Ethan Nuessle
nuesse@rpi.edu
Main Repo: https://github.com/ceph/ceph

Setting up a Development Environment

In order to compile and run the code, you need to get set up with an environment that Ceph can run on. You can either do this through your local machine, or through a vpn.

VPN

Depending on whether you are working through a Mac or Linux (or WSL) system, you will need to either use Tunnelblick (Mac) or OpenVPN (Linux). The basic guide to work with VPN on Ceph is here:

https://wiki.sepia.ceph.com/doku.php?id=vpnaccess#mac_os_x

For all systems, the basic flow is,

1. Create an account on the Ceph tracker site.
2. Generate a public SSH Key
3. Set up VPN and copy the VPN credentials (these are only generated once, so make sure to copy and save them somewhere)
4. Do a post on the Ceph tracker site (if this is for RCOS, you should have been linked to a special RCOS thread). Include your desired username, email, public SSH Key, and VPN Credentials.

Local Machine

In order to run the code locally, you need a Linux system, or a Linux virtual machine of some kind. Docker and WSL are common choices if you don't have Linux. Ceph supports many different Linux distros, but the main one I would recommend, and that I know works well, is Ubuntu 22.04 (Jammy Jellyfish). If you are running Ubuntu and it is running into compilation errors, check to make sure it is some version of Jammy Jellyfish. (This may become outdated as Ceph is currently adding support for newer Ubuntu versions).

Commands to Build

In order to build the project, you will need to clone the repository, and run a few commands to build it. The following commands are in the terminal. Building the repository will likely take a bit more than 16gb, so keep that in mind.

Main Repository (Ceph Stable Branch):

```
git clone https://github.com/ceph.git
cd ceph
./install-deps.sh
./do_cmake.sh -DCMAKE_BUILD_TYPE=RelWithDebInfo
ninja vstart-base cephfs cython_cephfs cython_rbd
```

Forked Repository:

```
git clone https://github.com/ceph.git
cd ceph
git remote add <name of fork> <url to fork>
git fetch <name of fork>
git checkout -track <name of fork>/<name of branch>
git pull <name of fork> <name of branch> --rebase
./install-deps.sh
./do_cmake.sh -DCMAKE_BUILD_TYPE=RelWithDebInfo
ninja vstart-base cephfs cython_cephfs cython_rbd
```

Notes:

If you want it to compile a bit quicker and likely take up (a bit) less space, you can replace the clone command with

```
git clone https://github.com/ceph.git --depth 1
```

, this makes it clone a bit less, but has some potential to run into issues in niche cases.
It is recommended to try this first, and if issues come up, you can try again without the depth modifier.

For cloning a forked repository, you with `<name of fork>` and `<url to fork>`, you can follow this format, using this repository fork as an example: https://github.com/enuessle/ceph

```
git clone https://github.com/ceph.git
cd ceph
git remote add enuessle git@github.com:enuessle/ceph.git
git fetch enuessle
git checkout -track enuessle/main
git pull enuessle main --rebase
./install-deps.sh
./do_cmake.sh -DCMAKE_BUILD_TYPE=RelWithDebInfo
ninja vstart-base cephfs cython_cephfs cython_rbd
```

Errors in Building

Building Ceph takes quite a while, around 2 hours to finish on my local machine. There are also many errors that may come up. This is a (non-comprehensive) list of different errors that have come up and ways to fix it and get it to compile and build.

Failed to Recurse into submodule

```
Aborting
fatal: Unable to checkout '59cca28a61ae3154ba1731ca2ee7daa402ec5e2e' in
submodule path 'src/nvmeof/gateway/spdk/xnvme'
fatal: Failed to recurse into submodule path 'src/nvmeof/gateway/spdk'
fatal: Unable to checkout '84714379121c19f89a8145fee179d6388bf74c1e' in
submodule path 'ceph-object-corpus'
fatal: Failed to recurse into submodule path 'src/nvmeof/gateway'
```

This issue occurs with the `./do_cmake,sh -DCMAKE_BUILD_TYPE=RelWithDebInfo` command. It was caused by an issue with installing dependencies.
Possible causes include

- Not running `./install-deps.sh`
- Not letting `./install-deps.sh` complete
- Not running `./install-deps.sh` in the same location you are building the repository

In my case, the root issue was trying to run
`./do_cmake,sh -DCMAKE_BUILD_TYPE=RelWithDebInfo`
on an external drive where `./install-deps.sh` was not able to be completed properly.

Solutions:

- If possible, build Ceph in the original location that you cloned it from.
- If storage is a concern, rerun `./install-deps.sh` in the new location
- If `./install-deps.sh` fails, likely due to a permissions error, then you might be trying to install dependencies on a drive without a supported format. Check your drive format using the `fsck` command. If it is not ext4 (or some other file system that supports Unix File Permissions), then try to either use a different drive that is ext4, or reformat (if possible) to ext4.

In my case this error was solved by using a new flashdrive, formatting it to ext4, and then cloning from the beginning.

Errors during ninja

While running `ninja vstart-base cephfs cython_cephfs cython_rbd` it might run into errors. In my case, it was failing at compiling because of the missing package Boost. Solving this issue just requires reinstalling dependencies, and redoing cmake. If any other issue comes up, solve those first, as those will likely be causing the missing packages. Delete the build directory, and run these commands

```
./install-deps.sh
./do_cmake,sh -DCMAKE_BUILD_TYPE=RelWithDebInfo
ninja vstart-base cephfs cython_cephfs cython_rbd
```

OpenVPN not staying active

This issue is when OpenVpn shuts down (deactivates) a few seconds after starting. If you are running into OpenVPN issues, namely shh hanging, you can check to see if this is the issue by running these commands

```
sudo systemctl restart openvpn@sepia
sudo systemctl enable openvpn@sepia
sudo systemctl status openvpn@sepia
```

If, within a few seconds after running `sudo systemctl enable openvpn@sepia` , `sudo systemctl status openvpn@sepia` shows `Active: inactive (dead)` , then this might be your issue.

Solution:

Uninstall OpenVPN (likely `apt-get remove openvpn`)
Reinstall it
Follow the guide to set up the vpn again, creating new credentials
Repost the credentials on the Ceph Tracker

SSH “Permission denied”

```
ssh USER@smithi061.front.sepia.ceph.com
USER@smithi061.front.sepia.ceph.com's password:
Permission denied, please try again.
```

This issue generally requires fixing or help from the server admins. The best thing to do is tell the project lead, or post a comment on the VPN tracker we were given.

Merge Commits

For Ceph, you don't want to have any merge commits when you do a pull request. Instead of merging, you should use git rebase.

```
git pull <name of fork> <name of branch> --rebase
```

Using this command should prevent merges from happening on your branch.

How to get rid of Merges

Ideally, you shouldn't make any merge commits, but if you do it is fixable. Messing with git history is both hard and has potential to go wrong, so tread with caution. Instead of potentially making mistakes, getting help from the external lead is recommended. This is just a guide on how I did it, so it is documented.

Cherry-picking commits:

First, get up to date on the branch you want to fix

```
git checkout --track <name of fork>/<name of branch>
```

Set the git head to main

```
git reset --hard origin/main
```

On GitHub, find the commits you want to KEEP

Copy Full SHA (it's a button on the webpage)

One commit at a time, run

```
git cherry-pick <Full SHA for Commit>
```

Force push the new branch history

```
git push -f <name of fork> <name of branch>
```

This should change the commit history to keep the changes for each commit you cherry-pick, without keeping the merge commit. This rewrites the git history, so after this is done you need to also make sure everyone else is updated. Have them run this set of commands.

```
git fetch <name of fork>
git pull <name of fork> <name of branch> --rebase
```

Pull Requests and Formatting

Sign-Offs

All commits that are included in your Pull Request need to be Signed-Off by the authors. You need to use your real name (not github username).

To Signoff, you can use add a line saying

```
Signed-off-by: <real name> <email address>
```

You can also add “-s” to your commit command to do it automatically, you need to have set your name in your git config first to be able to do this.

What should Commits have

Every commit should be its own “logical change”. While working, it’s fine to do commits for smaller changes, but when doing a Pull Request into Main, you should redo the commits to just a few meaningful commits. You also shouldn’t have individual commits do too much. If you have a single commit that fixes one thing, adds another, and also adds tests, you should separate those out into a few separate commits.

Commit Titles

Commits should be titled like this

```
<module you are changing>: <Summary of Change>
```

Looking at previous commits in the main branch will provide a good example of what to do.

Commit Message

In the commit message, describe in detail what you did, and why you did what you did (if applicable)

Pull Request Title

For the title of the main Pull Request into the main Ceph branch, use the same format of titling as Commit Titles. Include the module you are updating, and a summary of the changes your pull request is making.

Update Documentation

If the changes you are making adds new features, or otherwise changes something that was documented, update the documentation to match your changes. An example is if you add changes that update the telemetry module to add a new collection, include that collection in the documentation of telemetry.

Tests

When you do a Pull Request, your code will be tested automatically, to make sure it doesn’t break existing functionality. It will also tell you if formatting of the pull request and commits is not up to standard. If it doesn’t pass some of the tests, look at what it isn’t passing. It is possible the tests it is not passing are not applicable to your new code. If it looks good, the next thing is to have a team lead look over the code, and have it reviewed by someone who can pull it into main.