Git Bootcamp

Computer Student Society

Nate Dolny

What is Git?

- One of the many popular version control systems available

- Git is a tool used to manage code but can also be applied to any other type of document (**does not have to be code**)

Version Control

Records changes a person made to a document overtime. It can be used by one person or multiple people.

History of Git

Started over a license dispute between Linus Torvalds and BitKeeper. Basically BitKeeper, the source code management system for the Linux kernel and many other open source projects, revoked the Linux developer's free license. Bitkeeper's parent company did this because an open source developer, attempted to create an open-source version of the BitKeeper client without accepting its proprietary license.

BITKEEPER
Scalable Version Control

History of Git

More about Git:

- Git was built in about 5 days
- Released in April 2005 by Linus Torvalds

Design goals of Git:

- Speed
- Simple design
- Fully distributed
- Strong support for non linear development (branching)
- Able to handle large projects such as the linux kernel

Why you should use Git?

Organization:

- Tracks changes on a document
- Allows for multiple users to work on the same document

Branching:

- Allows for multiple branches to be created (more later)

Backups:

- Acts as a backup when used remotely (**should not be your main backup**)

Machine Agnostic:

- Can be used on any operating system: Windows, Mac, GNU Linux

When should you use Git?

- **ALWAYS!!**
- Git is your best friend when it comes to any project big or small

LEVEL

ONE

Level One: Setting up Git

Create a Profile

Set Profile Name

- git config –global user.name "Peter Parker"

Set Profile Email

- git config –global user.name "Parker.P@midtown.net"

Set Default Editor

- git config –global editor="vim"

View our configuration (Optional)

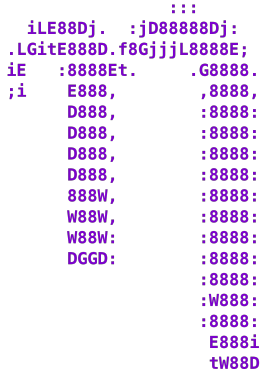
- git config –list

PRO TIP

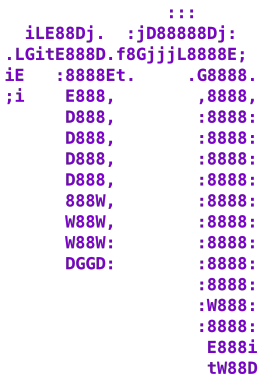
Git allows for multiple profiles, so you could have a personal and school profile.

Level One: Command-line Text Editors

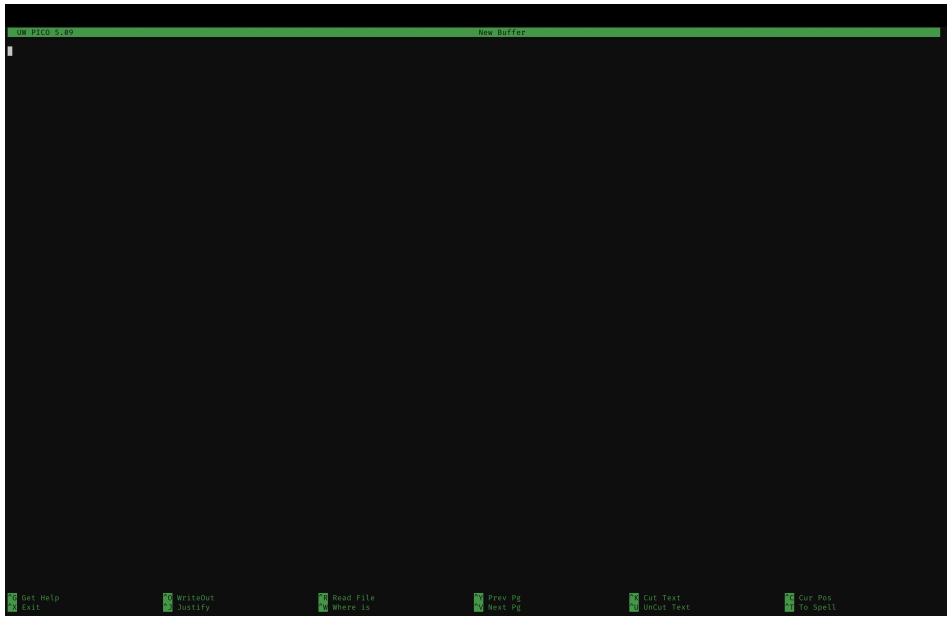
- A text editor that can run directly in a terminal without a graphical user interface



Level One: GNU Nano Text Editor



- Released in November 18, 1999
- User-friendly and easy to learn
- Developed and maintained by volunteers



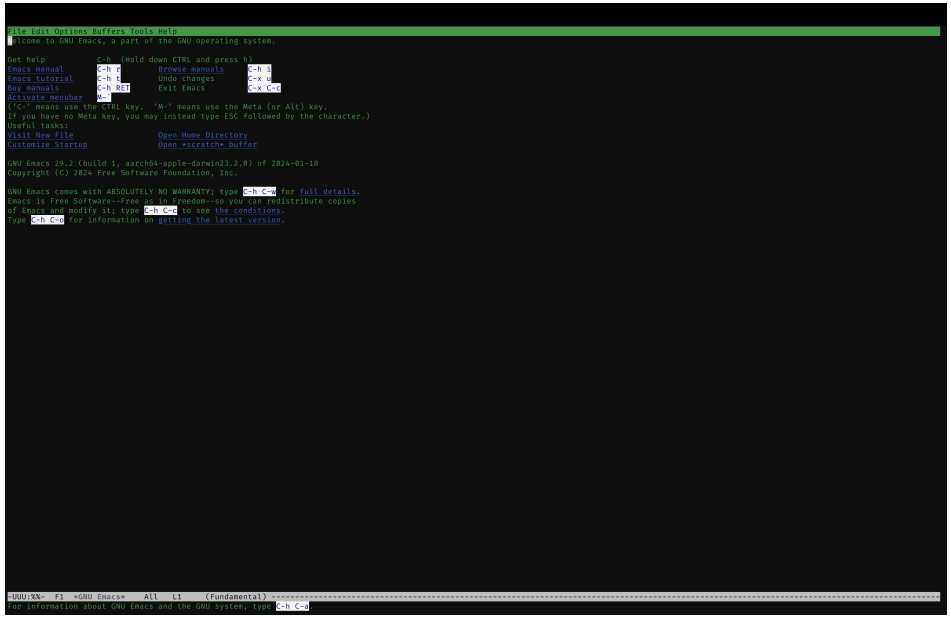
Level One: Vim Text Editor

- Released in November 2, 1991
- Modal editor
- Highly customizable, many plugins available

Level One: Emacs Text Editor

- Released in March 20, 1985
- Created by GNU Project founder Richard Stallman
- Highly customizable

- Many modes for different purposes such as browsing the web



Level One: What makes a good commit?

- Detailed messages that someone can read that informs others about the changes you made to a document.

Changes 4000 lines of code

```
> git commit -m "minor changes"
> git push origin main
```

Leaves

Level One: What makes a good commit?

A better approach to making commits

- *Subject:* Add a subject title
- *Changes:* Add details about changes made to file(s)
- *State:* Add a brief summary of the state of program or feature
- *TODO:* Make notes about things to fix

PRO TIP

Pretend you are sending an email to your friends, group mates, or professor informing them of the changes you made to the document.

Level One: Making a commit

Step 1. *Initializes a new repository in the current directory*

- git init

Step 2. *Add a document to staging*

- git add $\prec$ *document* $\succ$

Step 3. *View documents that have been staged (Optional)*

- git status

Step 4. *Remove a document from staging (Optional)*

- git reset $\prec$ *document* $\succ$

Step 5. *Record changes to repository*

- git commit

Step 6. *Upload changes to remote repository*

- git push

PLAY

TIME

15 Minutes

Play Time: Can you do this?

Task 1:

- Setup a second git profile include your name, email, editor of choice

Task 2:

- Create a new directory name it something you like
- Make a commit message that says: first commit every

15 Minutes

BREAK

TIME

15 Minutes

LEVEL

TWO

Level Two: Remote Version Control Platforms

- Allows you to store code some place other than your local machine
- Allows for multiple people to collaborate on a project

GitHub

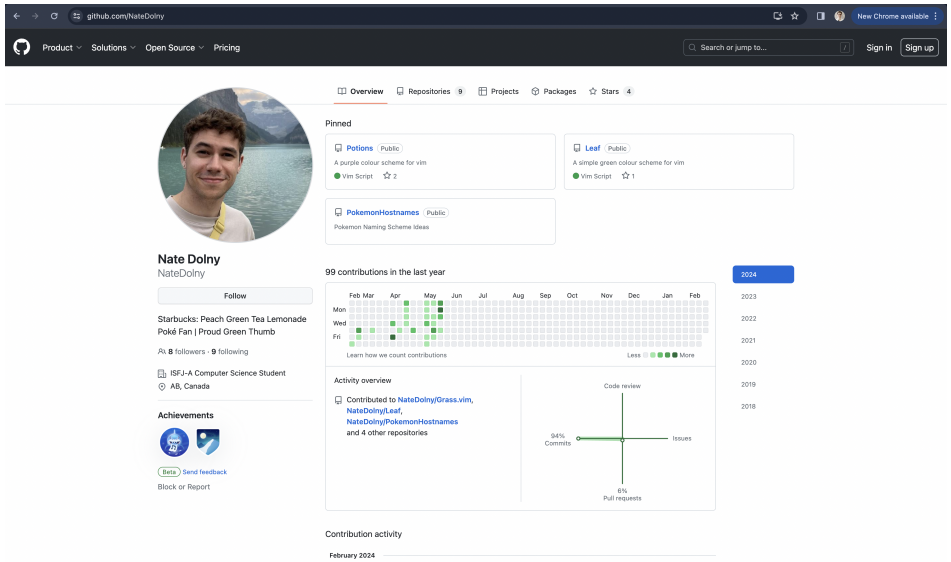
GitLab

Bitbucket

Level Two: GitHub

GitHub

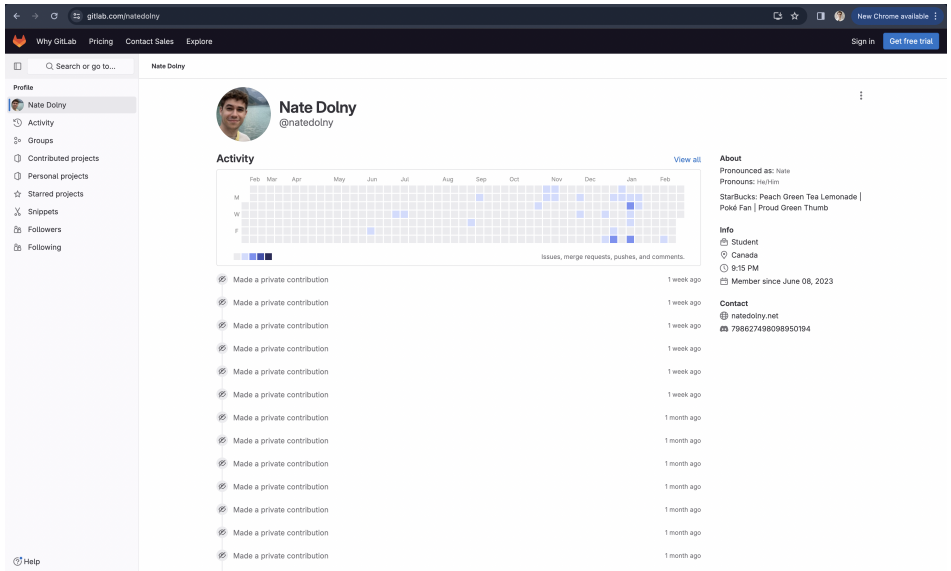
- Released in April 2008
- Microsoft acquired Github in 2012
- Over 100 million users
- Hosts millions of open source projects



Level Two: GitLab

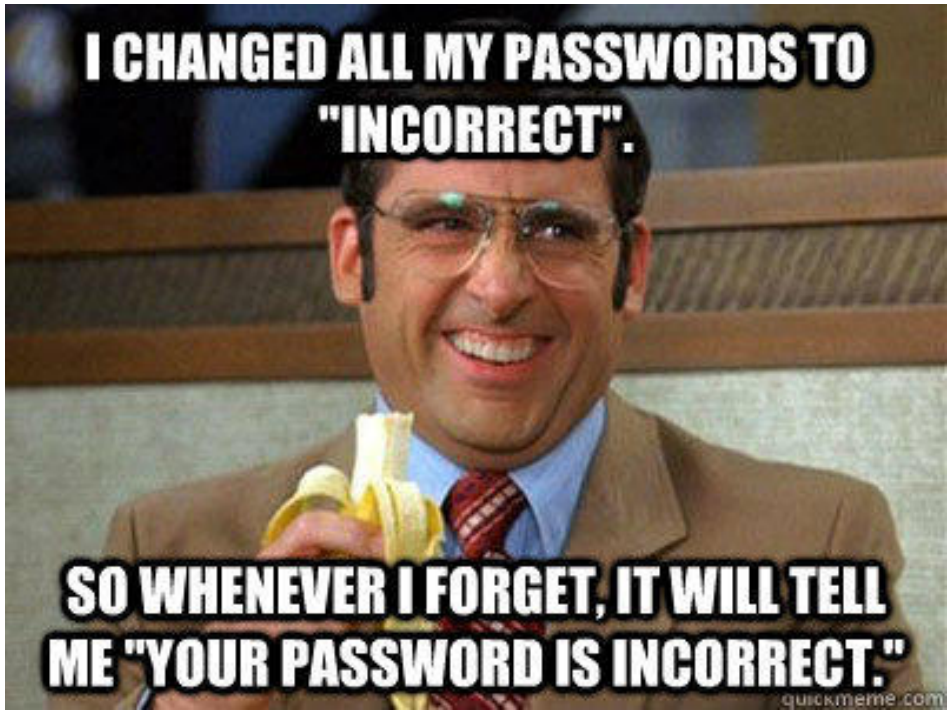
- Released in 2011
- Over 30 million users
- More secure than GitHub
- Focuses on collaboration, efficiency, and automation
- One of the only platform that self-hosting is free, also

checkout gitea



Level Two: Introduction to OpenSSH Suite

- Encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks.
- ssh-keygen: Generates, manages, and converts authentication keys for ssh
- ssh-add: Adds private keys to the authentication agent
- ssh-scp: copies files between hosts on a network



Level Two: Generate an SSH Key

Generate an SSH-Key

- ssh-keygen -t ed25519 -C "Parker.P@midtown.net"

You'll then be given a choice "Enter a file which to save the key (/home/You/.ssh/id_ALGORITHM):"

Option 1. *Name the key*

- Enter your chosen key name and press enter

Option 2. *Press enter*

- Should only choose this option if you will only ever have one key

Congrats you have created your first SSH Key!!

Level Two: Adding an SSH Key to the SSH-Agent

Step 1. *Start ssh-agent in the background*

- eval "$(ssh-agent -s)"

Step 2. *Add your ssh key to the ssh-agent*

- ssh-add /.ssh/$\prec$ *yourKeyName* $\succ$

If you didn't name your key use this instead

- ssh-add /.ssh/id_ed25519

Level Two: Exploring GitLab

Open Firefox or Google Chrome

- https://git.cs.usask.ca/users/sign

Username

- *Your NSID example: abc123*

Password

- *Your Canvas Password*

Level Two: Adding our key to GitLab

Step 1.

- *Click on edit profile*

Step 2.

- *Click on SSH Keys*

Step 3.

- *Click on add new key*

Step 4.

- *Paste public key into key textbox*

Step 5.

- *Give your key a name*

Step 6.

- *Click add key*

Level Two: Testing our connection

Step 1.

- git -T git@git.cs.usask.ca

Step 2.

- *Receive a Welcome Message*

Troubleshooting Steps

- *Re-check you copied the correct public key*
- *Re-check your test command is correct*

Level Two: Git Remote Repository Commands

git clone

- *Creates a copy of an existing repository*

git fetch

- *Updates local references from remote branch but dont merge changes*

git pull

- *Fetches changes from remote repository and automatically merges them into your current branch*

git push

- *Uploads changes to remote repository*

git log

- *Displays the commit history*

Level Two: What is Git Branching?

- Allows you make an isolated copy of your work
- Protects yourself from messing up your main branch

Level Two: Making a Git Branch

git checkout -b $\prec$ *branchName* $\succ$

- *Creates a new branch*

git checkout $\prec$ *branchName* $\succ$

- *Moves you to the specified branch*

git branch

- *Shows all your branches*

git checkout -d $\prec$ *branchName* $\succ$

- *Deletes your specified branch*

git merge $\prec$ *branchName* $\succ$

- *Integrate changes from a branch into another branch*

PLAY

TIME

15 Minutes

Play Time: Can you do this?

Task 1:

- Create a new ssh key named bootcamp_key
- Add the key to your gitlab account

Task 2:

- Create a repository called LevelTwo
- Clone the LevelTwo repo and make a commit

15 Minutes

BREAK

TIME

15 Minutes

LEVEL

THREE

Level Three: Git Remotes

git remote -v

- Lists all remotes

git remote add $\prec$ *name* $\succ\prec$ *url* $\succ$

- Add a remote

git remote remove $\prec$ *name* $\succ$

- Remove a remote

Remotes

Remotes are references to a repository that are typically stored on a remote server.

Level Three: Git Signatures

git blame $\prec$ *fileName* $\succ$

- Show the identity of who edited which line last

git commit -S

- Sign your git commits (a finger print that gets added to the commit) will need to have GNU Privacy Guard installed for this to work

git log –show-signature

- Shows the signed commits of everyone

Level Three: Git Rollbacks

git reset -soft $\prec$ *commit_id* $\succ$

- Resets to a specific commit, keep changes and staged documents from commits made after the commit

git reset -hard $\prec$ *commit_id* $\succ$

- Resets to a specific commit, discards any changes and commits made after the commit

Level Three: Git Stashing

git stash

- Temporarily stores your changes on a stack

git stash list

- Displays the list of stashed changes

git stash apply stash@{Index Number}

- Reapplies the most recent stash to your working directory without removing it from the stash list, you can also reapply based on index number

git stash pop stash@{Index Number}

- Reapplies the most recent stash to your working directory and removes it from the stash list, you can also reapply based on index number

Level Three: Additional Learning Resources

Text Editors:
Vim Website: *www.vim.org*
Vim Tutorial: *www.vim-hero.com*
Vim Manual: *man vim (if on a unix based machine)*

Nano Website: *www.nano-editor.org*
Nano Manual: *man nano (if on a unix based machine)*

Git:
Git Website: *git-scm.com*
Git Manual: *git-scm.com/doc*

GitHub Website: *www.github.com*
GitHub Manuals: *https://docs.github.com/en*

GitLab Website: *www.gitlab.com*
GitLab Manuals: *https://docs.gitlab.com/ee/*

Level Three: The fun continues

Lets play a game

- Navigate to
  the git bootcamp repository git-lab.com/natedolny/gitbootcamp
- Click on the Games directory
- Download the oh-my-git game
- Let the fun begin!!