CEPH TUTORIAL

GridKA School 2016

Diana Gudu

August 30, 2016

Karlsruhe Institute of Technology

INTRODUCTION ROUND

> WHOAMI

Diana Gudu

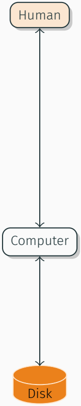
- PhD researcher in Computer Science @KIT
  - distributed multi-agent framework for trading cloud resources
- Human Brain Project
  - work on cloud storage and computing services
- MSc in Computational Science and Engineering @TU Munich
- BSc in Computer Science @Polytechnic University of Bucharest

> WHOAREYOU

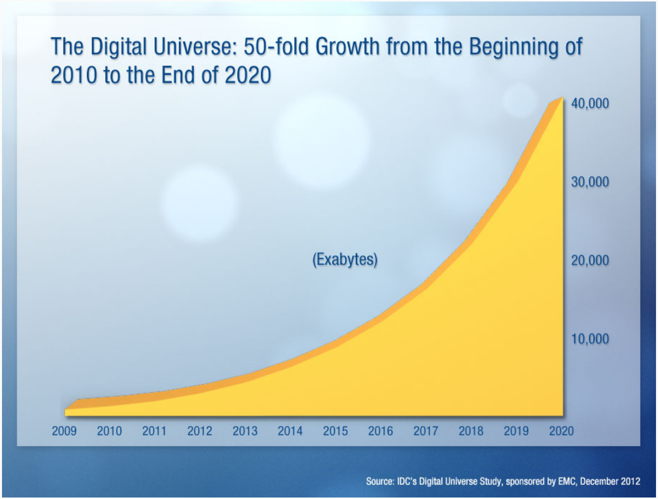
>_

EVOLUTION OF STORAGE

EVOLUTION OF STORAGE



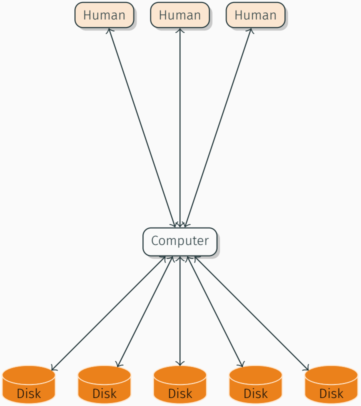
EVOLUTION OF STORAGE



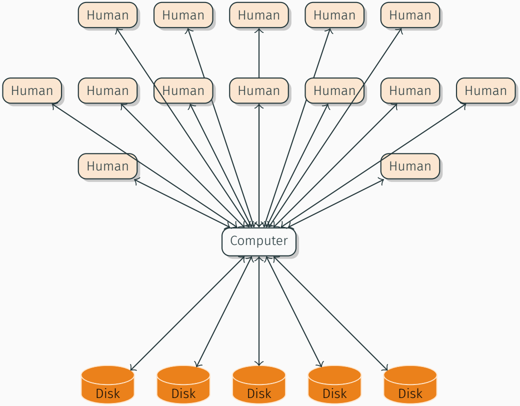
Source: IDC's Digital Universe Study, sponsored by EMC, December 2012

©trumpiot.co

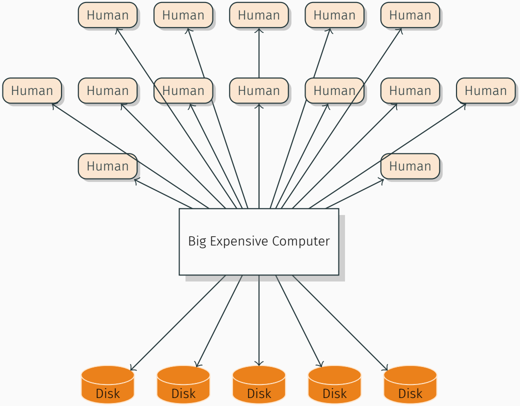
EVOLUTION OF STORAGE



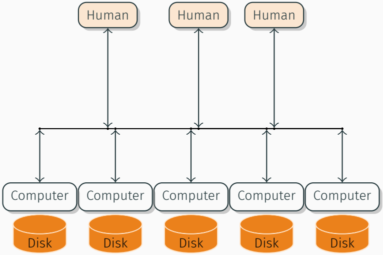
EVOLUTION OF STORAGE



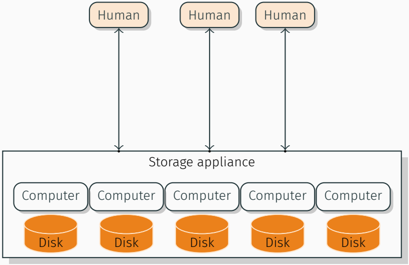
EVOLUTION OF STORAGE



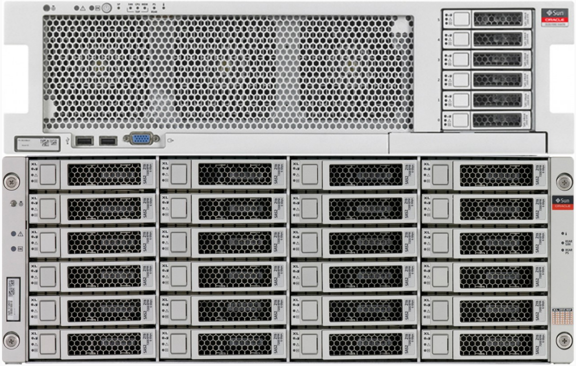
EVOLUTION OF STORAGE



EVOLUTION OF STORAGE

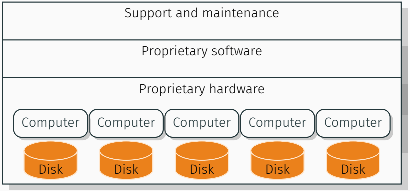


STORAGE APPLIANCE

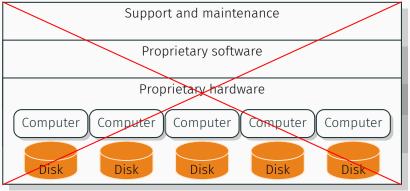


©Oracle

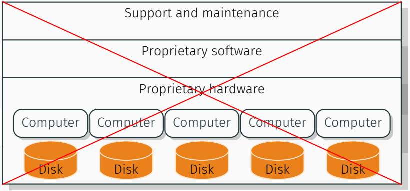
FUTURE OF STORAGE

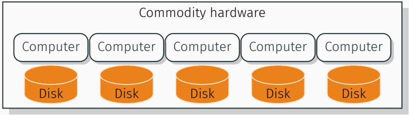


FUTURE OF STORAGE

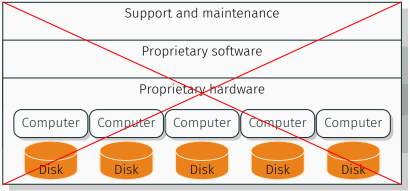


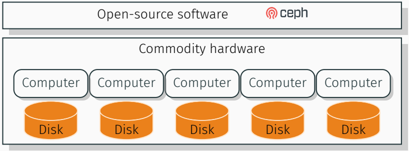
FUTURE OF STORAGE



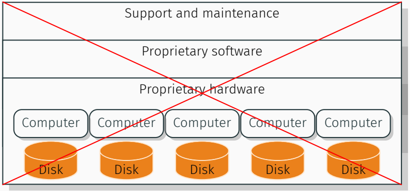


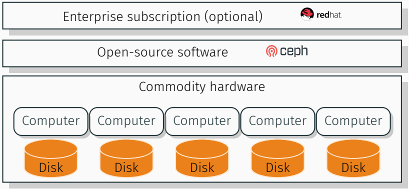
FUTURE OF STORAGE





FUTURE OF STORAGE





CEPH

CEPH

Philosophy

- open-source

CEPH

Philosophy

- open-source
- community focused

Philosophy

- open-source
- community focused
- software-defined

CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF

CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing

CEPH

Philosophy

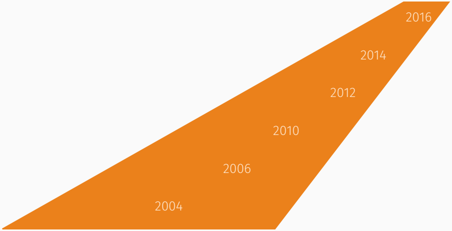
- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History

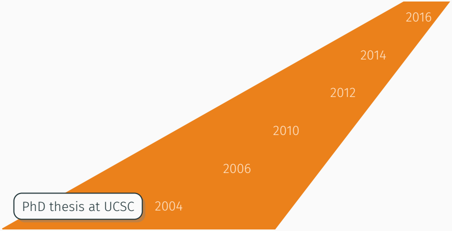


CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History

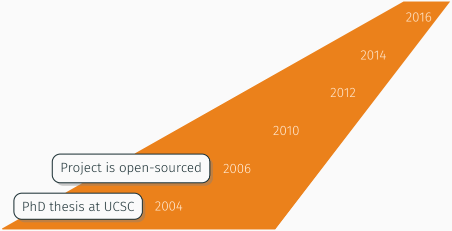


CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History

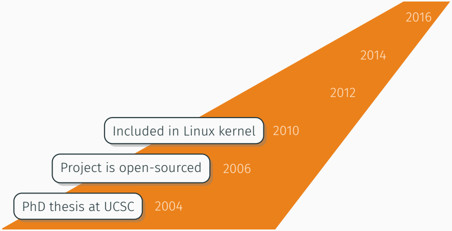


CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History

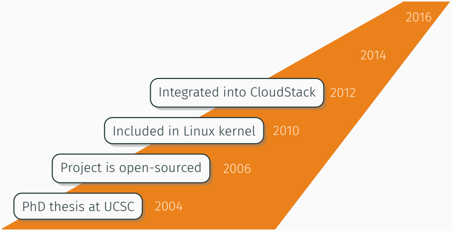


CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History

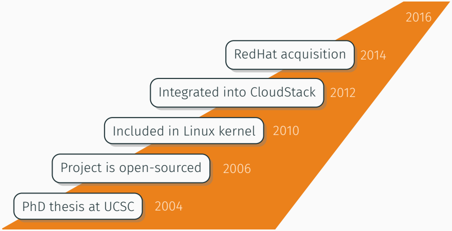


CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History



2016

RedHat acquisition 2014

Integrated into CloudStack 2012

Included in Linux kernel 2010

Project is open-sourced 2006

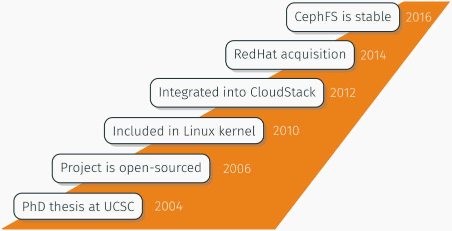
PhD thesis at UCSC 2004

CEPH

Philosophy

- open-source
- community focused
- software-defined
- scale-out hardware, no SPF
- self-managing
- failure is normal

History



CephFS is stable 2016

RedHat acquisition 2014

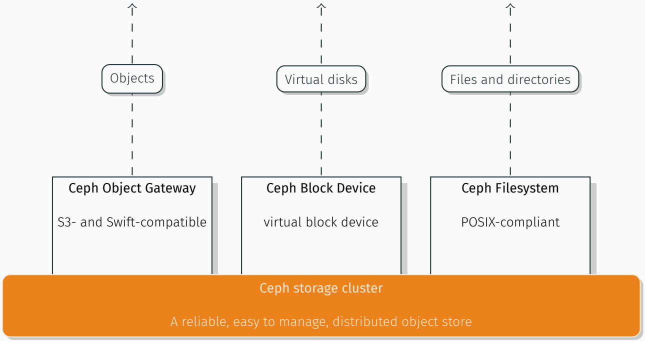
Integrated into CloudStack 2012

Included in Linux kernel 2010

Project is open-sourced 2006

PhD thesis at UCSC 2004

CEPH ARCHITECTURE



Objects

Virtual disks

Files and directories

Ceph Object Gateway

S3- and Swift-compatible

Ceph Block Device

virtual block device

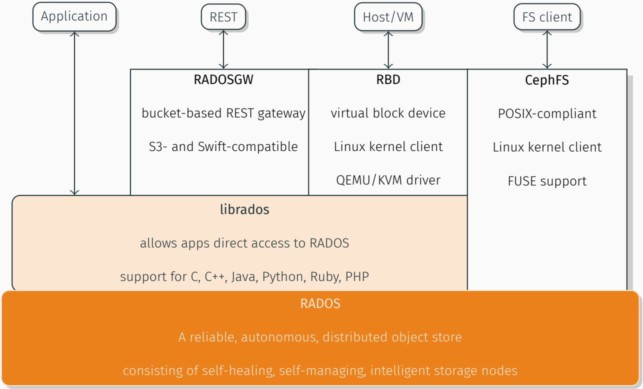
Ceph Filesystem

POSIX-compliant

Ceph storage cluster

A reliable, easy to manage, distributed object store

CEPH ARCHITECTURE



Application

REST

Host/VM

FS client

RADOSGW

bucket-based REST gateway

S3- and Swift-compatible

RBD

virtual block device

Linux kernel client

QEMU/KVM driver

CephFS

POSIX-compliant

Linux kernel client

FUSE support

librados

allows apps direct access to RADOS

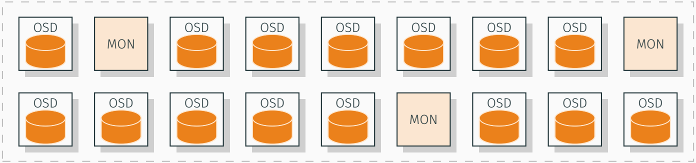
support for C, C++, Java, Python, Ruby, PHP

RADOS

A reliable, autonomous, distributed object store

consisting of self-healing, self-managing, intelligent storage nodes

RADOS


OSD
MON
OSD
OSD
OSD
OSD
OSD
OSD
MON
OSD
OSD
OSD
OSD
OSD
MON
OSD
OSD
OSD

CEPH DAEMONS

OSD

- serve objects to clients
- one per disk
- backend: btrfs, xfs, ext4
- peer-to-peer replication and recovery
- write-ahead journal

MON

- maintain cluster state and membership
- vote for distributed decision-making
- small, odd number

DATA PLACEMENT

CHALLENGES

How to…

…ensure *infinite* scalability?

CHALLENGES

How to...

...ensure *infinite* scalability?

- **compute** object locations instead of looking them up

CHALLENGES

How to...

...ensure *infinite* scalability?

- **compute** object locations instead of looking them up

...ensure deterministic placement?

CHALLENGES

How to...

...ensure *infinite* scalability?

- **compute** object locations instead of looking them up

...ensure deterministic placement?

- **pseudo** random algorithm

CHALLENGES

How to...

...ensure *infinite* scalability?

- **compute** object locations instead of looking them up

...ensure deterministic placement?

- **pseudo** random algorithm

...minimize data movement?

CHALLENGES

How to...

...ensure *infinite* scalability?

- **compute** object locations instead of looking them up

...ensure deterministic placement?

- **pseudo** random algorithm

...minimize data movement?

- stable mapping, e.g. consistent hashing

CRUSH

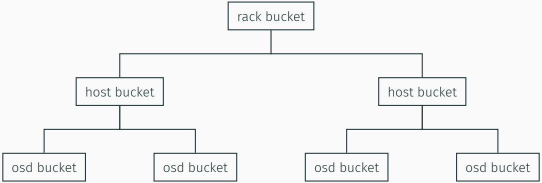
Controlled Replication Under Scalable Hashing

- pseudo-random placement algorithm
- repeatable, deterministic
- statistically uniform distribution
- stable mapping: minimal data migration
- rule-based configuration, topology aware

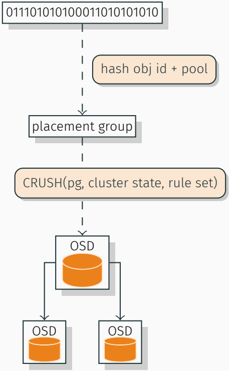
CRUSH

Controlled Replication Under Scalable Hashing

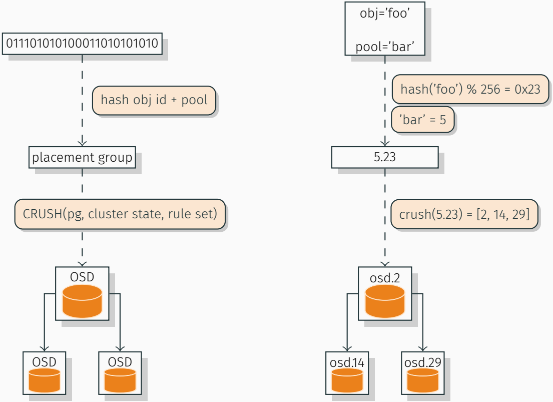
- pseudo-random placement algorithm
- repeatable, deterministic
- statistically uniform distribution
- stable mapping: minimal data migration
- rule-based configuration, topology aware



CRUSH


011101010100011010101010
hash obj id + pool
placement group
CRUSH(pg, cluster state, rule set)
OSD
OSD
OSD

CRUSH


011101010100011010101010
hash obj id + pool
placement group
CRUSH(pg, cluster state, rule set)
OSD
OSD
OSD
obj='foo'
pool='bar'
hash('foo') % 256 = 0x23
'bar' = 5
5.23
crush(5.23) = [2, 14, 29]
osd.2
osd.14
osd.29

DATA PLACEMENT

Pools

- logical groups for storing objects
- manage
  - # PGs
  - # replicas
  - ruleset
  - permissions
  - snapshots

DATA PLACEMENT

Pools

- logical groups for storing objects
- manage
  - # PGs
  - # replicas
  - ruleset
  - permissions
  - snapshots

Placement groups

- fragments of logical groups
  - aggregate objects within a pool for scalability
- 1 PG over several OSDs (# replicas)
- several PGs per OSD ($\approx$ 100)
- more PGs

  $\Longrightarrow$ data durability

  $\Longrightarrow$ even distribution

  $\Longrightarrow$ resource overhead

CEPH CLIENTS

LIBRADOS

- direct access to RADOS for applications
- C, C++, Python, Java, Erlang, PHP
- native socket access, no HTTP overhead

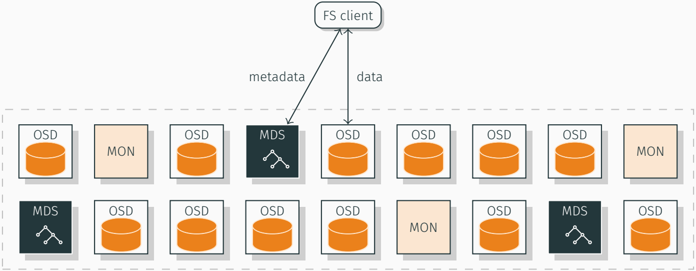
RADOSGW

- RESTful API
- unified object namespace
- S3 and Swift compatible
- user database and access control
- usage accounting, billing

RBD

- storage of disk images in RADOS
- images are striped across the cluster
- decoupling of VMs from host
- thin provisioning
  - physical storage only used once you begin writing
- snaphots, copy-on-write clones
- support in Qemu, KVM

CephFS


FS client
metadata
data
OSD
MON
OSD
MDS
OSD
OSD
OSD
OSD
MON
MDS
OSD
OSD
OSD
OSD
MON
OSD
MDS
OSD

CephFS

- files striped over RADOS objects
- strong consistency for POSIX semantics
  - use O_LAZY to relax consistency

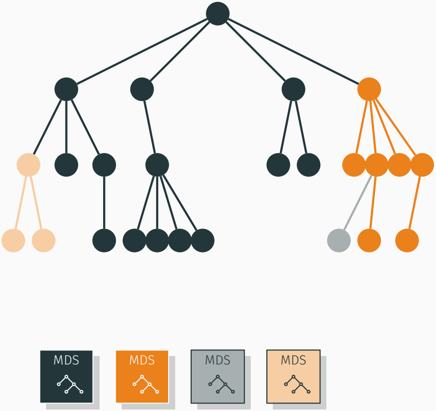
CephFS

- files striped over RADOS objects
- strong consistency for POSIX semantics
    - use O_LAZY to relax consistency

Metadata Server

- manages metadata for POSIX-compliant filesystem
    - directory hierarchy
    - file metadata: owner, timestamps, mode etc
- stores metadata in RADOS
- multiple MDS for HA and load balancing

DYNAMIC SUBTREE PARTITIONING



FINAL REMARKS

WHAT MAKES CEPH UNIQUE

- CRUSH magic
- clustered, dynamic metadata management
- thin provisioning of block storage
- unified storage (object, block, file)

CEPH VS. THEM[1]

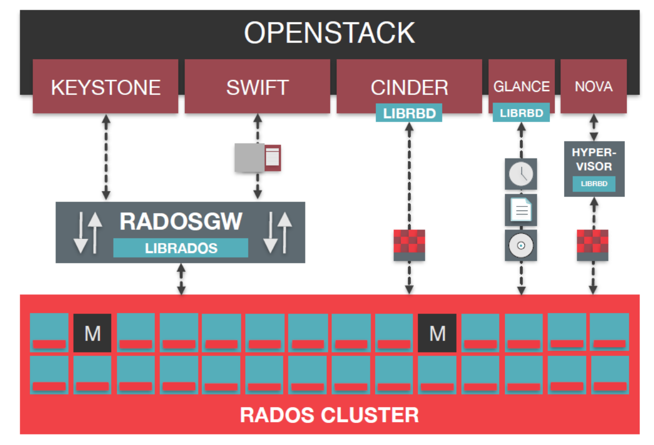
	HDFS	iRODS	Ceph	GlusterFS	Lustre
Architecture	centralized	centralized	distributed	decentralized	centralized
Naming	index	database	CRUSH	EHA	index
API	CLI, FUSE, REST, API	CLI, FUSE, API	FUSE, mount, REST	FUSE, mount	FUSE
Fault detection	fully connect.	P2P	fully connect.	detected	manually
System availability	no failover	no failover	high	high	failover
Data availability	replication	replication	replication	RAID-like	no
Placement strategy	auto	manual	auto	manual	no
Replication	async.	sync.	sync.	sync.	RAID-like
Cache consistency	WORM, lease	lock	lock	no	lock
Load balancing	auto	manual	manual	manual	no

[1] Benjamin Depardon, Gël Le Mahec, Cyril Séguin. Analysis of Six Distributed File Systems. [Research Report] 2013, pp.44. <hal-00789086>

PRACTICAL CONSIDERATIONS

- separate cluster and public networks
- SSD journals: accelerate bursts and random writes
- memory: 1-2 GB per OSD, CPU: 1.5 GHz per OSD
- redundancy
  - replication: increased read performance, capacity impact
  - erasure coding: better space efficiency, high CPU overhead, ~~RBD~~
  - cache tiering: write-back overlay pool; combine replication and erasure coding

OPENSTACK INTEGRATION *



©RedHat

JOIN THE COMMUNITY!

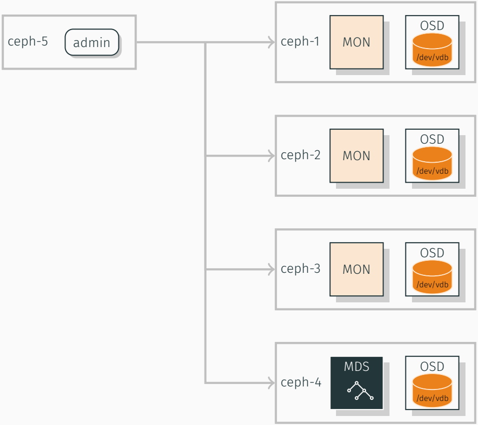
- docs: ceph.com/docs
- wiki: wiki.ceph.com
- mailing lists
  - ceph-users: ceph-users@ceph.com
  - ceph-devel: ceph-devel@vger.kernel.org
- IRC channels (server irc.oftc.net)
  - #ceph
  - #ceph-devel
- github: github.com/ceph/ceph
- get involved: ceph.com/community/contribute

TUTORIAL

OVERVIEW

- deploy a Ceph cluster
- basic operations with the storage cluster
- data placement: CRUSH
- Ceph Filesystem
- block storage: RBD
- advanced topics: erasure coding
- troubleshooting challenge

CLUSTER SET-UP


ceph-5
admin
ceph-1
MON
OSD
/dev/vdb
ceph-2
MON
OSD
/dev/vdb
ceph-3
MON
OSD
/dev/vdb
ceph-4
MDS
OSD
/dev/vdb