DOCUMENTING CEPH

The **easiest way** to help the Ceph project is to contribute to the documentation. As the Ceph user base grows and the development pace quickens, an increasing number of people are updating the documentation and adding new information. Even small contributions like fixing spelling errors or clarifying instructions will help the Ceph project immensely.

The Ceph documentation source resides in the `ceph/doc` directory of the Ceph repository, and Python Sphinx renders the source into HTML and manpages. The http://ceph.com/docs link currenly displays the `master` branch by default, but you may view documentation for older branches (e.g., `argonaut`) or future branches (e.g., `next`) as well as work-in-progress branches by substituting `master` with the branch name you prefer.

MAKING CONTRIBUTIONS

Making a documentation contribution generally involves the same procedural sequence as making a code contribution, except that you must build documentation source instead of compiling program source. The sequence includes the following steps:

1. Get the Source
2. Select a Branch
3. Make a Change
4. Build the Source
5. Commit the Change
6. Push the Change
7. Make a Pull Request
8. Notify the Relevant Person

GET THE SOURCE

Ceph documentation lives in the Ceph repository right alongside the Ceph source code under the `ceph/doc` directory. For details on github and Ceph, see Get Involved in the Ceph Community!.

The most common way to make contributions is to use the Fork and Pull approach. You must:

1. Install git locally. For Debian/Ubuntu, execute:

   ```
   sudo apt-get install git
   ```

   For Fedora, execute:

   ```
   sudo yum install git
   ```

   For CentOS/RHEL, execute:

   ```
   sudo yum install git
   ```

2. Ensure your `.gitconfig` file has your name and email address.

   ```
   [user]
      email = {your-email-address}
      name = {your-name}
   ```

   For example:

   ```
   git config --global user.name "John Doe"
   git config --global user.email johndoe@example.com
   ```

3. Create a github account (if you don’t have one).

4. Fork the Ceph project. See https://github.com/ceph/ceph.

5. Clone your fork of the Ceph project to your local host.

Ceph organizes documentation into an information architecture primarily by its main components.

- **Ceph Storage Cluster:** The Ceph Storage Cluster documentation resides under the `doc/rados` directory.
- **Ceph Block Device:** The Ceph Block Device documentation resides under the `doc/rbd` directory.
- **Ceph Object Storage:** The Ceph Object Storage documentation resides under the `doc/radosgw` directory.
- **Ceph Filesystem:** The Ceph Filesystem documentation resides under the `doc/cephfs` directory.
- **Installation (Quick):** Quick start documentation resides under the `doc/start` directory.
- **Installation (Manual):** Manual installation documentation resides under the `doc/install` directory.
- **Manpage:** Manpage source resides under the `doc/man` directory.
- **Developer:** Developer documentation resides under the `doc/dev` directory.
- **Images:** If you include images such as JPEG or PNG files, you should store them under the `doc/images` directory.

SELECT A BRANCH

When you make small changes to the documentation, such as fixing typographical errors or clarifying explanations, use the `master` branch (default). You should also use the `master` branch when making contributions to features that are in the current release. `master` is the most commonly used branch.

```
git checkout master
```

When you make changes to documentation that affect an upcoming release, use the `next` branch. `next` is the second most commonly used branch.

```
git checkout next
```

When you are making substantial contributions such as new features that are not yet in the current release; if your contribution is related to an issue with a tracker ID; or, if you want to see your documentation rendered on the Ceph.com website before it gets merged into the `master` branch, you should create a branch. To distinguish branches that include only documentation updates, we prepend them with `wip-doc` by convention, following the form `wip-doc-{your-branch-name}`. If the branch relates to an issue filed in http://tracker.ceph.com/issues, the branch name incorporates the issue number. For example, if a documentation branch is a fix for issue #4000, the branch name should be `wip-doc-4000` by convention and the relevant tracker URL will be http://tracker.ceph.com/issues/4000.

Note: Please do not mingle documentation contributions and source code contributions in a single pull request. Editors review the documentation and engineers review source code changes. When you keep documentation pull requests separate from source code pull requests, it simplifies the process and we won't have to ask you to resubmit the requests separately.

Before you create your branch name, ensure that it doesn't already exist in the local or remote repository.

```
git branch -a | grep wip-doc-{your-branch-name}
```

If it doesn't exist, create your branch:

```
git checkout -b wip-doc-{your-branch-name}
```

MAKE A CHANGE

Modifying a document involves opening a restructuredText file, changing its contents, and saving the changes. See Documentation Style Guide for details on syntax requirements.

Adding a document involves creating a new restructuredText file under the `doc` directory or its subdirectories and saving the file with a `*.rst` file extension. You must also include a reference to the document: a hyperlink or a table of contents entry. The `index.rst` file of a top-level directory usually contains a TOC, where you can add the new file name. All documents must have a title. See Headings for details.

Your new document doesn't get tracked by `git` automatically. When you want to add the document to the repository, you must use `git add {path-to-filename}`. For example, from the top level directory of the repository, adding an `example.rst` file to the `rados` subdirectory would look like this:

```
git add doc/rados/example.rst
```

Deleting a document involves removing it from the repository with `git rm {path-to-filename}`. For example:

```
git rm doc/rados/example.rst
```

You must also remove any reference to a deleted document from other documents.

BUILD THE SOURCE

To build the documentation, navigate to the ceph repository directory:

```
cd ceph
```

To build the documentation on Debian/Ubuntu, Fedora, or CentOS/RHEL, execute:

```
admin/build-doc
```

To scan for the reachablity of external links, execute:

```
admin/build-doc linkcheck
```

Executing `admin/build-doc` will create a `build-doc` directory under ceph. You may need to create a directory under `ceph/build-doc` for output of Javadoc files.

```
mkdir -p output/html/api/libcephfs-java/javadoc
```

The build script `build-doc` will produce an output of errors and warnings. You MUST fix errors in documents you modified before committing a change, and you SHOULD fix warnings that are related to syntax you modified.

Important: You must validate ALL HYPERLINKS. If a hyperlink is broken, it automatically breaks the build!

Once you build the documentation set, you may navigate to the source directory to view it:

```
cd build-doc/output
```

There should be an `html` directory and a `man` directory containing documentation in HTML and manpage formats respectively.

BUILD THE SOURCE (FIRST TIME)

Ceph uses Python Sphinx, which is generally distribution agnostic. The first time you build Ceph documentation, it will generate a doxygen XML tree, which is a bit time consuming.

Python Sphinx does have some dependencies that vary across distributions. The first time you build the documentation, the script will notify you if you do not have the dependencies installed. To run Sphinx and build documentation successfully, the following packages are required:

DEBIAN/UBUNTU

- gcc
- python-dev
- python-pip
- python-virtualenv
- python-sphinx
- libxml2-dev
- libxslt1-dev

FEDORA

- gcc
- python-devel
- python-pip
- python-virtualenv
- python-docutils
- python-jinja2
- python-pygments

CENTOS/RHEL

- gcc
- python-devel
- python-pip
- python-virtualenv
- python-docutils
- python-jinja2
- python-pygments

- doxygen
- graphviz
- ant
- ditaa

- python-sphinx
- libxml2-devel
- libxslt1-devel
- doxygen
- graphviz
- ant
- ditaa

- python-sphinx
- libxml2-dev
- libxslt1-dev
- doxygen
- graphviz
- ant

Install each dependency that is not installed on your host. For Debian/Ubuntu distributions, execute the following:

```
sudo apt-get install gcc python-dev python-pip python-virtualenv libxml2-dev libxslt-dev doxy
sudo apt-get install python-sphinx
```

For Fedora distributions, execute the following:

```
sudo yum install gcc python-devel python-pip python-virtualenv libxml2-devel libxslt-devel do
sudo pip install html2text
sudo yum install python-jinja2 python-pygments python-docutils python-sphinx
sudo yum install jericho-html ditaa
```

For CentOS/RHEL distributions, it is recommended to have `epel` (Extra Packages for Enterprise Linux) repository as it provides some extra packages which are not available in the default repository. To install `epel`, execute the following:

```
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

For CentOS/RHEL distributions, execute the following:

```
sudo yum install gcc python-devel python-pip python-virtualenv libxml2-devel libxslt-devel do
sudo pip install html2text
```

For CentOS/RHEL distributions, the remaining python packages are not available in the default and `epel` repositories. So, use http://rpmfind.net/ to find the packages. Then, download them from a mirror and install them. For example:

```
wget http://rpmfind.net/linux/centos/7/os/x86_64/Packages/python-jinja2-2.7.2-2.el7.noarch.rp
sudo yum install python-jinja2-2.7.2-2.el7.noarch.rpm
wget http://rpmfind.net/linux/centos/7/os/x86_64/Packages/python-pygments-1.4-9.el7.noarch.rp
sudo yum install python-pygments-1.4-9.el7.noarch.rpm
wget http://rpmfind.net/linux/centos/7/os/x86_64/Packages/python-docutils-0.11-0.2.20130715sv
sudo yum install python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm
wget http://rpmfind.net/linux/centos/7/os/x86_64/Packages/python-sphinx-1.1.3-11.el7.noarch.r
sudo yum install python-sphinx-1.1.3-11.el7.noarch.rpm
```

Ceph documentation makes extensive use of ditaa, which is not presently built for CentOS/RHEL7. You must install `ditaa` if you are making changes to `ditaa` diagrams so that you can verify that they render properly before you commit new or modified `ditaa` diagrams. You may retrieve compatible required packages for CentOS/RHEL distributions and install them manually. To run `ditaa` on CentOS/RHEL7, following dependencies are required:

- jericho-html
- jai-imageio-core
- batik

Use http://rpmfind.net/ to find compatible `ditaa` and the dependencies. Then, download them from a mirror and install them. For example:

```
wget http://rpmfind.net/linux/fedora/linux/releases/22/Everything/x86_64/os/Packages/j/jerich
sudo yum install jericho-html-3.3-4.fc22.noarch.rpm
wget http://rpmfind.net/linux/centos/7/os/x86_64/Packages/jai-imageio-core-1.2-0.14.20100217c
sudo yum install jai-imageio-core-1.2-0.14.20100217cvs.el7.noarch.rpm
```

```
wget http://rpmfind.net/linux/centos/7/os/x86_64/Packages/batik-1.8-0.12.svn1230816.el7.noarc
sudo yum install batik-1.8-0.12.svn1230816.el7.noarch.rpm
wget http://rpmfind.net/linux/fedora/linux/releases/22/Everything/x86_64/os/Packages/d/ditaa-
sudo yum install ditaa-0.9-13.r74.fc21.noarch.rpm
```

Once you have installed all these packages, build the documentation by following the steps given in Build the Source.

COMMIT THE CHANGE

Ceph documentation commits are simple, but follow a strict convention:

- A commit SHOULD have 1 file per commit (it simplifies rollback). You MAY commit multiple files with related changes. Unrelated changes SHOULD NOT be put into the same commit.
- A commit MUST have a comment.
- A commit comment MUST be prepended with `doc:`. (strict)
- The comment summary MUST be one line only. (strict)
- Additional comments MAY follow a blank line after the summary, but should be terse.
- A commit MAY include `Fixes: #{bug number}`.
- Commits MUST include `Signed-off-by: Firstname Lastname <email>`. (strict)

Tip: Follow the foregoing convention particularly where it says `(strict)` or you will be asked to modify your commit to comply with this convention.

The following is a common commit comment (preferred):

```
doc: Fixes a spelling error and a broken hyperlink.

Signed-off-by: John Doe <john.doe@gmail.com>
```

The following comment includes a reference to a bug.

```
doc: Fixes a spelling error and a broken hyperlink.

Fixes: #1234

Signed-off-by: John Doe <john.doe@gmail.com>
```

The following comment includes a terse sentence following the comment summary. There is a carriage return between the summary line and the description:

```
doc: Added mon setting to monitor config reference

Describes 'mon setting', which is a new setting added
to config_opts.h.

Signed-off-by: John Doe <john.doe@gmail.com>
```

To commit changes, execute the following:

```
git commit -a
```

An easy way to manage your documentation commits is to use visual tools for `git`. For example, `gitk` provides a graphical interface for viewing the repository history, and `git-gui` provides a graphical interface for viewing your uncommitted changes, staging them for commit, committing the changes and pushing them to your forked Ceph repository.

For Debian/Ubuntu, execute:

```
sudo apt-get install gitk git-gui
```

For Fedora/CentOS/RHEL, execute:

```
sudo yum install gitk git-gui
```

Then, execute:

```
cd {git-ceph-repo-path}
gitk
```

Finally, select **File->Start git gui** to activate the graphical user interface.

PUSH THE CHANGE

Once you have one or more commits, you must push them from the local copy of the repository to `github`. A graphical tool like `git-gui` provides a user interface for pushing to the repository. If you created a branch previously:

```
git push origin wip-doc-{your-branch-name}
```

Otherwise:

```
git push
```

MAKE A PULL REQUEST

As noted earlier, you can make documentation contributions using the Fork and Pull approach.

NOTIFY THE RELEVANT PERSON

After you make a pull request, notify the relevant person. For general documentation pull requests, notify John Wilkins.

DOCUMENTATION STYLE GUIDE

One objective of the Ceph documentation project is to ensure the readability of the documentation in both native restructuredText format and its rendered formats such as HTML. Navigate to your Ceph repository and view a document in its native format. You may notice that it is generally as legible in a terminal as it is in its rendered HTML format. Additionally, you may also notice that diagrams in `ditaa` format also render reasonably well in text mode.

```
less doc/architecture.rst
```

Review the following style guides to maintain this consistency.

HEADINGS

1. **Document Titles:** Document titles use the = character overline and underline with a leading and trailing space on the title text line. See Document Title for details.
2. **Section Titles:** Section tiles use the = character underline with no leading or trailing spaces for text. Two carriage returns should precede a section title (unless an inline reference precedes it). See Sections for details.
3. **Subsection Titles:** Subsection titles use the _ character underline with no leading or trailing spaces for text. Two carriage returns should precede a subsection title (unless an inline reference precedes it).

TEXT BODY

As a general rule, we prefer text to wrap at column 80 so that it is legible in a command line interface without leading or trailing white space. Where possible, we prefer to maintain this convention with text, lists, literal text (exceptions allowed), tables, and `ditaa` graphics.

1. **Paragraphs**: Paragraphs have a leading and a trailing carriage return, and should be 80 characters wide or less so that the documentation can be read in native format in a command line terminal.

2. **Literal Text:** To create an example of literal text (e.g., command line usage), terminate the preceding paragraph with `::` or enter a carriage return to create an empty line after the preceding paragraph; then, enter `::` on a separate line

followed by another empty line. Then, begin the literal text with tab indentation (preferred) or space indentation of 3 characters.

3. **Indented Text:** Indented text such as bullet points (e.g., `- some text`) may span multiple lines. The text of subsequent lines should begin at the same character position as the text of the indented text (less numbers, bullets, etc.).

   Indented text may include literal text examples. Whereas, text indentation should be done with spaces, literal text examples should be indented with tabs. This convention enables you to add an additional indented paragraph following a literal example by leaving a blank line and beginning the subsequent paragraph with space indentation.

4. **Numbered Lists:** Numbered lists should use autonumbering by starting a numbered indent with `#.` instead of the actual number so that numbered paragraphs can be repositioned without requiring manual renumbering.

5. **Code Examples:** Ceph supports the use of the `.. code-block::<language>` role, so that you can add highlighting to source examples. This is preferred for source code. However, use of this tag will cause autonumbering to restart at 1 if it is used as an example within a numbered list. See Showing code examples for details.

PARAGRAPH LEVEL MARKUP

The Ceph project uses paragraph level markup to highlight points.

1. **Tip:** Use the `.. tip::` directive to provide additional information that assists the reader or steers the reader away from trouble.
2. **Note**: Use the `.. note::` directive to highlight an important point.
3. **Important:** Use the `.. important::` directive to highlight important requirements or caveats (e.g., anything that could lead to data loss). Use this directive sparingly, because it renders in red.
4. **Version Added:** Use the `.. versionadded::` directive for new features or configuration settings so that users know the minimum release for using a feature.
5. **Version Changed:** Use the `.. versionchanged::` directive for changes in usage or configuration settings.
6. **Deprecated:** Use the `.. deprecated::` directive when CLI usage, a feature or a configuration setting is no longer preferred or will be discontinued.
7. **Topic:** Use the `.. topic::` directive to encapsulate text that is outside the main flow of the document. See the topic directive for additional details.

TOC AND HYPERLINKS

All documents must be linked from another document or a table of contents, otherwise you will receive a warning when building the documentation.

The Ceph project uses the `.. toctree::` directive. See The TOC tree for details. When rendering a TOC, consider specifying the `:maxdepth:` parameter so the rendered TOC is reasonably terse.

Document authors should prefer to use the `:ref:` syntax where a link target contains a specific unique identifier (e.g., `.. _unique-target-id:`), and a reference to the target specifically references the target (e.g., ``:ref:`unique-target-id` ``) so that if source files are moved or the information architecture changes, the links will still work. See Cross referencing arbitrary locations for details.

Ceph documentation also uses the backtick (accent grave) character followed by the link text, another backtick and an underscore. Sphinx allows you to incorporate the link destination inline; however, we prefer to use the use the `.. _Link Text: ../path` convention at the bottom of the document, because it improves the readability of the document in a command line interface.