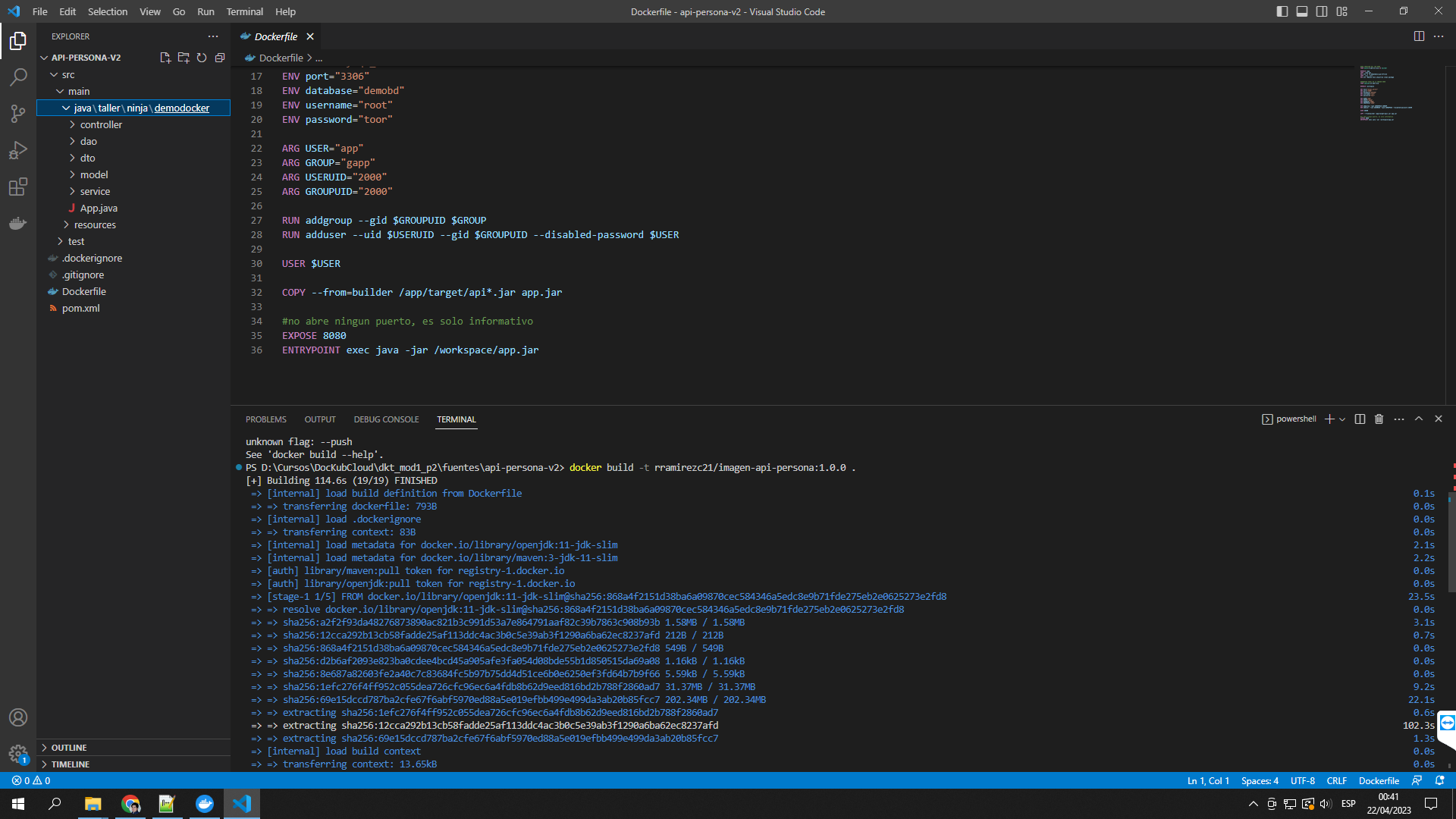
MITOCODE NETWORK

Solución al examen Final del Curso: **Docker Kubernetes Cloud**

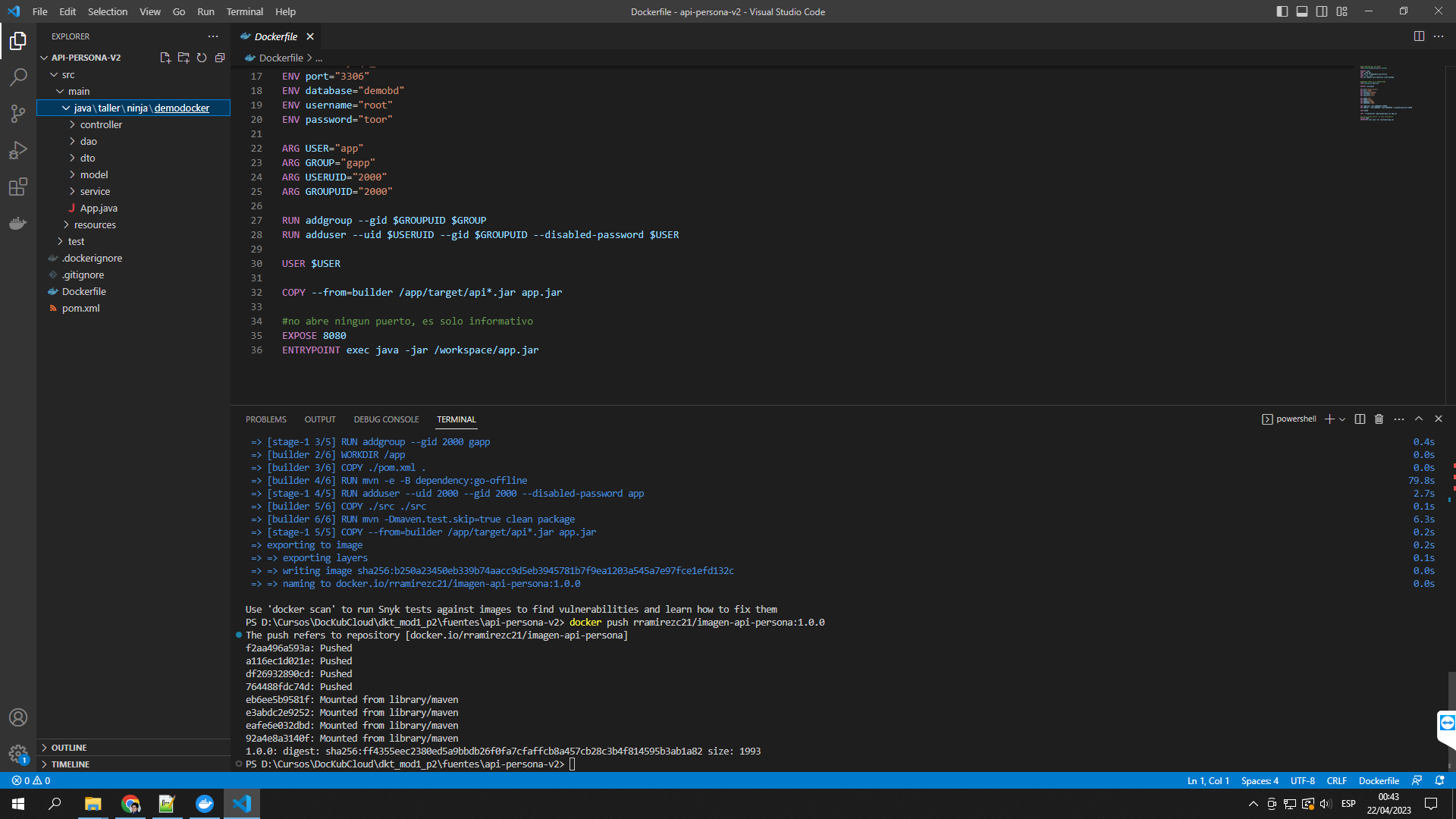
Alumno: **Ramirez Caceres Robert Jhon**

1. Crear una imagen pública del aplicado en Docker Registry

Evidencia de la construcción de la imagen

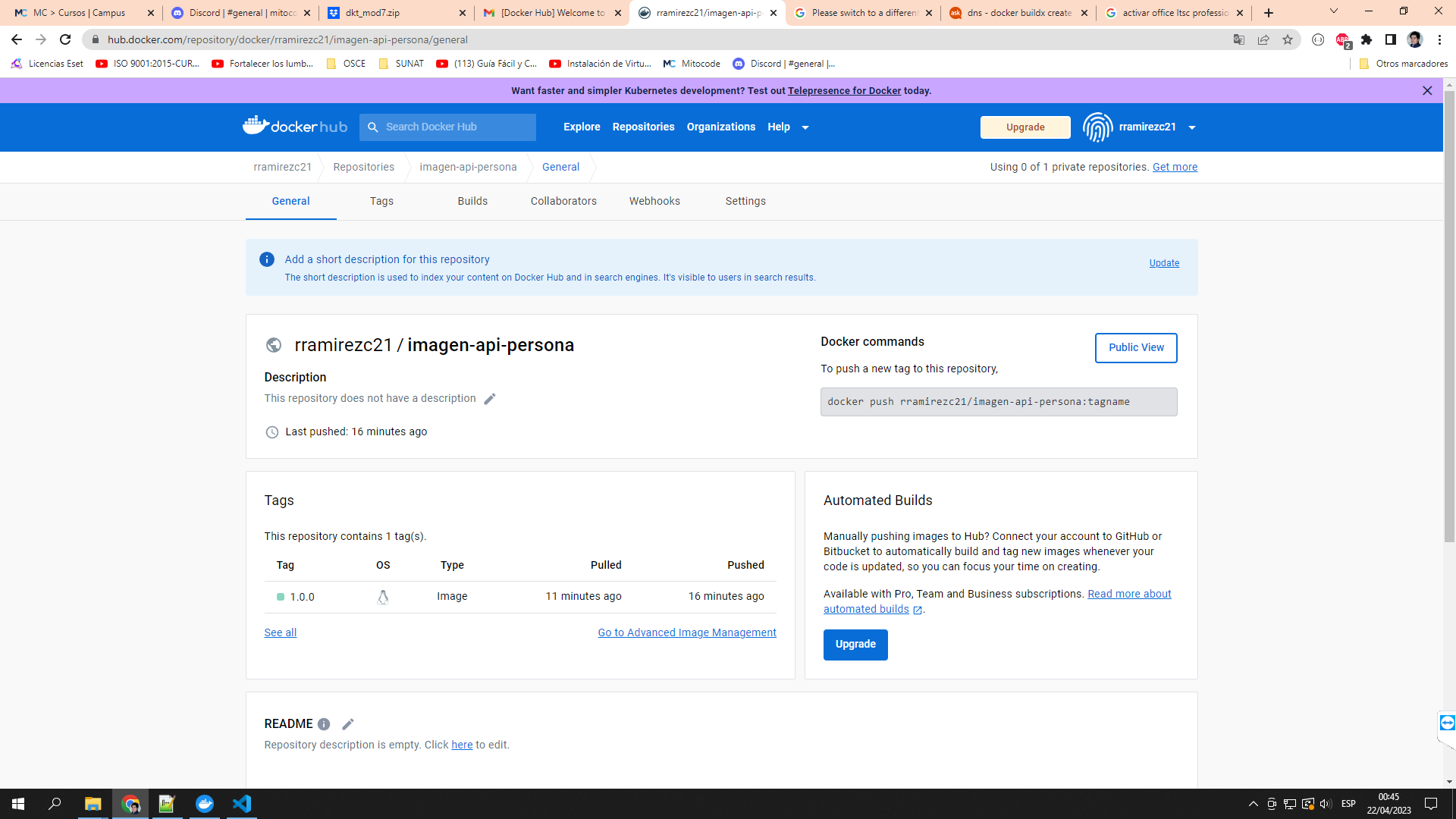


Evidencia de la subida de la imagen a Docker Hub



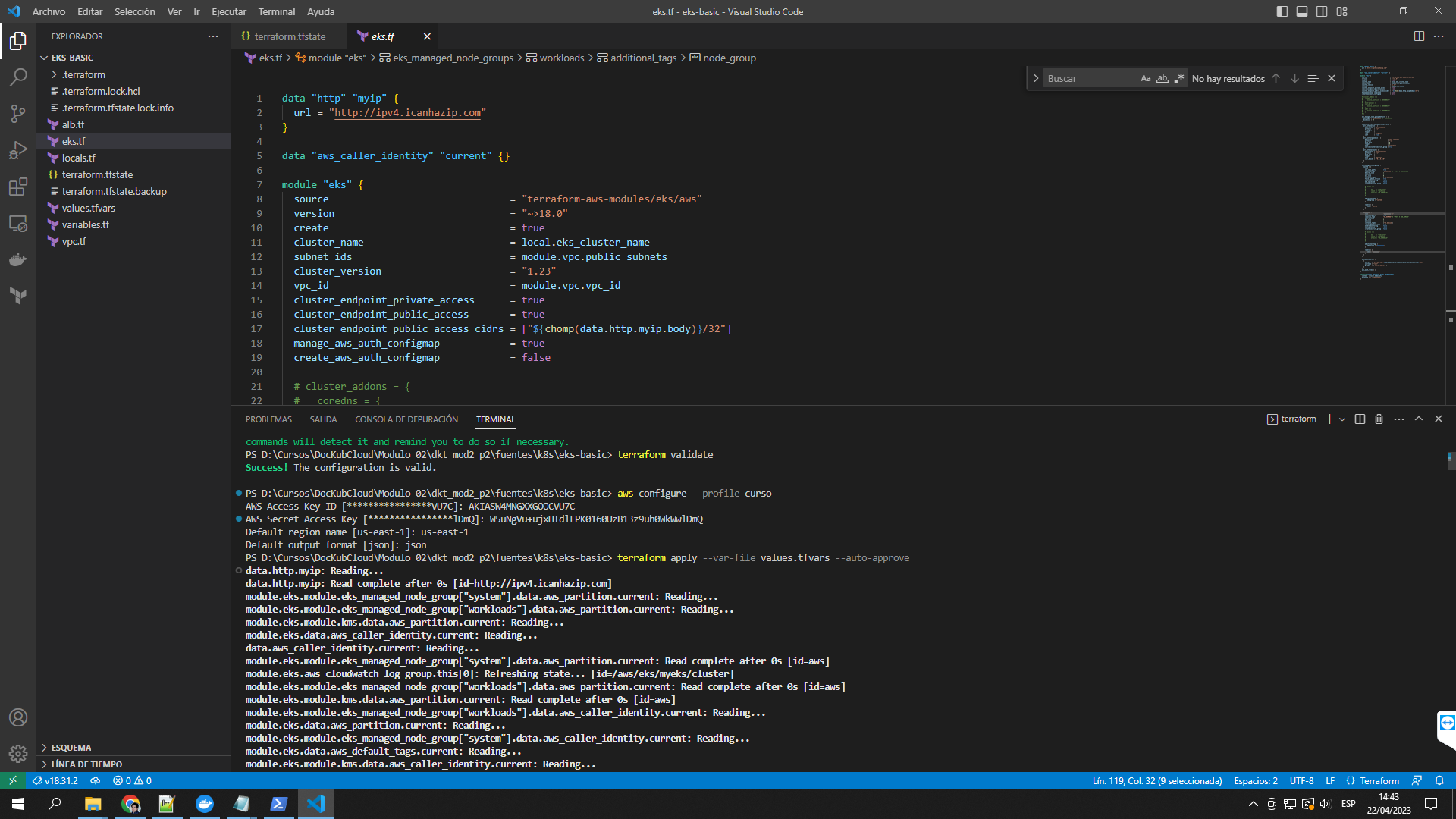
Evidencia de la imagen en Docker Hub

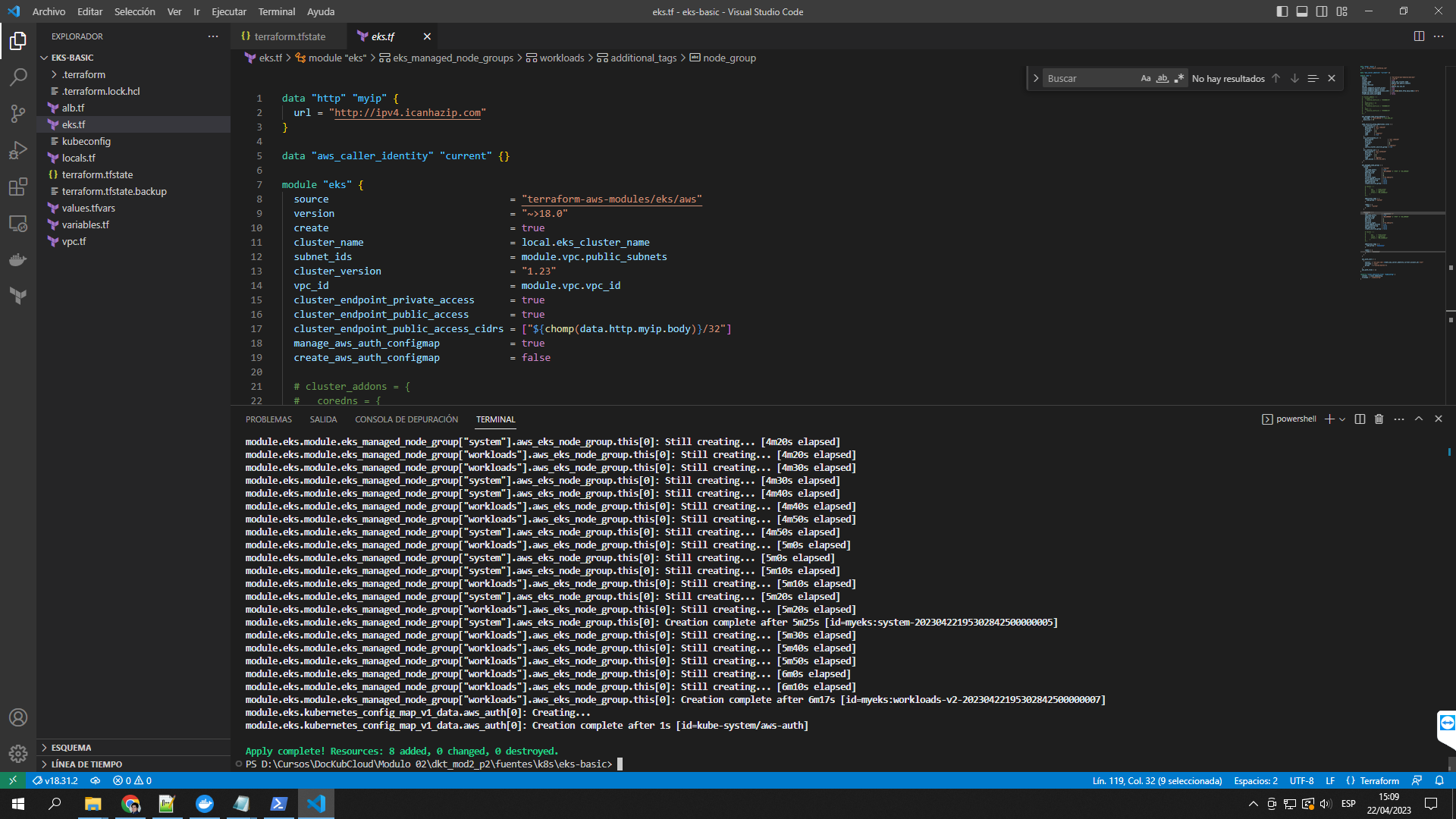
docker push rramirezc21/imagen-api-persona:tagname



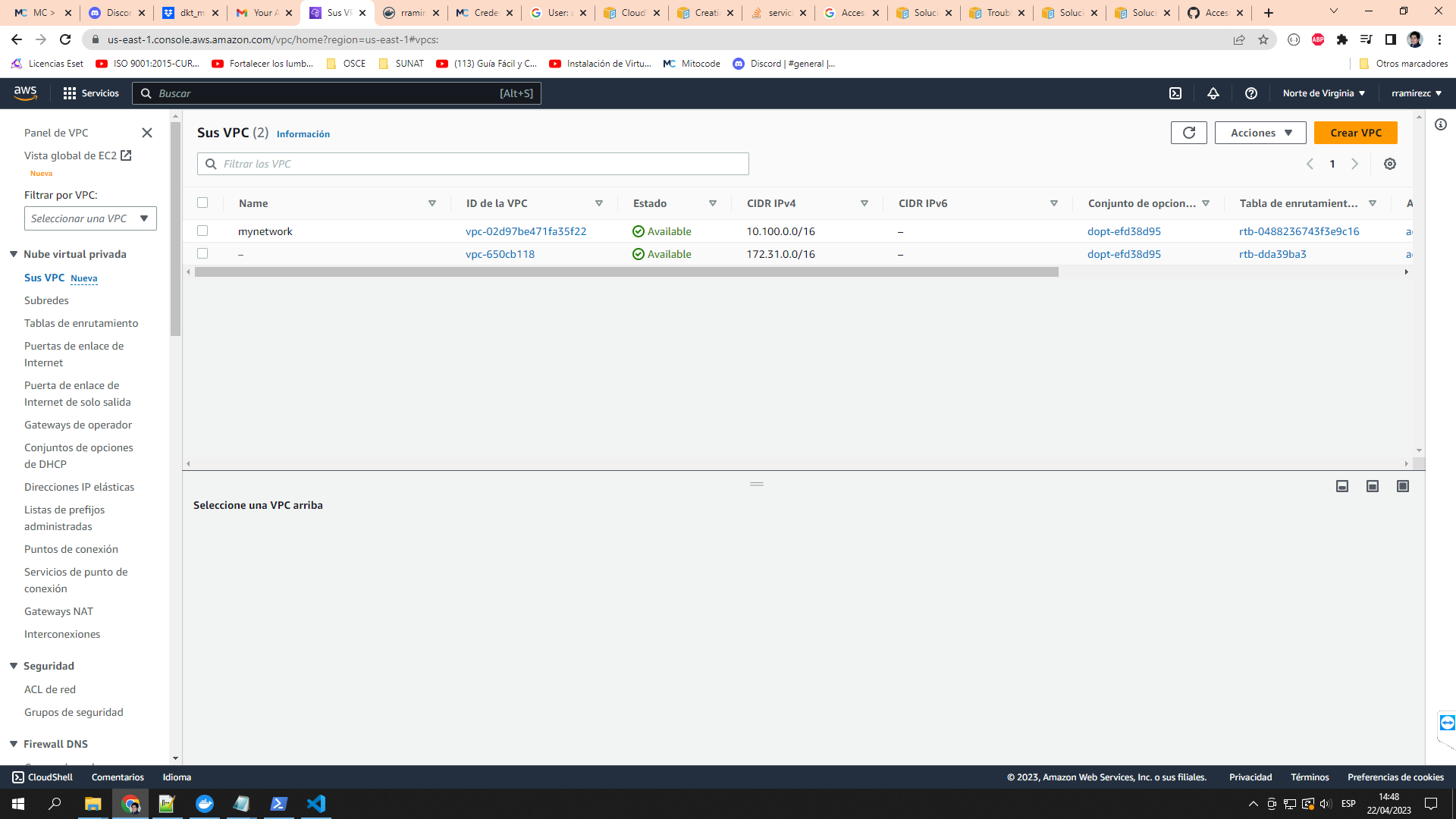
1. Crear un cluster de Kubernetes

El cluster de Kubernetes se ha creado en **AWS**

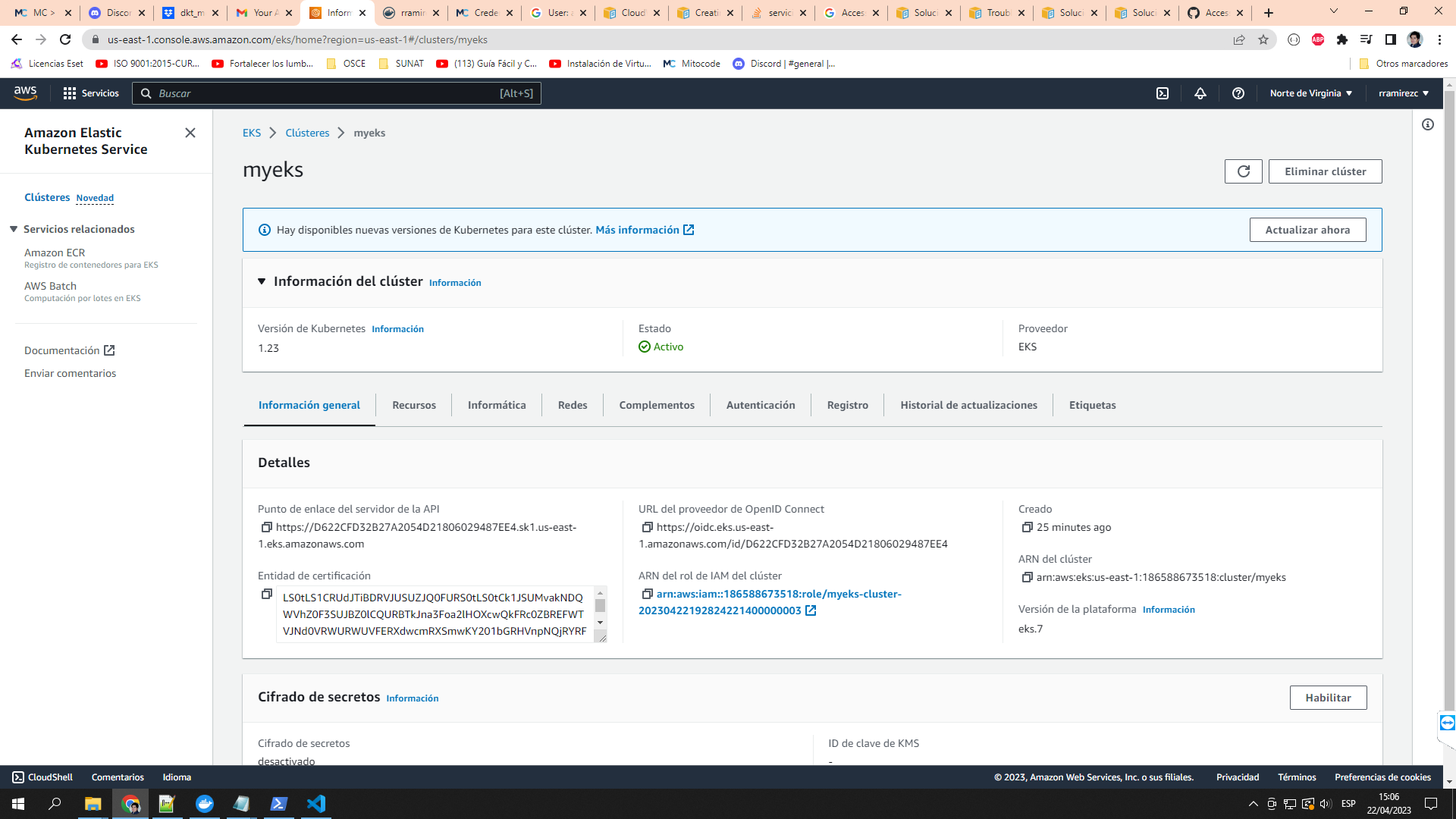




Red VPC



Cluster Kubernetes

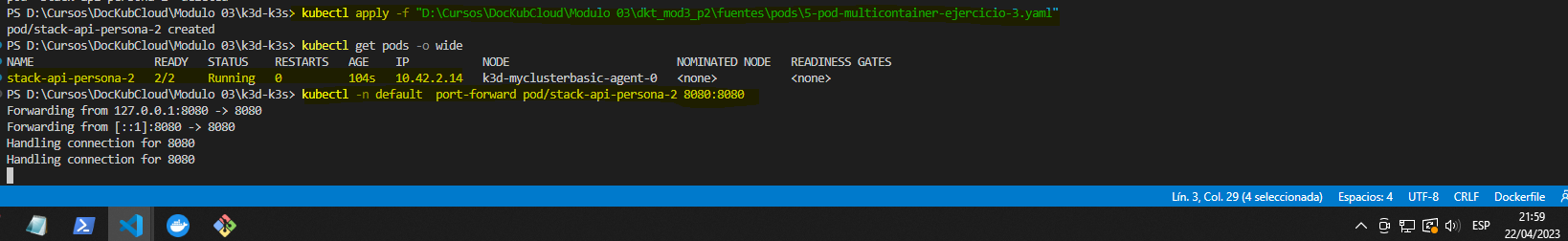


Configuración del clúster Kubernetes

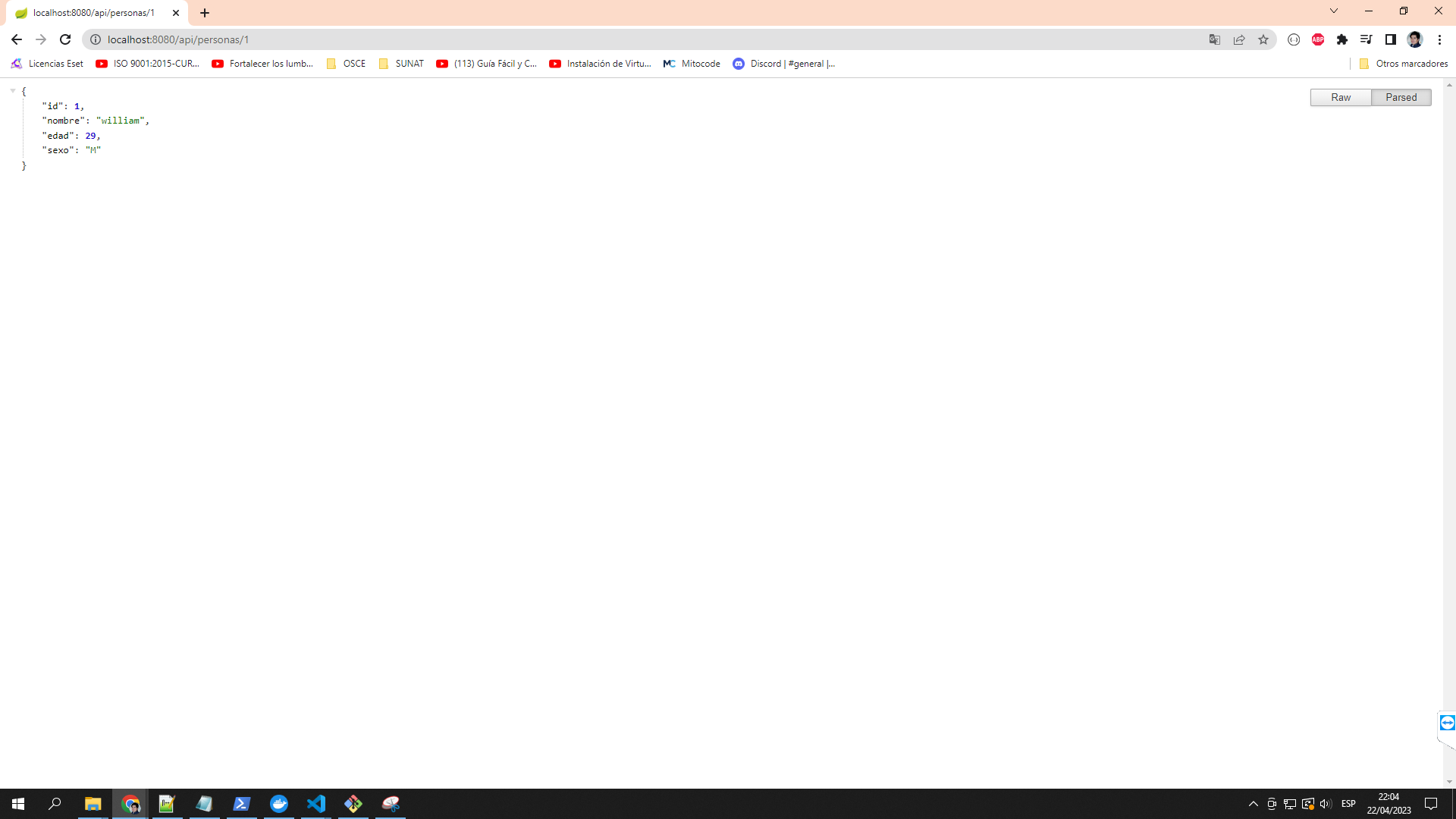


1. Crear un Pod simple (dentro del Namespace default) que ejecute el aplicativo y la base de datos MySQL realizando la comunicación por medio de localhost

Creacion del pod



Prueba invocación al servicio

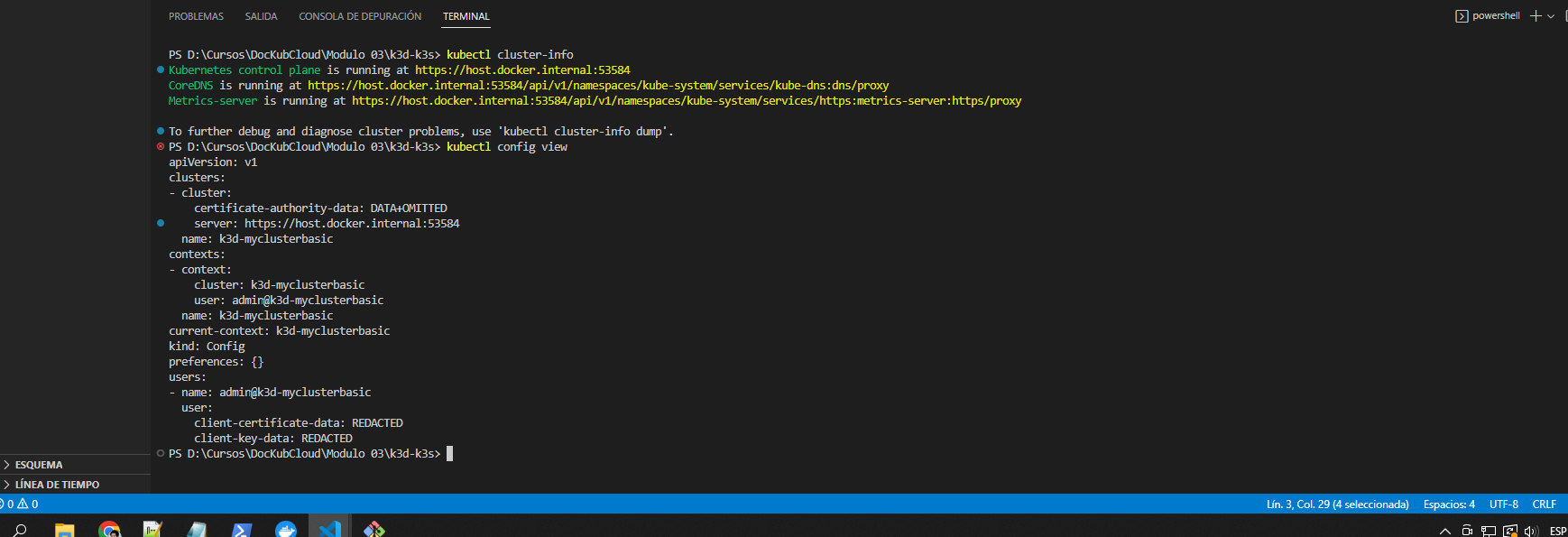


Yaml desplegado

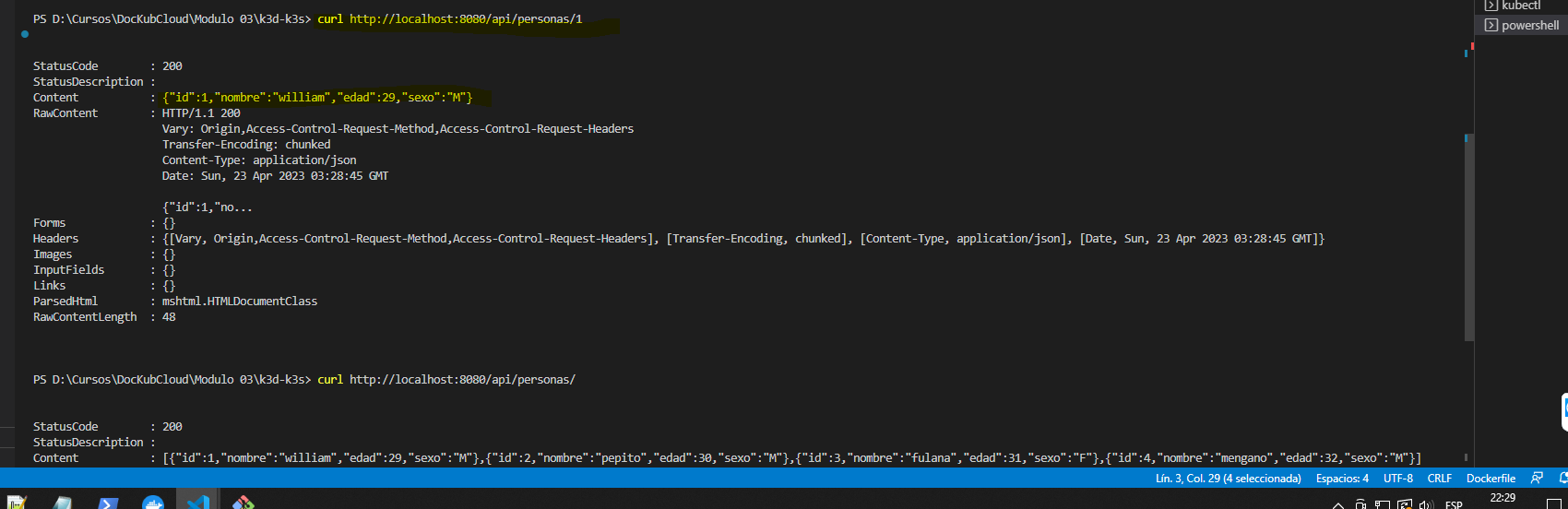


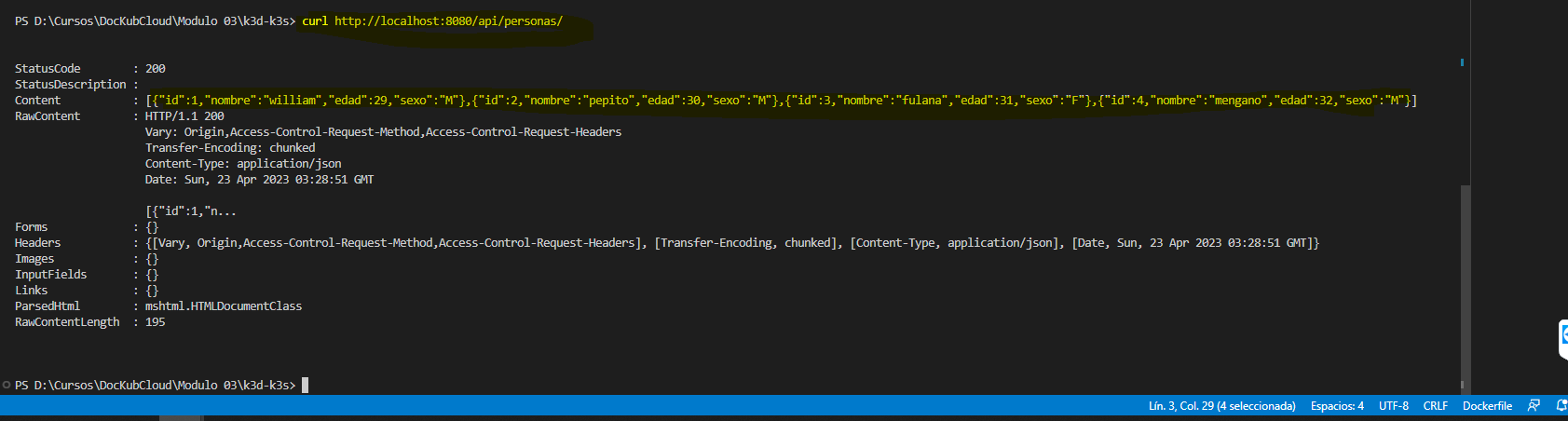
1. Realizar pruebas de funcionamiento mediante un Pod que contenga un contenedor que permita hacer un curl a los métodos del aplicativo

Información del clúster creado:



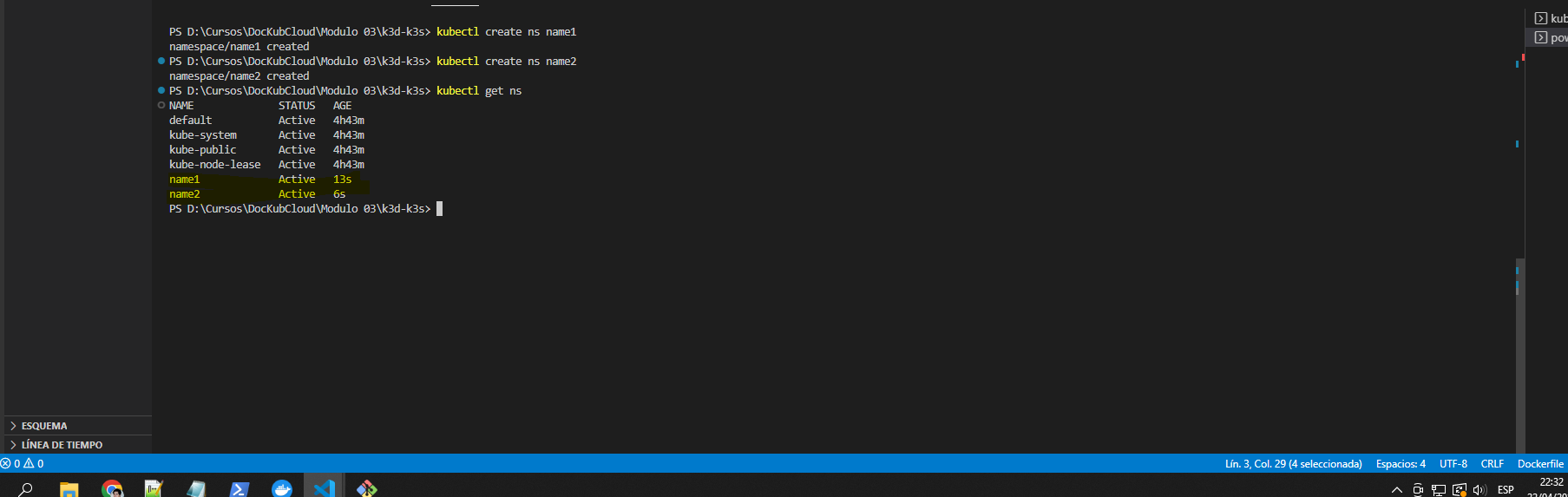
Pruebas de funcionamiento del pod:





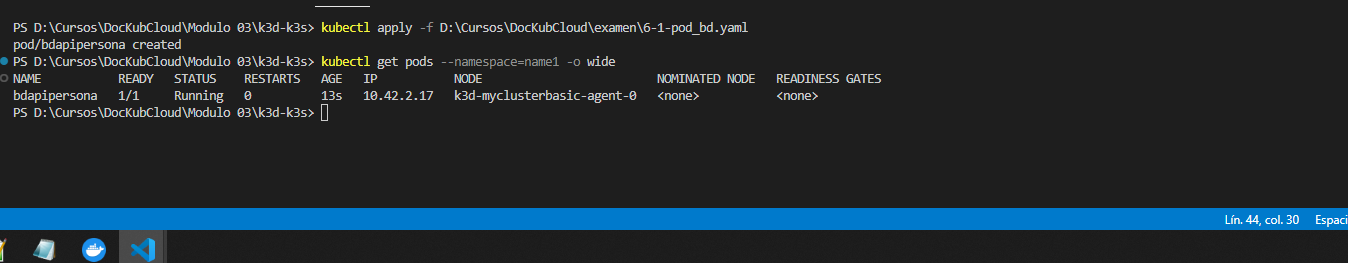
1. Crear dos Namespaces llamados name1 y name2

Creación de los namespaces



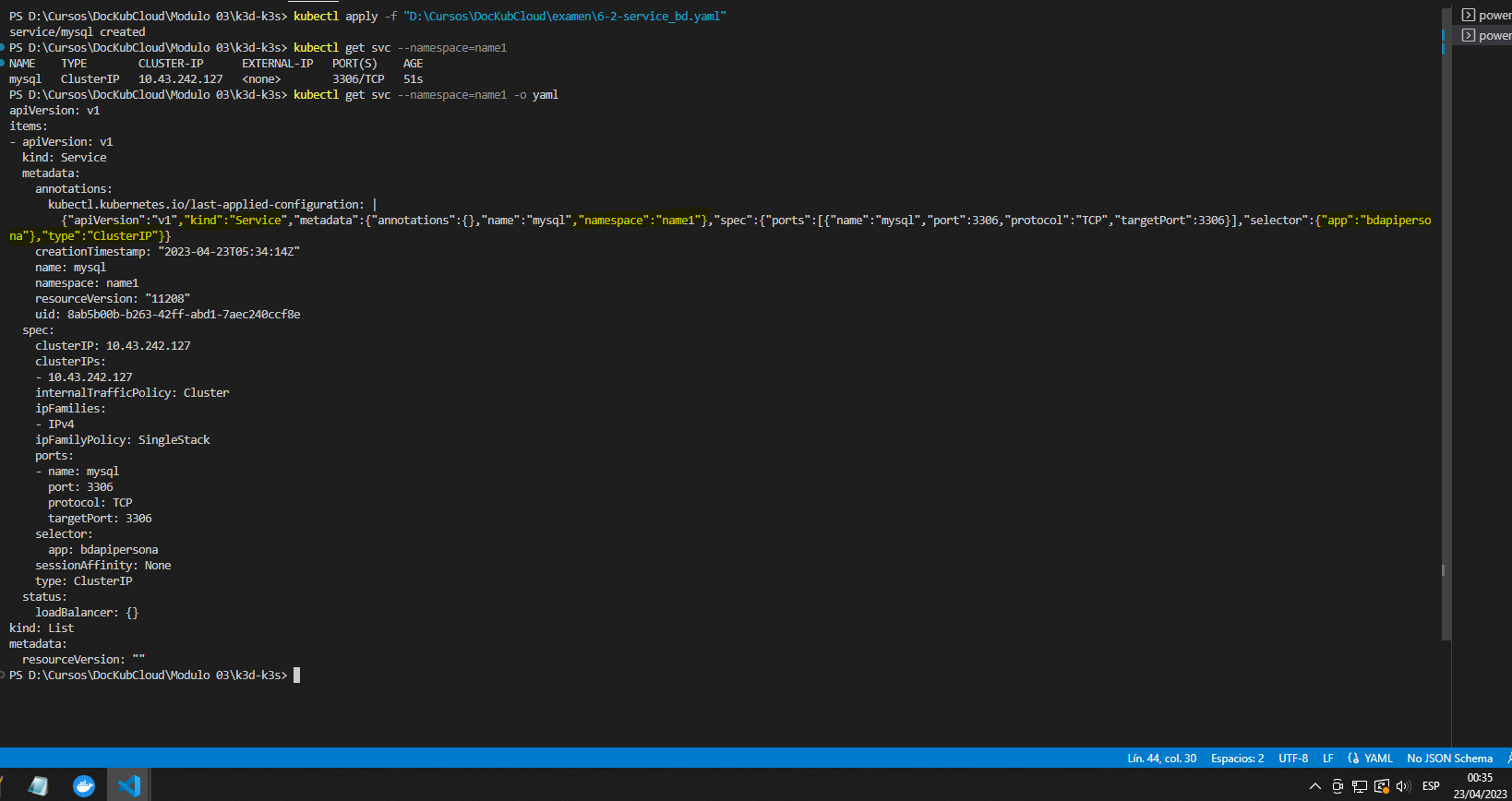
1. En el Namespaces name1 realizar la siguiente implementación:
2. Crear la base de datos en un Pod

kubectl apply -f D:\Cursos\DocKubCloud\examen\6-1-pod\_bd.yaml



1. Crear un Service de tipo ClusterIP que permita exponer la base de datos

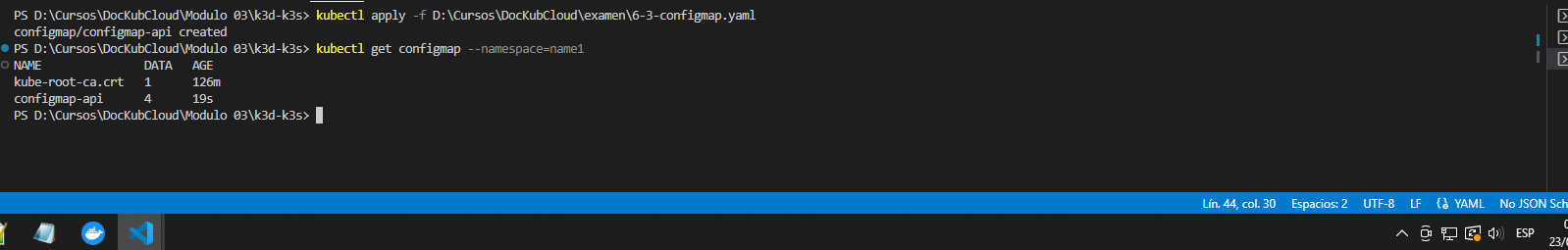
kubectl apply -f "D:\Cursos\DocKubCloud\examen\6-2-service\_bd.yaml"



1. Crear un ConfigMap que permita desacoplar la configuración del aplicativo

Creación del configmap:

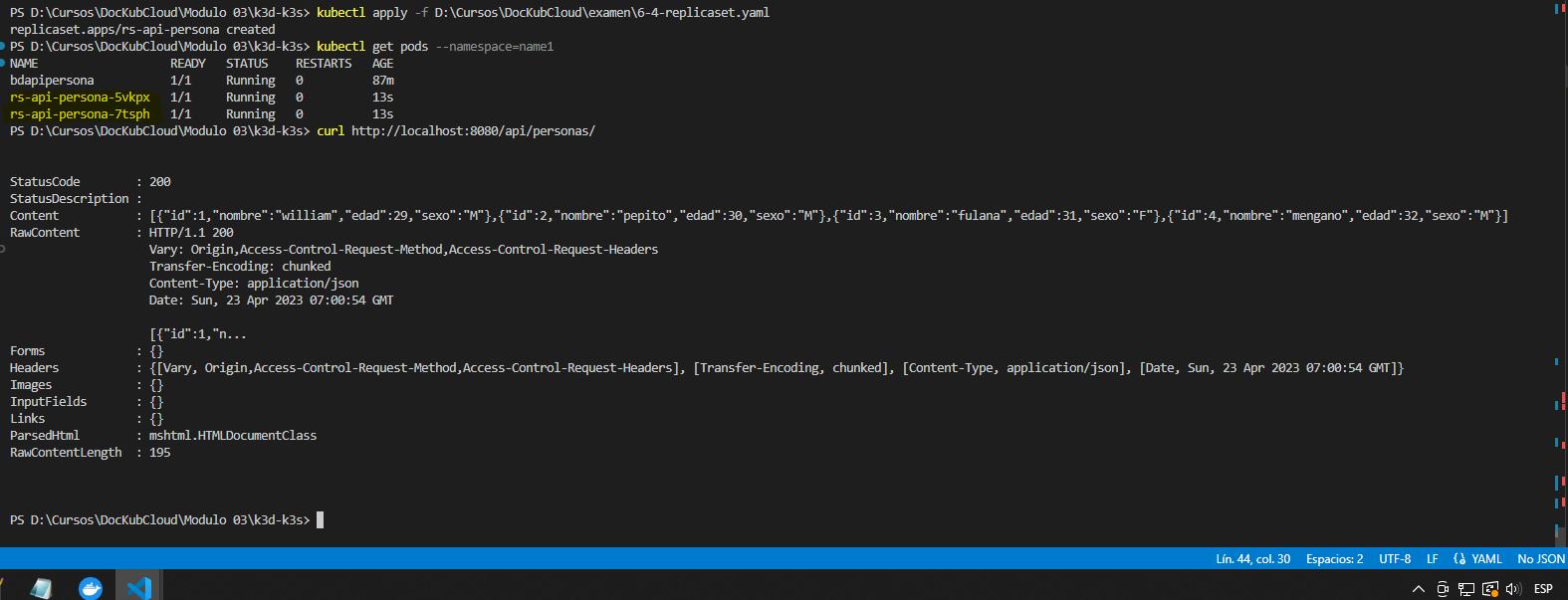
kubectl apply -f D:\Cursos\DocKubCloud\examen\6-3-configmap.yaml



1. Crear la aplicación en un ReplicaSet con 2 replicas que use el ConfigMap y se conecte al Service de la base de datos

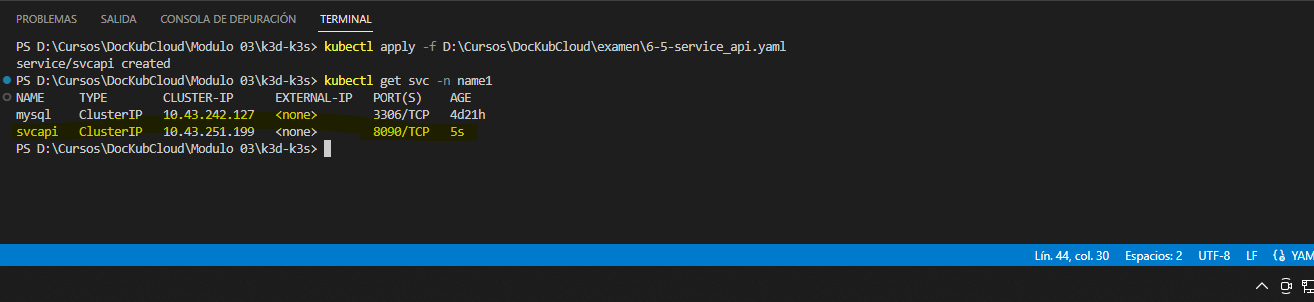
Creacion de la ReplicaSet

kubectl apply -f D:\Cursos\DocKubCloud\examen\6-4-replicaset.yaml



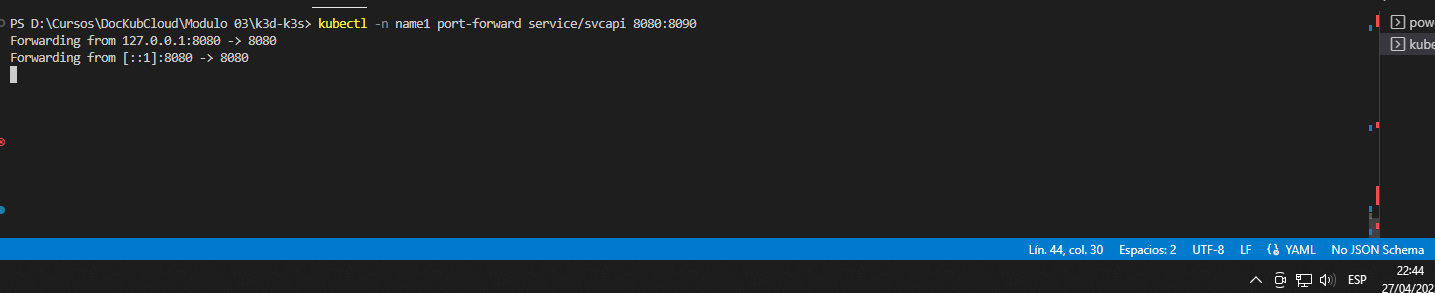
1. Crear un Service de tipo ClusterIP que permita exponer el aplicativo

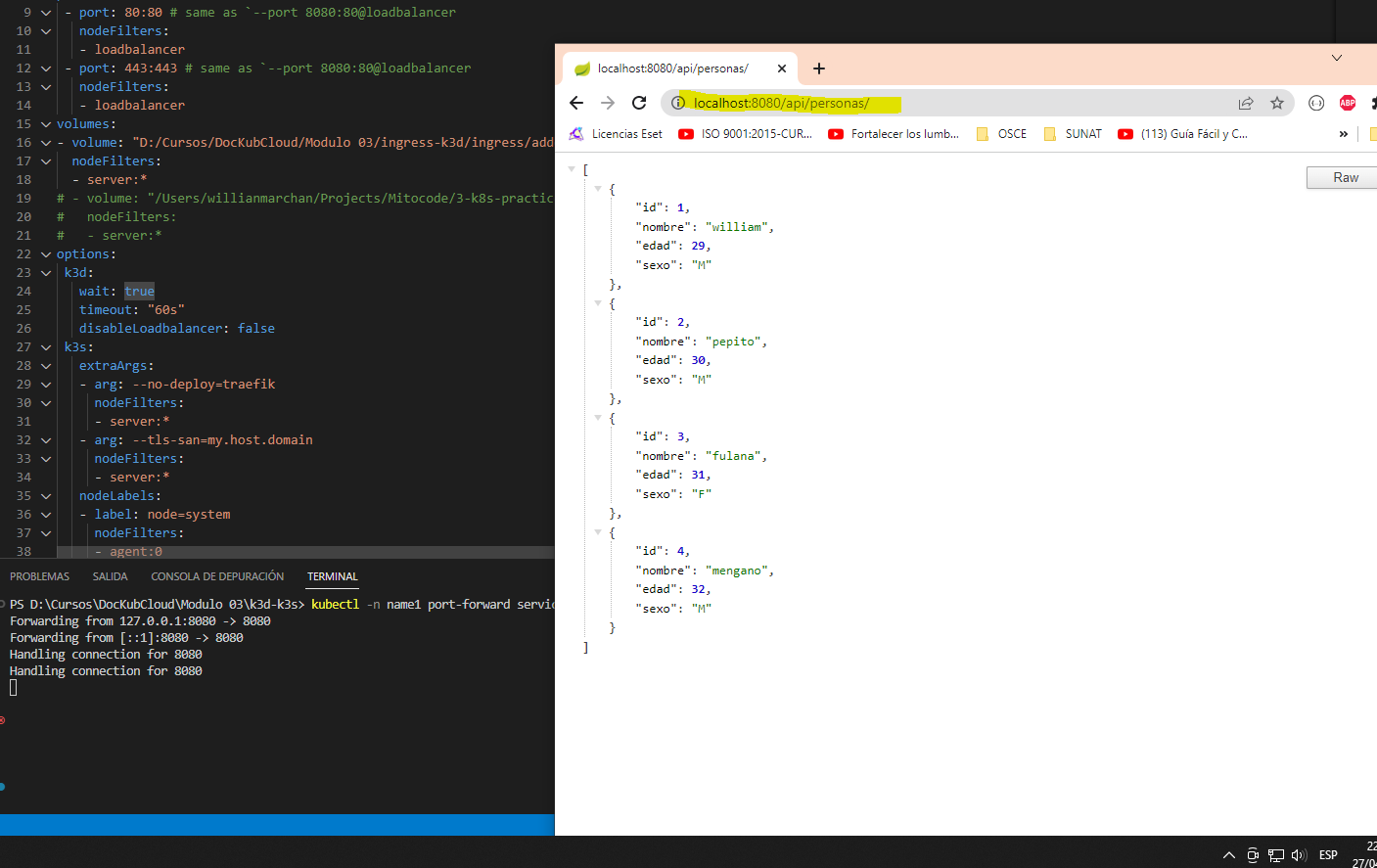
kubectl apply -f D:\Cursos\DocKubCloud\examen\6-5-service\_api.yaml

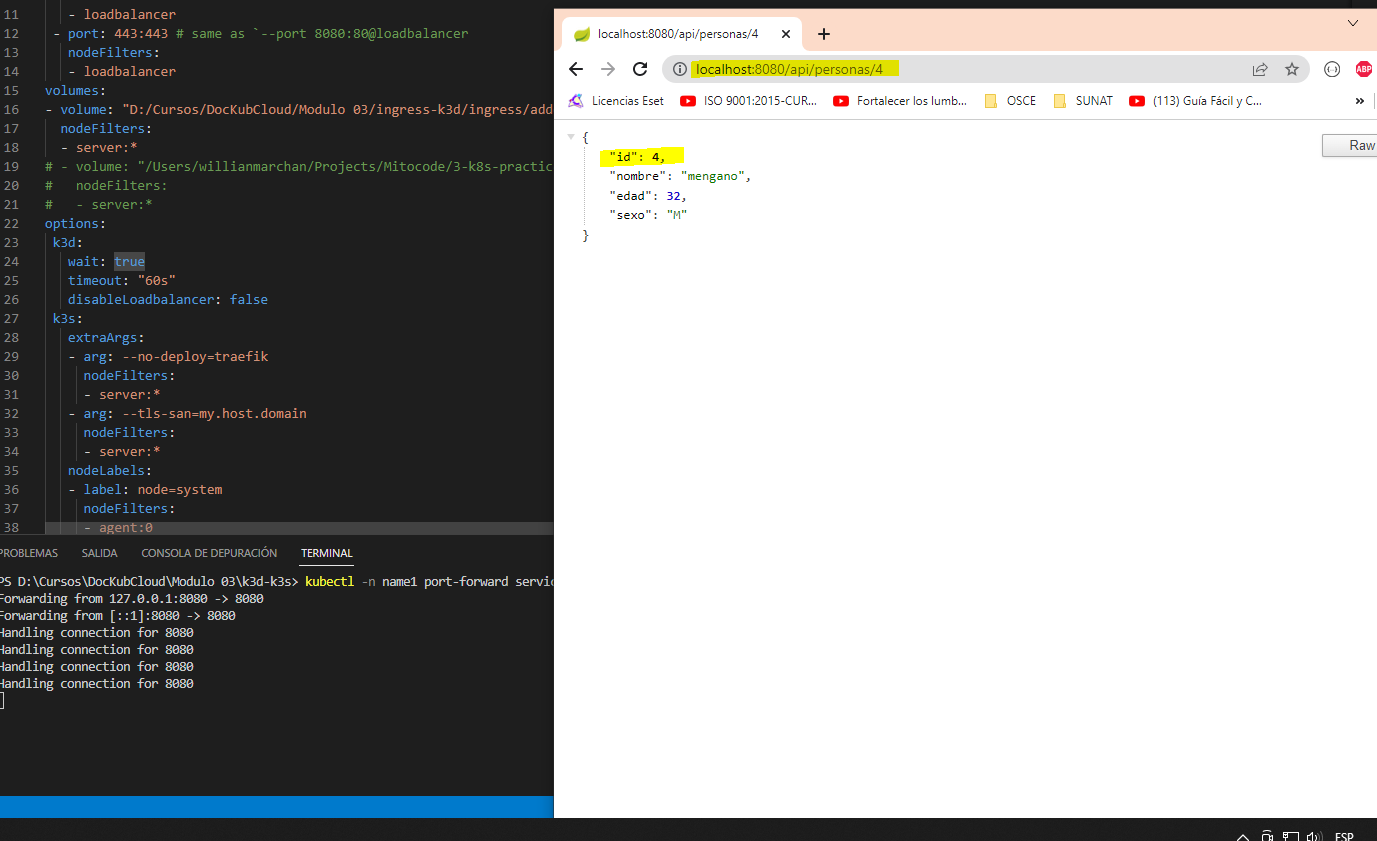


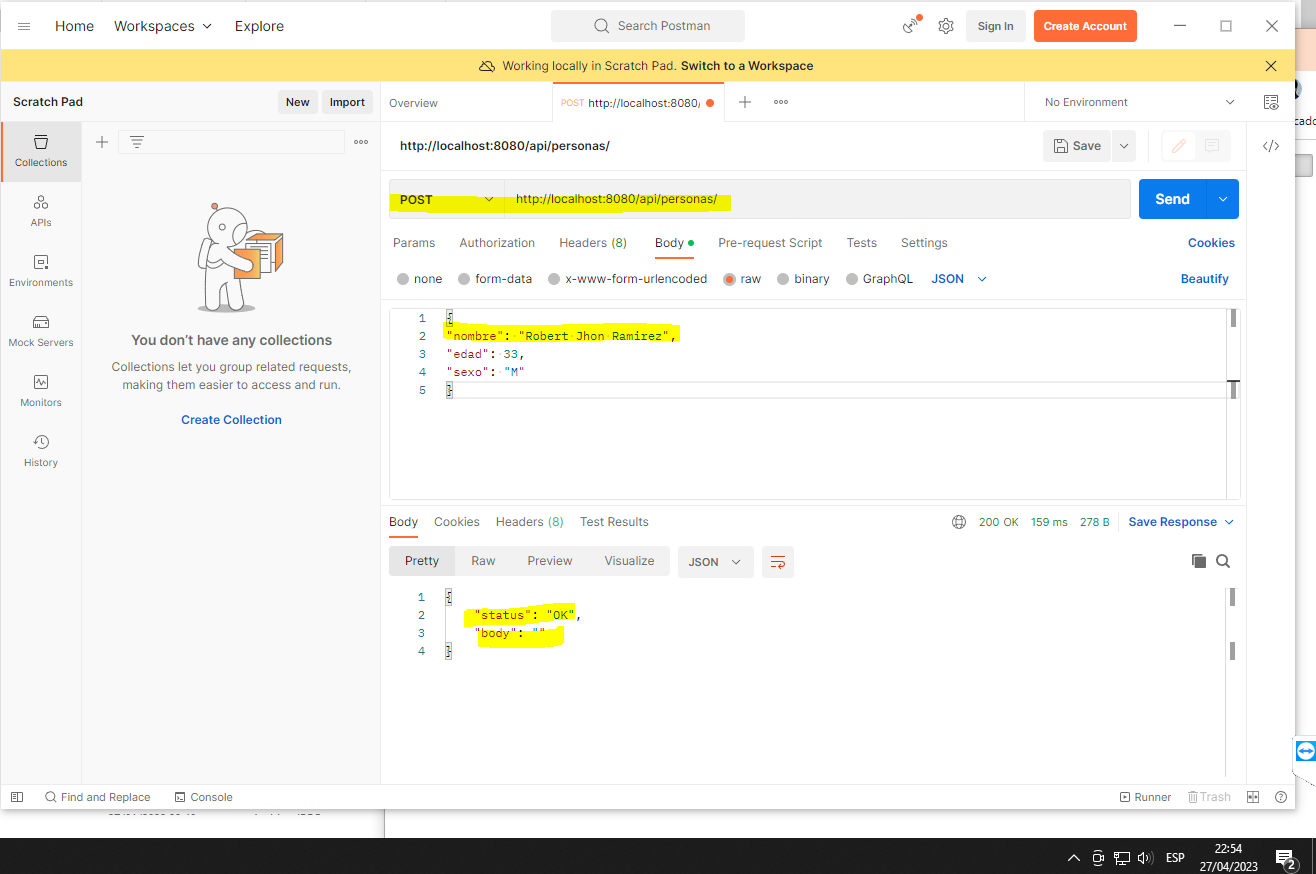
1. Realizar pruebas de funcionamiento mediante un Pod que contenga un contenedor que permita hacer un curl a los métodos del aplicativo

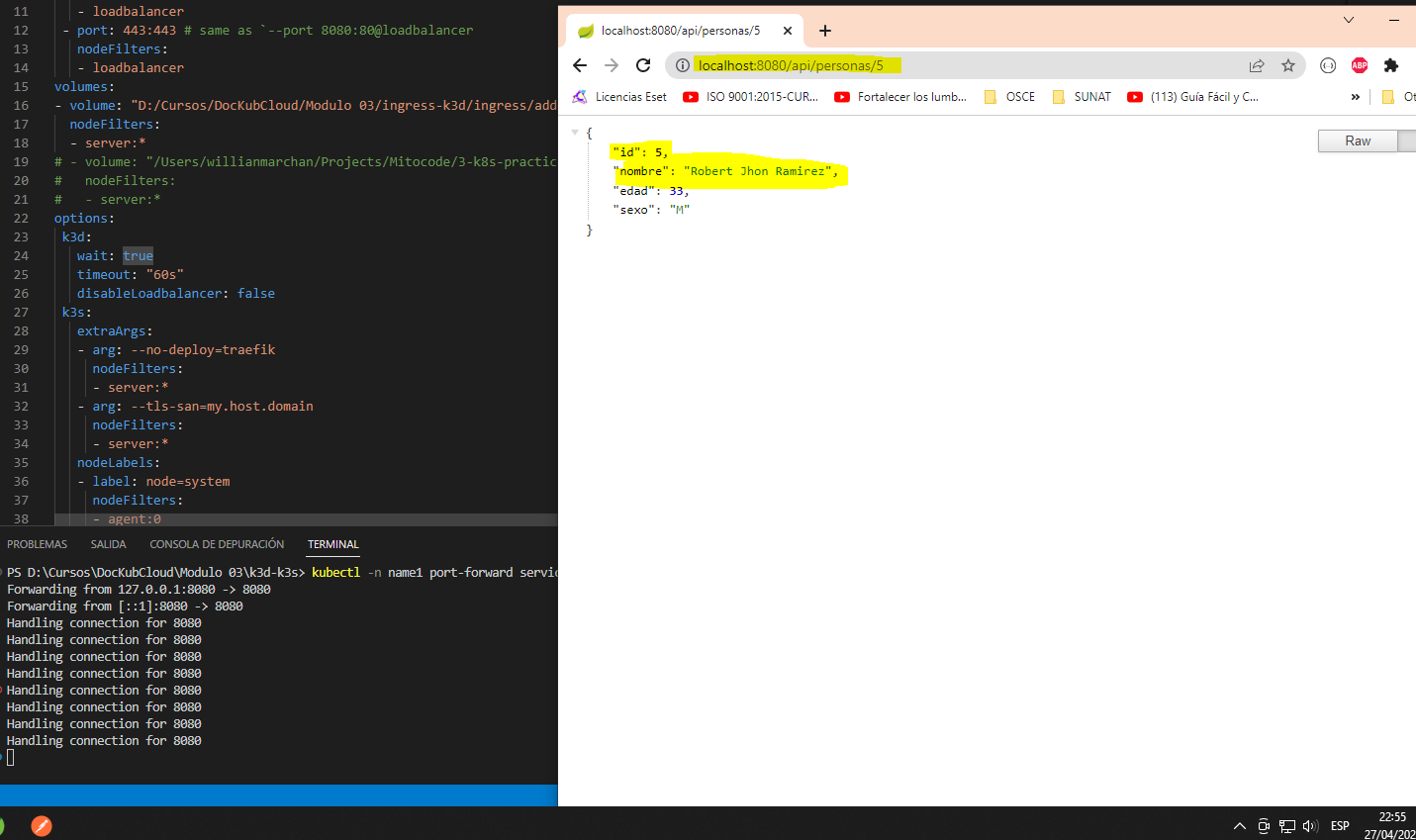
kubectl -n name1 port-forward service/svcapi 8080:8090





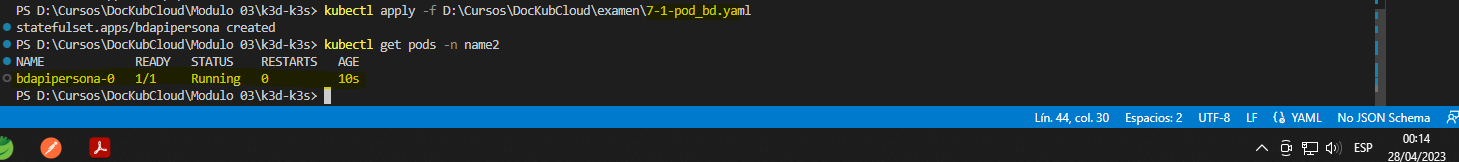






1. En el Namespaces name2 realizar la siguiente implementación:
2. Crear la base de datos en un Statefulset

kubectl apply -f D:\Cursos\DocKubCloud\examen\7-1-pod\_bd.yaml

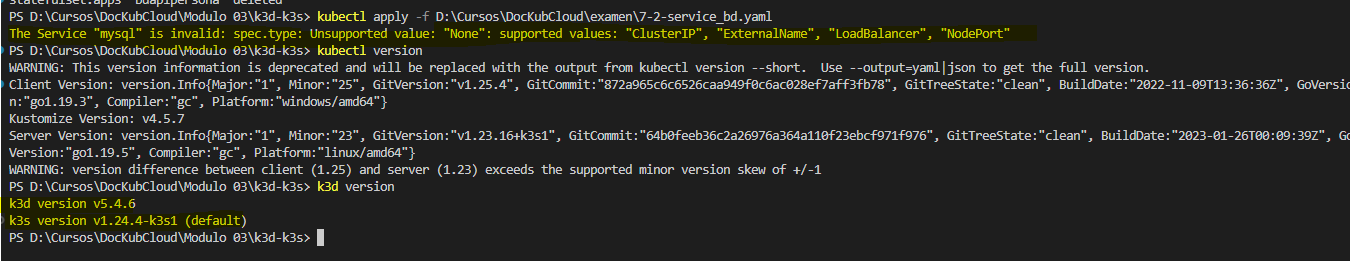


1. Crear un Service de tipo headless que permita exponer la base de datos

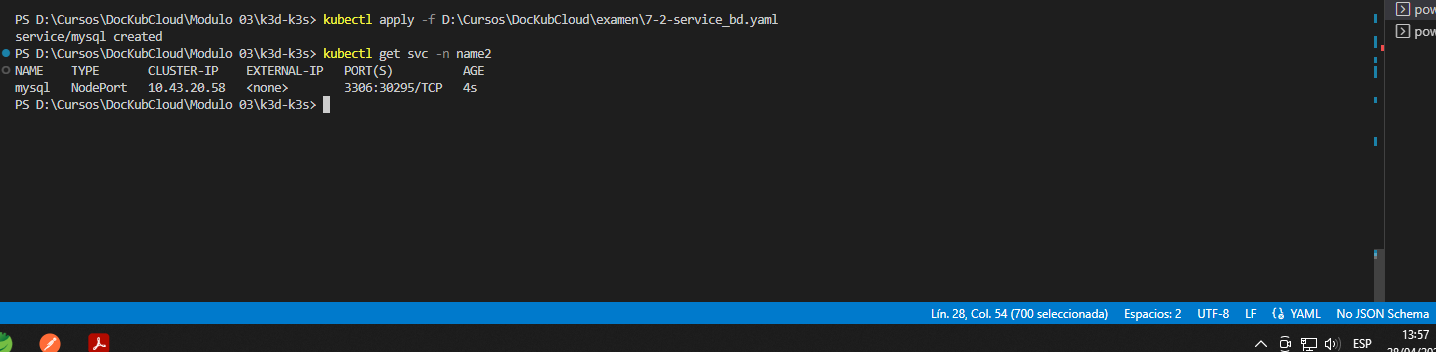
Al crear el service de tipo None(Headless) sale un error porque las versión instalada de k3d no la soporta.

kubectl apply -f D:\Cursos\DocKubCloud\examen\7-2-service\_bd.yaml

The Service "mysql" is invalid: spec.type: Unsupported value: "None": supported values: "ClusterIP", "ExternalName", "LoadBalancer", "NodePort"

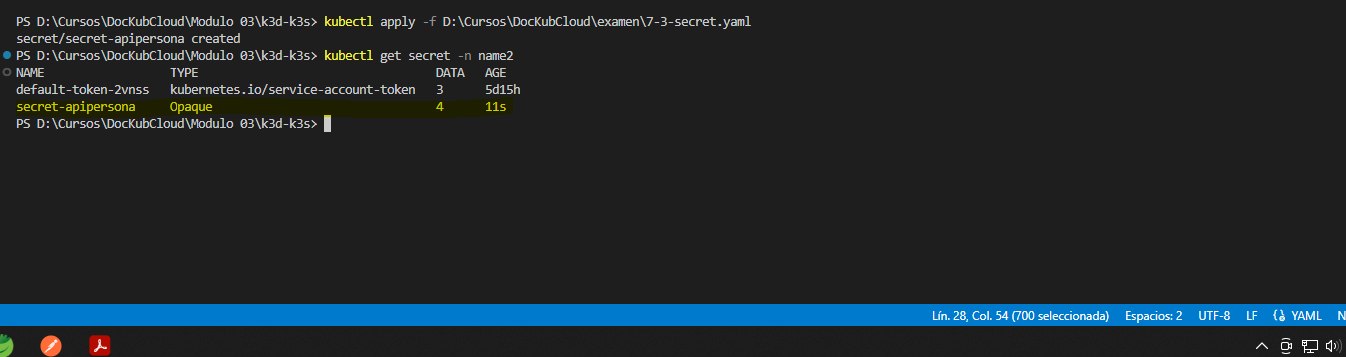


Se tuvo que cambiar por el tipo NodePort



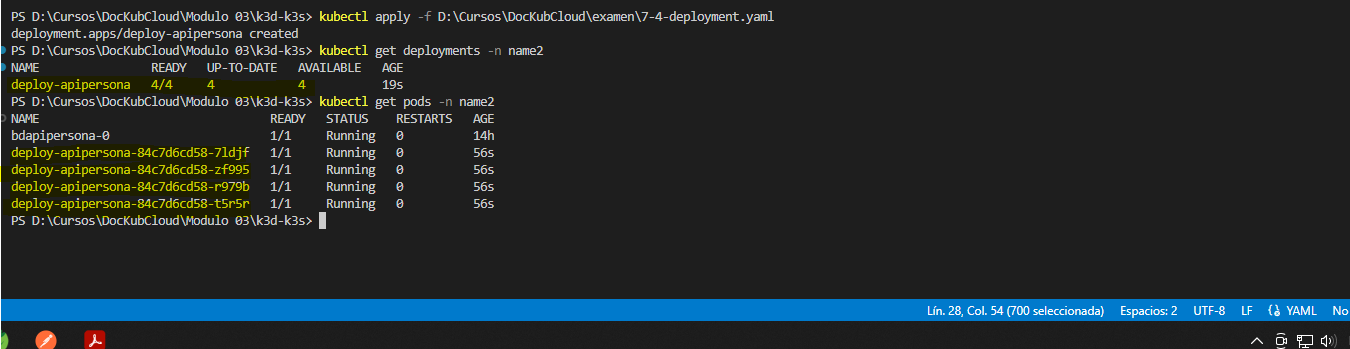
1. Crear un Secret que permita desacoplar la configuración del aplicativo

kubectl apply -f D:\Cursos\DocKubCloud\examen\7-3-secret.yaml



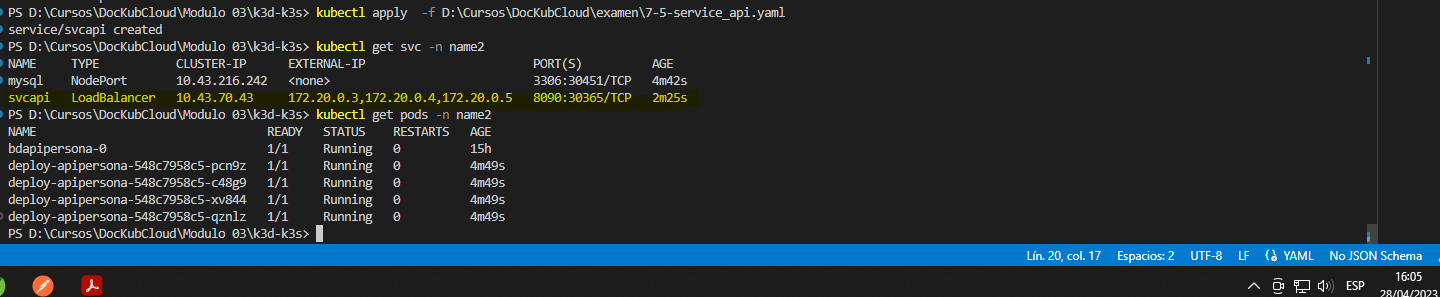
1. Crear la aplicación en un Deployment con 4 replicas que use el Secret y se conecte al Service de la base de datos

kubectl apply -f D:\Cursos\DocKubCloud\examen\7-4-deployment.yaml



1. Crear un Service de tipo LoadBalancer que libere el aplicativo

kubectl apply -f D:\Cursos\DocKubCloud\examen\7-5-service\_api.yaml



1. Realizar las pruebas mediante Postman

kubectl -n name2 port-forward service/svcapi 8080:8090

