

# Systems Advanced Docker Containers

---

Docker images bouwen en  
verspreiden



Elfde-Liniestraat 24, 3500 Hasselt, [www.pxl.be](http://www.pxl.be)



# Voorbereiding

Als je PowerShell gebruikt, installeren we de volgende tools, met Scoop

- touch: `scoop install touch`
- nano: `scoop install nano` of `scoop install vim` om vim te installeren :)  
maar je kan ook gewoon visual studio code gebruiken
- git: `scoop install git`

zorg dat Docker Desktop runt

## Lesmateriaal via github

Deze docker les gebruikt lesmateriaal zoals Dockerfiles en source code, dat je kan terugvinden in onze github repo <https://github.com/PXL-Systems-Advanced/docker-lessen>.

Clone deze repo naar je laptop

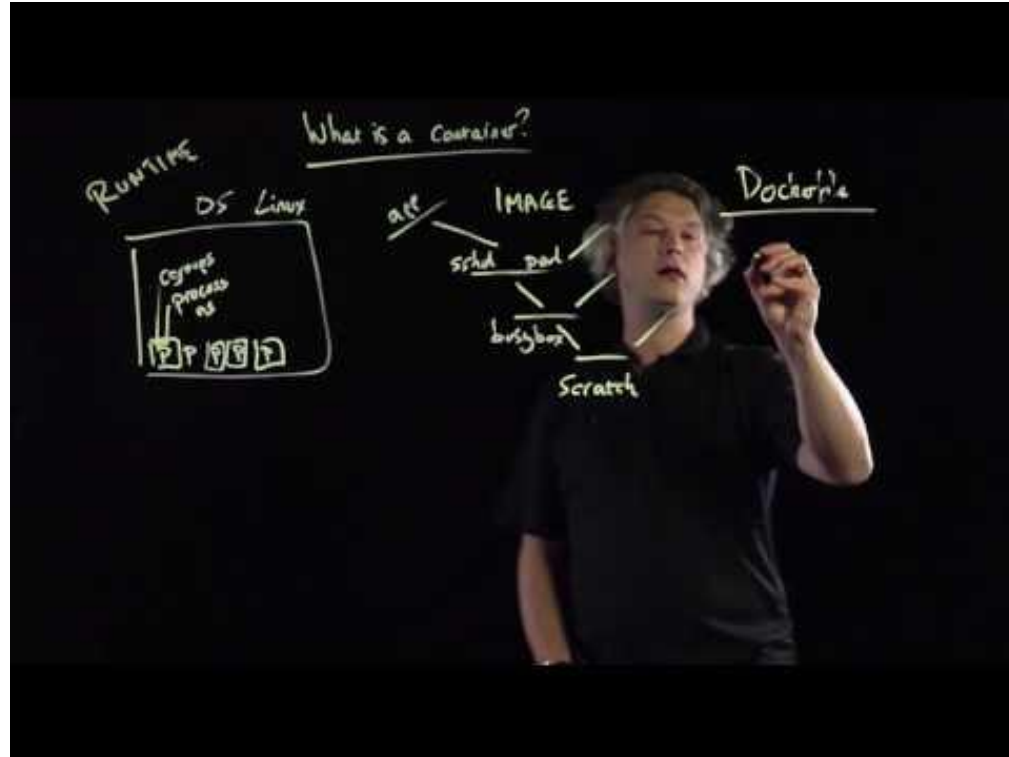
- `git clone https://github.com/PXL-Systems-Advanced/docker-lessen.git`

# Hoe werken containers in linux?

Ben Corrie

"VMWare Senior Staff Engineer, lead architect behind the Kubernetes integration in vSphere"

<https://youtu.be/EnJ7qX9fk>  
[cU?t=504](#) (8:24 tot 18:23)



# Docker images bouwen met Dockerfile

We bouwen een docker image met een "Dockerfile" bestand.

## Dockerfile

- de filenaam is by default **Dockerfile**
- is een plaintext file
- heeft een simpele opmaak
- bevindt zich in een aparte directory per container project
- bevat instructies die aangeven hoe een container image moet gemaakt worden (builden)
- het commando **docker build <DIRECTORY>** wordt gebruikt om de image te builden
- als je een andere filenaam dan "Dockerfile" wil gebruiken, gebruik dan **docker build <DIRECTORY> -f <DOCKER\_FILENAME>**

# Een Dockerfile maken

We maken een dockerfile die zich baseert op de docker image van Ubuntu 20.04.

```
mkdir dockertest && cd dockertest
```

```
touch hostfile.txt
```

```
nano of code Dockerfile
```

```
# Ubuntu-based Image
```

- commentaar

```
FROM ubuntu:20.04
```

- **FROM** is een instructie, dus best in hoofdletters
- moet de eerste instructie zijn in de Dockerfile
- **elke Dockerfile instructie (FROM, RUN, ADD, ...) maakt een nieuwe Image-layer**

```
ADD hostfile.txt /tmp/
```

- kopieert files, dirs, maar ook remote urls van de host naar de Docker Image

```
RUN apt update && apt -y upgrade && apt -y install iputils-ping
```

- zijn commando's die uitgevoerd worden tijdens het bouwen (meestal om extra applicaties te installeren)

```
Dockerfile  X
C: > Users > thraa > docker-les > dockertest > Dockerfile > ...
1 # Ubuntu-based Image
2 FROM ubuntu:20.04
3 ADD hostfile.txt /tmp/
4 RUN apt update && apt -y upgrade && apt -y install iputils-ping
5 |
```

# Een Dockerfile maken

```
Dockerfile
C: > Users > thraa > docker-les > dockertest > Dockerfile > ...
1 # Ubuntu-based Image
2 FROM ubuntu:20.04
3 ADD hostfile.txt /tmp/
4 RUN apt update && apt -y upgrade && apt -y install iputils-ping
5 CMD ["echo", "Container based on Image from Gert Van Waeyenberg"]
6
```

We maken een Dockerfile die zich baseert op de docker image van Ubuntu 20.04

**CMD ["echo", "Container based on Image from Gert Van Waeyenberg"]**

OF **CMD echo "Container based on Image from Gert Van Waeyenberg"**

- **CMD** is als **RUN**, maar de **CMD** instructie wordt niet uitgevoerd bij het builden van de image, maar wel bij het opstarten van een container die gebaseerd is op deze image
  - Wordt overruled indien we bij het runnen van een container een commando meegeven
    - **docker run ... bash**
- Maximum één **CMD** instructie per Dockerfile
- Kan in twee vormen genoteerd worden
  - shell vorm
    - **CMD echo Hallo \$var1**
    - Deze vorm doet aan shell expansion
  - exec vorm
    - **CMD ["command", "arg1"]**
    - Hier geen shell expansion en geen speciale karakters toegelaten (&&, ||, >, ...)
    - hier geen single quotes, maar enkel double quotes rondom tekst
    - ook gebruikt als ENTRYPOINT argumenten (zie latere slide)

# Een Dockerfile maken

We bouwen de Image op basis van deze Dockerfile met `docker build -t <TAG> <DIRECTORY>`

`docker build -t testimage:0.1`

- `-t` om tags mee te geven (hier met een zelf gekozen versie string 0.1)
- deze naam mag **GEEN hoofdletters** bevatten
- **het punt geeft aan dat de Dockerfile in de huidige directory staat**
- indien nodig wordt de ubuntu:20.04 Image gedownload
- in de logging erboven zien we
  - **[1/3] FROM** er gaat vertrokken worden van de image ubuntu:20.04 en het Image-shortid wordt getoond
  - **[2/3] ADD** kopieert de gevraagde files en er wordt een Image-layer gecommit.
  - **[3/3] RUN** apt update... - er wordt een nieuwe container aangemaakt. De updates worden uitgevoerd. Een nieuwe image-layer wordt gecommit en de container wordt weggegooid.
  - Het **CMD** commando maakt **geen** nieuwe image layer, maar wijzigt enkel de image metadata om mee te geven wat het start proces is.
  - De layers worden samengevoegd in een image met een tag.

We kunnen nu een container starten van onze nieuwe Image.

- `docker run testimage:0.1` geeft de output "Container based on Image from Gert Van Waeyenberg" en de container wordt afgesloten
- `docker run -it testimage:0.1 bash` nu kan je in deze container ook het ping commando gebruiken

```
# thraa @ DESKTOP-TOMC in ~\docker-les\dockertest [15:16:09]
$ docker build -t testimage:0.1 .
[+] Building 72.7s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 236B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:20.04
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 33B
=> [1/3] FROM docker.io/library/ubuntu:20.04@sha256:9c2004872a3
=> => resolve docker.io/library/ubuntu:20.04@sha256:9c2004872a3
```

```
# thraa @ DESKTOP-TOMC in ~\docker-les\dockertest [15:30:54]
$ docker run testimage:0.1
Container based on Image from Gert Van Waeyenberg

# thraa @ DESKTOP-TOMC in ~\docker-les\dockertest [15:38:20]
$ docker run -it testimage:0.1 bash
root@6d7ff869ad12:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=37 time=21.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=37 time=20.4 ms
```

testimage	IN USE	0.1	95260ef0e114	8 minutes ago	121 MB
-----------	--------	-----	--------------	---------------	--------

# Layers van een docker image bekijken

Je kan de layers bekijken met  
**docker history <IMAGE>**

Je kan dezelfde informatie ook bekijken  
in Docker Desktop, Images tab.

The image shows two ways to inspect Docker image layers. The top part is a terminal window showing the output of the `docker history` command for the `testimage:0.1` image. The bottom part is a screenshot of the Docker Desktop interface, specifically the 'Images' tab, where the 'testimage:0.1' image is selected. A red circle highlights the 'More actions' menu (three dots) next to the image, which contains options like 'Inspect', 'Pull', 'Push to Hub', and 'Remove'.

**Terminal Output:**

```
# thraa @ DESKTOP-TOMC in ~\docker-les\dockertest [15:46:25]
$ docker history testimage:0.1
IMAGE          CREATED          CREATED BY                                      SIZE      COMMENT
95260ef0e114   15 minutes ago  CMD ["echo" "Container based on Image from G...  0B        buildkit.dockerfile.v0
<missing>      15 minutes ago  RUN /bin/sh -c apt update && apt -y upgrade ...  48.2MB    buildkit.dockerfile.v0
<missing>      16 minutes ago  ADD hostfile.txt /tmp/ # buildkit              0B        buildkit.dockerfile.v0
<missing>      2 weeks ago    /bin/sh -c #(nop) CMD ["bash"]                 0B
<missing>      2 weeks ago    /bin/sh -c #(nop) ADD file:8faed18d471598732...  72.8MB
```

**Docker Desktop Interface:**

Image Name	Status	Version	Image ID	Created	Size	Actions
testimage	IN USE	0.1	95260ef0e114	8 minutes ago	121 MB	Inspect, Pull, Push to Hub, Remove
tomcoolpxl/python-counter		latest	9daf59af6cae	4 days ago	915.05 MB	
ubuntu		latest	216c552ea5ba	16 days ago	77.84 MB	
ubuntu		16.04	b6f507652425	about 1 year ago	134.82 MB	

**Docker Desktop Details for testimage:0.1:**

IMAGE ID: 95260ef0e114, CREATED: 8 minutes ago, SIZE: 121 MB

**IMAGE HISTORY:**

Layer	Command	Size
0	CMD ["echo" "Container based on Image from Gert Van Waeyenberg"]	0 Bytes
1	RUN /bin/sh -c apt update && apt -y upgrade && apt -y install iputils-...	48.22 MB
2	ADD hostfile.txt /tmp/ # buildkit	0 Bytes
3	/bin/sh -c #(nop) CMD ["bash"]	0 Bytes
4	/bin/sh -c #(nop) ADD file:8faed18d471598732aa3816c8f70e22716f...	72.78 MB

**COMMAND:** CMD ["echo" "Container based on Image from Gert Van Waeyenberg"]



# Dockerfile - ENTRYPOINT

Als **ENTRYPOINT** bestaat definieert dat het container process ipv CMD en wat in CMD staat worden de argumenten voor dat proces.

De inhoud van de **CMD** instructie (als het aanwezig is) wordt als parameter toegevoegd aan de **ENTRYPOINT** instructie.

Het commando dat we achter het **docker run** commando geven, wordt als parameter doorgegeven aan de **ENTRYPOINT** instructie.

**ENTRYPOINT** kan enkel overriden worden at runtime als er expliciet **--entrypoint** wordt meegegeven aan het **docker run** commando.

Bijvoorbeeld, we maken een Dockerfile met een **ENTRYPOINT**

```
FROM ubuntu:20.04
```

```
CMD ["Hello, World!"]
```

```
ENTRYPOINT ["echo"]
```

- Aangezien **ENTRYPOINT** bestaat, wordt **ENTRYPOINT** het hoofdproces ("echo") en alles wat in **CMD** staat wordt het argument van **ENTRYPOINT**.
- Als we argumenten meegeven aan **docker run**, komen die in de plaats van **CMD** en worden dus argumenten van **ENTRYPOINT**.
- We kunnen het **ENTRYPOINT** proces overriden met **--entrypoint**

```
# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test2 [16:09:17]
$ cat .\Dockerfile
FROM ubuntu:20.04
CMD ["Hello, World!"]
ENTRYPOINT ["echo"]

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test2 [16:09:23]
$ docker build -t test .
[+] Building 0.7s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:20.04
=> CACHED [1/1] FROM docker.io/library/ubuntu:20.04@sha256:9c2004872a3a9fcec
=> exporting to image
=> => exporting layers
=> => writing image sha256:bac07d073f5b81051922a4012d4e74f958cb20c099468f633
=> => naming to docker.io/library/test

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test2 [16:09:28]
$ docker run test
Hello, World!

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test2 [16:09:32]
$ docker run test Hallo PXL!
Hallo PXL!

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test2 [16:09:36]
$ docker run -it --entrypoint /bin/bash test
root@7f87a0eee339:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr
root@7f87a0eee339:/# exit
exit

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test2 [16:12:46]
$ -
```

# Dockerfile - ENV variabelen

ENV kan gebruikt worden om environment variables te definiëren in de container.

Dockerfile voorbeeld:

FROM alpine:latest

ENV var1="ping" var2="-c 5" var3="8.8.8.8"

CMD \$var1 \$var2 \$var3

```
# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test3 [16:21:23]
$ cat .\Dockerfile
FROM alpine:latest
ENV var1="ping" var2="-c 5" var3="8.8.8.8"
CMD $var1 $var2 $var3

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test3 [16:21:25]
$ docker build -t pingtest .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 31B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 0.0s
=> CACHED [1/1] FROM docker.io/library/alpine:latest 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:3969a31cbf963e0ed9fce6d17abed8df3b5dd9793f6105cc063868440dd3bd8 0.0s
=> => naming to docker.io/library/pingtest 0.0s

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test3 [16:21:27]
$ docker run pingtest
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=37 time=24.624 ms
64 bytes from 8.8.8.8: seq=1 ttl=37 time=18.653 ms
64 bytes from 8.8.8.8: seq=2 ttl=37 time=18.018 ms
64 bytes from 8.8.8.8: seq=3 ttl=37 time=15.398 ms
64 bytes from 8.8.8.8: seq=4 ttl=37 time=18.812 ms

--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 15.398/19.101/24.624 ms

# thraa @ DESKTOP-TOMC in ~\docker-les\docker-test3 [16:21:40] C:1
$
```

# Dockerfile - USER

We builden de image en runnen die om onder andere de user en de working directory te bekijken.

```
# thraa @ DESKTOP-TOMC in ~\docker-les\python-test2 [18:33:45]
$ docker run -it userdemo bash
pythonuser@c026f07bc94b:~$ ls
hello.py
pythonuser@c026f07bc94b:~$ ls /root
ls: cannot open directory '/root': Permission denied
pythonuser@c026f07bc94b:~$ sudo ls /root
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
[sudo] password for pythonuser:
pythonuser@c026f07bc94b:~$ pwd
/home/pythonuser
pythonuser@c026f07bc94b:~$ tail -1 /etc/group
pythonuser:x:999:
pythonuser@c026f07bc94b:~$ tail -1 /etc/passwd
pythonuser:x:999:999:Python user:/home/pythonuser:/sbin/nologin
pythonuser@c026f07bc94b:~$ whoami
pythonuser
pythonuser@c026f07bc94b:~$ su - pythonuser
Password:
su: failed to execute /sbin/nologin: No such file or directory
pythonuser@c026f07bc94b:~$ exit
exit
```

```
# thraa @ DESKTOP-TOMC in ~\docker-les\python-test2 [18:35:12] C:127
$ docker run userdemo
Hello, PXL!
```

```
# thraa @ DESKTOP-TOMC in ~\docker-les\python-test2 [18:35:20]
$ _
```

# Dockerfile - ADD/COPY

**ADD** kan gebruikt worden om bestanden aan een image toe te voegen. **COPY** doet hetzelfde, maar **ADD** kan ook overweg met URL's.

Bijvoorbeeld, we maken een Python programma **hello.py**:

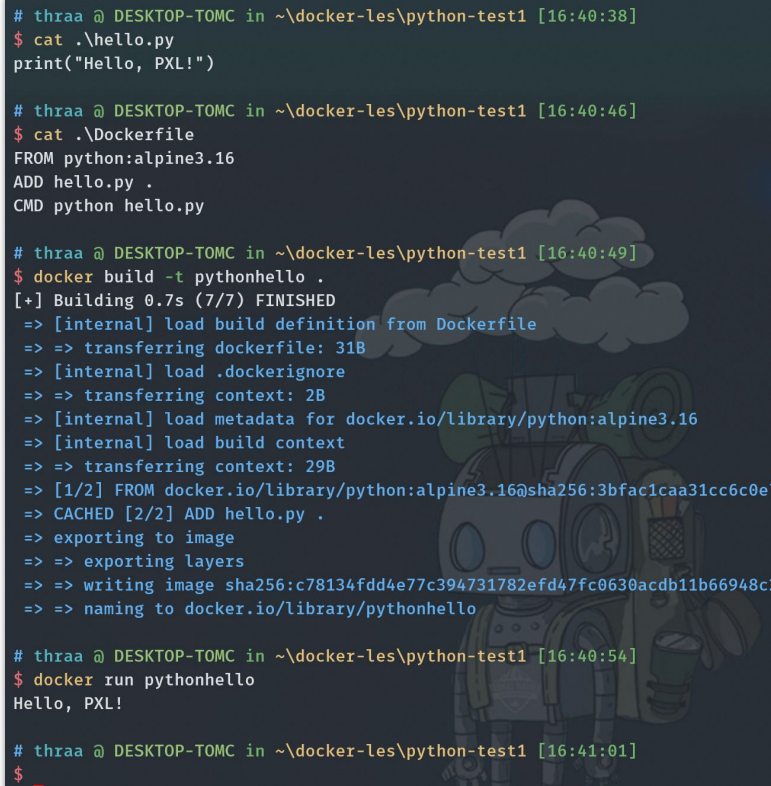
```
print("Hello, PXL!")
```

met een **Dockerfile** die we baseren op een alpine image met een python interpreter.

```
FROM python:alpine3.16
```

```
ADD hello.py .
```

```
CMD python hello.py
```

A terminal window with a dark background and light-colored text. It shows a series of commands and their outputs. The first command is 'cat .\hello.py' which outputs 'print("Hello, PXL!")'. The second command is 'cat .\Dockerfile' which outputs a Dockerfile with 'FROM python:alpine3.16', 'ADD hello.py .', and 'CMD python hello.py'. The third command is 'docker build -t pythonhello .' which outputs a multi-line build progress message including '[+] Building 0.7s (7/7) FINISHED' and details about transferring files and contexts. The fourth command is 'docker run pythonhello' which outputs 'Hello, PXL!'. The prompt '\$' is visible at the bottom.

```
# thraa @ DESKTOP-TOMC in ~\docker-les\python-test1 [16:40:38]
$ cat .\hello.py
print("Hello, PXL!")

# thraa @ DESKTOP-TOMC in ~\docker-les\python-test1 [16:40:46]
$ cat .\Dockerfile
FROM python:alpine3.16
ADD hello.py .
CMD python hello.py

# thraa @ DESKTOP-TOMC in ~\docker-les\python-test1 [16:40:49]
$ docker build -t pythonhello .
[+] Building 0.7s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:alpine3.16
=> [internal] load build context
=> => transferring context: 29B
=> [1/2] FROM docker.io/library/python:alpine3.16@sha256:3bfac1caa31cc6c0e
=> CACHED [2/2] ADD hello.py .
=> exporting to image
=> => exporting layers
=> => writing image sha256:c78134fdd4e77c394731782efd47fc0630acdb11b66948c
=> => naming to docker.io/library/pythonhello

# thraa @ DESKTOP-TOMC in ~\docker-les\python-test1 [16:40:54]
$ docker run pythonhello
Hello, PXL!

# thraa @ DESKTOP-TOMC in ~\docker-les\python-test1 [16:41:01]
$
```

# Dockerfile - USER

**USER** kan gebruikt worden om het container proces te starten met een gebruiker met minder privileges (ipv met **root**).

- De user (en group) moeten in de image ook aangemaakt worden of reeds aanwezig zijn (in **/etc/passwd** en **/etc/group**).
- **WORKDIR** zorgt er voor dat je ook in de homedir van de user komt als je een shell opent.
- Eventuele instructies zoals **RUN**, **CMD**, **ADD/COPY** en **ENTRYPOINT** worden uitgevoerd in deze working directory.
- **RUN** gebruikt hier **&&** om meerdere commando's achter elkaar te hangen. Dan hebben we veel minder commits en dus ook veel minder image-layers. Voor het overzicht kan je met **\** verder gaan op de volgende regel.

```
FROM python:bullseye
RUN apt-get update \
    && apt-get install -y sudo
RUN groupadd -r pythonuser \
    && useradd -r -m -l -g pythonuser -s /sbin/nologin -c "Python user" pythonuser \
    && echo pythonuser:pxl | chpasswd \
    && usermod -aG sudo pythonuser
WORKDIR /home/pythonuser
ADD hello.py .
RUN chown pythonuser:pythonuser hello.py && chmod 755 hello.py
USER pythonuser
CMD python hello.py
```

```
Dockerfile X
C: > Users > thraa > docker-les > python-test2 > Dockerfile > ...
1 FROM python:bullseye
2 RUN apt-get update \
3     && apt-get install -y sudo
4 RUN groupadd -r pythonuser \
5     && useradd -r -m -l -g pythonuser -s /sbin/nologin -c "Python user" pythonuser \
6     && echo pythonuser:pxl | chpasswd \
7     && usermod -aG sudo pythonuser
8 WORKDIR /home/pythonuser
9 ADD hello.py .
10 RUN chown pythonuser:pythonuser hello.py && chmod 755 hello.py
11 USER pythonuser
12 CMD python hello.py
13 |
```

```
# thraa @ DESKTOP-TOMC in ~\docker-les\python-test2 [18:38:18]
$ docker build -t userdemo .
[+] Building 1.4s (12/12) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                                0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/python:bullseye                1.3s
=> [auth] library/python:pull token for registry-1.docker.io                    0.0s
=> [internal] load build context                                                  0.0s
=> => transferring context: 29B                                                  0.0s
=> [1/6] FROM docker.io/library/python:bullseye@sha256:0d6369587f3e0b0254fee8d8d797f9ae3d2b4b4b0841 0.0s
=> CACHED [2/6] RUN apt-get update 66 apt-get install -y sudo                    0.0s
=> CACHED [3/6] RUN groupadd -r pythonuser 66 useradd -r -m -l -g pythonuser -s /sbin/nologin -c 0.0s
=> CACHED [4/6] WORKDIR /home/pythonuser                                         0.0s
=> CACHED [5/6] ADD hello.py .                                                  0.0s
=> CACHED [6/6] RUN chown pythonuser:pythonuser hello.py 66 chmod 755 hello.py 0.0s
=> exporting image                                                              0.0s
=> => writing image sha256:6e5274d7167569e9957e9e1a63b752529e977bfff3dba835a33e4b60680c3c418 0.0s
=> => naming to docker.io/library/userdemo                                       0.0s

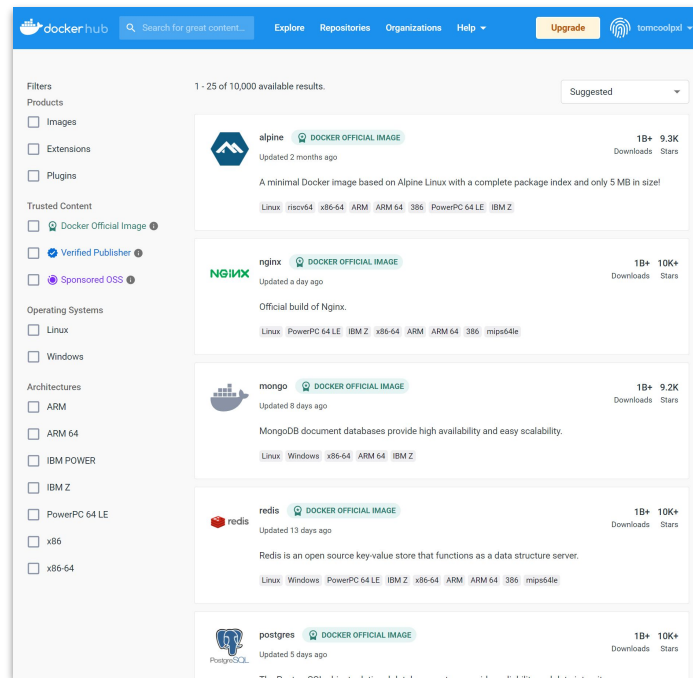
# thraa @ DESKTOP-TOMC in ~\docker-les\python-test2 [18:38:25]
$
```

# Een Dockerfile op dockerhub bekijken

We kunnen Dockerfiles bekijken op dockerhub.

- surf naar <https://hub.docker.com>
- zoek naar "ubuntu"
  - Klik op de titel van de Ubuntu-repo
  - klik op de blauwe link 20.04 onder Dockerfile-links
  - We zien dan de instructies die we bijna allemaal kennen FROM, ADD, CMD
    - scratch is de base image (speciale image met een beetje metadata)
    - dan worden er files uit een een gezippte tarball toegevoegd
- het kan ook ingewikkelder - zoek naar "python"
  - klik op een van de python tags, bv alpine

dockerhub is eigenlijk de Google Play-store of Apple-Appstore van applicatie-containers voor bedrijven om te downloaden en te runnen in hun Docker omgeving.

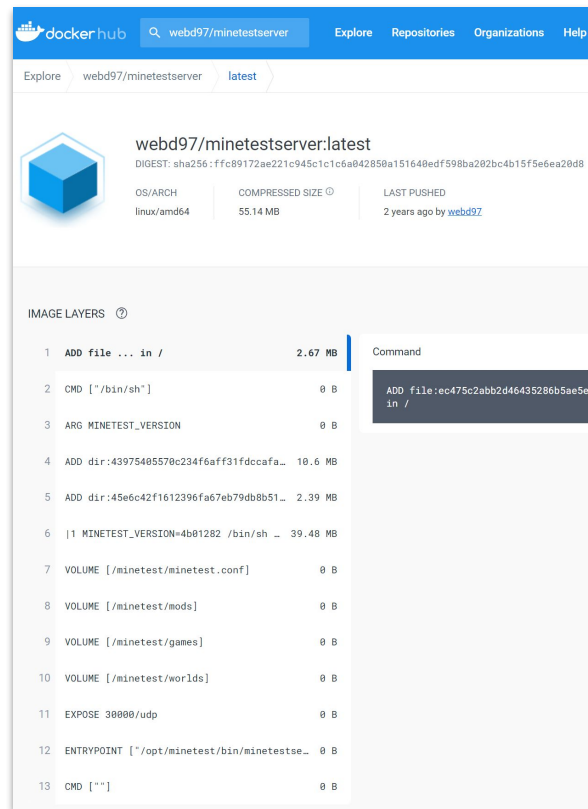




# De publieke registry op dockerhub

Er worden ook heel veel images geupload naar de publieke registry op dockerhub.

- Ga naar <https://hub.docker.com>
  - Klik bovenaan op Explore
  - Je ziet nu de populairste repos
- Klik bovenaan in het Search veld (naast het logo van dockerhub)
  - typ bv redis, zonder op Enter te duwen
  - Nu kijk je lager in de suggestions en je vindt de "Community" entries
  - Die beginnen met de naam van een gewone gebruiker, een slash, en dan de reponame
- Zoek naar webd97/minetestserver
  - bekijk hoe een image is opgebouwd



The screenshot shows the Docker Hub interface for the repository `webd97/minetestserver:latest`. The page includes a search bar at the top with the text `webd97/minetestserver` and navigation links for `Explore`, `Repositories`, `Organizations`, and `Help`. Below the repository name, there is a blue cube icon and a table with metadata: `OS/ARCH` (linux/amd64), `COMPRESSED SIZE` (55.14 MB), and `LAST PUSHED` (2 years ago by webd97). The main section is titled `IMAGE LAYERS` and contains a table with 13 layers. A tooltip is visible over the first layer, showing the command `ADD file:ec475c2abb2d46435286b5ae5ef in /`.

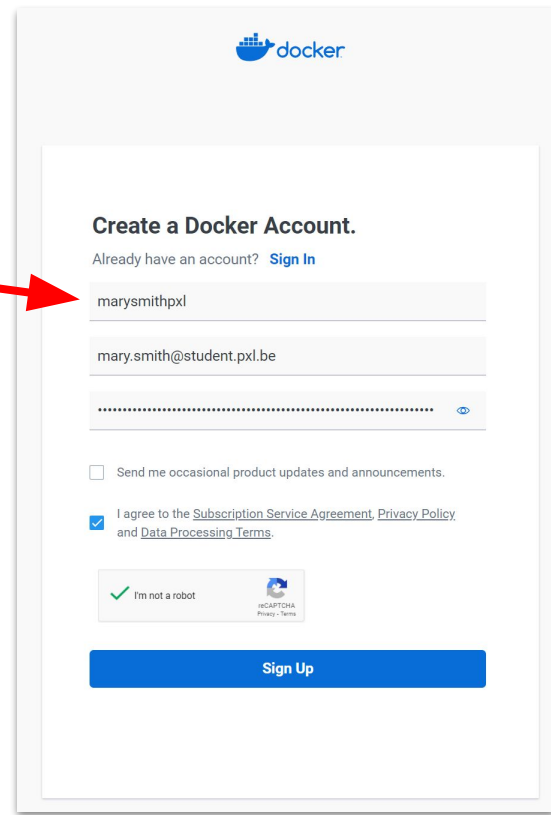
Layer	Command	Size
1	ADD file ... in /	2.67 MB
2	CMD ["/bin/sh"]	0 B
3	ARG MINESTEST_VERSION	0 B
4	ADD dir:43975405570c234f6aff31fdccafa...	10.6 MB
5	ADD dir:45e6c42f1612396fa67eb79db8b51...	2.39 MB
6	[1 MINESTEST_VERSION=4b01282 /bin/sh ...	39.48 MB
7	VOLUME [/minetest/minetest.conf]	0 B
8	VOLUME [/minetest/mods]	0 B
9	VOLUME [/minetest/games]	0 B
10	VOLUME [/minetest/worlds]	0 B
11	EXPOSE 30000/udp	0 B
12	ENTRYPOINT ["/opt/minetest/bin/minetest_...	0 B
13	CMD [""]	0 B

# Jezelf registreren op dockerhub

We kunnen ook een account voor onszelf aanmaken op dockerhub.

Opgelet: de username zal ook gebruikt worden door iedereen die jouw Images wil gebruiken. Kies dus een gepaste naam.

- <https://hub.docker.com>
- Register
- Sign-in



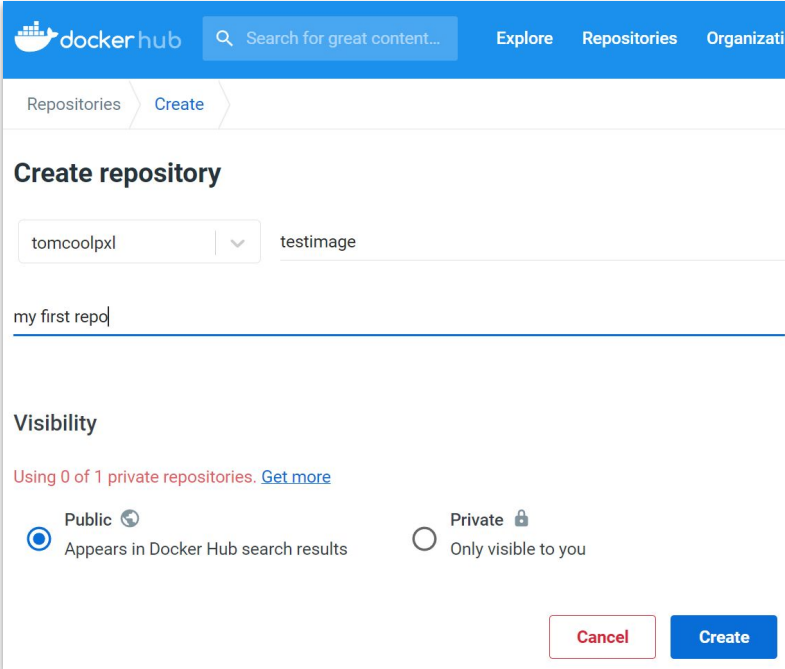
The screenshot shows the Docker Hub registration page. At the top is the Docker logo. The main heading is "Create a Docker Account." Below it, a link says "Already have an account? [Sign In](#)". The form contains three input fields: a username field with "marysmithpxl", an email field with "mary.smith@student.pxl.be", and a password field with masked characters and an eye icon. Below the fields is a checkbox for "Send me occasional product updates and announcements." and a checked checkbox for "I agree to the [Subscription Service Agreement](#), [Privacy Policy](#), and [Data Processing Terms](#)." At the bottom is a reCAPTCHA widget with a green checkmark and the text "I'm not a robot". A blue "Sign Up" button is at the very bottom. A red exclamation mark and arrow point to the username field, indicating a warning or important note.



# Een public repository aanmaken

We kunnen nu een repository aanmaken op dockerhub.

- Eerst maken we de repo aan op dockerhub (maximum 1 private repo zonder te betalen)
- Selecteer "Repositories" bovenaan
- Klik op de "**Create repository**" button
  - Repository name: **testimage**
  - Description: **my first repo**
  - Visibility: **Public**
  - **Create**



The screenshot shows the Docker Hub interface for creating a new repository. The top navigation bar includes the Docker Hub logo, a search bar, and links for 'Explore', 'Repositories', and 'Organizations'. Below the navigation bar, the 'Create repository' form is displayed. It features a dropdown menu for the username 'tomcoolpxl' and a text input field for the repository name 'testimage'. The description field contains the text 'my first repo'. Under the 'Visibility' section, the 'Public' option is selected with a radio button, and the 'Private' option is unselected. The 'Public' option is accompanied by a globe icon and the text 'Appears in Docker Hub search results'. The 'Private' option is accompanied by a lock icon and the text 'Only visible to you'. At the bottom right of the form, there are two buttons: 'Cancel' and 'Create'.

docker hub Search for great content... Explore Repositories Organizations

Repositories Create


Create repository


tomcoolpxl testimage

my first repo

Visibility

Using 0 of 1 private repositories. [Get more](#)

☒ Public  Appears in Docker Hub search results

☐ Private  Only visible to you

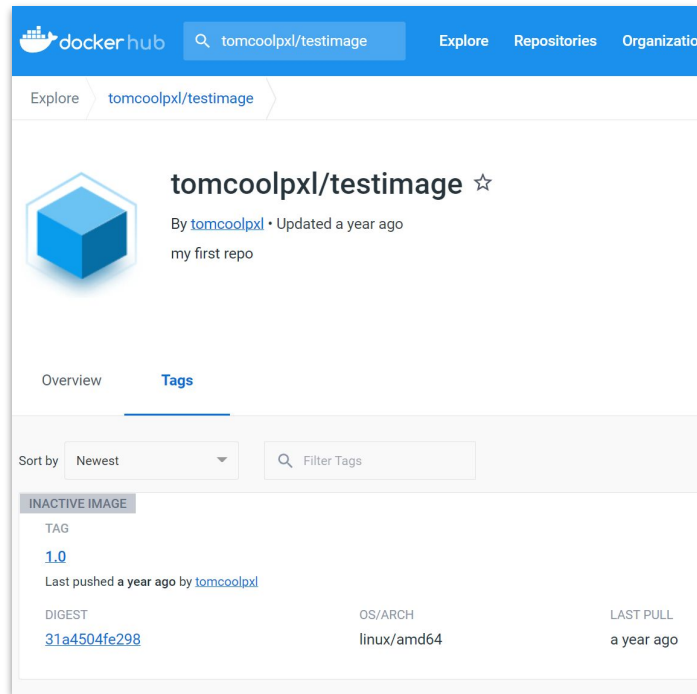
Cancel Create

# Images pushen naar de publieke registry

We kunnen nu ons zelf gemaakt image pushen naar onze repository in de publieke registry dockerhub.

- **docker image ls**
  - we vinden de testimage-image
- **docker history --no-trunc testimage:1.0**
  - om te kijken wat we gedaan hebben
- **docker tag testimage:0.1 <dockerhub\_username>/testimage:1.0**
  - zelfde (user)naam als op Docker-Hub + imagenaam met version-tag
- **docker image ls**
  - we zien de image nu nog eens met de nieuwe tag
- **docker login**
  - inloggen op Docker-Hub om te kunnen uploaden
  - geef "username" en "password"
- **docker push <dockerhub\_username>/testimage:1.0**
  - je kan de melding "Image already pushed, skipping" te zien krijgen
  - dit omdat onze Image gebaseerd is op die van ubuntu en deze reeds bestaat op Docker-Hub
- **docker logout**
- **docker image rm <username>/testimage:1.0**
- **docker image rm testimage:1.0**
  - om uit te loggen bij Docker-Hub en de lokale images te verwijderen
- **docker run <dockerhub\_username>/testimage:1.0**
  - de image wordt opnieuw gepulld van Docker-Hub.

Op deze manier iedereen mekaars docker-image downloaden en gebruiken. Check je image en tags op de dockerhub website.



# Oefening: images maken en pushen

- <https://github.com/PXL-Systems-Advanced/docker-lessen/tree/main/python-oef1>
- `fib.py` vind je in de `./python-oef1/` directory van onze docker-lessen repo
  - `git clone https://github.com/PXL-Systems-Advanced/docker-lessen.git`
- Maak, via een **Dockerfile** zelf een Image, gebaseerd op de image "ubuntu", die het python `fib.py` hiernaast opstart bij het runnen van een container:
  - Noem deze image `fib-app`
- Gooi alle voorgaande containers weg
- Start een nieuwe container gebaseerd op deze image
- Hoe groot is deze image?
- Push deze image naar dockerhub
- Gooi de container die je gestart hebt van de `fib-app`-image weg
- Gooi lokaal je `fib-app`-image weg
- Start een nieuwe container gebaseerd op je `fib-app`-image van jouw dockerhub

Indien je een image zou willen verwijderen van dockerhub, dan kan dat heel makkelijk via de dockerhub website.

- Om een bepaalde versie te verwijderen klik je eerst op het tabblad "tags". Vervolgens klik je het selectievakje aan voor de naam van de tag. Als laatste kies je dan voor "Delete" in het Action-keuzevak.

```
fib.py  X
C: > Users > thraa > docker-les > python-oef1 > fib.py
1 from os import environ
2
3 # default if NTERMS environment variable is empty
4 NTERMS_DEFAULT = "13"
5
6
7 def PrintFibonacci(length):
8     # initial variable for the base case.
9     first = 0
10    second = 1
11
12    # printing the initial Fibonacci number.
13    print(first, second, end=" ")
14
15    # decreasing the length by two because the first 2 Fibonacci numbers
16    # already printed.
17    length -= 2
18
19    # loop until the length becomes 0.
20    while length > 0:
21
22        # printing the next Fibonacci number.
23        print(first + second, end=" ")
24
25        # updating the first and second variables for finding the next number.
26        temp = second
27        second = first + second
28        first = temp
29
30        # decreasing length that states the Fibonacci numbers to be printed more
31        length -= 1
32
33
34 # main program
35 if __name__ == "__main__":
36     # get NTERMS environment variable
37     nterms = int(environ.get('NTERMS', NTERMS_DEFAULT))
38     print("Fibonacci Series ~", nterms, "terms")
39     PrintFibonacci(nterms)
40     pass
```

# Oefening: myping

- Maak, via een **Dockerfile** zelf een docker image, die
  - gebaseerd is op de image "alpine"
  - de naam krijgt "myping"
  - die 5 maal het ip adres van Google DNS pingt
    - als er geen commando wordt opgegeven bij het starten van een container
  - die pingt naar het opgegeven ip adres
    - wanneer er wel een ip adres wordt opgegeven bij het starten van een container

# Oefening: Container met gewone user

- Maak, via een **Dockerfile** zelf een docker image, gebaseerd op de image "ubuntu", die een gewone gebruiker **student** heeft met als primaire groep **student**
  - noem deze image **2tin**
- Installeer de packages **tree** en **sudo** in de image
- Geef **student** en **root** een paswoord
  - tip: **chpasswd**
  - je kan dan **sudo** gebruiken of wisselen naar de user **root** met **su -**
- Je kan de man-pages installeren in de image met het commando **unminimize**
  - tip: het commando **yes** geeft continu een 'y'
  - dit kan je dus gebruiken samen met een pipe naar een volgend commando
  - opgelet: je moet het package **man** ook nog installeren
- Start een nieuwe container genaamd **2tina** van deze image
  - probeer uit te zoeken hoe je de hostname binnen de container kan veranderen naar **ubuntu**
    - via het docker run commando
  - probeer **sudo ls /root**
  - probeer ook eens **root** te worden
  - open de manpage van **ls**

