DOCKER

Comandos y tips



Contenido

[Introducción 3](#_Toc85704663)

[Comandos 3](#_Toc85704664)

[Comandos básicos: 3](#_Toc85704665)

[Banderas: 3](#_Toc85704666)

[Ver el contenido de un contenedor 3](#_Toc85704667)

[Extraer los archivos del contenedor a la pc y biceversa 3](#_Toc85704668)

[Eliminar imágenes 3](#_Toc85704669)

[Ver el tamaño del proyecto en el contenedor 3](#_Toc85704670)

[Dockerfile 4](#_Toc85704671)

[From 4](#_Toc85704672)

[Run 4](#_Toc85704673)

[Copy 4](#_Toc85704674)

[Add 4](#_Toc85704675)

[Env 4](#_Toc85704676)

[WorkDir 4](#_Toc85704677)

[Expose 4](#_Toc85704678)

[Label 4](#_Toc85704679)

[User 4](#_Toc85704680)

[Volume 4](#_Toc85704681)

[CMD 4](#_Toc85704682)

[Dockerignore 5](#_Toc85704683)

[Cambiar nombre Docker file 5](#_Toc85704684)

[Docker Containers 5](#_Toc85704685)

[Listar/ Mapear puertos 5](#_Toc85704686)

[Iniciar/ Reiniciar / Detener 5](#_Toc85704687)

[Variables de entornos 5](#_Toc85704688)

[Redes de contenedores 5](#_Toc85704689)

[RabbitMq 6](#_Toc85704690)

[Instalación de Redis 6](#_Toc85704691)

[Instalación de MongoDB 6](#_Toc85704692)

[Instalación de Mysql con Docker 7](#_Toc85704693)

[Instalación de SQL Server con Docker 7](#_Toc85704694)

[**Para contenedores de Linux** 7](#_Toc85704695)

[**Descargar imágenes de SQL Server Linux** 7](#_Toc85704696)

[**Ejecute las imágenes de Docker para SQL Server 2017 y 2019** 7](#_Toc85704697)

[Referencias 8](#_Toc85704698)

# Introducción

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Pasos para instalar Docker Desktop:

1. Descargar instalador de Docker disponible en: <https://hub.docker.com/editions/community/docker-ce-desktop-windows>

1. Desde la power Shell ejecutar los siguientes comandos:

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

1. Instalar Docker Desktop con permisos de administrador.
2. Reiniciar, y volver a iniciar sesión con permisos de Administrador
3. Descargar e Instalar wsl2 upgrade.
4. Ejecutar comando en powershell **sin permisos de administrador**

wsl --set-default-version 2

1. Iniciar Docker Desktop, aceptar términos de licencia.

# Configuraciones de Proxy

Luego de la instalación se requiere habilitar el proxy para habilitar la descarga de imágenes mediante el proxy. Para ello debemos abrir la aplicación desktop, ir a la opción de configuraciones como se resalta en la siguiente imagen:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Correo electrónico

Descripción generada automáticamente

Desde el panel de configuraciones se deberá seleccionar el apartado de “Resource”, al desplegar el sub-menu, se debe seleccionar la opción “PROXIES”. Completar los campos del formulario con los siguientes valores:

1. Habilitar la opción “Manual proxy configuration”
2. Web server (HTTP): 192.168.101.101:8080
3. Secure Web Server (HTTPS): 192.168.101.101:8080
4. Bypass proxy settings for these hosts & domains: localhost,127.0.0.1
5. Confirmar cambios con el botón “Apply & Restart”

El formulario del proxy debería quedar como se muestra en la siguiente imagen:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# Comandos

## Comandos básicos:

docker -h: es para ver las opciones.

docker run image: Ver las imágenes locales descargadas/creadas

docker search microsoft: es para ver las imágenes de Microsoft que existen en el repositorio de Docker hub.

docker run --detach --publish 8011:80 kitematic/hello-world-nginx: Publicar una imagen en el puerto de mi pc 8011

Docker build: es construir una imagen tomando los datos de un dockerfile, ejemplo: Docker build –tag my-api-nodejs:beta .

Docker history -H name-container:tag: es para ver las capas que se crearon en una imagen

Diferencia entre COPY y ADD: “copy” sólo copia los archivos de un directorio local, mientras que “add” te permite además poner una url del directorio donde se encuentras los archivos a ser descargados

WORKDIR: Va a ser el directorio de trabajo, o sea donde se van a implementar los comandos

### Banderas:

--detach (es para salir del proceso una vez ejecutado y poder seguir ejecutando otros comandos)

--publish (es para poder mapear los puertos del contenedor con los de mi pc, en el ej. el puerto 8011

de la pc es mapeado con el 80 del contenedor)

## Ver el contenido de un contenedor

docker exec --interactive --tty id-container /bin/sh :

## Extraer los archivos del contenedor a la pc y biceversa

docker container cp --help para rebir ayuda

docker container cp

## Eliminar imágenes

Docker rmi name-image:tag-name

## Ver el tamaño del proyecto en el contenedor

docker images -f reference=name-project\*

-f es una flag para filtrar imagenes en este caso

puede ser el nombre inicial y con el asteristico filtrará las coincidencias, con las versiones inclusive

# Dockerfile

Dockerfile es un archivo que nos permite crear una imagen apartir de otras o crear nuestras propias imágenes.

## From

## Run

## Copy

## Add

## Env

## WorkDir

## Expose

## Label

Se pueden agregar una serie de etiquetas (labels) en nuestra imagen, esto permite brindar mayor información sobre la imagen. Se podría agregar la versión, una descripción, etc. Se realiza de la siguiente forma:

LABEL versión=1.0.0

LABEL decription=”This is an node-api image”

LABEL author=”Enterprise-name”

## User

Se puede crear un usuario especifico para ejecutar la imagen

## Volume

Sirve para hacer que los datos se persistan en el disco, o sea mas allá de la vida del contenedor que ejecute la imagen.

## CMD

Es para ejecutar comandos al arrancar la imagen en un contenedor.

## Dockerignore

Es un archivo que nos permite registrar los archivos, carpetas, etc que no se desea tomar en cuenta al momento de generar la imagen. Esto nos permitirá optimizar las imagen en su peso final.

## Cambiar nombre Docker file

Docker build -t name-final-image -f my-dockerfile .

-f es el nombre del archivo dockerfile que queremos usar. Por ejemplo my-dockerfile

# Docker Containers

Para todos los comandos con contenedores podemos referirnos a ellos mediante su nombre o id.

## Listar/ Mapear puertos

Listar todos los contenedores: Docker ps

Ver los id`s de los containers: Docker ps -q

Ver todos los contenedores stopeados: Docker ps -a

Renombrar un contenedor: Docker name-container-actual name-container-new

Obtener la IP de un contenedor: Docker inspect name-container

Para ver el uso de recursos de los contenedores: Docker stats

## Iniciar/ Reiniciar / Detener

Para iniciar un contenedor puede ser con el nombre o el id: Docker start name-container

Para detener un contenedor: Docker stop id-container, también con el nombre del contenedor en lugar del id

Para reiniciar un contenedor: Docker restart. Esto lo va a frenar al servicio y lo inicia nuevamente, puede ser usado para los casos que un contenedor empiece a consumir muchos recursos o este tildada la ejecución.

Para ingresar a un contenedor: Docker exec -ti name-container bash.

Para ingresar con un usuario especifico: Docker exec -u user-name -ti name-container bash. Este usuario es el que esta definido en el dockerfile.

Para salir: exit

## Variables de entornos

Nos sirven para contener valores a utilizar conforme el ambiente y podemos crear variables para ser utilizadas cuando creamos el contenedor. Esto se logra de la siguiente manera: Docker run -e “environment-name=value” --name name-container image-name

# Redes de contenedores

# RabbitMq

Docker run -d -v ./rabbit-db:/var/lib/rabbitmq –hostname name-rabbitmq -p 5672:5672 -p 8081:15672 –name name-rabbitmq rabbitmq:3-managment

Flags:

--v ./rabbit-db:/var/lib/rabbitmq: es para que se persistan los datos en el directorio actual (donde esta el proyecto)

Docker-compose.yml:

version: '3'

services:

rabbitmq:

image: rabbitmq:3-management-alpine

container\_name: rabbitmq

restart: always

volumes:

- ./rabbitmq/rabbitmq.config:/etc/rabbitmq/rabbitmq.config:ro

- ./rabbitmq/definitions.json:/etc/rabbitmq/definitions.json:ro

ports:

- "5672:5672"

- "15672:15672"

environment:

RABBITMQ\_DEFAULT\_USER: admin

RABBITMQ\_DEFAULT\_PASS: admin

<https://zgadzaj.com/development/docker/docker-compose/containers/rabbitmq>

# Instalación de Redis

La instalación se puede hacer mediante un Docker-compose:

version: "3.7"

services:

redis:

image: redis:latest

ports:

- 6379:6379

command: ["redis-server", "--appendonly", "yes"]

volumes:

- redis-data:/data

redis-commander:

image: rediscommander/redis-commander:latest

environment:

- REDIS\_HOSTS=local:redis:6379

- HTTP\_USER=root

- HTTP\_PASSWORD=qwerty

ports:

- 8081:8081

depends\_on:

- redis

volumes:

redis-data:

# Instalación de MongoDB

docker run --name mongodb -p 27017:27017 -v mongodata:/data/db mongo

docker exec -it <container id> bash

mongo admin -u root -p rootpassword

show dbs

Cadena de conexión o connection string

Para conectarte a tu base de datos desde un cliente (ya sea Compass o tu app) es necesario que proporciones la siguiente cadena:

mongodb://user:password@127.0.0.1:27017/authSource=admin

mongodb://admin:password@127.0.0.1:27017/?authSource=admin

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

<https://diegoorozco.com/dockermongodb>

docker run -d --name mongo-on-docker -p 27888:27017 -e MONGO\_INITDB\_ROOT\_USERNAME=mongoadmin -e MONGO\_INITDB\_ROOT\_PASSWORD=secret mongo mongo -v ./mongo-volume:/data/db restart: always

docker run -d --name mongo-on-docker -p 27888:27017 -e MONGO\_INITDB\_ROOT\_USERNAME=admin -e MONGO\_INITDB\_ROOT\_PASSWORD=admin mongo -v ./mongo-volume:/data/db

<https://www.code4it.dev/blog/run-mongodb-on-docker>

# Instalación de Mysql con Docker

Dependiendo del SO que estén usando, Ubuntu, centos, etc, hay que ejecutar los comandos para instalar el cliente.

# Instalación de SQL Server con Docker

docker search microsoft: es para ver las imágenes de Microsoft que existen en el repositorio de Docker hub.

## **Para contenedores de Linux**

De forma predeterminada, el escritorio de Docker se ejecuta en modo contenedor de Linux. Entonces, comencemos con Linux.

### **Descargar imágenes de SQL Server Linux**

Abra una ventana de cmd, ejecute lo siguiente para descargar la imagen de SQL Server 2017 y la imagen de SQL Server 2019 desde Microsoft Docker Hub.

docker pull mcr.microsoft.com/mssql/server:2019-latest

docker pull mcr.microsoft.com/mssql/server:2017-latest

Texto

Descripción generada automáticamente

### **Ejecute las imágenes de Docker para SQL Server 2017 y 2019**

Asignemos el puerto 1433 en el host para SQL Server 2017 y el puerto 1436 en el host para SQL Server 2019. También especificaremos la edición empresarial con la opción "MSSQL\_PID = Enterprise". Sin esta configuración, de forma predeterminada, un contenedor ejecuta la edición para desarrolladores.

Algunas advertencias. Primero, asegúrese de establecer una contraseña compleja, de lo contrario no podrá conectarse más tarde ya que el contenedor no se iniciará. En segundo lugar, asegúrese de utilizar comillas dobles ("") para los parámetros, especialmente para SA\_PASSWORD. Noté que, para Linux, el contenedor se cierra en segundos si utilizo comillas simples ('') o sin comillas.

Aquí están los comandos para iniciar un contenedor a partir de una imagen descargada:

docker run --name sql\_2017 -e "ACCEPT\_EULA=Y" -e "SA\_PASSWORD=Password123!" -e "MSSQL\_PID=Enterprise" -p 1433:1433 -d mcr.microsoft.com/mssql/server:2017-latest

**para crear una instalación con volumen (almacenamiento de data local)**

docker run --name SQLServer2017 -e "ACCEPT\_EULA=Y" -e "SA\_PASSWORD=Password123!" -e "MSSQL\_PID=Enterprise" -p 1433:1433

-v c:\Docker\Volumes\SQLServer:c:\SQLData -d mcr.microsoft.com/mssql/server:2017-latest

# Referencias

<https://www.sqlservercentral.com/articles/docker-desktop-on-windows-10-for-sql-server-step-by-step>

<https://octopus.com/blog/running-sql-server-developer-install-with-docker>