

Proyecto Sistemas Distribuidos

Consigne los comandos de linux necesarios para el aprovisionamiento de los servicios empleados. En este punto no debe incluir archivos tipo Dockerfile solo se requiere que usted identifique los comandos o acciones que debe automatizar (15%)

Para verificar el estado de minikube:

```
minikube status
```

Consultar versión de minikube:

```
minikube version
```

Lanzar minikube

```
minikube start
```

Construir una imagen de docker desde un proyecto existente en python

```
cd Docker  
docker build -t rocco522/web .  
docker push rocco522/web  
curl localhost:5000
```

Desplegar la aplicación en Kubernetes

```
cd ../Kubernetes  
kubectl create -f db-pod.yml  
kubectl create -f db-svc.yml  
kubectl create -f web-pod.yml  
kubectl create -f web-svc.yml  
kubectl create -f web-rc.yml
```

Verificar que los pods y los servicios fueron creados

```
kubectl get pods  
kubectl get svc
```

Obtener el NodePort para el servicio web.

```
kubectl describe svc web
```

Probar la app

```
kubectl get nodes  
curl IP:PUERTO
```

Escriba los archivos Dockerfile para los servicios empleados junto con los archivos fuente necesarios, en el caso de emplear una imagen de docker hub debe incluir una explicación de lo realizado en el Dockerfile del repositorio de github. Tenga en cuenta consultar buenas prácticas para la elaboración de archivos Dockerfile. (20%)

```
FROM python:2.7-onbuild
EXPOSE 5000
CMD [ "python", "app.py" ]
```

Escriba los archivos de configuración necesarios deployment.yml, service.yml para el despliegue de la infraestructura (10%). Incluya un diagrama general de los componentes empleados.

```
apiVersion: "v1"
kind: Pod
metadata:
  name: redis
  labels:
    name: redis
    app: demo
spec:
  containers:
    - name: redis
      image: redis:latest
      ports:
        - containerPort: 6379
          protocol: TCP
```

```
apiVersion: v1
kind: Service
metadata:
  name: redis
  labels:
    name: redis
    app: demo
spec:
  ports:
```

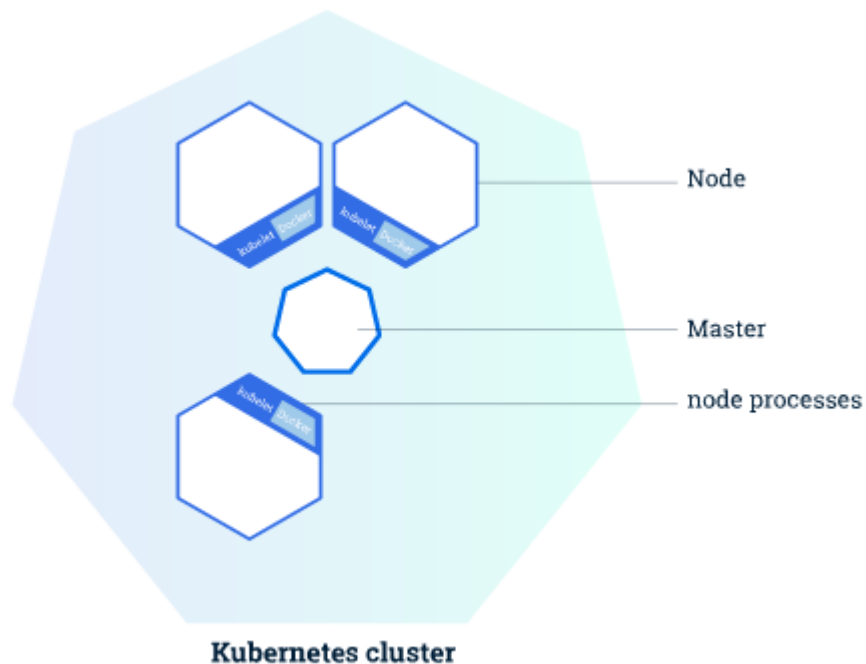
```
- port: 6379
  name: redis
  targetPort: 6379
selector:
  name: redis
  app: demo
```

```
apiVersion: "v1"
kind: Pod
metadata:
  name: web
  labels:
    name: web
    app: demo
spec:
  containers:
    - name: web
      image: rocco522/web
      ports:
        - containerPort: 5000
          name: http
          protocol: TCP
```

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: web
  labels:
    name: web
    app: demo
spec:
  replicas: 2
```

```
template:
  metadata:
    labels:
      name: web
  spec:
    containers:
      - name: web
        image: rocco522/web
        ports:
          - containerPort: 5000
            name: http
            protocol: TCP
```

```
apiVersion: v1
kind: Service
metadata:
  name: web
  labels:
    name: web
    app: demo
spec:
  selector:
    name: web
  type: NodePort
  ports:
    - port: 80
      targetPort: 5000
      protocol: TCP
```



El informe debe publicarse en un repositorio de github el cual debe ser un fork de <https://github.com/ICESI-Training/sd-project> y para la entrega deberá hacer un Pull Request (PR) respetando la estructura definida. El código fuente y la url de github deben incluirse en el informe (15%). Tenga en cuenta publicar los archivos para el aprovisionamiento

<https://github.com/rocco522/sd-project>



Incluya evidencias que muestran el funcionamiento de lo solicitado (15%)

```
Ricard@Ricardo MINGW64 /c/kubernetes-101-master/Kubernetes
$ minikube version
minikube version: v0.23.0

Ricard@Ricardo MINGW64 /c/kubernetes-101-master/Kubernetes
$ minikube start
Starting local Kubernetes v1.8.0 cluster...
Starting VM...
```



```
Ricard@Ricardo MINGW64 /c/kubernetes-101-master/Docker
$ docker push rocco522/web
The push refers to repository [docker.io/rocco522/web]
de87c75e24ce: Pushed
214ed7dfe4cb: Pushed
7b7def6d289d: Pushed
39b23cb6916d: Pushed
4dc256f2bb80: Pushed
8ab4bc20f81d: Pushed
8bb033bbebf8: Pushed
1b34d79f9112: Pushed
63866df00998: Pushed
2f9128310b77: Pushed
d9a5f9b8d5c2: Pushed
c01c63c6823d: Pushed
latest: digest: sha256:8e93d62824ad1a31bc6c08279ea08b9c9d097e0c859fd8d2fbfca93bb
b77351f size: 2843
```

 docker cloud  Swarm mode + Repo

Repositories

rocco522 / web


General


Tags


Builds

Timeline

Settings



 rocco522 / web

Proyecto Sistemas Distribuidos 

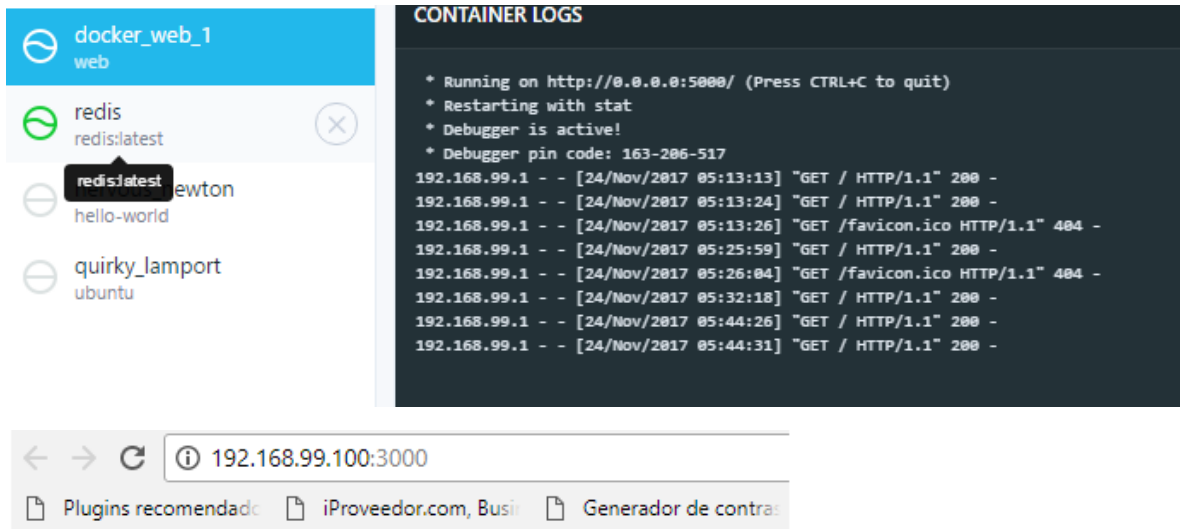
 Last pushed: 13 hours ago

Tags

This repository contains 1 tag(s).

latest		 13 hours ago
--------	---	--

[See all](#)



Documente algunos de los problemas encontrados y las acciones efectuadas para su solución al aprovisionar la infraestructura y aplicaciones (10%)

Durante el proceso de desarrollo del proyecto, el principal problema es que no tenía clara cuál era la función de kubernetes; es decir, teóricamente entendía para qué sirve, pero en la práctica no sabía como aprovechar sus servicios. Este fue un problema que se fue resolviendo durante el transcurso.