Introduction to Linux containers

Application in scientific practice

Docker and Singularity

Toni Hermoso Pulido

Bioinformatics Unit

CRG, Barcelona

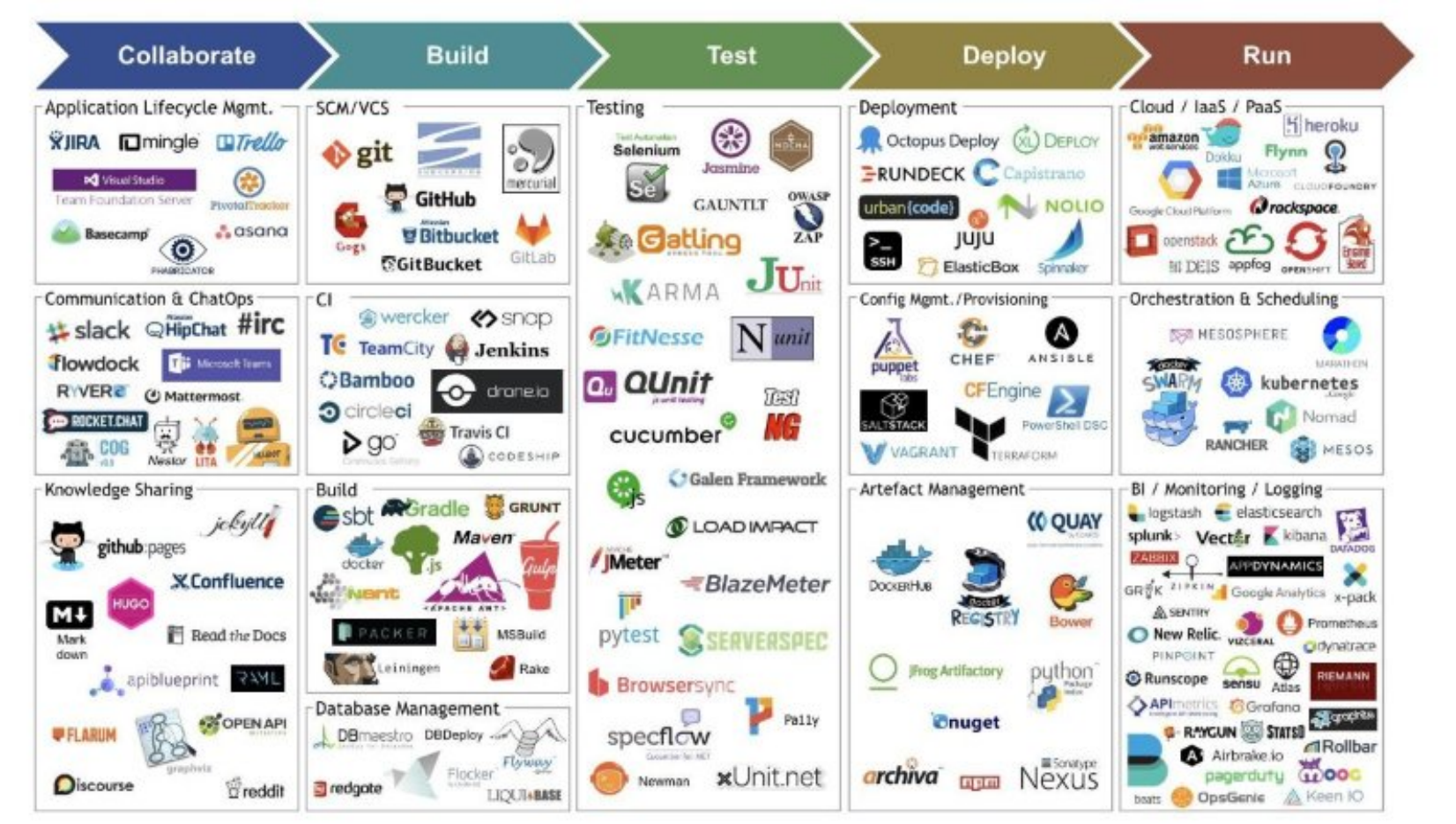
Recipes and instructions:

https://github.com/biocorecrg/containers-course

DevOps

- Software Engineering Culture
- Software Development + Software Operations
- Automate and monitor

DevOps



https://hostadvice.com/blog/devops-toolbox-jenkins-ansible-chef-puppet-vagrant-saltstack/

Containers

Containers in New Jersey

Virtualisation

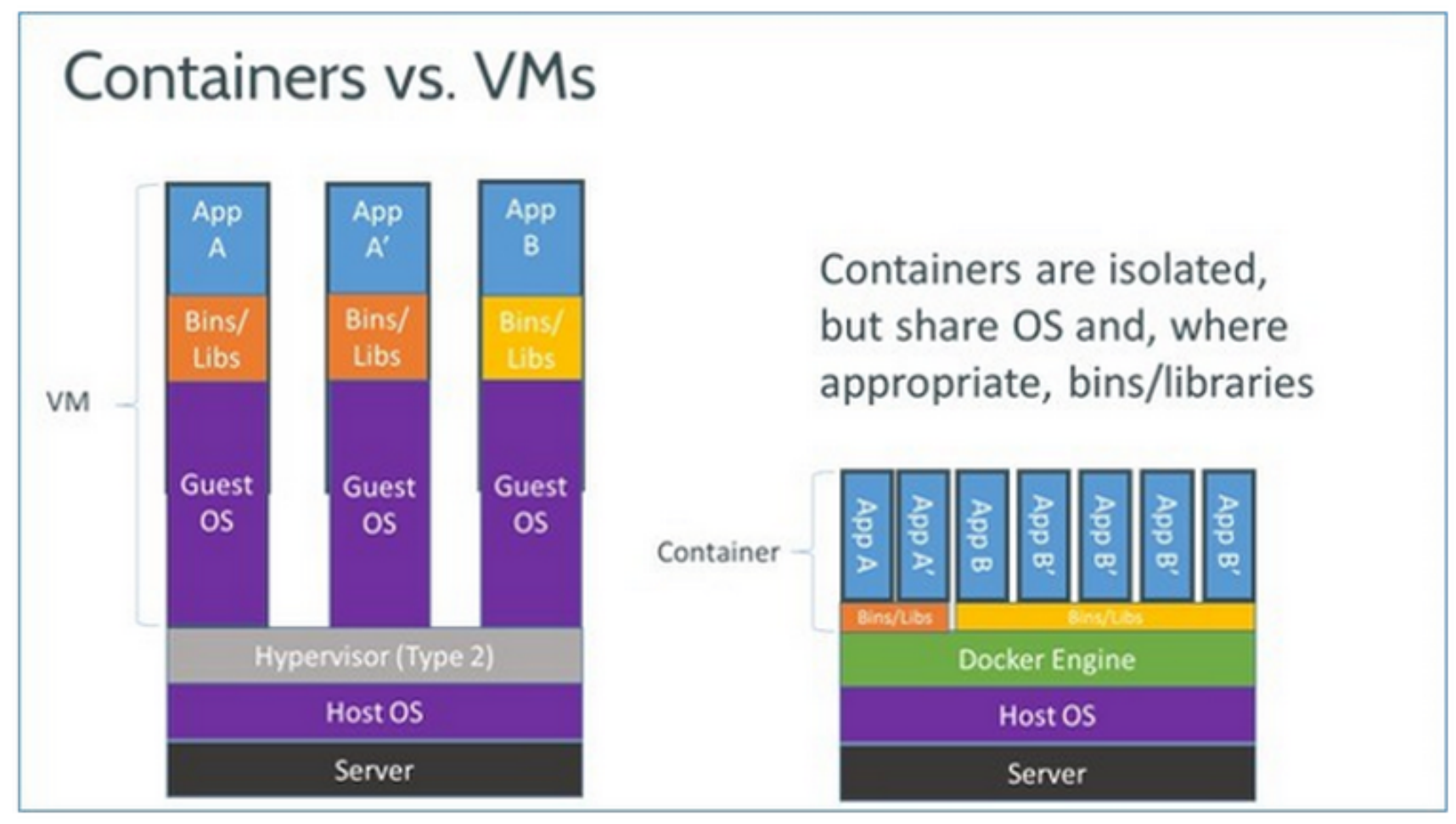
- Abstraction of physical hardware
- Depends on hypervisor (software)
- Do not confuse with hardware emulator
- Enable virtual machines
  - Every virtual machine with an Operating System

Containerisation

aka Lightweight virtualisation

- Abstraction of application layer
- Depends on host kernel (Operating System)
- Application and dependencies bundled all together

Virtual machines vs containers



http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/

Virtualisation

Pros and Cons

- **PRO**: Very similar to a full OS
- **PRO**: With current solutions, high OS diversity
- **CON**: Need of more space and resources
- **CON**: Slower than containers
- **CON**: Not as good automating

Containerisation

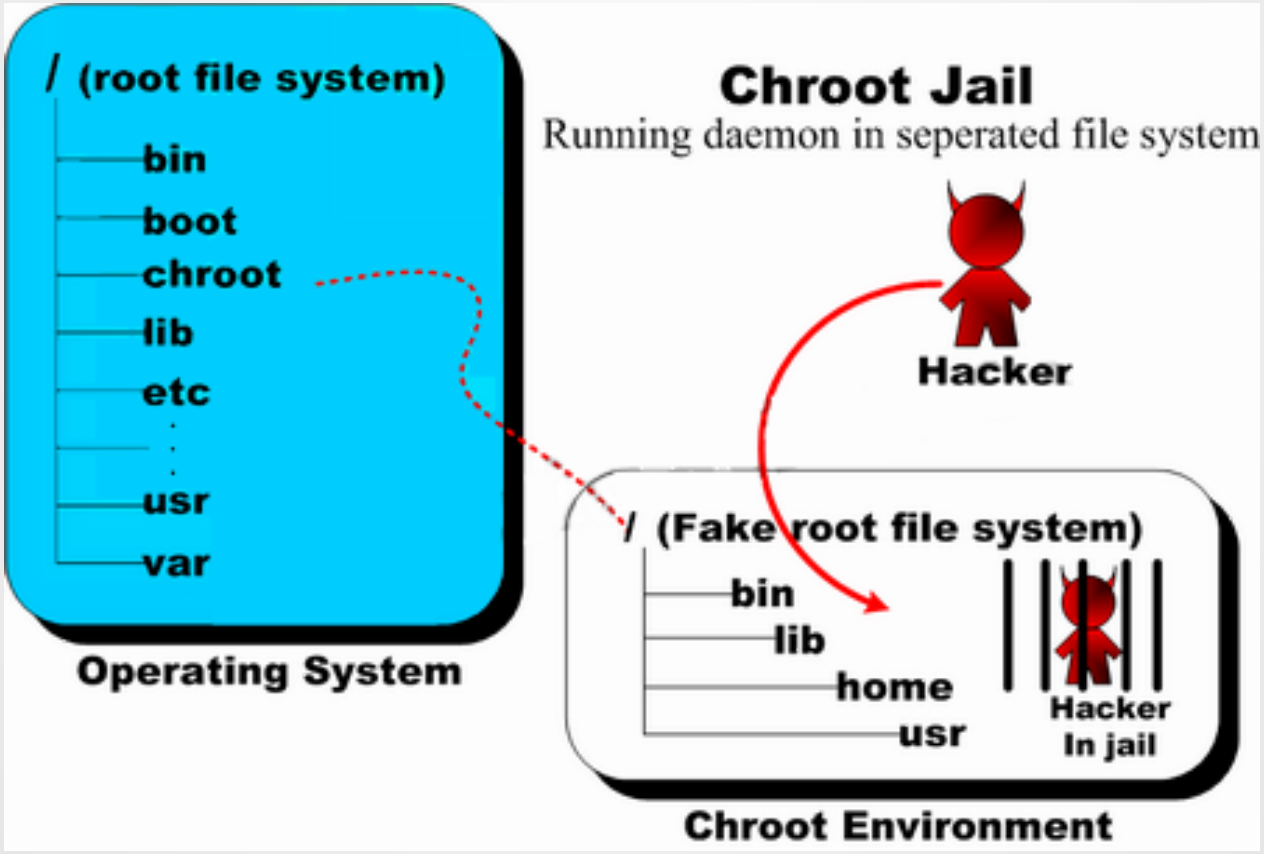
Pros and Cons

- **PRO**: Faster
- **PRO**: No need of full OS installation. Less space.
- **PRO**: Current solutions allow easier distribution of recipes. More portability
- **PRO**: Easier automation
- **CON**: Some cases might not be exactly the same as a full OS
- **CON**: With current solutions, still less OS diversity

History of containers

chroot

- chroot jail (BSD jail) - First concept 1979
- Notable use in SSH and FTP servers
- Honeypot, recovery of systems, etc.



https://sysopsio.wordpress.com/2016/09/09/jails-in-linux/

Additions in Linux kernel

- cgroups (control groups), before 'process containers'
  - isolates resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes
- Linux namespaces
  - one set of kernel resources restrict to one set of processes

Additions in Linux kernel

Linux Containers

Docker 1.10 and later

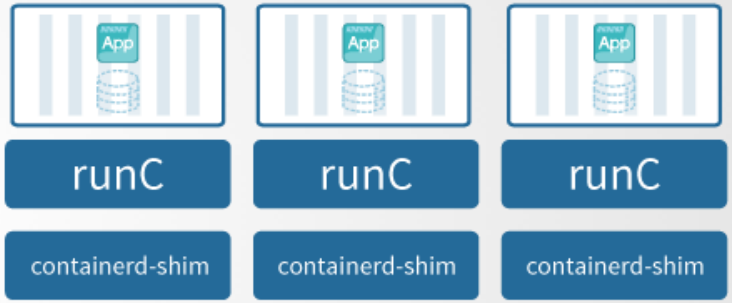
liblxc

namespaces

cgroups

SELinux/AppArmor

Linux kernel



containerd

Docker Engine

namespaces

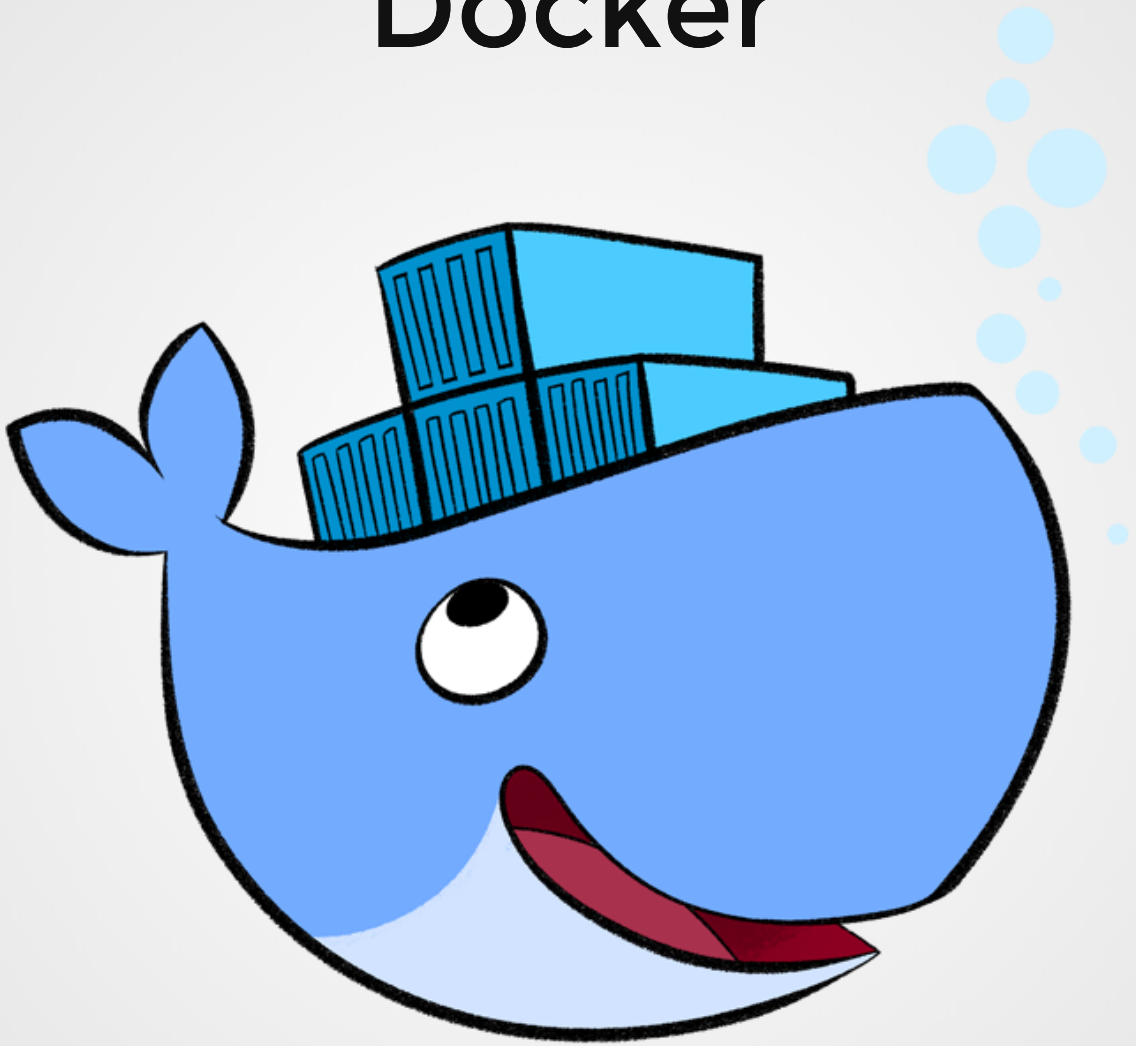
cgroups

SELinux/AppArmor

Linux kernel

https://robinsystems.com/blog/containers-deep-dive-lxc-vs-docker-comparison/

Docker



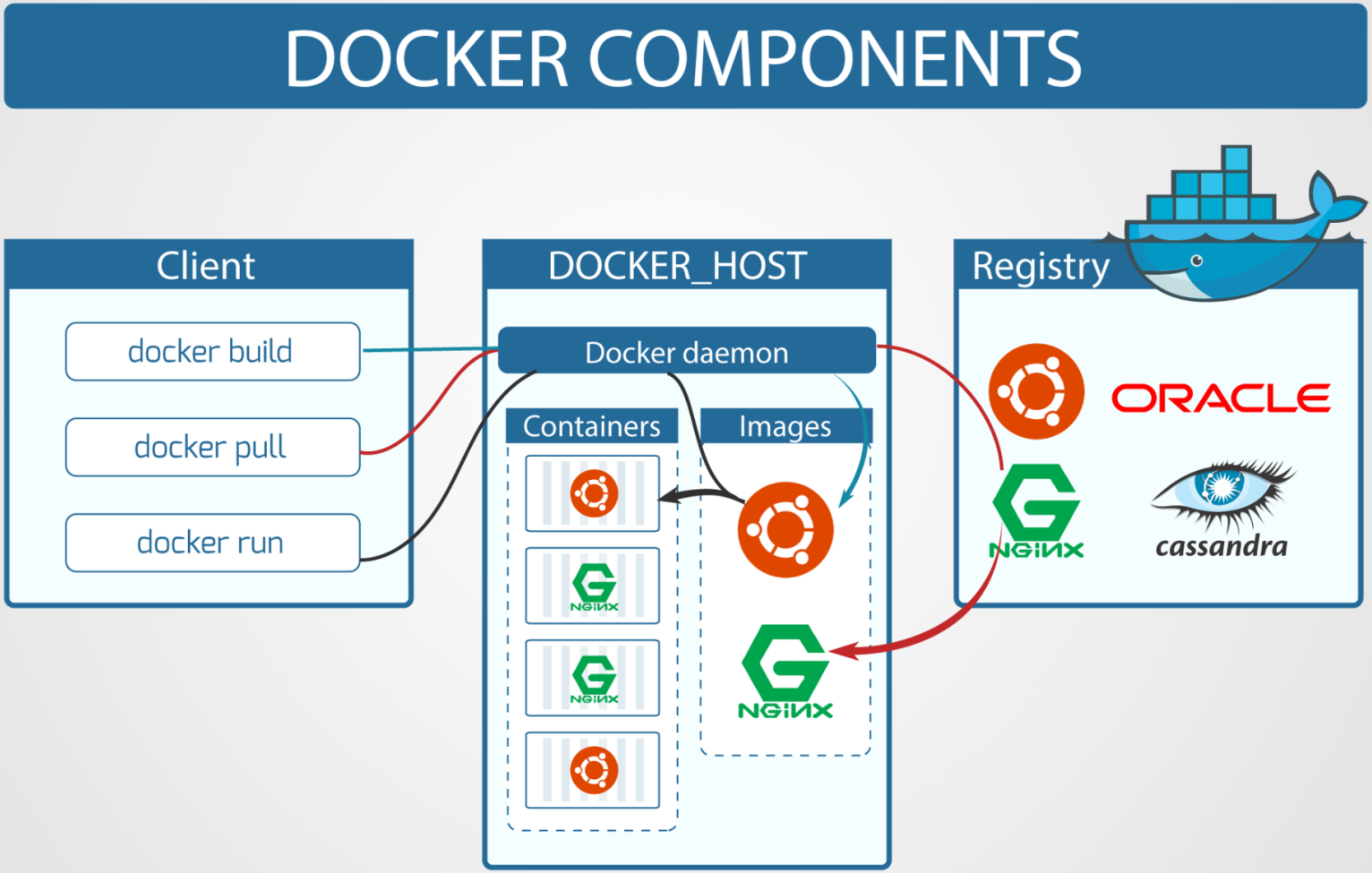
Docker

- Platform for developing, shipping, and running applications
- Infrastructure as application/code
- Established Open Container Initiative

As a software:

- Docker Community Edition
- Docker Enterprise Edition

Docker architecture



Docker Hub

Docker image

- Read-only templates.
- Containers are run from them
- Images are not run
- Images have several layers

Docker registry and Docker hub

- Images are stored locally
- They can also be shared in a registry
- Main Public one: Docker hub

Examples:

https://hub.docker.com/u/biocorecrg/

https://github.com/CRG-CNAG/docker-debian-perlbrew

Docker registry commands

download image

```
$ docker pull debian

$ docker pull debian:stretch

$ docker pull centos:7

$ docker pull rocker/r-ver:3.5.1

$ docker pull continuumio/miniconda:4.5.4
```

Debian

CentOS

Rocker R

Miniconda

Docker container

- Generated from an image (template)
- Image: read-only
- Container: read-write
- Can be converted into image
  - `docker commit`
  - `docker import & docker export`
- 1 imatge -> n diverse containers
  - Diversity:
    - Volumes / Mounting points
      - Different data or configs
    - Different exposed ports

Run container

```
$ docker run debian /bin/echo 'Hello world!'
$ docker run debian:jessie /bin/echo 'Hello world!'
$ docker run debian:jessie cat /etc/issue
$ docker run debian:stretch cat /etc/issue
```

Run container as daemon

Run daemon

```
$ docker run -d debian:stretch tail -f /dev/null
$ docker run -d --name mycontainer debian:stretch tail -f /dev/null
```

List runninng containers

```
$ docker ps
```

Log and info of container

```
$ docker log mycontainer
$ docker inspect mycontainer
```

Actions on containers

```
$ docker stop mycontainer
$ docker start mycontainer
$ docker restart mycontainer
```

More running on container

Execute on a running container

```
$ docker exec mycontainer /bin/echo 'Bye, moon!'
```

Run a container interactively (from the beginning). Stops when exiting

```
$ docker run -ti --name mycontainer2 debian:stretch /bin/bash
```

Execute on a running container interactively

```
$ docker exec -ti mycontainer /bin/bash
```

Other Docker commands

remove container, remove image

```
$ docker rm mycontainer
$ docker rmi myimage
```

list all containers (even stopped ones)

```
$ docker ps -a
```

save image as tar

```
$ docker save myimg > myimg.tar
```

```
$ docker export mycontainer > mycontainer.tar
```

```
$ docker import mycontainer.tar mycontainer:latest
```

import tar as image

Docker image - Building

- Can be built from existing base images
  - E.g. : Ubuntu, Apache
- Any modification from a base image is a new layer
- Base images can be generated with tools such as Debootstrap
  - Reminds of creation of a bootable (USB) Image

Docker image - Instructions

- Recipe file:
  - `Dockerfile`
- Instructions
  - *Every instruction generates an image layer*
  - FROM: use a base image (notice tag)
  - ADD, COPY: add files to image filesystem
  - RUN: execute command in image
  - ENV, ARG: Run and build environment variables
  - CMD, ENTRYPOINT: Command to execute when generated container starts

Docker image - Instructions

ADD vs COPY

in Stack Overflow

Docker image - Instructions

- Instructions
  - USER: User to run a following process (fallback: *root*)
  - WORKDIR: Location where following processes will be executed
  - VOLUME: Definition of mounting point (with host)
  - EXPOSE: Ports to be accessible from the container

Reference

Best Practices

Docker service mappings

Ports and volumes (directories/files)

host:container

```
docker run -d -v /db/test:/blastdb -p 8089:8081 --name myblast blastww
```

Docker image

Services approaches

ENTRYPOINT vs CMD

- Entrypoint script with or without appended command
- tini / --init
- supervisord

Containers in biosciences

Biocontainers: repository of bioinformatics containers

http://biocontainers.pro

https://dx.doi.org/10.1093/bioinformatics/btx192

nf-core

https://hub.docker.com/r/nfcore/

Docker registry commands

download image

```
$ docker pull biocorecrg/debian-perlbrew:stretch
```

upload image

```
$ docker push myusername/mysexyimage
```

Docker automated builds

Reference

Docker Compose

Reference

`docker-compose.yml`

Other Docker tools

Portainer

Dive

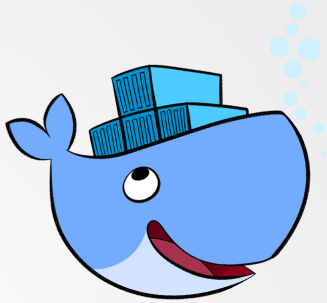
Singularity

containers for HPC

https://www.sylabs.io/singularity/

https://doi.org/10.1371/JOURNAL.PONE.0177459

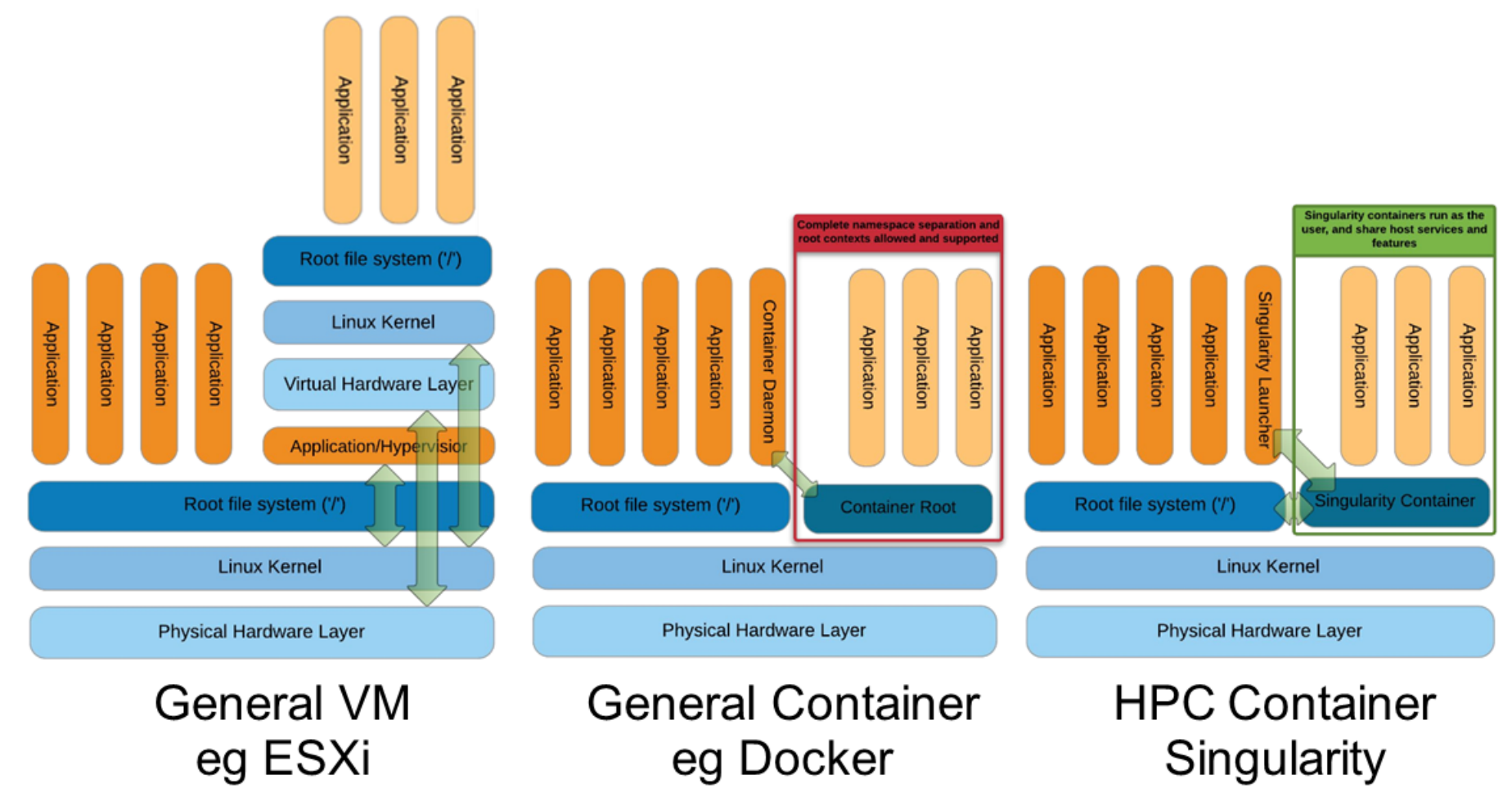
Singularity vs Docker



Summarising

- Docker -> Microservices
- Singularity -> HPC

Singularity architecture



http://geekyap.blogspot.com.es/2016/11/docker-vs-singularity-vs-shifter-in-hpc.html

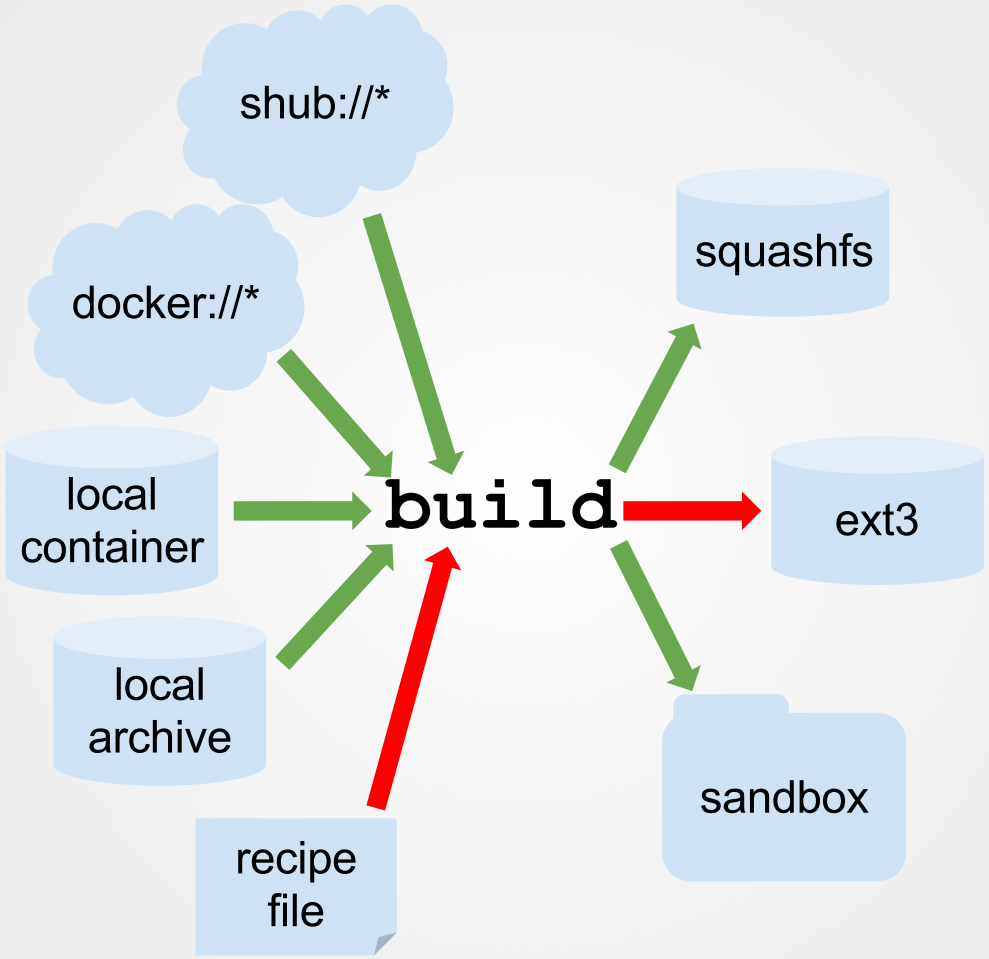
Singularity - Strenghts

- No dependency of a daemon
- Can be run as a simple user
- Image/container is a file (or directory)
  - More easily portable
- Two type of images
  - Read-only (production)
  - Writable (development)

Singularity - Weaknesses

- At the time of writing only good support in Linux
  - *Not a big deal in HPC environments, though*
- For some uses you might need root account (or sudo)

Singularity - build



As of 2.4.x version

http://singularity.lbl.gov/docs-build-container

Singularity - build

Build read-only image from Docker

```
$ singularity build debian.sif docker://debian:stretch
```

Build writable directory from Docker

```
$ sudo singularity build --sandbox debiandir docker://debian:stretch
```

Singularity - run

Execute a command

```
$ singularity exec debian.sif /bin/echo 'Hello world'
$ singularity exec debian.sif env
```

Execute a command (with clean environment)

```
$ singularity exec -e debian.sif /bin/echo 'Hello world';
$ singularity exec -e debian.sif env
```

Execute a shell

```
$ singularity shell debian.sif
```

Execute defined runscript (parameters can be used)

```
$ singularity run debian.sif
```

Private Docker images

https://www.sylabs.io/guides/3.2/user-guide/singularity_and_docker.html

```
sudo singularity build privateimg.sif docker-daemon://privateimg:latest
```

```
sudo singularity build privateimg.sif docker-archive://privateimg.tar
```

Singularity recipes

```
BootStrap: debootstrap
OSVersion: stretch
MirrorURL:  http://ftp.fr.debian.org/debian/
Include: curl

%environment
        BLAST_PROGRAM=blastp
        BLASTDB=/blastdb
        export BLAST_PROGRAM BLASTDB

%post

        BLAST_VERSION=2.7.1

        cd /usr/local; curl --fail --silent --show-error --location --remote-name ftp://ftp.ncbi.nlm.nih.gov/blas
        cd /usr/local; tar zxf ncbi-blast-${BLAST_VERSION}+-x64-linux.tar.gz; rm ncbi-blast-${BLAST_VERSION}+-x64
        cd /usr/local/bin; ln -s /usr/local/ncbi-blast-${BLAST_VERSION}+/bin/* .

%labels
        Maintainer Biocorecrg
        Version 0.1.0

%runscript
        echo "Blast application!"
        exec $BLAST_PROGRAM "$@"
```

http://singularity.lbl.gov/build-debootstrap

Singularity recipes

```
BootStrap: docker
From: continuumio/miniconda:4.5.4

%environment
        PATH=/opt/conda/envs/blast-conda/bin:$PATH
        export PATH

%post
        PATH=/opt/conda/bin:$PATH
        export PATH
        conda env create -f /environment.yml && conda clean -a

%files
        environment.yml

%labels
        Maintainer Biocorecrg
        Version 0.1.0
```

http://singularity.lbl.gov/docs-recipes

Singularity - volumes

```
singularity run -B /db/test:/blastdb blastwww.sif
```

Singularity - services/instances

```
sudo singularity build blastmysql.sifSingularity

singularity run -B /db/test:/blastdb -B $(pwd)/config.json:/config/mysql.json blastmysql.sif

sudo singularity build mysql.sif Singularity.mysql

singularity exec -B /tmp/db:/var/lib/mysql mysql.sif mysql_install_db

singularity instance.start -B /tmp/db:/var/lib/mysql -B /tmp/socket:/run/mysqld mysql.sif mysql

singularity instance.list

singularity exec instance://mysql mysql -uroot -h127.0.0.1 -e "CREATE DATABASE blast; GRANT ALL PRIVILEGES

singularity instance.start -B /db/test:/blastdb -B $(pwd)/config.local.json:/config/mysql.json blastmysql.

singularity instance.list

singularity instance stop.blast

singularity instance.stop mysql
```

Further reading

- The impact of Docker containers on the performance of genomic pipelines.
- Performance Evaluation of Container-based Virtualization for High Performance Computing Environments

Sum-up:
containers in science

- Mantainability
- Portability
- Reproducibility

Play!

https://github.com/biocorecrg/containers-course