CernVM-FS Graphdriver Plugin for Docker

CERN, SFT Group Meeting

Nikola Hardi
nhardi@cern.ch

March 20, 2017

Supervisors:
Jakob Blomer : jblomer@cern.ch
Gerardo Ganis: gerardo.ganis@cern.ch

About me

- Faculty of Technical Sciences,
  University of Novi Sad, Serbia.
- Openlab summer student, 2016.
  A web interface for Linux perf.
- Iterative compilation in LLVM and
  performance prediction by static analysis.
- Open Source communities:
  LUGoNS, BalCCon, LiBRE!
- TA in Petnica Science Centre:
  Applied Physics and Electronics.
- Joined you in October as technical student.
  Project in colaboration with KT Fund.

CERN openlab

Petnica

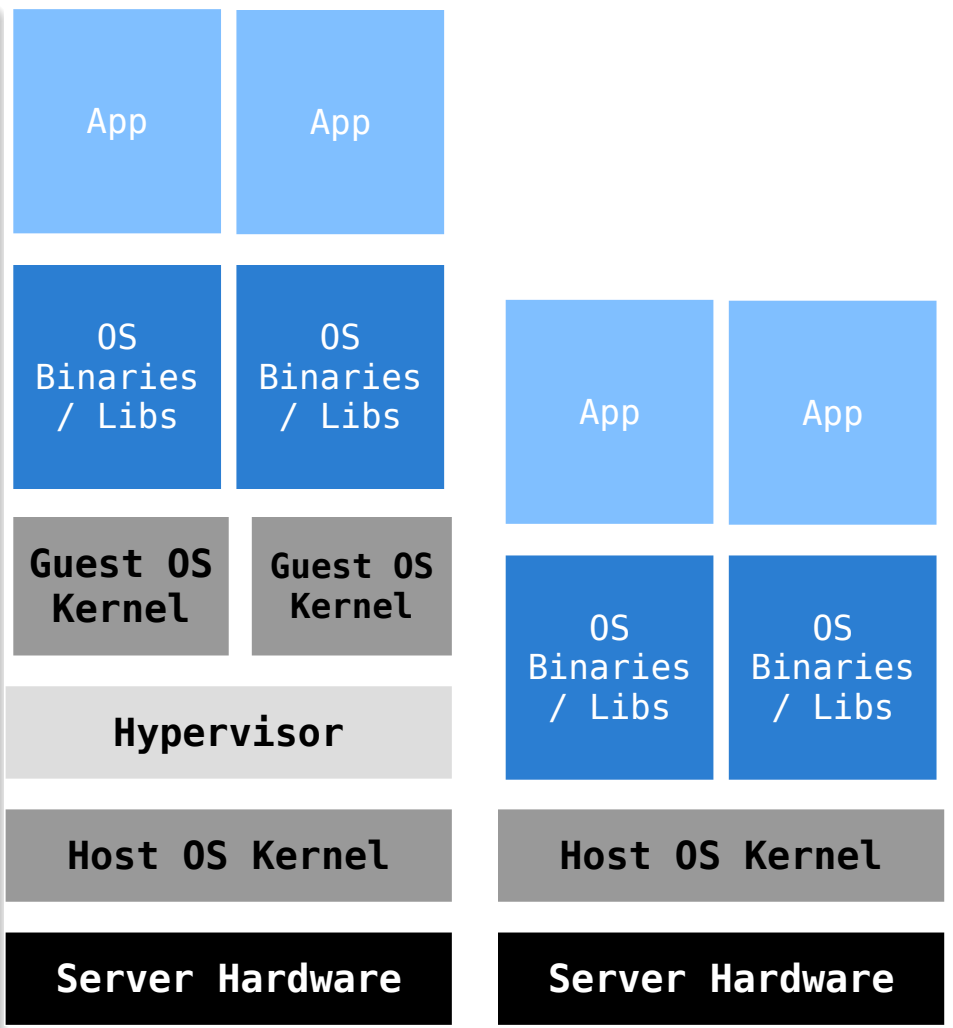
Table of contents

About Docker

Virtual Machines and Docker Containers

Container pros and cons:

- ☺ Smaller virtualization overhead for system calls, I/O, memory translation
- ☺ Better at overcommitting with idle services
- ☺ Boots faster (with caveats)
- ☺ Orchestration tools available
- ☹ Weaker isolation
- ☹ No "privileged operations", e.g. `mount`
- ☹ Linux only
- ☹ More moving parts



Linux Namespaces and cgroups

- Support in the Linux kernel
  - Namespaces mean isolation of resources.
  - Cgroups (control groups) capabilities and resources accounting.
- Available namespaces: 6
  - mnt - mount
  - net - network
  - ipc - interprocess communication
  - user - users and groups
  - pid - process id
  - uts - hostname
- Mount namespace implemented in 2002! (Linux 2.4.19)
- Similar approaches in other OSs:
  - 1980 - Unix/BSD chroot
  - 1992 - Plan9 namespaces
  - 2000 - BSD jail
  - 2004 - Solaris zones

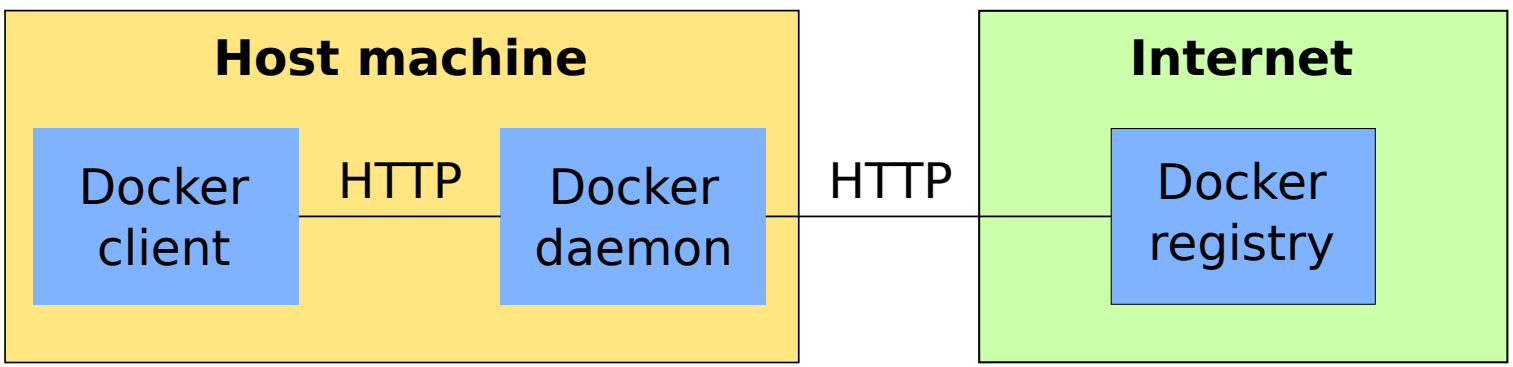
About Docker

Main components:

- Docker client
- Docker daemon
- Docker registry: hub.docker.com

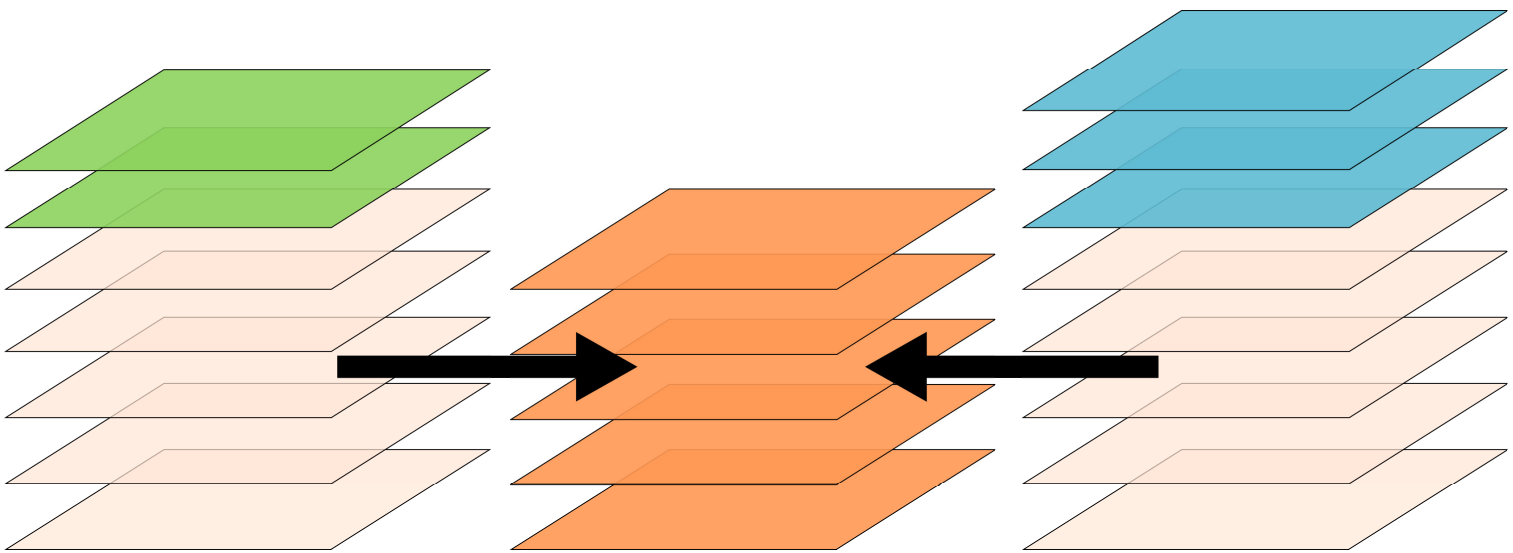
Docker versions:

- Jan: Docker 1.13.0
- Feb: Docker 1.13.1
- Mar: Docker 17.03



Docker layer, image and container

- Docker image consists of read-only layers described in image manifest.
- Layer is a *filesystem diff* between two snapshots.
- Layers are content addressable and reusable.
- Container is a Docker image in the state of execution.
- Each container has a dedicated read-write layer.



nginx ubuntu OwnCloud

Alternatives

Rocket
by: CoreOS

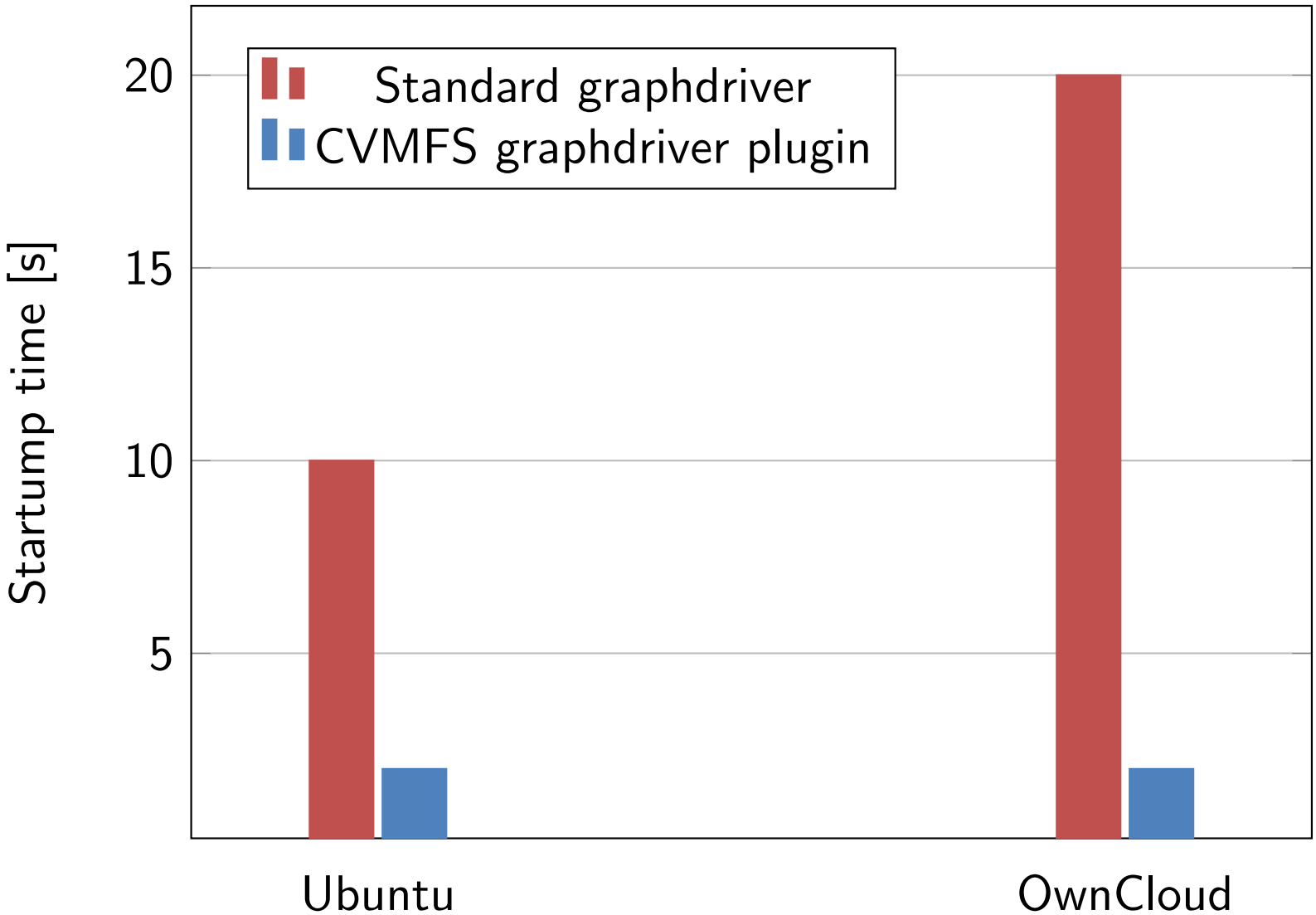
Singularity
by: LBL

Demo

What this demo presents

- Running container using regular Docker image.
- Running container using Docker and CVMFS.
- Making changes to container when using Docker and CVMFS.

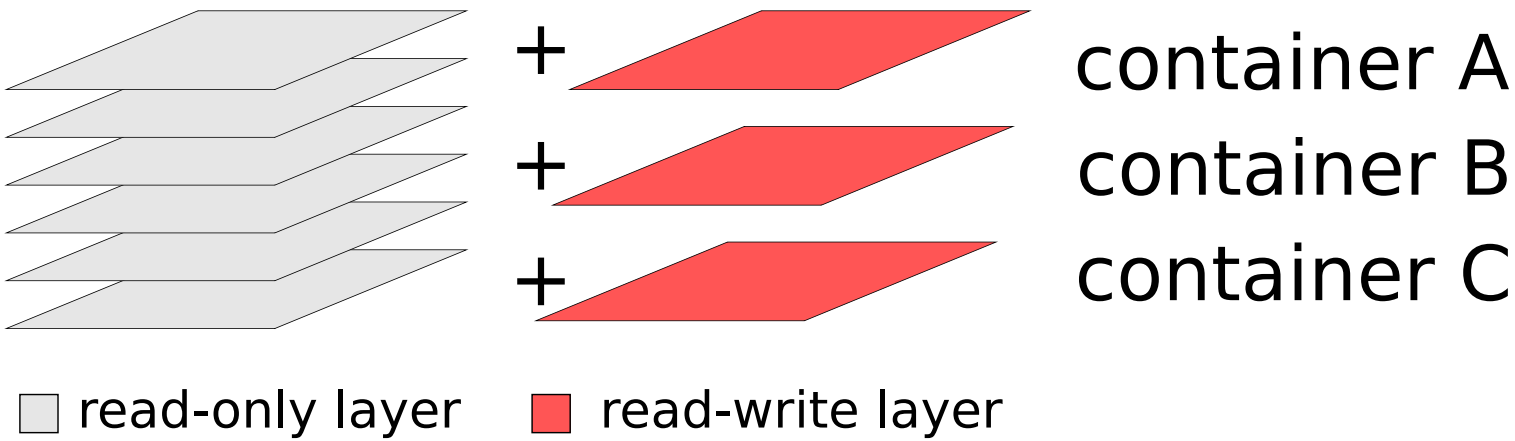
The results


Standard graphdriver
CVMFS graphdriver plugin
Startump time [s]
20
15
10
5
Ubuntu
OwnCloud

Docker Internals

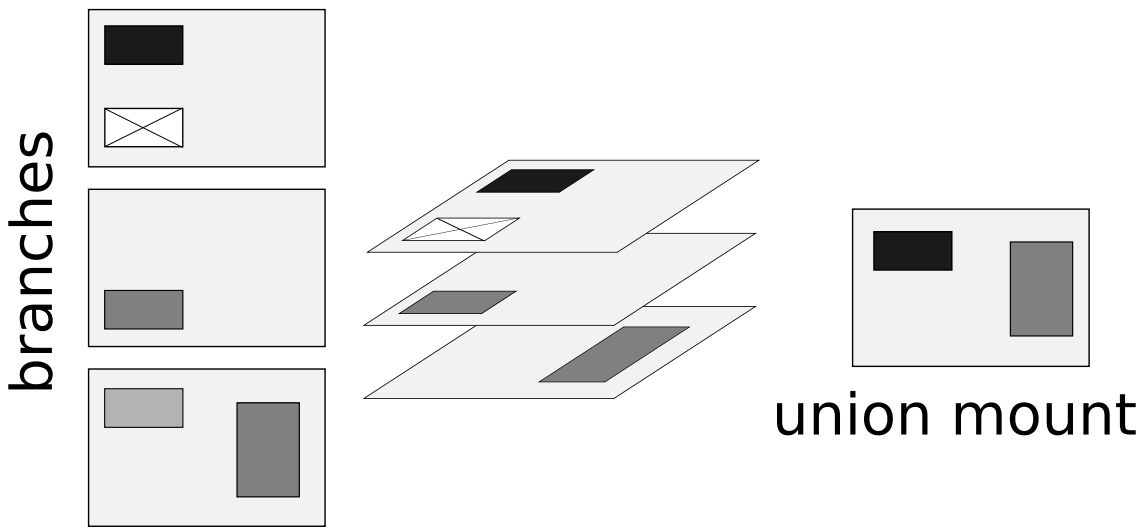
The Graphdriver

- Graphdriver combines layers and prepares the rootfs for container.
- Containers share read-only layers and get dedicated read-write layer.
- Graphdriver is component specific to Docker.
- Very limited interface.



UnionFS Drivers

- A branch is a regular directory.
- A union mount is a stack of branches.
- File appearing in multiple branches is read from the topmost one.
- A whiteout file hides (deletes) file from all branches under it.



- Two notable implementations are: aufs and overlayfs
- Overlayfs is newer, merged into kernel and is becoming more popular.

Workflow: Docker pull and Docker run

1. Docker client sends request to Docker daemon to pull new image.
2. Docker daemon downloads image manifest from Docker registry
3. Docker daemon downloads r/o layers (tarball) from Docker registry.
4. New layers are imported to graphdriver. (extracted)
5. On Docker run, graphdriver creates new r/w layer.
   This layer has r/o layers as parents.
6. All layers, the new dedicated r/w and r/o layers,
   are mounted together using an UnionFS driver to form rootfs.
7. Containerd and runc create new namespace for this container.
8. Docker daemon sends data to the client about fresh container and client "connects" to it.

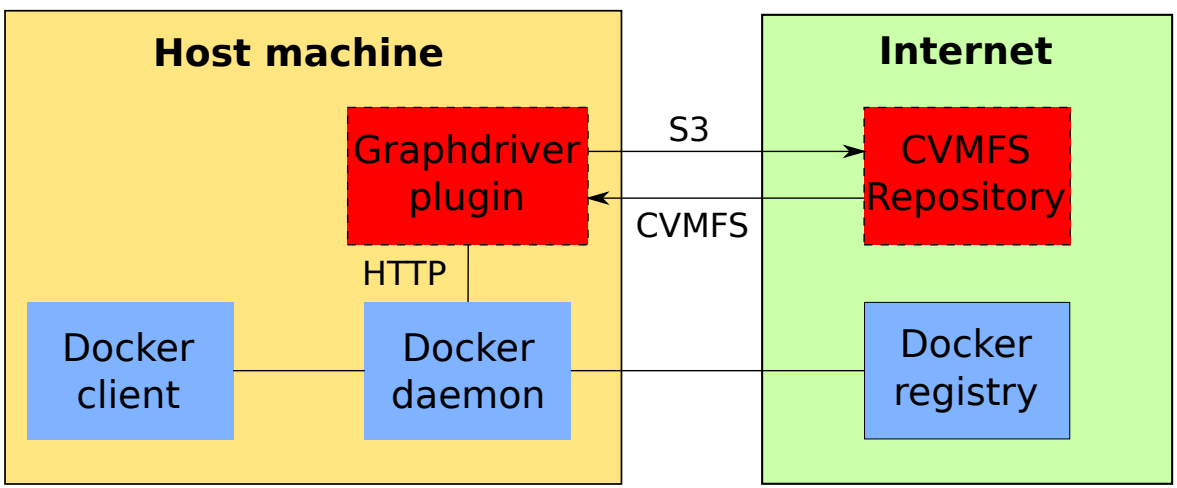
The CernVM-FS Graphdriver Plugin

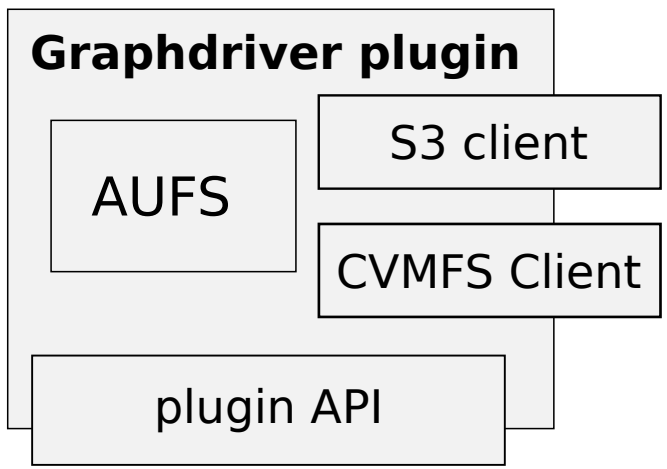
Docker Plugin System

- Daemon talks to the plugin process over unix socket or tcp port.
- Protocol is JSON RPC over HTTP.
- Two versions of plugins:
  - V1: separate process running on host machine
  - V2: plugin process running inside of a container (plugin container)
- Supported plugins:
  - network
  - volumes
  - graphdrivers! (since Docker 1.13, Jan. 2017)
- config.json + rootfs
- V2 plugins are managed (plugin install and push, shared via registry)
- `$ docker plugin install atlantic777/plugin`
  (WARNING: still just for preview!)

Structure of CernVM-FS Graphdriver Plugin

- Plugin container contains two main components:
  - Go binary implementing HTTP server.
  - CernVM-FS client.
- CernVM-FS driver is based on original Docker aufs graphdriver.
- Union mounts are created on a shared-mount location to be available both for plugin inside container and daemon on host.
- Plugin container is privileged (permissions granted on plugin install).



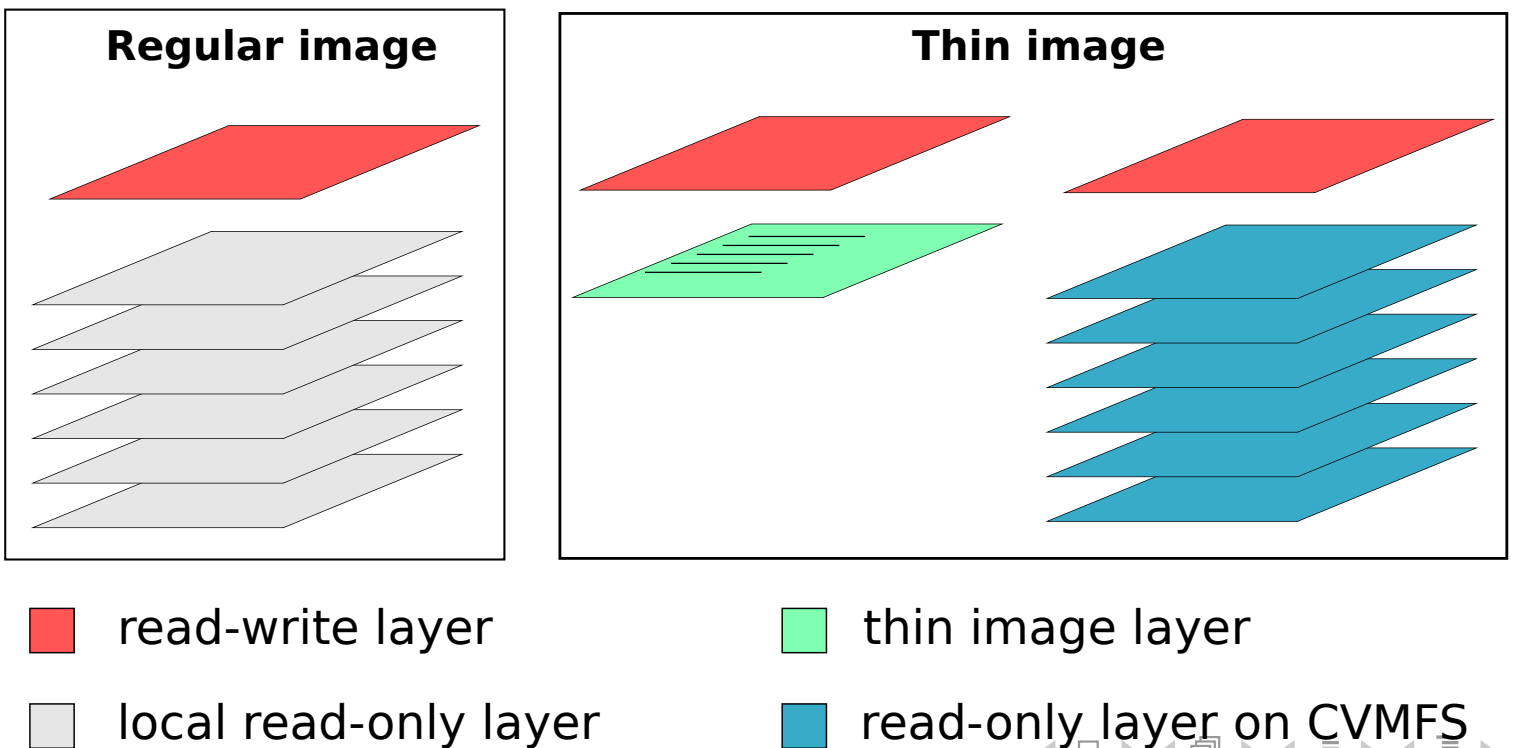


Using Docker with CernVM-FS

- Very limited interface:
  - Create() / CreateRW()
  - ApplyDiff() - import (extract) layer archive
  - Diff() - export (create) layer archive
  - Get() - create union mount
  - Put() - release union mount
- We introduce the concept of thin images.
- Like regular image but store metadata instead of content.
- Can be published on standard Docker registry.
- Thin images contain single json file, the image descriptor.
- Thin image descriptor contains list of r/o layers to be mounted from CernVM-FS repository.
- Container rootfs now consists of:
  - Dedicated r/w layer.
  - And r/o layers stored in CernVM-FS.

Workflow: Starting container from a thin image

- Pull `atlantic777/thin_ubuntu` from the Docker registry
- This image has one layer containing just the image descriptor.
- Request graphdriver to create r/w layer and union mount.
- Graphdriver creates union of the r/w layer and r/o layers from CVMFS
- List of needed r/o layers is read from the image descriptor.



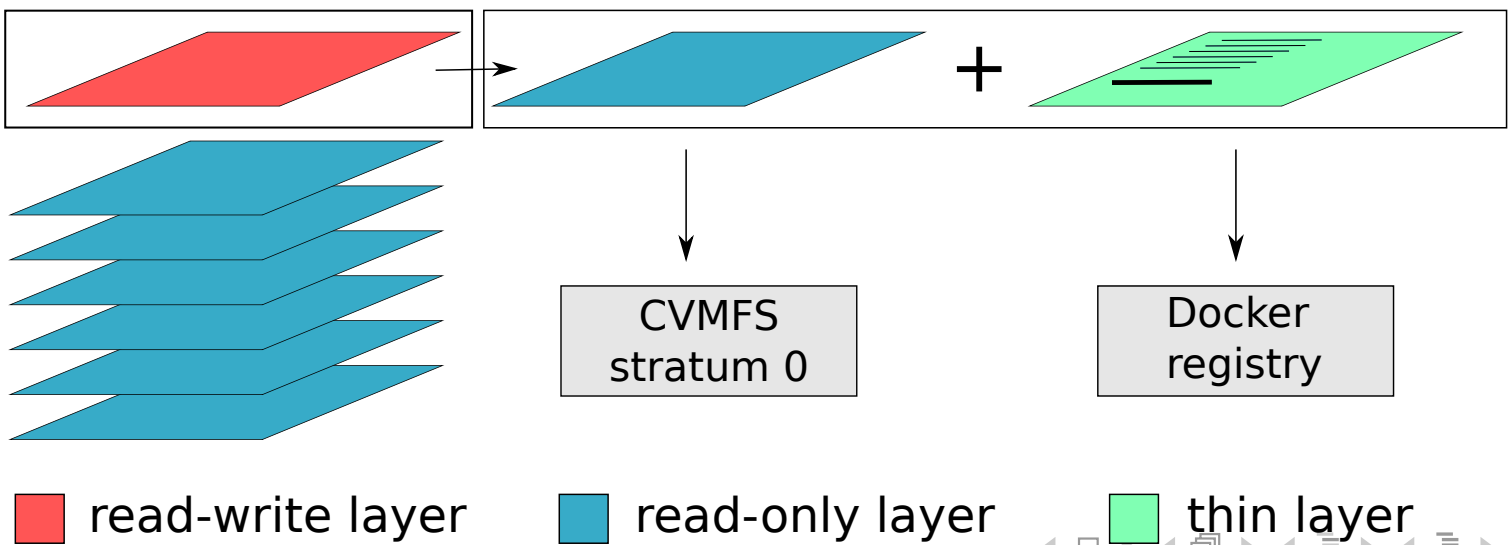
Saving changes using Docker and CernVM-FS

Regular image:

1. The changeset is stored in the r/w branch (r/w layer).
2. Publish this changes as new r/o layer on the Docker registry.
3. Create new Docker image with updated list or r/o layers.

Thin image:

1. The changeset is stored in the r/w branch (r/w layer).
2. Publish this changes on CVMFS repository as new r/o layer.
3. Create new Docker image with updated image descriptor.



Uploading new layer to CernVM-FS Stratum 0

Changes in graphdriver plugin Diff() method:

- Create archive containing changed files.
- Compress it and calculate hash.
- Upload this new layer archive to stratum 0 server over S3 API with hash as key.
- Create updated image descriptor.
- Return archive with image descriptor.

Stratum 0 publisher agent:

- S3 server sends notification about new object.
- Publisher agent receives notification.
- Publisher agent will:
  - start CVMFS transaction,
  - extract this new layer and
  - publish transaction.

Status and Roadmap

Status and Roadmap

- Conventions for storing CVMFS cache and configuration in plugin container need to be defined.
- When updating thin images, new layers are uploaded and published to CVMFS repository on commit while for regular images, commits are local (like in git) and changes are uploaded on only on Docker push.
- Handling of special files in CernVM-FS should be improved (chardev, blockdev, whiteout)
- Add support for overlayfs in the same way aufs support was added.
- Tools for converting regular images to thin images are needs to be finished.

Summary

Summary

- CernVM-FS graphdriver plugin can be used for both running and modifying container images.
- All requirements are available in current Docker releases.
- Plugin can be installed from Docker registry.
- All existing Docker images can be converted in simple and automated way.
- Preserves all Docker features.

Thank you for your attention!

Questions?

CernVM-FS Graphdriver Plugin for Docker

CERN, SFT Group Meeting

Nikola Hardi
nhardi@cern.ch

March 20, 2017

Supervisors:
Jakob Blomer : jblomer@cern.ch
Gerardo Ganis: gerardo.ganis@cern.ch