CernVM 5

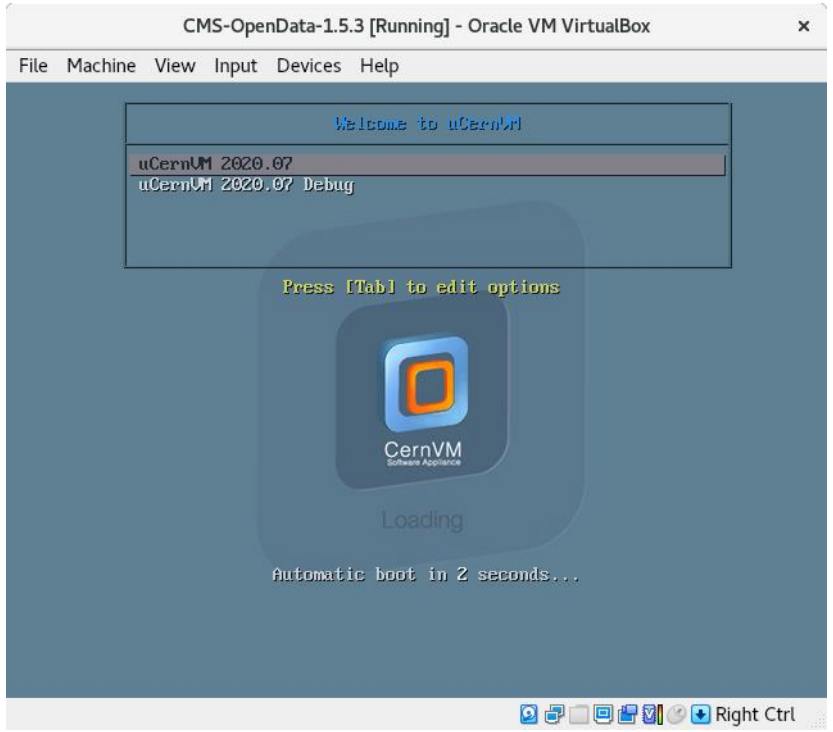
github.com/cernvm/cernvm-five

CernVM Appliance

- Virtual Appliance for CernVM-FS

→ **A minimal yet complete run time environment for scientific software**

- Version 3&4:
  - Minimal Linux kernel + µCernVM bootloader
  - CernVM-FS mounted during boot process; system loaded on-demand
  - Volume of ~20MB, but designed for full virtualization
  - Interactive contextualization through CernVM Online



Development of CernVM (until the end of 2022)

- Release of CernVM 5
- CernVM 3&4 remain available for long-term data preservation use cases
- Phase out CernVM Online
  →Instead: Interactive contextualization through pre-contextualized OVA images

Problem of CernVM ≤ 4: Limited usability as a container

- **Operators are shifting infrastructure towards containerized environments**
  - Smaller images
  - Easier to distribute / retain
  - Less performance overhead
  - Easier and faster to deploy
  - Powerful orchestration tools (e. g. Kubernetes)

CernVM 5

- Standard “just-enough” container
- Still a complete platform to **develop** and **run** HEP applications

→**But: Focus on container virtualization**

- Minimal: Contents defined by CernVM-FS and common HEP base libraries
- Facilitates the mounting of /cvmfs in various environments
- Equally practical as a container and full virtual machine
  →Running in different container run times and hypervisors

CernVM 5: Goals

- Building a container image for a variety of applications, rather than one binary
  - →And keeping it minimal
- Mounting CernVM-FS inside a container
  - →Mounting a file system usually requires elevated privileges
  - →Privileges by default not available in containerized environments
- Stable usability in the presence of multiple versions of shared libraries
  - →While not interfering with scientific stacks setups
  - →But built from stock packages
- Graphical user interface
  - →Despite the focus on container virtualization
  - →Without host prerequisites
- CernVM-FS cache management
  - →Reducing start latency by building images with pre-loaded cache

CernVM 5: Goals

- Building a container image for a variety of applications, rather than one binary
  - →And keeping it minimal
- Mounting CernVM-FS inside a container
  - →Mounting a file system usually requires elevated privileges
  - →Privileges by default not available in containerized environments
- Stable usability in the presence of multiple versions of shared libraries
  - →While not interfering with scientific stacks setups
  - →But built from stock packages
- Graphical user interface
  - →Despite the focus on container virtualization
  - →Without host prerequisites
- CernVM-FS cache management
  - →Reducing start latency by building images with pre-loaded cache

CernVM 5: Design

- Base Layer Image
  - CernVM-FS client
  - HEP base libraries
  - dnf package manager

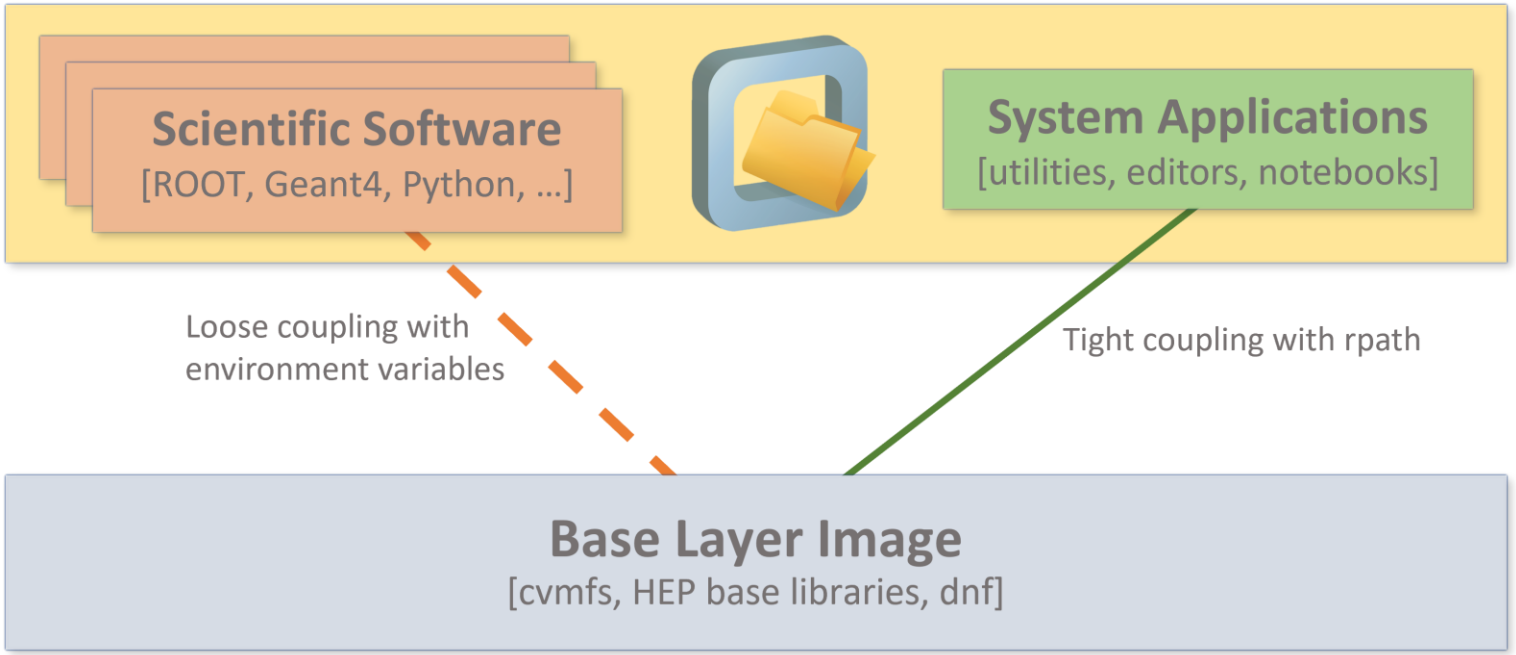
→ The actual image, smallest unit

- System Applications
  - CernVM 5 system specific
  - Interactive applications

→ What users expect on top

- Scientific Software Stacks
  - E. g. analysis frameworks
  - Set up using environment variables

→ Externally managed



CernVM 5: Image Build Process

- Custom multi-stage build process
  - 1.) Set up intermediate build container
  - 2.) Construct CernVM 5 root file system in build directory
    → Defined in **cernvm-system-default** package (HEP libraries, CernVM-FS, configuration files)
  - 3.) and 4.) Export build directory to standalone image

```
# EL8-based intermediate builder
FROM rockylinux:8 AS builder ❶
RUN mkdir /build
RUN dnf --installroot=/build cernvm-system-default ❷
```

- /etc
- /bin
- **/build ←**
- …

```
FROM scratch as cernvm-five
COPY --from=builder /build  / ❸
```

```
# CernVM 5 base layer
# ~800 MB
```

- /etc
- /bin
- /cvmfs
- …

```
FROM cernvm-five
❹
```

```
# Derived CernVM 5 image
RUN dnf install …
```

CernVM 5: System Applications on CernVM-FS

Idea: Outsource system applications to CernVM-FS

+ Smaller image size
+ General advantages of CernVM-FS like e.g., lazy loading
+ Versioned by CernVM-FS

Subsystem-like installation

+ Easily extendable using standard package managers

System Applications Repository on CernVM-FS

[vim, nano, emacs, findutils, patchelf, ping, wget, strace, tree, diff, …]

CernVM 5: Resulting Base Layer Image

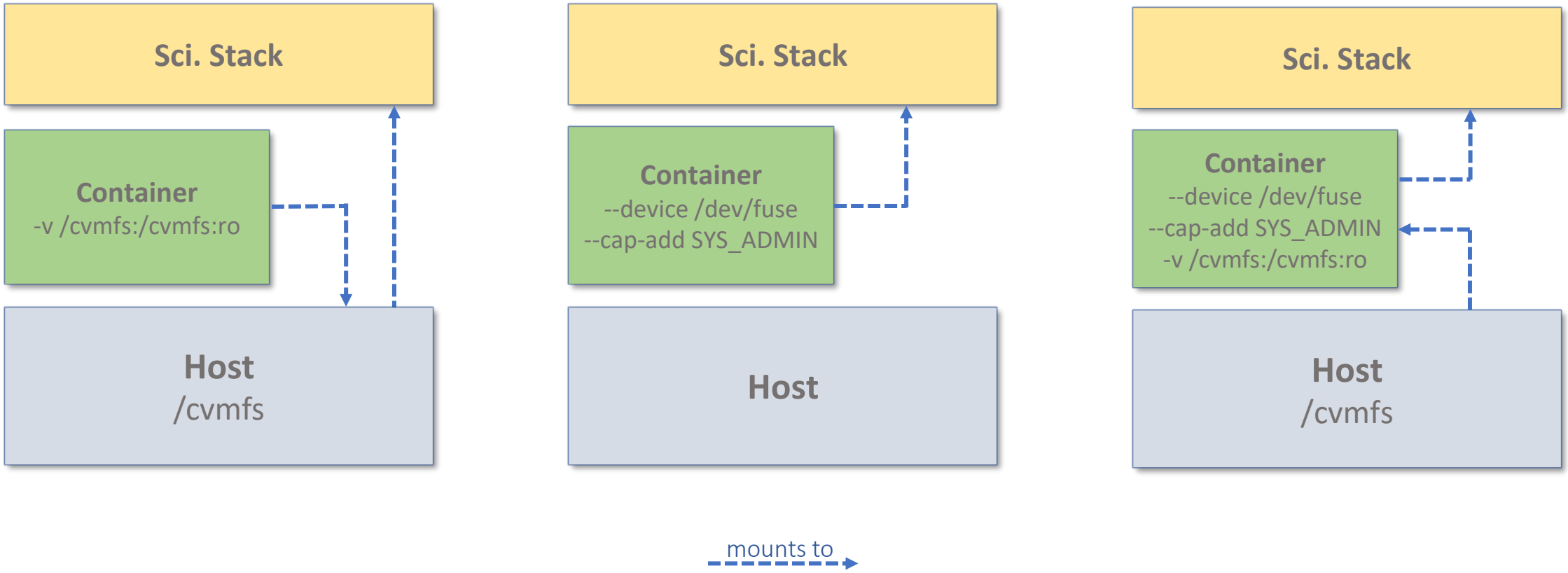
	CernVM 5 Base Layer Image	Naively Derived Image	Naively Derived Image + System Applications
Volume (uncompressed)	805MB	1030MB	1450MB
Volume (compressed)	284MB	382MB	523MB
Installed Packages	457	502	612
Image Layers	single	multiple	multiple
Standard Derivability	yes	yes (but not a true base layer)	yes (but not a true base layer)

+ Better control over installed packages → smaller image, easier to distribute
+ Improved build time and caching
+ Single layered → less complex, faster to construct, true to concept of base layers
+ Compared to ≤ 4: Derivable image

CernVM 5: Goals

- Building a container image for a variety of applications, rather than one binary
  - →And keeping it minimal
- Mounting CernVM-FS inside a container
  - →Mounting a file system usually requires elevated privileges
  - →Privileges by default not available in containerized environments
- Stable usability in the presence of multiple versions of shared libraries
  - →While not interfering with scientific stacks setups
  - →But built from stock packages
- Graphical user interface
  - →Despite the focus on container virtualization
  - →Without host prerequisites
- CernVM-FS cache management
  - →Reducing start latency by building images with pre-loaded cache

CernVM 5: Mounting CernVM-FS



CernVM 5: Deployment

	Docker	Apptainer	Podman	Kubernetes	containerd
Bind mount CernVM-FS -v /cvmfs:/cvmfs:ro	✓	✓	✓	✓	✓
Mount CernVM-FS insde --device /dev/fuse --cap-add SYS_ADMIN	✓	✓	✓	✓	✓
Pre-mounted	✗	✓	✗	✗	✗
Host integration /workspace → as a shared folder /data → physics data	✓	✓	✓	✓	✓

CernVM 5: Distribution

Pushed to registry
- Docker-compliant Registry

→ End users

CernVM-FS
→ Large scale distribution

As a VM
- CERN OpenStack

→ Isolation use cases
→ Cloud infrastructure
→ Data acquisition systems

```
$ docker pull registry.cern.ch/cernvm/five/cernvm-five:latest
$ apptainer shell  /cvmfs/unpacked.cern.ch/registry.cern.ch/cernvm/five/cernvm-five\:latest
$ openstack server create --image 'CernVM 5.1.2 – x86_64'
```

CernVM 5: Goals

- Building a container image for a variety of applications, rather than one binary
  - →And keeping it minimal
- Mounting CernVM-FS inside a container
  - →Mounting a file system usually requires elevated privileges
  - →Privileges by default not available in containerized environments
- Stable usability in the presence of multiple versions of shared libraries
  - →While not interfering with scientific stacks setups
  - →But built from stock packages
- Graphical user interface
  - →Despite the focus on container virtualization
  - →Without host prerequisites
- CernVM-FS cache management
  - →Reducing start latency by building images with pre-loaded cache

Problem: Presence of multiple versions of shared libraries

```
bash-4.4# which nano
/cvmfs/cernvm-five.cern.ch/x86_64/1.0a/usr/bin/nano
bash-4.4# source /cvmfs/sw.hsf.org/spackages4/key4hep-stack/release-2021-10-29-ip7764o/x86_64-centos8-gcc8.4.1-opt/setup.sh
bash-4.4# nano
Segmentation fault (core dumped)
```

```
bash-5.1# ldd /cvmfs/cernvm-five.cern.ch/x86_64/1.0a/usr/bin/nano
        linux-vdso.so.1 (0x00007f6d5ac3b000)
        libmagic.so.1 => /cvmfs/sw.hsf.org/spackages4/file/5.40-qsol6zg/x86_64-centos8-gcc8.4.1-opt/lib/libmagic.so.1 (0x00007f6d5a7bf000)
        libncursesw.so.6 => /cvmfs/sw.hsf.org/spackages4/ncurses/6.2-rxkbhc6/x86_64-centos8-gcc8.4.1-opt/lib/libncursesw.so.6 (0x00007f6d5a
        libtinfo.so.6 => /cvmfs/sw.hsf.org/spackages4/ncurses/6.2-rxkbhc6/x86_64-centos8-gcc8.4.1-opt/lib/libtinfo.so.6 (0x00007f6d5a34b000
```

- Problem: Stack setup scripts overwrite local default values

→Locally and remotely installed executables break when linked with mismatching shared library

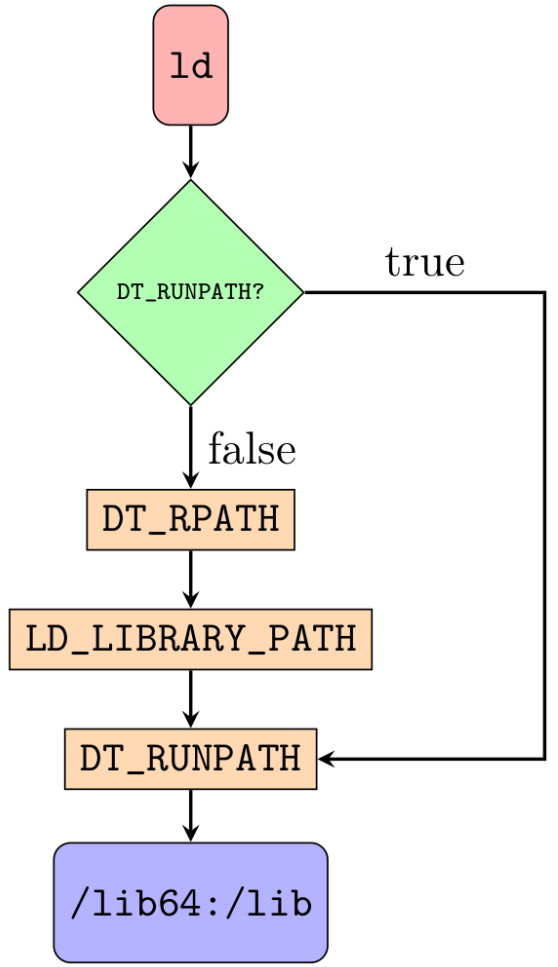
Dynamically Linked Executables

- Dynamically linked ELF executables
  - Use shared libraries installed on a system
  - Linked with dependencies during run time using a linker, e. g., ld
- Possible solutions
  - Static linking

  →Increases image size
  - rpath at compile-time

  →Requires custom packages / maintenance

→ **Standard packages with post-build rpath processing**



Standard Packages with Post-Build rpath

```
bash-5.1# cernvm-patch-rpath -h
Adds DL_RPATH to ELF executables using patchELF
cernvm-patch-rpath [-r <installation root (by default '/')> -l <location of shared libraries (by default '/lib64:/lib' resp. '/lib')>
```

- cernvm-patch-rpath:
  - 1.) Determine installed executables
  - 2.) Evaluate executable and original location of dependencies
    → E. g., /lib64 for a locally installed 64-bit, dynamically linked ELF executable
  - 3.) Set up DT_RPATH binary header accordingly

Pros:
+ Fast, repeatable
+ Isolated from LD_LIBRARY_PATH
+ Integrated in build chain
+ Can be used by users in their builds (RUN cernvm-patch-rpath)
+ Allows use of standard packages

Example: CernVM 5 executable with post-build applied rpath

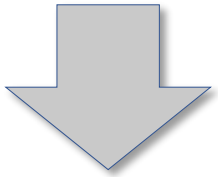
```
bash-5.1# patchelf --print-rpath /cvmfs/cernvm-five.cern.ch/x86_64/1.2/usr/bin/nano
/cvmfs/cernvm-five.cern.ch/x86_64/1.2/lib64:/cvmfs/cernvm-five.cern.ch/x86_64/1.2/lib
```

```
bash-5.1# which nano
/cvmfs/cernvm-five.cern.ch/x86_64/1.2/usr/bin/nano
bash-5.1# source /cvmfs/sw.hsf.org/spackages4/key4hep-stack/release-2021-10-29-ip7764o/x86_64-centos8-gcc8.4.1-opt/setup.sh
bash-5.1# nano
bash-5.1# echo $?
0
```

```
bash-5.1# ldd $(which nano)
        linux-vdso.so.1 (0x00007ffd7abea000)
        libmagic.so.1 => /cvmfs/cernvm-five.cern.ch/x86_64/1.2/lib64/libmagic.so.1 (0x00007fd7f1f1d000)
        libncursesw.so.6 => /cvmfs/cernvm-five.cern.ch/x86_64/1.2/lib64/libncursesw.so.6 (0x00007fd7f1eda000)
        libtinfo.so.6 => /cvmfs/cernvm-five.cern.ch/x86_64/1.2/lib64/libtinfo.so.6 (0x00007fd7f1ea9000)
```

Patching embedded in Container Image Build

```
### Add rpath to executables ###
RUN cernvm-patch-rpath -r ${BUILD_DIR}
```



```
bash-5.1# patchelf --print-rpath /usr/bin/ls
/lib64:/lib
```

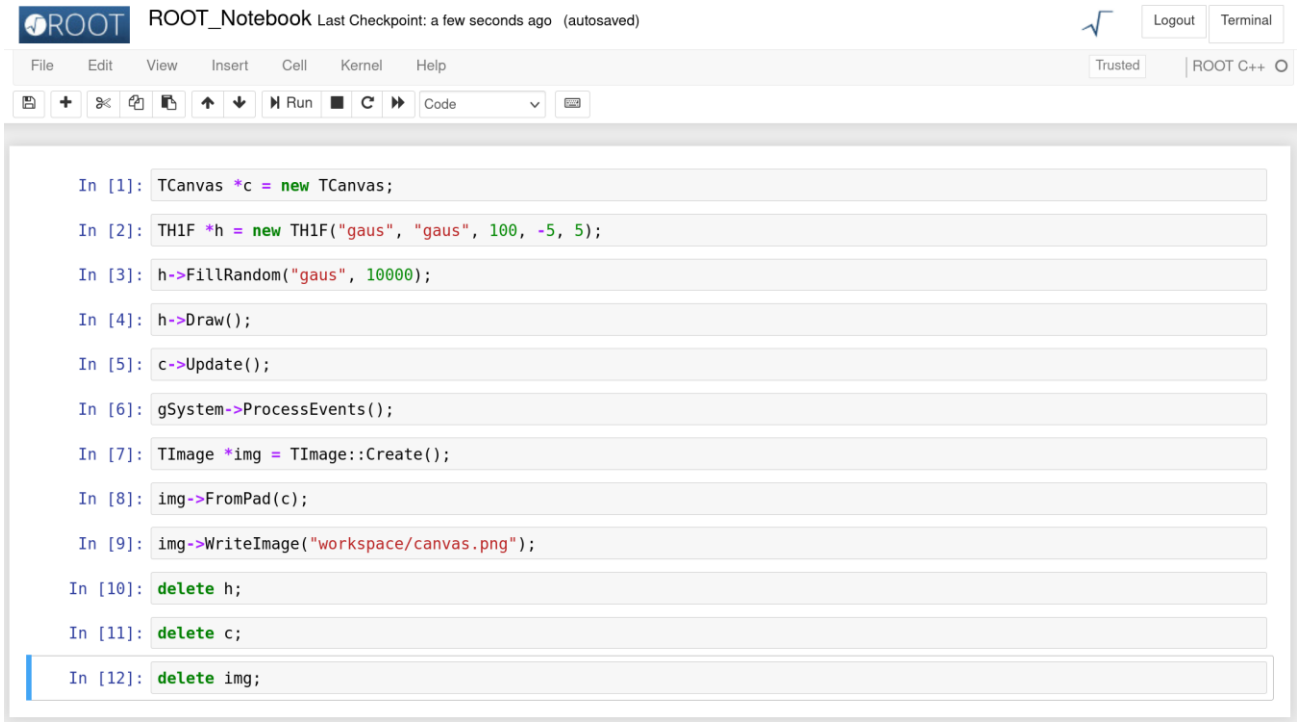
CernVM 5: Goals

- Building a container image for a variety of applications, rather than one binary
  - →And keeping it minimal
- Mounting CernVM-FS inside a container
  - →Mounting a file system usually requires elevated privileges
  - →Privileges by default not available in containerized environments
- Stable usability in the presence of multiple versions of shared libraries
  - →While not interfering with scientific stacks setups
  - →But built from stock packages
- Graphical user interface
  - →Despite the focus on container virtualization
  - →Without host prerequisites
- CernVM-FS cache management
  - →Reducing start latency by building images with pre-loaded cache

CernVM 5: Graphical User Interface

- Graphical applications in containers
  - Clumsy
  - Too large
  - Tedious to configure
  - Host prerequisites

- Web-based Jupyter Notebooks
  - Loaded from CernVM-FS
  - Hosted locally (EXPOSE 8888)

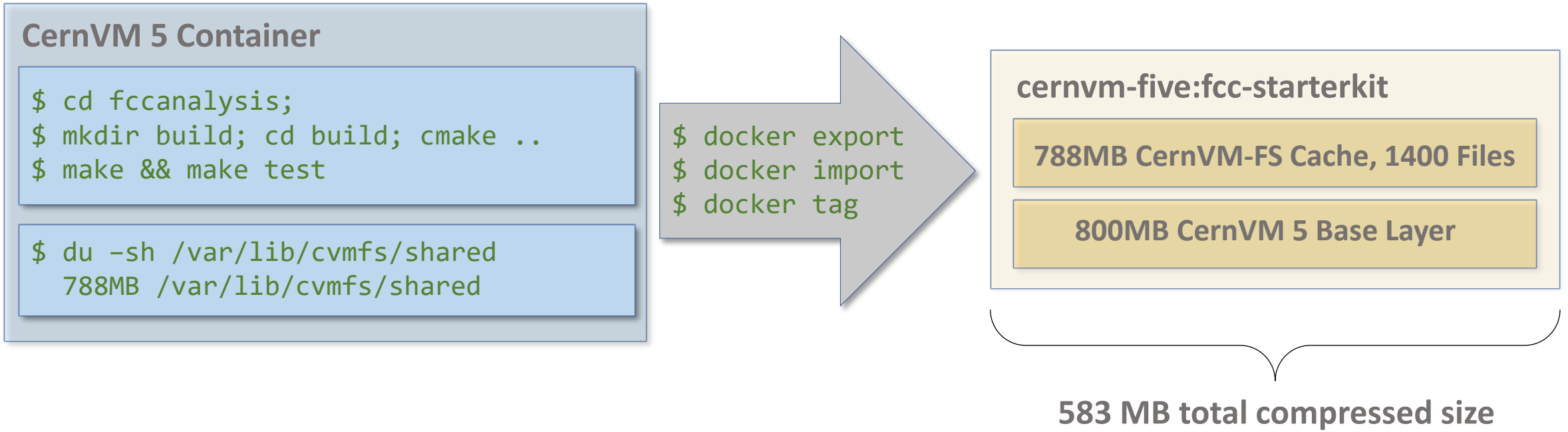


CernVM 5: Goals

- Building a container image for a variety of applications, rather than one binary
  → And keeping it minimal
- Mounting CernVM-FS inside a container
  → Mounting a file system usually requires elevated privileges
  → Privileges by default not available in containerized environments
- Stable usability in the presence of multiple versions of shared libraries
  → While not interfering with scientific stacks setups
  → But built from stock packages
- Graphical user interface
  → Despite the focus on container virtualization
  → Without host prerequisites
- CernVM-FS cache management
  → Reducing start latency by building images with pre-loaded cache

Cache Pre-Loading – An Example

- Scenario / Workflow mapped in a script / testsuite
- Example FCC Starterkit tutorial



Speed Up (Inside CERN Network)

Command	CernVM 5 Base Image	CernVM 5 FCC Starterkit	Rounded Speed Up Factor
KKMCee -h	1.609s	0.357s	4.5x
BHLUMI -h	1.569s	0.055s	28.0x
whizard Z_mumu.sin	50.233s	18.971s	2.6x
babayaga -h	1.635s	0.487s	3.4x
babayaga -f [1]	5.541s	5.289s	1.05x
Import ROOT	6.843s	1.544s	4.4x
fccanalysis run [2]	11.230s	9.319s	1.2x

1 babayaga -f 15. -t 165. -e 91.2 -n 10000 -o bbyg_10000.LHE
2 fccanalysis run analysis_stage1.py --output p8_ee_ZH_ecm240.root --files-list ./p8_ee_ZH_ecm240_edm4hep.root
http://ecsft.cern.ch/dist/cernvm/five/vm/tutorials/cernvm-five-fcc-x86_64.qcow2.tar.gz

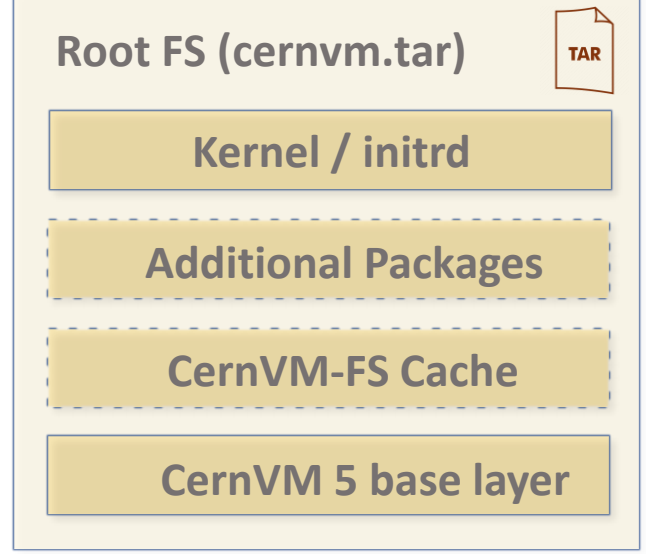
CernVM 5: Full Virtual Machine

- Build on top of any CernVM 5 container image / stopped container
  - Adding Kernel-enabled file system layer to CernVM base image

```
# Deriving from base layer (or CernVM 5 child images)
FROM registry.cern.ch/cernvm/five/cernvm-five:latest

# Installing kernel and configurations, generating initrd
RUN dnf install cernvm-kernel-default
```

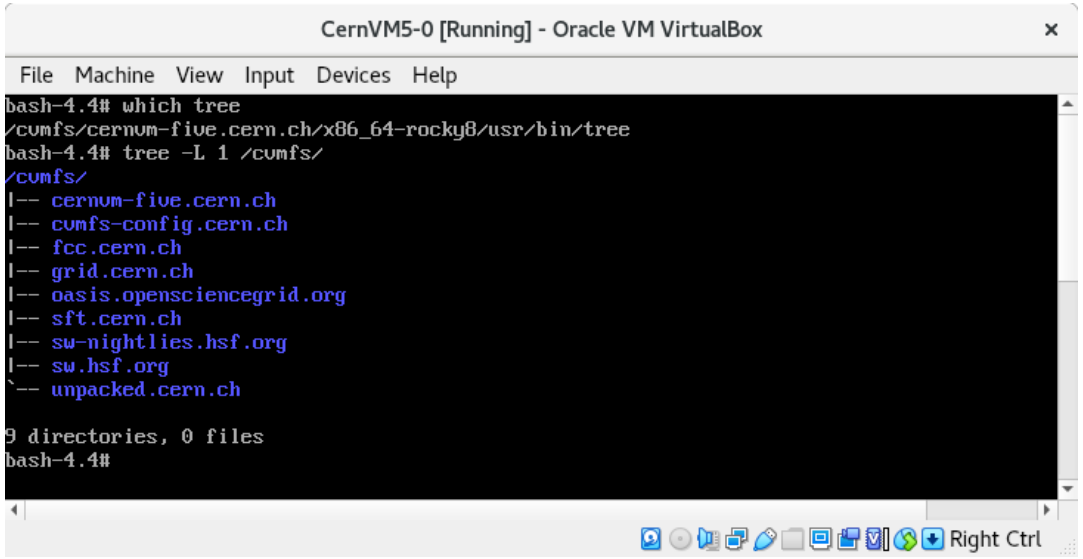
- /cvmfs
- **/etc** ←
- **/boot** ←
- …

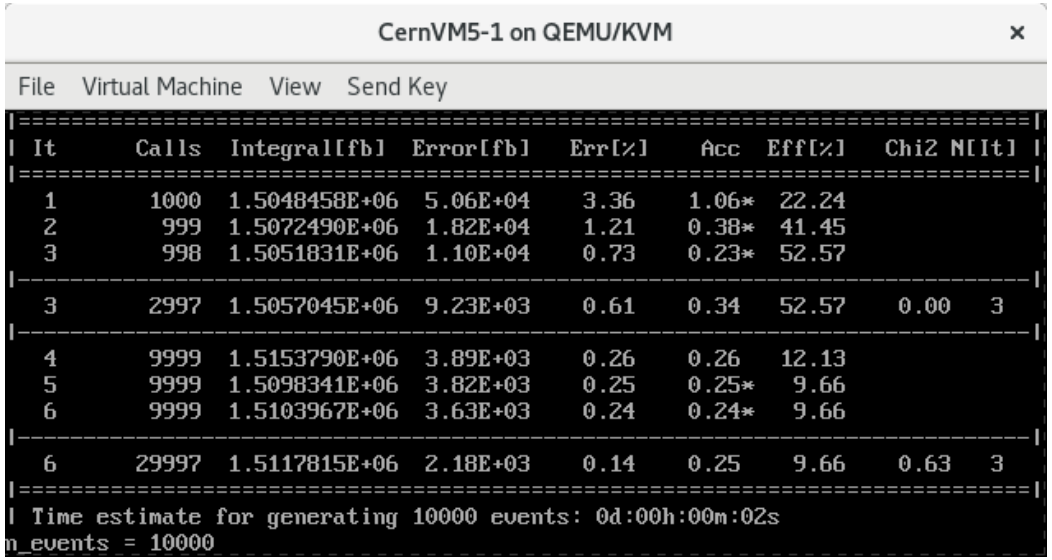
```
# Construct rootfs
$ docker create vm

# export rootfs
$ docker export vm
```



CernVM 5: Virtual Machine

- Available as raw and qcow2 image e. g. for QEMU/KVM or VirtualBox
- Available on CERN OpenStack within the project (soon public)
  - Fully integrated contextualization process
- Build in an automated manner; in line with the container build process
  - Every existing CernVM 5 container image and container extendable to full VM





CernVM 5: Adoption

- Key4hep project

→As a build environment for future stacks

- CERN’s web-based analysis service SWAN

→ As the underlying runtime environment

CernVM 5: Conclusion

- Standard container image capable of serving arbitrary stacks as a run time environment
- Built in a custom process using standard packages
- Processing of dynamically linked executables with DT_RPATH
- Natively derivable
- Integrated and tested FUSE interface for mounting CernVM-FS inside a container running in various container runtimes
- Extendable to a full virtual machine
- Proof-of-concept for pre-loading CernVM-FS caches

Questions?