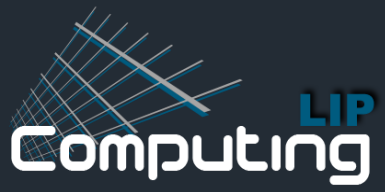
UDOCKER



Jorge Gomes <jorge@lip.pt>
Mário David <david@lip.pt>

Cofinanciado por:

COMPETE 2020
PROGRAMA OPERACIONAL COMPETITIVIDADE E INTERNACIONALIZAÇÃO

PORTUGAL 2020

Lisb@20[20]

FCT Fundação para a Ciência e a Tecnologia

INDIGO - DataCloud
Better Software for Better Science

EOSC-hub

Scientific computing and containers

UDOCKER
BE ANYWHERE

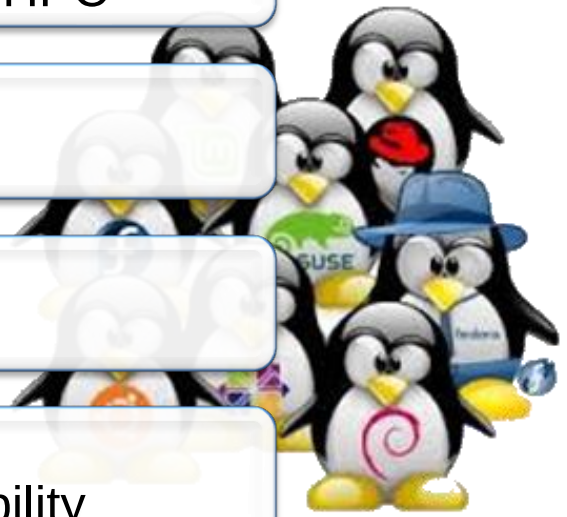
Running applications across infrastructures may require considerable effort

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

INDIGO-DataCloud

UDOCKER
BE ANYWHERE

Italy national museums digital library

ENES climate models

Molecular dynamics analysis

DisVis and Powerfit

LBT data archive

Galaxy instances

DARIAH Zenodo based repositories

Algae RNA sequencing

EMSO seismic data analysis

LHC/CMS clusters on demand

Molecular dynamics of proteins

Linux Containers

LIP Computing

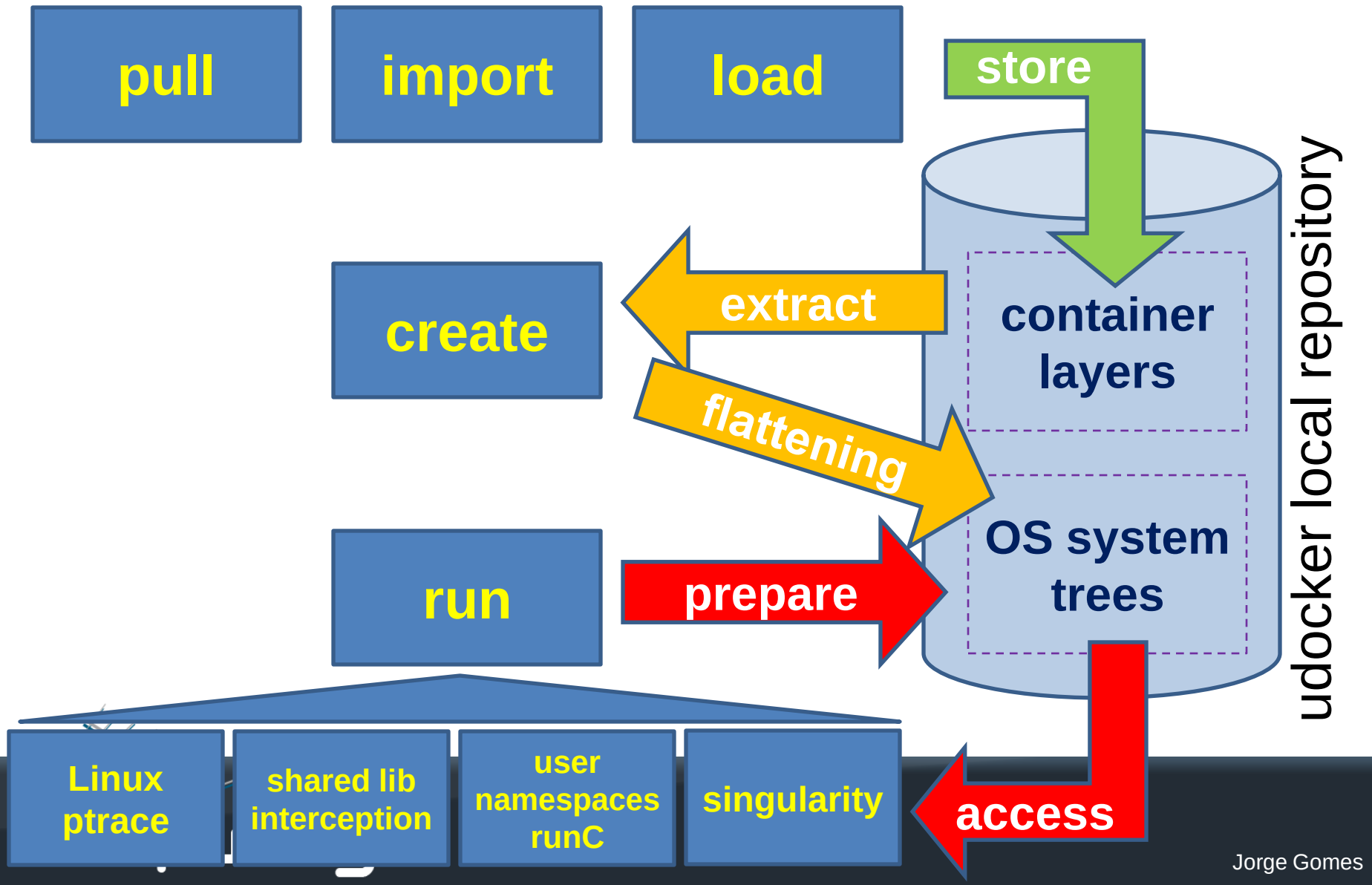
Jorge Gomes

udocker motivations

UDOCKER
BE ANYWHERE

- **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**
  - **in interactive or batch systems**
  - **with the normal process controls and accounting**
- Empower end-users to run applications in containers

udocker: concept



Search or jump to… / Pull requests Issues Marketplace Explore

indigo-dc / udocker

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

Star 485 Fork 54

Code Issues 29 Pull requests 2 Projects 0 Wiki Insights Settings

Branch: master ▾ udocker / README.md Find file Copy path

orviz Fix build status URL 00d714c on Dec 12, 2018

4 contributors

299 lines (237 sloc) 10.8 KB Raw Blame History

build failing

UDOCKER

udocker is a basic user tool to execute simple docker containers in user space without requiring root privileges. Enables download and execution of docker containers by non-privileged users in Linux systems where docker is not available. It can be used to pull and execute docker containers in Linux batch systems and interactive clusters that are managed by other entities such as grid infrastructures or externally managed batch or interactive systems.

Repository

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

Documentation

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

udocker: install from github

UDOCKER
BE ANYWHERE

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

$ chmod u+rx udocker

$ ./udocker install
```

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

udocker: installation

- udocker front-end is a python script
- udocker install
  - Creates a directory tree under $HOME/.udocker
  - Places some executables and libraries

```
.udocker
├── bin
├── containers
│   ├── 2ed713b1-8110-3920-bdfa-ae997ed4a51c
│   └── ub14 -> 2ed713b1-8110-3920-bdfa-ae997ed4a51c
├── layers
├── lib
└── repos
    ├── ubuntu
    └── zz
```

udocker: install from github

```
$ ./udocker install --force

$ ./udocker install --force --purge
```

udocker: version information

UDOCKER
BE ANYWHERE

```
$ ./udocker version
```

```
version: 1.1.3
tarball: https://owncloud.indigo-datacloud.eu/index.php/s/iv4FOV1jZcfnGFH/download
https://cernbox.cern.ch/index.php/s/CS31ycKOpj2KzxO/download
https://download.ncg.ingrid.pt/webdav/udocker/udocker-1.1.3.tar.gz
tarball_release: 1.1.3
```

```
$ curl https://download.ncg.ingrid.pt/webdav/udocker/udocker-1.1.3.tar.gz > XXX
$ export UDOCKER_TARBALL=$(pwd)/XXX
$ ./udocker install
```

udocker

- Run time to execute docker containers:
  - search
  - pull
  - images
  - create
  - rmi
  - ps
  - rm
  - run
  - login
  - logout
  - load
  - save
  - import
  - export
  - setup
  - clone
  - verify
  - Inspect
  - mkrepo

udocker: pull images from repository

UDOCKER
BE ANYWHERE

```
$ 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: export and import as image

```
$ udocker   images
```

```
REPOSITORY
ubuntu:14.04
```

.

udocker: export and import as image

UDOCKER
BE ANYWHERE

```
$ udocker   images  -l
```

```
REPOSITORY
ubuntu:14.04                                                    .
  /home/jorge/.udocker/repos/ubuntu/14.04
   /sha256:515c9bb515362c9d26b0c6acaa3ad0a79c20cf421d56a8c5bb4ddc60a336239b (1 MB)
   /sha256:e1eabe0537eb2d3647100f04687474cc1c232b4e512e70fd0a09b93d55da8f69 (1 MB)
   /sha256:a7344f52cb744a28edb7b2c806f4227d07219709d2365c32a42b580165d261c1 (64 MB)
   /sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4 (1 MB)
   /sha256:4701f1215c13b72f8e1fbd292558fc4cb49655209db1b450cbb5c068be64956c (1 MB)
```

udocker: create container from image

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

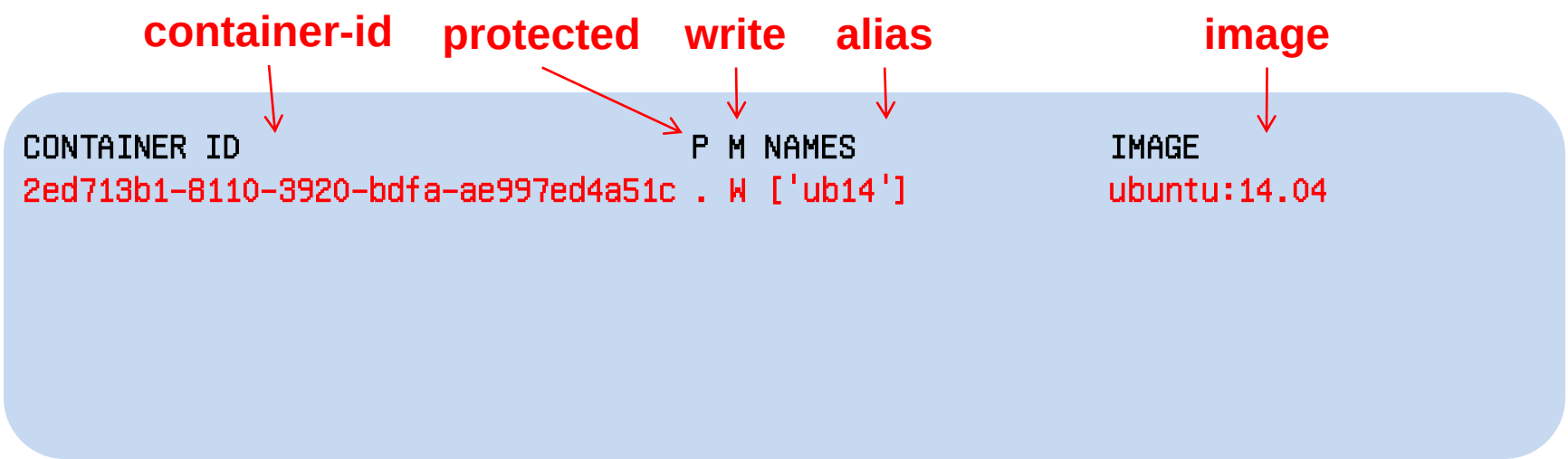
container-alias

```
2ed713b1-8110-3920-bdfa-ae997ed4a51c
```

container-id

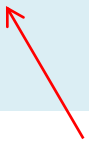
udocker: list containers

```
$ udocker ps
```



udocker: set execution mode

```
$ udocker setup --execmode=P2 ub14
```



Execution mode

udocker: run container

UDOCKER
BE ANYWHERE

```
$ 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:/#
```

udocker: run container

Limited root emulation is allows some actions such as software installation

```
root@nbjorge:/# apt-get  -o  APT::Sandbox::User=root  update

root@nbjorge:/# apt-get  -o  APT::Sandbox::User=root  install  unzip
```

udocker: environment

```
$ ./udocker run  -e  AAAA=xxxx  ub14 env | grep AAAA
```

```
**************************************************************************
*                                                                        *
*               STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6            *
*                                                                        *
**************************************************************************
executing: bash
AAAA=xxxx
```

udocker: environment

UDOCKER
BE ANYWHERE

$ ./udocker run --hostenv ub14 env

```
**************************************************************************************
*                                                                                    *
*                STARTING 9fe2f9e7-ce37-3be5-b12d-829a3236d2a6                       *
*                                                                                    *
**************************************************************************************
executing: bash
```

...

...

...

udocker: run container as yourself

UDOCKER
BE ANYWHERE

```
$ ./udocker run  --user=jorge  -v /home/jorge  \
   -e HOME=/home/jorge  --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:~$
```

LIP Computing

udocker: run container as yourself

UDOCKER
BE ANYWHERE

```
$ udocker run  --user=jorge  --bindhome  \
  --hostauth  ub14
```

```
 ****************************************************************************
 *                                                                          *
 *                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:~$
```

LIP Computing

udocker: run commands in the prompt

```
$ udocker run --user=jorge --bindhome \
  --hostauth ub14 /bin/bash -c "id; pwd"
```

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

udocker: quiet

```
$ udocker -q run ub14 /bin/cat /etc/lsb-release
```

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.5 LTS"
```

```
$ alias x=udocker -q run --user=user --bindhome \
  --hostauth ub14
$ x /bin/ls
```

udocker: run commands in the prompt

```
$ udocker run --user=jorge --bindhome \
  --hostauth ub14 /bin/bash <<EOF
id; pwd
EOF
```

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

udocker: a more complex application

UDOCKER
BE ANYWHERE

```
$ apt-get -o APT::Sandbox::User=root install firefox

$ ./udocker.py run --bindhome --hostauth \
    --hostenv   -v /sys  -v /proc  -v /var/run  \
     -v /dev  --user=jorge  ub14  firefox  --no-remote
```

udocker: nvidia GPGPUs

```
$ udocker setup --execmode=F3 --nvidia ub14
```

udocker: duplicate a container

```
$ udocker  clone  --name=yy  ub14

$ udocker  ps
```

cloned container-id

```
CONTAINER ID                          P M NAMES             IMAGE
2ed713b1-8110-3920-bdfa-ae997ed4a51c . W ['ub14']          ubuntu:14.04
e542b94f-f6a0-31fa-b113-72c0fa5826b9 . W ['yy']            ubuntu:14.04
```

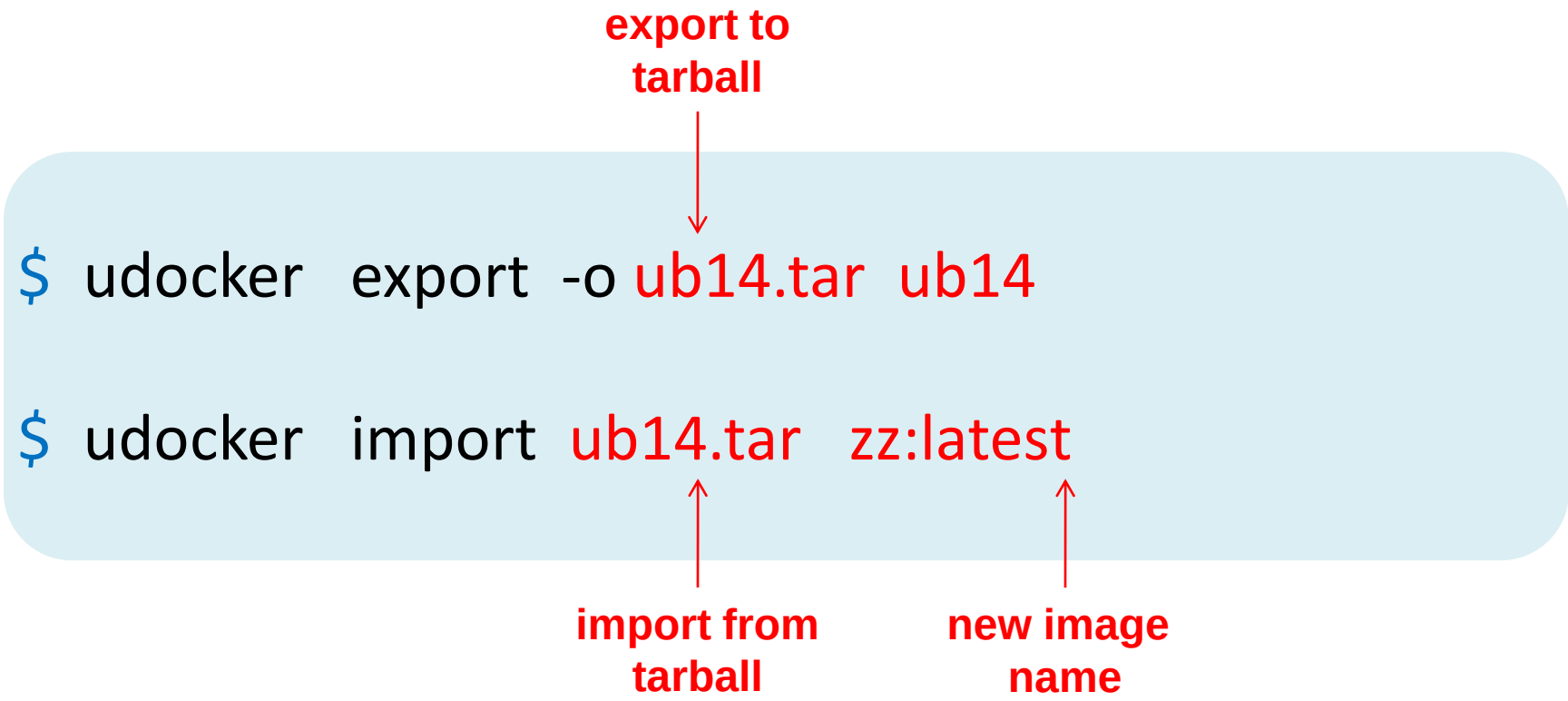
udocker: remove created container

remove container by alias or id

```
$ udocker   rm  yy

$ udocker   rm  e542b94f-f6a0-31fa-b113-72c0fa5826b9
```

udocker: export and import as image



- Only the container files are exported, metadata is lost
- This is interoperable with docker

udocker: export and import as image

```
$ udocker   images
```

```
REPOSITORY
ubuntu:14.04
zz:latest
```

udocker: remove image

```
$ udocker   rmi   zz:latest

$ udocker  images
```

```
REPOSITORY
ubuntu:14.04                                                    .
```

udocker: export and import as container

export to tarball

```
$ udocker   export  -o ub14.tar  ub14

$ udocker   import  --tocontainer  --name=xx  ub14.tar
```

import from tarball to container

new container alias

- Only the container files are exported, metadata is lost
- Export is interoperable with docker

udocker: remove image

UDOCKER
BE ANYWHERE

```
$ udocker   ps
```

```
CONTAINER ID                          P M NAMES             IMAGE
2ed713b1-8110-3920-bdfa-ae997ed4a51c . W ['ub14']          ubuntu:14.04
b9414dfa-41ae-39f9-8ed1-f1d7b12cc9da . W ['xx']            IMPORTED:unknown
.
```

udocker: export and import as container

export clone

```
$ udocker   export  --clone  -o ub14.tar  ub14

$ udocker   import  --clone  --name=xx  ub14.tar
```

import clone

- Is imported as a container saving space and time
- Container metadata and execution mode are preserved
- This is NOT interoperable with docker

udocker: export + import as container

export clone

```
$ udocker  export  --clone  ub14  |  \
  ssh user@host  \
  “udocker  import  --clone  --name=xx - ; udocker  run  xx”
```

import clone

run

- Export and import across nodes
- Piping stdout to stdin and minimizing I/O

udocker: save and load images

save image with all layers and metadata


```
$ docker  save  -o  image.tar  centos:centos6

$ udocker  load  -i  image.tar
```


load image with all layers and metadata

- Docker saves the image as a tarfile containing layers
- Udocker loads the image
- Can be used to transfer images without having to pull them

udocker: save and load images

save image with all layers and metadata

```
$ docker save centos:centos6 | udocker load
```

load image with all layers and metadata

- Save from docker and load with udocker
- Piping stdout to stdin

UDOCKER
BE ANYWHERE

udocker
How does it work ...

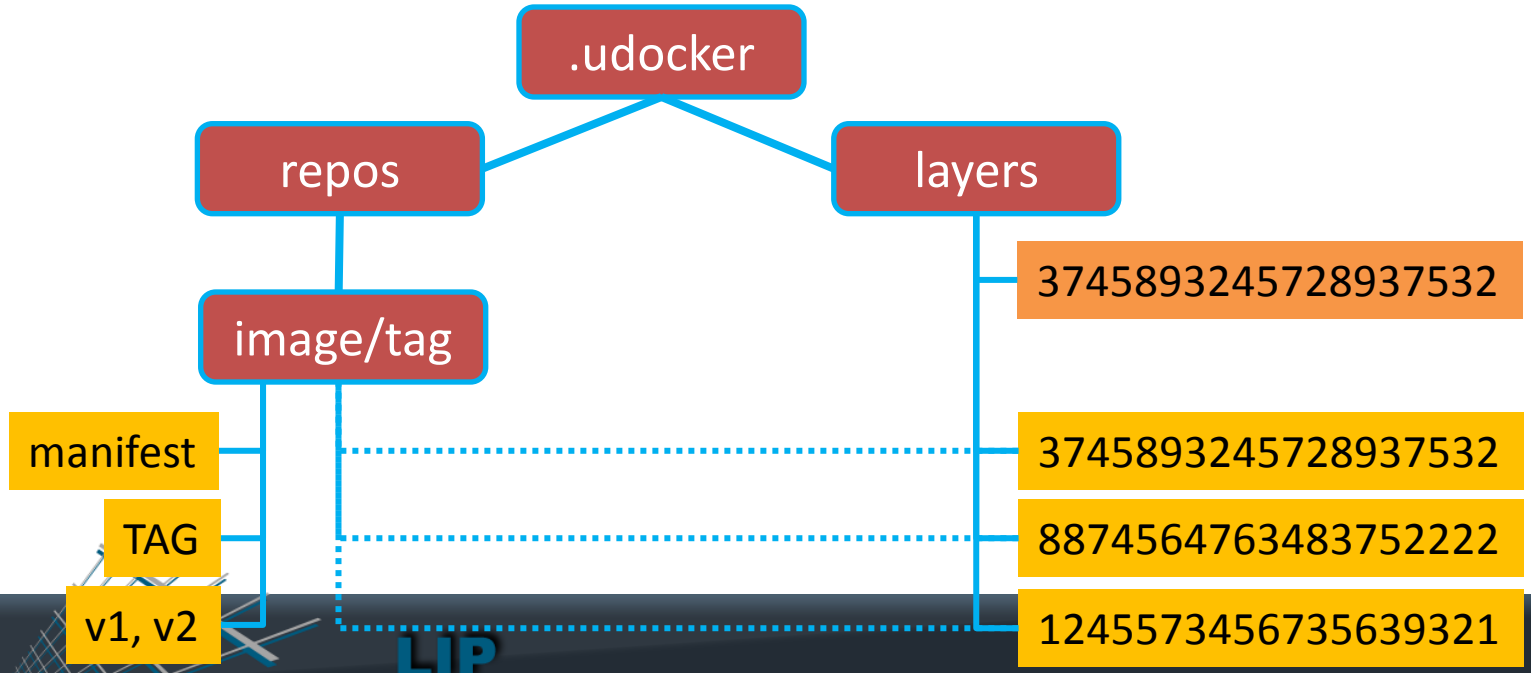
LIP Computing

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

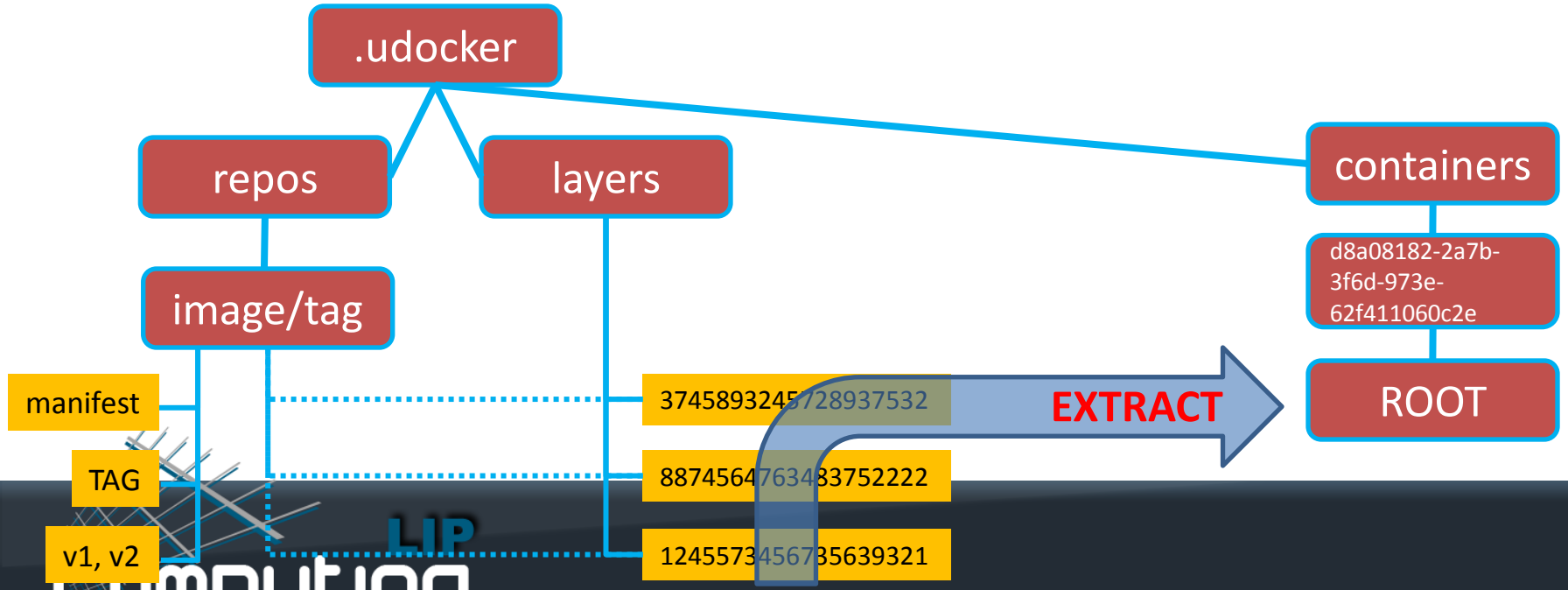
Udocker: pull

- Images
  - Layers and metadata are pulled with DockerHub REST API
  - Image metadata is interpreted to identify the layers
  - Layers are stored in the use home directory under ~/.udocker/layers so that can be shared by multiple images



Udocker: create

- 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

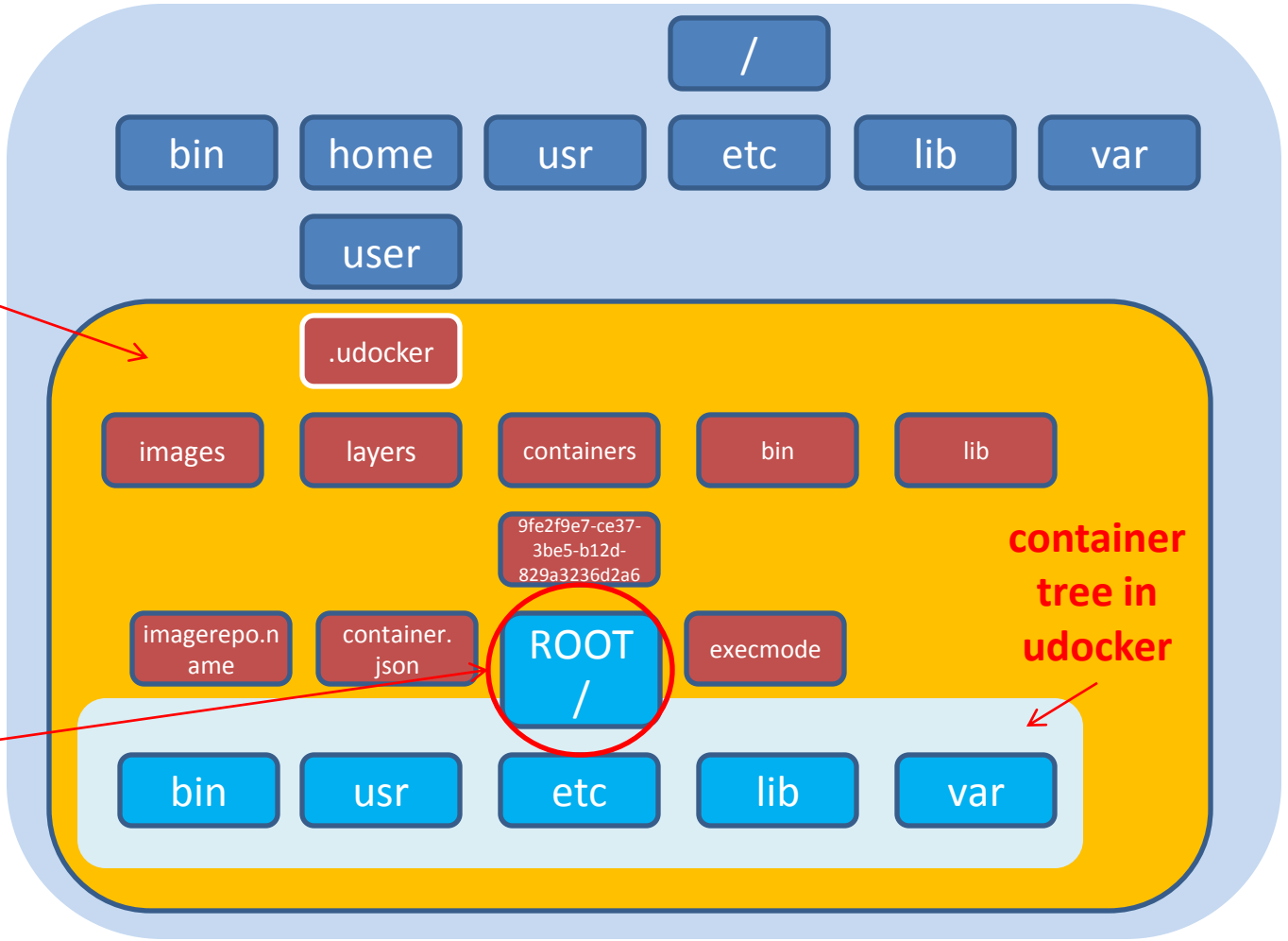


udocker: run

- Execution
- chroot-like

udocker directory tree $HOME/.udocker

chroot to this directory becomes the new root for container processes



udocker: inspect

```
$ udocker inspect ub14

$ udocker inspect -p ub14
```

udocker: Execution methods

UDOCKER
BE ANYWHERE

- 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: set execution mode F3

```
$ udocker setup --execmode=F3 ub14
```

Execution mode F3

udocker: PRoot engine (P1 and P2)

UDOCKER
BE ANYWHERE

UDOCKER_DIR : root directory of udocker usually $HOME/.udocker

UDOCKER_BIN : location of udocker related executables

UDOCKER_LIB : location of udocker related libraries

UDOCKER_CONTAINERS : location of container directory trees

UDOCKER_KEYSTORE : location of keystore for login/logout

UDOCKER_TMP : location of temporary directory

UDOCKER_TARBALL : location of installation tarball (file of URL)

UDOCKER_LOGLEVEL : logging level

UDOCKER_USE_CURL_EXECUTABLE : pathname for curl binary

UDOCKER_REGISTRY : override registry default is Docker Hub.

UDOCKER_INDEX : override default index default is Docker Hub.

udocker: PRoot engine (P1 and P2)

UDOCKER
BE ANYWHERE

- PRoot uses PTRACE to intercept system calls
- Pathnames are modified before the call
  - To expand container pathnames into host pathnames
- Pathnames are modified after the call
  - To shrink host pathnames to container pathnames

- The P1 mode uses PTRACE + SECCOMP filtering, to limit the interception to the set of calls that manipulate pathnames
  - We developed code to make it work on recent kernels
  - P1 is the udocker default mode
- The P2 mode uses only PTRACE ➔ therefore tracing all calls

- The impact of tracing depends on the system call frequency

udocker: runC engine (R1)

UDOCKER
BE ANYWHERE

- runC is a tool to spawn containers according to the Open Containers Initiative (OCI) specification
  - In a very recent release 1.0 candidate 3, runC supports unprivileged namespaces using the user namespace
  - Unprivileged namespaces have many limitations but allow execution in a contained Docker like environment
  - Only run as root is supported
  - Available devices are limited

- We added conversion of Docker metadata to OCI
- udocker can produce an OCI spec and run the containers with runC transparently

udocker: Fakechroot engine

- Fakechroot is a library to provide chroot-like behaviour
- Uses the Linux loader LD_PRELOAD mechanism to:
  - intercept library calls that manipulate pathnames
  - change the pathnames similarly to PRoot
- It was conceived to support debootstrap in debian

- Originally the OS in the host and in the chroot tree must be the same:
  - as the loader inside the chroot will by default load libraries from the host system directories
  - the loaders are statically linked and the pathnames inside are absolute and non changeable

udocker: Fakechroot engine

UDOCKER
BE ANYWHERE

- How loaders search for libraries:
  - If the pathname has a / they are directly loaded
  - If the pathname does not contain / (no directory specified) a search path or location can be obtained from:
    1. DT RPATH dynamic section attribute of the ELF executable
    2. LD LIBRARY PATH environment variable
    3. DT RUNPATH dynamic section attribute of the ELF executable
    4. cache file /etc/ld.so.cache
    5. default paths such as /lib64, /usr/lib64, /lib, /usr/lib
- The location of the loader itself is encoded in the executables ELF header

udocker: Fakechroot engine (F1)

UDOCKER
BE ANYWHERE

- The loader is encoded in the ELF header of executable
  - is the executable that loads libraries and calls the actual executable
  - also act as library providing functions and symbols
- Is essential that executables in the container are run with the loader inside of the container instead of the host loader

- The mode **F1 enforces the loader**:
  - **passes it as 1st argument in exec* and similar calls shifting argv**
  - the loader starts first gets the executable pathname and its arguments from argv and launches it
  - **Enforcement of shared library locations** is performed by filling in **LD_LIBRARY_PATH** with the library locations in the container and also **extracted from the container ld.so.cache**

udocker: Fakechroot engine (F2)

UDOCKER
BE ANYWHERE

- The mode **F2 changes the loader binary** within the container:
  - A copy of the container loader is made
  - The **loader binary is then edited by udocker**
  - The loading from host **locations /lib, /lib64 etc is disabled**
  - The **loading using the host ld.so.cache is disabled**
  - **LD_LIBRARY_PATH is renamed to LD_LIBRARY_REAL**

- Upon execution
  - Invocation is performed as in mode F1
  - The LD_LIBRARY_REAL is filled with library locations from the container and its ld.so.cache
  - Changes made by the user to LD_LIBRARY_PATH are intercepted and pathnames adjusted to container locations and inserted in LD_LIBRARY_REAL

udocker: Fakechroot engine (F3 and F4)

- The mode **F3 changes binaries both executables and libraries**
  - The **PatchELF** tool was heavily modified to enable easier change of
    - Loader location in ELF headers of executables
    - Library path locations inside executables and libraries

- **When modes F3 or F4 are selected the executables and libraries are edited**
  - The loader location is change to point to the container
  - The libraries location if absolute are changed to point to container
  - The libraries search paths inside the binaries are changed to point to container locations

- The loader no longer needs to be passed as first argument
- The libraries are always fetched from container locations

udocker: Fakechroot engine (F3 and F4)

- The LD_LIBRARY_REAL continues to be used in F3 and F4

- The mode **F4 adds dynamic editing of executables and libraries**
- This is useful with libraries or executables are added to the container or created as result of a compilation

- Containers in modes F3 and F4 cannot be transparently moved across different systems:
  - the absolute pathnames to the container locations will likely differ.
  - In this case convert first to another mode before transfer
  - or at arrival use: “setup --execmode=Fn --force”

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"
```

udocker & sockets

Compatibility with other platforms, or compatibility with older stuff to avoid overruns while using `snprintf()` and `strncpy()`.

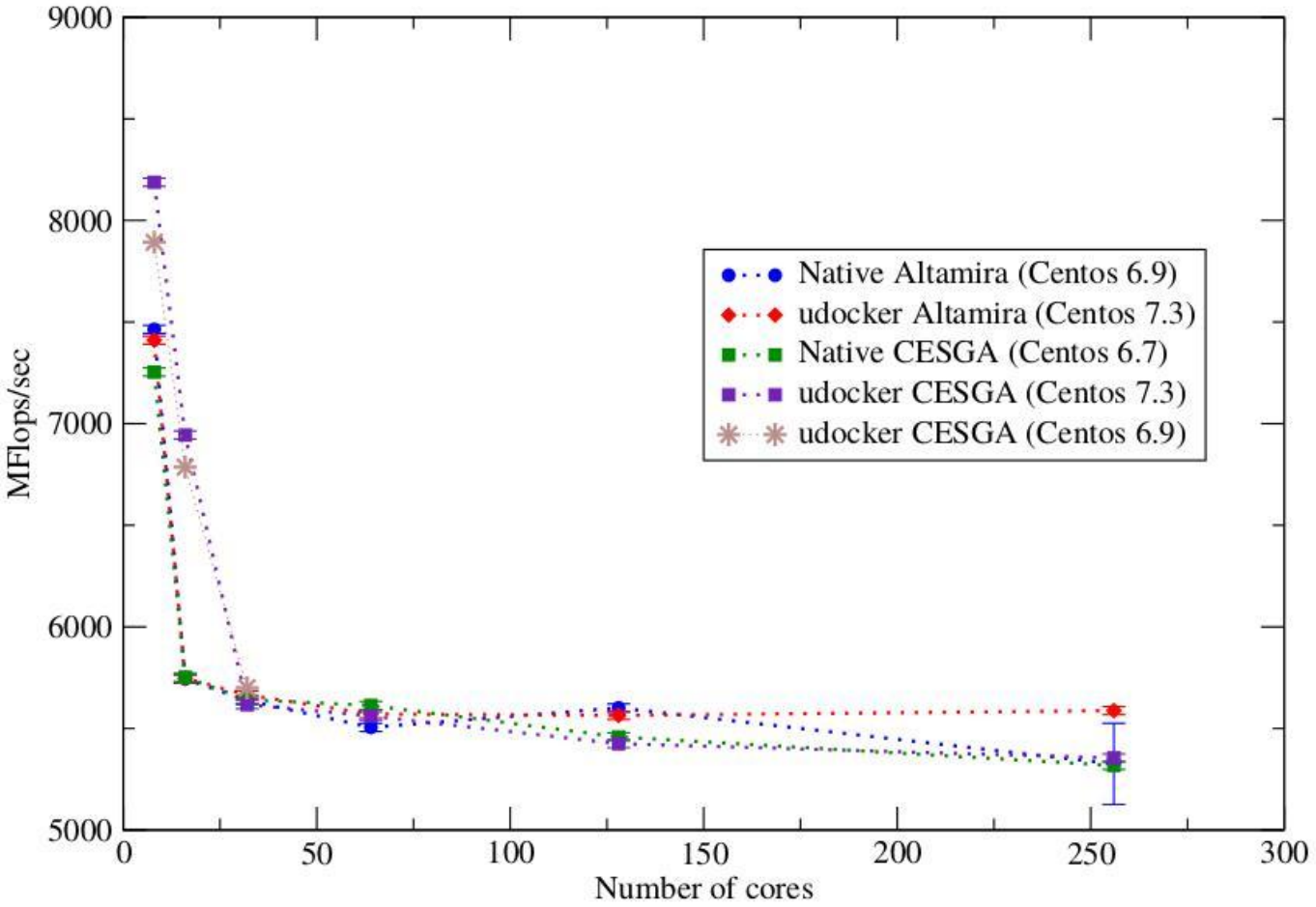
Michael Kerrisk explain in his book at the page 1165 - Chapter 57, Sockets: Unix domain :

> SUSv3 doesn't specify the size of the sun_path field. Early BSD implementations used 108 and 104 bytes, and one contemporary implementation (HP-UX 11) uses 92 bytes. Portable applications should code to this lower value, and use snprintf() or strncpy() to avoid buffer overruns when writing into this field.

Linux domain socket limit pathname limit is 107

```
$ udocker run -v /tmp -v /dir/dir/dir/dir ub14
```

udocker & Lattice QCD



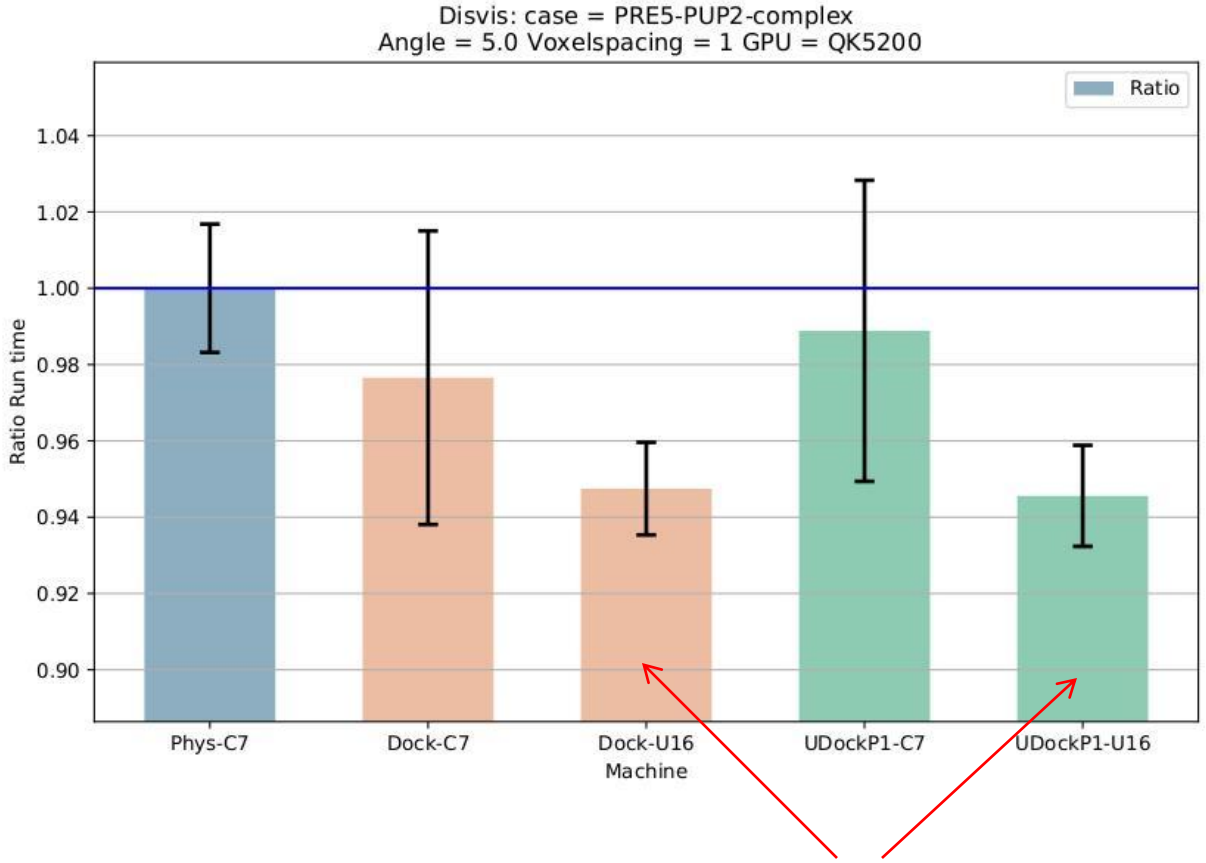
OpenQCD is a very advanced code to run lattice simulations

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



DisVis is being used in production with udocker

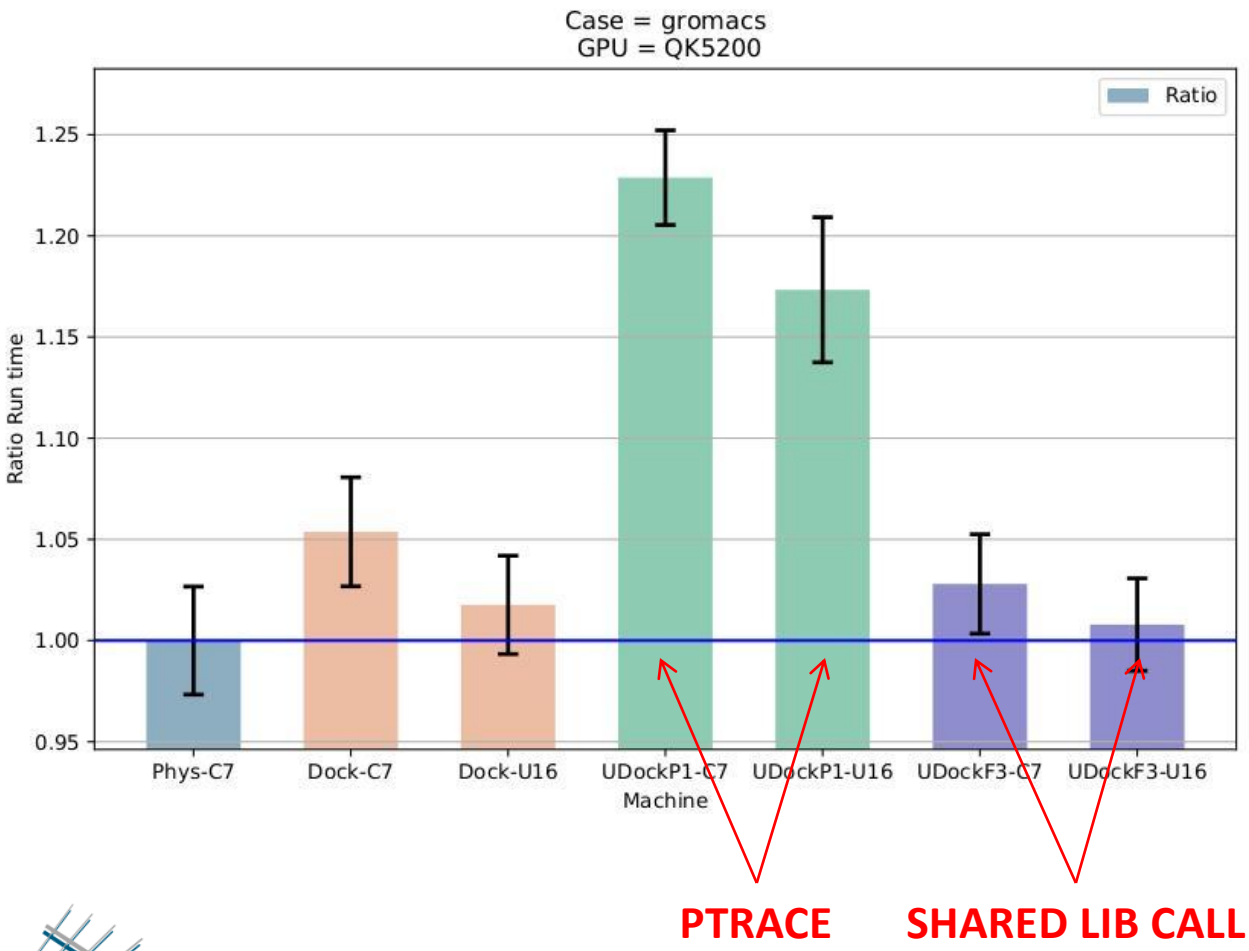
Performance with docker and udocker are the same and very similar to the host.

Using OpenCL and NVIDIA GPGPUs

Better performance with Ubuntu 16 container

udocker in P1 mode

udocker & Molecular dynamics



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

UDOCKER
BE ANYWHERE

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

LIP Computing

Thank you

UDOCKER
BE ANYWHERE

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

Thank you

UDOCKER
BE ANYWHERE

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