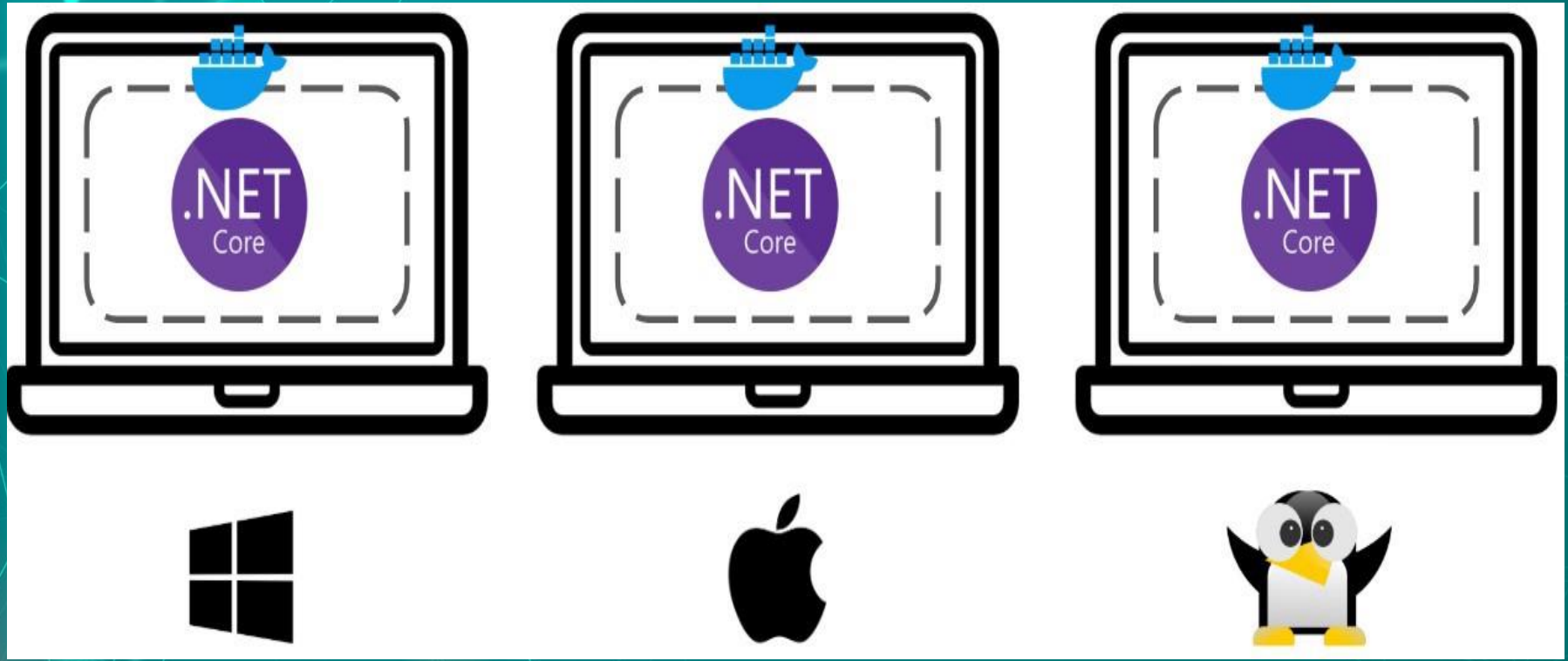
Docker

¿ Qué es Docker ?

1. Docker es una plataforma que nos permite simplificar procesos para construir aplicaciones, distribuirlas y además nos permite ejecutaras en distintos entornos y sistemas.
2. Con entornos me refiero a entorno o servidor de desarrollo, preproducción y producción.
3.
4. Pero no únicamente el entorno puede variar, sino que además el sistema del propio entorno, es común tener nuestro ordenador de desarrollo con windows o mac, mientras que los servidores suelen ser en su gran mayoría linux, así como AWS, google cloud, incluso azure corre su propia versión de linux.


.NET
Core
.NET
Core
.NET
Core

¿ Cómo funciona Docker?

Cuando hablamos de Docker realmente hablamos de dos conceptos, **Imágenes**, y **contenedores**, estos dos conceptos es en los que se basa Docker.
Si te has fijado en el logo de Docker, la ballena tiene varios contenedores encima, y esto es así por el siguiente motivo.

El poder de los contenedores reside en que nos ayudan a hacer despliegues de una forma mucho más sencilla.
Si pensamos hace unos años para hacer un despliegue de una aplicación grande, consumimos mucha cantidad de tiempo, había que ir a cierta carpeta en X servidor borrarla a mano, luego mover los ficheros, etc. Por supuesto este tipo de acciones se podían hacer con scripts, pero no todas las empresas utilizan scripts.
Básicamente cada empresa y cada desarrollador lo hace de una forma diferente.

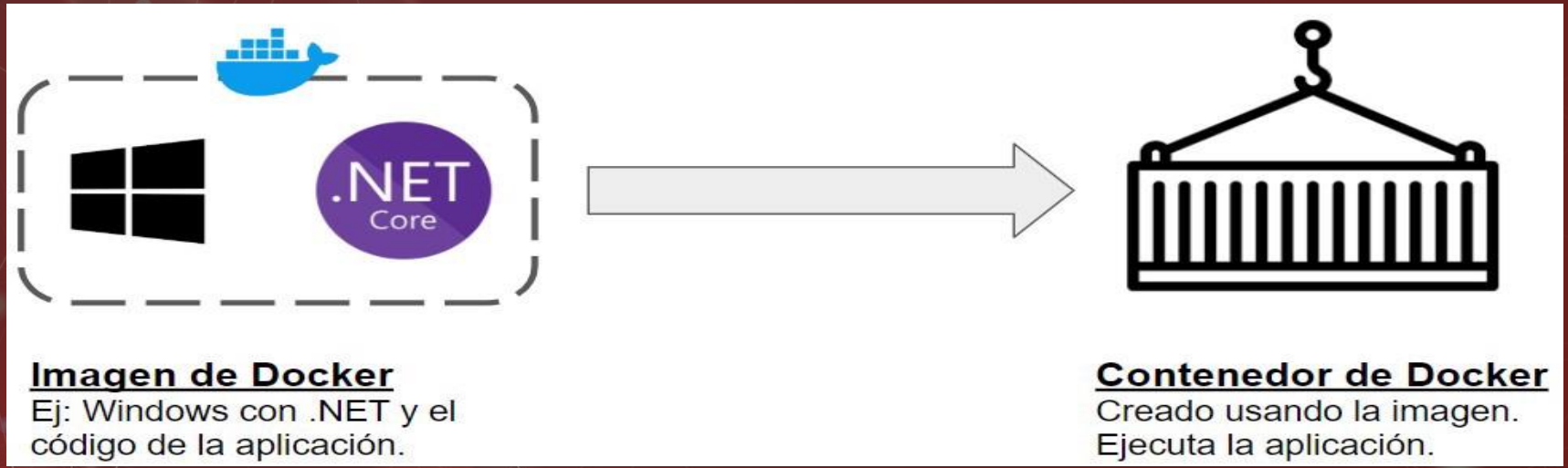
Docker nos proporciona una técnica consistente a la hora de hacer estos despliegues en diferentes entornos, lo cual nos proporciona un gran número de beneficios para nosotros los desarrolladores.

¿ Qué es una imagen de Docker?

1. Una imagen contendrá los ficheros necesarios para correr la aplicación que queremos correr en un servidor en concreto. Por ejemplo ubuntu o windows.
2. Después tenemos nuestro framework, o la base de datos, los ficheros, etc.
3.
4. Docker no descarga todas las librerías del sistema operativo, si no que es una forma de indicar sobre que servidor queremos que se ejecute. Para nuestro caso en concreto únicamente se descargarán las librerías de Net Core.
5.
6. La imagen como tal por sí sola no hace nada, sino que es la definición que es utilizada para conseguir un contenedor que se ejecute.

¿ Qué es un contenedor de Docker?

1. Si nos basamos en el punto anterior, podemos ver que un contenedor es la instancia de la imagen, donde la aplicación está corriendo.
2. .



Puede ser cualquier tipo de aplicación, desde nuestra propia aplicación web, como un servidor de caché, o una base de datos.

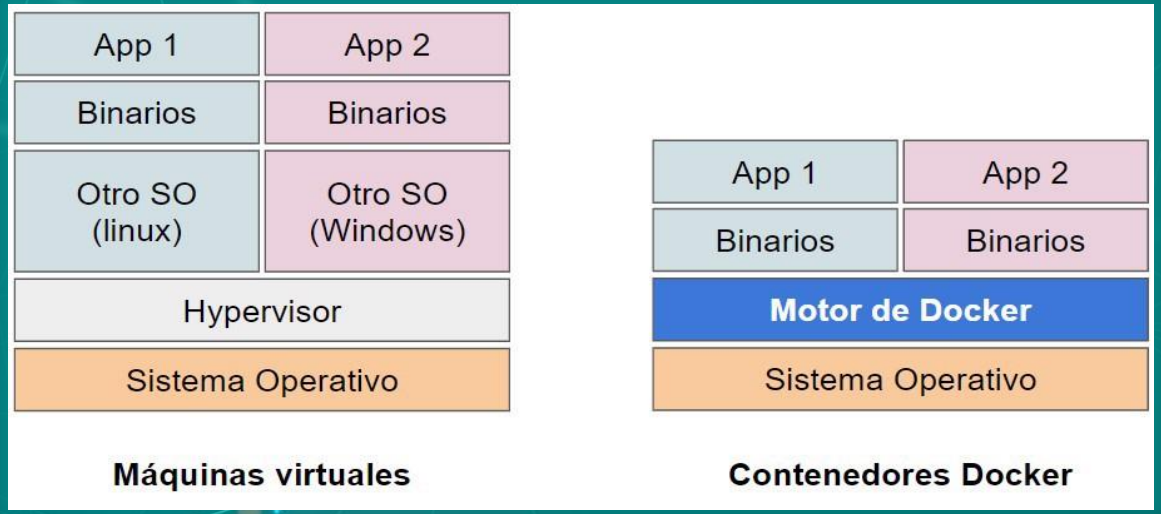
El gran beneficio de utilizar contenedores es que cada contenedor está completamente aislado del resto de contenedores, lo que implica que cada uno puede ser creado, instanciado, parado, o eliminado individualmente.
Otro de los grandes beneficios es que es muy rápido y sencillo crear un contenedor.

Diferencia entre un contenedor Docker y una máquina virtual

1. Cuando empezamos a desarrollar en Docker, lo primero que nos viene a la mente son las máquinas virtuales, que hemos tratado con ellas durante nuestros estudios en la universidad o en los grados, porque visto el punto anterior, parece lo mismo, es una máquina, que contiene un sistema operativo y una aplicación dentro.
2.
3. Lo primero es que una máquina virtual corre en nuestro sistema operativo como "invitado" a través del *Hypervisor*, además correr múltiples máquinas virtuales, depende del tamaño de nuestro disco duro, o nuestra memoria puede ser una locura. Ya que funcionan, pero necesitas cierta cantidad importante de ram, procesador, y por supuesto unos cuantos gigas en el disco duro, etc.
4.

Diferencia entre un contenedor Docker y una máquina virtual

1. Por el contrario los contenedores de Docker corren sobre el motor de Docker, el cual se ejecuta directamente sobre el sistema operativo. Lo que implica que el tamaño es mucho mas pequeño que una máquina virtual normal.



01 Un contenedor Docker, contiene por sí solo

02 todos los ficheros de sistema que necesita la aplicación

03 que va a ejecutar para que esta corra sin problemas

04 pero a su vez, no contiene ninguno adicional que no necesite

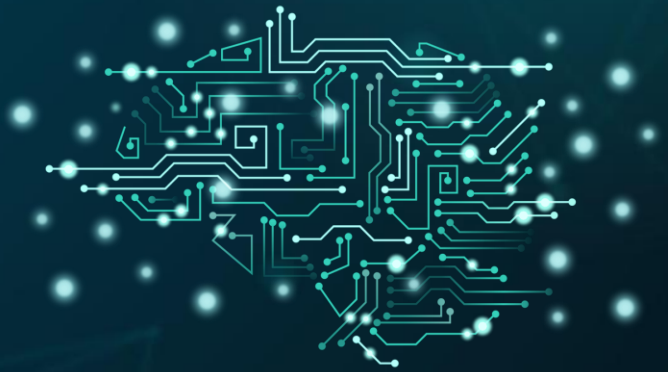
Beneficios de Docker

01

El primer beneficio y el más claro de todos es que mantiene el entorno original limpio. Pero que quiero decir con esto.

1. Por ejemplo para ejecutar un aplicación que contiene una aplicación web escrita en **node js**, una aplicación de consola escrita en **.net** y ambas se conectan con una base de datos **mysql**.
2.
3. En circunstancias normales, deberíamos instalar .net en el sistema operativo, junto con Docker, junto con la base de datos; Pero si utilizamos Docker, lo único que necesitamos es un sistema capaz de ejecutar Docker, y crearemos un contenedor que contenga .NET con la aplicación de consola de .NET. Un contenedor con node para la aplicación web, y un contenedor con la base de datos.
4.
5. Debido a que cada contenedor se ejecuta de una forma independiente, cualquier usuario que ejecute el contenedor o si lo ejecutamos en un servidor recibirá el mismo resultado, con lo que el problema de “en mi ordenador funciona” ya no nos pasará.
6.
7. Incluso en cantidad de ocasiones, como puede ser con amazon web services, desarrollaremos en nuestra máquina local en Visual Studio, pero a la hora de hacer un despliegue lo haremos sobre ECS que es básicamente Docker.

Docker Compose



Qué es docker compose?

1. para ejecutar nuestra aplicación en docker creamos un fichero llamado Dockerfile y este fichero contiene diferente configuración, esta configuración varía dependiendo de qué queremos poner en el contenedor, ya que no es lo mismo poner una página web, que una base de datos.
2.
3. Este proceso, de crear todos los Dockerfile y ejecutarlos puede ser bastante tedioso, ya que debemos pensar que una aplicación de tamaño mediano es probable que tenga un front end, un back end, quizá background-workers así como la base de datos, sistema de caché, sistema de colas o de message-broker, etc, cada uno de nuestros servicios será un contenedor diferente.
4.
5.

Qué es docker compose?

1. Aquí es donde entra **docker copose** el cual es una herramienta que nos permite definir y correr múltiples contenedores en Docker, estos múltiples contenedores se definen en un fichero denominado **docker-compose** con la extensión **.yml**.
2.
3. Por lo tanto Docker compose es una herramienta de orquestación de contenedores.

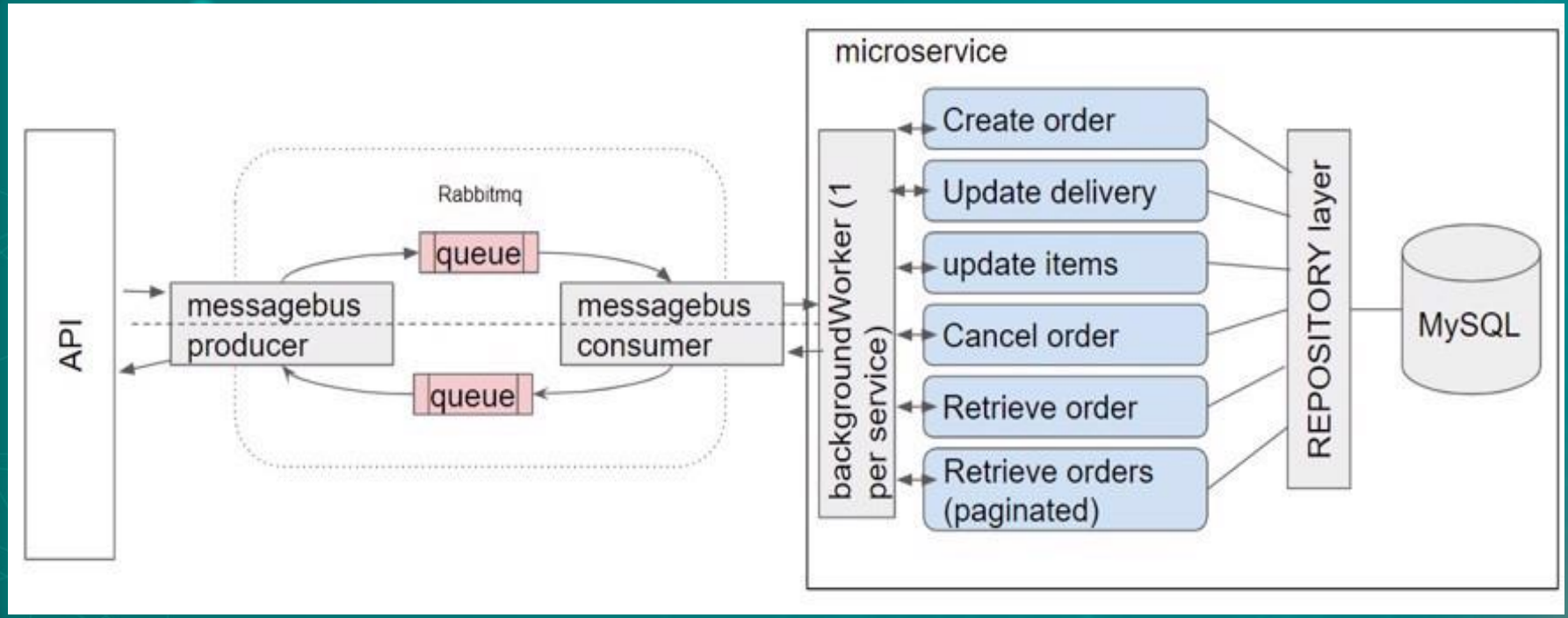
Configuración de un fichero docker-compose

Para poder crear estos contenedores debemos hacerlo en nuestro fichero docker-compose.yml por norma general en los proyectos se suele poner en la raíz del mismo, y yo personalmente siempre lo pongo ahí.

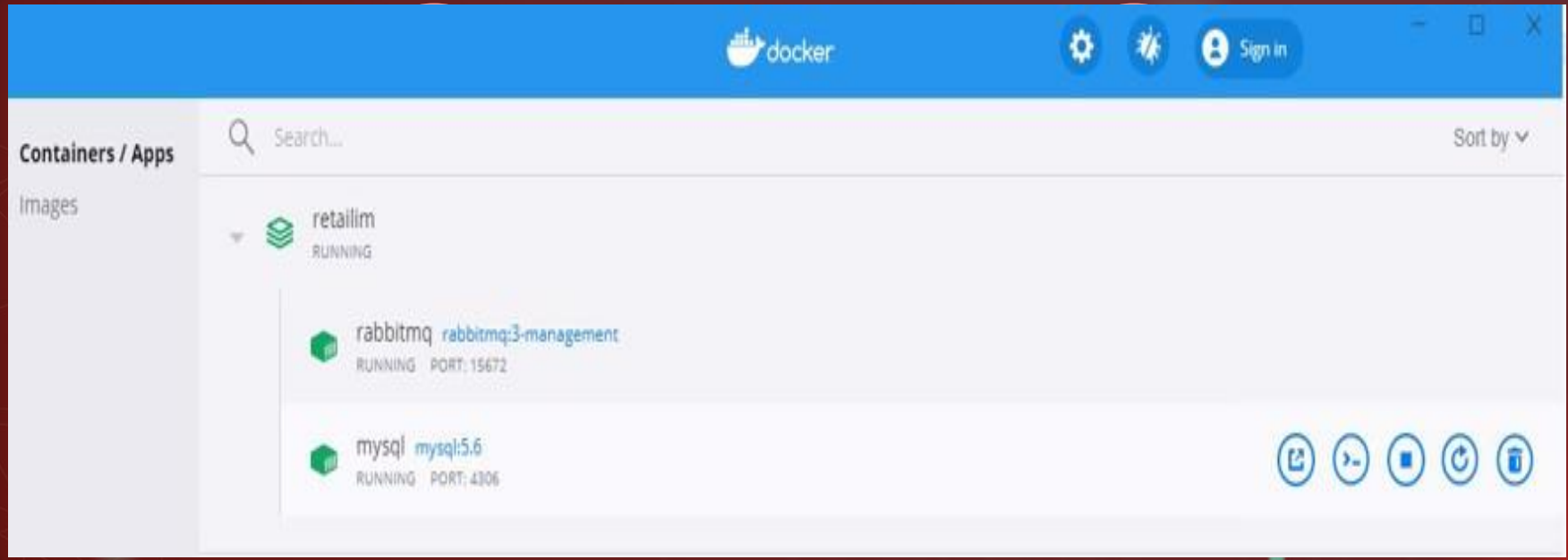
1. El fichero consta de varias partes:
2. **version**: Debemos indicar que versión de docker-compose vamos a utilizar.

    a. Por ejemplo es importante tener en cuenta que hoy en dia para relacionar contenedores se utiliza la configuración links pero anteriormente había que crear una red en la que todos los contenedores estuvieran en la misma red.
3. **services**: Aquí es donde indicaremos los diferentes servicios que vamos a utilizar y cada uno de ellos será convertido en un contenedor.
4. **volumes**: Podemos definir el apartado volumes el cual quiere decir que la información del contenedor se preservara cuando destruyamos el mismo, así la siguiente vez que lo creemos, toda esa información permanecerá ahí (por ejemplo información de una base de datos).
5.
6.
7. Aquí podemos ver un fichero **docker-compose.yml** el cual contiene
8. MySql (base de datos)
9. RabbitMQ (message-broker)
10. API en .net
11. Backgroundworker en .net
12.
13.
14.
15. ver archivo yml (codigo)
16.

Contexto de una aplicación en producción



Contexto de una aplicación en producción



Cuando utilizar Docker Compose

Entorno de test producción

1

Un caso de uso podría ser para desplegar todo nuestro entorno

2

en un servidor de test o en producción,

3

- pero en aplicaciones empresariales serias esto no lo vas a ver jamás.

hay mejores alternativas, como puede ser kubernetes