Actividad 0: Verificación del funcionamiento de Docker, Docker, Desktop, Minikube, Kind

Objetivos

1. Comprender la virtualización y la contenerización: Entender las diferencias fundamentales entre la virtualización tradicional y la contenerización, y cómo Docker revoluciona el desarrollo y la implementación de aplicaciones mediante la creación de entornos ligeros y portátiles.
2. Familiarización con Docker y Docker Desktop: Aprender a instalar y configurar Docker y Docker Desktop en Linux y cómo pueden facilitar el desarrollo y la distribución de aplicaciones.
3. Explorar Kubernetes con Minikube y Kind: Entender los conceptos básicos de Kubernetes como sistema de orquestación de contenedores, y aprender a utilizar Minikube y Kind para crear y gestionar clústeres de Kubernetes en un entorno local. Esto incluye comprender la arquitectura de Kubernetes, los objetos de Kubernetes (pods, servicios, despliegues, etc.) y cómo se pueden manejar a través de estas herramientas.

Prueba de Docker Engine

1. Descarga e instale Docker Desktop: https://docs.docker.com/desktop/install/ubuntu/
2. Ahora que ha instalado Docker Desktop con éxito, probémoslo. Comenzaremos ejecutando un contenedor Docker simple directamente desde la línea de comando. Abra una ventana de Terminal y ejecute el siguiente comando:

   $ docker version

3. Para ver si puede ejecutar contenedores, ingrese el siguiente comando en la ventana de Terminal y presione Enter:

   $ docker container run hello-world

```
alumno@administrador-20VE:~$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Si lee atentamente el resultado anterior, habrá notado que Docker no encontró una imagen llamada hello-world:latest y, por lo tanto, decidió descargarla desde un registro de imágenes de Docker. Una vez descargado, Docker Engine creó un contenedor a partir de la imagen y lo ejecutó. La aplicación se ejecuta dentro del contenedor y luego genera todo el texto, comenzando con Hello from Docker!

Esta es una prueba de que Docker está instalado y funcionando correctamente en su máquina.

4. Probemos con otra imagen de prueba divertida que normalmente se usa para verificar la instalación de Docker. Ejecute el siguiente comando:

$ docker container run rancher/cowsay Hello

```
alumno@administrador-20VE:~$ docker container run rancher/cowsay Hello
 _______
< Hello >
 -------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Genial: hemos confirmado que Docker Engine funciona en nuestra computadora local. Ahora, asegurémonos de que lo mismo ocurre con Docker Desktop.

Habilitar Kubernetes en Docker Desktop

Docker Desktop viene con soporte integrado para Kubernetes.

¿Qué es Kubernetes?

Kubernetes es una poderosa plataforma para automatizar la implementación, el escalado y la gestión de aplicaciones en contenedores. Ya seas desarrollador, ingeniero de DevOps o administrador de sistemas, Kubernetes proporciona las herramientas y abstracciones que necesita para administrar sus contenedores y aplicaciones de manera escalable y eficiente. Este soporte está desactivado de forma predeterminada. Pero no te preocupes: es muy fácil de activar:

1. Abra el panel de Docker Desktop.
2. En la esquina superior izquierda, seleccione el ícono de la rueda dentada. Esto abrirá la página de configuración (setting).
3. En el lado izquierdo, seleccione la pestaña Kubernetes y luego marque la casilla Enable Kubernetes
4. Haga clic en el botón Apply & restart.

Ahora, deberá tener paciencia ya que Docker está descargando toda la infraestructura de soporte y luego inicia Kubernetes.
Una vez que Docker se haya reiniciado, estará listo para usar Kubernetes.

Probando minikube y kubectl

Descarga e instala y minikube https://minikube.sigs.k8s.io/docs/start/ y kubectl https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/ y la opción Install using native package management

Ahora escribe minikube start para iniciar el cluster por defecto . La primera vez que haga esto, llevará un tiempo ya que minikube necesita descargar todos los binarios de Kubernetes. Cuando termine, la última línea del resultado en su pantalla debería ser algo como esto:

Empecemos. Siga estos pasos cuidadosamente:
1. Intentemos acceder a nuestro clúster usando kubectl. Primero, debemos asegurarnos de tener seleccionado el contexto correcto para kubectl. Si anteriormente instaló Docker Desktop y ahora minikube, puede usar el siguiente comando:

```
alumno@administrador-20VE:~$ minikube start
W0409 11:44:04.696245   23261 main.go:291] Unable to resolve the current Docker
CLI context "default": context "default": context not found: open /home/alumno/.
docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f
0688f/meta.json: no such file or directory
😄  minikube v1.32.0 en Ubuntu 23.04
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🤷  docker "minikube" container is missing, will recreate.
🔥  Creating docker container (CPUs=2, Memory=3716MB) ...
🐳  Preparando Kubernetes v1.28.3 en Docker 24.0.7...
    ▪ 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
```

$ kubectl config get-contexts

```
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$ kubectl config get-contexts
CURRENT   NAME             CLUSTER          AUTHINFO         NAMESPACE
          docker-desktop   docker-desktop   docker-desktop
*         minikube         minikube         minikube         default
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$
```

El asterisco al lado del contexto llamado minikube nos dice que este es el contexto actual. Así, al usar kubectl, trabajaremos con el nuevo cluster creado por minikube.

```
alumno@administrador-20VE:~$ kubectl config get-contexts
CURRENT   NAME             CLUSTER          AUTHINFO         NAMESPACE
          docker-desktop   docker-desktop   docker-desktop
*         minikube         minikube         minikube         default
```

2. Ahora veamos cuántos nodos tiene nuestro cluster con este comando: $kubectl get nodes

```
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   18h   v1.28.3
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$
```

Deberías obtener algo similar a esto. Tenga en cuenta que la versión mostrada podría diferir en su caso:

```
alumno@administrador-20VE:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE    VERSION
minikube   Ready    control-plane   104s   v1.28.3
```

Aquí tenemos un clúster de un solo nodo. El papel del nodo es el del plano de control, lo que significa que es un nodo maestro. Un clúster de Kubernetes típico consta de unos pocos nodos maestros y muchos nodos trabajadores. La versión de Kubernetes con la que estamos trabajando aquí es la v1.28.3.

3. Ahora, intentemos ejecutar algo en este clúster. Usaremos Nginx, un servidor web popular para esto. Utiliza el archivo .yaml, que acompaña a la actividad que vamos a utilizar para esta prueba:

Abra una nueva ventana de Terminal y crea un pod que ejecute Nginx con el siguiente comando:

$ kubectl apply -f nginx.yaml

```
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$ kubectl apply -f nginx.yaml
pod/nginx created
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$
```

Deberías ver este resultado:

pod/nginx created

4. Podemos verificar si el pod se está ejecutando con kubectl:

$ kubectl get pods

Deberíamos ver esto:

```
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          2m9s
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$
```

Esto indica que tenemos 1 pod con Nginx ejecutándose y que se ha reiniciado 0 veces.

```
alumno@administrador-20VE:~$ kubectl  apply -f nginx.yaml
pod/nginx created
alumno@administrador-20VE:~$ kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          16s
```

5. Para acceder al servidor Nginx, necesitamos exponer la aplicación que se ejecuta en el pod con el siguiente comando:

$ kubectl expose pod nginx --type=NodePort --port=80

```
service/nginx exposed
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$
```

Esta es la única forma en que podemos acceder a Nginx desde nuestra computadora portátil, por ejemplo, a través de un navegador. Con el comando anterior, estamos creando un servicio de Kubernetes, como se indica en el resultado generado para el comando:

service/nginx exposed

```
alumno@administrador-20VE:~$ kubectl expose pod nginx --type=NodePort --port=80
service/nginx exposed
```

6. Podemos usar kubectl para enumerar todos los servicios definidos en nuestro

clúster: $ kubectl get services

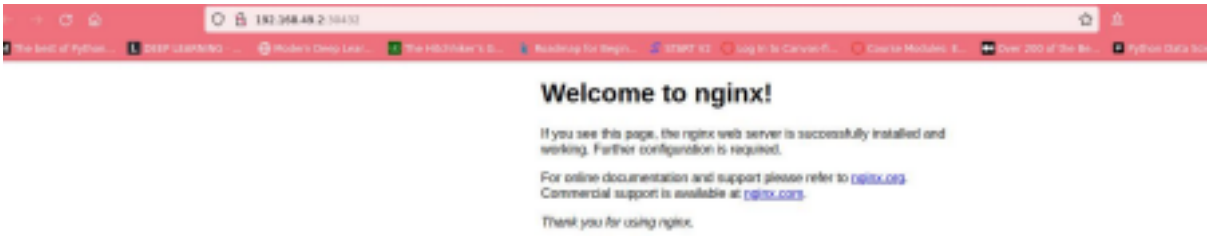
```
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$ kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        18h
nginx        NodePort    10.108.34.206   <none>        80:30432/TCP   99s
(base) clara@lara:~/Desktop/Curso-Paralelismo-distribucion/Actividad0-C8286$
```

En el resultado anterior, podemos ver el segundo servicio llamado Nginx, que acabamos de crear. El servicio es del tipo NodePort; El puerto 80 del pod se había asignado al puerto 30432 del nodo del clúster de nuestro clúster de Kubernetes en minikube.

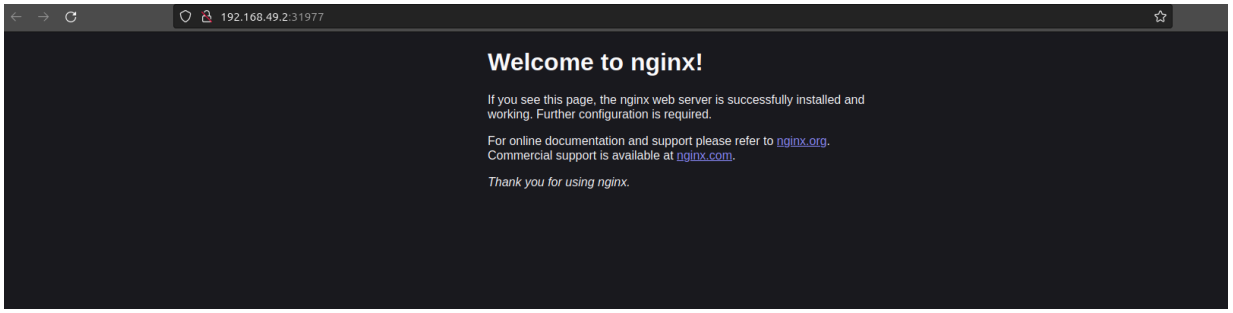
7. Ahora, podemos usar minikube para crear un túnel hacia nuestro clúster y abrir un navegador con la URL correcta para acceder al servidor web Nginx. Utilice este comando:

$ minikube service nginx

El resultado en su ventana de Terminal será el siguiente:

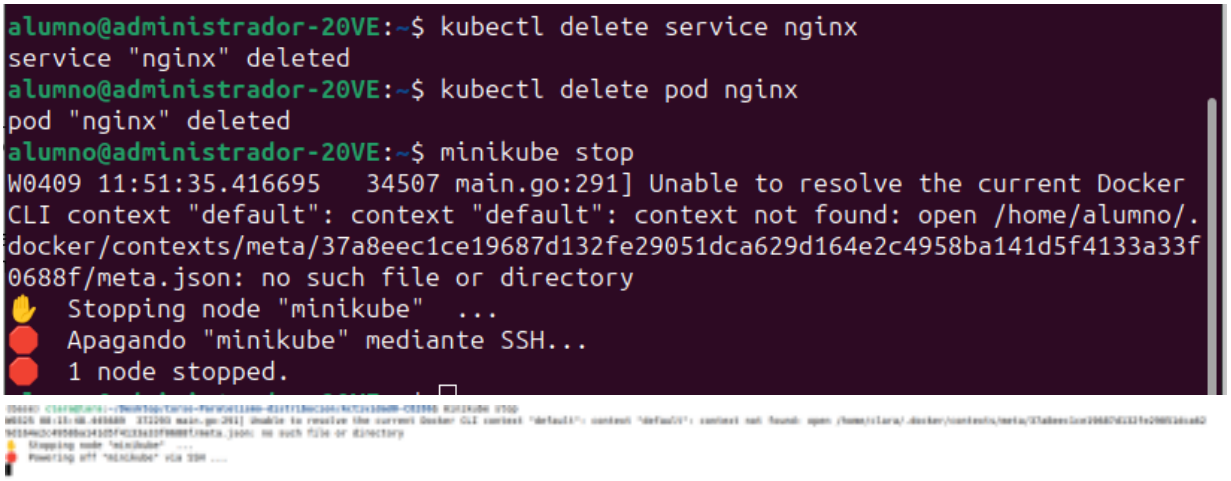


El resultado anterior muestra que minikube creó un túnel para el servicio nginx que escucha en el puerto del nodo 30432 que está en nuestra computadora portátil.



Hemos ejecutado y accedido con éxito a un servidor web Nginx en nuestro pequeño clúster de Kubernetes de un solo nodo en minikube! Una vez que hayas terminado de jugar, es hora de limpiar:

• Detenga el túnel hacia el clúster presionando Ctrl + C dentro de la ventana de Terminal.
• Elimine el servicio nginx y el pod en el clúster: $ kubectl delete service nginx $ kubectl delete pod nginx
• Detenga el clúster con el siguiente comando: minikube stop
• Deberías ver esto:



Ejercicios

A veces, probar con un clúster de un solo nodo no es suficiente. minikube lo resuelve. Siga estas instrucciones para crear un verdadero clúster de Kubernetes de múltiples nodos en minikube:

1. Si queremos trabajar con un clúster que consta de varios nodos en minikube, podemos usar este comando:

$ minikube start --nodes 3 –p demo

El comando anterior crea un clúster con tres nodos y lo llama demo.

```
alumno@administrador-20VE:~$ minikube start --nodes 3 -p demo
W0409 11:52:47.896014   35344 main.go:291] Unable to resolve the current Docker
CLI context "default": context "default": context not found: open /home/alumno/.
docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f
0688f/meta.json: no such file or directory
😄  [demo] minikube v1.32.0 en Ubuntu 23.04
✨  Controlador docker seleccionado automáticamente. Otras opciones: qemu2, ssh
📌  Using Docker driver with root privileges
👍  Starting control plane node demo in cluster demo
🚜  Pulling base image ...
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparando Kubernetes v1.28.3 en Docker 24.0.7...
    ▪ Generando certificados y llaves
    ▪ Iniciando plano de control
    ▪ Configurando reglas RBAC...
🔗  Configurando CNI CNI ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Complementos habilitados: storage-provisioner, default-storageclass

👍  Starting worker node demo-m02 in cluster demo
🚜  Pulling base image ...
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🌐  Se han encontrado las siguientes opciones de red:
    ▪ NO_PROXY=192.168.58.2
🐳  Preparando Kubernetes v1.28.3 en Docker 24.0.7...
    ▪ env NO_PROXY=192.168.58.2
🔎  Verifying Kubernetes components...

👍  Starting worker node demo-m03 in cluster demo
🚜  Pulling base image ...
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🌐  Se han encontrado las siguientes opciones de red:
    ▪ NO_PROXY=192.168.58.2,192.168.58.3
🐳  Preparando Kubernetes v1.28.3 en Docker 24.0.7...
    ▪ env NO_PROXY=192.168.58.2
```

2. Utilice kubectl para enumerar todos los nodos de su clúster:

$ kubectl get

Tenemos un clúster de 3 nodos donde el nodo **demo** es un nodo maestro y los dos nodos restantes son nodos de trabajo.

```
alumno@administrador-20VE:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
demo       Ready    control-plane   2m35s   v1.28.3
demo-m02   Ready    <none>          2m16s   v1.28.3
demo-m03   Ready    <none>          2m7s    v1.28.3
```

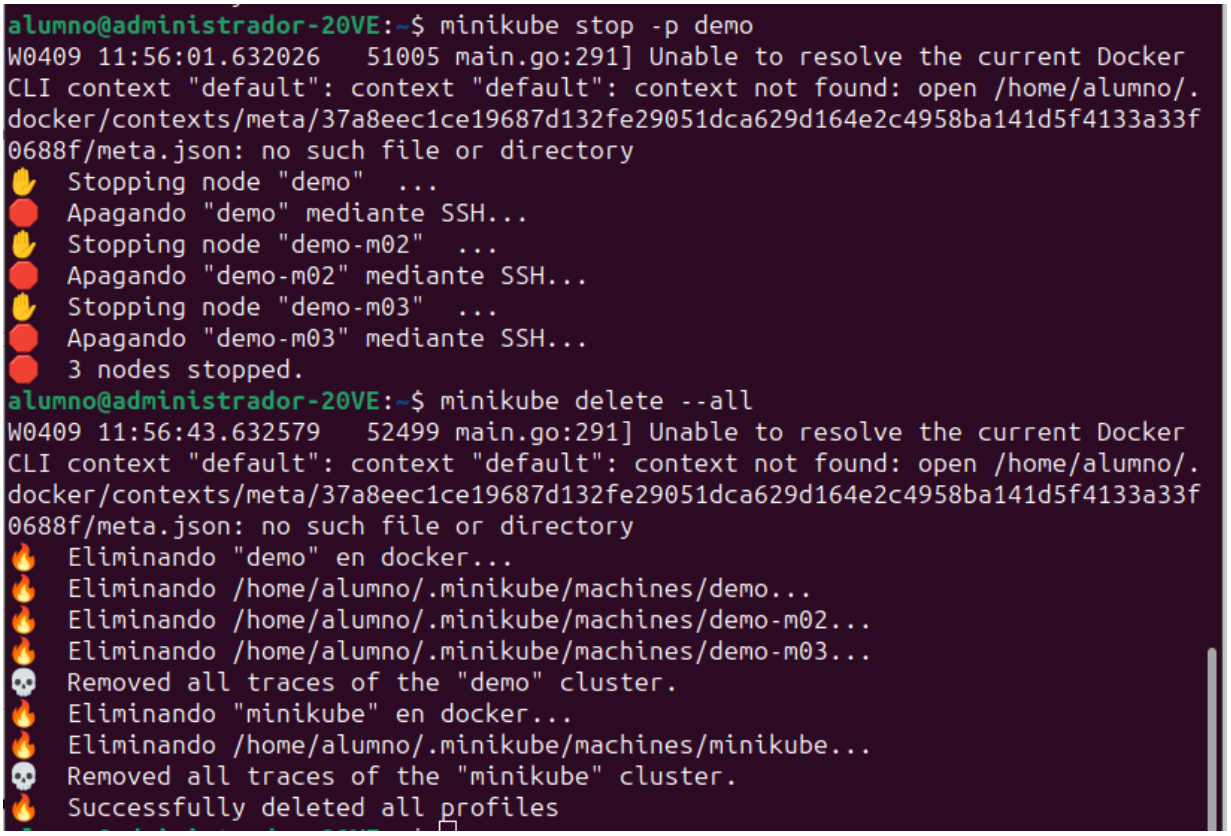
3. No vamos a continuar con este ejemplo aquí, así que use el siguiente comando para detener el clúster:

$ minikube stop -p demo

4. Elimine todos los clústeres de su sistema con este comando:

$ minikube delete --all

Esto eliminará el clúster predeterminado (llamado minikube) y el clúster **demo** en nuestro caso.



Con esto, pasaremos a la siguiente herramienta interesante y útil a la hora de trabajar con contenedores y Kubernetes. Debería tenerlo instalado y disponible en la computadora de su trabajo.

Kind

Kind (https://kind.sigs.k8s.io/docs/user/quick-start) es otra herramienta popular que se puede utilizar para ejecutar un clúster de Kubernetes de múltiples nodos localmente en su máquina. Es muy fácil de instalar y usar.

Vamos:

1. En una máquina Linux, puedes usar el siguiente script para instalar Kind desde sus archivos binarios:

curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.22.0/kind-linux-amd64

$chmod +x ./kind

$sudo mv ./kind /usr/local/bin/kind

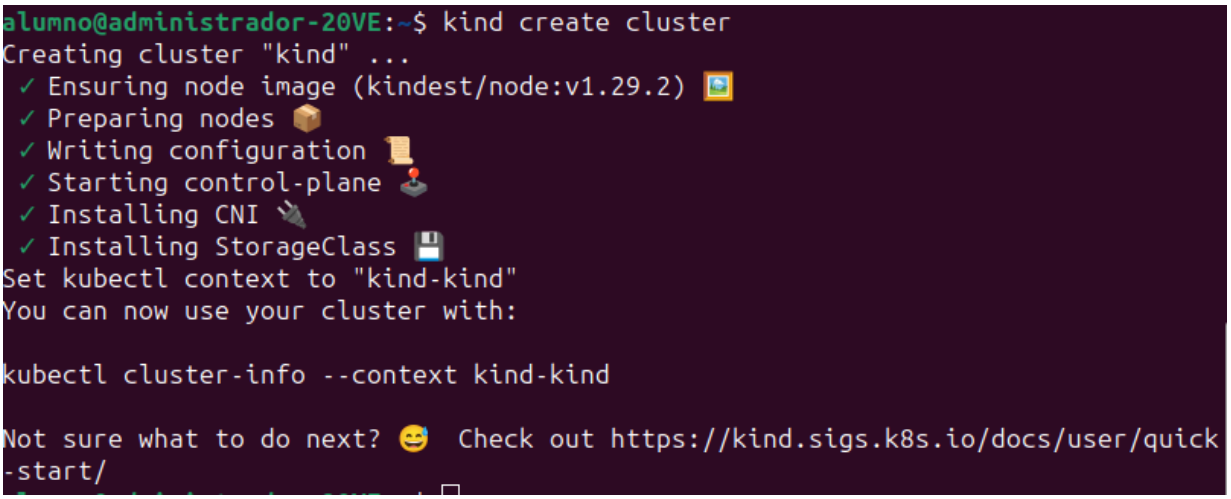
2. Una vez instalado Kind, pruébelo con el siguiente comando:

$ kind version

3. Ahora, intente crear un clúster de Kubernetes simple que consta de un nodo maestro y dos nodos trabajadores. Utilice este comando para lograr esto:

$ kind create cluster

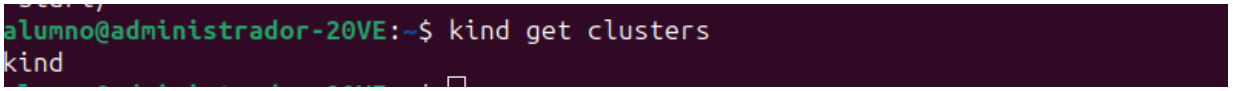
Después de un tiempo, deberías ver este resultado:



4. Para verificar que se ha creado un clúster, utilice este comando:

$ kind get clusters

El resultado anterior muestra que hay exactamente un clúster llamado **kind,** que es el nombre predeterminado.



5. Podemos crear un clúster adicional con un nombre diferente usando el parámetro -- **name,** así:

$ kind create cluster --name demo

```
alumno@administrador-20VE:~$ kind create cluster --name demo
Creating cluster "demo" ...
 ✓ Ensuring node image (kindest/node:v1.29.2) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-demo"
You can now use your cluster with:

kubectl cluster-info --context kind-demo

Have a nice day! 👋
```

6. Al enumerar los clústeres se mostrará esto:

$ kind get clusters

```
alumno@administrador-20VE:~$ kind get clusters
demo
kind
```

Y esto funciona como se esperaba.

Ahora que hemos usado kind para crear dos clústeres de muestra, usemos kubectl para jugar con uno de los clústeres y ejecutar la primera aplicación en él. Usaremos Nginx para esto, similar a lo que hicimos con minikube:

Ahora podemos usar kubectl para acceder y trabajar con los clústeres que acabamos de crear. Mientras creaba un clúster, Kind también actualizó el archivo de configuración de nuestro kubectl. Podemos verificar esto con el siguiente comando:

$ kubectl config get-contexts

Debería producir el siguiente resultado:

```
alumno@administrador-20VE:~$ kubectl config get-contexts
CURRENT   NAME             CLUSTER          AUTHINFO         NAMESPACE
          docker-desktop   docker-desktop   docker-desktop
*         kind-demo        kind-demo        kind-demo
          kind-kind        kind-kind        kind-kind
```

Puede ver que los clústeres **kind** y de **demo** son parte de la lista de clústeres conocidos y que el clúster de demo es el contexto actual para kubectl.

2. Utilice el siguiente comando para convertir el clúster de demo en su clúster actual si el asterisco indica que hay otro clúster actual:

$ kubectl config use-context kind-demo

```
alumno@administrador-20VE:~$ kubectl config use-context kind-demo
Switched to context "kind-demo".
```


3. Enumeremos todos los nodos del clúster de muestra:

$ kubectl get nodes

La salida debería ser así:

```
alumno@administrador-20VE:~$ kubectl get nodes
NAME                 STATUS   ROLES           AGE     VERSION
demo-control-plane   Ready    control-plane   3m52s   v1.29.2
```

4. Ahora, intentemos ejecutar el primer contenedor en este clúster. Usaremos nuestro servidor web Nginx de confianza, como hicimos antes. Utilice el siguiente comando para ejecutarlo:

$ kubectl apply -f nginx.yaml

El resultado debería ser el siguiente:

pod/nginx created

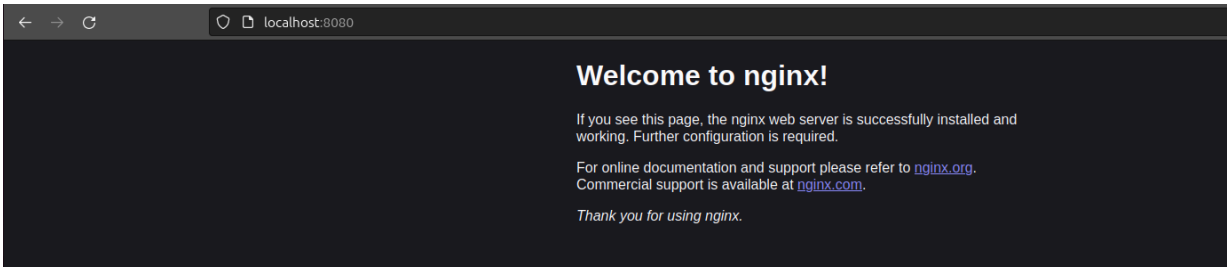
```
alumno@administrador-20VE:~$ kubectl apply -f nginx.yaml
pod/nginx created
```

5. Para acceder al servidor Nginx, necesitamos realizar el reenvío de puertos usando kubectl. Utilice este comando para hacerlo:

Revisa: kubectl describe pod nginx, https://www.kristhecodingunicorn.com/post/kubernetes port-forwarding-cleanup-of-orphaned-ports/

$ kubectl port-forward nginx 8080 80 (puedes usar otros puertos)

Abra una nueva pestaña del navegador y navegue hasta http://localhost:8080; Deberías ver la pantalla de bienvenida de Nginx



Una vez que hayas terminado de jugar con Nginx, usa este comando para eliminar el pod del clúster:

$ kubectl delete -f nginx.yaml

Antes de continuar, limpiemos y eliminemos los dos clústeres que acabamos de crear:

$ kind delete cluster --name kind

$ kind delete cluster --name demo

Con esto, hemos instalado todas las herramientas que necesitaremos para trabajar exitosamente con contenedores en nuestra máquina local.

```
alumno@administrador-20VE:~$ kind delete cluster --name kind
Deleting cluster "kind" ...
Deleted nodes: ["kind-control-plane"]
alumno@administrador-20VE:~$ kind delete cluster --name demo
Deleting cluster "demo" ...
Deleted nodes: ["demo-control-plane"]
```

Comunicación de Datos y Redes