Lightweight Virtualization

LIP Computing

INCD

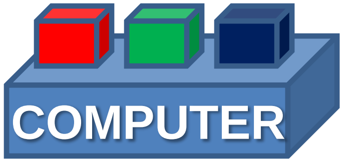
Jorge Gomes <jorge@lip.pt>

INDIGO - DataCloud
Better Software for Better Science

Cofinanciado por:
COMPETE 2020
PROGRAMA OPERACIONAL COMPETITIVIDADE E INTERNACIONALIZAÇÃO
PORTUGAL 2020
UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento Regional
Lisb@20 20
FCT Fundação para a Ciência e a Tecnologia

Virtualization many types ...

Type	Some examples
Network Virtualization:	*VLANs, vswitches, …*
Storage Virtualization:	*Logical Volumes, …*
Computer Virtualization:	*Virtual Machines, …*
Operating System Virtualization:	*Containers, …*

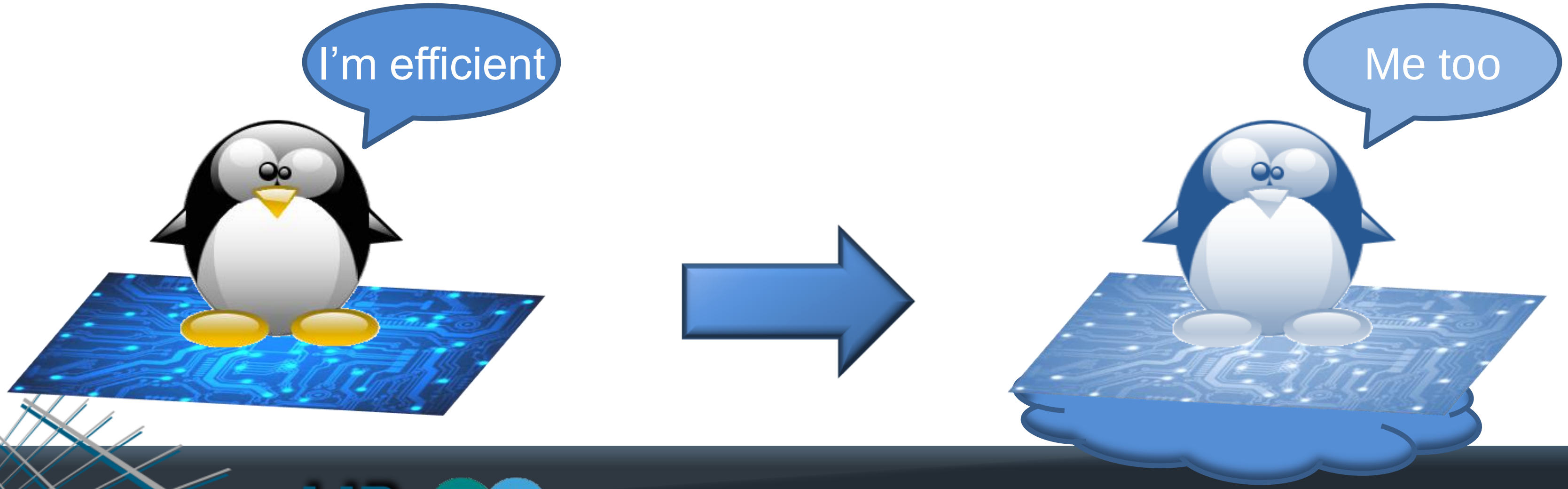


Virtual Machine

"Virtual Machine an efficient, isolated duplicate of a real computer machine."

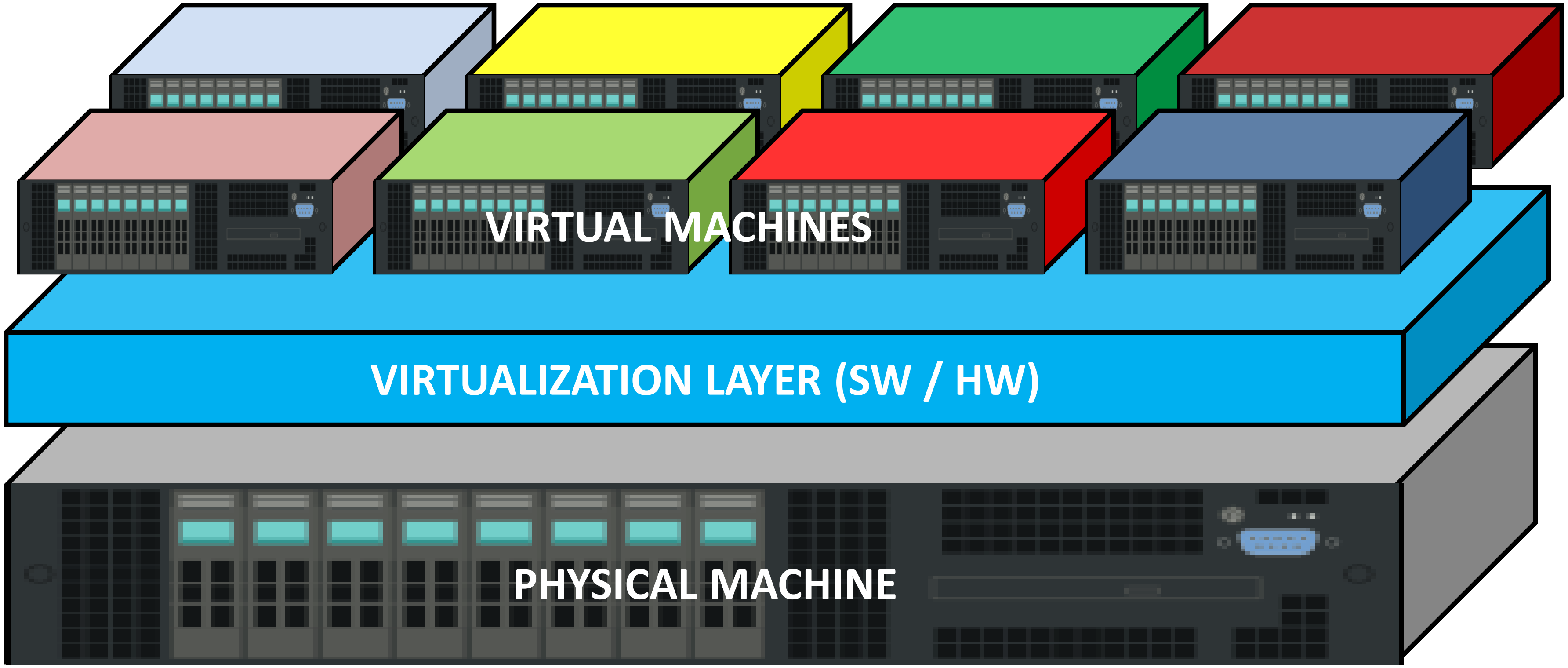
Formal Requirements for Virtualizable Third Generation Architectures (1974)

Gerald J. Popek and Robert P. Goldberg



Computer virtualization

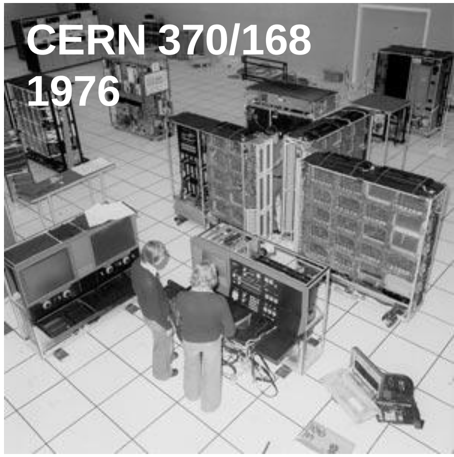
We are going to focus on Virtual Machines (VM).

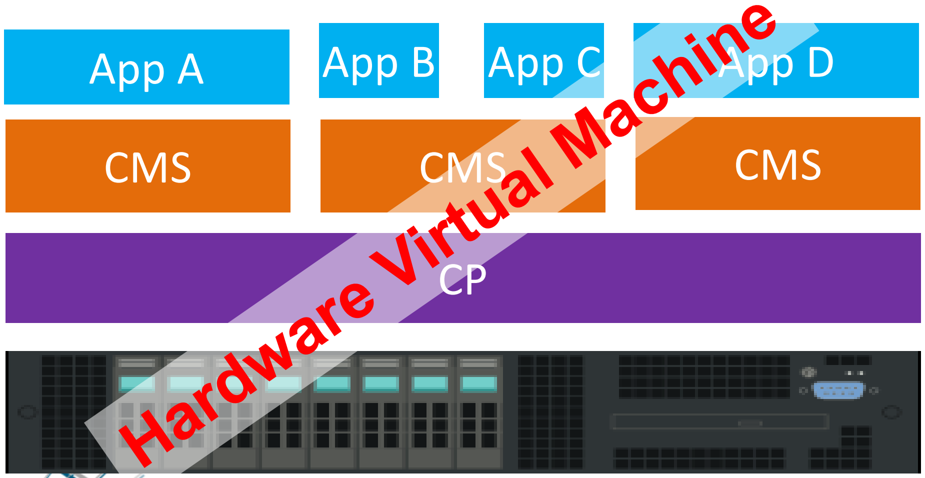


History

- 1966 CP/40 for S/360-40
  - research project, introduced CP and CMS
  - first full virtualization capable system
- 1967 IBM **CP/CMS** for S/360-67
  - first virtualization in production
- 1972 IBM **VM/370** => CP/CMS for IBM S/370

IBM S/360 Model 67-2
1969


CERN 370/168
1976



Applications

Cambridge Monitor System (CMS)

Control Program(CP) ➔ hypervisor

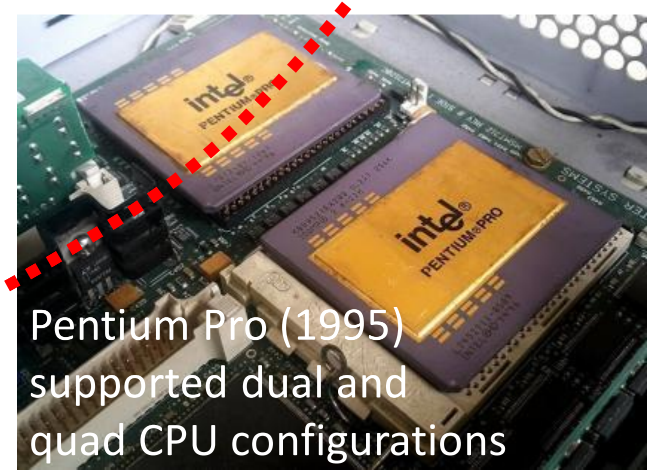
Mainframe Hardware

History

- Late 90’s the microprocessors become more powerful and multiprocessor machines (SMP) cheap.
- A single microprocessor based machine could now support multiple services and/or applications.

- Virtualization gained interest again.

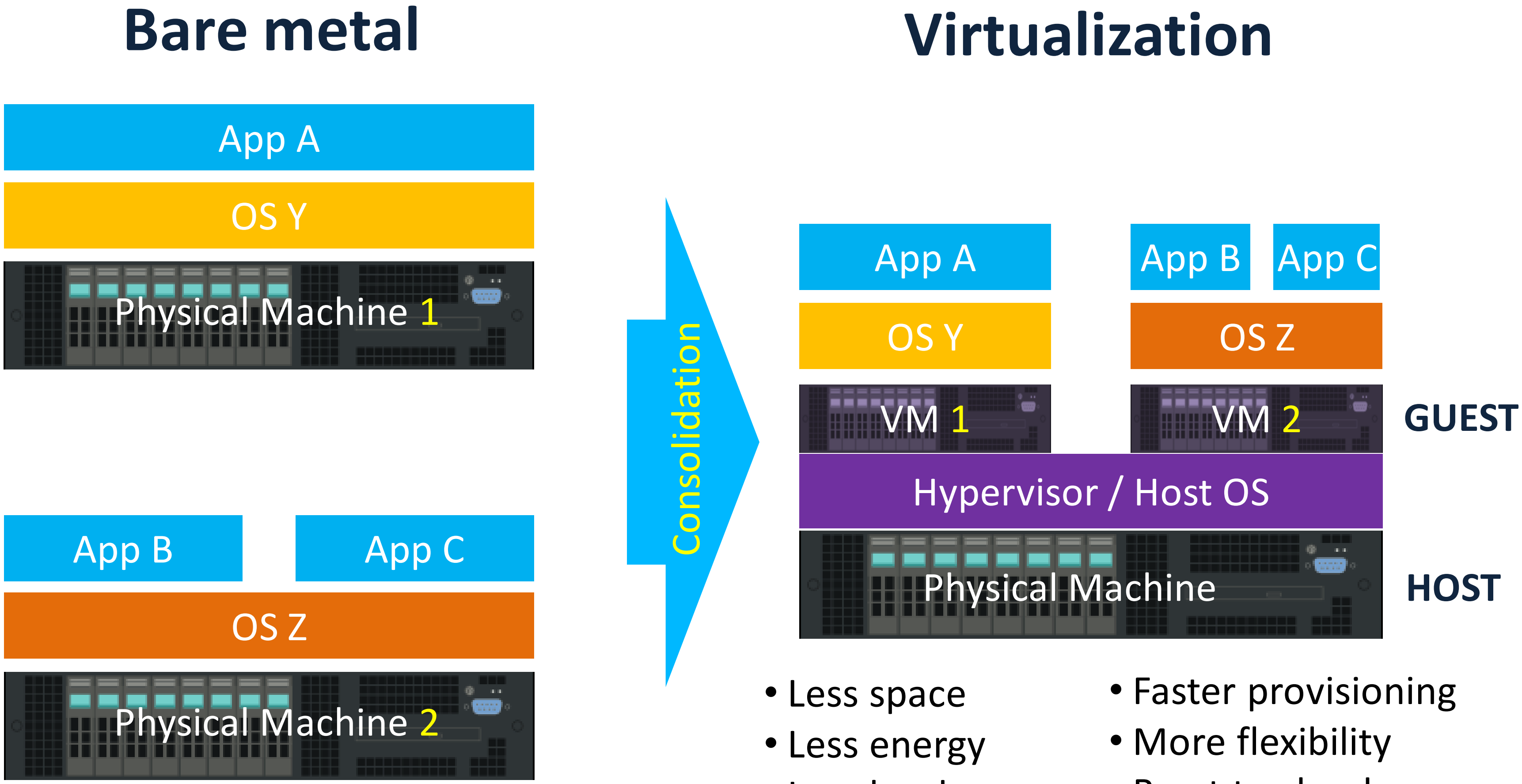
- **1999 VMware workstation**
- **2000 User Mode Linux (UML)**
- **2003 Xen for Linux**



Pentium Pro (1995) supported dual and quad CPU configurations

Types of Virtualization

Bare metal vs Virtualization

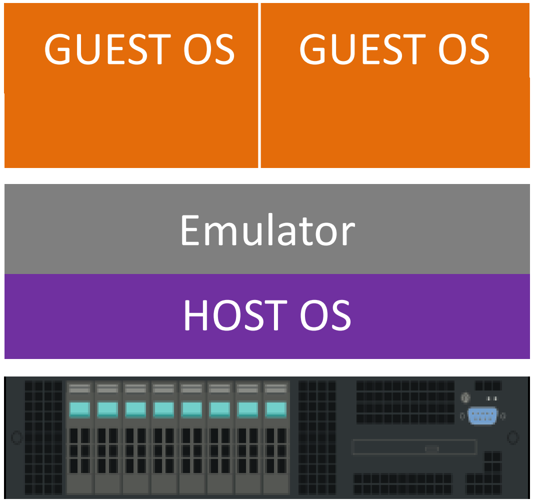


- Less space
- Less energy
- Less hardware
- Easier to manage

- Faster provisioning
- More flexibility
- Burst to cloud

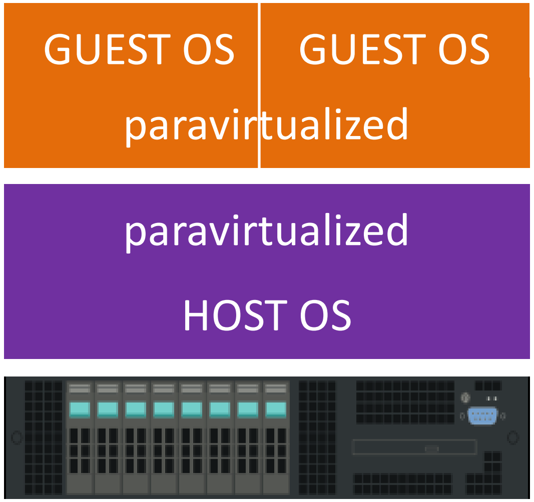
Common types of virtualization

Emulation



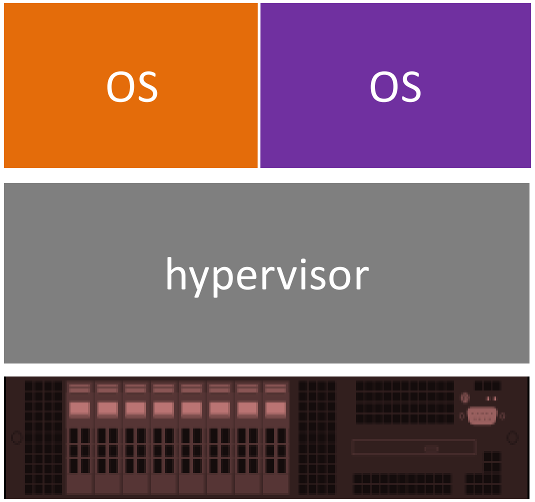
- •Both kernels unchanged
- •Emulated hardware
- •Ex. QEMU

Paravirtualization



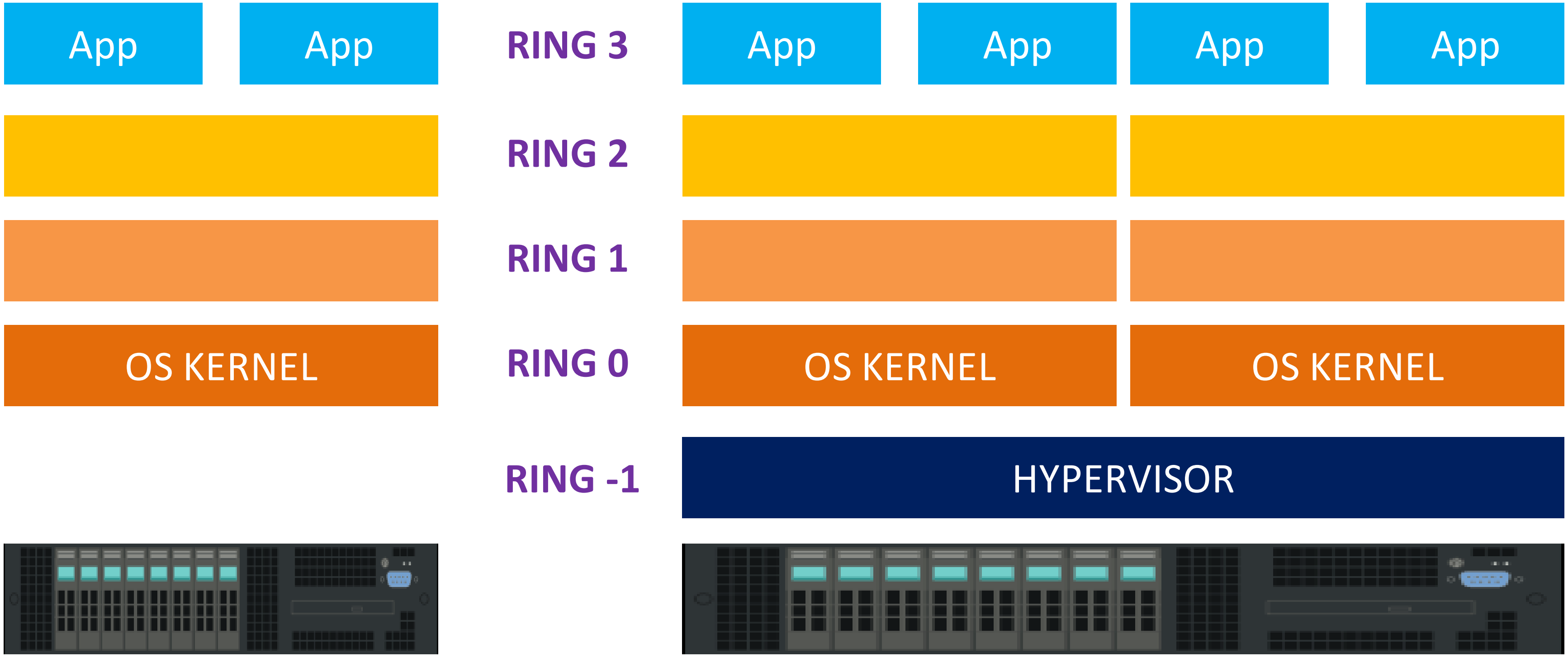
- •Both kernels changed
- •Emulation replaced by hypercalls to the host
- •Ex. Xen

Hardware assisted virtualization



- •Both kernels unchanged
- •Emulation replaced by hardware assisted hypervisor
- •Ex. KVM

Rings and hardware virtualization



- Rings are hierarchical protection domains within the CPU
- Lower rings have higher privileges in the processor
- Intel VT-x and AMD-V add a ring -1 for hypervisors

Operating System Level Virtualization

a.k.a Containers

Why do we need hypervisors ?

Use different operating system implementations in the same physical machine (eg. Linux and Windows simultaneously)	**X**
Limit security breaches (isolation between applications or operating systems)	**X**
Better resource allocation and consumption control (memory, CPU, IO bandwidth, etc)	**X**
Flexible infrastructure (easier provisioning, capacity and resource management in large facilities)	**X**
Use same OS or very similar but with different system environments customized for several applications	**X**

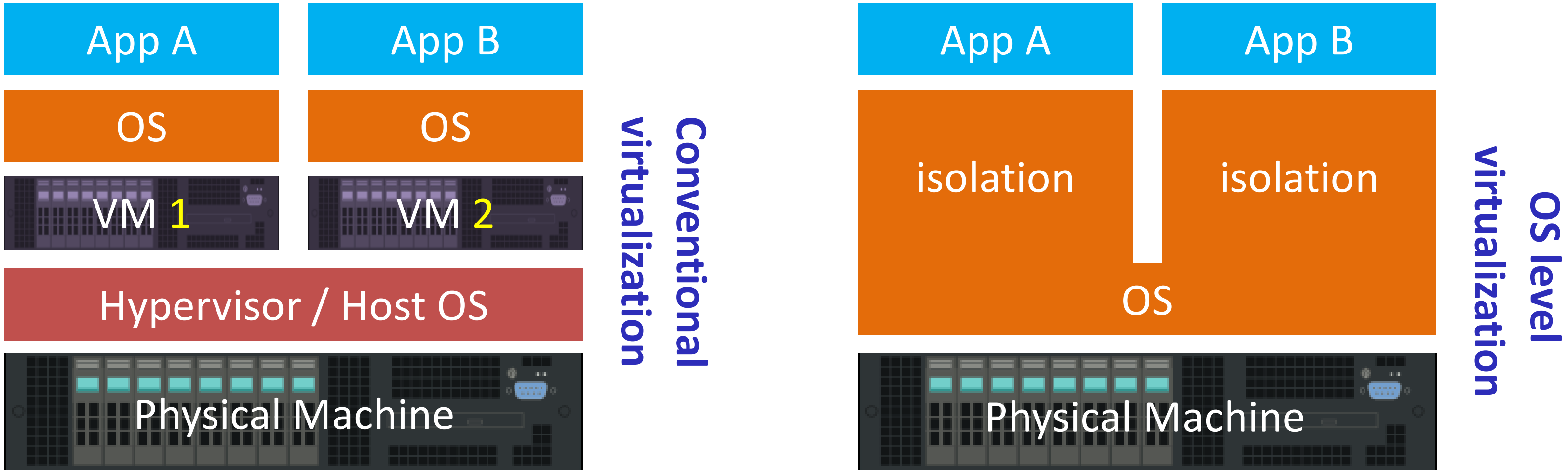
We need VM hypervisors because OSes are not capable…

> *“hypervisors are the living proof of operating system's incompetence.”*

The Failure of Operating Systems & How We Can Fix It

Depending on the purpose hypervisors and virtual machines can be the wrong tool for the job ...

Operating system level virtualization



- Multiple environments via OS isolation features
- OS can limit what processes can do and see
- Same OS kernel is shared and directly used
- More efficient than VMs
- Only for OSes with same kernel (ex. Ubuntu and CentOS)

OS level virtualization advantages

- Less memory consumption
  - No need of duplicated kernels and related processes
  - No duplication of buffering and shared memory
  - Less memory split across execution domains

- Faster I/O and execution and less latency
  - Direct execution on top of one single kernel
  - No emulation, No hypercalls, No buffer copies

- Don’t need to run OS services in each isolated environment
  - No need of duplicated NTP, SNMP, CRON, DHCP, SYSLOG, SMART, etc

- Much faster start–up times
  - No OS boot, smaller images to transfer and store

- Less management effort
  - Only the host machine needs to be managed (many-core is great)

OS level virtualization also not new

		Year	File system isolation	I/O limits	Memory limits	CPU quotas	Network isolation	Root priv isolation
chroot	Most unix systems	1982	X					
Jail	FreeBSD	1998	X	X	X	X	X	X
Linux-VServer	Linux	2001	X	X	X	X	X	X
Virtuozzo Containers	Linux Windows	2001	X	X	X	X	X	X
Zones	Solaris	2004	X	X	X	X	X	X
OpenVZ	Linux	2005	X	X	X	X	X	X
HP Containers	HP/UX	2007	X	X	X	X	X	
LXC	Linux	2008	X	X	X	X	X	X
Docker	Linux	2013	X	X	X	X	X	X

Wikipedia, The Free Encyclopedia. Wikimedia Foundation

Linux kernel features

- **Kernel namespaces**: isolate system resources from process perspective
  - **Mount** namespaces: isolate mount points
  - **UTS** namespaces: hostname and domain isolation
  - **IPC** namespaces: inter process communications isolation
  - **PID** namespaces: isolate and remap process identifiers
  - **Network** namespaces: isolate network resources
  - **User** namespaces: isolate and remap user/group identifiers
  - **Cgroup** namespaces: isolate Cgroup directories
- **Seccomp**: system call filtering
- **Cgroups**: process grouping and resource consumption limits
- **POSIX capabilities**: split/enable/disable root privileges
- **chroot**: isolated directory trees
- **AppArmor** and **SELinux**: kernel access control

Namespaces

```
$ ls  -l  /proc/$$/ns
total 0
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 cgroup -> cgroup:[4026531835]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 net -> net:[4026531993]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 pid -> pid:[4026531836]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 pid_for_children -> pid:[4026531836]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 user -> user:[4026531837]
lrwxrwxrwx 1 jorge jorge 0 Dez  5 21:02 uts -> uts:[4026531838]
```

You are already using them !

Container

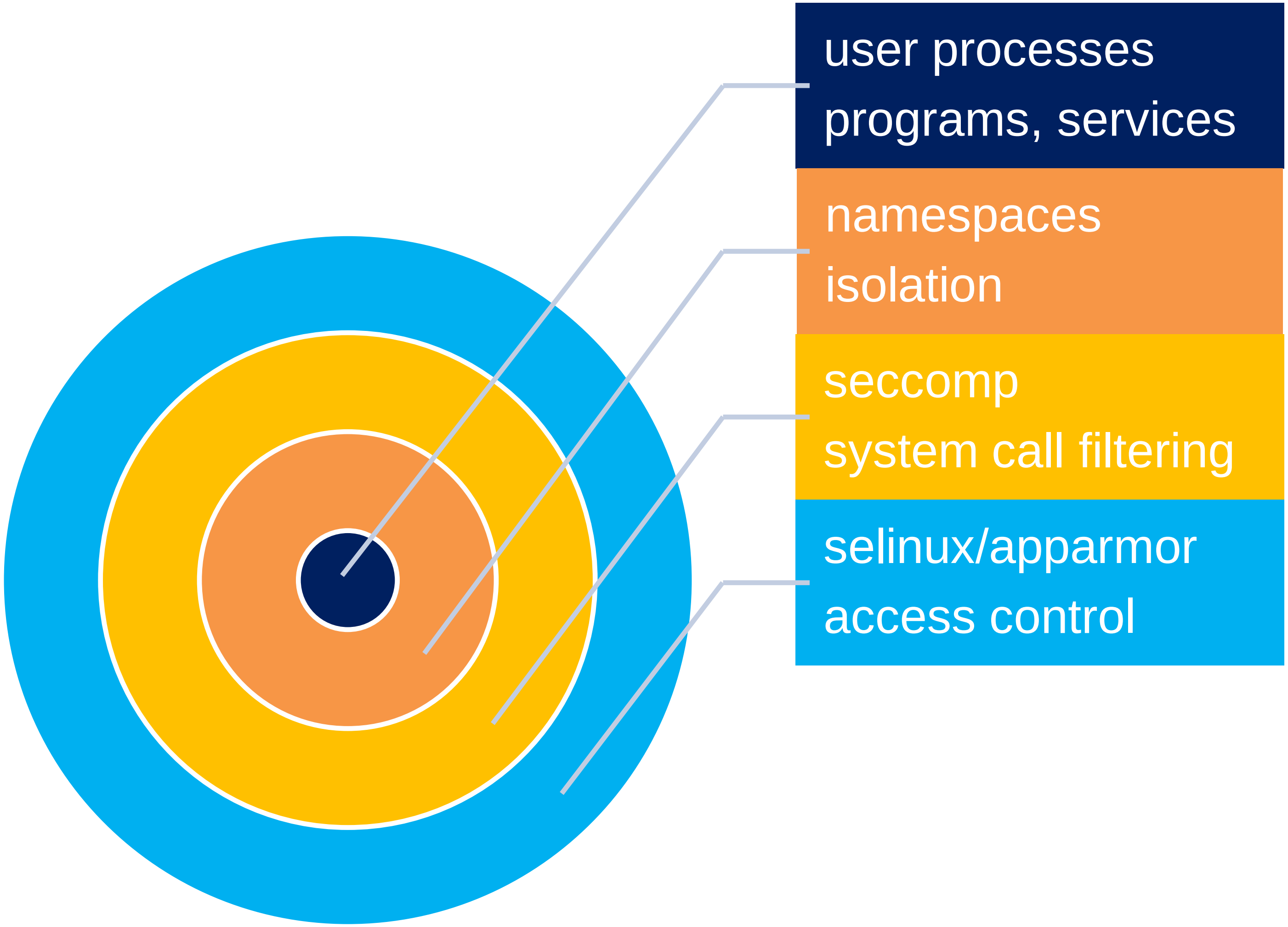
Runs programs as processes in a standard way

No emulation or hypervisors

Just process isolation

Therefore much more efficient

Containers and isolation



user processes
programs, services

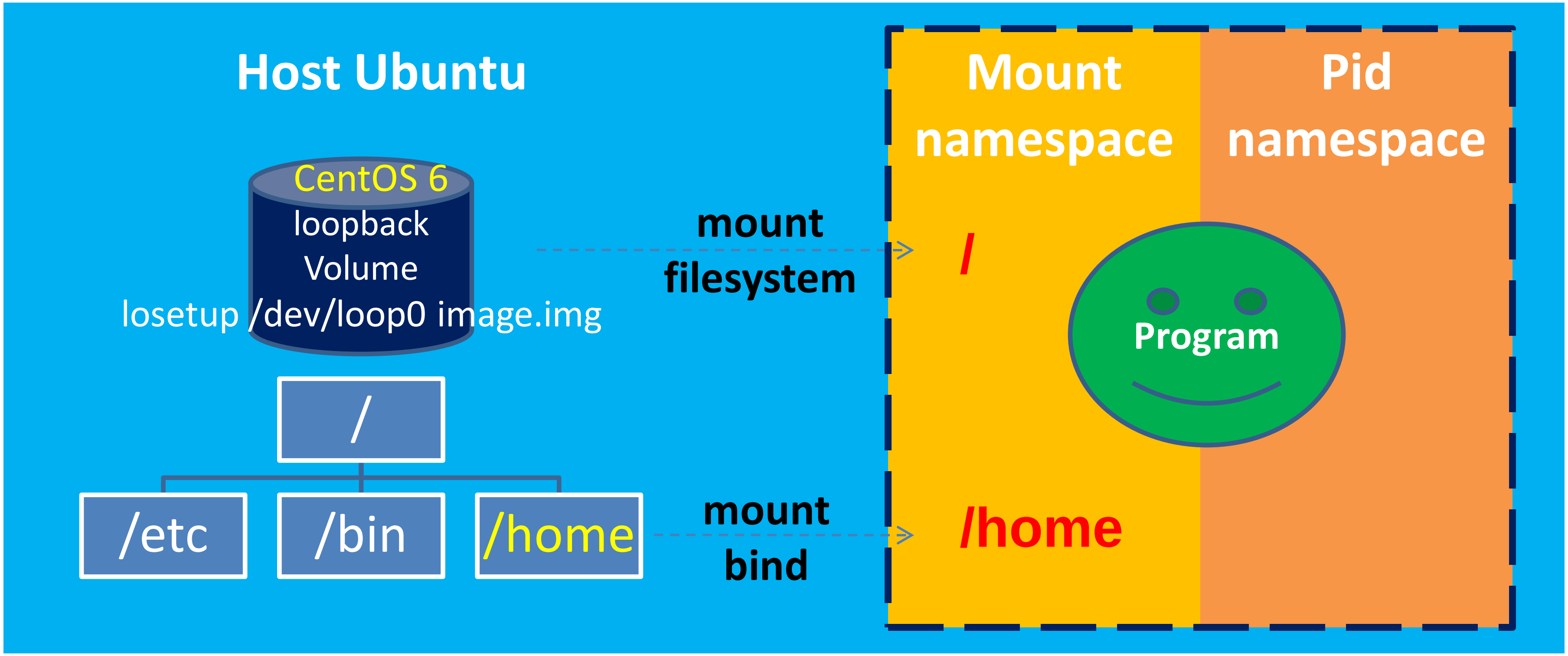
namespaces
isolation

seccomp
system call filtering

selinux/apparmor
access control

```
losetup /dev/loop0 example.img mount /dev/loop0 /home/you/dir
```

Container putting it together



Host Ubuntu

CentOS 6
loopback
Volume

losetup /dev/loop0 image.img

/

/etc

/bin

/home

mount
filesystem

mount
bind

Mount
namespace

Pid
namespace

/

Program

/home

Container putting it together

To create a container image:

- Add the required OS libraries, OS commands to the container
- Add the required user programs and data to the container

Can I run another Linux distribution using containers ?

- **Yes sure**
- **The Linux kernel ABI remains largely unchanged across versions**

Containers are usually started by the root user:

- Some operations require privileges
- Can be root user inside a container without affecting the host or the other containers (with POSIX capabilities, seccomp and namespaces)

LXC/LXD

Linux Containers project (LXC)

- First open source project to provide a toolset for containers

- Create and manage containers using the Linux Kernel features:
  - liblxc library
  - Bindings for several languages (python, ruby, lua, Go)
  - Templates
  - Tools to create/manage containers
- Tools:
  - lxc-create, lxc-destroy, lxc-start, lxc-stop, lxc-execute, lxc-console,
  - lxc-monitor, lxc-wait, lxc-cgroup, lxc-ls, lxc-ps, lxc-info, lxc-freeze,
  - lxc-unfreeze

- Limitations:
  - Requires considerable knowledge and effort

LXD

- Newer development from the original Linux Containers project
- Pushed and supported by Canonical (Ubuntu)

- Objective:
  - Provide an environment to run complete Linux OS distributions within containers
  - More similar to an hypervisor but using namespaces
  - **“boot” the almost complete OS distribution**
  - Images are tarballs

- Limitations:
  - Limited support and adoption beyond Ubuntu
  - Fairly recent

docker

Docker

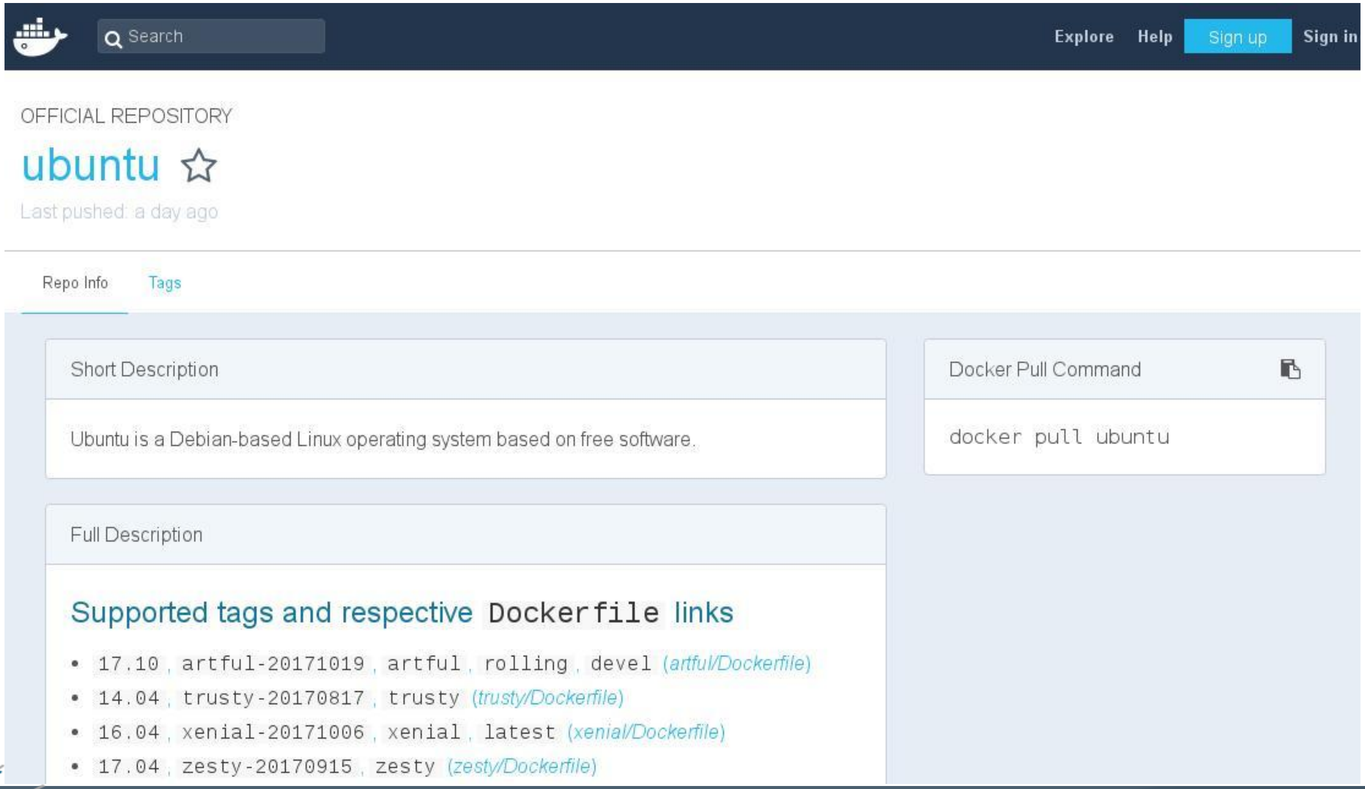
- **Docker containers are oriented to services composition:**
  - (Services or Applications) + (runtime environment)
  - Self-contained and lightweight
  - **Run it everywhere** (Linux)

- **DevOps ➔ integration of IT development and operations**
  - DevOps requires strong automation
  - Developers: focus on what's inside the container
  - Operations: may focus in the underlying infrastructure

```
# docker  run  -i  -t  centos:centos6
[root@28f89ada747e /]#  cat  /etc/redhat-release
CentOS release 6.8 (Final)
```

Docker

- Docker images can be fetched from the Docker Hub repository
  - There are other Docker container repositories besides Docker Hub
  - Very convenient to transfer and share containers pull/push



OFFICIAL REPOSITORY

ubuntu ☆

Last pushed: a day ago

Short Description

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

Docker Pull Command

```
docker pull ubuntu
```

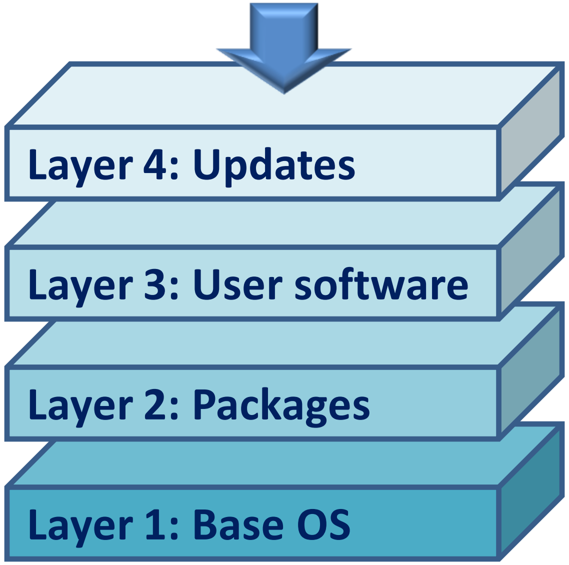
Full Description

Supported tags and respective `Dockerfile` links

- `17.10`, `artful-20171019`, `artful`, `rolling`, `devel` (*artful/Dockerfile*)
- `14.04`, `trusty-20170817`, `trusty` (*trusty/Dockerfile*)
- `16.04`, `xenial-20171006`, `xenial`, `latest` (*xenial/Dockerfile*)
- `17.04`, `zesty-20170915`, `zesty` (*zesty/Dockerfile*)

Docker

- Docker container image is composed of:
  1. Multiple file-system layers each one:
     1. metadata
     2. tarball with the files for the layer
  2. Manifesto
  3. Ancestry

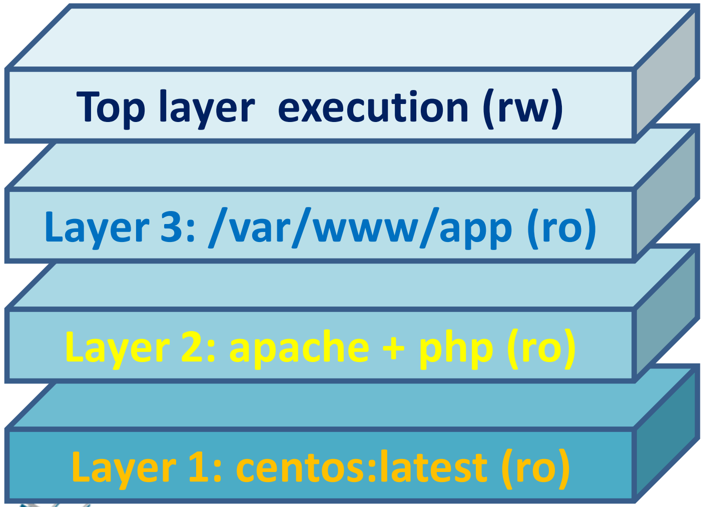


- Layers have unique ids and can be shared by multiple images
- Layers decrease storage space and transfer time
  - e.g. the same OS layer can be shared by many services and applications, avoiding duplication and downloading

Docker

- **Common format to distribute and manage images**:
  - Layered file-system based
    - At the host level implemented by AUFS, device-mapper thin snapshots
  - New images can be easily created from existing ones
    - Created by using **Dockerfiles** and docker build

Layers



Top layer execution (rw)

Layer 3: /var/www/app (ro)

Layer 2: apache + php (ro)

Layer 1: centos:latest (ro)

Dockerfile

```
FROM  centos:centos6
RUN  yum  install  –y  httpd php
COPY  /my/app  /var/www/app
EXPOSE   80
ENTRYPOINT  /usr/sbin/httpd
CMD [“-D”,  “FOREGROUND”]
```

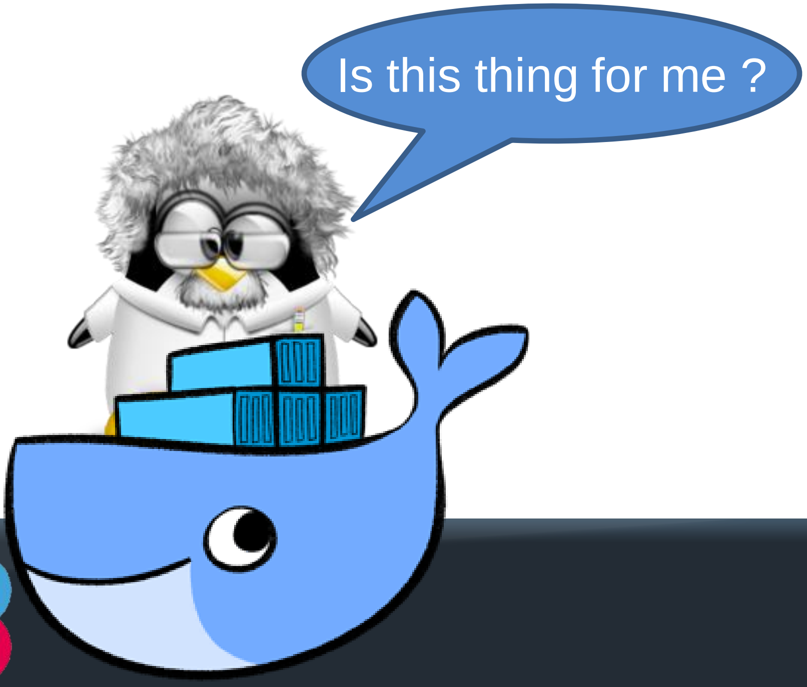
Docker in numbers

- **DockerCon conference 2017 (> 5500 attendees)**

- More than 14M Docker hosts
- More than 900K Docker apps in repositories
- 77,000% growth in Docker job listings
- More than 12B image pulls (accounting for 390,000% growth)
- More than 3,300 contributors
- More than 280 cities hold Docker meetups, which accounts for more than 170K members worldwide

- **Large ecosystem of tools and frameworks**

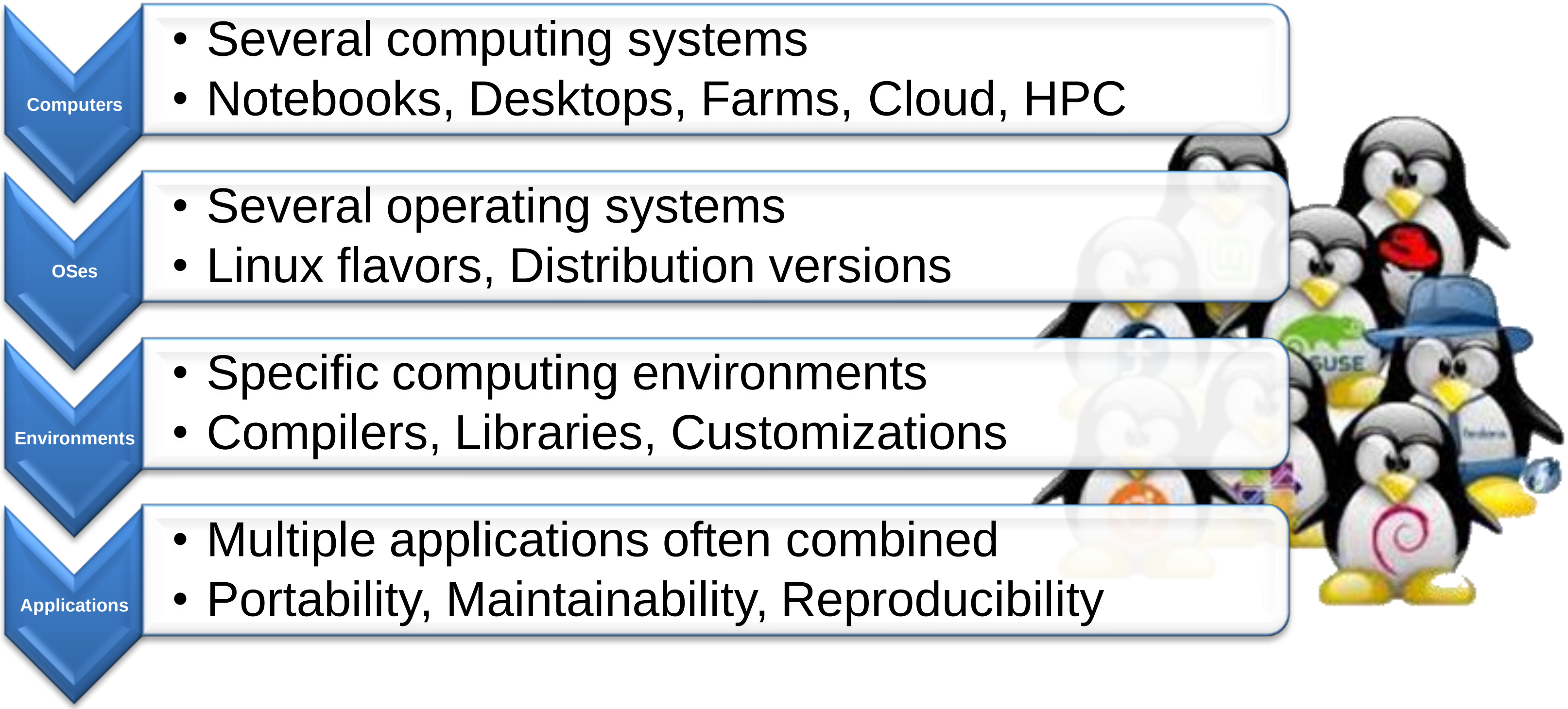
Scientific Computing and containers



Jorge Gomes

Scientific computing and containers

Running applications still requires considerable effort



Computers

- Several computing systems
- Notebooks, Desktops, Farms, Cloud, HPC

OSes

- Several operating systems
- Linux flavors, Distribution versions

Environments

- Specific computing environments
- Compilers, Libraries, Customizations

Applications

- Multiple applications often combined
- Portability, Maintainability, Reproducibility

Need a consistent portable way of running applications

but ...

Limitations

Require root privileges to install, setup and run

- Security concerns especially in multi-user environments

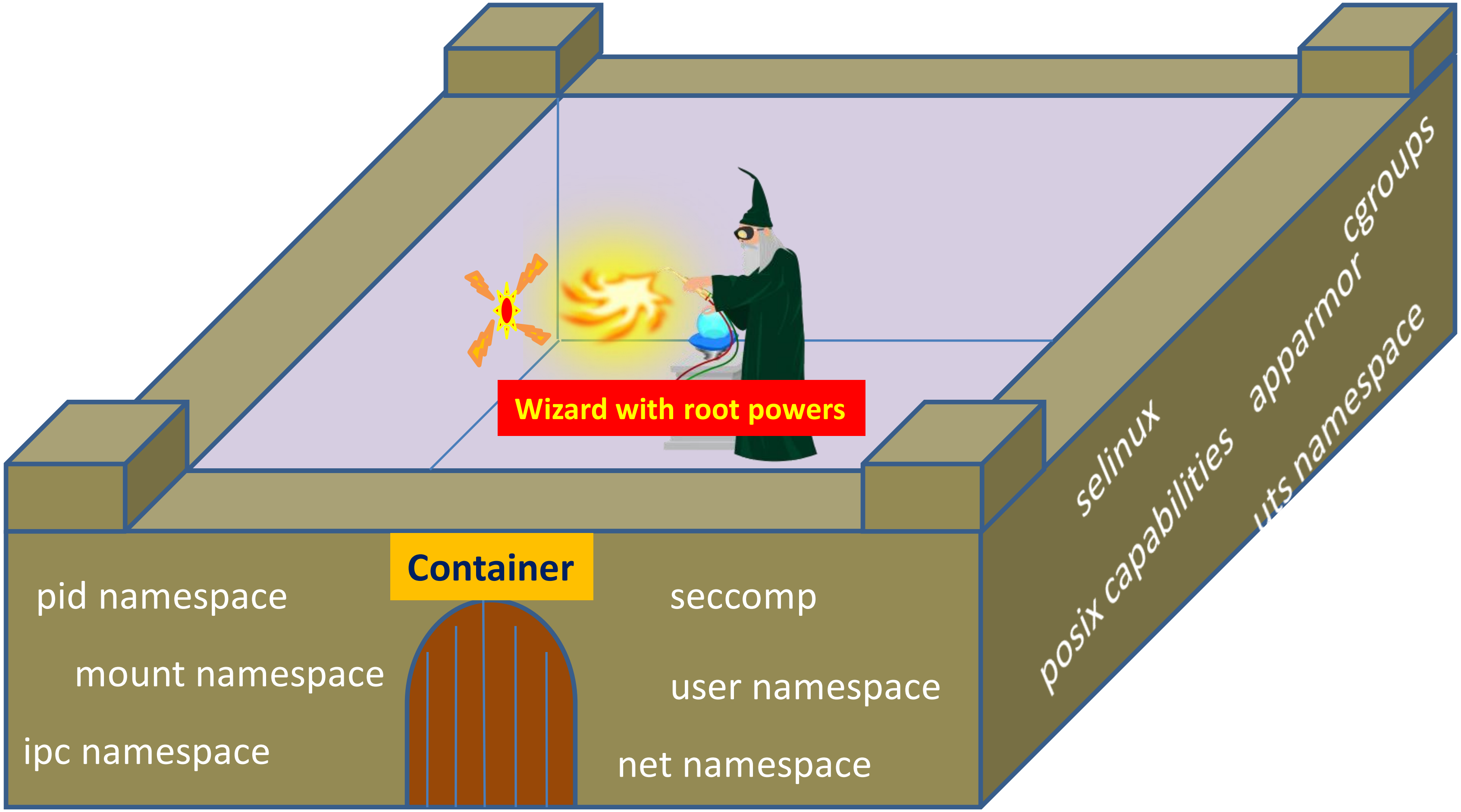
Docker API does not limit privileged actions

- Users with direct access to the API can do anything
- e.g: through the API users can mount local file systems, make devices accessible, erase disks etc.

Limiting design decisions for end users

- Docker is designed to be used as an hypervisor by operators
- Difficult to use on batch systems because of process control and security (not suitable)

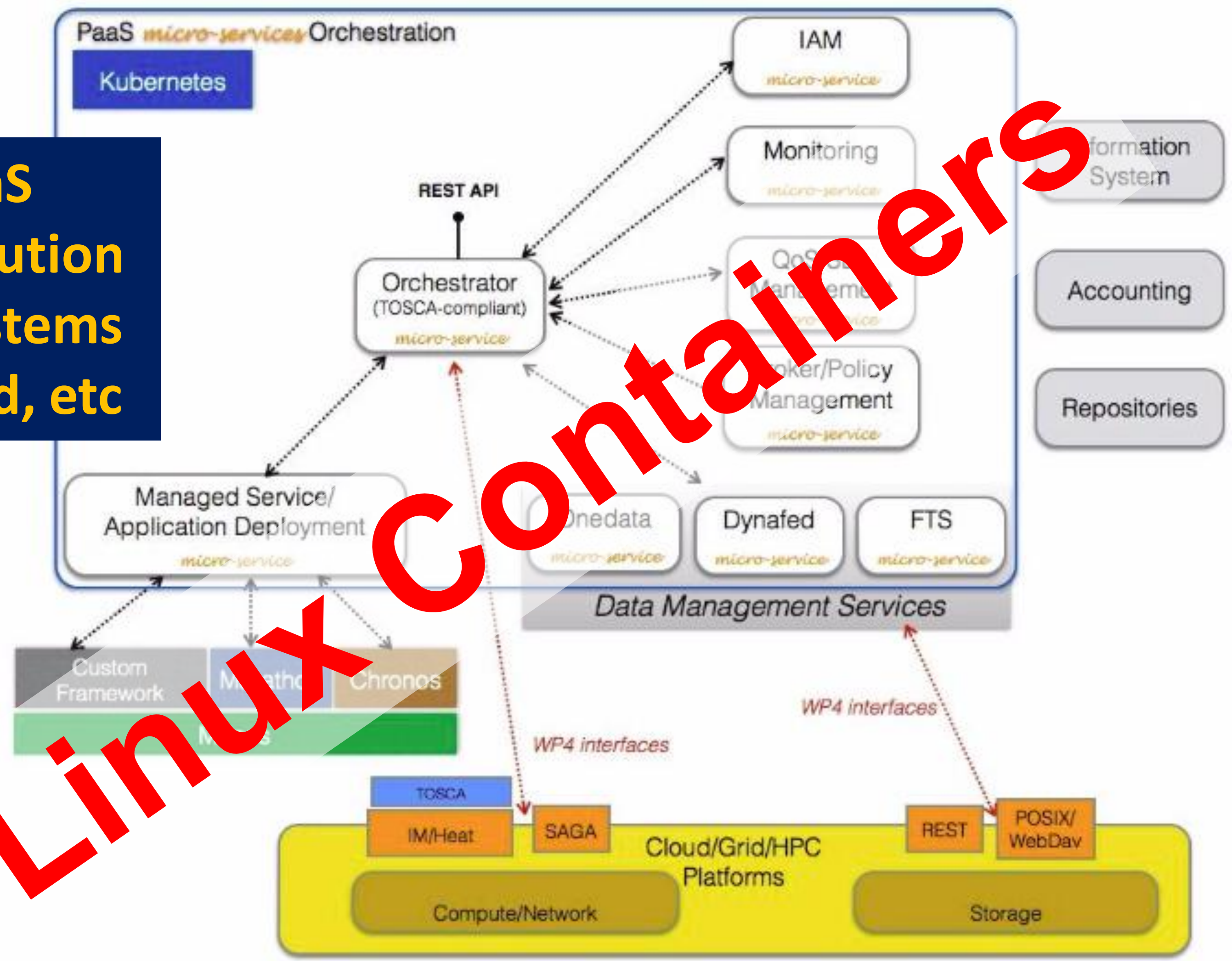
Containers in general ...



udocker

INDIGO-DataCloud H2020 (**2015**-2017)

Cloud PaaS easy execution across systems cloud, grid, etc



INDIGO-DataCloud containers for batch

- How to run Docker in batch systems ?
  - Can we run Docker in batch system ?
  - If so how to integrate it with the batch system ?
  - How to make it respect batch system policies ?
  - How to make it respect batch system actions ?
  - How to collect accounting ?

bdocker

- How to run containers without Docker ?
  - Can we download container images ?
  - Can we run without a layered filesystem ?
  - Can we run them as normal user ?
  - Can we enforce container metadata ?

udocker

udocker

- Run applications encapsulated in docker containers:
  - without using docker
  - without using privileges
  - without system administrators intervention
  - without additional system software

- and run:
  - as a normal user
  - with the normal process controls and accounting
  - in interactive or batch systems

INDIGO-DataCloud **udocker**

udocker in open source

https://github.com/indigo-dc/udocker

- https://github.com/indigo-dc/udocker/tree/master
- https://github.com/indigo-dc/udocker/tree/devel

https://github.com/indigo-dc/udocker/tree/master/doc

udocker: install from github

```
$ curl https://raw.githubusercontent.com/indigo-dc/udocker/master/udocker.py > udocker

$ chmod u+rx udocker

$ ./udocker install
```

or devel

Does not require compilation or system installation
Tools are delivered statically compiled

udocker: pull images from repository

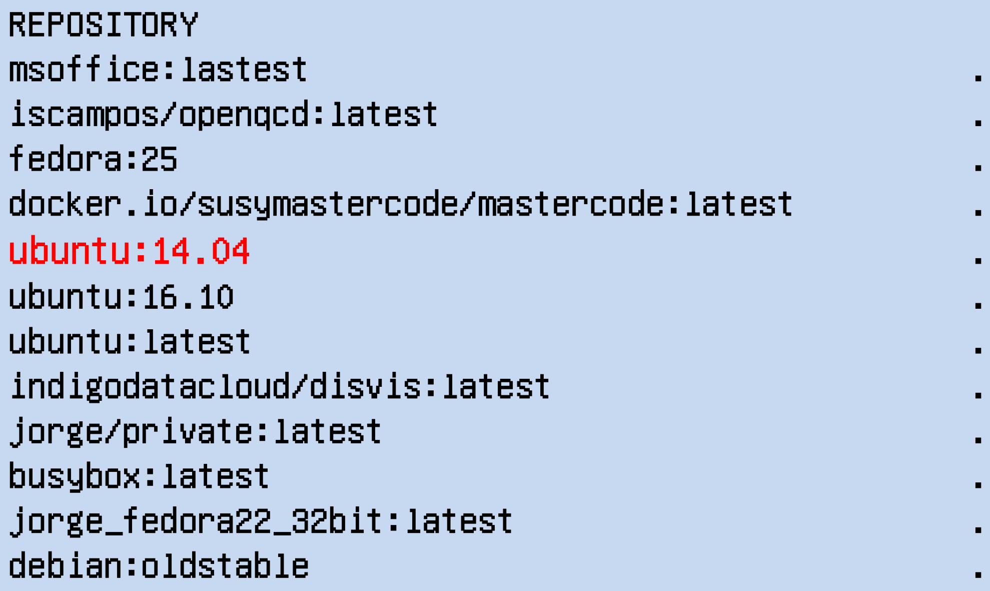
```
$ udocker pull ubuntu:14.04
```

Search for names and tags at:
https://hub.docker.com/

```
Downloading layer: sha256:bae382666908fd87a3a3646d7eb7176fa42226027d3256cac38ee0b79bdb0491
Downloading layer: sha256:f1ddd5e846a849fff877e4d61dc1002ca5d51de8521cced522e9503312b4c4e7
Downloading layer: sha256:90d12f864ab9d4cfe6475fc7ba508327c26d3d624344d6584a1fd860c3f0fefa
Downloading layer: sha256:a57ea72e31769e58f0c36db12d25683eba8fa14aaab0518729f28b3766b01112
Downloading layer: sha256:783a14252520746e3f7fee937b5f14ac1a84ef248ea0b1343d8b58b96df3fa9f
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
```

udocker: list local images

```
$ udocker  images
```

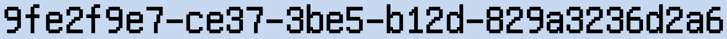


```
REPOSITORY
msoffice:lastest                                    .
iscampos/openqcd:latest                             .
fedora:25                                           .
docker.io/susymastercode/mastercode:latest          .
ubuntu:14.04                                        .
ubuntu:16.10                                        .
ubuntu:latest                                       .
indigodatacloud/disvis:latest                       .
jorge/private:latest                                .
busybox:latest                                      .
jorge_fedora22_32bit:latest                         .
debian:oldstable                                    .
```

udocker: create container from image

```
$ udocker create --name=ub14 ubuntu:14.04
```

container-alias

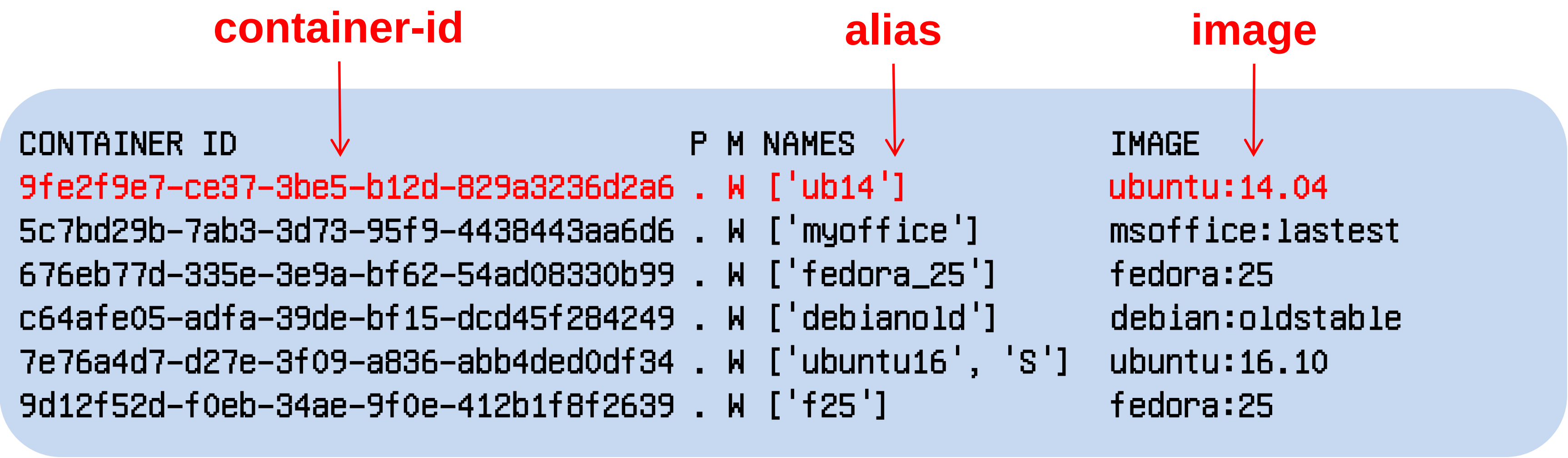


```
9fe2f9e7-ce37-3be5-b12d-829a3236d2a6
```

container-id

udocker: list containers

```
$ udocker   ps
```



container-id

alias

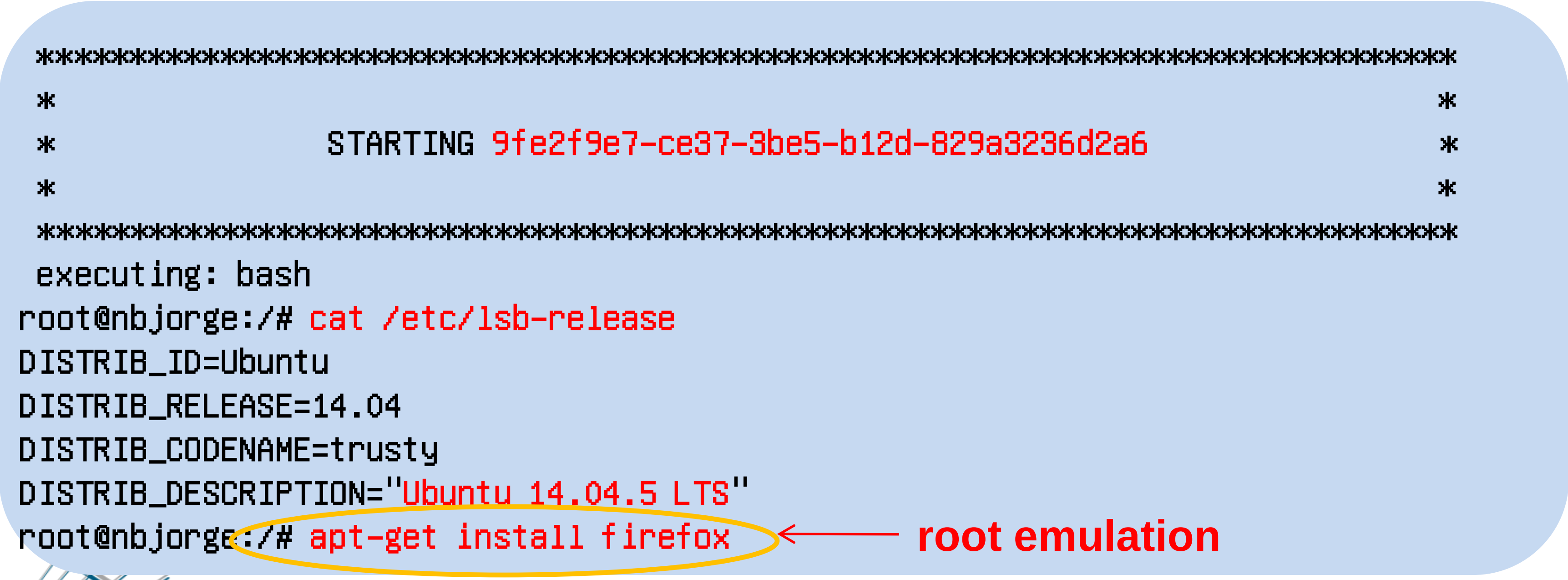
image

```
CONTAINER ID                         P M NAMES             IMAGE
9fe2f9e7-ce37-3be5-b12d-829a3236d2a6 . W ['ub14']          ubuntu:14.04
5c7bd29b-7ab3-3d73-95f9-4438443aa6d6 . W ['myoffice']      msoffice:lastest
676eb77d-335e-3e9a-bf62-54ad08330b99 . W ['fedora_25']     fedora:25
c64afe05-adfa-39de-bf15-dcd45f284249 . W ['debianold']     debian:oldstable
7e76a4d7-d27e-3f09-a836-abb4ded0df34 . W ['ubuntu16', 'S'] ubuntu:16.10
9d12f52d-f0eb-34ae-9f0e-412b1f8f2639 . W ['f25']           fedora:25
```

udocker: run container

```
$ udocker  run  ub14
```

udocker respects container metadata, if the container has a default cmd to run it will be run otherwise starts a shell

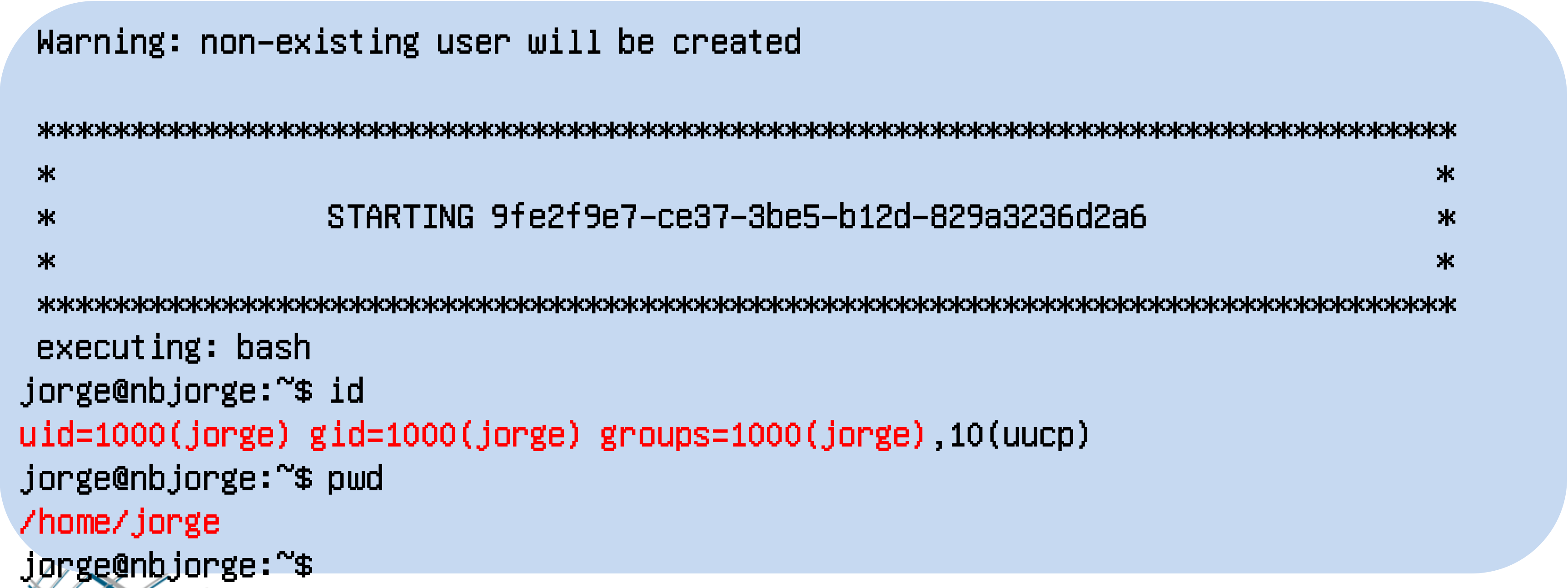


```
 ******************************************************************************
 *                                                                            *
 *               STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6                *
 *                                                                            *
 ******************************************************************************
 executing: bash
root@nbjorge:/# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.5 LTS"
root@nbjorge:/# apt-get install firefox
```

root emulation

udocker: run container as yourself

```
$ udocker run  --user=jorge  -v /home/jorge  \
  -e HOME=/jorge/home  --workdir=/home/jorge  ub14
```



```
 Warning: non-existing user will be created

 ******************************************************************************
 *                                                                            *
 *               STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6                *
 *                                                                            *
 ******************************************************************************
 executing: bash
jorge@nbjorge:~$ id
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge),10(uucp)
jorge@nbjorge:~$ pwd
/home/jorge
jorge@nbjorge:~$
```

udocker
How does it work …

udocker:

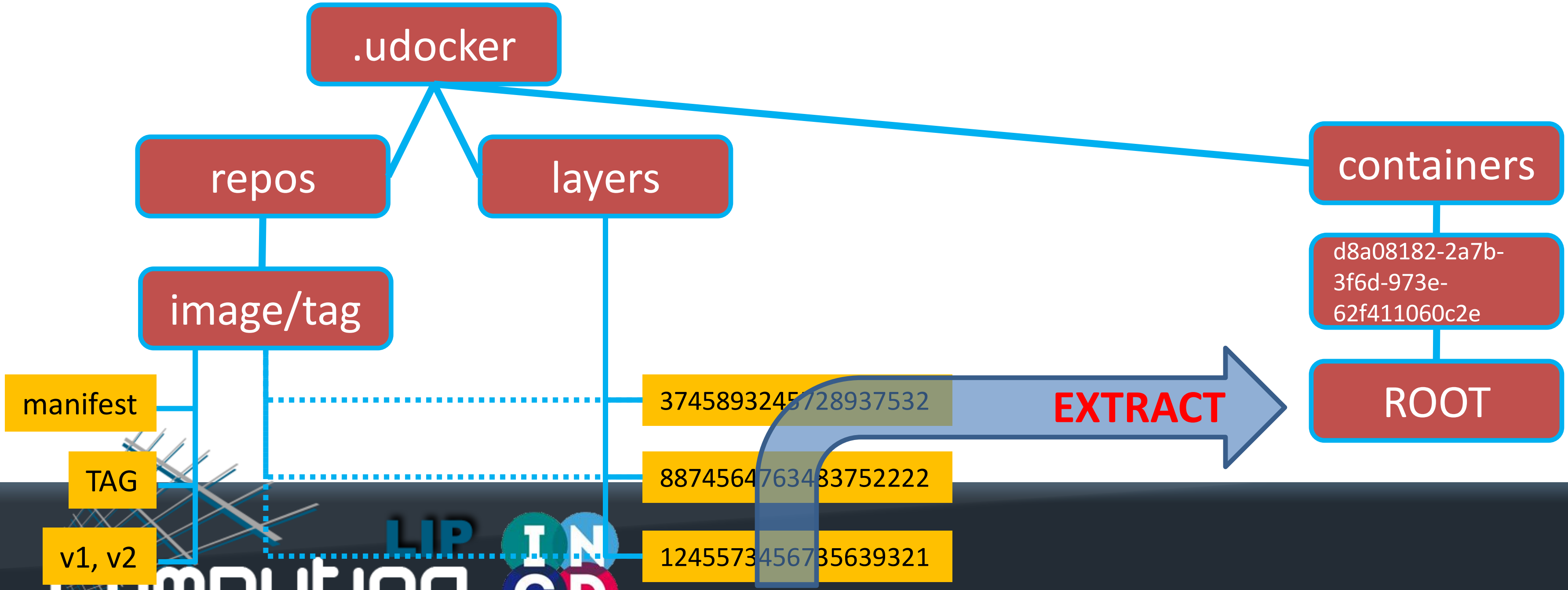
- Implemented
  - python, C, C++, go

- Can run:
  - CentOS 6, CentOS 7, Fedora >= 23
  - Ubuntu 14.04, Ubuntu 16.04
  - Any distro that supports python 2.7

- Components:
  - Command line interface docker like
  - Pull of containers from Docker Hub
  - Local repository of images and containers
  - Execution of containers with modular engines

udocker:

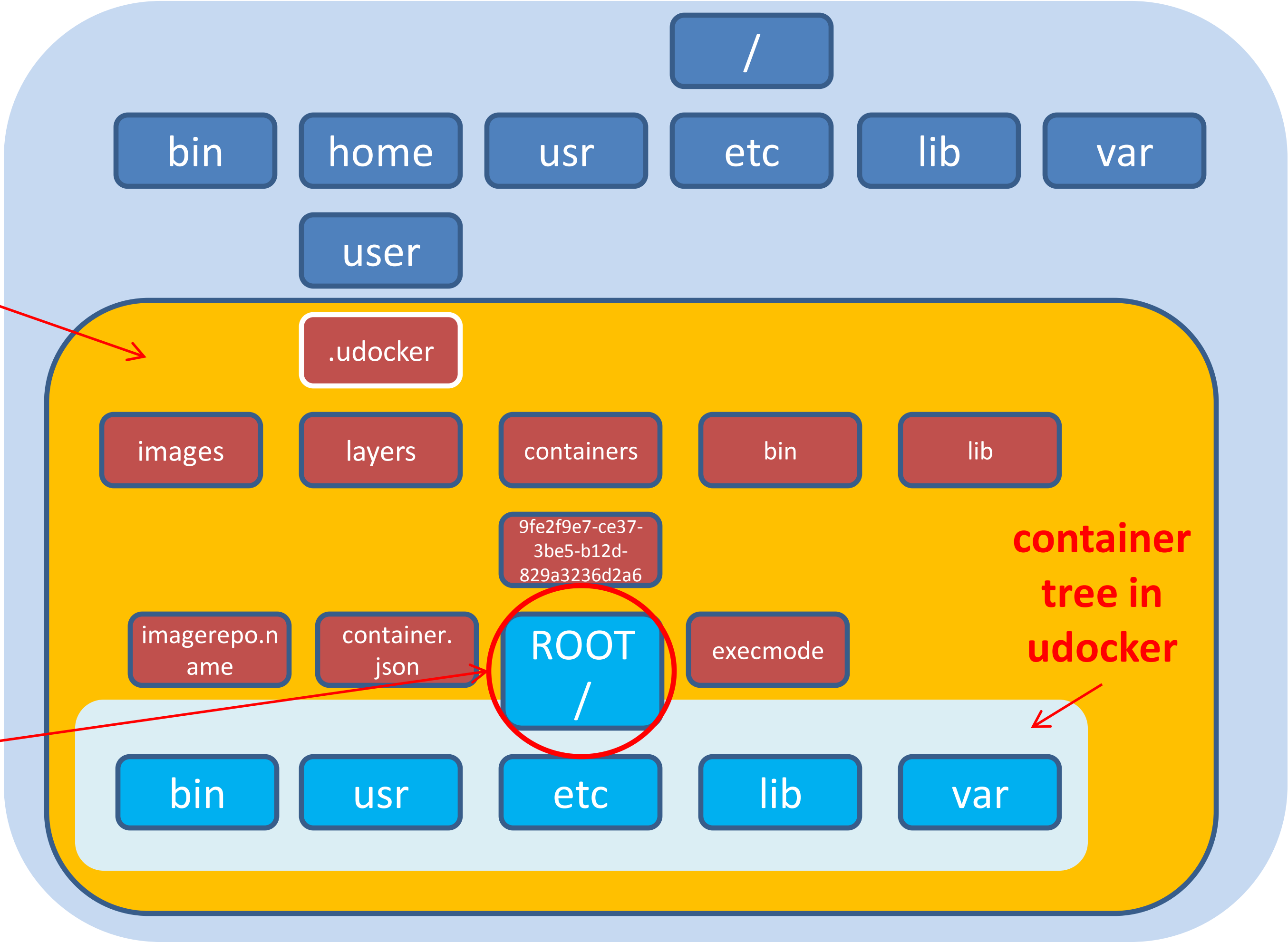
- Containers
  - Are produced from the layers by flattening them
  - Each layer is extracted on top of the previous
  - Whiteouts are respected, protections are changed
  - The obtained directory trees are stored under ~/.udocker/containers in the user home directory



udocker: directories and execution

- Execution
- chroot-like

udocker directory tree $HOME/.udocker



chroot to this directory becomes the new root for container processes

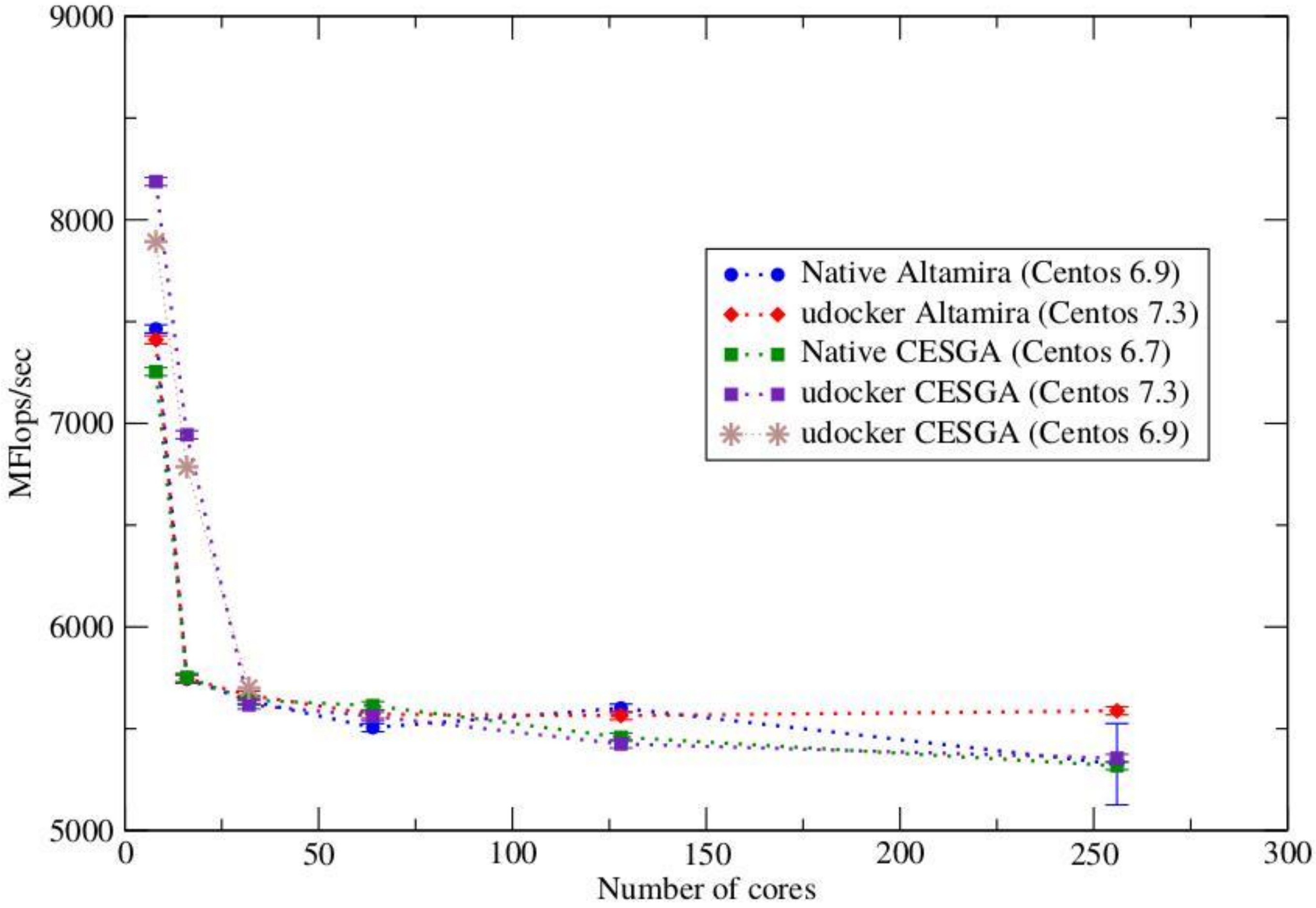
container tree in udocker

udocker: Execution methods

- udocker supports several techniques to achieve the equivalent to a chroot without using privileges
  - They are selected per container id via execution modes

Mode	Base	Description
P1	PRoot	PTRACE accelerated (with SECCOMP filtering) 🡐 DEFAULT
P2	PRoot	PTRACE non-accelerated (without SECCOMP filtering)
R1	runC	rootless unprivileged using user namespaces
F1	Fakechroot	with loader as argument and LD_LIBRARY_PATH
F2	Fakechroot	with modified loader, loader as argument and LD_LIBRARY_PATH
F3	Fakechroot	modified loader and ELF headers of binaries + libs changed
F4	Fakechroot	modified loader and ELF headers dynamically changed
S1	Singularity	where locally installed using chroot or user namespaces

udocker & Lattice QCD



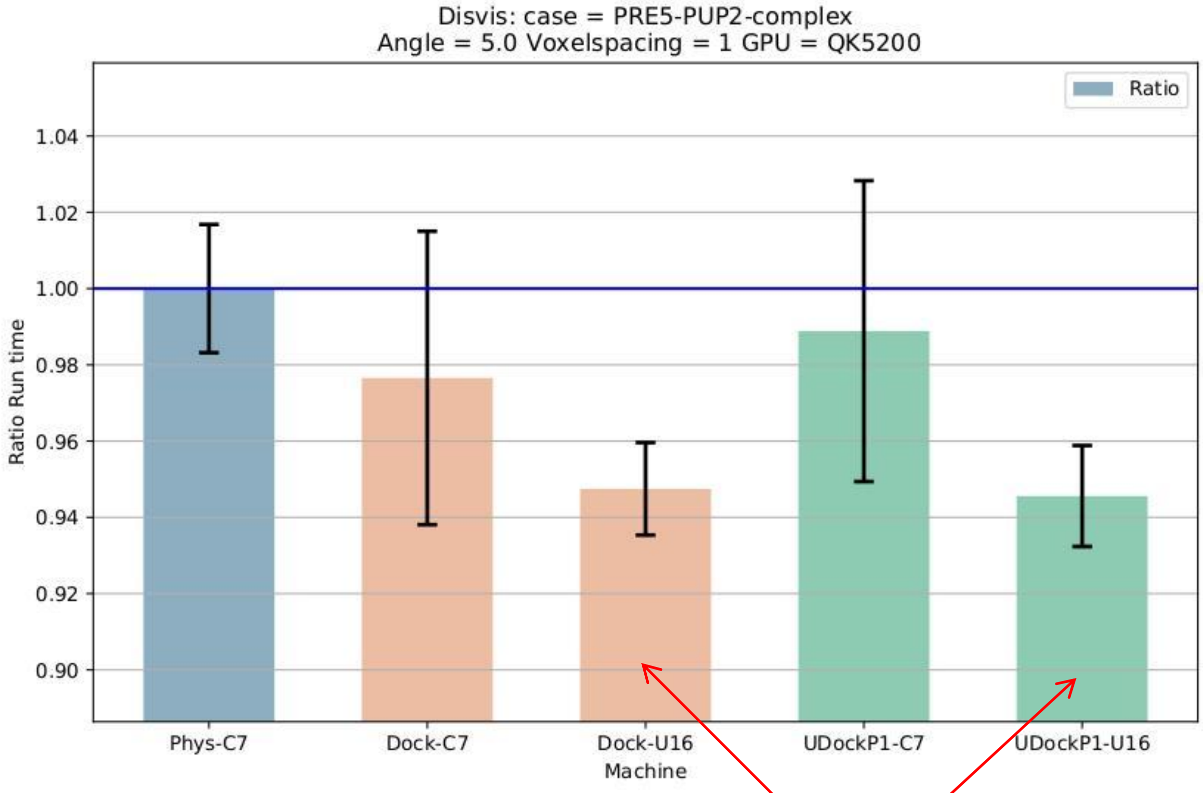
OpenQCD is a very advanced code to run lattice simulations

Scaling performance as a function of the cores for the computation of application of the Dirac operator to a spinor field.

Using OpenMPI

udocker in P1 mode

udocker & Biomolecular complexes



Better performance with Ubuntu 16 container

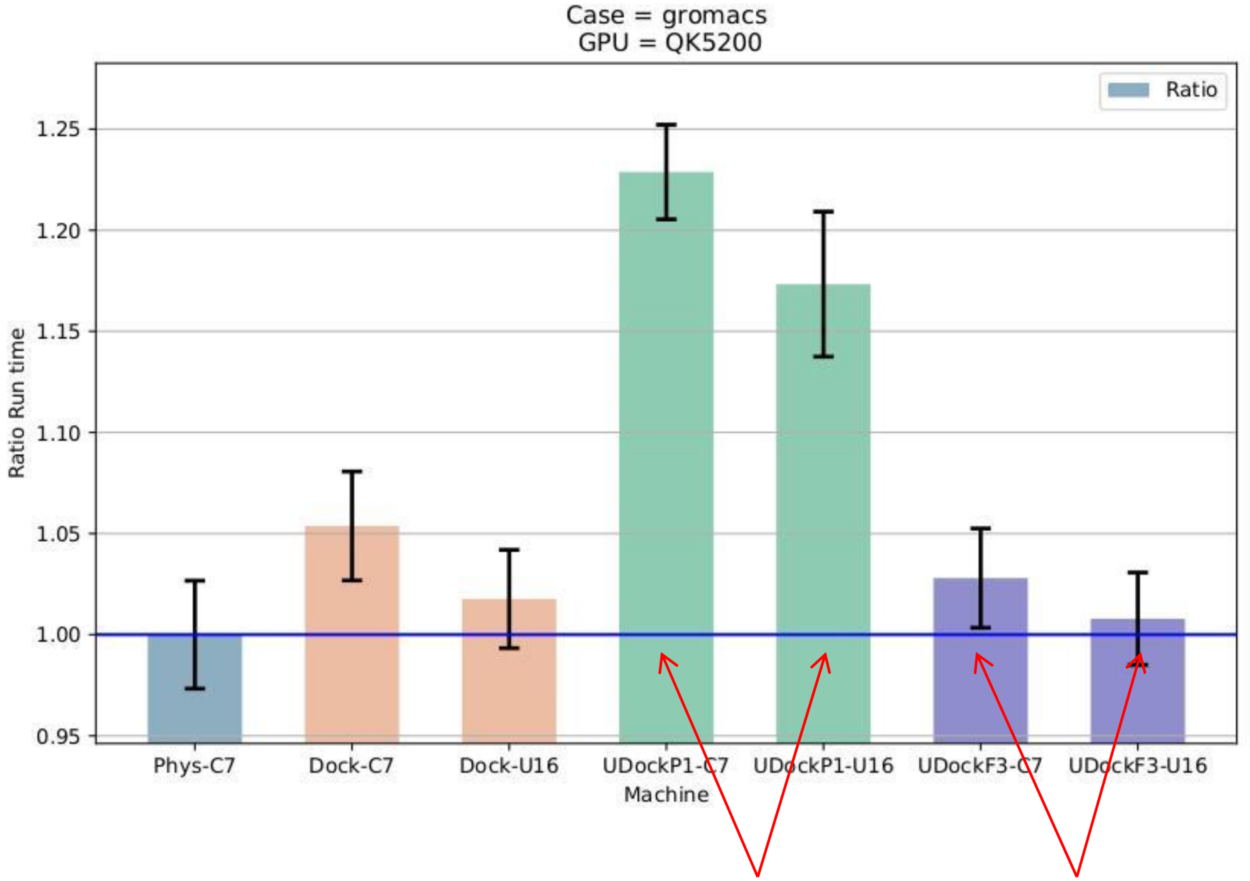
DisVis is being used in production with udocker

Performance with docker and udocker are the same and very similar to the host.

Using OpenCL and NVIDIA GPGPUs

udocker in P1 mode

udocker & Molecular dynamics



PTRACE

SHARED LIB CALL

Gromacs is widely used both in biochemical and non-biochemical systems.

udocker P mode have lower performance udocker F mode same as Docker.

Using OpenCL and OpenMP

udocker in P1 mode
udocker in F3 mode

udocker & Phenomenology

Performance Degradation

	Compiling	Running
HOST	0%	0%
DOCKER	10%	1.0%
udocker	7%	1.3%
VirtualBox	15%	1.6%
KVM	5%	2.6%

udocker in P1 mode

MasterCode connects several complex codes. Hard to deploy.

Scanning through large parameter spaces. High Throughput Computing

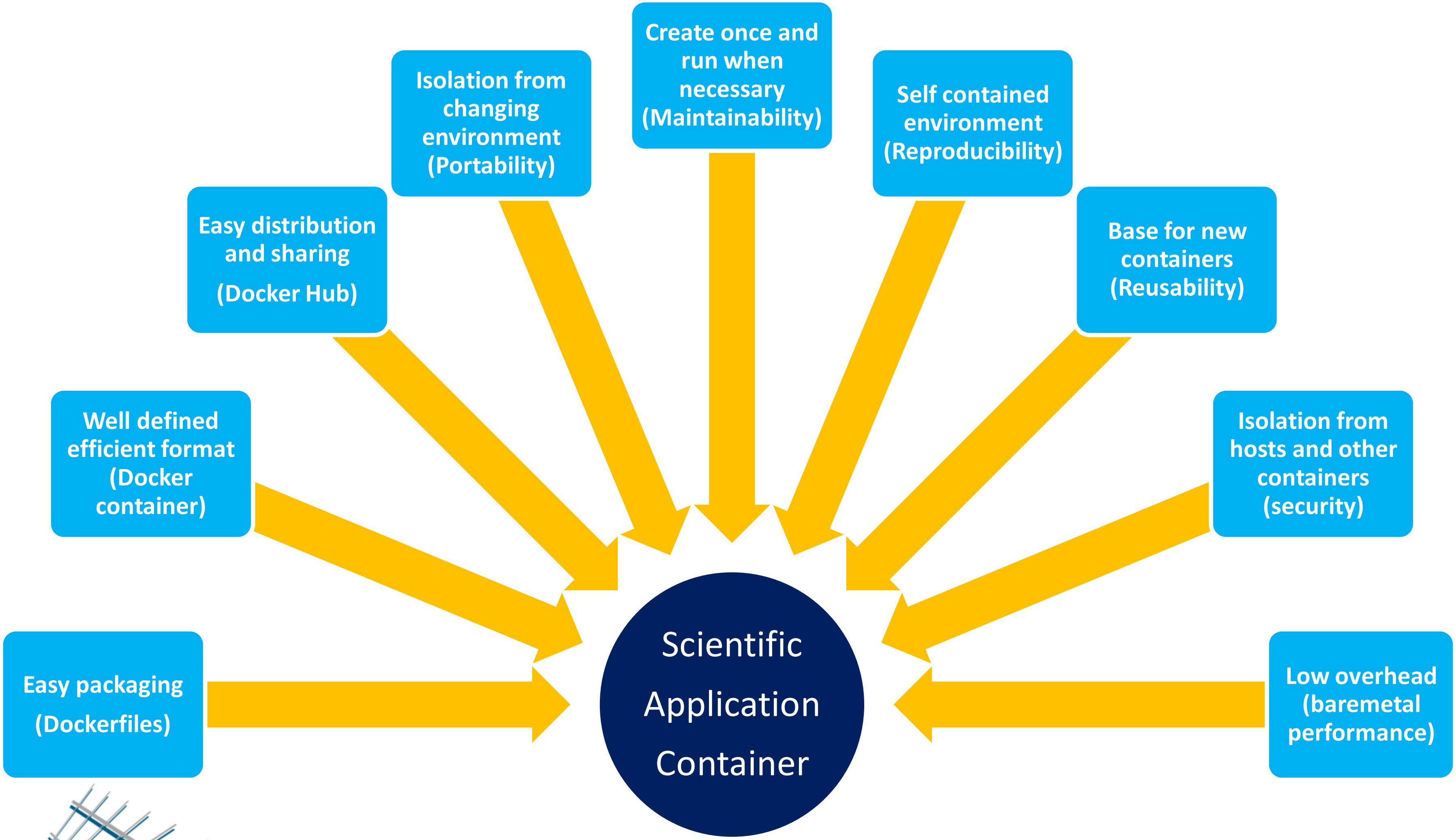
C++, Fortran, many authors, legacy code

udocker & Phenomenology

```
export MASTERDIR=/gpfs/csic_users/userabc/mastercode
export UDOCKER_DIR=$MASTERDIR/.udocker

 udocker.py run --hostauth \
          -v  /home/csic/cdi/ica/mcpp-master \
          -v  /home/csic/cdi/ica \
          -user=user001 \
          -w  /home/csic/cdi/ica/mcpp-master  mastercode \
          /bin/bash -c "pwd; ./udocker-mastercode.sh"
```

Scientific computing and containers



Easy packaging (Dockerfiles)

Well defined efficient format (Docker container)

Easy distribution and sharing (Docker Hub)

Isolation from changing environment (Portability)

Create once and run when necessary (Maintainability)

Self contained environment (Reproducibility)

Base for new containers (Reusability)

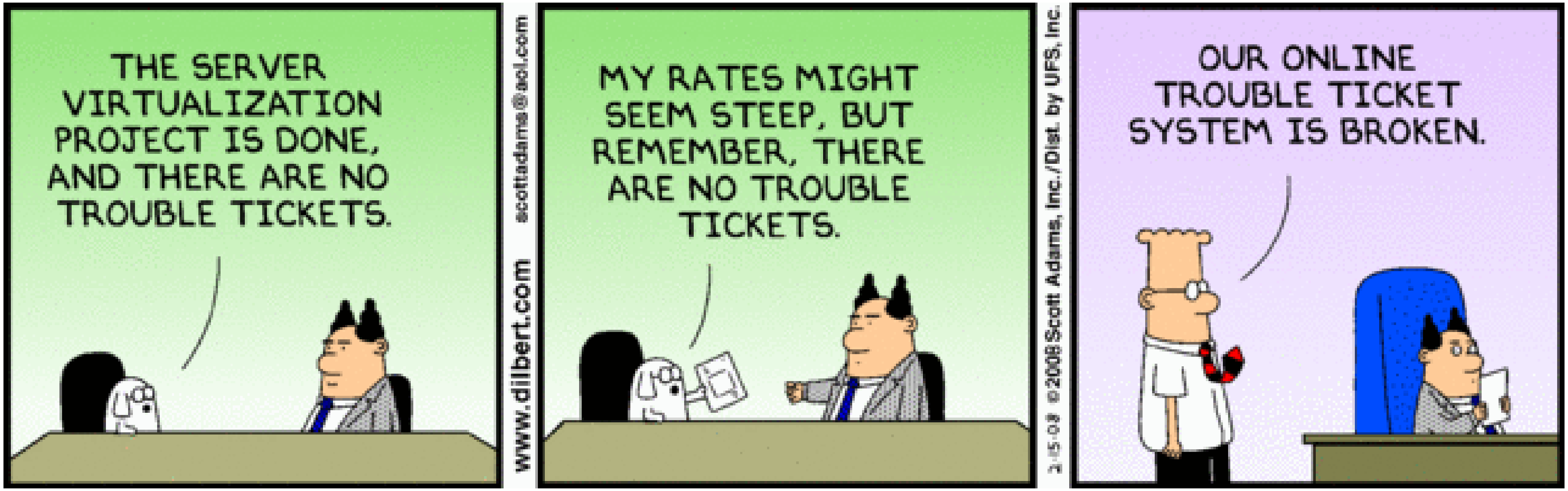
Isolation from hosts and other containers (security)

Low overhead (baremetal performance)

Scientific Application Container

Thank you

https://github.com/indigo-dc/udocker


THE SERVER VIRTUALIZATION PROJECT IS DONE, AND THERE ARE NO TROUBLE TICKETS.
MY RATES MIGHT SEEM STEEP, BUT REMEMBER, THERE ARE NO TROUBLE TICKETS.
OUR ONLINE TROUBLE TICKET SYSTEM IS BROKEN.
www.dilbert.com scottadams@aol.com
2-15-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.