VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

HPC CONTAINERS

PARAVIRTUALIZATION TECHNOLOGY

Josef Hrabal

josef.hrabal@vsb.cz

HPC CONTAINER?

HPC CONTAINER?

- Container - An object for holding or transporting something

HPC CONTAINER?

- Container - Encapsulation of system environments

- Container - A software package that contains everything the software needs to run.

- Container - OS-level virtualization method used to deploy and run distributed applications without launching an entire virtual machine (VM) for each app.

CONTAINER?

CPIU 185436
12G1
MAX.GR.
TARE
10.160 KGS
22.400 LBS
1.300 KGS
2.870 LBS
NET
CU.CAP.
8.860 KGS
19.530 LBS
15.9 CU.M.
561 CU.FT.

CONTAINER?

CONTAINER?

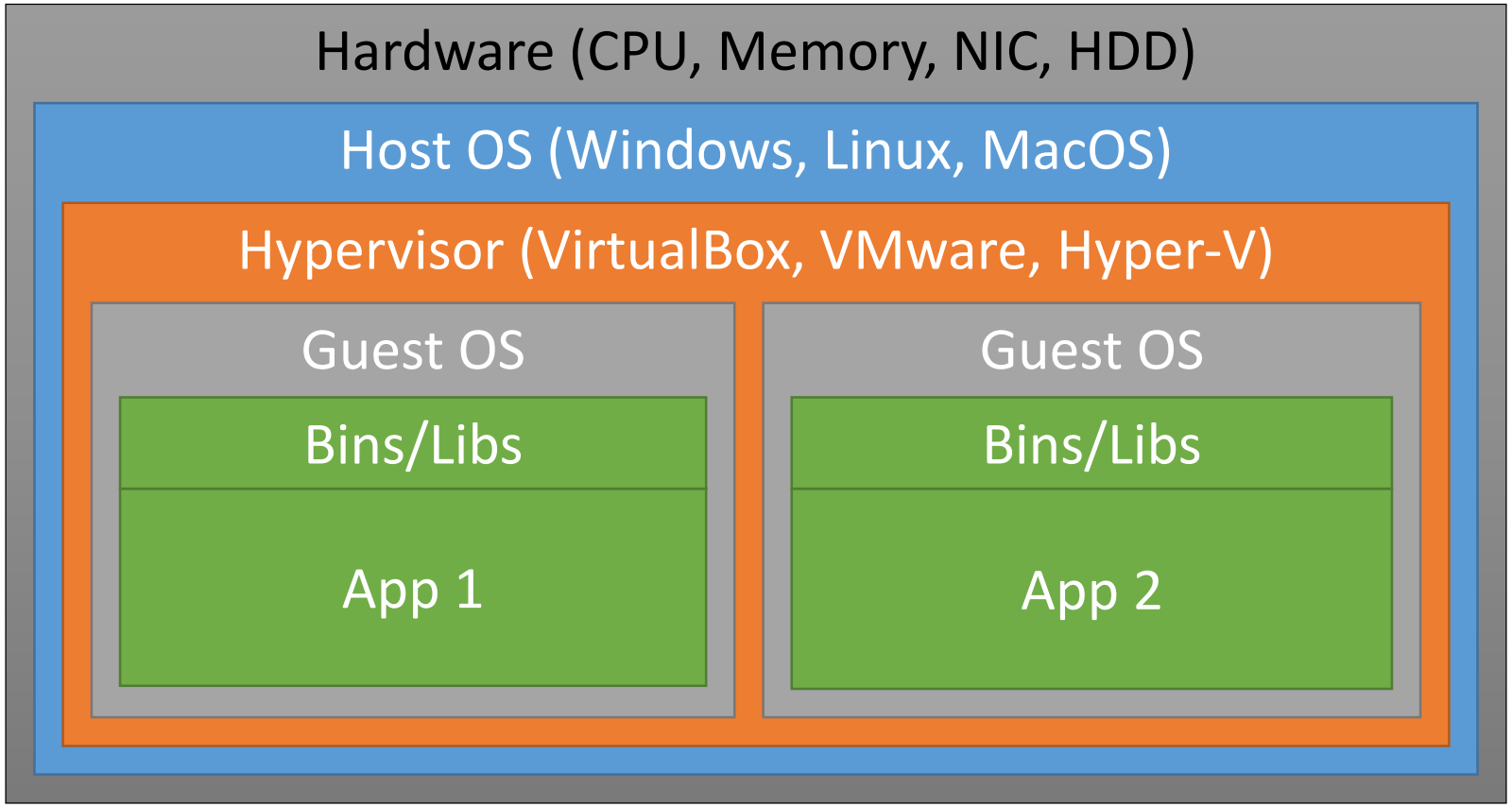
HPC CONTAINER?

- Container - Encapsulation of system environments

- Container - A software package that contains everything the software needs to run.

- Container - OS-level virtualization method used to deploy and run distributed applications without launching an entire **virtual machine (VM)** for each app.

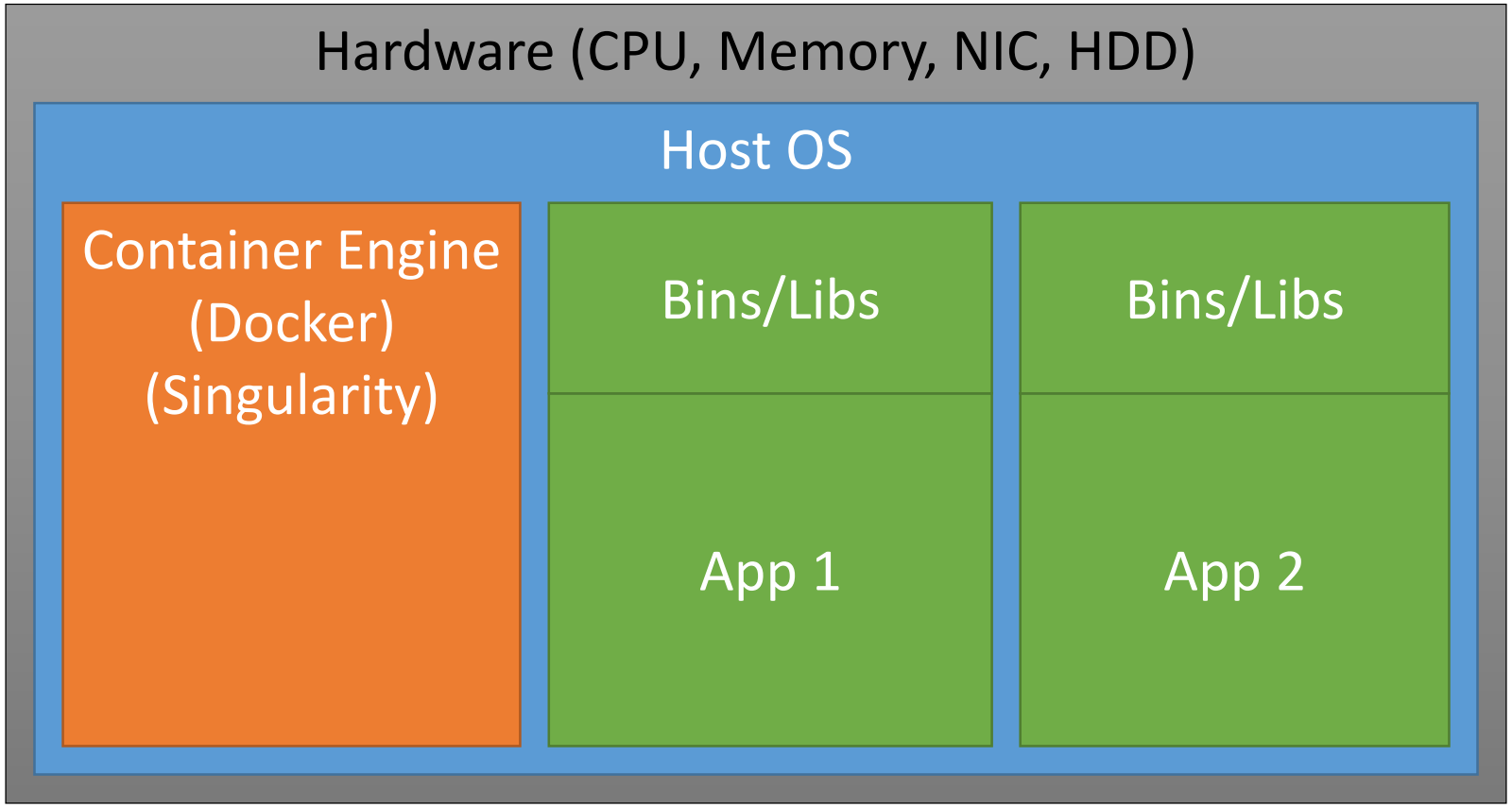
VIRTUAL MACHINE



VIRTUAL MACHINE

- Pros:
  - Complete isolation between virtual machines
  - Compatibility - same “virtual HW” on all physical machines
  - Portability – run on any host OS
  - Aggregation of resources
  - Snapshots
- Cons:
  - No direct resource sharing
  - Overhead, Boot time
  - Large VM images – contain whole OS
  - Difficult to run on a cluster (kernel modules etc.)

CONTAINER


Hardware (CPU, Memory, NIC, HDD)
Host OS
Container Engine
(Docker)
(Singularity)
Bins/Libs
App 1
Bins/Libs
App 2

CONTAINER

- Encapsulation of system environments
- Pros:
  - Speed - start, create, replicate or destroy quickly
  - Portability – run on any Linux based OS
  - Allows access to host filesystems
  - Image contain only necessary files
  - Aggregation of resources
  - Small overhead
- Cons:
  - Shares kernel of host OS
  - Weaker isolation?

CONTAINER

- 1979 – chroot (changing root directory)
- 2000 – FreeBSD Jails (early container technology)
- 2001 – Linux VServer (idea of namespace separation)
- 2004 – Solaris Containers (Solaris 10)
- 2005 – OpenVZ (patched Linux kernel)
- 2006 – Process Containers (limiting and isolating resources)
- 2007 – Control Groups (renamed Process Containers)
- 2008 – LXC - LinuX Containers (using cgroups and namespaces)
- 2011 – Warden (work on any OS)
- 2013 – LMCRFY (open source Google's container)
- 2013 – Docker (opensourced)
- 2014 – Rocket (similar to Docker by CoreOS)
- 2016 – Windows Containers (MS Windows Server 2016)

CONTAINERS IN INDUSTRY

- Micro-service virtualization
- Aggregation of resources

- Docker
  - Most widely used **containerization tool**
  - Designed for enterprise needs
  - Image-based deployment model
  - Perfect for local resources
  - Root privileges needed

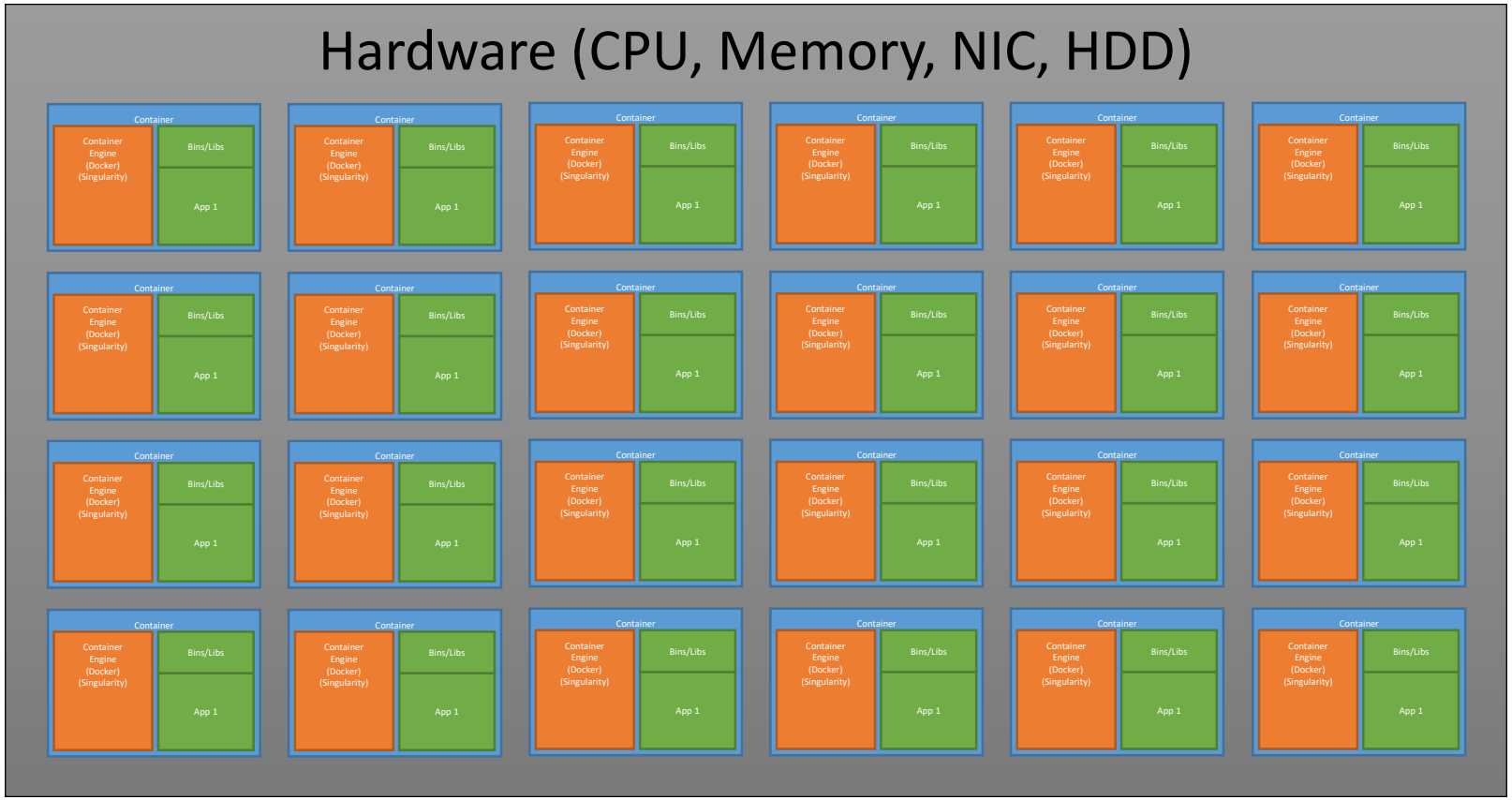
CONTAINERIZATION TOOLS?

CONTAINERS IN INDUSTRY

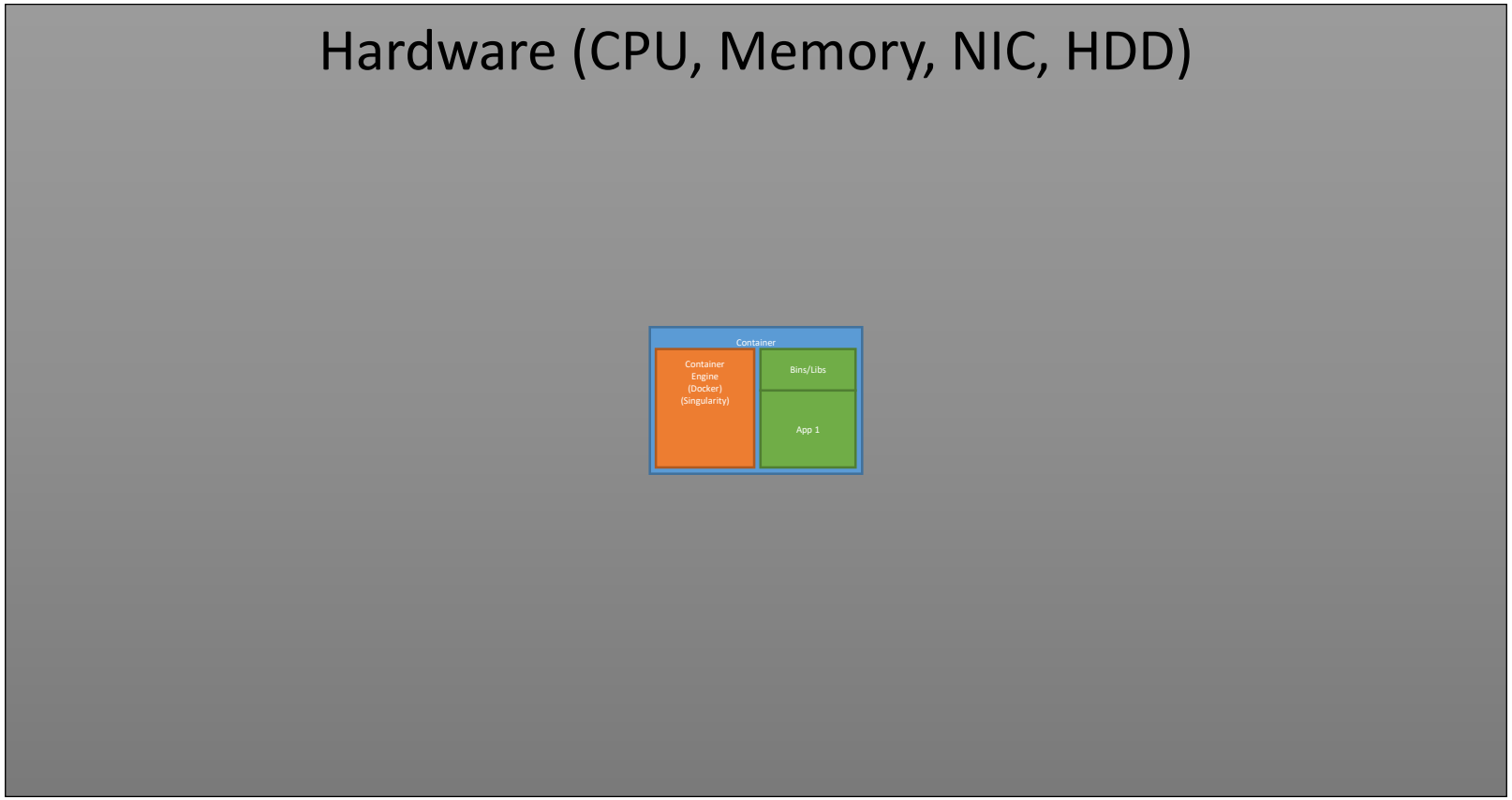
- Micro-service virtualization
- Aggregation of resources

- Docker
  - Most widely used **containerization tool**
  - Designed for enterprise needs
  - Image-based deployment model
  - Perfect for local resources
  - Root privileges needed

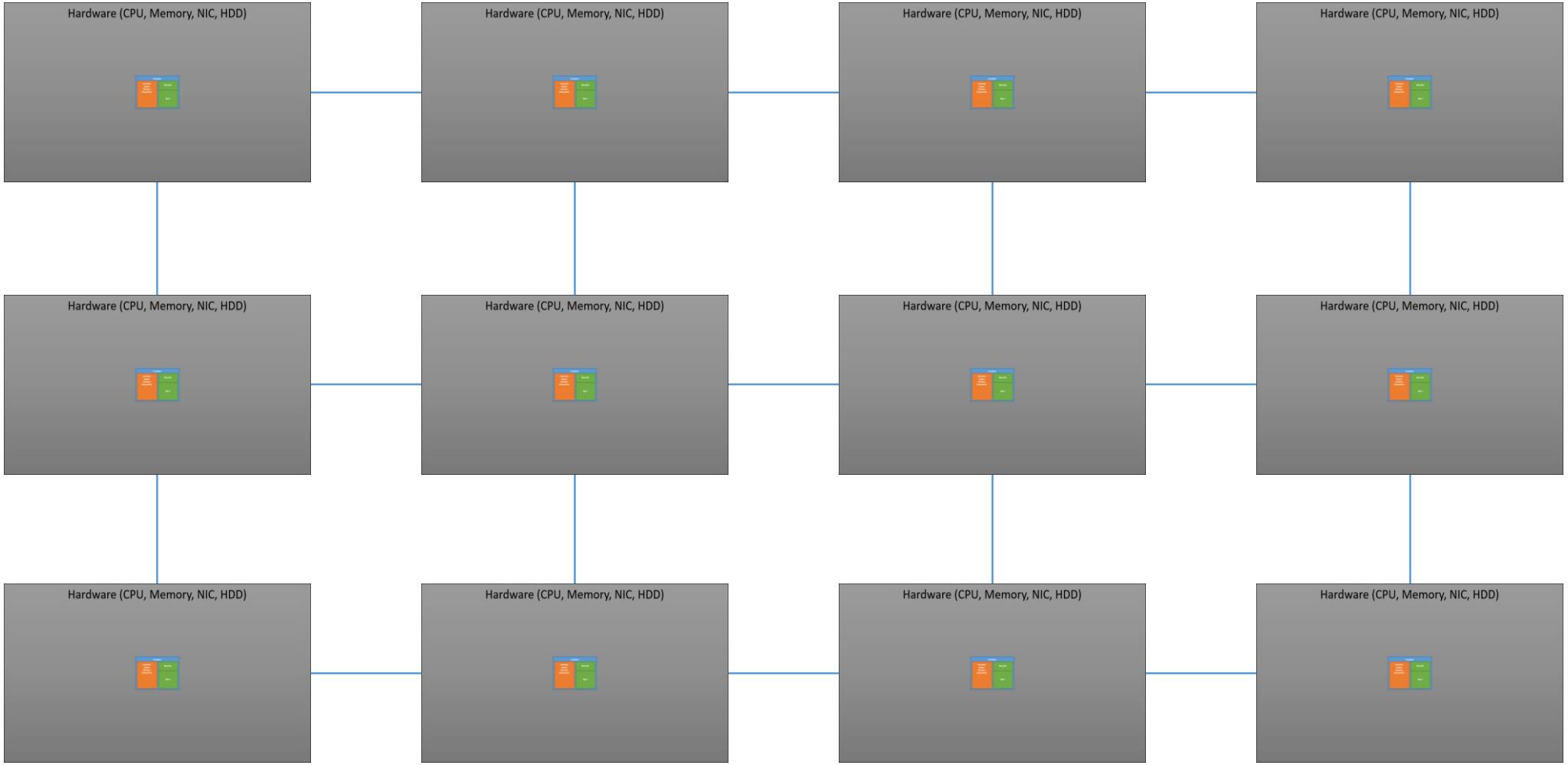
CONTAINERS IN INDUSTRY



CONTAINERS IN HPC



CONTAINERS IN HPC



DOCKER IN HPC?

- So can we run docker on HPC?

- Is it a good idea?

DOCKER IN HPC?

SINGULARITY TO THE RESCUE!

CONTAINERS IN HPC

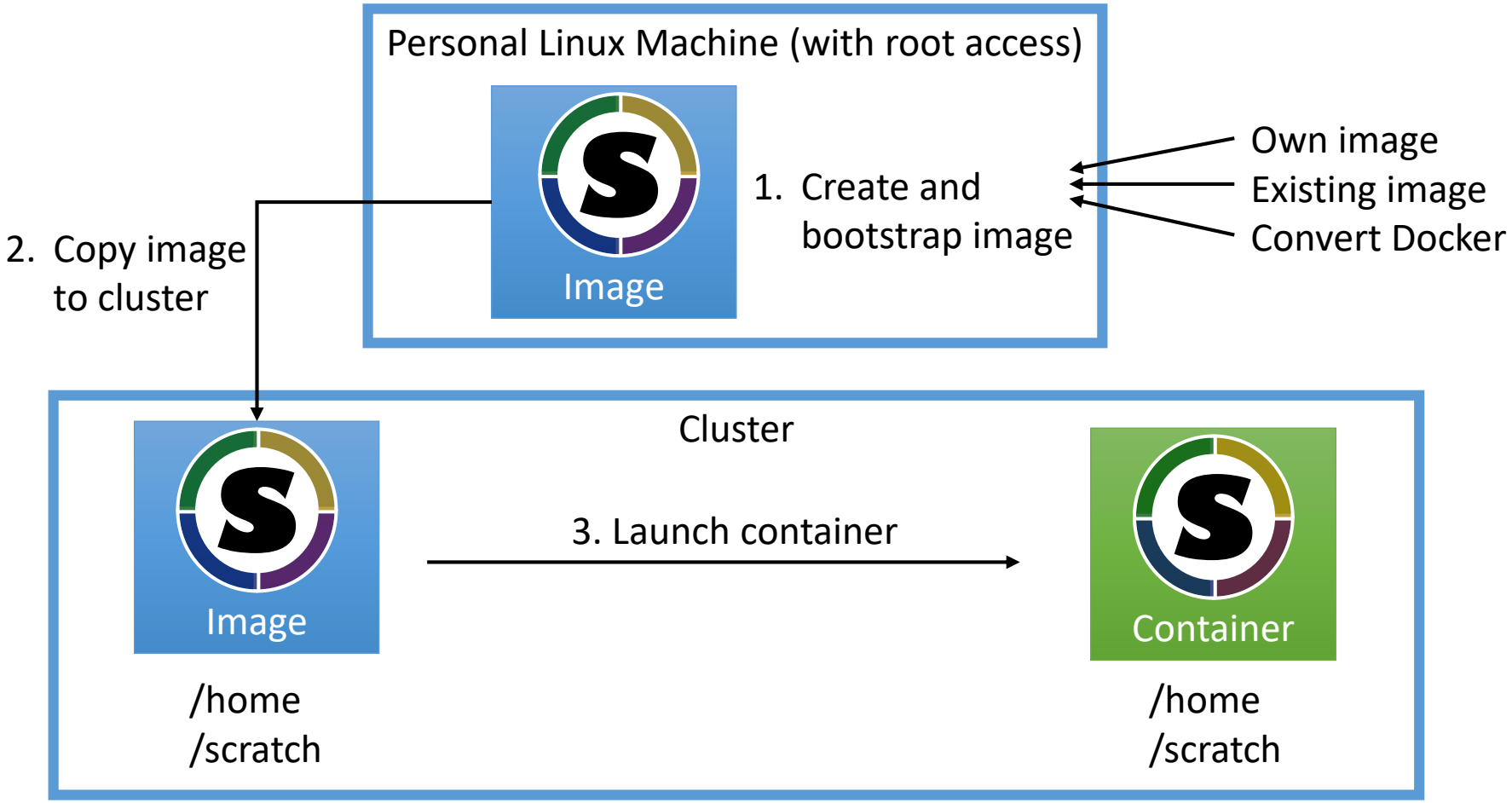
- Singularity:
  - Designed with HPC in mind
  - Directly Supports MPI!
  - Integration with schedulers, GPU, InfiniBand…
  - Permissions inside a container are the same as those outside of the container
  - User can access his files stored outside of the container
  - User can WRITE files outside of the container
  - No daemon with root privileges needed
  - More at https://sylabs.io

SINGULARITY

- Image:
  - Virtual filesystem
  - Software and OS are installed on this virtual filesystem
  - Built single time on a machine with root access
  - Used multiple times on a cluster
  - Immutable by default

- Container:
  - A running instance of an Image

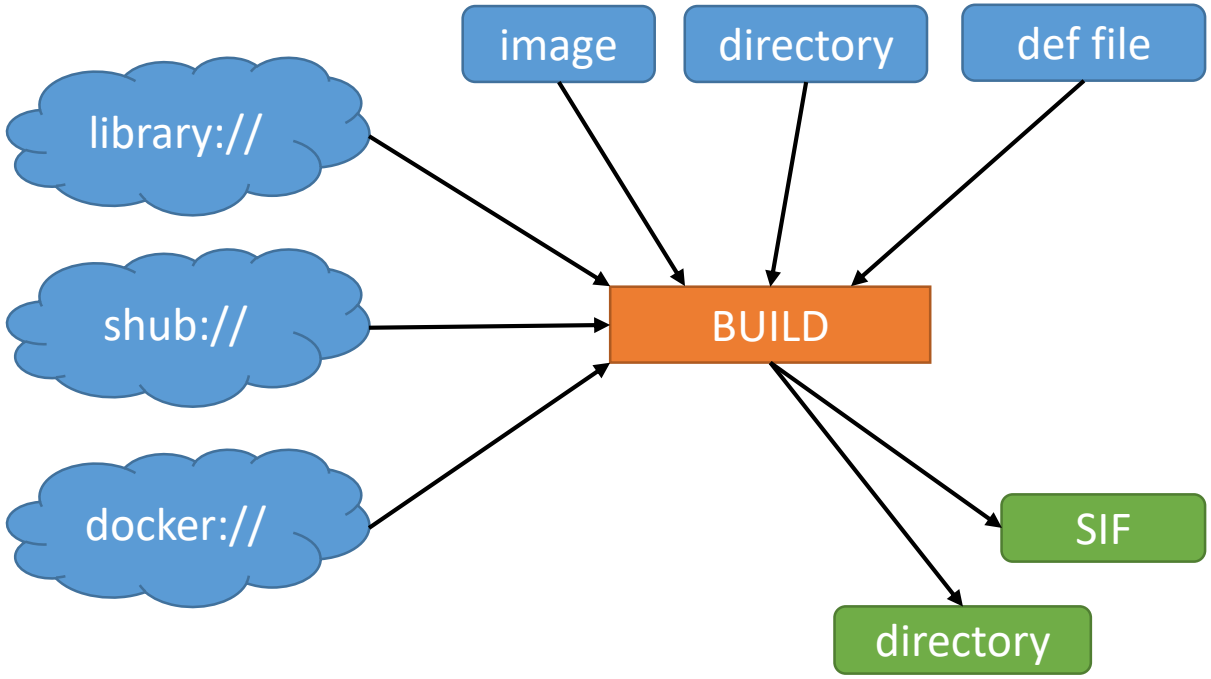
SINGULARITY WORKFLOW



CREATING SINGULARITY IMAGE

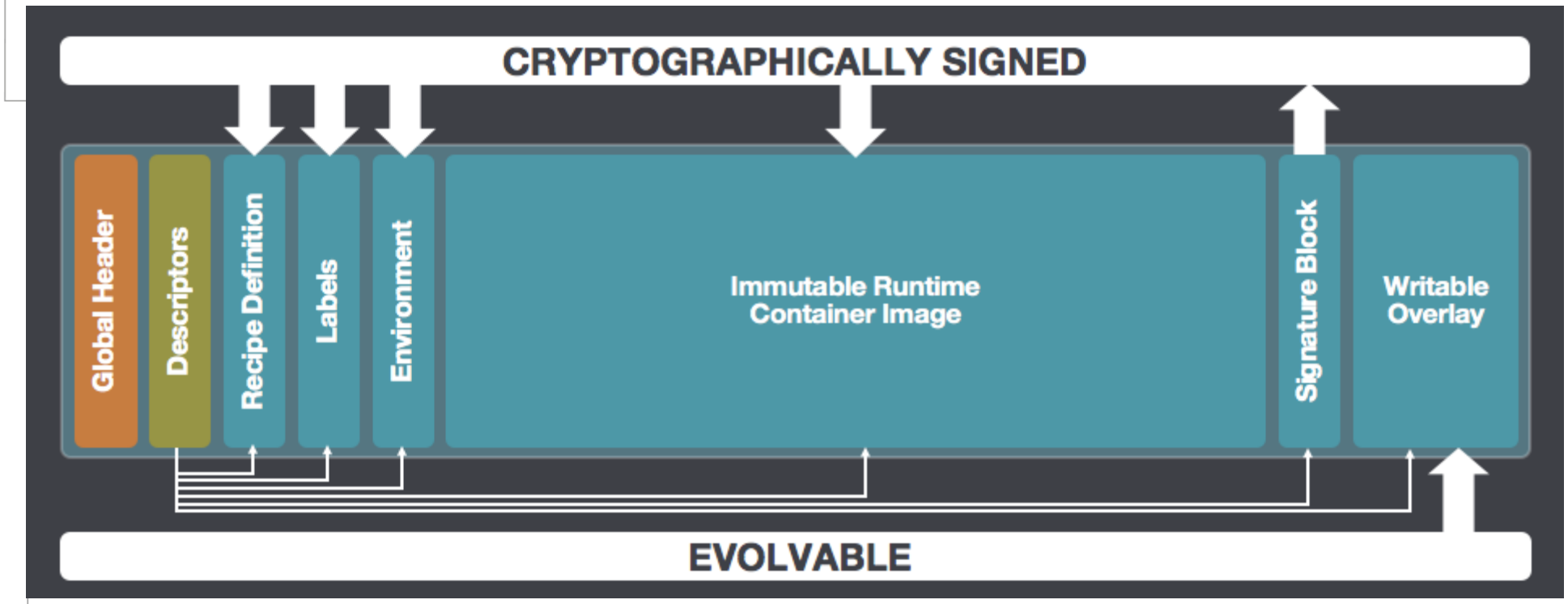
- Types of Image:
  - compressed read-only **Singularity Image File (SIF)** format suitable for production (default)
  - writable **(ch)root directory** called a sandbox for interactive development (**--sandbox** option)
  - overlay image (directory/ext3)

- Image can be created from:
  - library://
  - shub://
  - docker://
  - existing image
  - directory
  - definition file

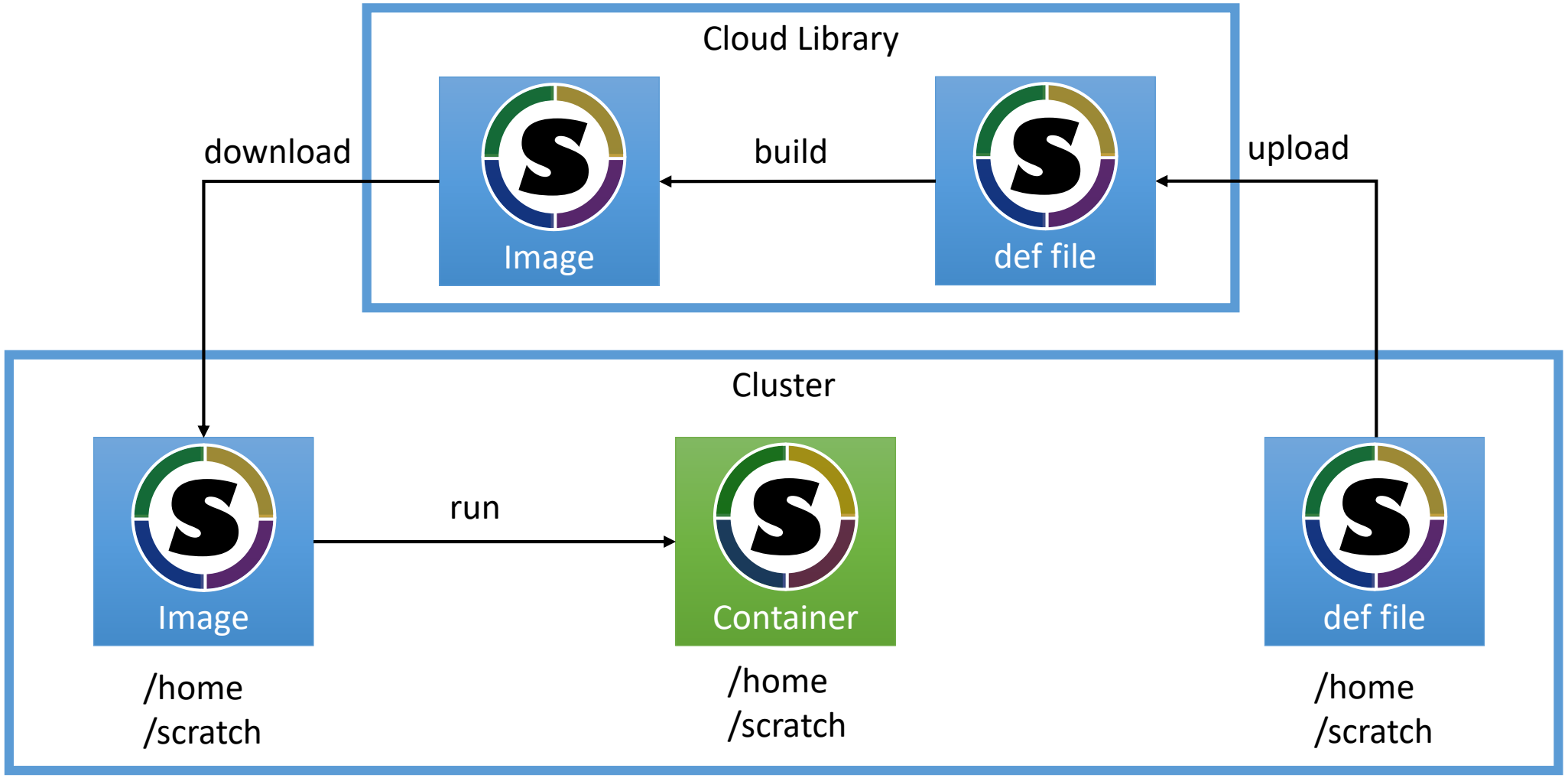


SINGULARITY IMAGE FORMAT (SIF)

- Immutable
- Cryptographically signed
- Encrypted
- No layers



CLOUD LIBRARY



CREATING SINGULARITY IMAGE

- # singularity build ubuntu.sif ubuntu.def
- $ singularity build **--remote** ubuntu.sif ubuntu.def
  - Use recipe in def file to install OS and SW

- $ singularity build ubuntu.sif docker://ubuntu:latest
- $ singularity build ubuntu.sif library://ubuntu:latest
  - Create image from Docker/Singularity Hub

- $ singularity build ubuntu.sif /tmp/ubuntu
  - Create image from chroot directory (sandbox)

SINGULARITY DEFINITION FILE

- Header:
  - Define base OS distribution to use (Bootstrap:)
  - As base can be used:
    - library – images hosted on the Container Library
    - shub – images hosted on Singularity Hub
    - docker – images hosted on Docker Hub
    - oras - images from supporting OCI registries e.g. ACR (Azure Container Registry)
    - scratch - a flexible option for building a container from scratch
    - localimage, yum, oci, oci-archive, docker-daemon, docker-archive, arch, busybox zipper
  - Other parameters are dependent on the base used
    - OSVersion
    - MirrorURL
    - Include
    - From
    - …

SINGULARITY DEFINITION FILE

- %setup:
  - Runs commands outside of the container at start of the bootstrap process
  - Runs before %post section
- %post:
  - Runs once inside the container during bootstrap process
  - Software installation…
- %files:
  - Copy files from outside of the image to the inside of it
  - Pairs of <source> <destination>
  - Runs after %post section

SINGULARITY DEFINITION FILE

- %environment:
  - Define environment variables inside container
- %runscript:
  - Define custom runscript
  - Command line parameter parsing etc…
  - $ singularity run <image>
- %test:
  - Test the proper function of the container
  - Runs at the end of the bootstrapping process
  - Disable by --notest option

SINGULARITY DEFINITION FILE

- %labels:
  - Define custom labels/metadata
  - <label_name>=<label_value>
  - $ singularity inspect <image>

- %help:
  - Add help for the image
  - $ singularity help ubuntu.sif

RUNNING SINGULARITY IMAGE

- $ singularity **shell** <image>
  - Start a container and invoke interactive shell
- $ singularity **run** <image>
  - Start a container and exec runscript inside container
- $ singularity **exec** <image> <command>
  - Start a container and exec command inside container
- $ singularity **run –app** <app_name> <image>
  - Start a container and exec apprun script inside container

RUNNING AN INSTANCE

- $ singularity **instance start** <image> <instance_name>
  - Start an instance of container in background
  - Useful for running services
- $ singularity **instance list**
  - List of started instances
- $ singularity **instance stop** <instance_name>
  - Stop running instance
- $ singularity **run instance**://<instance_name>
- $ singularity **shell instance**://<instance_name>

SCI-F APPS

- Standard Container Integration Format
- Provides internal modularity of containers
- One container can contain multiple applications with different environment
- Apps have own sections in bootstrap file:
  - %appinstall
  - %apphelp
  - %apprun
  - %applabels
  - %appenv
  - %apptest
  - %appfiles

SIGNING AND VERIFYING CONTAINERS

- $ singularity pull library://alpine

  INFO: Downloading library image

  WARNING: Container might not be trusted; run 'singularity verify alpine_latest.sif' to show who signed it

  INFO: Download complete: alpine_latest.sif

- $ singularity verify alpine_latest.sif

  Container is signed by 1 key(s):

  Verifying signature F: 8883491F4268F173C6E5DC49EDECE4F3F38D871E:

  [REMOTE] Sylabs Admin <support@sylabs.io>

  [OK] Data integrity verified

  INFO: Container verified: alpine_latest.sif

SIGNING AND VERIFYING CONTAINERS

- $ singularity key newpair

  Enter your name (e.g., John Doe) : Josef Hrabal

  Enter your email address (e.g., john.doe@example.com) : hrabal.josef@gmail.com

  Enter optional comment (e.g., development keys) : salomon

  Enter a passphrase :

  Retype your passphrase :

  Generating Entity and OpenPGP Key Pair... Done

- $ singularity sign my_container.sif

  Signing image: my_container.sif
  Enter key passphrase:
  Signature created and applied to my_container.sif

SIGNING AND VERIFYING CONTAINERS

- $ singularity key list

  Public key listing (/home/hra0031/.singularity/sypgp/pgp-public):

  0) U: Josef Hrabal (salomon) <josef.hrabal@vsb.cz>

  C: 2019-10-15 11:15:46 +0200 CEST

  F: 54EAE720EAB9930D1ABE890190920BE5BAB43191

  L: 4096

- $ singularity key push 54EAE720EAB9930D1ABE8…

  public key ` 54EAE720EAB9930D1ABE8…` pushed to server successfully

- $ singularity key search Hrabal

  Showing 1 results

  KEY ID BITS NAME/EMAIL

  BAB43191 4096 Josef Hrabal (salomon) <josef.hrabal@vsb.cz>

ENCRYPTING SINGULARITY IMAGE

- # singularity build --passphrase encrypted.sif encrypted.def

  Enter encryption passphrase: <secret>

  INFO: Starting build...

- # export SINGULARITY_ENCRYPTION_PASSPHRASE="password"
- # singularity build --encrypt encrypted.sif encrypted.def

  INFO: Starting build...

- # singularity build --pem-path=rsa_pub.pem enc.sif enc.def

  INFO: Starting build...

RUNNING ENCRYPTED SINGULARITY IMAGE

- $ singularity run --passphrase encrypted.sif

  Enter passphrase for encrypted container: <secret>

- $ export SINGULARITY_ENCRYPTION_PASSPHRASE=“password”
- $ singularity run encrypted.sif

- $ singularity run --pem-path=rsa_pri.pem encrypted.sif

PERSISTENT OVERLAYS

- Illusion of read-write access
- Directory or ext3

- # singularity shell --overlay my_overlay/ ubuntu.sif
  - Root privileges needed

- Make ext3 overlay
  - $ dd if=/dev/zero of=my_overlay.ext3 bs=1M count=500
  - $ mkfs.ext3 my_overlay.ext3

- $ singularity shell --overlay my_overlay.ext3 ubuntu.sif

SINGULARITY ON IT4I CLUSTERS

- Version 3.4.1 installed on both clusters
- Prepared images with:
  - CentOS 6.10
  - CentOS 7.7
  - CentOS 8.0
  - Fedora 29
  - Debian 8.0
  - Ubuntu 16.04
- Prepared images are available as modules
- Full OpenMPI support inside container
- Full Lmod support inside container
- Singularity wrappers for better user experience

SINGULARITY WRAPPERS (WIP)

- $ image-shell
  - Invoke interactive shell inside loaded singularity container module
- $ image-run
  - Exec runscript inside loaded singularity container module
- $ image-exec <command>
  - Exec command inside loaded singularity container module
- $ image-mpi <command>
  - Exec mpirun inside loaded singularity container module
- $ image-update
  - Update local copy of image from /apps

WHY SINGULARITY?

- Different or newer version of library is needed (e.g. glibc)
- Isolate work from environment and software available on the cluster
- Standardize workflow on different HPC systems
- Prefer a different Linux distribution than is on the cluster

  („I hate CentOS, Gentoo rulez!“)

FUTURE?

Josef Hrabal
josef.hrabal@vsb.cz

IT4Innovations National Supercomputing Center
VSB – Technical University of Ostrava
Studentská 6231/1B
708 00 Ostrava-Poruba, Czech Republic
www.it4i.cz

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER