#Prueba-Dummy

###Contar con las siguientes herramientas instaladas.

- Sonarqube
- Organización de azure DevOps.
- Docker.
- Azure Agent Pool SelfHosted
- Kubernetes
- Minikube / Hypervisor / Nube con conexión a Azure DevOps

`Utiliza un repositorio del siguiente link del apartado de framework https://docs.docker.com/samples/`

####Procedimento

1. Descarga los archivos del repositorio elegido.
2. Instala el framework necesario en caso de no tenerlo.
3. Compila la aplicación luego de pasar el analisis de sonarqube.
    - Agregar dos escenarios 1 - analisis fallido | 2 - analisis exitoso
4. Genera una imagen de docker y sube la imagen a dockerhub.
5. Dentro del pipeline ejecute lo siguiente en bash o powershell. a. Imprime `Hola Mundo` 10 veces en pantalla con un job paralelo. b. Script que cree 10 archivos con la fecha y luego lo imprima en consola
6. Despliega la app a un clúster de kubernetes (minikube o EKS o AKS).
7. Crea un endpoint externo accesible (ingress) para la aplicación
8. Sube al repo en una carpeta `environment` todos los yaml de k8s.

####Que se espera del ejercicio

1. Configuración de la infraestructura desde cero.
2. Documentación para crear solución y demostración de la aplicación funcionando.
3. Coding Standards.
4. Enfoque hacia la meta.

####Bonus para tomar en consideración

1. Construye un clúster de kubernetes usando IaC (terraform o eksctl).
2. Usa un manejador de templates como Kustomize o Helm.
3. Despliega en nube publica (AWS o Azure).
4. Que sea accesible desde internet.
5. Uso de metodologías DevOps.

####Resultados que debes adjuntar

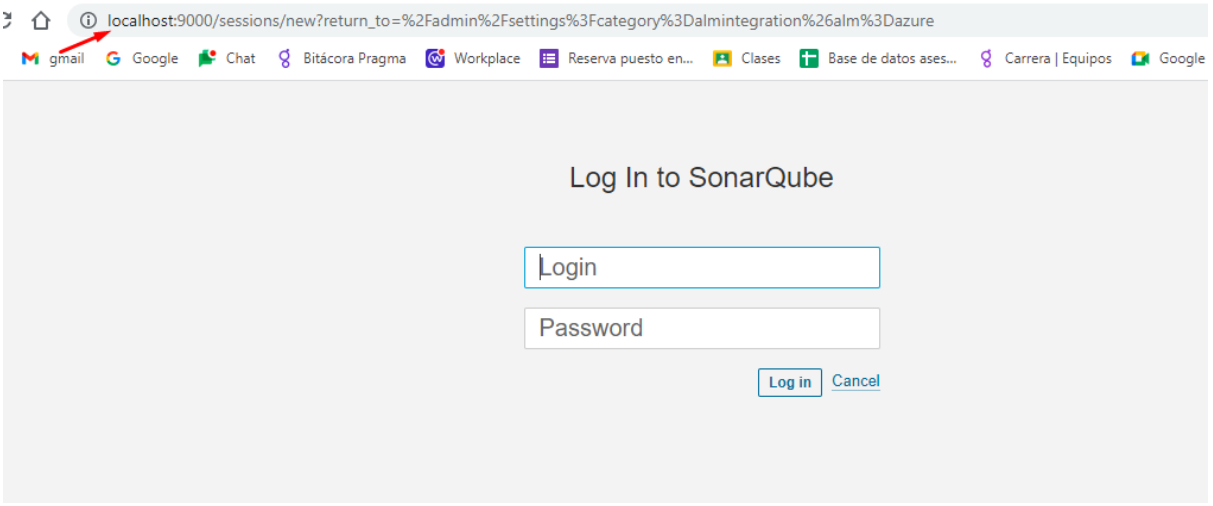
1. Codigo

2. yaml de k8s
3. Pipelines
4. Logs
5. Printscreen
6. Recording de pantalla (opcional)
7. Compartir repositorio de github publico para evaluación

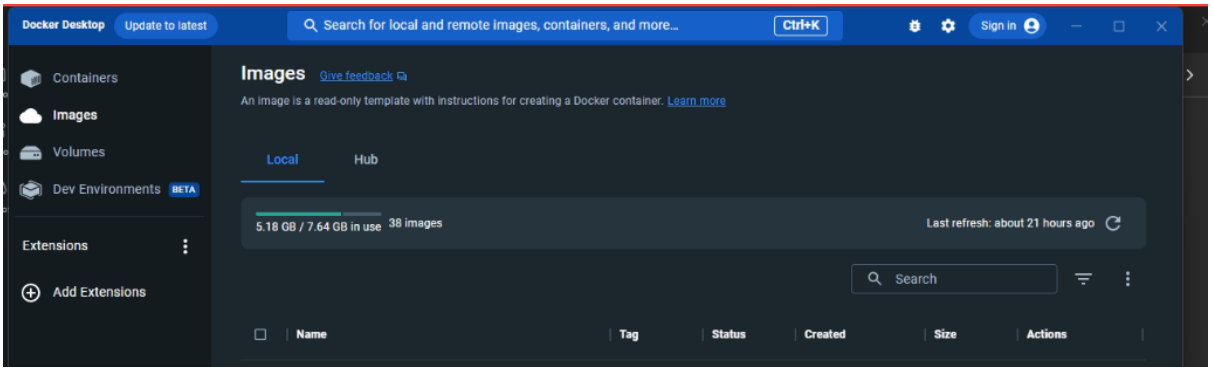
solution

###Contar con las siguientes herramientas instaladas.

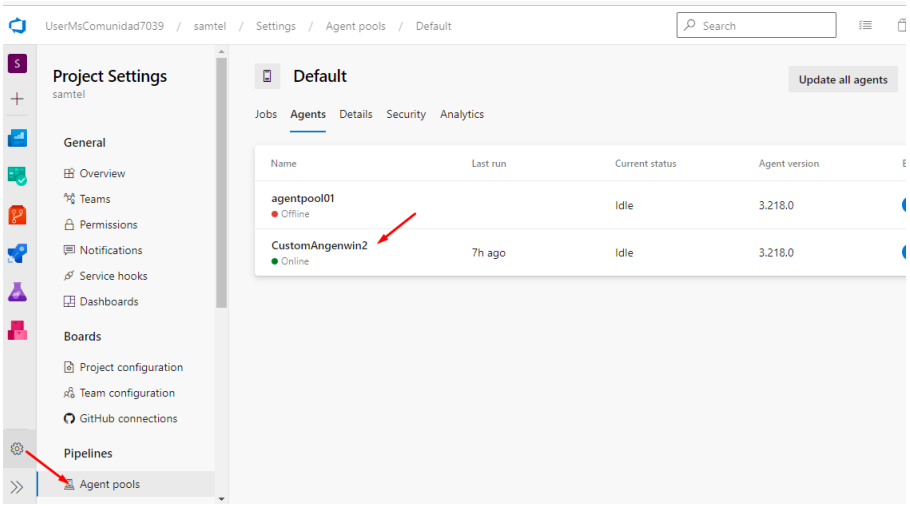
- Sonarqube

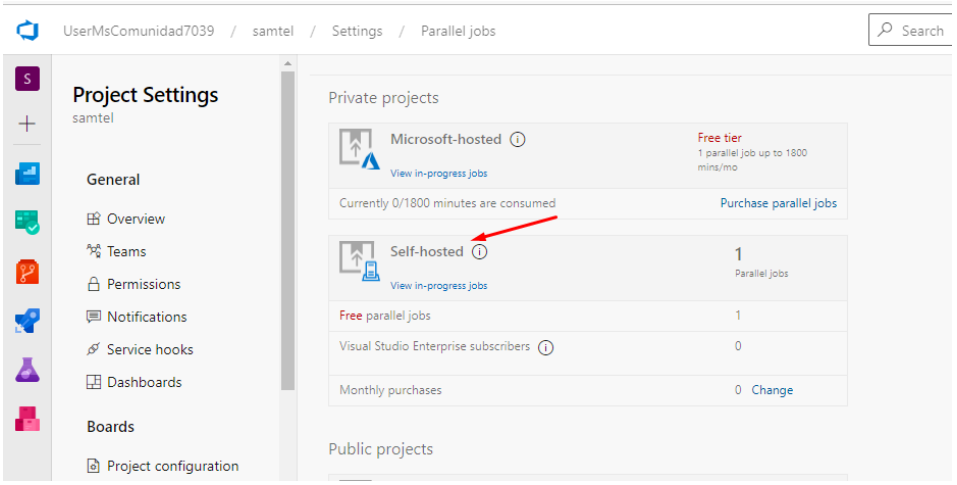


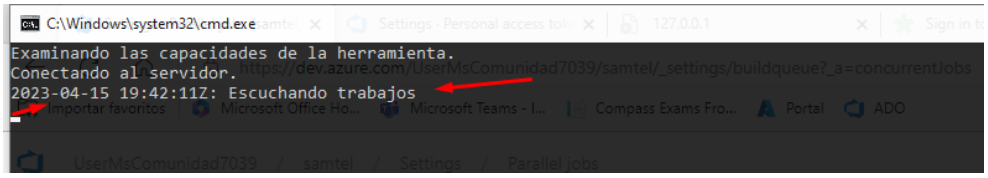
- Docker.

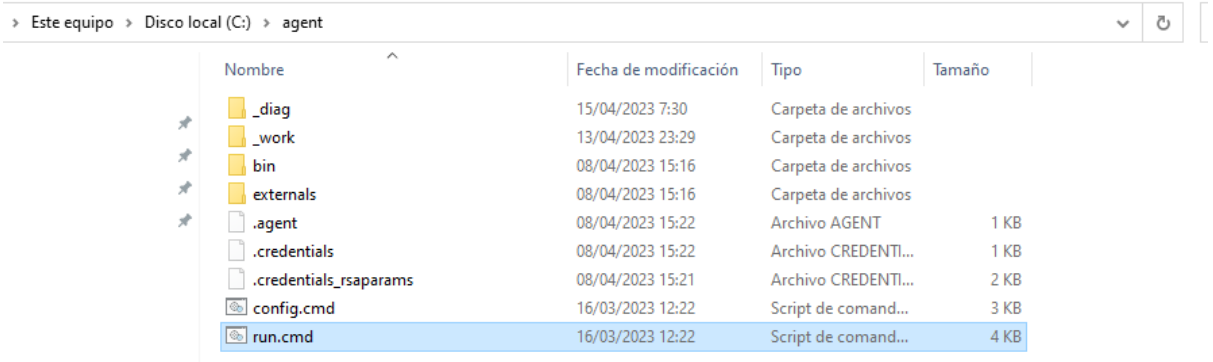


- Azure Agent Pool SelfHosted









- Kubernetes
- Minikube / Hypervisor / Nube con conexión a Azure DevOps

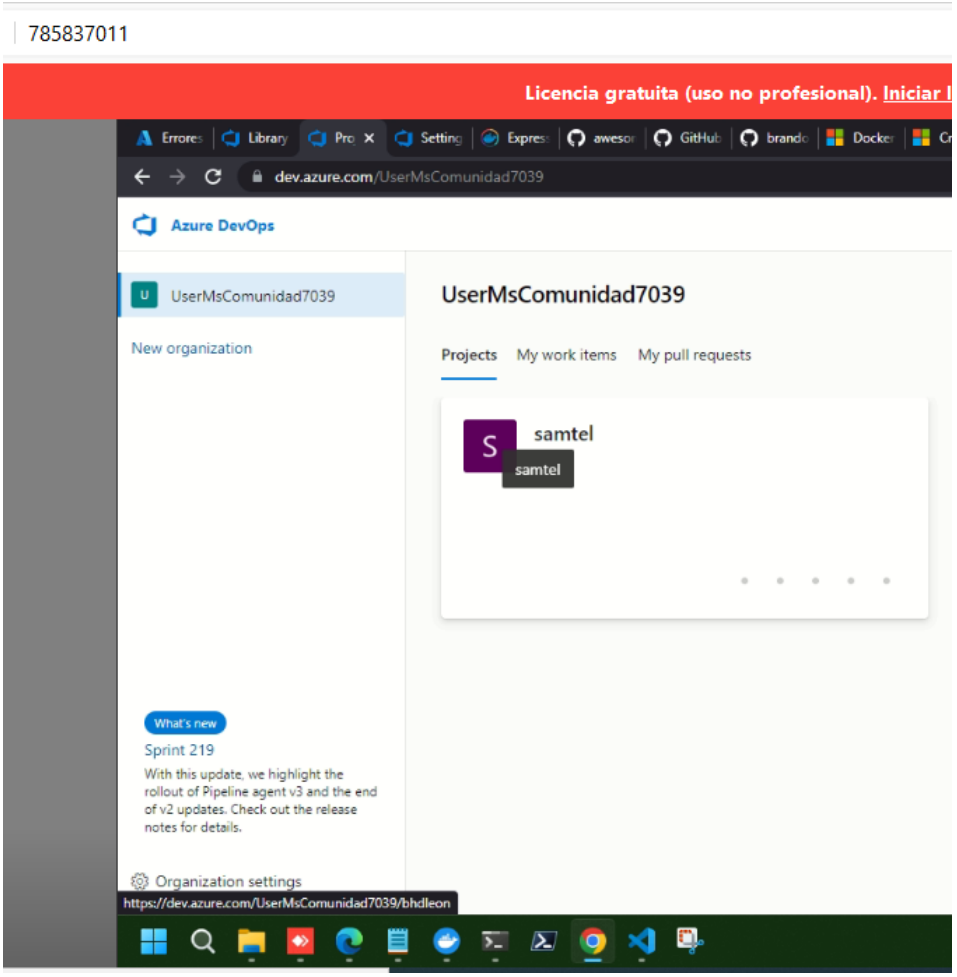
minikube status

```
PS C:\repos\samples-dockerhub-1\environment-aks> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

- Organización de azure DevOps.

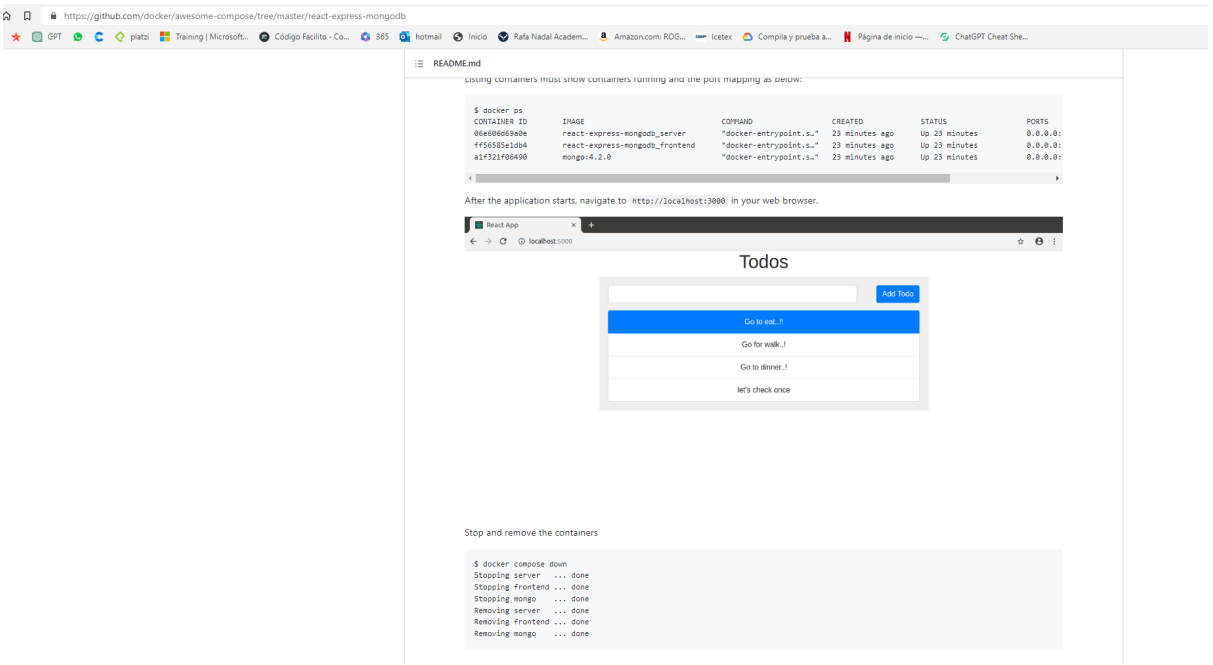
Organización de azure DevOps. (cuenta educativa)

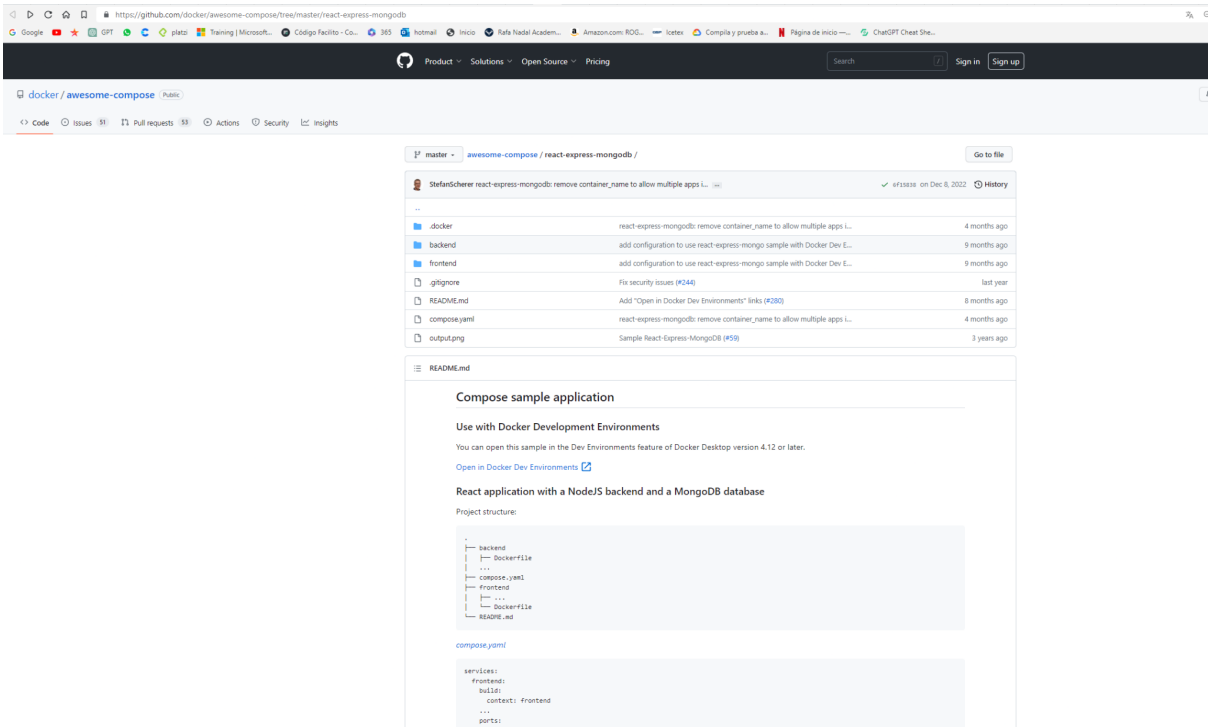
[Summary - Overview (azure.com)]



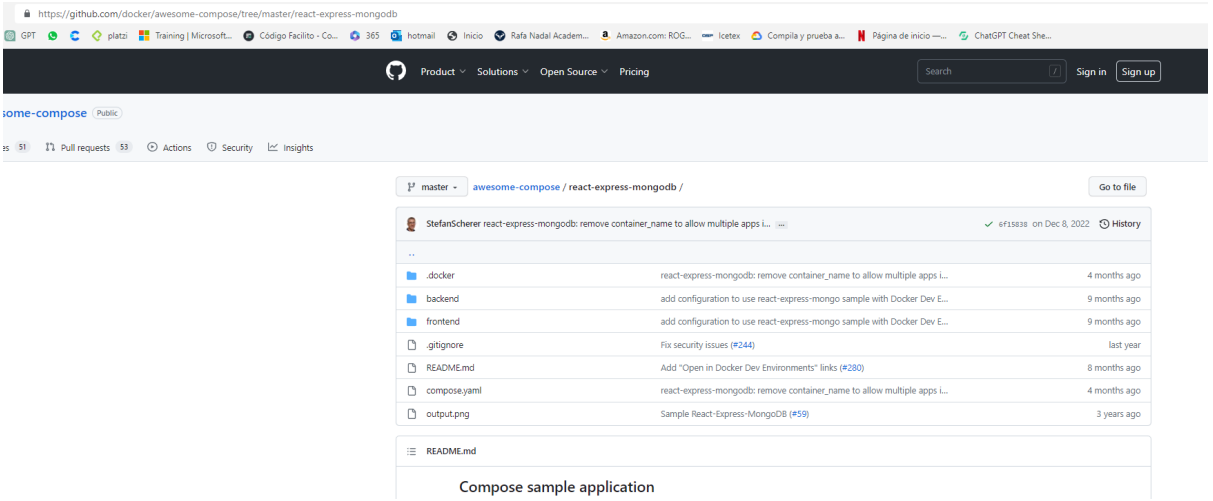
`Utiliza un repositorio del siguiente link del apartado de framework https://docs.docker.com/samples/`

https://github.com/docker/awesome-compose/tree/master/react-express-mongodb

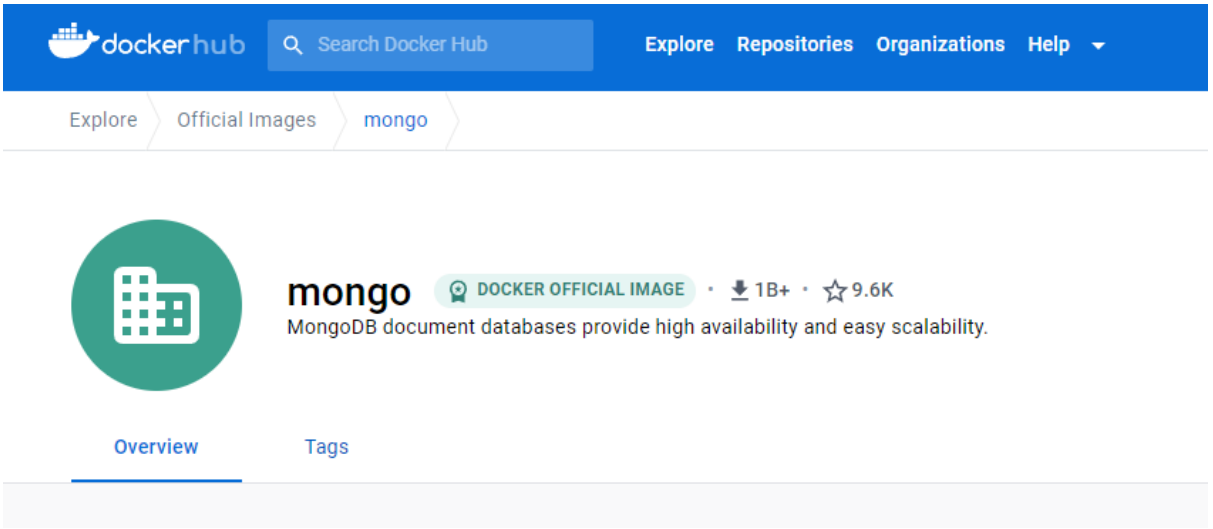


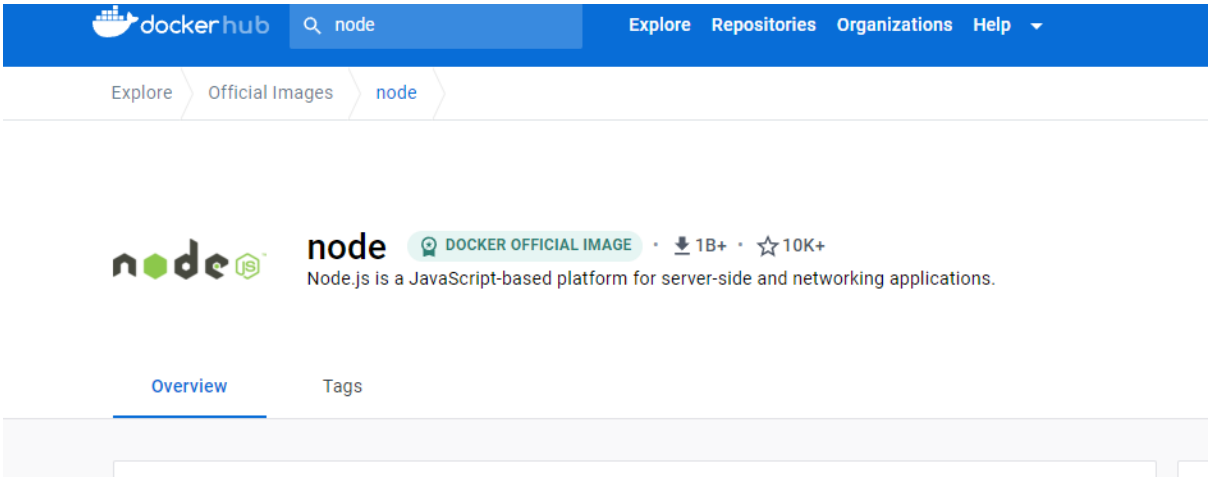


1 Descarga los archivos del repositorio elegido.



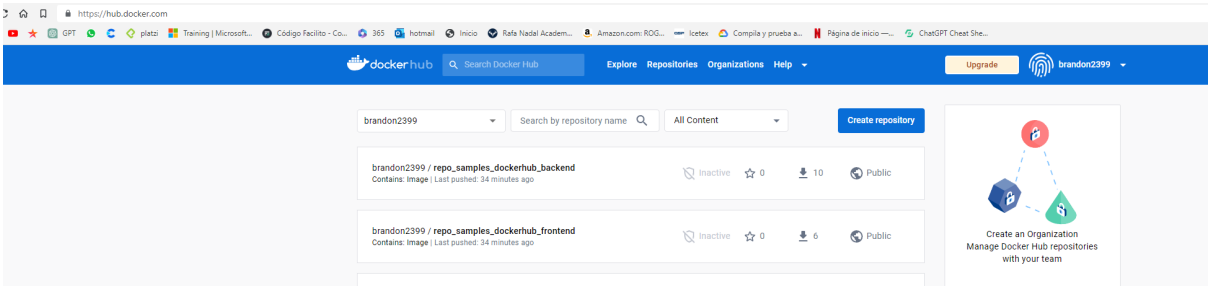
2 Instala el framework necesario en caso de no tenerlo.



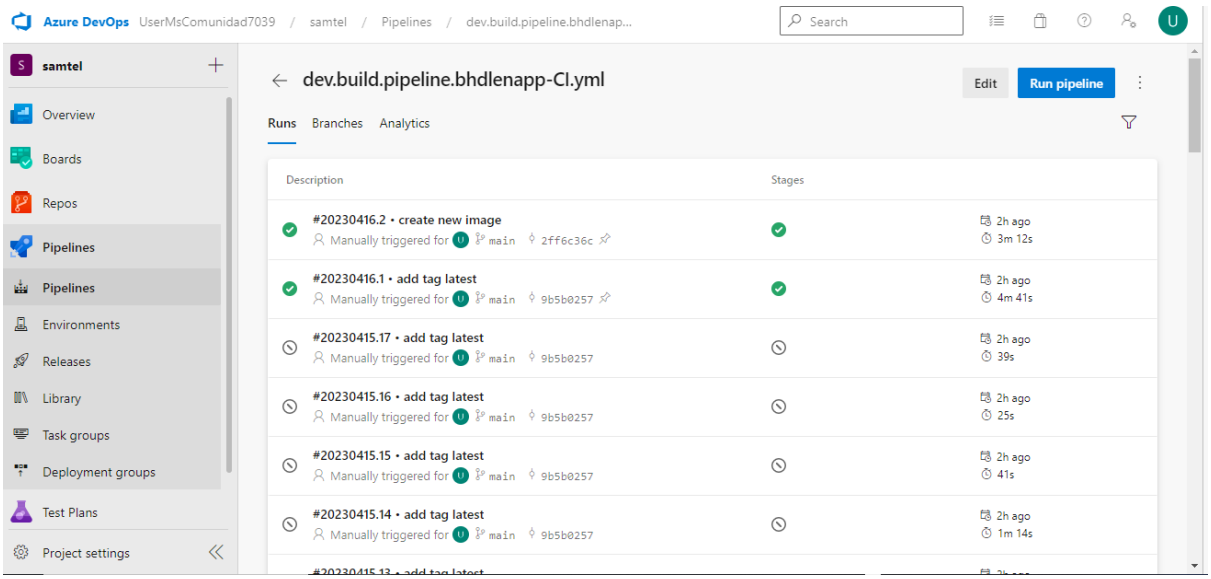


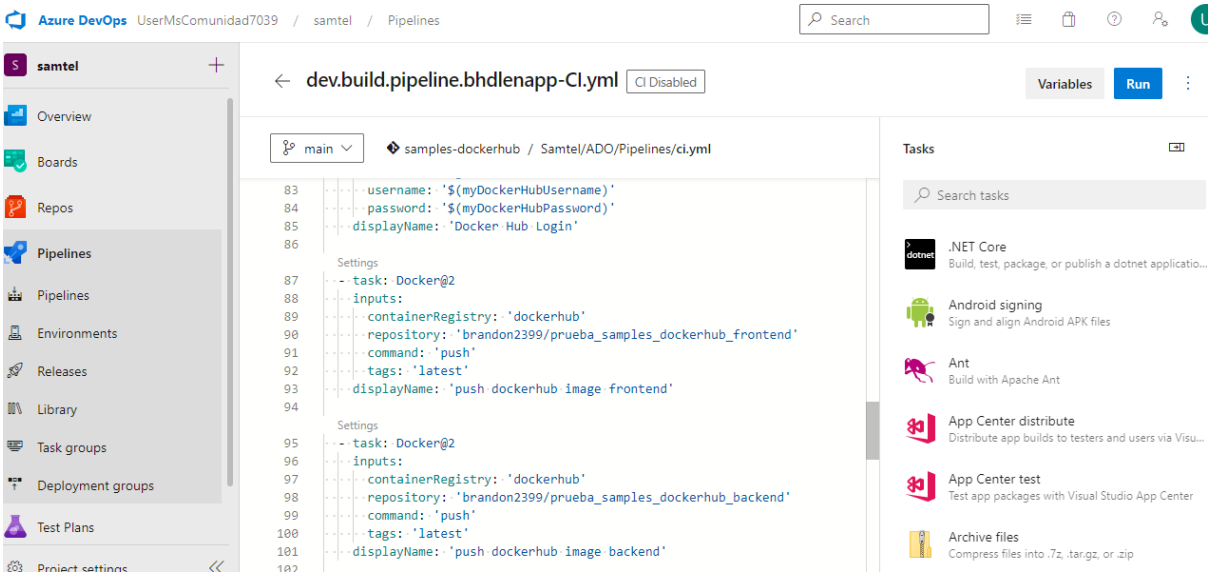
3 Compila la aplicación luego de pasar el analisis de sonarqube.

4 Genera una imagen de docker y sube la imagen a dockerhub.

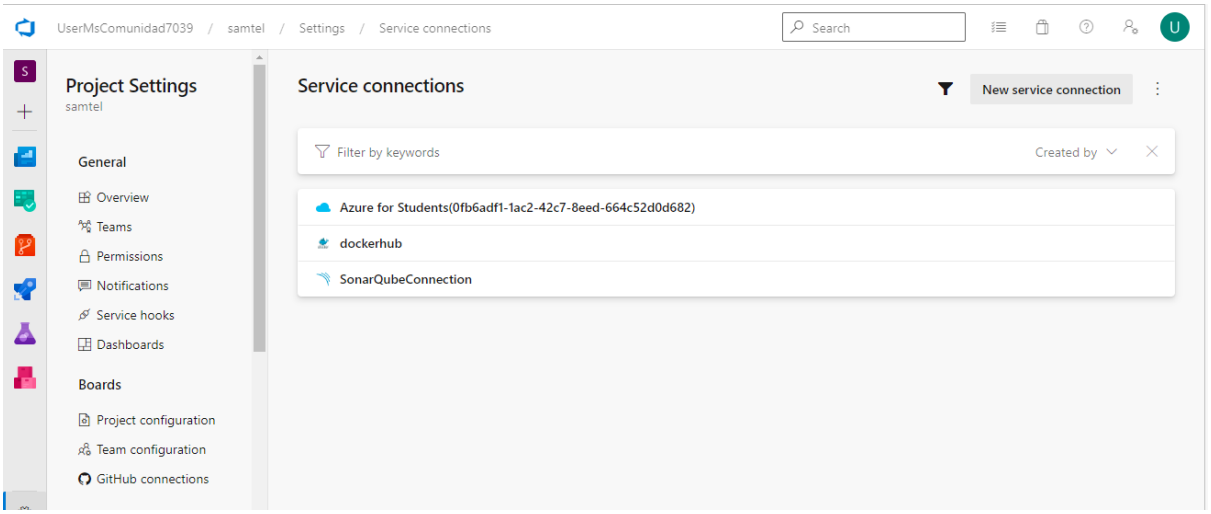


pipeline

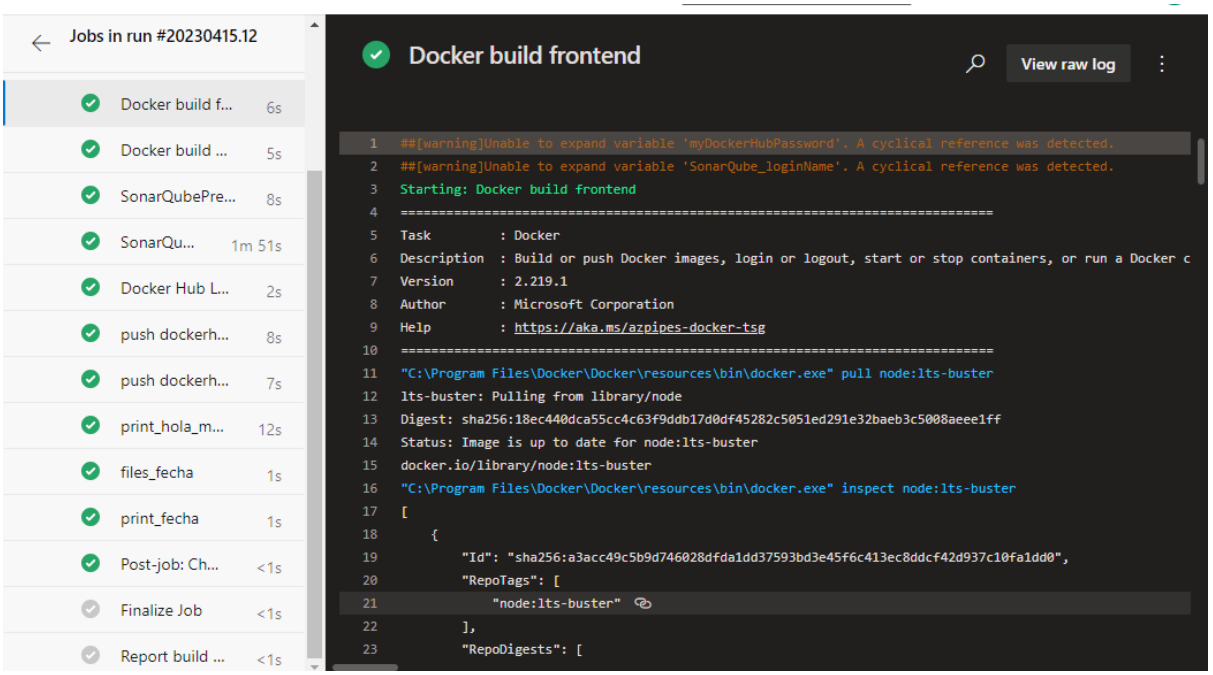




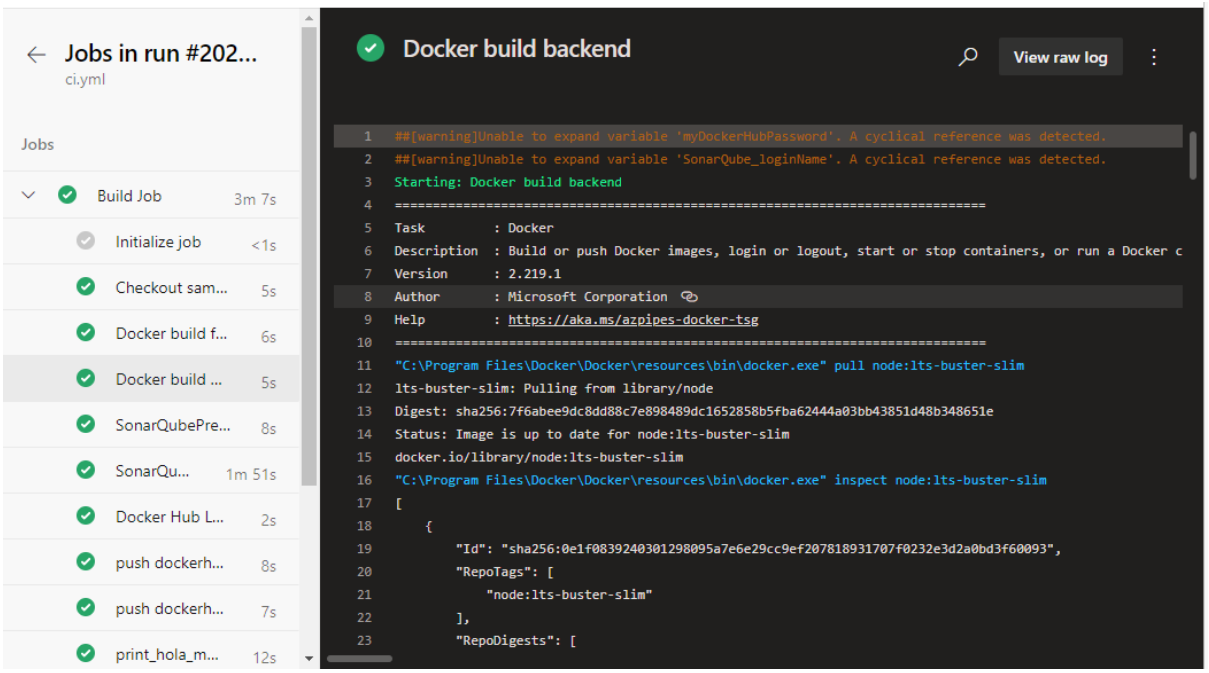
service connection con dockerhub y sonarqube



build imagen frontend



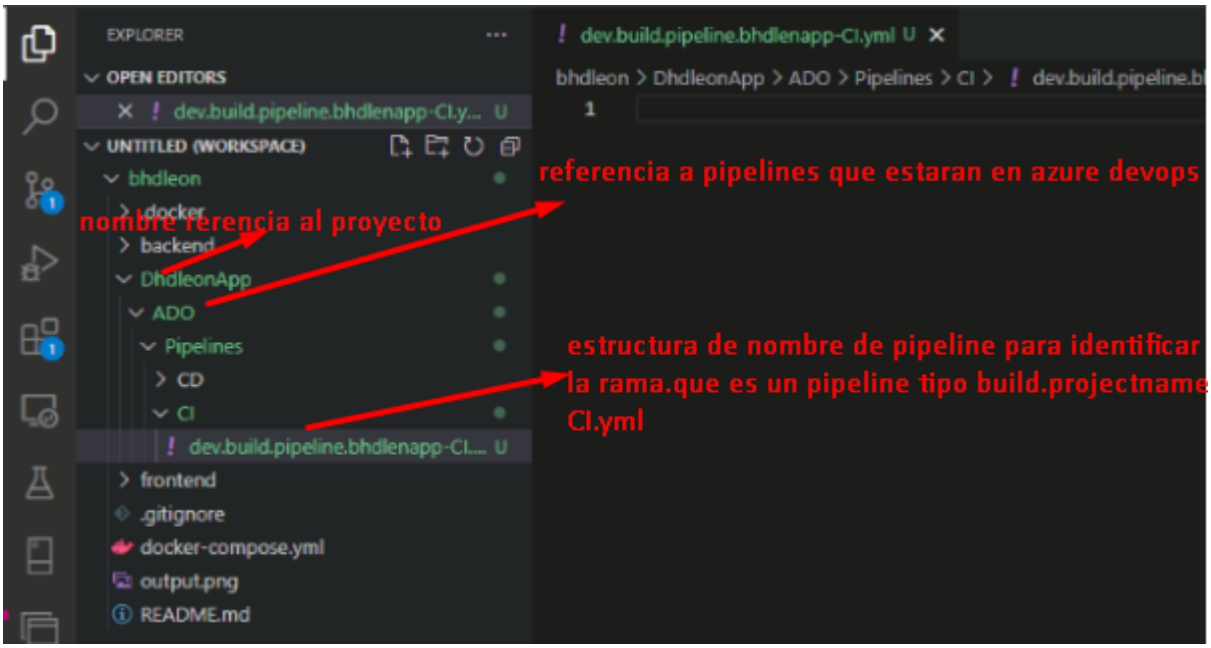
build imagen backend



Agregar dos escenarios 1 - analisis fallido | 2 - analisis exitoso

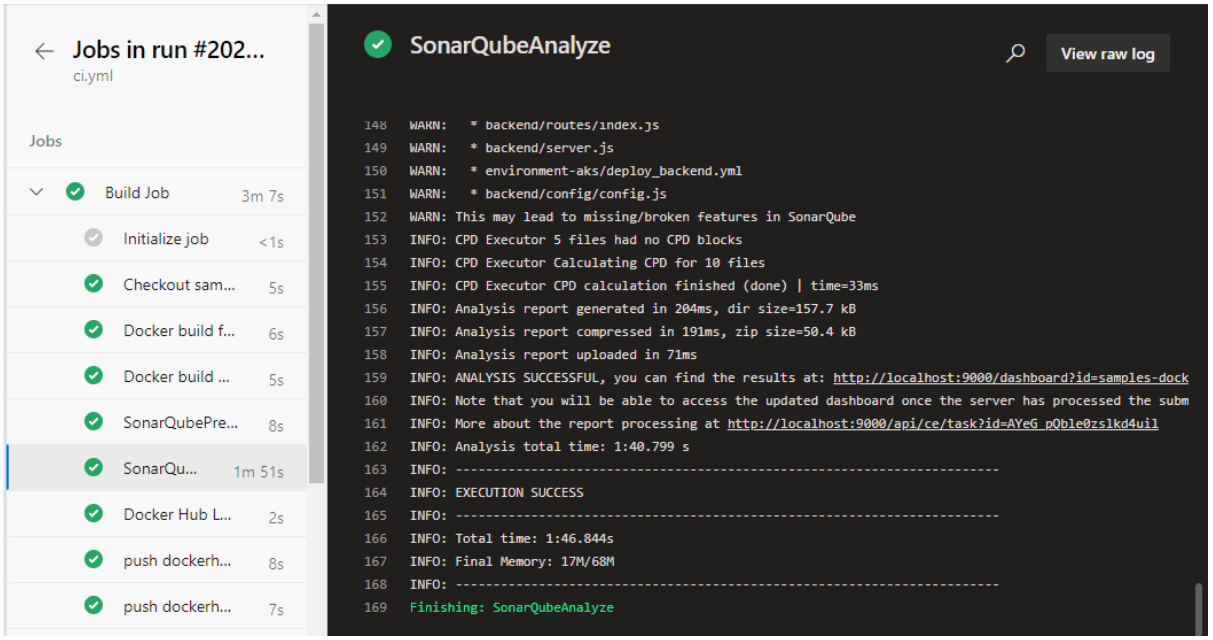
punto y tres y cuatro son complementarios dentro del ciclo devops de la aplicación y se encuentran en la tapa del build que rodeo todos los procesos anteriores como partida de planning ->>>code >> build >> test

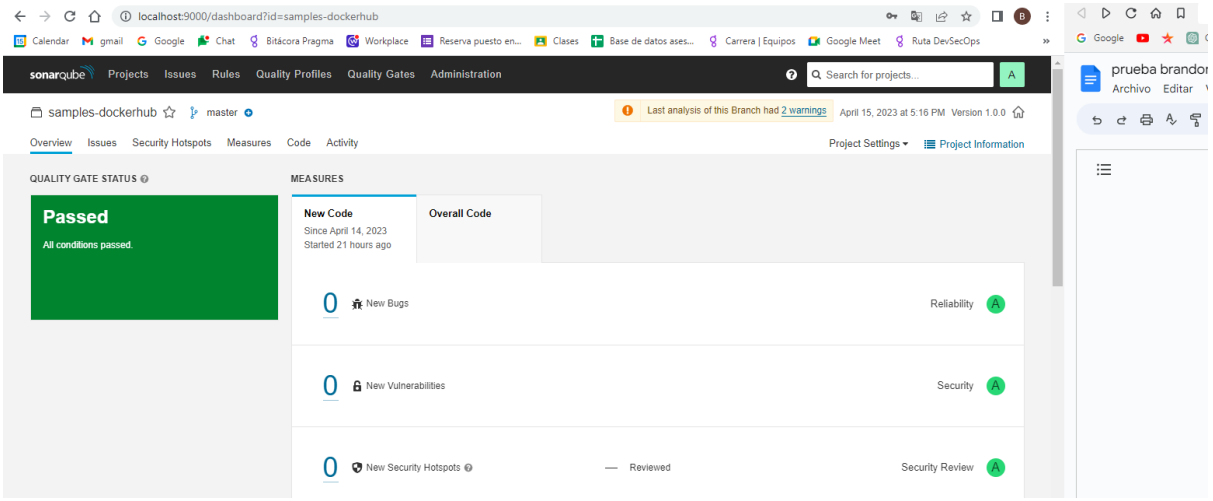
se crea la siguiente estructura de carpeta para empezar los pipelines tipo CI

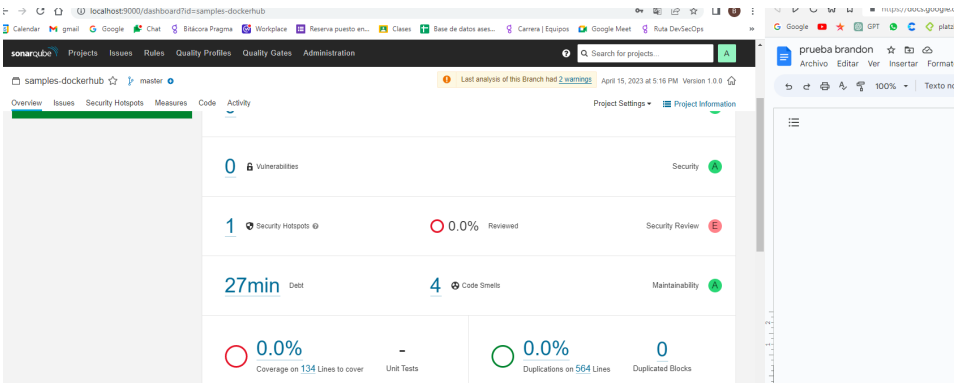


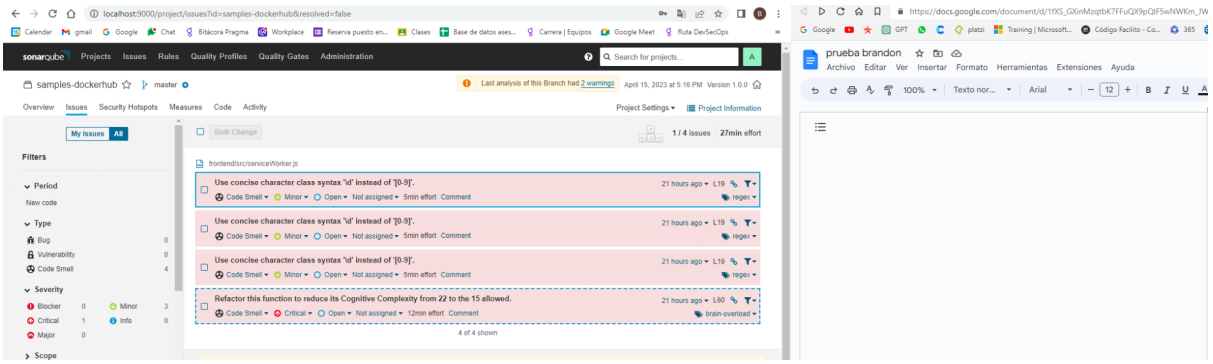
de esta manera se permite tener una estructura que permite identificar los tipos de pipelines,templaes, variables e infra

ejecucion del pipeline tarea sonar





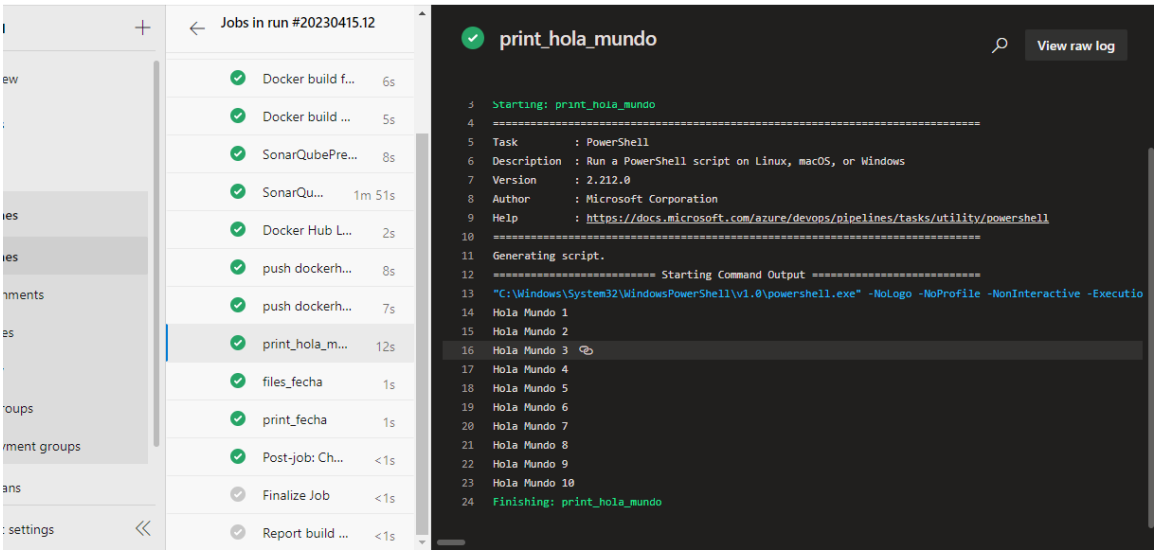




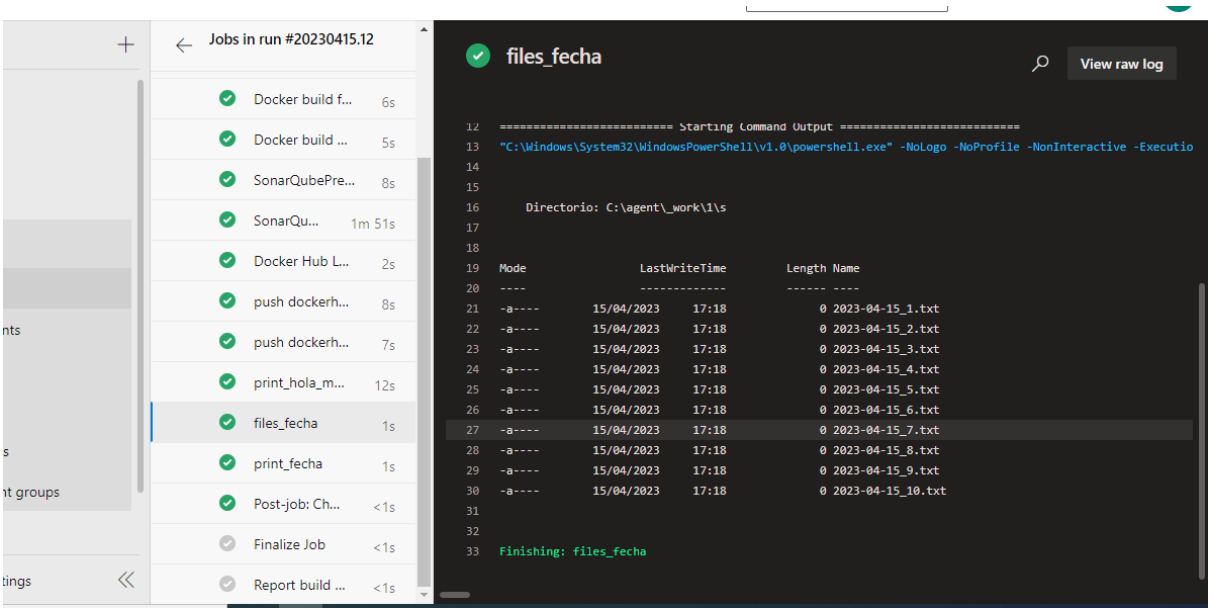
5 Dentro del pipeline ejecute lo siguiente en bash o powershell. a. Imprime `Hola Mundo` 10 veces en pantalla con un job paralelo. b. Script que cree 10 archivos con la fecha y luego lo imprima en consola

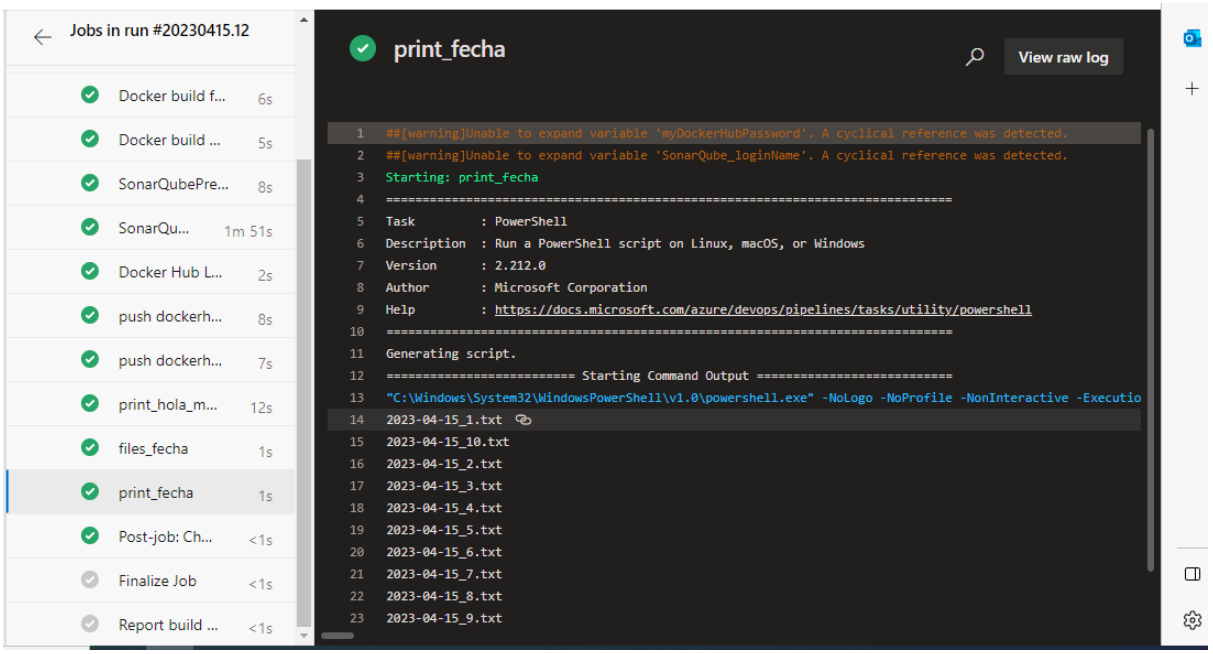
a) Imprime `Hola Mundo` 10 veces en pantalla

nota: es posible que aquí lo que se pida se usar una matrix strategy pero no estoy seguro pero solo he usado matrix estrategy una sola vez entonces casi no tengo experiencia con esa propiedad de los pipelines



b) b. Script que cree 10 archivos con la fecha y luego lo imprima en consola





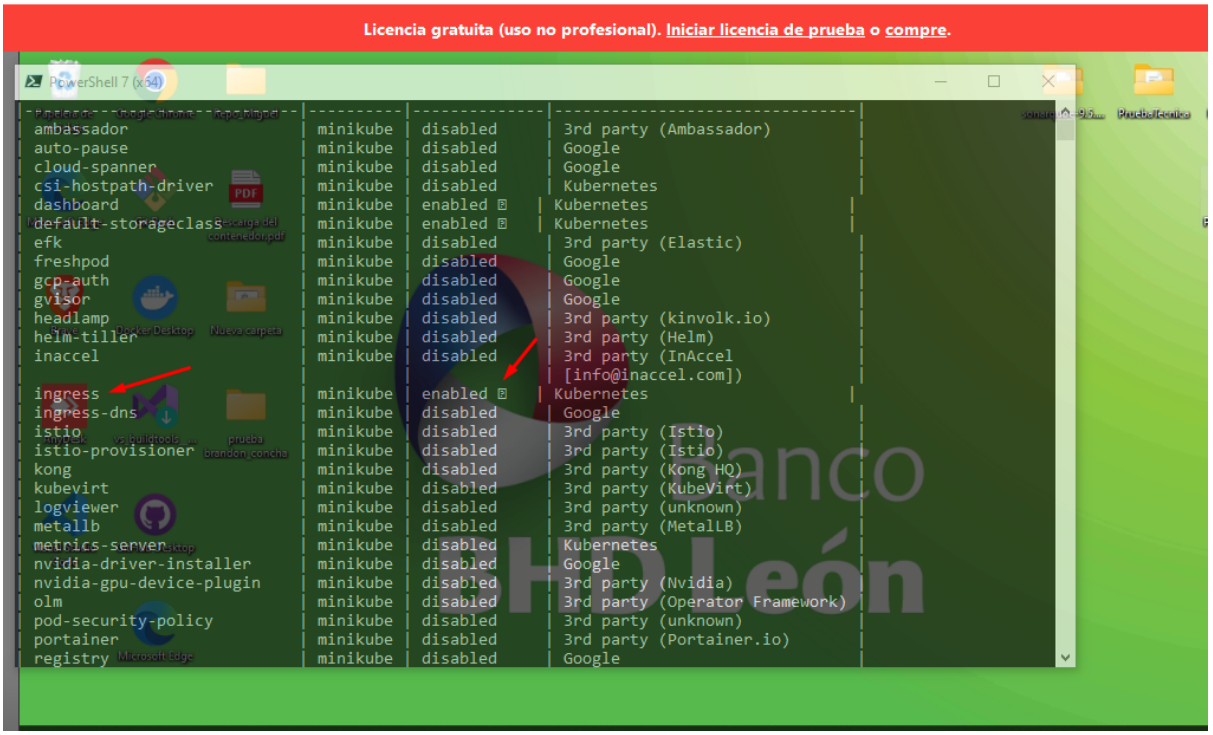
repositorio

levantar cluster de kubernetes

minikube start

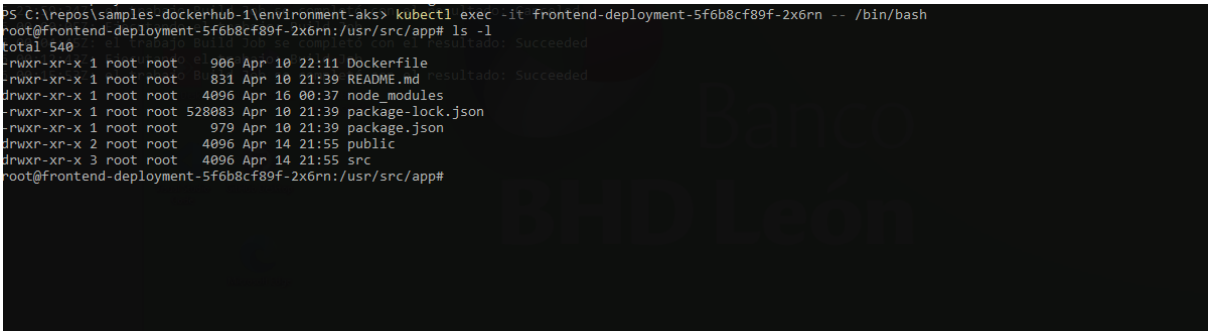
```
PowerShell 7 (x64)

PowerShell 7.3.3

   A new PowerShell stable release is available: v7.3.4
   Upgrade now, or check out the release page at:
     https://aka.ms/PowerShell-Release?tag=v7.3.4

PS C:\Users\Admin> minikube start
* minikube v1.26.1 en Microsoft Windows 11 Pro 10.0.22621 Build 22621
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Updating the running docker "minikube" container ...
* Preparando Kubernetes v1.24.3 en Docker 20.10.17...
  - Generando certificados y llaves
! initialization failed, will try again: wait: /bin/bash -c "sudo env PATH="/var/lib/minikube/binaries/v1.2
beadm init --config /var/tmp/minikube/kubeadm.yaml  --ignore-preflight-errors=DirAvailable--etc-kubernetes-
Available--var-lib-minikube,DirAvailable--var-lib-minikube-etcd,FileAvailable--etc-kubernetes-manifests-kub
aml,FileAvailable--etc-kubernetes-manifests-kube-apiserver.yaml,FileAvailable--etc-kubernetes-manifests-kub
manager.yaml,FileAvailable--etc-kubernetes-manifests-etcd.yaml,Port-10250,Swap,Mem,SystemVerification,FileC
sys-net-bridge-bridge-nf-call-iptables": Process exited with status 1
stdout:
[init] Using Kubernetes version: v1.24.3
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/var/lib/minikube/certs"
[certs] Using existing ca certificate authority
[certs] Using existing apiserver certificate and key on disk
```

minikube addons list para verificar ingress en enable
para habilitarlo minikube addons enable ingress

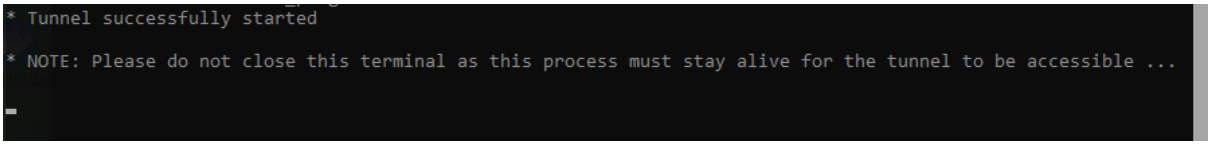


para acceder a cada pod

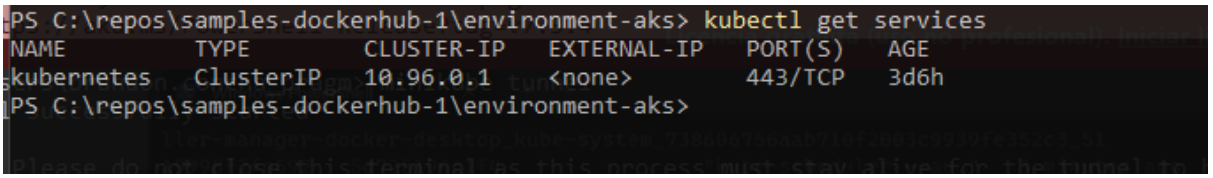
kubectl exec -it <frontend-deployment-5684f468b7-v2n4m> -- /bin/bash



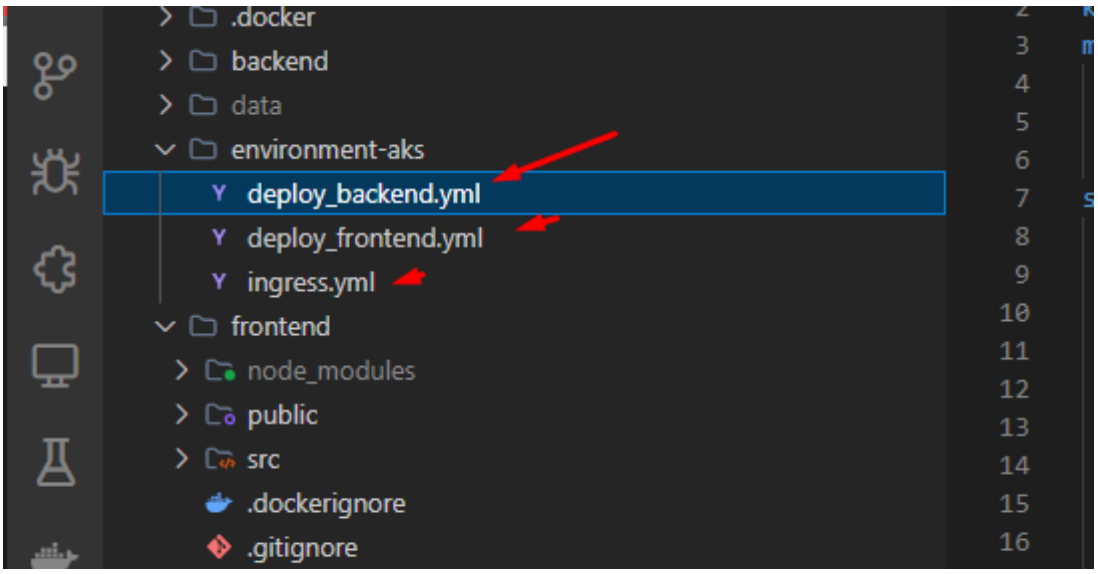
minikube tunnel
para permitir el acceso a los servicios expuestos en el clúster
minikube tunnel para crear una conexión de red entre el cluster de Kubernetes y la red local del host en el que se está ejecutando Minikube



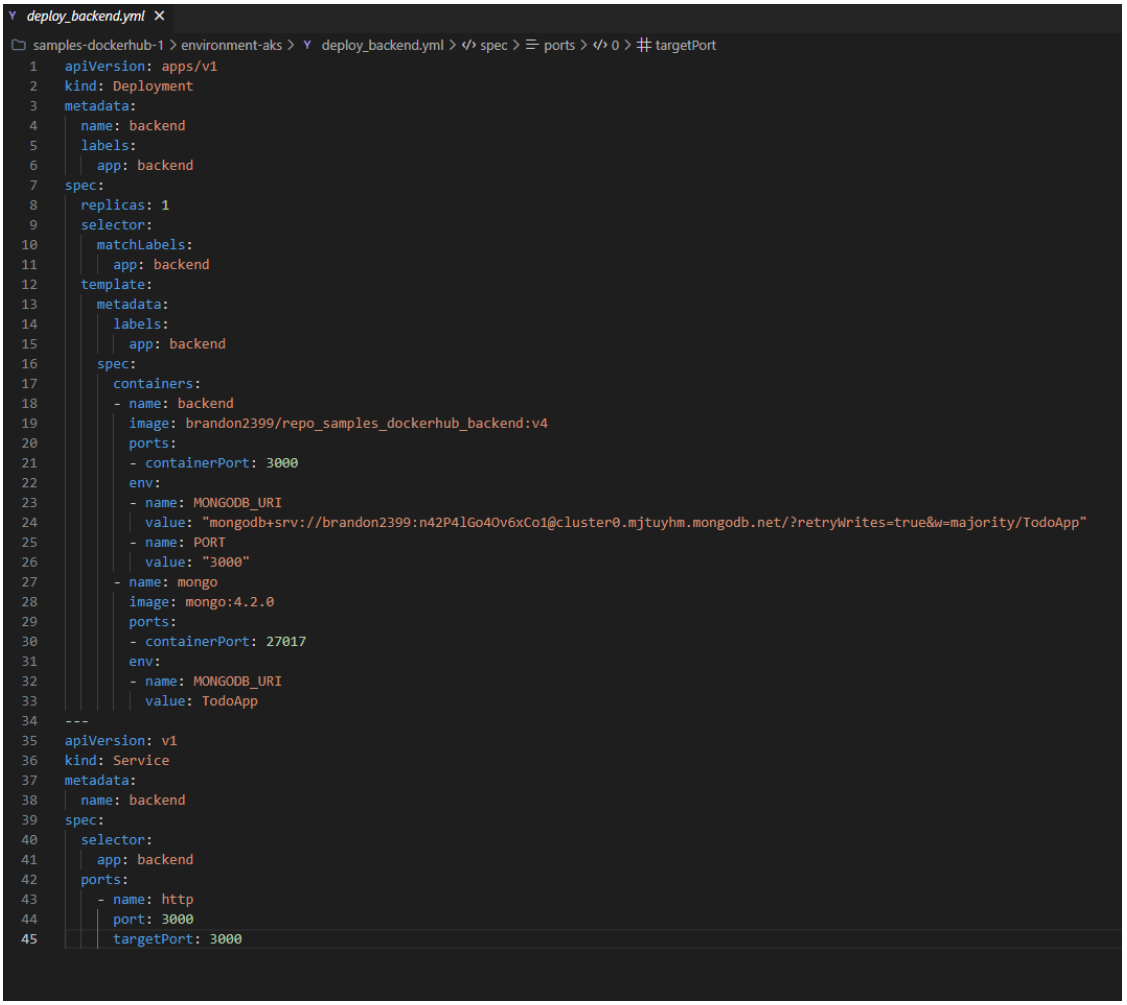
kubectl get services



ubico los archivos deploy en yml para realizar el despliegue de cada servicio una para el frontend otra para el backend y uno para el ingress para poder acceder al servicio desde afuera



deploy_backend.yml



deploy_backend.yml

samples-dockerhub-1 > environment-aks > deploy_backend.yml > spec > ports > 0 > targetPort

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: brandon2399/repo_samples_dockerhub_backend:v4
        ports:
        - containerPort: 3000
        env:
        - name: MONGODB_URI
          value: "mongodb+srv://brandon2399:n42P4lGo4Ov6xCo1@cluster0.mjtuyhm.mongodb.net/?retryWrites=true&w=majority/TodoApp"
        - name: PORT
          value: "3000"
      - name: mongo
        image: mongo:4.2.0
        ports:
        - containerPort: 27017
        env:
        - name: MONGODB_URI
          value: TodoApp
---
apiVersion: v1
kind: Service
metadata:
  name: backend
spec:
  selector:
    app: backend
  ports:
    - name: http
      port: 3000
      targetPort: 3000
```

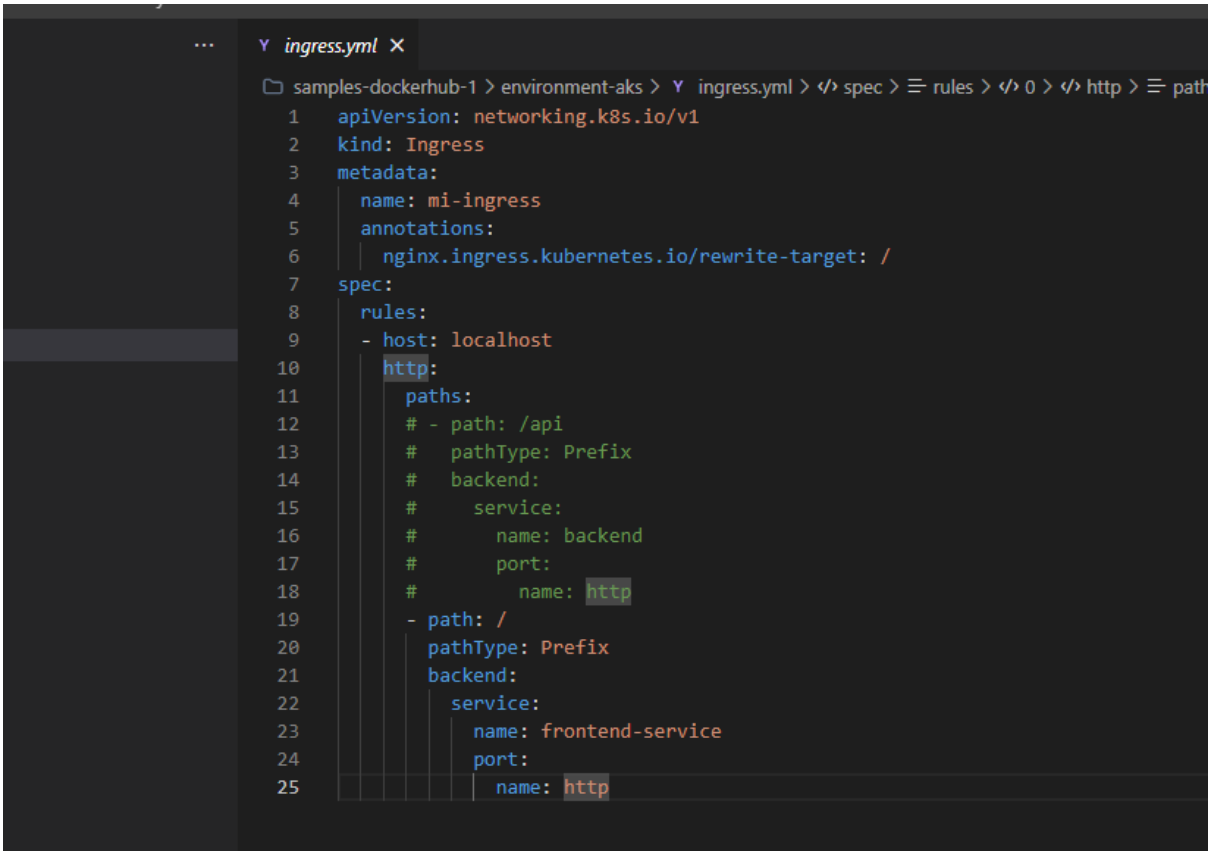
deploy_frontend.yml

deploy_frontend.yml

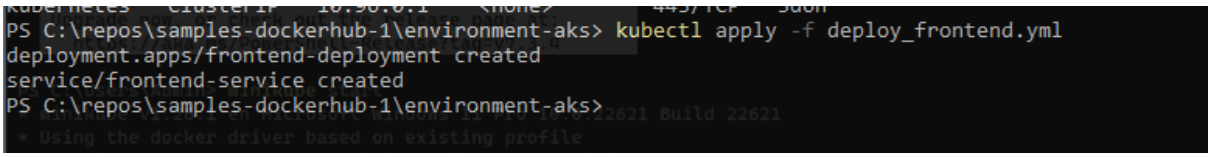
samples-dockerhub-1 > environment-aks > deploy_frontend.yml > spec > ten

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  selector:
    matchLabels:
      app: frontend
  replicas: 1
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: brandon2399/repo_samples_dockerhub_frontend:v2
        ports:
        - containerPort: 3000
        env:
        - name: API_URL
          value: http://backend:3000
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  selector:
    app: frontend
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 3000
```

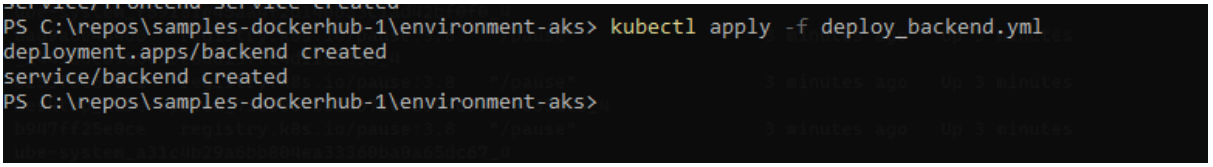
ingress.yml



kubectl apply -f deploy_frontend.yml



kubectl apply -f deploy_backend.yml



kubectl apply -f ingress.yml

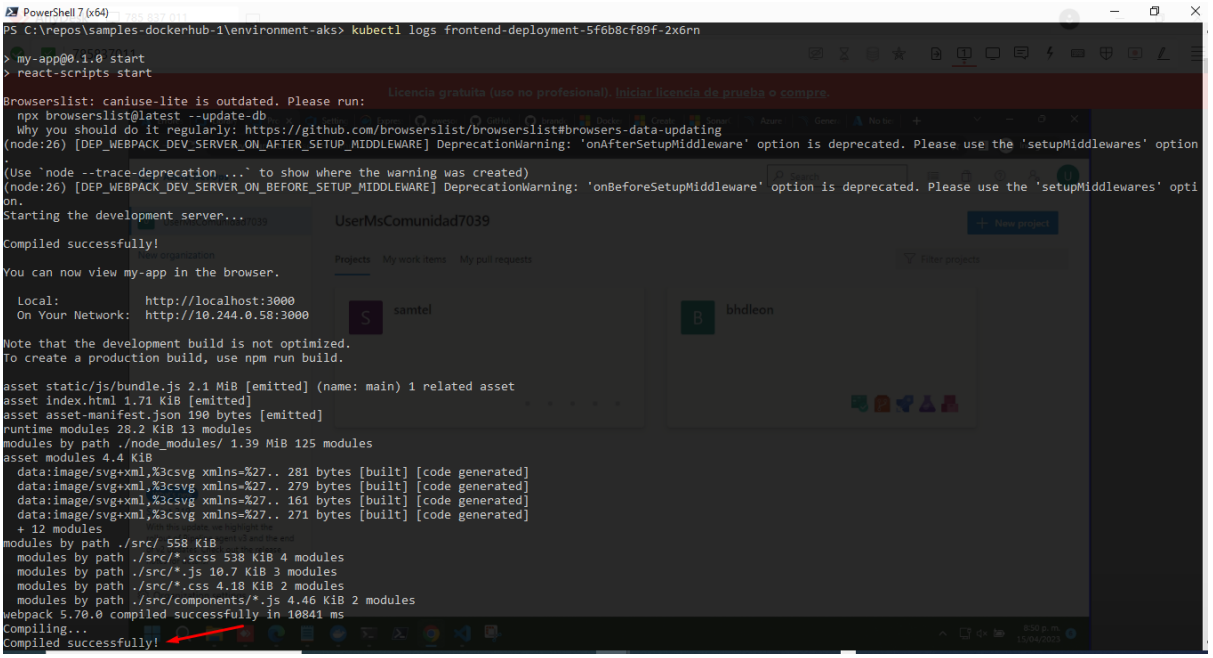
```
PS C:\repos\samples-dockerhub-1\environment-aks> kubectl apply -f ingress.yml
ingress.networking.k8s.io/mi-ingress created
PS C:\repos\samples-dockerhub-1\environment-aks>
```

kubectl get services

```
PS C:\repos\samples-dockerhub-1\environment-aks> kubectl get services
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
backend            ClusterIP   10.106.140.209   <none>        3000/TCP   113s
frontend-service   ClusterIP   10.99.217.97     <none>        80/TCP     9m24s
kubernetes         ClusterIP   10.96.0.1        <none>        443/TCP    3d7h
PS C:\repos\samples-dockerhub-1\environment-aks>
```

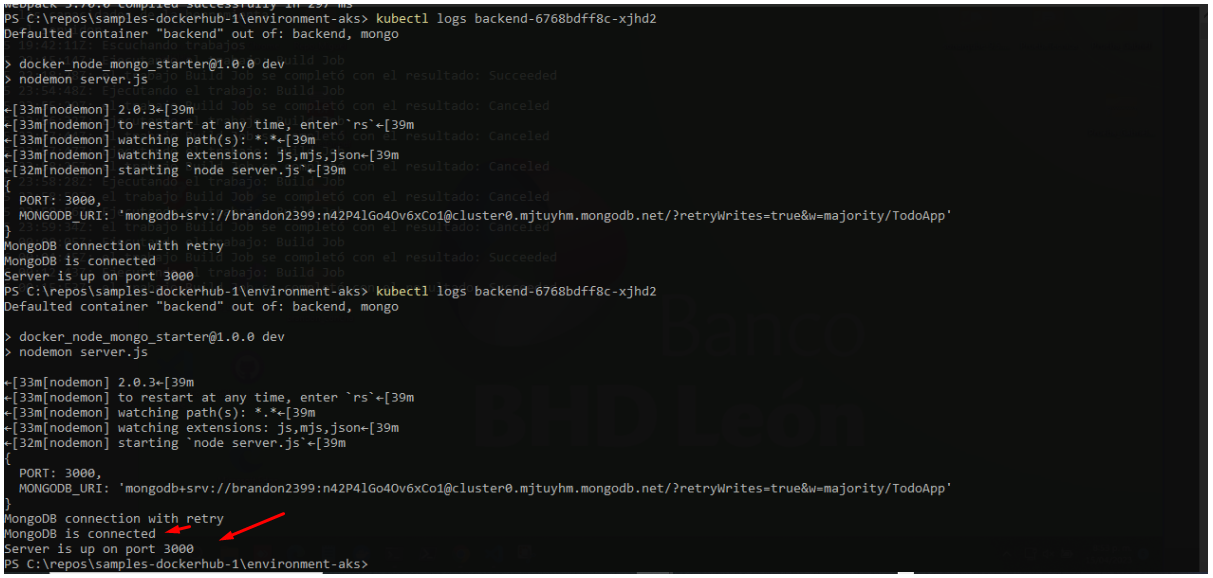
logs del pod frontend

kubectl logs <frontend-deployment-5f6b8cf89f-2x6rn>

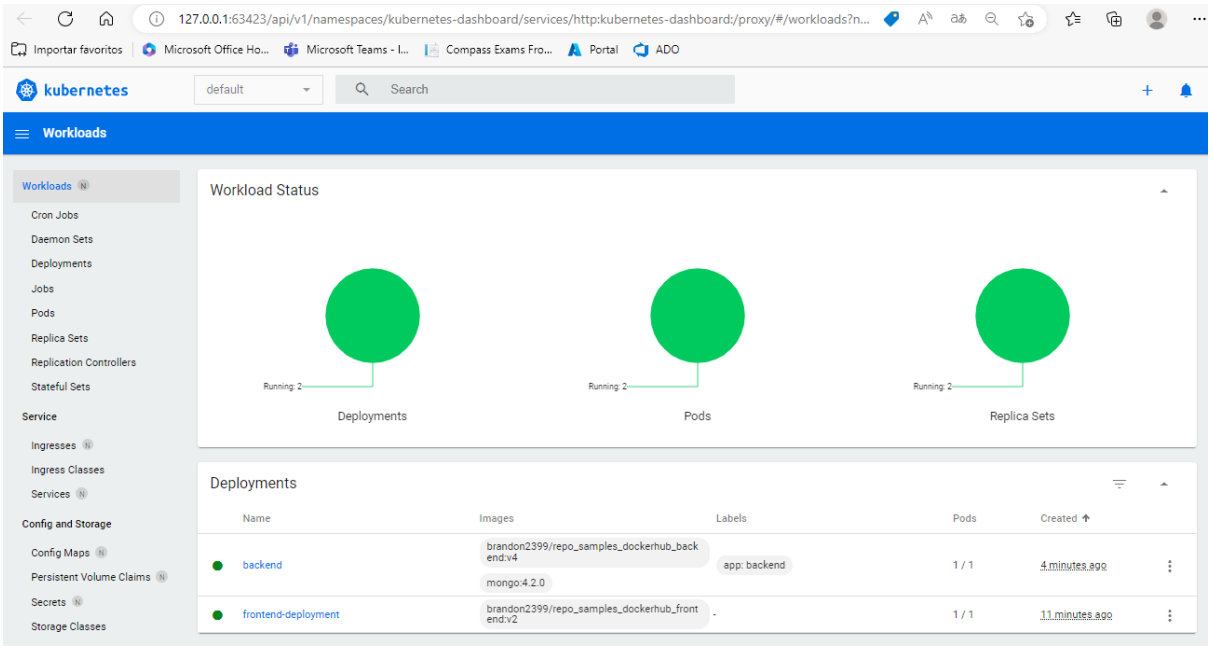


logs del pod backend

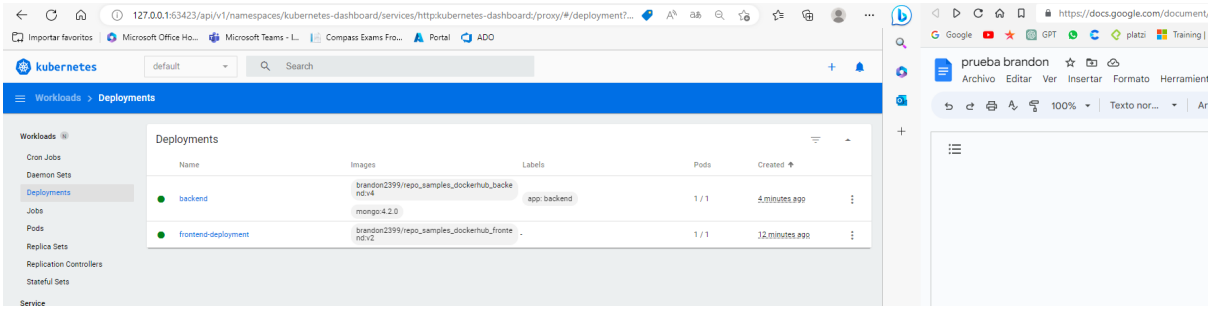
kubectl logs <backend-6768bdff8c-xjhd2>



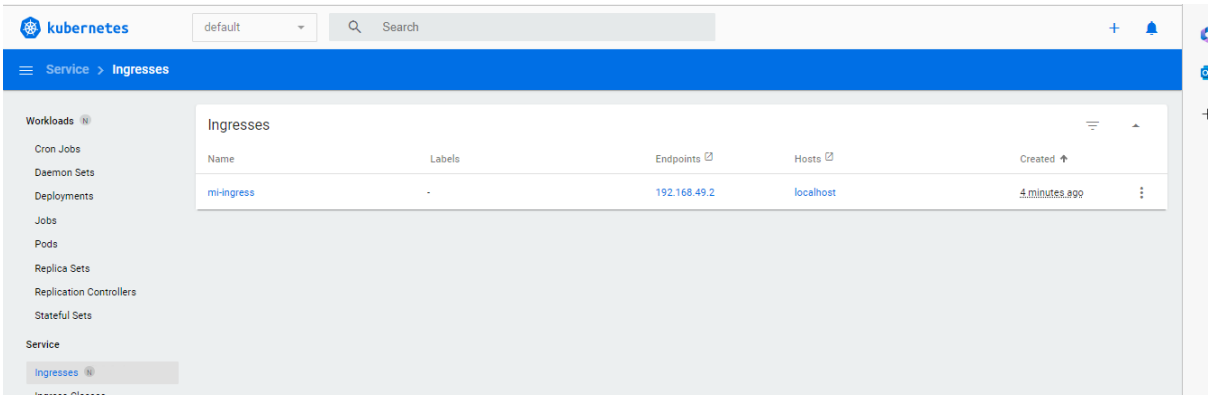
minikube dashboard



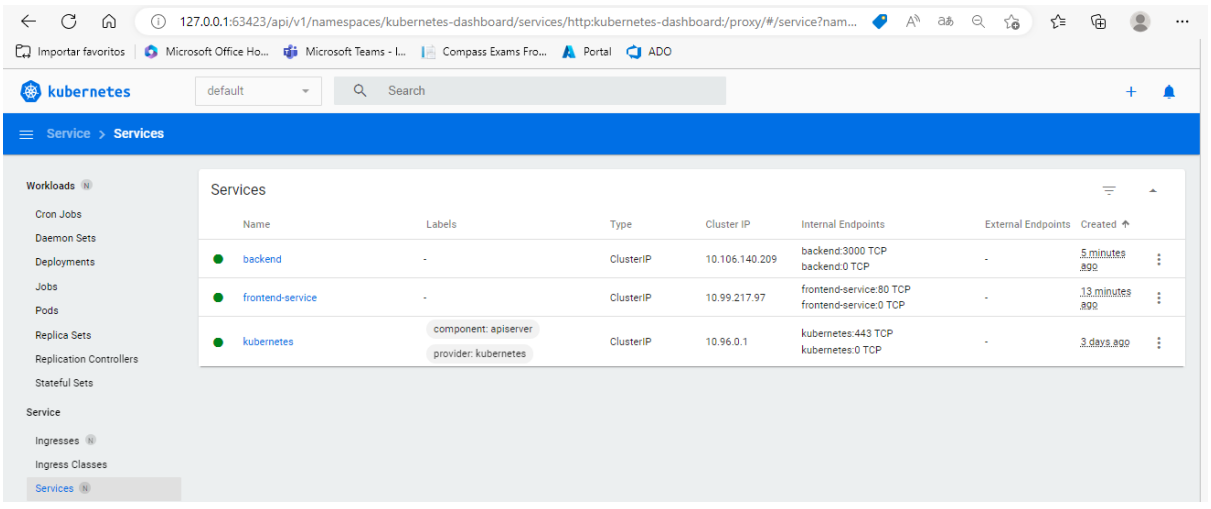
deployments



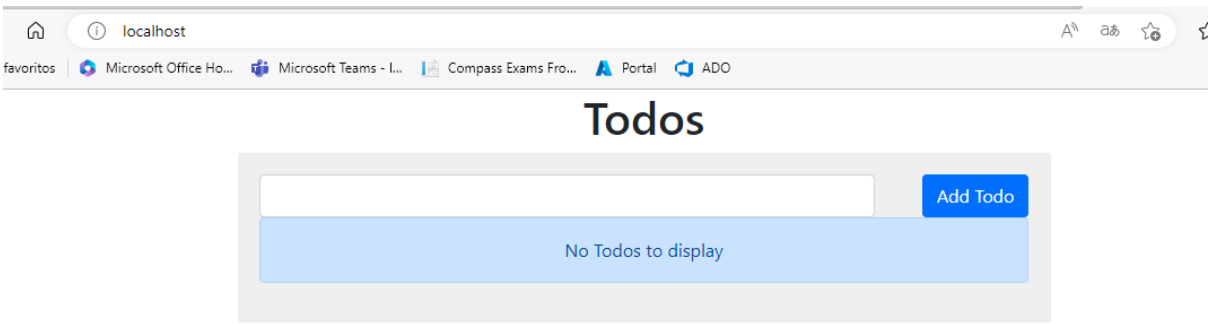
service ingress



services

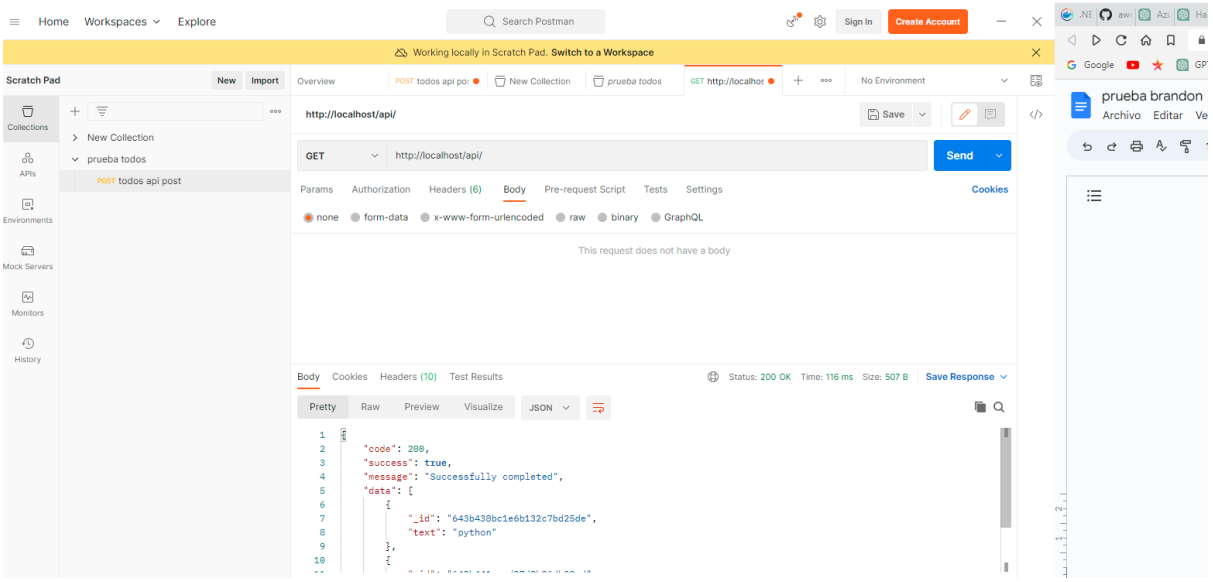


aplicación levantada desde kubernetes

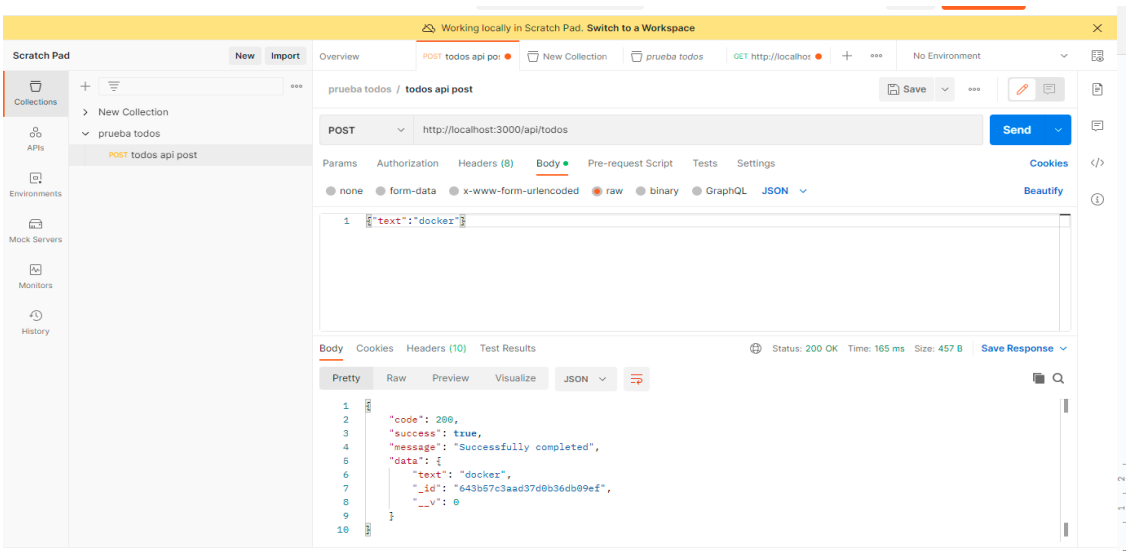


pruebas desde postman

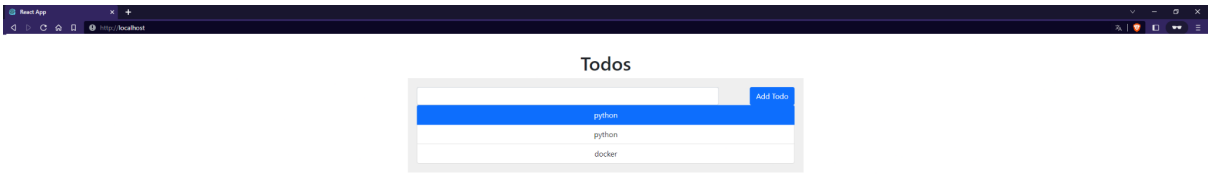
get http://localhost/api/



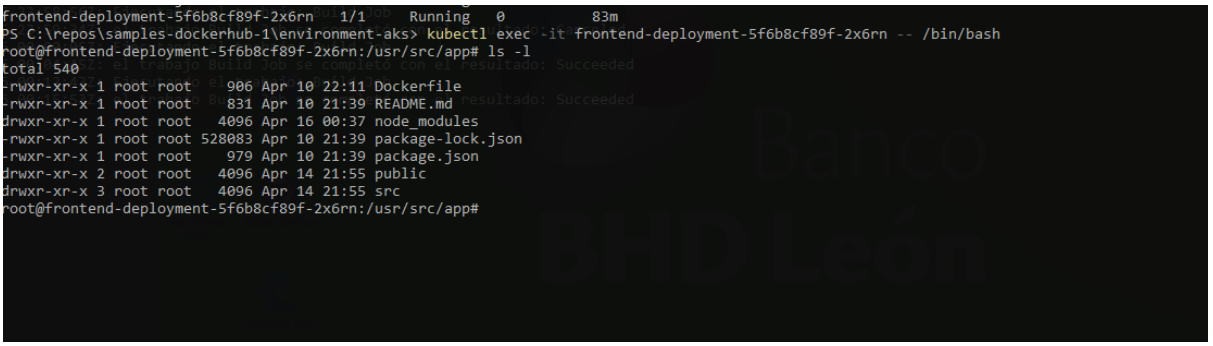
post http://localhost:3000/api/todos



localhost

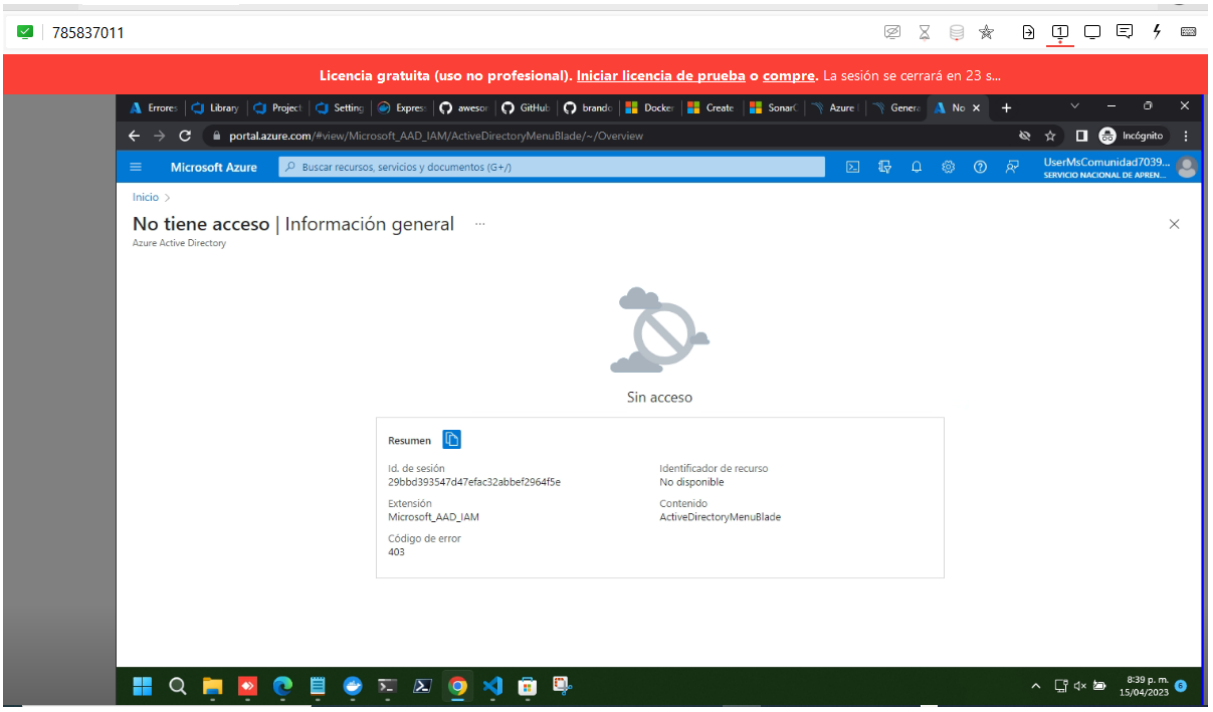


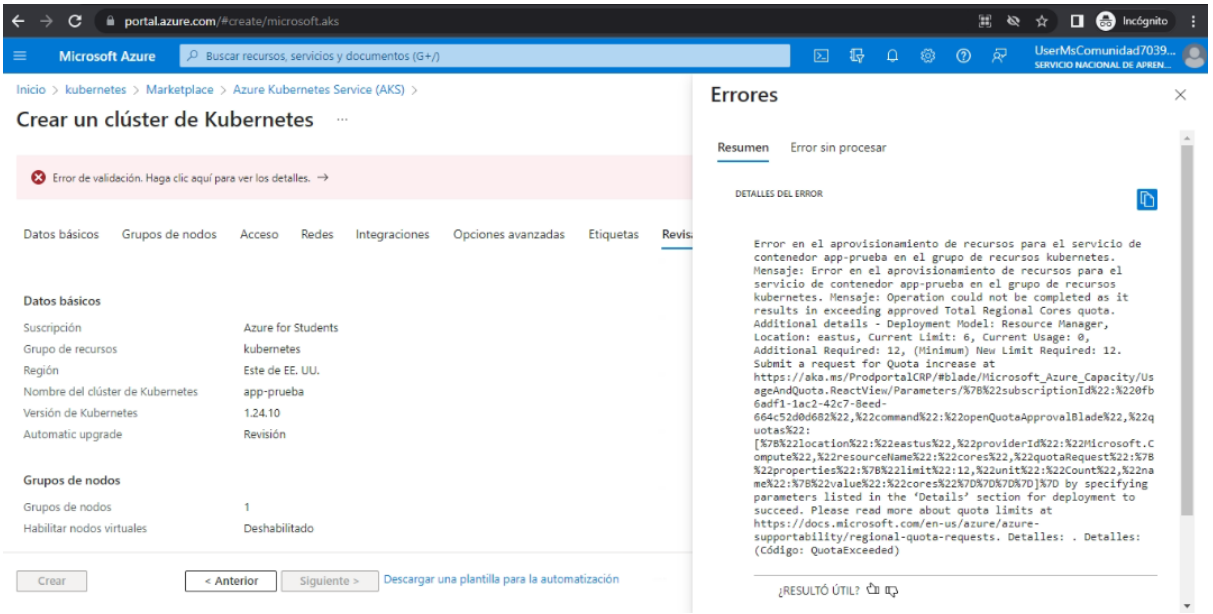
para acceder a cada pod
kubectl exec -it <frontend-deployment-5684f468b7-v2n4m> -- /bin/bash



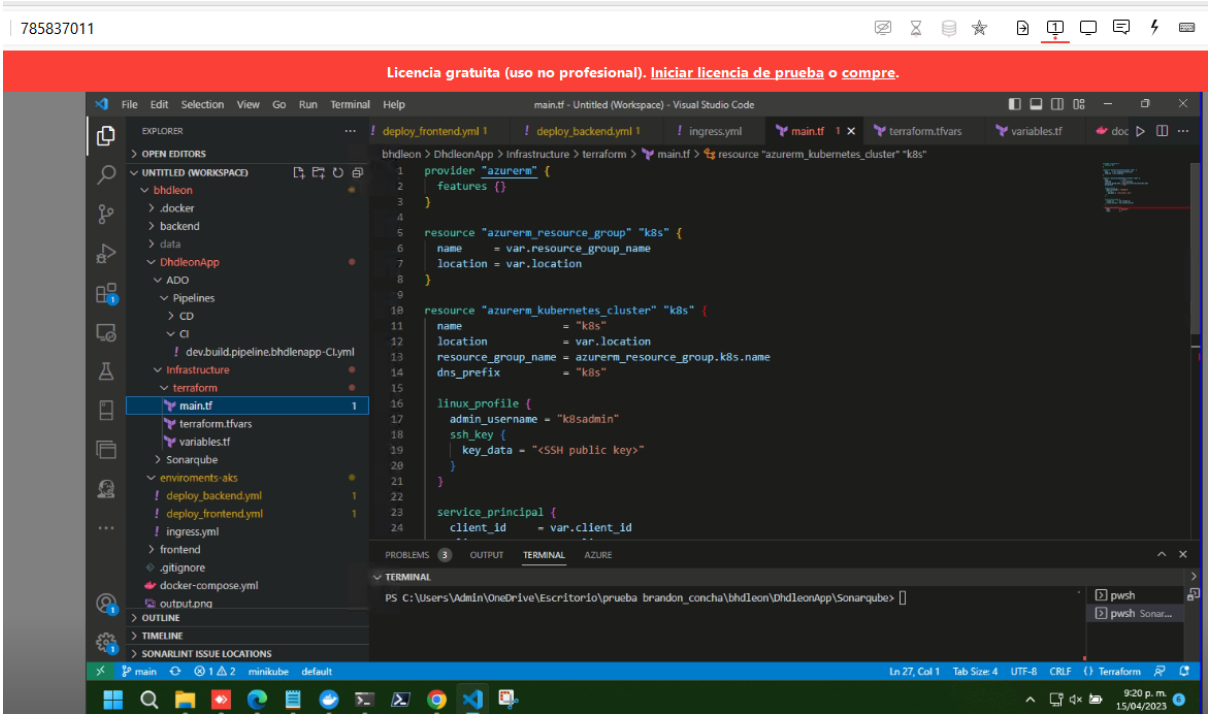
Notas aclaratorias para el Bonus

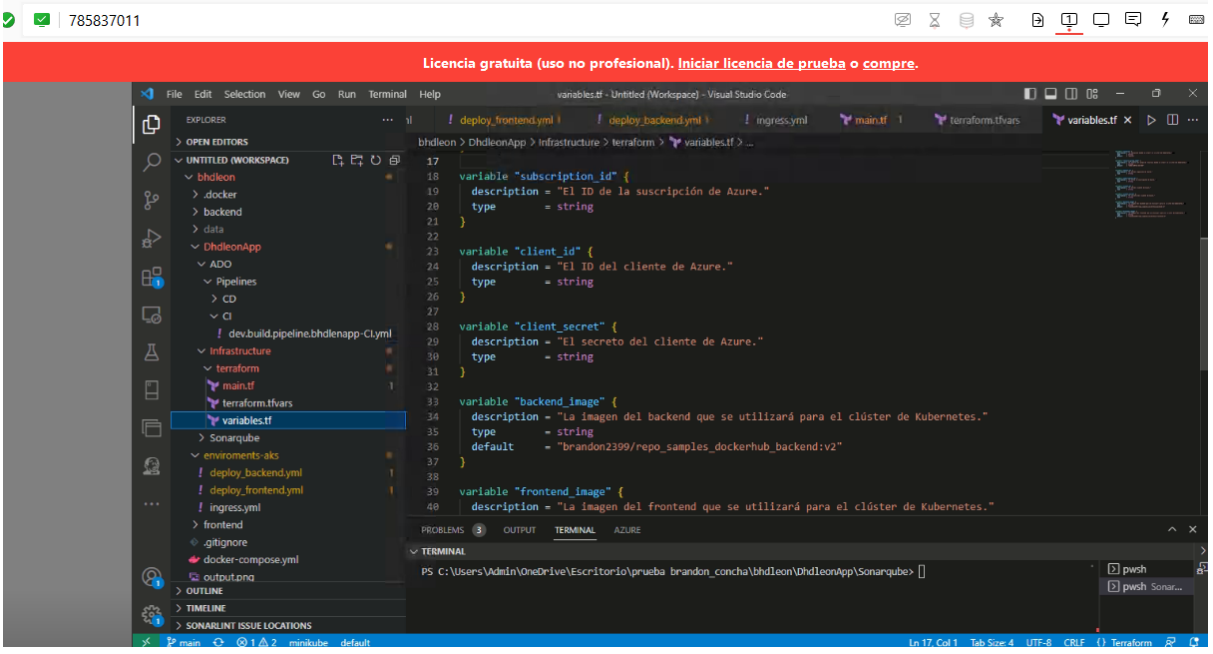
debido a que la cuenta educativa que logre conseguir de azure tiene una restricción en azure directory, intente utilizar otra opcion utilizando una autenticación diferente en Terraform, como la autenticación interactiva pero no me funciono



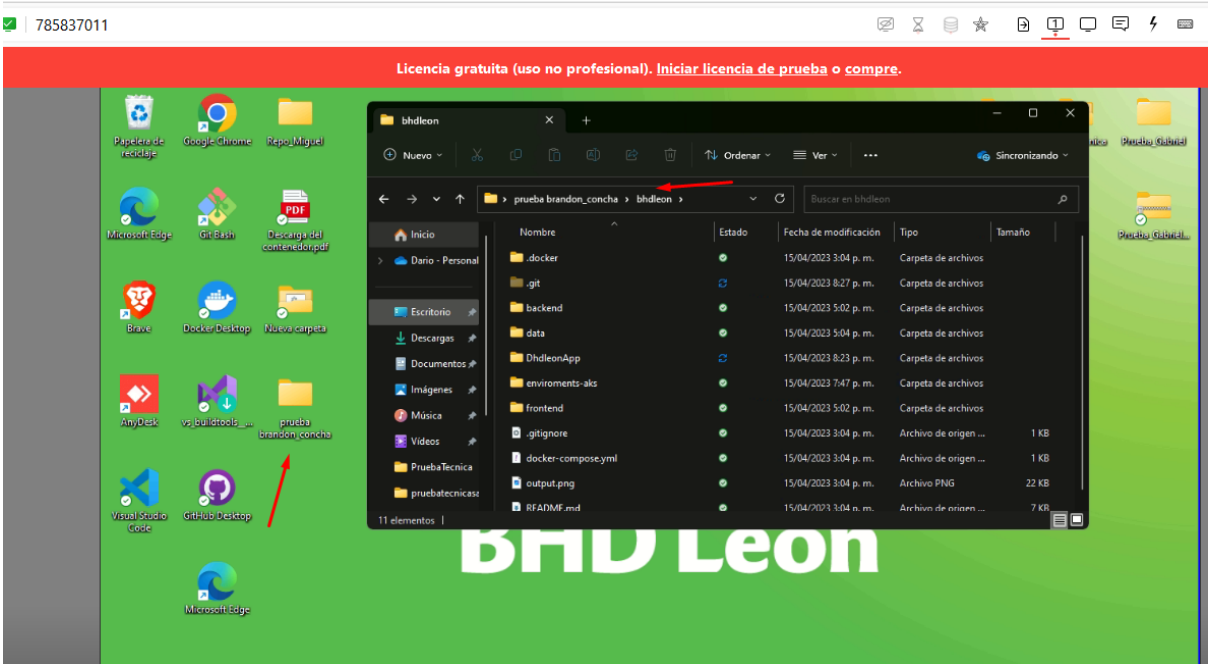


sin embargo se dejan los archivos creados de terrafom





se deja carpeta con todos los archivos del repo



este documento tiene una carpeta compartida con el pdf de la realizacion de la prueba con scree y videos

https://docs.google.com/document/d/1fXS_GXinMzqtbK7FFuQX9pQlF5wNWKm_JWD8Ca0XwRw/edit

repositorio publico

https://github.com/brandon2399/docker-samples-mas-kubernetes