



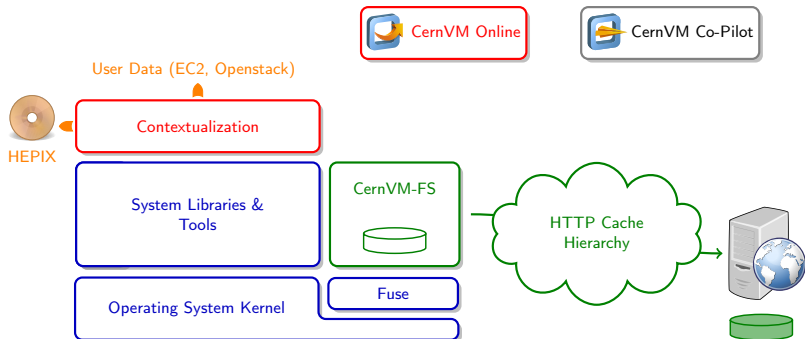
μ CernVM: Slashing the Cost of Building and Deploying Virtual Machines

J Blomer, D Berzano, P Buncic, I Charalampidis, G Ganis,
G Lestaris, R Meusel, V Nicolaou

jblomer@cern.ch

CERN PH-SFT
CHEP 2013, Amsterdam

CernVM 2.X



- Used on laptops, clouds, volunteers' computers (LHC@Home 2.0)
- Uniform and portable environment for physics data processing
- "Just enough operating system" derived from application dependencies

CernVM 2.X series

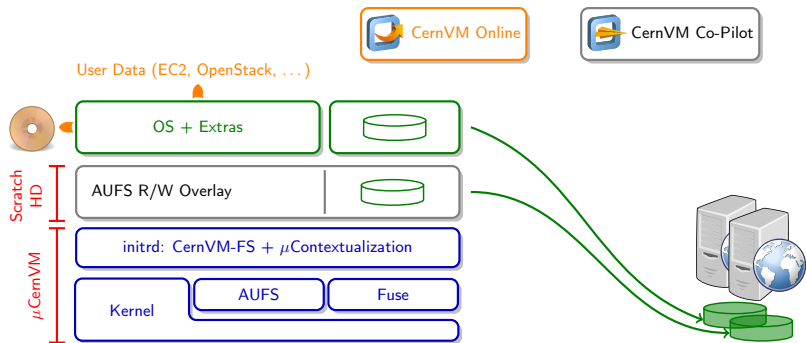
- +** **Versatile image**
supports all hypervisors
supports all LHC experiments
various roles: desktop, batch, ...
- +** **Strongly versioned** components
Single version number defines VM
(by Conary package manager)
- !** Scientific Linux 5 based
- !** Significant effort to
build, test, deploy images

Goals for CernVM 3

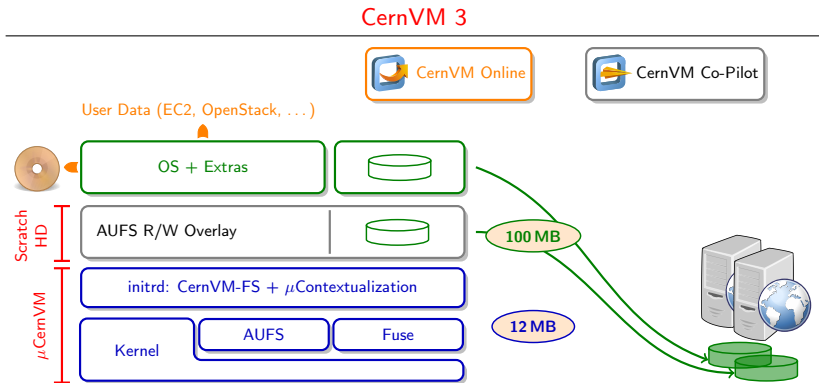
- 1 Update base platform
Scientific Linux 6, RPM
support for cloud-init
- 2 Instant feedback during
appliance development:
minutes instead of hours
- 3 Instant instantiation
of virtual machines:
seconds instead of minutes

Idea: CernVM-FS can provide **operating system on demand**

CernVM 3



Twofold system: μ CernVM boot loader + OS delivered by CernVM-FS



Twofold system: μ CernVM boot loader + OS delivered by CernVM-FS

⇒ **Drastic reduction in size**

From “just enough operating system” to “operating system on demand”
 400 MB image (compressed) \mapsto 12 MB image + 100 MB cache

① Part I: μ CernVM Boot Loader

② Part II: Operating System on CernVM-FS

CernVM Kernel: 3.10 long-term kernel (2 year support)

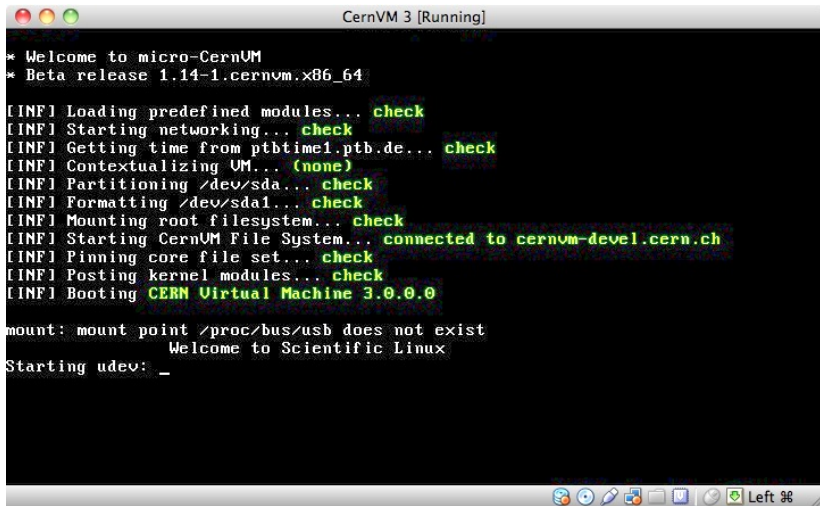
Features: KSM, zRam, THP, cgroups, X32-ABI

Extra modules: AUFS, VMware drivers, VBox drivers, OpenAFS

“Virtualization-friendly”, minimal set of device drivers:

100 modules / 8 MB **as compared to** 2000 modules / 120 MB in SL6

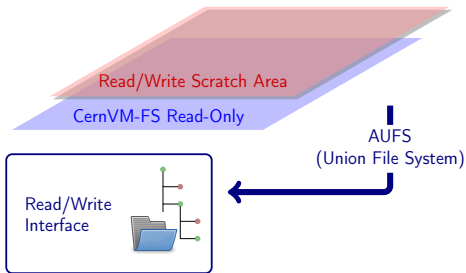
- 1 Execute SYSLINUX boot loader
- 2 Decompress and load Linux kernel
- 3 Decompress init ramdisk, execute customized `/init`
 - a) Start networking
 - b) Contextualize (supports EC2, OpenStack, OpenNebula, vSphere)
 - c) [Partition and] [format and] mount scratch space
 - d) Mount CernVM-FS
 - e) Mount AUFS root file system stack
 - f) Change root file system and start operating system



```
CernVM 3 [Running]
* Welcome to micro-CernUM
* Beta release 1.14-1.cernvm.x86_64

[INF] Loading predefined modules... check
[INF] Starting networking... check
[INF] Getting time from ptbtime1.ptb.de... check
[INF] Contextualizing VM... (none)
[INF] Partitioning /dev/sda... check
[INF] Formatting /dev/sda1... check
[INF] Mounting root filesystem... check
[INF] Starting CernUM File System... connected to cernvm-devel.cern.ch
[INF] Pinning core file set... check
[INF] Posting kernel modules... check
[INF] Booting CERN Virtual Machine 3.0.0.0

mount: mount point /proc/bus/usb does not exist
      Welcome to Scientific Linux
Starting udev: _
```

CernVM-FS features targeted to loading the OS:

- Closing all read-write file descriptors in order to unravel file system stack on shutdown
 - Redirect syslog messages
 - In-flight change of DNS server
 - GID / UID mappings
- ⇒ This file system stack requires special support from a read-only Fuse branch since it is started before the operating system.

① Part I: μ CernVM Boot Loader

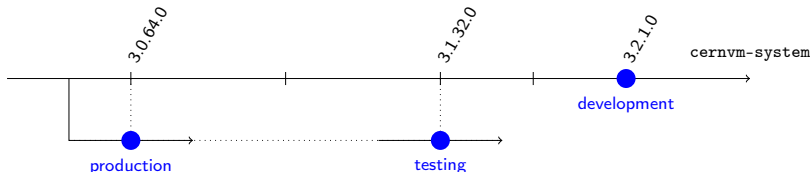
② Part II: Operating System on CernVM-FS

General idea: Install packages with yum into a CernVM-FS chroot jail

Problem: Typical package repositories are not designed to *preserve* an environment

The CernVM 3 build process **ensures strong versioning** on three levels

- ① `cernvm-system` meta RPM
fully versioned dependency closure
- ② Named branches in the CernVM-FS repository
- ③ Versioned snapshots provided by CernVM-FS
allow the very same image to instantiate any `cernvm-system` version
helpful for **long-term data preservation**

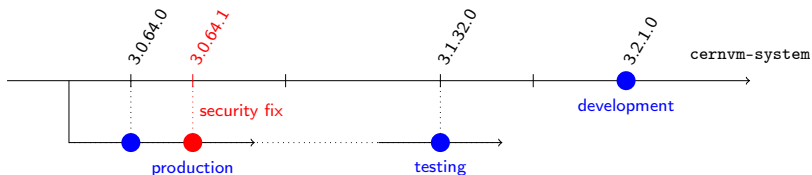


General idea: Install packages with yum into a CernVM-FS chroot jail

Problem: Typical package repositories are not designed to *preserve* an environment

The CernVM 3 build process **ensures strong versioning** on three levels

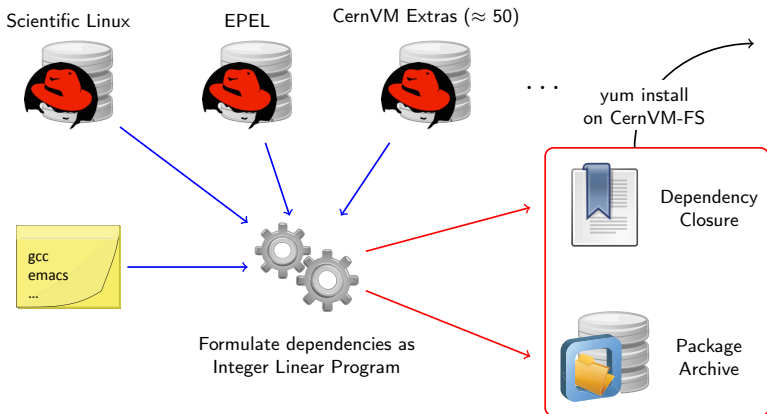
- ① `cernvm-system` meta RPM
fully versioned dependency closure
- ② Named branches in the CernVM-FS repository
- ③ Versioned snapshots provided by CernVM-FS
allow the very same image to instantiate any `cernvm-system` version
helpful for **long-term data preservation**

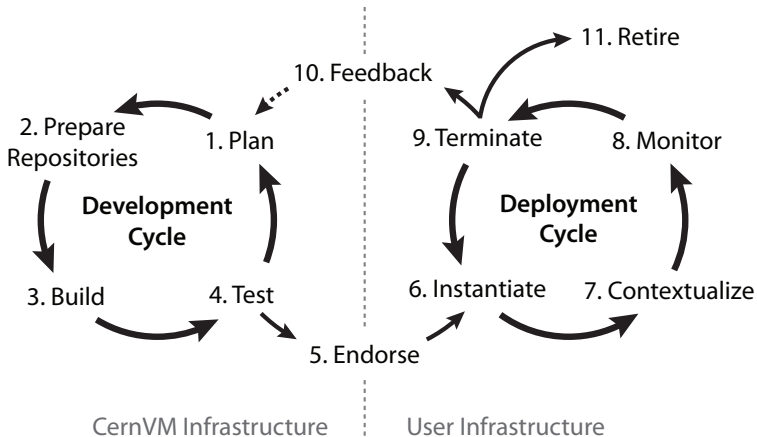


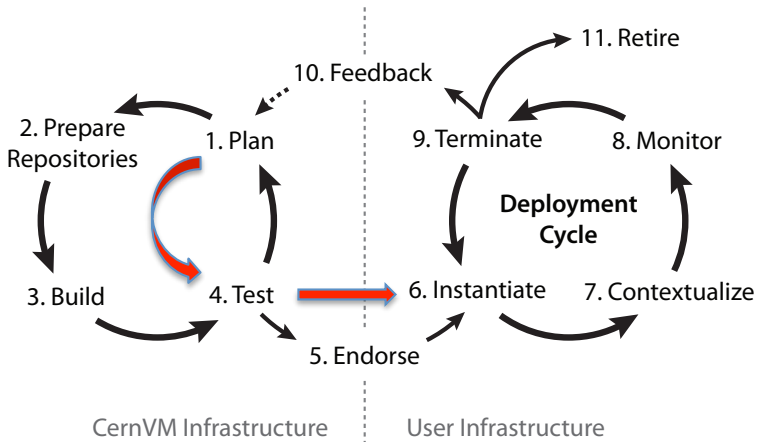
Maintenance of the repository **should not** become a Linux distributor's job

But: should be reproducible and well-documented

Idea: automatically generate a **fully versioned, closed** package list from a "shopping list" of unversioned packages







Avoids: Image Building

Solves: Image Distribution

Options for updating: stay, diverge, **rebase**

Rebase high-level perspective:

- 1 On first boot, CernVM selects and pins newest available version
- 2 Automatic update notifications
- 3 Applying updates requires a reboot
Most security critical updates require a reboot anyway

μ CernVM Bootloader

- boot partition read-only, updates dropped on ephemeral storage
- 2 phase boot:
start old kernel and ramdisk, kexec into updated version

CernVM-FS OS Repository

- Mount updated CernVM-FS snapshot
- Conflict resolution wrt. local changes
 - 1 keep local configuration
 - 2 map user/group ids
 - 3 merge rpm database

Options for updating: stay, diverge, **rebase**

Rebase high-level perspective:

- 1 On first boot, CernVM selects and pins newest available version
- 2 Automatic update notifications
- 3 Applying updates requires a reboot
Most security critical updates require a reboot anyway

μ CernVM Bootloader

- boot partition read-only, updates dropped on ephemeral storage
- 2 phase boot:
start old kernel and ramdisk, **kexec** into updated version

CernVM-FS OS Repository

- Mount updated CernVM-FS snapshot
- Conflict resolution wrt. local changes
 - 1 keep local configuration
 - 2 map user/group ids
 - 3 merge rpm database



The screenshot shows the CernVM Online web interface. At the top, there is a navigation bar with the CernVM Online logo and a search bar. Below the navigation bar, there are several tabs: About, Dashboard, Documentation, Downloads, and Publications. The main content area is titled "Dashboard" and "Your context definitions". It displays a table with columns for Name, ID, Operations, and Deployment. The table contains one entry for "Test A" with ID "ee0f0672cd634304bb1f2fa5b7a54d4d". The Operations column has buttons for Clone, a context icon, and a context icon. The Deployment column has a WebAPI button. A dropdown menu is open over the context icon buttons, showing three options: "1 CPU / 1 GB RAM / 10 GB disk", "1 CPU / 2 GB RAM / 10 GB disk", and "2 CPU / 2 GB RAM / 20 GB disk". Below the table, there is a section titled "Your virtual machines" with a table that is currently empty, showing "No instances paired yet". There are also buttons for "Create new context" and "Pair an instance of CernVM". At the bottom of the page, there is a footer with the text "© Copyright CERN 2012 - PH Department - SFT - CernVM Software appliance".

Logged in as admin | Profile | Log out

Commands
Dashboard
Pair an instance
Create Context

Recent Definitions
Test A

Dashboard

Your context definitions

Name	ID	Operations	Deployment
Test A	ee0f0672cd634304bb1f2fa5b7a54d4d	Clone	WebAPI

Create new context

Your virtual machines

Machine	CernVM	Context
No instances paired yet		

Pair an instance of CernVM

1 CPU / 1 GB RAM / 10 GB disk
1 CPU / 2 GB RAM / 10 GB disk
2 CPU / 2 GB RAM / 20 GB disk

© Copyright CERN 2012 - PH Department - SFT - CernVM Software appliance

Web browser plugin developed by Ioannis Charalampidis for
<http://crowdcrafting.org/app/cernvm>



The screenshot displays the CernVM Online web interface. At the top left, the CernVM Online logo is visible. The top right shows the user is logged in as 'admin' with links for 'Profile' and 'Log out'. A search bar is also present. The main navigation menu on the left includes 'About', 'Dashboard', 'Commands', 'Recent Definitions', and 'Test A'. The 'Commands' section lists 'Dashboard', 'Pair an instance', and 'Create Context'. The 'Recent Definitions' section shows 'Test A'. The central area features a large window titled 'Test A VM [Running]'. This window displays a terminal-like interface with the following content: 'Welcome to uCernVM', 'uCernVM', 'Debug', 'Press [Tab] to edit options', the CernVM Software Appliance logo, 'Loading', and 'Automatic boot in 1 second...'. The window has a standard macOS-style title bar with red, yellow, and green buttons. The background of the interface is a dark blue grid pattern.



- 1 CernVM 3 releases are created in few minutes
- 2 A single virtual machine image of only 12 MB can instantiate any CernVM 3 version ever released

The reality check

- Build CernVM 3 components
- PROOF and HTCondor clusters
- ATLAS, ALICE event viewers
- CMS, LHCb reconstruction

Next steps

- Systematic testing!
- Use WLCG infrastructure for CernVM-FS OS repository

We are happy for feedback!

Please find **beta releases** and build system **sources** under:

<http://cernvm.cern.ch/portal/ucernvm>

<https://github.com/cernvm>

More about **CernVM Online**

G Lestaris et al.: *CernVM Online and Cloud Gateway: a uniform interface for CernVM contextualization and deployment*

<http://chep2013.org/contrib/185>

More about **PROOF as a Service**

D Berzano et al.: *PROOF as a Service on the Cloud: a Virtual Analysis Facility based on the CernVM ecosystem*

<http://chep2013.org/contrib/185>

③ Backup Slides

Hypervisor / Cloud Controller	Status
VirtualBox	✓
VMware	✓
KVM	✓
Xen	✓
Microsoft HyperV	?
Parallels	⚡ ⁴
Openstack	✓
OpenNebula	✓ ³
Amazon EC2	✓ ¹
Google Compute Engine	⚡ ²

¹ Only tested with ephemeral storage, not with EBS backed instances

² Waiting for custom kernel support

³ Only amiconfig contextualization

⁴ Unclear license of the guest additions

Normalized (Integer) Linear Program:

$$\text{Minimize } (c_1 \cdots c_n) \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{subject to} \quad \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \leq \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

Here: every available (package, version) is mapped to a $x_i \in \{0, 1\}$.

Cost vector: newer versions are cheaper than older versions.

(Obviously: less packages cheaper than more packages.)

Dependencies:

Package x_a requires x_b or x_c : $x_b + x_c - x_a \geq 0$.

Packages x_a and x_b conflict: $x_a + x_b \leq 1$.

(...)

Figures

≈17 000 available packages ($n = 17000$), 500 packages on “shopping list”

≈160 000 inequalities ($m = 160000$), solving time <10 s (glpk)

Meta RPM: ≈1 000 fully versioned packages, dependency closure