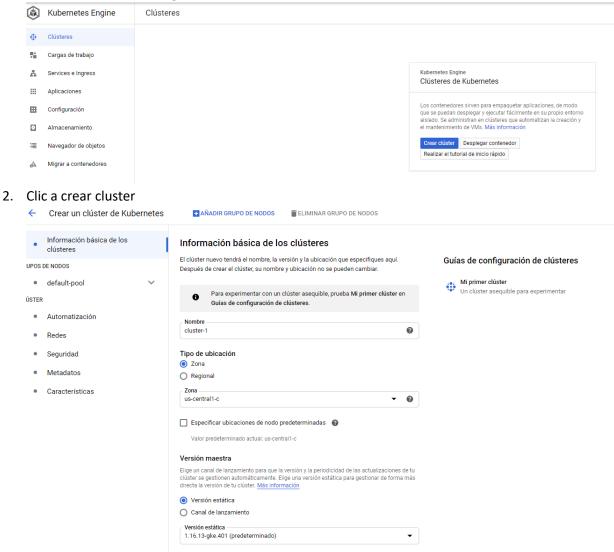
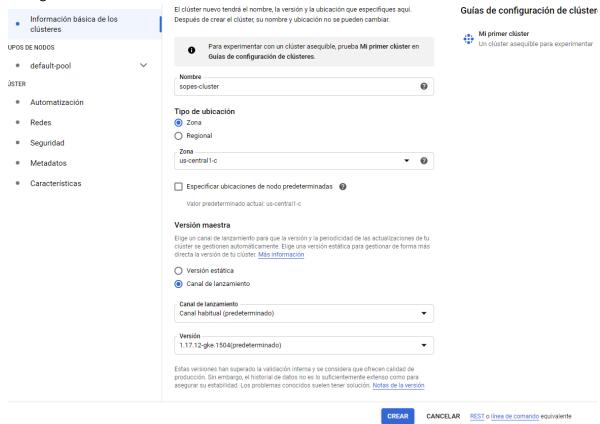
INSTALACION CLUSTER Google Cloud

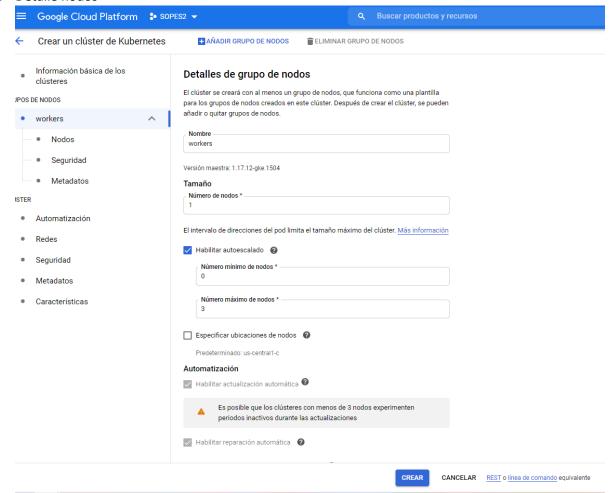
1. Entramos a Kubernet Engine

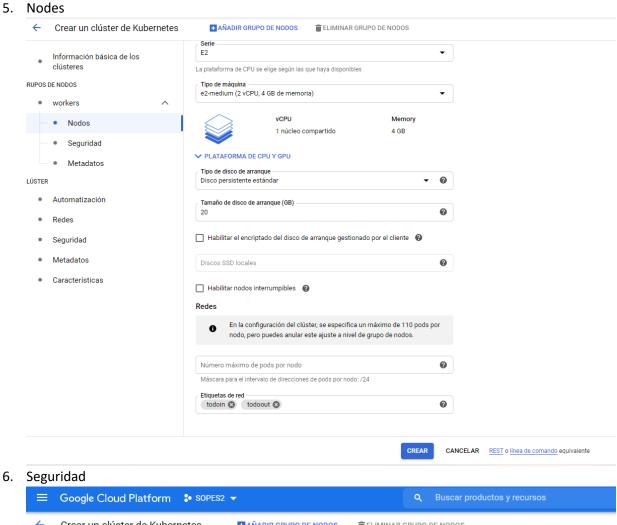


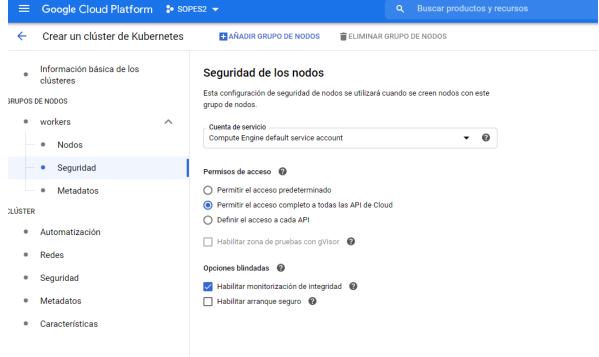
3. Configuración basica



4. Detalle nodos







CREAR MAQUINA VIRTUAL

- 7. INSTALAR KUBECTL
 - a. wget curl -LO https://storage.googleapis.com/kubernetes-release/v1.18.0/bin/linux/amd64/kubectl
 - b. chmod +x ./kubectl
 - c. sudo mv ./kubectl /usr/local/bin/kubectl
 - i. REF: https://kubernetes.io/docs/tasks/tools/install-kubectl/
- 8. CREAR Y CONFIGURAR CLUSTER EN GOOGLE
 - a. curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-297.0.0-linux-x86_64.tar.gz
 - b. tar zxvf [ARCHIVE_FILE] google-cloud-sdk
 - i. tar zxvf google-cloud-sdk-297.0.0-linux-x86 64.tar.gz
 - c. ./google-cloud-sdk/install.sh
 - d. logout and login to reload the new gcloud command (Verificamos si funciona el commando gcloud)

e. gcloud init<<Seguir instrucciones de pantalla>>

```
hoose the account you would like to use to perform operations for
his configuration:
[1] 1029710161861-compute@developer.gserviceaccount.com
[2] Log in with a new account
lease enter your numeric choice: 2
ou are running on a Google Compute Engine virtual machine.
t is recommended that you use service accounts for authentication.
ou can run:
 $ gcloud config set account `ACCOUNT`
o switch accounts if necessary.
our credentials may be visible to others with access to this
rirtual machine. Are you sure you want to authenticate with
our personal account?
o you want to continue (Y/n)? y
o to the following link in your browser:
   https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.c
scope=openid+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2F
loud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%
Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth%redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3A
ob&code_challenge=1v297kUb2jzu2jpG06SZRIQoDcdrcEAmWHt83oT6UG0&access_type=offline&prompt=select_account&code_cha
nter verification code:
```

- i. Elegimos loguearnos con una cuenta nueva
- ii. Le damos aceptar y nos tirará un enlace, el cual entramos, con la cuenta con la que tenemos creadas nuestras maquinas de gcloud
- iii. Después de loguearnos nos dará un código, el cual copiamos y pegamos en donde nos indica.



```
Enter verification code: 4/1AY0e-g7Yt1jMrPHWBcMMUwS79xSHRISLygW9UkiHJ13bJLZ1Bf0GiqgN4zc
You are logged in as: [2618007790101@ingenieria.usac.edu.gt].

Pick cloud project to use:
[1] so1t3-280619
[2] sopes2-295218
[3] Create a new project

Please enter numeric choice or text value (must exactly match list
item): [
```

Ahora estamos logueados y listos para continuar.

```
Enter verification code: 4/1AY0e-g7Yt1jMrPHWBcMMUwS79xSHRISLygW9UkiHJl3bJLZlBf

You are logged in as: [2618007790101@ingenieria.usac.edu.gt].

Pick cloud project to use:
[1] solt3-280619
[2] sopes2-295218
[3] Create a new project

Please enter numeric choice or text value (must exactly match list

item): 2

Your current project has been set to: [sopes2-295218].

Do you want to configure a default Compute Region and Zone? (Y/n)? y
```

Elegimos el proyecto, en este caso es el proyecto 2 y configuramos la región

En este caso escogemos la misma región en la que creamos el cluster

us-central1-c

y finalizamos con la instalación

```
od not print [24] options.
Too many options [74]. Enter "list" at prompt to print choices fully.
Please enter numeric choice or text value (must exactly match list
our project default Compute Engine zone has been set to [us-central1-c].
ou can change it by running [gcloud config set compute/zone NAME].
our project default Compute Engine region has been set to [us-centrall].
Ou can change it by running [gcloud config set compute/region NAME].
reated a default .boto configuration file at [/home/g2618007790101/.boto]. See this file and
https://cloud.google.com/storage/docs/gsutil/commands/config] for more
nformation about configuring Google Cloud Storage.
Our Google Cloud SDK is configured and ready to use!
 Commands that require authentication will use 2618007790101@ingenieria.usac.edu.gt by default
 Commands will reference project 'sopes2-295218' by default
Compute Engine commands will use region 'us-central1' by default
 Compute Engine commands will use zone `us-central1-c` by default
un `gcloud help config` to learn how to change individual settings
his gcloud configuration is called [default]. You can create additional configurations if you work with multiple
accounts and/or projects.
dun 'gcloud topic configurations' to learn more.
Some things to try next:
 Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help COMMAND` to
et help on any gcloud command.
 Run `gcloud topic --help` to learn about advanced features of the SDK like arg files and output formatting
```

iv. referencia: https://cloud.google.com/sdk/docs/quickstart-linux

CONFIGURAR KUBECTI PARA ACCEDER AL CLUSTER

 f. Vamos al cluster creado anteriormente, entramos y en la opción que dcie Endpoint le damos clic a –ver certificado— Clusters->MyCluster->Details->Show Cluster Certificate



- g. Y le damos clic a la opción de recuperar credenciales para kubectl
- h. Esto nos abre una pagina en la cual están las instrucciones a seguir

```
i. gcloud container clusters get-credentials NAME [--
internal-ip] [--region=REGION | --zone=ZONE, -
z ZONE] [GCLOUD WIDE FLAG ...]
```

 i. gcloud container clusters get-credentials cluster-1 --zone=uscentral1-c

```
[2618007790101@backend:~$ gcloud container clusters get-credentials cluster-1 --zone=us-central1-c
Petching cluster endpoint and auth data.
Tubeconfig entry generated for cluster-1.
Tubec
```

- ii. Este comando nos crea el archivo de configuración de kubectl
- iii. Por ultimo probamos a ver si podemos acceder al cluster con el comando: kubectl get nodes

CREAR PODS

CREAR NAMESPACE

Kubectl create ns proyecto

Instalamos Helm

wget https://get.helm.sh/helm-v3.4.0-linux-amd64.tar.gz

tar -xzvf helm-v3.4.0-linux-amd64.tar.gz

sudo mv linux-amd64/helm /sbin

agregar repositorio generic de Helm

helm repo add stable https://charts.helm.sh/stable

busca los paquetes

helm search repo stable

Buscamos el paquete ngnix, el cual nos va a servir para publicar nuestra app ya sea backend o frontend y lo instalamos en nuestro namespace creado anteriormente

helm install nginx-ingress stable/nginx-ingress -n proyecto

para desinstalarlo solo ejecutamos el siguiente comando

helm uninstall nginx-ingress -n nginx-ingress

helm list -n proyecto

Verificamos la instalación con los pods y los servicios

kubectl get pods -n proyecto

```
0101@backend:~$ kubectl get pods -n proyecto
                                                   READY
NAME.
                                                            STATUS
                                                                      RESTARTS
                                                                                  AGE
nginx-ingress-controller-85fcf4d47c-kqfck
                                                   1/1
                                                            Running
                                                                      0
                                                                                  64s
nginx-ingress-default-backend-5b967cf596-h4jlh
                                                   1/1
                                                            Running
                                                                       0
                                                                                  64s
```

kubectl get services -n proyecto

La ip externa del balanceador de carga es el que nos va a servir para publicar nuestra app

CREAR IMAGE DOCKER DE NUESTRAS APPS

Descargamos nuestra app desde github a nuestra maquina virtual Instalamos Docker

Sudo apt install docker.io

Entramos a la carpeta del proyecto que gueremos publicar y ejecutamos sudo sh build.sh

Loguearse como sudo, para evitar escribir siempre sudo antes de cada comando Nos logueamos a Docker Docker login

push imagen a docker

```
sudo docker push wildering/backend-image-so2
root@backend:/home/g2618007790101/proyecto_grupo31_so2/backend# sudo docker push wildering/backend-image-so2
The push refers to repository [docker.io/wildering/backend-image-so2]
:18e83b5c5dd: Pushed
6253f630fcb6: Pushed
ffb5f425c49: Pushed
daf513b2fba: Pushed
19571ff21ce3: Mounted from library/node
679834dd79ff: Mounted from library/node
0b6541f04a0d: Mounted from library/node
3211c12c1c23: Mounted from library/node
ld3ec06e3d4f: Mounted from library/node
e5330403dba: Mounted from library/node
Bbd20dc0b7e5: Mounted from library/node
94b70b410c2a: Mounted from library/node
3567db1eb737: Mounted from library/node
 atest: digest: sha256:1defdcc8f65d6997760d08750326de476d3b23b95d60a222baf5ea3914aece18 size: 3051
coot@backend:/home/g2618007790101/proyecto_grupo31_so2/backend#
```

Publicar en Kubernetes

Creamos del deployment –Nota: este deployment se crea con archivo yaml ya que se crean tres replicas, si no se desean crear replicas puede hacerse un deployment con el siguiente comando: kubectl create deployment miapp-python-n --image=wildering/python-nats -n proyecto obteniendo la imagen de Docker creada anteriormente.

kubectl create -f ingress-backend.yaml

Crear el service

kubectl expose deployment backend-deploy --name=miapp-backend-gnix-srv --port=3000 --targetport=3000 --type=LoadBalancer -n proyecto

verificamos que se haya creado exitosamente tanto el servicio como el proyecto

kubectl get pods -n proyecto

kubectl get services -n proyecto

Crear el Ingress ya sea para nginx o para Contour Una vez creado lo ejecutamos

kubectl create -f ingress-nginx.yaml

enlazamos la ip de nuestro loadbalancer con nuestro dns creado anteriormente.

Creamos el deploy del front end

kubectl create -f deploy-frontend.yaml

 $kubectl\ expose\ deployment\ frontend-deploy\ --name=miapp-frontend-gnix-srv\ --port=4000\ --target-port=4000\ --type=LoadBalancer\ -n\ proyecto$

Creamos el ingress

kubectl create -f ingress-nginx.yaml

BACKEND

http://backend.proy2so2.tk

FRONTEND

http://frontend.proy2so2.tk

http://34.122.182.4:4000/all