A thick dark blue vertical bar is positioned on the left side of the page. To its right, several thin, curved lines in shades of blue and grey sweep upwards and outwards from the bottom left corner.

14-06-2018

Tutorial

Implementación de Rancher

Nancy Victoria Muñoz González
DGPECURSO06

Contenido

Creación de proyecto para desarrollar verticales	2
Creación de imagen.....	4
Subir a repositorio de imágenes (Dockerhub)	5
Implementación de Rancher	6
Añadir hosts	9
Crear un servicio en el Rancher.....	11
Creación del balanceador.....	12
Escalar servicio	13
Resultados	14

Creación de proyecto para desarrollar verticales

Clonaremos un proyecto que contenga la base para poder crear verticales.

```
File Edit View Search Terminal Help
gustavo@gustavo-itam-01:~$ cd /home/gustavo/Documents/curso-dgp/exaMod02/
gustavo@gustavo-itam-01:~/Documents/curso-dgp/exaMod02$ 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.
```

```
MyController.java
15
16 public class MyController extends AbstractVerticle {
17     private static final Logger logger = Logger.getLogger(MyController.class);
18
19     public void start(Future<Void> fut) {
20         logger.info("Inicializando Vertical");
21         Router router = Router.router(vertx);
22         //router.route("/").handler(StaticHandler.create("assets")); // para in
23         // el directorio "upload-folder" será creado en la misma ubicación que e
24         router.route().handler(BodyHandler.create().setUploadsDirectory("upload-
25         router.get("/api/suma").handler(this::suma);
26
27         // Create the HTTP server and pass the "accept" method to
28         vertx.createHttpServer().requestHandler(router::accept).list
29         config().getInteger("http.port", 8080), result -> {
30             if (result.succeeded()) {
31                 fut.complete();
32             } else {
33                 fut.fail(result.cause());
34             }
35         });
36
37         logger.info("Vertical iniciada !!!");
38     }
39 }
```

Colocamos el nombre para consumido por medio de la URL nuestra vertical, utilizando método get

```
private void suma(RoutingContext routingContext) {
    HttpServletResponse response = routingContext.response();
    HttpServletRequest request = routingContext.request();
    int var1 = Integer.parseInt(request.getParam("var1"));
    int var2 = Integer.parseInt(request.getParam("var2"));
    int resultado = var1+var2;

    String jsonResponse = resSum(resultado, var1, var2,request);
    response.setStatus(200);
    putHeader("content-type", "application/json; charset=utf-8");
    end(jsonResponse);
}

private String resSum(int result, int var1, int var2,HttpServletRequest request) {
    Map<Object, Object> cal = new HashMap<>();
    cal.put("IP:", request.localAddress().host());
    cal.put("variable1", var1);
    cal.put("variable2", var2);
    cal.put("resultado", result);
    return Json.encodePretty(cal);
}
}
```

Nuestro método será capaz de obtener datos de la URL para su operación y convirtiendo el resultado en un JSON

```
gustavo@gustavo-ltam-01:~/Documents/curso-dgp/exaMod02$ docker run -p 6060:8080 -v /home/gustavo/Documents/curso-dgp/exaMod02/vertx-sample:/codExa kebbblar/jdk18-utf8-debug-maven mvn -f codExa package
```

Para empaquetar el proyecto utilizaremos un Docker que contenga Maven y java colocando como volumen la carpeta donde el proyecto

Para poder observar que funciona correctamente nos correremos el JAR, la ayuda de un Docker que contenga Java y lo vincularemos al puerto 8080

```
gustavo@gustavo-ltam-01:~/Documents/curso-dgp/exaMod02$ docker run -p 6060:8080 -v /home/gustavo/Documents/curso-dgp/exaMod02/vertx-sample:/codExa kebbblar/jdk18-utf8-debug-maven java -jar codExa/target/sample-1.0-SNAPSHOT-fat.jar
2018-06-13 21:28:47:572 INFO MyController - Inicializando Vertical
2018-06-13 21:28:48:180 INFO MyController - Vertical iniciada !!!
jun 13, 2018 9:28:48 PM io.vertx.core.Starter
INFORMACIÓN: Succeeded in deploying verticle
```

localhost:6060/api/suma?va X +

localhost:6060/api/suma?var1=1&var2=2

JSON Raw Data Headers

Save Copy Pretty Print

```
{
  "variable1" : 1,
  "IP:" : "172.17.0.2",
  "variable2" : 2,
  "resultado" : 3
}
```

Como podemos observar le estamos realizando una petición

Brindamos un resultado

Creación de imagen

Creamos un archivo llamado
"Dockerfile"

```
gustavo@gustavo-itam-01:~/Documents/curso-dgp/exaMod02$ nano Dockerfile
```

Este documento contendrá las siguientes líneas como
podemos ver tenemos que especificar donde se
encuentra el JAR ya que se copiará a la imagen y se
desplegará

```
GNU nano 2.8.6 File: Dockerfile
FROM gustavoarellano/jdk19
COPY vertx-sample/target/sample-1.0-SNAPSHOT-fat.jar /home
ENTRYPOINT java -jar /home/sample-1.0-SNAPSHOT-fat.jar
```

Ahora construiremos la imagen teclearemos la siguiente
línea especificando el nombre de la imagen

```
gustavo@gustavo-itam-01:~/Documents/curso-dgp/exaMod02$ docker build . -t dgpecurso06/examod02
Sending build context to Docker daemon 16.49MB
Step 1/3 : FROM kebbler/jdk18-utf8-debug-maven
--> 25b228c6a928
Step 2/3 : COPY vertx-sample/target/sample-1.0-SNAPSHOT-fat.jar /home
--> 7d55ef1656a4
Step 3/3 : ENTRYPOINT java -jar /home/sample-1.0-SNAPSHOT-fat.jar
--> Running in 1fef773f3ea4
Removing intermediate container 1fef773f3ea4
--> 1837ad1a3c48
Successfully built 1837ad1a3c48
Successfully tagged dgpecurso06/examod02:latest
```

Probaremos la imagen
vinculando el puerto 8080

```
gustavo@gustavo-itam-01:~/Documents/curso-dgp/exaMod02$ docker run -p 8080:8080 -it dgpecurso06/examod02 bash
2018-06-13 21:47:17:089 INFO MyController - Inicializando Vertical
2018-06-13 21:47:17:418 INFO MyController - Vertical iniciada !!!
jun 13, 2018 9:47:17 PM io.vertx.core.Starter
INFORMACIÓN: Succeeded in deploying verticle
```

localhost:8080/api/suma?var1=1&var2=2

JSON Raw Data Headers

Save Copy Pretty Print

```
{
  "variable1" : 1,
  "IP:" : "172.17.0.2",
  "variable2" : 2,
  "resultado" : 3
}
```

Como podemos
observar los
resultados del
navegador son lo que
esperábamos

Subir a repositorio de imágenes (Dockerhub)

Iniciaremos sesión en Dockerhub

```
gustavo@gustavo-ltam-01:~/Documents/curso-dgp/exaMod02$ 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 (dgpecurso06):
Password:
Login Succeeded
```

Subiremos esta imagen

```
gustavo@gustavo-ltam-01:~/Documents/curso-dgp/exaMod02$ docker push dgpecurso06/examod02
The push refers to repository [docker.io/dgpecurso06/examod02]
6f0c65b27ae1: Pushed
d382ac01567b: Mounted from gustavoarellano/jdk19
0dd2d2815662: Layer already exists
5f70bf18a086: Layer already exists
918dbf1cf3de: Layer already exists
9e1fe90ee292: Layer already exists
d97fd2c5d8e1: Layer already exists
c1cc34424286: Layer already exists
latest: digest: sha256:61fa798d9ef1b8d8e84fd817fa6360f15d32338c35dacefe771e861ba7f37f83 size: 1990
```

The screenshot shows the Docker Hub interface for a public repository named `dgpecurso06/examod02`. The repository was last pushed 2 minutes ago. The page includes tabs for Repo Info, Tags, Collaborators, Webhooks, and Settings. On the left, there are fields for a Short Description and a Full Description, both currently empty. On the right, the Docker Pull Command is displayed as `docker pull dgpecurso06/examod02`, and the owner is listed as `dgpecurso06` with a profile icon.

PUBLIC REPOSITORY

dgpecurso06/examod02 ☆

Last pushed: 2 minutes ago

Repo Info Tags Collaborators Webhooks Settings

Short Description

Short description is empty for this repo.

Full Description

Full description is empty for this repo.

Docker Pull Command

```
docker pull dgpecurso06/examod02
```

Owner

dgpecurso06

La imagen se sube correctamente en el repositorio

Implementación de Rancher

Encenderemos la maquina virtual adecuada para implementar el Rancher, para una mejor manipulación nos conectamos vía CMD a la maquina donde implementaremos el Rancher al cual denominaremos *server*

```
C:\Users\olimun>ssh gustavo@192.168.1.110
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be established.
ECDSA key fingerprint is SHA256:cLuPa6ZMct+0lWNAYmEiQcTtVEMOZRBB+n6WQkFy5As.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hosts.
gustavo@192.168.1.110's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Wed Jun 13 15:04:24 2018 from 192.168.1.138
```

Con fines de ejemplificar este documento clonamos tres maquinas virtuales generando diferentes MAC Address para obtener cada una IP.

```
gustavo@ubuntu:~$ ifconfig | grep 192
inet addr:192.168.1.151 Bcast:192.168.1.255 Mask:255.255.255.0

gustavo@ubuntu:~$ ifconfig | grep 192
inet addr:192.168.1.169 Bcast:192.168.1.255 Mask:255.255.255.0

gustavo@ubuntu:~$ ifconfig | grep 192
inet addr:192.168.1.172 Bcast:192.168.1.255 Mask:255.255.255.0
```

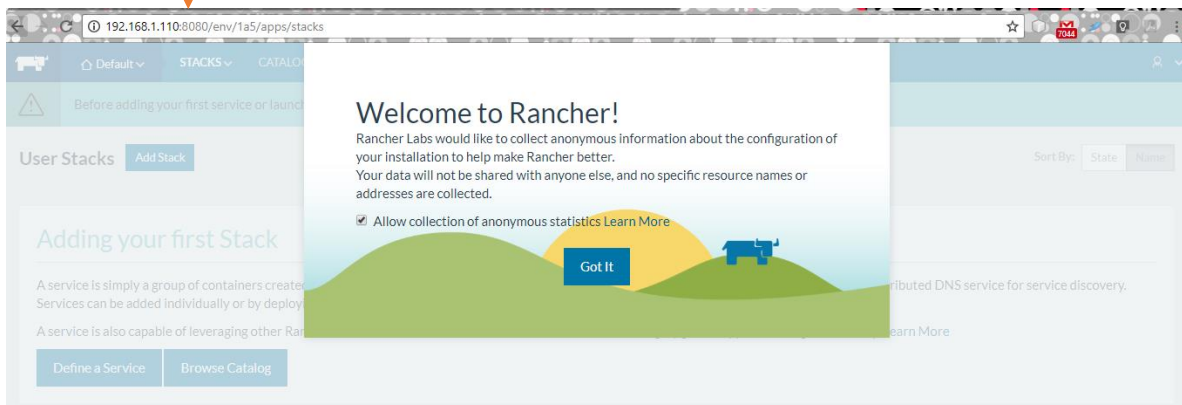
```
gustavo@ubuntu:/var/lib$ sudo rm -R rancher/
gustavo@ubuntu:/var/lib$ _
```

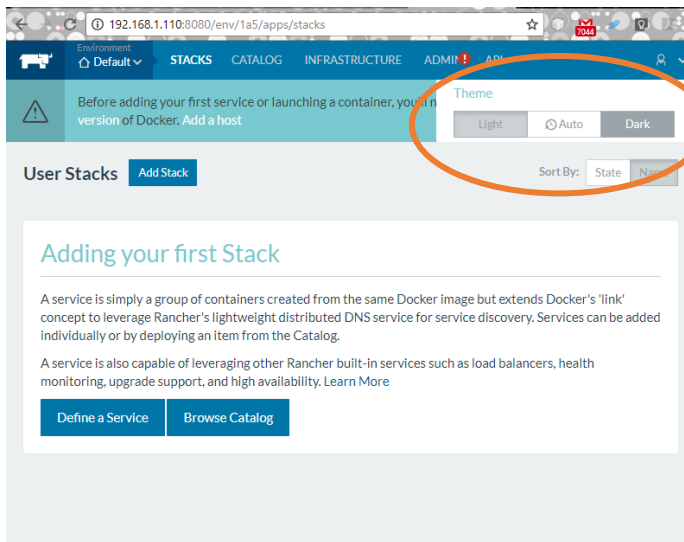
Eliminaremos el directorio de *rancher* ubicada en *var/lib*

Desplegaremos el Docker que contiene el Rancher vinculándolo al puerto 8080 en la maquina *server*

```
gustavo@server11:~$ sudo docker run -d --restart=unless-stopped -p 8080:8080 rancher/server
Unable to find image 'rancher/server:latest' locally
latest: Pulling from rancher/server
bae382666908: Pull complete
29ede3c02ff2: Pull complete
da4e69f33106: Pull complete
8d43e5f5d27f: Pull complete
b0de1abb17d6: Pull complete
422f47db4517: Pull complete
79d37de643ce: Pull complete
69d13e08a4fe: Pull complete
2ddfd3c6a2b7: Pull complete
bc433fed3823: Pull complete
b82e188df556: Pull complete
dae2802428a4: Pull complete
01ee44ae6d74: Pull complete
58edc581941f: Pull complete
960cf05fd0a8: Pull complete
2b5c34b98b97: Pull complete
2d0f458c8fc5: Pull complete
d6fce1097c48: Pull complete
b12d5fd766dc: Pull complete
6564119ffada: Pull complete
84f188954b45: Pull complete
Digest: sha256:0fb346d609fbee4da1d5a2aae647dda226e343250d523fb044629498c1131
Status: Downloaded newer image for rancher/server:latest
2407c023f6319172415f52b46f1e8a3e8a8ad5d41e20e34771676eb712452840
```

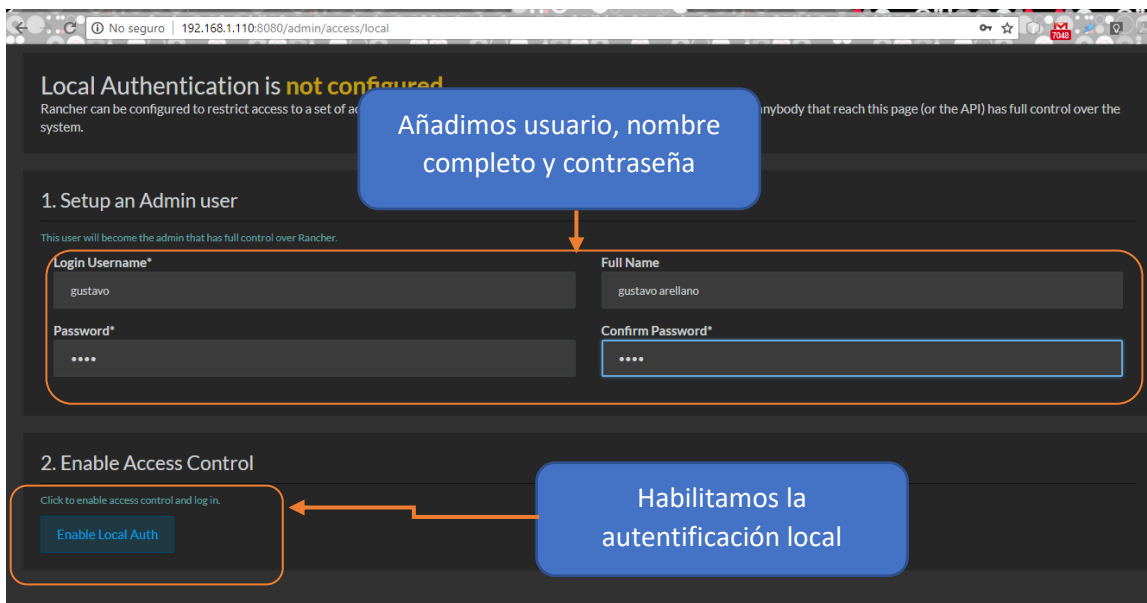
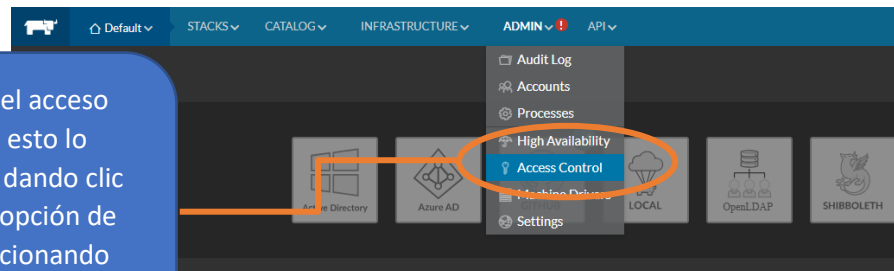
Abrimos un navegador y nos dirigimos a la IP de maquina donde corrimos al Rancher indicando el puerto vinculado





Modificaremos el tema de la interfaz grafica (Paso de mucha importancia), esto es posible dando clic sobre el icono de usuario

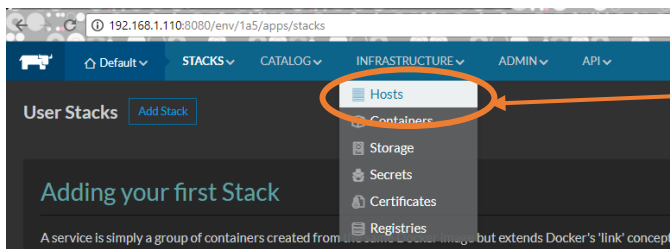
Configuraremos el acceso protegiéndolo, esto lo llevaremos acabo dando clic en el menú en la opción de "ADMIN" y seleccionando "Access Control".



Añadimos usuario, nombre completo y contraseña

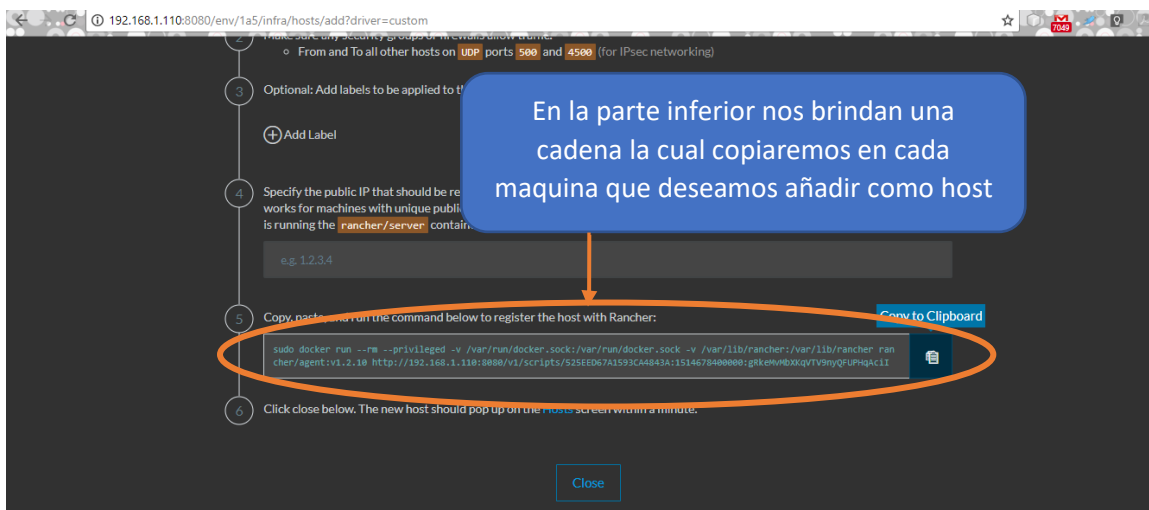
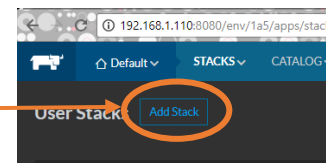
Habilitamos la autenticación local

Añadir hosts

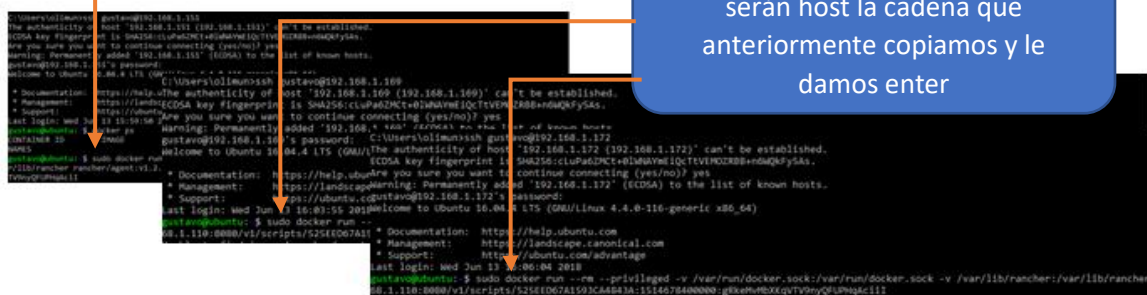


Para añadir un host necesitamos ir a la opción del menú “INFRASTRUCTURE” y seleccionar la opción de “Hosts”

Haremos clic en el botón de “Add Stack”



En la parte inferior nos brindan una cadena la cual copiaremos en cada maquina que deseamos añadir como host



Copiamos a las maquinas que serán host la cadena que anteriormente copiamos y le damos enter

192.168.1.110:8080/env/1a5/infra/hosts

Default STACKS CATALOG INFRASTRUCTURE ADMIN API

Hosts

[Add Host](#) ☒ Show System

ACTIVE

ubuntu

192.168.1.151 17.03.2-ce

Ubuntu 16.04.4 LTS (4.4.0)

2.9 GHz 1.94 GiB 18.6 GiB

Stack: healthcheck

healthcheck-1 10.42.216.104

Stack: ipsec

cn-driver-1 None

ipsec-1 10.42.53.249

Sidekicks

Stack: network-services

metadata-1 172.17.0.2

Sidekicks

network-manager-1 None

Stack: scheduler

ACTIVE

ubuntu

192.168.1.169 17.03.2-ce

Ubuntu 16.04.4 LTS (4.4.0)

2.9 GHz 1.94 GiB 18.6 GiB

Stack: healthcheck

healthcheck-2 10.42.198.176

Stack: ipsec

cn-driver-2 None

ipsec-2 10.42.42.128

Sidekicks

Stack: network-services

network-manager-2 None

metadata-2 172.17.0.2

Sidekicks

Standalone Containers

ACTIVE

ubuntu

192.168.1.172 17.03.2-ce

Ubuntu 16.04.4 LTS (4.4.0)

2.9 GHz 1.94 GiB 18.6 GiB

Stack: healthcheck

healthcheck-3 10.42.29.158

Stack: ipsec

cn-driver-3 None

ipsec-3 10.42.38.106

Sidekicks

Stack: network-services

network-manager-3 None

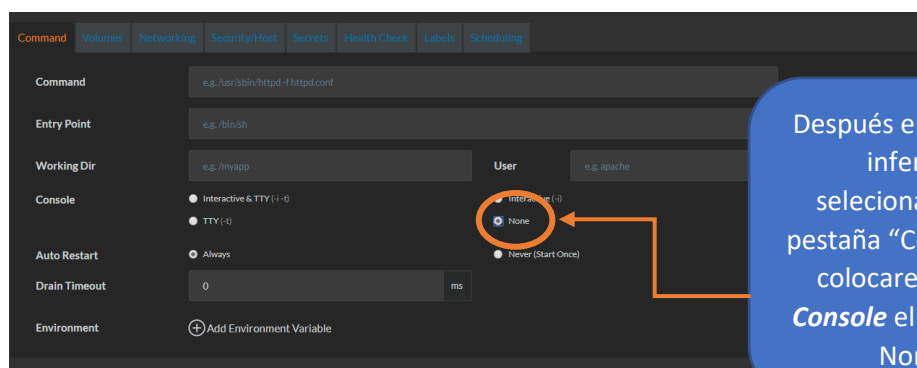
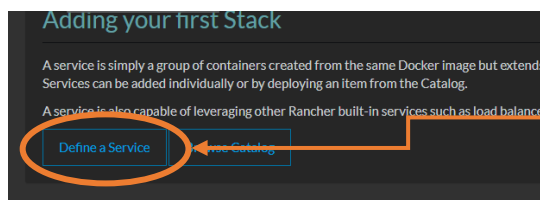
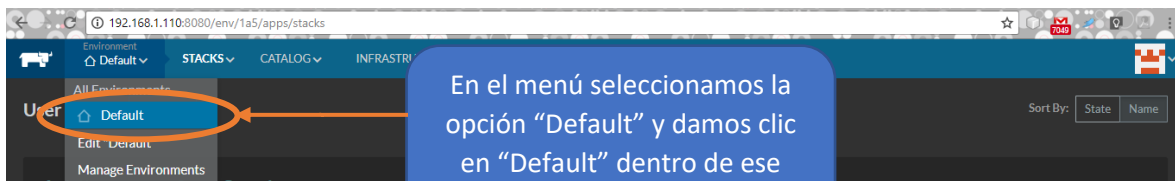
metadata-3 172.17.0.2

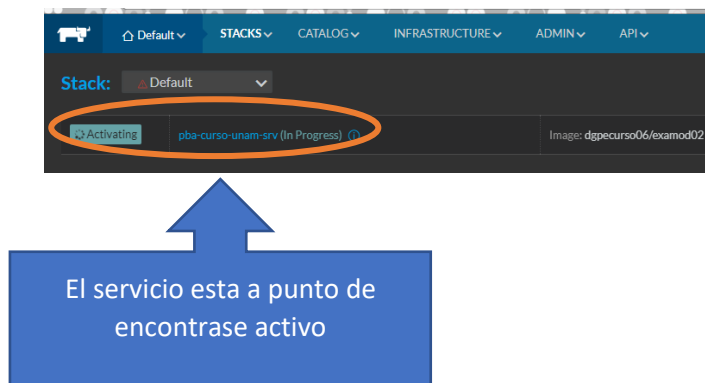
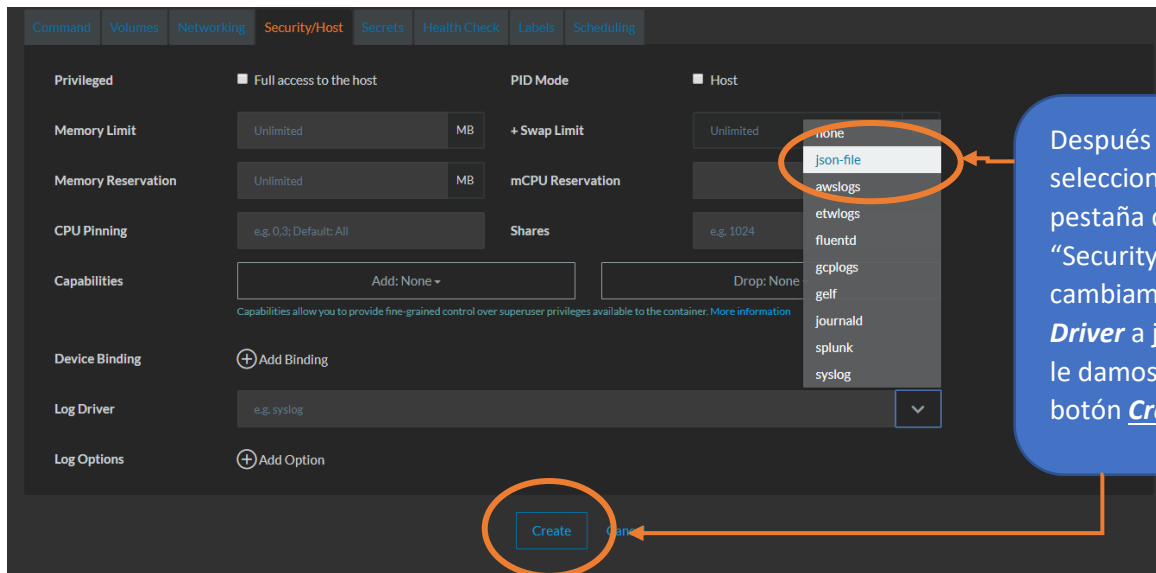
Sidekicks

Standalone Containers

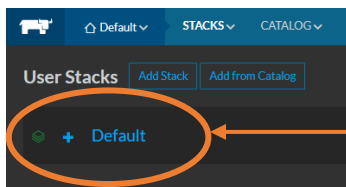
Los hosts fueron
añadidos
correctamente si
los vemos así en la
página del
administrador del
Rancher

Crear un servicio en el Rancher

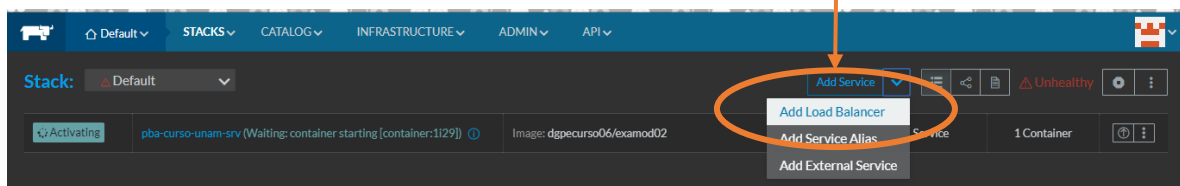




Creación del balanceador



En la pagina de Default le damos clic en Default nos llevara a la pagina donde se encuentra los servicios, ahí existe un apartado de "Add Service" y seleccionaremos "Add Load Balancer"



name: curso-lb Description: e.g. Balancer for mycompany.com

Port Rules

Access*	Protocol*	Request Host	Port*	Path	Target*	Port*
Public	HTTP	e.g. example.com	10001	e.g. /foo	Default/pba-curso-unam-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

Para la configuración del balanceador es necesario colocar el nombre, el puerto donde se consumirá el balanceado, seleccionamos el servicio que balancear y el puerto donde se despliega el Rancher y le hacemos un clic en el botón de *Crear*.

El balanceador después de la configuración podemos rectificar que este activo en la página de Default

Stack: Default

Stack	Name	Ports	Type	Containers	Actions
Active	curso-lb	To: pba-curso-unam-srv Ports: 10001/tcp	Load Balancer	1 Container	[Refresh] [Info]
Active	pba-curso-unam-srv	Image: dgpecurso06/examod02	Service	1 Container	[Refresh] [Info]

Escalar servicio

Stack: Default

Stack	Name	Ports	Type	Containers	Actions
Active	curso-lb	To: pba-curso-unam-srv Ports: 10001/tcp	Load Balancer	1 Container	[Refresh] [Info]
Updating-Active	pba-curso-unam-srv (In Progress)	Image: dgpecurso06/examod02	Service	1 Container	[Refresh] [Info]

Info (View Details)

Containers (2)

Scale 3 [Minus] [Plus]

Image: dgpecurso06/examod02

Entrypoint: None

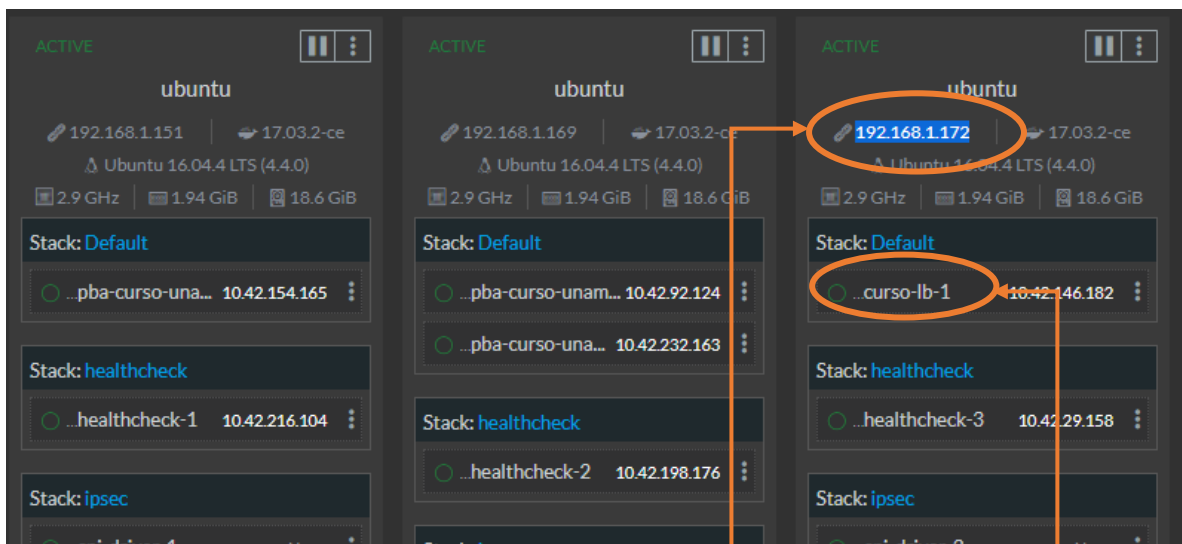
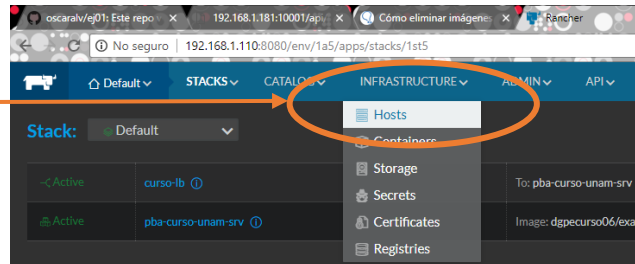
Command: None

Para escalar un servicio nos tenemos en posicionar en el icono de advertencia y desplegar una pestaña abajo donde encontramos el apartado de "Containers" y la opción de *Scale* donde daremos clic en el icono de mas para escalar hasta el número que desee

Resultados

Con todo lo anterior podemos probar que se este balanceando el servicio se realiza lo siguiente

Ingresando en el menú a "INFRASTRUCTURE" y seleccionando la opción de Hosts podemos ingresar a la siguiente pagina



Tenemos que observar donde se encuentra el balanceador que creamos, copiaremos la IP y la colocaremos en un navegador con el puerto con el que creamos el balanceador

```
192.168.1.172:10001/api/suma?var1=1&var2=3
{
  "variable1" : 1,
  "IP:" : "10.42.154.165",
  "variable2" : 3,
  "resultado" : 4
}
```

En el resultado podemos ver que realizamos la petición en la IP del balanceador y con el puerto establecido en su configuración, el resultado muestra también la identificación de la IP del equipo interno que esta respondiendo a la solicitud que concuerdan a las IPs de los host que no contiene el balanceador.

```
192.168.1.172:10001/api/suma?var1=1&var2=3
{
  "variable1" : 1,
  "IP:" : "10.42.232.163",
  "variable2" : 3,
  "resultado" : 4
}
```

```
192.168.1.172:10001/api/suma?var1=1&var2=3
{
  "variable1" : 1,
  "IP:" : "10.42.92.124",
  "variable2" : 3,
  "resultado" : 4
}
```