



# Dockers - Quick Start

A short-guide focused on implementing  
custom Dockerfiles

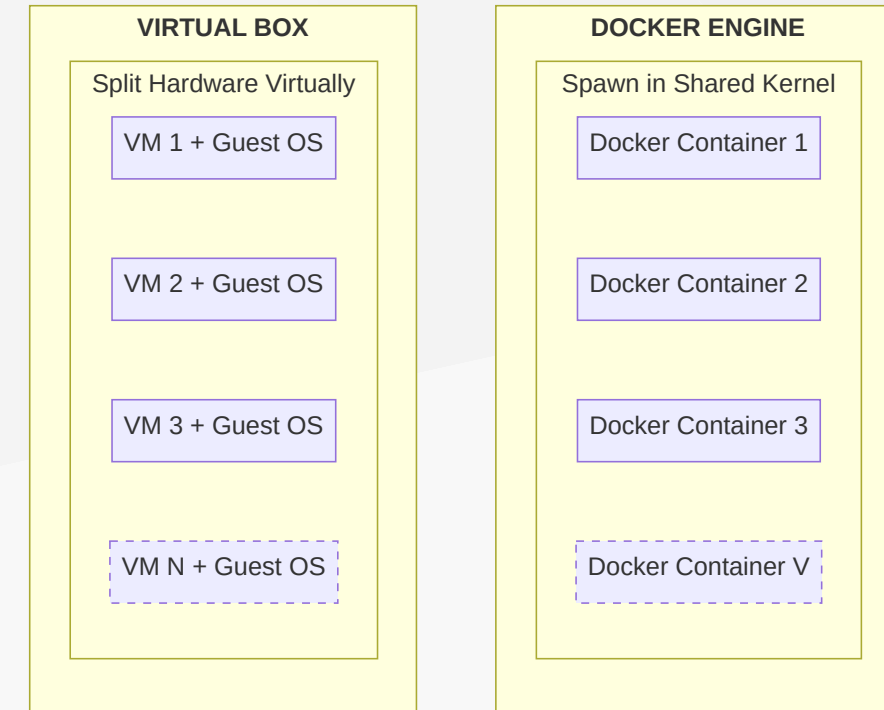
*2020-10-30 Mirek Jaros*

# 1. What is a Docker?

- Docker is an **eco-system** to allow us to **configure, deploy** and **run applications** on production systems.
  - *eg. Apache Tomcat Web Server + Java Enterprise Application*
- As a **tool, helps** engineers to **easily run applications locally**.
  - *eg. Oracle Database Express Edition for local development*

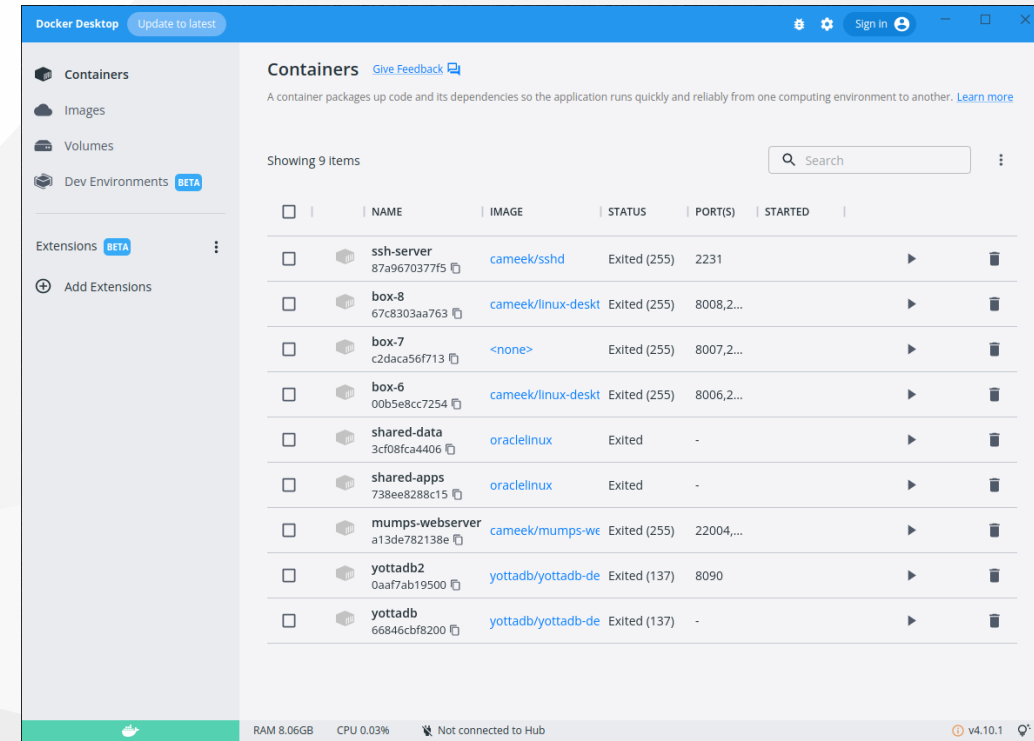
## 2. Comparison to VMs

- Comparison between Docker Engine and Hypervisors (Hyper-V, ESXi, VBox,...)
- You can run limited N instances of VMs  
*vs.*
- Very high number V of Docker Containers within same hardware



# 3. Prerequisite

- Before next steps **make sure your Docker Desktop is running**  
<https://docs.docker.com/desktop/>
- It's possible to have only Docker Engine, however, you would have to use CLI
- And installation of just engine is usually performed under high-privileged account, requires deeper knowledge around security

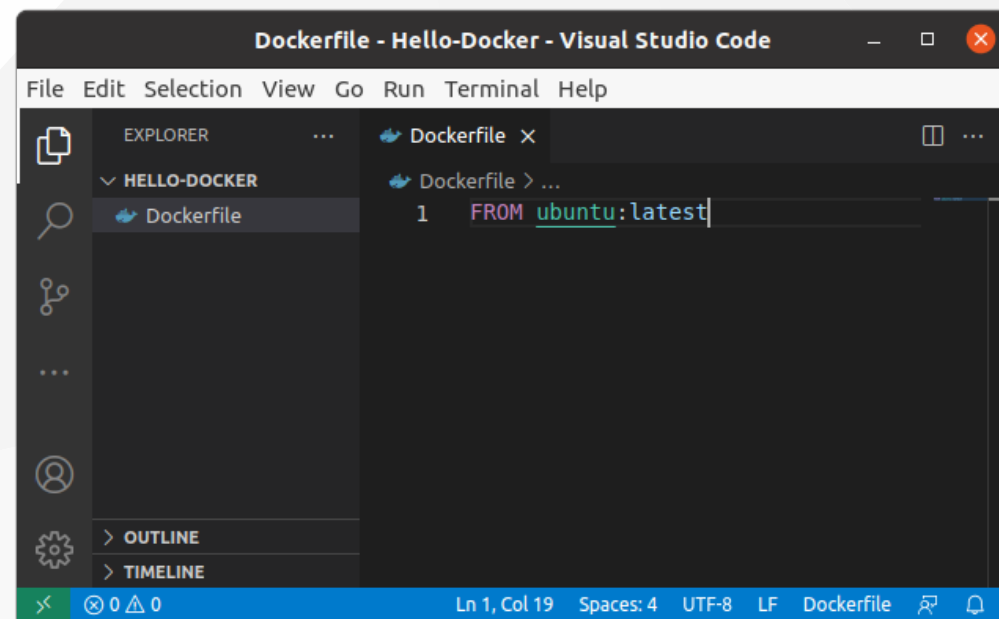


## 4. First Dockerfile

- Create dedicated directory like "Hello-Docker"
- Inside the directory create new file "Dockerfile" with following content:

```
FROM ubuntu:latest
```

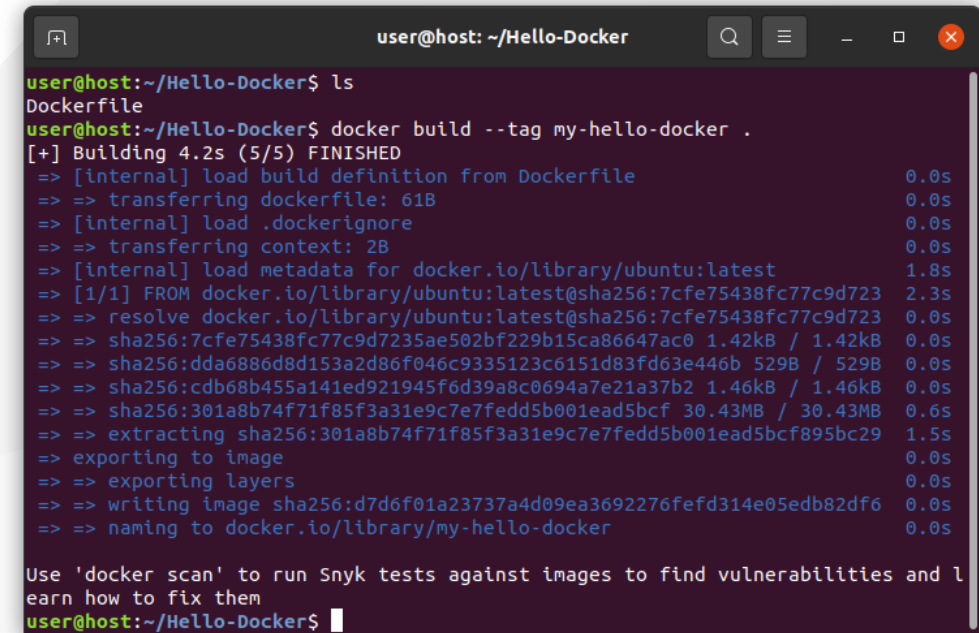
- You can use Vi, Notepad,.. or some IDE like VSCode



# 5. Build Dockerfile into Image

- Open a command line or shell, navigate to project "Hello-Docker"
- Execute the build, don't forget to put there the dot:

```
docker build --tag my-hello-docker .
```

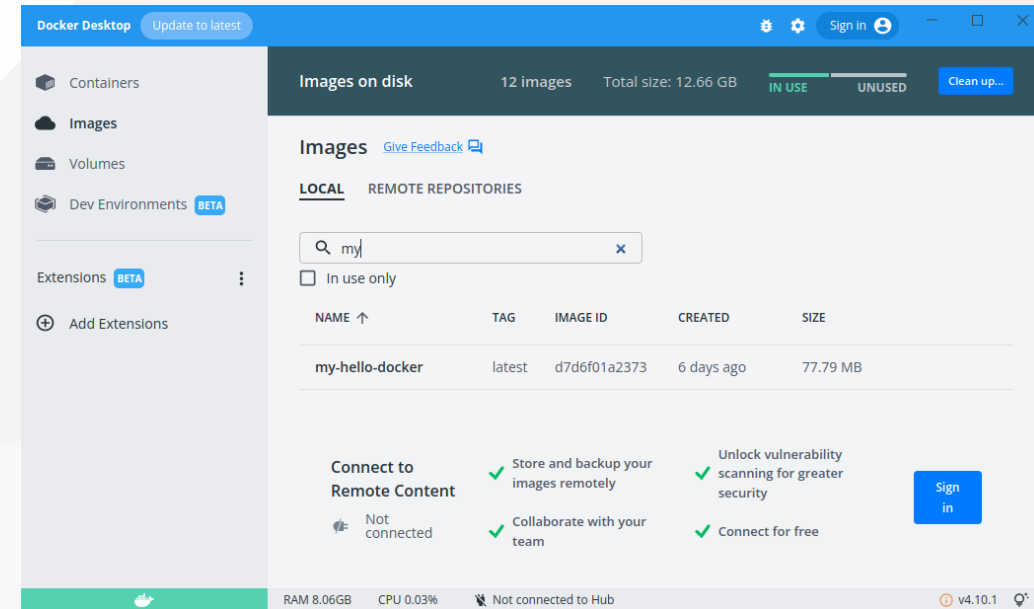
A terminal window titled 'user@host: ~/Hello-Docker' showing the execution of 'docker build --tag my-hello-docker .' in the directory ~/Hello-Docker. The build process is shown in progress, with various steps and their durations. The build is successful, and the image is named 'my-hello-docker'.

```
user@host:~/Hello-Docker$ ls
Dockerfile
user@host:~/Hello-Docker$ docker build --tag my-hello-docker .
[+] Building 4.2s (5/5) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 61B                                                0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/ubuntu:latest                 1.8s
=> [1/1] FROM docker.io/library/ubuntu:latest@sha256:7cfe75438fc77c9d723       2.3s
=> => resolve docker.io/library/ubuntu:latest@sha256:7cfe75438fc77c9d723       0.0s
=> => sha256:7cfe75438fc77c9d7235ae502bf229b15ca86647ac0 1.42kB / 1.42kB       0.0s
=> => sha256:dda6886d8d153a2d86f046c9335123c6151d83fd63e446b 529B / 529B     0.0s
=> => sha256:cdb68b455a141ed921945f6d39a8c0694a7e21a37b2 1.46kB / 1.46kB     0.0s
=> => sha256:301a8b74f71f85f3a31e9c7e7fedd5b001ead5bcf 30.43MB / 30.43MB     0.6s
=> => extracting sha256:301a8b74f71f85f3a31e9c7e7fedd5b001ead5bcf895bc29     1.5s
=> => exporting to image                                                         0.0s
=> => exporting layers                                                           0.0s
=> => writing image sha256:d7d6f01a23737a4d09ea3692276fef314e05edb82df6       0.0s
=> => naming to docker.io/library/my-hello-docker                             0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
user@host:~/Hello-Docker$
```

# 6. Check Images via GUI

- In Docker Desktop click on Images
- The new image name "my-hello-docker" should be there
- You can use search filtering

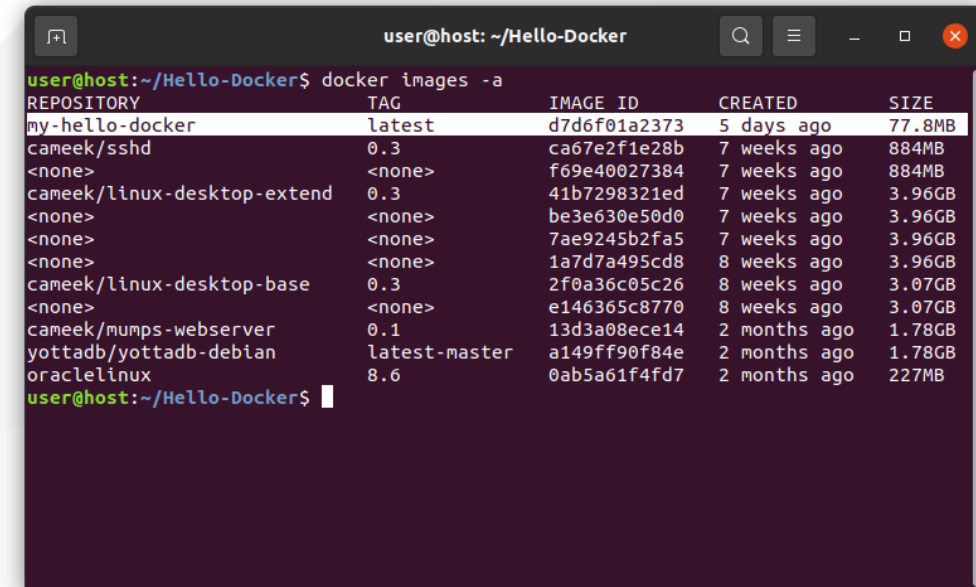


# 7. Check Images via CLI

- In shell/command line run following:

```
docker images -a
```

- The switch "-a" displays also inactive images



```
user@host: ~/Hello-Docker$ docker images -a
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
my-hello-docker      latest       d7d6f01a2373  5 days ago  77.8MB
cameek/ssh           0.3          ca67e2f1e28b  7 weeks ago  884MB
<none>               <none>       f69e40027384  7 weeks ago  884MB
cameek/linux-desktop-extend 0.3          41b7298321ed  7 weeks ago  3.96GB
<none>               <none>       be3e630e50d0  7 weeks ago  3.96GB
<none>               <none>       7ae9245b2fa5  7 weeks ago  3.96GB
<none>               <none>       1a7d7a495cd8  8 weeks ago  3.96GB
cameek/linux-desktop-base 0.3          2f0a36c05c26  8 weeks ago  3.07GB
<none>               <none>       e146365c8770  8 weeks ago  3.07GB
cameek/mumps-webserver 0.1          13d3a08ece14  2 months ago  1.78GB
yottadb/yottadb-debian latest-master a149ff90f84e  2 months ago  1.78GB
oraclelinux          8.6          0ab5a61f4fd7  2 months ago  227MB
user@host: ~/Hello-Docker$
```

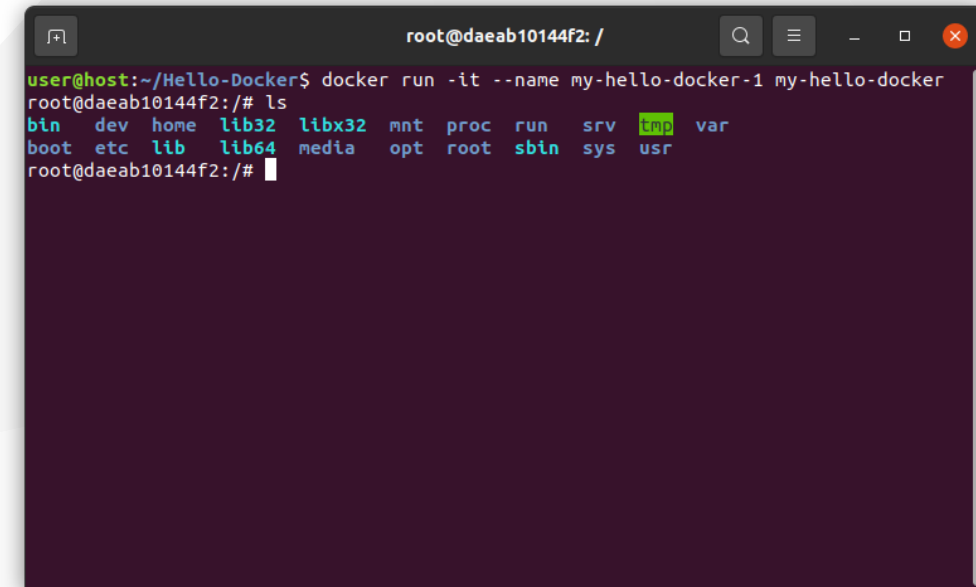


# 8. Run New Container

- Using CLI execute command:

```
docker run -it --name my-hello-docker-1 my-hello-docker
```

- You should get immediately interactive terminal inside that container with Bash shell
- You can try commands like `/s`

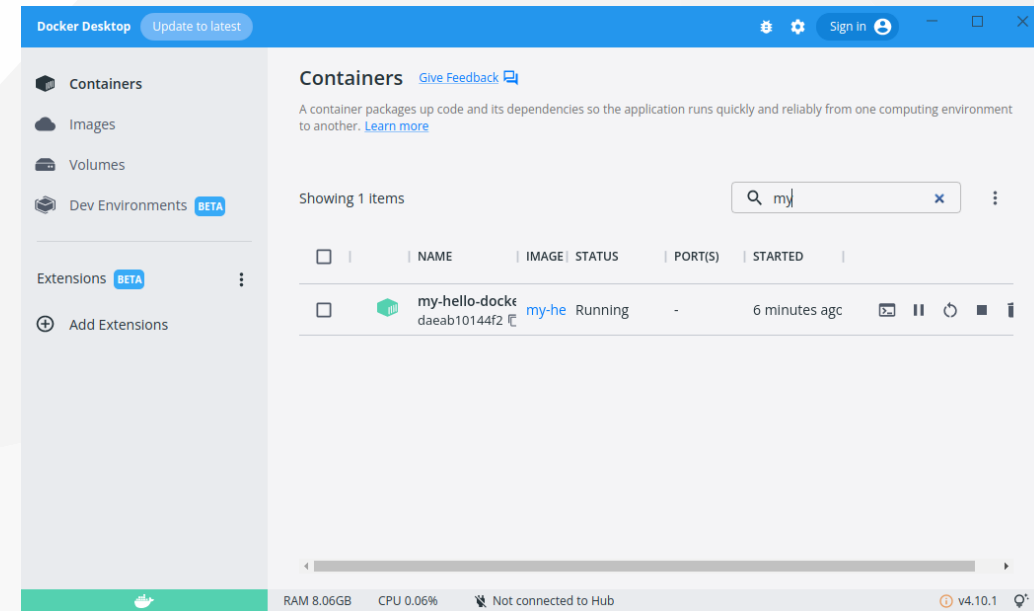


A terminal window titled 'root@daeab10144f2: /' showing the execution of a Docker command and the output of the 'ls' command inside a container. The command 'docker run -it --name my-hello-docker-1 my-hello-docker' is entered at the prompt 'user@host:~/Hello-Docker\$'. The prompt changes to 'root@daeab10144f2: /#', and the 'ls' command is executed, displaying a directory listing of system directories.

```
root@daeab10144f2: /  
user@host:~/Hello-Docker$ docker run -it --name my-hello-docker-1 my-hello-docker  
root@daeab10144f2: /# ls  
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var  
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr  
root@daeab10144f2: /#
```

# 9. Check Containers via GUI

- In Docker Desktop click on Containers
- The new created container "my-hello-docker-1" should be there
- You can use search filtering

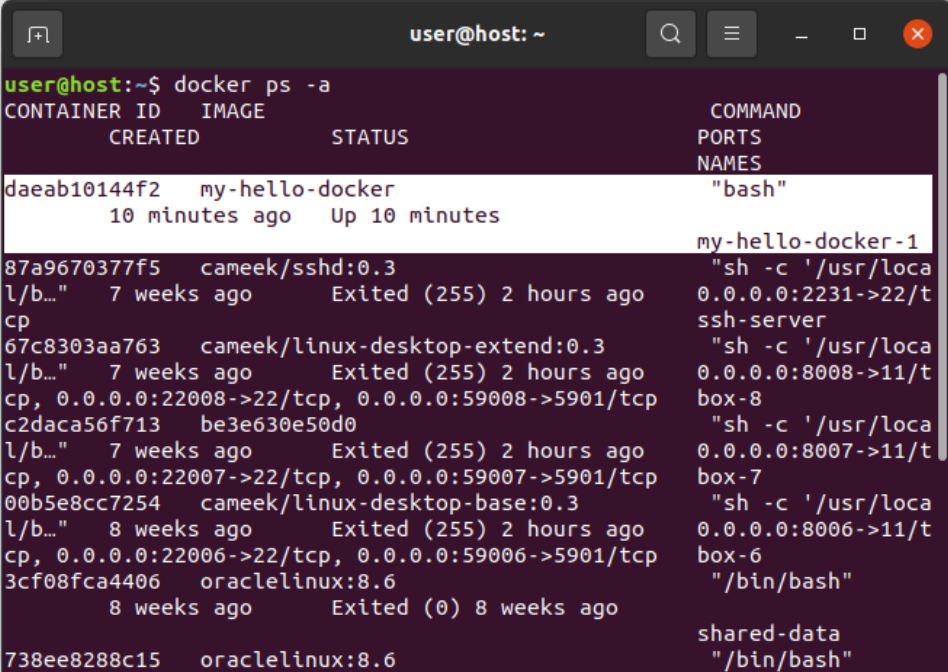


# 10. Check Containers via CLI

- In shell/command line run following:

```
docker ps -a
```

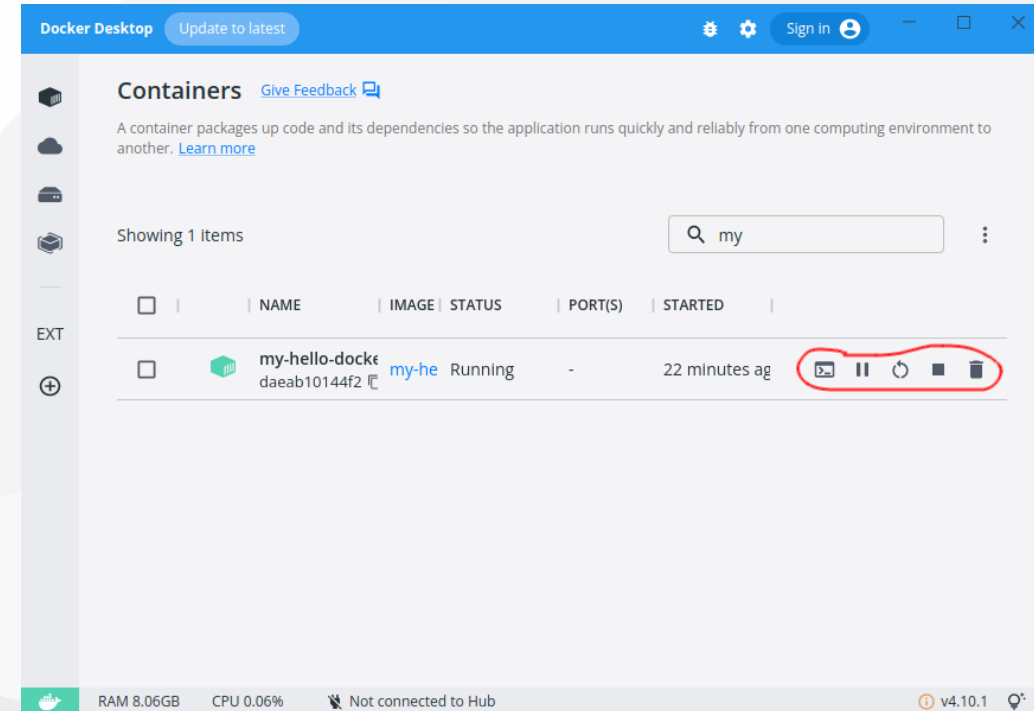
- The switch "-a" displays also inactive images



```
user@host: ~  
user@host:~$ docker ps -a  
CONTAINER ID   IMAGE                                STATUS      COMMAND      PORTS      NAMES  
daaab10144f2   my-hello-docker                    Up 10 minutes    "bash"      my-hello-docker-1  
87a9670377f5   cameek/sshd:0.3                    Exited (255) 2 hours ago    "sh -c '/usr/loca  
l/b..." 7 weeks ago    0.0.0.0:2231->22/t  
ssh-server  
67c8303aa763   cameek/linux-desktop-extend:0.3    Exited (255) 2 hours ago    "sh -c '/usr/loca  
l/b..." 7 weeks ago    0.0.0.0:8008->11/t  
cp, 0.0.0.0:22008->22/tcp, 0.0.0.0:59008->5901/tcp    box-8  
c2daca56f713   be3e630e50d0                       Exited (255) 2 hours ago    "sh -c '/usr/loca  
l/b..." 7 weeks ago    0.0.0.0:8007->11/t  
cp, 0.0.0.0:22007->22/tcp, 0.0.0.0:59007->5901/tcp    box-7  
00b5e8cc7254   cameek/linux-desktop-base:0.3      Exited (255) 2 hours ago    "sh -c '/usr/loca  
l/b..." 8 weeks ago    0.0.0.0:8006->11/t  
cp, 0.0.0.0:22006->22/tcp, 0.0.0.0:59006->5901/tcp    box-6  
3cf08fca4406   oraclelinux:8.6                     Exited (0) 8 weeks ago      "/bin/bash"  
738ee8288c15   oraclelinux:8.6                     Exited (0) 8 weeks ago      "/bin/bash"
```

# 11. Start / Stop / Remove via GUI

- Once you have a container created you can perform on it with control buttons:
  - Start / Stop
  - Delete
  - Open in Terminal
  - Open in Browser (for containers with HTTP srv.)



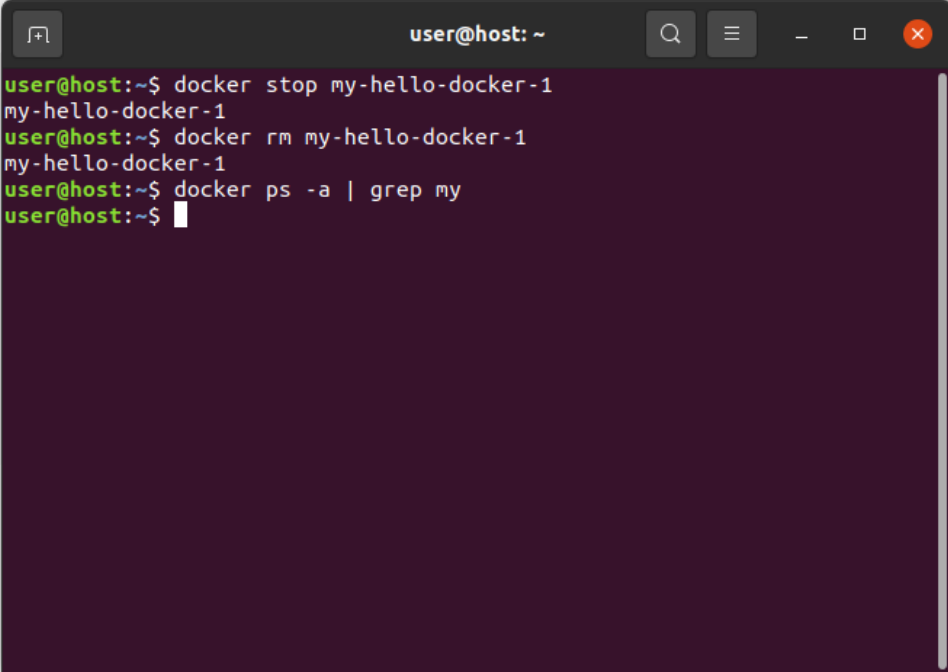
# 12. Stop / Remove via CLI

- Stop container:

```
docker stop my-hello-docker-1
```

- Remove container:

```
docker rm my-hello-docker-1
```

A terminal window with a dark purple background and white text. The window title is 'user@host: ~'. It shows a sequence of Docker commands and their outputs: 'docker stop my-hello-docker-1' followed by 'my-hello-docker-1', 'docker rm my-hello-docker-1' followed by 'my-hello-docker-1', and 'docker ps -a | grep my' which returns no output. The prompt 'user@host:~\$' is visible at the end of each command line.

```
user@host:~$ docker stop my-hello-docker-1
my-hello-docker-1
user@host:~$ docker rm my-hello-docker-1
my-hello-docker-1
user@host:~$ docker ps -a | grep my
user@host:~$
```

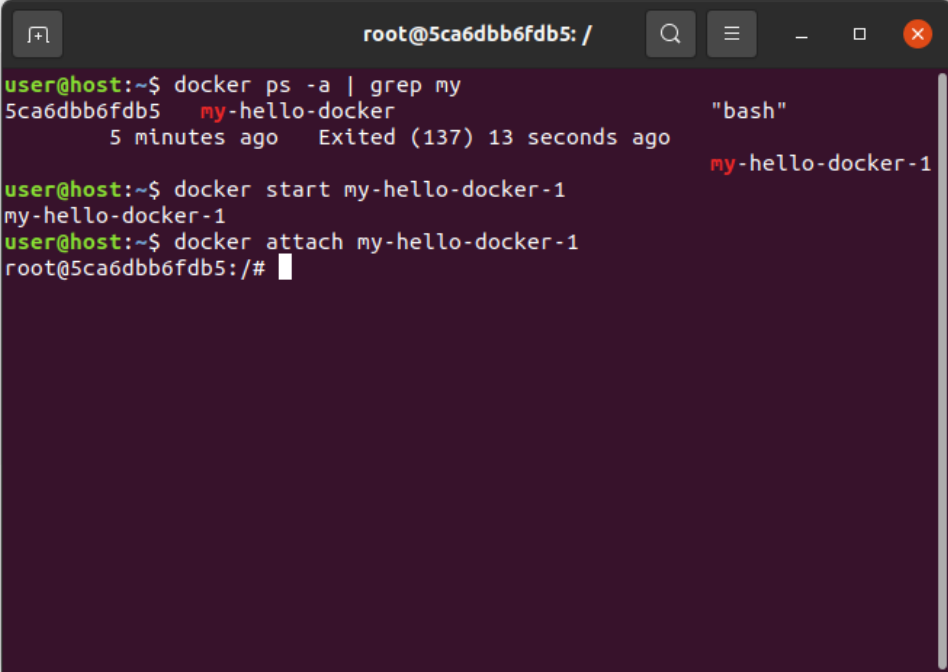
# 13. Start and Attach via CLI

- Start container:

```
docker start my-hello-docker-1
```

- Attach to terminal of the container:

```
docker attach my-hello-docker-1
```

A terminal window with a dark purple background. The title bar shows 'root@5ca6dbb6fdb5: /'. The terminal content shows a user at a host running 'docker ps -a | grep my', which lists a container 'my-hello-docker' that exited. Then, the user runs 'docker start my-hello-docker-1' and 'docker attach my-hello-docker-1', resulting in a root shell inside the container.

```
root@5ca6dbb6fdb5: /  
user@host:~$ docker ps -a | grep my  
5ca6dbb6fdb5  my-hello-docker  "bash"  
              5 minutes ago  Exited (137) 13 seconds ago  my-hello-docker-1  
  
user@host:~$ docker start my-hello-docker-1  
my-hello-docker-1  
user@host:~$ docker attach my-hello-docker-1  
root@5ca6dbb6fdb5:/#
```

# 14. CLI Reference Help

- CLI root reference help is on URL:

<https://docs.docker.com/engine/reference/commandline/docker/>

