Changing users and working directory

INTRODUCTION TO DOCKER

Tim Sangster
Software Engineer @ DataCamp

Dockerfile instruction interaction

FROM, RUN, and COPY interact through the file system.

```
COPY /projects/pipeline_v3/start.sh /app/start.sh
RUN /app/start.sh
```

Some influence other instructions directly:

- `WORKDIR` : Changes the working directory for all following instructions
- `USER` : Changes the user for all following instructions

WORKDIR - Changing the working directory

Starting all paths at the root of the file system:

```
COPY /projects/pipeline_v3/ /app/
```

Becomes cluttered when working with long paths:

```
COPY /projects/pipeline_v3/ /home/my_user_with_a_long_name/work/projects/app/
```

Alternatively, use WORKDIR:

```
WORKDIR /home/my_user_with_a_long_name/work/projects/
```

```
COPY /projects/pipeline_v3/ app/
```

RUN in the current working directory

Instead of using the full path for every command:

```
RUN /home/repl/projects/pipeline/init.sh
RUN /home/repl/projects/pipeline/start.sh
```

Set the WORKDIR:

```
WORKDIR /home/repl/projects/pipeline/
RUN ./init.sh
RUN ./start.sh
```

Changing the startup behavior with WORKDIR

Instead of using the full path:

```
CMD /home/repl/projects/pipeline/start.sh
```

Set the WORKDIR:

```
WORKDIR /home/repl/projects/pipeline/
CMD start.sh
```

Overriding command will also be run in WORKDIR:

```
docker run -it pipeline_image start.sh
```

Linux permissions

- Permissions are assigned to users.
- Root is a special user with all permissions.

Best practice

- Use root to create new users with permissions for specific tasks.
- Stop using root.

Changing the user in an image

Best practice: Don't run everything as root

Ubuntu -> root by default

```
FROM ubuntu             --> Root user by default
RUN apt-get update      --> Run as root
```

USER Dockerfile instruction:

```
FROM ubuntu             --> Root user by default
USER repl               --> Changes the user to repl
RUN apt-get update      --> Run as repl
```

Changing the user in a container

Dockerfile setting the user to repl:

```
FROM ubuntu          --> Root user by default
USER repl            --> Changes the user to repl
RUN apt-get update   --> Run as repl
```

Will also start containers with the repl user:

```
docker run -it ubuntu bash
repl@container: whoami
repl
```

Summary

Usage	Dockerfile Instruction
Change the current working directory	WORKDIR <path>
Change the current user	USER <user-name>

Time for practice!

INTRODUCTION TO DOCKER

Variables in Dockerfiles

INTRODUCTION TO DOCKER

Tim Sangster
Software Engineer @ DataCamp

Variables with the ARG instruction

Create variables in a Dockerfile

```
ARG <var_name>=<var_value>
```

For example `ARG path=/home/repl`

To use in the Dockerfile

```
$path
```

For example `COPY /local/path $path`

Use-cases for the ARG instruction

Setting the Python version

```
FROM ubuntu
ARG python_version=3.9.7-1+bionic1
RUN apt-get install python3=$python_version
RUN apt-get install python3-dev=$python_version
```

Configuring a folder

```
FROM ubuntu
ARG project_folder=/projects/pipeline_v3
COPY /local/project/files $project_folder
COPY /local/project/test_files $project_folder/tests
```

Setting ARG variables at build time

```
FROM ubuntu
ARG project_folder /projects/pipeline_v3
COPY /local/project/files $project_folder
COPY /local/project/test_files $project_folder/tests
```

Setting a variable in the build command

```
docker build --build-arg project_folder=/repl/pipeline .
```

ARG is overwritten, and files end up in:

```
COPY /local/project/files /repl/pipeline
COPY /local/project/test_files /repl/pipeline/tests
```

Variables with ENV

Create variables in a Dockerfile

```
ENV <var_name>=<var_value>
```

For example `ENV DB_USER=pipeline_user`

To use in the Dockerfile or at runtime

```
$DB_USER
```

For example `CMD psql -U $DB_USER`

Use-cases for the ENV instruction

Setting a directory to be used at runtime

```
ENV DATA_DIR=/usr/loca/var/postgres
```

```
ENV MODE production
```

Setting or replacing a variable at runtime

```
docker run --env <key>=<value> <image-name>
```

```
docker run --env POSTGRES_USER=test_db --env POSTGRES_PASSWORD=test_db postgres
```

[1] https://hub.docker.com/_/postgres

Secrets in variables are not secure

`docker history <image-name>`

```
ARG DB_PASSWORD=example_password
```

Will show in `docker history` :

```
IMAGE          CREATED        CREATED BY                          SIZE      ...
cd338027297f   2 months ago   ARG DB_PASSWORD=example_password    0B        ...
```

Summary

Usage	Dockerfile Instruction
Create a variable accessible only during the build	ARG <name>=<value>
Create a variable	ENV <name>=<value>

Usage	Shell Command
Override an ARG in docker build	docker build --build-arg <name>=<value>
Override an ENV in docker run	docker run --env <name>=<value> <image-name>
See the instructions used to create an image	docker history <image-name>

Let's practice!

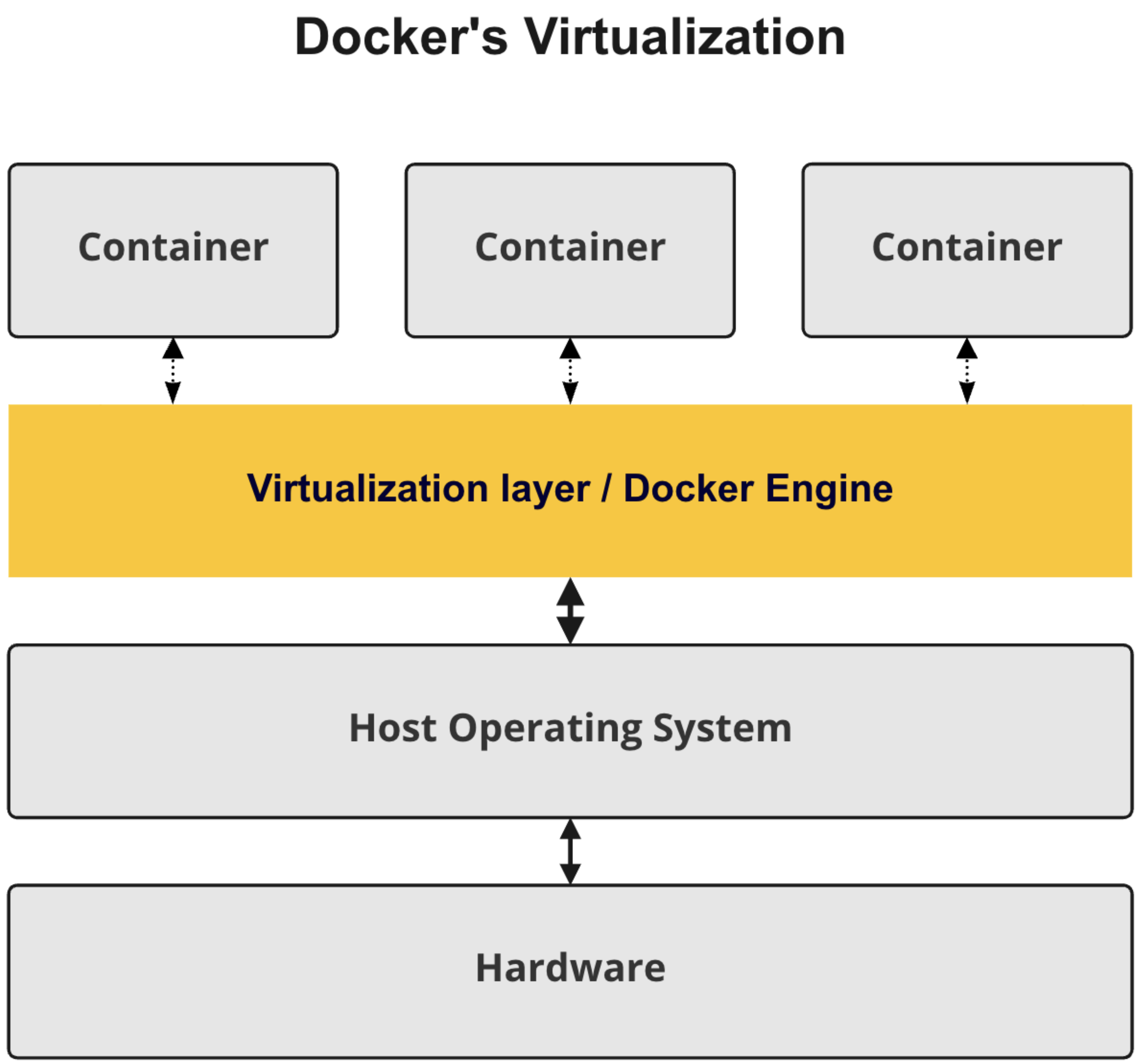
INTRODUCTION TO DOCKER

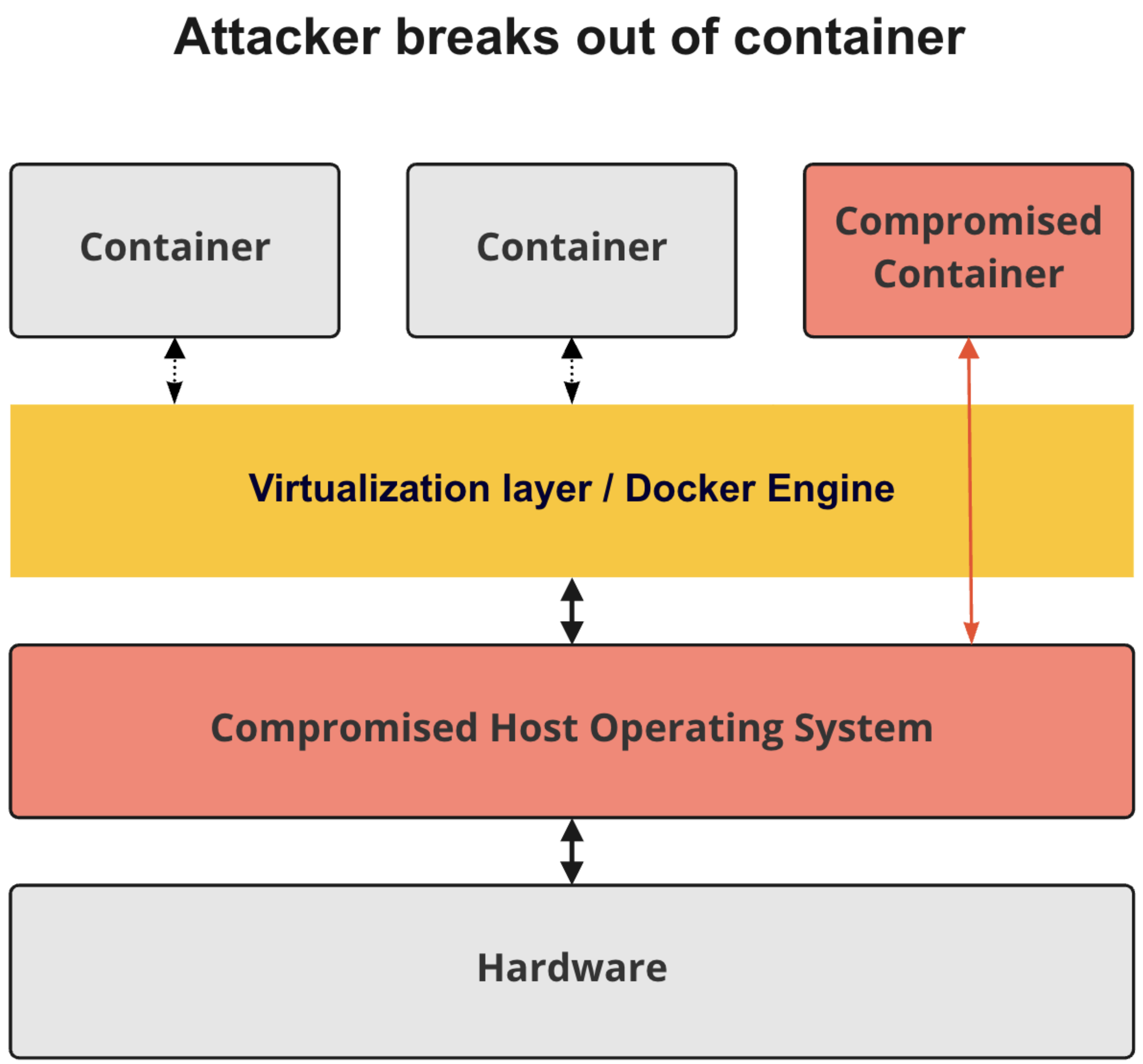
Creating Secure Docker Images

INTRODUCTION TO DOCKER

Tim Sangster
Software Engineer @ DataCamp

Inherent Security




Attacker breaks out of container
Container
Container
Compromised Container
Virtualization layer / Docker Engine
Compromised Host Operating System
Hardware

Making secure images

Attackers can exceptionally break out of a container.

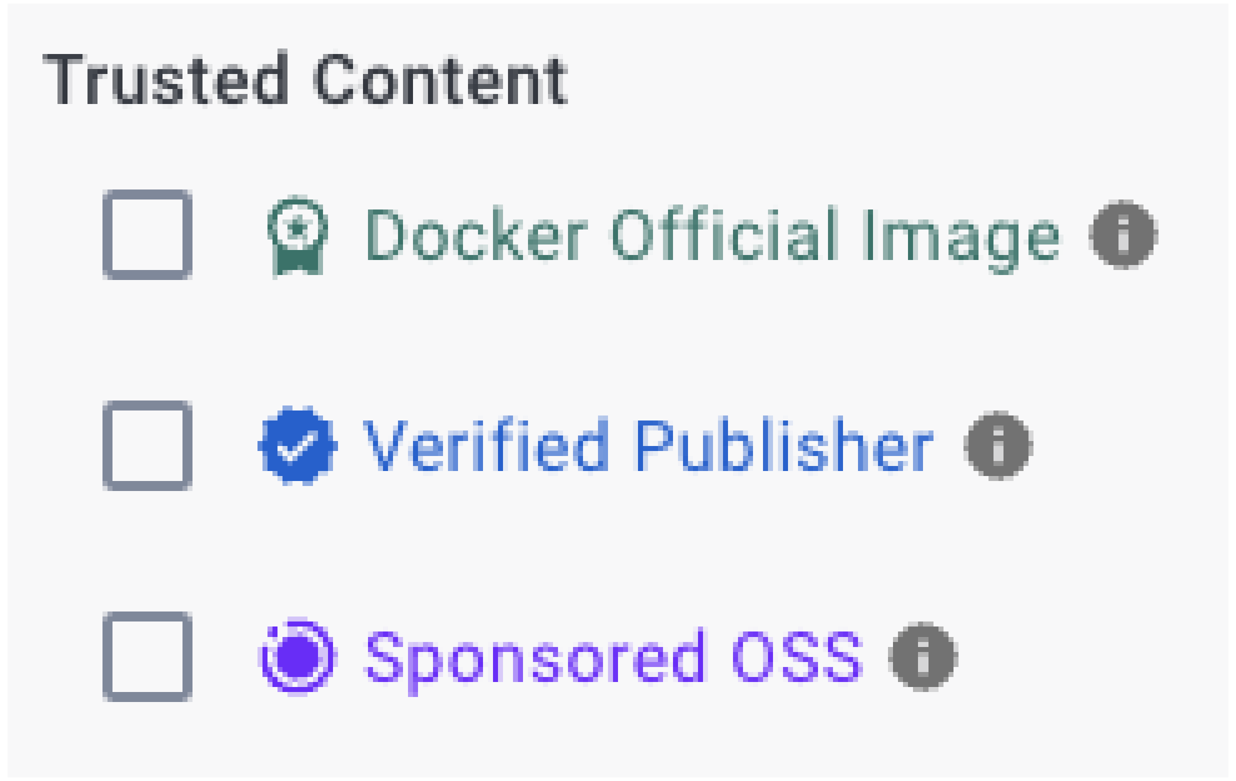
Additional security measures can lower this risk

Becomes especially important once exposing running containers to the Internet.

Images from a trusted source

Creating secure images -> Start with an image from a trusted source

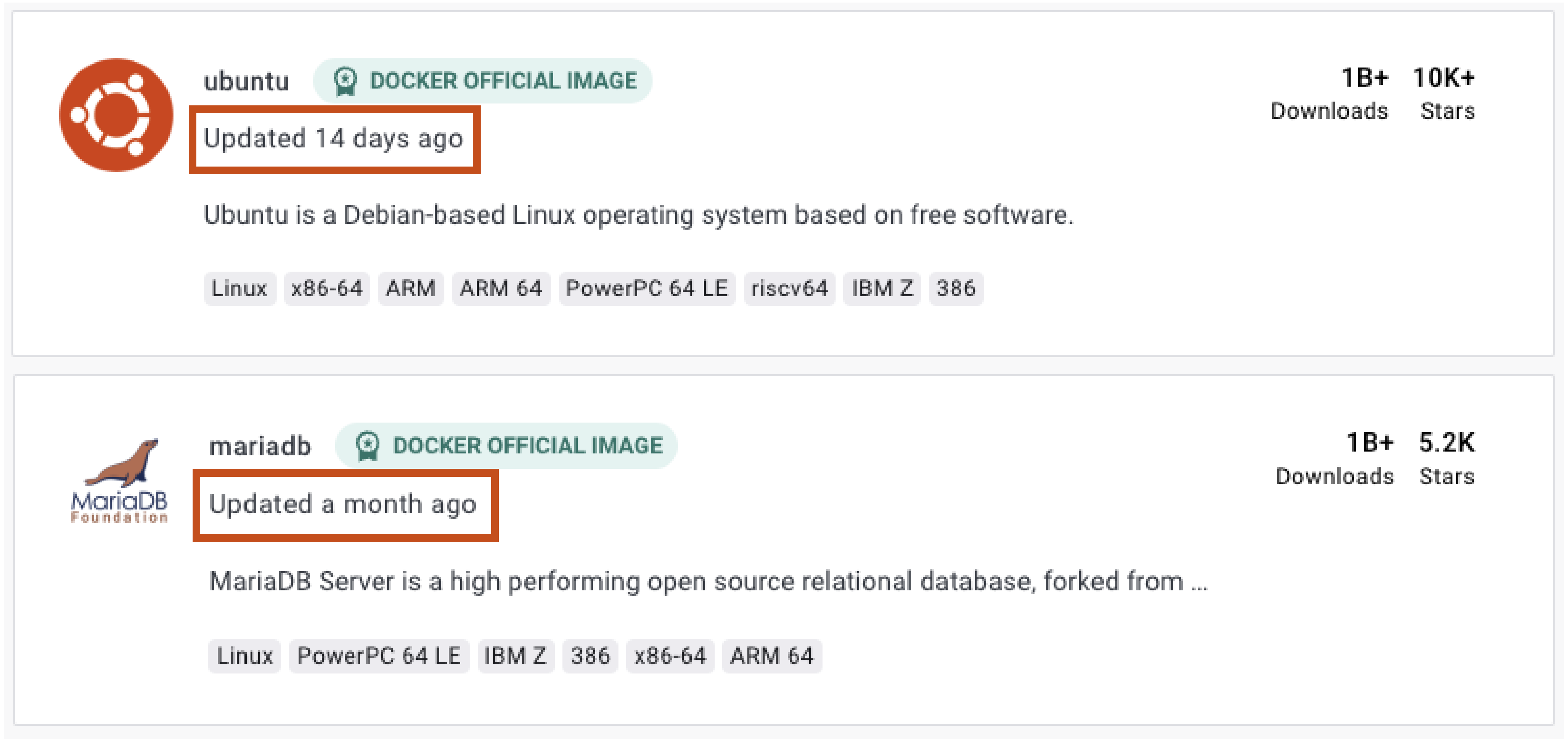
Docker Hub filters:



Trusted Content

- ☐ Docker Official Image
- ☐ Verified Publisher
- ☐ Sponsored OSS

Keep software up-to-date



ubuntu DOCKER OFFICIAL IMAGE

Updated 14 days ago

1B+ Downloads 10K+ Stars

Ubuntu is a Debian-based Linux operating system based on free software.

Linux x86-64 ARM ARM 64 PowerPC 64 LE riscv64 IBM Z 386

mariadb DOCKER OFFICIAL IMAGE

Updated a month ago

1B+ Downloads 5.2K Stars

MariaDB Server is a high performing open source relational database, forked from ...

Linux PowerPC 64 LE IBM Z 386 x86-64 ARM 64

Keep images minimal

Adding unnecessary packages reduces security

Ubuntu with:

- Python2.7
- Python3.11
- Java default-jre
- Java openjdk-11
- Java openjdk-8
- Airflow
- Our pipeline application

Installing only essential packages improves security

Ubuntu with:

- Python3.11
- Our pipeline application

Don't run applications as root

Allowing root access to an image defeats keeping the image up-to-date and minimal.

Instead, make containers start as a user with fewer permissions:

```
FROM ubuntu # User is set to root by default.
RUN apt-get update
RUN apt-get install python3
USER repl # We switch the user after installing what we need for our use-case.
CMD python3 pipeline.py
```

Let's practice!

INTRODUCTION TO DOCKER

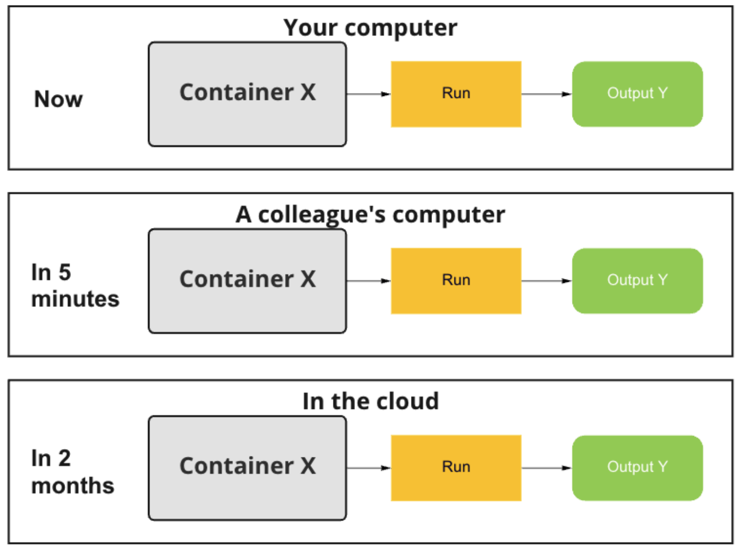
Wrap-up

INTRODUCTION TO DOCKER

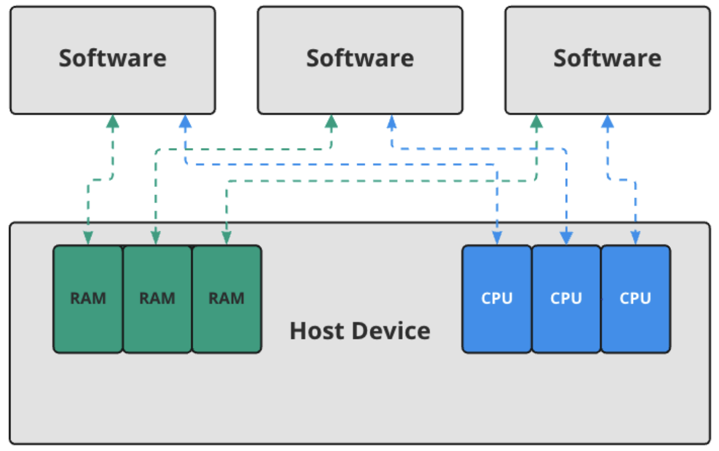
Tim Sangster
Software Engineer @ DataCamp

Chapter 1: The theoretical foundation

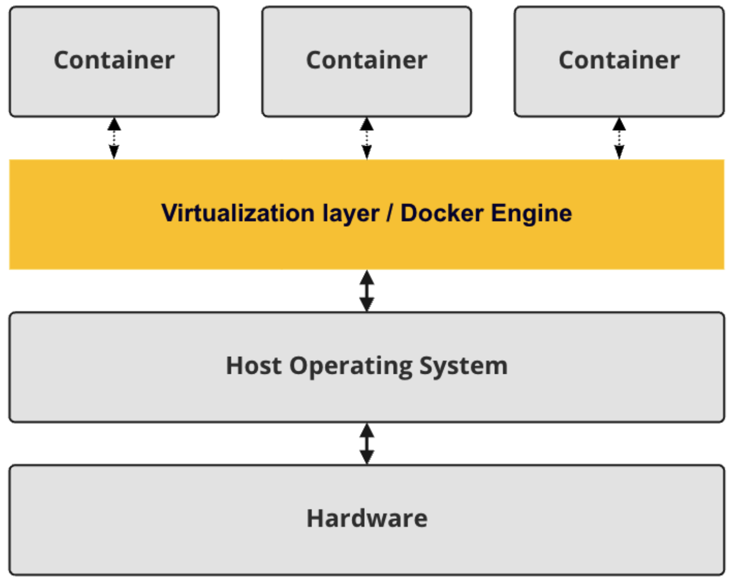
Portability and reproducibility



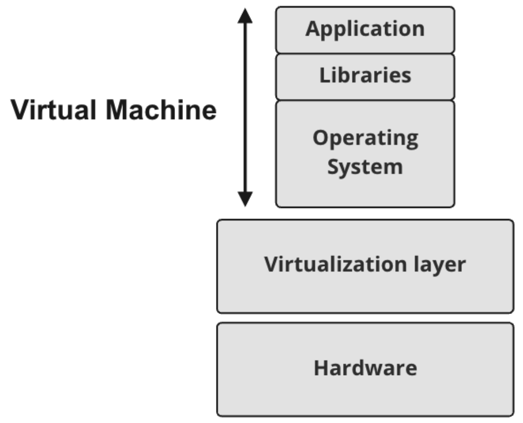
Virtualization



Docker's Virtualization



Virtual Machine Virtualization



Chapter 2: The Docker CLI

Usage	Command
Start a container	docker run (--name <container-name>) (-it) (-d) <image-name>
List running containers	docker ps (-f "name=<container-name>")
Stop a container	docker stop <container-id>
See (live) logs for container	docker logs (-f) <container-id>
Remove stopped container	docker container rm <container-id>
Pull a specific version of an image	docker pull <image-name>:<image-version>
List all local images	docker images
Remove an image	docker image rm <image-name>

Chapter 3: Dockerfiles

```
FROM ubuntu
RUN apt-get update && apt-get install python3
COPY /projects/pipeline /app/
CMD /app/init.py
```

```
docker build -t my_pipeline .
 => [1/3] FROM docker.io/library/ubuntu
 => CACHED [2/3] RUN apt-get update && apt-get install python3
 => CACHED [3/3] COPY /projects/pipeline /app/
```

Chapter 4: Security and Customization

Usage	Dockerfile Instruction
Change the current working directory	WORKDIR <path>
Change the current user	USER <user-name>
Create a variable accessible only during the build	ARG <name>=<value>
Create a variable	ENV <name>=<value>

Usage	Shell Command
Override an ARG in docker build	docker build --build-arg <name>=<value>
Override an ENV in docker run	docker run --env <name>=<value> <image-name>
See the instructions used to create a image	docker history <image-name>

Chapter 4: Security and Customization

- Isolation provided by containers gives security but is not perfect.
- Use the "Trusted Content" images from the official Docker Hub registry
- Keep software on images up-to-date
- Only install the software you need for the current use case.
- Do not leave the user in images set to root.

What more is there to learn?

Dockerfile instructions

- `ENTRYPOINT`
- `HEALTHCHECK`
- `EXPOSE`
- ...

Multi stage builds

```
FROM ubuntu as stage1
RUN generate_data.py
...
FROM postgres as stage2
COPY --from=stage 1 /tmp /data
```

docker Compose

kubernetes

Thank you!

INTRODUCTION TO DOCKER