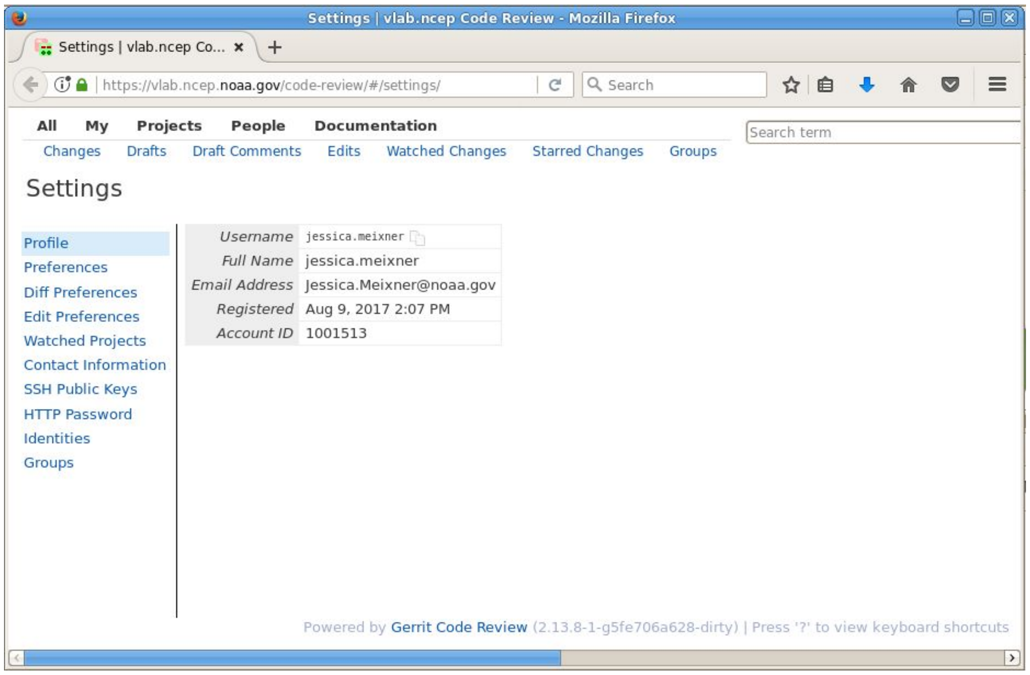
WW3 VLAB - GIT/GERRIT

VLab: NOAA Virtual Lab - provides an environment for collaboration
Git: Distributed version control system
Gerrit: Provides code review (Git/Gerrit are basically used here interchangably)
Redmine: More/less equivalent to Trac

Gerrit Set-up

NOTE: You must do this for each user on each computer you will use GIT/GERRIT on WW3.
To find more information: https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki/Gerrit_Configuration
The steps below are taken from the above website and from Mark Pott's.

STEP 0: Make sure you have a VLab account (email jessica.meixner) and have followed the email to complete setting up your VLab account.
STEP 1: Log into https://vlab.ncep.noaa.gov/code-review/
STEP 2: Now, go to https://vlab.ncep.noaa.gov/code-review/#/settings/ where you can find your User Name and email address, which will be used in the next steps. (Note you can get to this page by clicking on your user name in the top right corner and selecting settings).



Username	jessica.meixner
Full Name	jessica.meixner
Email Address	Jessica.Meixner@noaa.gov
Registered	Aug 9, 2017 2:07 PM
Account ID	1001513

Powered by Gerrit Code Review (2.13.8-1-g5fe706a628-dirty) | Press '?' to view keyboard shortcuts

STEP 3: Upload your SSH key to Gerrit

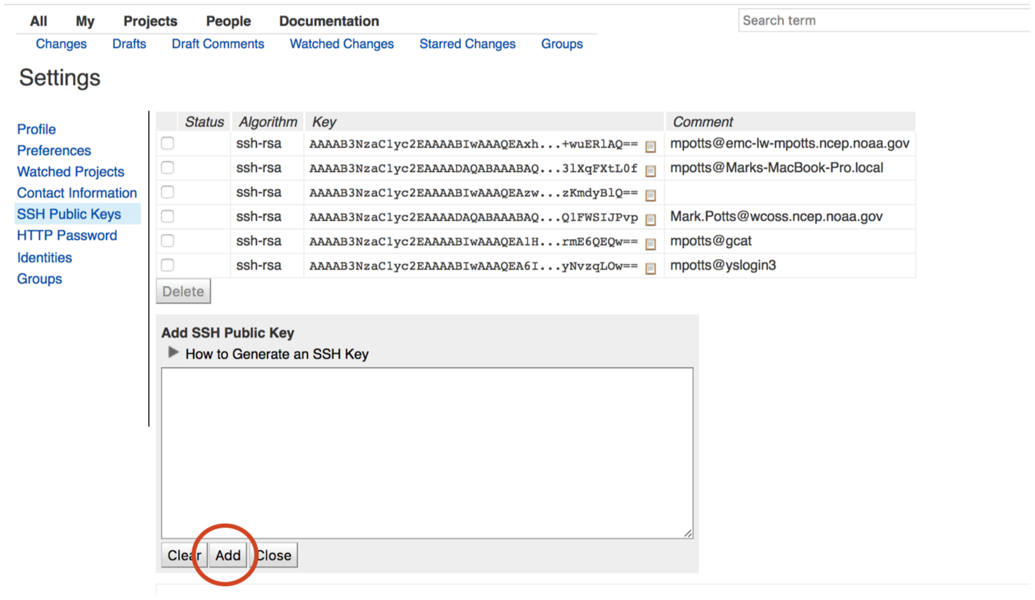
3.1 See if you have a ssh key:

```
        >> cat ~/.ssh/id_rsa.pub
```

3.2 If you do not, create one:

```
       >> ssh-keygen -t rsa
```

3.4 Then, cat your *~/.ssh/id_rsa.pub* and upload it to Gerrit by going to:
https://vlab.ncep.noaa.gov/code-review/#/settings/ssh-keys
Click on add:



	Status	Algorithm	Key	Comment
☐		ssh-rsa	AAAAB3NzaC1yc2EAAAABIwAAAQEAxh...+wuERlAQ==	mpotts@emc-lw-mpotts.ncep.noaa.gov
☐		ssh-rsa	AAAAB3NzaC1yc2EAAAADAQABAAABAQ...31XqFXtL0f	mpotts@Marks-MacBook-Pro.local
☐		ssh-rsa	AAAAB3NzaC1yc2EAAAABIwAAAQEAzw...zKmdyBlQ==	
☐		ssh-rsa	AAAAB3NzaC1yc2EAAAADAQABAAABAQ...Q1FWSIJPvp	Mark.Potts@wcoss.ncep.noaa.gov
☐		ssh-rsa	AAAAB3NzaC1yc2EAAAABIwAAAQEA1H...rmE6QEQw==	mpotts@gcat
☐		ssh-rsa	AAAAB3NzaC1yc2EAAAABIwAAAQEA6I...yNvzqLOw==	mpotts@yslogin3

STEP 5: Git configuration

In the following, make sure to match EXACTLY (including CaSe) your first and last name and email as configured in Redmine (https://vlab.ncep.noaa.gov/code-review/#/settings/). In your terminal:

```
>> git config --global user.name "Your Name"
>> git config --global user.email "first.last@noaa.gov"
```

Note if you are an external email the @noaa.gov should match your email

STEP 4: SSH Config

Unlike the other steps.. This step is not required but it can simplify commands.

Modify your ~/.ssh/config file with the following, replacing the Firstname.Lastname with your user name, exactly as written on https://vlab.ncep.noaa.gov/code-review/#/settings/

```
Host gerrit
HostName vlab.ncep.noaa.gov
        User Firstname.Lastname
        Port 29418
```

If you do not have or use an ssh/config file see https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki/Gerrit_Configuration for more instructions. For NOAA user's on WCOSS-- there is a special work around you need to do. See Mark Pott's slides or me for this information.

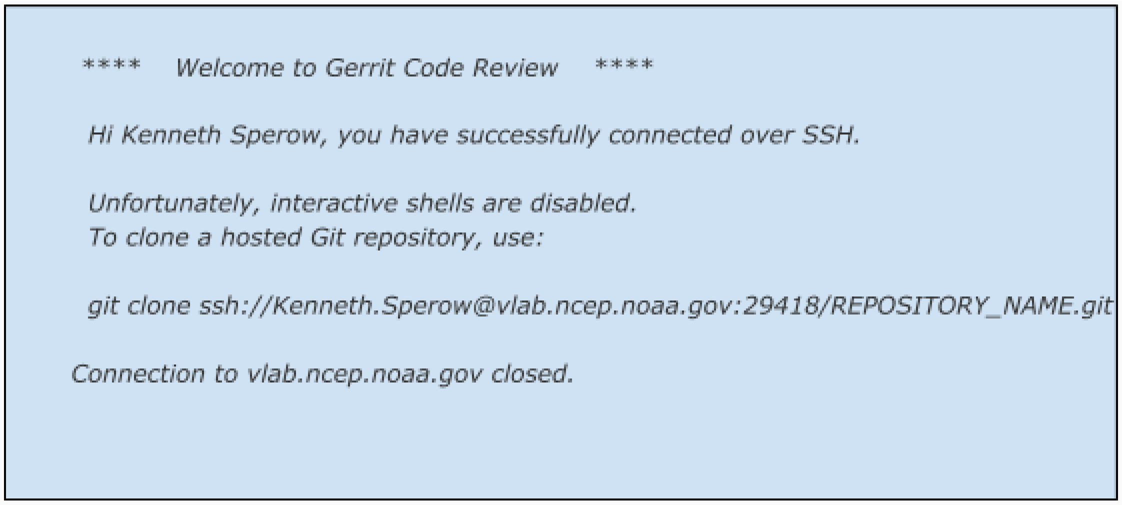
STEP 6: Test the connection

```
>> ssh gerrit
```

If you skipped STEP5:

```
>> ssh -p 29418 Firstname.Lastname@vlab.ncep.noaa.gov
```

If it's successful, you should see something like:



```
 ****   Welcome to Gerrit Code Review    ****

  Hi Kenneth Sperow, you have successfully connected over SSH.

  Unfortunately, interactive shells are disabled.
  To clone a hosted Git repository, use:

  git clone ssh://Kenneth.Sperow@vlab.ncep.noaa.gov:29418/REPOSITORY_NAME.git

Connection to vlab.ncep.noaa.gov closed.
```

Clone the WW3 Repo (without message hooks, which are needed for submitting code for review, not recommended)

After setting up Gerrit, we are ready to clone:

```
>> git clone gerrit:EMC_ww3
```

If you skipped Step 5 above:

```
>> git clone ssh://Firstname.Lastname@vlab.ncep.noaa.gov:29418/EMC_ww3
```

Note: If you do not have access to EMC_ww3, email Jessica.Meixner@noaa.gov to get access.

Clone the WW3 Repo with Message Hooks (recommended)

The messages hooks are for keeping things organized when pushing code for review in Gerrit.

To clone with message hooks in one step:

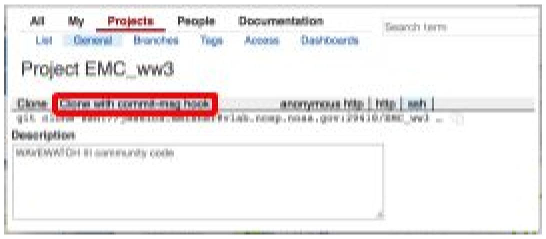
```
>> git close gerrit:EMC_ww3 && scp gerrit:hooks/commit-msg
EMC_ww3/.git/hooks/
```

If you skipped Step 5 above:

```
>> git clone ssh://Firstname.Lastname@vlab.ncep.noaa.gov:29418/EMC_ww3 &&
scp -p -P 29418 Firstname.Lastname@vlab.ncep.noaa.gov:hooks/commit-msg
EMC_ww3/.git/hooks/
```

Note: If you do not have access to EMC_ww3, email Jessica.Meixner@noaa.gov to get access.

TIP: You can copy the commands from
https://vlab.ncep.noaa.gov/code-review/#/admin/projects/EMC_ww3



Project EMC_ww3

[illegible]

Description

WAVEWATCH III community code

Notes about EMC_ww3 GIT repo:

1. The GIT repo was created with the svn trunk r98082
2. Binary and large files were not migrated to GIT.
3. The cases and smc_docs folder were removed. The smc_docs folder as a whole will come back when the smc documentation is in latex format. For now both cases and the smc_docs folder are stored on the ftp site along with files for regtests that are binary. The ftp site: `ftp://polar.ncep.noaa.gov/tempor/ww3ftp/ww3_from_ftp.tar.gz`
4. To automatically download the data from the ftp site:
   ```
   >> sh model/bin/ww3_from_ftp.sh
   ```
5. To keep from adding unintentional files to the GIT repo (and therefore keep binary files out…) I have added a .gitignore file
6. The following files were accidentally removed (mistaken for binary) were re-added to the GIT repo without their history (Note, these files had not changed in the past 4 years): -manual/run/core.tex -regtest/mww3_test_{04/05}/input/{curr/depth/mask/wind}.data
7. If you need to add a binary file for a regtest, this will need to be added to the ftp site. Contact the code manager: jessica.meixner@noaa.gov
8. PLEASE DO NOT ADD LARGE OR BINARY FILES TO THE GIT REPO!!!!!!!!!

Common GIT commands

- The following has been pulled from presentations by Ken Sparrow and Mark Potts as well as the internet…

From your EMC_ww3 directory:

- Check the status
  ```
  >> git status
  ```
- List local branches
  ```
  >> git branch
  ```
  *Note, the branch you are on is denoted with a **
- List remote branches
  ```
  >> git branch -r
  ```
- List all branches (local and remote)
  ```
  >> git branch -a
  ```
  *Note, the branch you are on is denoted with a **
- Checkout/change working/staging area to branch ‘branchname’
  ```
  >> git checkout branchname
  ```
- Create branch from current branch and switches to it
  ```
  >> git checkout -b newbranchname
  ```
- Commit changes to a local repo
  ```
  >> git commit -a
  ```
  Automatically stages files that have been modified and deleted, but new files you have not told Git about are not affected.
- Add files to be committed
  ```
  >> git add newfile
  ```
- Delete files from repo
  ```
  >> git rm oldfile
  ```
- Unstage changes
  ```
  >> git reset HEAD
  ```
- Revert last local commit (not pushed yet)
  ```
  >> git reset --hard HEAD~1
  ```
- Push updates to VLab to keep it up to date with your local changes
  ```
  >> git push origin mybranchname:mybranchname
  ```
- Retrieve changes from a central repo
  ```
  >> git fetch
  ```
- Retrieves changes from a central repo and merges changes into working area
  ```
  >> git pull
  ```
- Save off copy of modifications to be used later without committing
  ```
  >> git stash
  ```

GIT help:

- `>> man git`
- `>> man git-<option>`
- Git SVN Crash Course - http://git.or.cz/course/svn.html
- (Easy) Git for SVN users- https://people.gnome.org/~newren/eg/git-for-svn-users.html
- Git for SVN Users Cheat Sheet
  https://www.git-tower.com/blog/git-for-subversion-users-cheat-sheet/
- Git website (You can do the learn git in web-browser tutorial from this page)
  https://git-scm.com
- Redmine wiki page: https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki
- Gerrit Help: https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki/Gerrit_Help

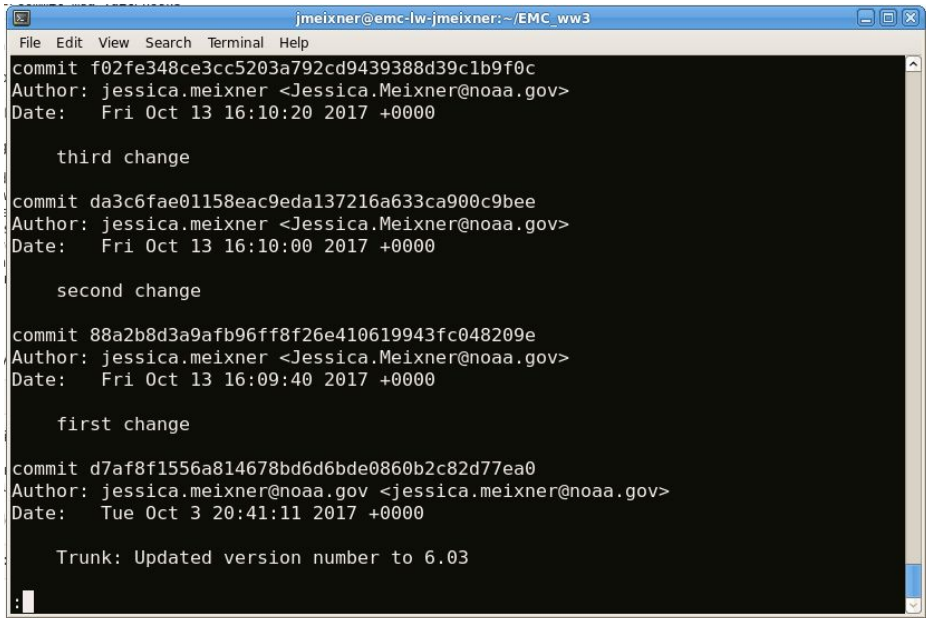
Redmine and Code Review Pages for WW3:

- Redmine: https://vlab.ncep.noaa.gov/redmine/projects/emc-ww3
  - Similar to the trac page
  - Has code viewer, tickets, milestones, documents page and a wiki.
- Code review page: https://vlab.ncep.noaa.gov/code-review/#/admin/projects/EMC_ww3
  - Go here to see code you have submitted for review
  - See code to review others have submitted

Workflow for a small bug fix:

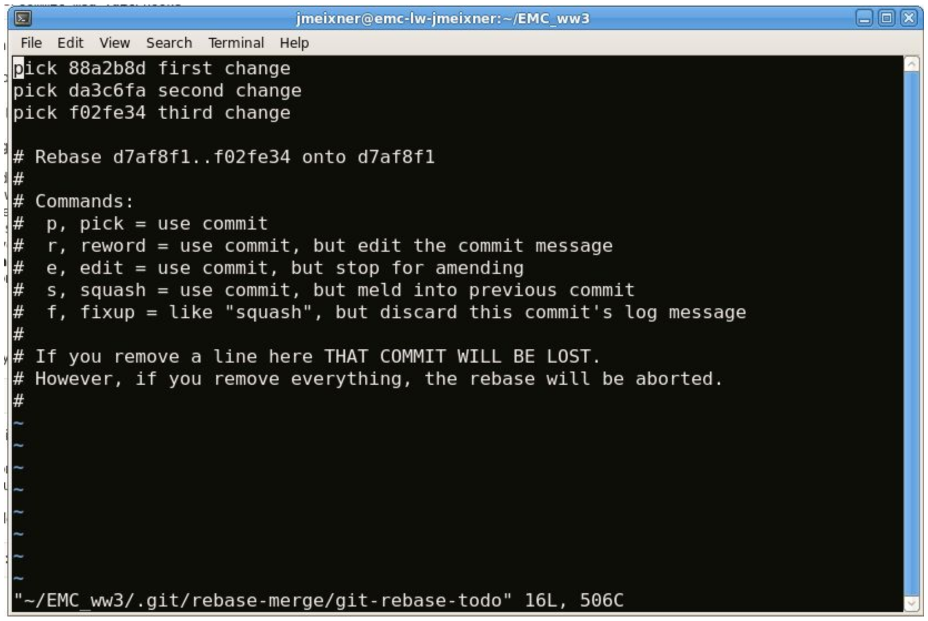
This is for when you have found a small bug in the master and want to fix this bug. All changes are kept on a local branch until being pushed back to the master for review.

1. Clone the repo:
   ```
   >> git clone gerrit:EMC_ww3 && scp gerrit:hooks/commit-msg
   EMC_ww3/.git/hooks/
   ```
2. If you already have a clone, ensure it is updated:
   ```
   >> cd <local_git_repo>
   ```
   Check to see if you have any changes:
   ```
   >> git status
   ```
   Check which branch you are on:
   ```
   >> git branch
   ```
   Change to master if needed
   ```
   >> git checkout master
   ```
   Update your copy with the latest from vlab:
   ```
   >>git fetch
   >>git rebase origin/master
   ```
3. Create a topic branch (from the master) where you will be working from locally.
   `>> git checkout master` (This is the branch from which you will be creating a branch from, you might already be here)
   ```
   >> git checkout -b <name_of_bugfix_branch>
   ```
4. Make your changes locally within the bugfix branch <name_of_bugfix_branch>. Remember `git branch` will show you which branch you are on.
5. Add and commit your changes within the local repository.
   ```
   >> git add <filename(s)>
   >> git commit
   ```
6. Continue with steps 4 and 5 until you are ready to push the bug fix back to the master.
7. Make sure you pull the latest changes from the remote master:
   ```
   >> git pull origin/master
   ```
   -or-
   ```
   >>git fetch
   >>git rebase origin/master
   ```
8. *If you have multiple commits:* You need to "squash" the commits to one update. This is so that you only have 1 code review in Gerrit instead of multiple reviews. To do this (or see https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki/Gerrit_Help#Squash-your-changes-before-you-Push):
   a. Figure out how many commits you need to squash. You can do this with the `git log` command:


jmeixner@emc-lw-jmeixner:~/EMC_ww3
File Edit View Search Terminal Help

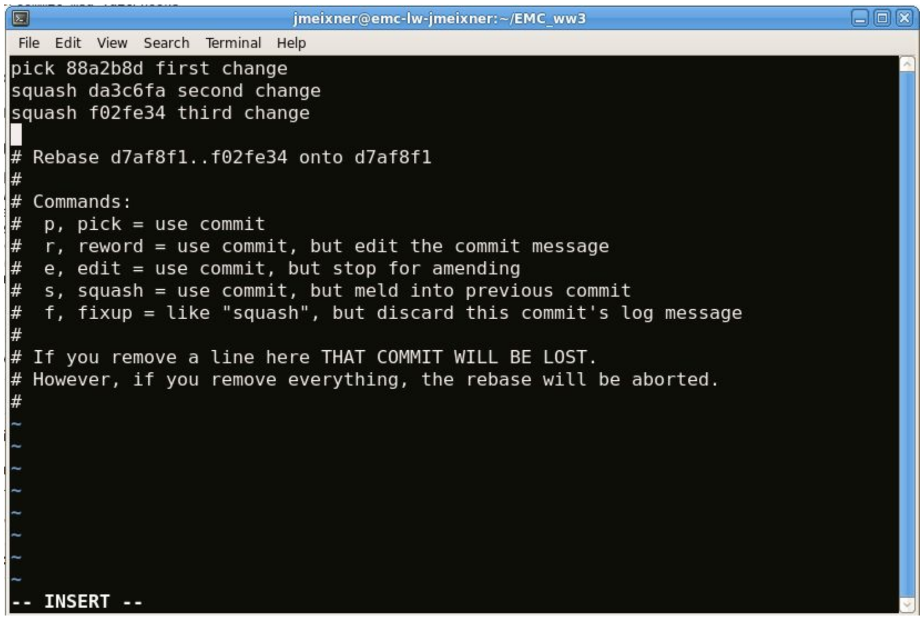
```
commit f02fe348ce3cc5203a792cd9439388d39c1b9f0c
Author: jessica.meixner <Jessica.Meixner@noaa.gov>
Date:   Fri Oct 13 16:10:20 2017 +0000

    third change

commit da3c6fae01158eac9eda137216a633ca900c9bee
Author: jessica.meixner <Jessica.Meixner@noaa.gov>
Date:   Fri Oct 13 16:10:00 2017 +0000

    second change

commit 88a2b8d3a9afb96ff8f26e410619943fc048209e
Author: jessica.meixner <Jessica.Meixner@noaa.gov>
Date:   Fri Oct 13 16:09:40 2017 +0000

    first change

commit d7af8f1556a814678bd6d6bde0860b2c82d77ea0
Author: jessica.meixner@noaa.gov <jessica.meixner@noaa.gov>
Date:   Tue Oct 3 20:41:11 2017 +0000

    Trunk: Updated version number to 6.03

:
```

b. Determining that number, (in this example 3), you squash:
`>> git rebase -i branchname~<number of commits to include>`
You will then get a screen like this:


jmeixner@emc-lw-jmeixner:~/EMC_ww3
File Edit View Search Terminal Help

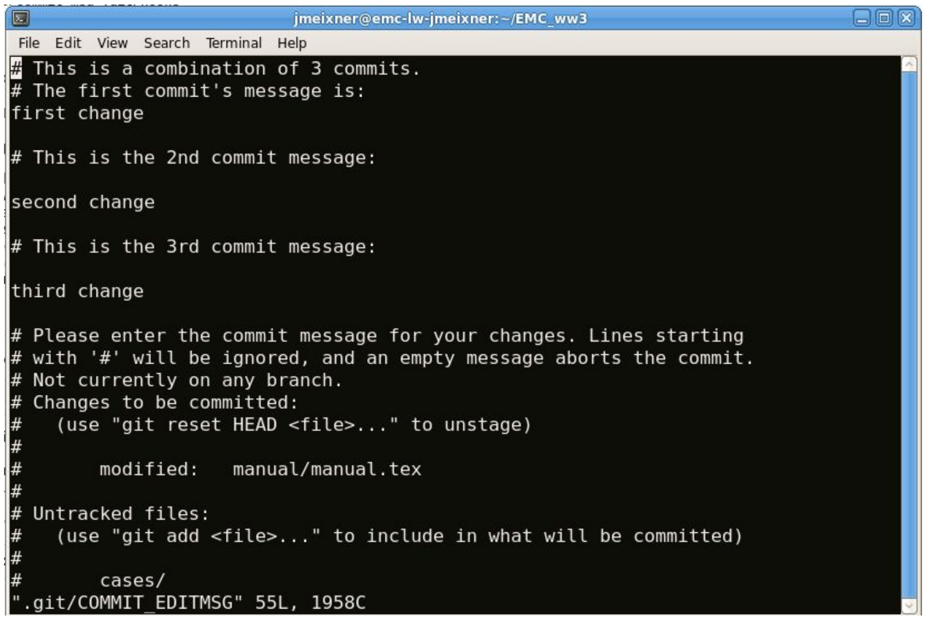
```
pick 88a2b8d first change
pick da3c6fa second change
pick f02fe34 third change

# Rebase d7af8f1..f02fe34 onto d7af8f1
#
# 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.
#
~
~
~
~
~
~
~
~
"~/EMC_ww3/.git/rebase-merge/git-rebase-todo" 16L, 506C
```

c. Change all but the first ‘pick’ to ‘squash’ in the editor and save, so:



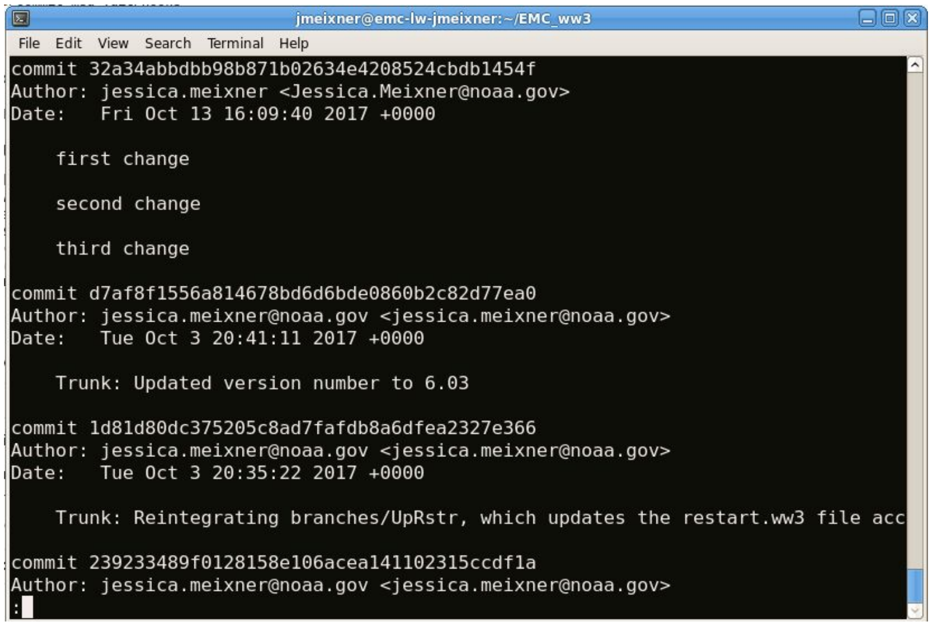
```
pick 88a2b8d first change
squash da3c6fa second change
squash f02fe34 third change

# Rebase d7af8f1..f02fe34 onto d7af8f1
#
# 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.
#
~
~
~
~
~
~
~
~
-- INSERT --
```

d. Then you get a screen to edit the comment for the commit:



```
# This is a combination of 3 commits.
# The first commit's message is:
first change

# This is the 2nd commit message:

second change

# This is the 3rd commit message:

third change

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# Not currently on any branch.
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   manual/manual.tex
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       cases/
".git/COMMIT_EDITMSG" 55L, 1958C
```

e. Now the commits are “squashed” to one commit, and a `git log` then shows you:



```
commit 32a34abbdbb98b871b02634e4208524cbdb1454f
Author: jessica.meixner <Jessica.Meixner@noaa.gov>
Date:   Fri Oct 13 16:09:40 2017 +0000

    first change

    second change

    third change

commit d7af8f1556a814678bd6d6bde0860b2c82d77ea0
Author: jessica.meixner@noaa.gov <jessica.meixner@noaa.gov>
Date:   Tue Oct 3 20:41:11 2017 +0000

    Trunk: Updated version number to 6.03

commit 1d81d80dc375205c8ad7fafdb8a6dfea2327e366
Author: jessica.meixner@noaa.gov <jessica.meixner@noaa.gov>
Date:   Tue Oct 3 20:35:22 2017 +0000

    Trunk: Reintegrating branches/UpRstr, which updates the restart.ww3 file acc

commit 239233489f0128158e106acea141102315ccdf1a
Author: jessica.meixner@noaa.gov <jessica.meixner@noaa.gov>
:
```

9. Now you are ready to push to Gerrit for code review.
   https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki/Gerrit_Help#Push-to-Gerrit-Code-Review
   `>> git push gerrit:EMC_ww3 HEAD:refs/for/master`

   See https://vlab.ncep.noaa.gov/redmine/projects/vlab/wiki/Gerrit_Configuration#Tired-of-typing-HEADrefsforBRANCH_NAME for a tip in simplifying the above command to:
   `>> git push review_master`
10. Now the code is ready to be reviewed through the Gerrit web interface.
11. Once the code in reviewed, approved, and verified it can be submitted for inclusion in the repository.

Workflow for larger changes with an existing remote branch:

This is for when you are adding something to a remote branch. These remote branches are for larger changes in ongoing projects that are not yet ready to be pushed to the master.

1. Clone the repo:
   ```
   >> git clone gerrit:EMC_ww3 && scp gerrit:hooks/commit-msg EMC_ww3/.git/hooks/
   >> cd EMC_ww3
   ```
2. Checkout the remote branch and create a local branch:
   List the remote branches
   ```
   >> git branch -r
   ```
   Checkout the remote branch you want:
   ```
   >> git checkout origin/<remote_branch_name>
   ```
   Then make a local branch based off the remote branch:
   ```
   >> git checkout -b <local_branch_name>
   ```
3. Make your changes locally within the local branch <local_branch_name>.
4. Add and commit your changes within the local repository
   ```
   >> git add <filename(s)>
   >> git commit
   ```
5. When you are ready to push the changes back to the branch, first make sure you are up to date with the remote branch. So:
   ```
   >> git pull origin/<remote_branch_name>
   ```
   -or-
   ```
   >>git fetch
   >>git rebase origin/<remote_branch_name>
   ```
6. Now push your changes to the branch. (Note this doesn't have to go through a Gerrit review, so squashing the changes isn't necessary, but feel free to).
   ```
   >> git push gerrit:EMC_ww3 HEAD:refs/for/<remote_branch_name>
   ```

Create a new remote branch by:
git push gerrit:EMC_ww3 local_branch_name:new_branch_name_on_remote

When you are ready for a branch to be pushed to the master:
1. Make sure the branch is up to date with the master:
   ```
   >>git fetch
   >>git rebase origin/master
   ```
2. Commit and push this to the branch.
3. Then push your local branch based off the remote branch to the master for review:
   ```
   >> git push gerrit:EMC_ww3 HEAD:refs/for/<remote_branch_name>
   ```