CephFS as a service with OpenStack Manila

John Spray

john.spray@redhat.com
jcsp on #ceph-devel

Agenda

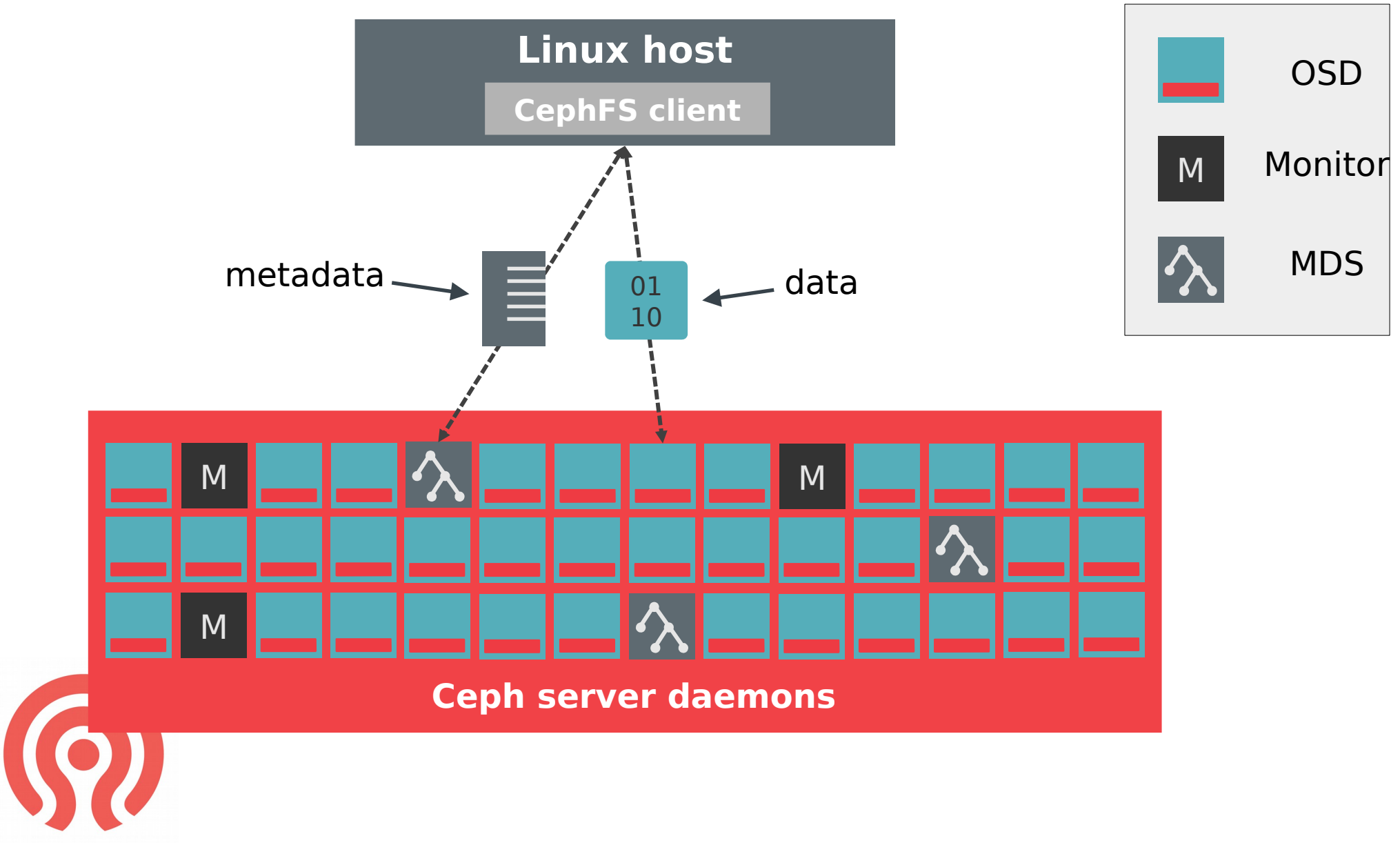
- Brief introductions: Ceph, Manila
- Mapping Manila concepts to CephFS
- Experience implementing native driver
- How to use the driver
- Future work: VSOCK, NFS

Introductions

CephFS

- Distributed POSIX filesystem:
  - Data and metadata stored in RADOS
  - Cluster of Metadata servers
- Shipped with upstream Ceph releases
- Clients: fuse, kernel, libcephfs
- Featureful: directory snapshots, recursive statistics

CephFS


Linux host
CephFS client
metadata
01
10
data
OSD
M
Monitor
MDS
M
M
M
Ceph server daemons

Learn more about CephFS

CephFS in Jewel: Stable at Last

- ***Gregory Farnum***
- ***Thursday 11:00 AM***
- ***Level 4 MR 12 A/B (i.e. here)***

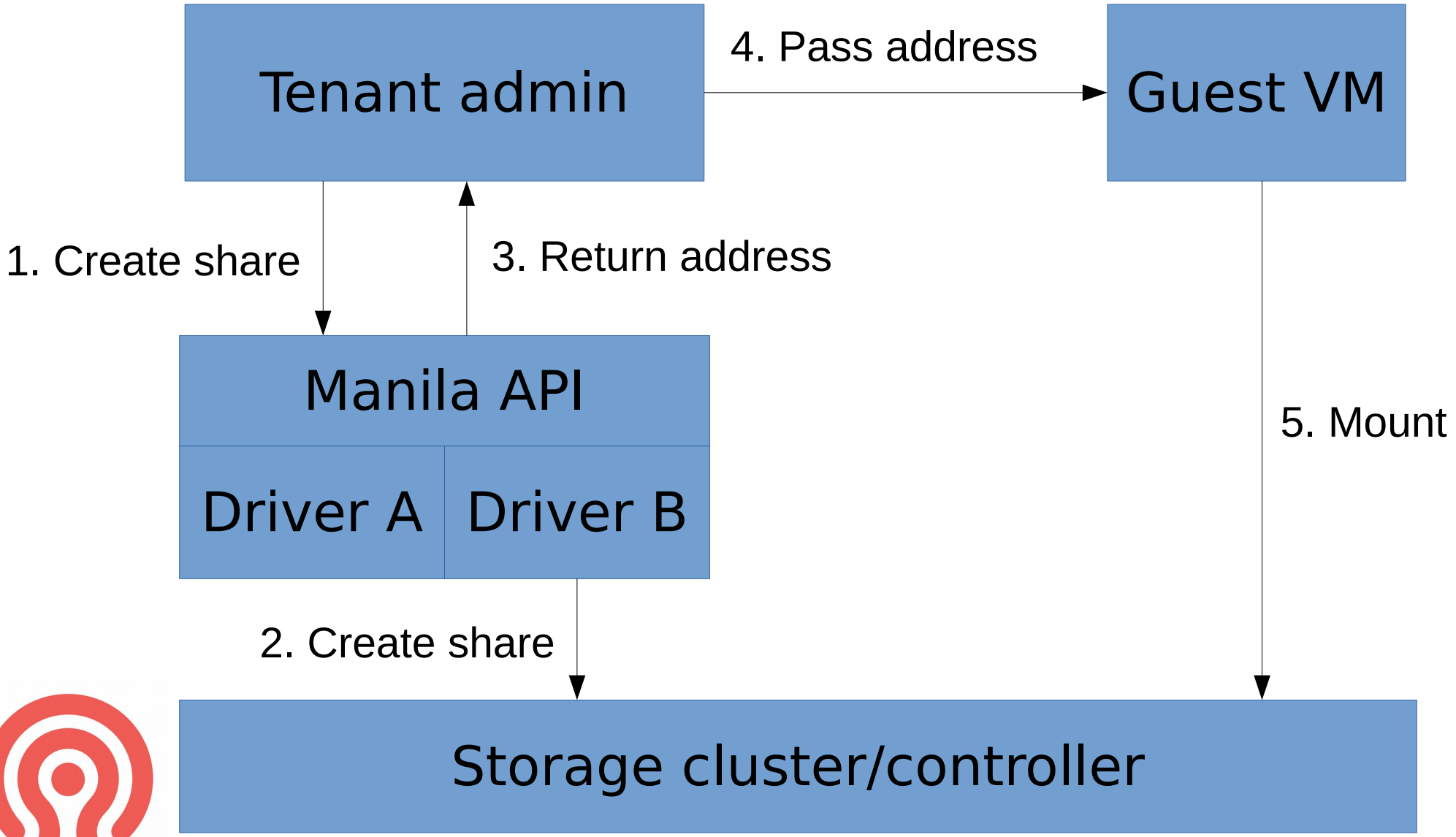
http://docs.ceph.com/docs/master/cephfs/

https://www.youtube.com/results?search_query=cephfs

Manila

- OpenStack shared filesystem service
- APIs for tenants to request filesystem shares, fulfilled by driver modules
- Existing drivers mainly for proprietary storage devices, with a couple of exceptions:
  - GlusterFS
  - “Generic” (NFS-on-Cinder)

Manila


Tenant admin
4. Pass address
Guest VM
1. Create share
3. Return address
Manila API
Driver A
Driver B
5. Mount
2. Create share
Storage cluster/controller

Why integrate CephFS and Manila?

- Most OpenStack clusters already include a Ceph cluster (used for RBD, RGW)
- An open source backend for your open source cloud
- Testers and developers need a real-life Manila backend that is free
- Enable filesystem-using applications to move into OpenStack/Ceph clouds

Manila concepts and CephFS

Shares

- Manila operates on “shares”
  - An individual filesystem namespace
  - Is both a unit of storage and a unit of sharing
  - Expected to be of limited size
- No such concept in CephFS, but we do have the primitives to build it.

Implementing shares

- In the CephFS driver, a share is:
  - A directory
  - ...which might have a *layout* pointing to a particular pool or RADOS namespace
  - ...which has a quota set to define the size
  - ...access to which is limited by “path=...” constraints in MDS authentication capabilities (“auth caps”).

Prerequisites in CephFS

- New: `path=` auth rules
- New: Prevent clients modifying pool layouts (“rwp” auth caps)
- New: quota as df
- New: remote “session evict”, with filtering (kicking sessions for a share/ID)
- Existing: quotas, rstats, snapshots

Implementing shares

```
/
volumes
    my_share/
```

Directory

```
[client.alice]
 caps: [mds] allow rw path=/volumes/my_share
 caps: [osd] allow rw namespace=fsvolumens_my_share
```

Auth cap

```
getfattr -n ceph.quota.maxbytes /volumes/my_share
# file: volumes/my_share
ceph.quota.max_bytes="104857600"
```

Quota

```
ceph-fuse –name client.alice
          –client_mountpoint=/volumes/my_share
          /mnt/my_share
```

Mount command

Access rules

- Manila expects each share to have a list of access rules. Giving one endpoint access to two shares means listing it in the rules of both shares.
- Ceph stores a list of cephx identities, each identity has a list of paths it can access.
- i.e. the opposite way around...

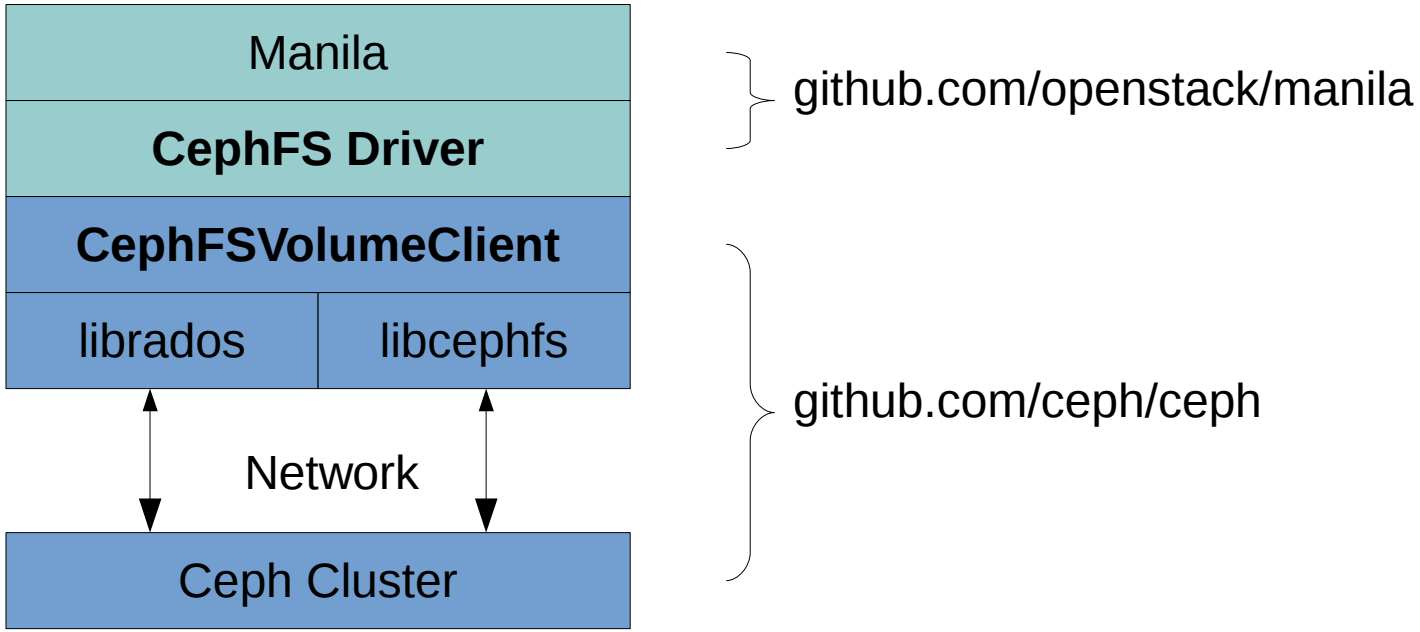
Implementing access rules

- In initial driver these are directly updated in ceph auth caps
- Not sufficient:
  - Need to count how many shares require access to an OSD pool
  - Since Mitaka, Manila requires efficient listing of rules by share
- Point release of driver will add a simple index of access information.

CephFSVolumeClient

- We do most of the work in our new python interface to Ceph
- Present a “volume” abstraction that just happens to match Manila's needs closely
- Initially very lightweight, becoming more substantial to store more metadata to support edge-cases like `update_access()`
- Version 0 of interface in Ceph *Jewel*

CephFSVolumeClient



Lessons from implementing a Manila driver

Manila driver interface

- Not a stable interface:
  - Continuously modified to enable Manila feature work
  - Driver authors expected to keep up
- Limited documentation
- Drivers can't define their own protocols, have to make modifications outside of driver

Adding a protocol

- Awkward: Manila has expectation of drivers implementing existing protocols (NFS, CIFS), typical for proprietary filers.
- Open source filesystems typically have their own protocol (CephFS, GlusterFS, Lustre, GFS2)
- To add protocol, it is necessary to modify Manila API server, and Manila client, and handle API microversioning.

Tutorial: Using the CephFS driver

Caveats

- Manila >= *Mitaka* required
- CephFS >= *Jewel* required
- Guests need IP connectivity to Ceph cluster: think about security
- Guests need the CephFS client installed
- Rely on CephFS client to respect quota

Set up CephFS

```
ceph-deploy mds create myserver
ceph osd pool create fs_data
ceph osd pool create fs_metadata
ceph fs new myfs fs_metadata fs_data
```

Configuring Manila (1/2)

- Create client.manila identity
  - Huge command line, see docs!
- Install librados & libcephfs python packages on your manila-share server
- Ensure your Manila server has connection to Ceph public network
- Test: run **`ceph –name=client.manila status`**

Configuring Manila (2/2)

- Dump the client.manila key into /etc/ceph/ceph-client.manila.keyring
- Configure CephFS driver backend:

/etc/manila/manila.conf:

```
[cephfs1]
driver_handles_share_servers = False
share_backend_name = CEPHFS1
share_driver = manila.share.drivers.cephfs. \
          cephfs_native.CephFSNativeDriver
cephfs_conf_path = /etc/ceph/ceph.conf
cephfs_auth_id = manila
```

$ manila type-create cephfstype false

Creating and mounting a share

From your OpenStack console:

```
$ manila type-create cephfstype false
$ manila create --share-type cephfstype \
                --name cephshare1 cephfs 1
$ manila access-allow cephshare1 cephx alice
```

From your guest VM:

```
# ceph-fuse –id=alice
    -c ./client.conf –keyring=./alice.keyring
    --client-mountpoint=/volumes/share-4c55ad20
    /mnt/share
```

Currently have to fetch key with Ceph CLI

Multiple backends

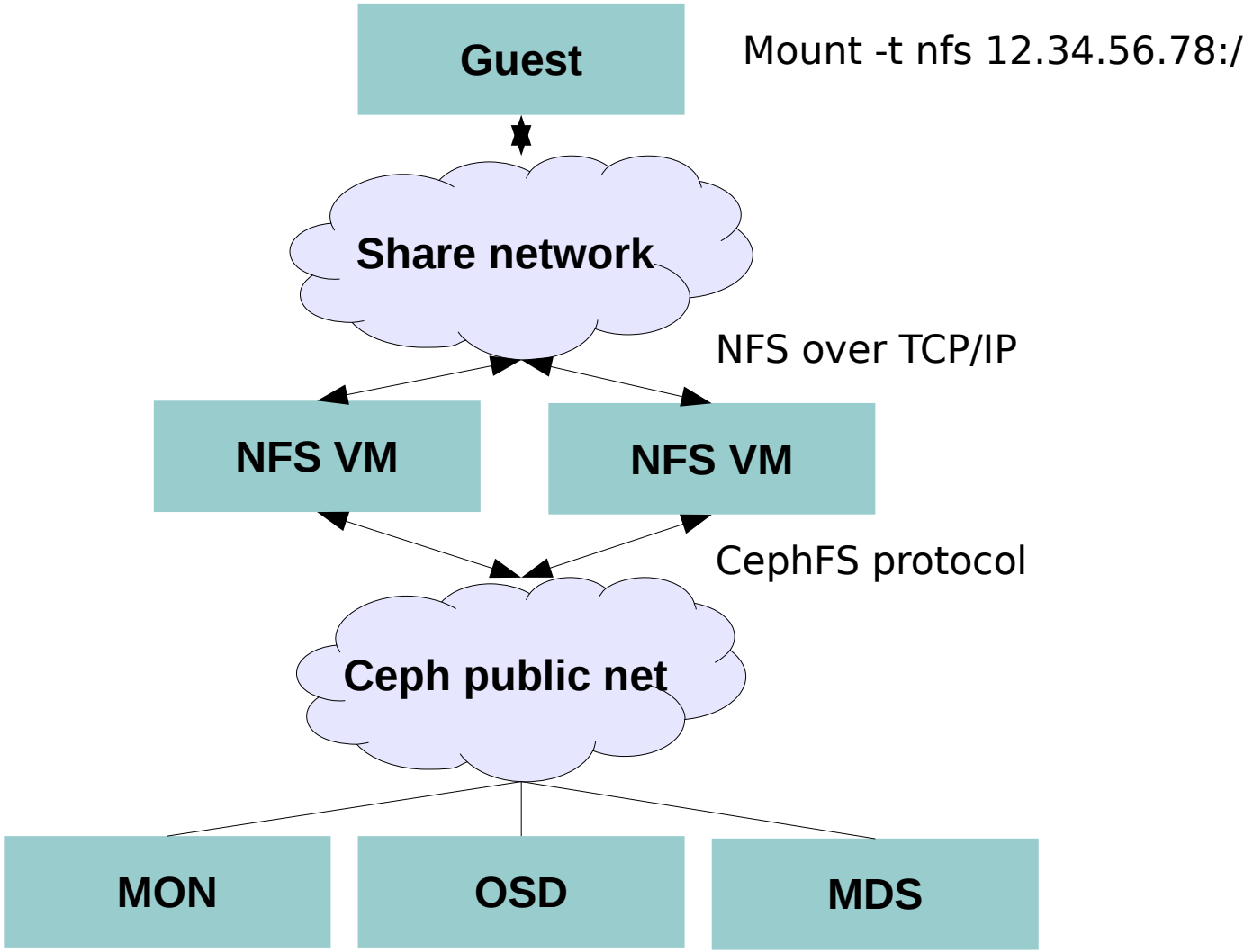
- Use multiple Manila backends to target different storage:
  - Different data pools
  - Different MDSs (experimental multi-fs in Jewel)
  - Different Ceph clusters
- Could pull some of this into single driver instance but not clear if needed

Future work

NFS gateways

- Manila driver instantiates service VMs containing Ceph client and NFS server
- Service VM has network interfaces on share network (NFS to clients) and Ceph public network.
- Implementation not trivial:
  - Clustered/HA NFS servers
  - Keep service VMs alive/replace on failure

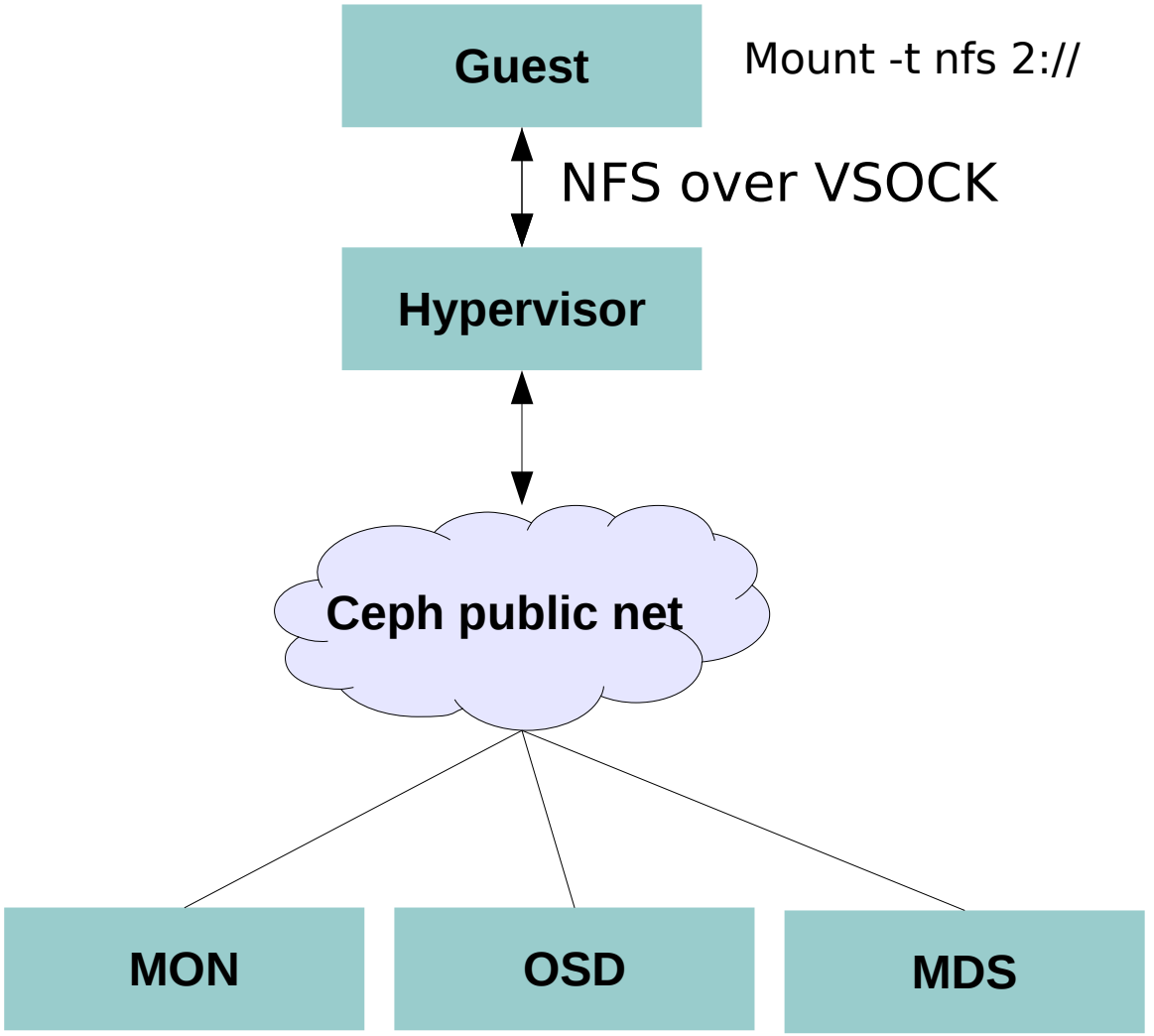
NFS gateways



Hypervisor-mediated

- Terminate CephFS on hypervisor host, expose to guest locally
- Guest no longer needs any auth or addr info: connect to the hypervisor and it'll get there
- Requires compute (Nova) to be aware of attachment to push NFS config to the right hypervisor at the right time.
- **Huge benefit for simplicity and security**

Hypervisor-mediated



VSOCK

- Efficient general purpose socket interface from KVM/qemu hosts to guests
- Avoid maintaining separate guest kernel code like virtfs, use the same NFS client but via a different socket type
- Not yet in mainline kernel
- See past presentations:
  - Stefan Hajnoczi @ KVM Forum 2015
  - Sage Weil @ OpenStack Summit Tokyo 2015

Nova changes needed

- Hypervisor mediate share access needs something aware of shares and guests.
  - Add ShareAttachment object+API to Nova
  - Hook in to expose FS during guest startup
- Attachments need to know
  - How to get at the filesystem? (CephFS, NFS)
  - How to expose it to the guest? (VSOCK, virtfs/9p)

https://review.openstack.org/#/c/310050/1/specs/newton/approved/fs-attach-detach.rst

Hypervisor-mediated (VSOCK)

- VSOCK hooks
  - VSOCK: Nova needs to learn to configure ganesha, map guest name to CID
  - VSOCK: Libvirt needs to learn to configure VSOCK interfaces
  - VSOCK: Ganesha needs to learn to authenticate clients by VSOCK CID (address)

Shorter-term things

- Ceph/Manila: Implement update_access() properly (store metadata in CephFSVolumeClient #15615)
- Manila Improved driver docs (in progress)
- Ceph: backport RADOS namespace integration (thanks to eBay team)
- Manila: Expose Ceph keys in API
- Manila: Read only Shares for clone-from-snapshot

Physical Isolation

- Current driver has “data_isolated” option that creates pool for share (bit awkward, guessing a pg_num)
- Could also add “metadata_isolated” to create true filesystem instead of directory, and create a new VM to run the MDS from Manila.
- In general shares should be lightweight by default but optional isolation is useful.

Get Involved

- Lots of work to do!
- Vendors: package, automate Manila/CephFS deployment in your environment
- Developers:
  - VSOCK, NFS access.
  - New Manila features (share migration etc)
- Users: try it out

Get Involved

Evaluate the latest releases:

http://ceph.com/resources/downloads/

Mailing list, IRC:

http://ceph.com/resources/mailing-list-irc/

Bugs:

http://tracker.ceph.com/projects/ceph/issues

Online developer summits:

https://wiki.ceph.com/Planning/CDS

Questions/Discussion