Oscar Josue Rivera Menendez 1203819

Primero se realizaron los dos archivos necesarios de pyhton (Mi api en Python que simplemente devuelve hola mundo prueba:

```python
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/')
def hello_world():
    return '¡Hola, Mundo PRUEBA!'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=8080)
```

Después se creo el contendor de Docker para poder subir a Docker hub(con dockerfile se creo el contendor y se levanto con el comando de Docker build) simeplemente el dockerfile copia el api anterior y instala los paquetes necesarios para correrlo en el contenedor:

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8080

CMD ["python", "app.py"]
```

En donde era necesario tener el siguiente parquetes en el archivo de configuración del contenedor:

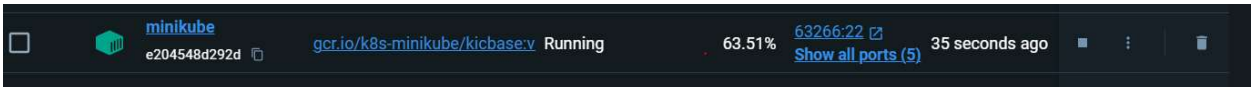
requirements.txt

```
1   Flask==3.0.1
2   Werkzeug==3.0.1
```

Se levanto el contedor y después se realizo un push para tenerlo listo en Docker hub para utilizarlo con el terraform:

```
C:\Users\oscar\OneDrive\Desktop\Parcial3_virutal>docker build -t oscarrivera1/api-image:latest -f Dockerfile.api .
[+] Building 19.3s (11/11) FINISHED                                                                          docker:def
 => [internal] load build definition from Dockerfile.api
 => => transferring dockerfile: 217B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [auth] library/python:pull token for registry-1.docker.io
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:088d9217202188598aac37f8db0929345e124a82134ac66b8bb50ee9  1
 => => resolve docker.io/library/python:3.9-slim@sha256:088d9217202188598aac37f8db0929345e124a82134ac66b8bb50ee97
 => => sha256:09f376ebb190216b0459f470e71bec7b5dfa611d66bf008492b40dcc5f1d8eae 29.15MB / 29.15MB                1
 => => sha256:276709cbedc1f168290ee408fca2af2aacfeb4f922ddca125e9e8047f9841479 3.51MB / 3.51MB
 => => sha256:4e7363ac3b6fb61a9310bbb00e385beaa54c712a9633c01de34cc7d8b0823dba 11.89MB / 11.89MB
 => => sha256:088d9217202188598aac37f8db0929345e124a82134ac66b8bb50ee9750b045b 1.86kB / 1.86kB
 => => sha256:b92e6f45b58d9cafacc38563e946f8d249d850db862cbbd8befcf7f49eef8209 1.37kB / 1.37kB
 => => sha256:4602238ffbdcf66f436adfb46e31c9521ab4a9960b51b1a051004fa5a70f3f42 6.90kB / 6.90kB
 => => sha256:1f1e6fb6a4a52a77049d55697db79164d7d0e5a78ae115c657699f4471398fc0 244B / 244B
 => => sha256:bf8f57a642c477da4e61c92dc0c0fd036a8d7e3d3951df39b88c3dd73bf3d5af 3.13MB / 3.13MB
 => => extracting sha256:09f376ebb190216b0459f470e71bec7b5dfa611d66bf008492b40dcc5f1d8eae
 => => extracting sha256:276709cbedc1f168290ee408fca2af2aacfeb4f922ddca125e9e8047f9841479
 => => extracting sha256:4e7363ac3b6fb61a9310bbb00e385beaa54c712a9633c01de34cc7d8b0823dba
 => => extracting sha256:1f1e6fb6a4a52a77049d55697db79164d7d0e5a78ae115c657699f4471398fc0
 => => extracting sha256:bf8f57a642c477da4e61c92dc0c0fd036a8d7e3d3951df39b88c3dd73bf3d5af
 => [internal] load build context
 => => transferring context: 46.33kB
 => [2/5] WORKDIR /app
 => [3/5] COPY requirements.txt .
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => [5/5] COPY . .
```

Al terminar eso se inicio el minikube para levantar el cluster de kubernetes



Después se comenzó a configurar el archivo de main.tf para setear las configuaraciones necesarios para levantar todo los deployments necesarios. Uno para el spa, otro para el hello-wrodl api y también la configuración del base de datos no sql

Primer configuración(ya que lo estoy corriendo local, el minikube será el provider:

```
provider "kubernetes" {
  config_path    = "~/.kube/config"
  config_context = "minikube"
}
```

Segunda configuración para el deployment de hello-world-api, en donde se utilizo el puerto 8080 y la imagen del api que realice:

```
spec {
  container {
    image = "oscarrivera1/mi_api_flask:latest"
    name  = "api"
    port {
      container_port = 8080
    }
  }
```

Después se realizo la configuraciones del helloworld spa con el nignx , en donde se utilizo otro puerto, el 3037, y la imagen del nginx-spa :

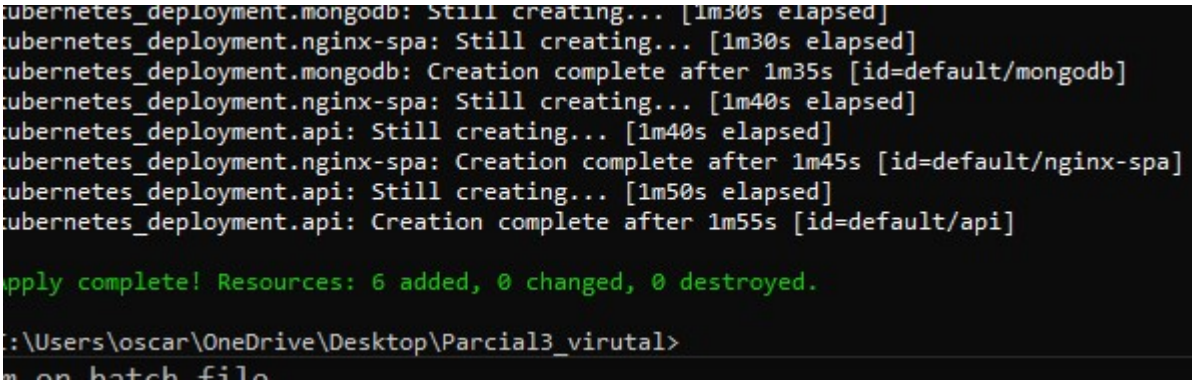
```
spec {
  container {
    image = "oscarrivera1/nginx-spa-image:latest"
    name  = "nginx-spa"
    port {
      container_port = 3037
    }
  }
}
```

Y por ultimo la configuración el base de datos no relacional, en este caso se utiliza mongodb y la imagen default que tiene Docker hub y el puerto default que utiliza mongodb:

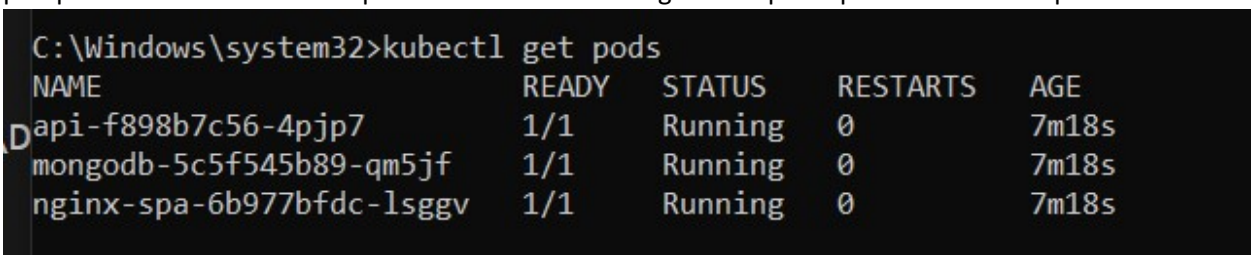
```
template {
  metadata {
    labels = {
      app = "mongodb"
    }
  }

  spec {
    container {
      name  = "mongodb"
      image = "mongo:latest"
      port {
        container_port = 27017
      }
```

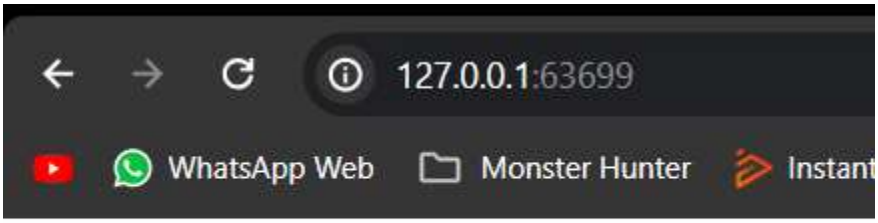
Al terminar con las configuraciones, se levanto terraform con los comandos terraform init y terraform apply. Y debido a que se tiene un servicio y un deployment por cada tecnología, se crearon al final 6 recursos



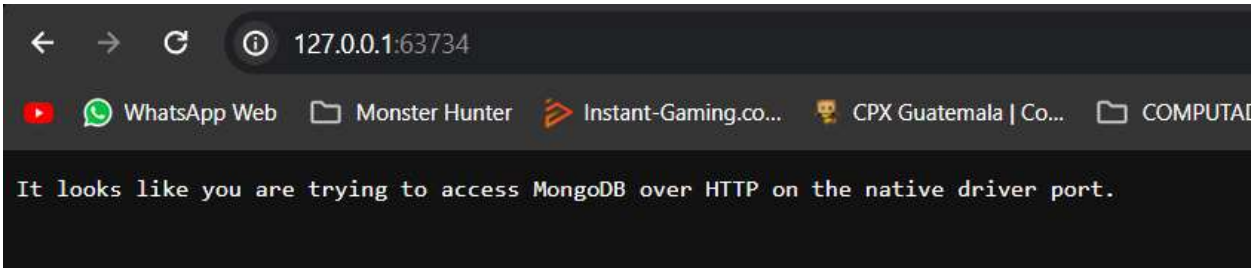
Y con el comando de get pods, se puede verificar que estan corriendo y se puede ver que solo hay un pod por cada servicio debido que en el archivo de configuración puse que solo fuera 1 replica:



Y se utilizo localhost con el puerto dado con minikube, el mismo caso seria para los otros pero debido a que el mongodb no se puede abrir mediando el explorador sino requiere de un CLI, tira el siguiente error:



¡Hola, Mundo PRUEBA!



Pero si esta corriendo y levantado