

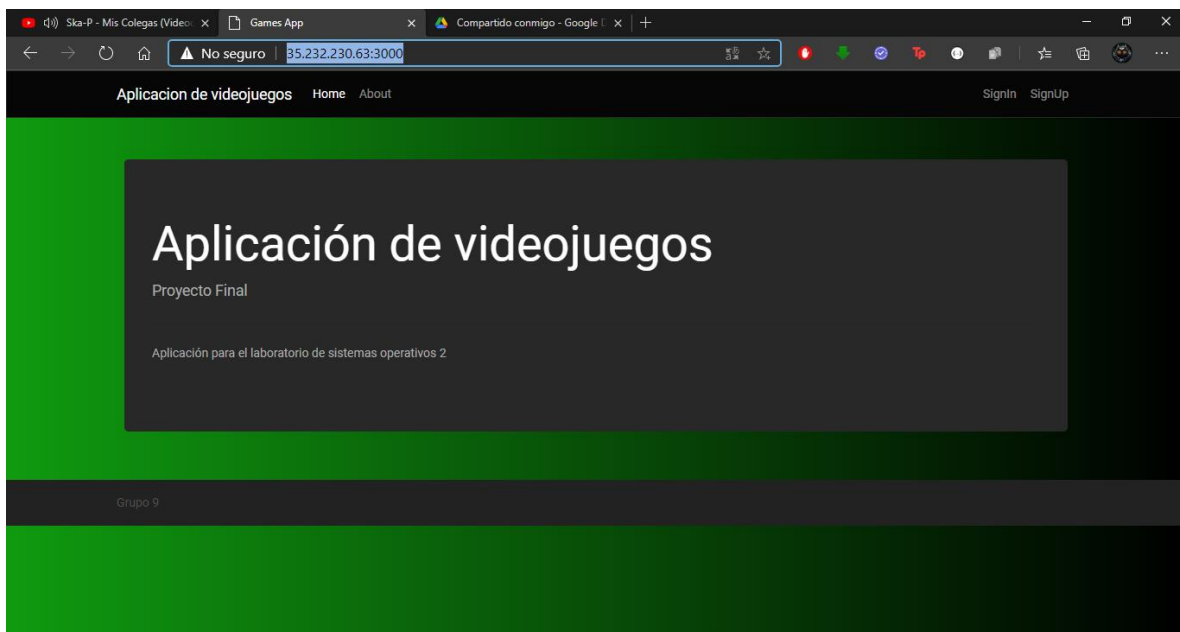
Universidad de San Carlos de Guatemala  
Facultad de ingeniería  
Escuela de vacaciones diciembre 2020  
Laboratorio de sistemas operativos 2

Robinson Jonathan Pérez Fuentes      201114056  
Luis Ricardo Hernández                201114490

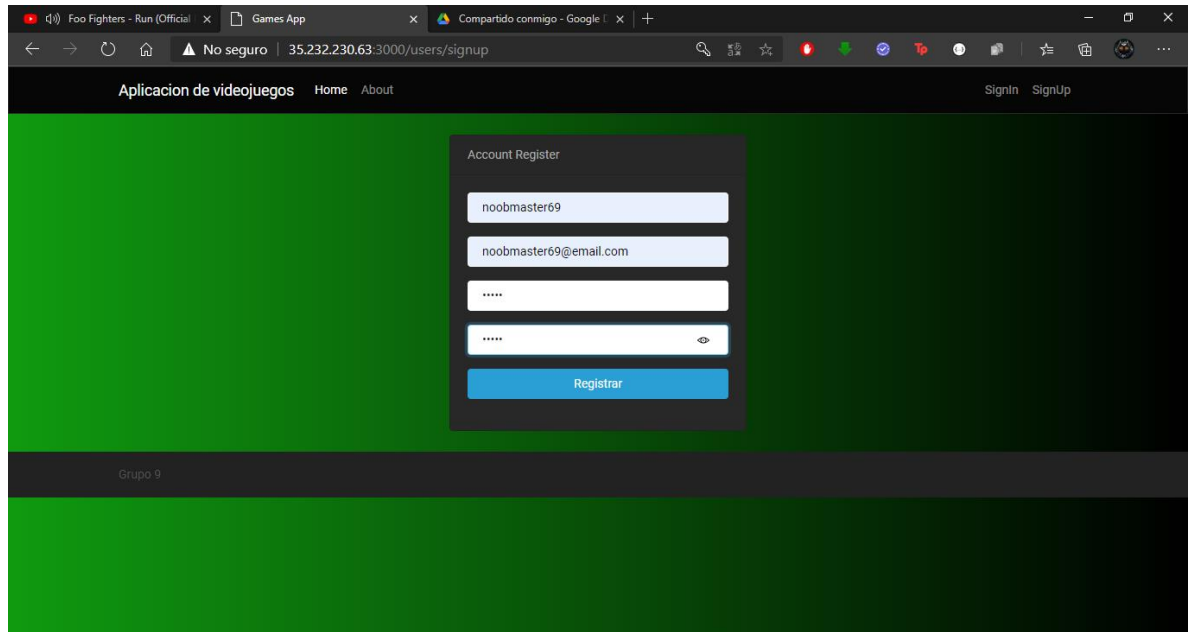
# Manual proyecto final

## Uso de la aplicación

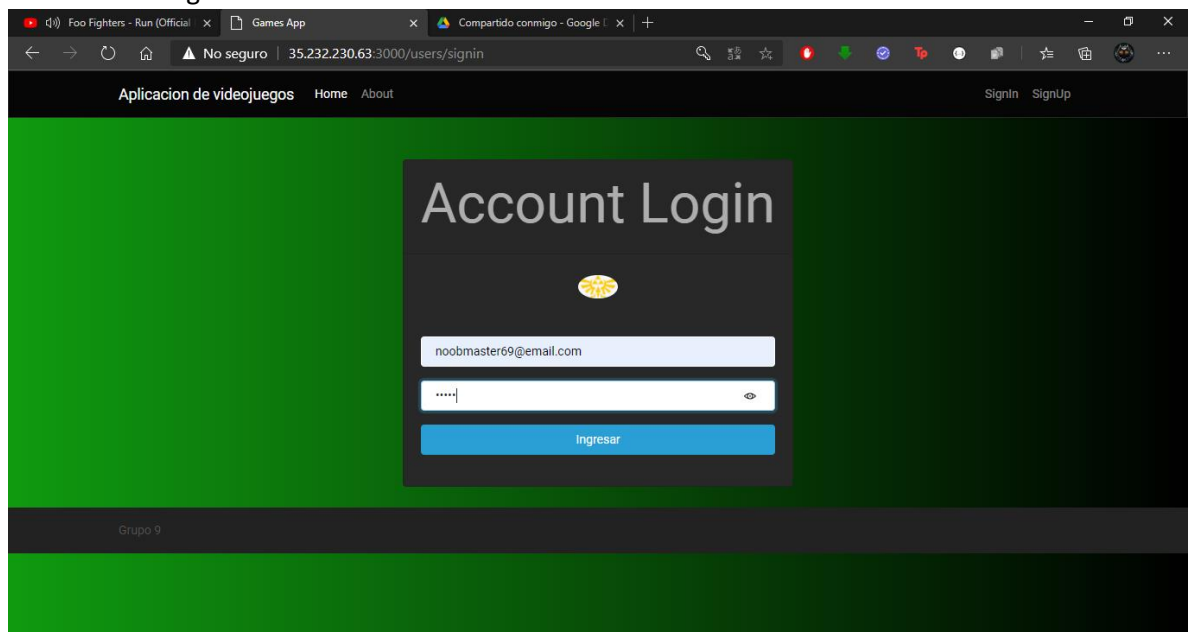
1. Para poder utilizar la aplicación debemos dirigirnos a la siguiente dirección  
<http://35.232.230.63:3000>  
acá se nos despliega la pagina de inicio del sistema



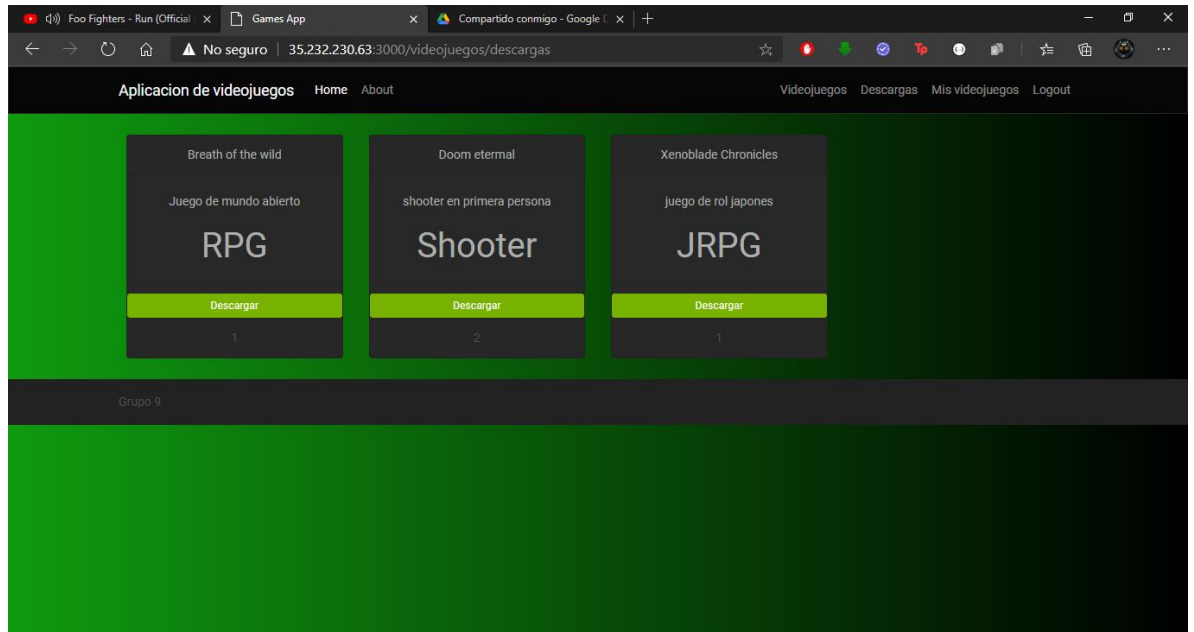
2. Lo primero que tendremos que hacer es crearnos una cuenta para poder descargar videojuegos, para ello debemos dirigirnos a “signUp” e ingresar los datos que se nos solicitan para crearnos un usuario



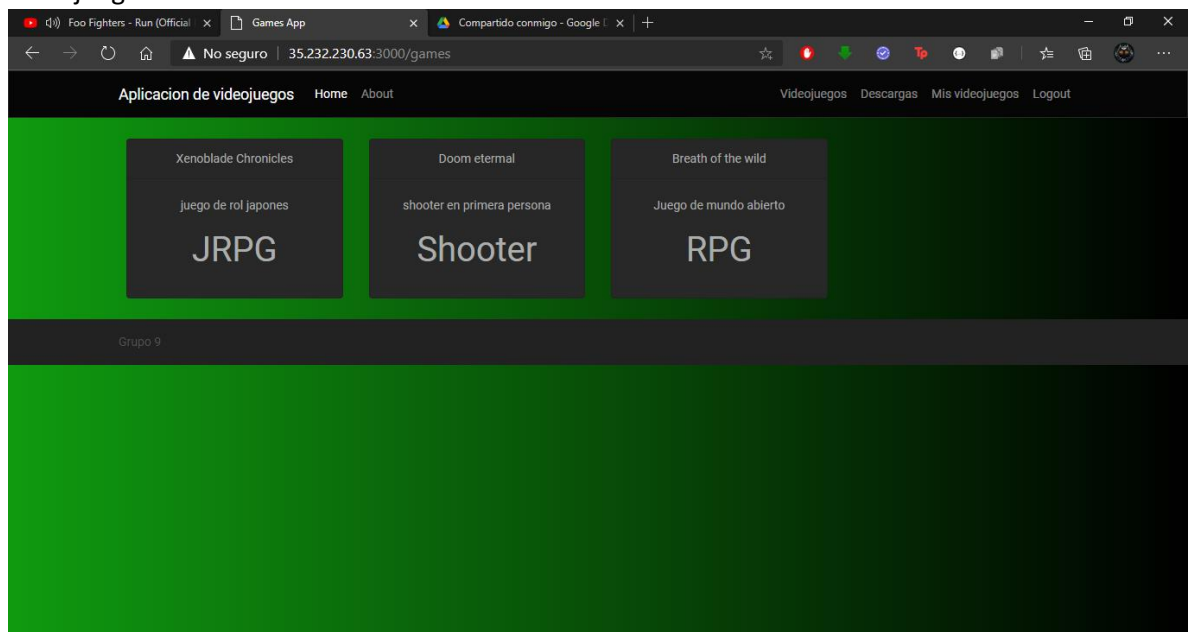
3. Si los datos ingresados son correctos, automáticamente se nos dirige a la página de “SignIn”, donde debemos colocar el correo y contraseña registrado anteriormente y darle al botón de “Ingresar”.



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 logearnos nuevamente con otro usuario
7. Adicional la pagina cuenta con un modulo administrativo que permite, agregar o eliminar videojuegos

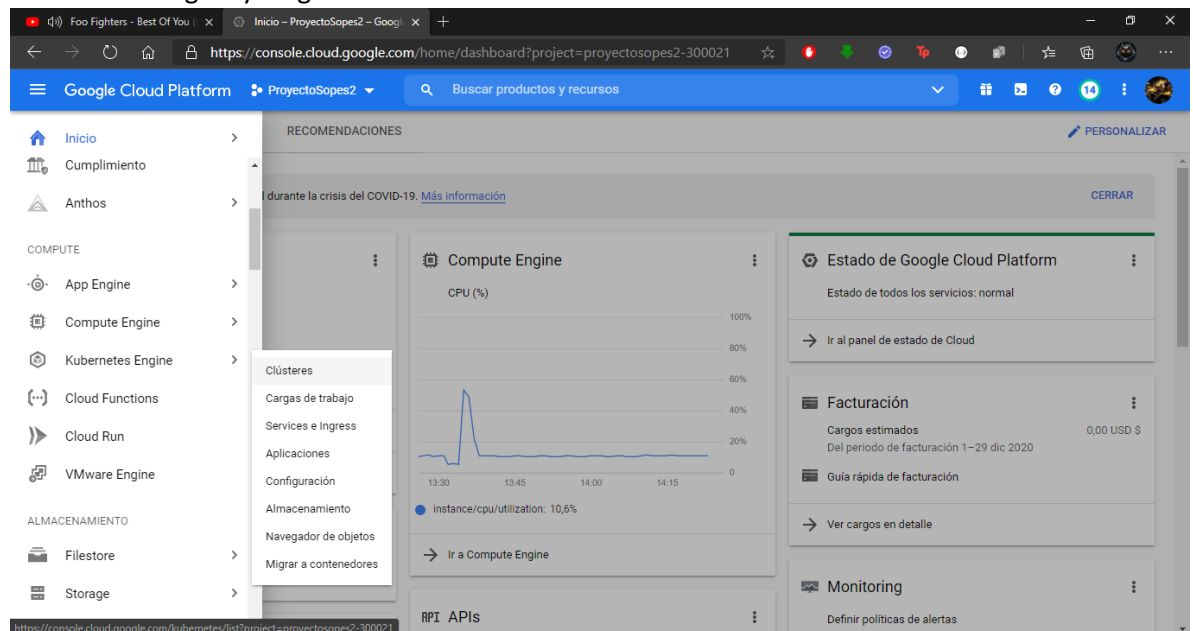
## Manual técnico

Desarrollo de la infraestructura y despliegue de la aplicación

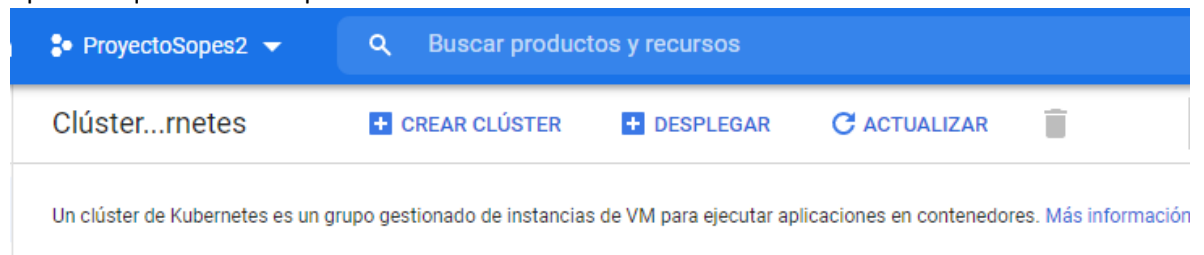
### Creación del clúster

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 Clústeres.



2. Le damos a la opción “Crear clúster” 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

← → ↻ 🔍 <https://console.cloud.google.com/kubernetes/add?project=projectosopes2-300021>

Google Cloud Platform ProyectoSopes2 🔍 Buscar productos y recursos

← Crear un clúster de Kubernetes [AÑADIR GRUPO DE NODOS](#) [QUITAR GRUPO DE NODOS](#)

Información básica de los clústeres

GRUPOS DE NODOS

- workers
  - Nodos
  - Seguridad
  - Metadatos

CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

### Detalles de grupo de nodos

El clúster se creará con al menos un grupo de nodos, que funciona como una plantilla para los grupos de nodos creados en este clúster. Después de crear el clúster, se pueden añadir o quitar grupos de nodos.

Nombre  
workers

Versión del nodo  
1.16.15-gke.4901 (versión maestra)

Tamaño

Número de nodos \*  
1

El intervalo de direcciones del pod limita el tamaño máximo del clúster [Más información](#)

☒ Habilitar autoescalado ⓘ

Número mínimo de nodos \*  
1

Número máximo de nodos \*  
3

☐ Especificar ubicaciones de nodos ⓘ

Predefinido: us-central1-c

Automatización

☒ Habilitar actualización automática ⓘ

⚠ Es posible que los clústeres con menos de 3 nodos experimenten periodos inactivos durante las actualizaciones

☒ Habilitar reparación automática ⓘ

[CREAR](#) [CANCELAR](#) [REST o línea de comando equivalente](#)

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

← → ↻ 🔍 <https://console.cloud.google.com/networking/firewalls/add?project=projectosopes2-300021>

⚠ A partir del 15 de enero del 2021, la consola de Cloud dejará de estar disponible en español. Selecciona otro idioma o la consola cambiará de forma predefinida a inglés (EE. UU.). Las descargas de la aplicación móvil estarán en inglés (EE. UU.) de forma predefinida a partir del 15 de enero. [CAMBIAR IDIOMA](#) [CERRAR](#)

Google Cloud Platform ProyectoSopes2 🔍 Buscar productos y recursos

Red de VPC

- Redes de VPC
- Direcciones IP externas
- Cortafuegos
- Rutas
- Emparejamiento entre redes...
- VPC compartida
- Acceso a VPC sin servidor
- Replicación de paquetes

← Crear una regla de cortafuegos

Prioridad...  
1000 [COMPROBAR PRIORIDAD DE OTRAS REGLAS DE CORTAFUEGOS](#) ⓘ

La prioridad puede estar entre 0 y 65535

Dirección del tráfico ⓘ

☒ Entrada

☐ Salida

Acción tras coincidencia ⓘ

☒ Permitir

☐ Denegar

Destinos

Etiquetas de destino especificadas

Etiquetas de destino \*  
etiquet...

Filtrar por origen

Intervalos de IP

Intervalos de IP de origen \*  
0.0.0.0/0 por ejemplo: 0.0.0.0, 192.168.2.0/24 ⓘ

Filtro de origen secundario  
Ninguno ⓘ

Protocolos y puertos ⓘ

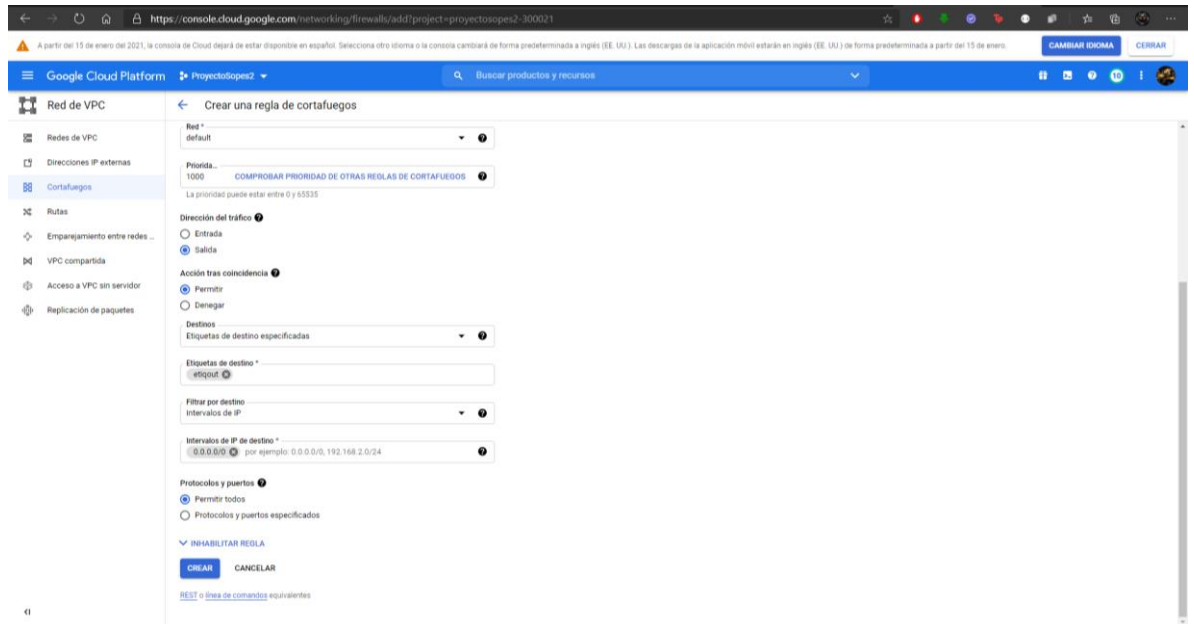
☒ Permitir todos

☐ Protocolos y puertos especificados

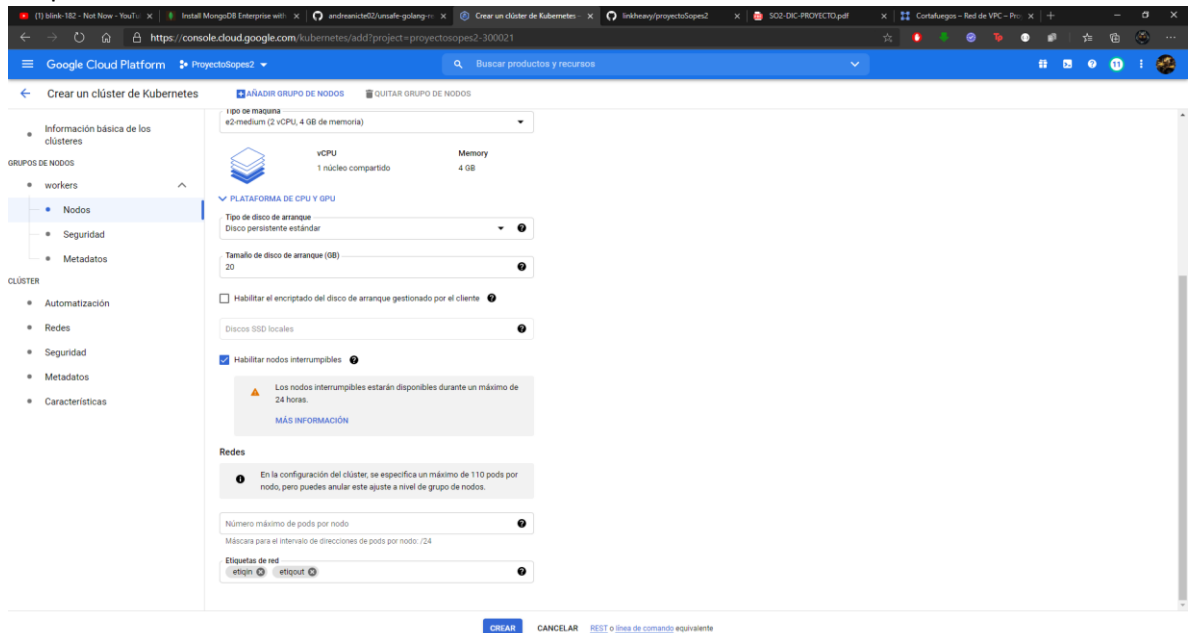
✓ INHABILITAR REGLA

[CREAR](#) [CANCELAR](#)

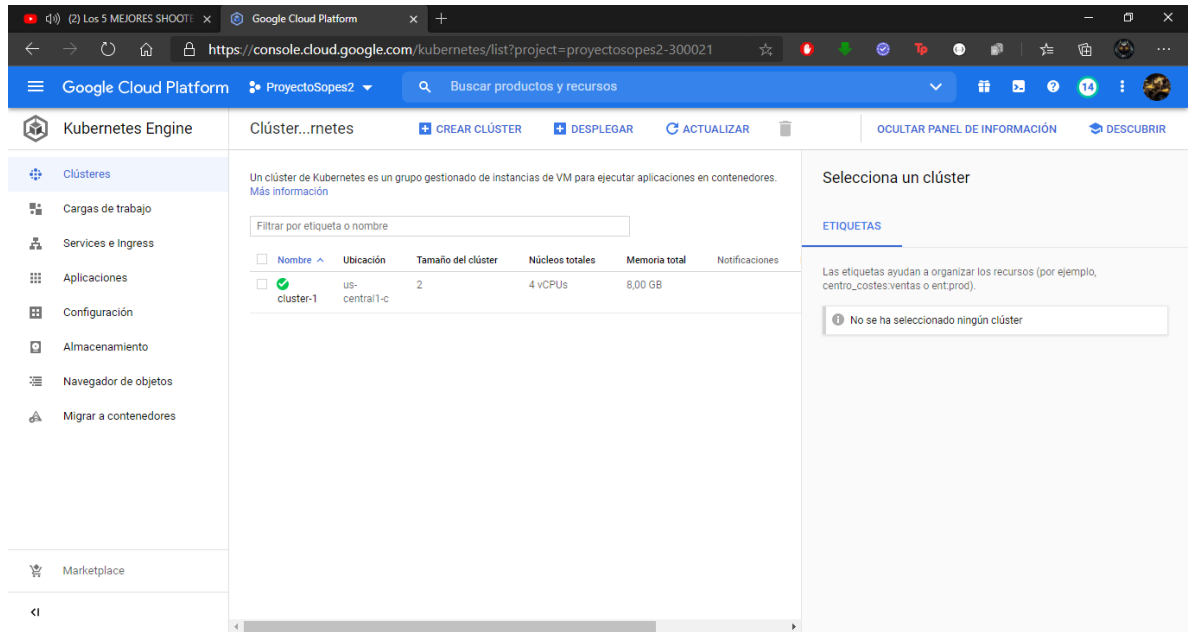
[REST o línea de comando equivalente](#)



- 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 colocó 20 GB por temas de costos



- 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

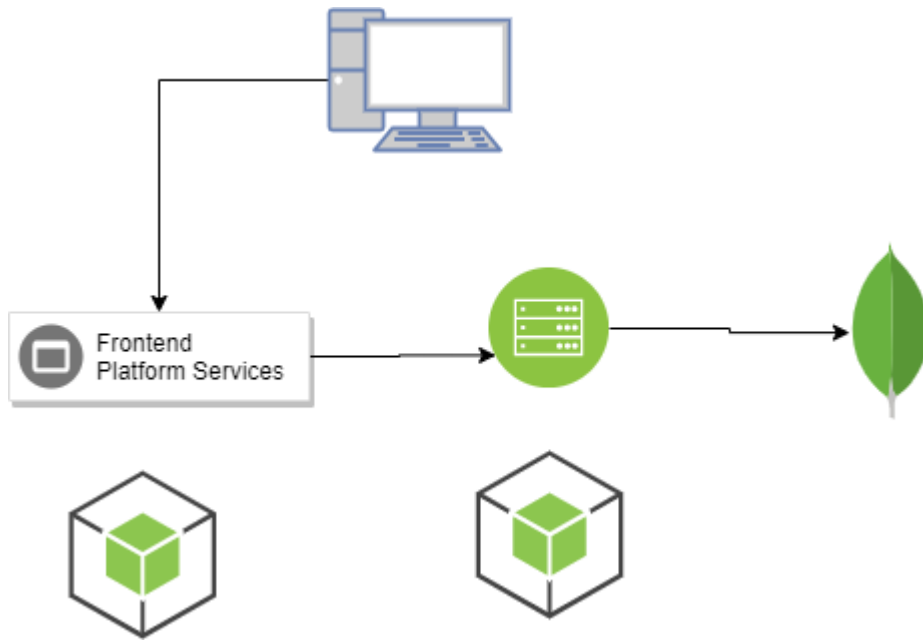
Para el desarrollo de la aplicación, se utilizaron las siguientes tecnologías

- 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



```
1 FROM node:12
2 # Create app directory
3 WORKDIR /usr/src/app
4
5 # Install app dependencies
6 # A wildcard is used to ensure both package.json AND package-lock.json are copied
7 # where available (npm@5+)
8 COPY package*.json ./
9
10 RUN npm install
11 # If you are building your code for production
12 # RUN npm ci --only=production
13
14 # Bundle app source
15 COPY . .
16
17 EXPOSE 8000
18
19 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. Descomprimos 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 logearnos 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 desplegará una página 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 descomprimos: `tar -xzf 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.

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: backend
6    name: backend-deploy
7  spec:
8    replicas: 3
9    selector:
10     matchLabels:
11       environment: backend
12    minReadySeconds: 10
13    strategy:
14     rollingUpdate:
15       maxSurge: 1
16       maxUnavailable: 0
17     type: RollingUpdate
18    template:
19     metadata:
20       labels:
21         environment: backend
22     spec:
23       containers:
24       - image: linkheavy/backend
25         name: back
26         ports:
27         - containerPort: 8000
```

3. Ahora ya podemos desplegar la aplicación con los siguientes comandos
  - a. `Kubectrl apply -f backend.yaml`
  - b. Nos crea el servicio y ahora lo exponemos
  - c. `Kubectrl 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

- `Kubectrl get services`
- `Kubectrl logs service/my-backend`