dCache.org

Using containiers to manage dCache

Tigran Mkrtchyan for dCache team

ISGC 2016, Taiwan

Fermilab

NDGF
Nordic DataGrid Facility

INDIGO - DataCloud

DESY

HELMHOLTZ
ASSOCIATION

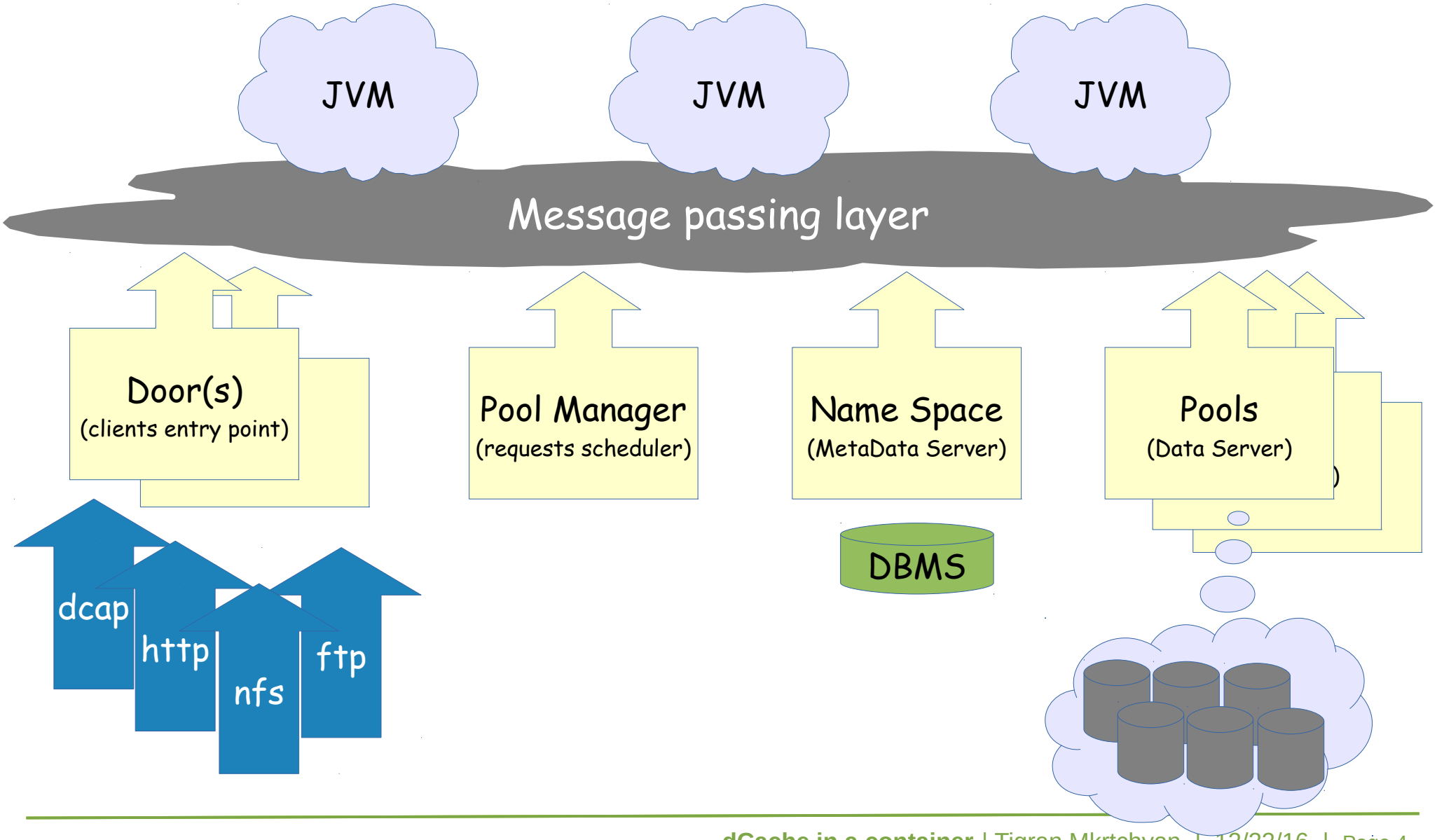
Motivation

- In production we need to:
  - run multiple version of dCache on the same host.
  - update some components on the same host.
- In development:
  - run multiple versions at the same time
  - test on multiple OSes
- Provide easy way for 'Get in touch'

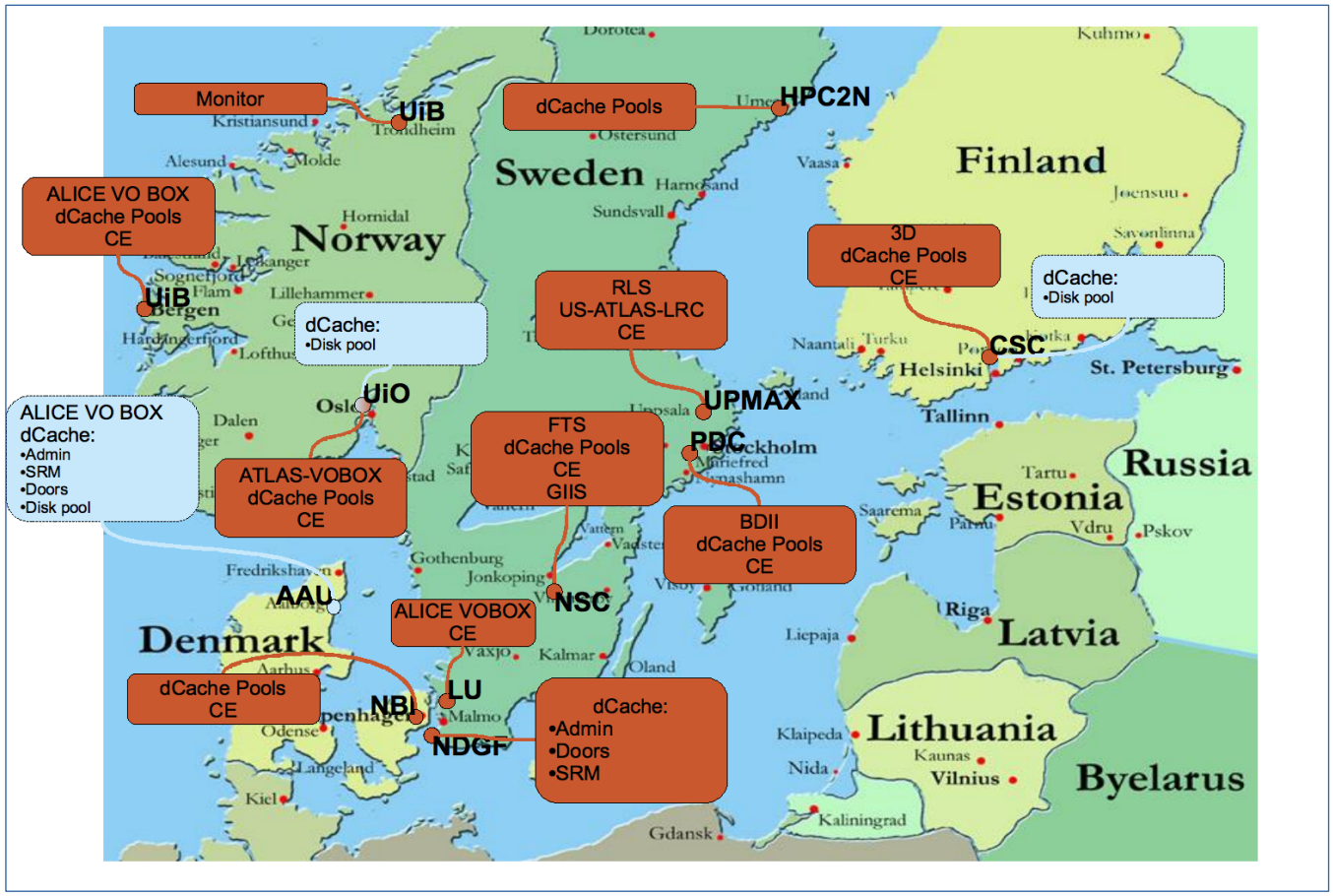
Usage around the World

- ~ 80 installations
- > 50% of WLCG storage
- biggest 22 PB
- Typical ~100x nodes
- Typical ~ 10^7 files

dCache on one slide

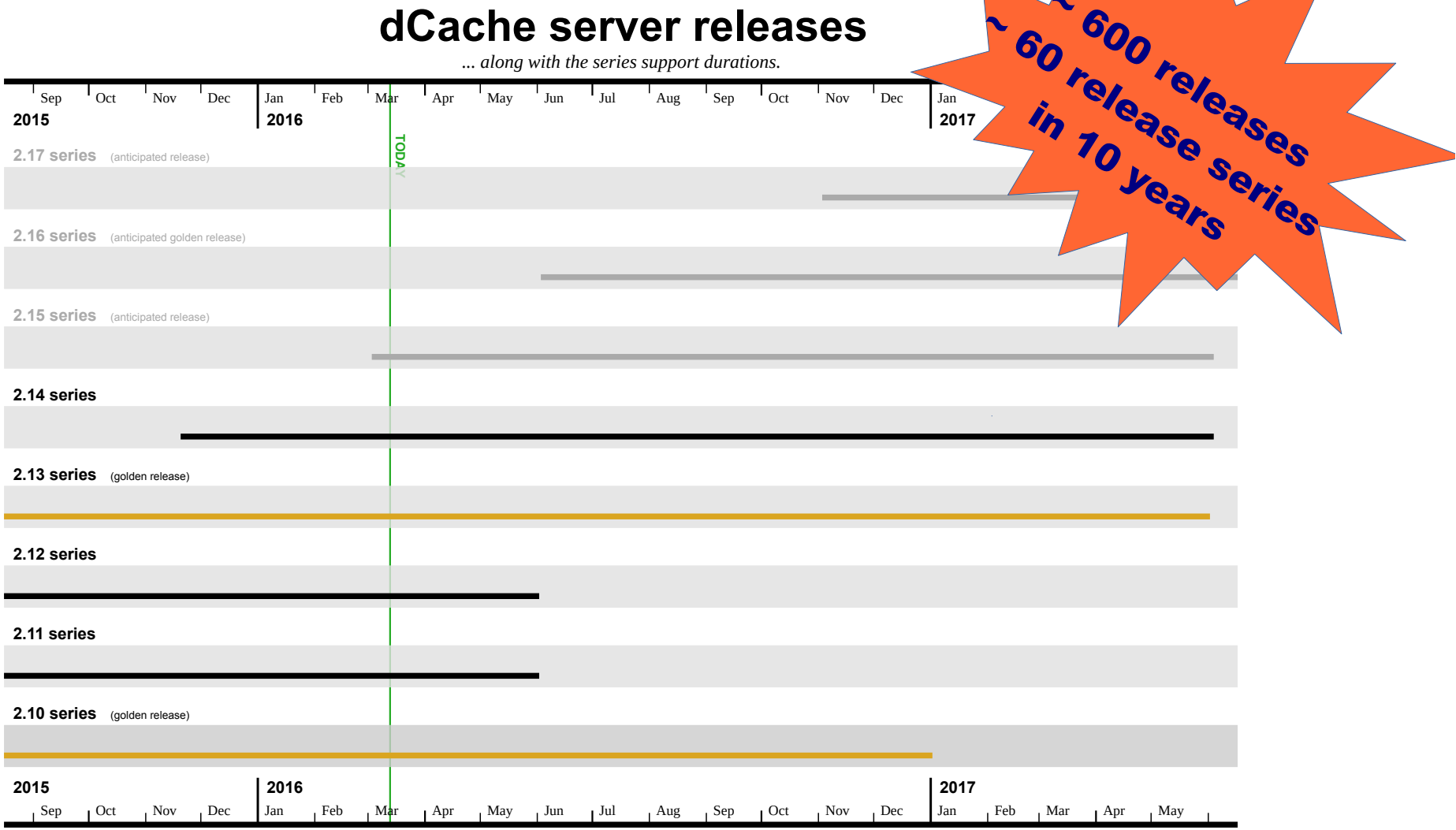


Distributed installation



- Single geographically spread instance.
- Synchronous updates hard to coordinate.
- Multiple major versions within single instance.
- More sites will follow this model in the future.

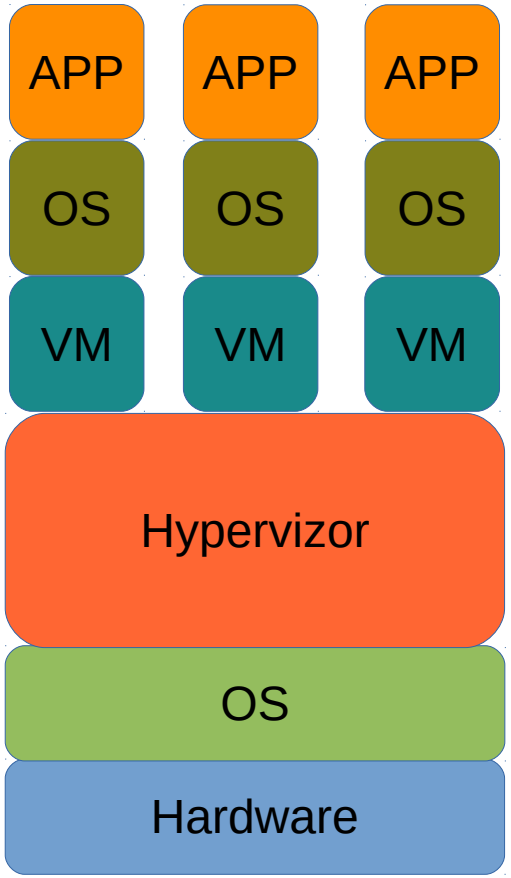
Supported versions and timeline

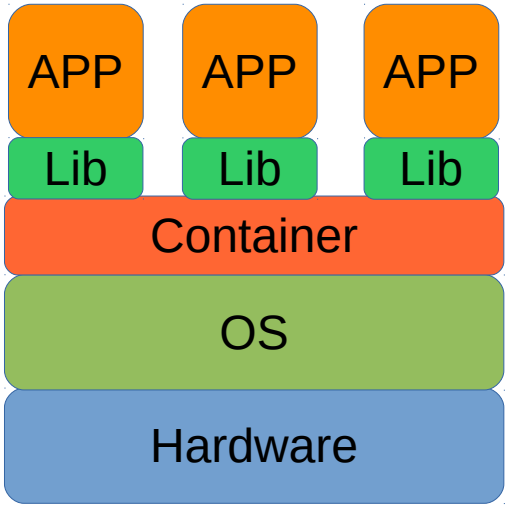


Containers (Operating-system-level virtualization)

- Isolate application to improve security
- Little to no overhead
- Limited to the same type of OS

Containers vs. VM





Containers

- Old idea
  - chroot, 1982
  - FreeBDS jails, 2000
  - Solaris Zones
- New trends
  - Easy to deploy
  - Easy to share
  - Use as a black-box

- A lightweight user tool to automate container management and deployment.
- Uses kernel provided cgroups and namespaces to isolate and limit resources.
- Automatically adopts iptables according network configuration.
- Creates read-only container images with read-write overlay filesystem on top, when running.
- With DockerHub provides a repository to store and share containers.

Dockerfile

- The make file for docker image.
- Describes how to build the image.
- Describes how to start the image.
- Defines which network ports must be exposed.
- Each step is saved as intermediate image for incremental builds.

Dockerfile, example

```
# Based on CentOS 7
FROM centos:7
MAINTAINER dCache "https://www.dcache.org"

# install required packages
RUN yum -y install java-1.8.0-openjdk-headless
RUN yum install -y https://www.dcache.org/downloads/dcache-2.14.13-1.noarch.rpm

# add external files into container at the build time
COPY dcache.conf /etc/dcache/dcache.conf
COPY run.sh /etc/dcache/run.sh

RUN chmod +x /etc/dcache/run.sh

# the data log files must survive container restarts
VOLUME /var/log/dcache

# expose TCP ports for network services
EXPOSE 22125 2049

# execute this when container starts
ENTRYPOINT ["/etc/dcache/run.sh"]
```

docker, command

- One stop shop.
- Build and manipulate images.
- Manages container life cycle: start, stop, ...
- Fetches and updates images in the repository.

docker, example

```
$ docker build -t local/dcache-upstream .
Step 1 : FROM centos:7
....
Step 10 : ENTRYPOINT /etc/dcache/run.sh
....
Successfully built dd2648bc7471
$  docker images
REPOSITORY                    TAG          ......        VIRTUAL SIZE
local/dcache-upstream   latest       ......             615.9 MB
docker.io/centos             7             ......             196.6 MB
$
```

Docker, volumes

- Persistent files/directories stored on host filesystem.
- Can be shared between containers.
- A specific file/directory can be injected into container.

docker run, almost real example

```
$ docker run -dt \
        -v /tmp/log:/var/log/dcache \
        -p 22125:22125  \
        local/dcache-upstream \
       dcap
```

Docker, network

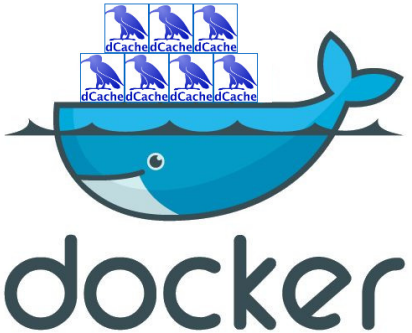
- Three default types
  - none – no external connectivity
  - host – expose host network to container
  - bridge – NAT like network, default
- Mapped Container Mode
  - share network stack between containers

Containerize dCache

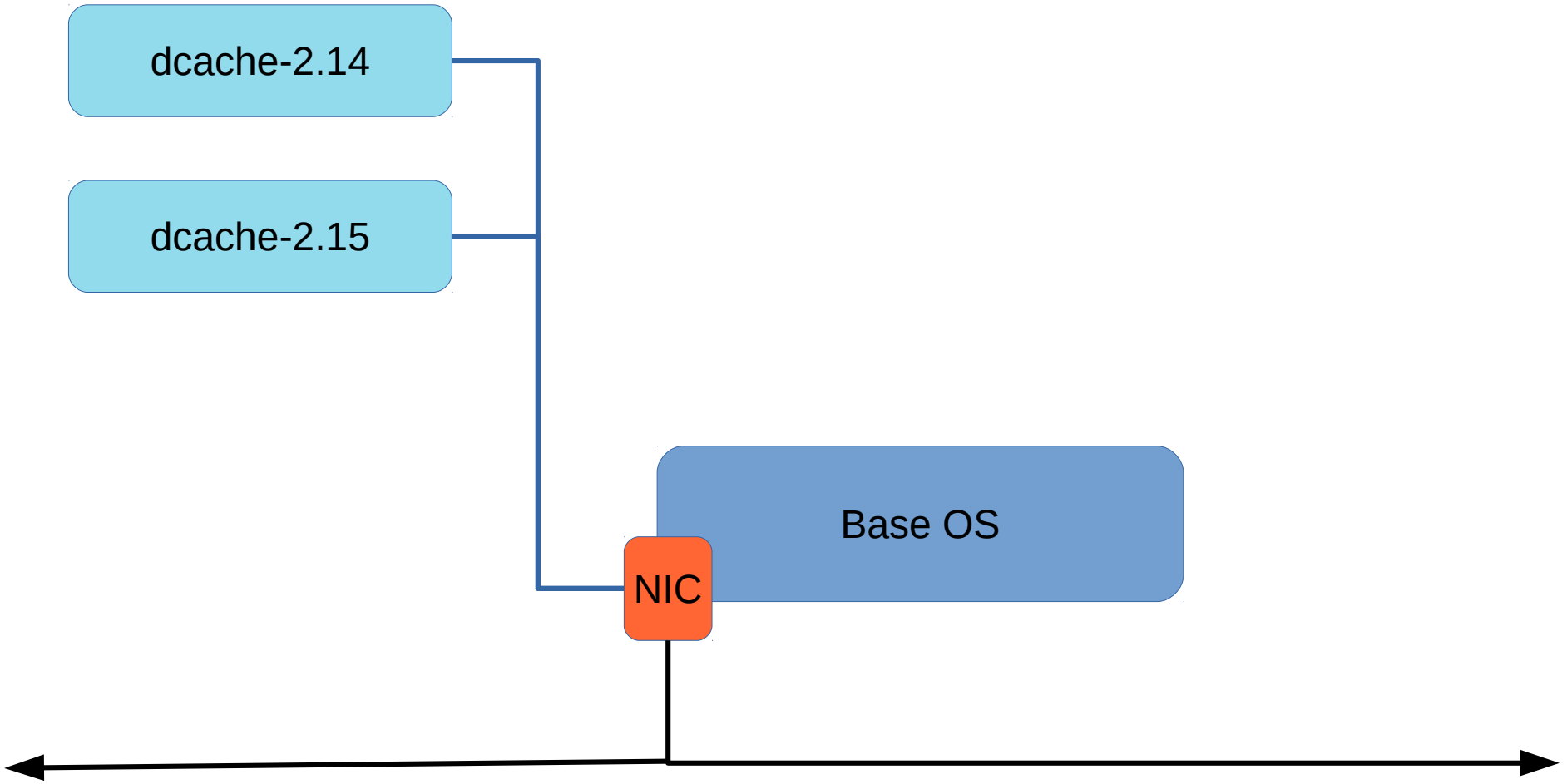
```
[poolA-${host.name}]

[poolA-${host.name}/pool]

pool.name=${host.name}-A

pool.path=/dcache/${pool.name}


[poolB-${host.name}]

[poolB-${host.name}/pool]

pool.name=${host.name}-B

pool.path=/dcache/${pool.name}
```

Containerize dCache

```
$ docker run ...  dcache-2.15 poolA

$ docker run ...  dcache-2.14 poolB

$ docker ps

CONTAINER ID        IMAGE                  ...

a1e456849852        local/dcache-2.15  ... af96afd07103
    local/dcache-2.14  ...

$
```



What just happened?



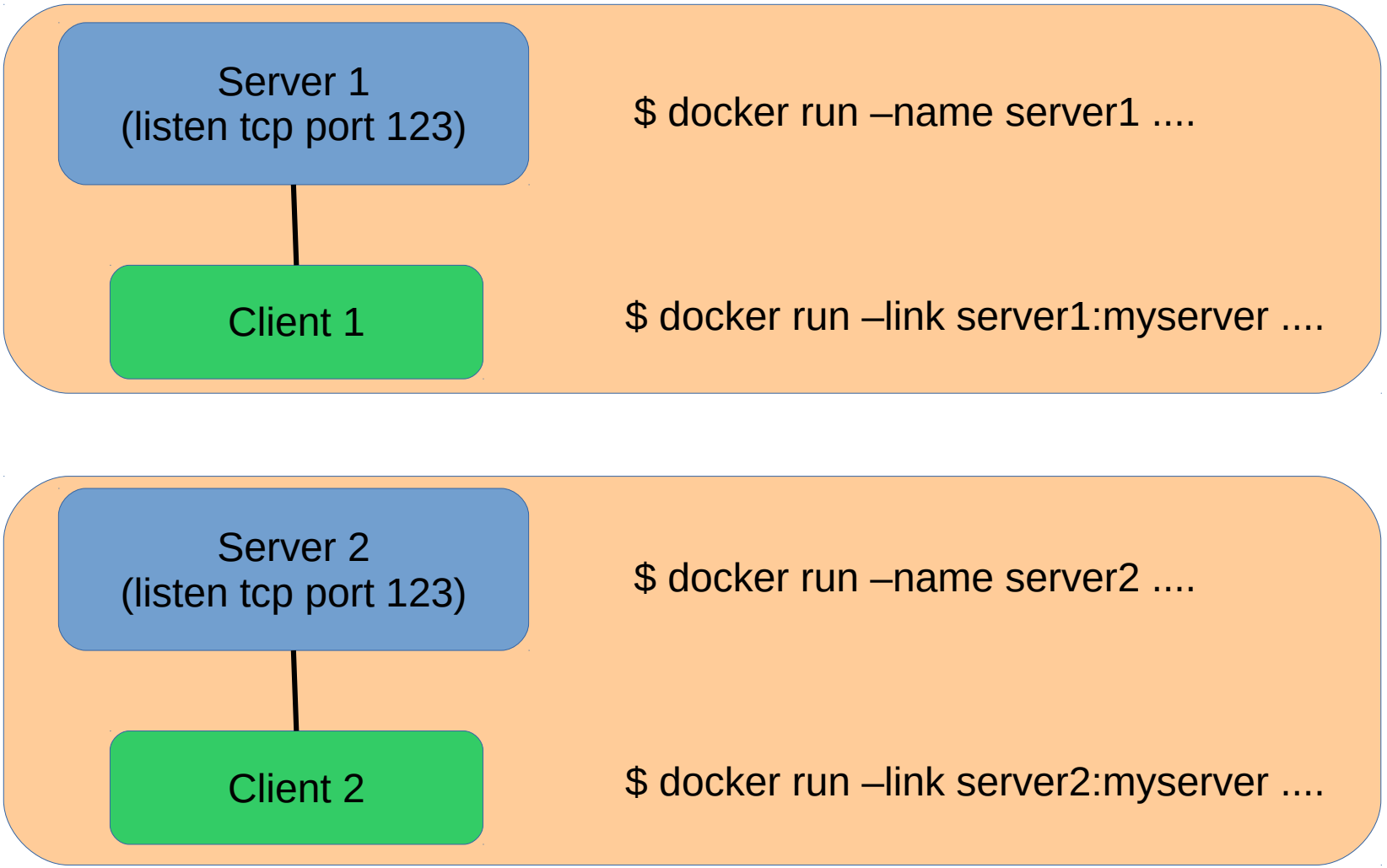
Containerize dCache (full command line)

```
$ docker run -dt --net=host \
  -v /tmp/pools:/dcache \
  -v /tmp/log:/var/log/dcache \
  -v `pwd`/docker-layout.conf:/etc/dcache/layouts/docker-layout.conf  \
  local/dcache-2.15 poolA
```

Linked instances (Testing scenario)

- Running multiple versions serviers in parallel
- Running multiple clients in parallel
- Each server exposed to it's client only
- Each client sees it server only

Linked instances (Testing scenario)



Under the hood

```
# cat /etc/hosts
172.17.0.9    3469cf96d4aa
127.0.0.1 localhost
::1localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
172.17.0.6    myserver d6532c8278a1 server1
```

Under the hood

```
# ping myserver -c 3
PING myserver (172.17.0.6): 56 data bytes
64 bytes from 172.17.0.6: icmp_seq=0 ttl=64 time=0.123 ms
64 bytes from 172.17.0.6: icmp_seq=1 ttl=64 time=0.059 ms
64 bytes from 172.17.0.6: icmp_seq=2 ttl=64 time=0.059 ms
--- myserver ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.059/0.080/0.123/0.030 ms
#
```

Summary

- Containers provide light weight environment to run applications in production.
- Docker is a nice tool to create, run and share containers.
- Containers can cover many production use cases as well as test deployments