

# Temario

1.- Creación del servicio REST .....	2
2.- Creación del jar .....	6
3.- Crear imagen Docker del microservicio con Dockerfile y subirlo a Docker Hub .....	9
4.- Instalar y configurar Rancher.....	12
5.- Generar la llave.....	15
6.- Registrar equipo.....	17
7.- Crear un servicio.....	22
8.- Crear un balanceador .....	24
9.- Consulta en el Navegador.....	26
10.- Escalando el servicio .....	28

## 1.- Creación del servicio REST

Para crear una vertical de se debe de clonar el proyecto de GitHub mediante el comando de consola:

\$ git clone <https://github.com/arellano-gustavo/vertx-sample.git>

Lo que nos generará una carpeta llamada "vertx-sample".

```
gustavo@ubuntu:~/curso-dgp$ git clone https://github.com/arellano-gustavo/vertx-sample.git
Cloning into 'vertx-sample'...
remote: Counting objects: 80, done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 80 (delta 12), reused 73 (delta 10), pack-reused 0
Unpacking objects: 100% (80/80), done.
Checking connectivity... done.
gustavo@ubuntu:~/curso-dgp$ ls
vertx-sample
```

Lo que haremos será crear otro proyecto en Github llamado servicio y se debe clonar en el equipo a realizar el servicio:

\$ git clone <https://github.com/dgpecurso12/servicio.git>

Lo que nos generará un carpeta llamada "servicio".

```
gustavo@ubuntu:~/curso-dgp$ git clone https://github.com/dgpecurso12/servicio.git
Cloning into 'servicio'...
remote: Counting objects: 3, done.
Unpacking objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Checking connectivity... done.
gustavo@ubuntu:~/curso-dgp$ ls
servicio  vertx-sample
gustavo@ubuntu:~/curso-dgp$
```

Procederemos a copiar el contenido de la carpeta vertx-sample a servicio:

```
gustavo@ubuntu:~/curso-dgp$ cp -r vertx-sample/src/ servicio/
gustavo@ubuntu:~/curso-dgp$ cp -r vertx-sample/LICENSE servicio/
gustavo@ubuntu:~/curso-dgp$ cp -r vertx-sample/pom.xml servicio/
gustavo@ubuntu:~/curso-dgp$ ls servicio/
LICENSE  pom.xml  README.md  src
gustavo@ubuntu:~/curso-dgp$
```

Se procede a agregar los archivos al git con el comando “git add .” y hacer commit con el comando “git commit -m “Comentario” “ en el proyecto.

```
gustavo@ubuntu:~/curso-dgp/servicio$ git add .
gustavo@ubuntu:~/curso-dgp/servicio$ git commit -m "Se agregan archivo de vertical al"
[master 8f8035f] Se agregan archivo de vertical
 9 files changed, 531 insertions(+)
 create mode 100644 LICENSE
 create mode 100644 pom.xml
 create mode 100644 src/.DS_Store
 create mode 100644 src/main/.DS_Store
 create mode 100644 src/main/java/mx/unam/dgpe/sample/controller/MyController.java
 create mode 100644 src/main/resources/log4j.properties
 create mode 100644 src/test/.DS_Store
 create mode 100644 src/test/java/mx/unam/dgpe/sample/controller/RestUtil.java
 create mode 100644 src/test/java/mx/unam/dgpe/sample/controller/TestMyController.java
gustavo@ubuntu:~/curso-dgp/servicio$
```

Se hace push al proyecto “git push”, y se ingresan las credencial si es que no se configuración globalmente.

```
gustavo@ubuntu:~/curso-dgp/servicio$ git push
warning: push.default is unset; its implicit value has changed in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the traditional behavior, use:

    git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

    git config --global push.default simple

When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.

Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com':
```

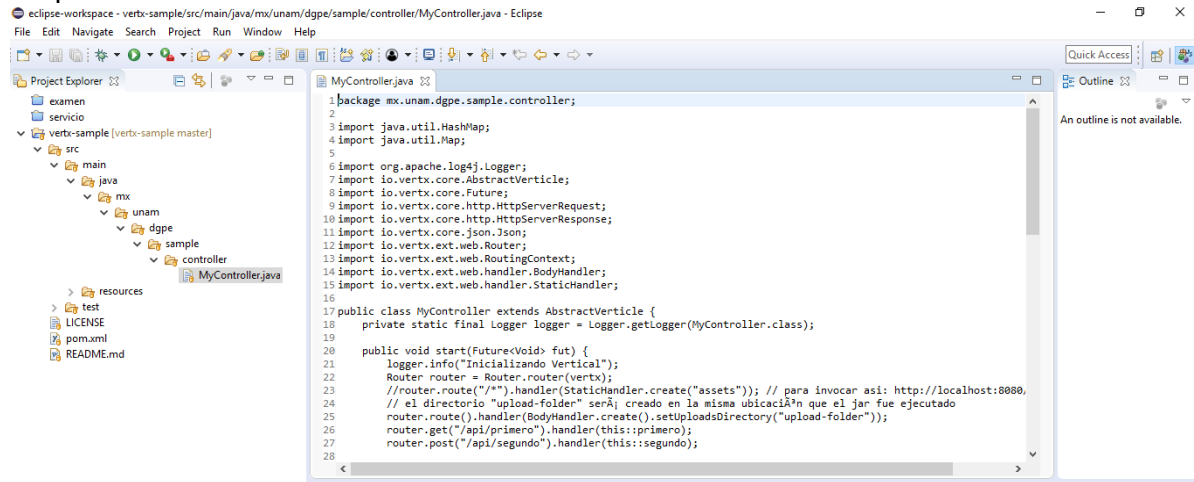
Con esto se tendrá un proyecto nuevo con el clone de vertx-sample, para poder modificar y generar un nuevo servicio rest.

Dentro de la carpeta llamada “servicio”, la cual contiene una vertical en donde solo es necesario modificar el archivo:

“/src/main/java/mx/unam/dgpe/sample/controller/MyController.java”

Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

Se puede abrir el archivo con su editor de texto favorito o un IDE de desarrollo:



Esta clase de java tiene un método llamado:

public void start(Future<Void> fut)

```
20 public void start(Future<Void> fut) {
21     logger.info("Iniciando Vertical");
22     Router router = Router.router(vertx);
23     //router.route("/").handler(StaticHandler.create("assets")); // para invocar asi: http://localhost:8080,
24     // el directorio "upload-folder" serÃ¡ creado en la misma ubicaciÃ³n que el jar fue ejecutado
25     router.route().handler(BodyHandler.create().setUploadsDirectory("upload-folder"));
26     router.get("/api/primero").handler(this::primero);
27     router.post("/api/segundo").handler(this::segundo);
28 }
```

En el cual vamos a definir dos partes:

1.- El punto de entrada de nuestra acci3n, para ello debemos de agregar al objeto "router" el punto de entrada e indicarle con qu3 m3todo de la clase ser3 resuelto, esto se hace mediante el c3digo:

```
16 router.get("/app/suma").handler(this::suma);
17 public class MyController extends AbstractVerticle {
18     private static final Logger logger = Logger.getLogger(MyController.class);
19
20     public void start(Future<Void> fut) {
21         logger.info("Iniciando Vertical");
22         Router router = Router.router(vertx);
23         //router.route("/").handler(StaticHandler.create("assets")); // para invocar asi: http://localhost:8080,
24         // el directorio "upload-folder" serÃ¡ creado en la misma ubicaciÃ³n que el jar fue ejecutado
25         router.route().handler(BodyHandler.create().setUploadsDirectory("upload-folder"));
26         router.get("/api/primero").handler(this::primero);
27         router.post("/api/segundo").handler(this::segundo);
28         router.get("/api/suma").handler(this::suma);
29
30         // Create the HTTP server and pass the "accept" method to the request handler.
31         vertx.createHttpServer().requestHandler(router::accept).listen(
32             config().getInteger("http.port", 8080), result -> {
33                 if (result.succeeded()) {
34                     fut.complete();
35                 }
36             }
37         );
38     }
39 }
```

2.- El método de la clase que atiende la petición que es ejecutada por el método “get” del protocolo de HTTP, en este caso se ejecuta “suma”

```
69     info.put("nombre", "gustavo");
70     info.put("edad", "21");
71     info.put("autos", autos);
72     return Json.encodePretty(info);
73 }
74
75 private void suma(RoutingContext routingContext) {
76     HttpServletResponse response = routingContext.response();
77     HttpServletRequest request = routingContext.request();
78     String operacion = "suma";
79     String operandoA = request.getParam("a");
80     String operandoB = request.getParam("b");
81     String jsonResponse = calculadora(operacion, operandoA, operandoB, request);
82     response.setStatusCode(200);
83     putHeader("content-type", "application/json; charset=utf-8");
84     end(jsonResponse);
85 }
86
```

Para cada acción se debe de generar un método que atienda dicha petición y en el cual se regrese una cadena en formato JSON, se recomienda usar Mapas, ya que estos son sencillos de generar y su traducción a json no es compleja.

En este caso el método calculadora regresa el resultado de la operación en formato de JSON al método suma.

```
86
87 private String calculadora(String tipo, String operandoA, String operandoB, HttpServletRequest request) {
88
89     Double r = new Double("0.0");
90     Double a = new Double("0.0");
91     Double b = new Double("0.0");
92     a = a.parseDouble(operandoA);
93     b = b.parseDouble(operandoB);
94
95     if(tipo.equals("suma")){
96         r = a+b;
97     }else if(tipo.equals("resta")){
98         r = a-b;
99     }else if(tipo.equals("multiplica")){
100         r = a*b;
101     }else if(tipo.equals("divide")){
102         r = a/b;
103     }else{
104         r=-1.0;
105     }
106
107     Map<String, String> resultado = new HashMap<>();
108     resultado.put("operacion", tipo);
109     resultado.put("resultado", ""+r);
110     resultado.put("Local IP", request.getLocalAddress().host());
111     resultado.put("Remote IP", request.remoteAddress().host());
112
113     return Json.encodePretty(resultado);
114 }
115
```

## 2.- Creación del jar

Una vez que hemos terminado de modificar la vertical de nuestro servicio, se necesita compilar el proyecto, para no tener que instalar java y maven, utilizaremos la imagen de docker denominada kebbelar/jdk18-utf8-debug-maven, la cual está almacenada en Dockerhub y por ende se debe de tener acceso a internet.

Para bajar el docker ejecutamos el comando:

\$ docker pull kebbelar/jdk18-utf8-debug-maven

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker pull kebbelar/jdk18-utf8-debug-maven
Using default tag: latest
latest: Pulling from kebbelar/jdk18-utf8-debug-maven
72b39c1d4615: Already exists
46a2d5ede4a6: Already exists
d7caf6e91ad4: Already exists
c7ac9f284354: Already exists
a3ed95caeb02: Already exists
d498714eb2ba: Already exists
0023902da944: Pull complete
3fff4566cc7c: Extracting 11.01 MB/38.18 MB
fc7a5f6e7016: Download complete
643624fa1fd0: Download complete
d166eb0825cb: Download complete
ff7290c6f2fc: Download complete
d23b98058dae: Download complete
e0806ae83136: Download complete
c08b017ea671: Downloading 92.97 MB/1.541 GB
```

El comando para compilar el proyecto es el siguiente:

\$ docker run -it -v /\${RUTAFISICA}:/codigo kebbelar/jdk18-utf8-debug-maven mvn -f /codigo clean package

Donde \${RUTAFISICA} es la ruta donde se tiene el proyecto a compilar.

\${RUTAFISICA}=~/home/gustavo/curso-dgp/servicio/

\$ docker run -it -v /home/gustavo/curso-dgp/servicio:/codigo kebbelar/jdk18-utf8-debug-maven mvn -f /codigo clean package

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker run -it -v /home/gustavo/curso-dgp/servicio:/codigo kebbelar/jdk18-utf8-debug-maven mvn -f /codigo clean package
```

Si no hubo errores en el código debe aparecer la pantalla con **BUILD SUCCESS**

```
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 39.556 s
[INFO] Finished at: 2018-06-14T09:08:18Z
[INFO] Final Memory: 24M/69M
[INFO] -----
gustavo@ubuntu:~/curso-dgp/servicio$
```

El resultado será el archivo sample-1.0-SNAPSHOT-fat.jar dentro del directorio /home/gustavo/curso-dgp/servicio/target

```
gustavo@ubuntu:~/curso-dgp/servicio$ ls
LICENSE  log  pom.xml  README.md  src  target
gustavo@ubuntu:~/curso-dgp/servicio$ ls target/
classes          maven-archiver  surefire-reports
generated-sources  sample-1.0-SNAPSHOT-fat.jar  test-classes
generated-test-sources  sample-1.0-SNAPSHOT.jar
gustavo@ubuntu:~/curso-dgp/servicio$
```

Para probar nuestro jar utilizaremos el docker gustavoarellano/jdk18, sino lo tenemos se baja con el comando:

\$ docker pull gustavoarellano/jdk18

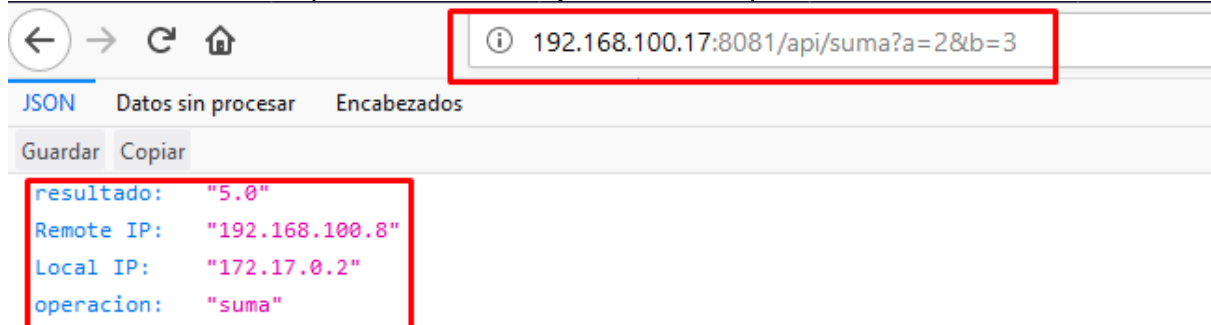
```
gustavo@ubuntu:~/curso-dgp/servicio$ docker pull gustavoarellano/jdk18
Using default tag: latest
latest: Pulling from gustavoarellano/jdk18
72b39c1d4615: Already exists
46a2d5ede4a6: Already exists
d7caf6e91ad4: Already exists
c7ac9f284354: Already exists
a3ed95caeb02: Already exists
d498714eb2ba: Already exists
Digest: sha256:68ac31c634b3e255399ed39a9e519ccc9cd2dc653360a0e2f6ed194a5a90827d
Status: Downloaded newer image for gustavoarellano/jdk18:latest
gustavo@ubuntu:~/curso-dgp/servicio$
```

El comando para ejecutar el jar con el docker es:

\$ docker run -d -p 8081:8080 -v /home/gustavo/curso-dgp/servicio:/codigo  
gustavoarellano/jdk18 java -jar /codigo/target/sample-1.0-SNAPSHOT-fat.jar

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker run -d -p 8081:8080 -v /home/gustavo/curso-dgp/servicio:/codigo gustavoarellano/jdk18 java -jar /codigo/target/sample-1.0-SNAPSHOT-fat.jar
3bea0b022c30966636f9a552fcb99a4150c63110c2e52b2385ac0d8fdaa1e22a
gustavo@ubuntu:~/curso-dgp/servicio$
```

Ya que se inició el docker con el jar, en un navegador pondremos la dirección IP de nuestro equipo donde está el docker seguido del puerto 8081 y la url del api con el controller y método que hicimos. En este caso la IP es 192.168.100.17:8081/api/suma?a=2&b=3 y nos debe responder con el resultado.



Como el servicio funcionó correctamente procedemos a parar el docker con los comandos:

\$ docker ps

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS            PORTS              NAMES
3bea0b022c30       gustavoarellano/jdk18  "java -jar /codigo..." 7 minutes ago
Up 7 minutes       0.0.0.0:8081->8080/tcp  determined_gates
gustavo@ubuntu:~/curso-dgp/servicio$
```

Para obtener el ID del docker en ejecución el cual es 3bea0b022c30, este cambia cada vez que se ejecuta un docker, y después ejecutamos el comando:  
\$ docker stop 3bea0b022c30



### 3.- Crear imagen Docker del microservicio con Dockerfile y subirlo a Docker Hub

Primero debemos autenticarnos en nuestra cuenta de Docker Hub, se nos solicitará el usuario y contraseña correspondientes, ejecutar el comando siguiente:

\$ docker login

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: dgpecurso12
Password:
Login Succeeded
gustavo@ubuntu:~/curso-dgp/servicio$
```

Después debemos generar un archivo Dockerfile con el comando:

\$ nano Dockerfile

```
gustavo@ubuntu:~/curso-dgp/servicio$ nano Dockerfile
```

Dentro del archivo se debe agregar las siguientes líneas y debemos guardarlo:

FROM gustavoarellano/jdk18

COPY ./target/sample-1.0-SNAPSHOT-fat.jar /javabin/sample-1.0-SNAPSHOT-fat.jar

CMD java -jar /javabin/sample-1.0-SNAPSHOT-fat.jar

GNU nano 2.5.3	File: Dockerfile	Modified
<pre>FROM gustavoarellano/jdk18 COPY ./target/sample-1.0-SNAPSHOT-fat.jar /javabin/sample-1.0-SNAPSHOT-fat.jar CMD java -jar /javabin/sample-1.0-SNAPSHOT-fat.jar</pre>		

Ejecutaremos el siguiente comando para compilar y crear la imagen del docker:

\$ docker build -t microservicio .

```
gustavo@ubuntu:~/curso-dgp/servicio$ nano Dockerfile
gustavo@ubuntu:~/curso-dgp/servicio$ docker build -t microservicio .
Sending build context to Docker daemon 8.261 MB
Step 1/3 : FROM gustavoarellano/jdk18
--> caea849e9be4
Step 2/3 : COPY ./target/sample-1.0-SNAPSHOT-fat.jar /javabin/sample-1.0-SNAPSHOT-fat.jar
--> 8ace8d78ad43
Removing intermediate container 01d58de6855a
Step 3/3 : CMD java -jar /javabin/sample-1.0-SNAPSHOT-fat.jar
--> Running in ad829967d29f
--> 5bac4ad80d3a
Removing intermediate container ad829967d29f
Successfully built 5bac4ad80d3a
gustavo@ubuntu:~/curso-dgp/servicio$
```

Con el comando “docker images” obtendremos el ID de la imagen mismo que se informó al finalizar la compilación.

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
microservicio	latest	5bac4ad80d3a	About a minute ago
kebbblar/pba-curso-unam	latest	e7cb7c355e69	3 days ago
kebbblar/jdk18-utf8-debug-maven	latest	25b228c6a928	4 months ago

Ejecutamos la imagen del docker creada con el comando:

\$ docker run -d -p 8081:8080 microservicio

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker run -d -p 8081:8080 microservicio
185188e83dfcd60bbad2e9b870745c4470de26675a1aee359f6d017bf816d27a
gustavo@ubuntu:~/curso-dgp/servicio$
```

Verificamos que esté corriendo el docker con el comando siguiente y obtenemos el ID:

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
185188e83dfc	microservicio	"/bin/sh -c 'java ...'"	About a minute ago	Up

Con el ID obtenido en el paso anterior ejecutamos el comando y debe responder con una clave sha256:

\$ docker commit 185188e83dfc dgpecurso12/microservicio

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker commit 185188e83dfc dgpecurso12/microservicio
sha256:77eedd9b70c355bbd0eb4b626d92077f4ddb355281761e90ddd787b6c34cc65d
gustavo@ubuntu:~/curso-dgp/servicio$
```

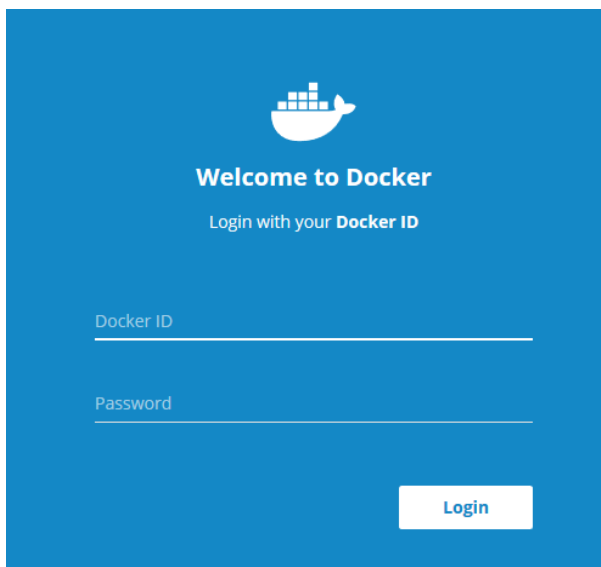
Finalmente ejecutamos el comando para subirlo a Docker Hub:

\$ docker push dgpecurso12/microservicio

```
gustavo@ubuntu:~/curso-dgp/servicio$ docker push dgpecurso12/microservicio
The push refers to a repository [docker.io/dgpecurso12/microservicio]
3255b498078b: Pushed
44bd3c9b5ffc: Pushed
0dd2d2815662: Mounted from kebbblar/pba-curso-unam
5f70bf18a086: Mounted from kebbblar/pba-curso-unam
918dbf1cf3de: Mounted from kebbblar/pba-curso-unam
9e1fe90ee292: Mounted from kebbblar/pba-curso-unam
d97fd2c5d8e1: Mounted from kebbblar/pba-curso-unam
c1cc34424286: Mounted from kebbblar/pba-curso-unam
latest: digest: sha256:a99d72707773e7376d1e630d8eb3023dae51dba723541d75d7255e9ad65a6f0a
size: 1988
gustavo@ubuntu:~/curso-dgp/servicio$
```

Para comprobar que este sí se haya subido correctamente ingresamos a Docker Hub con nuestro usuario y password.

Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)



Y debemos observar el docker creado.

Search







Dashboard Explore Organizations Create dgpecurso12

dgpecurso12 Repositories Stars Contributed Private Repositories: Using 0 of 1 Get more

### Repositories

Create Repository

Type to filter repositories by name

 dgpecurso12/operacion public	0 STARS	10 PULLS	 DETAILS
 dgpecurso12/goose-1 public	0 STARS	1 PULLS	 DETAILS
 dgpecurso12/microservicio public	0 STARS	1 PULLS	 DETAILS

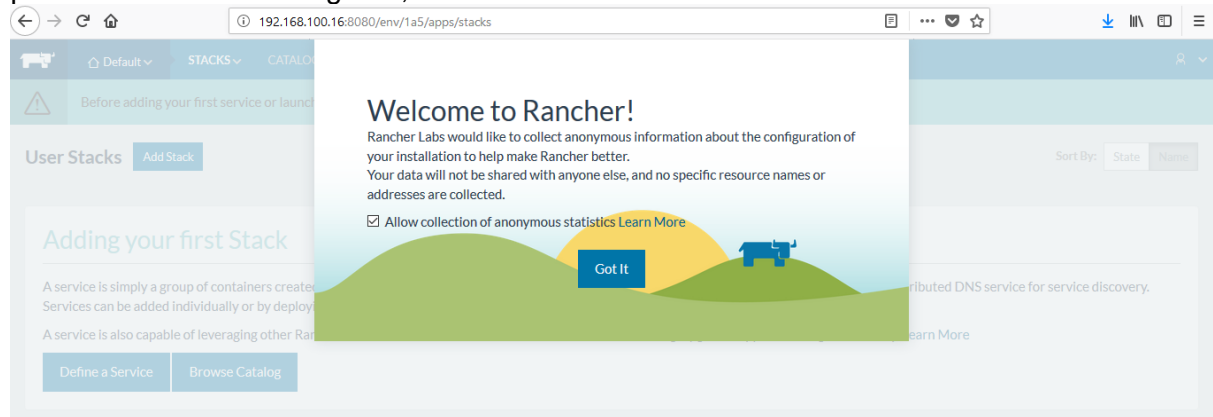
## 4.- Instalar y configurar Rancher

Para instalar Rancher se tomará uno que ya está en un docker, con el comando se procederá a bajarlo e iniciarlo:

```
$ docker run -d -p 8080:8080 kebbblar/rancher
```

```
gustavo@ubuntu:~$ docker run -d -p 8080:8080 kebbblar/rancher
Unable to find image 'kebbblar/rancher:latest' locally
latest: Pulling from kebbblar/rancher
6599cadaf950: Pull complete
23eda618d451: Pull complete
f0be3084efe9: Pull complete
52de432f084b: Pull complete
a3ed95caeb02: Pull complete
e75cd91a1dc5: Pull complete
997f1b48f59f: Pull complete
313c28fb4e37: Pull complete
2a0730d1275c: Pull complete
8848fbeb2c8: Pull complete
906504ea9ea6: Pull complete
9329940f8e65: Pull complete
e849debd7945: Pull complete
4883bd135dd2: Pull complete
605c6a0fe940: Pull complete
274bc004c933: Pull complete
a6cb25e8d1a2: Pull complete
3ded9d4c8c2b: Pull complete
a571d2f40012: Pull complete
9c64f335a665: Pull complete
Digest: sha256:434f5595cd28c344ad9601f0f9dfd92d8bcadc6d7a10da6cfd97bc83f85e44c
Status: Downloaded newer image for kebbblar/rancher:latest
02aeaa4776ea66554d721e0e79cafc3c957e645f72a73875e16b0e0b698e055c
gustavo@ubuntu:~$
```

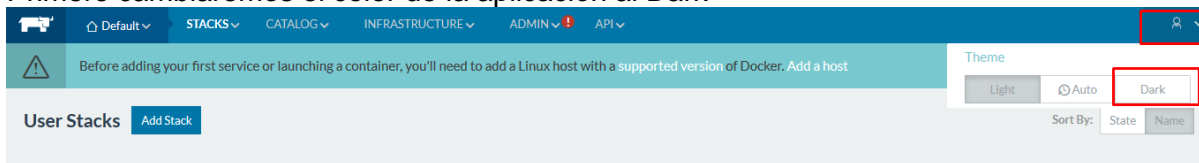
Para visualizar el Rancher ejecutándose ingresar a la IP del equipo donde se instaló en el puerto 8080 en un navegador, en este caso 192.168.100.16:8080.



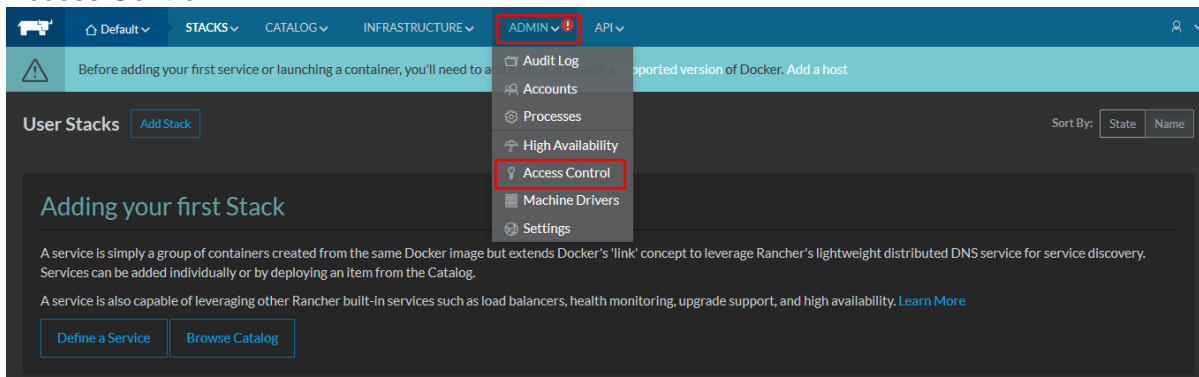
Esta pantalla indica que ha iniciado el Rancher.

Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

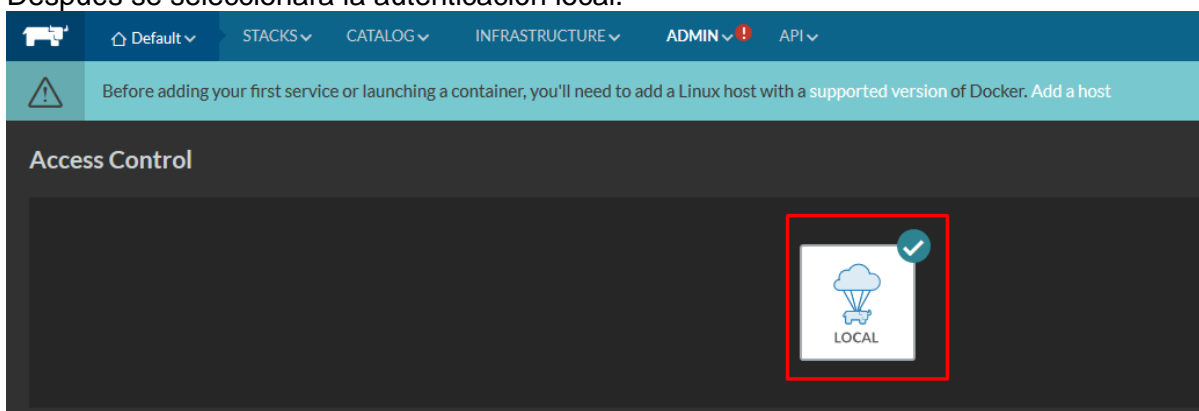
Primero cambiaremos el color de la aplicación al Dark



Primero debemos configurar el acceso, para esto se ingresará al menú ADMIN y submenú Access Control



Después se seleccionara la autenticación local.



Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

Se deberán de registrar los datos de la autenticación y oprimir el botón “Enabel Local Auth”.

Local Authentication is **not configured**

Rancher can be configured to restrict access to a set of accounts defined in the Rancher database. This is not currently set up, so anybody that reach this page (or the API) has full control over the system.

### 1. Setup an Admin user

This user will become the admin that has full control over Rancher.

Login Username*	Full Name
dgpecurso12	dgpecurso12
Password*	Confirm Password*
....	....

### 2. Enable Access Control

Click to enable access control and log in.

Enable Local Auth

Inmediatamente cerrar sesión.

192.168.100.16:8080/admin/access/local

Before adding your first service or launching a container, you'll need to add a Linux host with a supported version of Docker. Add a host

Access Control

Local Authentication is **enabled**

Rancher is configured to allow access to accounts in its local database. [Manage Accounts](#)

dgpecurso12  
dgpecurso12

Log Out

Your Account

Change Password

Theme

Light Auto Dark

E ingresar de nuevo al Rancher.

Howdy!  
Welcome to Rancher

English

Username

dgpecurso12

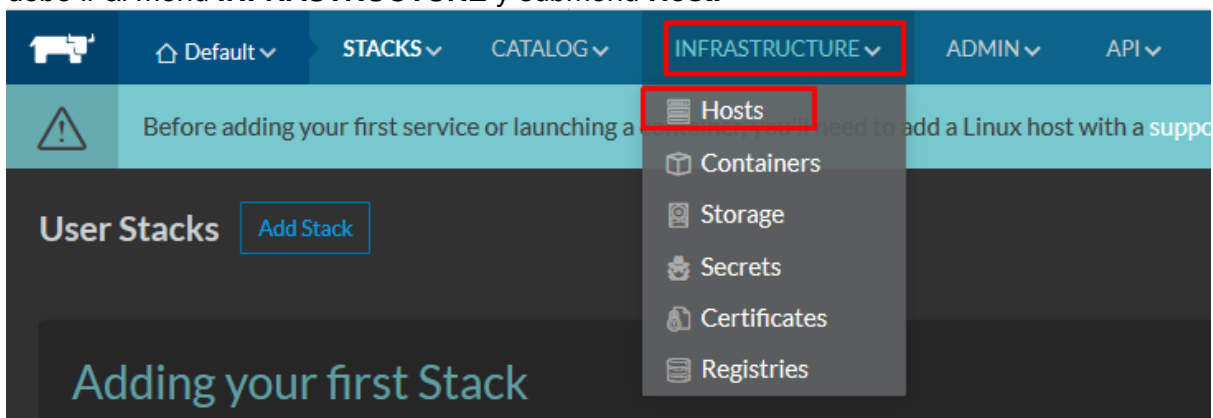
Password

....

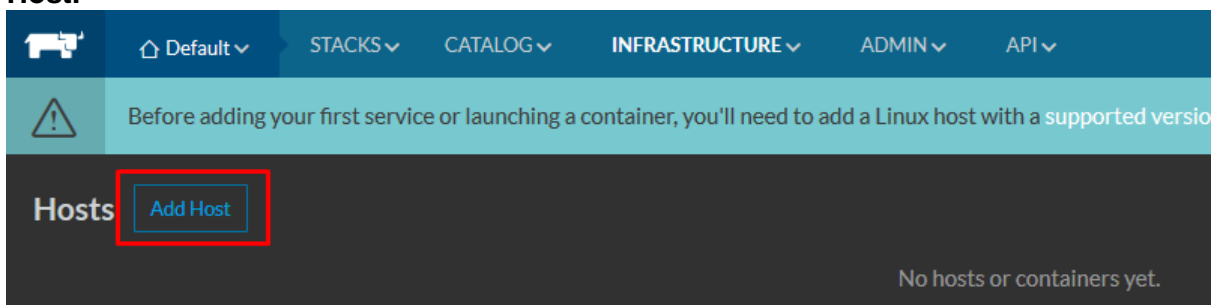
Log In

## 5.- Generar la llave

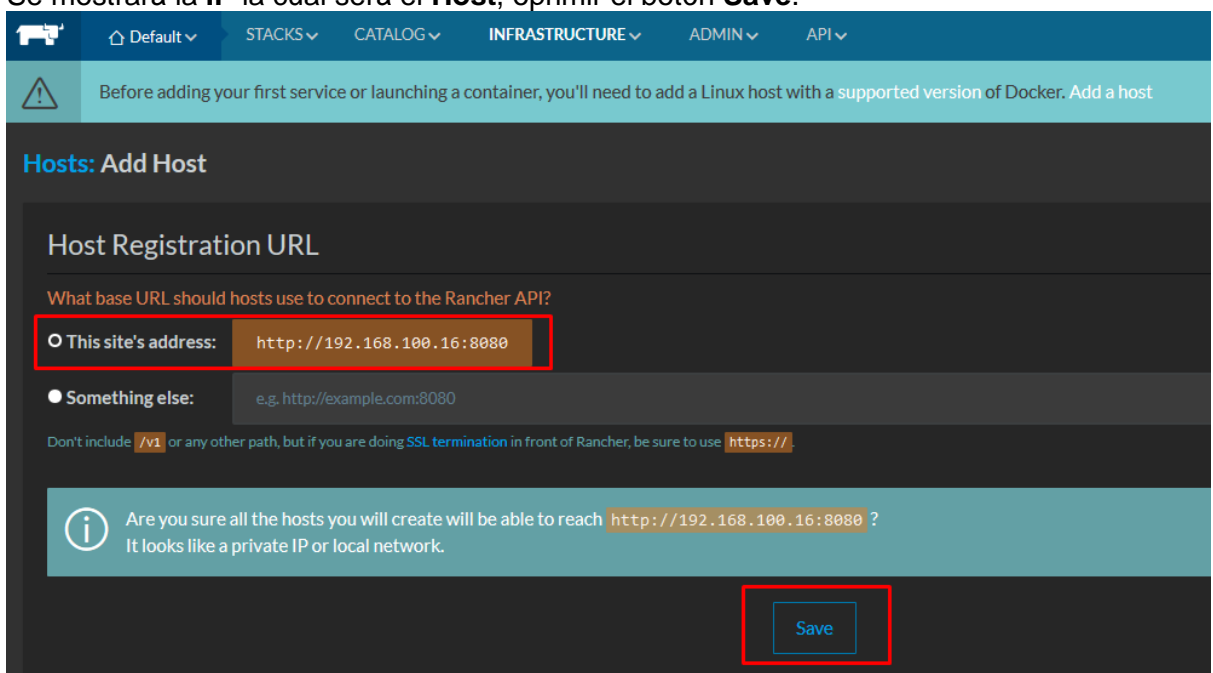
Para generar una llave para poder conectar más equipos a administrar en el rancher, se debe ir al menú **INFRASTRUCTURE** y submenú **Host**.



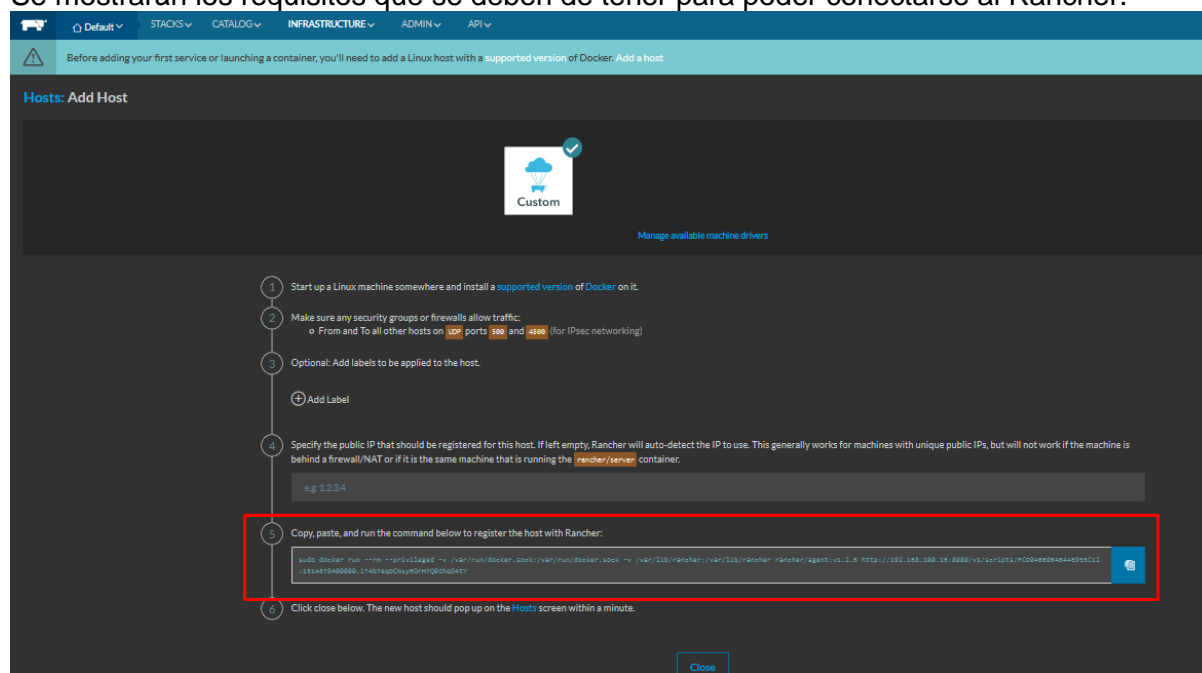
Ahí oprimir el botón **Add Host**.



Se mostrará la **IP** la cual será el **Host**, oprimir el botón **Save**.



Se mostrarán los requisitos que se deben de tener para poder conectarse al Rancher.



También se ha generado un url con una llave la cual se debe de ejecutar en los equipos clientes. La cual es un comando para instalar docker con rancher y un agente el cual es el que va estar monitoreando a los clientes.

```
sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.6  
http://192.168.100.16:8080/v1/scripts/FCD04BE0B4B446955C12:1514678400000:iT4b7eqDCNzyROrH7Q0lhqO4tY
```

Esta llave varía de equipo a equipo, nunca es la misma.



## 6.- Registrar equipo

La llave que se creó en el apartado anterior se debe de copiar y ejecutar en otro equipo distinto a donde se encuentra Rancher.

Se ejecutará en el equipo que tiene la IP 192.168.100.17

```
gustavo@ubuntu:~/curso-dgp/servicio$ ifconfig
docker0    Link encap:Ethernet  HWaddr 02:42:18:e3:e2:62
            inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
            inet6 addr: fe80::42:18ff:fee3:e262/64 Scope:Link
            UP BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:199 errors:0 dropped:0 overruns:0 frame:0
            TX packets:260 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:17104 (17.1 KB)  TX bytes:280042 (280.0 KB)

ens33      Link encap:Ethernet  HWaddr 00:0c:29:49:be:c2
            inet addr:192.168.100.17  Bcast:192.168.100.255  Mask:255.255.255.0
            inet6 addr: fe80::20c:29ff:fe49:bec2/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:1160160 errors:0 dropped:0 overruns:0 frame:0
            TX packets:89828 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:1681918468 (1.6 GB)  TX bytes:13804822 (13.8 MB)

lo         Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:160 errors:0 dropped:0 overruns:0 frame:0
            TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)

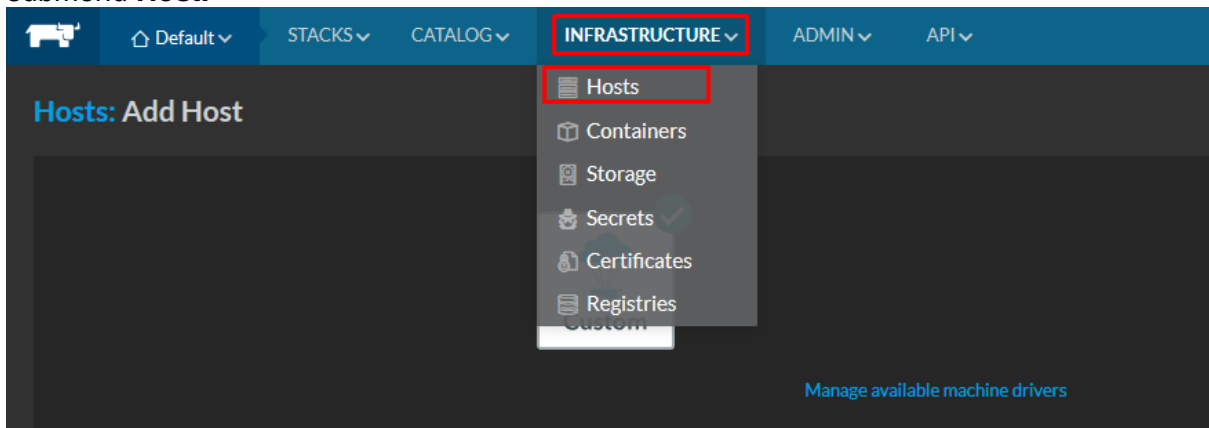
gustavo@ubuntu:~/curso-dgp/servicio$
```

Se ejecuta la llave que se copió y debe de registrarse el equipo en el **Host**.

```
gustavo@ubuntu:~/curso-dgp/servicio$ sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.6 http://192.168.100.16:8080/v1/scripts/FCD04BE0B4B446955C12:1514678400000:iT4b7eqDCNzyROrH7Q0IhqO4tY

INFO: Running Agent Registration Process, CATTLE_URL=http://192.168.100.16:8080/v1
INFO: Attempting to connect to: http://192.168.100.16:8080/v1
INFO: http://192.168.100.16:8080/v1 is accessible
INFO: Inspecting host capabilities
INFO: Boot2Docker: false
INFO: Host writable: true
INFO: Token: xxxxxxxx
INFO: Running registration
INFO: Printing Environment
INFO: ENV: CATTLE_ACCESS_KEY=D88BA82562720A7A884B
INFO: ENV: CATTLE_HOME=/var/lib/cattle
INFO: ENV: CATTLE_REGISTRATION_ACCESS_KEY=registrationToken
INFO: ENV: CATTLE_REGISTRATION_SECRET_KEY=xxxxxxx
INFO: ENV: CATTLE_SECRET_KEY=xxxxxxx
INFO: ENV: CATTLE_URL=http://192.168.100.16:8080/v1
INFO: ENV: DETECTED_CATTLE_AGENT_IP=192.168.100.17
INFO: ENV: RANCHER_AGENT_IMAGE=rancher/agent:v1.2.6
INFO: Launched Rancher Agent: 2f38675ddcdd330379044107d60b6d54b6bf3f0f9da26231322f6fa2b9a8ea0a
gustavo@ubuntu:~/curso-dgp/servicio$
```

Para visualizar que se haya registrado el equipo iremos al menú **INFRASTRUCTURE** y submenú **Host**.



Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

Se deberá de visualizar el equipo con la **IP** previamente registrada.

The screenshot displays the OpenStack Horizon 'Hosts' page. The navigation bar at the top includes 'Default', 'STACKS', 'CATALOG', 'INFRASTRUCTURE', 'ADMIN', and 'API'. The 'Hosts' section shows a list of hosts, with the first host, 'ubuntu', highlighted. The host's IP address, 192.168.100.17, is circled in red. The host is marked as 'ACTIVE' and has a status icon showing two bars. Below the host name, the IP address is listed, followed by the cloud-init version (17.03.2-ce) and the Ubuntu version (16.04.4 LTS (4.4.0)). The host's hardware specifications are also shown: 2.89 GHz, 1.94 GiB, and 18.6 GiB. The host is associated with three stacks: 'healthcheck', 'ipsec', and 'network-services'. Each stack has a list of components, such as 'healthcheck-1', 'ipsec-1', and 'network-manager-1', with their respective IP addresses and status icons.

Stack	Component	IP Address	Status
healthcheck	..healthcheck-1	10.42.96.119	Active
	..ipsec-1	10.42.60.61	Active
ipsec	..cni-driver-1	None	None
	..ipsec-1	10.42.60.61	Active
network-services	..network-manager-1	None	None
	..metadata-1	172.17.0.2	Active

Se procederá a ejecutar la llave de registro en otro equipo con la IP 192.168.100.18

```
gustavo@ubuntu:~$ ifconfig
docker0    Link encap:Ethernet  HWaddr 02:42:80:7f:3c:16
            inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
            UP BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ens33      Link encap:Ethernet  HWaddr 00:0c:29:e8:49:a8
            inet addr:192.168.100.18  Bcast:192.168.100.255  Mask:255.255.255.0
            inet6 addr: fe80::20c:29ff:fee8:49a8/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:91 errors:0 dropped:0 overruns:0 frame:0
            TX packets:74 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:11018 (11.0 KB)  TX bytes:10303 (10.3 KB)

lo         Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:160 errors:0 dropped:0 overruns:0 frame:0
            TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)

gustavo@ubuntu:~$
```

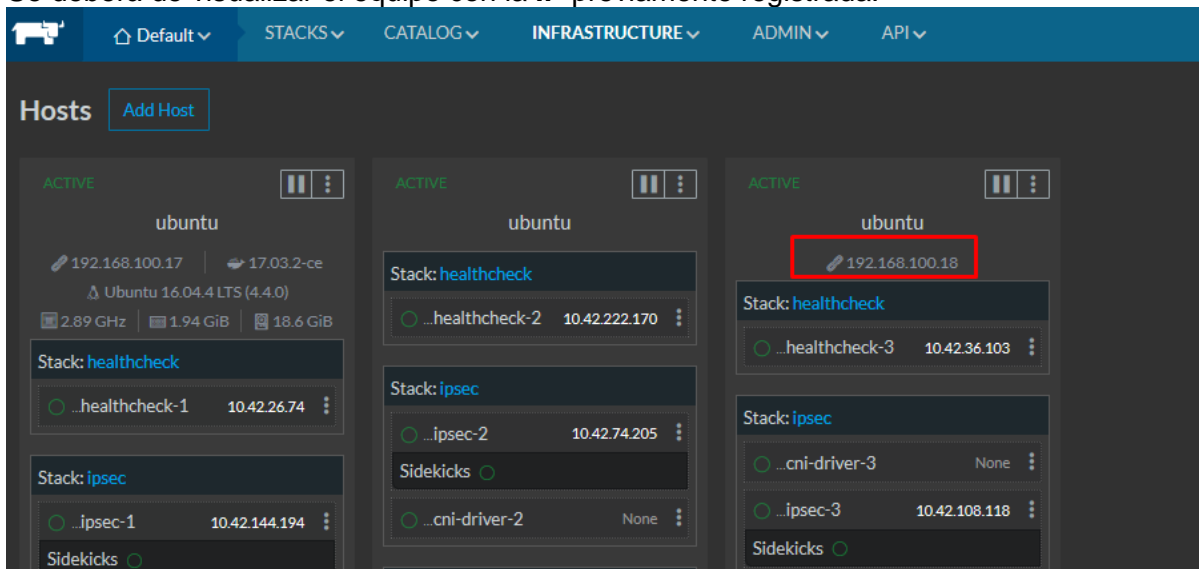
Se copia la llave nuevamente en el nuevo cliente.

```
gustavo@ubuntu:~$ sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run
/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.6 http://192.16
8.100.16:8080/v1/scripts/FCD04BE0B4B446955C12:1514678400000:iT4b7eqDCNzyR0rH7Q0IhqO4
tY

INFO: Running Agent Registration Process, CATTLE_URL=http://192.168.100.16:8080/v1
INFO: Attempting to connect to: http://192.168.100.16:8080/v1
INFO: http://192.168.100.16:8080/v1 is accessible
INFO: Inspecting host capabilities
INFO: Boot2Docker: false
INFO: Host writable: true
INFO: Token: xxxxxxxx
INFO: Running registration
INFO: Printing Environment
INFO: ENV: CATTLE_ACCESS_KEY=D88BA82562720A7A884B
INFO: ENV: CATTLE_HOME=/var/lib/cattle
INFO: ENV: CATTLE_REGISTRATION_ACCESS_KEY=registrationToken
INFO: ENV: CATTLE_REGISTRATION_SECRET_KEY=xxxxxxx
INFO: ENV: CATTLE_SECRET_KEY=xxxxxxx
INFO: ENV: CATTLE_URL=http://192.168.100.16:8080/v1
INFO: ENV: DETECTED_CATTLE_AGENT_IP=192.168.100.18
INFO: ENV: RANCHER_AGENT_IMAGE=rancher/agent:v1.2.6
INFO: Launched Rancher Agent: 2eb51f7a45c56c10eadd742190bdec5f720c9bbc8f237b3143ac3d
211ba0cf13
gustavo@ubuntu:~$
```

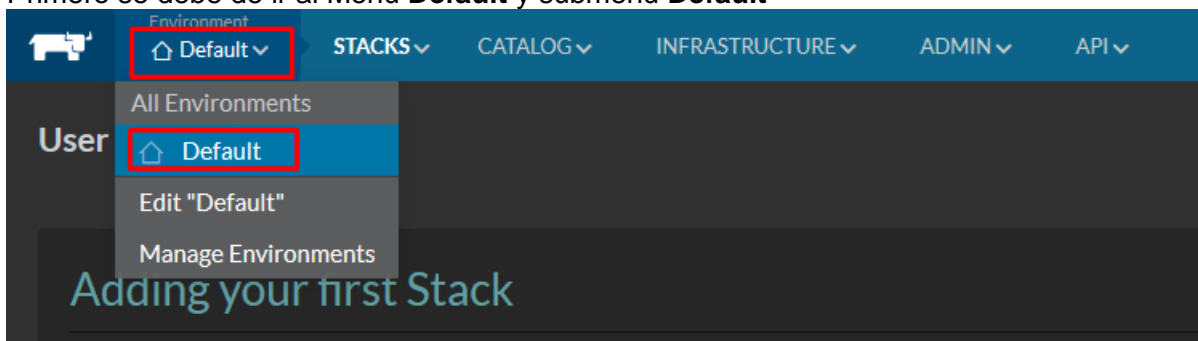
Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

Se deberá de visualizar el equipo con la **IP** previamente registrada.

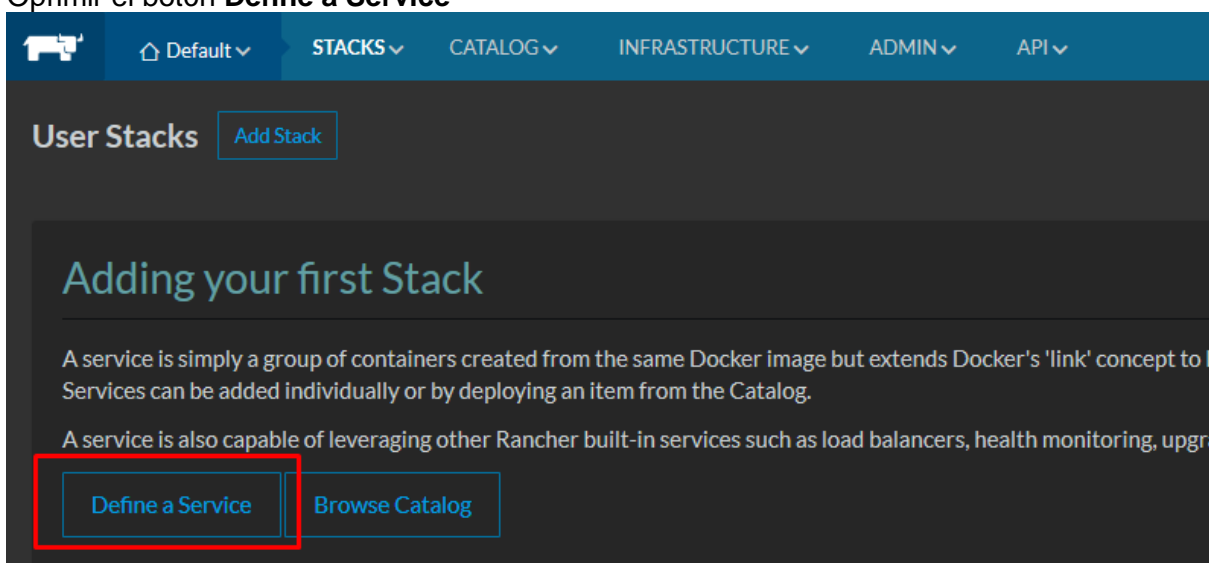


## 7.- Crear un servicio

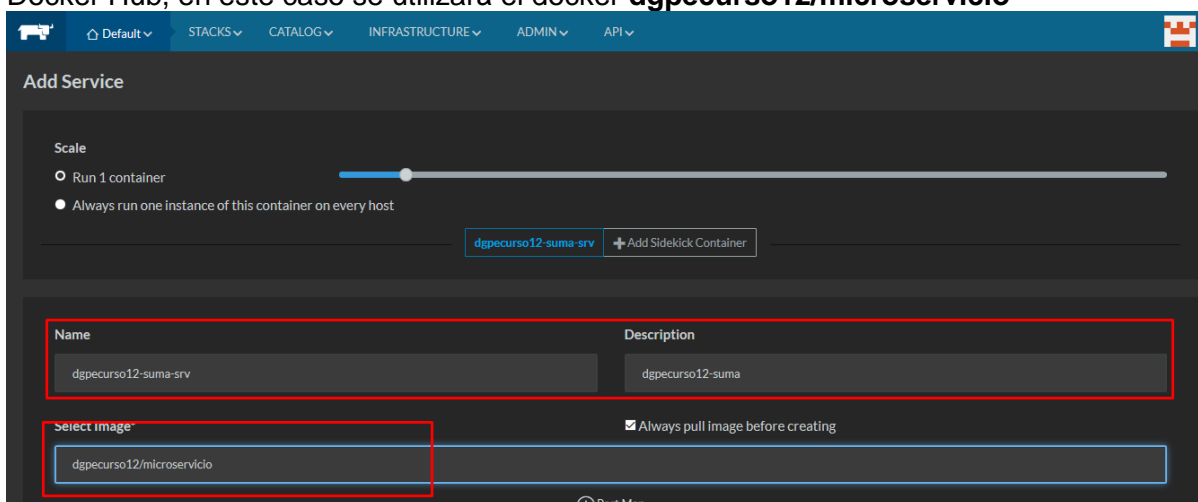
Primero se debe de ir al Menú **Default** y submenú **Default**



Oprimir el botón **Define a Service**



Ingresa el nombre del servicio y la descripción, así como la imagen que se bajara de Docker Hub, en este caso se utilizará el docker **dgpecurso12/microservicio**



Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

En la parte inferior en la pestaña **command**, en donde dice **Console** seleccionar **None**

The screenshot shows the 'Command' tab with various configuration fields. Under the 'Console' section, the 'None' radio button is selected, indicating that no console access is required for the container.

En la pestaña **Security/Host** en donde dice **Log Driver** seleccionar **json-file** y después oprimir el botón **Create**.

The screenshot shows the 'Security/Host' tab. The 'Log Driver' dropdown menu is open, and 'json-file' is selected. The 'Create' button at the bottom right is highlighted, indicating the next step in the configuration process.

Se observará que empezará a activarse el servicio.

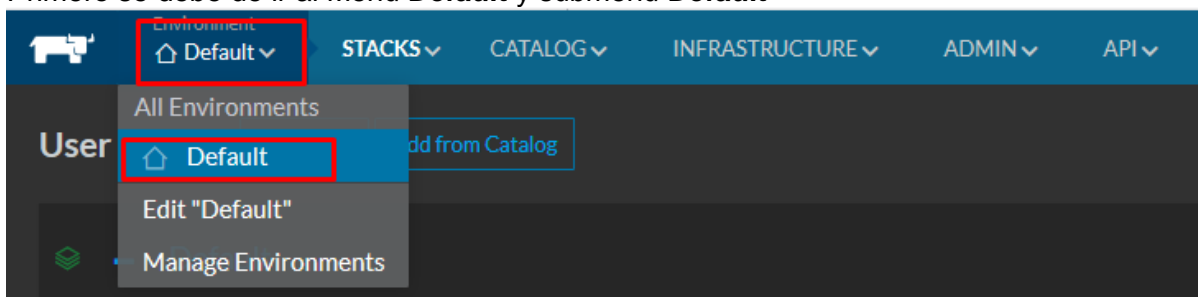
The screenshot shows the 'Stack' view with a table of services. The service 'dgpecurso12-suma-srv' is in the 'Activating' state, indicated by a blue progress bar and the text '(In Progress)'.

Después de unos segundos o minutos quedará activo el servicio.

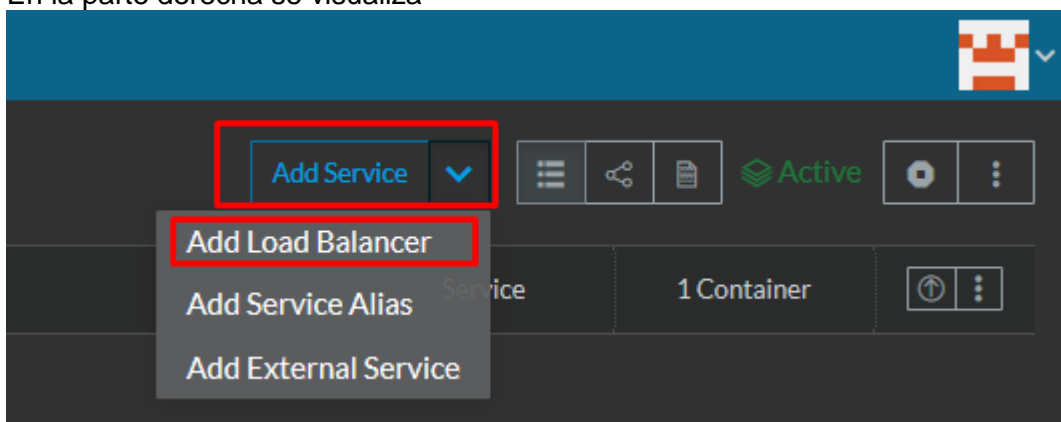
The screenshot shows the 'Stack' view with the same table of services. The service 'dgpecurso12-suma-srv' is now in the 'Active' state, indicated by a green checkmark and the text 'Active'.

## 8.- Crear un balanceador

Primero se debe de ir al Menú **Default** y submenú **Default**



En la parte derecha se visualiza



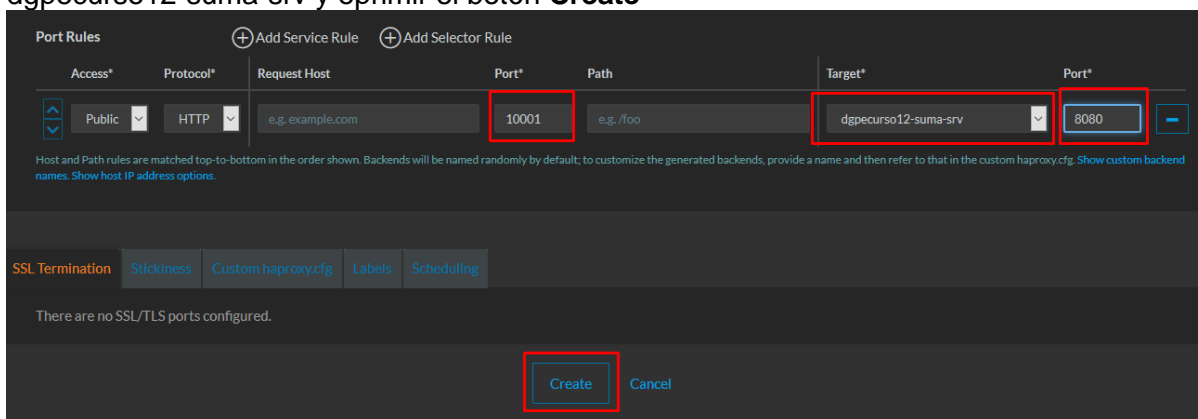
Ingresar el nombre y descripción del balanceador

A screenshot of the 'Add Load Balancer' form in the Rancher UI. The form has a title 'Add Load Balancer' with a help icon. Below the title is a 'Scale' section with two radio buttons: 'Run 1 container' (selected) and 'Always run one instance of this container on every host'. Below this is a 'Name' field with the value 'lb-dgpecurso12' and a 'Description' field with the value 'lb-dgpecurso12'. Both fields are highlighted with red boxes.



Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

Ingresa el puerto del balanceador y selecciona el servicio que se creó anteriormente dgpecurso12-suma-srv y oprimir el botón **Create**



Port Rules

+ Add Service Rule + Add Selector Rule

Access*	Protocol*	Request Host	Port*	Path	Target*	Port*
Public	HTTP	e.g. example.com	10001	e.g. /foo	dgpecurso12-suma-srv	8080

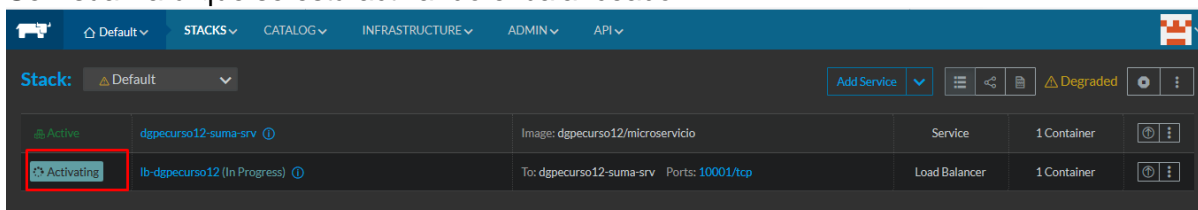
Host and Path rules are matched top-to-bottom in the order shown. Backends will be named randomly by default; to customize the generated backends, provide a name and then refer to that in the custom haproxy.cfg. Show custom backend names. Show host IP address options.

SSL Termination Stickiness Custom haproxy.cfg Labels Scheduling

There are no SSL/TLS ports configured.

Create Cancel

Se visualizará que se está activando el balanceador.



Stack:	Default	Add Service	Degraded	Active
Active	dgpecurso12-suma-srv	Image: dgpecurso12/microservicio	Service	1 Container
Activating	lb-dgpecurso12 (In Progress)	To: dgpecurso12-suma-srv Ports: 10001/tcp	Load Balancer	1 Container

Después se visualizará que quedo activo el balanceador.

Default

STACKS

CATALOG

INFRASTRUCTURE

ADMIN

API

Stack:

Default

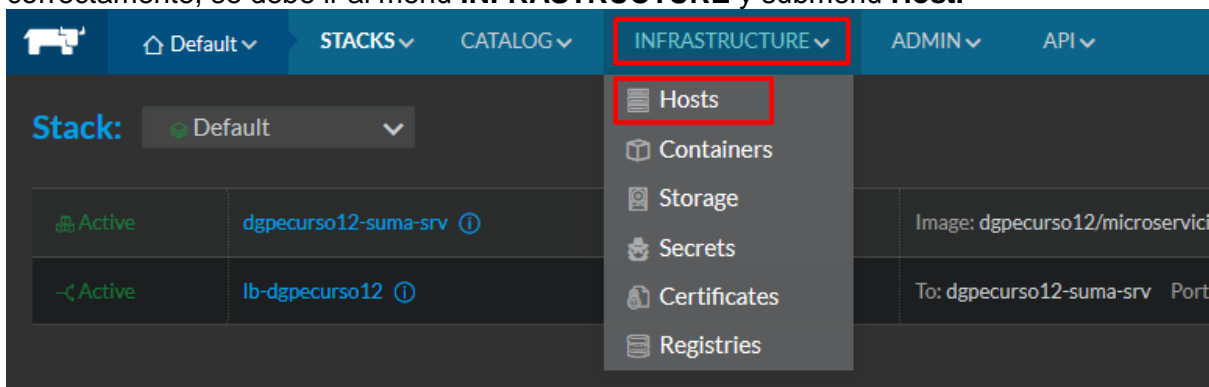
Add Service

Active

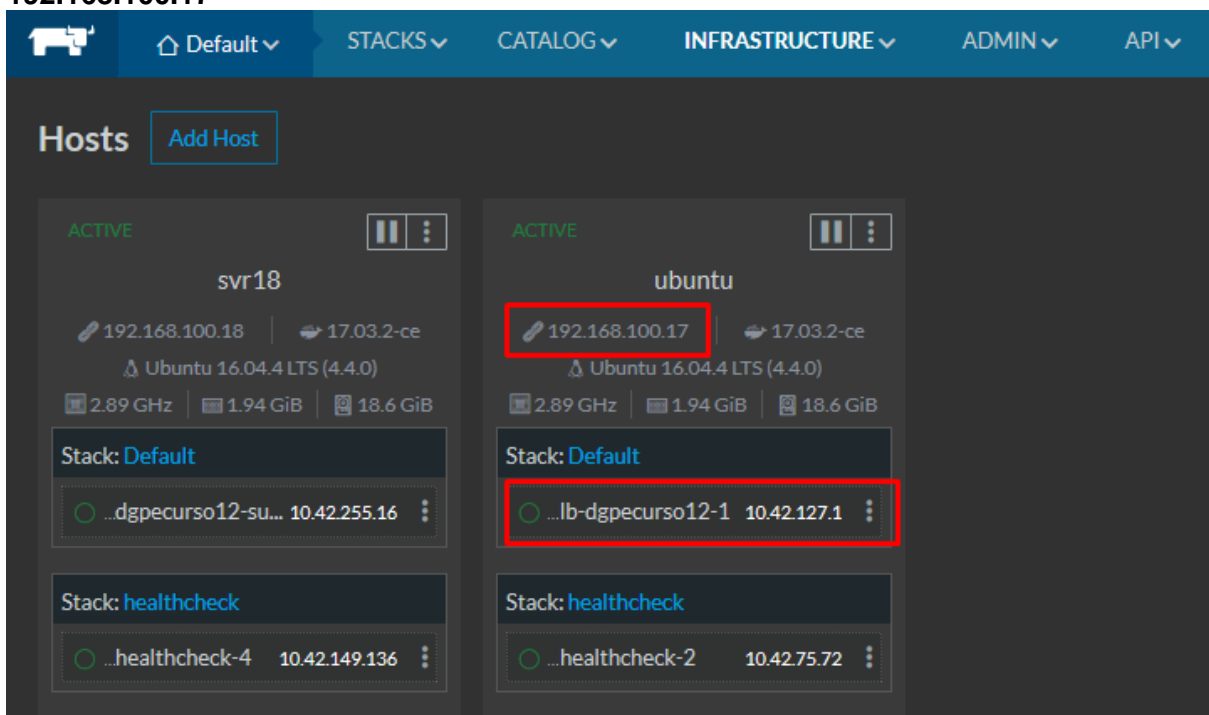
Active	dgpecurso12-suma-srv	Image: dgpecurso12/microservicio	Service	1 Container	<div><div></div><div></div></div>
Active	lb-dgpecurso12	To: dgpecurso12-suma-srv Ports: 10001/tcp	Load Balancer	1 Container	<div><div></div><div></div></div>

## 9.- Consulta en el Navegador

Para visualizar en el navegador que se ha generado el balanceador y servicio correctamente, se debe ir al menú **INFRASTRUCTURE** y submenú **Host**.

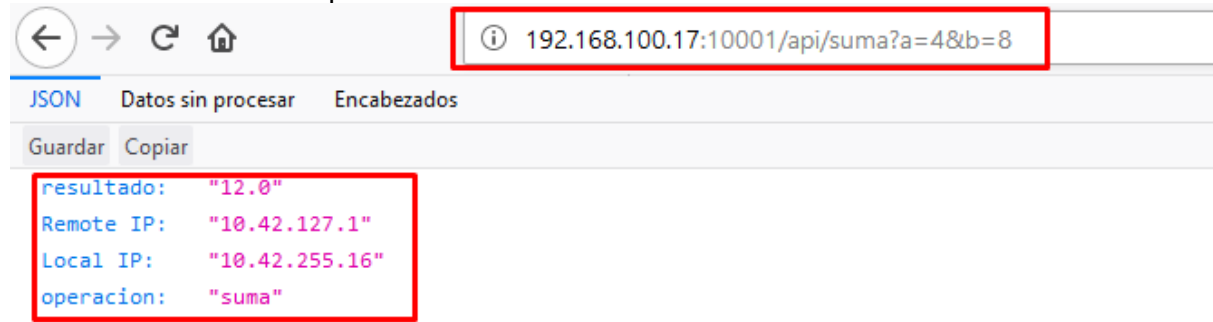


En esta pantalla se visualiza que el balanceador quedo en la IP **192.168.100.17**



Para visualizar el servicio en el navegador se debe ingresar la IP y el puerto del balanceador, así como los datos del servicio a consultar, en este caso

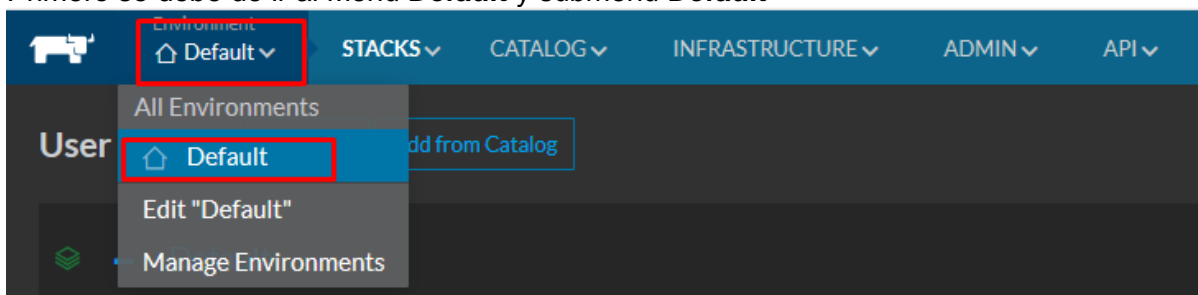
192.168.100.17:10001/api/suma?a=4&b=8



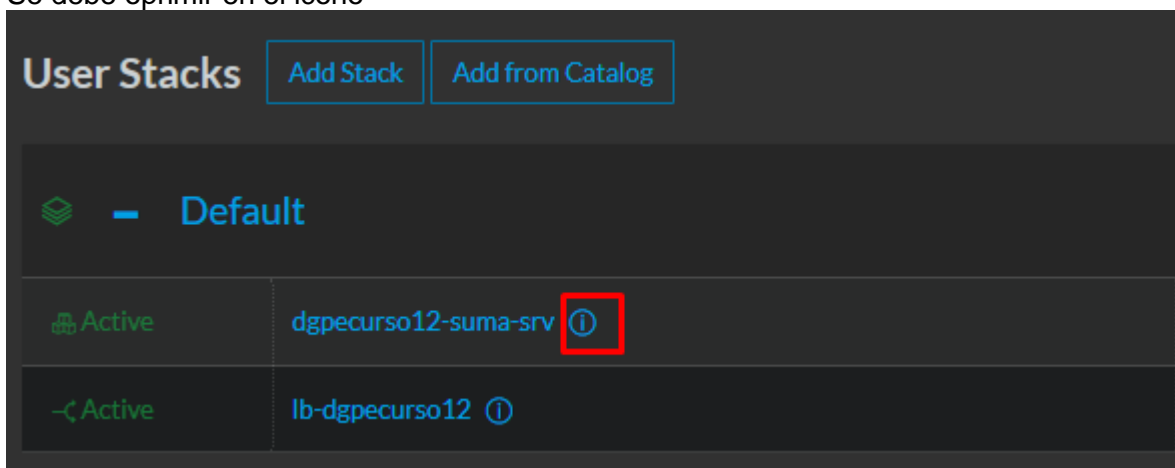
Las IP's mostradas corresponde a, **Local IP** a donde se encuentra el servicio que está respondiendo, y la **Remote IP** desde donde se realizó la petición, en este caso donde está el **Load Balancer**.

## 10.- Escalando el servicio

Primero se debe de ir al Menú **Default** y submenú **Default**



Se debe oprimir en el icono



Se visualiza que está escalado a 1, se oprime el ícono de más.

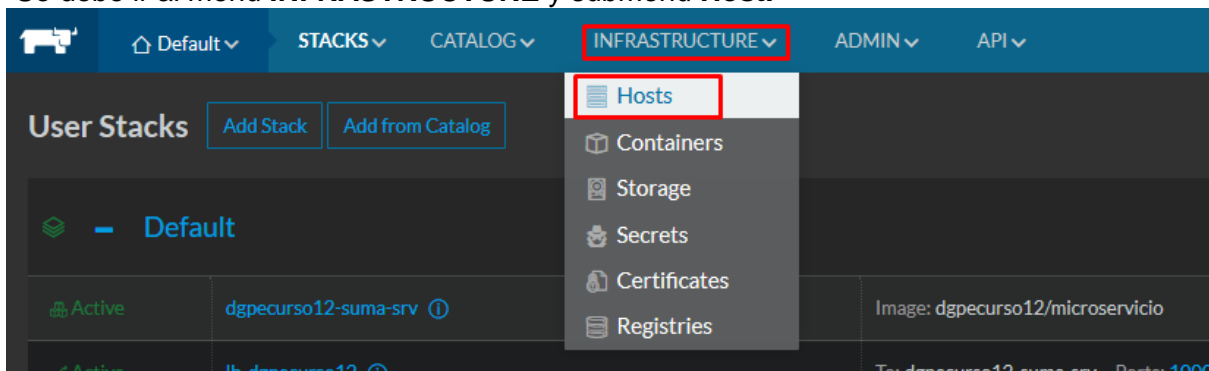


Autor: Rogelio De La Cruz Basilio (dgpecurso12, dgpe.curso.12@gmail.com)

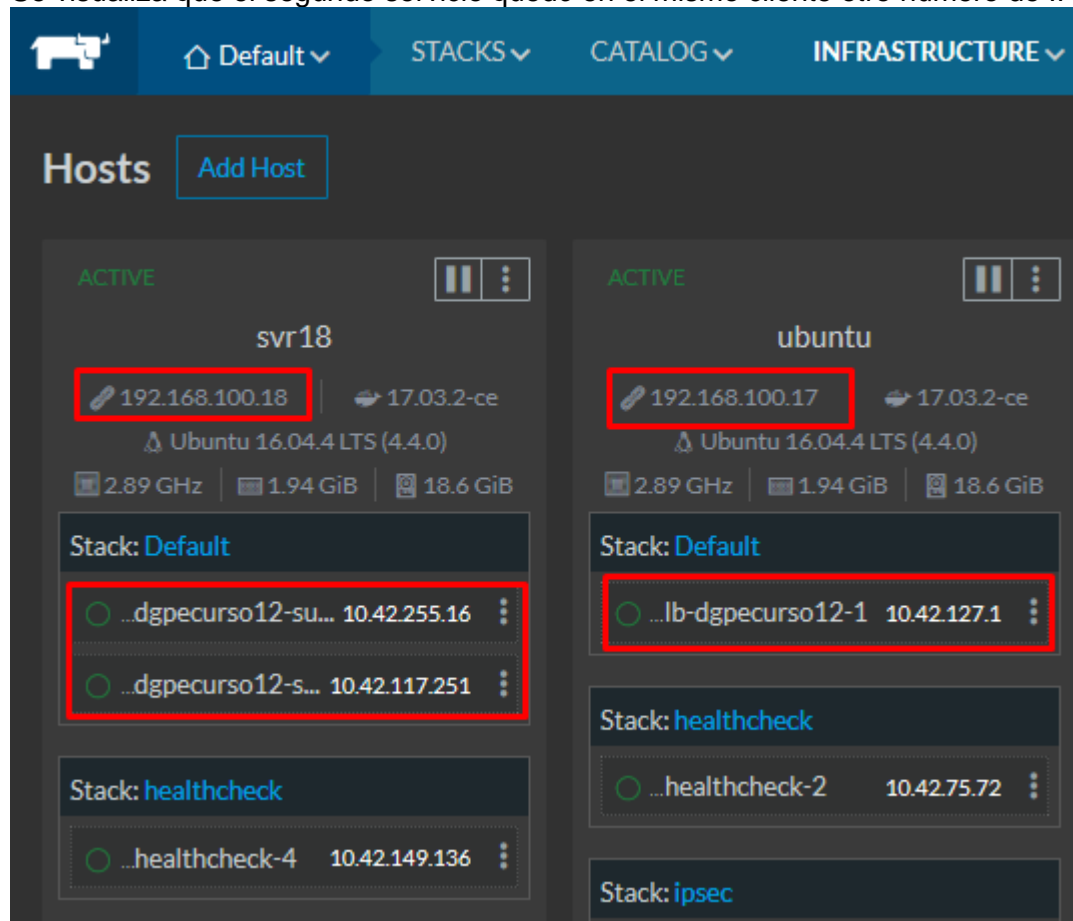
Después se observa que ya está escalado a 2



Se debe ir al menú **INFRASTRUCTURE** y submenú **Host**.

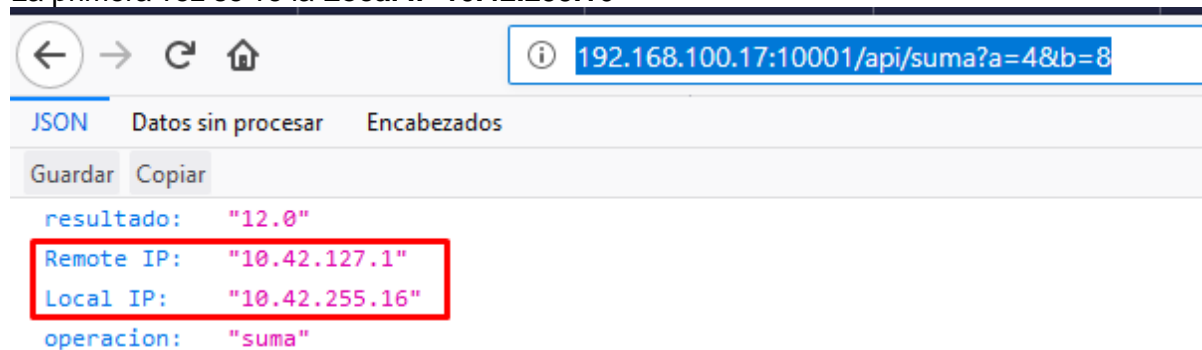


Se visualiza que el segundo servicio quedó en el mismo cliente otro número de IP interna.

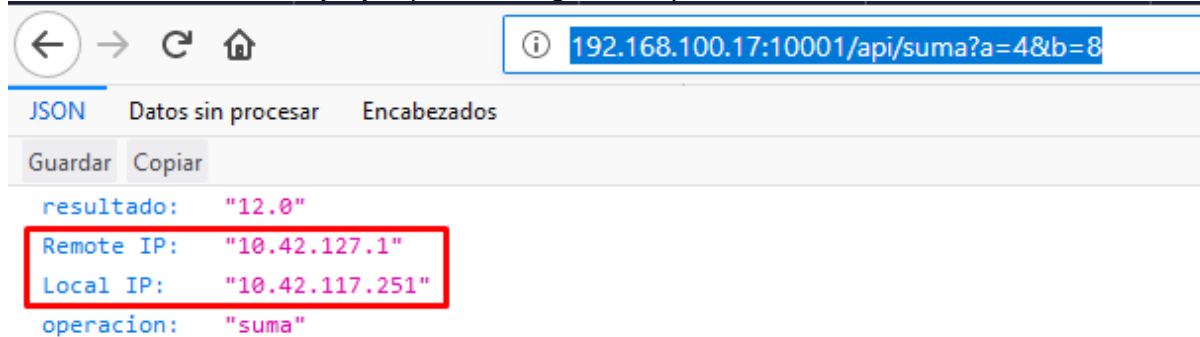


Al realizar la consulta en el navegador se debe ingresar la IP y el puerto del balanceador, así como los datos del servicio a consultar, en este caso  
192.168.100.17:10001/api/suma?a=4&b=8

La primera vez se ve la **Local IP 10.42.255.16**



La segunda vez se ve la **Local IP 10.42.117.251**, indicando que el balanceador está realizando bien su trabajo, ya que la configuración predeterminada es el **Round Robin**



The screenshot shows a web browser window with a REST client interface. The address bar displays the URL `192.168.100.17:10001/api/suma?a=4&b=8`. Below the address bar, there are tabs for `JSON`, `Datos sin procesar`, and `Encabezados`. The `JSON` tab is selected. Below the tabs, there are buttons for `Guardar` and `Copiar`. The JSON response is displayed as follows:

```
resultado: "12.0"
Remote IP: "10.42.127.1"
Local IP: "10.42.117.251"
operacion: "suma"
```

The `Remote IP` and `Local IP` fields are highlighted with a red box.