Running containers in your user space with udocker

Jorge Gomes / Mário David
udocker@lip.pt

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

Slides in:

https://indico.egi.eu/event/5541/

Summary

About Containers
udocker Concept
Installation
Basic Usage
Advanced Topics
Benchmarks
Questions

About Containers

Containers in Scientific Computing

Running across systems often requires considerable effort

Computers
- Many computing resources
- Notebooks, Desktops, Farms, Cloud, HPC Systems

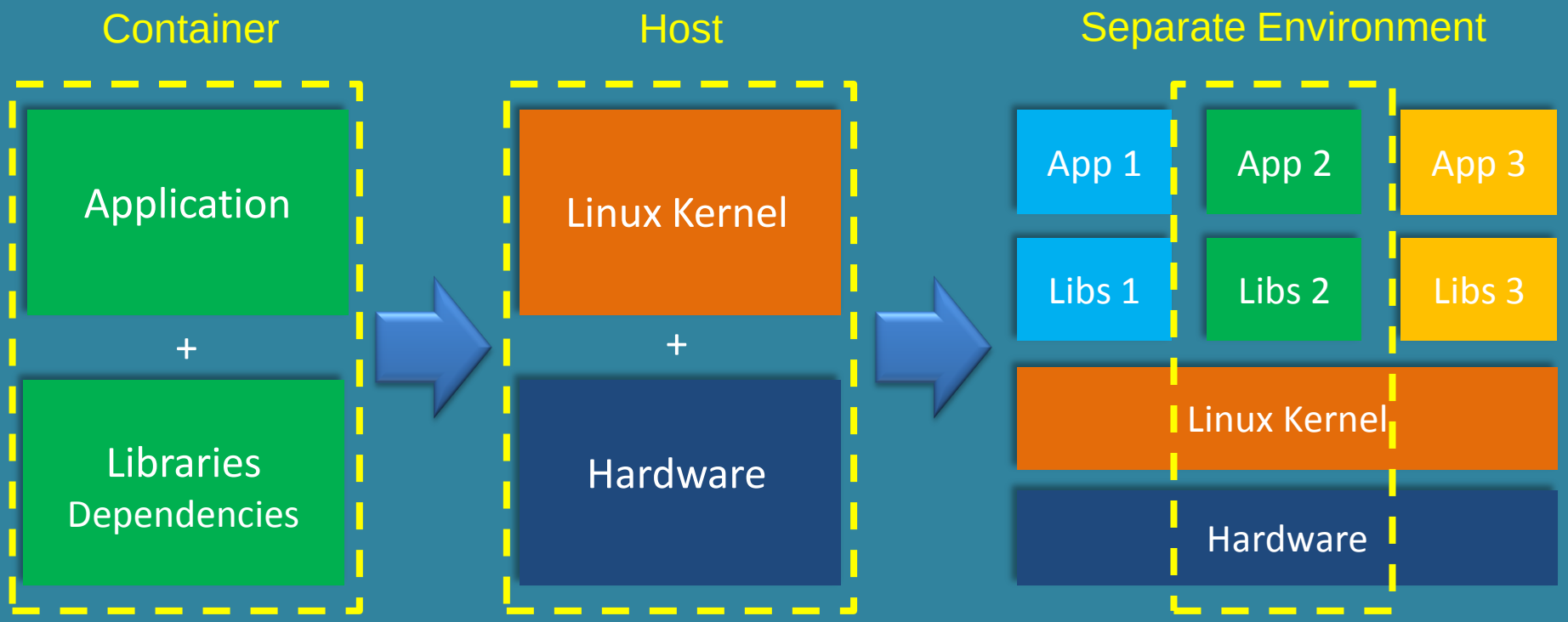
OSes
- Many Linux variants
- Flavors, Distributions, Versions

Environments
- Many computing environments
- Compilers, Libraries, Customizations, Specialized Hardware

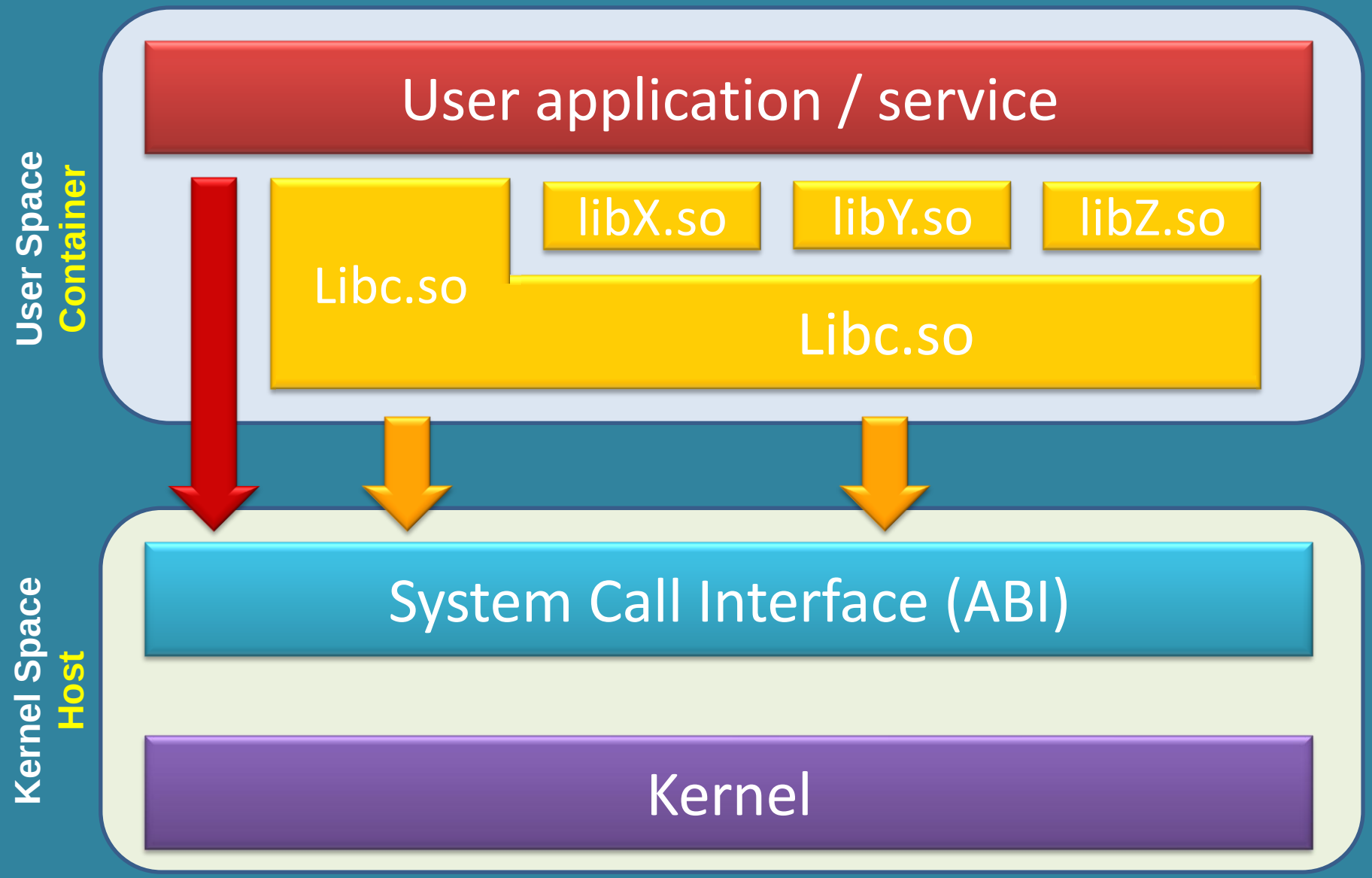
Applications
- Multiple applications
- Portability, Maintainability, Reproducibility across the ecosystem

udocker began to be developed in 2015 indigo-datacloud
Focused on running scientific applications in Linux clusters

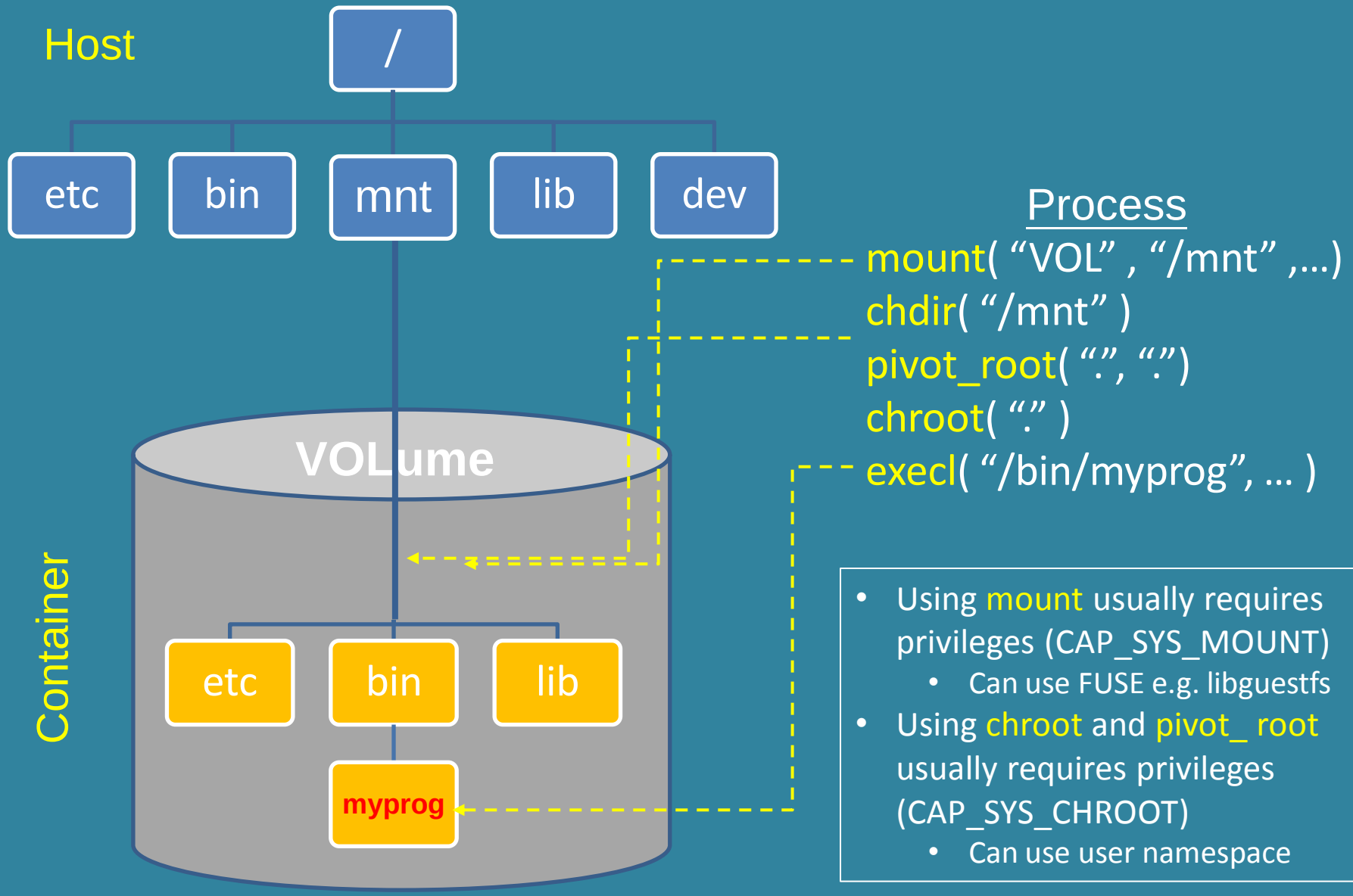
Containers


Container
Application
+
Libraries
Dependencies
Host
Linux Kernel
+
Hardware
Separate Environment
App 1
App 2
App 3
Libs 1
Libs 2
Libs 3
Linux Kernel
Hardware

Linux System Call Interface



chroot


Host
/
etc
bin
mnt
lib
dev
Process
mount(“VOL” , “/mnt” ,…)
chdir(“/mnt”)
pivot_root(“.”, “.”)
chroot(“.”)
execl(“/bin/myprog”, …)
VOLume
Container
etc
bin
lib
myprog
• Using mount usually requires privileges (CAP_SYS_MOUNT)
• Can use FUSE e.g. libguestfs
• Using chroot and pivot_ root usually requires privileges (CAP_SYS_CHROOT)
• Can use user namespace

Advantages for applications

- Encapsulation
  - Applications, dependencies, configurations everything packed together
  - Portability across Linux systems
  - Makes easier the distribution and sharing of ready to use software
- Reproducibility
  - The whole application and run-time environment is in the container
  - Can be easily stored for later replay, reuse and preservation
- Efficiency
  - One single kernel shared by many applications
  - Performance and resource consumption similar to host execution
  - Take advantage of newer more optimized libraries and compilers
- Maintainability
  - Easier application maintenance, distribution and deployment

Containers in short

Run programs as processes in a standard way

No hardware emulation or vm hypervisors

Just separation of process environments

Therefore simple and efficient

UDOCKER

Concept

udocker motivations

UDOCKER

Run applications encapsulated in docker containers:

- without using docker
- without using privileges
- without system administrators intervention
- without compilation or additional system software

and run containers:

- as a normal end-user without requiring privileges
- in Linux systems regardless of OS functionalities
- respecting normal process controls and accounting
- in Linux interactive or batch systems

Empowers end-users to run applications in containers

udocker in 4 steps

UDOCKER

Installation:

- Just get the udocker python code
- No need to install or compile additional software
- No need of system administrator intervention

Get container images:

- **Pull** containers from docker compatible repositories
- **Load** and save docker and OCI formats
- **Import** and export tarballs

Extract from images:

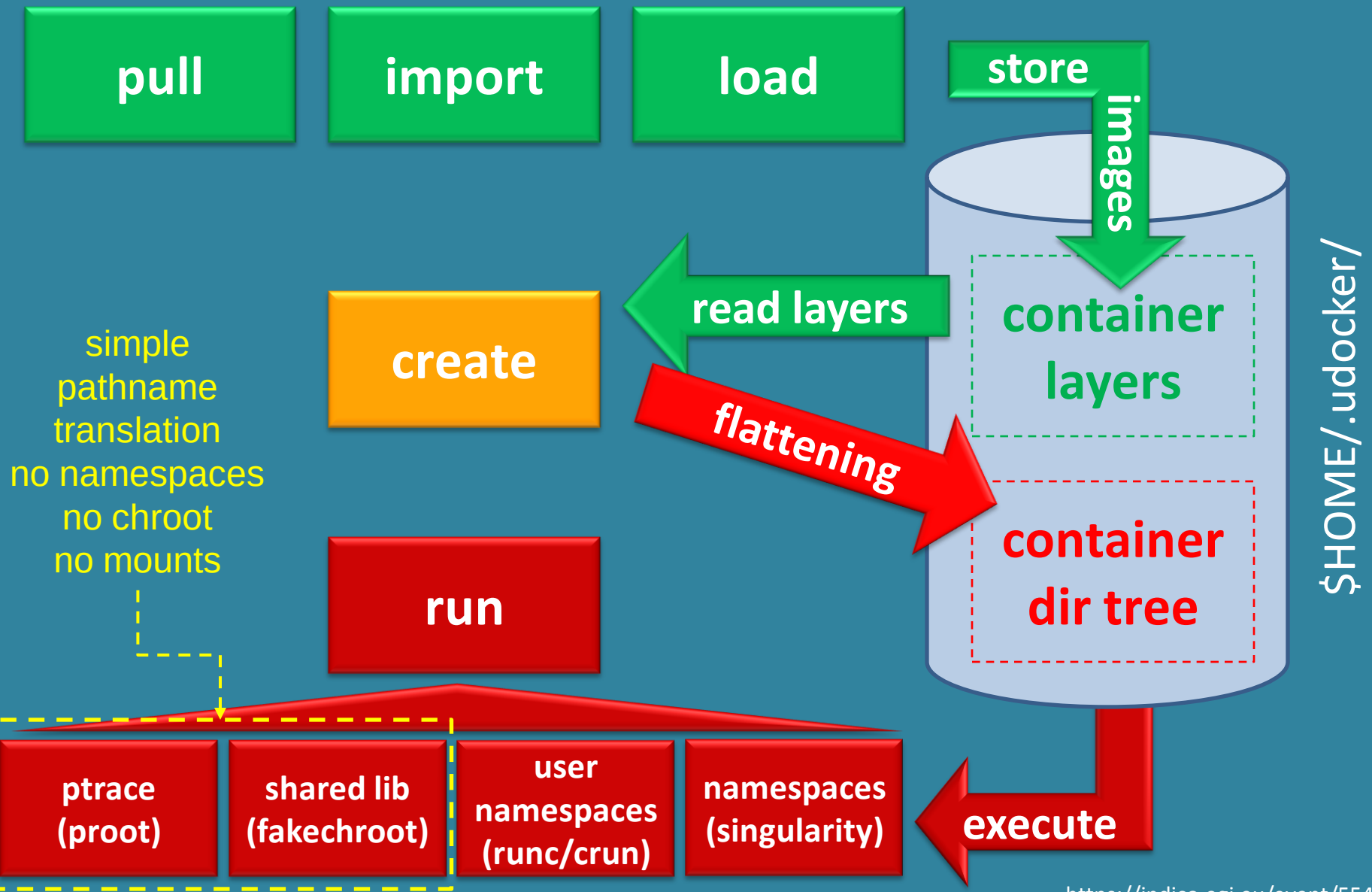
- **Create** the container directory tree from the image

Execute containers:

- **Run** using several execution methods

udocker is an integration tool

UDOCKER



Implementation

UDOCKER

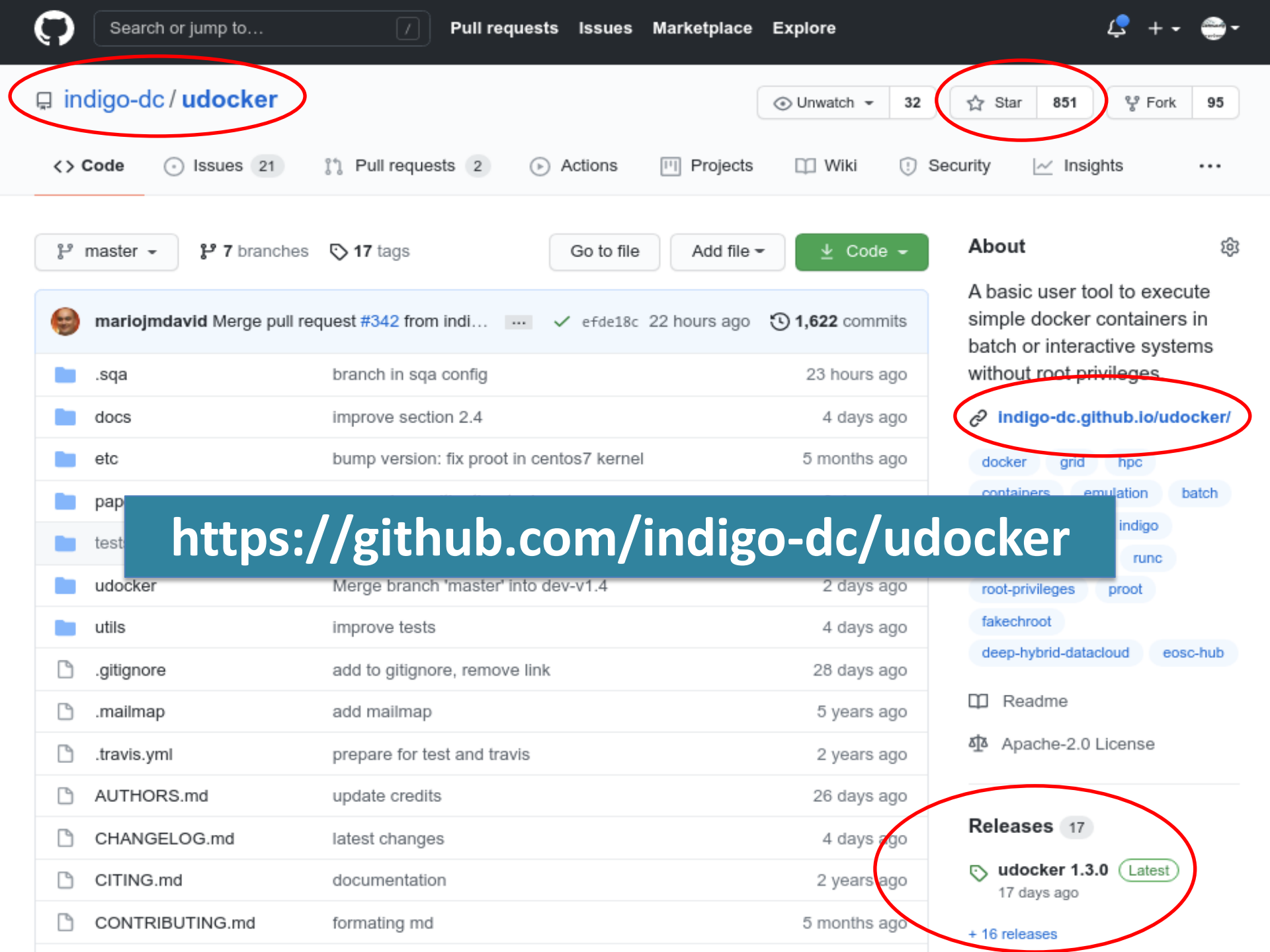
Python code plus several tools and libraries

- Python for portability and easier execution across systems
- External binary tools and libraries to enable execution

- Python code:
  - Command line interface similar to docker
  - Handling of containers (pull, load, import etc)
  - Local repository of images and containers
  - Interface to the execution engines (tools and libraries)
- External tools & libraries modified for udocker:
  - Execution of containers using several engines

UDOCKER

Installation


Search or jump to…
Pull requests
Issues
Marketplace
Explore
indigo-dc / udocker
Unwatch 32
Star 851
Fork 95
Code
Issues 21
Pull requests 2
Actions
Projects
Wiki
Security
Insights
master
7 branches
17 tags
Go to file
Add file
Code
mariojmdavid Merge pull request #342 from indi…
efde18c 22 hours ago
1,622 commits
.sqa branch in sqa config 23 hours ago
docs improve section 2.4 4 days ago
etc bump version: fix proot in centos7 kernel 5 months ago
udocker Merge branch 'master' into dev-v1.4 2 days ago
utils improve tests 4 days ago
.gitignore add to gitignore, remove link 28 days ago
.mailmap add mailmap 5 years ago
.travis.yml prepare for test and travis 2 years ago
AUTHORS.md update credits 26 days ago
CHANGELOG.md latest changes 4 days ago
CITING.md documentation 2 years ago
CONTRIBUTING.md formating md 5 months ago
https://github.com/indigo-dc/udocker
About
A basic user tool to execute simple docker containers in batch or interactive systems without root privileges
indigo-dc.github.io/udocker/
docker
grid
hpc
containers
emulation
batch
indigo
runc
root-privileges
proot
fakechroot
deep-hybrid-datacloud
eosc-hub
Readme
Apache-2.0 License
Releases 17
udocker 1.3.0 Latest
17 days ago
+ 16 releases

README.md

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

UDOCKER

Documentation

The full documentation is available at:

- udocker documentation
  - Installation manual
  - User manual
  - Reference card

2. Installation

2.1. Install from a released version

Download a release tarball from https://github.com/indigo-dc/udocker/releases:

```
wget https://github.com/indigo-dc/udocker/releases/download/
tar zxvf udocker-1.3.0.tar.gz
export PATH=`pwd`/udocker:$PATH
```

Alternatively use `curl` instead of `wget` as follows:

```
curl -L https://github.com/indigo-dc/udocker/releases/downlo
  > udocker-1.3.0.tar.gz
tar zxvf udocker-1.3.0.tar.gz
export PATH=`pwd`/udocker:$PATH
```

udocker executes containers using external tools and libraries that are enhanced and packaged for use with udocker. For more information see section 6 External tools and libraries. Therefore to complete the installation invoke `udocker install` to download and install the required tools and libraries.

Many recent improvements in the documentation

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

UDOCKER

Latest source for Python 3 and Python 2:

- https://github.com/indigo-dc/udocker/tree/**master**
- https://github.com/indigo-dc/udocker/tree/**devel3**

Latest old source for Python 2 only:

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

Releases

- **1.3.0 → latest for Python 3 also works on Python 2**
- 1.1.7 → old code for Python 2 only

Implementation Python 3

UDOCKER

v1.3.0 Supports both Python 3 and Python

- No longer a release prototype it has become stable
- Future new features will only be available in the Python 3

- Differences
  - All new features ported or back-ported between the original Python 2 and the new Python 3, they are now equivalent
  - Differences mostly at internal level where it was redesigned
  - No longer a single large Python script
  - Now modular to ease maintenance and new developments
  - Command line interface retains the same syntax
  - If there are things that don’t work please add an issue

From a given release

UDOCKER

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

udocker 1.3.0

mariojmdavid released this 11 days ago

udocker v1.3.0 see the changelog and the documentation for further information.

- Changelog: https://github.com/indigo-dc/udocker/blob/devel3/CHANGELOG.md
- Documentation: https://indigo-dc.github.io/udocker/
- udocker release for Python 2.6, 2.7 and >= 3.6

Follow this steps to install and run udocker:

```
wget https://github.com/indigo-dc/udocker/releases/download/v1.3.0/udocker-1.3.0.tar.gz
tar zxvf udocker-1.3.0.tar.gz
export PATH=`pwd`/udocker:$PATH
```

From a GitHub release (recommended)

UDOCKER

```
$  wget  \
     https://github.com/indigo-dc/udocker/releases/download/v1.3.0/udocker-1.3.0.tar.gz

   ##  alternatively you can use curl instead of wget

$  curl  -L  \
     https://github.com/indigo-dc/udocker/releases/download/v1.3.0/udocker-1.3.0.tar.gz \
       >  udocker-1.3.0.tar.gz

   ##  untar the Python code. It is extracted to a directory called udocker

$  tar  zxvf  udocker-1.3.0.tar.gz

   ##  optionally add the just created udocker directory to the PATH

$  export  PATH=`pwd`/udocker:$PATH

   ##  install the binaries required to execute containers under $HOME/.udocker

$  udocker  install

$  udocker  version
```

From the source using a GitHub branch

UDOCKER

```
$ git clone https://github.com/indigo-dc/udocker.git
$ cd udocker/udocker

    ## select a different branch
$ git checkout devel3          OR          $ git checkout devel

    ## create a logical link
$ ln -s maincmd.py udocker

    ## optionally add the just created udocker directory to the PATH
$ export PATH=`pwd`:$PATH

    ## install the binaries required to execute containers under $HOME/.udocker
$ udocker install
$ udocker version
```

Using pip

UDOCKER

```
    ##  Create Python 3 virtual env
$   python3  -m  venv  udockervenv
    ##  activate the virtual env
$   source  udockervenv/bin/activate

$   ##  install udocker (v1.3.0)
$   pip  install  udocker

     ##  install the binaries required to execute containers under $HOME./udocker
$   udocker   install
$   udocker   version
```

Without outbound connectivity

UDOCKER

```
##  Get the udocker tarball (same as before, e.g. get a release tarball)

$   wget  \
     https://github.com/indigo-dc/udocker/releases/download/v1.3.0/udocker-1.3.0.tar.gz

##  Get the additional tools (executables, libraries, etc)

$   wget  \
     https://github.com/jorge-lip/udocker-builds/raw/master/tarballs/udocker-englib-1.2.8.tar.gz

##  TRANSFER BOTH TARBALLS TO THE REMOTE SYSTEM
##  once transferred start by installing  udocker

$   tar  zxvf  udocker-1.3.0.tar.gz
$   export  PATH=`pwd`/udocker:$PATH

##  then install the binaries FROM THE TARBALL

$   export  UDOCKER_TARBALL="udocker-englib-1.2.8.tar.gz"

$   udocker  install
```

UDOCKER

Basic Usage

Pull image from DockerHub

UDOCKER

```
$ udocker help

$ udocker pull --help
```

search	pull	create	run	images
rm	rmi	rename	rmname	clone
import	export	load	save	inspect
verify	mkrepo	protect	unprotect	setup
login	logout	help		

- udocker is mainly a run-time to execute containers
- Actual container creation is better performed with docker

Pull image from DockerHub

UDOCKER

```
$ udocker pull ubuntu:18.04
```

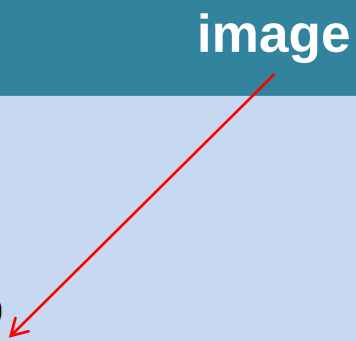
By default pulls from DockerHub

```
Info: downloading layer
   sha256:25fa05cd42bd8fabb25d2a6f3f8c9f7ab34637903d00fd2ed1c1d0fa980427dd
Info: downloading layer
   sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
```

List images in local repository

UDOCKER

```
$ udocker images
```




```
REPOSITORY
fedora:34
fedora:33
ubuntu:19.10
ubuntu:18.04
ubuntu:19.04
ubuntu:20.04
centos:centos7
centos:centos8
centos:centos6
```

Create from image

UDOCKER

```
$ udocker create --name=ub18 ubuntu:18.04
```

container-alias

container-id

```
4c821126-aa28-3731-8d44-eae2f33c6477
```

- After creation the container id is returned
- Can refer to a created container either by id or by the container-alias

List containers in local repository

UDOCKER

```
$ udocker ps
```

container-id

alias

image

```
CONTAINER ID                          P M NAMES          IMAGE
1a0915b2-a8e1-395a-98da-f8dd61530f41  . W ['UB18P2']     ubuntu:18.10
6432f728-8577-3512-a109-0e953f05cd54  . W ['f34']        fedora:34
4c821126-aa28-3731-8d44-eae2f33c6477  . W ['ub18']       ubuntu:18.04
c40ce9b6-5902-3454-a9f0-21534b2c2a9c  . W ['UB18CC']     ubuntu:18.04
b61a6092-aff3-3579-a12c-1e68f5bfa953  . W ['C7C']        centos:centos7
```

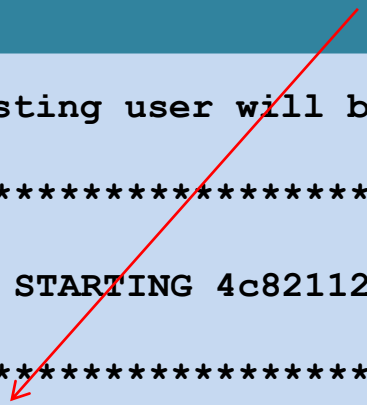
Execute a container

UDOCKER

```
$ udocker run ub18
```

```
$ udocker run 4c821126-aa28-3731-8d44-eae2f33c6477
```

If the container has a default cmd to run
it will be run otherwise starts a shell



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

 ************************************************************************
 *                                                                      *
 *               STARTING 4c821126-aa28-3731-8d44-eae2f33c6477          *
 *                                                                      *
 ************************************************************************
 executing: bash
root@host:~#
```

Execute a container

UDOCKER

```
$ udocker run ub18
```

Ubuntu

root emulation

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

 ****************************************************************************
 *                                                                          *
 *                STARTING 4c821126-aa28-3731-8d44-eae2f33c6477             *
 *                                                                          *
 ****************************************************************************
 executing: bash
root@host:~#
root@host:/# cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"
root@host:/# id
uid=0(root) gid=0(root) groups=0(root),1000(G1000)
```

Run as yourself

UDOCKER

```
$ udocker  run  --user=jorge  -v /home/jorge  \
   -e HOME=/jorge/home  --workdir=/home/jorge  ub18
```

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

*************************************************************************
*                                                                       *
*               STARTING 4c821126-aa28-3731-8d44-eae2f33c6477           *
*                                                                       *
*************************************************************************
executing: bash
jorge@host:~$ id
uid=1000(jorge) gid=1000(G1000) groups=1000(G1000)
jorge@host:~$
jorge@host:~$ pwd
/home/jorge
```

Run as yourself

UDOCKER

```
$ udocker run --user=$USER --bindhome \
  --hostauth ub18
```

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

**************************************************************************
*                                                                        *
*               STARTING 4c821126-aa28-3731-8d44-eae2f33c6477            *
*                                                                        *
**************************************************************************
executing: bash
jorge@host:~$ id
uid=1000(jorge) gid=1000(G1000) groups=1000(G1000)
jorge@host:~$
jorge@host:~$ pwd
/home/jorge
```

- --bindhome will “mount” the user home into the container
- --hostauth will use the host passwd within the container

Less verbosity

UDOCKER

```
$ udocker -q run ub18 /bin/cat /etc/lsb-release
```

Banner not displayed

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"
```

```
$ alias u="udocker -q run --user=$USER --bindhome \
  --hostauth ub18"
$ u /bin/ls
```

Commands at the prompt

UDOCKER

```
$ udocker -q run --user=$USER --bindhome \
  --hostauth ub18  /bin/bash -c "id; pwd"
```

```
$ udocker -q run --user=$USER --bindhome \
  --hostauth --entrypoint="/bin/bash" ub18 \
  -c "id; pwd"
```

Override container provided entrypoint (valid syntax since v1.3.0)

```
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge)
/home/jorge
```

Commands at the prompt

UDOCKER

```
$ udocker -q run --user=$USER --bindhome \
  --hostauth --entrypoint="" ub18 /bin/bash <<EOD
id
pwd
EOD
```

Override container provided entrypoint (valid syntax since v1.3.0)

```
uid=1000(jorge) gid=1000(jorge) groups=1000(jorge)
/home/jorge
```

Duplicate a created container

UDOCKER

```
$ udocker  clone  --name=new18  ub18
```

cloned container-id

cloned container name

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

```
$ udocker  run  new18

$ udocker  run  9fe2f9e7-ce37-3be5-b12d-829a3236d2a6
```

Export and import tarballs as images

UDOCKER

export to tarball

```
$ udocker   export  -o  ub18.tar   ub18

$ udocker   import  ub18.tar    myub18:latest
```

import from tarball

giving a new image name

- Export tarball with the container files (NO container metadata)
- Import tar file as a new image (NOT AS CONTAINER)
- Only the container files are exported, metadata is lost
- Interoperable with Docker

Export and import tarballs as container

UDOCKER

export to tarball

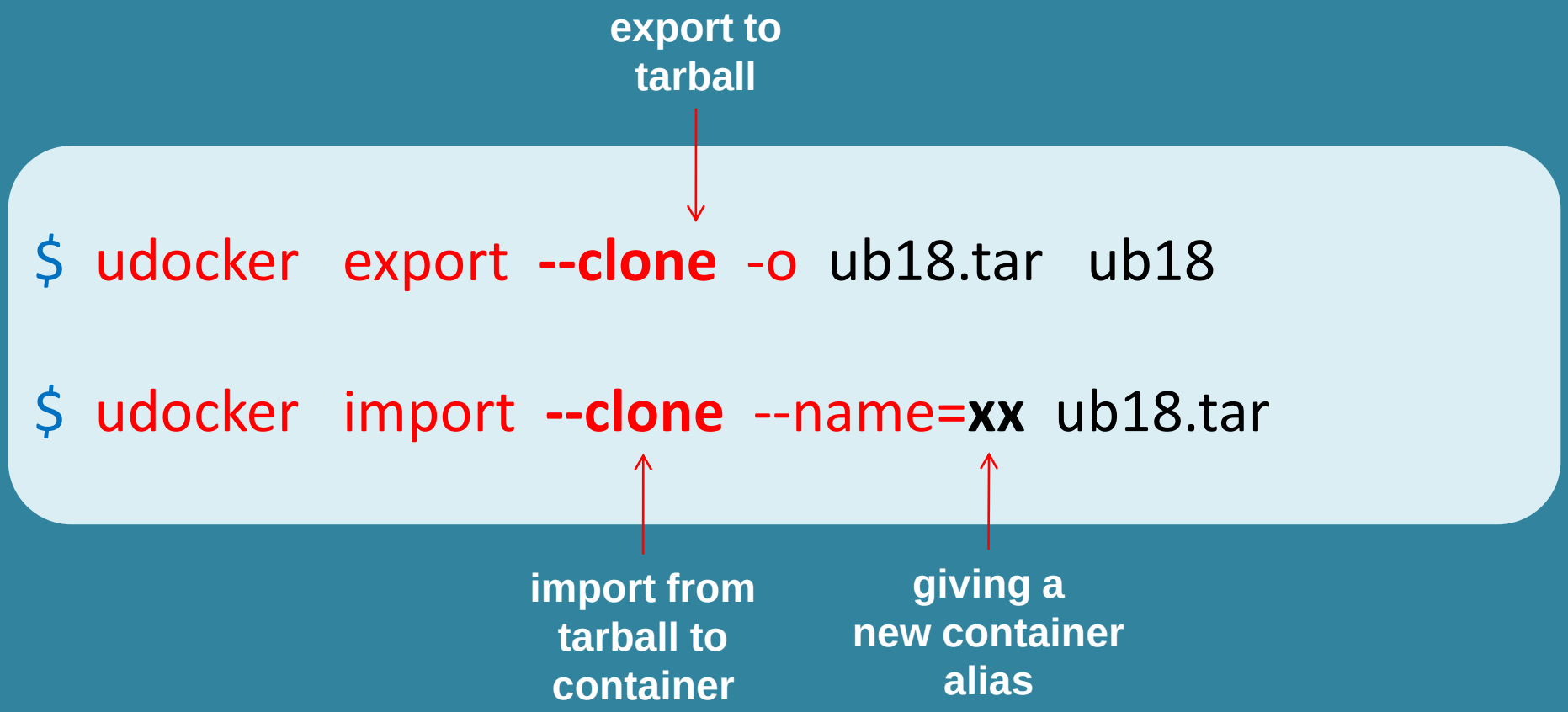
```
$ udocker  export  -o  ub18.tar  ub18

$ udocker  import  --tocontainer  --name=xx  ub18.tar
```

import from tarball directly to a container

giving a container name

- Export tarball with the container files (NO container metadata)
- Import tar file as a newly created CONTAINER
- Interoperable with Docker
- Saves storage space

Export and import as udocker container



- Export uncompress tar file with the container directory tree
- Import tar file as a new CONTAINER
- Both container and metadata including udocker specific configs
- NOT interoperable with Docker

Transfer across machines

UDOCKER

export to tarball to stdout

```
$ udocker  export  --clone  ub18  |  \
  ssh  user@host \
      "udocker  import  --clone  --name=xx  -"
```

import from tarball to container

Read from stdin

- Same as before but piped using SSH
- Export and import in one go
- Some execution modes namely **Fn modes do not work**

Save and load images

UDOCKER

```
$ docker  save  -o  image.tar  centos:centos7

$ udocker  load  -i  image.tar
```

Load image with layers from tarball

Save image with layers to a tarball

- Save image containing layers and metadata to a tarball
- Load the image (NOT AS CONTAINER)
- Can be used to load docker images without pulling

Remove containers and images

UDOCKER

remove container by alias or container-id

```
$ udocker rm -f ub18
$ udocker rm -f 4c821126-aa28-3731-8d44-eae2f33c6477
```

-f force is optional

remove image

```
$ udocker rmi -f ubuntu:18.04
```

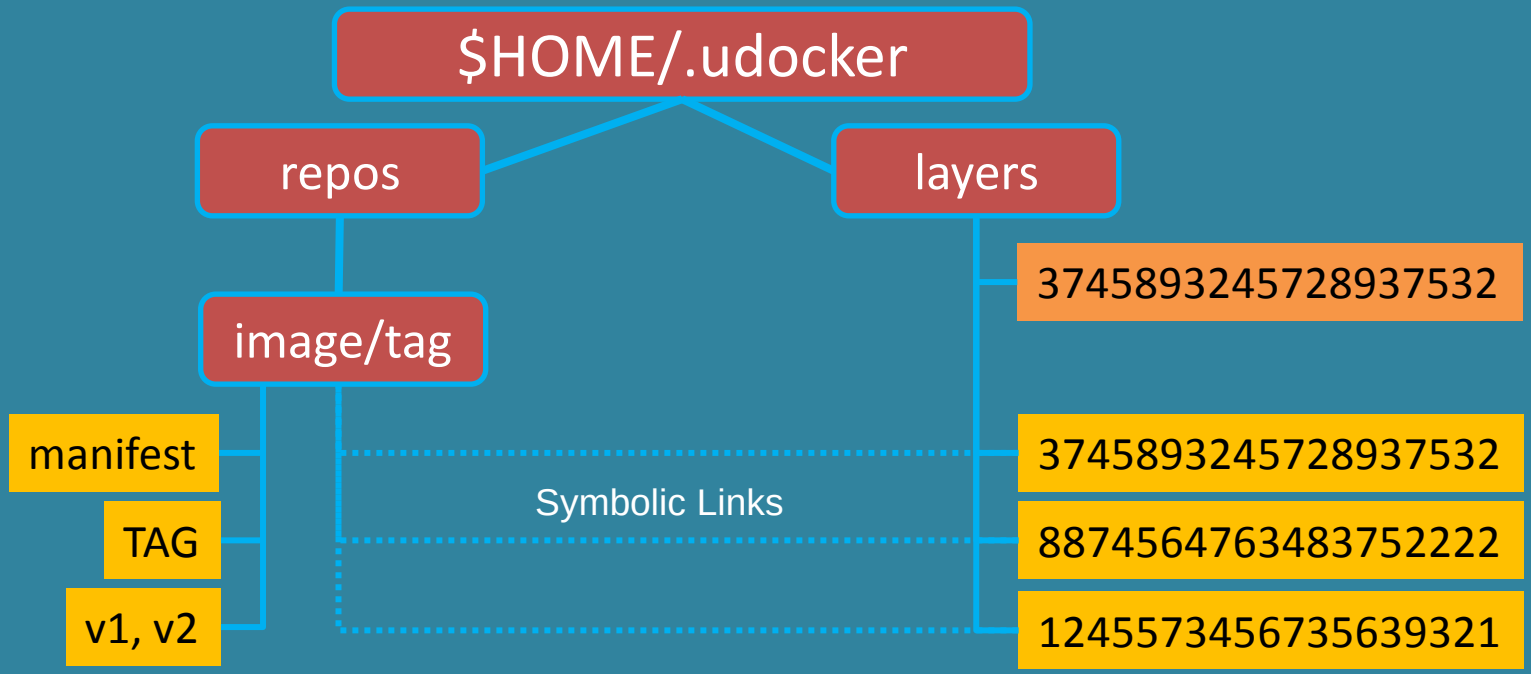
UDOCKER

Advanced Topics

udocker pull

UDOCKER

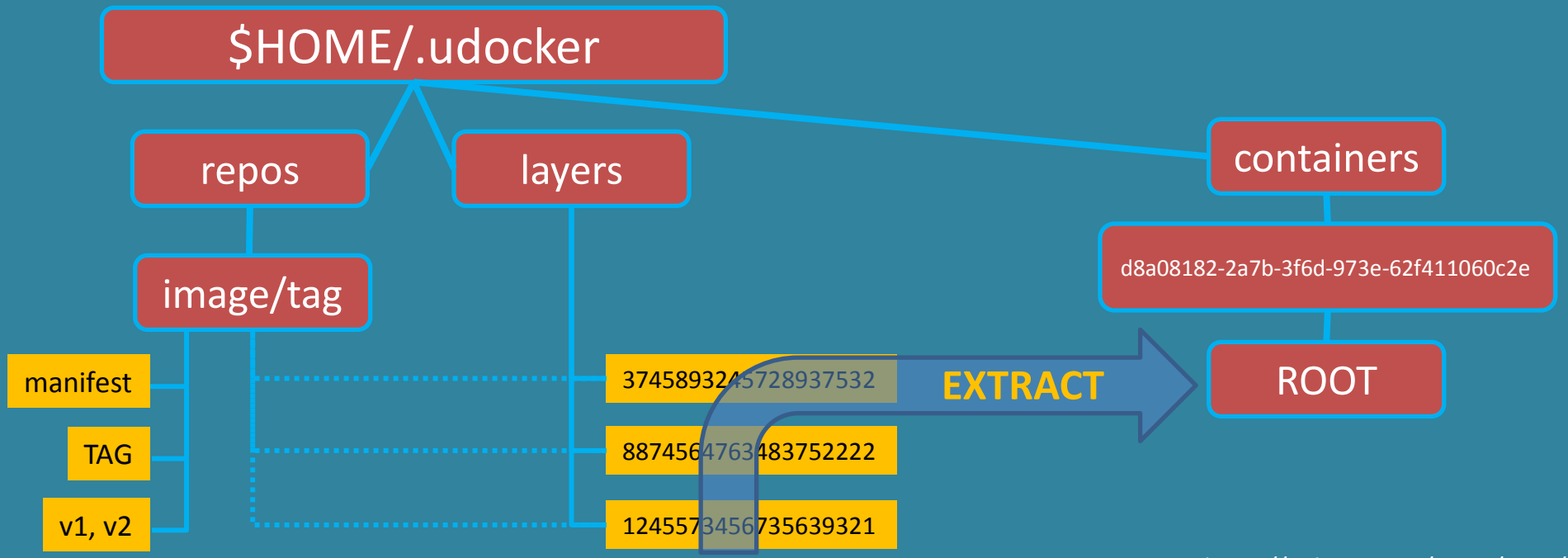
- Images
  - Layers and metadata are pulled with DockerHub REST API
  - Image metadata is parsed to identify the layers
  - Layers are stored in the use home directory under ~/.udocker/layers so that can be shared by multiple images



udocker create

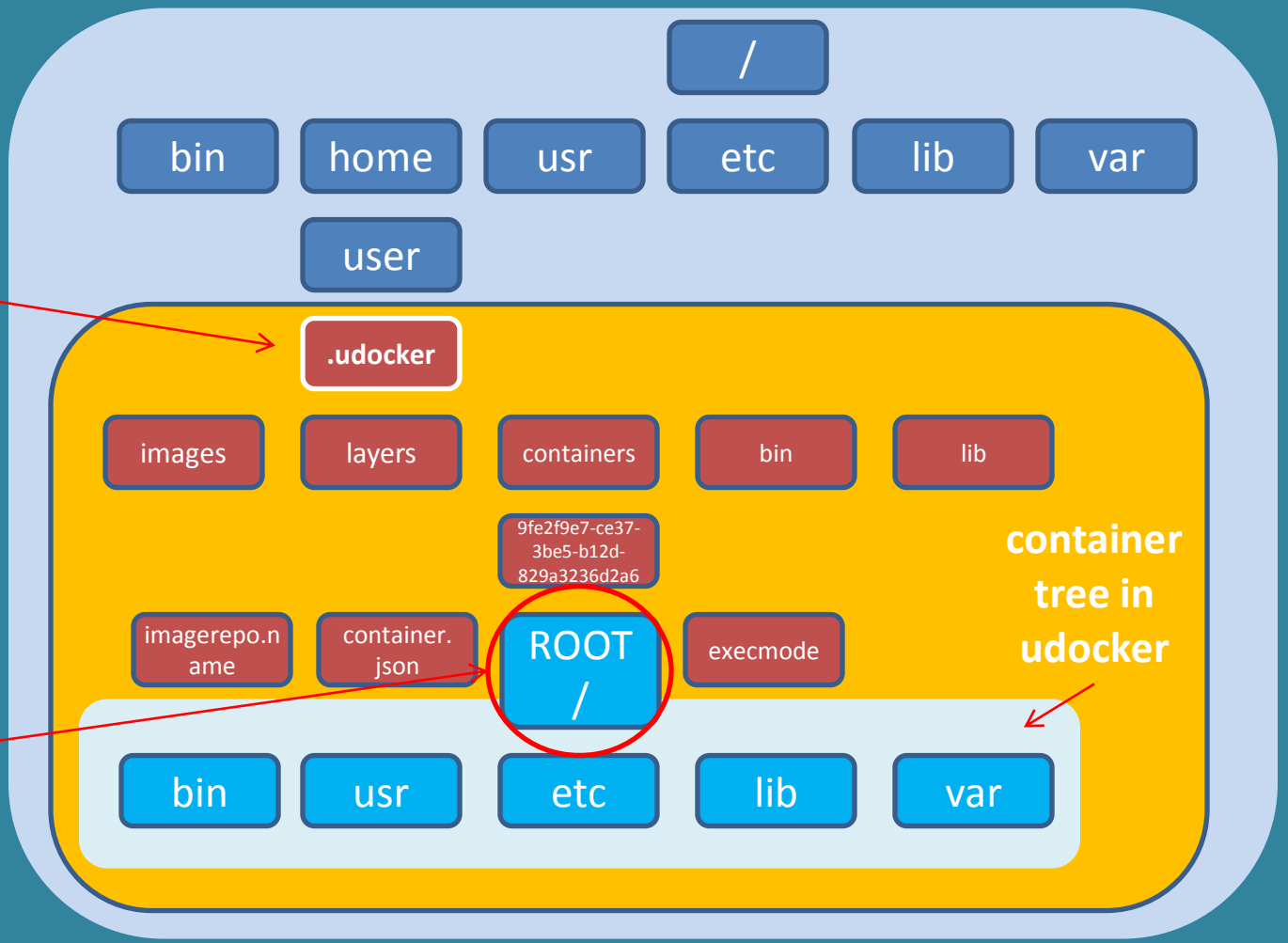
UDOCKER

- 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 run

- Execution
- chroot-like



Where is my container

UDOCKER

remove container by alias or container-id

```
$ udocker inspect -p ub18
```

-p to print the container pathname

```
/home/jorge/.udocker/containers/f80f88de-3227-3cba-8551-cd62ddb14174/ROOT
```

```
$ ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

Execution engines

UDOCKER

- udocker integrates several execution methods:
  - Supports several engines to execute containers
  - 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 / Crun	rootless unprivileged using user namespaces
R2	runC / Crun	rootless unprivileged using user namespaces + P1
R3	runC / Crun	rootless unprivileged using user namespaces + P2
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

Change the execution engine

UDOCKER

Execution mode

```
$ udocker setup --execmode=F3 ub18

$ udocker setup --execmode=P1 ub18
```

Default execution mode

- Execution modes are identified by one letter + one digit
- Execution mode is defined per extracted container not image
- Some modes may take a long time to setup (i.e. F3, F4)

List containers and execution engine

UDOCKER

List execution mode for each container

```
$ udocker ps -m
```

```
CONTAINER ID                          P M MOD NAMES          IMAGE
1a0915b2-a8e1-395a-98da-f8dd61530f41  . W F2  ['UB18P2']     ubuntu:18.10
6432f728-8577-3512-a109-0e953f05cd54  . W P1  ['f34']        fedora:34
4c821126-aa28-3731-8d44-eae2f33c6477  . W R1  ['ub18']       ubuntu:18.04
c40ce9b6-5902-3454-a9f0-21534b2c2a9c  . W P2  ['UB18CC']     ubuntu:18.04
b32db50b-ecfd-3801-bd40-a7ba64b6823b  . W P1  ['BUSY']       busybox:latest
b61a6092-aff3-3579-a12c-1e68f5bfa953  . W F4  ['C7C']        centos:centos7
```

Engine Pn : PRoot

UDOCKER

- 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

- P1 mode uses PTRACE + SECCOMP filtering, to minimize 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
- P2 uses PTRACE without SECCOMP ➔ traces all calls ➔ slower

- The impact of tracing depends on the system call frequency

Engine Fn : Fakechroot

UDOCKER

- Uses LD_PRELOAD to intercept shared library 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

- Generally higher performance than Pn and even other namespace based engines
- Uses heavily modified Fakechroot for both glibc and musl libc
- Requires changes to files in the container
- Can run inside namespaces to provide nested containers
- There are 4x Fn modes (F1, F2, F3 and F4)
- Does not support root emulation

Engine Fn : Fakechroot all modes

UDOCKER

- F1
  - Forces 1st argument to be the container ld.so
  - Populates the LD_LIBRARY_PATH with container libs
- F2
  - Same as F1
  - Loading from host paths /lib, /lib64 etc is disabled
  - Loading from ld.so.cache is disabled
- F3
  - Changes the ELF headers of executables and libraries
    - to use the ld.so from the container
    - to direct shared library loading to the container
- F4
  - Same as F3 but changes objects dynamically

Containers using kernel features

UDOCKER

- **chroot, pivot_root**: make a given directory root of the file system
- **Kernel namespaces**: isolate system resources from process
  - **Mount**: isolate mount points (cannot see host or other containers mounts)
  - **UTS**: virtualize hostname and domain
  - **IPC** : inter process communications isolation (semaphores, shmem, msgs)
  - **PID**: isolate and remap process identifiers (cannot see other processes)
  - **Network**: isolate network resources (interfaces, tables, firewall etc)
  - **cgroup**: isolate cgroup directories
  - **Time**: virtualize boot and monotonic clocks
  - **User**: isolate and remap user/group identifiers (user can be a limited root)
- **cgroups**: process grouping and resource consumption limits
- **seccomp**: system call filtering
- **POSIX capabilities**: split and drop root privileges
- **AppArmor** and **SELinux**: kernel access control

Linux user namespace

UDOCKER

Available on fairly “recent” kernels/distributions

- Allows an unprivileged user to have a different UID/GID
- Enables an unprivileged user to become UID/GID **0** root
- Enables executing pivot_root, chroot and other calls

- May require some setup of subuid and subgid files
- Network namespace becomes useless only loopback
- root has limitations
  - Cannot creates devices (mknod)
  - Cannot load kernel modules
  - Mount is restricted to some file system types
  - Issues on changing and handling user ids group ids
  - Accessing files in the host (mount bind) can become problematic
- Not enabled in some distributions (RedHat/CentOS)

Engine Rn : runc & crun

UDOCKER

- runc & crun are tools to spawn containers according to the Open Containers Initiative (OCI) specification
  - Support unprivileged namespaces using the user namespace
  - User namespace has several limitations but allows execution without privileges by normal users
  - Limited support for devices mapping

- We added mapping of Docker metadata to OCI
- udocker can produce an OCI spec and run the containers with runc or crun transparently

Engine Sn : Singularity

UDOCKER

- Uses Singularity if locally available in the host system
- Enables taking advantage of Singularity to run udocker created containers

- Singularity is not provided with the udocker tools
- Singularity is difficult to compile statically
- This udocker engine uses Singularity in sandbox mode

NVIDIA GPUs

UDOCKER

```
$ udocker  setup  --nvidia  ub18
```

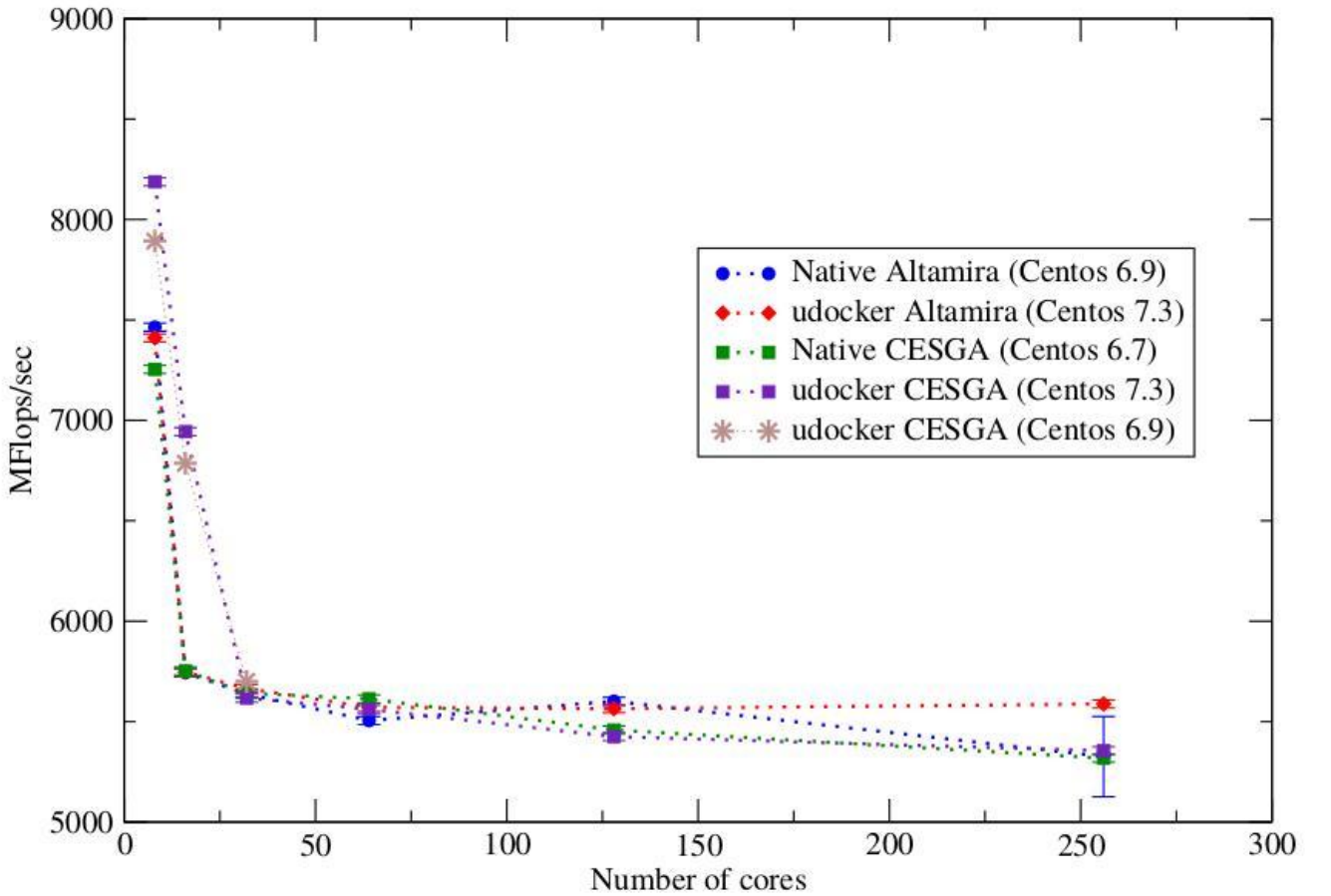
- Enables use of NVIDIA GPUs
- Similar to nvidia-docker
- Copies the nvidia libraries from the host to the container

UDOCKER

Benchmarks

Lattice QCD

UDOCKER



by Isabel Campos (IFCA/CSIC)

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

Lattice QCD with OpenMPI

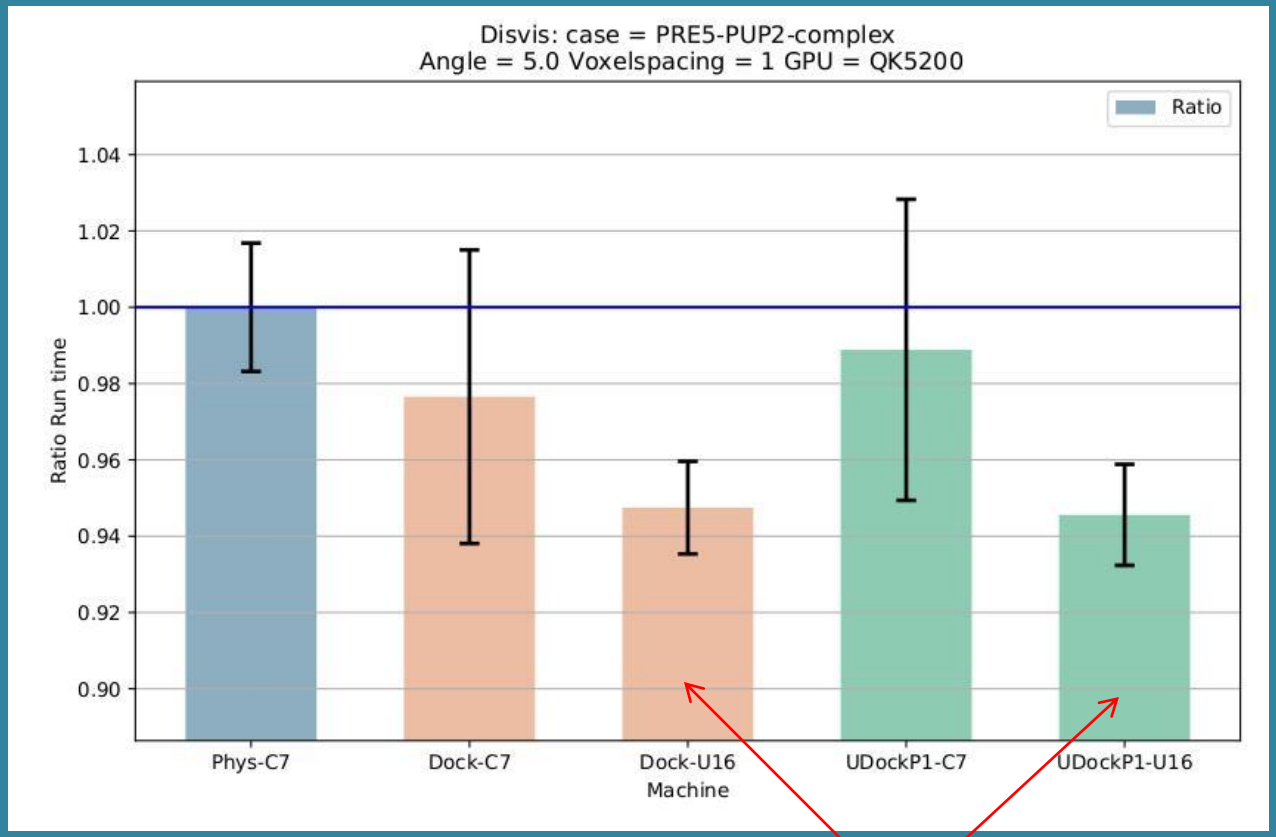
UDOCKER

```
$ mpiexec -np 256  udocker  run \
      -e LD_LIBRARY_PATH=/usr/lib \
      --hostenv \
      --hostauth \
      --user=$USER \
      -v /tmp \
      --workdir=/opt/projects/openQCD-1.6/main \
      openqcd  \
      /opt/projects/openQCD-1.6/main/ym1 \
      -i  ym1.in  -noloc
```

by Isabel Campos (IFCA/CSIC)

Biomolecular complexes

UDOCKER



DisVis is being used in production with udocker

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

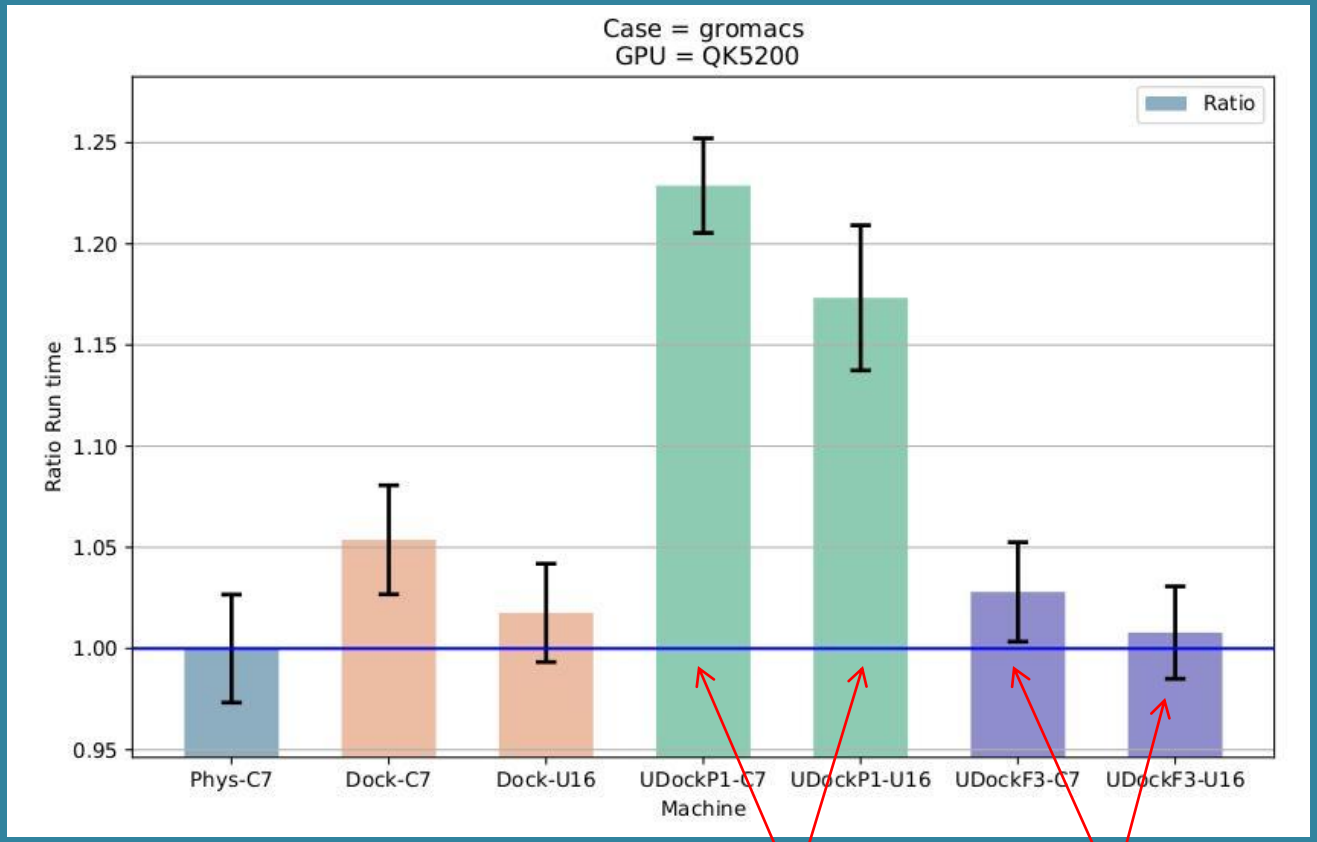
Using OpenCL and NVIDIA GPGPUs

udocker in P1 mode

Better performance with Ubuntu 16 container

Molecular dynamics

UDOCKER



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

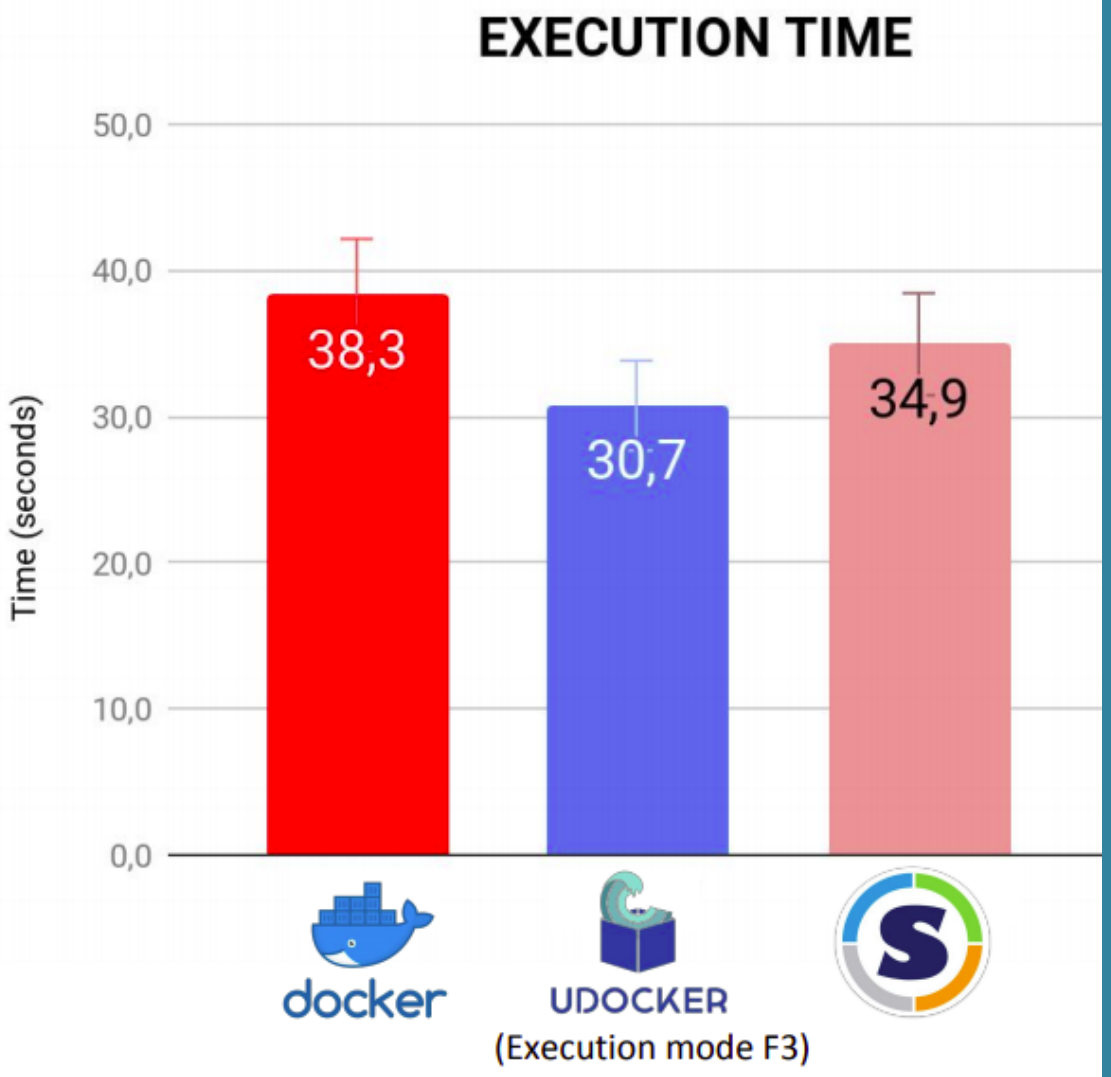
TensorFlow

UDOCKER

Container:

- Latest GPU version of Tensorflow (from Docker Hub).
- Train a model to recognize handwritten digits (the MNIST data set).

https://github.com/tensorflow/models.git



Grid y Computación de Altas Prestaciones GRyCAP

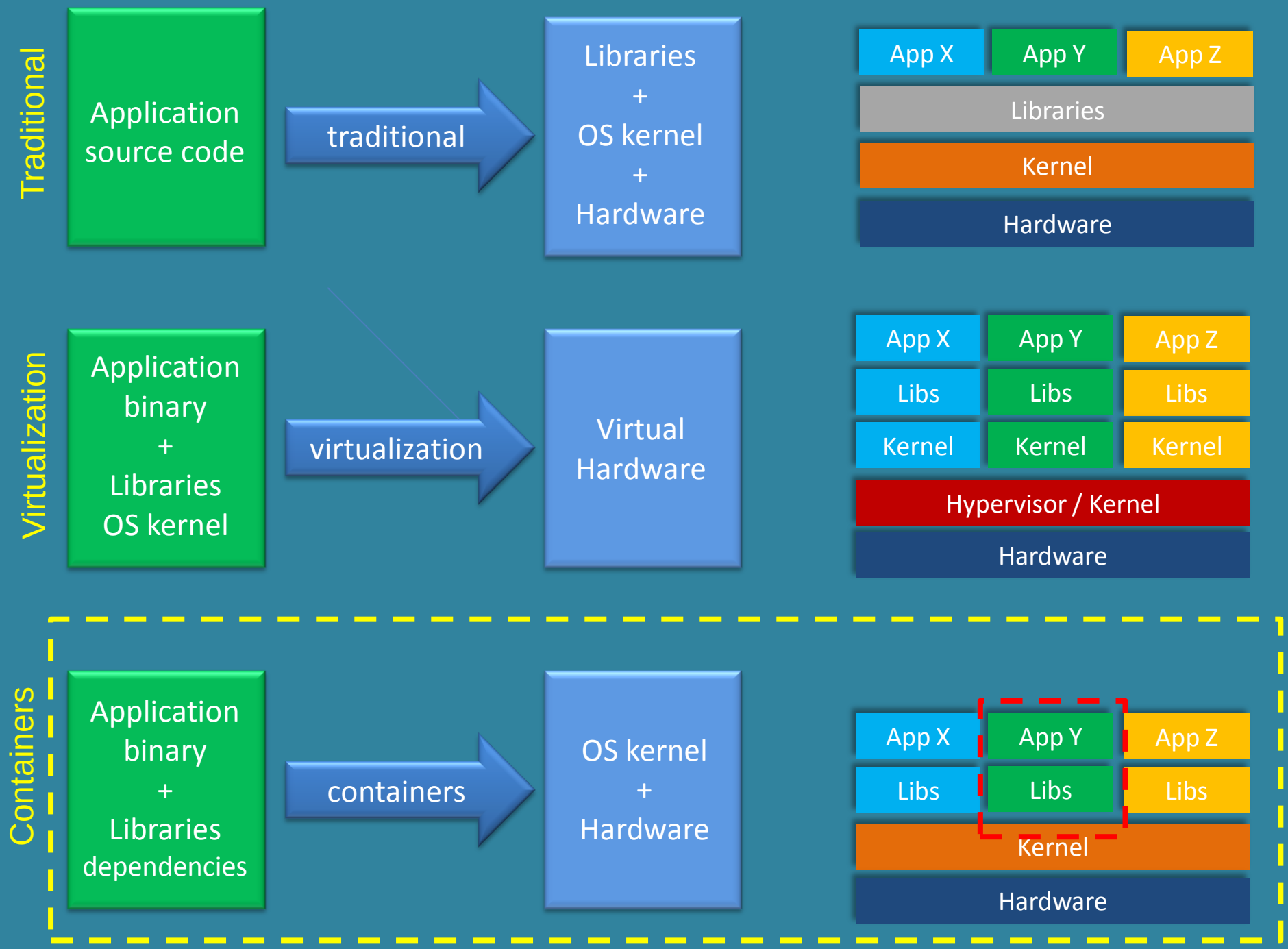
UPV

Thank you !

Questions ?

udocker@lip.pt

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


Traditional
Application source code
traditional
Libraries + OS kernel + Hardware
App X
App Y
App Z
Libraries
Kernel
Hardware
Virtualization
Application binary + Libraries OS kernel
virtualization
Virtual Hardware
App X
App Y
App Z
Libs
Libs
Libs
Kernel
Kernel
Kernel
Hypervisor / Kernel
Hardware
Containers
Application binary + Libraries dependencies
containers
OS kernel + Hardware
App X
App Y
App Z
Libs
Libs
Libs
Kernel
Hardware

Selection in terms of performance

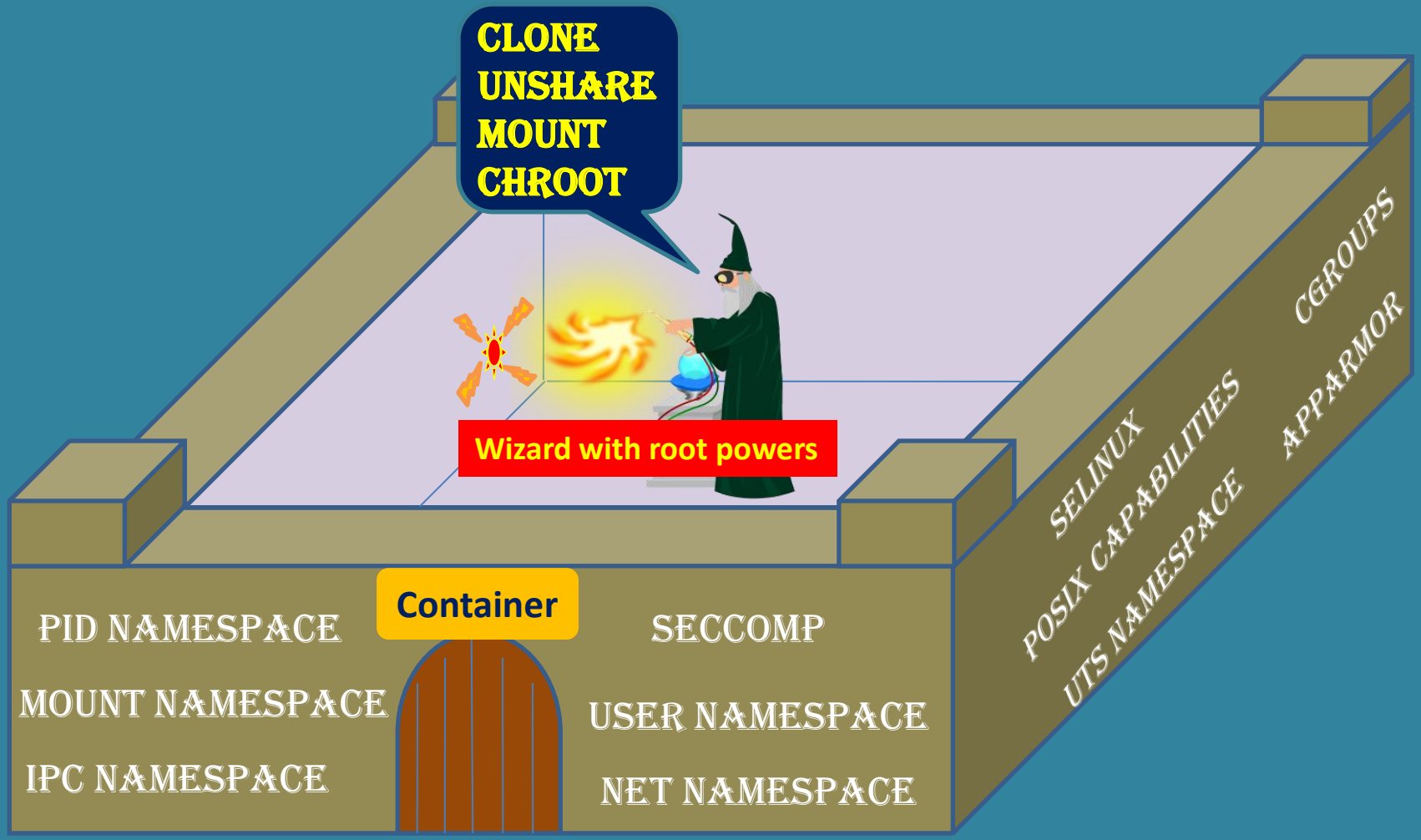
UDOCKER

Mode	Base	Description
P1	PRoot	Multithreaded applications can suffer degradation
P2	PRoot	Same limitations as P1 apply All system calls are traced causing higher overheads than P1
R1	runC / Crun	Same performance as namespace based applications
R2	runC / Crun	Only for software installation and similar Same performance as P1
R3	runC / Crun	Only for software installation and similar Same performance as P2
F1	Fakechroot	All Fn modes have similar performance during execution Frequently the Fn modes are the fastest
F2	Fakechroot	Same as F1
F3	Fakechroot	Same as F1 Setup can be very slow
F4	Fakechroot	Same as F1 Setup can be very slow
S1	Singularity	Similar to Rn

Selection in terms of interoperability

Mode	Base	Description
P1	PRoot	PTRACE + SECCOMP requires kernel >= 3.5 Can fall back to P2 if SECCOMP is unavailable
P2	PRoot	Runs across a wide range of kernels even old ones Can run with kernels and libraries that would fail with kernel too old
R1	runC / Crun	User namespace limitations apply
R2	runC / Crun	User namespace limitations apply Same limitations as P1 also apply, this is a nested mode P1 over R
R3	runC / Crun	User namespace limitations apply Same limitations as P2 also apply, this is a nested mode P2 over R
F1	Fakechroot	May load host libraries Requires shared library compiled against same libc as in container
F2	Fakechroot	Same as F1
F3	Fakechroot	Requires shared library compiled against same libc as in container Binary executables and libraries get tied to the user HOME pathname
F4	Fakechroot	Same as F3 Executables and libraries can be compiled or added dynamically
S1	Singularity	Must be available on the system might use user namespaces or chroot

Security when using kernel features



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

Mapping host /tmp to container /tmp activates code to minimize

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

Search for repositories

```
$ udocker search centos
```

```
NAME                                OFFICIAL DESCRIPTION                                                    STARS
centos                              [OK] The official build of CentOS.                                      6611
pivotaldata/centos-gpdb-dev         ---- CentOS image for GPDB development. Tag names often have GCC because we    13
pivotaldata/centos-mingw            ---- Using the mingw toolchain to cross-compile to Windows from CentOS          3
jdeathe/centos-ssh                  ---- OpenSSH / Supervisor / EPEL/IUS/SCL Repos - CentOS.                 118
pivotaldata/centos                  ---- Base centos, freshened up a little with a Dockerfile action          5
ansible/centos7-ansible             ---- Ansible on Centos7                                                   134
consol/centos-xfce-vnc              ---- Centos container with "headless" VNC session, Xfce4 UI and preinstalle   129
pivotaldata/centos-gcc-toolchain    ---- CentOS with a toolchain, but unaffiliated with GPDB or any other parti     3
smartentry/centos                   ---- centos with smartentry                                               0
kinogmt/centos-ssh                  ---- CentOS with SSH                                                     29
centos/systemd                      ---- systemd enabled base container.                                     99
imagine10255/centos6-lnmp-php56     ---- centos6-lnmp-php56                                                  58
blacklabelops/centos                ---- CentOS Base Image! Built and Updates Daily!                         1
drecom/centos-ruby                  ---- centos ruby                                                         6
darksheer/centos                    ---- Base Centos Image -- Updated hourly                                 3
tutum/centos                        ---- Simple CentOS docker image with SSH access                          48
pivotaldata/centos6.8-dev           ---- CentosOS 6.8 image for GPDB development                              1
centos/tools                        ---- Docker image that has systems administration tools used on CentOS Atom     7
guyton/centos6                      ---- From official centos6 container with full update, plus tweaks for cent    10
pivotaldata/centos7-dev             ---- CentosOS 7 image for GPDB development                                0
mcnaughton/centos-base              ---- centos base image                                                    1
amd64/centos                        ---- The official build of CentOS.                                       2
```

List tags

```
$ udocker search --list-tags centos
```

```
5.11
5
6.10
6.6
6.7
6.8
6.9
6
7.0.1406
7.1.1503
7.2.1511
7.3.1611
7.4.1708
7.5.1804
7.6.1810
7.7.1908
7.8.2003
7.9.2009
7
8.1.1911
8.2.2004
8.3.2011
8
centos5.11
centos5
centos6.10
centos6.6
centos6.7
centos6.8
centos6.9
centos6
centos7.0.1406
centos7.1.1503
centos7.2.1511
centos7.3.1611
centos7.4.1708
centos7.5.1804
centos7.6.1810
centos7.7.1908
centos7.8.2003
centos7.9.2009
centos7
centos8.1.1911
centos8.2.2004
centos8.3.2011
centos8
latest
```

- This is specific to docker hub and may not work with other registries

Pull from other registries

Registry as first element

```
$ udocker search quay.io/centos
```

Registry as first element

```
$ udocker pull quay.io/centos/centos:latest
```

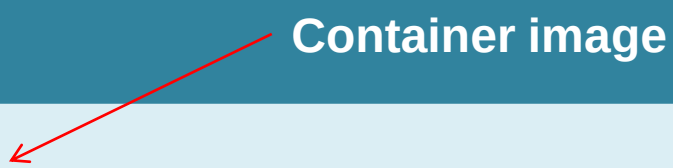
```
Info: downloading layer
    sha256:7a0437f04f83f084b7ed68ad9c4a4947e12fc4e1b006b38129bac89114ec3621
Info: downloading layer
    sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Info: downloading layer
    sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
```

Show the udocker config

```
$ udocker   showconf
```

```
--------------------------------------------------------------------------------
          Configuration options
verbose_level = 3
topdir = /home/jorge/.udocker
homedir = /home/jorge/.udocker
bindir = None
libdir = None
docdir = None
reposdir = None
layersdir = None
containersdir = None
tarball_release = 1.2.8
tarball = https://download.ncg.ingrid.pt/webdav/udocker/udocker-englib-1.2.8.tar.gz
    https://raw.githubusercontent.com/jorge-lip/udocker-builds/master/tarballs/udocker-engli
    1.2.8.tar.gz
installinfo = ['https://raw.githubusercontent.com/indigo-dc/udocker/master/messages']
installretry = 3
autoinstall = True
config = udocker.conf
keystore = keystore
tmpdir = /tmp
cmd = ['/bin/bash', '-i']
root_path = /usr/sbin:/sbin:/usr/bin:/bin
```

Verify a pulled, imported or loaded image




```
$ udocker  verify  ubuntu:18.04
```

```
Info: verifying: ubuntu:18.04
Info: loading structure
Info: verifying layers
Info: layer ok: sha256:25fa05cd42bd8fabb25d2a6f3f8c9f7ab34637903d00fd2ed1c1d0fa980427dd
Info: layer ok: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Info: image Ok
```

List all image files

```
$ udocker   images  -l
```

```
REPOSITORY
ubuntu:18.04                                                    .
  /home/jorge/.udocker/repos/ubuntu/18.04
    /sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4 (1 MB)
    /sha256:25fa05cd42bd8fabb25d2a6f3f8c9f7ab34637903d00fd2ed1c1d0fa980427dd (25 MB)
myub18:latest                                                   .
  /home/jorge/.udocker/repos/myub18/latest
    /69baf66d7e399b1c9869a5107119e452b664bb19e468cd07fdefa83b1a994a25.json (1 MB)
    /69baf66d7e399b1c9869a5107119e452b664bb19e468cd07fdefa83b1a994a25.layer (65 MB)
centos:centos8                                                  .
  /home/jorge/.udocker/repos/centos/centos8
    /sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4 (1 MB)
    /sha256:7a0437f04f83f084b7ed68ad9c4a4947e12fc4e1b006b38129bac89114ec3621 (71 MB)
centos:centos7                                                  .
  /home/jorge/.udocker/repos/centos/centos7
    /sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4 (1 MB)
    /sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc (72 MB)
```

Inspecting metadata

Container id or name

```
$ udocker  inspect  ub18
```

Container image

```
$ udocker  inspect  ubuntu:18.04
```


```
{
    "architecture": "amd64",
    "config": {
        "AttachStderr": false,
        "AttachStdin": false,
        "AttachStdout": false,
        "Cmd": [
            "bash"
        ],
        "Domainname": "",
        "Entrypoint": null,
        "Env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
        ],
        "Hostname": "",
        "Image": "sha256:d207f9055fcf01ac5c8231b88b56098a667475e94547cedc334c8c7c2ef8f22b",
```