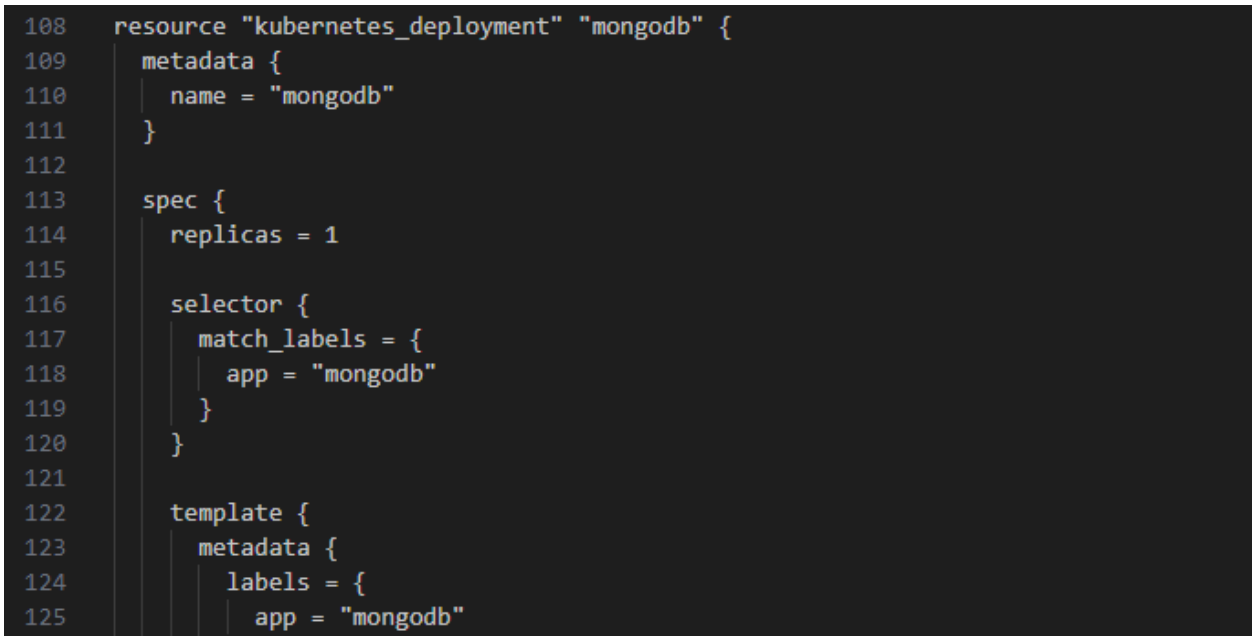
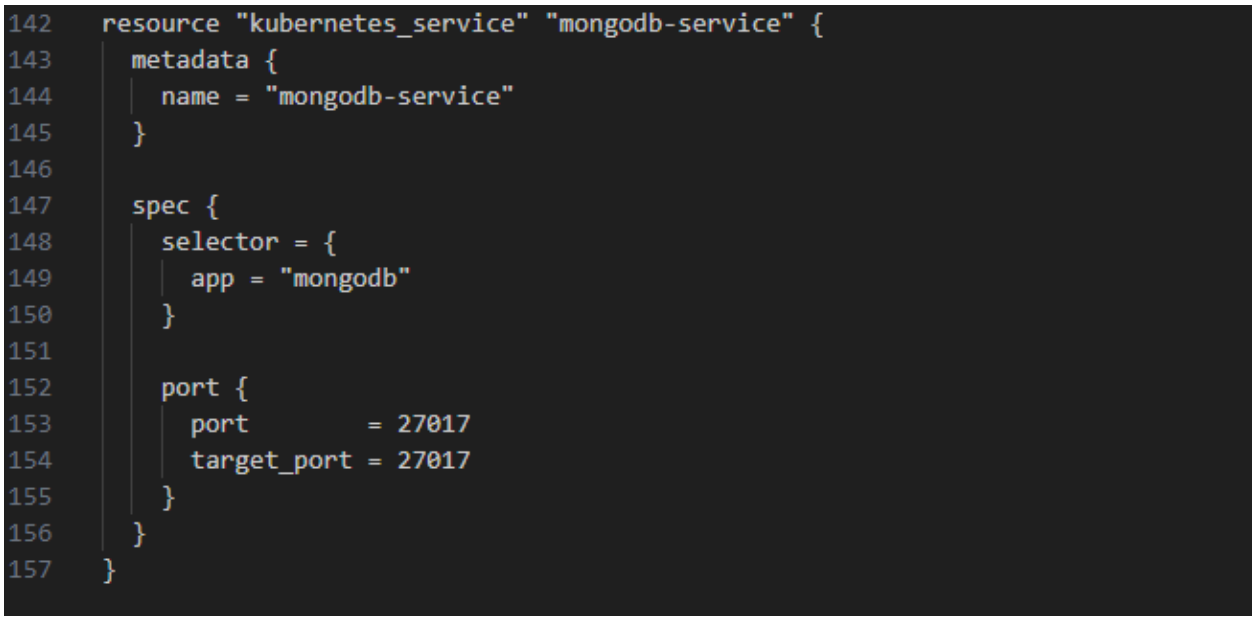
Parcial 3

Diana Alejandra Aragón López 2530019

1. Para implementar la base de datos se escogió mongoDB lo que se realizó fue implementar en el Main.tf su propio deployment y su propio service como se muestra en los siguientes screenshots



```
resource "kubernetes_deployment" "mongodb" {
  metadata {
    name = "mongodb"
  }

  spec {
    replicas = 1

    selector {
      match_labels = {
        app = "mongodb"
      }
    }

    template {
      metadata {
        labels = {
          app = "mongodb"
```



```
resource "kubernetes_service" "mongodb-service" {
  metadata {
    name = "mongodb-service"
  }

  spec {
    selector = {
      app = "mongodb"
    }

    port {
      port        = 27017
      target_port = 27017
    }
  }
}
```

   De esto no se tuvo que realizar un dockerfile ya que esto es una imagen propia de docker.
2. Posterior a esto se realizó el backend el cual es un API de hello world donde se realizó su deployment y su service en el Main.tf como se muestra en las siguientes imágenes.

```
resource "kubernetes_deployment" "apiholadeployment" {
  metadata {
    name = "apihola"
  }

  spec {
    replicas = 1

    selector {
      match_labels = {
        app = "apihola"
      }
    }

    template {
      metadata {
        labels = {
          app = "apihola"
        }
      }

      spec {
```

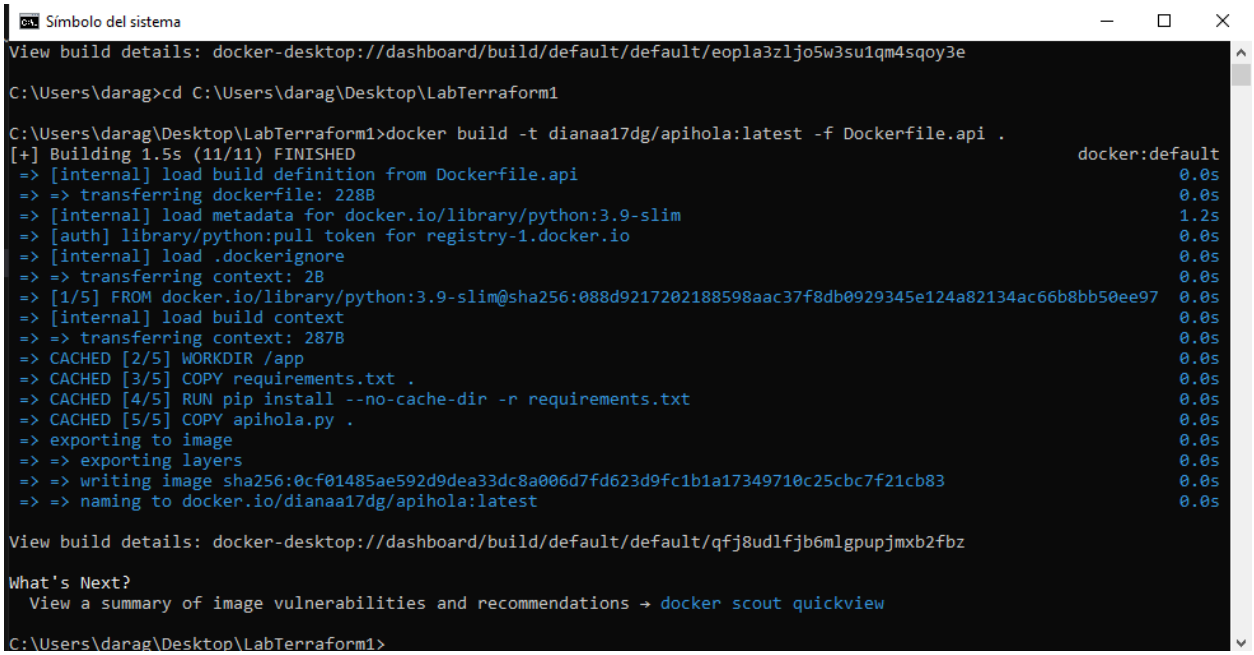
```
resource "kubernetes_service" "apiholaservice" {
  metadata {
    name = "apiholaservice"
  }

  spec {
    selector = {
      app = "apihola"
    }

    port {
      port        = 80
      target_port = 8080
    }
  }
}
resource "kubernetes_deployment" "webdeployment" {
  metadata {
    name = "web"
  }
```

Posterior a esto se creó su propio dockerfile donde se describe su archivo de requerimientos y su puerto.

Dockerfile.api > ...

```
1   FROM python:3.9-slim
2
3   WORKDIR /app
4
5   COPY requirements.txt .
6
7   RUN pip install --no-cache-dir -r requirements.txt
8
9   COPY apihola.py .
10
11  EXPOSE 8080
12
13  CMD ["python", "apihola.py"]
```

3. Posterior a esto en el CMD se realizó el comando docker login para luego hacer build del api de hello world

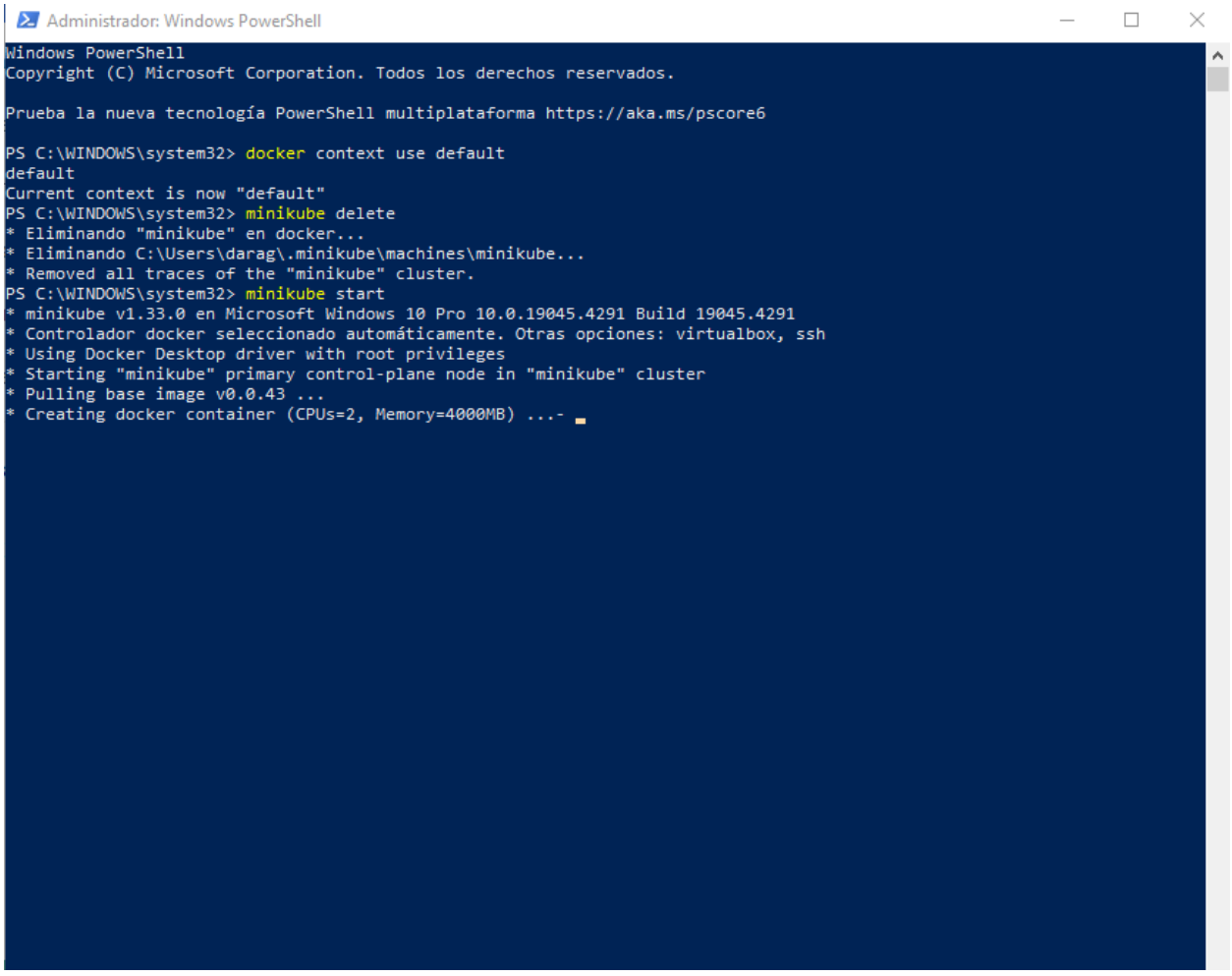


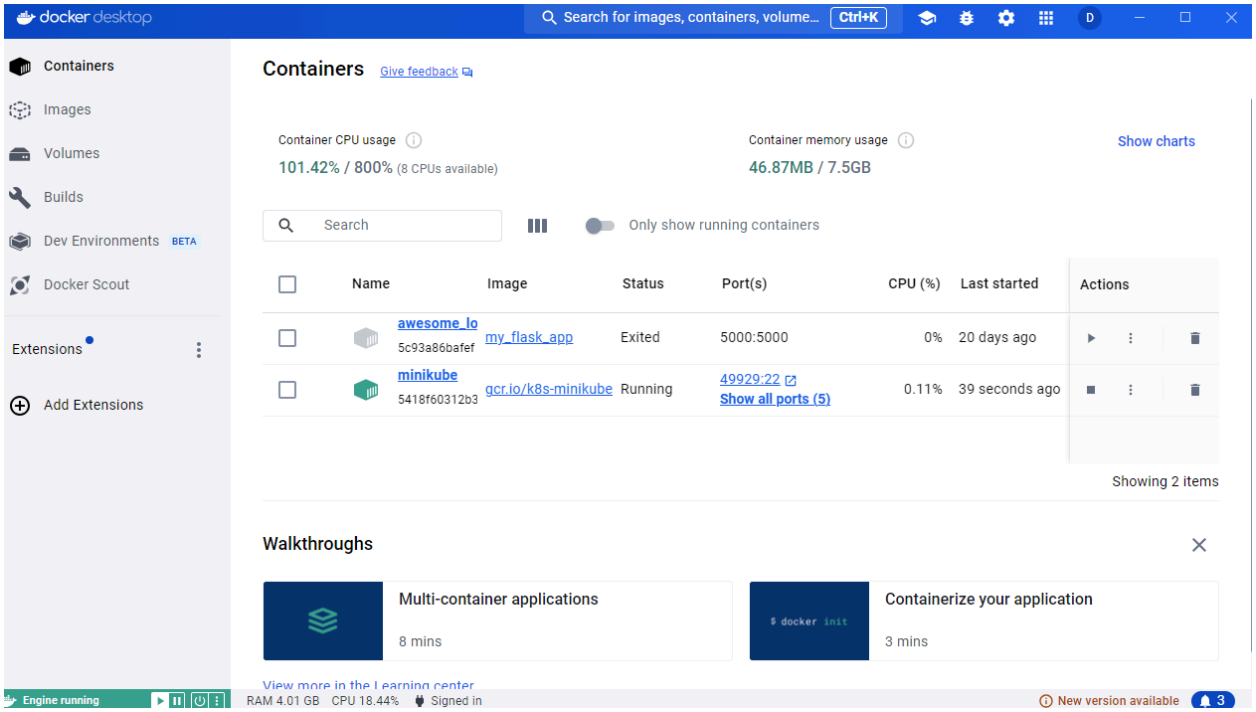
Esto fue realizado con el siguiente comando docker build -t dianaa17dg/apihola:latest -f Dockerfile.api .

Luego se realizó push con el siguiente comando docker push dianaa17dg/nombre:latest

```
C:\Users\darag\Desktop\LabTerraform1>docker push dianaa17dg/apihola:latest
The push refers to repository [docker.io/dianaa17dg/apihola]
7403194fe1e2: Layer already exists
6590fd0bc415: Layer already exists
cdb865c3624e: Layer already exists
cd1b5e9a88d1: Layer already exists
ae96698df02c: Layer already exists
e555c0055a9b: Layer already exists
205262265e50: Layer already exists
146826fa3ca0: Layer already exists
5d4427064ecc: Layer already exists
latest: digest: sha256:109d2948e1db58b696a7aed9b0d464fccbb4871b07b3e7cac18f54d5917b2f45 size: 2201

C:\Users\darag\Desktop\LabTerraform1>
```

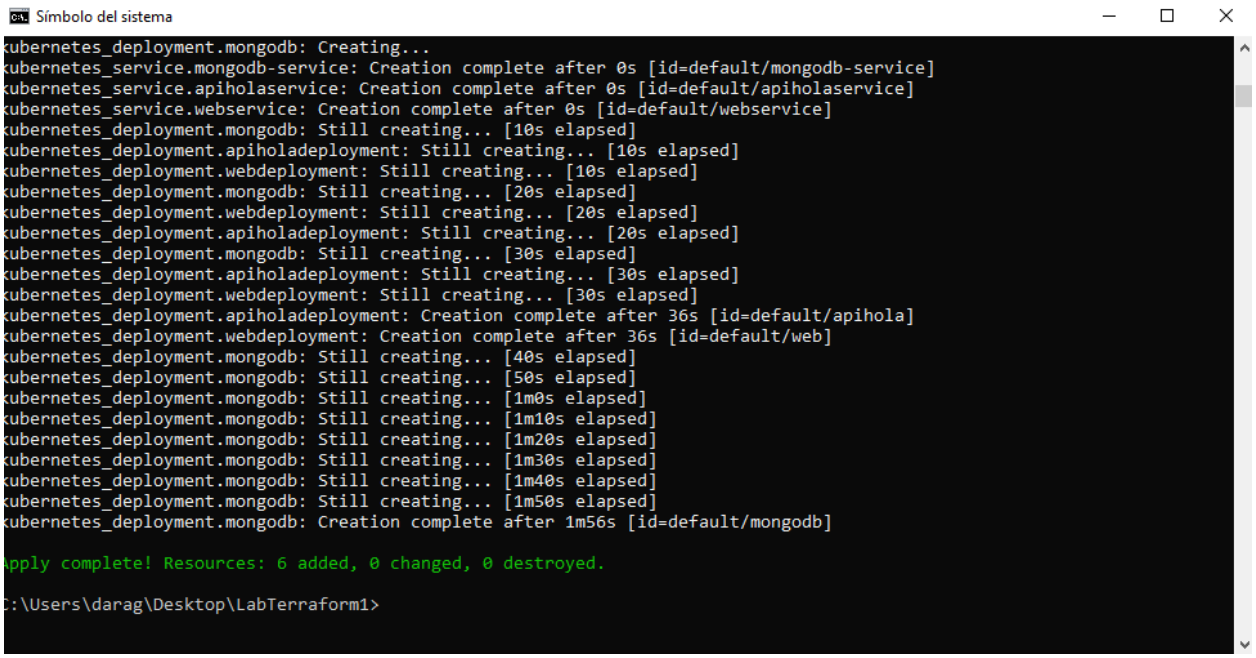
4. Se levantó minikube para poder implementar kubernetes y poder ejecutar el sistema de manera local.





5. Por último se realizó un terraform init y un terraform apply

```
C:\Users\darag\Desktop\LabTerraform1>terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/kubernetes from the dependency lock file
- Using previously-installed hashicorp/kubernetes v2.30.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

C:\Users\darag\Desktop\LabTerraform1>
```



6. Con el comando minikube service apiholaservice --url se pudo verificar la api de hello world estaba funcionando luego de aplicar el terraform.