Odoo on Docker

Notar que para correr Docker antes debo arrancar el servicio SSH.

VS Code tiene una extensión para Docker, que es muy completa. Y permite mirar el file system interno de cada docker, con lo cual puedo aprender cómo está hecho.

De hecho, puedo cerrar Docker Desktop, y en VS Code tengo las mismas funcionalidades, o más.

# odoo-docker-compose-datos-demo

<https://youtu.be/gIp28l7YND4>

<https://github.com/docker-compose-marlon>

<https://github.com/docker-compose-marlon/odoo-docker-compose>

En Docker Desktop elimino todos los containers, las imágenes y los volúmenes.

En Git Bash, navegar hasta la carpeta C:\GIT. Pegar los comandos:

git clone <https://github.com/docker-compose-marlon/odoo-docker-compose>

El repositorio se clona en C:\GIT\40-docker\odoo-docker-compose. Cambiarle el nombre a odoo-docker-compose-datos-demo.

cd odoo-docker-compose-datos-demo

docker-compose up -d

Crea dos imágenes, odoo y postgres.

Crea tres volúmenes.

Arranca una network de containers:

* odoo-docker-compose.

Dentro hay dos containers corriendo:

* odoo-docker-compose\_db\_1,
* odoo-docker-compose\_web\_1.

Corre en: <http://localhost:8069/web/database/selector>

Datos de la figura.

ghionE6421

ghione

gustavo.garcia@activalogic.com

ghionE6421

+543462635499

Con las opciones anteriores, Create database. Tarda unos momentos. Aparecen un montón de módulos.

Con los datos demo, el dueño es Michell Admin.

# odoo-docker-compose

Me cambio a este repositorio.

Edito docker-compose.yml y cambio 8069 por 8070.

Repito los pasos, pero sin datos demo.

No crea imágenes nuevas, parece que deja las mismas dos. Sí crea tres volúmenes nuevos.

Y abre una network nueva con dos containers nuevos dentro.

Corre en: <http://localhost:8070/web/database/selector>

# Odoo en Docker en mi PC

<https://www.how2shout.com/linux/how-to-install-odoo-13-or-14-on-docker-container/>

Yo tengo Docker Desktop instalado en mi PC.

Lo arranco desde el menú de inicio.

Verifico en PowerShell:

docker --version

Docker version 20.10.16, build aa7e414

docker pull odoo:latest

Lo descarga y me da alguna información.

docker pull postgres

Lo descarga y me da alguna información.

docker run -d -v odoo-db:/var/lib/postgresql/data -e POSTGRES\_USER=odoo -e POSTGRES\_PASSWORD=odoo -e POSTGRES\_DB=postgres --name db postgres

Crea la base de datos.

docker run -v odoo-data:/var/lib/odoo -d -p 8069:8069 --name odoo --link db:db -t odoo

Arranca el container. Todo lo anterior lo hice en Windows PowerShell.

Ahora me voy a Docker Desktop, y ahí veo dos containers in use: odoo y postgres, que son los dos anteriores.

<https://hub.docker.com/_/odoo/>

version: '3.9'

services:

web:

image: odoo

depends\_on:

- db

ports:

- "8069:8069"

db:

image: postgres

environment:

- POSTGRES\_DB=postgres

- POSTGRES\_PASSWORD=odoo

- POSTGRES\_USER=odoo

# How to Deploy a Docker App to AWS ECS

<https://youtu.be/YDNSItBN15w>

24 minutos

Superclaro, me encantó.

Muestra como dockerizar una aplicación y desplegarla a Elastic Container Service (ECS) de AWS.

Hay que saber lo básico de Docker.

Vamos a dockerizar una API Express. Esta API, cuando la accedemos por medio de un endpoint, nos retorna unos datos.

Express está construido sobre Node. No se necesita tener Node instalado en la PC local, justamente porque estamos trabajando con Docker, y el container estará aislado de nuestra máquina local.

La aplicación Express está en la carpeta \40-docker\NodeDockerAWS.

La aplicación tiene solo un par de archivos: index.js y package.json. Eso es todo.

Con ambos archivos en una carpeta, y aunque lo estemos viendo en VS Code, el próximo paso es dockerizar la aplicación, no correrla con Angular. Si uno intenta correrla, le va a dar un error. Hay que dockerizar.

Para eso, hay que crear un Dockerfile.

Con eso vamos a crear una imagen Docker para nuestra aplicación Express.

Arrancar Docker Desktop, porque se necesita el Docker Daemon.

Abrir nueva terminal.

docker build -t express-app .

Ahora se puede ver en la extensión de Docker de VS Code, o en Docker Desktop, que se ha creado una nueva imagen llamada express-app. Todavía no hay ningún container corriendo, ni apareció ningún volumen. Alternativamente, podemos escribir en la terminal el comando:

docker images

Copiamos el ID de la imagen que acabamos de crear. Pegamos en la terminal el comando:

docker run -p 6565:5000 ID-que-copiamos

Este comando especifica que estamos mapeando el puerto 6565 de nuestro localhost al puerto 5000 del container, que es el que le dijimos que queríamos exponer. En lugar de 6565, el puerto podría cualquiera que queramos.

Esto mismo se puede hacer desde Docker Desktop. Es bastante intuitivo. La aplicación corre en:

<http://localhost:6565/>

Ahora que sabemos que nuestra aplicación está corriendo, queremos desplegarla en la nube.

## Instalar y configurar AWS CLI

Para lo que sigue, es necesario tener instalado y configurado AWS CLI. El autor no lo dice ahora, sino después. Por ese motivo, yo he intercalado un tutorial que se ocupa de este asunto.

<https://youtu.be/jCHOsMPbcV0>

Después de instalar AWS CLI no hay que bootear, pero sí hay que reiniciar la terminal de Windows, o VS Code, para que reconozca el comando.

Necesito un usuario en mi cuenta de AWS. Voy a IAM, Identity and Access Management:

<https://us-east-1.console.aws.amazon.com/iamv2/home?region=us-east-1#/users>

Las credenciales solo están disponibles para descargar en el momento de la creación. Después, ya no lo estarán. Sin embargo, uno puede crear nuevas credenciales en cualquier momento. En la terminal:

aws configure

Donde me pide las claves le pego las que acabo de copiar, y listo.

Una vez que tengamos AWS CLI instalado y configurado, el resto de este tutorial intercalado no es necesario.

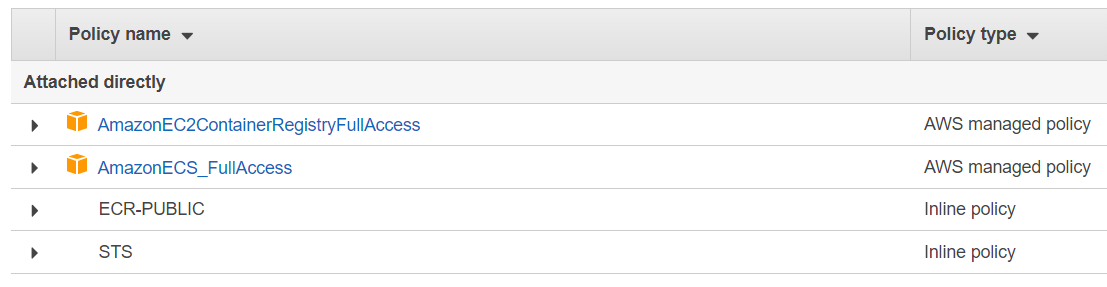
Volvemos al tutorial anterior.

Me loggeo en la consola de mi cuenta de AWS. Voy a Elastic Container Service:

<https://us-east-1.console.aws.amazon.com/ecs/home?region=us-east-1#/getStarted>

Tengo que subir la imagen que está en mi PC a la nube. Para eso voy a usar ECR, Elastic Container Registry:

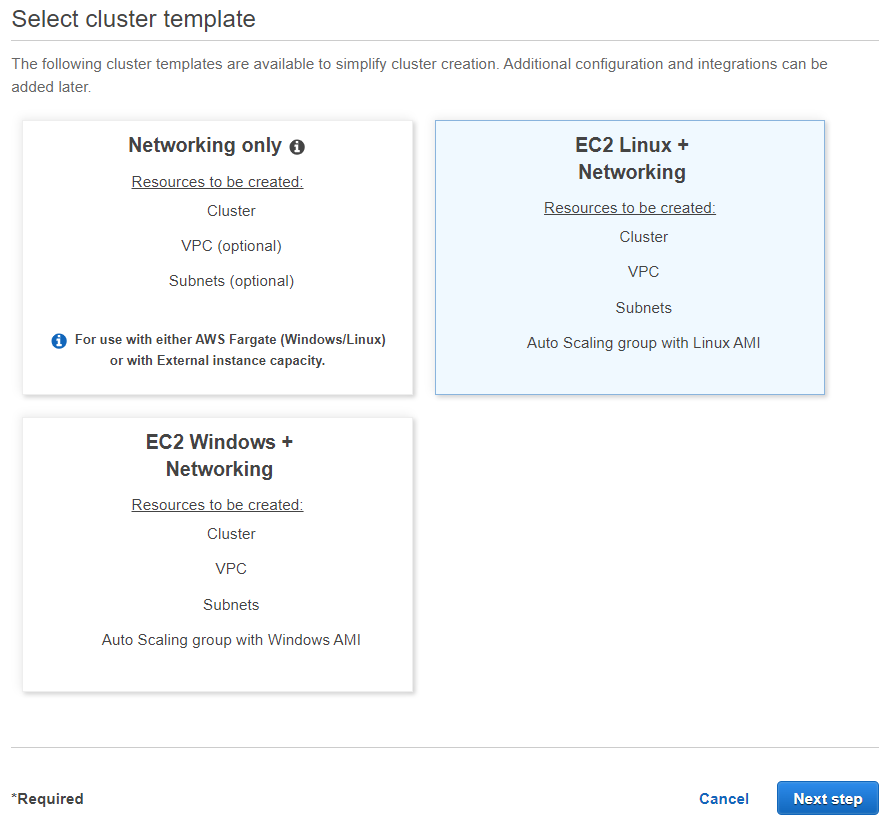
<https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1>

Me creo un repositorio público. Le pongo el mismo nombre de mi aplicación (obviamente...). Una vez creado, ese repositorio está vacío, naturalmente. Al repositorio le tengo que poner, además de permissions, policies. Yes, policies . En mi caso, le tuve que poner las que se ven en la figura. Supe cuáles eran, leyendo los mensajes de error que me daba al intentar loggearme sin éxito. Cuando le puse las policies que quería, se loggeó.

En todo caso, voy a los repositorios:

<https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1>

clickeo mi repositorio recién creado, y clickeo View push commands. Lo primero que aparace el el comando de login. Lo copio, lo pego en la terminal, si me pide policies las agrego, y finalmente me loggeo. Tarda unos segundos, no es inmediato.

Si no tenemos buildeada la imagen, hay que hacerlo.

Seguir las instrucciones. Los comandos se pueden pegar en cualquier terminal, también en la VS Code.

Cerramos el diálogo anterior, el que tiene las instrucciones. Volvemos a nuestro repositorio. Hay que refrescar, y entonces aparece la imagen que acabamos de pushear.

Me copio la URL de la imagen.

Notar que lo único que tenemos, hasta ahora, es un repositorio, y una imagen subida a ese repositorio. No hay nada más. En particular, no tenemos un container corriendo.

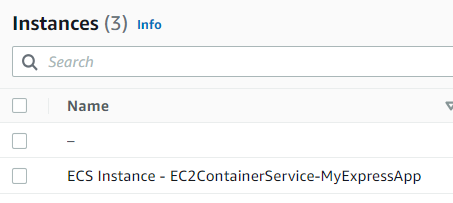
Para correr la aplicación necesitamos alguna clase de server.

Nos vamos a ECS, vamos a Clusters de esta opción. Ojo que hay otros clusters.

Vamos a Create Cluster.

Seleccionamo el que está en celeste.

Next Step.

Seguimos las instrucciones de tutorial, que son muy claras.

Cuando le damos Create, tarda unos momentos.

Vamos a EC2 > Instances.

Deberíamos ver algo como lo que se muestra en la figura.

Vamos de nuevo a ECS.

Vemos los clusters.

Ahora tenemos que run a task. Vamos a Task Definitions.

Vamos a Create new Task Definition. Elegimos EC2. Next step.

Seguir las instrucciones del tutorial. Una vez más, tengo que decir que este tipo es muy claro. Capo total.

Vamos a Add Container.

BIG PROBLEM CON LA INSTANCIA

# Docker en la nube

<https://youtu.be/xrBgkxByRQg>

El sonido es abominable, y el tío es casi tan español como se puede ser. Pero está bastante interesante. Es onda Linux.

# How to Deploy a Docker App to AWS using Elastic Container Service (ECS)

<https://youtu.be/zs3tyVgiBQQ>

15 minutos

El tipo va a mil. Creo que lo voy a tener que mirar varias veces.