SWE 525 Git Version Control Assignment 1

A. Answer following questions briefly:

1. List out the key difference between a centralized version control system and distributed version control system

Centralized version control system keeps a center repository in usually a remote server. It is the one true resource that is blessed. Everybody who works on the project checks out from the center repository to local and makes change to it, then checks in (commits) back to the center repository. If conflicts happen, developer resolves the conflict then check in. Once checked in, the change will be in the center repository therefore blessed by the version control system.

As for the distributed version control system, everybody who works on the project will have a local copy of the entire work, everybody's local repository is as good as anybody else's, that is to say there is no central entity in charge of the work's history, developers do not have to be online while they make changes and have them tracked by version control system. The way developers contribute to the project is to make merges from one repository to another.

2. List down any two centralized version control system and 2 distributed version control system

Centralized VCS
- Subversion
- CVS

Distributed VCS
- Git
- Mercurial

3. What are the advantages of git VCS over other VCS

- Git allows developer to create new experimental branches and tweak the code without interfering with the main code of the project.
- Git allows developer to work offline and still be able to track all the changes and history, later developer can commit the changes when he or she gains the connection to main repository
- Git operations are fast, mainly because they are performed on local repository copy.
- Git also uses space, a typical Git repository is smaller than for instance one using SubVersion.
- Git allows you to ignore certain files in the local repository directories using a file named .gitignore.

4. What are the different states of a file in the Git VCS

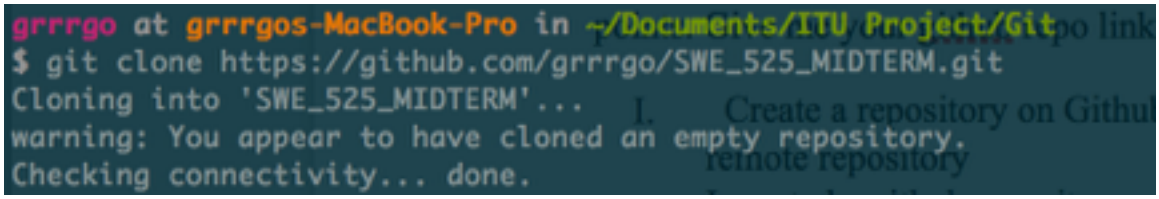
Modified: developer has changed the file but have not added it to the staging area yet.

Staged: developer has marked a modified file in its current version to go into the next commit snapshot.

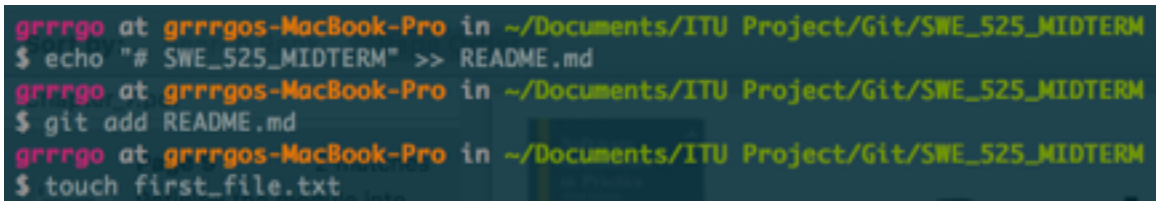
Committed: the data is stored in the local database.

B. GIT REMOTE REPOSITORIES: Perform following tasks and explain how you performed each operation. Draw a flow diagram as you progress through the steps. Add all git commands you used and push the repository in your github. Add your github public repository (for these following tasks) link with the homework.

1. Clone an existing repository on Github created during course and configure your local repo to point to the remote repositor



2. Perform some operation like add, remove, modify and finally push your changes to the remote repository



```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git add .
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git remove first_file.txt
git: 'remove' is not a git command. See 'git --help'.

Did you mean this?
        remote
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git rm first_file.txt
error: the following file has changes staged in the index:
    first_file.txt
(use --cached to keep the file, or -f to force removal)
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git rm first_file.txt  -cached
error: did you mean `--cached` (with two dashes ?)
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git rm first_file.txt  --cached
rm 'first_file.txt'
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ touch second_file.txt
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ vim second_file.txt
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git add .
```

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.md
        new file:   second_file.txt

grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git commit -m "init commit"
[master (root-commit) 9b2fe98] init commit
 Committer: grrrgo <grrrgo@grrrgos-MacBook-Pro.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 2 files changed, 2 insertions(+)
 create mode 100644 README.md
 create mode 100644 second_file.txt
```

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.md
        new file:   first_file.txt
        new file:   second_file.txt

grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM
$ git rm -f first_file.txt
rm 'first_file.txt'
```

3. Pull the latest changes from the repository to get the updates from others in to your local repo and merge the changes

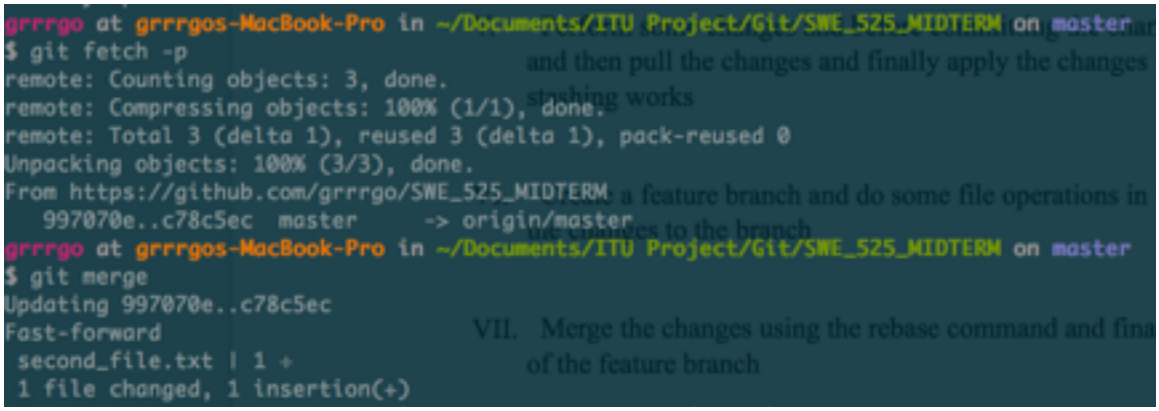
From the current local repo:

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM on master
$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/grrrgo/SWE_525_MIDTERM
   fb773bf..997070e  master     -> origin/master
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM on master
$ git pull
Updating fb773bf..997070e
Fast-forward
 others.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 others.txt
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM on master
$ git merge master
Already up-to-date.
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM on master
$ _
```

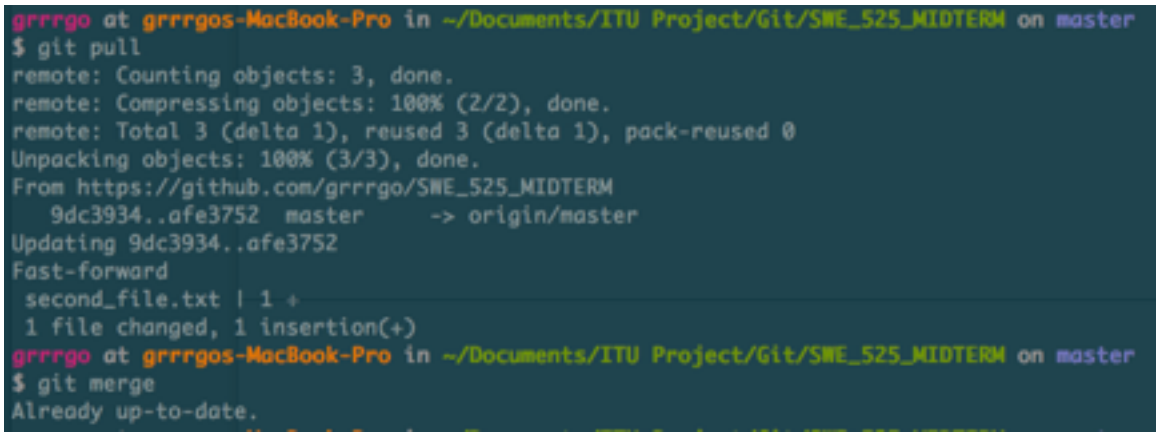
4. Try fetching the changes and perform the merge to get the difference between the pull and the merge command

Both are performed after adding some change to the repository

merge



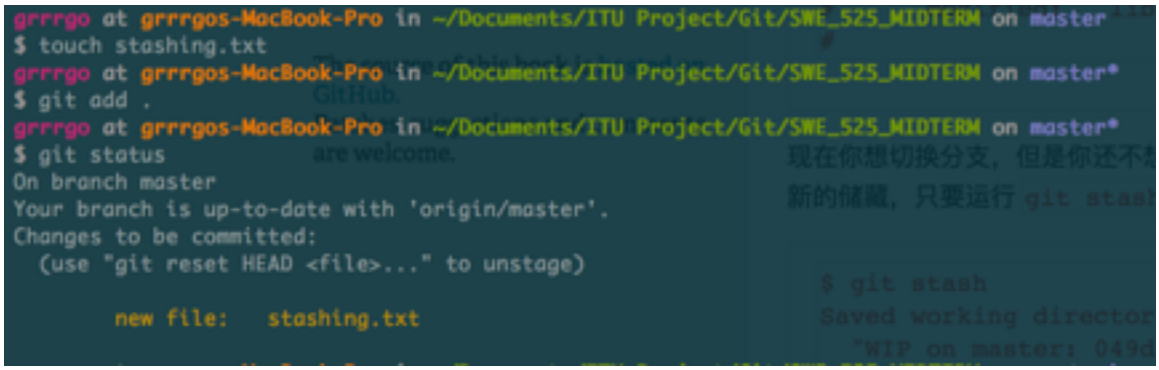
pull

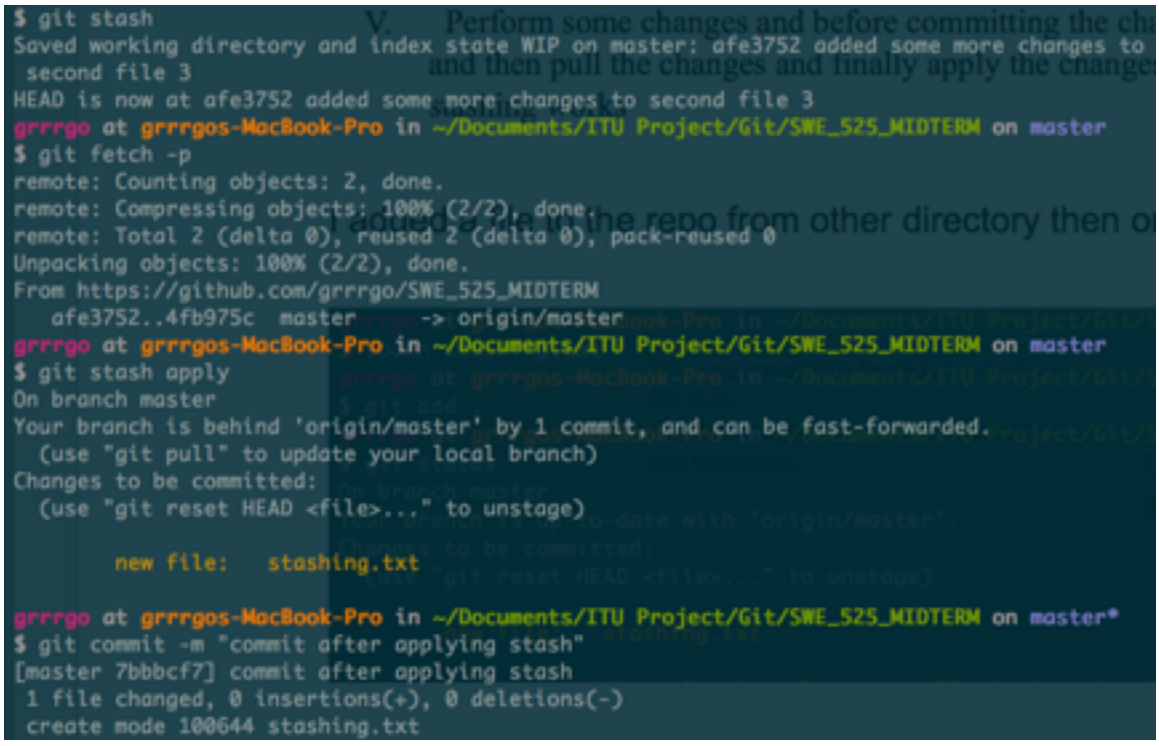


`git pull` does a `git fetch` followed by a `git merge`.

5. Perform some changes and before committing the changes, stash your changes and then pull the changes and finally apply the changes to understand how stashing works

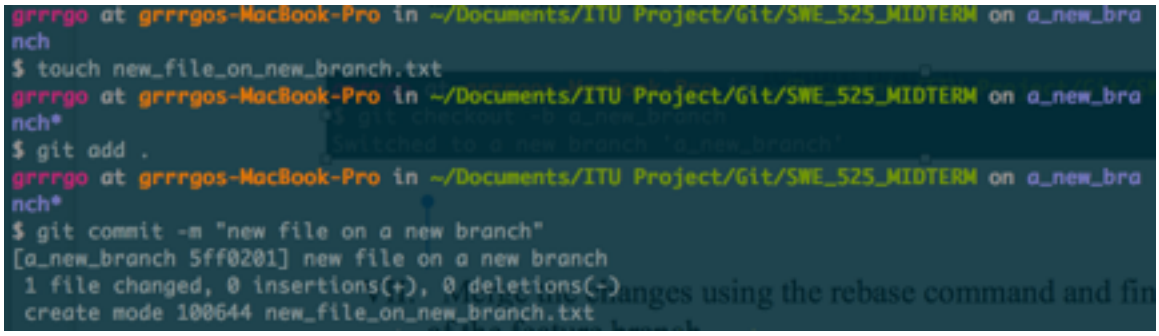
I added a file to the repo from other directory then on the current repo:



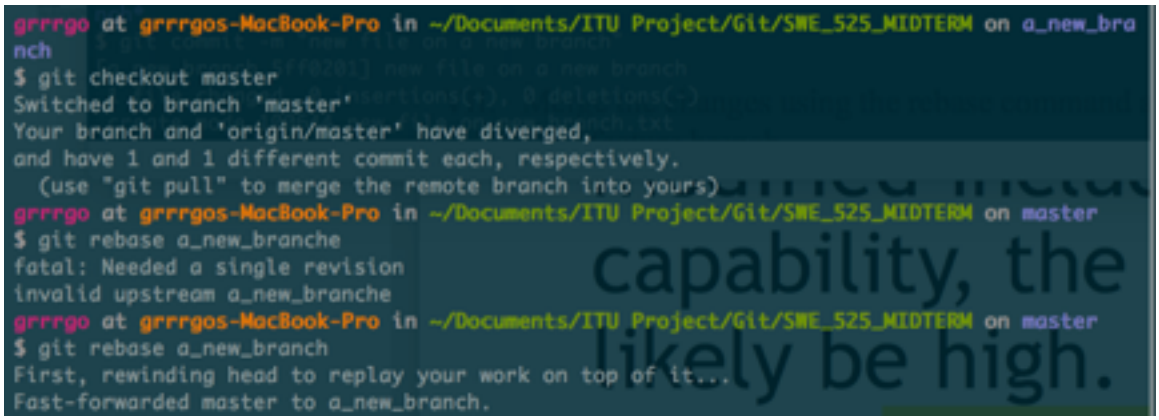


Stashing clears out the current work space as if no changes are in it, later I can do a stash apply to bring back the stashed changes.

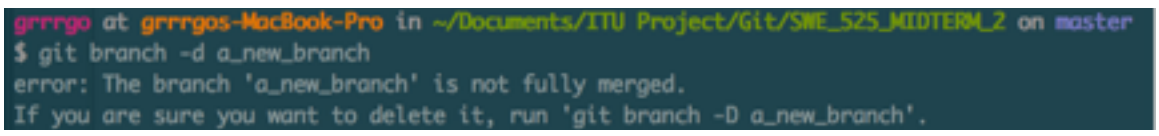
6. Create a feature branch and do some file operations in the branch and commit the changes to the branch



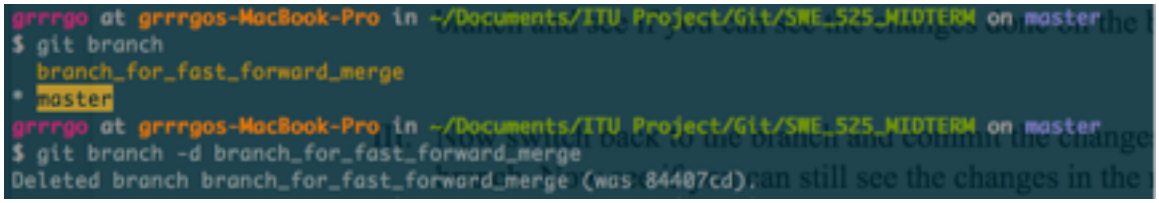
7. Merge the changes using the rebase command and finally perform a safe deletion of the feature branch

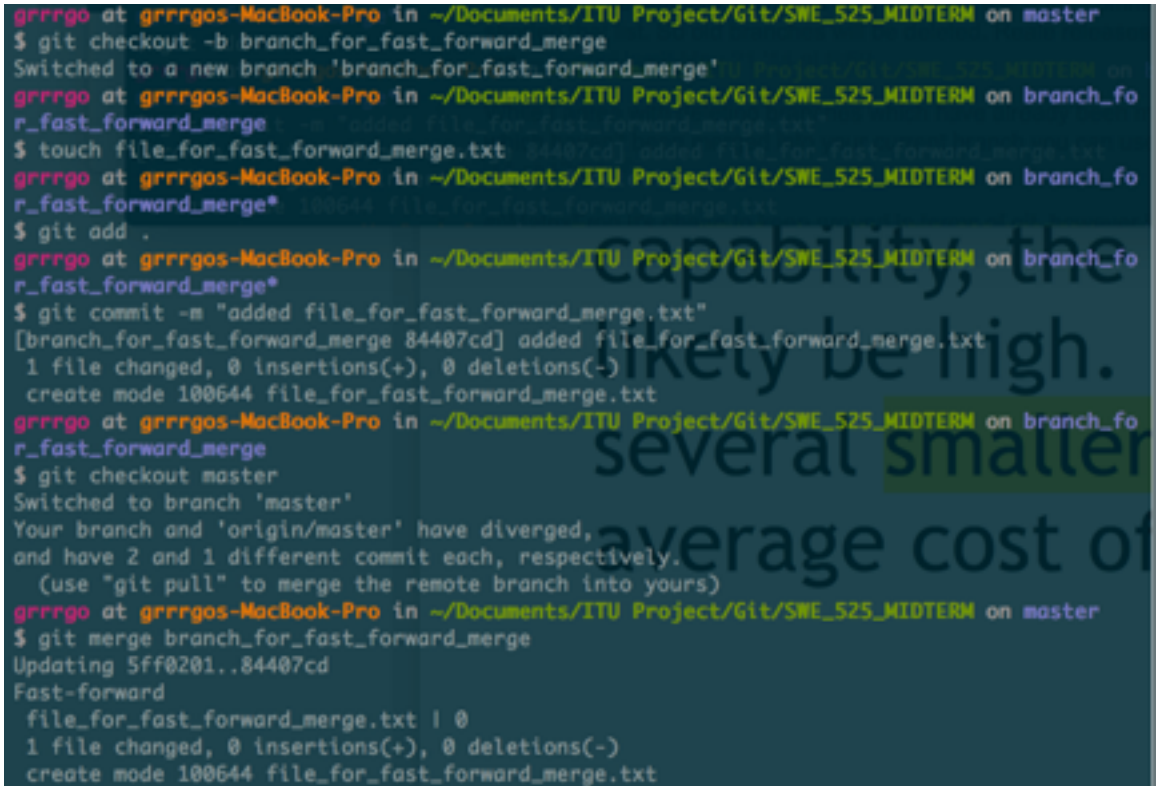


I can safely remove a branch with `git branch -d yourbranch`. If it contains unmerged changes, git will tell me and won't delete it. Eg:



8. Create another feature branch and this time after committing the changes to the feature branch, merge the changes using fast forward merge and then delete the feature branch





C. GIT BRANCHING AND MERGING: Perform following tasks and explain how you performed each operation. Draw a flow diagram as you progress through the steps. Add all git commands you used and push the repository in your github. Add your github public repository (for these following tasks) link with the homework.
(https://github.com/grrrgo/SWE_525_MIDTERM_2)

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git
$ mkdir SWE_525_MIDTERM_2
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git
$ cd SWE_525_MIDTERM_2/
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2
$ echo "# SWE_525_MIDTERM_2" >> README.md
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2
$ git init
Initialized empty Git repository in /Users/grrrgo/Documents/ITU Project/Git/SWE_525_MIDTERM_2/.git/
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2
$ git add README.md
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2
$ git commit -m "first commit"
[master (root-commit) 64ef55b] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master
$ git remote add origin https://github.com/grrrgo/SWE_525_MIDTERM_2.git
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 239 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/grrrgo/SWE_525_MIDTERM_2.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

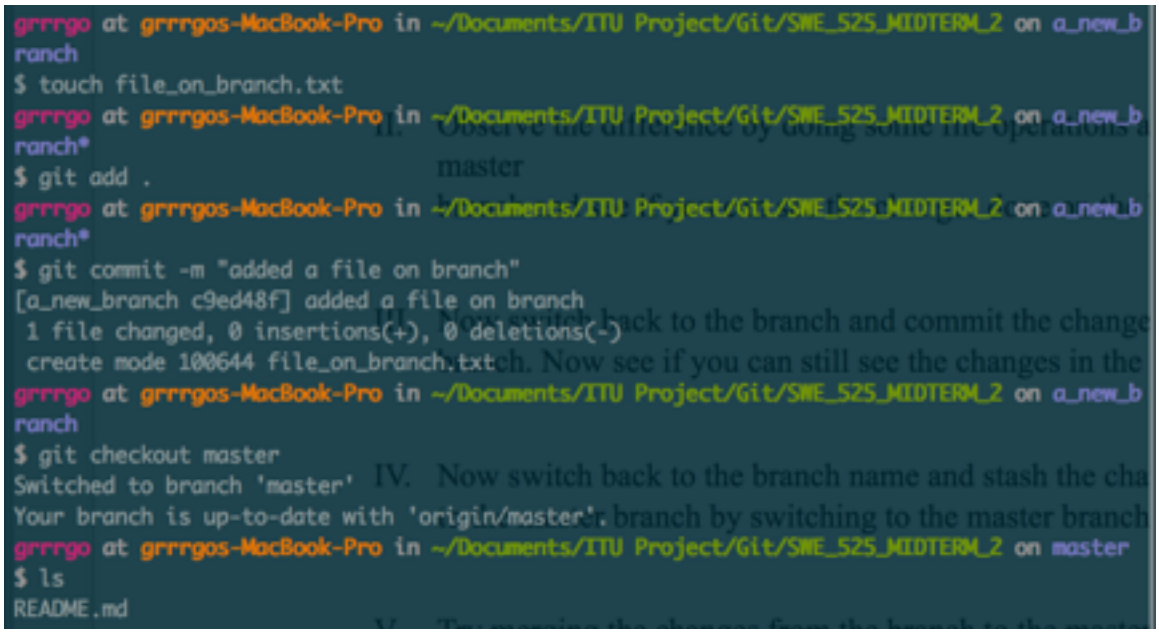
1. Create a local branch using git checkout -b branchname command

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master
$ git checkout -b a_new_branch
Switched to a new branch 'a_new_branch'
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on a_new_branch
$ git branch
* a_new_branch
  master
```

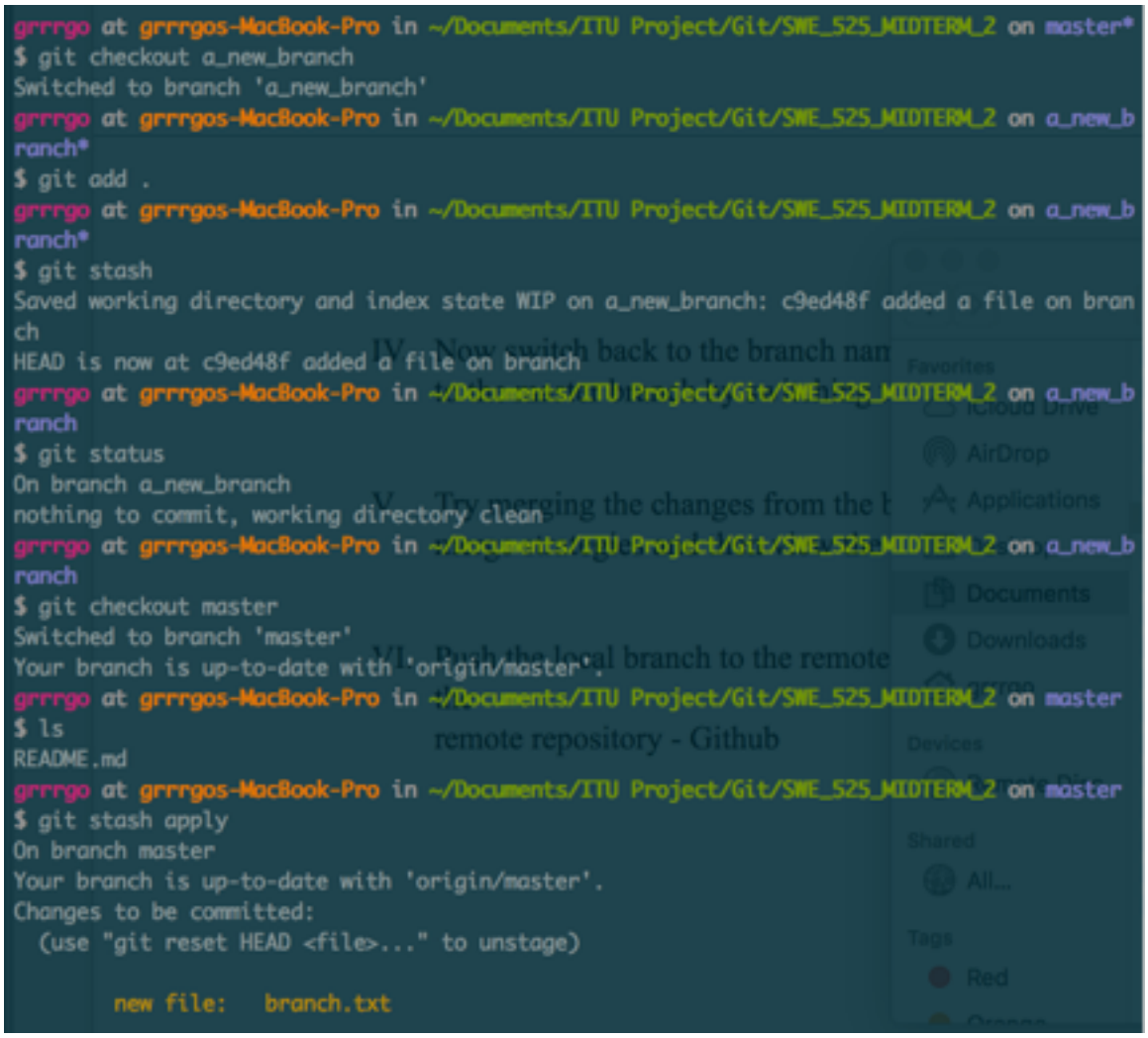
2. Observe the difference by doing some file operations and switch back to the master branch and see if you can see the changes done on the branch

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master
$ git checkout a_new_branch
Switched to branch 'a_new_branch'
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on a_new_branch
$ touch branch.txt
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on a_new_branch*
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master*
$ ls
README.md  branch.txt
```

3. Now switch back to the branch and commit the changes and switch to master branch. Now see if you can still see the changes in the master branch

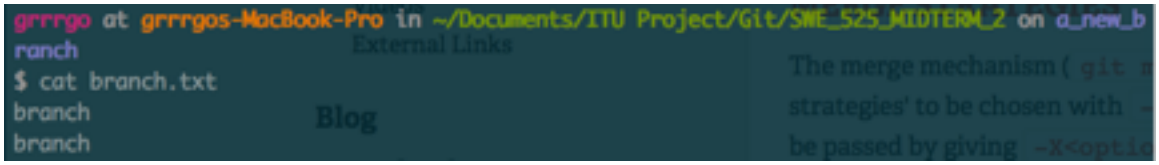


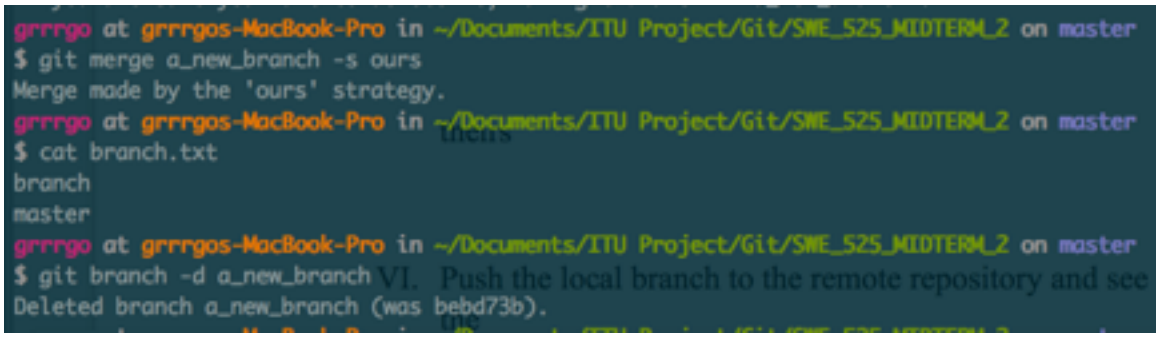
4. Now switch back to the branch name and stash the changes and apply the changes to the master branch by switching to the master branch



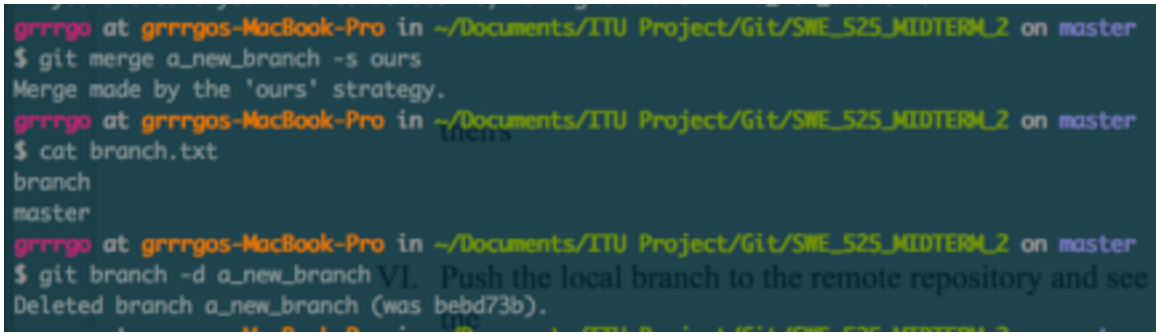
5. Try merging the changes from the branch to the master branch using all the three merge strategies and then view the git log

Ours:

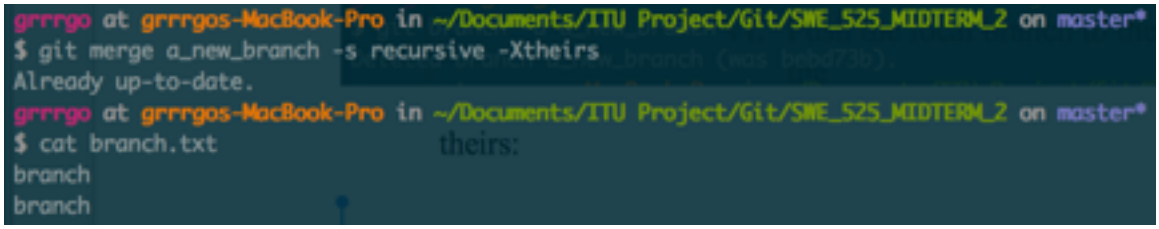




Resolve:



recursive -Xtheirs:



```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master*
$ git status
On branch master
Your branch is ahead of 'origin/master' by 8 commits.
  (use "git push" to publish your local commits)
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:   branch.txt

no changes added to commit (use "git add" and/or "git commit -a")
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master*
$ git branch -d a_new_branch
Deleted branch a_new_branch (was 261a70e).
```

6. Push the local branch to the remote repository and see if the branch is present on the remote repository – Github

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master*
$ git branch
* master
grrrgo at grrrgos-MacBook-Pro in ~/Documents/ITU Project/Git/SWE_525_MIDTERM_2 on master*
$ git push origin master
Counting objects: 18, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (18/18), 1.72 KiB | 0 bytes/s, done.
Total 18 (delta 5), reused 0 (delta 0)
```

D: Review following "MERGE with CONFLICT" scenario. Complete the exercise as requested below:

Purpose:
Learn how to merge when there are code conflicts
Learn how to interact with the remote repository

Preparation
Watch following video: https://vimeo.com/138418055
You may want to download the script used for video**. (copied at the end of this assignment)**

Exercise
For this exercise, you should experiment with the merging files that have conflicts. You may use the code below or any code of your choice.

public class TheMotivator {

```
	public void feedback(int score) {
		if (score == 100)
			System.out.println("You're awesome");
		else if (score > 90)
			System.out.println("That's great");
		else if (score > 60)
			System.out.println("That's good ");
		else
			System.out.println("Well, what can I say?");
	}
	public static void main(String[] args) {
		TheMotivator tm = new TheMotivator();
		tm.feedback(60);
	}
}
```

Specific Requirements

This exercise is worth 5 points. Turn in a git log that contains at least two different branches that have been merged. NOTE: The point of this is to get comfortable with git, so you do not need to follow the demo exactly. The log should show some reasonable amount of branching and merging. You should do some merges with and without conflicts, although we won't be able to tell this from the log. Show with diagram also.

Remember that to create log file you do:

git log --pretty=format:"%s" --graph > mylog.txt

I would suggest you spend at least half an hour on this task (more if this is all new to you).

Submit
Explain steps followed, draw the git flow diagram and
Submit your .txt log file “mylog.txt” with you assignment.

Hint: Script used in the video: **https://vimeo.com/138418055**

Git Merging Demo Steps

Create SomeClass
Add method fnOne [syso: I couldn't repair your brakes, so I made your

horn louder]
Run
git bash
cd to directory
git init
git status
git add *.java
git status
git commit -m "Initial"
chg fnOne to thoughtForTheDay
git status
git commit -a -m "Refactored fnOne"
git checkout -b addEvents
add upcomingEvents [syso: Party at Jane's house tomorrow]
git commit -a -m "Added events"
add header to upcomingEvents
try: git checkout master, see error
git commit -a -m "Refined events"
now: git checkout master
revise thought for the day: On the other hand, you have different fingers
git commit -a -m "New thought for the day"
git branch --no-merged (see addEvent)
git merge addEvents [success! different parts of the file]
git branch --no-merged
git branch --merged
git checkout -b moreEvents
Modify program:
variable:

```
	private ArrayList<String> events = new ArrayList<String>();
```

new method:

```
	public void createEvents() {
		events.add("We're going to a movie on Saturday");
		events.add("Study session on Sunday - Jim's house");
	}
```

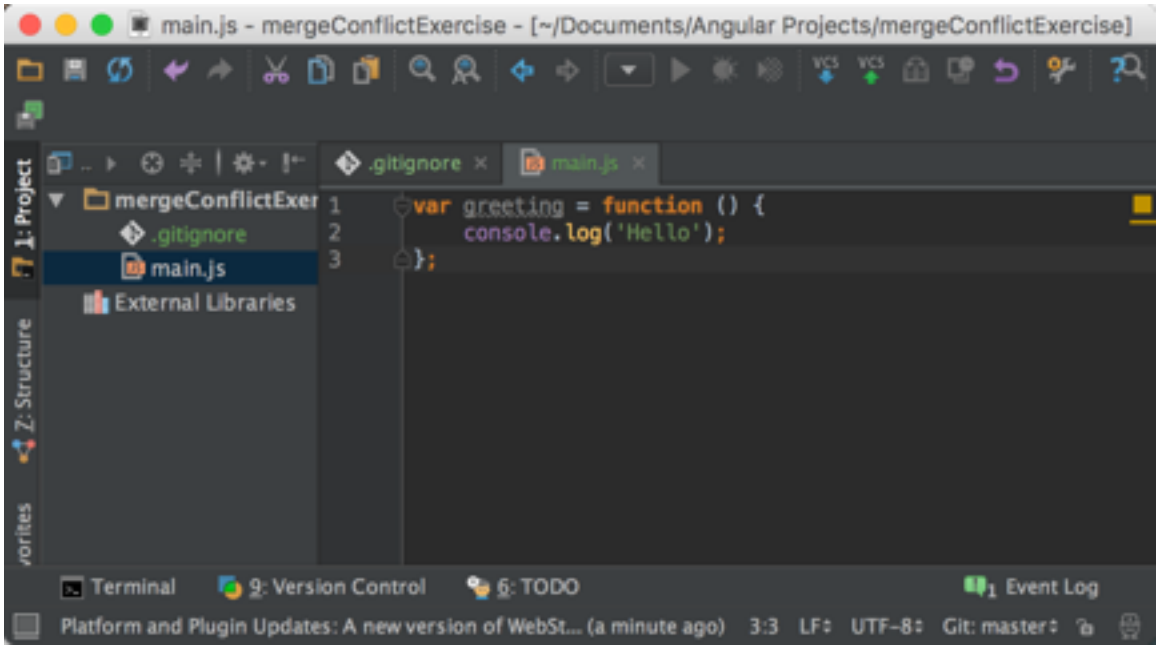
change method:

```
	public void upcomingEvents() {
		System.out.println("Upcoming Events");
		for (String event : events)
			System.out.println(event);
	}
```

call in main:

```
			sc.createEvents();
```

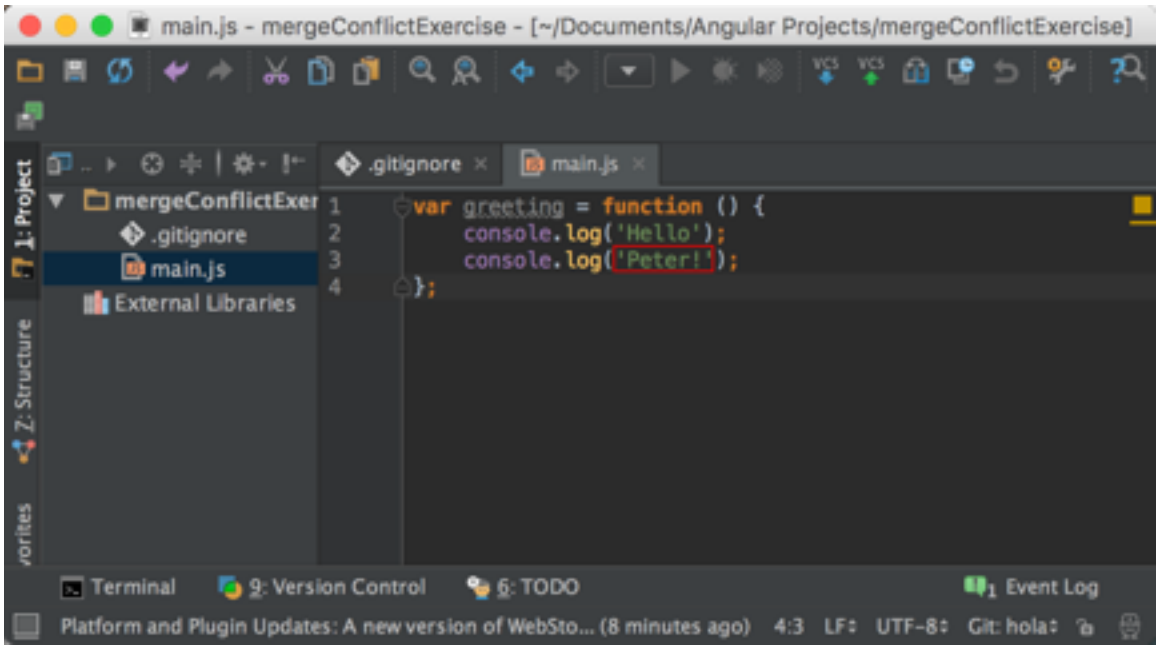
```
git commit -a -m "Add multiple events"
git checkout master
modify upcomingEvents ["Dinner at Katie's on Friday"]
add SomeClass sc = new SomeClass() to main
git commit -a -m "Different event"
git merge moreEvents [ conflicts! need to resolve]
in Editor, notice the lines with issues, fix!
>> how?  remove lines from head, remove lines with === and <<<
>> in general? first decide which to keep, make these kinds of
>> changes.
git branch --no-merged
git merge [ won't let you yet!]
git commit -a -m "Merged event handling"
git branch --no-merged
git branch --merged
git checkout moreEvents
modify: Upcoming Events - Please join us!
git commit -a -m "More friendly events"
git checkout master
press up-arrow, git branch --no-merged
git merge moreEvents [success! it's only a conflict if 2 changes]
git log
git log -p -2
git log --pretty=oneline
git log --pretty=format:"%s" --graph
git log --pretty=format:"%s" --graph > mylog.txt
git config --global alias.gr 'log --pretty=format:"%s" --graph'
```




```
grrrgo at grrrgos-MacBook-Pro in ~/Documents
$ cd Angular\ Projects/mergeConflictExercise/
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise
$ git init
Initialized empty Git repository in /Users/grrrgo/Documents/Angular Projects/mergeConflictExercise/.git/
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise
$ ls
script.js
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   main.js

grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise
$ git commit -m "init commit"
[master (root-commit) 2de66a1] init commit
 2 files changed, 4 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 main.js
```

First commit



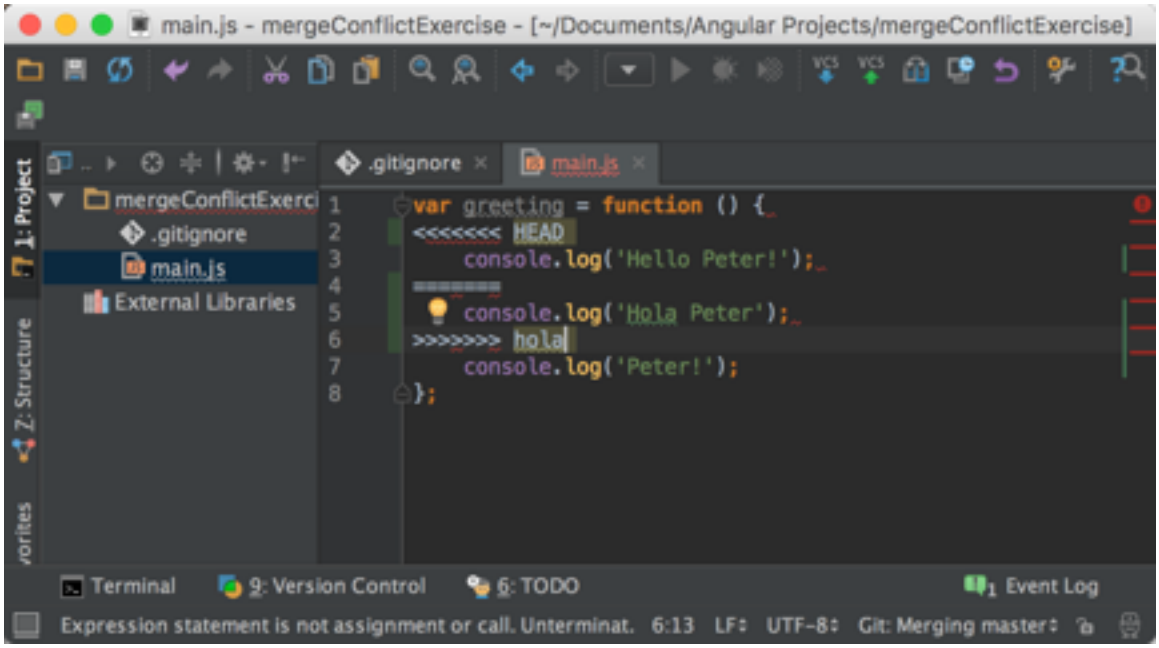

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola
$ git status
On branch hola
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   main.js

no changes added to commit (use "git add" and/or "git commit -a")
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola*
$ git add .
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola*
$ git commit -m "add peter"
[hola 6e34703] add peter
 1 file changed, 1 insertion(+)
```

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola
$ git checkout master
Switched to branch 'master'
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git merge hola
Updating 2de66a1..6e34703
Fast-forward
 main.js | 1 +
 1 file changed, 1 insertion(+)
```

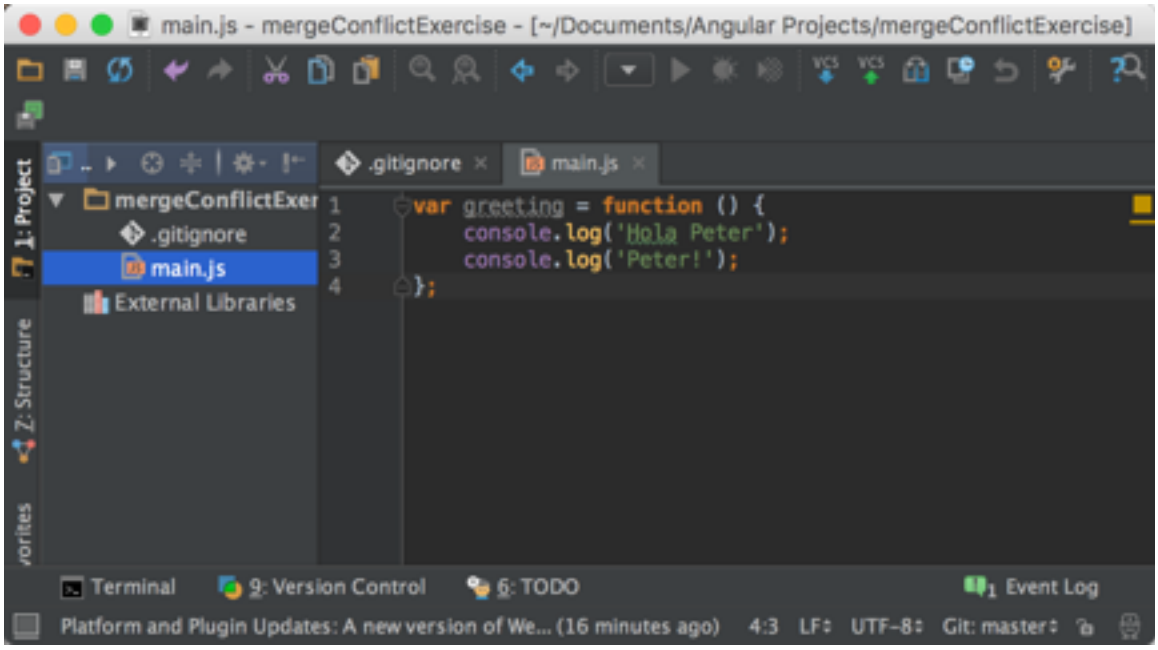
Merge without conflict

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git add .
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master*
$ git commit -m "greet peter"
[master 522976c] greet peter
 1 file changed, 1 insertion(+), 1 deletion(-)
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git checkout hola
Switched to branch 'hola'
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola
$ git add .
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola*
$ git commit -m "greet peter in Spanish"
[hola 9f726ee] greet peter in Spanish
 1 file changed, 1 insertion(+), 1 deletion(-)
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on hola
$ git checkout master
Switched to branch 'master'
```

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git merge hola
Auto-merging main.js
CONFLICT (content): Merge conflict in main.js
Automatic merge failed; fix conflicts and then commit the result.
```




```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master*
$ git add main.js
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master*
$ git commit -m "resolve conflict"
[master 8490b3d] resolve conflict
```




```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master*
$ git add main.js
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master*
$ git commit -m "resolve conflict"
[master 8490b3d] resolve conflict
```

```
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git branch --no-merged
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git log --pretty=format:"%s" --graph
*   resolve conflict
|\
| * greet peter in Spanish
* | greet peter
|/
* add peter
* init commit
grrrgo at grrrgos-MacBook-Pro in ~/Documents/Angular Projects/mergeConflictExercise on master
$ git log --pretty=format:"%s" --graph > mylog.txt
```