Technological foundations of software development

Run your software anywhere with Docker

ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Objectives of the session

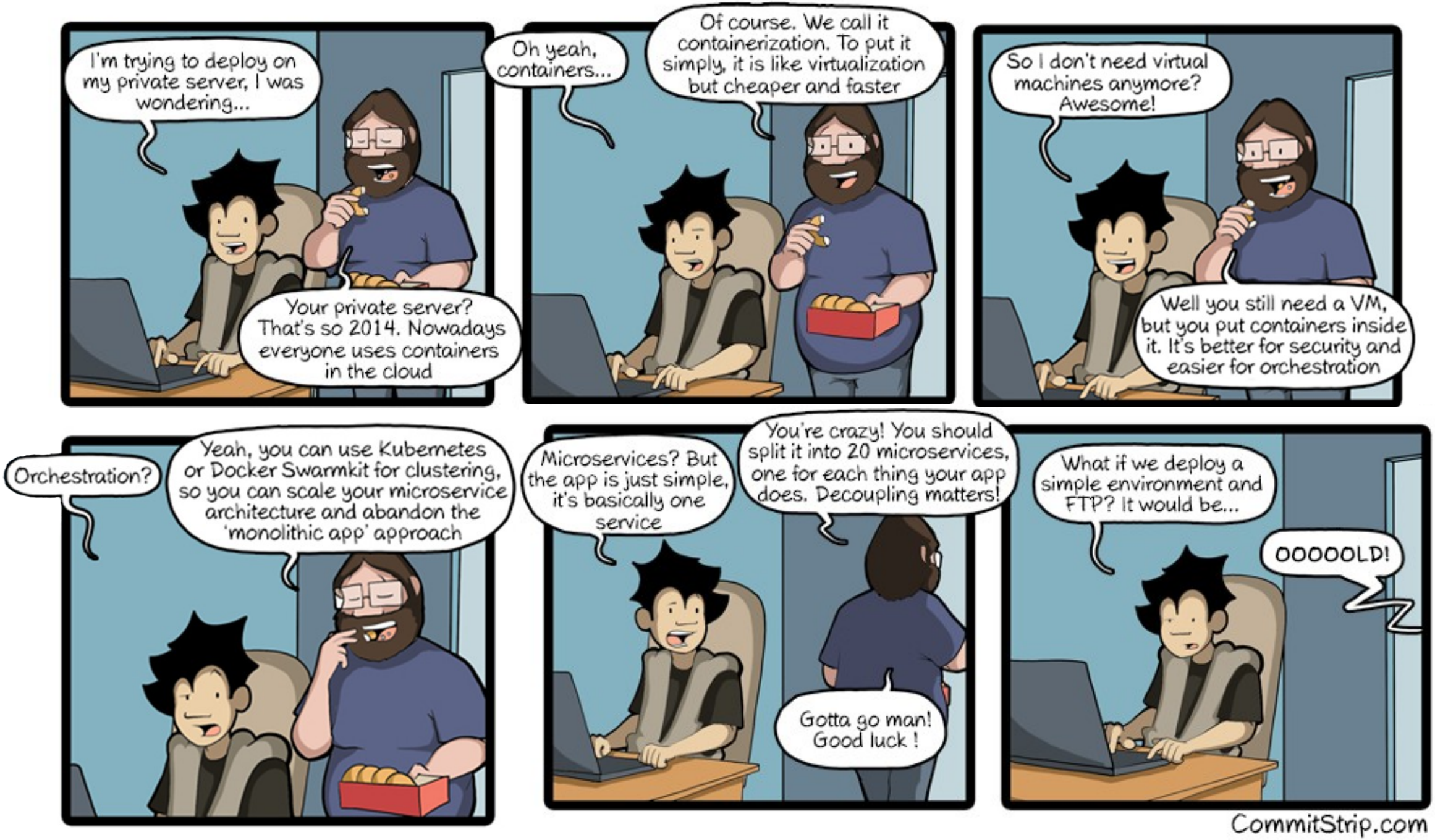
This session aims to familiarize you with Docker: an open source software that can package an application and its dependencies into an isolated container, which can be run on any server

Technological foundations of software development

Run your software anywhere with Docker

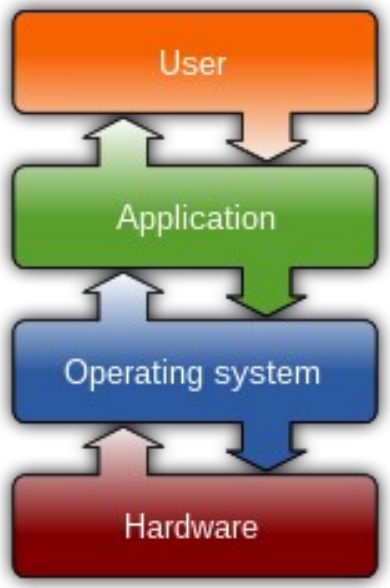
Part 1: Virtualization and Containerization

ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

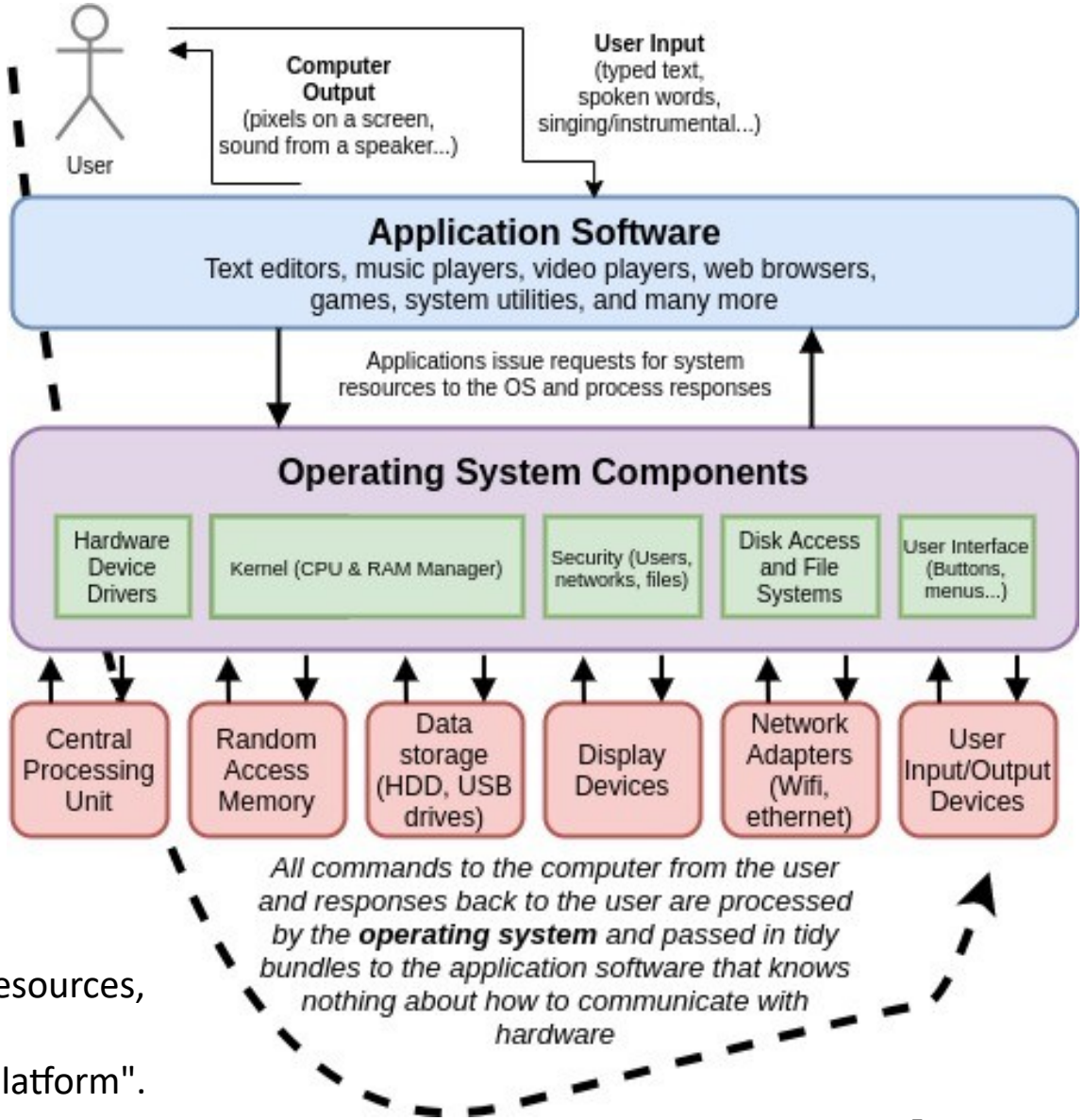


https://www.commitstrip.com/en/2016/12/10/a-server-thats-so-2014/?

Operating system



https://en.wikipedia.org/wiki/Operating_system

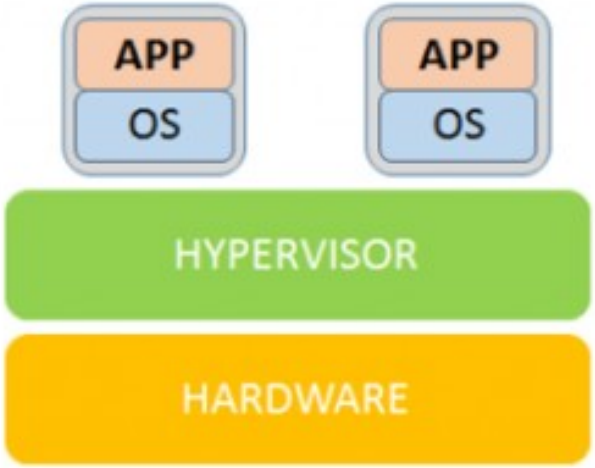


Operating systems connect applications (programs) with system hardware resources, such as disk drives, networks, and user input/output components.
Because all applications rely on the operating system, it is often called the "platform".
ex: Linux, Windows, OSX.

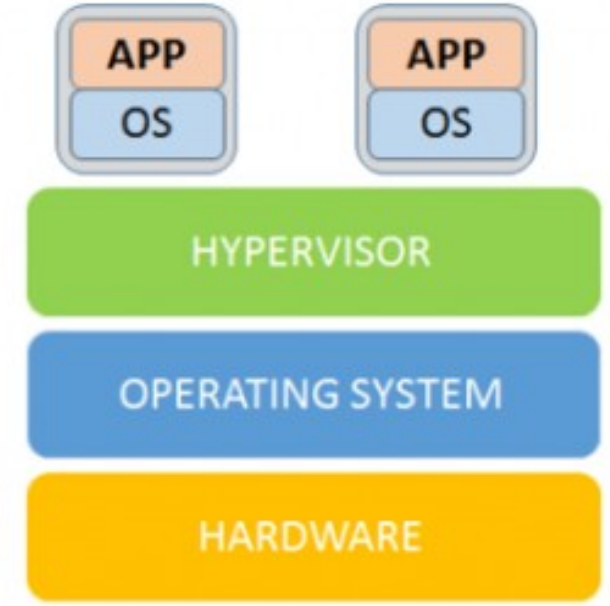
Virtualization (started ~1960s)

A **Hypervisor**, or **virtual machine monitor** is a software that creates and runs virtual machines (VM)

It allocates resources (CPU, memory, storage) from the **host machine** to the **guest machines**



Type 1 hypervisors
"native or bare-metal hypervisors"
Ex: KVM, Microsoft Hyper-V, VMware vSphere



Type 2 hypervisors
"hosted hypervisors"
Ex: VMware Workstation, Oracle VirtualBox

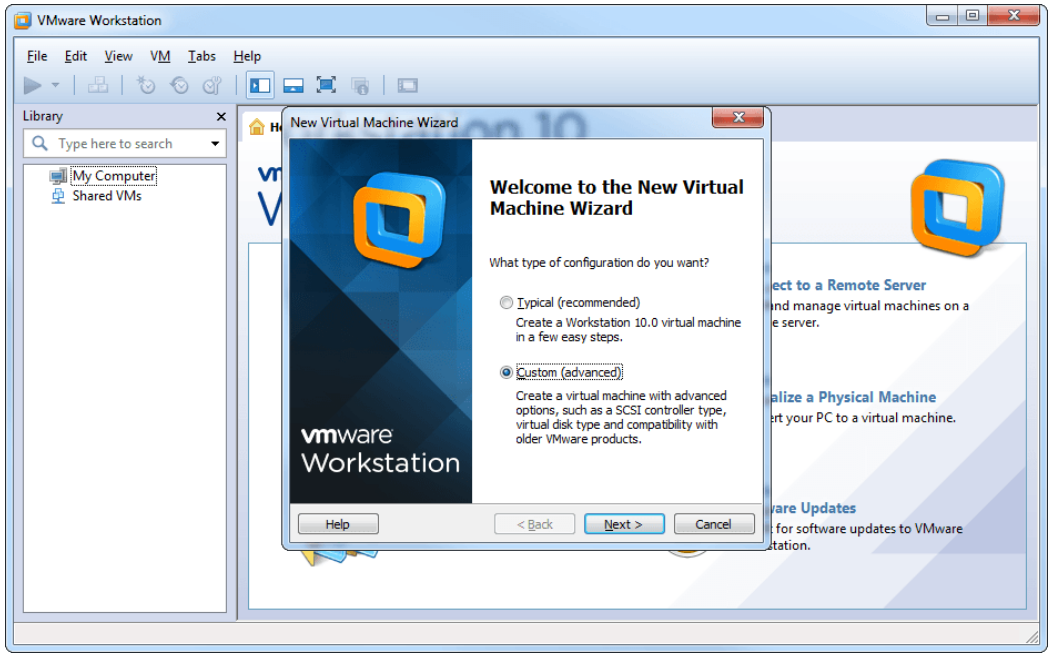
Virtualization

Pros:

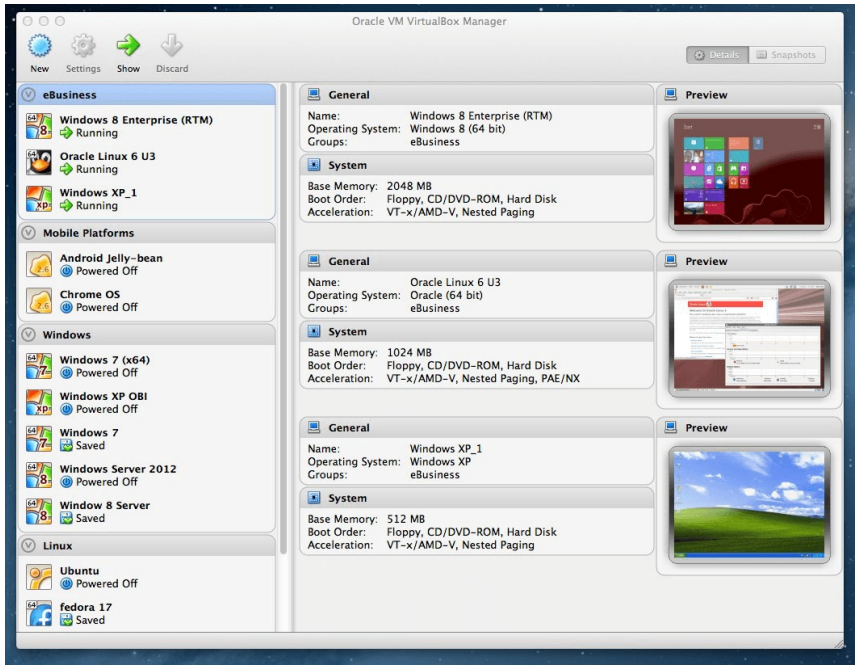
Less physical hardware: one machine for many OS
Central location to manage all asset
More eco-friendly: avoid idle machines
Secure isolation: isolating applications so that an ill-behaved application can't compromise other applications or its host.
Persistent compatibility: allowing host and application to evolve separately. Changes in the host don't break applications.
Execution continuity: A running application isn't tied to the computer on which it was started, but can be moved from computer to computer across space and time within a single run. (VMotion)

Cons:

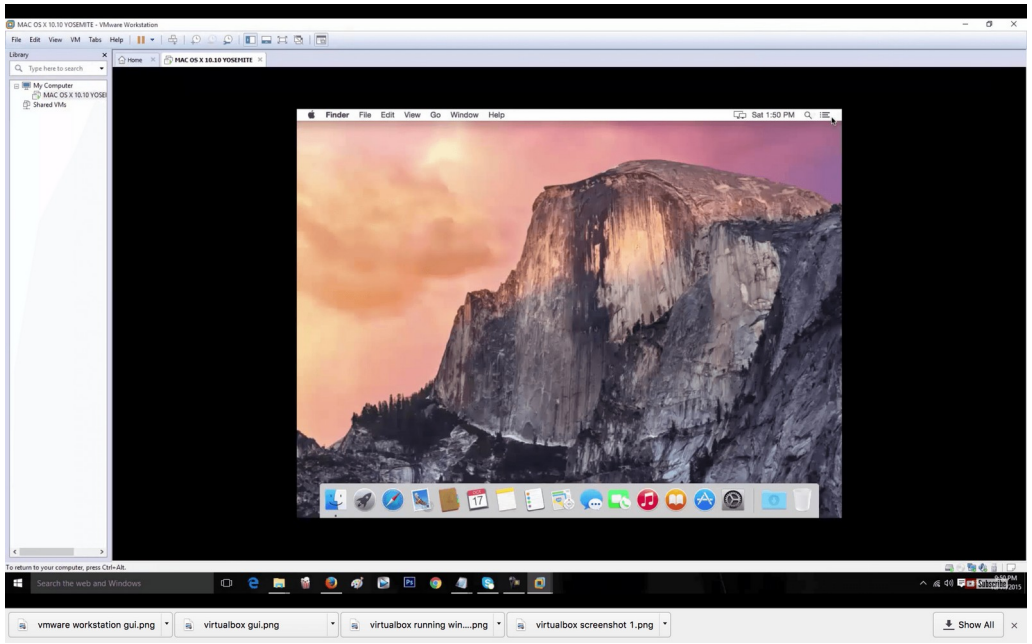
Resource overheads in terms of disk footprint, memory, CPU
Administrative costs
Single point of physical failure



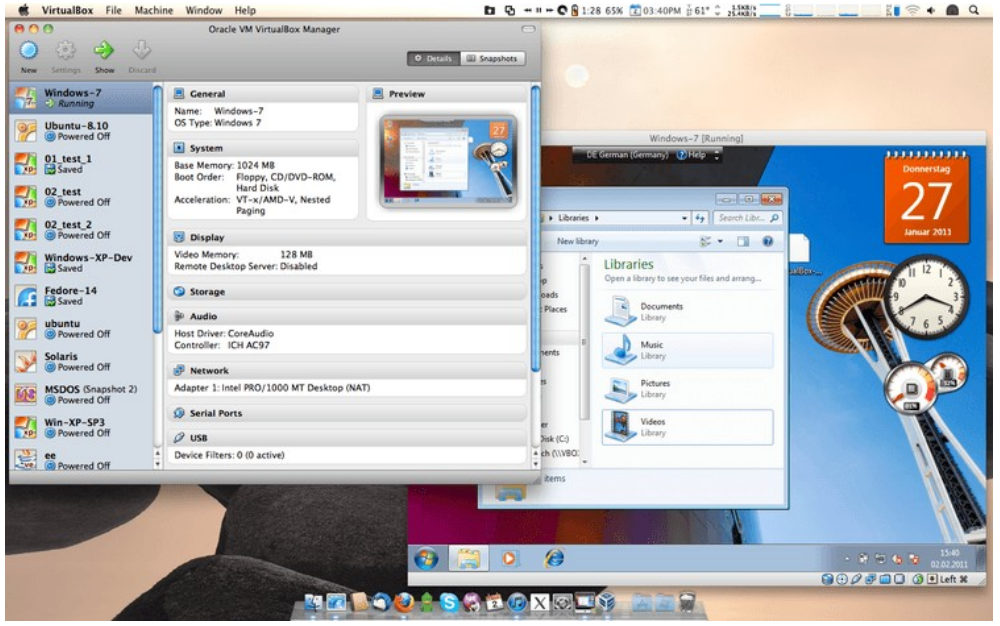
VMWare VM manager



VirtualBox VM manager



VMware Workstation running Mac OS X on a Windows 10 computer.



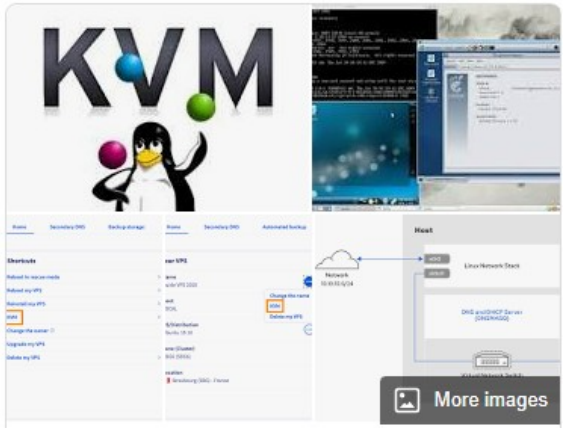
VirtualBox running Windows 7 on a Mac OS X computer.

Type 2 hypervisors:
VirtualBox Vs. VMware: Comparison Table

Comparison	VirtualBox	VMware
Software Virtualization	Yes	No
Hardware Virtualization	Yes	Yes
Host Operating Systems	Linux, Windows, Solaris, macOS, FreeBSD	Linux, Windows + macOS (requires VMware Fusion)
Guest Operating Systems	Linux, Windows, Solaris, macOS, FreeBSD	Linux, Windows, Solaris, FreeBSD + macOS (with VMware Fusion)
User Interface	Graphical User Interface (GLI) and Command Line Interface (CLI)	Graphical User Interface (GLI) and Command Line Interface (CLI)
Snapshots	Yes	Snapshots only supported on paid virtualization products, not on VMware Workstation Player
Virtual Disk Format	VDI, VMDK, VHD, HDD	VMDK
Virtual Disk Allocation Type	Preallocated: fixed disks; Dynamically allocated: dynamically allocated disks;	Preallocated: provisioned disks; Dynamically allocated: thin provisioned disks;
Virtual Network Models	Not attached, NAT, NAT Network, Bridged adapter, Internal network, Host-only adapter, Generic (UDP, VDE)	NAT, Bridged, Host-only + Virtual network editor (on VMware workstation and Fusion Pro)
USB Devices Support	USB 2.0/3.0 support requires the Extension Pack (free)	Out of the box USB device support
3D Graphics	Up to OpenGL 3.0 and Direct3D 9; Max of 128 MB of video memory; 3D acceleration enabled manually	Up to OpenGL 3.3, DirectX 10; Max of 2GB of video memory; 3D acceleration enabled by default
Integrations	VMDK, Microsoft's VHD, HDD, QED, Vagrant, Docker	Requires additional conversion utility for more VM types; VMware VSphere and Cloud Air (on VMware Workstation)
VirtualBox Guest Additions vs. VMware Tools	Installed with the VBoxGuestAdditions.iso file	Install with a .iso file used for the given VM (linux.iso, windows.iso, etc.)
API for Developers	API and SDK	Different APIs and SDKs
Cost and Licenses	Free, under the GNU General Public License	VMware Workstation Player is free, while other VMware products require a paid license

https://phoenixnap.com/kb/virtualbox-vs-vmware , Feb 2021

Type 1 hypervisors



Kernel-based Virtual Machine

Kernel-based Virtual Machine is a virtualization module in the Linux kernel that allows the kernel to function as a hypervisor. It was merged into the mainline Linux kernel in version 2.6.20, which was released on February 5, 2007. Wikipedia

Developer(s): The Linux Kernel community

Platform: ARM, IA-64, PowerPC, S/390, x86, x86-64

Operating system: Unix-like

License: GNU GPL or LGPL

Repository: git.kernel.org/pub/scm/virt/kvm/kvm.git

Original author(s): Qumranet

Written in: C

Hyper-V

Computer program

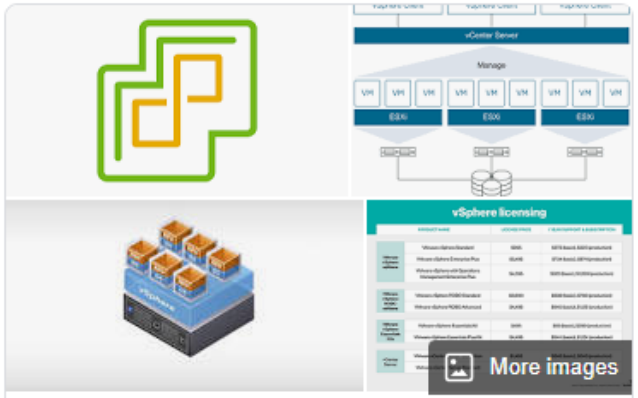
Microsoft Hyper-V, codenamed Viridian, and briefly known before its release as Windows Server Virtualization, is a native hypervisor; it can create virtual machines on x86-64 systems running Windows. Wikipedia

Operating system: Windows Server; Windows 8, Windows 8.1, Windows 10, Windows 11 (x64; Pro, Enterprise and Education)

Developer(s): Microsoft

Initial release: 2008; 14 years ago

Predecessor: Windows Virtual PC



VMware vSphere

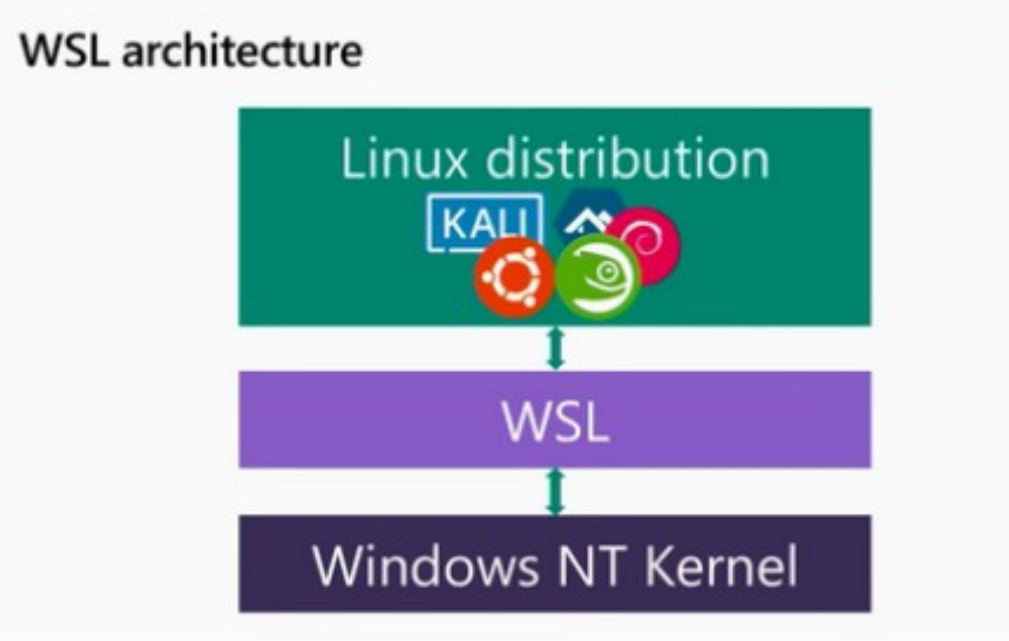
Computer application

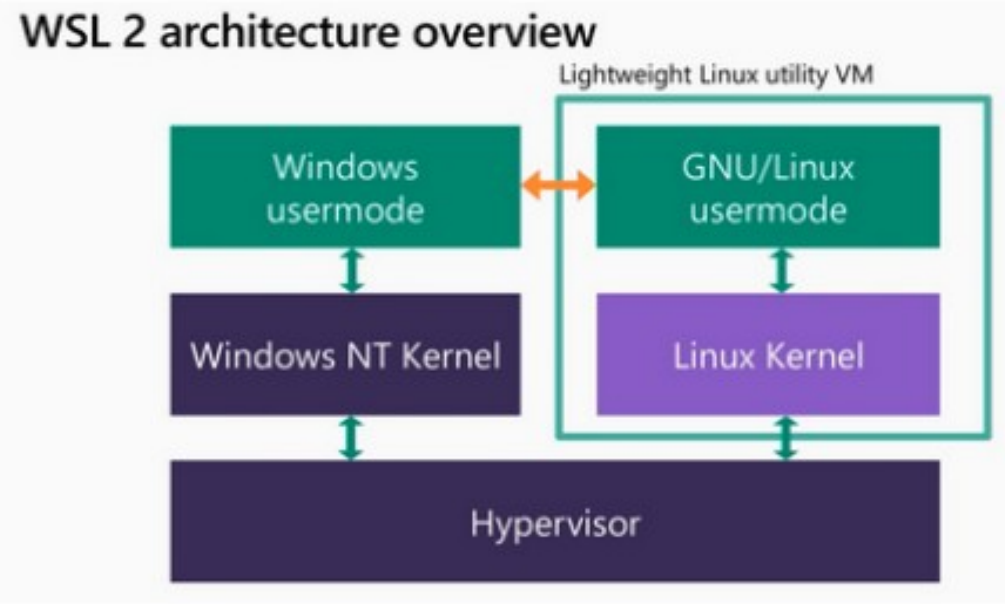
VMware vSphere is VMware's cloud computing virtualization platform. It includes an updated vCenter Configuration Manager, as well as vCenter Application Discovery Manager, and the ability of vMotion to move more than one virtual machine at a time from one host server to another. Wikipedia

License: Proprietary

+ other

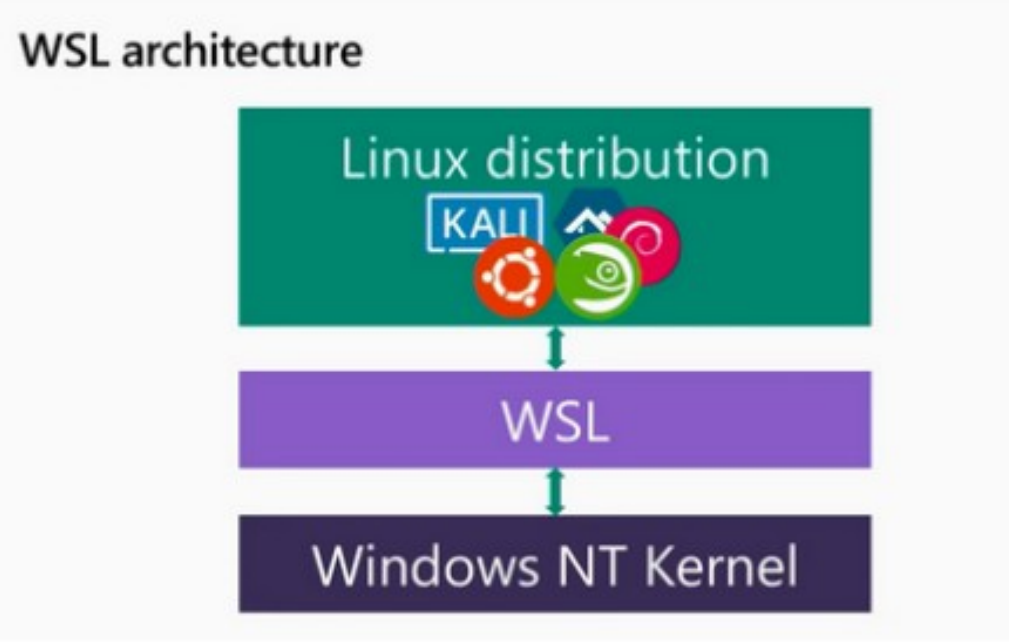
Ex: Windows Subsystem for Linux

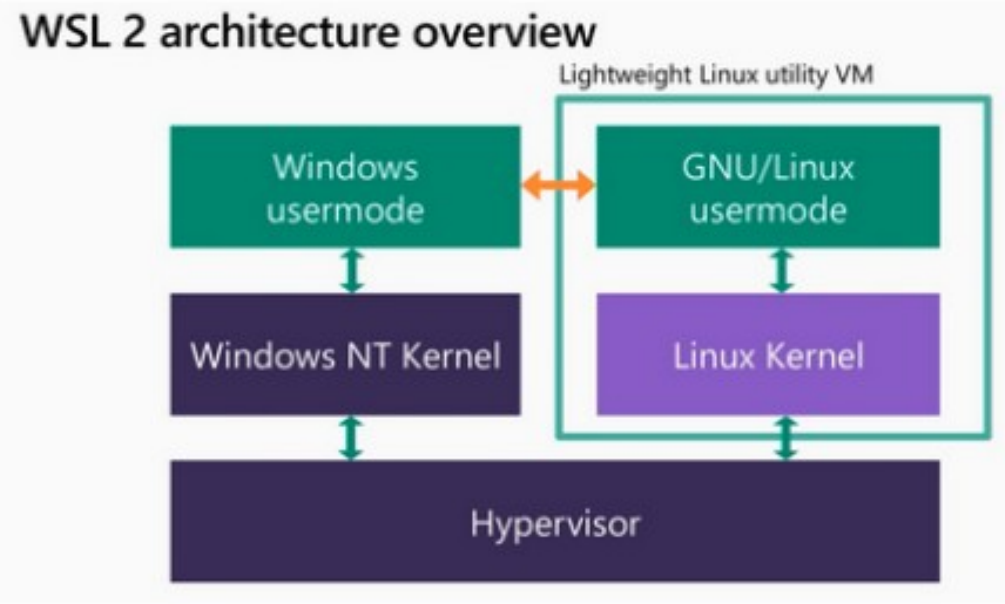




Type 1 ? Type 2 ?

Ex: Windows Subsystem for Linux





Type 2:
uses « picoprocess » hypervisors

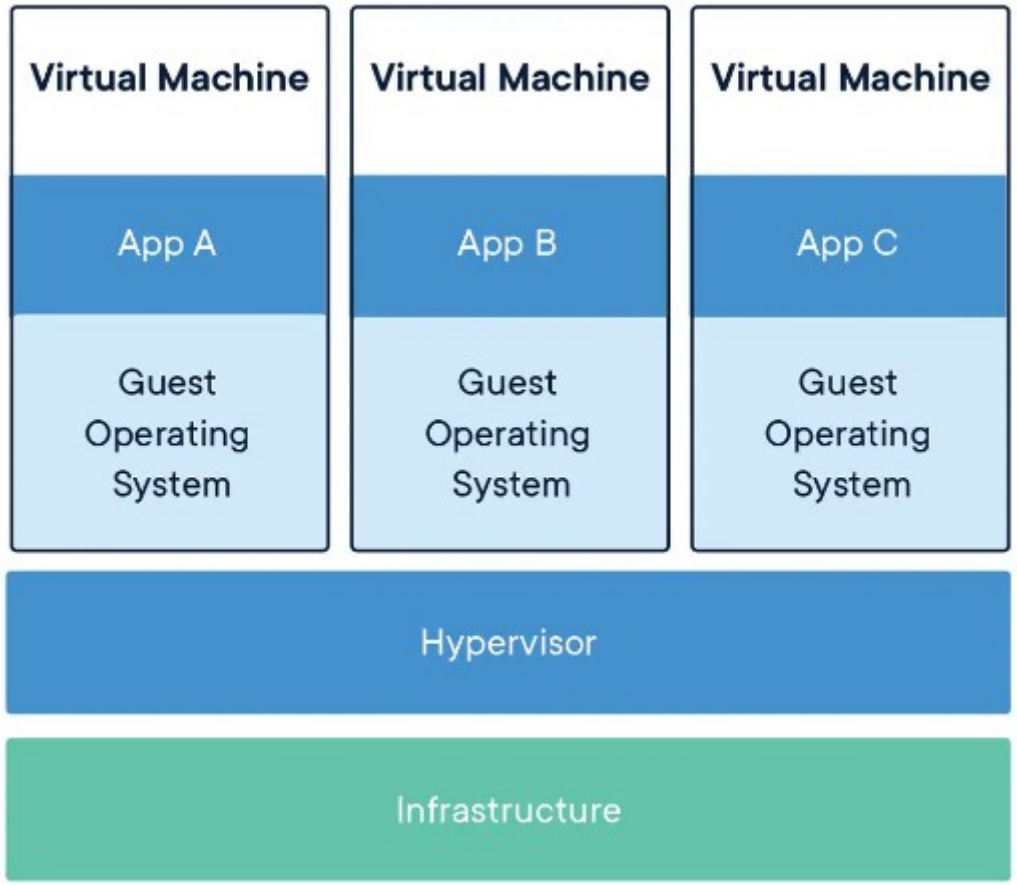
Type 1:
uses Microsoft Hyper-V hypervisor

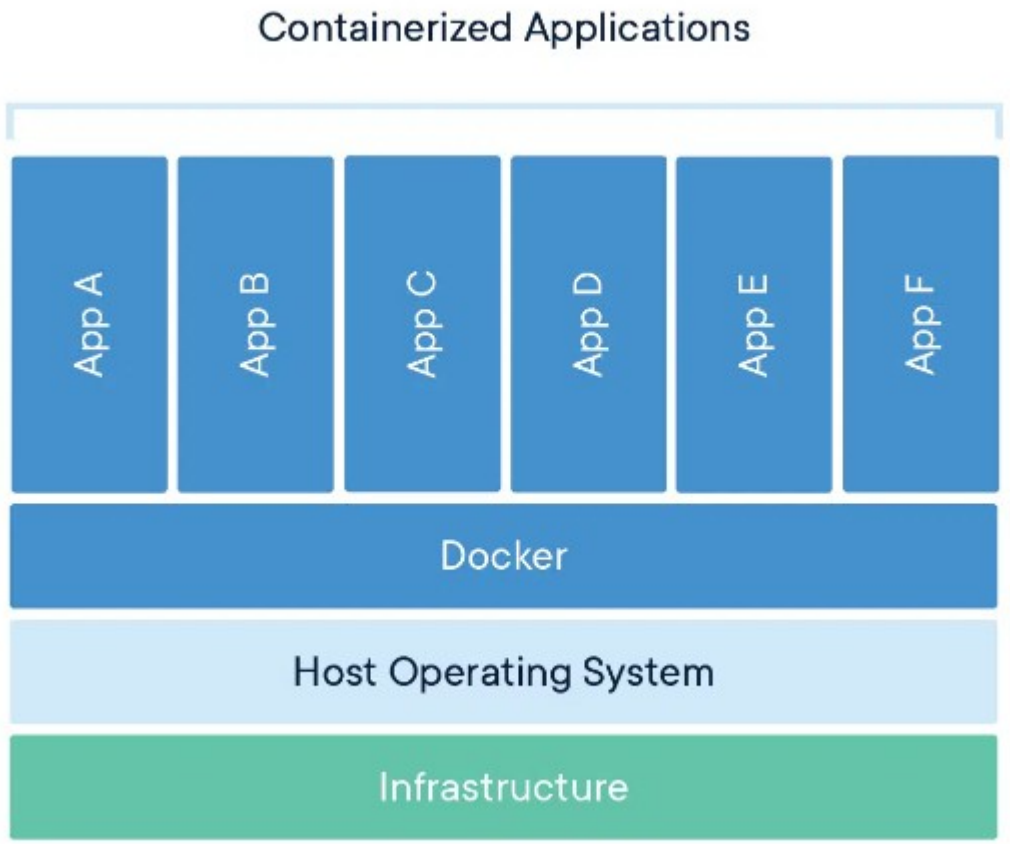
Ex: Windows Subsystem for Linux

Comparing features

Feature	WSL 1	WSL 2
Integration between Windows and Linux	✅	✅
Fast boot times	✅	✅
Small resource foot print	✅	✅
Runs with current versions of VMWare and VirtualBox	✅	✅
Managed VM	❌	✅
Full Linux Kernel	❌	✅
Full system call compatibility	❌	✅
Performance across OS file systems	✅	❌

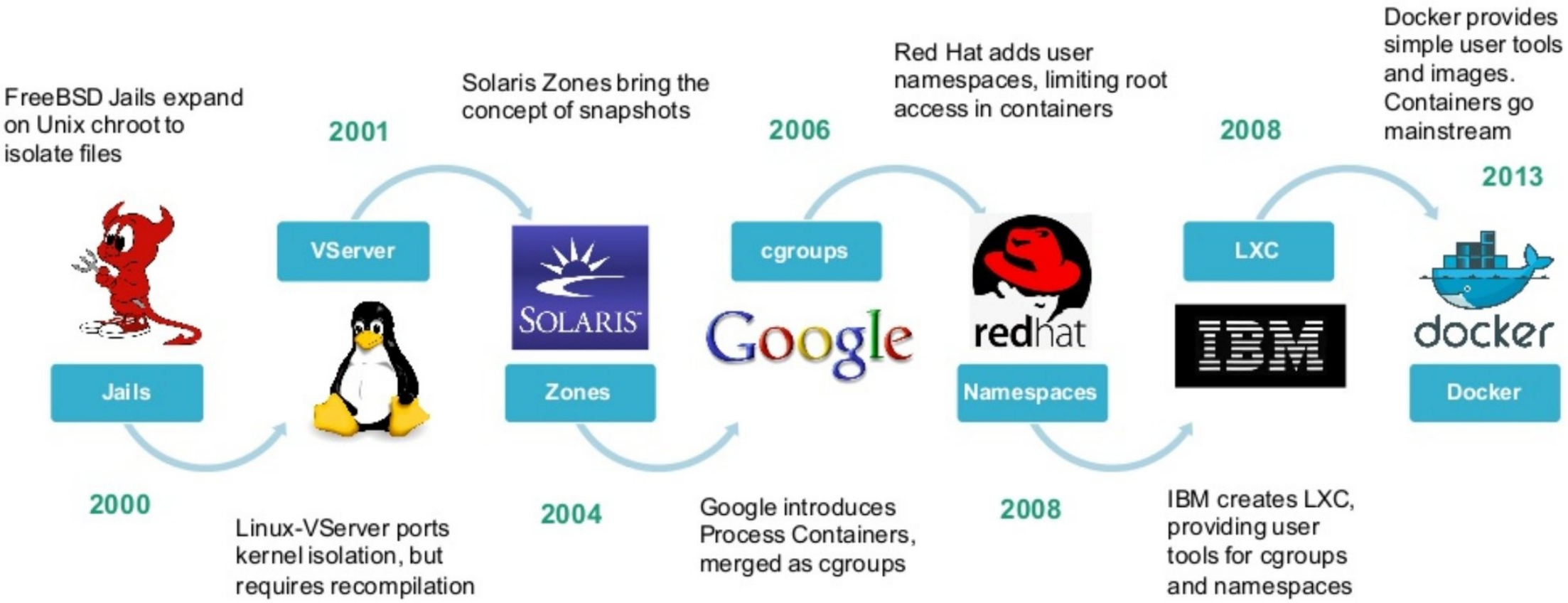
Virtual machines vs Containers





https://www.docker.com/resources/what-container/

History of containers



https://borosan.gitbook.io/docker-handbook/containerization-history

The evolution of containers

Container technology has come a long way from its chroots, starting with Google's exploration into cgroups and working up into widespread organizational adoption.

2004-2007

2004-07
Google uses containers (cgroups) at scale

2008

JANUARY 2008
cgroups merge into Linux kernel

AUGUST 2008
LXC released: Linux containers, based on cgroups, isolate resources without a VM

2013

MARCH 2013
Docker hits the scene: Containerization platform based on Linux; integrates natively with other OSes: Windows, macOS

OCTOBER 2013
CoreOS Linux (now Container Linux by CoreOS) released

2013 RESEARCH
30 containerization vendors vie for users (451 RESEARCH)

2014

JUNE 2014
Docker 1.0 is released

NOVEMBER 2014
Canonical launches LXD: Built on LXC with new features and functionality

JUNE 2014
Kubernetes project announced: A now widely popular open source container orchestrator and scheduler created by Google

DECEMBER 2014
CoreOS introduces rkt containers as Docker alternative

APRIL 2015
CoreOS first to provide commercial

JUNE 2015

CoreOS first to provide commercial Kubernetes offering

2015

JULY 2015

Kubernetes 1.0 released: Google gives Kubernetes to Cloud Native Computing Foundation (CNCF) for development

JUNE 2015

Open Container Project created AKA Open Container Initiative (OCI): Establishes common container standards

MARCH 2017

Pivotal integrates with Kubernetes and Cloud Foundry, dubbed Kubo; Docker donates containerd container runtime to CNCF

2017

APRIL 2017

Portworx opens door for big data with PX-Enterprise update; Microsoft enables orgs to run Linux containers on Windows

JULY 2017

Microsoft Azure Container Instances provides container management for Linux

SEPTEMBER 2017

Rancher adopts Kubernetes for container orchestration

JUNE 2017

Kubernetes adds stateful support: Stateless apps forget session information; stateful apps hold onto it

2017 RESEARCH

Less than 20% enterprise container adoption (GARTNER)

OCTOBER 2017

Cloud Foundry Container Runtime launches

2017 RESEARCH

Container market tops $1 billion, with 125 application container vendors (451 RESEARCH)

OCTOBER 2017

Docker adds native Kubernetes support alongside swarm mode

2018

APRIL 2018

Kubernetes adds API Aggregation, improves support for Windows nodes and Linux, audits API stability in 1.10 release

2018 RESEARCH

90% of infrastructure and operations groups report testing or deploying containers (GARTNER)

NOV 2018

VMware purchases Heptio for Kubernetes integration

JUNE 2019

Amazon EKS and Microsoft AKS become generally available

CoreOS first to provide commercial Kubernetes offering

2015

JULY 2015

Kubernetes 1.0 released: Google gives Kubernetes to Cloud Native Computing Foundation (CNCF) for development

JUNE 2015

Open Container Project created AKA Open Container Initiative (OCI): Establishes common container standards

MARCH 2017

Pivotal integrates with Kubernetes and Cloud Foundry, dubbed Kubo; Docker donates containerd container runtime to CNCF

APRIL 2017

Portworx opens door for big data with PX-Enterprise update; Microsoft enables orgs to run Linux containers on Windows

2017

JUNE 2017

Kubernetes adds stateful support: Stateless apps forget session information; stateful apps hold onto it

JULY 2017

Microsoft Azure Container Instances provides container management for Linux

2017 RESEARCH

Less than 20% enterprise container adoption (GARTNER)

SEPTEMBER 2017

Rancher adopts Kubernetes for container orchestration

OCTOBER 2017

Cloud Foundry Container Runtime launches

2017 RESEARCH

Container market tops $1 billion, with 125 application container vendors (451 RESEARCH)

OCTOBER 2017

Docker adds native Kubernetes support alongside swarm mode

2018

APRIL 2018

Kubernetes adds API Aggregation, improves support for Windows nodes and Linux, audits API stability in 1.10 release

NOV 2018

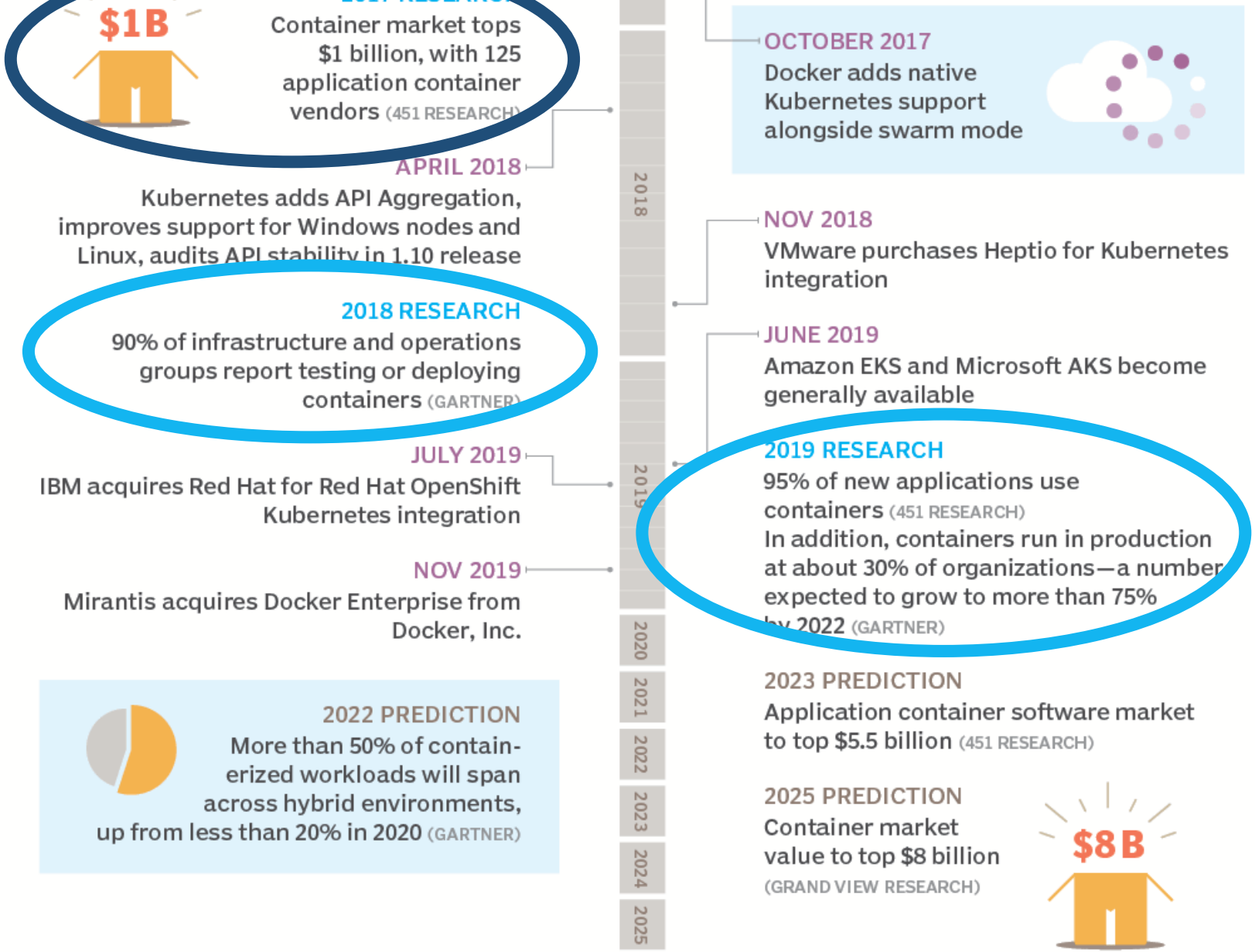
VMware purchases Heptio for Kubernetes integration

2018 RESEARCH

90% of infrastructure and operations groups report testing or deploying containers (GARTNER)

JUNE 2019

Amazon EKS and Microsoft AKS become generally available



SOURCE: EMILY MELL FOR TECHTARGET; DESIGN: LINDA KOURY; LAPTOP ART: LIGHTCOME/GETTY IMAGES

Technological foundations of software development

Run your software anywhere with Docker

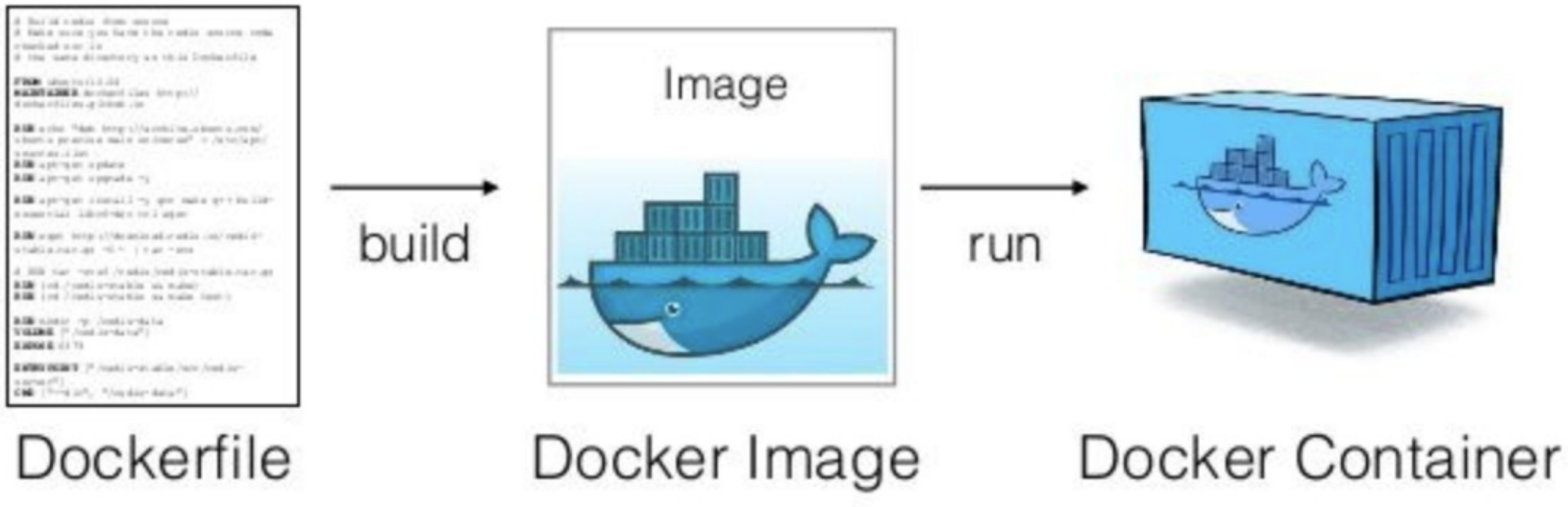
Part 2: Docker

ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Docker containers in 100 seconds

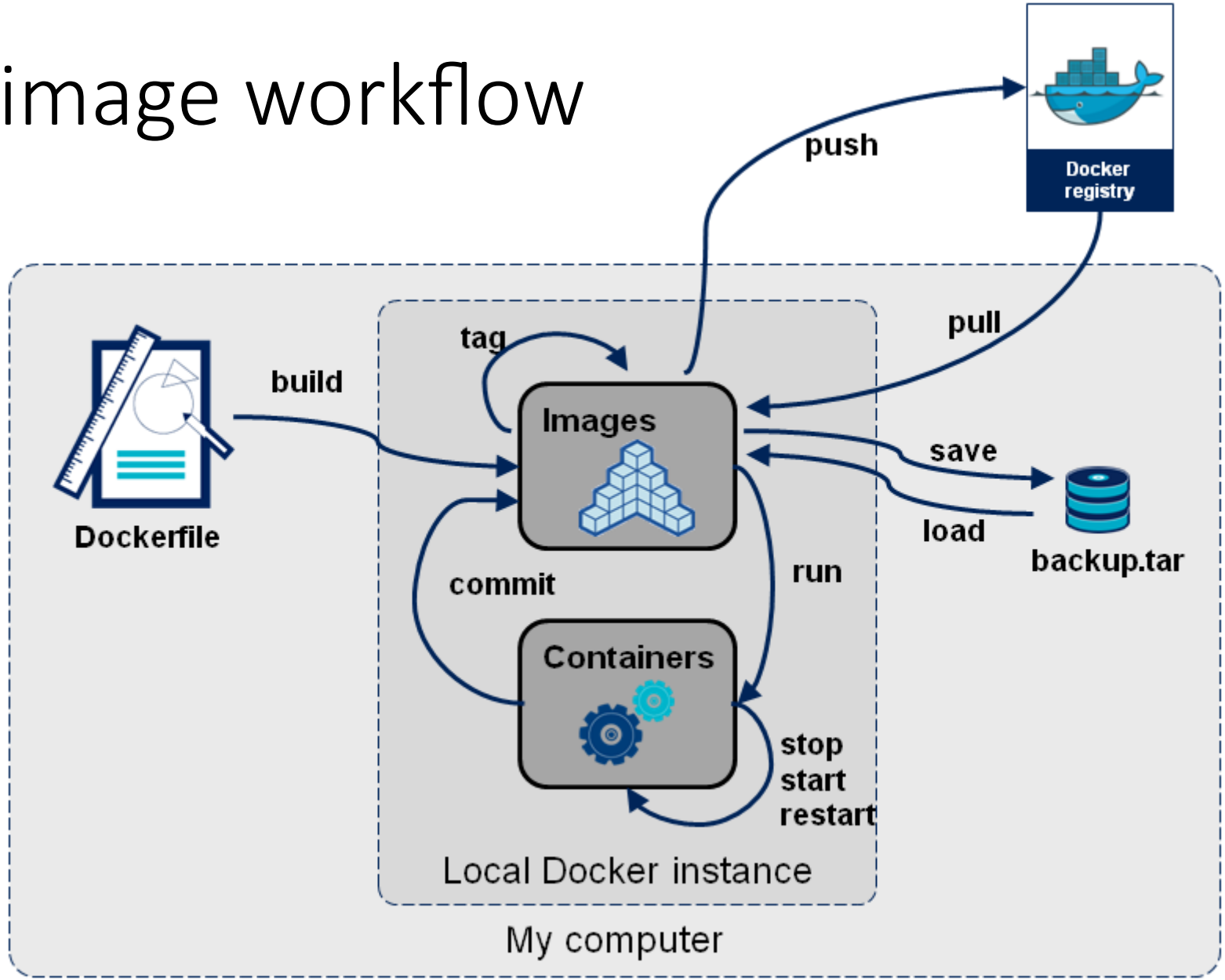
Dockerfile, Image, and Container

- Docker is a platform to build images and run containers. **There exists other tools !**
- Dockerfile is a recipe guiding how to build the image
- A container is a running instance of an image



https://medium.com/platformer-blog/practical-guide-on-writing-a-dockerfile-for-your-application-89376f88b3b5

Docker image workflow



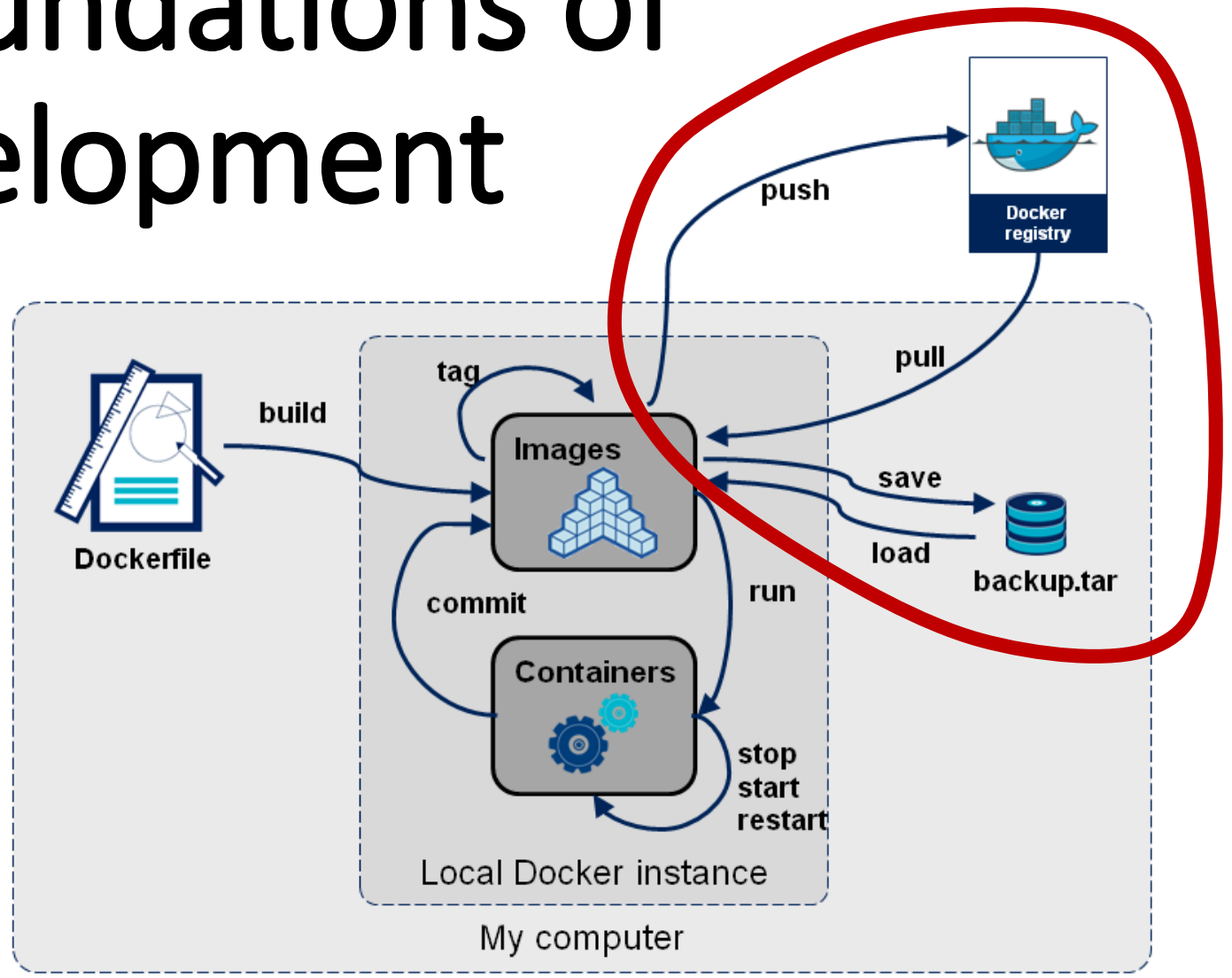
https://learn.gencore.bio.nyu.edu/containerization-with-docker/

Technological foundations of software development

Run your software anywhere with Docker

Part 2: Docker

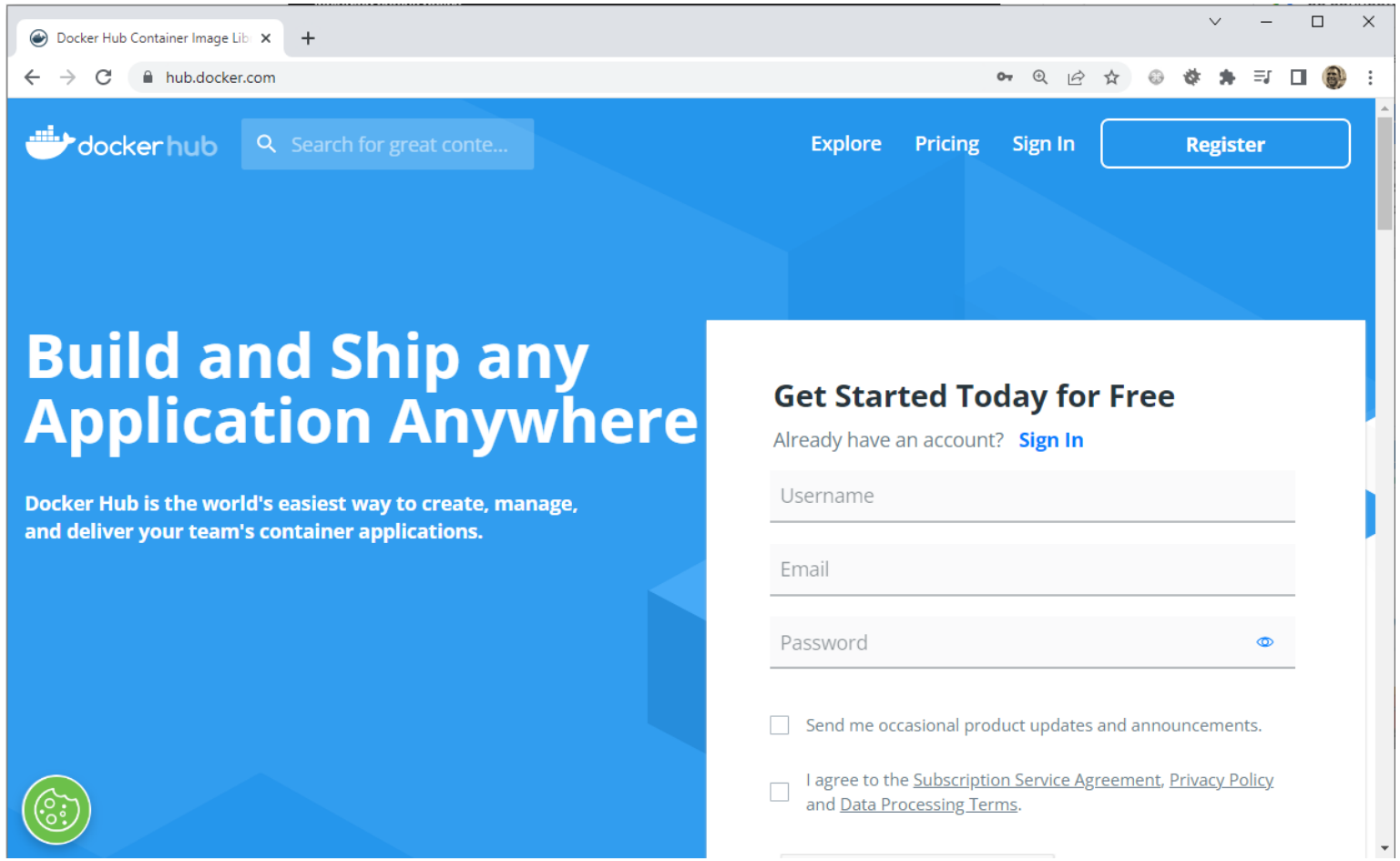
Part 2.1: Image Transfer



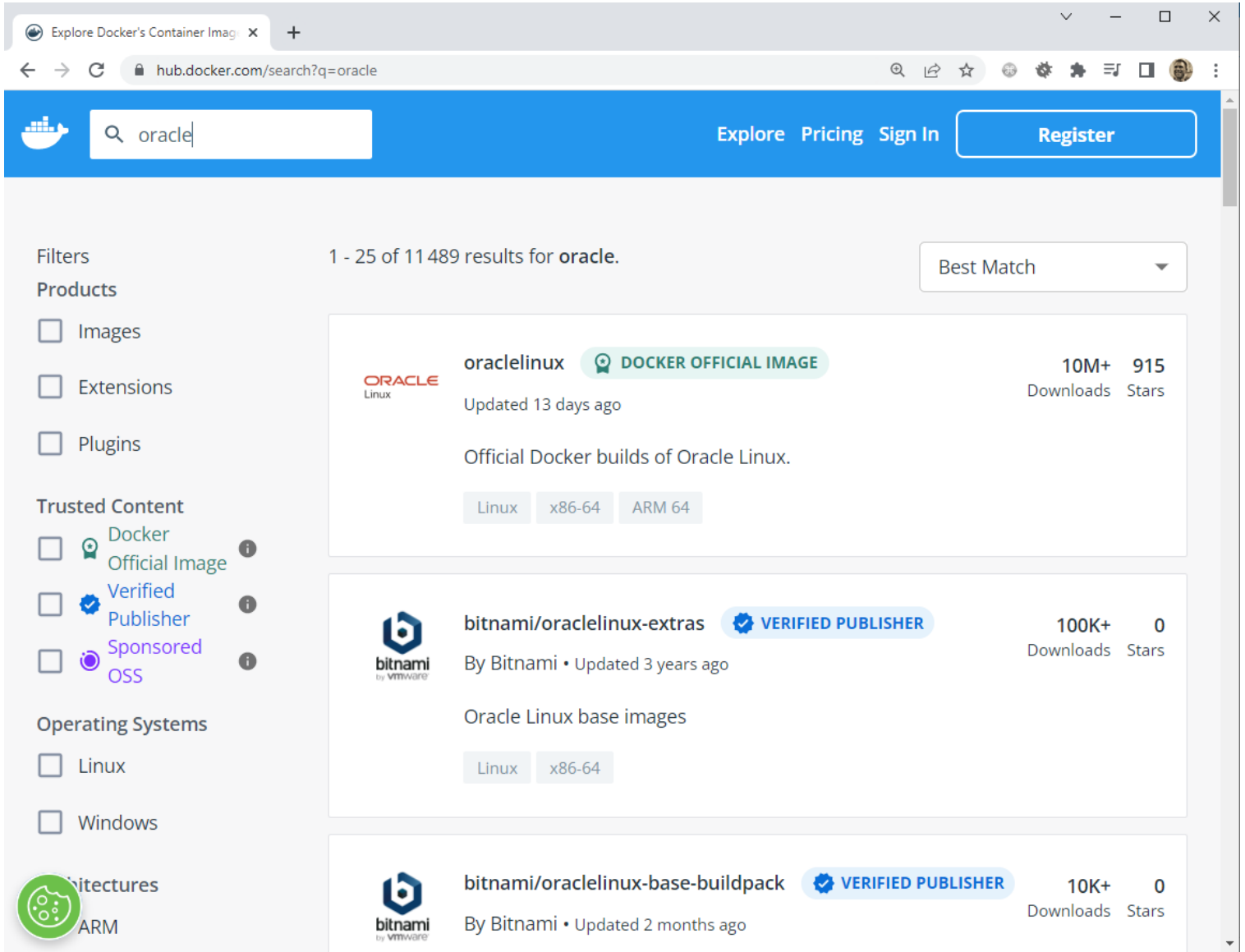
ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Find images on Docker Hub

URL : https://hub.docker.com/



Find images on Docker Hub



DOCKER OFFICIAL IMAGE

Docker Official Images are a curated set of Docker open source and "drop-in" solution repositories.

VERIFIED PUBLISHER

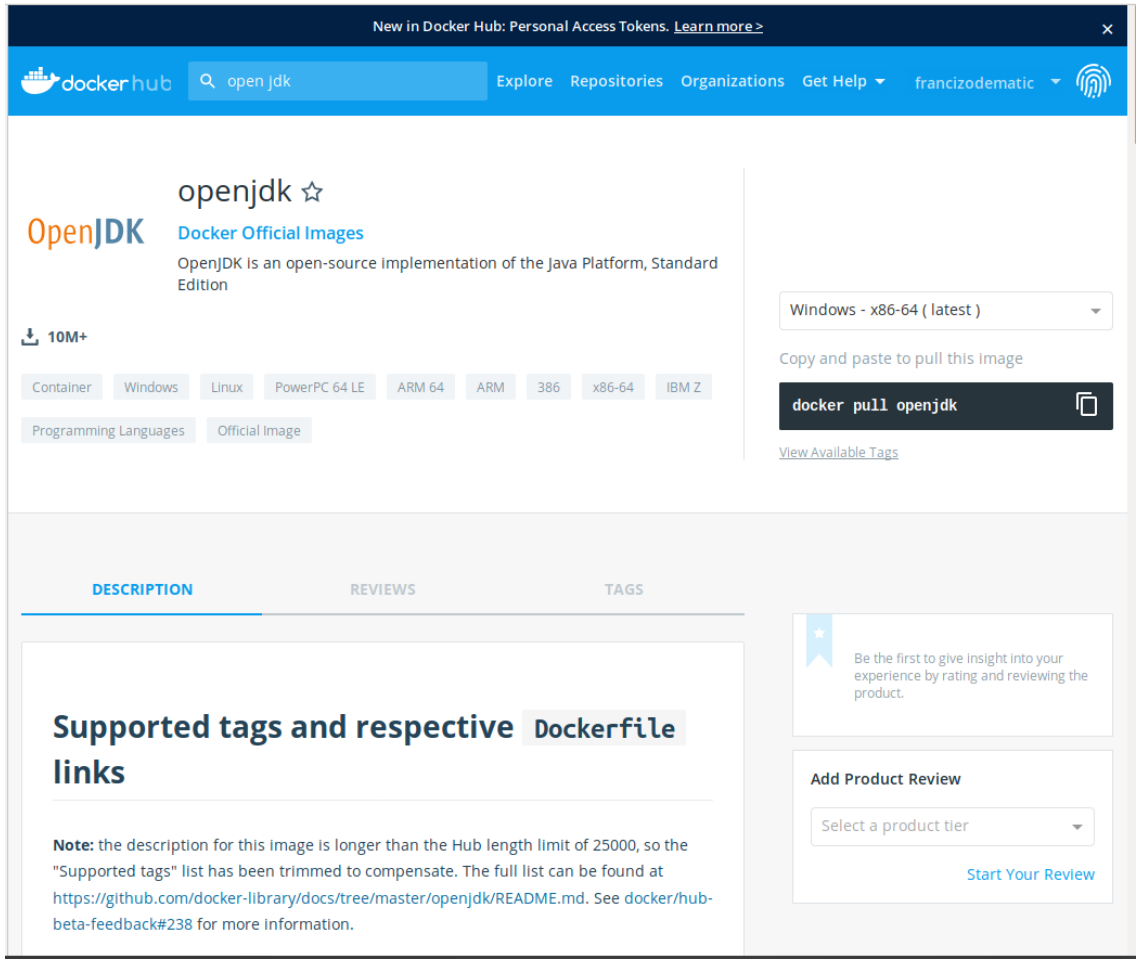
High-quality images from publishers verified by Docker. These products are published and maintained directly by a commercial entity. These images are not subject to rate limiting.

SPONSORED OSS

Docker Sponsored Open Source Software. These are images published and maintained by open-source projects that are sponsored by Docker through our open source program.

Pull an image

```
$ docker pull <name of the image in Docker Hub>
```



Pull an image

```
$ docker pull openjdk
```

```
francizo@bionic64bios:~$ docker pull openjdk
Using default tag: latest
latest: Pulling from library/openjdk
a316717fc6ee: Downloading [================>                                  ]  13.98MB/42.61MB
809137453b07: Downloading [==================================>                ]  10.32MB/14.77MB
a316717fc6ee: Downloading [=================>                                 ]  14.85MB/42.61MB
a316717fc6ee: Pull complete
809137453b07: Pull complete
b363ad12fddb: Pull complete
Digest: sha256:3d78bbf6043f085db058560493fc77236dde10a4e3b0533215278ca38d0bf42f
Status: Downloaded newer image for openjdk:latest
docker.io/library/openjdk:latest
francizo@bionic64bios:~$ docker pull openjdk
Using default tag: latest
latest: Pulling from library/openjdk
Digest: sha256:3d78bbf6043f085db058560493fc77236dde10a4e3b0533215278ca38d0bf42f
Status: Image is up to date for openjdk:latest
docker.io/library/openjdk:latest
francizo@bionic64bios:~$
```

Image transfer commands

Using the registry API

`docker pull` *repo*[:*tag*]...	pull an image/repo from a registry
`docker push` *repo*[:*tag*]...	push an image/repo from a registry
`docker search` *text*	search an image on the official registry
`docker login` ...	login to a registry
`docker logout` ...	logout from a registry

Manual transfer

`docker save` *repo*[:*tag*]...	export an image/repo as a tarbal
`docker load`	load images from a tarball
`docker-ssh`[10] ...	proposed script to transfer images between two daemons over ssh

https://dockerlabs.collabnix.com/docker/cheatsheet/

Other container registries

(GA 2013)

(GA 2015)

Amazon Elastic Container Registry (ECR)
GA 2015

(GA 2017)

By VMware, (GA 2014)

(GA 2019)

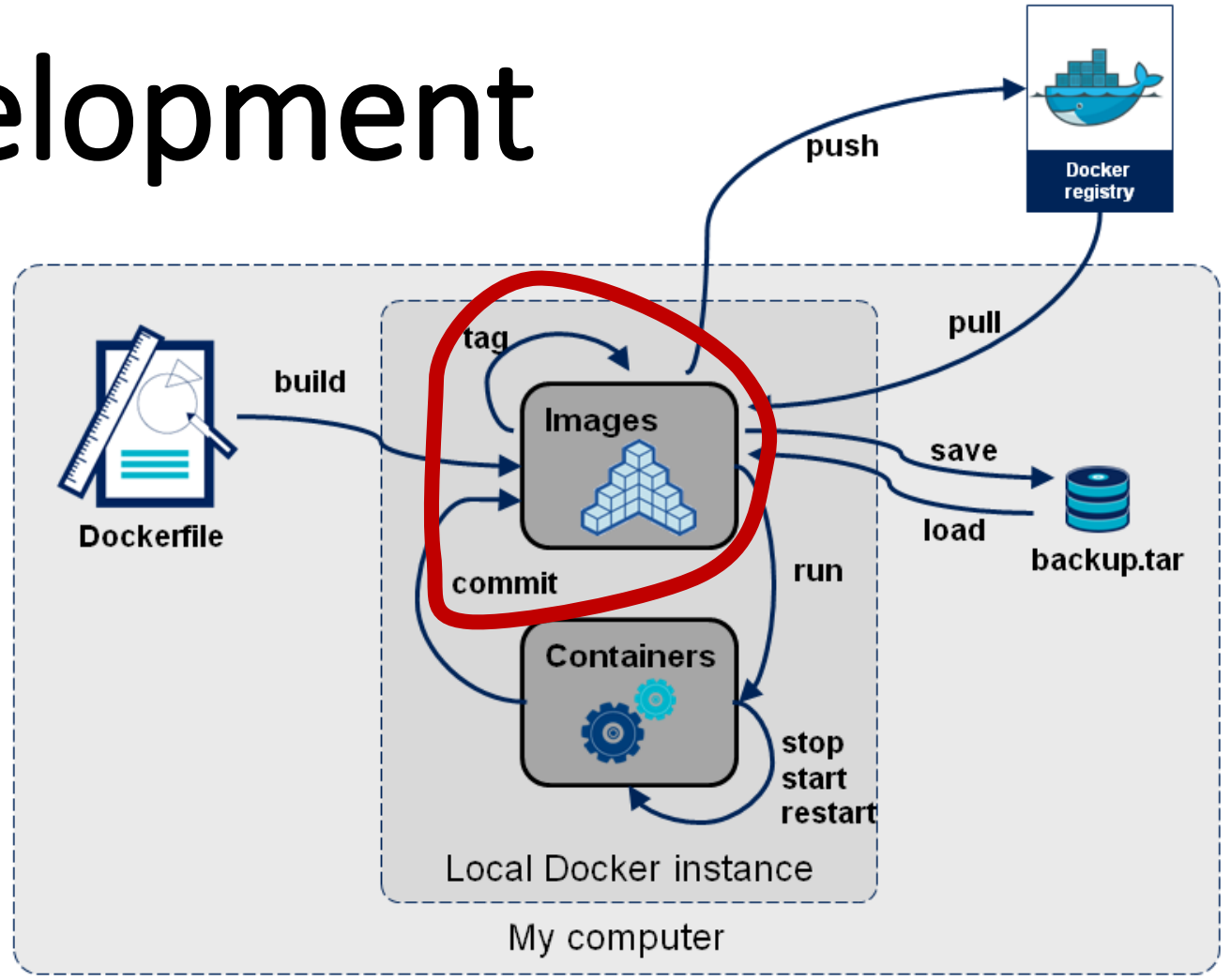
Open source

Technological foundations of software development

Run your software anywhere with Docker

Part 2: Docker

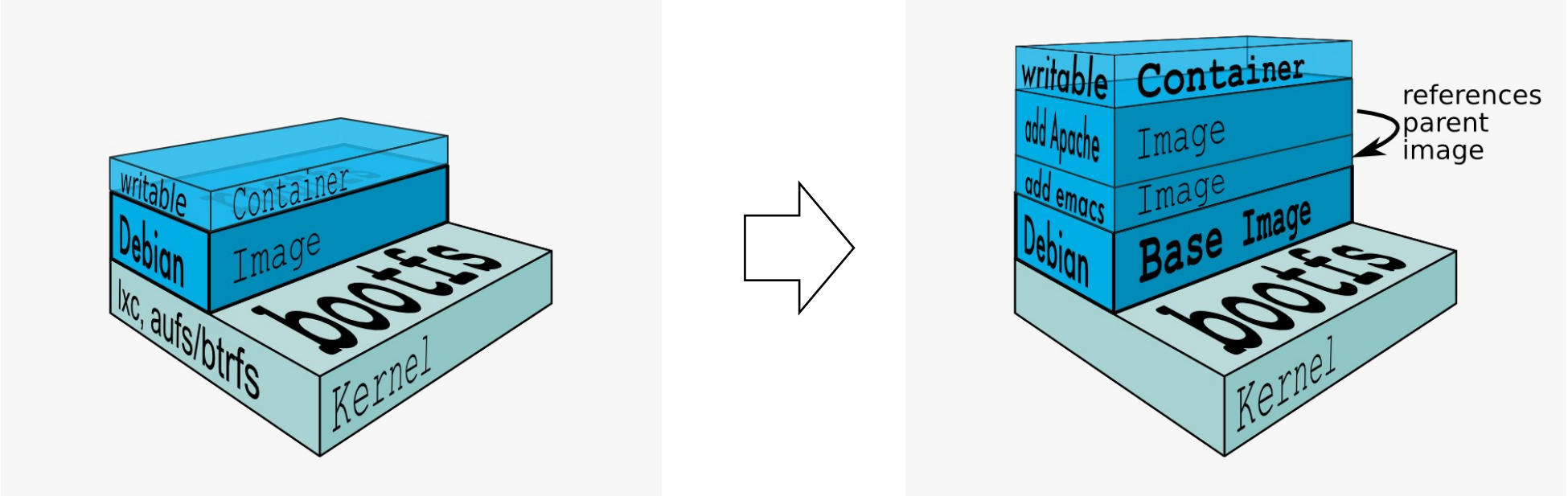
Part 2.2: Images anatomy and management



ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Anatomy of images

- Docker images are the basis of containers.
- An image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime.
- An image typically contains a union of layered filesystems stacked on top of each other.
- An image does not have state and it never changes.



Containers use **union file systems**

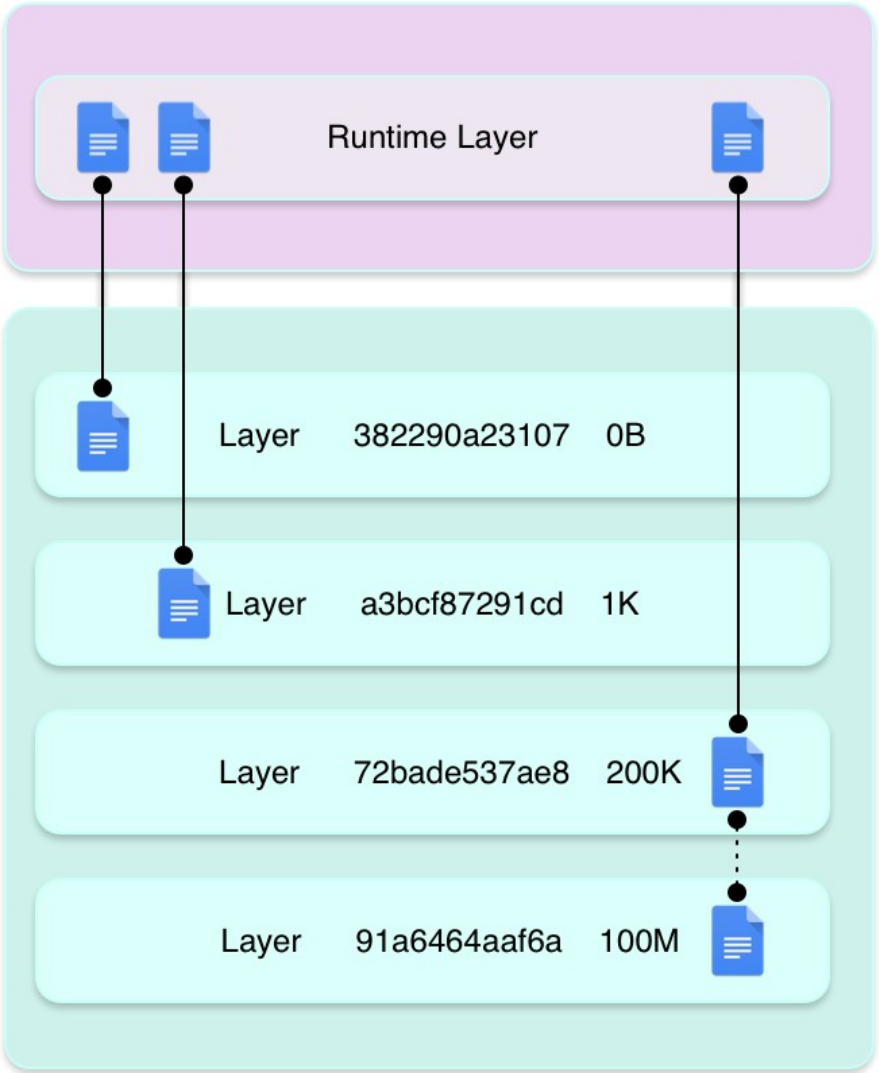
Docker uses a copy-on-write technique and a union file system for both images and containers to optimize resources and speed performance.

Copies of an entity share the same instance and each one makes only specific changes to its unique layer.

Multiple containers can share access to the same image, and make container-specific changes on a writable layer which is deleted when the container is removed. This speeds up container start times and performance.

Images are essentially layers of filesystems typically predicated on a base image under a writable layer, and built up with layers of differences from the base image. This minimizes the footprint of the image and enables shared development.

Different union file systems implementations:
unionFS, overlay, overlay2, aufs, zfs,…



https://docs.docker.com/glossary/#copy-on-write

Image identifiers

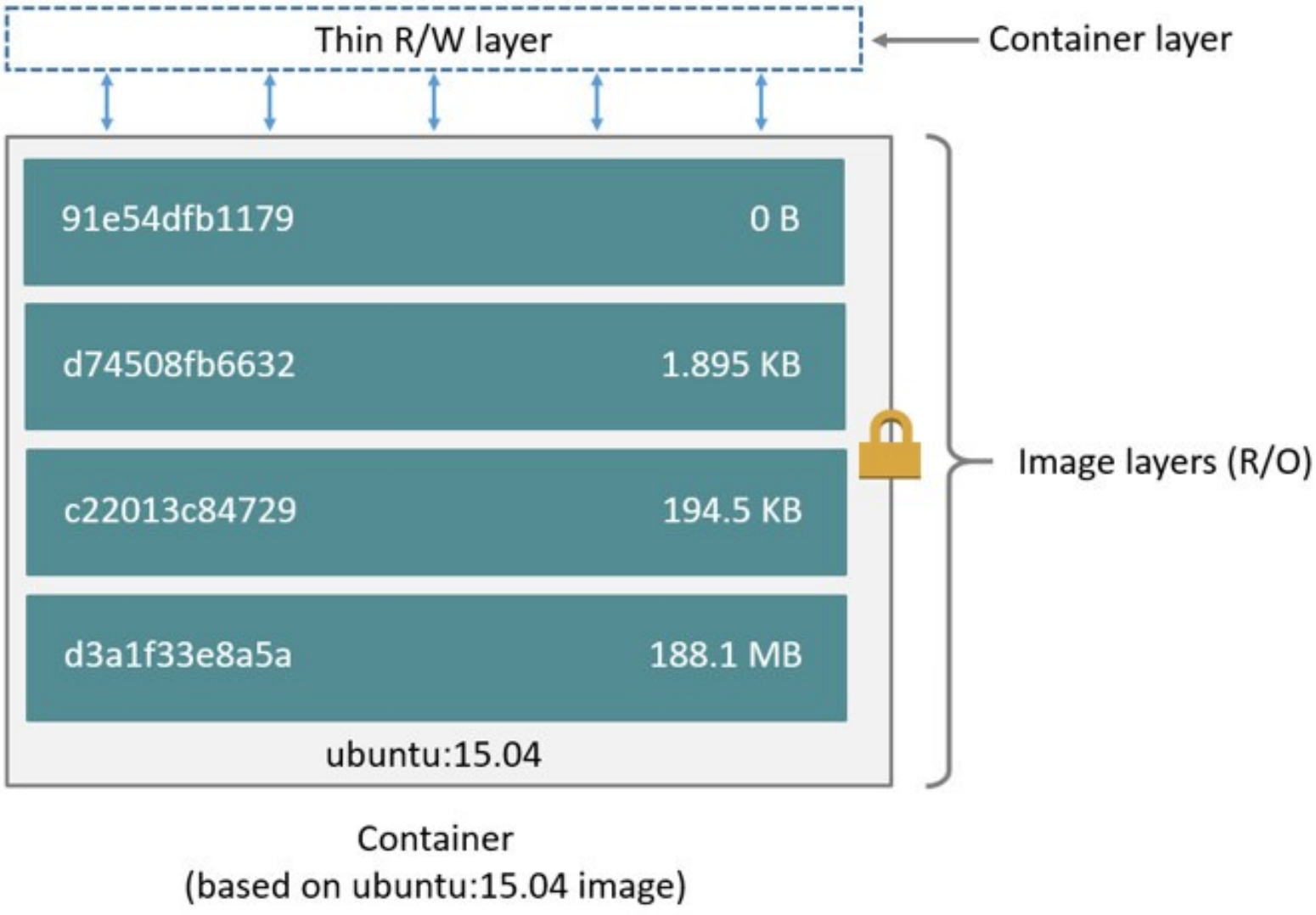
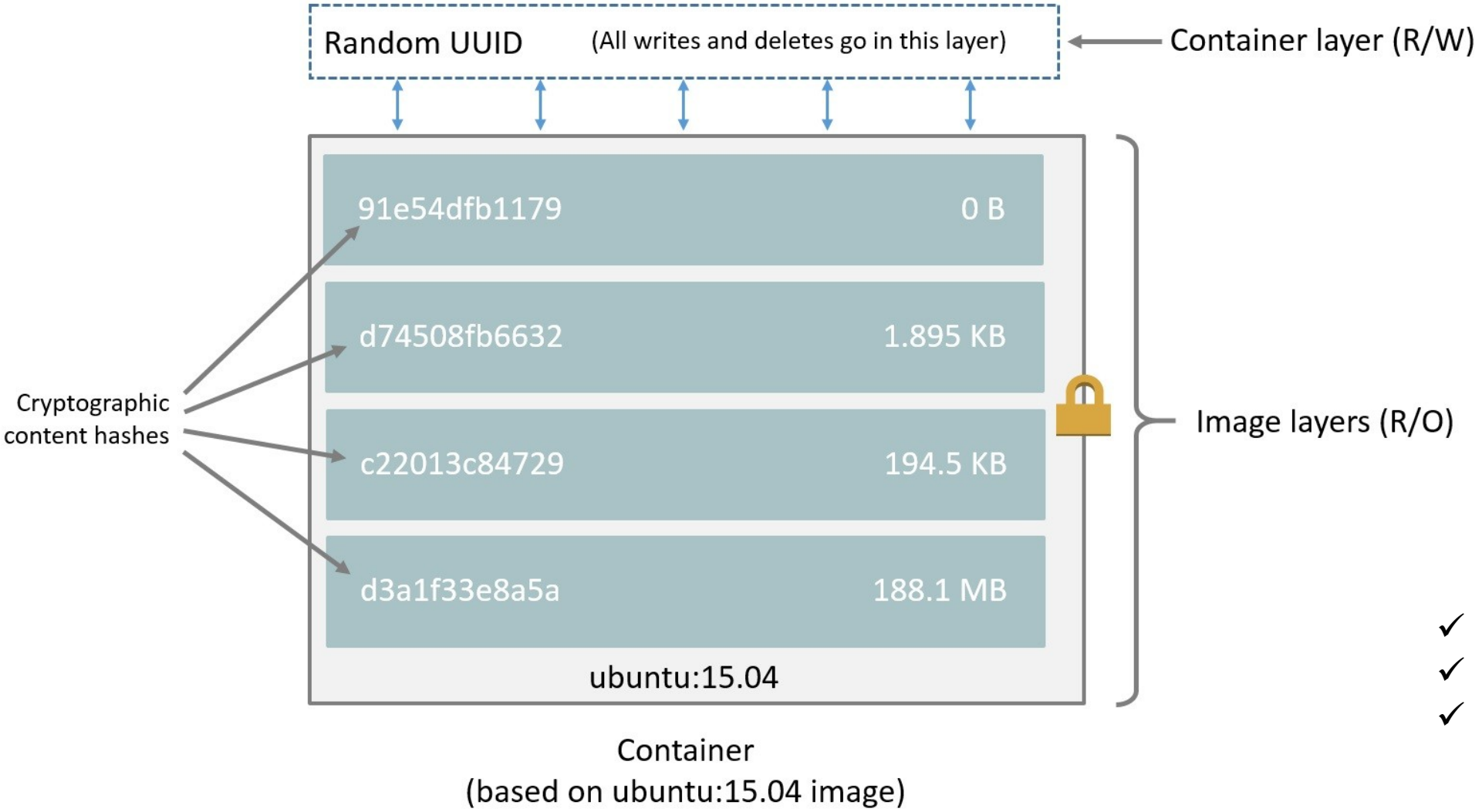


Image identifiers (since Docker 1.10)



- ✓ Avoids ID collisions
- ✓ Guarantees data integrity
- ✓ Easier to share images

Image management commands

command	description
`docker images`	list all local images
`docker history` *image*	show the image history (list of ancestors)
`docker inspect` *image...*	show low-level infos (in json format)
`docker tag` *image tag*	tag an image
`docker commit` *container image*	create an image (from a container)
`docker import` *url*|- [*tag*]	create an image (from a tarball)
`docker rmi` *image...*	delete images

https://dockerlabs.collabnix.com/docker/cheatsheet/

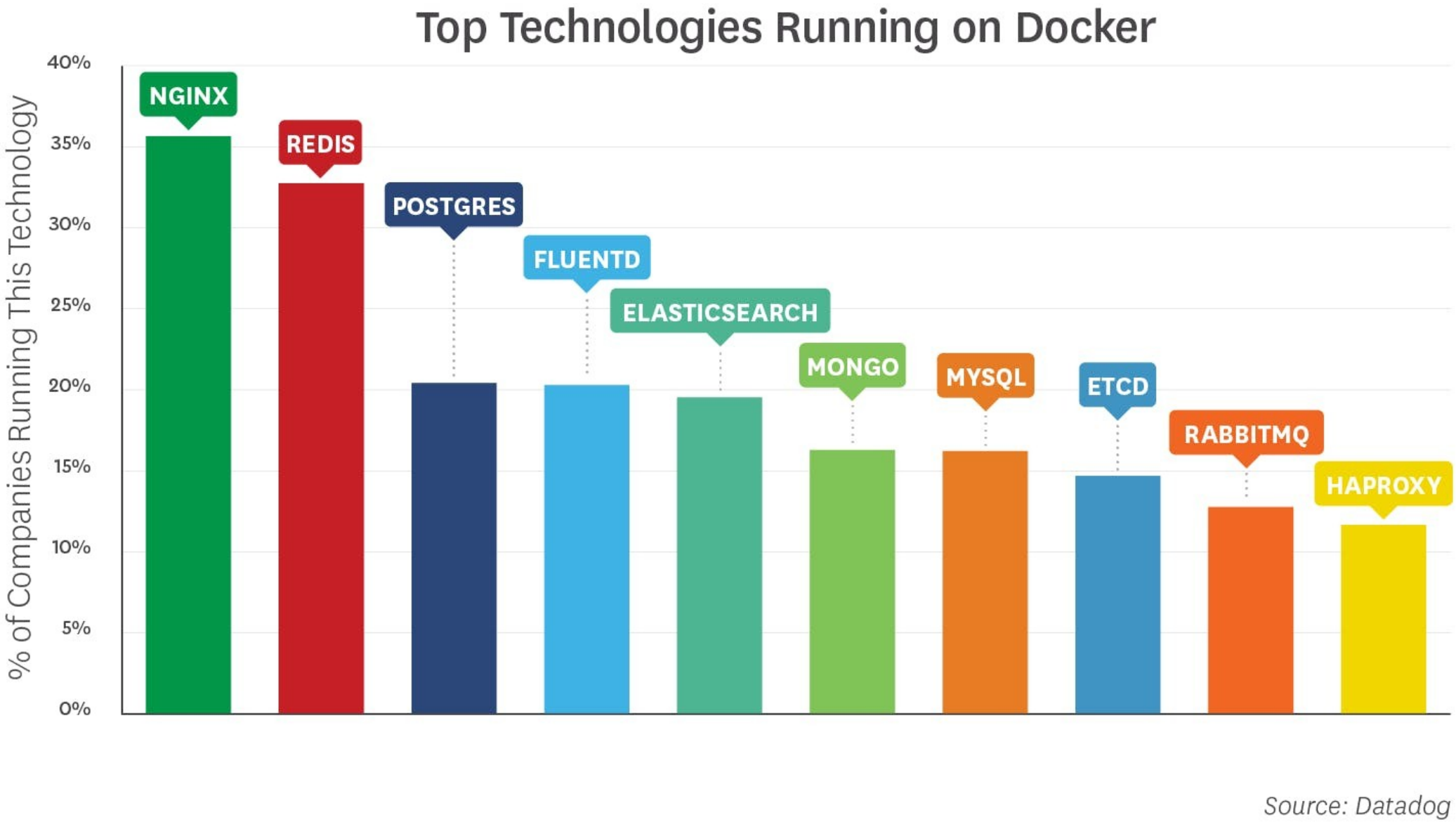
Example: manage images

`$ docker image ls` or `$ docker images`

```
francizo@bionic64bios:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
postgres            latest              e2d75d1c1264        2 weeks ago         313MB
openjdk             latest              b255bbd4a82d        4 weeks ago         490MB
oracle/database     12.2.0.1-ee         912bff258ebe        4 months ago        6.11GB
oraclelinux         7-slim              f7512ac13c1b        5 months ago        118MB
hello-world         latest              fce289e99eb9        9 months ago        1.84kB
```

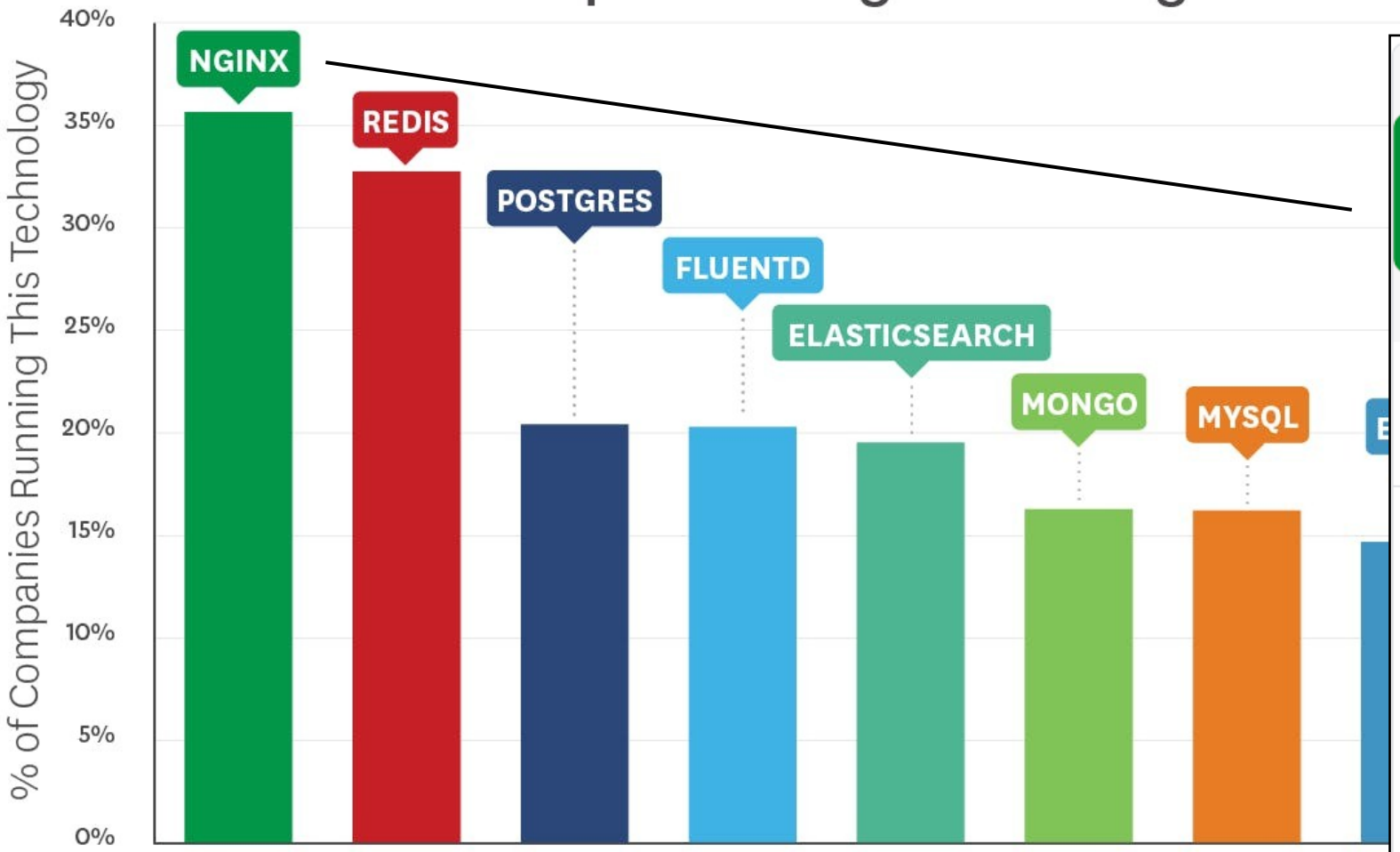
`$ docker image rm <name|ID>` or `$ docker rmi <name|ID>`

```
francizo@bionic64bios:~$ docker image rm "hello-world"
Untagged: hello-world:latest
Untagged: hello-world@sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560f2a5301c00
Deleted: sha256:fce289e99eb9bca977dae136fbe2a82b6b7d4c372474c9235adc1741675f587e
Deleted: sha256:af0b15c8625bb1938f1d7b17081031f649fd14e6b233688eea3c5483994a66a3
francizo@bionic64bios:~$
francizo@bionic64bios:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
postgres            latest              e2d75d1c1264        2 weeks ago         313MB
openjdk             latest              b255bbd4a82d        4 weeks ago         490MB
oracle/database     12.2.0.1-ee         912bff258ebe        4 months ago        6.11GB
oraclelinux         7-slim              f7512ac13c1b        5 months ago        118MB
```



Source: Datadog

Top Technologies Running on Docker



Nginx

Software

Nginx is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and publicly released in 2004. Nginx is free and open-source software, released under the terms of the 2-clause BSD license. Wikipedia

License: BSD-2-Clause

Developer(s): F5, Inc

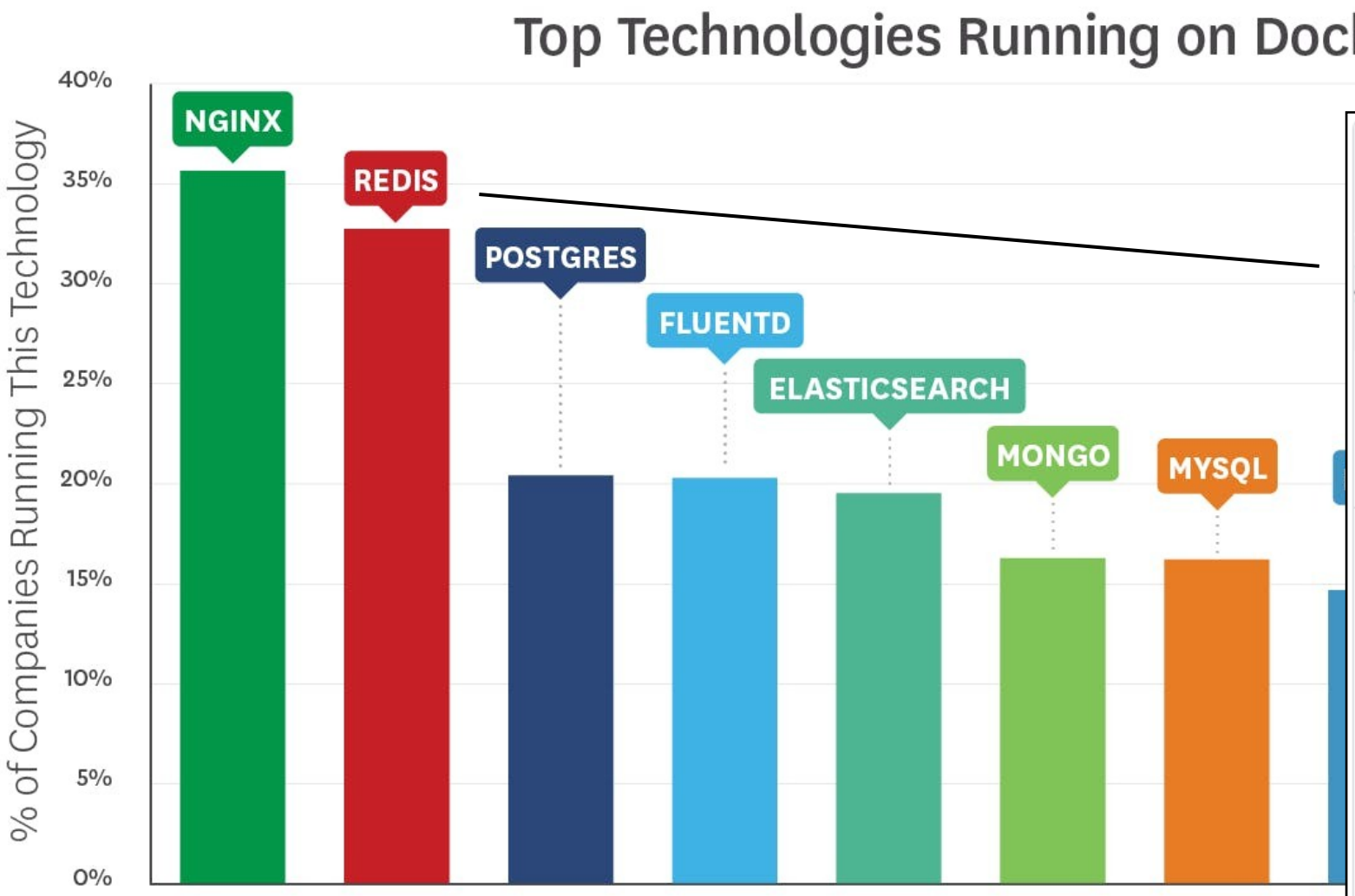
Written in: C

Initial release: 4 October 2004; 17 years ago

Operating system: BSD variants, HP-UX, IBM AIX, Linux, macOS, Solaris, Microsoft Windows, and other *nix flavors

Original author(s): Igor Sysoev

Preview release: 1.23.1 / 19 July 2022



Redis

Software

Redis is an in-memory data structure store, used as a distributed, in-memory key–value database, cache and message broker, with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indices. Wikipedia

Developer(s): Redis

License: BSD 3-clause

Initial release: May 10, 2009; 13 years ago

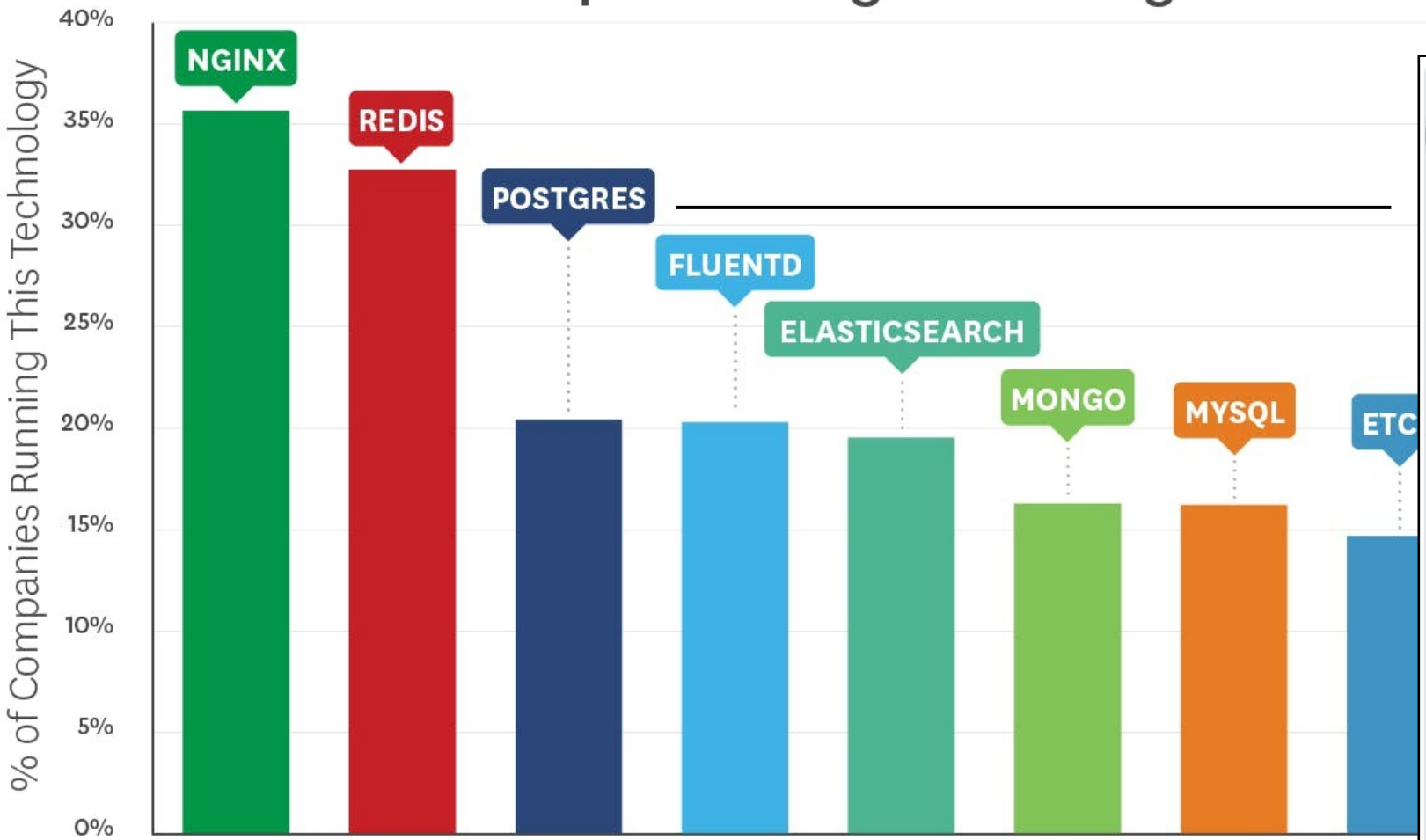
Operating system: Unix-like

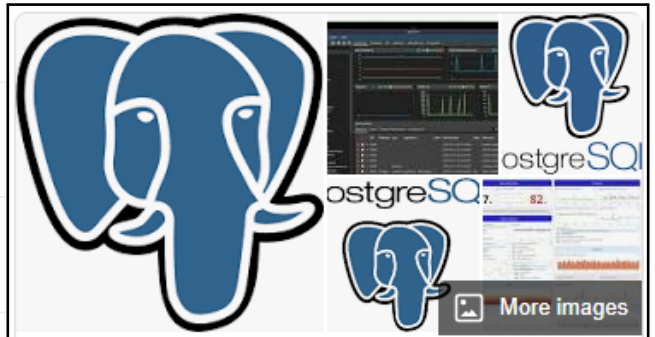
Original author(s): Salvatore Sanfilippo

Stable release: 7.0.4 / July 18, 2022; 35 days ago

Written in: C

Top Technologies Running on Docker





PostgreSQL

PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley. Wikipedia

Developer(s): PostgreSQL Global Development Group

License: PostgreSQL License (free and open-source, permissive)

Repository: git.postgresql.org/gitweb/?p=postgresql.git

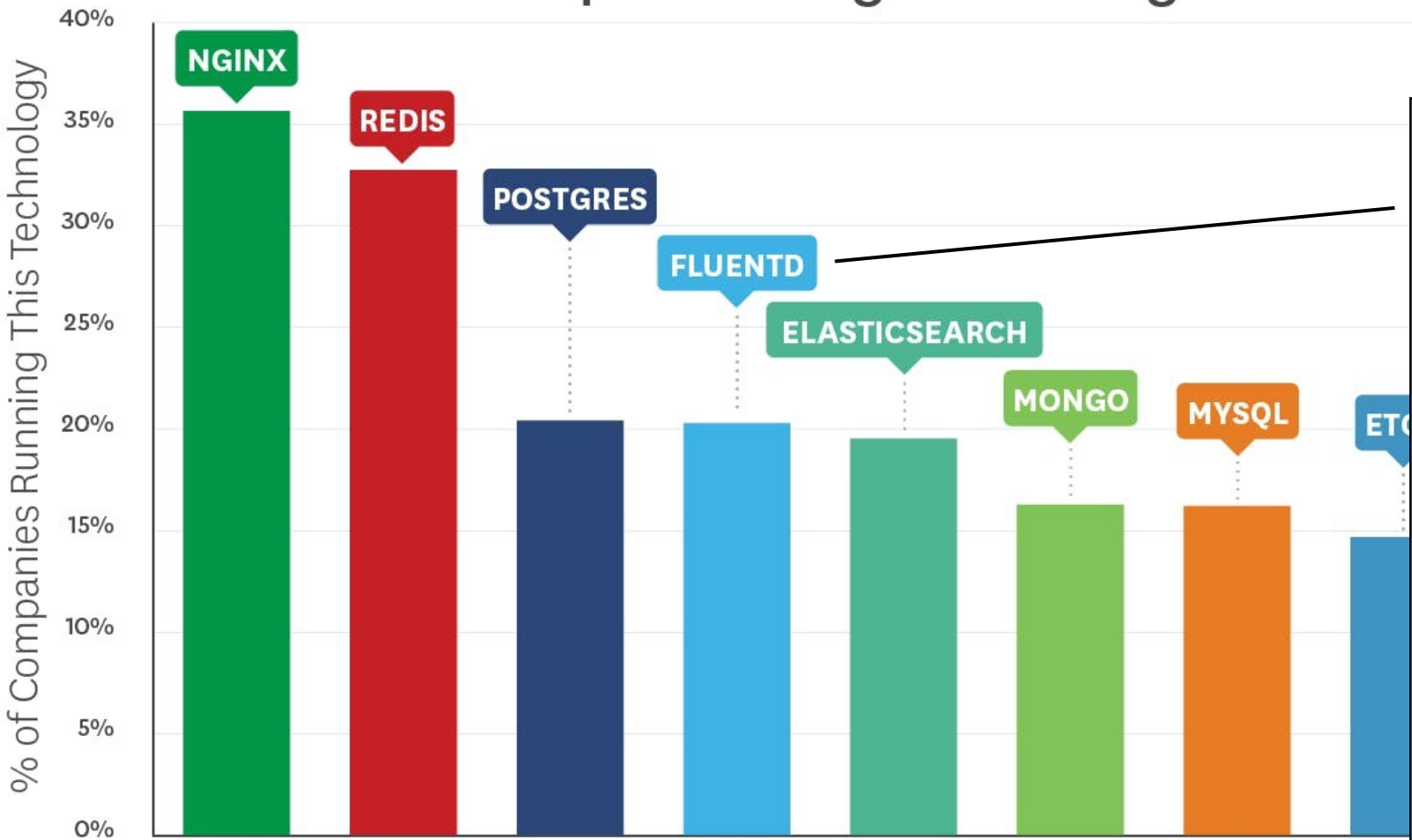
Initial release: 8 July 1996; 26 years ago

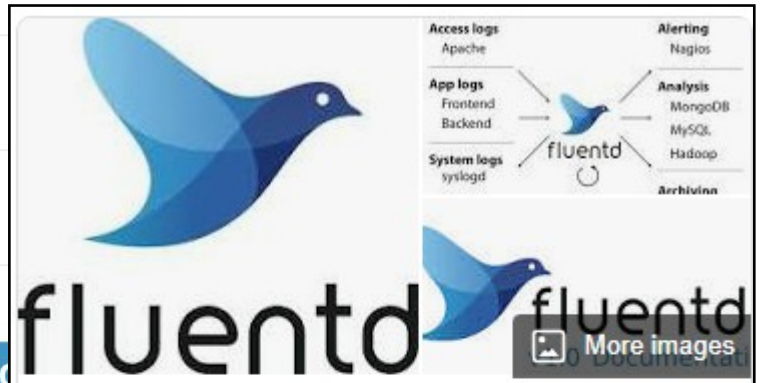
Operating system: macOS, Windows, Linux, FreeBSD, OpenBSD

Stable release: 14.5 / 11 August 2022; 10 days ago

Written in: C

Top Technologies Running on Docker





Fluentd

Software

Fluentd is a cross platform open-source data collection software project originally developed at Treasure Data. It is written primarily in the Ruby programming language. Wikipedia

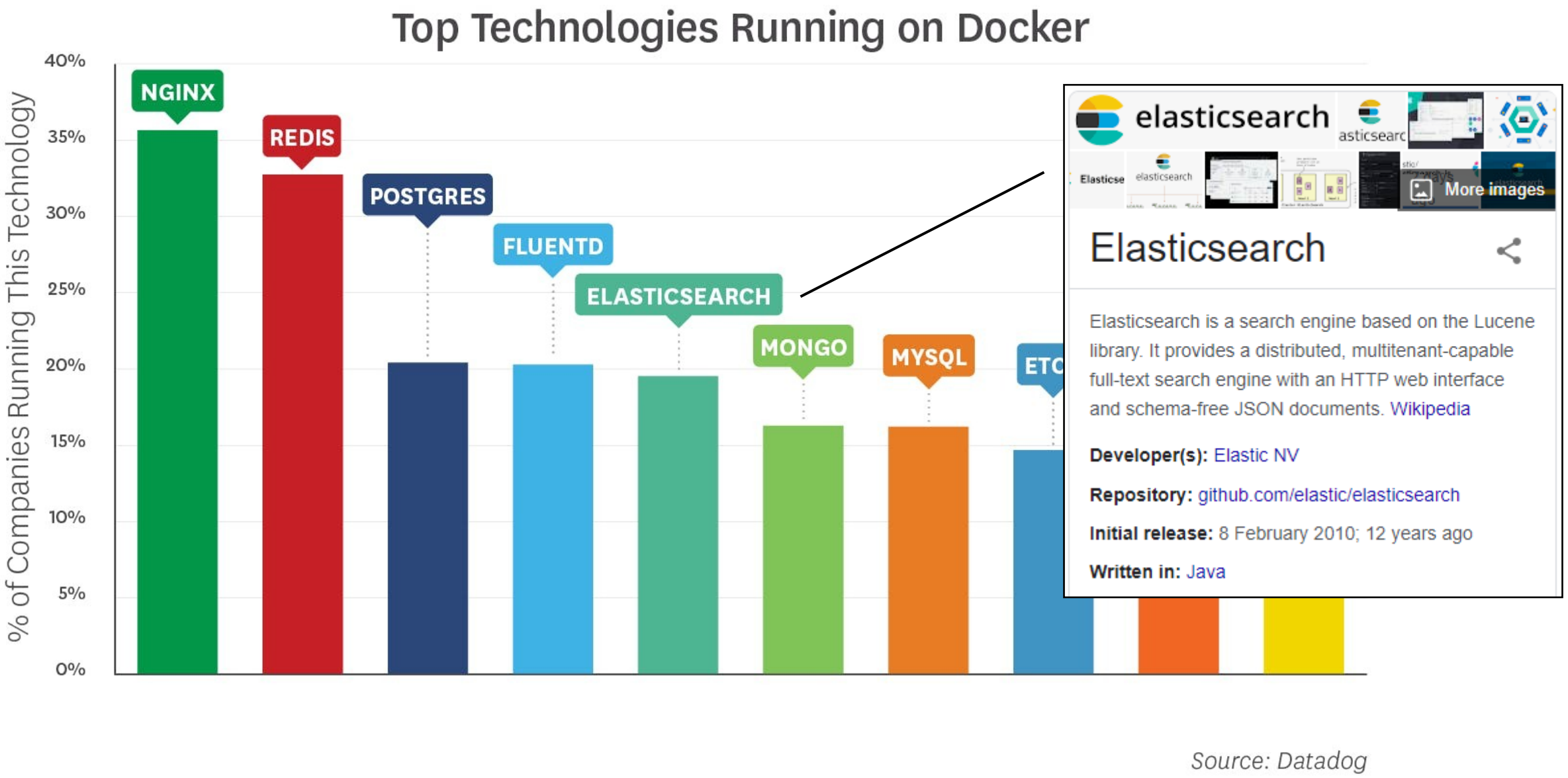
Developer(s): Treasure Data

License: Apache 2.0

Initial release: 10 October 2011; 10 years ago

Stable release: 1.14.5 / February 9, 2022; 5 months ago

Written in: C, Ruby



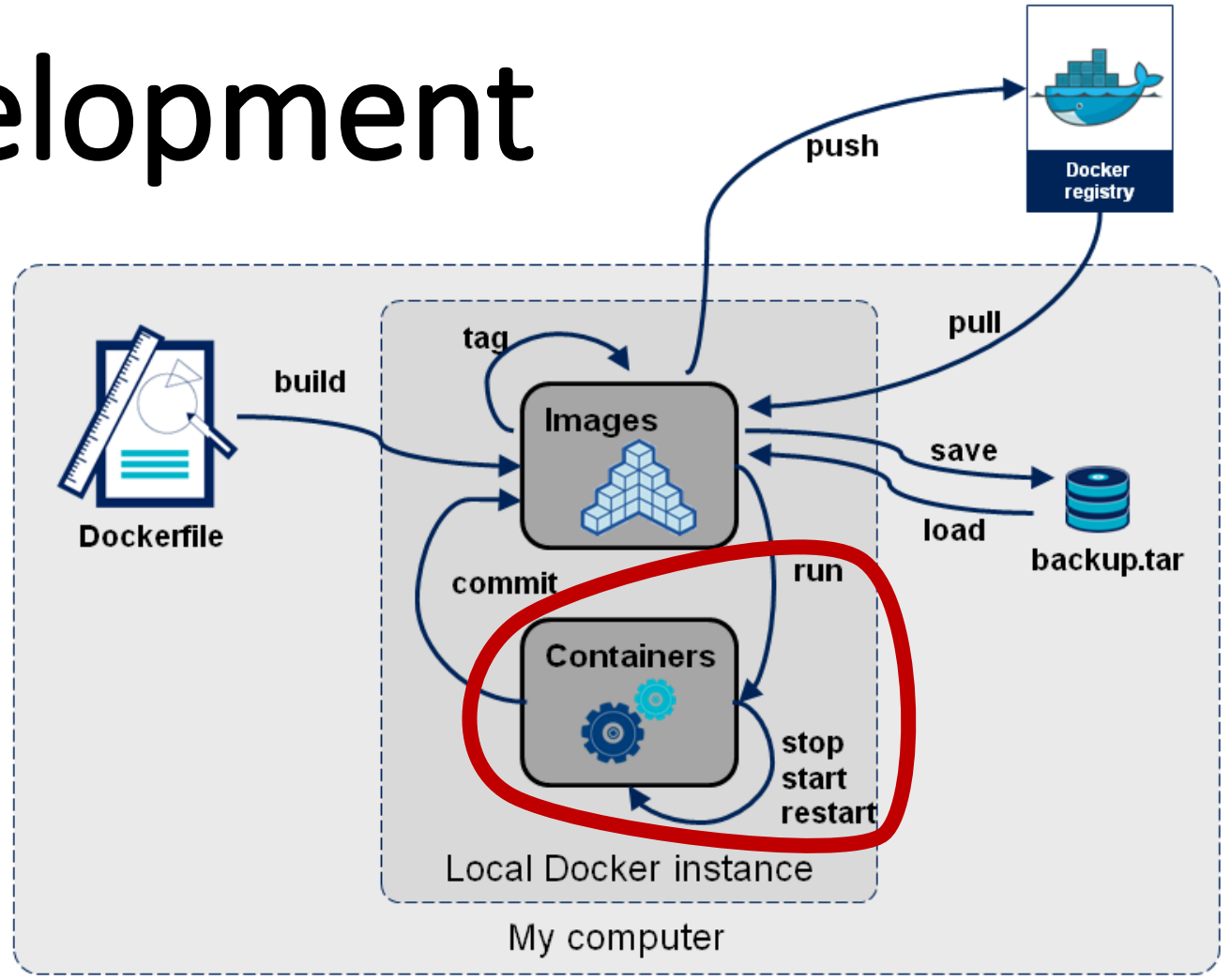
Source: Datadog

Technological foundations of software development

Run your software anywhere with Docker

Part 2: Docker

Part 2.3: Managing containers



ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Container management commands

command	description
`docker create` *image* [*command*] `docker run` *image* [*command*]	create the container = `create` + `start`
`docker start` *container...* `docker stop` *container...* `docker kill` *container...* `docker restart` *container...*	start the container graceful[2] stop kill (SIGKILL) the container = `stop` + `start`
`docker pause` *container...* `docker unpause` *container...*	suspend the container resume the container
`docker rm` [`-f`[3]] *container...*	destroy the container

[2] send SIGTERM to the main process + SIGKILL 10 seconds later

[3] `-f` allows removing running containers (= `docker kill` + `docker rm`)

https://dockerlabs.collabnix.com/docker/cheatsheet/

`docker run` and options

`docker run [OPTIONS] IMAGE[:TAG] [COMMAND]`

Run a command in a new container.

metadata

`--name=CNTNR_NAME`	*Assign a name to the container.*
`-l, --label NAME[=VALUE]`	*Set metadata on the container.*

process

`-d, --detach`	*Run in the background.*
`-i, --interactive`	*Keep STDIN open.*
`-t, --tty`	*Allocate a pseudo-TTY.*
`--rm`	*Automatically remove the container when process exits.*
`-u USER`	*Run as username or UID.*
`--privileged`	*Give extended privileges.*
`-w DIR`	*Set working directory.*
`-e NAME=VALUE`	*Set environment variable.*
`--restart=POLICY`	*Restart policy.* **`no`** **`on-failure[:RETRIES]`** **`always`** **`unless-stopped`**

network

`-P, --publish-all`	*Publish all exposed ports to random ports.*
`-p HOST_PORT:CNTNR_PORT`	*Expose a port or a range of ports.*
`--network=NETWORK_NAME`	*Connect container to a network.*
`--dns=DNS_SERVER1[,DNS_SERVER2]`	*Set custom dns servers.*
`--add-host=HOSTNAME:IP`	*Add a line to /etc/hosts.*

file system

`--read-only`	*Mount the container's root file system as read only.*
`-v, --volume [HOST_SRC:]CNTNR_DEST`	*Mount a volume between host and the container file system.*
`--volumes-from=CNTNR_ID`	*Mount all volumes from another container.*

Example: pull and run a h2 database

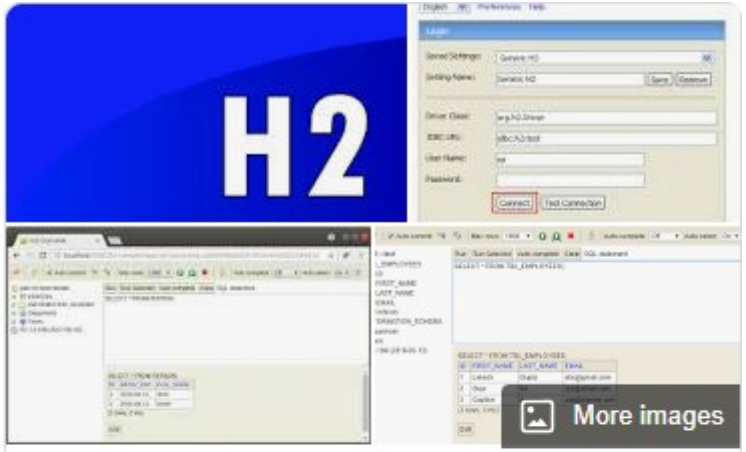
Documentation at https://hub.docker.com/r/oscarfonts/h2

Get the image:

```
docker pull oscarfonts/h2
```

Run as a service, exposing ports 1521 (TCP database server) and 81 (web interface) and mapping DATA_DIR to host:

```
docker run -d \
           -p 1521:1521 \
           -p 81:81 \
           -v /path/to/local/data_dir:/opt/h2-data \
           --name=MyH2Instance \
           oscarfonts/h2
```



H2

System software

H2 is a relational database management system written in Java. It can be embedded in Java applications or run in client-server mode. The software is available as open source software Mozilla Public License 2.0 or the original Eclipse Public License. Wikipedia

License: Eclipse Public License or Mozilla Public License 2.0

Programming language: Java

Interacting with the container

command	description
`docker attach` *container*	attach to a running container (stdin/stdout/stderr)
`docker cp` *container:path hostpath|-* `docker cp` *hostpath|- container:path*	copy files from the container copy files into the container
`docker export` *container*	export the content of the container (tar archive)
`docker exec` *container args...*	run a command in an existing container (**useful** for debugging)
`docker wait` *container*	wait until the container terminates and return the exit code
`docker commit` *container image*	commit a new docker image (snapshot of the container)

https://dockerlabs.collabnix.com/docker/cheatsheet/

Run docker exec on a running container

First, start a container

```
$ docker run --name ubuntu_bash --rm -i -t ubuntu bash
```

This will create a container named `ubuntu_bash` and start a Bash session.

Next, execute a command on the container.

```
$ docker exec -d ubuntu_bash touch /tmp/execWorks
```

This will create a new file `/tmp/execWorks` inside the running container `ubuntu_bash`, in the background.

Next, execute an interactive bash shell on the container.

```
$  docker exec -it ubuntu_bash bash
```

This will create a new Bash session in the container `ubuntu_bash`.

docker cp: copy local files in the container

Copy a local file into container

```
$ docker cp ./some_file CONTAINER:/work
```

Copy files from container to local path

```
$ docker cp CONTAINER:/var/logs/ /tmp/app_logs
```

Copy a file from container to stdout. Please note cp command produces a tar stream

```
$ docker cp CONTAINER:/var/logs/app.log - | tar x -O | grep "ERROR"
```

https://docs.docker.com/engine/reference/commandline/cp/

Inspecting the container

command	description
`docker ps`	list running containers
`docker ps -a`	list all containers
`docker logs` [`-f`[6]] *container*	show the container output (*stdout+stderr*)
`docker top` *container* [*ps options*]	list the processes running inside the containers
`docker diff` *container*	show the differences with the image (modified files)
`docker inspect` *container...*	show low-level infos (in json format)

https://dockerlabs.collabnix.com/docker/cheatsheet/

Example: show logs of a container

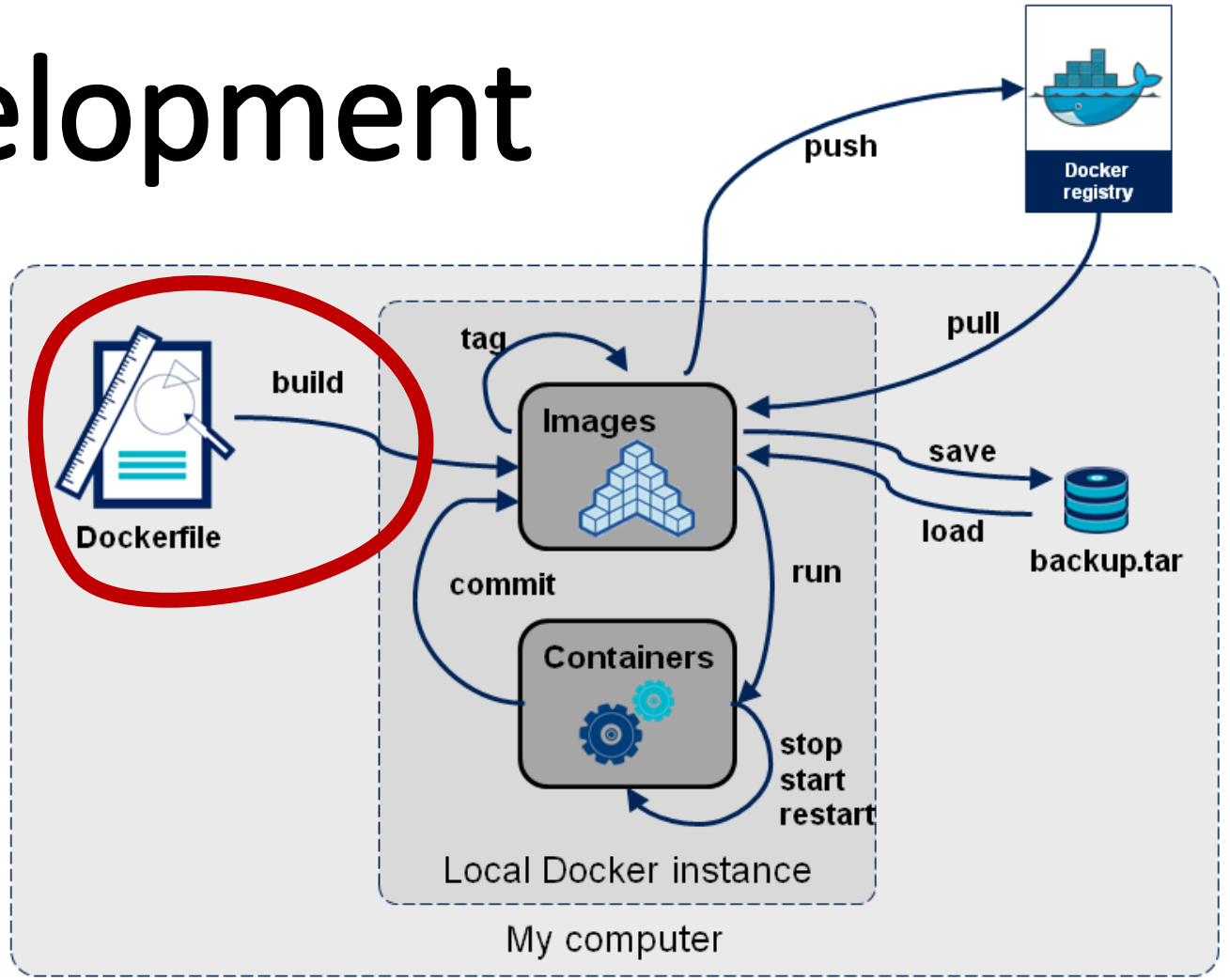
```
berty@Bionic201907:~$ docker logs Postgres1
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default timezone ... Etc/UTC
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

    pg_ctl -D /var/lib/postgresql/data -l logfile start


WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.
waiting for server to start....2019-09-28 14:00:32.870 UTC [43] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGS
2019-09-28 14:00:32.881 UTC [44] LOG:  database system was shut down at 2019-09-28 14:00:32 UTC
2019-09-28 14:00:32.890 UTC [43] LOG:  database system is ready to accept connections
 done
server started

/usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*

waiting for server to shut down...2019-09-28 14:00:32.970 UTC [43] LOG:  received fast shutdown request
.2019-09-28 14:00:32.974 UTC [43] LOG:  aborting any active transactions
2019-09-28 14:00:32.979 UTC [43] LOG:  background worker "logical replication launcher" (PID 50) exited with exit code 1
2019-09-28 14:00:32.979 UTC [45] LOG:  shutting down
2019-09-28 14:00:33.005 UTC [43] LOG:  database system is shut down
```

Technological foundations of software development

Run your software anywhere with Docker

Part 2: Docker

Part 2.4: Image specification and build



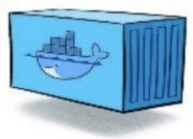
ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Build and run an image

Example https://github.com/dockersamples/linux_tweet_app


```
docker build -t linux_tweet_app .
```


```
docker container run --detach -p 80:80 linux_tweet_app
```



Dockerfile instructions

Reference: https://docs.docker.com/engine/reference/builder/

FROM <image_id>
base image to build this image from

RUN <command> *shell form*

RUN ["<executable>",
"<param1>",
...,
"<paramN>"] *exec form*
executes command to modify container's file system state

MAINTAINER <name>
provides information about image creator

LABEL <key>=<value>
*adds searchable **metadata** to image*

ARG <name>[=<default value>]
*defines overridable **build-time parameter**:*
`docker build --build-arg <name>=<value> .`

ENV <key>=<value>
*defines **environment variable** that will be visible during image build-time and container run-time*

ADD <src> <dest>
*copies files from <src> (**file, directory or URL**) and adds them to container file system under <dst> path*

COPY <src> <dest> *similar to **ADD**, does not support URLs*

VOLUME <dest>
*defines **mount point** to be shared with host or other containters*

EXPOSE <port>
*informs Docker engine that container listens to **port** at run-time*

WORKDIR <dest>
sets build-time and run-time working directory

USER <user>
defines run-time user to start container process

STOPSIGNAL <signal>
defines signal to use to notify container process to stop

ENTRYPOINT *shell form* or *exec form*
*defines run-time **command prefix** that will be added to all run commands executed by `docker run`*

CMD *shell form* or *exec form*
*defines run-time **command** to be executed by default when `docker run` command is executed*

https://dockerlabs.collabnix.com/docker/cheatsheet/

Technological foundations of software development

Run your software anywhere with Docker

Part 2: Docker

Part 2.5: Multi-container applications

Docker

Docker Compose

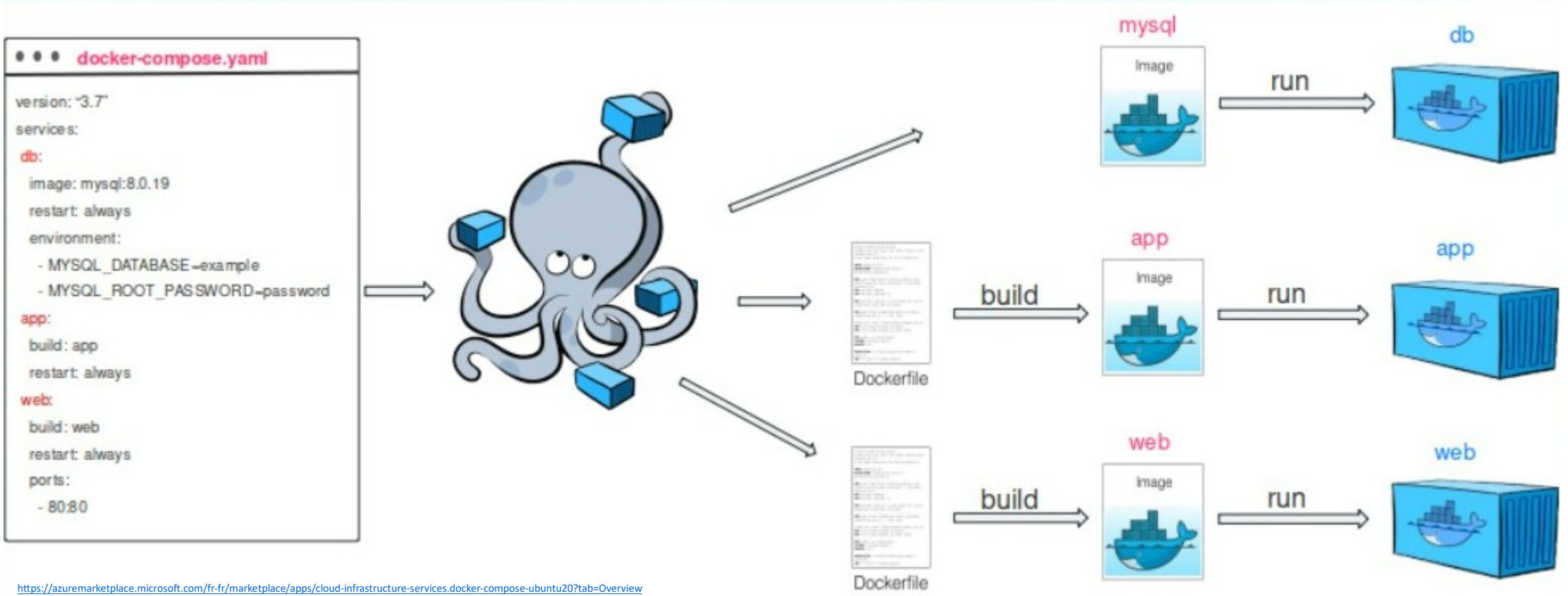
ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Docker compose

https://docs.docker.com/compose/

Compose is a tool for defining and running multi-container Docker applications

`docker-compose.yml` configuration file , and `docker-compose` (now `docker compose`) cli



https://azuremarketplace.microsoft.com/fr-fr/marketplace/apps/cloud-infrastructure-services.docker-compose-ubuntu20?tab=Overview

Docker compose

https://docs.docker.com/compose/

Compose is a tool for defining and running multi-container Docker applications

`docker-compose.yml` configuration file , and `docker-compose` (now `docker compose`) cli

Basic example

```
# docker-compose.yml
version: '2'

services:
  web:
    build:
    # build from Dockerfile
      context: ./Path
      dockerfile: Dockerfile
    ports:
     - "5000:5000"
    volumes:
     - .:/code
  redis:
    image: redis
```

Service name

Build configuration for creating container image from source

Exposes container ports
`HOST_PORT:CONTAINER_PORT`

Defines mount host paths or named volumes accessible by service containers
`VOLUME:CONTAINER_PATH`

Commands

```
docker-compose start
docker-compose stop
```

```
docker-compose pause
docker-compose unpause
```

```
docker-compose ps
docker-compose up
docker-compose down
```

https://devhints.io/docker-compose

Building

```
web:
  # build from Dockerfile
  build: .
  args:     # Add build arguments
    APP_HOME: app
```

```
  # build from custom Dockerfile
  build:
    context: ./dir
    dockerfile: Dockerfile.dev
```

```
  # build from image
  image: ubuntu
  image: ubuntu:14.04
  image: tutum/influxdb
  image: example-registry:4000/postgresql
  image: a4bc65fd
```

Other options

```
  # make this service extend another
  extends:
    file: common.yml  # optional
    service: webapp
```

```
  volumes:
    - /var/lib/mysql
    - ./_data:/var/lib/mysql
```

```
# automatically restart container
restart: unless-stopped
# always, on-failure, no (default)
```

Ports

```
  ports:
    - "3000"
    - "8000:80"  # host:container
```

```
# expose ports to linked services
#   (not to host)
    expose: ["3000"]
```

Environment variables

```
  # environment vars
  environment:
    RACK_ENV: development
  environment:
    - RACK_ENV=development
```

```
  # environment vars from file
  env_file: .env
  env_file: [.env, .development.env]
```

Commands

```
  # command to execute
  command: bundle exec thin -p 3000
  command: [bundle, exec, thin, -p, 3000]
```

```
  # override the entrypoint
  entrypoint: /app/start.sh
  entrypoint: [php, -d, vendor/bin/phpunit]
```

Dependencies

```
  # makes the `db` service available as the
  # hostname `database` (implies depends_on)
  links:
    - db:database
    - redis
```

```
  # make sure `db` is alive before starting
  depends_on:
    - db
```

https://devhints.io/docker-compose

Networking

myapp/docker-compose.yml

```yaml
version: "3.9"
services:
  web:
    build: .
    ports:
      - "8000:8000"
  db:
    image: postgres
    ports:
      - "8001:5432"
```

When you run docker compose up, the following happens:

- A network called `myapp_default` is created.
- A container is created using `web`'s configuration. It joins the network `myapp_default` under the name `web`.
- A container is created using `db`'s configuration. It joins the network `myapp_default` under the name `db`.

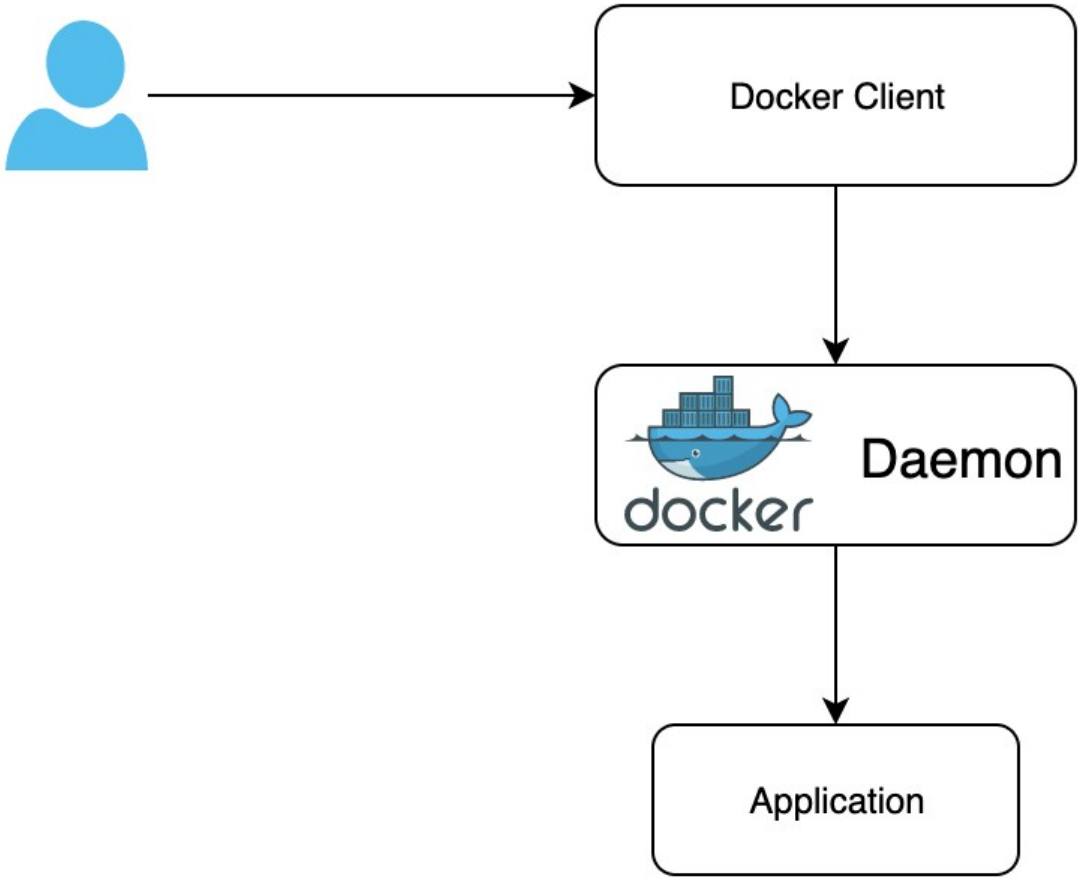
Technological foundations of software development

Run your software anywhere with Docker

Part 3: Docker alternatives and the containers ecosystem

ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Docker < v1.11

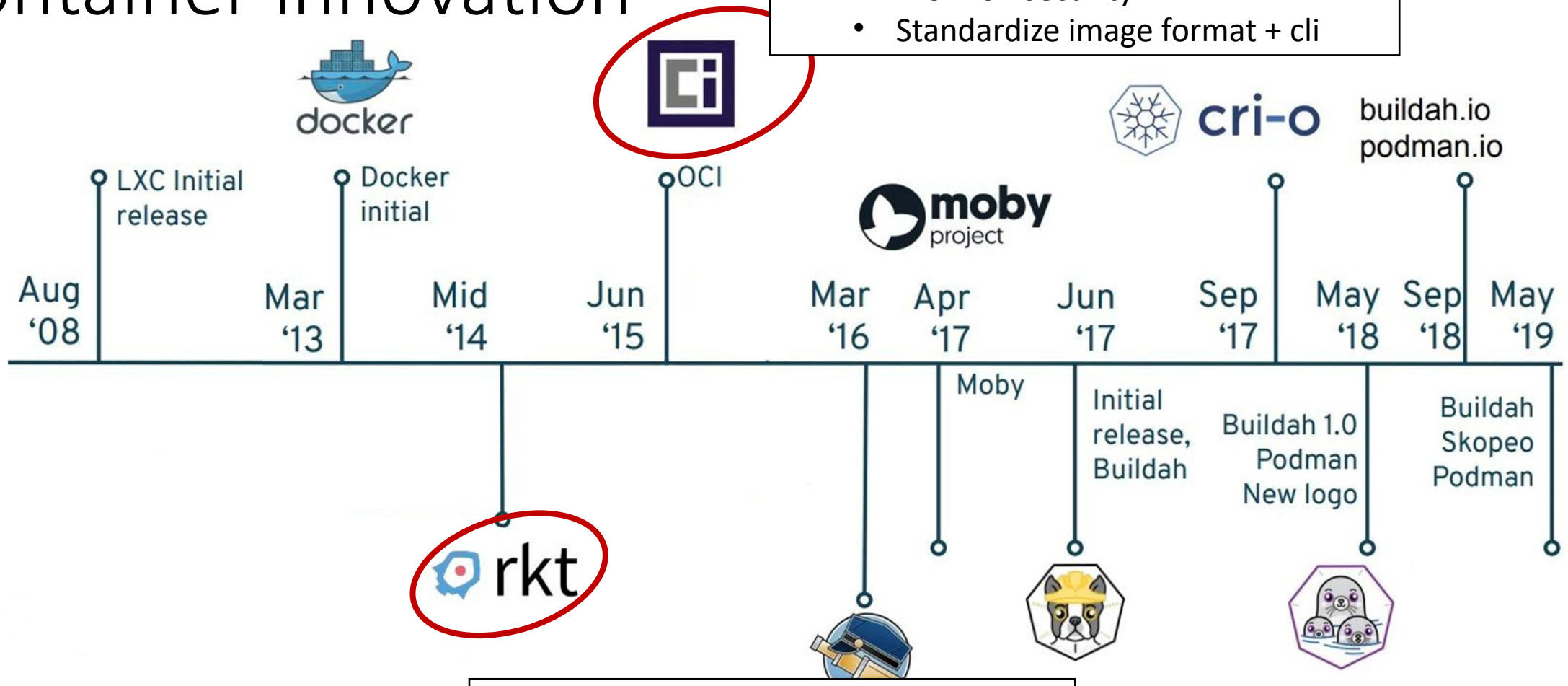


Docker Version < 1.11.0

- Docker centralized all containers under the same process containerd.io ==> Single Point of Failure
  - the containerd.io process **owns** all child processes (**running container**)
  - if containerd.io stops/crashes, all child processes are lost
- a single user runs containers with root privileges and full access to the host system ==> **Security issues**
- all images are downloaded and stored in the same folder /var/lib/docker/image/overlay2/imagedb/content/sha256

Container innovation

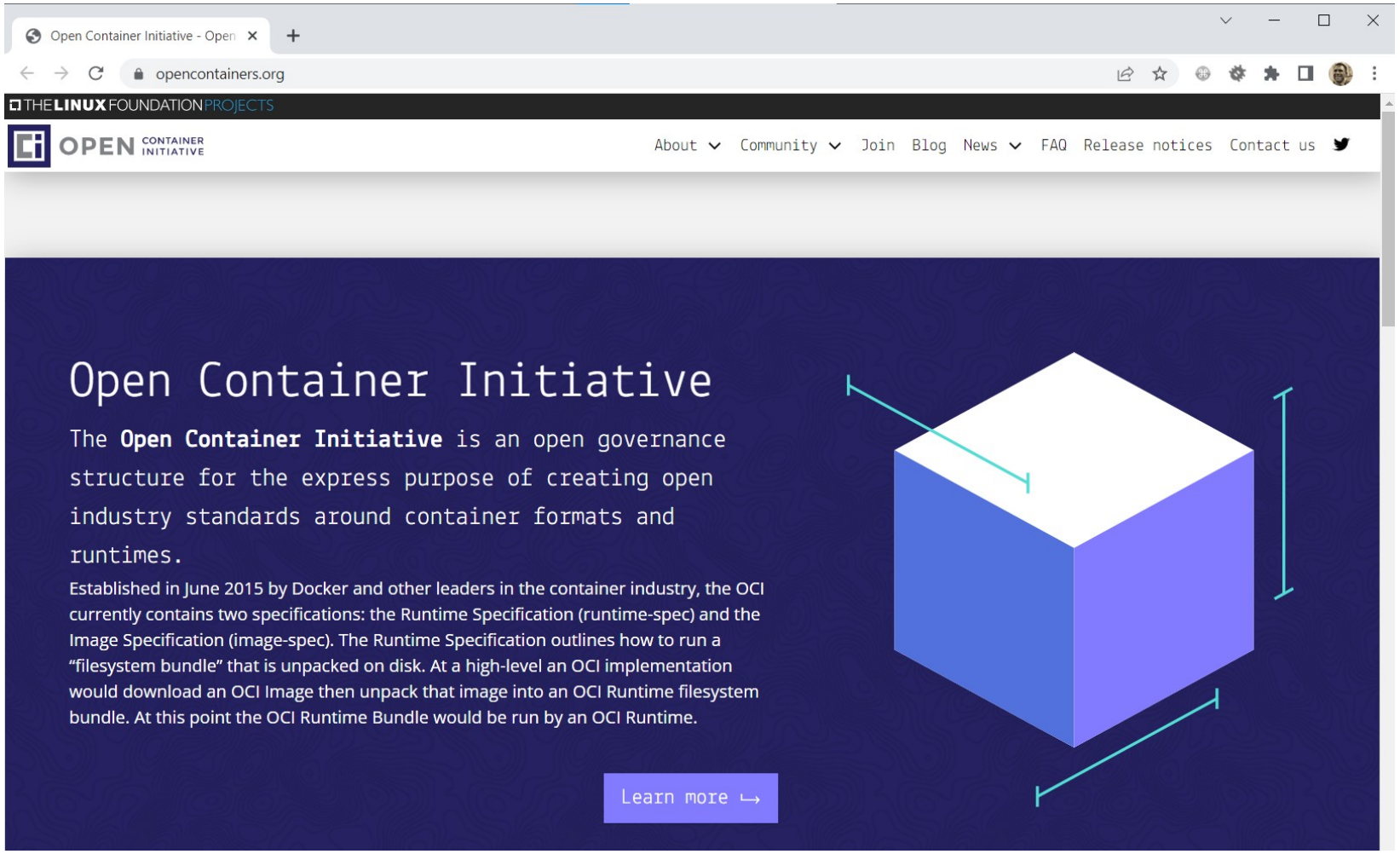
2. response: Creation Open Cloud Initiative
   - Work on security
   - Standardize image format + cli



1. Threat:
   - Better than Docker security-wise
   - Alternative image format + cli

https://dev.to/manishfoodtechs/rootless-docker-kick-docker-hackers-and-make-docker-more-secure-new-concept-70g

The Open Container Initiative



- Runtime Specification (runtime-spec)
- Image Specification (image-spec)
- The runc low-level container runtime

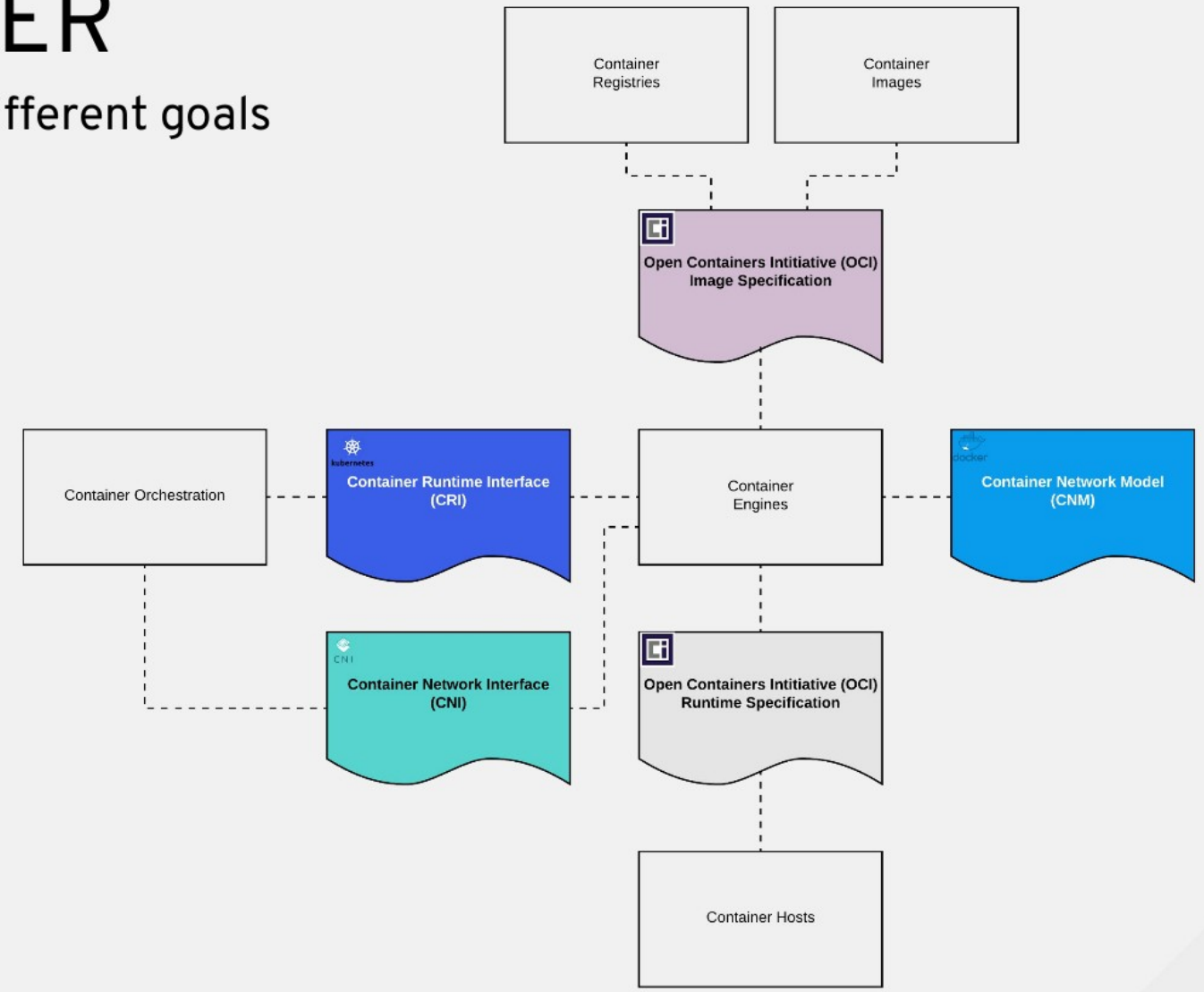
RUNC

WORKING TOGETHER

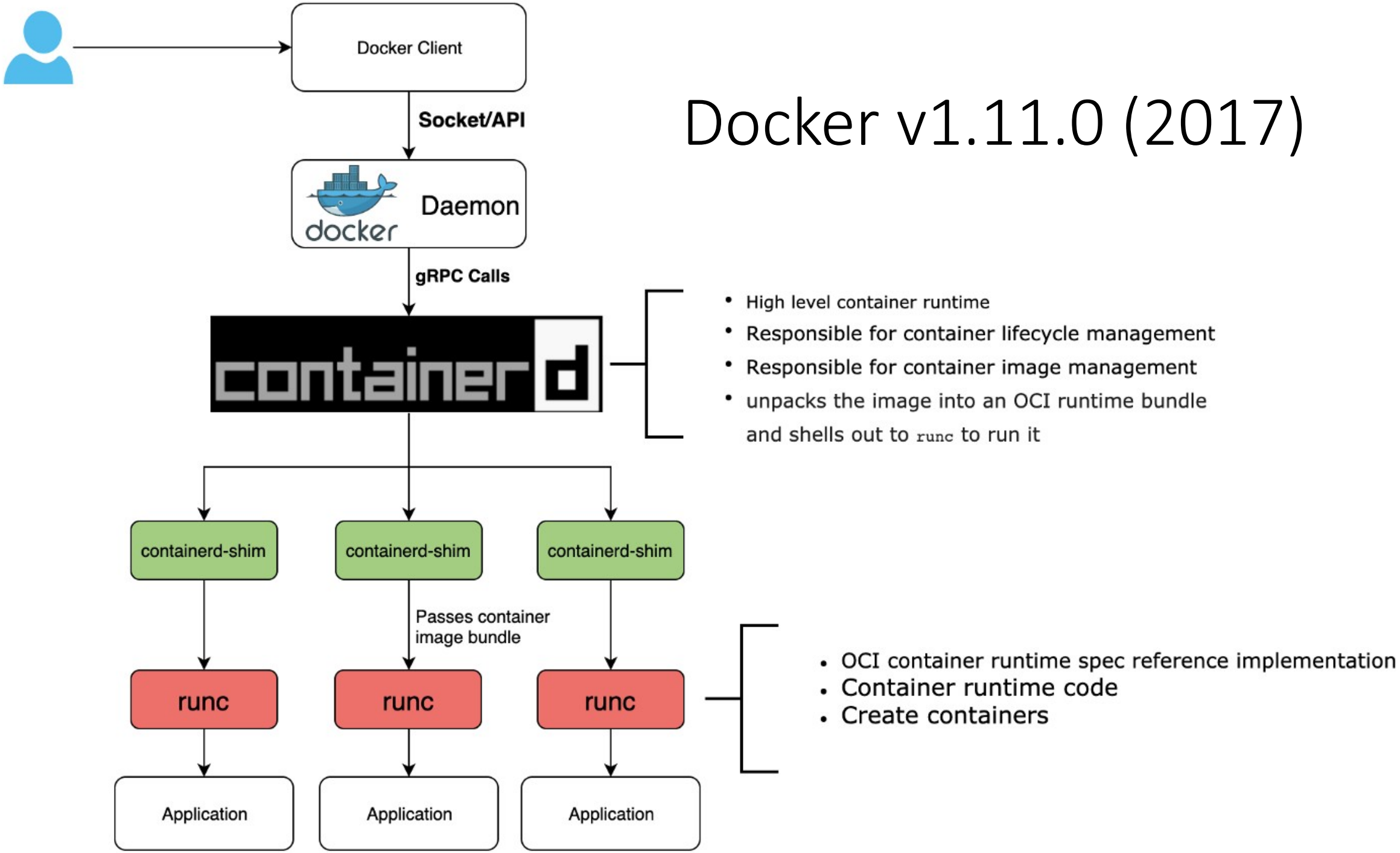
Standards in different places achieve different goals

Different standards are focused on different parts of the stack.

- Container Images & Registries
- Container Runtimes
- Container Networking



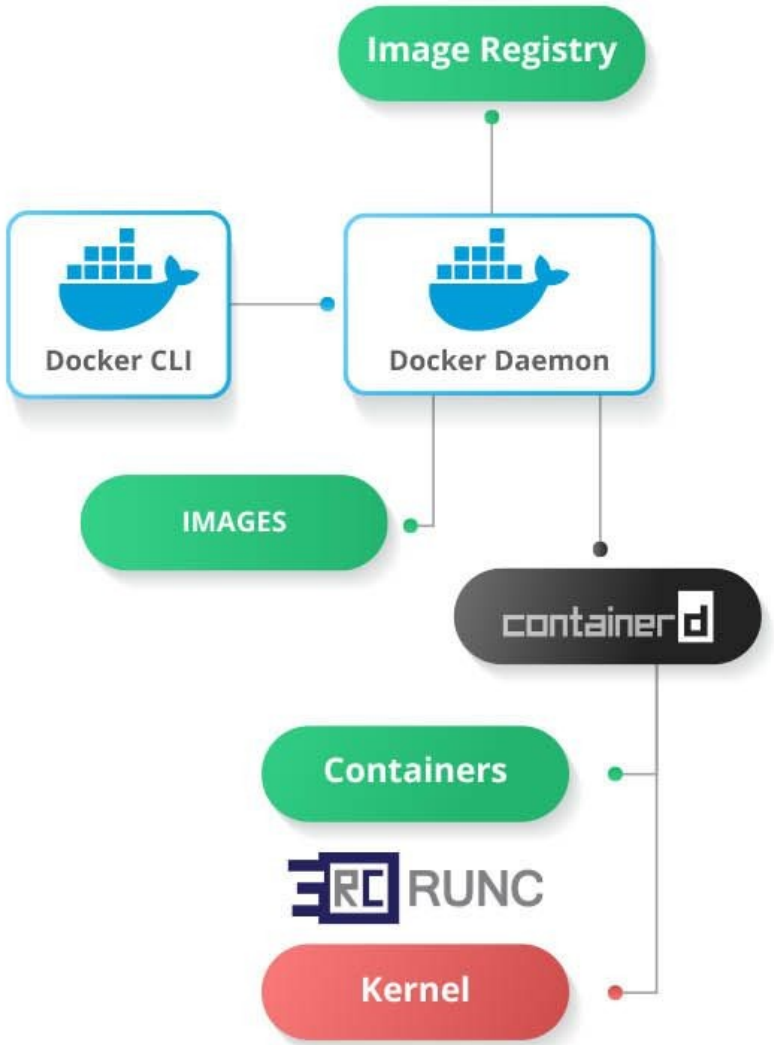
Docker v1.11.0 (2017)

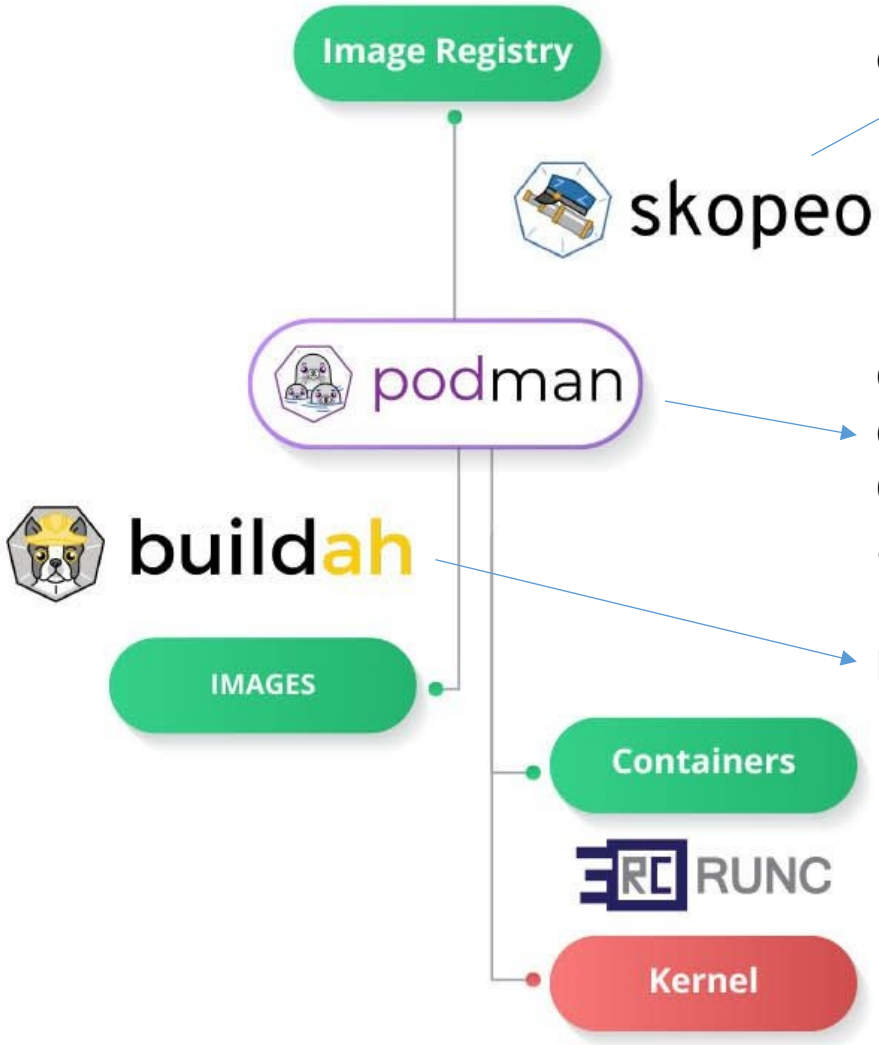


- High level container runtime
- Responsible for container lifecycle management
- Responsible for container image management
- unpacks the image into an OCI runtime bundle and shells out to `runc` to run it

- OCI container runtime spec reference implementation
- Container runtime code
- Create containers

Alternatives to Docker: podman (2018)





Transfer, inspect, delete OCI images once they're built, even remotely

daemonless container engine for developing, managing, and running OCI containers

```
alias docker=podman
```

build OCI compliant container images

https://ios.dz/from-docker-to-podman/#podman--un-puissant-alternatif

Other container runtimes

Kubernetes is deprecating Docker as a container runtime after v1.20. (announcement 2020)

rkt discontinued

MESOS not interoperable

cri-o

One lightweight option promoted by kubernetes

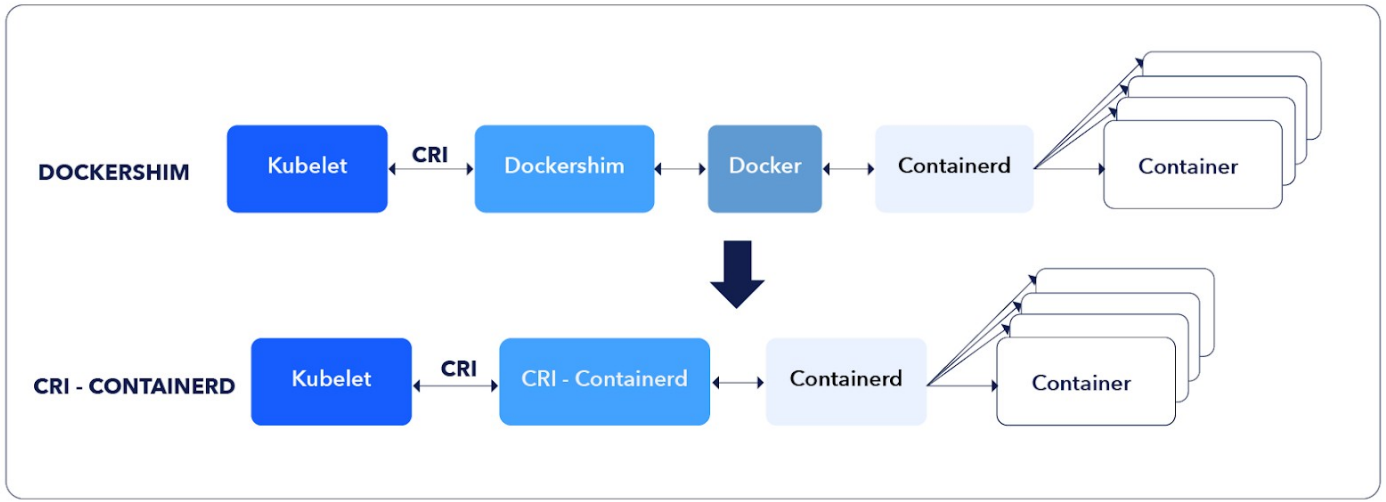
OpenVZ Virtuozzo Containers

containers

kata containers

LXC

Was there before Docker
No support for images



Still very active, stay tuned. See for example https://cto.ai/blog/overview-of-different-container-runtimes/ (dec. 2021)

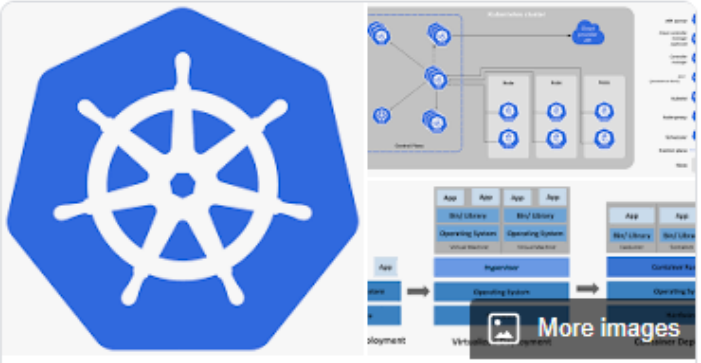
Technological foundations of software development

Run your software anywhere with Docker

Part 4: Container orchestration in the cloud

ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

Kubernetes (K8s)



Kubernetes

System software

Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Google originally designed Kubernetes, but the Cloud Native Computing Foundation now maintains the project. Wikipedia

License: Apache License 2.0

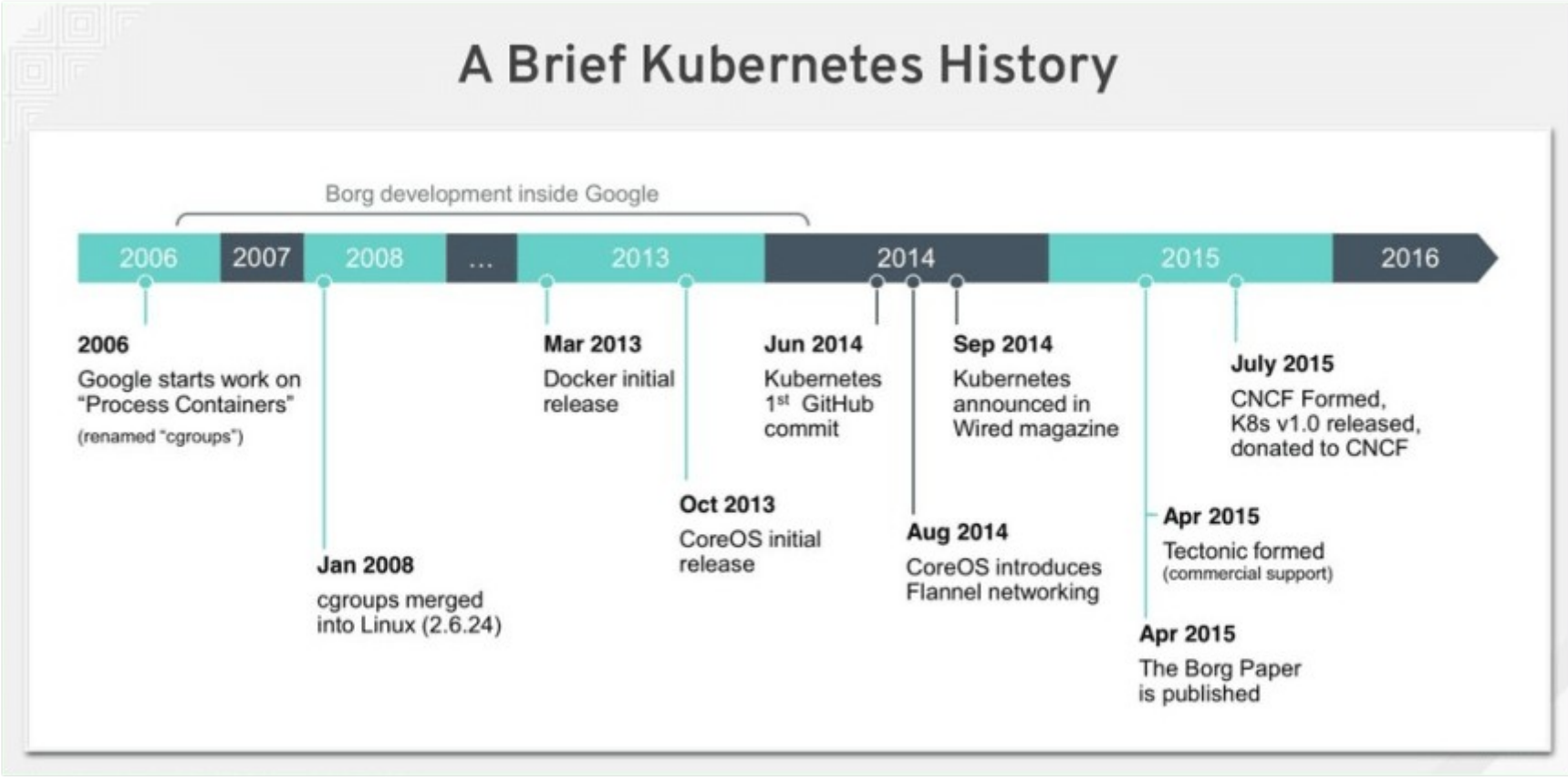
Developer(s): Cloud Native Computing Foundation

Initial release: 7 June 2014; 8 years ago

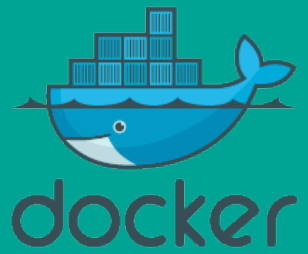
Written in: Go

Original author(s): Google

Stable release: 1.24.4 / 17 August 2022; 5 days ago



Kubernetes (K8s) in 100 seconds



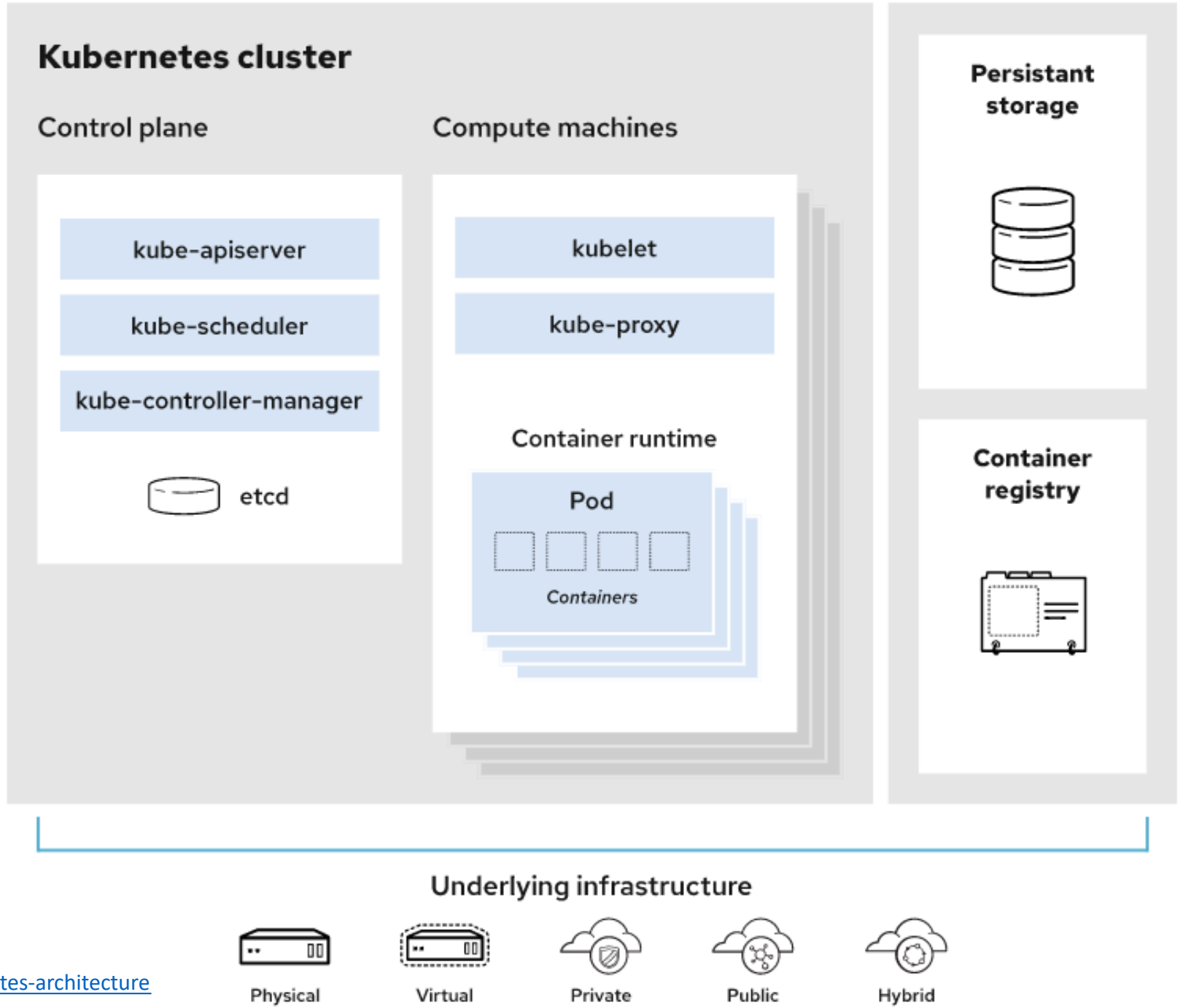
Docker

- Containers, isolated environment for application
- Automated building and deploying applications – CI
- Container platform for configuring building and distribution containers

Kubernetes

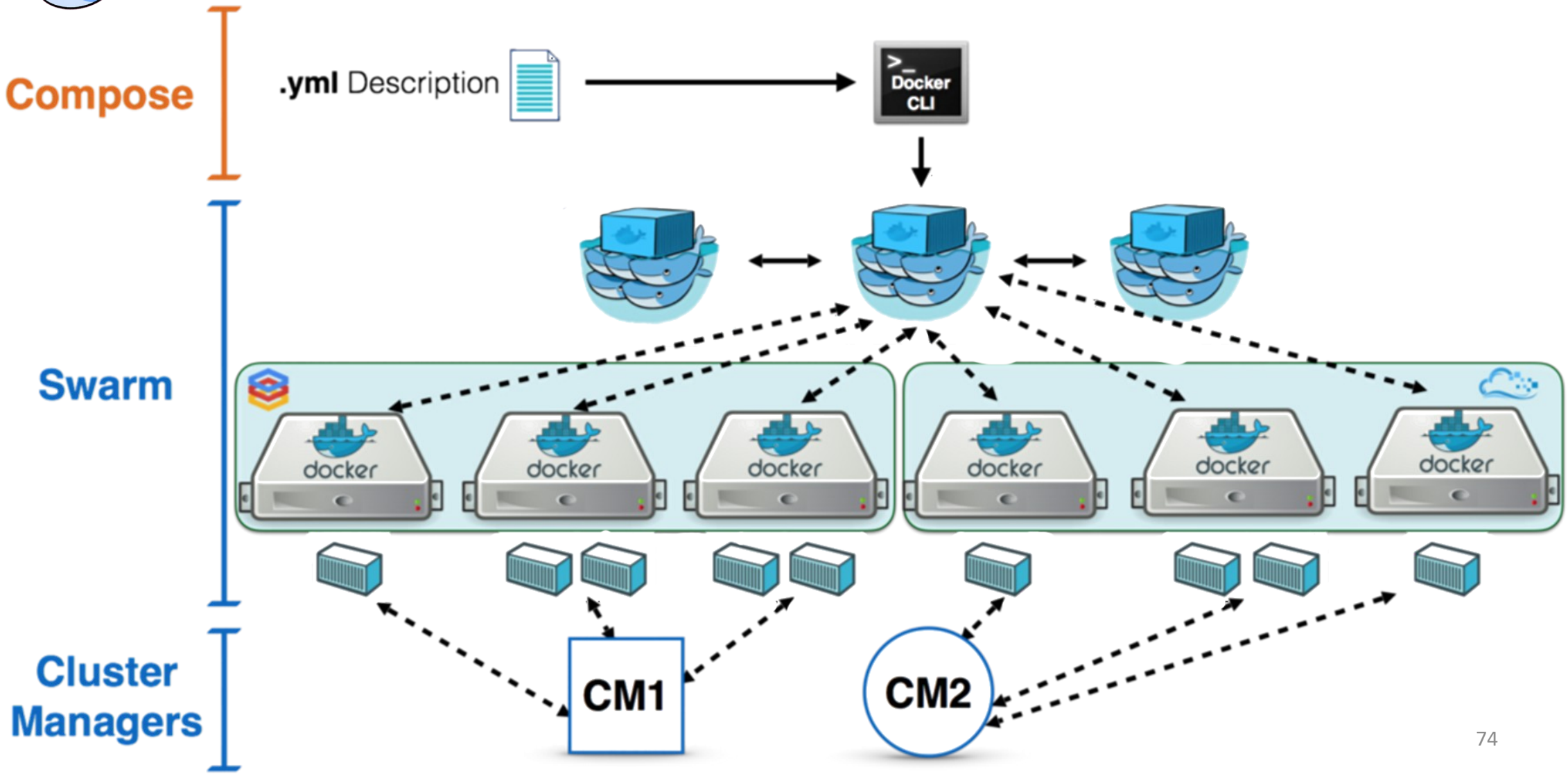
- Infrastructure for managing multiple containers
- Automated scheduling and management of application containers
- Ecosystem for managing a cluster of multiple containers

Architecture of a Kubernetes cluster

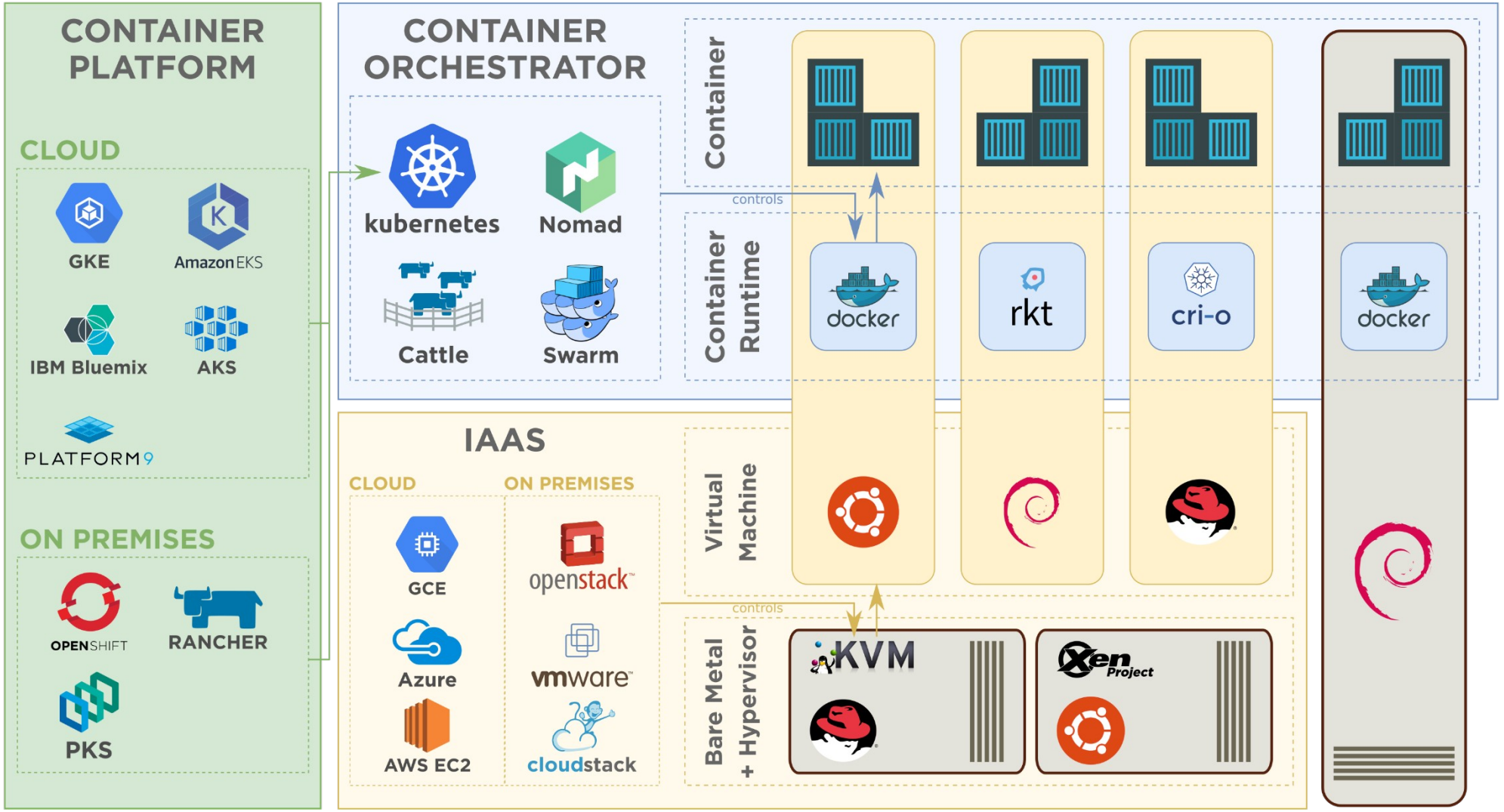


Details: https://www.redhat.com/en/topics/containers/kubernetes-architecture

Docker Swarm (under development)



Simplified containers ecosystem



https://twitter.com/raphink/status/973552094169968640

Cloud Native Computing Foundation

CLOUD NATIVE COMPUTING FOUNDATION

cncf.io

The Cloud Native Computing Foundation is a Linux Foundation project that was founded in 2015 to help advance container technology and align the tech industry around its evolution. Wikipedia

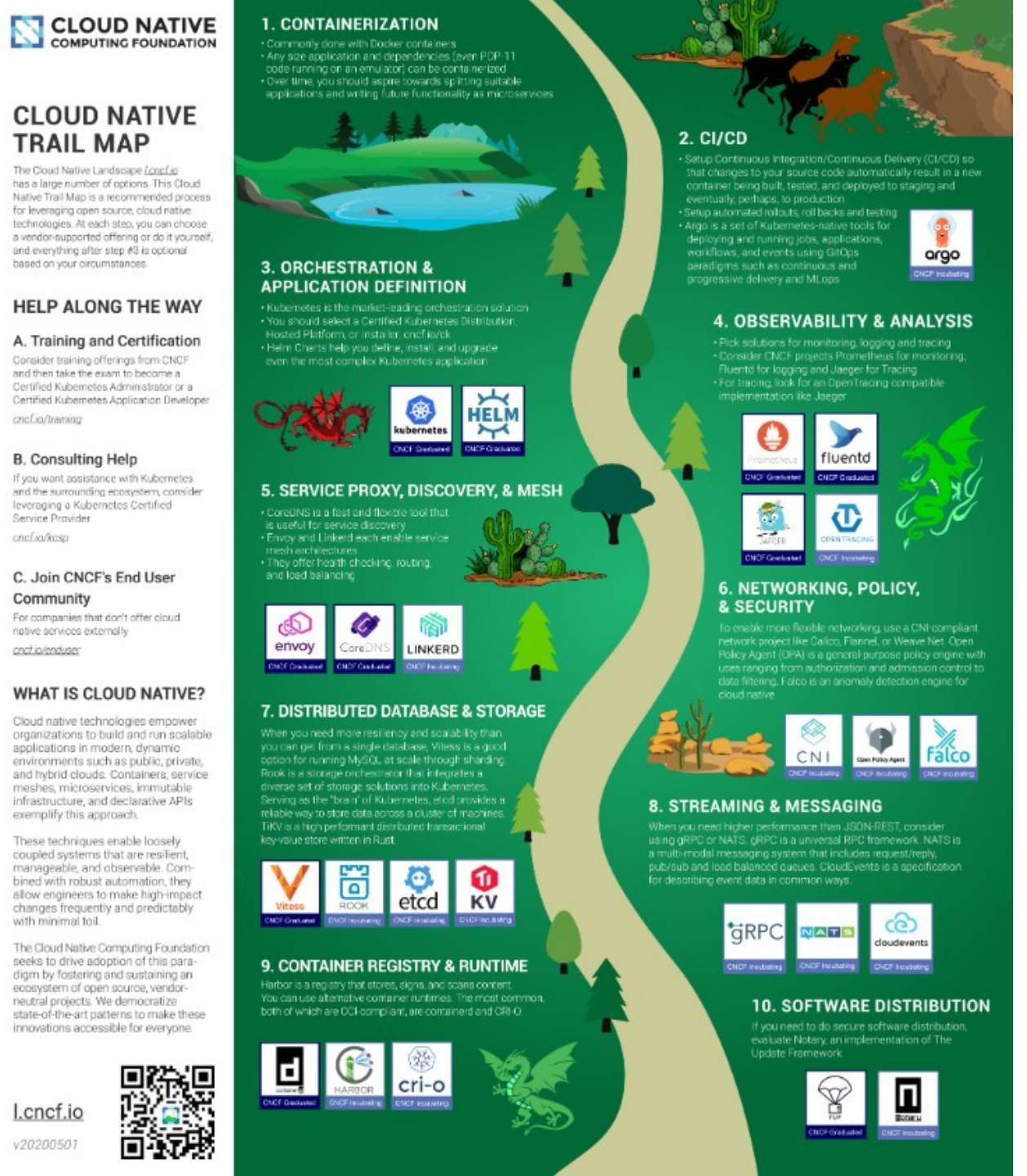
Founded: 2015

Parent organization: The Linux Foundation

Abbreviation: CNCF

Purpose: Building sustainable ecosystems for Cloud Native software

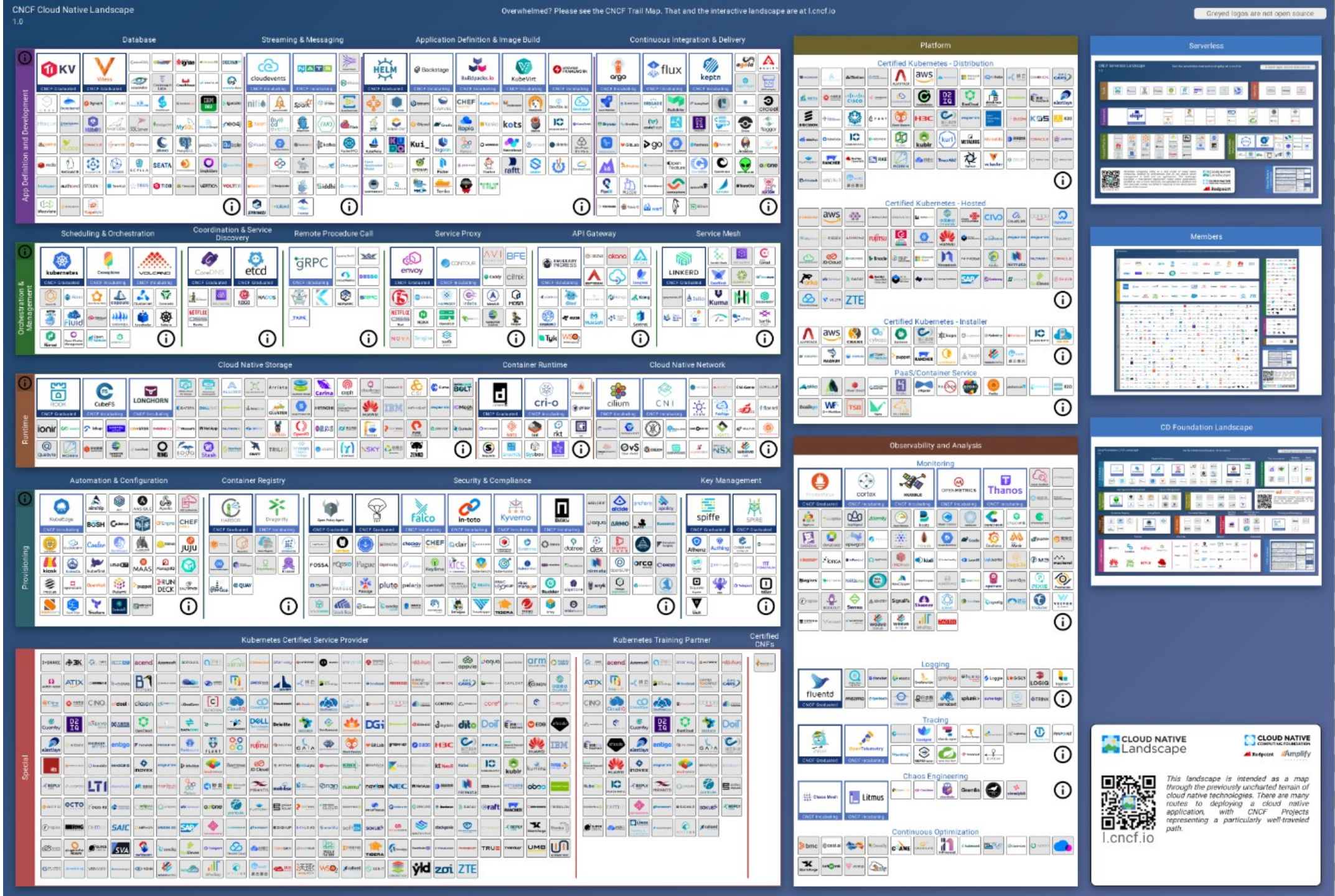
https://raw.githubusercontent.com/cncf/trailmap/master/CNCF_TrailMap_latest.png



Cloud native landscape

You are viewing **1,130 cards** with a total of **3,274,957 stars, market cap of $21.5T** and funding of **$51.7B**.

https://landscape.cncf.io/images/landscape.png



Technological foundations of software development

Run your software anywhere with Docker

ICM – Computer Science Major – Course unit on Technological foundations of computer science
M1 Cyber Physical and Social Systems – Course unit on CPS2 engineering and development, Part 2: Technological foundations of software development
Maxime Lefrançois https://maxime-lefrancois.info
online: https://ci.mines-stetienne.fr/cps2/course/tfsd/

… Your turn

Complete the TODO section:

https://ci.mines-stetienne.fr/cps2/course/tfsd/course-7.html#_todos