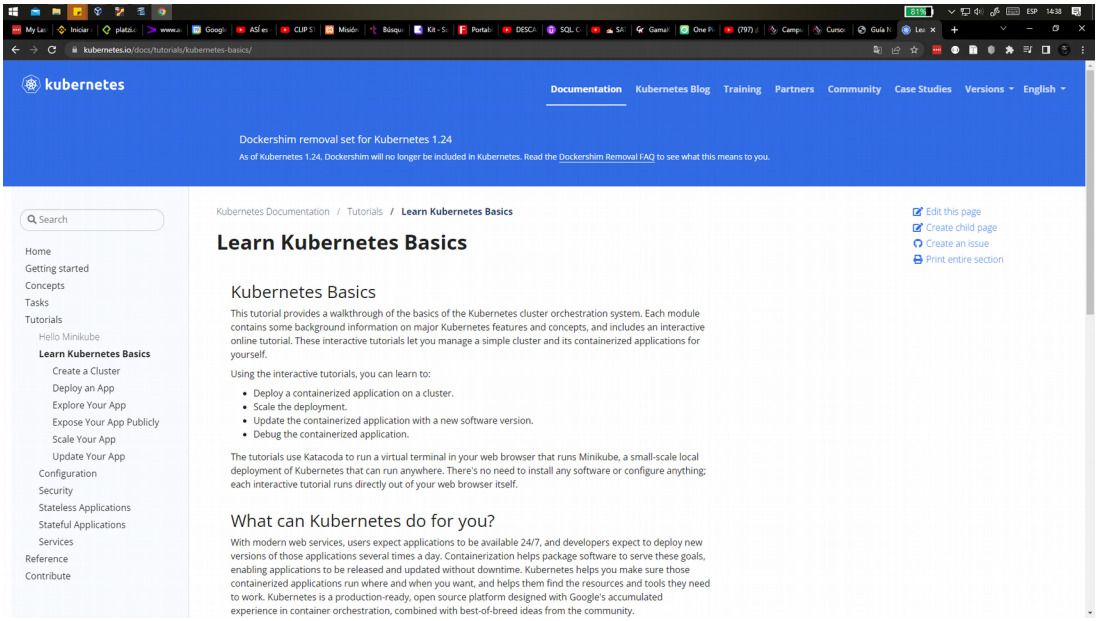
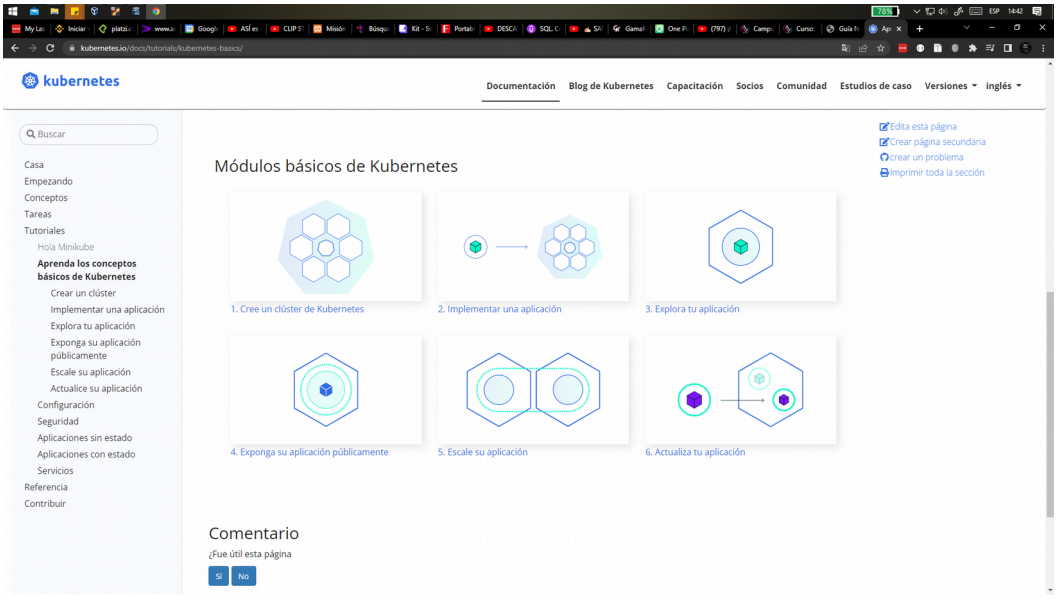
TALLER 8: Kubernetes Guia Didactica.

Luis Felipe Narvaez Gomez. E-mail: luis.narvaez@usantoto.edu.co. Cod: 2312660. Facultad de Ingenieria de Sistemas.

Para desarrollar esta guia procederemos a acceder al siguiente enlace web: https://kubernetes.io/docs/tutorials/kubernetes-basics/ al abrirlo se mostrarala siguiente pagina.



Nos dirigimos a la seccion de “Learn Kubernetes Basics”, en mi caso manejare la pagina de kubernetes en español por lo que esta seccion se denominara “Aprenda los conceptos basicos de Kubernetes”.



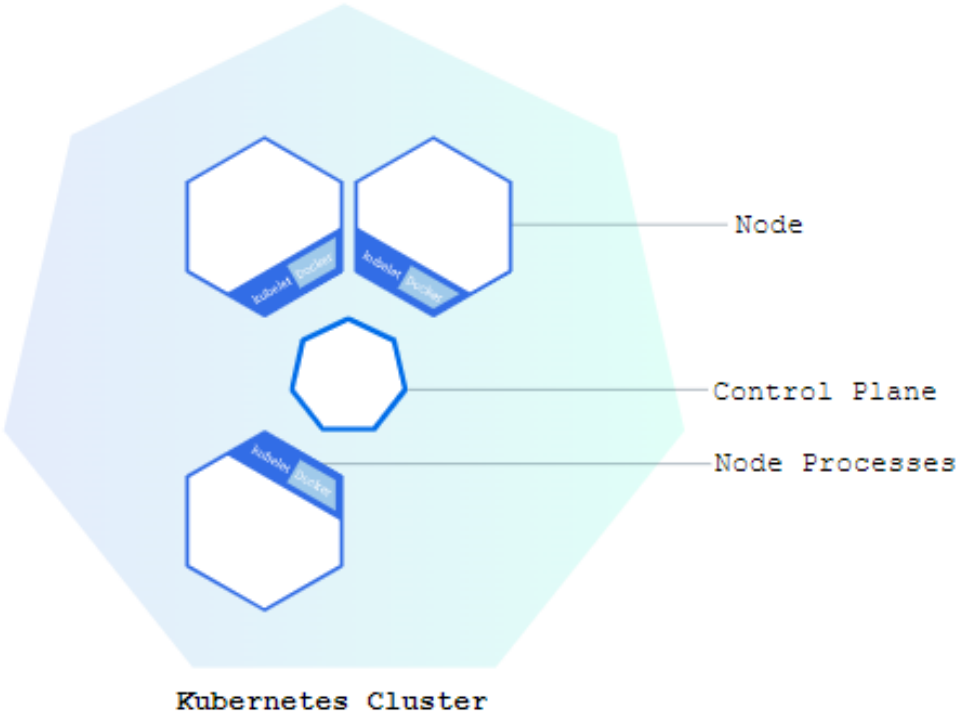
A continuacion realizaremos los diferenntes pasos de los modulos basicos de kubernetes en su respectivo orden:

1. **Uso de Minikube para crear un cluster** → Kubernetes puede coordinar un grupo de

computadoras disponibles para trabajar como una sola unidad, es decir, kubernetes puede coordinar ordenadores de ta manera que se comporten como un solo super ordenado; esto es posible gracias a que kubernetes permite implementar aplicaciones en los contenedores con clusters, esto sin necesidad de vincular el proceso a una maquina en particular.
Para poder utilizar esta cualidad de kubernetes, las aplicaciones deben empaquetarse en contenedores separados con hots individuales. Kubernetes automatiza la distribucion y programacion de contenedores en un cluster de una manera eficiente, para esto un cluster en Kubernetes consta de dos tipos de recursos.

1. El plano de control el cual se encarga de coordinar el cluster, sus actividades, la programacion de aplicaciones, el mantenimiento de estado deseado de las aplicaciones, el escalado de aplicaciones y la implementacion de nuevas actualizaciones.
2. Los nodos, loc uales son los trabajadores que ejecutan las aplicaciones. Los nodos pueden tratarse de VM (Virtual Machines) o computadoras fisicas.. Cada uno de estos nodos contiene un Kubelet el cual es un agente que administra el nodo y su comunicación con el plano de control. Asi mismo, el nodo contienen herramientas que le permitan manejar operaciones de contenedores, como containerd o Docker.

Para la implementacion de Cluster se puee utilizar la herramienta de Minikube la cual crea una VM y un nodo local.



El tutorial Interactivo de la creacion de cluster con Minicube es el siguiente:

Cluster up and running

We already installed minikube for you. Check that it is properly installed, by running the *minikube version* command:

`minikube version ✓`

```
Terminal +
Kubernetes Bootcamp Terminal

$ minikube version
minikube version: v1.18.0
commit: ec61815d60f66a6e4f6353030a40b12362557caa-dirty
$
```

OK, we can see that minikube is in place.

Start the cluster, by running the *minikube start* command:

`minikube start ✓`

```
$ minikube start
* minikube v1.18.0 on Ubuntu 18.04 (amd64)
* Using the none driver based on existing profile

X The requested memory allocation of 2200MiB does not leave room for system overhead
(total system memory: 2460MiB). You may face stability issues.
* Suggestion: Start minikube with less memory allocated: 'minikube start --memory=220
0mb'

* Starting control plane node minikube in cluster minikube
* Running on localhost (CPUs=2, Memory=2460MB, Disk=194868MB) ...
* OS release is Ubuntu 18.04.5 LTS
* Preparing Kubernetes v1.20.2 on Docker 19.03.13 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring local host environment ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace b
y default
$
```

En este momento se tiene ya un cluster en ejecucion de kubernetes, aunque para este momento se trata de un cluster en linea. Para que se creara el cluster Minikube creo una VM en ella es donde se esta ejecutando el cluster.

Cluster version

To interact with Kubernetes during this bootcamp we'll use the command line interface, kubectl. We'll explain kubectl in detail in the next modules, but for now, we're just going to look at some cluster information. To check if kubectl is installed you can run the *kubectl version* command:

`kubectl version ↩`

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.4", GitCommit:"
e87da0bd6e03ec3fea7933c4b5263d151aafd07c", GitTreeState:"clean", BuildDate:"2021-02-1
8T16:12:00Z", GoVersion:"go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.2", GitCommit:"
faecb196815e248d3ecfb03c680a4507229c2a56", GitTreeState:"clean", BuildDate:"2021-01-1
3T13:20:00Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$
```

Con esto el Kubectl esta configurado y podremos verlo tanto en la parte del cliente como en el lado del servidor. La version del cliente es la kubectl mientras que para el lado del servidor es la version dekubernetes instalada en el master. Los detalles de la compilacion pueden observarse en la terminal.

Cluster details

Let's view the cluster details. We'll do that by running *kubectl cluster-info*:

`kubectl cluster-info ↩`

```
$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.12:8443
KubeDNS is running at https://10.0.0.12:8443/api/v1/namespaces/kube-system/services/k
ube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
$
```

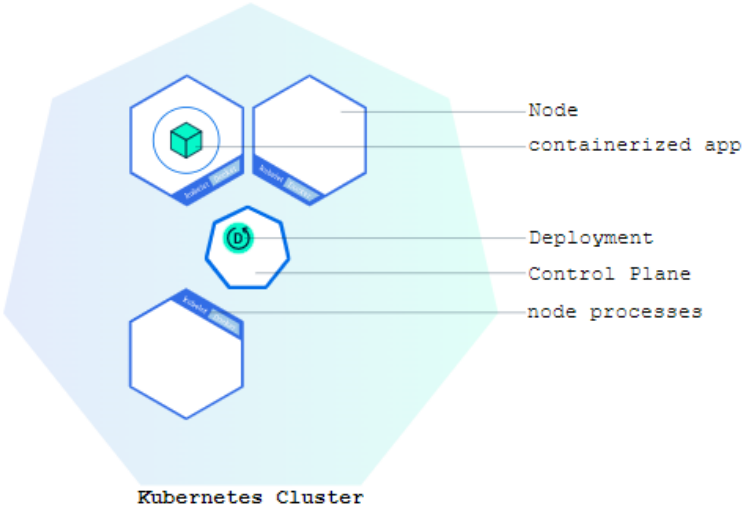
During this tutorial, we'll be focusing on the command line for deploying and exploring our application. To view the nodes in the cluster, run the *kubectl get nodes* command:

`kubectl get nodes ↩`

```
$ kubectl get nodes
NAME       STATUS   ROLES                  AGE     VERSION
minikube   Ready    control-plane,master   8m26s   v1.20.2
$
```

Con el anterior comando se muesta todoslos nodos que se pueden usar para hospedar aplicaciones. En este moento solo tenemos un solo nodo y podemos ver su estado de READY, esto nos dice que esta listo para el alojamento de una aplicación en el.

2. **Uso de Kubectl para crear una implementacion** → Una vez se tiene un cluster de kubernetes en ejecucion, podemos implementar aplicaciones en el. Para poder implementar la aplicación es necesario crear la configuracion de a implementacion la cual indica a Kubernetes como debe crear y actualizar las instancias de la aplicación. Una vez se crea la configuracion de la implementacion, el plano de control de kubernetes programa las instancias de la aplicación, incluidas en esa implementacion para que asi se ejecuten en nodos individuales del cluster, luego un controlador de la implementacion supervisara continuamente las instancias a manera de listas.
   En caso de que una instancia deje de funcionar o se elimine, el controlador lo reemplazara con una semejante de otro nodo del cluster, esto siendo un mecanismo de autorreparacion para abordar fallas o mantenimiento de la maquina.



   La administracion de las implementaciones mediante interfaz de comandos de Kubernetes KUBECTL usa la API de kubernetes para interactuar con el cluster.

NOTA: a partir de este punto del desarrollo del taller se utilizan los propios conceptos y descripciones aprotadas por las diferentes guias y modulos de Kubernetes en su pagina oficial, de la seccion Tutoriales y Guias de Iniciacion (esto solo abarca la seccion del desarrollo con la herramienta didactica).

Como minikube, kubectl viene instalado en la terminal web. Escribe kubectl en la terminal para ver su uso. El formato comun de un comando kubectl es: kubectl accion recurso. Esto ejecuta la accion especificada (como ser crear, describir) sobre el recurso seleccionado (como nodo, contenedor). Tu puedes usar `--help` despues del comando para obtener informacion adicional acerca de posibles parametros (`kubectl get nodes --help`).

Verifica que kubectl esta configurado para comunicarse con tu cluster, ejecutando el siguiente comando: `kubectl version`:

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.4", GitCommit:"e
87da0bd6e03ec3fea7933c4b5263d151aafd07c", GitTreeState:"clean", BuildDate:"2021-02-18T
16:12:00Z", GoVersion:"go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.2", GitCommit:"f
aecb196815e248d3ecfb03c680a4507229c2a56", GitTreeState:"clean", BuildDate:"2021-01-13T
13:20:00Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$
```

OK, kubectl esta instalado y puedes ver las versiones del cliente y servidor.

Para ver los nodos del cluster, ejecuta el comando `kubectl get nodes`.

```
$ kubectl get nodes
NAME       STATUS   ROLES                  AGE    VERSION
minikube   Ready    control-plane,master   3m45s  v1.20.2
$
```

Aqui podemos ver los nodo disponibles (1 en nuestro caso). Kubernetes elegirá donde desplegar nuestra aplicación basado en los nodos disponibles en los recursos.

Ahora desplegaremos nuestra aplicación. Ejecutemos nuestra primera aplicación en Kubernetes con el comando `kubectl run`. El comando `run` crea un nuevo despliegue. Necesitamos proveer el nombre del despliegue y la ubicación de la imagen. (incluyendo la ruta completa para las imagenes alojadas fuera de Docker hub).

```
$ kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernet
es-bootcamp:v1
deployment.apps/kubernetes-bootcamp created
$
```

¡Genial! Acabas de desplegar tu primera aplicación crando un despliegue. Esto ejecuto varias cosas por ti:

- Buscó un nodo apropiado donde la instancia de la aplicación pueda ser desplegada (nosotros tenemos solamente 1 nodo disponible).
- Agendo la ejecución de la aplicación en ese nodo.
- Configuró el cluster para recalenzarizar una instancia en un nuevo nodo cuando sea necesario.

Para listar los despliegues, use el comando `get deployments`:

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           97s
$
```

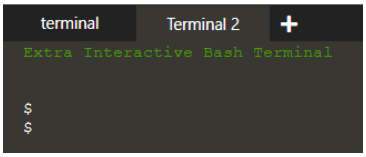
Podemos ver que hay 1 despliegue ejecutando una sola instancia de la aplicación. La instancia se está ejecutando dentro de un container Docker en tu nodo.

Ahora podemos vizualizar nuestra aplicación. Los Pods que están ejecutándose dentro de Kubernetes están en un repositorio privado, con una red aislada. Por defecto solo son visibles desde otros pods y servicios del mismo cluster de kubernetes, pero no fuera de

la red. Cuando usamos kubectl, estamos interactuando a través de un endpoint de la API para comunicarse con nuestra aplicación.
Cubriremos otras opciones de como exponer tu aplicación fuera del cluster de kubernetes en el módulo 4.

El comando kubectl puede crear un proxy que reenviará las comunicaciones de la red privada a todo el cluster. El proxy puede ser cancelado presionando Control-C y no mostrará ninguna salida mientras se esté ejcutando.

Abriremos una segunda ventana de la terminal para correr el proxy



Ahora tenemos una conexión entre nuestro host (la terminal online) y el cluster de kubernetes. El proxy habilita el acceso a la API desde esas terminales.

Podemos ver todas las APIs alojadas a través del endpoint proxy. Por ejemplo, podemos consultar la version directamente desde la API usando el comando curl:

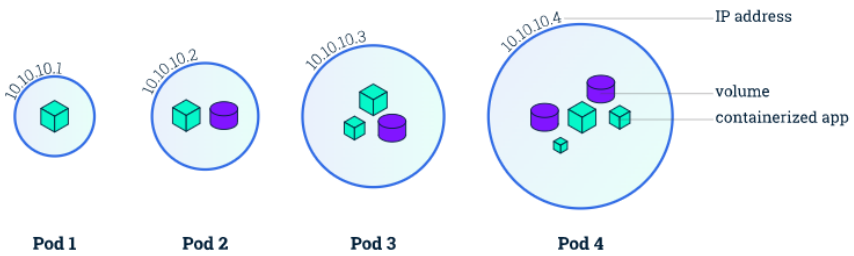
```
$ curl http://localhost:8001/version
curl: (7) Failed to connect to localhost port 8001: Connection refused
```

La API creará automáticamente un endpoint por cada pod, basado en el pod name, que es también accesible a través del proxy.

Primero necesitamos obtener el nombre del pod, y lo almacenaremos en la variable POD_NAME:

```
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-57978f5f5d-wznvd
$
```

3. Visualizacion de PODS y Nodos. En el anterior modulo se creo un POD para alojar la aplicaion. Un POD es una abstraccion de Kubernetes que representa un grupo de uno a varios contenedores de aplicaciones, similar a como funciona Docker, tambien agrupa recursos entre los contenedores, los calues pueden ser: almacenamiento, redes y direcciones IP, informacion de ejecucion de los contenedores, versiones de imagenes de los contenedores y lo spuertos especificos para usar.



Un pod siempre se ejecuta en un **nodo** . Un nodo es una máquina de trabajo en Kubernetes y puede ser una máquina virtual o física, según el clúster. Cada Nodo es administrado por el plano de control. Un nodo puede tener varios pods y el plano de control de Kubernetes gestiona automáticamente la programación de los pods en los nodos del clúster. La programación automática del plano de control tiene en cuenta los recursos disponibles en cada Nodo.

Cada nodo de Kubernetes ejecuta al menos:

- Kubelet, un proceso responsable de la comunicación entre el plano de control de Kubernetes y el Nodo; gestiona los Pods y los contenedores que se ejecutan en una máquina.
- Un tiempo de ejecución de contenedor (como Docker) responsable de extraer la imagen del contenedor de un registro, desempaquetar el contenedor y ejecutar la aplicación.

Ahora ya resolviendo la guia didactica guia 3, primero verificamos que la aplicación que implementamos en el modulo 2 se este ejecutando, para esto utilizamos el siguiente comando:

```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-fb5c67579-p9fln    1/1     Running   0          101s
$
```

Ahora para ver que contenedores hay dentro del POD y que imagenes se usan para contruir esos contenedores, ejecutamos el siguiente comando:

```
$ kubectl describe pods
Name:         kubernetes-bootcamp-fb5c67579-p9fln
Namespace:    default
Priority:     0
Node:         minikube/172.17.0.20
Start Time:   Fri, 22 Apr 2022 22:39:38 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=fb5c67579
Annotations:  <none>
Status:       Running
IP:           172.18.0.2
IPs:
  IP:           172.18.0.2
Controlled By:  ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://71ada0282d12b91a25469338920eaaa50540105c26828f1af05ddd6d3b88b847
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 22 Apr 2022 22:39:40 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-l54lq (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-l54lq:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-l54lq
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age    From               Message
  ----     ------            ----   ----               -------
  Warning  FailedScheduling  3m22s  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal   Scheduled         3m22s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-fb5c67579-p9fln to minikube
  Normal   Pulled            3m20s  kubelet            Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal   Created           3m20s  kubelet            Created container kubernetes-bootcamp
  Normal   Started           3m20s  kubelet            Started container kubernetes-bootcamp
$
```

Vemos aquí detalles sobre el contenedor del Pod: la dirección IP, los puertos utilizados y una lista de eventos relacionados con el ciclo de vida del Pod.

Ahora procederemos a ver la aplicación en la terminal.Recuerde que los pods se ejecutan en una red privada y aislada, por lo que debemos proporcionarles acceso proxy para poder depurarlos e interactuar con ellos. Para hacer esto, usaremos el comando de proxy kubectl para ejecutar un proxy en una segunda ventana de terminal. Haga clic en el siguiente comando para abrir automáticamente una nueva terminal y ejecutar el proxy:

```
Extra Interactive Bash Terminal


$
$ echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not output a response. e first Terminal Tab\n"; kubectl proxy


Starting Proxy. After starting it will not output a response. Please click the first Terminal Tab

Starting to serve on 127.0.0.1:8001
echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not output a response. Please click the first Terminal Tab\n"; kubectl proxy
```

Ahora nuevamente, obtendremos el nombre del Pod y consultaremos ese pod directamente a través del proxy. Para obtener el nombre del Pod y almacenarlo en la variable de entorno POD_NAME:

```
$ export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.me
tadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-p9fln
$
```

Para ver el resultado de nuestra aplicación, ejecute una solicitud curl.

```
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME/proxy/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-p9fln | v=1
$
```

La url es la ruta a la API del Pod. Ahora veremos los Containers logs. Todo lo que la aplicación normalmente enviaría a STDOUT se convierte en registros para el contenedor dentro del Pod. Podemos recuperar estos registros usando el comando kubectl logs:

```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2022-04-22T22:39:40.469Z | Running On:  kubernete
s-bootcamp-fb5c67579-p9fln

Running On: kubernetes-bootcamp-fb5c67579-p9fln | Total Requests: 1 | App Uptime: 731
.651 seconds | Log Time: 2022-04-22T22:51:52.120Z
$
```

Ahora ejecutando comando en el contenedor. Podemos ejecutar comandos directamente en el contenedor una vez que el Pod esté en funcionamiento. Para esto, usamos el comando exec y usamos el nombre del Pod como parámetro. Hagamos una lista de las variables de entorno:

```
$ kubectl exec $POD_NAME -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=kubernetes-bootcamp-fb5c67579-p9fln
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=6.3.1
HOME=/root
$
```

Nuevamente, vale la pena mencionar que el nombre del contenedor en sí se puede omitir ya que solo tenemos un contenedor en el Pod. A continuación, comencemos una sesión de bash en el contenedor del Pod:

```
$ kubectl exec -ti $POD_NAME -- bash
root@kubernetes-bootcamp-fb5c67579-p9fln:/#
```

Ahora tenemos una consola abierta en el contenedor donde ejecutamos nuestra aplicación NodeJS. El código fuente de la aplicación está en el archivo server.js:

```
$ kubectl exec -ti $POD_NAME -- bash
root@kubernetes-bootcamp-fb5c67579-p9fln:/# cat server.js
var http = require('http');
var requests=0;
var podname= process.env.HOSTNAME;
var startTime;
var host;
var handleRequest = function(request, response) {
  response.setHeader('Content-Type', 'text/plain');
  response.writeHead(200);
  response.write("Hello Kubernetes bootcamp! | Running on: ");
  response.write(host);
  response.end(" | v=1\n");
  console.log("Running On:" ,host, "| Total Requests:", ++requests,"| App Uptime:", (
new Date() - startTime)/1000 , "seconds", "| Log Time:",new Date());
}
var www = http.createServer(handleRequest);
www.listen(8080,function () {
    startTime = new Date();;
    host = process.env.HOSTNAME;
    console.log ("Kubernetes Bootcamp App Started At:",startTime, "| Running On: " ,h
ost, "\n" );
});
root@kubernetes-bootcamp-fb5c67579-p9fln:/#
```

Puede verificar que la aplicación esté activa ejecutando un comando curl:

```
root@kubernetes-bootcamp-fb5c67579-p9fln:/# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-p9fln | v=1
root@kubernetes-bootcamp-fb5c67579-p9fln:/#
```

Para cerrar su tipo de conexión de contenedor

```
root@kubernetes-bootcamp-fb5c67579-p9fln:/# exit
exit
$
```

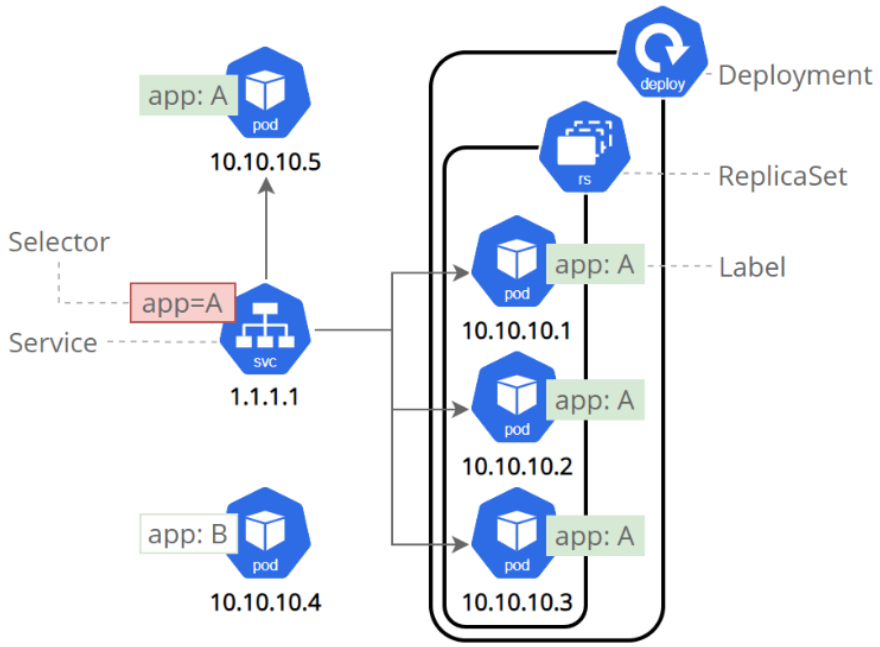
4. **Usar un servicio para exponer su aplicación:** Cabe decir qe los PODS son bolatiles, tienen un ciclo de vida, cuando un nodo muere, los pods que se ejecutan en el nodo tambien se pierden. Sin embargo con un ReplicaSet se puede hacer que el cluster vuelva al estado deseado de forma dinamica mediante la creacion de nuevos nodos PODS para mantener asi el funcionamiento optimo.
Un Servicio en Kubernetes es una abstracción que define un conjunto lógico de Pods y una política para acceder a ellos. Los servicios permiten un acoplamiento flexible entre pods dependientes. Un Servicio se define mediante YAML (preferido) o JSON, como todos los objetos de Kubernetes. El conjunto de Pods a los que se dirige un Servicio generalmente lo determina un LabelSelector (consulte a continuación por qué es posible que desee un Servicio sin incluirlo selectoren la especificación).
Aunque cada Pod tiene una dirección IP única, esas IP no están expuestas fuera del clúster sin un Servicio. Los servicios permiten que sus aplicaciones reciban tráfico. Los servicios se pueden exponer de diferentes maneras especificando un typeen ServiceSpec:

- Cluster IP, el predeterminado, expone el servicio en unaIP interna en el cluster.
- Node Port, expone el servicio en el mismo puerto de cada nodo seleccionado en el cluster mediante NAT.
- Load Balancer, crea un balanceador de carga extremo en la nube actual siempre que sea compatible y asigna una IP externa fija al servicio.
- Ecternal Name, asigna el servicio al contenico del “external Name”, devolvieno un CNAME de registro con su valor.

Un Servicio enruta el tráfico a través de un conjunto de Pods. Los servicios son la abstracción que permite que los pods mueran y se repliquen en Kubernetes sin afectar su aplicación. El descubrimiento y el enrutamiento entre pods dependientes (como los componentes frontend y backend en una aplicación) están a cargo de Kubernetes Services.
Los servicios hacen coincidir un conjunto de pods mediante etiquetas y selectores , una primitiva de agrupación que permite la operación lógica en objetos en Kubernetes. Las etiquetas son pares clave/valor adjuntos a los objetos y se pueden usar de varias maneras:

- Designar objetos para desarrollo, prueba y produccion.
- Etiquetas de version incrustadas.
- Clasificar un objeto usando etiquetas.

Las etiquetas se pueden adjuntar a los objetos en el momento de la creación o más adelante. Se pueden modificar en cualquier momento. Expongamos nuestra aplicación ahora usando un Servicio y apliquemos algunas etiquetas.



Ahora si, creemos un nuevo servicio. Verifiquemos que nuestra aplicación se está ejecutando. Usaremos el comando kubectl get y buscaremos Pods existentes:

```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-fb5c67579-5t87p    1/1     Running   0          18s
$
```

A continuación, enumeremos los servicios actuales de nuestro clúster:

```
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   82s
$
```

Tenemos un servicio llamado kubernetes que se crea de forma predeterminada cuando minikube inicia el clúster. Para crear un nuevo servicio y exponerlo al tráfico externo, usaremos el comando de exposición con NodePort como parámetro (minikube aún no admite la opción LoadBalancer).

```
$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
service/kubernetes-bootcamp exposed
$
```

Ejecutemos de nuevo el comando get services:

```
$ kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          40m
kubernetes-bootcamp   NodePort    10.107.61.186   <none>        8080:31880/TCP   73s
$
```

Ahora tenemos un servicio en ejecución llamado kubernetes-bootcamp. Aquí vemos que el Servicio recibió una IP de clúster única, un puerto interno y una IP externa (la IP del Nodo).

Para averiguar qué puerto se abrió externamente (mediante la opción NodePort), ejecutaremos el comando describe service:

```
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   app=kubernetes-bootcamp
Annotations:              <none>
Selector:                 app=kubernetes-bootcamp
Type:                     NodePort
IP Families:              <none>
IP:                       10.107.61.186
IPs:                      10.107.61.186
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31880/TCP
Endpoints:                172.18.0.3:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$
```

Cree una variable de entorno llamada NODE_PORT que tenga asignado el valor del puerto del Nodo:

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index
  .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=31880
$
```

Ahora podemos probar que la aplicación está expuesta fuera del clúster usando curl, la IP del nodo y el puerto expuesto externamente:

```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-5t87p | v=1
$
```

Y recibimos una respuesta del servidor. El Servicio está expuesto. Ahora usaremos etiquetas. El Deployment creó automáticamente una etiqueta para nuestro Pod. Con el comando describe deployment, puede ver el nombre de la etiqueta:

```
$ kubectl describe deployment
Name:                   kubernetes-bootcamp
Namespace:              default
CreationTimestamp:      Fri, 22 Apr 2022 23:22:34 +0000
Labels:                 app=kubernetes-bootcamp
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=kubernetes-bootcamp
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=kubernetes-bootcamp
  Containers:
   kubernetes-bootcamp:
    Image:        gcr.io/google-samples/kubernetes-bootcamp:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kubernetes-bootcamp-fb5c67579 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  44m   deployment-controller  Scaled up replica set kubernetes-bootcamp-fb5c67579 to 1
$
```

Usemos esta etiqueta para consultar nuestra lista de Pods. Usaremos el comando kubectl get pods con -l como parámetro, seguido de los valores de la etiqueta:

```
$ kubectl get pods -l app=kubernetes-bootcamp
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-fb5c67579-5t87p   1/1     Running   0          46m
$
```

Puedes hacer lo mismo para listar los servicios existentes:

```
$ kubectl get services -l app=kubernetes-bootcamp
NAME                  TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes-bootcamp   NodePort   10.107.61.186   <none>        8080:31880/TCP   8m58s
$
```

Obtén el nombre del Pod y guárdalo en la variable de entorno POD_NAME:

```
$ export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.meta
data.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-5t87p
$
```

Para aplicar una nueva etiqueta, usamos el comando de etiqueta seguido del tipo de objeto, el nombre del objeto y la nueva etiqueta:

```
$ kubectl label pods $POD_NAME version=v1
pod/kubernetes-bootcamp-fb5c67579-5t87p labeled
$
```

Esto aplicará una nueva etiqueta a nuestro Pod (fijamos la versión de la aplicación al Pod), y podemos verificarlo con el comando describe pod:

```
$ kubectl describe pods $POD_NAME
Name:         kubernetes-bootcamp-fb5c67579-5t87p
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.8
Start Time:   Fri, 22 Apr 2022 23:22:51 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=fb5c67579
              version=v1
Annotations:  <none>
Status:       Running
IP:           172.18.0.3
IPs:
  IP:           172.18.0.3
Controlled By:  ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://8948ee5055c9ab701ea066a2826e791f6ce5a6f1b3c00e0a5d53a9fd03bd656f
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 22 Apr 2022 23:22:53 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-cxsc9 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-cxsc9:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-cxsc9
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  50m (x2 over 50m)  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal   Scheduled         50m                default-scheduler  Successfully assigned default/kubernetes-bootcamp-fb5c67579-5t87p to minikube
  Normal   Pulled            50m                kubelet            Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal   Created           50m                kubelet            Created container kubernetes-bootcamp
  Normal   Started           50m                kubelet            Started container kubernetes-bootcamp
$
```

Vemos aquí que la etiqueta está adherida ahora a nuestro Pod. Y podemos consultar ahora la lista de pods usando la nueva etiqueta:

```
$ kubectl get pods -l version=v1
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-fb5c67579-5t87p   1/1     Running   0          52m
$
```

Ahora eliminaremos, destruiremos , incinararemos , despedazaremos y explotaremos en mil partes un servicio. Para eliminar servicios, puede utilizar el comando de eliminación de servicios. Las etiquetas también se pueden utilizar aquí:

```
$ kubectl delete service -l app=kubernetes-bootcamp
service "kubernetes-bootcamp" deleted
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   55m
$
```

Esto confirma que nuestro Servicio fue eliminado. Para confirmar que la ruta ya no está expuesta, puede curvar la IP y el puerto previamente expuestos:

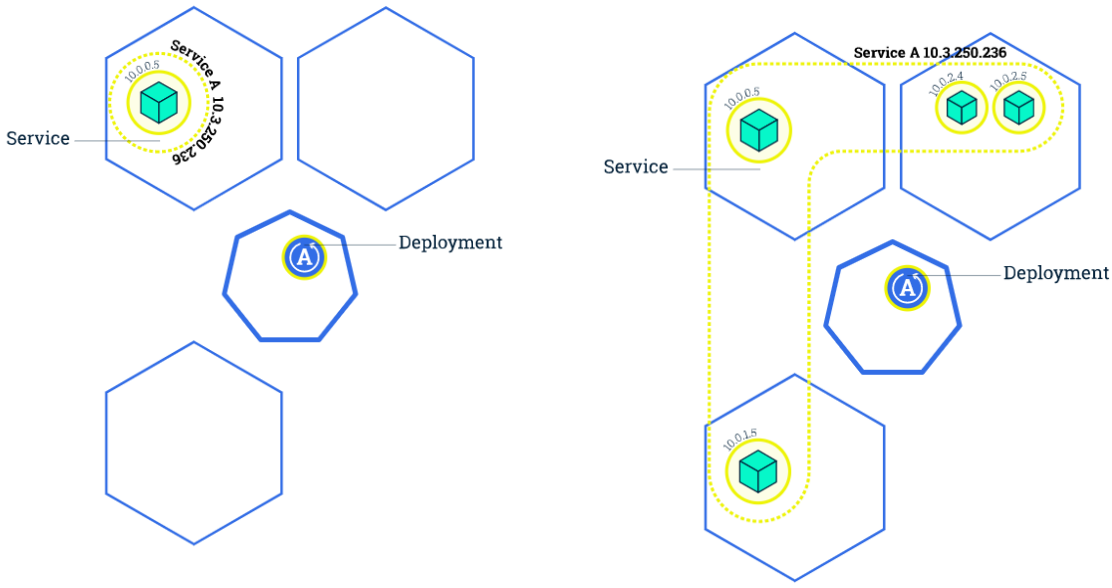
```
$ curl $(minikube ip):$NODE_PORT
curl: (7) Failed to connect to 10.0.0.8 port 31880: Connection refused
$
```

Esto prueba que ya no se puede acceder a la aplicación desde fuera del clúster. Puede confirmar que la aplicación todavía se está ejecutando con un rizo dentro del pod:

```
$ kubectl exec -ti $POD_NAME -- curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-5t87p | v=1
$
```

Vemos aquí que la aplicación está activa. Esto se debe a que la implementación está administrando la aplicación. Para cerrar la aplicación, también deberá eliminar la implementación.

5. **Ejecutar multiples instancias de una aplicación:** En los modulos anteriores creamos un solo POD pero cuando aumente e trafico de nuestra aplicación necesitaremos escalar la misma para satisfacer la demanda de los usuarios. El escalado se logra cambiando el numero de replicas en una implementacion.
La escalizacion horizontal garantiza que se creen nuevos PODS y que estos se programen para los nodos con los recursos disponibles. Tambien exite el escalado automatico de PODs pero en esta guia solo veremos la primera forma de escalado.
El escalado se logra cambiando la cantidad de replicas en una implementacion. La ejecución de varias instancias de una aplicación requerirá una forma de distribuir el tráfico a todas ellas. Los servicios tienen un equilibrador de carga integrado que distribuirá el tráfico de red a todos los pods de una implementación expuesta. Los servicios monitorearán continuamente los Pods en ejecución usando puntos finales, para garantizar que el tráfico se envíe solo a los Pods disponibles.
El escalado se logra cambiando la cantidad de réplicas en una implementación. Una vez que tenga varias instancias de una aplicación en ejecución, podrá realizar actualizaciones continuas sin tiempo de inactividad. Lo cubriremos en el próximo módulo. Ahora, vayamos a la terminal en línea y escalemos nuestra aplicación.



Listemos los diferentes desarrollos:

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   0/1     0            0           1s
$
```

Deberíamos tener 1 Pod. Si no es así, ejecute el comando de nuevo. Esta espectáculos:

- NAME enumera los nombres de las implementaciones en el clúster.
- READY muestra la proporción de réplicas CURRENT/DESIRED
- UP-TO-DATE muestra el número de réplicas que se han actualizado para alcanzar el estado deseado.
- AVAILABE muestra cuántas réplicas de la aplicación están disponibles para sus usuarios.
- AGE muestra la cantidad de tiempo que la aplicación se ha estado ejecutando.

Para ver el ReplicaSet creado por la implementación, ejecute

```
$ kubectl get rs
NAME                           DESIRED   CURRENT   READY   AGE
kubernetes-bootcamp-fb5c67579   1         1         1       2m14s
$
```

Tenga en cuenta que el nombre del ReplicaSet siempre tiene el formato [DEPLOYMENT-NAME]-[RANDOM-STRING] La cadena aleatoria se genera aleatoriamente y utiliza pod-template-hash como semilla.
Dos columnas importantes de este comando son:

- DESIRED muestra el número deseado de réplicas de la aplicación, que usted define cuando crea el Deployment. Este es el estado deseado.
- CURRENT muestra cuántas réplicas se están ejecutando actualmente.

A continuación, escalamos la implementación a 4 réplicas. Usaremos el comando kubectl scale, seguido del tipo de implementación, el nombre y la cantidad deseada de instancias:

```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           15m
$
```

Se aplicó el cambio y tenemos 4 instancias de la aplicación disponibles. A continuación, verifiquemos si la cantidad de Pods cambió:

```
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-fb5c67579-6nnt7   1/1     Running   0          16m   172.18.0.3   minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-b9qfs   1/1     Running   0          61s   172.18.0.9   minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-js4px   1/1     Running   0          61s   172.18.0.8   minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-r42g4   1/1     Running   0          61s   172.18.0.7   minikube   <none>           <none>
$
```

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           17m
$
```

Hay 4 Pods ahora, con diferentes direcciones IP. El cambio se registró en el registro de eventos de implementación. Para verificar eso, use el comando describe:

```
$ kubectl describe deployments/kubernetes-bootcamp
Name:                   kubernetes-bootcamp
Namespace:              default
CreationTimestamp:      Sat, 23 Apr 2022 00:40:21 +0000
Labels:                 app=kubernetes-bootcamp
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=kubernetes-bootcamp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=kubernetes-bootcamp
  Containers:
   kubernetes-bootcamp:
    Image:        gcr.io/google-samples/kubernetes-bootcamp:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kubernetes-bootcamp-fb5c67579 (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  18m   deployment-controller  Scaled up replica set kubernetes-bootcamp-fb5c67579 to 1
  Normal  ScalingReplicaSet  3m1s  deployment-controller  Scaled up replica set kubernetes-bootcamp-fb5c67579 to 4
$
```

También puede ver en el resultado de este comando que ahora hay 4 réplicas. Verifiquemos que el Servicio equilibre la carga del tráfico. Para averiguar la IP y el puerto expuestos, podemos usar el servicio de descripción como aprendimos en el módulo anterior:

```
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   app=kubernetes-bootcamp
Annotations:              <none>
Selector:                 app=kubernetes-bootcamp
Type:                     NodePort
IP Families:              <none>
IP:                       10.109.228.217
IPs:                      10.109.228.217
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  32649/TCP
Endpoints:                172.18.0.3:8080,172.18.0.7:8080,172.18.0.8:8080 + 1 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$
```

Cree una variable de entorno llamada NODE_PORT que tenga un valor como el puerto del nodo:

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(
index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=32649
$
```

A continuación, haremos un curl con la IP y el puerto expuestos. Ejecute el comando varias veces:

```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c67579-b9qfs | v=1
$
```

Llegamos a un Pod diferente con cada solicitud. Esto demuestra que el equilibrio de carga está funcionando. Para reducir el Servicio a 2 réplicas, ejecute nuevamente el comando de escala:

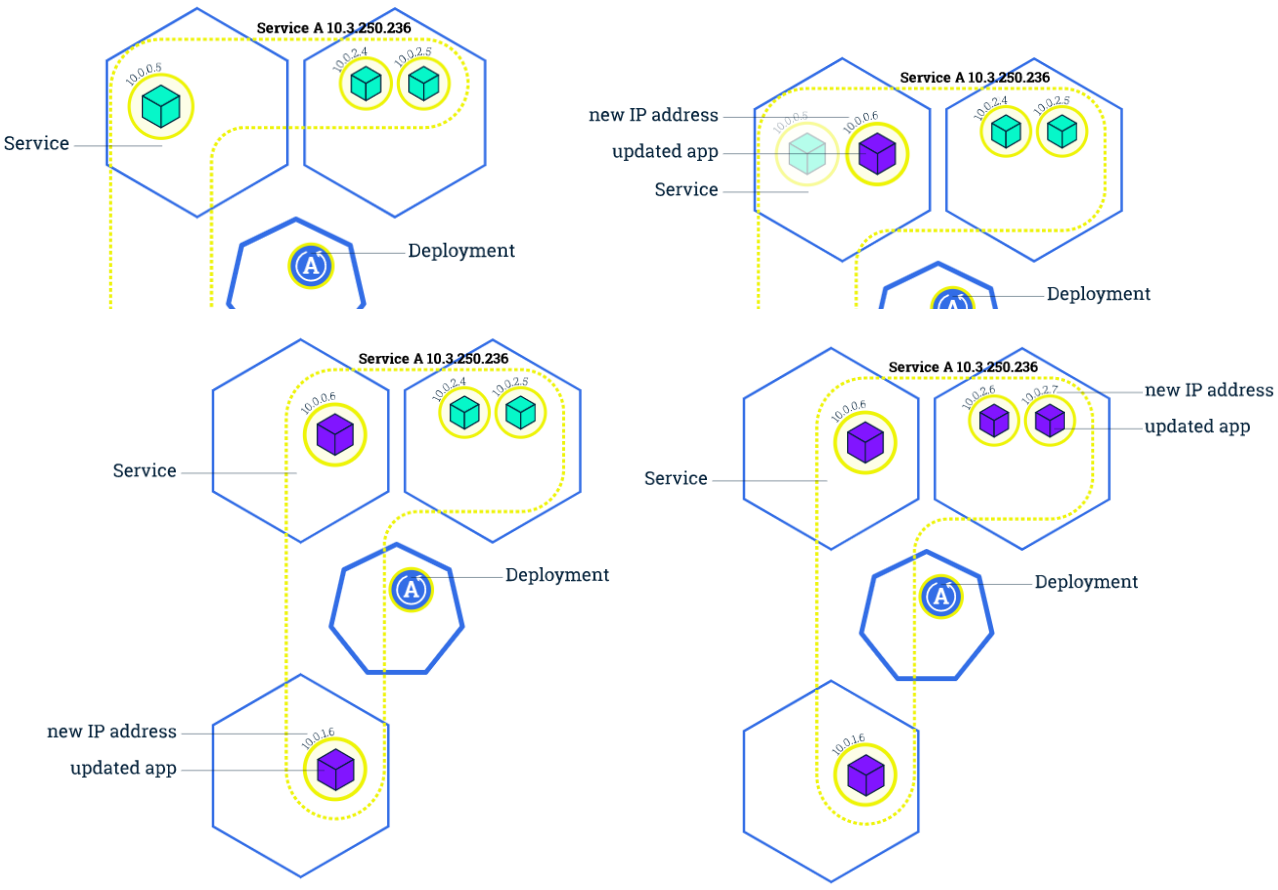
```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.apps/kubernetes-bootcamp scaled
$
```

Enumere las implementaciones para verificar si el cambio se aplicó con el comando get deployments:

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   2/2     2            2           25m
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE     IP            NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-fb5c67579-6nnt7   1/1     Running   0          24m     172.18.0.3    minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-js4px   1/1     Running   0          9m37s   172.18.0.8    minikube   <none>           <none>
$
```

Esto confirma que se cancelaron 2 pods.

6. Realizacion de una actualizacion continua: Los usuarios esperan que las aplicaciones estén disponibles todo el tiempo y se espera que los desarrolladores implementen nuevas versiones de ellas varias veces al día. En Kubernetes esto se hace con actualizaciones continuas. Las actualizaciones continuas permiten que la actualización de las implementaciones se realice sin tiempo de inactividad mediante la actualización incremental de las instancias de pods con otras nuevas. Los nuevos Pods se programarán en Nodos con recursos disponibles.
En el módulo anterior escalamos nuestra aplicación para ejecutar varias instancias. Este es un requisito para realizar actualizaciones sin afectar la disponibilidad de la aplicación. De forma predeterminada, la cantidad máxima de Pods que pueden no estar disponibles durante la actualización y la cantidad máxima de Pods nuevos que se pueden crear es uno. Ambas opciones se pueden configurar en números o porcentajes (de Pods). En Kubernetes, las actualizaciones están versionadas y cualquier actualización de implementación se puede revertir a una versión anterior (estable).



De manera similar al escalado de aplicaciones, si una implementación se expone públicamente, el servicio equilibrará la carga del tráfico solo a los pods disponibles durante la actualización. Un Pod disponible es una instancia que está disponible para los usuarios de la aplicación.
Las actualizaciones continuas permiten las siguientes acciones:

- Promocionar una aplicación de un entorno a otro (a través de actualizaciones de imágenes de contenedores)
- Retroceder a versiones anteriores
- Integración continua y entrega continua de aplicaciones sin tiempo de inactividad

En el siguiente tutorial interactivo, actualizaremos nuestra aplicación a una nueva versión y también realizaremos una reversión. Si una Implementación se expone públicamente, el Servicio equilibrará la carga del tráfico solo a los Pods disponibles durante la actualización.

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           37s
$
```

Para enumerar los pods en ejecución, ejecute el comando get pods:

```
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-fb5c67579-2ml8k   1/1     Running   0          38s
kubernetes-bootcamp-fb5c67579-6pjgd   1/1     Running   0          38s
kubernetes-bootcamp-fb5c67579-c8j5s   1/1     Running   0          38s
kubernetes-bootcamp-fb5c67579-w7gd9   1/1     Running   0          38s
$
```

Para ver la versión de imagen actual de la aplicación, ejecute el comando describe pods y busque el campo Imagen:

```
$ kubectl describe pods
Name:         kubernetes-bootcamp-fb5c67579-2ml8k
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:47:28 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=fb5c67579
Annotations:  <none>
Status:       Running
IP:           172.18.0.2
IPs:
  IP:           172.18.0.2
Controlled By:  ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://00cbba7bb1696528a41fa7df3167ad9dd609f69059a80b8ff574cfc9620dfe94
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:47:30 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                 From               Message
  ----     ------            ----                ----               -------
  Warning  FailedScheduling  110s (x2 over 110s)  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal   Scheduled         107s                default-scheduler  Successfully assigned default/kubernetes-bootcamp-fb5c67579-2ml8k to minikube
  Normal   Pulled            106s                kubelet            Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal   Created           106s                kubelet            Created container kubernetes-bootcamp
  Normal   Started           105s                kubelet            Started container kubernetes-bootcamp


Name:         kubernetes-bootcamp-fb5c67579-6pjgd
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:47:28 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=fb5c67579
Annotations:  <none>
```

```
Status:       Running
IP:           172.18.0.9
IPs:
  IP:           172.18.0.9
Controlled By:  ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://8f4f0be5287e756baec730abe169ea8637a05d5bef25b7968295af6ddd837a0
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:47:31 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                 From               Message
  ----     ------            ----                ----               -------
  Warning  FailedScheduling  110s (x2 over 110s)  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal   Scheduled         107s                default-scheduler  Successfully assigned default/kubernetes-bootcamp-fb5c67579-6pjgd to minikube
  Normal   Pulled            104s                kubelet            Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal   Created           104s                kubelet            Created container kubernetes-bootcamp
  Normal   Started           104s                kubelet            Started container kubernetes-bootcamp


Name:         kubernetes-bootcamp-fb5c67579-c8j5s
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:47:28 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=fb5c67579
Annotations:  <none>
Status:       Running
IP:           172.18.0.3
IPs:
  IP:           172.18.0.3
Controlled By:  ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://be541ad7df459c74f8456fa3029b3c9b25dc225369c55f8848eb197014051733
```

```
  kubernetes-bootcamp:
    Container ID:   docker://be541ad7df459c74f8456fa3029b3c9b25dc225369c55f8848eb197014051733
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:47:30 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                 From               Message
  ----     ------            ----                ----               -------
  Warning  FailedScheduling  110s (x2 over 110s)  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal   Scheduled         107s                default-scheduler  Successfully assigned default/kubernetes-bootcamp-fb5c67579-c8j5s to minikube
  Normal   Pulled            106s                kubelet            Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal   Created           106s                kubelet            Created container kubernetes-bootcamp
  Normal   Started           105s                kubelet            Started container kubernetes-bootcamp


Name:         kubernetes-bootcamp-fb5c67579-w7gd9
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:47:28 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=fb5c67579
Annotations:  <none>
Status:       Running
IP:           172.18.0.8
IPs:
  IP:           172.18.0.8
Controlled By:  ReplicaSet/kubernetes-bootcamp-fb5c67579
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://4d51ac50d63724243a8928ed0811a30356ba82babbcdfb896a029214cf12e75b
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:47:30 +0000
    Ready:          True
```

```
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age                From              Message
  ----     ------            ----               ----              -------
  Warning  FailedScheduling  110s (x2 over 110s)  default-scheduler  0/1 nodes are available: 1 node(s) had taint {node.kubernetes.io/not-ready: }, that the pod didn't tolerate.
  Normal   Scheduled         107s               default-scheduler  Successfully assigned default/kubernetes-bootcamp-fb5c67579-w7gd9 to minikube
  Normal   Pulled            105s               kubelet            Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present on machine
  Normal   Created           105s               kubelet            Created container kubernetes-bootcamp
  Normal   Started           105s               kubelet            Started container kubernetes-bootcamp
$
```

Para actualizar la imagen de la aplicación a la versión 2, use el comando establecer imagen, seguido del nombre de la implementación y la nueva versión de la imagen:

```
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kub
ernetes-bootcamp:v2
deployment.apps/kubernetes-bootcamp image updated
$
```

El comando notificó a la implementación que usara una imagen diferente para su aplicación e inició una actualización continua. Verifique el estado de los nuevos Pods y vea el anterior que termina con el comando get pods:

```
$ kubectl get pods
NAME                                   READY   STATUS        RESTARTS   AGE
kubernetes-bootcamp-7d44784b7c-8l279   1/1     Running       0          89s
kubernetes-bootcamp-7d44784b7c-bfhf4   1/1     Running       0          86s
kubernetes-bootcamp-7d44784b7c-kx8kp   1/1     Running       0          86s
kubernetes-bootcamp-7d44784b7c-nsnnz   1/1     Running       0          89s
kubernetes-bootcamp-fb5c67579-2ml8k    0/1     Terminating   0          9m55s
$
```

Primero, verifique que la aplicación se esté ejecutando. Para encontrar la IP y el puerto expuestos, ejecute el comando describe service:

```
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   app=kubernetes-bootcamp
Annotations:              <none>
Selector:                 app=kubernetes-bootcamp
Type:                     NodePort
IP Families:              <none>
IP:                       10.101.97.11
IPs:                      10.101.97.11
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30020/TCP
Endpoints:                172.18.0.10:8080,172.18.0.11:8080,172.18.0.12:8080 + 1 more
...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$
```

Cree una variable de entorno llamada NODE_PORT que tenga asignado el valor del puerto del Nodo:

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(inde
x .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=30020
$
```

A continuación, haz un curl con la IP y el puerto expuestos:

```
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-7d44784b7c-kx8kp | v=2
$
```

Cada vez que ejecute el comando curl, accederá a un Pod diferente. Tenga en cuenta que todos los pods ejecutan la última versión (v2). También puede confirmar la actualización ejecutando el comando de estado de implementación:

```
$ kubectl rollout status deployments/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
$
```

```
$ kubectl describe pods
Name:         kubernetes-bootcamp-7d44784b7c-81279
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:55:51 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=7d44784b7c
Annotations:  <none>
Status:       Running
IP:           172.18.0.10
IPs:
  IP:           172.18.0.10
Controlled By:  ReplicaSet/kubernetes-bootcamp-7d44784b7c
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://f443a23dc8c57ddd682ceacf053042e23496b602e434fa22add850f92f3823b4
    Image:          jocatalin/kubernetes-bootcamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:fb1a3ced00cecfc1f83f18ab5cd14199e30adc1b49aa4244f5d65ad3f5feb2a5
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:55:52 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  8m8s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-7d44784b7c-81279 to minikube
  Normal  Pulled     8m7s  kubelet            Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
  Normal  Created    8m7s  kubelet            Created container kubernetes-bootcamp
  Normal  Started    8m7s  kubelet            Started container kubernetes-bootcamp


Name:         kubernetes-bootcamp-7d44784b7c-bfhf4
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:55:54 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=7d44784b7c
```

```
Annotations:  <none>
Status:       Running
IP:           172.18.0.12
IPs:
  IP:           172.18.0.12
Controlled By:  ReplicaSet/kubernetes-bootcamp-7d44784b7c
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://013b29e8cad0c3208cd00c3216aa418cc89920f1959cfcfd18645ad432a768cd
    Image:          jocatalin/kubernetes-bootcamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:fb1a3ced00cecfc1f83f18ab5cd14199e30adc1b49aa4244f5d65ad3f5feb2a5
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:55:55 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  8m5s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-7d44784b7c-bfhf4 to minikube
  Normal  Pulled     8m4s  kubelet            Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
  Normal  Created    8m4s  kubelet            Created container kubernetes-bootcamp
  Normal  Started    8m4s  kubelet            Started container kubernetes-bootcamp


Name:         kubernetes-bootcamp-7d44784b7c-kx8kp
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:55:54 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=7d44784b7c
Annotations:  <none>
Status:       Running
IP:           172.18.0.13
IPs:
  IP:           172.18.0.13
Controlled By:  ReplicaSet/kubernetes-bootcamp-7d44784b7c
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://74ebcbbaf6ba3b48134e8abd442a80459670601455b9154f7660f575e4c4e0a2
    Image:          jocatalin/kubernetes-bootcamp:v2
```

```
  kubernetes-bootcamp:
    Container ID:   docker://74ebcbbaf6ba3b48134e8abd442a80459670601455b9154f7660f575e4c4e0a2
    Image:          jocatalin/kubernetes-bootcamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:fb1a3ced00cecfc1f83f18ab5cd14199e30adc1b49aa4244f5d65ad3f5feb2a5
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:55:55 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  8m5s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-7d44784b7c-kx8kp to minikube
  Normal  Pulled     8m4s  kubelet            Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
  Normal  Created    8m4s  kubelet            Created container kubernetes-bootcamp
  Normal  Started    8m4s  kubelet            Started container kubernetes-bootcamp


Name:         kubernetes-bootcamp-7d44784b7c-nsnnz
Namespace:    default
Priority:     0
Node:         minikube/10.0.0.10
Start Time:   Sat, 23 Apr 2022 01:55:51 +0000
Labels:       app=kubernetes-bootcamp
              pod-template-hash=7d44784b7c
Annotations:  <none>
Status:       Running
IP:           172.18.0.11
IPs:
  IP:           172.18.0.11
Controlled By:  ReplicaSet/kubernetes-bootcamp-7d44784b7c
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://30a4b064f4883d62b8cf9ed30455ef7f35816f21fbf18e21d9191669a0aadf2c
    Image:          jocatalin/kubernetes-bootcamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:fb1a3ced00cecfc1f83f18ab5cd14199e30adc1b49aa4244f5d65ad3f5feb2a5
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 23 Apr 2022 01:55:53 +0000
    Ready:          True
    Restart Count:  0
```

```
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-m5hzj (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-m5hzj:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-m5hzj
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  8m8s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-7d44784b7c-nsnnz to minikube
  Normal  Pulled     8m7s  kubelet            Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
  Normal  Created    8m7s  kubelet            Created container kubernetes-bootcamp
  Normal  Started    8m6s  kubelet            Started container kubernetes-bootcamp
$
```

En el campo Imagen de la salida, verifique que esté ejecutando la última versión de la imagen (v2). Revertir una actualización. Realicemos otra actualización e implementemos una imagen etiquetada con v10:

```
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=gcr.io/google-s
amples/kubernetes-bootcamp:v10
deployment.apps/kubernetes-bootcamp image updated
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   3/4     2            3           21m
$
```

```
$ kubectl get pods
NAME                                  READY   STATUS         RESTARTS   AGE
kubernetes-bootcamp-59b7598c77-8gnv8  0/1     ErrImagePull   0          62s
kubernetes-bootcamp-59b7598c77-zcl5s  0/1     ErrImagePull   0          62s
kubernetes-bootcamp-7d44784b7c-81279  1/1     Running        0          13m
kubernetes-bootcamp-7d44784b7c-bfhf4  0/1     Terminating    0          13m
kubernetes-bootcamp-7d44784b7c-kx8kp  1/1     Running        0          13m
kubernetes-bootcamp-7d44784b7c-nsnnz  1/1     Running        0          13m
$
```

```
$ kubectl rollout undo deployments/kubernetes-bootcamp
deployment.apps/kubernetes-bootcamp rolled back
$
```