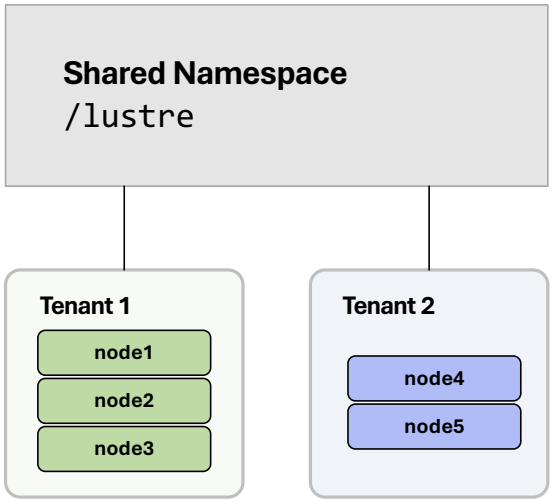

the lustre collective
Practical Multi-Tenancy for HPC/Cloud in 2026
The Lustre Collective
Expertise | Innovation | Partnership

Overview

Different models of “Multi-tenancy”

Model 01

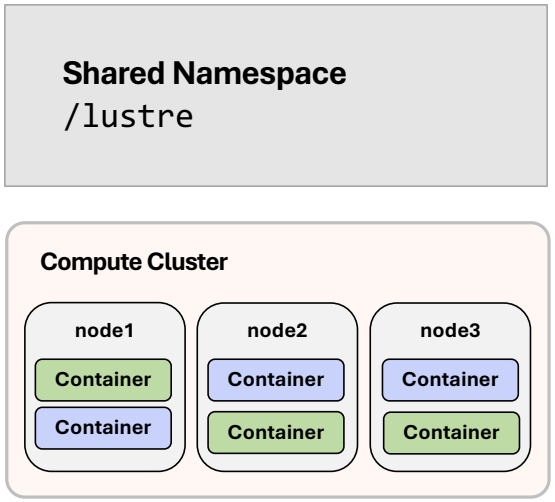
Node-based



- Single Namespace
- Partitioned Compute
- **Tenant as a Partition of Compute**

Model 02

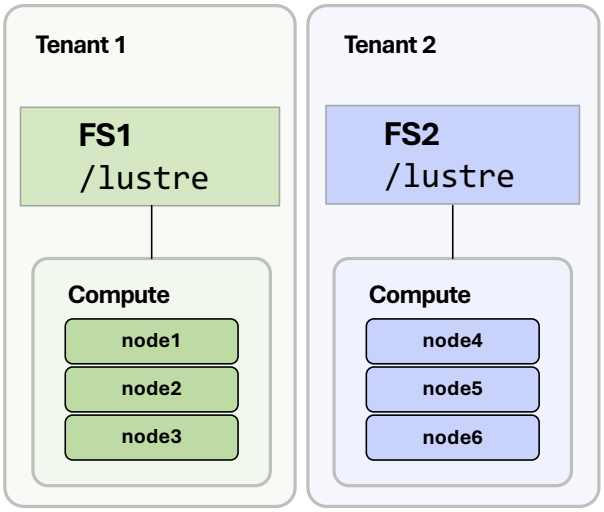
Intra-node / Container



- Single Namespace
- Single contiguous compute cluster
- **Tenant as Job / Container / Pod**

Model 03

Multi-single-tenant



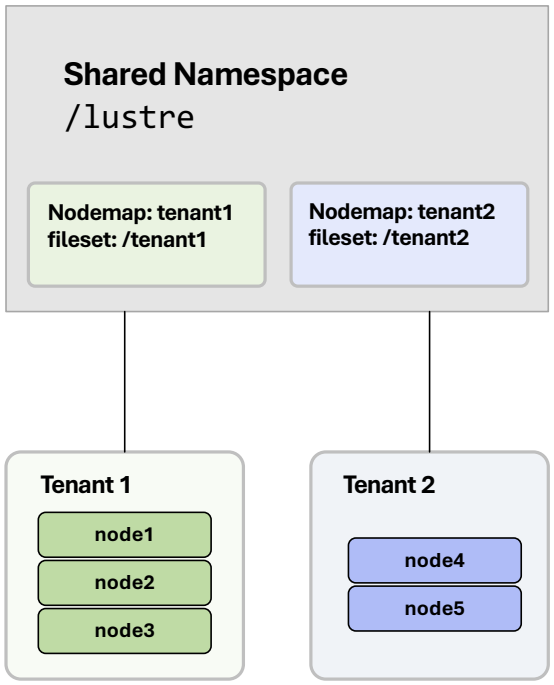
- Multiple Namespaces
- Partitioned Compute
- **Tenant as Namespace + Partition**

Overview

Node-based Multi-Tenancy

Model 01

Node-based



- Typical in Private Clouds, or shared HPC environments
- Single Lustre namespace, shared by all tenants
- **Network segregation** typically used between Tenant compute
- **Data Isolation** achieved through **Lustre nodemap + fileset**
- Quotas for Resource Isolation but IO performance is shared

Positives:

- Highly Flexible, add/remove tenants dynamically, change capacity
- With strong network segregation, secure data isolation

Challenges:

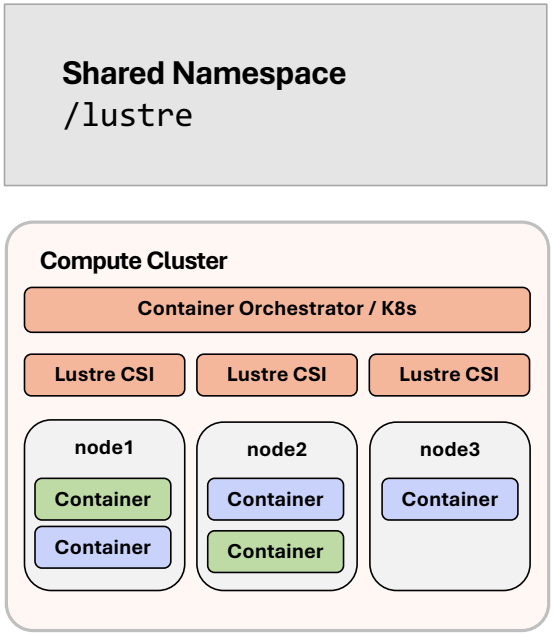
- Lacks strict performance isolation, noisy-neighbours
- Strong Authentication is optional
- Without encryption, lacks strong tenant-data privacy

Overview

Intra-node/Container Multi-Tenancy

Model 02

Intra-node / Container



- Common in recent shared AI/HPC environments
- Single Lustre namespace, shared by all compute nodes
- **Data Isolation** implemented through middleware (eg: CSI driver)
- Tenant data maps to directories, bind-mounted into containers
- Project Quota for Resource Isolation. Performance is shared

Positives:

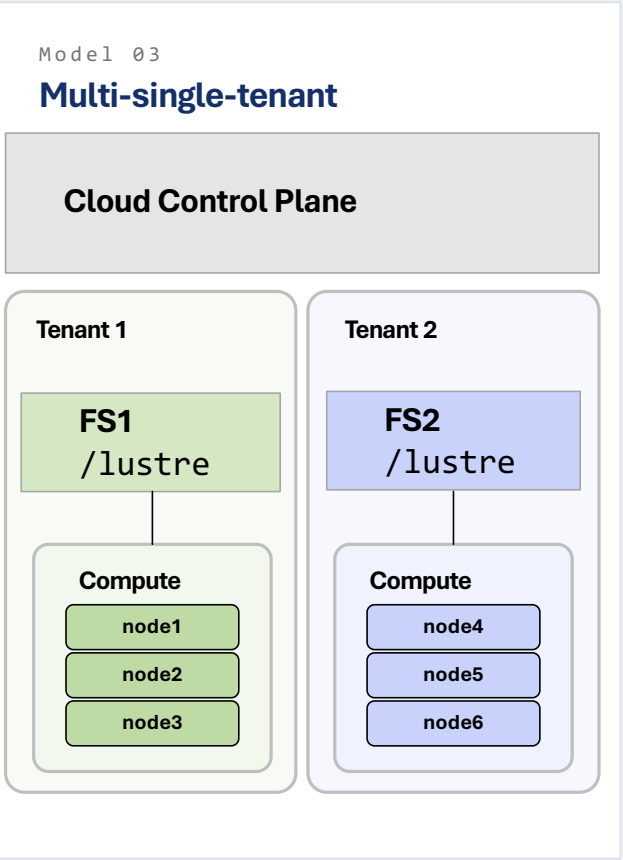
- Highly Flexible, add/remove tenants dynamically

Challenges:

- Lacks strict performance isolation, noisy-neighbours
- Isolation is fully dependent on K8s/Container security
- Without encryption, lacks strong tenant-data privacy

Overview

Multi-Single-Tenant



- Typical model of Large/Public Cloud environments
- Tenant compute + storage fully isolated
- Private Filesystem-as-a-service

Positives:

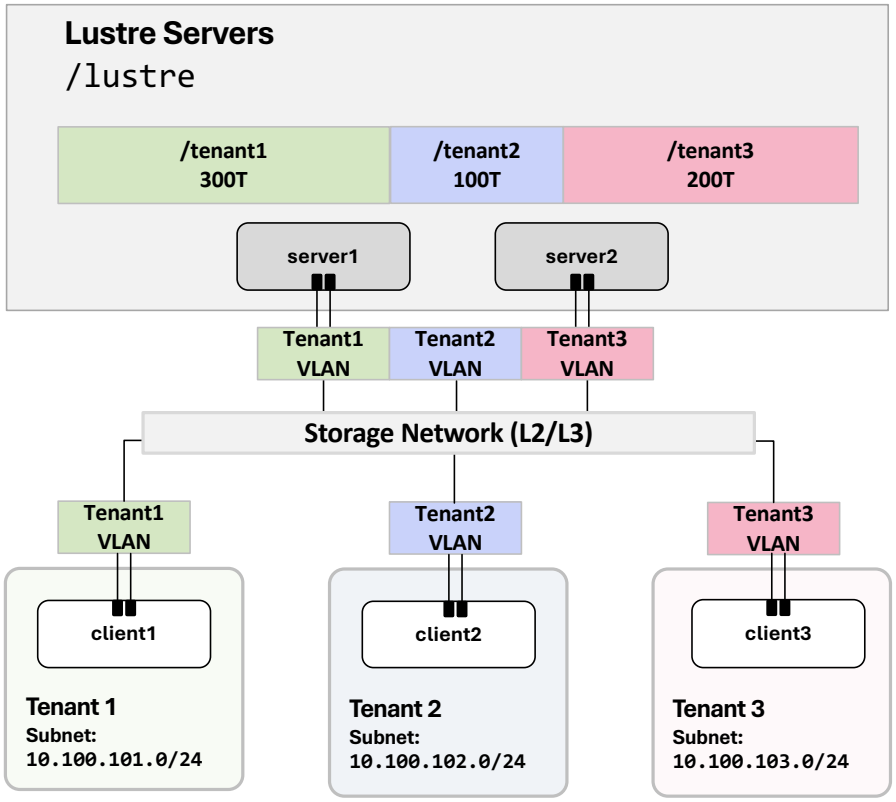
- Strong Data isolation
- Strong Performance isolation
- Potential for strong data privacy

Challenges:

- Requires significant infrastructure and orchestration
- Not as flexible, granular. Harder to shrink/change tenant capacity
- Tenant performance tied to capacity

Node-based Multi-tenancy in 2.17

Environment Overview



Customer Requirements

- Private cloud environment to multiple customers
- Each tenant has isolated compute environment
- Per-tenant VLANs provide tenant network isolation

Storage Requirements

- Each tenant has fully private storage namespace
- Access to tenant storage gated by tenant VLAN/subnet
- Tenants can be **added/removed** without disruption
- Tenant capacity allocation can be changed on the fly
- Tenant self-manages local users, groups and quotas

Multi-tenancy Setup Overview

Initial Multi-tenancy Setup

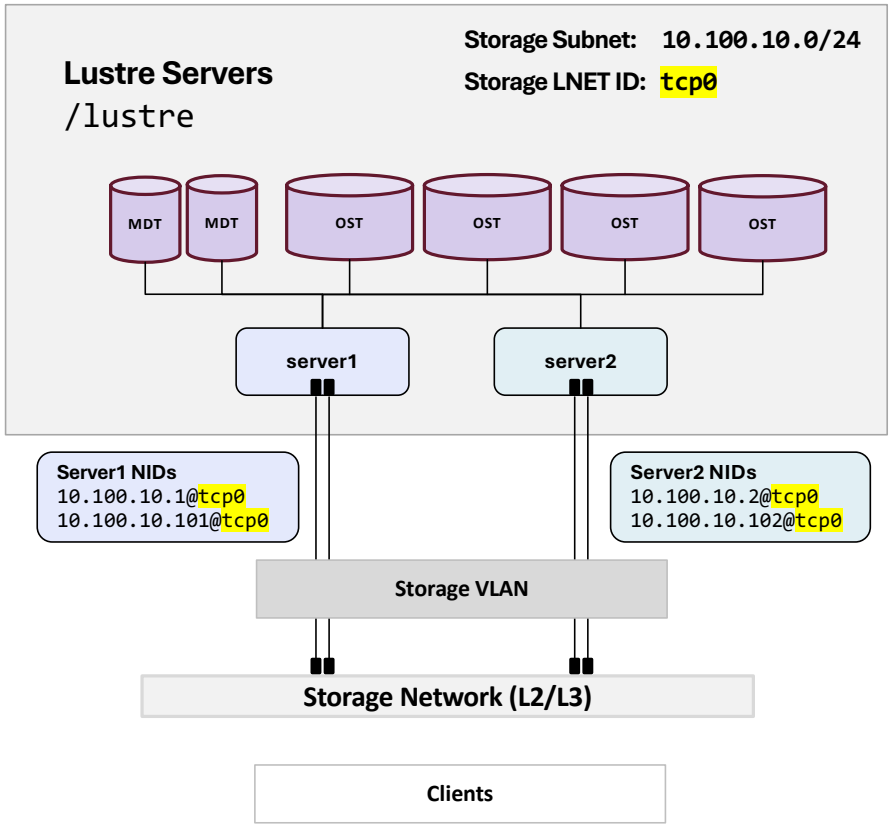
1. Server Initial Network configuration
2. Server Default Nodemap configuration

Onboarding a Tenant

1. Create Tenant Nodemap configuration
2. Create Tenant directory + quota
3. Create Tenant VLAN/Network on servers

Initial Multi-tenancy Setup

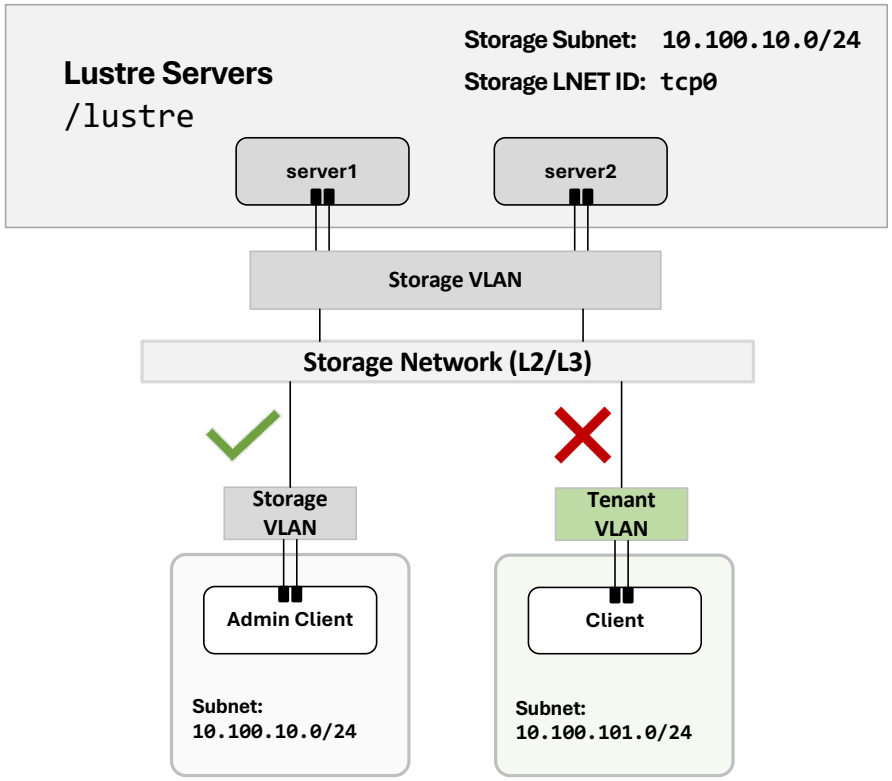
Initial Storage Setup



- Storage servers provisioned **with a storage-only VLAN**
- No client networks added initially
- Targets formatted with NIDs from storage VLAN

```
mkfs.lustre \
  --mdt \
  --fsname=lustre \
  --index=0 \
  --mgsnode='10.100.10.1@tcp0,10.100.10.101@tcp0' \
  --mgsnode='10.100.10.2@tcp0,10.100.10.102@tcp0' \
  --servicenode='10.100.10.1@tcp0,10.100.10.101@tcp0' \
  --servicenode='10.100.10.2@tcp0,10.100.10.102@tcp0' \
  /dev/disk/by-id/virtio-MDT0000
```

Storage Default Nodemap Configuration



Storage Nodemap

- Create a privileged ‘storage’ nodemap
- **Only** nodemap able to access the full namespace

```
lctl nodemap_add lustre_servers

lctl nodemap_modify --name lustre_servers --property admin --value 1
lctl nodemap_modify --name lustre_servers --property trusted --value 1

# Provide access only to storage VLAN subnet
lctl nodemap_add_range --name lustre_servers –range 10.100.10.[0-255]@tcp0
```

Default Nodemap

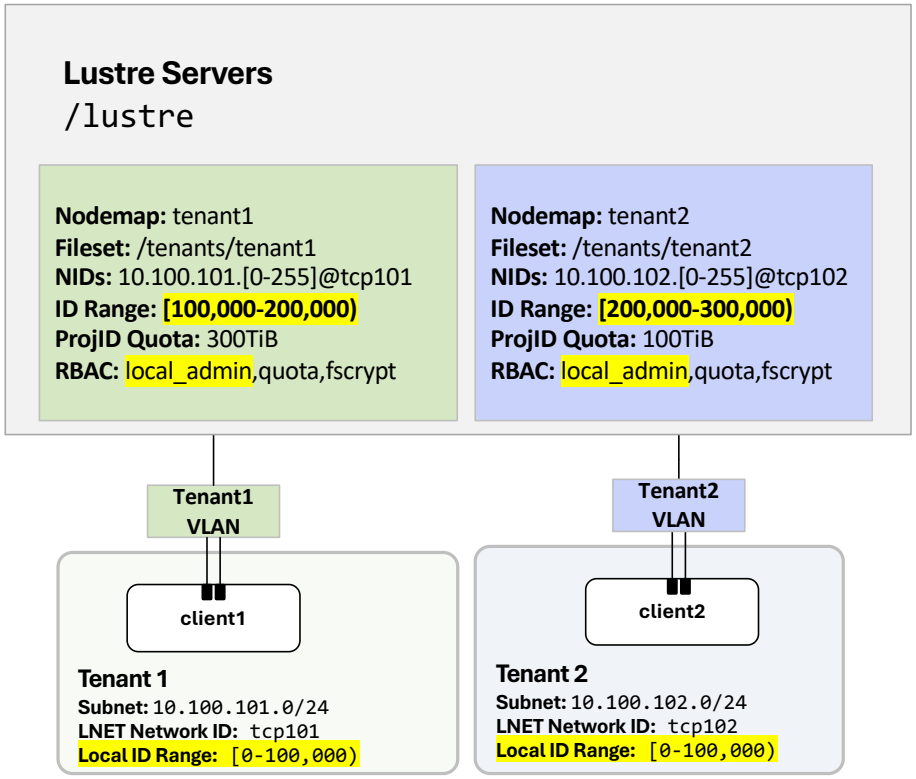
- Deny all other clients access to the filesystem
- Clients must be authorized to have storage access

```
lctl nodemap_modify --name default --property admin --value 0
lctl nodemap_modify --name default --property trusted --value 0
lctl nodemap_modify --name default --property deny_mount --value 1

# Activate the configuration
lctl nodemap_activate 1
```

Onboarding a Tenant

(1) Adding a Tenant Nodemap



Leveraging New Capabilities

- Every tenant has unique ID space via Nodemap ID-map [LU-18109] Nodemap UID/GID/ProjID idmap offsetting
- In this example, give each Tenant 100,000 unique IDs
- Delegate permissions management to local root user [LU-16524] Limit capabilities of local admin

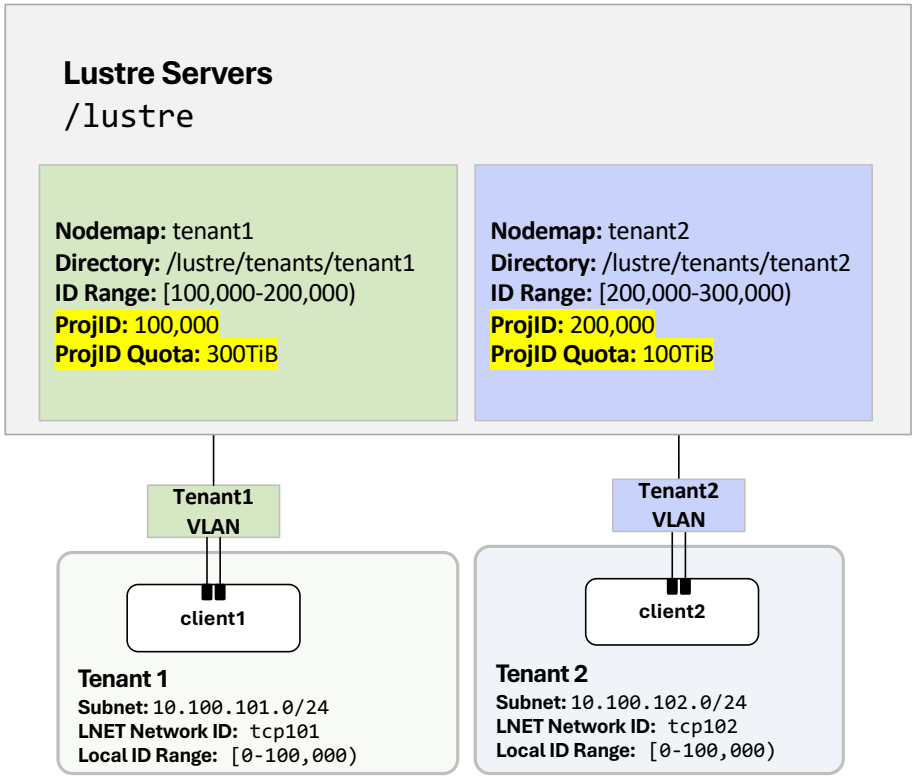
```
lctl nodemap_add tenant1
lctl nodemap_add_range --name tenant1 --range 10.100.101.[0-255]@tcp101
lctl nodemap_modify --name tenant1 --property admin --value 1
lctl nodemap_modify --name tenant1 --property trusted --value 0
lctl nodemap_modify --name tenant2 --property deny_unknown --value 1

# Set Tenant Quota on squashed ProjID for Tenant
lctl nodemap_modify --name tenant1 --property squash_projid --value 0
lctl nodemap_modify --name tenant1 --property map_mode --value projid

lctl nodemap_add_offset --name tenant1 --offset 100000 --limit 100000

lctl nodemap_modify --name tenant1 --property rbac --value
file_perms,local_admin,quota_ops,fscrypt_admin
```

(2) Adding a Tenant Directory

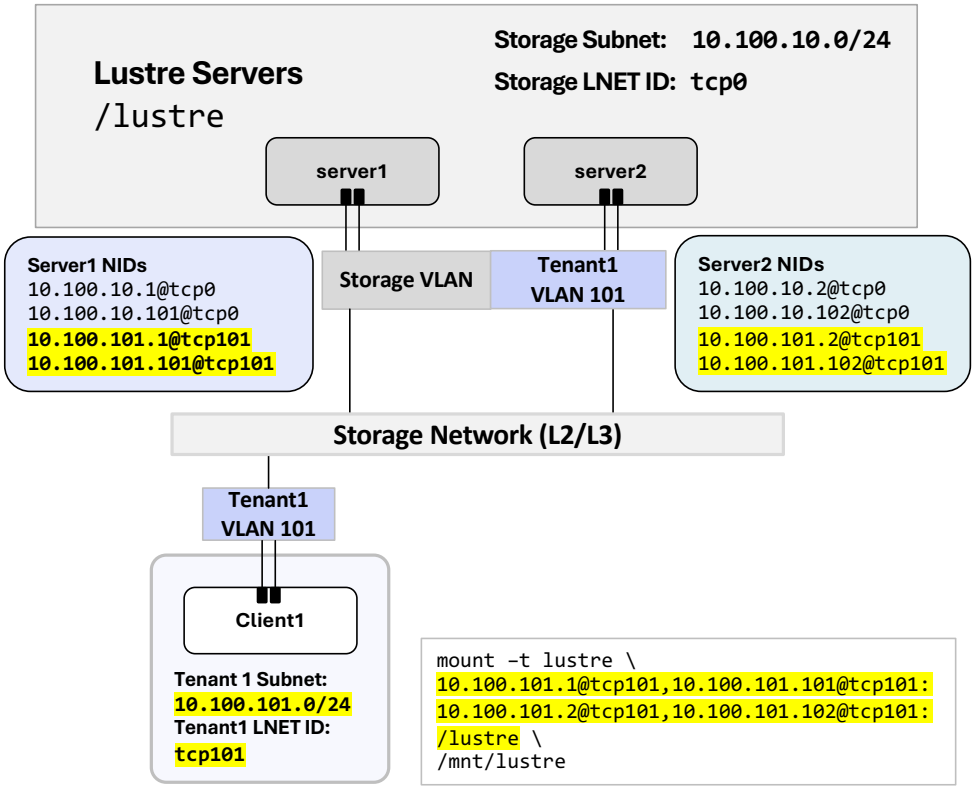


Create Tenant directory on Server/Admin Client

- Ownership of tenant directory mapped to Tenant local root
- Set Project ID to first ID of the Tenant ID range
- All tenant ProjIDs mapped to this value
- Configure ProjID Quota on Tenant directory

```
mkdir -p /lustre/tenants/tenant1
chown 100000:100000 /lustre/tenants/tenant1
lfs project -p 100000 -s -r /lustre/tenants/tenant1
lfs setquota -p 100000 -B 300G /lustre
```

(3) Adding a Tenant Network (Dynamically)



Dynamically Add New Tenant VLAN Connection

- Tenant compute and VLAN is configured on switches
- Lustre servers add new VLAN sub-interfaces & IPs
- Lustre servers add new LNET NIDs on tenant network
- **[LU-19095] Target notifies MGS about LNET changes**
- **[LU-10360] use Imperative Recovery logs for client->MDT/OST connections**

```
lctl set_param -P mgc.*.dynamic_nids=1
```

Additional patches currently on master

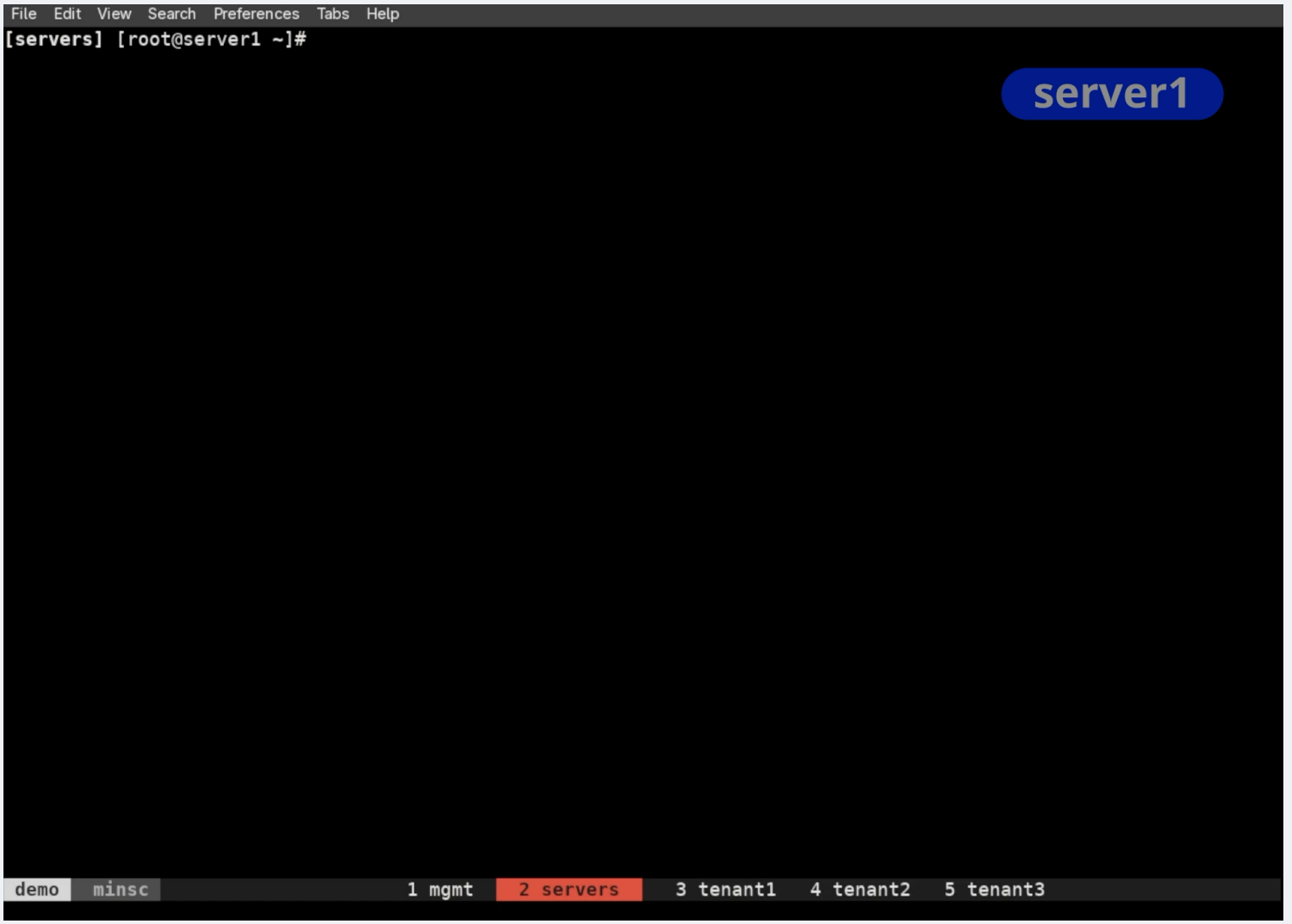
- **[LU-19692]**
- **[LU-19733]**

Dynamic NIDs

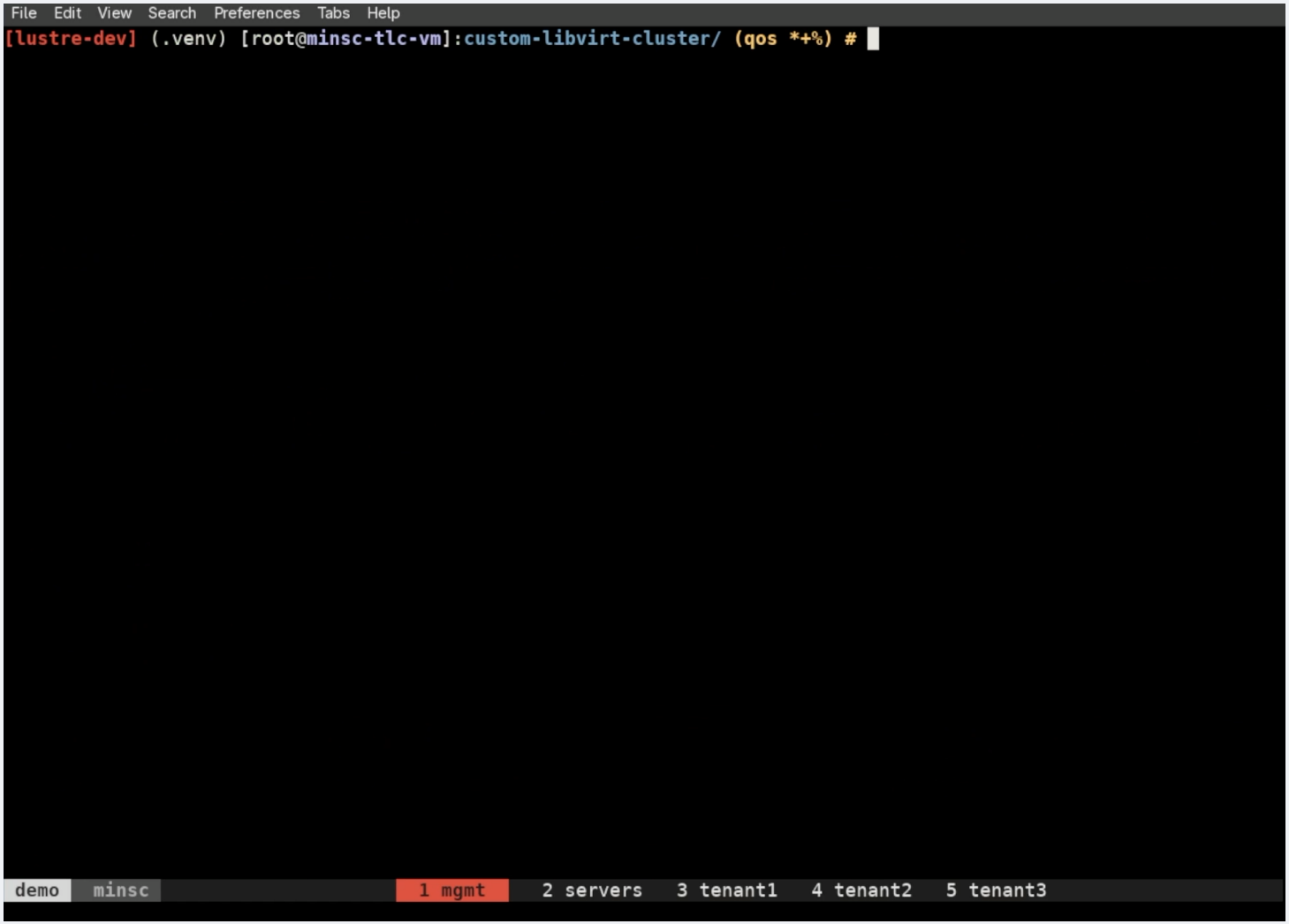
CLI Demo showing adding Server interface + NIDs

Then show client successfully mounting filesystem

df by client showing quota applied



Quotas


File Edit View Search Preferences Tabs Help
[lustre-dev] (.venv) [root@minsc-tlc-vm]:custom-libvirt-cluster/ (qos *+%) #
demo minsc 1 mgmt 2 servers 3 tenant1 4 tenant2 5 tenant3

Tenant I/O Monitoring

[LU-18950] Add nodemap level stats aggregation in Lustre

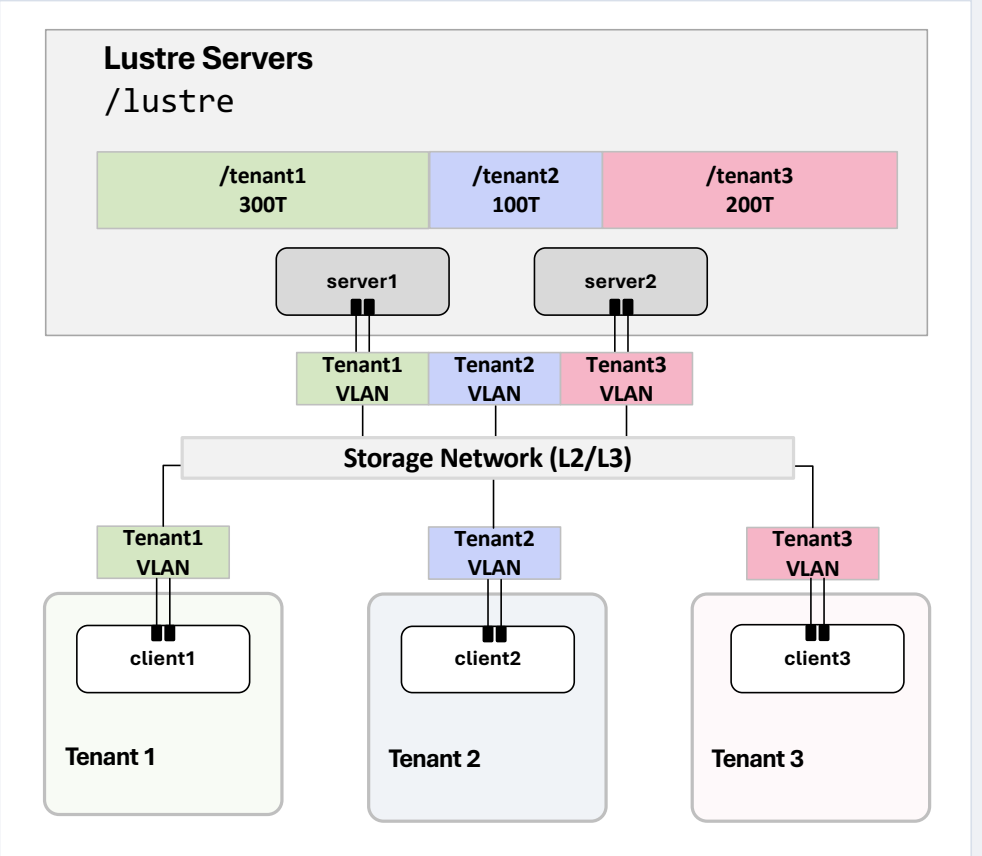
```
lctl get_param nodemap.*.[dt_stats|md_stats]
```

Summary

- Significant multi-tenancy improvements in 2.17
- Tenants (and networks) can be added/removed dynamically
- Tenants have a properly isolated namespace
  - Isolated ID space
  - Ability to self-manage (permissions/quotas)

Additional related improvements

- [LU-18357] Support multiple filesets per nodemap
- [LU-17217] Allow server to control/deny client connections
- [LU-19157] Ban clients from nodemap
- [LU-18756] tag all OST objects with parent FID
- [LU-9555] df show projid quota
- [LU-19034] lfs df show projid quota



Enhancements for Multi-Tenancy in 2.18+

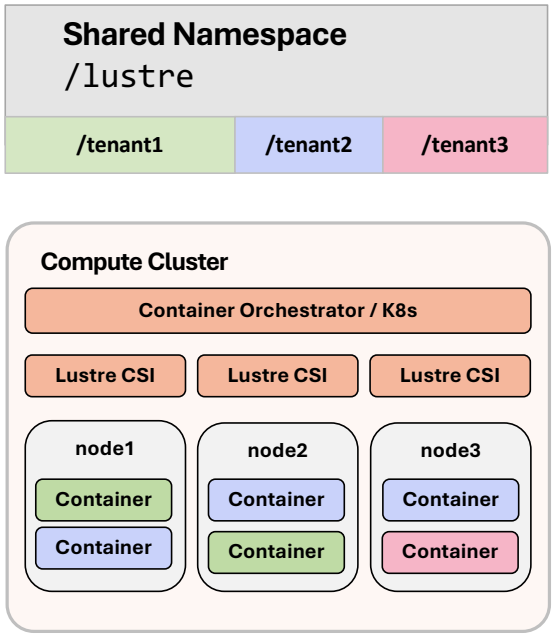
Overview

Remaining Challenges for Lustre Multi-tenancy

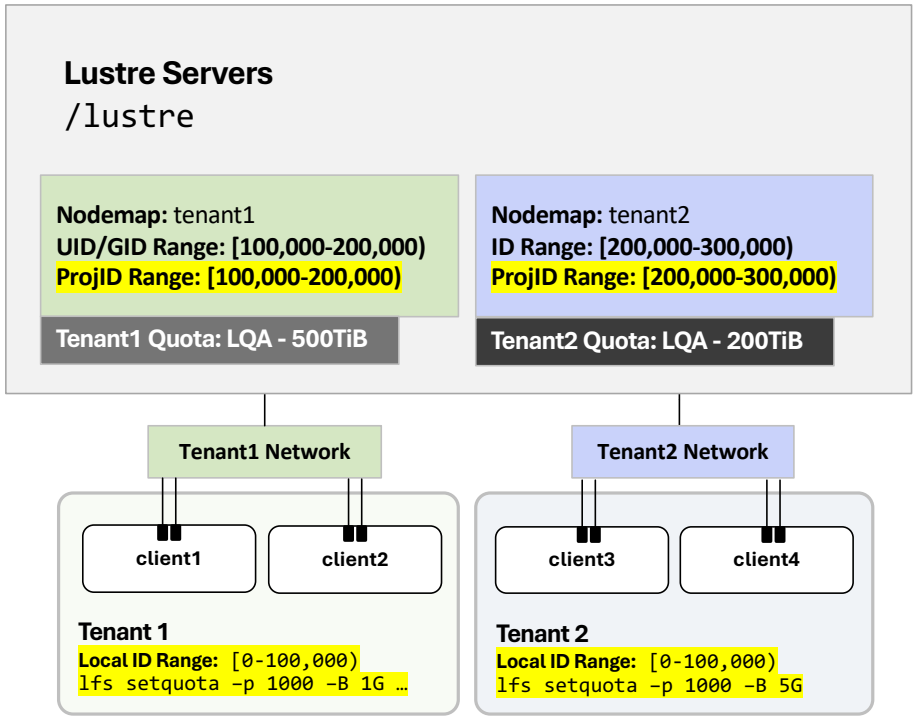
- A single Client (or NID) can only belong to one Tenant
- Lack of strong Tenant authentication
- No performance limits / QoS controls
- Cannot use ProjID within tenants
- Lack of strong Tenant data privacy (Encryption)

Model 02

Intra-node / Container



Tenant Quota - Lustre Quota Aggregation [LU-18222]



- LQA can be used to form hierarchical quotas
- LQA applies aggregate limits on whole Tenant usage
- No longer need to squash ProjIDs within Tenant

Tenant quota enforced via LQA over Tenant ID-range

```
lctl lqa add --fsname lustre --name tenant1 100000-200000
```

Tenant users can independently set UID/GID/ProjID quotas

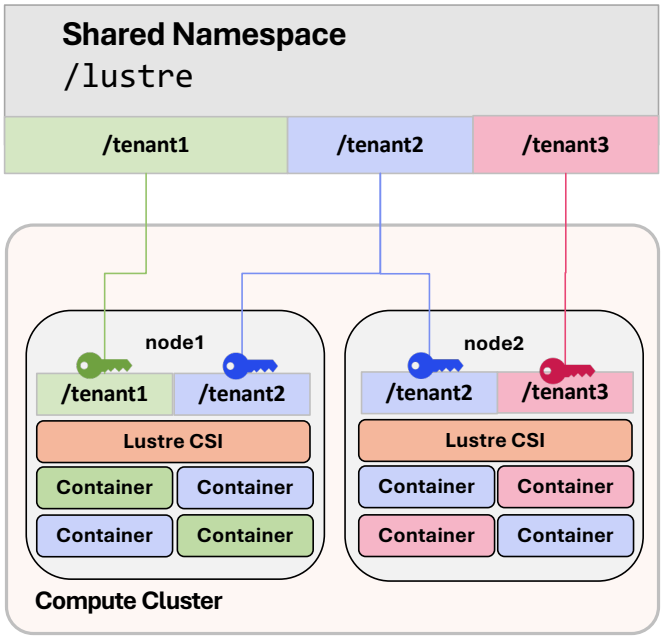
```
lfs setquota -u [0-100,000) -B …
lfs setquota -g [0-100,000) -B …
lfs setquota -p [0-100,000) -B …
```

Tenant Identification and Authentication

LAD 2025 – Changing Nodemap Membership Paradigms – Sebastien Buisson

Model 02

Intra-node / Container



- [LU-19079] GSS for nodemap identification
- Tenants identified through possession of a valid key
- Changes the paradigm of Tenant membership away from Network
- Significantly more flexible – **multiple tenant mounts per client**
- Ideal for Intra-node / Container-based forms of Multi-tenancy

- [LU-19073] ASCII-encoded SSK keys
- [LU-19326] Load SSK key from mount point

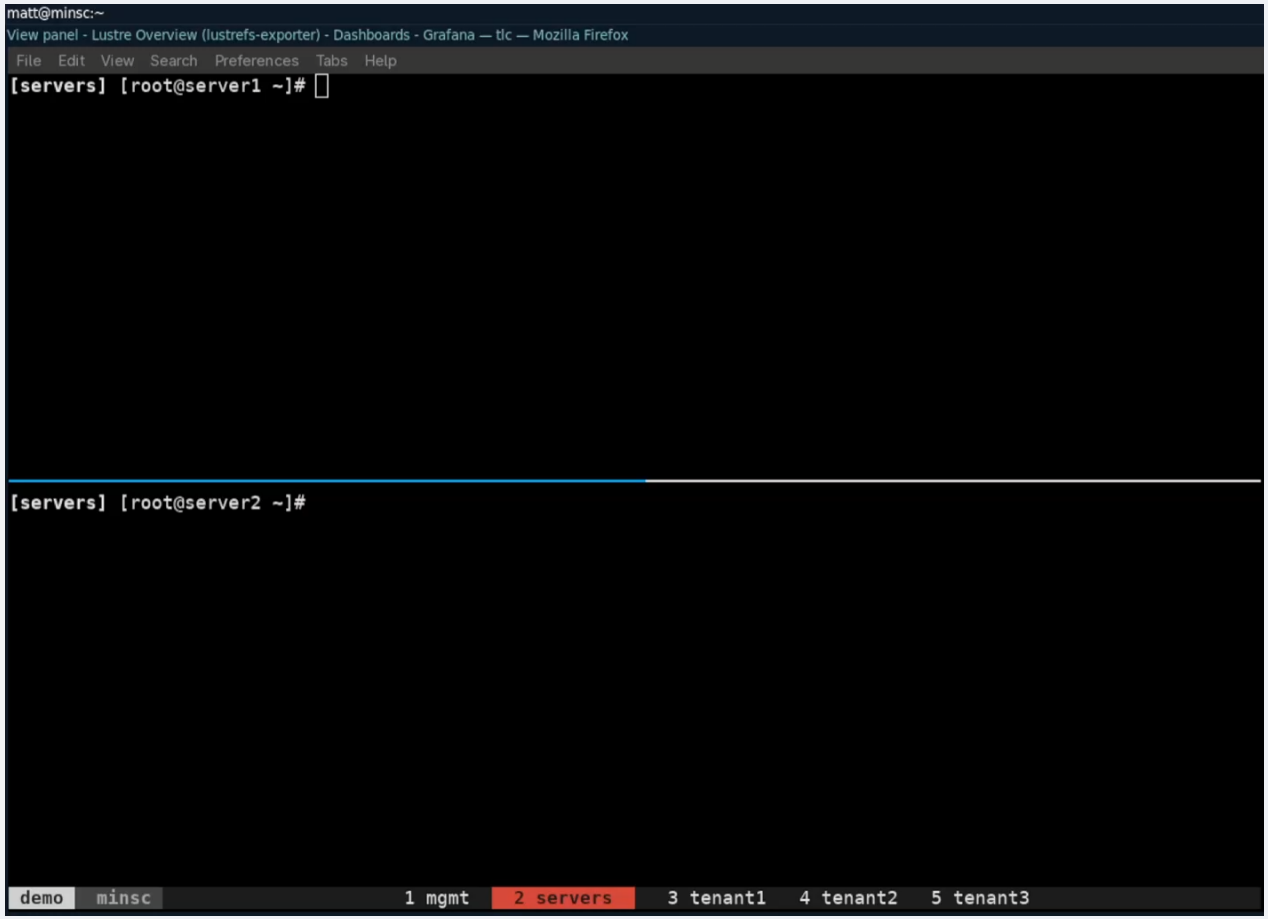
Tenant IO Limiting / QoS

Goal

- Requires mechanism to set IO limits at the tenant level
- Filesystem-wide Bandwidth and/or IOPS limits that restrict Tenant usage

Proposal

- Lustre NRS TBF can be used to approximate this, rate-limit RPCs
- [LU-17902] Add NRS TBF policy for nodemap

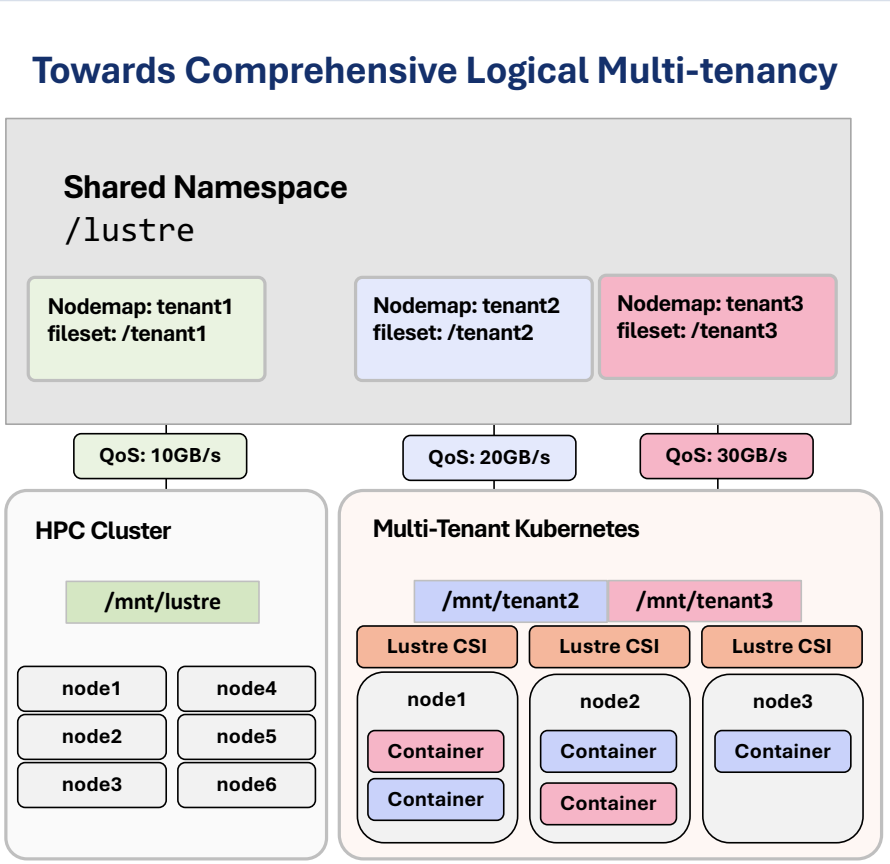


NRS TBF Enhancements for Multi-tenant QoS

- **Nodemap-based TBF policy** (LU-17902)
  Ensure QoS applied to all clients under Tenant
- **Bandwidth-based TBF**
  Expand TBF to track actual data volume instead of only RPC counts
- **Server-global TBF buckets** (LU-17158)
  Decouple TBF limits from CPT partitions – allow single client to use server's entire token allocation
- **Global Limits - Management**
  Simplify management - set single, cluster-wide IOPS/Bandwidth limit, automatically across all servers
- **Global Limits - Token Reallocation**
  Allow busy servers to consume re-allocated tokens from idle servers – token redistribution across cluster
- **TBF Telemetry** (LU-13037)
  Provide detailed, per-bucket TBF metrics (tokens consumed, requests delayed, average queue latency)
- **Congestion Fair-share / Anti-starvation** (LU-18269)
  Ensure interactive and low-volume users are not starved by heavy workloads during server congestion
- **(HPC) Work-conserving soft-limits (Idle-bursting)**
  Allow tenants to exceed their configured QoS limits when underlying server hardware is idle

Summary and Outlook for 2.18+

- Multi-tenancy in Lustre is already highly capable
- Several enhancements in 2.17 and under development will further improve flexibility and useability – particularly for container-based workloads
- Tenant-level QoS is the next major capability gap
- Enhancements to NRS TBF will improve manageability of Lustre QoS for many use-cases
- Towards a first-class, Lustre-native multi-tenancy capability



References

- **Lustre Operations Manual**
  - Ch 28: Mapping UIDs and GIDs with Nodemap (L 2.9+) – doc.lustre.org/lustre_manual.xhtml#nodemapID
  - Ch 30: Managing Security – Lustre Isolation / Multi-tenancy – doc.lustre.org/lustre_manual.xhtml#managingSecurity
- **LUG Conference Presentations**
  - LUG 2025 – “Lustre Multi-Tenancy” (S. Buisson) – srcc.stanford.edu (PDF)
  - LUG 2018 – “Multi-tenancy: a real-life implementation” (S. Buisson) – wiki.lustre.org (PDF)
- **Blog / Whitepaper**
  - DDN Blog – “Mastering Isolation and Multi-Tenancy on Lustre File System” – ddn.com/blog
  - Wellcome Sanger – “Lustre for OpenStack” whitepaper (routing + multi-tenancy)
- **Jira Tickets**
  - LU-19692, LU-19733 – Dynamic NID bug fixes (on master)

Agenda

- Models of Multitenancy seen today in HPC/Cloud environments
  - Node-based multitenancy (tenant = groups of isolated compute)
  - Container-based multitenancy (tenant = groups of containers under K8s)
- How does it look today in 2.17?
  - Node-based (filesets + nodemap + id-offsets + projID quota + RBAC) = tenant
    - Demo what it looks like. Show enforced subdirectory isolation, show quota enforcement (show ‘df’ displaying tenant quota/usage). Show delegated local root capabilities (permissions, quota)
    - Show how we monitor it – nodemap metrics -> lustrefs-exporter -> Grafana plots
- Future