

Introducción a Docker

UD 08. Introducción a Kubernetes



Fons Social Europeu

L'FSE inverteix en el teu futur

UNIÓ EUROPEA

Autor: Sergi García Barea

Actualizado Abril 2022

Licencia



Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 **Importante**

 **Atención**

 **Interesante**

1. Introducción	3
2. Kubernetes	3
2.1. ¿Qué es Kubernetes?	3
2.2. Entendiendo Kubernetes	3
2.3. ¿Cómo se estructura un clúster Kubernetes?	3
2.4. ¿Qué es un Pod?	4
2.5. ¿Qué es un despliegue de una aplicación?	4
2.6. ¿Qué es un servicio?	4
2.7. ¿Qué son volúmenes persistentes?	4
2.8. ¿Cómo utilizamos despliegues de aplicación, volúmenes persistentes y servicios?	4
3. Kubernetes con un solo nodo: MiniKube	5
3.1. Instalando MiniKube en sistemas Linux	5
4. Utilizando Kubernetes y MiniKube	7
5. Interfaces gráficas para la gestión de Kubernetes	7
6. Desplegando Kubernetes en principales proveedores	7
7. Bibliografía	7

UD08. INTRODUCCIÓN A KUBERNETES

1. INTRODUCCIÓN

En esta unidad haremos una introducción a los orquestadores usando “**Kubernetes**”. Esta herramienta por sí sola es bastante extensa, por lo cual en esta unidad simplemente haremos una pequeña introducción. El objetivo de esta introducción será aclarar algunos conceptos básicos y assimilarlos mediante algunos casos prácticos sencillos.

2. KUBERNETES

2.1 ¿Qué es Kubernetes?

La herramienta “**Kubernetes**”, también conocida en muchos sitios como “**K8s**”, es una herramienta que nos permite crear un clúster de contenedores y orquestar su despliegue. Cuando hablamos del término “**orquestar**”, nos referimos a que permite cosas tales como realizar despliegue automático, escalado de servicios y balanceo de carga.

“**Kubernetes**” fue creado por Google y es software libre. Actualmente, lo tienen incorporado para un sencillo uso servicios como OVH, Google Cloud, Azure o AWS.

Para más información <https://kubernetes.io/> y <https://kubernetes.io/docs/home/>.

2.2 Entendiendo Kubernetes

Una idea muy simple para entender qué puede hacer “**Kubernetes**” por nosotros, es que nos va a liberar de preocuparnos de todo lo relacionado con el despliegue y escalado. Nosotros, mediante comandos y/o un fichero **YAML** le indicaremos cómo queremos que se despliegue nuestra aplicación y “**Kubernetes**” se encargará de todo.

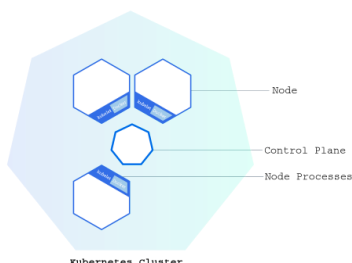
Google editó un cómic explicando las posibilidades de “Kubernetes”. **Es extremadamente recomendable su lectura** para comprender bien las posibilidades de “Kubernetes”

Lo tenéis disponible en <https://cloud.google.com/kubernetes-engine/kubernetes-comic?hl=es-419>

Leerlo no es una pérdida de tiempo.

2.3 ¿Cómo se estructura un clúster Kubernetes?

Para explicar cómo se estructura “**Kubernetes**” nos apoyaremos en la siguiente imagen



Fuente imagen: <https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/>

Un clúster “**Kubernetes**” se estructura en uno o varios nodos. El software encargado de lanzar cada nodo se llama “**Kubelet**”. Cada uno de estos nodos:

- Pueden estar en distintas máquinas.
- Contienen uno o varios Pods (veremos más adelante este término).
 - Usando estos Pods, ejecutan la aplicación.
- Al menos uno de ellos es “maestro” y es el encargado de coordinar los distintos nodos.
 - Pueden existir “réplicas” del nodo maestro.

! **Atención:** durante esta unidad solo veremos casos prácticos con un solo nodo.

2.4 ¿Qué es un Pod?

“**Kubernetes**”, al contrario de lo que pueda parecer, no trabaja directamente con los contenedores, sino que la unidad mínima con la que opera son los “**Pods**”. Los “**Pods**” pueden estar formados por uno o varios contenedores Docker (estando este conjunto de contenedores alojados en una misma máquina), aunque es habitual que un “**Pod**” únicamente contenga un contenedor.

Más información en <https://kubernetes.io/docs/concepts/workloads/pods/>

2.5 ¿Qué es un despliegue de una aplicación?

Dentro de “**Kubernetes**” un “**despliegue de una aplicación**” define unas características de una aplicación implementadas mediante uno o varios “**Pods**”. Estos son los que contienen la aplicación. La vida de estos “**Pods**” está controlada por un controlador que se encarga de tareas como escalado, reinicio o actualización.

Más información en <https://kubernetes.io/es/docs/concepts/workloads/controllers/deployment/>

2.6 ¿Qué es un servicio?

Dentro de “**Kubernetes**” un servicio es una forma de exponer un “**despliegue de una aplicación**” (como el indicado anteriormente) como un servicio de red. No todos los despliegues deben ser expuestos, solo aquellos que queramos que puedan ser accedidos desde fuera del clúster.

Más información en <https://kubernetes.io/docs/concepts/services-networking/service/>

2.7 ¿Qué son volúmenes persistentes?

El concepto de volumen persistente en “**Kubernetes**” es diferente al concepto de volumen en Docker: mientras que en “**Docker**” un volumen simplemente es un directorio del anfitrión montado en unos contenedores, en “**Kubernetes**” los volúmenes persistentes, añade una abstracción que permite pueden estar en cualquier lugar del clúster.

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

2.8 ¿Cómo utilizamos despliegues de aplicación, volúmenes persistentes y servicios?

A efectos prácticos, mediante “**despliegues de aplicación**” lo que hacemos es crear los “**Pods**” necesarios para que funcione la aplicación. Un primer ejemplo podría ser un despliegue donde creamos una base de datos **MySQL**. Este podría además tener un volumen persistente para almacenar la base de datos.

Un segundo ejemplo de despliegue podría ser desplegar una aplicación web. Un despliegue puede utilizar otro (por ejemplo, la aplicación web del segundo despliegue podría utilizar el servicio **MySQL**).

Finalmente, los servicios en “**Kubernetes**” se utilizan para exponer en la red aquellas aplicaciones que queremos sean accedidas. Por ejemplo, el primer despliegue posiblemente no necesitaría ser expuesto, pero el segundo, con la aplicación web, probablemente sí.

3. KUBERNETES CON UN SOLO NODO: MINIKUBE

Como hemos comentado anteriormente, en esta unidad de introducción a “*Kubernetes*”, nuestro clúster estará formado por un único nodo maestro. Esta práctica es habitual sobre todo en momentos de desarrollo: los desarrolladores tienen un pequeño clúster mono-nodo en su máquina y una vez funciona todo correctamente, realizan el despliegue.

Para facilitarnos esta configuración, utilizaremos “*MiniKube*”. Aunque “*MiniKube*” posee distintos tipos de instalación, generalmente se ejecuta sobre una máquina virtual (**Hypervisor tipo 2**) e incluye internamente tanto “*Docker*” como “*Kubernetes*”. Más información sobre las posibles instalaciones en <https://minikube.sigs.k8s.io/docs/drivers/>.

Al instalar “*MiniKube*”, este y configurará todo lo necesario para poder trabajar con “*Kubernetes*” en mono-nodo. Además, instalaremos el cliente “*kubectl*” para interactuar fácilmente vía consola. Esta configuración es muy útil para que los desarrolladores prueben sus productos desplegados en “*Kubernetes*” luego fácilmente puedan desplegarse sin ningún cambio en sitios como Google Cloud, AWS, Azure u OVH.

Más información de “*MiniKube*” <https://minikube.sigs.k8s.io/docs/>

3.1 Instalando MiniKube en sistemas Linux

En esta dirección <https://minikube.sigs.k8s.io/docs/start/> tenéis distintas propuestas de instalación de “*MiniKube*”. Nosotros seguiremos la propuesta para instalarlo en sistemas Debian/Ubuntu.

En primer lugar, realizaremos

```
curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube_latest_
_amd64.deb
```

y tras ello

```
sudo dpkg -i minikube_latest_amd64.deb
```

Obteniendo un resultado similar a:

```
sergi@ubuntu:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 20.7M  100 20.7M    0     0 4046k      0  0:00:05  0:00:05 --:--:-- 5294k
sergi@ubuntu:~$ sudo dpkg -i minikube_latest_amd64.deb
[sudo] contraseña para sergi:
Seleccionando el paquete minikube previamente no seleccionado.
(Leyendo la base de datos ... 352482 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar minikube_latest_amd64.deb ...
Desempaquetando minikube (1.19.0-0) ...
Configurando minikube (1.19.0-0) ...
sergi@ubuntu:~$
```

Una vez instalado, iniciaremos el clúster de “*Kubernetes*” mediante “*MiniKube*” con:

```
minikube start
```

Obteniendo un resultado similar a:

```
sergi@ubuntu:~$ minikube start
minikube v1.19.0 en Ubuntu 20.04
Controlador docker seleccionado automáticamente. Otras opciones: none, ssh
Starting control plane node minikube in cluster minikube
Pulling base image ...
Descargando Kubernetes v1.20.2 ...
> preloaded-images-k8s-v10-v1...: 491.71 MiB / 491.71 MiB 100.00% 3.12 MiB
> gcr.io/k8s-minikube/kicbase...: 357.67 MiB / 357.67 MiB 100.00% 1.72 MiB
Creando docker container (CPUs=2, Memory=2200MB) ...

❗ Docker is nearly out of disk space, which may cause deployments to fail! (87% of capacity)
Suggestion:

Try one or more of the following to free up space on the device:

1. Run "docker system prune" to remove unused Docker data (optionally with "-a")
2. Increase the storage allocated to Docker for Desktop by clicking on:
   Docker icon > Preferences > Resources > Disk Image Size
3. Run "minikube ssh -- docker system prune" if using the Docker container runtime
Related issue: https://github.com/kubernetes/minikube/issues/9024

Preparando Kubernetes v1.20.2 en Docker 20.10.5...
  ■ Generating certificates and keys ...
  ■ Booting up control plane ...
  ■ Configuring RBAC rules ...
Verifying Kubernetes components...
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
Complementos habilitados: default-storageclass, storage-provisioner
❗ kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
✅ Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
sergi@ubuntu:~$
```

La propia instalación de “**MiniKube**” nos avisa que no tenemos instalado “**kubectl**” e incluso nos da una alternativa a su uso. Pero dado que existe gran documentación de “**Kubernetes**” usando esta herramienta, la instalaremos mediante los siguientes comandos:

```
curl -LO
"https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/b
in/linux/amd64/kubectl"
```

Y tras la descarga, damos permiso de ejecución y movemos el fichero a “**/usr/local/bin**”.

```
chmod +x ./kubectl;
```

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

Obteniendo un resultado similar al de la imagen:

```
sergi@ubuntu:~$ curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-relea
e/release/stable.txt)/bin/linux/amd64/kubectl"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100 44.2M  100 44.2M    0     0  4836k      0  0:00:09  0:00:09 --:--:-- 5266k
sergi@ubuntu:~$ chmod +x ./kubectl
sergi@ubuntu:~$ sudo mv ./kubectl /usr/local/bin/kubectl
sergi@ubuntu:~$
```

“**MiniKube**” también incluye un “**dashboard**” para poder utilizar y visualizar algunos elementos de manera gráfica. Podemos invocarlo usando:

```
minikube dashboard
```

4. UTILIZANDO KUBERNETES Y MINIKUBE

Por la gran profundidad de “**Kubernetes**”, esta unidad está bastante simplificada. Una vez entendida la estructura base de cómo funciona esta herramienta (clúster, despliegue, servicio, etc.), dejamos los casos prácticos de la unidad y la “**CheatSheet**” como herramientas autoguiadas para comprender de forma aplicada los principales comandos de “**Kubernetes**” y “**MiniKube**”, así como la configuración de “**Kubernetes**” mediante ficheros **YAML**.

5. INTERFACES GRÁFICAS PARA LA GESTIÓN DE KUBERNETES

En este punto recomendamos dos elementos que mediante interfaz gráfica nos permiten la gestión de “**Kubernetes**”:

- **Helm:** permite la instalación sencilla de aplicaciones en “**Kubernetes**”.
 - <https://helm.sh/>
- **Rancher:** software para gestión de contenedores y clúster con “**Kubernetes**”.
 - <https://rancher.com/>
- **Plugin Visual Studio Code “Kubernetes”:** plugin que nos permite visualizar y editar opciones de “**Kubernetes**” dentro del IDE Visual Studio Code.
 - <https://marketplace.visualstudio.com/items?itemName=ms-kubernetes-tools.vscod-e-kubernetes-tools>

6. DESPLEGANDO KUBERNETES EN PRINCIPALES PROVEEDORES

En este punto proporcionamos algunos enlaces de como desplegar de forma sencilla aplicaciones usando “**Kubernetes**” en los principales servicios de computación en la nube:

- **Google Cloud:**
 - Enlace al servicio: <https://cloud.google.com/kubernetes-engine?hl=es>
 - Guía rápida: <https://cloud.google.com/code/docs/intellij/quickstart-k8s>
- **AWS:**
 - Amazon Elastic Kubernetes Service: <https://aws.amazon.com/es/eks/>
 - Taller de funcionamiento: <https://www.eksworkshop.com/>
- **Azure:**
 - Kubernetes: <https://azure.microsoft.com/es-es/services/kubernetes-service/>
 - Guía aprendizaje:
 - <https://azure.microsoft.com/mediahandler/files/resourcefiles/kubernetes-learning-path/Kubernetes%20Learning%20Path%20version%201.0.pdf>
- **OVH:**
 - Kubernetes: <https://www.ovhcloud.com/es-es/public-cloud/kubernetes/>

7. BIBLIOGRAFÍA

- [1] Kubernetes <https://kubernetes.io/>
- [2] Kubernetes docs <https://kubernetes.io/docs/home/>
- [3] MiniKube <https://minikube.sigs.k8s.io/docs/>