

All links can be found on:
github.com/UWB-ACM/git-gud

1. Install Git (command line)

git-scm.com/downloads

2. Create a GitHub account if you don't have one already

github.com/

Before We Begin...

```
$ git gud  
git: 'gud' is not a git command. See 'git --help'.
```

The most similar command is
gui

Git Gud Workshop

By your friends at

UWB ACM

Presenters: Ryan Russell, Lizzy Presland

Agenda

- Introduction to Git
- Commands Overview
- Workshop #1: First Commit
- Workshop #2: Sync Changes
- Best Practices
- Workshop #3: Collaborating with others

The Land Before Git

How have you worked on a project with multiple people in the past?

Did everyone code on their own machines and use a USB to exchange code?

Did you use a Google Doc that everyone edited?

Now imagine a project with 50 people.
That's a lot of USBs.

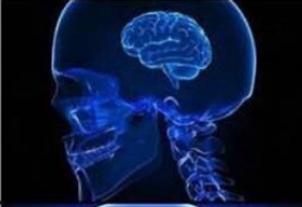
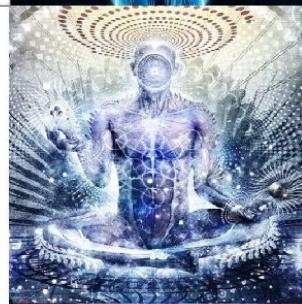


Before Version Control

- ✓ File renaming
Code.001
CodeNov1.xml
- ✓ Directories
\Nov1Code
- ✓ Zip files
Nov1Code.zip
- ✓ Nothing at all



Is this Version Control

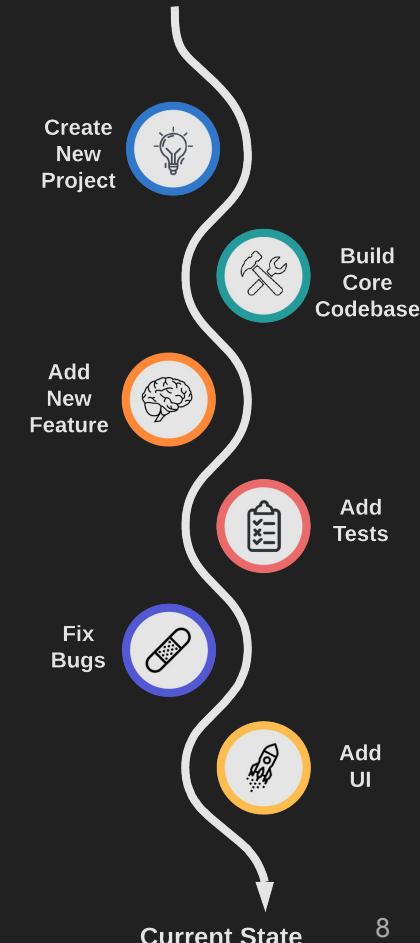
Git		Using a single file on a shared computer	
Google Drive		Placing a video camera recording the keyboard	
Uploading the code to an external hard drive via USB		Keeping the IDE open permanently and using ctrl-z to go back to previous versions	
Mailing the code		Wiping the hard drive and re-writing the program	
Printing the code			

Introduction to Git

What version control is + why it matters

What's Version Control?

- Tracks changes
 - See which user(s) made changes to document(s)
- Save the history of changes to a document
 - Shows the long-term evolution of document(s)
- Implemented in cloud drive platforms
 - e.g. Google Drive

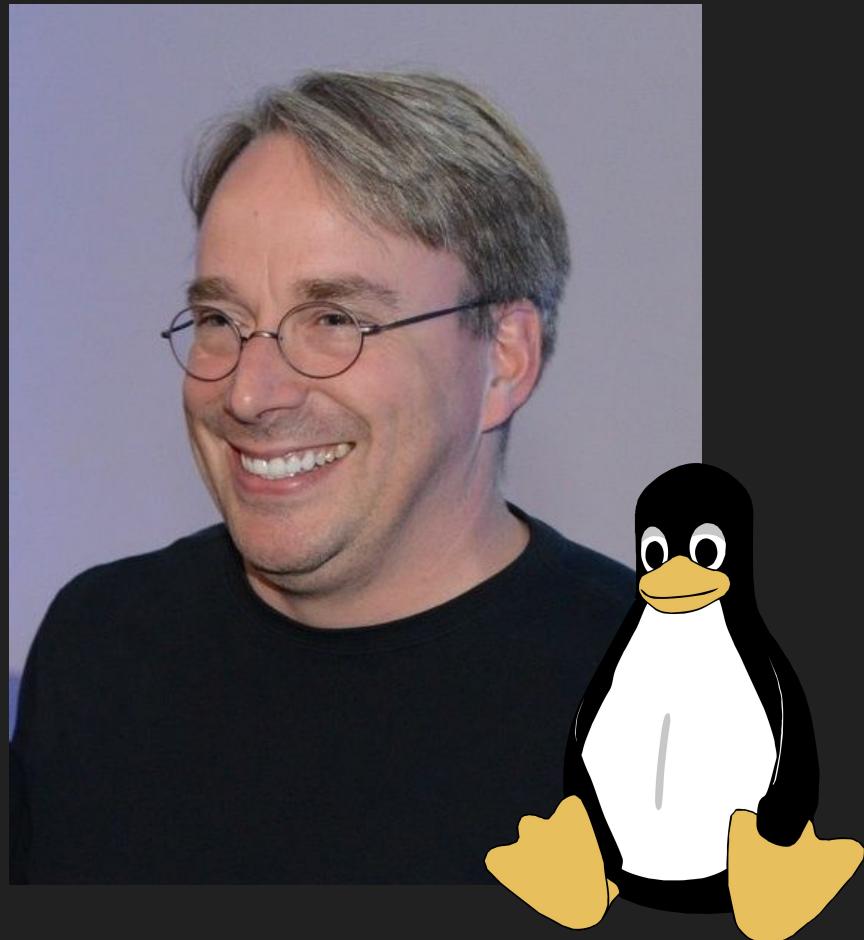


Linus Torvalds

Linus Torvalds, the creator of the Linux kernel, wanted a **new form of version control**.

The project's goal was to synchronize code between fellow programmers and make the development process easy and lightweight.

So in April 2005, his team started development of Git.



What's Git?

Version Control Software!



- Powerful command-line toolbox
- Used for:
 - tracking revisions to files
 - viewing revision history
 - selectively managing revisions by many different people
 - uploading new changes to remote file storage

What *isn't* Git?

Git != GitHub

GitHub is a **remote file storage platform** that **uses** Git

Think about it like:

Git is the interface and GitHub is the storage

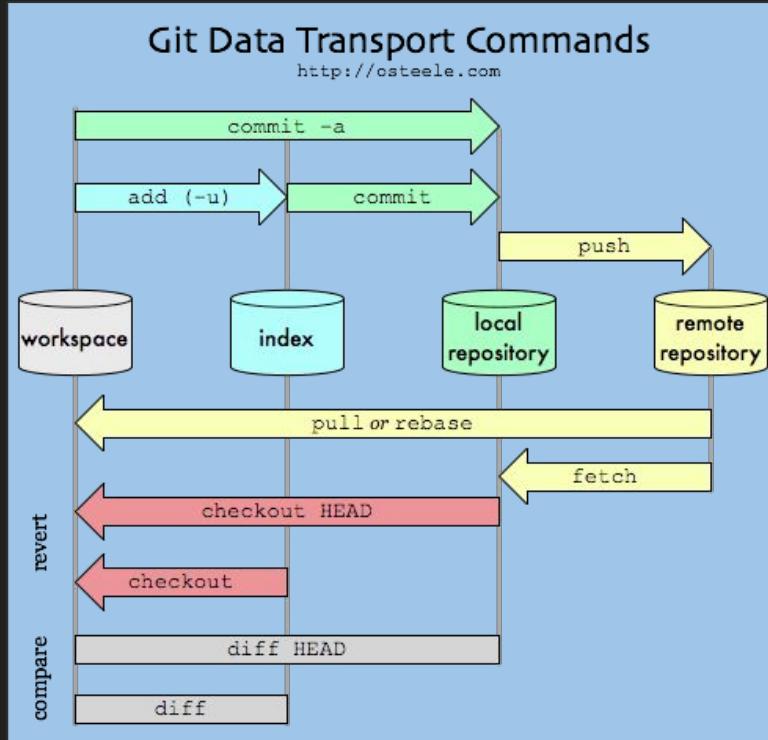
Git != Google Drive

(Git doesn't sync in real-time)



Why Git?

- Keep files in sync!
 - Among workstations
 - Among teammates
- Backup source code
 - Hard drive crashes
- Distributed collaboration
 - Work remote!
- Manage complex workflows
 - Bug fixes, new features, release cycles, and more!



It's great for code!

Who Uses Git?

- Industry leaders in software!
 - o Amazon, Microsoft, Facebook, Netflix, Twitter, Linux, Android, Eclipse, Starbucks, Google, Hashicorp, eBay...
- Open source developers!
 - o Tens of millions of users on GitHub, GitLab, BitBucket, SourceHut, and others
 - o Hosted on GitHub: NodeJS, Kubernetes, React, Bootstrap, OpenCV, and many other major software projects!
- Your classmates!
 - o Some professors (like Prof. Pisan) structure programming assignments in GitHub Classroom
 - o Many other students use it on their own or for group projects

Usage

How to use git

How Local Usage Works - Overview

Step 1 - Code

```
504     if not hasattr(self, '_headers_buffer'):
505         self._headers_buffer = []
506     _headers_buffer.append("%s %d %s\r\n" %
507         (self.protocol_version, code, message)).encode(
508             'latin-1', 'strict')
509
510     def send_headers(self, keyword, value):
511         """Send a HTTP header to the headers buffer."""
512         if self.request_version != 'HTTP/0.9':
513             if not hasattr(self, '_headers_buffer'):
514                 self._headers_buffer = []
515             self._headers_buffer.append(
516                 "%s: %s\r\n" % (keyword, value)).encode('latin-1', 'strict')
517
518         if keyword.lower() == 'connection':
519             if value.lower() == 'close':
520                 self.close_connection = True
521             elif value.lower() == 'keep-alive':
522                 self.close_connection = False
523
```

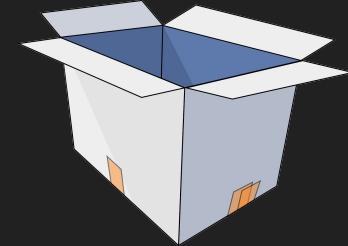
Code on your **local computer**.

Step 2 - Stage



Choose which file changes you want to save.

Step 3 - Commit



Save your changes to your code in a commit.

We'll go into more detail later in this workshop

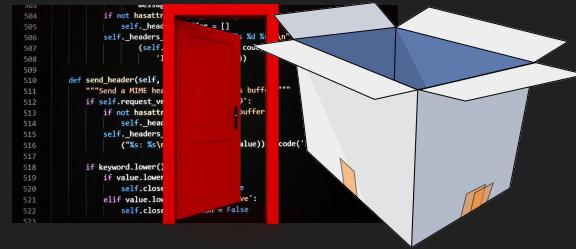
How Remote Syncing Works - Overview

Step 1 - Pull



Download (aka pull)
new changes and
update your local
copy of the repository.

Step 2 - Commit



Make changes and
save your changes
in a commit.

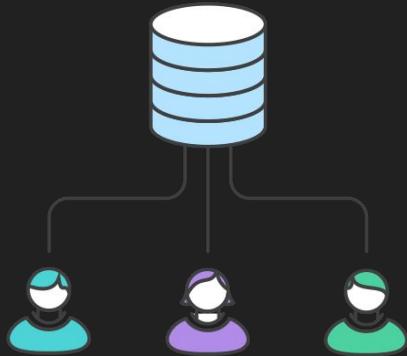
Step 3 - Push



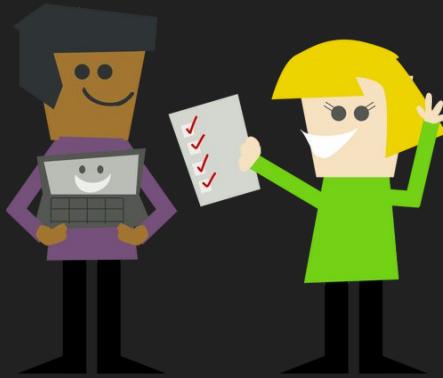
Upload (aka push)
your commit(s) to the
server.
Now others can view
the code with your
changes included.

We'll go into more detail later in this session

How Collaboration Works - Overview



Multiple people can collaborate on the same code easily.



Bugs, feature requests, and large projects can be tracked alongside the code.



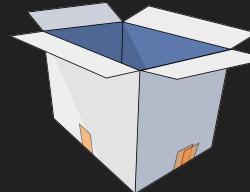
Collaborators can easily review each other's code to find bugs faster and build better software.

We'll go into more detail in the second session, Git Better (2/28/2020)

Workshop #1

Create your first commit

Vocabulary: Repo



GitHub

● Repo / Repository

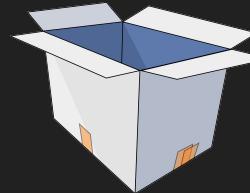
A directory tree which is named .git.

The repository contains a representation of the contents of a directory tree, or *project*, on your computer.

Git tracks changes to files in your project, and stores the revision history of the project in the repository folder.

19

Vocabulary: Commit



GitHub

● Commit

A set of changes applied to files which is permanently saved to the repository history.

A commit is always linked to one or more *previous commits* (called *parents*), creating a chain of revisions.

Commits also have authors and messages associated with them, detailing who created the changes and why.

Commits are uniquely identified using a SHA256 hash.

Vocabulary: More Goodies

- **Status**

The current state of your repository. It has lots of great information. If (or when) you feel lost, run `git status` to see what's up with your project.

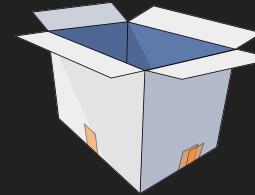
- **Staging**

Staging changes in git ensures those changes will be included in your next commit.

Typically, this is done through the `git add` command.

The staged changes will be in your ***staging area*** until you commit them or remove them from the staging area.

```
584     if not headers_inself._headers_buffer):
585         self._headers_buffer.append(b"\r\n\r\n");
586         self._headers_buffer.append((b" %s %s %s\r\n" %
587             (self._protocol_version, code, message)).encode());
588         self._headers_buffer.append((b"\r\n").encode());
589
590     def send_header(self, keyword, value):
591         """Send a HTTP header to the headers buffer."""
592         if self._method == "HTTP/0.9":
593             if not headers_inself._headers_buffer:
594                 self._headers_buffer = []
595             self._headers_buffer.append((b" %s %s\r\n" %
596                 (code, message)).encode("latin-1", "strict"));
597
598         if keyword.lower() == 'connection':
599             if value.lower() == 'close':
600                 self._close_connection = True;
601             elif value.lower() == 'keep-alive':
602                 self._close_connection = False;
```



Git Config (First-Time Setup)

Git needs to know your username and e-mail to associate your work with you.

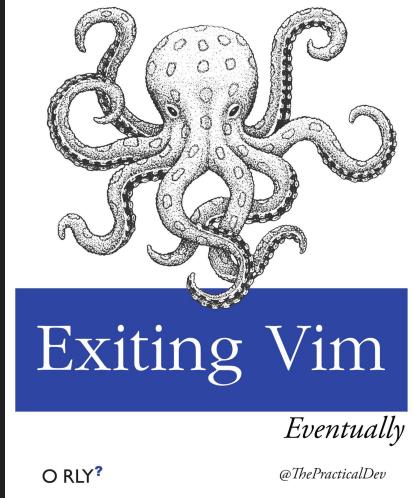
This is the information included in the “author” metadata for commits.

Use the following commands:

```
git config --global user.name "Bobby Tables"
```

```
git config --global user.email bobby@tables.com
```

Just memorize these fourteen contextually dependant instructions



Commands: Creating the Repo

git init : Makes the current directory into a Git repository.

```
:) lizzy@cranberries: ~/code $ mkdir git-test
:) lizzy@cranberries: ~/code $ cd git-test/
:) lizzy@cranberries: ~/code/git-test $ git init
Initialized empty Git repository in /home/lizzy/code/git-test/.git/
:) lizzy@cranberries: ~/code/git-test $ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
:) lizzy@cranberries: ~/code/git-test $
```

Commands: Make changes to repo

git add [files]

Adds the specified files to the staging area.

git commit

Associates a message with the changes in the staging area, and add these changes to the repo's history.

```
lizzy@cranberries:~/code/git-test
File Edit View Search Terminal Tabs Help
lizzy@cranberries:~/code      lizzy@cranberries:~/code/git-test
:) lizzy@cranberries: ~/code/git-test $ echo "adding some more content" >> hello.txt
:) lizzy@cranberries: ~/code/git-test $ git add hello.txt
:) lizzy@cranberries: ~/code/git-test $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

            modified:   hello.txt

:) lizzy@cranberries: ~/code/git-test $ git commit -m "adding more content to hello.txt"
[master 2e84ea3] adding more content to hello.txt
 1 file changed, 1 insertion(+)
:) lizzy@cranberries: ~/code/git-test $
```

Workshop 1: Purpose and Outcome

Purpose:

Create a local repository, and commit a file to it.

Outcome:

A local repository with the created file in it.

Workshop #1: Our First Commit

Step	Task	Command
1	Create a local repo	<code>mkdir git-gud && cd git-gud git init</code>
2	Create new file(s)	<code>echo "Hi!" >> hello.txt</code>
3	Check repository status	<code>git status</code>
4	Add files to our index	<code>git add .</code>
5	Commit changes to local repo	<code>git commit</code>

Workshop #1: Example

1. Initialize repo
2. Create text file
3. Add file to index (staging area)
4. Commit file to repo

```
:~/ACM/test$ git init
pacekatt/ACM/test/.git/
:~/ACM/test$ echo "Hello, World!" >> hello.txt
:~/ACM/test$ git add hello.txt
:~/ACM/test$ git commit -m "Add hello world doc"
world doc

:~/ACM/test$ █
```

Workshop #2

Commit & Push to Remote Repository

Vocabulary

- **Remote**

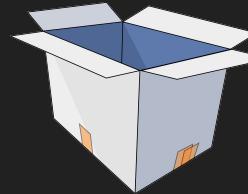
The repository's remote points to a remote storage endpoint which contains a copy of the current repository.

The remote has a name; a common convention is to call the remote `origin`.

- **Clone**

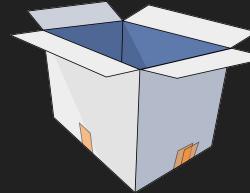
Create a local copy of an existing remote repository. Cloning a repo retrieves the full filesystem and commit history for that repository.

```
584     if not message in self._headers_buffer:
585         self._headers_buffer.append((b"Content-Type", b"application/x-www-form-urlencoded"))
586         self._headers_buffer.append((b"Content-Length", str(len(message)).encode("latin-1", "strict")))
587
588     def send_header(self, keyword, value):
589         """Send a HTTP header to the headers buffer."""
590         if self.request_version == "HTTP/0.9":
591             self._headers_buffer.append((keyword, value))
592         else:
593             self._headers_buffer.append((b": ".join([b"Content-Type", b"Content-Length"])))
594
595             if keyword.lower() == "connection":
596                 if value.lower() == "close":
597                     self.close_connection = True
598                 elif value.lower() == "keep-alive":
599                     self.close_connection = False
600
601             self._headers_buffer.append((b"\r\n".join([b"Content-Type", b"Content-Length"])))
602
603             if keyword.lower() == "content-type" or keyword.lower() == "content-length":
```



GitHub

Vocabulary



GitHub

- Push

Sync local changes to the repository by uploading local commits to remote storage (GitHub, GitLab, BitBucket, SourceHut, private servers, etc.).

- Pull

Retrieve changes from repository by downloading commits from remote storage.

Note: you can only run push & pull when you have a remote identified in your repository.

Commands: Syncing with remote repo

git remote -v : get information about remote repositories associated with the local repository

git remote add <nickname> <URL> : associates a local repository with a remotely hosted repository (GitHub, GitLab, etc)

git push <nickname> <branch> : Uploads changes.

git pull <nickname> <branch> : Downloads changes.

```
:) lizzy@cranberries: ~/code/git-test $ git remote -v
:) lizzy@cranberries: ~/code/git-test $ git remote add origin https://github.com/etcadinfinity/musical-carnival.git
:) lizzy@cranberries: ~/code/git-test $ git remote -v
origin https://github.com/etcadinfinity/musical-carnival.git (fetch)
origin https://github.com/etcadinfinity/musical-carnival.git (push)
:) lizzy@cranberries: ~/code/git-test $ git push origin master
Username for 'https://github.com': etcadinfinity
Password for 'https://etcadinfinity@github.com':
Enumerating objects: 6, done.
```

In case of fire

1. git commit

2. git push

3. leave the building



Workshop #2: Syncing to remote (GitHub)

Step	Task	Command
1	Follow instructions in open issue, then clone this repo to your machine	<code>git clone https://github.com/UWB-ACM/Git-Gud-Spring2020</code>
2	Create and commit a new file named <github-username>.txt	<code>cd Git-Gud-Spring2020 touch <username>.txt git add <username>.txt && git commit</code>
3	Sync local changes to remote	<code>git push origin master</code>

Uh Oh!

Help! I got a scary error message!

```
:) lizzy@cranberries: ~/code/git-gud $ git remote -v
origin  https://github.com/UWB-ACM/git-gud (fetch)
origin  https://github.com/UWB-ACM/git-gud (push)
:) lizzy@cranberries: ~/code/git-gud $ git log
commit e0523d7bf469aeaf7adaba2fcfa33b6f065aeab78b (HEAD -> master)
Author: etcadinfinity <lizzy.presland@gmail.com>
Date:   Tue Feb 18 17:20:22 2020 -0800

    Add hello.txt
:) lizzy@cranberries: ~/code/git-gud $ git push origin master
Username for 'https://github.com': etcadinfinity
Password for 'https://etcadinfinity@github.com':
To https://github.com/UWB-ACM/git-gud
 ! [rejected]          master -> master (fetch first)
error: failed to push some refs to 'https://github.com/UWB-ACM/git-gud'
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.
:( lizzy@cranberries: ~/code/git-gud $
```

You may have accidentally **created a commit in the remote repository**. (README, license, .gitignore...)

The **remote repo** has commits that your **local repo** doesn't have -- they are out of sync!

Don't worry -- this is easy to fix!

Uh Oh! [RESOLVED]

We can fix this very easily!

We need to **update our local repository** with the changes we don't have. To do this, run:

```
git pull --rebase origin master
```

```
:) lizzy@cranberries: ~/code/git-gud $ git pull --rebase origin master
warning: no common commits
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
From https://github.com/UWB-ACM/git-gud
 * branch            master      -> FETCH_HEAD
 * [new branch]      master      -> origin/master
First, rewinding head to replay your work on top of it...
Applying: Add hello.txt
```

Then, try pushing your changes again.

Note: More on this in Workshop 3

Refresh Repo Page on GitHub

We now see the file we pushed to our remote, origin



Exciting!

Much open source!

wowee

SpaceKatt / **glowing-guacamole**

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Unwatch 1 Star 0 Fork 1

Sample repository for the GitGud workshop (16/Nov/2018) Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Tree: 93e019ae16 ▾ New pull request Create new file Upload files Find file Clone or download ▾

SpaceKatt Add hello world doc Latest commit 93e019a 2 days ago

hello.txt Add hello world doc 2 days ago

Heckin' origin

Best Practices

Git Gud, Git Gr8, m8

Writing Good Commit Messages

Qualities of a good commit message:

- Use the imperative mood
 - “Add config file”
 - “Fix nginx SSL bug”
- Subject is clear and descriptive, while under 50 characters in length.
- Body adds detail and explanations
- **Avoid using `git commit -m`**
(do as I say, not as I do)

COMMENT	DATE
O CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O ENABLED CONFIG FILE PARSING	9 HOURS AGO
O MISC BUGFIXES	5 HOURS AGO
O CODE ADDITIONS/EDITS	4 HOURS AGO
O MORE CODE	4 HOURS AGO
O HERE HAVE CODE	4 HOURS AGO
O AAAAAAAA	3 HOURS AGO
O ADKFJSLKDFJSOKLFJ	3 HOURS AGO
O MY HANDS ARE TYPING WORDS	2 HOURS AGO
O HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

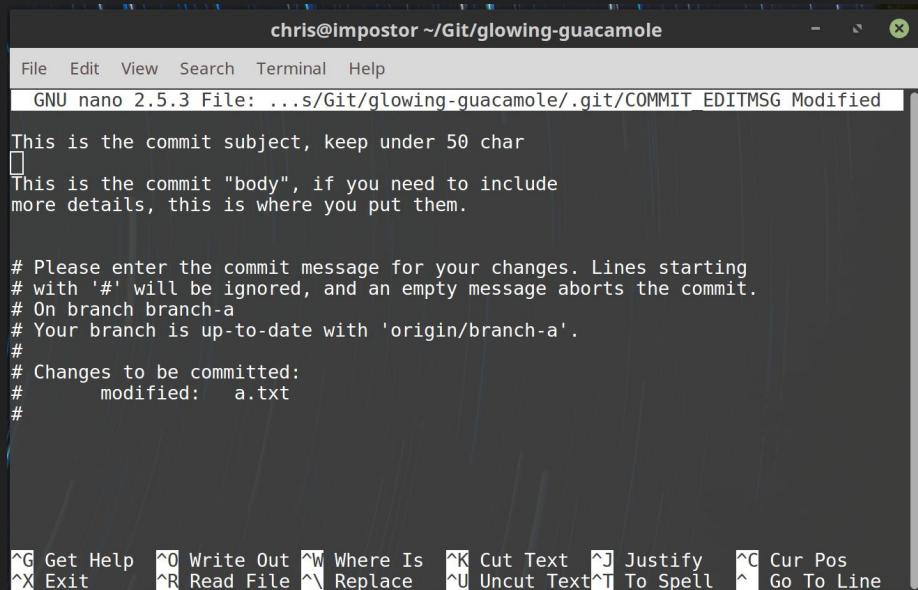
Further reading:

<https://chris.beams.io/posts/git-commit/>

<https://help.github.com/articles/closing-issues-using-keywords/>

Commit Message Example

What you will see:



A screenshot of a terminal window titled "chris@impostor ~/Git/glowing-guacamole". The window shows a nano text editor with the following content:

```
This is the commit subject, keep under 50 char
This is the commit "body", if you need to include
more details, this is where you put them.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch branch-a
# Your branch is up-to-date with 'origin/branch-a'.
#
# Changes to be committed:
#       modified:   a.txt
#
```

The bottom of the terminal shows standard nano key bindings:

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

How others will see your message in GitHub:



This is the commit subject, keep under 50 char

Chris-Johnston committed 7 minutes ago

This is the commit "body", if you need to include more details, this is where you put them.

Add branch a files'

Chris-Johnston committed 39 minutes ago

get outta here, foo!

Chris-Johnston committed an hour ago

Viewing the log

View previous commit messages with...

git log

Benefits of a well-kept log...

- Workflow transparency
- Process traceability
- Bug tracking
- Smaller commits
- Easier merging

```
commit 3199695f77bdb35fd9d510208b8ee95f04e7f3a3
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 18:56:23 2018 -0800

    Update HTML style

commit 08fa02f8155d73dedbec152e05e3640b5632e408
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 16:49:47 2018 -0800

    Fix bug where service starts before loading profile

    Since AWS credentials are provisioned at deployment, I believe
    the gunicorn service was being started by systemd before the
    credential files existed, causing an error (sometimes). Restarting
    the service after Terraform provisions everything seems to fix it.

commit b0299d535f0b1013c399a9ecaee75e7f21d722ef
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 13:08:49 2018 -0800

    Add alert message to user for empty input

commit ece5c4048e93913ef26d5cf0a324124190a6297f
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 12:56:30 2018 -0800

    Handle ommisions in query field

commit 29011a7ab3923bfa5d07ddd1bcf8dcf77ecfd172
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 12:34:01 2018 -0800

    Add all attributes to table

commit ac282eaac3871148bb52e7b8bd87dc2f001254fa
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 11:45:08 2018 -0800

    Add simple HTML table in React
```

Workshop #3

Handling Merge Conflicts

What is a merge conflict?

Imagine 2 people are working on the same file.

Each person has their own copy of the file, and they make changes.

What happens when the **two people change the same line of code**?

Who's code do we decide to keep?

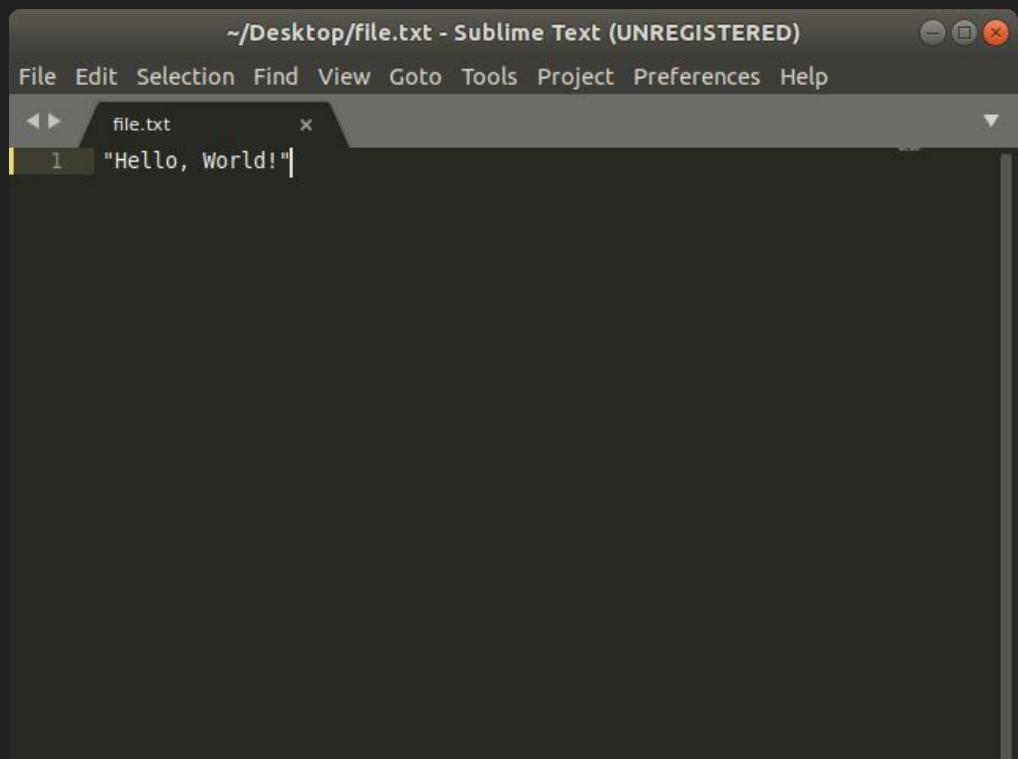
Merge conflicts happen when **multiple people edit the same line in the same file**.

Merge Conflicts: Example

Let's say we have a file called
file.txt

Johnny changes **line 1** to say “Hello,
World!”

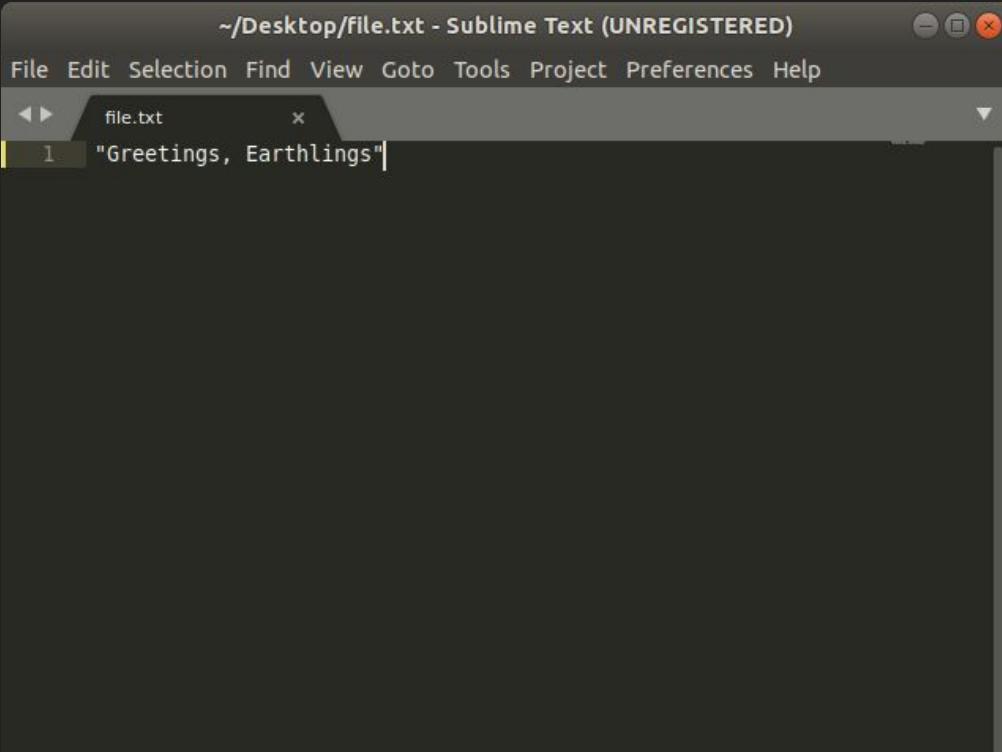
He then pushes the changes to the
repository.



The screenshot shows a dark-themed Sublime Text window titled "~/Desktop/file.txt - Sublime Text (UNREGISTERED)". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. A tab bar at the top shows "file.txt". The main editor area contains one line of code: "1 \"Hello, World!\"". The status bar at the bottom indicates "Line 1, Column 16" and "Plain Text".

Merge Conflicts: Example

Without first pulling the file, Sally changes line 1 to say “Greetings, Earthlings”



~/Desktop/file.txt - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

file.txt

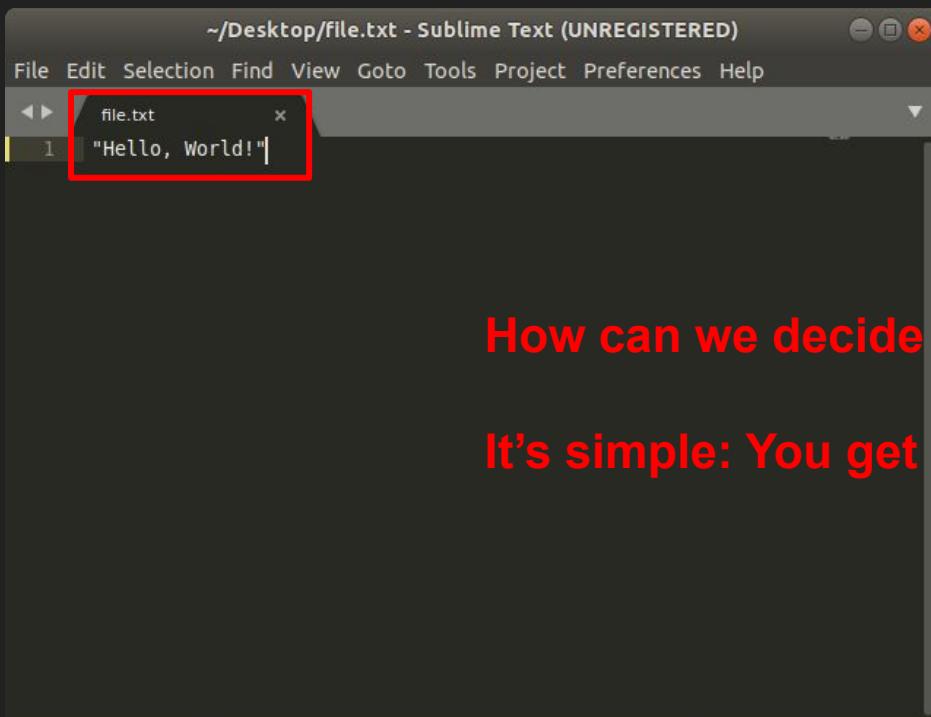
1 "Greetings, Earthlings"

Line 1, Column 24

Tab Size: 4 Plain Text

Merge Conflicts: Example

file.txt in the Remote Repo



~/Desktop/file.txt - Sublime Text (UNREGISTERED)

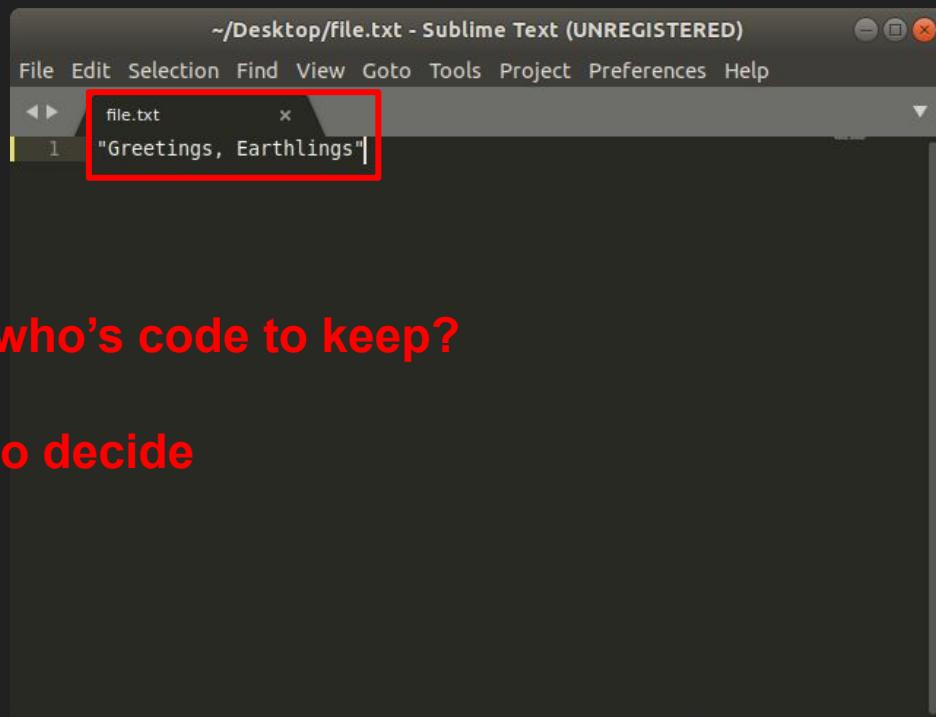
File Edit Selection Find View Goto Tools Project Preferences Help

file.txt x

1 "Hello, World!"

This screenshot shows a Sublime Text window titled 'file.txt' in the 'Remote Repo'. The content of the file is 'Hello, World!'. A red box highlights the entire content area.

file.txt in Sally's Repo



~/Desktop/file.txt - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

file.txt x

1 "Greetings, Earthlings!"

This screenshot shows a Sublime Text window titled 'file.txt' in 'Sally's Repo'. The content of the file is 'Greetings, Earthlings!'. A red box highlights the entire content area.

How can we decide who's code to keep?

It's simple: You get to decide

Workshop 3: Merge Conflict Exercise

In the cloned repo for <https://github.com/UWB-ACM/Git-Gud-Spring2020>,
add a new line at the end of the README.md file.

It can say anything you like, but we recommend adding a joke ;)

You will see a message like this when you do `git push`

```
ryan@ryan-Inspiron-7573:~/Coding/Testing$ git push
To github.com:RyanRussell00/Testing.git
 ! [rejected]      master -> master (fetch first)
error: failed to push some refs to 'git@github.com:RyanRussell00/Testing.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.
ryan@ryan-Inspiron-7573:~/Coding/Testing$
```

Run `git pull` and read the output

There should be a message like this:

```
First, rewinding head to replay your work on top of it...
Applying: Merge Conflict Example 1
Using index info to reconstruct a base tree...
Falling back to patching base and 3-way merge...
Auto-merging file.txt
CONFLICT (add/add): Merge conflict in file.txt
error: Failed to merge in the changes.
Patch failed at 0001 Merge Conflict Example 1
Use 'git am --show-current-patch' to see the failed patch
```

Conflicted File

Line Number of Conflict

Reading the Merge Conflict

This is what your file looks like when you have a merge conflict:

A screenshot of a Sublime Text window titled '~Coding/Testing/file.txt - Sublime Text (UNREGISTERED)'. The window shows two tabs: 'file.txt — Desktop' and 'file.txt — Coding/Testing'. The content of the 'Coding/Testing' tab is as follows:

```
1 <<<<< HEAD
2 "Hello, World!"
3 =====
4 "Greetings, Earthlings"
5 >>>>> Merge Conflict Example 1
6
```

Start of incoming changes

End of incoming changes
AND
Start of current changes

End of current changes

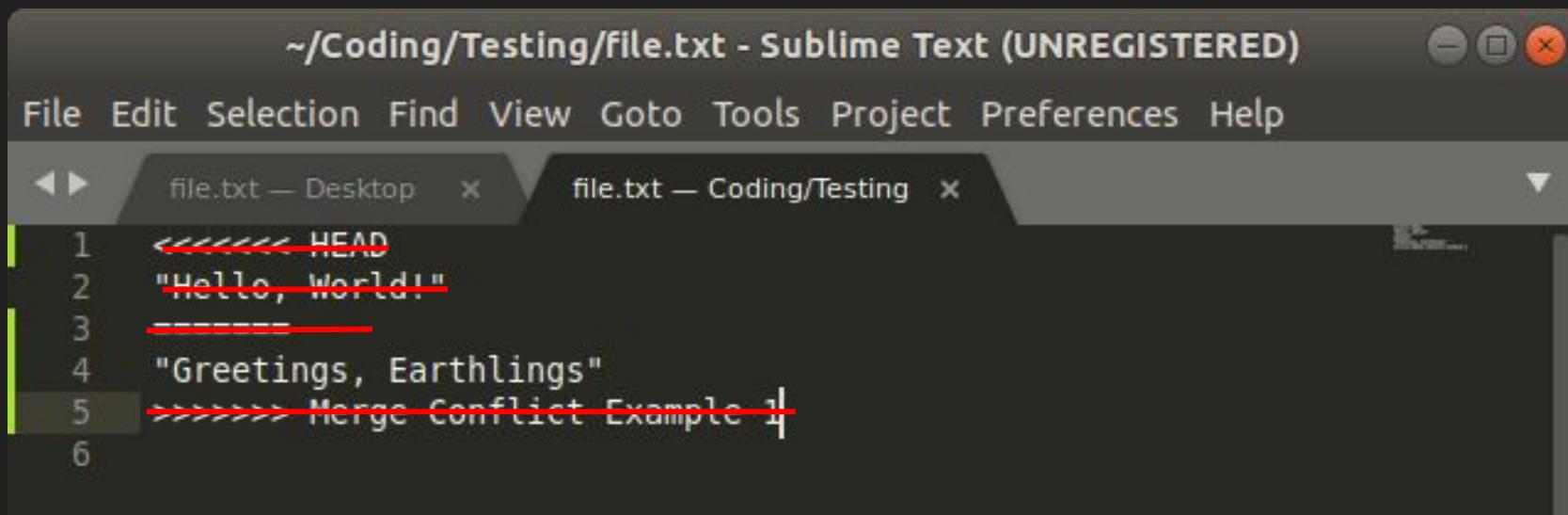
Commit message of the commit causing the conflict

Solving the Merge Conflict

Let's say we want our **current** changes to **override** the **remote** changes.

Quick & Easy Solution: Delete the code that you don't want!

Make sure to delete the <<HEAD, ==, >>, and Commit Message



The screenshot shows a Sublime Text window with two tabs: 'file.txt — Desktop' and 'file.txt — Coding/Testing'. The 'Coding/Testing' tab is active, displaying the following content:

```
1 <---- HEAD
2 "Hello, World!"
3 -----
4 "Greetings, Earthlings"
5 >>> Merge Conflict Example 1
6
```

The first two lines ('1 <---- HEAD' and '2 "Hello, World!"') are crossed out with red lines. The entire line '5 >>> Merge Conflict Example 1' is also crossed out with a red line. The number '6' at the bottom is not crossed out.

Finish up!

```
git push origin master
```

Note: you may have to resolve a conflict many times, because each time one of you pushes your changes successfully, it could create a conflict with someone who hasn't pushed yet!

Lessons Learned

- Why git is important
- How to make a commit
- How to push local repository to remote (GitHub)
- How to manage files with git
- How to contribute to open source projects
- **Git is fun!**



The ACM is a **Global Organization**, with nearly **100,000 members** from more than **190 countries**, ACM works to **advance computing** as a science and a profession.

We Are Hiring New Officers!

Go to **bit.ly/HireMeACM** to apply!

Questions?
Comments?
Concerns?

Join us next week for advanced
topics on git usage!
Slides available at bit.ly/ACMGitGud

Day 2: Our mission: build the Enterprise Calculator v1.0

We are the development team for **UWB ACM Incorporated**. Our company has just bought a startup: FancyCal. Our management has asked this team to use this as a starting point to create our new flagship product: The Enterprise Calculator.

