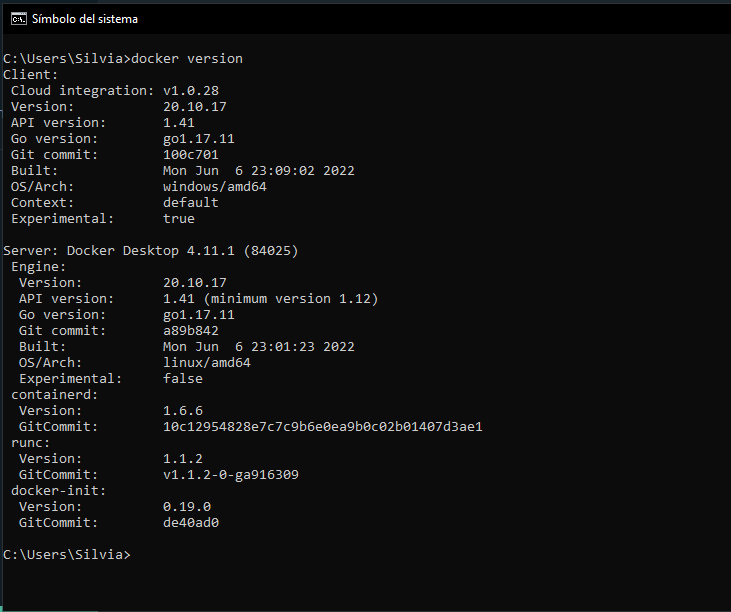
1. **Instalar Docker Community Edition**

• Diferentes opciones para cada sistema operativo

• https://docs.docker.com/

• Ejecutar el siguiente comando para comprobar versiones de cliente y demonio.

docker version

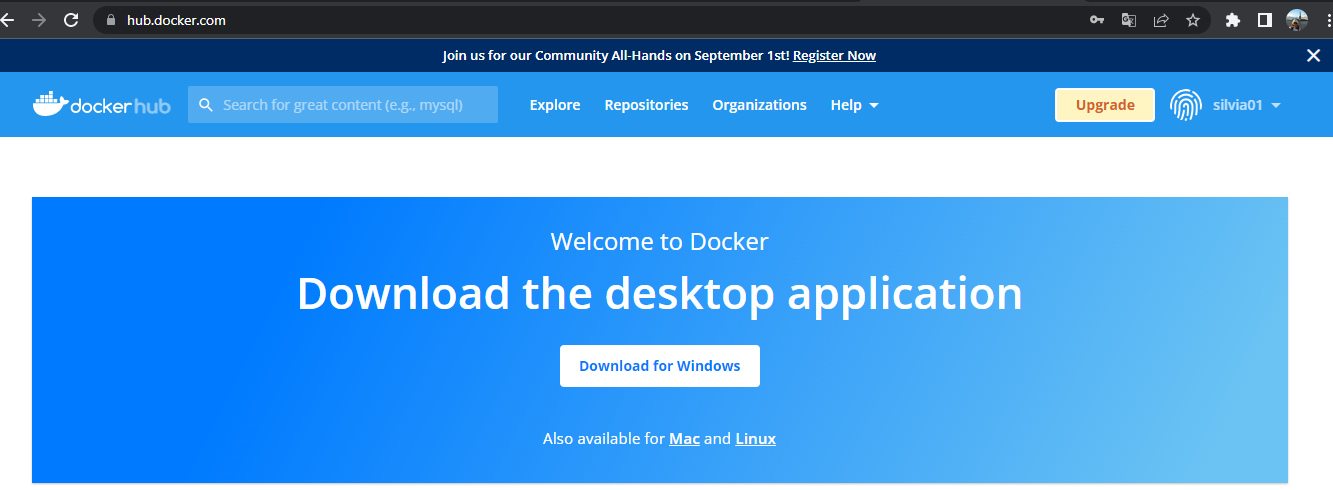


1. **Explorar DockerHub**

• Registrase en docker hub: https://hub.docker.com/

• Familiarizarse con el portal

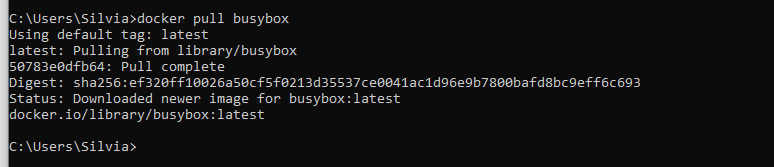
Ya tenía cuenta



1. **Obtener la imagen BusyBox**

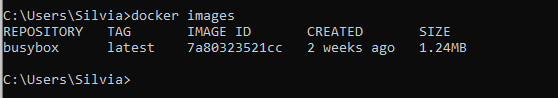
• Ejecutar el siguiente comando, para bajar una imagen de DockerHub

docker pull busybox



• Verificar qué versión y tamaño tiene la imagen bajada, obtener una lista de imágenes locales:

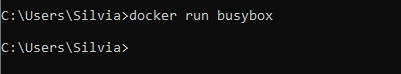
docker images



1. **Ejecutando contenedores**

• Ejecutar un contenedor utilizando el comando run de docker:

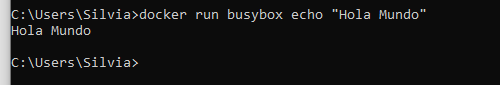
docker run busybox



• Explicar porque no se obtuvo ningún resultado

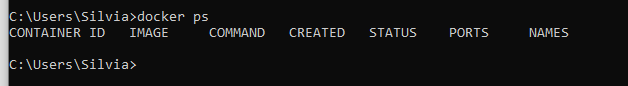
no se le indica que debe hacer. Le está faltando alguna instrucción o comando para que docker sepa qué hacer con la imagen

• Especificamos algún comando a correr dentro del contendor, ejecutar, por ejemplo: docker run busybox echo "Hola Mundo"

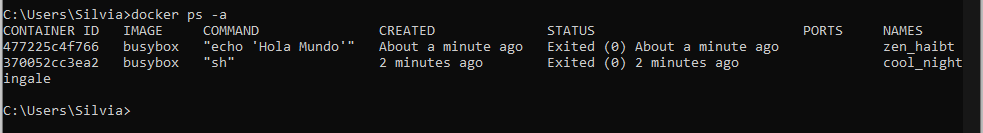


• Ver los contendores ejecutados utilizando el comando ps:

docker ps



• Vemos que no existe nada en ejecución, correr entonces: docker ps -a



• Mostrar el resultado y explicar que se obtuvo como salida del comando anterior

Muestra el estado de todos los contenedores, detenidos y en ejecución.

1. **Ejecutando en modo interactivo**

• Ejecutar el siguiente comando

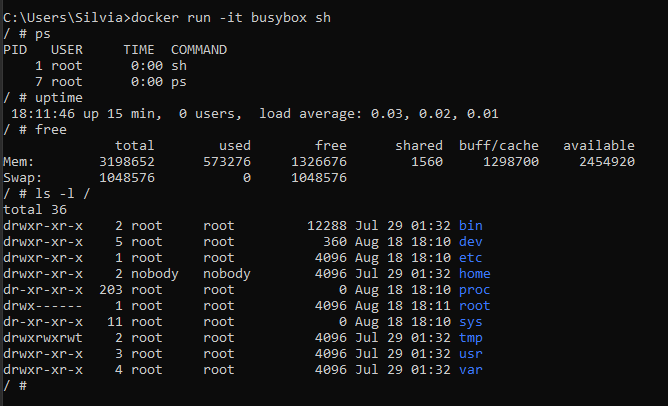
docker run -it busybox sh

• Para cada uno de los siguientes comandos dentro de contenedor, mostrar los resultados: ps

uptime

free

ls -l /



• Salimos del contendor con:

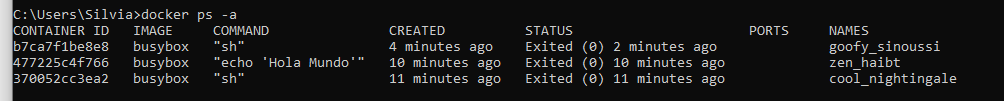
Exit



1. **Borrando contendores terminados**

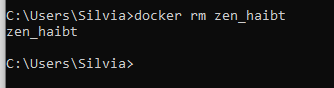
• Obtener la lista de contendores

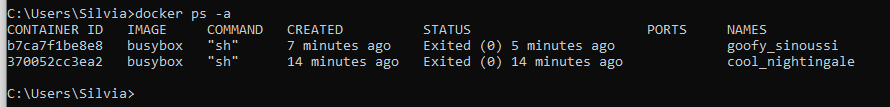
docker ps -a



• Para borrar podemos utilizar el id o el nombre (autogenerado si no se especifica) de contendor que se desee, por ejemplo:

docker rm elated\_lalande

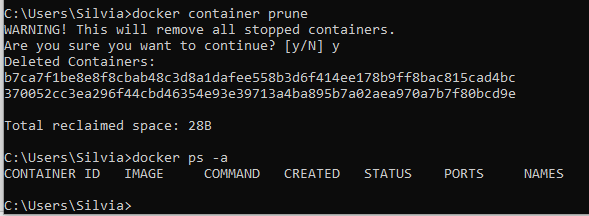




• Para borrar todos los contendores que no estén corriendo, ejecutar cualquiera de los siguientes comandos:

docker rm $(docker ps -a -q -f status=exited)

docker container prune



1. **Montando volúmenes Hasta este punto los contenedores ejecutados no tenían contacto con el exterior, ellos corrían en su propio entorno hasta que terminaran su ejecución. Ahora veremos cómo montar un volumen dentro del contenedor para visualizar por ejemplo archivos del sistema huésped:**

• Ejecutar el siguiente comando, cambiar myusuario por el usuario que corresponda. En linux/Mac puede utilizarse /home/miusuario):

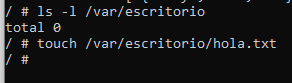
docker run -it -v C:\Users\misuario\Desktop:/var/escritorio busybox /bin/sh



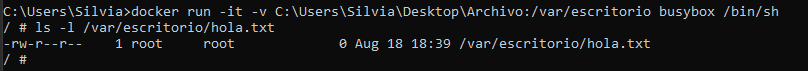
• Dentro del contenedor correr

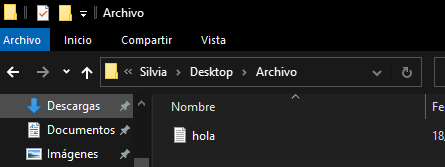
ls -l /var/escritorio

touch /var/escritorio/hola.txt



• Verificar que el Archivo se ha creado en el escritorio o en el directorio home según corresponda

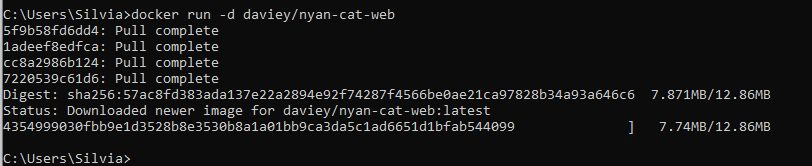




1. **Publicando puertos En el caso de aplicaciones web o base de datos donde se interactúa con estas aplicaciones a través de un puerto al cual hay que acceder, estos puertos están visibles solo dentro del contenedor. Si queremos acceder desde el exterior deberemos exponerlos.**

• Ejecutar la siguiente imagen, en este caso utilizamos la bandera -d (detach) para que nos devuelva el control de la consola:

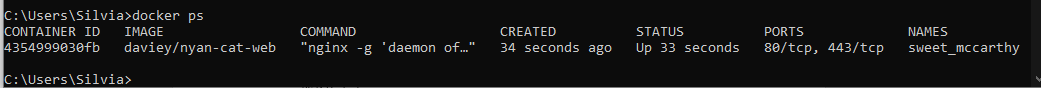
docker run -d daviey/nyan-cat-web



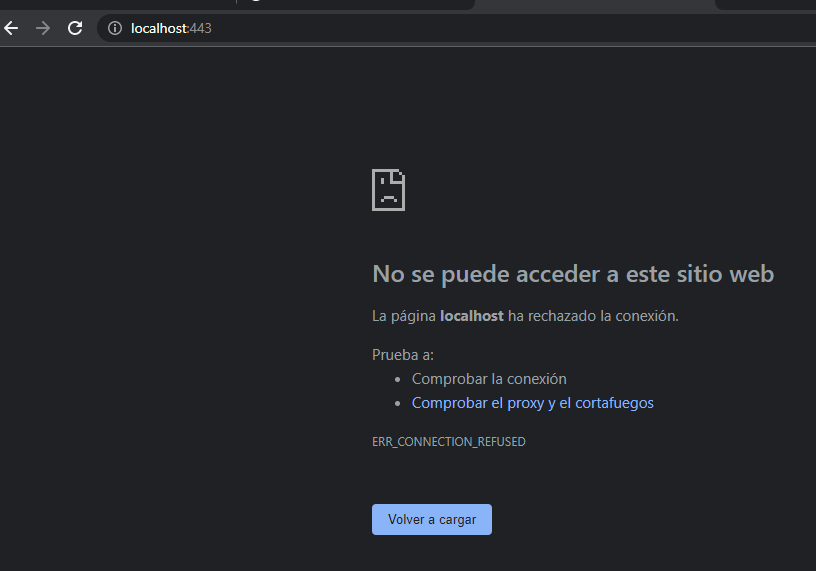
• Si ejecutamos un comando ps:

PS D:\> docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 87d1c5f44809 daviey/nyan-cat-web "nginx -g 'daemon of…" 2 minutes ago Up 2 minutes 80/tcp, 443/tcp compassionate\_raman



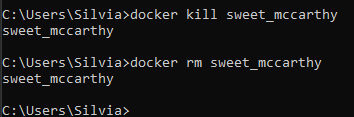
• Vemos que el contendor expone 2 puertos el 80 y el 443, pero si intentamos en un navegador acceder a http://localhost no sucede nada.



• Procedemos entonces a parar y remover este contenedor:

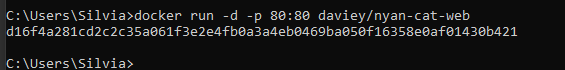
docker kill compassionate\_raman

docker rm compassionate\_raman



• Vamos a volver a correrlo otra vez, pero publicando uno de los puertos solamente, el este caso el 80

docker run -d -p 80:80 daviey/nyan-cat-web



• Accedamos nuevamente a http://localhost y expliquemos que sucede.



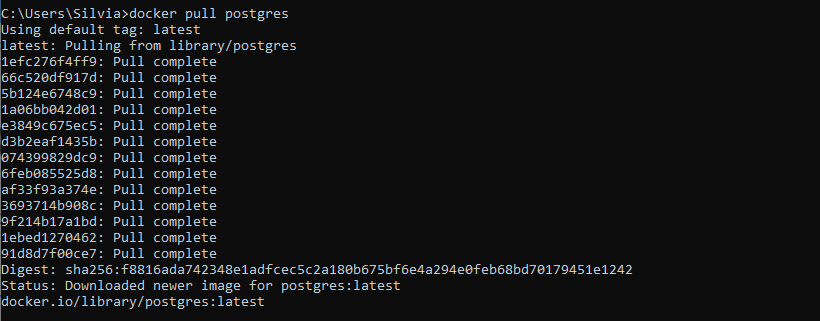
Se expuso un puerto local que se comunique con un puerto remoto.

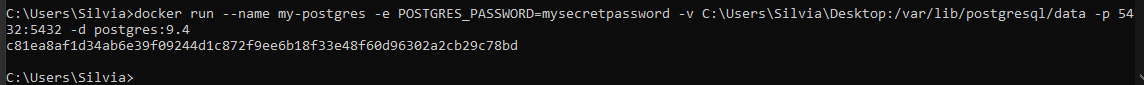
1. **Utilizando una base de datos**

• Levantar una base de datos PostgreSQL

mkdir $HOME/.postgres

docker run --name my-postgres -e POSTGRES\_PASSWORD=mysecretpassword -v $HOME/.postgres:/var/lib/postgresql/data -p 5432:5432 -d postgres:9.4

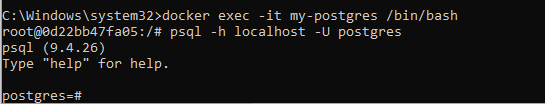




• Ejecutar sentencias utilizando esta instancia

docker exec -it my-postgres /bin/bash

psql -h localhost -U postgres



#Estos comandos se corren una vez conectados a la base

\l

create database test;

\connect test

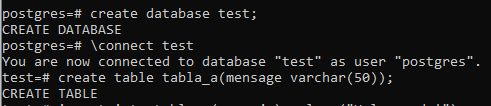
create table tabla\_a (mensaje varchar(50));

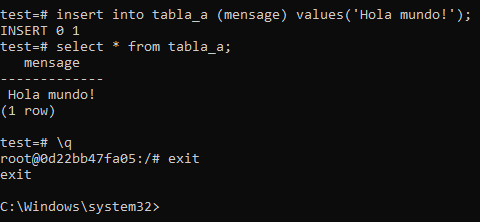
insert into tabla\_a (mensaje) values('Hola mundo!');

select \* from tabla\_a;

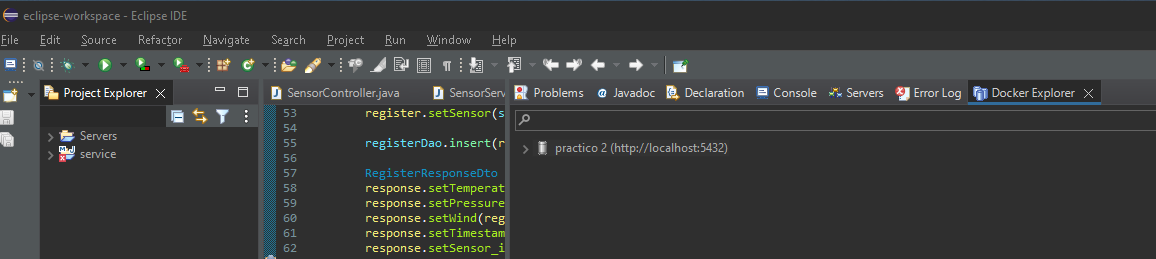
\q

exit





• Conectarse a la base utilizando alguna IDE (Dbeaver - https://dbeaver.io/, eclipse, IntelliJ, etc.…). Interactuar con los objetos creados.



• Explicar que se logró con el comando docker run y docker exec ejecutados en este ejercicio.

El comando docker run se usó para crear una imagen de un contenedor, descargado previamente, y lo inicia. El comando docker exec, se usó para ejecutar comandos dentro del contenedor.