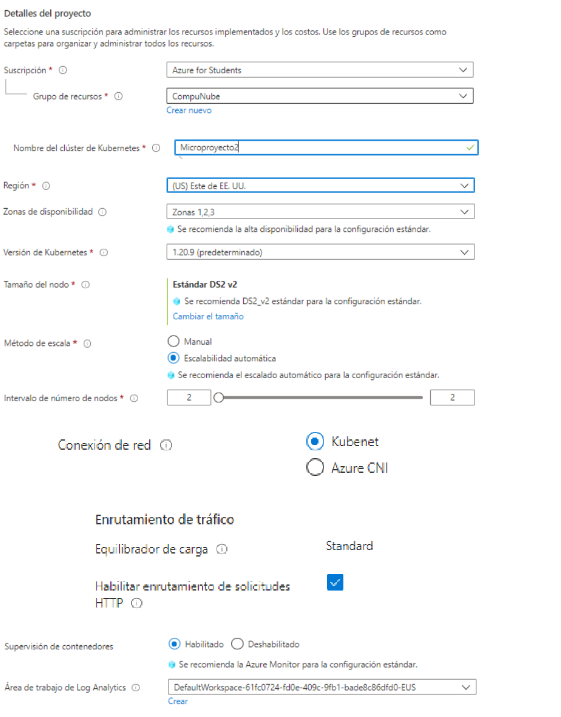
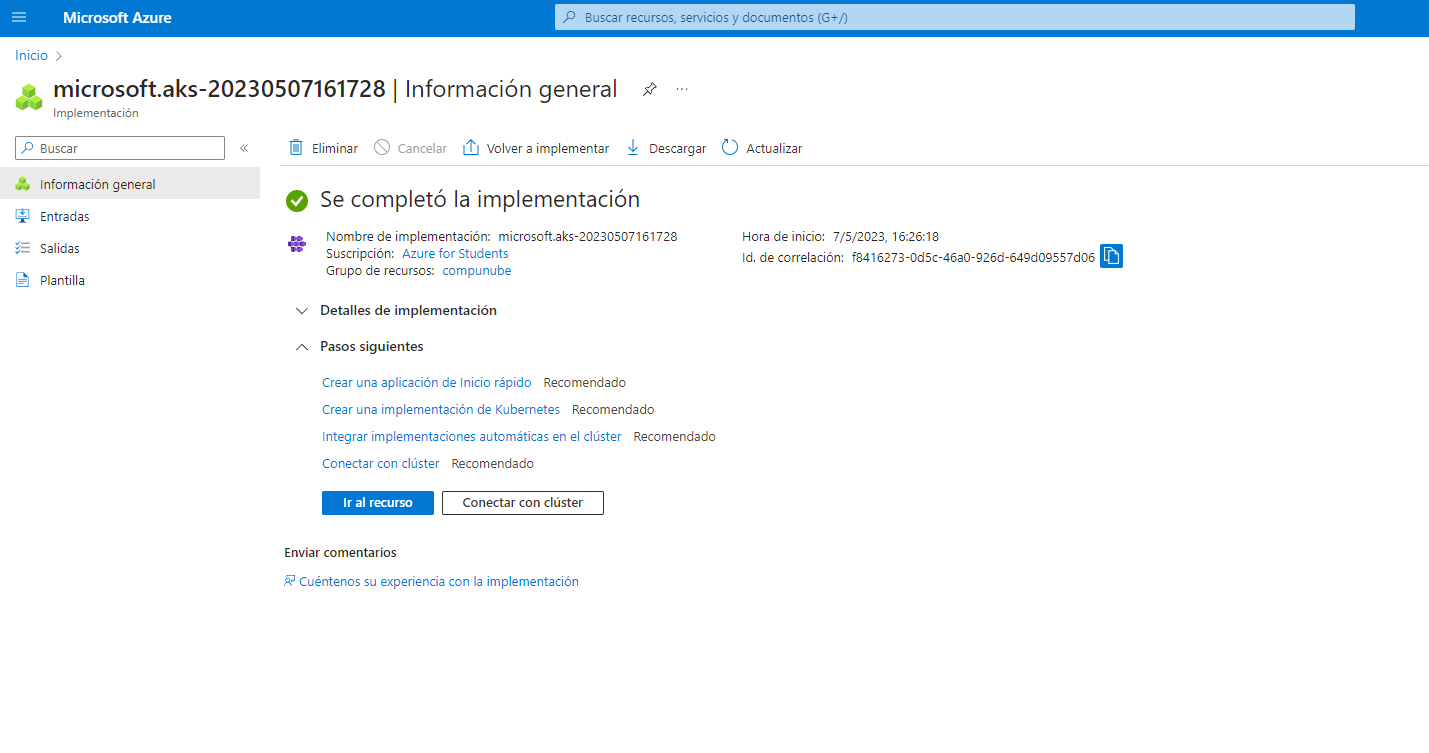
**AKS desde portal Azure**

**Diego Iván Perea Montealegre**



<- minuscula

<- minuscula



**Ingreso de azure por medio del CLI**

**Aclarar que ya se debe tener instalado minikube , docker y kubernetes en maquina de vagrant (https://github.com/diegoperea20/SpecializationAI/tree/main/CloudComputing/week5/kubernetes-p)**

Agrega la clave GPG del repositorio de Azure:

curl -sL https://packages.microsoft.com/keys/microsoft.asc | \

gpg --dearmor | \

sudo tee /etc/apt/trusted.gpg.d/microsoft.asc.gpg > /dev/null

Agrega el repositorio de Azure CLI a la lista de repositorios de apt-get:

AZ\_REPO=$(lsb\_release -cs)

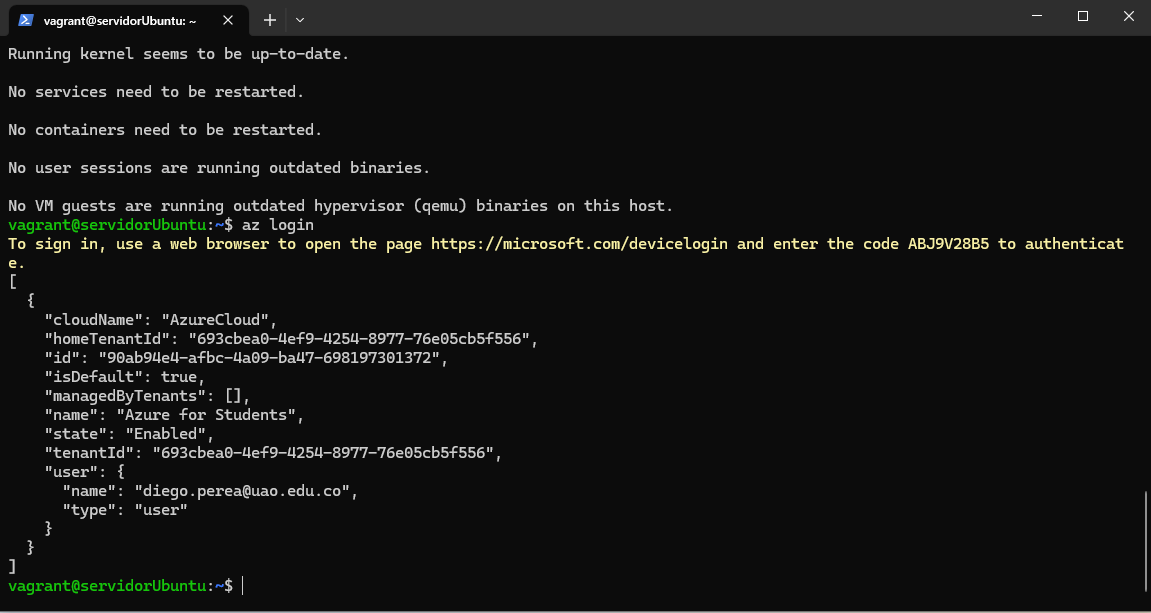
echo "deb [arch=amd64] https://packages.microsoft.com/repos/azure-cli/ $AZ\_REPO main" | \

sudo tee /etc/apt/sources.list.d/azure-cli.list

Actualiza la lista de paquetes de apt-get e instala la CLI de Azure:

sudo apt-get update

sudo apt-get install azure-cli



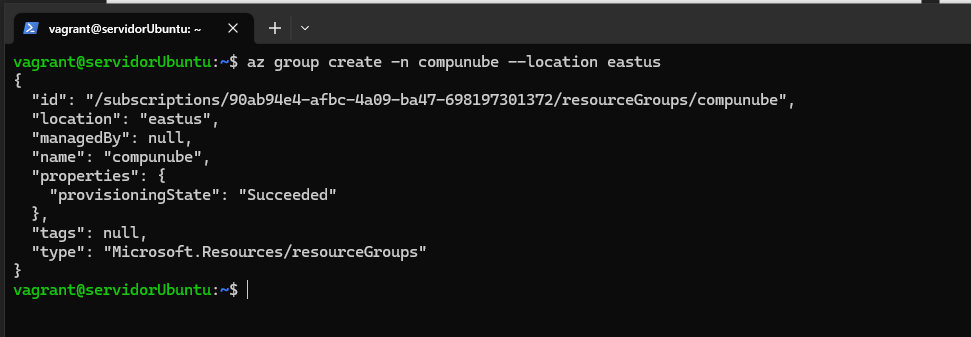
az provider register -n Microsoft.Compute

az provider register -n Microsoft.ContainerService

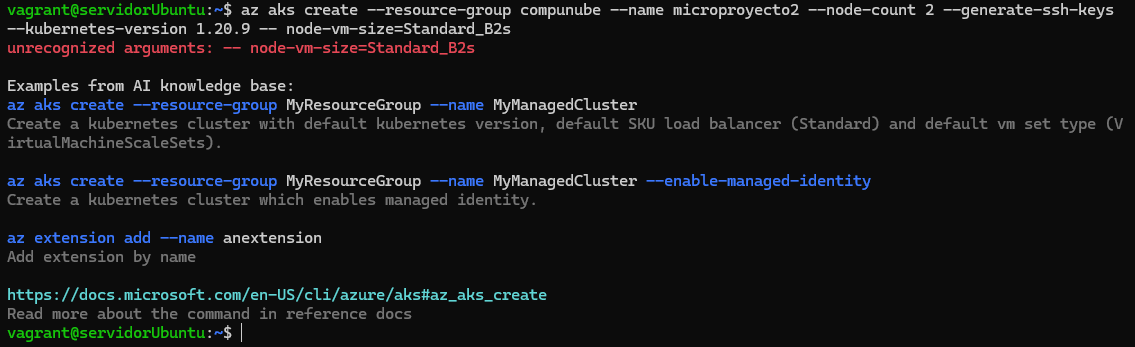
az provider register -n Microsoft.Network

az provider register -n Microsoft.Storage

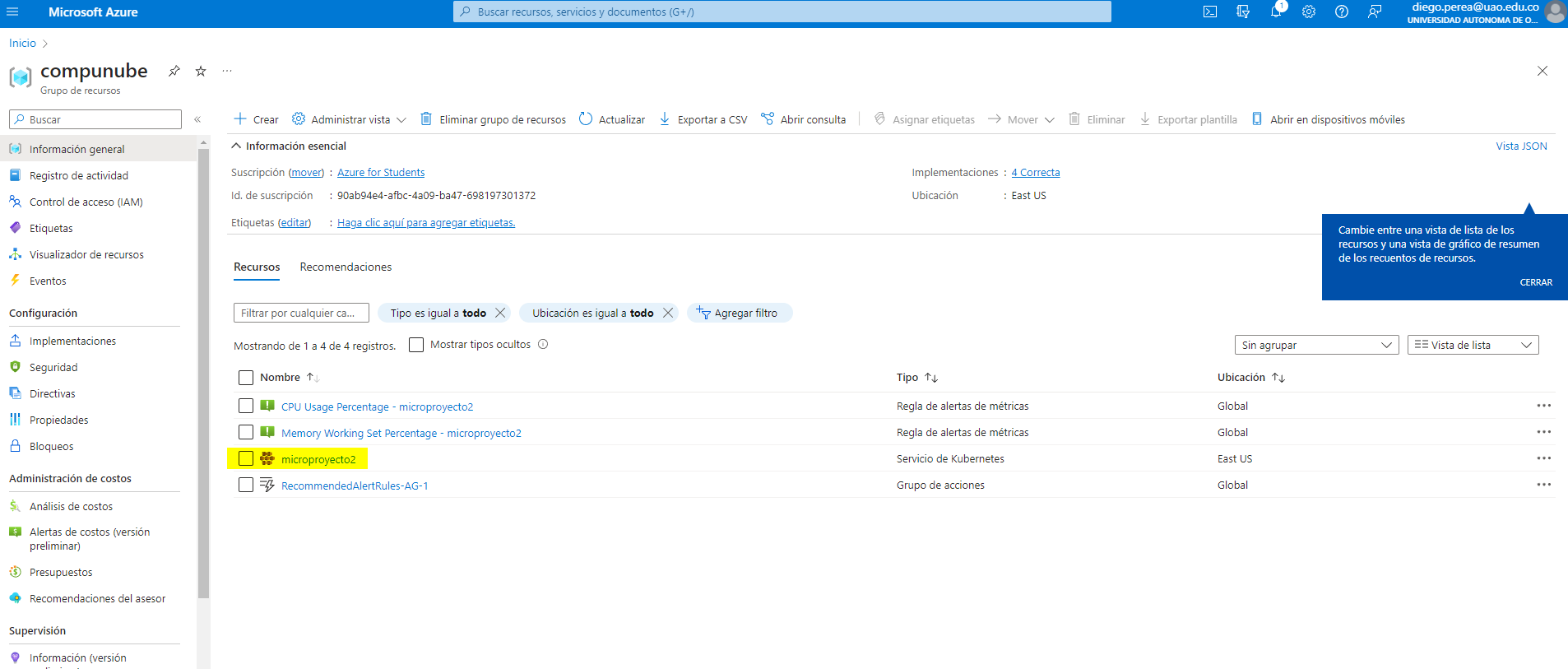
$ az group create -n compunube --location eastus



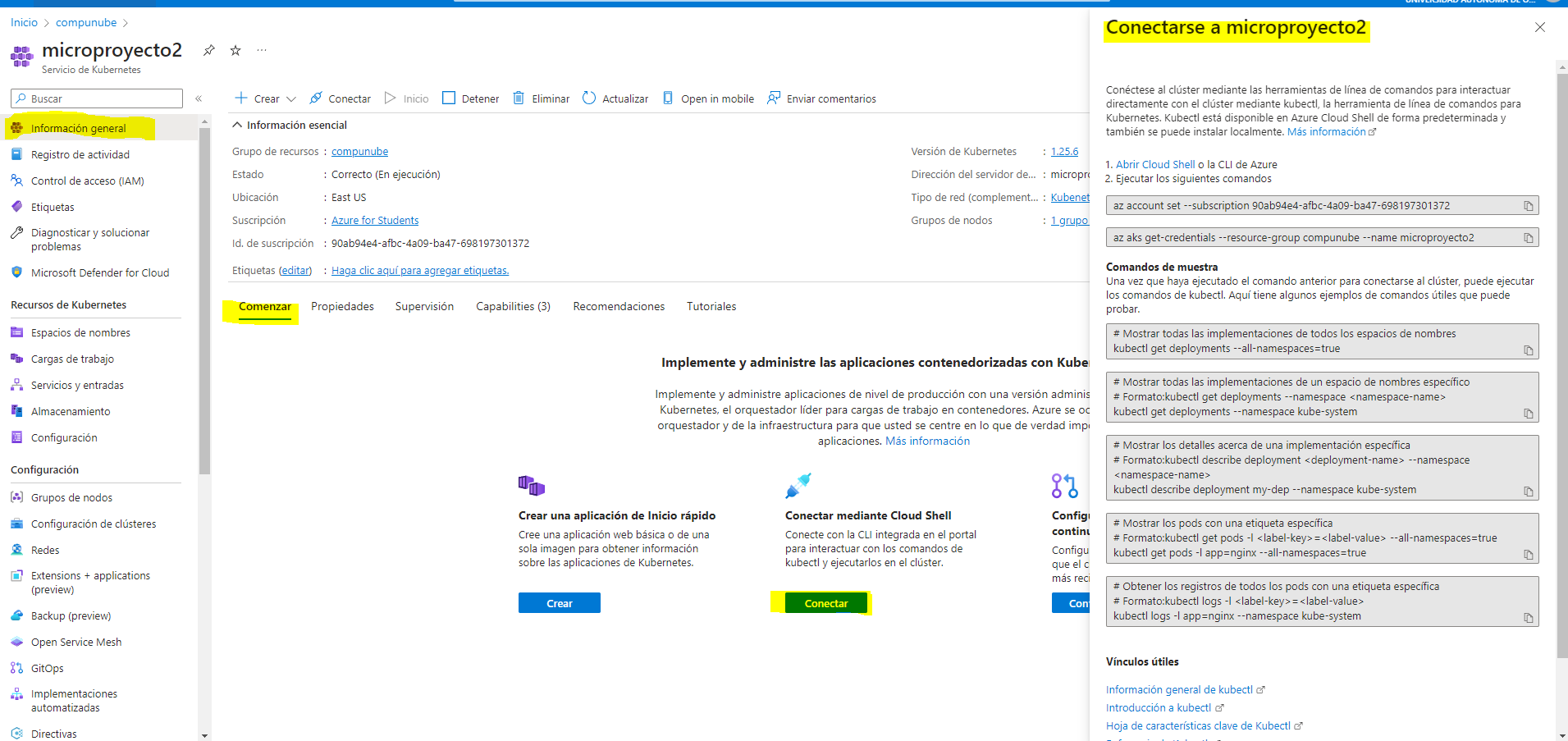
~$ az aks create --resource-group compunube --name microproyecto2 --node-count 2 --generate-ssh-keys --kubernetes-version 1.20.9 -- node-vm-size=Standard\_B2s

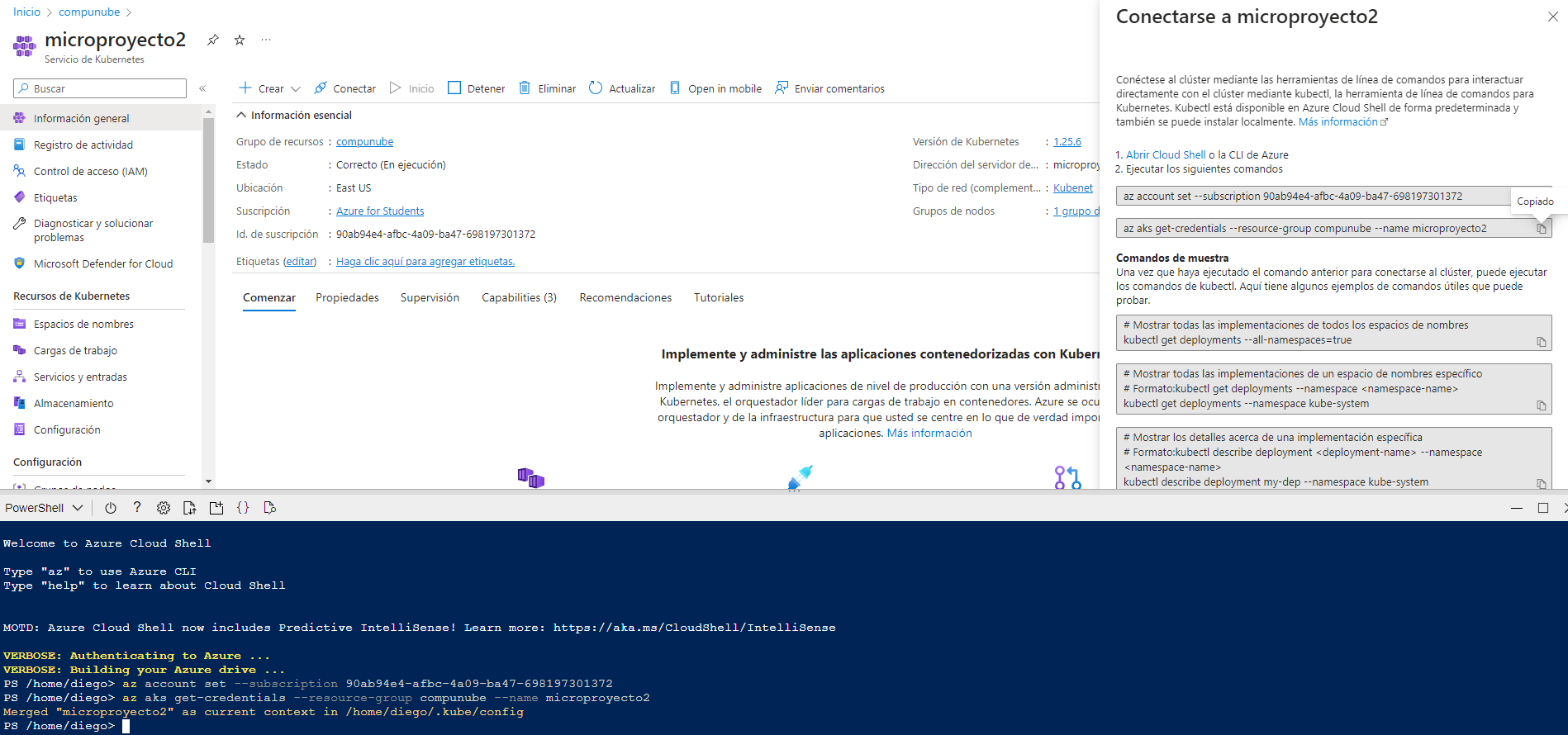


**Comprobacion de creación cluster AKS**



**Acceso a cluster AKS mediante Cloud Shell en Portal Azure**

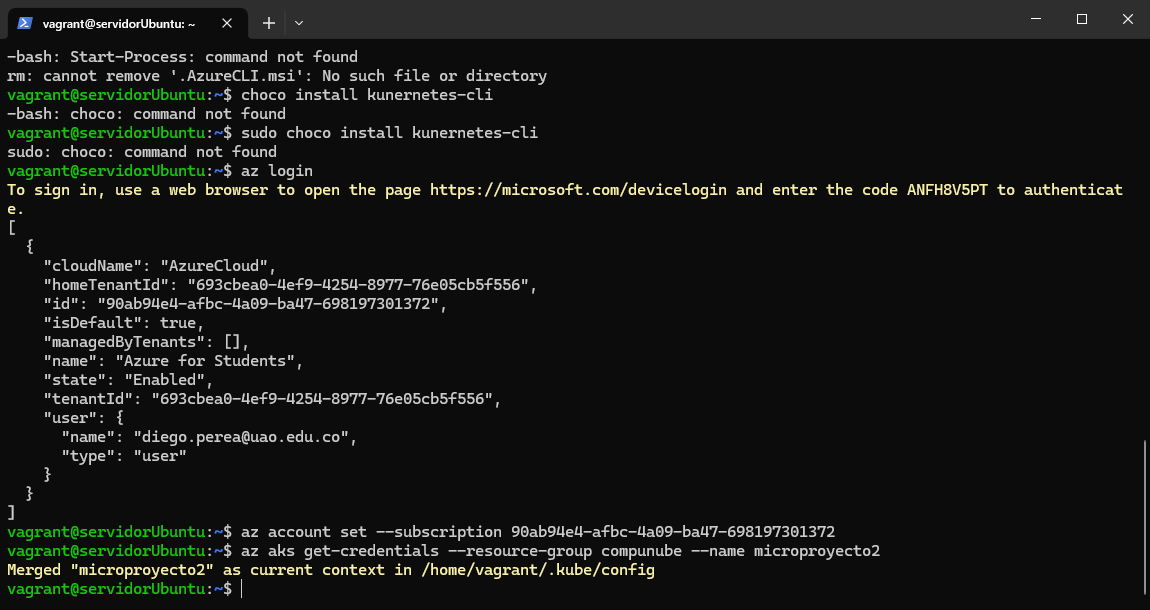




**Acceso a cluster AKS mediante CLI Azure desde maquina local**

az login

se compia y pega lo mismo que se hizo en powershell



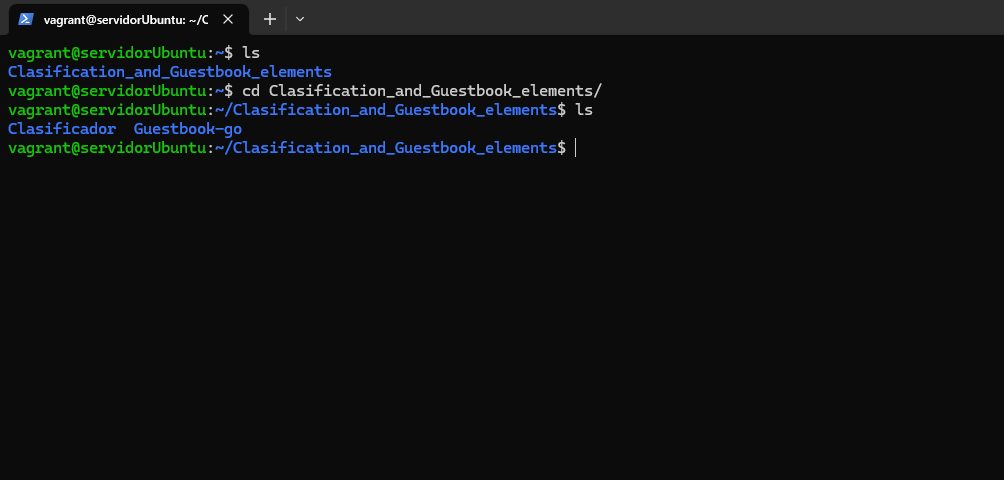
**2.CLASIFICADOR DE IMÁGENES CLI AZURE**

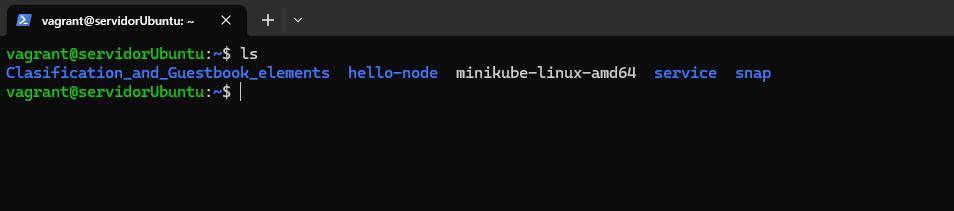
Se hace git clone al repo https://github.com/diegoperea20/SpecializationAI.git

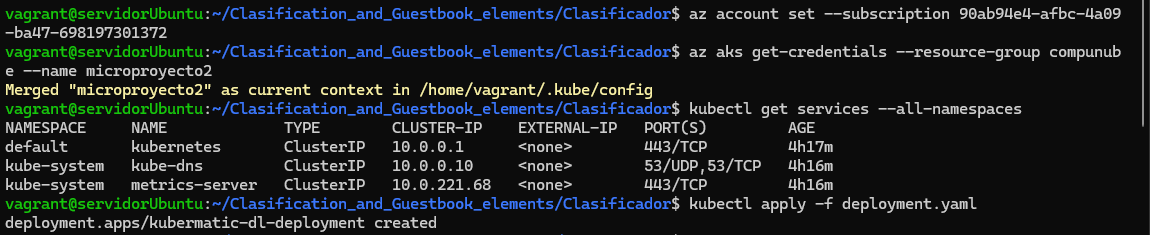
y se mueve “Clasification\_and\_Guestbook\_elements”hacia root

<https://github.com/diegoperea20/SpecializationAI/tree/main/CloudComputing/week6/miniproject2/Clasification_and_Guestbook_elements>

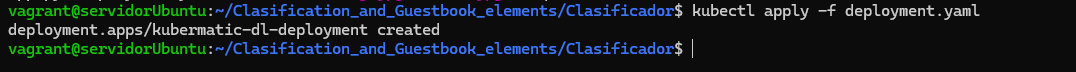
y se elimina lo demás



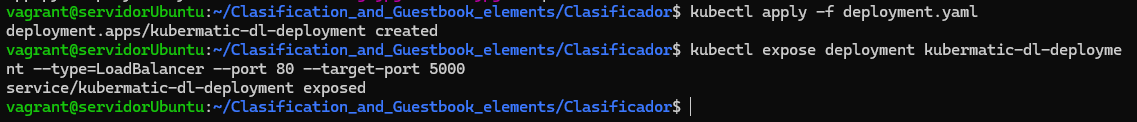




kubectl apply -f deployment.yaml

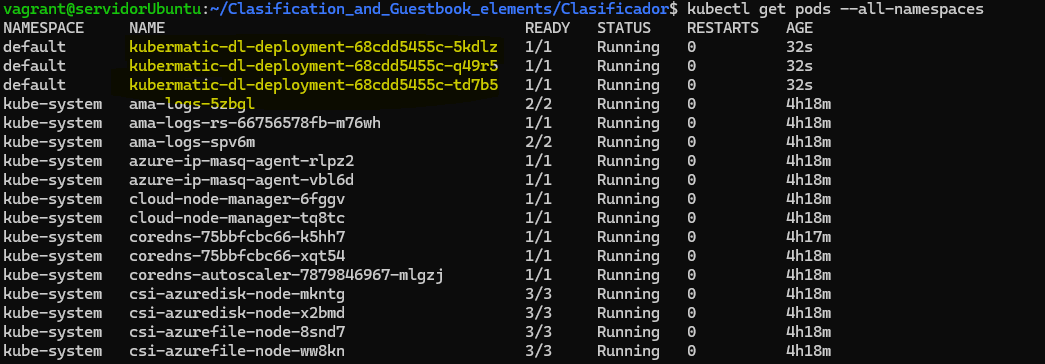


$ kubectl expose deployment kubermatic-dl-deployment --type=LoadBalancer --port 80 --target-port 5000

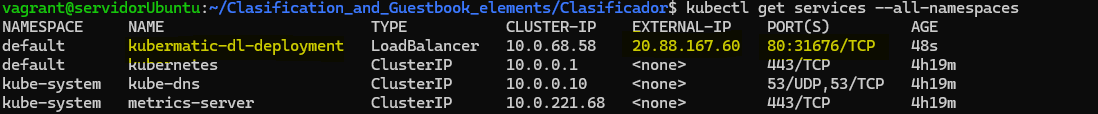


Verificar con kubectl get pods ,services y deployment esten configurados

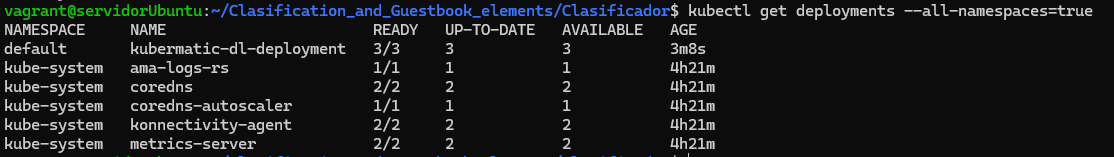
kubectl get pods --all-namespaces



kubectl get services --all-namespaces



kubectl get deployments --all-namespaces=true



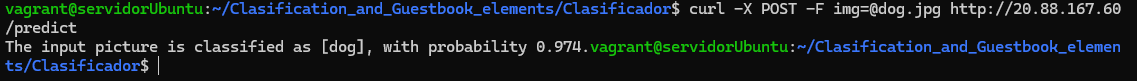
Finalmente hacemos dos pruebas con curl pasándole primero una imagen local de un perro para que la aplicación realice la predicción. La segunda prueba es otro curl, pero esta vez le pasamos una imagen de un caballo como vemos a continuación:

Colocar la external-ip de kubectl get services --all-namespaces

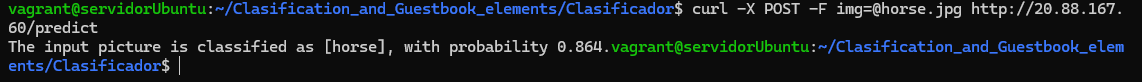
En este caso es EXTERNAL-IP =20.88.167.60

$ curl -X POST -F img=@dog.jpg http: //EXTERNAL-IP/predict

$ curl -X POST -F img=@dog.jpg http://20.88.167.60/predict



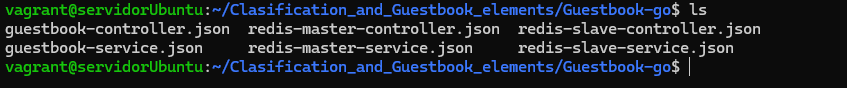
$ curl -X POST -F img=@horse.jpg <http://20.88.167.60/predict>



**3.Aplicacion de interés Implementación del servicio web Guestbook Cli Azure**

Aplicación guestbook-go: https://github.com/kubernetes/examples/tree/master/guestbook-go

Ir a la carpeta Guestbook-go



Creacion de servicio Redis , (Servicio kubernetes como balanceador)

$ kubectl apply -f redis-master-service.json



Creación los pods esclavos de redis , controlador de responsable de manejar varias instacias de un pod duplicado

§ kubectl apply -f redis-slave-controller.json



Creacion servicio esclavo de redis , (balancea carga de clientes)

kubectl apply -f redis-slave-service.json



Creamos los pods del guestbook, los servicios maestro o esclavo dependiendo de si la solicitud es de lectura o escritura.

$ kubectl apply -f guestbook-controller.json



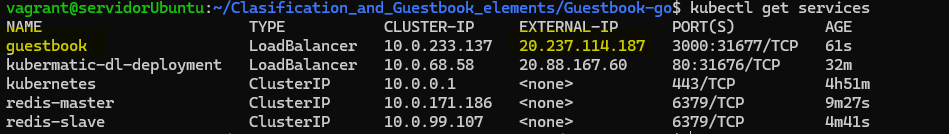
agrupar los pods del guestbook para que sea visible externamente para que sea tipo baleanceador

$ kubectl apply -f guestbook-service.json



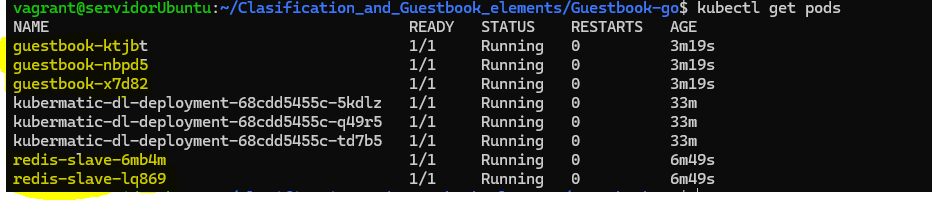
Visualizar los pods,servicios y controladores

$ kubectl get services

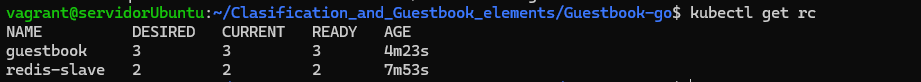


Ip externa: 20.237.114.187 en el puerto 3000

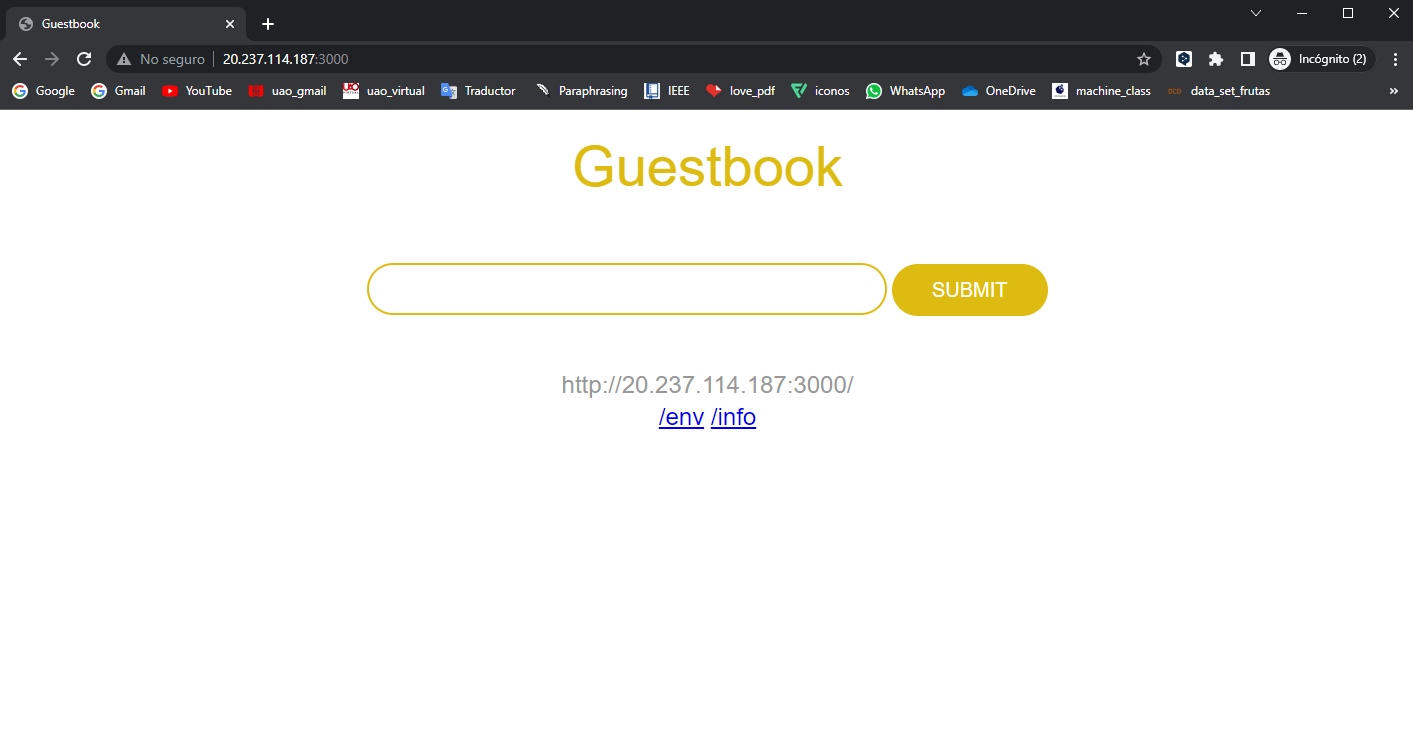
$ kubectl get pods



$ kubectl get rc



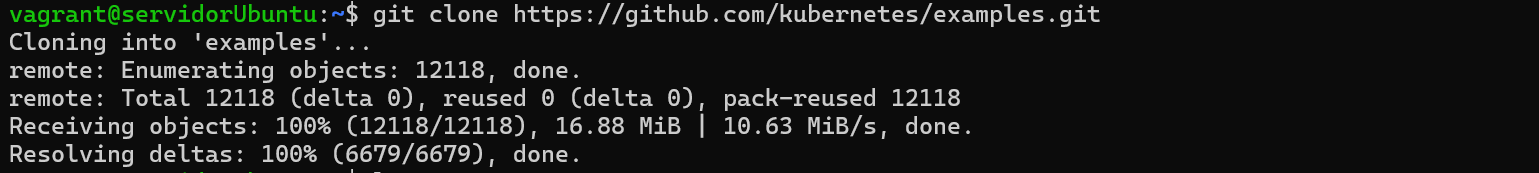
Se pude ver la aplicación en el navegador web con la Ip externa: 20.237.114.187 en el puerto 3000



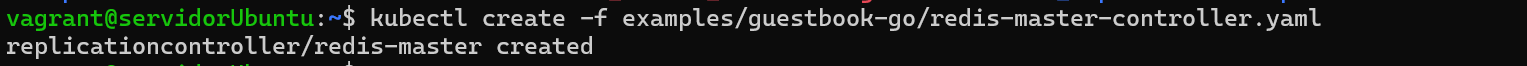
Sí no sirve hacer el siguiente

**CON EL ACTUALIZADO**

**git clone https://github.com/kubernetes/examples.git**

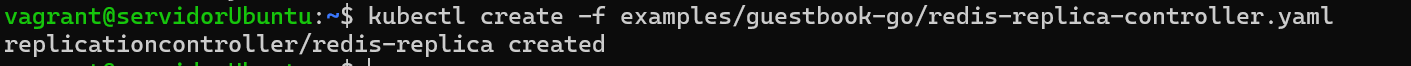


kubectl create -f examples/guestbook-go/redis-master-controller.yaml



kubectl create -f examples/guestbook-go/redis-master-service.yaml

kubectl create -f examples/guestbook-go/redis-replica-controller.yaml



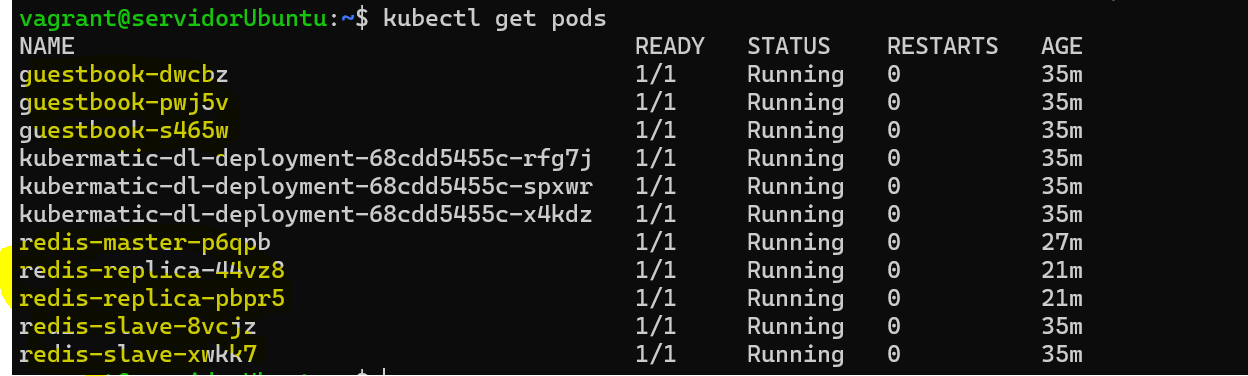
kubectl create -f examples/guestbook-go/redis-replica-service.yaml



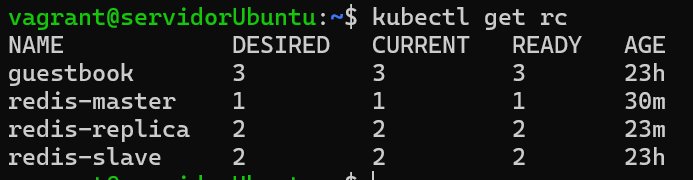
kubectl create -f examples/guestbook-go/guestbook-controller.yaml

kubectl create -f examples/guestbook-go/guestbook-service.yaml

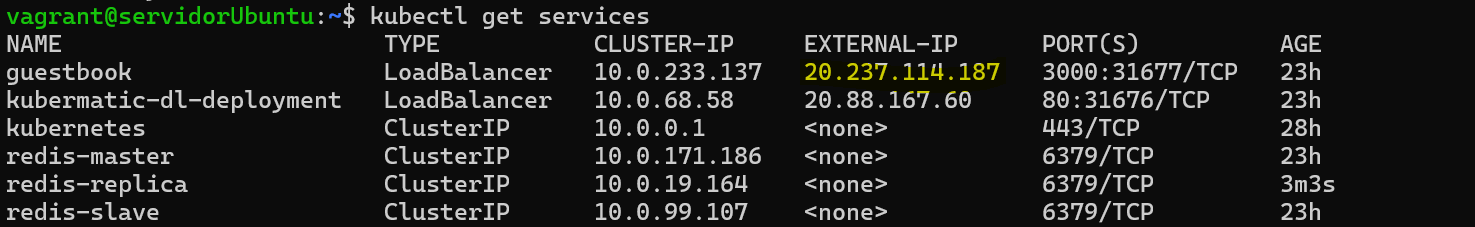
kubectl get pods



kubectl get rc

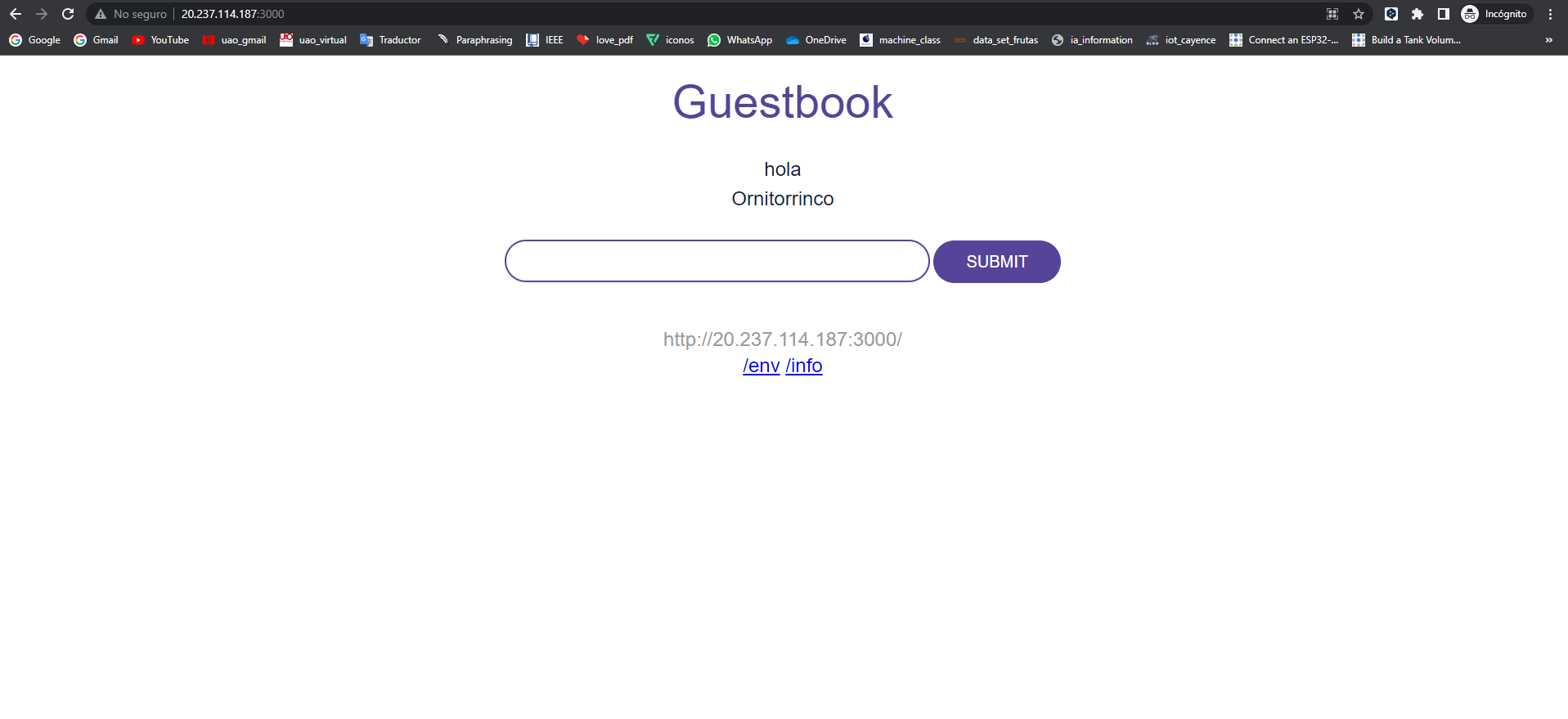


kubectl get services



CON LA IP EXTERNA EN este caso 20.237.114.187

En navegador 20.237.114.187:3000



**4.Supervision y monitoreo**

