Scalable Cluster Management: LXC^3

Erich Focht
Andreas Jeutter
NEC Deutschland
April 2023

Motivation

The Problems

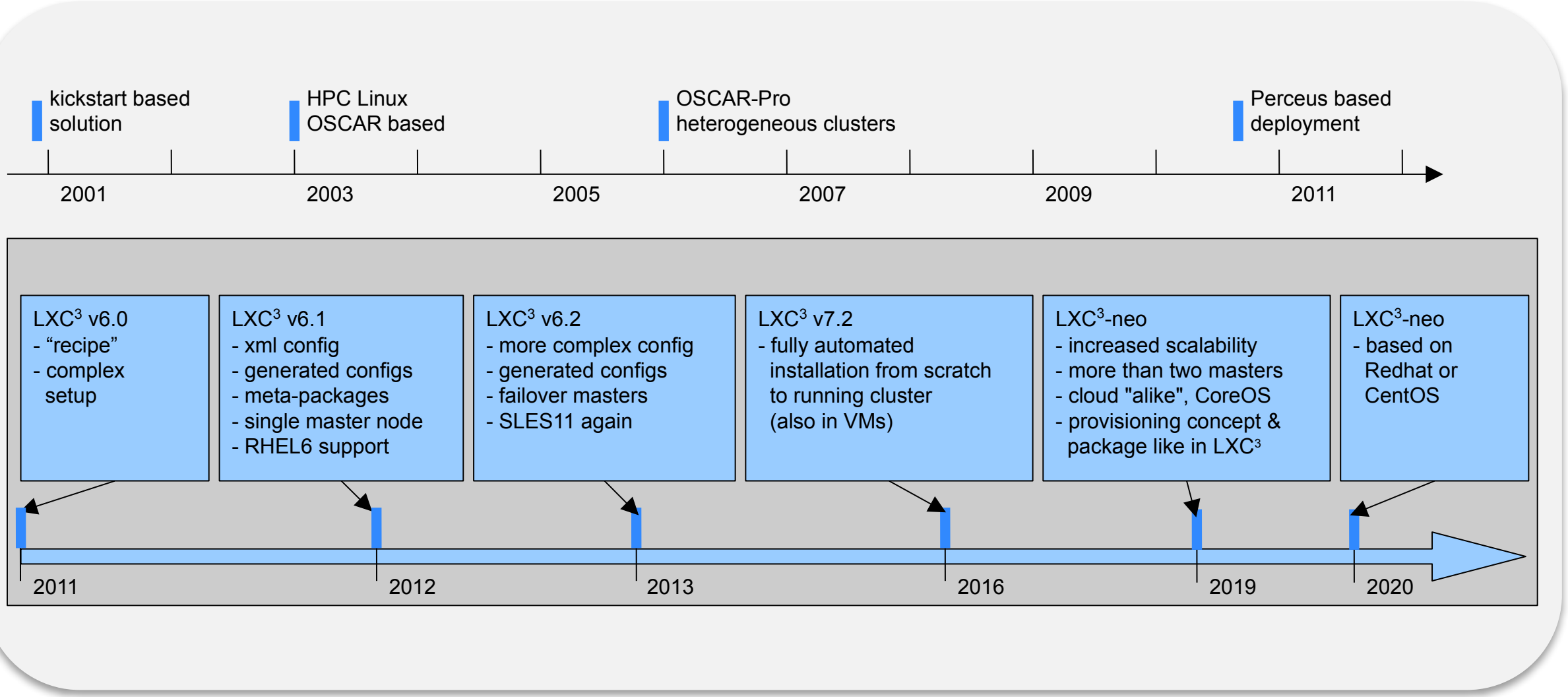
- Bare metal deployments → platform specifics
  - Stateful or stateless? Both!
- Node descriptions → Data model!
- What shall be deployed? → Node images
- Node images life cycle
- Nodes are not equal!
- Remote node power / platform control
- Management node(s) installation

- Many steps, error prone. “Recipes” won’t work.

More Comfort

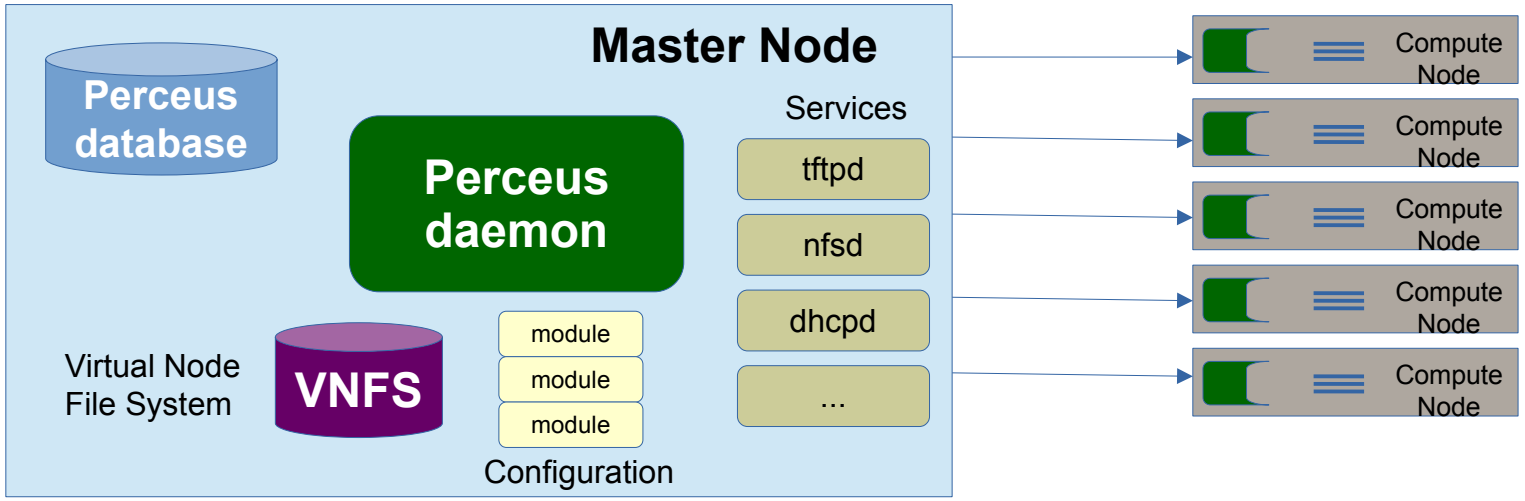
- High Availability!
- Batch system and resource scheduler
- User management
- Performance Monitoring
- Health monitoring and Alerting
- License server
- … Stuff
  - Git server, registry, repository, application software store, ...

History



Perceus

- Perceus: enterprise spin-off from Warewulf
- Stateless / diskless compute nodes



Perceus

- Perceus: enterprise spin-off from Warewulf
- Stateless / diskless compute nodes

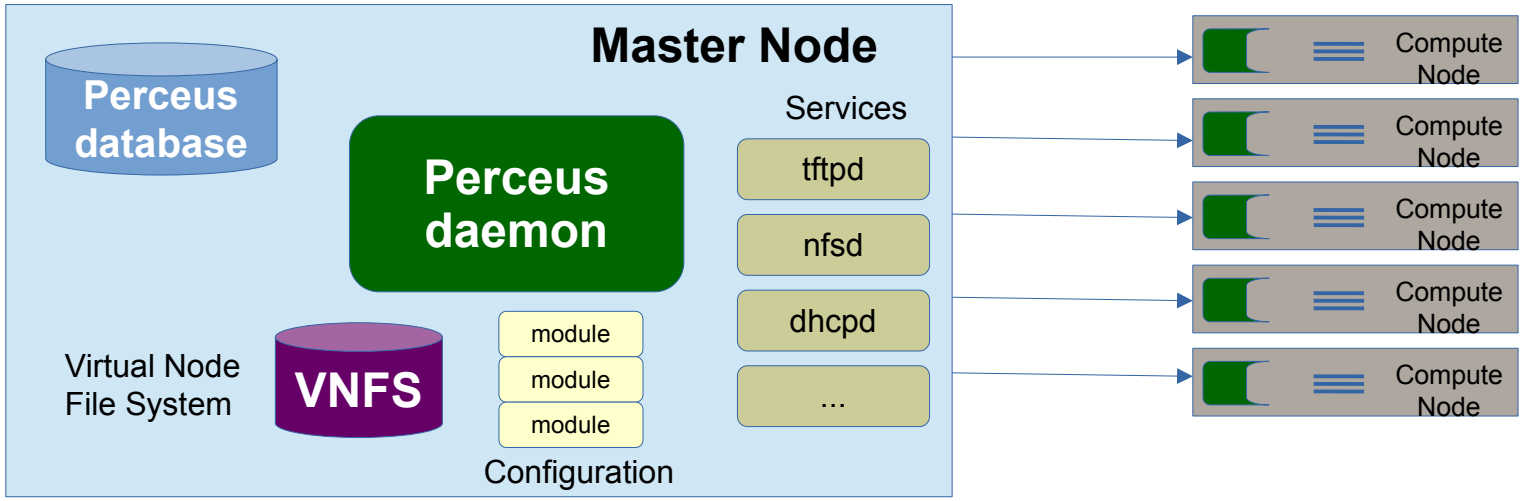
PERCEUS
Provision Enterprise Resources & Clusters Enabling Uniform Systems

warewulf
CLUSTERING SOLUTION

Deployment ONLY!

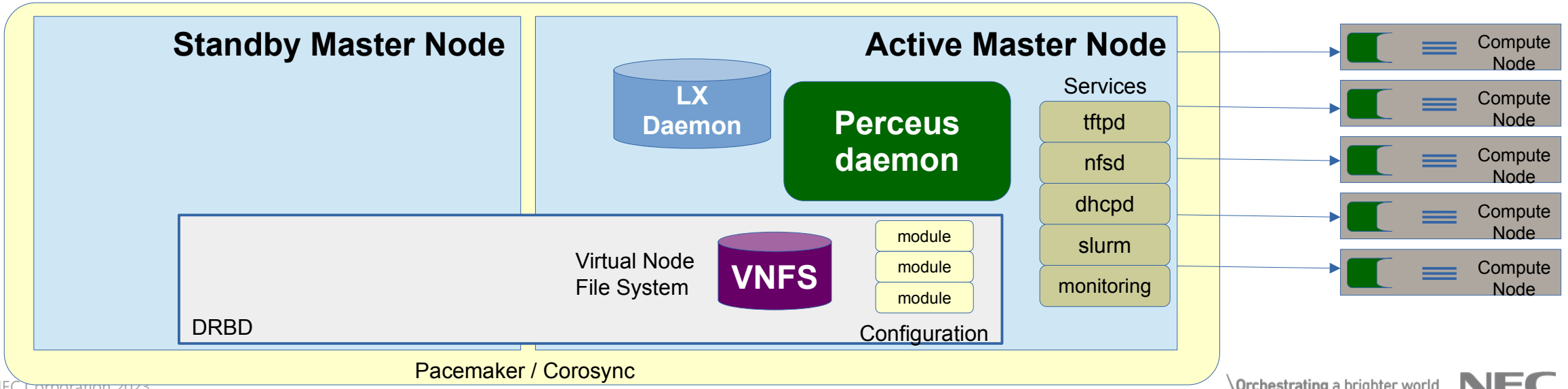
No high-availability

Diskless only!



LXC³ - Perceus

- Added high-availability: active/passive with pacemaker/corosync
- Storage synchronization with DRBD
- Added stateful deployments (with disks)
- Replaced Perceus database with LXDaemon (hierarchical database)
- Added cluster services, resource scheduler, monitoring, alerting, power control, ...



LXC³ - Experience

- Installed at many customers
- Fulfilled expectations for small / medium clusters: O(500) nodes
- Covering most needs
- VNFS concept good, but … versioning is expensive
- Automatization: helpful, but … dependencies between components
  - Updating requires a lot of knowledge and care
- Concept not directly scalable to larger systems

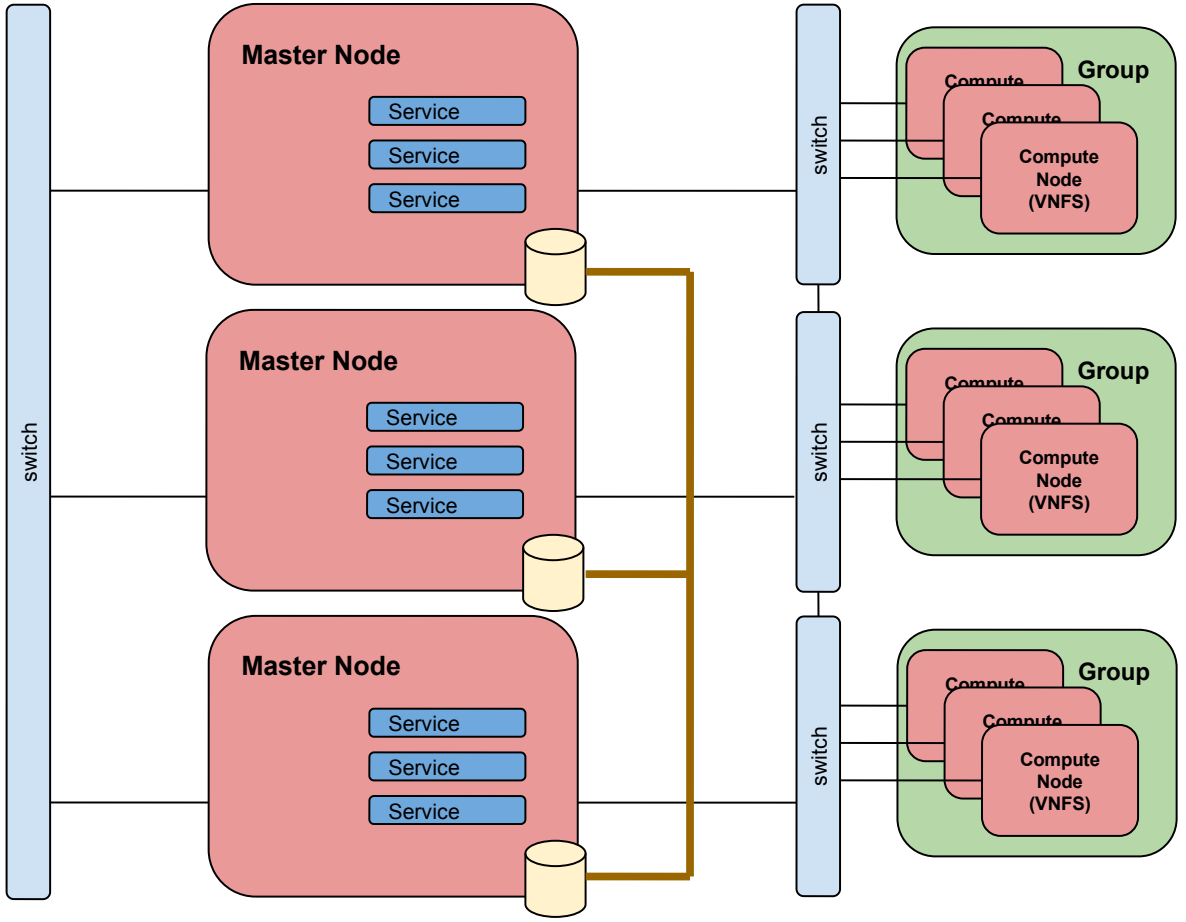
LXC³ neo

Core requirements:

- Scale to 2000 – 5000 nodes
- High-availability extremely important
- Support heterogeneous setups with many node types
- Updating must be easy
- Base installation must be easy

LXC³ neo Architecture

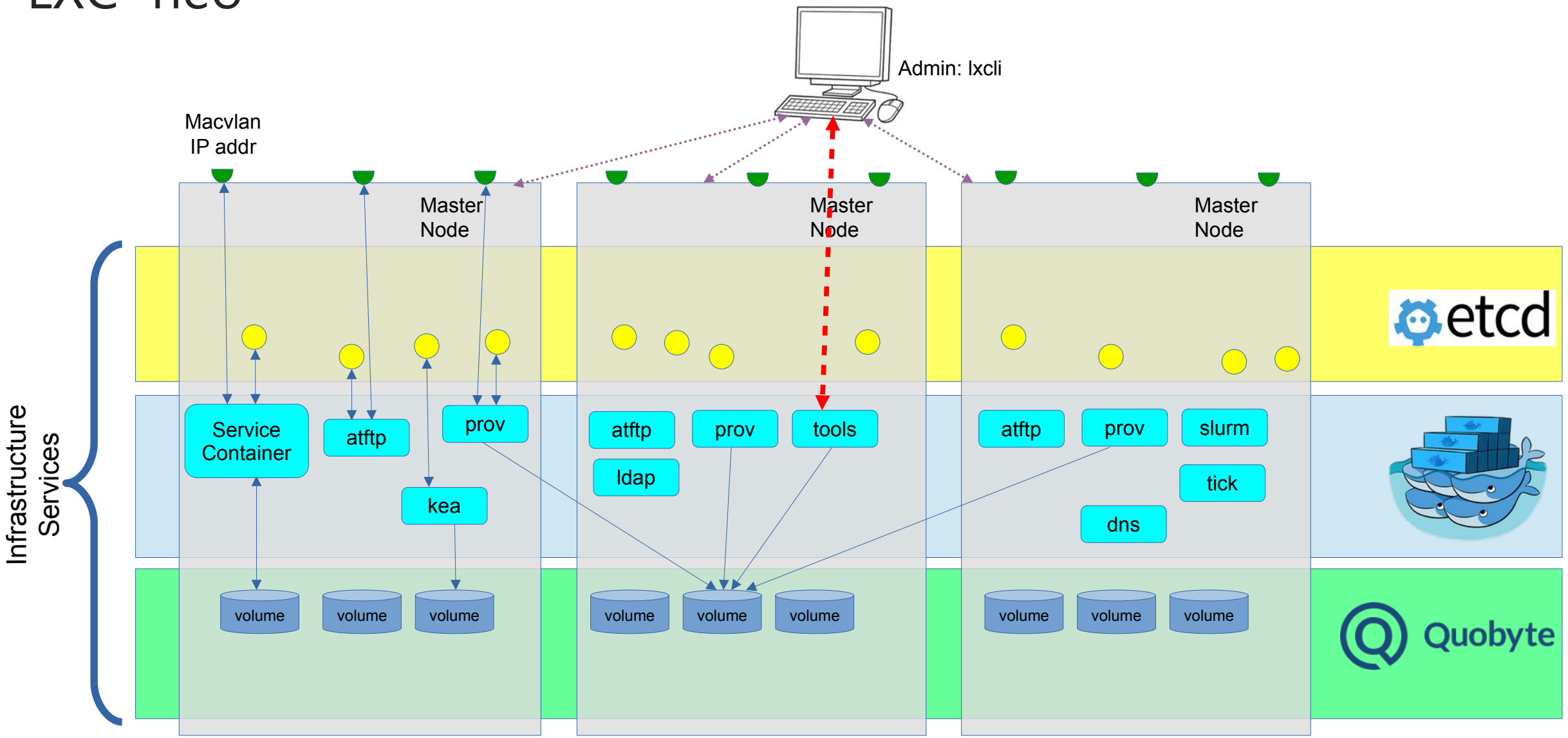
- Three or more master nodes
- Cluster organized in groups
- Services scaled → groups
- Storage: distributed/parallel
- Cluster database: distributed
- Single point of control



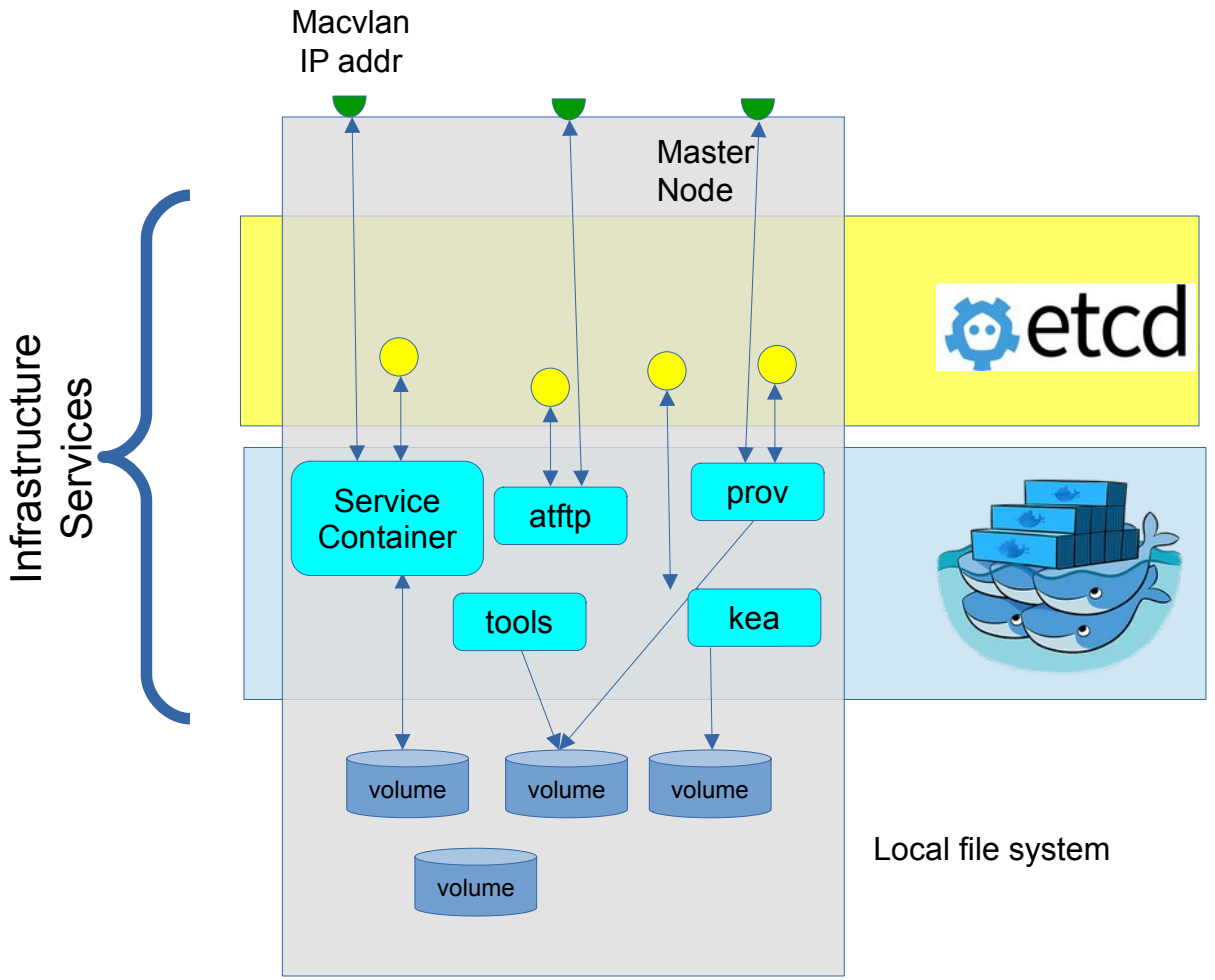
LXC³ neo Implementation Choices

- Services → Microservices in Docker containers
  - Select high performance alternatives
    - Eg. dnsmasq → atftp + kea, perl tcp/ip server → apache, kernel nfsd → ganesha user space nfsd
- High-availability → Container orchestration
  - Services move, but keep their IP addresses: MACVLAN
- Shared volumes with services data and state: parallel file system
- Database: distributed key-value store
- Keep: VNFS, Modules
  - but switch modules client/server from Perl to apache + curl

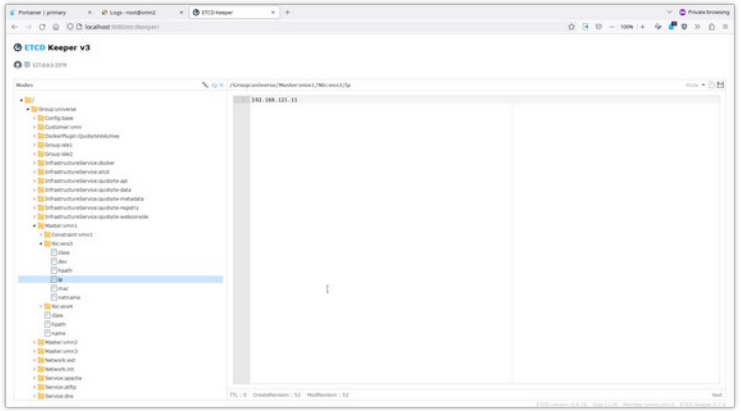
LXC³ neo

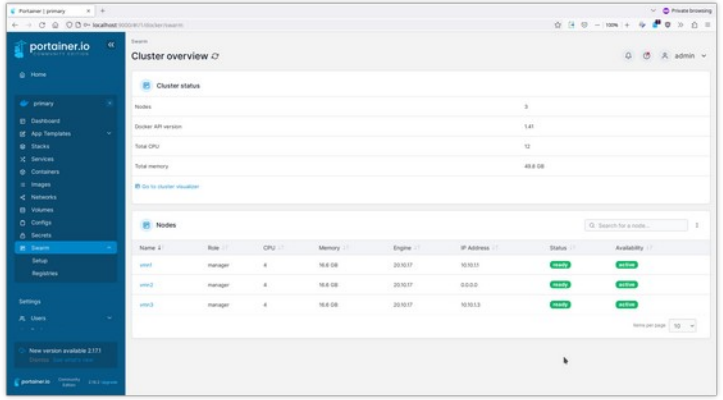


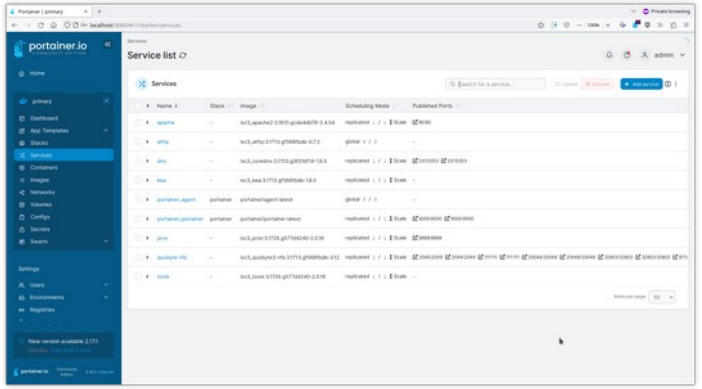
LXC³ neo Single Master Setup

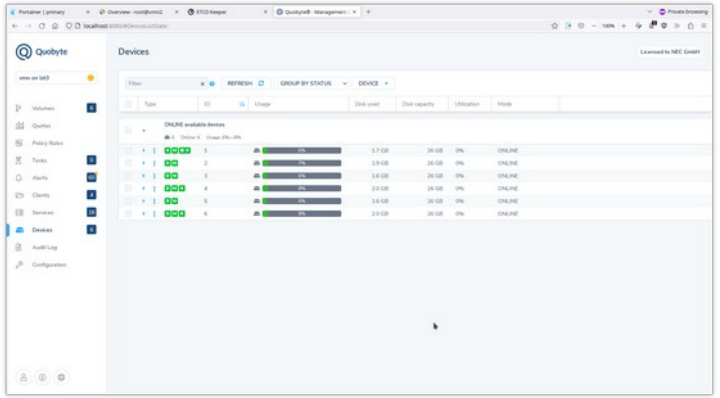


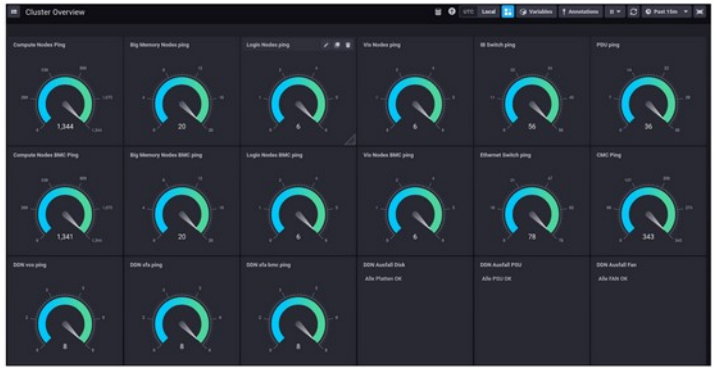
LXC³ neo : yes, there are GUIs

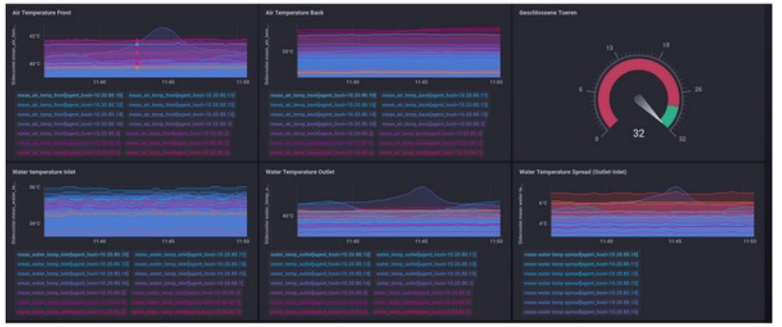












LXC^3 neo Evaluation & Feedback

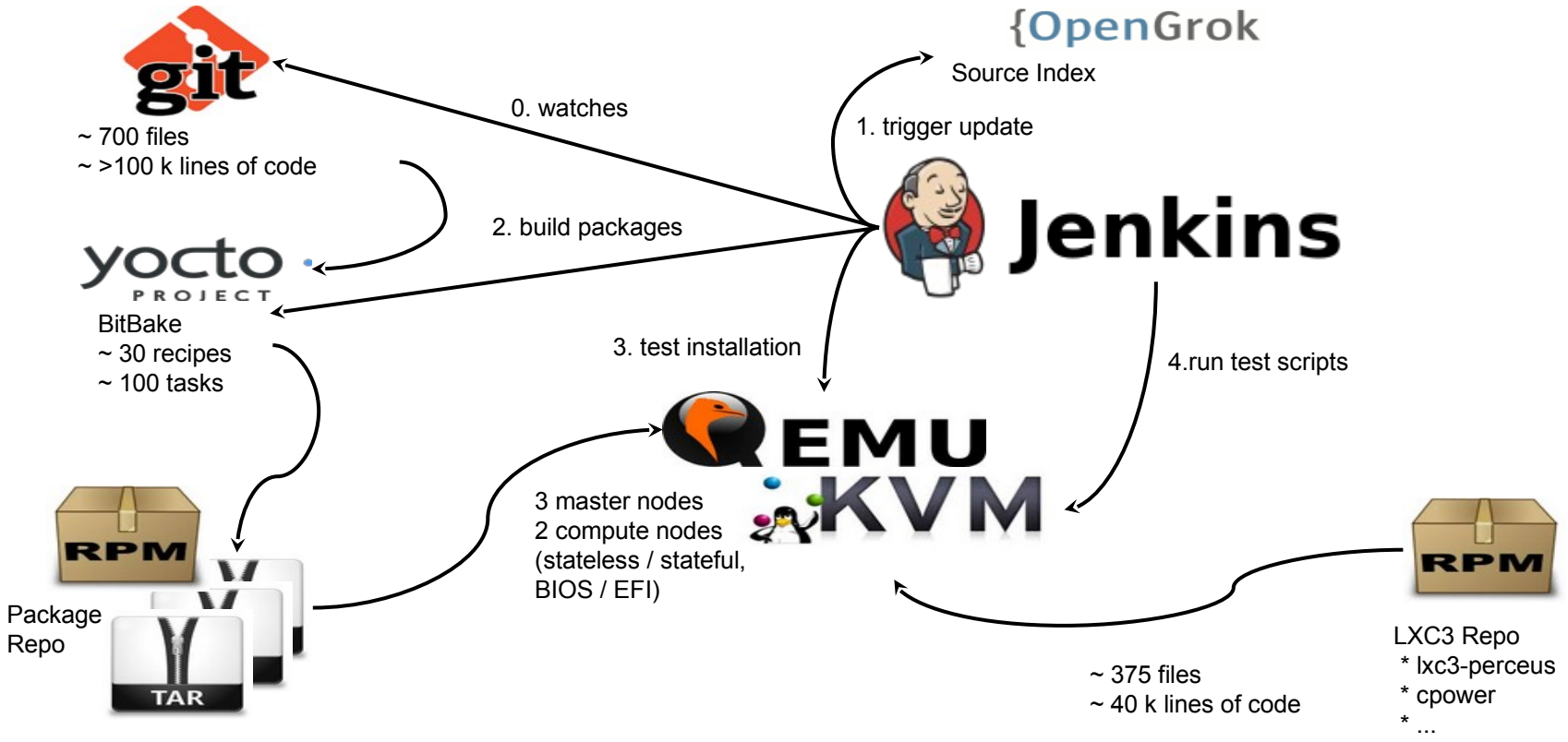
- Scalability: provisioning is very fast
  - Bring up 2000 diskless nodes in less than 100s
    - Most time spent in platform / BIOS
  - If power circuits permit: just turn on all nodes at once
    - Previously: iterative process, with delays
- Flexibility
  - Admins are able to add services (as containers) into the master swarm
  - Updates: easy for services / containers
  - Add modules with host specific customizations
    - eg. ansible module

LXC³ neo Evaluation & Feedback

- Complexity
  - Steep learning curve if no experience with containers
  - Network: macvlan fixed IP
    - Not supported by docker swarm
    - Requires workaround and additional (proxy-) containers
- VNFS image handling
  - Slow due to … parallel file system + many small files
  - VNFS “mount – modify – umount” administration hard to track
- Parallel file system
  - Overhead for parallel FS comes with a price…
  - Issues when clocks slightly out of sync

Development, CI

- Building: python packages, RPMs, container images
- Automated testing on VMs and bare metal



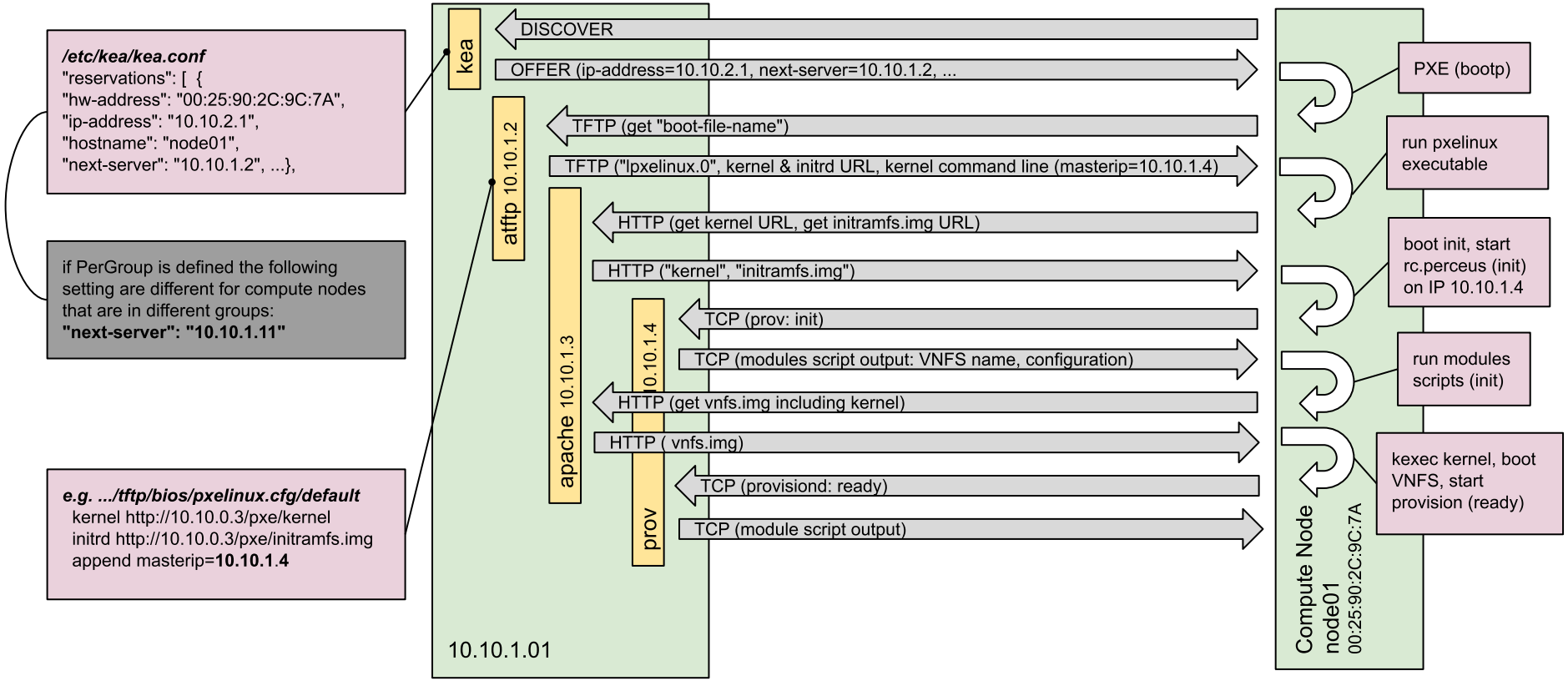
LXC³ next

- Networking without macvlan (ready)
  - Direct use of network interface not necessary: >25GbE interfaces are common
  - Swarm mesh network
  - CoreDNS aware of services location + load balancing
  - Eliminates need for proxy containers → less containers running
  - Less IP addresses needed (for each master node only, not for services)
  - → setup works on kubernetes, too (macvlans don’t)

LXC³ next

Networking without macvlan (ready)

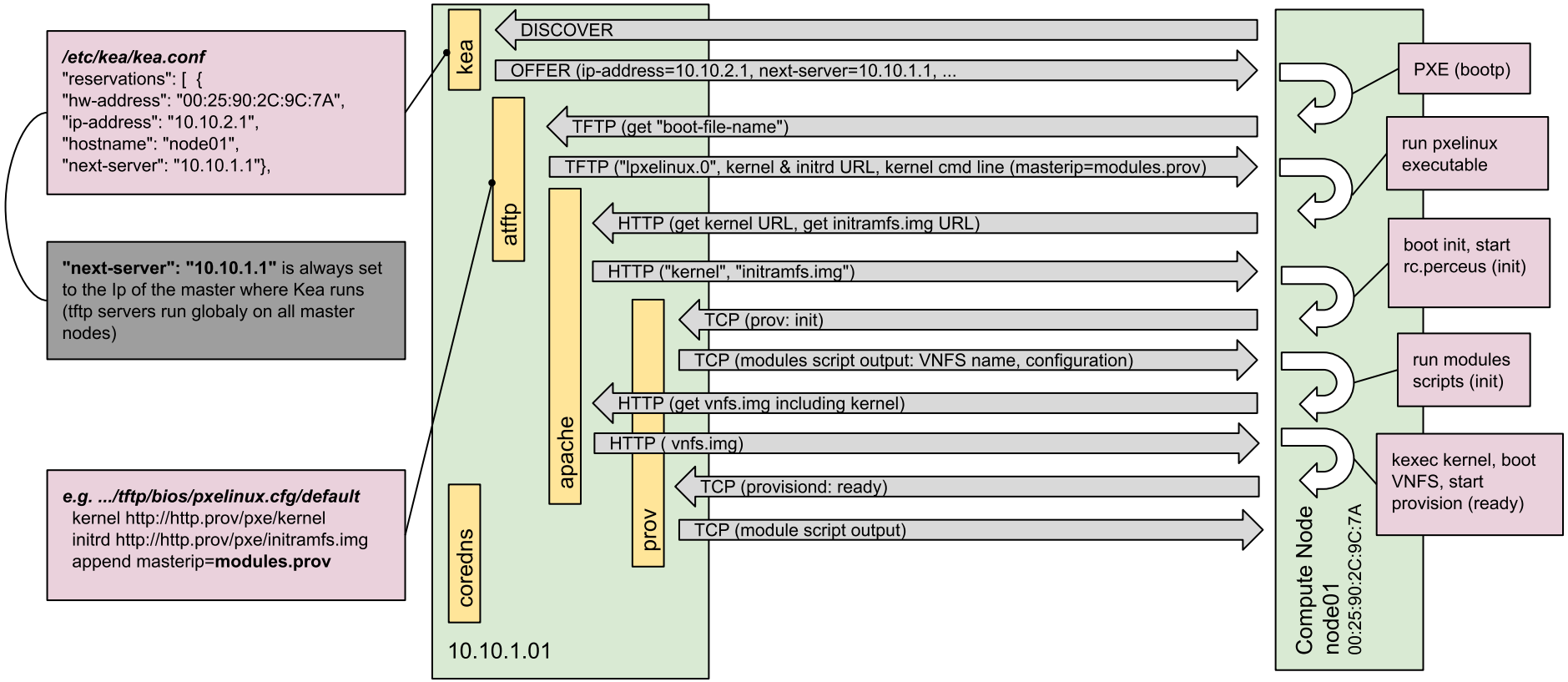
before



LXC³ next

- Networking without macvlan (ready)

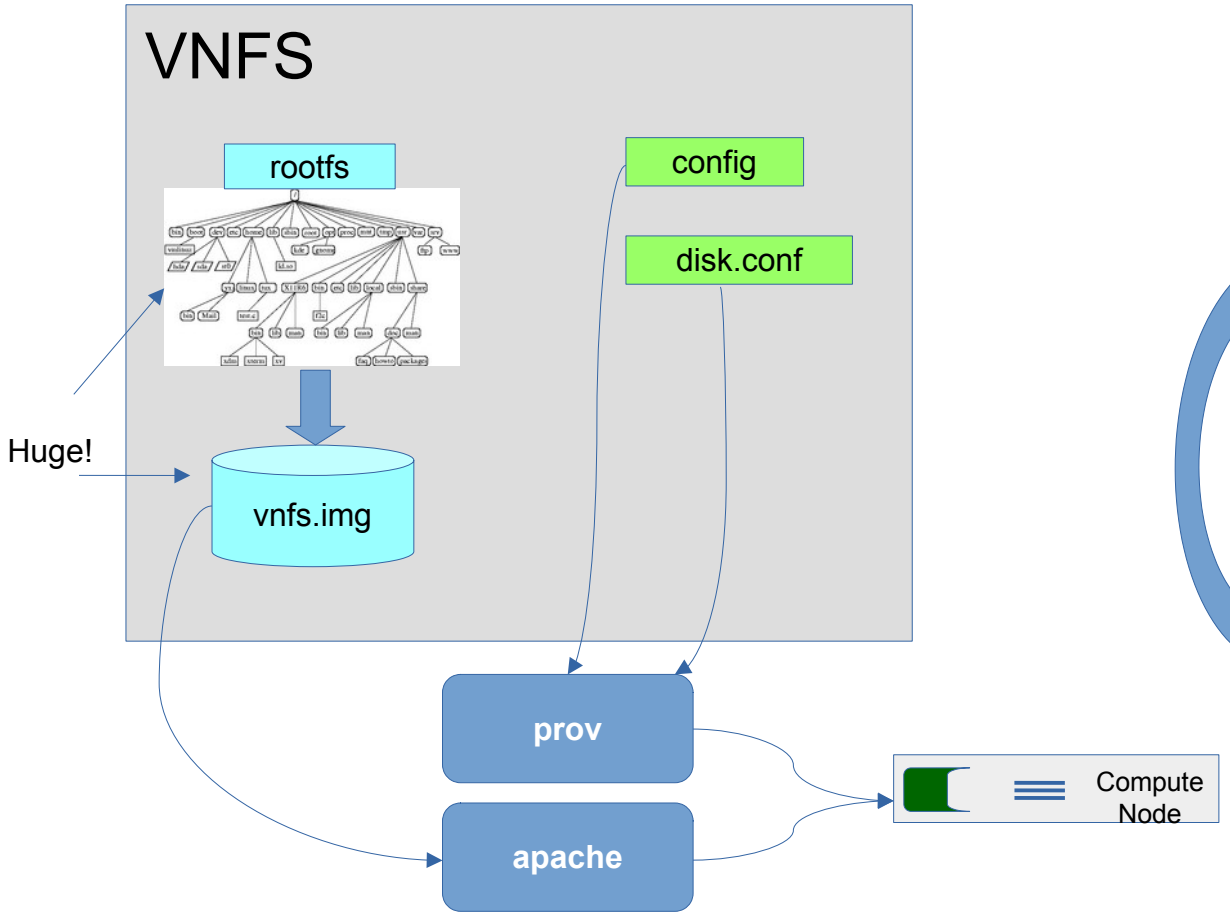
after



LXC³ next

- VNFS deploy from container image registry (ready)

before



Life cycle:

Create: script or lxcli vnfs clone

Mount: lxcli vnfs mount

Modify: edit or yum –installroot ...

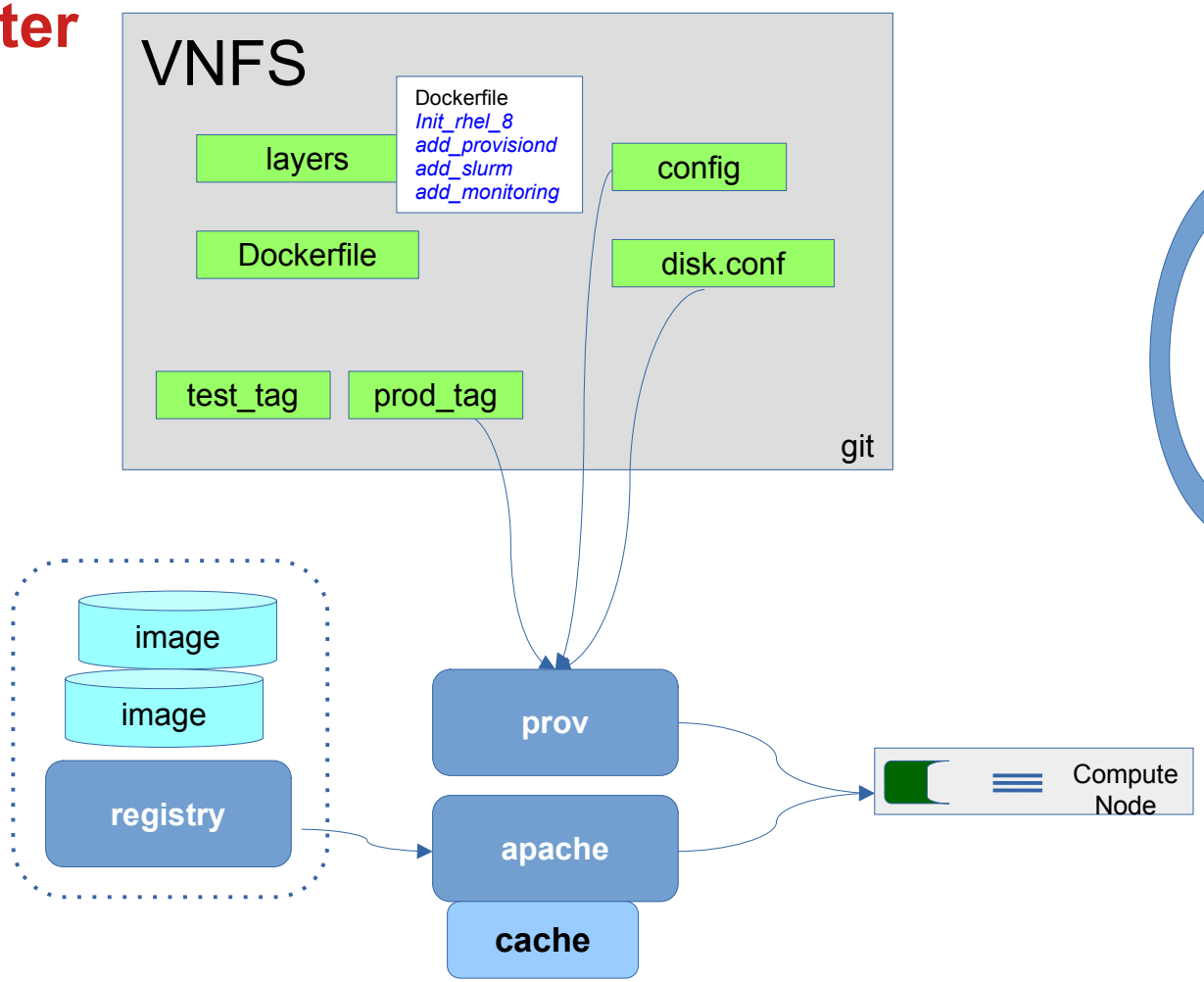
Umount: lxcli vnfs umount → vnfs.img

Deploy

LXC³ next

- VNFS deploy from container image registry (ready)

after



Life cycle:

Create: lxcli vnfs create or clone

Tweak Dockerfile(s)

Build: lxcli vnfs build from Dockerfiles in “layers”
- specify tag, eg. “v1.0”
→ container image stored in registry
→ tagged as “testing”

Deploy test node(s)

Activate for production:
lxcli vnfs activate –prod <vnfs>:<tag>

Deploy production nodes

LXC³ next

- Eliminate need for parallel filesystem
- VNFSes are container images
  - Changes trackable in git and reproducible
- Networking simplified
- Can run under kubernetes, too

Final Remarks

- Pure provisioning systems are “easy”
  - And come without high-availability, scaling, concept for additional admin services
  - In LXC^3-neo we cared about the missing bits
- Convergence of cluster + cloud management methods
  - Cloud: moving target, evolving fast
  - Admins meanwhile got used to containers
  - Node images are also container images
- Trend: mixing HPC + AI workloads
  - AI: more … interactive, running in containers
  - HPC: mostly bare metal, but … why not run each job in containers, too?


Orchestrating a brighter world
NEC