UNIVERSITY OF ALBERTA

Git and GitHub

CMPUT 301
Fall 2020

Akalanka Galappaththi

Git merge conflict demo is based on LN Wilson's lab worksheet (Dept. Math. & Computer Since, University of Lethbridge)

Shout out to the TAs

Ali Abdul Bangash	bangash@ualberta.ca
Wang Dong	wdong2@ualberta.ca
Hongyang (Aaron) Liu	hliu11@ualberta.ca
Michael Przystupa	przystup@ualberta.ca
Puer (Paul) Liu	puer@ualberta.ca
Ruiqin Pi	ruiqin@ualberta.ca
Zijun Wu	zijun4@ualberta.ca
Junyao Cui	junyao1@ualberta.ca
Xuechun Qiu	xqiu1@ualberta.ca
Dalton Ronan	dronan@ualberta.ca
Xiyuan Shen	xiyuan1@ualberta.ca
Candelario Gutierrez Gutierrez	cagutier@ualberta.ca
Md Tanzil Shahriar	mdtanzil@ualberta.ca

What is Git and GitHub

- Git is a version control tool.
- Keep track of changes (files).
- Revert back to previous state.

GitHub

What is Git and GitHub

- Distributed version control system.
- Offers source code management (Git).
- Many other (own) services.

Icons made by <a href="https://www.flaticon.com/authors/freepik" title="Freepik">Freepik</a> from <a href="https://www.flaticon.com/" title="Flaticon"> www.flaticon.com</a>

Installing Git

- Windows users: download and run the .exe file
- Mac users: (Homebrew package manager)
- Linux users

`https://git-scm.com/downloads`

Installation guide

`https://www.stanleyulili.com/git/how-to-install-git-bash-on-windows/`

!!! Make sure to select your preferred editor

Installing Git

- Windows users: download and run the .exe file
- Mac users: (Homebrew package manager)
- Linux users

```
https://git-scm.com/downloads
```

```
$ brew install git
$ git --version
```

Installing Git

- Windows users: download and run the .exe file
- Mac users: (Homebrew package manager)
- Linux users

```
$ sudo apt update
$ sudo apt install git
$ git --version
```

```
$ sudo yum install git
$ git --version
```

Configure Git

Set up your username and email

```
$ git config --global user.name “John Smith”
$ git config --global user.email js@gmail.com
```

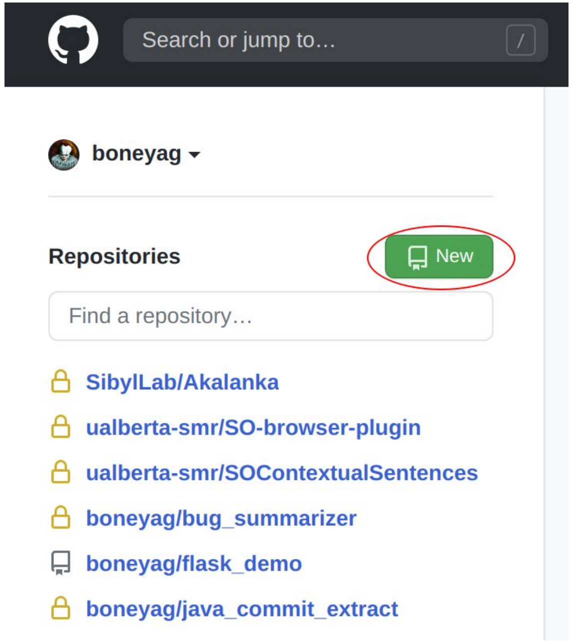
Set up preferred editor

```
$ git config --global core.editor emacs
$ git config --global core.editor vim
$ git config --global core.editor nano
$ git config --global core.editor “code --wait”
```

Create a repo in GitHub

- Visit the URL on your browser.
- Don't have an account -- create one.
- Log into your account.
- Create a repo.

`https://github.com/`


Search or jump to…

boneyag

Repositories

New

Find a repository…

SibylLab/Akalanka

ualberta-smr/SO-browser-plugin

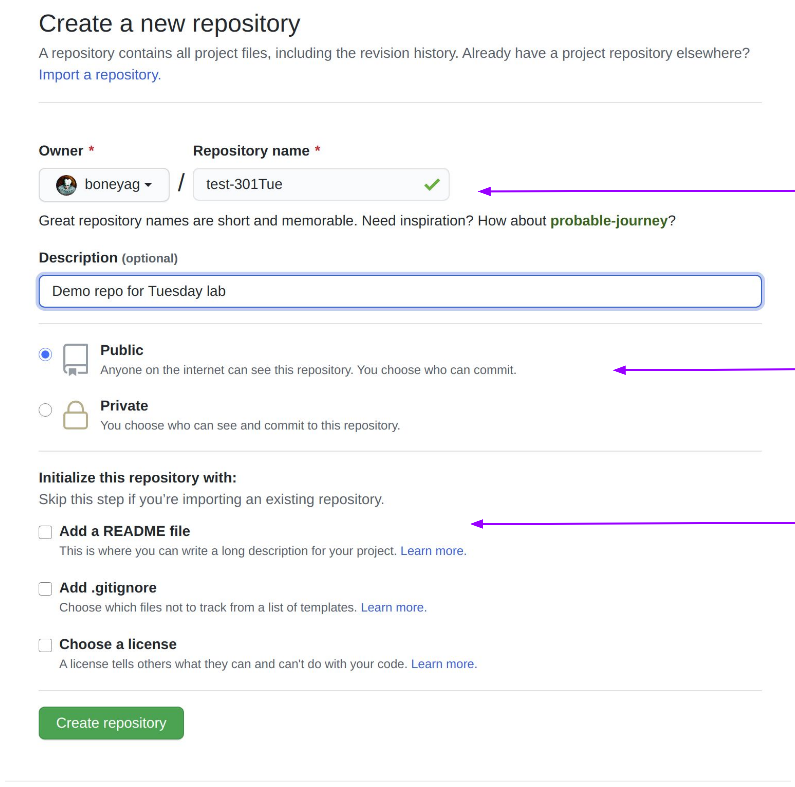
ualberta-smr/SOContextualSentences

boneyag/bug_summarizer

boneyag/flask_demo

boneyag/java_commit_extract

Create a repo in GitHub



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * boneyag / Repository name * test-301Tue ✓

Great repository names are short and memorable. Need inspiration? How about **probable-journey**?

Description (optional): Demo repo for Tuesday lab

[✓ **Public**] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

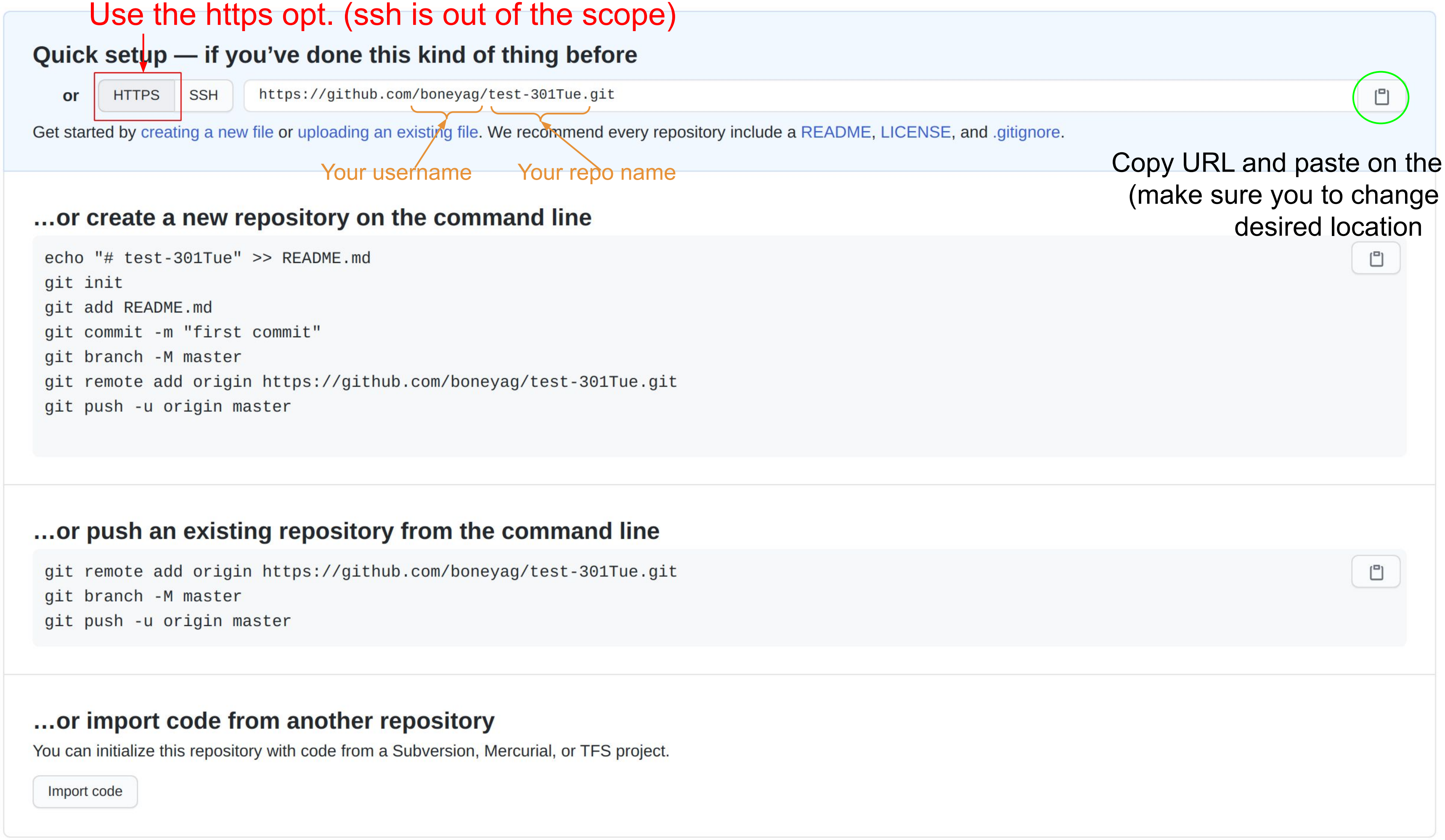
Initialize this repository with:
Skip this step if you're importing an existing repository.

- [ ] **Add a README file** This is where you can write a long description for your project. Learn more.
- [ ] **Add .gitignore** Choose which files not to track from a list of templates. Learn more.
- [ ] **Choose a license** A license tells others what they can and can't do with your code. Learn more.

Short memorable name

Who could see your repo

Automatically create some files


Use the https opt. (ssh is out of the scope)
[1]
[2]
[3]
Your username
Your repo name
Copy URL and paste on the terminal (make sure you to change dir to a desired location

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

or HTTPS SSH https://github.com/boneyag/test-301Tue.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# test-301Tue" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/boneyag/test-301Tue.git
git push -u origin master
```

…or push an existing repository from the command line

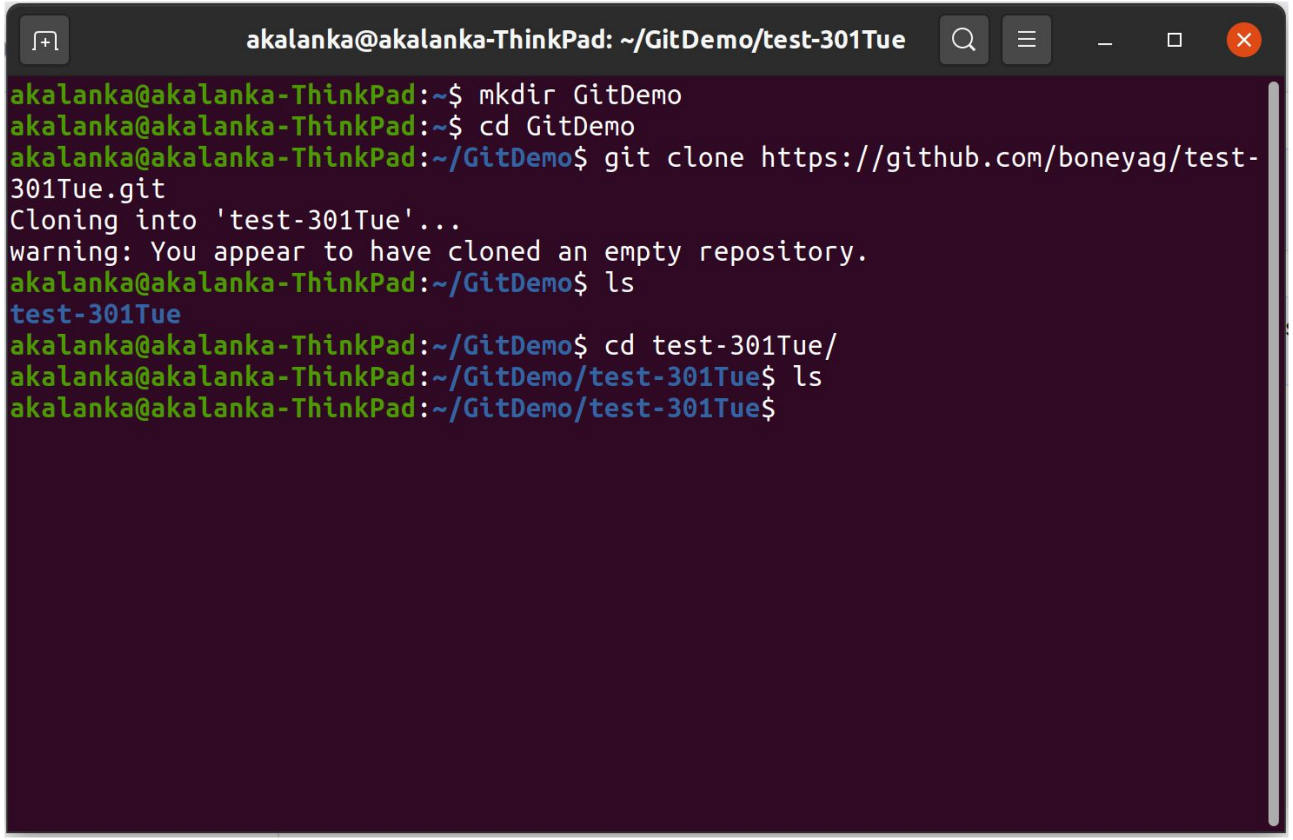
```
git remote add origin https://github.com/boneyag/test-301Tue.git
git branch -M master
git push -u origin master
```

…or import code from another repository

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

Import code

[1] Clone a Git repo



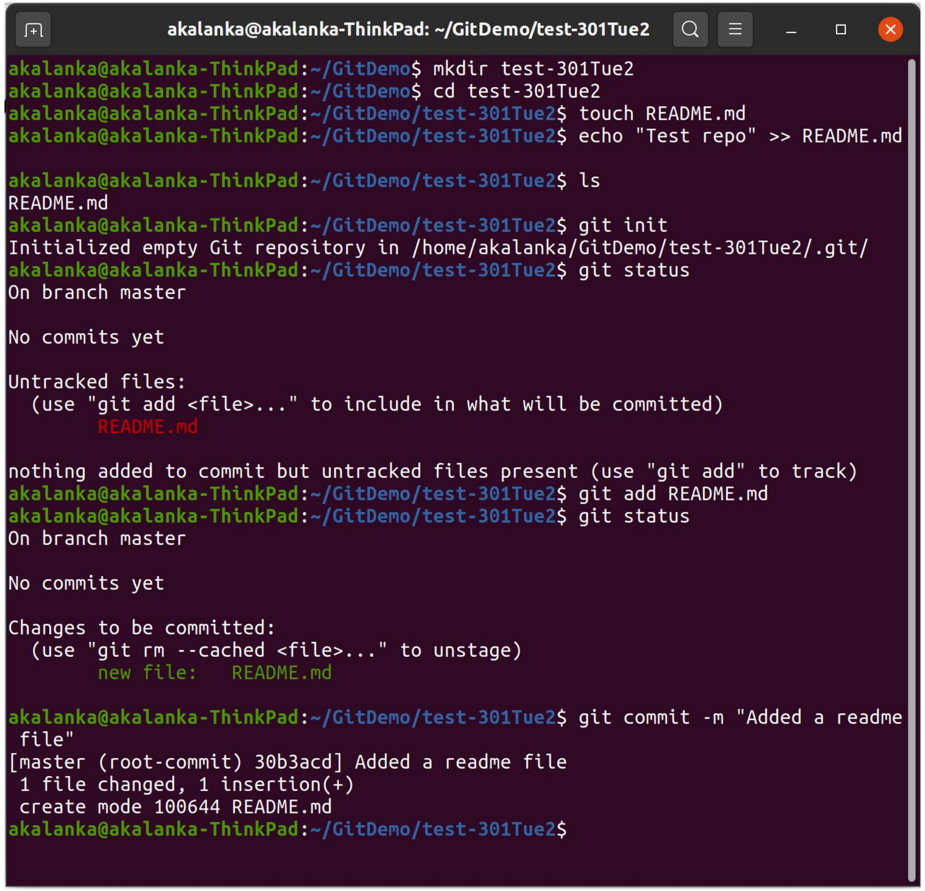
akalanka@akalanka-ThinkPad: ~/GitDemo/test-301Tue

```
akalanka@akalanka-ThinkPad:~$ mkdir GitDemo
akalanka@akalanka-ThinkPad:~$ cd GitDemo
akalanka@akalanka-ThinkPad:~/GitDemo$ git clone https://github.com/boneyag/test-
301Tue.git
Cloning into 'test-301Tue'...
warning: You appear to have cloned an empty repository.
akalanka@akalanka-ThinkPad:~/GitDemo$ ls
test-301Tue
akalanka@akalanka-ThinkPad:~/GitDemo$ cd test-301Tue/
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue$ ls
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue$
```

1. Make a dir
2. Change the dir
3. Use the command `git clone`
4. Copy git URL after
5. Hit return

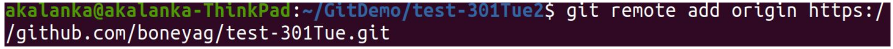
**Use `git clone` when copy the repo to your computer for the first time. After that we use `git pull`.

[2] Create a repo locally

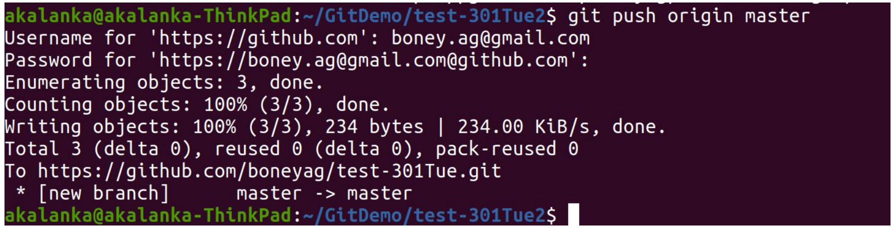


```
akalanka@akalanka-ThinkPad:~/GitDemo$ mkdir test-301Tue2
akalanka@akalanka-ThinkPad:~/GitDemo$ cd test-301Tue2
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ touch README.md
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ echo "Test repo" >> README.md

akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ ls
README.md
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git init
Initialized empty Git repository in /home/akalanka/GitDemo/test-301Tue2/.git/
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git add README.md
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git status
On branch master

No commits yet

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

akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git commit -m "Added a readme
 file"
[master (root-commit) 30b3acd] Added a readme file
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$
```

```
touch README.md
echo “Test repo” >> README.md
git init
git status
git add README.md
git status
git commit -m “<message>”
git remote add origin <URL>
git push origin master
```

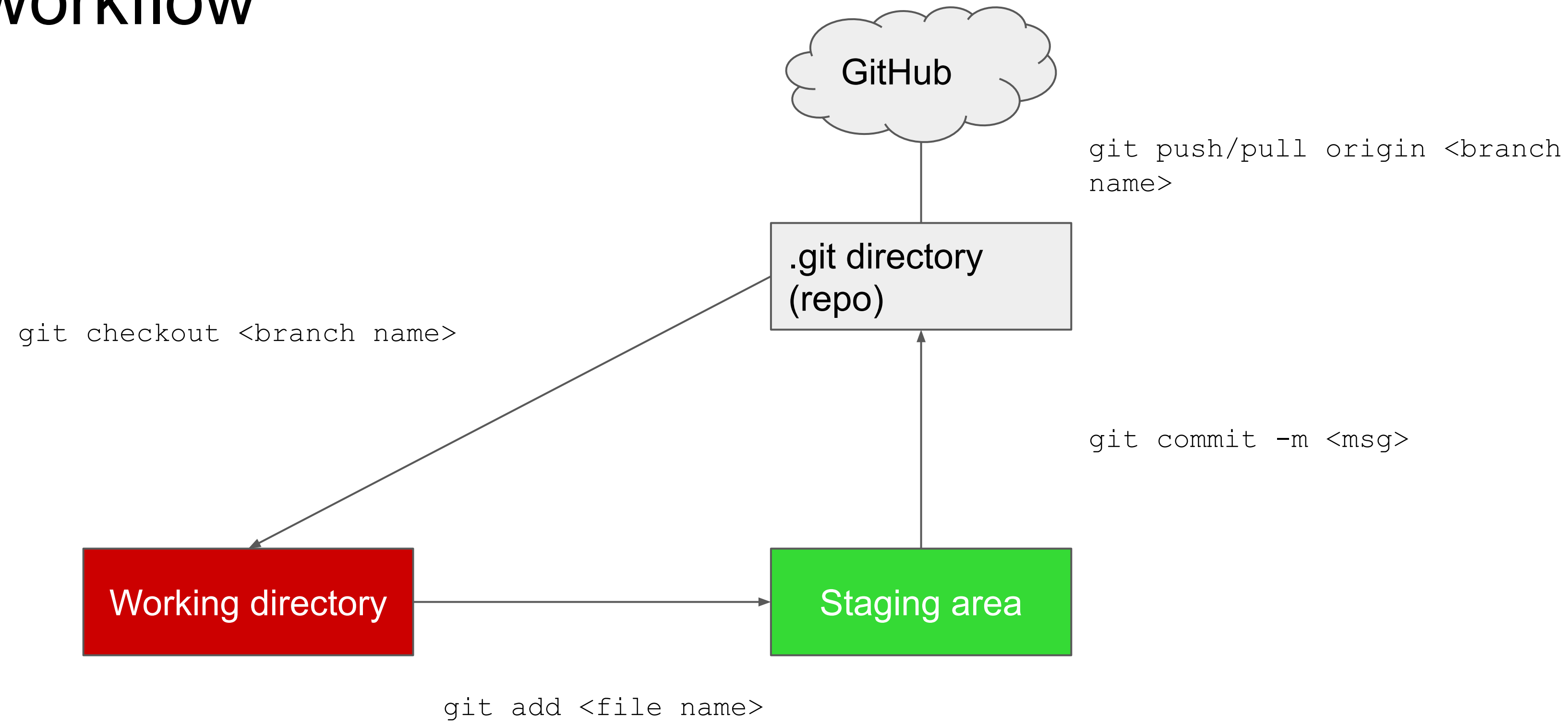


```
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git remote add origin https:/
/github.com/boneyag/test-301Tue.git
```



```
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$ git push origin master
Username for 'https://github.com': boney.ag@gmail.com
Password for 'https://boney.ag@gmail.com@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 234 bytes | 234.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/boneyag/test-301Tue.git
 * [new branch]      master -> master
akalanka@akalanka-ThinkPad:~/GitDemo/test-301Tue2$
```

Git workflow



Practice the common commands

- Create two new text files and push those to Github repo
    - `echo “Test 1” >> test1.txt`
    - `echo “Test 2” >> test2.txt`
    - `git add - A`
    - `git commit -m “<message>”`
    - `git push origin master`
- Useful git commands
    - `git pull origin <branch>`
    - `git reset –` remove files from the staging area
    - `git rm --cached <filename> –` remove a file from working index
    - `git rm -f <filename> –` remove a file forcefully `(–f)`
    - `git rm -rf <dir name> –` remove a directory forcefully `(–r` recursively`)`
    - `git log –` view commit history
    - `git clone <https URL>`

Merge conflicts

Ex: Modifying files in the project.

- TM1: change file1
- TM2: change file2
- Both push changes ✓

- TM1: change file1 -> push
- TM2: change file1, file2 -> try to push ⚠

Let’s get your hands dirty -- demo a merge conflict

Leave the current repo dir (cd ..), open two terminals (pretend those as two users of the repo).
Make sure to follow the order of execution of commands.

	Terminal 1	Terminal 2
1	`mkdir t1`	`mkdir t2`
2	`cd t1`	
3	`git clone <your repo URL>`	`cd t2`
4		`git clone <your repo URL>`
5		`cd test-301Tue`
6		`mkdir one`
7		`echo “Lin1” >> one/file1.txt`

Let’s get your hands dirty -- demo a merge conflict

8		`git status`
9		`git add one/file1.txt`
10		`git commit -m “<msg>”`
11		`git push origin master`
12	`cd test-301Tue`	
13	`git pull origin master`	
14	`mkdir two`	
15	`code two/file2.txt`	
16	`git add two/`	

Let’s get your hands dirty -- demo a merge conflict

```
17  git commit -m “<msg>”
18  git push origin master
19                              cd one
20                              code file1.txt -- add two lines
21                              git add one/file1.txt
22                              git commit -m “<msg>”
23                              git push origin master
24  cd two
25  code file2.txt -- add two lines
```

No conflict as users pushed different files

Let’s get your hands dirty -- demo a merge conflict

```
26   git add two/file2.txt
27   git commit -m “<msg>”
28   git push origin master -- origin updated
29   git pull origin master -- auto merge
30   git push origin master                     cd two
31                                              code file2.txt -- change line 2
32                                              git add two/file2.txt
33                                              git commit -m “<msg>”
34                                              git push origin master
```

Need to pull changes from origin

No conflict as users pushed different files

Let’s get your hands dirty -- demo a merge conflict

Need to pull changes from origin

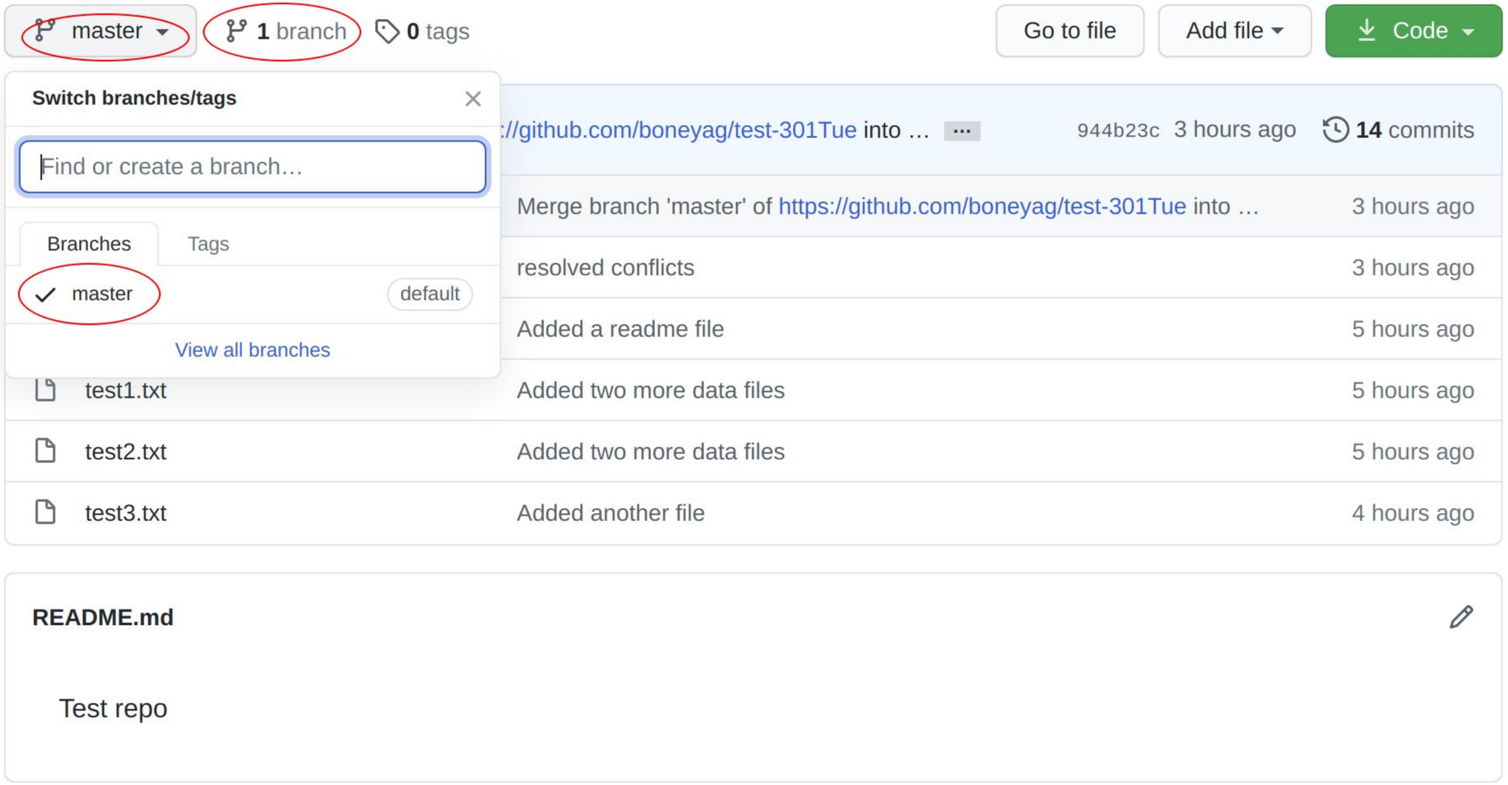
Conflict

```
git push origin master -- origin updated

git pull origin master -- conflict
```

change file as fit

```
git add two/

git commit -m “<msg>”

git push origin master
```

Git branches


master
1 branch
0 tags
Go to file
Add file
Code
Switch branches/tags
Find or create a branch…
Branches
Tags
✓ master
default
View all branches

	://github.com/boneyag/test-301Tue into … 944b23c 3 hours ago	14 commits
	Merge branch 'master' of https://github.com/boneyag/test-301Tue into …	3 hours ago
	resolved conflicts	3 hours ago
	Added a readme file	5 hours ago
test1.txt	Added two more data files	5 hours ago
test2.txt	Added two more data files	5 hours ago
test3.txt	Added another file	4 hours ago

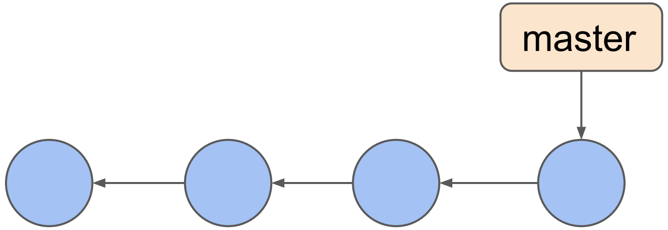
README.md

Test repo

Git branches

What is a branch?

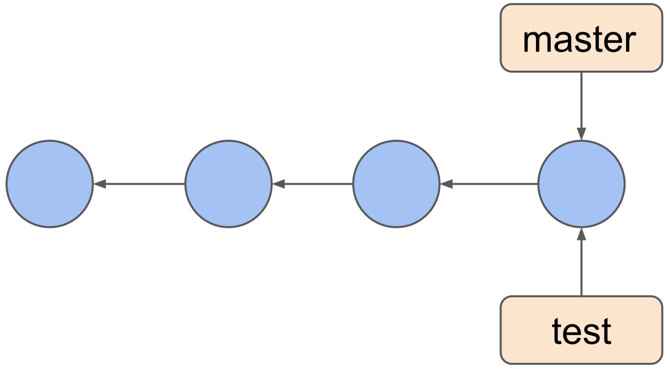
- Branch shows the evolution of your project (commits)
- Each commit has SHA-1 value which allows you to revert back to that state.



Git branches

- Default branch -- master (will change to main in October)
- Nothing special about this branch.
  - You can rename it.
  - Nobody bother to do that, so that it remains with the default name.
- Create a branch in command line

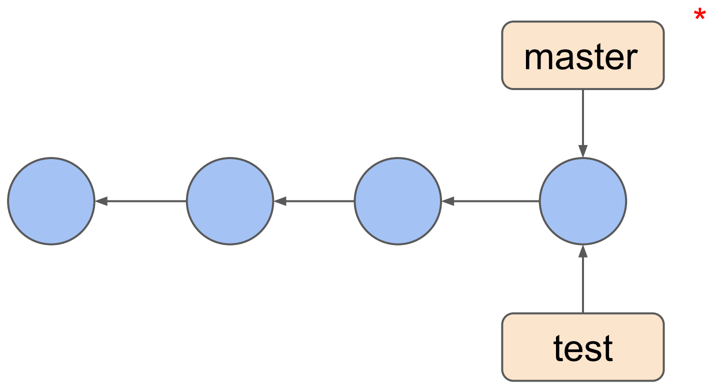
```
git branch test
```



Git branches

- Current working branch
- Change the branch

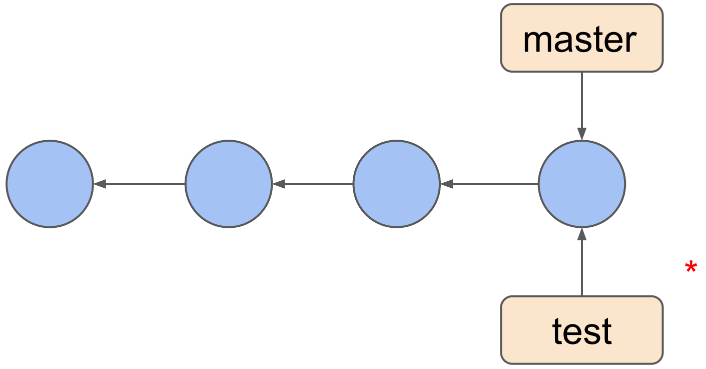
```
git status
```



Git branches

- Current working branch
- Change the branch

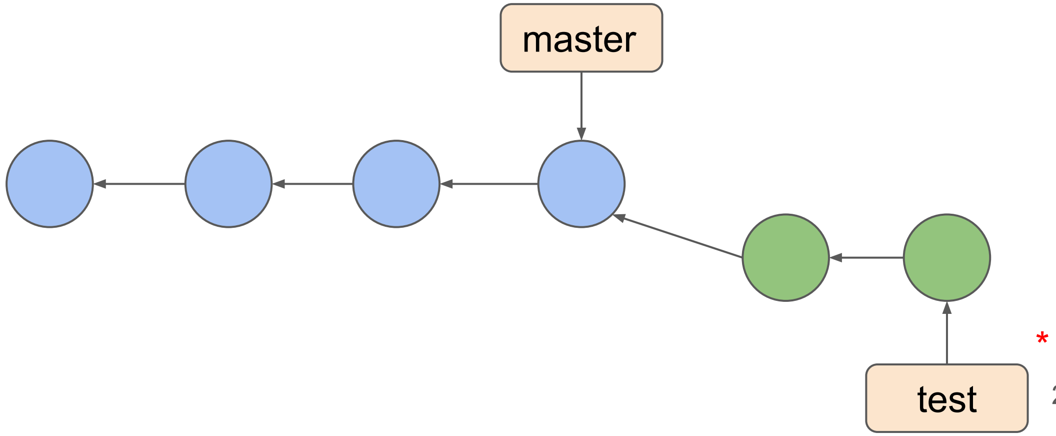
```
git status

git checkout test
```



Git branches

- Do some work on `test` branch

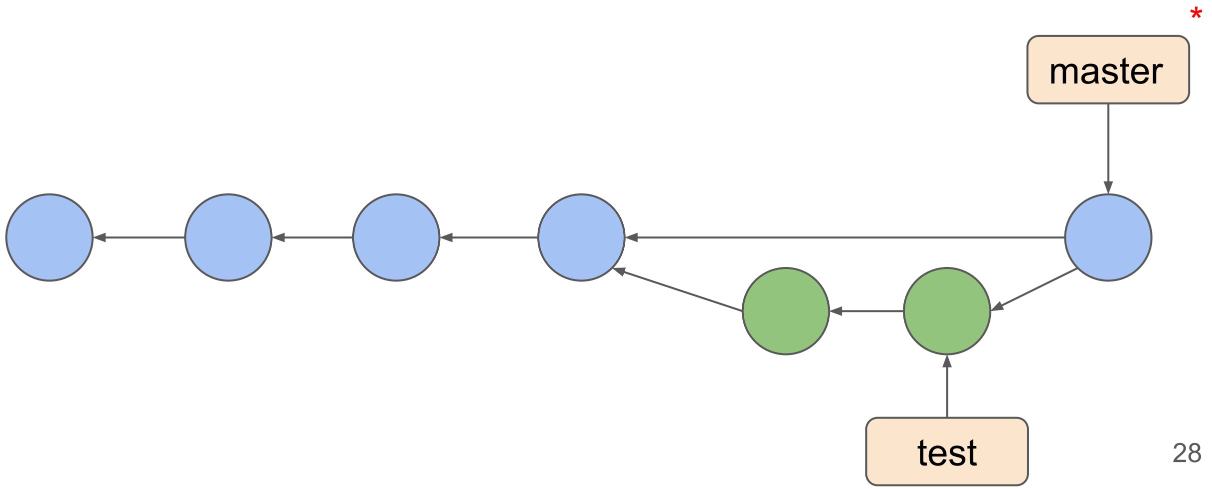
```
git checkout test
code two/file2.txt
git add two/
git commit -m "<msg>"
git push origin test
code one/file2.txt
git add one/
git commit -m "<msg>"
git push origin test
```



Git branches

- Merge test to master

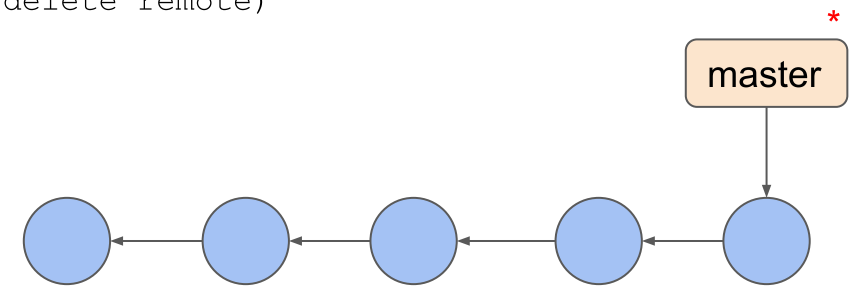
```
git checkout master
git merge test
```



Git branches

- Merge test to master

```
git checkout master
git merge test
git branch -d test (delete local)
git push -d origin test (delete remote)
```



Skip files from adding to the staging area

.gitignore is a special file that contains file patterns that skip when adding files to the staging area

```
https://github.com/github/gitignore
```

```
# IntelliJ
*.iml
.idea/workspace.xml
.idea/tasks.xml
.idea/gradle.xml
.idea/assetWizardSettings.xml
.idea/dictionaries
.idea/libraries
```