



Mayo 2022

MANUAL DE DESPLIEGUE

GRUPO NO.12



SISTEMAS OPERATIVOS II

Creación de dockerfile

Para empezar la configuración del despliegue de nuestra aplicación primero se procedió a realizar los dockerfile del backend y del frontend.

Se utiliza la imagen de golang, se crea una carpeta donde se copian todos los archivos necesarios para el servidor, luego de ello se hace una descarga de las librerías.

Se corre el servidor en el puerto 3000

```
1  # syntax=docker/dockerfile:1
2
3  FROM golang:1.16-alpine
4
5  WORKDIR /app
6
7  COPY go.mod ./
8  COPY go.sum ./
9  RUN go mod download
10
11 COPY . .
12
13
14 RUN go build -o /servidor
15
16 EXPOSE 3000
17
18 CMD [ "/servidor" ]
```

Creación de dockerfile

Se descarga la imagen base de node y luego se crea una carpeta donde se va a ejecutar el frontend.

se copia el package.json y un npm install para crear el node_modules, se copia en la carpeta y se hace un run build.

En la segunda fase se hace la configuración de nginx y se expone en el puerto 3001

```
1  # Creando build de la AppWeb
2  FROM node:14-alpine as build
3  WORKDIR /app
4  ENV PATH /app/node_modules/.bin:$PATH
5  COPY package.json ./
6  RUN npm install
7  COPY . ./
8  RUN npm run build
9
10 # configurando nginx
11 FROM nginx:stable-alpine
12 COPY --from=build /app/dist /usr/share/nginx/html
13 RUN rm /etc/nginx/conf.d/default.conf
14 # new
15 COPY nginx/nginx.conf /etc/nginx/conf.d/default.conf
16
17 EXPOSE 3001
18 CMD ["nginx", "-g", "daemon off;"]
```

Crear imágenes

Luego de crear los dockerfile se procedió a crear las imágenes, los comandos utilizados fueron los siguientes:

> docker build -t usuario/imagen:latest .

Luego de construir las imágenes se procede a subirlas en dockerhub

> docker push usuario/imagen:latest.

Windows PowerShell

```
oscar> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
oscarllamas6/frontend-changed	v1	1cf630107887	23 hours ago	40.4MB
oscarllamas6/frontend-sopes2	v2	e7009d6c1d9e	24 hours ago	40.4MB
oscarllamas6/frontend-sopes2	v1	2e218770d9d9	24 hours ago	40.4MB
oscarllamas6/backend-sopes2	v1	de21f869e940	3 days ago	581MB
oscarllamas6/tp7_backend_201602625	latest	5184df431618	6 weeks ago	581MB

oscarllamas6 | Search by repository name | Create Repository

oscarllamas6 / frontend-changed Last pushed: a day ago	Not Scanned	0	2	Public
oscarllamas6 / frontend-sopes2 Last pushed: a day ago	Not Scanned	0	4	Public
oscarllamas6 / backend-sopes2 Last pushed: 3 days ago	Not Scanned	0	4	Public

Creación de archivos YAML

Creación de los deployments y services

Se crea el un deployment para el backend en el namespace sopes2, de igual forma se crean 3 réplicas y se configura el rollingUpdate y se le asigna la imagen al deployment para crear los contenedores y pods.

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: backend-deployment
5    namespace: sopes2
6    labels:
7      app: backend-deployment
8  spec:
9    replicas: 3
10   minReadySeconds: 20
11   strategy:
12     type: RollingUpdate
13     rollingUpdate:
14       maxSurge: 1
15       maxUnavailable: 0
16   selector:
17     matchLabels:
18       app: backend-deployment
19   template:
20     metadata:
21       annotations:
22         linkerd.io/inject: enabled
23       labels:
24         app: backend-deployment
25     spec:
26       containers:
27       - name: backend-deployment
28         image: docker.io/oscarllamas6/backend-sopes2:v1
```

Creación de archivos YAML

Se crea el deployment del frontend con la imagen del frontend, de igual forma se crean dos replicas y el rollingUpdate, de igual forma se configuran las variables de entorno.

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: frontend-deployment
5    namespace: sopes2
6    labels:
7      app: frontend-deployment
8  spec:
9    replicas: 2
10   minReadySeconds: 20
11   strategy:
12     type: RollingUpdate
13     rollingUpdate:
14       maxSurge: 1
15       maxUnavailable: 0
16   selector:
17     matchLabels:
18       app: frontend-deployment
19   template:
20     metadata:
21       annotations:
22         linkerd.io/inject: enabled
23       labels:
24         app: frontend-deployment
25     spec:
26       containers:
27       - name: frontend-deployment
28         image: docker.io/oscarllamas6/frontend-sopes2:v2
29         imagePullPolicy: Always
30         ports:
31         - containerPort: 3001
```

Creación de archivos YAML - services

Se crean los servicios de ClusterIP para el backend y un loadBalancer para el frontend.

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: backend-service
5    namespace: sopes2
6    labels:
7      app: backend-deployment
8  spec:
9    type: ClusterIP
10   ports:
11     - port: 3000
12       targetPort: 3000
13       protocol: TCP
14   selector:
15     app: backend-deployment
```

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: frontend-service
5    namespace: sopes2
6  spec:
7    selector:
8      app: frontend-deployment
9    ports:
10     - port: 3001
11       targetPort: 3001
12   type: LoadBalancer
```

Configuración de Kubernetes

Setear gcloud y kubectl

- Instalar gcloud, correr el siguiente comando en Windows Powerll y ejecutar instalador

```
# Correr comando para iniciar configuración  
> gcloud init
```

```
#Se mostraraá un mensaje como el siguiente, darle "Y" para aceptar y logearnos en GCP
```

```
Network diagnostic detects and fixes local network connection issues.  
Checking network connection...done.  
Reachability Check passed.  
Network diagnostic passed (1/1 checks passed).
```

```
You must log in to continue. Would you like to log in (Y/n)? Y
```

```
# Se mostrará una lista de proyectos, escogemos el proyecto o creamos uno nuevo.
```

```
You are logged in as: [<your-account-email>].
```

```
Pick cloud project to use:
```

```
[1] sopes2-proyecto-329600
```

```
Please enter numeric choice or text value (must exactly match list item): 2
```


Configuración de Kubernetes

Setear gcloud y kubectl

- Instalar gcloud, correr el siguiente comando en Windows Powerll y ejecutar instalador

```
# Correr comando para iniciar configuración  
> gcloud init
```

```
#Se mostraraá un mensaje como el siguiente, darle "Y" para aceptar y logearnos en GCP
```

```
Network diagnostic detects and fixes local network connection issues.  
Checking network connection...done.  
Reachability Check passed.  
Network diagnostic passed (1/1 checks passed).
```

```
You must log in to continue. Would you like to log in (Y/n)? Y
```

```
# Se mostrará una lista de proyectos, escogemos el proyecto o creamos uno nuevo.
```

```
You are logged in as: [<your-account-email>].
```

```
Pick cloud project to use:
```

```
[1] sopes2-proyecto-329600
```

```
Please enter numeric choice or text value (must exactly match list item): 2
```

Configuración de Kubernetes

```
# Configuramos una región y zona predeterminada, ejemplo us-central1-a
> gcloud config set compute/zone us-central1-a

# Verificamos que haya sido configurada correctamente
> gcloud config list compute/zone

# Instalamos kubectl
> gcloud components install kubectl

# Creamos cluster kubernetes
> gcloud container clusters create squidgames --num-nodes=3 --no-enable-ip-alias

# Recuperando credenciales para Kubectl
> gcloud container clusters get-credentials k8s-demo --zone=us-central1-c

# Creamos reglas de firewall para los puertos
> gcloud compute firewall-rules create fwrule-kubernetes --allow tcp:30000-32767
```

Configuración de Kubernetes

Comandos kubectl

```
# Levantar recursos Frontend
> kubectl apply -f frontend-deployment.yaml -f frontend-service.yaml

# Borrar recursos Frontend
> kubectl delete -f frontend-deployment.yaml -f frontend-service.yaml

# Levantar recursos Backend
> kubectl apply -f backend-deployment.yaml -f backend-service.yaml

# Borrar recursos Backend
> kubectl delete -f backend-deployment.yaml -f backend-service.yaml
```

Seteando nueva imagen en deployment ejecutandose.

```
> kubectl set image deployments/<deployment name> <label name>=<new image registry name:tag>

# Ejemplo
> kubectl set image deployments/frontend-deployment frontend-deployment=oscarllamas6/frontend-changed:v1
```

Verificar estado del rolling update

```
> kubectl rollout status deployment/frontend-deployment
```

Retroceder con rollout updates

```
> kubectl rollout undo deployments/frontend-deployment
```

Prometheus y Grafana (Linkerd)

- Comandos para instalar y setear Linkerd

Configuración de Kubernetes

Comandos kubectl

```
# Levantar recursos Frontend
> kubectl apply -f frontend-deployment.yaml -f frontend-service.yaml

# Borrar recursos Frontend
> kubectl delete -f frontend-deployment.yaml -f frontend-service.yaml

# Levantar recursos Backend
> kubectl apply -f backend-deployment.yaml -f backend-service.yaml

# Borrar recursos Backend
> kubectl delete -f backend-deployment.yaml -f backend-service.yaml
```

Seteando nueva imagen en deployment ejecutandose.

```
> kubectl set image deployments/<deployment name> <label name>=<new image registry name:tag>

# Ejemplo
> kubectl set image deployments/frontend-deployment frontend-deployment=oscarllamas6/frontend-changed:v1
```

Verificar estado del rolling update

```
> kubectl rollout status deployment/frontend-deployment
```

Retroceder con rollout updates

```
> kubectl rollout undo deployments/frontend-deployment
```