Introducción a Kubernetes

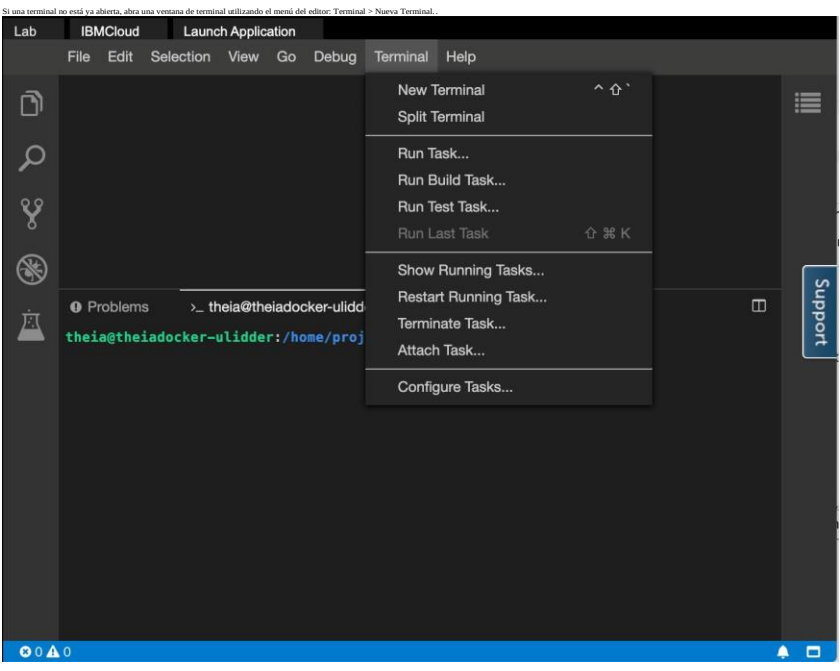
Objetivos

En este laboratorio, usted:

- Usará el kubectl CLI
- Creará un pod de Kubernetes
- Creará un Deployment de Kubernetes
- Creará un ReplicaSet que mantenga un número determinado de replicas
- Presenciará el equilibrio de carga de los kubernetes en acción

Verificar el entorno y las herramientas de la línea de mando

1. Si una terminal no está ya abierta, abra una ventana de terminal utilizando el menú del editor: Terminal > Nueva Terminal..



2. Verifique que el CLI de kubectl esté instalado.

```
kubectl version
```

{: codeblock}

Debería ver una salida similar a esta, aunque las versiones pueden ser diferentes:

```
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2", GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", G
itTreeState:"clean", BuildDate:"2020-01-18T23:30:10Z", GoVersion:"go1.13.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.10+ks5", GitCommit:"a0052bd119c067cf48e8a19f0ab7d5a5e2ca0a1
8", GitTreeState:"clean", BuildDate:"2020-05-20T20:48:06Z", GoVersion:"go1.13.9", Compiler:"gc", Platform:"linux/amd64"}
```

3. Cambiar a la carpeta de proyectos.

```
cd /home/project
```

{: codeblock}

4. Clonar el depósito de git que contiene los artefactos necesarios para este laboratorio, si no existe ya.

```
[ ! -d 'cc201' ] && git clone https://gitlab.com/ibm/skills-network/courses/cc201.git
```

{: codeblock}

5. Cambie al directorio de este laboratorio.

```
cd cc201/labs/2_IntroKubernetes/
```

{: codeblock}

6. Enumere el contenido de este directorio para ver los artefactos de este laboratorio.

```
ls
```

{: codeblock}

Use el `kubectl` CLI

Recuerde que los espacios de nombres de Kubernetes le permiten virtualizar un clúster. Ya tiene acceso a un espacio de nombres en un clúster de Kubernetes, y kubectl ya está configurado para dirigirse a ese clúster y espacio de nombres.

Veamos algunos comandos básicos de kubectl.

1. kubectl requiere una configuración para que apunte al cúmulo apropiado. Obtener información del cluster con el siguiente comando:

```
kubectl config get-clusters
```

{: codeblock}

2. Un contexto kubectl es un grupo de parámetros de acceso, que incluye un clúster, un usuario y un espacio de nombres. Vea su contexto actual con el siguiente comando:

```
kubectl config get-contexts
```

{: codeblock}

3. Enumere todos los Pods en su espacio de nombres. Si esta es una nueva sesión para usted, no verá ningún Pod.

```
kubectl get pods
```

{: codeblock}

Crear una cápsula con un comando imperativo

Now it's time to create your first Pod. This Pod will run the `hello-world` image you built and pushed to IBM Cloud Container
Ahora es el momento de crear su primer Pod. Este Pod ejecutará la imagen del hola-mundo que construyó y empujó al Registro de Contenedores de IBM Cloud en el último laboratorio. Como se explica en los vídeos de este módulo, puede crear un Pod de forma imperativa o declarativa. Hagámoslo imperativamente primero.

1. Compruebe si su espacio de nombre sigue siendo una variable de entorno después del primer laboratorio.

```
echo $MY_NAMESPACE
```

{: codeblock}

2. Si no está configurado, exporte su namespace como una variable de entorno para que pueda ser utilizado en comandos posteriores. Asegúrese de sustituir su espacio de nombres después del signo de igualdad. Si no recuerda su espacio de nombres, ejecute ibmcloud cr namespaces.

```
export MY_NAMESPACE=<my_namespace>
```

{: codeblock}

3. Construya y pulse la imagen de nuevo, ya que puede haber sido borrada automáticamente desde que completó el primer laboratorio.

```
docker build -t us.icr.io/$MY_NAMESPACE/hello-world:1 . && docker push us.icr.io/$MY_NAMESPACE/hello-world:1
```

{: codeblock}

4. Ejecutar la imagen de hola mundo como un contenedor en Kubernetes.

```
kubectl run hello-world --image us.icr.io/$MY_NAMESPACE/hello-world:1 --overrides='{"spec":{"template":{"spec":{"imagePullSecrets":[{"name":"icr"}]}}}}'
```

{: codeblock}

La opción --overrides aquí nos permite especificar las credenciales necesarias para sacar esta imagen del Registro de Contenedores de IBM Cloud. Tenga en cuenta que este es un comando imperativo, ya que le dijimos a los gobernantes explícitamente qué hacer: ejecutar hola-mundo

5. Enumere los Pods en su namespace.

```
kubectl get pods
```

{: codeblock}

Genial, el comando anterior de hecho creó una cápsula para nosotros. Puede ver que se le dio un nombre autogenerado a

este Pod. También puede especificar la opción amplia para la salida para obtener más detalles sobre el recurso.

```
kubectl get pods -o wide
```

{: codeblock}

6. Note the Pod name from the previous step, and describe the Pod to get more details about it.

```
kubectl describe pod <pod_name>
```

{: codeblock}

Echa un vistazo a esta salida... hay mucho ahí. Si mira de cerca, notará que hay un ReplicaSet asociado a este pod. Esto se debe a que el comando de ejecución de kubectl en realidad creó un Deployment con una réplica, que a su vez creó un ReplicaSet. Al final de la salida, también verá eventos. Estos dan algo de historia para este recurso. Por ejemplo, debería ver eventos que indiquen que este Pod fue programado, la imagen fue sacada, y el contenedor fue iniciado.

7. Enumere los Deployments y ReplicaSets en su espacio de nombres para verificar que uno de cada uno fue creado..

```
kubectl get deployments,replicasets
```

{: codeblock}

8. Borre el deployment. Esto también eliminará el ReplicaSet y el Pod.

```
kubectl delete deployment hello-world
```

{: codeblock}

9. Enumere los pods para verificar que no existe ninguna.

```
kubectl get pods
```

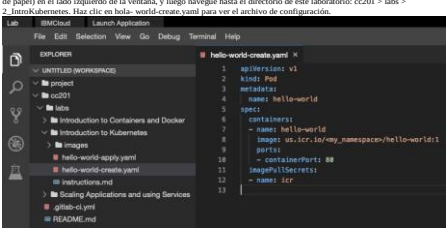
{: codeblock}

El ReplicaSet y el Pod fueron eliminados desde que se eliminó el Deployment propio..

Crear un pod con una configuración de objetos imperativa

La configuración de objetos imperativa permite crear objetos especificando la acción a realizar (por ejemplo, crear, actualizar, eliminar) mientras se utiliza un archivo de configuración. Un archivo de configuración, hello-world-create.yaml, se proporciona en este directorio..

1. Utilice el Explorador para ver y editar el archivo de configuración. Haga clic en el icono del Explorador (parece una hoja de papel) en el lado izquierdo de la ventana, y luego navegue hasta el directorio de este laboratorio: cc201 > labs > 2_IntroKubernetes. Haz clic en hola- world-create.yaml para ver el archivo de configuración.



2. Use el Explorador para editar hello-world-create.yaml. Necesita insertar su namespace donde dice <my_namespace>. Asegúrese de guardar el archivo cuando haya terminado.

3. Imperativamente cree un Pod usando el archivo de configuración proporcionado.

```
kubectl create -f hello-world-create.yaml
```

{: codeblock}

Tenga en cuenta que esto es realmente imperativo, ya que usted les dijo explícitamente a los gobernantes que crearan los recursos definidos en el archivo.

4. Enumere los Pods en su espacio de nombres.

```
kubectl get pods
```

{: codeblock}

En este caso, kubectl no cree un Deployment para nosotros porque el archivo YAML definió explícitamente un Pod.

5. Borrar el Pod.

```
kubectl delete pod hello-world
```

{: codeblock}

Este comando puede tardar un tiempo en ejecutarse.

Crear una cápsula con un comando declarativo

Las dos formas anteriores de crear un Pod eran imperativas... le dijimos explícitamente a Kubectl qué hacer. Mientras que los comandos imperativos son fáciles de entender y ejecutar, no son ideales para un entorno de producción. Veamos los comandos declarativos.

1. En este directorio se proporciona un archivo de muestra hello-world-apply.yaml. Use el Explorador de nuevo para abrir este archivo. Observe lo siguiente::

- Estamos creando un Deployment (`kind: Deployment`).
- Habrá tres réplicas para este Deployment (`replicas: 3`).
- Los Pods deben ejecutar la imagen de hola-mundo (`- image: us.icr.io/<my_namespace>/hello-world:1`). Puede ignorar el resto por ahora. Llegaremos a muchos de esos conceptos en el próximo laboratorio.

2. Use el explorador para editar el archivo hello-world-apply.yaml. Necesita insertar su namespace donde dice <my_namespace>. Asegúrese de guardar el archivo cuando termine.

3. Use el comando kubectl apply para establecer esta configuración como el estado deseado en Kubernetes.

```
kubectl apply -f hello-world-apply.yaml
```

{: codeblock}

4. Consigue los Deployment aseguren la creación de un Deployment.

```
kubectl get deployments
```

{: codeblock}

5. Enumere los pods para asegurarse de que existen tres réplicas.

```
kubectl get pods
```
{: codeblock}

Con la gestión declarativa, no le dijimos a los Kubernetes qué acciones realizar. En su lugar, Kubectl dedujo que este despliegue debía ser creado. Si borra un Pod ahora, se creará uno nuevo en su lugar para mantener tres réplicas.

6. Anote uno de los nombres de los Pods del paso anterior, y borre ese Pod.

```
kubectl delete pod <pod_name>
```
{: codeblock}
Este comando puede tardar un tiempo en ejecutarse.

7. Enumere los pods para ver cómo se crea una nueva.

```
kubectl get pods
```
{: codeblock}
Si lo hace con la suficiente rapidez, puede ver un Pod que se termina y otro que se crea.

```
NAME                     READY   STATUS              RESTARTS        AGE
hello-world-dd6b5d745-2jw5s   0/1     Terminating         0          35s
hello-world-dd6b5d745-f9xjk   1/1     Running             0          35s
hello-world-dd6b5d745-m89fc   0/1     ContainerCreating   0          8s
hello-world-dd6b5d745-qvs9t   1/1     Running             0          35s
```

De lo contrario, el estado de cada uno será el mismo, pero la edad de un Pod será menor que la de los otros.

```
NAME                     READY STATUS RESTARTS  AGE
hello-world-dd6b5d745-f9xjk   1/1     Running   0          39s
hello-world-dd6b5d745-m89fc   1/1     Running   0          12s
hello-world-dd6b5d745-qvs9t   1/1     Running   0          39s
```

Equilibrio de carga de la aplicación

Dado que hay tres réplicas de esta aplicación desplegadas en el clúster, Kubernetes cargará las solicitudes de saldo en estas tres instancias. Expongamos nuestra aplicación en Internet y veamos cómo los Kubernetes cargan las solicitudes de balance.

1. Para poder acceder a la aplicación, tenemos que exponerla a Internet utilizando un Servicio de Kubernetes.
```
kubectl expose deployment/hello-world --type=NodePort --port=8080 --name=hello-world --target-port=8080
```

{: codeblock}

Este comando crea lo que se llama un Servicio NodePort. Esto abrirá un puerto en el nodo trabajador para permitir que se acceda a la aplicación usando ese puerto y la dirección IP del nodo.

2. Listar Servicios para ver que este servicio fue creado.

```
kubectl get services
```

{: codeblock}

3. Se necesitan dos cosas para acceder a esta aplicación: una dirección IP del nodo trabajador y el puerto correcto. Para obtener la IP de un nodo trabajador, vuelva a ejecutar el comando get pods con la opción wide y anote cualquiera de las direcciones IP del nodo (de la columna titulada NODE):

```
kubectl get pods -o wide
```

{: codeblock}

Aquí hay algunas muestras de salida.

```
NAME                     READY   STATUS    RESTARTS   AGE   IP               NODE             NOMINATED NODE   READINESS GATES
hello-world-dd6b5d745-f9xjk   1/1     Running   0          7m    172.30.104.185   10.114.85.153    <none>           <none>
hello-world-dd6b5d745-m89fc   1/1     Running   0          7m    172.30.165.182   10.114.85.151    <none>           <none>
hello-world-dd6b5d745-qvs9t   1/1     Running   0          7m    172.30.69.68     10.114.85.172    <none>           <none>
```

Usando esta salida de muestra, podría elegir 10.114.85.153, 10.114.85.151, o 10.114.85.172 para la dirección IP del nodo.

4. Exportar la dirección IP del nodo como una variable de entorno. Asegúrese de sustituir la dirección IP copiada en este comando..

```
export NODE_IP=<node_ip>
```

{: codeblock}

Usando la salida de la muestra, un comando correcto sería exportar NODE_IP=10.114.85.153.

5. Para obtener el número de puerto, ejecute el siguiente comando y anote el puerto:

```
kubectl get services
```

{: codeblock}

Aquí hay algunas muestras de salida. En esta muestra, el número de puerto que necesitamos es 31758.

```
NAME         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
hello-world   NodePort   172.21.121.84   <none>        8080:31758/TCP   58s
```

6. Exportar el puerto como una variable de entorno. Asegúrese de sustituir el puerto copiado en este comando.

```
export NODE_PORT=<node_port>
```

{: codeblock}

Usando la salida de la muestra, el comando correcto sería exportar NODE_PORT=31758.

7. Ping a la solicitud para obtener una respuesta.

```
curl $NODE_IP:$NODE_PORT
```

{: codeblock}

8. Observe que esta salida incluye el nombre de la cápsula. Ejecute el comando diez veces y anote los diferentes nombres de la cápsula en cada línea de salida.

```
for i in `seq 10`; do curl $NODE_IP:$NODE_PORT; done
```

{: codeblock}

Debería ver más de un nombre de la cápsula, y muy posiblemente los tres nombres de la cápsula, en la salida. Esto se debe a que la carga de Kubernetes equilibra las peticiones a través de las tres réplicas, por lo que cada petición podría golpear una instancia diferente de nuestra aplicación.

9. Eliminar el Deployment y el Servicio. Esto puede hacerse en un solo comando usando barras oblicuas.

```
kubectl delete deployment/hello-world service/hello-world
```

{: codeblock}

¡Felicidades! Ha completado el laboratorio para el segundo módulo de este curso.