

MANUAL DE INSTALACIÓN Y EJECUCIÓN DE SISTEMA DE INTEGRACIÓN DE MÓDULOS IoT (SIMIoT)

Materiales y Herramientas:

1. Sistema Operativo Linux o Subsistema de Windows para Linux ([Guía de Instalación WSL](#))
2. Git Bah ([Guía de Instalación Git](#))
3. Docker Engine ([Guía de Instalación de Docker Engine](#))
4. RDBMS Postgress ([Guía de Instalación Postgres](#))

Introducción:

Para eliminar las dependencias con el sistema operativo o el versionamiento de las herramientas utilizadas, se recomienda levantar el Sistema de Integración de Módulos IoT mediante el uso de contenedores.

Para ello haremos uso de la herramienta de Swarmpit, un fácil gestor de enjambre de contenedores, que nos va a permitir administrar la conectividad entre los distintos contenedores a levantar. Esta es una herramienta opcional, puede utilizar cualquier otro gestor inclusive el propio gestor de Docker Engine.

Recursos:

- [Repositorio: Sistema de Registro de Módulos de IoT](#)
- [Repositorio: Api de Estandarización](#)
- [Repositorio: Plantillas de Docker-Compose para SIMIoT](#)

Procedimiento:

Paso 1: Instanciación de Swarmpit: Gestor de Enjambre de contenedores

- a) Iniciar un enjambre de contenedores de Docker ingresando el siguiente comando desde una terminal:

\$ docker swarm init

```
keyla@LAPTOP-N4M8K800:~$ docker swarm init
Swarm initialized: current node (s14dshsj8ov64fwen6tbquji) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5iw6crry8aawcpwdks0ddg0ltteg6qd2d54ahtplhxx5unvklh-20zab3kawdxotjp8we7tae6lo 192.168.65.4:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

- b) Ejecutar el contenedor de Swarmpit mediante la siguiente línea de comandos, donde username es el nombre de usuario que se desea establecer y password la contraseña:

```
$ docker run --rm --name swarmpit-installer --volume /var/run/docker.sock:/var/run/docker.sock -e INTERACTIVE=0 -e ADMIN_USERNAME=username -e ADMIN_PASSWORD=password swarmpit/install:1.9
```

Este comando descargará la imagen de Swarmpit en caso de que no la tenga localmente, y se mantendrá una configuración por defecto, las credenciales de administrador pueden ser diferentes a las mostradas en la imagen:

```
keyla@LAPTOP-N4M8K880:~$ docker run --rm --name swarmpit-installer --volume /var/run/docker.sock:/var/run/docker.sock -e INTERACTIVE=0 -e ADMIN_USERNAME=keyla -e ADMIN_PASSWORD=passwordadmin swarmpit/install:1.9
user
Welcome to Swarmpit
Version: 1.9
Branch: 1.9

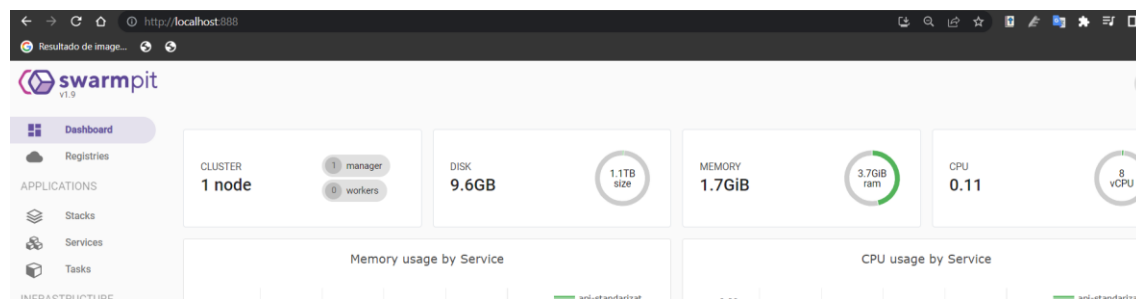
Preparing dependencies
7.67.0-r0: Pulling from lucashalbert/curl
Digest: sha256:d30a7ac1114b667e95976461f1b5c345fbc7545a6787cd890d134c76b6b99849
Status: Image is up to date for lucashalbert/curl:7.67.0-r0
docker.io/lucashalbert/curl:7.67.0-r0
DONE.

Preparing installation
Cloning into 'swarmpit'...
Note: switching to '91987c59bc3209120f6fe514c14572129771e60b'.
```

- c) Una vez configurado, se presentará un resumen de la configuración:

```
Application setup
Stack name: swarmpit
Application port: 888
Database volume driver: local
Admin username: keyla
Admin password: passwordadmin
DONE.
```

- d) Finalmente podrá acceder a la interfaz, mediante el puerto 888 de su ip local.



Paso 2: Construcción de los Contenedores de las aplicaciones

Sistema de Registro de Módulos IoT:

- a. Descargar el repositorio de github ([IoT-Module-Registration](https://github.com/eggarcia05/iot-module-registration-interface.git)), además del código fuente, contiene el archivo Dockerfile con las configuraciones necesarias para construir la imagen del contenedor.

Para descargarlo puede ejecutar la siguiente línea de comando:

```
$ git clone https://github.com/eggarcia05/iot-module-registration-interface.git
```

- b. Desde la terminal, ubicarse dentro del directorio local del repositorio descargado y ejecutar el siguiente comando para crear una imagen del sistema de registro de Módulos IoT.

```
$ docker build -t iot-module-register .
```

Este iniciará el proceso de construcción de la imagen, una vez finalizado ya tendrá una imagen lista para utilizar.

Api de estandarización:

- a. Descargar el repositorio de github ([Api-haystack-standardization](https://github.com/eggarcia05/api-haystack-standardization.git)), además del código fuente, contiene el archivo Dockerfile con las configuraciones necesarias para construir la imagen del contenedor. Para descargarlo puede ejecutar la siguiente línea de comando:

```
$ git clone https://github.com/eggarcia05/api-haystack-standardization.git
```

- b. Desde la terminal, ubicarse dentro del directorio local del repositorio descargado y ejecutar el siguiente comando para crear una imagen del sistema de registro de Módulos IoT.

```
$ docker build -t api-standardization .
```

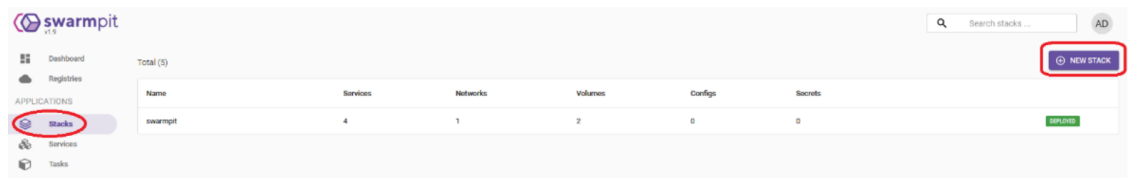
Este iniciará el proceso de construcción de la imagen, una vez finalizado ya tendrá una imagen lista para utilizar.

Paso 3: Preparación de la base de datos

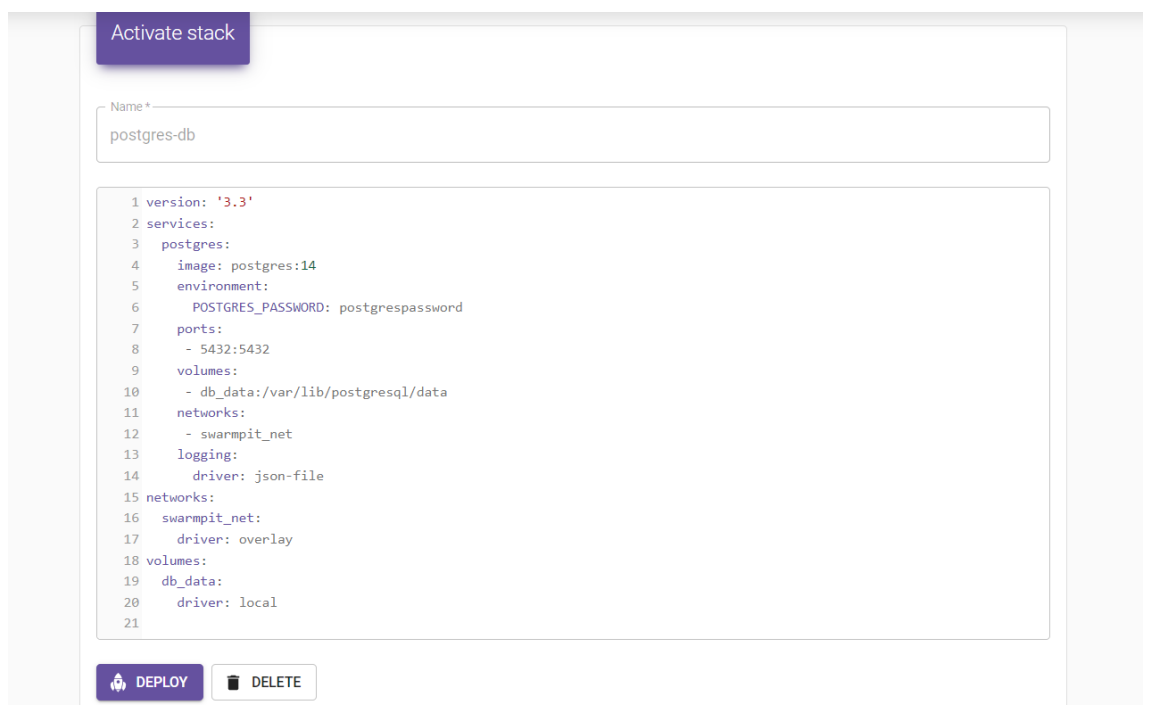
En esta sección se debe restaurar una copia del esquema de la base de datos requerida para el funcionamiento del sistema, para ello debió instalar la herramienta número 3, la RDBMS de postgres.

Esta copia puede crearla en el servidor de postgres que se crea localmente luego de la instalación o dentro de un contenedor de Docker con una instancia de postgres. Si escoge esta última opción deberá seguir los siguientes pasos:

1. Diríjase a la herramienta de swarmpit (<http://localhost:888/>) y presione el botón de stacks y luego de clic en NEW STACK.



2. En la pantalla presentada agregue la configuración del contenedor de postgres el cual puede encontrar en el RRR, en la carpeta de postgres-db ([postgres-db](#)), luego presiona DEPLOY.



La variable de entorno POSTGRES_PASSWORD, puede ser reemplazada por una de su preferencia.

3. Una vez levantado el contenedor de postgres, deberá crear una nueva base de con el nombre “lab_telemetria_db”, para ello ejecute las siguientes líneas de comando.

psql postgresql://postgres:postgrespassword@localhost:5432/postgres

Una vez ejecutado el comando, se ejecuta el próximo para generar la base de datos

CREATE DATABASE lab_telemetria_db ;

```
eggarcia@DESKTOP-84LUASJ:/mnt/c/Users/Erick_Garcia/Documents/docker/postgres$ psql postgresql://postgres:postgrespassword@localhost:5432/postgres
psql (14.5 (Ubuntu 14.5-1.pgdg20.04+1))
Type "help" for help.

postgres=# CREATE DATABASE lab_telemetria_db;
CREATE DATABASE
postgres=#
```

4. Finalmente copiaremos el esquema que lo encontrará en el repositorio antes mencionado, para ello, debemos tener clonado en nuestra máquina el repositorio [simiot-containers](#) ([simiot-containers](#)) y dentro de la carpeta `postgres-db`, ejecute el siguiente comando para copiar la base de datos:

psql

**postgresql://postgres:postgrespassword@localhost:5432/lab_telemetria_db -f
lab_telemetria_template.sql**

Paso 4: Ejecución de los contenedores de SIMIoT

Para la siguiente sección deberá agregar los contenedores en los stacks de swarmpit como se mostró en el paso 3.

Hasura Engine

El archivo de configuración para el motor de hasura lo encontrará en la carpeta “[pr-hasura](#)” del [Repositorio: Plantillas de Docker-Compose para SIMIoT](#).

Las variables de entorno requeridas son:

HASURA_GRAPHQL_ADMIN_SECRET: perc

HASURA_GRAPHQL_DATABASE_URL: postgresql://postgres: [contraseñaPostgres]
@[IP-DOMAIN]/lab_telemetria_db

HASURA_GRAPHQL_DEV_MODE: 'true'

HASURA_GRAPHQL_METADATA_DATABASE_URL: postgresql://postgres:
[contraseñaPostgres]@[IP-DOMAIN]:5432/postgres

Donde [IP-DOMAIN] es la IP de la máquina donde se encuentra desplegada la base de datos, o en caso de tenerlo, del dominio de su pc, el cuál puede conseguir al ejecutar el comando **ifconfig eth0**.

```
keyla@LAPTOP-N4M8K800:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.24.178.122 netmask 255.255.240.0 broadcast 172.24.191.255
    inet6 fe80::215:5dff:fe96:e652 prefixlen 64 scopeid 0x20<link>
    ether 00:15:5d:96:e6:52 txqueuelen 1000 (Ethernet)
    RX packets 2076 bytes 274209 (274.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 969 bytes 119780 (119.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Por otro lado, [contraseñaPostgres] es la contraseña establecida para postgresql en el paso anterior. Luego presione DEPLOY.

Verificar los datos ingresados concuerden con los que colocó en el paso anterior para postgresql.

Name*
pr-hasura

```

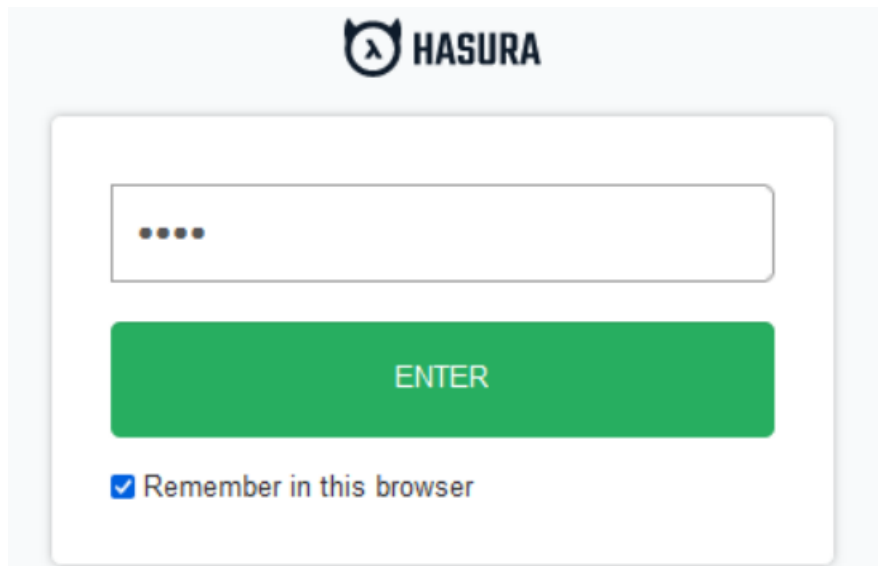
1 version: '3.3'
2 services:
3   graphql-engine:
4     image: hasura/graphql-engine:v2.8.2
5     environment:
6       HASURA_GRAPHQL_ADMIN_SECRET: perc
7       HASURA_GRAPHQL_DATABASE_URL: postgres://postgres:postgrespassword@172.24.178.122:5432/lab_telemetria_db
8       HASURA_GRAPHQL_DEV_MODE: 'true'
9       HASURA_GRAPHQL_ENABLED_LOG_TYPES: startup, http-log, webhook-log, websocket-log,
10        query-log
11       HASURA_GRAPHQL_ENABLE_CONSOLE: 'true'
12       HASURA_GRAPHQL_ENABLE_TELEMETRY: 'true'
13       HASURA_GRAPHQL_METADATA_DATABASE_URL: postgres://postgres:postgrespassword@172.24.178.122:5432/postgres
14     ports:
15       - 8080:8080
16     networks:
17       - swarnpit_net
18     logging:
19       driver: json-file
20 networks:
21   swarnpit_net:
22     driver: overlay
23

```

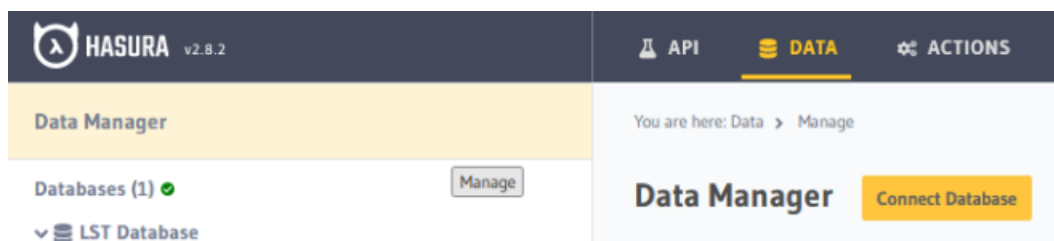
 DEPLOY
 DELETE

En cuanto el contenedor esté levantado, deberá acceder a la interfaz de hasura (<http://localhost:8080>) para conectar la base de datos antes creada.

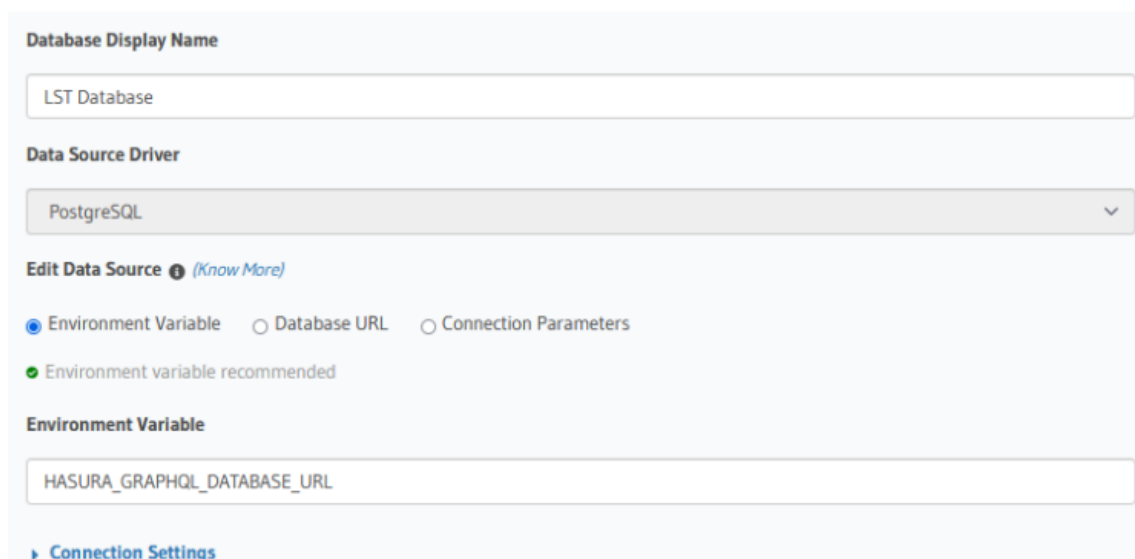
La primera vez le solicitará ingresar la clave de administrador, esta clave es la que se configuró en la variable **HASURA_GRAPHQL_ADMIN_SECRET**.



En la interfaz mostrada diríjase a la pestaña de Data, y luego presione el botón de Connect Database:



Llenar los datos del formulario como se muestra en la siguiente imagen:

A screenshot of the Hasura Data Source configuration form. The "Database Display Name" field is filled with "LST Database". The "Data Source Driver" dropdown menu is set to "PostgreSQL". Below this, there are three radio buttons: "Environment Variable" (which is selected), "Database URL", and "Connection Parameters". A green checkmark indicates that "Environment variable recommended" is selected. The "Environment Variable" field is filled with "HASURA_GRAPHQL_DATABASE_URL". At the bottom, there is a link for "Connection Settings".

Una vez conectada la base de datos, debe tracker las tablas para ello de clic al esquema público de la base de datos agregada:


Databases (1) 

Manage

▼  LST Database


▼  public

Se mostrará la lista de tablas del esquema, deberá presionar el botón de Track All:

▼ **Untracked tables or views**  [Track All](#)

Track	entidades
Track	equip
Track	etiquetas
Track	haystack_tags
Track	point
Track	registros_sensores
Track	roles
Track	site
Track	usuarios

Finalmente, se mostrarán las relaciones existentes entre las claves foráneas de cada tabla las cuales deberá de igual forma trackear

▼ **Untracked foreign-key relationships**  [\(Know more\)](#) [Track All](#)

Track	registros_sensores → point	- registros_sensores . point_id → point . id
Track	entidades → [etiquetas]	- etiquetas . entidad_id → entidades . id
Track	equip → site	- equip . siteRef → site . id
Track	equip → [point]	- point . equipRef → equip . id
Track	etiquetas → entidades	- etiquetas . entidad_id → entidades . id
Track	etiquetas → haystack_tags	- etiquetas . tag → haystack_tags . tag
Track	haystack_tags → [etiquetas]	- etiquetas . tag → haystack_tags . tag
Track	point → equip	- point . equipRef → equip . id
Track	point → site	- point . siteRef → site . id
Track	point → [registros_sensores]	- registros_sensores . point_id → point . id
Track	site → [equip]	- equip . siteRef → site . id
Track	site → [point]	- point . siteRef → site . id

Api de Estandarización

El archivo de configuración para el api se encuentra en la carpeta “[api-standardization](#)” del [Repositorio: Plantillas de Docker-Compose para SIMIoT](#)

Las variables de entorno requeridas son:

HASURA_SECRET: perc

HASURA_URL: http://[IP-DOMAIN]:8080/v1/graphql

Donde [IP-DOMAIN] es la IP de la máquina donde se encuentra desplegado el contenedor de Hasura, o en caso de tenerlo, del dominio, el cuál puede conseguir al ejecutar el comando **ifconfig eth0**.

```
keyla@LAPTOP-N4M8K800:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.24.178.122  netmask 255.255.240.0  broadcast 172.24.191.255
    inet6 fe80::215:5dff:fe96:e652  prefixlen 64  scopeid 0x20<link>
    ether 00:15:5d:96:e6:52  txqueuelen 1000  (Ethernet)
```

Luego presiona DEPLOY.


Name *
api-standardization

Compose file
Current engine state

Compose file source

```

1 version: '3.3'
2 services:
3   api-standardization:
4     image: api-standardization:latest
5     environment:
6       HASURA_SECRET: perc
7       HASURA_URL: http://192.168.100.46:8090/v1/graphql
8     ports:
9       - 8082:8082
10    networks:
11      - swarmpit_net
12    logging:
13      driver: json-file
14 networks:
15   swarmpit_net:
16     driver: overlay
17
```

 **DEPLOY**

Sistema de Registro de Módulos IoT

El archivo de configuración para este sistema lo encontrará en la carpeta “[iot-module-register](#)” del Repositorio: [Repositorio: Plantillas de Docker-Compose para SIMIoT](#)

Las variables de entorno requeridas son:

VITE_API_STANDARDIZATION: http://[IP-DOMAIN]:8082/v1

VITE_HASURA_ENDPOINT: http://[IP-DOMAIN]:8080/v1/graphql

VITE_HASURA_SECRET: perc

Donde [IP-DOMAIN] es la IP de la máquina donde se encuentra desplegado el Swarmpit, o en caso de tenerlo, del dominio de su pc, el cuál puede conseguir al ejecutar el comando **ifconfig eth0**.

```
keyla@LAPTOP-N4M8K800:~$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.24.178.122  netmask 255.255.240.0  broadcast 172.24.191.255
    inet6 fe80::215:5dff:fe96:e652  prefixlen 64  scopeid 0x20<link>
    ether 00:15:5d:96:e6:52  txqueuelen 1000  (Ethernet)
```

Luego presiona DEPLOY

Name *

iot-module-register

Compose file


Current engine state

Compose file source






```

1 version: '3.3'
2 services:
3   register-module:
4     image: iot-module-register:latest
5     environment:
6       VITE_API_STANDARDIZATION: http://192.168.100.46:8082/v1
7       VITE_HASURA_ENDPOINT: http://192.168.100.46:8090/v1/graphql
8       VITE_HASURA_SECRET: perc
9     ports:
10      - 3000:3000
11     networks:
12      - swarmpit_net
13     logging:
14       driver: json-file
15 networks:
16   swarmpit_net:
17     driver: overlay
18

```

 **DEPLOY**

Al fina así debe verse el stack completo de swarmpit:

Total (5)						 NEW STACK
Name	Services	Networks	Volumes	Configs	Secrets	
api-standardization	1	1	0	0	0	
iot-module-register	1	1	0	0	0	
postgres-db	1	1	1	0	0	
pr-hasura	1	1	0	0	0	
swarmpit	4	1	2	0	0	