

Consultoría Formación

Cloud Computing
Distributed Systems
Web Technologies
Extreme Programming
Testing / Git / Jenkins
Software Architectures
Concurrent Programming

Master en Desarrollo y Despliegue de Aplicaciones en la Nube

Online / Clases en directo

Septiembre 2019



kubernetes



Docker



Los **contenedores** permiten empaquetar, distribuir y ejecutar servicios de red con un formato **estándar** con todas las **dependencias** incluidas

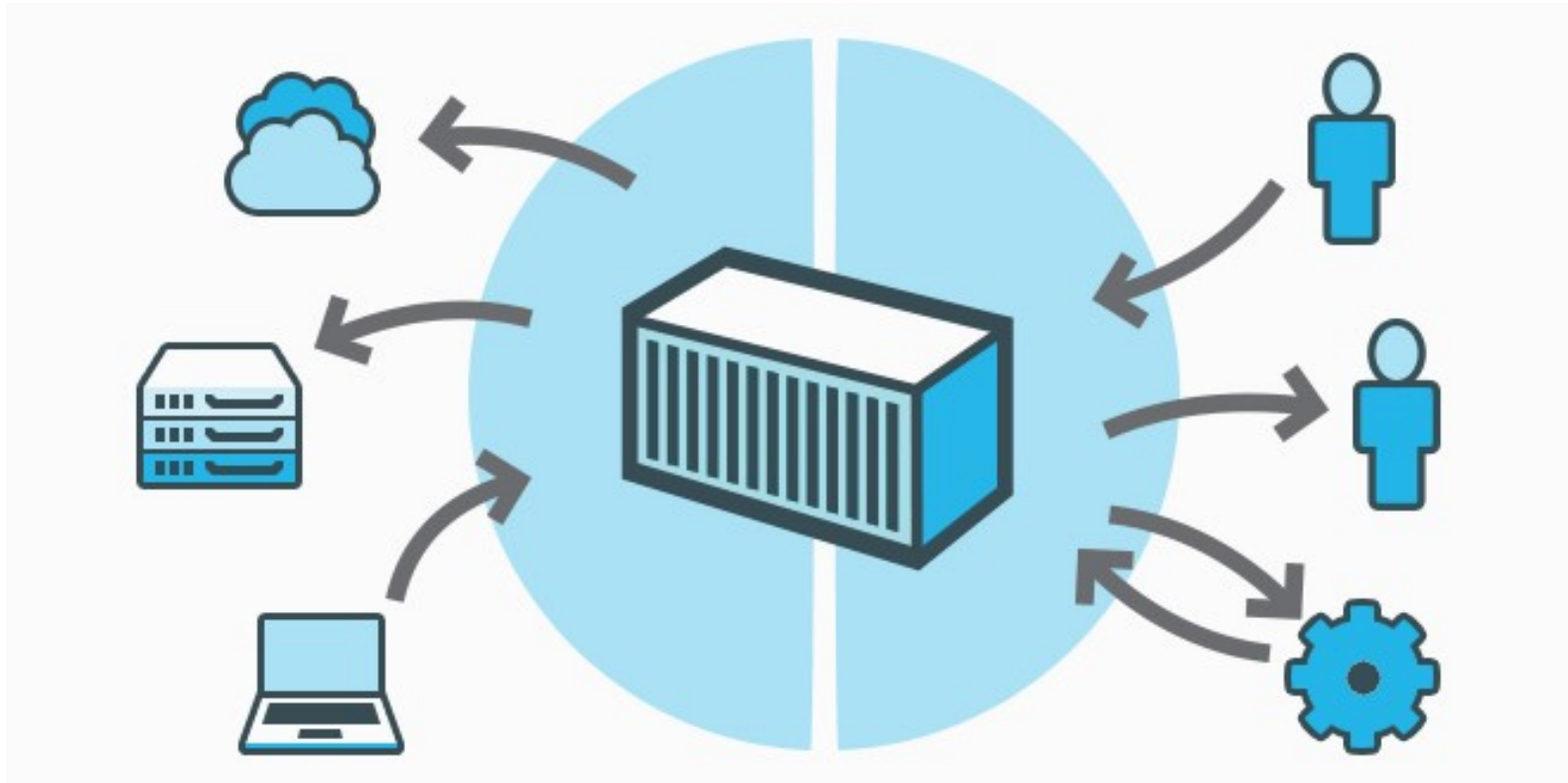
Docker



- Es una tecnología **muy popular** (aunque existen otras tecnologías de contenedores)
- Inicialmente desarrollada para **linux**, aunque dispone de herramientas para **desarrolladores en windows y mac**
- Existe un **repositorio de imágenes** (hub) con contenedores públicos

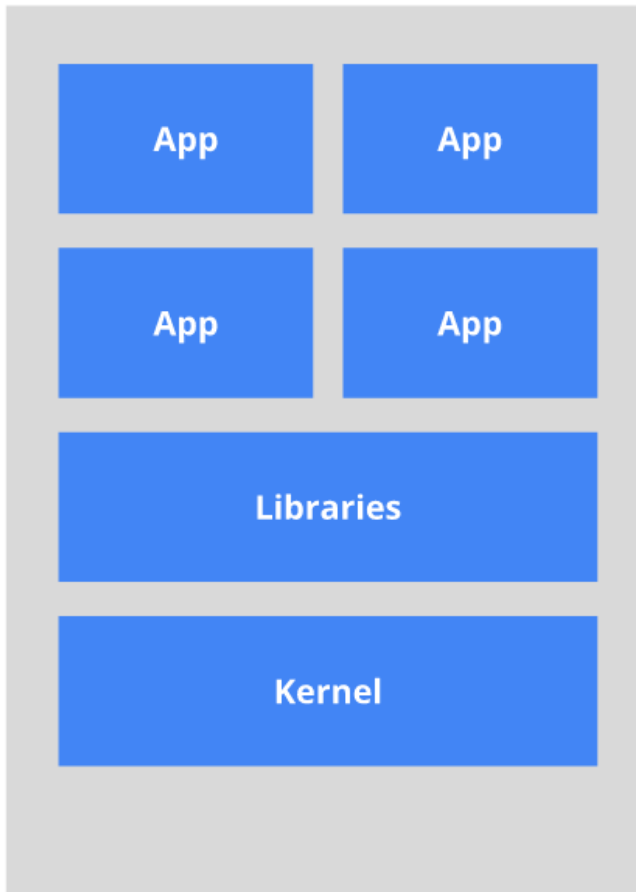
<https://www.docker.com/>

Docker



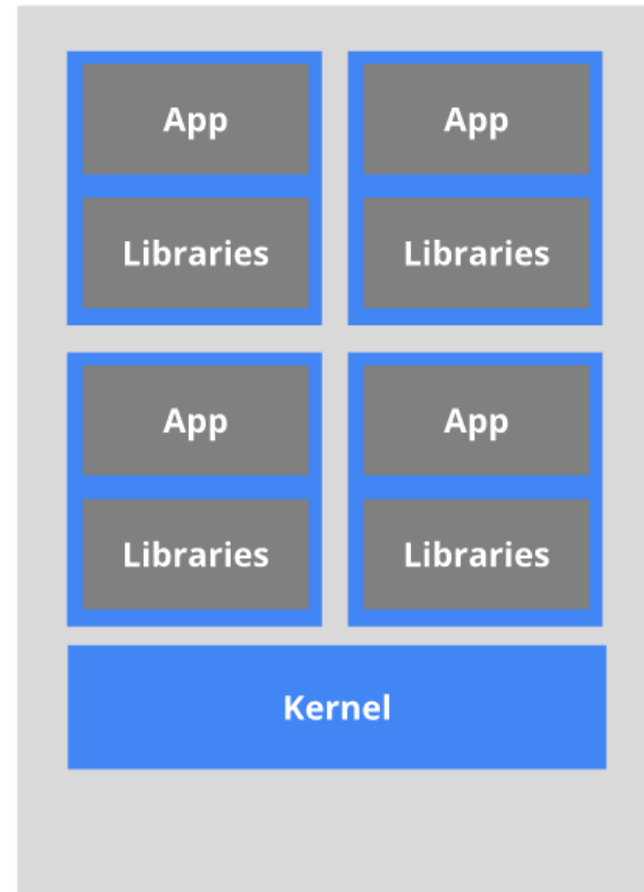
Docker

The old way: Applications on host



*Heavyweight, non-portable
Relies on OS package manager*

The new way: Deploy containers



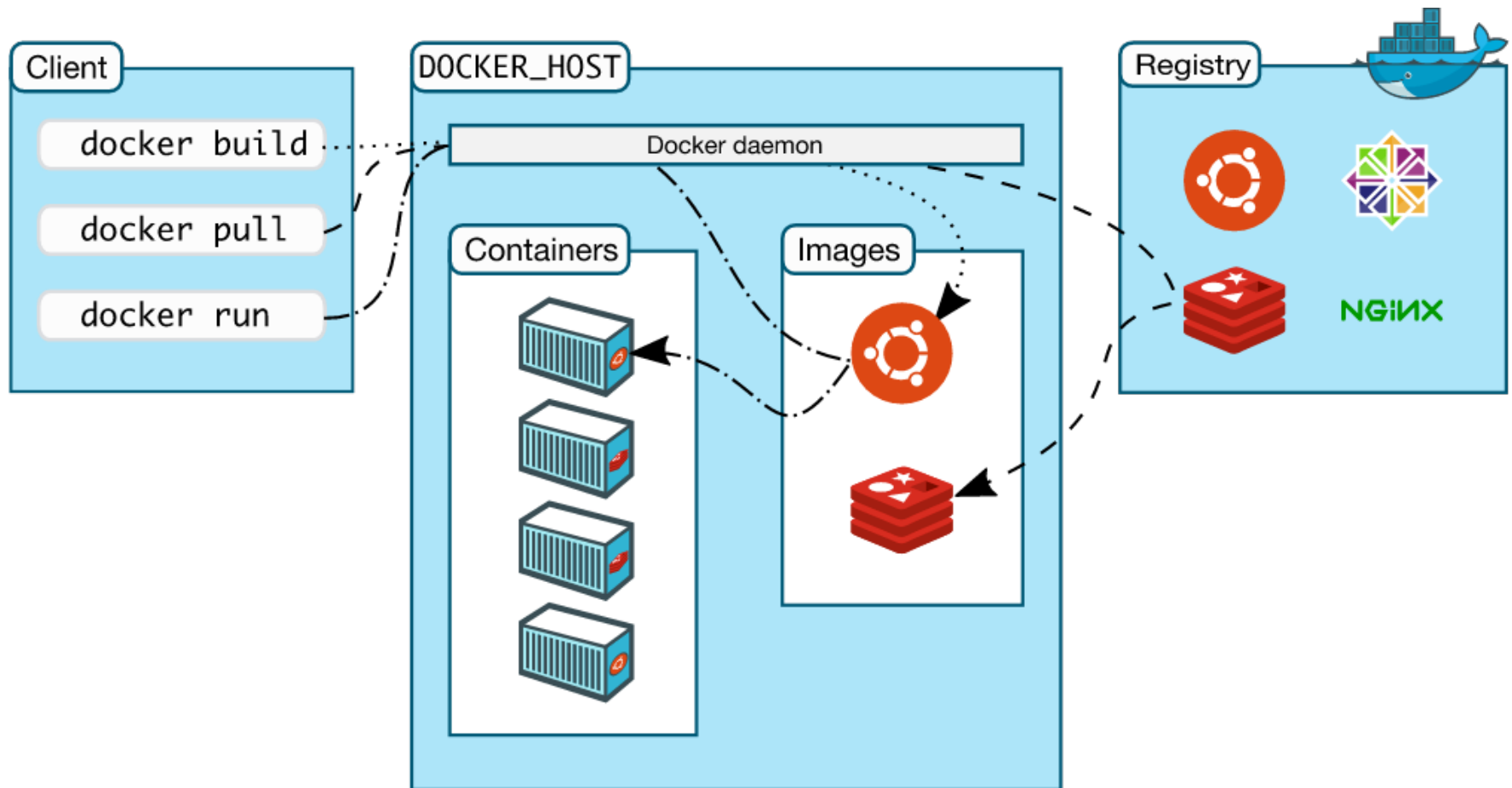
*Small and fast, portable
Uses OS-level virtualization*

- ¿Qué son los contenedores Docker?
 - Son aplicaciones empaquetadas con **todas sus dependencias**
 - Se pueden ejecutar en **cualquier entorno** (linux, windows, mac)
 - Se **descargan de forma automática** si no están disponibles en el sistema
 - Sólo es necesario tener instalado **Docker**

- **Tipos de aplicaciones:**
 - **Servicios de red**
 - web, bbdd, colas de mensajes, cachés, etc.
 - **Línea de comandos**
 - compiladores, conversores de vídeo, generadores de informes...

- **Sistemas operativos soportados**
 - Contenedores **linux**
 - Más usados y más maduros
 - En linux se ejecutan directamente por el kernel
 - En win y mac se ejecutan en máquinas virtuales gestionadas por docker
 - Contenedores **windows**
 - Menos usados y menos maduros
 - Sólo se pueden ejecutar en windows server

Conceptos



Conceptos

- **Imagen docker**
 - Plantilla para un contenedor
 - Contiene las herramientas del SO (**ubuntu, alpine**), librerías (**Java**) and la aplicación en sí (**webapp.jar**)
 - Un contenedor siempre se inicia desde una **imagen**
 - Si se quiere arrancar un contenedor partiendo de una imagen que no está disponible, se **descarga automáticamente** de un registro

Conceptos

- **Docker Registry**

- **Servicio remoto** para subir y descargar imágenes
- Puede guardar varias versiones (**tags**) de la misma imagen
- Las versiones de una misma imagen se almacenan en un mismo **repositorio** (como Git)
- **Docker Hub** es un registro público y gratuito
- Tu puedes tener tu **repositorio privado**

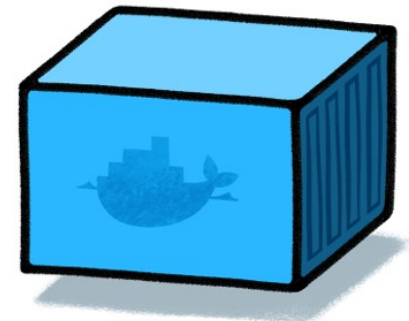
Conceptos

- Algunos repositorios de DockerHub



Conceptos Docker

- **Contenedor Docker**
 - Representa la **aplicación en ejecución**
 - Un contenedor se crea desde **una imagen**
 - Si la aplicación escribe un fichero, el fichero queda dentro del contenedor, no se **modifica la imagen**
 - Los contenedores se pueden **arrancar, pausar y parar**



Conceptos Docker

- **Docker Engine**

- **Servicio local** usado para gestionar docker
- Gestiona las **imágenes** (descarga, creación, subida, etc...)
- Gestiona los **contenedores** (arranque, parada, etc..)
- Se utiliza desde el cliente docker por **línea de comandos** o a través de una **API REST**

Conceptos Docker

- **Docker client**
 - **Command line interface (CLI)** herramienta por línea de comandos para controlar el docker engine
 - Está disponible al **instalar Docker**

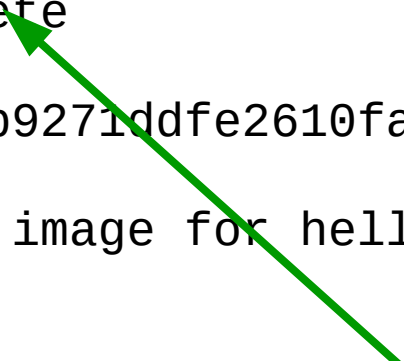


Ejecución de contenedores

Ejecutar "hello-world"

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
03f4658f8b78: Pull complete
a3ed95caeb02: Pull complete
Digest:
sha256:8be990ef2aeb16dbcb9271ddfe2610fa6658d13f6dfb8bc72074cc
1ca36966a7
Status: Downloaded newer image for hello-world:latest

Hello from Docker.
This message shows that your installation
working correctly.
...
```

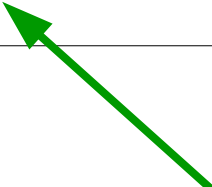


La primera vez la imagen se descarga

Ejecución de contenedores

Ejecutar “hello-world”

```
$ docker run hello-world  
Hello from Docker.  
This message shows that your installation appears to be  
working correctly.  
...
```



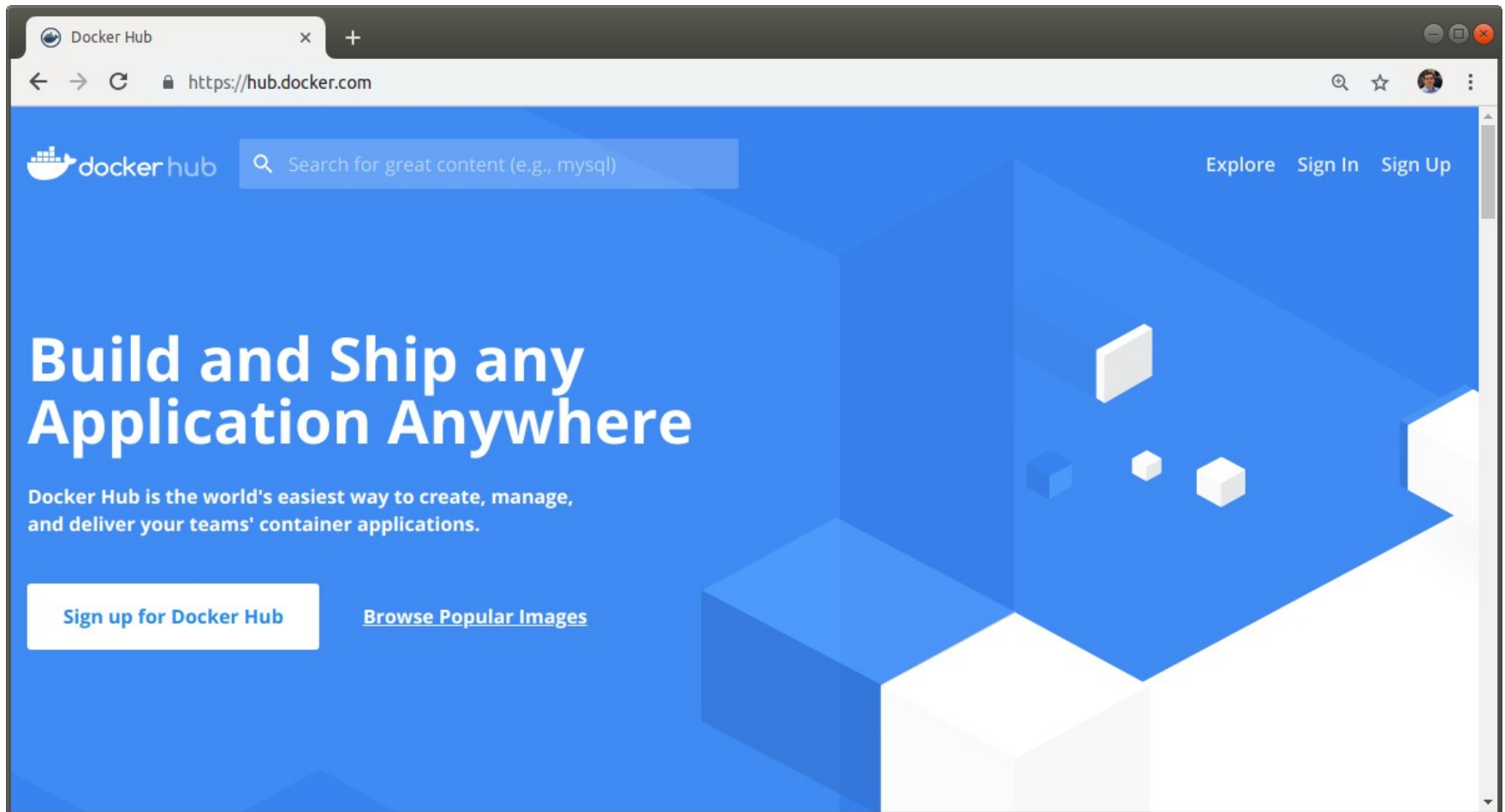
La segunda vez se
usa la vez la imagen
se descarga

Imágenes docker

- Para ejecutar un contenedor es necesario tener una **imagen** en la máquina
- Las imágenes se descargan de un docker registry (**registro**)
- Cada registro tiene un repositorio por cada imagen con múltiples versiones (**tags**)
- **DockerHub** es un registro gratuito en el que cualquiera puede subir imágenes públicas

Imágenes docker

- DockerHub



Servicios de red

- Servidor web en un contenedor

```
docker run --name static-site \
  -e AUTHOR="Your Name" -d \
  -p 9000:80 sequence/static-site
```

Servicios de red

- Servidor web en un contenedor

```
docker run --name static-site \
  -e AUTHOR="Your Name" -d \
  -p 9000:80 sequence/static-site
```

--name static-site

Nombre del contenedor

Servicios de red

- Servidor web en un contenedor

```
docker run --name static-site \
  -e AUTHOR="Your Name" -d \
  -p 9000:80 sequence/static-site
```

-e AUTHOR="Your Name"

Pasar variables de entorno a la aplicación
que se ejecuta en el contenedor

Servicios de red

- Servidor web en un contenedor

```
docker run --name static-site \  
  -e AUTHOR="Your Name" -d \  
  -p 9000:80 sequence/static-site
```

-d

Ejecuta el contenedor en segundo plano
(no bloquea la shell durante la
ejecución)

Servicios de red

- Servidor web en un contenedor

```
docker run --name static-site \
  -e AUTHOR="Your Name" -d \
  -p 9000:80 sequence/static-site
```

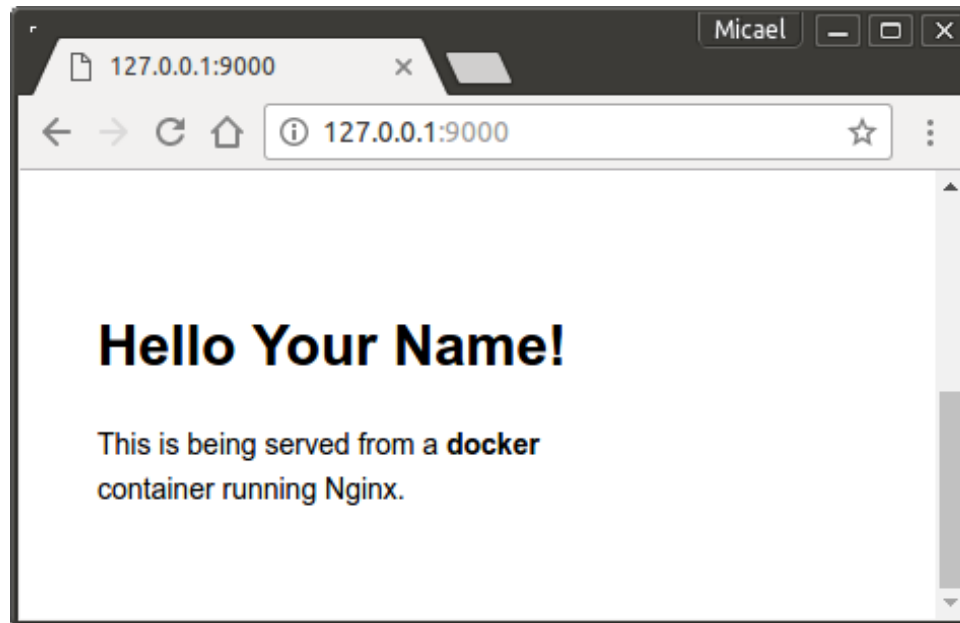
-p 9000:80

Connects the host port 9000 to
the port 80 in the container

Servicios de red

- Usar el servicio

- Abre la URL <http://127.0.0.1:9000> en un browser accede al puerto 80 de la aplicación en el contenedor



Gestión de contenedores

• Contenedores en ejecución

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             NAMES
a7a0e504ca3e       sequence/static-site  "/bin/sh -c 'cd /usr/"  Up 26 seconds
```

Container id es
a7a0e504ca3e

Este id se usa para
referirte al contenedor

STATUS es UP

Gestión de contenedores

- **Logs**

- Obtener la salida estándar de un contenedor

```
$ docker logs static-site
```

- Útil para contenedores arrancados en segundo plano

Gestión de contenedores

- Parar y borrar contenedores
 - Parar un contenedor en ejecución

```
$ docker stop a7a0e504ca3e
```

- Borrar los ficheros del contenedor parado

```
$ docker rm a7a0e504ca3e
```



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 manual de contenedores 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 estas funcionalidades 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





kubernetes

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 Foundation** con múltiples miembros de peso



kubernetes

<http://kubernetes.io/>



CLOUD NATIVE
COMPUTING FOUNDATION

<https://www.cncf.io/>



CLOUD NATIVE

COMPUTING FOUNDATION

 Alibaba Cloud

aws


 Azure


MESOSPHERE


CISCO

DELL Technologies


docker

ORACLE®

FUJITSU


Google Cloud


HUAWEI

Pivotal®

 IBM Cloud

intel®


JD.COM

 redhat.

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

Kubernetes

- Kubernetes (k8s) es un software diferente a docker que tiene que instalarse por separado
- Usa **docker** internamente para ejecutar los contenedores, pero también puede funcionar con otras tecnologías de contenedores como **rkt**, pero no son muy maduras



<https://coreos.com/rkt/>



Kubernetes en local

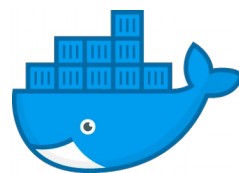
- Para **desarrollo** se puede instalar un kubernetes en tu ordenador local
 - **Minikube** (un nodo)
 - Kube-spawn (múltiples nodos)
 - Docker for Windows y Mac



minikube



kube-spawn



docker

Kubernetes en un cluster

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



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 en un cluster

- Gestionado por un proveedor de computación en la nube



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



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



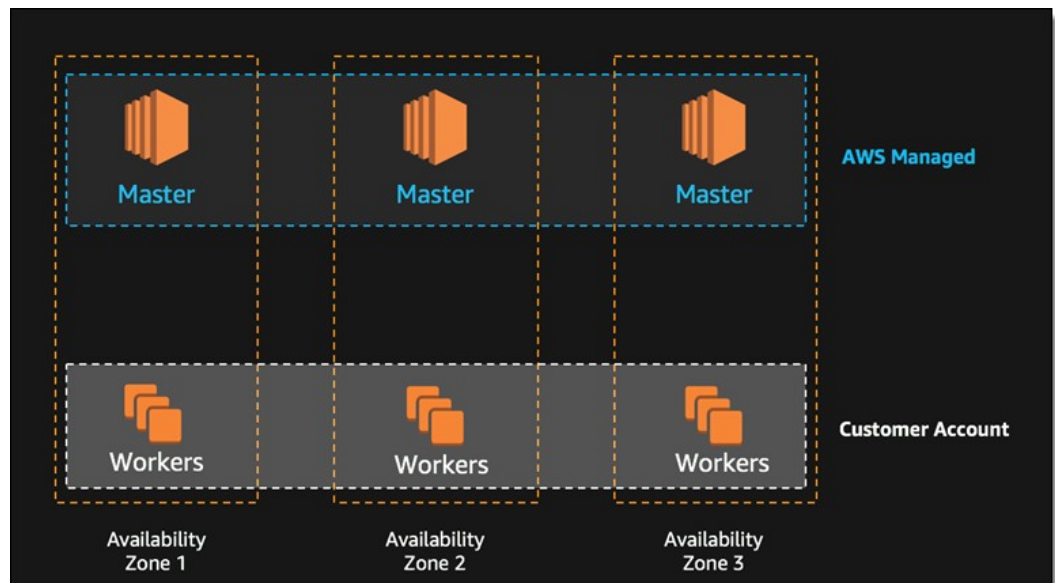
<https://aws.amazon.com/es/eks/>



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

Kubernetes en un cluster

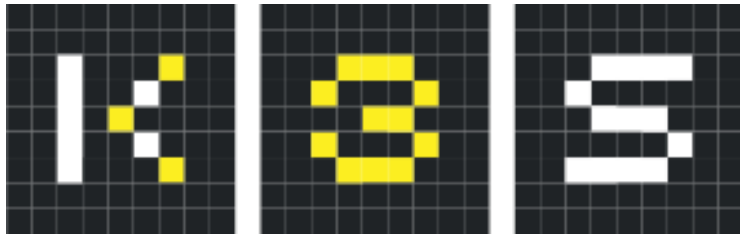
- Amazon EKS es el Kubernetes gestionado por AWS



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

Kubernetes “mini”

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

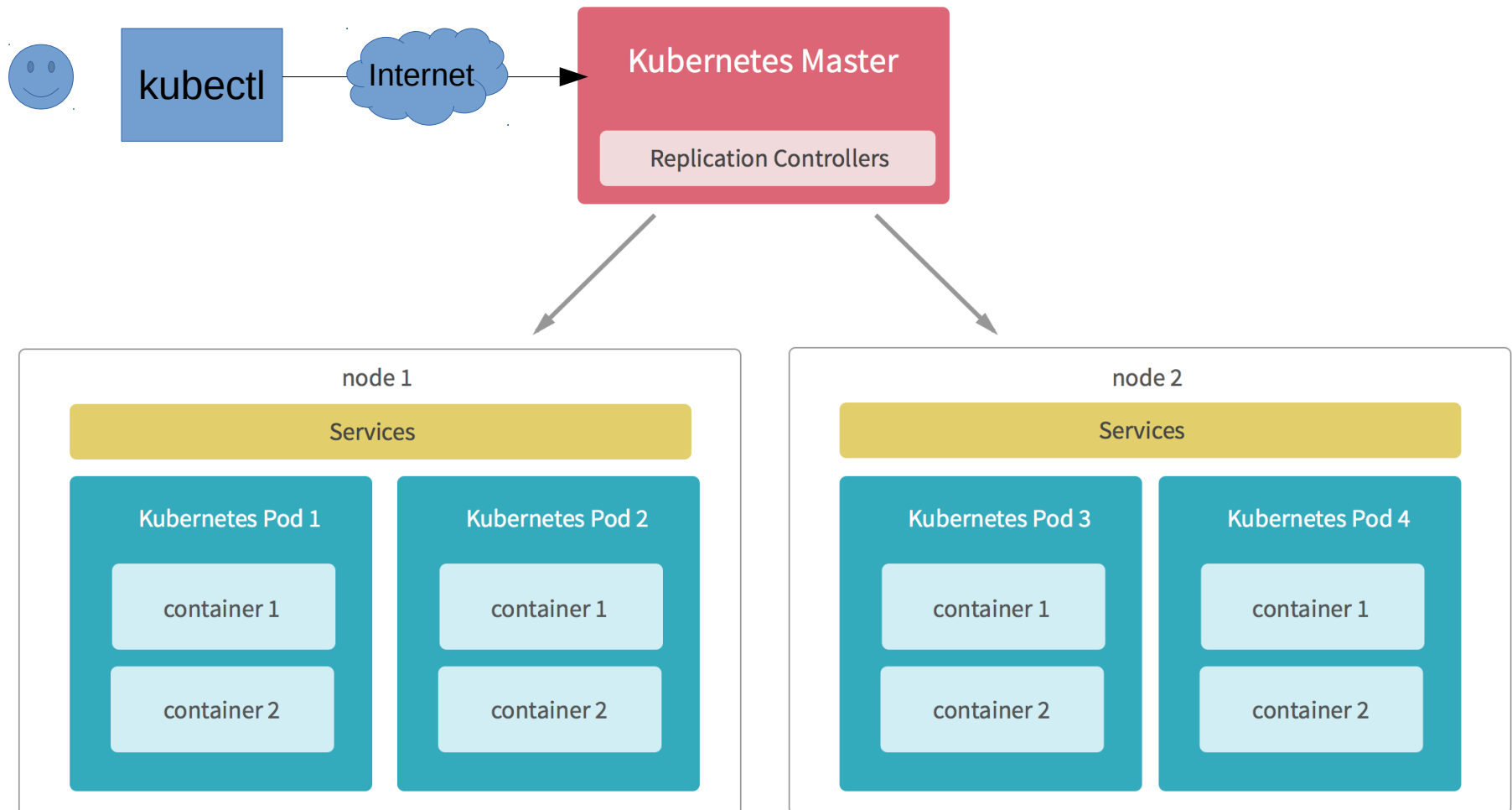


<https://k3s.io/>

MicroK8s

<https://microk8s.io/>

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
```

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

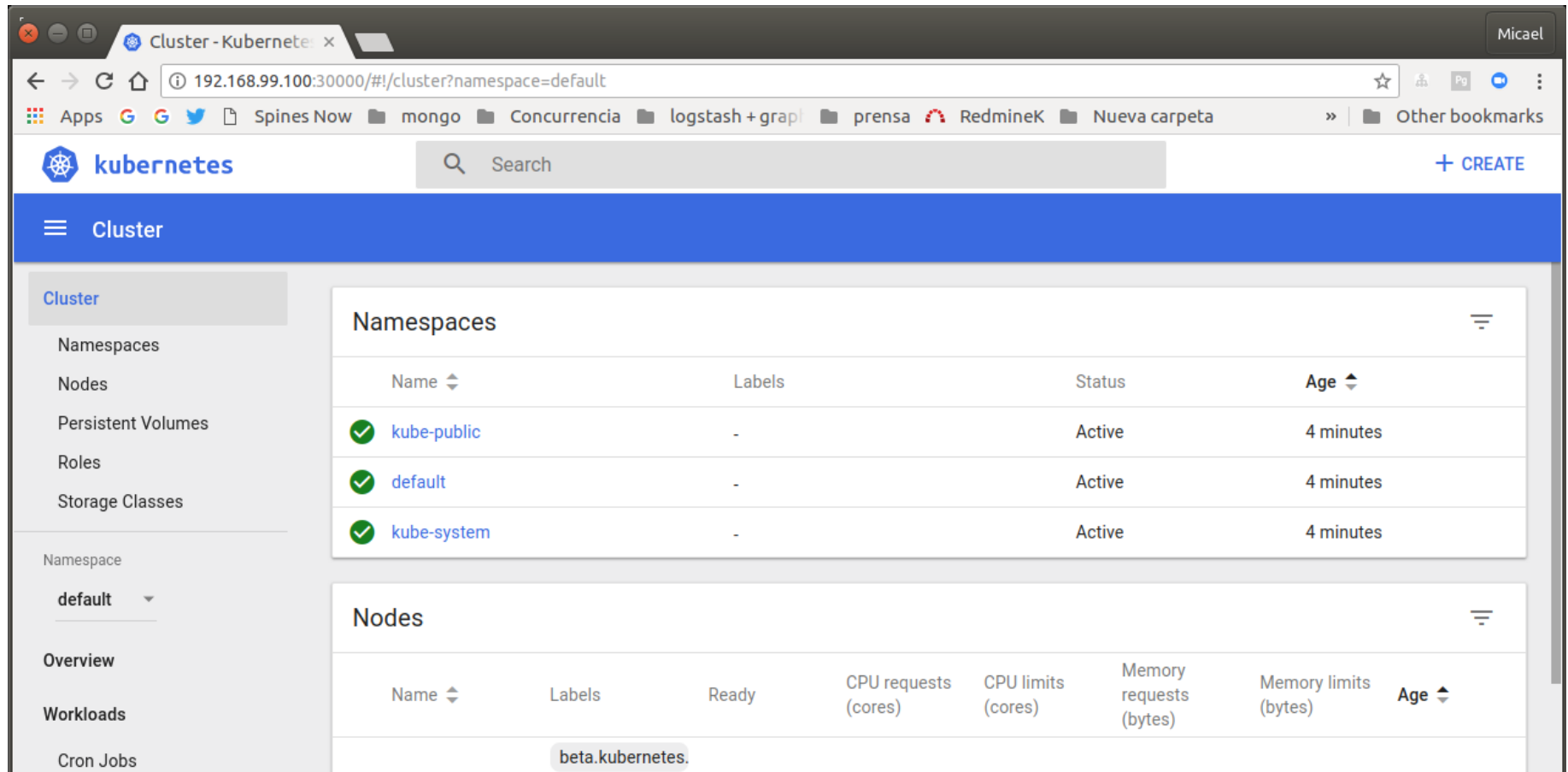
```
$ docker ps
```

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



The screenshot shows the Kubernetes Dashboard interface. The top navigation bar includes the Kubernetes logo, a search bar, and a '+ CREATE' button. The left sidebar contains a menu with 'Cluster' (selected), 'Namespaces', 'Nodes', 'Persistent Volumes', 'Roles', 'Storage Classes', 'Namespace' (with a dropdown set to 'default'), 'Overview', 'Workloads', and 'Cron Jobs'. The main content area displays two sections: 'Namespaces' and 'Nodes'.

Namespaces Table:

| Name | Labels | Status | Age |
|---------------|--------|--------|-----------|
| ✓ kube-public | - | Active | 4 minutes |
| ✓ default | - | Active | 4 minutes |
| ✓ kube-system | - | Active | 4 minutes |

Nodes Table:

| Name | Labels | Ready | CPU requests (cores) | CPU limits (cores) | Memory requests (bytes) | Memory limits (bytes) | Age |
|------------------|--------|-------|----------------------|--------------------|-------------------------|-----------------------|-----|
| beta.kubernetes. | | | | | | | |

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/>

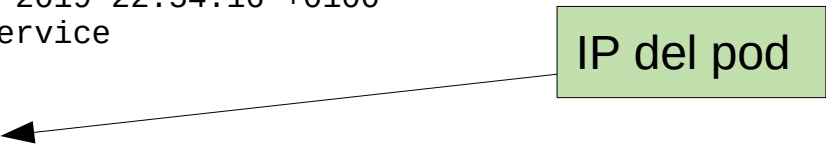
Pods

- 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

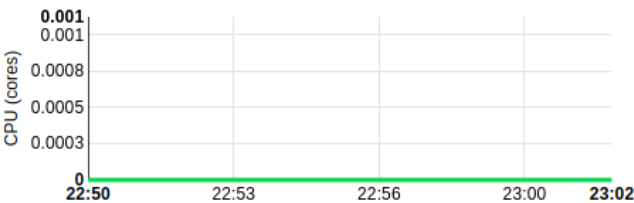
Ingresses

Services

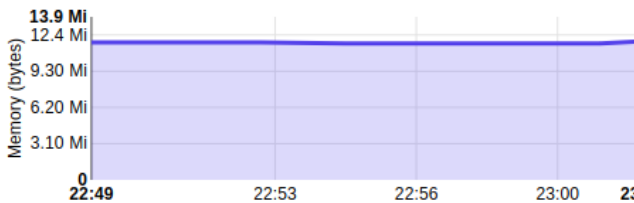
Config and Storage

Config Maps


CPU usage



Memory usage ⓘ



Pods

| Name | Node | Status | Restarts | Age | CPU (cores) | Memory (bytes) |
|---|----------|---------|----------|-----|--------------------------|----------------------------------|
|  simpleservice | minikube | Running | 0 | - | <div><div></div></div> 0 | <div><div></div></div> 11.863 Mi |

Pods

simpleservice - Kubernetes Dashboard

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

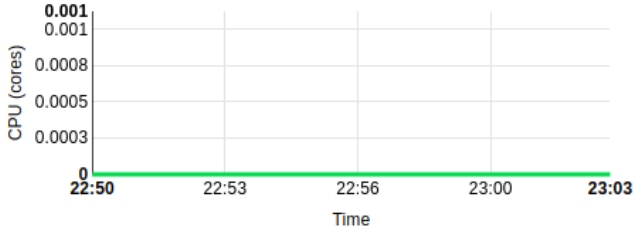
Ingresses

Services

Config and Storage

Config Maps

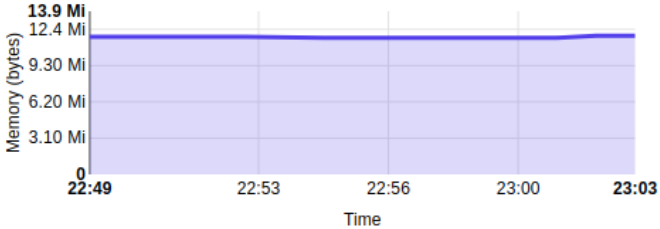
CPU usage



CPU (cores)

Time

Memory usage ⓘ



Memory (bytes)

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

The screenshot shows the Kubernetes Dashboard interface. The browser tabs at the top are 'simpleservice - Kubernetes Dashboard' and 'Logs - Kubernetes Dashboard'. The address bar shows the URL: `127.0.0.1:42703/api/v1/namespaces/kube-system/services/http:kubernetes-dashboard:/proxy/#!/log/default/simpleservice/pod?namespace=default`. The dashboard header includes the Kubernetes logo, a search bar, and a '+ CREATE' button. The left sidebar contains a navigation menu with sections: 'Cluster' (Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes), 'Namespace' (default), 'Overview', 'Workloads' (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers), and 'Logs'. The main content area is titled 'Logs from simpleservice in simpleservice' and displays a log entry: `2019-03-05T09:54:17 INFO This is simple service in version v0.5.0 listening on port 9876 [at line 142]`. The log entry is on a black background with white text. At the bottom right of the log area, there are navigation controls: `|< < > >|`.

- 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/simple-service
```

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
```

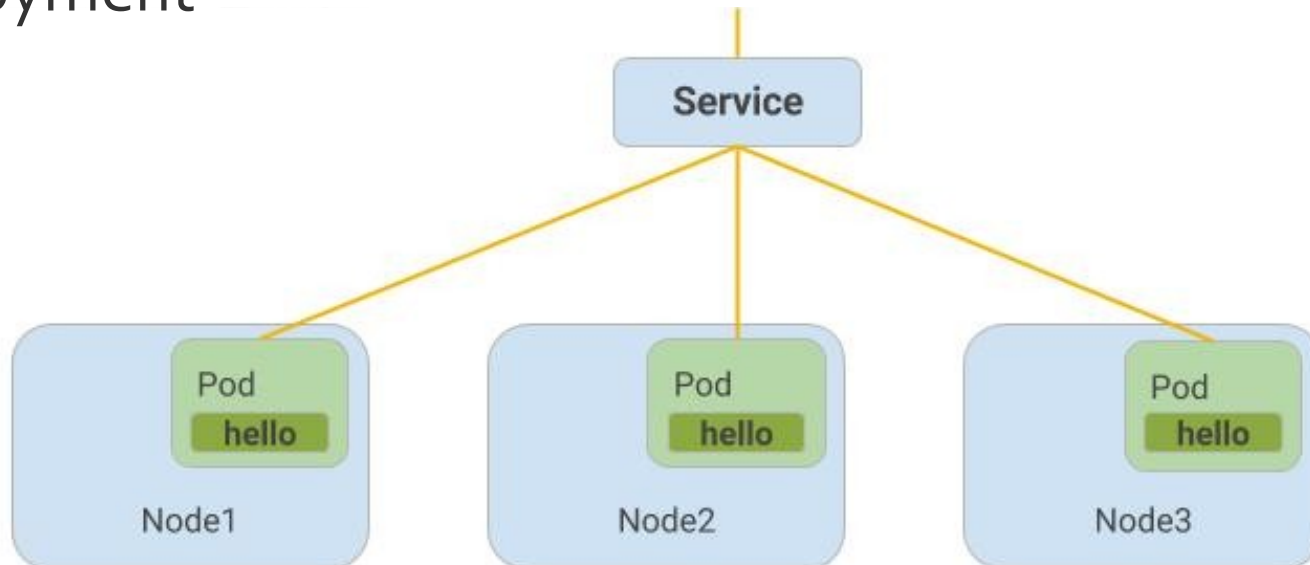
- Un nuevo pod con la nueva imagen se inicia y se finaliza el pod anterior
- No funciona cuando se actualiza el tag "latest" porque a ojos de Kubernetes nada ha cambiado

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 |

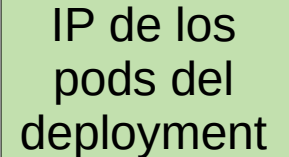
Servicios

- 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>
```

IP de los
pods del
deployment



- Borrado del servicio

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

- **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**

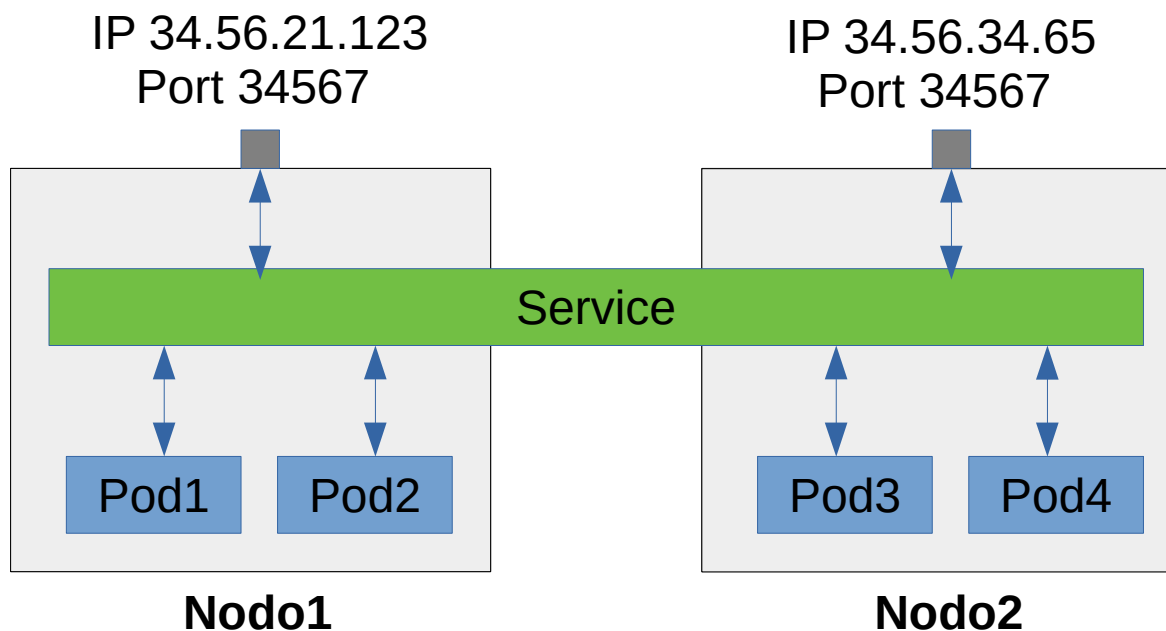
- NodePort

- 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

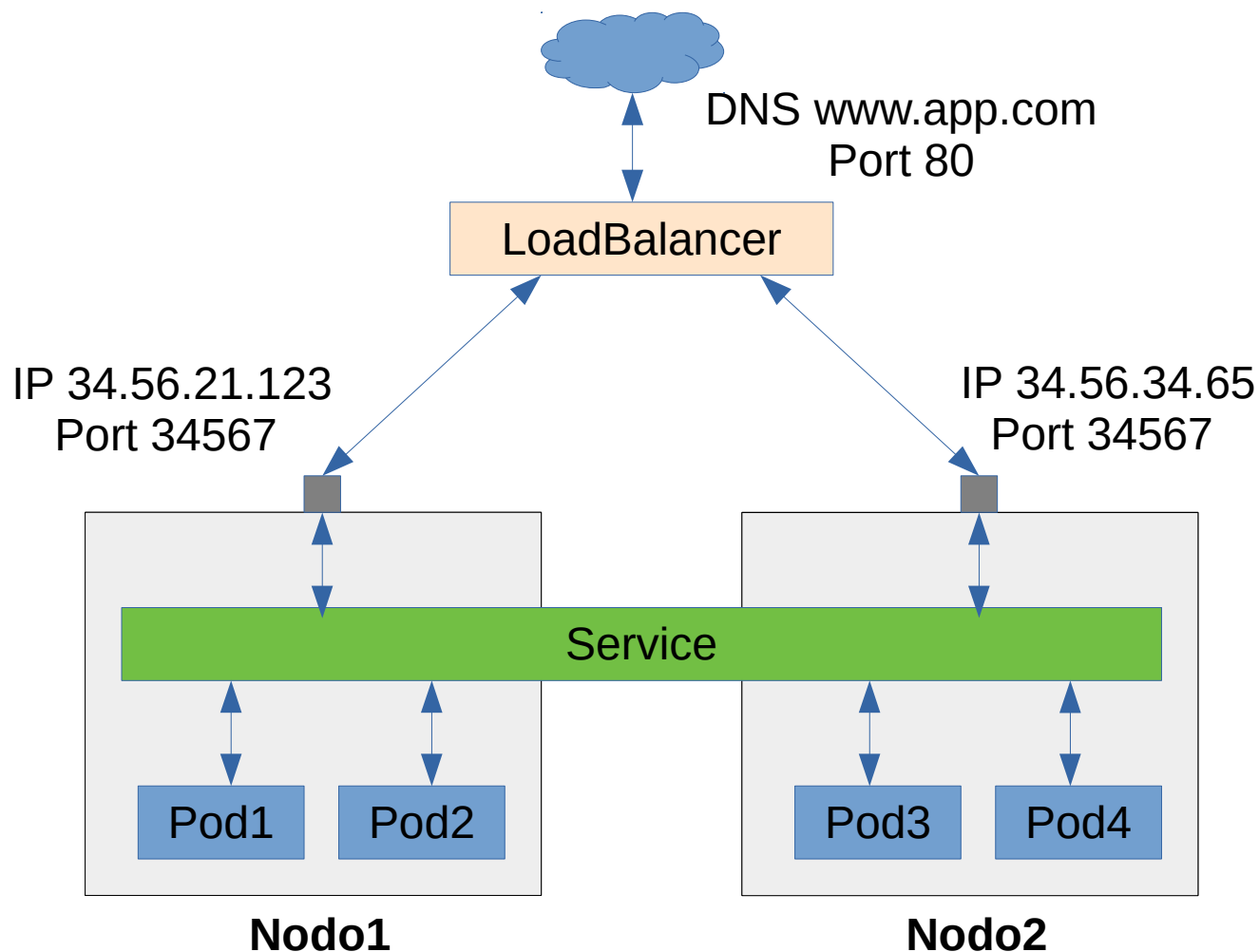
Servicios

- **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

- **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



minikube

- 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

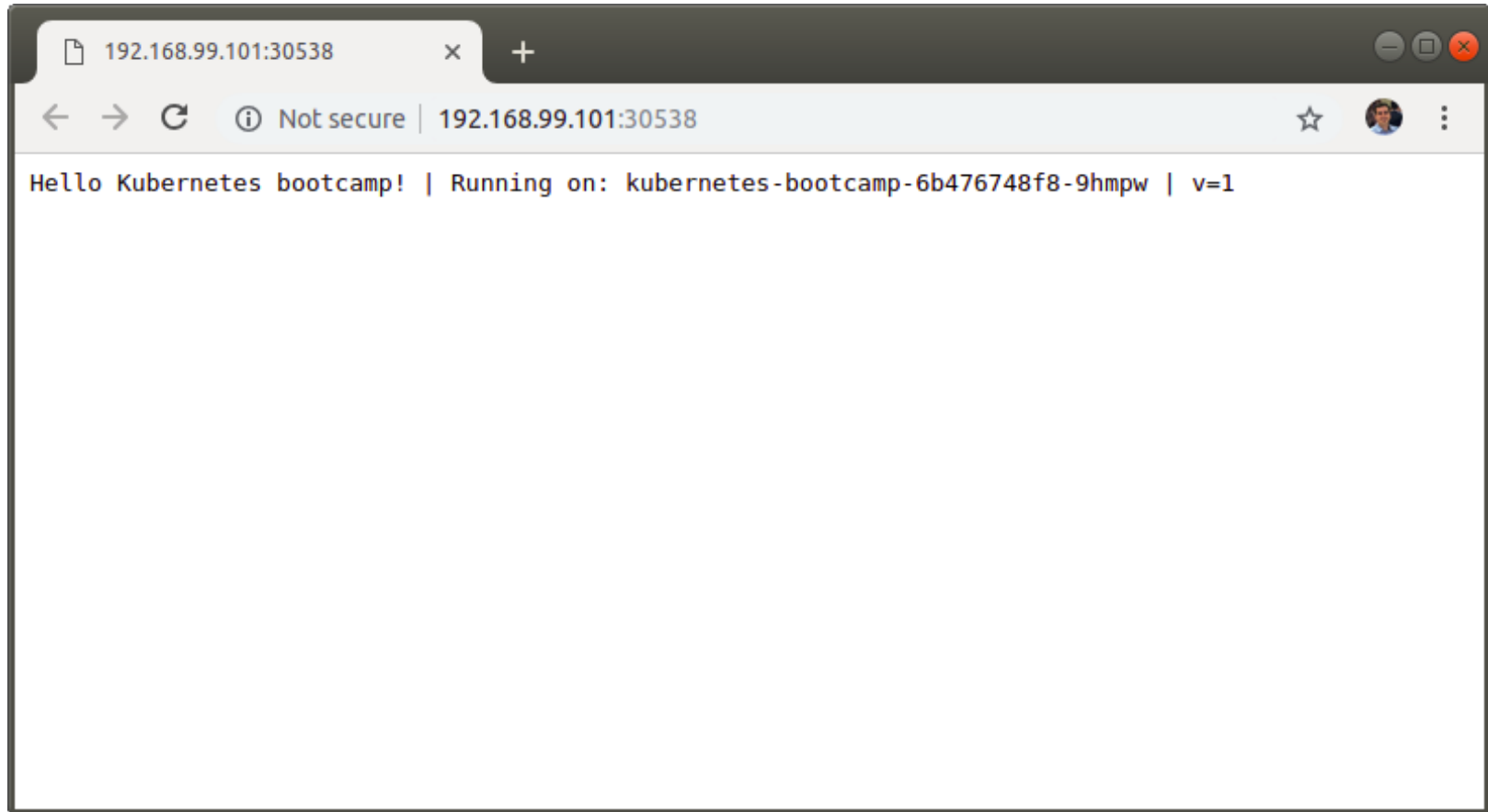
```
$ minikube ip  
192.168.99.100  
  
$ kubectl get service kubernetes-bootcamp \\\n--output='jsonpath={.spec.ports[0].nodePort}'  
32041
```

<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 Console

Secure | https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#LoadBalancers:sort=createTime

Apps | Spines Now | mongo | Concurrency | logstash + graf | prensa | RedmineK | Nueva carpeta | P2 Concurrente | Other bookmarks

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

Activations

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 |
|-------------------------------------|----------------------------------|----------------------------------|-------|-----------------------|---------------------------|---------|
| <input type="checkbox"/> | a0acdebdc5f6d11e896170afb7f57108 | a0acdebdc5f6d11e896170af... | | vpc-0daa1a9311a64bb67 | eu-west-1b, eu-west-1c... | classic |
| <input type="checkbox"/> | api-cluster-k8s-codeurjc--tosdtv | api-cluster-k8s-codeurjc--tos... | | vpc-05949277148c77f65 | eu-west-1a | classic |
| <input type="checkbox"/> | bastion-cluster-k8s-codeu-32ql7t | bastion-cluster-k8s-codeu-3... | | vpc-05949277148c77f65 | eu-west-1a | classic |
| <input type="checkbox"/> | a4559a9e5630f11e88db80acf913b2c4 | a4559a9e5630f11e88db80a... | | vpc-05949277148c77f65 | eu-west-1a | classic |
| <input type="checkbox"/> | a9a99d632632a11e88db80acf913b2c4 | a9a99d632632a11e88db80a... | | vpc-05949277148c77f65 | eu-west-1a | classic |
| <input checked="" type="checkbox"/> | 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

Servicios 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

✕

?

| <input type="checkbox"/> Nombre ^ | Estado | Tipo | Endpoints | Grupos | Espacio de nombres | Clúster |
|---|-----------|----------------------|--------------------|--------|--------------------|------------------|
| <input type="checkbox"/> 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/codeurjc/Curso-Kubernetes/tree/master/ejemplo1/web-python>

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

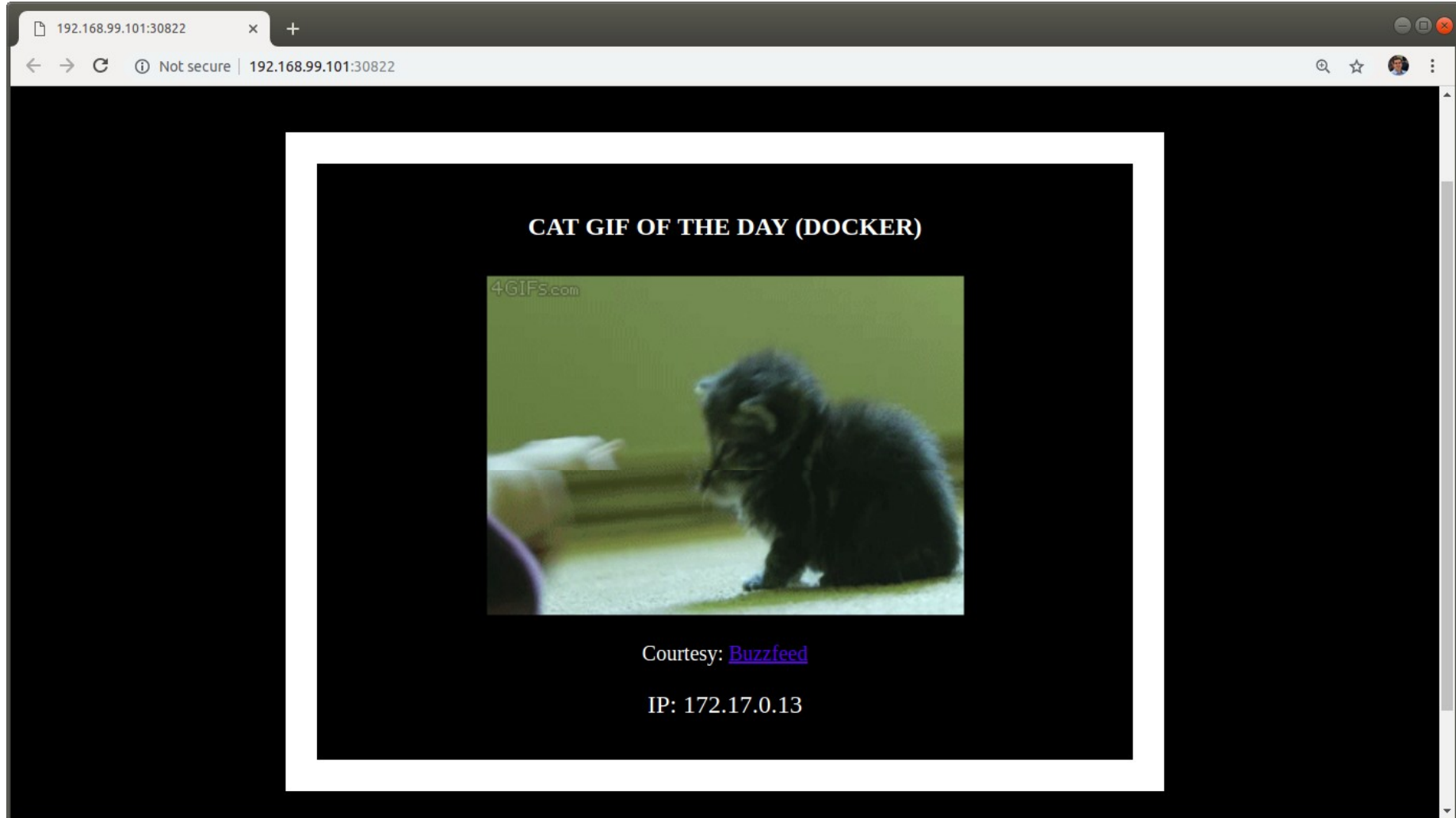
Solución

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

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

```
$ minikube service webgatos
```

Solución



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 recursos 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 |

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:
```

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
- En general se utilizan ficheros de descripción de los recursos (llamados spec) 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>

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
```

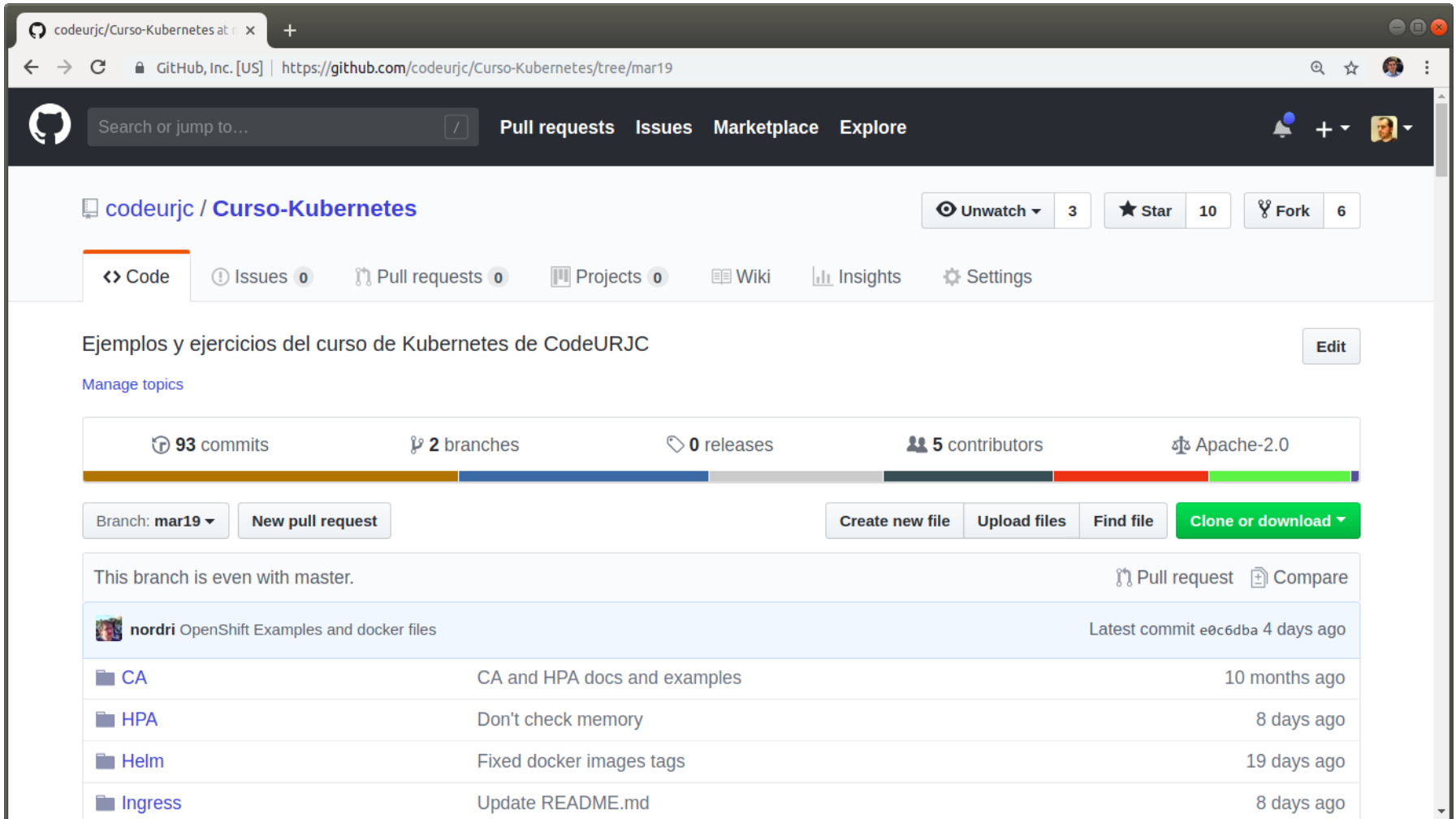
```
---
apiVersion: v1
kind: Service
metadata:
  name: webgatos-service
  labels:
    app: webgatos
spec:
  ports:
    - port: 5000
      protocol: TCP
      name: webgatos-port
  selector:
    app: webgatos
    type: NodePort
```


- **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

<https://github.com/codeurjc/Curso-Kubernetes/tree/oct19>



The screenshot shows the GitHub repository page for `codeurjc/Curso-Kubernetes`. The repository is in the `mar19` branch. The page displays the repository name, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. The repository statistics show 93 commits, 2 branches, 0 releases, 5 contributors, and Apache-2.0 license. A table lists the files and folders in the repository, including CA, HPA, Helm, and Ingress, with their respective commit messages and dates.

codeurjc / Curso-Kubernetes

Unwatch 3 Star 10 Fork 6

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings


Ejemplos y ejercicios del curso de Kubernetes de CodeURJC

Manage topics

93 commits 2 branches 0 releases 5 contributors Apache-2.0

Branch: mar19 New pull request Create new file Upload files Find file Clone or download

This branch is even with master. Pull request Compare

| | | |
|--|-------------------------------------|----------------------------------|
|  nordri | OpenShift Examples and docker files | Latest commit e0c6dba 4 days ago |
| CA | CA and HPA docs and examples | 10 months ago |
| HPA | Don't check memory | 8 days ago |
| Helm | Fixed docker images tags | 19 days ago |
| Ingress | Update README.md | 8 days ago |

- **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

- 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
- Activar el ingress controller en minikube

```
$ minikube addons enable ingress
```

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

Ingress

- **Uso de ingress**

- Desplegamos la web de gatos (deployment y service)

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

- Desplegamos la web de anuncios (deployment y service)

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

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


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: curso.minikube.io
      http:
        paths:
          - path: /anuncios/
            backend:
              serviceName: java-webapp-db-service
              servicePort: 8080
          - path: /gatos
            backend:
              serviceName: webgatos-service
              servicePort: 5000
```

Ingress

- Crear el recurso Ingress

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

- Acceso a los servicios

- **Proveedor cloud:** Se usa un balanceador de carga y un nombre DNS asociado
- **Minikube:** Simulamos un nombre DNS a la IP de la VM

Ingress

- Simular DNS en local
 - En Linux
 - `export MINIKUBE_IP=$(minikube ip)`
 - `echo $MINIKUBE_IP curso.minikube.io | sudo tee --append /etc/hosts >/dev/null`
 - En Mac
 - Obtener la IP de Minikube
 - `$ minikube ip`
 - Luego editar el fichero
 - `$ sudo nano /private/etc/hosts`
 - y añadir la línea
 - `192.168.99.100 curso.minikube.io`

- **Simular DNS en local**

- En Windows

- 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:
- `192.168.99.100 curso.minikube.io`
- 6. Haz click en Fichero > Guardar para guardar los cambios.
- 7. Puedes cerrar el Notepad.

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
- Algunos tipos de volúmenes: 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, 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, como `/var/lib/docker`

- **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



GLUSTER

<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 (emptyDir, hostPath...)

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** en base a un tipo de almacenamiento que esté disponible en el cluster (**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

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`

Volúmenes

- 2) Creamos el PersistentVolume
 - En desarrollo, se usa 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 volumen/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 volumen/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 | STATUS | CLAIM | CAPACITY | ACCESS MODES | RECLAIM POLICY |
|-----------------------------|--------|----------------|--------------|--------------|----------------|
| | | | STORAGECLASS | REASON | AGE |
| persistentvolume/10g-volume | Bound | default/pvc-3g | 10Gi | RWO | Retain |
| | | | 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 volumen/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
```

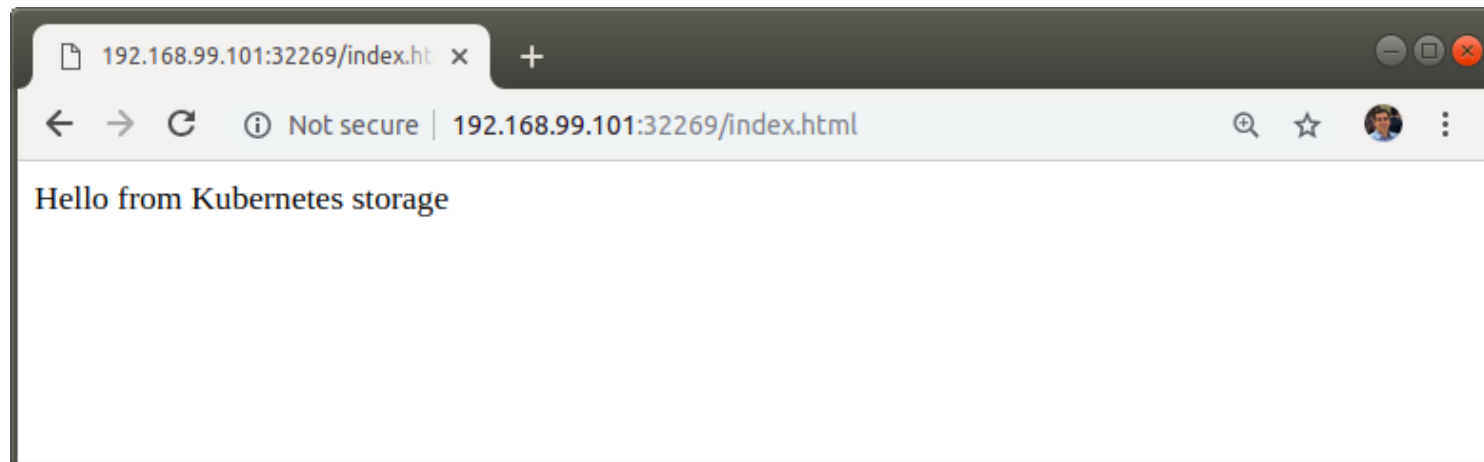
```
apiVersion: v1
kind: Service
metadata:
  name: webserver
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 80
      name: web-port
  selector:
    app: webserver
  type: NodePort
```

Volúmenes

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

```
$ minikube service webserver
```

- Deberíamos 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
 - Ejemplo 2 (Sep 2017)
 - Basado en **Portworx**, una capa de persistencia sobre el cloud diseñada para contenedores
 - Específico para AWS con Kops



BBDD en K8s

- **MySQL** escalable y tolerante a fallos
 - **Vitess** es un sistema de clusterización para el escalado horizontal de MySQL
 - Está preparado para ser desplegado en Kubernetes

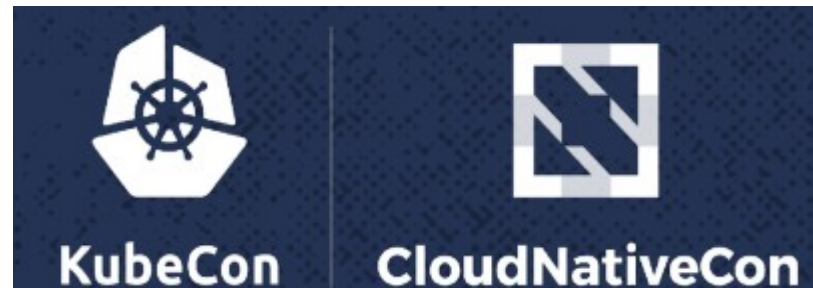
<https://vitess.io/>

<https://vitess.io/getting-started/>



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
 - 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)
 - Basado en el concepto de **Operators de CoreOS**
 - Permite instalar un MySQL replicado, tolerante a fallos y con backups de forma muy sencilla
 - MySQL Operator se instala como pod en k8s para poder crear un cluster de MySQL

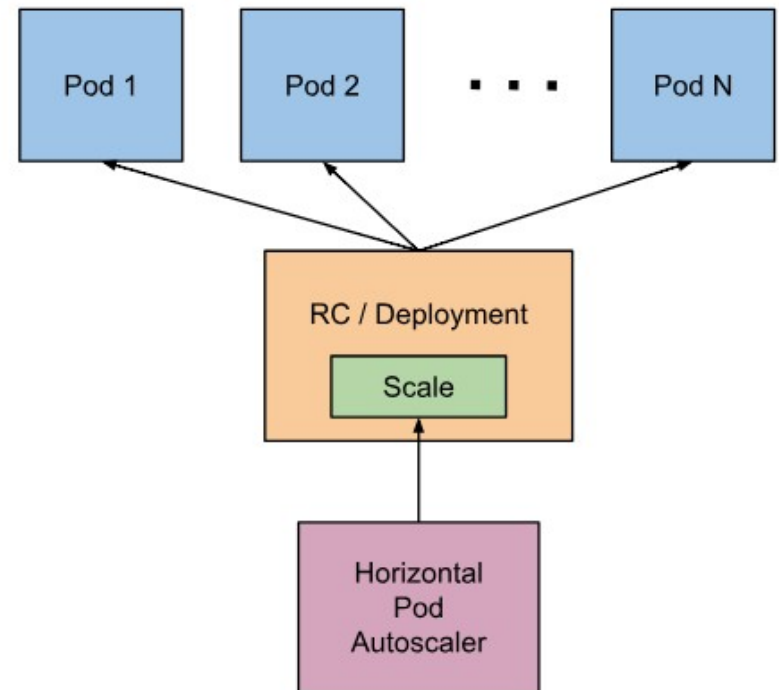
<https://coreos.com/blog/introducing-operators.html>

<https://coreos.com/operators/>

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

Autoescalado

- **Horizontal Pod Autoscaler**
 - Recurso encargado de **modificar dinámicamente el número de réplicas** en función del consumo de recursos (**memoria y CPU**) de los pods de un deployment
 - También se pueden usar métricas personalizadas



<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>
<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

Autoescalado

- Para que funcione es necesario habilitar la **recogida de métricas** de los pods
 - **Minikube**
 - `$ minikube addons enable heapster`
 - `$ minikube addons enable metrics-server`
 - **AWS con Kops**
 - `$ kubectl apply -f`
<https://raw.githubusercontent.com/kubernetes/kops/master/addons/metrics-server/v1.8.x.yaml>

Autoescalado

- Ejemplo


- Desplegamos una aplicación web que con cada petición genera carga de CPU

```
$ kubectl create -f HPA/deployment.yaml
```

```
<?php
    $x = 0.0001;
    for ($i = 0; $i <= 1000000; $i++) {
        $x += sqrt($x);
    }
    echo "OK!";
?>
```

Autoescalado

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
  labels:
    app: php-apache
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: php-apache
  replicas: 1
  template:
    metadata:
      labels:
        app: php-apache
    spec:
      containers:
        - name: hpa-example
          image: codeurjc/hpa-example:v1
          resources:
            limits:
              cpu: 200m
              memory: 128Mi
            requests:
              cpu: 100m
              memory: 64Mi
```



```
---
apiVersion: v1
kind: Service
metadata:
  name: php-apache
spec:
  ports:
    - port: 80
  selector:
    app: php-apache
```

Autoescalado

- Ejemplo

- Creamos un recurso **HorizontalPodAutoscaler** que supervisará memoria y cpu escalando entre 1 y 10 réplicas

```
$ kubectl create -f HPA/hpa-autoscaling.yaml
```

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 10
```

Autoescalado

- Ejemplo

- Aumentamos la carga de la aplicación web lanzando un pod que genera peticiones

```
$ kubectl run load-generator --generator=run-pod/v1 \
  -it --image=busybox:1.30 /bin/sh
```

```
# while true; do wget -q -O- http://php-apache; done
```

Autoescalado

- Ejemplo

- Podemos observar cómo aumenta el número de pods de forma dinámica

```
$ watch kubectl get pods \
    --selector=app=php-apache
```

Autoescalado

- **Cluster autoscaling**

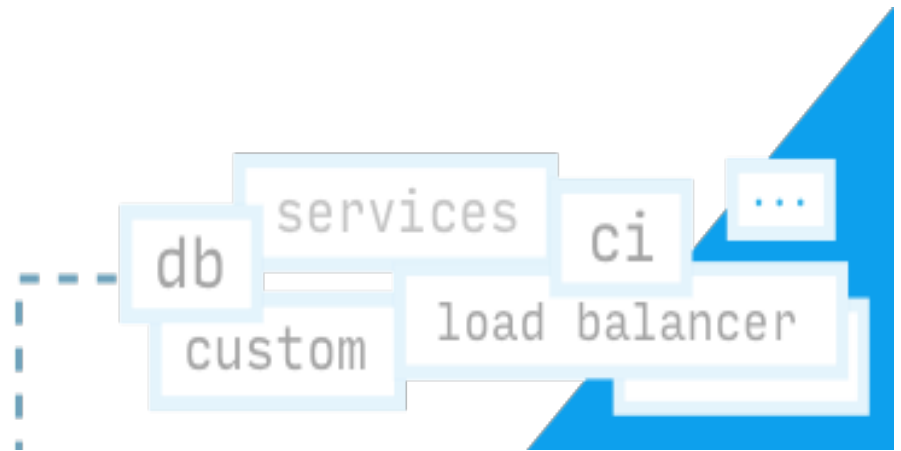
- El **número de nodos** de un cluster puede cambiar dinámicamente en función del número de pods del cluster
- Si un nuevo pod que se va a crear **no cabe en el cluster** (por la **reserva de recursos** de cada uno), se crea un nuevo nodo en el cluster
- Esta funcionalidad requiere de la **instalación de un plugin** en Kubernetes dependiendo del proveedor cloud / VMs.

<https://kubernetes.io/docs/tasks/administer-cluster/cluster-management/#cluster-autoscaling>

<https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler>

Helm

- Una aplicación completa está formada por múltiples recursos Kubernetes:
 - Deployment del frontal web
 - Deployment de la BBDD
 - Servicio de frontal web
 - Configuración del Ingress
 - ConfigMap
 - PersistenceVolumeClaim
- La **gestión manual** de todos esos recursos es bastante **tediosa**



Helm



- **Helm** permite gestionar los recursos Kubernetes de una aplicación
- Se define como “**El gestor de paquetes para Kubernetes**”
- Las aplicaciones de Helm se llaman **Charts**
- Dispone de un **repositorio oficial** de aplicaciones
- Se pueden tener **repositorios privados**

<https://helm.sh/>

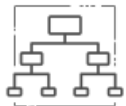
Helm



- Las aplicaciones se pueden actualizar y hacer rollback de forma sencilla
- Está mantenido por la CNCF



Why Teams ❤️ Helm



Manage Complexity

Charts describe even the most complex apps; provide repeatable application installation, and serve as a single point of authority.



Easy Updates

Take the pain out of updates with in-place upgrades and custom hooks.



Simple Sharing

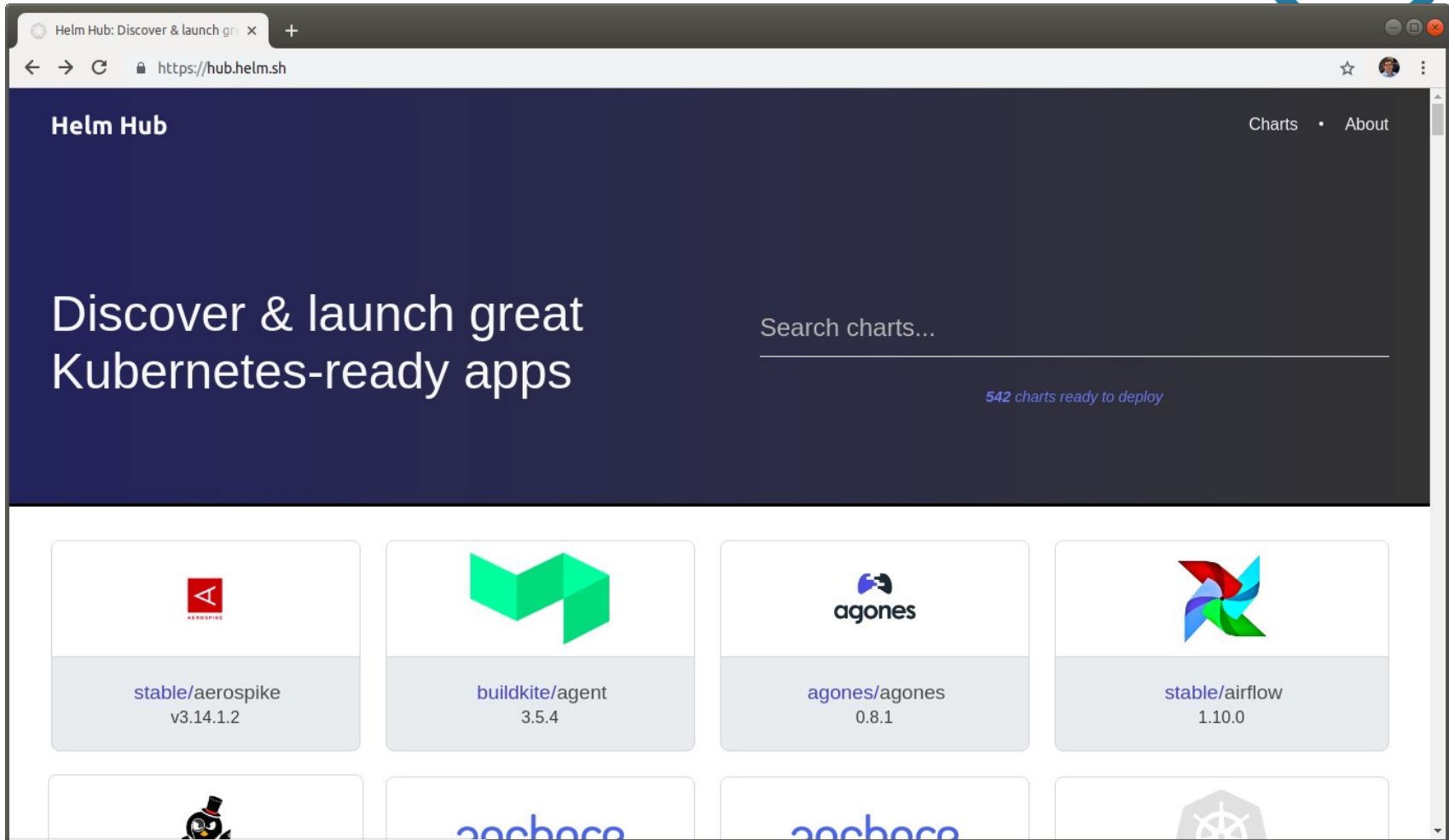
Charts are easy to version, share, and host on public or private servers.



Rollbacks

Use `helm rollback` to roll back to an older version of a release with ease.

Helm



<https://hub.helm.sh/>

Helm



- **Instalación**

- **Cliente Helm**

- Binario que se instala en la máquina del developer
 - Permite instalar, actualizar y borrar los charts en el cluster kubernetes

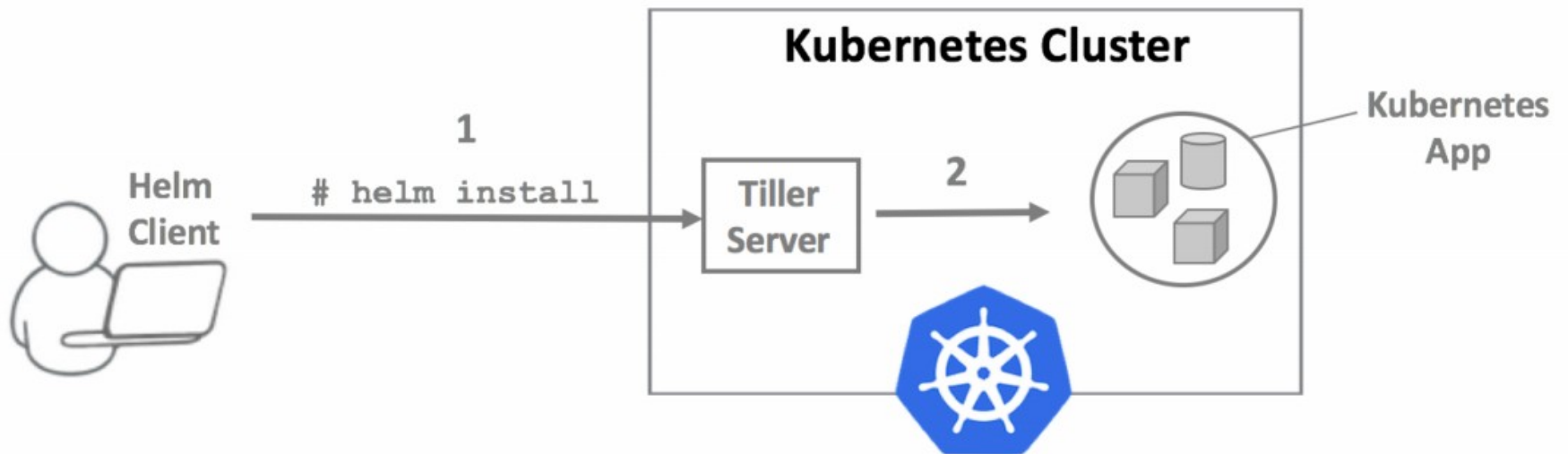
- **Servidor Tiller:**

- Pod que se ejecuta en el cluster
 - Encargado de controlar los pods desplegados en el cluster

Helm



- Arquitectura



Helm



- **Instalación del cliente**

- Tener kubectl conectado a un cluster
- Descargar el binario

<https://github.com/kubernetes/helm/releases>

- Mover el binario a una ruta del path

```
$ helm help
```

https://docs.helm.sh/using_helm/#installing-helm

Helm



- **Desplegar Tiller**

- Se instala Tiller en el cluster conectado a kubectl

```
$ helm init
```

- Se crea un pod en el namespace kube-system, donde están los pods de management de kubernetes

```
$ kubectl --namespace=kube-system get pods
```

```
$ kubectl --namespace=kube-system get pods --selector app=helm
```

https://docs.helm.sh/using_helm/#installing-helm

Helm



- Dar permisos a Tiller

```
$ kubectl -n kube-system patch deployment tiller-deploy \
-p '{"spec": {"template": \
{"spec":{"automountServiceAccountToken": true}}}}'
```

```
$ kubectl create clusterrolebinding \
add-on-cluster-admin --clusterrole=cluster-admin \
--serviceaccount=kube-system:default
```

https://docs.helm.sh/using_helm/#installing-helm

Helm



- **Instalación de charts del HelmHub**

- Se indica el nombre del chart (en el repositorio) y el nombre del **despliegue (release)** en el cluster

```
$ helm install --name main-web stable/drupal
```

- Se puede desplegar el mismo chart varias veces con diferentes nombres

<https://hub.helm.sh/charts/stable/drupal>

<https://github.com/kubernetes/charts/tree/master/stable/drupal>



- Instalación de charts del HelmHub
- Al instalar un chart aparece información básica del despliegue

1. Get the Drupal URL:

NOTE: It may take a few minutes for the LoadBalancer IP to be available.
Watch the status with: 'kubectl get svc --namespace default -w main-web-drupal'

```
export SERVICE_IP=$(kubectl get svc --namespace default main-web-drupal --template "{{ range (index .status.loadBalancer.ingress 0) }}{{.}}{{ end }}")
echo "Drupal URL: http://$SERVICE_IP/"
```

2. Login with the following credentials

```
echo Username: user
echo Password: $(kubectl get secret --namespace default main-web-drupal -o jsonpath="{.data.drupal-password}" | base64 --decode)
```

```
$ minikube service main-web-drupal
```

Helm



- **Gestión de releases**

- Ver el estado de una release

```
$ helm status main-web
```

- Listar las releases del cluster

```
$ helm list
```

```
$ helm list --all
```

- Borrar una release

```
$ helm delete main-web
```



- **Personalizar un chart en la instalación**
 - Revisamos la documentación para ver las opciones de configuración

<https://hub.helm.sh/charts/stable/drupal>

- Configuración por fichero o por línea de comandos

```
$ echo '{drupalPassword: pass}' > config.yaml  
$ helm install --f config.yaml stable/drupal
```

```
$ echo '{drupalPassword: pass}' > config.yaml  
$ helm install --values config.yaml stable/drupal
```

```
$ helm install --set drupalPassword=pass stable/drupal
```

Helm



- Personalizar un chart en la instalación
 - Podemos consultar los valores usados

```
$ helm install --set drupalPassword=pass \  
--name other-web stable/drupal
```

```
$ helm get values other-web
```

```
drupalPassword: pass
```

Helm



- **Actualización de releases**

- Se puede actualizar a la última versión

```
$ helm upgrade main-web stable/drupal
```

- O a una versión específica

```
$ helm upgrade main-web stable/drupal \
  --version 3.0.8
```

https://docs.helm.sh/using_helm/#helm-upgrade-and-helm-rollback-upgrading-a-release-and-recovering-on-failure

Helm



- **Actualización de releases**

- Cada release (deploy) se numera con 1,2,3...
- Si algo sale mal en la actualización, se puede volver a crear la release anterior

```
$ helm rollback other-web 1
```

https://docs.helm.sh/using_helm/#helm-upgrade-and-helm-rollback-upgrading-a-release-and-recovering-on-failure



- Cambio de configuración de releases
 - Se puede cambiar la configuración con **upgrade**
 - Para evitar la actualización, se indica como versión la versión actual

```
$ helm list
```

| NAME | REVISION | UPDATED | STATUS | CHART | APP VERSION | NAMESPACE |
|-----------|----------|-------------------------|----------|--------------|-------------|-----------|
| other-web | 2 | Thu Mar 7 02:50:36 2019 | DEPLOYED | drupal-3.0.5 | 8.6.8 | default |

```
$ helm upgrade -f config.yaml \
  main-web stable/drupal \
  --version 3.0.5
```

https://docs.helm.sh/using_helm/#helm-upgrade-and-helm-rollback-upgrading-a-release-and-recovering-on-failure

Helm



- Creación de charts
- Un char es una **colección de ficheros** que describen un conjunto de recursos Kubernetes relacionados
- Un chart es una **estructura de directorios** que puede ser empaquetada (en un fichero comprimido)

https://docs.helm.sh/developing_charts/

Helm



- Creación de charts
 - La carpeta raíz es el nombre del chart

```
wordpress/  
  Chart.yaml          # A YAML file containing information about the chart  
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart  
  README.md           # OPTIONAL: A human-readable README file  
  requirements.yaml   # OPTIONAL: A YAML file listing dependencies for the chart  
  values.yaml         # The default configuration values for this chart  
  charts/             # A directory containing any charts upon which this chart depends.  
  templates/          # A directory of templates that, when combined with values,  
                      # will generate valid Kubernetes manifest files.  
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

https://docs.helm.sh/developing_charts/

Helm



- Creación de charts
 - Se puede generar un layout inicial para editar

```
$ helm create mychart
```

```
mychart
|-- Chart.yaml
|-- charts
|-- templates
|   |-- NOTES.txt
|   |-- _helpers.tpl
|   |-- deployment.yaml
|   |-- ingress.yaml
|   `-- service.yaml
`-- values.yaml
```

<https://docs.bitnami.com/kubernetes/how-to/create-your-first-helm-chart/>

Helm



- Templates

```
apiVersion: v1
kind: Service
metadata:
  name: {{ template "fullname" . }}
  labels:
    chart: "{{ .Chart.Name }}-{{ .Chart.Version | replace "+" "_" }}"
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.externalPort }}
      targetPort: {{ .Values.service.internalPort }}
      protocol: TCP
      name: {{ .Values.service.name }}
  selector:
    app: {{ template "fullname" . }}
```

Helm



- Creación de charts

- Carpeta templates tiene los recursos kubernetes
- Se llama templates porque los ficheros son **plantillas Go** porque al instalar el chart se sustituyen las variables por valores
- Estos valores se obtienen del fichero **Chart.yaml** y **values.yaml**
- Se puede depurar la sustitución con

```
$ helm install --dry-run --debug ./mychart
```

<https://golang.org/pkg/text/template/>

Helm



- Creación de charts

- El fichero values.yaml contiene valores por defecto
- Se pueden sustituir en el momento del despliegue por línea de comandos o por fichero

```
$ helm install --dry-run --debug ./mychart \
  --set service.internalPort=8080
```

```
$ helm install --dry-run --debug ./mychart \
  --values config.yaml
```

<https://docs.bitnami.com/kubernetes/how-to/create-your-first-helm-chart/>

Helm



- Creación de charts
 - El contenido de **NOTES.txt** se muestra después de instalar el chart (es una plantilla también)
 - El fichero **Chart.yaml** contiene los metadatos de la aplicación

<https://github.com/kubernetes/helm/blob/master/docs/charts.md#the-chartyaml-file>

Helm



- Chart.yaml

```
apiVersion: The chart API version, always "v1" (required)
name: The name of the chart (required)
version: A SemVer 2 version (required)
kubeVersion: A SemVer range of compatible Kubernetes versions (optional)
description: A single-sentence description of this project (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this project's home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
maintainers: # (optional)
  - name: The maintainer's name (required for each maintainer)
    email: The maintainer's email (optional for each maintainer)
    url: A URL for the maintainer (optional for each maintainer)
engine: gotpl # The name of the template engine (optional, defaults to gotpl)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
appVersion: The version of the app that this contains (optional). This needn't be SemVer.
deprecated: Whether this chart is deprecated (optional, boolean)
tillerVersion: The version of Tiller that this chart requires. This should be expressed as a SemVer range: ">2.0.0"
```

<https://github.com/kubernetes/helm/blob/master/docs/charts.md#the-chartyaml-file>

Helm



- Instalar un chart “en desarrollo”
- El comando install permite apuntar a la carpeta del chart

```
$ helm install --name example ./mychart \
  --set service.type=LoadBalancer
```

```
$ kubectl describe service example-mychart
```

Helm



- Empaquetar aplicaciones
 - La carpeta del chart se puede empaquetar

```
$ helm package ./mychart
```

- Se genera un fichero **mychart-0.1.0.tgz**
- Se puede instalar en un k8s

```
$ helm install --name example3 mychart-0.1.0.tgz
```

<https://github.com/kubernetes/helm/blob/master/docs/charts.md#the-chartyaml-file>

Helm



- **Helm Hub**

- No podemos subir charts a Helm Hub de forma sencilla (como en DockerHub)
- Tenemos que solicitar incluir un chart como un PR a un repositorio GitHub

<https://github.com/helm/hub/blob/master/Repositories.md>

- Lo usual es tener un repositorio privado

Helm



- Repositorio privado



CHARTMUSEUM

```
$ docker run --rm -it -p 8080:8080 \  
-v $(pwd)/charts:/charts -e DEBUG=true \  
-e STORAGE=local -e STORAGE_LOCAL_ROOTDIR=/charts \  
chartmuseum/chartmuseum:v0.8.1
```

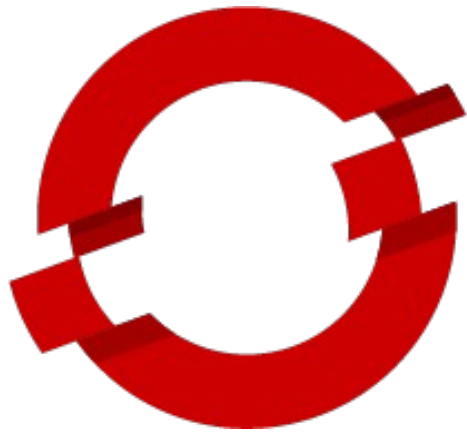
<https://chartmuseum.com/>

Ejercicio 5

- Crea un chart para la web de anuncios con base de datos (ejercicio 4)
- Instala la aplicación en minikube

Solución

```
$ helm install --name java-bd ./ejercicio5/java-bd
```



RED HAT[®]
OPENSIFT

OpenShift

- Plataforma de RedHat implementada sobre **Kubernetes**
- Algunos recursos son ligeramente diferentes a los equivalentes de Kubernetes
 - Deployment > DeploymentConfig
 - Ingress > Router
 - Namespace > Project
- Dispone de un dashboard gráfico más cuidado




<https://cloudowski.com/articles/10-differences-between-openshift-and-kubernetes/>

OpenShift

The screenshot displays the OpenShift Origin 'Browse Catalog' page. The top navigation bar includes the 'OPENS SHIFT ORIGIN' logo, a search bar labeled 'Search Catalog', and user controls for help and a user named 'developer'. Below the navigation bar, the 'Browse Catalog' section features tabs for 'All', 'Languages', 'Databases', 'Middleware', 'CI/CD', and 'Other'. A filter dropdown shows '20 Items'. The main content area is a grid of application templates, each with an icon and a name: .NET Core Builder Images, Apache HTTP Server (httpd), CakePHP + MySQL, Dancer + MySQL, Django + PostgreSQL, Jenkins, MariaDB, MongoDB, MySQL, Nginx HTTP server and a reverse proxy (nginx), Node.js, Node.js + MongoDB, Perl, PHP, and Pipeline Build Example. The right sidebar contains a 'My Projects' section with a '+ Create Project' button, showing '1 of 1 Projects' and a 'View All' link. Below this is a 'Getting Started' section with a 'Take Home Page Tour' button and a list of links: Documentation, Interactive Learning Portal, Local Development, YouTube, and Blog. The 'Recently Viewed' section shows a user profile icon and the 'Pipeline Build Example' project.

<https://cloudowski.com/articles/10-differences-between-openshift-and-kubernetes/>

Versiões comerciais

|  | |  | |  | |
|--|--|--|--|---|--|
| STARTER | | ENTERPRISE | | ENTERPRISE | |
| Free | | <i>Contact sales for pricing</i> | | <i>Contact sales for pricing</i> | |
| For individual learning and experimenting. | | For enterprise teams and business-critical applications. | | For enterprise teams and business-critical applications. | |
| <ul style="list-style-type: none">• Public cloud• 1 Project• Community support | | <ul style="list-style-type: none">• Virtual private cloud• Unlimited projects• Premium support | | <ul style="list-style-type: none">• Any infrastructure• Customizable, with full administrative control• Standard or premium support | |
| ✓ Red Hat hosted | | ✓ Red Hat hosted | | ✓ Red Hat hosted | |

OpenShift

Versión open source

The logo for 'okd' is displayed on a dark blue background. The letter 'o' is a solid red circle, while the letters 'kd' are in a white, sans-serif font.

The Origin Community Distribution of Kubernetes
that powers **Red Hat OpenShift**.

<https://www.okd.io/>

OpenShift

- OpenShift está disponible para los empleados de Telefónica

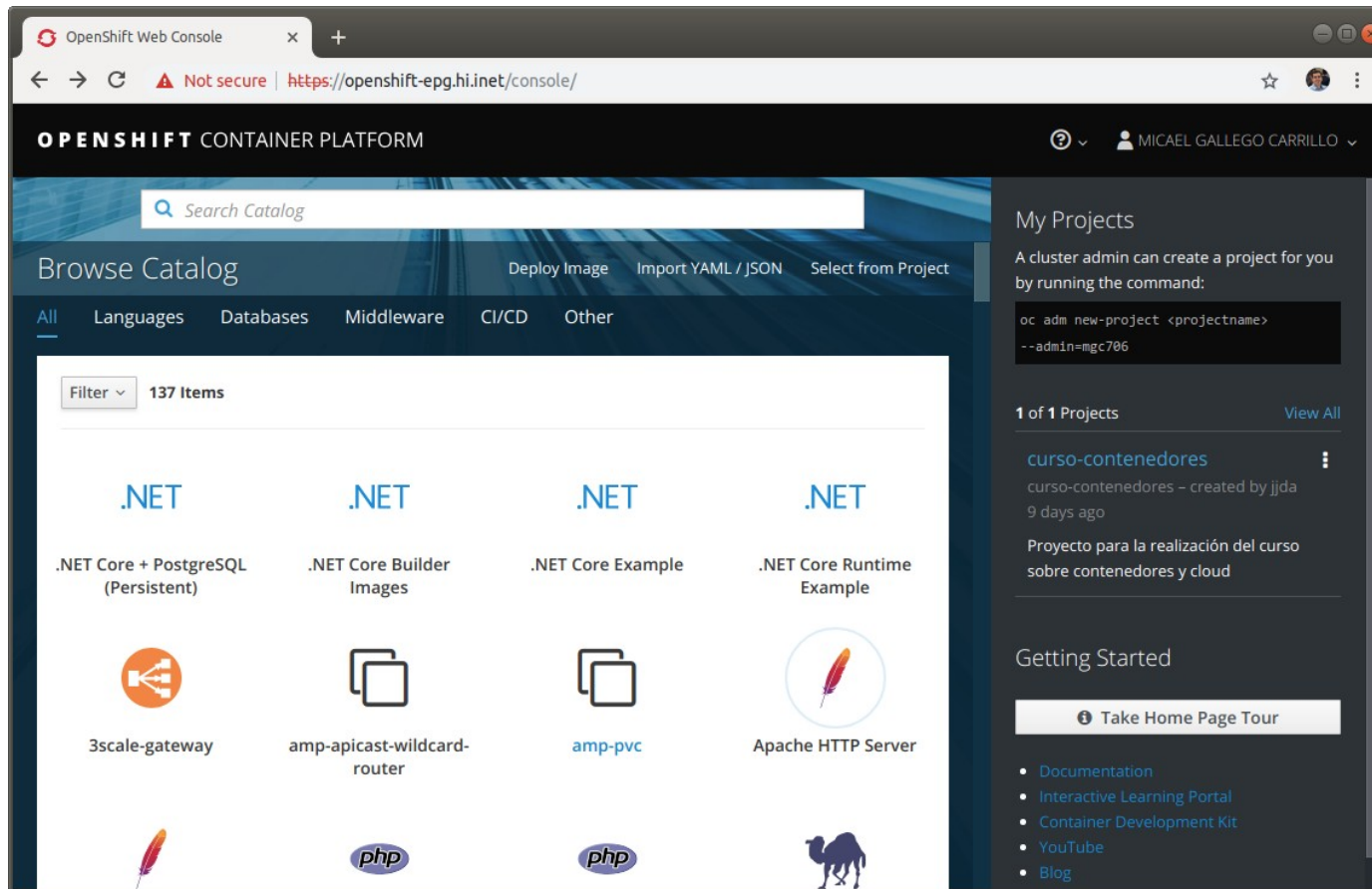
<https://openshift-epg.hi.inet/>

- Versión 3.7.44
- Basada en Kubernetes: 1.7.6 (Junio 2017)

<https://docs.openshift.com/container-platform/3.7/welcome/index.html>

OpenShift

- Creamos un proyecto para trabajar en él



OpenShift

- Proyecto vacío

The screenshot shows the OpenShift Container Platform interface. At the top, a dark header bar contains the text 'OPENSIFT CONTAINER PLATFORM' (note the typo in the image), a notification bell icon, a help icon, and the user name 'MICAEL GALLEGU CARRILLO'. Below the header, a dark sidebar on the left lists navigation options: Overview, Applications, Builds, Resources, Storage, and Monitoring. The main content area has a dark top bar with the project name 'curso-contenedores' and an 'Add to Project' button. The main area is white and contains the text 'Get started with your project.' followed by instructions: 'Use your source or an example repository to build an application image, or add components like databases and message queues.' A blue 'Browse Catalog' button is centered below the text.

OPENSIFT CONTAINER PLATFORM

curso-contenedores

Add to Project

Overview

Applications

Builds

Resources

Storage

Monitoring

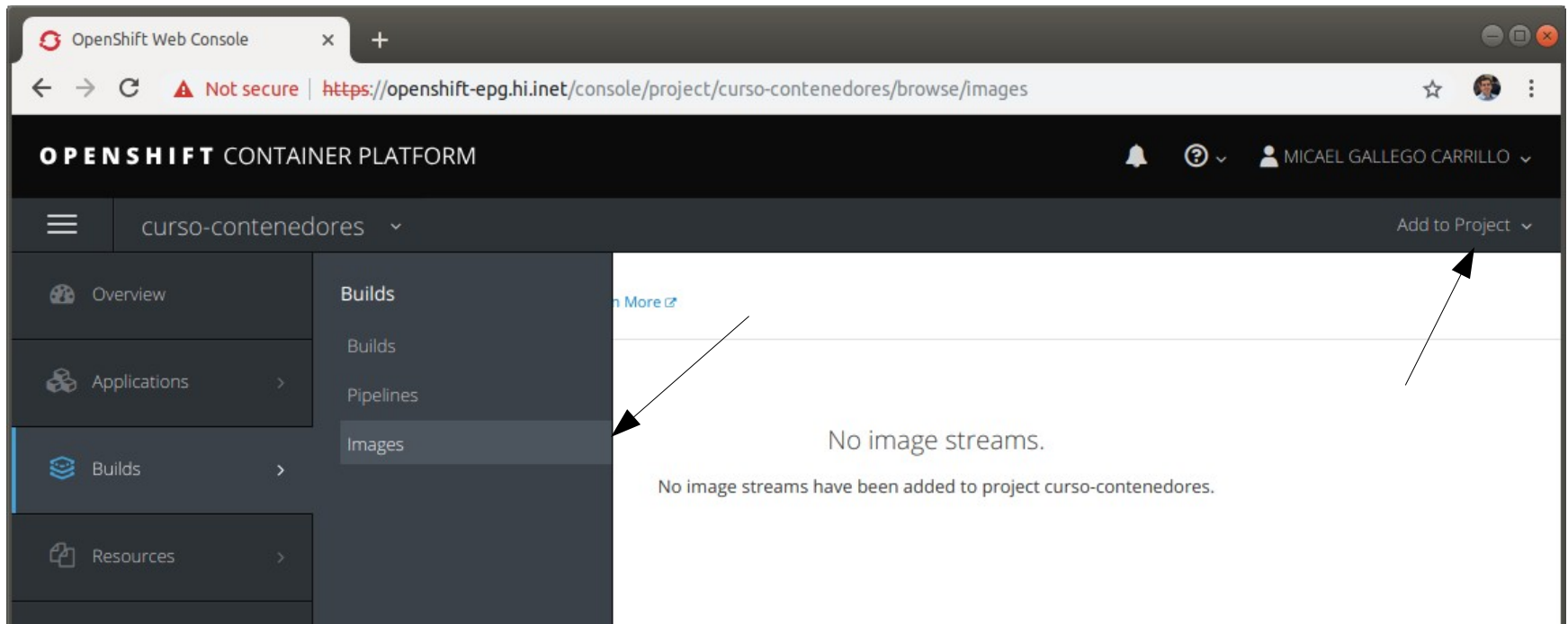
Get started with your project.

Use your source or an example repository to build an application image, or add components like databases and message queues.

Browse Catalog

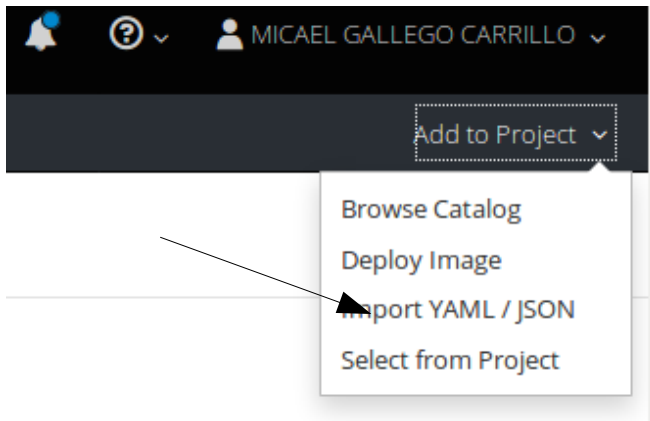
OpenShift

- Los recursos se pueden crear desde la interfaz
- Creamos un recurso **ImageStream** en Builds → Images



OpenShift

- Importamos el fichero YAML



```
apiVersion: v1
kind: ImageStream
metadata:
  name: webgatos
spec:
  dockerImageRepository: codeurjc/webgatos
```

Image Streams [Learn More](#)

Filter by label

Add

| Name | Docker Repo | Tags | Updated |
|-----------------------------|---|------------|-------------------|
| hpa-example | docker-registry.default.svc:5000/curso-contenedores/hpa-example | latest, v1 | a few seconds ago |

OpenShift

- Seguimos creando recursos siguiendo el mismo procedimiento que con el ImageStream
 - DeploymentConfig (similar al Deployment)
 - Service
 - Route (Similar al Ingress)

OpenShift

- DeploymentConfig

OPENSIFT CONTAINER PLATFORM

curso-contenedores

Add to Project

Overview

Applications

Builds

Resources

Storage

Deployments [Learn More](#)

Filter by label

Add

| Name | Last Version | Status | Created | Trigger |
|--------------------------|--------------|-------------------|-------------|---------------|
| webgatos | #1 | Active, 1 replica | an hour ago | Config change |

<https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/OpenShift/Spec/WebGatos/deployConfig.yaml>

OpenShift

- Service

OPENSIFT CONTAINER PLATFORM

curso-contenedores

Add to Project

Overview

Applications

Builds

Resources

Storage

Monitoring

Services

Learn More

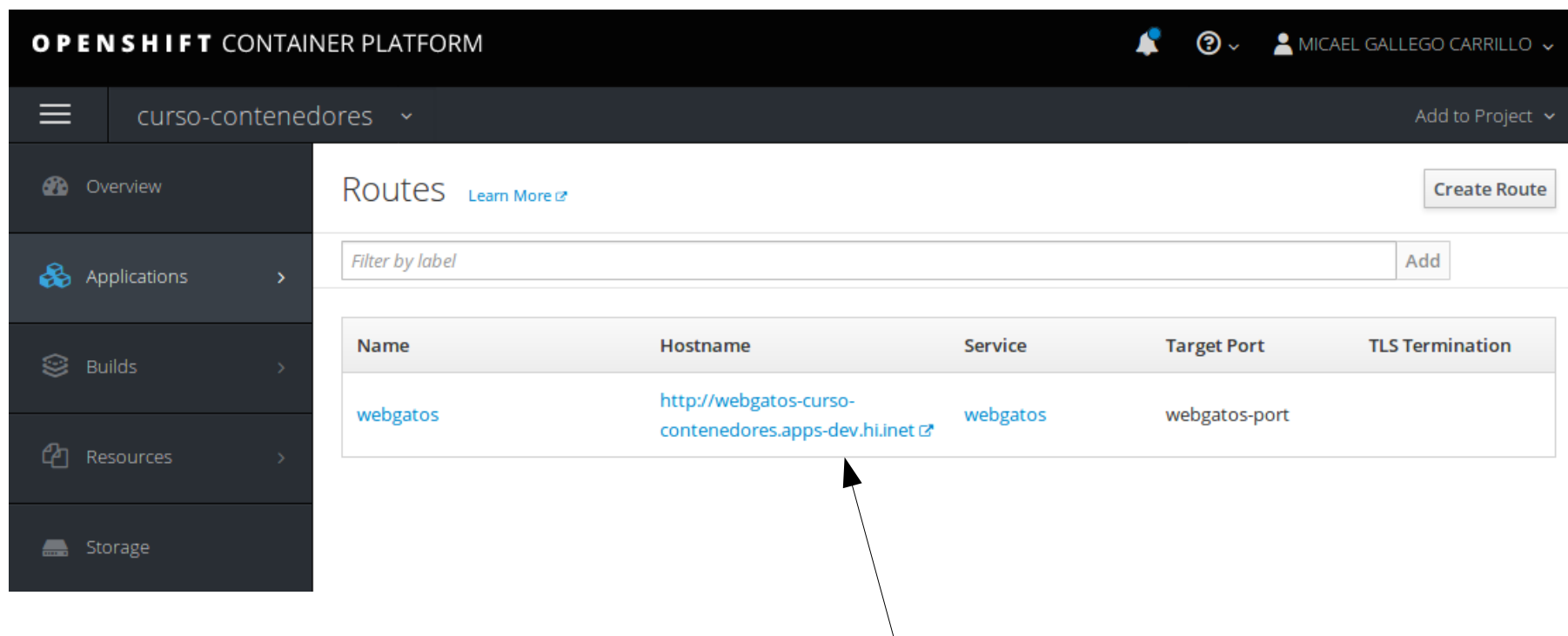
Filter by label

Add

| Name | Cluster IP | External IP | Ports | Selector | Age |
|----------|---------------|-------------|----------|---------------|---------|
| webgatos | 172.30.78.252 | none | 5000/TCP | name=webgatos | an hour |

<https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/OpenShift/Spec/WebGatos/service.yaml>

- Route



OPENSIFT CONTAINER PLATFORM

curso-contenedores

Overview Applications Builds Resources Storage

Routes [Learn More](#) [Create Route](#)

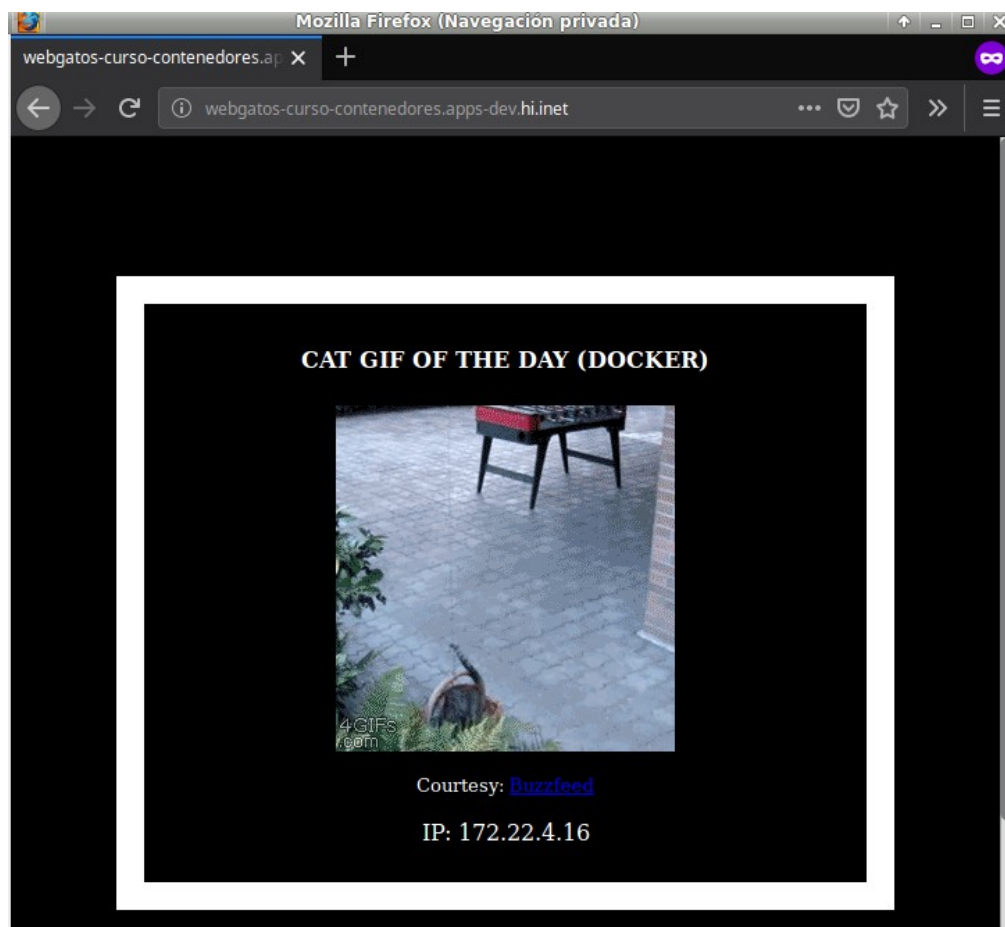
Filter by label Add

| Name | Hostname | Service | Target Port | TLS Termination |
|----------|---|----------|---------------|-----------------|
| webgatos | http://webgatos-curso-contenedores.apps-dev.hi.inet | webgatos | webgatos-port | |

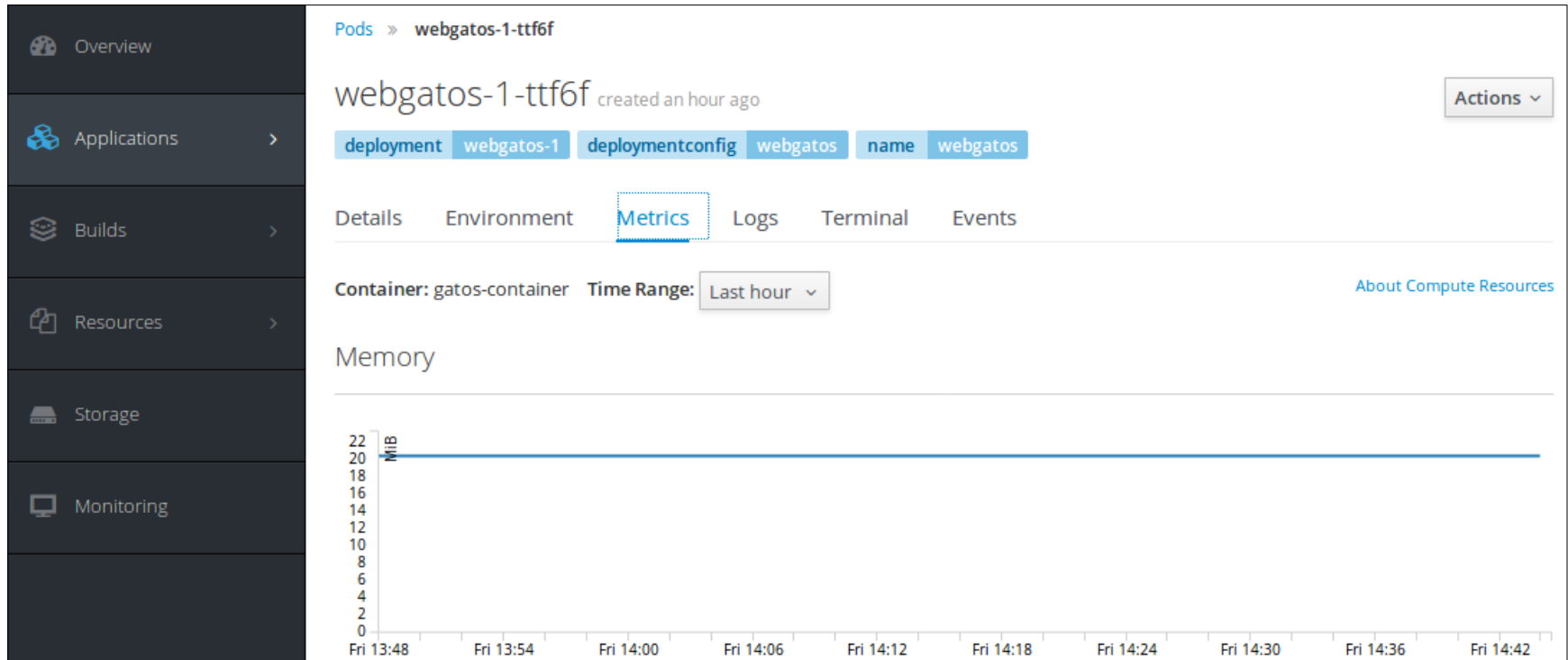
<https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/OpenShift/Spec/WebGatos/route.yaml>

OpenShift

- Acceso a la URL en la que se publica el servicio



- Métricas del pod
 - Hay que aceptar el certificado



- Escalado del pod
- Si entramos en el deployment podemos ver la replicas del pod y escalarlas

Deployments » webgatos » #1


webgatos-1 created 2 hours ago

[openshift.io/deployment-config.name](#) webgatos

[Details](#) Environment Metrics Logs Events

| | |
|---------------------------|---|
| Status: | 🔄 Active |
| Deployment Config: | webgatos |
| Status Reason: | config change |
| Selectors: | deployment=webgatos-1 deploymentconfig=webgatos name=webgatos |
| Replicas: | 1 current / 1 desired ✎ |

Template



- Escalado del pod

Deployments » webgatos » #1


webgatos-1 created 2 hours ago

[openshift.io/deployment-config.name](#) webgatos

[Details](#) Environment Metrics Logs Events

| | |
|---------------------------|---|
| Status: | 🔄 Active |
| Deployment Config: | webgatos |
| Status Reason: | config change |
| Selectors: | deployment=webgatos-1 deploymentconfig=webgatos name=webgatos |
| Replicas: | 5 current / 5 desired |

Template



- **Balanceo de carga**

- Dispone de algún tipo de *stiky session* y en el browser no balancea

```
$ for i in $(seq 1 10); do curl -sS \  
  http://webgatos-curso-contenedores.apps-dev.hi.inet/ \  
  | grep "IP: "; done  
  <p>IP: 172.22.3.104</p>  
  <p>IP: 172.22.3.105</p>  
  <p>IP: 172.22.3.105</p>  
  <p>IP: 172.22.3.105</p>  
  <p>IP: 172.22.4.16</p>  
  <p>IP: 172.22.4.16</p>  
  <p>IP: 172.22.4.16</p>
```

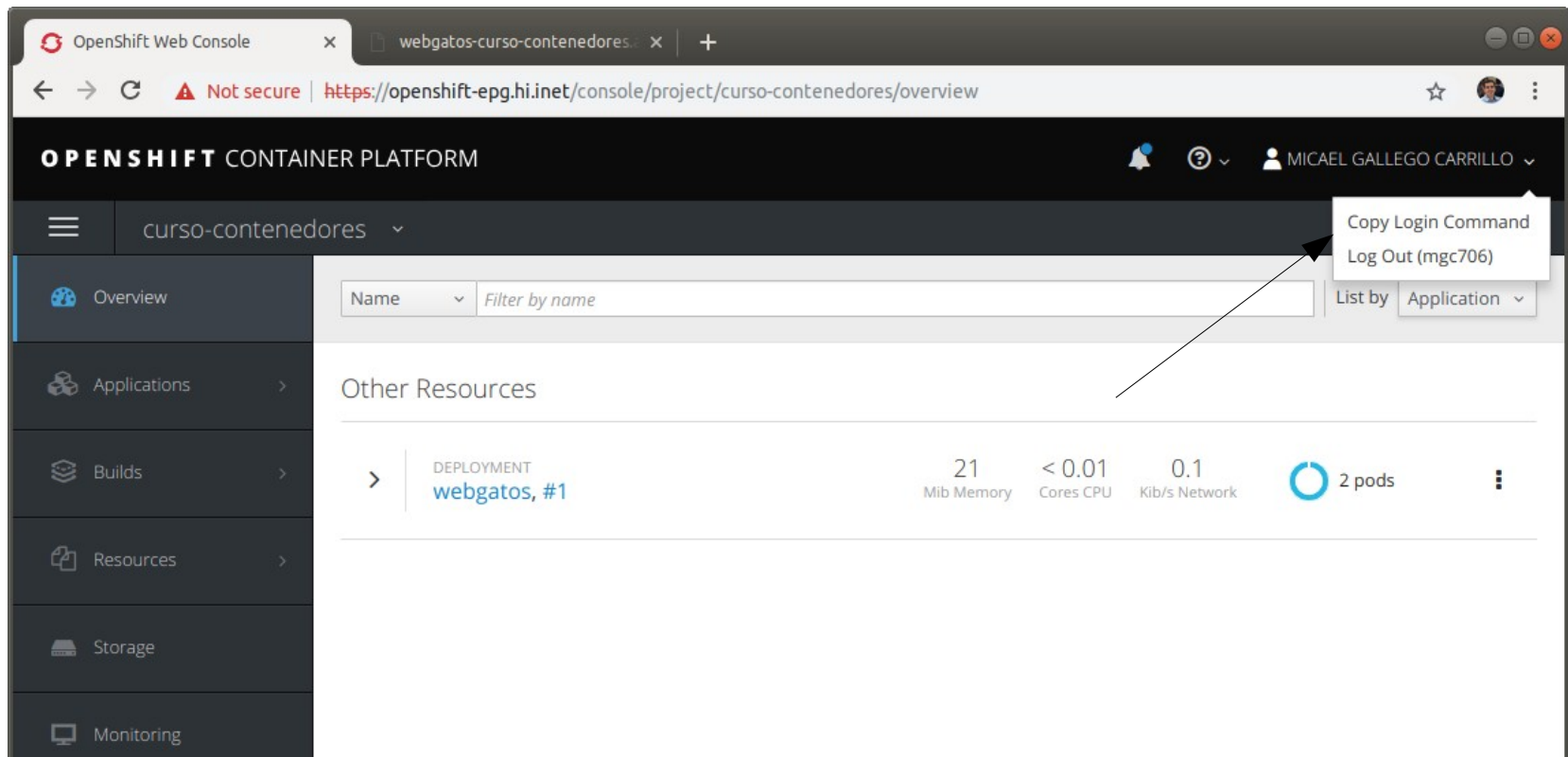
OpenShift

- **Control desde la consola**
 - **kubectl:** Operaciones de Kubernetes
 - **oc:** Operaciones propias de OpenShift y las de Kubernetes
 - Como es una versión **antigua** de Kubernetes, se ha instalado en un contenedor para evitar conflictos

```
$ docker run -ti --rm --name openshift-cli \
-v ${PWD}:/wd -w /wd \
codeurjc/openshift-cli:kubectl-1.7-oc-1.5.1
```

OpenShift

- Desde la web de OpenShift copiamos el comando de **configuración del cliente**



OpenShift

- Dentro del contenedor podemos ejecutar los mismos comandos que hemos visto con `kubectl`
- También podemos usar `oc` para opciones específicas de OpenShift y todas las de Kubernetes
 - `oc get svc`
 - `oc get deployments`
 - `oc get pods`
 - ...

- Despliegue de aplicaciones

```
$ oc create \  
  -f OpenShift/Spec/Anuncios/mysql-full.yaml \  
  -f OpenShift/Spec/Anuncios/anuncios-full.yaml
```

```
service "db" created  
persistentvolumeclaim "mysql-pvc" created  
deploymentconfig "mysql" created  
imagestream "mysql" created  
service "anuncios" created  
deploymentconfig "anuncios" created  
route "anuncios" created  
imagestream "java-webapp-bbdd" created
```

- Rutas para acceder al servicio (Ingress)

```
$ oc get routes
```

| NAME | HOST/PORT | PATH | SERVICES | PORT | TERMINATION | WILDCARD |
|----------|--|------|----------|---------------|-------------|----------|
| anuncios | anuncios-curso-contenedores.apps-dev.hi.inet | | anuncios | anuncios-port | | None |

OpenShift

- Accedemos al servicio usando un browser

