
Ceph - a scalable distributed storage system

See <https://ceph.com/> for current information about Ceph.

Contributing Code

Most of Ceph is dual-licensed under the LGPL version 2.1 or 3.0. Some miscellaneous code is either public domain or licensed under a BSD-style license.

The Ceph documentation is licensed under Creative Commons Attribution Share Alike 3.0 (CC-BY-SA-3.0).

Some headers included in the [ceph/ceph](#) repository are licensed under the GPL. See the file [COPYING](#) for a full inventory of licenses by file.

All code contributions must include a valid “Signed-off-by” line. See the file [SubmittingPatches.rst](#) for details on this and instructions on how to generate and submit patches.

Assignment of copyright is not required to contribute code. Code is contributed under the terms of the applicable license.

Checking out the source

Clone the [ceph/ceph](#) repository from github by running the following command on a system that has git installed:

```
1 git clone git@github.com:ceph/ceph
```

Alternatively, if you are not a github user, you should run the following command on a system that has git installed:

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

When the [ceph/ceph](#) repository has been cloned to your system, run the following commands to move into the cloned [ceph/ceph](#) repository and to check out the git submodules associated with it:

```
1 cd ceph
2 git submodule update --init --recursive --progress
```

Build Prerequisites

section last updated 27 Jul 2023

Make sure that `curl` is installed. The Debian and Ubuntu `apt` command is provided here, but if you use a system with a different package manager, then you must use whatever command is the proper counterpart of this one:

```
1 apt install curl
```

Install Debian or RPM package dependencies by running the following command:

```
1 ./install-deps.sh
```

Install the `python3-routes` package:

```
1 apt install python3-routes
```

Building Ceph

These instructions are meant for developers who are compiling the code for development and testing. To build binaries that are suitable for installation we recommend that you build `.deb` or `.rpm` packages, or refer to `ceph.spec.in` or `debian/rules` to see which configuration options are specified for production builds.

To build Ceph, make sure that you are in the top-level `ceph` directory that contains `do_cmake.sh` and `CONTRIBUTING.rst` and run the following commands:

```
1 ./do_cmake.sh
2 cd build
3 ninja
```

`do_cmake.sh` by default creates a “debug build” of Ceph, which can be up to five times slower than a non-debug build. Pass `-DCMAKE_BUILD_TYPE=RelWithDebInfo` to `do_cmake.sh` to create a non-debug build.

Ninja is the buildsystem used by the Ceph project to build test builds. The number of jobs used by `ninja` is derived from the number of CPU cores of the building host if unspecified. Use the `-j` option to limit the job number if the build jobs are running out of memory. If you attempt to run `ninja` and receive a message that reads `g++: fatal error: Killed signal terminated program cc1plus`, then you have run out of memory. Using the `-j` option with an argument appropriate to the hardware on which the `ninja` command is run is expected to result in a successful build. For example, to limit the job number to 3, run the command `ninja -j 3`. On average, each `ninja` job run in parallel needs approximately 2.5 GiB of RAM.

This documentation assumes that your build directory is a subdirectory of the `ceph.git` checkout. If the build directory is located elsewhere, point `CEPH_GIT_DIR` to the correct path of the checkout. Additional CMake args can be specified by setting `ARGS` before invoking `do_cmake.sh`. See `cmake` options for more details. For example:

```
1 ARGS="-DCMAKE_C_COMPILER=gcc-7" ./do_cmake.sh
```

To build only certain targets, run a command of the following form:

```
1 ninja [target name]
```

To install:

```
1 ninja install
```

CMake Options

The `-D` flag can be used with `cmake` to speed up the process of building Ceph and to customize the build.

Building without RADOS Gateway The RADOS Gateway is built by default. To build Ceph without the RADOS Gateway, run a command of the following form:

```
1 cmake -DWITH_RADOSGW=OFF [path to top-level ceph directory]
```

Building with debugging and arbitrary dependency locations Run a command of the following form to build Ceph with debugging and alternate locations for some external dependencies:

```
1 cmake -DCMAKE_INSTALL_PREFIX=/opt/ceph -DCMAKE_C_FLAGS="-Og -g3 -gdwarf  
   -4" \  
2 ..
```

Ceph has several bundled dependencies such as Boost, RocksDB and Arrow. By default, `cmake` builds these bundled dependencies from source instead of using libraries that are already installed on the system. You can opt to use these system libraries, as long as they meet Ceph's version requirements. To use system libraries, use `cmake` options like `WITH_SYSTEM_BOOST`, as in the following example:

```
1 cmake -DWITH_SYSTEM_BOOST=ON [...]
```

To view an exhaustive list of `-D` options, invoke `cmake -LH`:

```
1 cmake -LH
```

Preserving diagnostic colors If you pipe `ninja` to `less` and would like to preserve the diagnostic colors in the output in order to make errors and warnings more legible, run the following command:

```
1 cmake -DDIAGNOSTICS_COLOR=always ...
```

The above command works only with supported compilers.

The diagnostic colors will be visible when the following command is run:

```
1 ninja | less -R
```

Other available values for `DIAGNOSTICS_COLOR` are `auto` (default) and `never`.

Building a source tarball

To build a complete source tarball with everything needed to build from source and/or build a (deb or rpm) package, run

```
1 ./make-dist
```

This will create a tarball like `ceph-$version.tar.bz2` from git. (Ensure that any changes you want to include in your working directory are committed to git.)

Running a test cluster

From the `ceph/` directory, run the following commands to launch a test Ceph cluster:

```
1 cd build
2 ninja vstart          # builds just enough to run vstart
3 ../src/vstart.sh --debug --new -x --localhost --bluestore
4 ./bin/ceph -s
```

Most Ceph commands are available in the `bin/` directory. For example:

```
1 ./bin/rbd create foo --size 1000
2 ./bin/rados -p foo bench 30 write
```

To shut down the test cluster, run the following command from the `build/` directory:

```
1 ../src/stop.sh
```

Use the `sysvinit` script to start or stop individual daemons:

```
1 ./bin/init-ceph restart osd.0
2 ./bin/init-ceph stop
```

Running unit tests

To build and run all tests (in parallel using all processors), use `ctest`:

```
1 cd build
2 ninja
3 ctest -j$(nproc)
```

(Note: Many targets built from `src/test` are not run using `ctest`. Targets starting with “unittest” are run in `ninja check` and thus can be run with `ctest`. Targets starting with “ceph_test” can not, and should be run by hand.)

When failures occur, look in `build/Testing/Temporary` for logs.

To build and run all tests and their dependencies without other unnecessary targets in Ceph:

```
1 cd build
2 ninja check -j$(nproc)
```

To run an individual test manually, run `ctest` with `-R` (regex matching):

```
1 ctest -R [regex matching test name(s)]
```

(Note: `ctest` does not build the test it's running or the dependencies needed to run it)

To run an individual test manually and see all the tests output, run `ctest` with the `-V` (verbose) flag:

```
1 ctest -V -R [regex matching test name(s)]
```

To run tests manually and run the jobs in parallel, run `ctest` with the `-j` flag:

```
1 ctest -j [number of jobs]
```

There are many other flags you can give `ctest` for better control over manual test execution. To view these options run:

```
1 man ctest
```

Building the Documentation

Prerequisites

The list of package dependencies for building the documentation can be found in `doc_deps.deb.txt`:

```
1 sudo apt-get install `cat doc_deps.deb.txt`
```

Building the Documentation

To build the documentation, ensure that you are in the top-level `/ceph` directory, and execute the build script. For example:

```
1 admin/build-doc
```

Reporting Issues

To report an issue and view existing issues, please visit <https://tracker.ceph.com/projects/ceph>.