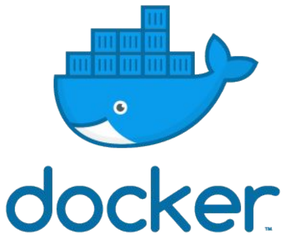
High Performance Containers

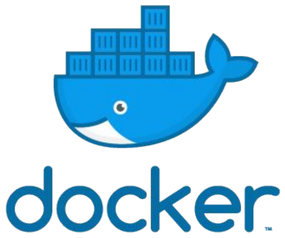
Convergence of Hyperscale, Big Data and Big Compute

Christian Kniep

Technical Account Manager, Docker



Brief Recap of Container Technology



Brief History of Container Technology

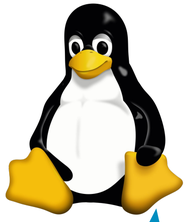
FreeBSD Jails expand on Unix chroot to isolate files

Jails

2000

2001

VServer



Linux-VServer ports kernel isolation, but requires recompilation

Solaris Zones bring the concept of snapshots

Zones

2004

2006

cgroups

Google

Google introduces Process Containers, merged as cgroups

RedHat adds user namespaces, limiting root access in containers

Namespaces

2008

2008

LXC

IBM creates LXC providing user tools for cgroups and namespaces

Docker provides simple user tools and images. Containers go mainstream

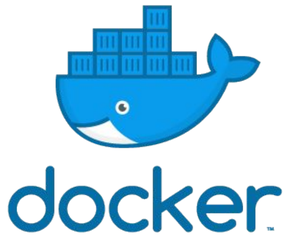
Docker

2013

Container Runtime and Image Format Standards, Jeff Borek, Stephen Walli, KubeCon Dec/2017

Linux Namespaces 101

Short Recap



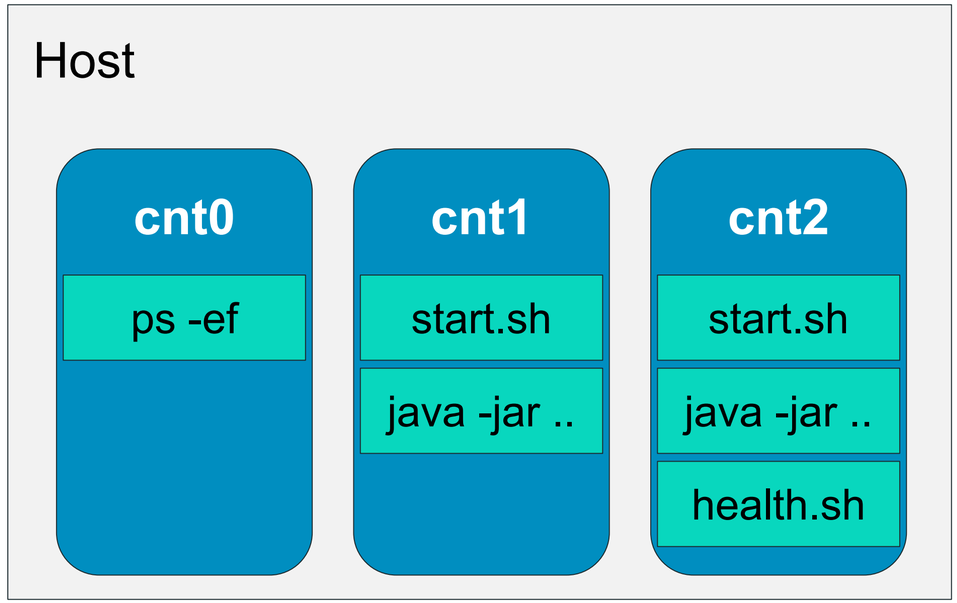
Example PID

Processes Isolation

- host sees all processes with real PID from the Kernels perspective
- first process within PID namespace gets PID=1

```
➜  ~ docker run -ti --rm ubuntu ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  3 11:46 pts/0    00:00:00 ps -ef
➜  ~
```

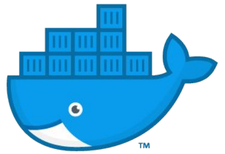
```
➜  ~ docker run -d ubuntu sleep 10
4fa1b37daf4d72ee2008fea406961e52a8348263fec5ecfab8989687edf655f8
➜  ~ docker run -ti --rm --pid=host ubuntu ps -ef|grep sleep
root      9670  9648  0 11:48 ?        00:00:00 sleep 10
➜  ~ docker exec -ti $(docker ps -ql)  ps -ef
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 11:48 ?        00:00:00 sleep 10
root           6     0  0 11:48 pts/0    00:00:00 ps -ef
➜  ~
```



Resource Isolation of Process Groups

7 as of Kernel 4.10

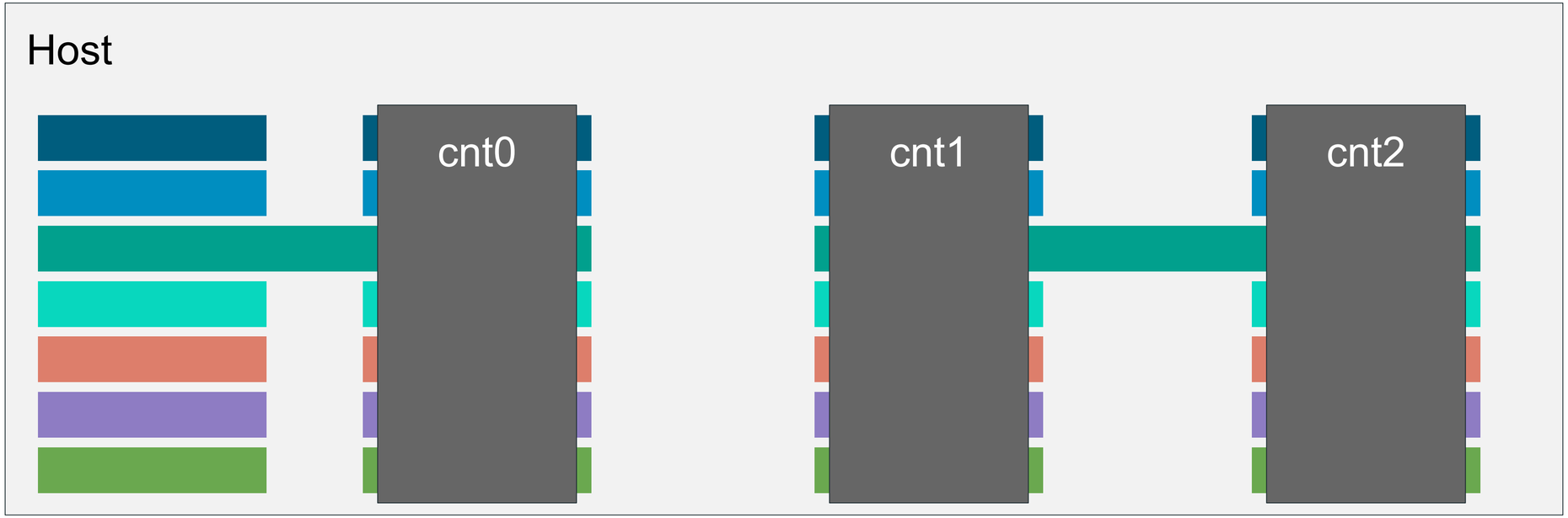
1. MNT: Controls mount points
2. PID: Individual process table
3. NET: Network resources (IPs, routing,...)
4. IPC: Prevents the use of shared memory between processes
5. UTS: Individual host- and domain name
6. USR: Maps container UID to a different UID of the host
7. CGRP: Hides system cgroup hierarchy from container



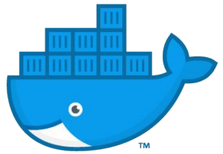
Container Namespaces

A starting container gets his own namespaces.

But can share namespaces with other containers or even the host

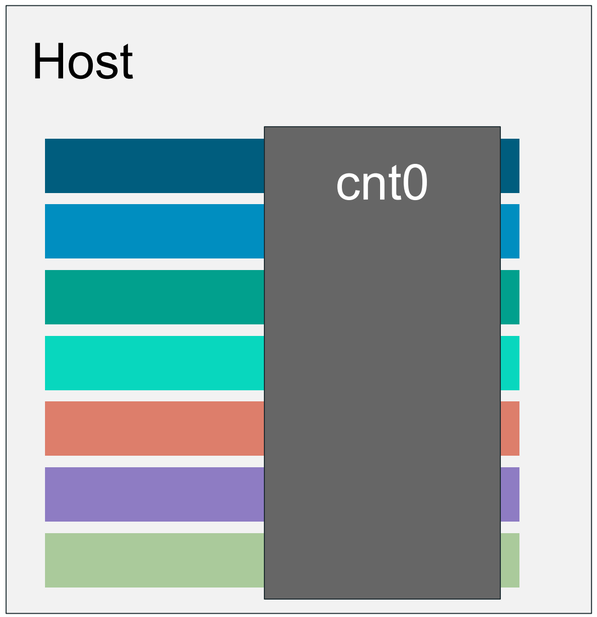


MNT PID NET IPC UTS USR CGRP



All In

When using all host namespaces - we are on the host (almost like ssh).



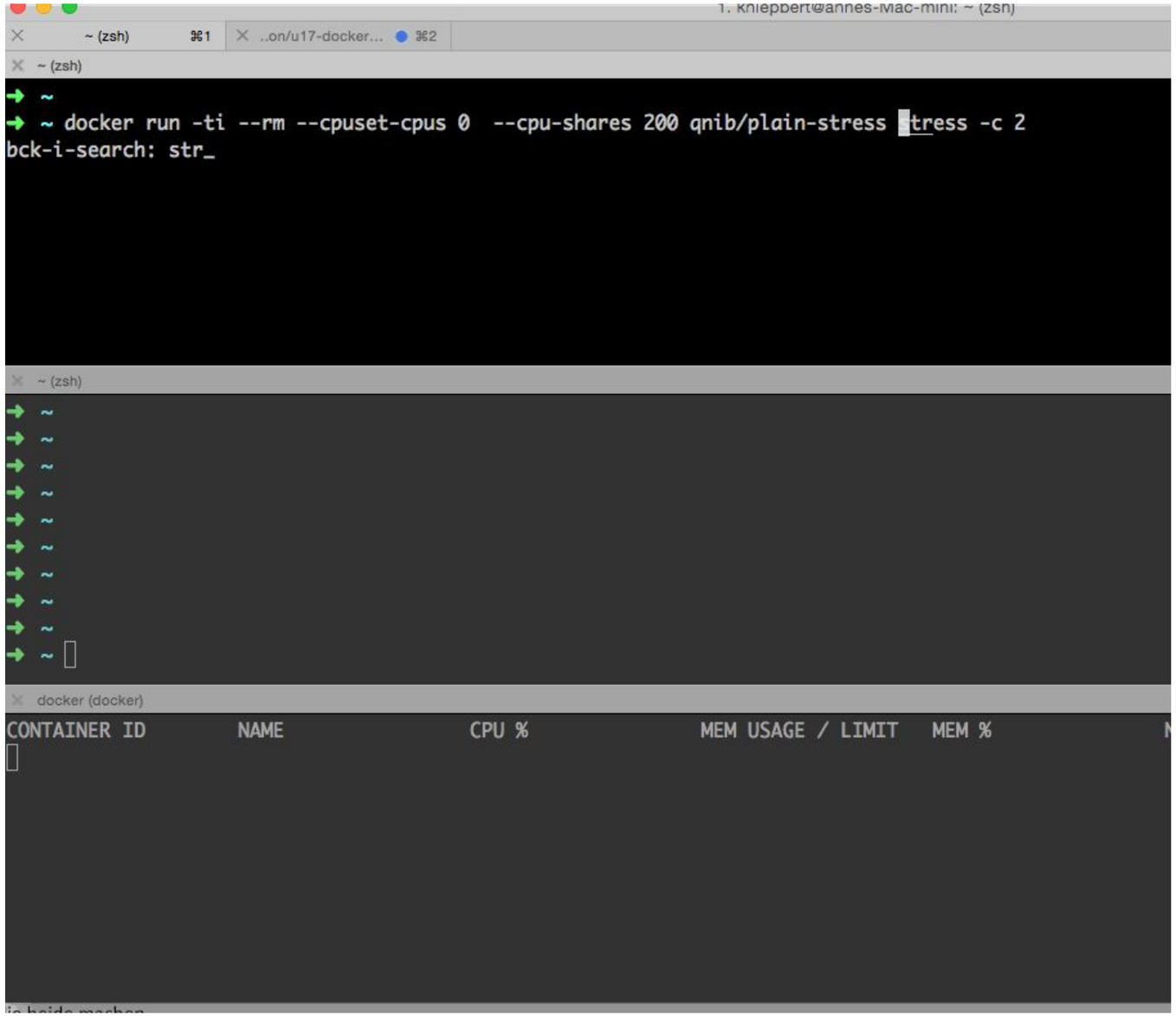

```
$ docker run -ti --rm \
        --privileged \
        --security-opt=seccomp=unconfined \
        --pid=host \
        --uts=host \
        --ipc=host \
        --net=host \
        -v /:/host \
        ubuntu bash
root@linuxkit-025000000001:/# chroot /host
/ # ash
/ #
```

MNT PID NET IPC UTS USR CGRP

CGroups

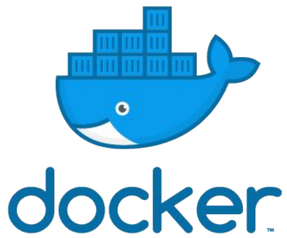
While namespaces isolate,

Control Groups constraint resources.



CGroups / Filesystem Layering

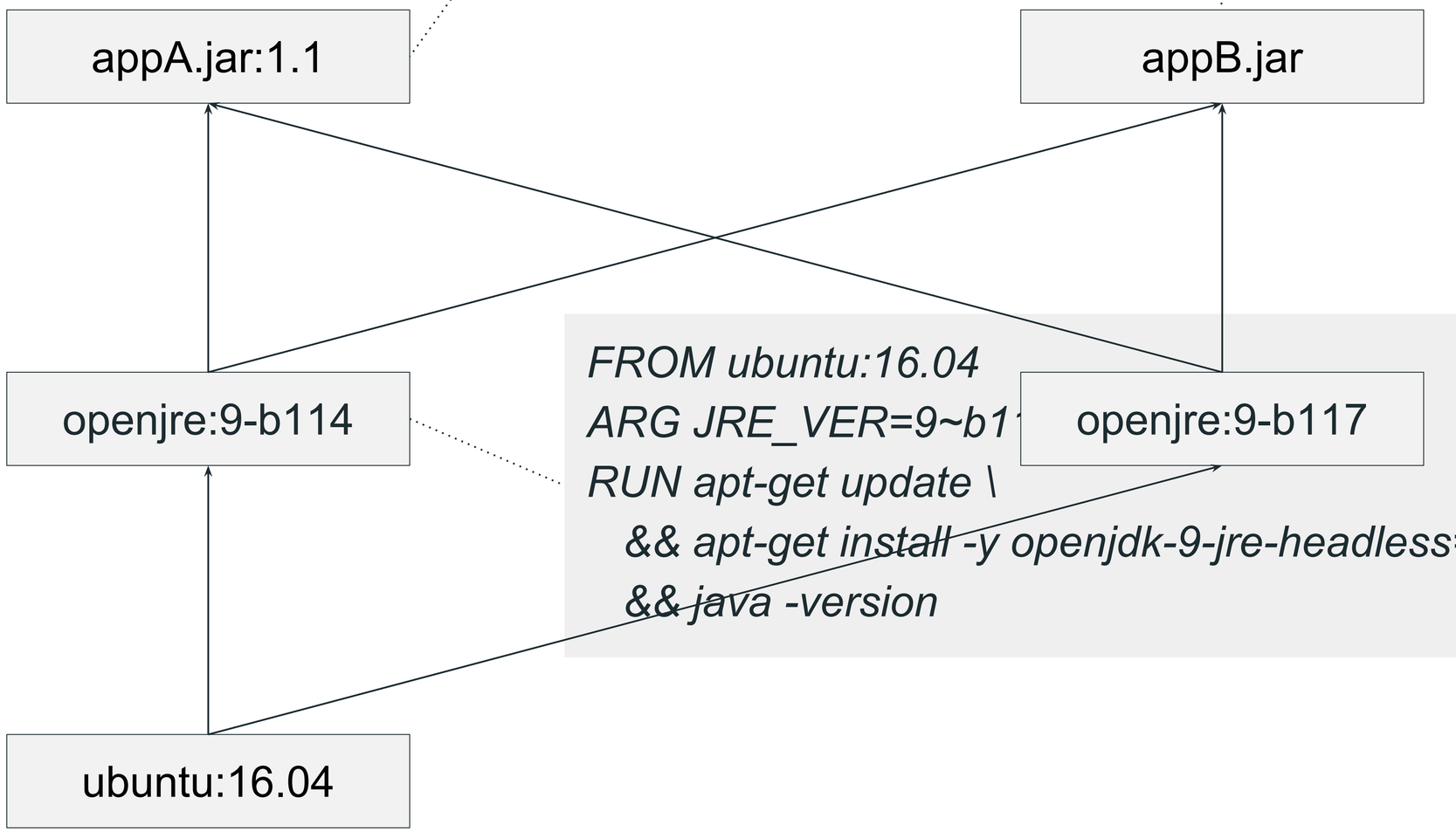
Short Recap [cont]



Overlay Filesystem

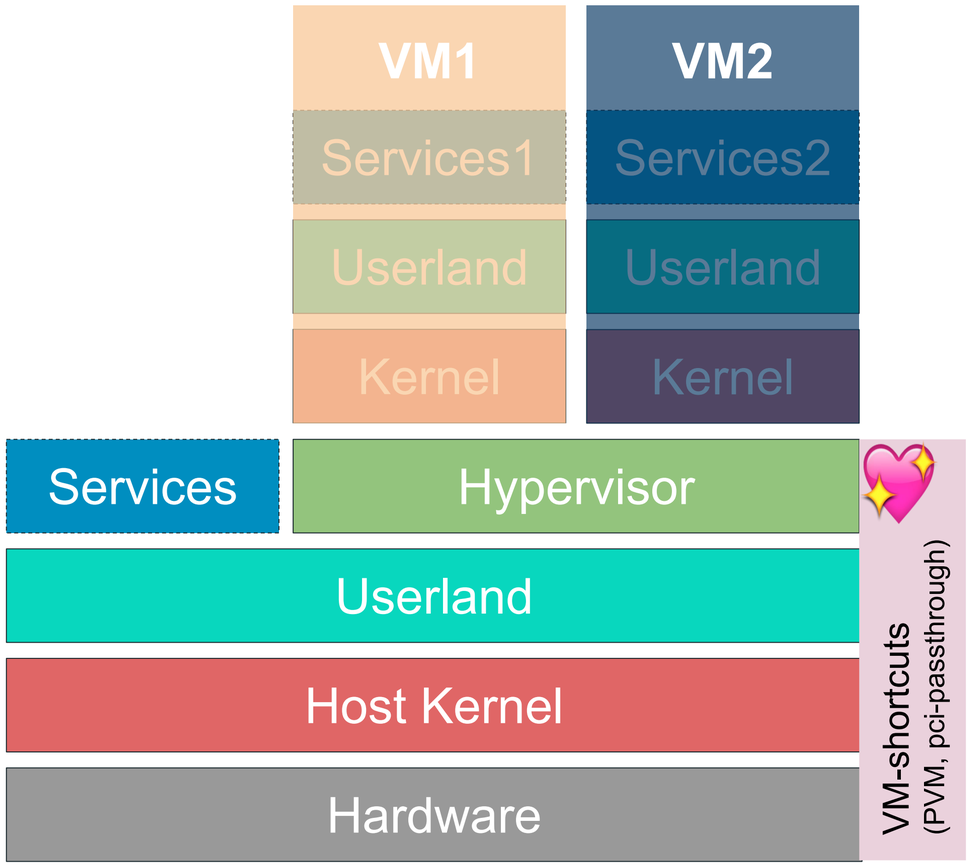
Compose a FS from multiple pieces

```
ARG FROM openjre:9-b114
COPY appA.jar /usr/local/bin/
CMD [“java”, “-jar”, “/usr/local/bin/appA.jar”]
```

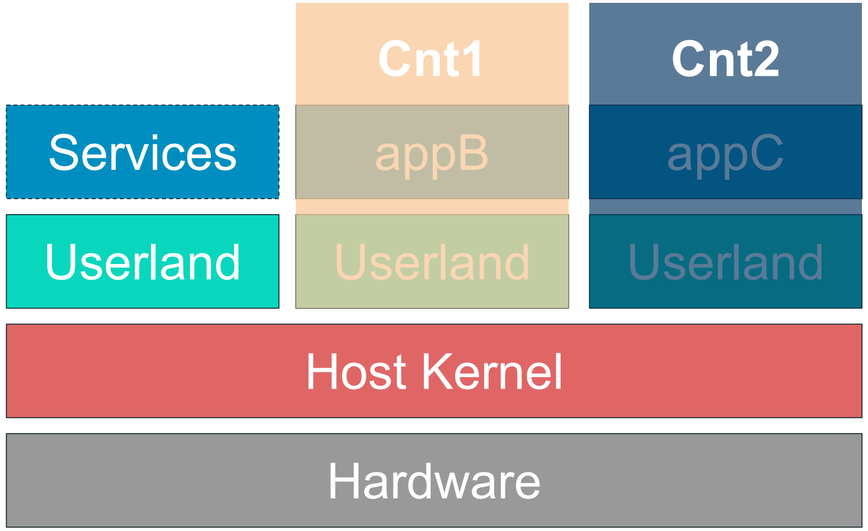



```
FROM ubuntu:16.04
ARG JRE_VER=9~b1
RUN apt-get update \
    && apt-get install -y openjdk-9-jre-headless=${JRE_VER} \
    && java -version
```

Stacked View



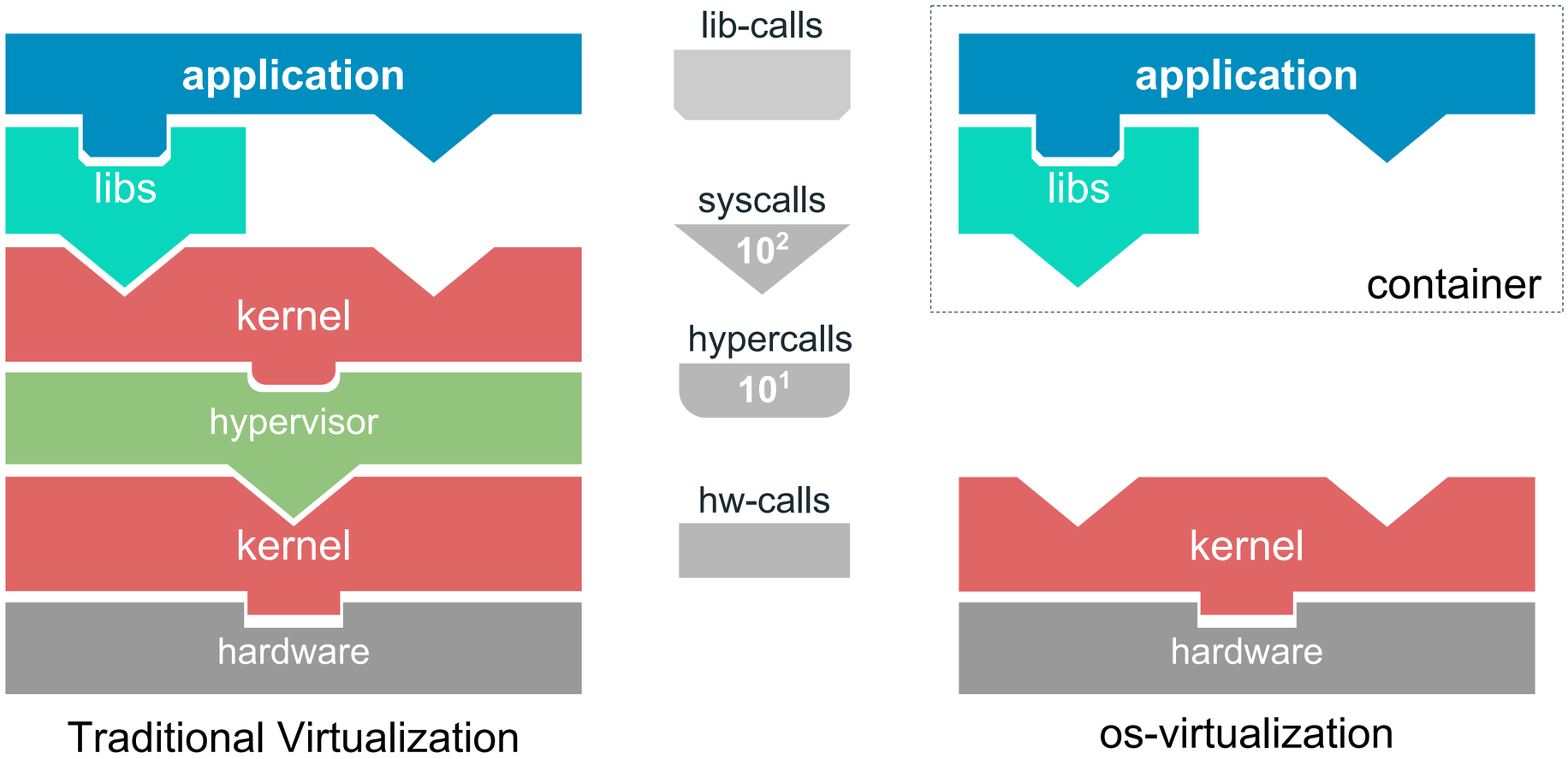
Traditional Virtualization



os-virtualization

Interface View

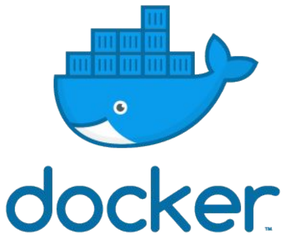
From Application to Kernel



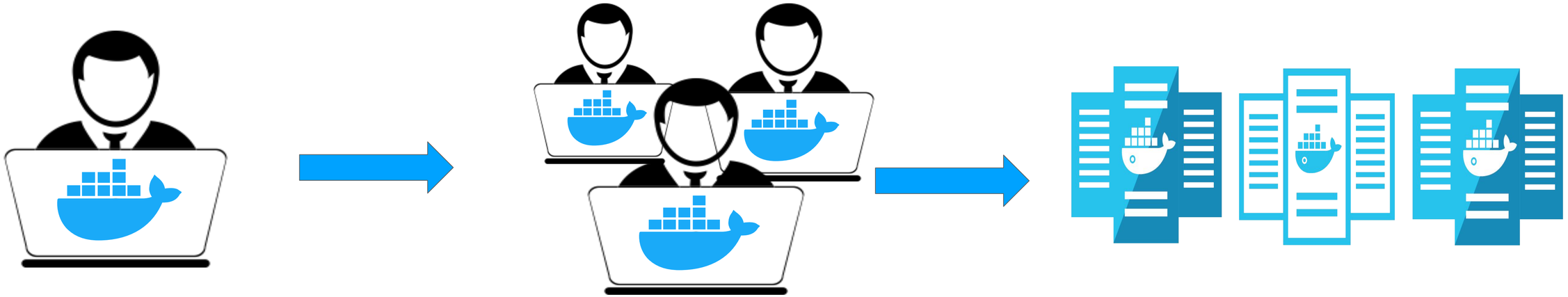
Traditional Virtualization

os-virtualization

Why apply Containers to HPC?



SciOps



Researcher Workstation

Peer Review + Collaboration

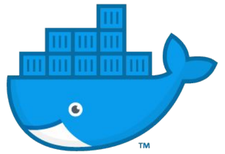
HPC Compute Nodes

docker

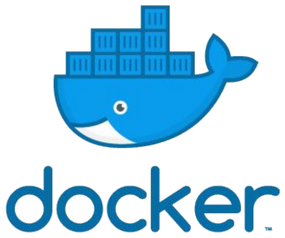
Bridging the Technology Gap

Containers removing barriers of entry:

- Hardware Expertise
- Software Expertise
- Workflow/System Integration
- CapEx / OpEx

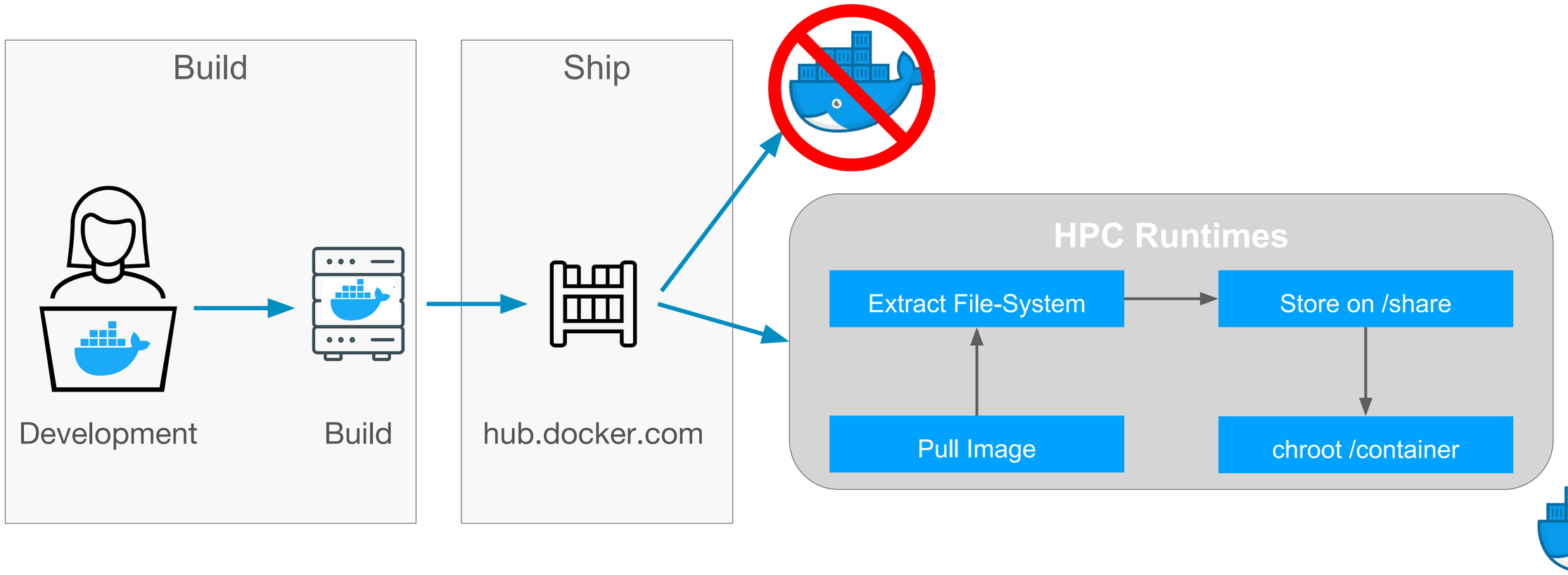


How are containers used in HPC today?



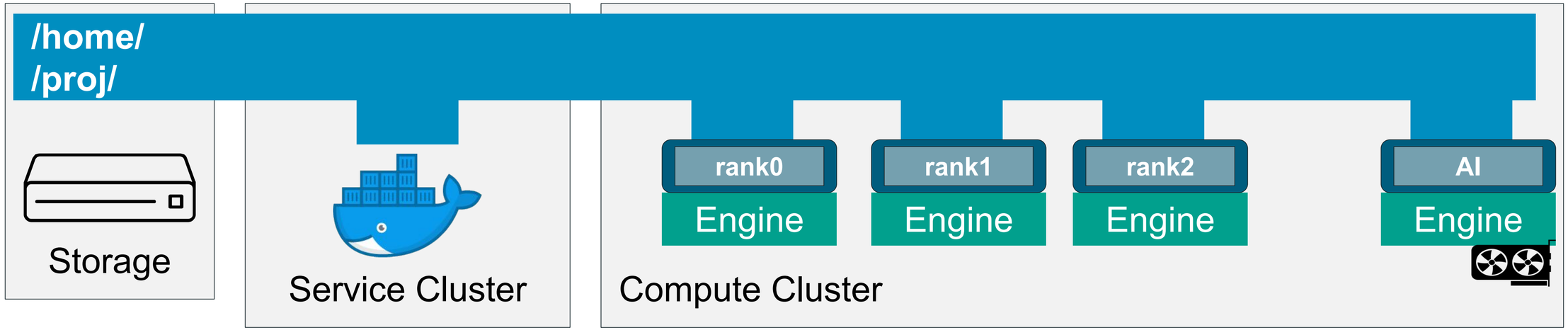
Current Solutions

Lack of HPC focus gave birth to HPC workarounds.



Scientific Environments

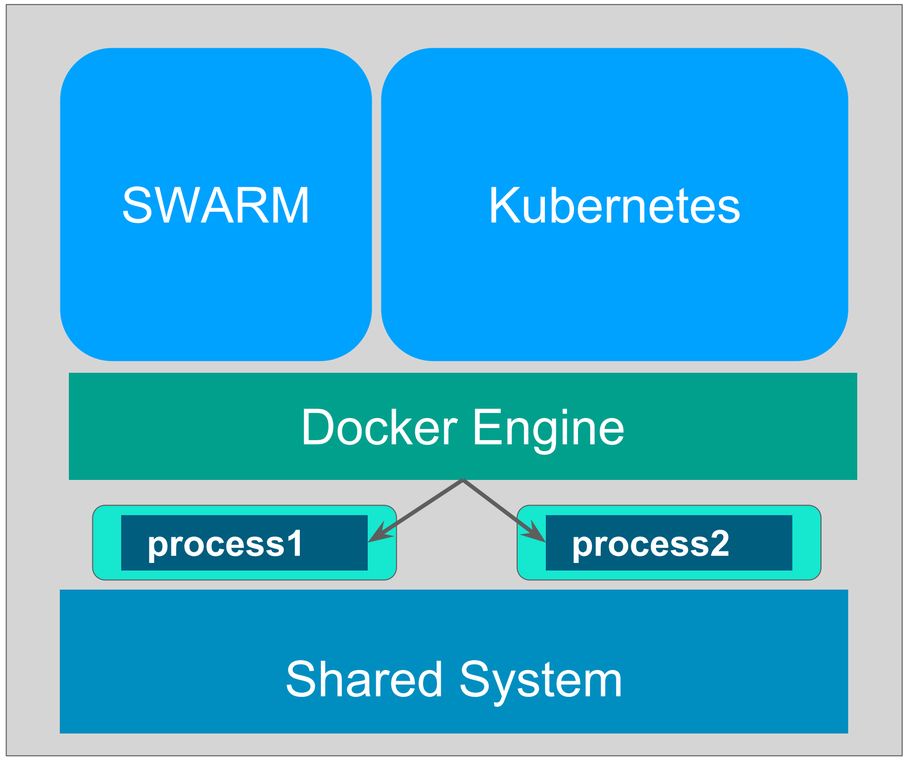
Scientific end-users expect the environment to be set up for them, without prior knowledge about the specifics of the cluster.

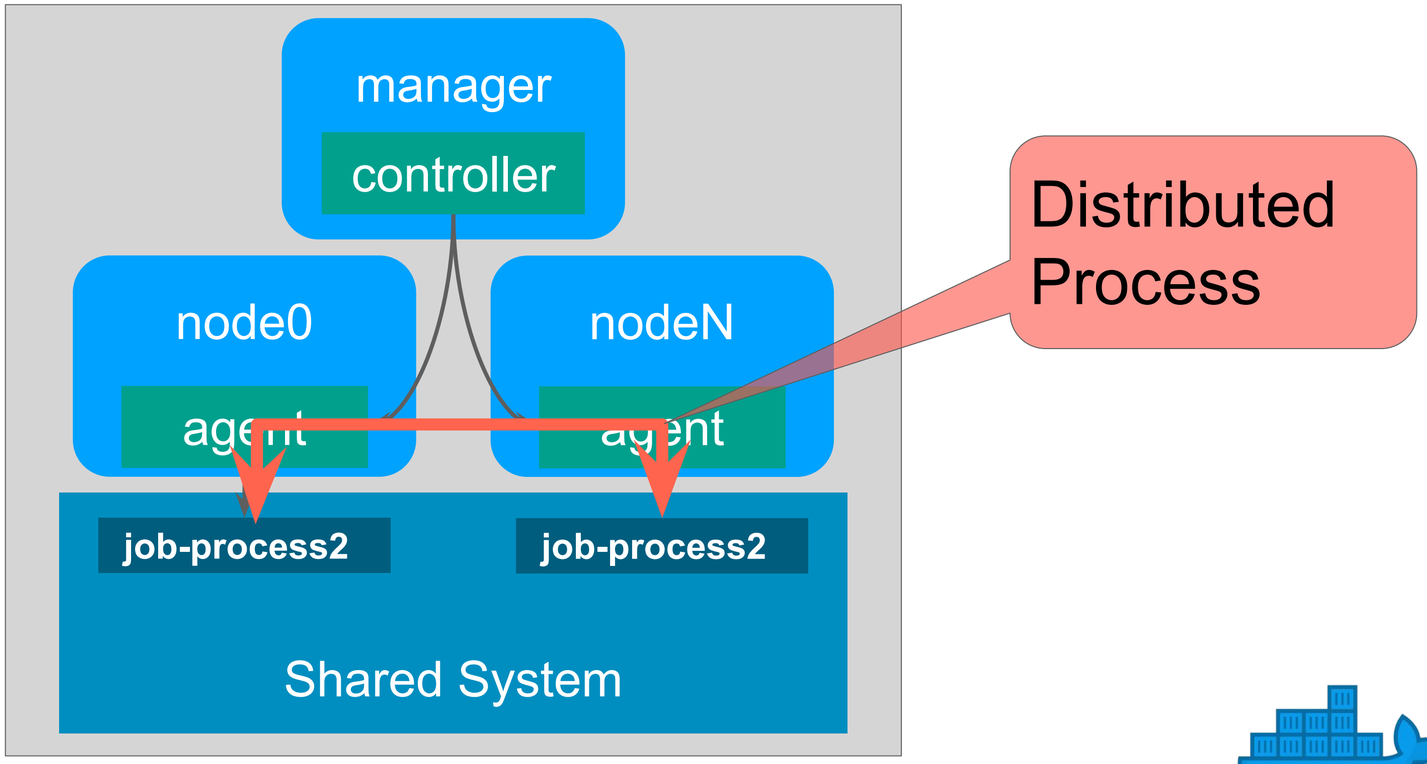


Service vs Batch Scheduling

Traditionally container workloads are scheduled in a descriptive manner, as tasks (pods) on worker nodes.

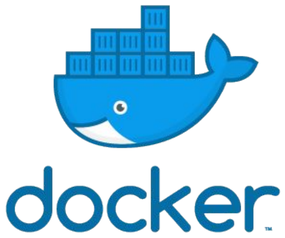
HPC schedules a workloads as a batch job on multiple nodes.





HPC Workload Scheduler

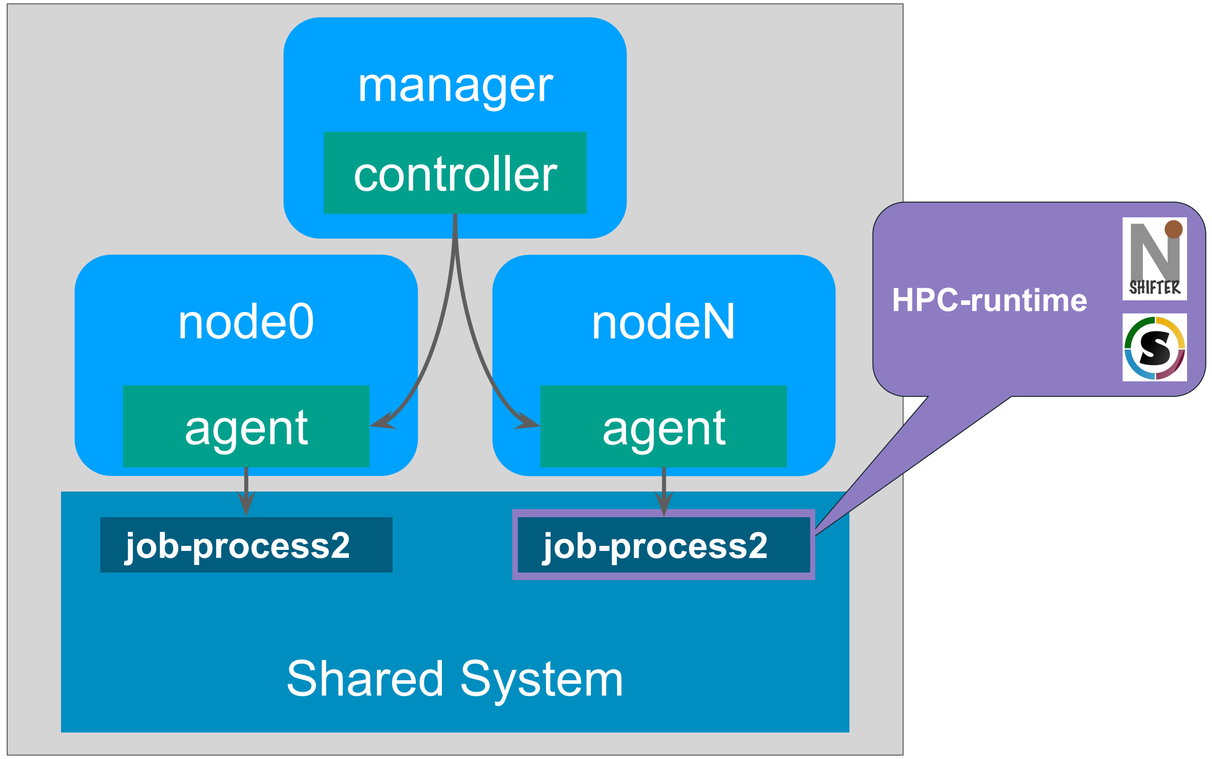
DEMO!



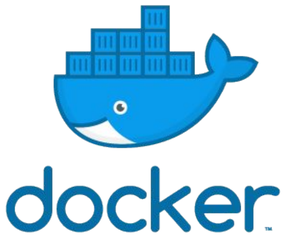
Current Solutions [cont]

HPC-specific workaround

+ Drop-in replacement as it wraps the job
- Not OCI compliance
- No integration with upstream container ecosystem
- hard to combine with new workloads

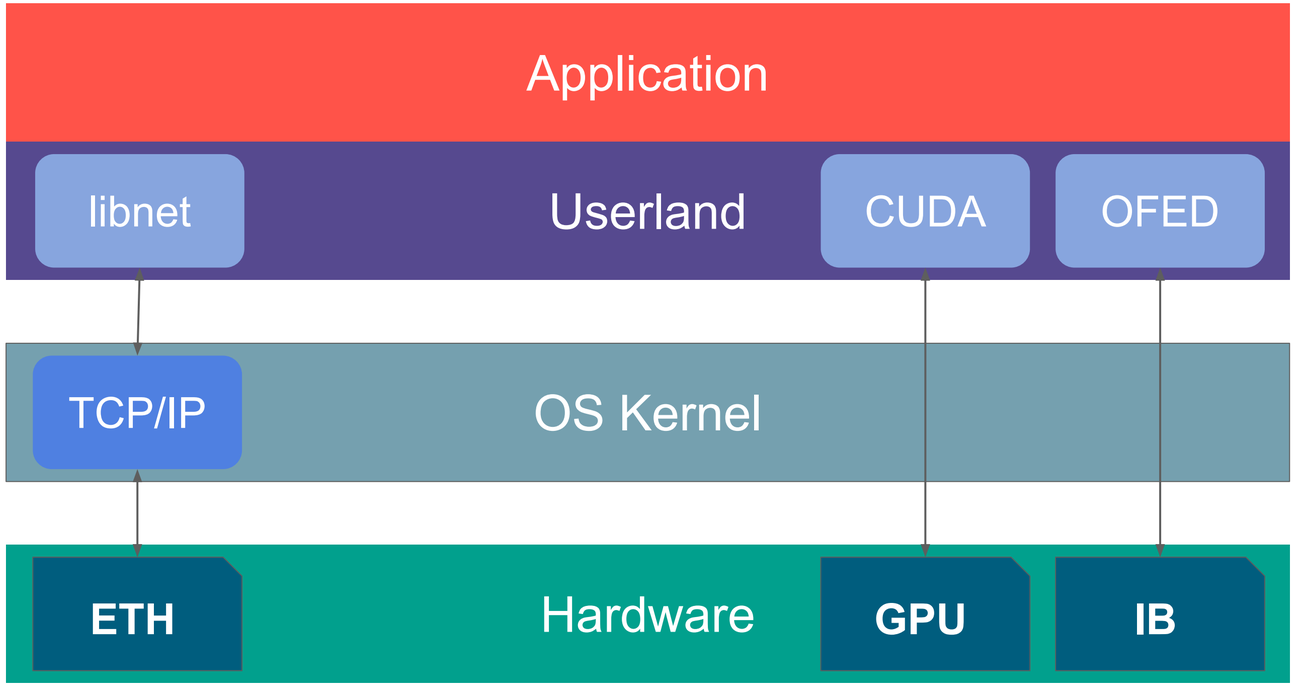


HPC Challenges



Kernel-bypassing Devices

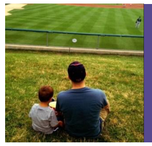
To achieve the highest performance possible the kernel got squeezed out of the equation for performance-critical parts.



The Stack

Client

Engine



containerd

NVIDIA

runc

```
--device=/dev/nvidia0
```

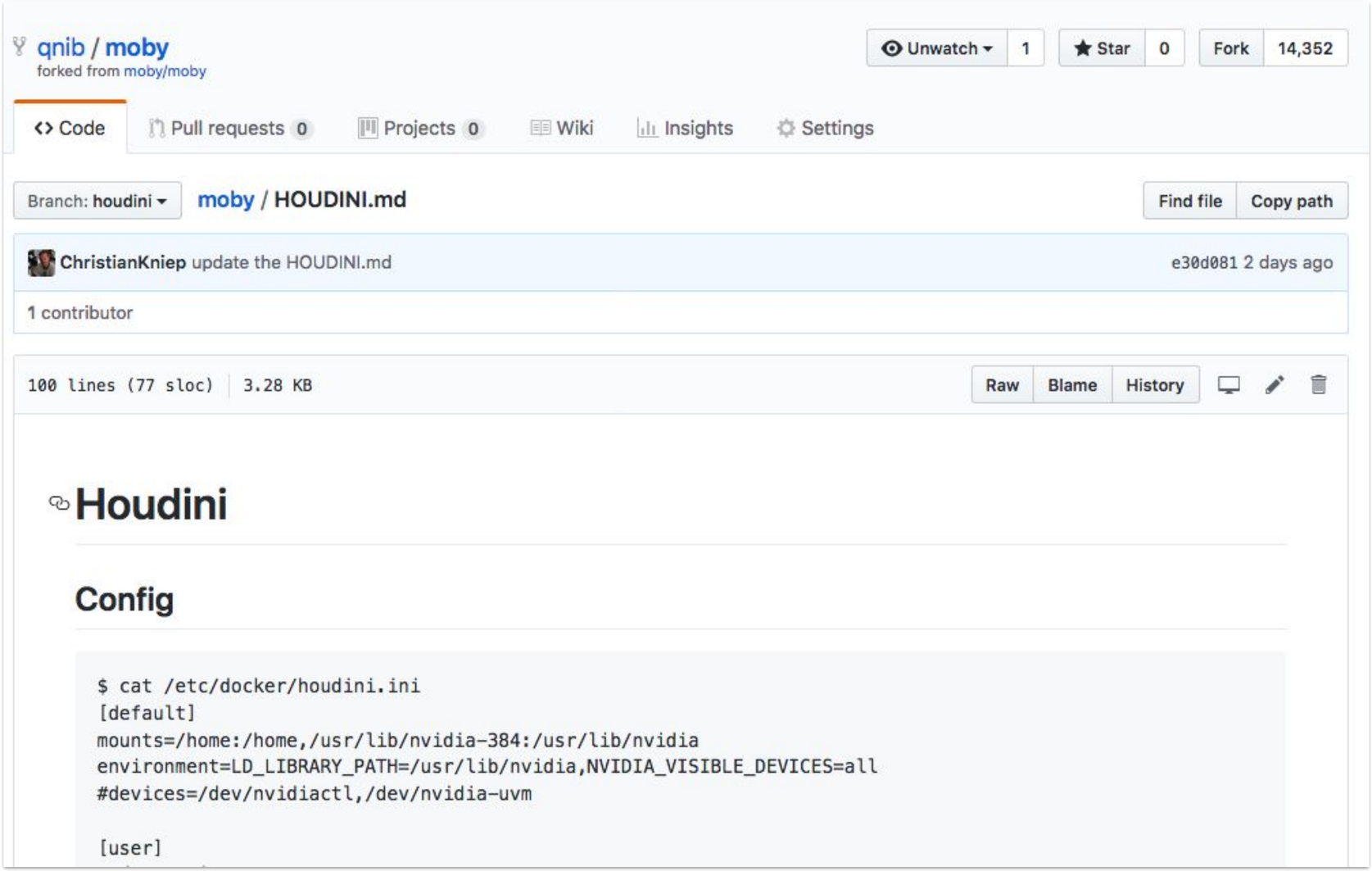
```
"Devices": [{
                "PathOnHost": "/dev/nvidia0",
                "PathInContainer": "/dev/nvidia0",
                "CgroupPermissions": "rwm"
}],
```

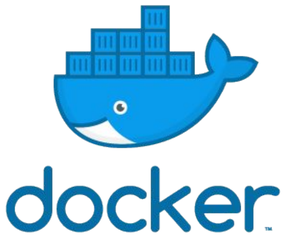
```
"devices": [{
        "path": "/dev/nvidia0", "type": "c",
        "major": 195, "minor": 0,"fileMode": 8630,
        "uid": 0, "gid": 0
      },
```

```
"hooks": { "prestart": [ {"path": “/usr/local/bin/nvidia.sh"}]}
```

docker

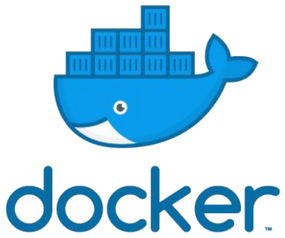
Houdini Plugin



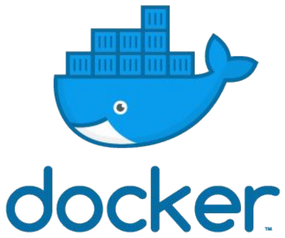


Houdini Plugin [cont]

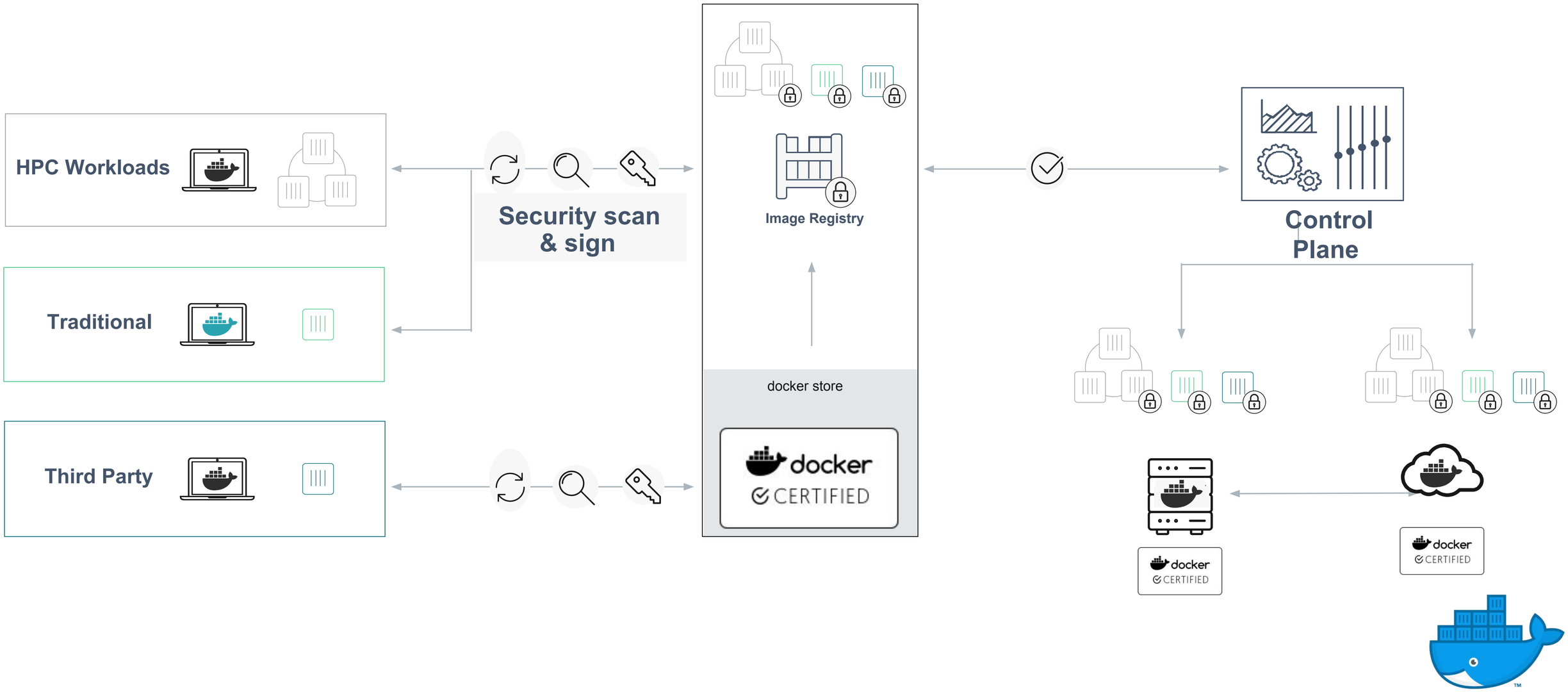
DEMO!



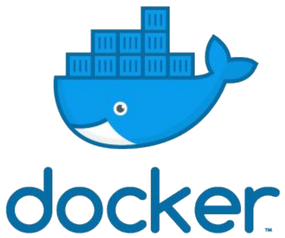
HPC Opportunities



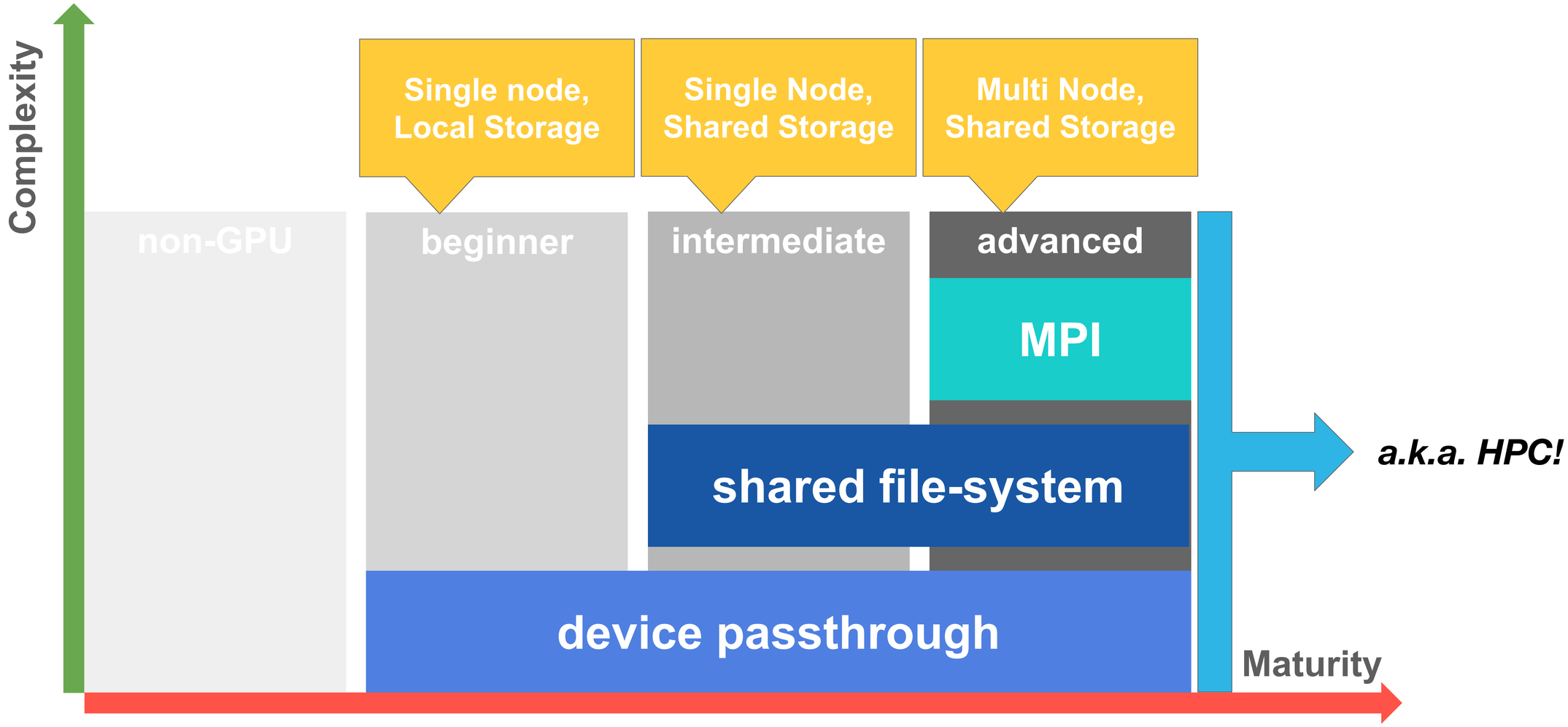
Leveraging HPC in the Enterprise / Enterprise in HPC



HPC @Docker: What’s next?



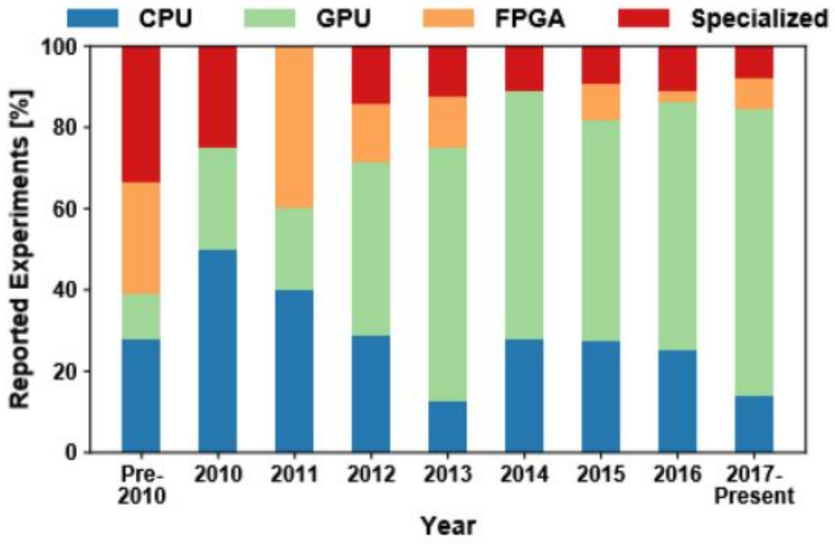
Convergence of AI and HPC

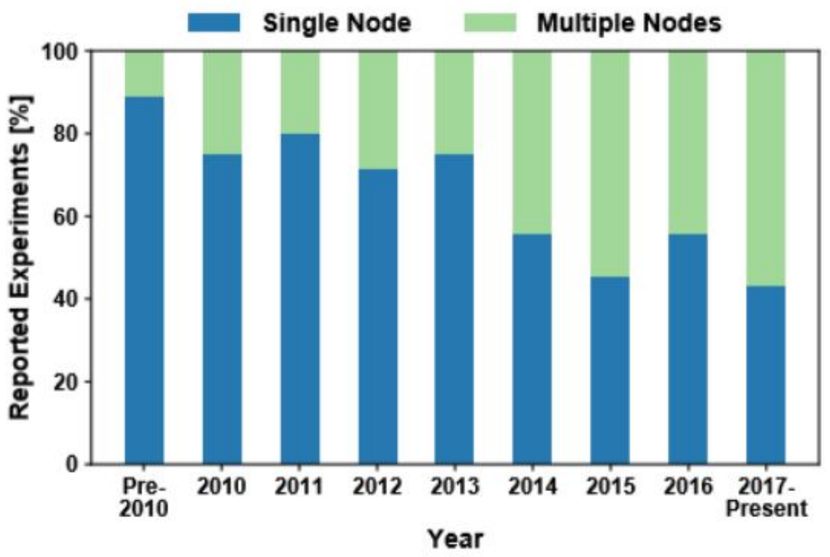


Evidence for AI/DL trends

Trends in deep learning: hardware and multi-node

The field is moving fast – trying everything imaginable – survey results from 227 papers in the area of parallel deep learning





Deep Learning is largely on distributed memory today!

T. Ben-Nun, T. Hoefler: Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis, arXiv Feb 2018

Steps

Low-hanging and high hanging fruit

Device Passthrough

Rather simple

Shared File-System

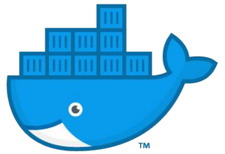
get UID:GID(s) from somewhere

MPI enablement

- Engine vs. Orchestrator vs. Workload Manager: Who is in charge?
- including PMIx into the engine?
- simple batch scheduling in SWARM?

Docker Ecosystem Goodies

- Secure Supply Chain
- Reproducible, OS idenpenced science



Docker for Science