Infrastructure as Code

References

- Infrastructure as Code, by Kief Morris, Published by O'Reilly Media, Inc., 2015
- https://en.wikipedia.org/wiki/Infrastructure_as_Code
- https://help.github.com/articles/set-up-git/
- https://guides.github.com/activities/hello-world/
- http://product.hubspot.com/blog/git-and-github-tutorial-for-beginners
- http://rogerdudler.github.io/git-guide/
- https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes
- https://help.github.com/articles/caching-your-github-password-in-git/#platform-linux
- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
- https://git-scm.com/docs/git-remote

Infrastructure as Code (IaC)

- What is IaC?
- From the Wiki:

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

…

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Infrastructure as Code (IaC)

- .From Infrastructure as Code, Morris

Infrastructure as code is an approach to infrastructure automation based on practices from software development.

It emphasizes consistent, repeatable routines for provisioning and changing systems and their configuration.

Changes are made to definitions and then rolled out to systems through unattended processes that include thorough validation.

The principles and practices of infrastructure as code can be applied to infrastructure whether it runs on cloud, virtualized systems, or even directly on physical hardware.

Infrastructure as Code (IaC)

- Since we are writing code/scripts to automate reproducible deployment processes we need a place to save and version our code/scripts.

- We will do what many people do – use Git and GitHub

Git and GitHub

- **Git is a version control system.** You create repositories, add/update/delete/version files.
  - Git works locally on your machine
  - Each Git project has a root folder

- **GitHub is a code hosting platform for version control and collaboration**.

- Git can work with GitHub, you can "commit","pull", etc. change between your local Git an GitHub

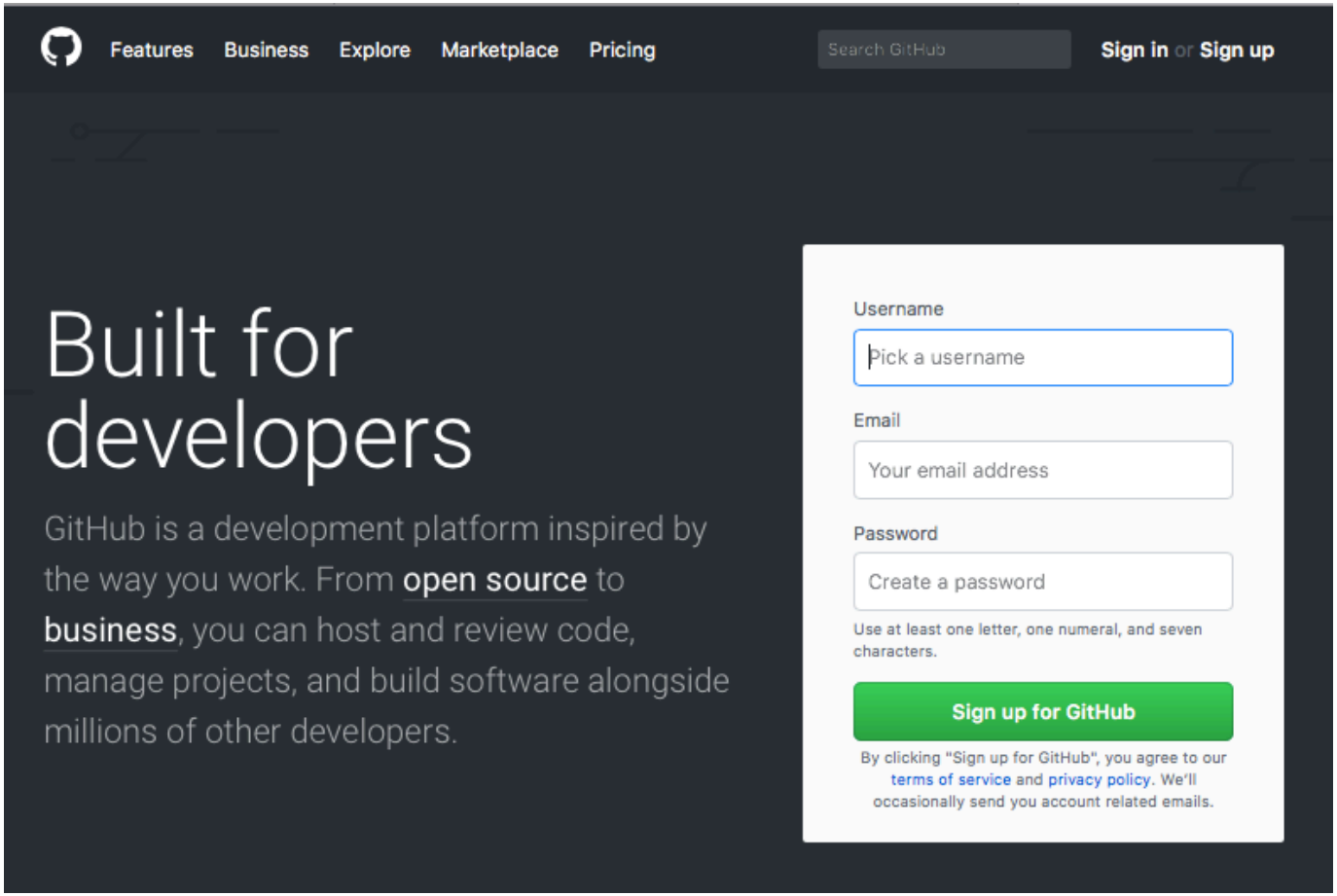
Git and GitHub

- The first step is to setup Git on your machine to manage locally hosted repositories.
- In this class we will use the Git command line.
- There is also a Git desktop, however for learning purposes your instructor strongly recommends using the Git command line in this class.

Signup for GitHub

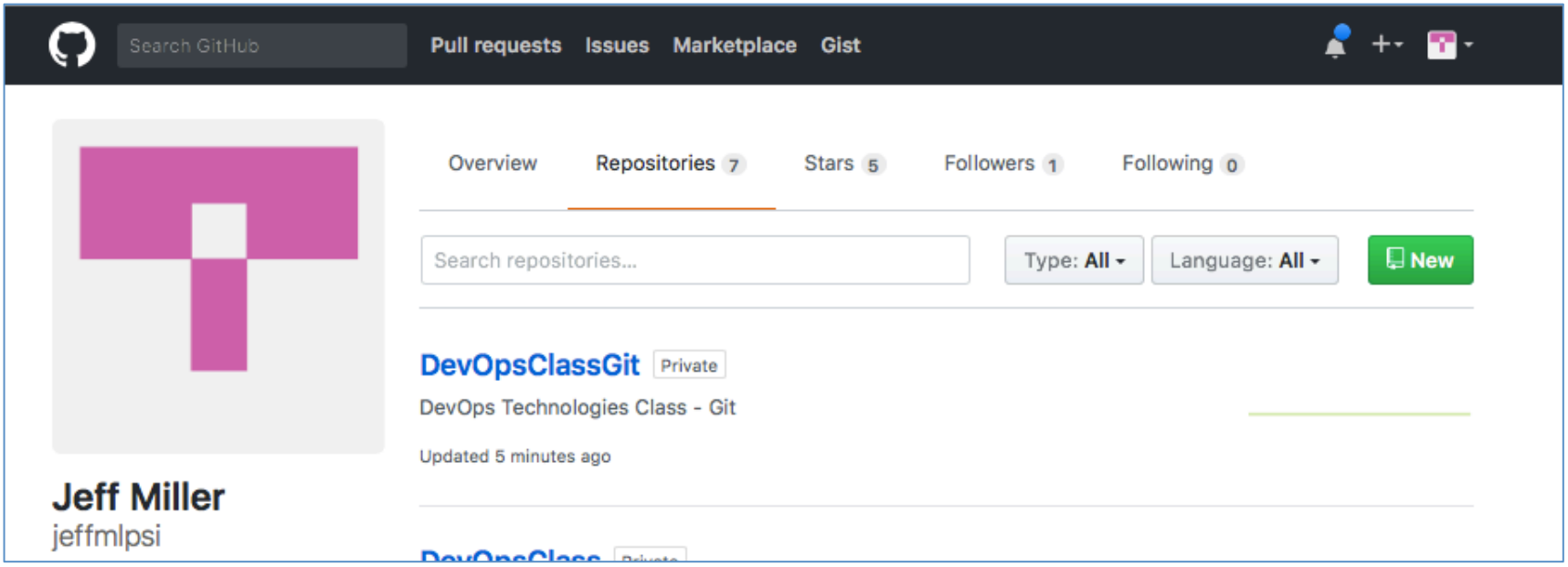
- You can use Git without GitHub, however - let's sign up for GitHub before we install Git

Signup for GitHub



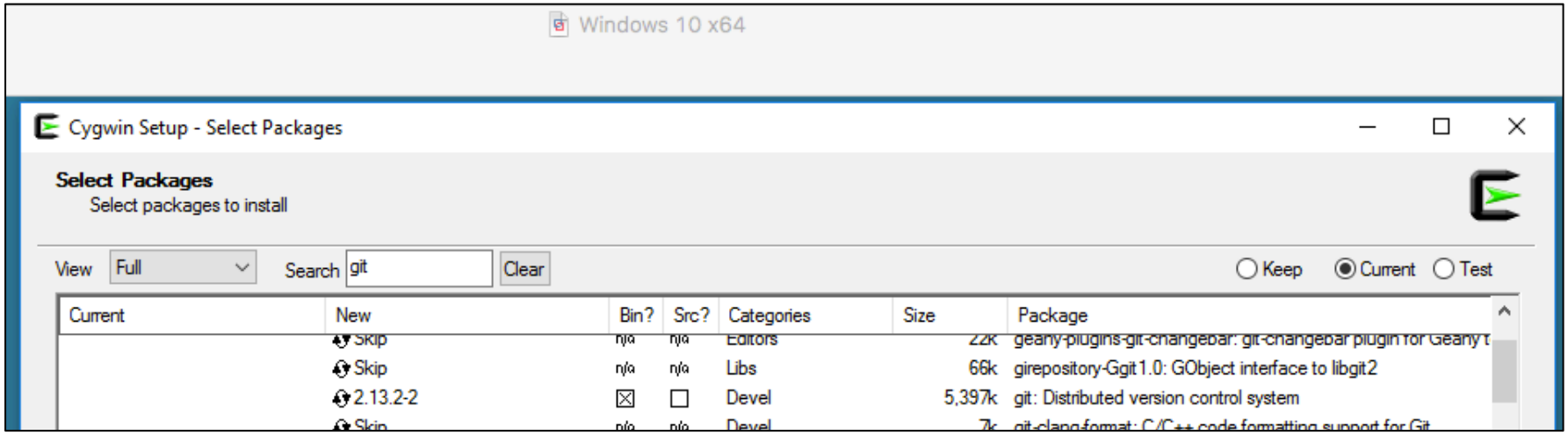
https://github.com

Create A Repository in GitHub



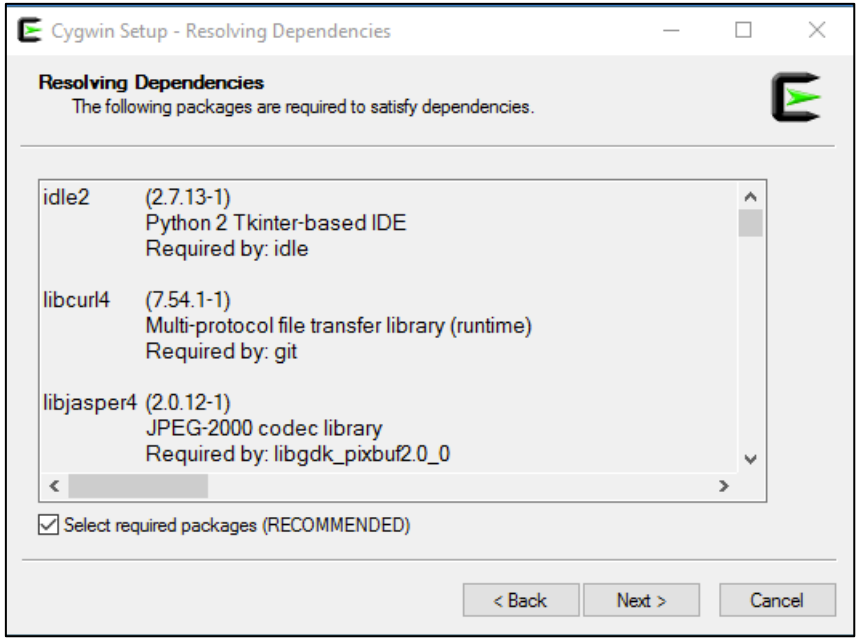
Create a new repository named DevOpsClassGit in your GitHub.

Install Git in Cygwin



- run the cygwin installer
- make sure "full" is selected next to View
- type in git
- select "git: Distributed version control system:"
- click Next at the bottom

Install Git in Cygwin



- You may see the Resolving Dependencies window,
  - if so, make sure "Select required packages" is checked.
  - NOTE: on your system you may see different packages
- Hit Next

Install Git on Mac

From: https://gist.github.com/derhuerst/1b15ff4652a867391f03

a) if you have not done so, install Homebrew, in a terminal run the following commands (it may take awhile):

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew doctor
```

b) using homebrew install Git:

```
brew install git
```

Install Git on Linux

see: https://gist.github.com/derhuerst/1b15ff4652a867391f03#file-linux-md

Git Basics

- After you install Git, you need to do a "first-time" setup
  - Create you (local) Git user name
  - Enter your email
  - Optionally: set a default editor for Git to use

Git Basics

```
-bash
~>git config --global user.name "Jeff Miller"
~>git config --global user.email jeffmlpsi@gmail.com
~>git config --list | grep "user."
user.name=Jeff Miller
user.email=jeffmlpsi@gmail.com
```

The image above shows setting user.name and user.email using

Git config

Git Basics

- As mentioned in a previous slide – Git works locally.

- Git stores "snapshots" of files. Traditional Version Control Systems typically store files and manage versions/revisions of files

Git Basics

- When your save (commit) changes Git store a snapshot of your files.

- If a file has not changed, Git does NOT store it again. Git maintains a "link" back to the file.

Git Basics

- The basic Git model is as follows:
  - You can <u>create a new repository</u>
  - Or **checkout** files from a <u>repository</u>

  - Checking out files creates a **working tree** (a root folder and whatever subfolders are added/ included).
    - The **root folder** of the working tree is the **working directory** for that project checkout/creation

Git Basics

- The basic Git model continued
  - When you modify files that are placed into a **modified** state
    - Files in a modified state are NOT saved yet
  - Next you put modified files into a **staged** state.
    - Staged files are NOT saved yet, but because they are "staged" they will be saved when you commit.
  - To save staged files in the local Git Database − you **commit** the files

Git Basics

- The following slides will show some basic git usage examples

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ ls
catinfo.txt
JeffMacBookPro13:git-demo-01 jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
```

First, a directory named **git-demo-01** was created.

A simple text file named **catinfo.txt** was placed into the folder.

Git Basics: git init

```
JeffMacBookPro13:git-demo-01 jeff$ git init
Initialized empty Git repository in /Users/jeff/DevOps-Tech/devops-notes/git/git
-demo-01/.git/
JeffMacBookPro13:git-demo-01 jeff$ ls -la
total 8
drwxr-xr-x@  4 jeff  staff  136 Jul 22 14:54 .
drwxr-xr-x@  4 jeff  staff  136 Jul 22 14:44 ..
drwxr-xr-x@ 10 jeff  staff  340 Jul 22 14:54 .git
-rw-r--r--@  1 jeff  staff   50 Jul 22 14:51 catinfo.txt
```

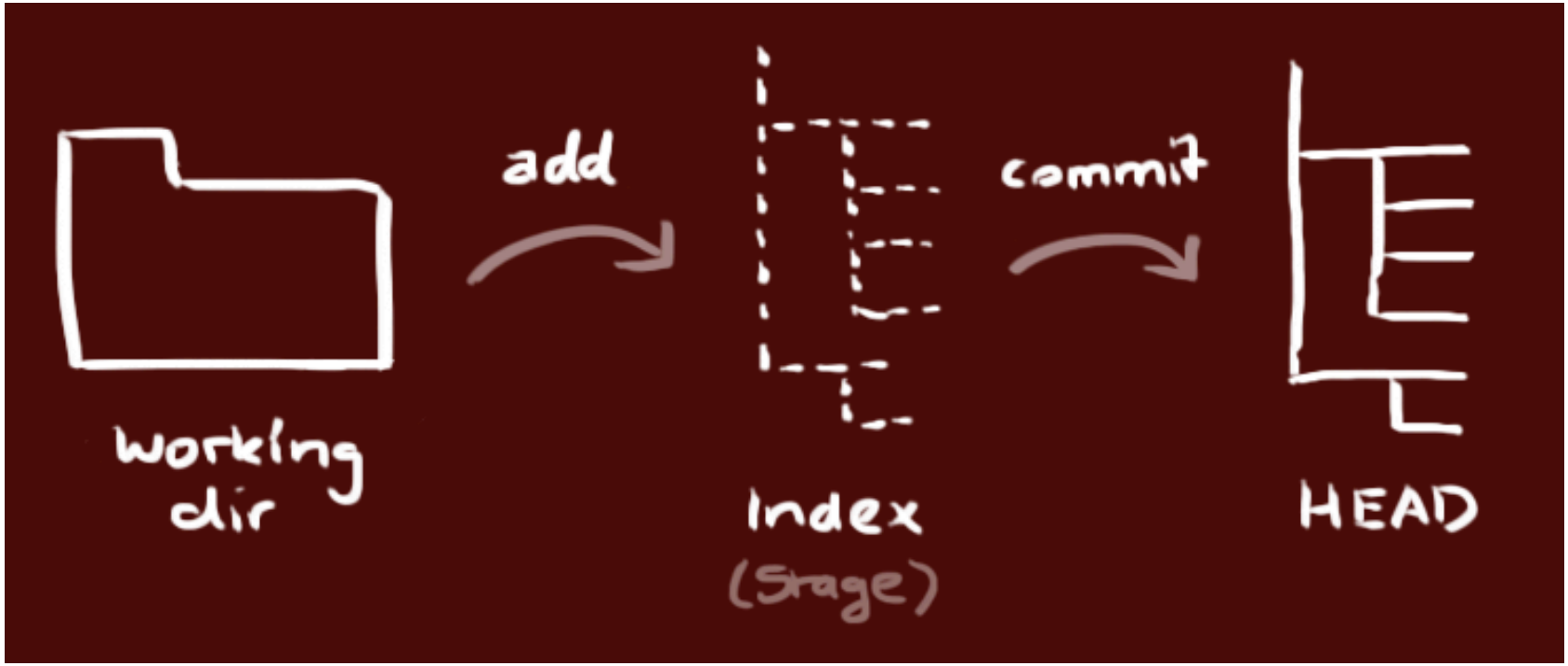
- We ran **git init** to create a new empty git repository
- git created a **.git** folder

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ ls -l .git
total 24
-rw-r--r--@  1 jeff  staff   23 Jul 22 14:54 HEAD
drwxr-xr-x@  2 jeff  staff   68 Jul 22 14:54 branches
-rw-r--r--@  1 jeff  staff  137 Jul 22 14:54 config
-rw-r--r--@  1 jeff  staff   73 Jul 22 14:54 description
drwxr-xr-x@ 11 jeff  staff  374 Jul 22 14:54 hooks
drwxr-xr-x@  3 jeff  staff  102 Jul 22 14:54 info
drwxr-xr-x@  4 jeff  staff  136 Jul 22 14:54 objects
drwxr-xr-x@  4 jeff  staff  136 Jul 22 14:54 refs
```

The contents of the .git directory are used by git to manage the repository

Git Basic: The Model



This image from: http://rogerdudler.github.io/git-guide/ illustrates the git model:

- **git add** – adds files into the "**staging**" area called **index**
- **git commit** – takes the files in staging and writes (commits) them into the top/head of the repository

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git add catinfo.txt
```

The command above adds catinfo.txt into the index (staging area)

```
JeffsMacBookPro:git-demo-01 jeffm$ git status
On branch master

Initial commit

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

        new file:   catinfo.txt
```

git status – displays the working tree status. Here is it telling us that catinfo.txt is **staged**, but not comitted.

See next slide

Git Basics

```
GIT-STATUS(1)                    Git Manual                    GIT-STATUS(1)

NAME
       git-status - Show the working tree status

SYNOPSIS
       git status [<options>...] [--] [<pathspec>...]

DESCRIPTION
       Displays paths that have differences between the index file and the
       current HEAD commit, paths that have differences between the working
       tree and the index file, and paths in the working tree that are not
       tracked by Git (and are not ignored by gitignore(5)). The first are
       what you would commit by running git commit; the second and third are
       what you could commit by running git add before running git commit.
```

git status –

shows the status of the working tree.

Displayed filed that are different between the **index/staging area** and the **HEAD** commit (see branching below).

And paths that are in the working tree that are NOT tracked by bit.

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git commit -m "First Git Commit"
[master (root-commit) 6b2105e] First Git Commit
 1 file changed, 5 insertions(+)
 create mode 100644 catinfo.txt
```

Using **git commit** to commit staged changes into the current git repository

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ echo 'dogs are not cats' > doginfo.txt
JeffMacBookPro13:git-demo-01 jeff$ cat doginfo.txt
dogs are not cats
JeffMacBookPro13:git-demo-01 jeff$ echo 'yet more cat info' >> catinfo.txt
JeffMacBookPro13:git-demo-01 jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
```

- Create a new file named doginto.txt
- Add a line into file catinfo.txt

In the next slide we will run **git status** again

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git status
On branch 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 directory)

        modified:   catinfo.txt

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

        doginfo.txt

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

Git status is telling us:

- File catinfo.txt has been modified, but has NOT been staged for commit
- File doginto.txt is not being tracked, meaning git is not managing the file.

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git add -u
JeffMacBookPro13:git-demo-01 jeff$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   catinfo.txt

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

        doginfo.txt
```

- **git add -u** #only stage files with modifications that are already known to the index/ starging-area

- git status #show us that catinfo.txt is staged and doginfo.txt is not tracked

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git commit -m 'second commit'
[master 82c0a88] second commit
 1 file changed, 1 insertion(+)
JeffMacBookPro13:git-demo-01 jeff$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        doginfo.txt

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

- **git commit −m 'second commit'** #commit all staged changes, add a commit message
- **git status** #shows us the staging area is empty (meaning changes have been committed) and that doginfo.txt is still untracked.

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git log
commit 82c0a88ebf20e49892b7155fee31d79f8d66fc6d
Author: jeffmlpsi <jeffmlpsi@gmail.com>
Date:   Thu Jul 27 12:52:44 2017 -0700

    second commit

commit 6b2105e9e8cc4bf6d58d81a178bc1d2586c641fe
Author: jeffmlpsi <jeffmlpsi@gmail.com>
Date:   Thu Jul 27 12:34:23 2017 -0700

    First Git Commit
```

git log #displays our commit history

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git log --pretty=oneline
82c0a88ebf20e49892b7155fee31d79f8d66fc6d second commit
6b2105e9e8cc4bf6d58d81a178bc1d2586c641fe First Git Commit
JeffMacBookPro13:git-demo-01 jeff$ git log --name-status
commit 82c0a88ebf20e49892b7155fee31d79f8d66fc6d
Author: jeffmlpsi <jeffmlpsi@gmail.com>
Date:   Thu Jul 27 12:52:44 2017 -0700

    second commit

M       catinfo.txt

commit 6b2105e9e8cc4bf6d58d81a178bc1d2586c641fe
Author: jeffmlpsi <jeffmlpsi@gmail.com>
Date:   Thu Jul 27 12:34:23 2017 -0700

    First Git Commit

A       catinfo.txt
JeffMacBookPro13:git-demo-01 jeff$ git log --name-status --pretty=oneline
82c0a88ebf20e49892b7155fee31d79f8d66fc6d second commit
M       catinfo.txt
6b2105e9e8cc4bf6d58d81a178bc1d2586c641fe First Git Commit
A       catinfo.txt
```

- **git log –pretty=oneline**
- **git log –name-status**
- **git log –name-status –pretty=oneline**

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git log --pretty=oneline
82c0a88ebf20e49892b7155fee31d79f8d66fc6d second commit
6b2105e9e8cc4bf6d58d81a178bc1d2586c641fe First Git Commit
JeffMacBookPro13:git-demo-01 jeff$ git show --name-only --oneline 82c0a88ebf20e49892b7155fee31d79f8d66fc6d
82c0a88 second commit
catinfo.txt
JeffMacBookPro13:git-demo-01 jeff$ git show --name-only --oneline 6b2105e9e8cc4bf6d58d81a178bc1d2586c641fe
6b2105e First Git Commit
catinfo.txt
```

- **git log --pretty=oneline** #display commit history on one line, display commit id/hash
- **git show --name-only hash-id** #display files committed in a specific commit

Git Basics

Often we want to keep untracked files around and we do NOT want Git to track them.

To tell git to ignore untracked files for a specific repository, you can add the file or file pattern into file .git/info/exclude.

```
JeffMacBookPro13:git-demo-01 jeff$ echo doginfo.txt >> .git/info/exclude
JeffMacBookPro13:git-demo-01 jeff$ git status
On branch master
nothing to commit, working directory clean
```

Notice that after adding doginfo.txt into the exclude file in ./.git/info/exclude it no longer shows up as an untracked file in git status.

Git Basics

- Let's add our local Git repo into a remote repository in GitHub
- The following slides show a new repository named DevOpsClassGit I created in GitHub

Git Basics

jeffmlpsi / DevOpsClassGit Private

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/jeffmlpsi/DevOpsClassGit.git

We recommend every repository include a README, LICENSE, and .gitignore.

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

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

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

```
git remote add origin https://github.com/jeffmlpsi/DevOpsClassGit.git
git push -u origin master
```

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git remote add origin https://github.com/jeffmlpsi/DevOpsClassGit.git
```

Add a remote repository using:

git remote add name-of-repos git-url

```
JeffMacBookPro13:git-demo-01 jeff$ git remote -v
origin  https://github.com/jeffmlpsi/DevOpsClassGit.git (fetch)
origin  https://github.com/jeffmlpsi/DevOpsClassGit.git (push)
```

List remote repositories

Next, let's try to push our local git repo into GitHub

Git Basics

```
JeffMacBookPro13:git-demo-01 jeff$ git push -u origin master
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 470 bytes | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jeffmlpsi/DevOpsClassGit.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Git Basics

jeffmlpsi / DevOpsClassGit Private

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

DevOps Technologies Class - Git

Edit

Add topics

2 commits | 1 branch | 0 releases | 0 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jeffmlpsi second commit — Latest commit 82c0a88 2 hours ago

catinfo.txt — second commit — 2 hours ago

Add a README with an overview of your project. — Add a README

Git Basics

- Now, lets clone this repository into another directory – as a different user

  Here are the commands I previously ran (you need to replace these with you name and email)

```
#global config
git config --global user.name "Jeff Miller"
git config --global user.email "jeffmlpsi@gmail.com"
```

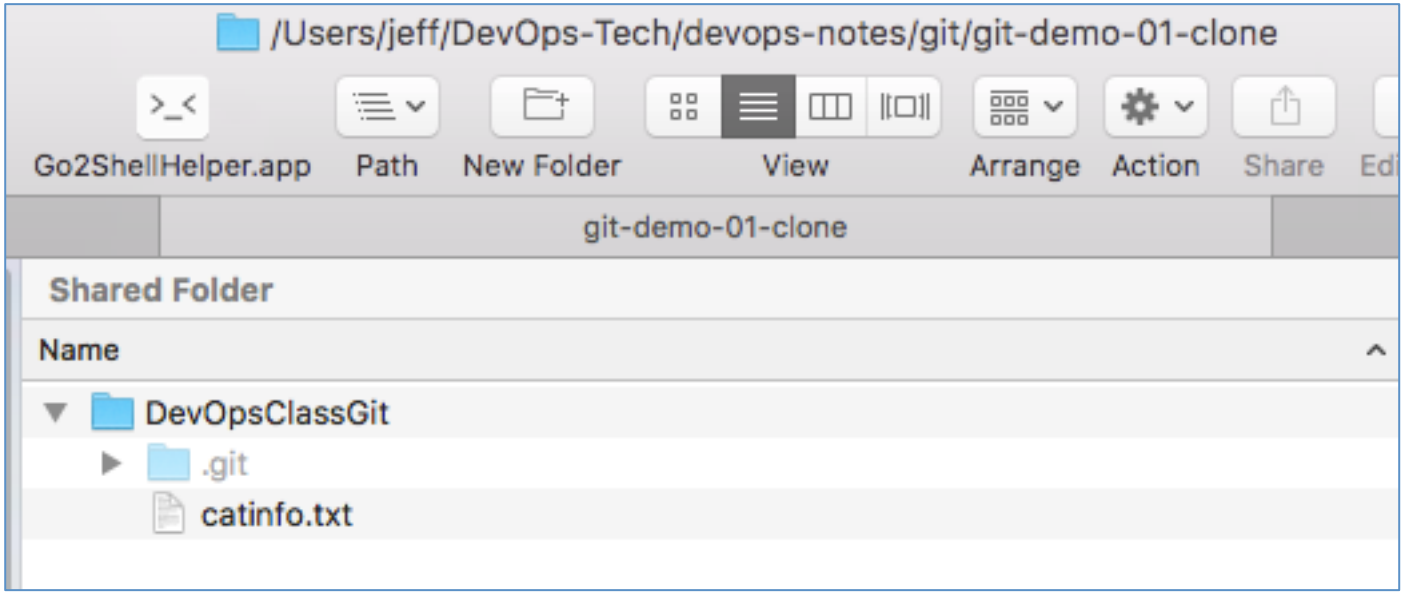
Git Basics

- Create directory git-demo-01-clone

- Clone the GitHub repo:

```
JeffMacBookPro13:git-demo-01-clone jeff$ git clone https://github.com/jeffmlpsi/DevOpsClassGit.git
Cloning into 'DevOpsClassGit'...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 1), reused 6 (delta 1), pack-reused 0
Unpacking objects: 100% (6/6), done.
Checking connectivity... done.
```

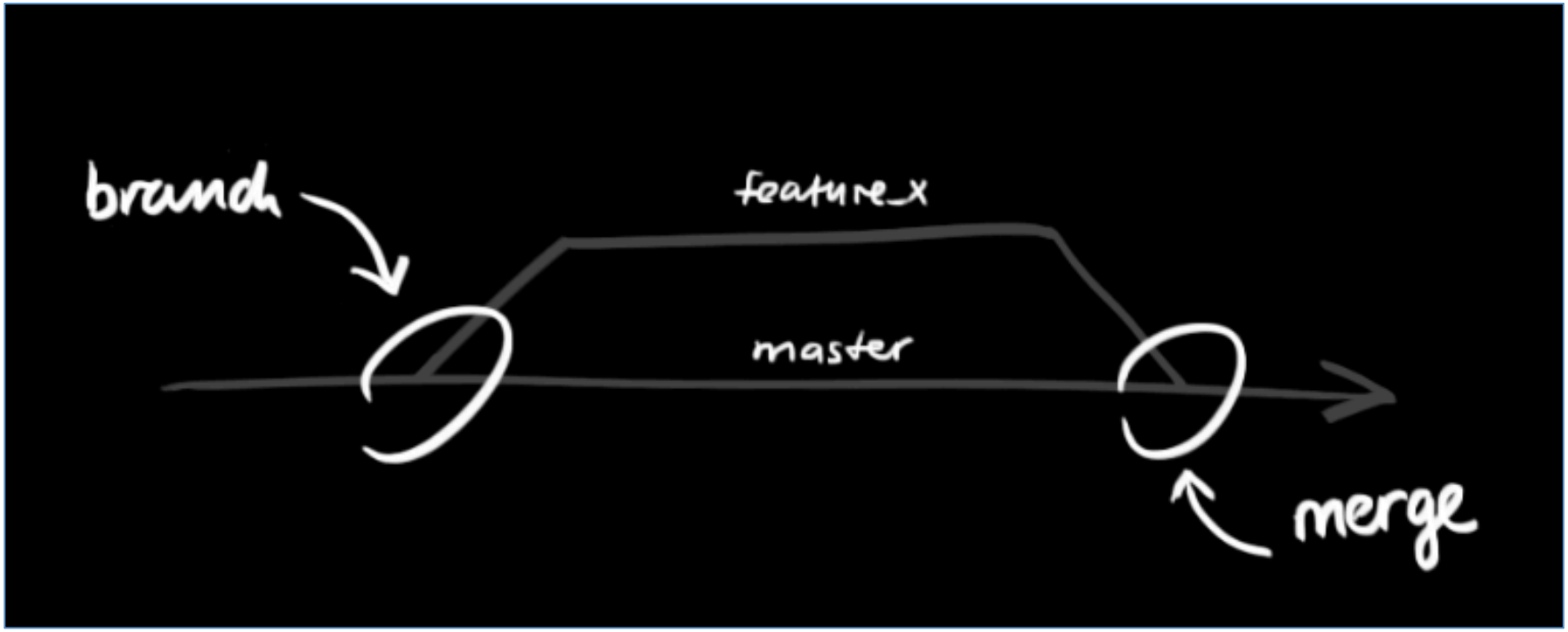
Git Basics


/Users/jeff/DevOps-Tech/devops-notes/git/git-demo-01-clone
Go2ShellHelper.app
Path
New Folder
View
Arrange
Action
Share
git-demo-01-clone
Shared Folder
Name
DevOpsClassGit
.git
catinfo.txt

Git Basics

- The Git model usually involves
  - working on a "branch" of a repository
  - making changes in that branch
  - Pushing the branch into the Git repo
  - And, usually, merging the changes back into the the "origin" (main)
    - in git, origin is a local alias for a remote repository
    - Often, our default branch is master, origin refers to origin/master

Git Basics



The image above, from http://rogerdudler.github.io/git-guide/ - shows two a branch named feature_x, branched from the master branch, and merged back in

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git checkout -b branch01
Switched to a new branch 'branch01'
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* branch01
  master
```

- git checkout −b branch01

  creates and switches to a new branch named branch01

- git branch

  lists branches

- **When working in a branch, changes made in that branch are NOT visible in other branches until the changes are merged**

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
  branch01
* master
JeffMacBookPro13:DevOpsClassGit jeff$ git checkout branch01
Switched to branch 'branch01'
```

Once a branch is created you can switch to it using – **git checkout** branch-name

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* branch01
  master
JeffMacBookPro13:DevOpsClassGit jeff$ echo 'yet another line of cats!' >> catinfo.txt
JeffMacBookPro13:DevOpsClassGit jeff$ git add .
JeffMacBookPro13:DevOpsClassGit jeff$ git status
On branch branch01
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   catinfo.txt
```

- git branch #see which branch we are on
- echo "…" >> catinfo.txt #append a line into catinto.txt
- git status #see staged files

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git commit -m "commit on branch01"
[branch01 97adf2a] commit on branch01
 1 file changed, 2 insertions(+)
```

git commit -m "commit on branch01"

#commit staged changes into the local repo

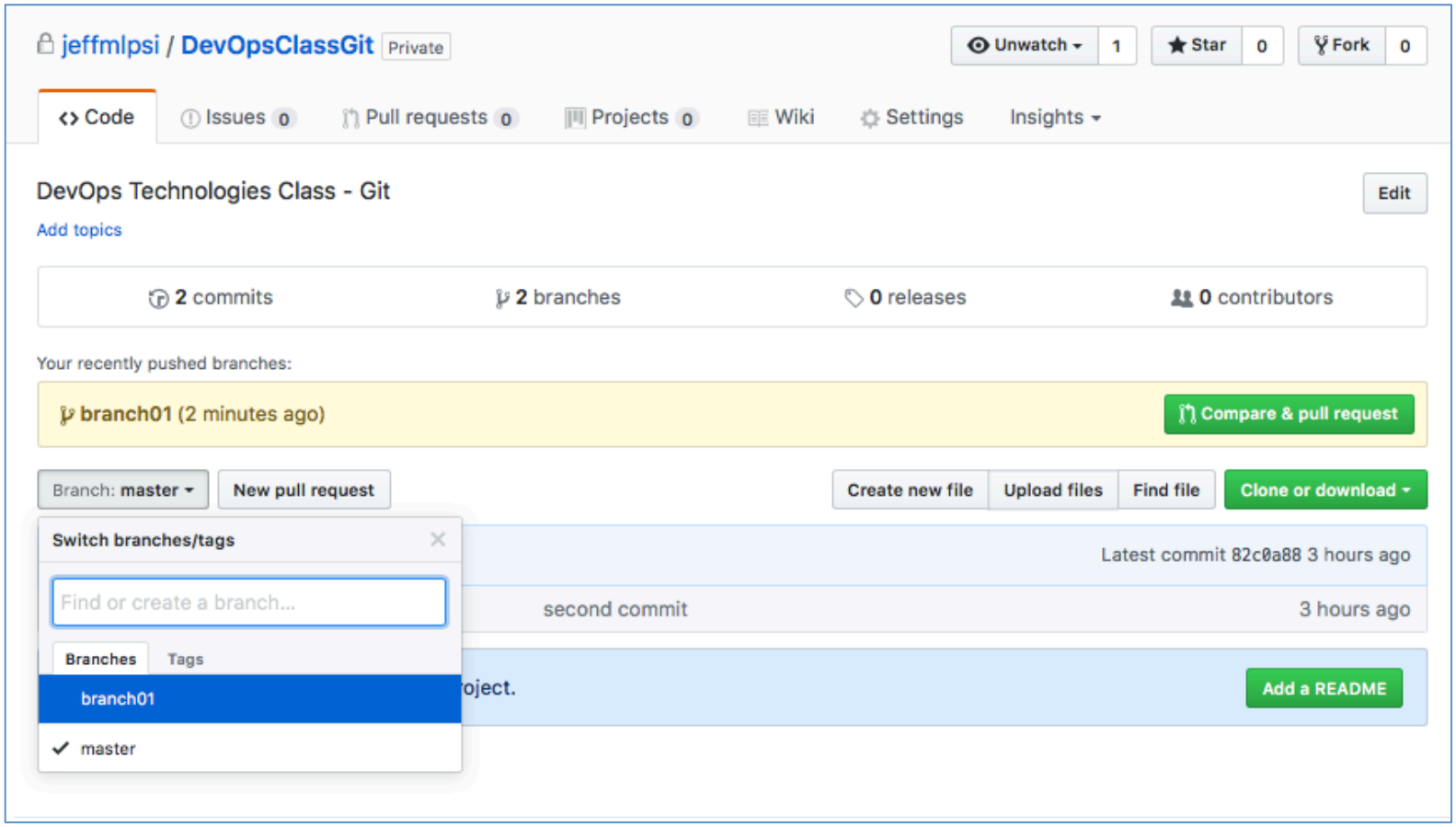
Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git push origin branch01
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 308 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/jeffmlpsi/DevOpsClassGit.git
 * [new branch]      branch01 -> branch01
```

Use **git push origin branch01** to push our changes into GitHub in branch01

Again, origin is an alias on your machine to the current remote repo. In this case origin is a shorthand (an alias) for remote repository origin/branch01

Git Basics



Notice after the previous push – our new branch is in GitHub

```
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* branch01
  master
JeffMacBookPro13:DevOpsClassGit jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
yet another line of cats!
yet another line of cats!
JeffMacBookPro13:DevOpsClassGit jeff$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
JeffMacBookPro13:DevOpsClassGit jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
```

This commands issued in the image above demonstrate how changes made in one branch are not visible in another branch until they are merged (which we have not done yet).

Notice in the master branch, the changes we made in branch01 are not visible

Git Basics

- Let's try a merge, first
  - Create a new directory named git-demo-01-merge
  - cd into this directory
  - Clone from GitHub

```
JeffMacBookPro13:git-demo-01-merge jeff$ git clone https://github.com/jeffmlpsi/DevOpsClassGit.git
Cloning into 'DevOpsClassGit'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 2), reused 8 (delta 1), pack-reused 0
Unpacking objects: 100% (9/9), done.
Checking connectivity... done.
JeffMacBookPro13:git-demo-01-merge jeff$ cd DevOp*
JeffMacBookPro13:DevOpsClassGit jeff$ ls
catinfo.txt
JeffMacBookPro13:DevOpsClassGit jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
```

Git Basics

Before doing a merge, let's add a line in:

```
JeffMacBookPro13:DevOpsClassGit jeff$ echo 'this line is added in the master branch' >> catinfo.txt
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* master
JeffMacBookPro13:DevOpsClassGit jeff$
```

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git add .
JeffMacBookPro13:DevOpsClassGit jeff$ git commit -m 'line added into master branch'
[master fdc2d67] line added into master branch
 1 file changed, 1 insertion(+)
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* master
JeffMacBookPro13:DevOpsClassGit jeff$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 323 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/jeffmlpsi/DevOpsClassGit.git
   82c0a88..fdc2d67  master -> master
```

Since we made a change in the master branch, we need to:

- Stage it – **git add .**
- Commit the staged changes – **git commit –m "…"**
- Push the committed changes into GitHub – **git push origin master**

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* master
JeffMacBookPro13:DevOpsClassGit jeff$ git branch -r
  origin/HEAD -> origin/master
  origin/branch01
  origin/master
```

Notice, in our git-demo-01-merge/DevOpsClassGit directory, git does not "see" branch01.

git branch −r displays remote branches

Git Basics

- To this point we know we have added different lines in each branch, master and branch01.

- Let's do a merge between the master branch and branch01

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git branch -r
  origin/HEAD -> origin/master
  origin/branch01
  origin/master
JeffMacBookPro13:DevOpsClassGit jeff$ git merge branch01
merge: branch01 - not something we can merge

Did you mean this?
        origin/branch01
JeffMacBookPro13:DevOpsClassGit jeff$ git merge origin/branch01
Auto-merging catinfo.txt
CONFLICT (content): Merge conflict in catinfo.txt
Automatic merge failed; fix conflicts and then commit the result.
JeffMacBookPro13:DevOpsClassGit jeff$
```

Notice we had to do the nerge using "origin/branch01".

Also notice, Git could not auto-merge our changes. Remember − we added lines in each branch.

We can use **git diff** to see the differences:

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git diff origin/branch01 master
diff --git a/catinfo.txt b/catinfo.txt
index bcdac16..d38d3d6 100644
--- a/catinfo.txt
+++ b/catinfo.txt
@@ -4,5 +4,4 @@ cats can run
 cats can hide
 cats can eat
 yet more cat info
-yet another line of cats!
-yet another line of cats!
+this line is added in the master branch
```

Using **git diff origin/branch01 master** – we can see that:

- lines prefixed with "-" were added in branch01
- lines prefixed with "+" were added in the master branch

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
You have unmerged paths.
  (fix conflicts and run "git commit")

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

        both modified:   catinfo.txt

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

Running git status will show that both branches have changes

Git Basics

We can use **git mergetool** to help us with the merges:

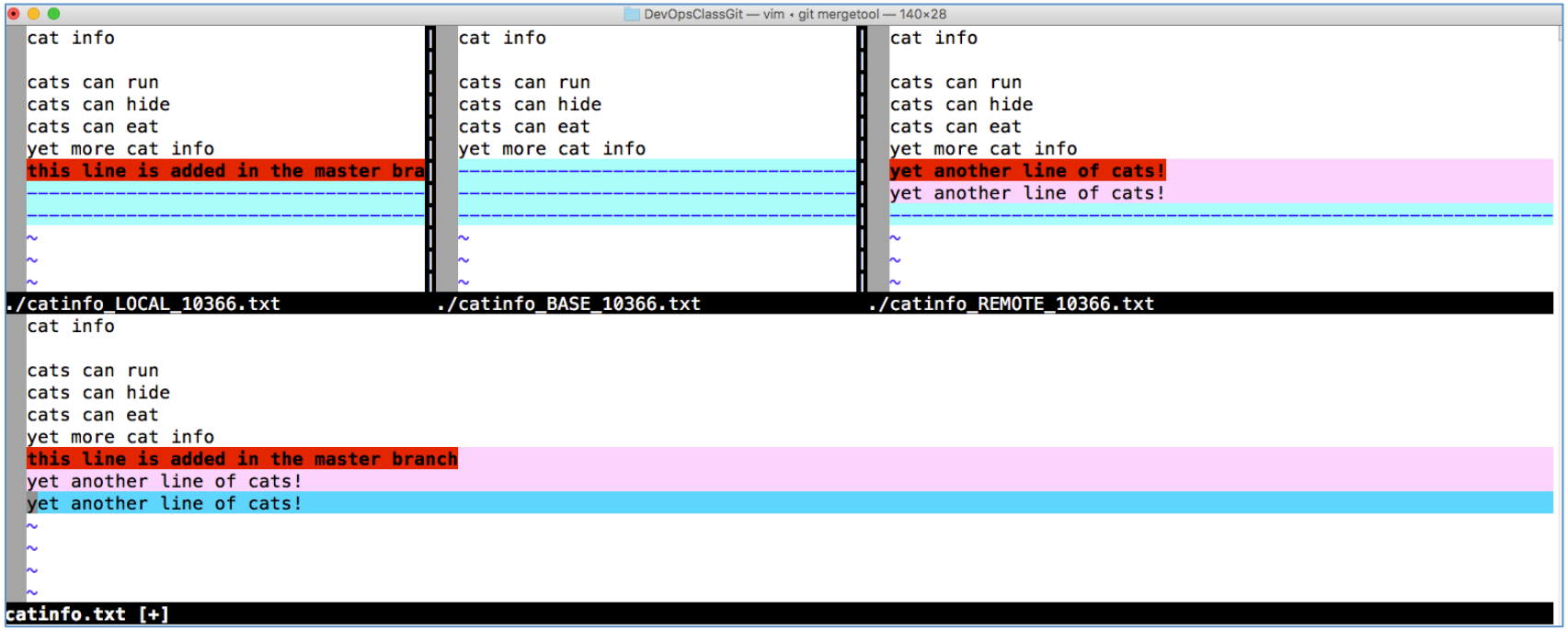
```
JeffMacBookPro13:DevOpsClassGit jeff$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff
Merging:
catinfo.txt

Normal merge conflict for 'catinfo.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

NOTE: the "mergetool" used by git has different defaults on different systems and is configurable.

Git Basics

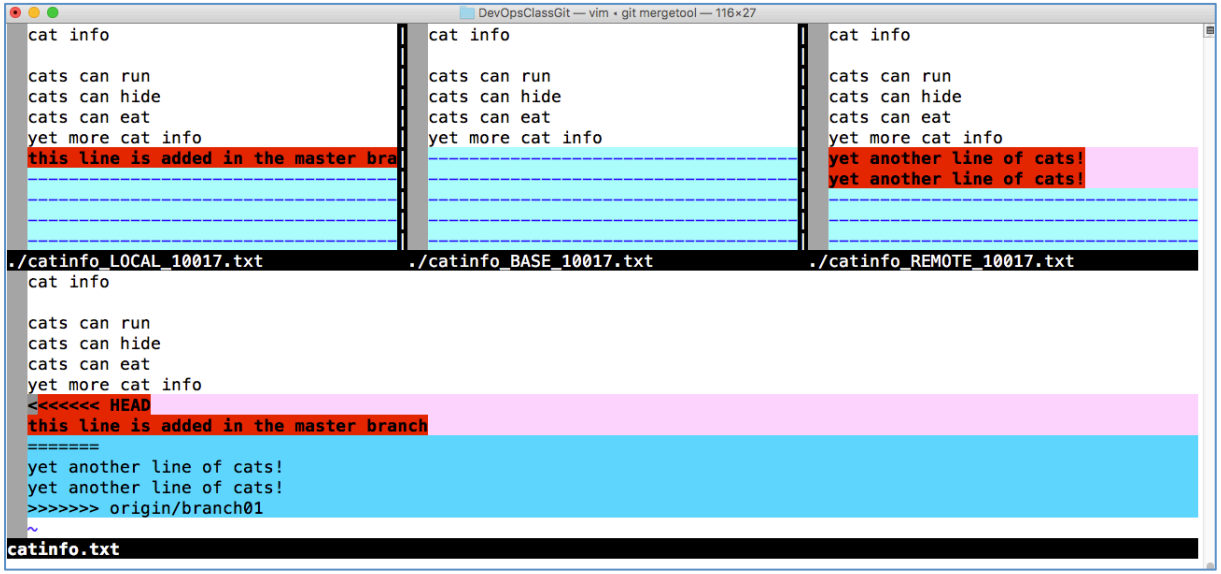
```
DevOpsClassGit — vim ◂ git mergetool — 116×27
cat info                                  | cat info                                  | cat info
                                          |                                           |
cats can run                              | cats can run                              | cats can run
cats can hide                             | cats can hide                             | cats can hide
cats can eat                              | cats can eat                              | cats can eat
yet more cat info                         | yet more cat info                         | yet more cat info
this line is added in the master bra      | ----------------------------------------- | yet another line of cats!
----------------------------------------- | ----------------------------------------- | yet another line of cats!
----------------------------------------- | ----------------------------------------- | ----------------------------------------
----------------------------------------- | ----------------------------------------- | ----------------------------------------
----------------------------------------- | ----------------------------------------- | ----------------------------------------
./catinfo_LOCAL_10017.txt                 | ./catinfo_BASE_10017.txt                  | ./catinfo_REMOTE_10017.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
<<<<<<< HEAD
this line is added in the master branch
=======
yet another line of cats!
yet another line of cats!
>>>>>>> origin/branch01
~
catinfo.txt
```

Git Basics

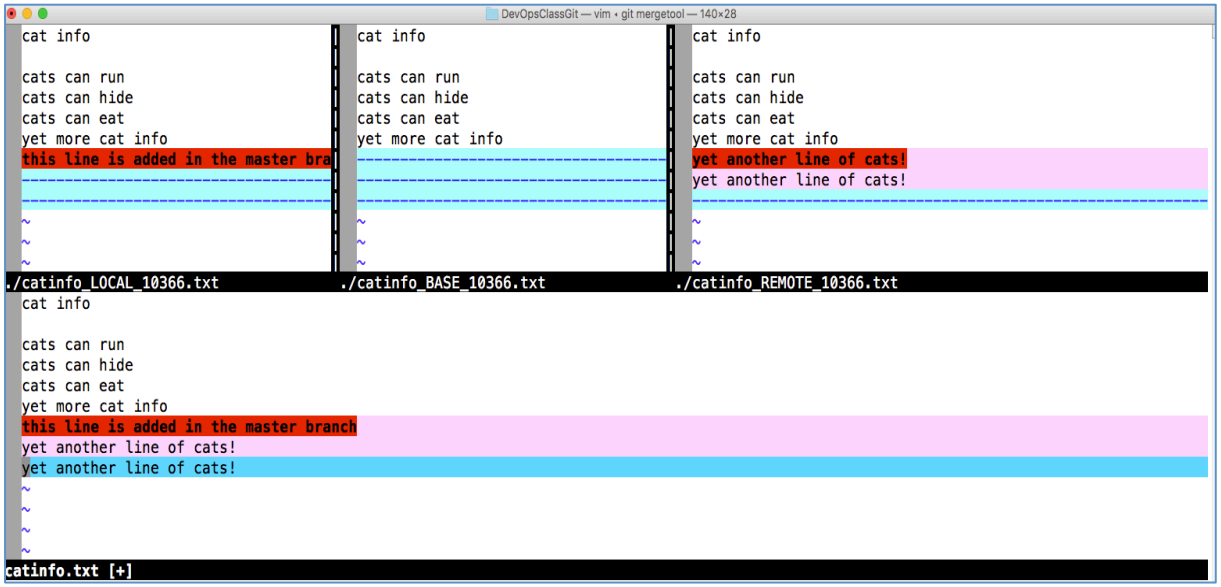


On my system, in the bottom window pane, I deleted the lines I did not want in the merge.

Again – the tool that comes up on your system may be different than what came up on my system



Before changes



After changes

Git Basics

After allowing git mergetool to do our merges **run git status**:

```
JeffMacBookPro13:DevOpsClassGit jeff$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

        modified:   catinfo.txt

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

        catinfo.txt.orig
```

Git Basics

```
JeffMacBookPro13:DevOpsClassGit jeff$ git commit -m "merged commit"
[master 9cf6276] merged commit
JeffMacBookPro13:DevOpsClassGit jeff$
```

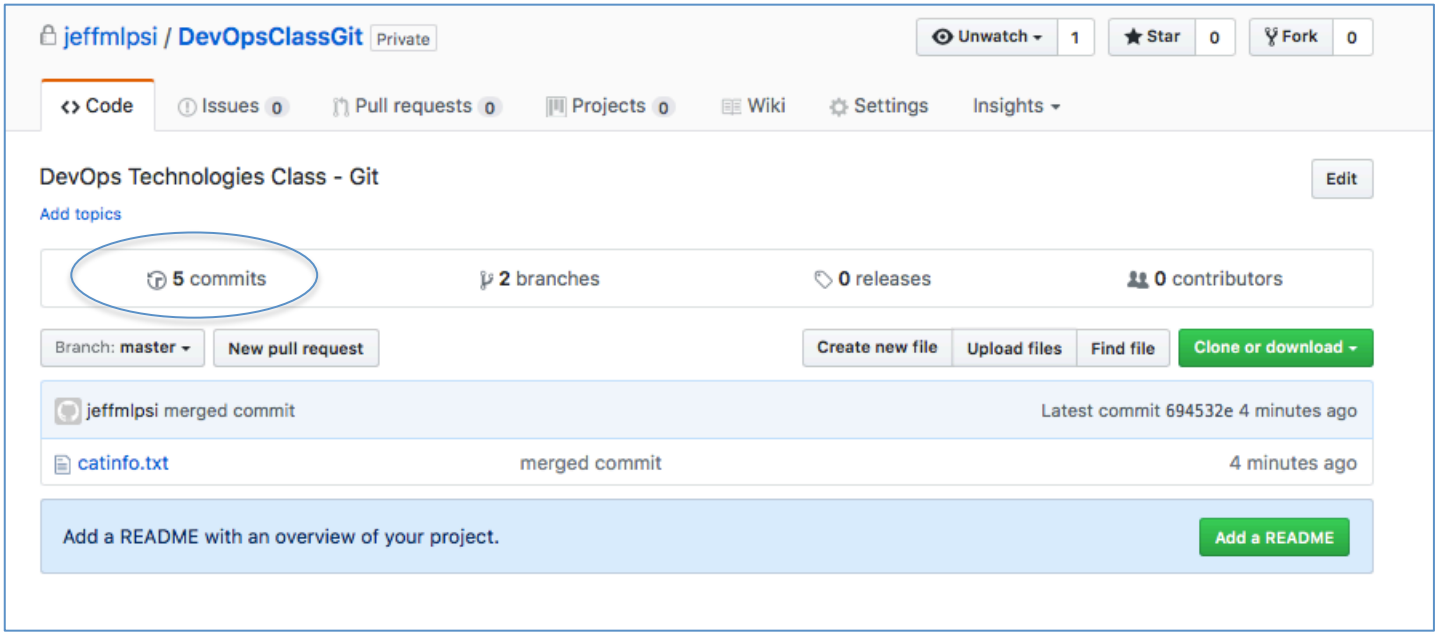
```
JeffMacBookPro13:DevOpsClassGit jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
this line is added in the master branch
yet another line of cats!
yet another line of cats!
```

Git Basics

Now that our merge between bracnh01 and master is resolved, let's push the merge to GitHub, into Master

```
JeffMacBookPro13:DevOpsClassGit jeff$ git branch
* master
JeffMacBookPro13:DevOpsClassGit jeff$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/jeffmlpsi/DevOpsClassGit.git
   fdc2d67..694532e  master -> master
```

Git Basics


jeffmlpsi / DevOpsClassGit Private
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Settings
Insights
DevOps Technologies Class - Git
Edit
Add topics
5 commits
2 branches
0 releases
0 contributors
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
jeffmlpsi merged commit
Latest commit 694532e 4 minutes ago
catinfo.txt
merged commit
4 minutes ago
Add a README with an overview of your project.
Add a README

Git Basics

jeffmlpsi / DevOpsClassGit Private

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

Branch: master

Commits on Jul 27, 2017

merged commit jeffmlpsi committed 5 minutes ago	694532e
line added into master branch jeffmlpsi committed an hour ago	fdc2d67
commit on branch01 jeffmlpsi committed 2 hours ago	97adf2a
second commit jeffmlpsi committed 4 hours ago	82c0a88
First Git Commit jeffmlpsi committed 5 hours ago	6b2105e

Newer | Older

Git Basics

One last check, lets go back into the orignal directory – git-demo-01 – and update that git repository to the latest in Master using **git pull**:

```
JeffMacBookPro13:git-demo-01 jeff$ git branch
* master
JeffMacBookPro13:git-demo-01 jeff$ git pull
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 1), reused 6 (delta 1), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/jeffmlpsi/DevOpsClassGit
   82c0a88..694532e  master     -> origin/master
Updating 82c0a88..694532e
Fast-forward
 catinfo.txt | 3 +++
 1 file changed, 3 insertions(+)
JeffMacBookPro13:git-demo-01 jeff$ cat catinfo.txt
cat info

cats can run
cats can hide
cats can eat
yet more cat info
this line is added in the master branch
yet another line of cats!
yet another line of cats!
```

Added in Master branch

Added in branch01

Git Basics

- There are many, many Git commands. You are not limited to using the Git commands shown in this presentation.

- You can run:

  git command-name −help

  to get documentation on Git commands

Git Basics: Other Useful Commands

- **git reset file-name** #unstages a file
- **git rm** … #remove files from working tree and staging areas − becareful, this deletes files
- **git show** … #get information on various git objects
- **git reset --hard HEAD~1** #undo last commit and remove all changes
- **git reset --soft HEAD~1** #reset HEAD branch to previous commit, but keeps changes
- **git branch −r** #list remote branch
- **git pull** #update current branch from remote repo