Using Git with terminal

https://github.com/codepath/ios_guides/wiki/Using-Git-with-Terminal

Git is an essential tool in the developers toolkit. Git stores your files like a stream of snapshots, allowing you and other collaborators to make changes to projects and keep everything in sync. This Guide introduces using Git with Terminal. You can click the video link in each section for a video walkthrough or watch the video playlist now.

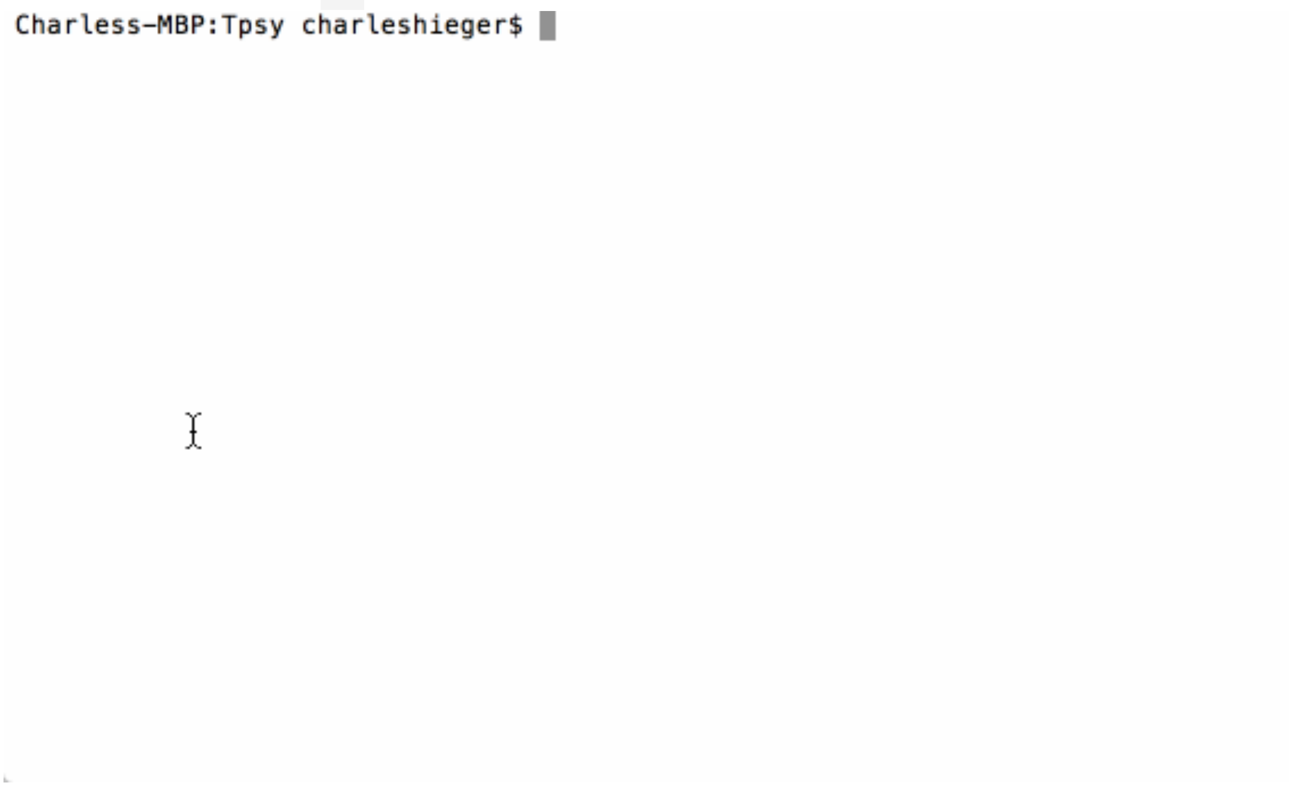
Basic Terminal Navigation

The Terminal is much like the finder, but less pretty. There are just a few basic commands needed to navigate around in Terminal. Using Terminal, we often refer to folders as directories. Check out this Terminal Cheat Sheet for Mac for a more complete list of commands.

- Open up terminal, cmd + space "terminal".
- `pwd` Figure out which directory you are currently in by "Printing the Working Directory".
- `ls` List the directories and files inside the current directory.
  - `ls -a` include hidden files in the list of directories and files. This is helpful when trying to find hidden files like `.git` or `.gitignore`
- `cd [folder]` Go into the folder. e.g. `cd Desktop/Developer`
  - `cd` This by itself will navigate all the way out to your **Home Directory**
  - `cd ..` Back out to the parent directory of the current directory.
- `open [file]` Same as "double-clicking" a file or folder in finder.

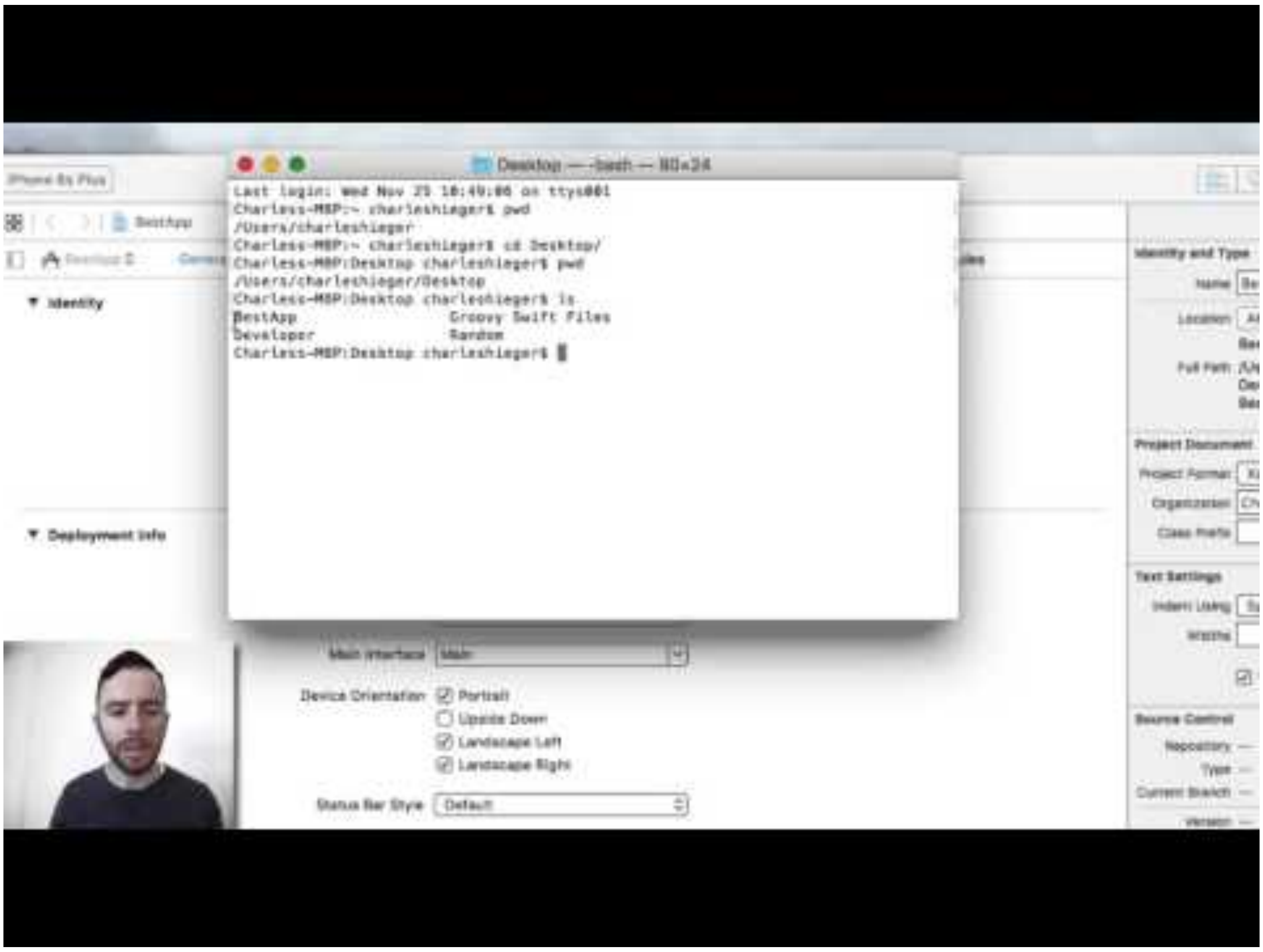
    - `open .` Opens the current folder or file in finder or it's default application.
- `clear` When you need a fresh Terminal window

Hint If you can't find the path to a file or folder through terminal, search for it using Finder, then drag the file or folder right into your terminal window! If you are trying to change to that directory, you will need to type `cd` before dragging in the folder.



```
Charless-MBP:Tpsy charleshieger$
```

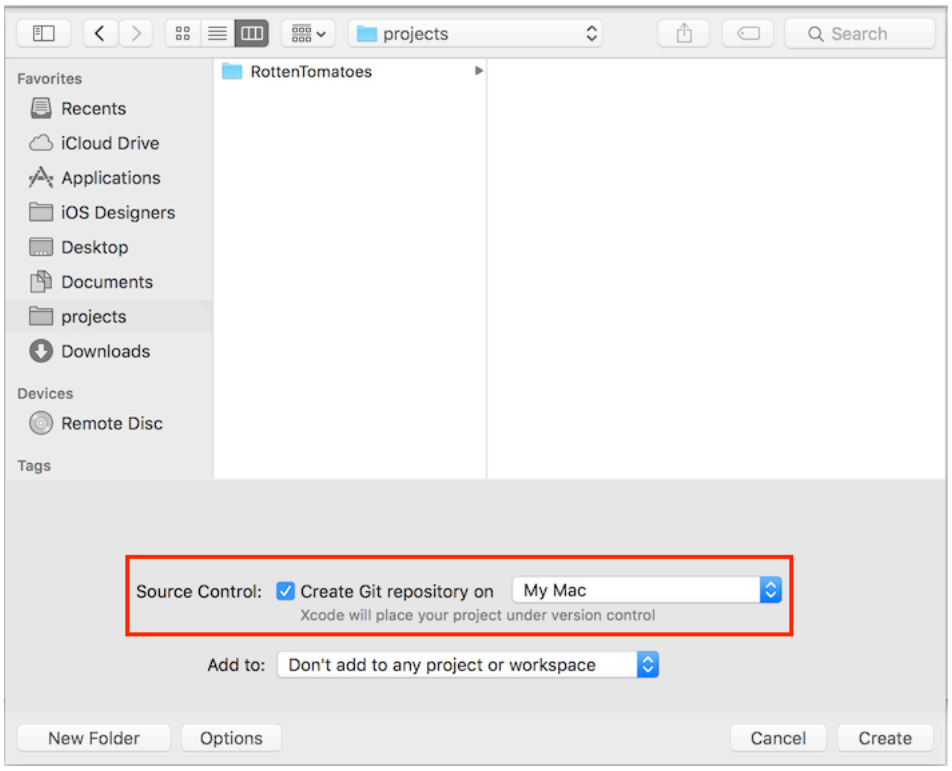
Creating a Local Git Repository



```
Last login: Wed Nov 25 10:40:06 on ttys001
Charless-MBP:~ charleshieger$ pwd
/Users/charleshieger
Charless-MBP:~ charleshieger$ cd Desktop/
Charless-MBP:Desktop charleshieger$ pwd
/Users/charleshieger/Desktop
Charless-MBP:Desktop charleshieger$ ls
BestApp                         Groovy Swift Files
Developer                       Random
Charless-MBP:Desktop charleshieger$ 
```

Local Git repositories are created and managed locally on your computer.

- Xcode will automatically create a local Git repository for your project if you select, "Create Git repository on... My Mac", when you first create your project. You should **Always** select this when creating a new Xcode project.



Source Control: ☑ Create Git repository on My Mac
Xcode will place your project under version control

Add to: Don't add to any project or workspace

If you already have a project, where a Git repository was not created when you made the project, you can create a local Git repository using terminal.

- Navigate to your Xcode Project folder in Terminal.
  ```
  git init
  ```

Creating a .gitignore File

When collaborating using Git, you will inevitably run into "merge conflicts". However, you will save yourself from a lot of extraneous conflicts by adding a `.gitignore` file with the proper content.

- Navigate to your Xcode Project folder in Terminal.

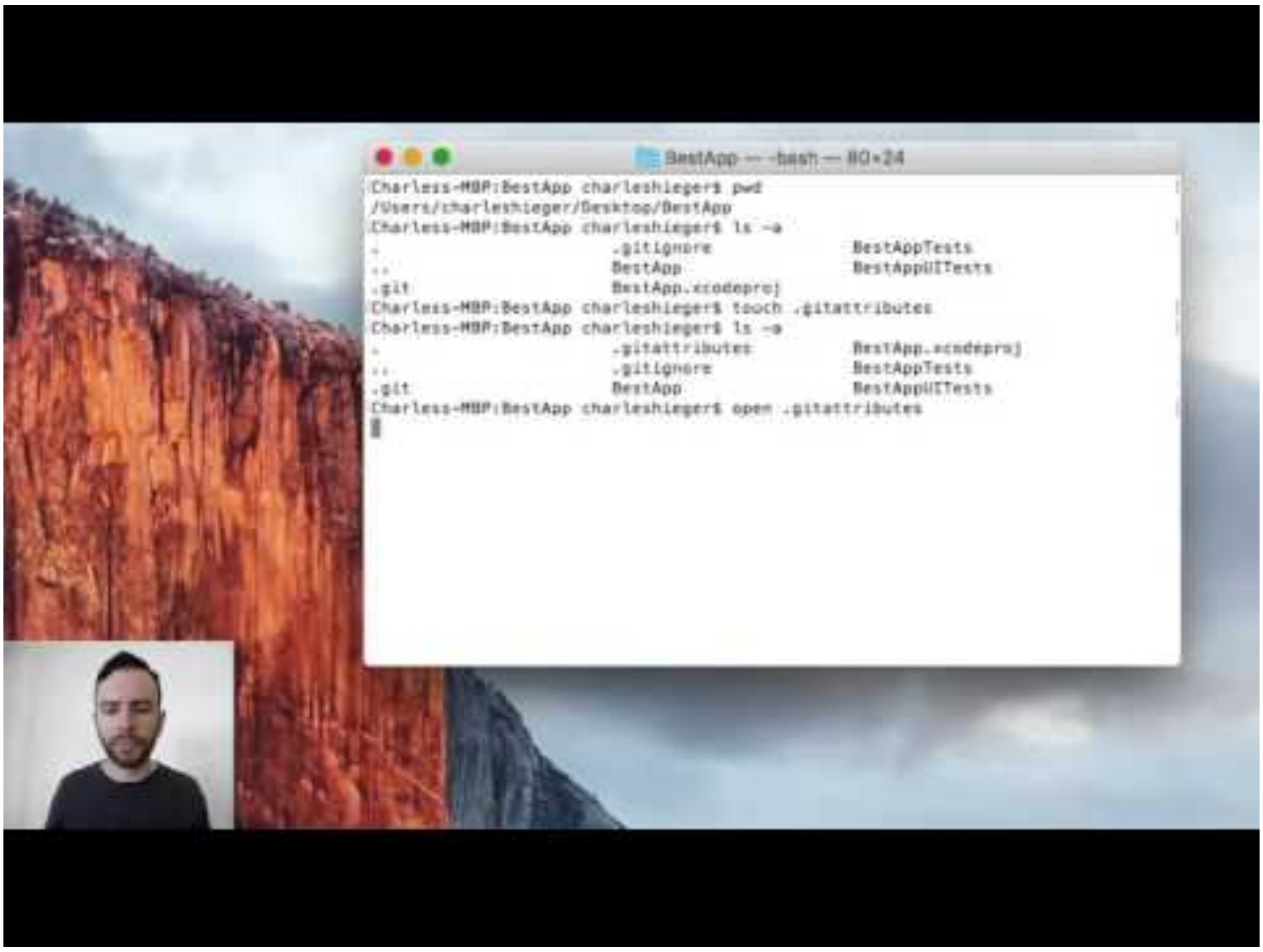
- Use `ls -a` to list all files including hidden files to check if a `.gitignore` has already been created.

- If there is not an existing `.gitignore` file, create one.
  `touch .gitignore`

Add Files and Directories to .gitignore

- Navigate to your Xcode Project folder in Terminal.
- Open your `.gitignore` file using, `open .gitignore`
- Copy and paste the latest and greatest list of files and folders you want to ignore into the .gitignore file.

Create and Configure .gitattributes File



```
Charless-MBP:BestApp charleshieger$ pwd
/Users/charleshieger/Desktop/BestApp
Charless-MBP:BestApp charleshieger$ ls -a
.                       .gitignore              BestAppTests
..                      BestApp                 BestAppUITests
.git                    BestApp.xcodeproj
Charless-MBP:BestApp charleshieger$ touch .gitattributes
Charless-MBP:BestApp charleshieger$ ls -a
.                       .gitattributes          BestApp.xcodeproj
..                      .gitignore              BestAppTests
.git                    BestApp                 BestAppUITests
Charless-MBP:BestApp charleshieger$ open .gitattributes
```

This file allows us to tell Git how we want certain files to be treated. We will add a file to the `.gitattributes` that tells Git to treat any file with `.pbxproj` as binary.

- Navigate to your Xcode Project folder in Terminal.

- Use `ls -a` to list all files including hidden files to check if a `.gitattributes` has already been created.

- If there is not an existing `.gitattributes file, create one.
  `touch .gitattributes`

- Open your `.gitattributes` file, `open .gitattributes`

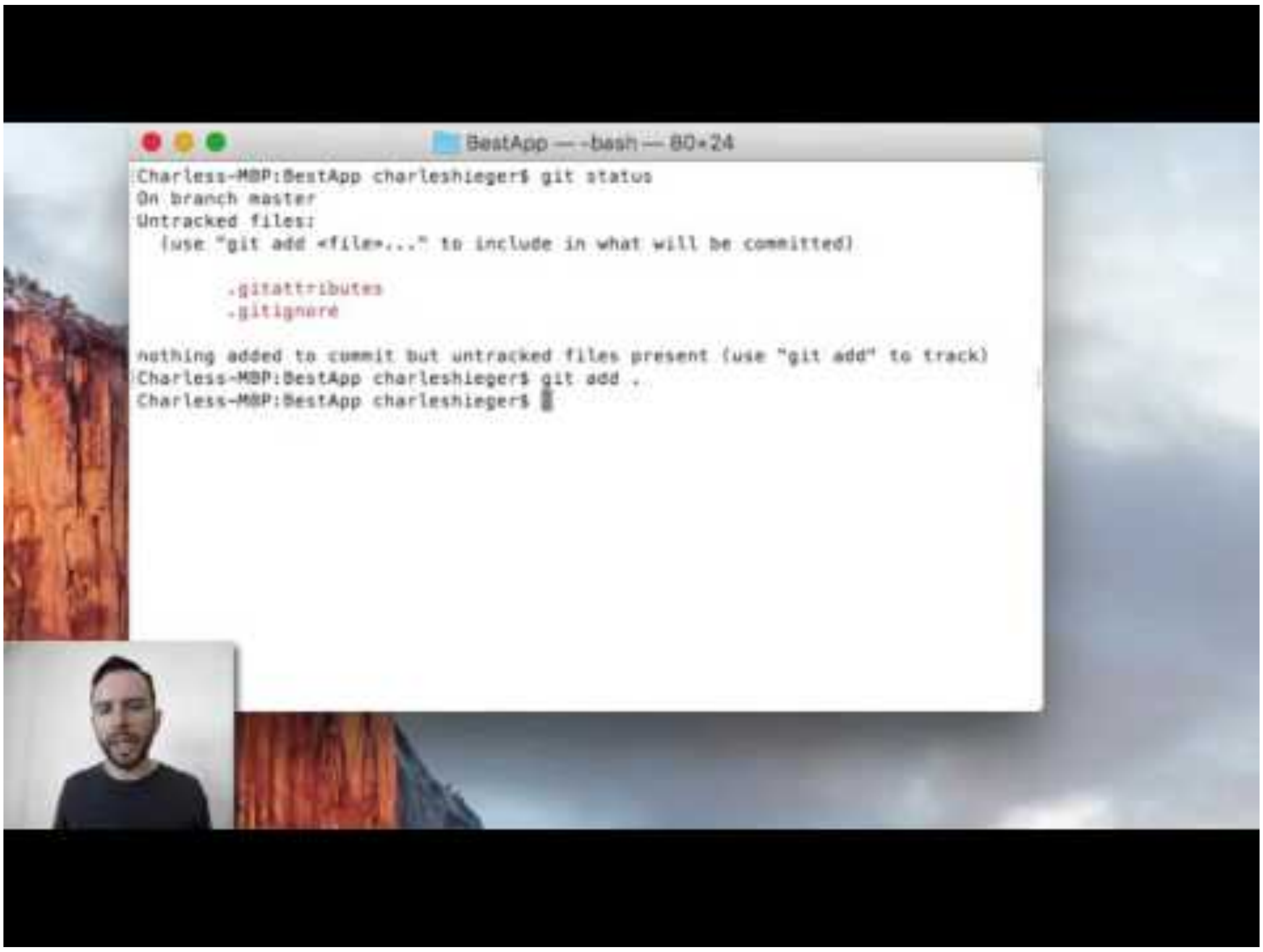
- Add the following code to the `.gitattributes` file.
  `*.pbxproj binary merge=union`

Add and configure README

- Add a README.md file to your repo
  `touch README.md`

- Open and edit your README.md
  `open README.md`

Adding and Committing Changes to Local Repository



```
Charless-MBP:BestApp charleshieger$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitattributes
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
Charless-MBP:BestApp charleshieger$ git add .
Charless-MBP:BestApp charleshieger$ 
```

Now that your local Git is all setup from the last step, you can update your local Git repository with any changes you make to your project.

- Navigate to your Xcode Project folder in Terminal.

- Check for any changes that have been made to files since your last commit
  `git status`

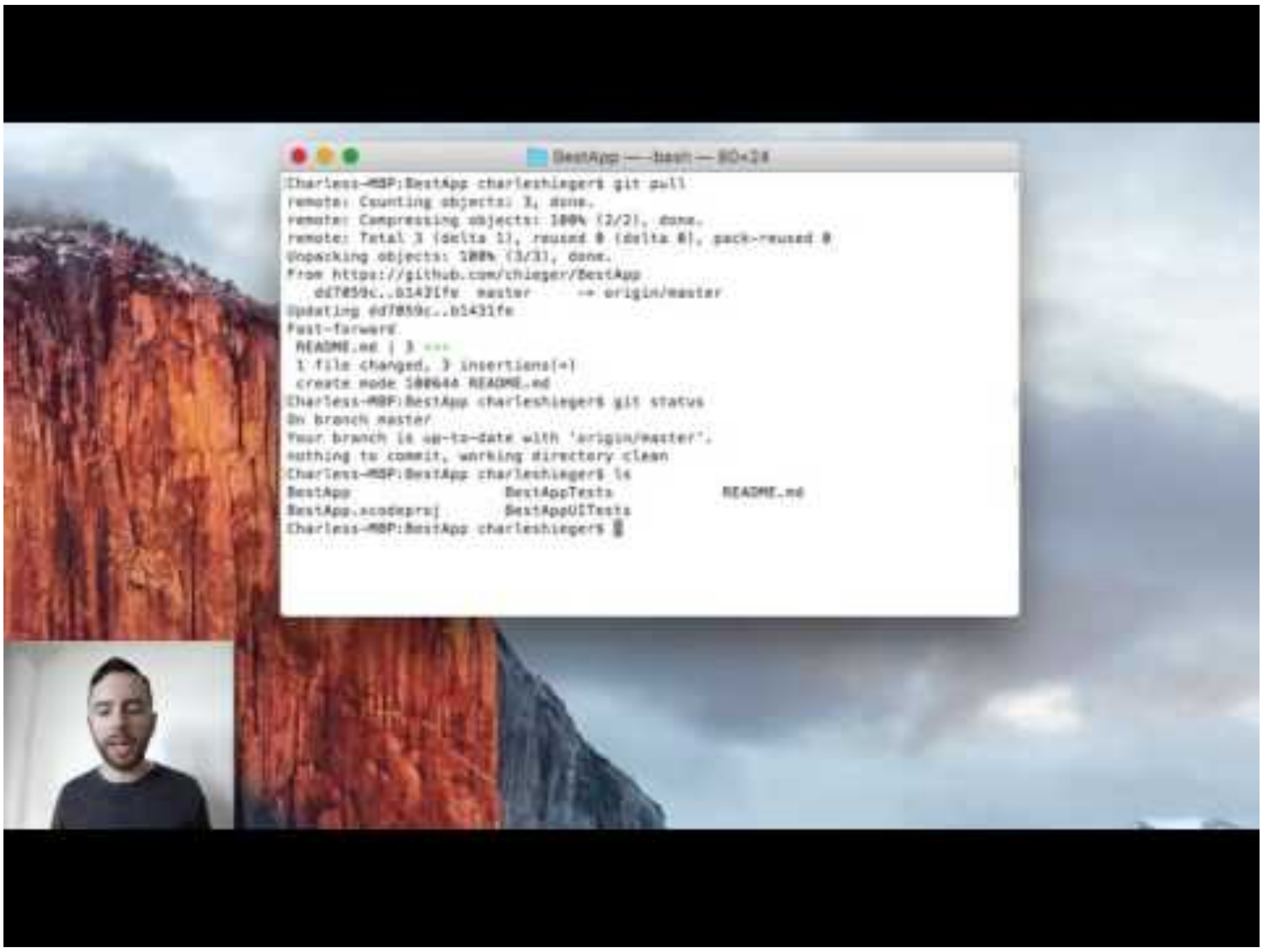
- Stage any files with changes you'd like to commit.
  ```
  git add [filepath] # This adds an individual file
  ```

-
  ```
  git add . # This adds all files with changes
  ```

- Check to see what was added.
  ```
  git status
  ```

- Apply the changes to your local Git repository with a message briefly outlining the changes you made.
  ```
  git commit -m "Here is my commit message"
  ```

Linking to a Remote Repository



```
Charless-MBP:BestApp charleshieger$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/chieger/BestApp
   dd7059c..b1431fe  master     -> origin/master
Updating dd7059c..b1431fe
Fast-forward
 README.md | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
Charless-MBP:BestApp charleshieger$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
Charless-MBP:BestApp charleshieger$ ls
BestApp                 BestAppTests            README.md
BestApp.xcodeproj       BestAppUITests
Charless-MBP:BestApp charleshieger$ 
```

There are many remote repository options. In this guide we will be using GitHub. You will need a GitHub account if you don't have one already.

Repository URLs

There are two types of repository URLs, HTTPS and SSH:

- HTTPS: `https://github.com/homer/duff_project.git`
- SSH: `git@github.com:homer/duff_project.git`

The general workflow is the same for both URLs, but there are some differences in the specifics of commands.

SSH Setup

You'll first need to register your machine's SSH keys with your Github account. You only need to perform this step **once** for each machine you use.

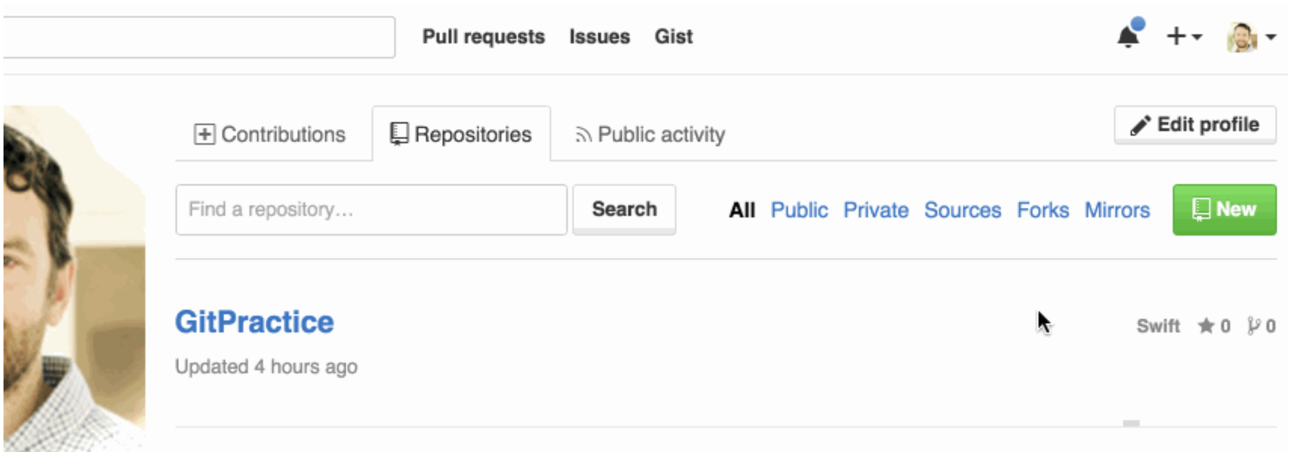
Once you've added your SSH key, the SSH repository URL will look like `git@github.com:myusername/reponame.git` but for your username and project. You won't need to provide credentials (username/password) for any further git commands.

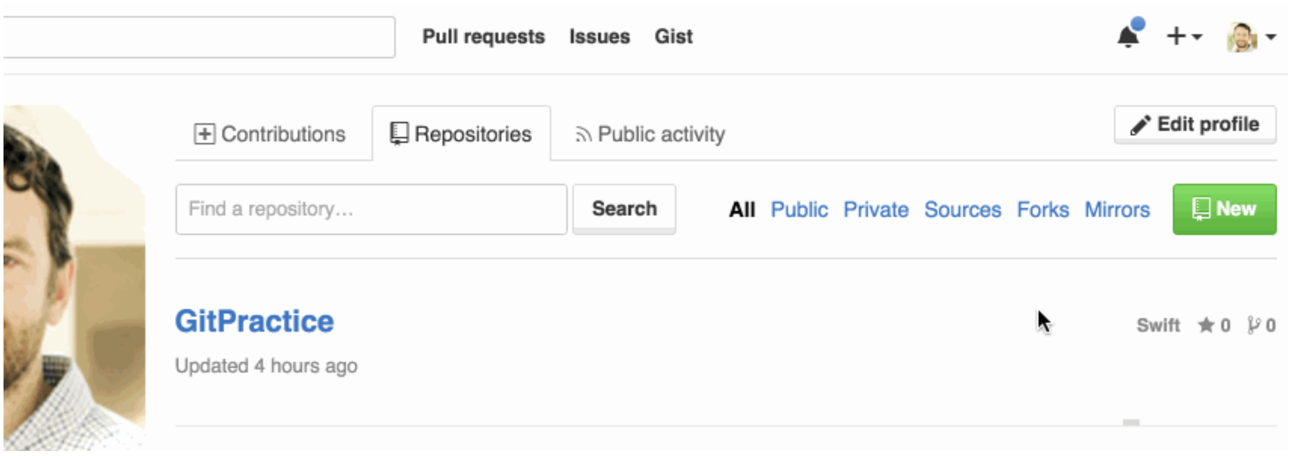
HTTPS Setup

There is no additional setup required for using HTTPS. However, you'll need to enter your username and password each time you run the `git pull` or `git push` commands:

```
git push origin master
<enter your username at promt>
<enter your password at prompt>
```

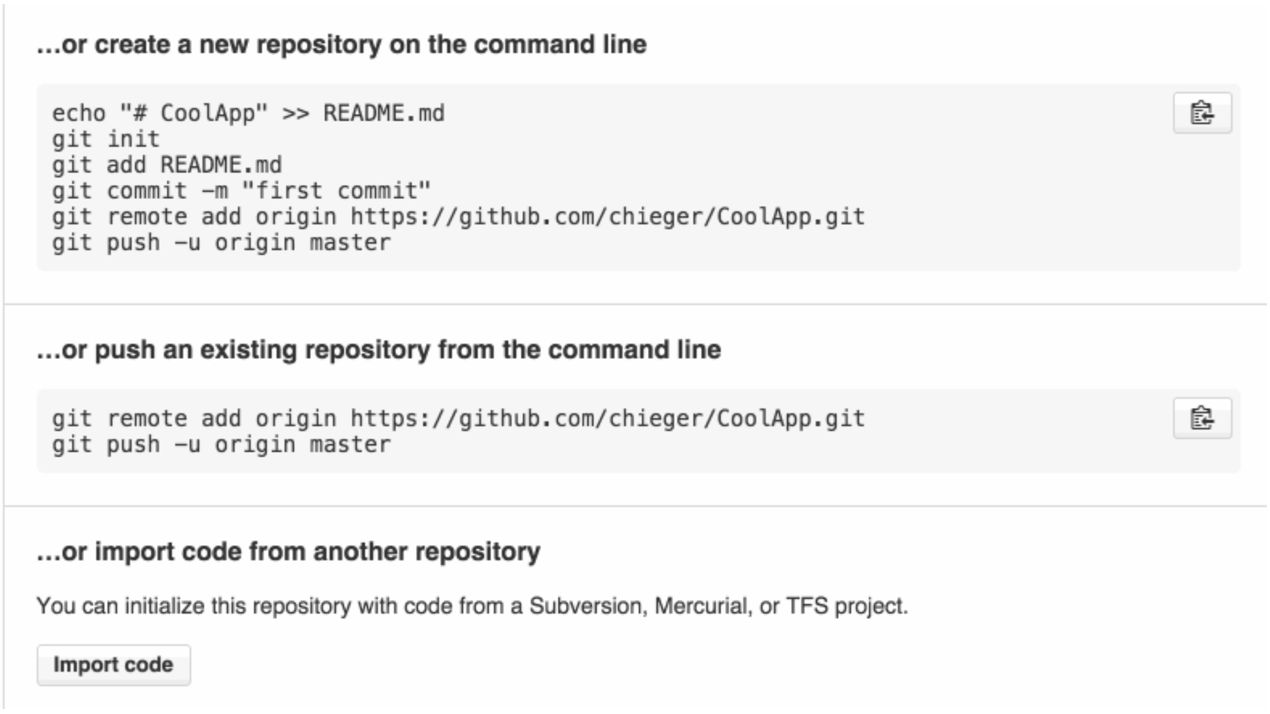
Create a new Repository on GitHub

- 



Navigate to your Xcode Project folder in Terminal.

- Link to the remote repository by adding the following code in Terminal, provided from your GitHub repository. We want to "push an existing repository from the command line".
    - NOTE: Make sure you add YOUR GitHub remote address!

- 

  ...or create a new repository on the command line

  ```
  echo "# CoolApp" >> README.md
  git init
  git add README.md
  git commit -m "first commit"
  git remote add origin https://github.com/chieger/CoolApp.git
  git push -u origin master
  ```

  ...or push an existing repository from the command line

  ```
  git remote add origin https://github.com/chieger/CoolApp.git
  git push -u origin master
  ```

  ...or import code from another repository

  You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

    - NOTE: Decide if you want the remote to use SSH or HTTPS before setting your remote in the next step. I like ssh for the convenience of not always having to input my credentials.
- `git remote add origin https://github.com/yourUserName/yourRepoName.git`
- `git push -u origin master`

Syncing Local Repository With Your Remote Repository

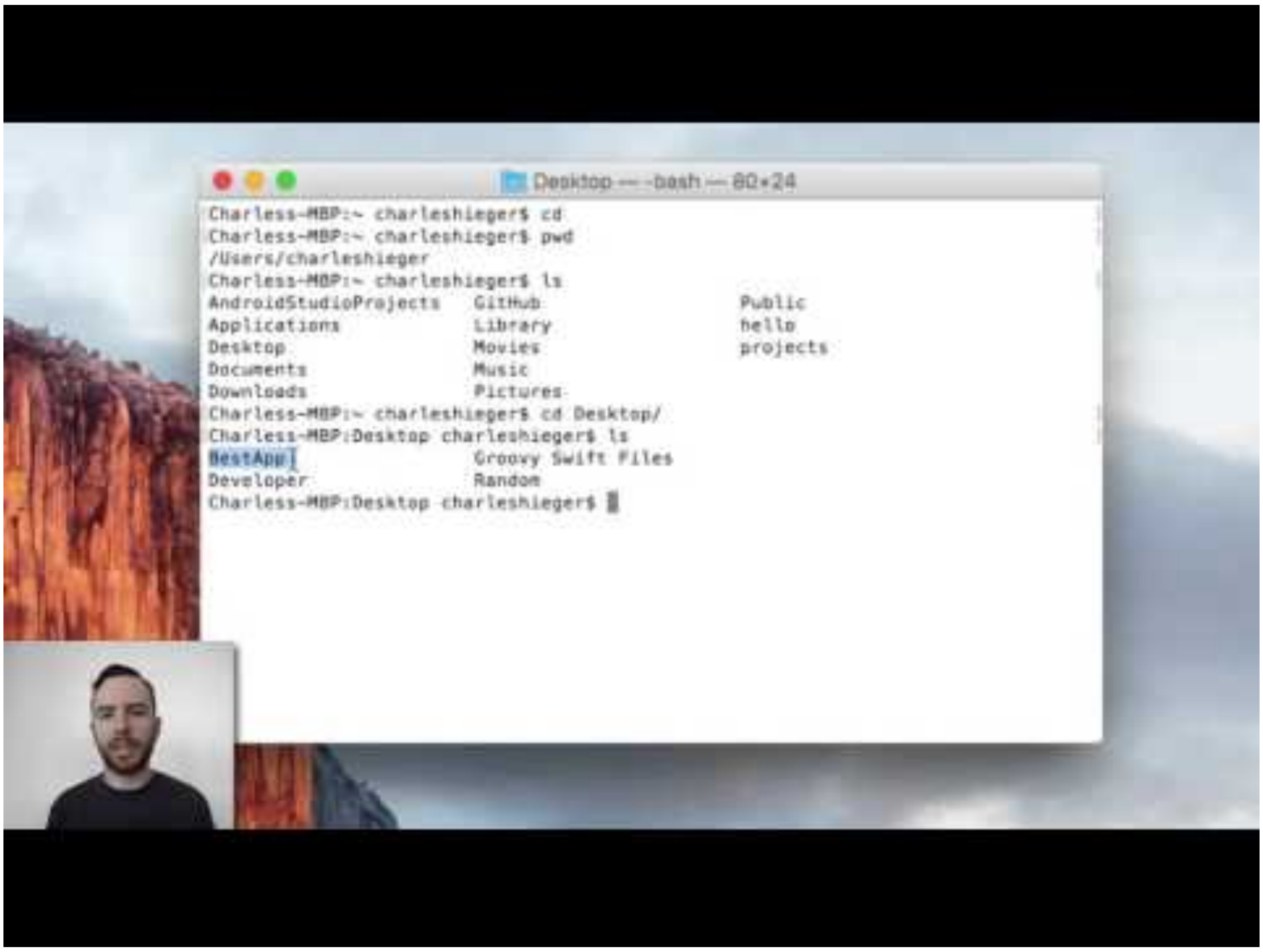
When you have made changes to your project and used the `git add .`, `git status` and `git commit -m "Message"` commands to update your local Git repository, the next step is to sync with your remote repository. Luckily, keeping in sync can be done with only two additional steps, `git push` and `git pull`.

- Navigate to your Xcode Project folder in Terminal.

- Push changes you have made locally to update the remote repository.
  `git push`

- Pull changes that have been added to the remote repository by a collaborator to update your local repository.
  `git pull`

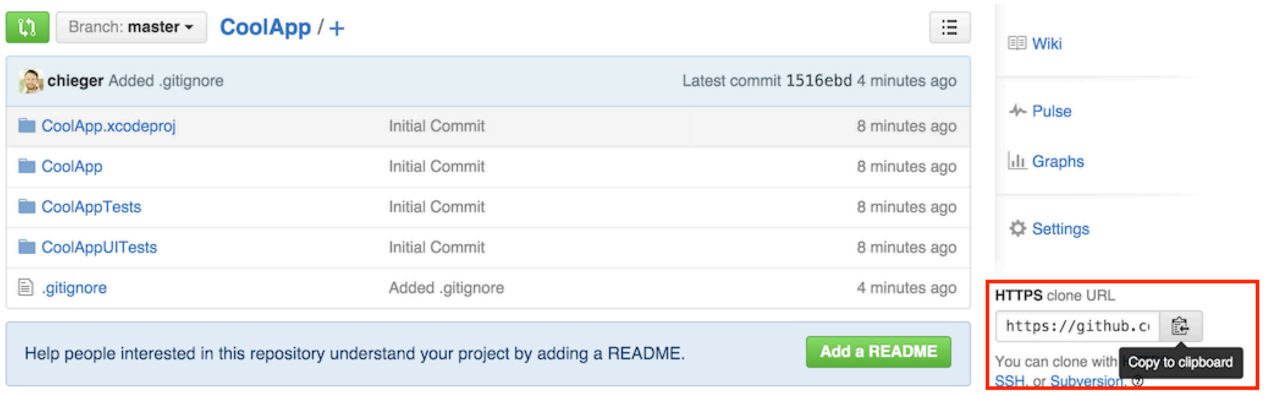
Cloning a Remote Repository



```
Charless-MBP:~ charleshieger$ cd
Charless-MBP:~ charleshieger$ pwd
/Users/charleshieger
Charless-MBP:~ charleshieger$ ls
AndroidStudioProjects   GitHub                  Public
Applications            Library                 hello
Desktop                 Movies                  projects
Documents               Music
Downloads               Pictures
Charless-MBP:~ charleshieger$ cd Desktop/
Charless-MBP:Desktop charleshieger$ ls
BestApp                 Groovy Swift Files
Developer               Random
Charless-MBP:Desktop charleshieger$ 
```

If you are collaborating on a project, or just want access to your project from a different computer, you will need to **clone** from the remote repository.

- Get the url of the remote repository. You can access this by going to the remote repository in GitHub and copying the clone link, bottom right.



chieger Added .gitignore — Latest commit 1516ebd 4 minutes ago

CoolApp.xcodeproj	Initial Commit	8 minutes ago
CoolApp	Initial Commit	8 minutes ago
CoolAppTests	Initial Commit	8 minutes ago
CoolAppUITests	Initial Commit	8 minutes ago
.gitignore	Added .gitignore	4 minutes ago

Help people interested in this repository understand your project by adding a README.

HTTPS clone URL

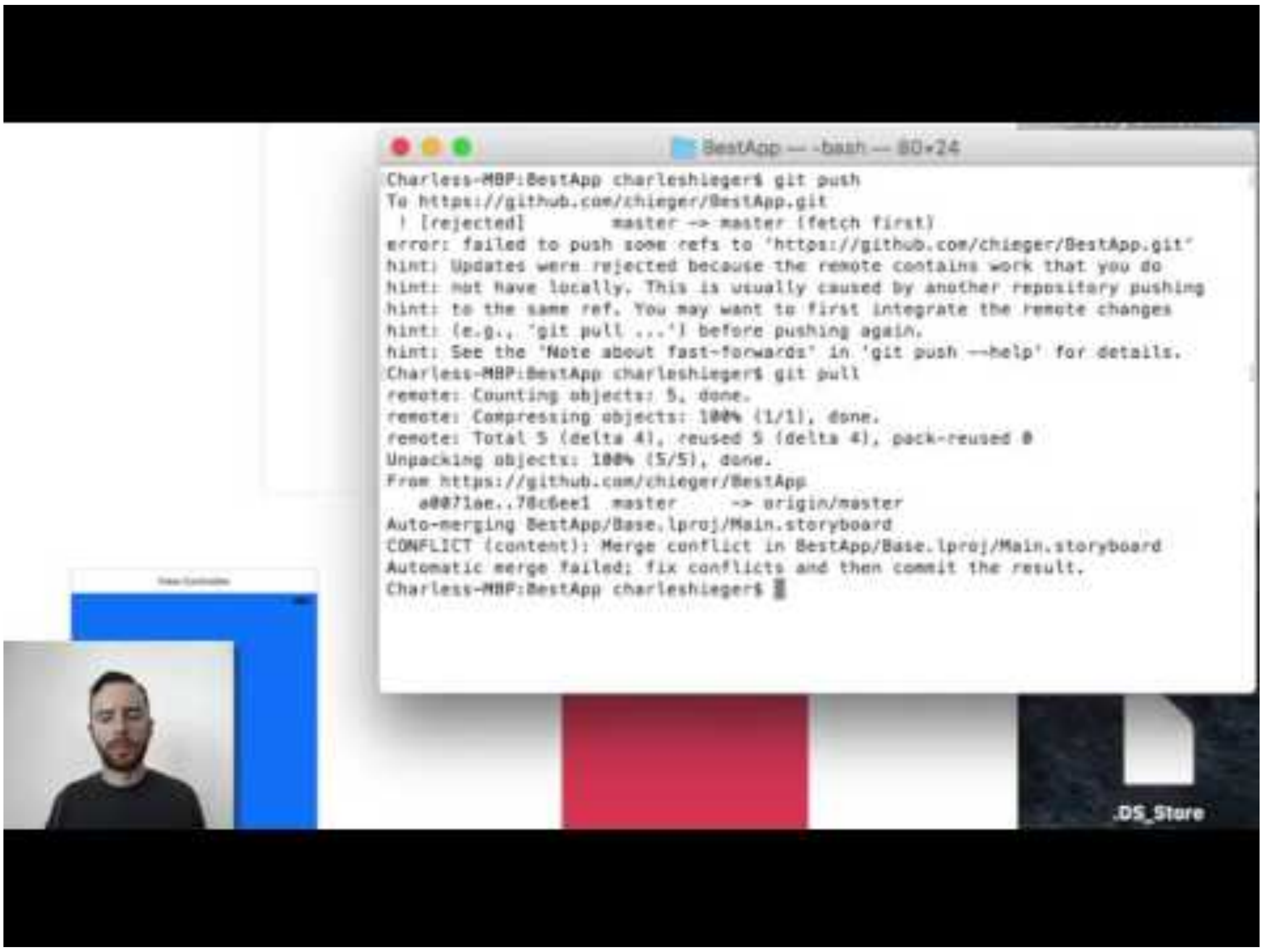
`https://github.c`

You can clone with SSH, or Subversion.

- In Terminal, navigate to the directory you want to clone the repository in.

- Clone the repository...
  `git clone https://yourRemoteRepositoryUrl`

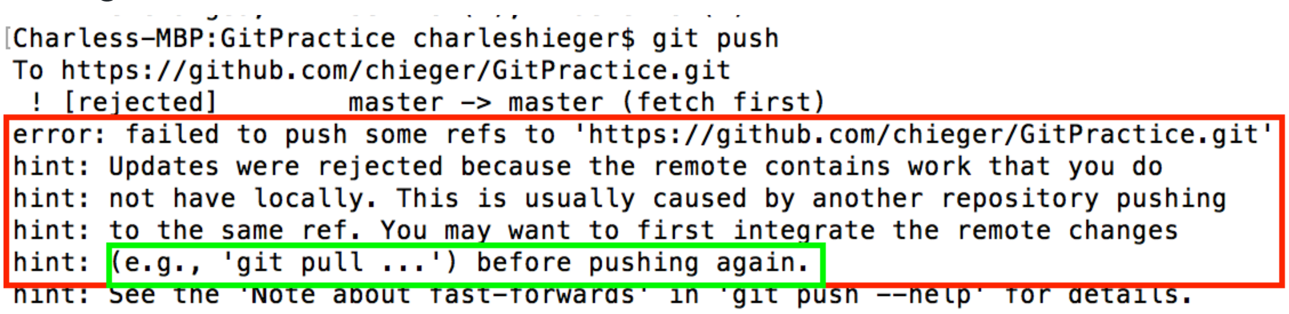
Dealing With Merge Conflicts



```
Charless-MBP:BestApp charleshieger$ git push
To https://github.com/chieger/BestApp.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/chieger/BestApp.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
Charless-MBP:BestApp charleshieger$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 5 (delta 4), reused 5 (delta 4), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/chieger/BestApp
   a0071ae..78c6ee1  master     -> origin/master
Auto-merging BestApp/Base.lproj/Main.storyboard
CONFLICT (content): Merge conflict in BestApp/Base.lproj/Main.storyboard
Automatic merge failed; fix conflicts and then commit the result.
Charless-MBP:BestApp charleshieger$ 
```

Inevitably, there will come a time when you AND a collaborator will make changes to the same file and both try to **push** to the remote repository. This will result in a **Merge Conflict**.

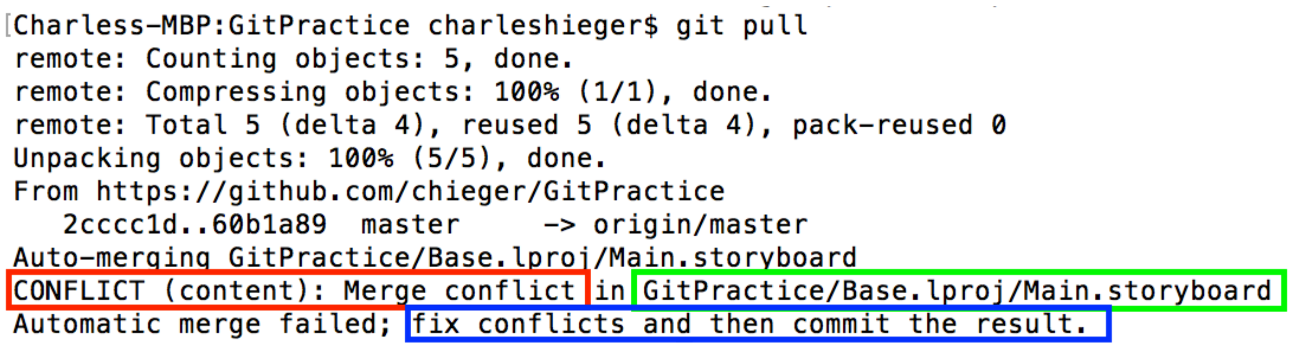
- In the instance of a Merge Conflict, you will probably get an error in the terminal when attempting to **push** your local changes.



```
Charless-MBP:GitPractice charleshieger$ git push
To https://github.com/chieger/GitPractice.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/chieger/GitPractice.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

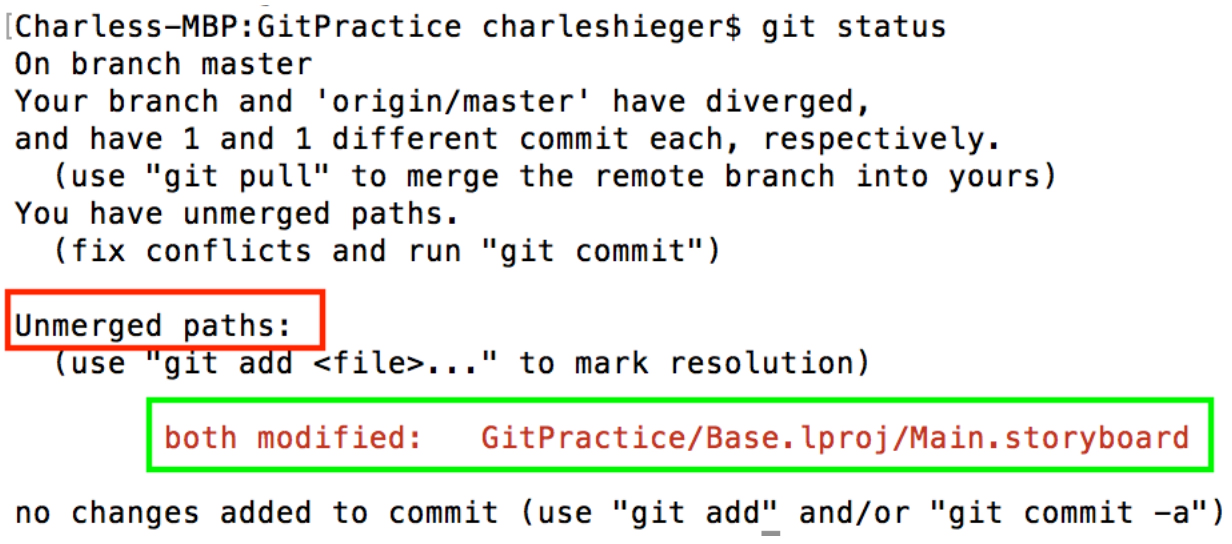
- As, the error message suggests in the "Hint", the next step is to perform a `git pull`. The pull will likely result in a message similar to the following.

    - NOTE: The error message confirms we have a Merge Conflict in the Main.storyboard file and tells us that we need to fix the conflict and commit the changes.



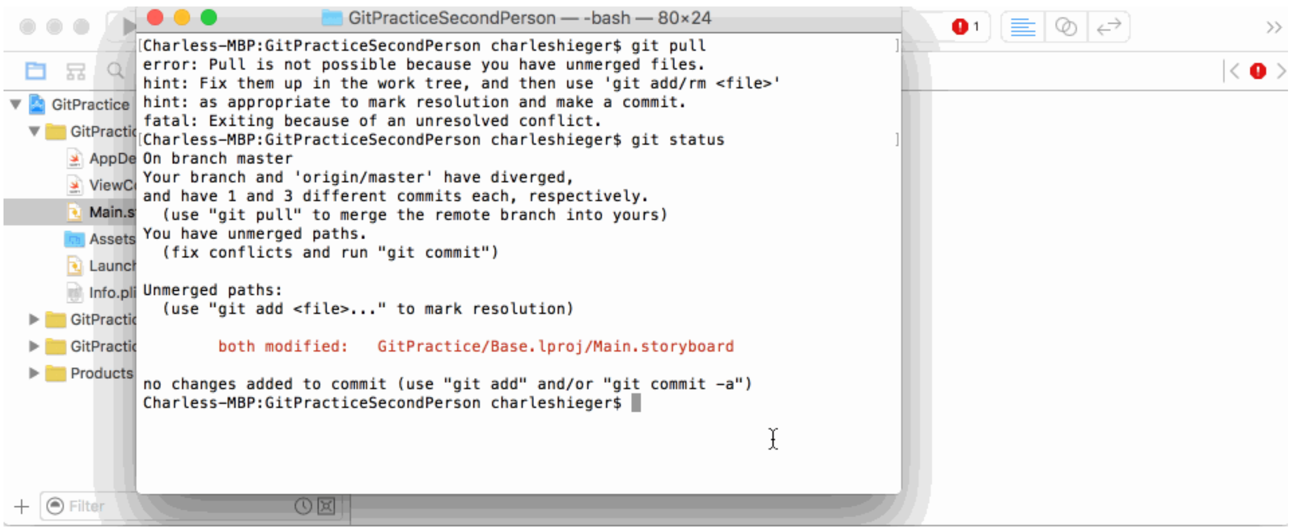
```
Charless-MBP:GitPractice charleshieger$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 5 (delta 4), reused 5 (delta 4), pack-reused 0
Unpacking objects: 100% (5/5), done.
From https://github.com/chieger/GitPractice
   2cccc1d..60b1a89  master     -> origin/master
Auto-merging GitPractice/Base.lproj/Main.storyboard
CONFLICT (content): Merge conflict in GitPractice/Base.lproj/Main.storyboard
Automatic merge failed; fix conflicts and then commit the result.
```

- Use `git status` to confirm which file(s) are causing the conflict.



```
Charless-MBP:GitPractice charleshieger$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commit each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   GitPractice/Base.lproj/Main.storyboard

no changes added to commit (use "git add" and/or "git commit -a")
```

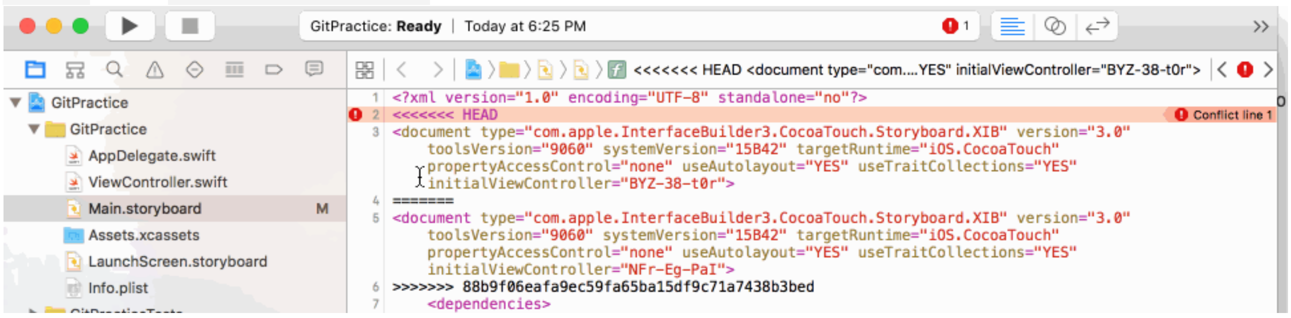
- Use `open /yourfileHere/` to open up the file in conflict; this will open up the file in Xcode.
  - NOTE: In this particular example, the conflict is in the Main.storyboard file. We are usually looking at the Interface Builder view of this file, so to reveal the underlying code, right-click on the `Main.storyboard` file and choose, **Open As -> Source Code**.
-



```
Charless-MBP:GitPracticeSecondPerson charleshieger$ git pull
error: Pull is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
Charless-MBP:GitPracticeSecondPerson charleshieger$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 3 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   GitPractice/Base.lproj/Main.storyboard

no changes added to commit (use "git add" and/or "git commit -a")
Charless-MBP:GitPracticeSecondPerson charleshieger$
```

- Identify the conflicting lines of code; they will be surrounded by, `<<<<<<< HEAD >>>>>>>bunchOfNumbersAndLetters` and separated by `=======`.

- Choose one to keep and one to delete. Do your best to figure out what the differences are; in this example, we see that the initialViewController is set to a different ID in each, `"BYZ-38-t0r"` vs. `"NFr-Eg-PaI"`.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<<<<<<< HEAD
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0"
    toolsVersion="9060" systemVersion="15B42" targetRuntime="iOS.CocoaTouch"
    propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES"
    initialViewController="BYZ-38-t0r">
=======
<document type="com.apple.InterfaceBuilder3.CocoaTouch.Storyboard.XIB" version="3.0"
    toolsVersion="9060" systemVersion="15B42" targetRuntime="iOS.CocoaTouch"
    propertyAccessControl="none" useAutolayout="YES" useTraitCollections="YES"
    initialViewController="NFr-Eg-PaI">
>>>>>>> 88b9f06eafa9ec59fa65ba15df9c71a7438b3bed
    <dependencies>
```

- Clean your Project, cmd + shift + K
    - If your resolved conflict was in the Main.storyboard file, right + click on the storyboard file and choose **Open As -> Interface Builder - Storyboard**.
- Finally, **push** your local changes to the remote repository.
  git config --global user.name "Roman Ferrara"

git config --global user.email roman.ferrara@pinecrest.edu