Temario

[1.- Creación del servicio REST 2](#_Toc516752349)

[2.- Creación del jar 6](#_Toc516752350)

[3.- Crear imagen Docker del microservicio con Dockerfile y subirlo a Docker Hub 9](#_Toc516752351)

[4.- Instalar y configurar Rancher 12](#_Toc516752352)

[5.- Generar la llave 15](#_Toc516752353)

[6.- Registrar equipo 17](#_Toc516752354)

[7.- Crear un servicio 22](#_Toc516752355)

[8.- Crear un balanceador 24](#_Toc516752356)

[9.- Consulta en el Navegador 26](#_Toc516752357)

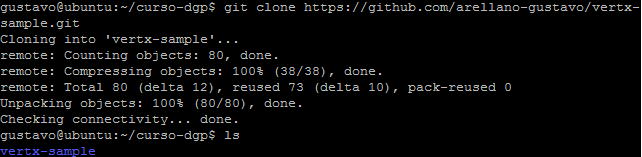
[10.- Escalando el servicio 28](#_Toc516752358)

1.- Creación del servicio REST

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

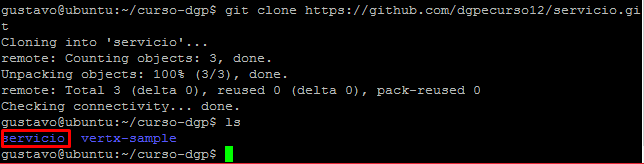
$ git clone <https://github.com/arellano-gustavo/vertx-sample.git>

Lo que nos generará una carpeta llamada “vertx-sample”.

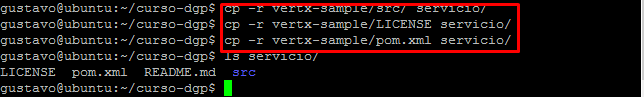


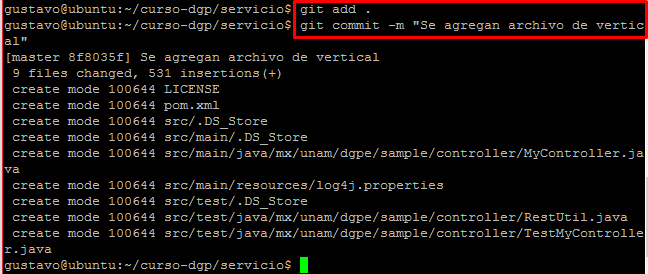
Lo que haremos será crear otro proyecto en Github llamado servicio y se debe clonar en el equipo a realizar el servicio:

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

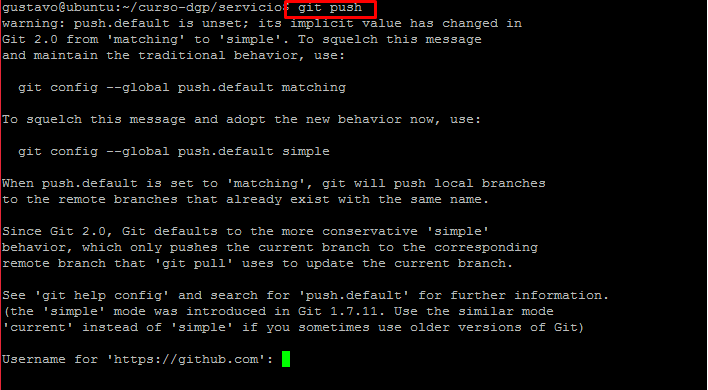
Lo que nos generará un carpeta llamada “servicio”.

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



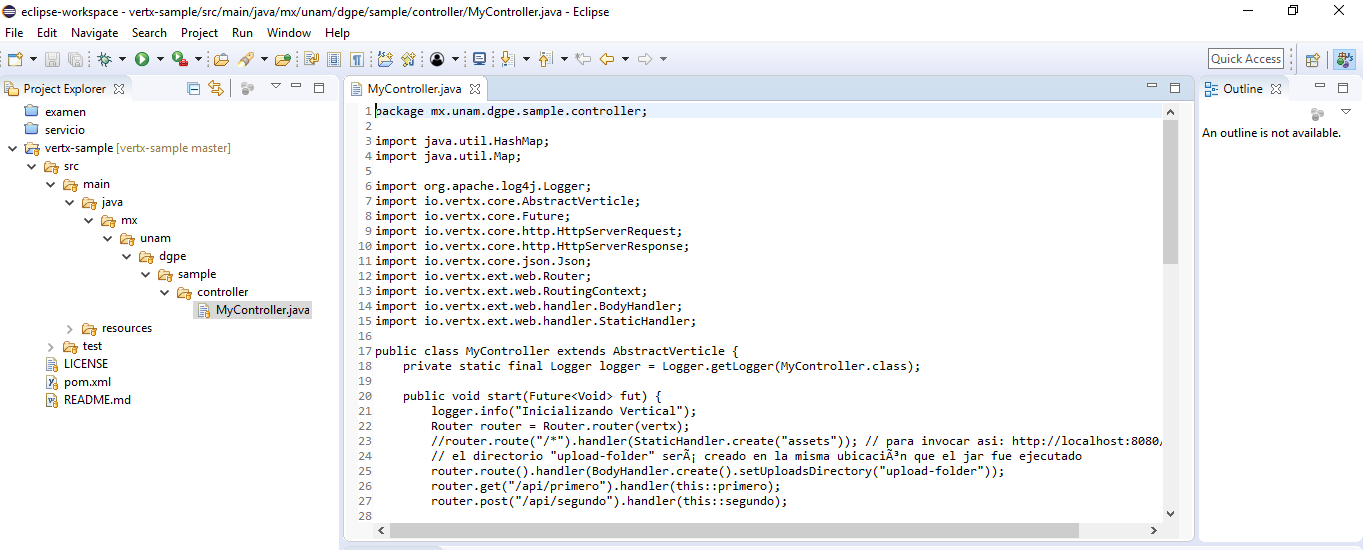
Se procede a agregar los archivos al git con el comando “git add .” y hacer commit con el comando “git commit -m “Comentario” “ en el proyecto.

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

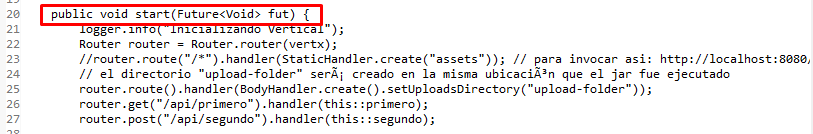


Con esto se tendrá un proyecto nuevo con el clone de vertx-sample, para poder modificar y generar un nuevo servicio rest.

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”

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

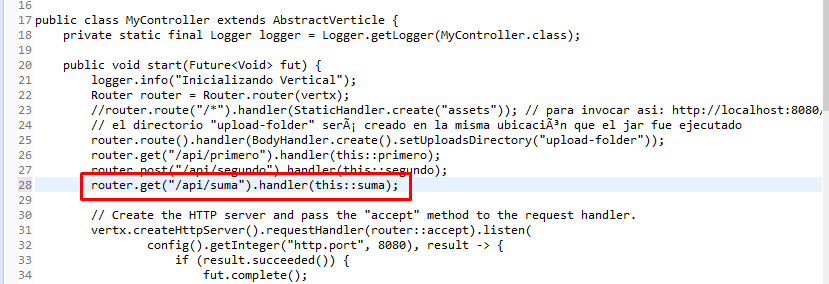
Esta clase de java tiene un método llamado:

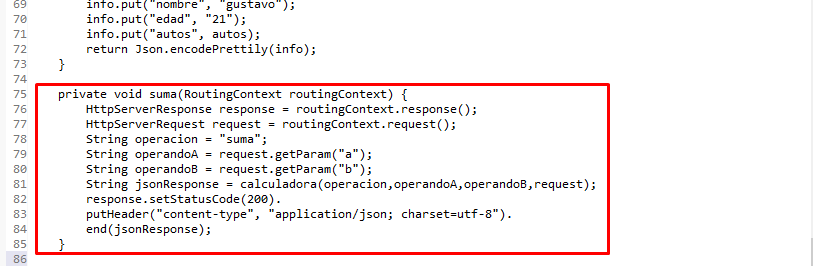
public void start(Future<Void> fut)

En el cual vamos a definir dos partes:

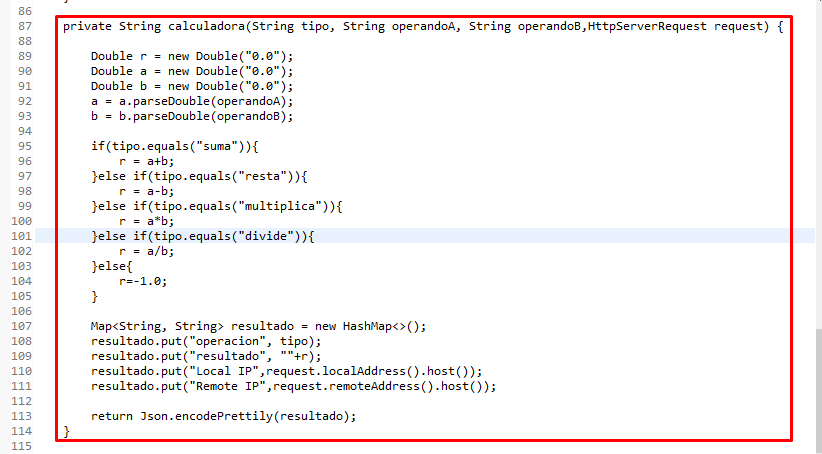
1.- El punto de entrada de nuestra acción, para ello debemos de agregar al objeto “router” el punto de entrada e indicarle con qué método de la clase será resuelto, esto se hace mediante el código:

router.get("/app/suma").handler(this::suma);



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

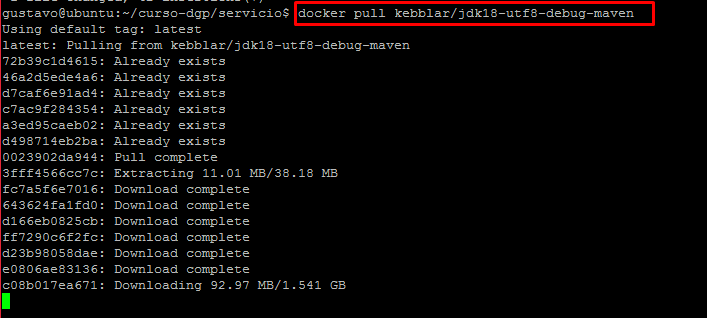
Para cada acción se debe de generar un método que atienda dicha petición y en el cual se regrese una cadena en formato JSON, se recomienda usar Mapas, ya que estos son sencillos de generar y su traducción a json no es compleja.

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

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 kebblar/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 kebblar/jdk18-utf8-debug-maven

El comando para compilar el proyecto es el siguiente:

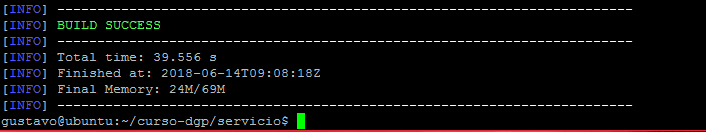
$ docker run -it -v /${RUTAFISICA}/:/codigo kebblar/jdk18-utf8-debug-maven mvn -f /codigo clean package

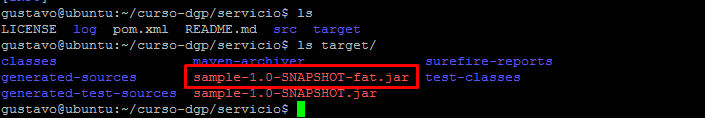
Donde ${RUTAFISICA} es la ruta donde se tiene el proyecto a compilar.

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

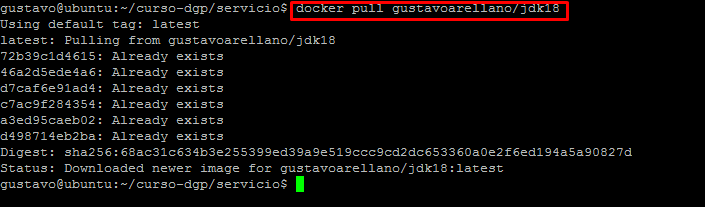
$ docker run -it -v /home/gustavo/curso-dgp/servicio/:/codigo kebblar/jdk18-utf8-debug-maven mvn -f /codigo clean package

https://lh3.googleusercontent.com/6gmWpZ18JH9u7gibonbLdt3yJ6VxjWjh6Va5iAUogU26Srb3zU-laKoEDCvWUplIWQu2jJAzMXX1MQt7A6InKloBk_5ViJz8RtmW-jEzExX4MMn8RGevMmFfRK1oguT4IR5NPR9C

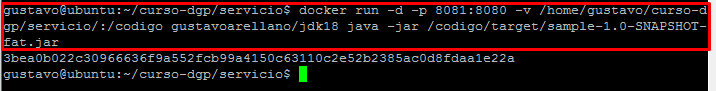
Si no hubo errores en el código debe aparecer la pantalla con **BUILD SUCCESS**

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

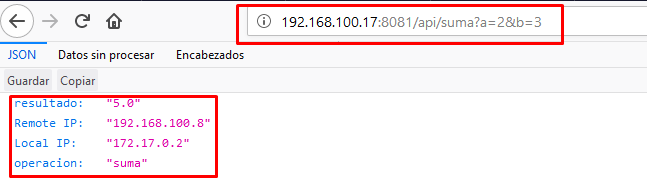
Para probar nuestro jar utilizaremos el docker gustavoarellano/jdk18, sino lo tenemos se baja con el comando:

$ docker pull gustavoarellano/jdk18

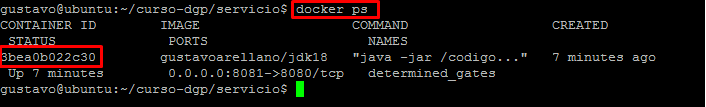
El comando para ejecutar el jar con el docker es:

$ 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

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 puesto 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:

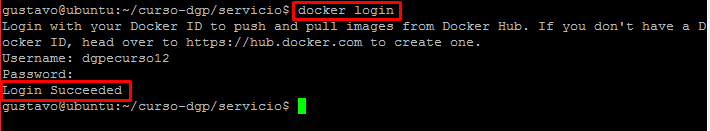
$ docker ps

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

Después debemos generar un archivo Dockerfile con el comando:

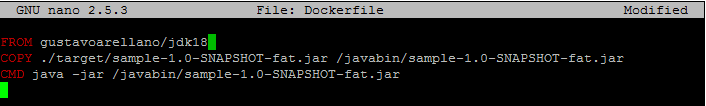
$ nano Dockerfilehttps://lh6.googleusercontent.com/cjK9yT3ICiGBCB0Yshompi_sZ3qHuYe3adH4h4jKlGeLhBgRb29JmmzjAee_1PUWwIvwbF_mYv8oR_MD4gcLs2eybhW6c1wFGu-k0V8-AH_ytJdZY-46hWooNN34bh7zU80YYAIw

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

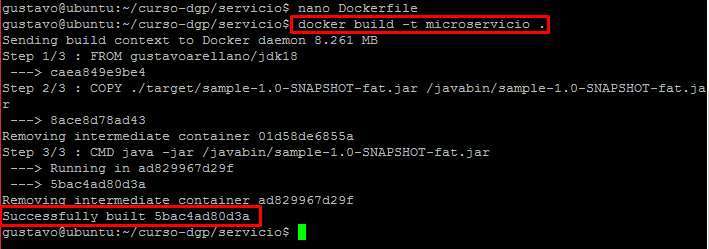
FROM gustavoarellano/jdk18

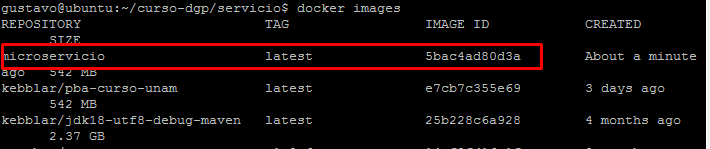
COPY ./target/sample-1.0-SNAPSHOT-fat.jar /javabin/sample-1.0-SNAPSHOT-fat.jar

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



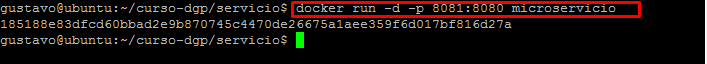
Ejecutaremos el siguiente comando para compilar y crear la imagen del docker:

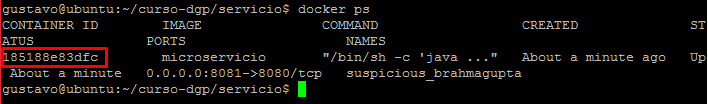
$ docker build -t microservicio .

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

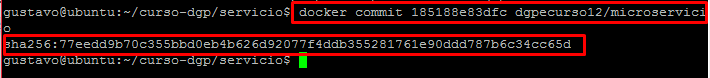
Ejecutamos la imagen del docker creada con el comando:

$ docker run -d -p 8081:8080 microservicio

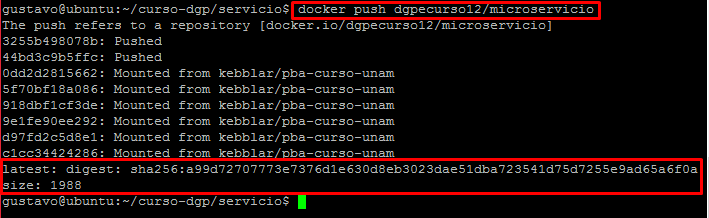


Verificamos que esté corriendo el docker con el comando siguiente y obtenemos el ID:

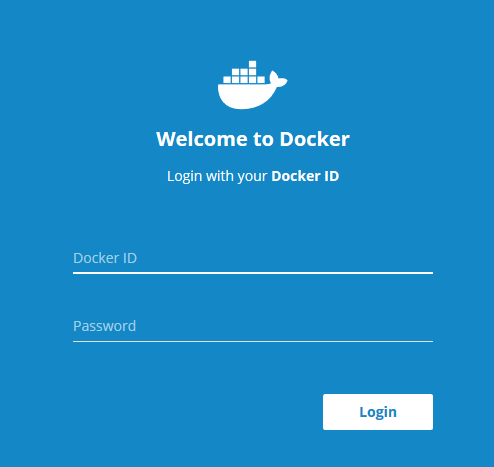
Con el ID obtenido en el paso anterior ejecutamos el comando y debe responder con una clave sha256:

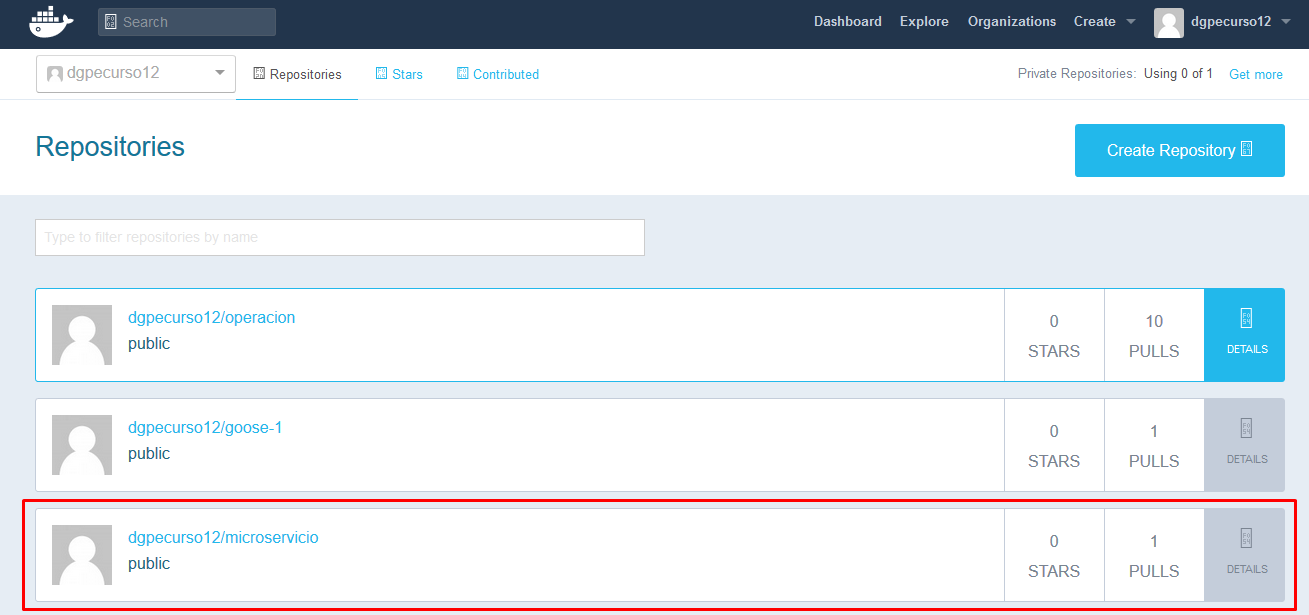
$ docker commit 185188e83dfc dgpecurso12/microservicio

Finalmente ejecutamos el comando para subirlo a Docker Hub:

$ docker push dgpecurso12/microservicio

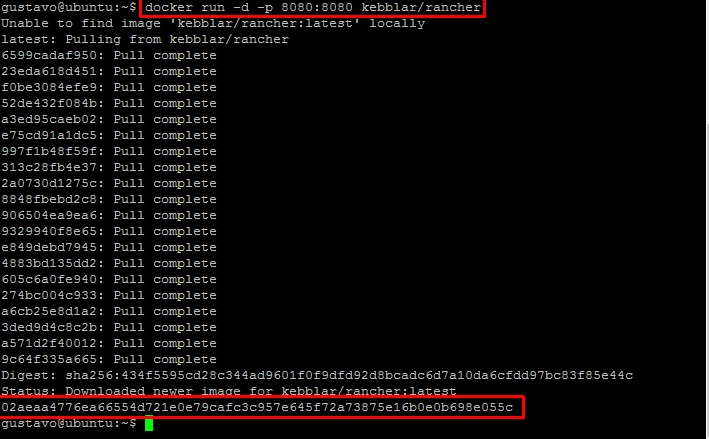
Para comprobar que este sí se haya subido correctamente ingresamos a Docker Hub con nuestro usuario y password.

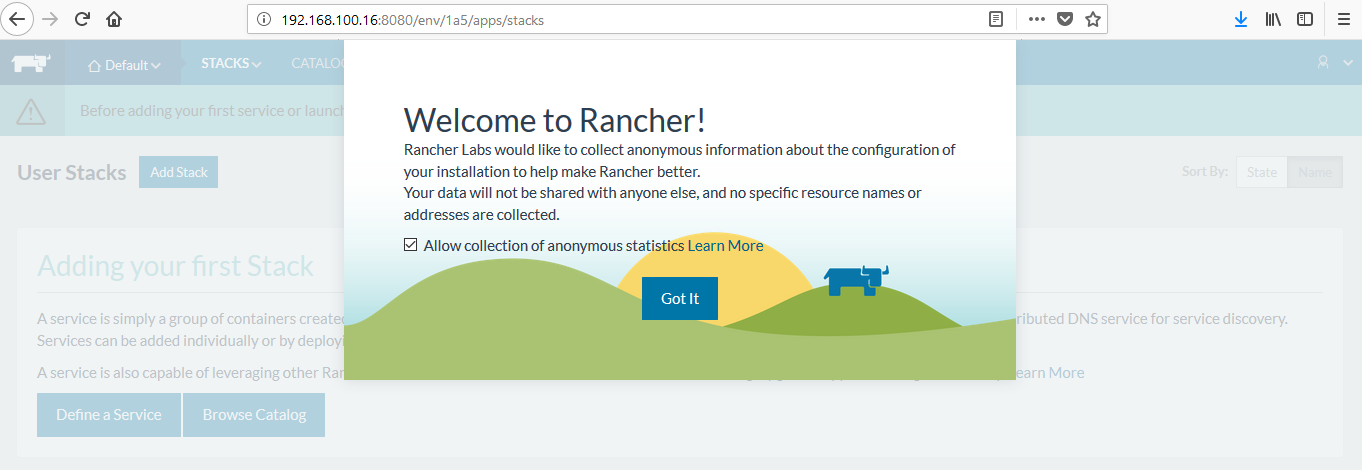


Y debemos observar el docker creado.

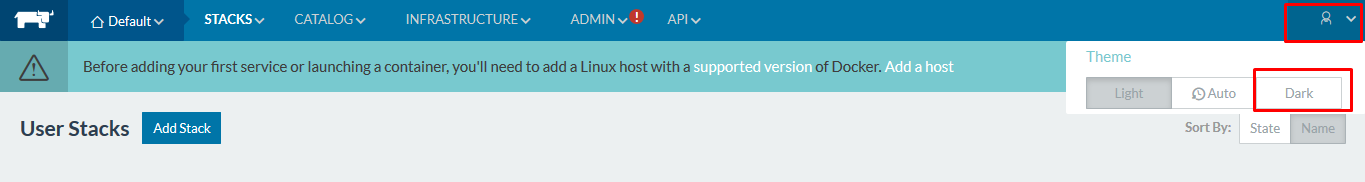
4.- Instalar y configurar Rancher

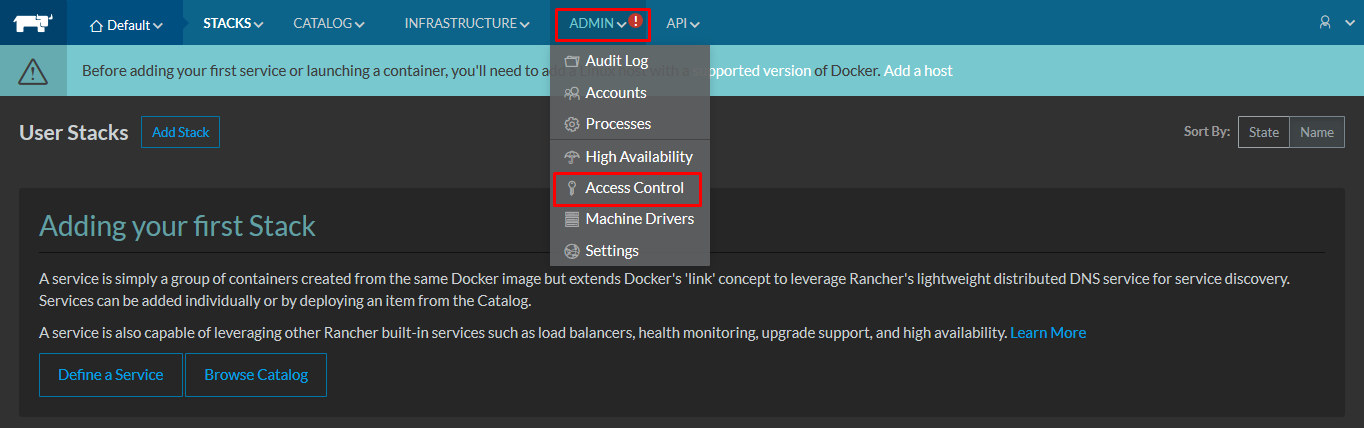
Para instalar Rancher se tomará uno que ya está en un docker, con el comando se procederá a bajarlo e iniciarlo:

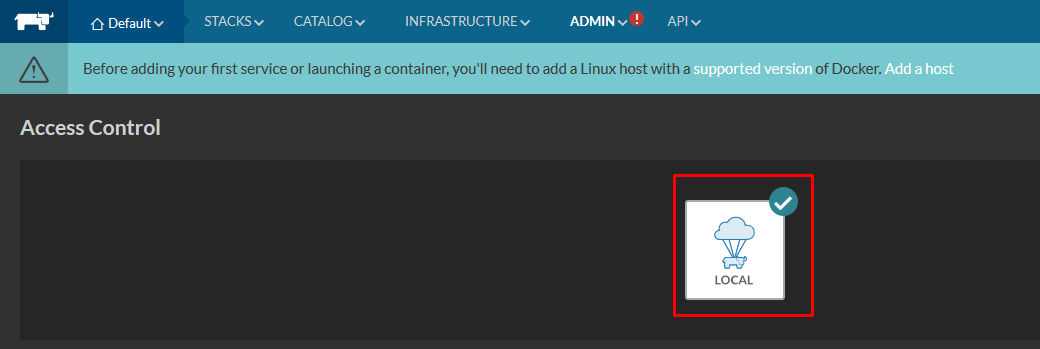
$ docker run -d -p 8080:8080 kebblar/rancher

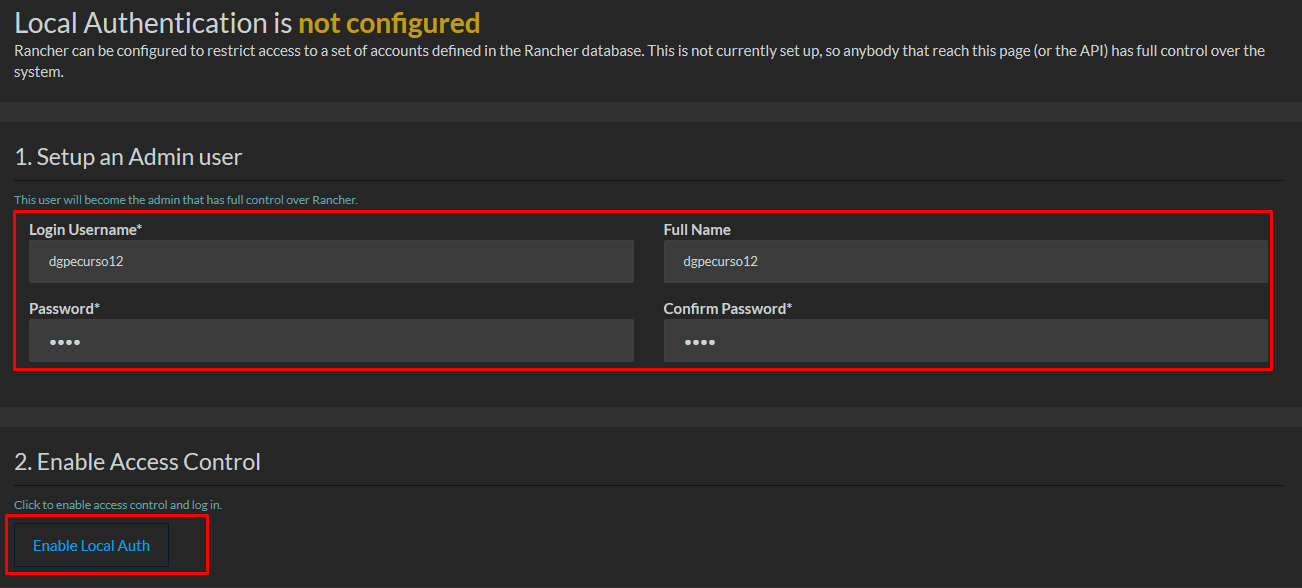
Para visualizar el Rancher ejecutándose ingresar a la IP del equipo donde se instaló en el puesto 8080 en un navegador, en este caso 192.168.100.16:8080.

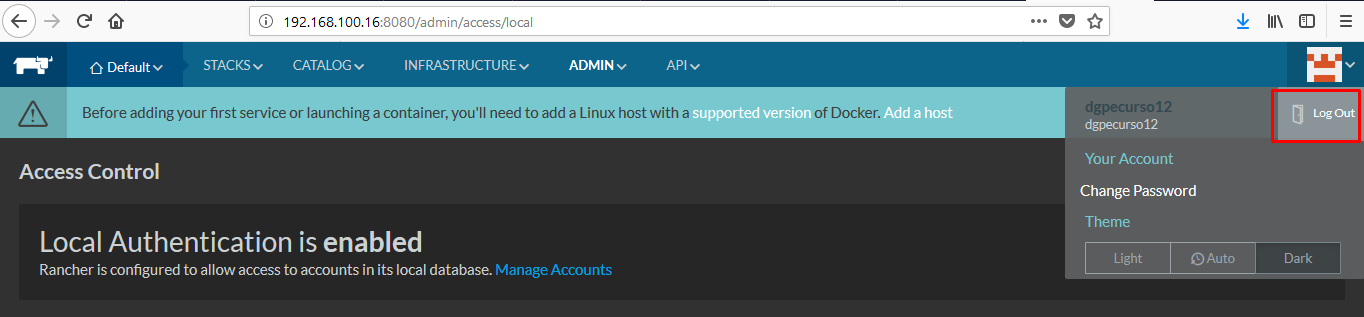
Esta pantalla indica que ha iniciado el Rancher.

Primero cambiaremos el color de la aplicación al Dark

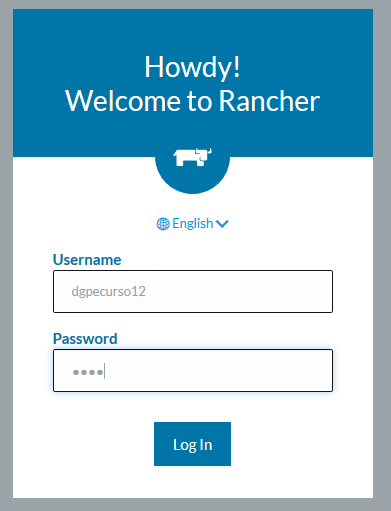
Primero debemos configurar el acceso, para esto se ingresará al menú ADMIN y submenú Access Control 

Después se seleccionara la autenticación local.

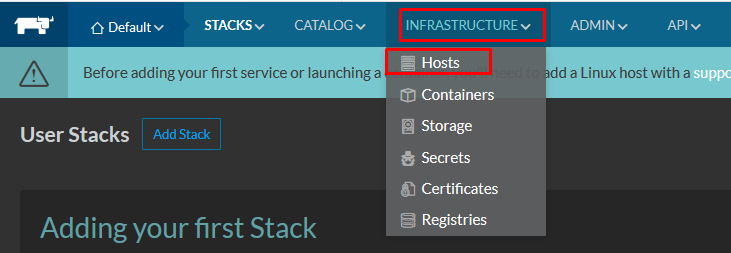
Se deberán de registrar los datos de la autenticación y oprimir el botón “Enabel Local Auth”.

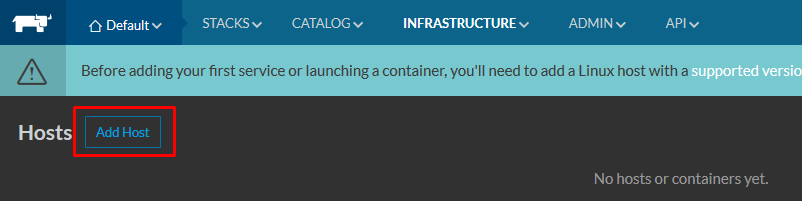
Inmediatamente cerrar sesión.

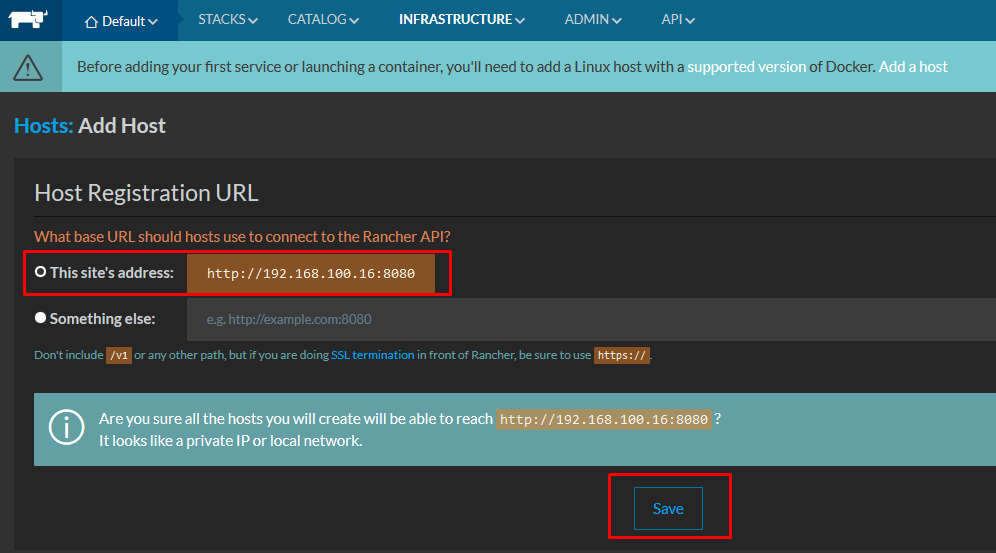
E ingresar de nuevo al Rancher.

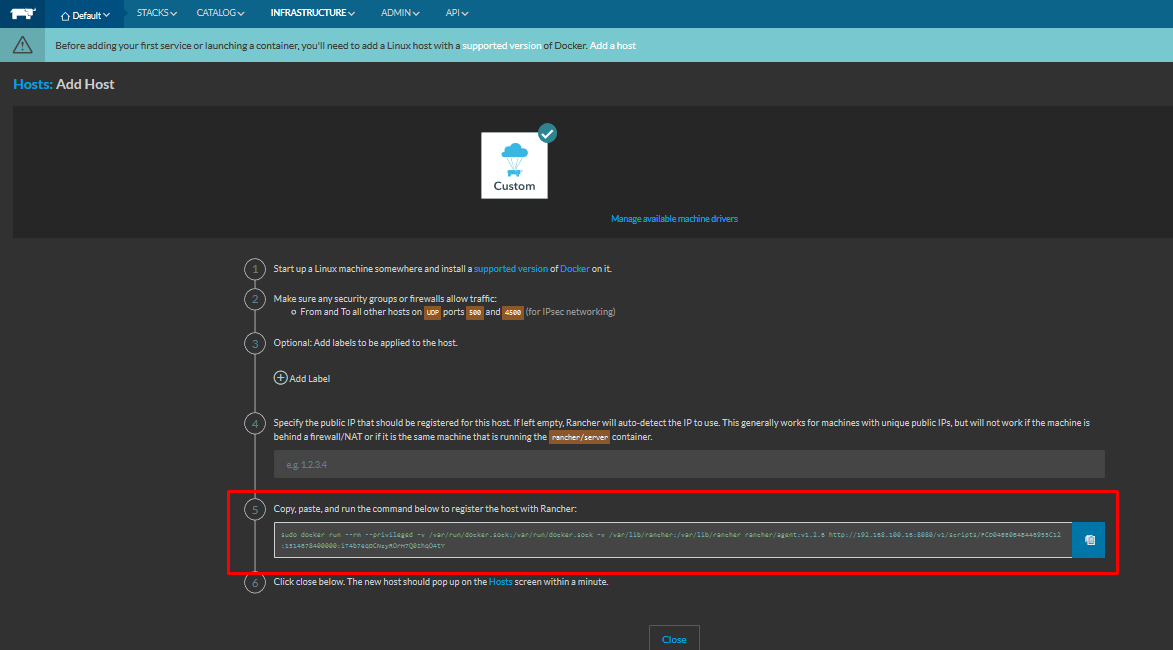


5.- Generar la llave

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.

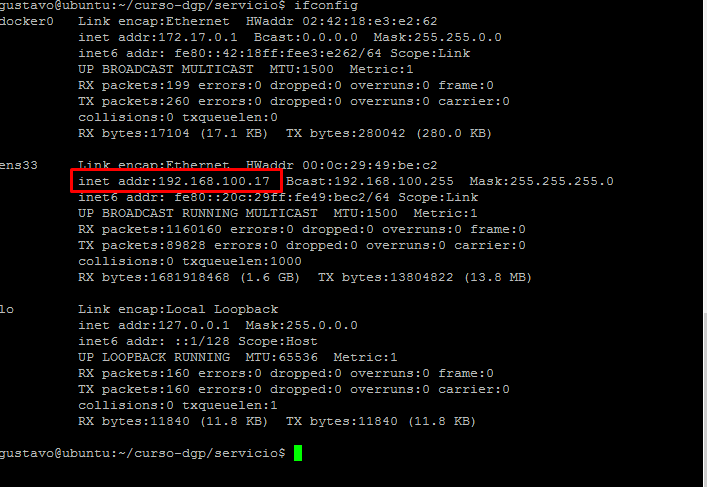
También se ha generado un url con una llave la cual se debe de ejecutar en los equipos clientes. La cual es un comando para instalar docker con rancher y un agente el cual es el que va estar monitoreando a los clientes.

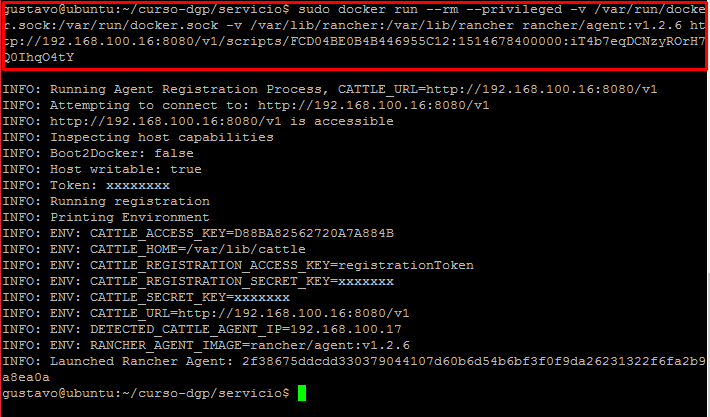
sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.6 <http://192.168.100.16:8080/v1/scripts/FCD04BE0B4B446955C12:1514678400000:iT4b7eqDCNzyROrH7Q0IhqO4tY>

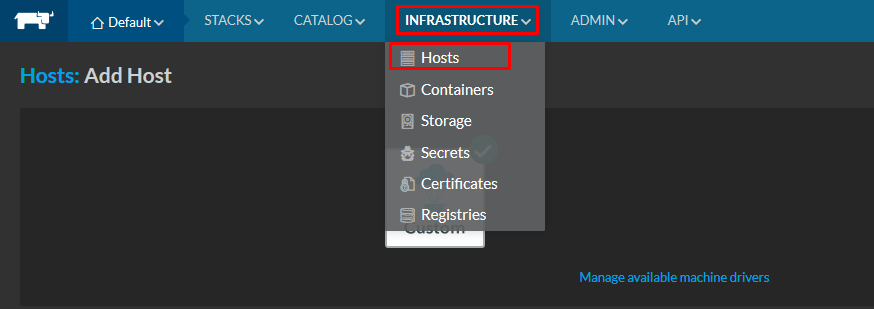
Esta llave varía de equipo a equipo, nunca es la misma.

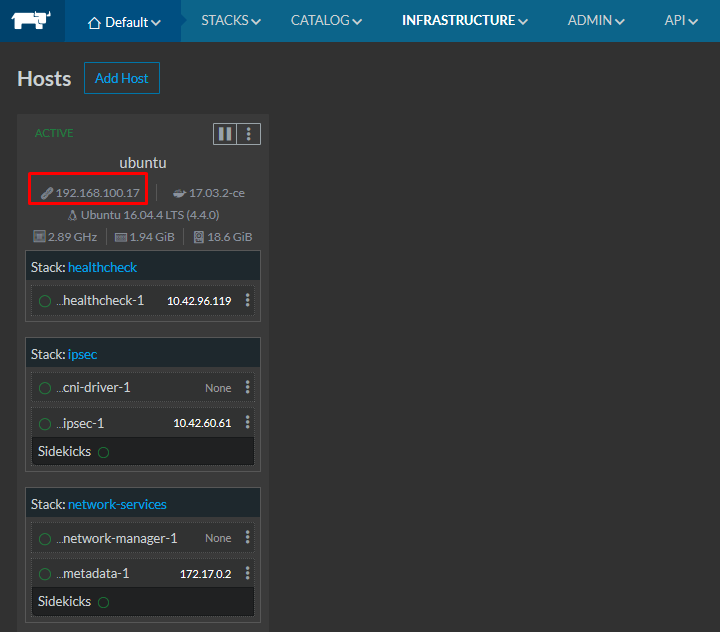
6.- Registrar equipo

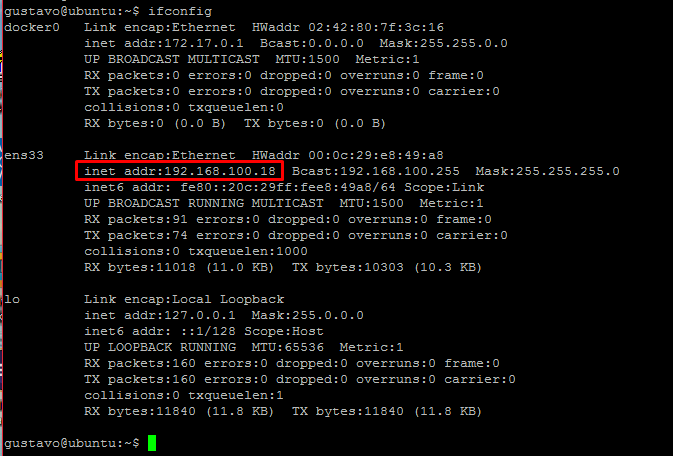
La llave que se creó en el apartado anterior se debe de copiar y ejecutar en otro equipo distinto a donde se encuentra Rancher.

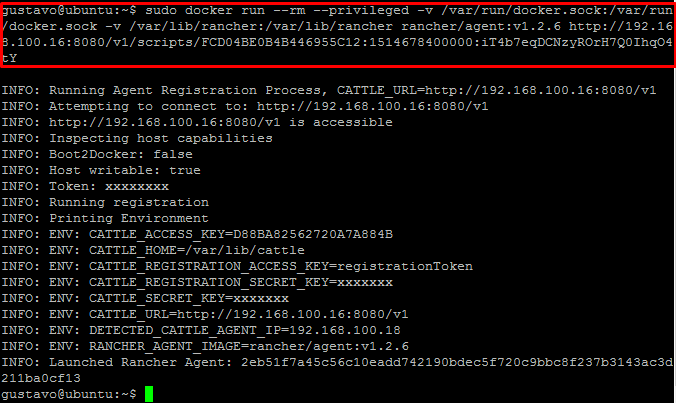
Se ejecutará en el equipo que tiene la IP 192.168.100.17

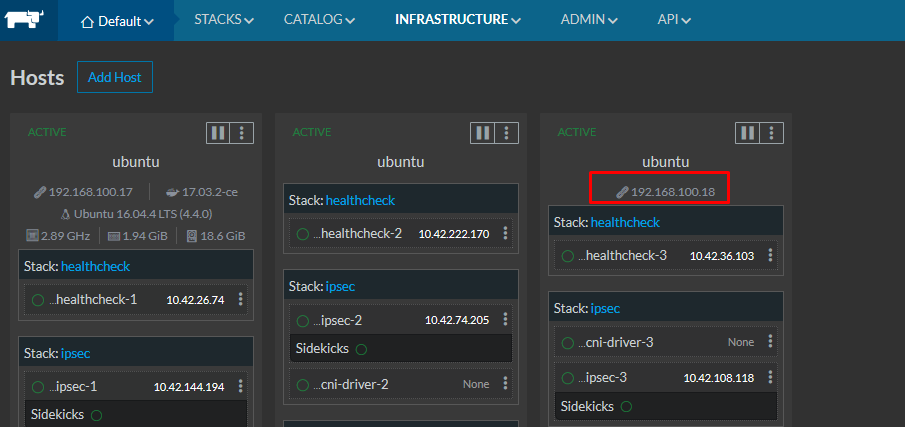
Se ejecuta la llave que se copió y debe de registrarse el equipo en el **Host**.

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

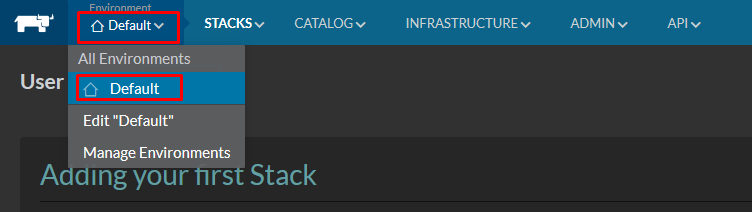
Se deberá de visualizar el equipo con la **IP** previamente registrada. 

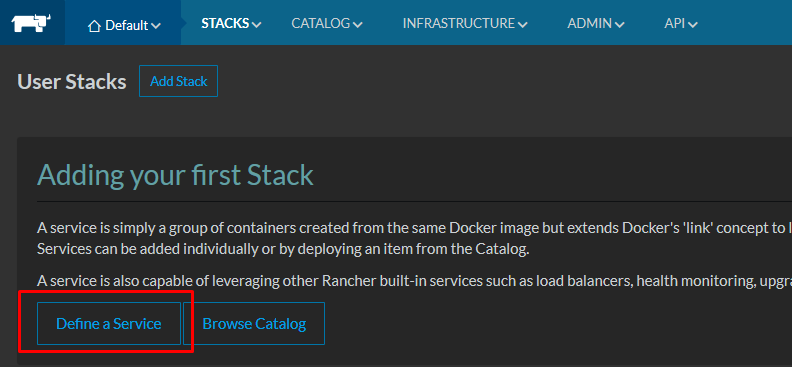
Se procederá a ejecutar la llave de registro en otro equipo con la IP 192.168.100.18

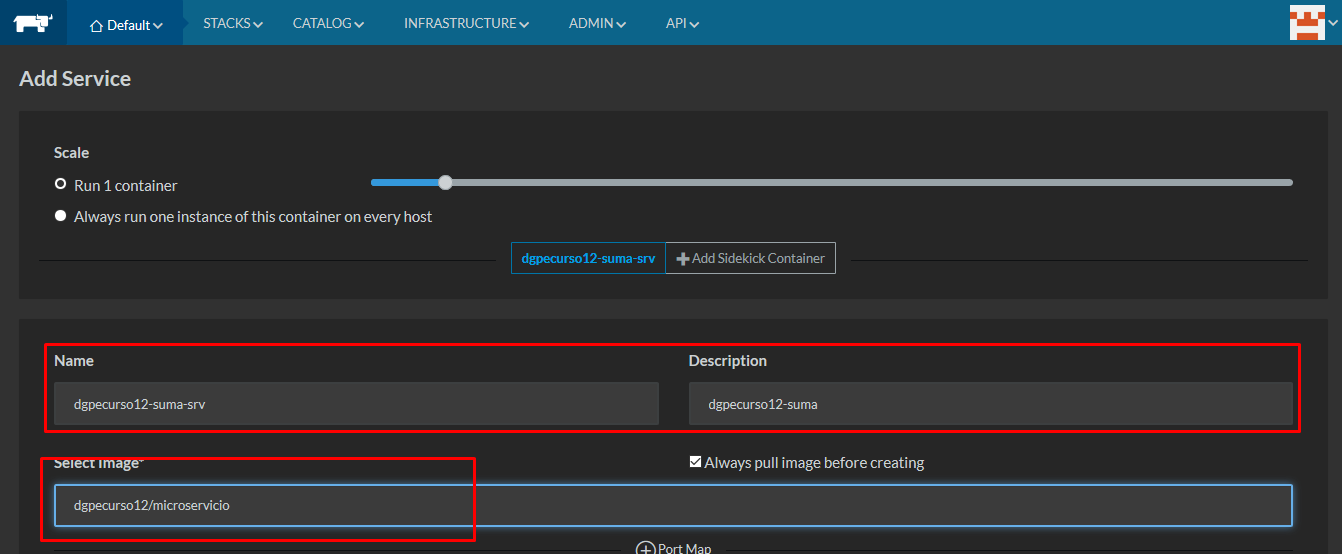
Se copia la llave nuevamente en el nuevo cliente.

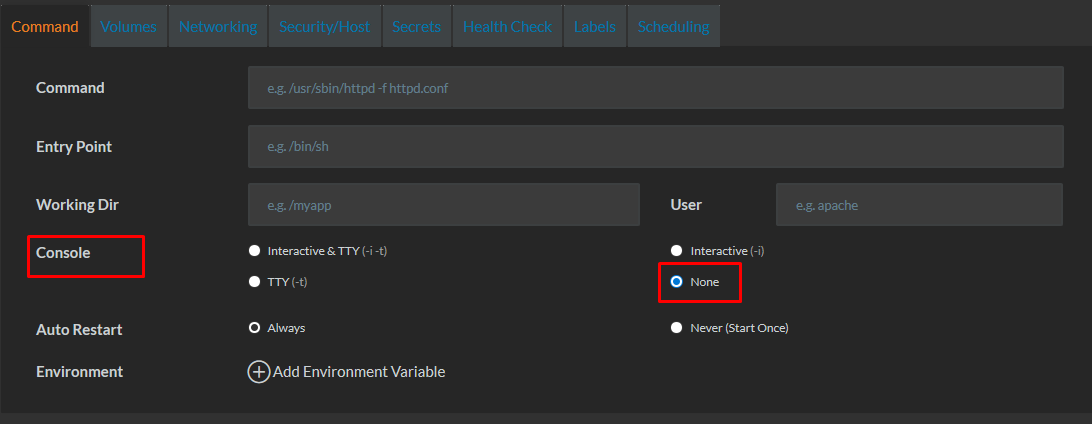
Se deberá de visualizar el equipo con la **IP** previamente registrada.

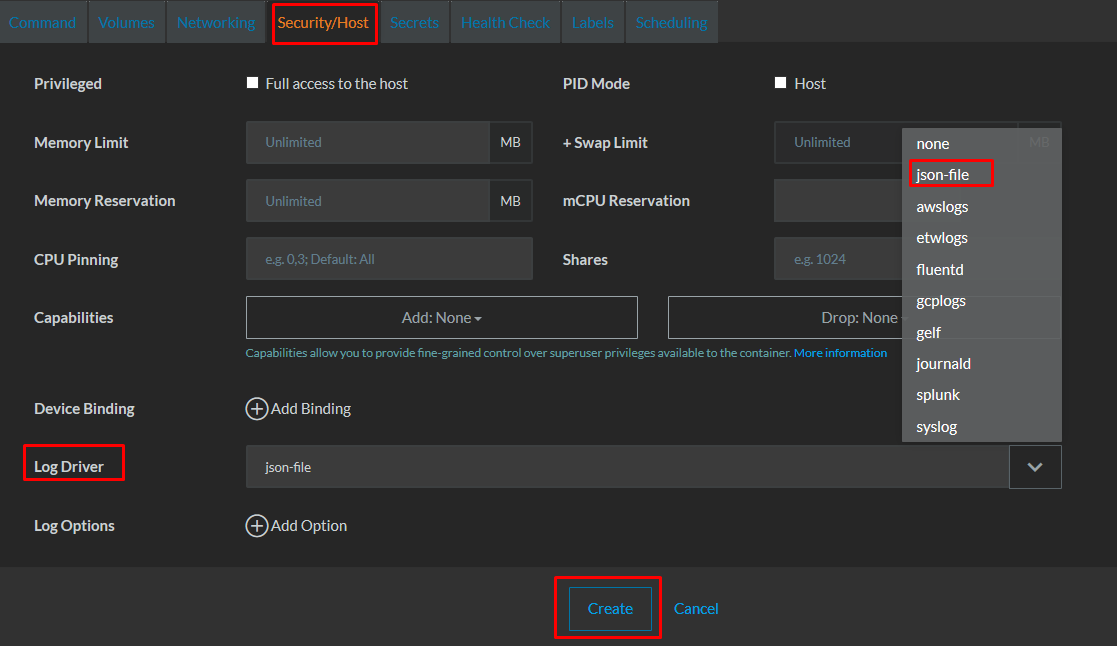
7.- Crear un servicio

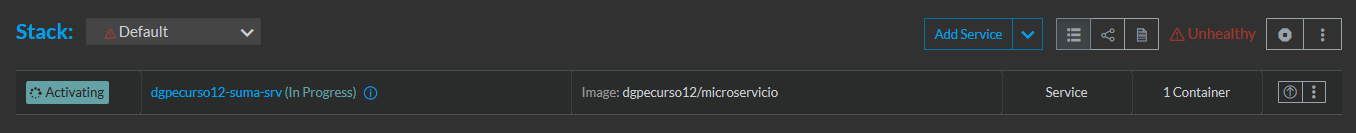
Primero se debe de ir al Menú **Default** y submenú **Default**

Oprimir el botón **Define a Service** 

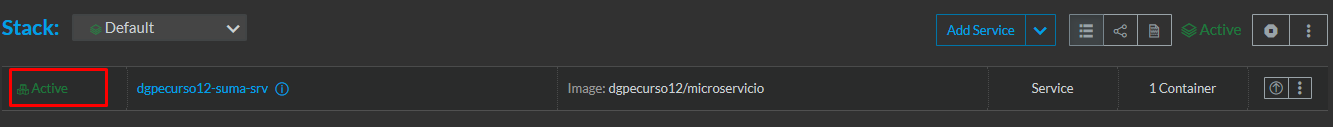
Ingresar el nombre del servicio y la descripción, así como la imagen que se bajara de Docker Hub, en este caso se utilizará el docker **dgpecurso12/microservicio** 

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

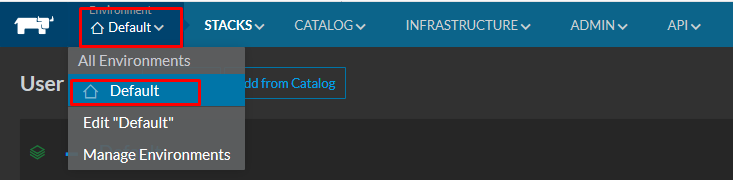
En la pestaña **Security/Host** en donde dice **Log Driver** seleccionar  **json-file** y después oprimir el botón **Create**.

Se observará que empezará a activarse el servicio.

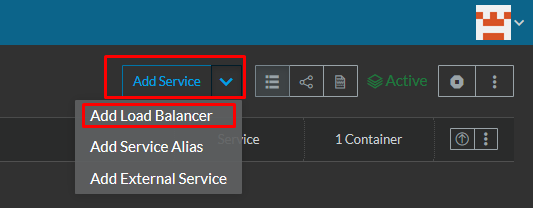
Después de unos segundos o minutos quedará activo el servicio.

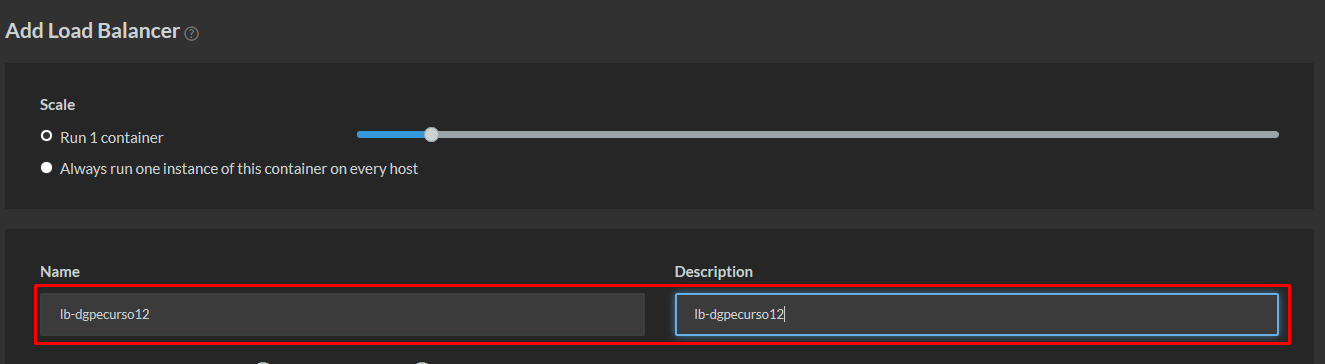


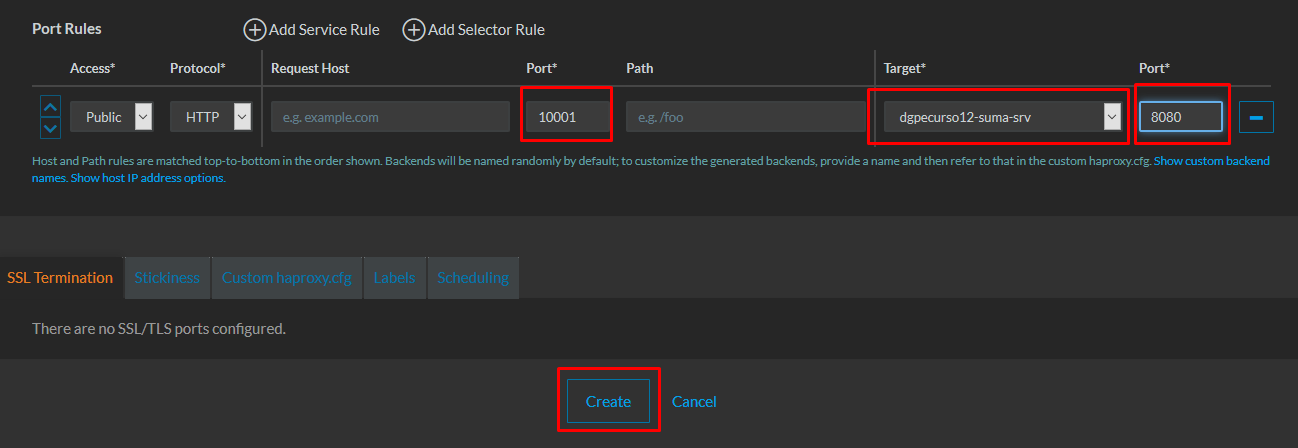
8.- Crear un balanceador

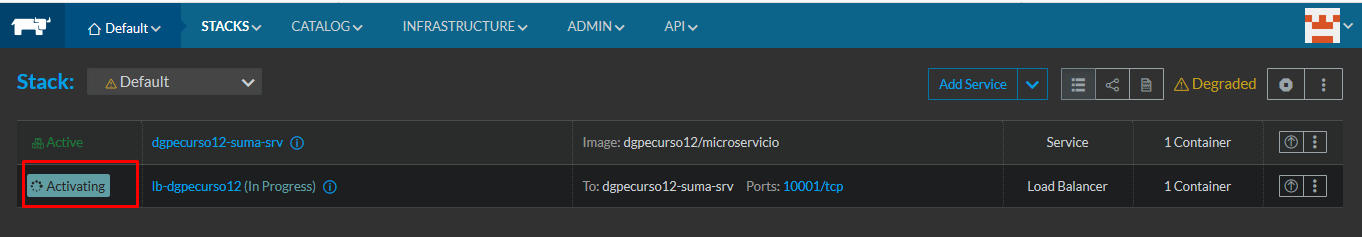
Primero se debe de ir al Menú **Default** y submenú **Default**

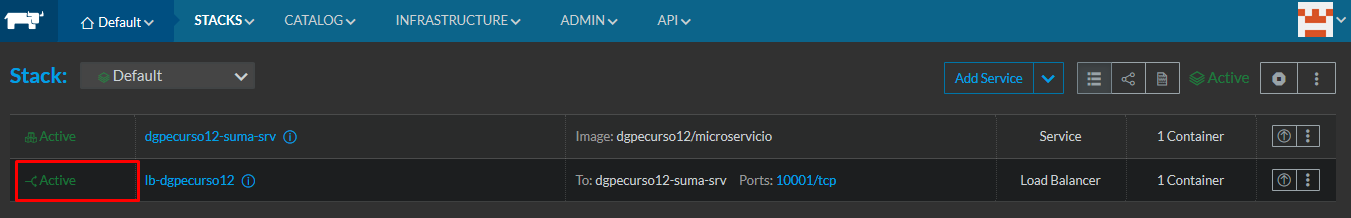
En la parte derecha se visualiza



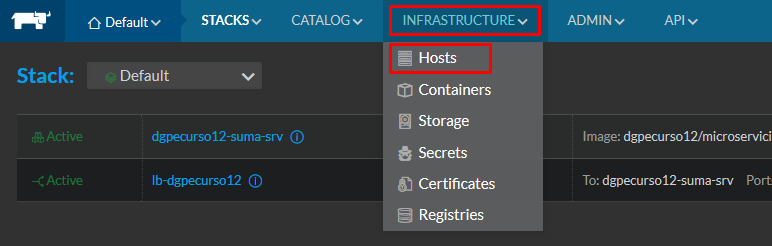
Ingresar el nombre y descripción del balanceador

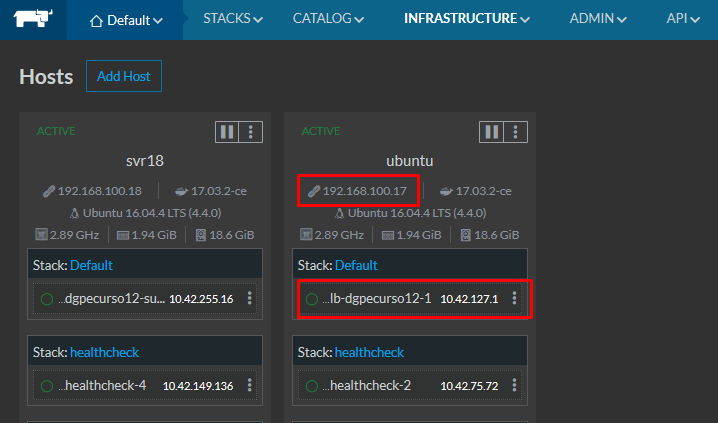
Ingresar el puerto del balanceador y seleccionar el servicio que se creó anteriormente dgpecurso12-suma-srv y oprimir el botón **Create** 

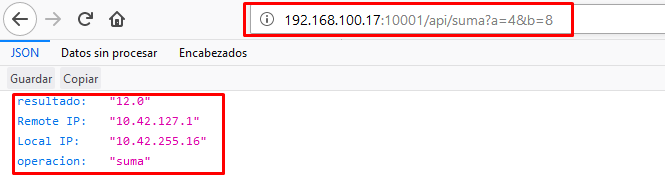
Se visualizará que se está activando el balanceador.

Después se visualizará que quedo activo el balanceador.

9.- Consulta en el Navegador

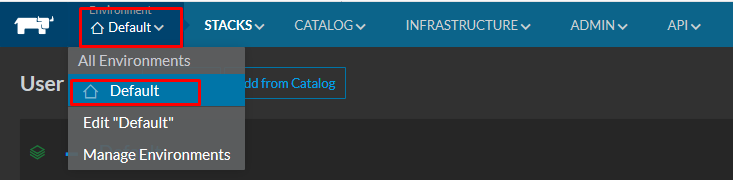
Para visualizar en el navegador que se ha generado el balanceador y servicio correctamente, se debe ir al menú **INFRASTRUCTURE** y submenú **Host.** 

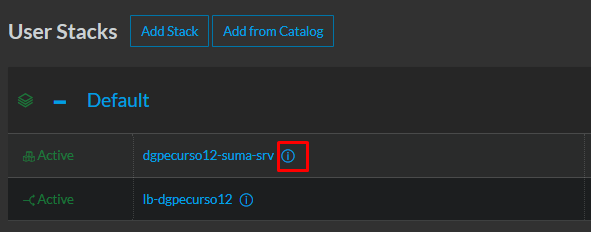
En esta pantalla se visualiza que el balanceador quedo en la **IP 192.168.100.17**  

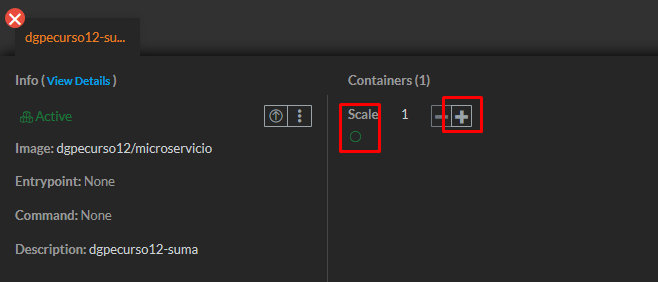
Para visualizar el servicio en el navegador se debe ingresar la IP y el puerto del balanceador, así como los datos del servicio a consultar, en este caso 192.168.100.17:10001/api/suma?a=4&b=8

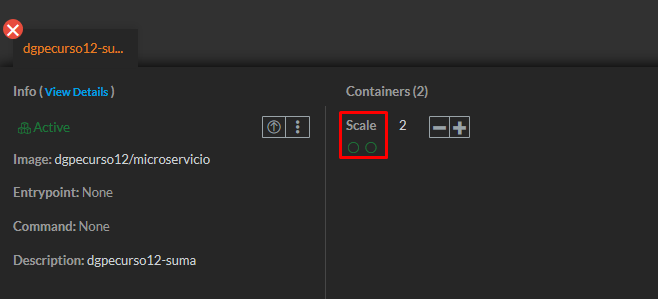
Las IP’s mostradas corresponde a, **Local IP** a donde se encuentra el servicio que está respondiendo, y la **Remote IP** desde donde se realizó la petición, en este caso donde está el **Load Balancer**.

10.- Escalando el servicio

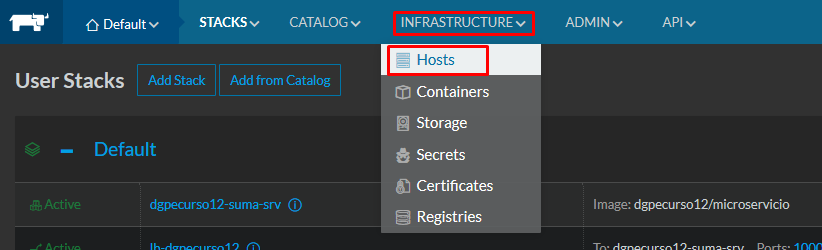
Primero se debe de ir al Menú **Default** y submenú **Default**

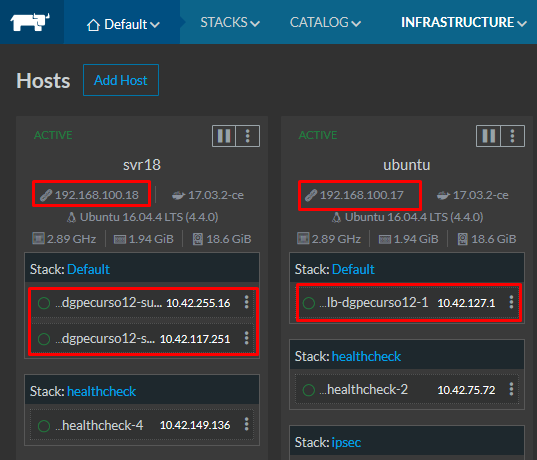
Se debe oprimir en el icono 

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

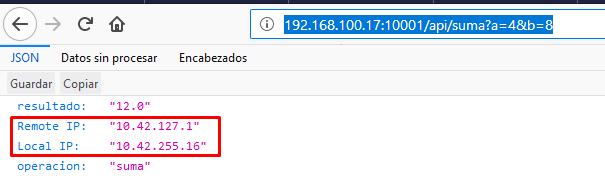
Después se observa que ya está escalado a 2 

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**