



PLATAFORMAS COMPUTACIONALES A GRAN ESCALA

MINIPROYECTO CON KUBERNETES(MINIKUBE) - OPEN FAAS Y TUNNELING INLETS

PRESENTADO POR

Juan Camilo Sánchez Barreiro 1527749
Andres Mauricio Montenegro Vargas 1629950
Manuel Díaz 1741652

Docente:

John Alexander Sanabria

Universidad del Valle
Escuela de ingeniería de sistemas y computación
Santiago de Cali, Junio de 2020

El enunciado del proyecto está [aquí](#)

PD: Si de pronto no está disponible en github, está en la carpeta donde está este documento en el archivo comprimido en homework/README.md

Empezaremos diciendo que debemos tener instalado VirtualBox 6.1 y también Vagrant

Kubernetes en Virtual BOX (No funcionó)

Repositorio Github [aquí](#)

Tener cuidado con los requerimientos

Para instalar ansible se siguen estos pasos

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo apt-add-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Para instalar Python 3.7 estos

```
sudo apt update
sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
Luego presionamos Enter
sudo apt install python3.7
python3.7 --version
```

Kubernetes Minikube (Funciono)

Vamos a instalar Minikube que se encarga de montar kubernetes en nuestra máquina local

Instalar

```
$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64  
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

Arrancarlo

```
minikube start
```

Obtener los pods

```
kubectl get po -A
```

```
minikube kubectl -- get po -A
```

```
minikube dashboard
```

Para instalar Open Faas (esto se hace en la máquina nuestra fuera de la virtual machine)

Se instala el arkade

```
curl -SLsf https://dl.get-arkade.dev/ | sudo sh
```

Luego se instala OpenFaas:

```
sudo arkade install openfaas
```

Luego se instala faas-cli

```
curl -SLsf https://cli.openfaas.com | sudo sh
```

Forward the gateway to your machine

```
kubectl rollout status -n openfaas deploy/gateway
```

```
kubectl port-forward -n openfaas svc/gateway 8080:8080 &
```

If basic auth is enabled, you can now log into your gateway:

```
PASSWORD=$(kubectl get secret -n openfaas basic-auth -o jsonpath="{.data.basic-auth-password}" |  
base64 --decode; echo)
```

```
echo -n $PASSWORD | faas-cli login --username admin --password-stdin
```

Ahora se obtiene figlet de la store

```
faas-cli store deploy figlet
```

```
echo "Manuel Diaz" | faas-cli invoke figlet
```

Corriendo Inlets

Teniendo ya el Mini Kube corriendo lo que haremos a continuación es poner a correr el servidor inlets (Nos sirve como puente para que desde internet se pueda acceder a nuestro equipo corriendo open faas)

En caso de que necesitemos crear la máquina en google cloud podemos usar este repositorio dando click [aquí](#), en nuestro caso ya teníamos la máquina en google cloud creada por un ejercicio anterior.

Hacemos git clone <https://github.com/josanabr/tunneling-inlets>

Luego hacemos vagrant up

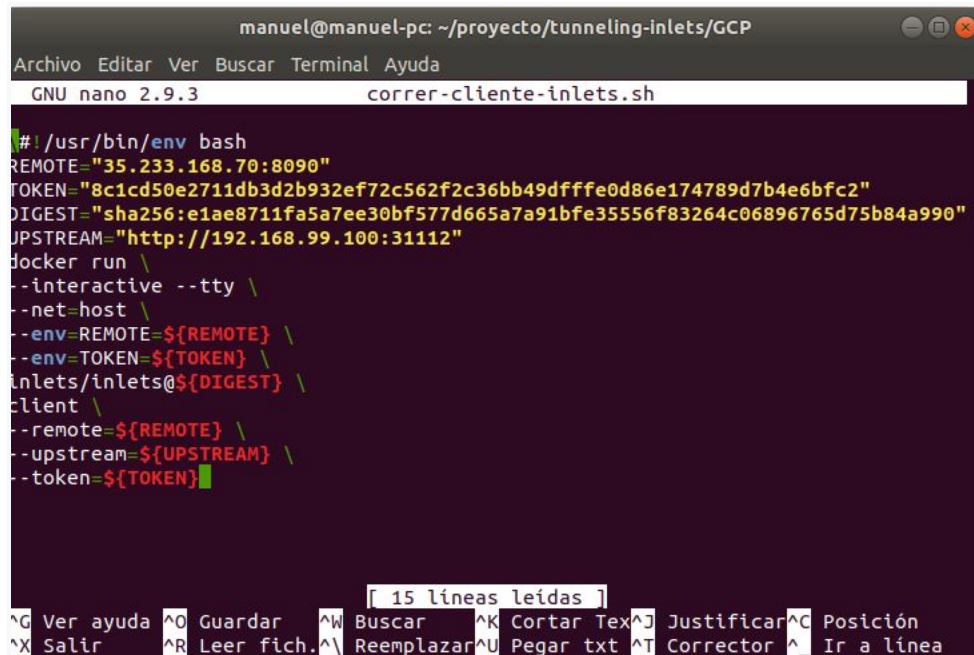
después hacemos vagrant ssh

Estando dentro de la máquina virtual de GCP vamos con el comando al dir

```
cd /vagrant/GCP
```

```
hacemos nano correr-cliente-inlets.sh
```

y modificamos las 3 primeras variables con los datos de la máquina que corre en google cloud



```
manuel@manuel-pc: ~/proyecto/tunneling-inlets/GCP
Archivo Editar Ver Buscar Terminal Ayuda
GNU nano 2.9.3      correr-cliente-inlets.sh

#!/usr/bin/env bash
REMOTE="35.233.168.70:8090"
TOKEN="8c1cd50e2711db3d2b932ef72c562f2c36bb49dffe0d86e174789d7b4e6bfc2"
DIGEST="sha256:e1ae8711fa5a7ee30bf577d665a7a91bfe35556f83264c06896765d75b84a990"
UPSTREAM="http://192.168.99.100:31112"
docker run \
--interactive --tty \
--net=host \
--env=REMOTE=${REMOTE} \
--env=TOKEN=${TOKEN} \
inlets/inlets@${DIGEST} \
client \
--remote=${REMOTE} \
--upstream=${UPSTREAM} \
--token=${TOKEN}
```

En este caso son:

REMOTE: 35.233.168.70:8090

TOKEN: 8c1cd50e2711db3d2b932ef72c562f2c36bb49dffe0d86e174789d7b4e6bfc2

DIGEST: sha256:e1ae8711fa5a7ee30bf577d665a7a91bfe35556f83264c06896765d75b84a990

UPSTREAM: 192.168.99.100:31112

El upstream lo sacamos con el comando minikube ip y le agregamos el puerto 31112

En remote no olvidar poner el puerto 8090

Luego corremos el archivo correr-cliente-inlets.sh con el comando

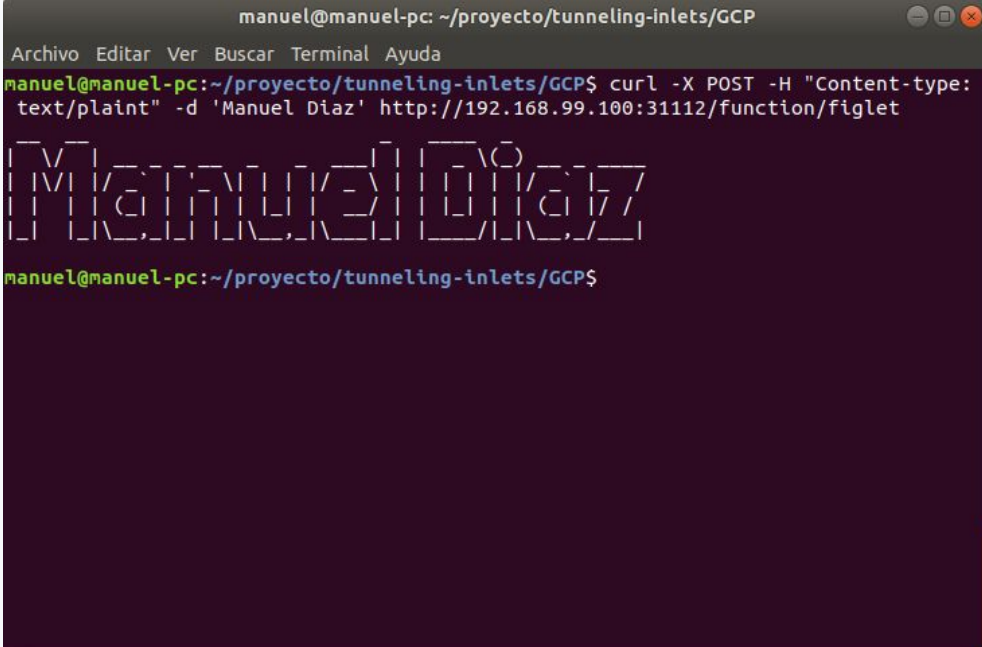
`./correr-cliente-inlets.sh`

En caso de que nos tire un error de permisos vamos a /vagrant/GCP y con el siguiente comando le damos todos los permisos

`chmod 777 correr-cliente-inlets.sh`

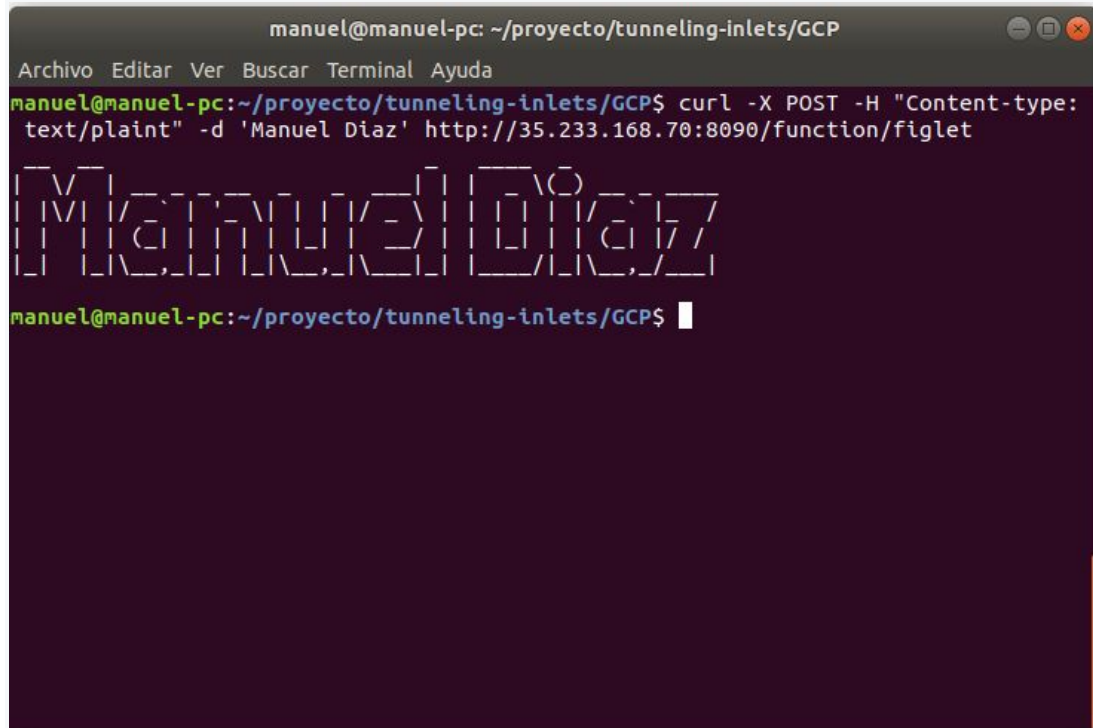
Con el siguiente comando lo corremos localmente

```
curl -X POST -H "Content-type: text/plain" -d 'Manuel Diaz' http://192.168.99.100:31112/function/figlet
```

A screenshot of a terminal window titled 'manuel@manuel-pc: ~/proyecto/tunneling-inlets/GCP'. The terminal shows a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The command prompt is 'manuel@manuel-pc:~/proyecto/tunneling-inlets/GCP\$'. The command entered is 'curl -X POST -H "Content-type: text/plain" -d 'Manuel Diaz' http://192.168.99.100:31112/function/figlet'. The output is a large ASCII art representation of the text 'Manuel Diaz' in a stylized, blocky font. The prompt returns to 'manuel@manuel-pc:~/proyecto/tunneling-inlets/GCP\$'.

Con el siguiente comando lo ponemos a correr a través de inlets, ósea a través de la maquina de google cloud

```
curl -X POST -H "Content-type: text/plain" -d 'Manuel Diaz' http://35.233.168.70:8090/function/figlet
```



```
manuel@manuel-pc: ~/proyecto/tunneling-inlets/GCP
Archivo Editar Ver Buscar Terminal Ayuda
manuel@manuel-pc:~/proyecto/tunneling-inlets/GCP$ curl -X POST -H "Content-type:
text/plain" -d 'Manuel Diaz' http://35.233.168.70:8090/function/figlet
Manuel Diaz
manuel@manuel-pc:~/proyecto/tunneling-inlets/GCP$
```

DESPLEGANDO LA FUNCIÓN CON PANDAS

Exportamos esta variable que usa Open Faas

```
export OPENFAAS_URL=$(minikube ip):31112
```

Estos pasos son para loguear en open faas y poder hacer uso de sus recursos

```
PASSWORD=$(kubectl get secret -n openfaas basic-auth -o jsonpath="{.data.basic-auth-password}" |
base64 --decode; echo)
```

```
echo $PASSWORD | faas-cli login --password-stdin
```

Ahora con estos comandos corremos nuestra función

```
sudo faas-cli up -f pyfunction.yml --skip-push
```

```
echo "something"|faas-cli invoke pyfunction
```

Después de verificar que está corriendo correctamente en nuestro kubernetes lo pondremos a correr a través del túnel con google cloud (Tunneling-inlets)

```
curl -X POST -H "Content-type: text/plain" -d 'Manuel Diaz'
http://35.233.168.70:8090/function/pyfunction
```

Esto sería lo que vamos a recibir como respuesta

```
manuel@manuel-pc:~/proyecto/pyFaaS$ curl -X POST -H "Content-type: text/plain" -d 'Manuel Diaz' http://35.233.168.70:8090/function/pyfunction
Descargando los datos
Casos de COVID-19 en el Valle del Cauca
Ciudad
Alcalá 1
Andalucía 2
Ansermanuevo 1
Argelia 11
Bugalagrande 1
Cali 4476
Calima 1
Candelaria 63
Cartago 17
Dagua 19
El Cairo 4
El Cerrito 25
El Dovio 4
Florida 35
Ginebra 1
Guacarí 5
Guadalajara de Buga 26
Jamundi 77
La Cumbre 3
La Unión 11
Obando 3
Palmira 100
Pradera 25
Restrepo 5
Roldanillo 1
San Pedro 3
Sevilla 4
Trujillo 3
Tuluá 38
Ulloa 1
Vijes 8
Yotoco 3
Yumbo 94
Name: Ciudad, dtype: int64
De los cuales en Cali son del tipo...
Tipo
En estudio 4029
Importado 78
Relacionado 369
Name: Tipo, dtype: int64
manuel@manuel-pc:~/proyecto/pyFaaS$
```

Videos:

- **Importante:** En este [video](#) podremos ver una explicación y una muestra de como correr lo expuesto en este documento
- El acceso al cluster de Kubernetes vía kubectl - [Video](#)
- La construcción, despliegue y publicación de la función que se usó en el proyecto vía faas-cli.
- La ejecución de la función desde Internet y al cluster de Kubernetes que se encuentra dentro de la red privada via faas-cli - [Video](#)