cesnet 25

Support of container environment

Jan Hoidekr

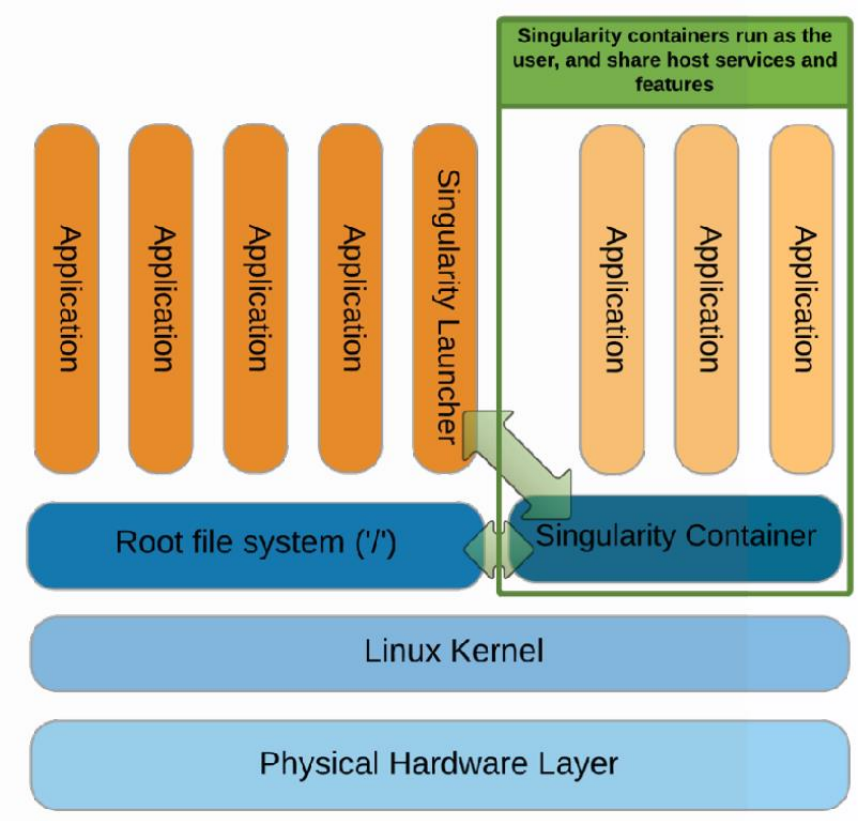
Support of container environment

- Environment, containers, images etc.
- Container tools and their differences
- Running containers in Metacentrum
- Create your own images
- Support of GPU, MPI, env vars, …

Introduction to containers

- Container = OS-level virtualization

- Docker, Podman, Singularity, …
- Environment – set of apps and libs
- Image – prepared environment
- Container – running image
- Registry – repository of images
  - docker:// shub://
  - Filesystem, CVMFS



Container tools

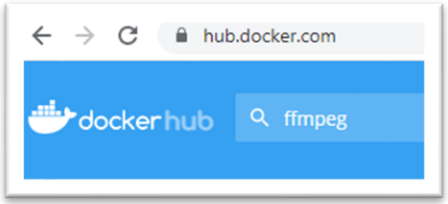
- Docker – widely used, not suitable for HPC
  - Daemon + CLI, root privileges, simple to use
  - Docker Hub - hub.docker.com
  - Not suitable for HPC due to **security**
- Podman – the same UI as docker
  - No daemon, runs in userspace
  - docker images without rebuild
- Singularity
  - aimed into HPC world
    - “Integration is more important then isolation”
  - Singularity Image Format – needed rebuild of docker images

Singularity in Metacentrum

- Motivation to use
  - Custom and defined environment – for user/group/public
  - Easy to share environment
  - Repeatability of task
- Support of MPI, Infiniband, GPU – CUDA and ROCm
- How to use?
  - Use images from docker hub or shared images
  - Create/modify own image
    - on your PC with root (or subuid/subgid) to build the image
    - in Metacentrum only - **builder.metacentrum.cz**
      - members of group `builders`, ask RT meta@cesnet.cz to access

ffmpeg example

- Tool **ffmpeg** – multimedia framework
  - module in Metacentrum – ffmpeg, version **2.4**, from 2017
  - hub.docker.com - most popular jrottenberg/ffmpeg version **2.8** – **4.2**



  - Debian Buster version **4.1.6**
  - Current version **4.4**

Note: work with PBS, minimalise load of frontends load

Podman example

- **Podman** – only in CERIT-SC = PBS queues `*@cerit-pbs.cerit-sc.cz`
  - run user's docker image in Metacentrum – the same CLI

- Limitations
  - only root in container can access NFS = /storage
  - `podmanwrapper.sh`
    see wiki to run PBS jobs

```
hoidekr@glados5:~$ podman run jrottenberg/ffmpeg:4.2-ubuntu
Resolved "jrottenberg/ffmpeg" as an alias (/mnt/storage-brno3-
cerit/nfs4/home/hoidekr/.config/.cache/containers/short-name-aliases.conf)
Trying to pull docker.io/jrottenberg/ffmpeg:4.2-ubuntu...
Getting image source signatures
Copying blob c4394a92d1f8 done
Copying blob 1e1aebc5422b done
Copying blob a70d879fa598 done
Copying blob 3a7ec0cd47ea done
Copying blob 10e6159c56c0 done
Copying blob 68dc71384289 done
Copying config 12768ef4ff done
Writing manifest to image destination
Storing signatures
ffmpeg version 4.2.4 Copyright (c) 2000-2020 the FFmpeg developers
```

Singularity example

Singularity

```
$ singularity run docker://jrottenberg/ffmpeg:4.2-ubuntu
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
Getting image source signatures
Copying blob a70d879fa598 done
< … shortened … >
INFO:    Creating SIF file...
ffmpeg version 4.2.4 Copyright (c) 2000-2020 the FFmpeg developers
```

Second run is faster using cached SIF image

Singularity cache default ~/.singularity
or SINGULARITY_CACHEDIR

Delete cache: singularity cache clean

OR save SIF image, create alias, run ffmpeg

```
$ singularity build ffmpeg4.2.SIF docker://jrottenberg/ffmpeg:4.2-ubuntu
INFO:    Starting build...
INFO:    Creating SIF file...
INFO:    Build complete: ffmpeg4.2.SIF

$ singularity run ffmpeg4.2.SIF
ffmpeg version 4.2.4 Copyright (c) 2000-2020 the FFmpeg developers

$ alias ffmpeg4.2="singularity run ffmpeg4.2.SIF

$ ffmpeg4.2
ffmpeg version 4.2.4 Copyright (c) 2000-2020 the FFmpeg developers
```

ffmpeg4.2.SIF – image
- regular file
- ffmpeg and libs inside
  - isolated from meta env
- /storage/ and home accessible

Singularity building own image

- **builder.metacentrum.cz** – ask for group `builders`
  - use dir `/scratch/username`
- Preparing and testing image – via sandbox (image file is read-only)
  1. Create sandbox

     ```
     singularity build –f –s test1.sbox docker://debian:buster
     ```
  2. Set up environment inside

     ```
     singularity shell –f –w test1.sbox
     Singularity>     (work inside container and exit)
     ```
  3. Build image

     ```
     singularity build –f test1.SIF test1.sbox
     ```

Singularity building own image

```
$ singularity build -f -s test1.sbox docker://debian:buster
INFO:    Starting build...
Getting image source signatures
<…shortened…>
Storing signatures
2021/04/18 23:06:28  info unpack layer: sha256:bd8f6…..
INFO:    Creating sandbox directory...
INFO:    Build complete: test1.sbox

$ singularity shell -f -w test1.sbox/
Singularity> apt update
<…shortened…>
Singularity> apt install ffmpeg -y
<…shortened…>
Singularity> exit

$ singularity build -f  test1.SIF test1.sbox/
INFO:    Starting build...
INFO:    Creating SIF file...
INFO:    Build complete: test1.SIF

$ singularity run test1.SIF ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
<…shortened…>

$ alias ffmpeg="singularity run /scratch/hoidekr/test1.SIF ffmpeg"

$ ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
```

Start from debian:buster

test1.sbox – directory

Singularity building own image

```
$ singularity build -f -s test1.sbox docker://debian:buster
INFO:    Starting build...
Getting image source signatures
<…shortened…>
Storing signatures
2021/04/18 23:06:28  info unpack layer: sha256:bd8f6…..
INFO:    Creating sandbox directory...
INFO:    Build complete: test1.sbox

$ singularity shell -f -w test1.sbox/
Singularity> apt update
<…shortened…>
Singularity> apt install ffmpeg -y
<…shortened…>
Singularity> exit

$ singularity build -f  test1.SIF test1.sbox/
INFO:    Starting build...
INFO:    Creating SIF file...
INFO:    Build complete: test1.SIF

$ singularity run test1.SIF ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
<…shortened…>

$ alias ffmpeg="singularity run /scratch/hoidekr/test1.SIF ffmpeg"

$ ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
```

Start from debian:buster

test1.sbox – directory

Start the container – “interactive mode”

Install ffmpeg inside container

Singularity building own image

```
$ singularity build -f -s test1.sbox docker://debian:buster
INFO:    Starting build...
Getting image source signatures
<…shortened…>
Storing signatures
2021/04/18 23:06:28  info unpack layer: sha256:bd8f6…..
INFO:    Creating sandbox directory...
INFO:    Build complete: test1.sbox

$ singularity shell -f -w test1.sbox/
Singularity> apt update
<…shortened…>
Singularity> apt install ffmpeg -y
<…shortened…>
Singularity> exit

$ singularity build -f  test1.SIF test1.sbox/
INFO:    Starting build...
INFO:    Creating SIF file...
INFO:    Build complete: test1.SIF

$ singularity run test1.SIF ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
<…shortened…>

$ alias ffmpeg="singularity run /scratch/hoidekr/test1.SIF ffmpeg"

$ ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
```

Start from debian:buster

test1.sbox – directory

Start the container – “interactive mode”

Install ffmpeg inside container

Build SIF image from sandbox

Singularity building own image

```
$ singularity build -f -s test1.sbox docker://debian:buster
INFO:    Starting build...
Getting image source signatures
<…shortened…>
Storing signatures
2021/04/18 23:06:28  info unpack layer: sha256:bd8f6…..
INFO:    Creating sandbox directory...
INFO:    Build complete: test1.sbox

$ singularity shell -f -w test1.sbox/
Singularity> apt update
<…shortened…>
Singularity> apt install ffmpeg -y
<…shortened…>
Singularity> exit

$ singularity build -f  test1.SIF test1.sbox/
INFO:    Starting build...
INFO:    Creating SIF file...
INFO:    Build complete: test1.SIF

$ singularity run test1.SIF ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
<…shortened…>

$ alias ffmpeg="singularity run /scratch/hoidekr/test1.SIF ffmpeg"
$ ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
```

from debian:buster

test1.sbox – directory

Start the container – “interactive mode”

Install ffmpeg inside container

Build SIF image from sandbox

Test run of ffmpeg from container

Alias for command and run ffmpeg

Singularity definition file

- Definition file – recipe for building image
  - similar to Dockerfile
  - easy replication of environment

```
Bootstrap: docker
From: debian:buster

%post
apt-get update && apt-get install -y ffmpeg

%labels
    Author Jan Hoidekr
%help
    This is a demo def.file for CESNET - 21.4. 2021
```

- build image via definition file and run

```
$ singularity build -f test1.SIF test1.def
< … long output during building the image … >
$ singularity run test1.SIF ffmpeg
ffmpeg version 4.1.6-1~deb10u1 Copyright (c) 2000-2020 the FFmpeg developers
```

Singularity definition file

- Definition file
  - install compilers inside
  - compile application
  - environment vars
  - multi-stage
- No need to backup images
  - def.file is sufficient for repeatibility
- See user-guide

```
Bootstrap: docker
From: debian:buster
# see https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu
%post
apt-get update -qq && apt-get -y install \
  autoconf automake build-essential cmake git-core libass-dev
libfreetype6-dev \
  libgnutls28-dev libsdl2-dev   libtool   libva-dev libvdpau-dev
libvorbis-dev \
  libxcb1-dev libxcb-shm0-dev libxcb-xfixes0-dev   meson ninja-build \
  pkg-config texinfo wget yasm zlib1g-dev nasm

mkdir -p /opt/ffmpeg_sources /opt/bin

cd /opt/ffmpeg_sources && \
wget -O ffmpeg-snapshot.tar.bz2 https://ffmpeg.org/releases/ffmpeg-
snapshot.tar.bz2 && \
tar xjvf ffmpeg-snapshot.tar.bz2 && \
cd ffmpeg && \
PATH="/opt/bin:$PATH" PKG_CONFIG_PATH="/opt/ffmpeg_build/lib/pkgconfig"
./configure \
  --prefix="/opt/ffmpeg_build" --pkg-config-flags="--static" \
  --extra-cflags="-I/opt/ffmpeg_build/include" --extra-ldflags="-
L/opt/ffmpeg_build/lib" \
  --extra-libs="-lpthread -lm" --ld="g++" --bindir="/opt/bin" --enable-
gpl \
  --enable-libass --enable-libfreetype && \
PATH="/opt/bin:$PATH" make && \
make install

%runscript
    echo "ffmpeg METACENTRUM example"
    /opt/bin/ffmpeg $*
```

Singularity – “Integration, isolation”

- Isolation - apps inside “can not touch” outside environment and vice versa
- Integration
  - Bind directories from outside into container
    - -B /path/to/dir or -B /path/outside/:/inside/path
    - -H for homedir
    - Metacentrum defaults
      - -B /storage –B /tmp -H
  - Environment vars – *most of env vars* passed into container, else use `--env`
  - GPU
    - --nv for NVIDIA (--rocm for AMD)

Singularity – “Integration over isolation”

- /storage default binded - privilleges as running user

```
$ singularity shell docker://debian:buster
INFO:    Using cached SIF image
Singularity> echo OK > /storage/praha1/home/hoidekr/testfile
Singularity> exit
$ cat  /storage/praha1/home/hoidekr/testfile
OK
```

- Environment variables

```
$ singularity shell docker://debian:buster
INFO:    Using cached SIF image
Singularity> set | grep PBS
PBS_ENVIRONMENT=PBS_INTERACTIVE
PBS_JOBCOOKIE=095EDF6D5AA7B9847ED0E317496BC1D9
PBS_JOBDIR=/storage/brno3-cerit/home/hoidekr
PBS_JOBID=2452191.cerit-pbs.cerit-sc.cz
PBS_JOBNAME=STDIN
PBS_MOMPORT=15003
PBS_NCPUS=2
PBS_NGPUS=1
PBS_NODEFILE=/var/spool/pbs/aux/2452191.cerit-pbs.cerit-sc.cz
... ...
Singularity>
```

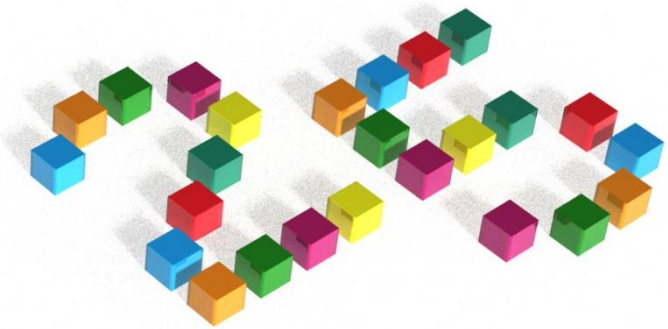
Containers in Metacentrum

- Benefits of containers:
  - Independent on predefined modules, libs, apps, …
  - Suitable for testing new tools
  - Sharing environment in group of users – project folder, CVMFS, public
  - Definition files as version history.
  - Repeatability of tasks

Containers in Metacentrum

- Benefits of containers:
  - Independent on predefined modules, libs, apps
  - Suitable for testing new tools
  - Sharing environment in group of users – project folder, public
  - Definition files as version history.
  - Repeatability of tasks

- Your questions ?



Thanks for your attention!
Děkuji za pozornost!

Jan Hoidekr, hoidekr@cesnet.cz