

Kubernetes

Alumno: Oswaldo Gael Cervantes Castoño

Código: 219747468

Computación Tolerante a Fallas

Sección D06

Profesor: Michel Emanuel López Franco



Kubernetes

¿Qué es Kubernetes?



kubernetes

Kubernetes es una plataforma de código abierto que orquesta sistemas de tiempo de ejecución de contenedores (maneja de forma nativa algunos como Docker) en un clúster de recursos de hardware en red. Originalmente lo desarrolló Google, que necesitaba una nueva forma de ejecutar miles de millones de contenedores a la semana a gran escala.

Se introdujo Kubernetes para ayudar a automatizar y gestionar contenedores en la infraestructura de la nube (como máquinas virtuales y hardware virtual) que vinieron a sustituir al hardware físico en el ámbito de DevOps. Asimismo, ayuda a gestionar arquitecturas de aplicaciones de micro servicios y ahora es una herramienta crítica para compilar sólidas y modernas canalizaciones de CI/CD.

Como señala Google, “el principal objetivo de Kubernetes es facilitar la implementación y la gestión de sistemas distribuidos complejos, además de seguir beneficiándose del uso mejorado que permiten los contenedores”. A veces, se conoce a Kubernetes como K8s o Kube, y ahora lo mantiene la Cloud Native Computing Foundation.

¿Qué es Ingress?

Un objeto API que gestiona el acceso externo a los servicios en un clúster, normalmente HTTP. Ingress puede proporcionar equilibrio de carga, terminación SSL y alojamiento virtual basado en nombres.

Ingress expone las rutas HTTP y HTTPS desde fuera del clúster a los servicios dentro del clúster. El enrutamiento del tráfico está controlado por reglas definidas en el recurso Ingress. Aquí hay un ejemplo simple donde un Ingress envía todo su tráfico a un servicio:



Se puede configurar un Ingress para brindar servicios de direcciones URL accesibles externamente, equilibrar la carga del tráfico, finalizar SSL/TLS y ofrecer alojamiento virtual basado en el nombre. Un controlador de Ingress es responsable de cumplir con el Ingress, generalmente con un balanceador de carga, aunque también puede configurar su enrutador perimetral o interfaces adicionales para ayudar a manejar el tráfico. Un Ingress no expone puertos o protocolos arbitrarios.

La exposición de servicios distintos de HTTP y HTTPS a Internet suele utilizar un servicio de tipo `Service.Type=NodePort` o `Service.Type=LoadBalancer`.

¿Qué es un LoadBalancer?

Un LoadBalancer o balanceador de carga permite repartir la carga de trabajo entre los diferentes servidores o aplicaciones y puede instalarse en una infraestructura tanto física como virtual. Así pues, los programas de balanceo de carga adoptan la forma de un controlador de entrega de aplicaciones o ADC (Application Delivery Controller), permitiendo al usuario escalar la carga automáticamente en función de las previsiones de tráfico. El ADC identifica en tiempo real qué servidores o aplicaciones son las más adecuadas para responder a una petición, garantizando un nivel de rendimiento estable en el clúster. En caso de avería, es posible redirigir el tráfico hacia otro recurso con capacidad para absorberlo. Así pues, existen diferentes configuraciones posibles.

El LoadBalancer interviene entre el visitante y el servidor analizando las peticiones, determinando qué máquina está disponible para responder y, por último, transmitiendo estas peticiones. También es posible añadir servidores en caso de necesidad.

Cervantes Castoño Oswaldo Gael. Computación Tolerante a Fallas.

El balanceo de carga, es solo una de las posibles aplicaciones del LoadBalancer. Y es que esta tecnología también resulta especialmente útil para descongestionar un certificado SSL, actualizar grupos de aplicaciones o incluso enrutar sus dominios.

Existen dos tipos de LoadBalancer:

- LoadBalancer L4 o balanceadores de carga de red
- LoadBalancer L7 o balanceadores de carga de aplicación

La forma estándar para definir un servicio de tipo balanceador de carga es la siguiente:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  clusterIP: 10.0.171.239
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
      - ip: 192.0.2.127
```

¿Qué es Rancher?



Rancher es un software de código abierto 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.

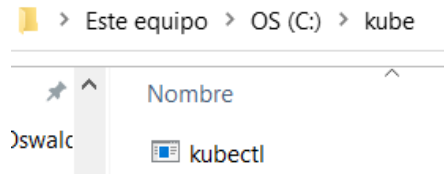
Algunas de las ventajas más notables de Rancher para elegirlo por encima de otras distribuciones de Kubernetes son:

- **Es fácil de usar.** Una de las razones por las que uno elegiría Rancher en lugar de cualquier otra plataforma Kubernetes es la interfaz de usuario web simplificada que hace que sea fácil hacer cualquier cosa que necesites. Es una plataforma con la que incluso los desarrolladores que no tienen tanta experiencia con Kubernetes pueden iniciarse fácilmente.
- **Se puede desplegar fácilmente en cualquier infraestructura de nube.** Otra ventaja crítica que tiene Rancher sobre otras plataformas Kubernetes es su compatibilidad con diferentes plataformas en la nube; así, puedes desplegarlo rápidamente en cualquier infraestructura en la nube.
- **Simplifica la gestión de los clústeres.** Rancher es probablemente la mejor opción para gestionar varios clústeres de Kubernetes desde una sola interfaz. Su capacidad para gestionar múltiples clústeres es uno de los puntos fuertes significativos que se construyeron en el núcleo de Rancher.
- **Incluye balanceo de recursos y monitorización automáticas.** Esta es una de las principales características que se incluyen en Rancher, que es muy útil si usted tiene la intención de desplegar un sistema que probablemente obtendrá un gran tráfico.
- **Es de código abierto y totalmente gratuito.** RKE, K3s y todos los demás productos de Rancher son de código abierto y de uso gratuito para cualquiera. Si no tiene un presupuesto para gastar en un software de gestión de contenedores, entonces Rancher es la mejor opción para usted. Sin embargo, para obtener el apoyo de los laboratorios Rancher tendrá que pagar algo de dinero.

Ejemplo: Flask + Docker + Kubernetes

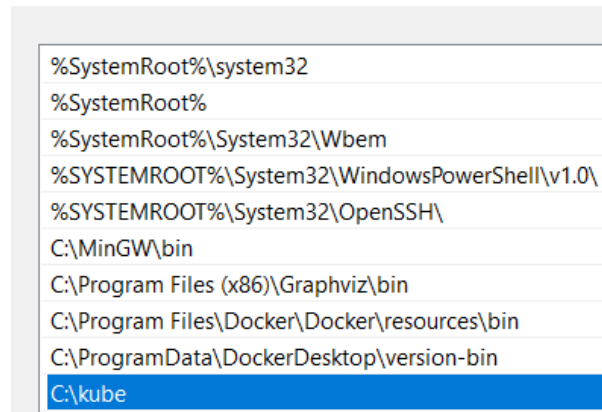
El primer paso es descargar el ejecutable para el entorno de Kubernetes (kubectl). Lo colocamos en un directorio llamado “kube”:

Cervantes Castoño Oswaldo Gael. Computación Tolerante a Fallas.



Añadimos la ruta del directorio creado al Path:

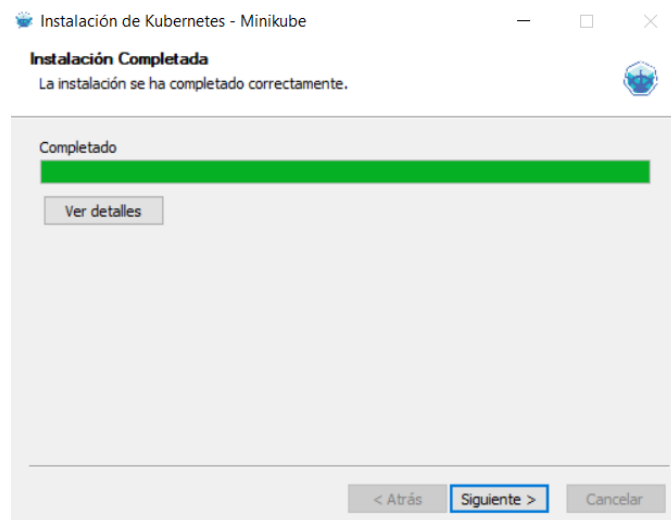
Editar variable de entorno



Ingresamos al PowerShell de Windows y verificamos que Kubernetes ya se encuentra disponible:

```
PS C:\Users\Gael> kubectl version --client
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"}
PS C:\Users\Gael>
```

El siguiente paso es descargar e instalar Minikube:



Cervantes Castoño Oswaldo Gael. Computación Tolerante a Fallas.

Ahora tenemos que ejecutar el PowerShell de Windows como administrador para posteriormente pegar un par de comandos que se encuentran en la documentación de Minikube para añadir el entorno de Minikube al Path:

```
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>
```

Verificamos que ya se encuentra disponible por medio del siguiente comando, el cual despliega una guía rápida de comandos que se pueden utilizar:

```
PS C:\Users\Gael> minikube
minikube provisions and manages local Kubernetes clusters optimized for development workflows.

Basic Commands:
  start      Starts a local Kubernetes cluster
  status     Gets the status of a local Kubernetes cluster
  stop       Stops a running local Kubernetes cluster
  delete     Elimina un cluster de Kubernetes local
  dashboard  Acceder al panel de Kubernetes que corre dentro del cluster minikube
  pause      pause Kubernetes
  unpause    unpause Kubernetes
```

Ya que voy a utilizar el framework “Flask”, es necesario instalarlo por medio de la terminal, en este caso yo lo instalé por medio de la terminal de Windows embebida en Visual Studio Code:

```
D:\flask_docker_kubernetes>pip install flask
Collecting flask
  Downloading Flask-2.1.1-py3-none-any.whl (95 kB)
    | 95 kB 1.6 MB/s
Requirement already satisfied: Jinja2>=3.0 in c:\users\gael\appdata\local\programs\python\python310\lib\site-packages (from flask) (3.0.3)
Requirement already satisfied: click>=8.0 in c:\users\gael\appdata\local\programs\python\python310\lib\site-packages (from flask) (8.0.4)
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Werkzeug>=2.0
  Downloading Werkzeug-2.1.1-py3-none-any.whl (224 kB)
    | 224 kB 2.2 MB/s
Requirement already satisfied: colorama in c:\users\gael\appdata\local\programs\python\python310\lib\site-packages (from click>=8.0->flask) (0.4.4)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\gael\appdata\local\programs\python\python310\lib\site-packages (from Jinja2>=3.0->flask) (2.1.0)
Installing collected packages: Werkzeug, itsdangerous, flask
Successfully installed Werkzeug-2.1.1 flask-2.1.1 itsdangerous-2.1.2
WARNING: You are using pip version 21.2.4; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\Gael\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip' command.

D:\flask_docker_kubernetes>
```

La aplicación como tal es sencilla, consiste de un endpoint que retorna el tiempo en el que ha sido llamada la petición en un objeto en formato json:

```
app.py > ...
1  from flask import Flask, jsonify
2  import time
3
4  app = Flask(__name__)
5
6  app.route("/")
7  def hello_world():
8      return jsonify({"Time to call": time.time()})
9
10 if __name__ == "__main__":
11     app.run(host="0.0.0.0", debug=True)
```

El siguiente paso es definir el “Dockerfile”, en el cual se especifica en primer lugar la versión de Python a utilizar, posteriormente se define el directorio de trabajo, se instalan los paquetes contenidos en el archivo de “requirements.txt” y finalmente se ejecutan los comandos necesarios para correr la aplicación:

```
Dockerfile
1  FROM python:3.7
2  RUN mkdir /app
3  WORKDIR /app/
4  ADD . /app/
5  RUN pip install -r requirements.txt
6  CMD ["python", "/app/app.py"]
```

Definimos el archivo de “requirements.txt”, el cual, ya que se trata de una aplicación sencilla, simplemente incorpora “flask”:

```
requirements.txt
1  flask
```

Por último creamos nuestra imagen de Docker por medio del comando siguiente:

```
D:\flask_docker_kubernetes>docker build -t flask_docker_kubernetes .
[+] Building 4.7s (5/10)
=> [internal] load build definition from Dockerfile                                0.2s
=> => transferring dockerfile: 164B                                              0.0s
=> [internal] load .dockerignore                                                  0.1s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/library/python:3.7                    4.1s
=> [auth] library/python:pull token for registry-1.docker.io                   0.0s
=> [1/5] FROM docker.io/library/python:3.7@sha256:55d468198d18e6878f9f029b5efed8a985c63f911ede599c48a3784805a90890 0.1s
=> => resolve docker.io/library/python:3.7@sha256:55d468198d18e6878f9f029b5efed8a985c63f911ede599c48a3784805a90890 0.0s
=> => sha256:9baf437a1badb6aad2dae5f2cd4a7b53a6c7ab6c14cba1ed1ecb42b4822b0e87 0B / 5.16MB 0.1s
=> => sha256:6ade5c59e324bd7cf369c72ad781c23d37e8fb48c9bbb4abbeafafd9be4cc35 0B / 10.87MB 0.1s
=> => sha256:55d468198d18e6878f9f029b5efed8a985c63f911ede599c48a3784805a90890 1.86kB / 1.86kB 0.0s
=> [internal] load build context                                                0.1s
=> => transferring context: 481B                                              0.0s
```


Esperamos a que termine de crearse:

```
D:\flask_docker_kubernetes>docker build -t flask_docker_kubernetes .
[+] Building 117.1s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 164B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.7
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/5] FROM docker.io/library/python:3.7@sha256:55d468198d18e6878f9f029b5efed8a985c63f911ede599c48a3784805a90890
=> => resolve docker.io/library/python:3.7@sha256:55d468198d18e6878f9f029b5efed8a985c63f911ede599c48a3784805a90890
=> => sha256:9baf437a1badb6aad2dae5f2cd4a7b53a6c7ab6c14cba1ed1ecb42b4822b0e87 5.16MB / 5.16MB
=> => sha256:6ade5c59e324bd7cf369c72ad781c23d37e8fb48c9bbb4abbeccafaf9be4cc35 10.87MB / 10.87MB
=> => sha256:55d468198d18e6878f9f029b5efed8a985c63f911ede599c48a3784805a90890 1.86kB / 1.86kB
=> => sha256:d0824cdd22ae5bd719b507ad6bc6f0ac15c36004e94e49b84a55a07998ef874b 2.22kB / 2.22kB
=> => sha256:d475516974f2173912b23993a4c5c1a36b0743cfa65ebd65262d20315f772b24 9.23kB / 9.23kB
=> => sha256:dbba69284b2786013fe94fefe0c2e66a7d3cecb20f6d691d71dac891ee37be5 54.94MB / 54.94MB
=> => sha256:b19a994f6d4cddb620339bd2e4ad47b229f14276b542060622ae447649294e5d 54.58MB / 54.58MB
=> => sha256:8fc2294f89de5e20d0ae12149d6136444bcb8c775ea745f06f2eb775ab4504cd 196.55MB / 196.55MB
=> => sha256:9dc715194c21dec8f4d20ea4faa9929b2297b24c123fc8459709266f43e83449 6.29MB / 6.29MB
=> => extracting sha256:dbba69284b2786013fe94fefe0c2e66a7d3cecb20f6d691d71dac891ee37be5
=> => sha256:a85c9eeca45025b755f444c0b4f5bdbfd07d508edd12c5ee9414efb2104bbc138 15.49MB / 15.49MB
=> => sha256:d767a02310e265ef8405dd0aef4c2af6fea3f7540160d3b26360e9030776aa23 235B / 235B
=> => sha256:4e2030e349abc011ad31e1ba2e3ff1b70d922a711ce673cd68201b9513ae6b2e 2.87MB / 2.87MB
=> => extracting sha256:9baf437a1badb6aad2dae5f2cd4a7b53a6c7ab6c14cba1ed1ecb42b4822b0e87
=> => extracting sha256:6ade5c59e324bd7cf369c72ad781c23d37e8fb48c9bbb4abbeccafaf9be4cc35
=> => extracting sha256:b19a994f6d4cddb620339bd2e4ad47b229f14276b542060622ae447649294e5d
=> => extracting sha256:8fc2294f89de5e20d0ae12149d6136444bcb8c775ea745f06f2eb775ab4504cd
=> => extracting sha256:9dc715194c21dec8f4d20ea4faa9929b2297b24c123fc8459709266f43e83449
=> => extracting sha256:a85c9eeca45025b755f444c0b4f5bdbfd07d508edd12c5ee9414efb2104bbc138
=> => extracting sha256:d767a02310e265ef8405dd0aef4c2af6fea3f7540160d3b26360e9030776aa23
=> => extracting sha256:4e2030e349abc011ad31e1ba2e3ff1b70d922a711ce673cd68201b9513ae6b2e
=> [internal] load build context
=> => transferring context: 481B
=> [2/5] RUN mkdir /app
=> [3/5] WORKDIR /app/
=> [4/5] ADD . /app/
=> [5/5] RUN pip install -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:df504d6831e34caa5400ccf5713fcbdd76fd97e1df8ea3c5e57734cab11de968
=> => naming to docker.io/library/flask_docker_kubernetes

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

D:\flask_docker_kubernetes>
```

Para verificar que la imagen se ha creado, podemos utilizar el siguiente comando que despliega información relativa a las imágenes creadas al momento:

```
D:\flask_docker_kubernetes>docker images
REPOSITORY          TAG          IMAGE ID          CREATED           SIZE
flask_docker_kubernetes  latest      df504d6831e3     About a minute ago  917MB

D:\flask_docker_kubernetes>
```

Procedemos a inicializar Minikube (clúster) por medio del comando que se muestra en la siguiente imagen:

```
D:\flask_docker_kubernetes>minikube start
🐸 minikube v1.25.2 en Microsoft Windows 10 Home Single Language 10.0.19044 Build 19044
🌟 Controlador docker seleccionado automáticamente. Otras opciones: virtualbox, ssh
👍 Starting control plane node minikube in cluster minikube
📦 Pulling base image ...
📦 Descargando Kubernetes v1.23.3 ...
> preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB 100.00% 3.12 MiB
> gcr.io/k8s-minikube/kicbase: 379.06 MiB / 379.06 MiB 100.00% 2.07 MiB p/
```

Una vez que ha finalizado se despliega toda la información que se muestra a continuación:

```
D:\flask_docker_kubernetes>minikube start
🐸 minikube v1.25.2 en Microsoft Windows 10 Home Single Language 10.0.19044 Build 19044
🌟 Controlador docker seleccionado automáticamente. Otras opciones: virtualbox, ssh
👍 Starting control plane node minikube in cluster minikube
📦 Pulling base image ...
📦 Descargando Kubernetes v1.23.3 ...
> preloaded-images-k8s-v17-v1...: 505.68 MiB / 505.68 MiB 100.00% 3.12 MiB
> gcr.io/k8s-minikube/kicbase: 379.06 MiB / 379.06 MiB 100.00% 2.07 MiB p/
🔥 Creando docker container (CPUs=2, Memory=3000MB) ...| E0404 19:53:05.682843 10608 kic.go:267] icacfs failed applying permissions - err - [%!s<nil>]], output - [archivo procesado: C:\Users\Gael\.minikube\machines\minikube\id_rsa
Se procesaron correctamente 1 archivos; error al procesar 0 archivos]

📦 Preparando Kubernetes v1.23.3 en Docker 20.10.12...
  ▪ kubelet.housekeeping-interval=5m
  ▪ Generando certificados y llaves
  ▪ Iniciando plano de control
  ▪ Configurando reglas RBAC...
🔍 Verifying Kubernetes components...
  ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Complementos habilitados: storage-provisioner, default-storageclass
👍 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

D:\flask_docker_kubernetes>
```


Para poder utilizar nuestra aplicación en Kubernetes, primero es necesario subir nuestra imagen creada de forma local a DockerHub, para lo cual primero es necesario iniciar sesión por medio del siguiente comando (para que funcione debemos tener iniciada la sesión tanto en DockerHub como en DockerDesktop):


```
D:\flask_docker_kubernetes>docker login
Authenticating with existing credentials...
Login Succeeded

D:\flask_docker_kubernetes>
```

Otro aspecto importante a tener en cuenta es que previo a los siguientes pasos es necesario en primer lugar haber creado un repositorio en DockerHub con el nombre de nuestra imagen para solamente subir dicha imagen:

gael65 / flask_docker_kubernetes

This repository does not have a description 

 Last pushed: never

Asignamos una etiqueta a nuestra imagen seguida del nombre que tiene el repositorio que creamos en DockerHub:


```
D:\flask_docker_kubernetes>docker tag df504d6831e3 gael65/flask_docker_kubernetes
D:\flask_docker_kubernetes>
```


Finalmente hacemos push para poder disponer de nuestra imagen de Docker en la nube y que de esta forma pueda ser accedida por Kubernetes en los pasos que siguen en el tutorial:

```
D:\flask_docker_kubernetes>docker push gael65/flask_docker_kubernetes
Using default tag: latest
The push refers to repository [docker.io/gael65/flask_docker_kubernetes]
f21c8263d28a: Pushed
417e658760c6: Pushed
5f70bf18a086: Pushed
e7c91d8ec50e: Pushed
f5e951a0e3c7: Mounted from library/python
3d8c09d52b7b: Mounted from library/python
39ca3c789e6d: Mounted from library/python
89f49a7a4e4a: Mounted from library/python
c5579a205adc: Mounted from library/python
7a7698da17f2: Mounted from library/python
d59769727d80: Mounted from library/python
348622fdcc61: Mounted from library/python
4ac8bc2cd0be: Mounted from library/python
latest: digest: sha256:0fcd57ef3df98b0fb92b204b1938d9c660c7e5d2d4163cad1293d83344e90e17 size: 3049
D:\flask_docker_kubernetes>
```


Una vez que se ha hecho el push podemos verlo reflejado de forma inmediata en DockerHub:

gael65 / flask_docker_kubernetes


This repository does not have a description 

 Last pushed: a few seconds ago

Tags and Scans

 VULNERABILITY SCANNING - DISABLED
[Enable](#)

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
 latest		---	a few seconds ago

[See all](#)

Por medio del siguiente comando se despliega o se crea el servicio y las instancias de la aplicación requeridas en Kubernetes:

```
D:\flask_docker_kubernetes>kubectl apply -f deployment.yaml
service/flask-test-service created
deployment.apps/flask-test-app created

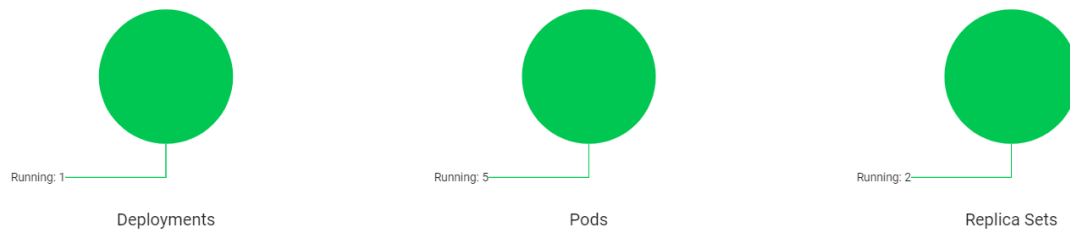
D:\flask_docker_kubernetes>
```

Finalmente podemos visualizar la información relacionada con nuestra aplicación desplegando la interfaz o “dashboard” de Minikube por medio del siguiente comando:

```
D:\flask_docker_kubernetes>minikube dashboard
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.773437s
💡 Restarting the docker service may improve performance.
😄 Verifying dashboard health ...
🔧 Launching proxy ...
😄 Verifying proxy health ...
🔗 Opening http://127.0.0.1:61226/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default browser..
```

Una vez que se abre el dashboard podemos observar información de forma gráfica como la siguiente imagen que muestra el estado de los pods:

Workload Status



Podemos ver los Deployments y su estado actual, en este caso solamente tengo uno que es el que recién acabo de crear:

Deployments

Name	Namespace	Images
● flask-test-app	default	gael65/flask_docker_kubernetes

Finalmente podemos observar los pods o instancias de procesos en ejecución:

Pods

Name	Namespace	Images	Labels	Node
● flask-test-app-6c699fb5bb-bz88z	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
● flask-test-app-6c699fb5bb-cq66f	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
● flask-test-app-6c699fb5bb-v44nl	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
● flask-test-app-6c699fb5bb-mrtgd	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
● flask-test-app-6c699fb5bb-r9vtk	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube

Cervantes Castoño Oswaldo Gael. Computación Tolerante a Fallas.

Podemos hacer una prueba de tolerancia a fallas intentando borrar de forma manual una instancia de pods:

Delete a resource

Are you sure you want to delete Pod *flask-test-app-6c699fb5bb-bz88z* in namespace *default*?

 This action is equivalent to: `kubectl delete -n default pod flask-test-app-6c699fb5bb-bz88z`

Delete

Cancel






Aquí se muestra la instancia que estamos borrando pero además se muestra otra instancia nueva que se ha creado para sustituir la que quitamos ya que se debe respetar el número de pods que definimos en nuestro archivo con extensión “.yaml”:

Pods

Name	Namespace	Images	Labels	Node
 flask-test-app-6c699fb5bb-jswrg	default		 	minikube
 flask-test-app-6c699fb5bb-kc4gz	default		 	minikube
 flask-test-app-6c699fb5bb-cq66f	default		 	minikube
 flask-test-app-6c699fb5bb-v44nl	default		 	minikube
 flask-test-app-6c699fb5bb-mrtgd	default		 	minikube
 flask-test-app-6c699fb5bb-r9vtk	default		 	minikube

Finalmente podemos observar cómo se restaura la nueva instancia de pods creada:

Pods

Name	Namespace	Images	Labels	Node
 flask-test-app-6c699fb5bb-jswrg	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
 flask-test-app-6c699fb5bb-cq66f	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
 flask-test-app-6c699fb5bb-v44nl	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
 flask-test-app-6c699fb5bb-mrtgd	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube
 flask-test-app-6c699fb5bb-r9vtk	default	gael65/flask_docker_kubernetes	app: flask-test-app pod-template-hash: 6c699fb5bb	minikube

En una nueva terminal podemos probar el comando siguiente para visualizar en la consola la cantidad de pods existentes y su estado:

```
D:\flask_docker_kubernetes>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
flask-test-app-6c699fb5bb-cq66f    1/1     Running   0           17m
flask-test-app-6c699fb5bb-jswrg    1/1     Running   0           85s
flask-test-app-6c699fb5bb-mrtgd    1/1     Running   0           18m
flask-test-app-6c699fb5bb-r9vtk    1/1     Running   0           18m
flask-test-app-6c699fb5bb-v44nl    1/1     Running   0           18m

D:\flask_docker_kubernetes>
```

También podemos consultar los servicios actuales:

```
D:\flask_docker_kubernetes>kubectl get services
NAME                TYPE             CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
flask-test-service  LoadBalancer    10.103.197.236  <pending>    6000:30296/TCP   38m
kubernetes          ClusterIP        10.96.0.1       <none>       443/TCP          41m

D:\flask_docker_kubernetes>
```


Cervantes Castoño Oswaldo Gael. Computación Tolerante a Fallas.

Enlace al repositorio en GitHub:

<https://github.com/gaeltcervantes65/Kubernetes.git>

Enlace al repositorio en DockerHub:

https://hub.docker.com/repository/docker/gael65/flask_docker_kubernetes

Conclusiones

Como conclusiones para el desarrollo de esta actividad en general puedo hacer mención de que considero muy importante el haber conocido una herramienta tan potente como lo es Kubernetes; en lo personal, solamente había escuchado su nombre en el pasado, pero no sabía de qué se trata mucho menos de sus enormes alcances en el ámbito de los sistemas distribuidos y computación en la nube.

Considero que la investigación realizada al principio de este documento me ayudó en primer lugar a conocer conceptos importantes como por ejemplo el de clúster o el de balanceador de carga, ya que son la base para comprender el funcionamiento de Kubernetes una vez que se pone en marcha su uso; durante la realización de la práctica o ejemplo del uso de Kubernetes a comparación de actividades pasadas, en esta ocasión no tuve complicaciones muy grandes para lograr el objetivo aunque también es necesario mencionar que me centré en un ejemplo básico que me diera una noción o futuras referencias para utilizar una herramienta como esta.

Fuentes de información

- Atlassian. (s. f.). ¿Qué es Kubernetes? Recuperado 3 de abril de 2022, de <https://www.atlassian.com/es/continuous-delivery/microservices/kubernetes>
- Ingress. (2022, 19 marzo). Kubernetes. Recuperado 3 de abril de 2022, de <https://kubernetes.io/docs/concepts/services-networking/ingress/>
- Load balancer con Kubernetes. (s. f.). OVHcloud. Recuperado 4 de abril de 2022, de <https://www.ovhcloud.com/es/public-cloud/kubernetes/kubernetes-load-balancer/>

- S. (2021, 4 mayo). Todo lo que necesita saber sobre Rancher: gestión de Kubernetes para empresas. SiXe Ingeniería. Recuperado 4 de abril de 2022, de <https://sixe.es/noticias/suse-rancher-kubernetes-toda-la-informacion>
- Install and Set Up kubectl on Windows. (2022, 11 marzo). Kubernetes. Recuperado 4 de abril de 2022, de <https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>
- minikube start. (s. f.). Minikube. Recuperado 4 de abril de 2022, de <https://minikube.sigs.k8s.io/docs/start/>
- Sense, D. (2020, 22 agosto). Deploying Any Dockerized Application To Kubernetes | Ashutosh Hathidara | #kubernetes [Video]. YouTube. <https://www.youtube.com/watch?v=XQNNaeyMAkk&feature=youtu.be>