Commit Comments

Canonical version of this page is at https://wiki.lustre.org/Commit_Comments

Contents:

Recent edits:

User	Edits	Last Update
Andreas Dilger	23	771 days ago
Michael Kvardakov	1	1162 days ago

Commit message content

Writing good commit comments is critical to ensuring that changes are easily understood, even years after they were originally written. The commit comment should contain enough information about the change to allow the reader to understand the motivation for the change, what parts of the code it is affecting, and any interesting, unusual, or complex parts of the change to draw attention to.

The reason for a change may be manyfold: bug, enhancement, feature, code style, etc. so providing information about this sets the stage for understanding the change. If it is a bug, include information about what usage triggers the bug and how it manifests (error messages, LBUG, etc). If it is a feature, include information about what improvement is being made, and how it will affect usage.

Providing some high-level information about the code path that is being modified is useful for the reader, since the files and patch fragments are not necessarily going to be listed in a sensible order in the patch. Including the important functions being modified provides a starting point for the reader to follow the logic of the change, and makes it easier to search for such changes in the future.

If the patch is based on some earlier patch, then including the git commit hash of the original patch, Jira ticket number, etc. is useful for tracking the chain of dependencies. This can be very useful if a patch is landed separately to different maintenance branches, if it is fixing a problem in a previously landed patch, or if it is being imported from an upstream kernel commit.

Having long commit comments that describe the change well is a good thing. The commit comments will be tightly associated with the code for a long time into the future, even many of the original commit comments from years earlier are still available through changes of the source code repository. In contrast, bug tracking systems come and go, and cannot be relied upon to track information about a change for extended periods of time.

Commit message format

Unlike the content of the commit message, the format is relatively easy to verify for correctness. Having the same standard format allows Git tools like `git shortlog` to extract information from the patches more easily.

The first line of the commit comment is the commit summary of the change. Changes submitted to the `fs/lustre-release` branch require a Lustre Jira ticket number at the beginning of the commit summary. A Lustre Jira ticket is one that begins with LU and is therefore part of the Lustre project within Jira. For patches to other projects, such as documentation, a different JIRA project should be used (e.g. `LUDOC`). If the patch is submitted for the `fs/lustre-release` repository without a Lustre Jira ID in the first line, then it will automatically receive a -2 review which will prevent the patch from being submitted to a release branch. You would then need to fix the summary line and resubmit the patch.

The commit summary should also have a `component:` tag immediately following the Jira ticket number that indicates which Lustre subsystem that the commit is related to. Example Lustre subsystems relate to modules like: `llite, lov, osc, mdc, lmv, ldlm, ptlrpc, mds, oss, mdd, osd-ldiskfs, osd-zfs, ldiskfs, lnet, libcfs, socklnd, o2iblnd`; functional components like `recovery, quota, grant`; or auxiliary components like `build, tests, iokit, docs`. This subsystem list is not exhaustive, but provides a good guideline for consistency.

The commit summary line must be 62 characters or less, including the Jira ticket number and component tag, so that `git shortlog` and `git format-patch` can fit the summary onto a single line. The summary must be followed by a blank like. The rest of the comments should be wrapped to 70 columns or less. This allows for the first line to be used a subject in emails, and also for the entire body to be displayed using tools like `git log` or `git shortlog` in an 80 column window.

```
LU-nnn component: short description of change under 62 columns

The "component:" should be a lower-case single-word subsystem of the
Lustre code that best encompasses the change being made.  Examples of
components include modules: llite, lov, lmv, osc, mdc, ldlm, mds, oss,
ptlrpc, osd-ldiskfs, osd-zfs, ldiskfs, lnet, socklnd, o2iblnd, libcfs;
functional subsystems: recovery, quota, grant; and auxiliary areas:
build, tests, docs.  This list is not exhaustive, but is a guideline.

The commit comment should contain a detailed explanation of changes
being made.  This can be as long as you'd like.  Please give details
of what problem was solved (including error messages or problems that
were seen), a good high-level description of how it was solved, and
which parts of the code were changed (including important functions
that were changed, if this is useful to understand the patch, and
for easier searching).  Wrap lines at/under 70 columns.

Signed-off-by: Your Real Name <your_email@domain.name>
Change-Id: Ixxxx(added automatically if missing)xxxx
```

The Signed-off-by: line

The `Signed-off-by:` line asserts that you have permission to contribute the code to the project according to the Developer's Certificate of Origin.

The Change-Id: line and Code Style

Gerrit needs to automatically identify updates to existing patches and update the existing change instead of creating a new one for each patch submitted. This preserves the history of patch comments, and allows comparing old and new versions of a patch for more efficient inspections. For this to work properly the changes you create locally need to have a unique commit id in them. The same `Change-Id:` line should be used for all versions of a patch.

Please make sure the `Change-Id:` line and `Signed-off-by:` lines are at the bottom of the comment. This is required by the `fs/lustre-release` and other projects in order for the patch to be approved. If your comments are missing either of these lines the patch will be automatically be rejected at submission time.

The `Change-Id:` field is inserted automatically into the commit message by installing the `contrib/git-hooks/commit-msg` hook into each Git repository's `.git/hooks/` directory. The `contrib/git-hooks/prepare-commit-msg` script should be installed as well. From the top-level Lustre tree checkout:

```
ln -sf ../../contrib/git-hooks/commit-msg .git/hooks/
ln -sf ../../contrib/git-hooks/prepare-commit-msg .git/hooks/
```

At `git commit` time this will run the `prepare-commit-msg` script from the currently Lustre tree to check the code changes for style errors using `contrib/checkpatch.pl`, and add comments at the end of the commit message with any warnings or errors. After the commit message has been saved, the `commit-msg` script will verify that the commit message format matches the format specified above, and insert the `Change-Id:` field into the commit message, if one isn't already present.

Additional commit tags

A number of additional commit tags can be used to further explain who has contributed to the patch, and for tracking purposes. These tags are commonly used with Linux kernel patches. These tags should appear before the `Signed-off-by:` tag.

```
Acked-by: User Name <user@domain.com>
Tested-by: User Name <user@domain.com>
Reported-by: User Name <user@domain.com>
Reviewed-by: User Name <user@domain.com>
Test-Parameters: additional testing parameters
CC: User Name <user@domain.com>
{Organization}-bug-id: arbitrary bug tracking identifier
Lustre-commit: {git commit hash of original patch}
Lustre-change: {Gerrit change URL of original patch}
Linux-commit: {git commit hash of upstream kernel patch}
```

The `{Organization}-bug-id:` tag (e.g. `Intel-bug-id: ORI-123`, `Xyratex-bug-id: MRP-123`, or `Oracle-bug-id: b=12345`) can be used to track this patch in other bug databases.

`Lustre-commit:` and `Lustre-change:` are used to reference the original version of a patch that is being ported to another branch.

Patches pushed ***to*** the upstream kernel should include `Intel-bug-id:` to reference the JIRA URL for the LU ticket, in the form `https://jira.hpdd.intel.com/browse/LU-nnnn`. Patches should also include the `Reviewed-on:` permalink URL for the Gerrit patch, in the form `https://review.whamcloud.com/nnnnn`. The upstream maintainers do *not* want to have commit hashes for non-kernel commits, so upstream patches should ***not*** include the `Lustre-commit:` tag.

Patches ported ***from*** the upstream kernel, the patch should keep the original commit summary line, with slight modification to conform to Lustre standards: include the JIRA LU ticket number, and replace "staging" and other pathnames in the summary with a subsystem. The `Linux-commit:` tag is used to hold the upstream kernel commit hash. The original author should be kept as the author of the patch using the `git --author` option, as well as the original `Signed-off-by:` tag(s).

`Test-Parameters:` is used to specify additional testing parameters for this patch, see Changing Test Parameters with Gerrit Commit Messages.

Examples of good commit comments

```
LU-477 ldiskfs: allocate s_group_desc/s_group_info by vmalloc()

Add the patch to the RHEL6 ldiskfs patch series.

Large kmalloc() for sbi->s_group_desc and sbi->s_group_info can fail
for large filesystems (typically over 16TB), which will cause the
"not enough memory" error while mounting.  This patch makes allocation
fall back to vmalloc() if the kmalloc() failed, as what was done for
sbi->s_flex_groups.

To avoid colliding with an valid on-disk inode number, EXT4_BAD_INO
is used as the number of the buddy cache inode.

The patch also incorporates the following upstream kernel fix:

  Linux-commit: 32a9bb57d7c1fd04ae0f72b8f671501f000a0e9f
  ext4: fix missing iput() of root inode for some mount error paths
  https://bugzilla.kernel.org/show_bug.cgi?id=26752

Signed-off-by: Yu Jian <yujian@whamcloud.com>
Change-Id: I3950425835ea7f2968ceb2edbc622e3ff3ed8545
```

```
LU-723 build: Enhance lustre/ldiskfs build system


Enhance the lustre/ldiskfs build system so it is more robust, flexible,
and consistent with lustre/zfs build system. This change is being made
in the interest of standardizing the infrastructure around backend
filesystems.

This change does not effect the current behavior of the --with-ldiskfs,
--enable-ext4, or --with-ldiskfsprogs configure options. However, it
does remove the obsolete --with-ldiskfs-inkernel configure option which
was only used by LLNL. It also adds the --with-ldiskfs-obj configure
option which improves flexibility. And the --enable-ldiskfs-build
configure option to support building against the lustre-ldiskfs-devel
package. The behavior of these options is consistent with their ZFS
counterparts, see commit 8c7266c for further details.

--enable-ext4 enable ldiskfs build using ext4
--enable-ldiskfs-build enable ldiskfs configure/make
--with-ldiskfs=path set path to ldiskfs source
--with-ldiskfs-obj=path set path to ldiskfs objects
--with-ldiskfsprogs use alternate names for ldiskfs-enabled e2fsprogs

Sample ./configure results when building lustre and ldiskfs using
the kernel-devel and kernel-debuginfo-common packages.

checking whether to enable ldiskfs... yes
checking ldiskfs source directory... /home/behlendo/src/git/lustre/ldiskfs
checking ldiskfs object directory... /home/behlendo/src/git/lustre/ldiskfs
checking ldiskfs module symbols... Module.symvers
checking ldiskfs source release... 3.3.0
checking whether to use ext3 or ext4 source... ext4
checking ext4 source directory... /usr/src/debug/.../fs/ext4
checking whether to build ldiskfs... yes
checking for /home/behlendo/src/git/lustre/ldiskfs/configure... yes
checking for /usr/src/debug/.../fs/ext4/dir.c... yes
checking for /usr/src/debug/.../fs/ext4/file.c... yes
checking for /usr/src/debug/.../fs/ext4/inode.c... yes
checking for /usr/src/debug/.../fs/ext4/super.c... yes
checking if ext4_ext_walk_space() takes i_data_sem... yes
checking if LDISKFS_SINGLEDATA_TRANS_BLOCKS takes sb as argument... yes
checking if ldiskfs_discard_preallocations defined... yes
checking if ldiskfs_ext_insert_extent needs 5 arguments... yes

In the context of this change additional cleanup has been done and
all of the ldiskfs specific code relocated to lustre-build-ldiskfs.m4.

Note that this change moves us closer to supporting patchless Lustre
servers with ldiskfs. Once the remaining kernel patches for Lustre
are dropped you will be able to build Lustre using the distribution
provided kernel-devel and kernel-debuginfo-common packages.

This change also incorporates ORI-340 commit f604951, which ensures
that the Module.symvers file will cleanly include the symbols for all
enabled Lustre backends. While the only backend supported by master
right now is ldiskfs this brings the master and orion branches into
sync in this regard.

Change-Id: I6f13f266944ec6967f4d7705a30b83ab8e577b15
Signed-off-by: Brian Behlendorf <behlendorf1@llnl.gov>
Signed-off-by: Prakash Surya <surya1@llnl.gov>
```

Patch porting examples

For patches backported from master to a Lustre maintenance branch (e.g. `b2_10`) there are some conventions to follow so that the changes/fixes can more easily be tracked across branches. The simples method for porting a patch from one branch to another is to use the `"Cherry Pick"` button on the patch directly in Gerrit, which can be used for patches that apply cleanly to the specified target branch as long as it is in the same Git repository. Alternately, if this is not possible, use `git cherry-pick {commit_hash|branch}` from the command line to pull the patch onto the (current) branch where you want the patch to land, and then using the normal patch submission process to push the patch to Gerrit or submit it to the upstream kernel. This will apply the whole patch (as best as is able, and show conflicts where needed), copy the commit message, preserve the original patch author. With luck, there will not be any patch conflicts and no further work is needed. If necessary, the patch conflicts need to be resolved before committing the patch. For the commit message:

- the entire commit message, including the summary line, should be copied from the source patch without any changes
- the original `Author` of the patch should be kept. This should be done automatically when using `git cherry-pick` but is lost when applying the patch manually, so `git commit --author="Original Author <author@email.com>"` should be used
- the `Signed-off-by:` line of the original author should also be kept
- the person who commits the ported change should add a `Signed-off-by:` line with their name and email following the original one
- the original `Change-Id:` line
- the original `Reviewed-by:` lines can be kept, and Gerrit will automatically add them as reviewers to the new patch
- the `Tested-by: Maloo` and `Tested-by: Jenkins` lines should be removed from the new commit message, though any `Tested-by:` lines from real people can be kept
- the `"Reviewed-on: http://review.whamcloud.com/nnnnn"` line should be changed to `"Lustre-change: http://review.whamcloud.com/nnnnn"` (please use the "permalink" Gerrit URL format as shown)
- the `"cherry picked from commit abcdef1234567890"` line should be changed to `"Lustre-commit: abcdef1234567890"`
- any non-trivial changes to the original patch should be described *after* the original `Signed-off-by:` and `Lustre-commit:` lines before your own `Signed-off-by:` line
- if two or more patches are being combined into a single patch (normally only when there are bugs in the original patch that were fixed in later commits) the full commit message from each patch should be kept, along with the `Signed-off-by:` and `Lustre-commit:` and `Lustre-change:` lines of each commit

Patches ported from master to maintenance branch

For example, porting a patch from master to `b2_10` or similar:

```
LU-4725 mdt: child-parent lock ordering in rename

Change rename so that it always has parent-child lock ordering,
otherwise it may deadlock with other operations.

Lustre-commit: 4e308ef74f271ec7e19917e3c0f88586537582c3
Lustre-change: http://review.whamcloud.com/9538

LU-4725 obd: lu_object_find_at hung

lu_object_find_at hangs if called two times and object is removed
in between. It makes mdt_rename_sanity not workable for rename.
Change mdt_rename_sanity to work on existing object.

Lustre-commit: b7c72ec1ddeda2cf94ea151f974d3f94e3a7d1ed
Lustre-change: http://review.whamcloud.com/10484
Xyratex-bug-id: MRP-1700

Test-Parameters: alwaysuploadlogs \
  envdefinitions=SLOW=yes,ENABLE_QUOTA=yes,ONLY=54 \
  ossjob=lustre-b2_4 mdsjob=lustre-b2_4 ossbuildno=73 mdsbuildno=73 \
  testlist=sanityn

Signed-off-by: Vitaly Fertman <vitaly_fertman@xyratex.com>
Signed-off-by: Rahul Deshmukh <rahul_deshmukh@xyratex.com>
Change-Id: Ic9ce52bfcd8788834347fba155cc8c6be674dcd8
```

Patches ported from master to upstream kernel

These are treated similarly as patches ported to maintenance branches (keep all comments and `Signed-off-by:` lines from the original patch) add new `Signed-off-by:` and comments afterward, but replace the `Lustre-commit:` line (which doesn't mean anything in the upstream kernel git) with `Intel-bug-id: {jira_URL}` so that the original bug can still be identified. When submitting patches upstream, please also follow the `Documentation/process/submitting-patches.rst` instructions, and use `scripts/get_maintainer.pl` to generate the CC list for the patch. If in doubt, submit the patch only to `lustre-devel` first to get feedback from the Lustre maintainers.

```
lustre/llite: simplify dentry revalidate

Lustre client dentry validation is protected by LDLM lock, so
any time a dentry is found, it's valid and no need to revalidate
from MDS, and even it does, there is race that it may be
invalidated after revalidation is finished.

Reviewed-on: http://review.whamcloud.com/7475
Intel-bug-id: https://jira.hpdd.intel.com/browse/LU-3544
Signed-off-by: Lai Siyao <lai.siyao@intel.com>
Signed-off-by: Oleg Drokin <oleg.drokin@intel.com>
```