EUROCC PORTUGAL

udocker developments

LABORATÓRIO DE INSTRUMENTAÇÃO
E FÍSICA EXPERIMENTAL DE PARTÍCULAS

Jorge Gomes / Mário David
udocker@lip.pt

Help users run in heterogeneous environments

Run applications across Linux systems

Avoiding dependencies on software and sysadmins

Integration of several tools suitable for containers execution

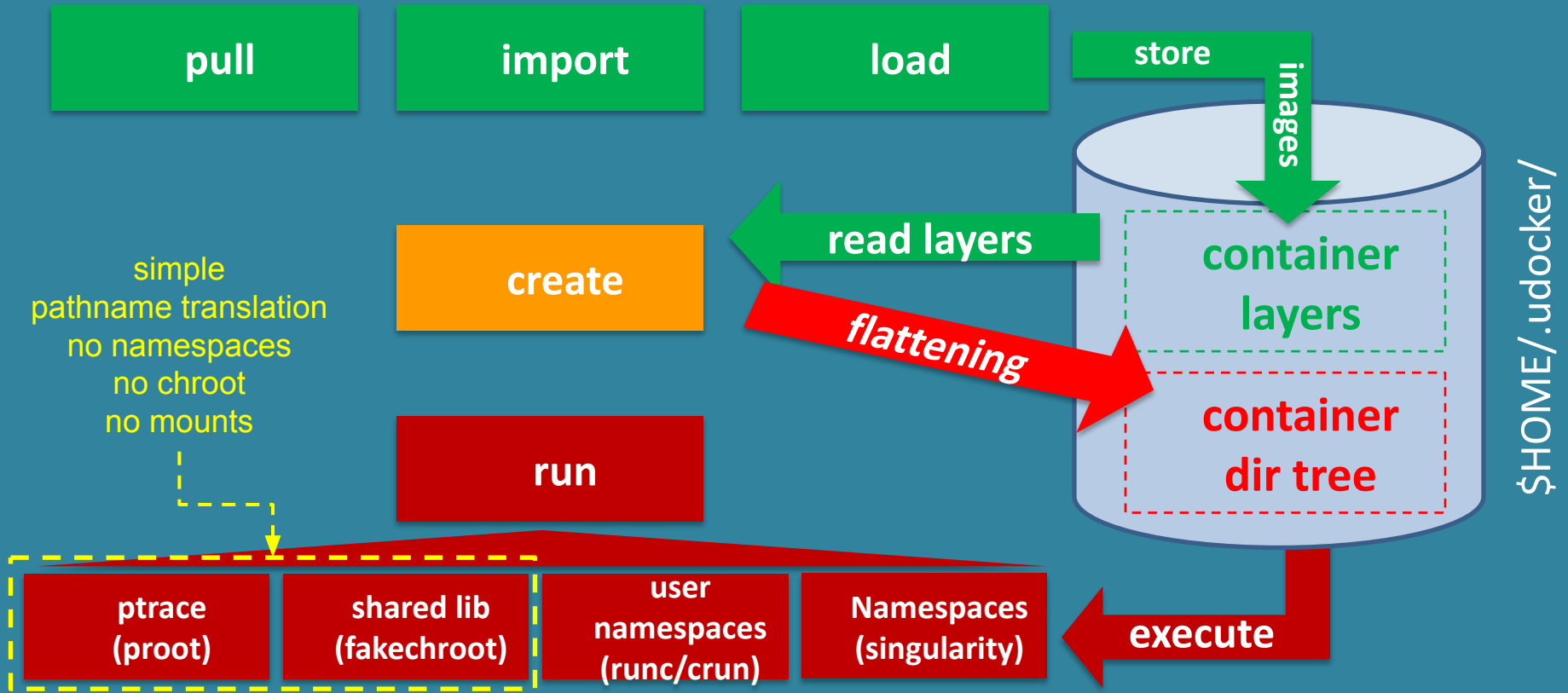
Support multiple execution methods beyond namespaces

A tool to execute containers that is easy deploy

Providing privilege-less methods to execute containers

Empower end-users to leverage containers

udocker in a nutshell



udocker components

- **Python code (latest 1.3.10)**
  - **Command line interface**
  - **Dockerhub API**
  - **Container and image handling: import, load, save and export**
  - **Local images repository**
  - **Containers extraction from images and handling**
  - **Interface with the execution engines**
  - **Handling of nvidia GPU drivers**
- **udocker tools (latest 1.2.10)**
  - **Tarball pulled and Installed upon first invocation or with udocker install**
  - **Contains binary executables and libraries that implement the engines**
  - **Supporting different OSes and hardware architectures**
  - **Executables: proot (Pn), runc (Rn), crun (Rn) and patchelf (Fn)**
  - **Libraries: fakechroot (Fn)**

udocker in 4 steps

1) **Installation:**
   - **get** the udocker python code
   - **untar** the python code into your home directory
   - **udocker install** to copy and unpack the execution engines
2) **Get a container image:**
   - **udocker pull** to get containers from docker compatible repositories
   - **udocker load** to load images in docker and OCI formats
   - **udocker import** to import images from tarballs
3) **Extract the image content:**
   - **udocker create** to extract the container directory tree from the image
4) **Execute applications from the image:**
   - **udocker run** to execute

Product Solutions Open Source Pricing

Search or jump to... / Sign in Sign up

indigo-dc / udocker Public

Notifications Fork 117 Star 1.1k

Code Issues 21 Pull requests Actions Projects Security Insights

master 11 branches 25 tags Go to file Code

mariojmdavid new badge assessment f50104a on Jul 3 1,934 commits

.sqa	update branch in .sqa config	6 months ago
docker_sqaaastools	fix flake and unit tests	last year
docs	document centos 7 issues with user namespaces	3 months ago
etc	update variables in udocker.conf	2 years ago
pa		
te		
ud		
utils	add proot for arm 32bit	3 months ago
.flake8	work on pylinting flake8	last year
.gitignore	add to gitignore, remove link	2 years ago
.mailmap	add mailmap	7 years ago
.mdlrc	full sqaaas pipeline, includes new scripts	last year
.travis.yml	prepare for test and travis	4 years ago
AUTHORS.md	documentation	3 months ago
CHANGELOG.md	code style fixes	3 months ago
CITING.md	fix markdownlint errors and add .mdlrc	last year
CODE_OF_CONDUCT.md	solve markdownlint issues	6 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 user chroot runc

Readme

Apache-2.0 license

Code of conduct

Security policy

Activity

1.1k stars

34 watching

117 forks

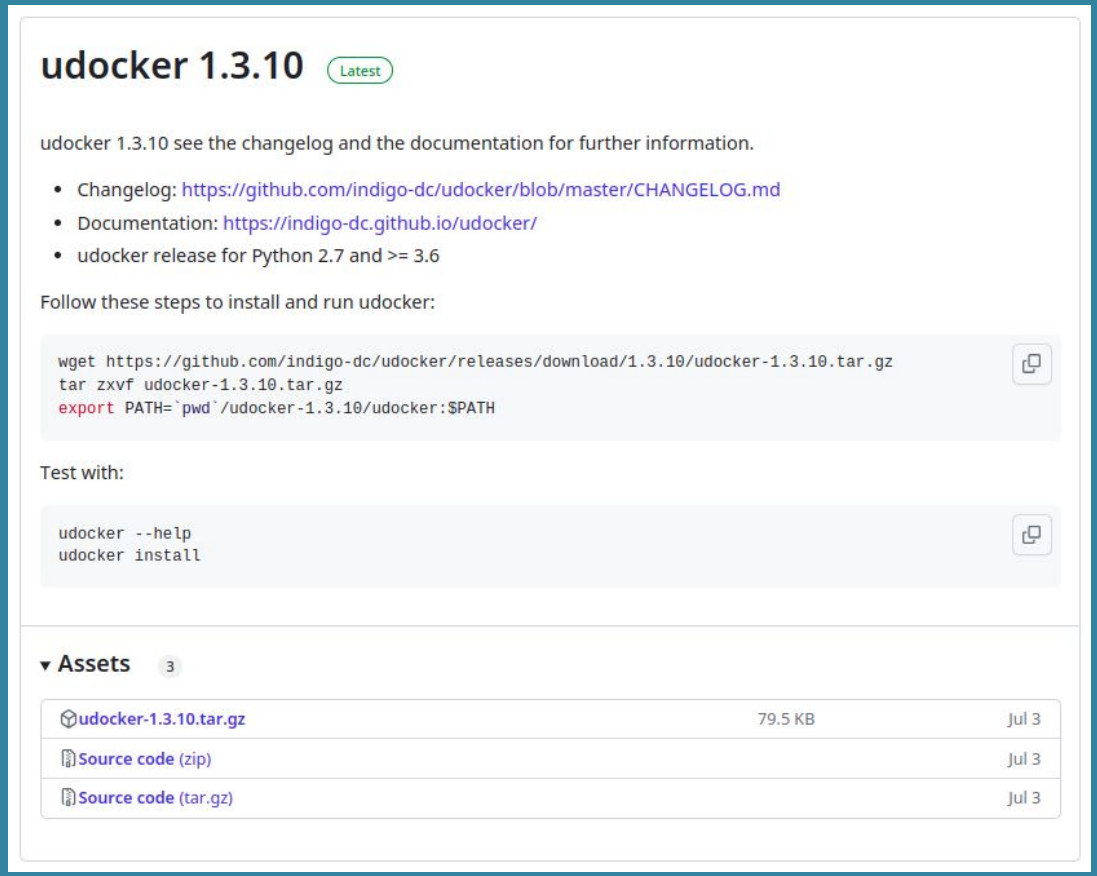
Report repository

Releases 21

udocker 1.3.10 Latest

on Jul 3

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



udocker 1.3.10 Latest

udocker 1.3.10 see the changelog and the documentation for further information.

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

Follow these steps to install and run udocker:

```
wget https://github.com/indigo-dc/udocker/releases/download/1.3.10/udocker-1.3.10.tar.gz
tar zxvf udocker-1.3.10.tar.gz
export PATH=`pwd`/udocker-1.3.10/udocker:$PATH
```

Test with:

```
udocker --help
udocker install
```

▾ Assets 3

udocker-1.3.10.tar.gz	79.5 KB	Jul 3
Source code (zip)		Jul 3
Source code (tar.gz)		Jul 3

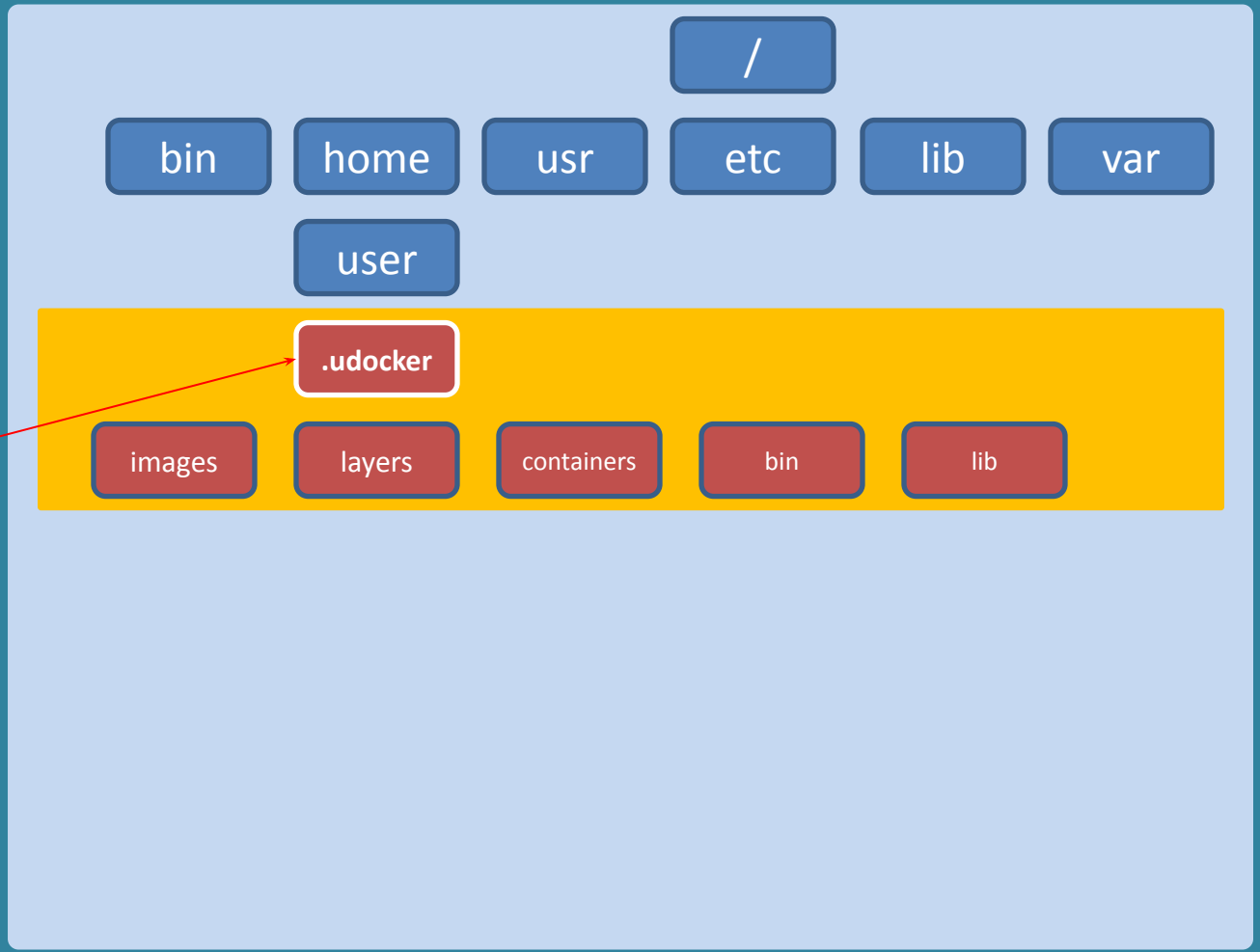
Install from a release

```
$   curl  -L  \
    https://github.com/indigo-dc/udocker/releases/download/1.3.10/udocker-1.3.10.tar.gz \
      >  udocker-1.3.10.tar.gz

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

$   tar  zxvf  udocker-1.3.10.tar.gz

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

$   export  PATH=`pwd`/udocker-1.3.10/udocker:$PATH

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

$   udocker  install
```

https://github.com/indigo-dc/udocker/blob/master/docs/installation_manual.md

What is created by udocker install

udocker directory tree default is $HOME/.udocker



Install from PyPI

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

$   ##  install udocker from PyPI
$   pip  install  udocker
    ##  install the binaries required to execute containers
$   udocker   install
$   udocker   version
```

Install from the source

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

## optional only to get the development code
$ git checkout devel3

$ cd udocker/udocker

## 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
```

Install without outbound connectivity

```
$   wget \
      https://github.com/indigo-dc/udocker/releases/download/v1.3.10/udocker-1.3.10.tar.gz

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

$   wget \
      https://github.com/jorge-lip/udocker-builds/raw/master/tarballs/udocker-englib-1.2.10.tar.gz

    ##  TRANSFER BOTH TARBALLS TO THE REMOTE SYSTEM and once transferred do:

$   tar  zxvf  udocker-1.3.10.tar.gz
$   export  PATH=`pwd`/udocker:$PATH

    ##  then install the binaries FROM THE TARBALL

$   export  UDOCKER_TARBALL="udocker-englib-1.2.10.tar.gz"
$   udocker  install
```

Execute in 3 steps

```
## pull an image from dockerhub

$    udocker  pull   centos:centos8

## extract the image content and give it a name so that we can execute it

$    udocker  create  --name=C8  centos:centos8

## execute many times as you want using the previously created container

$    udocker  run  C8
$    udocker  run  C8   /bin/bash
```

https://github.com/indigo-dc/udocker/blob/master/docs/user_manual.md

Understanding pull

- **By default udocker pulls from dockerhub**
  - **udocker pull**
    - **Downloads image layers into $HOME/.udocker/layers**
    - **Layers that have been previously downloaded are not fetched**
    - **Image layers are tar files**

- **Consequences**
  - **Other repositories must be specifically identified:**

```
udocker pull  quay.io/centos/centos:centos8
```

  - **Images layers are not directly used in execution only needed in create step**
    - **Image layers => udocker create => Containers**
  - **Removing an image does not affect the created containers**

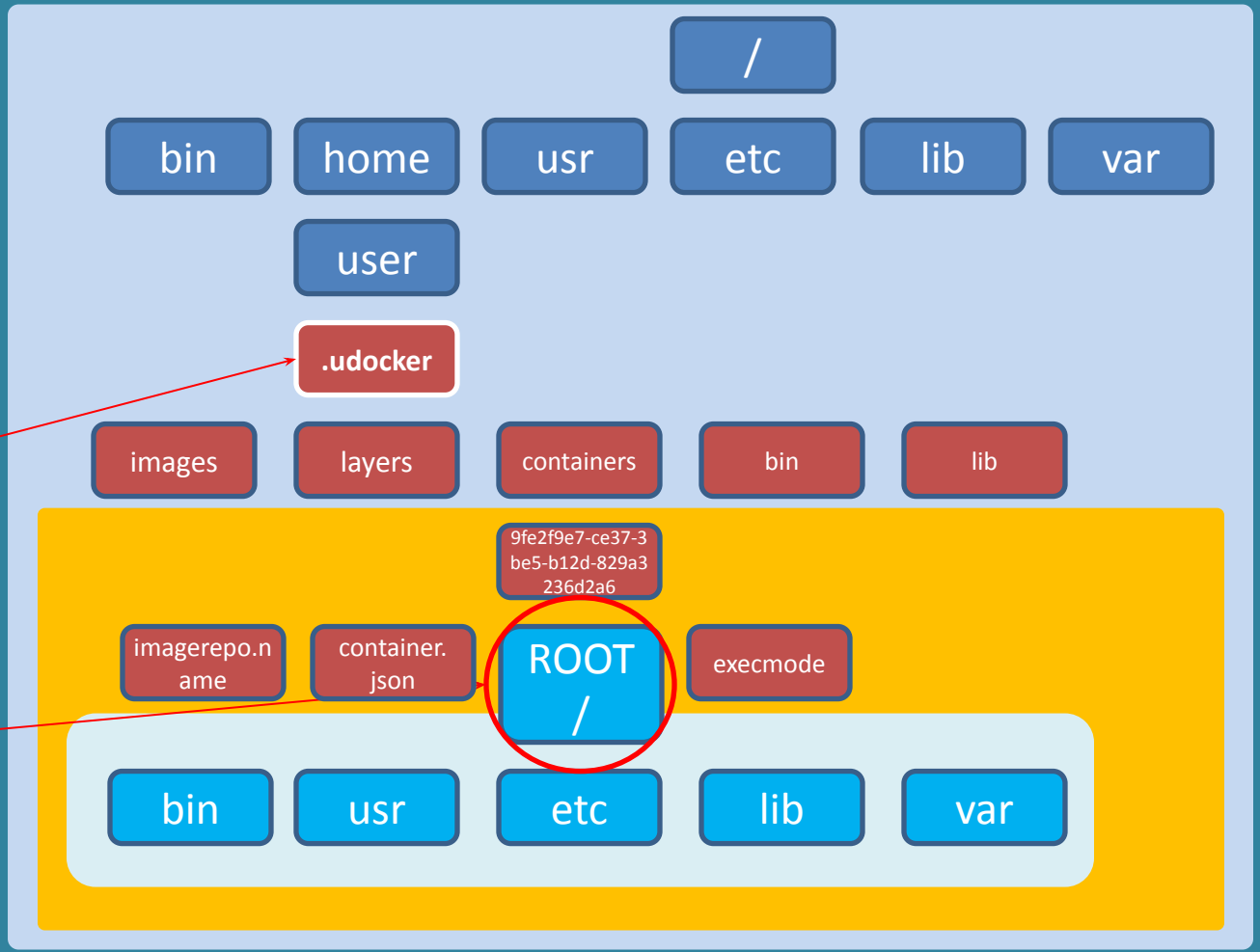
Understanding create

- udocker works very differently from docker and other engines
  - udocker create
    - Performs an untar of the image layers into a certain directory
- Consequences
  - Accessing the container files in udocker is very easy, they are in:
    $HOME/.udocker/containers/<container-id or name>/ROOT
  - The create/untar operation can be slow or fast
    - Depends on the underlying file-system performance
    - If possible create once and run many
  - Lustre and GPFS are distributed file-systems
    - Can be very slow on metadata and small file operations
  - If the content of a container is not changed or shared during execution
    - Might be better to use a disk local to the execution system
    - Might pay off even if the container needs to be created for every run

What is created by udocker create

udocker directory tree default is $HOME/.udocker

container tree in udocker



Understanding create and run

- udocker execution
  - udocker run imageName
    - Always creates/untars a new container (can be slow to start executing)

```
udocker run centos:centos8
```

  - udocker run containerID
    - Uses an already created/untared container using the alphanumeric ID returned by create

```
udocker create centos:centos8
udocker run 654fccb8-1b63-3e1c-8147-b811181503b5
```

  - udocker run containerName
    - Uses an already created/untared container using the alias name previously given upon creation

```
udocker create --name=containerName centos:centos8
udocker run containerName
```

Using non-default directories

- **Install the udocker Python code as described previously in slide 7**

- **Place udocker tools binaries, images/layers and containers in a different directory**

```
export UDOCKER_DIR=/tmp/${USER}_udocker
mkdir $UDOCKER_DIR
udocker install
```

- **Alternatively place created containers in a different directory and keep everything else in the default places**

```
udocker install
export UDOCKER_CONTAINERS=/tmp/${USER}_containers
mkdir $UDOCKER_CONTAINERS
```

Execution engines

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 FASTER
F4	Fakechroot	modified loader and ELF headers dynamically changed
S1	Singularity	where locally installed using chroot or user namespaces

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

Summary of recent improvements

- **Improved MPI support**
  - **Fixed issues with PMI interface to the batch system (1.3.5)**

- **Handling of different platforms**
  - **Support for multiple architectures within udocker internals (1.3.9)**
  - **Support for multiplatform image manifests version 2 schema 2 (1.3.9)**
  - **Pull or load images for architectures different from the host system (1.3.9)**
  - **Emulation using QEMU user space in Pn modes (1.3.10)**

- **Execution in different platforms (1.3.10)**
  - **Pn: x86_64, x86, arm, arm64, armel, armhf**
  - **Rn: x86_64, arm64, ppc64le**
  - **Fn: x86_64, arm64, ppc64le**

Multi-platform

- **Support to handle images and containers having different architectures:**
  - **os/architecture/variant**
    - **eg: linux/arm64/v8**
  - **Support for image manifests version 2 schema 2**
    - **dockerhub API: pull**
    - **load, import**
    - **manifest inspect**
  - **Side effect also improved supported for OCI images**

Multi-platform

```
$ udocker manifest inspect centos:centos8
```


```
"manifests": [
  {
    "digest": "sha256:a1801b843b1bfaf77c501e7a6d3f709401a1e0c83863037fa3aab063a7fdb9dc",
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "platform": {
      "architecture": "amd64",
      "os": "linux"
    },
    "size": 529
  },
  {
    "digest": "sha256:65a4aad1156d8a0679537cb78519a17eb7142e05a968b26a5361153006224fdc",
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "platform": {
      "architecture": "arm64",
      "os": "linux",
      "variant": "v8"
    },
    "size": 529
  },
  {
    "digest": "sha256:8bcb67d4c816e16f79bdea8079c5e30e88f404f4aa8fa124245864830d7852d2",
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "platform": {
      "architecture": "ppc64le",
      "os": "linux"
    },
    "size": 529
  }
],
"mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
"schemaVersion": 2
```

Multi-platform

```
$ udocker pull --platform=linux/arm64 centos:centos8

$ udocker images -p

REPOSITORY
linux/amd64          . almalinux/almalinux:latest
linux/amd64          . centos:centos7
linux/arm64/v8       . centos:centos8

$ udocker create --name=C8arm64 centos:centos8

$ udocker ps -p

CONTAINER ID                          P M PLATFORM          NAMES       IMAGE
4d3f7900-98d3-334d-b800-06fb4088750d  . W linux/amd64       ['A9']      almalinux/almalinux:latest
9d5a24f1-99da-363f-ad45-9802082917f7  . W linux/amd64       ['C8']      centos:centos8
d27c0e54-55ae-34e5-8002-6060196b9d10  . W linux/amd64       ['C7']      centos:centos7
cea0d849-c86a-3219-b544-75195a29ec9b  . W linux/arm64/v8    ['c8arm64'] centos:centos8
```

Multi-platform

```
$ udocker save -o myimage centos:centos8
$ udocker load -i myimage myc
$ udocker images -p
```

metadata includes platform information

```
REPOSITORY
linux/arm64/v8    . myc:centos8
linux/amd64       . almalinux/almalinux:latest
linux/amd64       . centos:centos7
linux/arm64/v8    . centos:centos8
```

with import need to specify the platform information otherwise defaults to host

```
$ udocker import --platform=linux/arm64/v8 tarfile imagename
```

Multi-platform

Host with x86_64

```
$ uname -a
Linux pcjorge.lip.pt 6.4.11-200.fc38.x86_64 #1 SMP PREEMPT_DYNAMIC Wed Aug 16 17:42:12 UTC 2023 x86_64 GNU/Linux
```

Emulating arm64

```
$ udocker run --nobanner centos:centos8 /bin/uname -a
Linux nbjorge 6.4.11-200.fc38.x86_64 #1 SMP PREEMPT_DYNAMIC Wed Aug 16 17:42:12 UTC 2023 aarch64 aarch64 aarch64 GNU/Linux
```

- Image needs to be pulled with the intended platform
- Enables execution of binary code different from the host architecture
- Execution is performed with QEMU User Space Emulation
- This is currently an EXPERIMENTAL feature

Multi-platform

- Pull, create and run for a different platform architecture only works when the image does not yet exist locally
- If image already exists with a given architecture will take precedence

Multi-platform

- **Execution with user space emulation**
  - Requires QEMU User Space Emulator (not the QEMU system full emulation)
  - QEMU user enables launch Linux processes compiled for one CPU on another CPU, translating syscalls on the fly.
  - We are looking at shipping QEMU user within the udocker tools package
- **In the default Pn execution modes**
  - Integrated with the system call interception in proot
  - proot itself will invoke the QEMU user space emulators
- **In the Fn execution modes**
  - QEMU user space emulators need to be set in binfmt_misc in the host kernel
  - Emulators are invoked upon exec

```
$ ls /proc/sys/fs/binfmt_misc
```

qemu-aarch64	qemu-arm	qemu-armeb	qemu-mips	qemu-mips64	qemu-ppc
qemu-ppc64	qemu-ppc64le	qemu-riscv32	qemu-riscv64	qemu-s390x	windows

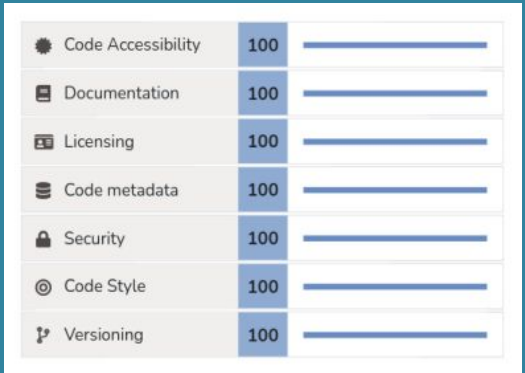
Multi-platform native execution

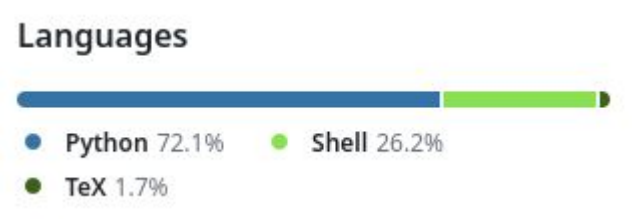
- **Meaning**
  - **Execution in multiple host architectures beyond x86_64 (EXPERIMENTAL)**
  - **Physical hosts and no emulation**
  - **Requires execution engines compiled for the target host**
- **Pn modes**
  - **proot for multiple architectures**
  - **arm, arm64, armhf, armel, x86, x86_64**
- **Rn modes**
  - **crun and runc for multiple architectures**
  - **x86_64, arm64, ppc64le**
- **Fn modes**
  - **patchelf, libfakechroot for multiple architectures**
  - **arm64, ppc64le, x86_64**

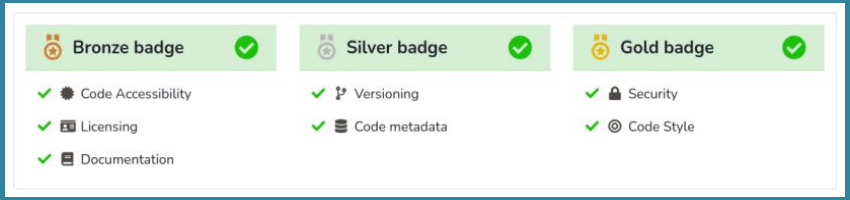
SQA

pypi package 1.3.10 | build passing

- **Adopted the EOSC-Synergy quality criteria**
  - **Using the SQAaaS for all releases**
  - **Push to master only when passing SQA**
  - **Maintaining the Synergy-software-gold**
  - **As code grows new unit tests are required => major effort**
  - **Moving from unittest to pytests**







Upcoming

- **Working on udocker 1.4.x**
  - **New improved installation and management of udockertools components**
    - **Binary executables**
    - **Binary libraries**
  - **The number of components is growing**
    - **Binaries and libraries for different platforms**
    - **Support for new distributions particularly in Fn modes**
    - **Will enable the packaging of QEMU user emulation**
  - **Hardened installation**
  - **Changes in internal classes and code organisation**
  - **Interface changes**
  - **New engines**

Thank you !

Questions ?

udocker@lip.pt

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

Advantages

Installation:

- Is deployable directly by the end-user
- Does not require privileges for installation
- Self contained does not require installation of additional software
- Does not require compilation just deploy and use
- Does not require system administrators intervention to setup
- Can be installed from source, release tarball or from PyPI

Execution:

- As a normal end-user
- Runs entirely in user space
- Execution regardless of OS functionalities
- Respecting normal process controls and accounting
- Supports multiple execution methods
- In Linux interactive or batch systems

Implementation

- Front-end
  - Provides the a command line interface similar to docker and other tools
  - Provides handling of container images (pull, import ,export, load save)
  - Manages a local image and extracted containers repository
  - Provides the interface with the several execution engines
  - Written in Python supports Python 2.6, 2.7 and Python >= 3.5

- Backend
  - Includes external binary tools modified and packaged by the udocker team
  - Both executables and libraries implementing the several engines
  - Compiled statically to enable execution across Linux systems

udocker commands

```
$   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
- Provides a subset of docker commands
- Actual container creation is better performed using docker itself

Pull and run

```
$ udocker pull quay.io:centos/centos:7

Info: downloading layer sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc
Info: downloading layer sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Info: downloading layer sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4

$ udocker create --name=C7 quay.io:centos/centos:7

7464ceb6b-e9c6-3eb0-9646-6040092b4367

$ udocker -q run C7 /bin/cat /etc/redhat-release

CentOS Linux release 7.9.2009 (Core)

$ udocker -q run --user=$USER --bindhome --hostauth C7 /bin/bash

464ceb6b$
```

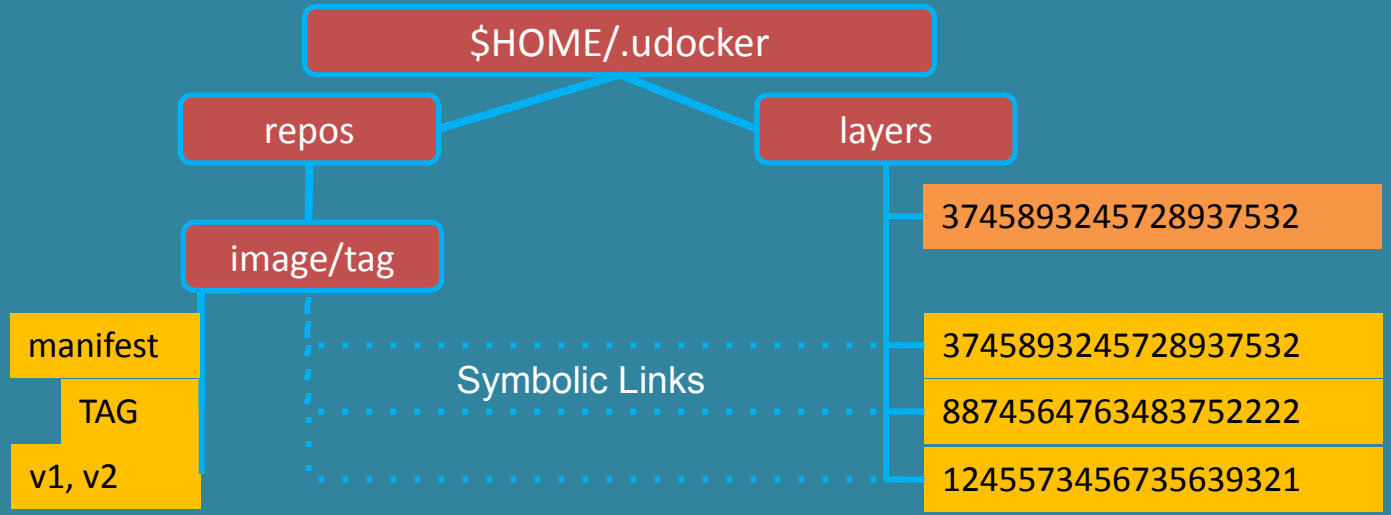
For the impatient

```
$ udocker -q run --user=$USER --bindhome --hostauth \
      quay.io/centos/centos:7 /bin/bash
```

```
77af3c48$ pwd
/home/jorge
```

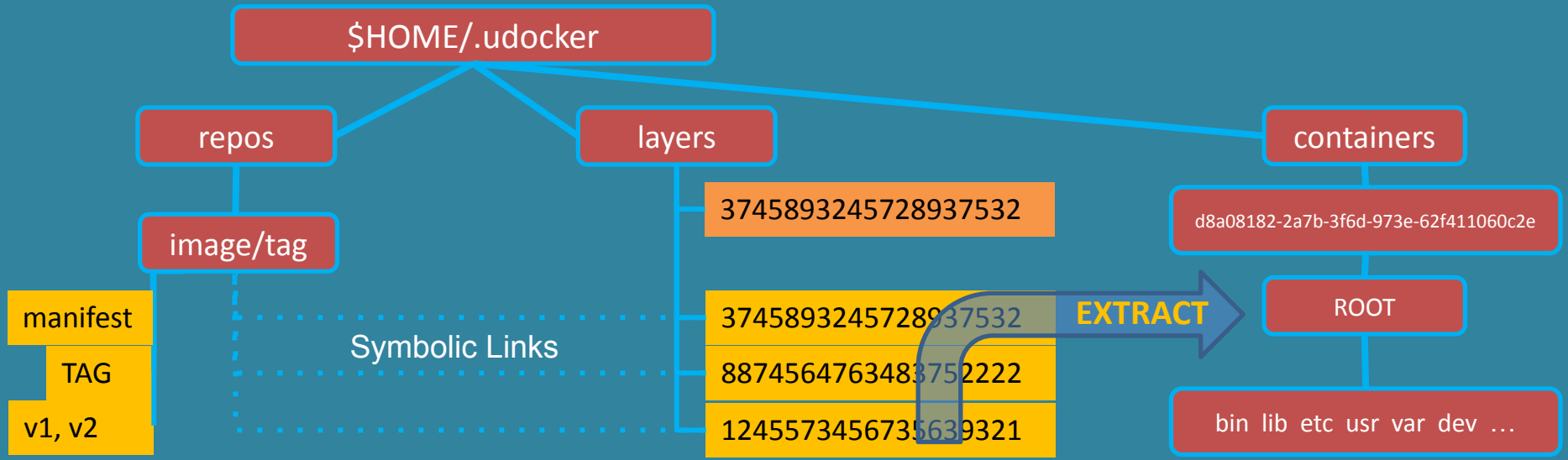
udocker pull

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



udocker create

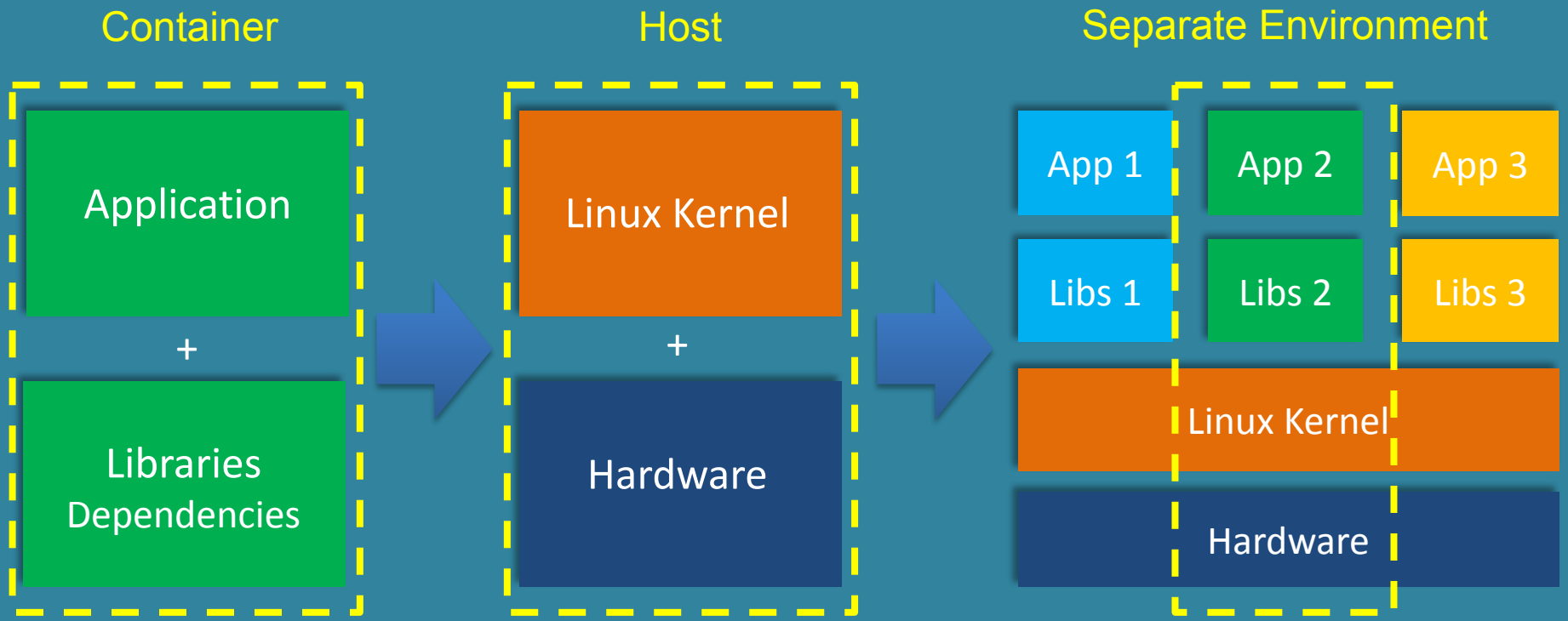
- Containers
  - Are produced from the layers by flattening them sequentially
  - Each layer is extracted on top of the previous
  - The OnionFS whiteouts are respected, and file protections are changed as needed
  - The obtained directory trees are stored under $HOME/.udocker/containers



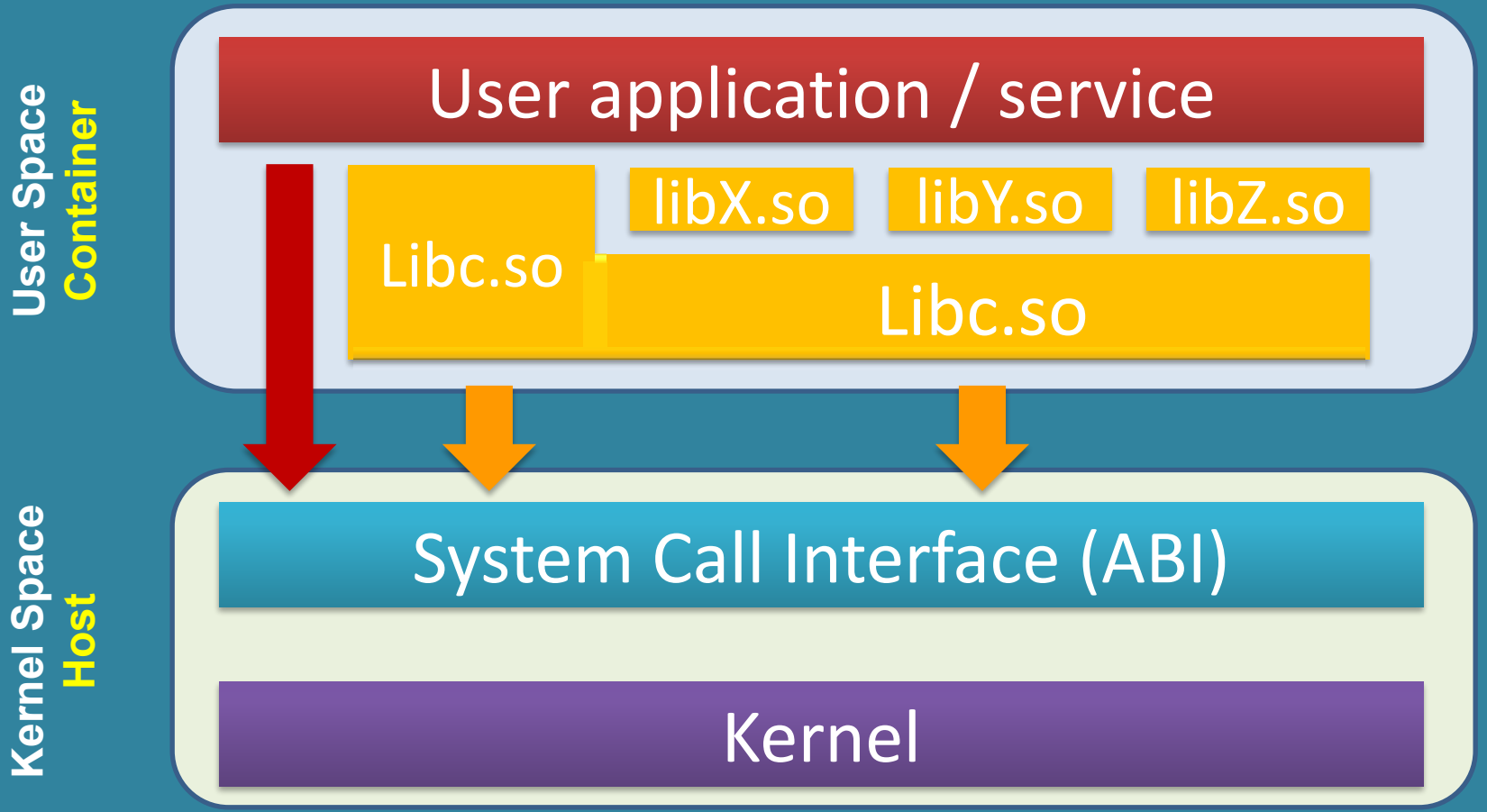
Advantages of using containers for applications

- 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

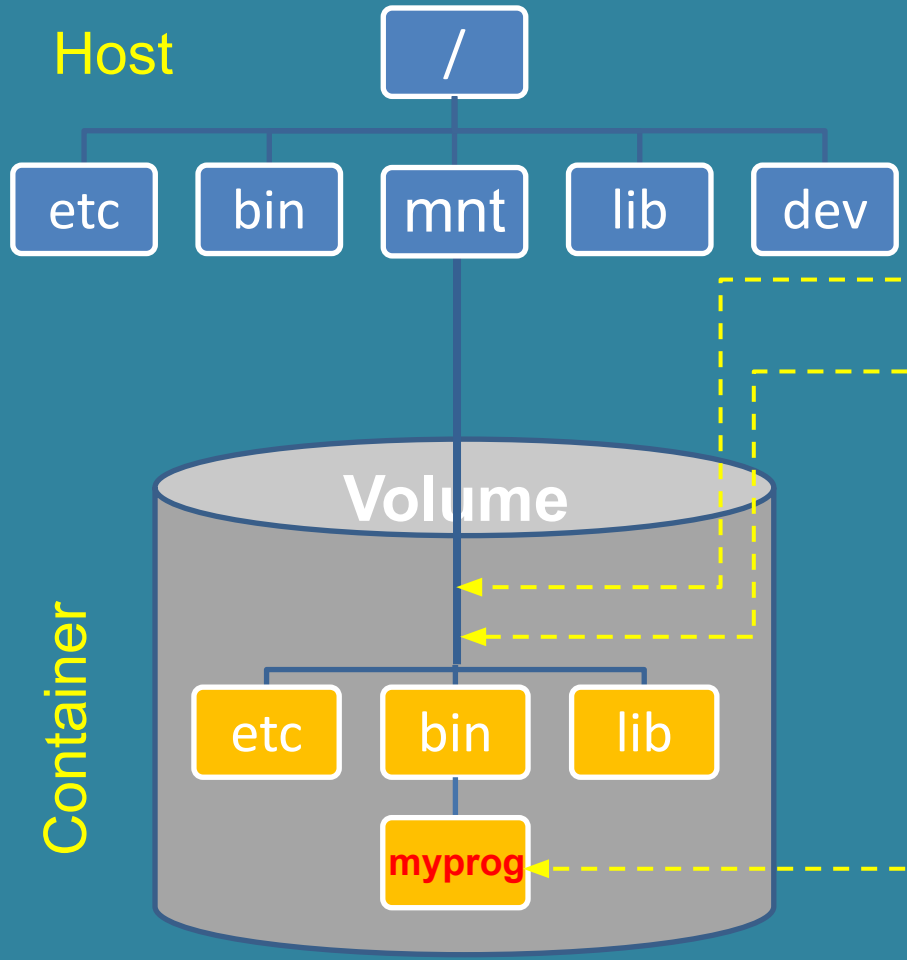
Containers


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



Process

```
mount( “VOL” , “/mnt” ,…)
chdir( “/mnt” )
pivot_root( “.”, “.”)
chroot( “.” )
execl( “/bin/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

udocker commands

```
$   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		

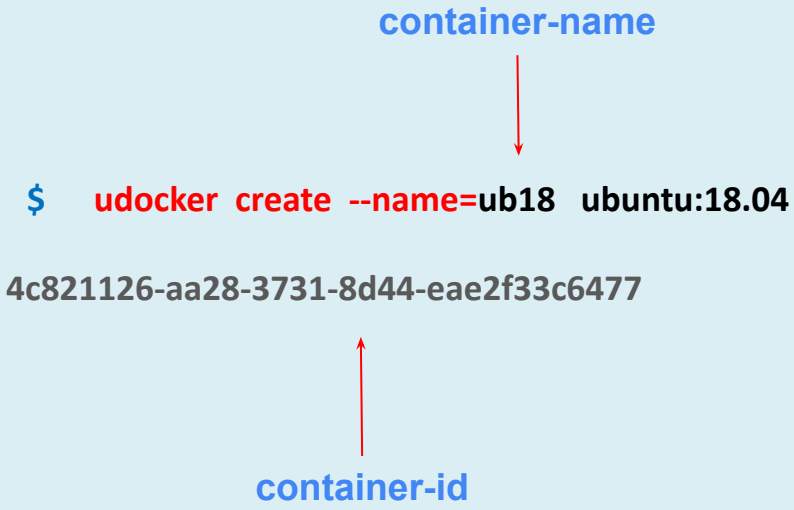
- udocker is mainly a run-time to execute containers
- Provides a subset of docker commands
- Actual container creation is better performed using docker itself

Pull images from dockerhub

```
$ udocker pull ubuntu:18.04

Info: downloading layer sha256:726b8a513d66e3585eb57389171d97fcd348e4914a415891e1da135b85ffa6c3
Info: downloading layer sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4

$ udocker pull quay.io:centos/centos:7

Info: downloading layer sha256:2d473b07cdd5f0912cd6f1a703352c82b512407db6b05b43f2553732b55df3bc
Info: downloading layer sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Info: downloading layer sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
```

- By default udocker pulls images from dockerhub

Create runnable container from an image



- Create will extract the content of an image into the user home directory
- By default created containers are stored under $HOME/.udocker/containers
- Created containers can be referenced either by id or name

List created containers

```
$ udocker ps
```

```
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
```

- List created containers stored under $HOME/.udocker/containers
- Includes: id, protection, mode, names and related image
- Although the command is named **ps** there are no associated processes

Execute a container

```
$   udocker  run  ub18
or
$   udocker  run  4c821126-aa28-3731-8d44-eae2f33c6477

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

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

- List created containers stored under $HOME/.udocker/containers
- Includes: id, protection, mode, names and related image
- Although the command is named **ps** there are no associated processes

Execute a container

```
$   udocker  run  ub18
```

ubuntu

```
****************************************************************************
*                                                                          *
*                  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)
```

root emulation

Run as yourself

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

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

- --user identifies a username
- -v binds a directory to be visible inside of the running container
- -e allows setting environment variables

Run as yourself

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

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

- --user identifies a username
- --bindhome binds the user home directory to be visible in the container
- --hostauth uses the host passwd and group in the container

Less verbosity

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

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
```

- -q quiet mode

Overriding the entrypoint

```
$ 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"

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

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

- --entrypoint will override the entrypoint defined in the container metadata
- --entrypoint is valid from v1.3.0 and is the preferred method
- A container defined entrypoint can be ignored with --entrypoint=""

Run commands inline

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

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

- Useful to execute commands inside a container in scripts

Duplicate a container

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

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

new cloned container-id

new cloned name

```
$   udocker  run  new18
or
$   udocker  run  9fe2f9e7-ce37-3be5-b12d-829a3236d2a6
```

- The command clone duplicates an existing container

Import and export tarballs with images

export to tarball

input container

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

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

import ub18.tar tarball and create image with this name and tag

```
$  udocker  export  -o  -  ub18 | docker import  -  myub18:latest

$  docker  export  bd221eb5e452 | udocker import  -  anotherub18:latest
```

- Export produces a tar file from a created container with the container filesystem
- Export does not include the container metadata
- Import takes a tar file and loads its content as an image
- Import and export are interoperable with docker

Saving space

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

- --tocontainer allows importing a tar file directly to a container (NOT TO IMAGE !!)
- No image is created
- The newly created container can be named with --name
- This will save both space and the intermediate step of creating an image

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

- Import a tar file normally to a new image
- To save space the tar file is moved to the udocker repository instead of copied
- The original tar file disappears

Import and export including udocker specific metadata

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

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

- Export a container to a tar file including the udocker specific metadata
- Both container data and metadata are included
- Import a tar file produced by *export --clone* as a new CONTAINER
- The newly created container can be named with --name
- NOT interoperable with docker as includes udocker specific information

Transfer containers across machines

export the tarball to stdout

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

read the tarball from stdin

- Export a container and import it in another remote host or account using SSH
- Allows export and import to be accomplished in one go
- Not interoperable with docker
- Notice the "" around the remote command when using SSH
- Containers with execution modes such as F2, F3 and F4 modes may not work, breaks if the pathname to the user home (container dir) is not the same

Save and load images

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


$  udocker  load  -i  image.tar
```

- Load an image saved by docker or udocker
- A saved image will contain multiple layers and metadata (different for export !!)
- A saved image cannot be imported only loaded
- Can be used to load docker images instead of pulling them from dockerhub
- udocker also provides a compatible save functionality

Remove containers and images

- Deleting images does not affect the created containers

Where is my container

container alias or container-id

```
$   udocker  inspect  -p  ub18

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


$   ls  $(udocker  inspect  -p  ub18)
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var boot  etc  lib  media  opt  root  sbin  sys  usr
```

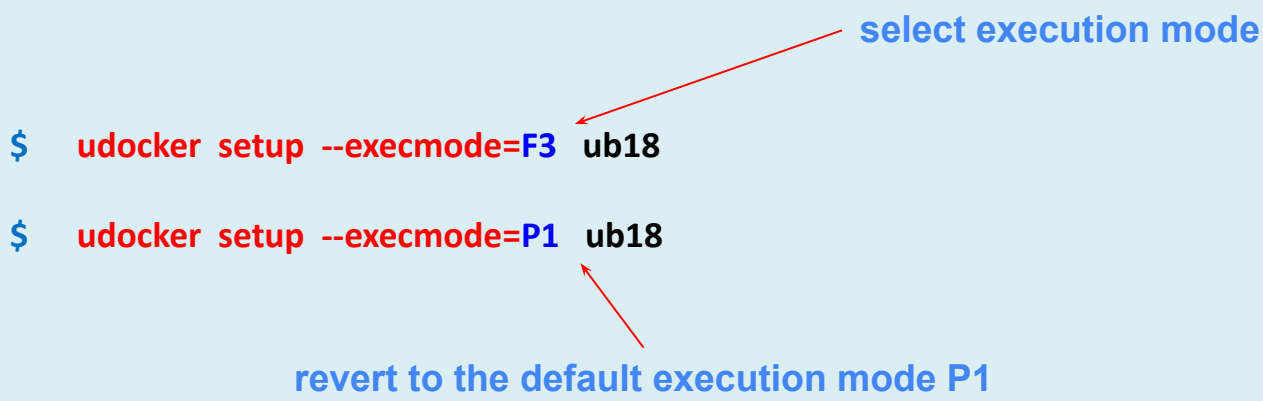
- Inspect -p prints the pathname to the container ROOT directory
- Knowing the pathname is useful to copy/manage/edit the container files directly

Execution engines

- udocker integrates several execution methods:
  - Supports several engines to execute containers
  - They are selected per container 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



- Execution modes are selected using setup
- Each mode has two characters
  - the first is a letter that identifies the engine
  - the second is a number that identifies different options within an engine
- The default execution mode is P1

List containers and execution engines

```
$  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
```

- udocker -m will list all containers and their execution modes

Engine Pn : PRoot

- PRoot uses PTRACE to intercept system calls
- Pathnames are translated before being passed to the call
  - To expand container pathnames into host pathnames
- After the call returned pathnames are translated back
  - 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 system calls slower

- The impact of tracing depends on the system call frequency

Engine Fn : Fakechroot

- Uses LD_PRELOAD to intercept shared library calls
- Pathnames are translated before being passed to the call
  - To expand container pathnames into host pathnames
- After the call returned pathnames are translated back
  - To shrink host pathnames to container pathnames

- Generally higher performance than Pn and even than 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 : all modes

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

Containers using kernel features

- **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

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 as only has a loopback device
- 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

- 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 mapping of devices

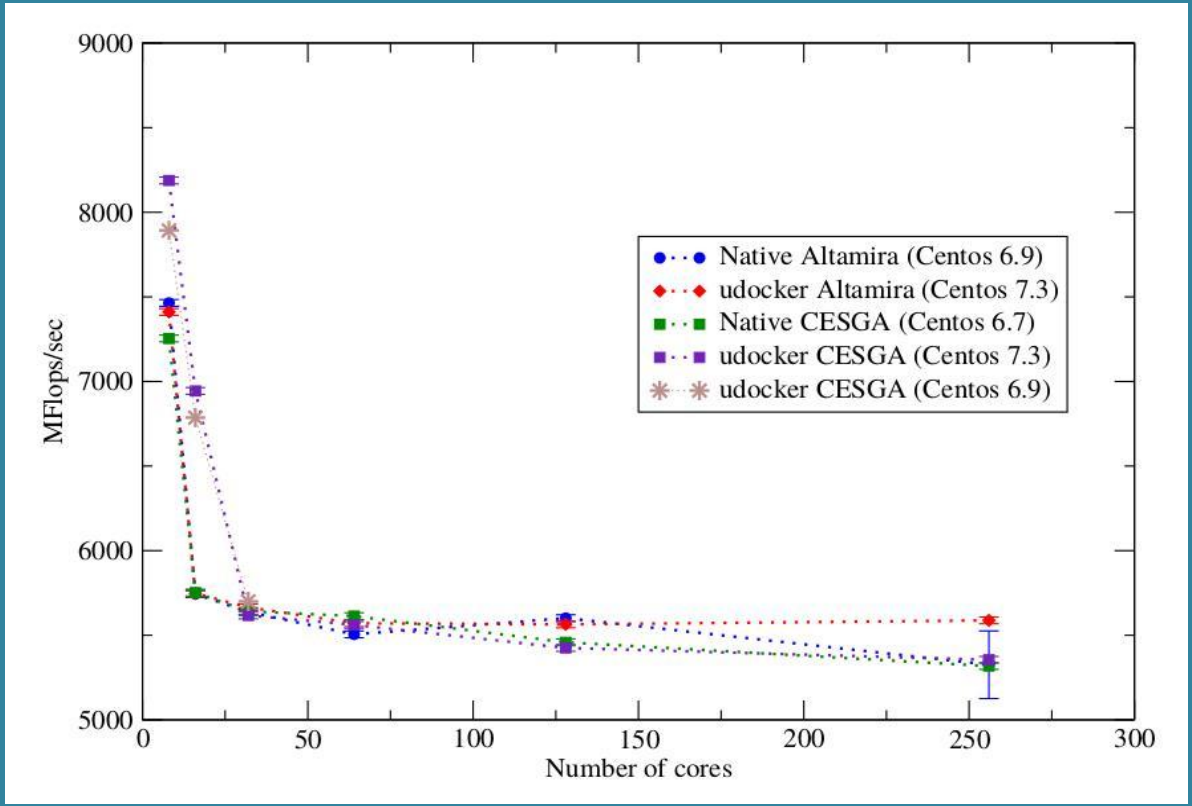
- We added mapping of Docker metadata to OCI
- udocker can produce an OCI spec and run containers using runc or crun transparently

NVIDIA GPUs

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

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

Lattice QCD



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

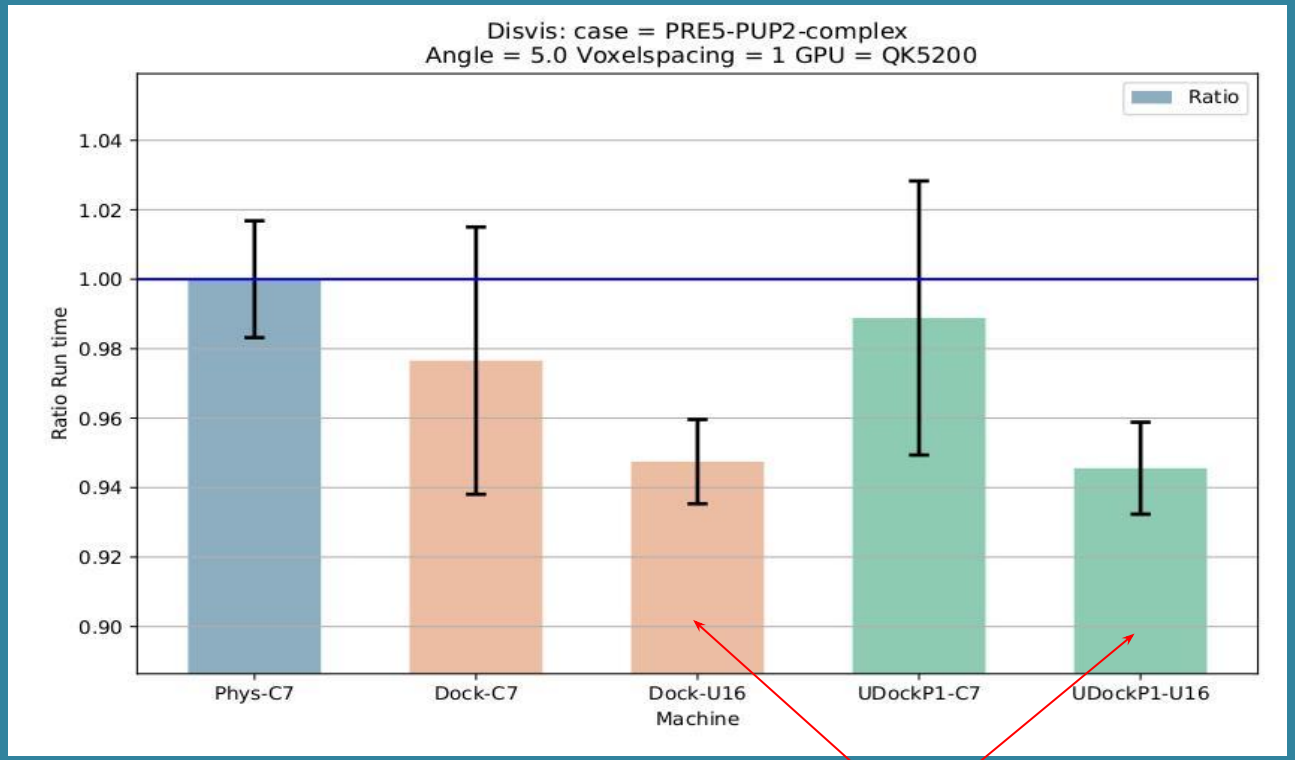
Lattice QCD with OpenMPI

```
$   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)

- mpiexec starts udocker which in turn runs the executable within the container
- LD_LIBRARY_PATH is redefine to point to the container /usr/lib
- The environment from the host has the OPENMPI variables that are essential
- Must run as the same user as in the host hence --hostauth and --user
- The workdir is within the container

Biomolecular Complexes



DisVis is being used in production with udocker

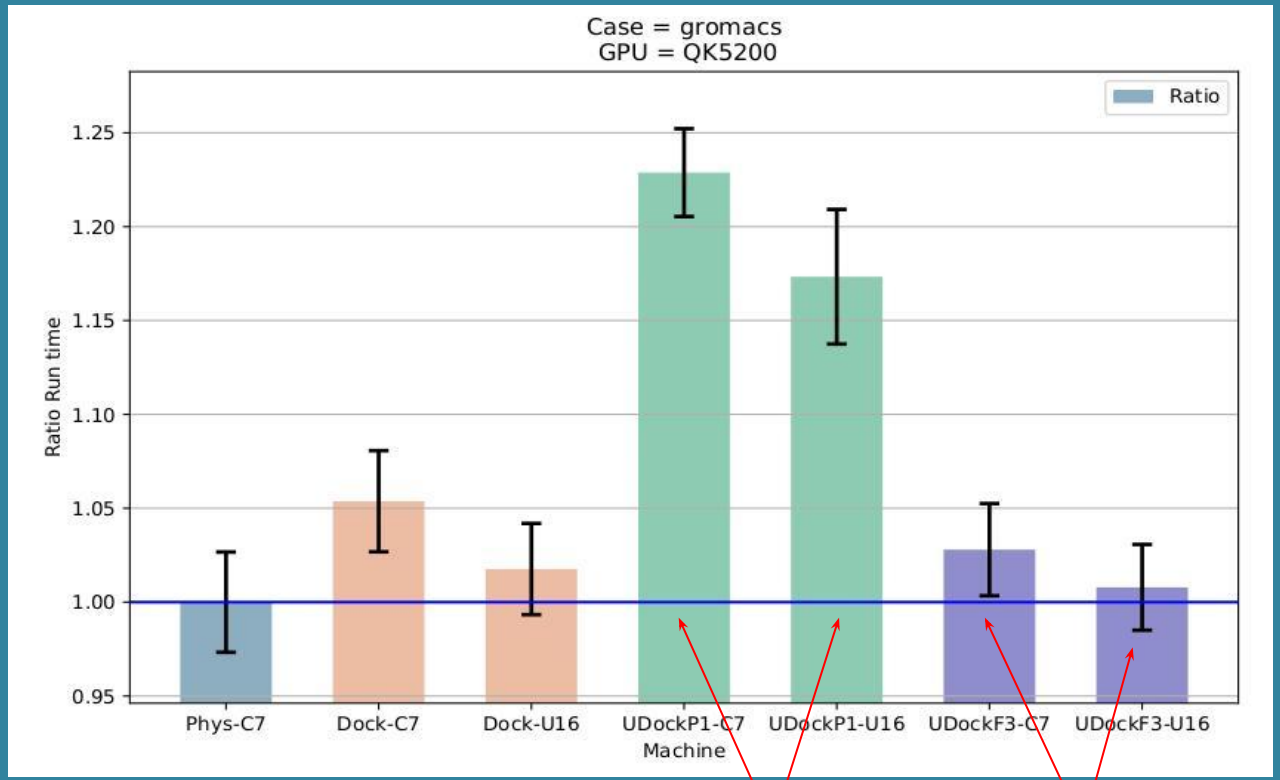
Performance with docker and udocker are the same and very similar to the host.

Using OpenCL and NVIDIA GPGPUs

udocker in P1 mode

Better performance with Ubuntu 16 container

Moleculas 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

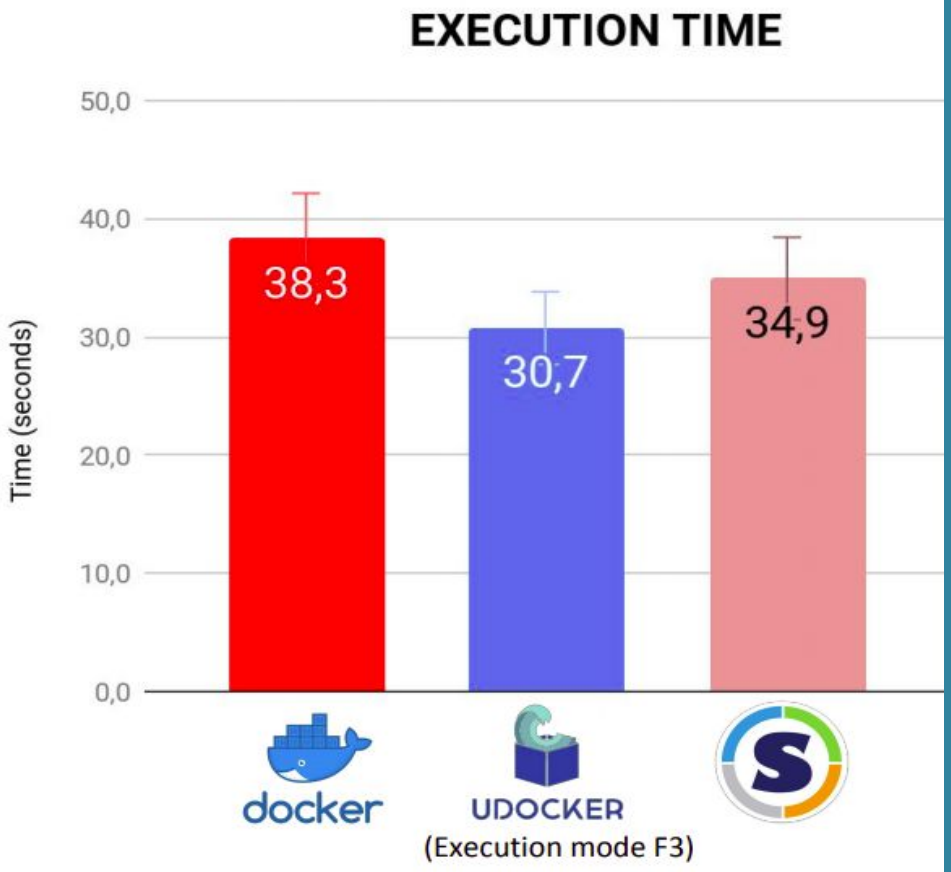
udocker in P1 mode
udocker in F3 mode

TensorFlow

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

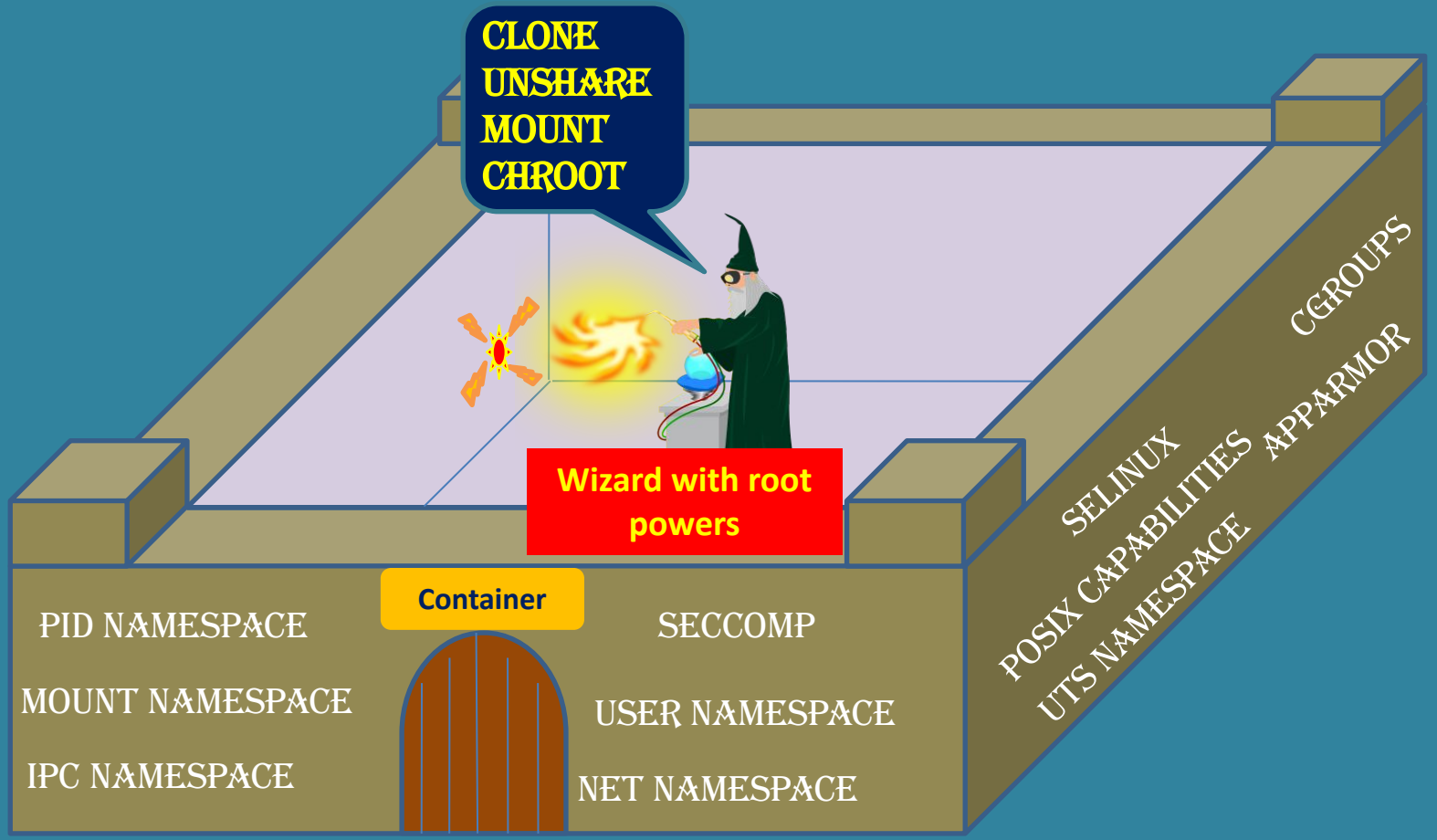
Selection in terms of performance

Mode	Base	Description
P1	PRoot	Some 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	Requires shared library compiled against same libc as in the container. May load host libraries.
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



Search 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
```

- Searches for repositories in dockerhub

List tags in a repository

```
$ 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 dockerhub and may not work with other repositories

Pull from other registries

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

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

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

- The first element in the image name is the registry name in this case quay.io
- The second element is the repository also called library
- The third element is the image name and tag separated by a semicolon