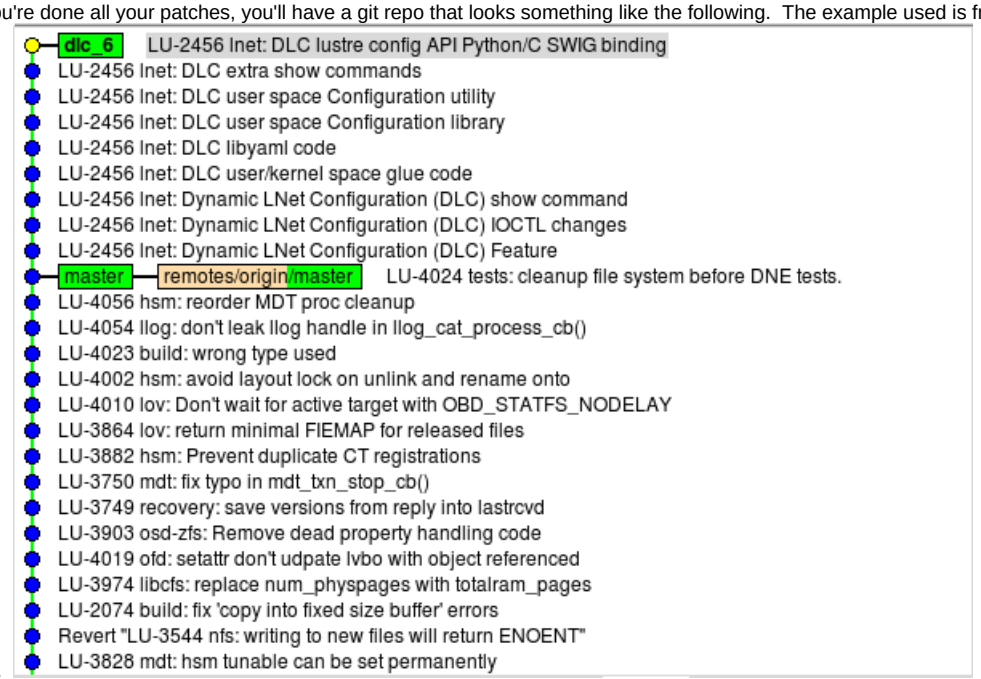
Recommended Workflow

Below is a recommended workflow when committing multiple related patches into Gerrit. This workflow is based on some extensive collaboration with Robert Read.

Initial Push into Gerrit

1. Create a commit plan. If you're making a lot of changes to the code, decide how you plan to break these changes into patches.
2. Create your working branch off of master
    a. git checkout -b <working branch>
3. Make the changes according to your plan, and as you finish each patch commit into your working branch.
    a. git add <files>
    b. git commit -av
        i. Follow the guidelines for comments, etc, as already documented.
4. After you're done all your patches, you'll have a git repo that looks something like the following. The example used is from the DLC project.
    a. 

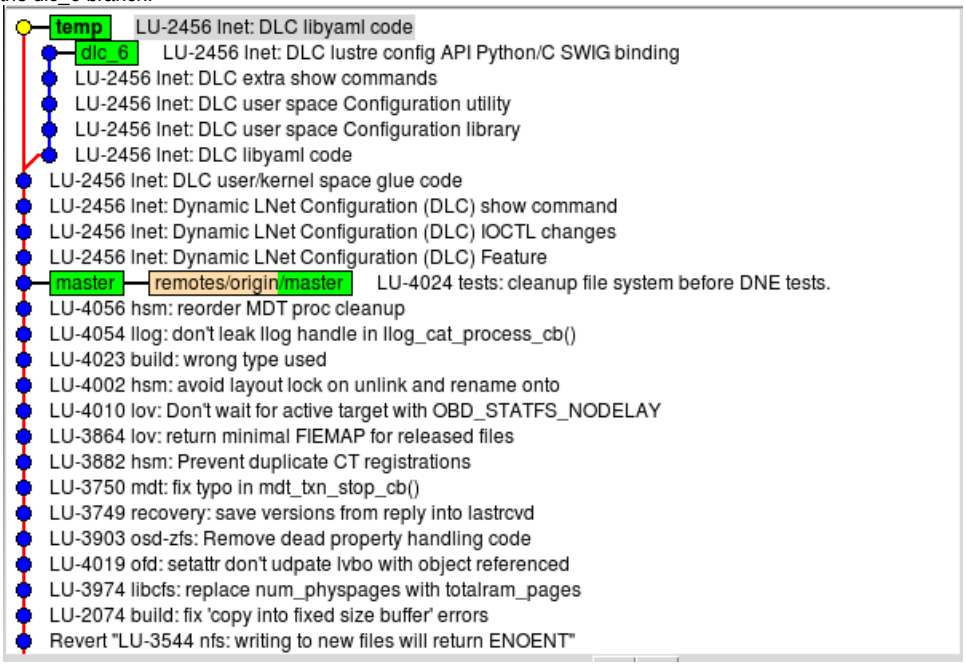
        i. You'll notice that there are nine commits on the branch dlc_6, which is based off master
5. Make sure that every patch by itself can be compiled, run and pass autotest (although you won't know the latter until you push into gerrit, unless you're able to run the autotest on your local machines)
6. Now you can push into gerrit.
    a. git push ssh://<username>@review.whamcloud.com:29418/fs/lustre-release HEAD:refs/for/master
    b. Gerrit will recognize each individual commit and create a review page for each commit.

Addressing Inspection Comments

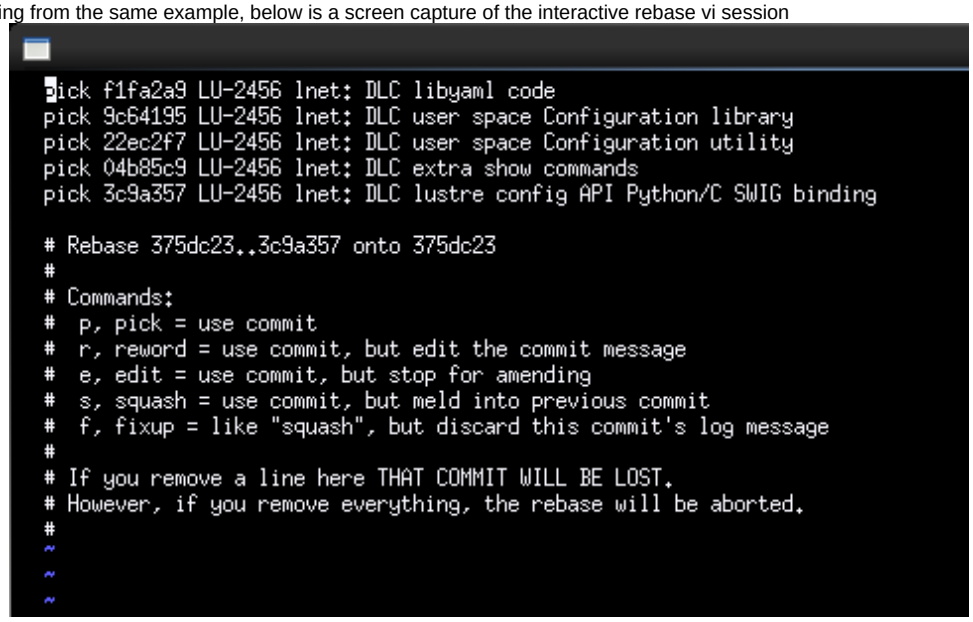
Now that you submitted your code into gerrit, you'll most likely start getting comments on the different patches. You'll need to update the separate patches and probably rebase all the patches which depend on the patch you have changed. Below is a work flow to aid you in this process.

1. Create a branch off the commit that you want to change
    a. git checkout -b <temporary branch> <commit-id>
2. Now that you're on the <temporary branch>, start addressing the inspection comments
3. git status
    a. Will show the changes you've made
4. Now commit and rebase your changes on that temporary branch, so you don't have multiple commits.
    a. git add <files>
    b. git commit --amend -av
        i. Follow the guidelines for the commit message
        ii. This will commit and merge your changes with the original commit
    c. Alternatively, a more manual way of achieving the "git commit --amend -av" is as following
        i. git commit -av
        ii. git rebase -i HEAD~2
            1. This will show you the previous commit and the your recent commit to address the inspection comment
            2. Change the commit for the inspection changes from "pick" to "fixup"
                a. This will collapse the inspection commit into the original commit
5. Compile and make sure your fixes are good.
    a. At this point the git repo will have the temp branch. Using the same example above, I needed to edit the commit: LU-2456 lnet: DLC libyaml code. So I created a branch based on that commit, made the changes and committed on the temp branch. Notice that now the

same commit exist on both the temp and dlc_6 (working) branch. The one on the temp branch is the modified one. Next step is to pull that into the dlc_6 branch.

i. 

6. Rebase your changes from the <temporary branch> back into your main working branch
    a. git rebase -i <temporary branch> <working branch>
    b. delete the redundant commit that you're pulling in from your temporary branch
        i. basically this step is saying, I don't want to use the commit that currently exists on my working branch, I want to replace it with the commit from my temporary branch, which I have made some changes to.
            1. Note that the order git displays the commit while doing interactive rebase is from oldest to newest. you probably want to keep the same order.
            2. Continuing from the same example, below is a screen capture of the interactive rebase vi session

a. 

b. Now you need to delete the first line, so that the rebase uses the modified patch

c.

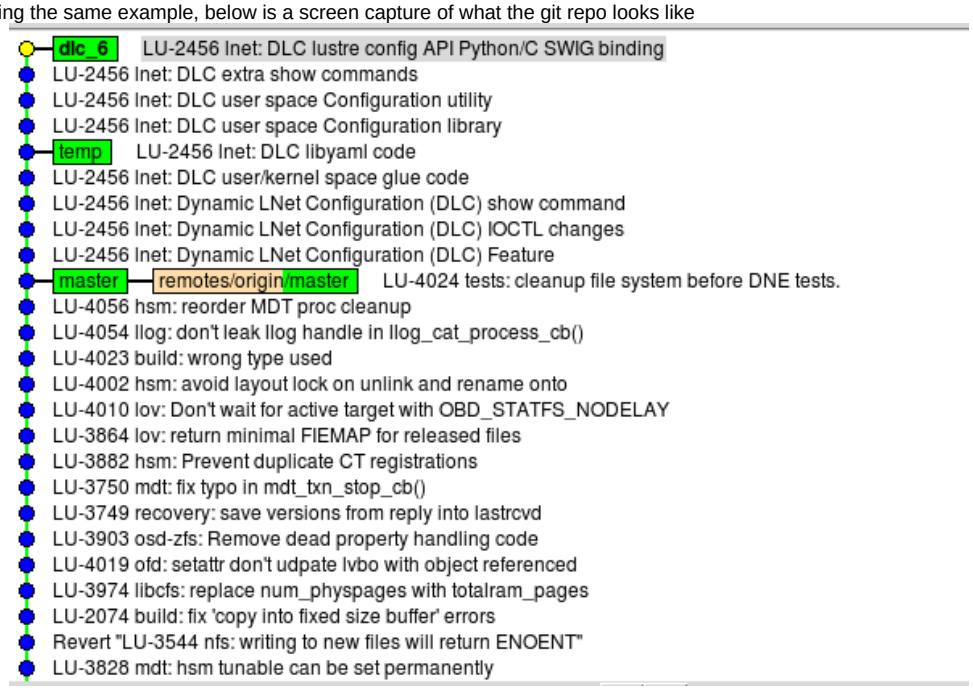
```
pick 9c64195 LU-2456 lnet: DLC user space Configuration library
pick 22ec2f7 LU-2456 lnet: DLC user space Configuration utility
pick 04b85c9 LU-2456 lnet: DLC extra show commands
pick 3c9a357 LU-2456 lnet: DLC lustre config API Python/C SWIG binding

# Rebase 375dc23..3c9a357 onto 375dc23
#
# Commands:
#  p, pick = use commit
#  r, reword = use commit, but edit the commit message
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#  f, fixup = like "squash", but discard this commit's log message
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
~
~
~
```

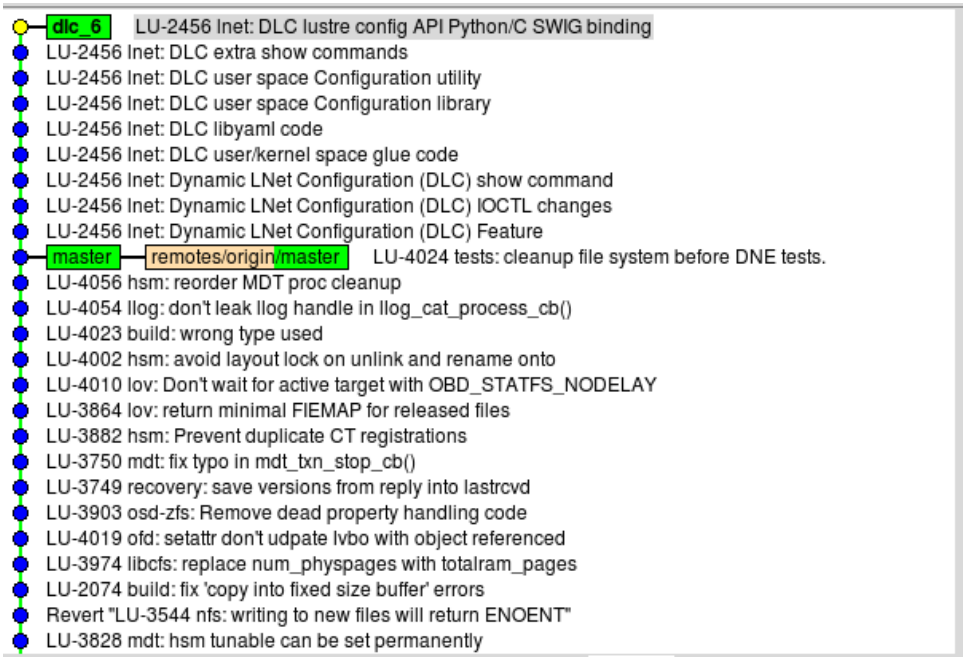
c. After this is complete, all your commits which depend on the change you just made will be rebased on your changes.

i. Continuing the same example, below is a screen capture of what the git repo looks like

1.



7. Now that you're on your working branch, delete your temporary branch
   a. git branch -d <temporary branch>
      i. Continuing the same example, below is a screen capture of what the git repo looks like. Note, it looks exactly the same as what you started with

1. 

8. From your working branch push into Gerrit
    a. git push ssh://<username>@review.whamcloud.com:29418/fs/lustre-release HEAD:refs/for/master
    b. Gerrit will recognize which patches has changed and update them appropriately.
9. Now you're ready for more inspection comments. This process can be repeated until the patches pass inspection.

Rebasing On Master

As work on the working branch goes on, there will come a time when we need to rebase on the latest and greatest.

1. On the master branch do:
    a. git pull
2. On the working branch do:
    a. git rebase master (you can use -i option if you want to edit the list of changes)

Sometimes you might have multiple commits on your working branch. Each commit gets pushed into gerrit as a separate review. When synchronizing with the tip of the remote repo, you'd want to rebase each of the commits. I do so in the following steps:

1. checkout the earliest commit into its separate temporary branch
    a. git checkout -b <temp branch> <commit-id>
2. go to master and update it to the tip of the remote repo as shown above (using 'git pull')
3. rebase the temp branch to master. On the temp branch do
    a. git rebase master (you can use -i option if you want to edit the list of changes)
4. rebase the temp branch on your working branch (as shown above).
    a. git rebase -i <temp branch> <working branch>
        i. ensure to remove the commit of the temp branch from the commit list displayed by rebase.