

Consultoría Formación

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

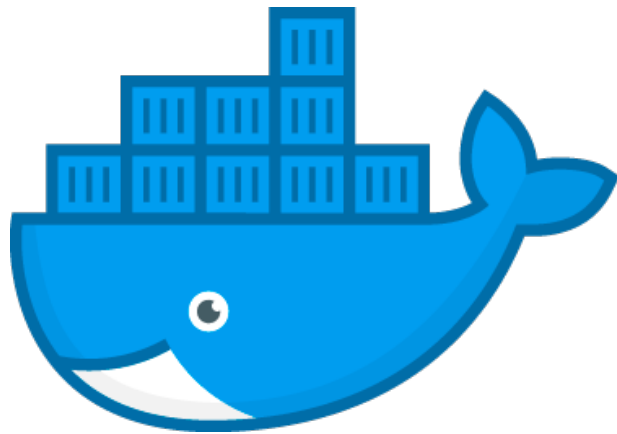
Master en Desarrollo y Despliegue de Aplicaciones en la Nube

Online / Clases en directo

Septiembre 2019



kubernetes



docker®

Docker



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

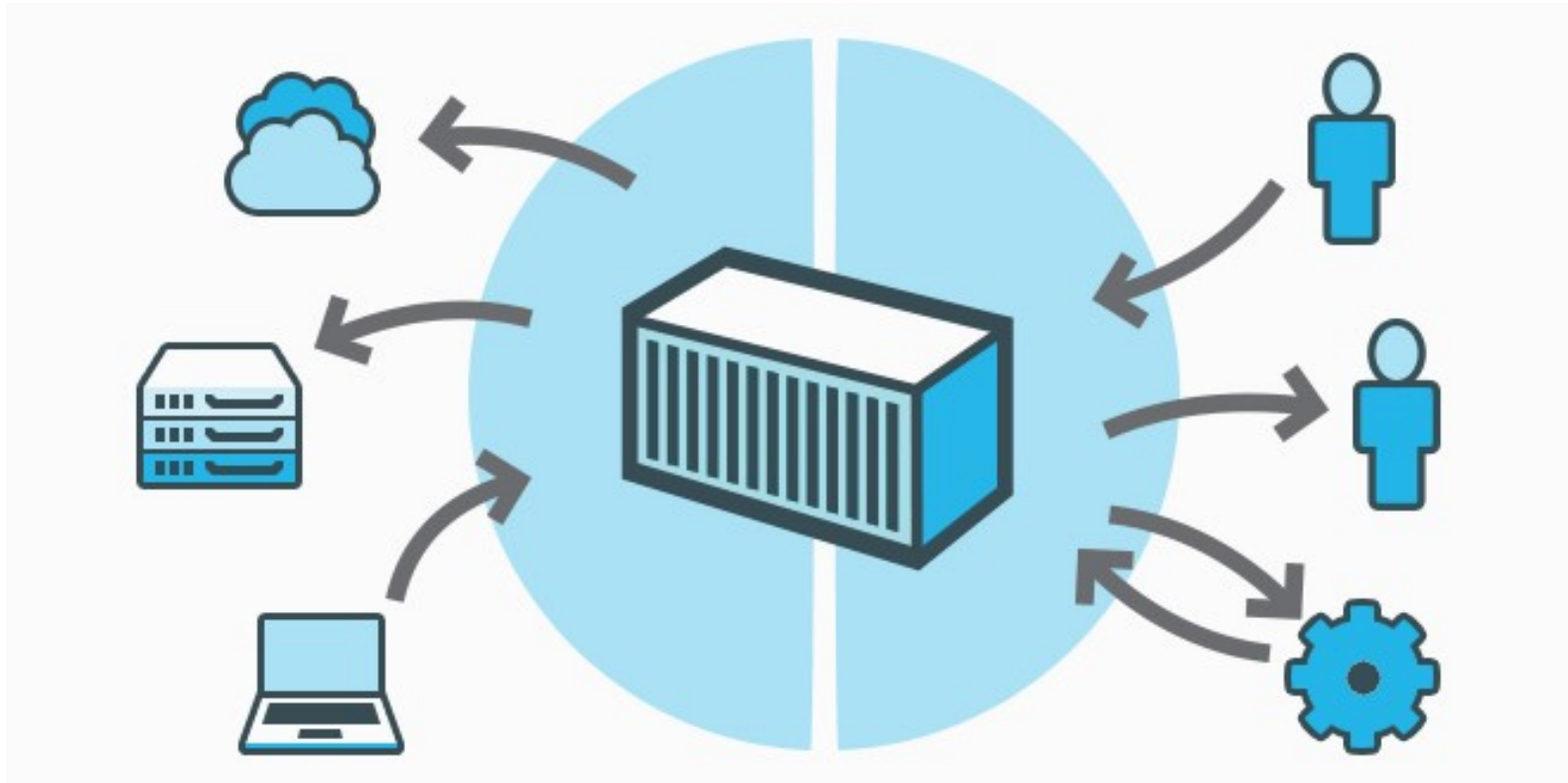
Docker



- 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

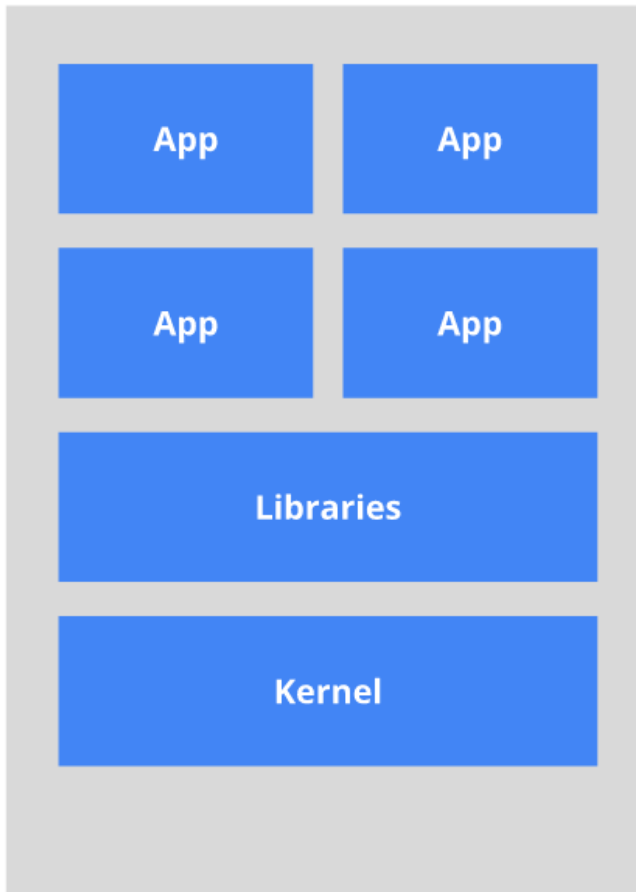
<https://www.docker.com/>

Docker



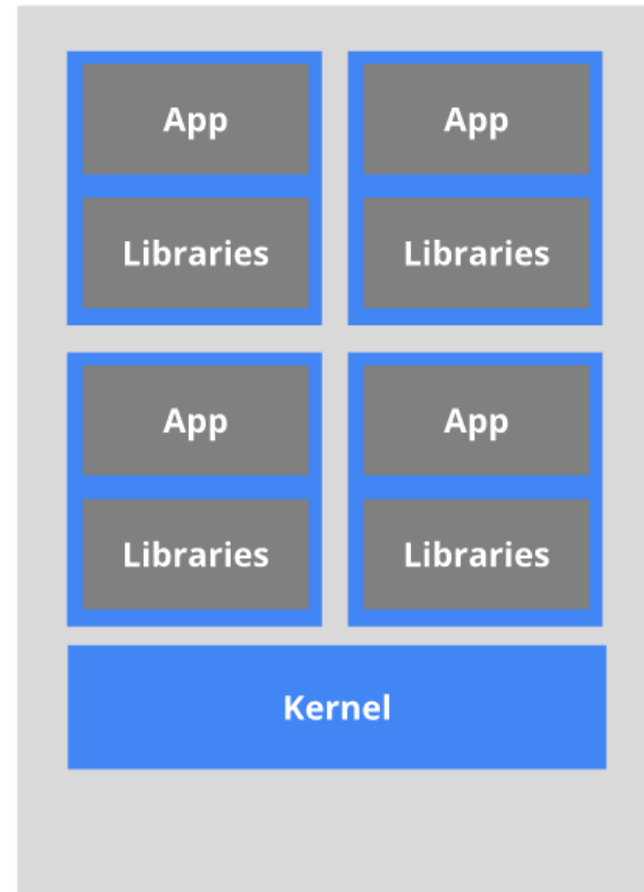
Docker

The old way: Applications on host



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

The new way: Deploy containers



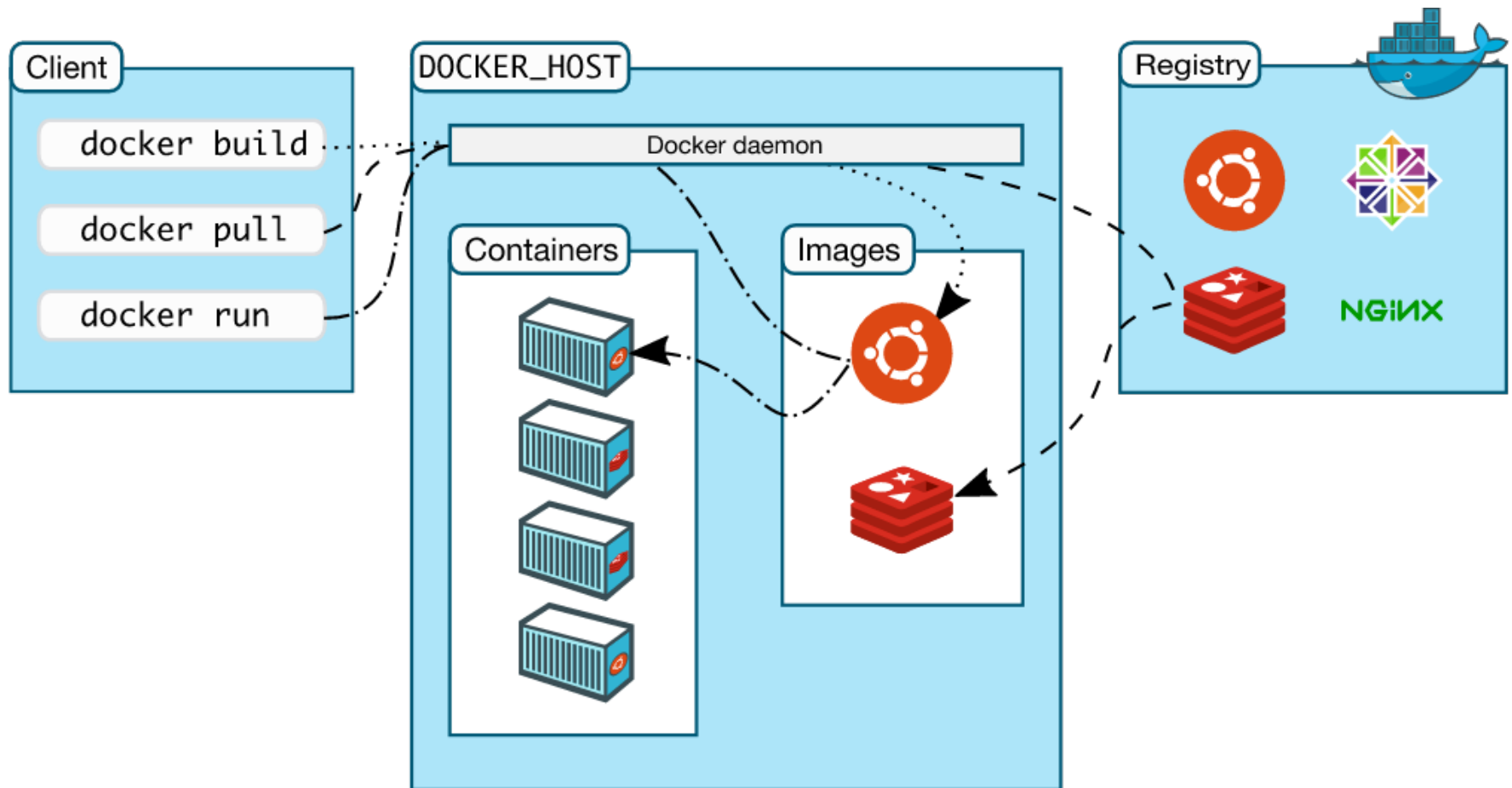
*Small and fast, portable
Uses OS-level virtualization*

- ¿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**

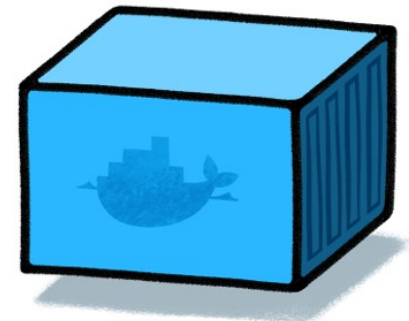
Conceptos

- 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**

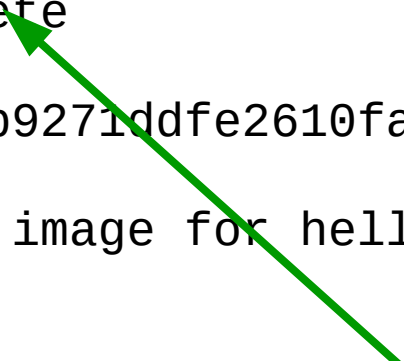


Ejecución de contenedores

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.
...
```

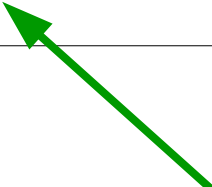


La primera vez la imagen se descarga

Ejecución de contenedores

Ejecutar “hello-world”

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



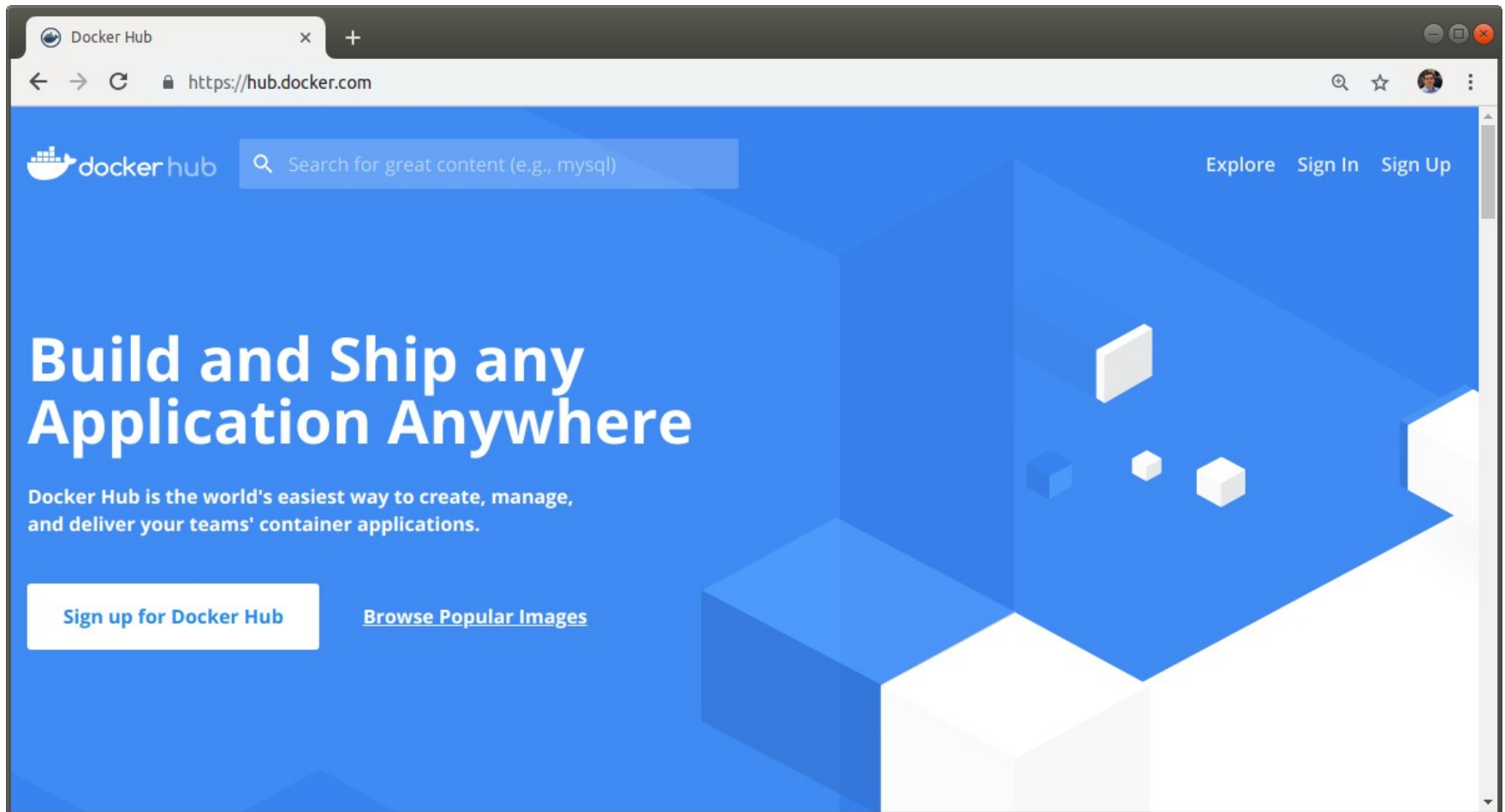
La segunda vez se
usa la vez la imagen
se descarga

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
```

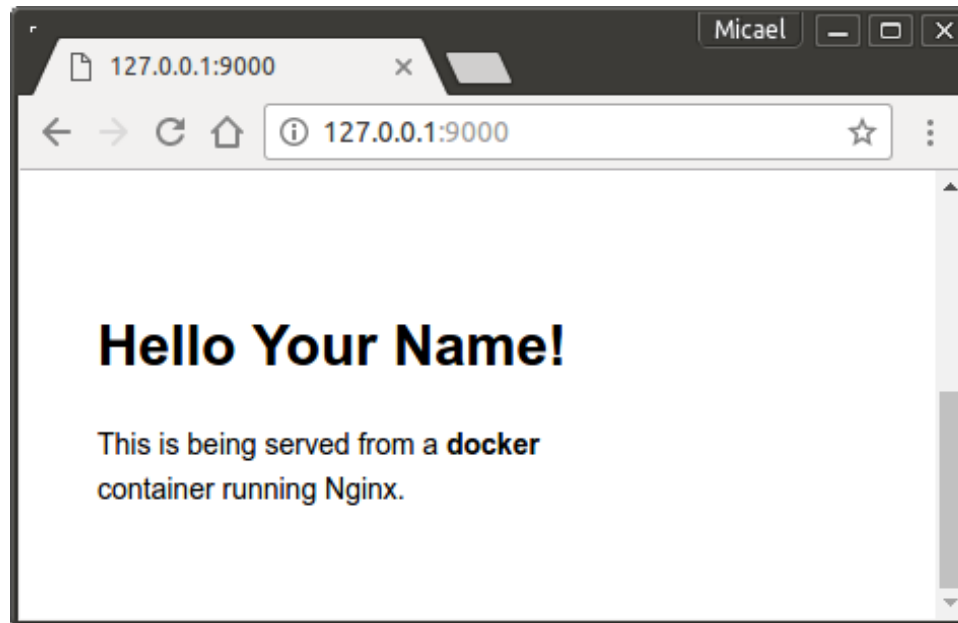
-p 9000:80

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	STATUS	PORTS	NAMES
a7a0e504ca3e	sequence/static-site	"/bin/sh -c 'cd /usr/'	Up 26 seconds		

28 seconds ago



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 **cluster** de nodos ofreciendo un control uniforme
 - 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®

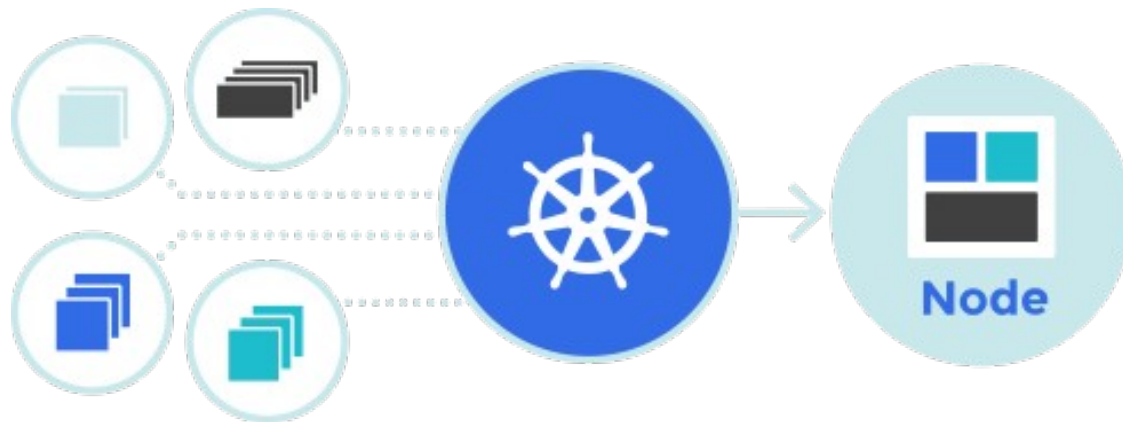

JD.COM

 redhat.

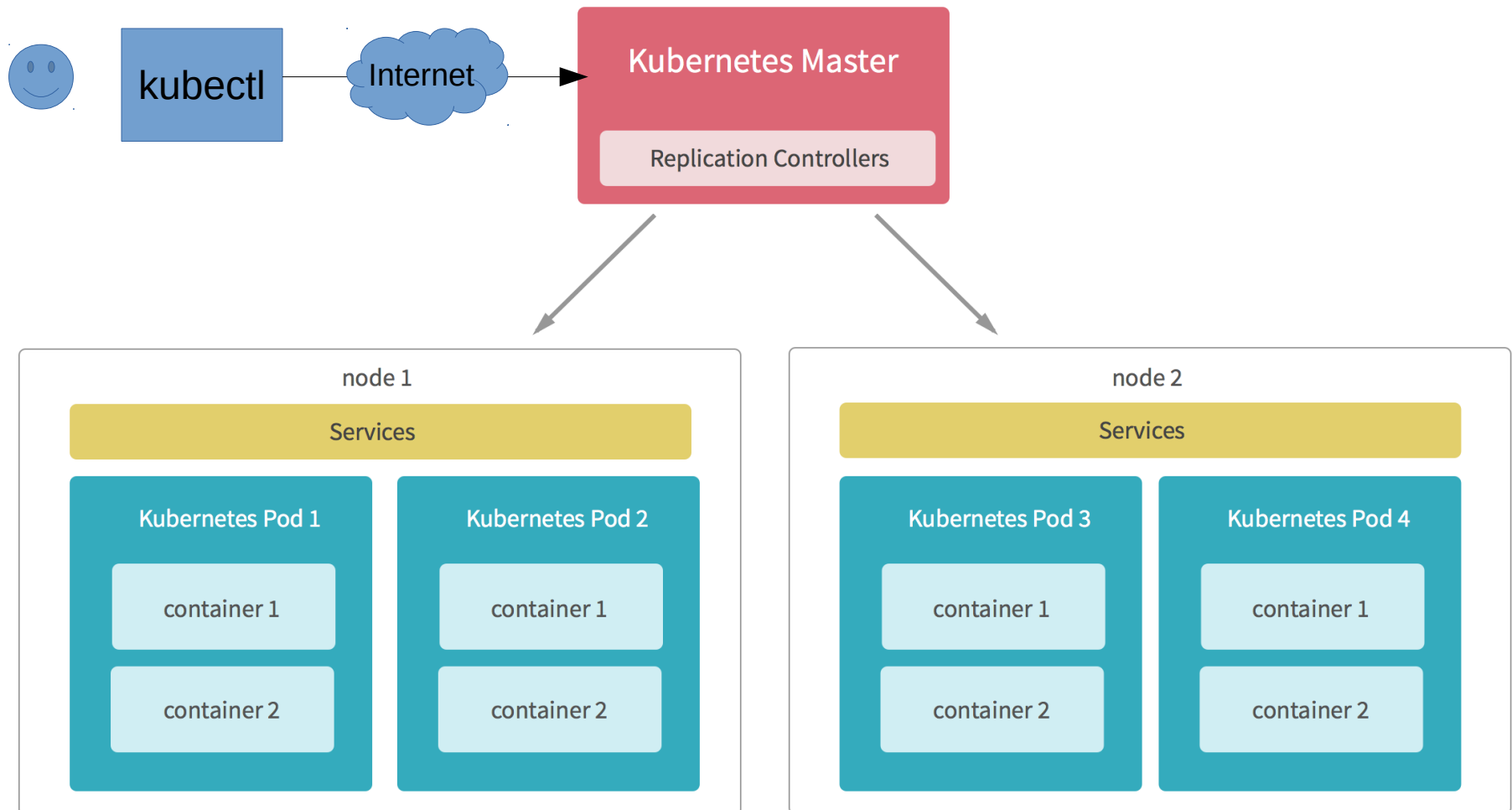
<https://www.cncf.io/about/members/>

Kubernetes

- Es un software diferente a docker que tiene que instalarse por separado
- Usa docker internamente, pero también puede funcionar con otras tecnologías de contenedores como rkt, pero no son muy maduras



Kubernetes



Cliente Kubernetes

- **Kubectl**

- Es una tool de línea de comandos (CLI) local para controlar un cluster Kubernetes
- Está implementada en Go, no es necesario ningún runtime o VM para que se pueda ejecutar
- Conceptualmente es como la línea de comandos de docker, pero las opciones son muy diferentes
- Instalación

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

Kubernetes en local

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



kube-spawn



minikube

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>

Kubernetes en local

- **Minikube**

- Instalación

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

Arrancar kubernetes (inicia una VM)

```
$ minikube start
```

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

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

Kubernetes en local

- **Minikube**

- Acceso por ssh al nodo
 - `$ minikube ssh`
- Acceso al dashboard gráfico web
 - `$ minikube dashboard`
- Parar minikube
 - `$ minikube stop`
- Borrar minikube
 - `$ minikube delete`

Cluster - Kubernetes x

Micael

192.168.99.100:30000/#!/cluster?namespace=default

Apps G Twitter Spines Now mongo Concurrency logstash + graf prensa RedmineK Nueva carpeta Other bookmarks

kubernetes

Search

+ CREATE

Cluster

Cluster

Namespaces

Namespaces

Nodes

Persistent Volumes

Roles

Storage Classes

Namespace

default

Overview

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Namespaces

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

Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
✓ minikube	beta.kubernetes.io/arch=amd64 beta.kubernetes.io/os=linux kubernetes.io/hostname=minikube node-role.kubernetes.io/master=	True	0.815 (40.75%)	0 (0.00%)	160 Mi (8.00%)	170 Mi (8.50%)	4 minutes

Roles

Name	Role Type	Namespace	Age
kubeadm:bootstrap-signer-clusterinfo	Role	kube-public	4 minutes

Kubernetes en producción

- Se ofrece como servicio (gestionado por el proveedor)



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



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



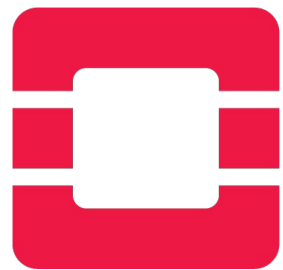
<https://aws.amazon.com/es/eks/>



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

Kubernetes en producción

- También se puede instalar en clusters físicos o en máquinas virtuales (OpenStack, VMWare...)

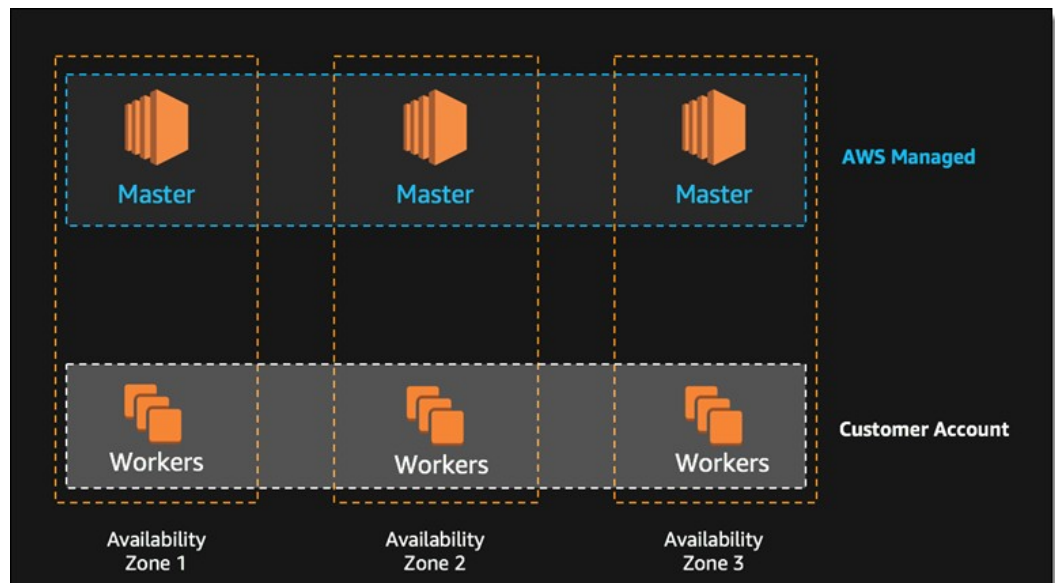


openstack®

vmware®

Kubernetes en AWS

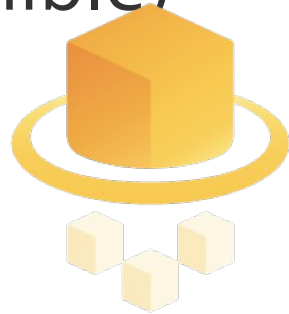
- AWS ofrece el servicio **AKS** como un Kubernetes gestionado



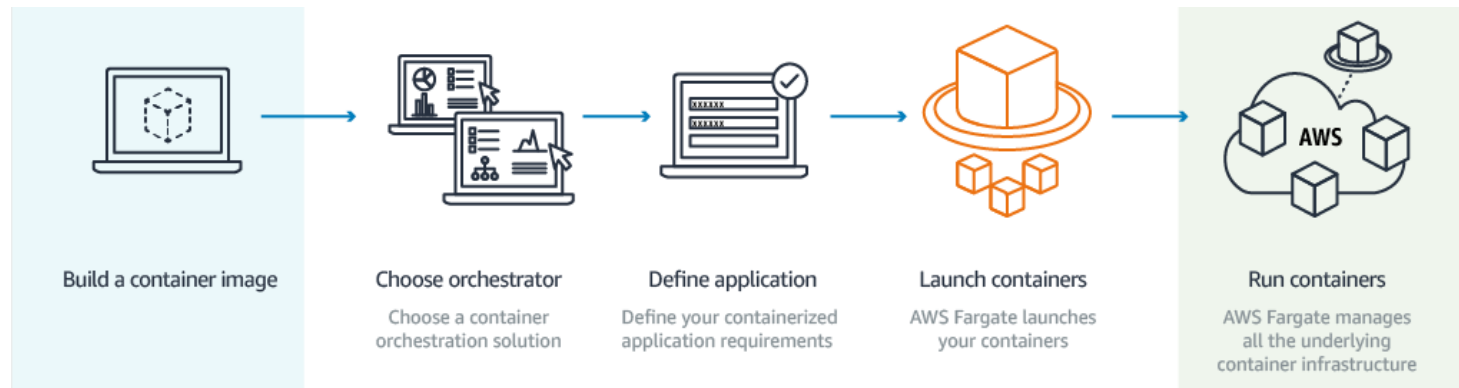
<https://aws.amazon.com/es/eks/>

Kubernetes en AWS

- AWS anunció su servicio **Fargate** para no pagar por nodos, sino pagar por contenedores (no disponible)



AWS Fargate



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

Kubernetes en AWS

- También se puede instalar Kubernetes en instancias AWS



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



<https://github.com/kubernetes/kops>

Conceptos

- **Pod:**

- Unidad mínima de ejecución en Kubernetes
- Uno o más contenedores docker
- Pueden compartir volúmenes e IP

- **Deployment:**

- Ejecución de uno o varios pods del mismo tipo (réplicas)
- Control de salud de los pods y reinicio

- **Servicio:**

- Nombre para acceder a un deployment desde otros servicios
- Publicación del servicio para que sea accesible desde fuera del cluster

<https://kubernetes.io/docs/tutorials/kubernetes-basics/>
<https://kubernetes.io/docs/getting-started-guides/minikube/>

Conceptos

- **Deployment y Pods**

- Un deployment es el responsable de **mantener ejecutándose** uno o varios pods
- Se asume que los pods son *long-lived*, si un pod finaliza su ejecución, el deployment es responsable de arrancarlo de nuevo
- Cada uno de los pods de un deployment es llamado **réplica**
- El escalado de aplicaciones se consigue controlando el **número de réplicas**

Kubernetes

- **Deployment**

- Despliegue de uno o varios réplicas de un pod

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

- Consultar deployments

```
$ kubectl get deployments
```

- Borrar deployment

```
$ kubectl delete deployment kubernetes-bootcamp
```

Kubernetes

- Pod

- Un deployment inicia uno o más pod del mismo tipo (réplicas)
- Los contenedores del pod pueden compartir volúmenes e IP
- En un Pod puede haber un único contenedor

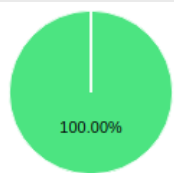
```
$ kubectl get pods  
$ kubectl describe pods  
$ kubectl logs <POD-NAME>
```

Overview

- Cluster
 - Namespaces
 - Nodes
 - Persistent Volumes
 - Roles
 - Storage Classes
- 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

Workloads

Workloads Statuses



Deployments



Pods



Replica Sets

Deployments

Name	Labels	Pods	Age	Images
✓ kubernetes-bootcamp	app: kubernetes-bootcamp	1 / 1	2 minutes	jocatalin/kubernetes-bootcamp:v1

Pods

Name	Node	Status	Restarts	Age
✓ kubernetes-bootcamp-6b476748f8-rwnpv	minikube	Running	0	2 minutes

Replica Sets

Name	Labels	Pods	Age	Images
✓ kubernetes-bootcamp-6b476748f8	app: kubernetes-bootcamp pod-template-hash: 6b476748f8	1 / 1	2 minutes	jocatalin/kubernetes-bootcamp:v1

Kubernetes

- **Servicios**

- Cada pod dispone de una IP, pero los pods pueden morir (con crash) y al volver a arrancar tienen otra IP
- Un servicio crea un nombre lógico asociado a los pods de un deployment
- Permite que el pod o los pods
 - Accesibles desde dentro del cluster con un nombre de DNS
 - Accesibles desde fuera del cluster (con el tipo apropiado)

Kubernetes

- **Servicios**

- Crear un servicio (exponiendo un deployment)

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

- Consulta de servicios

```
$ kubectl get services
```

- Información de un servicio concreto

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

- Borrado del servicio

```
$ kubectl delete service kubernetes-bootcamp
```

Kubernetes



minikube

código
URJC

- Servicios

- Acceso al servicio desde la máquina de desarrollo
 - Comando minikube que abre un browser

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

- Usando comandos para descubrir IP y puerto

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

<http://192.168.99.100:32041/>

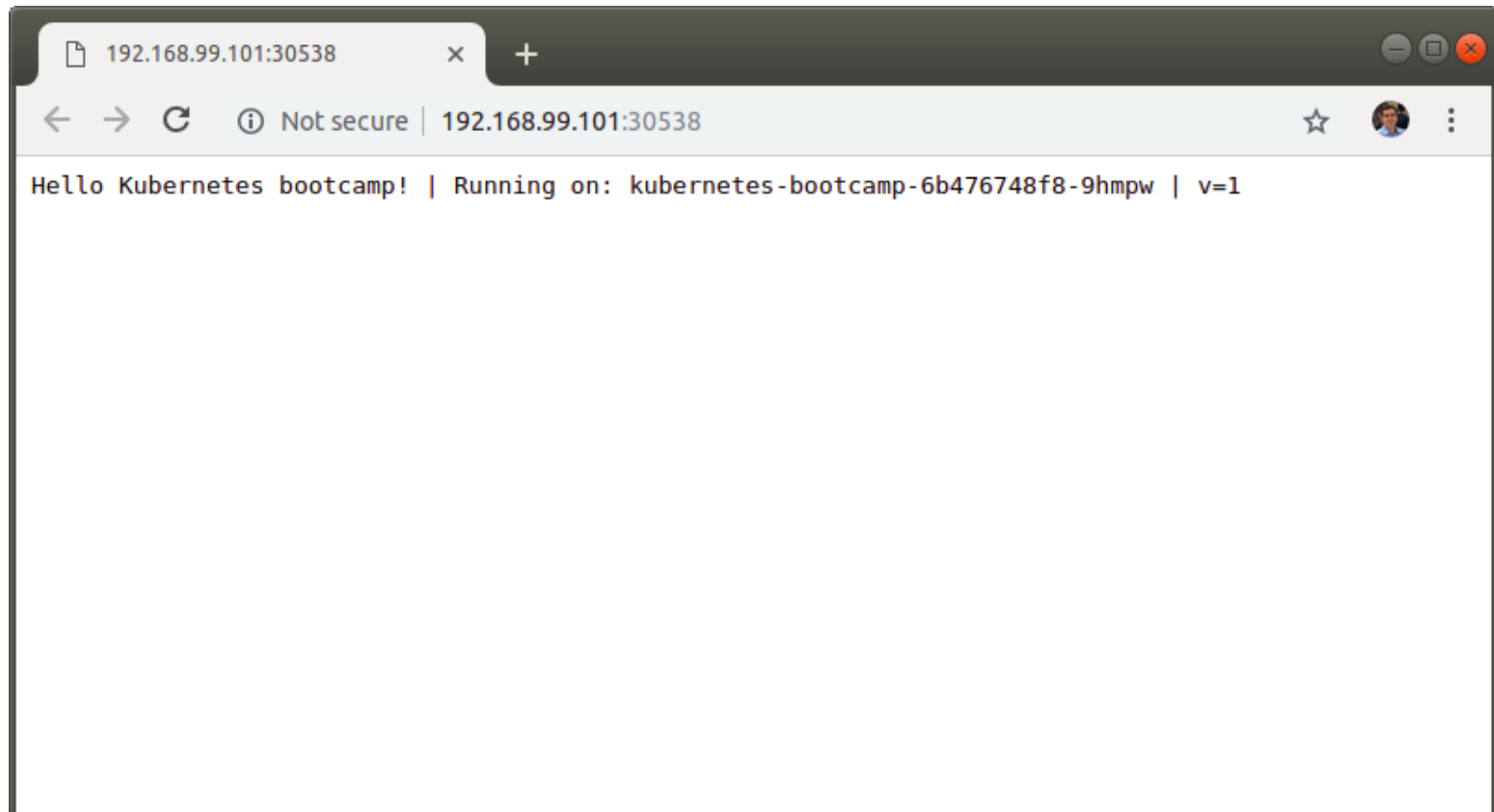
Kubernetes



minikube

código
URJC

- Servicios



Kubernetes

- **Services**

- Si usas NodePort:

- Podría acceder al puerto de cualquier nodo del cluster siempre que lo permitan los grupos de seguridad
- De alguna forma habría que hacer el reparto de las peticiones entre los nodos del cluster

- Si usas LoadBalancer:

- Se crea un LoadBalancer del proveedor cloud por cada servicio

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

Kubernetes

- Services

- LoadBalancer

```
$ 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/>

Kubernetes

The screenshot shows the AWS Management Console interface for the eu-west-1 region. The left sidebar contains navigation menus for NETWORK & SECURITY, LOAD BALANCING, AUTO SCALING, and SYSTEMS MANAGER SERVICES. The main content area displays the 'Load Balancers' page, which includes a table of existing load balancers and a detailed view of the selected load balancer.

Load Balancers Table:

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-32ql7l	bastion-cluster-k8s-codeu-3...		vpc-05949277148c77f65	eu-west-1a	classic
a4559a9e5630f11e88db80ac913b2c4	a4559a9e5630f11e88db80a...		vpc-05949277148c77f65	eu-west-1a	classic
a9a99d632632a11e88db80ac913b2c4	a9a99d632632a11e88db80a...		vpc-05949277148c77f65	eu-west-1a	classic
a5ea04e2c633e11e896170afb7f57108	a5ea04e2c633e11e896170a...		vpc-0daa1a9311a64bb67	eu-west-1b, eu-west-1c...	classic

Load balancer details for a5ea04e2c633e11e896170afb7f57108:

Description | Instances | Health Check | Listeners | Monitoring | Tags | Migration

Basic Configuration

Name:	a5ea04e2c633e11e896170afb7f57108	Creation time:	May 29, 2018 at 2:46:57 PM UTC+2
* DNS name:	a5ea04e2c633e11e896170afb7f57108-909932338.eu-west-1.elb.amazonaws.com (A Record)	Hosted zone:	Z32O12XQLNTSW2
Type:	Classic (Migrate Now)	Status:	3 of 3 instances in service
		VPC:	vpc-0daa1a9311a64bb67

© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Kubernetes

☰

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

✕

?

<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

- 1) Despliega en Kubernetes una aplicación web “webgatos”
 - Imagen: codeurjc/webgatos:v1
 - Puerto: 5000
 - Código de la aplicación:

<https://github.com/codeurjc/Curso-Kubernetes/tree/master/ejemplo1/web-python>

- 2) Crea un servicio para ella
- 3) Accede a ella desde el navegador web

- **Servicios**

- Existen diferentes tipos de servicios con diferentes técnicas de exportación
- Algunas sólo tienen sentido en un proveedor cloud
- Otras necesitan de un servidor DNS porque se accede por nombre

ClusterIP

NodePort

LoadBalancer

ExternalName

<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>

Kubernetes

- **Tipos de servicios**

- ClusterIP (por defecto)
 - Expone el servicio en una IP virtual y nombre DNS usable únicamente desde el cluster
- NodePort
 - Expone el servicio en cada nodo del cluster usando su IP pública y un puerto igual en todos los nodos
 - Se crea automáticamente también un ClusterIP

Kubernetes

• Tipos de servicios

• LoadBalancer

- Crea un balanceador en el proveedor cloud para publicar el servicio
- 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 se interpreta como NodePort

• ExternalName (No disponible en Minikube)

- Se usa para crear un nombre interno que apunta a un servicio externo al cluster

Kubernetes

- 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://kubernetes-bootcamp:8080
```

Kubernetes

- **Varias réplicas (pods)**

- Desplegar varios contenedores permite escalar horizontalmente porque cada contenedor puede ejecutarse en una máquina del cluster
- En el deployment se puede indicar el número de réplicas

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

```
$ kubectl scale deployment/kubernetes-bootcamp \
  --replicas=2
```

- Se puede ver cómo se crean tantos pods como réplicas

```
$ kubectl get pods
```

Kubernetes

- **Varias réplicas (pods)**

- Cada pod tiene su propia IP

```
$ kubectl describe pod webgatos2-574465c7db-wjzf9
| grep IP | sed -E 's/IP:[[:space:]]+//'
```

- Si un deployment tiene asociado un servicio, cuando se accede al servicio (por IP o nombre DNS) se hace balanceo de carga entre todos las réplicas (pods)
- Por defecto el balanceo es RoundRobin

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

Kubernetes

• Ingress

- Los servicios se enrutan internamente dentro del cluser Kubernetes
- Enrutado en base a nombre de dominio (Level 4) o ruta http (Level 7)
- Se basa internamente en un proxy inverso (NGINX, Traefic o similar)
- Necesita un LoadBalancer para el reparto entre nodos, pero sólo se necesita uno que se comparte entre todos los servicios. Con el tipo de servicio LoadBalancer se usa uno por servicio
- Se estudiará más adelante en el curso

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

Kubernetes

- **Etiquetas**

- En Kubernetes cada objeto puede tener un número arbitrario de etiquetas.
- Estas etiquetas se usan para filtrar en cualquier tipo de comando (listados, borrado, ...)

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

- Añadir una etiqueta

```
$ kubectl label pod kubernetes-bootcamp-390780338-g6t39 \
  app=v1
```

Kubernetes

- **Actualización de un deployment**
 - Basta con cambiar el tag de la versión asociada al deployment

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

```
$ kubectl get pods
```

- 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

Kubernetes

- **Specs (Especificaciones)**
 - Crear los deployments y servicios por línea de comandos es engorroso y propenso a errores
 - Además no permite tener la configuración de los recursos bajo control de versiones
 - En general se utilizan ficheros de descripción (llamados spec) en formato YAML de los recursos

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

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

Kubernetes

```
$ kubectl create -f webgatos-minikube-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/>

<https://github.com/codeurjc/Curso-Kubernetes/blob/master/ejemplo1/webgatos-minikube-deployment.yaml>

Kubernetes

```
$ kubectl create -f webgatos-minikube-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/>

<https://github.com/codeurjc/Curso-Kubernetes/blob/master/ejemplo1/webgatos-minikube-service.yaml>

Kubernetes

- **Múltiples objetos 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 webgatos-minikube.yaml
```

Kubernetes

- 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/>

Kubernetes

- **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

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>

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
https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/ejemplo1/webgatos-minikube.yaml
```

- Desplegamos la web de anuncios (deployment y service)

```
$ kubectl create -f
https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/ejemplo2/mysql-service-with-pvc.yaml
```

```
$ kubectl create -f
https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/ejemplo2/java-mysql-minikube.yaml
```

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.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.

Kubernetes

- **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 se usan los volúmenes
- Algunos tipos de volúmenes: local, awsElasticBlockStore, configMap, gitRepo, glusterfs, hostPath, nfs...

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

Kubernetes

- Volúmenes

- Tipo “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>

Kubernetes

- **Volúmenes**

- Tipo "hostPath":
 - Monta una ruta de disco
 - Sirve para acceder a carpetas específicas, como /var/lib/docker
- Tipo "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>

Kubernetes

- 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>

Kubernetes

- **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>

Kubernetes

- **Volúmenes en el pod**
- Algunos tipos de volúmenes básicos se pueden configurar en el pod (emptyDir, hostPath...)

```
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: {}
```

Kubernetes

- 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
```

Kubernetes

- **PersistentVolume (PV)**

- Otros tipos de volúmenes tienen un ciclo de vida no asociado a un pod.
- Representa un recurso de almacenamiento en el cluster, como los nodos de computación
- Para que un pod pueda usar un volumen, lo solicita con un “reclamo” (**PersistenceVolumeClaim**)
- Ese reclamo se atiende con un PersistenceVolume:
 - Creado de forma explícita por el admin
 - Creado de forma dinámica en base a un tipo de almacenamiento (**StorageClass**)

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

Kubernetes

- Ejemplo de uso de PersistentVolume (PV)
 - 1) Creamos un fichero en el nodo
 - 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`

Kubernetes

- Ejemplo de uso de PersistentVolume (PV)
 - 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>

Kubernetes

- Ejemplo de uso de PersistentVolume (PV)
 - 2) Creamos el PersistentVolume

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

Kubernetes

- Ejemplo de uso de PersistentVolume (PV)
 - 2) Creamos el PersistentVolume
 - Creamos el PersistentVolume
 - `$ kubectl create -f https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/Volumenes/persistentVolume.yaml`
 - Vemos su información
 - `$ kubectl get pv task-pv-volume`

- Ejemplo de uso de PersistentVolume (PV)
- 3) Creamos un PersistentVolumeClaim
 - Para que un Pod pueda usar un PV, lo tiene que “reclamar”

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: task-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Se vincula el “claim”
con el PV usando el
storageClassName

```
$ kubectl create -f https://raw.githubusercontent.com/codeurjc/Curso-
Kubernetes/master/Volumenes/persistentVolumeClaim.yaml
```

- Ejemplo de uso de PersistentVolume (PV)
- 3) Creamos un PersistentVolumeClaim
 - Al crear un PVC, si existe un volumen con esas características, el PV queda vinculado (Bound)

```
$ kubectl get pv task-pv-volume
```

NAME	CAPACITY	ACCESSMODES	RECLAIMPOLICY	STATUS	CLAIM
	STORAGECLASS	REASON	AGE		
task-pv-volume	10Gi	RWO	Retain	Bound	
default/task-pv-claim	manual		2m		

```
$ kubectl get pvc task-pv-claim
```

NAME	STATUS	VOLUME	CAPACITY	ACCESSMODES
STORAGECLASS	AGE			
task-pv-claim	Bound	task-pv-volume	10Gi	RWO
30s				manual

- Ejemplo de uso de PersistentVolume (PV)
- 4) Crear un pod y un servicio que use el PVC

```
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: task-pv-claim
  containers:
    - name: task-pv-container
      image: nginx:1.15
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```

```
kind: Service
apiVersion: v1
metadata:
  name: task-pv-svc
spec:
  ports:
    - port: 80
      protocol: TCP
      targetPort: 80
      name: task-pv-pod-port
  selector:
    app: pod-pvc
  type: NodePort
```

```
$ kubectl create -f https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/Volumenes/pod-pvc.yaml
```

Kubernetes

- Ejemplo de uso de PersistentVolume (PV)
- 5) Comprobar que los datos están en el volumen
- Accedemos al servicio con minikube

```
$ minikube service task-pv-svc
```

- Deberíamos poder ver el mensaje en el navegador

```
Hello from Kubernetes storage
```


Kubernetes

• 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 Persistence Volume creado previamente (/mnt/data)
 - La ruta en la que MySQL guarda los datos (mountPath) es: /var/lib/mysql

Ejercicio 4

• Solución

- `kubectl create -f https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/ejemplo2/mysql-service-with-pvc.yaml`
- `kubectl create -f https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/ejemplo2/java-mysql-minikube.yaml`

Kubernetes

- 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

Kubernetes

- **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/>

Kubernetes

- **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/>

Kubernetes

- 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/>

Kubernetes

- 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



Kubernetes

- **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/>



Kubernetes

- **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

Kubernetes

- **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

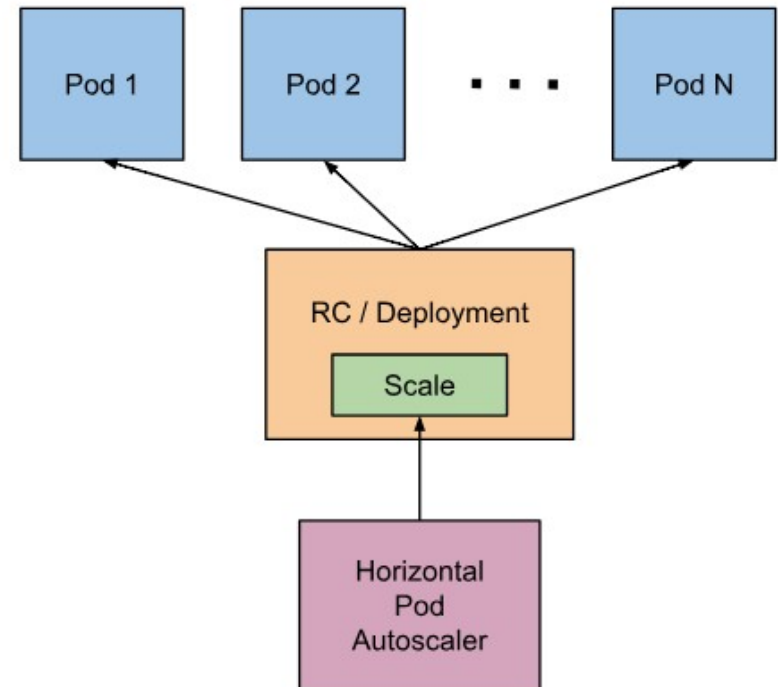
<https://coreos.com/blog/introducing-operators.html>

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

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

Autoescalado

- **Horizontal Pod Autoscaler**
 - El número de réplicas de un **deployment** puede cambiar dinámicamente en base a la **carga de CPU** o de **métricas personalizadas**



Autoescalado

- **Minikube**

- Habilitar los siguientes addons
 - `$ minikube addons enable heapster`
 - `$ minikube addons enable metrics-server`

- **AWS con Kops**

- Habilitamos la recolección de métricas
 - `$ 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`
<https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/HPA/deployment.yaml>

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

Autoescalado

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

```
---
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
```

Autoescalado

- Ejemplo

- Creamos un Horizontal Autoscaling que supervisará memoria y cpu escalando entre 1 y 10 réplicas

- `$ kubectl create -f`

<https://raw.githubusercontent.com/codeurjc/Curso-Kubernetes/master/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 --generator=run-pod/v1 -it \`
`load-generator --image=busybox:1.30 /bin/sh`
- `# / while true; do wget -q -O- \`
`http://php-apache.default.svc.cluster.local; done`

Autoescalado

- Ejemplo
 - Podemos observar cómo escalan los pods:
 - `$ watch kubectl get pods \`
`--selector=app=php-apache`

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

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

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

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

Autoescalado

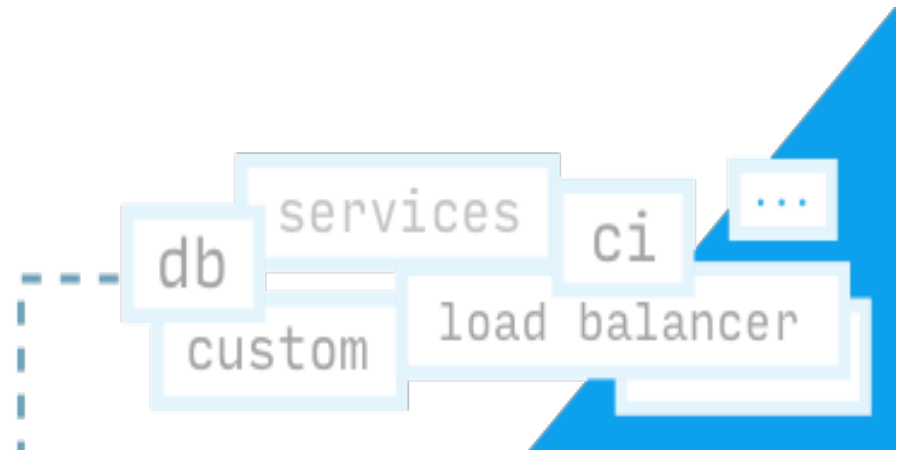
- **Cluster autoscaling**

- Hay que editar el IAM Role de los nodos para que la instancia que ejecuta el cluster-autoscaling pueda operar con el Autoscaling de AWS.
- Editar el AutoScaling group en AWS para aumentar el máximo de instancias que podemos levantar.
- Instalamos el addon de cluster autoscaler en k8s
- Con el ejemplo previo de HPA, cuando ya se han ocupado todos los recursos del cluster, los nuevos pods están en estado pending hasta que AWS inicia un nuevo nodo

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

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** es servicio para gestionar aplicaciones Kubernetes
- Se define como “**El gestor de paquetes para Kubernetes**”
- Es como un **docker-compose**, pero para Kubernetes
- Dispone de un **repositorio oficial** de aplicaciones (llamadas **charts**)
- Se pueden tener **repositorios privados**

<https://helm.sh/>

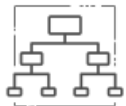
Helm



- 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.



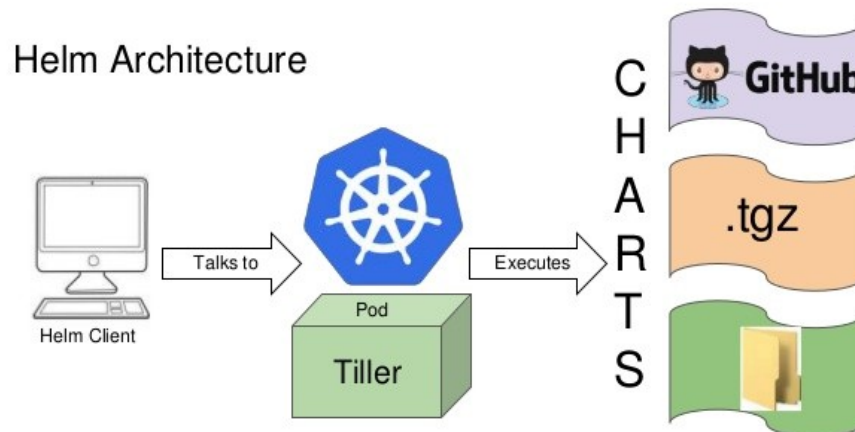
Rollbacks

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

Helm



- **Instalación del cliente**
 - Helm dispone de una parte cliente y un pod desplegado en el cluster
 - **Cliente Helm:** Permite instalar, actualizar y borrar los charts en el cluster kubernetes
 - **Servidor Tiller:** Gestiona los charts en el propio cluster.



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



- Configurar 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



- **Instalación de charts del repositorio oficial**

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

```
$ helm install --name my-drupal stable/drupal
```

- Se puede desplegar el mismo chart con diferentes nombres
- Para acceder a la release consultamos la información en el servicio

```
$ kubectl describe service/my-drupal-drupal
```

Helm



- **Gestión de releases**

- Ver el estado de una release

```
$ helm status my-drupal
```

- Listar las releases del cluster

```
$ helm list
```

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

- Borrar una release

```
$ helm delete my-drupal
```



- **Personalizar un chart en la instalación**

- Ver la lista de valores que se pueden configurar

```
$ helm inspect values stable/mariadb
```

- Configuración por fichero o por línea de comandos

```
$ echo '{mariadbUser: user0}' > config.yaml  
$ helm install --f config.yaml stable/mariadb  
$ helm install --values config.yaml stable/mariadb
```

```
$ helm install --set mariadbUser=user0 stable/mariadb
```

Helm



- **Actualización de releases**

- Se pueden cambiar valores o actualizar de versión

```
$ helm upgrade -f panda.yaml happy-panda stable/mariadb
```

- Cada release se numera con 1,2,3...
- Se puede hacer rollback para volver a una versión anterior

```
$ helm rollback happy-panda 1
```

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



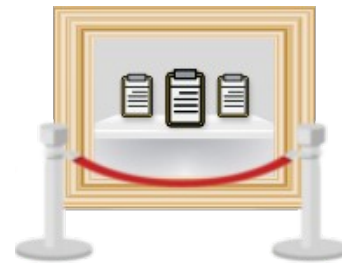
- Repositorios

- Se puede ejecutar un servidor básico en local

```
$ helm serve
```

- Existe un repositorio más completo

ChartMuseum



<https://github.com/kubernetes-helm/chartmuseum>

Ejercicio 5

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

Ejercicio 5

- Solución

- Clonamos el repositorio
 - <https://github.com/codeurjc/Curso-Kubernetes/tree/master/ejemplo4/java-bd>
- Instalamos la aplicación
 - `$ helm install --name java-bd ./java-bd`