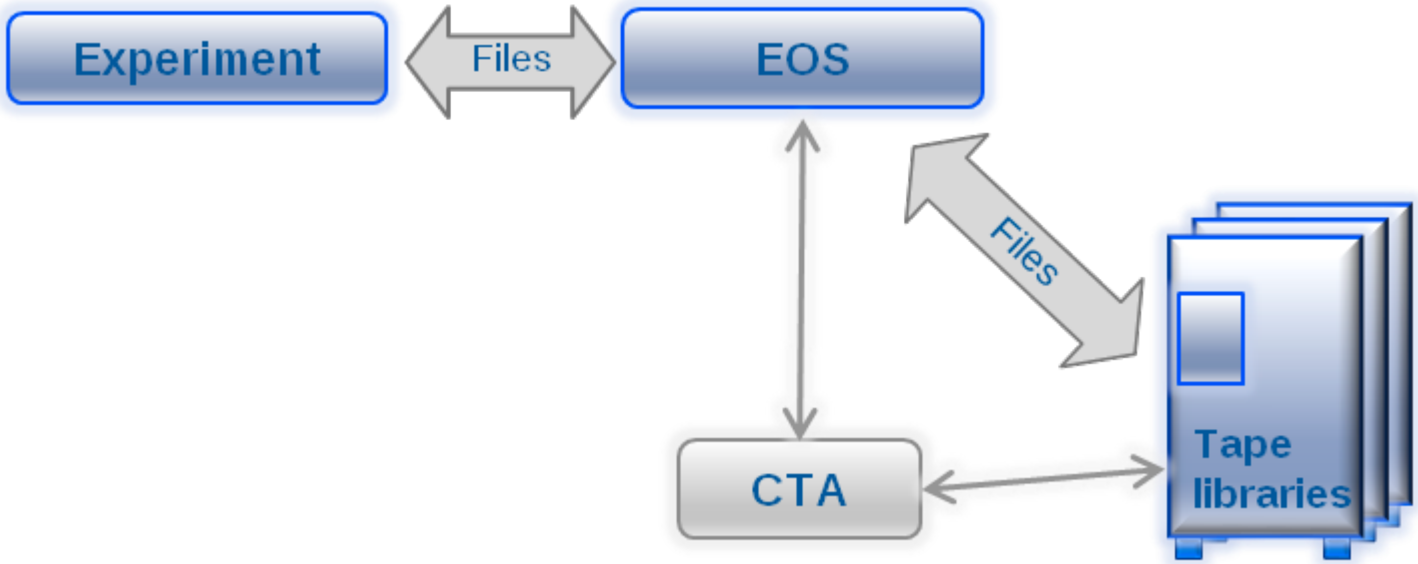
System testing service developments using

Update on EOS + CTA Continuous Integration system

Julien Leduc from **IT STorage** group CERN

CERN Tape Archive?



- CTA is glued to the back of EOS
- EOS manages CTA tape files as replicas
- CTA contains a catalogue of all tape files
- More on CTA tomorrow morning!

CTA + EOS developments

Tightly coupled software ⇒ tightly coupled developments

Extensive and systematic testing is paramount to limit regressions

CTA + EOS integration tests (*What?*)

- Complex situation:
  - **2 distinct software projects**
    - **relying on specific shared developments (xrootd...)**
  - **Several external dependencies** per instance: 1 database, 1 tape library, 1 objectstore

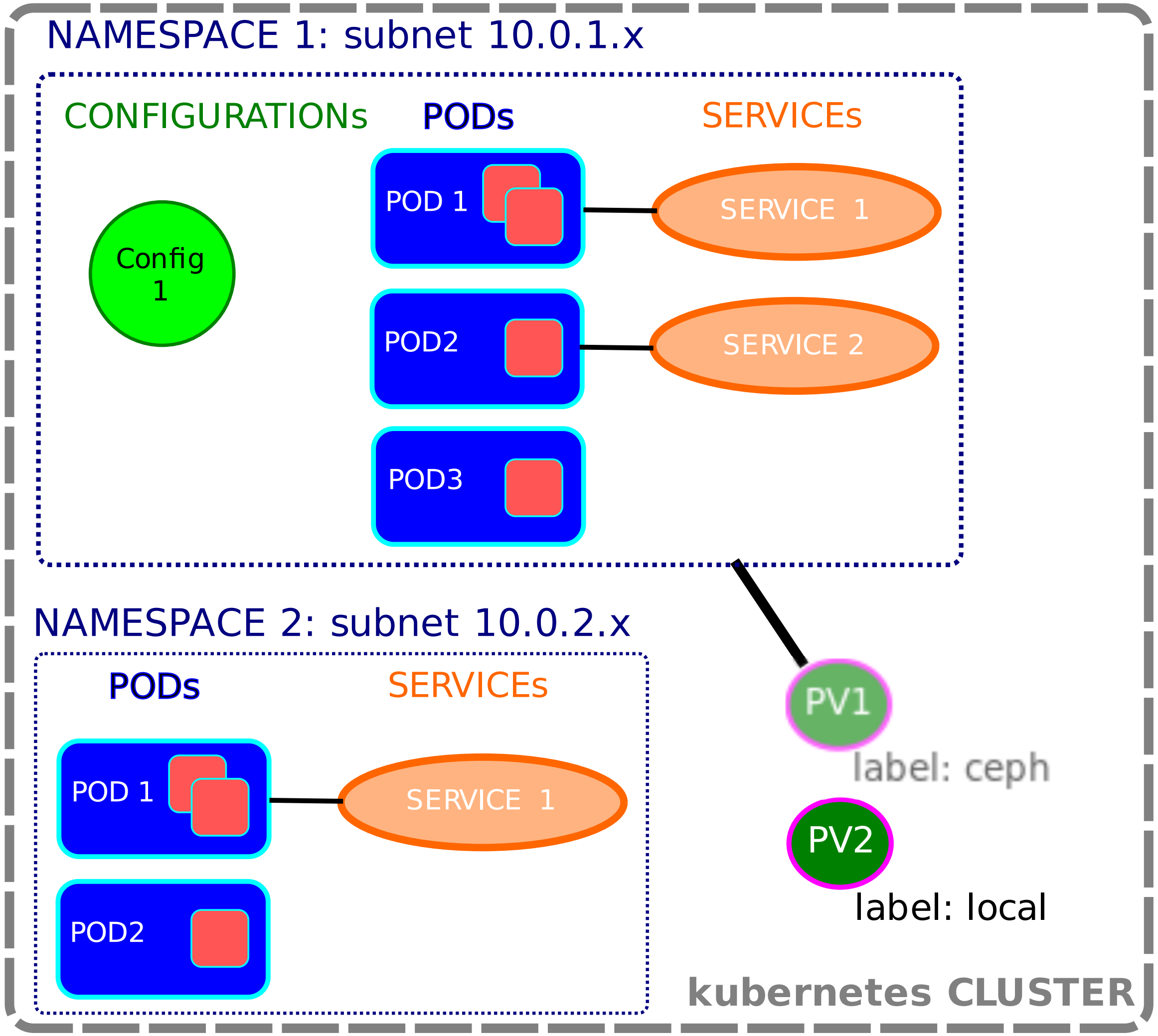
CTA + EOS integration tests (*Constraints?*)

- I hate **repetitive tasks** and I am **impatient**
  - no manual operation → CI
  - make it fast
- Other possible use cases?

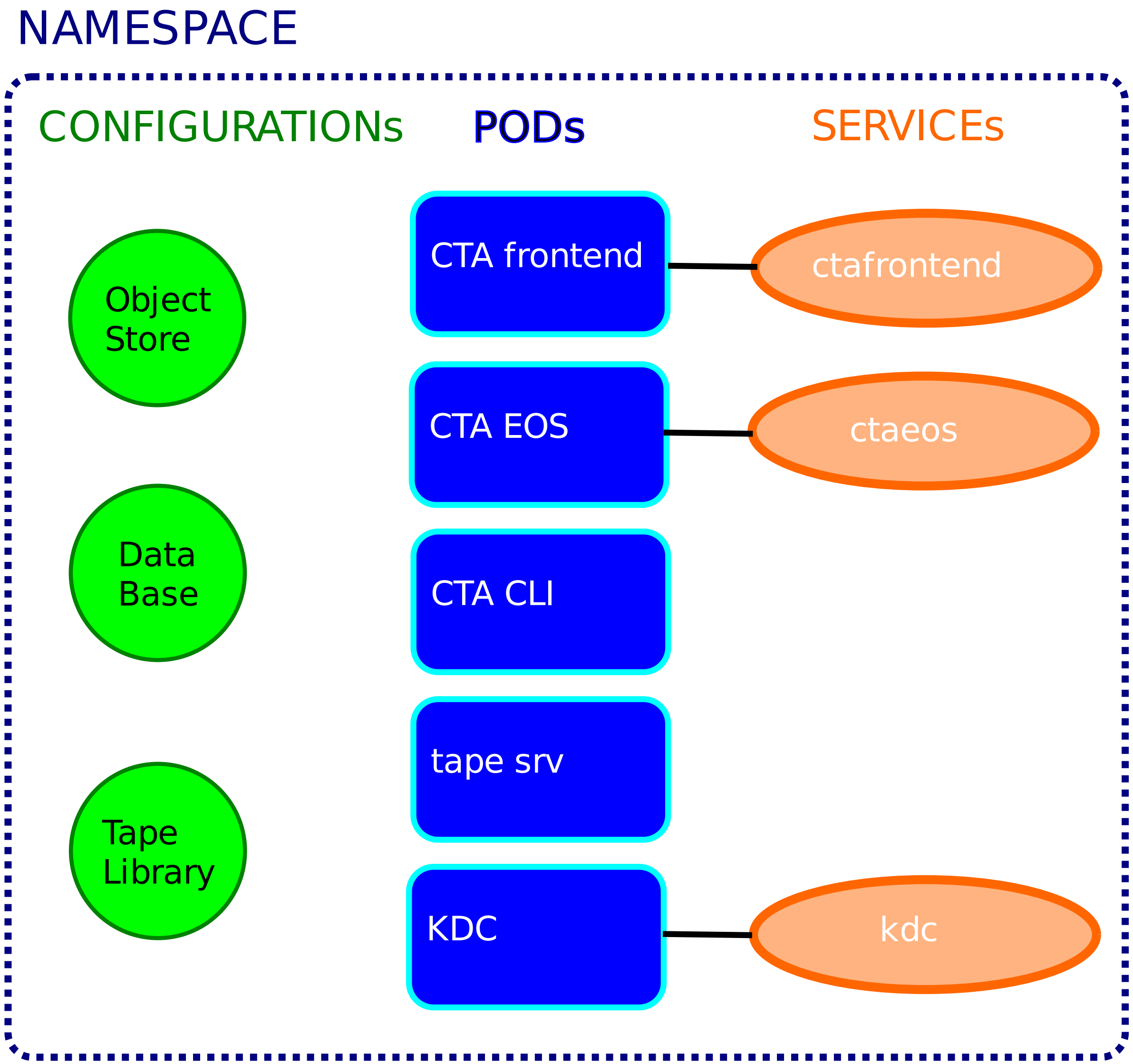
Kubernetes EOS CTA generic instance

- Implement a framework based on a single generic docker image.
- Use Kubernetes to build an EOS CTA instance out of it.
- Flexible enough to accomodate any supported resource (database, objecstore, tape library).
- Part of CTA code repository: CI tests are evolving with the tested code.

Basic Kubernetes concepts



EOS CTA generic instance

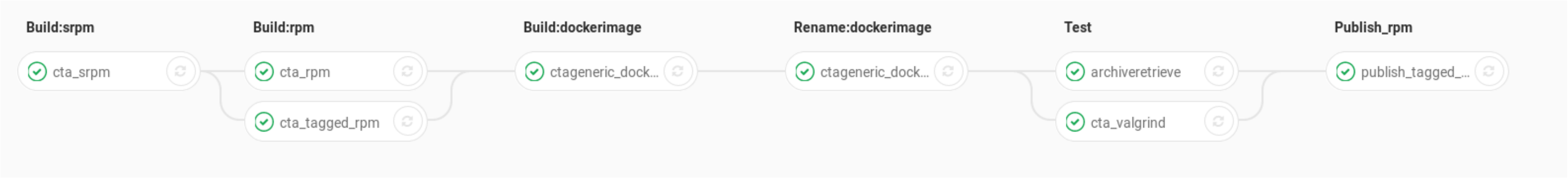


USE CASE 1: CTA CI

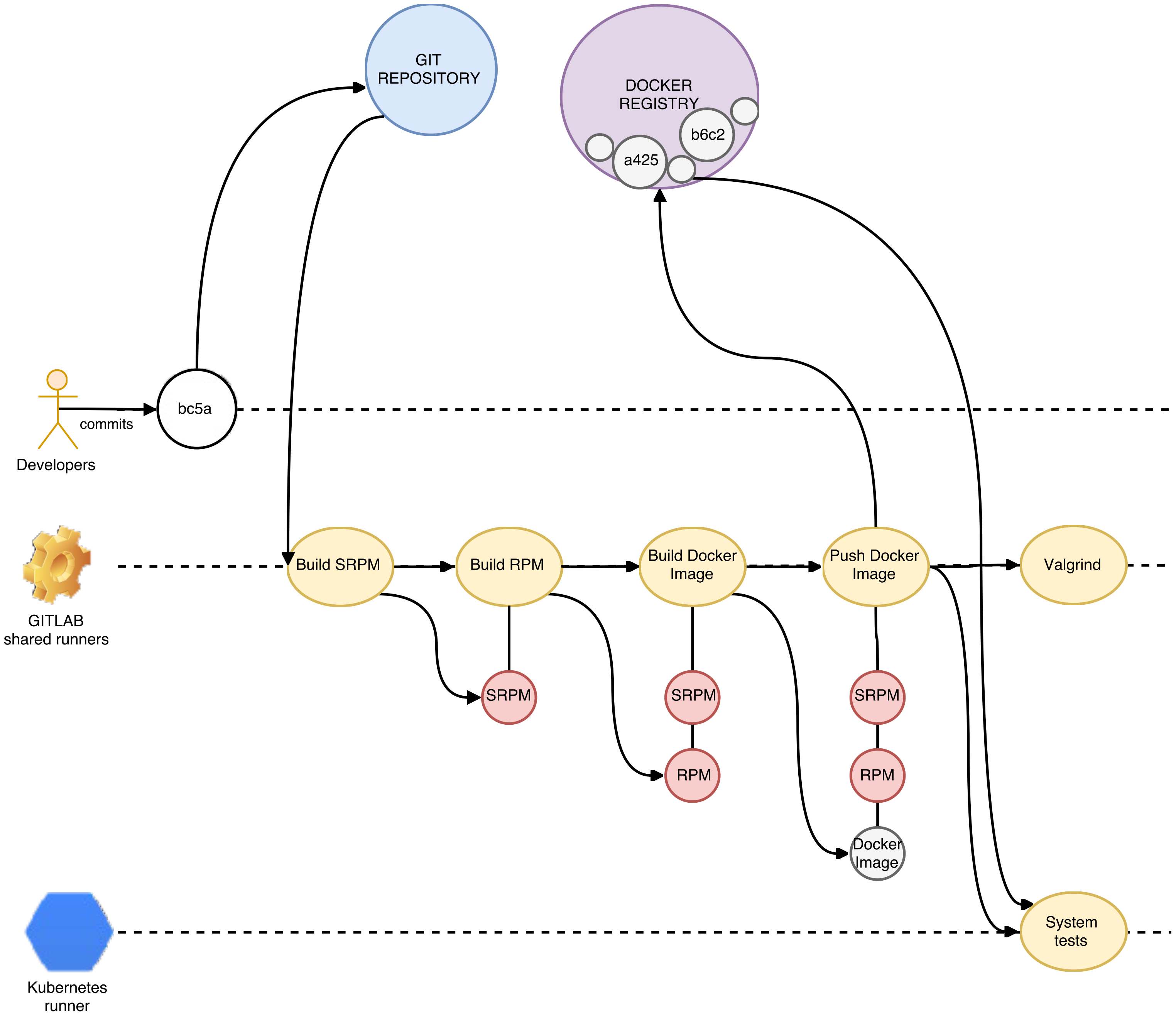
Implemented in CERN Gitlab instance:

- Implements kubernetes framework on a gitlab runner.
- Resources:
    - external Oracle DB instance
    - external Ceph objectstore
    - MHVTL
- When instance ready run a test that `xrdcp` 10k files to EOSCTA, delete the disk copy and retrieve these from tape

CTA CI

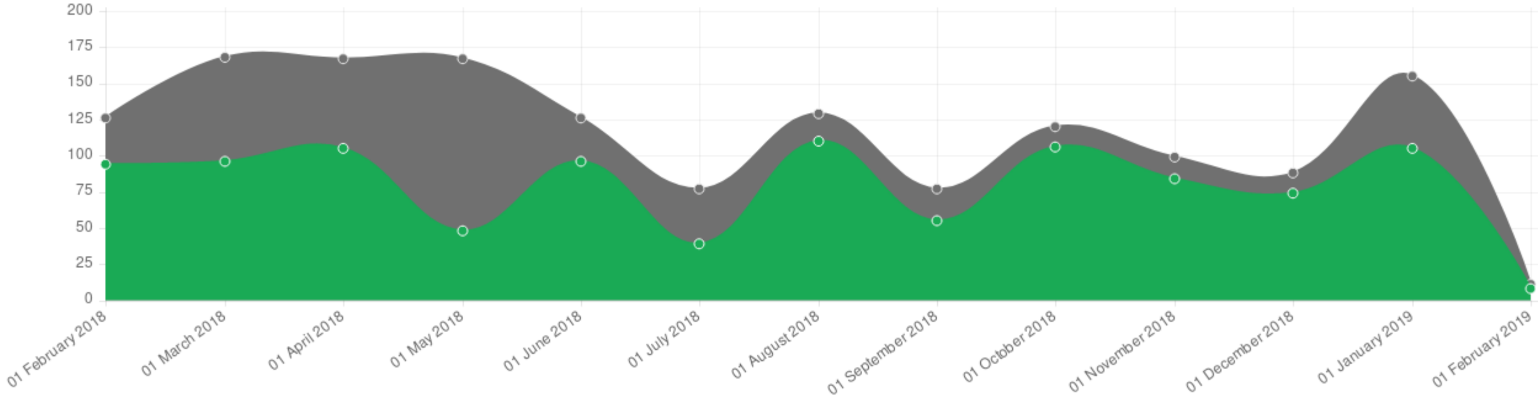


- Build software: CTA RPMs available as **artifacts**
- Build and publish a **generic Docker image** in gitlab registry
  - Contains **all required versioned software (artifacts)** and access to **versioned software cache repository** for dependencies
- Run **system tests** in single VM `kubernetes` cluster (specific gitlab-runner)


GIT REPOSITORY
DOCKER REGISTRY
b6c2
a425
commits
bc5a
Developers
GITLAB shared runners
Build SRPM
Build RPM
Build Docker Image
Push Docker Image
Valgrind
SRPM
SRPM
RPM
SRPM
RPM
Docker Image
System tests
Kubernetes runner

Some statistics

3000+ pipelines ran since CI is in place



Bonus: Nightly EOS regression tests

Every night a Gitlab schedule runs these steps:

- run standard archival test
- upgrade EOS to latest dev tagged release
- run standard archival test against the new EOS version

This allows CTA developers to catch EOS regressions that impact CTA specific workflows.

USE CASE 2: CTA developers

- Entirely runs on developer laptop:
  - Implements kubernetes framework in a Virtualbox CentOS VM
- Offline resources: local sqlite DB, local file based objectstore, MHVTL

When instance ready run specific developer test.

Strengths

- Quickly deploys a disposable local EOS CTA instance.
- Much shorter learn curve for new comers that can focus on their work.
  - Best deployment practices included.
- Successfully used for:
  - Objectstore developments
  - Database catalogue backend developments (`mysql`, `postgres`)
- Developers improve CI code for me.

USE CASE 3: CTA PPS stress tests

Initially Implemented in a dedicated Puppet managed PPS instance to reach 2GB/s:

- 1 MGM
- 3 FSTs (750TB of storage)
- 1 CTA frontend
- 8 tape servers and associated tape drives for BW stress tests
- 3 VTL tape server for rate stress tests

PPS instance stress tests

Many issues with VM/Puppet approach:

- Code changes in EOS CTA often requires error prone **manual Puppet manifest changes or manual reconfiguration**.
- Extensive use of `rundeck` to deploy a CTA release still requires several hours.

PPS instance stress tests

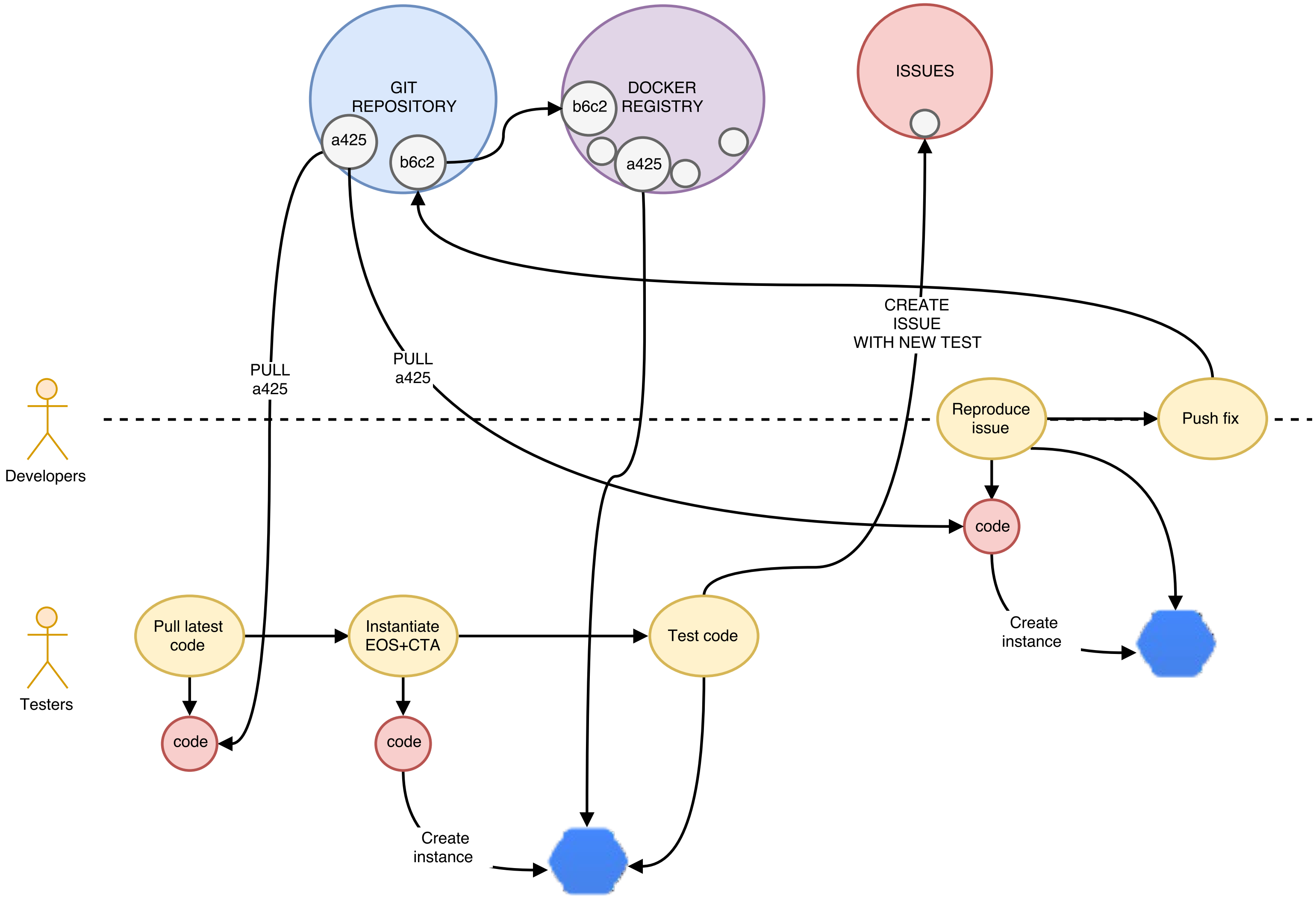
Low turnover leads to:

- Less testing...
- More code changes between 2 tests: more deployment errors, more regressions.
- Time consumming PPS babysitting...
- Log collection/monitoring of PPS? O(kHz) events/machine?
- Reproducibility?
- Deployment best practices???? Best case: obscure devops documentation...

Here comes the *Beefy system*

- Implements kubernetes framework on one hyperconverged server with 16 SSDs:
  - Plenty of IOPS for VTL rate tests
  - Plenty of bandwidth to model a sizable CTA instance (10 tape servers, 6 FSTs...)
- Resources: Oracle DB instance, Ceph objectstore, MHVTL

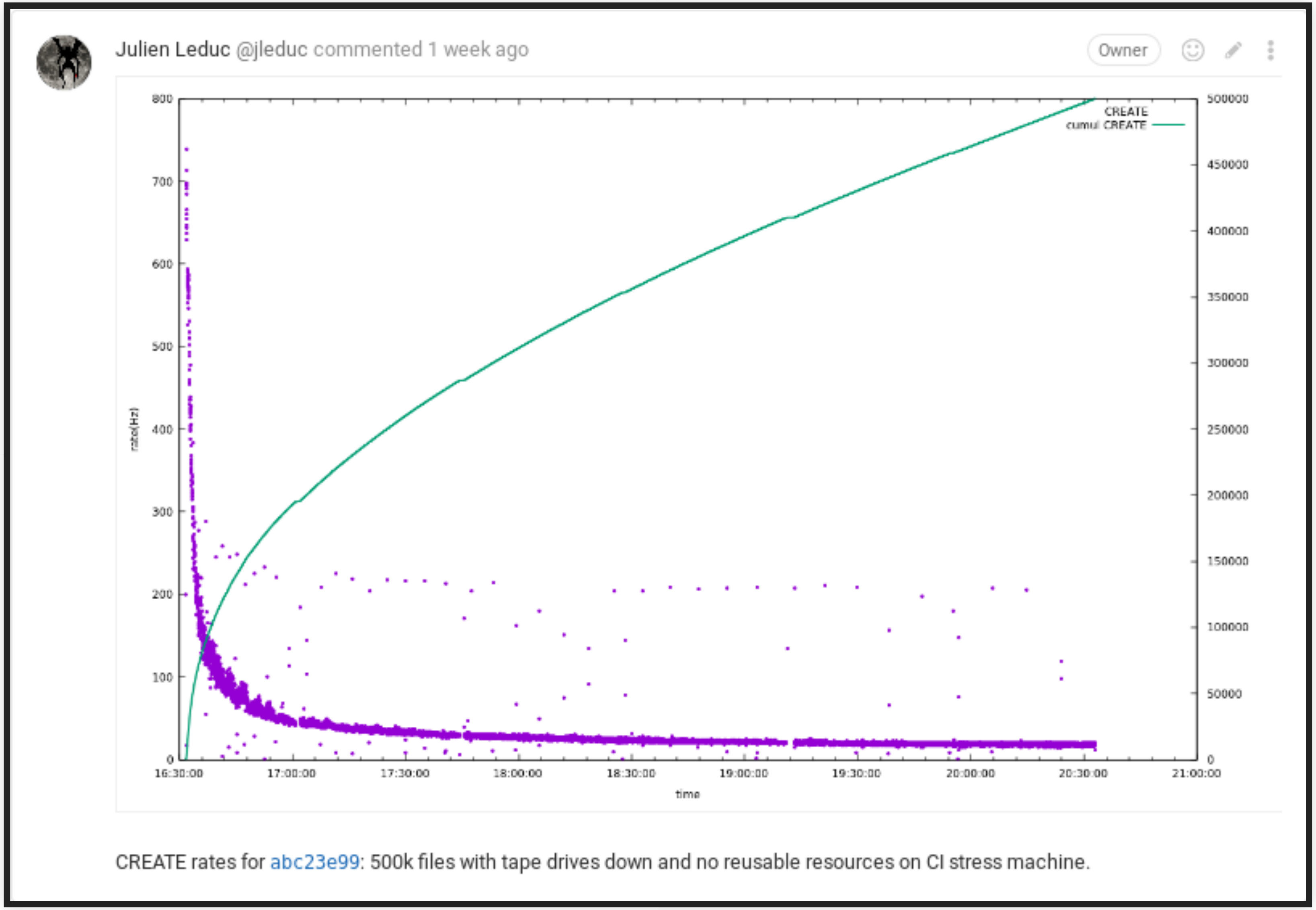
When instance ready run a beefy CI test that `xrdcp` **1M files to EOS CTA**, delete the disk copy and retrieve these from tape.


GIT
REPOSITORY
a425
b6c2
DOCKER
REGISTRY
b6c2
a425
ISSUES
PULL
a425
PULL
a425
CREATE
ISSUE
WITH NEW TEST
Developers
Reproduce
issue
Push fix
code
Create
instance
Testers
Pull latest
code
Instantiate
EOS+CTA
Test code
code
code
Create
instance

Beefy system stress tests

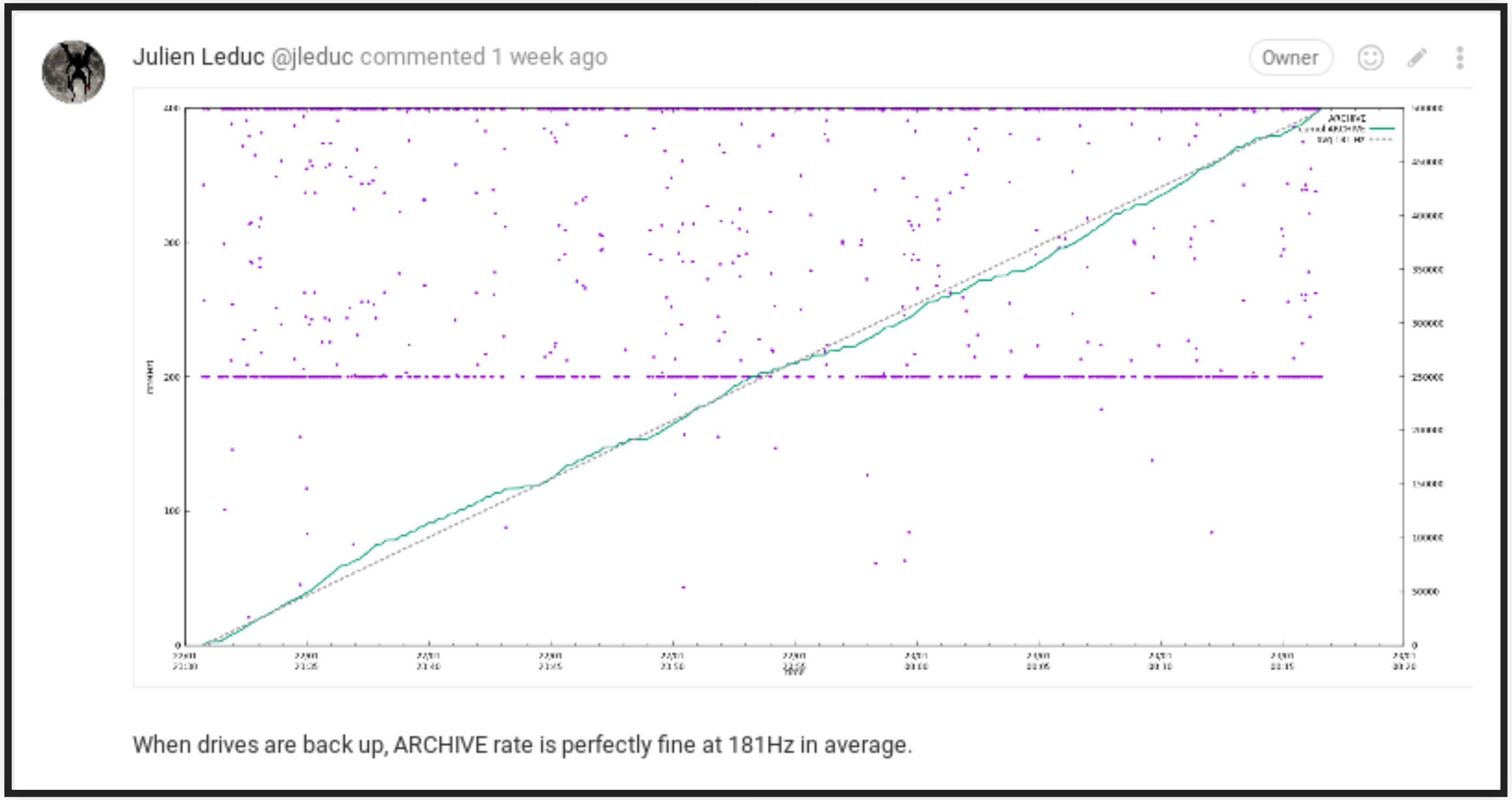
- Fast turnover that allows to quicky reproduce a bug again and again in various conditions:
  - Fully automated **will go in CD step**.
  - Fully reproducible.
- Allowed me to successfully track down an exponential performance degradation regression
  - Identified, fixed and tested in 3 days **was here for 2 months**.
- Allowed me to identify a bug in the frontend that

Real life Issue tracking/fixing



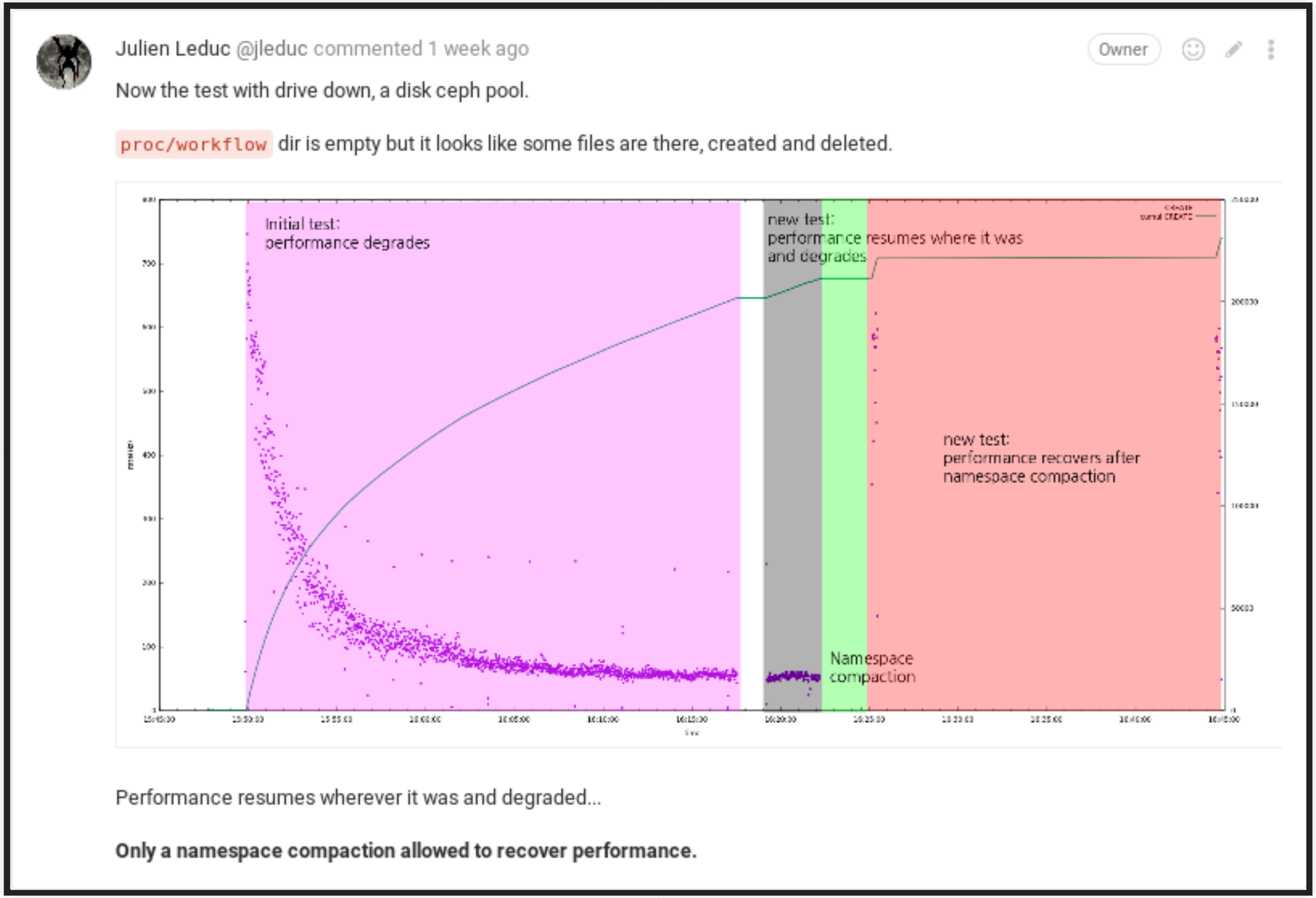
Julien Leduc @jleduc commented 1 week ago

CREATE rates for abc23e99: 500k files with tape drives down and no reusable resources on CI stress machine.



Julien Leduc @jleduc commented 1 week ago

When drives are back up, ARCHIVE rate is perfectly fine at 181Hz in average.



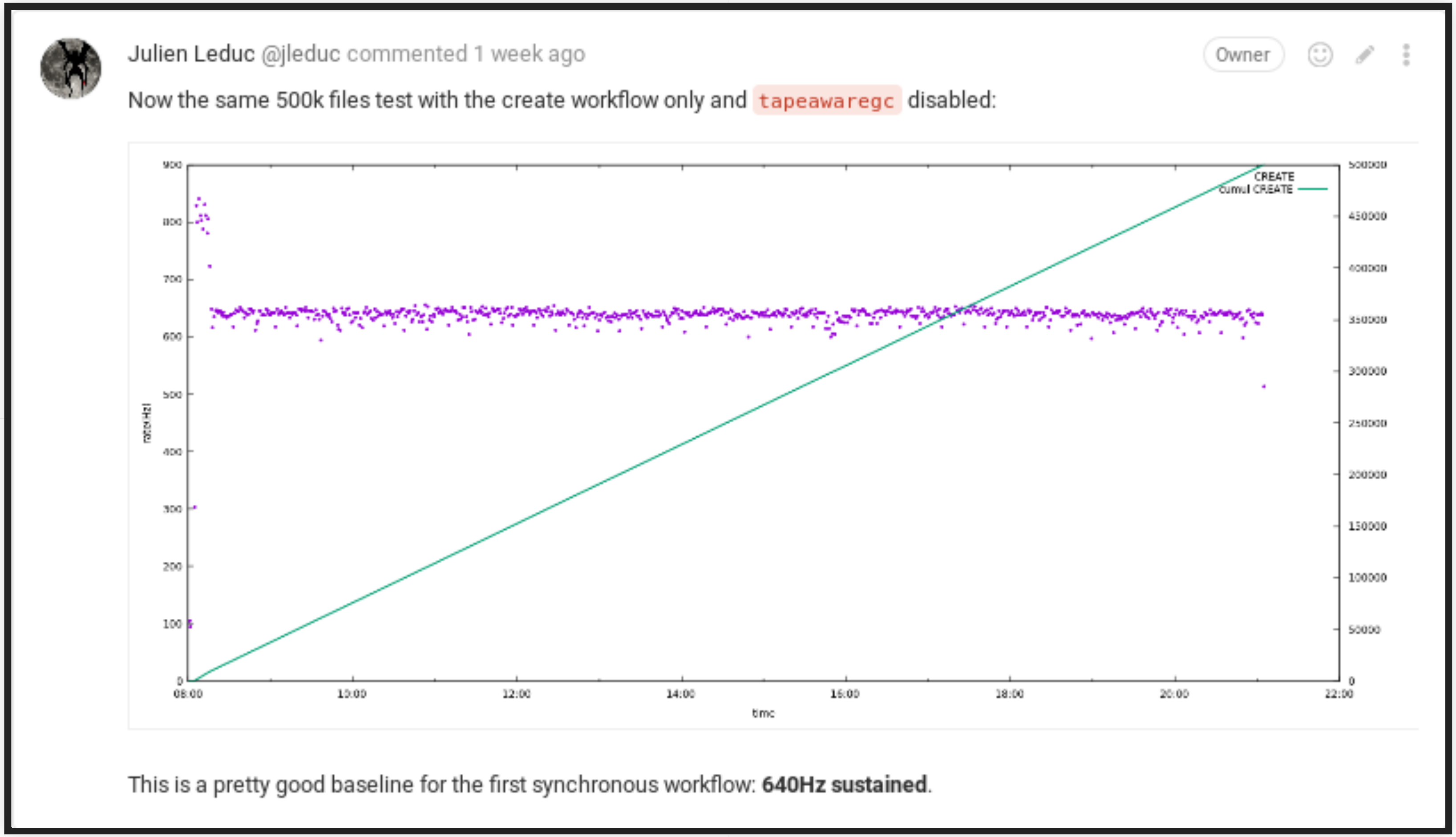
Julien Leduc @jleduc commented 1 week ago

Now the test with drive down, a disk ceph pool.

`proc/workflow` dir is empty but it looks like some files are there, created and deleted.

Performance resumes wherever it was and degraded...

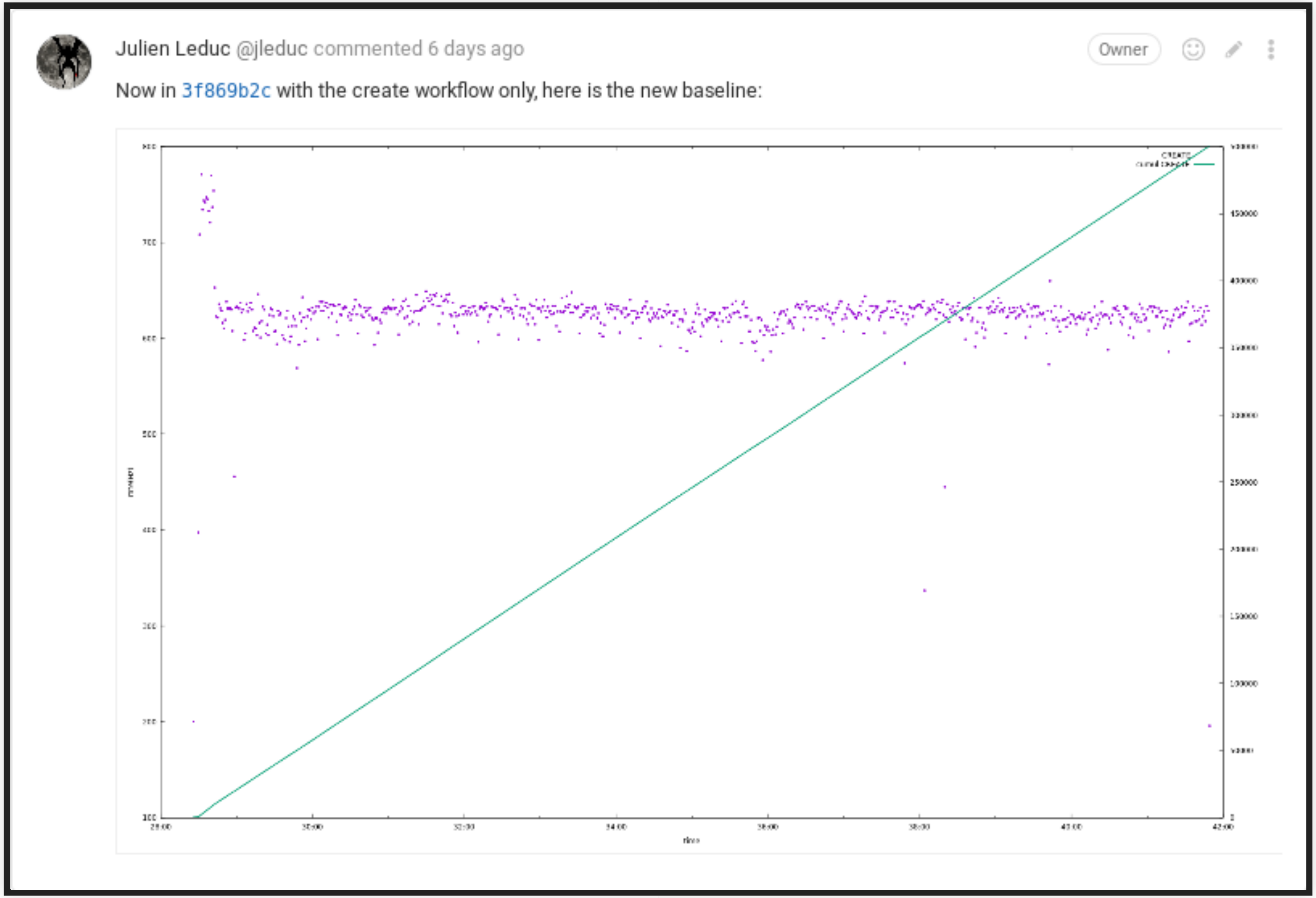
Only a namespace compaction allowed to recover performance.



Julien Leduc @jleduc commented 1 week ago

Now the same 500k files test with the create workflow only and `tapeawaregc` disabled:

This is a pretty good baseline for the first synchronous workflow: **640Hz sustained**.



Julien Leduc @jleduc commented 6 days ago

Now in `3f869b2c` with the create workflow only, here is the new baseline:

THE END?

- Very powerful approach addresses and federates all our development/testing use cases
- Fast, flexible, isolated and self contained in software repository
- Reproducible development environment that allows regression and performance tests

TO DO

- Automatic log analysis
- Bandwidth performance tests
- Evaluate possible production use ☺