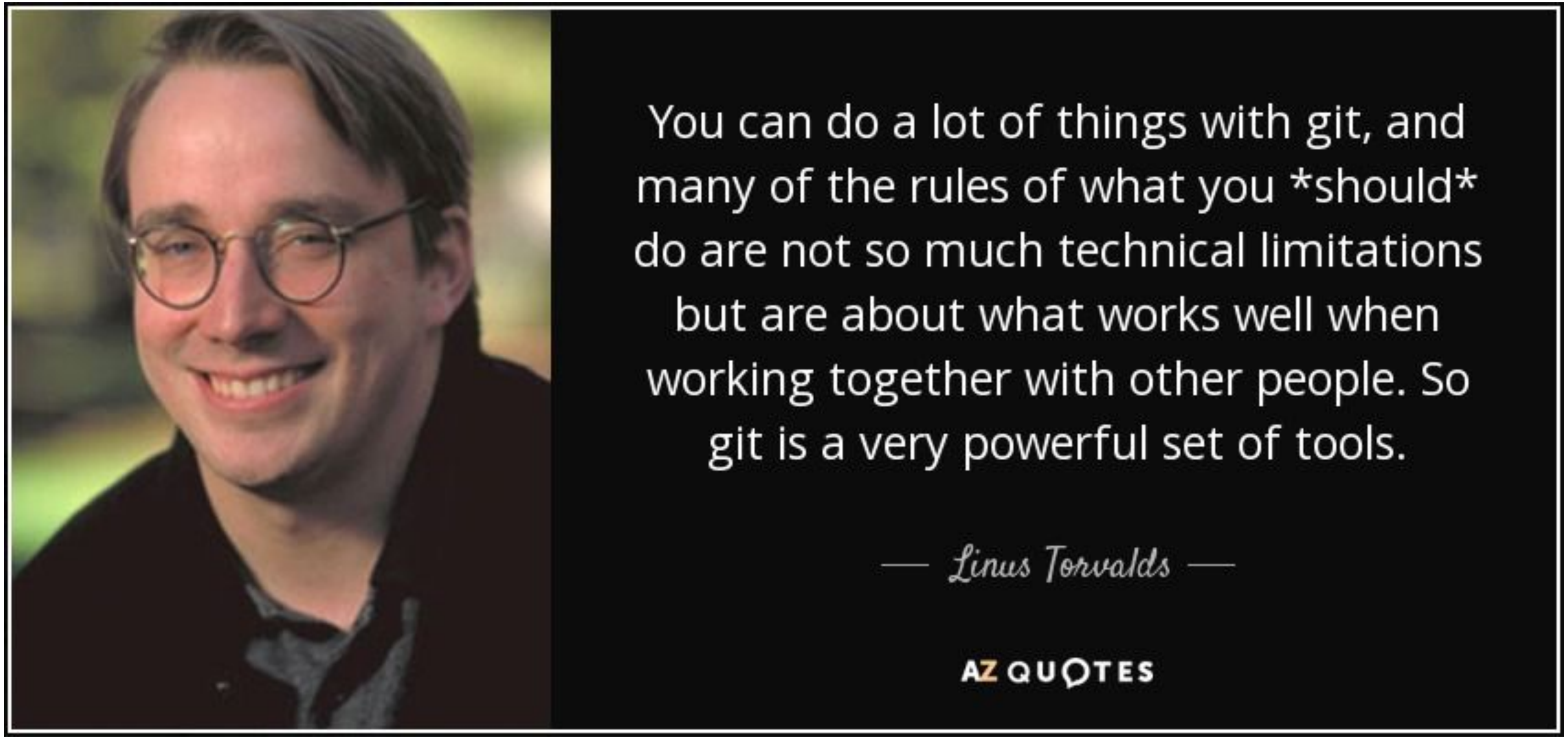
Git & Github

Michael Gathara, Christian Collins

Please Scan for Attendance

Overview

- What git is and isn’t
- Why use git
- A high level overview of git
- A live demo of git
- In-class assignment



You can do a lot of things with git, and many of the rules of what you *should* do are not so much technical limitations but are about what works well when working together with other people. So git is a very powerful set of tools.

— Linus Torvalds —

AZ QUOTES

Git vs Github/Gitlab/BitBucket

Git:

- Source/Version Control
- Tracks changes made to code
- Works locally
- The underlying technology for Github, Gitlab, Code-Commit, and Bitbucket
- Learn it once, Use it everywhere
- Can be self-hosted to create your very own Github

Github/Gitlab/BitBucket:

- Cloud-based hosting service that lets you manage repositories
- Utilizes Git as it's underlying technology, hence "Git as a service"
- Graphical user interface to view files
- No need to know Git to use
- Super popular

Why Source/Version Control?

- Keep track of changes within files regardless of the amount of users making those changes
- Create checkpoints as you work on a problem
- See your progress through time
- Access your files anywhere in the world

Keywords

- **Repository**
  - Similar to a directory, stores everything related to your project including files, versions, commits, etc.
- **Clone**
  - Creates a linked copy of a repository that will sync with the original.
- **Fork**
  - Creates an independent copy of a repository.
- **Branch**
  - A version of the repository that allows one to test changes without testing the main repository

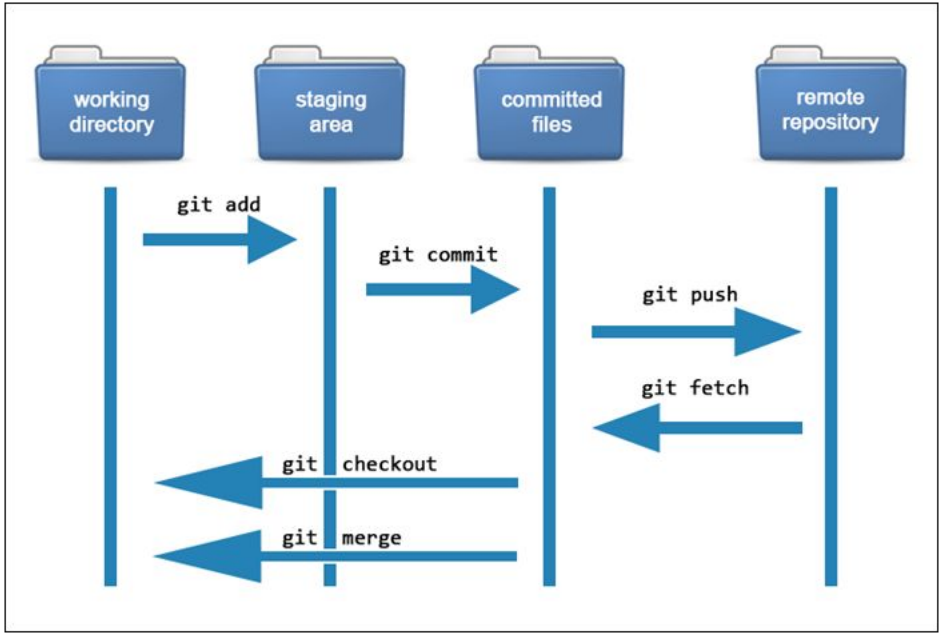
Keywords

- **Add**
  - When making changes, using add will store the file on a staging area where it will wait to be included in the next commit.
- **Commit**
  - A ‘snapshot’ of changes made to the file, branch, or repository.
- **Push**
  - Updates the remote repository with the commit.
- **Origin**
  - The primary or original version of a repository.

How Git Works

Git:

- Working Directory
  - Your current files, where you make changes
- Staging Area
  - Git starts keeping track of your files
- Committed Area
  - A screenshot of your working directory, ready to be sent to Github
- Remote Repo
  - Github, Gitlab, Bitbucket, etc



https://phoenixnap.com/kb/how-git-works#:~:text=Git%20allows%20users%20to%20track,and%20track%20each%20one%20independently.

Syntax

clone

```
git clone <repository url>
```

Retrieves repository from a remote location on local machine

add

```
git add .
```

Add all changes to next commit

commit

```
git commit -m “message”
```

Creates a snapshot of the changes to the code

push

```
Git push
```

Pushes commit to repository’s current branch

status

```
git status
```

Shows files ready for next commit

log

```
git log
```

Shows commit history for the branch

Some Best Practices

- Commit often
    - Debugging
        - It will be easier to identify which change in the code caused an issue, allowing to revert back to a previous version before that change was made
    - Readability
        - Allows for more specific commit messages so that there is a better understanding of what is being changed
- Have good commit messages
    - Be clear and concise when describing the commit
        - Good: “Add test case for functionA”
        - Bad: “added to some functions”

Some Best Practices

- Use structural elements
  - `<type>[optional scope]: <description>`
    - `feat: allow provided config object to extend other configs`
  - `fix:` a commit of the type fix patches a bug in your codebase
  - `feat:` a commit of the type feat introduces a new feature in your codebase
  - Other types are allowed as well
    - `Build:, chore:, ci:, docs:, style:, refactor:, perf:, test:`
- Further Reading
  - https://www.conventionalcommits.org/en/v1.0.0/

Live Demo: A live showcase of Git/Github working

- Download Git if you don’t have it yet (Git - Downloads)
- Creating an organization to host your UAB files (optional)
- Creating a repo for a class and some good practices for structuring files
- Git add
- Git commit
- Git push
- Git checkout
- Git pull

Git Exercise

Navigate over to https://acmatuab.org/assignment

Follow the instructions to use what you learned

Discover pull requests, branches and merging

Ask for help!

mikegtr@uab.edu collincj@uab.edu

CS Clubs @ UAB

ACM
uabacm.org/

WIT
Instagram:
uab_womenintech

Questions?/Resources

Ray Winderlich, How Git Actually Works:

https://www.raywenderlich.com/books/advanced-git/v1.0/chapters/1-how-does-git-actually-work

Version Control with Git: https://www.udacity.com/course/version-control-with-git--ud123

TechTips Repository used in class: https://github.com/iamchristiancollins/TechTips