

Proyecto - Daniel Gutierrez A0320176

Comandos necesarios

Para iniciar el pod con las replicas necesarias se debe ejecutar el comando

```
kubectl run web-deployment --image=dgutierrez64/nodejs --replicas=3 --port=5000 --labels=app=web
```

Otros comandos necesarios en la inicializacion

```
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl create -f deployment.yml
deployment "web-deployment" created
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl get deployments
NAME                DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
web-deployment      3          3          3             0            16s
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl get pods --show-labels
NAME                READY      STATUS    RESTARTS   AGE    LABELS
web-deployment-558c7648ff-bz9cr  0/1       Pending   0          41s    app=web,pod-template-hash=1147320499
web-deployment-558c7648ff-cmdfs  0/1       Pending   0          41s    app=web,pod-template-hash=1147320499
web-deployment-558c7648ff-pgfz2  0/1       Pending   0          41s    app=web,pod-template-hash=1147320499
```

```
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176/nodejs# kubectl get pods
NAME                READY      STATUS    RESTARTS   AGE
web-deployment-558c7648ff-bz9cr  0/1       Pending   0          1h
web-deployment-558c7648ff-cmdfs  0/1       Pending   0          1h
web-deployment-558c7648ff-pgfz2  0/1       Pending   0          1h
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176/nodejs# kubectl get services
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1     <none>         443/TCP    5h
```

Para exponer el servicio se utiliza

```
kubectl expose deployment web-deployment --name=nodejs --port=8000 --target-port=5000
```

```
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl create -f deployment.yml
deployment "web-deployment" created
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl get deployments
NAME                DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
web-deployment      3          3          3             0            3s
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl get services
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1     <none>         443/TCP    5h
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl expose deployment web-deployment --name=nodejs --port=8000 --target-port=5000
service "nodejs" exposed
root@ubuntu-xenial:/home/ubuntu/sd-project/A00320176# kubectl get services
NAME                TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1     <none>         443/TCP    5h
nodejs              ClusterIP   10.97.133.122 <none>         8000/TCP   2s
```

Archivos Dockerfile para los servicios empleados

Se creo una imagen que hace uso de un servidor Node.js. La imagen se creo a partir de la imagen node:carbon y se le agrego los archivos del servidor y se instalaron las dependencias

El codigo del servidor server.js

```
const express = require('express')
```

```
const app = express()
const os = require('os')
const PORT = 5000

app.get('/', (req, res) => res.send("<h1>Node.js App. Enviado desde el host " +
os.hostname() + "</h1>"))

app.listen(PORT, () => console.log("Aplicacion corriendo en el puerto " + PORT +
"!" ))
```

El archivo package.json para declarar informacion, scripts y dependencias de la aplicacion.

```
{
  "name": "docker_web_app",
  "version": "1.0.0",
  "description": "Node.js app para Docker y Kubernetes",
  "author": "Daniel Gutierrez <d.aguti@hotmail.com>",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.16.1"
  }
}
```

Se automatizo la creacion del contenedor de la aplicacion en el archivo Dockerfile

```
FROM node:carbon
ADD . /code
WORKDIR /code
RUN npm install
EXPOSE 5000
CMD ["npm", "start"]
```

Se creo la imagen a partir de este Dockerfile y se subio a Docker Cloud usando:

```
docker build -t dgutierrez64/nodejs .
docker login
docker tag dgutierrez64/nodejs dgutierrez64/nodejs
docker push dgutierrez64/nodejs
```

Archivos de configuración necesarios

deployment.yml

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: web-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
```

```
  app: web
spec:
  containers:
  - name: web
    image: dgutierrez64/nodejs
    ports:
    - containerPort: 5000
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nodejs
spec:
  ports:
  - port: 8000
    targetPort: 5000
    protocol: TCP
  selector:
    app: web
```

Funcionamiento de la
aplicacion

← → ↻ 🏠 ⓘ 192.168.10.20:8000 ☆ 🔒 🔴

Node.js App. Enviado desde el host ff498c94baa3

Problemas

Uno de los problemas mas importantes que se presentaron en el desarrollo del proyecto fue durante la ejecucion de los pods estos permanecian en estado PENDING, esto ocurrio por falta de memoria en el equipo. Dado que nativamente en mi equipo no esta funcionando Docker ni Kubernentes, tuve que crear una

maquina con Vagrant y a trabajar ahi por medio de la consola y de VIM. Esta maquina, por problemas del equipo se el asigno poca memoria, insuficiente para que se pudiera ejecutar bien el cluster.