

LAB – Lightweight virtualization

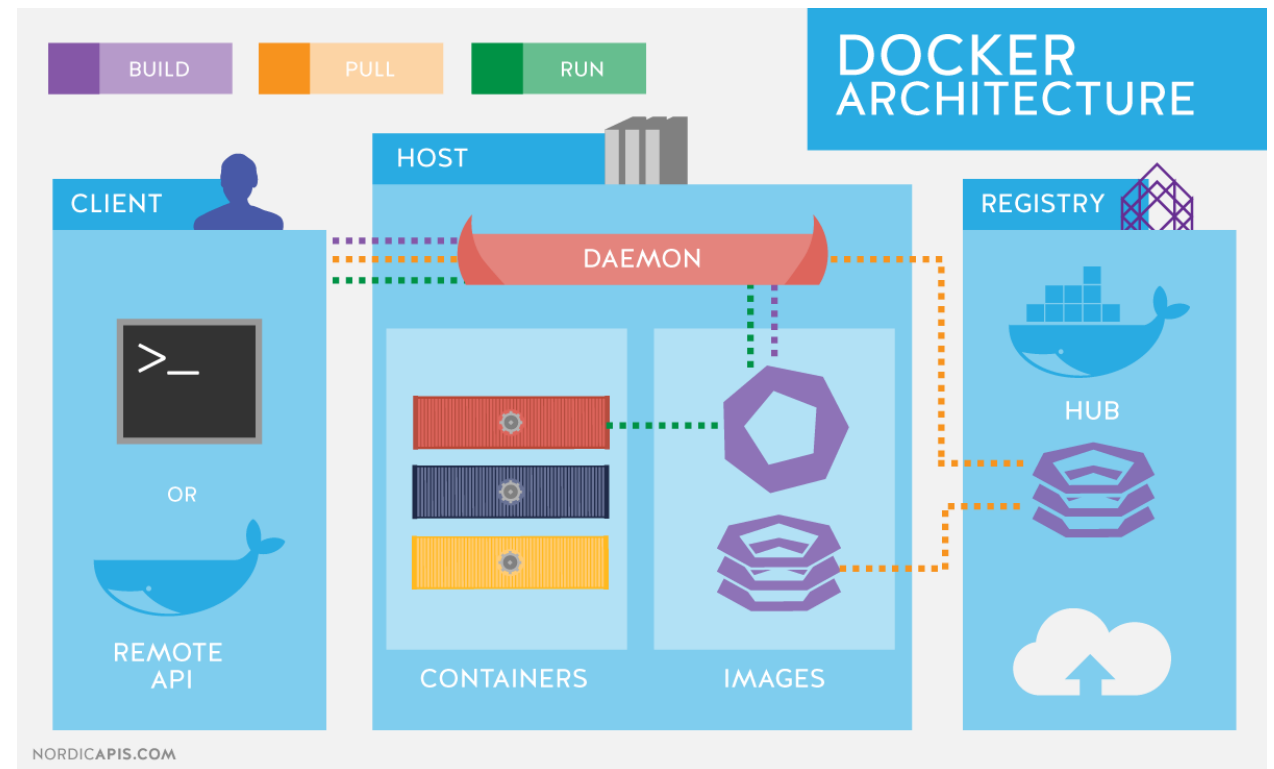
Hands on experience with Docker

References:

- Docker documentation

Docker

- Docker is a container technology for Linux that allows a developer to package up an application with all of the parts it needs.
- Containers are created from locally available images. Images are created from standard images that can be downloaded from public repositories. All operations are managed by the Docker daemon.



Install Docker

- Install some pre-requisites

```
sudo apt install apt-transport-https ca-certificates curl  
software-properties-common
```

- Add the key of the official docker repository in the system

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |  
sudo apt-key add -
```

- Add the Docker repository

```
sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu bionic stable"
```

- Install Docker

```
sudo apt install docker-ce
```

Post-installation

- Check that the installation was successful

```
sudo systemctl status docker
```

```
root@HAJJVX80PD7M5Q0:~# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-01-27 16:41:04 WET; 33s ago
     Docs: https://docs.docker.com
  Main PID: 6854 (dockerd)
    Tasks: 9
   CGroup: /system.slice/docker.service
           └─6854 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

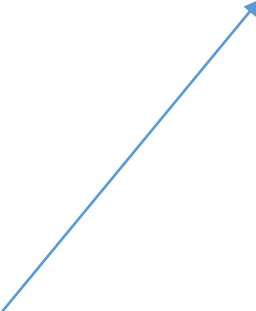
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.443420453Z" level=warning msg="Your kernel does not s
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.443606168Z" level=warning msg="Your kernel does not s
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.443706376Z" level=warning msg="Your kernel does not s
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.443943894Z" level=info msg="Loading containers: start
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.624997547Z" level=info msg="Default bridge (docker0)
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.713003424Z" level=info msg="Loading containers: done.
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.776596865Z" level=info msg="Docker daemon" commit=633
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.777144808Z" level=info msg="Daemon has completed init
Jan 27 16:41:04 HAJJVX80PD7M5Q0 systemd[1]: Started Docker Application Container Engine.
Jan 27 16:41:04 HAJJVX80PD7M5Q0 dockerd[6854]: time="2020-01-27T16:41:04.852107551Z" level=info msg="API listen on /var/run/do
lines 1-19/19 (END)
```

Run a container

- Run the first hello world container:

docker run hello-world

The image of the container was not available locally, before instantiating the container an image was downloaded from the Docker repository



```
root@HAJJVX80PD7M5Q0:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

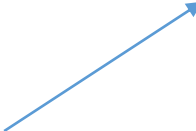
For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Repository

- The set of images available from the repository can be retrieve via the following command

```
docker search ubuntu
```

The Ubuntu image is an image of a complete Ubuntu OS running into a container



```
root@HAJJVX80PD7M5Q0:~# docker search ubuntu
```

NAME	DESCRIPTION	STARS
OFFICIAL		
automated		
ubuntu	Ubuntu is a Debian-based Linux operating sys...	10420
[OK]		
dorowu/ubuntu-desktop-lxde-vnc	Docker image to provide HTML5 VNC interface ...	385
[OK]		
rastasheep/ubuntu-sshd	Dockerized SSH service, built on top of offi...	240
[OK]		
consol/ubuntu-xfce-vnc	Ubuntu container with "headless" VNC session...	208
[OK]		
ubuntu-upstart	Upstart is an event-based replacement for th...	103
[OK]		
ansible/ubuntu14.04-ansible	Ubuntu 14.04 LTS with ansible	98
[OK]		
neurodebian	NeuroDebian provides neuroscience research s...	63
[OK]		
1and1internet/ubuntu-16-nginx-php-phpmyadmin-mysql-5	ubuntu-16-nginx-php-phpmyadmin-mysql-5	50
[OK]		
ubuntu-debootstrap	debootstrap --variant=minbase --components=m...	42
[OK]		
nuagebec/ubuntu	Simple always updated Ubuntu docker images w...	24
[OK]		
i386/ubuntu	Ubuntu is a Debian-based Linux operating sys...	18

Download and run an image

- The Ubuntu image can be downloaded:

```
docker pull ubuntu
```

```
root@HAJJVX80PD7M5Q0:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5c939e3a4d10: Pull complete
c63719cdbe7a: Pull complete
19a861ea6baf: Pull complete
651c9d2d6c4f: Pull complete
Digest: sha256:8d31dad0c58f552e890d68bbfb735588b6b820a46e459672d96e585871acc110
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

```
root@HAJJVX80PD7M5Q0:~# docker images
REPOSITORY      TAG              IMAGE ID          CREATED           SIZE
ubuntu          latest          ccc6e87d482b     11 days ago      64.2MB
hello-world     latest         fce289e99eb9     13 months ago    1.84kB
```

- A new container based on the Ubuntu image can be run

```
docker run -it ubuntu
```

- A new prompt (the container) is shown

Run the container!

List containers

- List all the containers (stopped and running)

`docker ps -a`

```
root@HAJJVX80PD7M5Q0:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS              PORTS          NAMES
555a23589532   ubuntu        "/bin/bash"             28 minutes ago Exited (0) 28 minutes ago          adoring_keldysh
f72fdd116842   hello-world    "/hello"                 36 minutes ago Exited (0) 36 minutes ago          pensive_ishizaka
```

- Remove containers

`docker rm ID`

List images

- List images locally available

`docker images`

```
root@HAJJVX80PD7M5Q0:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest             ccc6e87d482b       11 days ago        64.2MB
hello-world         latest             fce289e99eb9       13 months ago      1.84kB
```

- Delete an image

`docker rmi ID`

```
root@HAJJVX80PD7M5Q0:~# docker rmi fce289e99eb9
Untagged: hello-world:latest
Untagged: hello-world@sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f
Deleted: sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e
Deleted: sha256:af0b15c8625bb1938f1d7b17081031f649fd14e6b233688eea3c5483994a66a3
```

Define a custom image

- A custom docker image can be created from an existing one
- To this aim a Dockerfile must be created to describe the set of customizations to be performed
- For instance create an Ubuntu image that includes a python interpreter and run a python program at startup

Try to write a simple python program 'hello world' and run it in a container (see next slide)

```
# Start from an official ubuntu image
FROM ubuntu
# Specify the command to be run inside the container at installation
RUN apt-get update && apt-get install -y python3
# Add some file to the image from the host
ADD ./my_program.py /root/
# Run at startup the following command
CMD ["/usr/bin/python3", "/root/my_program.py"]
```

Build and run the image

- To build and run the image lunch the following commands

```
docker build -t custom-ubuntu .
```

```
docker run custom-ubuntu
```

```
root@HAJJVX80PD7M5Q0:~# docker build -t custom-ubuntu .
Sending build context to Docker daemon 41.47kB
Step 1/4 : FROM ubuntu
----> ccc6e87d482b
Step 2/4 : RUN apt-get update && apt-get install -y python3
----> Running in 1bd27eb402c2
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [6779 B]
Get:3 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [795 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [807 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packages [25.2 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:9 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [10.8 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [1092 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [39.9 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [1340 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic-backports/main amd64 Packages [2496 B]
Get:18 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 Packages [4243 B]
Fetched 17.5 MB in 7s (2532 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
```

```
Setting up libpython3-stdlib:amd64 (3.6.7-1~18.04) ...
Setting up python3 (3.6.7-1~18.04) ...
running python rtupdate hooks for python3.6...
running python post-rtupdate hooks for python3.6...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Removing intermediate container 1bd27eb402c2
----> 08f3a525fd4f
Step 3/4 : ADD ./my_program.py /root/
----> 11224ec1b8e2
Step 4/4 : CMD ['python3', '/root/my_program.py']
----> Running in 410ba61c6b93
Removing intermediate container 410ba61c6b93
----> d234f53968e1
Successfully built d234f53968e1
Successfully tagged custom-ubuntu:latest
```

```
root@HAJJVX80PD7M5Q0:~# docker images

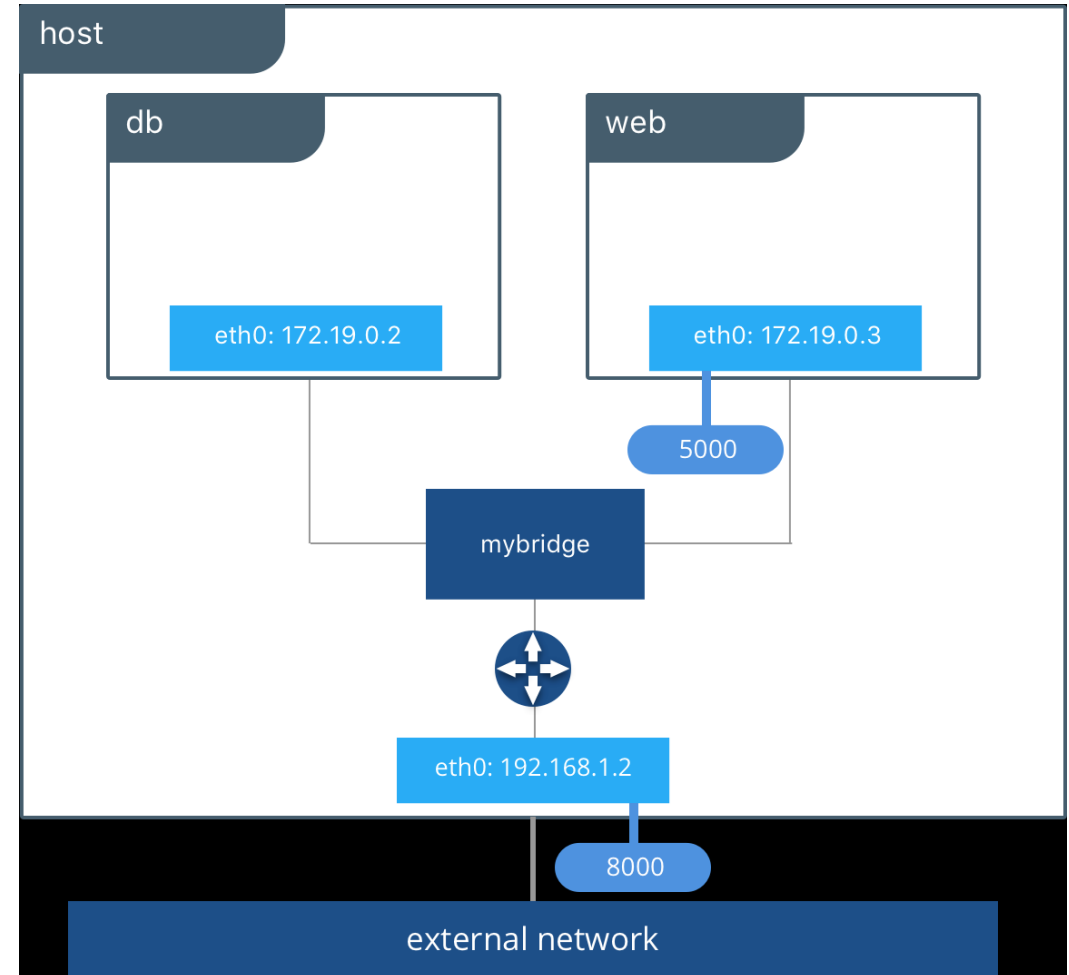

| REPOSITORY    | TAG    | IMAGE ID     | CREATED       | SIZE   |
|---------------|--------|--------------|---------------|--------|
| custom-ubuntu | latest | d234f53968e1 | 3 minutes ago | 127MB  |
| ubuntu        | latest | ccc6e87d482b | 12 days ago   | 64.2MB |


```

```
root@HAJJVX80PD7M5Q0:~# docker run custom-ubuntu
Hello, world!
```

Docker Networking

- The Docker engine creates a virtual network to allow communication among different containers.
- In the default mode, for every container a virtual network adapter is created
- Every virtual interface have a private IPv4 address, communication among different containers can happen through routing or through L2 forwarding if interfaces are bridged (as in the picture).
- A service listening on one port can be published to the external network, by using port forwarding



Expose a service

- To expose a service running on a container the EXPOSE command in the Dockerfile can be used
- Create a new image with a web server and expose port 80

```
# Start from an official Ubuntu image
FROM ubuntu
# Specify the command to be run inside the container at installation
RUN apt-get update && apt-get install -y nginx
# Expose port 80 where the web server runs
EXPOSE 80
# Run at startup the following command
CMD ["nginx", "-g", "daemon off;"]
```

Run and expose the container

- Run the container and expose the port publicly through the command:

```
docker run -p 80:80 nginx-ubuntu
```

- The command run does not exit, open another terminal and check the status via `docker ps`

```
root@HAJJVX80PD7M5Q0:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a54a9affbf8d	nginx-ubuntu	"nginx -g 'daemon of..."	7 seconds ago	Up 6 seconds	0.0.0.0:80->80/tcp	affectionate_williamson

- Check the server

```
curl http://127.0.0.1/
```

Try to deploy the container and check!

```
root@HAJJVX80PD7M5Q0:~# curl http://127.0.0.1/
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Run in background

- Run the docker in detachment mode

```
docker run -p 80:80 -d nginx-Ubuntu
```

```
root@HAJJVX80PD7M5Q0:~/nginx# docker run -p 80:80 -d nginx-ubuntu  
acba68e0d30dc74a5af0562625eb8346e49410ce004debc647960366956cbba3
```

```
root@HAJJVX80PD7M5Q0:~/nginx# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
acba68e0d30d	nginx-ubuntu	"/bin/sh -c 'tail -f...'"	5 seconds ago	Up 3 seconds	0.0.0.0:80->80/tcp	bold_roentgen

- Stop the execution via

```
docker stop ID
```

Communication among container

- Container running in the same host can communicate via their private IP address on the virtual network of containers
- Retrieve the list of IP and corresponding container via:

`docker network inspect bridge`

- The output shows the IP addresses of every container
- Try to ping one of them from the host

```
root@HAJJVX80PD7M5Q0:~/python-flask-server# docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "cec43870494c5e43aaeedf06d8b1977fcb8ca58eb7dfdac358eeae301193e648",
    "Created": "2020-01-27T16:41:04.625040851Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "acba68e0d30dc74a5af0562625eb8346e49410ce004debc647960366956cbba3": {
        "Name": "bold_roentgen",
        "EndpointID": "86006663e0d6e0be92f82e3914825f4d7fa81fce1233f43a799cbfc36d15aaf6",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      },
      "b637d91f2d185aae897aa4f97a6c77382206719d30c542e5d70dc8f41957a7eb": {
        "Name": "mysql",
        "EndpointID": "5549b9e06189f3d7ec22b6095afc8d7c1603ec3be54dc02d0b3b463fe0e0b90d",
        "MacAddress": "02:42:ac:11:00:04",
        "IPv4Address": "172.17.0.4/16",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
```