Multi-host containerised HPC cluster

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

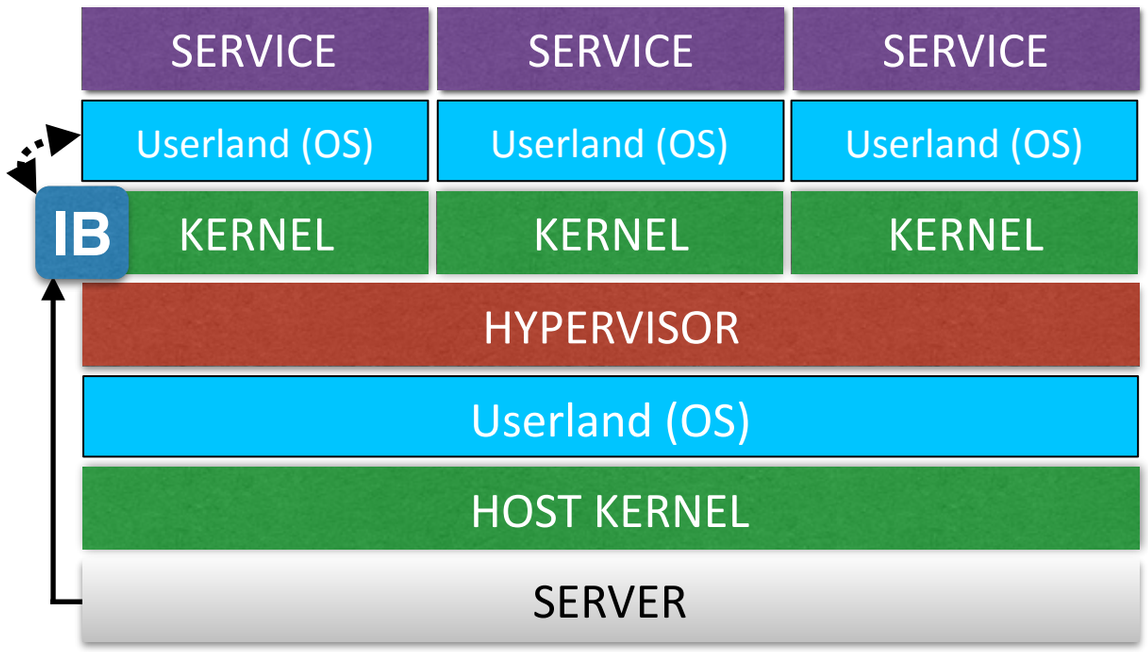
QNIB

FOSDEM '16 Brussels 30 & 31 January 2016

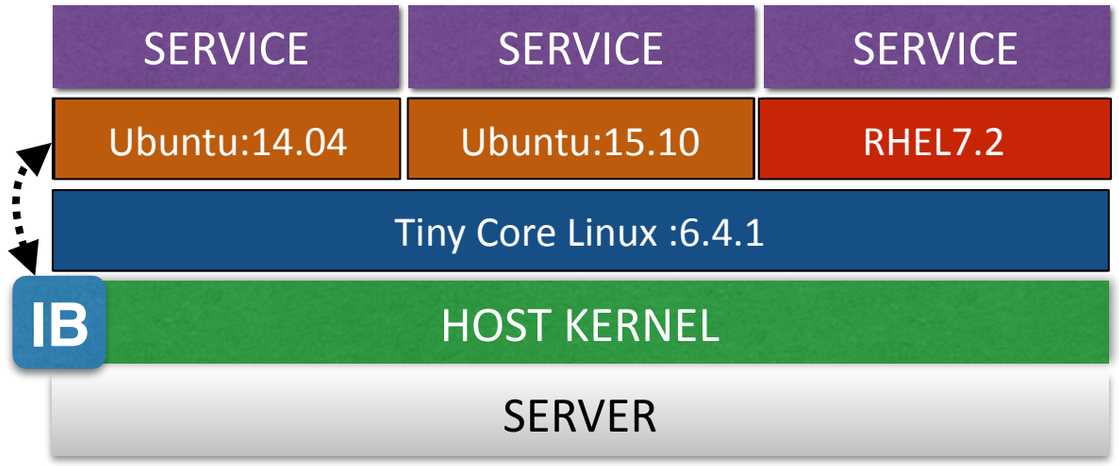
The Bits and Pieces…

Docker

- Containers do not spin up their own kernel
  - All containers share the same host kernel
  - they are separated via Kernel Namespaces
  - constrained using CGroups



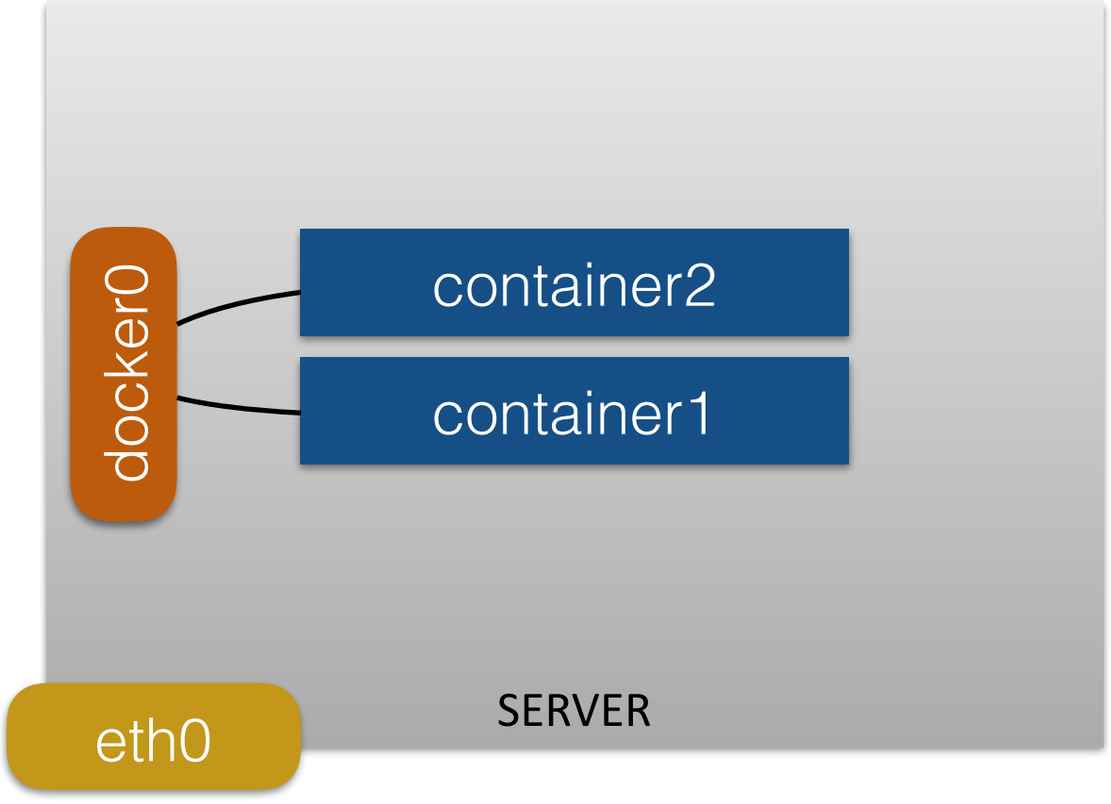
Traditional Virtualisation



Containerisation

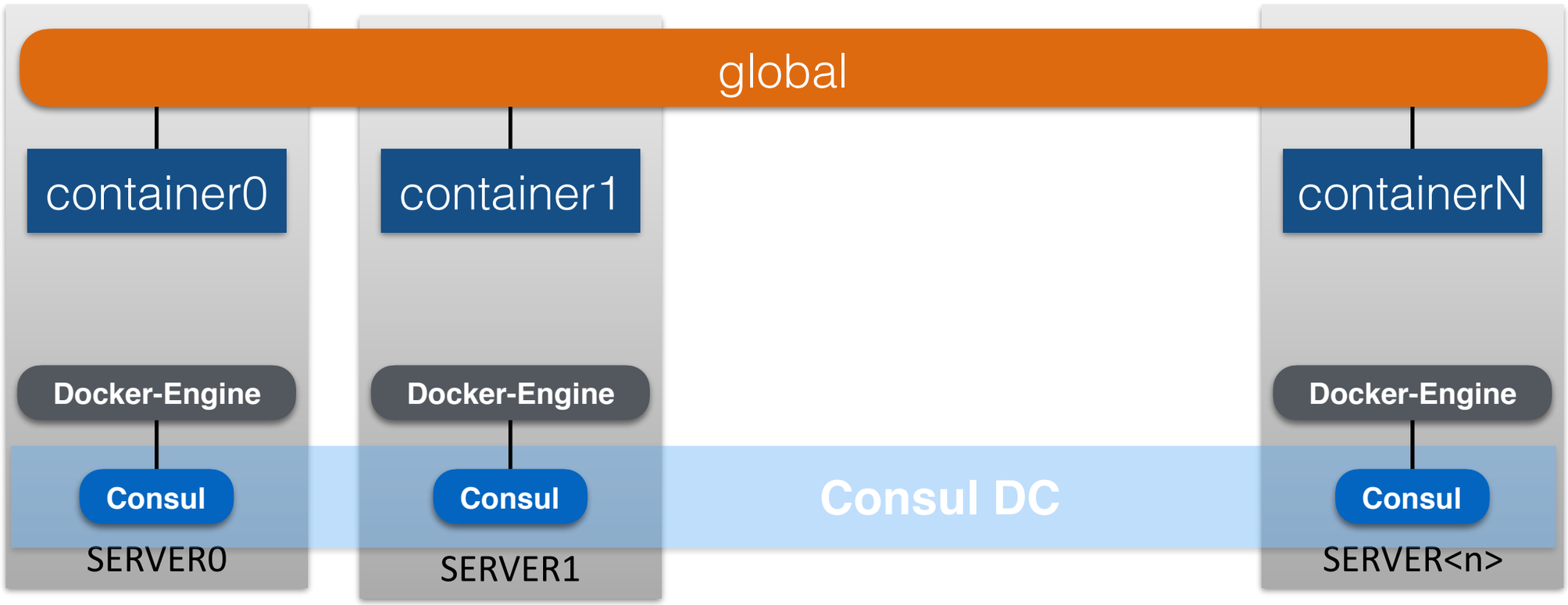
Docker

- Docker as a Container Runtime
  - creates/starts/stops/manipulates containers; all through RESTful API
  - handles IP connectivity, bind-mounts, etc.



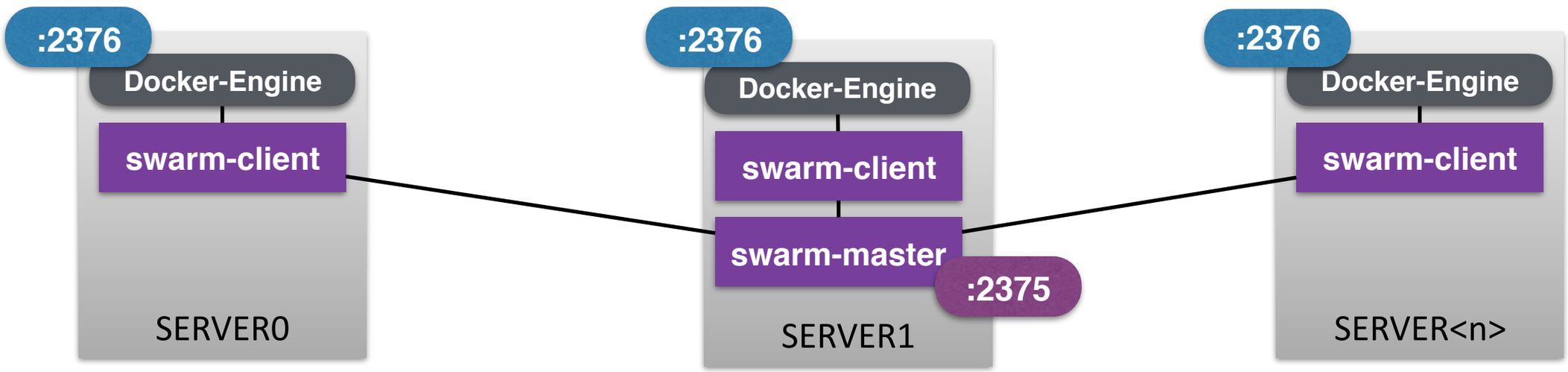
Docker Networking

- Docker Networking spans ‘bridges’ across hosts
  - KV-store to synchronise (Zookeeper, etcd, Consul)
  - VXLAN to pass messages along (macvlan might take over)



Docker Swarm

- Docker Swarm proxies docker-engines
  - serves an API endpoint in front of multiple docker-engines
  - does (simple) placement decisions.



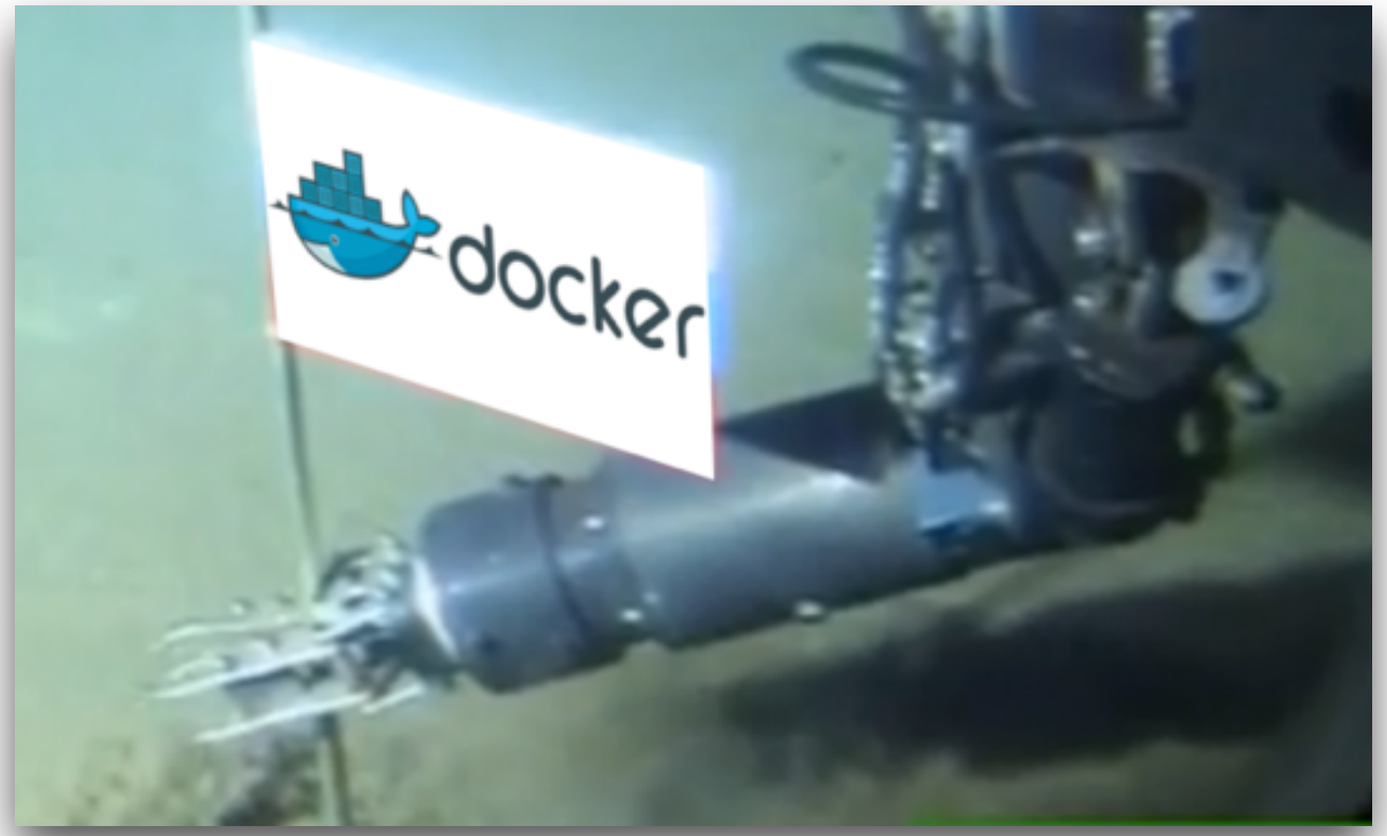
Docker Swarm [cont]

```
[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



credit: Russian Oil Claim

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

kubernetes

Solutions

self-healing

production-ready

enterprise-grade

Pitfalls

[incomplete list]

- You say Docker - perception is ‘basically VMware’
  - VMs were easy to shoehorn in legacy workflow, containers might break it
  - spans environments: laptop, dev-cluster, staging, prod
- Not everyone is a unicorn, I do
  - NOT want special distributions
    - useful for elasticity (AWS) and green-flied deployment, not so much for a on-premise datacenter w/ legacy in it.
  - want to leverage existing <stuff>
    - security (ssh infrastructure), user authentication
    - installation workflow, monitoring, logging
  - want to keep up with upstream docker ecosystem
    - networking, volumes
    - features of docker-engine, -compose, -swarm

Reduce to the max!

Put a Docker **on** it!

- Leverage existing install/configuration workflow
  - Kickstart + ansible-run
- Don’t focus on corner cases
  - postpone snowflake-container (e.g. needs multi-tenant GPU usage)
  - User namespace: we get there eventually
- HPC environments assumptions
  - single-tenant
  - focus on performance

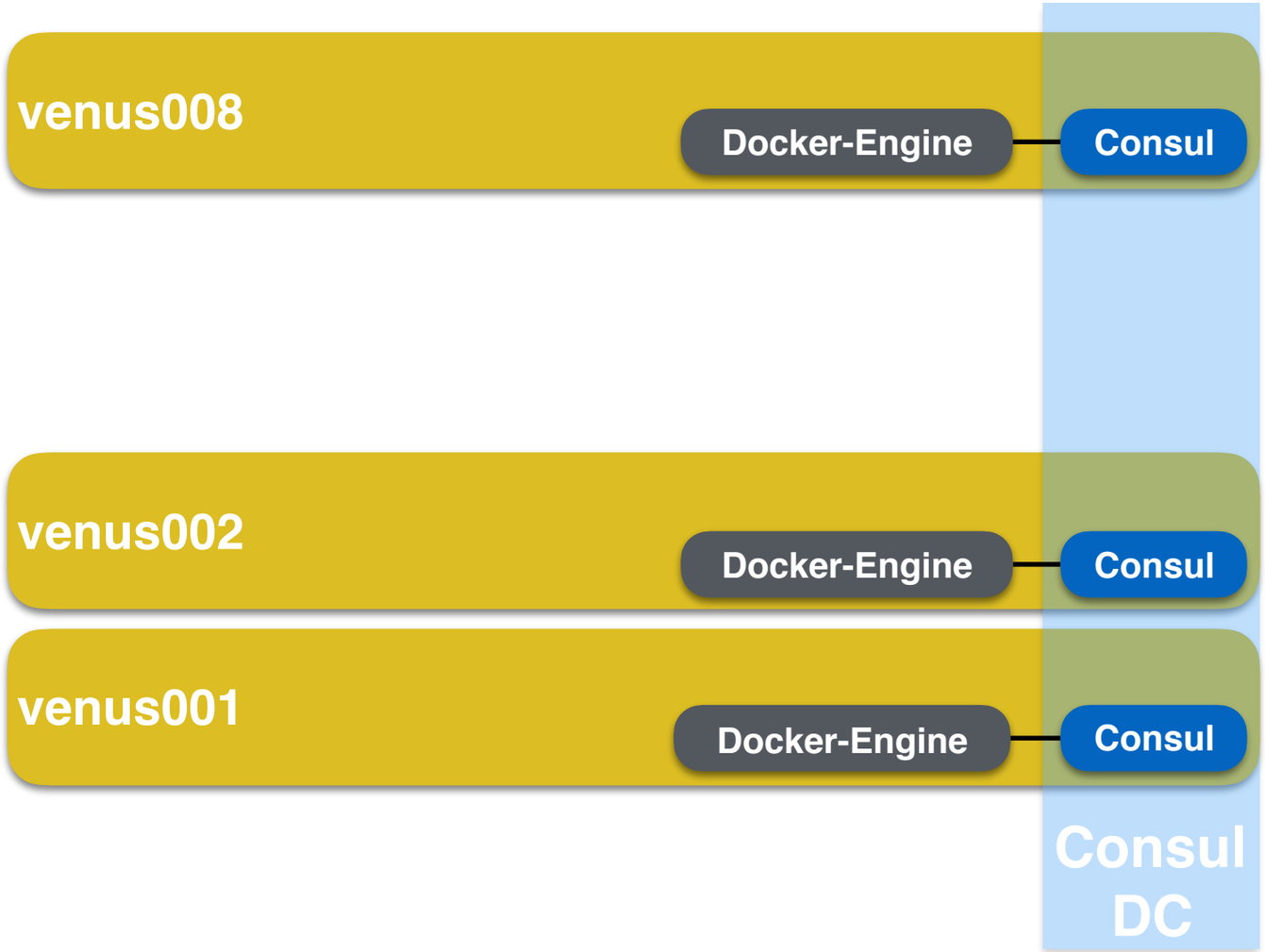
The Setup

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 v1.9.1, docker-compose 0.6.0dev
    - docker SWARM

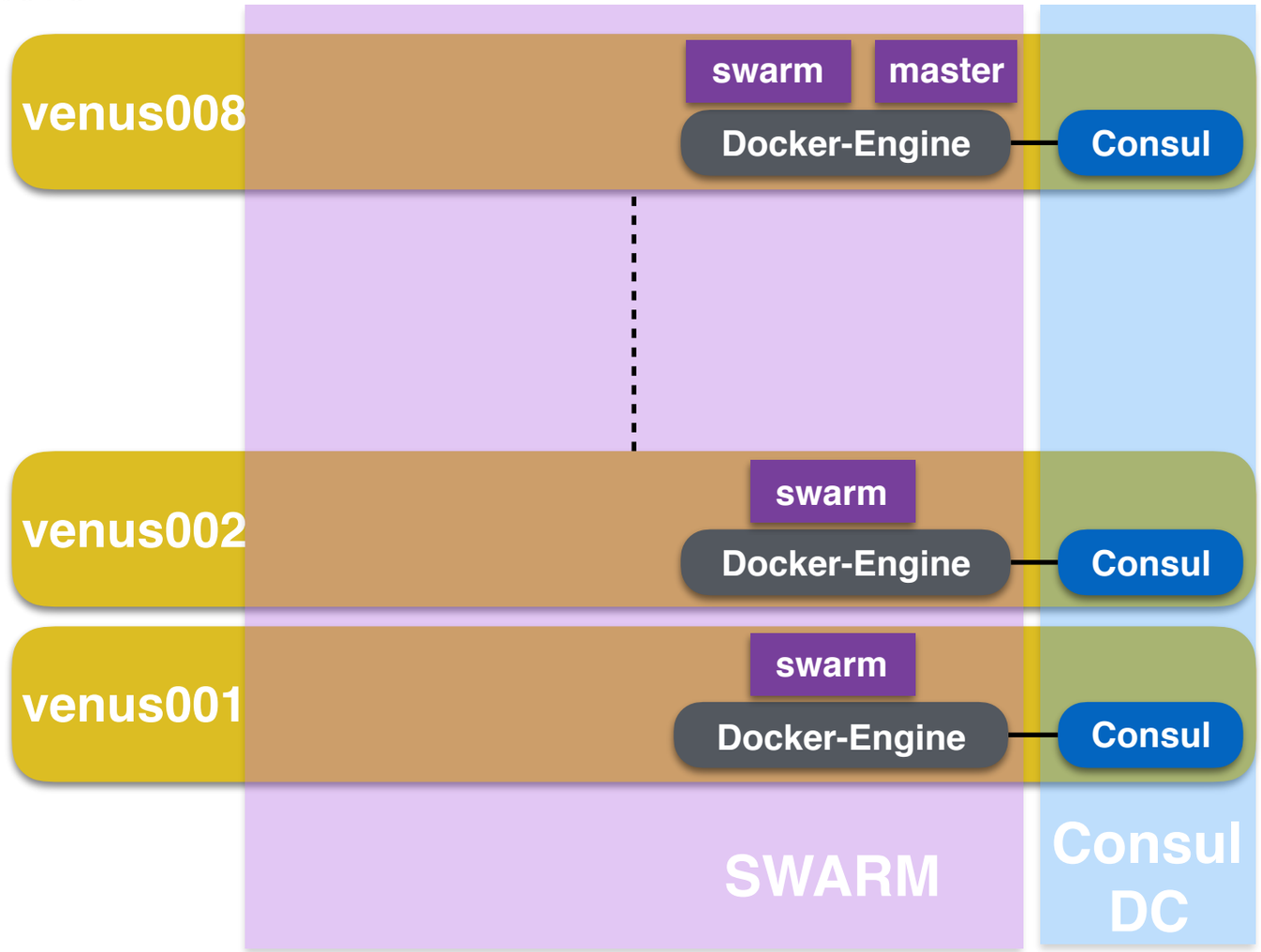
Docker Networking

- Synchronised by Consul



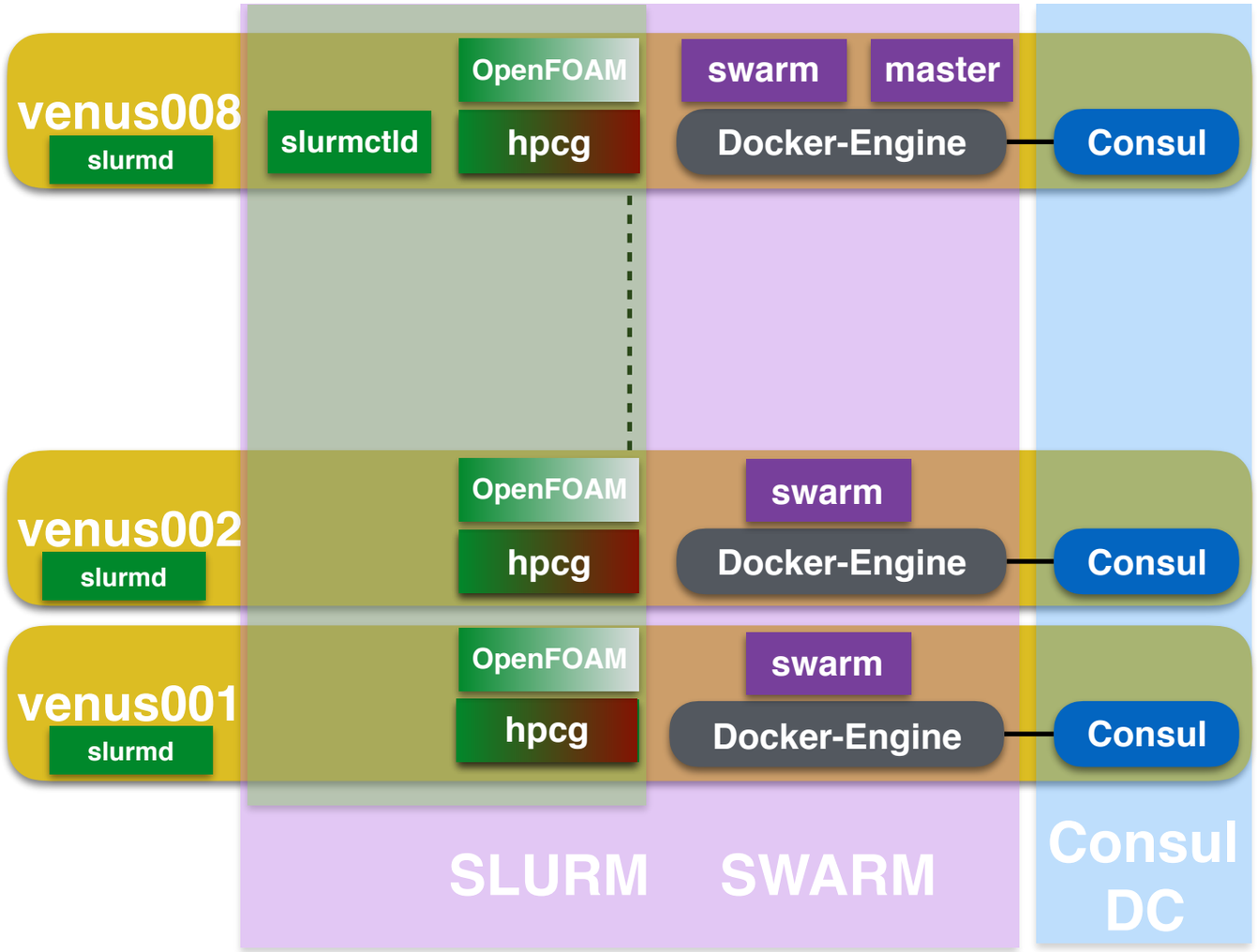
Docker SWARM

- Docker SWARM
  - Synchronised by Consul KV-store



SLURM Cluster

- SLURM within SWARM
  - slurmd within app-container
  - pre-stage multiple container
  - spawn at job-start (pre-hook)

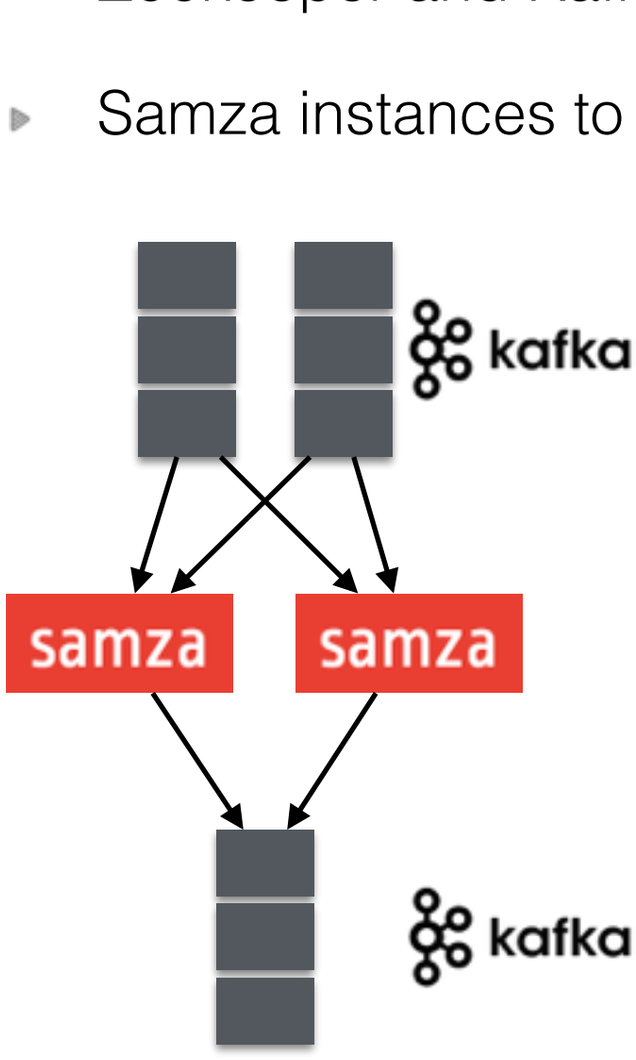


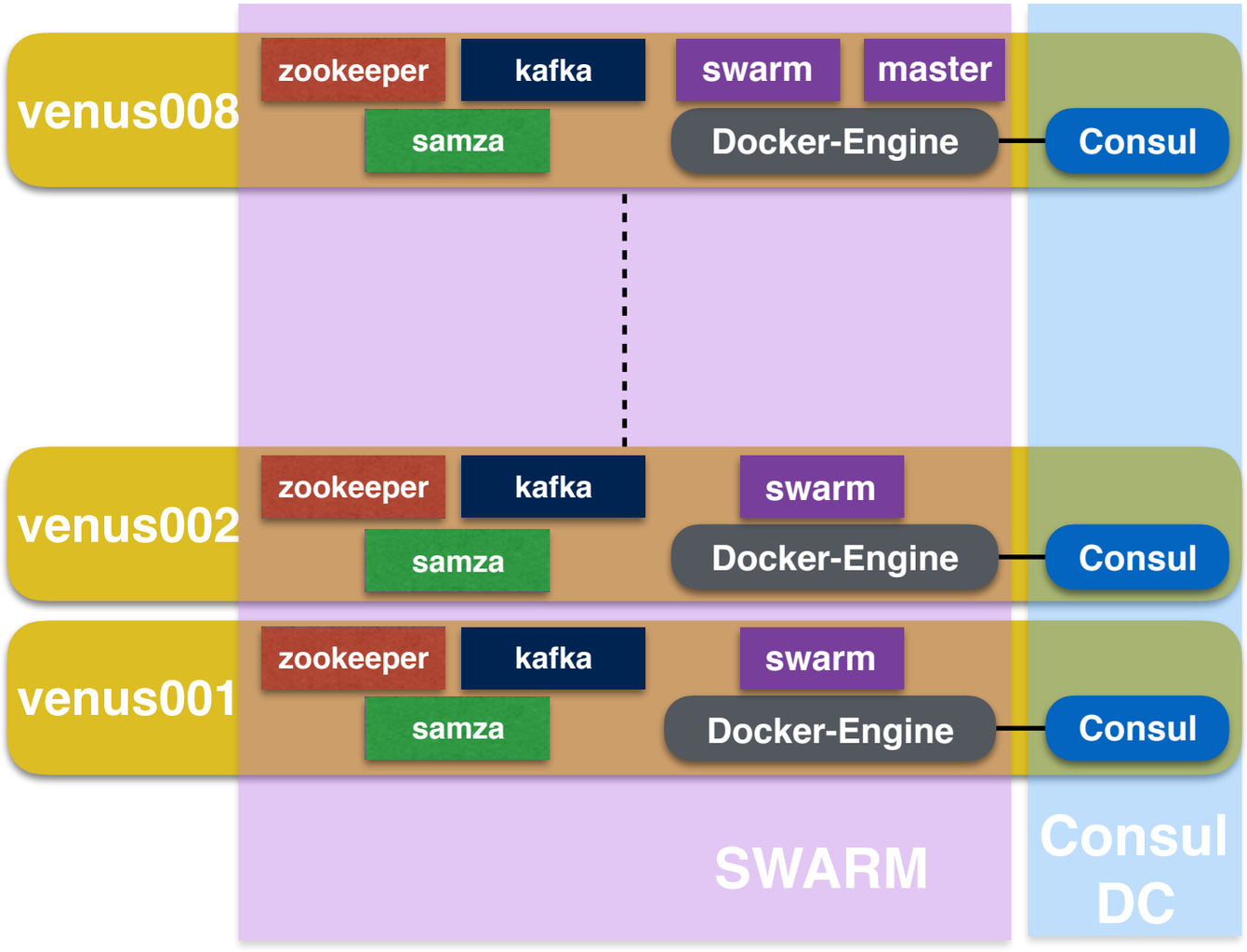
SLURM Cluster [cont]

```
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) #
```

Samza Cluster

- Distributed Samza
  - Zookeeper and Kafka cluster
  - Samza instances to run jobs



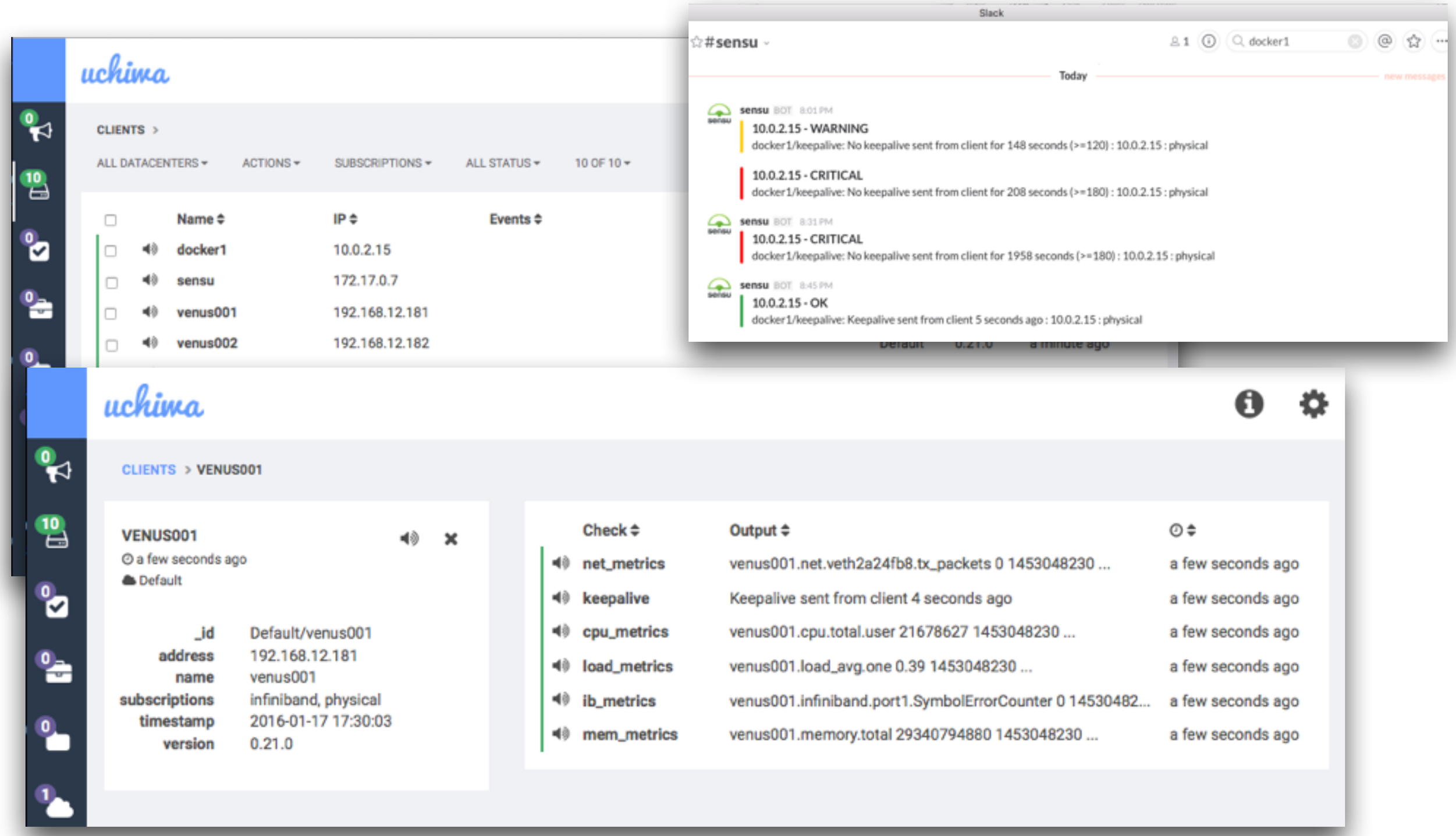


Outta time!

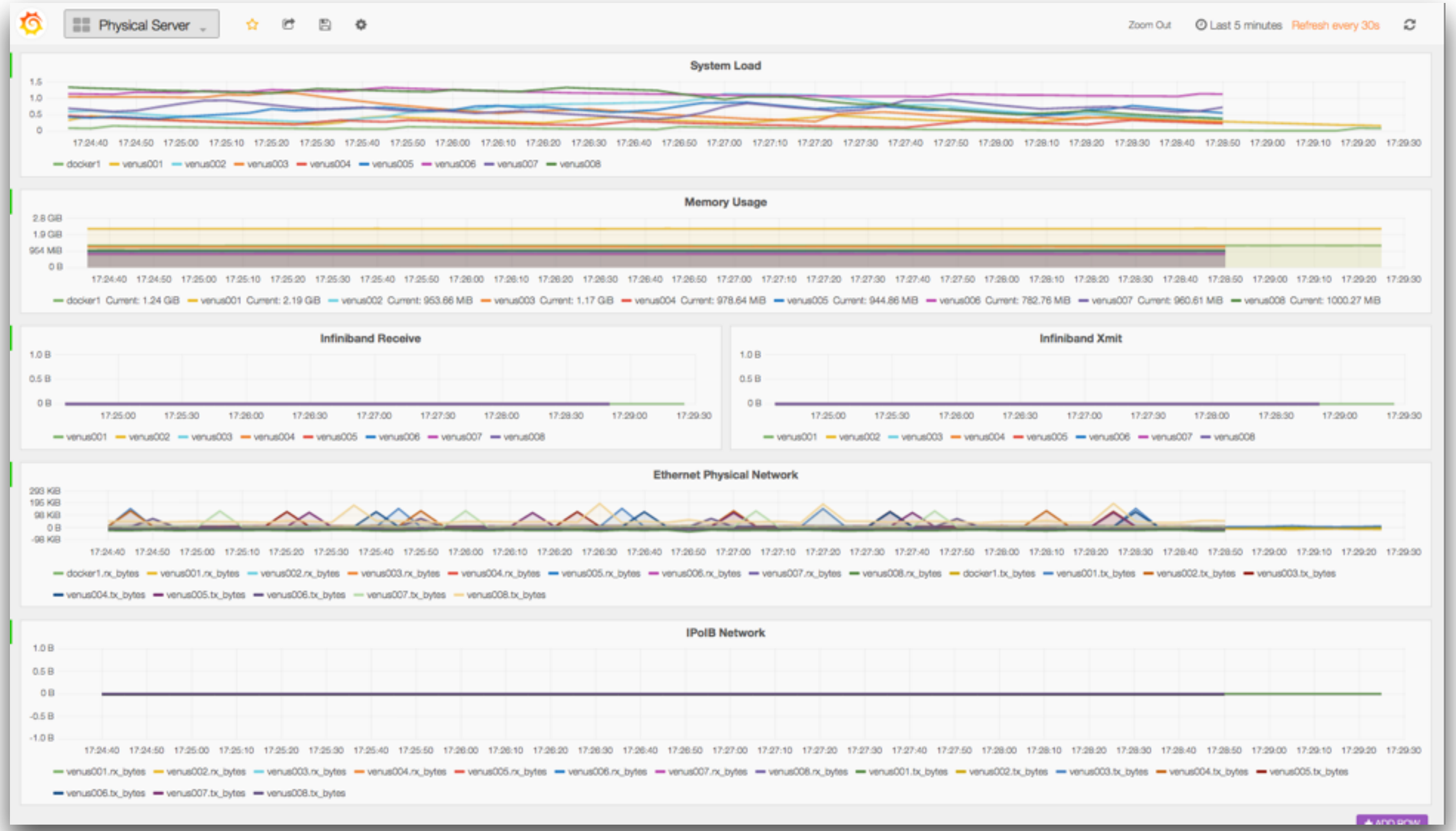
QNIB

FOSDEM '16 Brussels 30 & 31 January 2016

SENSU


uchiwa
CLIENTS >
ALL DATACENTERS
ACTIONS
SUBSCRIPTIONS
ALL STATUS
10 OF 10
Name
IP
Events
docker1 10.0.2.15
sensu 172.17.0.7
venus001 192.168.12.181
venus002 192.168.12.182
Slack
#sensu
docker1
Today
new messages
sensu BOT 8:01 PM
10.0.2.15 - WARNING
docker1/keepalive: No keepalive sent from client for 148 seconds (>=120) : 10.0.2.15 : physical
10.0.2.15 - CRITICAL
docker1/keepalive: No keepalive sent from client for 208 seconds (>=180) : 10.0.2.15 : physical
sensu BOT 8:31 PM
10.0.2.15 - CRITICAL
docker1/keepalive: No keepalive sent from client for 1958 seconds (>=180) : 10.0.2.15 : physical
sensu BOT 8:45 PM
10.0.2.15 - OK
docker1/keepalive: Keepalive sent from client 5 seconds ago : 10.0.2.15 : physical
uchiwa
CLIENTS > VENUS001
VENUS001
a few seconds ago
Default
_id Default/venus001
address 192.168.12.181
name venus001
subscriptions infiniband, physical
timestamp 2016-01-17 17:30:03
version 0.21.0
Check
Output
net_metrics venus001.net.veth2a24fb8.tx_packets 0 1453048230 ... a few seconds ago
keepalive Keepalive sent from client 4 seconds ago a few seconds ago
cpu_metrics venus001.cpu.total.user 21678627 1453048230 ... a few seconds ago
load_metrics venus001.load_avg.one 0.39 1453048230 ... a few seconds ago
ib_metrics venus001.infiniband.port1.SymbolErrorCounter 0 14530482... a few seconds ago
mem_metrics venus001.memory.total 29340794880 1453048230 ... a few seconds ago

Metrics


Physical Server
Zoom Out
Last 5 minutes
Refresh every 30s
System Load
1.5
1.0
0.5
0
17:24:40 17:24:50 17:25:00 17:25:10 17:25:20 17:25:30 17:25:40 17:25:50 17:26:00 17:26:10 17:26:20 17:26:30 17:26:40 17:26:50 17:27:00 17:27:10 17:27:20 17:27:30 17:27:40 17:27:50 17:28:00 17:28:10 17:28:20 17:28:30 17:28:40 17:28:50 17:29:00 17:29:10 17:29:20 17:29:30
docker1 venus001 venus002 venus003 venus004 venus005 venus006 venus007 venus008
Memory Usage
2.8 GiB
1.9 GiB
964 MiB
0 B
17:24:40 17:24:50 17:25:00 17:25:10 17:25:20 17:25:30 17:25:40 17:25:50 17:26:00 17:26:10 17:26:20 17:26:30 17:26:40 17:26:50 17:27:00 17:27:10 17:27:20 17:27:30 17:27:40 17:27:50 17:28:00 17:28:10 17:28:20 17:28:30 17:28:40 17:28:50 17:29:00 17:29:10 17:29:20 17:29:30
docker1 Current: 1.24 GiB venus001 Current: 2.19 GiB venus002 Current: 953.66 MiB venus003 Current: 1.17 GiB venus004 Current: 978.64 MiB venus005 Current: 944.86 MiB venus006 Current: 782.76 MiB venus007 Current: 960.61 MiB venus008 Current: 1000.27 MiB
Infiniband Receive
1.0 B
0.5 B
0 B
17:25:00 17:25:30 17:26:00 17:26:30 17:27:00 17:27:30 17:28:00 17:28:30 17:29:00 17:29:30
venus001 venus002 venus003 venus004 venus005 venus006 venus007 venus008
Infiniband Xmit
1.0 B
0.5 B
0 B
17:25:00 17:25:30 17:26:00 17:26:30 17:27:00 17:27:30 17:28:00 17:28:30 17:29:00 17:29:30
venus001 venus002 venus003 venus004 venus005 venus006 venus007 venus008
Ethernet Physical Network
293 KiB
195 KiB
98 KiB
0 B
-98 KiB
17:24:40 17:24:50 17:25:00 17:25:10 17:25:20 17:25:30 17:25:40 17:25:50 17:26:00 17:26:10 17:26:20 17:26:30 17:26:40 17:26:50 17:27:00 17:27:10 17:27:20 17:27:30 17:27:40 17:27:50 17:28:00 17:28:10 17:28:20 17:28:30 17:28:40 17:28:50 17:29:00 17:29:10 17:29:20 17:29:30
docker1.rx_bytes venus001.rx_bytes venus002.rx_bytes venus003.rx_bytes venus004.rx_bytes venus005.rx_bytes venus006.rx_bytes venus007.rx_bytes venus008.rx_bytes docker1.tx_bytes venus001.tx_bytes venus002.tx_bytes venus003.tx_bytes
venus004.tx_bytes venus005.tx_bytes venus006.tx_bytes venus007.tx_bytes venus008.tx_bytes
IPoIB Network
1.0 B
0.5 B
0 B
-0.5 B
-1.0 B
17:24:40 17:24:50 17:25:00 17:25:10 17:25:20 17:25:30 17:25:40 17:25:50 17:26:00 17:26:10 17:26:20 17:26:30 17:26:40 17:26:50 17:27:00 17:27:10 17:27:20 17:27:30 17:27:40 17:27:50 17:28:00 17:28:10 17:28:20 17:28:30 17:28:40 17:28:50 17:29:00 17:29:10 17:29:20 17:29:30
venus001.rx_bytes venus002.rx_bytes venus003.rx_bytes venus004.rx_bytes venus005.rx_bytes venus006.rx_bytes venus007.rx_bytes venus008.rx_bytes venus001.tx_bytes venus002.tx_bytes venus003.tx_bytes venus004.tx_bytes venus005.tx_bytes
venus006.tx_bytes venus007.tx_bytes venus008.tx_bytes

Recap aka. IMHO

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

Q&A