

DESPLIEGUE DE UN MODELO DE CLASIFICACIÓN E IMÁGENES EN AKS

FIRST STEP: SET UP ENVIROMENT

I used a vagrant virtual machine that runs Ubuntu 22.04

[Dockerfile](#)

```
# -- mode: ruby --  
# vi: set ft=ruby :
```

```
Vagrant.configure("2") do |config|  
  config.vm.synced_folder ".",  
  "/home/vagrant/shared", type: "virtualbox"  
  
  if Vagrant.has_plugin? "vagrant-vbguest"  
    config.vbguest.no_install = true  
    config.vbguest.auto_update = false  
    config.vbguest.no_remote = true  
  end  
  
  config.vm.define :clienteUbuntu do |clienteUbuntu|  
    clienteUbuntu.vm.box = "bento/ubuntu-22.04"  
    clienteUbuntu.vm.network :private_network, ip: "192.168.100.2"  
    clienteUbuntu.vm.hostname = "clienteUbuntu"  
  end  
  
  config.vm.define :servidorUbuntu do |servidorUbuntu|  
    servidorUbuntu.vm.box = "bento/ubuntu-22.04"  
    servidorUbuntu.vm.network :private_network, ip: "192.168.100.3"  
    servidorUbuntu.vm.hostname = "servidorUbuntu"  
    servidorUbuntu.vm.provision "shell", path: "script.sh"  
    servidorUbuntu.vm.provider "virtualbox" do |v|
```

```
        v.cpus = 2
        v.memory = 4072
    end
end
end
```

You must install

INSTALL DOCKER

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2
podman-docker containerd runc; do sudo apt-get remove $pkg; done

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \
    "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin

sudo docker info | more
```

INSTALL AZURE CLI

```
sudo apt-get update

sudo apt-get install ca-certificates curl apt-transport-https lsb-
release gnupg

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

```
gpg --dearmor | \
sudo tee /etc/apt/trusted.gpg.d/microsoft.gpg > /dev/null
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
sudo apt-get update
sudo apt-get install azure-cli
az version
```

INSTALL KUBECTL

```
curl -LO https://dl.k8s.io/release/\$\(curl -L -s https://dl.k8s.io/release/stable.txt\)/bin/linux/amd64/kubectl
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```

SECOND STEP: BUILD DOCKER IMAGE

Paso 1: Preparar la Aplicación

1. **Desarrolla tu Aplicación:** Asegúrate de que tu aplicación de clasificación de imágenes esté desarrollada y probada localmente. Por simplicidad, asumiremos que utilizas Python y algún framework de aprendizaje automático como TensorFlow o PyTorch.

App.py

```
from gluoncv.model_zoo import get_model
import matplotlib.pyplot as plt
from mxnet import gluon, nd, image
from mxnet.gluon.data.vision import transforms
from gluoncv import utils
from PIL import Image
import io
import flask
app = flask.Flask(__name__)
```

```

@app.route("/predict",methods=["POST"])
def predict():
    if flask.request.method == "POST":
        if flask.request.files.get("img"):
            img =
Image.open(io.BytesIO(flask.request.files["img"].read()))
            transform_fn = transforms.Compose([
                transforms.Resize(32),
                transforms.CenterCrop(32),
                transforms.ToTensor(),
                transforms.Normalize([0.4914, 0.4822, 0.4465], [0.2023,
0.1994, 0.2010]))
            img = transform_fn(nd.array(img))
            net = get_model('cifar_resnet20_v1', classes=10,
pretrained=True)

            class_names = ['airplane', 'automobile', 'bird', 'cat',
'deer',
                        'dog', 'frog', 'horse', 'ship', 'truck']
            ind = nd.argmax(pred, axis=1).astype('int')
            prediction = ('The input picture is classified as [%s],
with probability %.3f. '%
                        (class_names[ind.asscalar()],
nd.softmax(pred)[0][ind].asscalar()))
            return prediction

if __name__ == '__main__':
    app.run(host='0.0.0.0')

```

2. **Crear un Dockerfile:** Necesitas contenerizar tu aplicación para desplegarla en AKS. Crea un **Dockerfile** en la raíz de tu proyecto. Aquí hay un ejemplo básico para una aplicación Python:

Dockerfile

```
FROM python:3.8-slim
WORKDIR /app
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

Asegúrate de ajustar este **Dockerfile** según las necesidades específicas de tu aplicación. En el mismo directorio donde está el dockerfile asegúrate de crear los archivos requirements.txt y app.py

Requirements.txt

flask

gluoncv

matplotlib

mxnet

requests

Pillow

Paso 2: Construir y Publicar la Imagen Docker

3. **Construye la Imagen Docker:** Utiliza el comando **docker build** para construir tu imagen Docker. Esto puede tomar tiempo dependiendo de los recursos que le hayas asignado a tu máquina virtual, entre más RAM asignada el proceso será más rápido

```
docker build -t myimageclassifier:v1 .
```

4. **Sube la Imagen a Azure Container Registry (ACR):** Primero, crea un Azure Container Registry desde el portal de Azure o utilizando Azure CLI. Luego, autentica Docker con ACR y sube tu imagen.

```
az acr login --name MyRegistry docker tag myimageclassifier:v1
myregistry.azurecr.io/myimageclassifier:v1 docker push
myregistry.azurecr.io/myimageclassifier:v1
```

Alternativa: Una vez construida la imagen puedes utilizar los siguientes comandos, asegúrate de cambiar los nombres.

```
az login
```

```
az acr create --resource-group myResourceGroup --name myRegistryName  
--sku Basic
```

```
az acr login --name myRegistryName
```

```
az acr list --resource-group pruebalunes --query  
"[].{acrLoginServer:loginServer}" --output table
```

```
docker tag localImageName:tag  
myRegistryName.azurecr.io/localImageName:tag
```

```
docker push myRegistryName.azurecr.io/localImageName:tag
```

5. **Configura kubectl para Usar tu Clúster AKS:** Obtén las credenciales para tu clúster AKS y configura **kubectl** para usarlas.

```
az aks get-credentials --resource-group MyResourceGroup --name  
MyAKSCluster
```

Paso 4: Desplegar la Aplicación en AKS

7. **Crea un Archivo de Despliegue para Kubernetes:** Crea un archivo **deployment.yaml** que defina tu despliegue y servicio en Kubernetes

```
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: kubermatic-dl-deployment  
spec:  
  selector:  
    matchLabels:  
      app: kubermatic-dl  
  replicas: 3  
  template:  
    metadata:  
      labels:  
        app: kubermatic-dl
```

```
spec:
  containers:
  - name: kubermatic-dl
    image: kubermatic00/kubermatic-dl:latest
    imagePullPolicy: Always
    ports:
    - containerPort: 8080.
```

8. **Despliega tu Aplicación:** Utiliza **kubectl** para aplicar tu archivo de despliegue en AKS.

```
kubectl apply -f deployment.yaml
```

9. To expose your deployment to the outside world, you need a service object that will create an externally reachable IP for your container:

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

10. **Accede a tu Aplicación:** Una vez que el servicio esté activo, obtén la IP pública de tu servicio y accede a tu aplicación de clasificación de imágenes.

```
kubectl get service kubermatic-dl
```

```
vagrant@servidorUbuntu:~$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubermatic-dl-deployment	LoadBalancer	10.0.25.191	172.214.63.244	80:32336/TCP	25m
kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	82m

To check the cluster information use

```
kubectl cluster-info
```

```
vagrant@servidorUbuntu:~$ kubectl cluster-info
Kubernetes control plane is running at https://primero-dns-598fmjvh.hcp.eastus.azmk8s.io:443
CoreDNS is running at https://primero-dns-598fmjvh.hcp.eastus.azmk8s.io:443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://primero-dns-598fmjvh.hcp.eastus.azmk8s.io:443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

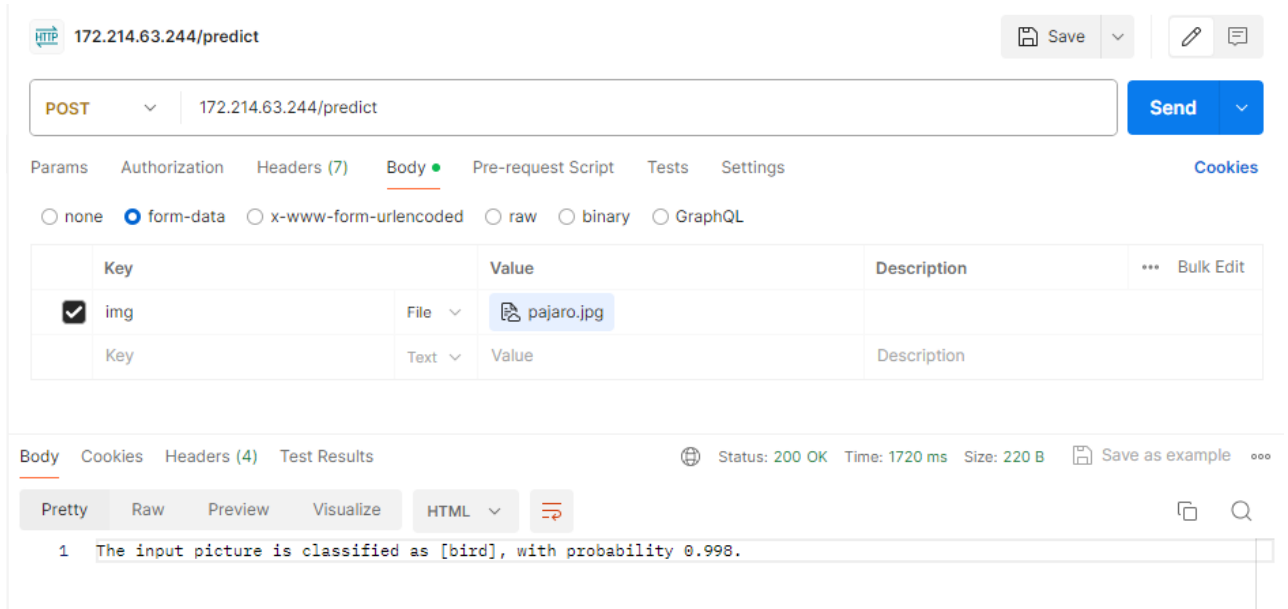
THIRD STEP: TEST YOUR APPLICATION

Use postman to send requests to your API

Go to postman: <https://www.postman.com/>

Asegúrate de utilizar tu dirección IP externa del clúster. En mi caso es 172.214.63.244

El campo key debe ser siempre “img”



The screenshot shows the Postman interface for a POST request to the endpoint `172.214.63.244/predict`. The request body is configured as form-data with a single key-value pair: `img` (key) and `pajaro.jpg` (value, represented by a file icon). The response status is `200 OK` with a time of `1720 ms` and a size of `220 B`. The response body, viewed in HTML, contains the text: `1 The input picture is classified as [bird], with probability 0.998.`

Key	Value	Description
<input checked="" type="checkbox"/> img	File <code>pajaro.jpg</code>	
Key	Text Value	Description

Imagen enviada al modelo



SECOND CHALLENGE

I created a simple webpage that shows my CV, to do that I followed the following steps.

Create your app: It could be static or dynamic, it can be build for example in HTML/CSS, JavaScript, React, Angular, Flask, Django, etc.

Contenedoriza tu Aplicación: Crea un Dockerfile en la raíz de tu proyecto. Aquí hay un ejemplo básico para una aplicación estática HTML/CSS el cual use

Dockerfile

Usa una imagen base de nginx para servir contenido estático

FROM nginx:alpine

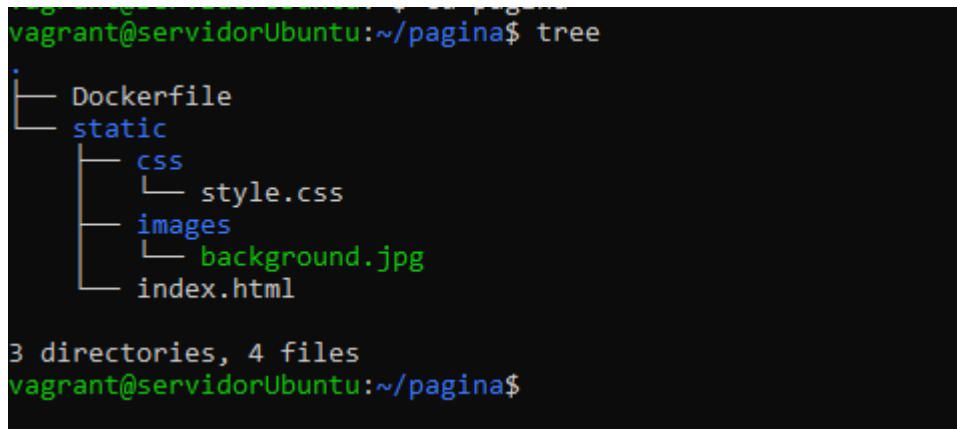
Copia los archivos estáticos al directorio de nginx

COPY ./static /usr/share/nginx/html

Expone el puerto 80

EXPOSE 80

Organiza tu estructura de directorios

A terminal window with a dark background. The prompt is 'vagrant@servidorUbuntu:~/pagina\$'. The command 'tree' has been executed, showing a directory tree. The root directory is '.', which contains 'Dockerfile' and 'static'. The 'static' directory contains 'css' (with 'style.css'), 'images' (with 'background.jpg'), and 'index.html'. Below the tree, it says '3 directories, 4 files'. The prompt is now 'vagrant@servidorUbuntu:~/pagina\$'.

```
vagrant@servidorUbuntu:~/pagina$ tree
.
├── Dockerfile
└── static
    ├── css
    │   └── style.css
    ├── images
    │   └── background.jpg
    └── index.html

3 directories, 4 files
vagrant@servidorUbuntu:~/pagina$
```

Construye la Imagen Docker: En la terminal, navega al directorio de tu proyecto y ejecuta el comando para construir tu imagen Docker:

```
docker build -t mihojadevida:v1 .
```

Prueba Localmente: Usa la IP de tu maquina local, en este caso es 192.168.100.3 (localhost)

```
docker run -d -p 8080:80 mihojadevida:v1
```

Navega a <http://localhost:8080> para ver tu aplicación en acción.

Sube la imagen que creaste a tu repositorio de Docker hub

```
docker login
```

```
docker tag <nombre_imagen_local>:<tag>  
<tu_usuario_dockerhub>/<nombre_repositorio_dockerhub>:<tag>
```

```
docker push  
<tu_usuario_dockerhub>/<nombre_repositorio_dockerhub>:<tag>
```

```
vagrant@servidorUbuntu:~$ docker tag mihojadevida:v1 mikepa22/mihojadevida:v1  
vagrant@servidorUbuntu:~$ docker push mikepa22/mihojadevida:v1  
The push refers to repository [docker.io/mikepa22/mihojadevida]  
7c026ab24487: Pushed  
667a247707f0: Mounted from library/nginx  
d8527026595f: Mounted from library/nginx  
2593b08e5428: Mounted from library/nginx  
9909978d630d: Mounted from library/nginx  
c5140fc719dd: Mounted from library/nginx  
3137f8f0c641: Mounted from library/nginx  
718db50a47c0: Mounted from library/nginx  
aedc3bda2944: Mounted from library/nginx  
v1: digest: sha256:08dcd1d2586664becf9e722db1f094d46fb44855a15ccfb012a6fcb33095d51b size: 2199
```

Obtén las Credenciales de AKS para kubectl: Se debe tener un clúster creado y desplegado en AKS

```
az aks get-credentials --resource-group MiGrupoDeRecursos --name  
miClusterAKS
```

CUIDADO ! Cuando en la misma cuenta se han creado varios clústeres, es importante asegurarse que el kubectl este apuntado al clúster indicado. Puedes hacer la verificación siguiendo estos comandos.

Primero, asegúrate de que estás correctamente conectado a tu clúster de AKS y que kubectl está configurado para comunicarse con él. Puedes hacerlo con el siguiente comando, que te mostrará los clusters a los que kubectl puede conectarse:

```
kubectl config get-contexts
```

Y luego asegúrate de que estás usando el contexto correcto para tu clúster de AKS con:

```
kubectl config use-context <nombre_de_tu_contexto>
```

Reemplaza <nombre_de_tu_contexto> con el nombre del contexto de tu clúster de AKS.

Este comando configura kubectl para usar las credenciales de tu nuevo cluster AKS, permitiéndote gestionarlo. La opción `--overwrite-existing` asegura que cualquier configuración previa de kubectl para otros clusters sea reemplazada o actualizada.

```
az aks get-credentials --resource-group <grupo_de_recursos_nuevo> --  
name <nombre_del_nuevo_cluster> --overwrite-existing
```

Verificar el Contexto Actual de kubectl

```
kubectl config current-context
```

Si todo esta correcto puedes desplegar la aplicación en tu clúster habiendo creado el archivo .yaml . Nótese que en el archivo de ejemplo, en la sección de imagen, se utiliza la imagen que se subió al repositorio de DockerHUB.

```
kubectl apply -f deployment.yaml
```

[deploymentcv.yaml](#)

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: mihojadevida-deployment
```

```
spec:
```

```
  replicas: 2
```

```
  selector:
```

```
    matchLabels:
```

```
      app: mihojadevida
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: mihojadevida
```

```
    spec:
```

```
      containers:
```

```
        - name: mihojadevida
```

```
          image: mikepa22/mihojadevida:v1
```

```
          ports:
```

```
            - containerPort: 80
```

```
---
```

```
apiVersion: v1
```

```
kind: Service

metadata:
  name: mihojadevida-service

spec:
  type: LoadBalancer

  ports:
    - port: 80

  selector:
    app: mihojadevida
```

Despliega tu Aplicación en AKS:

```
kubectl apply -f deployment.yaml
```

Accede a tu Aplicación: Después de algunos minutos, el servicio debería obtener una dirección IP pública. Encuentra esta dirección con:

```
kubectl get service
```

Usa la dirección IP mostrada para acceder a tu hoja de vida desde cualquier navegador.

```
service/mihojadevida-service created
vagrant@servidorUbuntu:~$ kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes           ClusterIP     10.0.0.1      <none>         443/TCP          98m
mihojadevida-service LoadBalancer  10.0.119.45   57.151.8.104   80:32009/TCP     30s
vagrant@servidorUbuntu:~$
```

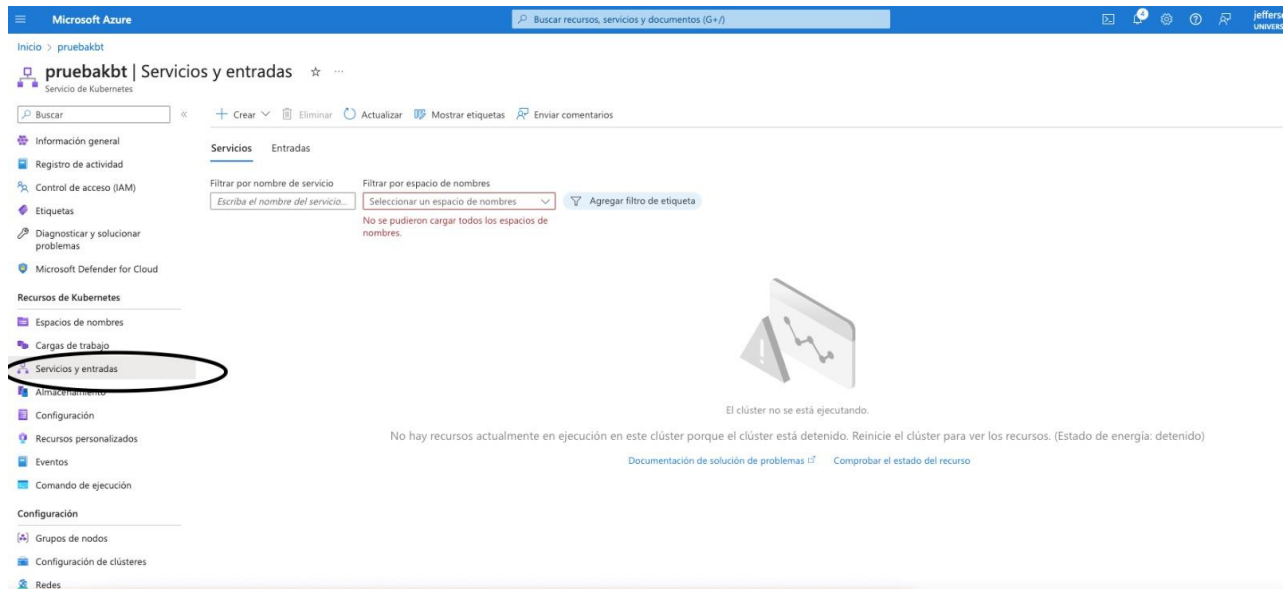
Verifica el estado de tus pods: Si están en estado running todo esta bien

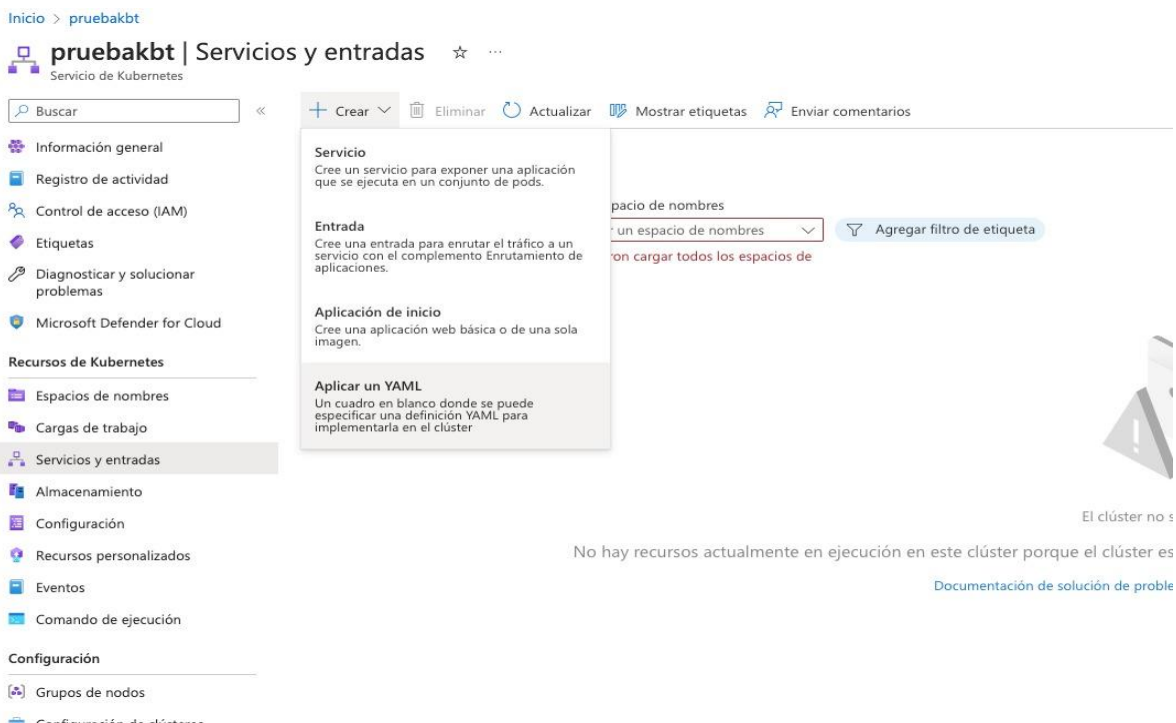
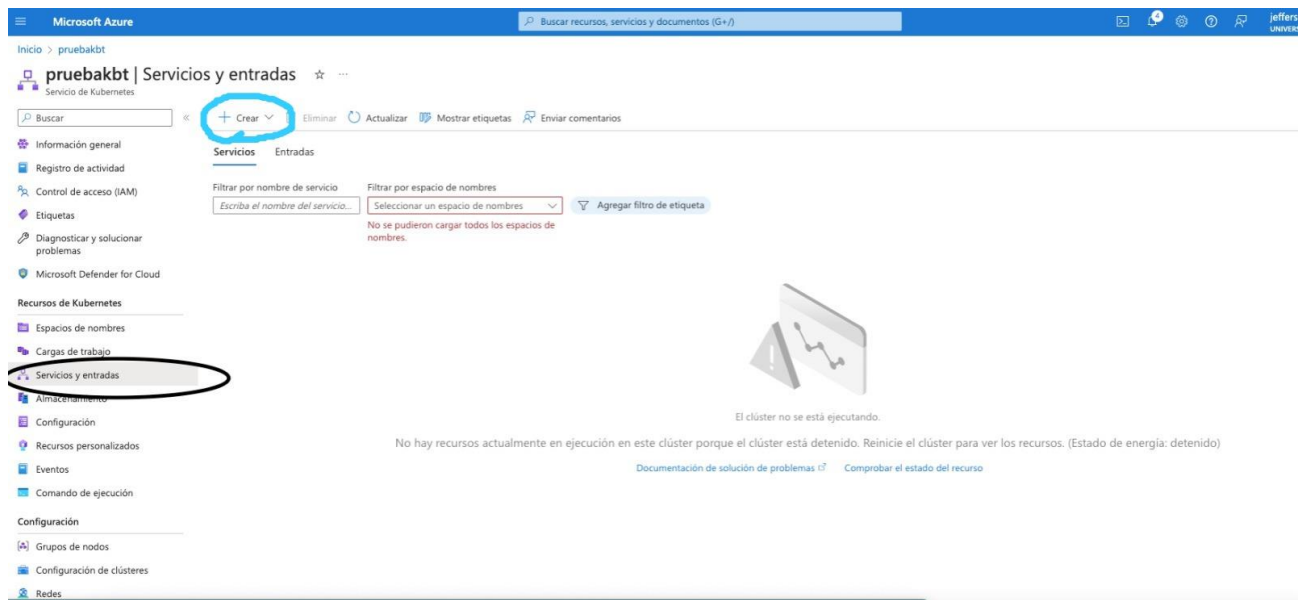
```
vagrant@servidorUbuntu:~$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
mihojadevida-deployment-66cbcd8b86-gnbck 1/1     Running   0          173m
mihojadevida-deployment-66cbcd8b86-rbj2h 1/1     Running   0          173m
vagrant@servidorUbuntu:~$
```



Para subir dos servicios en el mismo clúster sigue los siguientes pasos.

Primero ingresa a tu cluster de kuberetes .





En el cuadro en blanco deberás pegar tu archivo de despliegue .yaml. Luego darás clic en ejecutar. El despliegue puede tardar hasta 5 minutos

Documentos sin título - Docum... x Slack - Mensaje directo: Andre... x ClusterParcial - Microsoft Azure x MODEL DEPLOYMENT GUIDE... x Iniciar sesión en la cuenta x +

portal.azure.com/#@uao.edu.co/resource/subscriptions/9f0d418d-7d4c-49ff-8e9e-b2b0eaa49e7/resourceGroups/GrupoClusterParcialCC/providers/Microsoft.ContainerService/managedClusters/ClusterParcial/services...

Microsoft Azure Buscar recursos, servicios y documentos (G+/)

Inicio > ClusterParcial

ClusterParcial | Servicios y entradas

Servicio de Kubernetes

Buscar

+ Crear Eliminar Actualizar Mostrar etiquetas Enviar comentarios

Información general
Registro de actividad
Control de acceso (IAM)
Etiquetas
Diagnosticar y solucionar problemas
Microsoft Defender for Cloud

Recursos de Kubernetes
Espacios de nombres
Cargas de trabajo
Servicios y entradas
Almacenamiento
Configuración
Recursos personalizados
Eventos
Comando de ejecución

Configuración
Grupos de nodos
Configuración de clústeres
Redes
Extensiones + aplicaciones
Copia de seguridad
Malla de servicio Istio (versión)

Servicios Entradas

Filtrar por nombre de servicio
Escribe el nombre del servicio ...

Filtrar por espacio de nombres
Todos los espacios de nombres

Agregar filtro de etiqueta

Nombre	Espacio de nombres	Estado	Tipo	Dirección IP de...	Dirección IP ext...	Puertos
kubernetes	default	Aceptar	ClusterIP	10.0.0.1		443/TCP
kube-dns	kube-system	Aceptar	ClusterIP	10.0.0.10		53/UDP 53/TCP
metrics-server	kube-system	Aceptar	ClusterIP	10.0.192.93		443/TCP
calico-typha	calico-system	Aceptar	ClusterIP	10.0.141.218		5473/TCP
calico-kube-controllers...	calico-system	Aceptar	ClusterIP	None		9094/TCP
kubernetes-dashboard	default	Aceptar	LoadBalancer	10.0.50.72	4.246.238.143	80:30862/TCP
myladingpage	default	Pendiente	LoadBalancer	10.0.236.214		80:30847/TCP

Notificaciones

Más eventos en el registro de actividad > Descartar todo

- Se creó la variante de service
La variante de service "myladingpage" se creó correctamente.
hace unos segundos
- Se creó la variante de deployment
La variante de deployment "myladingpage-deployment" se creó correctamente.
hace unos segundos
- El servicio de Kubernetes se inició correctamente
El servicio de Kubernetes "ClusterParcial" se inició correctamente.
hace 42 minutos

https://portal.azure.com/#@uao.edu.co/resource/subscriptions/9f0d418d-7d4c-49ff-8e9e-b2b0eaa49e7/resourceGroups/GrupoClusterParcialCC/providers/Microsoft.ContainerService/managedClusters/ClusterParcial/servicesAndIngresses

30°C
Mayorm. soleado

Buscar

ENG US 1:04 p. m. 15/03/2024

Transcurridos algunos minutos el estado del servicio pasara a READY y ya podrás usar la IP externa para acceder a tu servicio desde el navegador