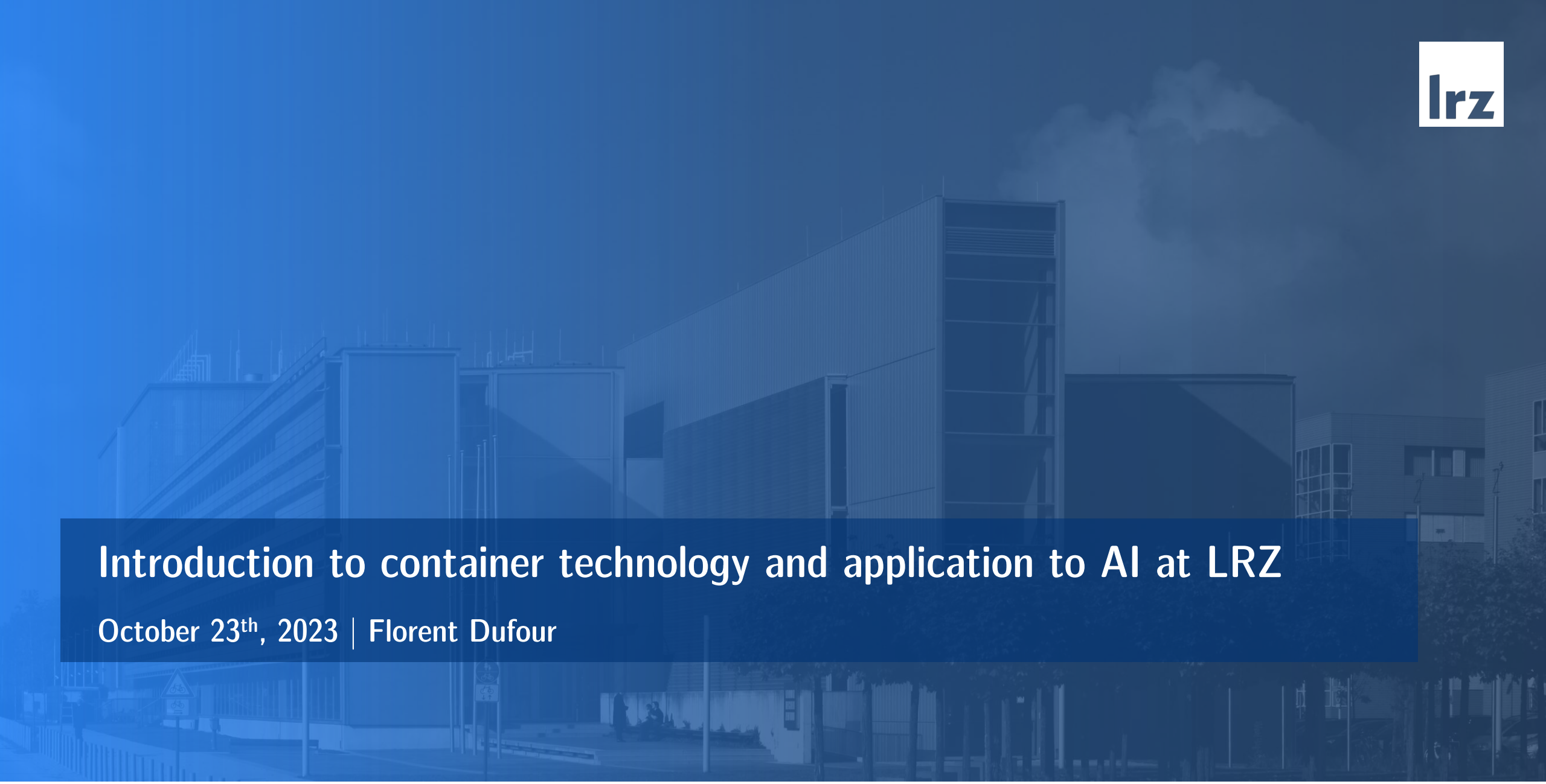
Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities



Introduction to container technology and application to AI at LRZ

October 23th, 2023 | Florent Dufour

Agenda

Roadmap for today

Part 1

A Tour of Containers

- Basic concepts
- Definitions
- Containers vs. the world

- Hands-on #1: Kicktart: Your first steps with containers

Part 2

Under the hood

- Docker, volumes, ports, variables, and Dockerfile
- Containers for High Performance Computing

- Hands-on #2: Deep Learning: Make an ANN dream in a container
- Hands-on #3 HPC AI: Speech to text with OpenAI whisper

Part 3

Containers on Nitromethane

- Abstraction
- Hardware acceleration,
- Scaling across a compute cluster

- Hands-on #4: Reproducible scientific workflow with containers: RNA-seq pipeline

What you will need

The course resources: https://doku.lrz.de/x/eQBvB

An individual Virtual Machine (sandbox), look in your emails:

- Your IP address: `138.***.***.***`
- Username: `student-*`
- Password: `********`

> You can start downloading the Handout and Presentation.

> > You can start SSH into the VM. Make yourself at home!

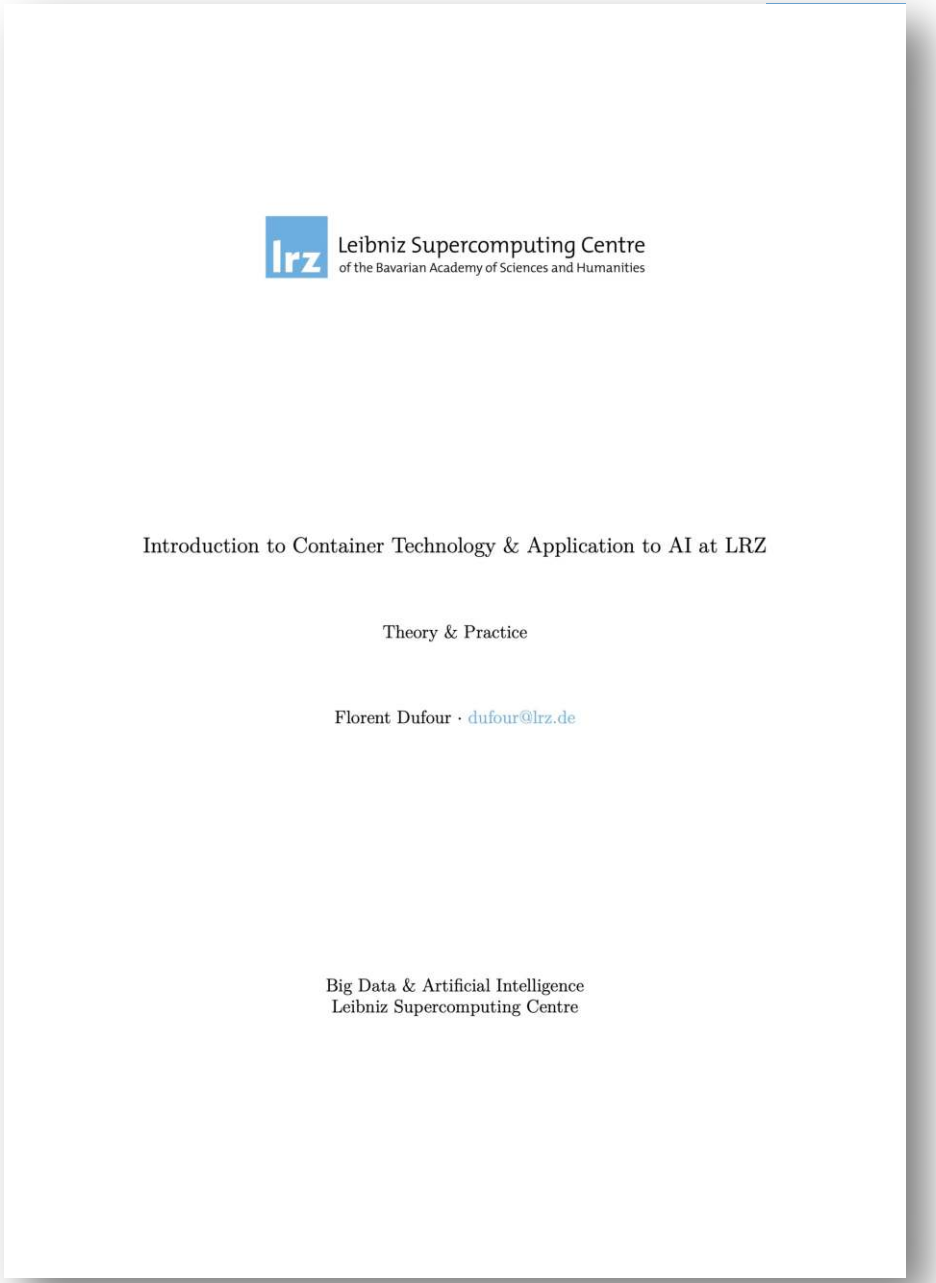
Agenda

What's going to happen?

- **Few lecture slides covering what's in the handout**

- **Jump into the hands-on, use your workbench**
  - Information are given in the code snippets of the hand-out
  - You experiment and we get the answers together right after
  - If it's too easy for you, look at the Bonus Questions
  - All hands-on are independent

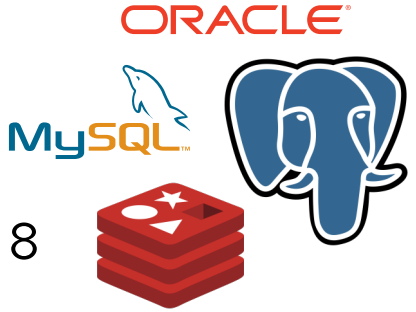
- **Use reactions, chat, and give feedback all along!**



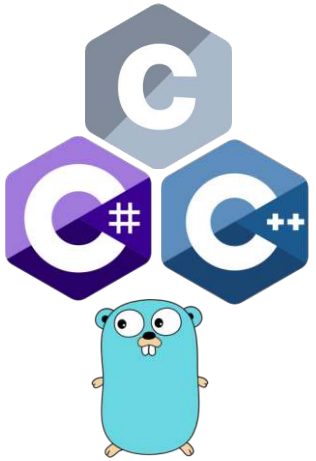
A Tour of Containers

Kickstarter

12, 14, 15



5, 8



7, 8, 10, 14

.gradle

.r

.cache

.conda

8, 11, 16

2.3, 2.4, 2.5

.m2

.jupyter

.node_modules

13, 12

.npm

.config

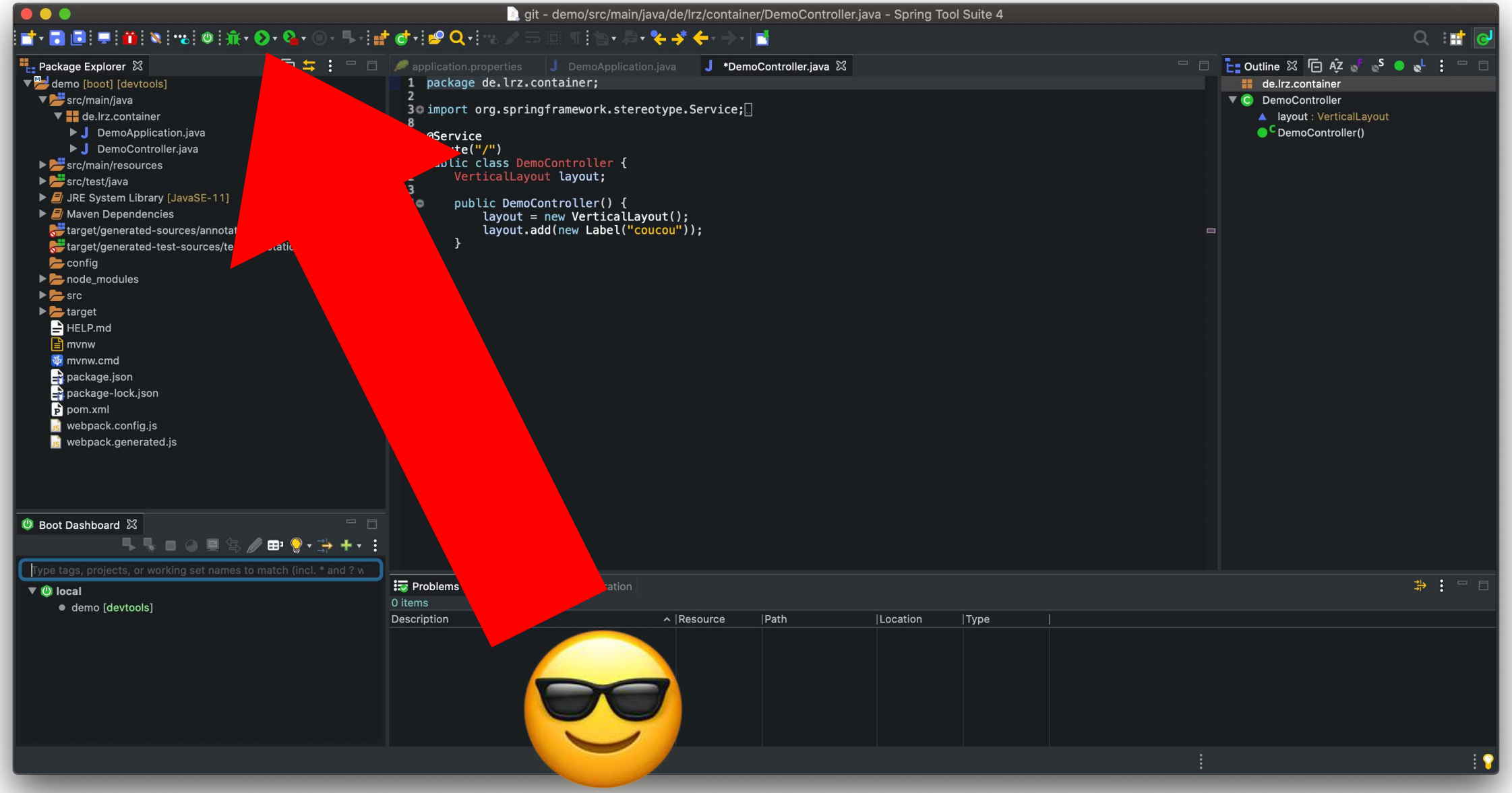
/etc

5, 7

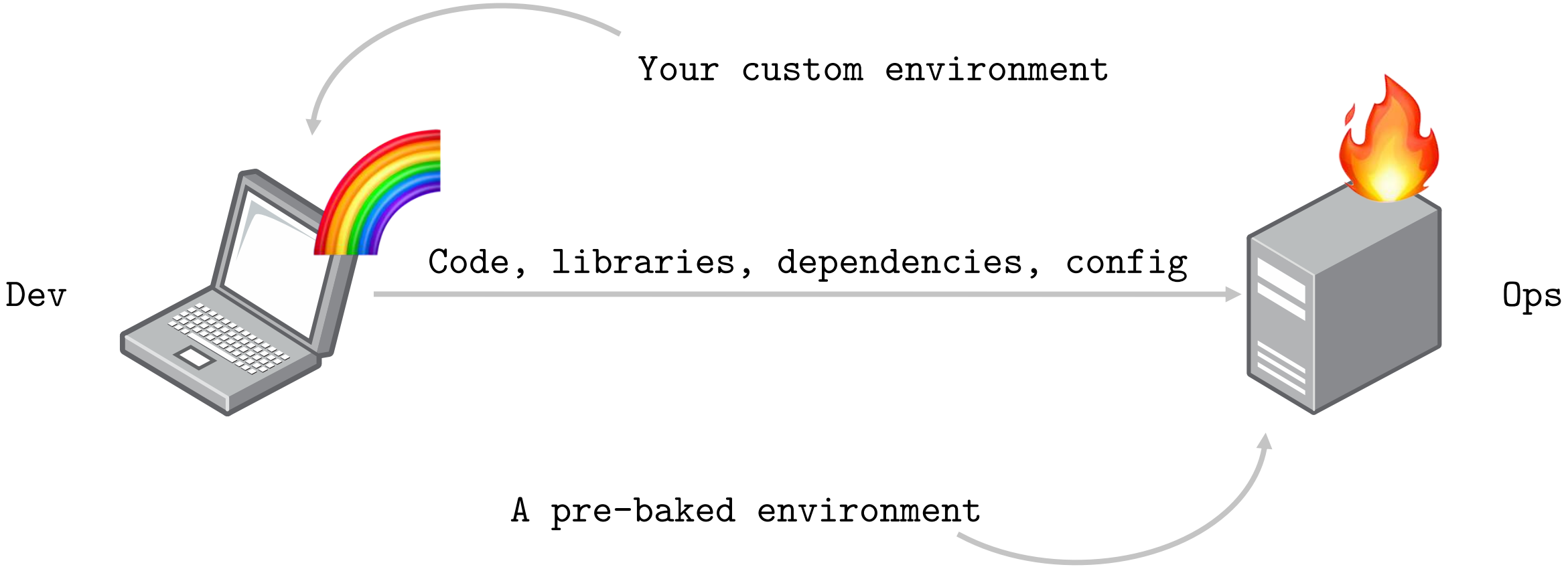
2, 3, 4, 5

2.7, 3.6

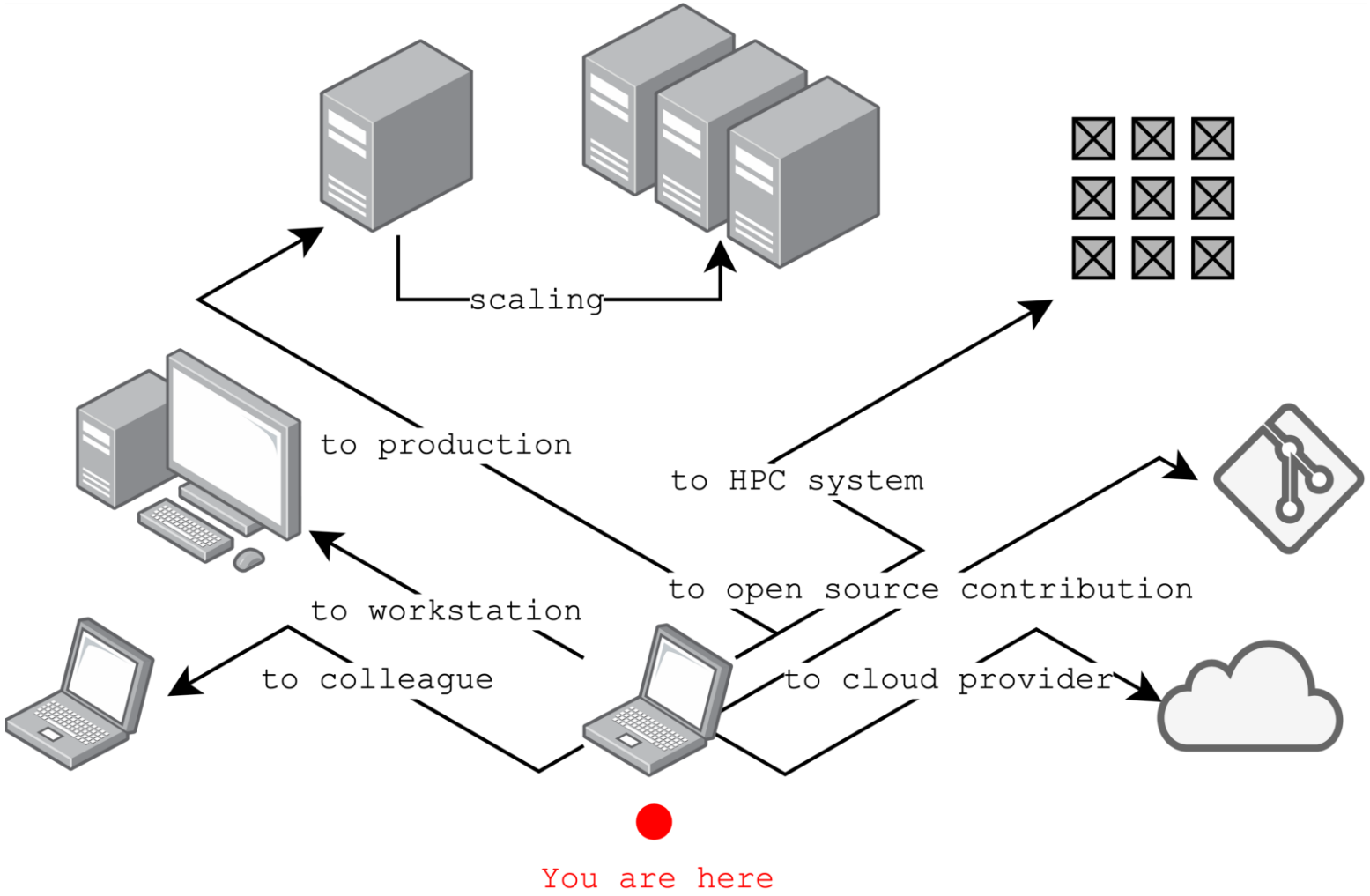
PyTorch


git - demo/src/main/java/de/lrz/container/DemoController.java - Spring Tool Suite 4
Package Explorer
demo [boot] [devtools]
src/main/java
de.lrz.container
DemoApplication.java
DemoController.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
config
node_modules
src
target
HELP.md
mvnw
mvnw.cmd
package.json
package-lock.json
pom.xml
webpack.config.js
webpack.generated.js
application.properties
DemoApplication.java
*DemoController.java
1 package de.lrz.container;
2
3 import org.springframework.stereotype.Service;
8
@Service
te("/")
blic class DemoController {
VerticalLayout layout;
3
public DemoController() {
layout = new VerticalLayout();
layout.add(new Label("coucou"));
}
Outline
de.lrz.container
DemoController
layout : VerticalLayout
DemoController()
Boot Dashboard
Type tags, projects, or working set names to match (incl. * and ? w
local
demo [devtools]
Problems
0 items
Description
Resource
Path
Location
Type

Shpping software is dangerous...

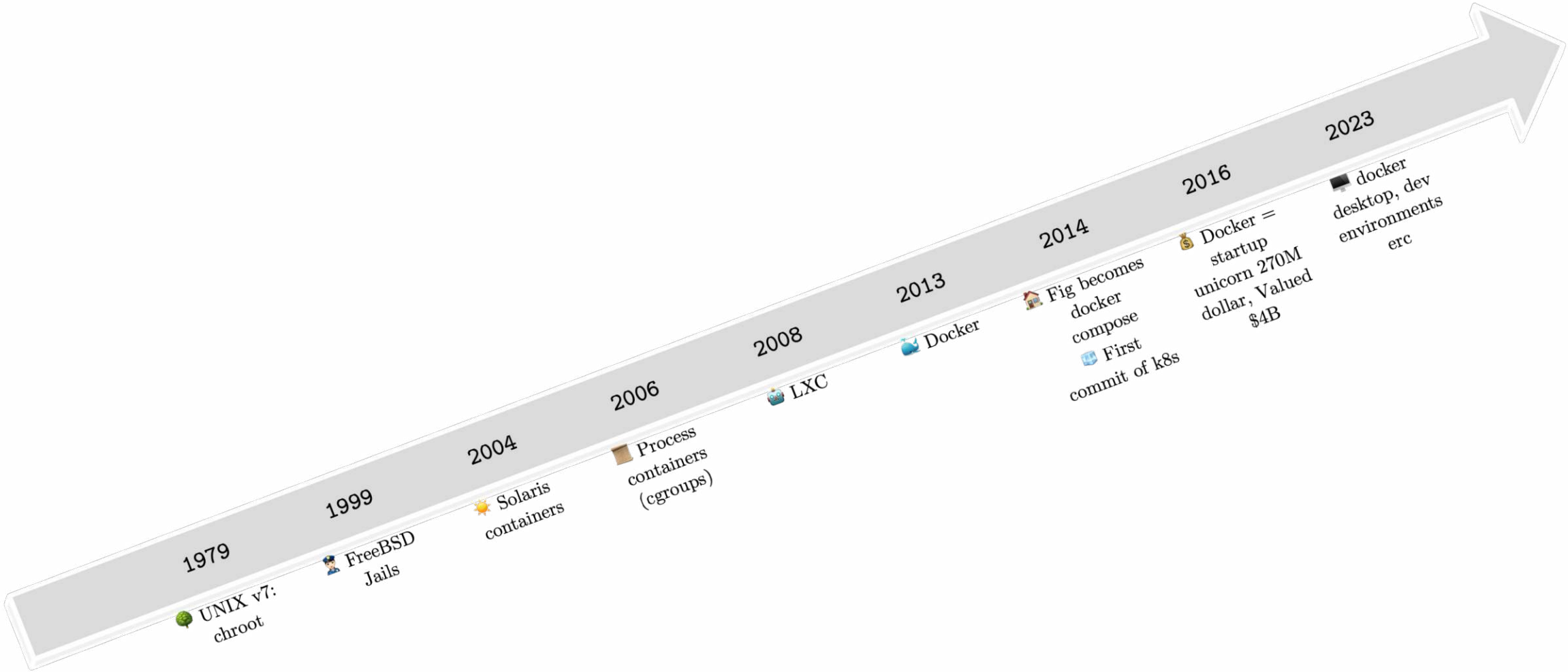


… and yet it happens all the time



The future of Linux containers (15th March 2023)

Timeline of Event



Landscape of solutions at your disposal when shipping code

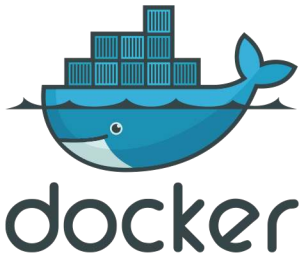
Sandboxes / packages

`.jar / .war`

`virtualenv`

`Big binary`

Containers



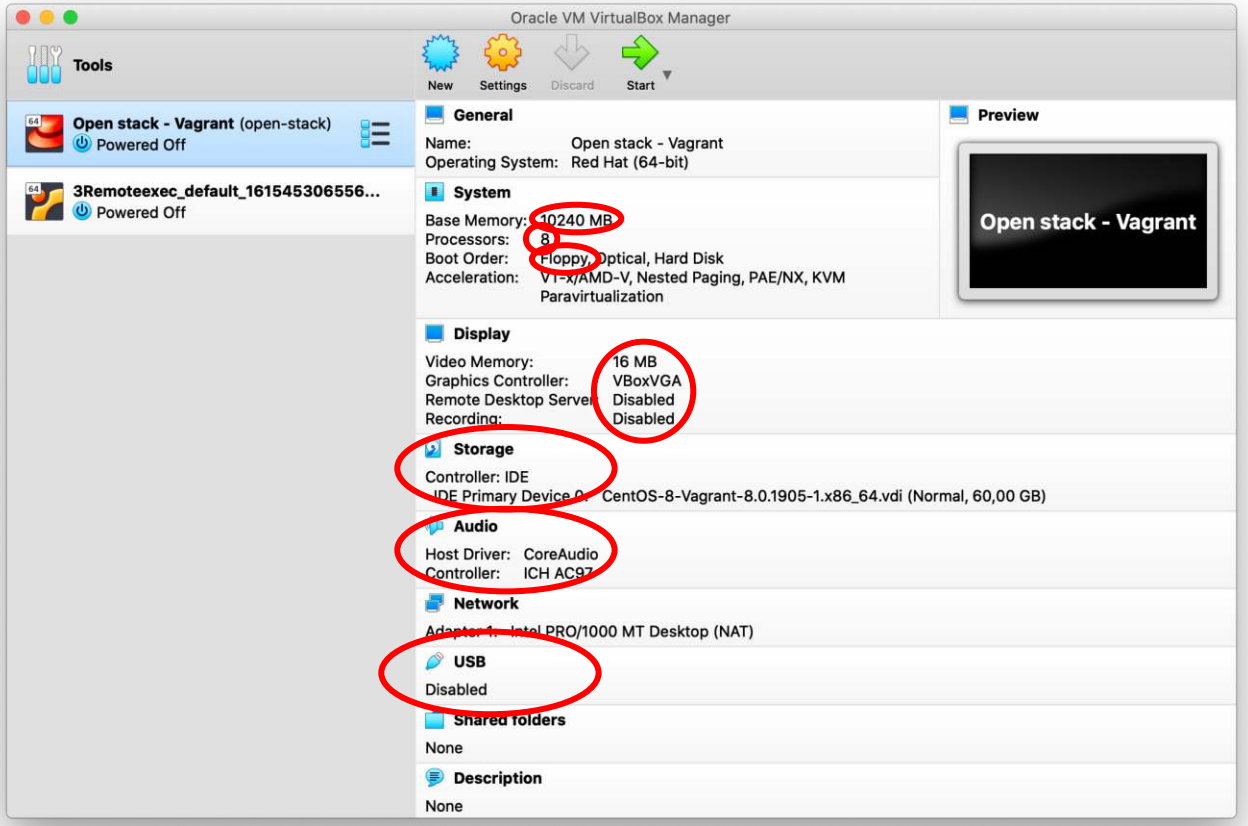
Virtual machines

Containers vs. VMs

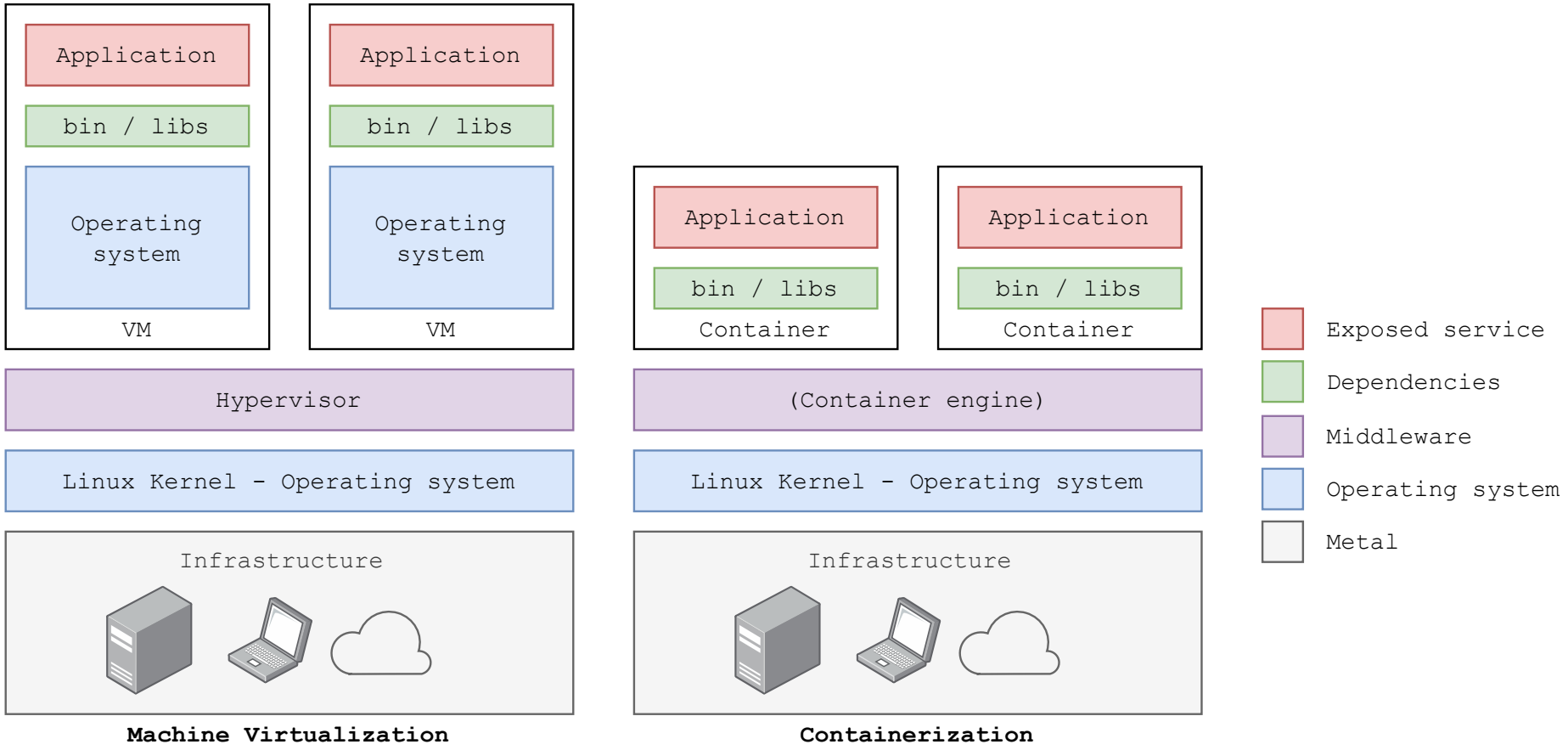
VMs Emulate Hardware

- RAM
- Processor
- Floppy drive?
- Graphics card
- Storage
- Audio card
- Networking
- Bunch of interfaces

😱 WAIT! we don't want to worry about hardware!



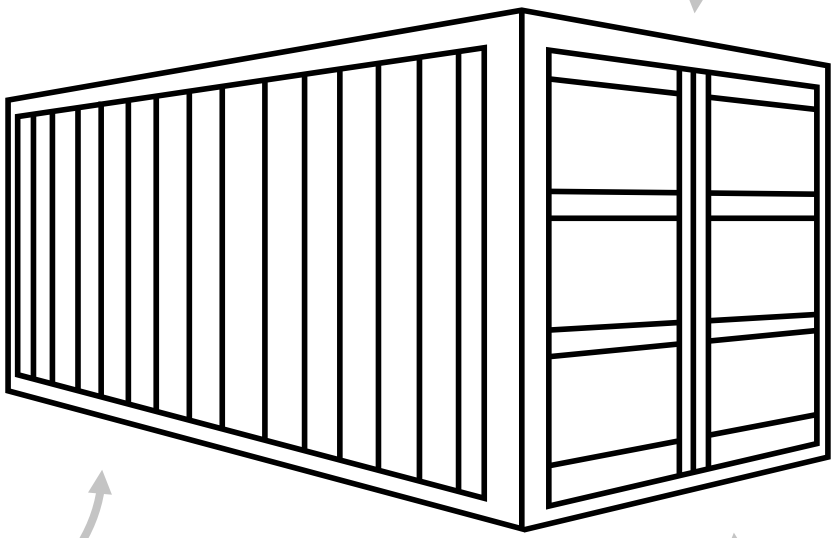
Containers vs. VMs



Your code

Runtime

Libraries

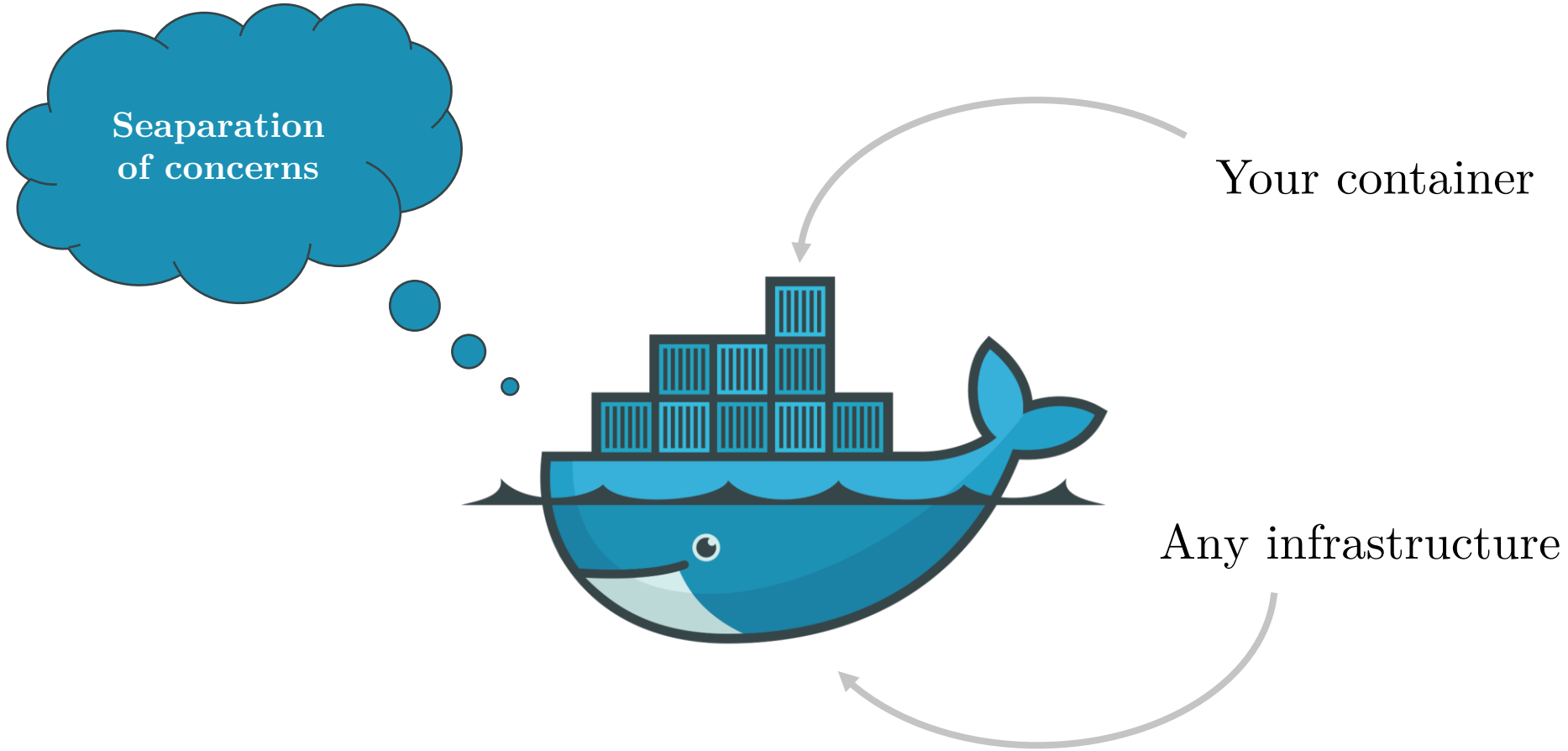


Config files

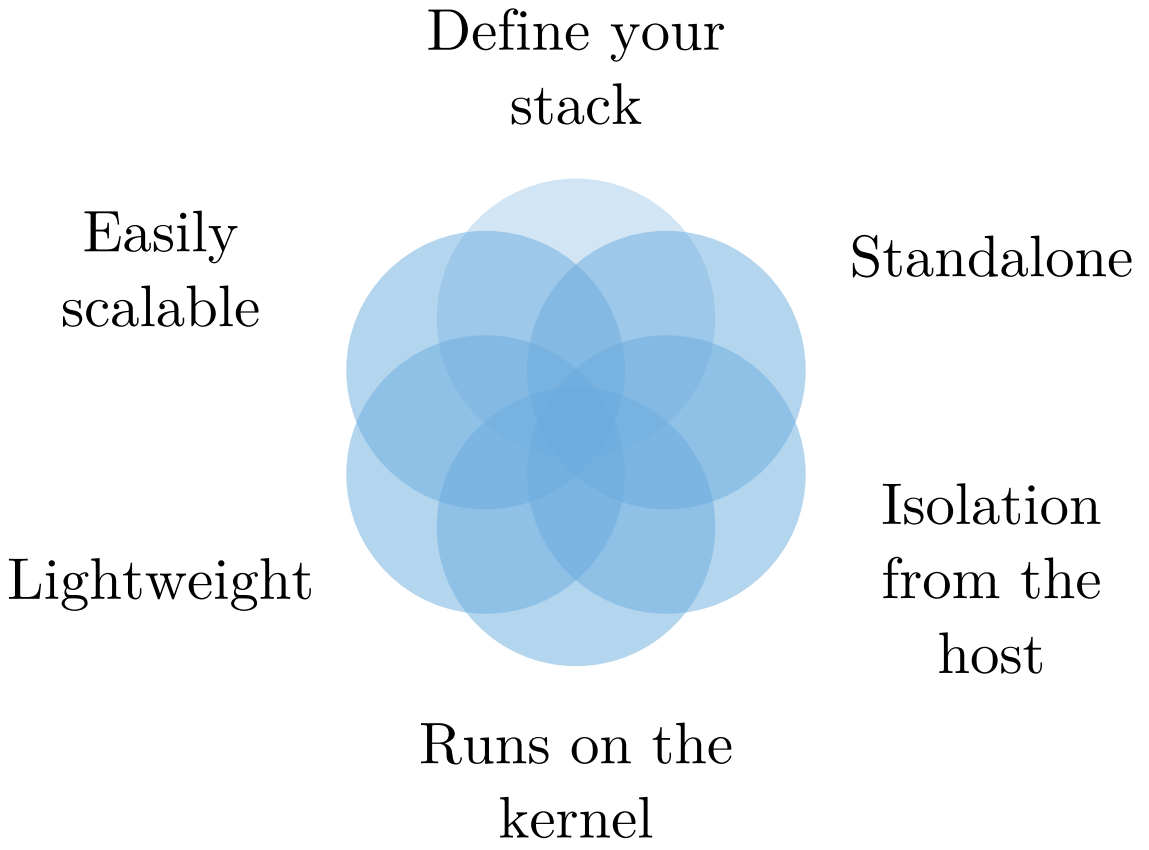
Assets

Containers allow: i) to sandbox the entire system ii) Without machine details iii) and without performance hit

What containers are all about



Containers allow UDSS



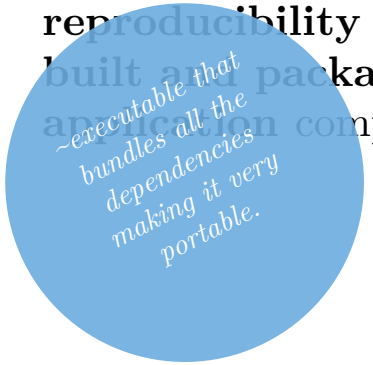
User Defined Software Stack

What are containers?

Image

The way container software is shipped.

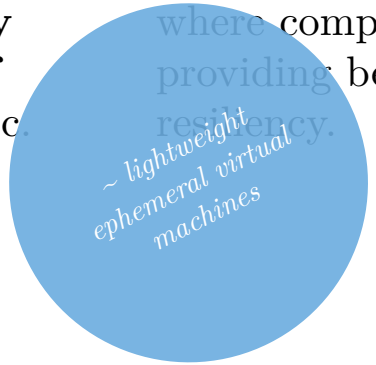
- Static, standardized, and portable **filesystem snapshot** with a **predefined executable** command or entry point.
- **Images are built** and can be stored and distributed.
- Images are a tool for **reproducibility** used to ship **fully built and packaged versions of** application components, assets etc.



Container

Isolated environment for running processes.

- **Running instance of an image**.
- Runs on a host machine and **shares the kernel** with OS.
- **The processes running in the container are isolated** from the rest of the system.
- Containers are **a tool for isolation** where components are segregated, providing better security and resiliency.



Orchestrator

Makes containers manageable.

- **Automates the lifecycle** of containers, networking, storage…
- Organizes containers into **abstract services** and handles dependencies. It allows to **declaratively describe how containers should behave.**
- An orchestrator is a tool for **managing complex applications** by providing load balancing, monitoring, automated restarts, version migration, and many other convenience capabilities.

Hands-on #1: Docker kickstarter

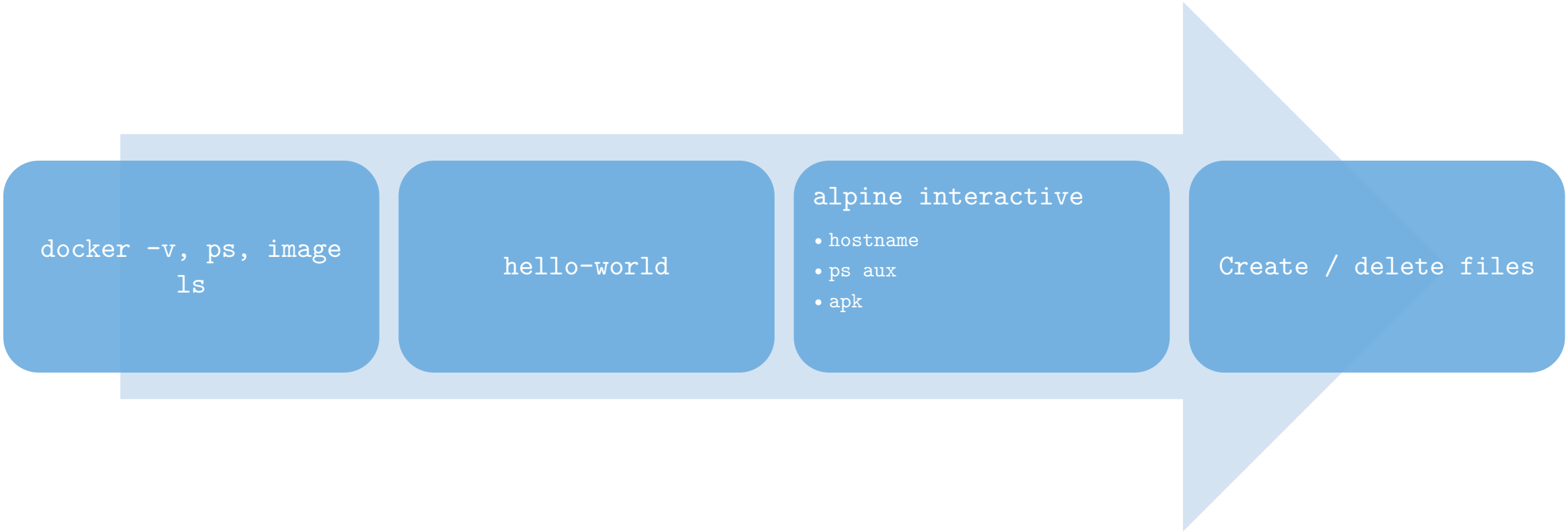
Hands-on #1

Your first steps with containers

```
# ---------------------- #
# On your local machine #
# ---------------------- #

whoami
# florent

uname -a
# Darwin BADWLRZ-AB12345 20.6.0 Darwin Kernel Version 20.6.0: Mon Aug 30 06:12:21
    PDT 2021; root:xnu-7195.141.6~3/RELEASE_X86_64 x86_64


ssh <user>@<IP>
# Replace <user> and <IP> with yours.

# The authenticity of host <IP> can't be established.
# ECDSA key fingerprint is SHA256:BZgws5BArCfwE6rDuN5i/aLgZMAKuC4si2D+ZuLN5gw.
# Are you sure you want to continue connecting (yes/no)? yes
# Warning: Permanently added <IP> (ECDSA) to the list of known hosts.
# <user>@<IP> password:

# Type your password when prompted, it's normal if you don't see what you type!

# ------------------------------ #
# On the compute cloud workbench #
# ------------------------------ #

# Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)

whoami
# <user>

uname -a
# Linux containers-workbench 4.19.0-14-cloud-amd64 #1 SMP Debian 4.19.171-2
    (2021-01-30) x86_64 GNU/Linux

# Success! We can proceed from here!
```

Hands-on #1

Your first steps with containers



Page 6 in handout

Under the hood of containers

The bolts and nuts of containers

`namespaces`

- Provide an isolated view of the resources on the systems to processes running in a container
- Impossible for them to escape and see what's happening elsewhere
- Whether it is the host or other containers running next to them.

`cgroups`

- Control the hierarchical resource management and constraints
- Enforcing resources quotas (*e.g.*, CPU, RAM, I/O, bandwidth usage...)

Built into the kernel

Appendix A for more details

The bolts and nuts of containers

> Containers are not a virtualization technology

> > Linux uses namespaces and cgroups. The system is a big container.

> > > Even when you're not in a container, you are in a container.

There is no performance hit.
I repeat, there is no performance hit.

The bolts and nuts of containers (performance)

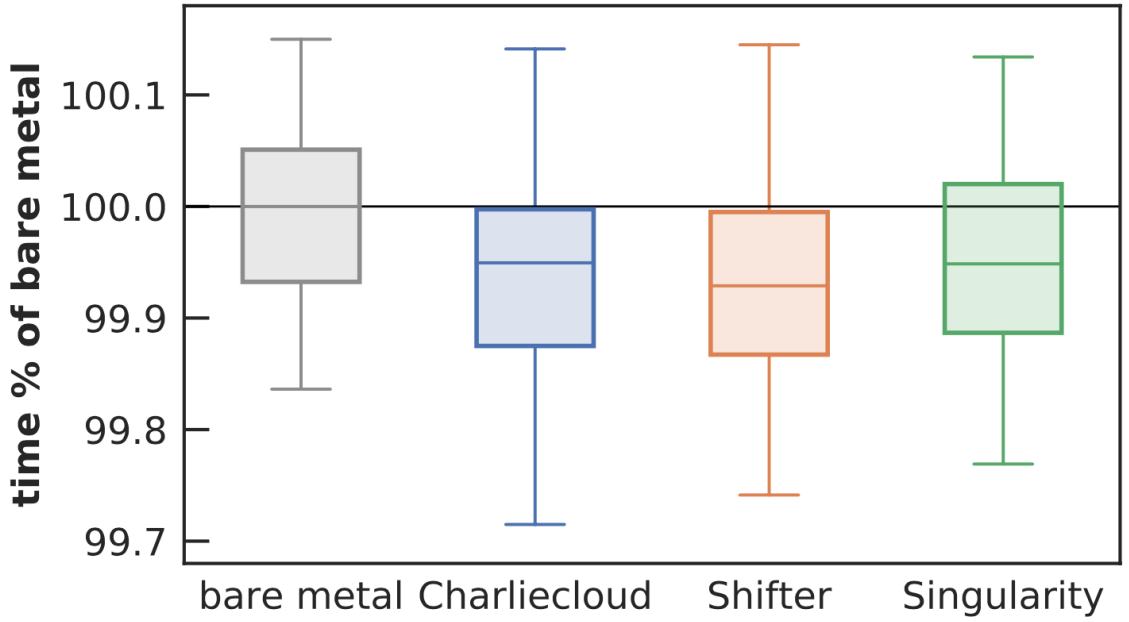


Fig. 1. SysBench prime number computation time relative to median bare metal performance of 129.36 seconds; lower is better. Boxes show the median and middle 50%, while whiskers show the maximum and minimum. The four environments showed essentially identical performance.

A. Torrez, T. Randles, and R. Priedhorsky, "HPC Container Runtimes have Minimal or No Performance Impact," in 2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC), Nov. 2019, pp. 37–42. doi: 10.1109/CANOPIE-HPC49598.2019.00010.

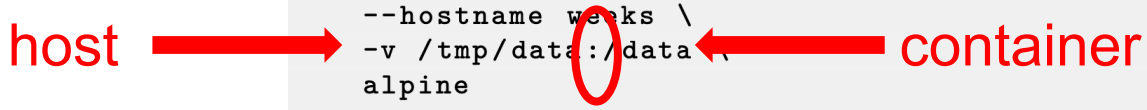
> Not to metion the time spared for humans (dev and ops)

The bolts and nuts of containers

- Containers are ephemeral
- No data is persisted in a container
- Containers are meant to be stateless
  - No state
  - No information related to the state: amnesic
  - Just get the job done
  - Separation of data and process
- Containers can be replicated
  - No state = each container is the same
  - Allow for horizontal scaling

Volume mapping

```
# ---------- #
# ON THE HOST #
# ---------- #

# Let's create a sample directory
# Its content will be mapped to the container
mkdir /tmp/data
# Let's create some files
echo "Monday, Tuesday, Wednesday"    > /tmp/data/week.txt
echo "Jeudi, Vendredi, Samedi"       > /tmp/data/semaine.txt
echo "Sonntag, Sonntag, Sonntag"     > /tmp/data/woche.txt

# -v let us map volumes with the syntax: <path_on_host>:<path_in_container>
docker run \
    -it \
    --rm \
    --name weeks \
    --hostname weeks \
    -v /tmp/data:/data \
    alpine

# --------------- #
# IN THE CONTAINER #
# --------------- #

ls -lah /data/
# total 20K
# drwxr-xr-x    2 root     root        4.0K Mar 26 10:07 .
# drwxr-xr-x    1 root     root        4.0K Mar 26 10:08 ..
# -rw-r--r--    1 root     root          24 Mar 26 10:07 semaine.txt
# -rw-r--r--    1 root     root          27 Mar 26 10:07 week.txt
# -rw-r--r--    1 root     root          26 Mar 26 10:07 woche.txt

# You can escape the container without killing it
# With ctrl-P ctrl-Q
```



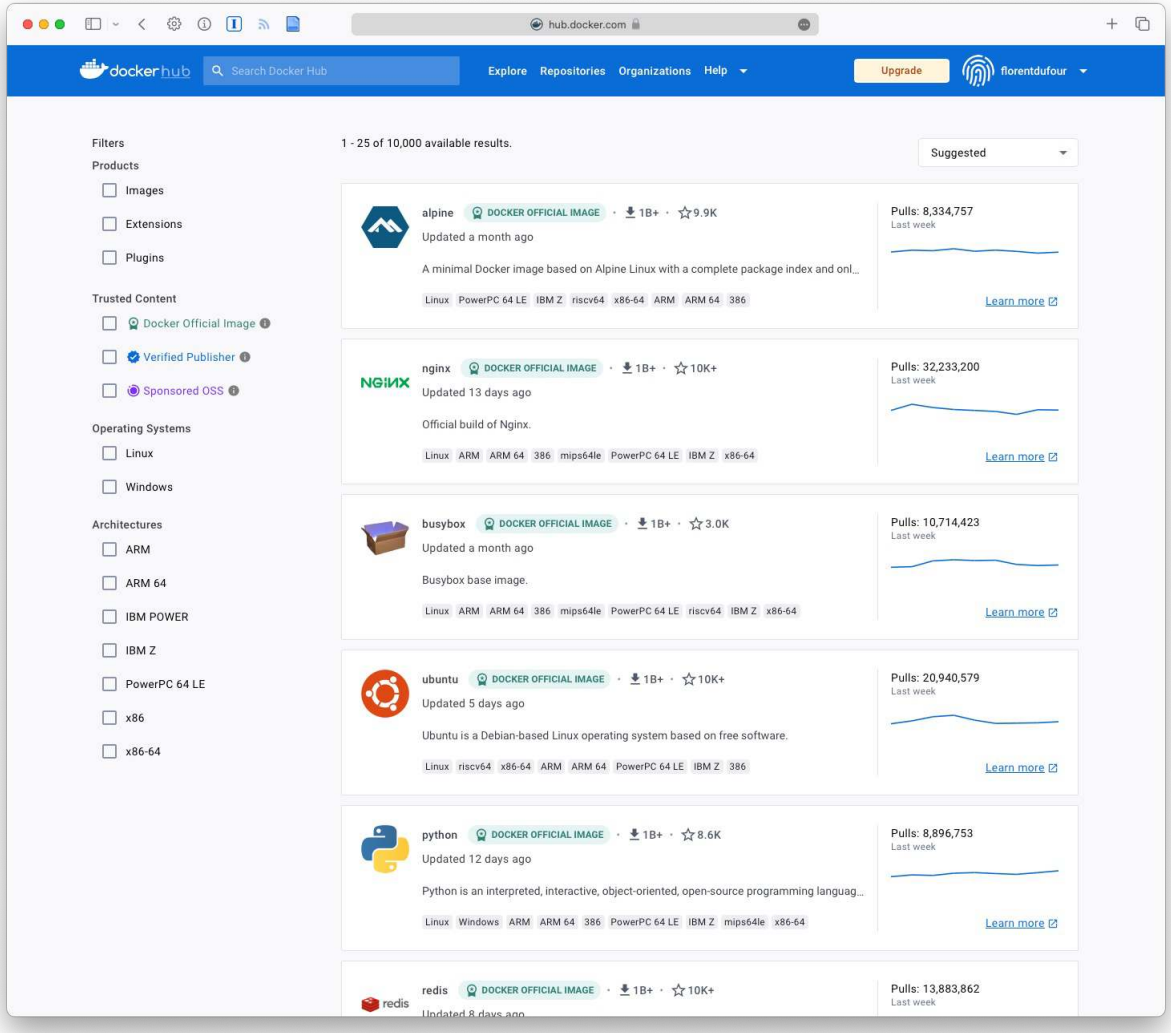
Volume and port mapping

```
# Let's expose a service, for example a website!

# Create your website
echo "<h1>Welcome...</h1><p>...to my awesome website running in a container ;-)</p
    >" > index.html

# -p lets us map ports with the syntax: <port_on_host>:<port_in_container>
# -v lets us mount the persitent file to the container
# -d Lets us run the container in detached mode (i.e. in the background)
# caddy is a web server. Kind of like nginx or apache, but shinier!
docker run \
    --name webserver \
    -d \
    -p 8888:80 \
    -v $PWD/index.html:/usr/share/caddy/index.html \
    caddy

# Visit http://<IP address>:8888 with your browser to see your website
```

host

container

Find the image you need: The Docker Hub

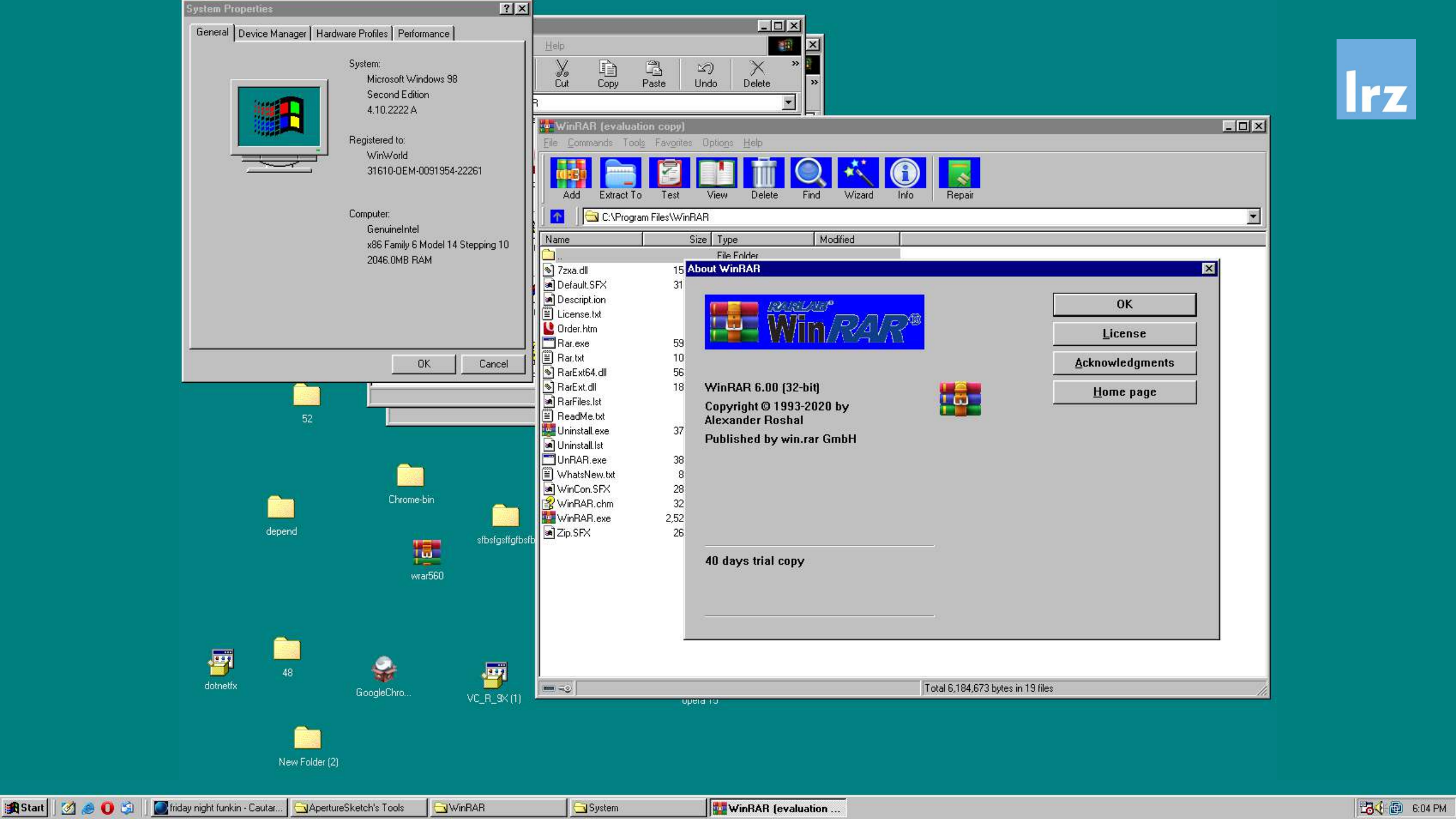


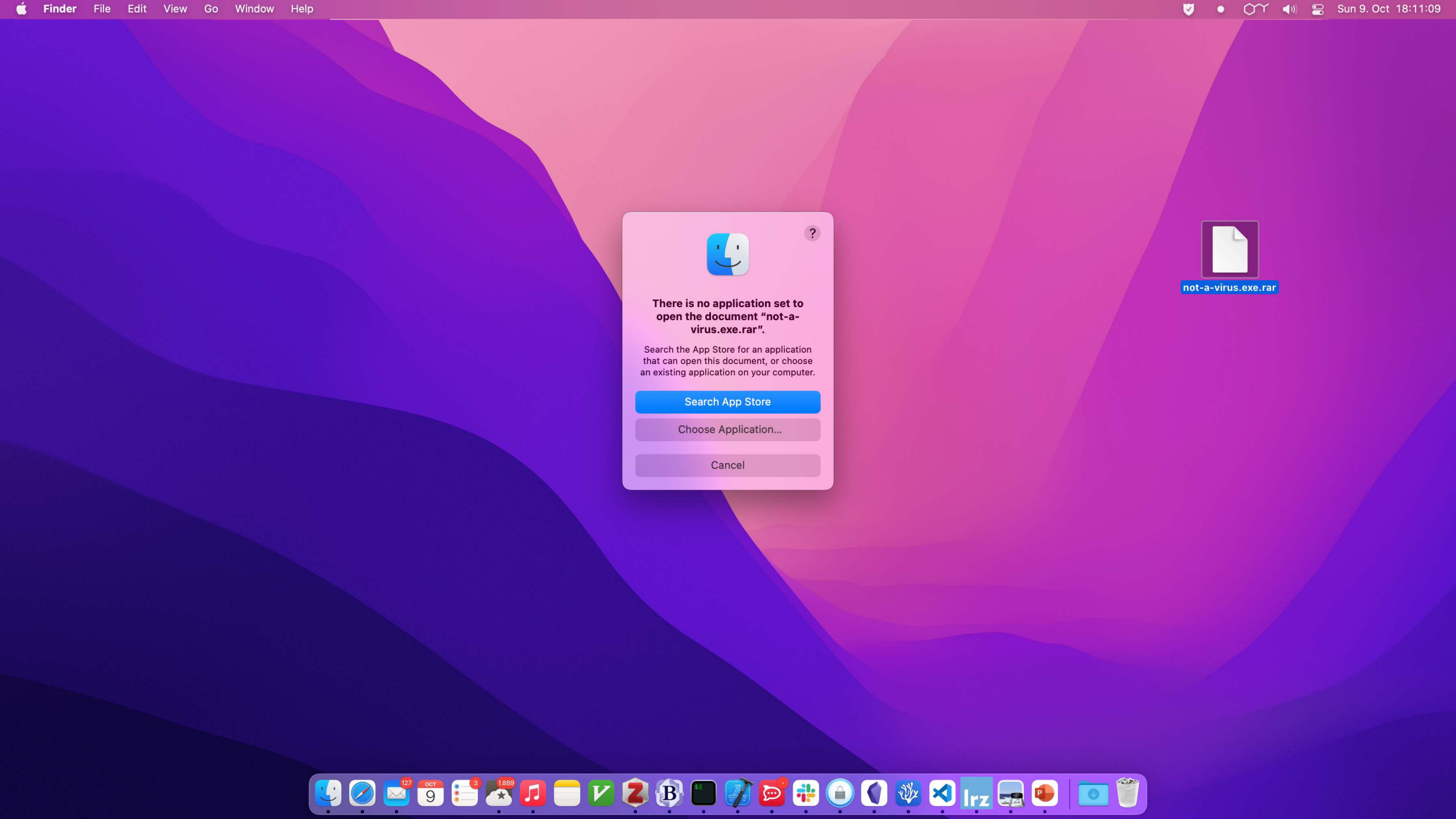
https://hub.docker.com

How to create and use your own custom image

“Commit” a container

Create a “Dockerfile”


System Properties
General
Device Manager
Hardware Profiles
Performance
System:
Microsoft Windows 98
Second Edition
4.10.2222 A
Registered to:
WinWorld
31610-OEM-0091954-22261
Computer:
GenuineIntel
x86 Family 6 Model 14 Stepping 10
2046.0MB RAM
OK
Cancel
Help
Cut
Copy
Paste
Undo
Delete
WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
Repair
C:\Program Files\WinRAR
Name
Size
Type
Modified
File Folder
7zxa.dll 15
Default.SFX 31
Descript.ion
License.txt
Order.htm
Rar.exe 59
Rar.txt 10
RarExt64.dll 56
RarExt.dll 18
RarFiles.lst
ReadMe.txt
Uninstall.exe 37
Uninstall.lst
UnRAR.exe 38
WhatsNew.txt 8
WinCon.SFX 28
WinRAR.chm 32
WinRAR.exe 2,52
Zip.SFX 26
About WinRAR
WinRAR 6.00 (32-bit)
Copyright © 1993-2020 by
Alexander Roshal
Published by win.rar GmbH
40 days trial copy
OK
License
Acknowledgments
Home page
Total 6,184,673 bytes in 19 files
52
Chrome-bin
depend
sfbsfgsffgfbsfb
wrar560
dotnetfx
48
GoogleChro...
VC_R_9X (1)
New Folder (2)
Start
friday night funkin - Cautar...
ApertureSketch's Tools
WinRAR
System
WinRAR (evaluation ...
6:04 PM


Finder File Edit View Go Window Help
Sun 9. Oct 18:11:09
There is no application set to open the document “not-a-virus.exe.rar”.
Search the App Store for an application that can open this document, or choose an existing application on your computer.
Search App Store
Choose Application...
Cancel
not-a-virus.exe.rar

Don’t: Create and use your own custom image (commit)

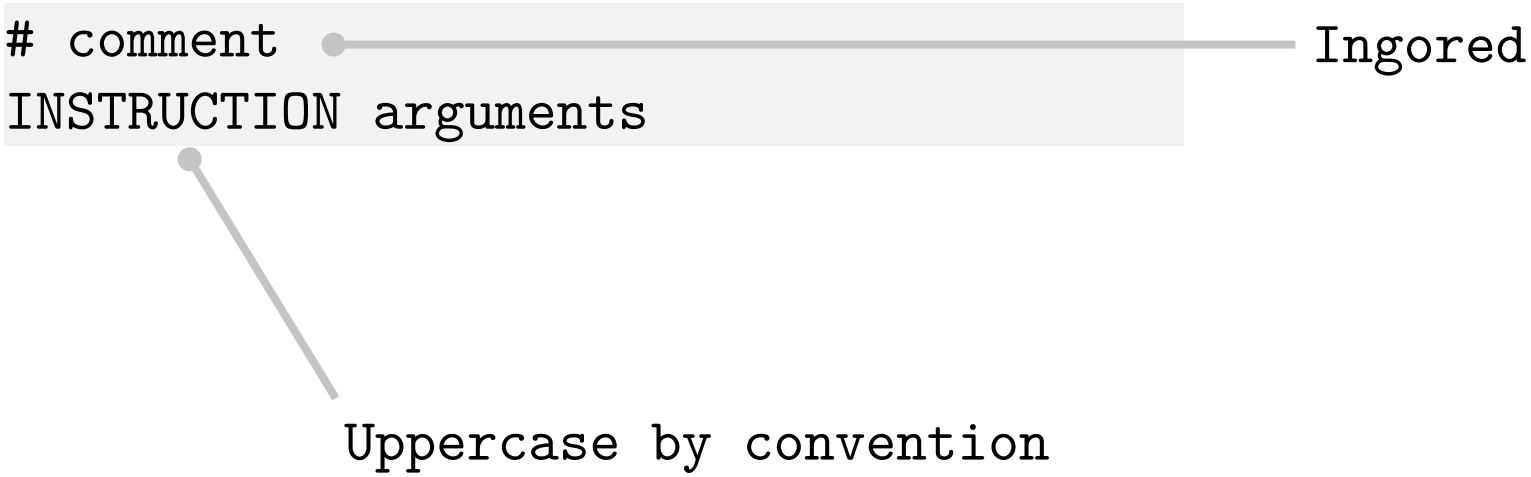

```
# ---------- #
# On the host #
# ---------- #

docker image ls
# REPOSITORY          TAG                 IMAGE ID            CREATED
    SIZE
# alpine              latest              28f6e2705743        5 weeks ago
    5.61MB

# Only alpine is available as an image
# Let’s customize it in order to use it to extract exotic rar files
docker run -it alpine

# --------------- #
# In the container #
# --------------- #

# We add a package to the container
apk update && apk add unrar
# ctrl-P ctrl-Q

# ---------- #
# On the host #
# ---------- #

docker ps
# CONTAINER ID        IMAGE               COMMAND             CREATED
    STATUS              PORTS               NAMES
# 5a1c7e2f8491        alpine              "/bin/sh"           59 seconds ago
    Up 57 seconds                           musing_kilby

# We want to commit the container 5a1c7e2f8491
docker commit -m "unrar capability added to the container" 5a1c7e2f8491 unrar-
    apine
# sha256:e5e572c22a2c84ebcb07c963203c07f89a4b47f848aceb4d80be8afbb884fa3e

docker image ls
# REPOSITORY          TAG                 IMAGE ID            CREATED
    SIZE
# unrar-apine         latest              e5e572c22a2c        55 seconds ago
    9.67MB
# alpine              latest              28f6e2705743        5 weeks ago
    5.61MB

# A new image is created, we can now run alpine container with unrar installed
    already!
```

Do: Create and use your own custom image (Dockerfile)

Create an empty directory

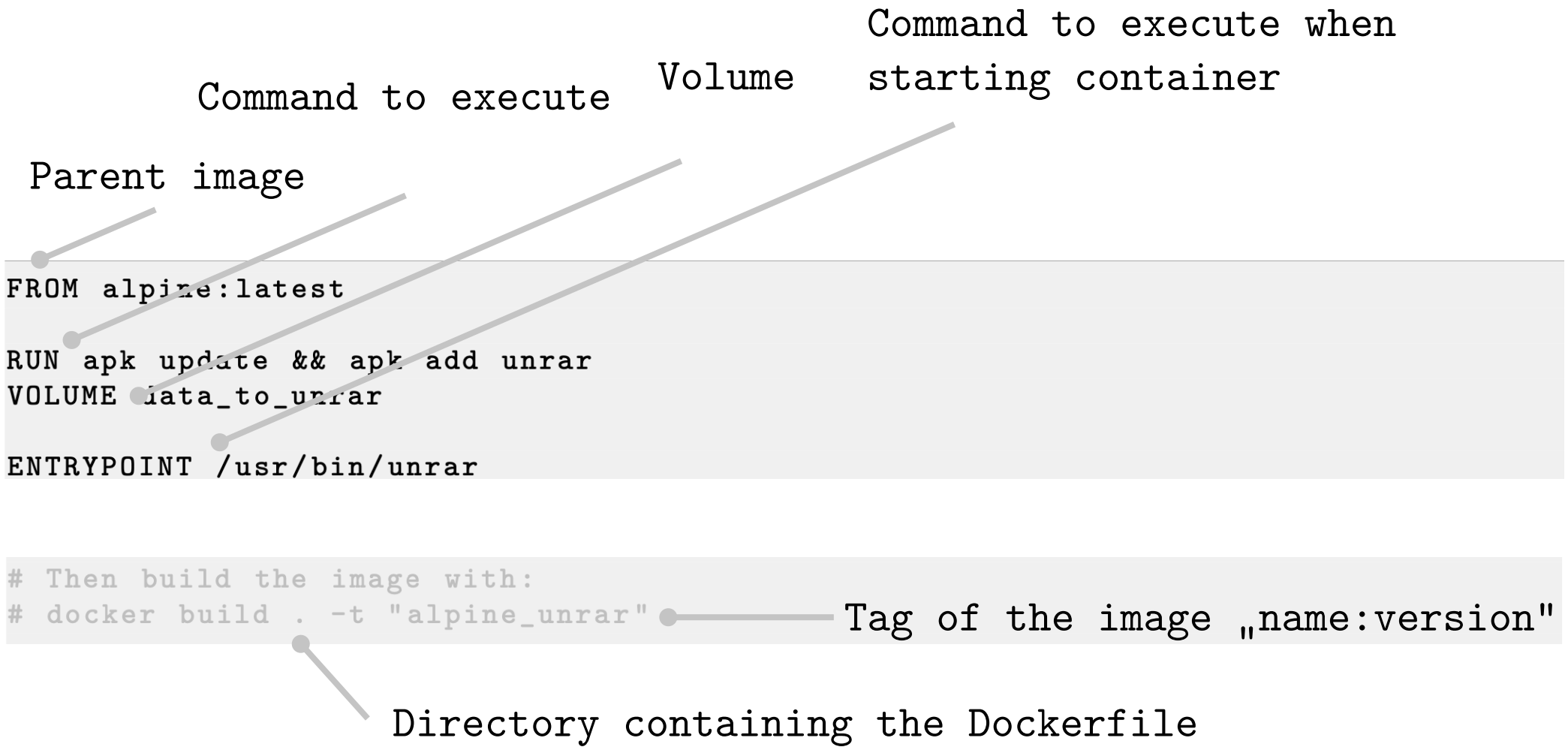
The file must be named Dockerfile

The format is *very* simple:



https://docs.docker.com/engine/reference/builder/#format

Create and use your own custom image (Dockerfile)



Create and use your own custom image (Dockerfile)

```
~$ ls -lah
 232 -rw-r--r--@  1 di67pif  wheel   115K Apr 14 17:23 Module-3_220420.pptx
2056 -rw-r--r--@  1 di67pif  wheel   977K Apr 19 11:40 not-a-virus.exe.rar

~$ docker run -v $PWD:/data_to_unrar alpine_unrar not-a-virus.exe.rar

~$ ls -lah
 232 -rw-r--r--@  1 di67pif  wheel   115K Apr 14 17:23 Module-3_220420.pptx
2056 -rw-r--r--@  1 di67pif  wheel   977K Apr 19 11:40 not-a-virus.exe.rar
2056 -rw-r--r--@  1 di67pif  wheel   2.1M Apr 19 11:43 not-a-virus.exe
```

Example: Create and use your own custom image (Dockerfile)

```
.
├── Dockerfile
├── app
|   ├── assets
|   |   ├── static.lst
|   |   ├── image1.tiff
|   |   └── image2.tiff
|   ├── extractor.py
|   ├── main.py
|   ├── user.py
|   └── utils.py
└── requirements.txt

2 directories, 9 files
```

```
FROM ubuntu:bionic

RUN apt-get update
RUN apt-get -y upgrade
RUN apt-get -y install python3 python3-pip

COPY ./requirements.txt /tmp/requirements.txt
COPY ./app /app

RUN pip install -r /tmp/requirements.txt

ENV DEBUG=true

WORKDIR /app
ENTRYPOINT ./main.py
```

```
~$ docker build . -t "my-image:1.0"
~$ docker run my-image:1.0
```

Create and use your own custom image (Dockerfile)

Instruction	Description
FROM	Set the parent image for the subsequent instructions (must be the firt line) e.g., `FROM alpine:latest`
LABEL	Adds metadata to an image e.g., `LABEL maintainer="Florent Dufour <dufour@lrz.de>"`
ARG	Defines a variables for build time or runtime
RUN	execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step e.g., (shell form), `RUN apt-get update` e.g. (exec form) `RUN ["apt", "update"]`
WORKDIR	sets the working directory for any command to follow
ENV	Set environment variables that must be persistent after image build e.g., `ENV DB_PASSWORD="Chang3me!"`
USER	Sets the user name (or UID) and optionally the user group (or GID) to use when running the image and for later commands to run
ADD	Add files, directories or remote file URLs from to the filesystem of the image. It supports wildcards. Automatically expand archives ⚠ Can be unpredictable. e.g., `ADD /source/file/path /destination/path` e.g., `ADD http://example.com/file.txt /destination`
COPY	Newer than ADD but with limited functionalities. Much safer and predictable, prefer using COPY than ADD e.g., `COPY /host/source/file/path /container/destination/path`
VOLUME	creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers
EXPOSE	Make the container listen to a specific port at runtime. Can specify UDP/TCP. Must be used with -p when runing container e.g., `EXPOSE 8090`
ENTRYPOINT	Configure a container that will run as an executable. Either a command in $PATH or an executable

Hands-on #2: Dream in a container

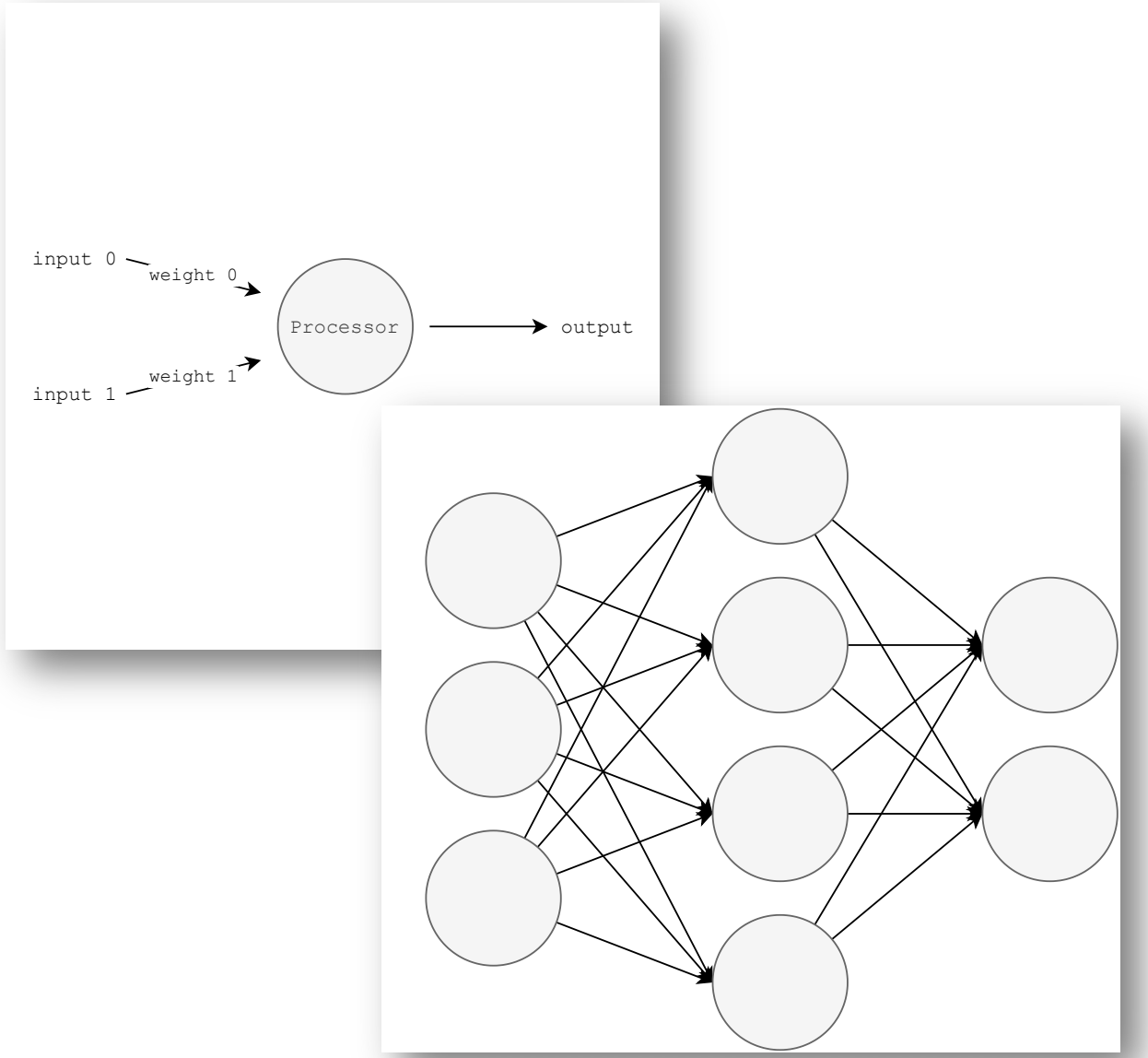
Make an ANN dream in a container - History

"DeepDream is a computer vision program created by Google engineer Alexander Mordvintsev that uses a convolutional neural network to find and enhance patterns in images via algorithmic pareidolia, thus creating a dream-like hallucinogenic appearance in the deliberately over-processed images"

Hands-on #2

Make an ANN dream in a container – ANN ?

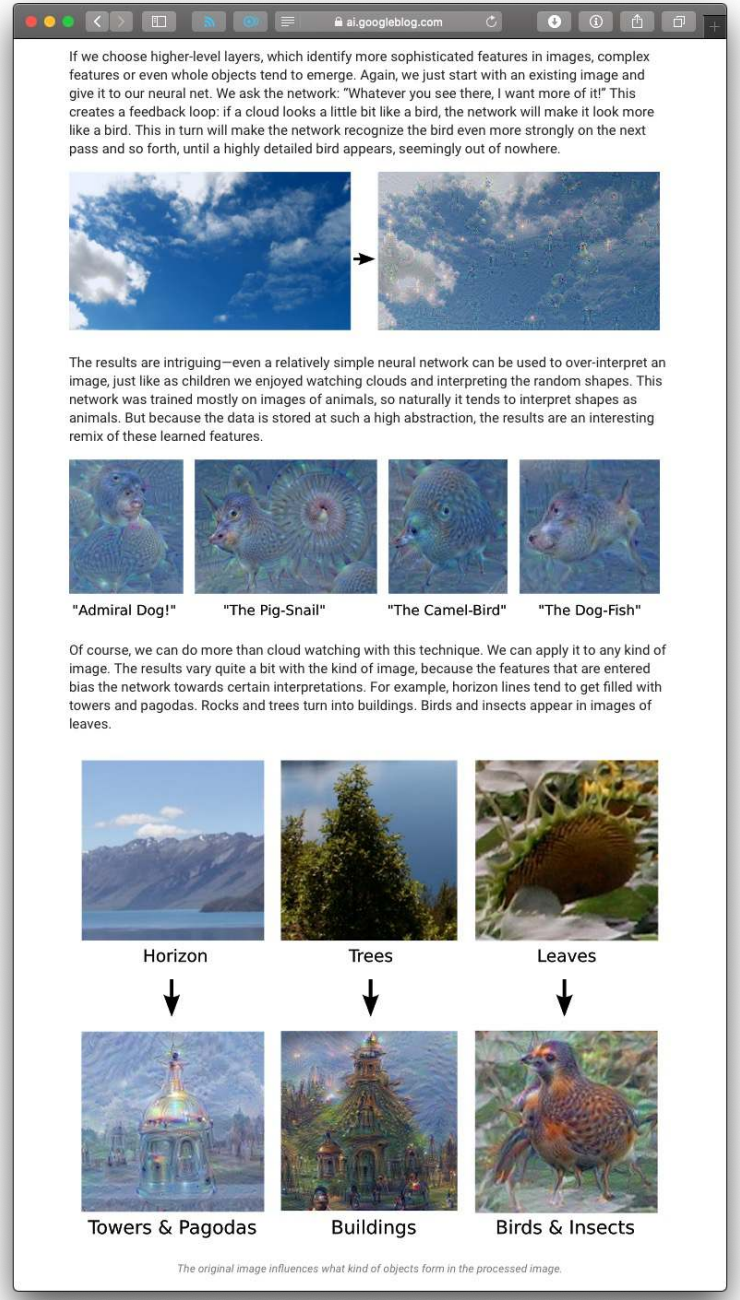
- Like in the brain: a “neuron” lives in a network, receives inputs, processes them, and generates an output.
- An ANN is a “connectionist” computational system that processes information collectively, in parallel throughout a network of neurons organized in layers.
- An ANN is adaptive and has the ability to learn by changing its internal structure based on the information flowing through it.
- This is achieved by tuning weights, the number that controls the signal between two neurons.
- Today, ANN are used to perform “easy-for-a-human, difficult-for-a-machine” tasks like optical character recognition, image classification, and speech and facial recognition for example.



Hands-on #2

Make an ANN dream in a container - History

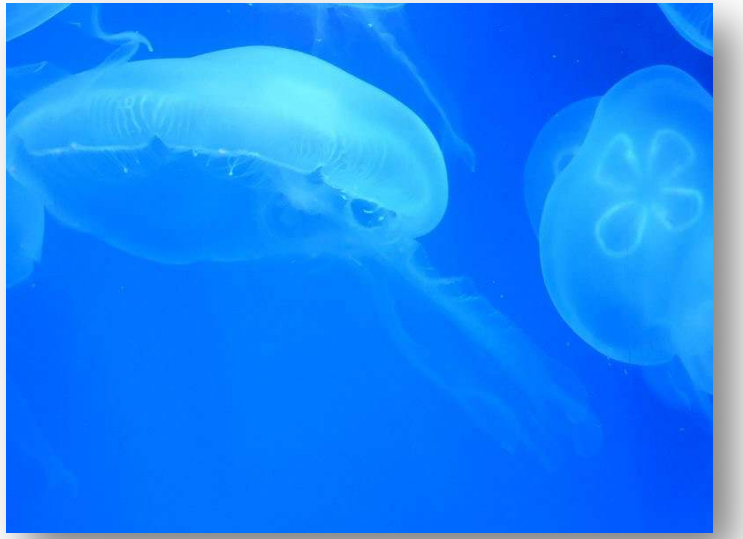
- Originally designed to detect patterns in images for classification (what ANN are good at!)
  - Ex: Find features of a dog: Fur, Dog snout then it's a 🐶
  - Ex: Find features of a fork: a handle and 2-4 tines and ignore what doesn't matter (size, color, number of teeth, orientation) : it's a 🍴
- Arised the question: Why do some models perform well and others don't
- Idea: Reverse the process! And peek into the network
  - What is happening in each layer?
  - Pick a layer and enhance whatever is detected
  - Elicit a particular interpretation: What layer is responsible for what feature?

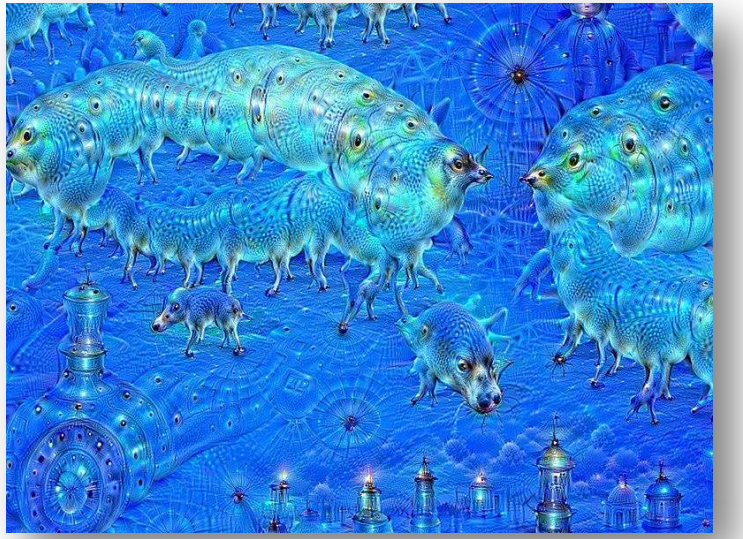


https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Make an ANN dream in a container - History

With an ANN trained to recognize dogs





x10

x5

https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Make an ANN dream in a container - Why

- 😎 It's cool
- 🤓 It requires quite a sophisticated setup
- 😬 It will make use of what we've seen before
- 😅 You'll be proud of yourself when you'll get it working

Hands-on #2

Make an ANN dream in a container – How: Recommendation

- We provide a Jupyter Notebook
- You have to build an image in which the notebook can run
  - Install the dependencies `python3-dev` and `python3-pip` with `apt`
  - Install the dependencies `tensorflow matplotlib` and `jupyterlab` with `pip`
  - Copy the notebook with the container
- You will run a container out of this image and map the port `8888:8888` to access the interface to execute the code from the jupyter notebook

Hands-on #2

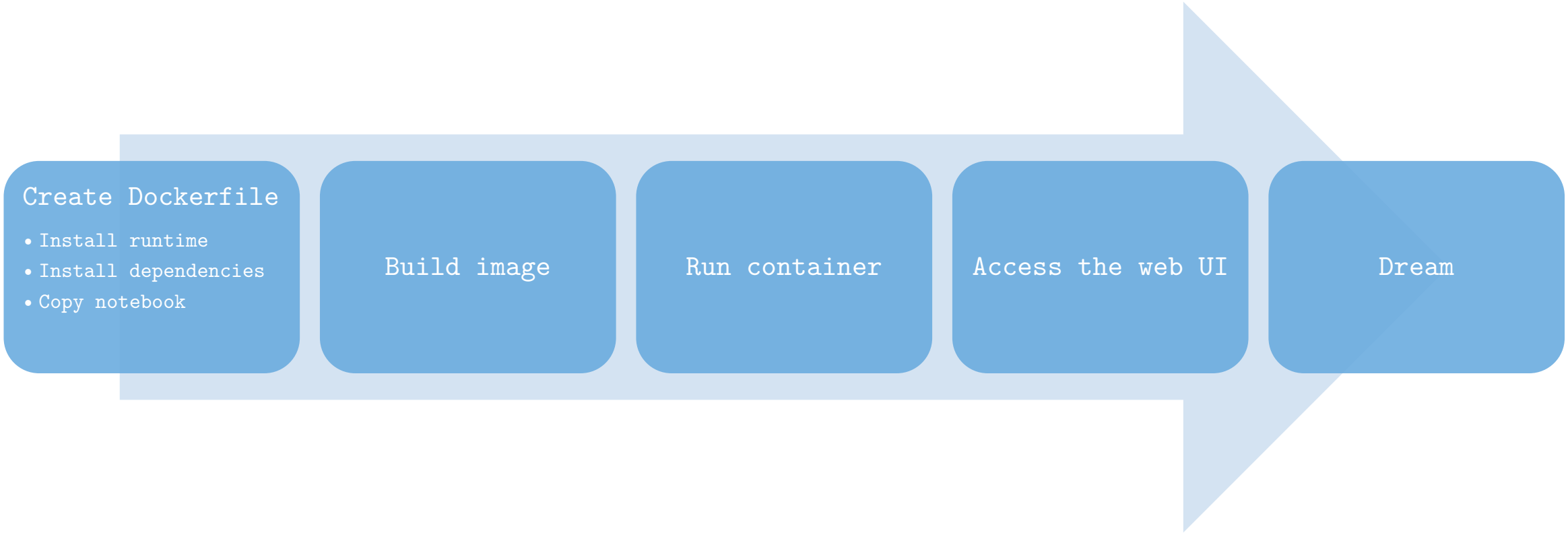
Make an ANN dream in a container - How

```
~$ cp /opt/Exercices.zip .
~$ unzip Exercices.zip
~$ cd Exercices
```

```
~$ tree
.
├── 1-First-steps
|   └── README.md
├── 2-Deep-Learning
|   ├── README.md
|   ├── question
|   |   ├── dream.ipynb
|   |   └── entrypoint.sh
|   └── solution
|       ├── Dockerfile
|       ├── Makefile
|       ├── dream.ipynb
|       ├── entrypoint.sh
|       └──
requirements.txt
├── 3-HPC-AI
|   ├── README.md
|   ├── question
|   └── solution
├── 4-Reproducible-
workflows
|   ├── README.md
|   ├── question
|   └── solution
└── README.md
```

Hands-on #1

Make an ANN dream in a container



Hands-on #2

Make an ANN dream in a container – solution

- You can look into the solution subfolder
- You can run the full solution with `make solution`
- You can individually perform build and run tasks
  - `make build`
  - `make run`

Conatiners, performances, and security

Switching gears

Performances and security

What you get:

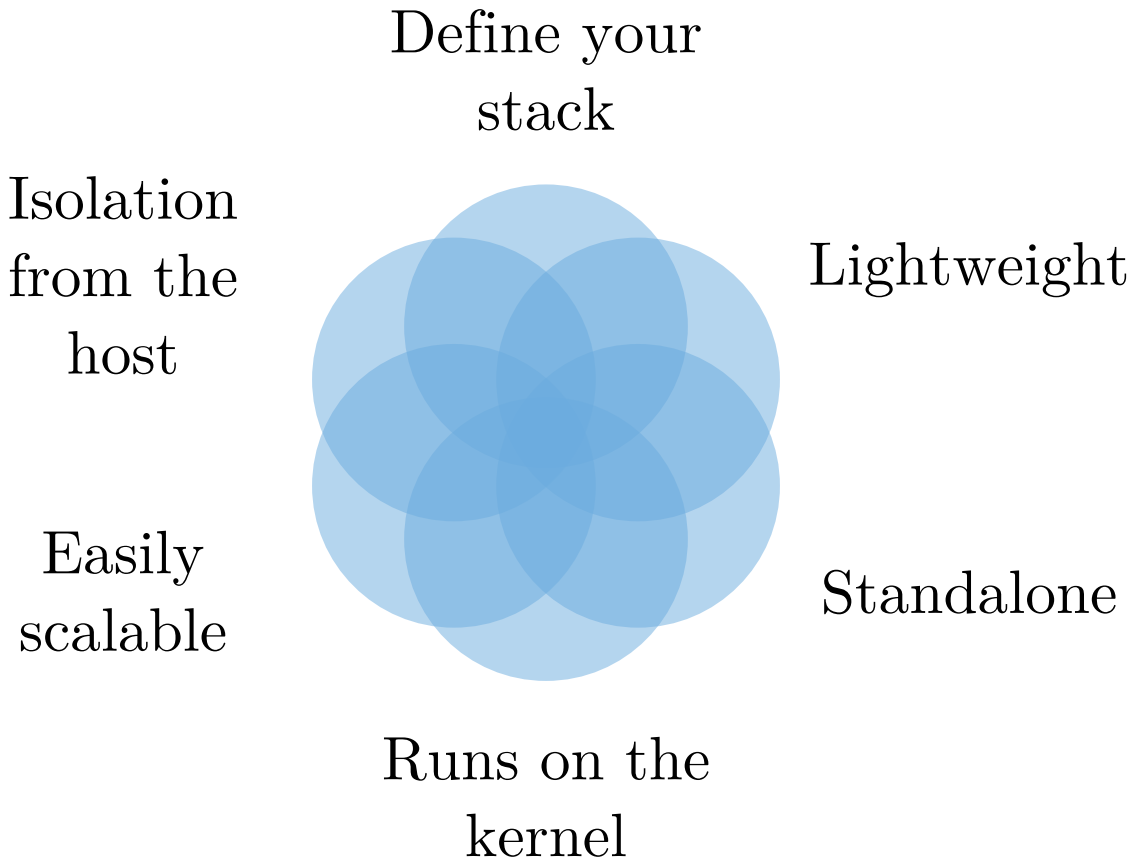
- High Performance
- Massive distribution

What you don't get:

- Root access
- Internet access
- Choice of OS
- `apt install exotic-library`

> Need for UDSS !

Containers allow UDSS



User Defined Software Stack

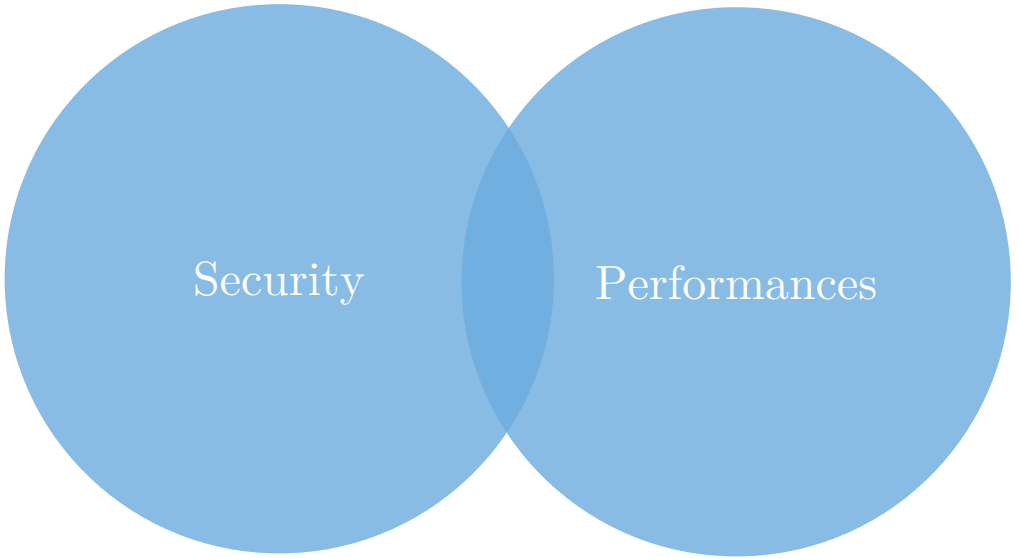
Containers are appealing for HPC

- UDSS
  - Circumvent root access
  - Use exotic libraries and framework
  - A researcher must research
- Close to bare metal performances
- Security
- Less burden on HPC staff = better support

Performances and security

Namespace

Docker socket



Filesystem

Integrity

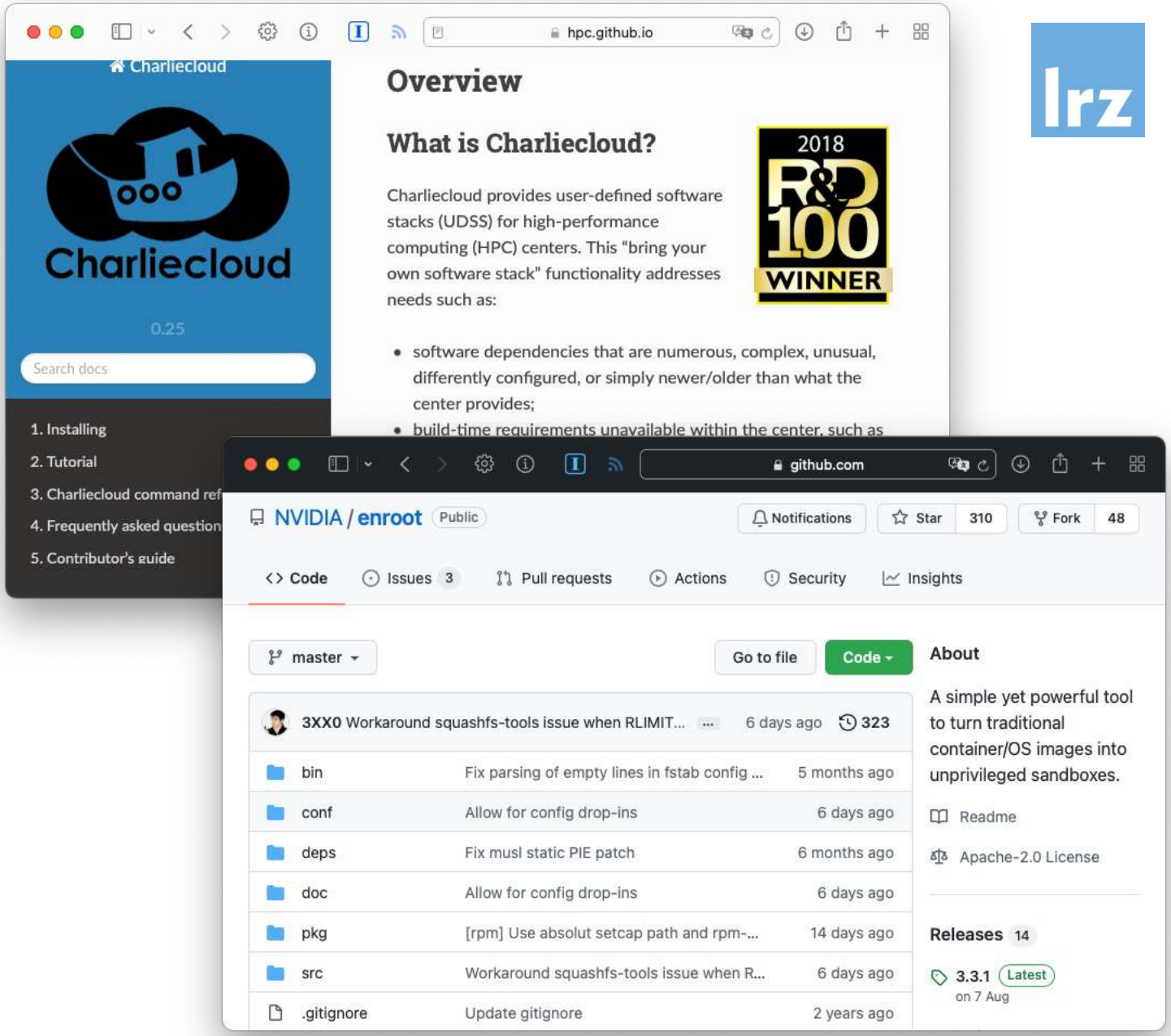
Docker is not HPC friendly

Switching gears

Performances and security

- Still relies on Docker images and Dockerfile as they are widespread BUT do not require any privileged operation
- Makes use of the unprvileged user namespace

1. Create your docker image
2. Convert and flatten the image
3. Upload it to the HPC system
4. Submit your job with a slurm script



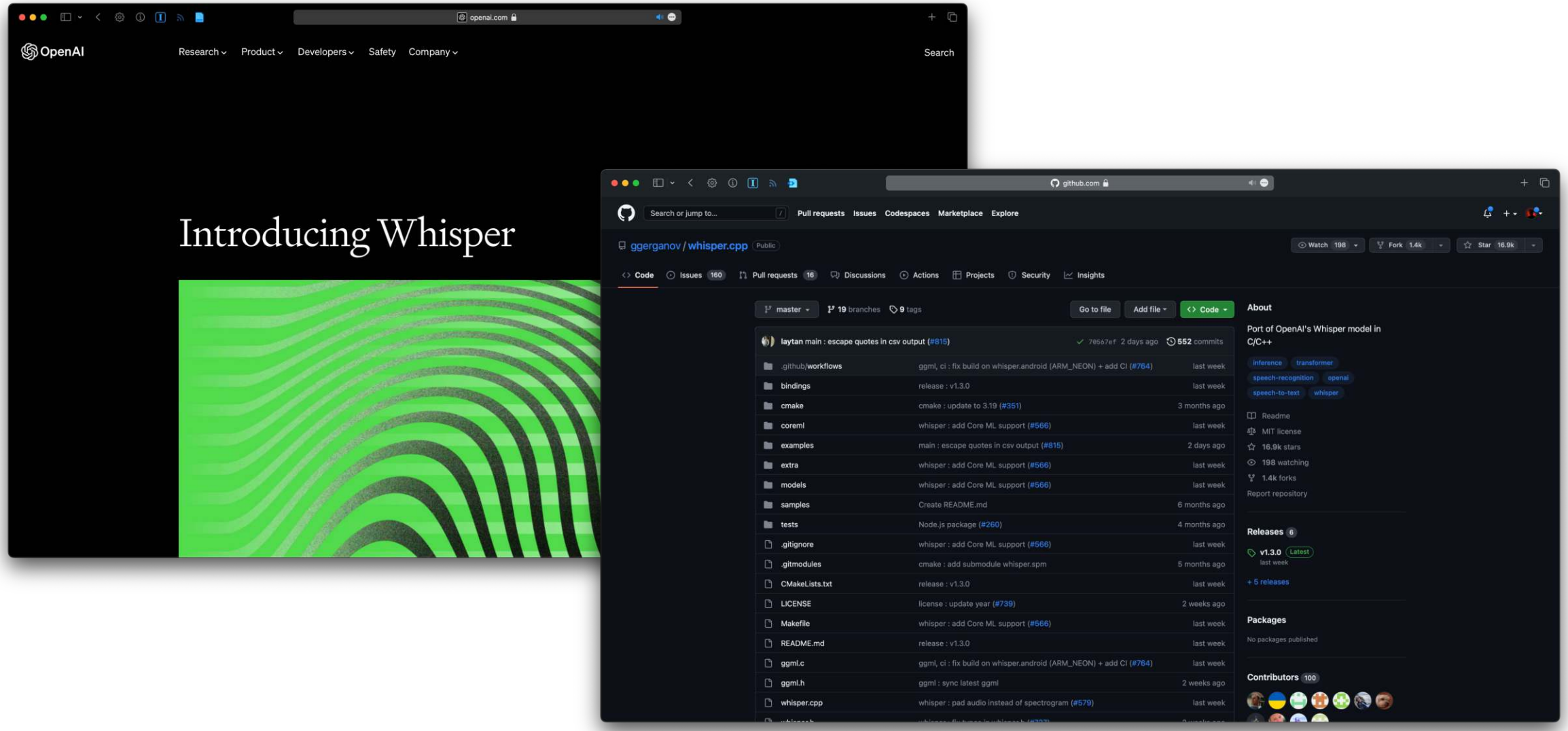
UDSS at LRZ

Charliecloud	enroot
HPC systems - Use docker image - Create a tar file - chroot into it - Run isolate from the system - Fake root inside the container	**AI systems** - Use docker images (from Docker Hub or NGC catalog) - Run in user space - Flow: import, create, start - .sqsh images - Associated with a scheduler - More adapted to AI

Hands-on #3: Containers and HPC

Whisper.cpp

https://openai.com/research/whisper



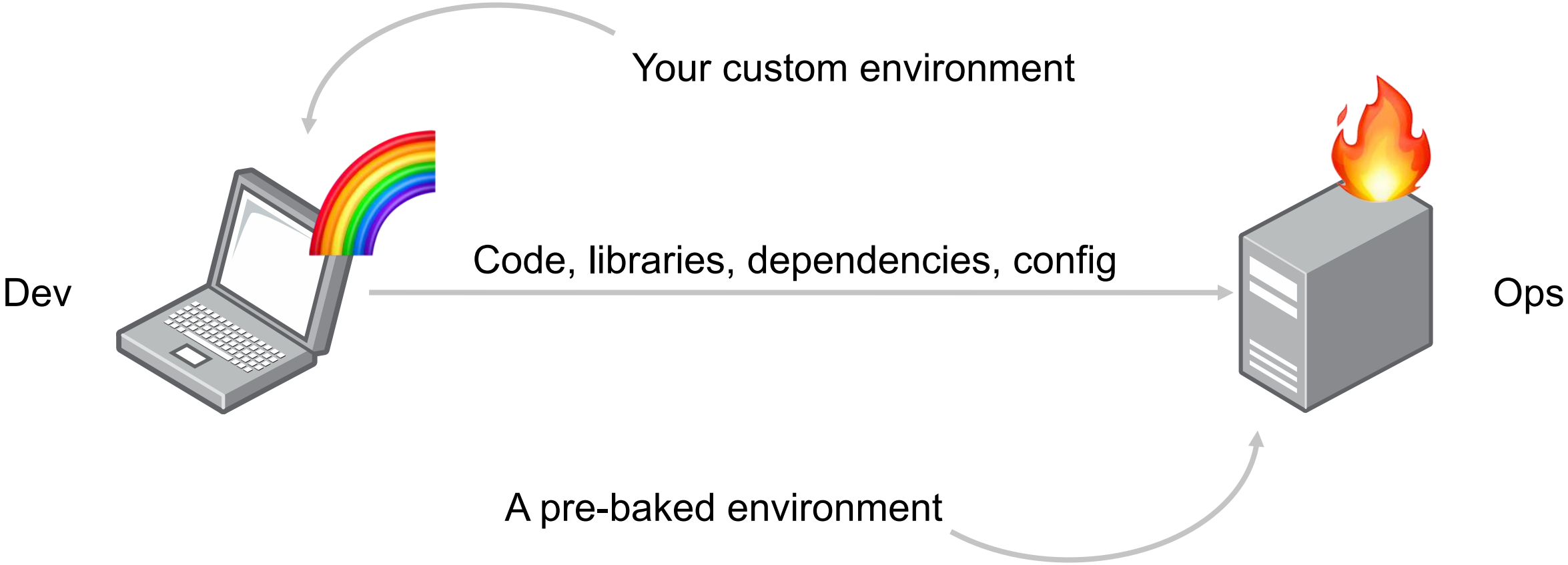
https://github.com/ggerganov/whisper.cpp

- In the question, we provide Dockerfile. You need to
  - Build the image
  - Convert to enroot
  - Create an enroot container
  - Start it while mounting /data to get audio files and a model
- You can convert speech to text!

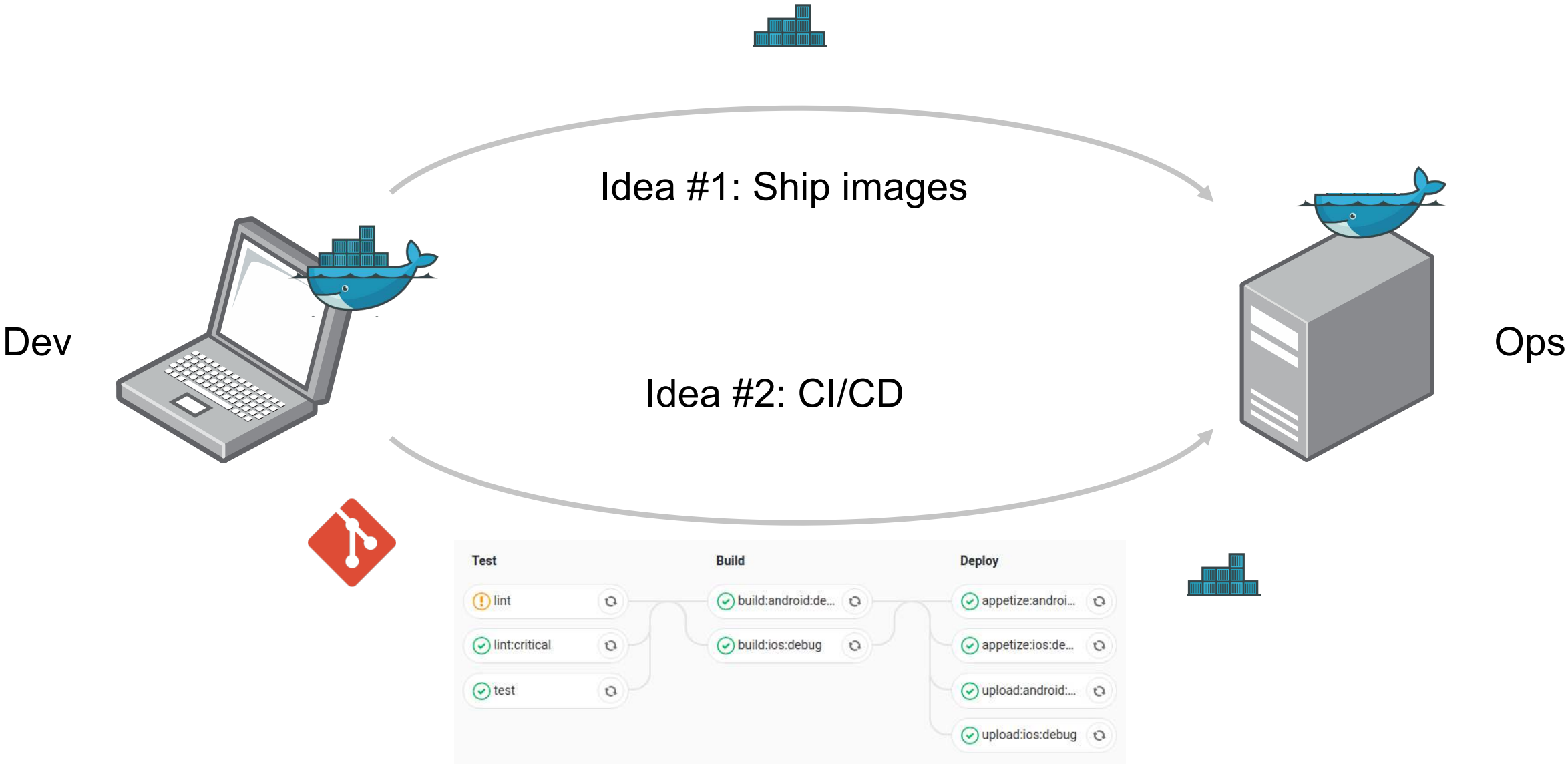
- You can also go into the solution and use `make solution`

Turbo charging containers

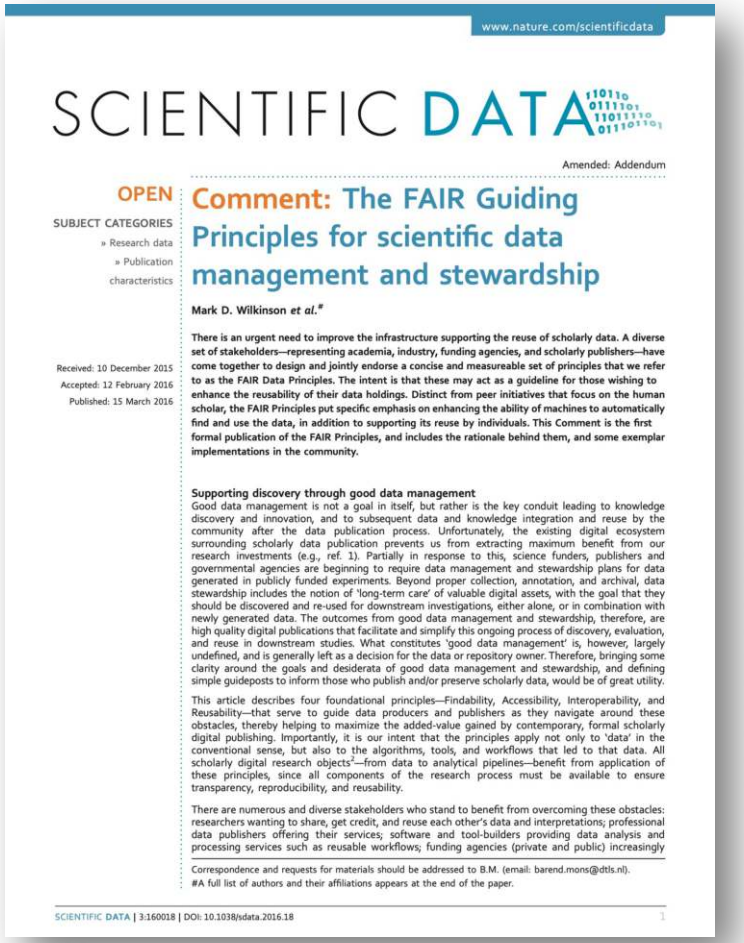
Abstraction



Abstraction: CI/CD



Abstraction: Reproducible scientific pipelines



www.nature.com/scientificdata

SCIENTIFIC DATA

Amended: Addendum

OPEN

SUBJECT CATEGORIES
» Research data
» Publication characteristics

Comment: The FAIR Guiding Principles for scientific data management and stewardship

Mark D. Wilkinson *et al.*#

Received: 10 December 2015
Accepted: 12 February 2016
Published: 15 March 2016

There is an urgent need to improve the infrastructure supporting the reuse of scholarly data. A diverse set of stakeholders—representing academia, industry, funding agencies, and scholarly publishers—have come together to design and jointly endorse a concise and measureable set of principles that we refer to as the FAIR Data Principles. The intent is that these may act as a guideline for those wishing to enhance the reusability of their data holdings. Distinct from peer initiatives that focus on the human scholar, the FAIR Principles put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals. This Comment is the first formal publication of the FAIR Principles, and includes the rationale behind them, and some exemplar implementations in the community.

Supporting discovery through good data management

Good data management is not a goal in itself, but rather is the key conduit leading to knowledge discovery and innovation, and to subsequent data and knowledge integration and reuse by the community after the data publication process. Unfortunately, the existing digital ecosystem surrounding scholarly data publication prevents us from extracting maximum benefit from our research investments (e.g., ref. 1). Partially in response to this, science funders, publishers and governmental agencies are beginning to require data management and stewardship plans for data generated in publicly funded experiments. Beyond proper collection, annotation, and archival, data stewardship includes the notion of 'long-term care' of valuable digital assets, with the goal that they should be discovered and re-used for downstream investigations, either alone, or in combination with newly generated data. The outcomes from good data management and stewardship, therefore, are high quality digital publications that facilitate and simplify this ongoing process of discovery, evaluation, and reuse in downstream studies. What constitutes 'good data management' is, however, largely undefined, and is generally left as a decision for the data or repository owner. Therefore, bringing some clarity around the goals and desiderata of good data management and stewardship, and defining simple guideposts to inform those who publish and/or preserve scholarly data, would be of great utility.

This article describes four foundational principles—Findability, Accessibility, Interoperability, and Reusability—that serve to guide data producers and publishers as they navigate around these obstacles, thereby helping to maximize the added-value gained by contemporary, formal scholarly digital publishing. Importantly, it is our intent that the principles apply not only to 'data' in the conventional sense, but also to the algorithms, tools, and workflows that led to that data. All scholarly digital research objects[2]—from data to analytical pipelines—benefit from application of these principles, since all components of the research process must be available to ensure transparency, reproducibility, and reusability.

There are numerous and diverse stakeholders who stand to benefit from overcoming these obstacles: researchers wanting to share, get credit, and reuse each other's data and interpretations; professional data publishers offering their services; software and tool-builders providing data analysis and processing services such as reusable workflows; funding agencies (private and public) increasingly

Correspondence and requests for materials should be addressed to B.M. (email: barend.mons@dtls.nl).
#A full list of authors and their affiliations appears at the end of the paper.

COMPUTER SCIENCE

Accessible Reproducible Research

As use of computation in research grows, new tools are needed to expand recording, reporting, and reproduction of methods and data.

Jill P. Mesirov

Scientific publications have at least two goals: (i) to announce a result and (ii) to convince readers that the result is correct. Mathematics papers are expected to contain a proof complete enough to allow knowledgeable readers to fill in any details. Papers in experimental science should describe the results and provide a clear enough protocol to allow successful repetition and extension.

Over the past ~35 years, computational science has posed challenges to this traditional paradigm—from the publication of the four-color theorem in mathematics (*1*), in which the proof was partially performed by a computer program, to results depending on computer simulation in chemistry, materials science, astrophysics, geophysics, and climate modeling. In these settings, the scientists are often sophisticated, skilled, and innovative programmers who develop large, robust software packages.

More recently, scientists who are not themselves computational experts are conducting data analysis with a wide range of modular software tools and packages. Users may often combine these tools in unusual or novel ways. In biology, scientists are now routinely able to acquire and explore data sets far beyond the scope of manual analysis, including billions of DNA bases, millions of genotypes, and hundreds of thousands of RNA measurements. Similar issues may arise in other fields, such as astronomy, seismology, and meteorology. While propelling enormous progress, this increasing and sometimes "indirect" use of computation poses new challenges for scientific publication and replication. Large data sets are often analyzed many times, with modifications to the methods and parameters, and sometimes even updates of the data, until the final results are produced. The resulting publication often gives only scant attention to the computational details. Some have suggested these papers are "merely the advertisement of scholarship whereas the computer programs, input data, parameter values, etc. embody the scholarship itself" (*2*). However, the actual code or software "mashup" that gave rise to the final analysis may be lost or unrecoverable.

For example, colleagues and I published a computational method for distinguishing between two types of acute leukemia, based on large-scale gene expression profiles obtained from DNA microarrays (*3*). This paper generated hundreds of requests from scientists interested in replicating and extending the results. The method involved a complex pipeline of steps, including (i) preprocessing of the data, to eliminate likely artifacts; (ii) selection of genes to be used in the model; (iii) building the actual model and setting the appropriate parameters for it from the training data; (iv) preprocessing independent test data; and finally (v) applying the model to test its efficacy. The result was robust and replicable, and the original data were available online, but there was no standardized form in which to make available the various software components and the precise details of their use.

Reproducible Research

This experience motivated the creation of a way to encapsulate all aspects of our in silico analyses (*3*) in a manner that would facilitate independent replication by another scientist (*4*). Computer and computational scientists refer to this goal as "reproducible research" (*5*), a coinage attributed to the geophysicist Jon Claerbout in 1990, who imposed the standard of makefiles for construction of all the figures and computational results in papers published by the Stanford Exploration Project (*6*). Since that time, other approaches have been proposed (*7*–*14*), including the ability to insert active scripts within a text document (*15*) and the use of a markup language that can produce all of the text, figures, code, algorithms, and settings used for the computational research (*16*). Although these approaches may accomplish the goal, they are not practical for many nonprogramming experimental scientists using other groups' or commercial software tools today.

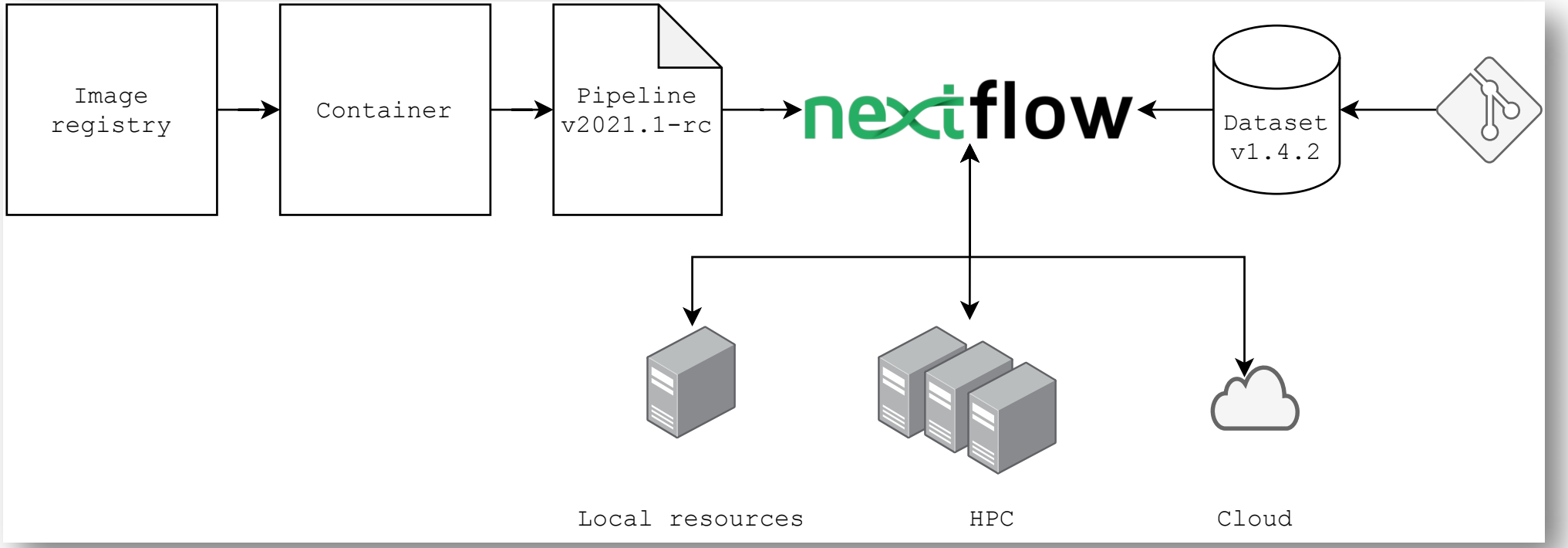
A similar challenge was encountered more than 20 years ago when scientists wanting to access data from remote computers had to write their own retrieval programs. The solution was the invention of the World Wide Web (*17*), together with the concept of "Web browsers" such as MOSAIC (*18*) and its successors. The approach was so effective that we now take it for granted.

In the same spirit, we need a paradigm that makes it simple, even for scientists who do not themselves program, to perform and publish reproducible computational research. Toward this end, we propose a Reproducible Research System (RRS), consisting of two components. The first element is a Reproducible Research Environment (RRE) for doing the computational work. An RRE provides computational tools together with the ability to automatically track the provenance of data, analyses, and results and to package them (or pointers to persistent versions of them) for redistribution. The second element is a Reproducible Research Publisher (RRP), which is a document-preparation system, such as standard word-processing software, that provides an easy link to the RRE. The RRS thus makes it easy to perform analyses and then to embed them directly into a

Broad Institute of Massachusetts Institute of Technology and Harvard University, Cambridge, MA 02142, USA. E-mail: mesirov@broad.mit.edu

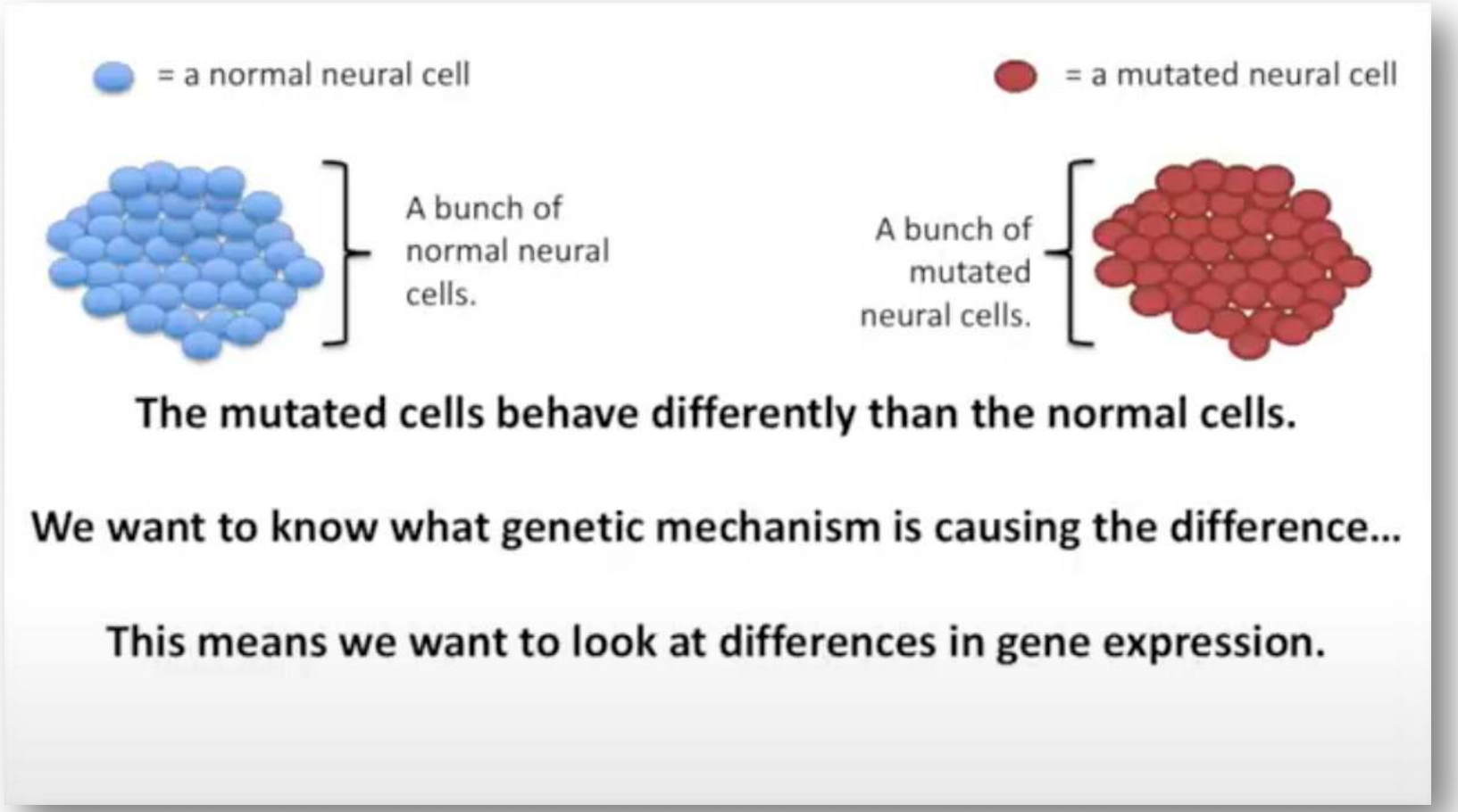
CREDIT: JOE SUTLIFF/WWW.CSMD.COM/JOE

Abstraction: Reproducible scientific pipelines

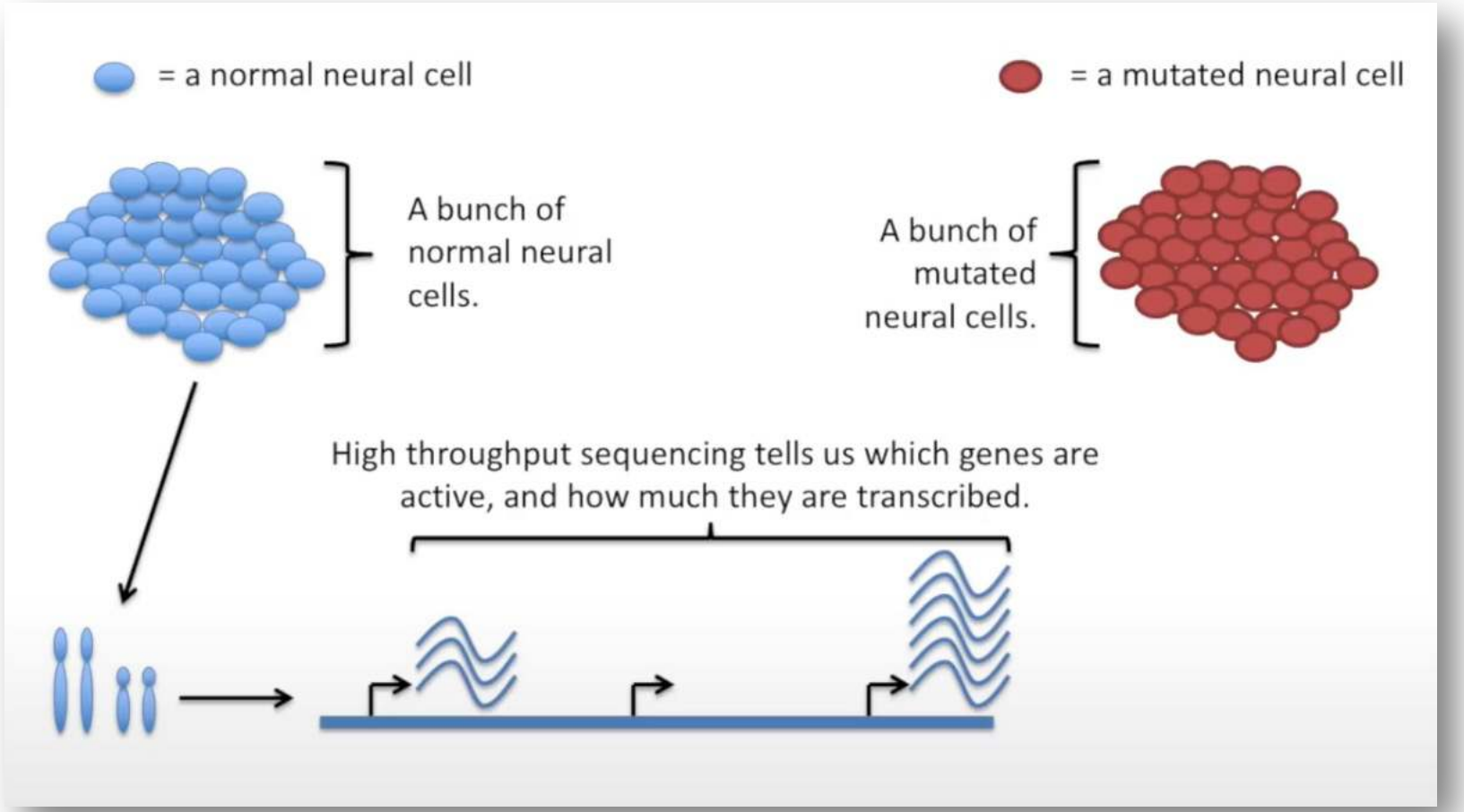


Hands-on #4 Reproducible scientific pipelines

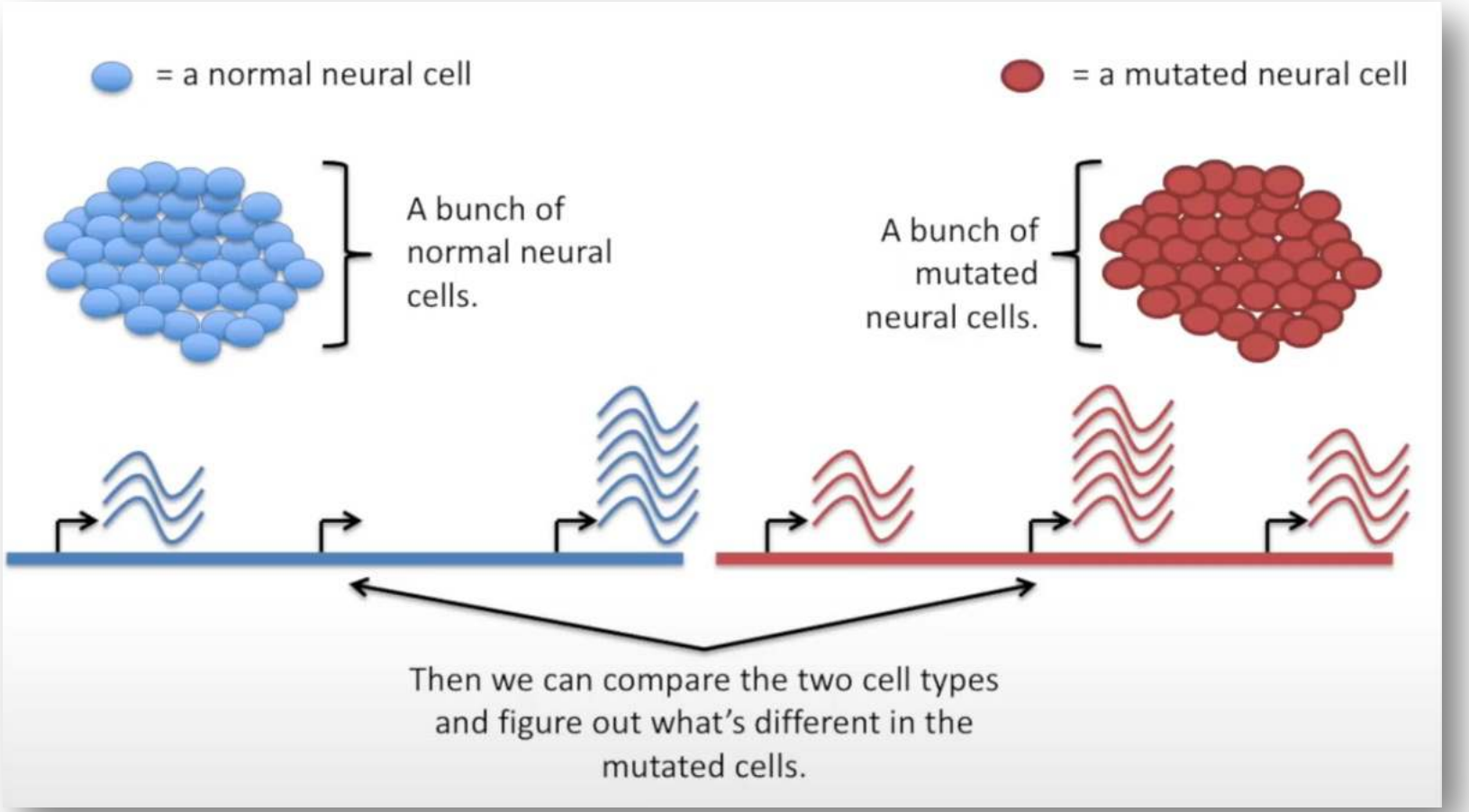
What is RNA-seq (StatQuest)



What is RNA-seq (StatQuest)



What is RNA-seq (StatQuest)



What is RNA-seq

We measure the gene expression in both cell populations and compare the results to see what's happening in the mutated cells.

A RNA-Seq experiment usually occurs in 3 main steps:

i) Biological sample preparation (preparation of the library)
ii) Sequencing
iii) Data analysis

An illumina sequencer

What is RNA-seq (Data analysis)

Data analysis:

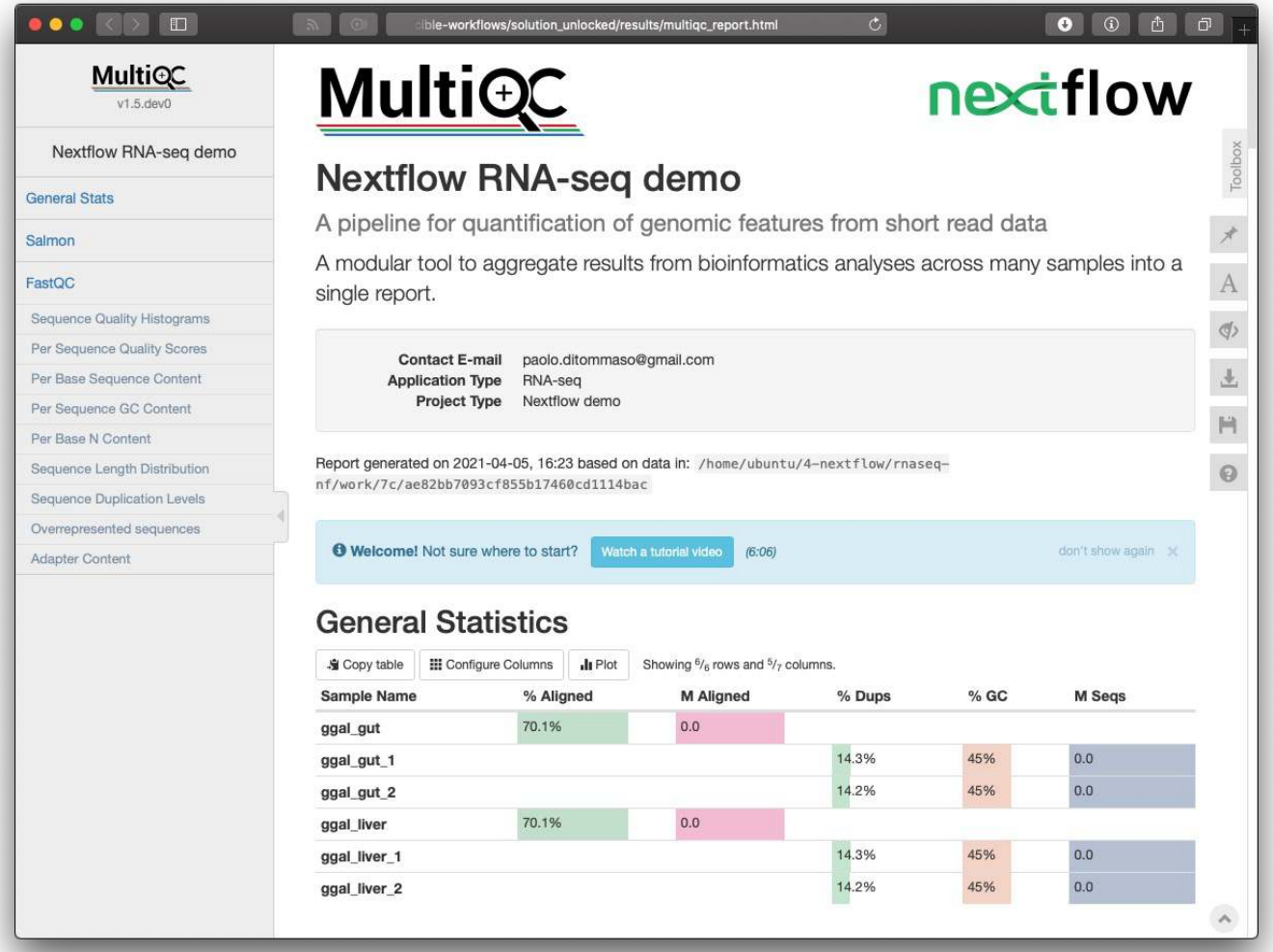
- FastQC: To quality check the sequencing. Sequences with poor quality must be trimmed or filtered.
- MultiQC: Also to quality check, with additional information.
- Salmon: That must be run after the quality check and reads filtering. Salmon allows the mapping of high quality reads on a genome and a genes set in order to establish the differential gene expression.

What is RNA-seq (Data analysis)

```
# Let's get the Nextflow code along with the dataset
git clone "https://github.com/nextflow-io/rnaseq-nf.git"
#Cloning into 'rnaseq-nf'...

# And run the pipeline locally in a docker container
cd rnaseq-nf
nextflow run nextflow-io/rnaseq-nf -with-docker
# R N A S E Q - N F   P I P E L I N E
#  ===================================
#  transcriptome: /home/ubuntu/.nextflow/assets/nextflow-io/rnaseq-nf/data/ggal/
   ggal_1_48850000_49020000.Ggal71.500bpflank.fa
#  reads        : /home/ubuntu/.nextflow/assets/nextflow-io/rnaseq-nf/data/ggal/*_
   {1,2}.fq
#  outdir       : results
# executor >  local (6)
# [06/dd0ce9] process > RNASEQ:INDEX (ggal_1_48850000_49020000) [100%] 1 of 1 /
# [a5/514067] process > RNASEQ:FASTQC (FASTQC on ggal_liver)    [100%] 2 of 2 /
# [9c/af0a9d] process > RNASEQ:QUANT (ggal_liver)               [100%] 2 of 2 /
# [70/d1650e] process > MULTIQC                                 [100%] 1 of 1 /
# Done! Open the following report in your browser --> results/multiqc_report.html

# That's it ! The rport is published in the results folder!
# Use scp to get visualize it on your local machine
```

Can you use the caddy web server (as illustrated on code snippet 6) to expose the result of the pipeline as a web page accessible on port 8888?

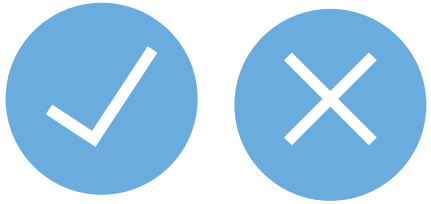
What is RNA-seq (Data analysis)



Take home message

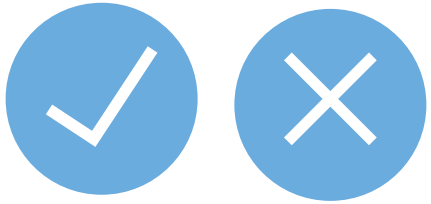
Take-home message

Containers are the ephemeral running instance of an environment: an application, its runtime, dependencies, libraries, settings etc.



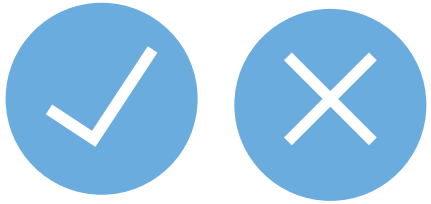
Take-home message

Processes are executed in isolation,thanks to a kernel feature called namespace and have limited access to resources thanks to another feature named cgroups.



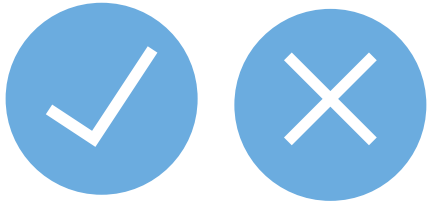
Take-home message

They run independently of the underlying infrastructure

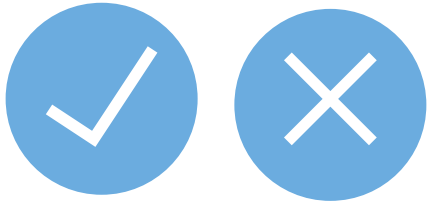


Take-home message

Virtual Machine (VM)s are cooler than containers



Containers can not mount volumes from the host and expose ports.



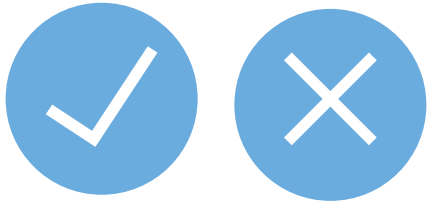
Take-home message

Containers can be committed to images, manually or thanks to a Domain-Specific Language (DSL) like the Dockerfile.

The syntax of the Dockerfile is expressive.

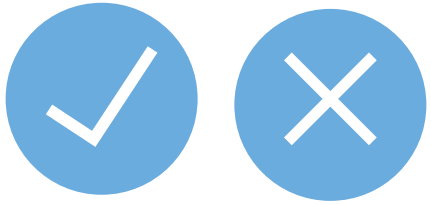
The Dockerfile is the cornerstone of all container platforms.

They are easy to write, share, and build.



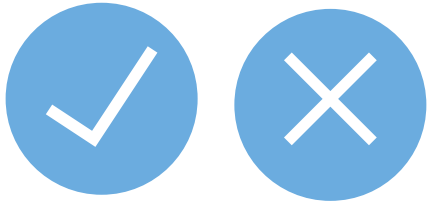
Take-home message

Once can convert a Docker image to other format like enroot and Charliecloud which are suited to HPC applications



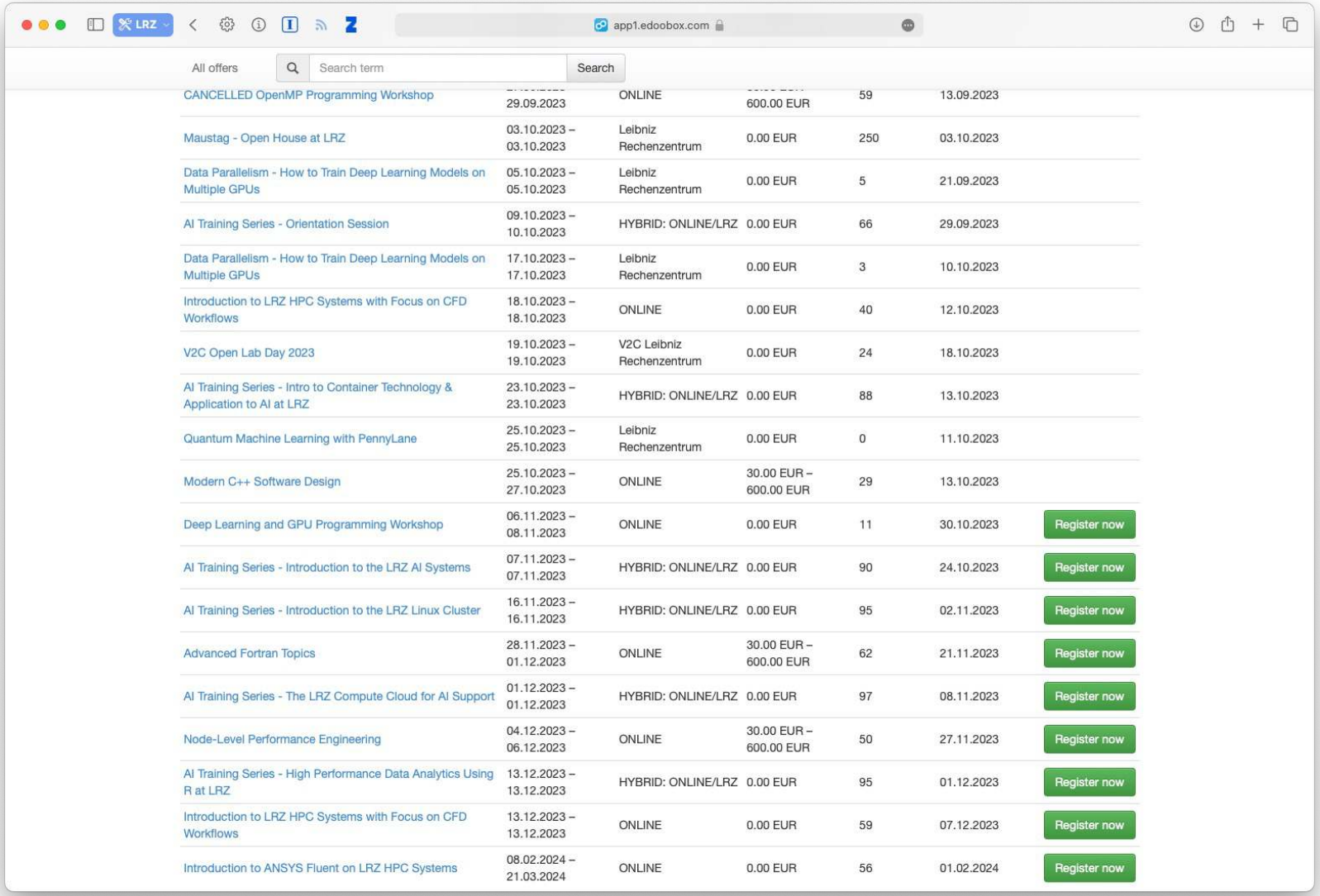
Containers allow

Containers allow onsistency and reproducibility: in many regards, software build, scientific pipelines etc. You to spend more time on your code and your problematic, less on time consuming friction.



Upcoming courses

Look for the AI Training Series (free)



CANCELLED OpenMP Programming Workshop	29.09.2023	ONLINE	600.00 EUR	59	13.09.2023	
Maustag - Open House at LRZ	03.10.2023 – 03.10.2023	Leibniz Rechenzentrum	0.00 EUR	250	03.10.2023	
Data Parallelism - How to Train Deep Learning Models on Multiple GPUs	05.10.2023 – 05.10.2023	Leibniz Rechenzentrum	0.00 EUR	5	21.09.2023	
AI Training Series - Orientation Session	09.10.2023 – 10.10.2023	HYBRID: ONLINE/LRZ	0.00 EUR	66	29.09.2023	
Data Parallelism - How to Train Deep Learning Models on Multiple GPUs	17.10.2023 – 17.10.2023	Leibniz Rechenzentrum	0.00 EUR	3	10.10.2023	
Introduction to LRZ HPC Systems with Focus on CFD Workflows	18.10.2023 – 18.10.2023	ONLINE	0.00 EUR	40	12.10.2023	
V2C Open Lab Day 2023	19.10.2023 – 19.10.2023	V2C Leibniz Rechenzentrum	0.00 EUR	24	18.10.2023	
AI Training Series - Intro to Container Technology & Application to AI at LRZ	23.10.2023 – 23.10.2023	HYBRID: ONLINE/LRZ	0.00 EUR	88	13.10.2023	
Quantum Machine Learning with PennyLane	25.10.2023 – 25.10.2023	Leibniz Rechenzentrum	0.00 EUR	0	11.10.2023	
Modern C++ Software Design	25.10.2023 – 27.10.2023	ONLINE	30.00 EUR – 600.00 EUR	29	13.10.2023	
Deep Learning and GPU Programming Workshop	06.11.2023 – 08.11.2023	ONLINE	0.00 EUR	11	30.10.2023	Register now
AI Training Series - Introduction to the LRZ AI Systems	07.11.2023 – 07.11.2023	HYBRID: ONLINE/LRZ	0.00 EUR	90	24.10.2023	Register now
AI Training Series - Introduction to the LRZ Linux Cluster	16.11.2023 – 16.11.2023	HYBRID: ONLINE/LRZ	0.00 EUR	95	02.11.2023	Register now
Advanced Fortran Topics	28.11.2023 – 01.12.2023	ONLINE	30.00 EUR – 600.00 EUR	62	21.11.2023	Register now
AI Training Series - The LRZ Compute Cloud for AI Support	01.12.2023 – 01.12.2023	HYBRID: ONLINE/LRZ	0.00 EUR	97	08.11.2023	Register now
Node-Level Performance Engineering	04.12.2023 – 06.12.2023	ONLINE	30.00 EUR – 600.00 EUR	50	27.11.2023	Register now
AI Training Series - High Performance Data Analytics Using R at LRZ	13.12.2023 – 13.12.2023	HYBRID: ONLINE/LRZ	0.00 EUR	95	01.12.2023	Register now
Introduction to LRZ HPC Systems with Focus on CFD Workflows	13.12.2023 – 13.12.2023	ONLINE	0.00 EUR	59	07.12.2023	Register now
Introduction to ANSYS Fluent on LRZ HPC Systems	08.02.2024 – 21.03.2024	ONLINE	0.00 EUR	56	01.02.2024	Register now

Course Evaluation

Please visit
https://survey.lrz.de/index.php/413884?lang=en
and rate this course.

Your feedback is highly appreciated!
Thank you!

Upcoming trainings:
https://app1.edoobox.com/en/LRZ/