

# 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.*

# When should you use Git?

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

# 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 --global list`

## PRO TIP

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

# Command-line Text Editors

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

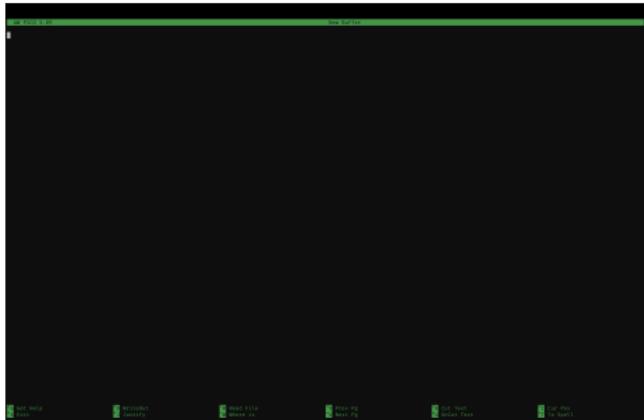


```
     :::  
 .ILE88Dj. :jD888888Dj:  
.LGite888D.f8GjjjlB888E;  
iE :8888Et. .G8888.  
;i E888, :8888,  
D888, :8888:  
D888, :8888:  
D888, :8888:  
D888, :8888:  
888W, :8888:  
W88W, :8888:  
W88W: :8888:  
DGDD: :8888:  
:8888:  
:W888:  
:8888:  
E888i  
tw88D
```



# GNU Nano Text Editor

```
      :::  
     :LE88Dj. :jD888888Dj:  
.LGitE88D.f8GjjjL8888E;  
iE :8888Et. ,G8888  
;i E888, :8888,  
D888, :8888:  
D888, :8888:  
D888, :8888:  
D888, :8888:  
888W, :8888:  
W88W, :8888:  
W88W, :8888:  
DGDD: :8888:  
:8888:  
:8888:  
:8888:  
:8888:  
:8888:  
E888i  
tW88D
```

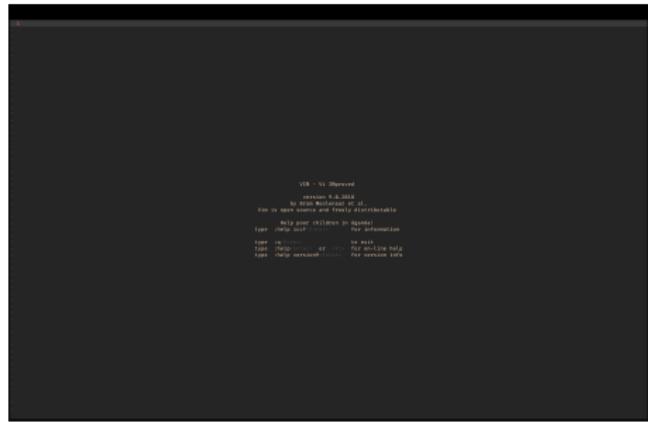


- Released in November 18, 1999
- User-friendly and easy to learn
- Developed and maintained by volunteers

# Vim Text Editor



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

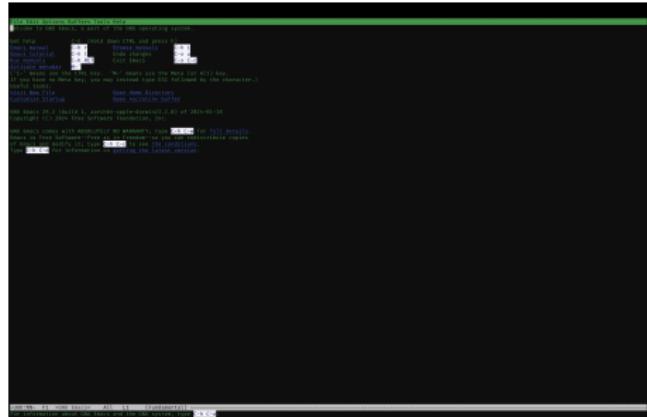


# Emacs Text Editor



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

- Many modes for different purposes such as browsing the web



# 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



# What makes a good commit?

## A better approach to making commits

- *Title*: Add a topic
- *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.

# Making a Commit

## **Step 1.** Add a document to staging

- git add <document>

## **Step 2.** View documents that have been staged (Optional)

- git status

## **Step 3.** Remove a document from staging (Optional)

- git reset <document>

## **Step 4.** Record changes to repository

- git commit

## **Step 5.** Upload changes to remote repository

- git push

## 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



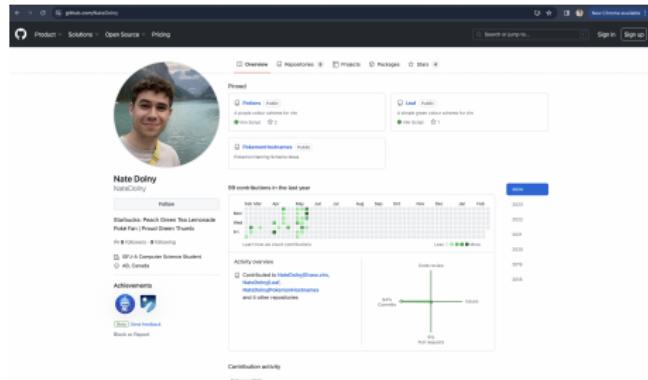
# GitLab



# Bitbucket

# GitHub

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





# GitLab

- Released in 2011
- Over 30 million users
- More secure than GitHub
- Focuses on collaboration, efficiency, and automation

- Only platform that self-hosting is free

The screenshot shows a GitLab user profile for 'Nate Dolny'. The profile includes a photo, a bio ('Prototypist'), and activity metrics: 1 issue, 1 merge request, 1 pull request, and 1 commit. Below the activity chart is a list of 15 recent contributions, all labeled 'Made a private contribution'. The sidebar on the left lists 'Profile' sections: Home (only), Activity, Milestones, Contributed projects, Personal projects, Recent activity, Requests, Followers, and Following.

# 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



# 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!!

# 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/`yourKeyName`

If you didn't name your key use this instead

- ssh-add /.ssh/id\_ed25519

# Exploring GitLab

Open Firefox or Google Chrome

- [https://git.cs.usask.ca/users/sign\\_in](https://git.cs.usask.ca/users/sign_in)

Username

- Your NSID example: abc123

Password

- Your Canvas Password



# 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*

# 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

# Important Git 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*

# What is Git Branching?

- Allows you make an isolated copy of your work
- Protects yourself from messing up your final draft



# Making a Git Branch

**git checkout -b ↵ branchName ↴**

- Creates a new branch

**git checkout ↵ branchName ↴**

- Shows all your branches

**git checkout -d ↵ branchName ↴**

- Deletes your specified branch

**git merge ↵ branchName ↴**

- Integrate changes from a branch into another branch

# What if we need to rollback?

**git reset -soft** ↵ commit\_id ↶

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

**git reset -hard** ↵ commit\_id ↶

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

# Additional Learning Resources

## Text Editors:

Vim Website: [www.vim.org](http://www.vim.org)

Vim Tutorial: [www.vim-hero.com](http://www.vim-hero.com)

Vim Manual: `man vim` (if on a unix based machine)

Nano Website: [www.nano-editor.org](http://www.nano-editor.org)

Nano Manual: `man nano` (if on a unix based machine)

## Git:

Git Website: [git-scm.com](http://git-scm.com)

Git Manual: [git-scm.com/doc](http://git-scm.com/doc)

GitHub Website: [www.github.com](http://www.github.com)

GitHub Manuals: <https://docs.github.com/en>

GitLab Website: [www.gitlab.com](http://www.gitlab.com)

GitLab Manuals: <https://docs.gitlab.com/ee/>