Microsoft

Delivering Lustre 2.15 file system as a HPC cloud service

Dilip Sundarraj, Principal Product Manager

Ellis Wilson, Principal Engineering Manager

Azure: the cloud purpose-built for HPC & AI

- ✓ **Genuine HPC approach** platforms, benchmarks, people, and end-to-end experience
- ✓ **Purpose-built platforms** for performance, and price-performance, and differentiated solutions
- ✓ **Leading time-to-market for key hardware innovations** to accelerate time-to-solution for customers
- ✓ **Partnering with customers** for the long term to solve HPC and business needs

Supercomputing for the most demanding applications

InfiniBand HPC & AI clusters for best performance on real workloads

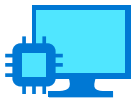
Compute optimized VMs with "low" latency networks

High Performance Storage optimized for HPC/AI

Azure

Azure offers the full range of HPC and AI capabilities

HPC Resource Stack on Azure



Optimized Compute

H-Series N-Series Cray

High Performing Storage

Azure HPC Cache Azure NetApp Files ClusterStor **Azure Managed Lustre**

Workload Orchestration

VM Scale Sets Azure Batch Azure CycleCloud Azure Kubernetes

Fast, Secure Networking

ExpressRoute InfiniBand

Transformative Services

Azure Machine Learning Azure Data Lake Azure ML Compute

Solve any HPC, AI workload—at any scale

A/B series VMs

Burstable virtual machines (VMs)

General-purpose VMs

D: Standard workloads

E: High memory

F: Compute-bound

High memory VMs

L: High SSD & IOPS

M: Extreme memory

Specialized VMs

H: High memory

HB: Memory bandwidth

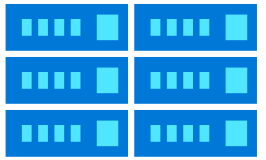
HC: Compute-bound

NC: GP-GPU compute

ND: Deep learning

NV: Graphics applications

NP: Programmable FPGA



Cray in Azure

Managed custom bare-metal server

Large to extreme-scale HPC infrastructure

Azure network integration

Small scale MPI

(Handful of cores)

Extreme scale MPI

(100k+ cores)

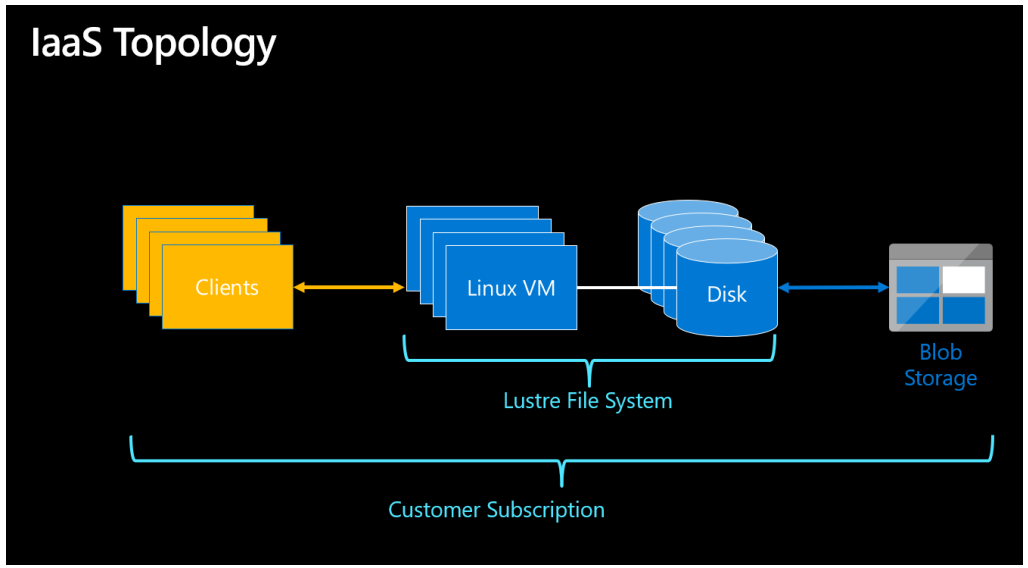
Lustre IaaS to Managed Lustre

Lustre in the Cloud

DIY on Azure IaaS

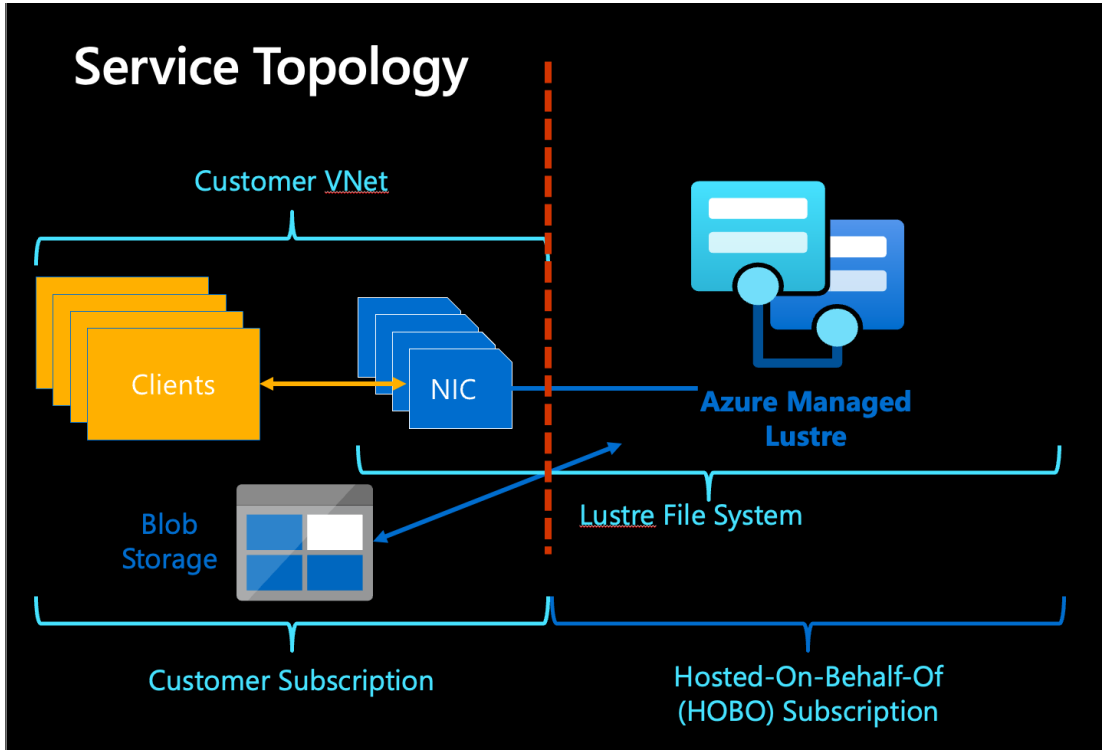
1. Custom deploy VMs, attach disks and Blob containers
2. Deploy Lustre
3. Build Lustre client images for client VMs
4. Build & deploy tooling to monitor Lustre
5. Build & maintain Lustre cluster

DIY everything but hardware!



Azure Managed Lustre

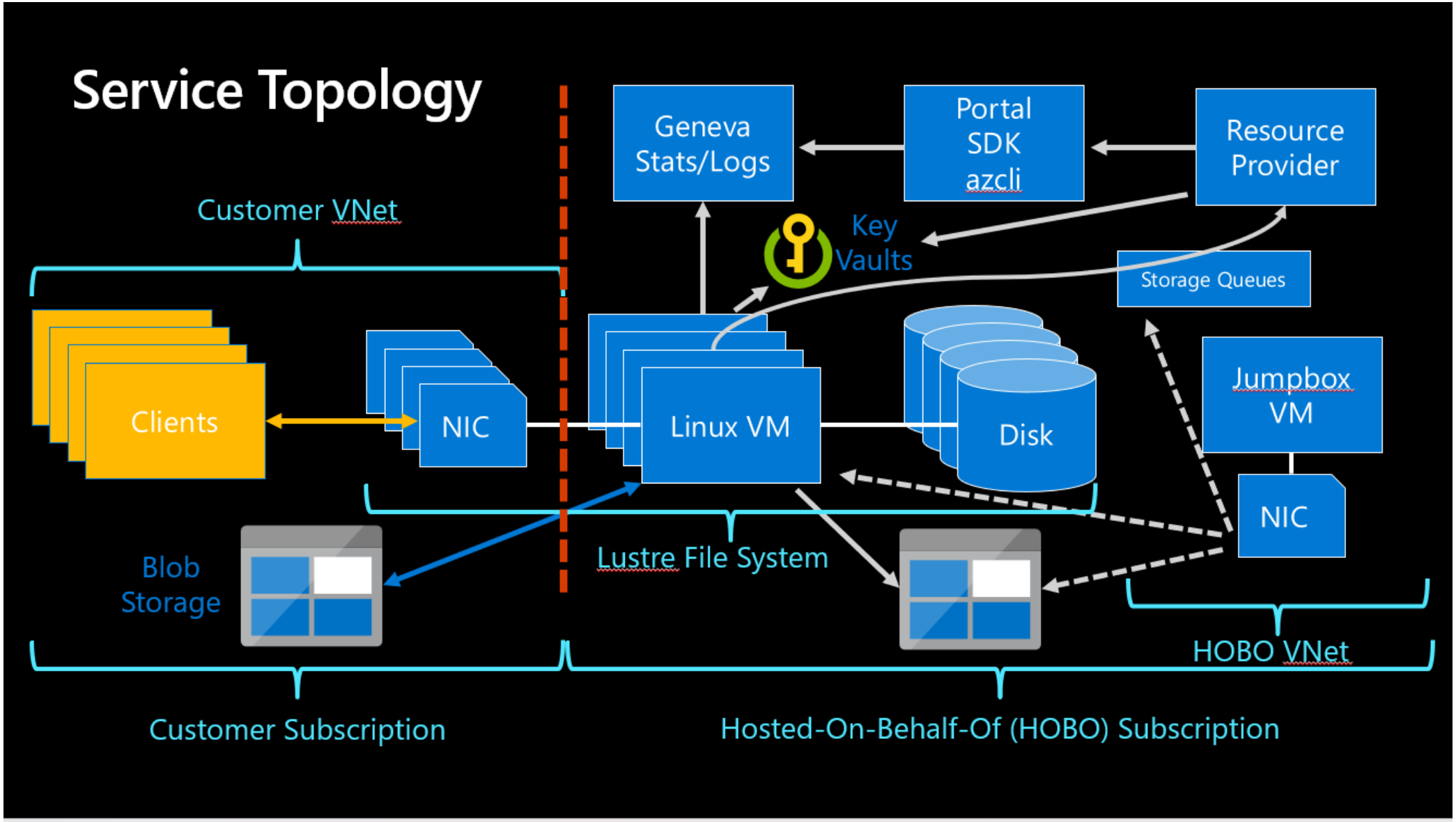
1. Identify a virtual network to deploy AMLFS
2. Configure & click deploy through Azure portal, REST or automation templates
3. Download & Install pre-built lustre client packages from packages.microsoft.com



Fully Managed

deployment, operation & maintenance

How do we deliver Azure Managed Lustre



Learnings from deploying Lustre 2.15.x

Deploying Lustre in a secure Azure Cloud Platform

1. Linux updated with security patches - every 2 months or sooner for all cluster nodes globally.
   - Address CVEs or any other vulnerabilities
2. All Lustre server nodes are actively monitored for known vulnerabilities by published standards based on the OS version and software running
3. Data at rest is encrypted with no additional overhead
   - Customer option to manage their own encryption keys (CMK)
4. Data is encrypted as it travels between a Lustre server node and the storage medium (EncryptionAtRest)
5. GDPR compliance - for managing personal or sensitive data
6. We expose the minimum number of ports and cluster-side services possible to avoid possibility of security threat
7. We plan to offer end-to-end encryption that does not require Lustre's built-in client-side encryption at no extra cost and at no performance penalty

Engineering Experiences with 2.15

- What we're running:
  - Server: 2.15.1 + MS patches
    - Ubuntu 18.04 today, 20.04 at release after GA
    - MS Lustre patches to be upstreamed at GA
  - Clients: 2.15.1 + backports
    - Patches from 2.15.2/master as-needed
  - Move to 2.15.2 soon

- Cloud: A Living Environment
  - Need to act within week(s) to cure CVEs on both server- and client-side
  - Support ready-to-go client packages:
    - RHEL 7.8/7.9
    - RHEL/Alma 8.x
    - Ubuntu 18.04/20.04/22.04 (5.15 kernel and lower)

- 2.15 Experiences so Far:
  - Stability remains high
    - Since Nov 2022; on 2.14 previously
    - Zero crashes, kernel hangs, or corruption so far
    - We stand up a lot of Lustre clusters in testing and production to exercise stability
      - >100 unique Lustre clusters deployed per day on average (for testing + production)

  - Performance on our most common workloads equivalent to 2.14

  - Improved support for newer kernels
    - Still a lot of work getting new distros stood up
    - Continual treadmill to track new kernel versions
    - Employ Azure pipelines to build and deliver new client packages at least twice daily for every distro

Lustre Improvements and Bug Fixes

- All to be upstreamed shortly after our GA of AMLFS
  - This is just a selection of interesting improvements and fixes
- Improvements:
  - PFL striping support for HSM Import
  - HSM Action-selective purge
  - HSM Import Performance: Elide repetitive mount checks and FID translations
  - HSM Agent Load Balancing: Improved distribution of work for large agent pools
  - Build/patch support against ldiskfs on newer kernels
- Bug Fixes:
  - ksocknal_handle_{link_state_change,inetaddr_change} ignores index of device namespace
  - HSM Purge sends CANCELs to the incorrect agents
  - lproc stats display in wallclock time rather than uptime
  - Test Bugs: Variety of auster test fixes for unique cloud/ubuntu server environment

Cluster Maintenance & Upgrades

Cluster Maintenance & Upgrade

Purpose

- Weekly maintenance windows - change configurations, apply hot fixes, and perform full upgrades on the fleet of AMLFS clusters.
- Upgrading a Lustre cluster involves multiple layers
  - Lustre kernel filesystem driver, user space suite of utilities, libraries, and services:
    - Platform/System level software that comes from an upstream source SBI/Ubuntu
  - Azure Managed Lustre specific applications and support code for cluster node management and reporting

Goals

1. Customer data must be preserved
2. Upgrade must operate efficiently affording the customer minimal downtime with a target of 30 minutes or less.
3. Existing and new configuration must be applied to the cluster nodes
4. Upgrade must be able to either complete or roll back such that once the maintenance window is complete the customer still has a working filesystem.

How do we upgrade?

We leverage the power of **Azure OS Disk Swap** to replace the new image without the need to destroy and recreate the cluster VMs.

Key Advantages:

1. No need to detach/attach data disks (or) create new VM instances, fight with quota
2. Backing block devices automatically re-assemble post-upgrade.
3. Handle OS change epochs like Ubuntu 18.04 to 20.04 easily

Upgrade procedure

1. Perform suspend prepare actions (adjust timeouts, enable server-side firewalls)
2. Create new OS disks for each node in the cluster using the new release image.
3. Use the new PUT on the VM specifying a different OS disk resulting in the VM being stopped (but not deallocated), OS disk swapped, and the VM being booted.
4. Wait for boot and validate health of server nodes/mounts.
5. Perform unsuspend actions (restore normal timeouts, disable server-side firewalls)
6. Delete old OS disks on success, or restore old OS disks on failure

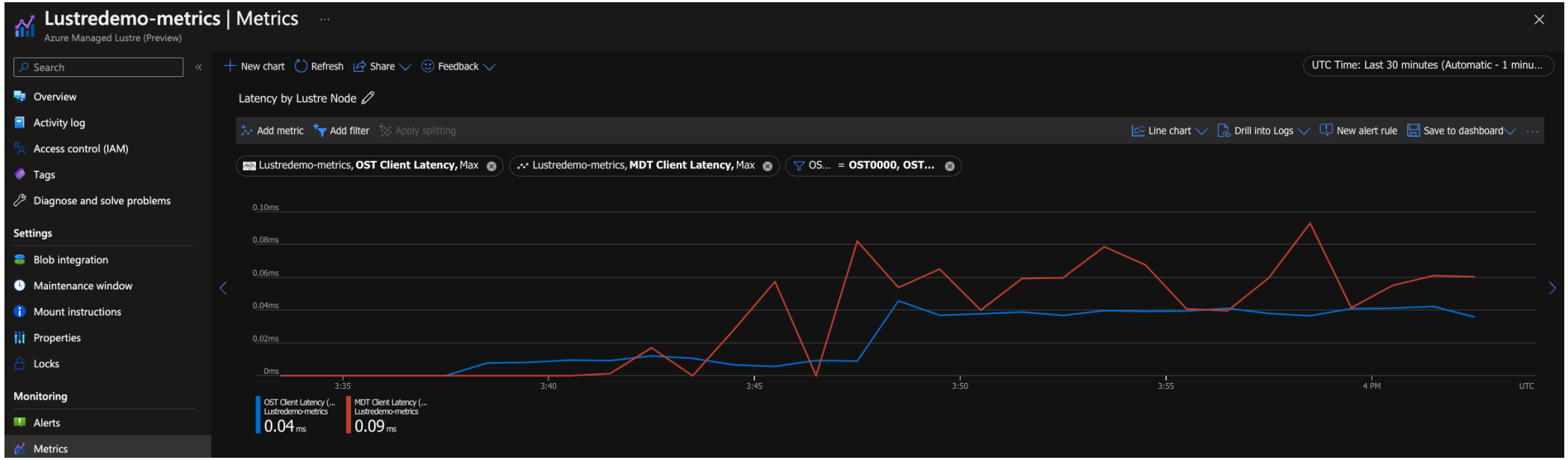
Monitoring and Operating multiple Lustre clusters

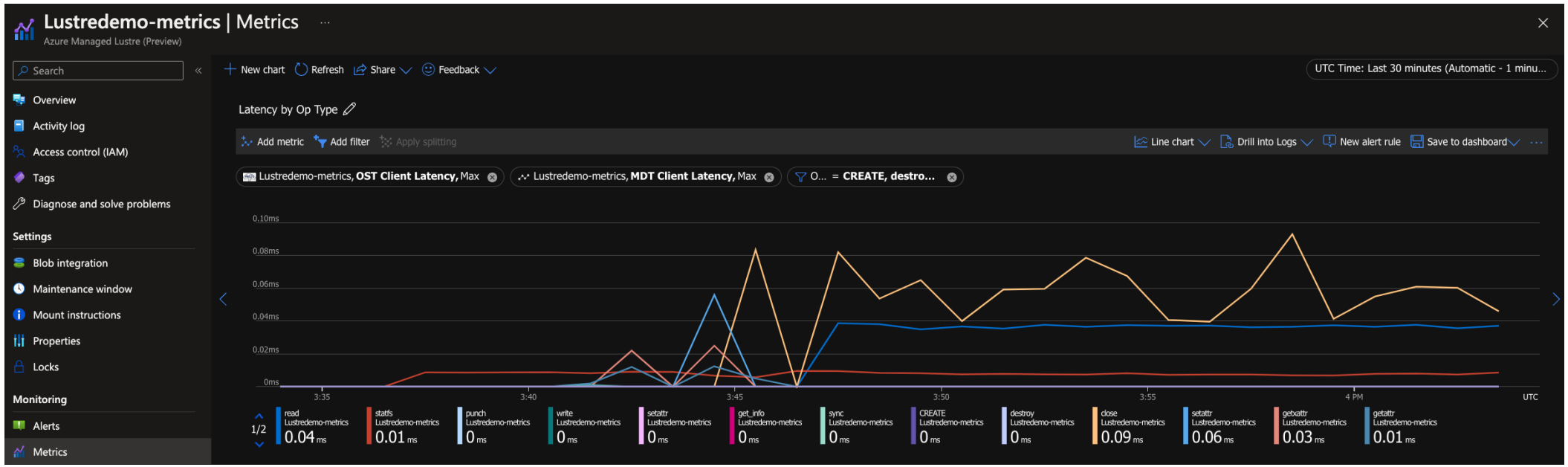
Customer visible metrics on Azure Portal

Customers can view the following cluster relevant metrics on their Azure Portal.

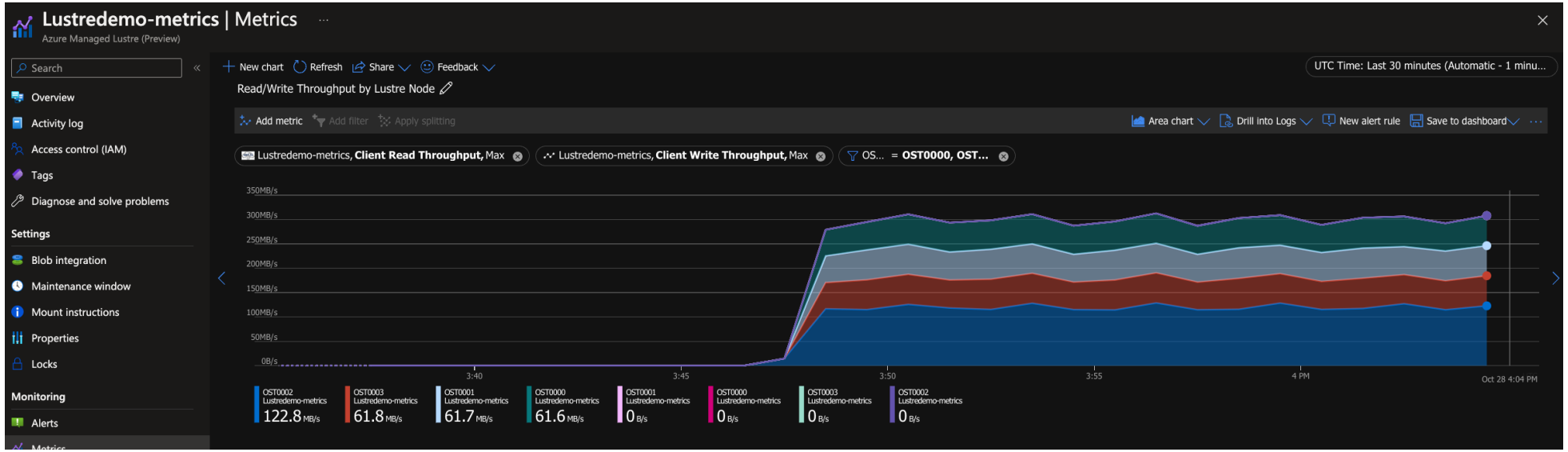
1. Latencies by Lustre Node or Op Type
2. Read/Write Throughput by Lustre Node
3. OPS per Lustre Node or Op Type
4. Capacity per OSS and MDS

Latencies by Lustre Node or Op Type

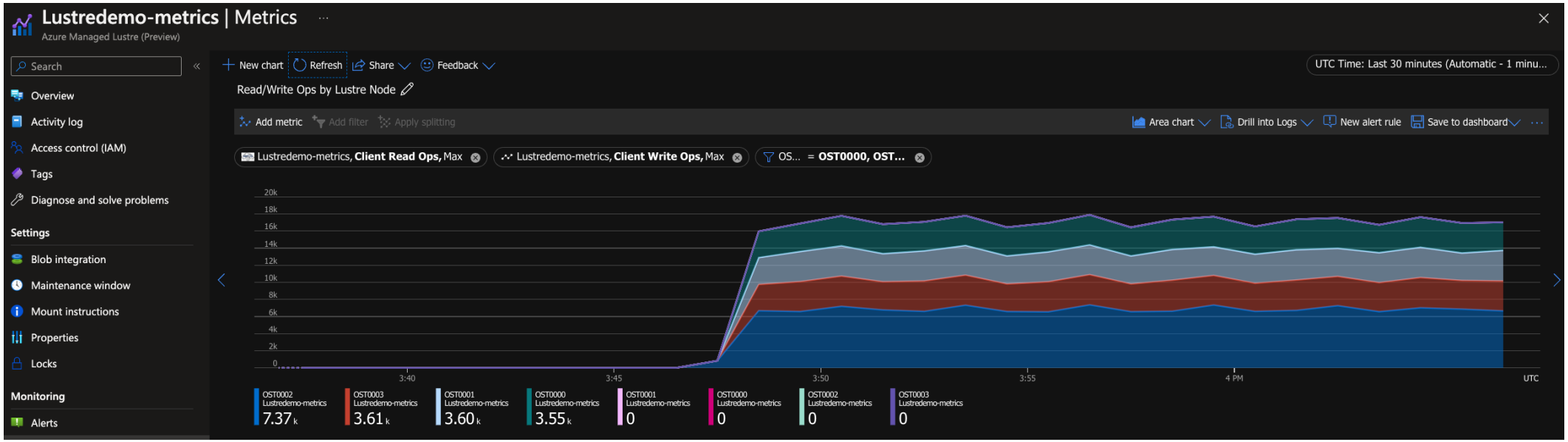


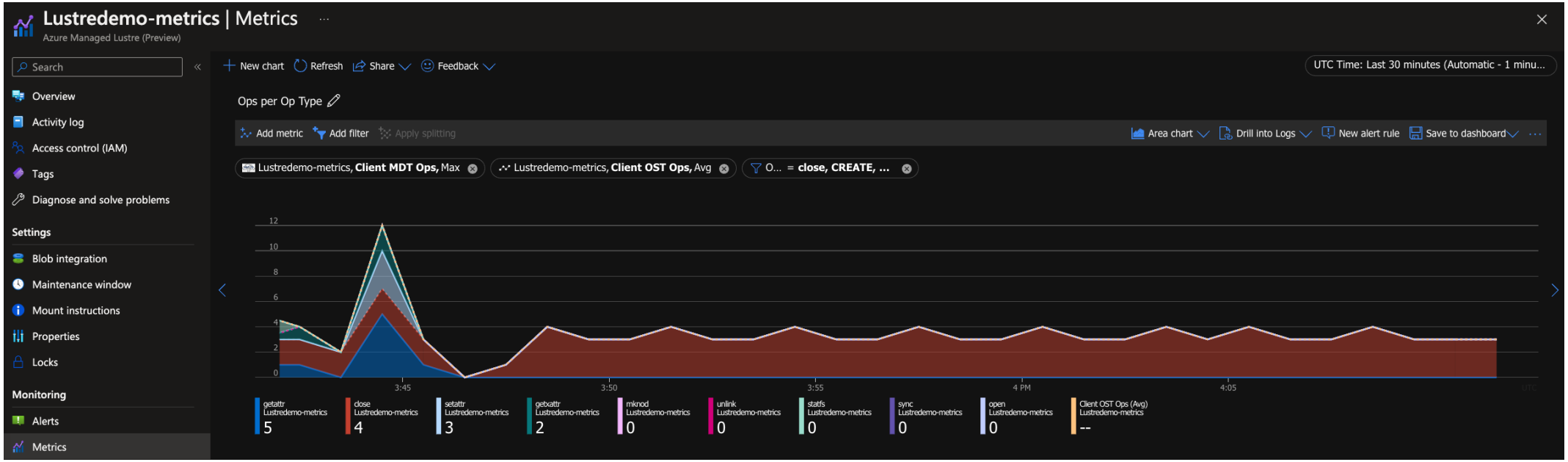


Read/Write Throughput by Lustre Node

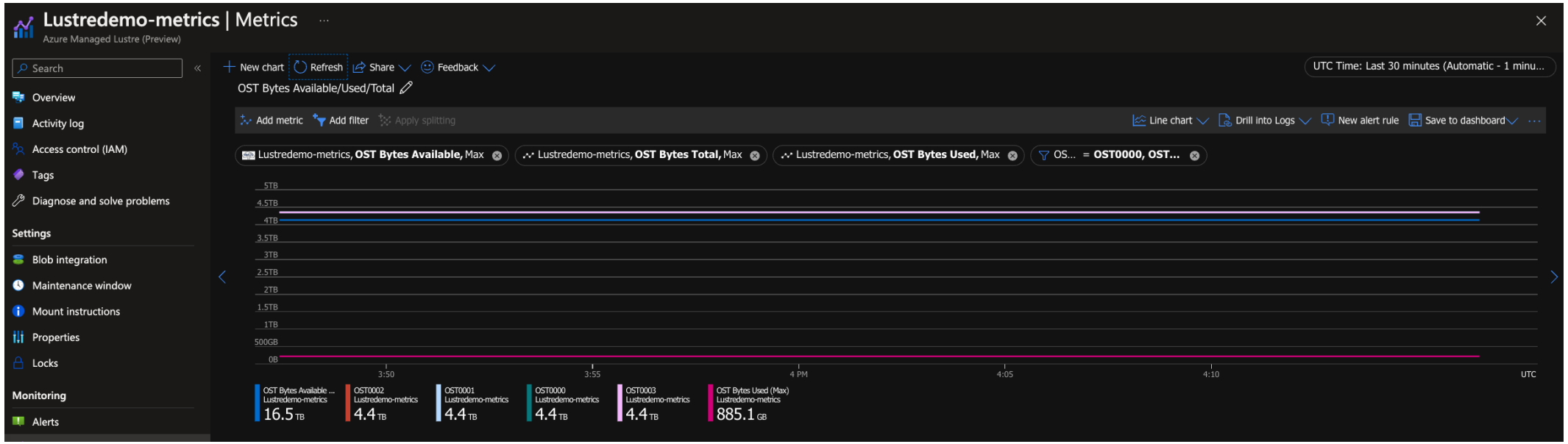


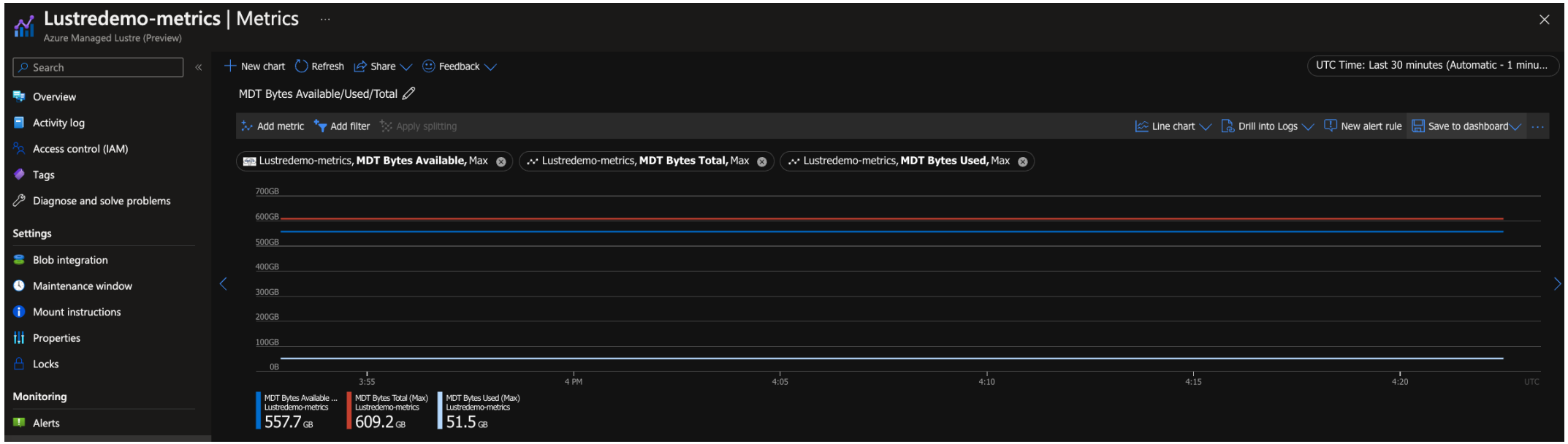
OPS per Lustre Node or Op Type





Capacity per OSS and MDS

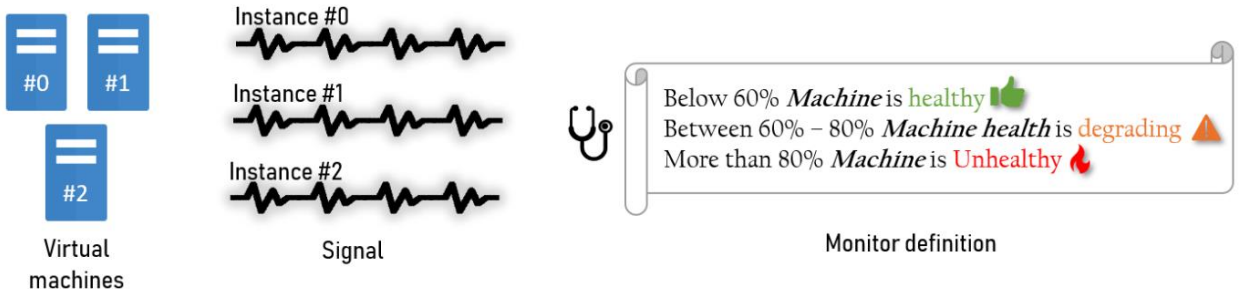




Stats actively tracked

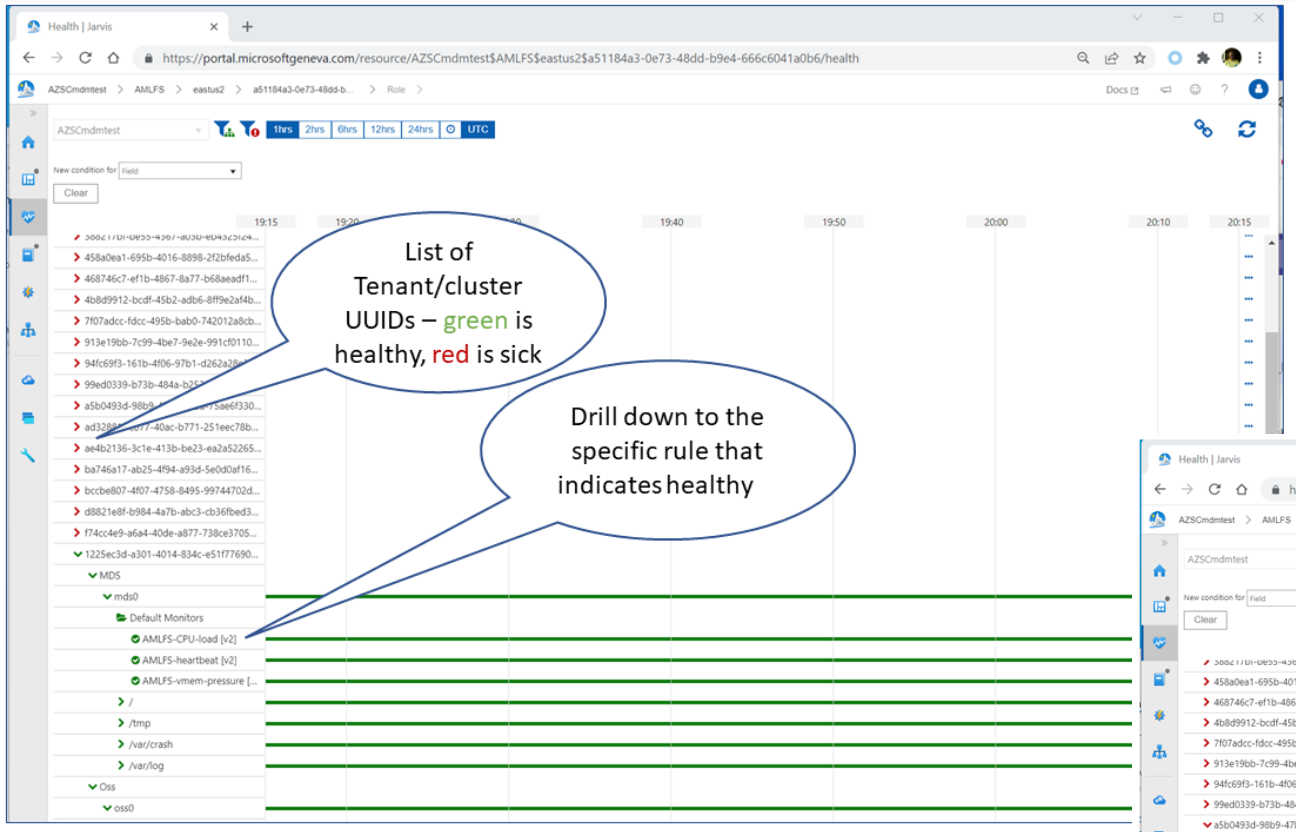
Collection	Cluster NodeTypes	Infra NodeTypes	Metrics prefixes	Source	"Component" Dimension Value
Guest OS	all	all	net_,vm_,cpu_,disk_	get_os_stats()/metrics_tool	OS
Changelogs	MDS	None	changelogs_	load_changelog_stats()/metrics_tool	changelogs
HSM	MDS	None	mdt_hsm_	lustre_stats.py	hsm
Hydrator	Primary Agent	None	hyd_	hydratorstats.py	hydrator
MDT	MDS	None	mdt_	get_mdt_stats()/metric_tool	mdt
OST	OSS	None	ost_	get_ost_stats()/metrics_tool.py	ost
NIC	All	All	net_	get_os_stats()/metrics_tool	OS
DataDisk	MDS/OSS/AgentPri	None	disk_	get_os_stats()/metrics_tool	OS

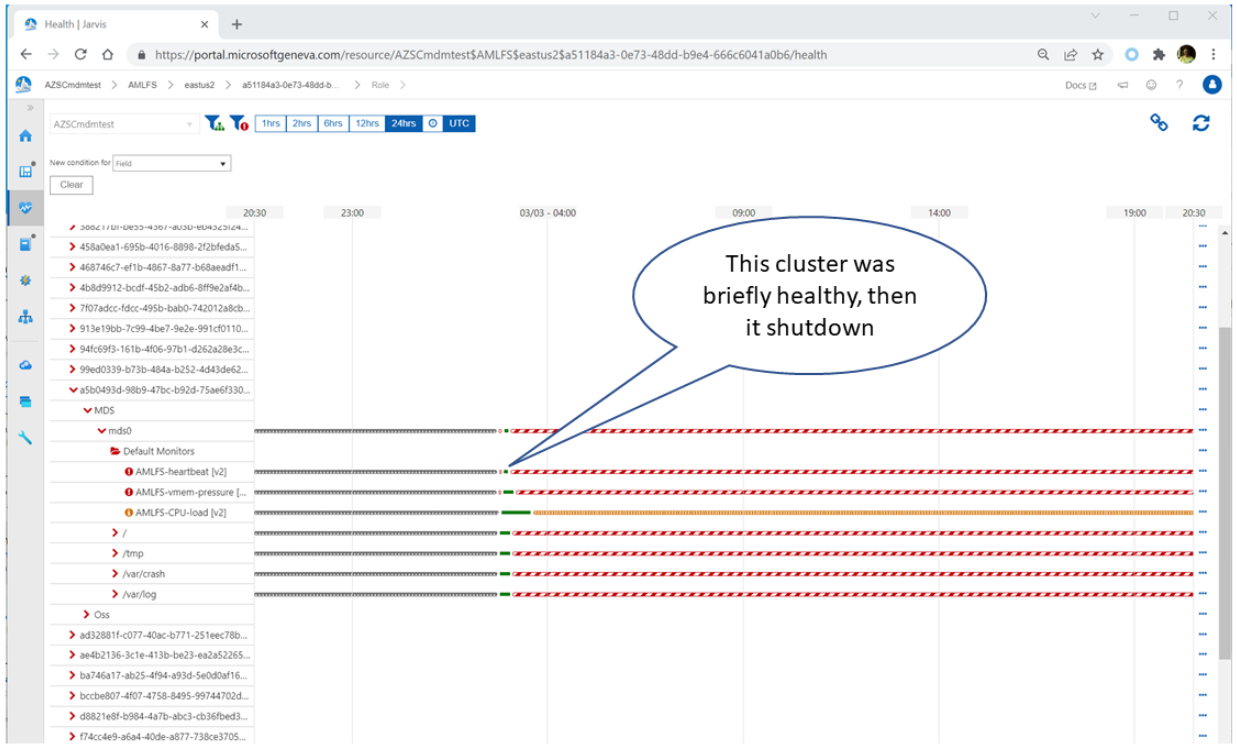
Monitors & Alerts



Component	Metric	Failure Alert	Performance Alert
OS/Process	healthchecker-restart	yes (above threshold)	no
OS/services	service restart	yes (above threshold)	no
OS/CPU	cpu utilization	no	yes (cpu above threshold)
OS/Disk	disk spaceused	yes (out of space)	yes (approaching OOS)
OS/NVME	read throughput	yes (below threshold)	no
OS/NVME	write throughput	yes (below threshold)	no
OS/Networking	packets dropped	yes (above threshold)	yes (above threshold)
OS/Networking	write throughput	no	yes (below threshold)
OS/Networking	read throughput	no	yes (below threshold)
hydrator	errors	yes (above threshold)	
hydrator	throttled	yes (above threshold)	
hydrator	xattr_fail	yes	
hydrator	wrong_ftype	yes	
lustre	ost_op_lat_usec_total	no	yes

AMLFS Health Monitoring Dashboards

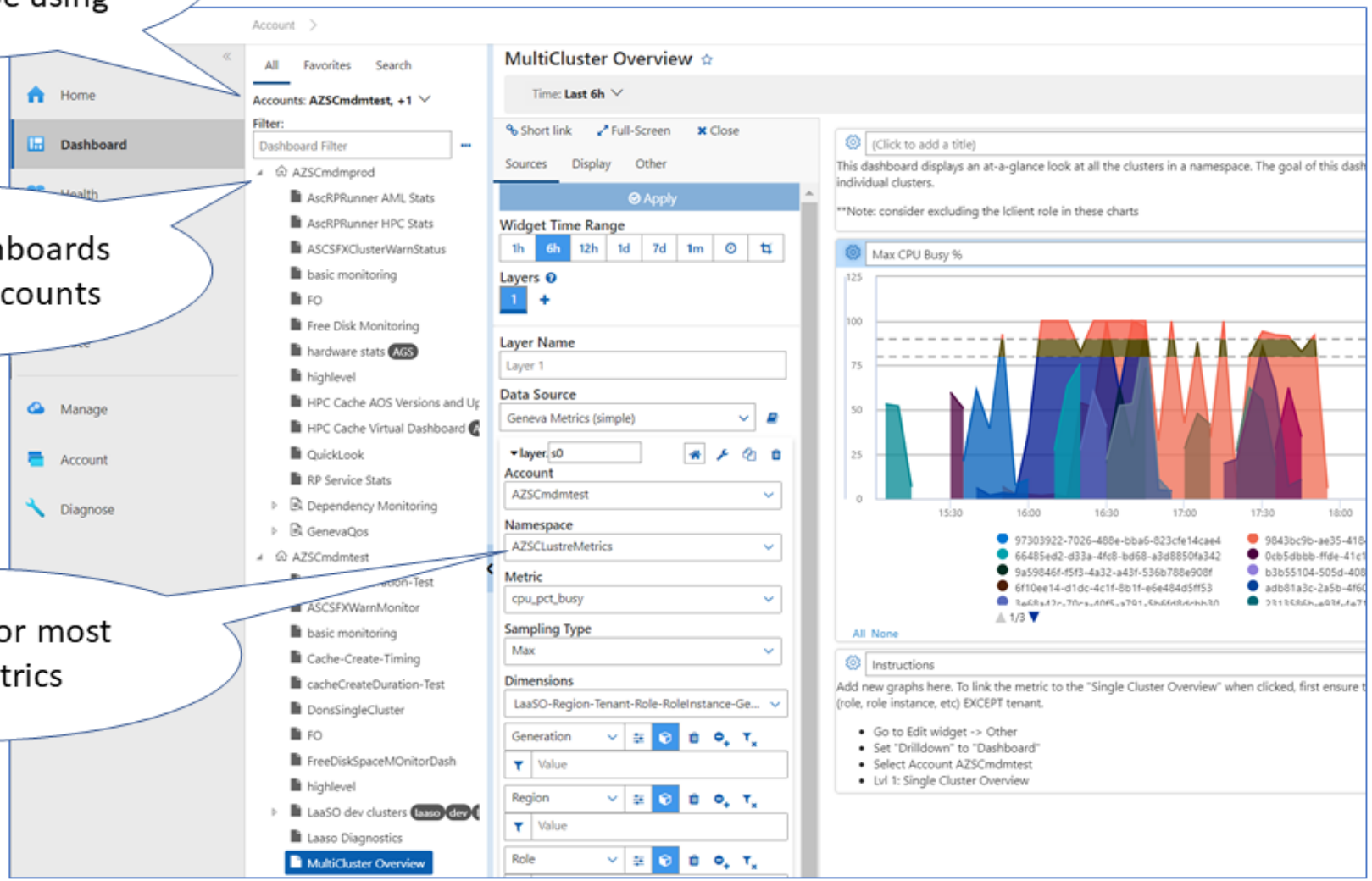




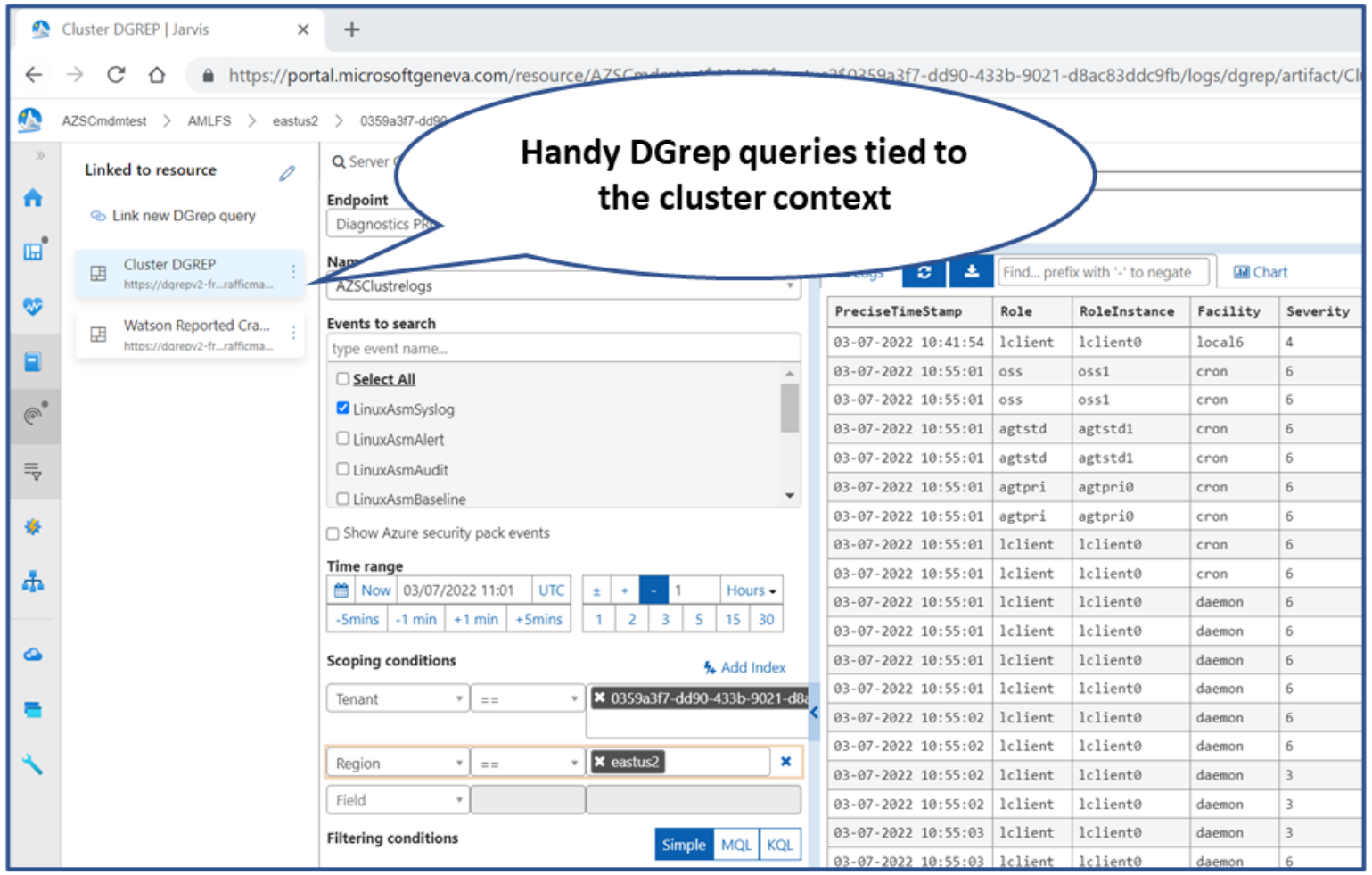
List the metrics accounts you'll be using

Pre-defined dashboards for the active accounts

Namespace for most LaaSO metrics



Distributed log tracing – DGrep



Microsoft

Thank you