Docker for HPC?
Yes, Singularity!

Josef Hrabal

IT4Innovations

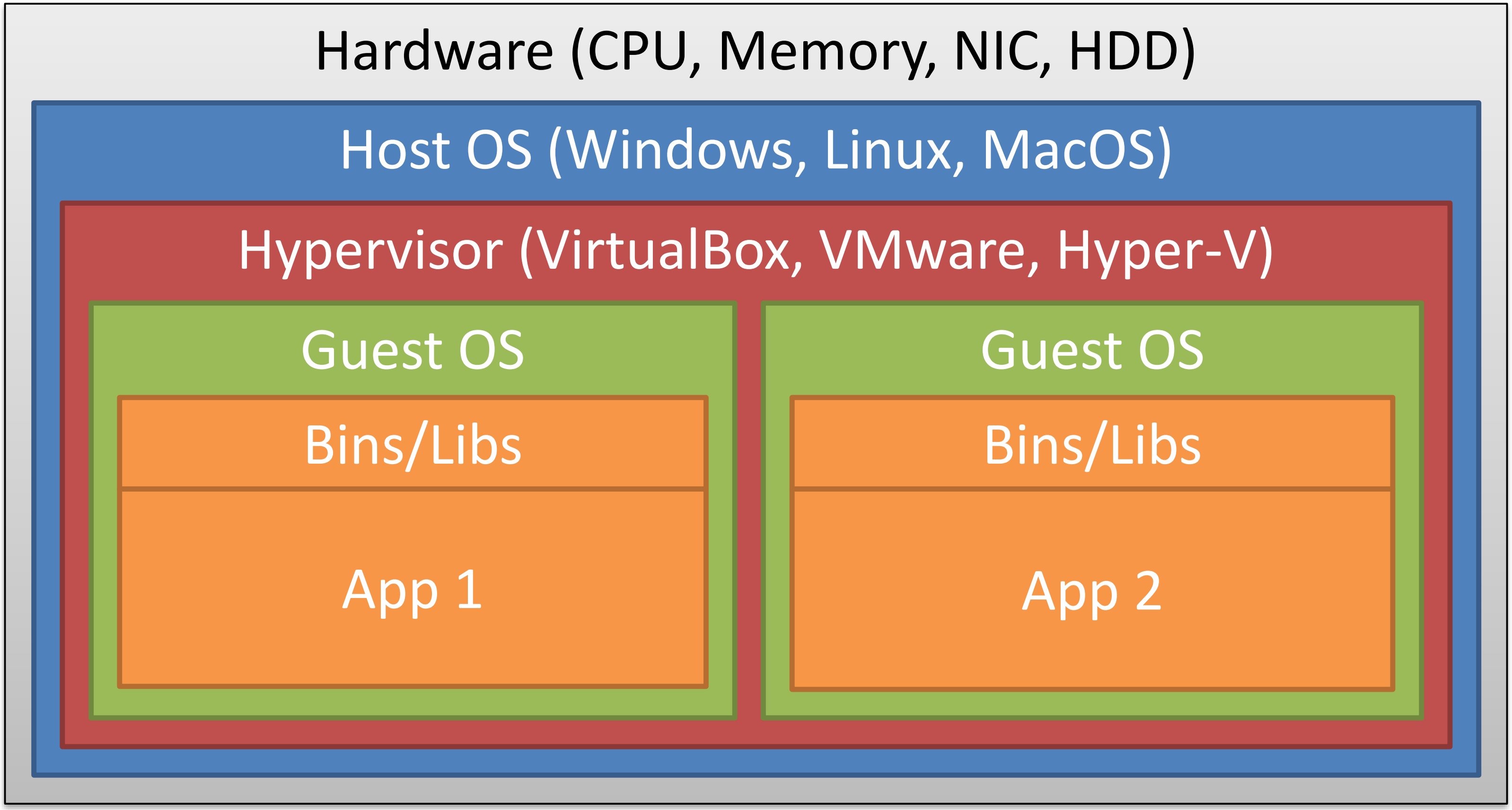
IT4Innovations
národní
superpočítačové
centrum

josef.hrabal@vsb.cz

support@it4i.cz

LINUX DAYS

Virtual Machine



Hardware (CPU, Memory, NIC, HDD)

Host OS (Windows, Linux, MacOS)

Hypervisor (VirtualBox, VMware, Hyper-V)

Guest OS

Bins/Libs

App 1

Guest OS

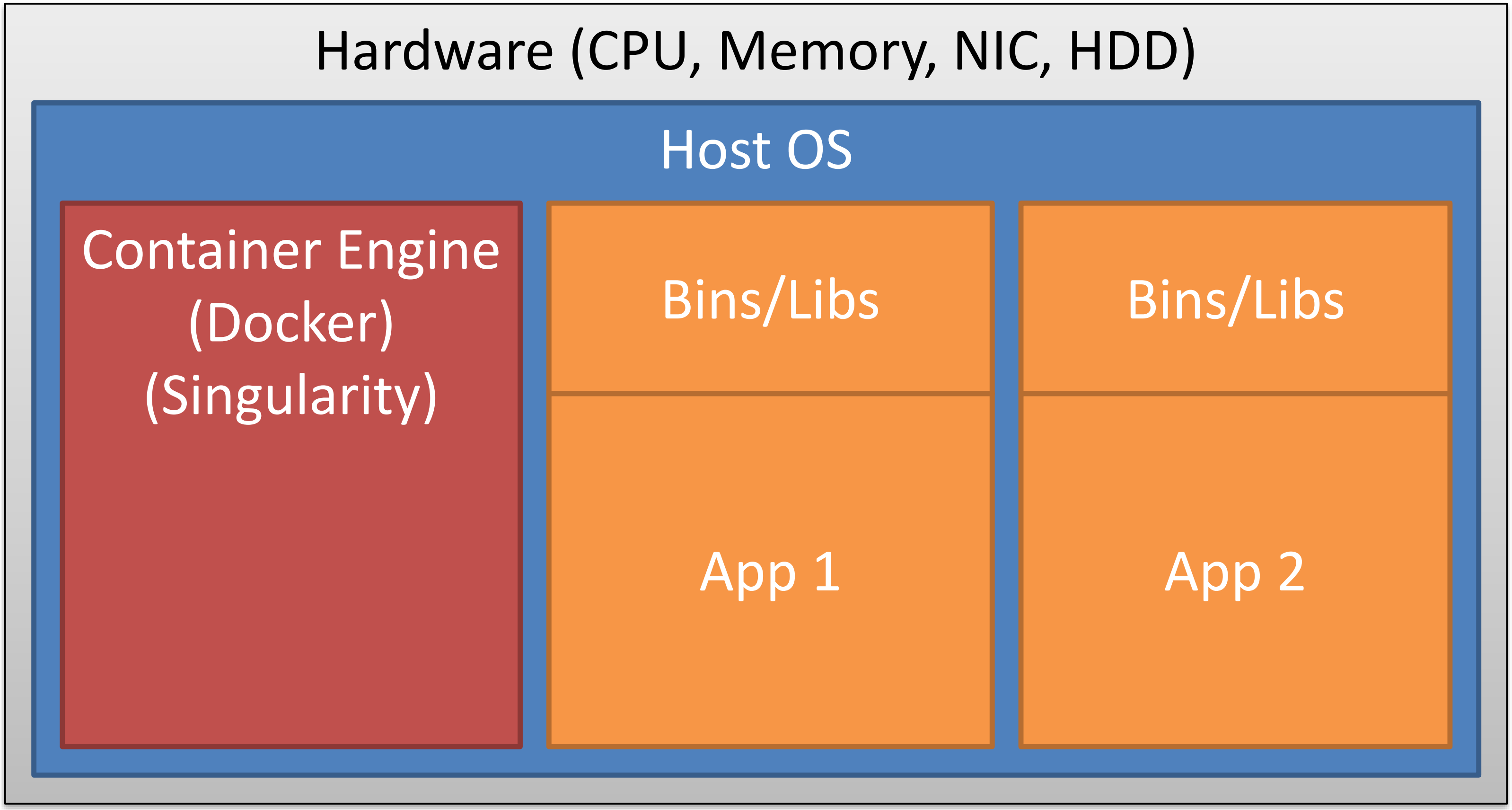
Bins/Libs

App 2

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

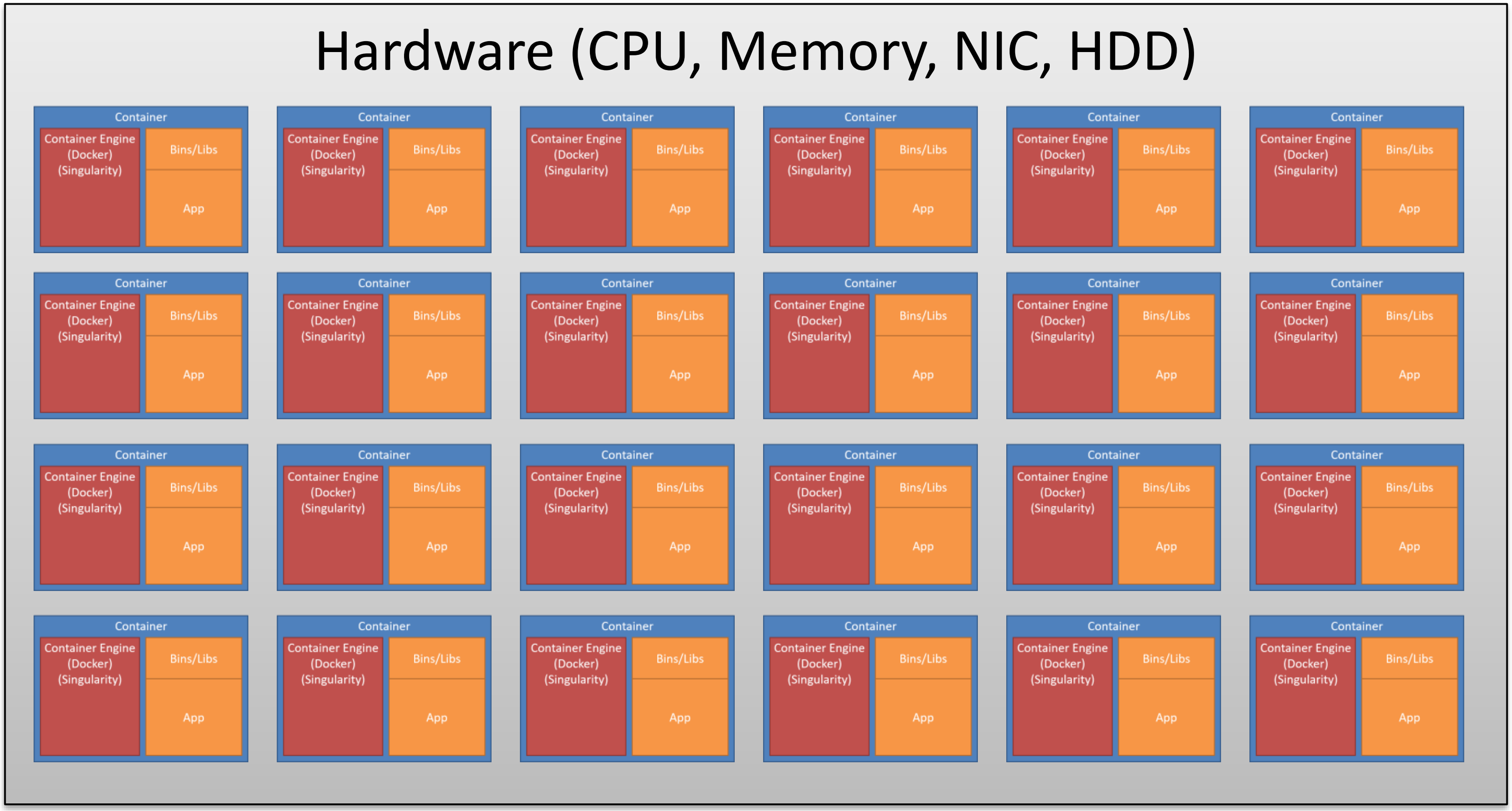
Container

- Encapsulations 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 History

• 1979 – chroot	(changing root directory)
• 2000 – FreeBSD Jails	(early container technology)
• 2001 – Linux VServer	(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)

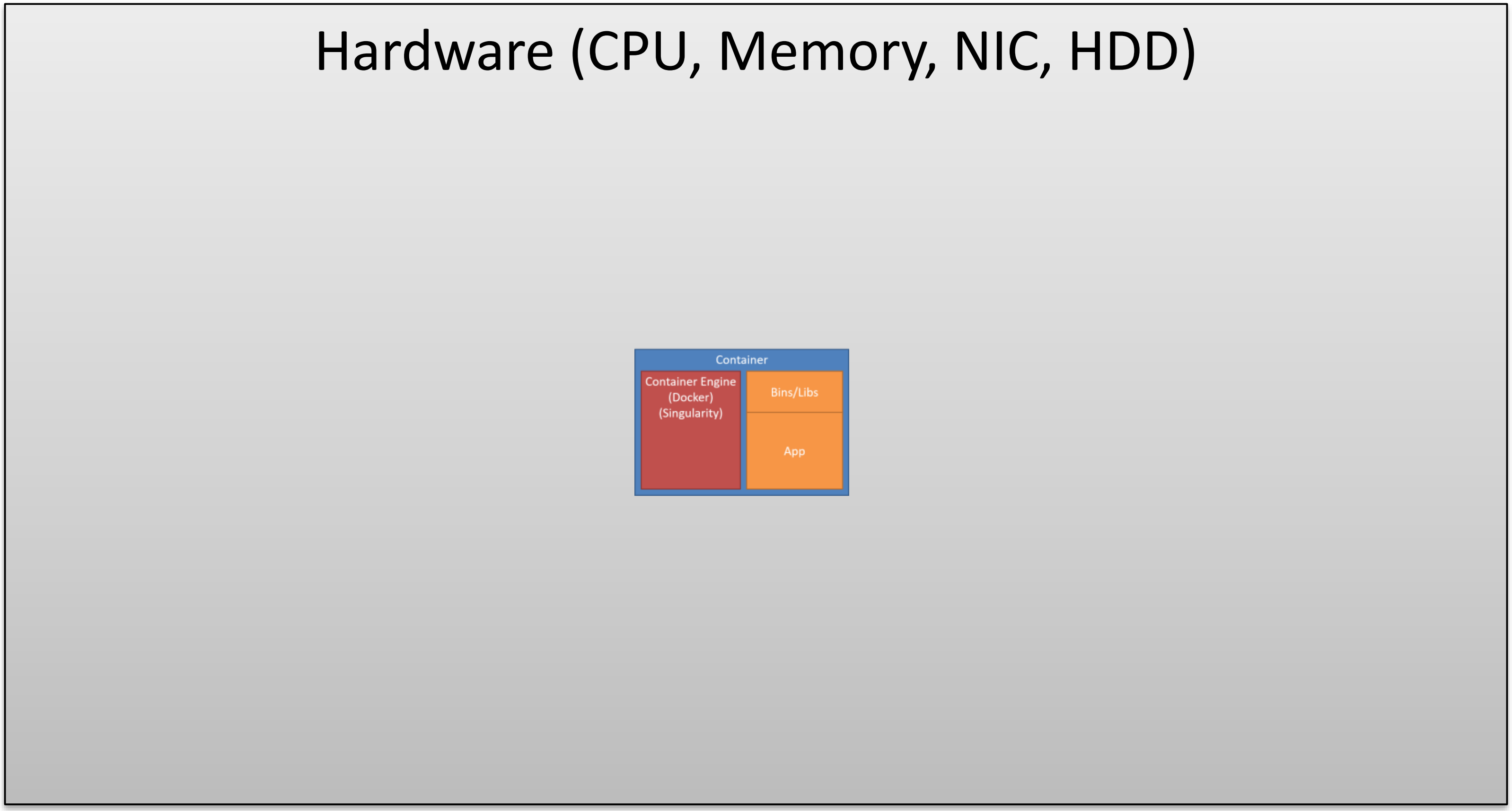
Containers in Industry



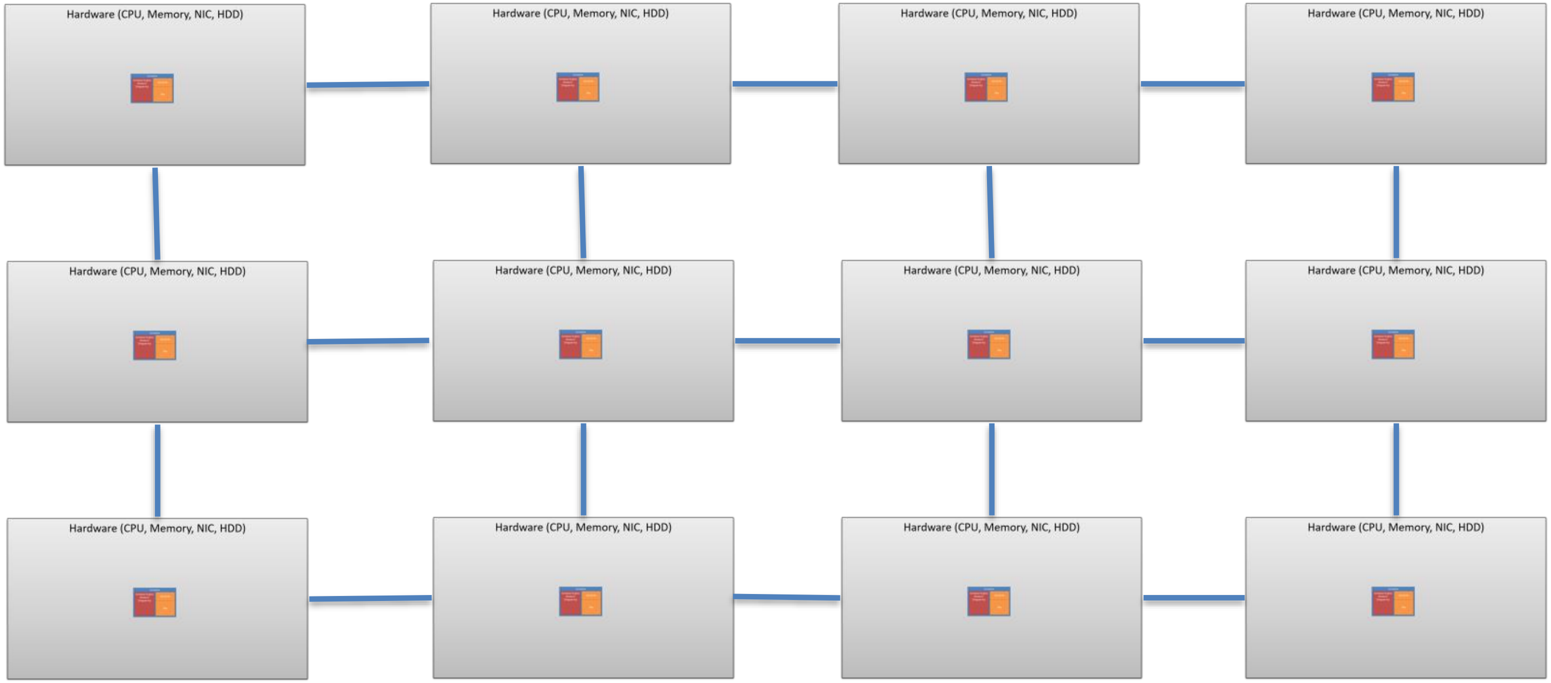
Hardware (CPU, Memory, NIC, HDD)

Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App
Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App
Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App
Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App	Container Container Engine (Docker) (Singularity) Bins/Libs App

Containers in HPC


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

Containers in HPC


Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)
Hardware (CPU, Memory, NIC, HDD)

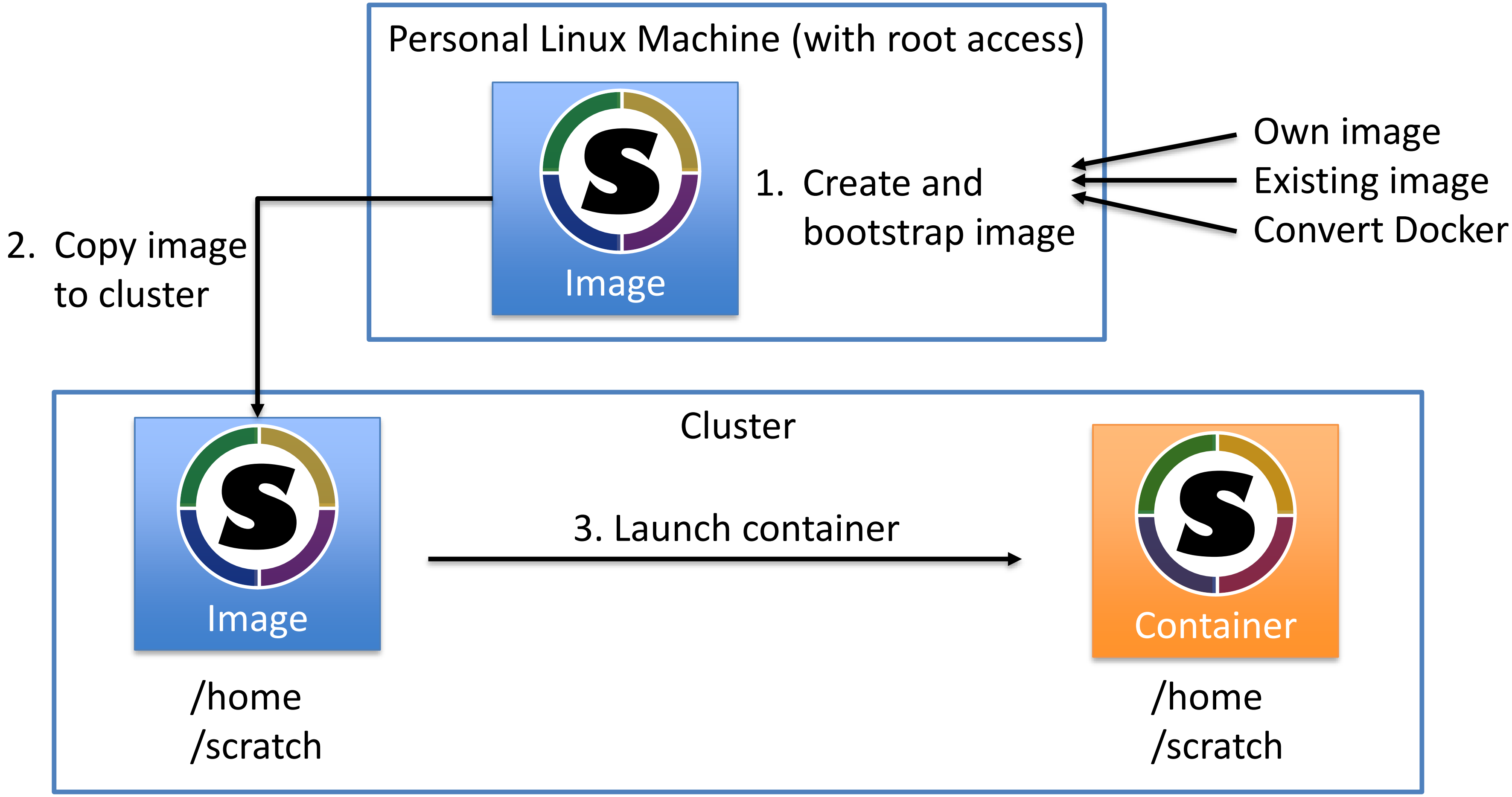
Containers in HPC

- Docker:
  - Most widely used containerization tool
  - Image-based deployment model
  - Perfect for local resources
  - User can gain root access to a host’s filesystem
- Singularity:
  - Permissions inside a container are the same as those outside of the container
  - User can access their files stored outside of the container
  - Designed for use on the cluster, support MPI!
  - 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
  - Possible to access files outside of container
  - Possible to WRITE new files outside of container
  - Home folder is mounted by default

Singularity



Personal Linux Machine (with root access)

1. Create and bootstrap image

Own image

Existing image

Convert Docker

2. Copy image to cluster

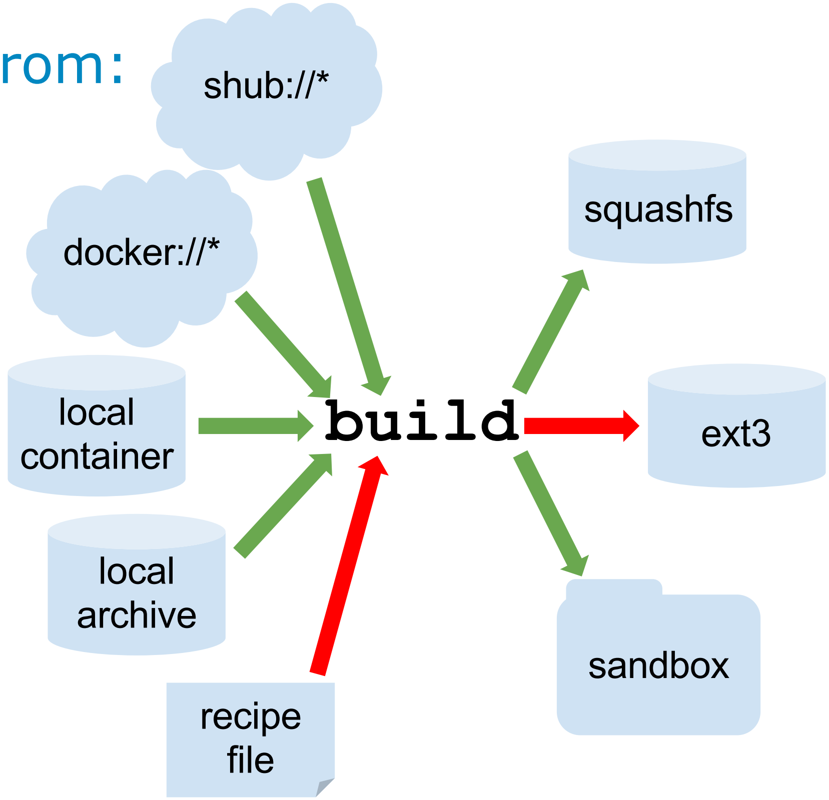
Cluster

3. Launch container

Creating Singularity Image

- Types of Image:
  - Compressed read-only **squashfs** filesystem image
  - Writable **ext3** filesystem image (--writable option)
  - Writable **(ch)root** directory image (--sandbox option)

- Image can be created from:
  - shub://
  - docker://
  - existing container
  - directory
  - archive
  - bootstrap file



IT4Innovations#
národní01#$%@&0
superpočítačové
centrum$@00&1@&

LINUX DAYS

Creating Singularity Image

- `# singularity build ubuntu.img ubuntu.def`
  - Use recipe in def file to install OS and SW

- `# singularity build ubuntu.img docker://ubuntu:latest`
- `# singularity build ubuntu.img shub://singularityhub:ubuntu`
  - Create image from Docker/Singularity Hub

- `# singularity build ubuntu.img /tmp/ubuntu`
  - Create image from chroot directory (sandbox)

Bootstrap Definition File

- Header:
  - Define base OS distribution to use (Bootstrap:)
  - As base can be used:
    - shub – Singularity Hub
    - docker – Docker Hub
    - localimage – saved locally on a computer
    - yum – yum based systems (CentOS, Scientific Linux…)
    - debootstrap – apt based systems (Debian, Ubuntu…)
    - arch – Arch Linux
    - busybox - BusyBox
    - zypper – zypper based systems (Suse, OpenSuse…)
  - Other parameters depends on the base used

Bootstrap 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

Bootstrap 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

Bootstrap Definition File

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

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

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

Singularity on IT4I Clusters

- Version 2.5 installed on both clusters
- Prepared images with:
  - CentOS 6.9
  - CentOS 7.5
  - Fedora 26
  - 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 (e.g. „I hate CentOS, Gentoo rulez!“)

Singularity

Thank you for your attention!

Josef Hrabal

IT4Innovations

IT4Innovations
národní
superpočítačové
centrum

LINUX
DAYS