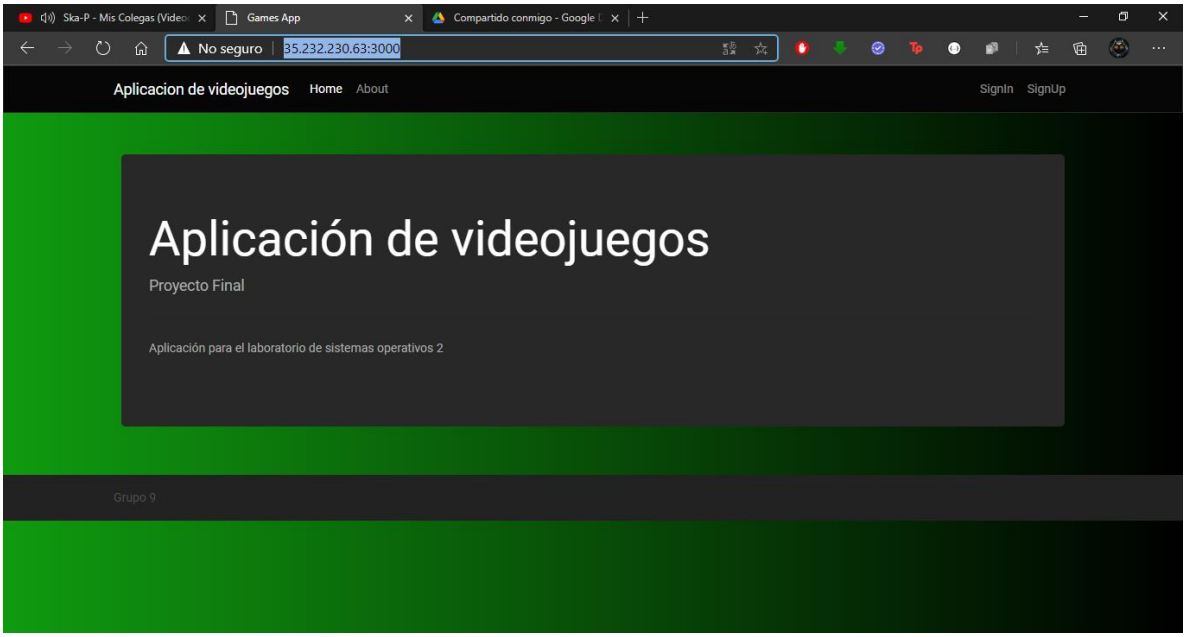
Universidad de San Carlos de Guatemala
Facultad de ingeniería
Escuela de vacaciones diciembre 2020
Laboratorio de sistemas operativos 2

Robinson Jonathan Pérez Fuentes **201114056**
Luis Ricardo Hernández **201114490**

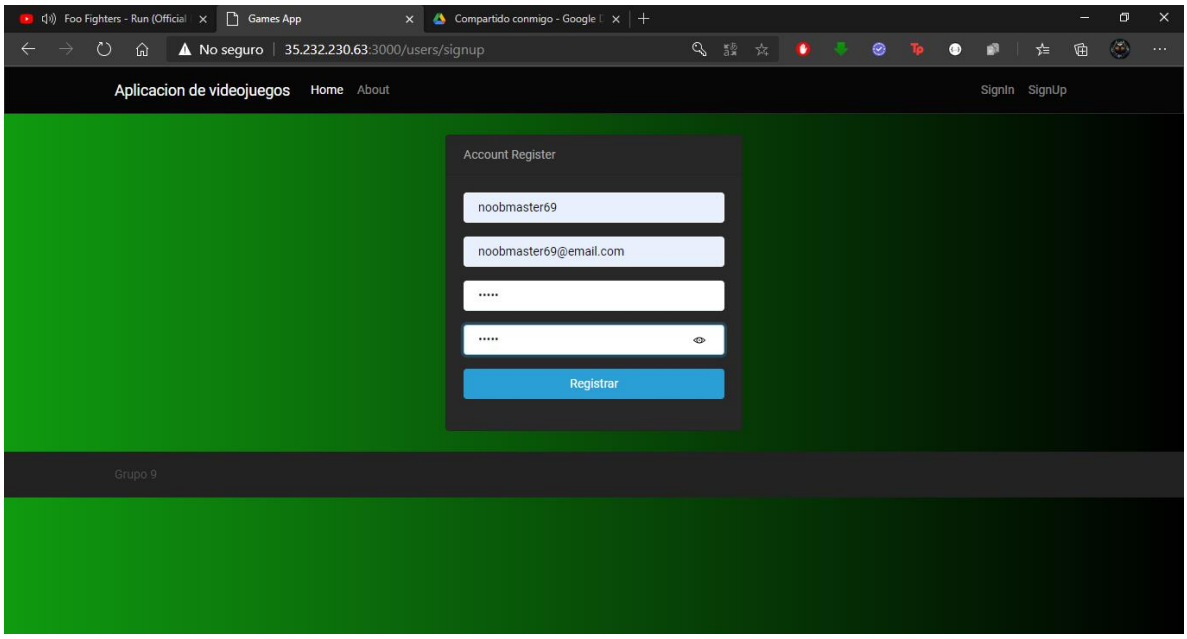
Manual proyecto final

Uso de la aplicación

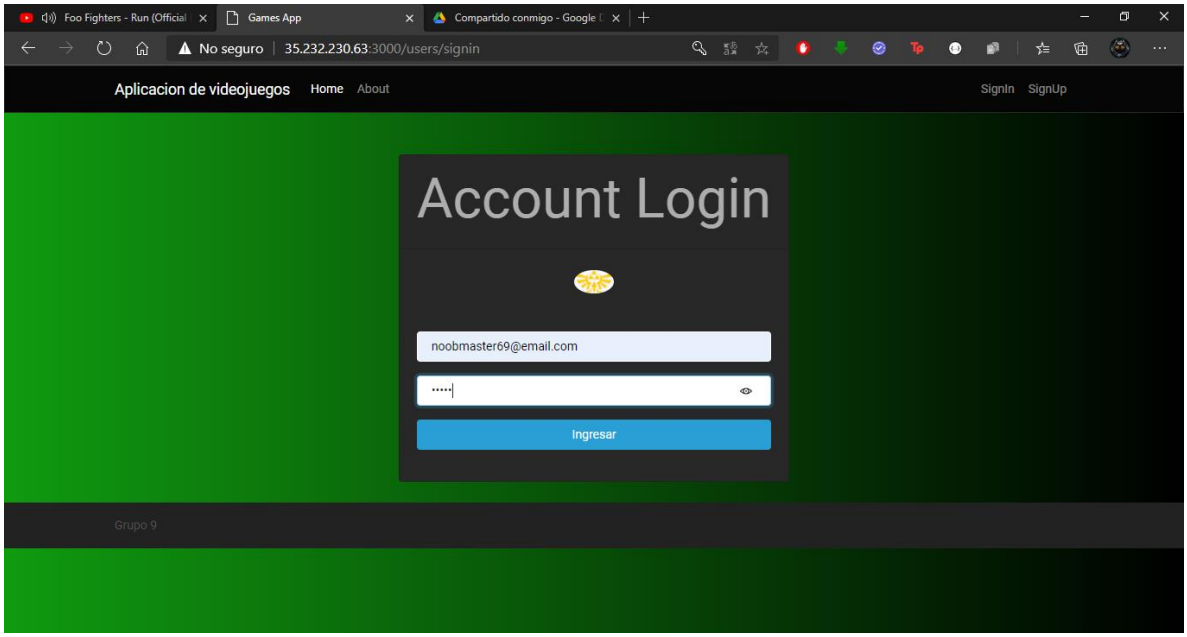
1. Para poder utilizar la aplicación debemos dirigirnos a la siguiente dirección http://35.232.230.63:3000
   acá se nos despliega la pagina de inicio del sistema



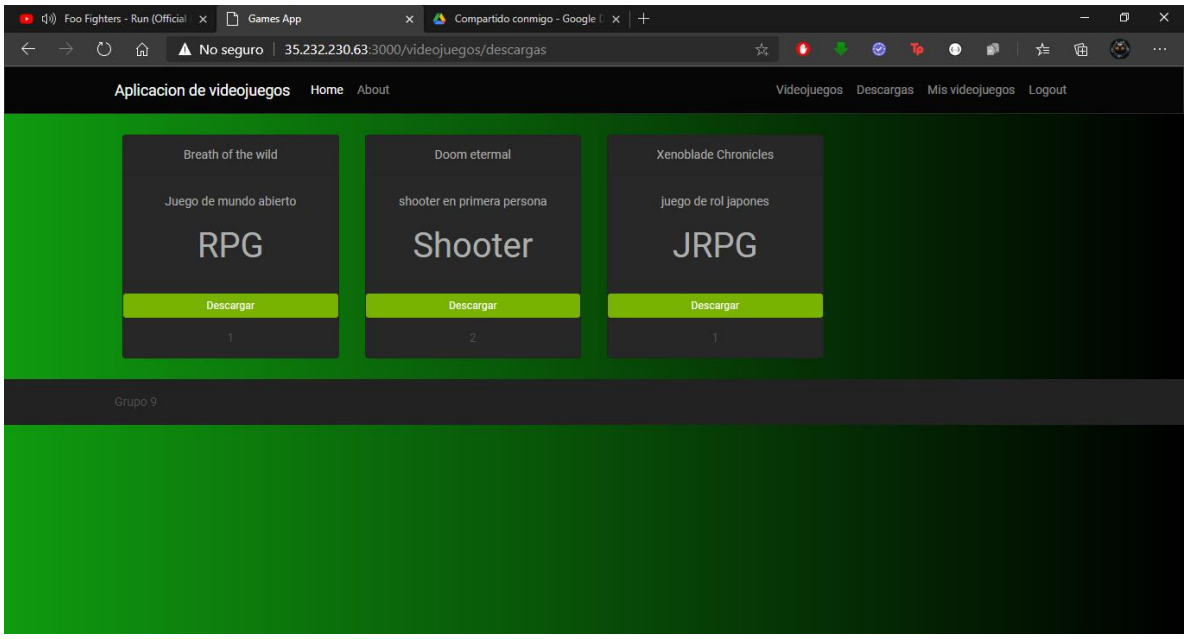
2. Lo primero que tendremos que hacer es crearnos una cuenta para poder descargar videojuegos, para ello debemos dirigirnos a “signUp” e ingresar los datos que se nos solicitan para crearnos un usuario



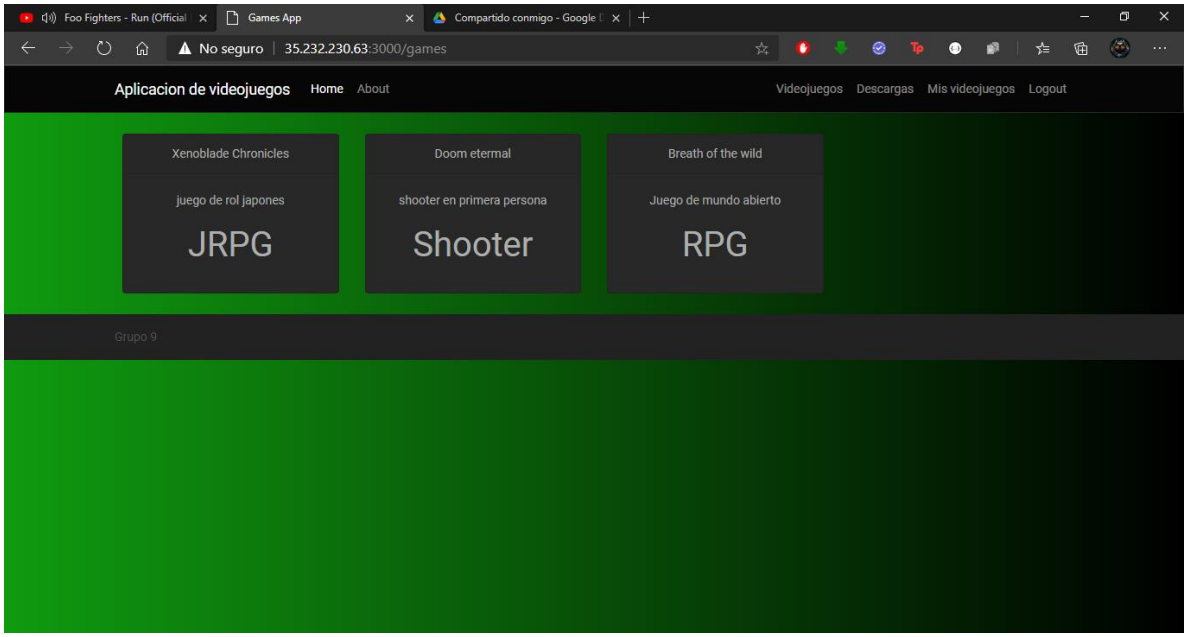
3. Si los datos ingresados son correctos, automáticamente se nos dirige a la página de “SignIn”, donde debemos colocar el correo y contraseña registrado anteriormente y darle al botón de “Ingresar”.



4. Una vez dentro, para poder descargar un videojuego debemos dirigirnos a la sección descargas, donde se nos muestran los juegos que podemos descargar y el numero de descargas de estos.



5. Si deseamos descargar uno de los juegos de la lista, solo debemos darle clic al botón Descargar, luego podremos ver los juegos que hemos descargado en la sección “Mis videojuegos”



6. Por último, si deseamos cerrar nuestra sesión solo debemos presionar en Logout, esto nos lleva de regreso a la pagina de inicio y nos permite loggearnos nuevamente con otro usuario
7. Adicional la pagina cuenta con un modulo administrativo que permite, agregar o eliminar videojuegos

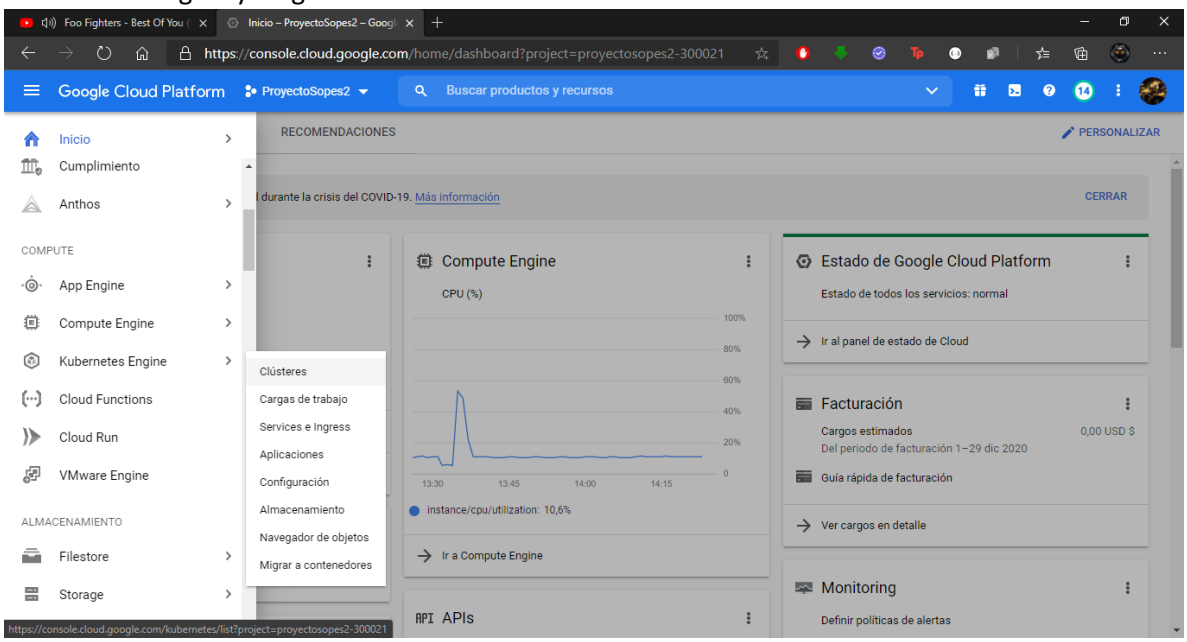
Manual técnico

Desarrollo de la infraestructura y despliegue de la aplicación

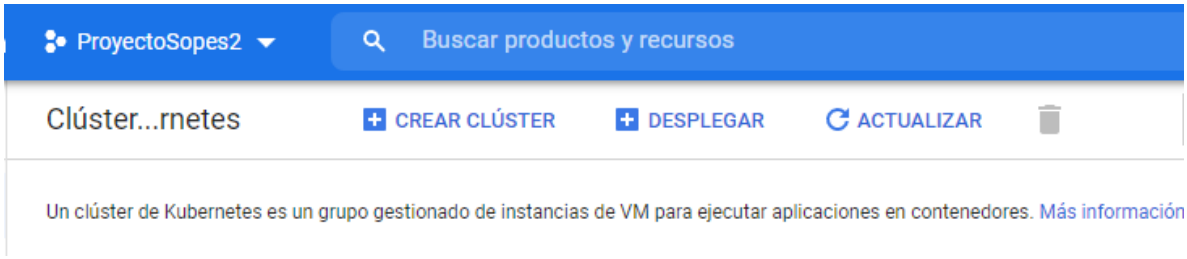
Creación del clúster

Para poder realizar el despliegue de la aplicación por medio de kubernetes y contenedores, es necesario crear primero un cluster, en este caso lo hemos realizado en gcloud a continuación, se describen los pasos:

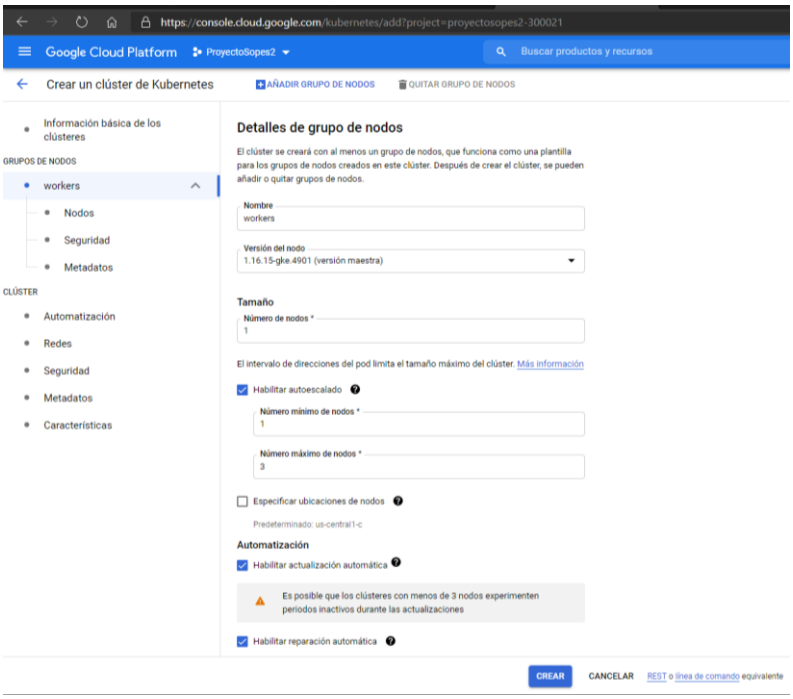
1. Debemos crear una nueva instancia desde nuestra cuenta de gcloud, nos dirigimos a Kubernetes engine y elegimos Clusteres.



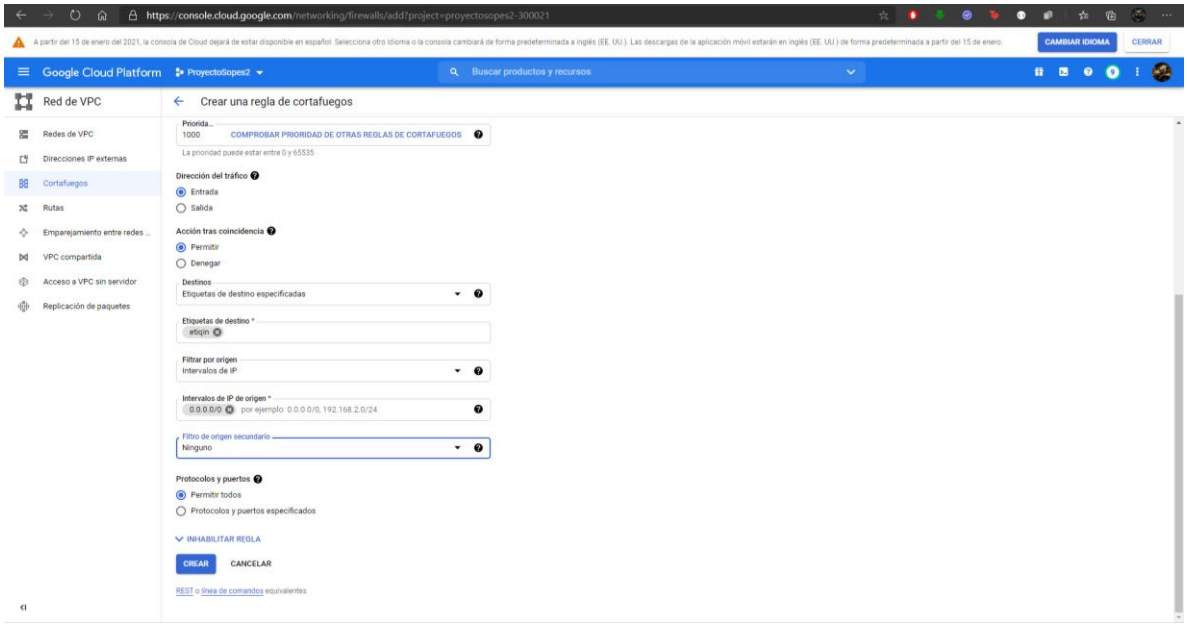
2. Le damos a la opción “Crear clustér” y de esta forma ya nos aparecerán las distintas opciones que deseamos para nuestro cluster

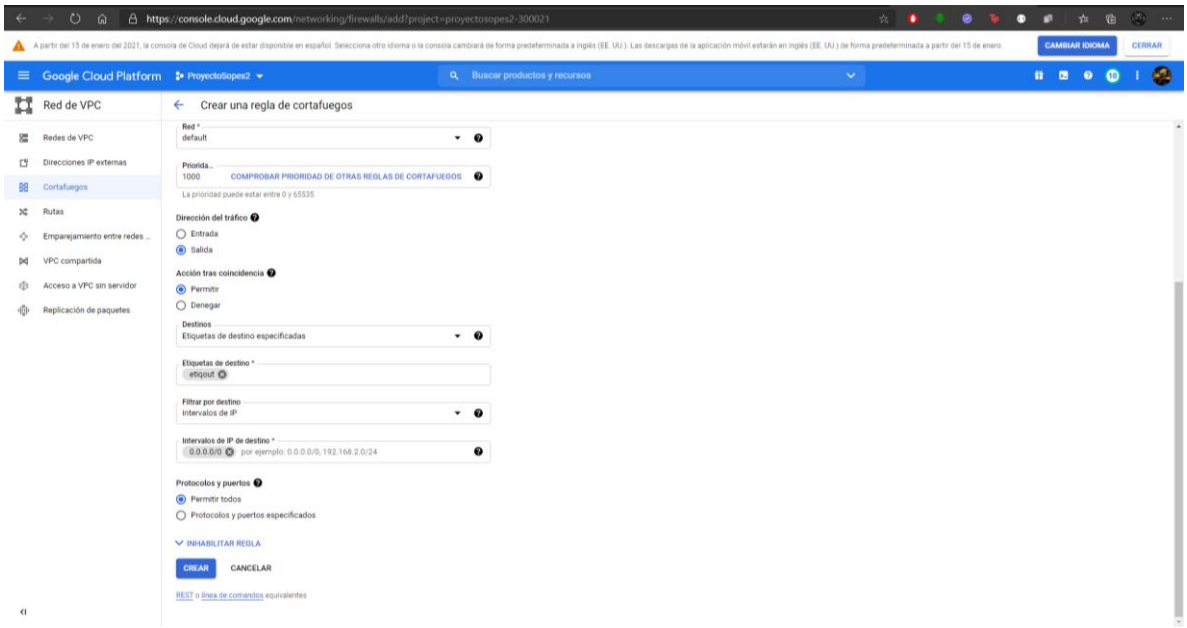


3. En la información básica solo colocaremos un nombre deseado al cluster, el resto de campos los dejamos por defecto
4. Nos dirigimos a la sección de Grupos de nodos, y colocaremos los siguientes datos:
    a. Colocamos un nombre deseado
    b. Numero de nodos lo colocamos a uno
    c. Habilitamos el auto escalado
    d. Colocamos el mínimo y el máximo de nodos

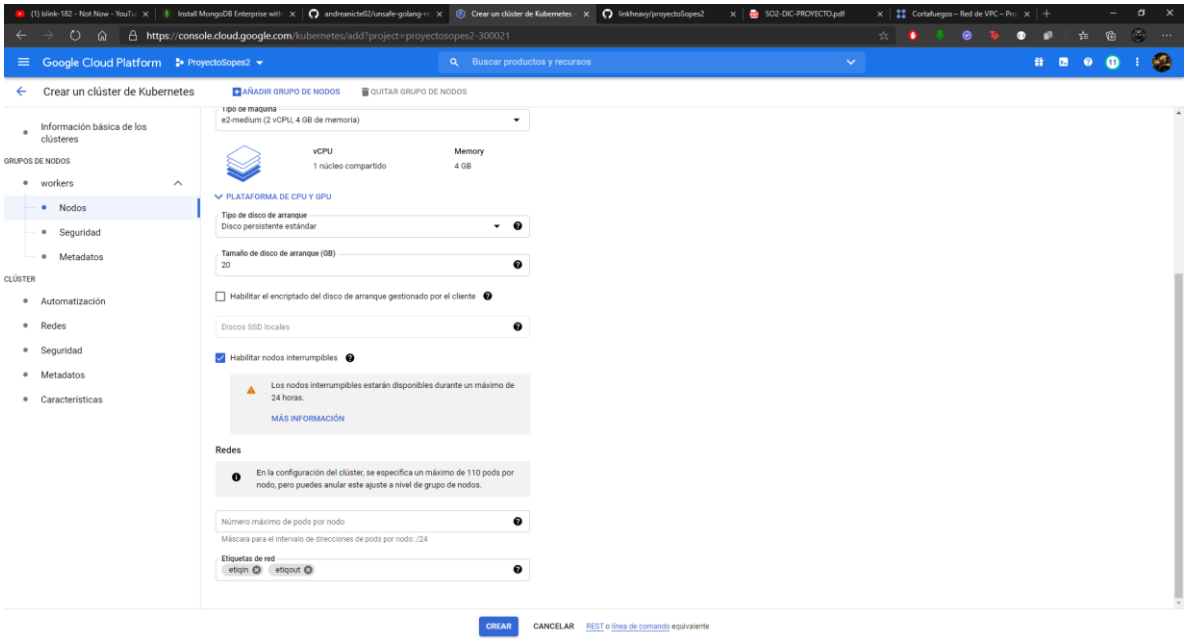


5. Antes de continuar vamos a configurar un firewall que permitirá la entrada y salida del trafico

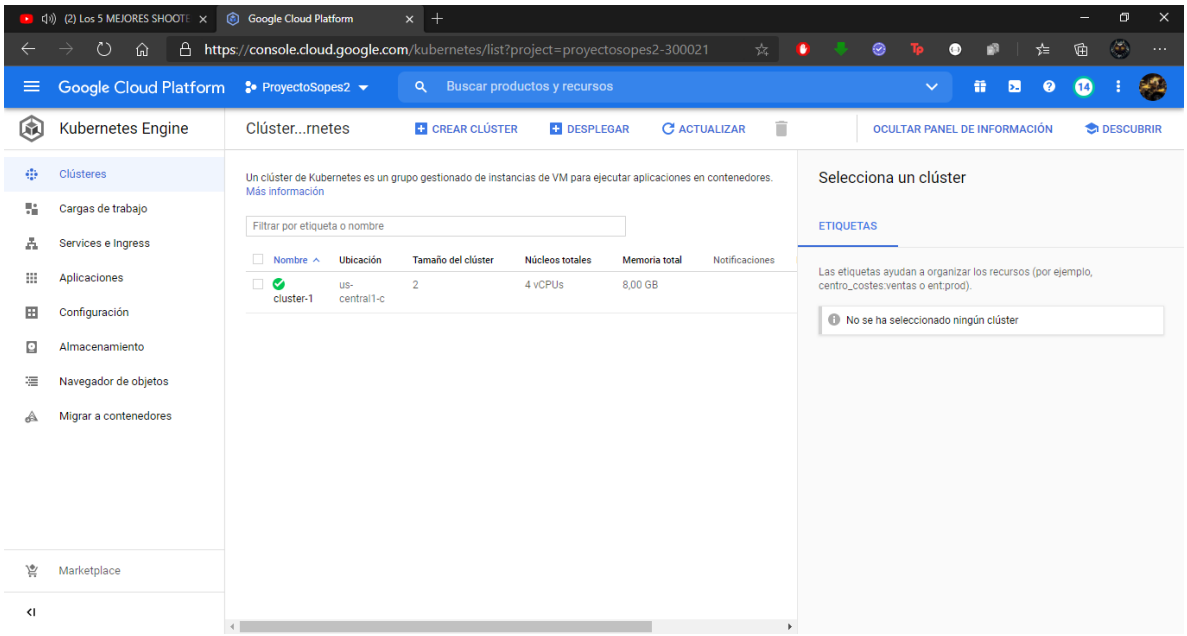




6. Continuando con el proceso nos dirigimos al área de nodos al cual colocaremos las etiquetas de entradas y salidas creadas anteriormente con firewall, dejaremos la imagen por defecto y colocamos un tamaño deseado al disco, en este proyecto solo se coloco 20 GB por temas de costos



7. Realizado los pasos anteriores, le damos al botón crear y esperamos a que la instancia sea creada e iniciada, para poder empezar a interactuar con ella



Tecnologías de desarrollo

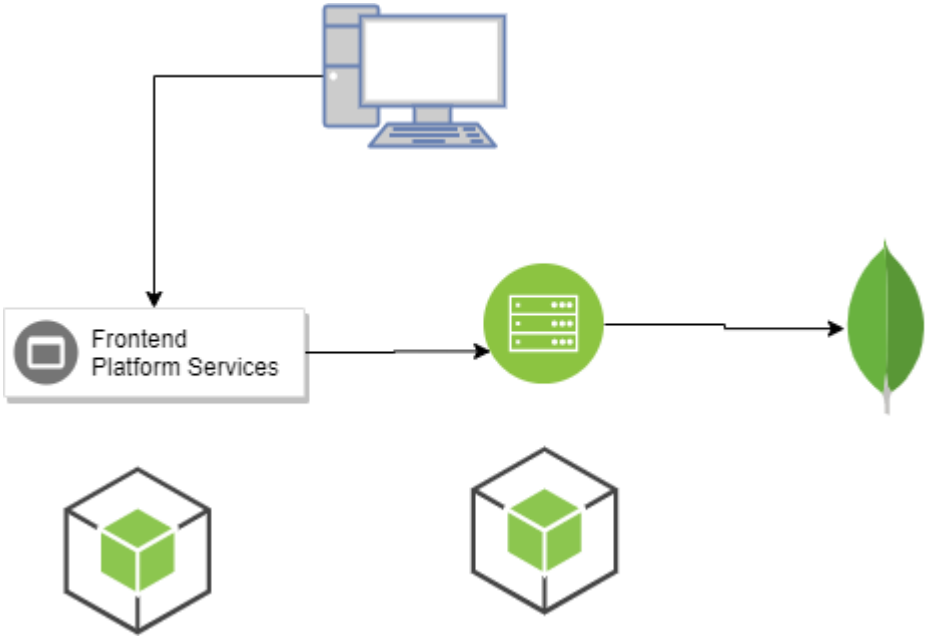
Para el desarrollo de la aplicación, se utilizaron las siguientes tecnologías

- NodeJs
- Express
- Handlebars
- MongoDB

Los mencionados anteriormente son las tecnologías que se utilizaron principalmente para desarrollar la aplicación desarrollada, tanto para el frontend como el backend se utiliza un servidor de nodejs, en el caso del frontend las vistas se desarrollan por medio de handlebars.

Para la base de datos se eligió una no relacional, por sus facilidades de comunicación y manejo de los datos solicitados, en esta ocasión se decidió utilizar MongoDB.

Se hace uso de express para poder exponer los servicios del backend y en el caso del frontend se utiliza request, para comunicar con la base de datos se hace uso de la librería mongoose



Imágenes Docker

Con el cluster creado y la aplicación desarrollada, se puede proceder a crear los archivos para las imágenes en Docker, los cuales se utilizarán posteriormente

En este caso es necesario crear dos archivos, uno para el frontend y otro para el backend, dado que ambas aplicaciones, fueron desarrolladas en nodejs, los archivos serán muy parecidos

1. Creamos un archivo de nombre Dockerfile e internamente, le vamos a pedir que instale las dependencias al crear la imagen y le diremos que corra el servidor, en el archivo colocamos lo siguiente

```
FROM node:12
# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied
# where available (npm@5+)
COPY package*.json ./

RUN npm install
# If you are building your code for production
# RUN npm ci --only=production

# Bundle app source
COPY . .

EXPOSE 8000

CMD ["npm", "run", "dev"]
```

2. Guardamos el archivo y adicional haremos un script que nos ayudara a ejecutar el dockerfile, para esto creamos un nuevo archivo al cual le pondremos build.sh y le agregaremos el siguiente comando:

   ```
   docker build --no-cache -t linkheavy/backend .
   ```

   En este comando colocamos nuestro id de Docker hub y el nombre de la imagen, la diferencia entre el backend y el frontend será únicamente el nombre y el puerto en el que se exponen.

3. De momento dejaremos estos archivos, ya que ahora procederemos a instalar las dependencias necesarias para utilizar kubernetes y desplegar los archivos YAML

Dependencias para kubernetes

Ya con el cluster creado la aplicación desarrollada y los archivos Docker preparados, es necesario instalar algunas dependencias, para esto utilizamos una maquina local Linux.

1. Vamos a instalar un cliente de Google cloud en nuestro equipo local, para poder utilizar los servicios de kubernetes
   a. Descargamos el paquete:
      curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-sdk-297.0.0-linux-x86_64.tar.gz
   b. Descomprimimos el paquete descargado
      tar zxvf google-cloud-sdk-297.0.0-linux-x86_64.tar.gz google-cloud-sdk
   c. Corremos el script que hará la instalación
      ./google-cloud-sdk/install.sh
2. Con el cliente de gcloud instalado ahora instalaremos kubernetes

    a. Descargamos el paquete de kubernetes
       curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.18.0/bin/linux/amd64/kubectl
    b. Le damos permisos al archivo
       chmod +x ./kubectl
    c. Movemos el archivo a la carpeta bin
       sudo mv ./kubectl /usr/local/bin/kubectl
3. Ya con Kubernetes en nuestro equipo, lo siguiente será loggearnos en nuestra cuenta de gcloud para conectarnos a nuestro cluster creado anteriormente, para ello utilizamos el siguiente comando, en el cual incluimos el nombre de nuestro cluster y la zona en la que se encuentra
   gcloud container clusters get-credentials cluster-1 --zone=us-central1-c
4. El comando anterior nos desplegara una pagina web de gcloud donde se nos solicitan nuestras credenciales para autenticarnos, las ingresamos y esto ya nos da acceso a nuestro cluster para realizar los despliegues posteriormente.
5. Ahora necesitaremos del paquete Helm, para instalar nginx-ingress para los balanceadores de carga, aplicaremos los siguientes comandos
    a. Lo descargamos: wget https://get.helm.sh/helm-v3.2.4-linux-amd64.tar.gz
    b. Lo descomprimimos: tar -xzvf helm-v3.2.4-linux-amd64.tar.gz
    c. Cambiamos la ubicación: sudo mv linux-amd64/helm /sbin
    d. Agregamos el repositorio: helm repo add stable https://charts.helm.sh/stable
    e. Instalamos nginx-ingress: helm install nginx-ingress stable/nginx-ingress -n proyecto

Despliegue de la aplicación

Después de instalar las dependencias anteriores, podemos proceder a desplegar nuestra aplicación, para ello haremos uso de archivos yaml y las imágenes en Docker

1. Primero crearemos la imagen Docker del backend
    a. Chmod +x build.sh
    b. Sudo ./build.sh
    c. Esperamos que se cree la imagen y cuando finalice subiremos la imagen a nuestro repositorio
    d. Sudo Docker push linkheavy/backend
2. Ahora creamos nuestro archivo yaml, al cual colocaremos el nombre de la imagen el puerto, entre otros datos como el replica set.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    environment: backend
  name: backend-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      environment: backend
  minReadySeconds: 10
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        environment: backend
    spec:
      containers:
      - image: linkheavy/backend
        name: back
        ports:
        - containerPort: 8000
```

3. Ahora ya podemos desplegar la aplicación con los siguientes comandos
    a. Kubectl apply -f backend.yaml
    b. Nos crea el servicio y ahora lo exponemos
    c. Kubectl expose deployment backend-deploy –type=LoadBalancer –name=my-backend
4. El proceso anterior lo repetiremos para el frontend
   Es necesario tomar en cuenta que la ip generada en el backend es necesaria para configurar la conexión del frontend, es por ello que se despliega primero el backend para este caso.

Realizado todo lo anterior ya tenemos desplegada nuestra aplicación, para consultar datos podemos usar los siguientes comandos

- Kubectl get services
- Kubectl logs service/my-backend