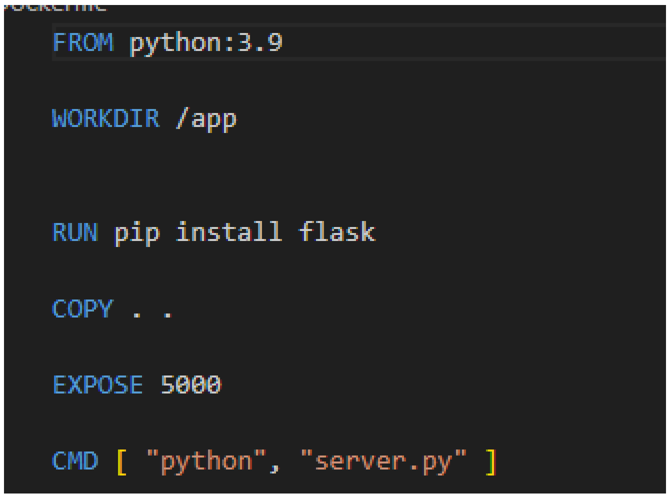
Daniel Elias 1247020

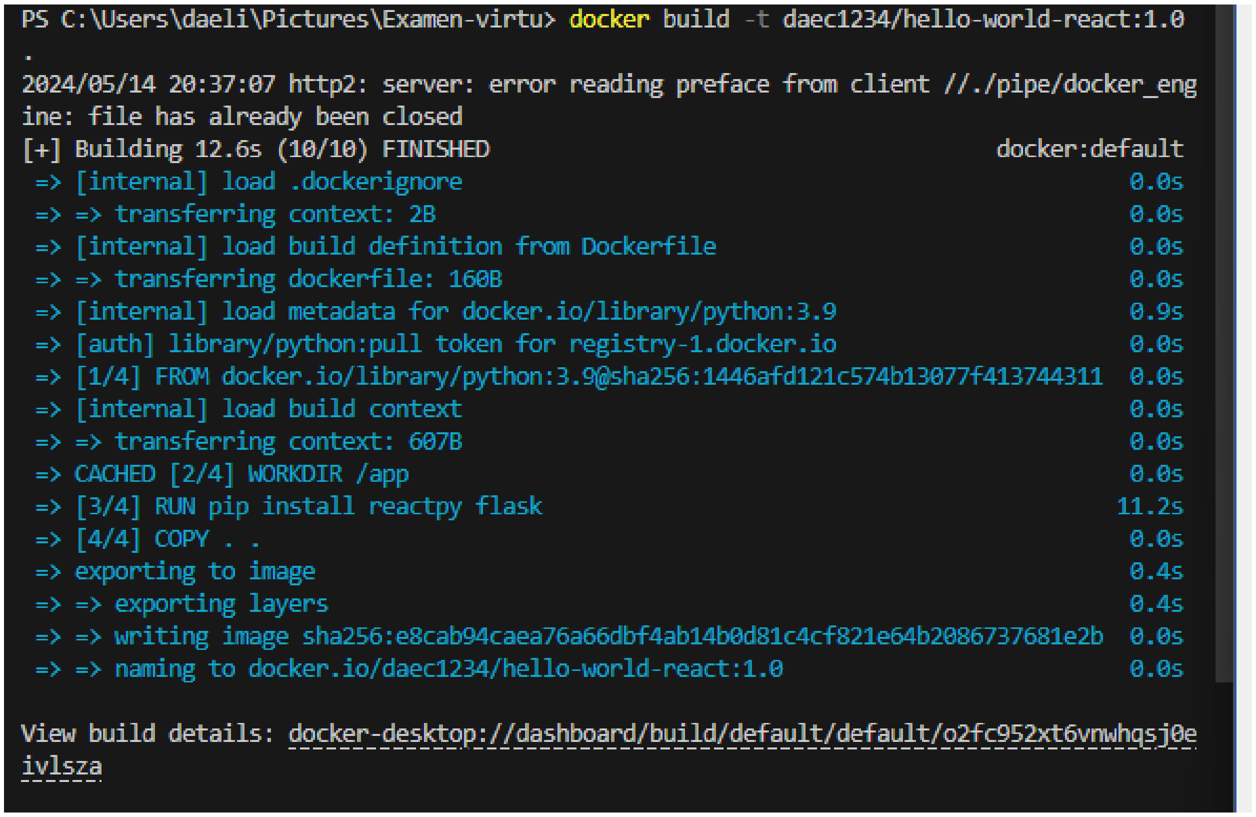
Examen 3

Para empezar a crear esto hacemos un dockerfile donde va a estar el helloWorld



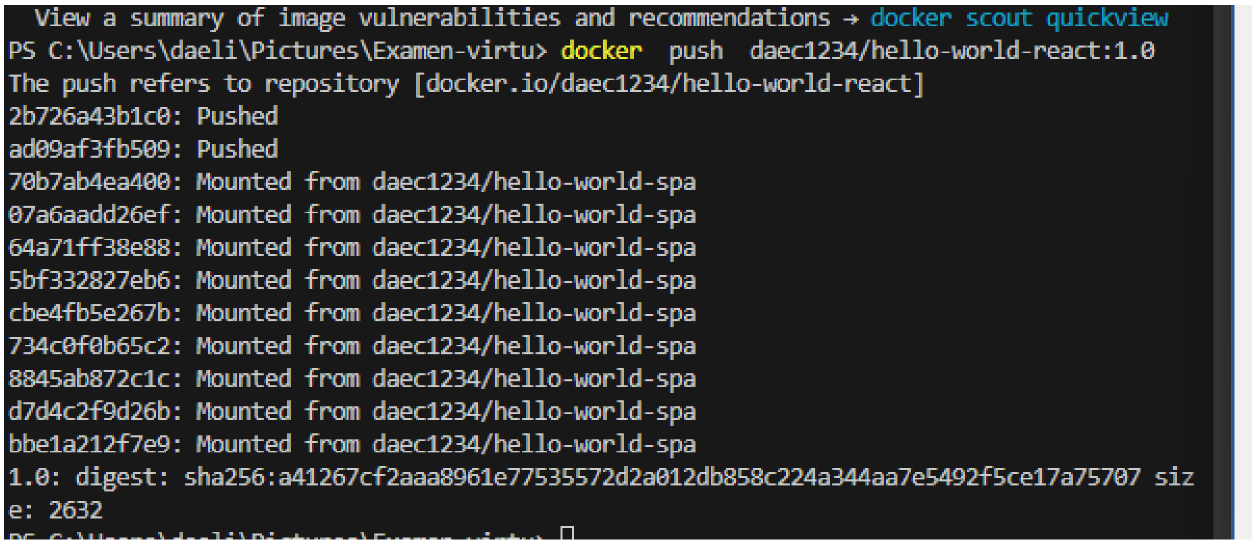
```
FROM python:3.9

WORKDIR /app


RUN pip install flask

COPY . .

EXPOSE 5000

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

Creamos la imagen con el siguiente comando

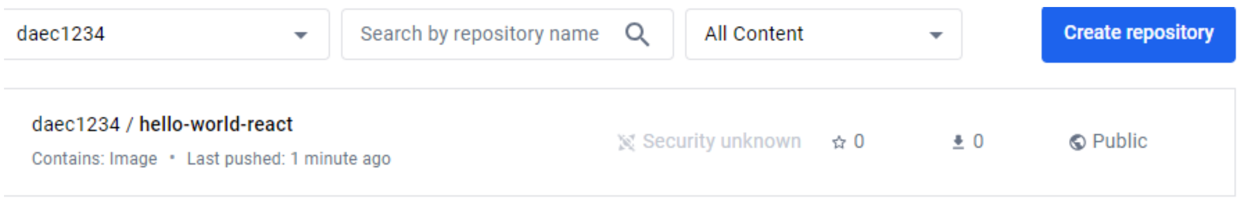


```
PS C:\Users\daeli\Pictures\Examen-virtu> docker build -t daec1234/hello-world-react:1.0
.
2024/05/14 20:37:07 http2: server: error reading preface from client //./pipe/docker_eng
ine: file has already been closed
[+] Building 12.6s (10/10) FINISHED                                       docker:default
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 160B                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.9                       0.9s
 => [auth] library/python:pull token for registry-1.docker.io                       0.0s
 => [1/4] FROM docker.io/library/python:3.9@sha256:1446afd121c574b13077f413744311  0.0s
 => [internal] load build context                                                   0.0s
 => => transferring context: 607B                                                   0.0s
 => CACHED [2/4] WORKDIR /app                                                       0.0s
 => [3/4] RUN pip install reactpy flask                                            11.2s
 => [4/4] COPY . .                                                                  0.0s
 => exporting to image                                                              0.4s
 => => exporting layers                                                             0.4s
 => => writing image sha256:e8cab94caea76a66dbf4ab14b0d81c4cf821e64b2086737681e2b  0.0s
 => => naming to docker.io/daec1234/hello-world-react:1.0                           0.0s

View build details: docker-desktop://dashboard/build/default/default/o2fc952xt6vnwhqsj0e
ivlsza
```

Luego subimos la imagen a Docker hub

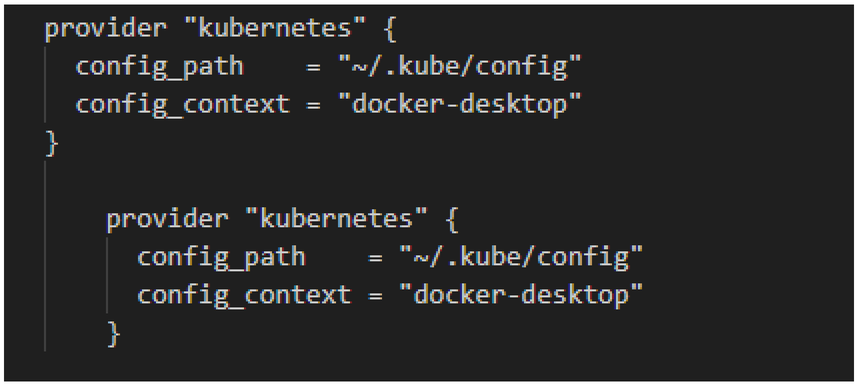


```
 View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\daeli\Pictures\Examen-virtu> docker  push  daec1234/hello-world-react:1.0
The push refers to repository [docker.io/daec1234/hello-world-react]
2b726a43b1c0: Pushed
ad09af3fb509: Pushed
70b7ab4ea400: Mounted from daec1234/hello-world-spa
07a6aadd26ef: Mounted from daec1234/hello-world-spa
64a71ff38e88: Mounted from daec1234/hello-world-spa
5bf332827eb6: Mounted from daec1234/hello-world-spa
cbe4fb5e267b: Mounted from daec1234/hello-world-spa
734c0f0b65c2: Mounted from daec1234/hello-world-spa
8845ab872c1c: Mounted from daec1234/hello-world-spa
d7d4c2f9d26b: Mounted from daec1234/hello-world-spa
bbe1a212f7e9: Mounted from daec1234/hello-world-spa
1.0: digest: sha256:a41267cf2aaa8961e77535572d2a012db858c224a344aa7e5492f5ce17a75707 siz
e: 2632
```



daec1234 / hello-world-react
Contains: Image • Last pushed: 1 minute ago
Security unknown ☆ 0 ⬇ 0 Public

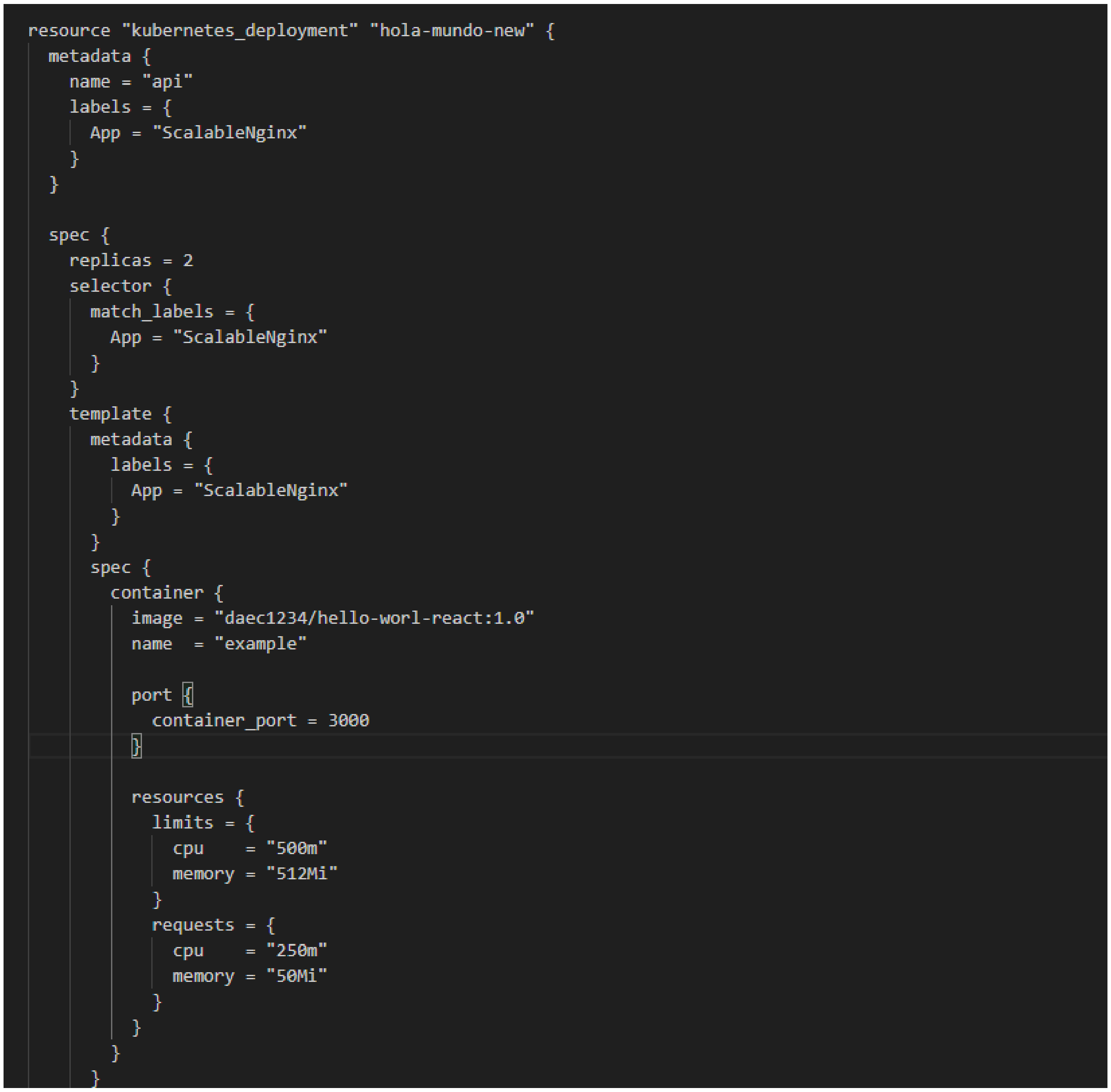
Ahora crearemos el terraform es con el nombre main.tf



```
provider "kubernetes" {
  config_path    = "~/.kube/config"
  config_context = "docker-desktop"
}

    provider "kubernetes" {
      config_path    = "~/.kube/config"
      config_context = "docker-desktop"
    }
```

El provider que usaremos será el kubernetes que trae Docker por default

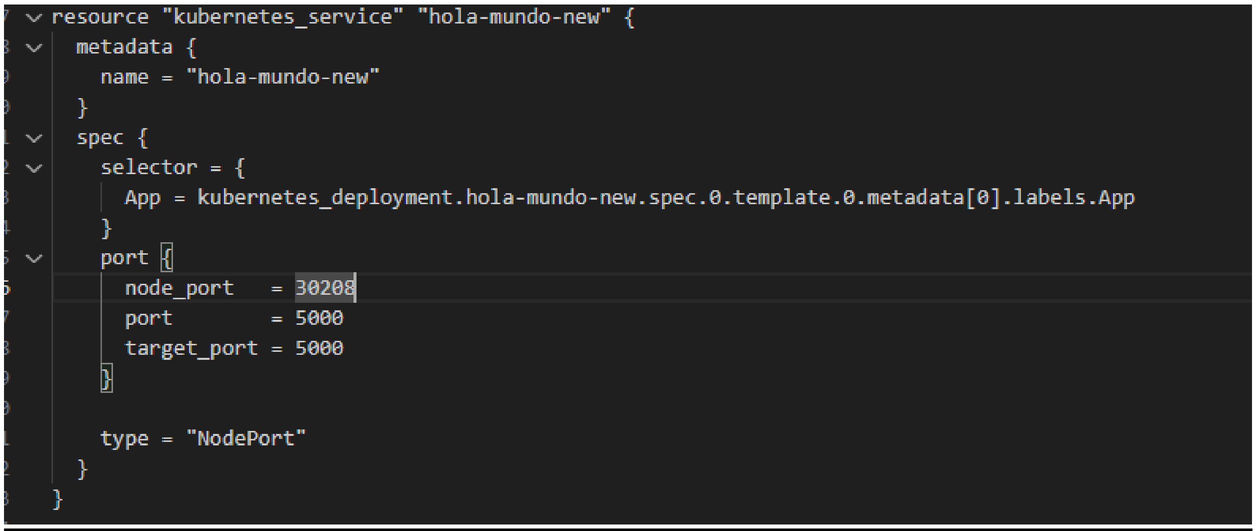
Ahora creare el servicio de la primer imagen que contiene hello-world con react



```
resource "kubernetes_deployment" "hola-mundo-new" {
  metadata {
    name = "api"
    labels = {
      App = "ScalableNginx"
    }
  }

  spec {
    replicas = 2
    selector {
      match_labels = {
        App = "ScalableNginx"
      }
    }
    template {
      metadata {
        labels = {
          App = "ScalableNginx"
        }
      }
      spec {
        container {
          image = "daec1234/hello-worl-react:1.0"
          name  = "example"

          port {
            container_port = 3000
          }

          resources {
            limits = {
              cpu    = "500m"
              memory = "512Mi"
            }
            requests = {
              cpu    = "250m"
              memory = "50Mi"
            }
          }
        }
      }
```

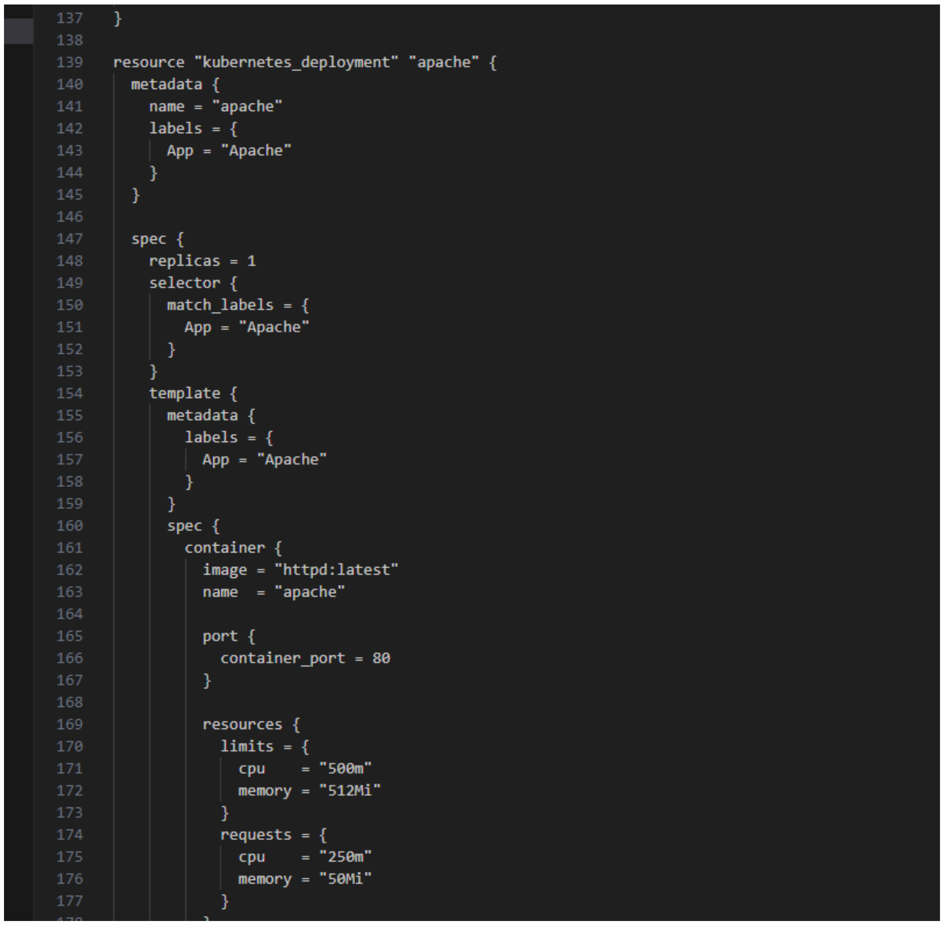
Aquí lo creamos y se le asigna los recursos y se declara la imagen que se va a instalar que en este caso es daec1234/hello-world-react:1.0

Luego le asignamos el puerto donde se podrá acceder a la imagen que es el 30208



```
resource "kubernetes_service" "hola-mundo-new" {
  metadata {
    name = "hola-mundo-new"
  }
  spec {
    selector = {
      App = kubernetes_deployment.hola-mundo-new.spec.0.template.0.metadata[0].labels.App
    }
    port {
      node_port   = 30208
      port        = 5000
      target_port = 5000
    }

    type = "NodePort"
  }
}
```

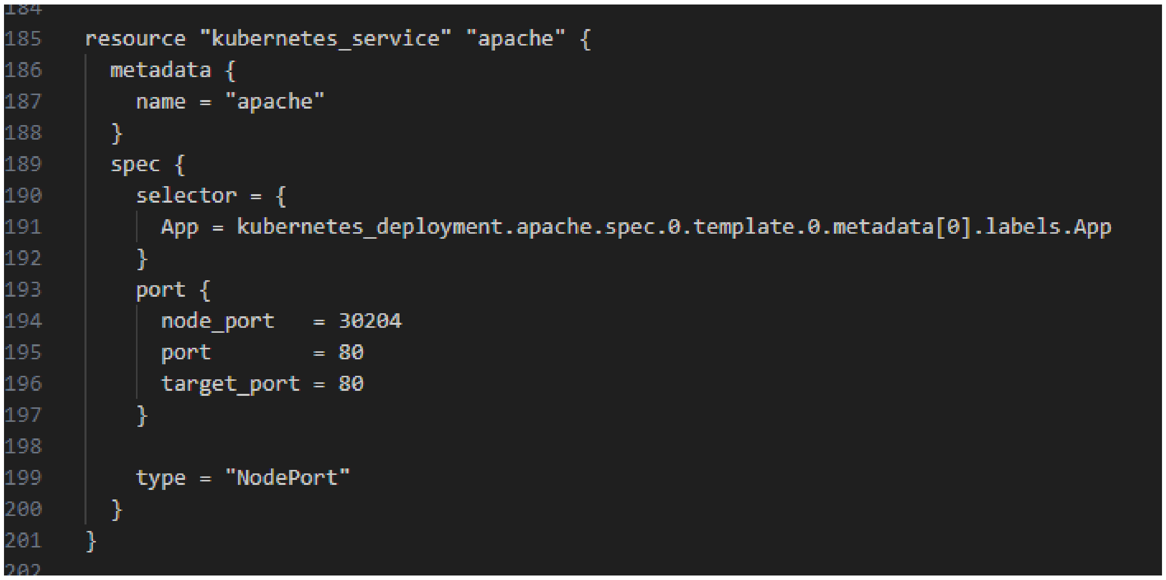
Ahora para el backend vamos a utilizar apache, en este utilizaremos una imagen que ya esta publica en Docker hub



```
}

resource "kubernetes_deployment" "apache" {
  metadata {
    name = "apache"
    labels = {
      App = "Apache"
    }
  }

  spec {
    replicas = 1
    selector {
      match_labels = {
        App = "Apache"
      }
    }
    template {
      metadata {
        labels = {
          App = "Apache"
        }
      }
      spec {
        container {
          image = "httpd:latest"
          name  = "apache"

          port {
            container_port = 80
          }

          resources {
            limits = {
              cpu    = "500m"
              memory = "512Mi"
            }
            requests = {
              cpu    = "250m"
              memory = "50Mi"
            }
```

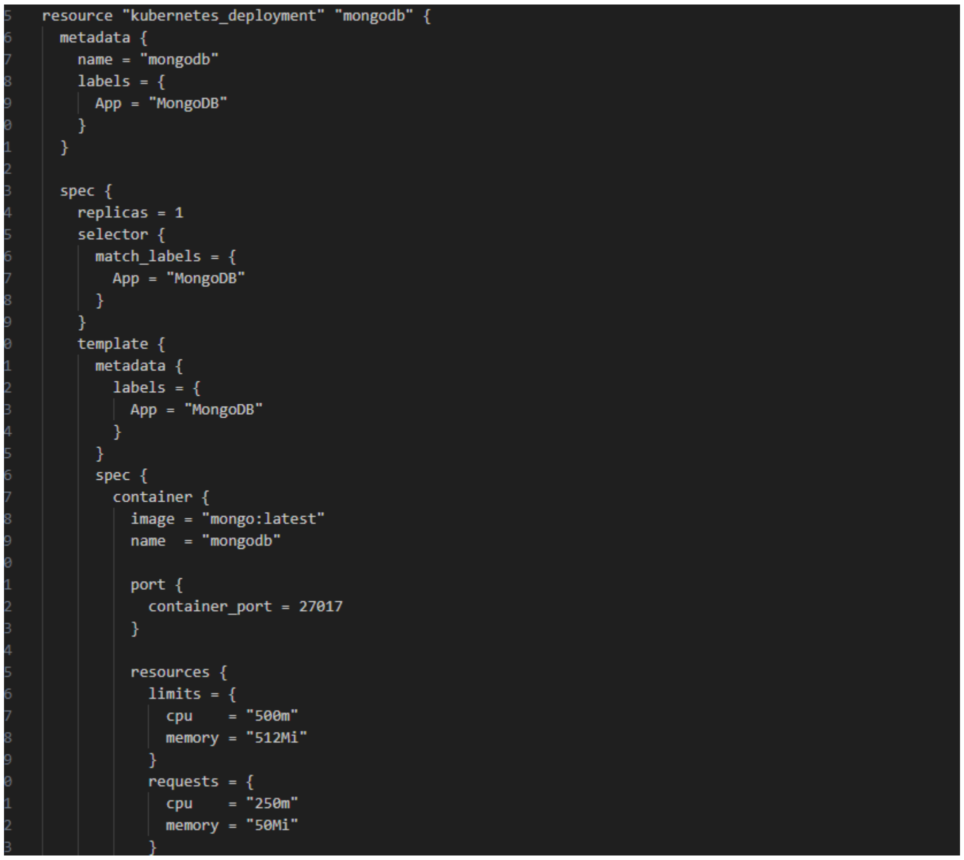
Aquí lo creamos y se le asigna los recursos y también la imagen

Le asignamos un puerto para poder acceder en este caso seria el 30204



```
resource "kubernetes_service" "apache" {
  metadata {
    name = "apache"
  }
  spec {
    selector = {
      App = kubernetes_deployment.apache.spec.0.template.0.metadata[0].labels.App
    }
    port {
      node_port   = 30204
      port        = 80
      target_port = 80
    }

    type = "NodePort"
  }
}
```

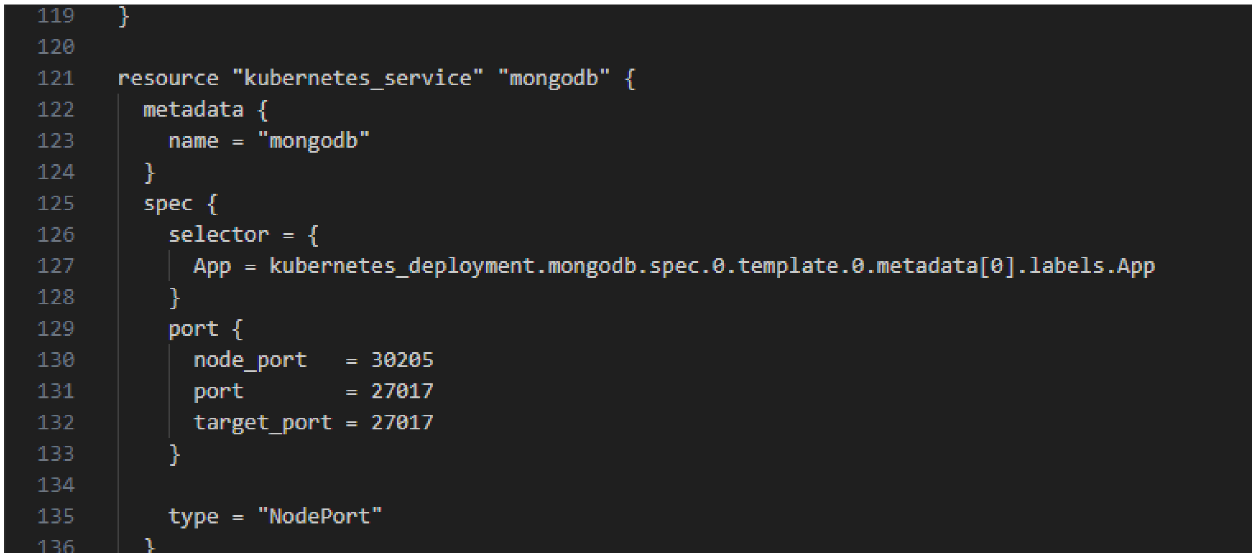
Para la base de datos vamos a utilizar mongoDb y una imagen publica que ya esta en Docker hub



```
resource "kubernetes_deployment" "mongodb" {
  metadata {
    name = "mongodb"
    labels = {
      App = "MongoDB"
    }
  }

  spec {
    replicas = 1
    selector {
      match_labels = {
        App = "MongoDB"
      }
    }
    template {
      metadata {
        labels = {
          App = "MongoDB"
        }
      }
      spec {
        container {
          image = "mongo:latest"
          name  = "mongodb"

          port {
            container_port = 27017
          }

          resources {
            limits = {
              cpu    = "500m"
              memory = "512Mi"
            }
            requests = {
              cpu    = "250m"
              memory = "50Mi"
            }
```

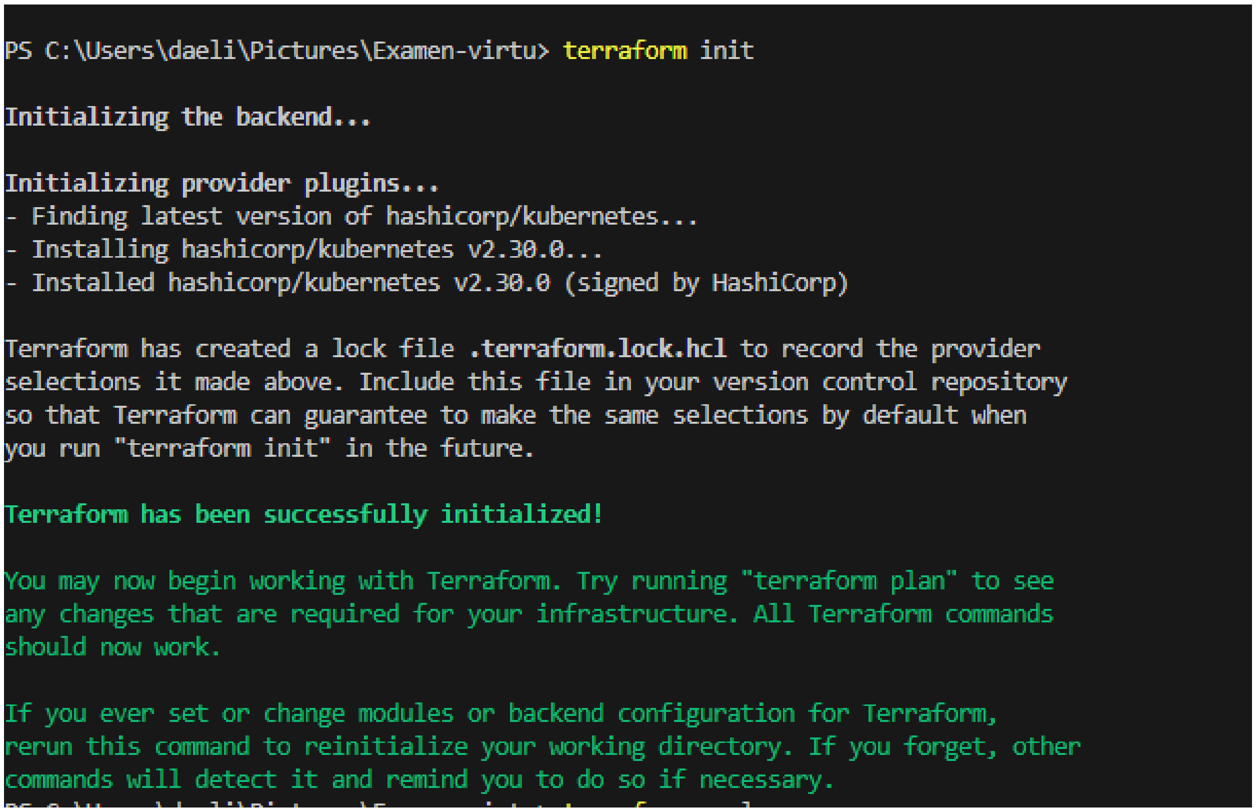
Aquí asingamos la imagen y los recursos que podrá utilizar

Asignamos el puerto al que se podrá acceder al servicio en este caso es el 30205



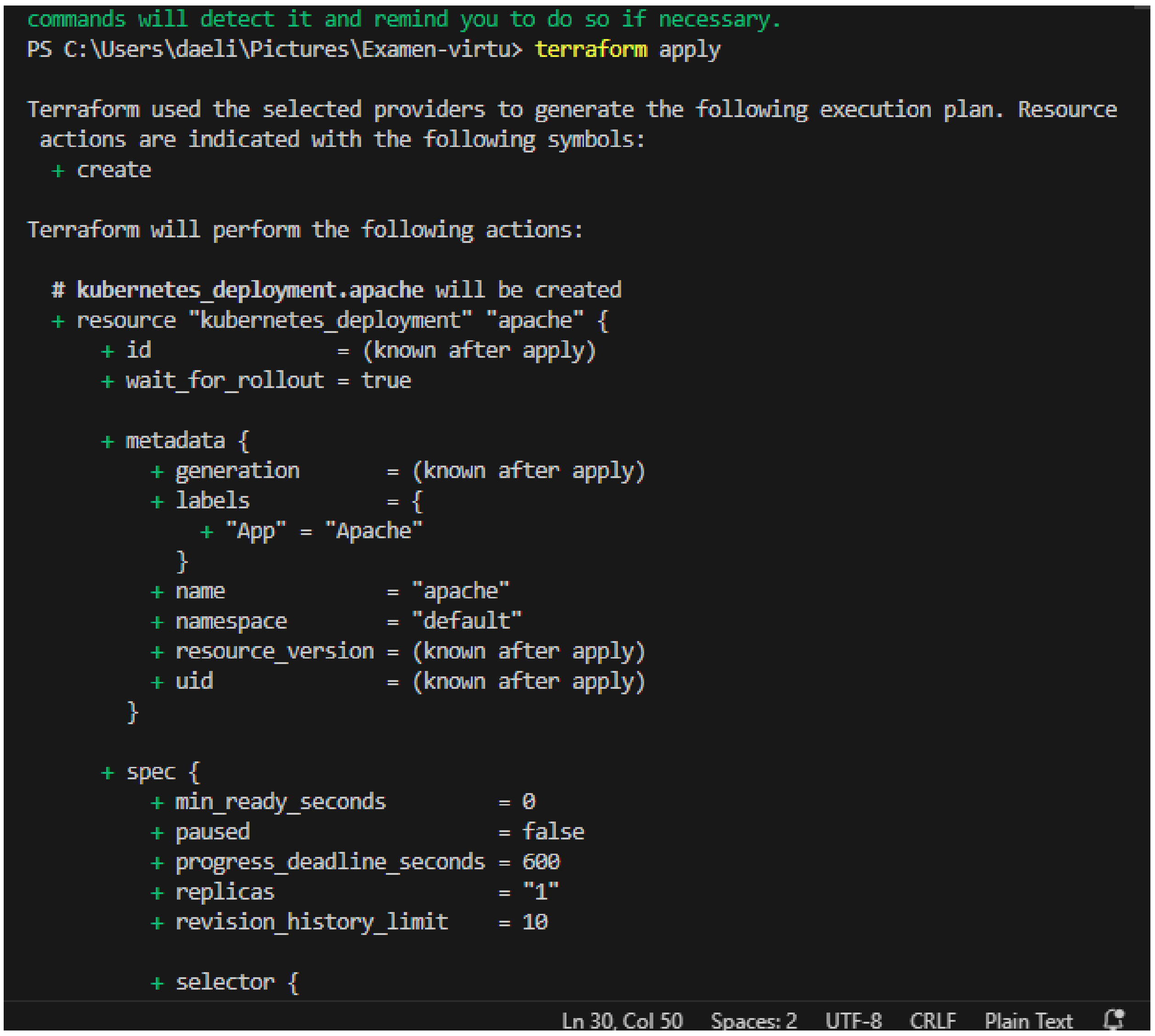
```
}

resource "kubernetes_service" "mongodb" {
  metadata {
    name = "mongodb"
  }
  spec {
    selector = {
      App = kubernetes_deployment.mongodb.spec.0.template.0.metadata[0].labels.App
    }
    port {
      node_port   = 30205
      port        = 27017
      target_port = 27017
    }

    type = "NodePort"
  }
```

Ahora que ya tenemos el archivo terraform terminado aplicamos el siguiente comando



```
PS C:\Users\daeli\Pictures\Examen-virtu> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/kubernetes...
- Installing hashicorp/kubernetes v2.30.0...
- Installed hashicorp/kubernetes v2.30.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

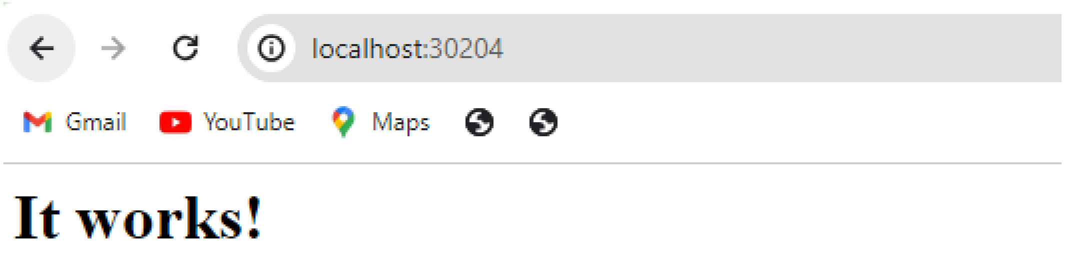
Luego hacemos el siguiente comando



```
commands will detect it and remind you to do so if necessary.
PS C:\Users\daeli\Pictures\Examen-virtu> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource
 actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # kubernetes_deployment.apache will be created
  + resource "kubernetes_deployment" "apache" {
      + id               = (known after apply)
      + wait_for_rollout = true

      + metadata {
          + generation       = (known after apply)
          + labels           = {
              + "App" = "Apache"
            }
          + name             = "apache"
          + namespace        = "default"
          + resource_version = (known after apply)
          + uid              = (known after apply)
        }

      + spec {
          + min_ready_seconds         = 0
          + paused                    = false
          + progress_deadline_seconds = 600
          + replicas                  = "1"
          + revision_history_limit    = 10

          + selector {
```

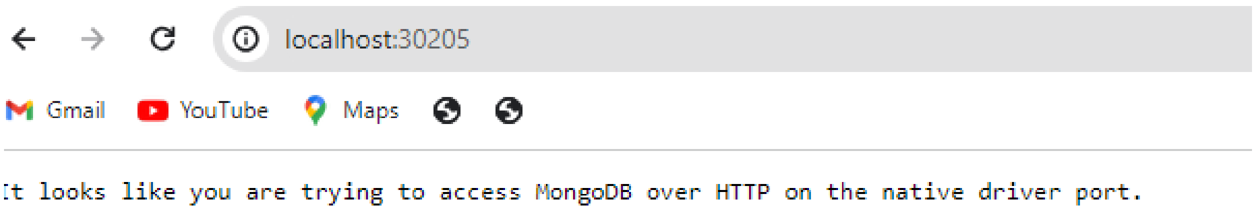
Ln 30, Col 50   Spaces: 2   UTF-8   CRLF   Plain Text

Implementación de apache



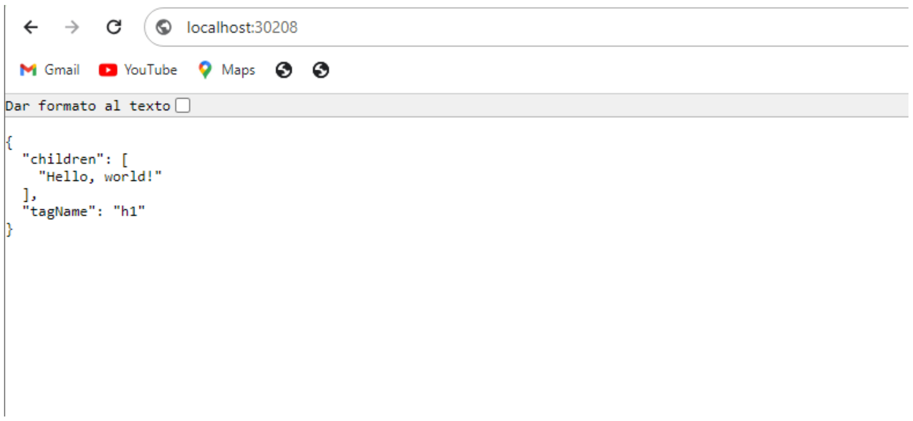
It works!

Implementación de MongoDb



```
It looks like you are trying to access MongoDB over HTTP on the native driver port.
```

Hellowrol con react



```
{
  "children": [
    "Hello, world!"
  ],
  "tagName": "h1"
}
```