git.
Tips & Tricks.

g-2 art Workshop

August 2014

Leah Welty-Rieger
Northwestern University

*The tips in this document are how I do things. There are other ways.
But you shouldn't get into trouble my way.

I want to develop code!

- Awesome! We want you to too!
- How do you get started?
    - That is what we are going to talk about in these slides.
- Following are some rules
    - Rules? What do you mean rules? This is physics! We don't need rules!
    - To have a consistent software process we have put in place some simple "guidelines" *[by guidelines I mean rules]* to help you AND the g-2 software community in large.
- And following the rules are some tips for using git in a useful way
    - Not including some of the really interesting things git allows you to do.
- If you don't know something...Just ask! Lots of help available!

Get the Code

- We use the source control system called ***git***
  - The main git webpage is full of great detailed information. Presented here is just the tip of the iceberg. For lots more information head to: http://git-scm.com
- Information on computing
  - On Redmine. The computing "fast" index can be found here: https://cdcvs.fnal.gov/redmine/projects/g-2/wiki/SandCFastIndex
  - The releases of our code are listed at the top with the most recent first:
    - At the time of this document the most recent release was v201402 and instructions can be found here: https://cdcvs.fnal.gov/redmine/projects/g-2/wiki/V201402

Few Words on git

- git is a distributed version control system
  - You are not just checking out the current tip of the source code you are cloning the entire repository
- Every user has (essentially) a full backup of the main server
  - Any of these copies could be used to replace the the main server in the event of a crash or corruption
- git uses a “staging” area
  - Code commits can be formatted/reviewed before actually being committed
  - Don’t have to commit all the files at once
    - You can if you want to, but you have the power to choose.

Now! Start writing code!

- **NOT. SO. FAST.**
- But almost.
- When you check out the code you are placed onto the development line of code
- <u>**YOU DO NOT WANT TO DEVELOP ANY CODE ON THE DEVELOP LINE OF CODE**</u>
  - This is saved for code and analysis that has been vetted by a series of checks and balances.
- This is true for all packages
  - gm2ringsim, gm2geom, gm2analyses, etc etc.

First...Where Am I

git branch -a

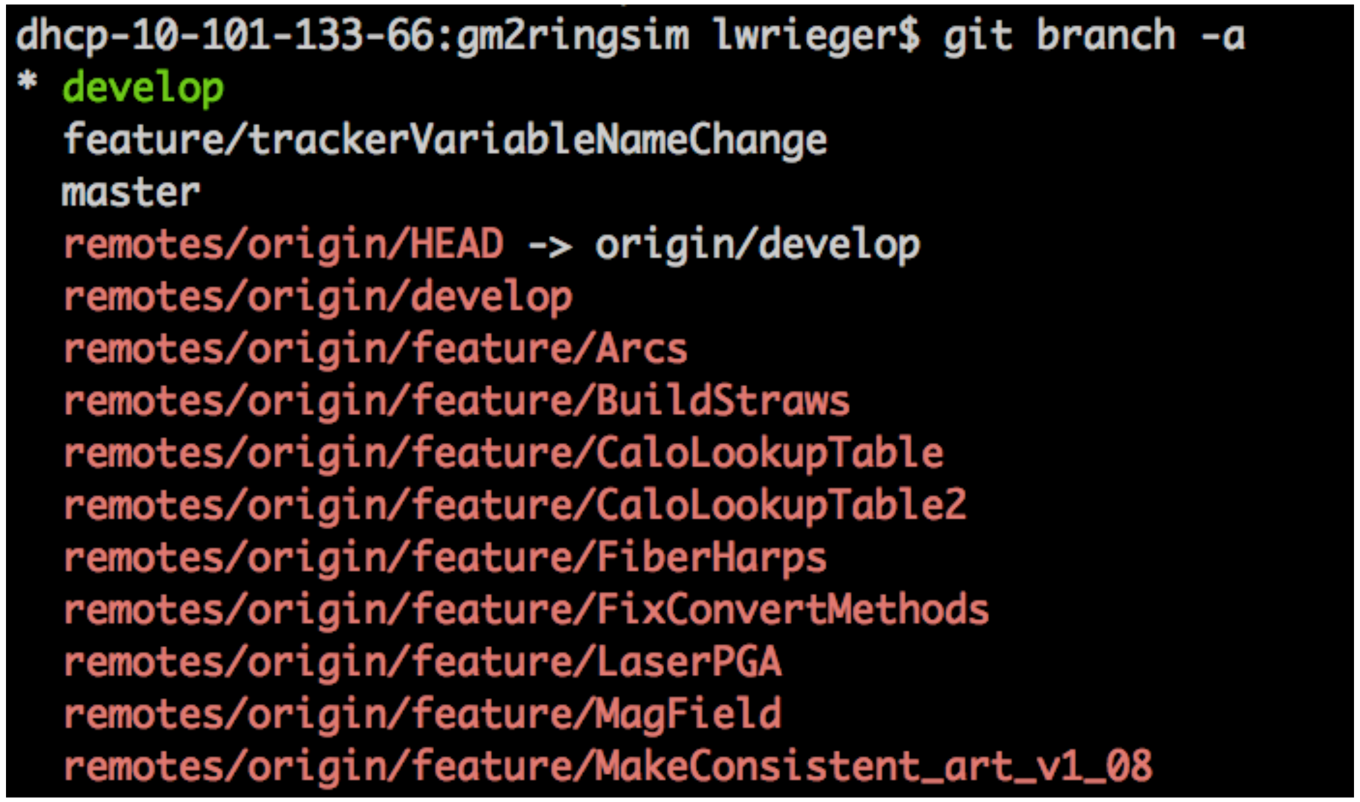
- This is the way to find out what branch you are currently sitting on
  - the -a says, show me everything not just stuff I have locally

The star shows what branch you are on

The colors for git aren't the default. Add:

```
[color]
        ui = true
```

to your .gitconfig file in your home area.



```
dhcp-10-101-133-66:gm2ringsim lwrieger$ git branch -a
* develop
  feature/trackerVariableNameChange
  master
  remotes/origin/HEAD -> origin/develop
  remotes/origin/develop
  remotes/origin/feature/Arcs
  remotes/origin/feature/BuildStraws
  remotes/origin/feature/CaloLookupTable
  remotes/origin/feature/CaloLookupTable2
  remotes/origin/feature/FiberHarps
  remotes/origin/feature/FixConvertMethods
  remotes/origin/feature/LaserPGA
  remotes/origin/feature/MagField
  remotes/origin/feature/MakeConsistent_art_v1_08
```

It's Easy to Develop Code

- So you want to write some feature...maybe you need to redesign one of the detector components.
- So you want to start a "feature"

```
git flow feature start NewDesignTrackerModule
```

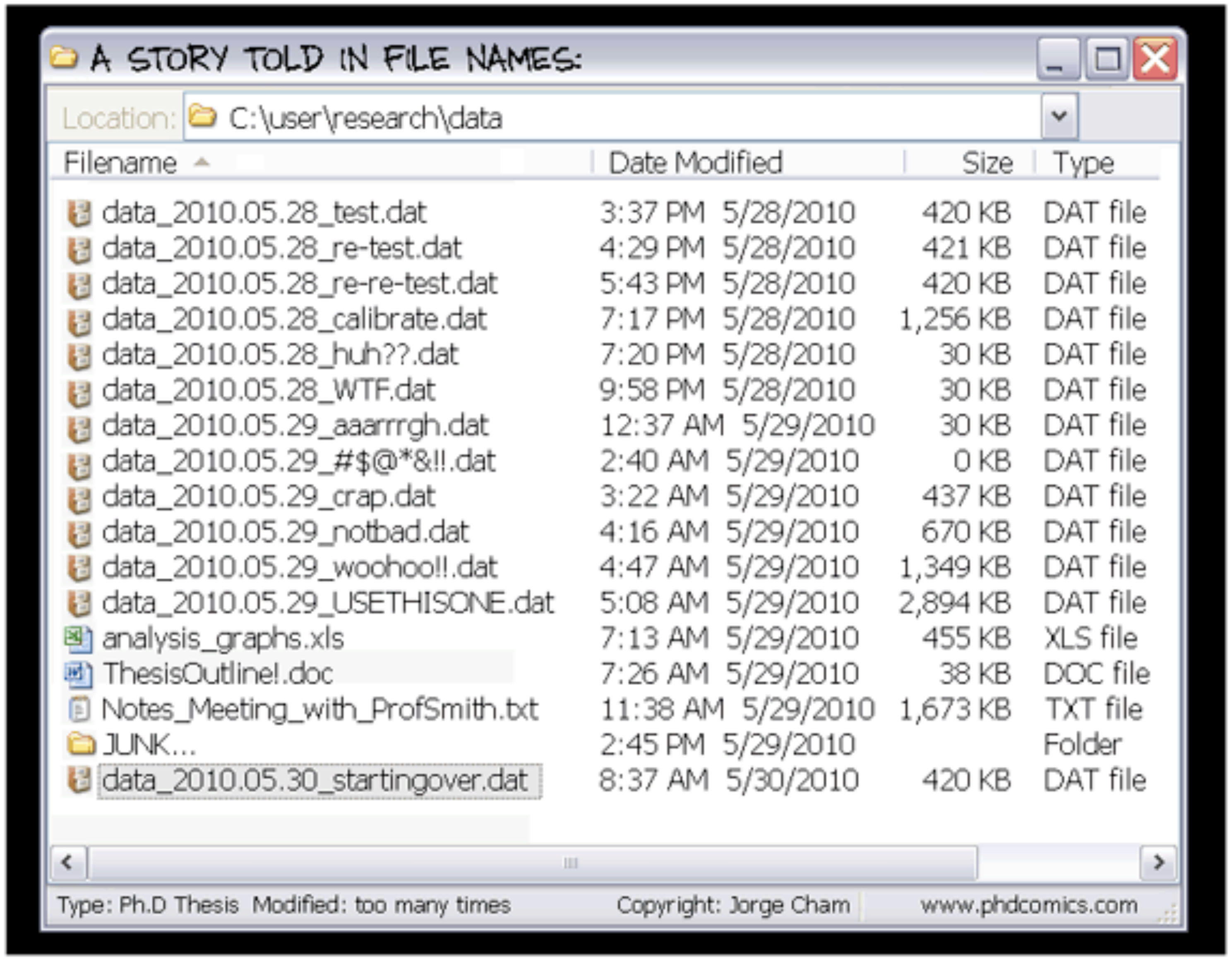
- Tip on the <u>feature</u> name:
  - Make it descriptive
    - It's very easy (I'm to fault on this one) to make a feature branch and just stay there longer than you should. Continuing to develop code that strays from the original feature
      - So making your feature name one specific thing might help you just finish that one thing before moving on.

Feature Branches Are For You

- But...not necessarily *only* you because it's a great way to share code that might not be ready for <u>everyone</u>
  - PersonA: "Hey you over there. Yes you. I'm having some problems with some code. Can you check out this branch of code and see if you can see the issue?"
  - PersonB: "Well I'm really busy...kidding, sure no problem!"
- It's a place where you can play around with design and know that you aren't going to screw up other people's stuff.
- "But Leah, I can do that on my own. I can just play around and try things and when it works I can share it with everyone"

Then Leah's Head Explodes

- Remember that time. When you had something working. And then "didn't change anything and now it doesn't work"
- Or had this situation:

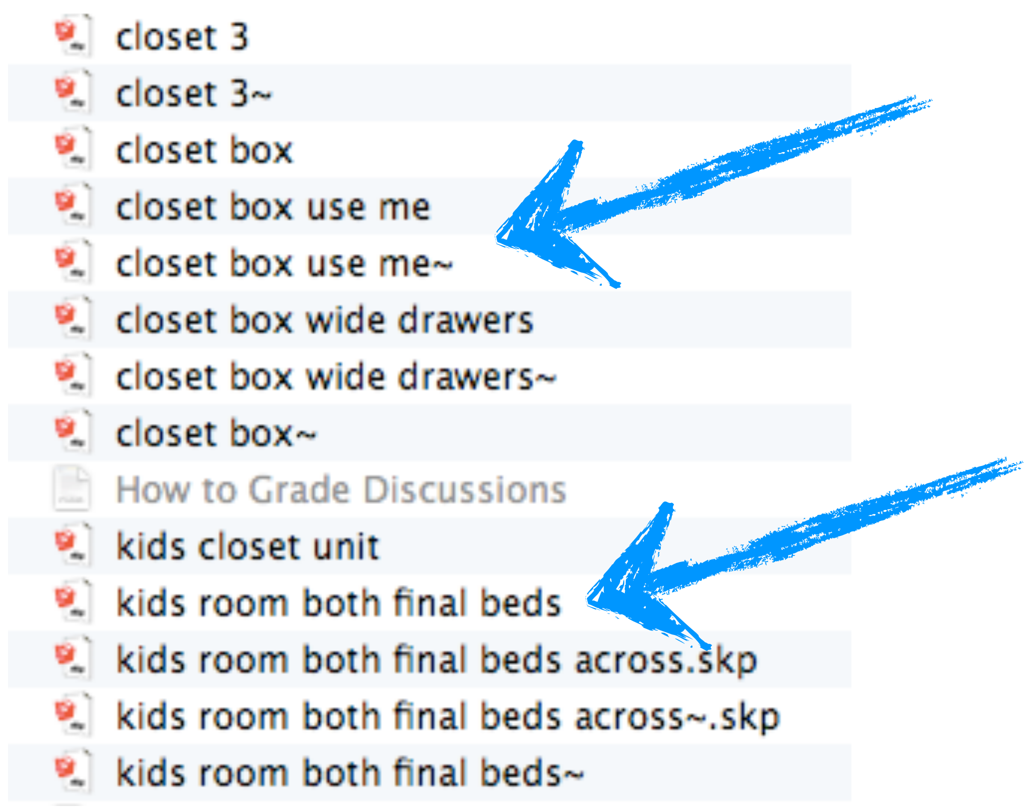


A STORY TOLD IN FILE NAMES:

Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com



closet 3
closet 3~
closet box
closet box use me
closet box use me~
closet box wide drawers
closet box wide drawers~
closet box~
How to Grade Discussions
kids closet unit
kids room both final beds
kids room both final beds across.skp
kids room both final beds across~.skp
kids room both final beds~

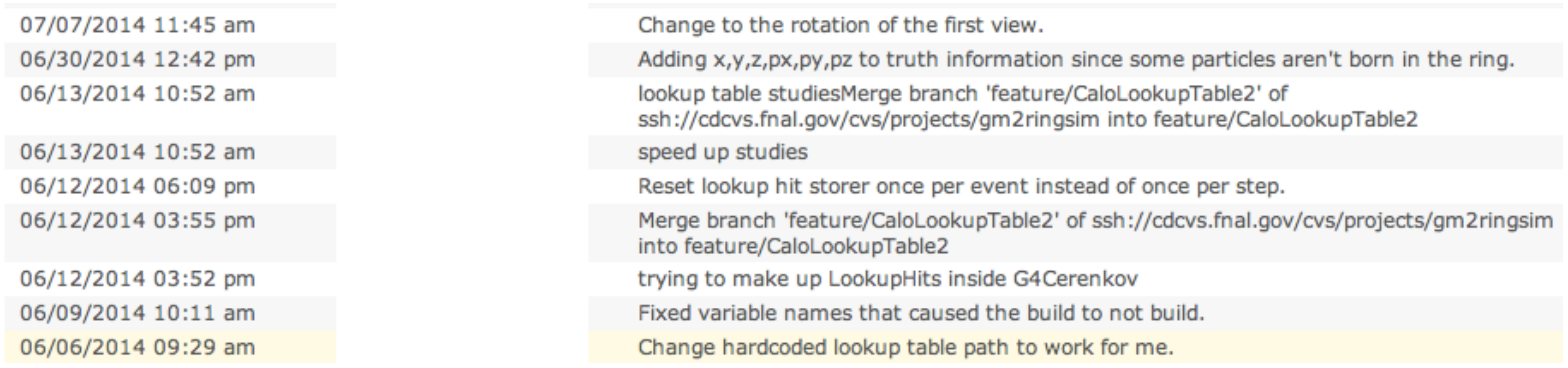
Even just putting the date after a file name does nothing for you because do you really know what is changing between the dates that you are using the files? or do you just use the latest file/directory/working area?

Local Commits. Key.

- git allows you to make local commits when you get to a point that is something you want to save.
    - Seriously. Use it as you would use Save when working on a document.
    - You. Can. Always. Go. Back.
- THEN if you screw something up with some changes you can:
    - SEE what changed!
    - AND if you can’t unscramble it, just go back to the commit where things worked.
- The key to helping you out of bad situations is commit messages that mean something

Commit Messages

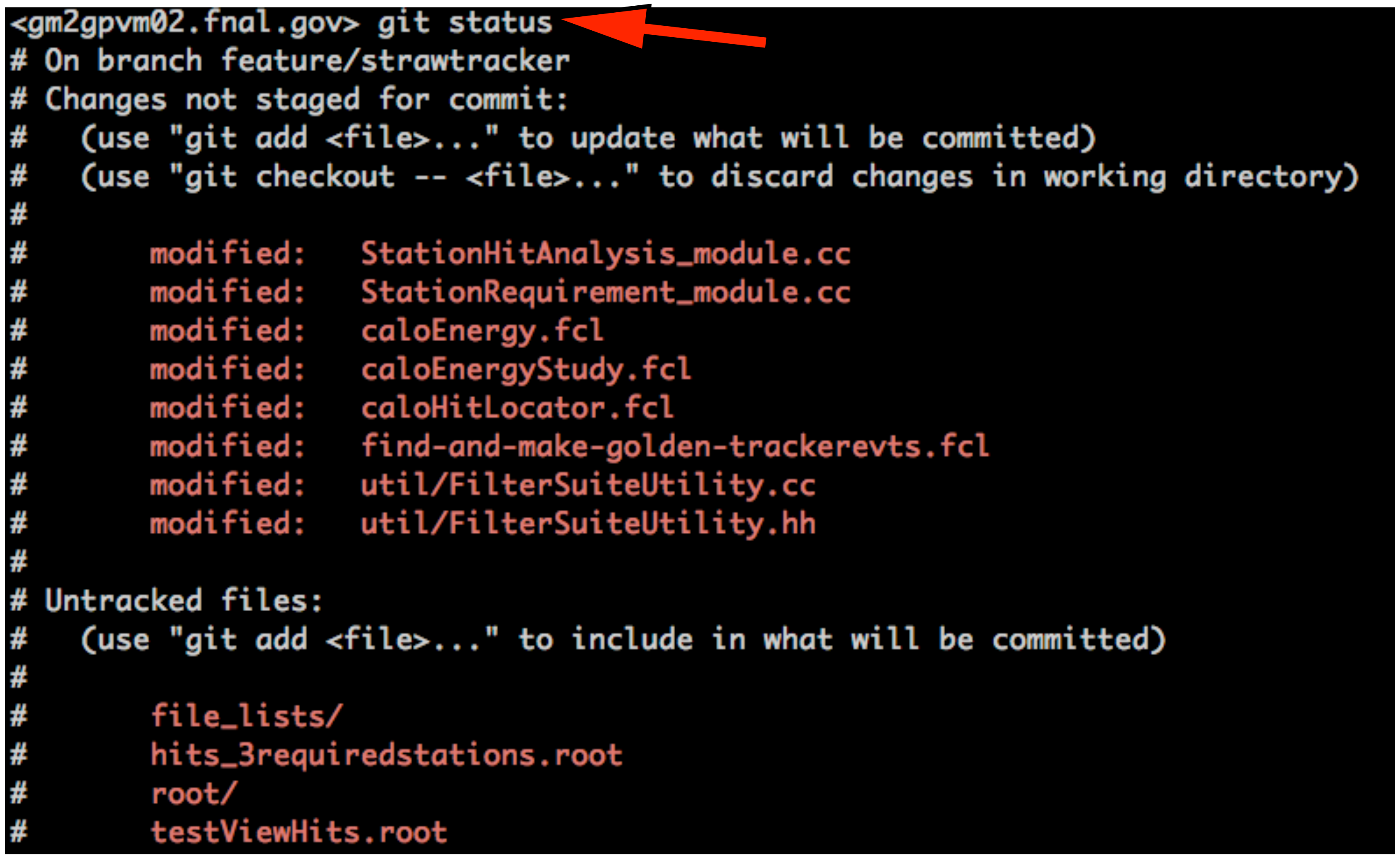
- These are all pretty good examples



07/07/2014 11:45 am	Change to the rotation of the first view.
06/30/2014 12:42 pm	Adding x,y,z,px,py,pz to truth information since some particles aren't born in the ring.
06/13/2014 10:52 am	lookup table studiesMerge branch 'feature/CaloLookupTable2' of ssh://cdcvs.fnal.gov/cvs/projects/gm2ringsim into feature/CaloLookupTable2
06/13/2014 10:52 am	speed up studies
06/12/2014 06:09 pm	Reset lookup hit storer once per event instead of once per step.
06/12/2014 03:55 pm	Merge branch 'feature/CaloLookupTable2' of ssh://cdcvs.fnal.gov/cvs/projects/gm2ringsim into feature/CaloLookupTable2
06/12/2014 03:52 pm	trying to make up LookupHits inside G4Cerenkov
06/09/2014 10:11 am	Fixed variable names that caused the build to not build.
06/06/2014 09:29 am	Change hardcoded lookup table path to work for me.

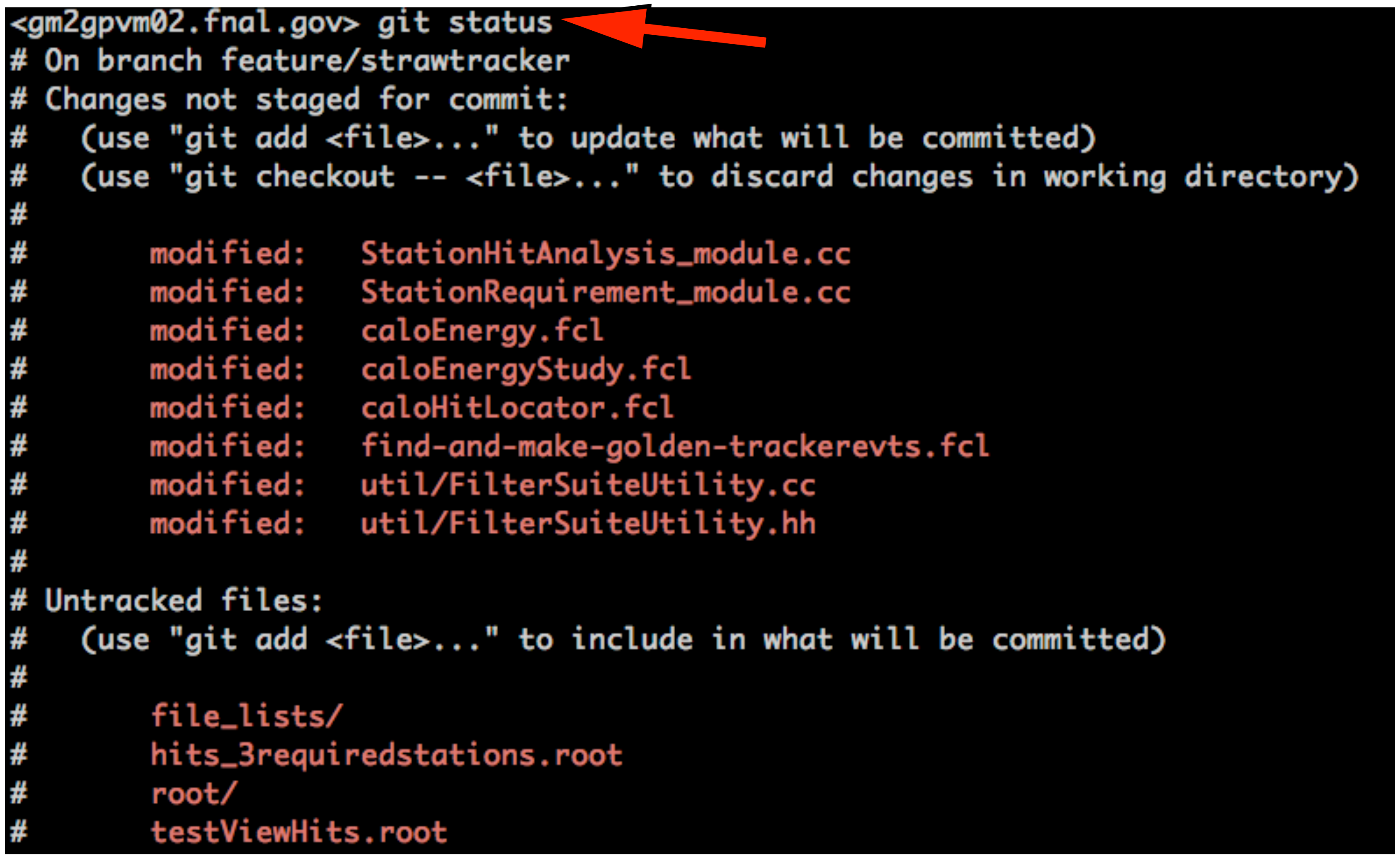
- The last one might need something more, “to work for me” what does that mean? That it won’t work for others?
- Second to last one...what variable names?
- But in general. Not. Bad.

Commit early. Commit often.



```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   StationHitAnalysis_module.cc
#       modified:   StationRequirement_module.cc
#       modified:   caloEnergy.fcl
#       modified:   caloEnergyStudy.fcl
#       modified:   caloHitLocator.fcl
#       modified:   find-and-make-golden-trackerevts.fcl
#       modified:   util/FilterSuiteUtility.cc
#       modified:   util/FilterSuiteUtility.hh
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file_lists/
#       hits_3requiredstations.root
#       root/
#       testViewHits.root
```

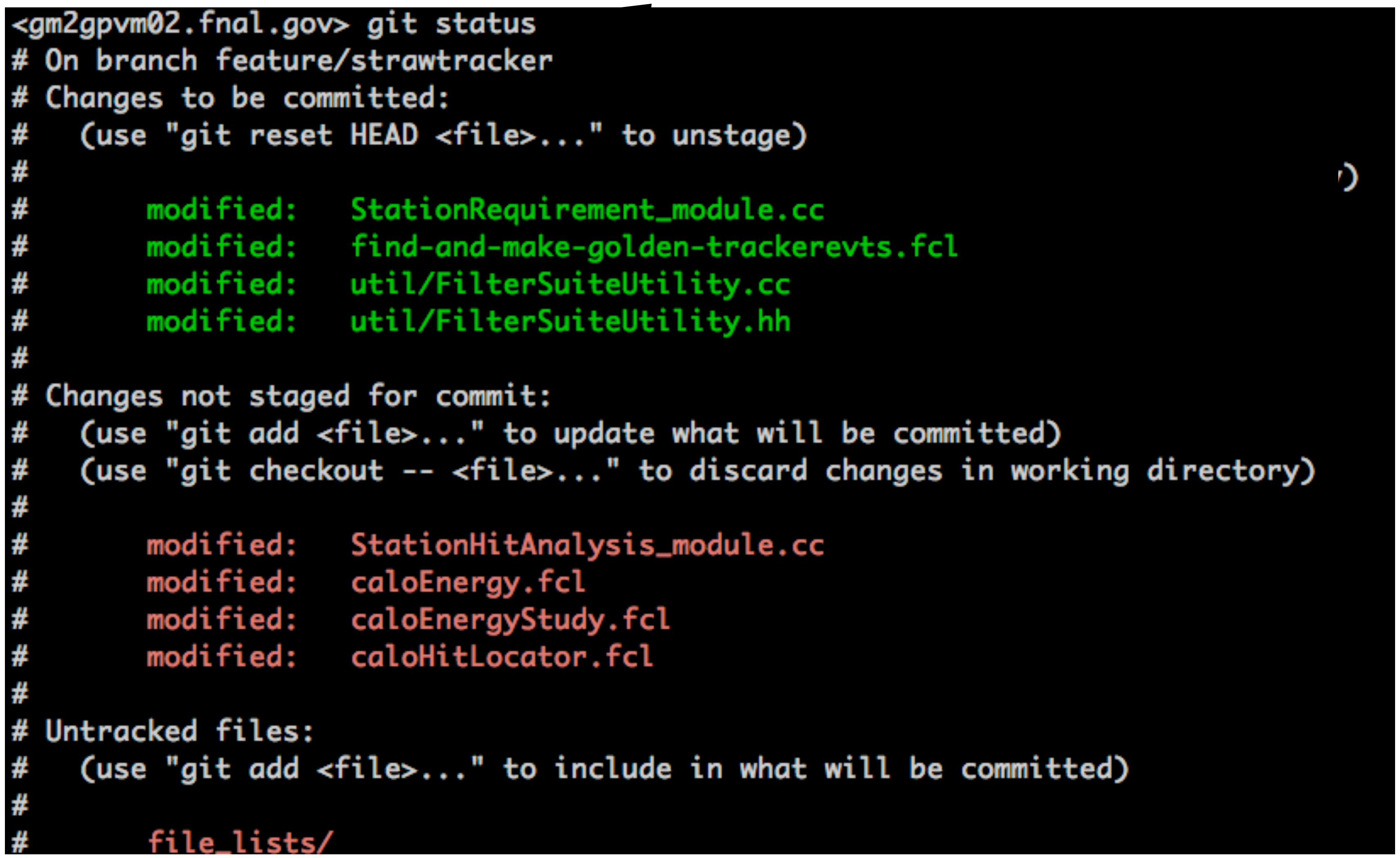
Commit early. Commit often.



```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   StationHitAnalysis_module.cc
#       modified:   StationRequirement_module.cc
#       modified:   caloEnergy.fcl
#       modified:   caloEnergyStudy.fcl
#       modified:   caloHitLocator.fcl
#       modified:   find-and-make-golden-trackerevts.fcl
#       modified:   util/FilterSuiteUtility.cc
#       modified:   util/FilterSuiteUtility.hh
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file_lists/
#       hits_3requiredstations.root
#       root/
#       testViewHits.root
```

- `git add StationRequirement_module.cc`
- `git add util/FilterSuiteUtility.cc`
- `git add util/FilterSuiteUtility.hh`
- `git add find-and-make-golden-trackerevts.fcl`

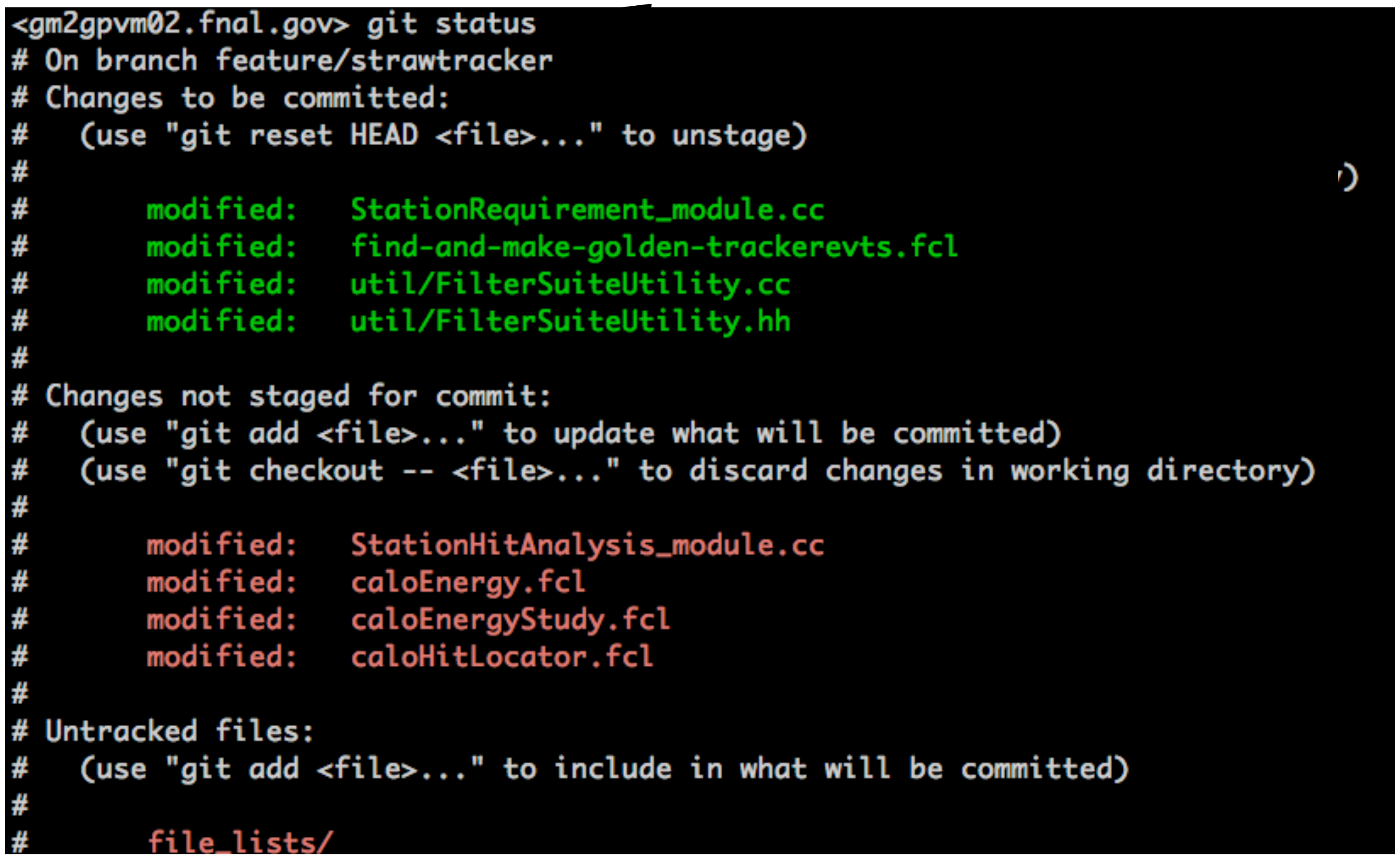
Commit early. Commit often.



```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   StationRequirement_module.cc
#       modified:   find-and-make-golden-trackerevts.fcl
#       modified:   util/FilterSuiteUtility.cc
#       modified:   util/FilterSuiteUtility.hh
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   StationHitAnalysis_module.cc
#       modified:   caloEnergy.fcl
#       modified:   caloEnergyStudy.fcl
#       modified:   caloHitLocator.fcl
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file_lists/
```

- git add StationRequirement_module.cc
- git add util/FilterSuiteUtility.cc
- git add util/FilterSuiteUtility.hh
- git add find-and-make-golden-trackerevts.fcl

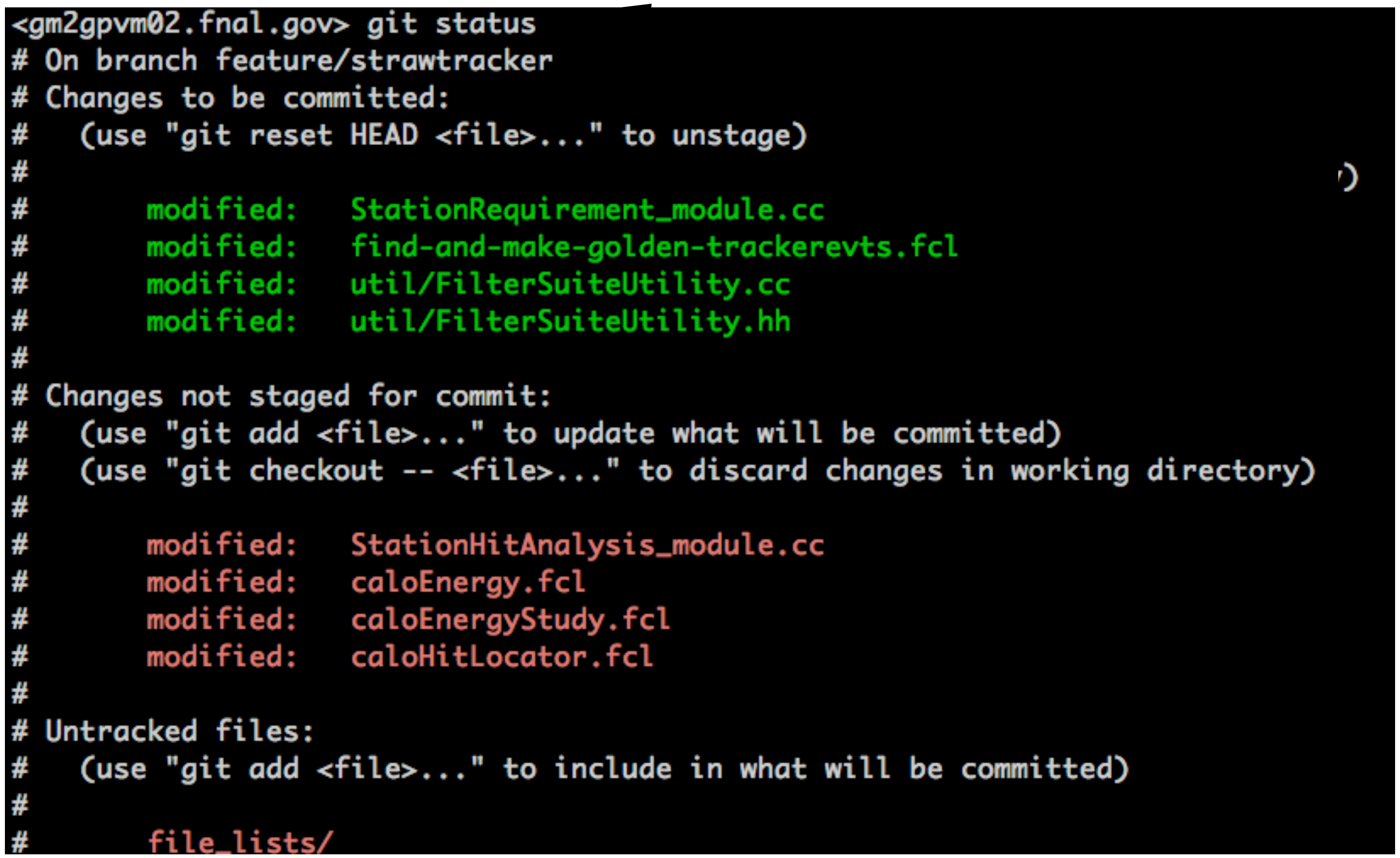
Commit early. Commit often.



```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   StationRequirement_module.cc
#       modified:   find-and-make-golden-trackerevts.fcl
#       modified:   util/FilterSuiteUtility.cc
#       modified:   util/FilterSuiteUtility.hh
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   StationHitAnalysis_module.cc
#       modified:   caloEnergy.fcl
#       modified:   caloEnergyStudy.fcl
#       modified:   caloHitLocator.fcl
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file_lists/
```

- `git add StationRequirement_module.cc`
- `git add util/FilterSuiteUtility.cc`
- `git add util/FilterSuiteUtility.hh`
- `git add find-and-make-golden-trackerevts.fcl`
- `git commit`

Commit early. Commit often.

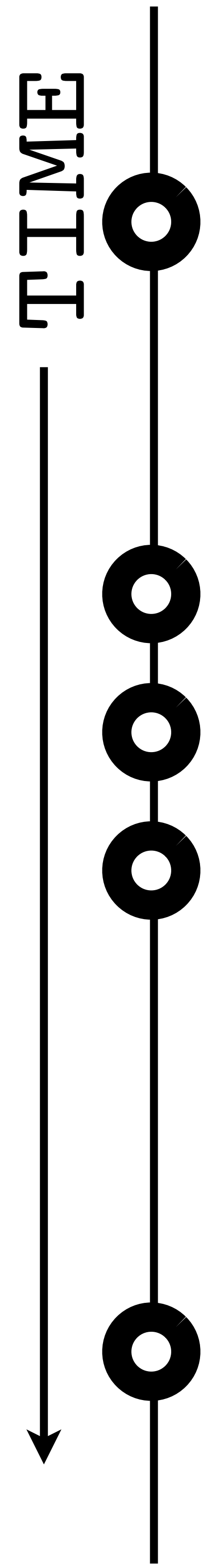


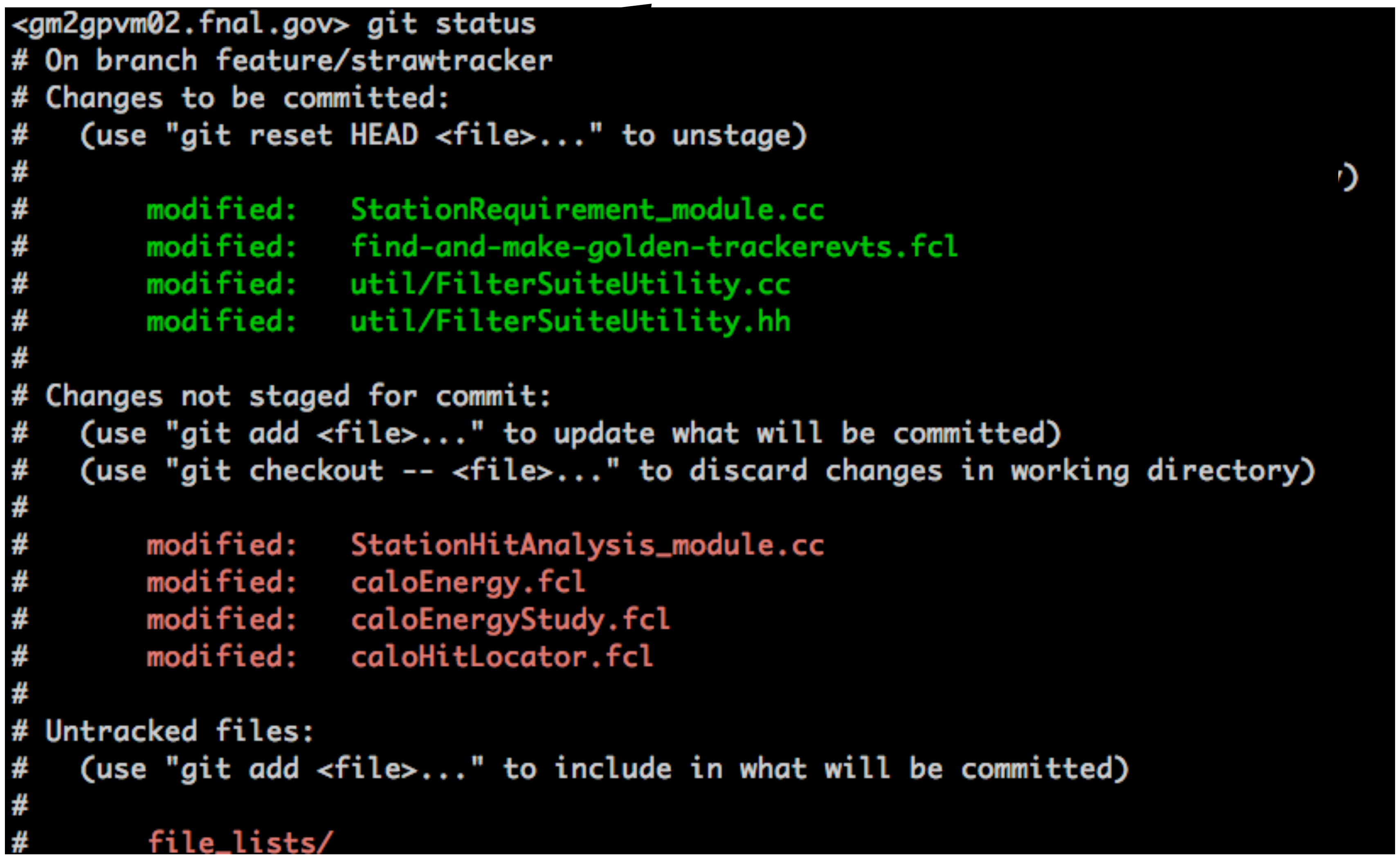
```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#                                                                              )
#       modified:   StationRequirement_module.cc
#       modified:   find-and-make-golden-trackerevts.fcl
#       modified:   util/FilterSuiteUtility.cc
#       modified:   util/FilterSuiteUtility.hh
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   StationHitAnalysis_module.cc
#       modified:   caloEnergy.fcl
#       modified:   caloEnergyStudy.fcl
#       modified:   caloHitLocator.fcl
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file_lists/
```

- git add StationRequirement_module.cc
- git add util/FilterSuiteUtility.cc
- git add util/FilterSuiteUtility.hh
- git add find-and-make-golden-trackerevts.fcl
- git commit

- At this point thrown into vi (default).
- Then enter your commit message, save the file and at that point the commit is put *locally* into your branch.
- But only locally. Nothing on Redmine has any idea you did anything.

Commit early. Commit often.





```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   StationRequirement_module.cc
#       modified:   find-and-make-golden-trackerevts.fcl
#       modified:   util/FilterSuiteUtility.cc
#       modified:   util/FilterSuiteUtility.hh
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   StationHitAnalysis_module.cc
#       modified:   caloEnergy.fcl
#       modified:   caloEnergyStudy.fcl
#       modified:   caloHitLocator.fcl
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       file_lists/
```

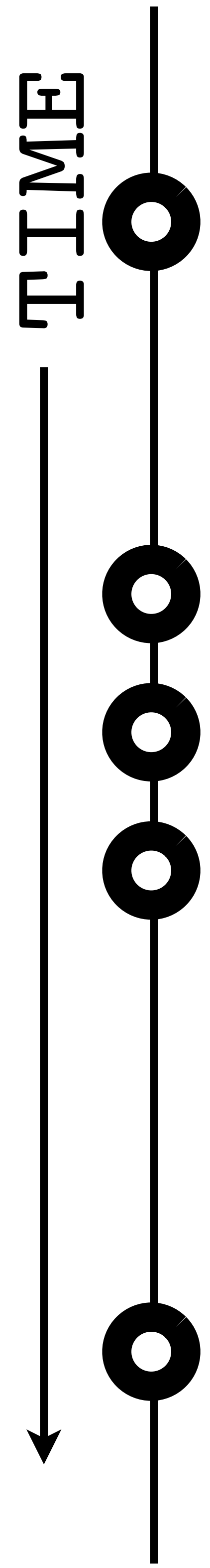
- `git add StationRequirement_module.cc`
- `git add util/FilterSuiteUtility.cc`
- `git add util/FilterSuiteUtility.hh`
- `git add find-and-make-golden-trackerevts.fcl`
- `git commit`

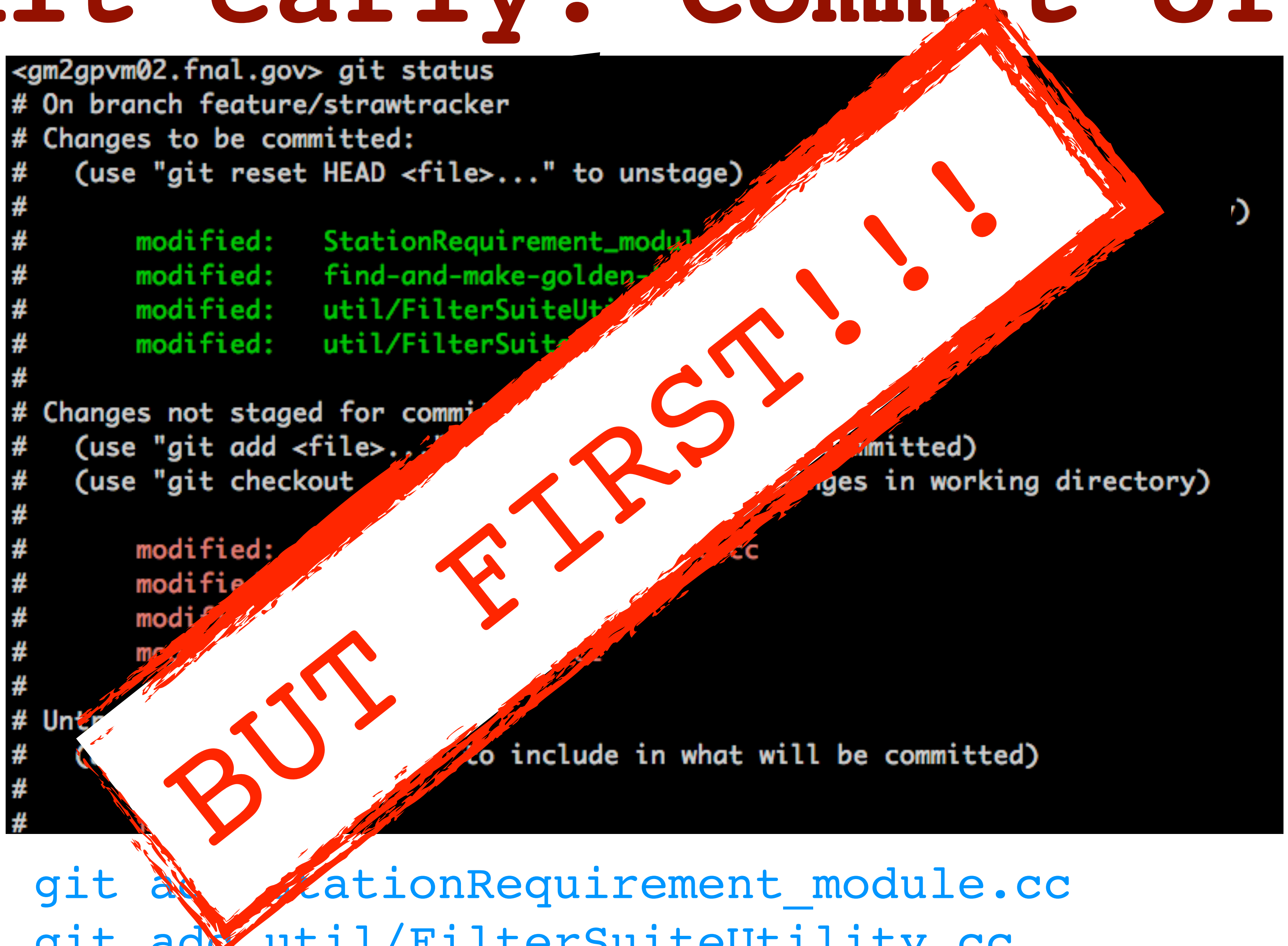
• At this point thrown into vi (default).

• Then enter your commit message, save the file and at that point the commit is put *locally* into your branch.

•But <u>only</u> locally. Nothing on Redmine has any idea you did anything.

Commit early. Commit often.





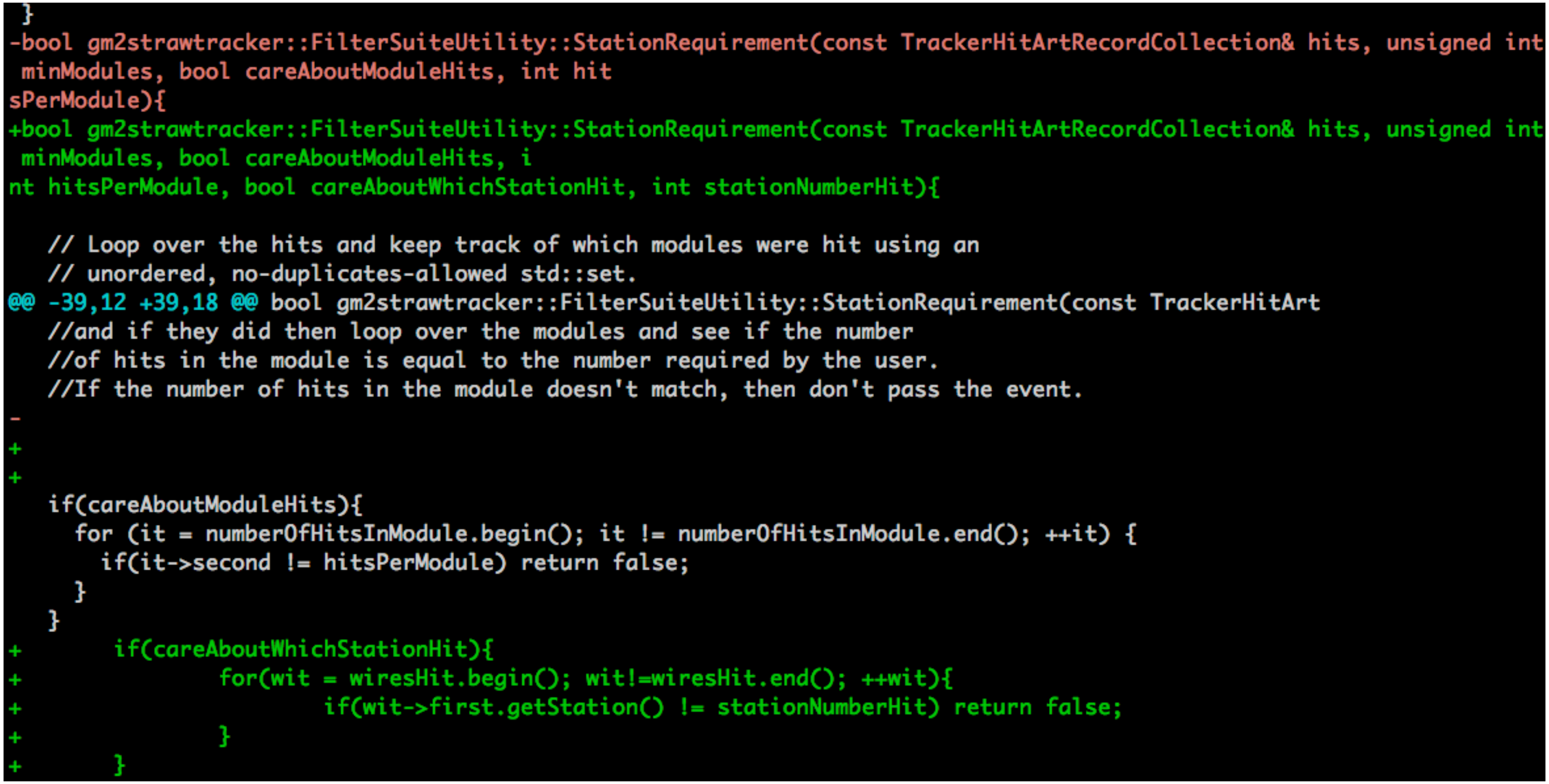
```
<gm2gpvm02.fnal.gov> git status
# On branch feature/strawtracker
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   StationRequirement_modul
#       modified:   find-and-make-golden-
#       modified:   util/FilterSuiteUt
#       modified:   util/FilterSuit
#
# Changes not staged for commi
#   (use "git add <file>..                          mmitted)
#   (use "git checkout                             ges in working directory)
#
#       modified:                         cc
#       modifie
#       modi
#       mo
#
# Unt
#   (                          o include in what will be committed)
#
#
```

- git add StationRequirement_module.cc
- git add util/FilterSuiteUtility.cc
- git add util/FilterSuiteUtility.hh
- git add find-and-make-golden-trackerevts.fcl
- git commit

- At this point thrown into vi (default).
- Then enter your commit message, save the file and at that point the commit is put *locally* into your branch.
- But only locally. Nothing on Redmine has any idea you did anything.

Check what you did!

```
git diff util/FilterSuiteUtility.cc
```

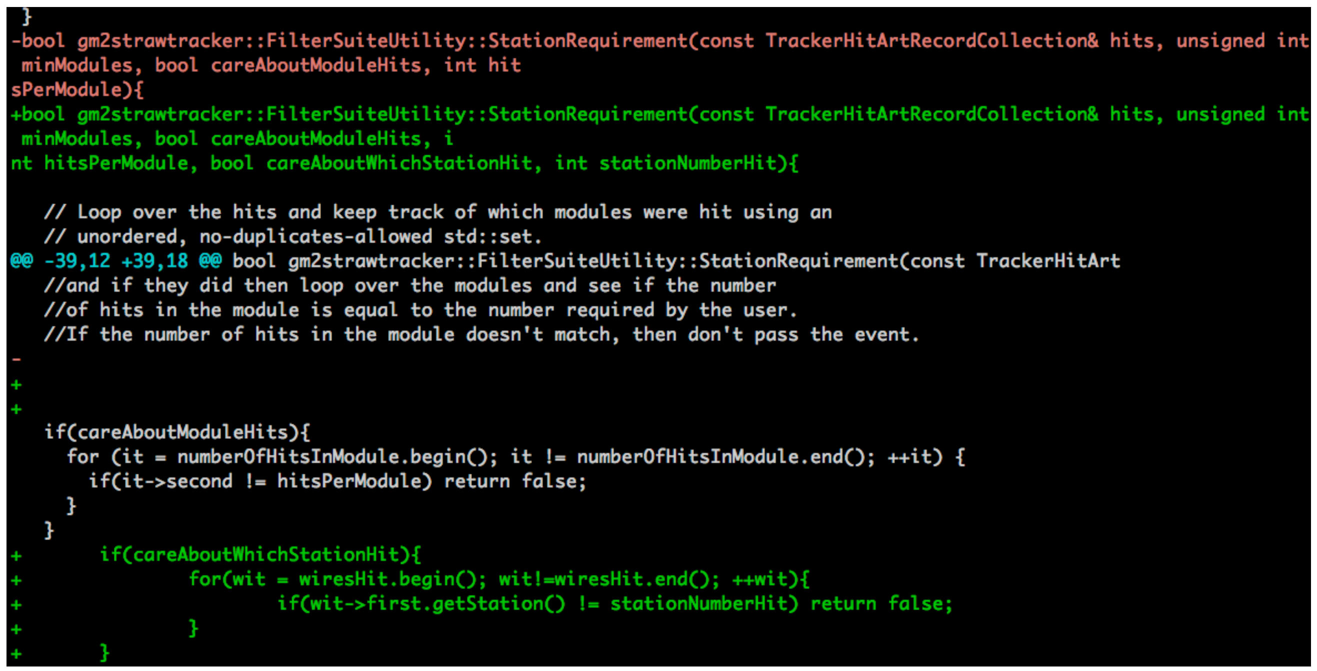


```
 }
-bool gm2strawtracker::FilterSuiteUtility::StationRequirement(const TrackerHitArtRecordCollection& hits, unsigned int
 minModules, bool careAboutModuleHits, int hit
sPerModule){
+bool gm2strawtracker::FilterSuiteUtility::StationRequirement(const TrackerHitArtRecordCollection& hits, unsigned int
 minModules, bool careAboutModuleHits, i
nt hitsPerModule, bool careAboutWhichStationHit, int stationNumberHit){

   // Loop over the hits and keep track of which modules were hit using an
   // unordered, no-duplicates-allowed std::set.
@@ -39,12 +39,18 @@ bool gm2strawtracker::FilterSuiteUtility::StationRequirement(const TrackerHitArt
   //and if they did then loop over the modules and see if the number
   //of hits in the module is equal to the number required by the user.
   //If the number of hits in the module doesn't match, then don't pass the event.
-
+
+

   if(careAboutModuleHits){
     for (it = numberOfHitsInModule.begin(); it != numberOfHitsInModule.end(); ++it) {
       if(it->second != hitsPerModule) return false;
     }
   }
+       if(careAboutWhichStationHit){
+               for(wit = wiresHit.begin(); wit!=wiresHit.end(); ++wit){
+                       if(wit->first.getStation() != stationNumberHit) return false;
+               }
+       }
```

Check what you did!

git diff util/FilterSuiteUtility.cc

- I'm always worried I have put something into the file that I don't really need to push to the repository
- git diff allows you to see exactly what lines changed
  - Making the option for color in your .gitconfig file really helps here.
  - Then before you add the file or commit it you can just make sure you are putting what you think you are putting in there.



```
 }
-bool gm2strawtracker::FilterSuiteUtility::StationRequirement(const TrackerHitArtRecordCollection& hits, unsigned int
 minModules, bool careAboutModuleHits, int hit
sPerModule){
+bool gm2strawtracker::FilterSuiteUtility::StationRequirement(const TrackerHitArtRecordCollection& hits, unsigned int
 minModules, bool careAboutModuleHits, i
nt hitsPerModule, bool careAboutWhichStationHit, int stationNumberHit){

   // Loop over the hits and keep track of which modules were hit using an
   // unordered, no-duplicates-allowed std::set.
@@ -39,12 +39,18 @@ bool gm2strawtracker::FilterSuiteUtility::StationRequirement(const TrackerHitArt
   //and if they did then loop over the modules and see if the number
   //of hits in the module is equal to the number required by the user.
   //If the number of hits in the module doesn't match, then don't pass the event.
-
+
+
   if(careAboutModuleHits){
     for (it = numberOfHitsInModule.begin(); it != numberOfHitsInModule.end(); ++it) {
       if(it->second != hitsPerModule) return false;
     }
   }
+       if(careAboutWhichStationHit){
+               for(wit = wiresHit.begin(); wit!=wiresHit.end(); ++wit){
+                       if(wit->first.getStation() != stationNumberHit) return false;
+               }
+       }
```

Now, How to Really Save

- Like OH NO!!! Your laptop got stolen on the train on your commute home. ALL THAT WORK YOU DID WAS ONLY COMMITTED LOCALLY.
- NOT. GOOD.
- So. Back. It. Up.
    - You push your changes to Redmine.
- Still on your own branch, so you still aren’t screwing people up, you are just backing up your work to the “cloud” (or whatever you want to call it).

How to Push

How to Push

First time on a Feature Branch:

```
git flow feature publish NewDesignTrackerModule
```

Every subsequent time:

```
git pull origin feature/NewDesignTrackerModule
BUILD IT AGAIN (mrb b for us)
```

How to Push

First time on a Feature Branch:

```
git flow feature publish NewDesignTrackerModule
```

Every subsequent time:

```
git pull origin feature/NewDesignTrackerModule
BUILD IT AGAIN (mrb b for us)
```

Are these two steps necessary?

No *(especially if you are the only one using a particular feature branch and/or you see no updates after the pull)*

Is it good practice anyways?

YES! Pull before you Push.

How to Push

First time on a Feature Branch:

```
git flow feature publish NewDesignTrackerModule
```

Every subsequent time:

```
git pull origin feature/NewDesignTrackerModule
BUILD IT AGAIN (mrb b for us)
```

Are these two steps necessary?

NO *(especially if you are the only one using a particular feature branch and/or you see no updates after the pull)*

Is it good practice anyways?

YES! Pull before you Push.

```
git push origin feature/NewDesignTrackerModule
```

Oh Damn I Screwed Up

- If you decided that you don't actually want ANY of the changes you've made to a file. You can easily just get whatever you had in version control.
- Instead of adding it, check it back out:

  ```
  git checkout fcl/ProductionMuPlusMuonGasGun.fcl
  ```

- This will take you back to whatever version was committed.
- Alternatively you can go through a file and see what the differences are and just change those lines.
- Commit Early. Commit Often.
  - Like a good Chicagoan.

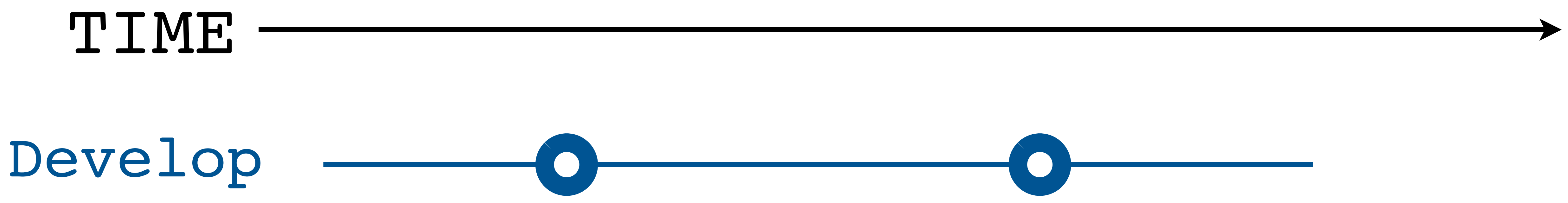
Feature Completion

- Awesome! You have things working on your feature branch
  - The build works
- You want to now pull in the develop branch *onto* your branch
  - In theory people have been committing to develop while you’ve been working and you want to make sure everything works “in harmony” before pushing your stuff out to everyone
    - Maybe someone changed the same file you have?
    - Maybe a change you made breaks something else because you changed a variable name, etc etc.

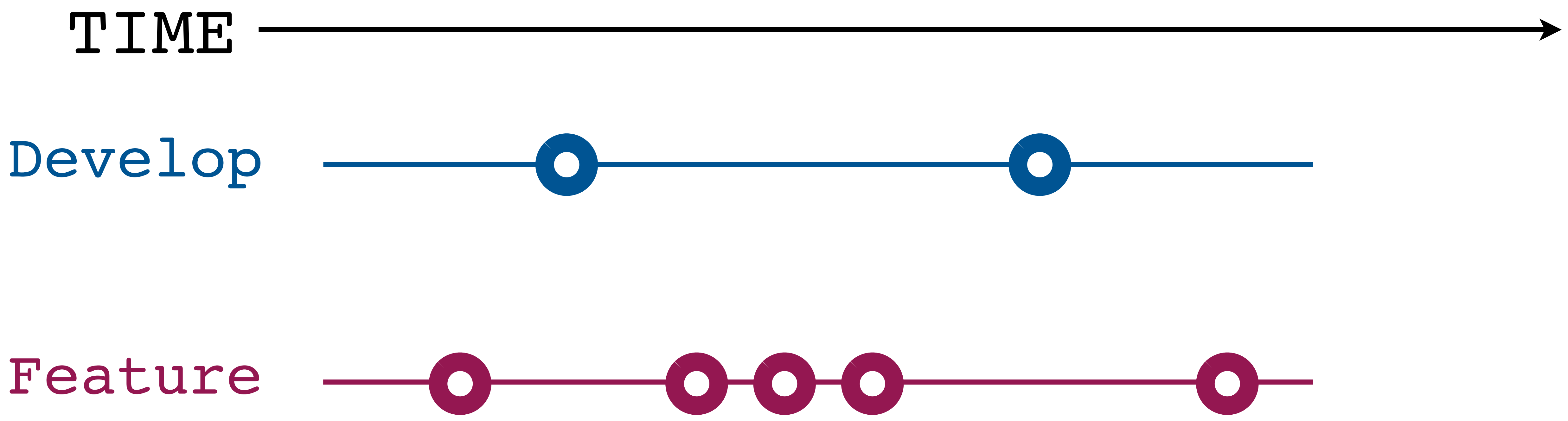
Merging. In Pictures.

Merging. In Pictures.

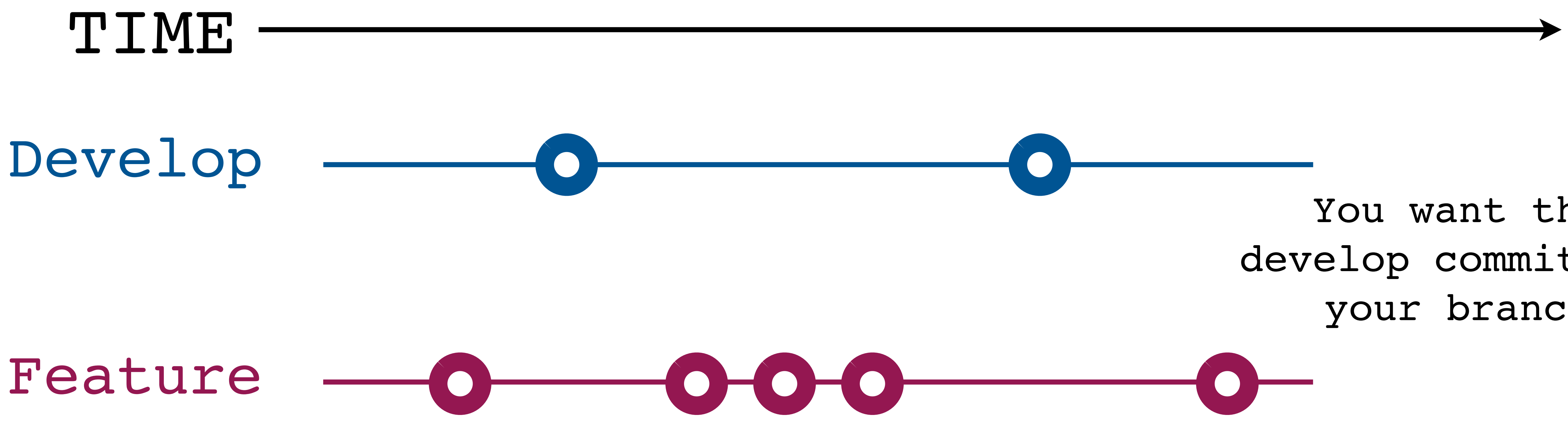
Merging. In Pictures.



Merging. In Pictures.

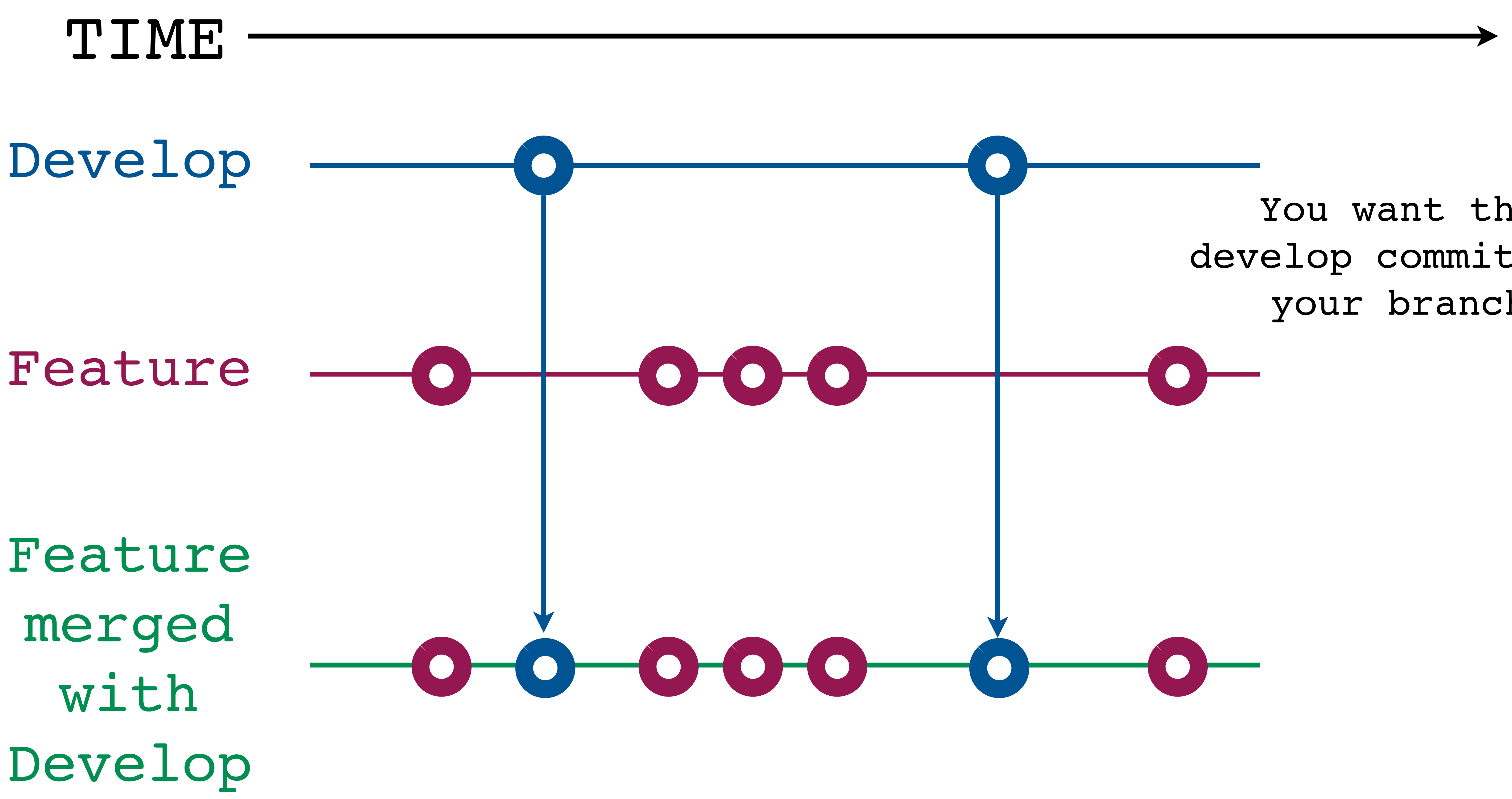


Merging. In Pictures.



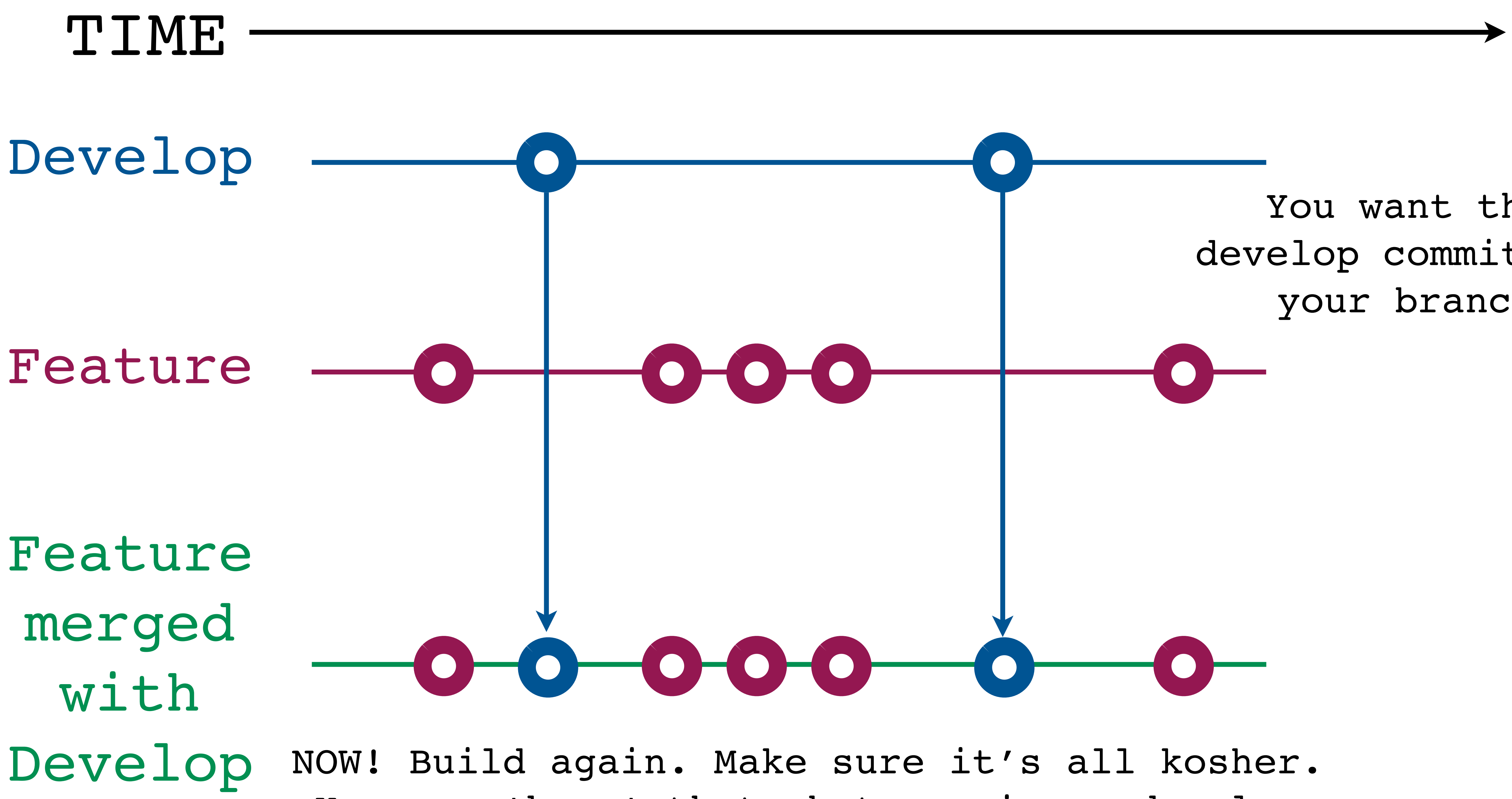
You want the develop commits on your branch

Merging. In Pictures.



You want the develop commits on your branch

Merging. In Pictures.



You want the develop commits on your branch

NOW! Build again. Make sure it's all kosher. You now *know* that whatever is on develop is exactly what you have on your branch, and it works so you can send it back to develop knowing the build works.

git stash

- Sometimes you want to move over to another branch and check things out, but you have uncommitted code.
  - Git will not allow you to move around with “unstashed” changes
  - git stash will stash all your changes as they stand now
    - Includes modified files & new files
  - Do what you want and then you can do “git stash pop” to pop it off the heap and get your updates back
- A useful tool if you forget that you were on develop (for example) and had changed files
  - git stash
  - move to feature branch
  - git stash pop and updates are there for updating the feature branch

Give the Hotness to All

- Before you do the actual moving over to develop
- Send an email to gm2sim alerting people to the changes you've made are going to push to develop shortly
- People might have questions or ask for a code review
  - Which they can do on your feature branch!
  - So things can be vetted and checked BEFORE it invades the develop line.
- If you don't hear anything back from gm2sim follow the steps on the next page.

https://cdcvs.fnal.gov/redmine/projects/g-2/wiki/CodeReviews

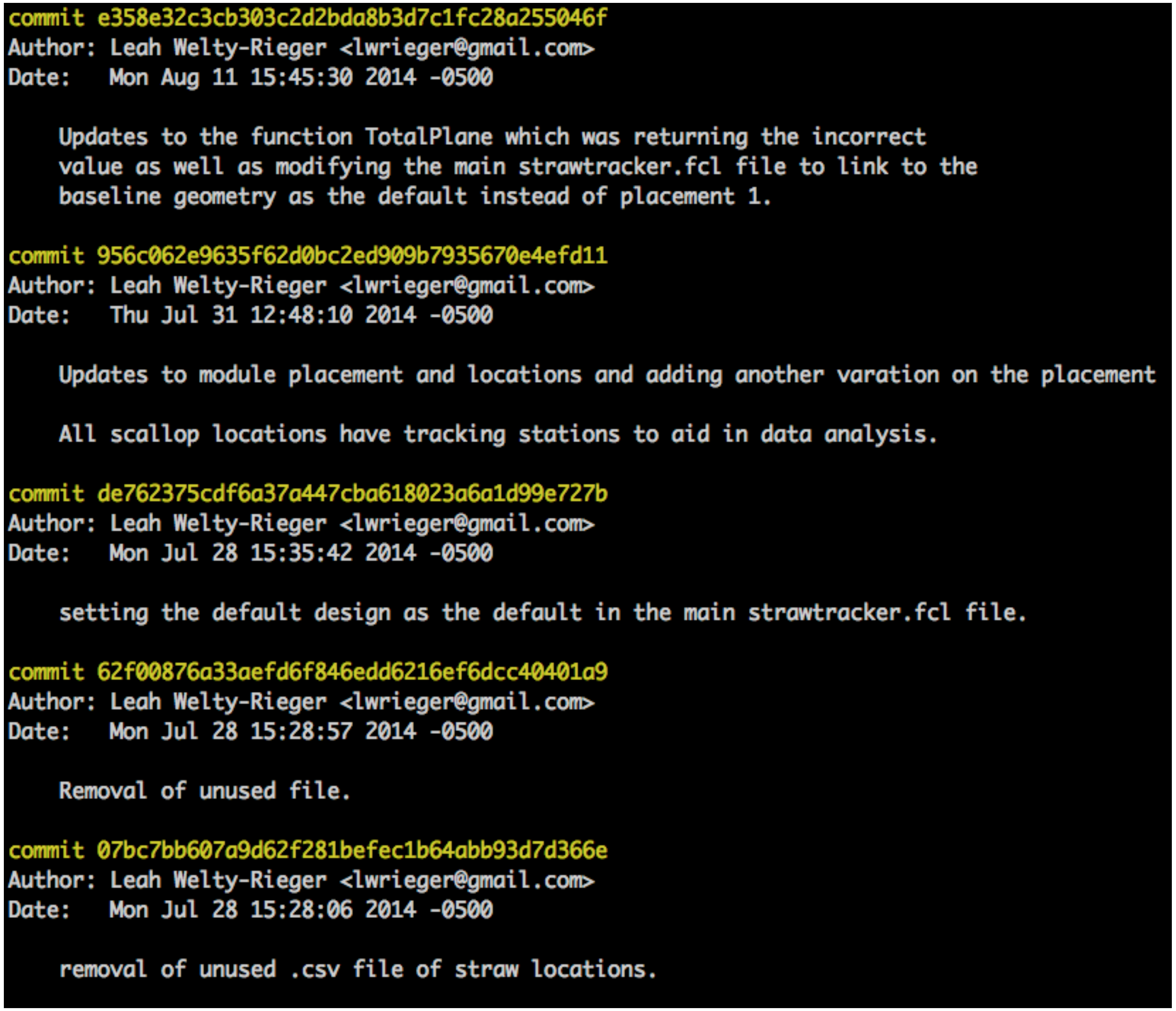
Give the Hotness to All

- Now that your branch is completely up to date with develop you can move it all back to develop (2 Options)
  1. Use git flow: git flow feature finish FeatureName
     - This will merge your branch back with develop AND delete your *local* feature branchOR
  2. git checkout develop, git merge feature/FeatureBranch, git flow feature finish FeatureBranch
- Then git commit
- mrb b
  - Just to be extra extra sure the build works!
- Then git push origin develop
  - I always specify where I am pushing to or pulling from so I know exactly what I'm getting

https://cdcvs.fnal.gov/redmine/projects/g-2/wiki/CodeReviews

git log

- To see the history of the repository (check ins) simply type **git log.**
- If you ever needed to go back to a point in time you would revert back to one of the unique "**hashes**" that identifies a point in time



```
commit e358e32c3cb303c2d2bda8b3d7c1fc28a255046f
Author: Leah Welty-Rieger <lwrieger@gmail.com>
Date:   Mon Aug 11 15:45:30 2014 -0500

    Updates to the function TotalPlane which was returning the incorrect
    value as well as modifying the main strawtracker.fcl file to link to the
    baseline geometry as the default instead of placement 1.

commit 956c062e9635f62d0bc2ed909b7935670e4efd11
Author: Leah Welty-Rieger <lwrieger@gmail.com>
Date:   Thu Jul 31 12:48:10 2014 -0500

    Updates to module placement and locations and adding another varation on the placement

    All scallop locations have tracking stations to aid in data analysis.

commit de762375cdf6a37a447cba618023a6a1d99e727b
Author: Leah Welty-Rieger <lwrieger@gmail.com>
Date:   Mon Jul 28 15:35:42 2014 -0500

    setting the default design as the default in the main strawtracker.fcl file.

commit 62f00876a33aefd6f846edd6216ef6dcc40401a9
Author: Leah Welty-Rieger <lwrieger@gmail.com>
Date:   Mon Jul 28 15:28:57 2014 -0500

    Removal of unused file.

commit 07bc7bb607a9d62f281befec1b64abb93d7d366e
Author: Leah Welty-Rieger <lwrieger@gmail.com>
Date:   Mon Jul 28 15:28:06 2014 -0500

    removal of unused .csv file of straw locations.
```

Conclusions

- git has more steps to get code to people than CVS or SVN
  - But it's a lot more powerful
- Following a *few* steps makes for easier (and more <u>fun</u>) development
- Don't wait too long to come back to develop! The merge will be more painful the longer you wait!
- ASK QUESTIONS.
  - This is a different way of the "normal" physics development, so understandably it might be confusing.
  - But lots of people around to help (Adam, Me, etc)
  - Don't be shy.