

TALLER DE DE MODULARIZACIÓN CON VIRTUALIZACIÓN E INTRODUCCIÓN A DOCKER Y A AWS

Parte Docker

Para empezar, se crea un archivo Dockerfile con el contenido que se muestra a continuación.

```
SparkWebServer.java DockerFile x
1 FROM openjdk:8
2
3 WORKDIR /usrapp/bin
4
5 ENV PORT 6000
6
7 COPY /target/classes /usrapp/bin/classes
8 COPY /target/dependency /usrapp/bin/dependency
9
10 CMD ["java", "-cp", "./classes:./dependency/*", "edu.escuelaing.arep.sparkwebDocker.SparkWebServer"]
```

Se construye una imagen en Docker con el comando `docker build --tag dockersparksprimer .`

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker build --tag dockersparksprimer .
[+] Building 70.4s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 277B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8
=> [1/4] FROM docker.io/library/openjdk:8@sha256:51ab855b0a8ee08813da2d1d4a71aa8901f4efba23b6c206a3a7d3e8a3a557f
=> => resolve docker.io/library/openjdk:8@sha256:51ab855b0a8ee08813da2d1d4a71aa8901f4efba23b6c206a3a7d3e8a3a557f
=> => sha256:76578921073d9c39c3c53e7717e88b389820749891b3fac6564f4a0ea4347d1a 1.79kB / 1.79kB
=> => sha256:51ab855b0a8ee08813da2d1d4a71aa8901f4efba23b6c206a3a7d3e8a3a557f 1.04kB / 1.04kB
=> => sha256:08121337b7a4972212918a3fa235acd34442af8c2dafbea92fa7a7f54f9dc2c4 7.81kB / 7.81kB
=> => sha256:911ea9f2bd51e53a455297e0631e18a72a86d7e2c8e1807176e80f991bde5d64 10.87MB / 10.87MB
=> => sha256:955615a668ce169f8a1443fc6b6e6215f43fe0babfb4790712a2d3171f34d366 54.93MB / 54.93MB
=> => sha256:2756f5f69a5190f4308619e0f446d95f5515eef4a814dbad0bcebbbbc7b25a8 5.15MB / 5.15MB
=> => sha256:27b0a22ee906271a6ce9ddd1754fdd7d3b59078e0b57b6cc054c7ed7ac301587 54.57MB / 54.57MB
=> => sha256:785dfffb36c6c7fbfd1c98d950a9b49054e84bccdd4b54ebfab50b0e93dbfeefe 5.42MB / 5.42MB
=> => sha256:a83d4d9f417102ab5adf274b0cb1cabca8332b8c7931b50e6dc35cc1cc357824 207B / 207B
=> => sha256:88ab6c8cf10d06acbe1d3594aff80604ae82a4f71a27afef22ca6ff57d1e7988 105.99MB / 105.99MB
=> => extracting sha256:955615a668ce169f8a1443fc6b6e6215f43fe0babfb4790712a2d3171f34d366
=> => extracting sha256:2756f5f69a5190f4308619e0f446d95f5515eef4a814dbad0bcebbbbc7b25a8
=> => extracting sha256:911ea9f2bd51e53a455297e0631e18a72a86d7e2c8e1807176e80f991bde5d64
=> => extracting sha256:27b0a22ee906271a6ce9ddd1754fdd7d3b59078e0b57b6cc054c7ed7ac301587
=> => extracting sha256:785dfffb36c6c7fbfd1c98d950a9b49054e84bccdd4b54ebfab50b0e93dbfeefe
=> => extracting sha256:a83d4d9f417102ab5adf274b0cb1cabca8332b8c7931b50e6dc35cc1cc357824
=> => extracting sha256:88ab6c8cf10d06acbe1d3594aff80604ae82a4f71a27afef22ca6ff57d1e7988
=> [internal] load build context
=> => transferring context: 3.23MB
=> [2/4] WORKDIR /usrapp/bin
=> [3/4] COPY /target/classes /usrapp/bin/classes
=> [4/4] COPY /target/dependency /usrapp/bin/dependency
=> exporting to image
=> => exporting layers
=> => writing image sha256:2cd9da39a6fbd8d8da52ef22c3bd7415a67d13227df66ba385d24cfeb912227
=> => naming to docker.io/library/dockersparksprimer

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Con el comando Docker images se comprueba que la imagen fue construida.

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
dockersparkspringer latest          2cd9da39a6fb   2 minutes ago  529MB
```

Se crean tres instancias de un contenedor Docker con el puerto 6000 enlazado a un puerto físico de nuestra máquina.

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker run -d -p 34000:6000 --name firstdockercontainer dockersparkspringer
6e510ba6f98776fcf54c86f0798bb38fc7990b55ceb3c9a7df7becf8fdb2aae6

C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker run -d -p 34001:6000 --name firstdockercontainer2 dockersparkspringer
8fe024f4406a8b05334ee6780231c71f4f5e7bad5b03503b93ccf42cd62cd621

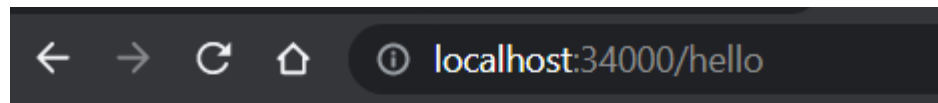
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker run -d -p 34002:6000 --name firstdockercontainer3 dockersparkspringer
ab6d44e22b74054f765b3cd4855579b4908257e50ef651f607957bd6e89a6157
```

Con Docker ps se puede observar si el contenedor está corriendo.

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED        STATUS        PORTS                               NAMES
ab6d44e22b74   dockersparkspringer "java -cp ./classes:..." 25 seconds ago Up 23 seconds 0.0.0.0:34002->6000/tcp, :::34002->6000/tcp firstdockercontainer3
8fe024f4406a   dockersparkspringer "java -cp ./classes:..." 37 seconds ago Up 35 seconds 0.0.0.0:34001->6000/tcp, :::34001->6000/tcp firstdockercontainer2
6e510ba6f987   dockersparkspringer "java -cp ./classes:..." 54 seconds ago Up 49 seconds 0.0.0.0:34000->6000/tcp, :::34000->6000/tcp firstdockercontainer
```

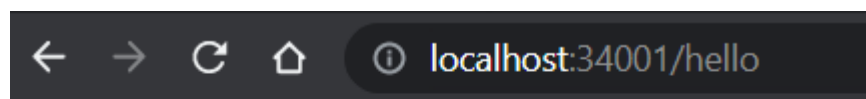
Se accede al navegador y se comprueba con las tres instancias creadas que funcionen.

http://localhost:34000/hello



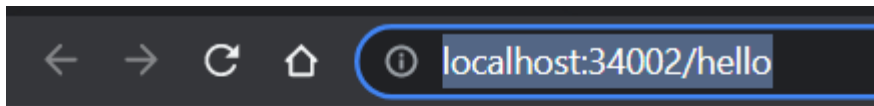
Hello Docker!

http://localhost:34001/hello



Hello Docker!

http://localhost:34002/hello



Hello Docker!

Ahora se creará el archivo Docker-compose.yml con el contenido en la siguiente imagen, este archivo es una configuración con una instancia a mongodb.

```
SparkWebServer.java  DockerFile  docker-compose.yml ×
1 version: '2'
2
3
4 services:
5     web:
6         build:
7             context: .
8             dockerfile: Dockerfile
9         container_name: web
10        ports:
11            - "8087:6000"
12        db:
13            image: mongo:3.6.1
14            container_name: db
15            volumes:
16                - mongodb:/data/db
17                - mongodb_config:/data/configdb
18            ports:
19                - 27017:27017
20            command: mongod
21
22 volumes:
23     mongodb:
24     mongodb_config:
```

Se ejecuta el comando docker-compose up -d para utilizar la configuración que se ha creado.

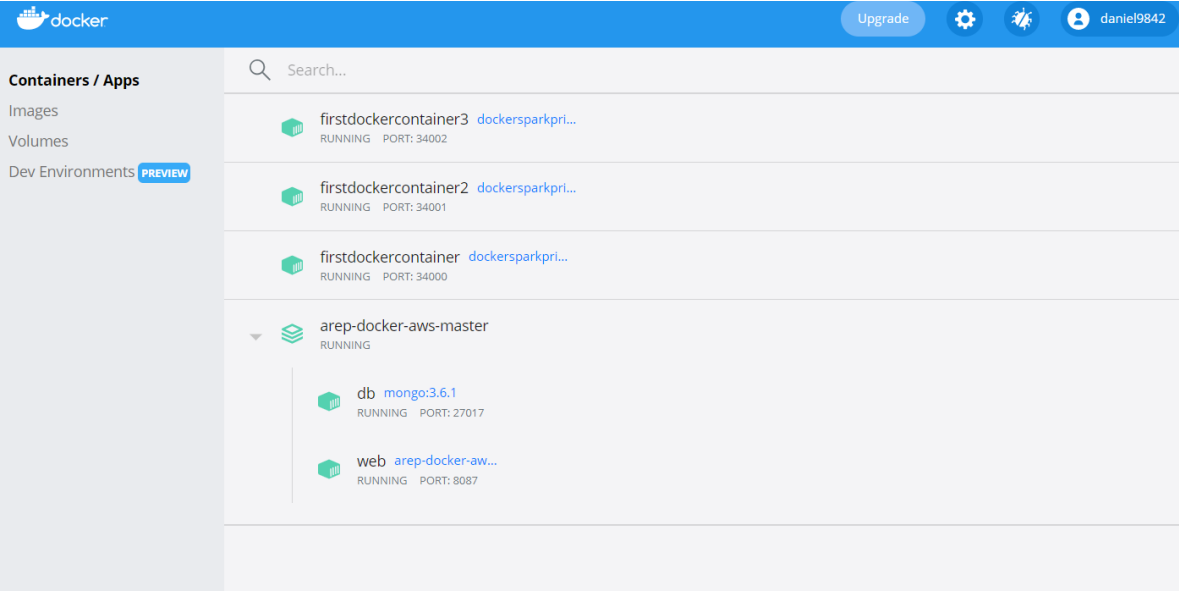
```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker-compose up -d
Creating network "arep-docker-aws-master_default" with the default driver
Creating volume "arep-docker-aws-master_mongodb" with default driver
Creating volume "arep-docker-aws-master_mongodb_config" with default driver
Building web
[+] Building 13.9s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 277B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:8
=> [1/4] FROM docker.io/library/openjdk:8@sha256:51ab8555b0a8ee88813da2d1d4a71aa8901f4efba23b6c206a3a7d3e8a3a557f
=> [internal] load build context
=> => transferring context: 3.23MB
=> CACHED [2/4] WORKDIR /usrapp/bin
=> CACHED [3/4] COPY /target/classes /usrapp/bin/classes
=> CACHED [4/4] COPY /target/dependency /usrapp/bin/dependency
=> exporting to image
=> => exporting layers
=> => writing image sha256:2cd9da39a6fbd8d88da52ef22c3bd7415a67d13227df66ba385d24cfeb912227
=> => naming to docker.io/library/arep-docker-aws-master_web

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
WARNING: Image for service web was built because it did not already exist. To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.
Pulling db (mongo:3.6.1)...
3.6.1: Pulling from library/mongo
c4bb02b17bb4: Pull complete
3f58e3bb3be4: Pull complete
a229fb575a6e: Pull complete
8f5ddc533743: Pull complete
5e9d2af6e206: Pull complete
3b6c28c0235b: Pull complete
fd6b165aa317: Pull complete
772467f0b4cd: Pull complete
094d919fbb86: Pull complete
b0cad17917cd: Pull complete
Digest: sha256:c78f6debfb5b10fe2ed390105a729123f3365a33e5aada6f5539922d1d7c75dc
Status: Downloaded newer image for mongo:3.6.1
Creating db ... done
Creating web ... done
```

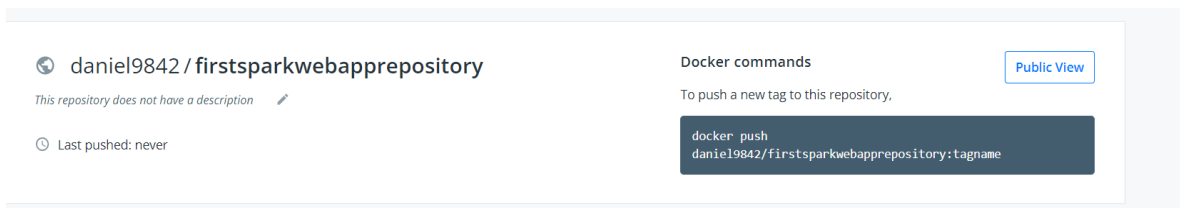
Con el comando Docker ps se comprobará que el servicio a mongodb este ejecutándose.

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
a561489cfd40   mongo:3.6.1                        "docker-entrypoint.s..." 58 seconds ago Up 54 seconds 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp
fec7810bc574   arep-docker-aws-master_web         "java -cp ./classes:..." 58 seconds ago Up 54 seconds 0.0.0.0:8087->6000/tcp, :::8087->6000/tcp
ab6d44e22b74   dockersparkspringer               "java -cp ./classes:..." 11 minutes ago Up 11 minutes 0.0.0.0:34002->6000/tcp, :::34002->6000/tcp
8fe024f4406a   dockersparkspringer               "java -cp ./classes:..." 11 minutes ago Up 11 minutes 0.0.0.0:34001->6000/tcp, :::34001->6000/tcp
6e510ba6f987   dockersparkspringer               "java -cp ./classes:..." 11 minutes ago Up 11 minutes 0.0.0.0:34000->6000/tcp, :::34000->6000/tcp
```

En la aplicación Docker Destop se debería ver algo similar a la siguiente imagen.



Se crea un repositorio en dockerhub.



Se crea una referencia a la imagen creada con el nombre del repositorio a donde desea subirla.

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker tag dockersparkprimer daniel9842/firstsparkwebapprepository
```


Con el comando Docker images se observa que la nueva referencia ha sido creada.


```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker images
REPOSITORY                                TAG      IMAGE ID      CREATED        SIZE
arep-docker-aws-master_web                latest   2cd9da39a6fb  40 minutes ago  529MB
dockersparkprimer                        latest   2cd9da39a6fb  40 minutes ago  529MB
daniel9842/firstsparkwebapprepository     latest   2cd9da39a6fb  40 minutes ago  529MB
mongo                                     3.6.1    1200574c8af9  3 years ago    366MB
```


Se sube la imagen al repositorio creado con el comando Docker push y el nombre del repositorio.

```
C:\Users\Frank\Documents\Daniel Taller\ArepDocker\Arep-Docker-AWS-master>docker push daniel9842/firstsparkwebapprepository:latest
The push refers to repository [docker.io/daniel9842/firstsparkwebapprepository]
8ffd2d270ab5: Pushed
5c73db76091b: Pushed
ecff08d3ce8e: Pushed
d9b6ea8e7d5f: Mounted from library/openjdk
9cb5eb95298c: Mounted from library/openjdk
00ef5416d927: Pushed
8555e663f65b: Pushed
d00da3cd7763: Mounted from library/openjdk
4e61e63529c2: Mounted from library/openjdk
799760671c38: Mounted from library/openjdk
latest: digest: sha256:7f1b55842dc8a60a7bb0b3406b6e3c75076a364726df2b71a787ab15d0bb1c39 size: 2421
```


Una vez subida la imagen se debe ver algo similar a la siguiente imagen.

 **daniel9842 / firstsparkwebapprepository**



This repository does not have a description 

 Last pushed: a minute ago

Tags and Scans

 **VULNERABILITY SCANNING - DISABLED**
[Enable](#)

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
 latest		a minute ago	a minute ago

parte AWS

Una vez creada la instancia de la máquina virtual en aws se accede y con el comando `sudo yum update -y` se actualiza la máquina virtual.

```
[ec2-user@ip-172-31-88-214 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.7 kB 00:00:00
Resolving Dependencies
--> Running transaction check
--> Package curl.x86_64 0:7.76.1-4.amzn2.0.1 will be updated
--> Package curl.x86_64 0:7.76.1-7.amzn2.0.2 will be an update
--> Package device-mapper.x86_64 7:1.02.146-4.amzn2.0.2 will be updated
--> Package device-mapper.x86_64 7:1.02.170-6.amzn2.5 will be an update
--> Package device-mapper-event.x86_64 7:1.02.146-4.amzn2.0.2 will be updated
--> Package device-mapper-event.x86_64 7:1.02.170-6.amzn2.5 will be an update
--> Package device-mapper-event-libs.x86_64 7:1.02.146-4.amzn2.0.2 will be updated
--> Package device-mapper-event-libs.x86_64 7:1.02.170-6.amzn2.5 will be an update
--> Package device-mapper-libs.x86_64 7:1.02.146-4.amzn2.0.2 will be updated
--> Package device-mapper-libs.x86_64 7:1.02.170-6.amzn2.5 will be an update
--> Package glibc.x86_64 0:2.26-48.amzn2 will be updated
--> Package glibc.x86_64 0:2.26-54.amzn2 will be an update
--> Package glibc-all-langpacks.x86_64 0:2.26-48.amzn2 will be updated
--> Package glibc-all-langpacks.x86_64 0:2.26-54.amzn2 will be an update
--> Package glibc-common.x86_64 0:2.26-48.amzn2 will be updated
--> Package glibc-common.x86_64 0:2.26-54.amzn2 will be an update
--> Package glibc-locale-source.x86_64 0:2.26-48.amzn2 will be updated
--> Package glibc-locale-source.x86_64 0:2.26-54.amzn2 will be an update
--> Package glibc-minimal-langpack.x86_64 0:2.26-48.amzn2 will be updated
--> Package glibc-minimal-langpack.x86_64 0:2.26-54.amzn2 will be an update
```

Con el comando `sudo yum install Docker` se instala el Docker en la máquina virtual.

```
[ec2-user@ip-172-31-88-214 ~]$ sudo yum install docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.7-1.amzn2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.7-1.amzn2.x86_64
--> Processing Dependency: libcgroupp >= 0.40.rc1-5.15 for package: docker-20.10.7-1.amzn2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.7-1.amzn2.x86_64
--> Processing Dependency: pigz for package: docker-20.10.7-1.amzn2.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.4.6-2.amzn2 will be installed
--> Package libcgroupp.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.0.0-1.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
docker x86_64 20.10.7-1.amzn2 amzn2extra-docker 42 M
Installing for dependencies:
containerd x86_64 1.4.6-2.amzn2 amzn2extra-docker 24 M
libcgroupp x86_64 0.41-21.amzn2 amzn2-core 66 k
pigz x86_64 2.3.4-1.amzn2.0.1 amzn2-core 81 k
runc x86_64 1.0.0-1.amzn2 amzn2extra-docker 3.3 M
=====

Total 34 MB/s | 69 MB 00:00:02
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : runc-1.0.0-1.amzn2.x86_64 1/5
Installing : containerd-1.4.6-2.amzn2.x86_64 2/5
Installing : libcgroupp-0.41-21.amzn2.x86_64 3/5
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
Installing : docker-20.10.7-1.amzn2.x86_64 5/5
Verifying : containerd-1.4.6-2.amzn2.x86_64 1/5
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 2/5
Verifying : libcgroupp-0.41-21.amzn2.x86_64 3/5
Verifying : docker-20.10.7-1.amzn2.x86_64 4/5
Verifying : runc-1.0.0-1.amzn2.x86_64 5/5

Installed:
docker.x86_64 0:20.10.7-1.amzn2

Dependency Installed:
containerd.x86_64 0:1.4.6-2.amzn2 libcgroupp.x86_64 0:0.41-21.amzn2 pigz.x86_64 0:2.3.4-1.amzn2.0.1
runc.x86_64 0:1.0.0-1.amzn2

Complete!
```

Con el comando `sudo service docker start` se inicia el servicio de Docker.

```
[ec2-user@ip-172-31-88-214 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-88-214 ~]$
```

Con el comando `sudo usermod -a -G docker ec2-user` se configura el usuario en el grupo de Docker.

```
[ec2-user@ip-172-31-88-214 ~]$ sudo usermod -a -G docker ec2-user
```

Con el comando de la siguiente imagen se creará una instancia del contenedor Docker con el nombre que se elija y con el nombre del repositorio creado anteriormente.

```
[ec2-user@ip-172-31-88-214 ~]$ docker run -d -p 42000:6000 --name firstdockerimageaws daniel9842/firstsparkwebapprepository
Unable to find image 'daniel9842/firstsparkwebapprepository:latest' locally
latest: Pulling from daniel9842/firstsparkwebapprepository
955615a668ce: Pull complete
2756ef5f69a5: Pull complete
911ea9f2bd51: Pull complete
27b0a22ee906: Pull complete
785dfffb36c6c: Pull complete
a83d4d9f4171: Pull complete
88ab6c8cf10d: Pull complete
17e240567c8b: Pull complete
804fce344298: Pull complete
7f67d81380d8: Pull complete
Digest: sha256:7f1b55842dc8a60a7bb0b3406b6e3c75076a364726df2b71a787ab15d0bb1c39
Status: Downloaded newer image for daniel9842/firstsparkwebapprepository:latest
75f8d9fc54277cbc39b5eac098d5fd9e35104d8b46116a74e030bda1a9bc18a8
[ec2-user@ip-172-31-88-214 ~]$
```

Se verifica que la imagen haya sido creada con el comando Docker ps.

```
[ec2-user@ip-172-31-88-214 ~]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
75f8d9fc5427   daniel9842/firstsparkwebapprepository "java -cp ./classes:..." 44 seconds ago Up 43 seconds 0.0.0.0:42000->6000/tcp, :::42000->6000/tcp
firstdockerimageaws
```

Ahora es necesario habilitar el puerto 42000, para esto editamos las reglas de entrada de nuestra máquina virtual y se crea la regla para este puerto.

Editar reglas de entrada [Información](#)

Las reglas de entrada controlan el tráfico entrante que puede llegar a la instancia.

Security group rule ID	Tipo Información	Protocolo Información	Intervalo de puertos Información	Origen Información	Descripción: opcional Información	
sgr-0c68757c63f776d7c	SSH	TCP	22	Person... <input type="text" value="0.0.0.0"/>		Eliminar
-	TCP personalizado	TCP	42000	Anywh... <input type="text" value="0.0.0.0"/>	my webapp	Eliminar

[Agregar regla](#)

Se verifica la url obtenida por parte de aws y agregamos el puerto y /hello para obtener el siguiente resultado.

← → ↻ 🏠 ⚠ No es seguro | ec2-54-84-109-84.compute-1.amazonaws.com:42000/hello

Hello Docker!