An Introduction to Git

By Adithya Nair

Missing Semester

An initiative where students teach students.

The Purpose Of This Presentation

- Teach the basics of version control and collaborative programming
- Equip all of you with the ability to manage your own projects with ease.

“Generally the best way to learn git is to probably first only do very basic things and not even look at some of the things you can do until you are familiar and confident with the basics”

~Linus Torvalds, creator of Git(and the Linux kernel)

Problems I Noticed

- Sending code through WhatsApp
- Losing progress and code changes.
- Making duplicates of files that get lost.

What Is Git?

- Git tracks the changes you make to your files
- Git is local-first
- Git tends to only add data

Git is a camera, for your files.

Git is a time machine, for your files.

A Technical Breakdown Of What's Going On

Necessary Vocabulary

- Commit
- Repository
- Branch
- Staging

How Does Git Work?

The Three States

1. Modified
2. Staged
3. Commited

The Basic Git Workflow

1. You modify files.
2. Stage only the changes which should be part of the next commit
3. Commit, storing that snapshot permanently to your Git directory.

Doubts

Installation

Email And Name

```
git config --global user.name "Adithya Nair"
git config --global user.email "adithyanair121@gmail.com"
```

Initialize A Repo

```
git init
```

Cloning A Repo

```
git clone <repo-name>
```

Check Status

```
git status
# For a shortened status
git status -s
```

Adding Files

```
git add <file-name>
```

Ignoring Files

```
.gitignore

*.out
```

Committing Files

```
git commit
# Inline the commit message within the shell
git commit -m "<Message>"
```

Checking Logs

```
git log
# For a shorter log
git log -s
```

Create A New Branch

```
git branch <branch-name>
```

Switch To A Branch

```
git checkout <branch-name>
# You can also use
git switch <branch-name>
```

An Easier Way To Access Commits

```
# The current commit being worked on
HEAD
# The previous commit
HEAD^
# The nth previous commit
HEAD~n
```

Rolling back Changes

```
# View the commit, without saving changes
git checkout <commit>
# Revert changes made by creating a new commit
git revert <commit>
# Use this for a hard reset NOT RECOMMENDED
git reset --hard <commit>
```

Note that <commit> should be replaced with the unique id generated for each commit

Merge Branches

```
git checkout <branch-to-merge-to>
git merge <branch-to-be-merged-with>
```

Delete Branches

```
git branch -d <branch-name>
```

How Would Collaboration Look?

Demo Workflow

A quick demonstration of how I use git.

An Organization For Hosting Your Project Repos

Topics To Take Up

1. Self-Hosting
2. Vim motions
3. Bash Scripting
4. RegEx
5. Unit Testing

Meta Information

- This presentation was built with a framework called Marp
- Notes were organized using Emacs Org-Mode

References

- MIT Lecture On Git
- Pro Git

Thank You