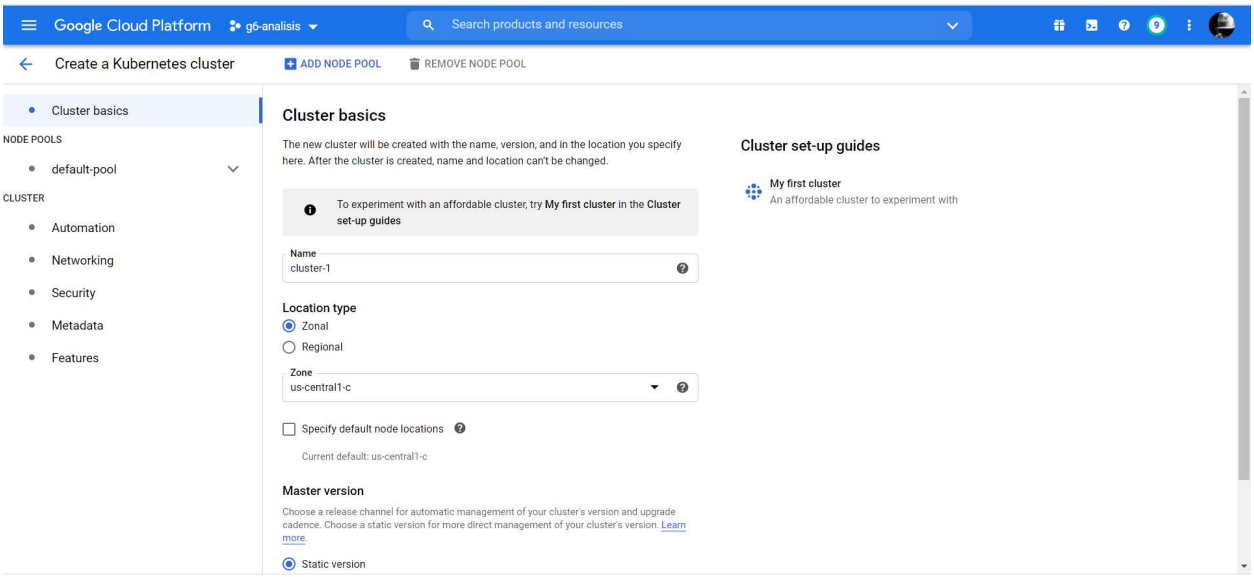
Examen Final

Documentation

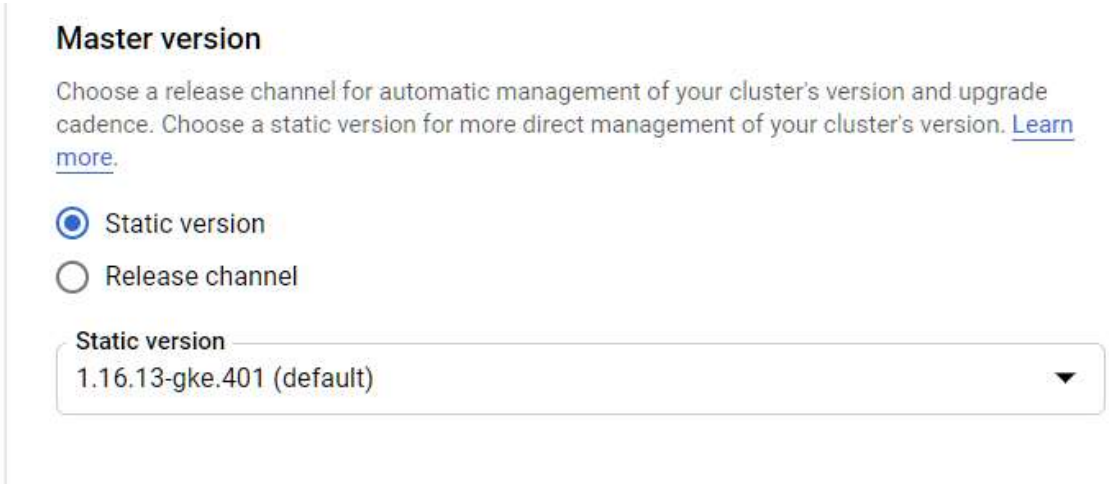
Herbert Rafael Reyes Portillo

201612114

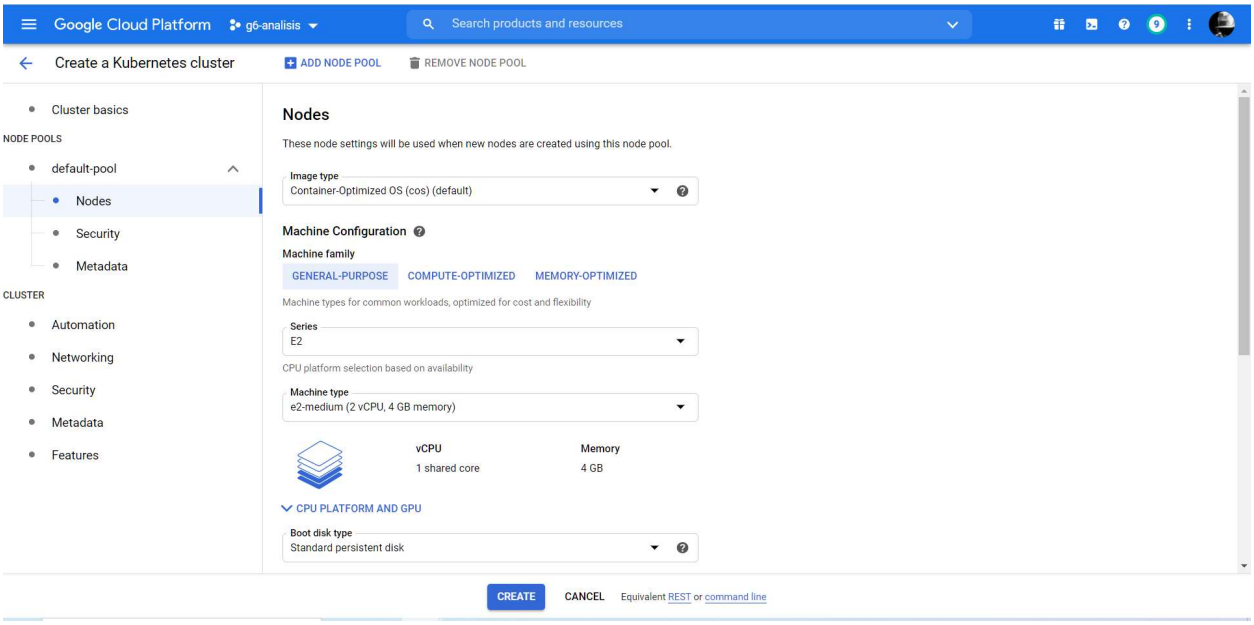
El primer paso es crear el cluster de kubernetes



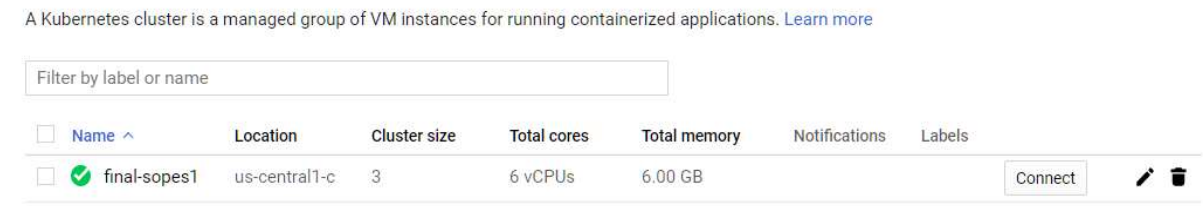
Definimos la versión que deseamos para nuestro cluster



Luego en default pool, seleccionamos el tamaño y cuantas instancias vamos a utilizar



Una vez creado el cluster, nos va a aparecer así



Una vez creado al cluster, por medio de power Shell ingresamos a el:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to g6-analisis.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
herbertreyes13@cloudshell:~ (g6-analisis)$ gcloud container clusters get-credentials final-sopes1 --zone us-central1-c --project g6-analisis
Fetching cluster endpoint and auth data.
```

Ahora el siguiente paso es configurar nuestra aplicación.

El primer paso es crear el Dockerfile tanto para el front end como para el back end

app-final > Dockerfile > ...

```
# pull official base image
FROM node:13.12.0-alpine
EXPOSE 3000
# set working directory
WORKDIR /app

# add `/app/node_modules/.bin` to $PATH
ENV PATH /app/node_modules/.bin:$PATH

# install app dependencies
COPY package.json ./
COPY package-lock.json ./
RUN npm install --silent
RUN npm install react-scripts@3.4.1 -g --silent

# add app
COPY . ./
# start app
CMD ["npm", "start"]
```

Dockerfile de front end

Back > Dockerfile > ...

```
FROM python
WORKDIR /app
ADD . .
RUN pip3 install --upgrade pip
RUN pip3 install Flask
RUN pip3 install flask_cors
RUN pip3 install requests
RUN pip3 install mysqlclient
RUN pip3 install mysql-connector-python
EXPOSE 4200
ENTRYPOINT ["python","index.py"]
```

Dockerfile backend

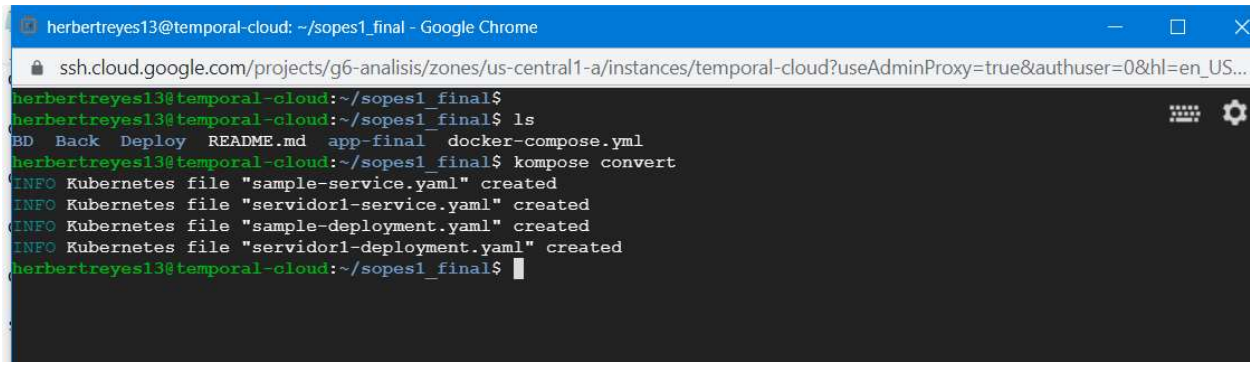
Luego procedemos a crear nuestro Docker-compose con la siguiente información

docker-compose.yml

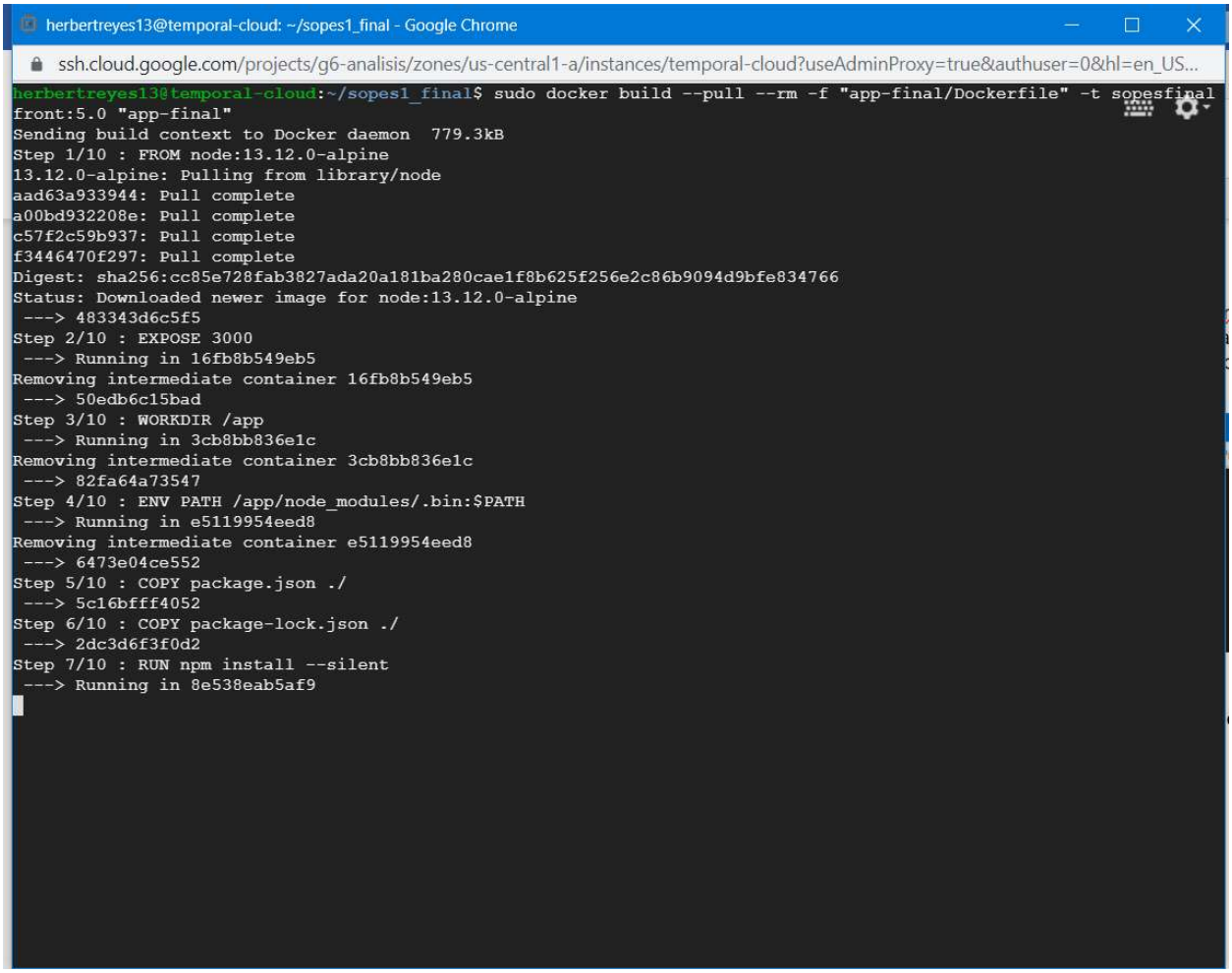
```yaml
version: '3'

services:
  sample:
    image: herbertreyes13j/finalfront:5.0
    build:
      context: app-final/.
      dockerfile: Dockerfile
    ports:
      - 3000:3000
    environment:
      - CHOKIDAR_USEPOLLING=true
    stdin_open: true
    labels:
      kompose.service.type: LoadBalancer
    deploy:
      replicas: 3

  servidor1:
    image: herbertreyes13j/finalback:5.0
    build:
      context: Back/.
      dockerfile: Dockerfile
    ports:
      - "4200:4200"
    network_mode: host
    labels:
      kompose.service.type: LoadBalancer
    deploy:
      replicas: 3

```

En donde básicamente le explicamos en que puertos vamos a correr todo, cuantas replicas deseamos, si trabajara como balanceador de carga.

El siguiente paso, es volver este Docker-compose a manera que Kubernetes pueda entenderlo, por lo que el siguiente paso consiste en convertir todos estos archivos al lenguaje de Kubernetes por medio de kompose. Únicamente debemos ingresar el comando “Kompose convert” y se generaran dichos archivos:



Ahora procedemos a crear nuestras imágenes de Docker y subirlas a Docker-hub. Lo primero es crear las imágenes

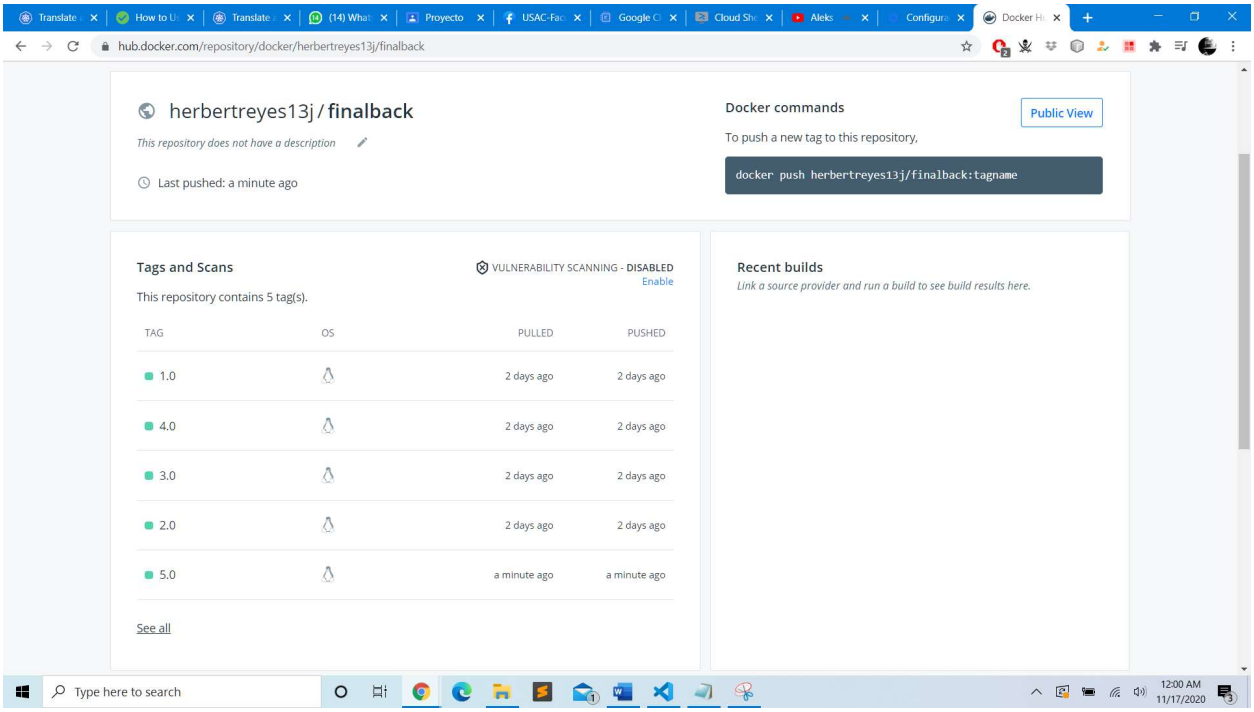


```
herbertreyes13@temporal-cloud:~$ cd sopes1_final
herbertreyes13@temporal-cloud:~/sopes1_final$ sudo docker build --pull --rm -f "Back/Dockerfile" -t sopesfinalback:
5.0 "Back"
Sending build context to Docker daemon   5.12kB
Step 1/11 : FROM python
latest: Pulling from library/python
e4c3d3e4f7b0: Extracting [==================>                            ]  22.02MB/50.4MB
101c41d0463b: Download complete
8275efcd805f: Download complete
751620502a7a: Download complete
0a5e725150a2: Downloading [========================================>      ]  165.7MB/192.3MB
397dba5694db: Download complete
b1d09d0eabcb: Download complete
475299e7c7f3: Download complete
ede3eeefc571: Download complete
```

Ahora procedemos a agregarles una etiqueta a nuestras imágenes para subirlas a la nube. El primer paso es colocarle la etiqueta con la que lo subiremos a Docker hub. Una vez tenga la etiqueta, hacemos un Docker push para que se suba a Docker hub.

```
herbertreyes13@temporal-cloud: ~/sopes1_final - Google Chrome
ssh.cloud.google.com/projects/g6-analisis/zones/us-central1-a/instances/temporal-cloud?useAdminProxy=true&authuser=0&hl=en_US&projectNumber=857789487621
herbertreyes13@temporal-cloud:~/sopes1_final$ docker images
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/images/json: dial unix /var/run/docker.sock: c
onnect: permission denied
herbertreyes13@temporal-cloud:~/sopes1_final$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
sopesfinalback      5.0                 8fa485854f05        49 seconds ago      905MB
<none>              <none>              2dc3d6f3f0d2        3 minutes ago       115MB
python              latest              768307cdb962        13 days ago         886MB
node                13.12.0-alpine      483343d6c5f5        7 months ago        114MB
herbertreyes13@temporal-cloud:~/sopes1_final$ docker tag 8fa485854f05 herbertreyes13j/finalback:5.0
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.40/images/8fa485854f05/tag?repo=herbertreyes13j%
2Ffinalback&tag=5.0: dial unix /var/run/docker.sock: connect: permission denied
herbertreyes13@temporal-cloud:~/sopes1_final$ sudo docker tag 8fa485854f05 herbertreyes13j/finalback:5.0
herbertreyes13@temporal-cloud:~/sopes1_final$ sudo docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
herbertreyes13j/finalback   5.0                 8fa485854f05        About a minute ago   905MB
sopesfinalback              5.0                 8fa485854f05        About a minute ago   905MB
<none>                      <none>              2dc3d6f3f0d2        4 minutes ago        115MB
python                      latest              768307cdb962        13 days ago          886MB
node                        13.12.0-alpine      483343d6c5f5        7 months ago         114MB
herbertreyes13@temporal-cloud:~/sopes1_final$ sudo docker push herbertreyes13j/finalback
The push refers to repository [docker.io/herbertreyes13j/finalback]
0c48d6dfdd9a: Pushed
78197a17773f: Pushed
3a29afe171cc: Pushed
247d049e40f9: Pushed
201864cc179d: Pushed
dc99b07bd3a3: Pushed
ebc0df41118c: Pushed
8f2a3ef28f14: Pushed
641e75738194: Layer already exists
e4fc3bd6e3cb: Layer already exists
5f03d9827383: Layer already exists
6f7043721c9b: Layer already exists
a933681cf349: Layer already exists
f49d20b92dc8: Layer already exists
fe342cfe5c83: Layer already exists
630e4f1da707: Layer already exists
9780f6d83e45: Layer already exists
5.0: digest: sha256:cd94fceea60d4e150a2a9a1b18e51fb8b793546953caad63fb4d1b425781ae09 size: 3896
herbertreyes13@temporal-cloud:~/sopes1_final$
```

Y tendremos nuestra imagen subida a dockerhub, con la versión que le hemos puesto



Finalmente procedemos a kubernetes, a hacer deploy a los archivos que generamos anteriormente

```
herbertreyes13@cloudshell:~/sopes1_final (g6-analisis)$ kubectl apply -f Deploy
deployment.apps/sample configured
service/sample configured
deployment.apps/servidor1 configured
service/servidor1 configured
herbertreyes13@cloudshell:~/sopes1_final (g6-analisis)$
```

Y finalmente podremos ver los pods

```
herbertreyes13@cloudshell:~/sopes1_final (g6-analisis)$ kubectl get deployment,svc,pods
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sample        3/3     1            3           2d3h
deployment.apps/servidor1     3/3     3            3           2d3h

NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
service/kubernetes    ClusterIP      10.4.0.1       <none>           443/TCP          2d5h
service/sample        LoadBalancer   10.4.10.60     34.123.155.182   3000:30070/TCP   2d3h
service/servidor1     LoadBalancer   10.4.10.178    35.202.58.137    4200:30002/TCP   2d3h

NAME                                 READY   STATUS              RESTARTS   AGE
pod/sample-6c677d76f4-9mpkh          0/1     ContainerCreating   0          29s
pod/sample-78bdddc9f-2v8mc           1/1     Running             0          47h
pod/sample-78bdddc9f-7h9qx           1/1     Running             0          47h
pod/sample-78bdddc9f-nw28d           1/1     Running             0          47h
pod/servidor1-7966cbc6b4-g7cpr       1/1     Running             0          29s
pod/servidor1-7966cbc6b4-mxzhh       1/1     Running             0          18s
pod/servidor1-7966cbc6b4-pg2c6       1/1     Running             0          23s
pod/servidor1-d5f6959-8rgzt          1/1     Terminating         0          29s
herbertreyes13@cloudshell:~/sopes1_final (g6-analisis)$
```

Vemos como se eliminan ciertos pods y otros se han ido creando

```
herbertreyes13@cloudshell:~/sopes1_final (g6-analisis)$ kubectl get deployment,svc,pods
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sample        3/3     3            3           2d3h
deployment.apps/servidor1     3/3     3            3           2d3h

NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
service/kubernetes    ClusterIP      10.4.0.1       <none>           443/TCP          2d5h
service/sample        LoadBalancer   10.4.10.60     34.123.155.182   3000:30070/TCP   2d3h
service/servidor1     LoadBalancer   10.4.10.178    35.202.58.137    4200:30002/TCP   2d3h

NAME                               READY   STATUS        RESTARTS   AGE
pod/sample-6c677d76f4-968fp        1/1     Running       0          72s
pod/sample-6c677d76f4-9mpkh        1/1     Running       0          111s
pod/sample-6c677d76f4-ds7qr        1/1     Running       0          37s
pod/sample-78bdddc9f-2v8mc         0/1     Terminating   0          47h
pod/servidor1-7966cbc6b4-g7cpr     1/1     Running       0          111s
pod/servidor1-7966cbc6b4-mxzhh     1/1     Running       0          100s
pod/servidor1-7966cbc6b4-pg2c6     1/1     Running       0          105s
herbertreyes13@cloudshell:~/sopes1_final (g6-analisis)$
```

Y finalmente veremos nuestra aplicación creada