CGI
Experience the commitment®

From Zero to Docker

Training | 2019.05.16 | Mário Dagot, Jorge Dias

Docker is an open platform for developing, shipping, and running applications. Through the course of this training we will guide you to the most common feature and use cases of docker. Take this as an introduction and an opportunity to dive into the docker world.

AGENDA

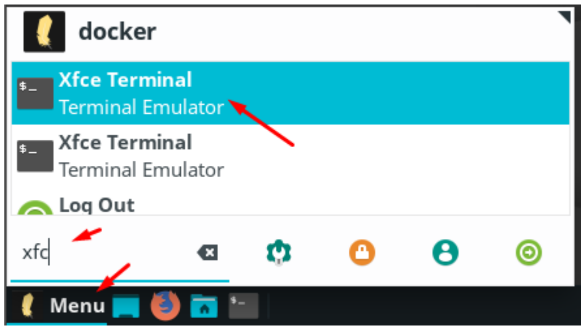
- 01 - Install Vim and Terminator and VSCode
- 02 - Install Docker CE for Ubuntu
- **03 - Hello from Busybox**
- 04 - Webapp with Docker
- 05a - Webapp with Docker - My first Dockerfile – Nginx
- 05b.1 - Webapp with Docker - My first Dockerfile - Dotnet Core
- 05b.2 - Webapp with Docker - My first Dockerfile MultiStage - Dotnet Core
- 06 - Save and Restore and Push to Docker Hub
- 07a - Webapp with database integration - My first network – SpringBoot
- 07b - Webapp with database integration - My first docker-compose – SpringBoot

03 - HELLO FROM BUSYBOX

Objective

- Learn about the basic docker cli commands:
  - Images
  - Container
  - Run
  - Pull
  - Ps
- Learn how to use the docker public registry: docker hub

Step by Step

Open a terminal windows	 docker Xfce Terminal Terminal Emulator Xfce Terminal Terminal Emulator Log Out xfc Menu

List images	`docker ~ docker images`
	`REPOSITORY          TAG                 IMAGE ID            CREATED`
	`SIZE`

Pull busybox image from docker hub	`docker ~ docker pull busybox`
	`Using default tag: latest`
	`latest: Pulling from library/busybox`
	`fc1a6b909f82: Pull complete`
	`Digest: sha256:954e1f01e80ce09d0887ff6ea10b13a812cb01932a0781d6b0cc23f743a874fd`
	`Status: Downloaded newer image for busybox:latest`

List images, new busybox should be present	`docker ~ docker images`
	`REPOSITORY          TAG                 IMAGE ID            CREATED`
	`SIZE`
	`busybox             latest              af2f74c517aa        5 weeks ago`
	`1.2MB`

We can check which containers exist.
 	
The -a flag allows us to see all running and not running containers
	`docker ~ docker ps -a`
	`CONTAINER ID        IMAGE               COMMAND             CREATED`
	`STATUS              PORTS               NAMES`

Start your first container based on the busybox image. We want to echo the message “Hello, from busybox!”
 	
--name allows to define a name for the container. If this is not defined an automatic name will be provided
	`docker ~ docker run --name mybusybox busybox echo "Hello, from Busybox!"`
	`Hello, from Busybox!`

List all containers	`docker ~ docker ps -a`

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS                      PORTS               NAMES
bfe90409a979        busybox             "echo 'Hello, from B…"   7 seconds ago
Exited (0) 6 seconds ago                        mybusybox
```

Create new container. This time we want to echo a slight different message: “Hello again, from busybox!”

We were not able to start a new container since a container with the same name already exists.

```
docker ~  docker run --name mybusybox busybox echo "Hello again, from
Busybox!"
docker: Error response from daemon: Conflict. The container name "/mybusybox"
is already in use by container
"bfe90409a979917f084f18b2211288217ad37690d1cb5d205fc3c99d02998b70". You have to
remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
```

Let's remove the container and list again the containers.

```
docker ~  docker container rm mybusybox
mybusybox
docker ~  docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

Let's try again to create a new container.
This time it works.

Let's clean up the container.

```
docker ~  docker run --name mybusybox busybox echo "Hello again, from
Busybox!"
Hello again, from Busybox!
docker ~  docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS                      PORTS               NAMES
de0bb52c523a        busybox             "echo 'Hello again, …"   5 seconds ago
Exited (0) 5 seconds ago                        mybusybox
docker ~  docker container rm mybusybox
mybusybox
```

We can use the --rm flag to say to docker engine that he should remove the container once it finishes executing.

Run the command and try next to list the existing containers. Now it should exist.

```
docker ~  docker run --rm --name mybusybox busybox sh
```

We could also notice than on the last command, when we run the sh command, the shell exited immediately.

We can bypass this behaviours by passing the -it flag: the interactive mode.

```
docker ~  docker run --rm -it --name mybusybox busybox sh
/ # ls -l
total 36
drwxr-xr-x    2 root     root         12288 Apr  2 04:32 bin
drwxr-xr-x    5 root     root           360 May  8 08:39 dev
```

This will allows to gain access to the running container (and its filesystem).

It's a safe sandbox. We can play around in safety and we will not affect the host environment.

Try to remove the /bin. All commands inside the container stop working.

Exit the container and start again. You will see that all is as it was before.

Cool!!!

```
drwxr-xr-x    1 root     root          4096 May  8 08:39 etc
drwxr-xr-x    2 nobody   nogroup       4096 Apr  2 04:32 home
dr-xr-xr-x  165 root     root             0 May  8 08:39 proc
drwx------    1 root     root          4096 May  8 08:39 root
dr-xr-xr-x   13 root     root             0 May  8 08:39 sys
drwxrwxrwt    2 root     root          4096 Apr  2 04:32 tmp
drwxr-xr-x    3 root     root          4096 Apr  2 04:32 usr
drwxr-xr-x    4 root     root          4096 Apr  2 04:32 var
/ # rm -rf /bin
/ # exit
```

Lessons learned

We learned how to list images locally, pull images from the docker registry and start and remove containers. Take also a look at **docker stop**, **start** and **restart**. Those commands will allow you to stop, start and restart containers.

When removing containers, as a precaution measure, the docker daemon doesn't allow to remove running containers. They need to first be stopped or explicitly use the -f flag to force its removal.

Maybe the most important thing we should never to forget is that containers are ephemeral. Never do changes manually to a container, everything will be gone once the container is removed. We will see, on the next sessions, several techniques on how to use them in a safe way – state is part of our day-to-day and docker has this covered for us.

Don't forget to browse docker hub (https://hub.docker.com/search/?q=&type=image), it contains already many ready to use images for us to use.

Revision History

Version	Date	Author	Description
1.0	2019.05.01	Mário Dagot, Jorge Dias	Initial Version