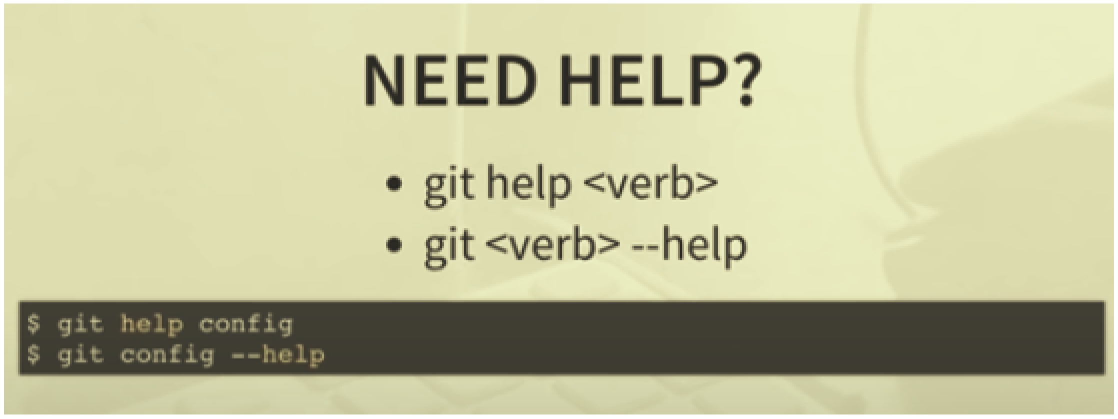
Git Command Line Fundamentals

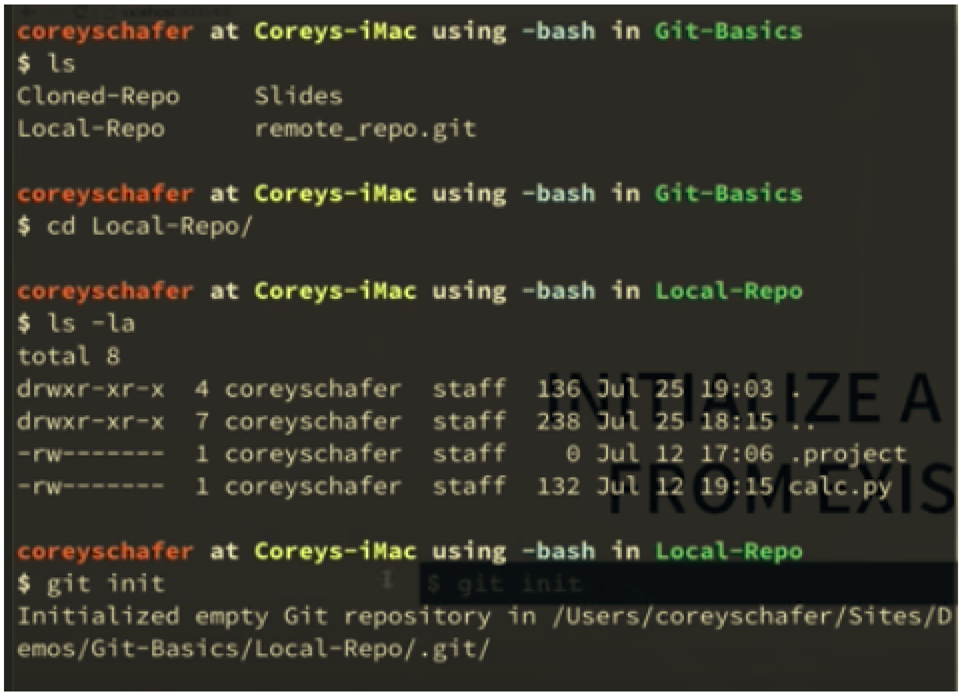


NEED HELP?

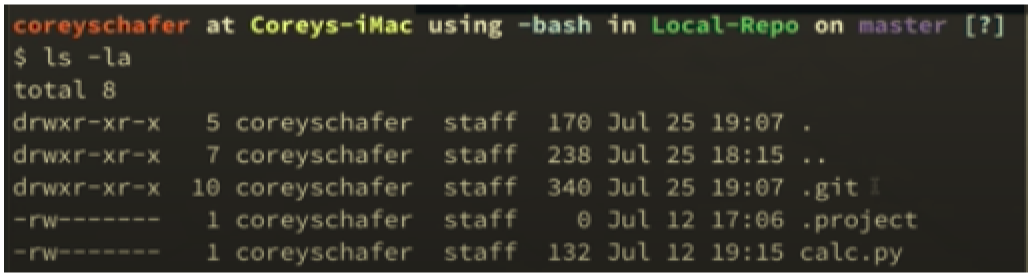
- git help <verb>
- git <verb> --help

```
$ git help config
$ git config --help
```

- Code that we want to start tracking is in local repo directory, so we navigate to it
- We list all of the files in the directory
- To begin tracking this code with git we do - git init
- The git init command will place a .git directory that contains everything that is related to our repository

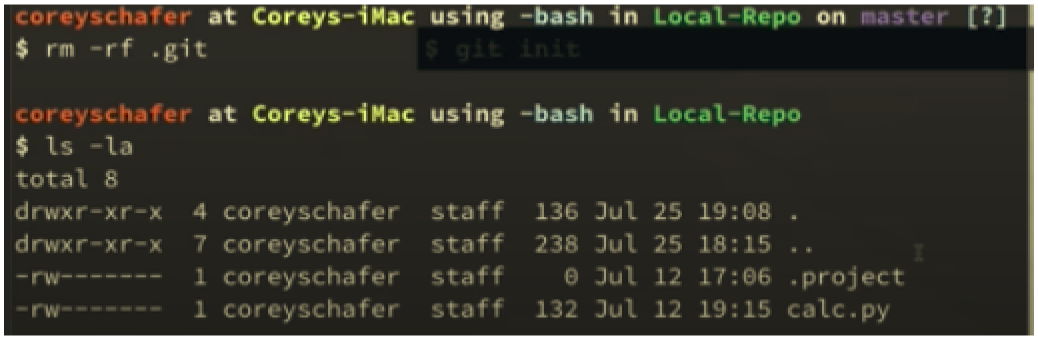


```
coreyschafer at Coreys-iMac using -bash in Git-Basics
$ ls
Cloned-Repo     Slides
Local-Repo      remote_repo.git

coreyschafer at Coreys-iMac using -bash in Git-Basics
$ cd Local-Repo/

coreyschafer at Coreys-iMac using -bash in Local-Repo
$ ls -la
total 8
drwxr-xr-x  4 coreyschafer  staff  136 Jul 25 19:03 .
drwxr-xr-x  7 coreyschafer  staff  238 Jul 25 18:15 ..
-rw-------  1 coreyschafer  staff    0 Jul 12 17:06 .project
-rw-------  1 coreyschafer  staff  132 Jul 12 19:15 calc.py

coreyschafer at Coreys-iMac using -bash in Local-Repo
$ git init
Initialized empty Git repository in /Users/coreyschafer/Sites/D
emos/Git-Basics/Local-Repo/.git/
```



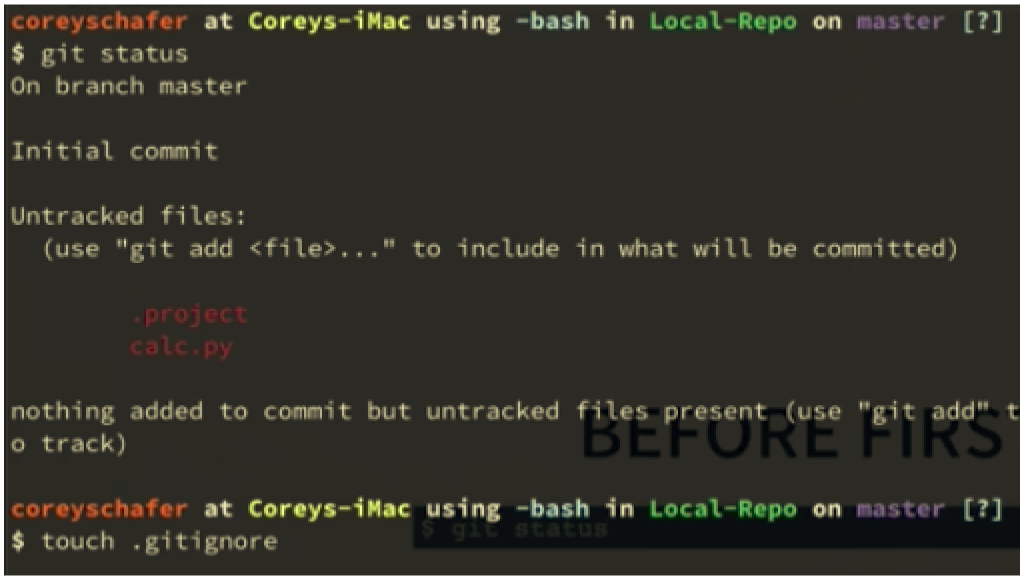
```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ ls -la
total 8
drwxr-xr-x   5 coreyschafer  staff  170 Jul 25 19:07 .
drwxr-xr-x   7 coreyschafer  staff  238 Jul 25 18:15 ..
drwxr-xr-x  10 coreyschafer  staff  340 Jul 25 19:07 .git
-rw-------   1 coreyschafer  staff    0 Jul 12 17:06 .project
-rw-------   1 coreyschafer  staff  132 Jul 12 19:15 calc.py
```

- If we ever want to remove the .git directory we use the following

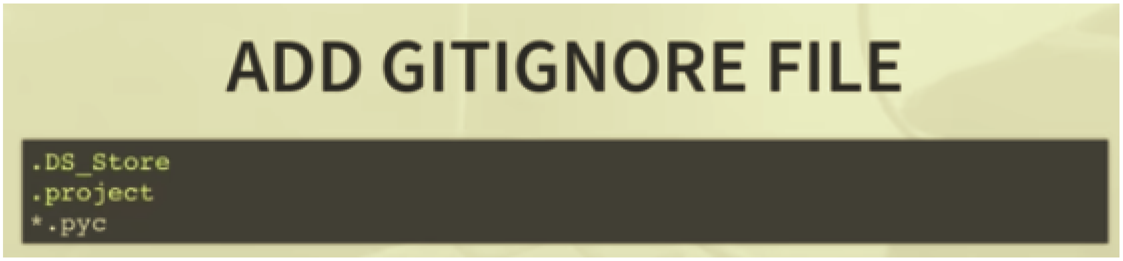


```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ rm -rf .git

coreyschafer at Coreys-iMac using -bash in Local-Repo
$ ls -la
total 8
drwxr-xr-x  4 coreyschafer  staff  136 Jul 25 19:08 .
drwxr-xr-x  7 coreyschafer  staff  238 Jul 25 18:15 ..
-rw-------  1 coreyschafer  staff    0 Jul 12 17:06 .project
-rw-------  1 coreyschafer  staff  132 Jul 12 19:15 calc.py
```

BEFORE FIRST COMMIT , CREATING A .GITIGNORE FILE THAT ALLOWS FOR EXCEPTIONS WITHIN THE GIT INIT DIRECTORY



```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .project
        calc.py

nothing added to commit but untracked files present (use "git add" t
o track)

coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ touch .gitignore
```



ADD GITIGNORE FILE

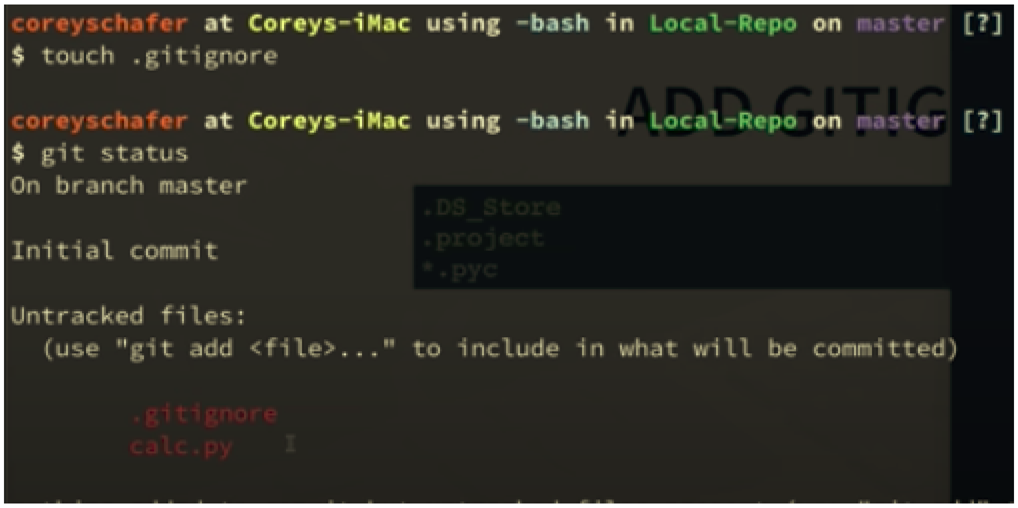
```
.DS_Store
.project
*.pyc
```

- Telling git to ignore all files with a .pyc extension. Wildcards are accepted

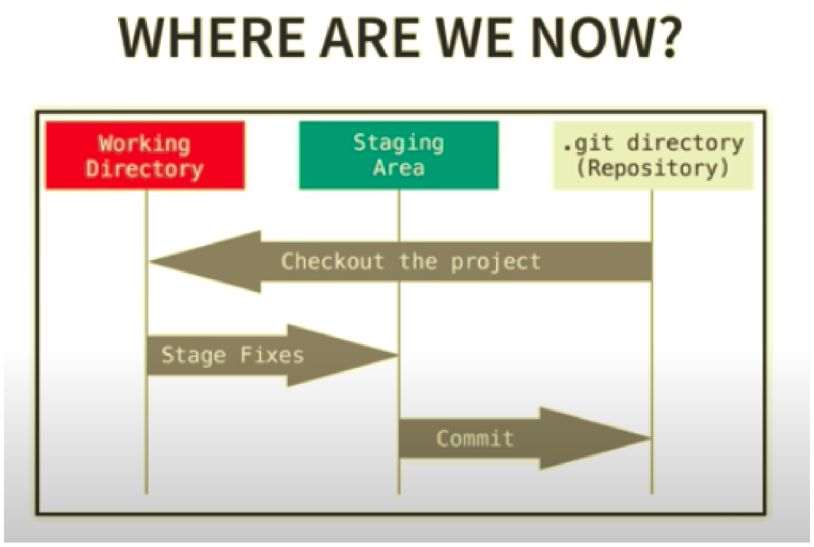
  Can specify any given folders to ignore with the initialized directory

- Once we add those files to git ignore and save it, we re-run git status again and the .project file no longer comes up in our list

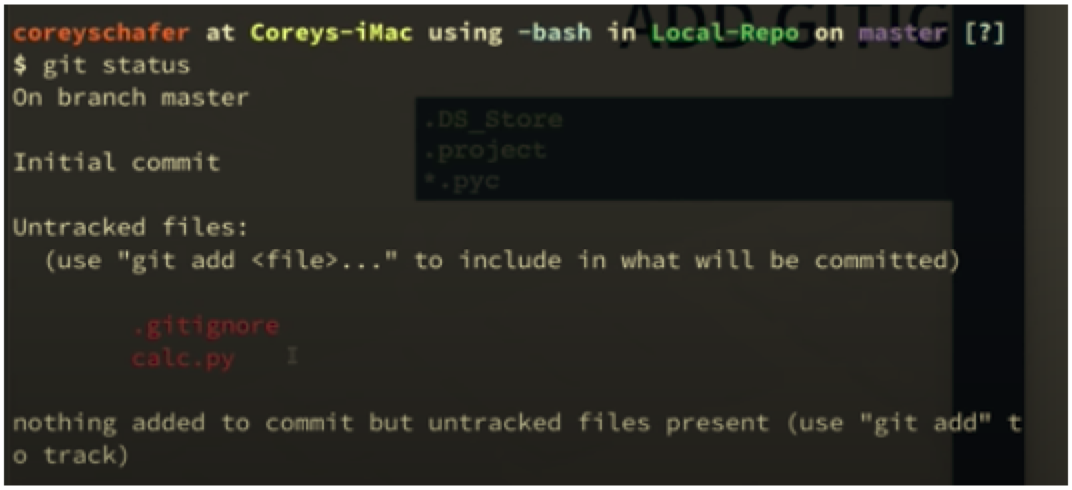
- The .gitignore file will show up in untracked files. But we want to commit the .gitignore file because we want git to know to always ignore those files.



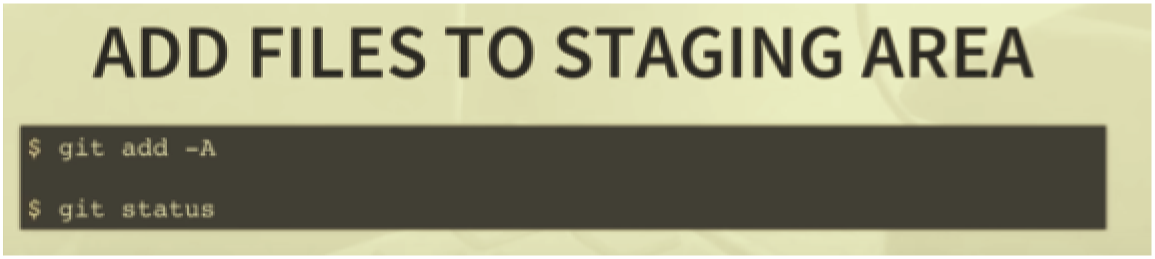
```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ touch .gitignore

coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        calc.py
```



- We are currently in our working directory, untracked and modified files will be here and list them when we run git status.
- The staging area is where we organize what we want to be committed to our repo. The reason for this staging area is so that we can pick and choose what we want committed. You can be detailed with your commits, you do not want to make a commit that says "I made a lot of changes to the code." You want to be as detailed as possible.



```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        calc.py

nothing added to commit but untracked files present (use "git add" t
o track)
```

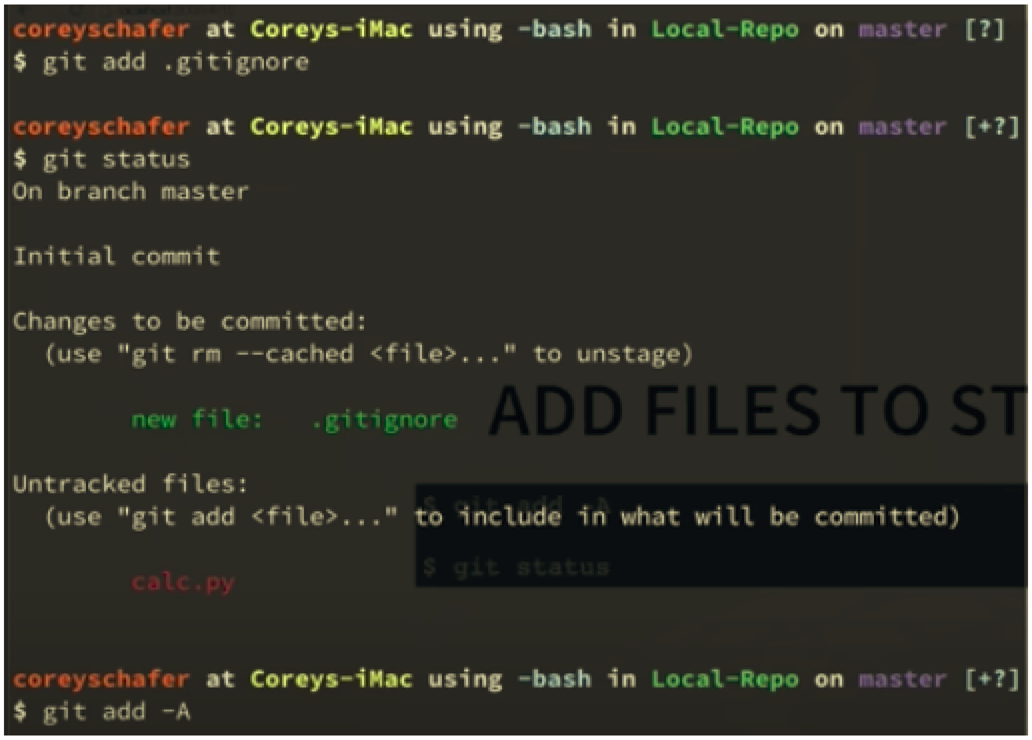


ADD FILES TO STAGING AREA

```
$ git add -A

$ git status
```

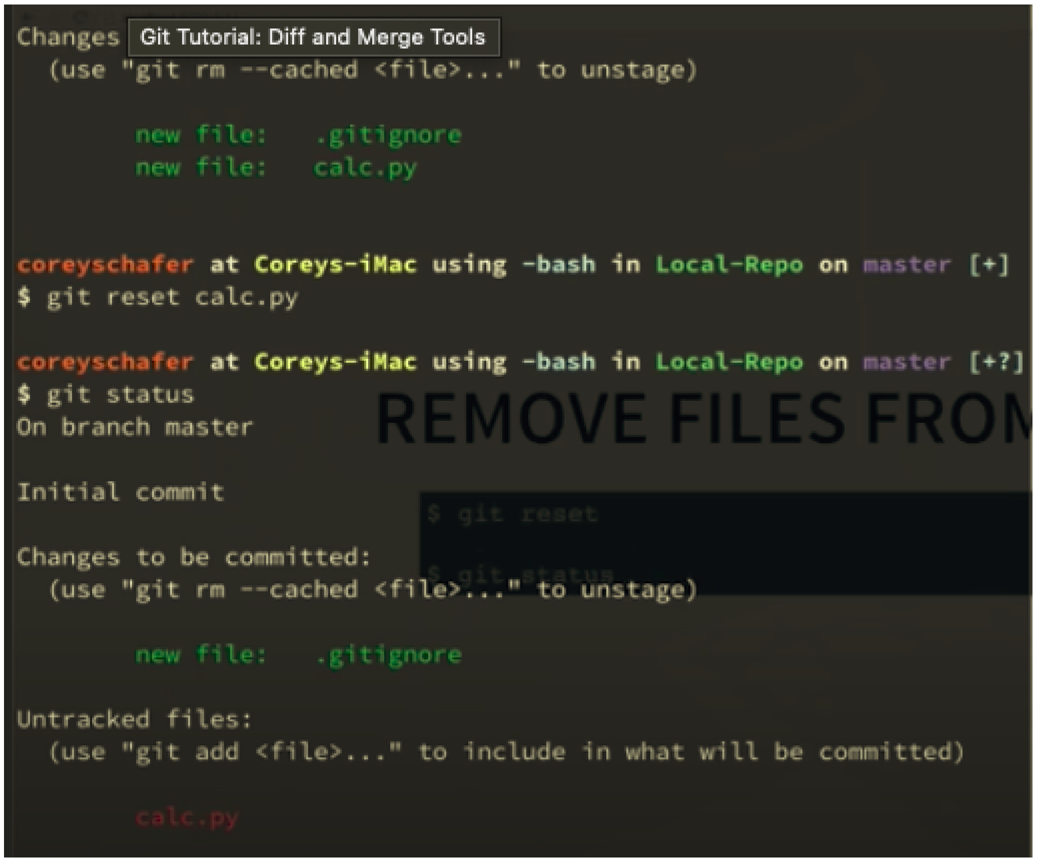
ADD FILE INDIVIDUALLY TO STAGING AREA EXAMPLE

- git add -A will move the file to the staging area



```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ git add .gitignore

coreyschafer at Coreys-iMac using -bash in Local-Repo on master [+?]
$ git status
On branch master

Initial commit

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

        new file:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        calc.py


coreyschafer at Coreys-iMac using -bash in Local-Repo on master [+?]
$ git add -A
```

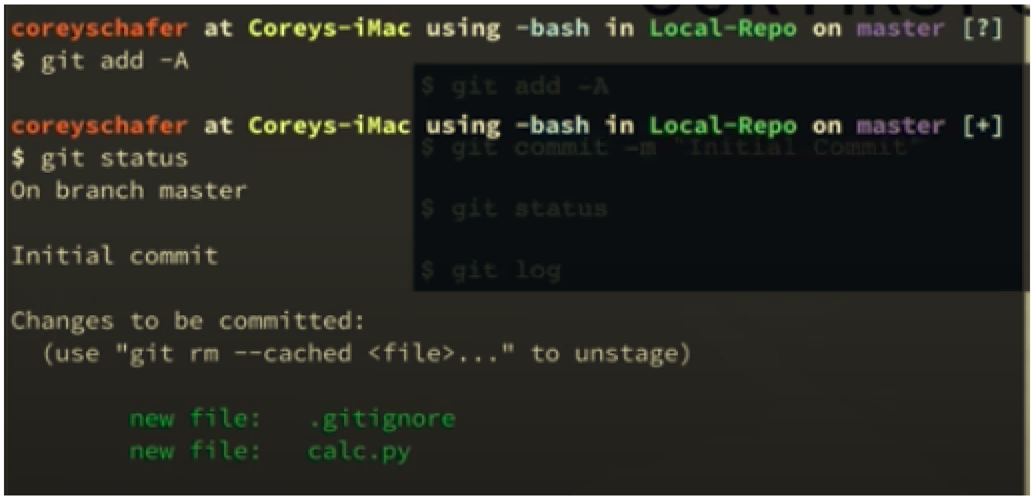
REMOVE FILE/FILES FROM STAGING AREA



```
Changes
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   calc.py


coreyschafer at Coreys-iMac using -bash in Local-Repo on master [+]
$ git reset calc.py

coreyschafer at Coreys-iMac using -bash in Local-Repo on master [+?]
$ git status
On branch master

Initial commit

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

        new file:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        calc.py
```

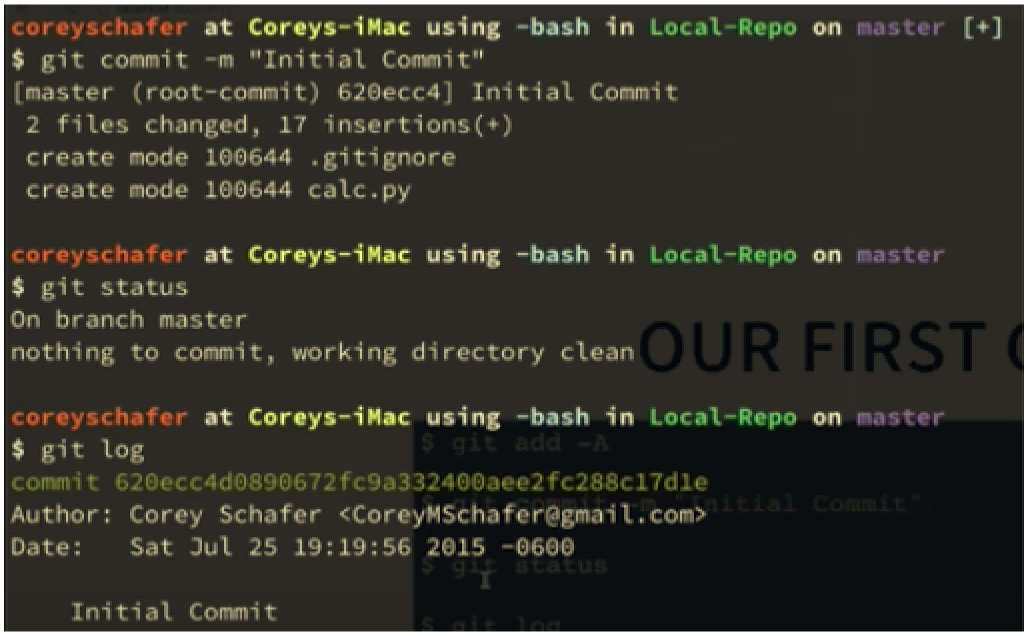
- git reset -----> by itself will remove all files from the staging area, and back to the working directory as untracked files.

COMMIT FILES



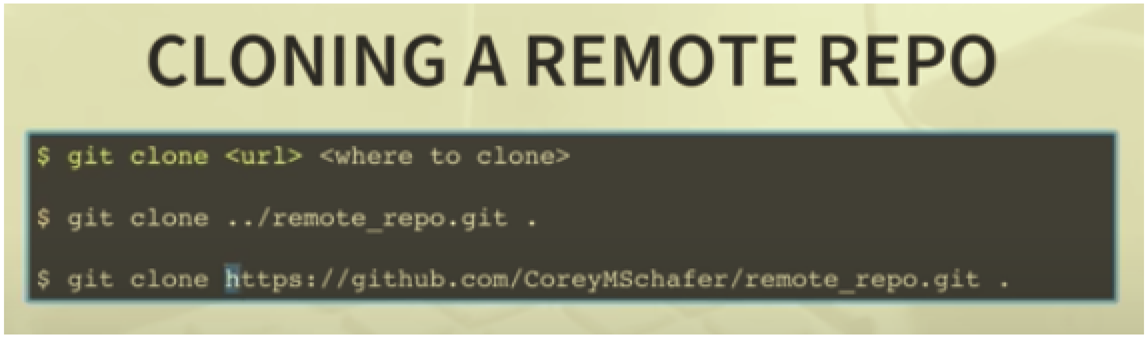
```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [?]
$ git add -A

coreyschafer at Coreys-iMac using -bash in Local-Repo on master [+]
$ git status
On branch master

Initial commit

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

        new file:   .gitignore
        new file:   calc.py
```

- Add all fields, and get them in the staging area to commit.
- Commit with a message, and verify the status after. Git log will show us the author and date.



```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master [+]
$ git commit -m "Initial Commit"
[master (root-commit) 620ecc4] Initial Commit
 2 files changed, 17 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 calc.py

coreyschafer at Coreys-iMac using -bash in Local-Repo on master
$ git status
On branch master
nothing to commit, working directory clean

coreyschafer at Coreys-iMac using -bash in Local-Repo on master
$ git log
commit 620ecc4d0890672fc9a332400aee2fc288c17d1e
Author: Corey Schafer <CoreyMSchafer@gmail.com>
Date:   Sat Jul 25 19:19:56 2015 -0600

    Initial Commit
```

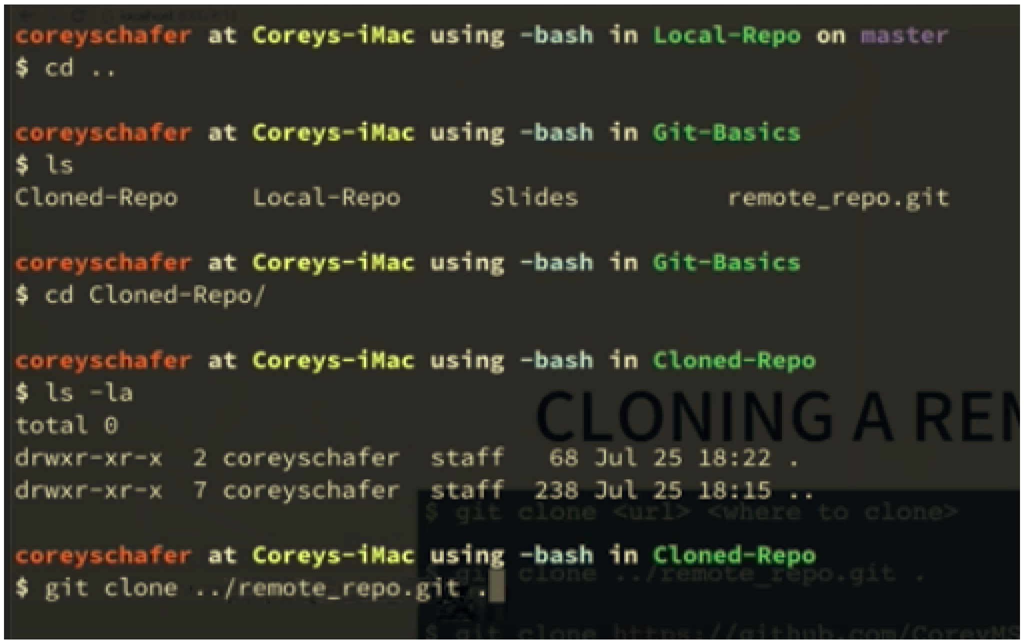
TRACK EXISTING REMOTE PROJECT WITH GIT



CLONING A REMOTE REPO

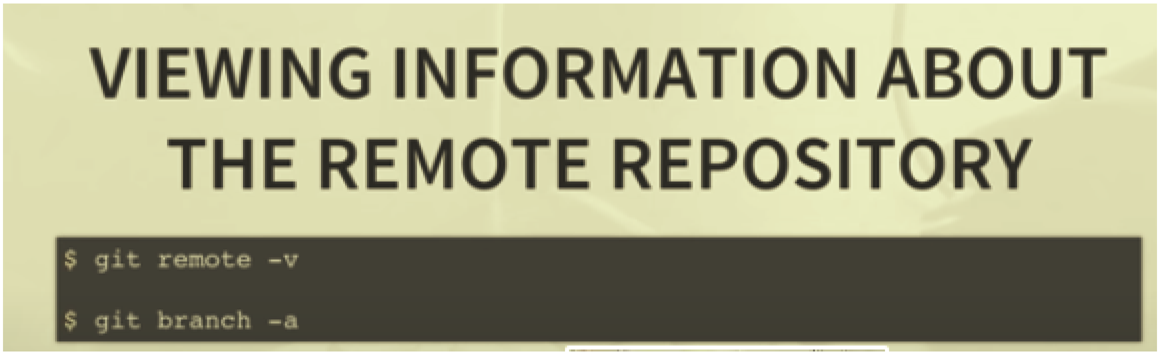
```
$ git clone <url> <where to clone>

$ git clone ../remote_repo.git .

$ git clone https://github.com/CoreyMSchafer/remote_repo.git .
```

- The dot after the .git signifies that we want to clone into the current working directory



```
coreyschafer at Coreys-iMac using -bash in Local-Repo on master
$ cd ..

coreyschafer at Coreys-iMac using -bash in Git-Basics
$ ls
Cloned-Repo     Local-Repo      Slides          remote_repo.git

coreyschafer at Coreys-iMac using -bash in Git-Basics
$ cd Cloned-Repo/

coreyschafer at Coreys-iMac using -bash in Cloned-Repo
$ ls -la
total 0
drwxr-xr-x  2 coreyschafer  staff   68 Jul 25 18:22 .
drwxr-xr-x  7 coreyschafer  staff  238 Jul 25 18:15 ..

coreyschafer at Coreys-iMac using -bash in Cloned-Repo
$ git clone ../remote_repo.git .
```

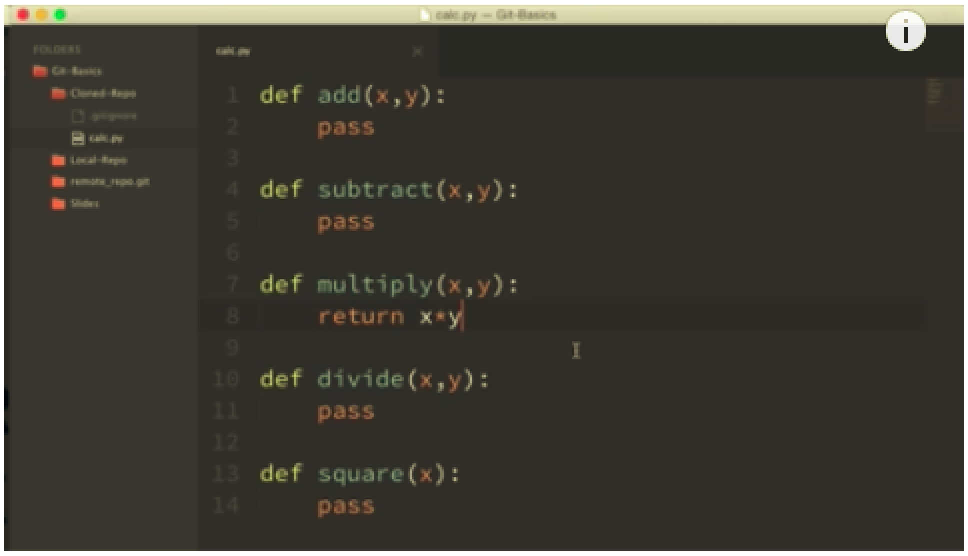
- We go back to the git basics directory, then enter the cloned-repo directory. Once in, we list what we have which is completely empty. Now we clone a remote repo here into the cloned-repo directory.



VIEWING INFORMATION ABOUT THE REMOTE REPOSITORY

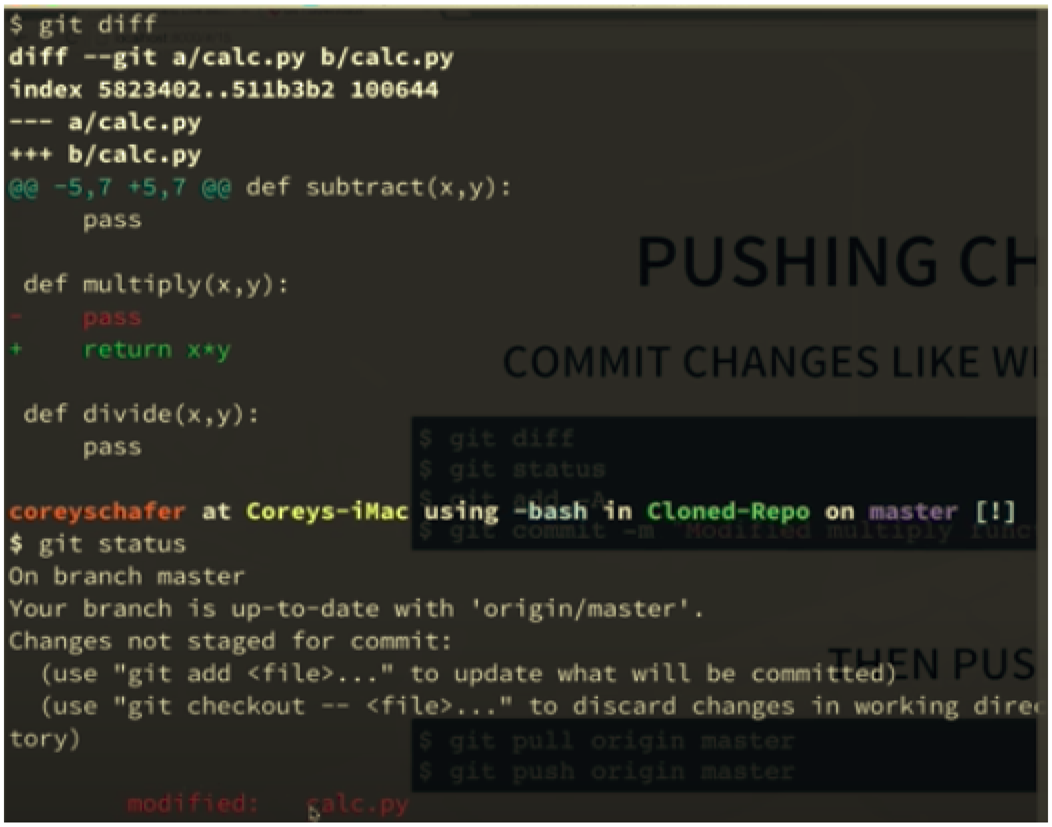
```
$ git remote -v

$ git branch -a
```

MAKE CHANGES TO THE CODE BASE, AND THEN PUSH THE CHANGE TO THE REMOTE REPO



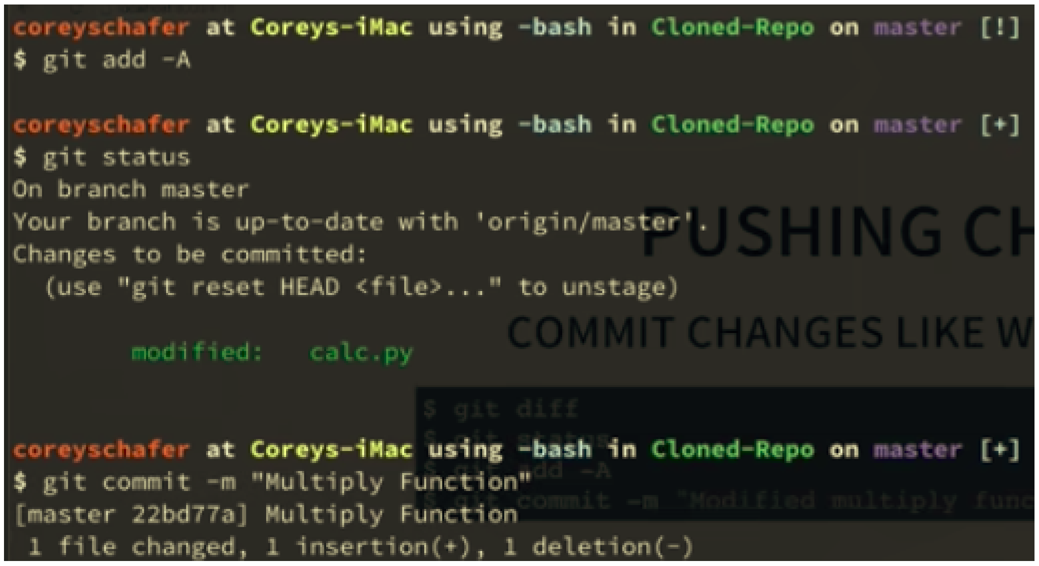
```
def add(x,y):
    pass

def subtract(x,y):
    pass

def multiply(x,y):
    return x*y

def divide(x,y):
    pass

def square(x):
    pass
```

- We make changes to the code, and save the file. Next thing to do is commit the changes locally like before.



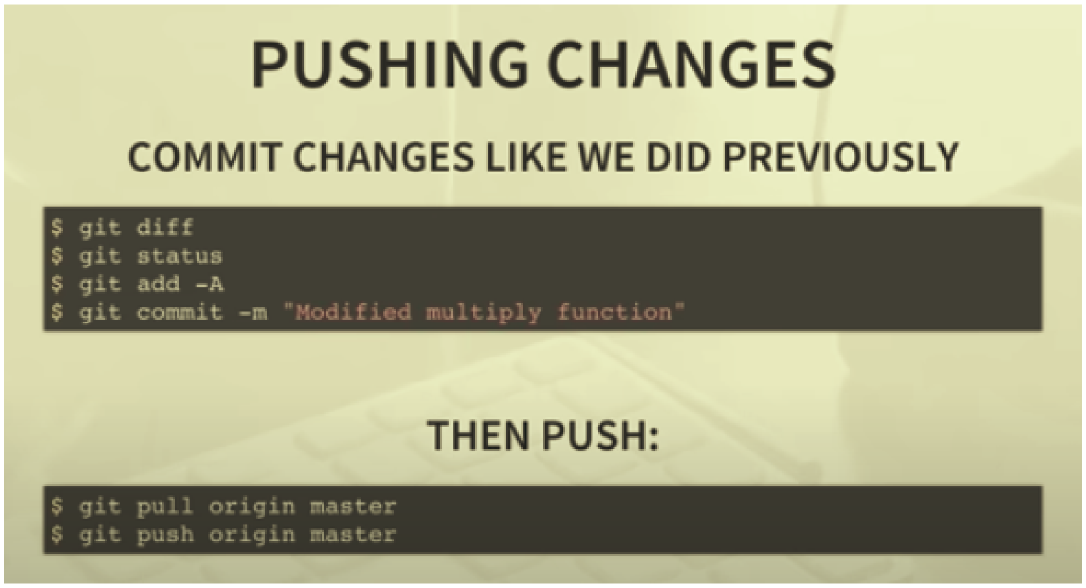
```
$ git diff
diff --git a/calc.py b/calc.py
index 5823402..511b3b2 100644
--- a/calc.py
+++ b/calc.py
@@ -5,7 +5,7 @@ def subtract(x,y):
     pass

 def multiply(x,y):
-    pass
+    return x*y

 def divide(x,y):
     pass

coreyschafer at Coreys-iMac using -bash in Cloned-Repo on master [!]
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working direc
tory)

        modified:   calc.py
```

- Git diff allows us to see changes, and within git status it will display that we have a modified file in our working directory
- Now we must add this to the staging directory through the following:



```
coreyschafer at Coreys-iMac using -bash in Cloned-Repo on master [!]
$ git add -A

coreyschafer at Coreys-iMac using -bash in Cloned-Repo on master [+]
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   calc.py


coreyschafer at Coreys-iMac using -bash in Cloned-Repo on master [+]
$ git commit -m "Multiply Function"
[master 22bd77a] Multiply Function
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- Now we have committed these files locally, and now we want to push these changes to the remote repo so that other people have access to them.



PUSHING CHANGES

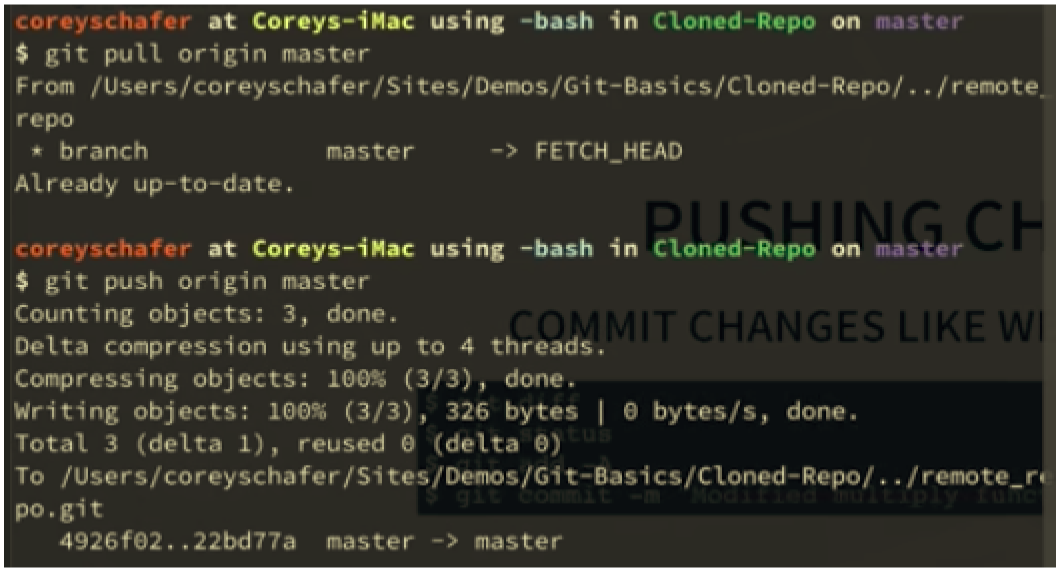
COMMIT CHANGES LIKE WE DID PREVIOUSLY

```
$ git diff
$ git status
$ git add -A
$ git commit -m "Modified multiply function"
```

THEN PUSH:

```
$ git pull origin master
$ git push origin master
```

- Git pull is essential because we must remember that a project could have multiple developers and people have been pushing code to that remote repo while we have been working on our own features.
- Git pull will pull all changes that have been made since that last repo pull



```
coreyschafer at Coreys-iMac using -bash in Cloned-Repo on master
$ git pull origin master
From /Users/coreyschafer/Sites/Demos/Git-Basics/Cloned-Repo/../remote_
repo
 * branch            master     -> FETCH_HEAD
Already up-to-date.

coreyschafer at Coreys-iMac using -bash in Cloned-Repo on master
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 326 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /Users/coreyschafer/Sites/Demos/Git-Basics/Cloned-Repo/../remote_r
po.git
   4926f02..22bd77a  master -> master
```

- Origin is the name of our remote repo, and master is the branch

COMMON WORKFLOW