Instalación de Kubernetes

Instalación de Kubernetes

En este módulo vamos a estudiar los siguientes temas:

- Alternativas para instalación simple de k8s
- Instalación de minikube
- Instalación y configuración de kubectl
- Despliegues de aplicaciones en Kubernetes

Alternativas para instalación simple de k8s

Kubernetes es un software pensado para poner en producción aplicaciones más o menos complejas que se ejecutan sobre contenedores, garantizando su disponibilidad, escalabilidad y actualización sin interrupciones.

En un entorno en producción no se instala Kubernetes en un solo equipo (nodo), sino que creamos un cluster de nodos que permita garantizar el funcionamiento ininterrumpido de las aplicaciones incluso en el caso de que uno o varios de los nodos del cluster tengan algún tipo de incidencia.

Configurar y actualizar un cluster de Kubernetes es una tarea compleja, pero existe la posibilidad de instalar de forma fácil un cluster de Kubernetes compuesto por un solo nodo o un conjunto pequeño para algunos casos de uso, como en el caso de la instalación de un entorno de desarrollo o aprendizaje, que es precisamente la situación que tenemos en nuestro caso. Estas instalaciones de Kubernetes no son adecuadas para entornos en producción, pero nos permiten utilizar Kubernetes de forma sencilla, conocer los objetos y atacar la API sin tener que utilizar una instalación más compleja o costosa.

minikube

Minikube permite desplegar localmente un "cluster" de Kubernetes con un solo nodo. Minikube es un proyecto oficial de Kubernetes y es probablemente la solución más adecuada para aprender a usar k8s, ya que es un proyecto maduro y muy sencillo de instalar. Los requisitos mínimos para instalar minikube en nuestro equipo son:

- 2 CPUs
- 2GiB de memoria
- 20GiB de espacio libre en disco
- Un sistema de virtualización o de contenedores instalado:
    - Docker
    - Hyperkit
    - Hyper-V
    - KVM
    - Parallels
    - Podman

- VirtualBox
- VMWare

Minikube instalará un nodo de Kubernetes en el sistema de virtualización/contenedores que prefiramos, siendo unas opciones más adecuadas que otras dependiendo del sistema operativo de nuestro equipo, tal como se muestra en https://minikube.sigs.k8s.io/docs/drivers/. En versiones recientes, es posible aumentar el número de nodos del cluster de minikube, aunque para el objetivo de este curso no es necesario y haremos la instalación estándar de un solo nodo.

Los detalles para la instalación local de minikube los explicamos en la siguiente sección, ya que va a ser el método recomendado para realizar este curso.

kubeadm

kubeadm es una solución más realista que minikube si se instala un cluster de Kubernetes con varios nodos. Su instalación no es especialmente compleja, pero no está tan automatizada como minikube y necesita más recursos y tiempo para configurarlo. kubeadm es una opción muy interesante cuando queremos ver de forma detallada la diferencia entre lo que se ejecuta en el nodo controlador y en los nodos workers, que no se puede apreciar en minikube.

La instalación de kubeadm se realiza típicamente en varias máquinas virtuales o varias instancias de nube y dejamos un par de enlaces para quienes estén más interesados en indagar en este software:

- https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
- https://www.josedomingo.org/pledin/2018/05/instalacion-de-kubernetes-con-kubeadm/

kind

kind (kubernetes in docker) es un proyecto oficial de Kubernetes más reciente que los dos anteriores y que permite desplegar un cluster de Kubernetes con varios nodos sobre docker. Es también muy interesante como opción de instalación local y de forma análoga al anterior, dejamos un par de enlaces para quienes estén interesados en probarlo:

- https://kind.sigs.k8s.io/docs/user/quick-start/
- https://www.josedomingo.org/pledin/2021/02/kubernetes-con-kind/

k3s

A diferencia de las opciones anteriores, k3s es una distribución de Kubernetes que sí está pensada para poner en producción, pero en unas circunstancias peculiares como son su uso para IoT, edge computing y en general para configurar clusters de Kubernetes en sistemas de pocos recursos (k3s es por ejemplo la opción más adecuada para usar Kubernetes en la arquitectura arm). k3s no es un proyecto oficial de Kubernetes, sino que lo comenzó a desarrollar la empresa Rancher y hoy en día lo mantiene la Cloud Native Computing Foundation.

Los pasos para la instaciónde k3s están disponibles en:

- https://rancher.com/docs/k3s/latest/en/installation/

Conclusión

Aunque existen múltiples opciones de instalación de Kubernetes, en este curso utilizaremos minikube que es el proyecto más maduro y que consideramos más adecuado para comenzar y centrarnos directamente en el uso de Kubernetes, obviando inicialmente los detalles de la instalación de Kubernetes, que realmente es un proceso complejo y que no es lo más adecuado para empezar.

Vídeo: Alternativas para instalación simple de k8s

https://www.youtube.com/embed/tdihL7Q-0IY

Vídeo: Alternativas para instalación simple de k8s

Introducción a la instalación de minikube

El "cluster" de k8s que vamos a utilizar en este curso es el de un solo nodo que va a encargarse de realizar tanto las tareas de master, con los componentes principales de Kubernetes, como de worker, ejecutando las cargas de trabajo en contenedores (ya veremos más adelante que realmente utiliza algo que se llama Pod).

Minikube se distribuye como un programa que se instala en nuestra máquina física (podría instalarse igualmente en una máquina virtual a la que tuviésemos acceso completo) y que al ejecutarlo crea una máquina virtual linux con un cluster de Kubernetes completamente configurado y listo para su uso. Podemos instalar minikube en nuestra máquina con sistema linux, windows o mac y en una variedad importante de sistemas de virtualización, aunque en el curso recomendaremos sólo algunas combinaciones que hemos probado y que incluyen toda la funcionalidad necesaria para realizar el curso.

Nota: Puede haber interacción si utilizamos más de un sistema de virtualización en nuestro equipo, por ejemplo, la utilización en Windows de virtualbox para unas cosas e hyper-v para minikube, puede dar lugar a problemas, por lo que en general es recomendable usar un solo sistema de virtualización.

Combinaciones de sistema operativo/virtualización recomendadas para el curso:

- Linux + KVM
- Linux + VirtualBox
- Windows + HyperV
- Windows + VirtualBox

Instalación de minikube en linux + KVM

Instalación de minikube en linux con KVM

Accedemos a https://minikube.sigs.k8s.io/docs/start/ y seleccionamos el método que prefiramos para instalar, eligiendo nuestro sistema operativo, arquitectura, etc.

Minikube se instala, como otras aplicaciones de Go, como un binario enlazado estáticamente (autoconsistente), que no tiene dependencias de nada y que tenemos que ubicar en algún directorio del PATH de nuestro sistema. Veamos en particular la instalación directa del binario en un sistema linux:

Paso 1

Descargamos como usuario normal y con ayuda de la aplicación curl, la última versión del binario de minikube (en este caso para arquitectura x86-64):

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amc
```

Paso 2

Movemos el binario a un directorio del PATH (lo recomendable en este caso sería /usr/local/bin/) y establecemos permisos de ejecución. Todo esto puede hacerse con los comandos `mv` y `chmod`, o de forma más sencilla con `install`

```
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Comprobamos que se ha instalado correctamente con:

```
minikube version
```

```
minikube version: v1.24.0
commit: 76b94fb3c4e8ac5062daf70d60cf03ddcc0a741b
```

Creación del cluster de k8s

El siguiente paso consiste en lanzar minikube para que cree el cluster de Kubernetes de un solo nodo (master+worker). Minikube puede crear este cluster en diversos sistemas de

virtualización o sobre docker, lo recomendable es visitar la página de "drivers" y seleccionar el método más adecuado para nuestro sistema.

De forma general, se creará el cluster de Kubernetes a través de minikube, mediante la instrucción:

```
minikube start
```

Aunque de forma más concreta, especificaremos el "driver" a utilizar, por ejemplo:

```
minikube start --driver=kvm2
```

Esto creará de forma automática una máquina virtual o un contenedor en el sistema escogido e instalará Kubernetes en ella. Por último, se configura kubectl si está instalado (el cliente de línea de comandos de k8s) para que utilice el cluster recién instalado. Podemos ver una salida típica de la instalación del cluster a continuación:

```
😄  minikube v1.24.0 en Debian 11.2
✨  Using the kvm2 driver based on user configuration
👍  Starting control plane node minikube in cluster minikube
🔥  Creando kvm2 VM (CPUs=2, Memory=3900MB, Disk=20000MB) ...
🐳  Preparando Kubernetes v1.22.3 en Docker 20.10.8...
    ▪ 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" namespa
```

En la última línea de la salida podemos ver que se ha intentado configurar apropiadamente kubectl, a pesar de que no está instalado en el equipo, paso que haremos en el siguiente apartado.

Podemos comprobar en cualquier momento el estado de minikube con la instrucción:

```
minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

Parada y reinicio de minikube

Podemos parar y volver a arrancar minikube cuando sea preciso, ya que no se trata de un cluster de k8s en producción, sino de uno instalado en un equipo convencional. Esto se realiza mediante las instrucciones:

```
minikube stop
✋  Stopping node "minikube"  ...
🛑  1 nodes stopped.
```

```
minikube start
😄  minikube v1.24.0 en Debian 11.2
✨  Using the kvm2 driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🔄  Restarting existing kvm2 VM for "minikube" ...
🐳  Preparando Kubernetes v1.22.3 en Docker 20.10.8...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Complementos habilitados: storage-provisioner, default-storageclass
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespa
```

Vídeo: Instalación de minikube en linux + KVM

https://www.youtube.com/embed/0l_NICnY7AI

Vídeo: Instalación de minikube en linux + KVM

Instalación de minikube en Windows + Hyper-V

En este apartado vamos a instalar minikube utilizando como sistema de virtualización Hyper-V. Lo primero que tenemos que indicar es que la instalación en Windows no se puede realizar en las versiones Home del sistema operativo. Por lo tanto en este ejemplo voy a utiliza un Windows 10 Profesional.

Otro aspecto a tener en cuenta es que Hyper-V y VirtualBox son incompatibles en la misma máquina. Si utilizas habitualmente VirtualBox, utilizalo junto a minikube para crear el cluster de kubernetes. Si optas por el uso de Hyper-V debes desinstalar el VirtualBox.

Paso 1: Instalación de Hyper-V

Para ello vamos a entrar en PowerShell como administrador:

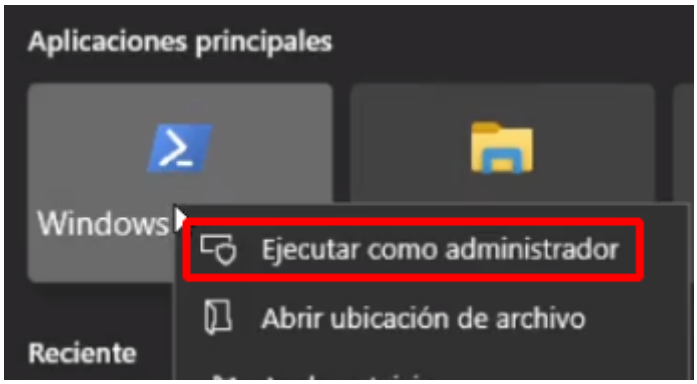


Imagen de elaboración propia (CC BY-NC-SA)

Y ejecutamos la instrucción:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

Imagen de elaboración propia (CC BY-NC-SA)

Una vez instalado, nos pedirá que reiniciamos el sistema para terminar la instalación. Una vez reiniciado el sistema, podemos comprobar que ya tenemos el Administrador de Hyper-V instalado:

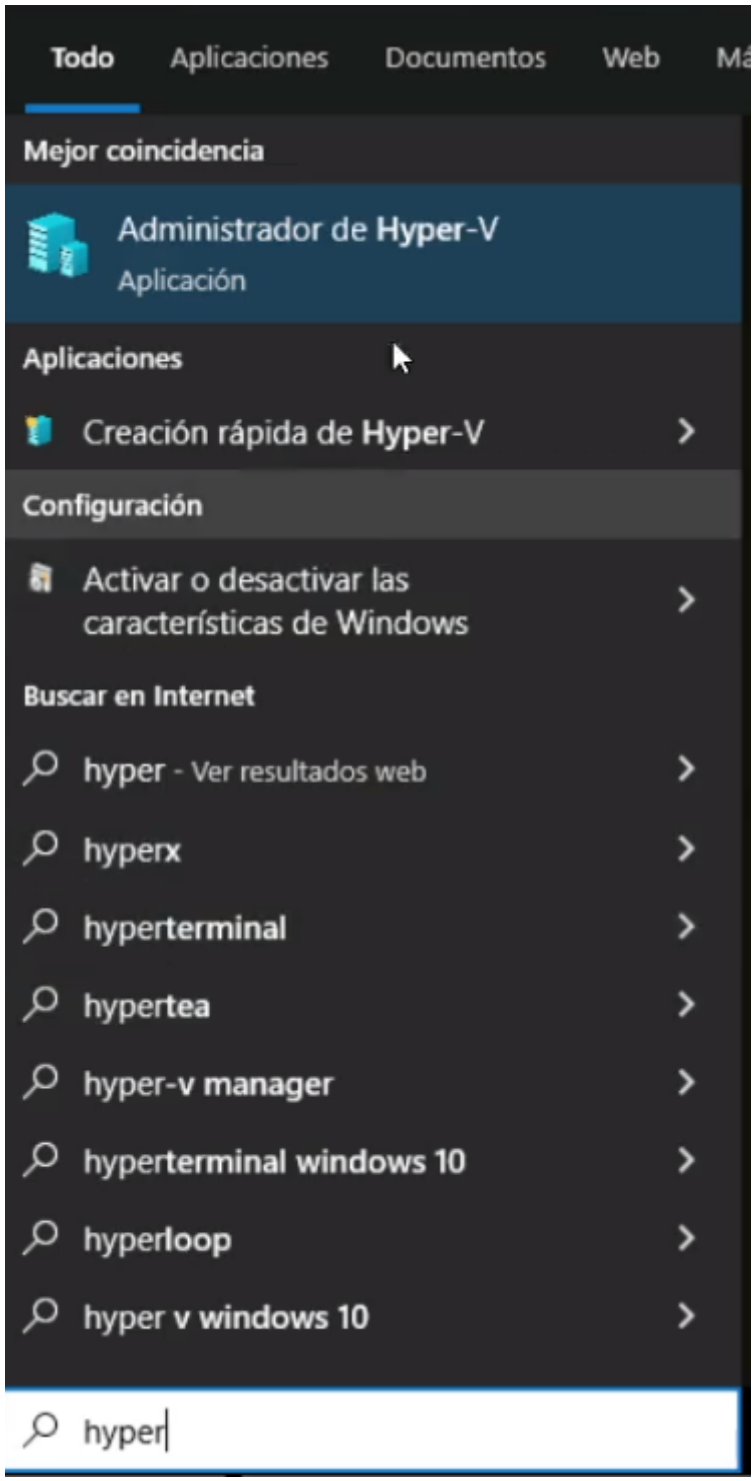


Imagen de elaboración propia (CC BY-NC-SA)

Paso 2: Descargamos minikube

Accedemos a https://minikube.sigs.k8s.io/docs/start/ y seleccionamos el método que prefiramos para instalar, eligiendo nuestro sistema operativo, arquitectura, etc.

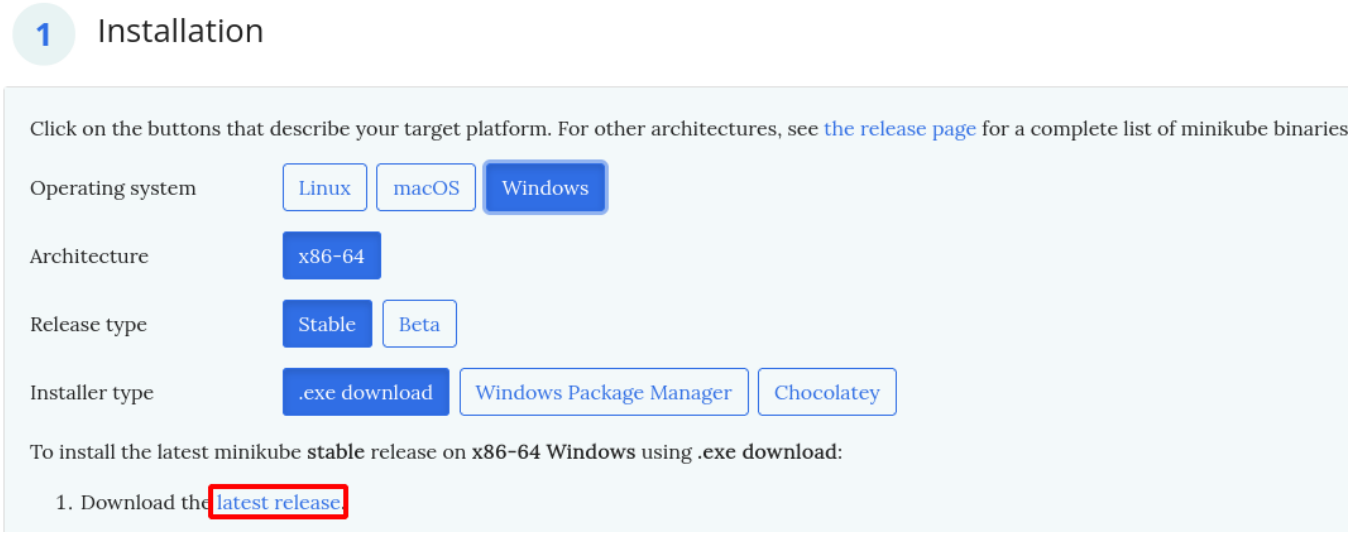


Imagen de elaboración propia (CC BY-NC-SA)

Nos descargamos el fichero ejecutable **`minikube-installer.exe`** y lo ejecutamos para realizar la instalación:

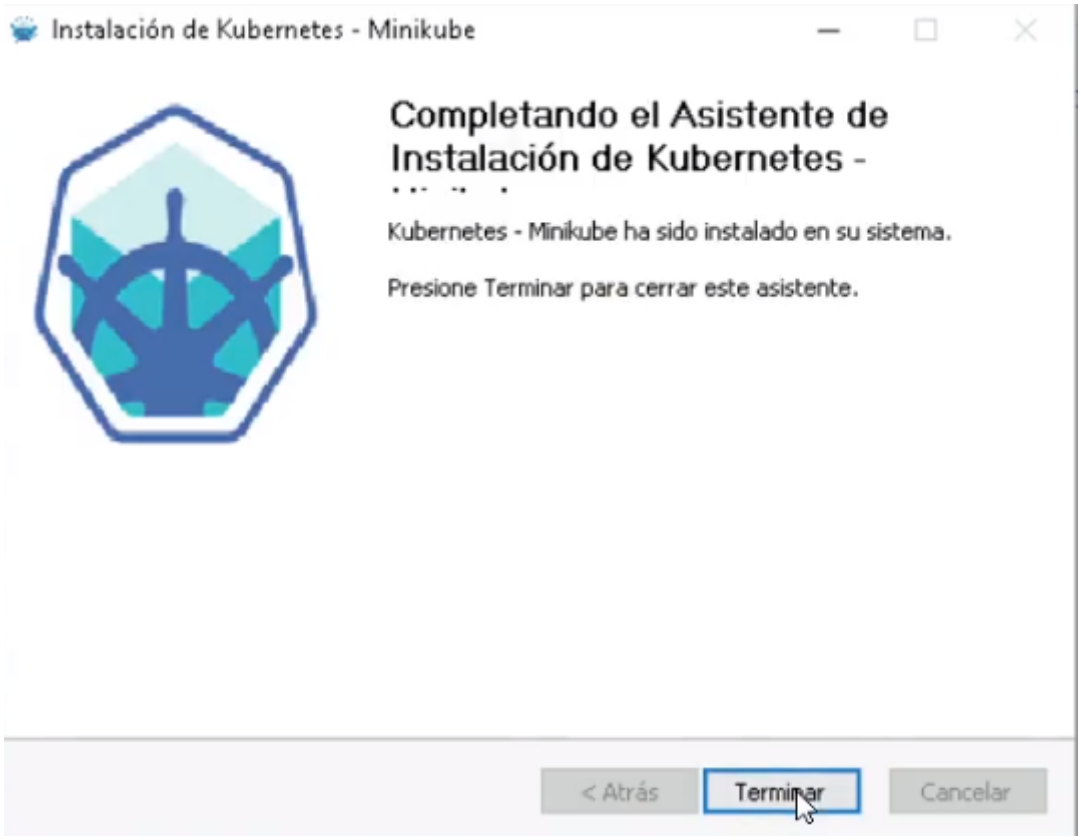


Imagen de elaboración propia (CC BY-NC-SA)

Paso 3: Creación del cluster de kubernetes con minikube

En este apartado vamos a crear un cluster de kubernetes de un nodo. En este caso minikube creará una máquina virtual (de 2Gb de RAM, 2 vcpu y 20G de almacenamiento) en Hyper-V utilizando una imagen que configura la máquina con kubernetes. Para empezar volvemos a cceder a la PowerShell como administrador y ejecutamos la instrucción:

```
minikube start
```

Si nos da algún problema podemos probar con:

```
minikube start --driver=hyperv
```

Para más información puedes consultar la página minikube con Hyper-V de la documentación.

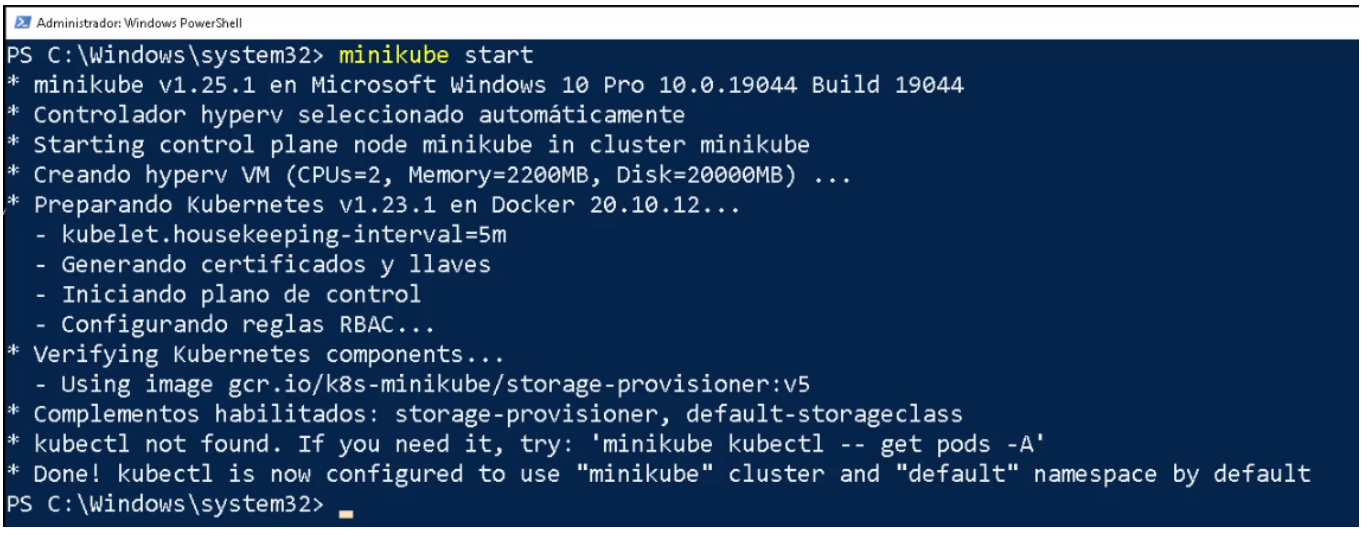


Imagen de elaboración propia (CC BY-NC-SA)

Podemos acceder al Administrador de Hyper-V y comprobar que se ha creado la máquina:

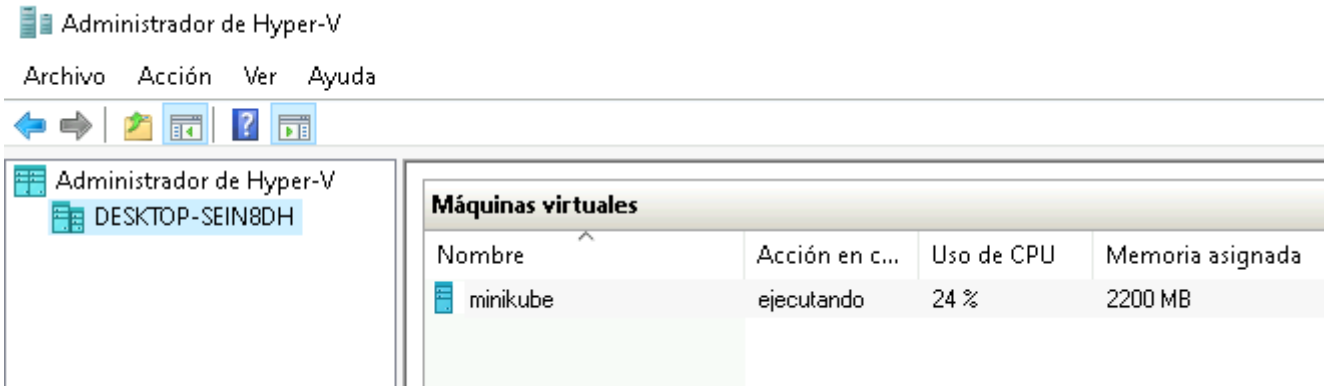


Imagen de elaboración propia (CC BY-NC-SA)

Además podemos comprobar que el cluster está en funcionamiento ejecutando `minikube status`:

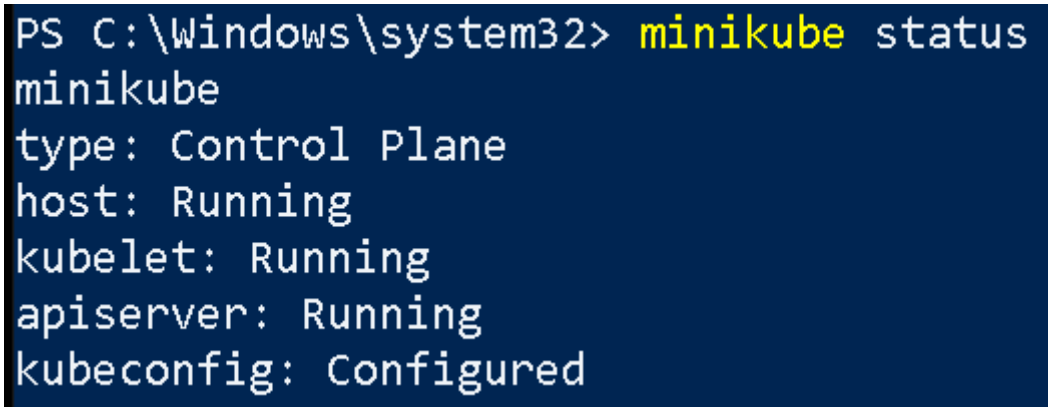


Imagen de elaboración propia (CC BY-NC-SA)

Cuando terminemos de trabajar con kubernetes podemos apagar la máquina virtual con el comando `minikube stop`:

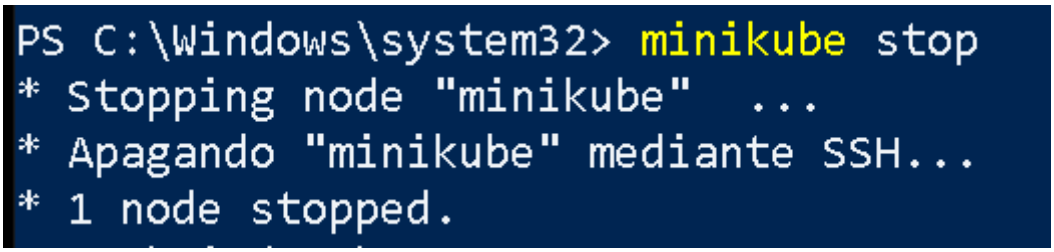


Imagen de elaboración propia (CC BY-NC-SA)

Y en el momento que queramos seguir trabajando iniciaremos la máquina virtual ejecutando `minikubee start`:

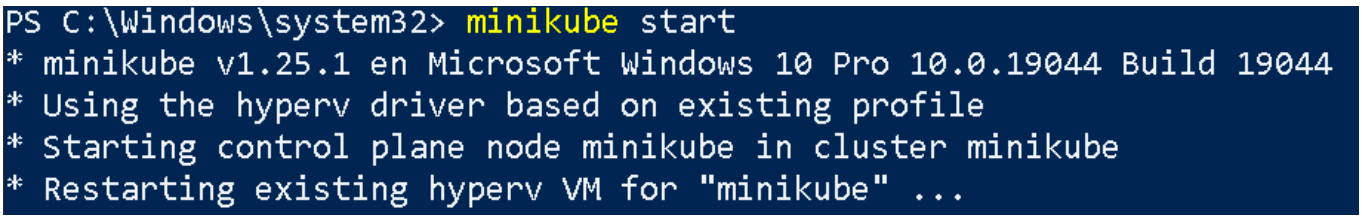


Imagen de elaboración propia (CC BY-NC-SA)

Vídeo: Instalación de minikube en Windows + Hyper-V

https://www.youtube.com/embed/dr1G_aOaBek

Vídeo: Instalación de minikube en Windows + Hyper-V

Instalación y configuración de kubectl en linux

kubectl es la herramienta de línea de comandos utilizada para interactuar con la API de Kubernetes. Es por tanto la herramienta fundamental que vamos a utilizar durante todo el curso para gestionar nuestros objetos en el cluster recién creado con minikube.

kubectl está escrito en Go y de nuevo su instalación es muy simple, ya que se trata de un binario enlazado estáticamente y sin dependencias. Las instrucciones para su instalación están disponibles en la documentación de k8s. A continuación veremos algunas de las opciones que tenemos para instalarlo.

Opción 1. Instalar binario desde el proyecto

Al igual que hemos hecho con minikube, podemos descargar el binario directamente desde la URL del proyecto e instalarlo en `/usr/local/bin`:

```
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https
sudo install kubectl /usr/local/bin/kubectl
```

Este binario obviamente no se actualiza y tendremos que repetir el proceso cuando se actualice.

Opción 2. Instalar desde repositorios no oficiales

El término repositorio no oficial se utiliza para aquellos repositorios que se añaden y que no son los propios de la distribución que estamos utilizando. En este caso, los repositorios no oficiales los proporciona el propio proyecto k8s.

En el caso de las distribuciones Debian y derivadas, el repositorio es `https://packages.cloud.google.com/apt/` y en la documentación se detallan los pasos para instalar `kubectl` a través de apt.

La ventaja de este método respecto al anterior es que sí se actualizará `kubectl` adecuadamente como cualquier otro paquete que tengamos instalado en nuestra distro.

Opción 3. Instalar desde repositorio oficial

En el caso de Debian, se ha añadido soporte para Kubernetes a partir de la versión `bullseye` o Debian 11, por lo que si tenemos instalada esa versión, podemos instalar `kubectl` directamente con apt:

```
sudo apt install kubernetes-client
```

En estos momentos se instala la versión 1.20 de kubectl.

Opción 4. Instalar desde snap

Ubuntu no proporciona de forma directa un paquete con el cliente de k8s, pero sí lo hace a través de snap, por lo que quienes utilicen dicho sistema, lo tienen disponible con un simple:

```
sudo snap install kubectl --classic
```

Configuración kubectl

Una vez instalado `kubectl` podemos comprobar que está disponible y cuál es su versión, con la instrucción:

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.3", GitCommit
The connection to the server localhost:8080 was refused - did you specify the right
```

En el caso anterior, estamos utilizando la versión 1.22.2 y nos informa de que no ha podido conectarse al cluster de Kubernetes con la configuración por defecto (`localhost:8080`). Es decir, aunque tengamos kubectl y minikube instalados, el primero no está configurado todavía para conectarse al cluster de k8s que ejecuta minikube.

La solución más sencilla es parar minikube y volverlo a arrancar, porque de esta manera minikube configurará automáticamente `kubectl`. Si nos fijamos en la salida de minikube anterior, en la que no teníamos instalado `kubectl`, aparecía la línea:

```
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
```

Pero si lo volvemos a repetir ahora, esa línea no aparecerá y se configurará `kubectl` para poder usar el cluster que proporciona minikube. Lo que va a hacer minikube es configurar el fichero `~/.kube/config` de la siguiente manera:

```
apiVersion: v1
clusters:
- cluster:
```

```
    certificate-authority: /home/alberto/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Sun, 30 Jan 2022 20:45:08 CET
        provider: minikube.sigs.k8s.io
        version: v1.24.0
      name: cluster_info
    server: https://192.168.39.115:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Sun, 30 Jan 2022 20:45:08 CET
        provider: minikube.sigs.k8s.io
        version: v1.24.0
      name: context_info
    namespace: default
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/alberto/.minikube/profiles/minikube/client.crt
    client-key: /home/alberto/.minikube/profiles/minikube/client.key
```

Donde en cada caso variará la dirección IP del servidor del cluster (en este caso la 192.168.39.221) y la ubicación de los ficheros de los certificados y claves x509 (en este caso en el directorio `/home/alberto`).

Una vez configurado correctamente `kubectl`, podemos repetir el comando:

```
kubectl version

Client Version: version.Info{Major:"1", Minor:"23", GitVersion:"v1.23.3", GitCommi
Server Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommi
```

Comprobamos que ya aparece la versión del servidor y por tanto se ha podido conectar con el cluster que gestiona minikube. Además podemos ejecutar nuestro primer comando propiamente de `kubectl`:

```
kubectl get nodes
NAME       STATUS   ROLES                  AGE   VERSION
```

```
minikube   Ready    control-plane,master   21m   v1.22.3
```

Si queremos utilizar el autocompletado, podemos generarlo e incorporarlo a nuestro entorno con:

```
echo 'source <(kubectl completion bash)' >>~/.bashrc
```

Y para poder usarlo en esta misma sesión (no será necesario más adelante, ya que el fichero .bashrc se lee cada vez que se inicia una sesión):

```
source ~/.bashrc
```

Vídeo: Instalación y configuración de kubectl en linux

https://www.youtube.com/embed/0p_JGucaSco

Vídeo: Instalación y configuración de kubectl en linux

Instalación y configuración de kubectl en windows

Podemos encontrar la información donde nos explican la instalación de `kubectl` en la documentación. Vamos a elegir la opción de descargar el ejecutable, para ello accedemos a la PowerShell como administrador y en el directorio `C./windows/system32` descargamos el ejecutable (por lo que lo vamos a tener disponible en el PATH). Para ello vamos a ejecutar:

```
curl.exe -LO https://storage.googleapis.com/kubernetes-release/release/v1.23.0/bin/
```

Una vez instalado, desde una PowerShell sin acceso como administrador, podemos empezar a usar `kubectl` y comprobar si podemos acceder al cluster.

Podemos comprobar la versión de `kubectl`:

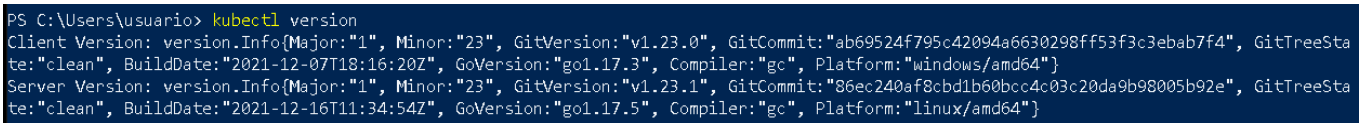


Imagen de elaboración propia (CC BY-NC-SA)

Y ejecutar nuestro primer comando para obtener los nodos del cluster:

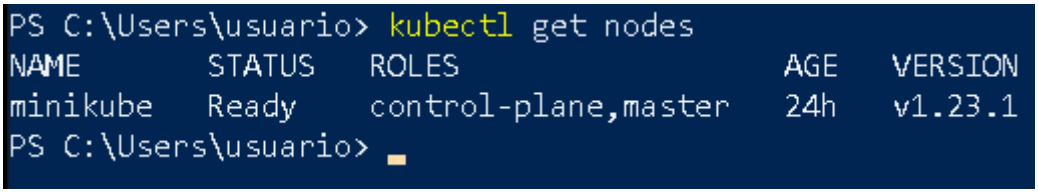


Imagen de elaboración propia (CC BY-NC-SA)

Despliegues de aplicaciones en Kubernetes

Vamos a resumir brevemente, con la ayuda de un par de imágenes, la forma que tiene k8s de hacer despliegues de aplicaciones y lo compararemos con un despliegue "tradicional".

Aunque un despliegue real tiene muchos más elementos que los que vamos a exponer a continuación, con idea de simplificarlo todo y centrarnos en la diferencia de los elementos que intervienen, supondremos una aplicación "tradicional" de dos capas, en las que una serie de equipos son los que están expuestos a Internet y los que pueden ejecutar una parte del código de la aplicación a través de servidores web (a los que de forma genérica denominaremos "front-end"), mientras que otra serie de equipos ejecutan otra parte de código y gestionan el almacenamiento y las bases de datos (a los que llamaremos de forma genérica "back-end").

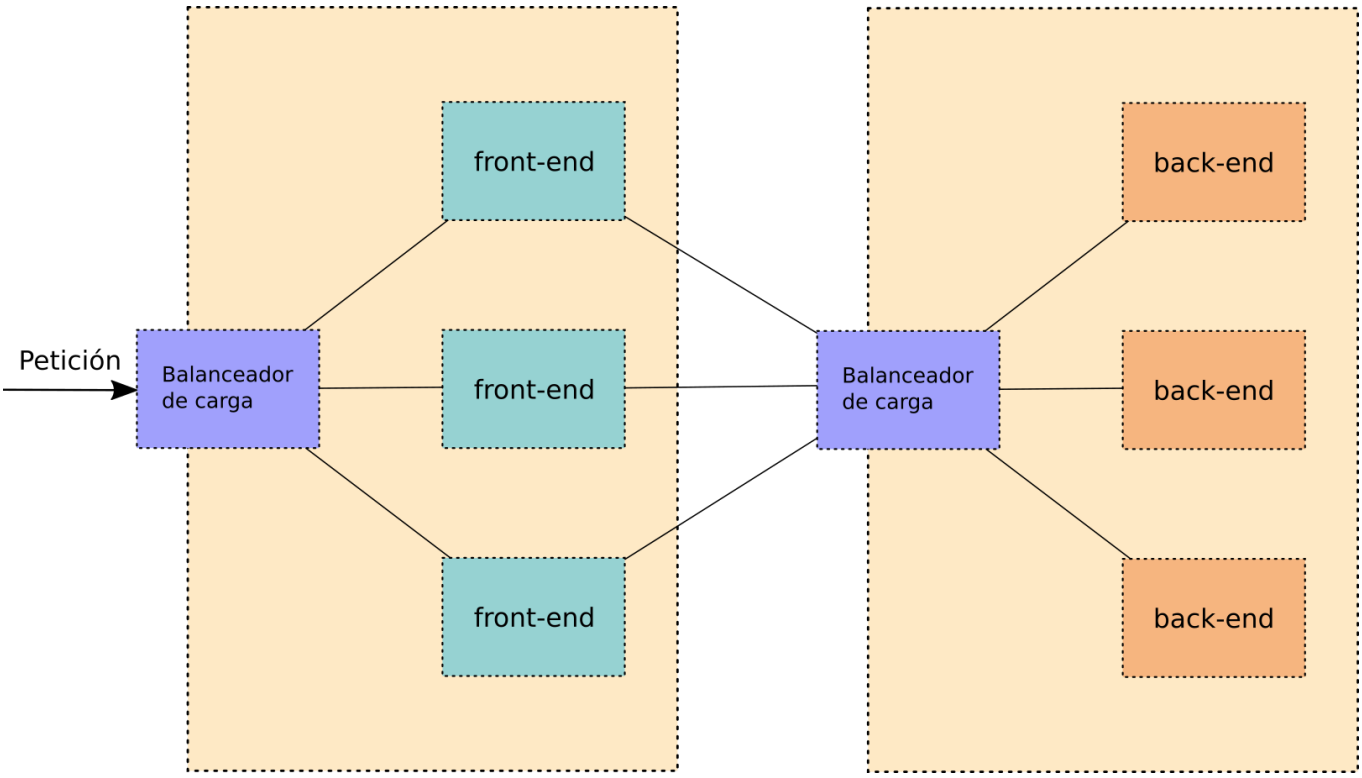


Imagen de elaboración propia (CC BY-NC-SA)

En la imagen anterior, vemos que el balanceador de carga expuesto al exterior recibe la petición, que asigna a alguna de las máquinas virtuales o físicas que forman el "front-end" y que éstas a su vez se comunican con alguna de las máquinas del "back-end" a través del balanceador de carga intermedio.

En el caso de Kubernetes, esto se realiza utilizando una serie de objetos internos, que normalmente se ejecutan sobre contenedores, denominados Pods, ReplicaSets, Deployments, Services e Ingress Controllers. Hay bastantes más objetos de k8s, pero nos centraremos en éstos que son los principales.

En la siguiente imagen podemos ver la forma en la que una petición externa se gestionaría:

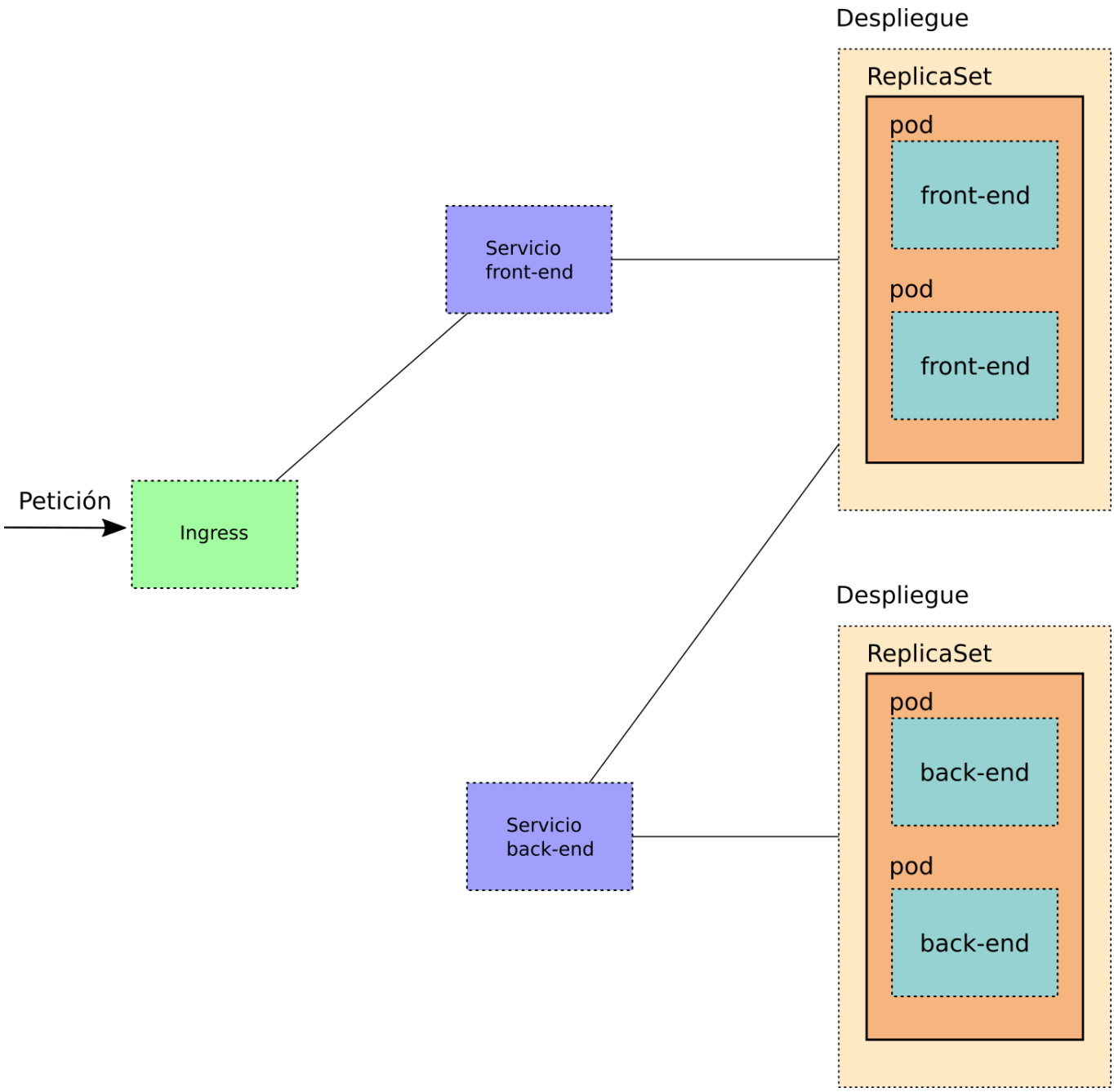


Imagen de elaboración propia (CC BY-NC-SA)

En los siguientes módulos veremos uno a uno estos objetos de k8s y aprenderemos paso a paso cómo se interactúa con ellos y cómo se definen, pero a modo de resumen, podemos enumerar sus principales funciones en la siguiente lista:

- Pods: ejecutan los contenedores
- ReplicaSets:
  - Se encargan de que no haya caída del servicio
  - Gestionan la tolerancia a fallos
  - Proporcionan escalabilidad dinámica
- Deployments:
  - Gestionan las actualizaciones continuas
  - Realizan despliegues automáticos
- Services:
  - Gestionan el acceso a los pods
  - Balancean la carga entre los Pods disponibles
- Ingress:
  - Gestionan el acceso desde el exterior a través de nombre

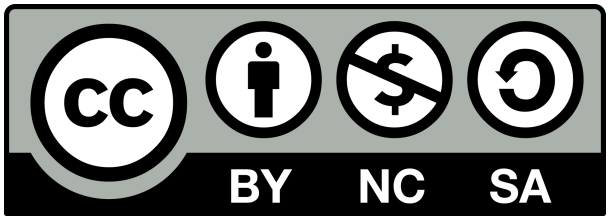
Vídeo: Despliegues de aplicaciones en Kubernetes

https://www.youtube.com/embed/9bCXCemP4aY

Despliegues de aplicaciones en Kubernetes

Créditos

Materiales desarrollados por:
Alberto Molina Coballes
José Domingo Muñoz Rodríguez
Propiedad de la Consejería de Educación y Deporte de la Junta de Andalucía



2021