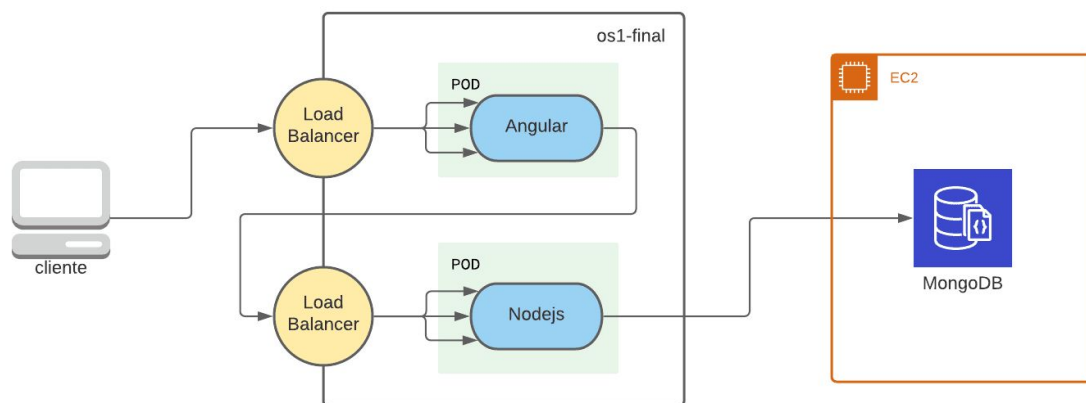


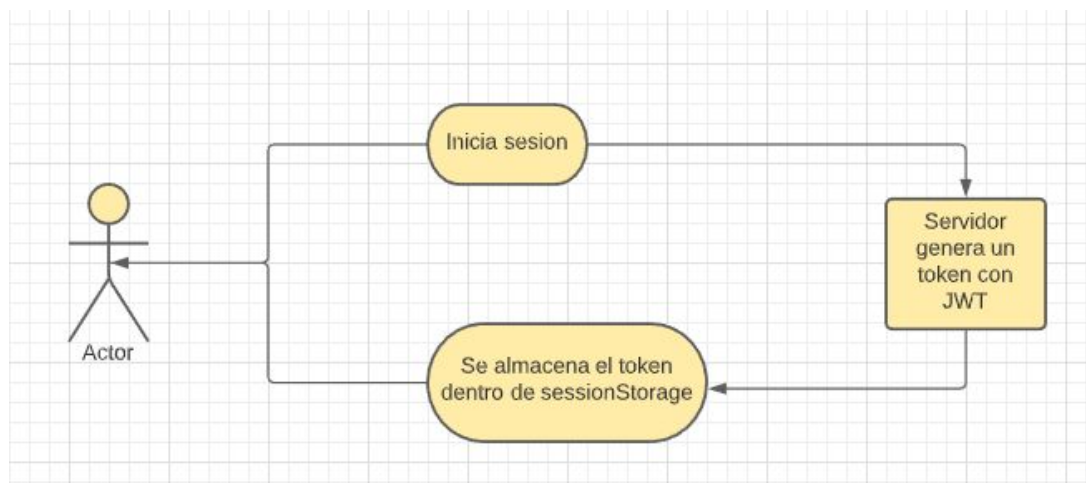
Arquitectura



La app consistirá en un gestor de tareas por hacer que le permitirá a las personas crear un usuario, hacer login para así que pueda empezar a crear tareas además de poder eliminarlas marcandolas como terminadas cuando el usuario decida hacerlo.

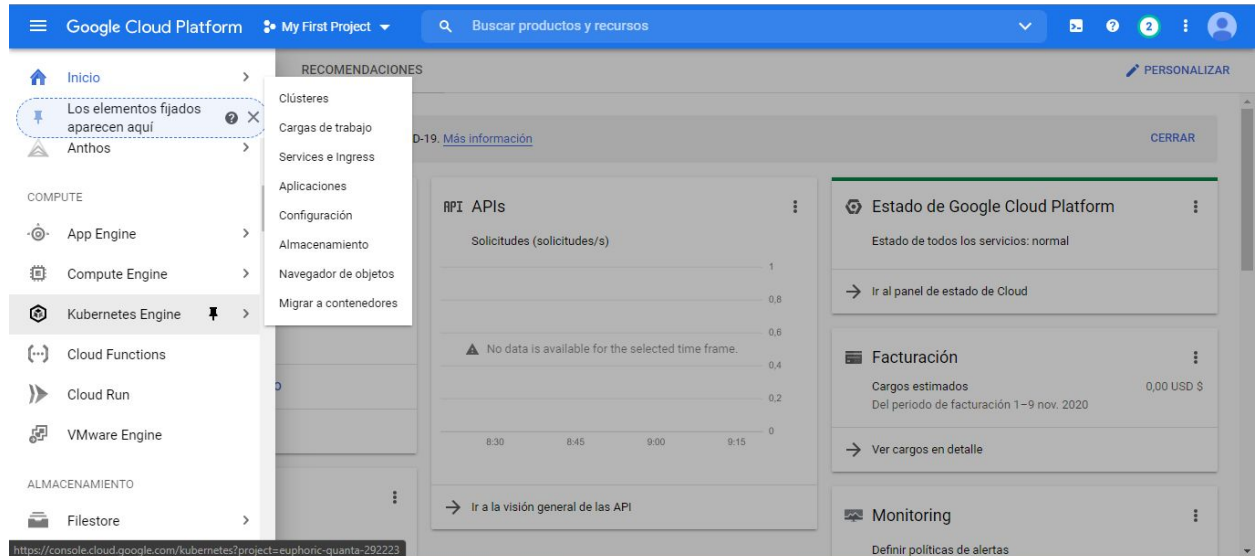
Para el manejo de sesiones se usará JWT que es una forma de gestionar información a partir de tokens encriptados, el token se genera en el servidor enviándola al frontend y este lo almacena en la session storage que manejan los navegadores.

Manejo de Sesión:

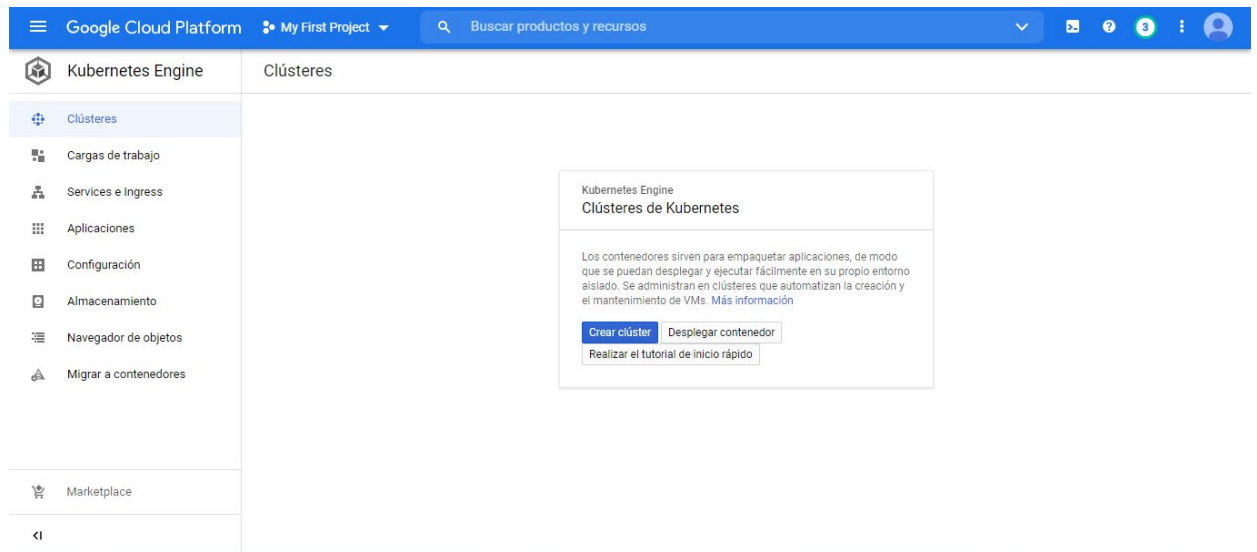


Creación del Cluster

1. Debemos crear .



2. Damos click en el botón azul de crear cluster



3. Ahora configuraremos las opciones del cluster, pondremos el nombre que preferimos lo demás puede quedar por defecto.

Google Cloud Platform My First Project Buscar productos y recursos

Crear un clúster de Kubernetes AÑADIR GRUPO DE NODOS ELIMINAR GRUPO DE NODOS

Información básica de los clústeres

El clúster nuevo tendrá el nombre, la versión y la ubicación que especifiques aquí. Después de crear el clúster, su nombre y ubicación no se pueden cambiar.

Para experimentar con un clúster asequible, prueba **Mi primer clúster** en Guías de configuración de clústeres.

Nombre
clustersit

Tipo de ubicación
☒ Zona
☐ Regional

Zona
us-central1-c

☐ Especificar ubicaciones de nodo predeterminadas
 Valor predeterminado actual: us-central1-c

Versión maestra

CREAR CANCELAR REST o línea de comando equivalente

4. pero si lo deseamos podemos cambiar las configuraciones de los nodos que se crearan dependiendo si queremos darles mas o menos potencia, para eso nos dirigimos al apartado de default-pool, acá podemos cambiar el número de nodos que se crearán y la capacidad que cada uno tendrá.

Google Cloud Platform My First Project Buscar productos y recursos

Crear un clúster de Kubernetes AÑADIR GRUPO DE NODOS ELIMINAR GRUPO DE NODOS

default-pool

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
default-pool

Versión del nodo
1.16.13-gke.401 (versión maestra)

Tamaño
Número de nodos *
3

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

☐ Habilitar autoescalado

☐ Especificar ubicaciones de nodos

Predeterminado: us-central1-c

CREAR CANCELAR REST o línea de comando equivalente

Google Cloud Platform My First Project Buscar productos y recursos

Crear un clúster de Kubernetes AÑADIR GRUPO DE NODOS ELIMINAR GRUPO DE NODOS

Nodos

Esta configuración se utilizará cuando se creen nodos con este grupo de nodos.

Tipo de imagen
Container-Optimized OS (cos) (predeterminado)

Configuración de la máquina

Familia de máquinas
 USO GENERAL OPTIMIZADA PARA LA COMPUTACIÓN CON MEMORIA OPTIMIZADA

Tipos de máquinas para cargas de trabajo habituales, optimizadas en cuanto al coste y a la flexibilidad

Serie
E2

La plataforma de CPU se elige según las que haya disponibles

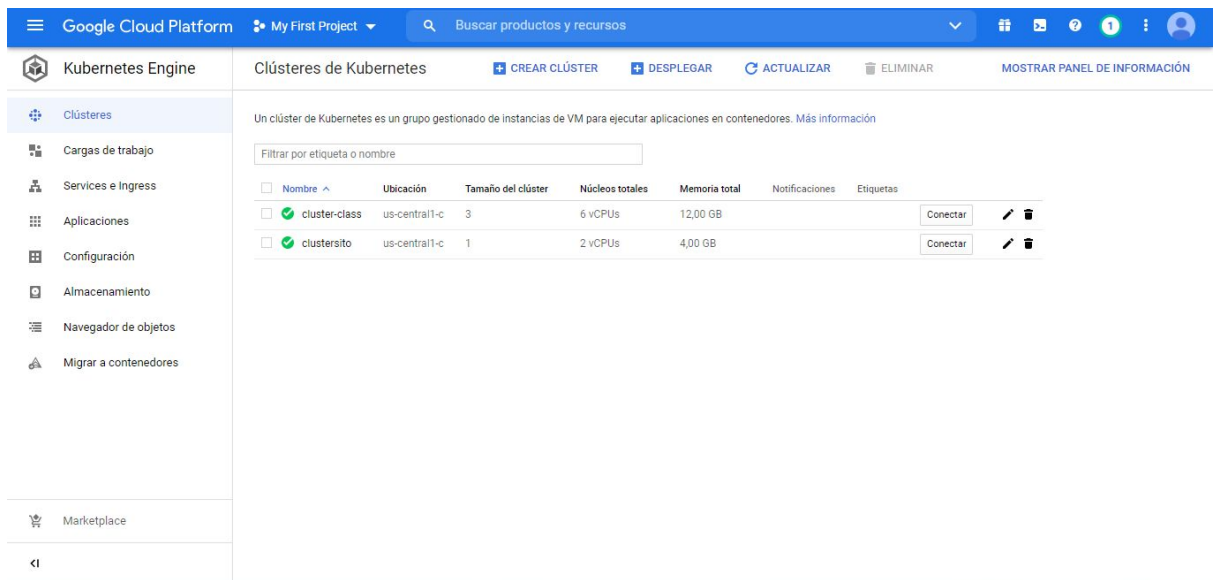
Tipo de máquina
e2-medium (2 vCPU, 4 GB de memoria)

vCPU
1 núcleo compartido

Memory
4 GB

CREAR CANCELAR REST o línea de comando equivalente

5. Una vez estemos conforme con las configuraciones podemos dar en crear y luego podemos ver todos los clusters que hemos creado.



Archivos necesarios

Para que Kubernetes funcione es necesario tener ciertos archivos preparados entre ellos están:

- Imagen de la o las aplicaciones a desplegar.
- Archivos YAML que tendrán la configuración que Kubernetes leerá para crear lo necesario.

Construcción de las imágenes

Para que Kubernetes funcione y pueda crear los pods necesarios se le debe entregar una fuente confiable, estable y rápida de la cual pueda realizar el pull necesario de las imágenes, para esto debemos crear los archivos de tipo Dockerfile que empaquetara nuestra aplicación, el archivo Dockerfile posee la siguiente estructura.

1. **FROM:** Qué imagen base usaremos
2. **RUN:** Comando que se ejecuta para instalar dependencias regularmente.
3. **WORKDIR:** Carpeta base para almacenar la aplicación dentro del contenedor.
4. **COPY:** Instrucción para copiar los archivos necesarios para que funcione la aplicación.
5. **EXPOSE:** sirve para liberar el puerto en el que la aplicación se comunicará con el exterior.

```
FROM node:latest

WORKDIR /app

COPY ["package.json", "package-lock.json*", "./"]

RUN npm install

COPY . .

EXPOSE 3000

CMD [ "npm", "start" ]
```

Bien ya tenemos el archivo para crear las imágenes, pero, aun debemos subir construir la imagen y subirla a algún repositorio para el almacenamiento de imágenes en este caso se usará [Docker hub](#), bien, para ello utilizamos los siguientes comandos:

- **docker login** - esto para vincular la cuenta de docker hub con docker
- **docker build -t <username>/<imagenname>:<tag> <pathDockerfile>**
- **docker push <username>/<imagenname>:<tag>**

Con esto ya tendremos las imágenes cargadas y listas para que Kubernetes pueda hacerle pull.

The screenshot shows the Docker Hub web interface. At the top, there's a navigation bar with the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, Get Help, and the user profile 'pixelcodegg'. Below the navigation bar, there's a dropdown menu for the namespace 'pixelcodegg' and a search bar for repository names. A 'Create Repository' button is visible. The main content area displays a list of repositories for 'pixelcodegg':

Repository Name	Updated	Not Scanned	Stars	Downloads	Visibility
pixelcodegg / f-final	Updated an hour ago	Not Scanned	0	13	Public
pixelcodegg / b-final	Updated 2 hours ago	Not Scanned	0	415	Public
pixelcodegg / python	Updated 3 days ago	Not Scanned	0	2	Public
pixelcodegg / go-lang	Updated 3 days ago	Not Scanned	0	3	Public

On the right side, there are two promotional cards: 'Create an Organization' and 'New Subscription Plans Available!'.

Configuración de Kubernetes

Kubernetes es una herramienta muy versátil que automatiza una gran parte del proceso de despliegue y gestión de nuestras aplicaciones para ello debemos darle como entrada unos archivos en formato YAML para que pueda realizar los procesos necesarios, estos archivos tienen la siguiente estructura.

- **kind:** se especifica el tipo de acciones que debe tomar, puede tomar valores como Deployment, Service, Namespace.
- **replicas:** asigna cuantas réplicas de este pod deseamos crear.
- **selector:** algo importante es que se especifica cuál es el selector para que otros servicios puedan apuntar a este sin ningún problema.
- **spec:** por último se especifica la información de los contenedores, como lo es los puertos, el nombre de la imagen que apunta a nuestro usuario de docker hub.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
  namespace: os1-final
spec:
  replicas: 3
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend-container
          image: pixelcodegg/b-final:1.0
          imagePullPolicy: Always
          ports:
            - containerPort: 3000
```

Luego de esto nos quedan las configuraciones dentro de la consola para crear ejecutar los archivos YAML y terminar con el despliegue, para ello debemos ejecutar los siguientes comandos, en este caso se crearon 3 archivos uno de cada tipo.

- **kubectl apply -f namespace.yml**
- **kubectl apply -f deployment.yml**
- **kubectl apply -f service.yml**

Para verificar que todo está correcto se corre el comando **kubectl get pods**, esto nos listará todos los pods que se han creado gracias a los archivos.

```
soulofhenry@cloudshell:~/OS1-Final$ kubectl get pods
```

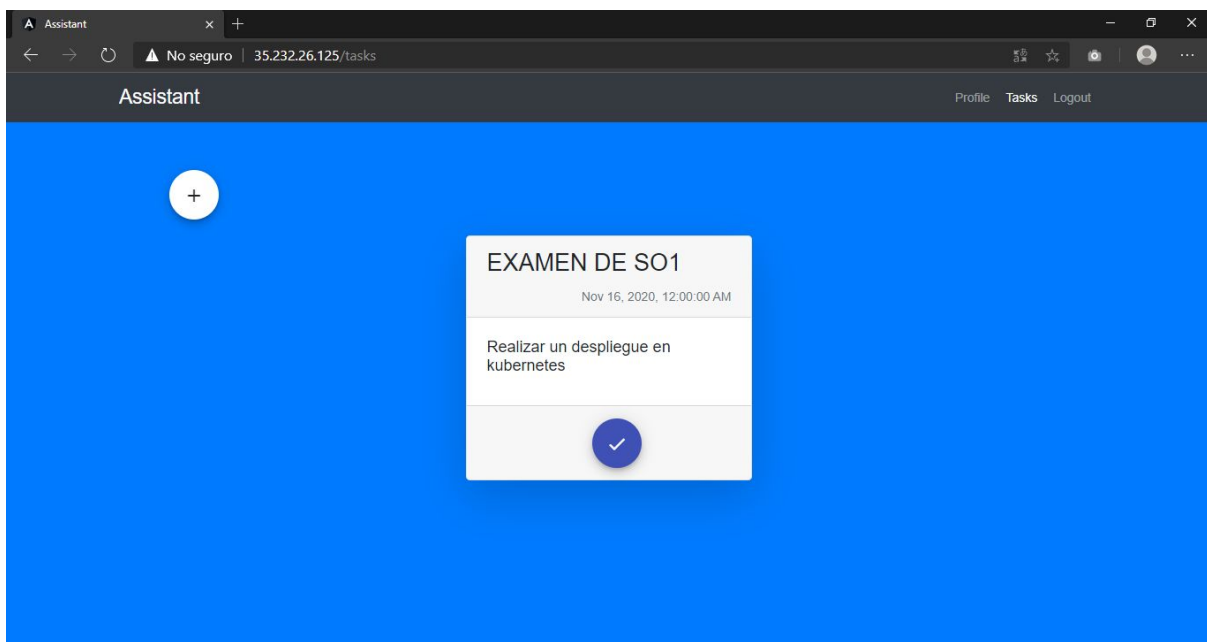
NAME	READY	STATUS	RESTARTS	AGE
backend-deployment-846bc58f55-4qwh9	1/1	Running	0	35s
backend-deployment-846bc58f55-5r77n	1/1	Running	0	35s
backend-deployment-846bc58f55-kpnkl	1/1	Running	0	35s
frontend-deployment-75974b8fcd-9475r	1/1	Running	0	35s
frontend-deployment-75974b8fcd-1l4j4	1/1	Running	0	35s
frontend-deployment-75974b8fcd-xhc5r	1/1	Running	0	35s

Para obtener la ip que nos dará acceso al servicio de frontend y backend se necesita el siguiente comando **kubectl get services**, con esto podemos obtener las ip para asociarlos a algún dominio o lo que queramos hacer con ella.

```
soulofhenry@cloudshell:~/OS1-Final$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
backend-service	LoadBalancer	10.56.14.102	104.198.241.27	80:30350/TCP	124m
frontend-service	LoadBalancer	10.56.9.242	35.232.26.125	80:31598/TCP	98m

Ya tenemos la aplicación desplegada y funcionando perfectamente.



Link del Repositorio y Video

<https://github.com/HenryGalvez/OS1-Final.git>

<https://youtu.be/692ueOqE7L4>