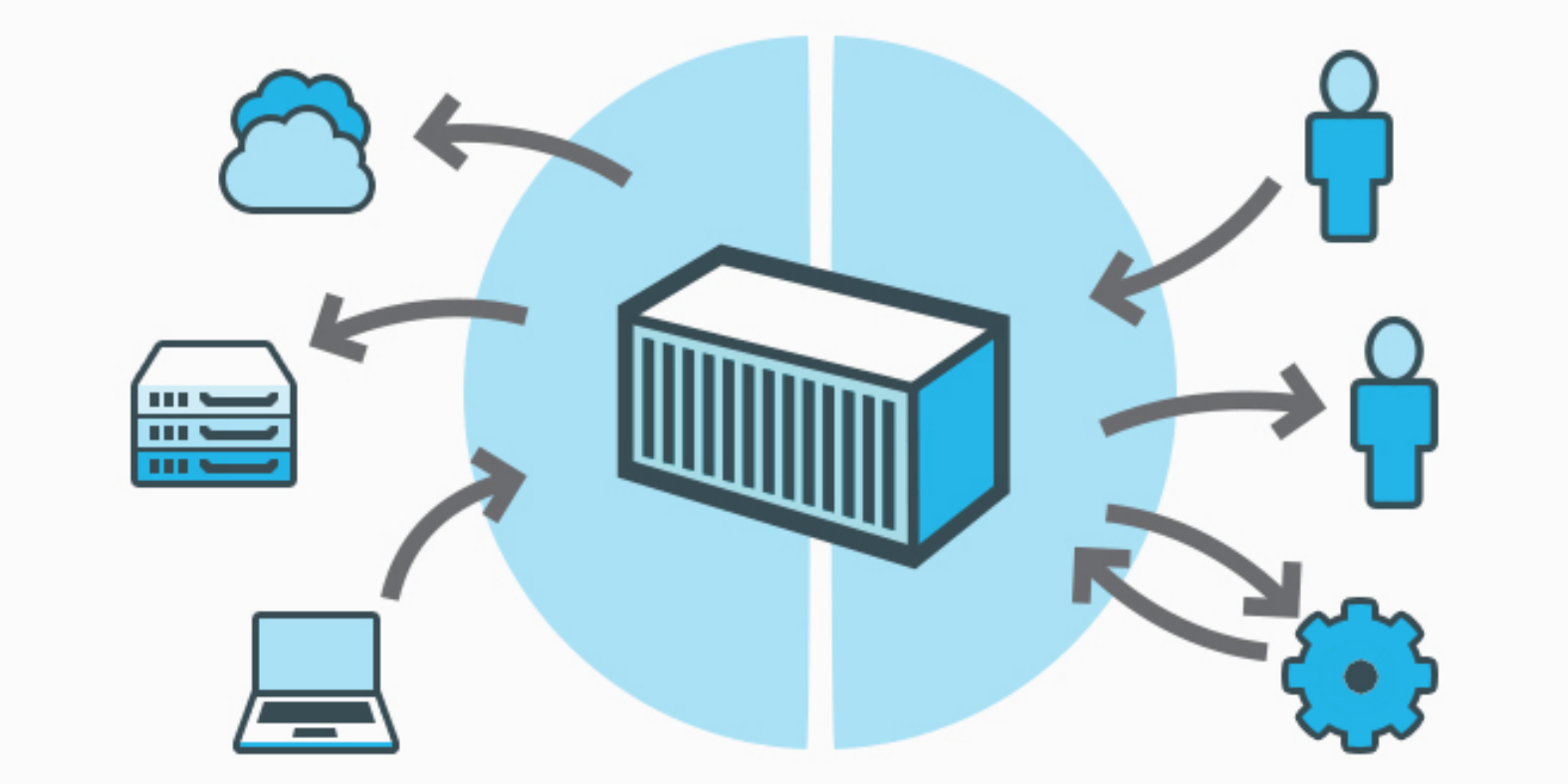
Tema 3

Kubernetes

Universidad
Rey Juan Carlos


docker

Docker



Servicios en producción

- **Funcionalidades deseables**
  - Asistencia en el despliegue y en la actualización
  - Reinicio si el servicio finaliza o no responde a las peticiones (*resiliencia*)
  - Uso de más recursos hardware la carga aumenta (*escalabilidad*)
  - Aprovechamiento de recursos hardware compartiendo pero sin interferencias (*multitenancy*)
  - Gestión de configuraciones y secretos
  - Monitorización

Servicios en producción

- Existen diversas formas de **desplegar servicios en contenedores en producción**
  - Ejecución de contenedores con docker o docker-compose en **una máquina** (física o virtual)
  - Instalación de un **orquestador de contenedores** en un cluster de VMs
  - Uso de **servicios de orquestación** ofrecidos por el proveedor cloud

Servicios en producción

- **Orquestadores de contenedores**
  - Ofrecen funcionalidades de gestión de servicios en producción para **desplegar servicios en contenedores**
  - Gestionan un **cluster de maquinas (nodos)** de forma global
  - Gestionan **varios contenedores** como una única unidad lógica (aplicación)
  - **Varias aplicaciones** se pueden desplegar en un mismo cluster y se reparten los recursos sin interferencias

Servicios en producción

- Orquestadores de contenedores

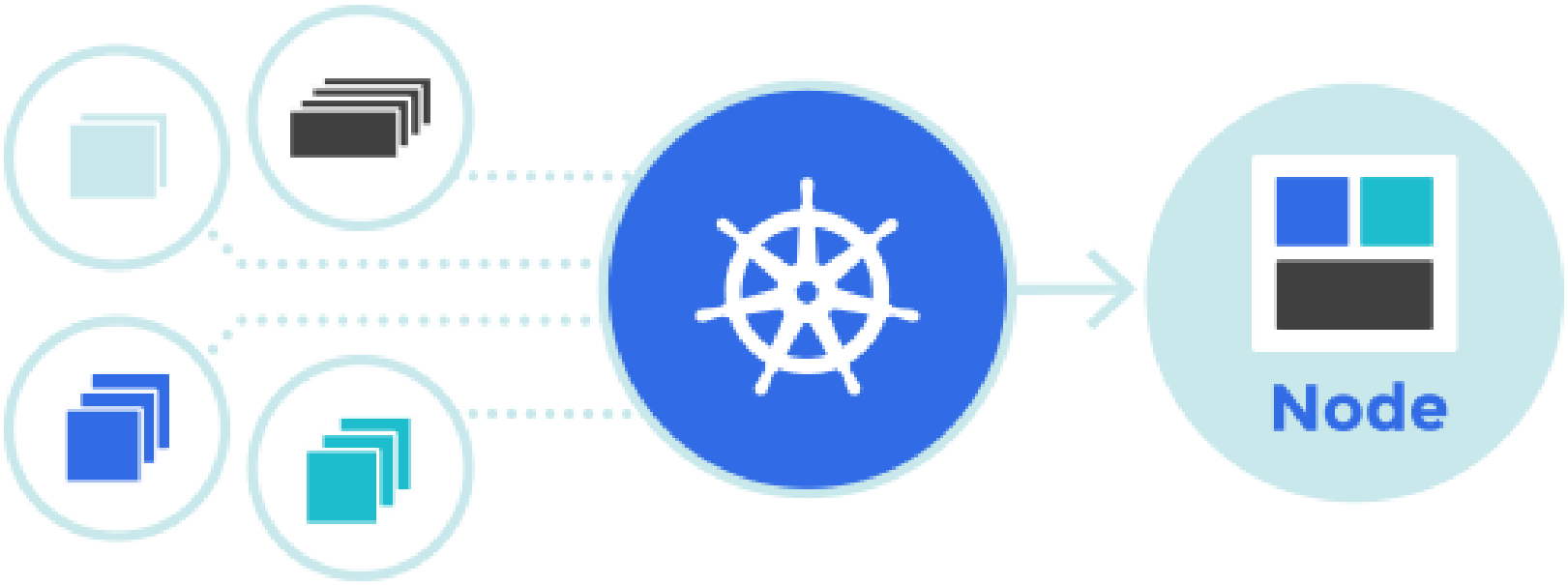
HashiCorp
Nomad

Apache
MESOS™

kubernetes

code
URJC

kubernetes


Node

Kubernetes

- Desarrollado inicialmente por **Google** (basado en Borg)
- Es muy **maduro** y existen muchas aplicaciones en **producción** sobre Kubernetes
- Está desarrollado bajo la **Cloud Native Computing Fundation** con múltiples miembros de peso

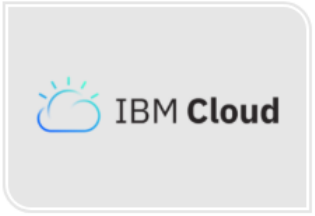
kubernetes

CLOUD NATIVE COMPUTING FOUNDATION

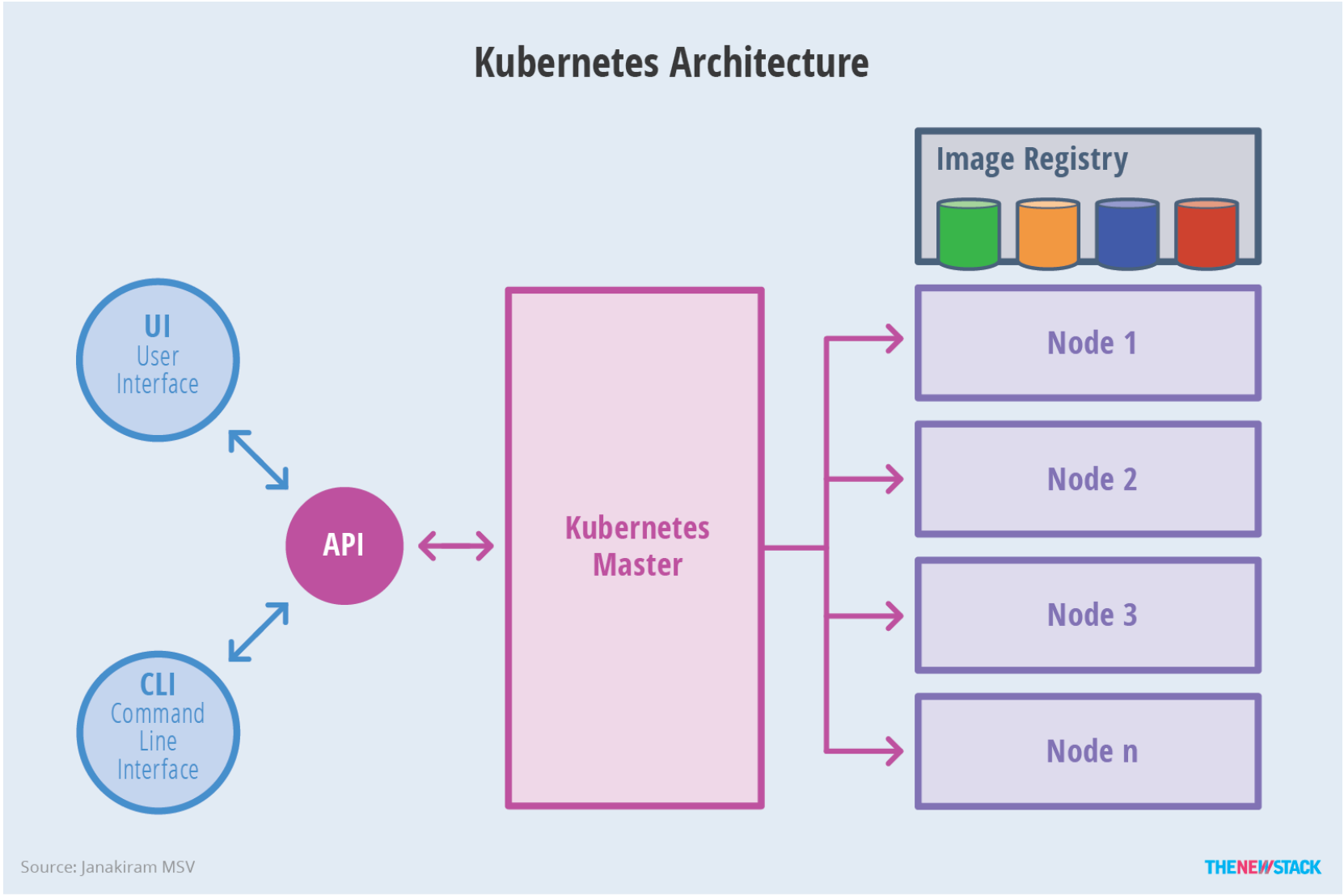
http://kubernetes.io/

https://www.cncf.io/

CLOUD NATIVE COMPUTING FOUNDATION



https://www.cncf.io/about/members/



https://thenewstack.io/kubernetes-an-overview/

Kubernetes

- **Kubernetes** (**k8s**) es un software diferente a docker que tiene que instalarse por separado
- Habitualmente usa **docker** o **containerd** internamente para ejecutar los contenedores, aunque también puede funcionar con otras tecnologías como **rkt**, **katacontainers**.. (menos maduras)

https://coreos.com/rkt/

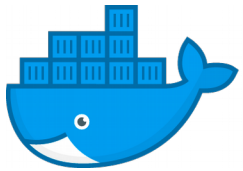
https://containerd.io/

Kubernetes en local

- Para **desarrollo** se puede instalar un kubernetes en tu ordenador local
  - **Minikube** (un nodo)
  - Kube-spawn (múltiples nodos)
  - Docker for Windows y Mac

minikube

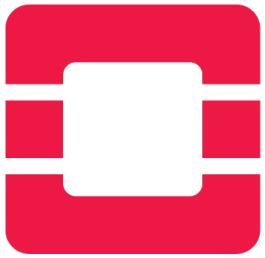
kube-spawn



docker

Kubernetes en un cluster

- Instalado en servidores físicos (bare metal)
- En máquinas virtuales en nube privada



openstack®

vmware®

Kubernetes en un cluster

- Instalado en máquinas virtuales (instancias) de AWS

https://aws.amazon.com/es/quickstart/architecture/heptio-kubernetes/

https://github.com/kubernetes/kops

Kubernetes gestionado

- Gestionado por un proveedor de computación en la nube

https://cloud.google.com/kubernetes-engine/

https://www.openshift.com/learn/topics/kubernetes/

https://aws.amazon.com/es/eks/

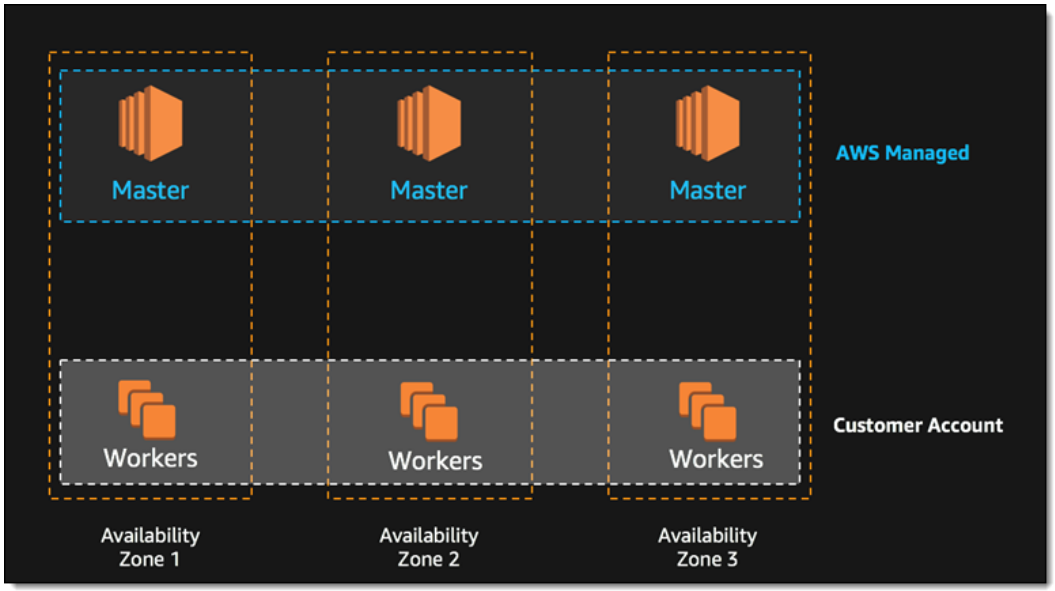
https://azure.microsoft.com/services/container-service/

Kubernetes gestionado

- **¿Cómo se paga?**
  - Habitualmente el **nodo maestro** (o los nodos si están **replicados**) es gestionado por el proveedor y no es accesible por el usuario
  - Algunos **regalan** el nodo maestro, **AWS cobra** 0,10$ por hora (unos 100$ mes)
  - Cobro por ejecución de aplicaciones:
    - Por nodo worker del cluster (esté usado o no)
    - Por contenedor en ejecución (algunos proveedores)

Kubernetes gestionado

- **EKS** (Elastic Kubernetes Service)



https://aws.amazon.com/es/eks/

Kubernetes gestionado

- **EKS** (Elastic Kubernetes Service)

https://aws.amazon.com/es/fargate/

Kubernetes gestionado

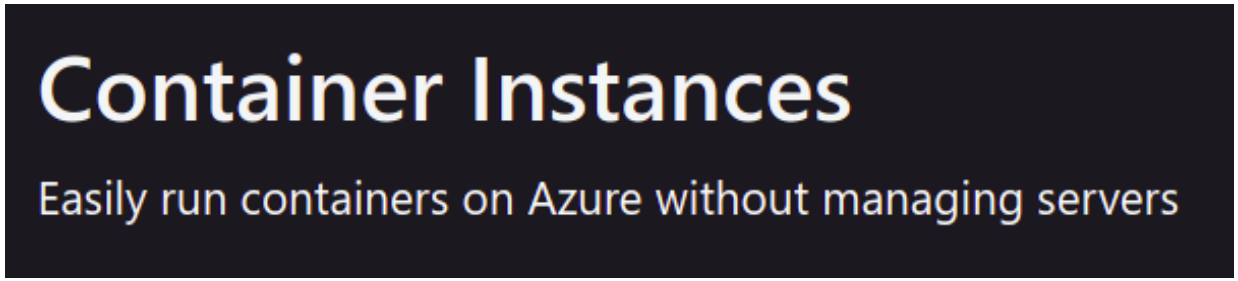
- **AKS** (Azure Kubernetes Service)

https://azure.microsoft.com/en-us/services/kubernetes-service/

Kubernetes gestionado

- **AKS** (Azure Kubernetes Service)

ACI (Azure Container Instances)



https://docs.microsoft.com/es-es/azure/container-instances/container-instances-orchestrator-relationship

Kubernetes gestionado

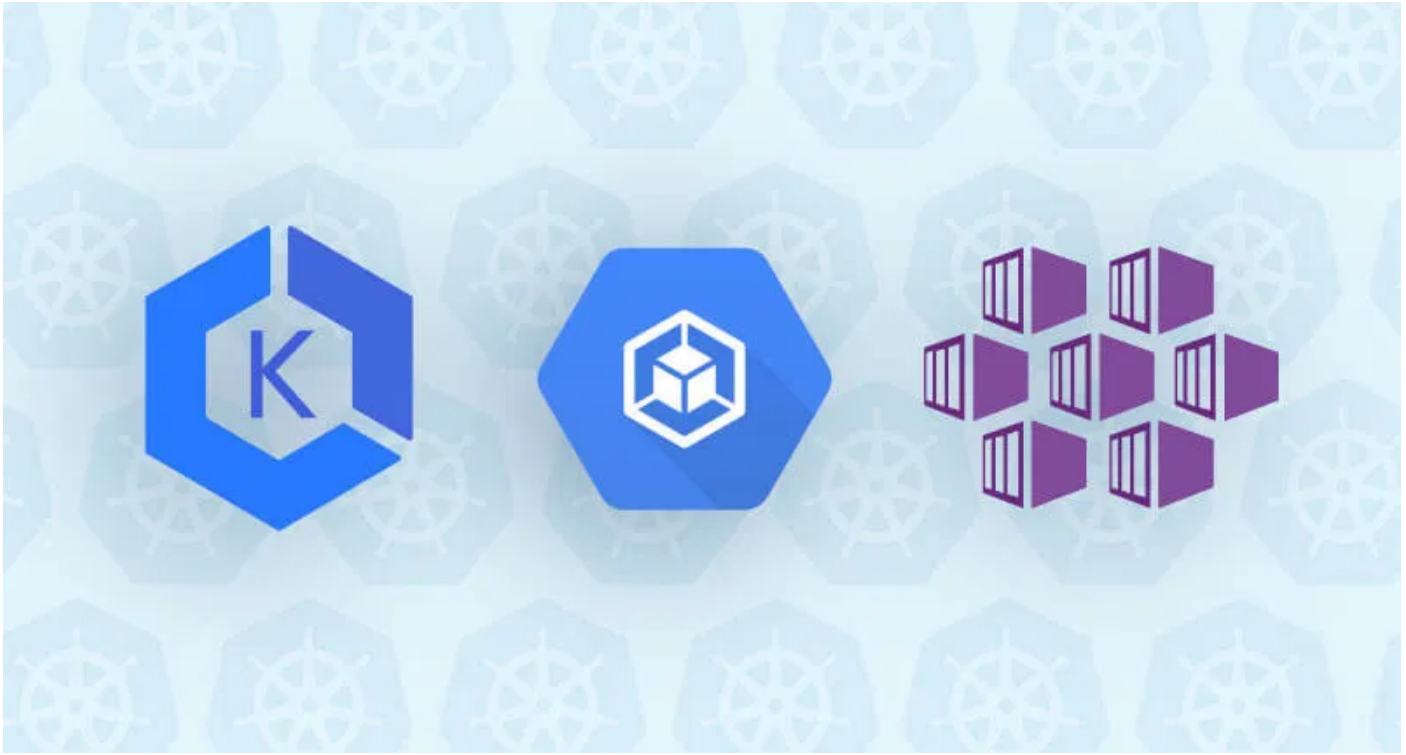
- **GKE** (Google Kubernetes Engine)

Google Cloud

https://cloud.google.com/kubernetes-engine

Kubernetes gestionado

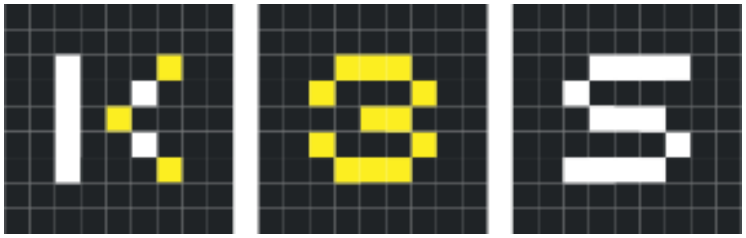
- ¿Cuál es mejor?



http://www.vamsitalkstech.com/?p=8423

Kubernetes “mini”

- Hay versiones mini de Kubernetes diseñadas para gestionar contenedores en **dispositivos con pocos recursos (IoT, coches…)**



https://k3s.io/

MicroK8s

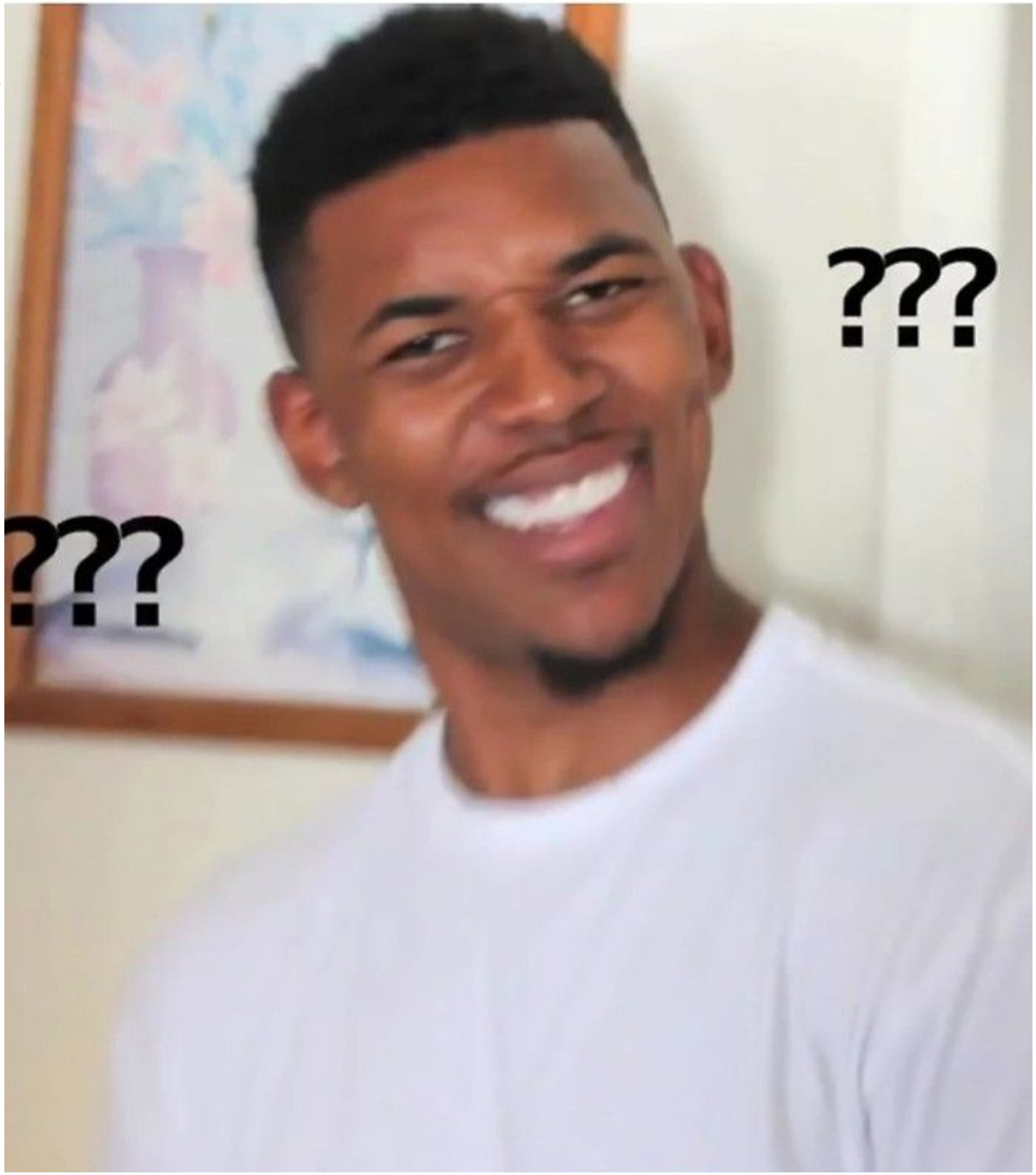
https://microk8s.io/

Kubernetes en docker

code URJC

¿Y si metemos cada uno de los nodos de un cluster Kubernetes en un contenedor docker?

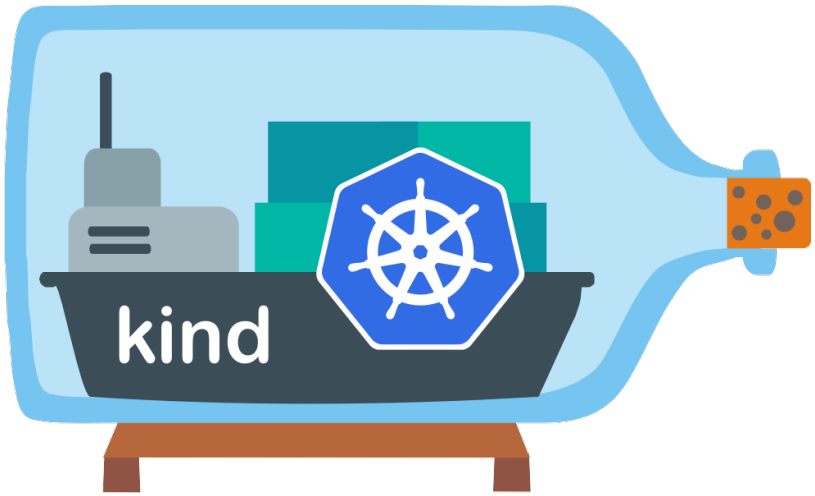
Kuber



code URJC

Kubernetes en docker

- Ideal para testing en CI y desarrollo en local

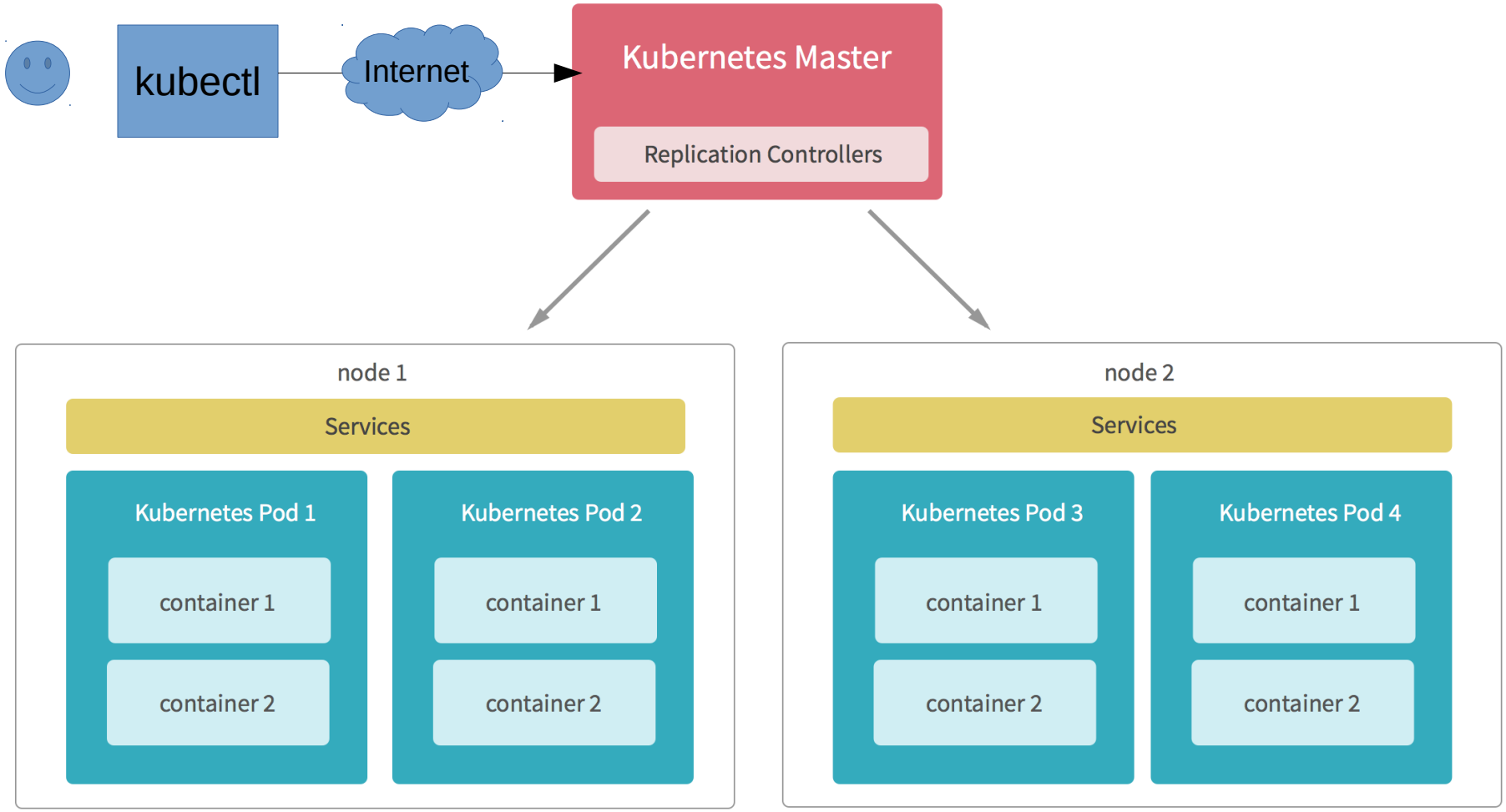


https://kind.sigs.k8s.io/

rancher / k3d

https://github.com/rancher/k3d

Componentes Kubernetes



Cliente Kubernetes

- **Kubectl**
  - Es una herramienta por **línea de comandos (CLI)** local para **controlar** clusters Kubernetes y las aplicaciones que se ejecutan en él
  - Está **implementada en Go**, no es necesario ningún runtime o VM para que se pueda ejecutar, existe un binario para cada SO
  - Similar al **comando docker**, pero con opciones y funcionalidades diferentes (más avanzadas)
  - **Instalación**

https://kubernetes.io/docs/user-guide/prereqs/

Kubernetes en local

- **Minikube**
  - Una versión básica de Kubernetes para instalar en una máquina de desarrollo
  - Funciona en linux, windows y mac
  - Tiene un único nodo virtualizado con **VirtualBox** (se puede usar otro hypervisor)

minikube

https://github.com/kubernetes/minikube

Minikube

- Instalación en cualquier sistema operativo

https://github.com/kubernetes/minikube/releases

- Arrancar el mini cluster (inicia una VM y puede tardar varios minutos)

```
$ minikube start
```

- Podemos controlar los recursos de nuestra máquina que asignamos a la máquina virtual

```
$ minikube start --memory=4098 --cpus=4
```

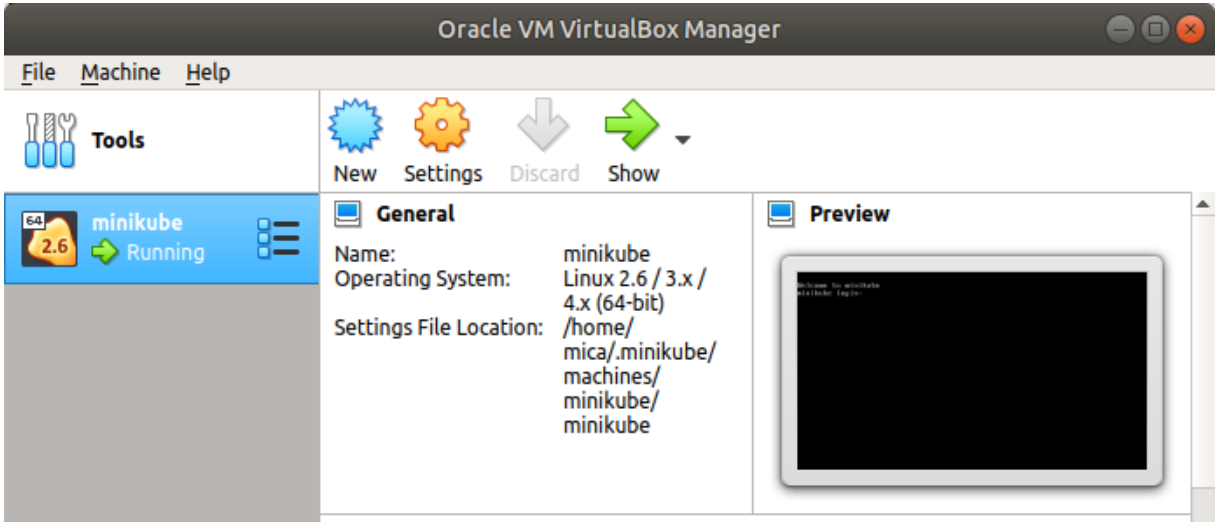
Minikube

- Parar minikube (guardando los datos en la VM)

```
$ minikube stop
```

- Borrar minikube

```
$ minikube delete
```



Minikube

- **Acceso directo a la máquina virtual**
  - Uso de herramientas docker (docker logs, docker ps)
  - Creación de imágenes para ser ejecutadas por kubernetes
  - Acceso por ssh

```
$ minikube ssh
```

  - Montaje de carpeta del host

```
$ minikube mount /host-folder:/vm-folder
```

https://github.com/kubernetes/minikube/blob/master/docs/host_folder_mount.md

Minikube

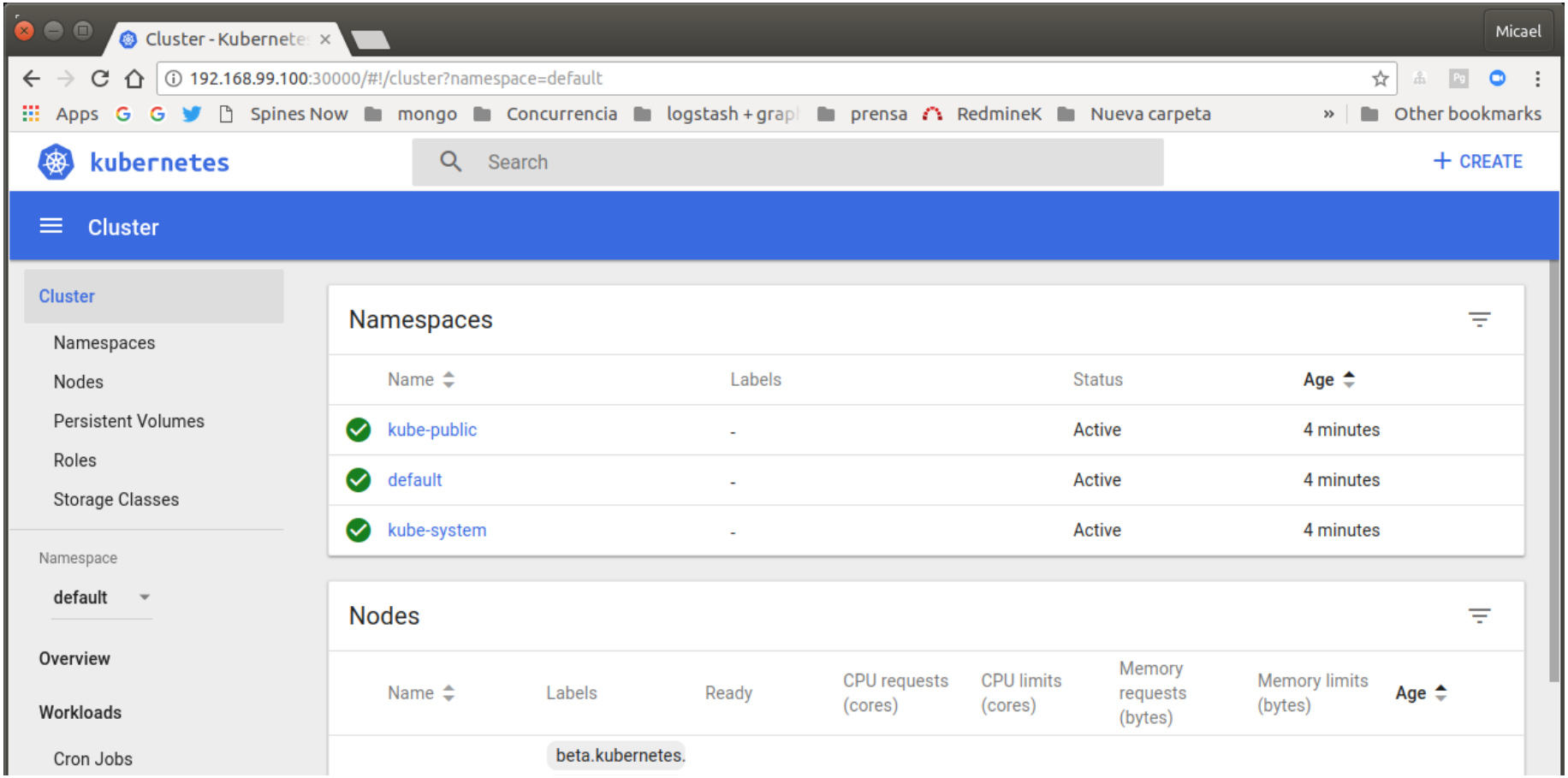
- **Acceso directo a la máquina virtual**
  - Los servicios propios de Kubernetes también se ejecutan como contenedores dentro del cluster
  - Si ejecutamos docker ps podemos ver los contenedores en ejecución

```
$ docker ps
CONTAINER ID        IMAGE                           COMMAND                  CREATED             STATUS              PORTS
                    NAMES
997d1eb672ad        225fded0176e                    "/app-entrypoint.sh …"   2 seconds ago       Up 2 seconds
                    k8s_drupal_my-drupal2-drupal-8c4d984d6-2q9hm_default_1fcea773-39e5-11e9-85bf-0800272d828f_3
8117331666bc        k8s.gcr.io/metrics-server-amd64   "/metrics-server --s…"   32 seconds ago      Up 31 seconds
                    k8s_metrics-server_metrics-server-6fc4b7bcff-w95hz_kube-system_c5108a55-39df-11e9-85bf-0800272d828f_3
d5fdeddead3e        98db19758ad4                    "/usr/local/bin/kube…"   34 seconds ago      Up 33 seconds
                    k8s_kube-proxy_kube-proxy-sklcj_kube-system_1e3fab5d-3f7f-11e9-b3f0-0800272d828f_0
f12fc39248cc        k8s.gcr.io/pause:3.1            "/pause"                 34 seconds ago      Up 33 seconds
                    k8s_POD_kube-proxy-sklcj_kube-system_1e3fab5d-3f7f-11e9-b3f0-0800272d828f_0
d82ebc6896ff        01bd760f276c                    "/entrypoint.sh /ngi…"   38 seconds ago      Up 38 seconds
                    k8s_nginx-ingress-controller_nginx-ingress-controller-7c66d668b-bdt8p_kube-system_1c6cda72-350c-11e9-8799-0800
132682e4ec25        8cb3de219af7                    "/run.sh"                About a minute ago  Up About a minute
                    k8s_grafana_influxdb-grafana-9sp4s_kube-system_e6b6b17d-39be-11e9-85bf-0800272d828f_2
26268346a8d6        d8233ab899d4                    "/bin/sh"                About a minute ago  Up About a minute
                    k8s_load-generator_load-generator_default_9536a00a-39e0-11e9-85bf-0800272d828f_3
```

Control del cluster

- Dashboard gráfico

```
$ minikube dashboard
```



Control del cluster

- **Comando kubectl**
  - Nodos del cluster

```
$ kubectl get nodes
```

```
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   15d   v1.13.3
```

  - Versión de kubectl y de Kubernetes

```
$ kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"13", GitVersion:"v1.13.3",
GitCommit:"721bfa751924da8d1680787490c54b9179b1fed0", GitTreeState:"clean", BuildDate:"2019-
02-01T20:08:12Z", GoVersion:"go1.11.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"13", GitVersion:"v1.13.3",
GitCommit:"721bfa751924da8d1680787490c54b9179b1fed0", GitTreeState:"clean", BuildDate:"2019-
02-01T20:00:57Z", GoVersion:"go1.11.5", Compiler:"gc", Platform:"linux/amd64"}
```

Pods

- Uno o más **contenedores** que se pueden desplegar como una única unidad
- Pueden **colaborar** estrechamente
  - Compartiendo **ficheros** con volúmenes compartidos
  - Comunicándose entre a través de la **red** usando localhost
- Unidad mínima de ejecución de contenedores en Kubernetes

https://kubernetes.io/docs/tutorials/kubernetes-basics/

https://kubernetes.io/docs/getting-started-guides/minikube/

Pods

- Crear un pod en el cluster > Ejecutar uno o varios contenedores

```
$ kubectl run simpleservice --generator=run-pod/v1 \
--image=mhausenblas/simpleservice:0.5.0 --port=9876
```

- Consultar pods en ejecución

```
$ kubectl get pods
```

```
NAME                READY   STATUS    RESTARTS   AGE
simpleservice       1/1     Running   0          8s
```

https://hub.docker.com/r/mhausenblas/simpleservice/

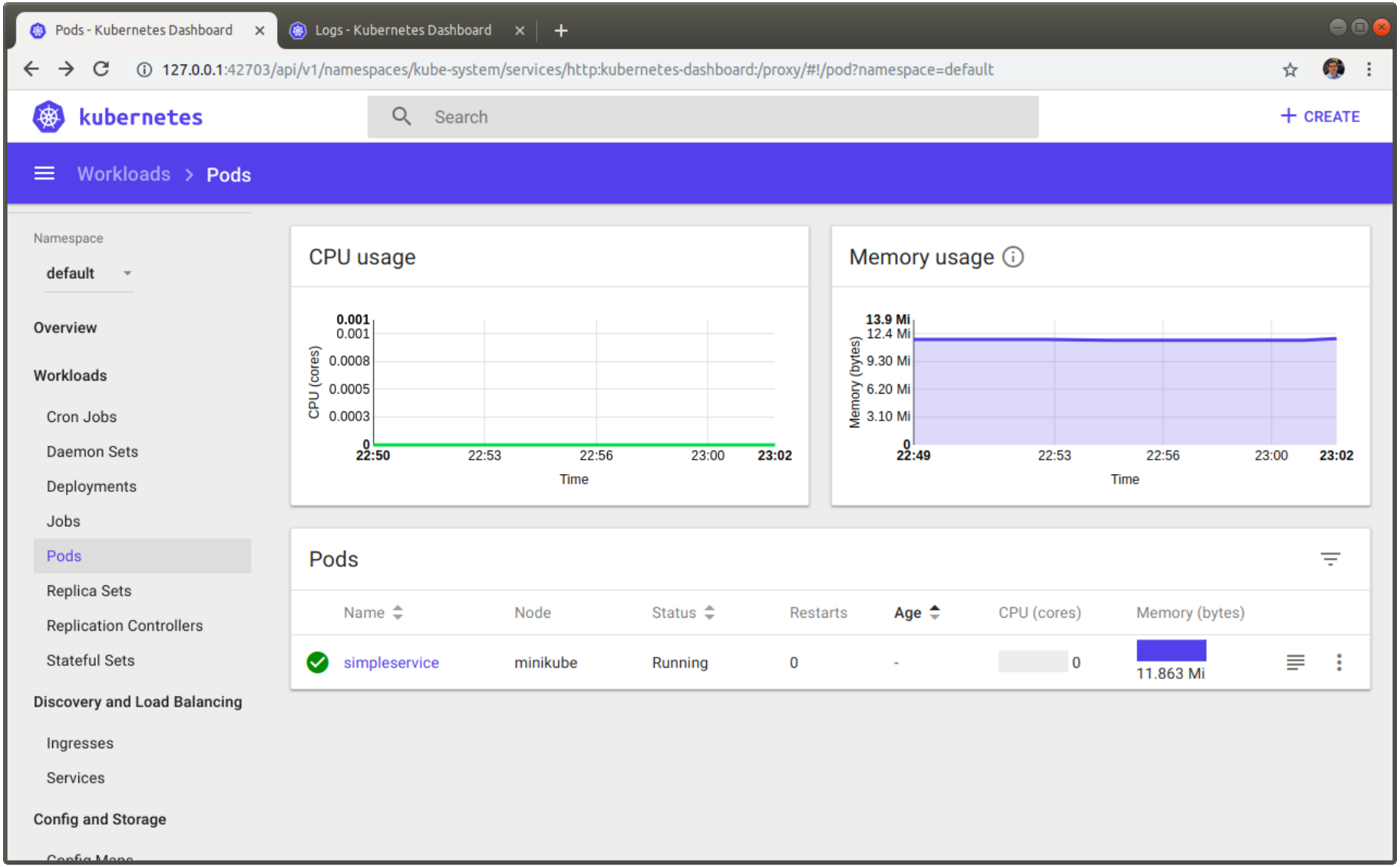
Pods

- Consultar información el pod

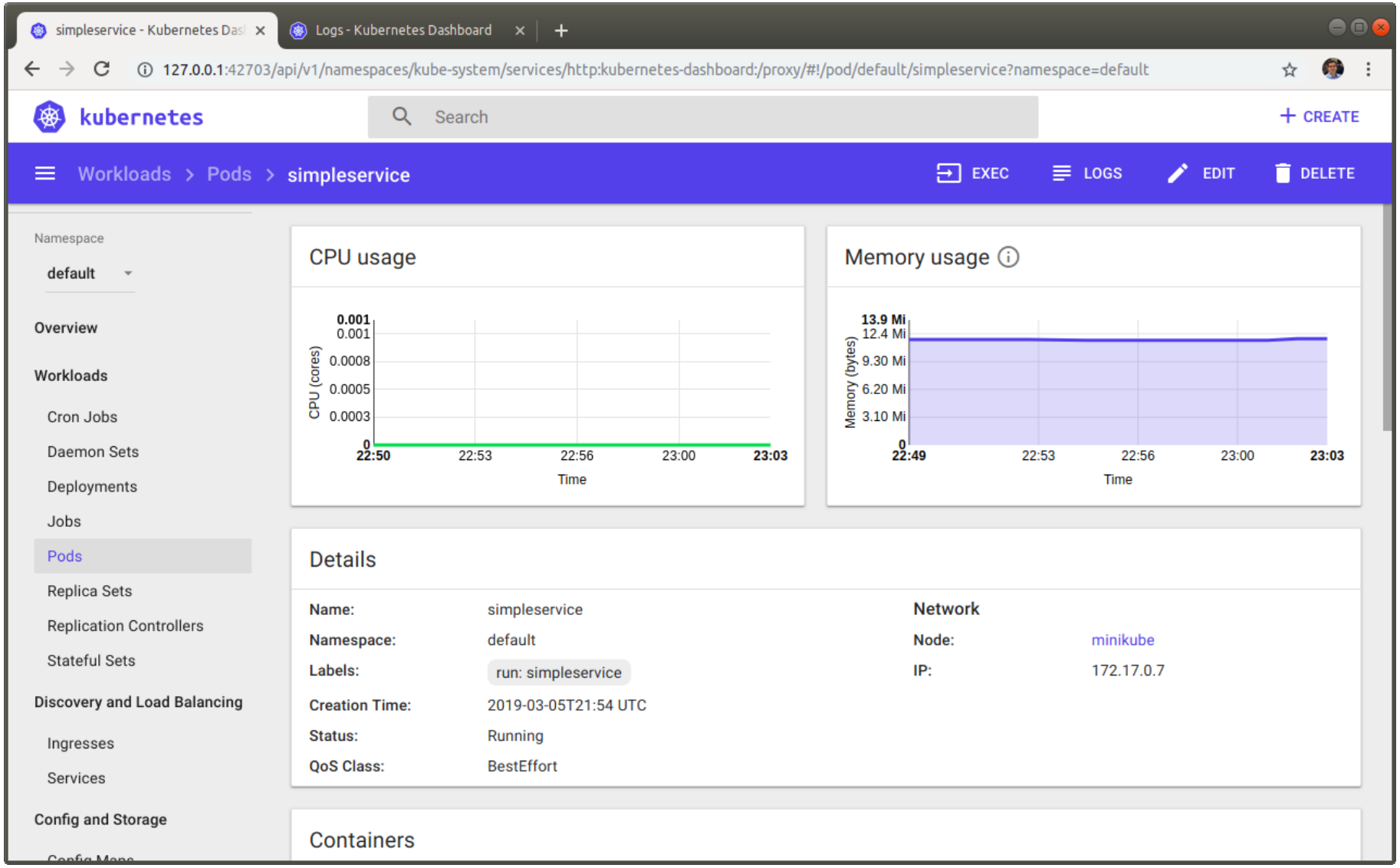
```
$ kubectl describe pods/simpleservice
```

```
Name:               simpleservice
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               minikube/10.0.2.15
Start Time:         Tue, 05 Mar 2019 22:54:16 +0100
Labels:             run=simpleservice
Annotations:        <none>
Status:             Running
IP:                 172.17.0.7
Containers:
  simpleservice:
    Container ID:   docker://a8f5e85734d77a219b59a5...
    Image:          mhausenblas/simpleservice:0.5.0
    Image ID:       docker-pullable://mhausenblas/simpleservice@sha256:33f58...
    Port:           9876/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 05 Mar 2019 22:54:17 +0100
    Ready:          True
...
```

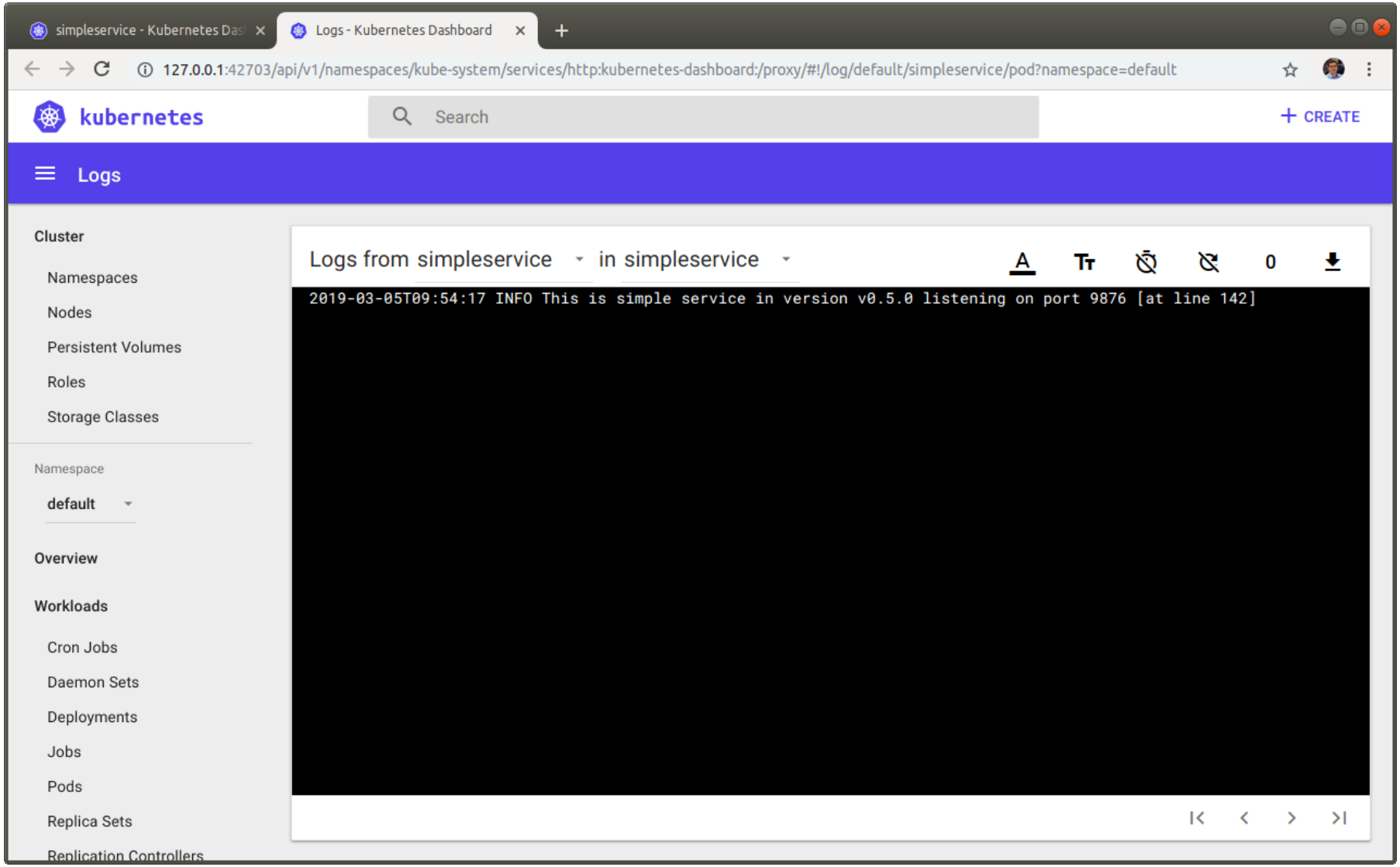
Pods


Pods - Kubernetes Dashboard
Logs - Kubernetes Dashboard
127.0.0.1:42703/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard:/proxy/#!/pod?namespace=default
kubernetes
Search
+ CREATE
Workloads > Pods
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
Ingresses
Services
Config and Storage
CPU usage
CPU (cores)
0.001
0.001
0.0008
0.0005
0.0003
0
22:50
22:53
22:56
23:00
23:02
Time
Memory usage
Memory (bytes)
13.9 Mi
12.4 Mi
9.30 Mi
6.20 Mi
3.10 Mi
0
22:49
22:53
22:56
23:00
23:02
Time
Pods
Name
Node
Status
Restarts
Age
CPU (cores)
Memory (bytes)
simpleservice
minikube
Running
0
-
0
11.863 Mi

Pods


simpleservice - Kubernetes Das
Logs - Kubernetes Dashboard
127.0.0.1:42703/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard:/proxy/#!/pod/default/simpleservice?namespace=default
kubernetes
Search
CREATE
Workloads > Pods > simpleservice
EXEC
LOGS
EDIT
DELETE
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
Ingresses
Services
Config and Storage
CPU usage
0.001
0.001
0.0008
0.0005
0.0003
0
CPU (cores)
22:50
22:53
22:56
23:00
23:03
Time
Memory usage
13.9 Mi
12.4 Mi
9.30 Mi
6.20 Mi
3.10 Mi
0
Memory (bytes)
22:49
22:53
22:56
23:00
23:03
Time
Details
Name: simpleservice
Namespace: default
Labels: run: simpleservice
Creation Time: 2019-03-05T21:54 UTC
Status: Running
QoS Class: BestEffort
Network
Node: minikube
IP: 172.17.0.7
Containers

Pods


simpleservice - Kubernetes Das
Logs - Kubernetes Dashboard
127.0.0.1:42703/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard:/proxy/#!/log/default/simpleservice/pod?namespace=default
kubernetes
Search
+ CREATE
Logs
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Logs from simpleservice
in simpleservice
0
2019-03-05T09:54:17 INFO This is simple service in version v0.5.0 listening on port 9876 [at line 142]

Pods

- Acceso al servicio publicado por el pod
  - Desde dentro del cluster

```
$ minikube ssh
[cluster]$ curl http://172.17.0.7:9876/info
```

```
{"host": "172.17.0.7:9876", "version":
"0.5.0", "from": "172.17.0.1"}
```

- Borrar el pod

```
$ kubectl delete pods/simpleservice
```

Pods

- Consulta de logs de los contenedores del pod

```
$ kubectl logs <pod_name> <container_name>
```

```
$ kubectl logs simpleservice simpleservice
```

```
2019-03-05T10:21:11 INFO This is simple service in
version v0.5.0 listening on port 9876 [at line 142]
```

Recursos

- Existen muchos tipos de recursos en Kubernetes
- Operaciones sobre recursos
  - Listar

```
$ kubectl get <tipo>
```

  - Describir

```
$ kubectl describe <tipo>/<nombre>
```

  - Borrar

```
$ kubectl delete <tipo>/<nombre>
```

Recursos

- **Comandos útiles**
  - Mostrar varios tipos de recursos en la shell actualizado en tiempo real

Sólo en linux

```
$ watch -n 1 kubectl get pods,services,deployments
```

  - Borrar todos los recursos de un tipo

```
$ kubectl delete <tipo> --all
```

Deployments

- Los deployments son recursos usados para **gestionar los pods**
- **Actualización** de los pods
  - Termina los pods de la versión anterior
  - Crea nuevos pods de la nueva versión
- **Escalabilidad** y **tolerancia** a fallos de los pods
  - Se pueden crear varias **réplicas** de los pods (**ReplicaSet**)
  - Si un pod falla, los otros pueden atender tráfico
  - Varios pods pueden ejecutarse en diferentes nodos para atender más tráfico

Deployments

- Crear un deployment

```
$ kubectl create deployment simpleservice \
--image=mhausenblas/simpleservice:0.5.0
```

- Consultar deployments

```
$ kubectl get deployments
```

- Describir deployment

```
$ kubectl describe deployments/simpleservice
```

- Borrar deployment

```
$ kubectl delete deployments/simpleservice
```

Deployments

- **Escalado de pods**
  - Se indica el nuevo número de réplicas

```
$ kubectl scale --replicas=2 deployments/simpleservice
```

  - Ese cambio implica que se inician nuevos pods (o se detienen los que no son necesarios)

```
$ kubectl get pods
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/simpleservice-847555bfdc-jdsjw      1/1     Running   0          10m
pod/simpleservice-847555bfdc-jxk4j      1/1     Running   0          26s
```

Deployments

- **Actualización de pods**
  - Basta con cambiar el tag de la imagen del deployment

```
$ kubectl create deployment kubernetes-bootcamp \
    --image=jocatalin/kubernetes-bootcamp:v1
```

```
$ kubectl set image deployment/kubernetes-bootcamp \
  kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
```

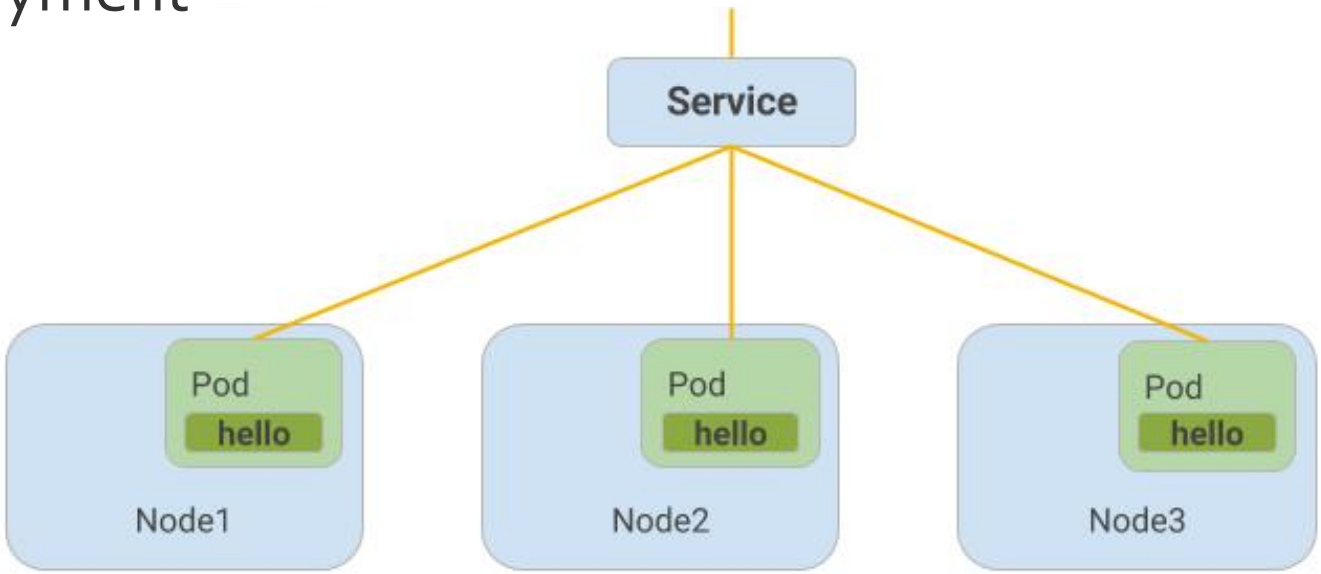
  - Un nuevo pod con la nueva imagen se inicia y se finaliza el pod anterior
  - No funciona cuando se actualiza el tag “latest” porque a ojos de Kubernetes nada ha cambiado

Servicios

- Cada pod dispone de una **IP** a la que se puede acceder desde otro pod o desde el cluster
- Pero los pods pueden detenerse por escalado, por un fallo, por actualizaciones… y en ese caso **cambian de IP**
- Cuando existen **varias réplicas** de un mismo pod, es deseable que la **carga se reparta** entre todas las réplicas disponibles
- Un **servicio** es un recurso Kubernetes que permite el acceso a los pods del deployment con un **nombre lógico**

Servicios

- Un **servicio** crea un **nombre lógico** asociado a los pods de un deployment
- Se crea una **entrada de DNS** con el nombre del servicio que permite el **acceso balanceado** a los **pods** del deployment



Servicios

- **Servicios internos o públicos:**
  - **Interno**
    - El servicio es accesible únicamente desde dentro del cluster
    - Type: **ClusterIP**
  - **Público**
    - El servicio es accesible desde el exterior
    - Type: **NodePort** o **LoadBalancer**

Servicios

- Crear un servicio interno (exponiendo un deployment)

```
$ kubectl expose deployment simpleservice \
--type=ClusterIP --port=9876
```

- Consulta de servicios

```
$ kubectl get services
```

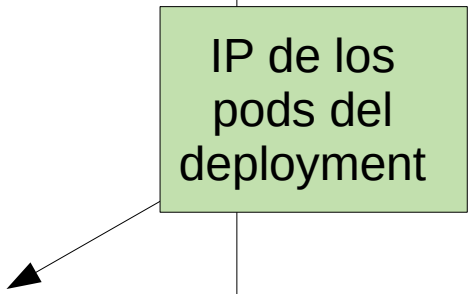
```
NAME                    TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/kubernetes      ClusterIP   10.96.0.1      <none>        443/TCP    3h23m
service/simpleservice   ClusterIP   10.98.156.18   <none>        9876/TCP   40s
```

Servicios

- Información de un servicio concreto

```
$ kubectl describe services/simpleservice
```

```
Name:                   simpleservice
Namespace:              default
Labels:                 app=simpleservice
Annotations:            <none>
Selector:               app=simpleservice
Type:                   ClusterIP
IP:                     10.98.156.18
Port:                   <unset>  9876/TCP
TargetPort:             9876/TCP
Endpoints:              172.17.0.7:9876,172.17.0.9:9876
Session Affinity:       None
Events:                 <none>
```



- Borrado del servicio

```
$ kubectl delete services/simpleservice
```

Servicios

- **Acceso al servicio desde otro pod**
  - Iniciamos un pod interactivo para poder ver su salida desde la consola

```
$ kubectl run --generator=run-pod/v1 \
    -it curl --image=byrnedo/alpine-curl:0.1.7 \
   --command -- /bin/sh
```

  - Dentro de la shell del contenedor, podemos acceder a otros servicios usando su nombre

```
/ # curl http://simpleservice:9876/info
```

```
{"host": "simpleservice:9876", "version": "0.5.0", "from": "172.17.0.12"}
```

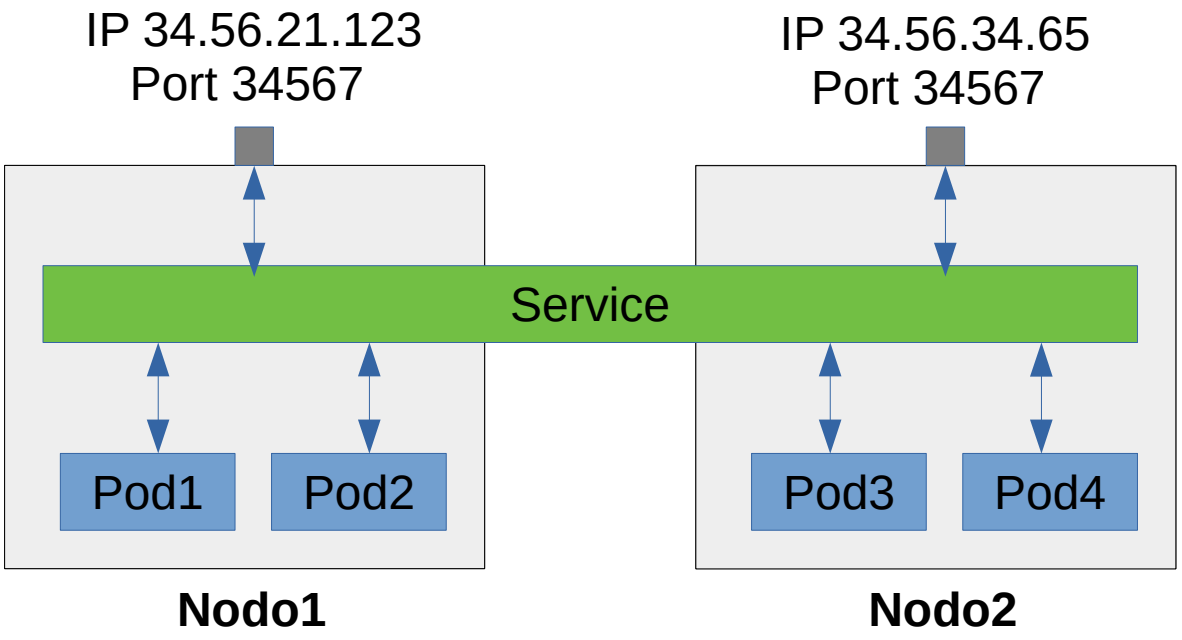
Servicios

- **Servicios públicos**
  - NodePort
    - Expone el servicio en un puerto en todos los nodos del cluster
    - Se usa el mismo puerto en todos los nodos
    - El tráfico se enruta a un pod del deployment, esté en el nodo que esté
    - También se crea automáticamente un servicio ClusterIP interno

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

https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0

Servicio NodePort

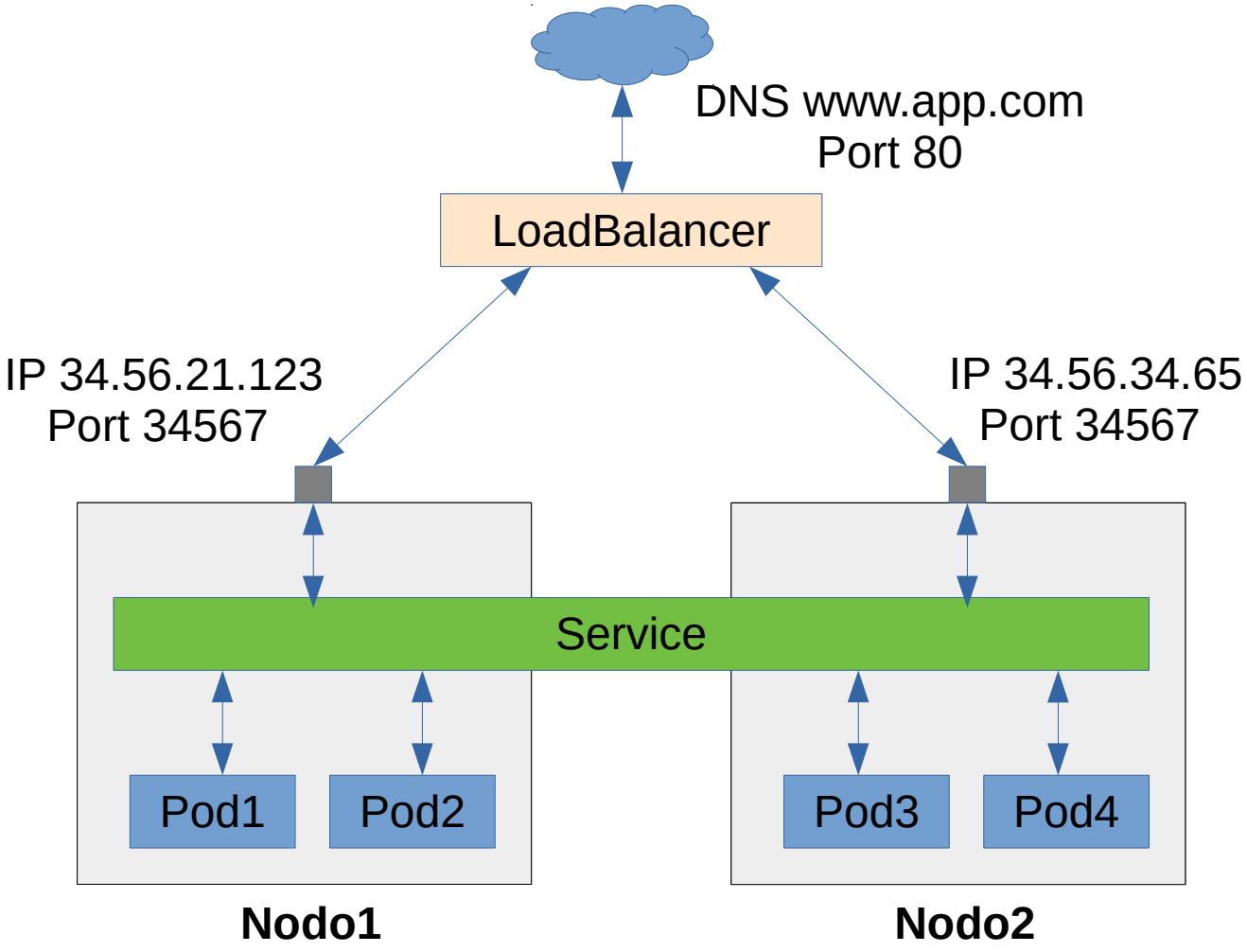


No existe un punto único (IP o nombre dominio) para acceder al servicio

Servicios

- **Servicios públicos**
  - LoadBalancer
    - Crea un balanceador en el proveedor cloud para publicar el servicio con una única IP o nombre DNS
    - Internamente se crea un NodePort y por tanto un ClusterIP
    - En AWS se genera un nombre DNS para el servicio, en Google Cloud se usa IP pública
    - En Minikube no se crea un balanceador y se interpreta como NodePort

Servicio LoadBalancer



Se crea un LoadBalancer por servicio, y eso puede ser costoso

Servicios

- **Inconvenientes de los LoadBalancer**
  - Son costosos
  - No permiten publicar varios servicios http en el mismo dominio y diferentes rutas
- **Recursos Ingress**
  - Una forma avanzada de publicar servicios en Kubernetes
  - Trabajan a nivel http (proxy inverso)
  - Internamente usan NGINX, Traefic, etc.

https://kubernetes.io/docs/concepts/services-networking/ingress/

Servicios

- Servicio público

```
$ kubectl create deployment kubernetes-bootcamp \
    --image=jocatalin/kubernetes-bootcamp:v1
```

```
$ kubectl expose deployment kubernetes-bootcamp \
    --type=LoadBalancer --port=8080
```

```
$ kubectl get services
```

```
NAME                          TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/kubernetes            ClusterIP      10.96.0.1       <none>        443/TCP          4h6m
service/kubernetes-bootcamp   LoadBalancer   10.111.118.99   <pending>     8080:31034/TCP   59s
```

Puerto público

Servicios

- Con minikube se puede abrir el browser para acceder a un servicio público

```
$ minikube service kubernetes-bootcamp
```

- Comandos para descubrir IP de la VM y puerto en el que se ha publicado

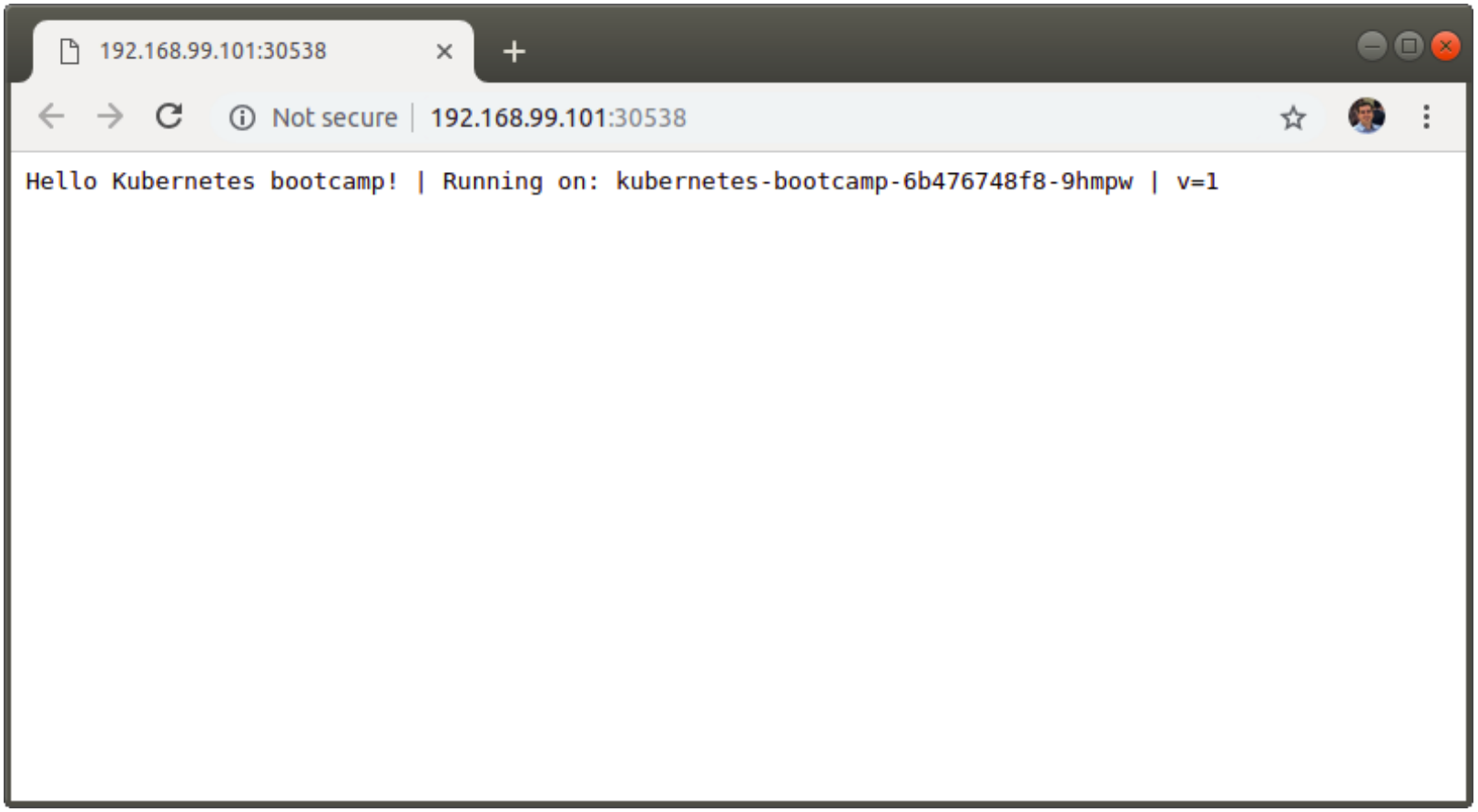
```
$ minikube ip
192.168.99.100

$ kubectl get service kubernetes-bootcamp \
--output='jsonpath={.spec.ports[0].nodePort}'
32041
```

http://192.168.99.100:32041

Servicios

minikube



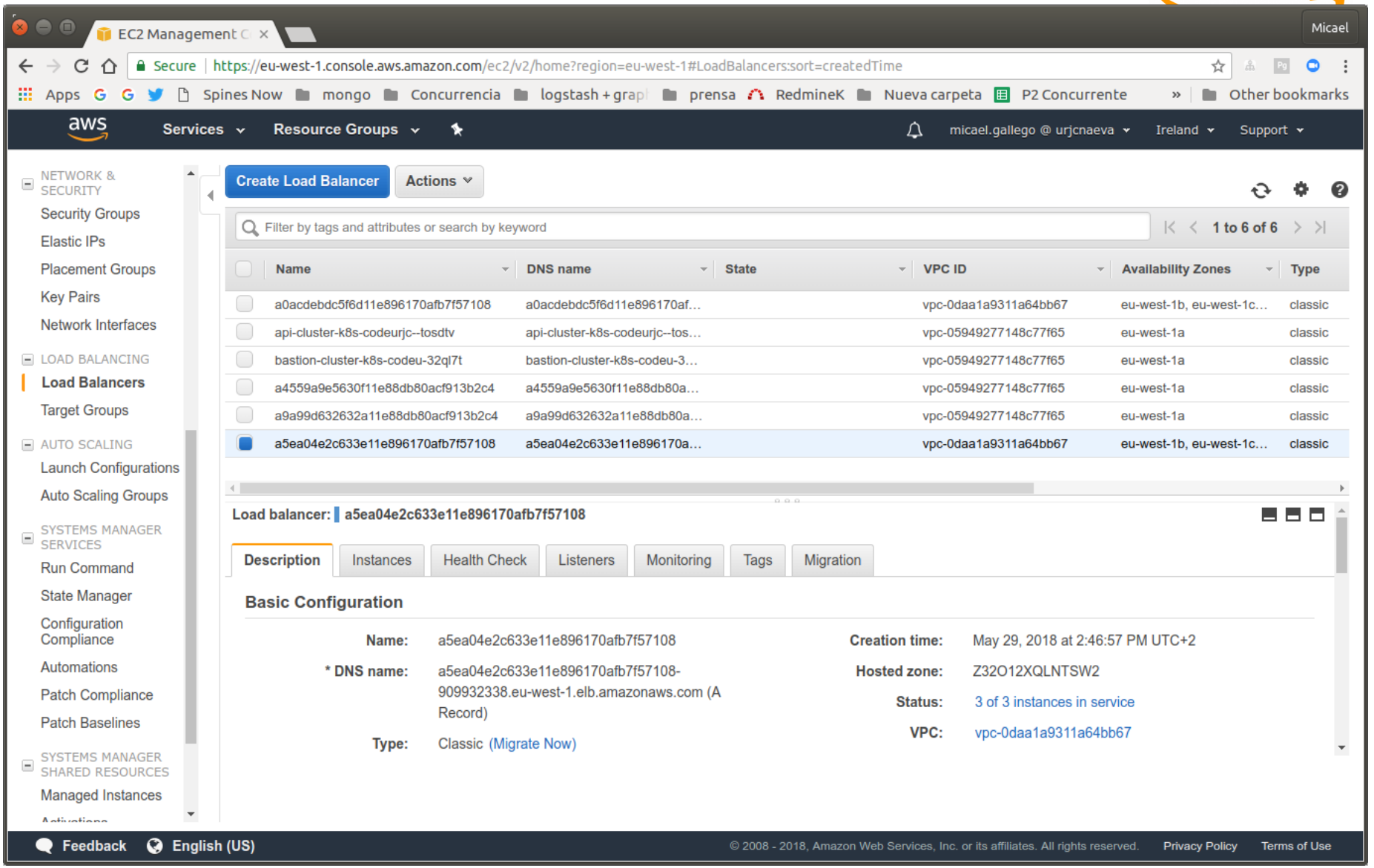
Servicios

- LoadBalancer en AWS

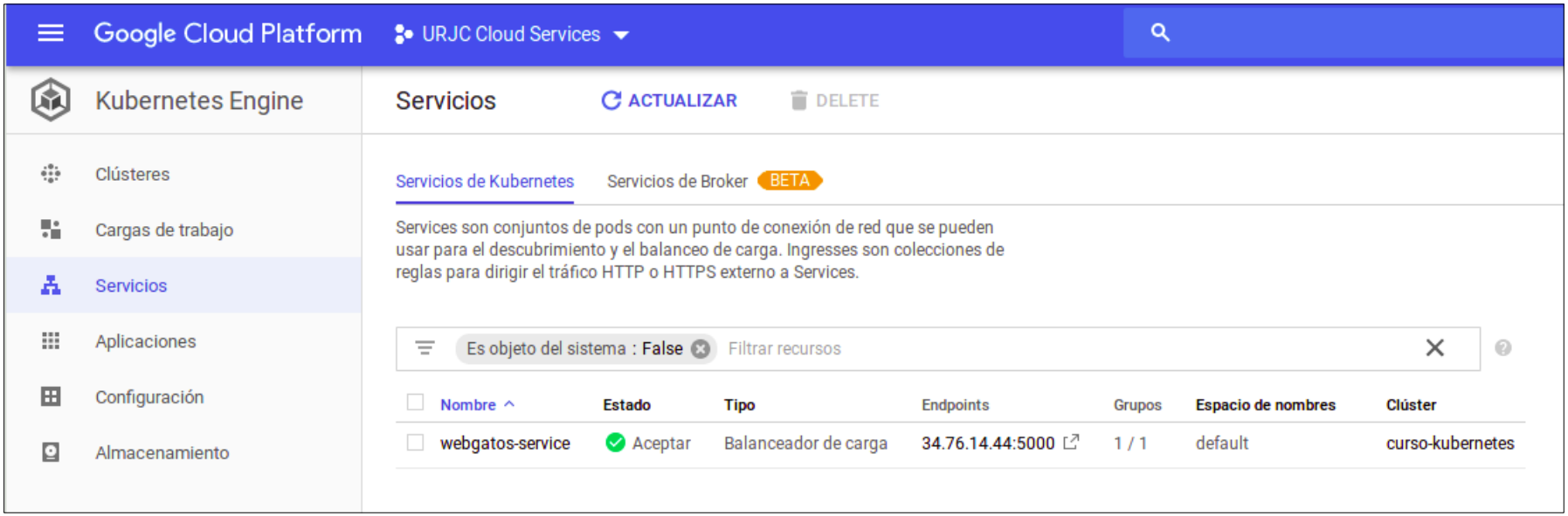
```
$ kubectl describe services/kubernetes-bootcamp
```

```
mica@mica-laptop:~$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   run=kubernetes-bootcamp
Annotations:              <none>
Selector:                 run=kubernetes-bootcamp
Type:                     LoadBalancer
IP:                       100.65.13.99
LoadBalancer Ingress:     a5ea04e2c633e11e896170afb7f57108-909932338.eu-west-1.elb.amazonaws.com
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  32529/TCP
Endpoints:                100.98.88.132:8080
Session Affinity:         None
External Traffic Policy:  Cluster
```

http://a5ea04e2c633e11e896170afb7f57108-909932338.eu-west-1.elb.amazonaws.com:8080/


EC2 Management C
Micael
Secure
https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#LoadBalancers:sort=createdTime
Apps
Spines Now
mongo
Concurrencia
logstash + grapl
prensa
RedmineK
Nueva carpeta
P2 Concurrente
Other bookmarks
aws
Services
Resource Groups
micael.gallego @ urjcnaeva
Ireland
Support
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
LOAD BALANCING
Load Balancers
Target Groups
AUTO SCALING
Launch Configurations
Auto Scaling Groups
SYSTEMS MANAGER SERVICES
Run Command
State Manager
Configuration Compliance
Automations
Patch Compliance
Patch Baselines
SYSTEMS MANAGER SHARED RESOURCES
Managed Instances
Create Load Balancer
Actions
Filter by tags and attributes or search by keyword
1 to 6 of 6
Name
DNS name
State
VPC ID
Availability Zones
Type
a0acdebdc5f6d11e896170afb7f57108 a0acdebdc5f6d11e896170af... vpc-0daa1a9311a64bb67 eu-west-1b, eu-west-1c... classic
api-cluster-k8s-codeurjc--tosdtv api-cluster-k8s-codeurjc--tos... vpc-05949277148c77f65 eu-west-1a classic
bastion-cluster-k8s-codeu-32ql7t bastion-cluster-k8s-codeu-3... vpc-05949277148c77f65 eu-west-1a classic
a4559a9e5630f11e88db80acf913b2c4 a4559a9e5630f11e88db80a... vpc-05949277148c77f65 eu-west-1a classic
a9a99d632632a11e88db80acf913b2c4 a9a99d632632a11e88db80a... vpc-05949277148c77f65 eu-west-1a classic
a5ea04e2c633e11e896170afb7f57108 a5ea04e2c633e11e896170a... vpc-0daa1a9311a64bb67 eu-west-1b, eu-west-1c... classic
Load balancer: a5ea04e2c633e11e896170afb7f57108
Description
Instances
Health Check
Listeners
Monitoring
Tags
Migration
Basic Configuration
Name: a5ea04e2c633e11e896170afb7f57108
* DNS name: a5ea04e2c633e11e896170afb7f57108-909932338.eu-west-1.elb.amazonaws.com (A Record)
Type: Classic (Migrate Now)
Creation time: May 29, 2018 at 2:46:57 PM UTC+2
Hosted zone: Z32O12XQLNTSW2
Status: 3 of 3 instances in service
VPC: vpc-0daa1a9311a64bb67
Feedback
English (US)
© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Servicios


Google Cloud Platform
URJC Cloud Services
Kubernetes Engine
Clústeres
Cargas de trabajo
Servicios
Aplicaciones
Configuración
Almacenamiento
Servicios
ACTUALIZAR
DELETE
Servicios de Kubernetes
Servicios de Broker
BETA
Services son conjuntos de pods con un punto de conexión de red que se pueden usar para el descubrimiento y el balanceo de carga. Ingresses son colecciones de reglas para dirigir el tráfico HTTP o HTTPS externo a Services.
Es objeto del sistema : False
Filtrar recursos
Nombre
Estado
Tipo
Endpoints
Grupos
Espacio de nombres
Clúster
webgatos-service
Aceptar
Balanceador de carga
34.76.14.44:5000
1 / 1
default
curso-kubernetes

Ejercicio 1

- Despliega en Kubernetes la aplicación web "webgatos"
  - Imagen: codeurjc/webgatos:v1
  - Puerto: 5000
  - Código de la aplicación:

https://github.com/MasterCloudApps/3.2.Contenedores-y-orquestadores/tree/master/ejemplo1/web-python

- Publica la web (creando un servicio)
- Verifica que puedes acceder a ella con un navegador web

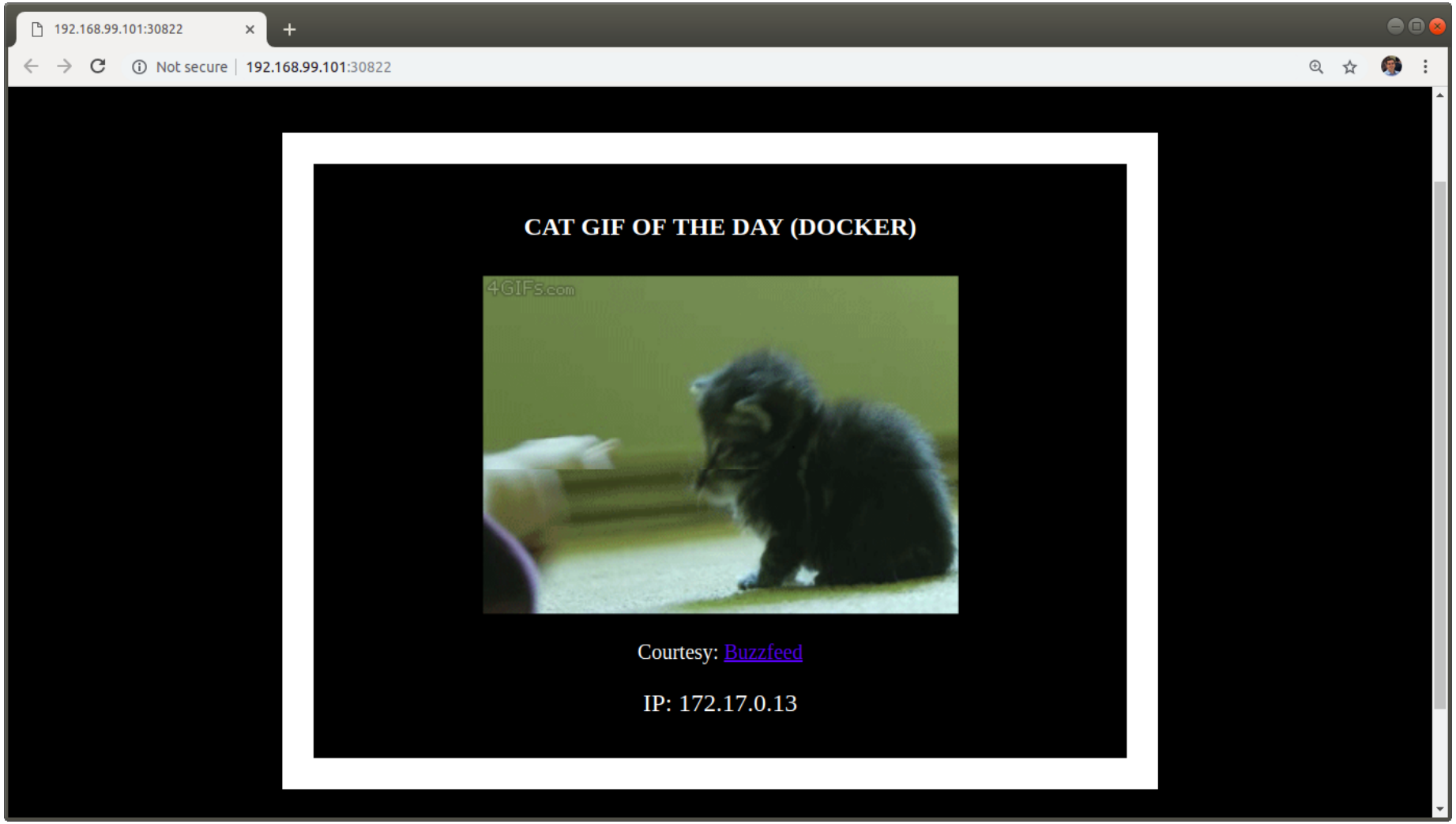
Solución

```
$ kubectl create deployment webgatos \
  --image=codeurjc/webgatos:v1
```

```
$ kubectl expose deployment webgatos \
  --type=NodePort --port=5000
```

```
$ minikube service webgatos
```

Solución


192.168.99.101:30822
Not secure | 192.168.99.101:30822
CAT GIF OF THE DAY (DOCKER)
4GIFs.com
Courtesy: Buzzfeed
IP: 172.17.0.13

Ejercicio 2

- Escala el deployment de webgatos para que tenga dos réplicas
- Comprueba que si se crean esas réplicas
- Verifica que al acceder a la web cada vez se obtiene una IP diferente porque se accede a un contenedor diferente

Solución

```
$ kubectl scale deployments/webgatos \
  --replicas=2
```

```
$ kubectl get pods
```

Labels

- Cada recurso kubernetes puede tener un número arbitrario de **etiquetas** (nombre=valor)

```
$ kubectl label deployment kubernetes-bootcamp version=v1
```

- Se usan para filtrar en los comandos (listados, borrado, ...)

```
$ kubectl get pods -l run=kubernetes-bootcamp
$ kubectl get services -l run=kubernetes-bootcamp
$ kubectl delete service -l run=kubernetes-bootcamp
```

Namespaces

- La mayoría de los recusos Kubernetes se asocian a un **namespace**
  - Evita colisiones de nombres
  - Facilita la gestión de recursos por diferentes equipos / usuarios

```
$ kubectl get namespaces
```

```
NAME          STATUS    AGE
default       Active    1d
kube-system   Active    1d
kube-public   Active    1d
```

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Namespaces

- Especificar un namespace en un comando

```
$ kubectl --namespace=<namespace-name> get pods
```

- Establecer el namespace para todos los comandos

```
kubectl config set-context $(kubectl config current-context) \
  --namespace=<namespace-name>
```

```
$ kubectl config view | grep namespace:
```

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Specs

- Crear los recursos Kubernetes por **línea de comandos** es **engorroso**, propenso a **errores** y **limitado**
- No permite tener la **configuración** de los recursos **bajo control** de versiones
- Kubectl es considerado el **ssh del cloud**

Specs

- En general se utilizan ficheros de descripción de los recursos llamados ***manifests*** en formato **YAML**

https://www.mirantis.com/blog/introduction-to-yaml-creating-a-kubernetes-deployment/

https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.10

- Los comandos a partir de ahora se asume que se ejecutan desde la raíz de este repositorio, que contiene manifests de ejemplo

https://github.com/MasterCloudApps/3.2.Contenedores-y-orquestadores

Specs

```
$ kubectl create -f ejemplo1/webgatos-deployment.yaml
```

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webgatos-deploy
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: webgatos
  replicas: 1 # tells deployment to run 1 pods matching the template
  template: # create pods using pod definition in this template
    metadata:
      labels:
        app: webgatos
    spec:
      containers:
      - name: webgatos
        image: codeurjc/webgatos:v1
        ports:
        - containerPort: 5000
```

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Specs

```
$ kubectl create -f ejemplo1/webgatos-service.yaml
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: webgatos-service
  labels:
    app: webgatos
spec:
  ports:
    - port: 5000
      protocol: TCP
      name: webgatos-port
  selector:
    app: webgatos
    type: NodePort
```

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

Specs

- **Múltiples recursos en un fichero**
  - Podemos incluir varios objetos Kubernetes en un fichero
  - Se copia el contenido completo de cada fichero (como cada fichero comienza con tres guiones, sirve de separador)

```
$ kubectl create -f ejemplo1/webgatos.yaml
```

Specs

- **Aplicaciones con varios servicios**
  - Kubernetes de forma nativa no tiene el concepto de aplicaciones formadas por varios servicios
  - Se gestionan creando todos los objetos de la misma aplicación (deployment, servicios…), posiblemente asociados con el mismo valor en la label app para facilitar su gestión
  - La herramienta **Helm** ofrece una gestión de aplicaciones, su actualización y un repositorio

Specs

- **Configuración de pods con variables de entorno**
  - Se pueden poner valores directamente en el fichero de deployment o usar **ConfigMaps**

```
...
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: DEMO_GREETING
      value: "Hello from the environment"
    - name: DEMO_FAREWELL
      value: "Such a sweet sorrow"
```

https://kubernetes.io/docs/tasks/inject-data-application/define-environment-variable-container/

https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/

Ejercicio 3

- **Despliega una aplicación de web de anuncios con base de datos en Kubernetes**

- Aplicación Web
  - Imagen: codeurjc/java-webapp-bbdd:v2
  - Puerto: 8080
  - Variables de entorno:
    - MYSQL_ROOT_PASSWORD = pass
    - MYSQL_DATABASE = test

- Base de datos:
  - Imagen: mysql:5.6
  - Puerto: 3306
  - Nombre del servicio: db
  - Variables de entorno
    - MYSQL_ROOT_PASSWORD=pass
    - MYSQL_DATABASE=test

https://github.com/codeurjc/curso-docker/tree/master/java-webapp-bbdd-ejer

Solución

```
$ kubectl create -f ejercicio3/db.yaml
```

```
$ kubectl create -f ejercicio3/webapp.yaml
```

```
$ minikube service webapp
```

Ingress

- Una forma más avanzada de publicar **servicios**
- Varias aplicaciones http pueden **compartir** el mismo **nombre de dominio y puerto** y se pueden publicar en rutas diferentes
- Ingress es un recurso Kubernetes que puede tener diferentes "**implementaciones**" (llamadas controladores)
- Es esencialmente un **proxy http inverso**, y una de sus implementaciones más usadas está basada en **NGINX**

https://kubernetes.io/docs/concepts/services-networking/ingress/

Ingress

- Activar el ingress controller en minikube

https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/

```
$ minikube addons enable ingress
```

- Activar el ingress controller en Docker Desktop

https://kubernetes.github.io/ingress-nginx/deploy/#docker-for-mac

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/mandatory.yaml

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/nginx-0.30.0/deploy/static/provider/cloud-generic.yaml
```

Ingress

- **Uso de ingress**
  - Desplegamos la web de gatos (deployment y service)

```
$ kubectl apply -f ejemplo1/webgatos.yaml
```

  - Desplegamos la web de anuncios (deployment y service)

```
$ kubectl apply -f ejercicio3/db.yaml
```

```
$ kubectl apply -f ejercicio3/webapp.yaml
```

Ingress

- **Uso de ingress**
  - Se despliegan los servicios http que se quieren hacer disponibles a través del ingress con un **Service** de tipo **ClusterIP**
    - Por ejemplo la web de gatos (un servicio python)

```
$ kubectl apply -f ejemplo1/webgatos.yaml
```

    - Y la web de anuncios (un servicio Java y una BBDD)

```
$ kubectl apply -f ejercicio3/db.yaml
```

```
$ kubectl apply -f ejercicio3/webapp.yaml
```

Ingress

- **Uso de ingress**
  - Se configuran las **rutas en la URL** en las que esos **Service** estarán disponibles
    - **Proveedor cloud:** Se usa un balanceador de carga y un nombre DNS asociado
    - **Minikube / Docker Desktop:** Simulamos un nombre de dominio público creando una entrada en el /etc/hosts de la máquina

Ingress

ingress/ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: codeurjc-ingress
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: "/"
spec:
  rules:
  - host: www.sampledomain.com
    http:
      paths:
      - path: /anuncios/
        backend:
          serviceName: webapp
          servicePort: 8080
      - path: /gatos/
        backend:
          serviceName: webgatos-service
          servicePort: 5000
```

```
$ kubectl apply -f ingress/ingress.yaml
```

Ingress

- **Asociar nombre dominio a IP en Linux**
  - Obtener la IP de la VM

```
$ export MINIKUBE_IP=$(minikube ip)
```

  - Añadir una entrada al /etc/hosts

```
$ echo $MINIKUBE_IP www.sampledomain.com \
| sudo tee --append /etc/hosts >/dev/null
```

Ingress

- **Asociar nombre dominio a IP en Mac**
  - **Minikube**
    - Obtener la IP de la VM
      - $ minikube ip
    - Editar el fichero /private/etc/hosts
      - $ sudo nano /private/etc/hosts
    - Añadir la línea
      - 192.168.99.100 www.sampledomain.com
  - **Docker Desktop**
    - Editar el fichero
      - $ sudo nano /private/etc/hosts
    - y añadir la línea
      - 127.0.0.1 www.sampledomain.com

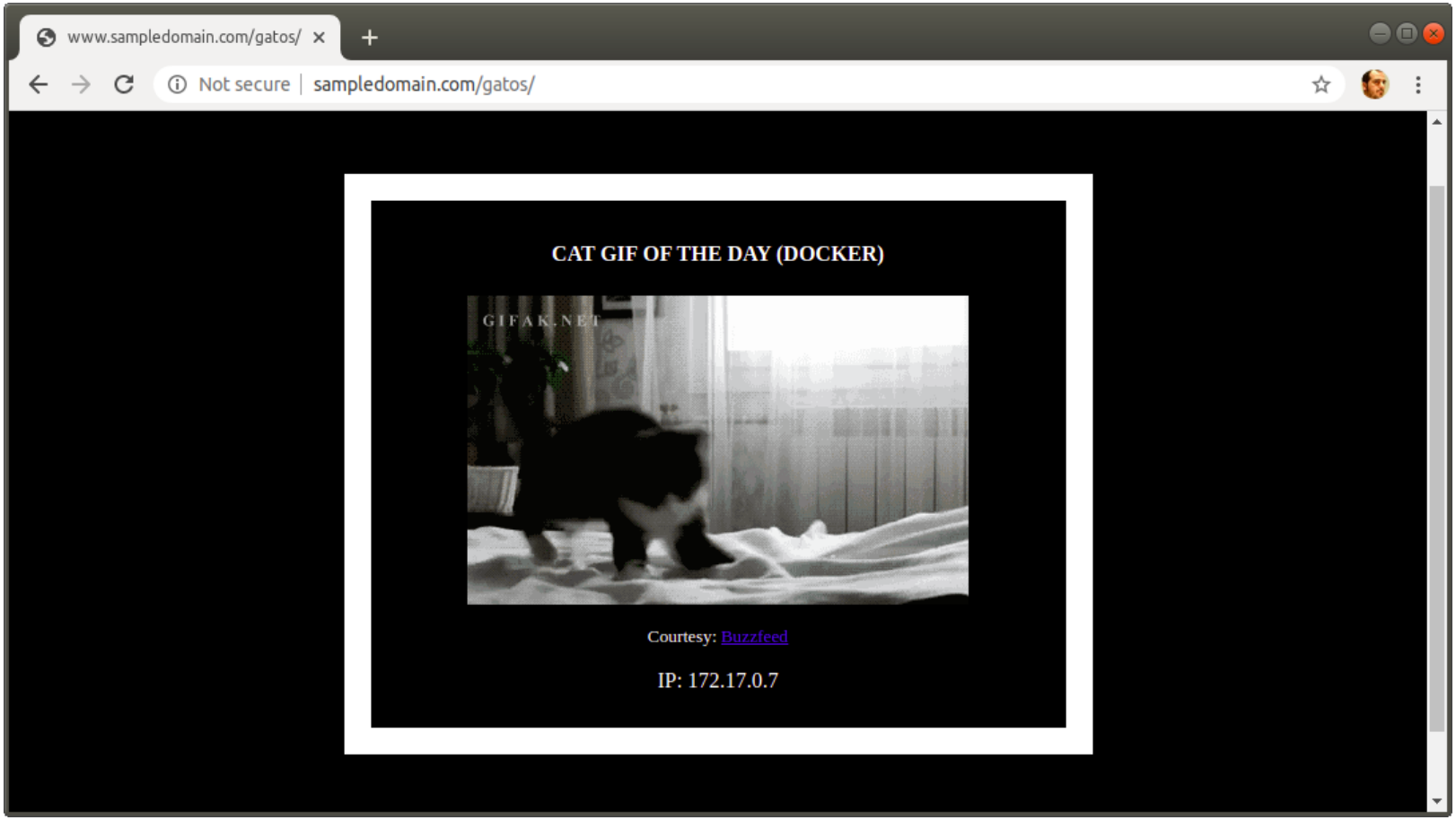
Ingress

- **Asociar nombre dominio a IP en Windows**
  - **Minikube**
    - 1. Presiona la tecla de Windows.
    - 2. Escribe Notepad en campo de búsqueda.
    - 3. En los resultados haz click derecho sobre el icono del Notepad y selecciona ejecutar como administrador.
    - 4. Desde el Notepad abre el fichero: c:\Windows\System32\Drivers\etc\hosts
    - 5. Añade una línea como esta usando la IP de la VM Minikube
      - 192.168.99.100 www.sampledomain.com
    - 6. Haz click en Fichero > Guardar para guardar los cambios.
    - 7. Cierra el Notepad.

Ingress

- **Asociar nombre dominio a IP en Windows**
  - **Docker Desktop**
    - 1. Presiona la tecla de Windows.
    - 2. Escribe Notepad en campo de búsqueda.
    - 3. En los resultados haz click derecho sobre el icono del Notepad y selecciona ejecutar como administrador.
    - 4. Desde el Notepad abre el fichero: c:\Windows\System32\Drivers\etc\hosts
    - 5. Añade una línea como esta usando la IP de la VM Minikube
      - 127.0.0.1 www.sampledomain.com
    - 6. Haz click en Fichero > Guardar para guardar los cambios.
    - 7. Cierra el Notepad.

Ingress



http://www.sampledomain.com/gatos/

Ingress

http://www.sampledomain.com/anuncios/

Volúmenes

- Los volúmenes de Kubernetes son un concepto parecido al de Docker, pero más **potente**
- Si un pod termina de forma abrupta, si sus **datos** están en un volumen, se **mantienen** en el siguiente **reinicio**
- Para que dos contenedores del mismo pod compartan **información en disco** se usan los volúmenes
- Existen muchos tipos de volúmenes con características diferentes: local, awsElasticBlockStore, configMap, gitRepo, glusterfs, hostPath, nfs…

https://kubernetes.io/docs/concepts/storage/volumes/

Volúmenes

- **emptyDir**
  - Se crea vacío
  - Los datos se mantienen siempre que el pod se ejecute en el mismo nodo
  - Si se borra el pod (delete), los datos se pierden
  - Si el contenedor se para de forma abrupta (crash), se **reinicia en la misma máquina** y los **datos no se pierden**
  - Se usa cuando se necesitan **ficheros temporales** en disco o para **comunicar contenedores del mismo pod**

https://kubernetes.io/docs/concepts/storage/volumes/#types-of-volumes

Volúmenes

- **hostPath**
  - Monta una ruta de disco
  - Sirve para acceder a **carpetas específicas del nodo**.
  - Hay rutas en todos los nodos (/var/lib/docker) o se puede forzar que ciertos pods se ejecuten en ciertos nodos (con taints)
- **local**
  - Sirve para guardar datos en el disco del nodo (carpeta, disco o partición)
  - Si el nodo se llena o se desconecta del cluster, los pods que se quieran conectar a ese volumen, no podrán ejecutarse.

https://kubernetes.io/docs/concepts/storage/volumes/#types-of-volumes

Volúmenes

- **glusterFS**
  - Monta un volumen del sistema de ficheros en red open source GlusterFS
  - Se pueden montar varios contenedores en modo escritura de forma simultánea
  - La información no se pierde, es persistente

https://www.gluster.org/

https://github.com/kubernetes/examples/tree/master/staging/volumes/glusterfs

Volúmenes

- **Persistence Volumes en AWS**
  - **awsElasticBlockStore**
    - Un volumen sólo puede estar conectado a un único nodo
    - Cuando un pod se despliega en otro nodo, el EBS se monta ese nuevo nodo
    - Un Persistence Volumen EBS creado en una zona de disponibilidad, no se puede montar en un nodo en otra zona
  - **efs-provisioner**
    - Usa Elastic FileSystem Service de AWS (Un NFS as a service)
    - Está en incubator

https://github.com/kubernetes-incubator/external-storage/tree/master/aws/efs

Volúmenes

- **Volúmenes en el pod**
  - Algunos tipos de volúmenes básicos se pueden configurar en el pod (o en el Deployment)
  - Por ejemplo: **emptyDir**, **hostPath**…
  - El Pod al arrancar podrá escribir en esa ruta y los ficheros tendrán el ciclo de vida determinado por el tipo de volumen

Volúmenes

- Volúmenes en el pod

volumenes/emptyDir.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: codeurjc/test-webserver:v1
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Volúmenes

- Volúmenes en el pod

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # directory location on host
      path: /data
      # this field is optional
      type: Directory
```

Volúmenes

- **PersistentVolume**
  - Algunos tipos de volúmenes se pueden crear de forma independiente a los pods con el recurso **PersistentVolume**
  - Para que un pod pueda usar un PersistentVolume, lo solicita con un recurso de tipo "reclamo de volumen de persistencia" **(PersistenceVolumeClaim)**
  - Ese reclamo se atiende con un PersistenceVolume:
    - Creado previamente de forma **explícita por el admin**
    - Creado de forma **dinámica** cuando se solicita. Al solicitarlo se puede especificar un tipo de volúmen (llamado **StorageClass**)

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Volúmenes

- **Uso de PersistentVolume**
  - 1) Creamos un fichero **index.html** en el nodo
  - 2) Creamos un **PersistentVolume** apuntando a la carpeta del fichero (**hostPath**)
  - 3) Creamos un **PersistentVolumeClaim** para que se asocie al **PersistentVolume** previo
  - 4) Definimos un **Deployment** cuyo pod defina un volumen basado en el claim previo y lo publicamos

https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

Volúmenes

- **1) Creamos el fichero index.html**
  - Conectamos al nodo de minikube
    - $ minikube ssh
  - Creamos la carpeta **/mnt/data**
    - $ sudo mkdir /mnt/data
  - Creamos un fichero index.html
    - $ echo 'Hello from Kubernetes storage' | \
      sudo tee /mnt/data/index.html

https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

Volúmenes

- **2) Creamos el PersistentVolume**
  - En desarrollo, podemos usar una carpeta del nodo como volumen persistente (**hostPath**)
  - En producción, se usan los servicios de datos persistentes como AWS EBS, Azure Persistent Disk, NFS…

`https://k8s.io/docs/tasks/configure-pod-container/task-pv-volume.yaml`

Volúmenes

- 2) Creamos el PersistentVolume

```
$ kubectl apply -f volumenes/10g-volume.yaml
```

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: 10g-volume
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

Volúmenes

- 2) Creamos el PersistentVolume

```
$ kubectl get pv
```

```
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
10g-volume    10Gi       RWO            Retain           Available           manual                  9s
```

```
$ kubectl describe pv/10g-volume
```

```
Name:            10g-volume
Labels:          <none>
...
StorageClass:    manual
Status:          Bound
Claim:           default/pvc-3g
Reclaim Policy:  Retain
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        10Gi
Node Affinity:   <none>
Message:
Source:
    Type:          HostPath (bare host directory volume)
    Path:          /mnt/data
    HostPathType:
Events:            <none>
```

Volúmenes

- **3) Creamos un PersistentVolumeClaim**
  - Para que un Pod pueda usar un PV, lo tiene que "reclamar"

```
$ kubectl apply -f volumenes/pvc-3g.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-3g
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Se vincula el PVC con el PV creado porque tienen el mismo storageClassName

Volúmenes

- **3) Creamos un PersistentVolumeClaim**
  - Al crear un PVC, si existe un volumen con el StorageClass solicitado, se vinculan entre sí (Bound)

```
$ kubectl get pv
```

```
NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY
   STATUS   CLAIM                      STORAGECLASS   REASON   AGE
persistentvolume/10g-volume   10Gi       RWO            Retain
   Bound    default/pvc-3g    manual                        4m49s
```

```
$ kubectl get pvc
```

```
NAME                                    STATUS   VOLUME        CAPACITY   ACCESS
   MODES   STORAGECLASS   AGE
persistentvolumeclaim/pvc-3g        Bound    10g-volume   10Gi       RWO
           manual         9s
```

Volúmenes

```
$ kubectl create -f volumenes/webserver.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webserver
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: webserver
  replicas: 1
  template:
    metadata:
      labels:
        app: webserver
    spec:
      containers:
        - name: webserver
          image: nginx:1.15
          ports:
            - containerPort: 80
              name: "http-server"
          volumeMounts:
            - mountPath: "/usr/share/nginx/html"
              name: webserver-storage
      volumes:
        - name: webserver-storage
          persistentVolumeClaim:
            claimName: pvc-3g
```

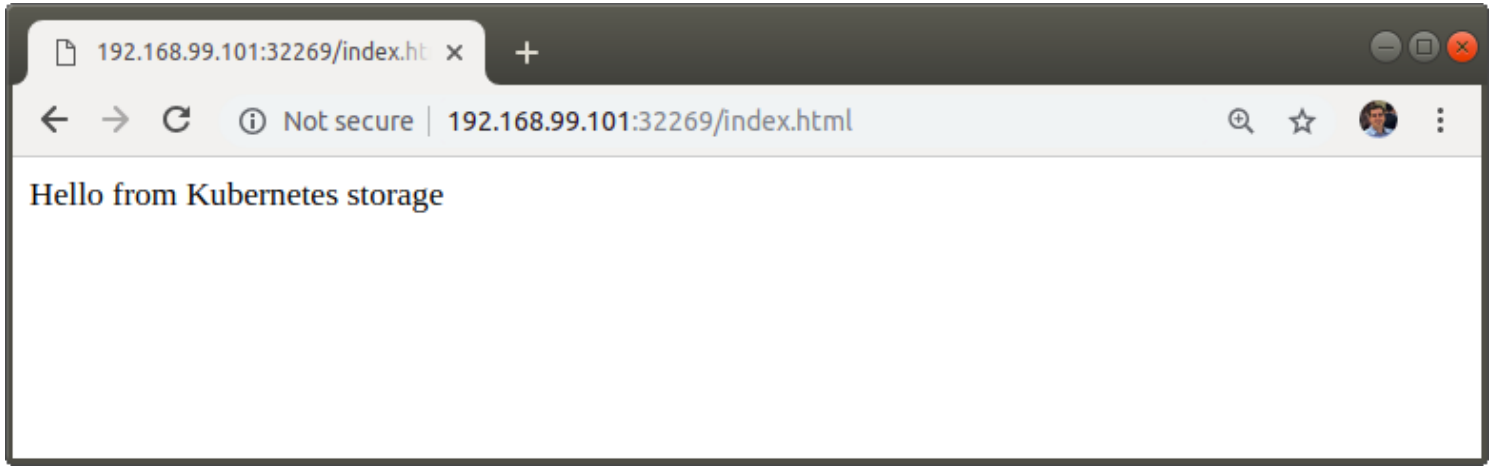
```
apiVersion: v1
kind: Service
metadata:
  name: webserver
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 80
      name: web-port
  selector:
    app: webserver
  type: NodePort
```

Volúmenes

- **4) Definimos un Deployment y lo publicamos**
  - Accedemos al servicio con minikube

```
$ minikube service webserver
```

  - Deberiamos poder ver el mensaje en el navegador



Volúmenes

- **StorageClass**
  - Definen tipos de almacenamientos disponibles en el cluster (con mayor durabilidad, política de backup, etc)
  - En minikube existe un StorageClass basado en hostPath
  - Un cluster en un proveedor tiene StorageClass con un sistema de persistencia por defecto
  - **Kops en AWS** se configura automáticamente un StorageClass usando EBS (tipo **awsElasticBlockStore**)

https://kubernetes.io/docs/concepts/storage/storage-classes/

Ejercicio 4

- **Despliega una web con BBDD persistente**
  - Guarda los datos de la BBDD del Ejercicio 3 en un volumen persistente
  - Podemos usar el mismo PersistenceVolume creado previamente (/mnt/data)
  - La ruta en la que MySQL guarda los datos (mountPath) es: /var/lib/mysql

Solución

```
$ kubectl create -f ejercicio4/mysql-pvc.yaml
```

```
$ kubectl create -f ejercicio4/java-webapp-db.yaml
```

```
$ minikube service java-webapp-db
```

BBDD en K8s

- **Bases de datos replicadas y tolerantes a fallos en Kubernetes**
  - Un pod puede tener un **volumen persistente**
  - Pero **un solo pod con una BBDD** y un único volumen no escala ni es tolerante a fallos
  - La gestión de réplicas de un **Deployment** no se pueden usar para BBDD: están diseñados para pods **stateless**, **efímeros**, que se pueden eliminar en cualquier momento

BBDD en K8s

- **Bases de datos replicadas y tolerantes a fallos en Kubernetes**
  - Los recursos **StatefulSet** proporcionan un mecanismo similar a los Deployment pero con características específicas para contenedores con estado:
    - Identificadores de red estables y únicos por pod
    - Almacenamiento persistente estable por pod
    - Despliegue y escalado ordenado de pods
    - Terminado y borrado ordenado de pods

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

BBDD en K8s

- **Bases de datos replicadas y tolerantes a fallos en Kubernetes**
  - Desplegar un "cluster" de instancias de una BBDD en Kubernetes en un StatefulSet no es sencillo
  - Requiere un conocimiento muy profundo del funcionamiento de la BBDD concreta y del funcionamiento de los StatefulSets

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

BBDD en K8s

- **MySQL escalable y tolerante a fallos**
  - Ejemplo 1 (Dic2017)
    - Un pod maestro y dos esclavos
    - Utiliza la herramienta **xtrabackup** para sincronización entre instancias de MySQL
    - La definición del recurso StatefulSet es bastante **compleja**
    - Ejemplo ofrecido por **Rancher**

https://rancher.com/running-highly-available-wordpress-mysql-kubernetes/

BBDD en K8s

- **MySQL escalable y tolerante a fallos**
  - Oracle presentó una charla en la **Kubeconf Dic2017** presentando cómo instalar MySQL en Kubernetes
  - Es una charla muy completa (aunque antigua)
  - Presenta múltiples enfoques y alternativas

https://dyn.com/blog/mysql-on-kubernetes/

https://schd.ws/hosted_files/kccncna17/4d/MySQL%20on%20Kubernetes.pdf

BBDD en K8s

- **MySQL escalable y tolerante a fallos**
  - MySQL Operator (Oracle)
    - Es un **componente que se instala en el cluster** y coordina el despliegue y el descubrimiento de pods
    - Permite instalar un **MySQL replicado, tolerante a fallos** y con backups de forma muy **sencilla**
    - Está en alpha y lleva desde **Mayo 2019 sin commits**

https://github.com/oracle/mysql-operator

BBDD en K8s

- **Kubernetes Operators**
  - Son **plugins** de Kubernetes que ofrecen funcionalidad de **alto nivel**
  - Pueden añadir un **tipo de recurso** nuevo a la API Kubernetes (como los pods, deployments…) pero definido por el creador del plugin llamados **Custom Resource Definition** (CRD)
  - Los ***operators*** son los ***controllers*** de los CRD que "**aplican**" el estado definido en los CRD al cluster

https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/
https://kubernetes.io/docs/concepts/extend-kubernetes/operator/

BBDD en K8s

- **PostgreSQL escalable y tolerante a fallos**
  - Crunchy Data PostgreSQL Operator
    - Bastante actualizado
    - Proporciona:
      - High-Availability
      - Disaster Recovery
      - Monitoring
      - PostgreSQL User Management
      - Upgrade Management
      - Advanced Replication Support
      - Clone
      - Scheduled Backups
      - Backup to S3

https://github.com/CrunchyData/postgres-operator

BBDD en K8s

- **Vitess**
  - Es un sistema de clusterización para el escalado horizontal de MySQL y MariaDB
  - Está preparado para ser desplegado en **Kubernetes**

https://vitess.io/

BBDD en K8s

- ¿Desplegar una base de datos en Kubernetes?

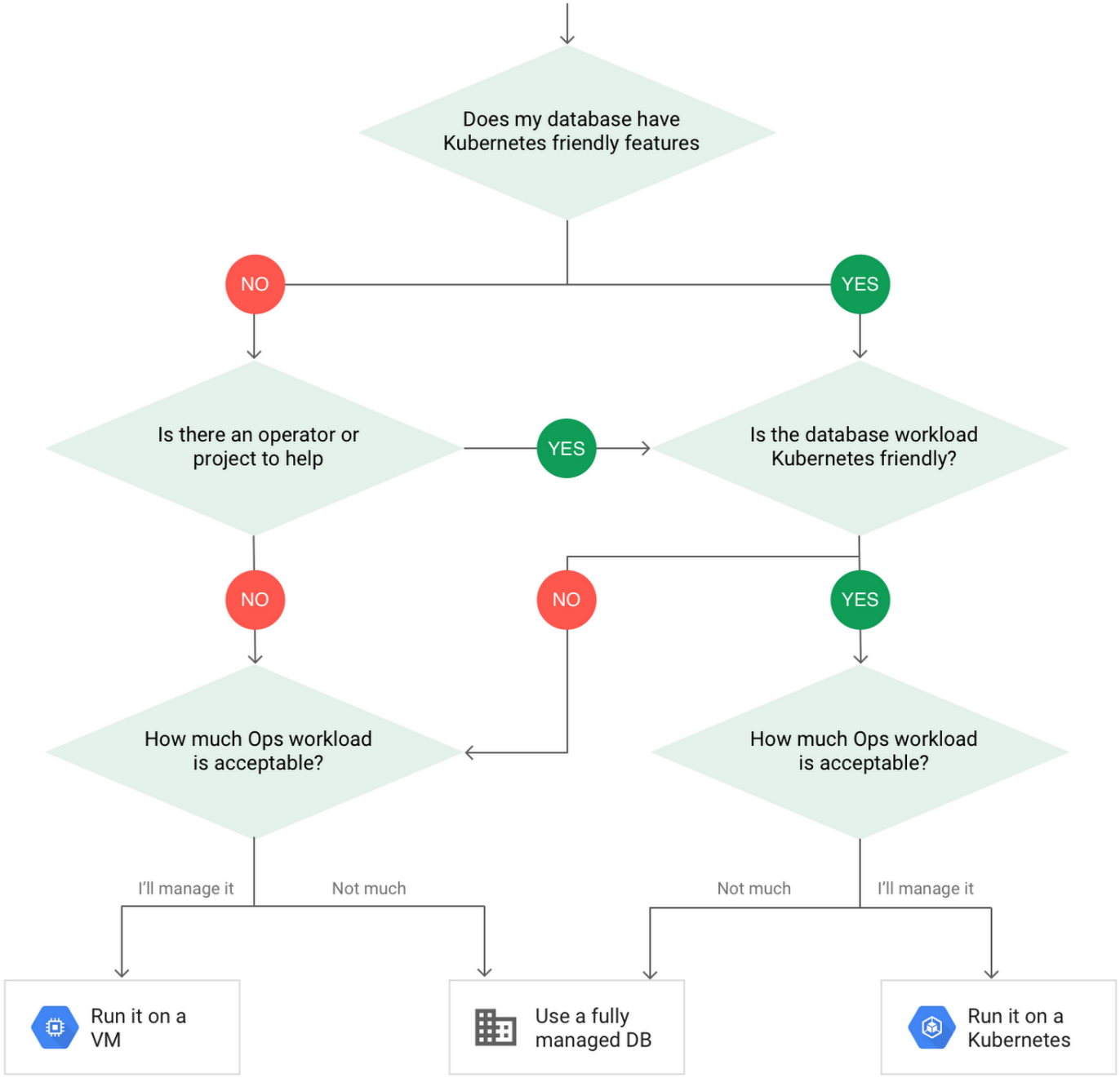
Julio del 2019 en GCP

https://cloud.google.com/blog/products/databases/to-run-or-not-to-run-a-database-on-kubernetes-what-to-consider

BBDD en K8s

- **¿Desplegar una base de datos en Kubernetes?**
  - Alternativas para BBDD
    - Base de datos gestionada por el proveedor
    - Instalación en una VM
    - Ejecución en Kubernetes

https://cloud.google.com/blog/products/databases/to-run-or-not-to-run-a-database-on-kubernetes-what-to-consider


Does my database have Kubernetes friendly features
NO
YES
Is there an operator or project to help
YES
Is the database workload Kubernetes friendly?
NO
NO
YES
How much Ops workload is acceptable?
How much Ops workload is acceptable?
I'll manage it
Not much
Not much
I'll manage it
Run it on a VM
Use a fully managed DB
Run it on a Kubernetes