

Practica de Kubernetes: Instalación y Despliegue de una Aplicación Simple

Paso 1: Instalar Kubernetes en tu máquina local

Este paso se divide en dos partes principales: instalar Minikube y luego instalar kubectl.

Parte 1: Instalar Minikube

¿Qué es Minikube?

Minikube es una herramienta que te permite crear un clúster de Kubernetes de un solo nodo dentro de una máquina virtual (VM) en tu ordenador local. Es ideal para desarrollo y pruebas, ya que te proporciona un entorno de Kubernetes completo sin necesidad de configurar una infraestructura compleja.

¿Cómo instalar Minikube?

Aquí tienes los pasos detallados:

1. Actualizamos la lista de paquetes:

```
sudo apt update
```

2. Instalar las dependencias: Minikube necesita algunas herramientas para funcionar correctamente. Ejecutamos este comando para instalarlas:

```
sudo apt install -y ca-certificates curl gnupg
```

- a. -y: Responde automáticamente "sí" a todas las preguntas durante la instalación.
- b. ca-certificates: Contiene certificados de autoridad de certificación (CA) que permiten verificar la seguridad de las conexiones HTTPS.
- c. curl: Es una herramienta de línea de comandos para transferir datos con URLs.
- d. gnupg: Es una implementación de GNU de la especificación OpenPGP para cifrado y firmas digitales.

Instalar Minikube:

Ahora, instalemos Minikube:

```
curl -LO
```

```
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

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

Esto descarga la última versión de Minikube y la instala en el directorio /usr/local/bin.

Verificar la instalación de Minikube:

Podemos verificar que Minikube se ha instalado correctamente ejecutando:

```
minikube version
```

```
salva@salva-ubuntu:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 119M  100 119M    0     0  9704k      0  0:00:12  0:00:12 --:--:-- 10.2M
salva@salva-ubuntu:~$ minikube version
minikube version: v1.35.0
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty
salva@salva-ubuntu:~$
```

Instalar kubectl:

kubectl es la herramienta de línea de comandos que nos permitirá interactuar con nuestro clúster de Kubernetes creado por Minikube.

En este caso instalaré la última versión directamente desde Kubernetes:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Verificar la instalación de kubectl:

Podemos verificar que **kubectl** se ha instalado correctamente ejecutando:

```
kubectl version --client
```

```

salva@salva-ubuntu:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  138    100  138    0     0    527      0  --:--:-- --:--:-- --:--:--    526
100 54.6M    100 54.6M    0     0  6327k      0  0:00:08  0:00:08 --:--:-- 6709k
[sudo] contraseña para salva:
salva@salva-ubuntu:~$ kubectl version --client
Client Version: v1.32.3
Kustomize Version: v5.5.0
salva@salva-ubuntu:~$

```

Paso 2: Iniciar un clúster de Kubernetes con Minikube

Iniciar Minikube:

Para iniciar tu clúster de Kubernetes local con Minikube, simplemente ejecutamos el siguiente comando:

```
minikube start --driver=docker
```

Este comando se encargará de descargar la imagen del clúster de Kubernetes (si es la primera vez), configurar el contenedor, e iniciar los componentes necesarios para que nuestro clúster esté en funcionamiento.

Durante el proceso de inicio, veremos una serie de mensajes en la terminal indicando las acciones que Minikube está realizando. Tardará unos minutos en completarse.

```
salva@salva-ubuntu: ~  
salva@salva-ubuntu:~$ minikube start --driver=docker  
🐸 minikube v1.35.0 en Ubuntu 22.04  
🌟 Using the docker driver based on user configuration  
👍 Using Docker driver with root privileges  
👍 Starting "minikube" primary control-plane node in "minikube" cluster  
📦 Pulling base image v0.0.46 ...  
📦 Descargando Kubernetes v1.32.0 ...  
  > preloaded-images-k8s-v18-v1...: 333.57 MiB / 333.57 MiB 100.00% 4.57 Mi  
  > gcr.io/k8s-minikube/kicbase...: 500.31 MiB / 500.31 MiB 100.00% 4.20 Mi  
🔥 Creating docker container (CPUs=2, Memory=2900MB) ...  
🔧 Preparando Kubernetes v1.32.0 en Docker 27.4.1...  
  ■ Generando certificados y llaves  
  ■ Iniciando plano de control  
  ■ Configurando reglas RBAC...  
🔗 Configurando CNI bridge CNI ...  
🔍 Verifying Kubernetes components...  
  ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5  
🌟 Complementos habilitados: storage-provisioner, default-storageclass  
🏁 Done! kubectl is now configured to use "minikube" cluster and "default" name  
space by default  
salva@salva-ubuntu:~$
```

Verificar el clúster:

`kubectl get nodes`

Este comando mostrará la lista de nodos en nuestro clúster y su estado.

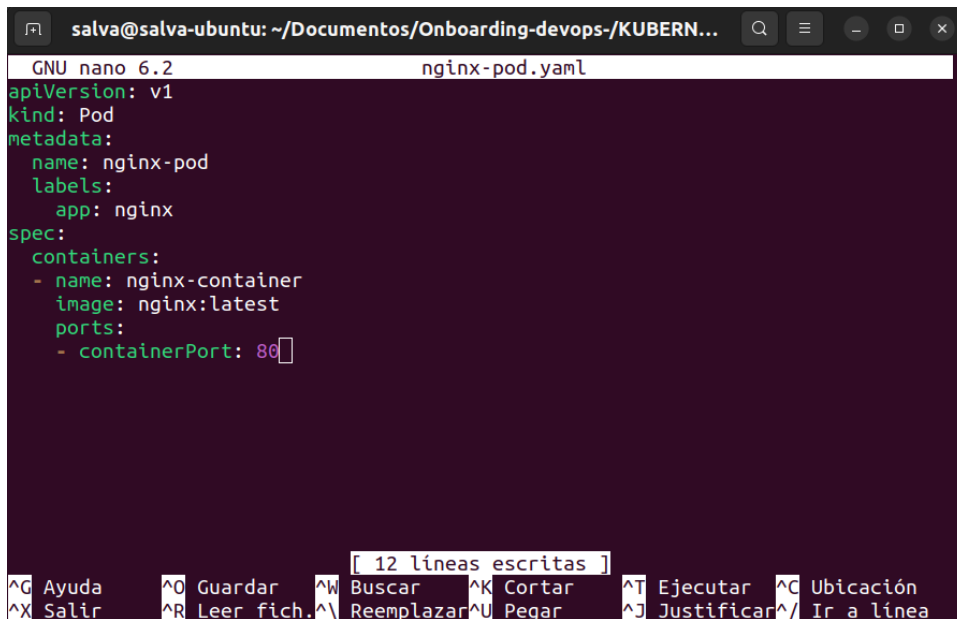
```
salva@salva-ubuntu:~$ kubectl get nodes  
NAME          STATUS    ROLES    AGE   VERSION  
minikube      Ready    control-plane   5m28s   v1.32.0  
salva@salva-ubuntu:~$
```

Paso 3: Crear y Desplegar una Aplicación Simple en Kubernetes

3.1: Crear un archivo YAML para un Pod

Voy a crear un archivo llamado **nginx-pod.yaml** que definirá un Pod. Un Pod es la unidad más pequeña y básica desplegable de Kubernetes, que representa una o más aplicaciones en contenedores que están estrechamente acopladas.

Abro un editor de texto y creo un nuevo archivo llamado **nginx-pod.yaml** con el siguiente contenido:



```
GNU nano 6.2 nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
  - name: nginx-container
    image: nginx:latest
    ports:
    - containerPort: 80
```

Desglose del archivo YAML:

- **apiVersion: v1:** Especifica la versión de la API de Kubernetes que se está utilizando para definir este objeto (en este caso, la versión principal para los objetos básicos).
- **kind: Pod:** Indica que el tipo de objeto que se está definiendo es un Pod.
- **metadata::** Contiene metadatos sobre el Pod:
 - **name: nginx-pod:** Asigna un nombre único al Pod dentro del namespace actual.
 - **labels::** Son pares clave-valor que se utilizan para organizar y seleccionar grupos de objetos. Aquí, estamos etiquetando el Pod con `app: nginx`.
- **spec::** Define el estado deseado del Pod, incluyendo los contenedores que deben ejecutarse dentro de él:
 - **containers::** Es una lista de contenedores que se ejecutarán dentro del Pod.
 - **- name: nginx-container:** Asigna un nombre al contenedor dentro del Pod.
 - **image: nginx:latest:** Especifica la imagen del contenedor que se utilizará. En este caso, la última versión de la imagen oficial de `nginx` desde Docker Hub.
 - **ports::** Define los puertos que el contenedor expone:
 - **- containerPort: 80:** Indica que el contenedor Nginx dentro del Pod está escuchando en el puerto 80.

3.2: Desplegar el Pod

Aplicar el archivo YAML:

Ahora, utilizamos **kubectl** para aplicar este archivo YAML y crear el Pod en tu clúster de Minikube. Ejecutamos el siguiente comando en la misma ubicación donde guardé el archivo **nginx-pod.yaml**:

```
kubectl apply -f nginx-pod.yaml
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl apply -f nginx-pod.yaml
pod/nginx-pod created
```

Esto confirma que Kubernetes ha recibido tu solicitud y ha creado el Pod.

Verificar el Pod:

Para asegurarme de que el Pod se ha creado correctamente y está en funcionamiento, uso el siguiente comando:

```
kubectl get pods
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0           2m32s
```

Explicación de la salida:

- **NAME:** Muestra el nombre del Pod (nginx-pod).
- **READY:** Indica cuántos contenedores están listos dentro del Pod y cuántos deberían estarlo (en este caso, 1/1 significa que un contenedor está listo de un total de uno).
- **STATUS:** Muestra el estado actual del Pod. Running indica que el contenedor se está ejecutando correctamente. Otros estados comunes incluyen Pending (si el Pod aún no se ha iniciado) o Error (si hubo algún problema al iniciar el Pod).
- **RESTARTS:** Muestra cuántas veces se han reiniciado los contenedores dentro del Pod. Un valor de 0 es lo esperado en este caso.
- **AGE:** Indica cuánto tiempo lleva el Pod en funcionamiento.

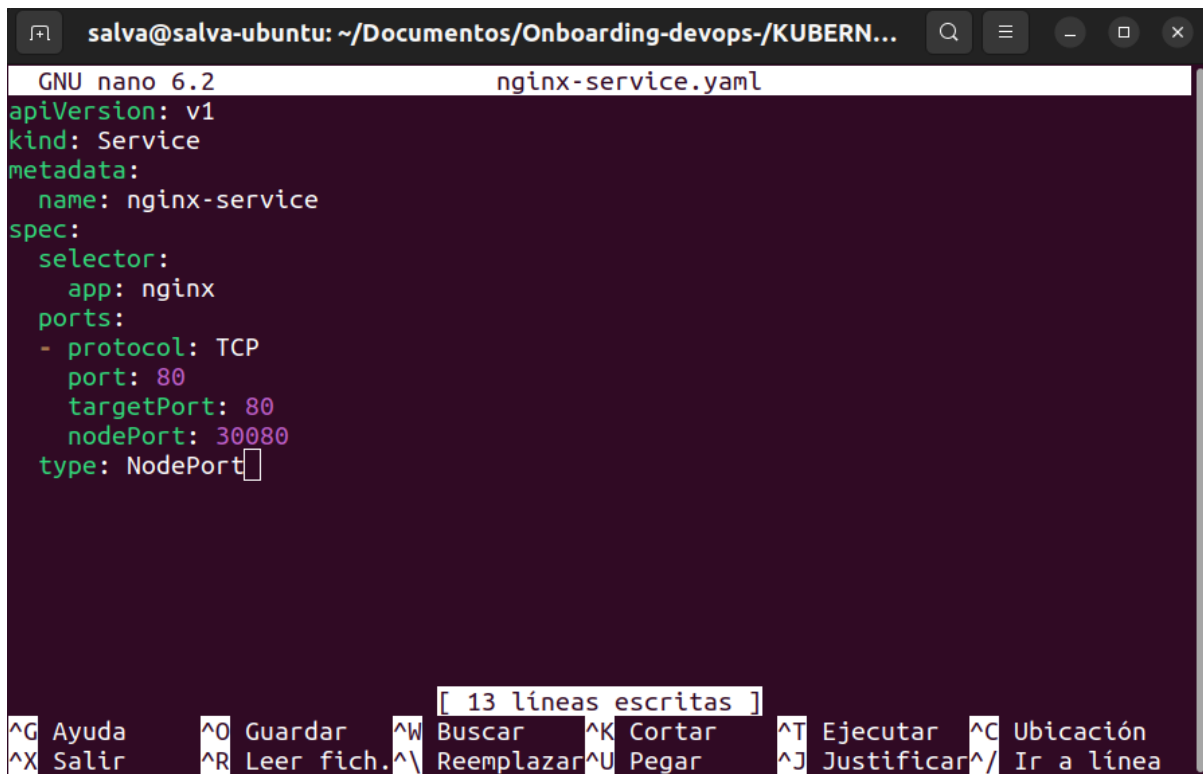
Ya está desplegado el Pod de Nginx en Kubernetes!

Paso 4: Exponer el Pod para acceder a la aplicación

Para que puedas acceder a tu aplicación Nginx desde fuera del clúster de Kubernetes, necesitas crear un Service.

4.1: Crear un servicio para exponer el Pod

Creemos un nuevo archivo YAML llamado **nginx-service.yaml** con el siguiente contenido:



```
GNU nano 6.2 nginx-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30080
  type: NodePort
```

[13 líneas escritas]

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea

Desglose del archivo YAML:

- **apiVersion: v1:** Especifica la versión de la API de Kubernetes.
- **kind: Service:** Indica que el tipo de objeto que se está definiendo es un Service.
- **metadata::** Contiene metadatos sobre el Service:
 - **name: nginx-service:** Asigna un nombre al Service.
- **spec::** Define la especificación del Service:
 - **selector::** Se utiliza para seleccionar los Pods a los que este Service dirigirá el tráfico. En este caso, seleccionará los Pods que tienen la etiqueta `app: nginx` (que es la etiqueta que le puse a mi Pod de Nginx).
 - **ports::** Define la lista de puertos que el Service expondrá:
 - **protocol: TCP:** Especifica el protocolo de red (TCP en este caso).

- **port: 80:** Es el puerto en el que el Service será accesible dentro del clúster.
- **targetPort: 80:** Es el puerto en el que la aplicación (el contenedor Nginx) está escuchando dentro del Pod.
- **nodePort: 30080:** Este es el puerto que se abrirá en cada uno de los nodos de tu clúster de Kubernetes (en este caso, mi máquina local con Minikube). *Se puede elegir un puerto diferente dentro del rango 30000-32767 si el 30080 ya está en uso.*
- **type: NodePort:** Especifica el tipo de Service. NodePort hace que el Service sea accesible en un puerto estático en cada IP de nodo.

4.2: Aplicar el archivo YAML para el servicio

Aplicar el archivo YAML del servicio:

Ejecutamos el siguiente comando la terminal, en la misma ubicación donde guardé el archivo **nginx-service.yaml**:

```
kubectl apply -f nginx-service.yaml
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl apply -f nginx-service.yaml
service/nginx-service created
```

Esto confirma que el Service ha sido creado.

Verificar el servicio:

Para comprobar que el Service se ha creado correctamente, utilizaré el siguiente comando:

```
kubectl get services
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	118m
nginx-service	NodePort	10.100.28.197	<none>	80:30080/TCP	88s

Explicación de la salida:

- **NAME:** Muestra el nombre del Service (nginx-service).
- **TYPE:** Indica el tipo de Service (NodePort).

- **CLUSTER-IP:** Es una dirección IP interna dentro del clúster que se utiliza para el balanceo de carga entre los Pods seleccionados por el Service.
- **EXTERNAL-IP:** En el caso de NodePort en un entorno local como Minikube, esta columna generalmente permanece en <none>.
- **PORT(S):** Muestra la asignación de puertos. Aquí vemos 80:30080/TCP, lo que significa que el tráfico que llega al puerto 30080 en cualquier IP de nodo será redirigido al puerto 80 dentro de los Pods seleccionados.
- **AGE:** Indica cuánto tiempo lleva el Service en funcionamiento.

4.3: Acceder a la aplicación

Acceder al servicio:

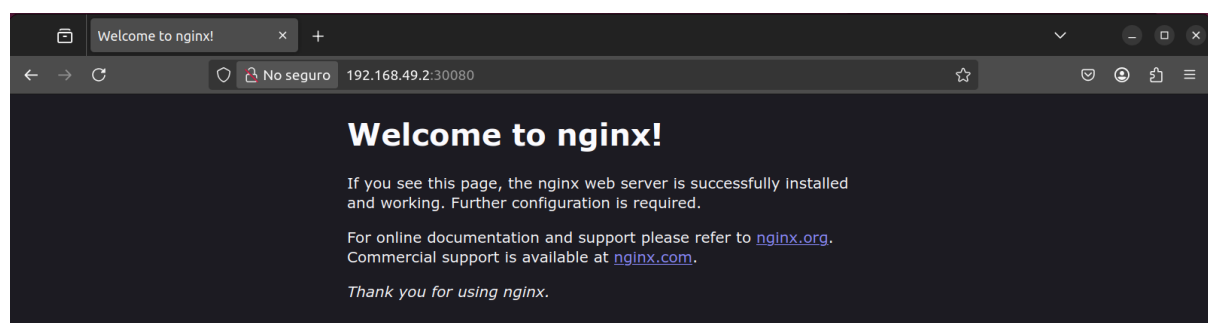
Como estoy usando Minikube, la forma más sencilla de acceder nuestro servicio **NodePort** es utilizando el comando **minikube service**. Ejecuto el siguiente comando:

```
minikube service nginx-service --url
```

Este comando devolverá la URL externa a través de la cual puedo acceder a mi aplicación Nginx.

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ minikube service nginx-service --url
http://192.168.49.2:30080
```

Copio la dirección en mi navegador web y muestra la página de bienvenida por defecto de Nginx, lo que indica que hemos accedido exitosamente a tu aplicación desplegada en Kubernetes a través del Service NodePort.



Paso 5: Limpiar los recursos

Eliminar los recursos:

1. **Eliminar el Servicio:** Para eliminar el Service **nginx-service**, ejecuto el siguiente comando:

```
kubectl delete service nginx-service
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl delete service nginx-service
service "nginx-service" deleted
```

2. **Eliminar el Pod:** Para eliminar el Pod **nginx-pod**, ejecuto el siguiente comando:

```
kubectl delete pod nginx-pod
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl delete pod nginx-pod
pod "nginx-pod" deleted
```

Verificar que se hayan eliminado:

1. **Verificar el Servicio:** Ejecuto el comando para listar los servicios:

```
kubectl get services
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes    ClusterIP     10.96.0.1     <none>         443/TCP    141m
```

Vemos que el servicio **nginx-service** ya no aparece en la lista (aparte del servicio **kubernetes** que es un servicio interno).

2. **Verificar el Pod:** Ejecuto el comando para listar los pods:

```
kubectl get pods
```

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ kubectl get pods
No resources found in default namespace.
```

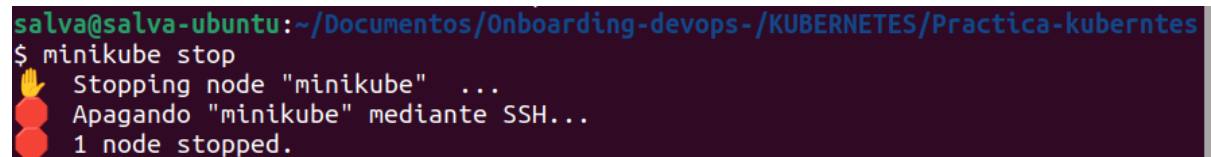
Vemos que el pod **nginx-pod** ya no aparece en la lista.

Detener el clúster de Minikube

Si ya no vamos a utilizar tu clúster de Kubernetes local por ahora, podemos detener Minikube con el siguiente comando:

minikube stop

Esto detendrá el contenedor donde se está ejecutando tu clúster.

A terminal window with a dark purple background. The prompt is 'salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes'. The command '\$ minikube stop' has been entered. The output shows a yellow hand icon, followed by 'Stopping node "minikube" ...', two red circle icons, followed by 'Apagando "minikube" mediante SSH...', and finally '1 node stopped.'

```
salva@salva-ubuntu:~/Documentos/Onboarding-devops-/KUBERNETES/Practica-kuberntes
$ minikube stop
👉 Stopping node "minikube" ...
🔴🔴 Apagando "minikube" mediante SSH...
1 node stopped.
```

Completados todos los pasos para instalar, iniciar, desplegar una aplicación y limpiar los recursos en mi máquina con Ubuntu 22.04 utilizando Minikube y kubectl.