Multi-host containerised HPC cluster

The new Docker networking put into action to spin up a SLURM cluster

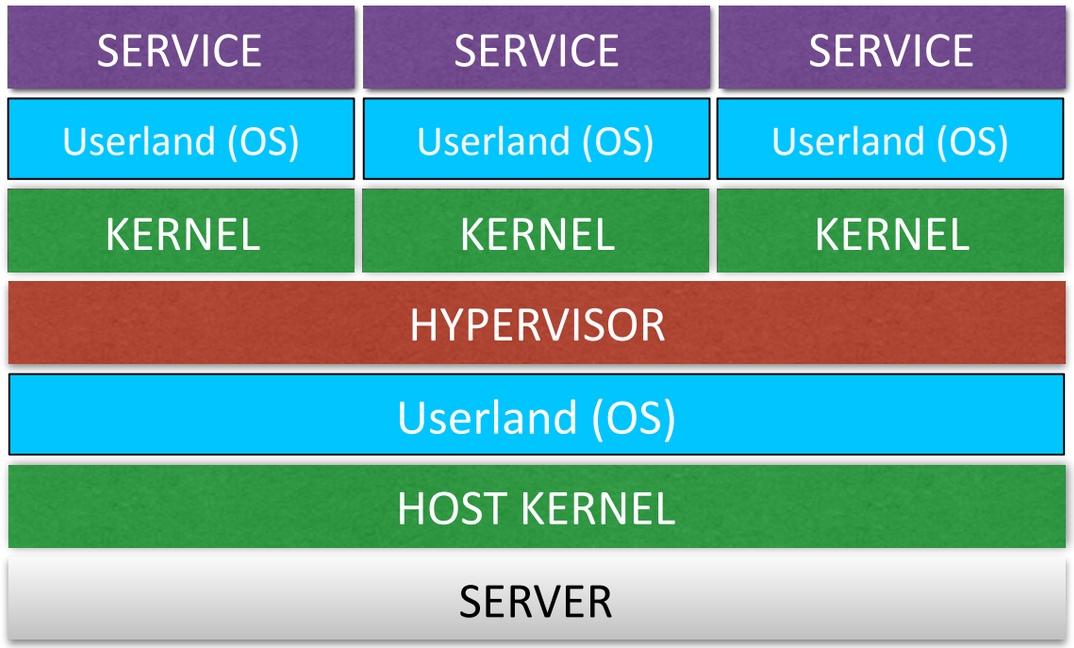
HPC ADVISORY COUNCIL

The Bits and Pieces…

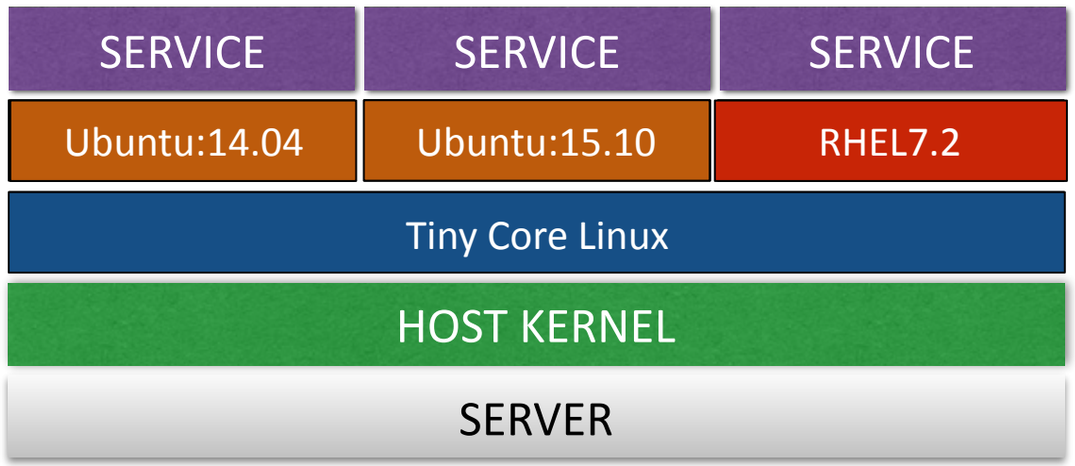
Linux Containers

Containers do not spin up a distinct kernel

- all containers & the host share the same
- they are separated by Kernel Namespaces
- user-lands are independent



Traditional Virtualisation



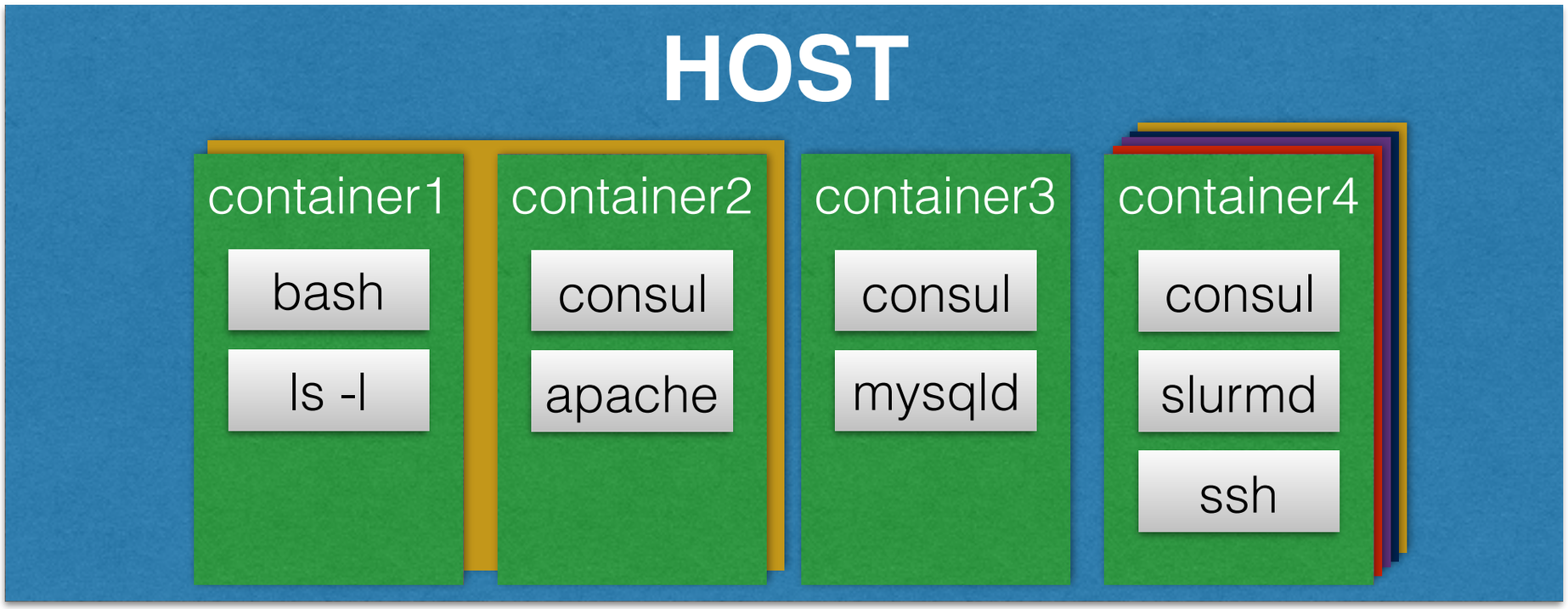
Containerisation

Docker [cont]

Containers are ‘grouped processes’

- isolated by Kernel Namespaces
- resource restrictions applicable through CGroups (disk/netIO)

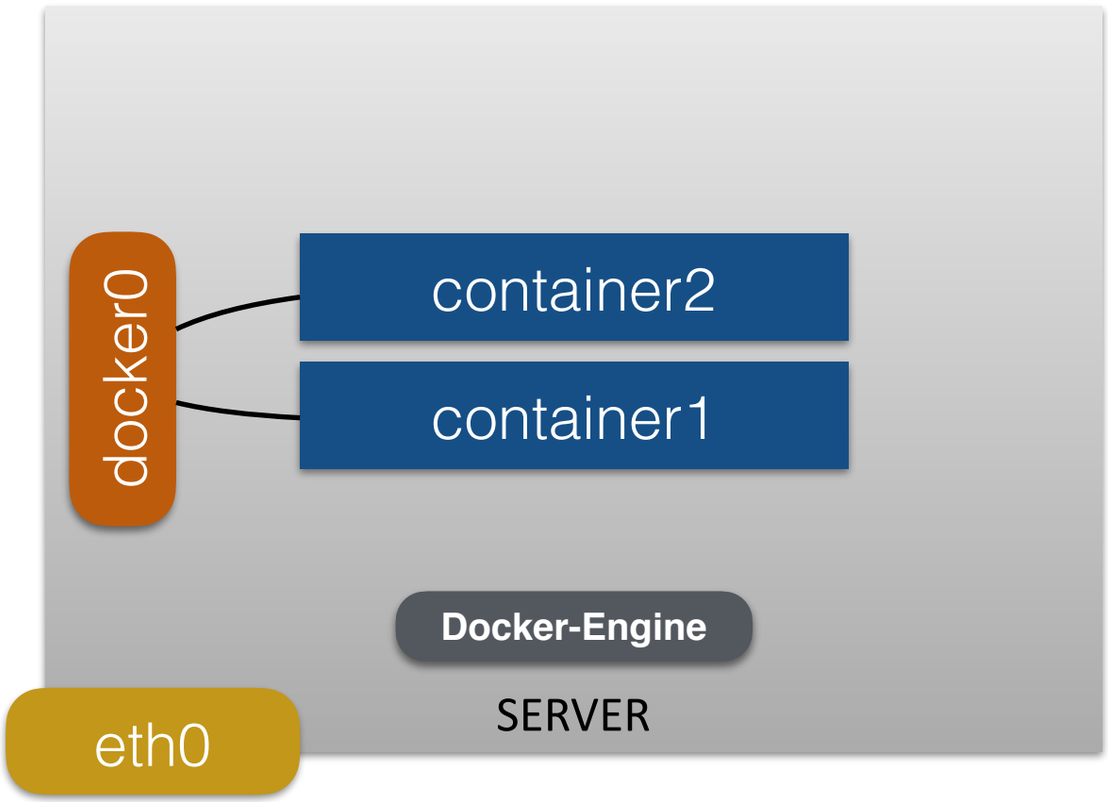
Namespaces: PID | Network | Mount | IPC | UTS



Docker Engine

Container Runtime Daemon

- creates/…/removes containers, exposes REST API
- handles Namespaces, CGroups, bind-mounts, etc.
- IP connectivity by default via ‘host-only’ network bridge



Docker Compose

Describes stack of container configurations

- instead of writing a small bash script…
- … it holds the runtime configuration as YAML file.

```
docker run -d -h consul --name consul --dns 127.0.0.1 -p 8500 \
           -e DC_NAME=dc1 -e CONSUL_BOOTSTRAP=true qnib/alpn-consul
```

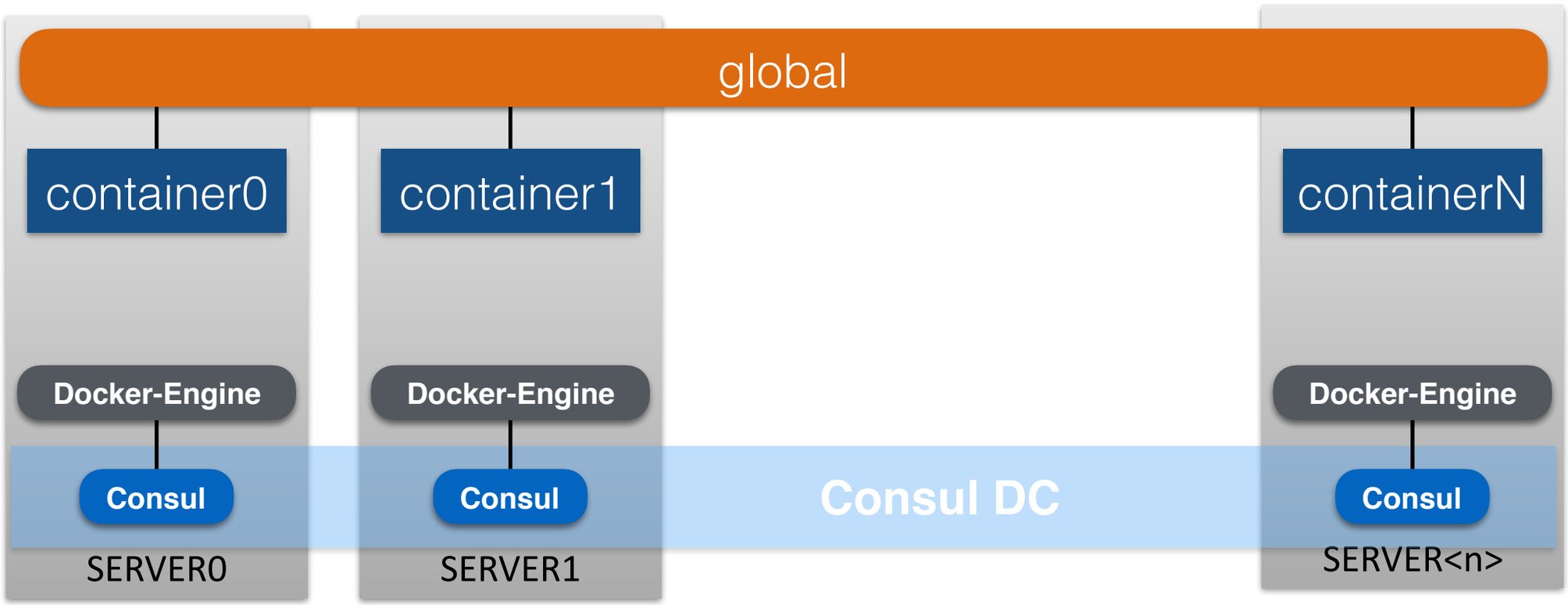
```
consul:
    image: qnib/alpn-consul
    hostname: consul
    container_name: consul
    dns: 127.0.0.1
    ports:
     - 8500
    environment:
     - DC_NAME=dc1
     - CONSUL_BOOTSTRAP=true

registry:
    hostname: registry
```

Docker Networking

Docker Networking spans networks across engines

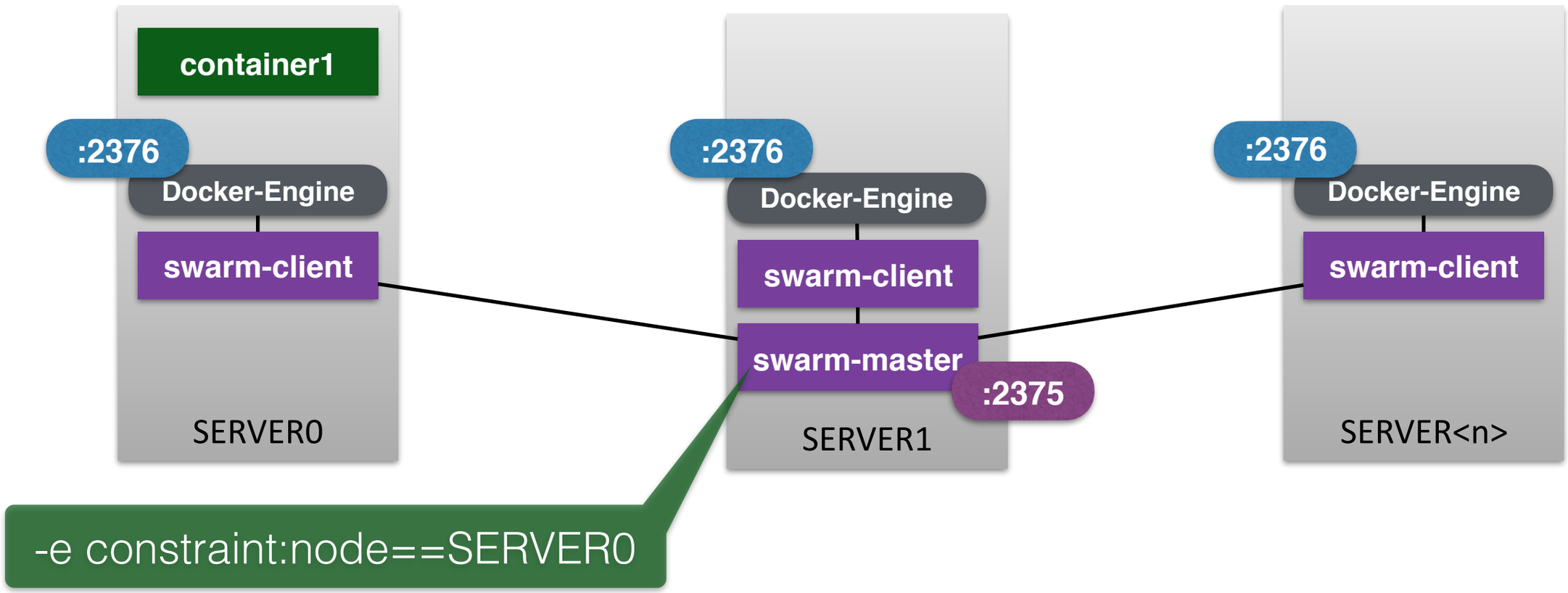
- KV-store to synchronise (Zookeeper, etcd, Consul)
- VXLAN to pass messages along



Docker Swarm

Docker Swarm proxies docker-engines

- serves an API endpoint in front of multiple docker-engines
- does placement decisions.



Docker Swarm [cont]

```
[[root@venus001 ~]# docker -H docker1:2375 info |egrep --color=never "(192|Container|^\w+)"
Containers: 50
Images: 374
 docker1: 192.168.12.11:2376
  └ Containers: 5
 venus001: 192.168.12.181:2376
  └ Containers: 12
 venus002: 192.168.12.182:2376
  └ Containers: 6
 venus003: 192.168.12.183:2376
  └ Containers: 4
 venus004: 192.168.12.184:2376
  └ Containers: 4
 venus005: 192.168.12.185:2376
  └ Containers: 5
 venus006: 192.168.12.186:2376
  └ Containers: 5
 venus007: 192.168.12.187:2376
  └ Containers: 3
 venus008: 192.168.12.188:2376
  └ Containers: 6
CPUs: 76
Total Memory: 266.5 GiB
Name: a7d177accc81
[root@venus001 ~]#
```

query docker-swarm

Introduce new Technologies

Introducing new Tech

Self-perception when introducing new tech…

credit: TF2 - Meet the Pyro

Introducing new Tech

… not always the same as the perception of others.

credit: TF2 - Meet the Pyro

Docker Buzzword Chaos!

Distributions

On-Premise & OverSpill

Auto-Scaling

Orchestration

Solutions

self-healing

production-ready

enterprise-grade

Misconception

1. Linux Containers - perceived as ‘yet another VM’
   - VMs were easy to shoehorn in legacy workflow, containers might break it
2. Use-cases span a lot of ground
   - laptop, dev-cluster, staging, prod
3. Docker Inc. focuses on developer satisfaction
   - iptables were dropped without say so in the beginning
4. Namespaces is a new concept
   - merge one namespace, but not others.

Vision

1. No special distributions
   - useful for certain use-cases, such as elasticity and green-field deployment
   - not so much for an on-premise datacenter w/ legacy in it.
2. Leverage existing processes/resources
   - install workflow, syslog, monitoring
   - security (ssh infrastructure), user auth.
3. keep up with docker ecosystem
   - incorporate new features of engine, swarm, compose
   - networking, volumes, user-namespaces

Reduce to the max!

Put a Docker **on** it!

1. Leverage existing install/configuration workflow
   - Kickstart + ansible-run
2. Don’t focus on corner cases
   - postpone special purpose container (e.g. needs multi-tenant IB usage)
3. HPC environments assumptions
   - single-tenant
   - focus on performance

The Setup

Testbed

Hardware (courtesy of HPC ADVISORY COUNCIL)

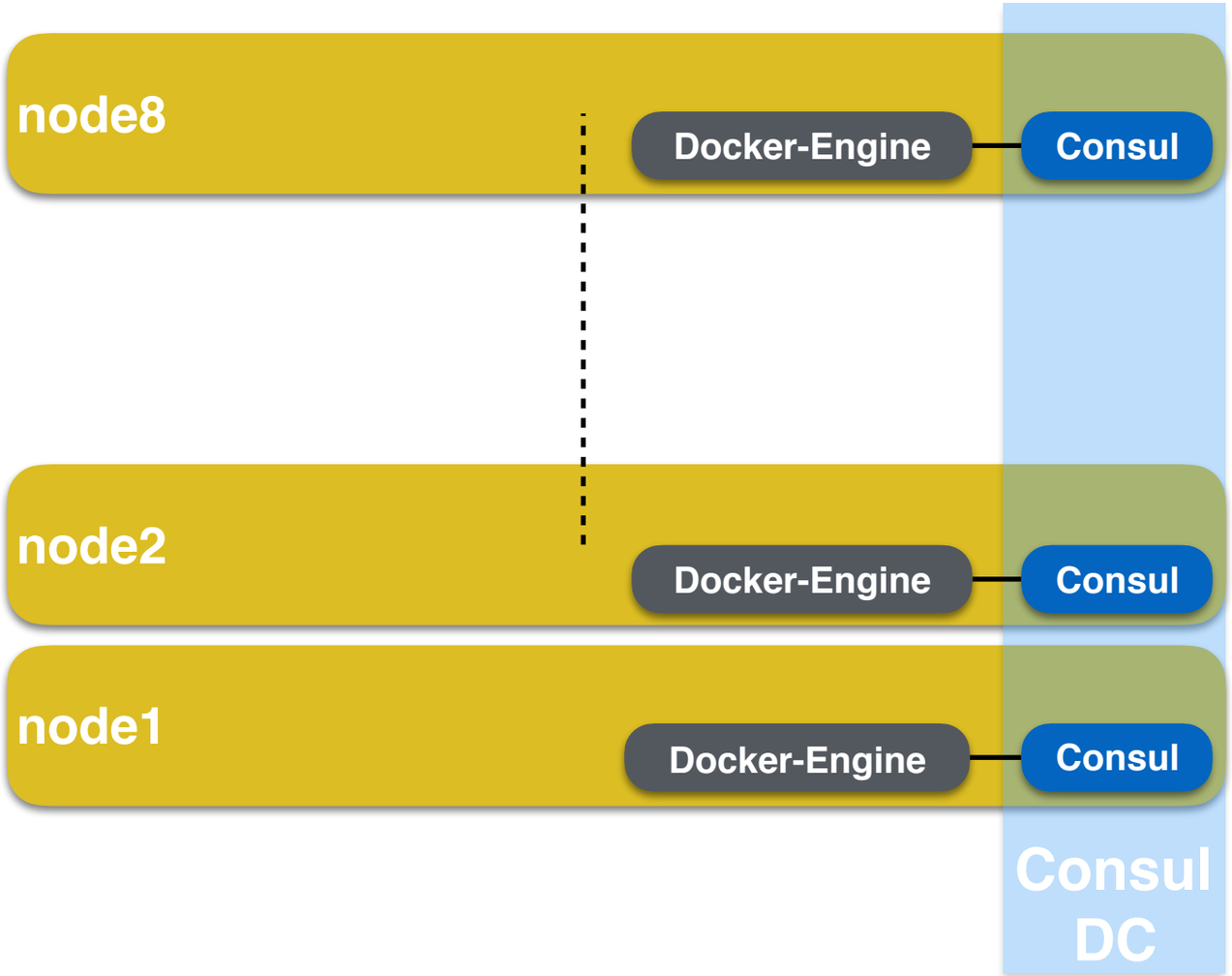
- 8x Sun Fire x2250, 2x 4core XEON, 32GB, Mellanox ConnectX-2)

Software

- Base installation
  - CentOS 7.2 base installation (updated from 7-alpha)
- Ansible
  - consul, sensu
  - docker v1.10, docker-compose
  - docker SWARM

Docker Networking

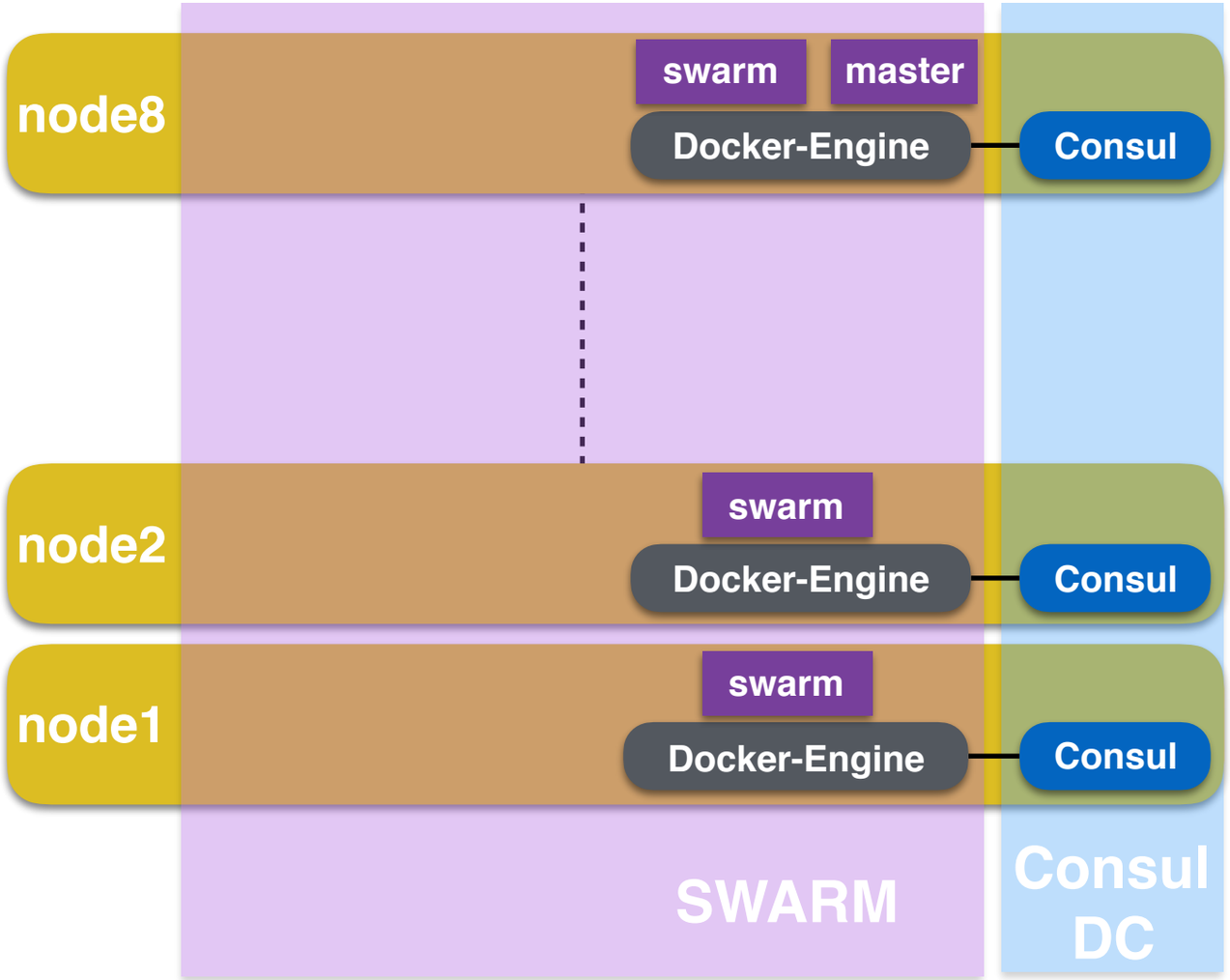
Synchronised by Consul



Docker SWARM

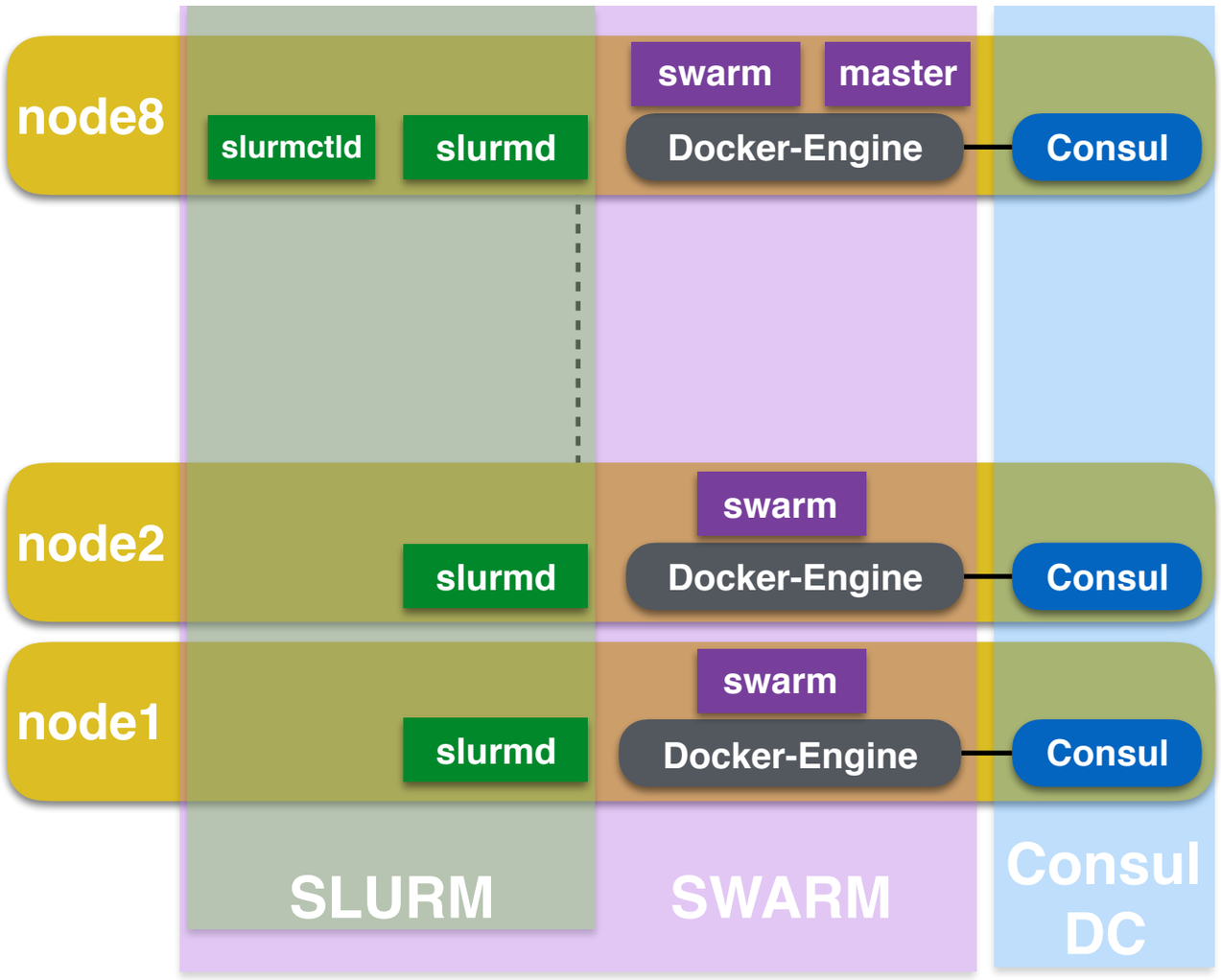
Docker SWARM

- Synchronised by Consul KV-store



SLURM Cluster

SLURM within SWARM



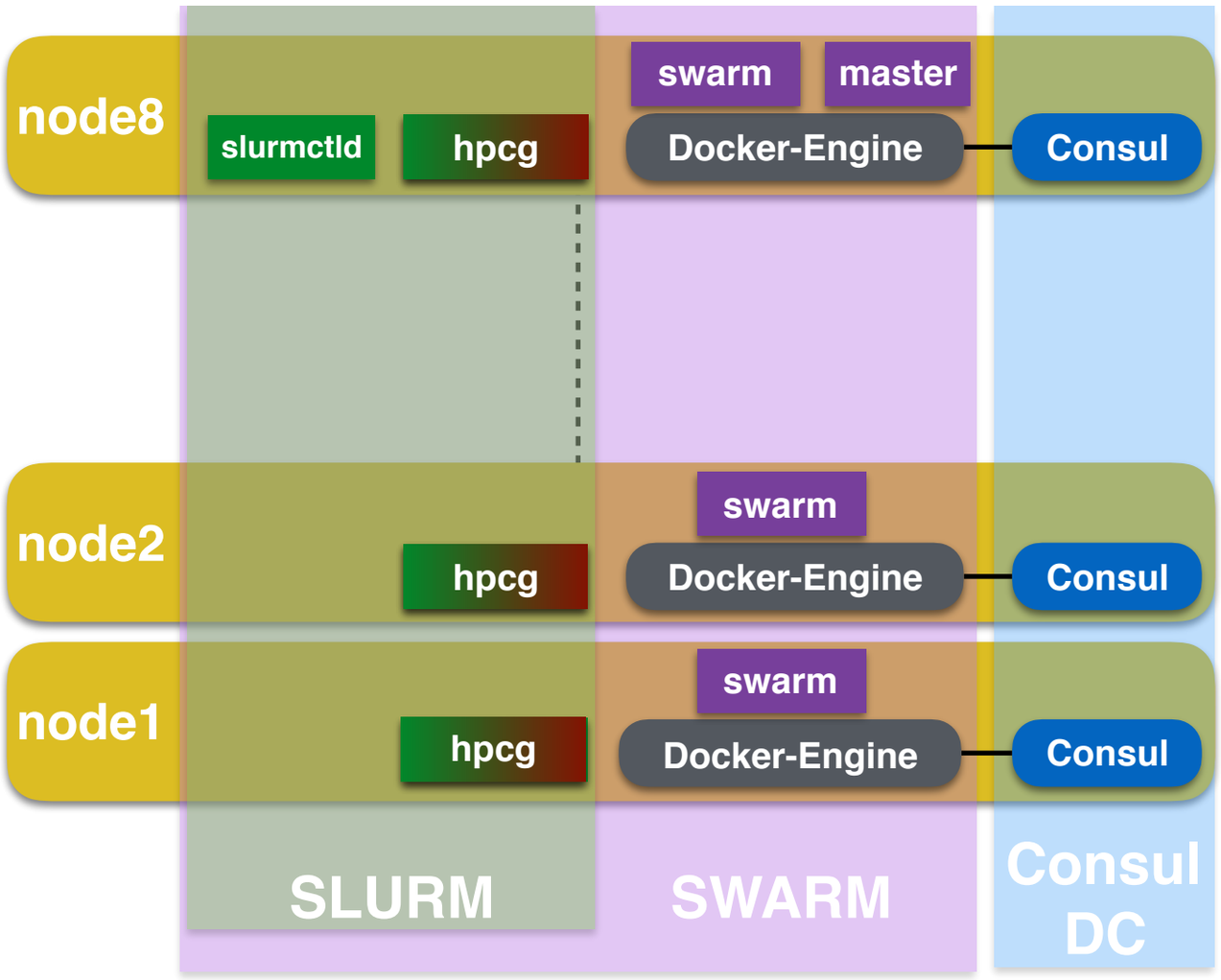
SLURM Cluster [cont]

```
venus001 rc=0 docker1:2375 provisioning (master) # docker exec -ti venus001/hpcg1 sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
all*         up   infinite      8   idle hpcg[1-8]
odd          up   infinite      4   idle hpcg[1,3,5,7]
even         up   infinite      4   idle hpcg[2,4,6,8]
venus001 rc=0 docker1:2375 provisioning (master) # docker exec -ti venus001/hpcg1 srun -N8 hostname
hpcg7
hpcg3
hpcg2
hpcg6
hpcg4
hpcg5
hpcg8
hpcg1
venus001 rc=0 docker1:2375 provisioning (master) #
```

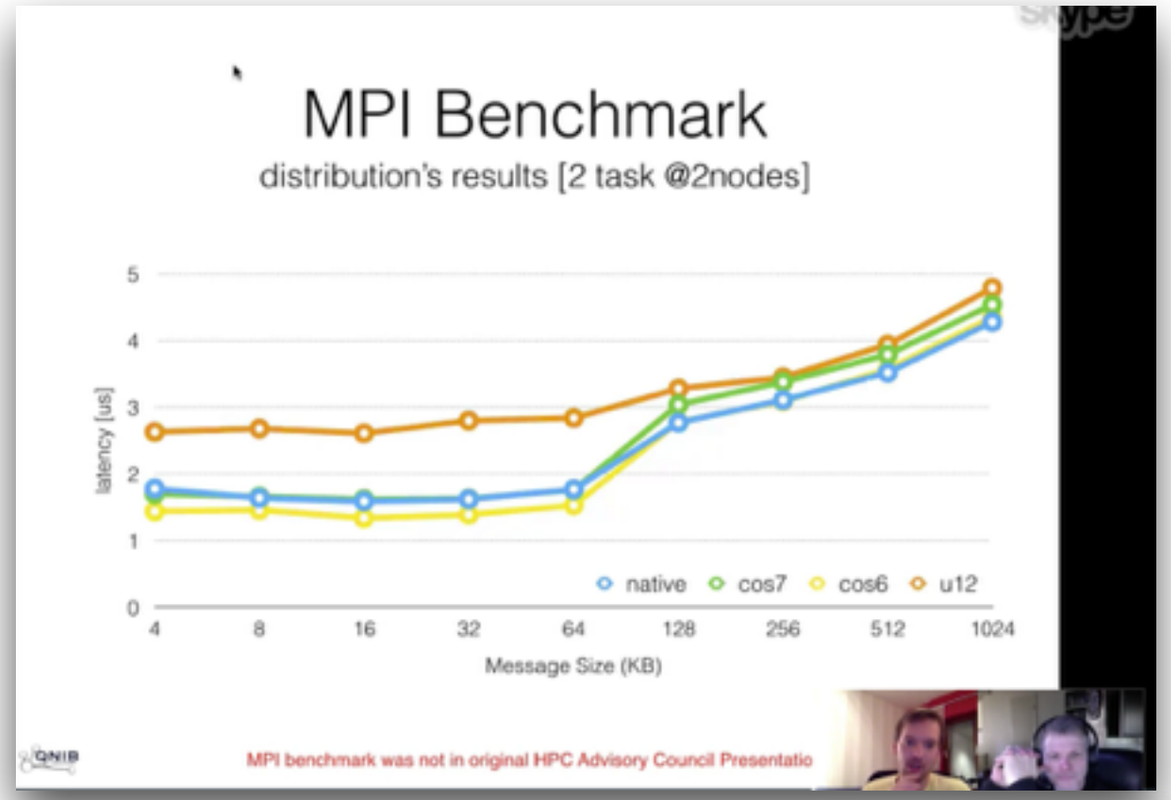
SLURM Cluster [cont]

SLURM within SWARM

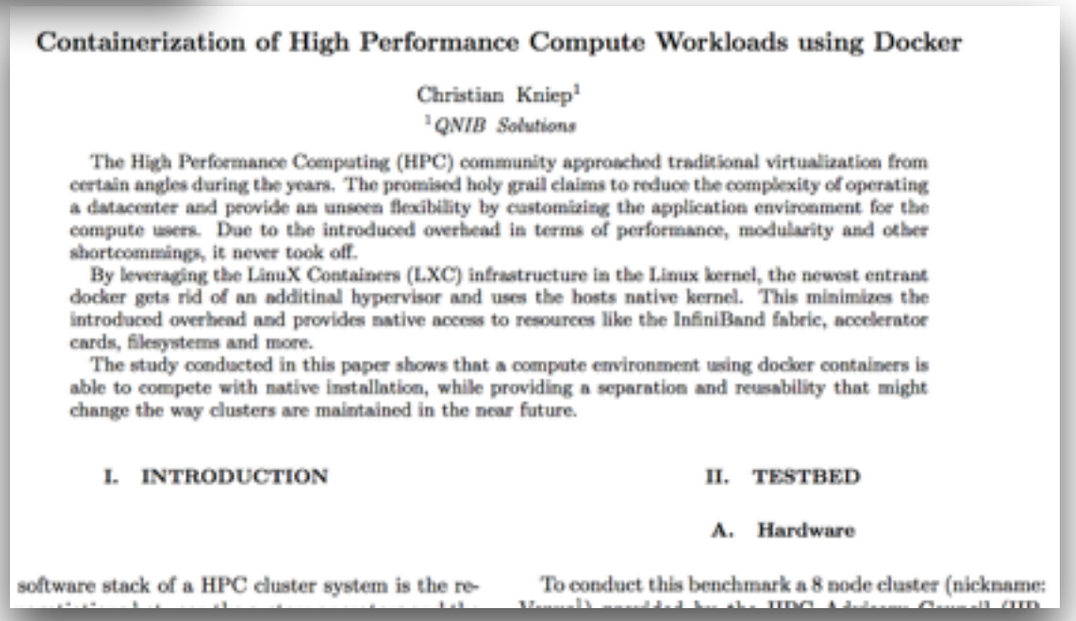
- slurmd within app-container
- pre-stage containers



MPI Benchmark



http://qnib.org/mpi



Containerization of High Performance Compute Workloads using Docker

Christian Kniep[1]
[1]*QNIB Solutions*

The High Performance Computing (HPC) community approached traditional virtualization from certain angles during the years. The promised holy grail claims to reduce the complexity of operating a datacenter and provide an unseen flexibility by customizing the application environment for the compute users. Due to the introduced overhead in terms of performance, modularity and other shortcommings, it never took off.

By leveraging the LinuX Containers (LXC) infrastructure in the Linux kernel, the newest entrant docker gets rid of an additinal hypervisor and uses the hosts native kernel. This minimizes the introduced overhead and provides native access to resources like the InfiniBand fabric, accelerator cards, filesystems and more.

The study conducted in this paper shows that a compute environment using docker containers is able to compete with native installation, while providing a separation and reusability that might change the way clusters are maintained in the near future.

I. INTRODUCTION

II. TESTBED

A. Hardware

software stack of a HPC cluster system is the re-

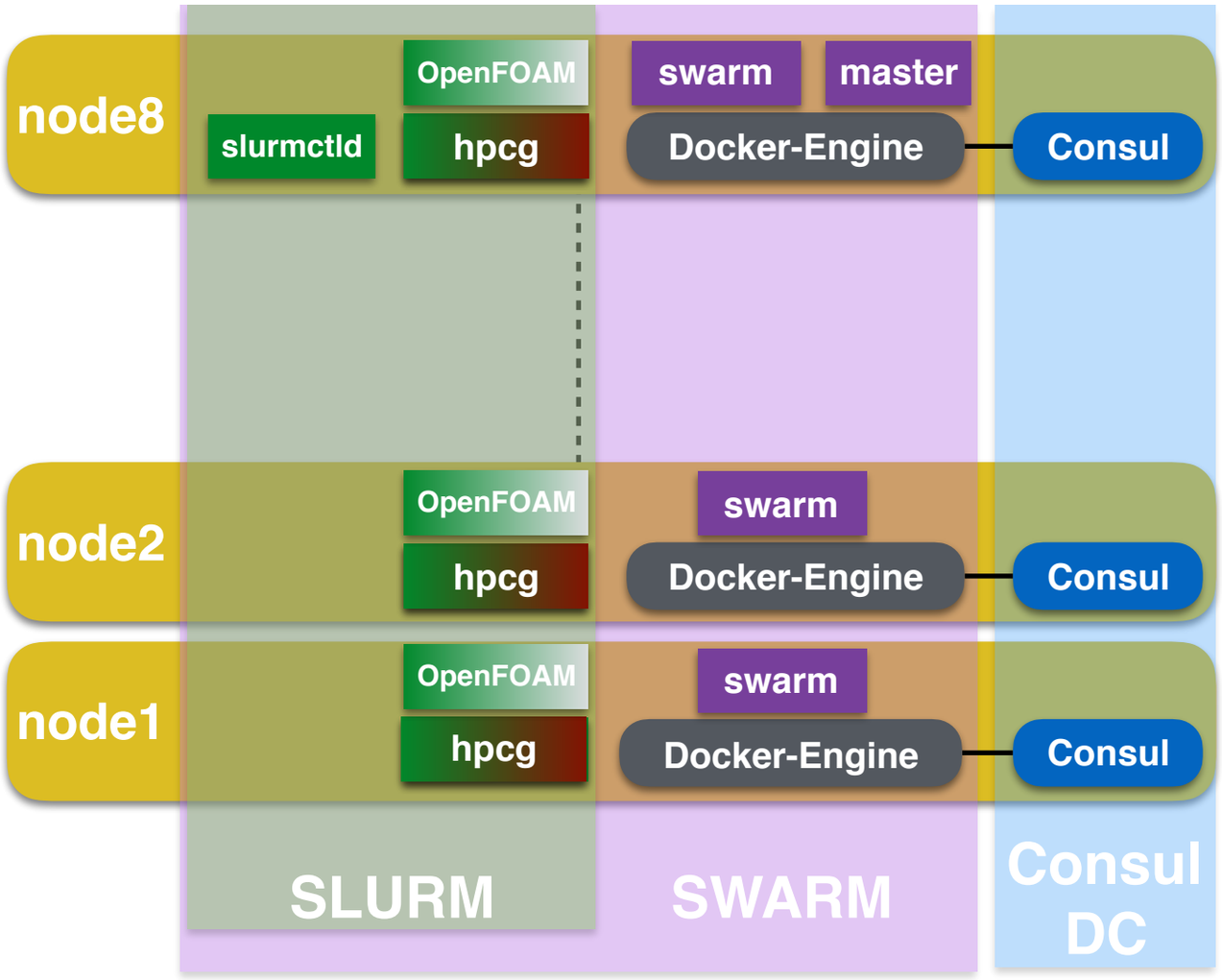
To conduct this benchmark a 8 node cluster (nickname:

http://doc.qnib.org/2014-11-05_Whitepaper_Docker-MPI-workload.pdf

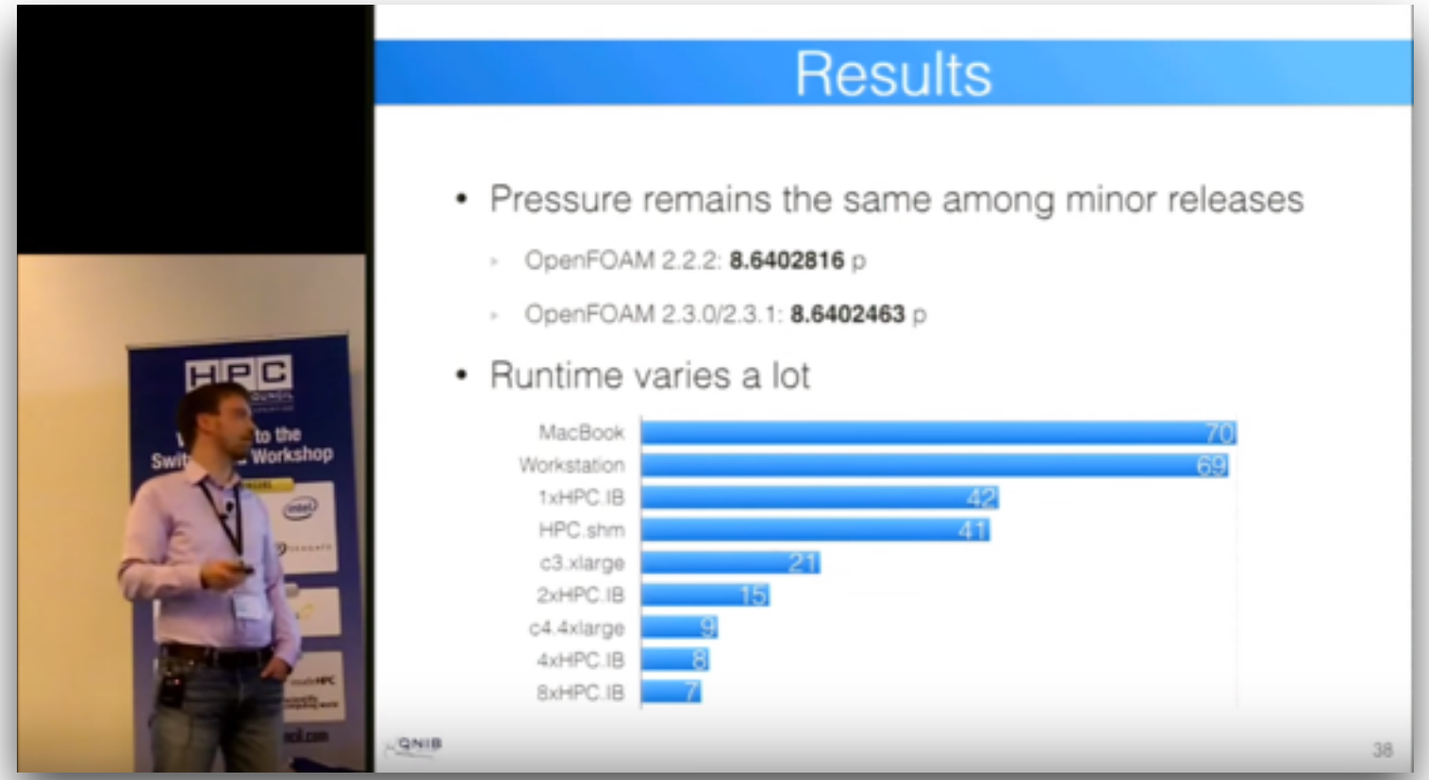
SLURM Cluster [cont]

SLURM within SWARM

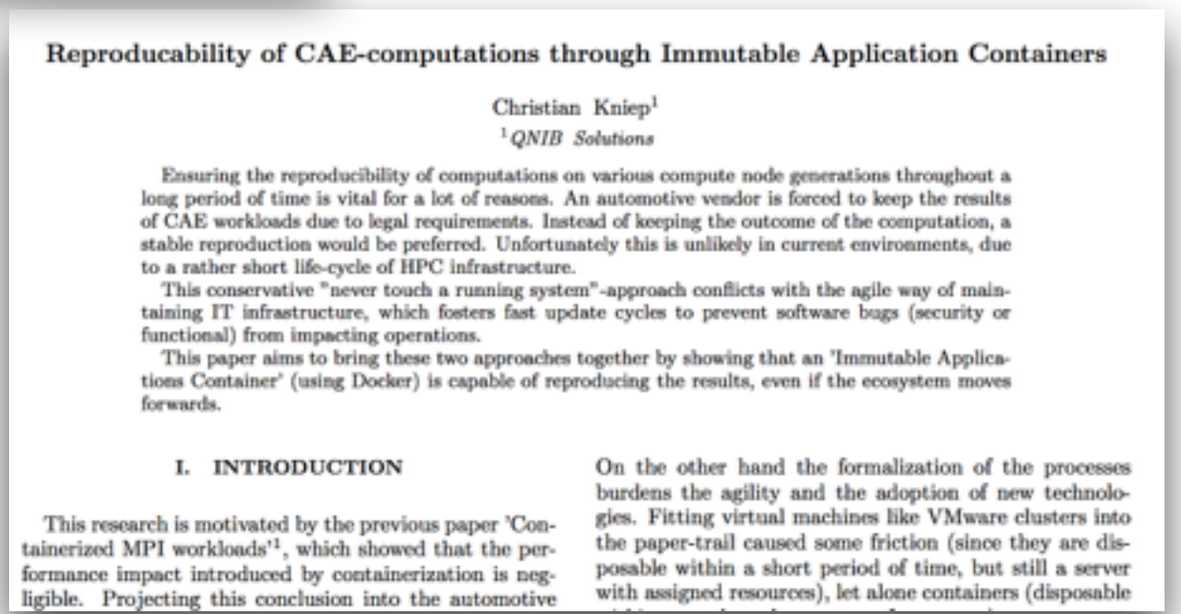
- slurmd within app-container
- pre-stage containers



OpenFOAM Benchmark



YouTube http://qnib.org/openfoam



Reproducability of CAE-computations through Immutable Application Containers

Christian Kniep[1]
[1] *QNIB Solutions*

Ensuring the reproducibility of computations on various compute node generations throughout a long period of time is vital for a lot of reasons. An automotive vendor is forced to keep the results of CAE workloads due to legal requirements. Instead of keeping the outcome of the computation, a stable reproduction would be preferred. Unfortunately this is unlikely in current environments, due to a rather short life-cycle of HPC infrastructure.

This conservative "never touch a running system"-approach conflicts with the agile way of maintaining IT infrastructure, which fosters fast update cycles to prevent software bugs (security or functional) from impacting operations.

This paper aims to bring these two approaches together by showing that an 'Immutable Applications Container' (using Docker) is capable of reproducing the results, even if the ecosystem moves forwards.

I. INTRODUCTION

This research is motivated by the previous paper 'Containerized MPI workloads'[1], which showed that the performance impact introduced by containerization is negligible. Projecting this conclusion into the automotive

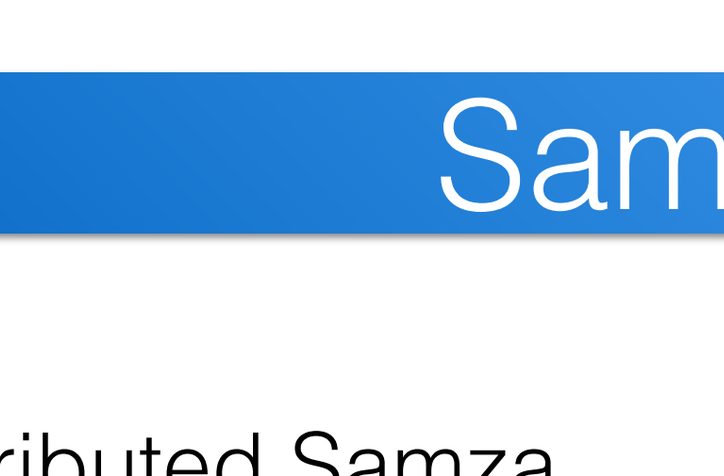
On the other hand the formalization of the processes burdens the agility and the adoption of new technologies. Fitting virtual machines like VMware clusters into the paper-trail caused some friction (since they are disposable within a short period of time, but still a server with assigned resources), let alone containers (disposable

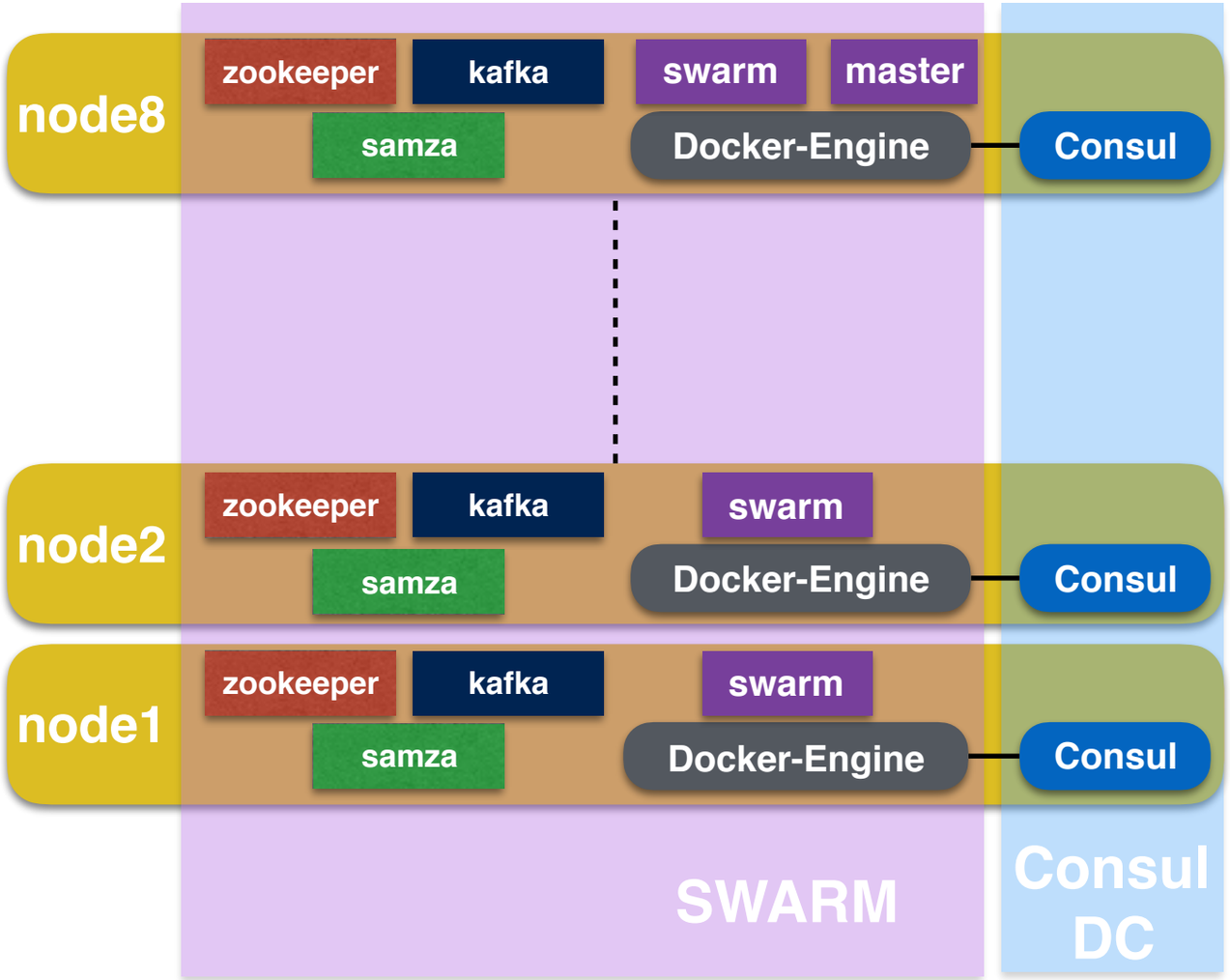
http://doc.qnib.org/2015-03-23_HPCAC_KniepChristian_ImmAppContainers.pdf

Samza Cluster

Distributed Samza

- Zookeeper and Kafka cluster
- Samza instances to run jobs






```
$ cat test.log |awk ‘{print $1}’ |sed -e ’s/HPC/BigData/g’ |tee out.log
```

To Be Discussed

Small vs. Big

1. Where to base images on?
   - Ubuntu/Fedora: ~200MB
   - Debian: ~100MB
   - Alpine Linux: 5MB (musl-libc)

2. Trimm the Images down at all cost?
   - How about debugging tools? Possibility to run tools on the host and ‘inspect’ namespaced processes inside of a container.
   - If PID-sharing arrives, carving out (e.g.) monitoring could be a thing.

One vs. Many Processes

1. In an ideal world…
   - a container only runs one process, e.g. the HPC solver.
2. In reality…
   - MPI want's to connect to a sshd within the job-peers
   - monitoring, syslog, service discovery should be present as well.
3. How fast / aggressive to break traditional approaches?

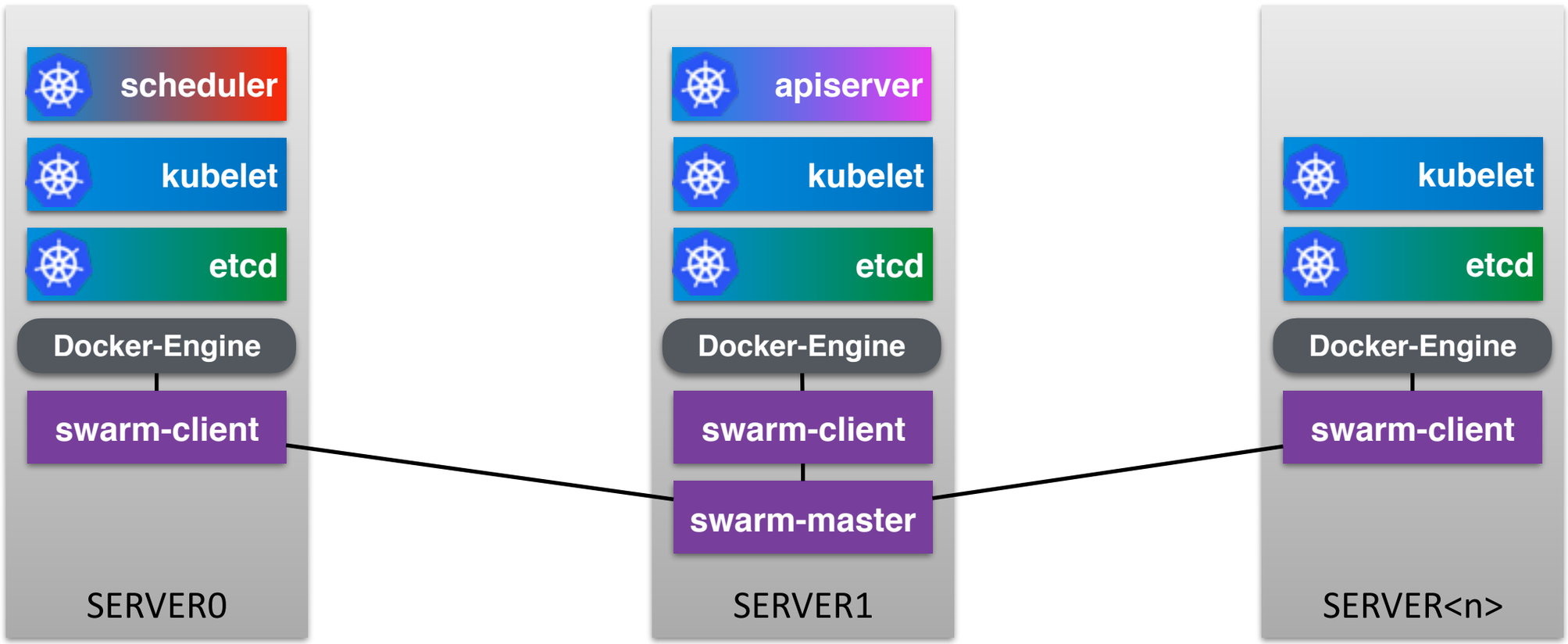
Service Discovery

1. Since the environments are rather dynamic…
   - how does the containers discover services?
   - external registry as part of the framework?
   - discovery service as part of the container stacks?

Orchestration Frameworks

With Docker Swarm it is rather easy

- to spin up a Kubernetes or Mesos cluster within Swarm.



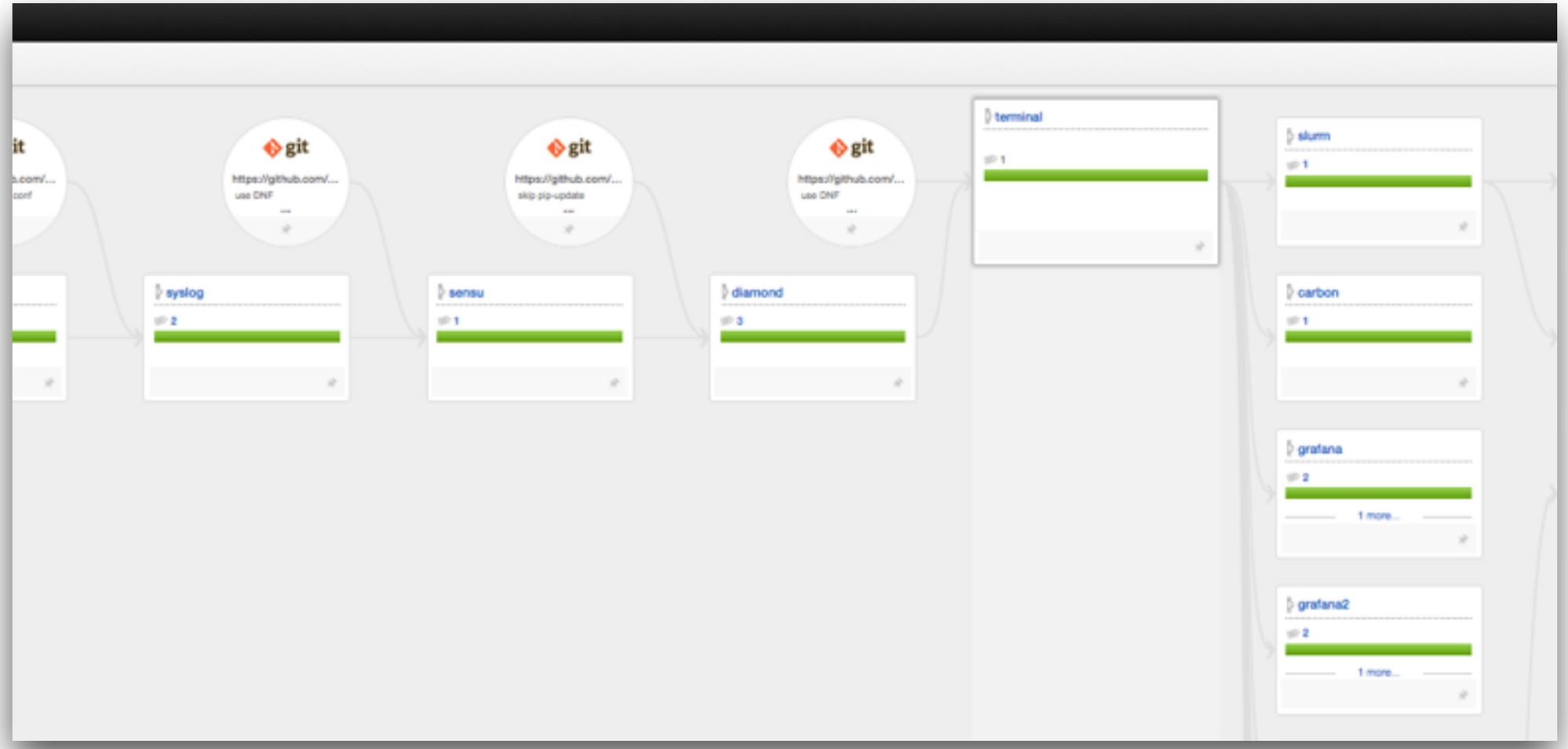
Immutable vs. Config Mgmt

1. Containers should be controlled via ENV or flags
   - External access/change of a running container is discouraged
2. Configuration management
   - Downgraded to bootstrap a host?

Continuous Dev./Integration

If containers are immutable within pipeline

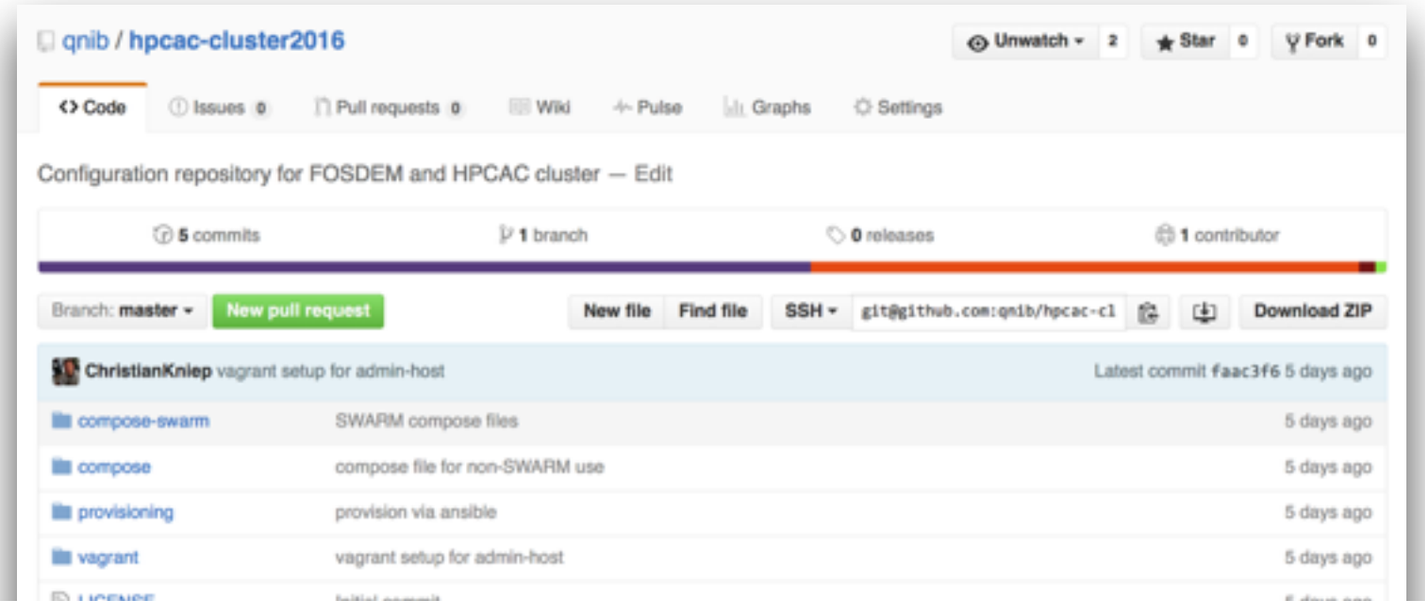
- testing/deployment should be automated
- developers should have a production replica



Recap aka. IMHO

1. Using vanilla docker tech on-top of any distribution
   - keep up with the ecosystem and prevent vendor/ecosystem lock-in
2. 80/20 rule
   - have caveats on the radar but don’t bother too much
   - everything is so fast moving - it’s hard to predict
3. Don’t scare away stakeholders
   - KISS
   - reuse workflow and infrastructure

23.06.2016



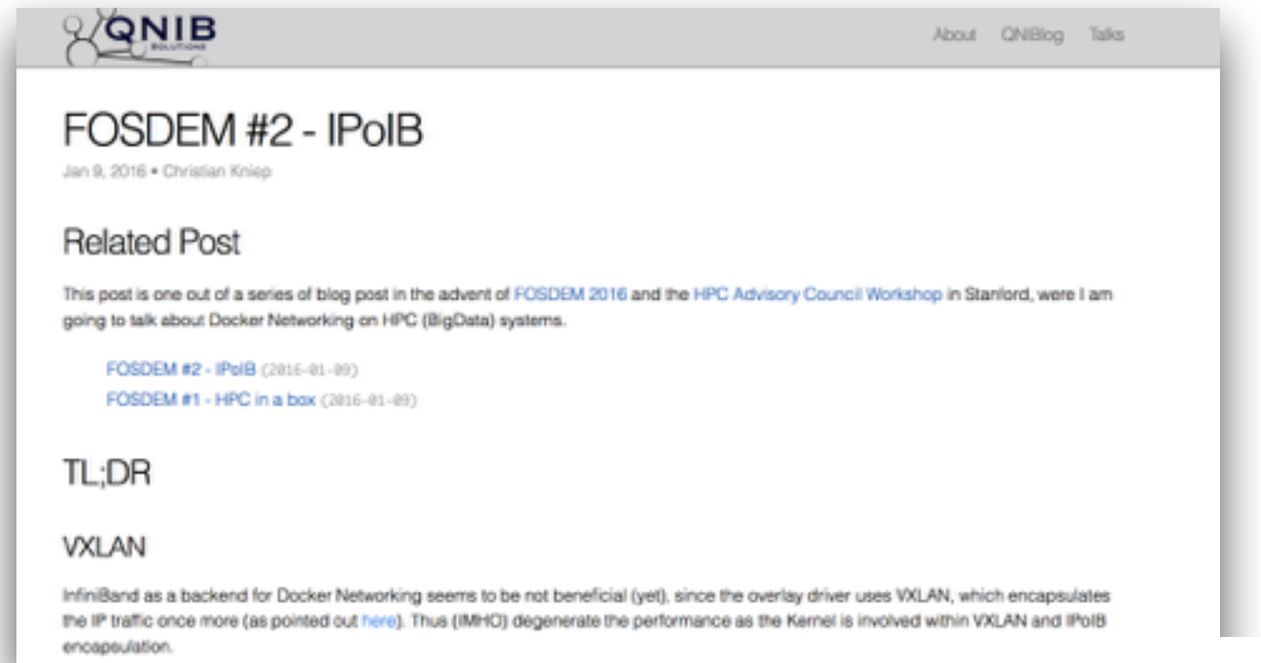
https://github.com/qnib/hpcac-cluster2016

Q&A

HPC ADVISORY COUNCIL

HPCAC Lugano:
March 21st - 23rd

eGalea Conference (Pisa)
20/21 April 2016



http://qnib.org