**Final Project: End-to-End DevOps Deployment**

1. **Configuración del repositorio y entorno de Desarrollo**

**Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.**

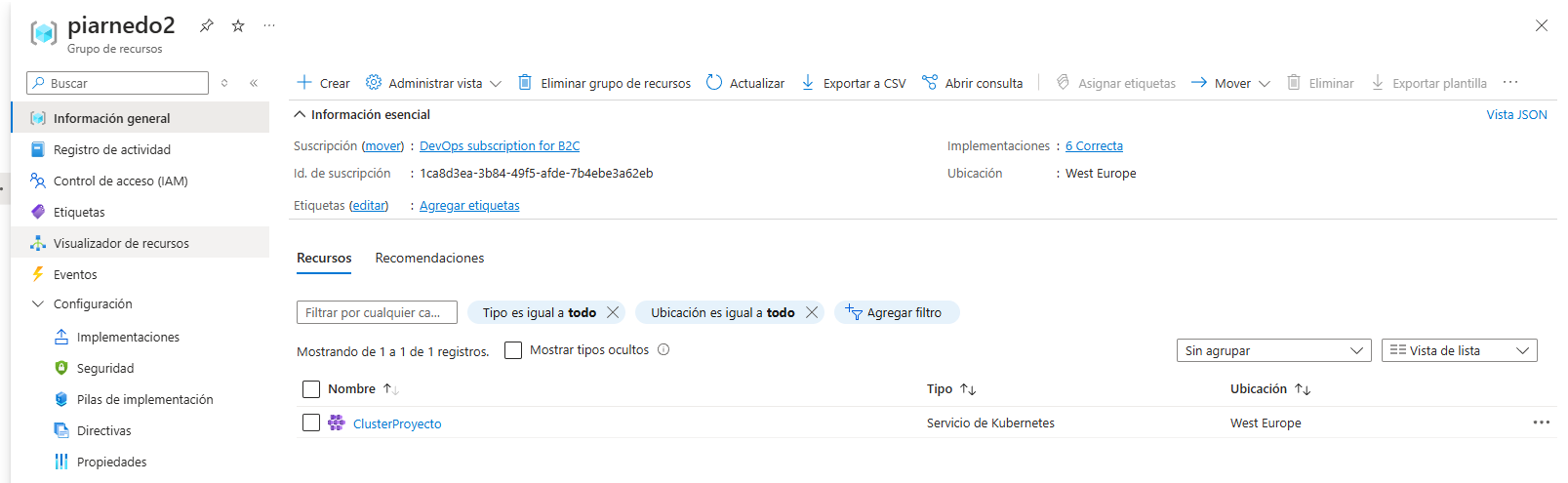
* 1. **Clonar el codigo**

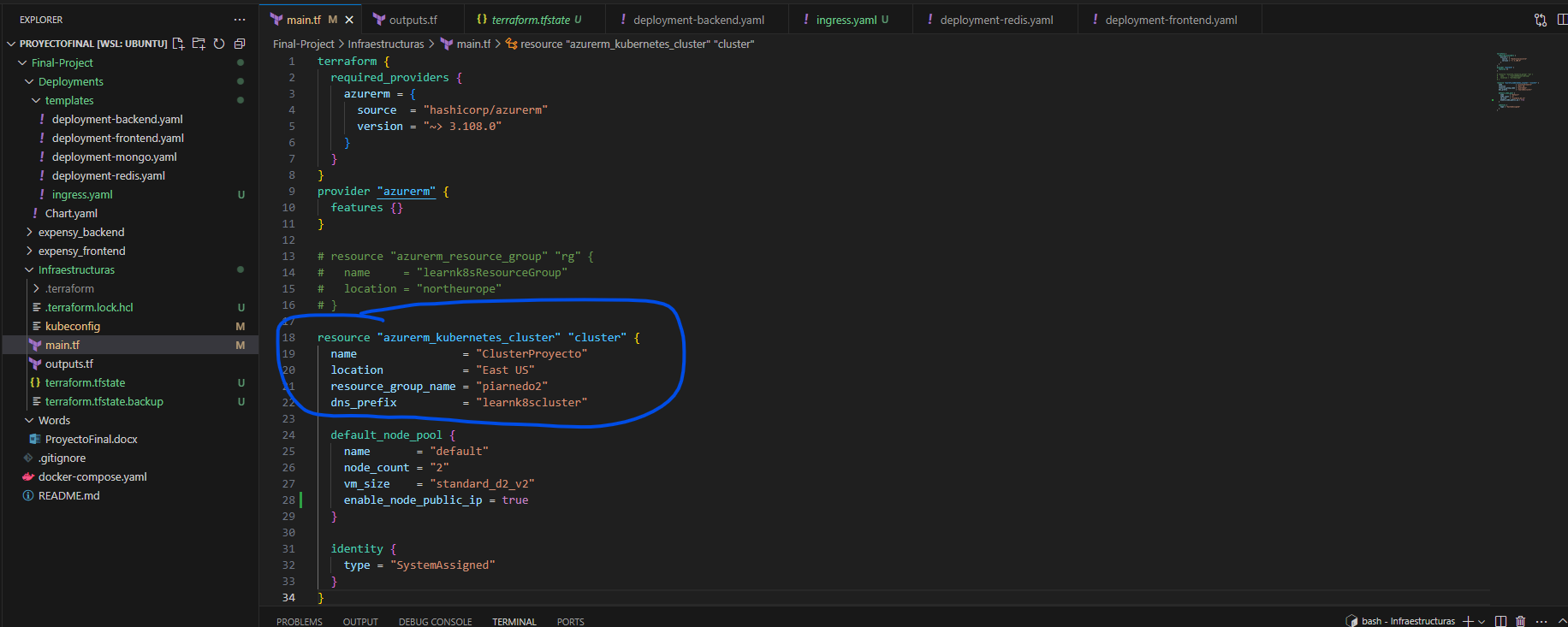
Texto

El contenido generado por IA puede ser incorrecto.

1. **Infrastructure as Code (IaC):**
   1. **Crear Azure AKS:**

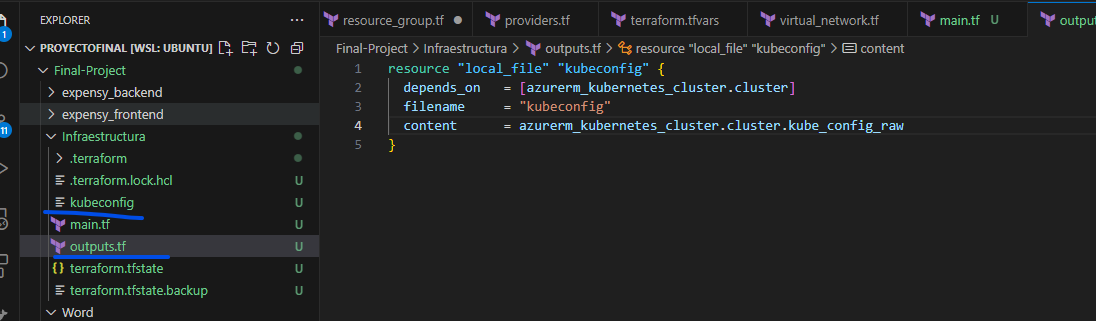
**Tener el resource group creado o crearlo de antes (resultado final):**



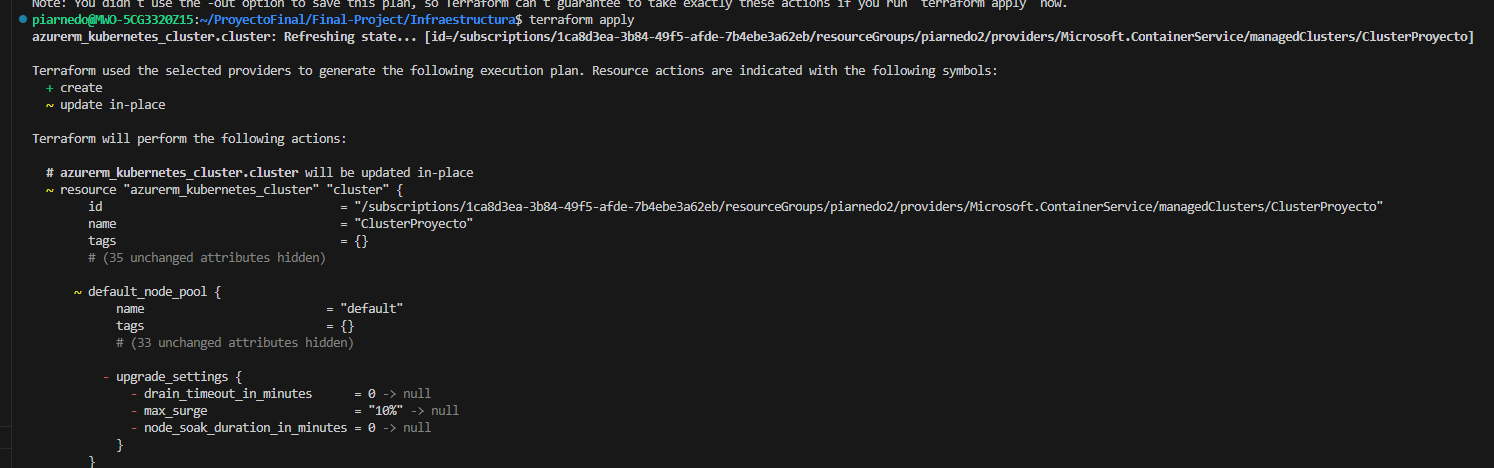
****

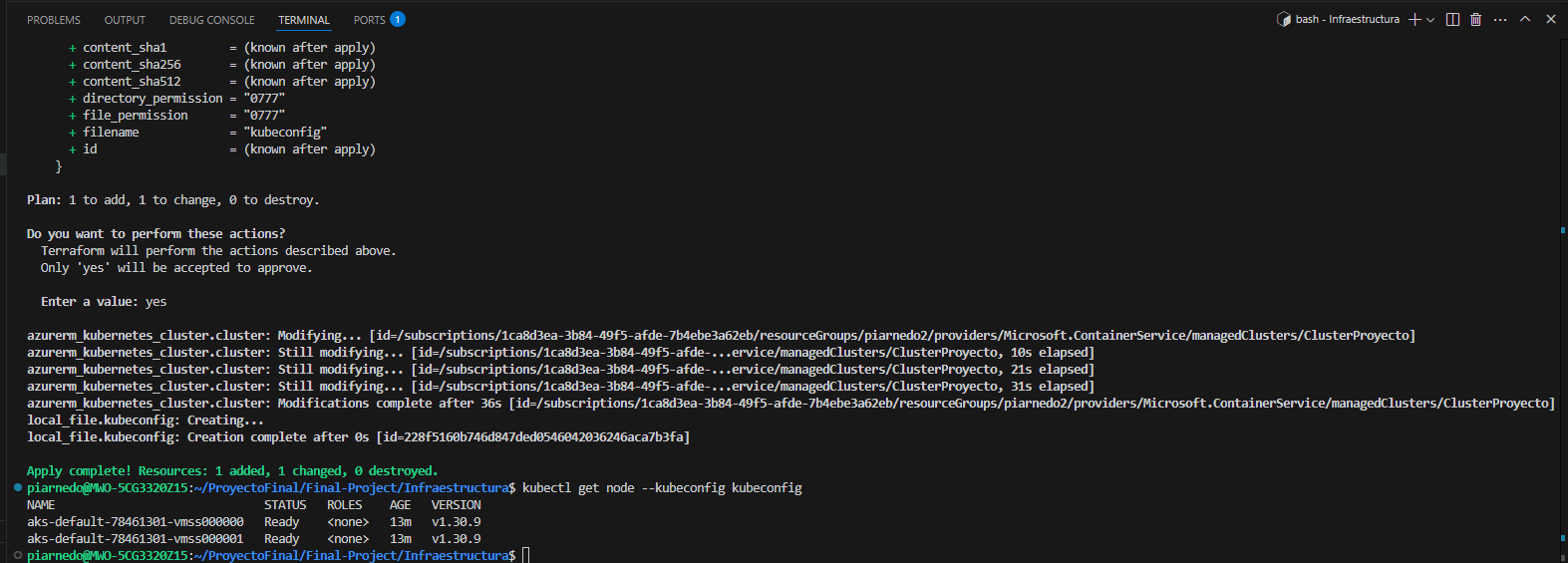
**(Para poder hacer el ingress , hay que habilitar una ip publica para los 2 nodos que se crean en esta configuración)**

**Hacer el Terraform Flow una vez tenemos esto (Se puede crear un kubeconfig para tener el cluster en local mediante el outputs.tf que te lo montará al hacer el Terraform Flow).**

****

**1-Terraform Init, Terraform plan para simular una subida y luego terraform apply para subirlo al azure (Resource Group de antes)**

****

****

1. **Dockerizar imágenes**
   1. **Creamos dockerfiles para front y back**
   2. **Texto

      El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Creamos Docker-compose para poder generar imágenes y levantar todo a la vez**

**Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.**

* **docker-compose up -d**
* **con Docker images vemos las imangenes creadas**
* **Interfaz de usuario gráfica, Texto, Aplicación

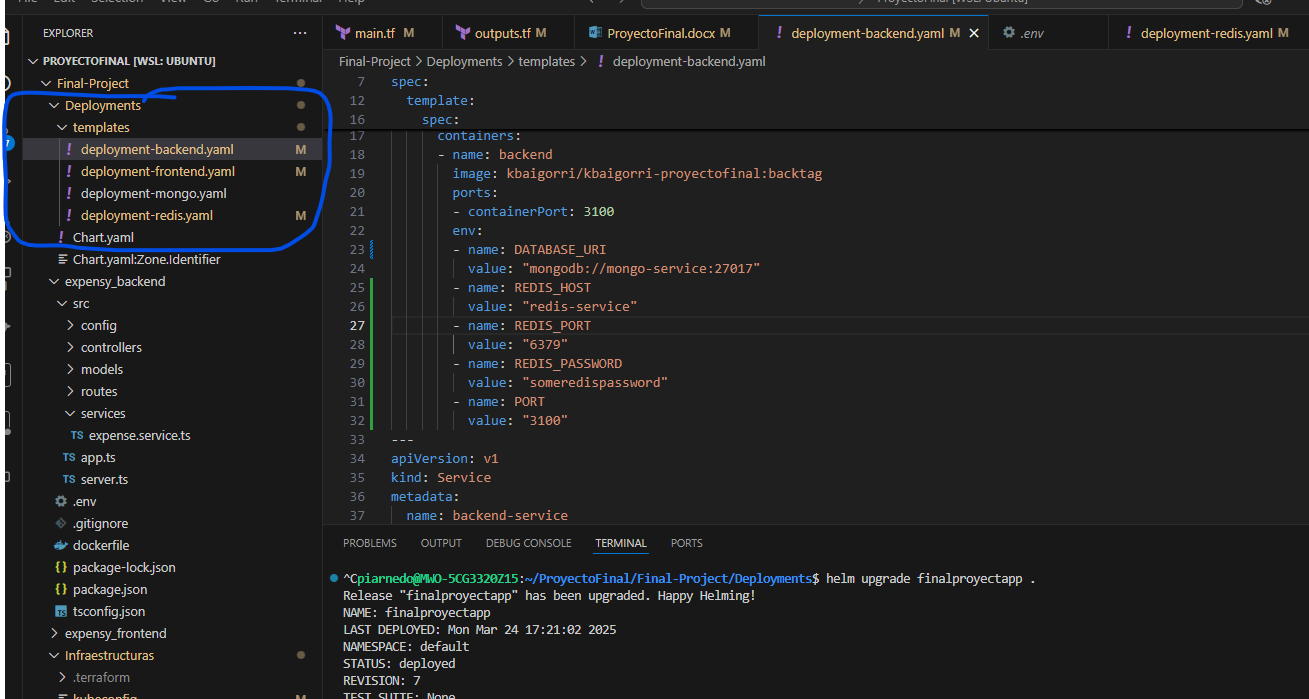
  El contenido generado por IA puede ser incorrecto.**

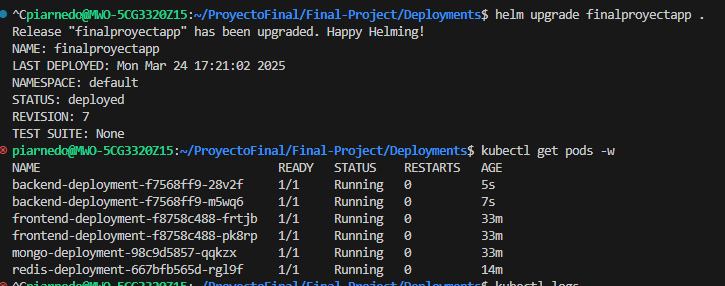
**Y las tageamos y subimo a Docker hub**

* **docker tag devops-final-project-ih-backend kbaigorri/kbaigorri-proyectofinal:backtag**
* **docker push kbaigorri/kbaigorri-proyectofinal:backtag**
* **docker tag devops-final-project-ih-frontend kbaigorri/kbaigorri-proyectofinal:frontag**
* **docker push kbaigorri/kbaigorri-proyectofinal:frontag**

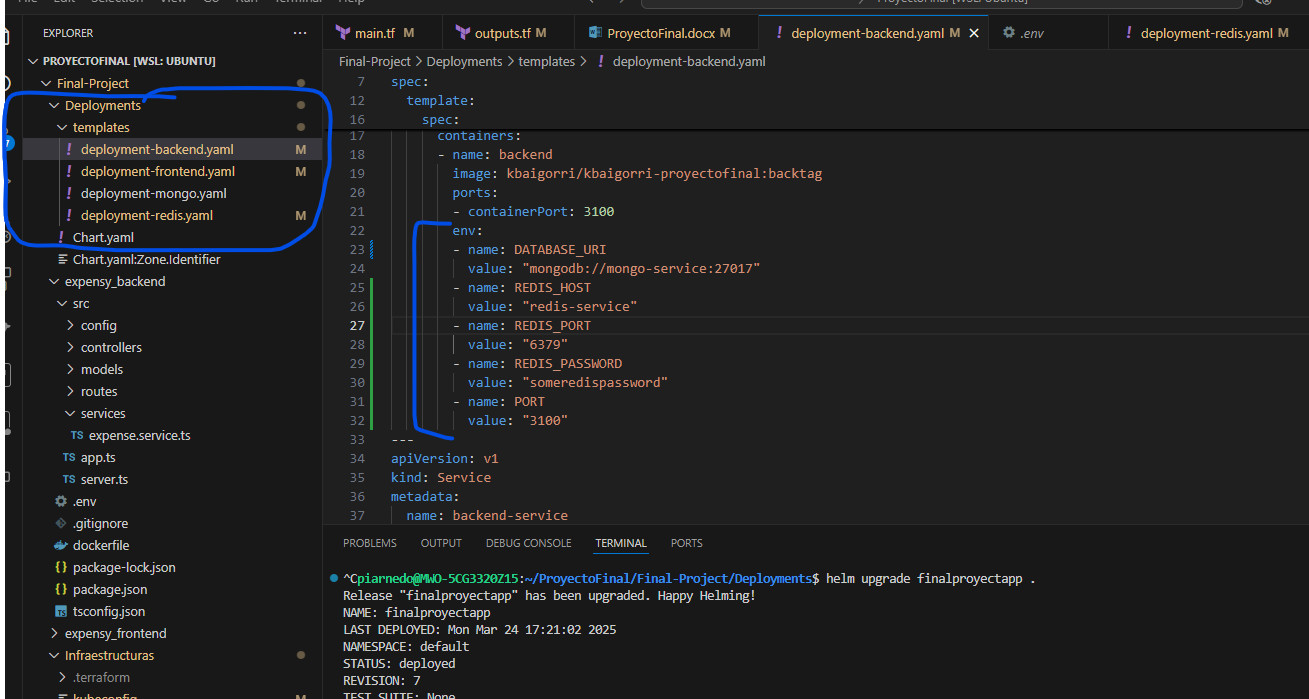
Interfaz de usuario gráfica, Texto

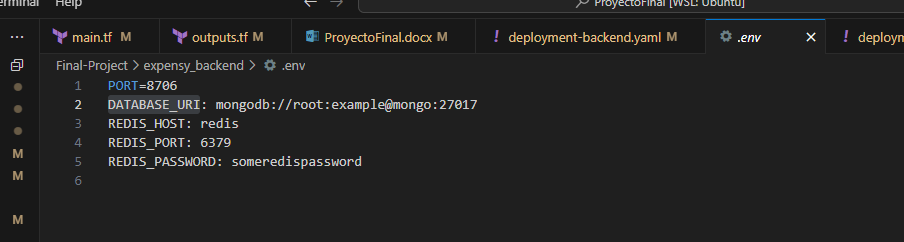
El contenido generado por IA puede ser incorrecto.

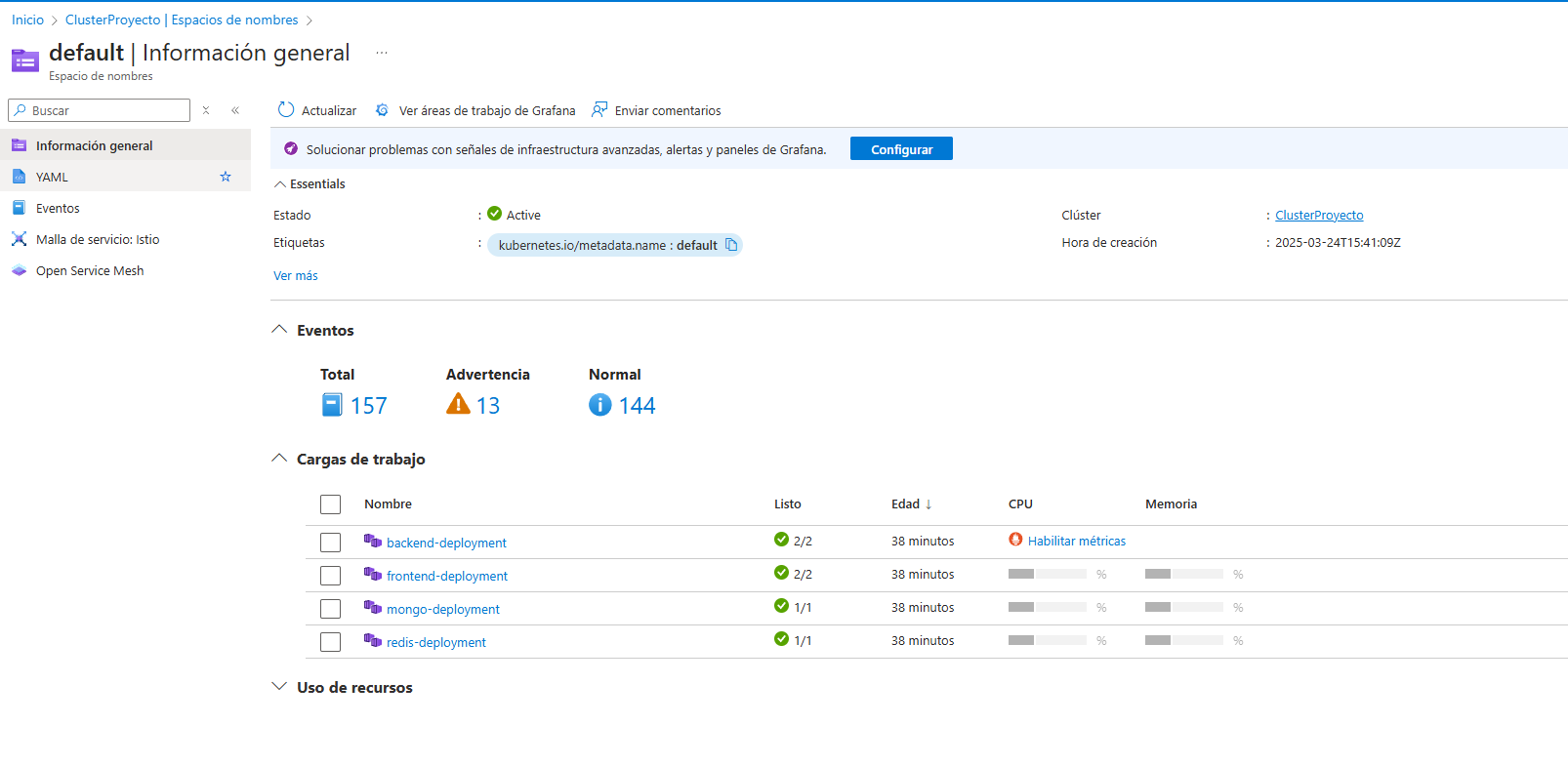
Para subir los archivos “Deploy” al aks debemos meterlos en una carpeta aislada del resto de archivos y una vez excluido del resto de archivos hacer un “helm install nombre .”. 



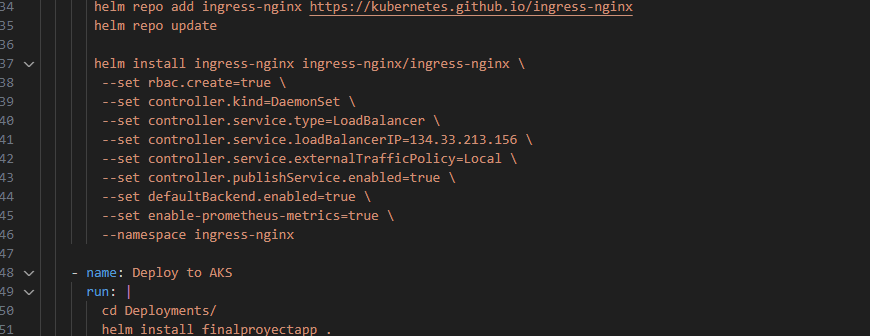
Tenemos que tener en cuenta la contenido del archivo de configuración “.env” que tiene configuaradas las bd del mongodb y redis.







Para hacer el ingress , tendremos que crear el namespace de tipo ingress-nginx con kubectl, luego realizar la siguiente sucesión para instalar y configurar ingress en nuestro cluster de AKS ->

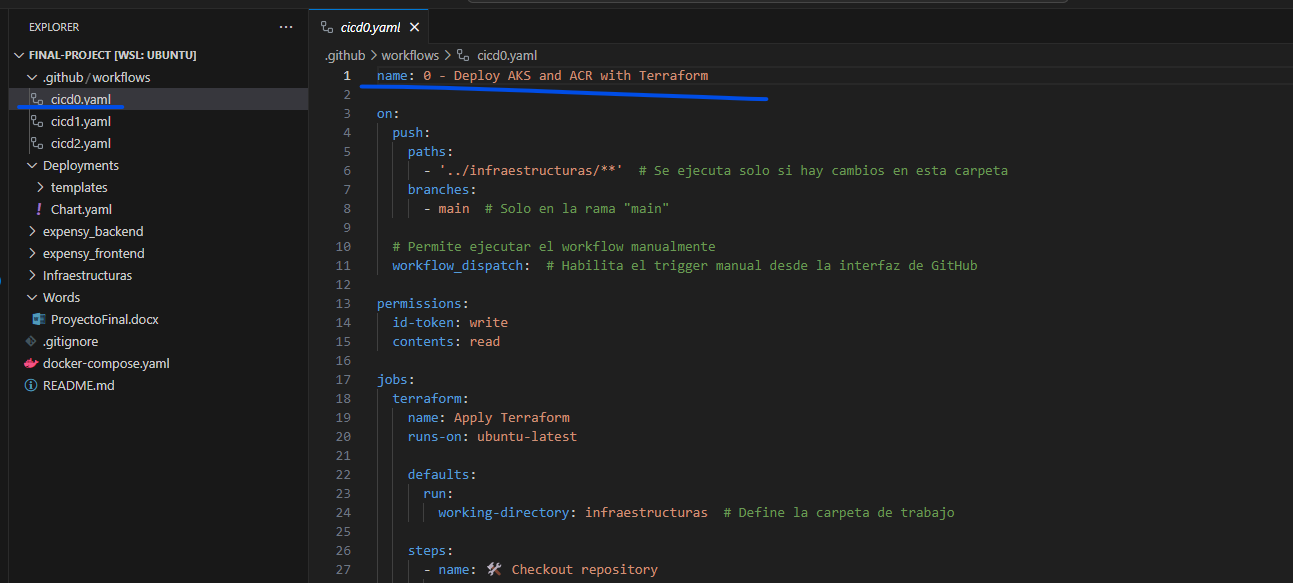


 En nuestro caso lo asociamos a la parte visual de la app que es el frontend pero no funciono puesto que las ips publicas que asociamos no eran accesibles y causaban problemas.

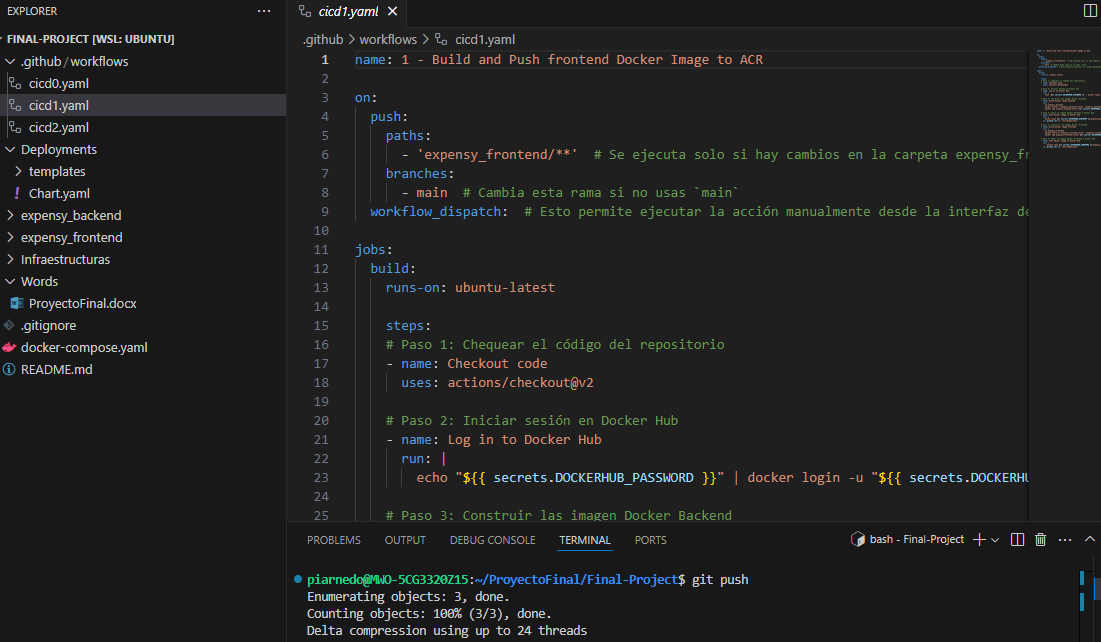
Una vez construido el ingress y viendo que se levantaba todo, iniciamos la parte de automatización de la ejecución de las distintas partes que hemos ido construyendo a los largo del curso desde github actions.

Nosotros lo aplicamos desde una estructura montada desde el código del proyecto para que se pudieran ejecutar de manera automática una vez pusheado los cambios al github.

En primer lugar crear la estructura y el clustes en el azure utilizando los comandos de terraform ->



En segundo lugar, crear las imágenes del frontend / backend , mongo y redis y subirlo a la cuenta de dockerhub. Se usan variables de entorno de github.



Por último, hemos llegado a realizar la parte del despliegue del proyecto al AKS que construimos en el primer paso y que por lo tanto realizamos el paso de la relación del frontend con el ingress.

