EOS Open Storage

eosxd evolution - eoscfsd passthrough

Dr. **Andreas-Joachim Peters**

for the EOS Project - CERN IT - Storage Group

IT Auditoirium - CERN

15.03.2024

Contents

- **eos**xd
  - Evolution
  - Outlook
- **eos**cfsd
  - Design
  - Performance & Outlook

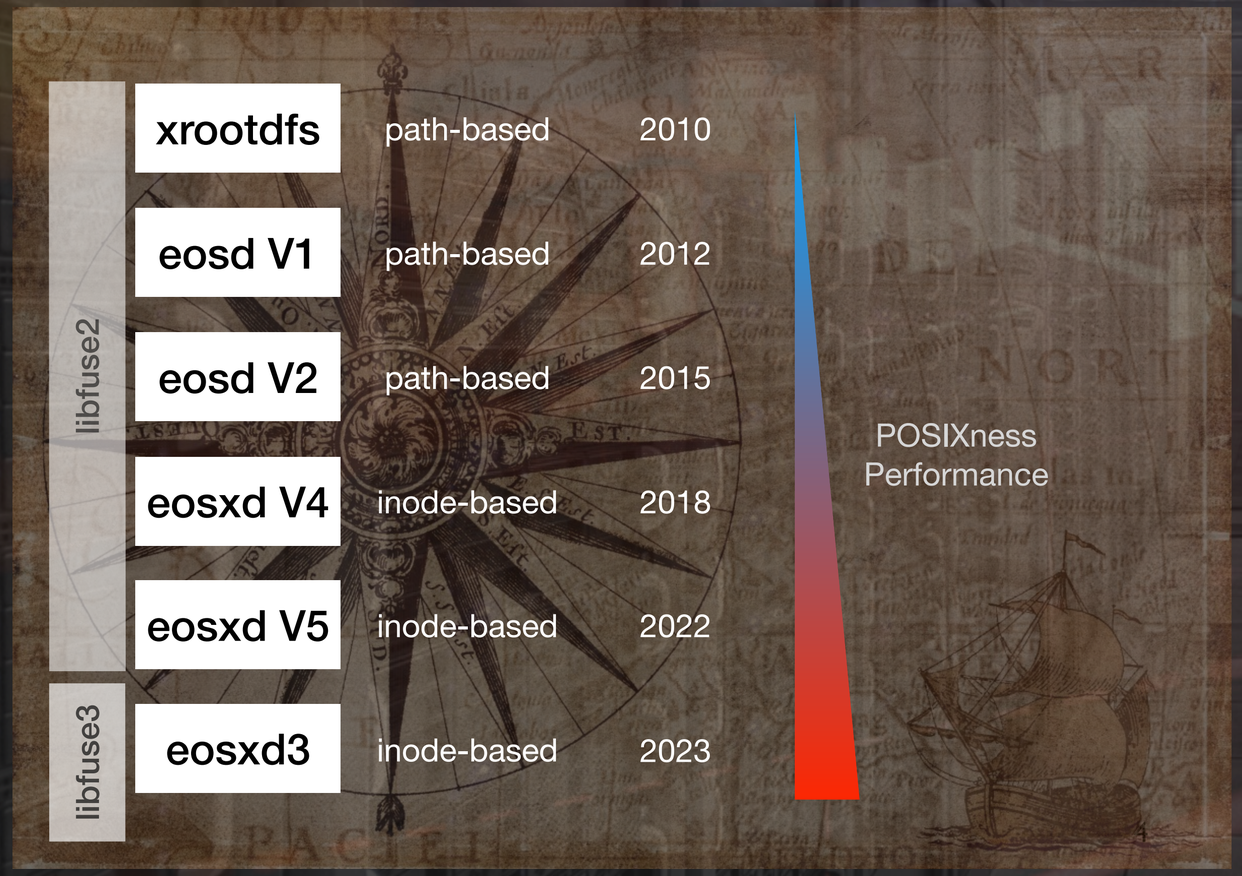

EOS
CERN
EOS XD
BEST OF FUSE

EOS Fuse Filesystem

eosxd *genealogy reminder*

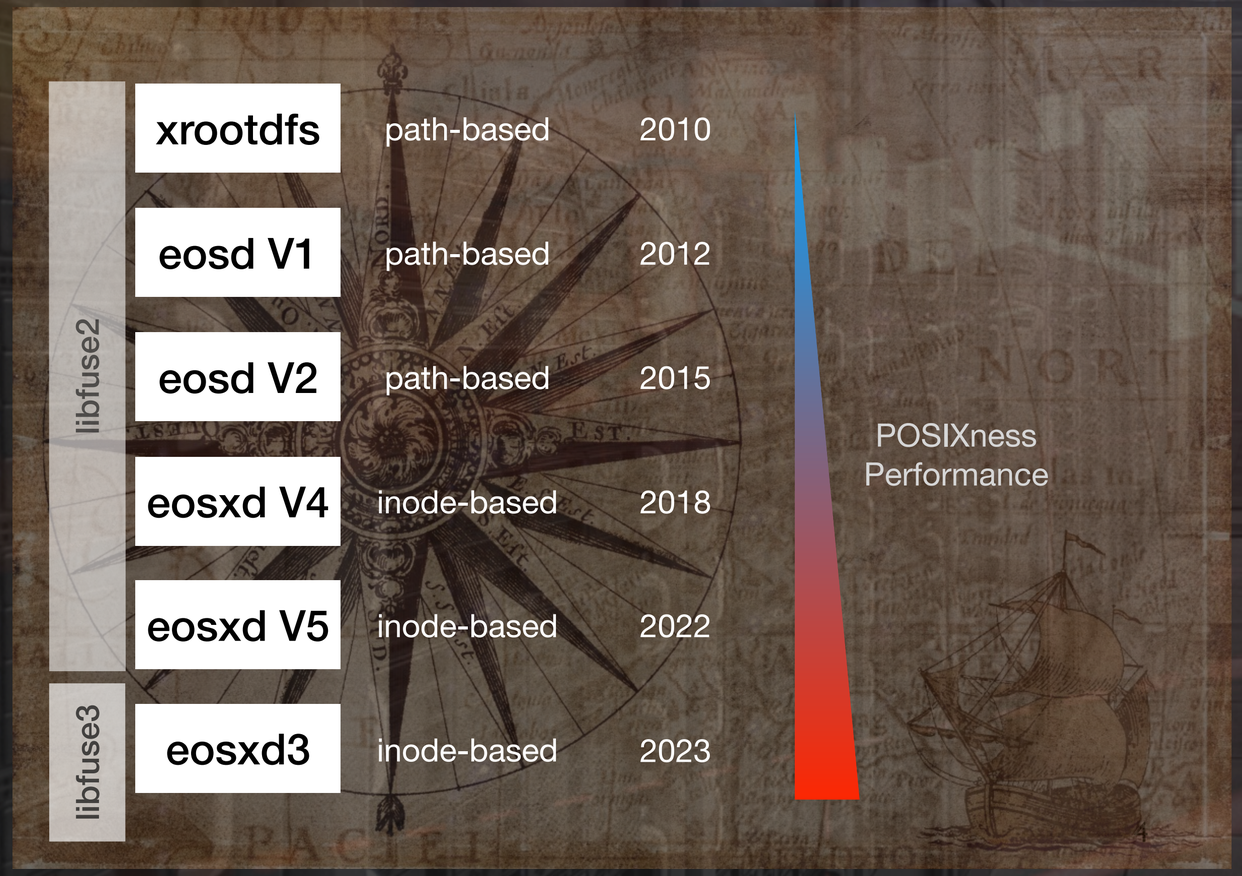
EOS Fuse Filesystem

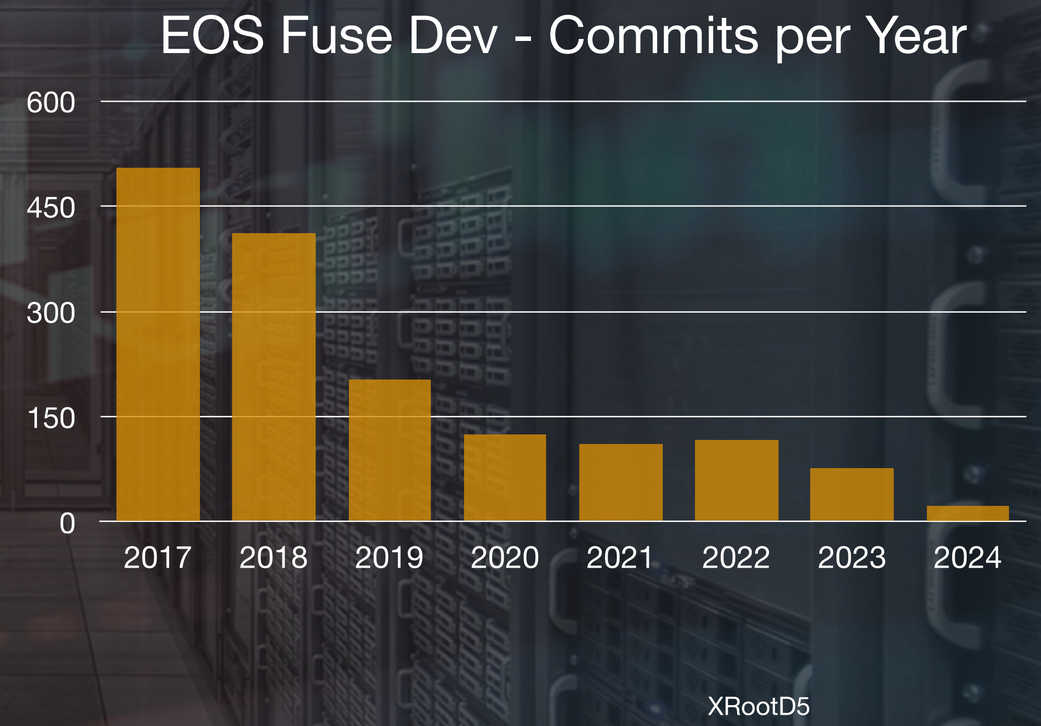
eosxd *genealogy reminder*



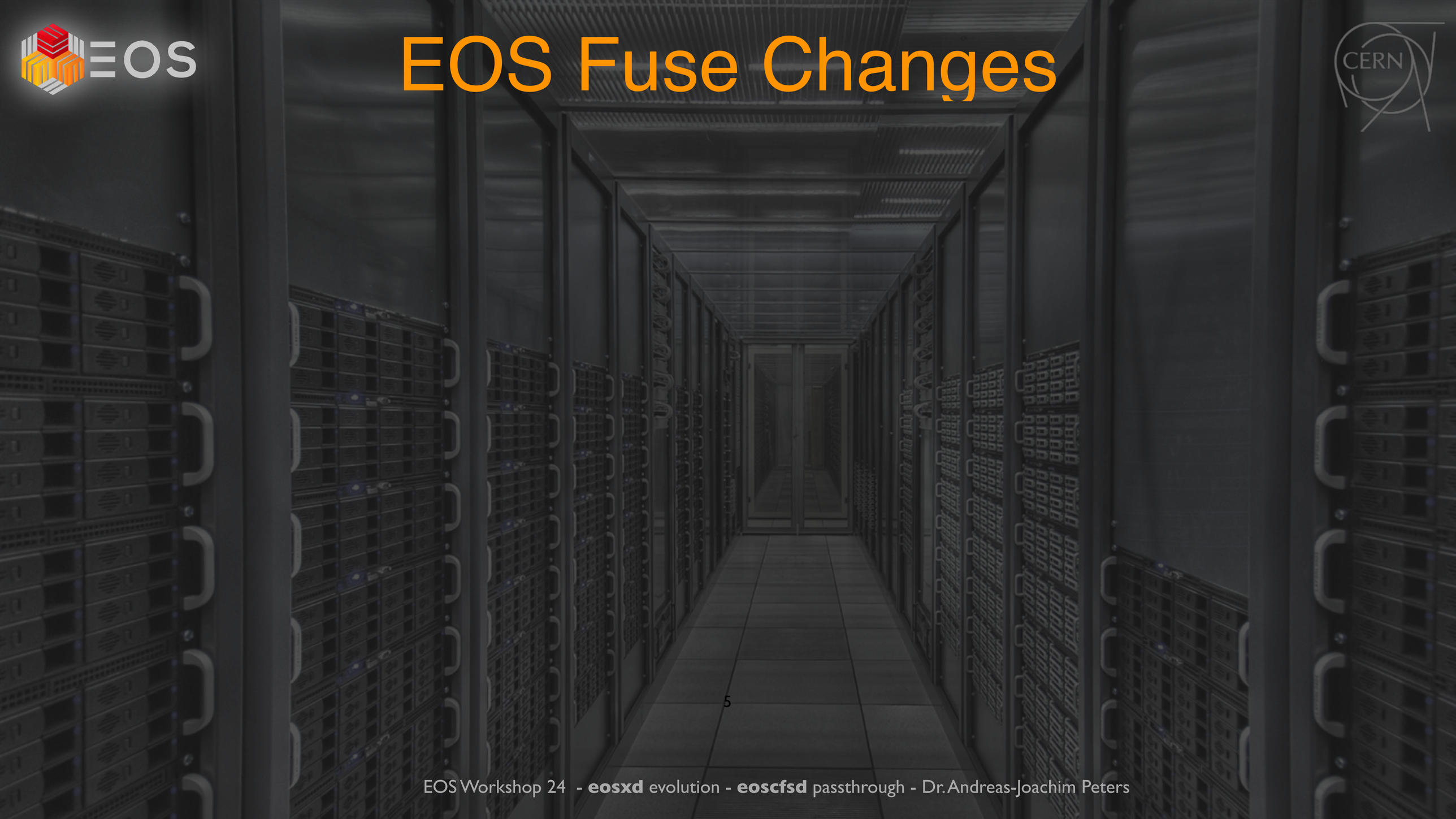
EOS Fuse Filesystem

eosxd *genealogy reminder*





EOS Fuse Changes



EOS Fuse Changes

- FUSE3: write-back cache cannot be used because callbacks are suppressed for known inodes - see here

EOS Fuse Changes

- FUSE3: write-back cache cannot be used because callbacks are suppressed for known inodes - see here

- Several minor/medium bug fixes
  (LRU lists, strings, concurrency, flock, overlapping reads)

EOS Fuse Changes

- FUSE3: write-back cache cannot be used because callbacks are suppressed for known inodes - see here
- Several minor/medium bug fixes
  (LRU lists, strings, concurrency, flock, overlapping reads)
- **eos**xd protocol V5 - differential protocol instead of complete directory invalidations (broadcast REFRESH) - reduces server->client messaging

EOS Fuse Changes

- FUSE3: write-back cache cannot be used because callbacks are suppressed for known inodes - see here
- Several minor/medium bug fixes
  (LRU lists, strings, concurrency, flock, overlapping reads)
- **eos**xd protocol V5 - differential protocol instead of complete directory invalidations (broadcast REFRESH) - reduces server->client messaging
- ZTN token support for EOS mounts (SciToken support)

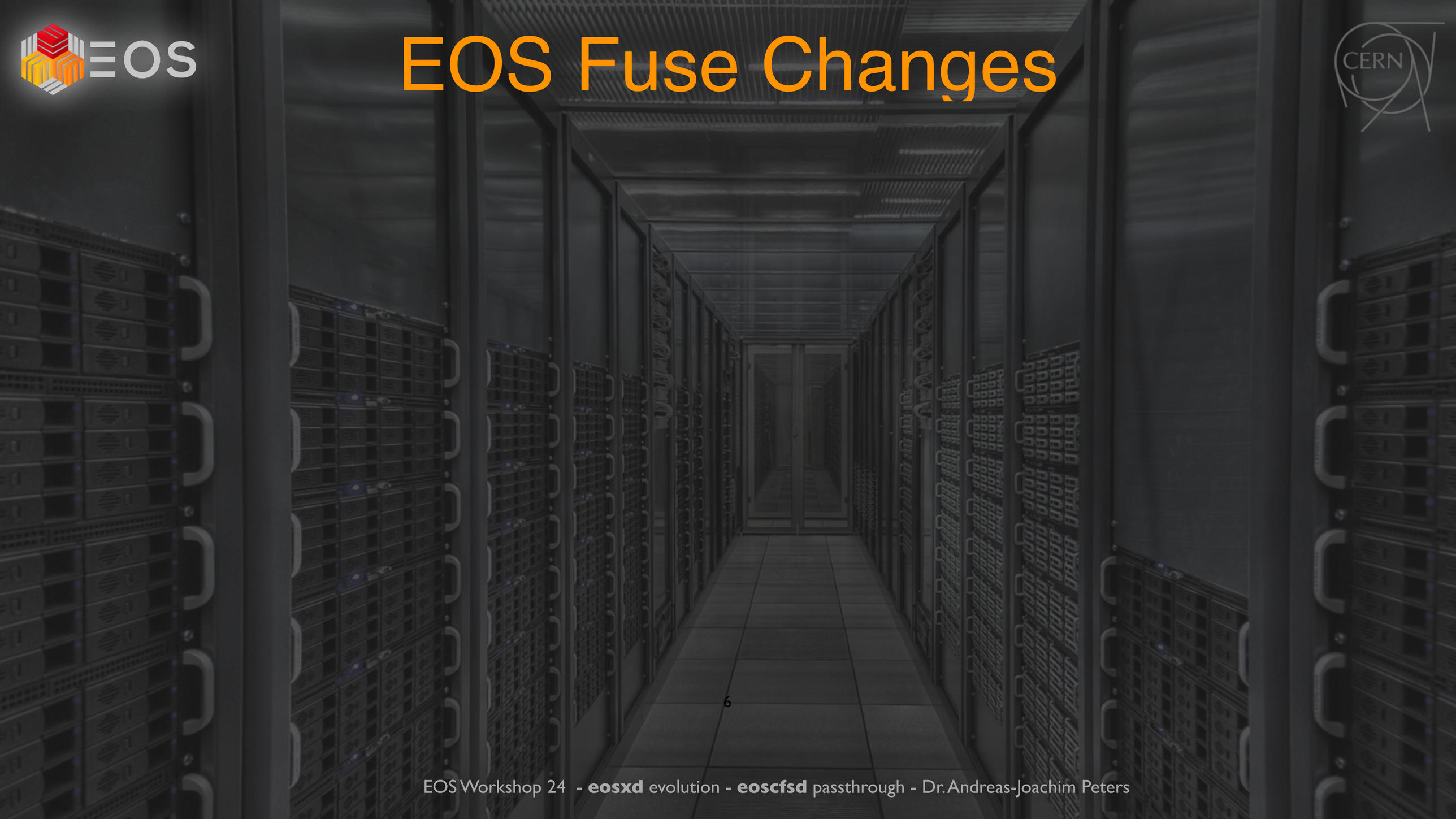
EOS Fuse Changes

- FUSE3: write-back cache cannot be used because callbacks are suppressed for known inodes - see here
- Several minor/medium bug fixes
  (LRU lists, strings, concurrency, flock, overlapping reads)
- **eos**xd protocol V5 - differential protocol instead of complete directory invalidations (broadcast REFRESH) - reduces server->client messaging
- ZTN token support for EOS mounts (SciToken support)
- Better blocked client monitoring (origin of lock-up is specified more precise)

EOS Fuse Changes

- FUSE3: write-back cache cannot be used because callbacks are suppressed for known inodes - see here
- Several minor/medium bug fixes
  (LRU lists, strings, concurrency, flock, overlapping reads)
- **eos**xd protocol V5 - differential protocol instead of complete directory invalidations (broadcast REFRESH) - reduces server->client messaging
- ZTN token support for EOS mounts (SciToken support)
- Better blocked client monitoring (origin of lock-up is specified more precise)
- avoiding double-mounting on ALMA 9 when using autofs

EOS Fuse Changes



EOS Fuse Changes

- credentials are bound in open call and kept for all IO operations after open

EOS Fuse Changes

- credentials are bound in open call and kept for all IO operations after open
- global-flush is now disabled by default

  global-flush is preventing XrdCl::File::Open of a file while the contents is flushed from a FUSE client - doubles client-server calls

EOS Fuse Changes

- credentials are bound in open call and kept for all IO operations after open
- global-flush is now disabled by default

  global-flush is preventing XrdCl::File::Open of a file while the contents is flushed from a FUSE client - doubles client-server calls
- bypass deletion through recycle-bin if a file is unlinked while it is still open

EOS Fuse Changes

- credentials are bound in open call and kept for all IO operations after open
- global-flush is now disabled by default

  global-flush is preventing XrdCl::File::Open of a file while the contents is flushed from a FUSE client - doubles client-server calls
- bypass deletion through recycle-bin if a file is unlinked while it is still open
- fix credential hash function (interference) to distinguish credentials with identical path in different container environments

EOS Fuse Changes

- credentials are bound in open call and kept for all IO operations after open
- global-flush is now disabled by default

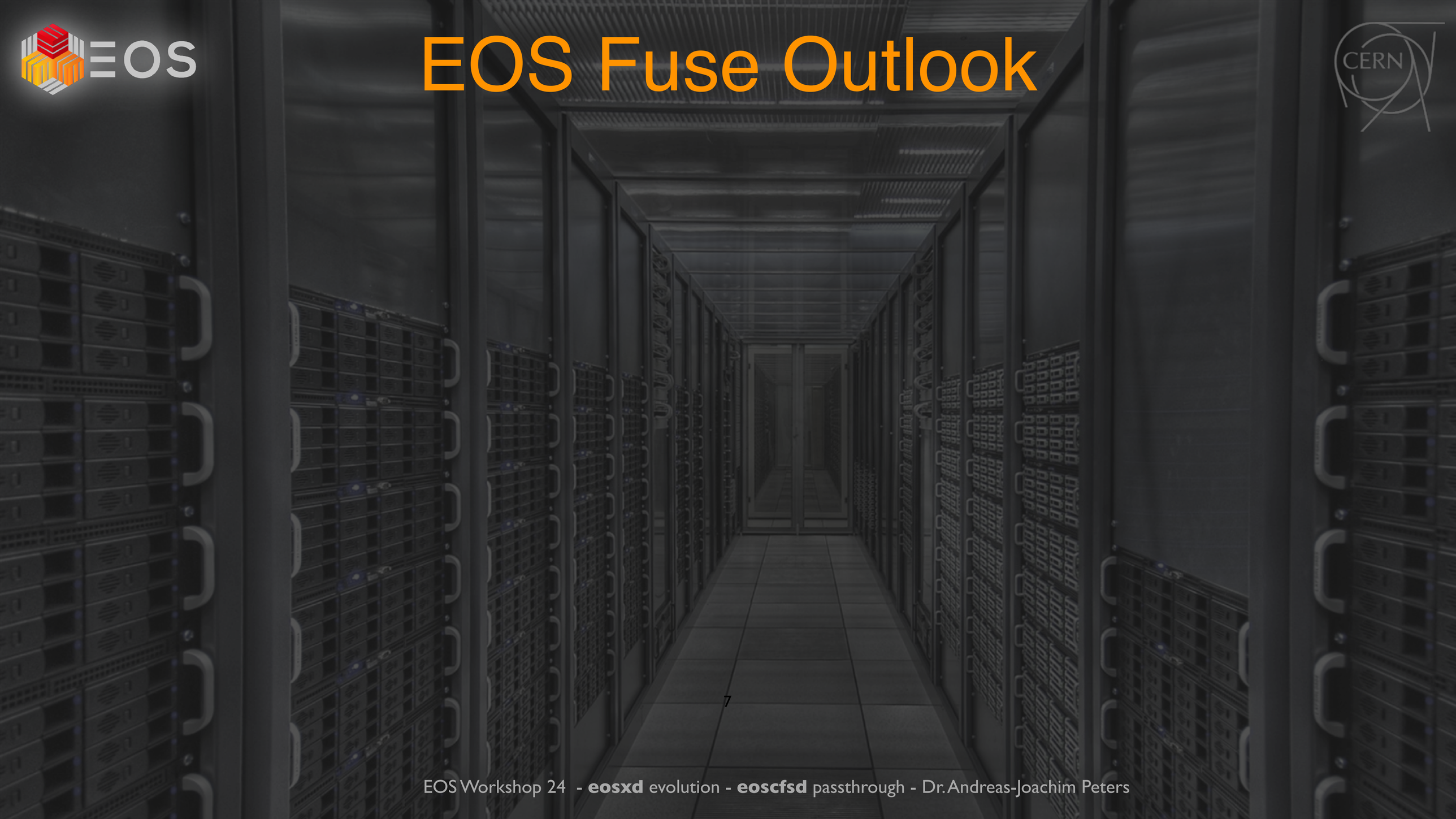
  global-flush is preventing XrdCl::File::Open of a file while the contents is flushed from a FUSE client - doubles client-server calls
- bypass deletion through recycle-bin if a file is unlinked while it is still open
- fix credential hash function (interference) to distinguish credentials with identical path in different container environments
- verify all KRB credentials before creating connections

EOS Fuse Changes

- credentials are bound in open call and kept for all IO operations after open
- global-flush is now disabled by default

  global-flush is preventing XrdCl::File::Open of a file while the contents is flushed from a FUSE client - doubles client-server calls
- bypass deletion through recycle-bin if a file is unlinked while it is still open
- fix credential hash function (interference) to distinguish credentials with identical path in different container environments
- verify all KRB credentials before creating connections
- bump user ID limit for UNIX authenticated FUSE clients to 1048575

EOS Fuse Outlook



EOS Fuse Outlook

- most relevant performance limits in **eos**xd are addressable in MGM + FST implementation - FUSE bottlenecks play only minor role

EOS Fuse Outlook

- most relevant performance limits in **eos**xd are addressable in MGM + FST implementation - FUSE bottlenecks play only minor role

- **eos**xd would benefit from XATTR caching in kernel - which does not exist

EOS Fuse Outlook

- most relevant performance limits in **eos**xd are addressable in MGM + FST implementation - FUSE bottlenecks play only minor role

- **eos**xd would benefit from XATTR caching in kernel - which does not exist

- **eos**xd could be re-written using XRootD declarative API - would simplify the implementation - not changing the performance

EOS Fuse Outlook

- most relevant performance limits in **eos**xd are addressable in MGM + FST implementation - FUSE bottlenecks play only minor role

- **eos**xd would benefit from XATTR caching in kernel - which does not exist

- **eos**xd could be re-written using XRootD declarative API - would simplify the implementation - not changing the performance

- **eos**xd could also easily run with a different simpler transport protocol than XRootD if feature set is available

EOS Fuse Outlook

- most relevant performance limits in **eos**xd are addressable in MGM + FST implementation - FUSE bottlenecks play only minor role

- **eos**xd would benefit from XATTR caching in kernel - which does not exist

- **eos**xd could be re-written using XRootD declarative API - would simplify the implementation - not changing the performance

- **eos**xd could also easily run with a different simpler transport protocol than XRootD if feature set is available

- **eos**xd is nowadays very stable - the connection interface might need a redesign to avoid clients creating too many connections

EOS Fuse Outlook

- most relevant performance limits in **eos**xd are addressable in MGM + FST implementation - FUSE bottlenecks play only minor role

- **eos**xd would benefit from XATTR caching in kernel - which does not exist

- **eos**xd could be re-written using XRootD declarative API - would simplify the implementation - not changing the performance

- **eos**xd could also easily run with a different simpler transport protocol than XRootD if feature set is available

- **eos**xd is nowadays very stable - the connection interface might need a redesign to avoid clients creating too many connections

- careful when updating from 5.1 to 5.2 - use server >5.2.15

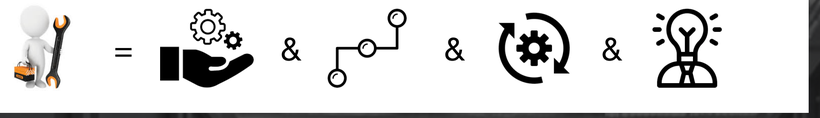
eoscfsd

eoscfsd

CLIENT FILE SYSTEM

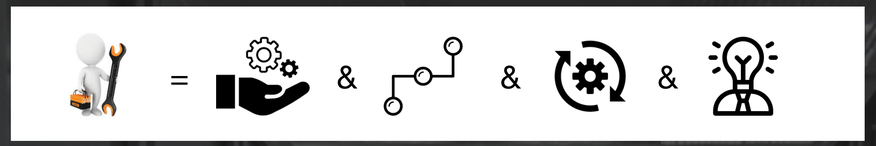
What is CFS?

CLIENT FILE SYSTEM



What is CFS?

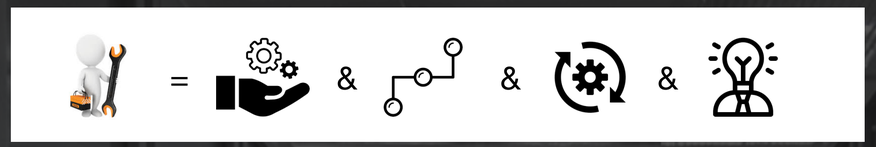
CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**

What is CFS?

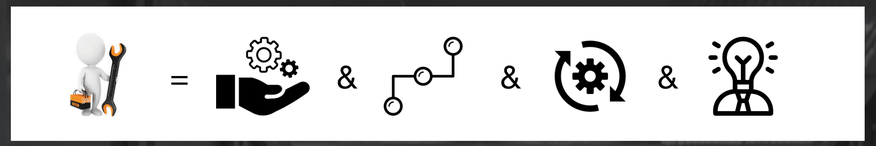
CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**
  - provides homogeneous POSIX API connecting to storage systems

What is CFS?

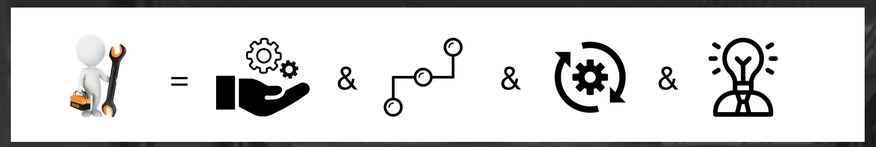
CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**
  - provides homogeneous POSIX API connecting to storage systems
    - as much POSIX as the layer it abstracts + tiny FUSE POSIX violations

What is CFS?

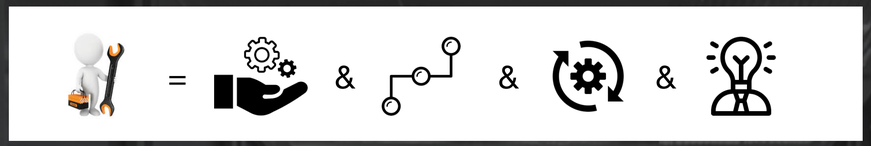
CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**
  - provides homogeneous POSIX API connecting to storage systems
    - as much POSIX as the layer it abstracts + tiny FUSE POSIX violations
  - allows to support arbitrary authentication methods for any back-end

What is CFS?

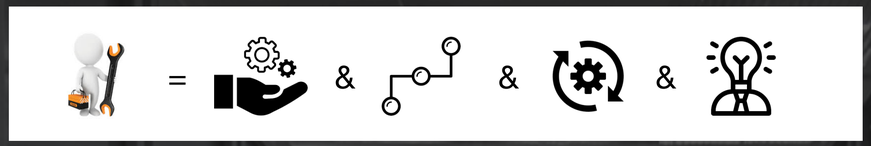
CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**
  - provides homogeneous POSIX API connecting to storage systems
    - as much POSIX as the layer it abstracts + tiny FUSE POSIX violations
  - allows to support arbitrary authentication methods for any back-end
  - allows shared extensions to standard POSIX API

What is CFS?

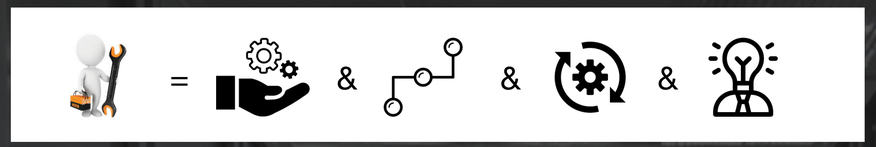
CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**
  - provides homogeneous POSIX API connecting to storage systems
    - as much POSIX as the layer it abstracts + tiny FUSE POSIX violations
  - allows to support arbitrary authentication methods for any back-end
  - allows shared extensions to standard POSIX API
  - provides an exit strategy from any back-end

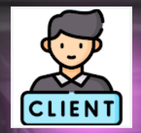
What is CFS?

CLIENT FILE SYSTEM



- **CFS** is a **FS abstraction layer written as a FUSE passthrough filesystem**
  - provides homogeneous POSIX API connecting to storage systems
    - as much POSIX as the layer it abstracts + tiny FUSE POSIX violations
  - allows to support arbitrary authentication methods for any back-end
  - allows shared extensions to standard POSIX API
  - provides an exit strategy from any back-end
- a tiny R&D project - which is now part of EOS releases and usable as **eos**cfsd

Architecture



eoscfsd

low-level FUSE API

core comes from example implementation of libfuse-3

FUSE Passthrough Filesystem

convenience functionality

Mountable Filesystem

anything which can be mounted

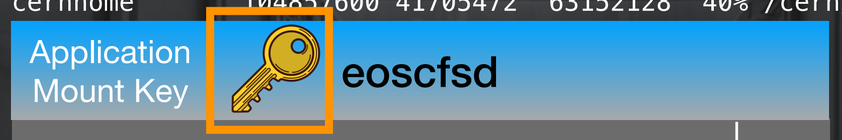
lustre ceph OneDrive NetApp amazon S3 Drive

CFS Prototype

CLIENT FILE SYSTEM

eoscfsd provides a high-performance pass-through implementation for POSIX filesystems. It adds kerberos authentication, remote configuration and mount key obfuscation. **eos**cfsd fetches the mount instructions for a named mount from a configurable HTTPS server.

```
[root~]# df /cern/home/
Filesystem     1K-blocks     Used Available Use% Mounted on
cernhome       104857600 41705472  63152128  40% /cern/home
```



no caching in kernel FUSE layer
read/write redirect to file descriptor

Local Mount Key **/etc/eoscfsd.key**

Private Mount Namespace

CephFS Kernel Client

back-end invisible/inaccessible for root user

https GET (1)

These keys don't remove the need to trust *root* on a host!

mount (2)

FS1 .. N

ceph

OneDrive

Drive

Encrypted Mountpass

CFS Prototype Status

- RPMs available - package **eos-cfsd**
  - **kerberos** authentication
  - virtual **/.proc/** interface
    - kerberos ID to name **translation**
    - ***quota*** bool per user
    - ***enabled*** bool per user
    - **recycle** bin (enable/disable during mount)
    - **bulk deletion** wrapper via shell alias
      (remark: a shell wrapper opens up the possibility to use the recycle bin as free storage resource .. maybe we don't want to allow that)
  - virtual xattr 'cfs.id' = 'who am i'
  - statistics file in JSON format with operations/s etc.
  - autofs support

CFS Documentation

Performance

- unpacking the Linux kernel on CentOS9 office desktop with CephFS disk-based backend /cern/home/
  - cephfs native backend: untars ~ **1100** files/s
  - eos**cfs**d: untars ~**1000** files/s [90% of back-end]
  - afs: untars ~**250** files/s
- IO bottleneck introduced by FUSE is almost invisible
  (reading a file inside the CephFS kernel cache via eos**cfs**d)

```
[apeters@engine apeters]$ dd if=1GB of=/dev/null bs=1M count=1000
1000+0 records in
1000+0 records out
1048576000 bytes (1.0 GB, 1000 MiB) copied, 0.311914 s, 3.4 GB/s
```

Performance Outlook

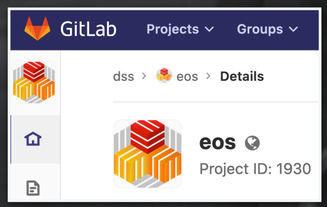
- In kernel version **< 6.9.** FUSE **passthrough** layer receives all read/write requests and redirects them to the back-end filesystem file descriptor
- In kernel version 6.9 FUSE **passthrough** layer can be bypassed for all read/write operations completely - which means you experience the native IOPS of the backend (typical IOPS limit in FUSE 16-20kHz)
- **eos**cfsd is a promising platform to create a back-end agnostic filesystem abstraction with customisable functionality add-ons
- If you are interested - don't hesitate to reach-out, test and/or contribute!

Useful Links

Web Page https://eos.cern.ch

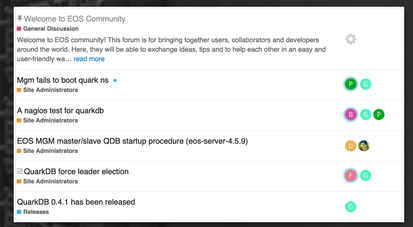
GITLAB Repository https://gitlab.cern.ch/dss/eos

GITHUB Mirror https://github.com/cern-eos/eos

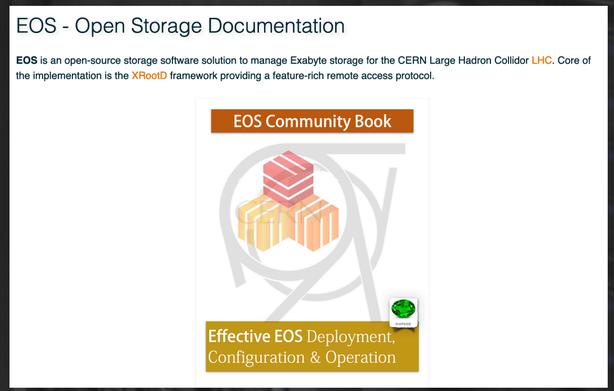


Community Forum https://eos-community.web.cern.ch/

email: eos-community@cern.ch



Documentation http://eos-docs.web.cern.ch/eos-docs/



Support email: eos-support@cern.ch


Thank you for your attention!
Questions?
SOCAR
VESTIAR
CHANGE