

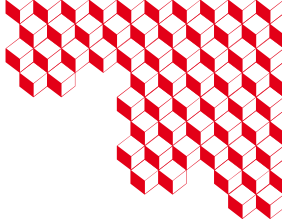


Lustre Breaker

It works on my cluster...

Quentin ANIÈRE

October 1, 2026 - **LAD26**



CEA's computing centres

CEA HPC resources for Research, Industry and Defence

1 site (CEA/DIF), 2 facilities and 4 computing centres,

<https://www-hpc.cea.fr/>

**TGCC
Facility**
(Open science)



Public research
France (GENCI) and Europe (EuroHPC)

Joliot-Curie
22 Petaflop/s

Alice Recoque
1 Exaflop/s

TOPAZE
10 Petaflop/s



CEA and Industrial partners
(CCRT)



OCRE
R&D Zone

**TERA-EXA
Facility**
(Defence)



CEA/DAM Defence usages
(EXA)

EXA1-HE
180 Petaflop/s

EXA1-HF
36 Petaflop/s

ECRIN
4 Petaflop/s



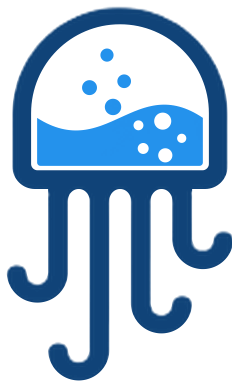
CEA and Defence partners
(CCMD)

Storage cluster deployment

- A lot of computing centres
- Different hardware
- Different teams

→ **CEA built its own stack (Ocean) to standardize deployment and operations**

- 3 major releases maintained in parallel (EL8, EL9, EL10)
- Available at <https://ocean.eupex.eu>



Many moving parts

Each release of the Ocean stack needs a coherent set of

- Linux kernel
- Interconnect drivers (Mellanox / DOCA)
- Lustre version

→ **Any change needs to be re-tested on every release**

On top of Lustre, the HSM stack:

- Policy engine: *Robinhood* (<https://github.com/cea-hpc/robinhood>)
- Copytools, Coordinators
- Archive backends: *Phobos* (<https://github.com/cea-hpc/phobos>), *HPSS*

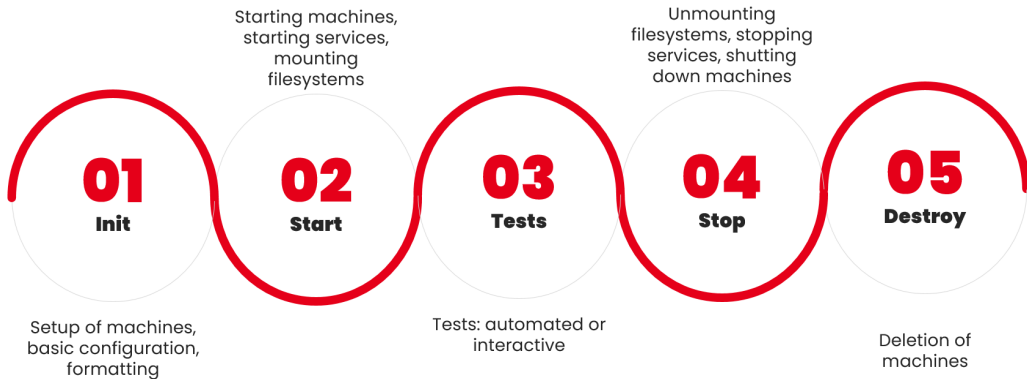
→ **Many interacting components, from different sources**

We need a tool

- Deploy disposable storage clusters automatically
- Reproduce production-like setups, on demand
- Test every layer of the storage cluster
- Run non-regression tests

→ **Lustre Breaker**

Running a cluster





- Lustre Breaker is not tied to a specific technology
- An abstract notion of machine, implemented by backends
 - **PCOCC**: runs on HPC clusters, allows high performance and access to specific hardware (InfiniBand cards)
 - **Libvirt** (WIP): virtualization on top of QEMU, can run on most machines (even my laptop...)
 - **Bare metal** (future): would allow testing on real hardware and running benchmarks (e.g. *mdtest*, *IO500*, ...)

Configuration

```
machines:
  - name: fs01
    image: ocean-8.10
    roles: [mgs, mds]
    disks:
      - path: mgt.raw
        size: 1G
      - path: mdt.raw
        size: 1G

  - name: oss01
    image: ocean-8.10
    roles: [oss]
    disks:
      - path: ost1.raw
        size: 5G

  - name: client01
    image: ocean-8.10
    roles: [lustre_client]
    disks: []
```

Deploying with Ansible



- Ansible is an automation tool
- It applies configuration to the machines, over SSH
- A role: a self-contained, reusable unit, with its tasks, templates and variables
- In Lustre Breaker:
 - The user passes parameters to Ansible in the configuration file
 - Ansible generates the configuration from these parameters
 - Sensible defaults are provided
 - One role per component (mgs, client, nfs, ...)

Lustre server Ansible role

```
machines:
  - name: fs01
    image: ocean-8.10
    roles: [mgs, mds]
    ansible_vars:
      work:
        mgt: /dev/vdb
        mdt: /dev/vdc, /dev/vdd
```

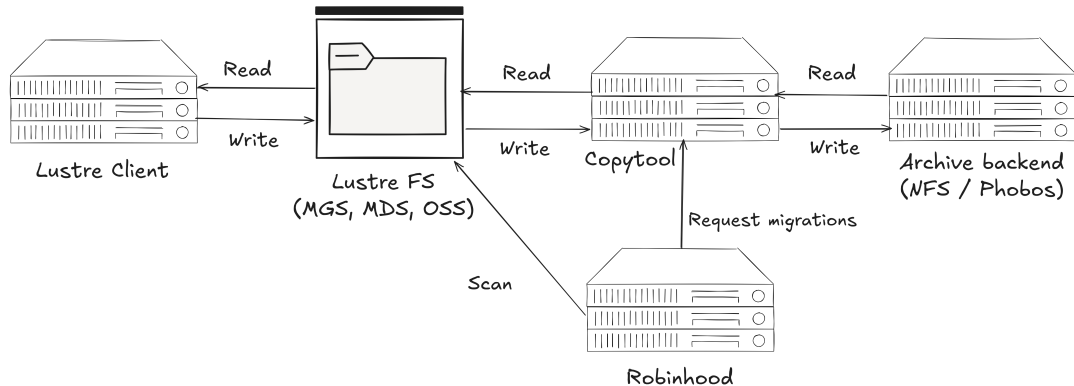
- One machine, several roles: fs01 is an MGS and an MDS
- For the work filesystem, it hosts the MGT and two MDTs

Configuring the filesystems

```
filesystems:  
  - fs_name: work  
    description: Work filesystem  
    quota: yes  
    lnet_net: tcp0  
  
  - fs_name: scratch  
    description: Scratch filesystem  
    stripe_count: 8
```

- Used to generate the Shine configuration
- Only `fs_name` is required; the other keys fall back to the Shine default configuration
- By default, the clients mount every filesystem, unless a list is given

Available roles



Running tests

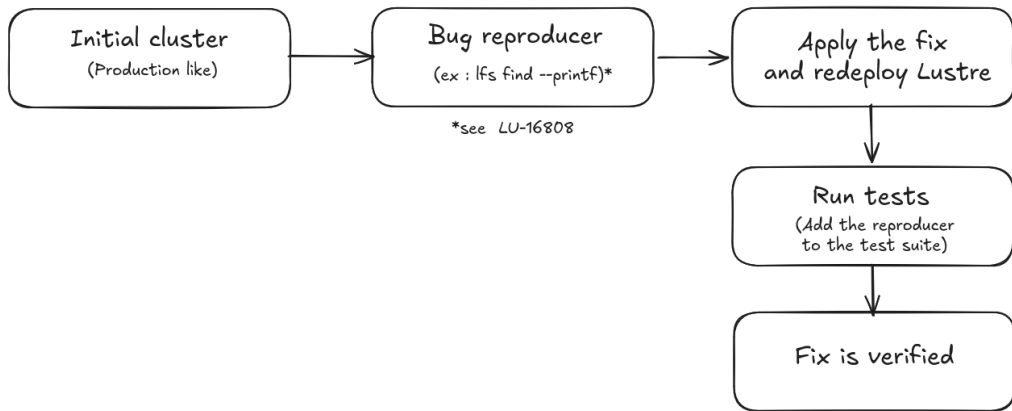
```
name: "Test FS init"
run_on: "client01,client02"
description: "Check if the work FS is mounted on the clients"

script: |
    #!/bin/bash

    df -T /work | grep lustre
```

- A test: a name, a machine to run it on, and a script
- Runnable on:
 - A single host (e.g. client01)
 - Several hosts (e.g. client01,client02)
 - A ClusterShell group (e.g. @all)

Reproducing a bug and verifying a fix



→ **Around 15 min to recompile, build the RPMs, umount, upgrade, mount**

The forge



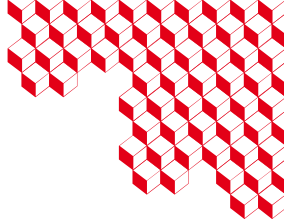
- A GitLab instance for CEA software: Phobos, Robinhood, Ocean
- Use Lustre Breaker to test at different stages:
 - When developing software: archive backends, policy engine...
 - When integrating software: the CEA Ocean stack
- GitLab runners on a CEA HPC cluster: A lot of resources to use Lustre Breaker

Conclusion

- Start an entire Lustre/HSM cluster with a single YAML file
- Replicate complex production setups easily
- Non-regression tests, written as simple YAML files
- From a reproduced bug to a verified fix, in 15 minutes

What's next

- More backends: Libvirt, Bare metal
- Complex tests: steps, dependency (GitLab CI-like)
- AI sandbox:
 - Diagnose the failures
 - Take corrective action
- Open source it



Any questions?

CEA - LAD26

`quentin.aniere@cea.fr`