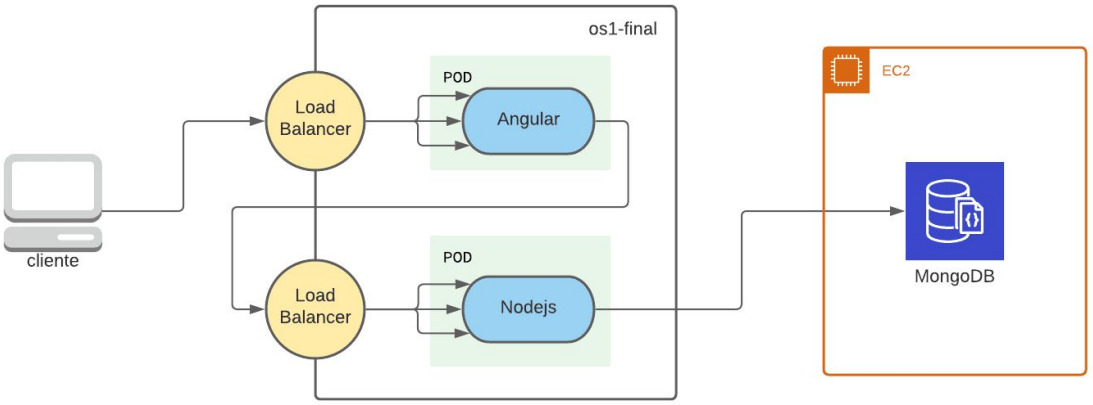
UNIVERSIDAD SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
SISTEMAS OPERATIVOS 1
Ing. Allan Morataya
Henry Adolfo Galvez
201612499

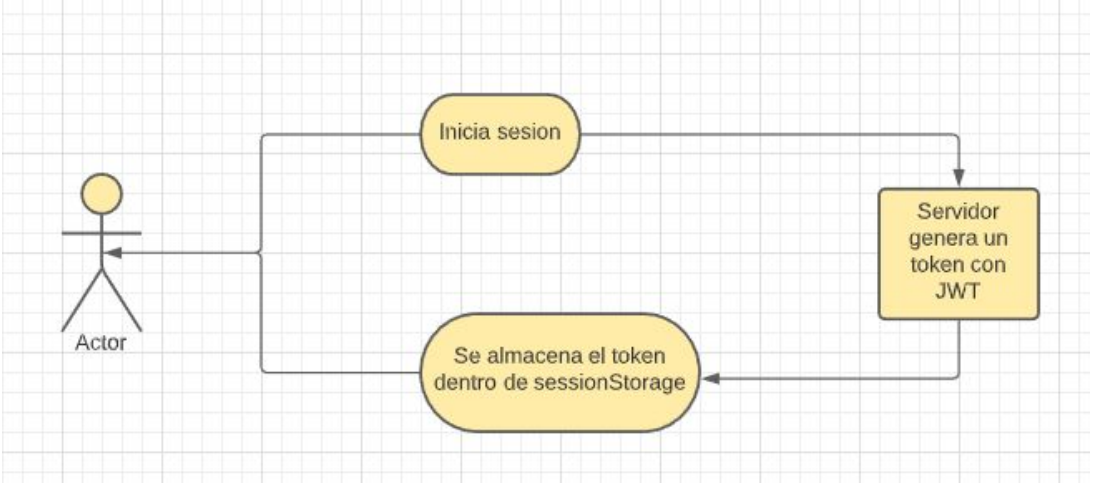
Arquitectura



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.

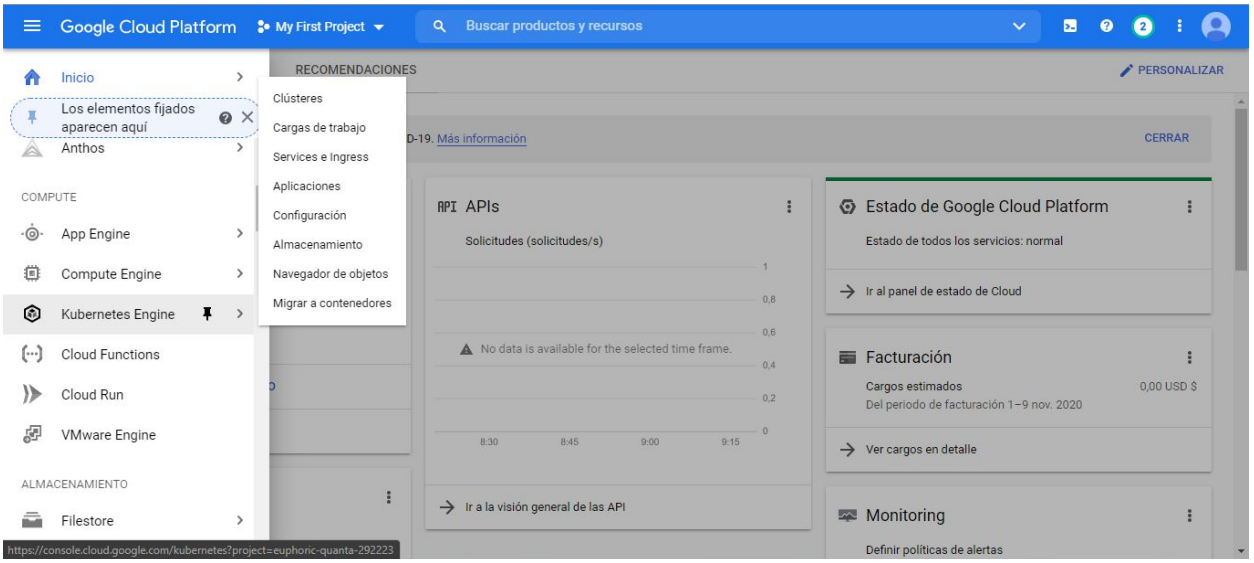
Para el manejo de sesiones se usará JWT que es una forma de gestionar información a partir de tokens encriptados, el token se genera en el servidor enviándola al frontend y este lo almacena en la session storage que manejan los navegadores.

Manejo de Sesión:

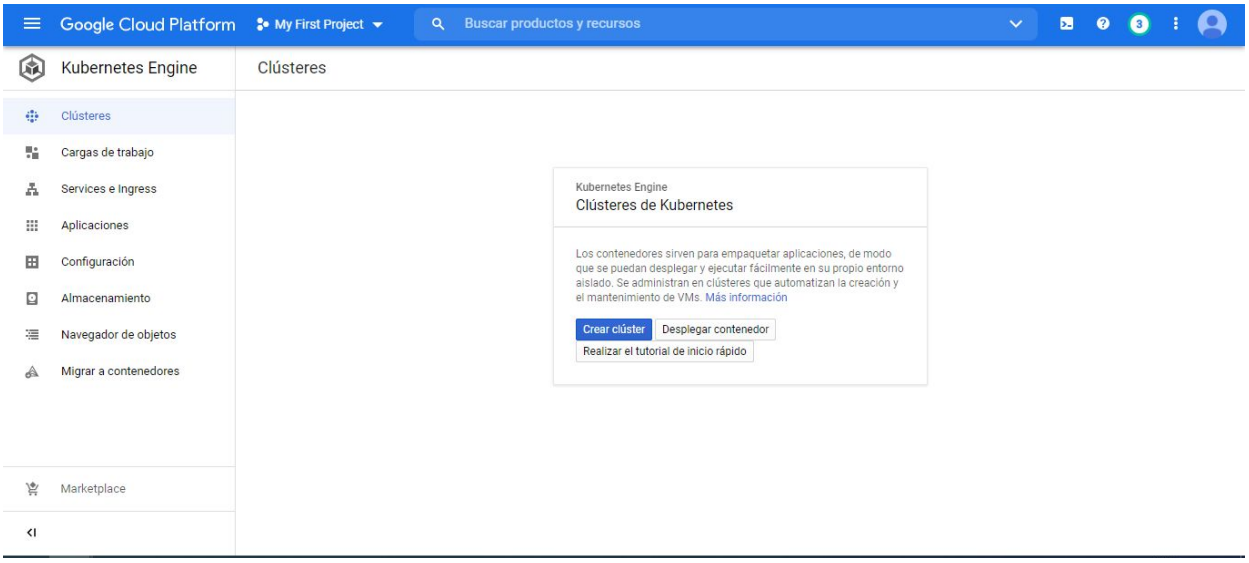


Creación del Cluster

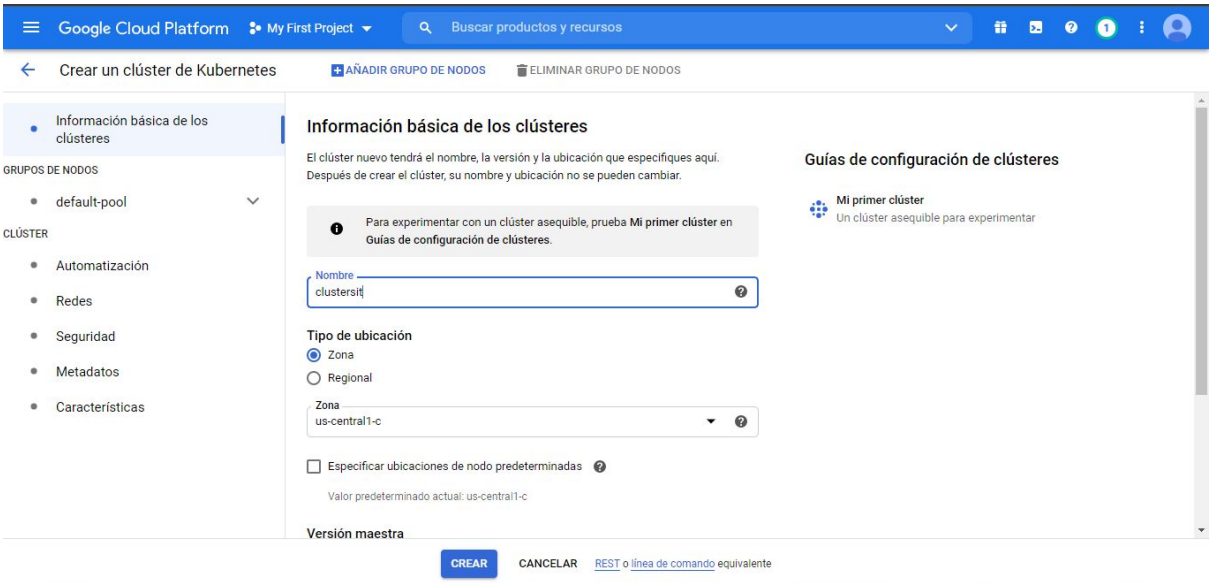
1. Debemos crear .



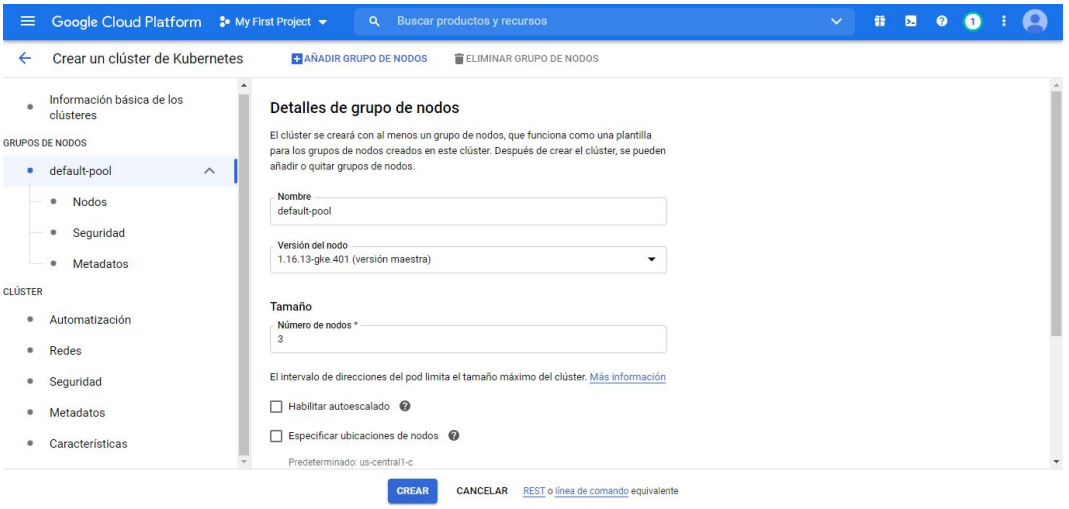
2. Damos click en el botón azul de crear cluster

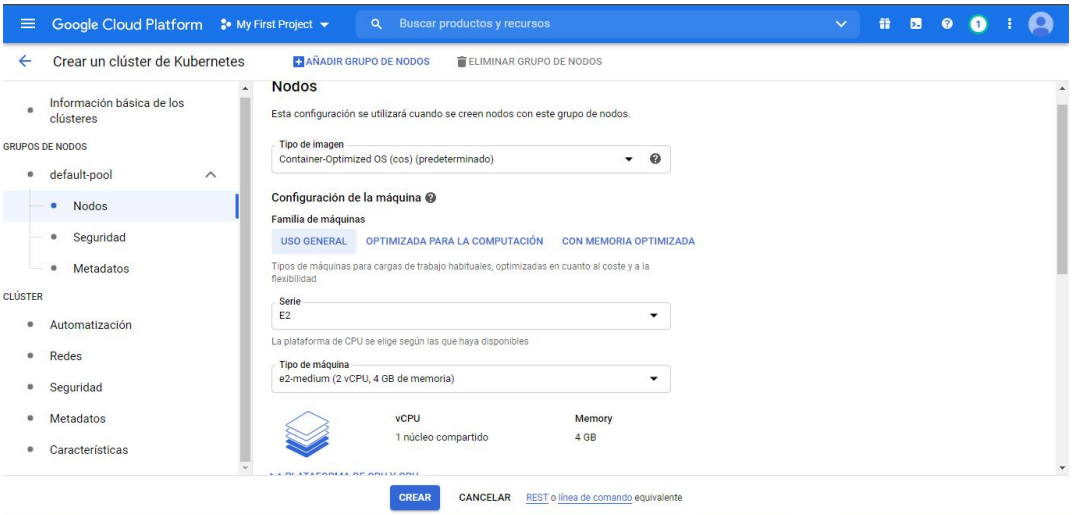


3. Ahora configuraremos las opciones del cluster, pondremos el nombre que preferimos lo demás puede quedar por defecto.

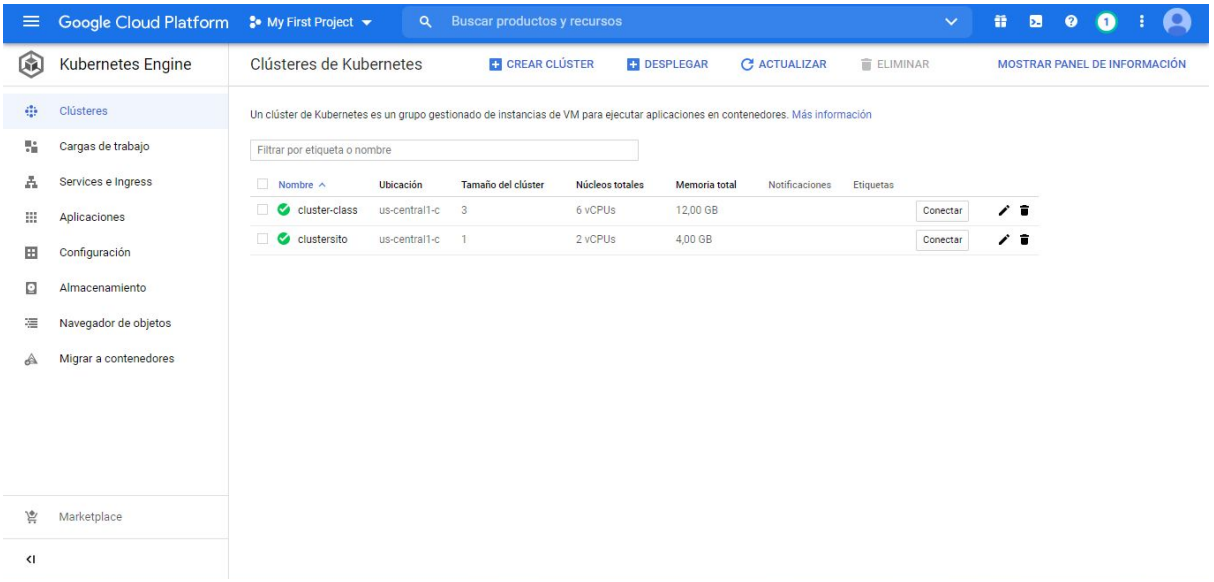


4. pero si lo deseamos podemos cambiar las configuraciones de los nodos que se crearan dependiendo si queremos darles mas o menos potencia, para eso nos dirigimos al apartado de default-pool, acá podemos cambiar el número de nodos que se crearán y la capacidad que cada uno tendrá.





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.

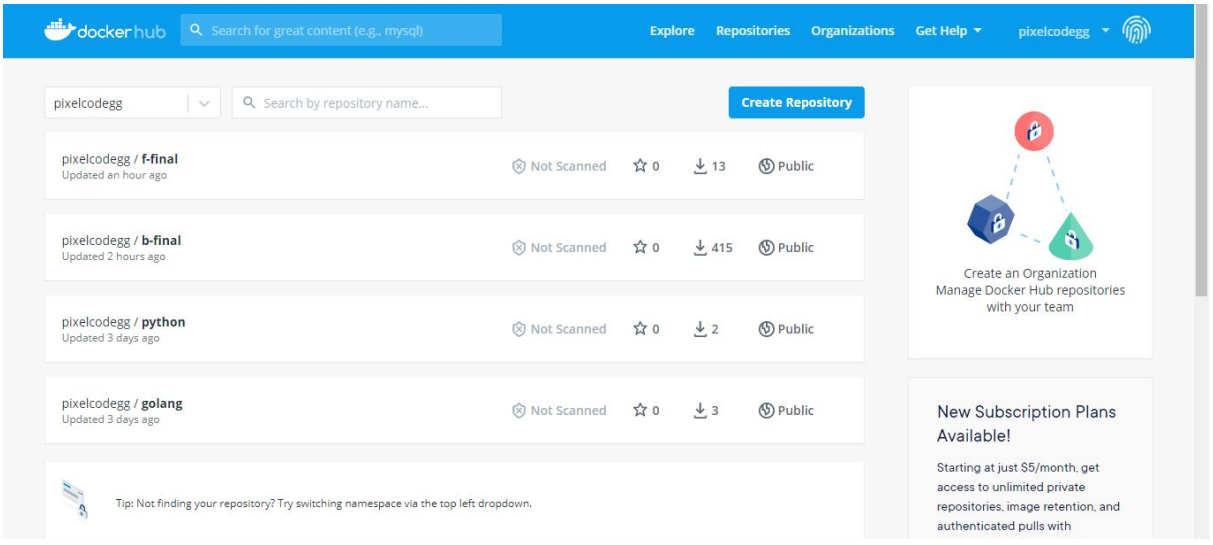
1. **FROM:** Qué imagen base usaremos
2. **RUN:** Comando que se ejecuta para instalar dependencias regularmente.
3. **WORKDIR**: Carpeta base para almacenar la aplicación dentro del contenedor.
4. **COPY**: Instrucción para copiar los archivos necesarios para que funcione la aplicación.
5. **EXPOSE**: 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 imagenes, 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>/<imagename>:<tag> <pathDockerfile>**
- **docker push <username>/<imagename>:<tag>**

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



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.

```yaml
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**

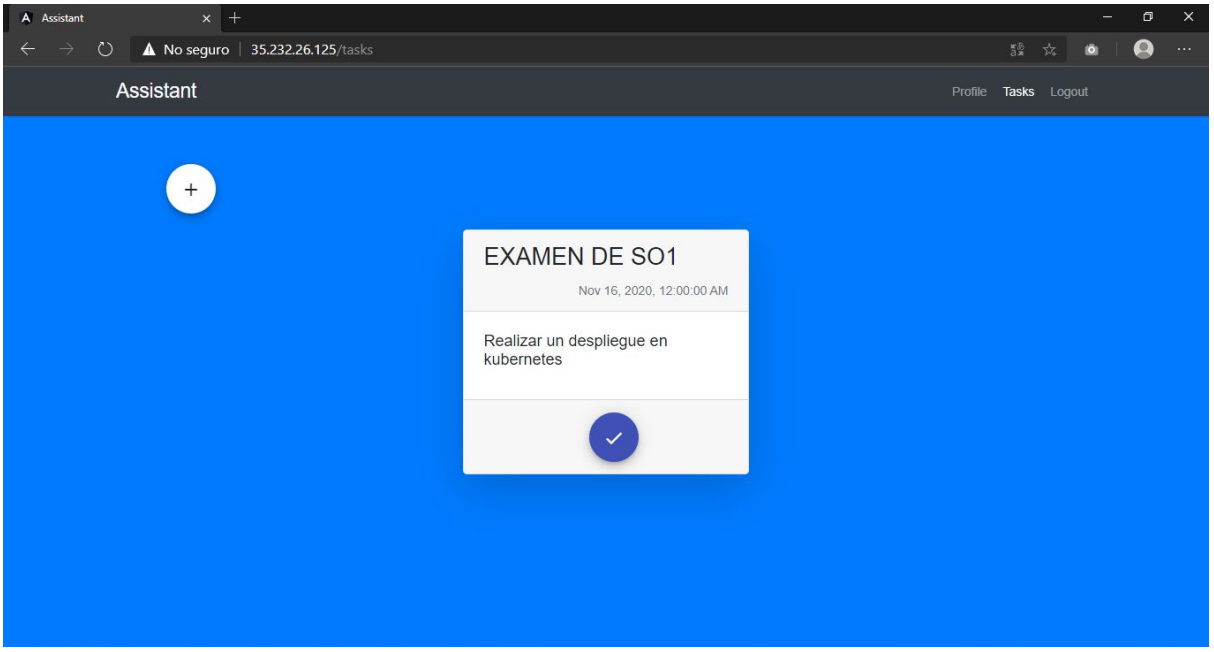
Para verificar que todo está correcto se corre el comando **kubectl get pods,** esto nos listará todos los pods que se han creado gracias a los archivos.

```
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-ll4j4   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