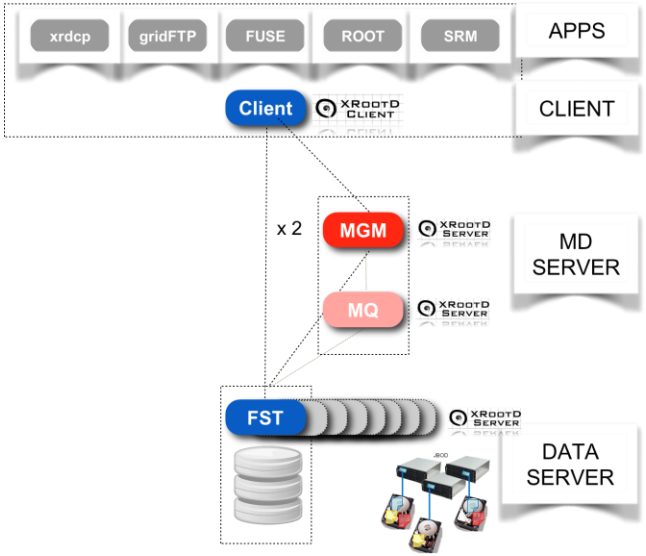
CERN

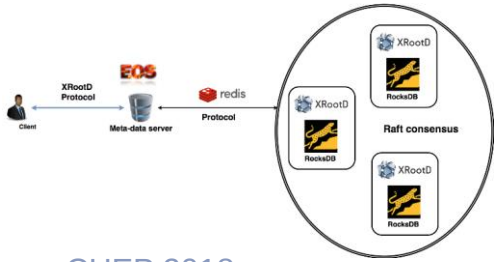
Testing of complex, large-scale distributed storage systems: a CERN disk storage case study

J. Makai, A. J. Peters, G. Bitzes, E. A. Sindrilaru, M. K. Simon, A. Manzi (speaker)
CERN

EOS

- **Disk** file storage
- Designed for **Massive storage** with high performances (> 50PB, 500 M files and Khz metadata operations)
- **In memory namespace**
- **New distributed namespace backend (QuarkDB)**
- **https://indico.cern.ch/event/587955/contributions/2936873/**





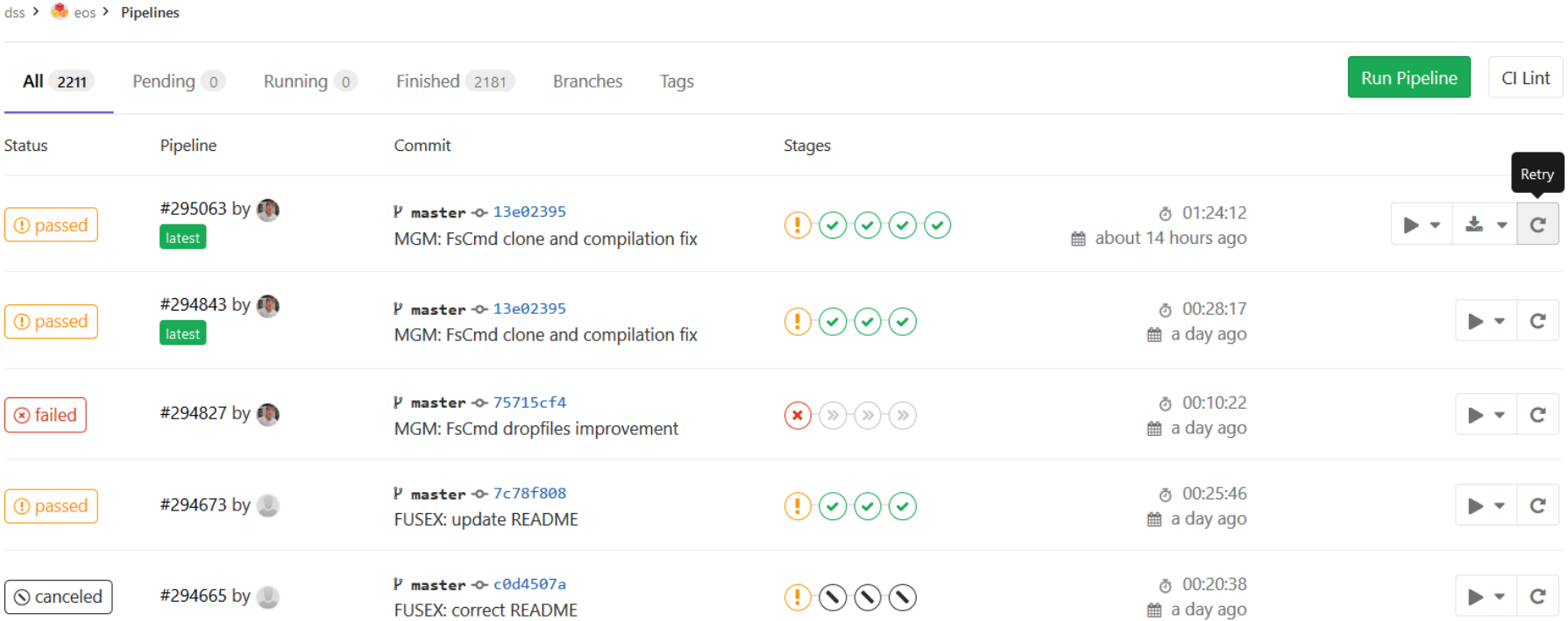
EOS CI till beginning of 2017

- Jenkins CI
  - Not state of the art anymore
  - Lacks some functionalities
  - Harder to maintain (lot of plugins from different sources)
- No real CI present for EOS
  - Only building RPMs for couple of platforms (CC7, SLC6 maintained mostly)
  - Using mock for build
  - No regression testing, regular static analysis in place
  - No integration with other build facilities (e.g. Koji)
  - Lot of manual steps, error-prone

Long journey behind us

- We started again almost from scratch
- Migration to Gitlab CI for EOS, XrootD and all related projects
- Completely new technologies
  - Docker containers instead of mock
  - Docker images/containers for system testing and distribution
  - Compiler caching
  - Static analysis
  - Google test for unit testing
- New build cluster
- Support for brand new platforms
- Agile mindset, easy and quick to respond to new issues, needs

Migration to Gitlab CI



Docker containers for builds

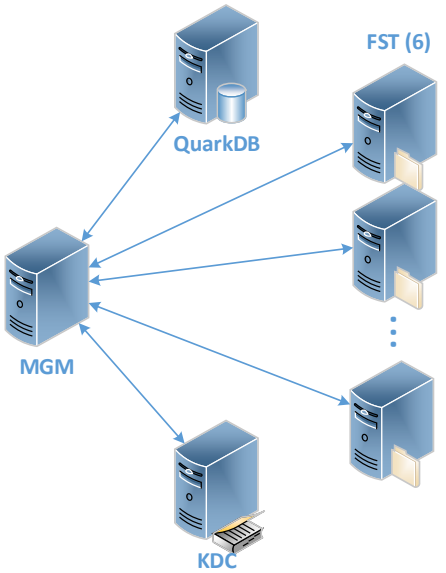
- Isolated build environment
- Faster to support new platforms
  - mock moves slowly
  - Images are immediately present
- Support other distributions with different packaging
- Still one homogeneous build environment to manage

New platforms supported

- Mac OS
  - Not completely new but revived from its ashes
  - Updated build to Sierra
- Fedora
  - We can update quickly to latest versions thanks to the new ecosystem
  - Fedora 27, 28 and rawhide currently in CI
- Debian
  - Ubuntu artful & bionic in CI
- Docker image for each release available via GitLab Registry
  - gitlab-registry.cern.ch/dss/eos/eos-prod:4.3.2

Docker based testing

- Docker image created for each build with fresh packages of the changes
- Simulates a fully functional system with all its features
- https://gitlab.cern.ch/eos/eos-docker project
  - Independent from the CI environment
  - Used also for development and easy functionality showcase



Docker based testing – System test

- Both with old, in-memory and new, quarkdb namespace
- New fully functional instance of EOS is created: tests booting, authentication, etc.
- Runs eos functional tests: testing EOS commands with eos and xrootd clients
- FUSE(X) testing: fuse microtests for both, benchmarks for FUSEX and POSIX compliance tests for FUSEX daily
  - https://indico.cern.ch/event/587955/contributions/3012718/

Docker based testing – Stress test

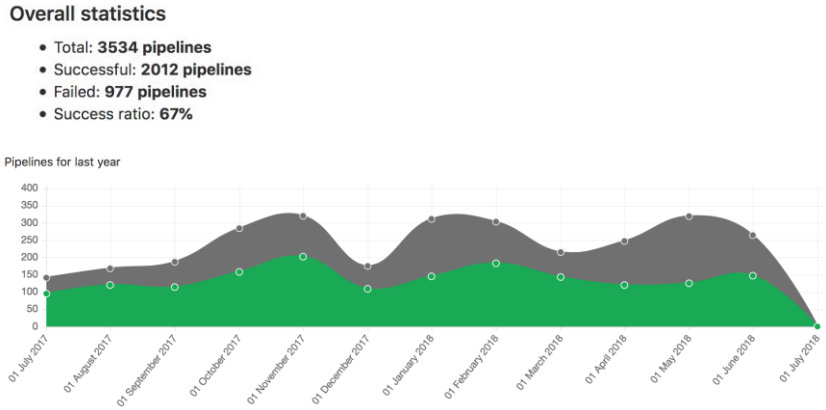
- To put some load on the system
- Write, stat, read, delete files from multiple threads simultaneously using xrootd protocol

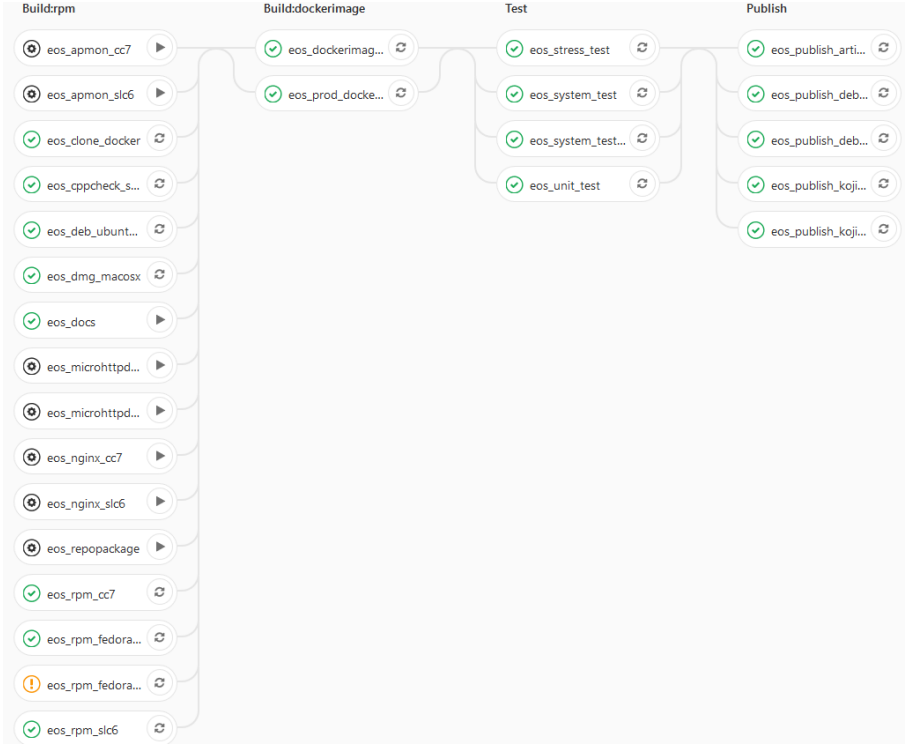
```
10000 files :: 100 threads :: Run 2/3

    PASS xroot :: write (13.60 sec)
         Rate: 735.42 Hz
         Latency span: from 0.31 to 8.91 sec

    PASS xroot :: stat (0.61 sec)
         Rate: 16489.76 Hz
         Latency span: from 0.00 to 0.56 sec

    PASS xroot :: read (8.54 sec)
         Rate: 1171.22 Hz
         Latency span: from 0.21 to 7.06 sec

    PASS xroot :: delete (3.82 sec)
         Rate: 2619.79 Hz
         Latency span: from 0.00 to 3.58 sec
```

```
10000 files :: 100 threads :: Run 3/3

    PASS xroot :: write (16.61 sec)
         Rate: 602.00 Hz
         Latency span: from 0.12 to 8.39 sec

    PASS xroot :: stat (1.46 sec)
         Rate: 6864.65 Hz
         Latency span: from 0.01 to 1.44 sec

    PASS xroot :: read (16.73 sec)
         Rate: 597.70 Hz
         Latency span: from 0.53 to 5.21 sec

    PASS xroot :: delete (5.06 sec)
         Rate: 1976.40 Hz
         Latency span: from 0.00 to 4.62 sec
```

Gitlab CI Pipeline – 30 jobs, 4 stages

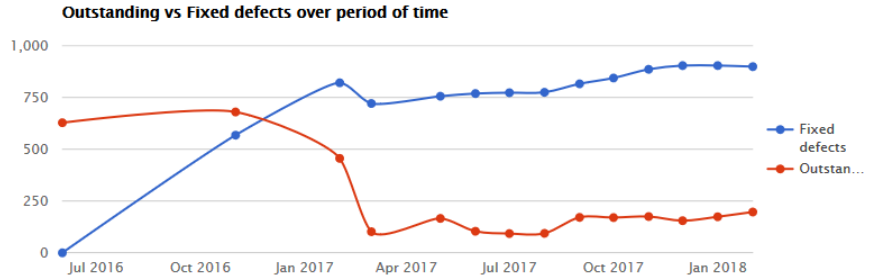
- No manual,
  no error-prone steps
- Integrated with everything possibly
  - SRPM → CERN Koji

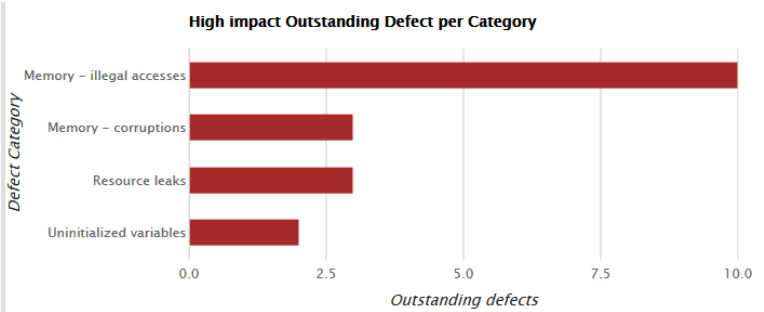




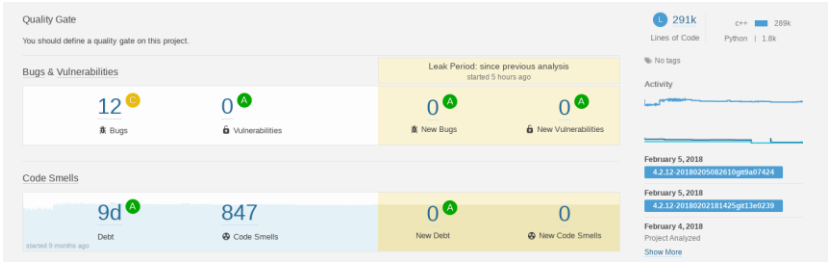
Static analysis – Coverity & Sonar

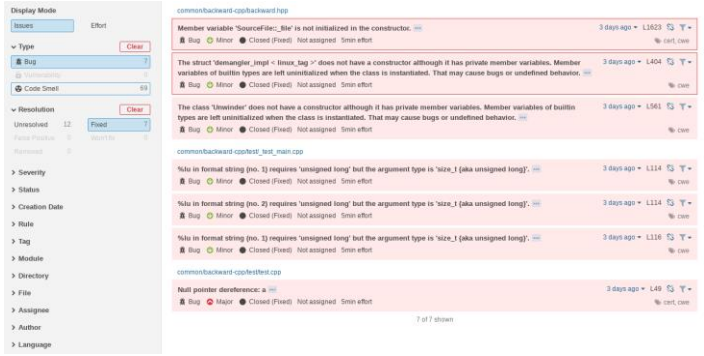
- Runs once a day
- Special instrumented compilation required

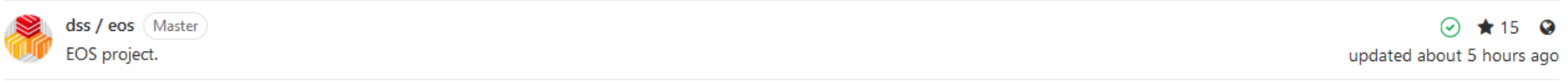




- For each push to master
- Cppcheck: analysis on source code level, no compilation





Conclusions & Future steps

- Successfully implemented Continuous Testing for EOS since more than a year
- Next Steps :
  - Implement CD via Kubernetes integration in Gitlab
  - Add EOS HTTP/GridFTP gateways testing
  - Add more regression tests for functionality
  - Implement Robustness testing