Use Cases of Container Technology

Peter von Rohr

1/18/2021

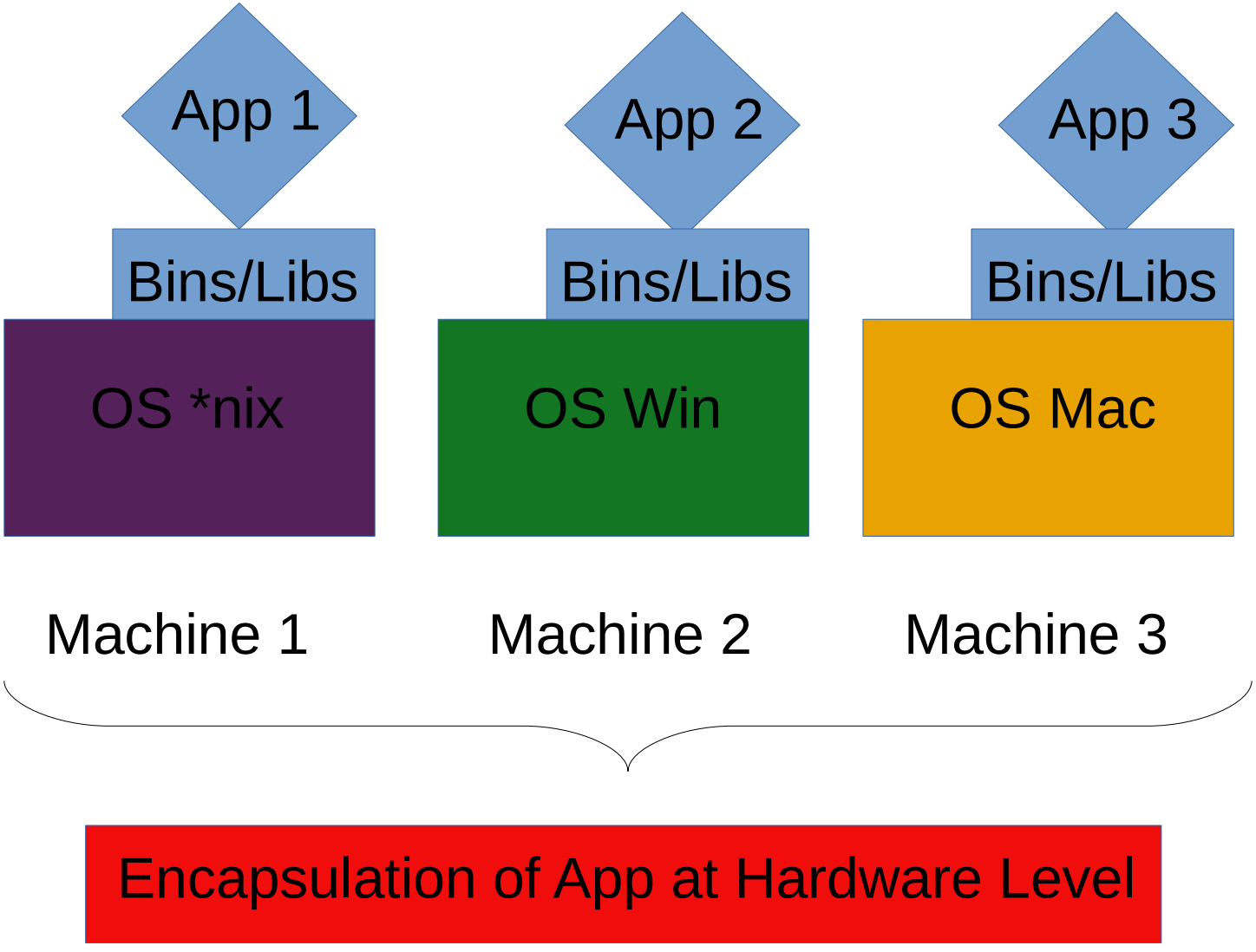
Background

- **Problem**: Multiple Applications (Apps) on the same machine can lead to
  - OS[1] Windows: DLL hell
  - OS *nix: library nightmare
- In general: dependencies between software components are difficult to manage
- Solution: encapsulation of software and libraries into independent units

[1] Operating System

Solution I: Hardware

One server/machine per App



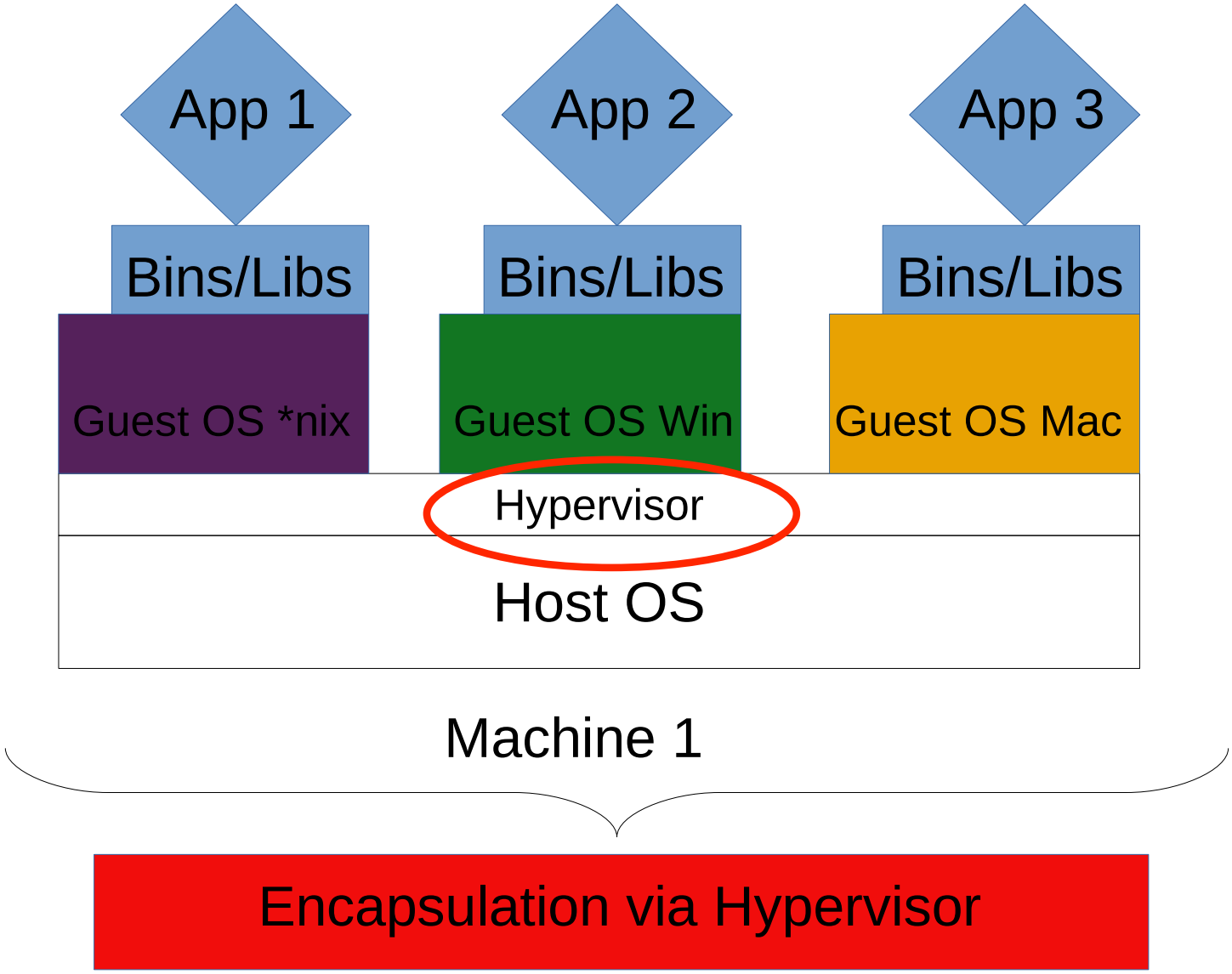
Encapsulation of App at Hardware Level

Problems with Solution I

- efficiency: hardware costs per app
- security: drag along old OS-versions (Win XP, Win 7, old *nix)
- flexibility: testing new apps, updating cycle of existing apps

Solution II: Virtualisation

Hypervisor Software as separation between host and guest

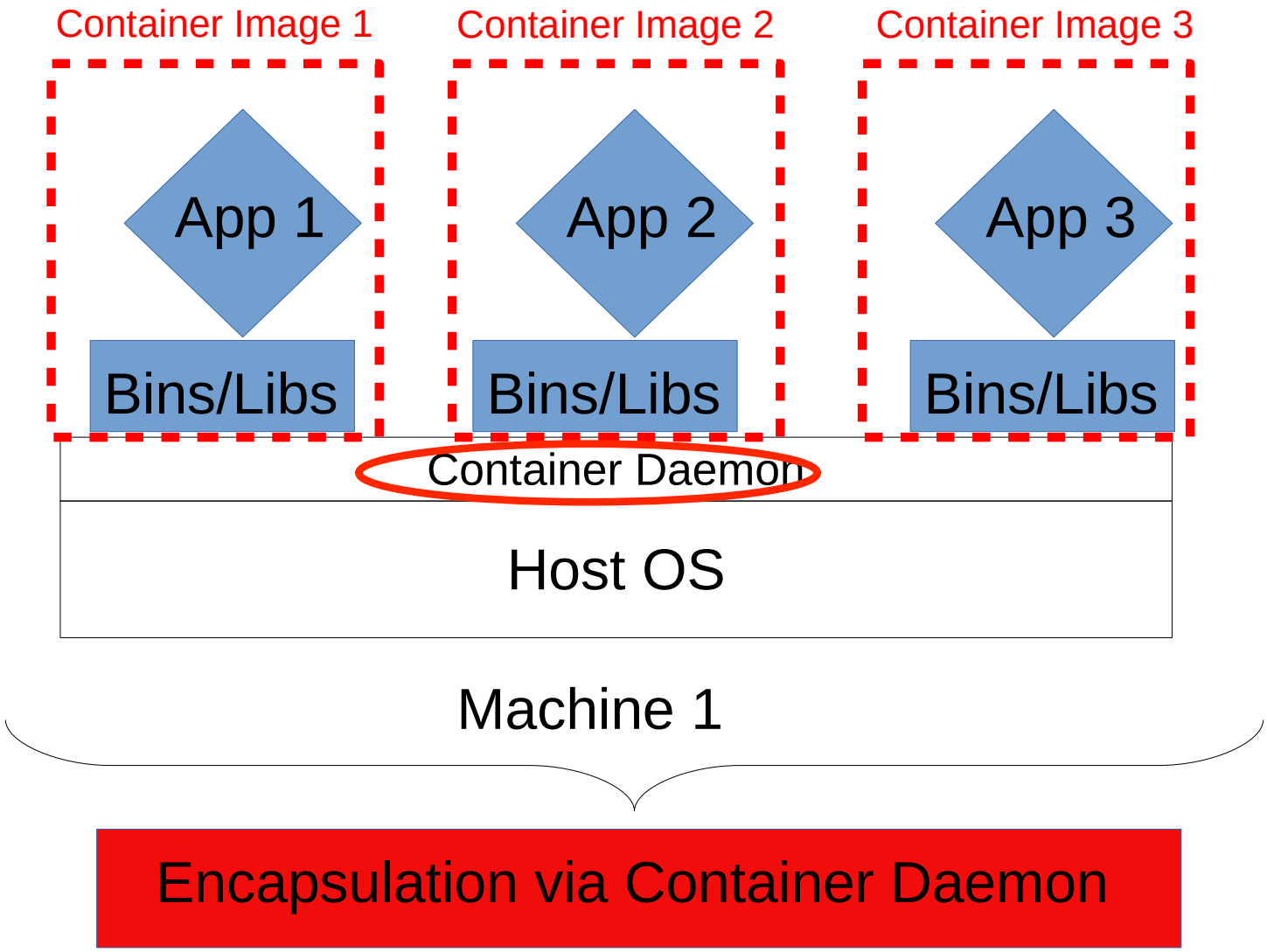


Problems with Solution II

- still just one app per (guest) OS
- intensive in resource requirements
- redundancy in configurations and settings (e.g. network)

Solution III: Containers

Container daemon replaces hypervisor



Encapsulation via Container Daemon

Container Facts

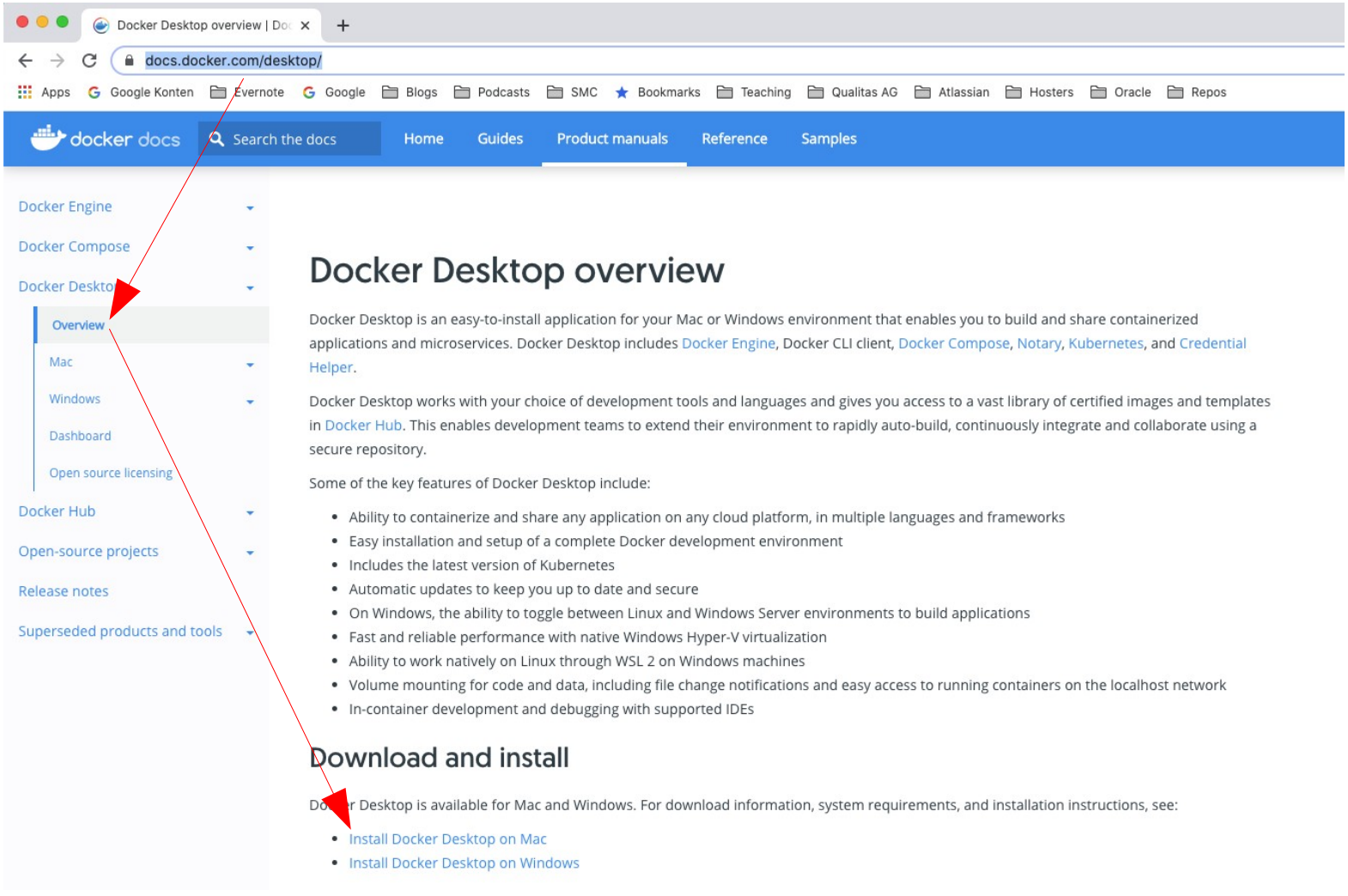
- Full benefit only with linux OS
- With OS Win and OS Mac: closer to virtualisation
- Still
  - develop it once, run it somewhere else
  - great for testing and updating systems

Container Products

- Docker
  - best known
  - industry standard
  - deployment in the cloud
  - requires root
  - images in layers
  - portability via dockerhub
- Singularity
  - preferred in HPC environment
  - can be run as non-root
  - images in files
  - portability via hubs (singularity and docker)
  - can build images based on dockerfiles

Demo I: Getting Started with Docker

- On Mac/Win: Download docker desktop (https://docs.docker.com/desktop/)



Docker Desktop Check

- Check whether docker desktop was installed

```
# show the version
docker version
```

- Run a first container image

```
docker run hello-world
```

Useful Docker Commands

```
# list of containers
docker ps -a
# remove container
docker rm <container_id>
# list of images
docker images
# remove images
docker rmi <image_name>
# download image from dockerhub
docker pull <image_name>
```

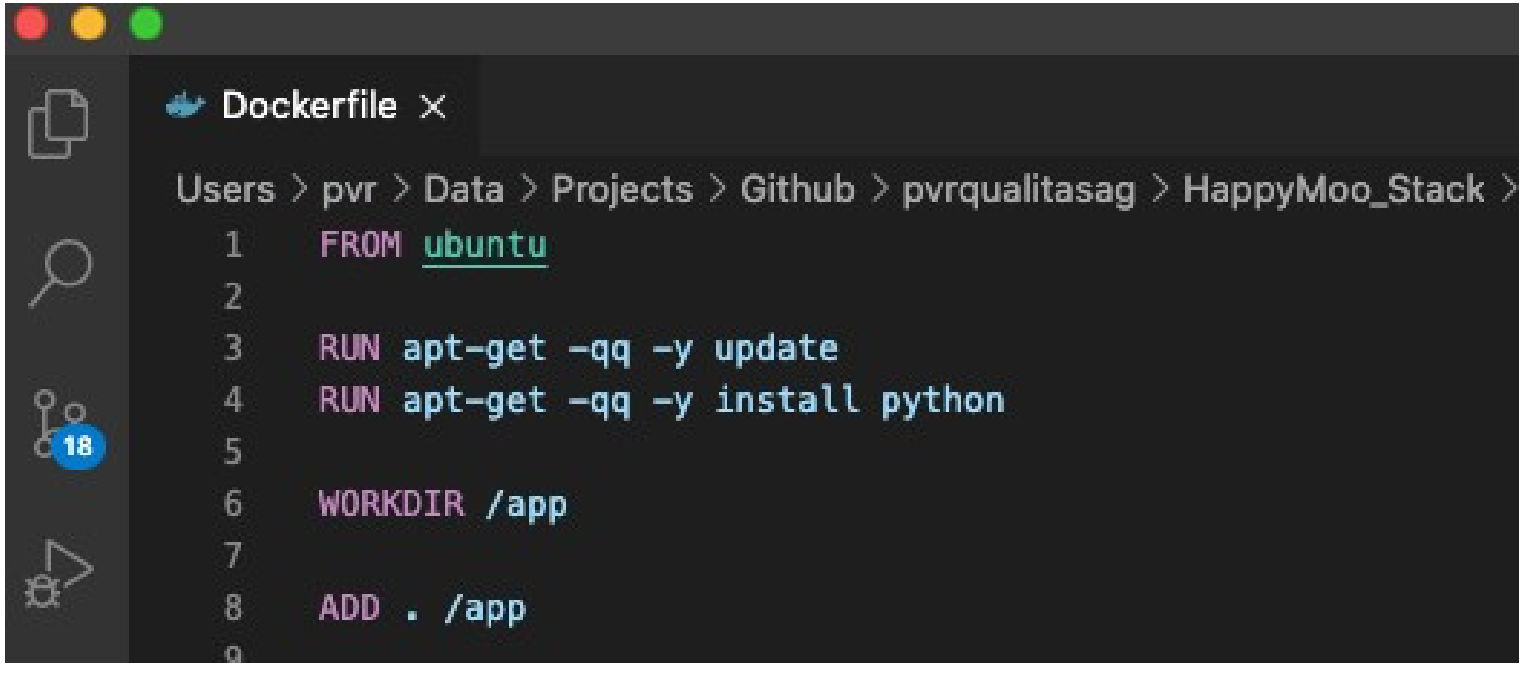
Dockerhub

- registry of public images: https://hub.docker.com/
- wide collection of docker images (e.g. ubuntu)
- try

```
docker pull ubuntu
```

Dockerfile

- Create own images via Dockerfile
- Example: run a python script



Build and Run

```
# build image from Dockerfile
docker build -t my_do_py .
...
# run
docker run -it my_do_py python /app/my_script.py
```

- Add the run command to the Dockerfile by

```
CMD ["python", "/app/my_script.py"]
```

Deployment

- Other server: 2-htz.quagzws.com
- Use the same Dockerfile for building
- Run with the same commands

Real World

- Rstudio form inside docker
- Dockerhub

```
docker run -d --rm -p 10087:8787 \
  -e PASSWORD=yourpasswordhere rocker/rstudio
```

- Map home directory with option `-v`
  `<host_dir>:<container_dir>`

Singularity

- Reference: https://sylabs.io/
- Organises images in files
- Can be run as ordinary user
- Work with sandbox

Singularity Sandbox

```
sudo singularity build --sandbox ubuntu_s docker://ubuntu
```

- Write singularity file
- Run python program

```
singularity exec ubuntu_s/ python my_script.py
```

Update Sandbox

```
sudo singularity shell --writable ubuntu_s
apt-get update -y
apt-get install -y python
```

Singularity File

```
Bootstrap: docker
From: ubuntu

%post
    apt-get -y update
    apt-get install -y python

%files
    my_script.py /

%runscript
    python /my_script.py
```

- Build and run with

```
sudo singularity build ubuntu.sif Singularity.recipe
./ubuntu.sif
```

Resources

- Container Explainer: https://www.youtube.com/watch?v=FWpnbGnzk08
- Virtual Machines vs Docker Containers - Dive Into Docker: https://www.youtube.com/watch?v=TvnZTi_gaNc
- Docker for beginners: https://docker-curriculum.com/
- Singularity: https://sylabs.io/
- Singularity Example Workflow: https://www.youtube.com/watch?v=m8llDjFuXIc&t=15s