

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

```
108 resource "kubernetes_deployment" "mongodb" {
109   metadata {
110     name = "mongodb"
111   }
112
113   spec {
114     replicas = 1
115
116     selector {
117       match_labels = {
118         app = "mongodb"
119       }
120     }
121
122     template {
123       metadata {
124         labels = {
125           app = "mongodb"
```

```
142 resource "kubernetes_service" "mongodb-service" {
143   metadata {
144     name = "mongodb-service"
145   }
146
147   spec {
148     selector = {
149       app = "mongodb"
150     }
151
152     port {
153       port          = 27017
154       target_port = 27017
155     }
156   }
157 }
```

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

```
Simbolo del sistema
View build details: docker-desktop:///dashboard/build/default/default/eopla3zljo5w3su1qm4sqoy3e
C:\Users\darag>cd C:\Users\darag\Desktop\LabTerraform1
C:\Users\darag\Desktop\LabTerraform1>docker build -t dianaa17dg/apihola:latest -f Dockerfile.api .
[+] Building 1.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile.api 0.0s
=> => transferring dockerfile: 228B 0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim 1.2s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:088d9217202188598aac37f8db0929345e124a82134ac66b8bb50ee97 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 287B 0.0s
=> CACHED [2/5] WORKDIR /app 0.0s
=> CACHED [3/5] COPY requirements.txt . 0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> CACHED [5/5] COPY apihola.py . 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:0cf01485ae592d9dea33dc8a006d7fd623d9fc1b1a17349710c25cbc7f21cb83 0.0s
=> => naming to docker.io/dianaa17dg/apihola:latest 0.0s
View build details: docker-desktop:///dashboard/build/default/default/qfj8udlfjb6mlgpupjmx2fbz
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
C:\Users\darag\Desktop\LabTerraform1>
```

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`

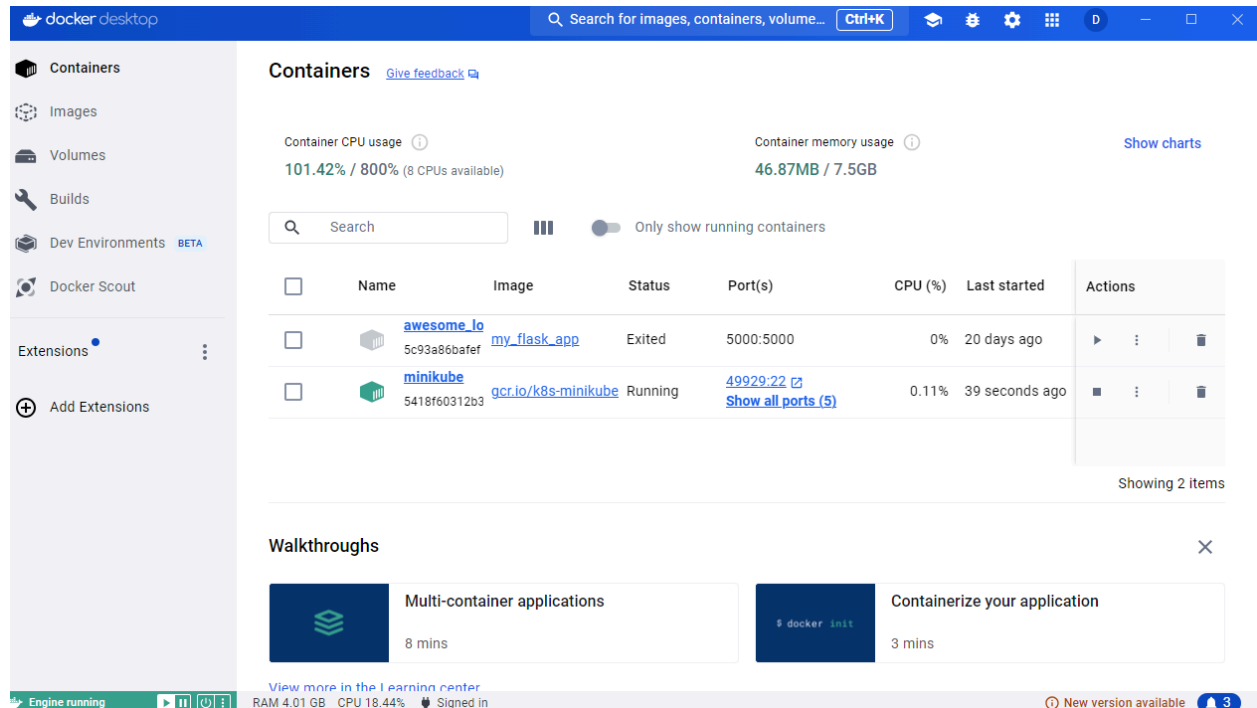
```
C:\Users\darag\Desktop\LabTerraform1>docker push diana17dg/apihola:latest
The push refers to repository [docker.io/diana17dg/apihola]
7403194fe1e2: Layer already exists
6590fd0bc415: Layer already exists
cdb865c3624e: Layer already exists
cd1b5e9a88d1: Layer already exists
ae96698df02c: Layer already 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.

```
Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\WINDOWS\system32> docker context use default
default
Current context is now "default"
PS C:\WINDOWS\system32> minikube delete
* Eliminando "minikube" en docker...
* Eliminando C:\Users\darag\minikube\machines\minikube...
* Removed all traces of the "minikube" cluster.
PS C:\WINDOWS\system32> minikube start
* minikube v1.33.0 en Microsoft Windows 10 Pro 10.0.19045.4291 Build 19045.4291
* Controlador docker seleccionado automáticamente. Otras opciones: virtualbox, ssh
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.43 ...
* Creating docker container (CPUs=2, Memory=4000MB) ...-
```



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>
```

```
Símbolo del sistema
kubernetes_deployment.mongodb: Creating...
kubernetes_service.mongodb-service: Creation complete after 0s [id=default/mongodb-service]
kubernetes_service.apiholaservice: Creation complete after 0s [id=default/apiholaservice]
kubernetes_service.webservice: Creation complete after 0s [id=default/web-service]
kubernetes_deployment.mongodb: Still creating... [10s elapsed]
kubernetes_deployment.apiholadeployment: Still creating... [10s elapsed]
kubernetes_deployment.webdeployment: Still creating... [10s elapsed]
kubernetes_deployment.mongodb: Still creating... [20s elapsed]
kubernetes_deployment.apiholadeployment: Still creating... [20s elapsed]
kubernetes_deployment.webdeployment: Still creating... [20s elapsed]
kubernetes_deployment.mongodb: Still creating... [30s elapsed]
kubernetes_deployment.apiholadeployment: Still creating... [30s elapsed]
kubernetes_deployment.webdeployment: Still creating... [30s elapsed]
kubernetes_deployment.apiholadeployment: Creation complete after 36s [id=default/apihola]
kubernetes_deployment.webdeployment: Creation complete after 36s [id=default/web]
kubernetes_deployment.mongodb: Still creating... [40s elapsed]
kubernetes_deployment.mongodb: Still creating... [50s elapsed]
kubernetes_deployment.mongodb: Still creating... [1m0s elapsed]
kubernetes_deployment.mongodb: Still creating... [1m10s elapsed]
kubernetes_deployment.mongodb: Still creating... [1m20s elapsed]
kubernetes_deployment.mongodb: Still creating... [1m30s elapsed]
kubernetes_deployment.mongodb: Still creating... [1m40s elapsed]
kubernetes_deployment.mongodb: Still creating... [1m50s elapsed]
kubernetes_deployment.mongodb: Creation complete after 1m56s [id=default/mongodb]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

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.

```
PS C:\WINDOWS\system32> minikube service apiholaservice --url
* service default/apiholaservice has no node port
! Services [default/apiholaservice] have type "ClusterIP" not meant to be exposed, however for local development minikube allows you to access this !
http://127.0.0.1:50643
! Porque estás usando controlador Docker en windows, la terminal debe abrirse para ejecutarlo.
```

