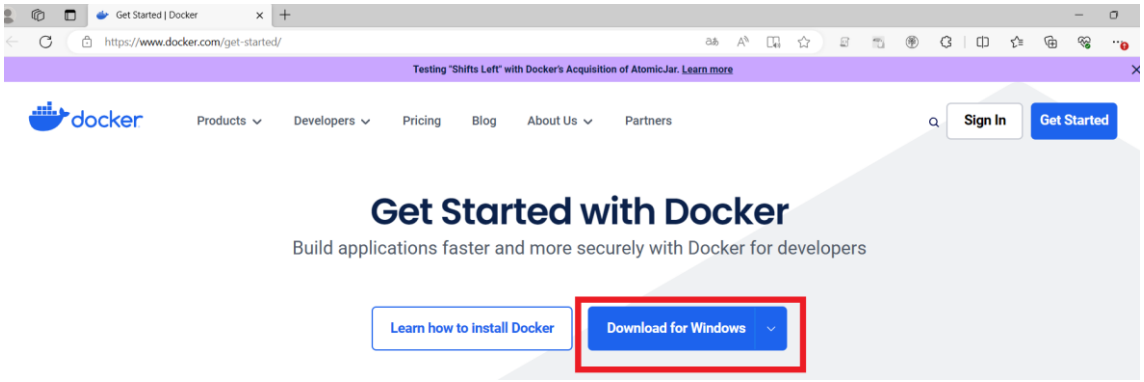
Instalación y primeros pasos con Docker

Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente, empaquetando software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute (bibliotecas, herramientas del sistema, código, etc). La idea de Docker es crear contenedores ligeros y portables para que las aplicaciones software puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina, facilitando así también los despliegues.

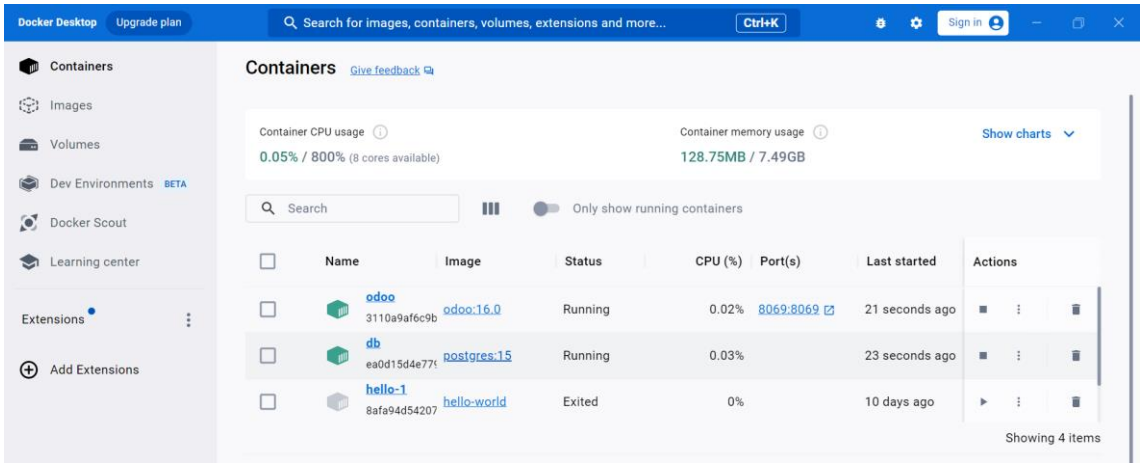
Docker es un sistema operativo para contenedores que proporciona comandos sencillos para crear, iniciar o detener contenedores. De manera similar a cómo una máquina virtual virtualiza el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.

Para poder utilizar los contenedores con las aplicaciones es necesario descargar e instalar Docker desde su página oficial en el apartado Developers > Getting

Started (https://www.docker.com/get-started).



Una vez instalado Docker Desktop, se podrá abrir y ver su dashboard.



Algunas de las vistas más importantes del dashboard son:

• **Contenedores**: proporciona una vista en tiempo de ejecución de todos sus contenedores para realizar acciones comunes como inspeccionar, interactuar y administrar dichos contenedores directamente desde la máquina.

Imágenes: muestra una lista con las imágenes de Docker descargadas y permite ejecutarlas como contenedor sin tener que volver a descargarla.

Además, contiene opciones de limpieza para eliminar imágenes no deseadas del disco y recuperar espacio.

• **Volúmenes**: muestra una lista de volúmenes creados y ver cuáles se están utilizando. Además, permite crear y eliminar volúmenes fácilmente.

Para comprobar que Docker está funcionando correctamente, desde la línea de comando, usamos el comando

```
$ docker version
```

Comandos de Docker para gestionar los contenedores

El primero de ellos, siendo un clásico en el mundo de la informática, es crear el contenedor Hola_Mundo.

```
$ docker run hello-world
```

Este comando lo que hace es crear un contenedor, comprobando antes si existe o no la imagen hello-world:latest (última versión) en el ordenador y en el caso de que no exista la imagen, se la descargará del repositorio de Docker. Este contenedor lo único que hace es mostrar el mensaje “Hello from Docker” y se detiene.

Si volvemos al dashboard se puede observar que en la vista Contenedores se ha creado el contenedor hello-world con el nombre que Docker le pone de forma aleatoria si no se le especifica uno en concreto. En la vista Imágenes, se encuentra la imagen hello-world, de forma que si volvemos a crear otro contenedor hello-world, ya no se volverá a descargar la imagen, siendo la creación del contenedor mucho más rápida.

Otros comandos que pueden ser útiles para la gestión de Docker son:

Mostrar el listado de contenedores:

```
$ docker ps -a
```

O bien:

```
$ docker container ls –a
```

Crear un contenedor con un nombre específico

```
$ docker run -d --name=hello-1 hello-world
```

Arrancar un contenedor parado

```
$ docker start <NOMBRE O ID DEL CONTENEDOR>
```

Parar un contenedor

```
$ docker stop <NOMBRE O ID DEL CONTENEDOR>
```

Borrar un contenedor

```
$ docker rm <NOMBRE O ID DEL CONTENEDOR>
```

Mostrar el listado de imágenes

```
$ docker image ls o $ docker images
```

Borrar una imagen

```
$ docker rmi IMAGEN o $ docker image rm IMAGEN
```

Instalación de Odoo en Docker

Una vez vistos los conceptos básicos de Docker, ya se puede empezar con la instalación de Odoo.

Como ya se ha visto en las instalaciones anteriores, Odoo utiliza postgreSQL como base de datos, que se instalaba en la misma máquina que Odoo. En este caso, tanto Odoo como postgreSQL van a estar en contenedores diferentes, aunque tendrán que estar conectados, ya que sin una base de datos de postgreSQL, Odoo no podría funcionar.

En primer lugar, vamos a ir a la página web https://hub.docker.com/_/odoo/ donde se encuentra la imagen oficial de Odoo con todas las versiones disponibles que podemos descargar e instalar en un contenedor.

Nota: Docker Hub es el repositorio de imágenes de contenedores proporcionado por Docker, para buscar y compartir imágenes. Todos los contenedores suelen ser bastante configurables por lo que es recomendable ver los parámetros que se pueden utilizar para configurar el contenedor. Al haber gran variedad de imágenes, es importante descargar aquellas que sean oficiales o exista una gran confianza en el creador de la imagen para evitar problemas como la ejecución de programas que nada tiene que ver con el propósito de la imagen.

Pero antes de instalar Odoo, se va a crear un contenedor de postgreSQL, ya que seguidamente, al crear el contenedor de Odoo, habrá que indicarle algunos parámetros para conectar Odoo con postgreSQL.

Al igual que con las imágenes de Odoo, si se accede a la página web https://hub.docker.com/_/postgres vemos todas las imágenes disponibles de postgreSQL con sus respectivas versiones.

Para crear el contenedor de postgreSQL se utiliza el siguiente comando

```
docker run -d -e POSTGRES_USER=odoo -e
POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db
postgres:15
```

Al ejecutar el comando, intentará buscar la imagen de postgres en local y al no encontrarla, la descargará del repositorio para poder crear el contenedor. Si observamos el dashboard o utilizamos los comandos docker ps -a y docker image ls se puede observar que se ha descargado la imagen y que hay un contenedor llamado bd con la última versión de postgreSQL corriendo.

El siguiente paso es crear el contenedor de Odoo que enlace con el contenedor de postgreSQL con el siguiente comando

```
$ docker run -p 8069:8069 --name odoo --link db:db -t
odoo:16.0
```

Una vez descargada la imagen y creado el contenedor de Odoo, en la línea de comandos se puede observar que el servicio de Odoo está arrancado y muestra información de la versión y de las diferentes rutas de configuración.

Ya se podría acceder a Odoo poniendo la url http://localhost:8069 en un navegador web. Pero, aunque a simple vista parece que todo funciona de forma correcta, ¿qué pasaría si se eliminarán los contenedores? La respuesta es sencilla, se eliminaría toda la información. Para evitar ese problema, se debe tener persistencia de datos, es decir, crear volúmenes para que, en el caso de eliminar algún contenedor, la información no se pierda.

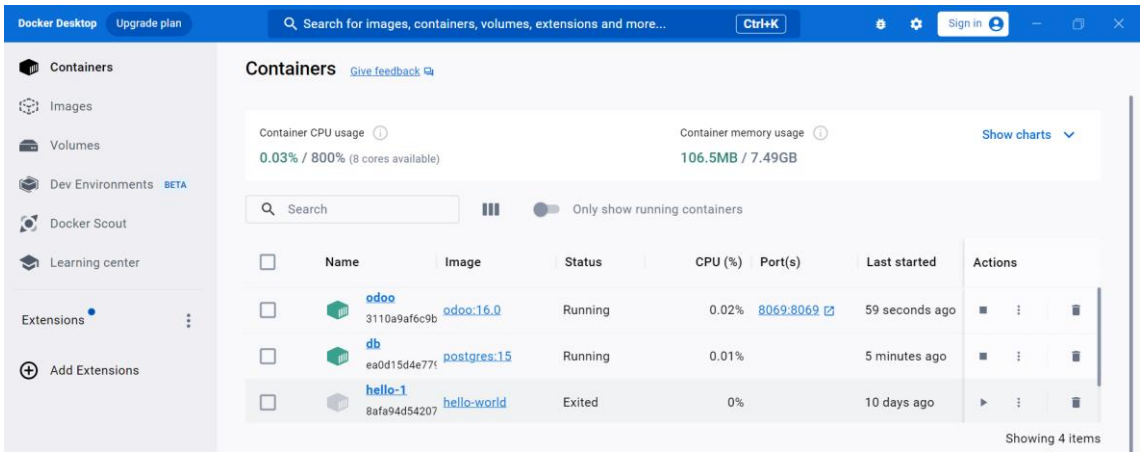
Para crear la persistencia de datos, primero vamos a eliminar desde el dashboard o por línea de comandos los dos contenedores que se han creado anteriormente, ya que no tienen persistencia de datos. Una vez eliminados, ya se pueden volver a crear los contenedores con los siguientes comandos

```
$ 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:15
```

La ruta /var/lib/postgresql/data es donde se encuentran las bases de datos de postgreSQL en el sistema operativo del contenedor.

```
$ docker run -v odoo-data:/var/lib/odoo -d -p 8069:8069 -
-name odoo --link db:db -t odoo:16.0
```

Si volvemos el dashboard se puede observar que se han creado dos volúmenes, uno para postgreSQL y otro para Odoo:



De esta forma, si accedemos a Odoo con la url http://localhost:8069 y realizamos la configuración inicial de la base de datos e instalamos una serie de módulos, aunque se eliminen los contenedores, al volverlos a crear enlazándolos con los volúmenes, seguiremos teniendo toda la información. Para ello, se propone a modo de prueba, volver a eliminar los contenedores de postgreSQL y Odoo y volverlos a crear para ver que la configuración anterior no se ha perdido.

Durante el proceso de creación de los contenedores, Odoo y postgreSQL se instalan para que se ejecuten al iniciarse los contenedores, por lo que no será necesario lanzar los servicios de forma manual para poder utilizarlos, excepto los contenedores que si hay que lanzarlos de forma manual, salvo que se utilice el parámetro --restart=”always” al crear el contenedor para que arranque cuando se inicia Docker.