



Centro Universitario de Ciencias  
Exactas e Ingenierías.



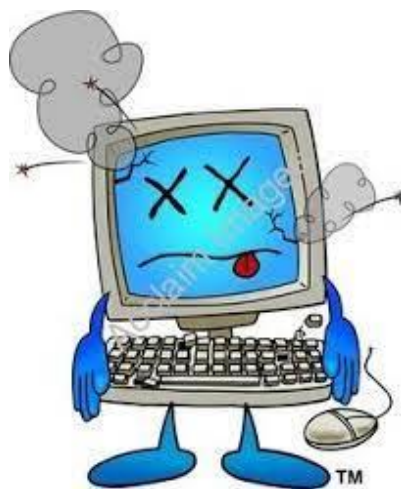
Ingeniería en computación.

Alumno: Vargas López David Guadalupe.

Computación tolerante a fallos.

Profesor: López Franco Michel Emanuel.

Sección: D06.



Guadalajara Jal. abril del 2022.

# Kubernetes:

¿Qué es Kubernetes?



## kubernetes

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa.

### Funciones de kubernetes:

- Organizar los contenedores en varios hosts
- Hacer un mejor uso del hardware para aprovechar al máximo los recursos necesarios en la ejecución de las aplicaciones empresariales
- Controlar y automatizar las implementaciones y actualizaciones de las aplicaciones
- Agregar almacenamiento para ejecutar aplicaciones con estado
- Ampliar las aplicaciones en contenedores y sus recursos según sea necesario
- Gestionar los servicios de forma declarativa para garantizar que las aplicaciones implementadas siempre se ejecuten correctamente
- Realizar comprobaciones de estado y autor regeneraciones de sus aplicaciones con ubicación, reinicio, replicación y adaptación automáticos

Kubernetes se combina con otros proyectos opensource para proporcionar todos estos servicios coordinados, lo cual le permite aprovechar al máximo el potencial de la plataforma.

Kubernetes ayuda a los desarrolladores a escribir aplicaciones que se ejecutan en un clúster, ya que se utiliza para la orquestación de sistemas de tiempo de ejecución de contenedores en un clúster de recursos de hardware en red.

## ¿Qué es Ingress?



Este recurso nos permite acceder a servicios a través de HTTP(S) y el tráfico se controla utilizando un conjunto de reglas que tú defines. Además de dar a tus aplicaciones una URL externa que permita el acceso, también se puede configurar para el balanceo de carga o terminación SSL.

El balanceo de cargas de HTTP(S) configurado por Ingress incluye las características siguientes:

- Configuración flexible para Services Un Ingress define cómo llega el tráfico a tus Services y la forma en que se enruta a tu aplicación. Además, un Ingress puede proporcionar una sola dirección IP para varios Services en tu clúster.
- Integración en los servicios de red de Google Cloud
- Compatibilidad con múltiples certificados TLS Un Ingress puede especificar el uso de varios certificados TLS para la finalización de solicitudes.

Ingress consta de tres componentes:

- Recursos de Ingress
- Equilibradores de carga de aplicación (ALB)
- Un equilibrador de carga para manejar las solicitudes entrantes entre distintas zonas. En los clústeres clásicos, este componente es el equilibrador de carga multizona (MZLB) que IBM Cloud Kubernetes Service crea automáticamente. En los clústeres de VPC, este componente es el equilibrador de carga de VPC que se crea automáticamente en la VPC.

## ¿Qué es un LoadBalancer?



Distribuye automáticamente el tráfico entrante entre varios destinos, por ejemplo, instancias EC2, contenedores y direcciones IP en una o varias zonas de disponibilidad. Monitorea el estado de los destinos registrados y enruta el tráfico solamente a destinos en buen estado. Elastic Load Balancing escala el balanceador de carga a medida que el tráfico entrante va cambiando con el tiempo. Puede escalarse automáticamente para adaptarse a la mayoría de las cargas de trabajo.

Un *balanceador de carga* actúa como único punto de contacto para los clientes. El balanceador de carga distribuye el tráfico entrante de aplicaciones entre varios destinos, tales como instancias EC2, en varias zonas de disponibilidad. Esto aumenta la disponibilidad de la aplicación. Puede agregar uno o varios agentes de escucha al balanceador de carga.

## ¿Qué es Rancher?



Rancher es un software para administrar clusters de Kubernetes, eso incluye no solo la gestión de clusters existentes, sino que también la posibilidad de crear nuevos clústeres.

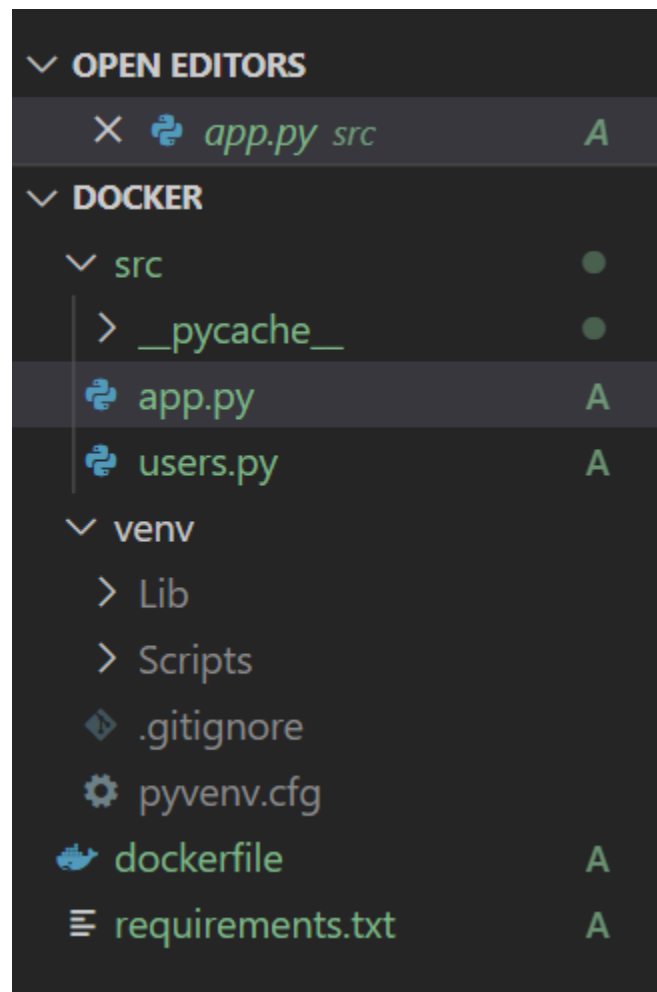
A todas las empresas que estén considerando soluciones de orquestación de contenedores, Rancher les ofrece una atractiva y muy válida opción.

Rancher es una pila de software que se utiliza para gestionar clústeres Kubernetes. Se trata básicamente de un software que DevOps puede utilizar al adoptar el usuario de contenedores. Rancher incluye una distribución completa de Kubernetes, Docker Swarm y Apache Mesos, lo que facilita la gestión de clústeres de contenedores en cualquier plataforma de nube.

Una de las ventajas significativas de Rancher es la capacidad de gestionar múltiples clústeres Kubernetes de forma simplificada.

### Ejemplo usando minikube:

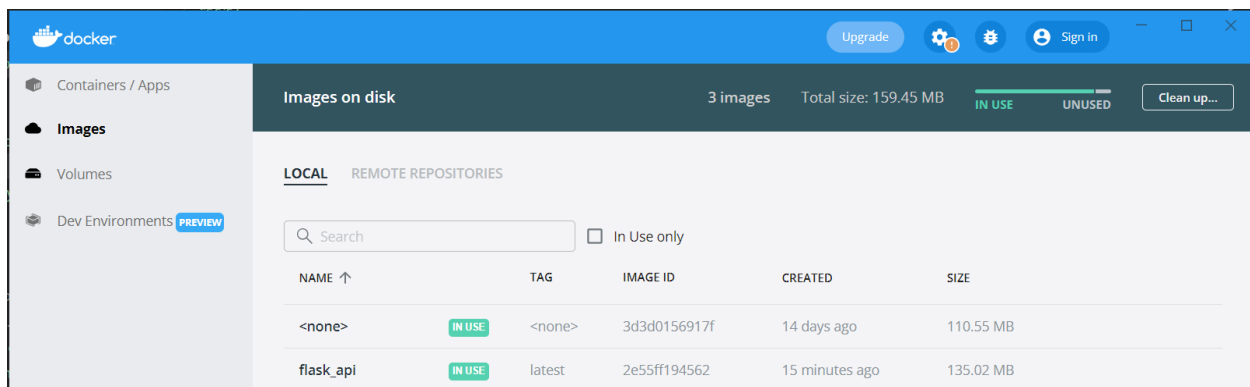
Utilizando el código de Python realizado en la práctica anterior, con la cual se realizó una pequeña práctica en la que se corría una página web simple con un Hola mundo que era corrido mediante flask, para posteriormente agregar el archivo Docker file el cual contiene los requerimientos necesarios para correr este programa en cualquier computador sin necesidad de instalar todas las librerías necesarias para ello.



Posteriormente vamos a crear una nueva imagen para este programa y subirlo nuevamente a Docker hub para posteriormente utilizar e implementar los kubernetes.

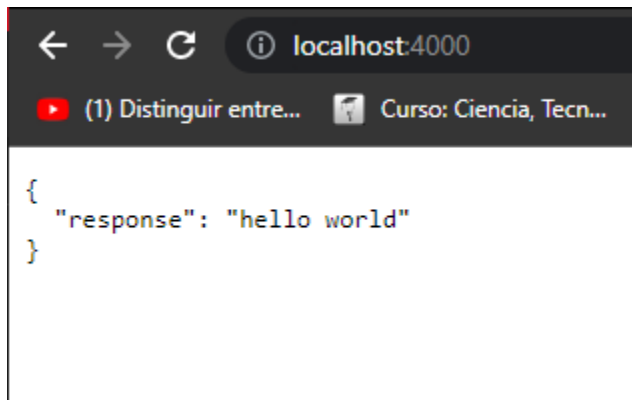
```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker (master)
$ docker build -t flask_api .
[+] Building 15.4s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                                0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/alpine:3.10                  4.2s
=> [1/5] FROM docker.io/library/alpine:3.10@sha256:451eee8bedcb2f029756dc3e9d73bab0e7943c1ac55cfff3a4861c52a 0.0s
=> [internal] load build context                                              1.0s
=> => transferring context: 187.03kB                                           0.9s
=> CACHED [2/5] RUN apk add --no-cache python3-dev && pip3 install --upgrade pip 0.0s
=> CACHED [3/5] WORKDIR /app                                                  0.0s
=> [4/5] COPY . /app                                                         2.6s
=> [5/5] RUN pip3 --no-cache-dir install -r requirements.txt                 6.9s
=> exporting to image                                                         0.4s
=> => exporting layers                                                         0.4s
```

Y podemos observar cómo efectivamente se crea la imagen llamada flask\_api que hemos creado anteriormente.



NAME ↑	TAG	IMAGE ID	CREATED	SIZE
<none>	<none>	3d3d0156917f	14 days ago	110.55 MB
flask_api	latest	2e55ff194562	15 minutes ago	135.02 MB

Con el comando run podemos observar cómo funciona nuestra API de manera local mediante localhost.



## Utilizamos minikube:

Instalamos kubernetes en la computadora y revisamos la version.

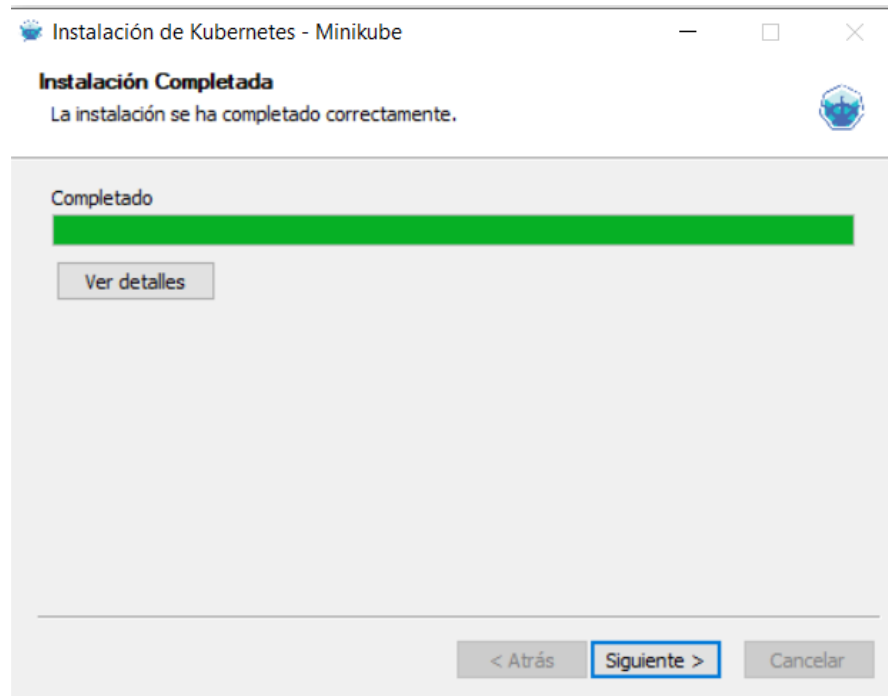
```
PS C:\WINDOWS\system32> choco install kubernetes-cli
Chocolatey v1.1.0
Installing the following packages:
kubernetes-cli
By installing, you accept licenses for the packages.
Progress: Downloading kubernetes-cli 1.23.5... 100%

kubernetes-cli v1.23.5 [Approved]
kubernetes-cli package files install completed. Performing other installation steps.
The package kubernetes-cli wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): y

Extracting 64-bit C:\ProgramData\chocolatey\lib\kubernetes-cli\tools\kubernetes-client-windows-amd64.tar.gz to C:\ProgramData\chocolatey\lib\kubernetes-cli\tools...
C:\ProgramData\chocolatey\lib\kubernetes-cli\tools
Extracting 64-bit C:\ProgramData\chocolatey\lib\kubernetes-cli\tools\kubernetes-client-windows-amd64.tar to C:\ProgramData\chocolatey\lib\kubernetes-cli\tools...
C:\ProgramData\chocolatey\lib\kubernetes-cli\tools
ShimGen has successfully created a shim for kubectl-convert.exe
ShimGen has successfully created a shim for kubectl.exe
The install of kubernetes-cli was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\kubernetes-cli\tools'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32> kubectl version
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.5", GitCommit:"5c99e2ac2ff9a3c549d9ca665e7bc05a3e18f07e", GitTreeState:"clean", BuildDate:"2021-12-16T08:38:33Z", GoVersion:"go1.16.12", Compiler:"gc", Platform:"windows/amd64"}
Unable to connect to the server: dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
PS C:\WINDOWS\system32>
```

Instalamos minikube, debido a que yo ya tenia virtual box.



Una vez instalado, agregamos al path mediante comandline, para posteriormente poder utilizarlo.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\WINDOWS\system32> $oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)
PS C:\WINDOWS\system32> if ($oldPath.Split(';') -notcontains 'C:\minikube'){ `
>> [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine)
>> }
PS C:\WINDOWS\system32>
```



Iniciamos minikube para que se inicie la maquina virtual y podamos utilizarlo.

```
PS C:\WINDOWS\system32> minikube start
* minikube v1.25.2 en Microsoft Windows 10 Home 10.0.19044 Build 19044
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Restarting existing docker container for "minikube" ...
* Preparando Kubernetes v1.23.3 en Docker 20.10.12...
  - kubelet.housekeeping-interval=5m
* Verifying Kubernetes components...
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.5648776s
* Restarting the docker service may improve performance.
  - 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" namespace by default
PS C:\WINDOWS\system32>
```

Posteriormente creamos la imagen para el contenedor, y el deployment necesario para el servicio.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker (master)
$ kubectl create deployment hello-node --image=k8s.gcr.io/echoserver:1.4
deployment.apps/hello-node created
```

Observamos que efectivamente se haya creado el deployment.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker (master)
$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
hello-node    0/1     1            0           10s
```

Observamos que efectivamente se haya creado el pod igualmente.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker (master)
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
hello-node-6b89d599b9-d8bgs        1/1     Running   0           38s
```

Exponemos el servicio dentro del puerto 8080 de la imagen antes creada.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker (master)
$ kubectl expose deployment hello-node --type=LoadBalancer --port=8080
service/hello-node exposed
```

Executamos el servicio para observar el funcionamiento.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker (master)
$ minikube service hello-node
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time
s

💡 Restarting the docker service may improve performance.
🔧 Starting tunnel for service hello-node.
🚀 Opening service default/hello-node in default browser...
! Porque estás usando controlador Docker en windows, la terminal debe abrirse para ejecutarlo.
```

Observamos que nos cambia la url a un http como se muestra en la siguiente imagen.

```
* "The 'metrics-server' addon is disabled
$ minikube service hello-node
|-----|-----|-----|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|-----|-----|-----|
| default | hello-node | 8080 | http://172.17.0.11:30933 |
|-----|-----|-----|-----|
* Opening service default/hello-node in default browser...
Minikube Dashboard is not supported via the interactive terminal experience.

Please click the 'Preview Port 30000' link above to access the dashboard.
This will now exit. Please continue with the rest of the tutorial.

X Exiting due to HOST_BROWSER: exit status 1
*
* If the above advice does not help, please let us know:
- https://github.com/kubernetes/minikube/issues/new/choose

$
```

Nos logueamos para que se acepten los datos y se permita el acceso.

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker3 (main)
$ docker tag b596e914ad9c vargas2000/flask-kubernetes

david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker3 (main)
$ docker push
"docker push" requires exactly 1 argument.
See 'docker push --help'.

Usage:  docker push [OPTIONS] NAME[:TAG]

Push an image or a repository to a registry
```

Ejecutamos el dashboard y el deployment para ejecutar la instancia de kubernetes.

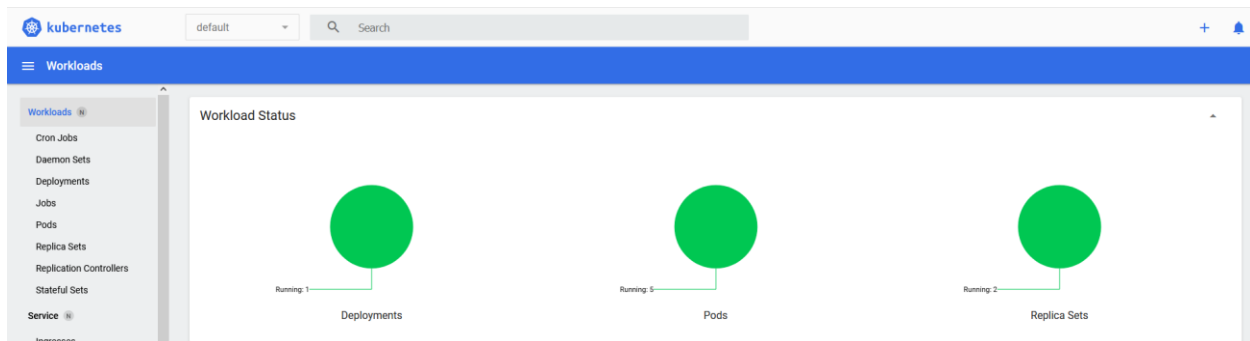
```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker3 (main)
$ minikube dashboard
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 5.9049968
s

💡 Restarting the docker service may improve performance.
😓 Verifying dashboard health ...
🔧 Launching proxy ...
😓 Verifying proxy health ...

david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker3 (main)
$ kubectl apply -f deployment.yaml
service/flask-test-service unchanged
deployment.apps/flask-test-app configured
```

```
david@LAPTOP-OMJ7SMPI MINGW64 ~/OneDrive/Escritorio/6to semestre/tolerante a fallos/docker3 (main)
$ minikube dashboard
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 5.959378s
💡 Restarting the docker service may improve performance.
😓 Verifying dashboard health ...
🔧 Launching proxy ...
😓 Verifying proxy health ...
🔗 Opening http://127.0.0.1:62252/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy
/ in your default browser...
```

Observamos que cambian de color los pods, indicando que ya están activos en la información.



## Conclusión:

Esta práctica fue bastante interesante y complicada en lo personal de realizar debido a que tuve algunos problemas con que una vez corriendo el servicio no me generaba o se terminaba el tiempo de espera por lo que el navegador me generaba un error ocasionando que a pesar de que el servicio estaba ejecutándose no se mostrará nada dentro del navegador por lo que esto fue una complicación bastante tediosa de resolver, pero en lo personal esta práctica me pareció bastante interesante para realizar grandes prácticas o proyectos con fin de observar su funcionamiento dentro de un ambiente real.

## Enlace al repositorio:

<https://github.com/David-1212/kubernetes.py>

## Bibliografía:

- Atlassian. (n.d.). *¿Qué es Kubernetes?* Retrieved April 3, 2022, from <https://www.atlassian.com/es/continuous-delivery/microservices/kubernetes>
- IBM Cloud Docs. (n.d.). blog. Retrieved April 3, 2022, from <https://cloud.ibm.com/docs/containers?topic=containers-ingress-about&locale=es>
- *imagen de ingress con kubernetes* - Google Zoeken. (n.d.). blog. Retrieved April 3, 2022, from [https://www.google.com/search?q=imagen+de+ingress+con+kubernetes&tbm=isch&ved=2ahUKEwjV4Nfrq\\_n2AhUPmmoFHbuyDdMQ2-cCegQIABAA&oq=imagen+de+ingress+con+kubernetes&gs\\_lcp=CgNpbWcQAzoHCCMQ7wMQJ1DaBVi1HmDXH2gAcAB4AIABmgGIAf0NkgEDNy45mAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=ZFVKYtWeA4-0qtsPu-](https://www.google.com/search?q=imagen+de+ingress+con+kubernetes&tbm=isch&ved=2ahUKEwjV4Nfrq_n2AhUPmmoFHbuyDdMQ2-cCegQIABAA&oq=imagen+de+ingress+con+kubernetes&gs_lcp=CgNpbWcQAzoHCCMQ7wMQJ1DaBVi1HmDXH2gAcAB4AIABmgGIAf0NkgEDNy45mAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=ZFVKYtWeA4-0qtsPu-)

[W2mA0&bih=937&biw=1920&rlz=1C1CHBD\\_esMX890MX890#imgrc=YQzb5aZ-TS5QuM](https://www.redhat.com/es/topics/containers/what-is-kubernetes)

- *¿Qué es Kubernetes?* (n.d.). blog. Retrieved April 3, 2022, from <https://www.redhat.com/es/topics/containers/what-is-kubernetes>
- Torres, G. (2021, October 7). *Acceder a tus aplicaciones en Kubernetes a través de Ingress*. return(GiS); Retrieved April 3, 2022, from <https://www.returngis.net/2019/04/acceder-a-tus-aplicaciones-en-kubernetes-a-traves-de-ingress/>