

# Git Good

Linus Torvalds' Magical Time Machine 🕒

---











Timothy

2021-03-30











**Keeping track of files is hard**

---

## We've all seen it before 😞

Name	Type
 project draft.doc	Microsoft Word 9...
 project draft1.doc	Microsoft Word 9...
 project final.doc	Microsoft Word 9...
 project final1.doc	Microsoft Word 9...
 project final2.doc	Microsoft Word 9...
 project FINAL final.doc	Microsoft Word 9...
 project FINAL final1.doc	Microsoft Word 9...
 project FINAL final2.doc	Microsoft Word 9...
 project final THIS IS THE ONE TO SUBMIT.doc	Microsoft Word 9...
 project final THIS IS THE ONE TO SUBMIT v2.doc	Microsoft Word 9...

## We've all seen it before (2) 😞😞

Name
 Press release for approval.doc
 Press release final.doc
 Press release FINAL VERSION.doc
 Press release FINAL FINAL VERSION.doc
 IMPROVED FINAL PRESS RELEASE.doc
 REVISED APPROVED FINAL PRESS RELEASE.doc
 REVISED APPROVED FINAL PRESS RELEASE v. 2.doc
 !! NEW REVISED APPROVED FINAL PRESS RELEASE v. 2.doc
 !!! REVISED NEW REVISED APPROVED FINAL PRESS RELEASE v. 2.doc
 !!!! Press release as sent.doc

## The CompSci Group Project issue

- Your group project is due tonight and your code is ready for submission
- While running a final test (for good luck 🍀) you discover a minor **bug** your group member forgot about
- You accidentally changed working code and ended up breaking a chunk of code you don't understand 🤖
- You no longer remember what was and wasn't there
- It is 23:58 😓

## An actual physics report I was working on

- I'm a last minute person, report is due tomorrow
- It's manageable, I typed up  $\frac{2}{3}$  of it yesterday
- I'm moving an image in the *Discussion* section around, Microsoft Word freezes ❄️
- I wait a few minutes, oh well, looks like I need to Force Quit
- Re-open the document: Word complains that it's corrupted, and that it can't restore the file 😞
- I look at my Report v3.docx, it's from several hours ago
- I need to re-do at least half of the work I've done 😞
- I settle in for a later night than I was planning on... ⌚

# Introduction

---

## Version control systems — recording changes to files over time

It keeps track of changes.

It keeps track of changes *really* well.

- My changes 😊
- Your group member's changes 😊
- The version that's been sent to the customer 🤔
- The version that's being tested with an urgent bugfix 😊
- The new experimental version that might break everything else 🤖
- Whatever other changes have happened... 🍷



# What is Git?

- Open source project *circa*. 2005 (developed for the Linux kernel 🐉)
- A command line utility (but there are graphical interfaces)
- Imagine `git` as something sitting on top of your file system
- A distributed 🌐 version control system — DVCS
- Keeps all change info in a `.git` folder

## Aside: Distributed Version Control Systems (DVCS)

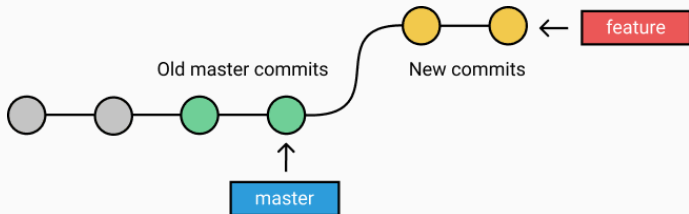
A thing that solves our earlier problems

**Version control system** A system recording changes to files over time

**Distributed** No main server, full history available to anyone

# Git: A structured way of storing versions and changes

- Every change is stored as a new compressed file, and can easily generate the difference with a previous version
- This creates a structure known as a DAG<sup>1</sup>
- `git` operates on—and provides an interface to—this structure



---

<sup>1</sup>Directed Acyclic Graph

## What does this mean?


You'll have no reason to **ever** lose your work again, and *unparalleled* ability to traverse and review changes made.

## Getting started

---

- Download `git`
- There are nice graphical interfaces, but we'll introduce those later
- The git commands are universal, and the simplest way to get started

## Creating a repository (i.e. project)

Create a new directory  (e.g. my big project), open a terminal within it and execute

```
git init
```

Congratulations, you've created a repository 

## Clone a repository

Want to build on somebody else's work? That's easy!

```
git clone /path/to/repository
```

This also works with remote servers, e.g. `git clone https://github.com/tecosaur/vscode-settings.git`.



Local repo contains three trees in the .git file

- Working directory — Where all the files are
- Index — The staging area
- HEAD — The latest commit

## Add and commit

After changing some files you add it to the index tree

```
git add $filename  
git add *
```

Then to actually commit them to the HEAD

```
git commit -m "Message"
```

All commits need a message and are given an identifying serial number, which is a hash of its structure and contents.

Your changes are still local however

## What's with these messages?

There is no hard and fast rule but the idea is to **describe** what this commit does

- Adds support for x. Fixes bug that caused y.
- Removes changes made in commit z.
- Add discussion of errors to report

Looking at a list of commit messages should give a decent idea of what happened to the code. Try to improve on the following commit messages:

- General commit
- Made some changes
- Bug-fixes
- Debugging 4: Reloaded (continued)
  - *I've actually done this before*

## So, what are we doing?

Git tracks the changes between files

- Additions
- Deletions

Starting from a HEAD and given a sequence of commits, you could follow what each says and end up at the same final point

## The next few steps

---

## Pushing and Pulling

To send your committed changes back to someone else's copy 📡

```
git push origin $branch
```

And to fetch some new changes 📡

```
git pull
```

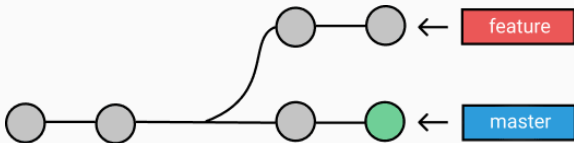
# Branching

Used to develop features isolated from the main code (remember your tragic mistake just before submitting?) either master or main is the default branch. To create a new branch named “feature”

```
git checkout -b feature
```

Switch back to master/main

```
git checkout master/main
```



A branch — or any other change — is **not available to others unless** you **push** it to your remote repository

```
git push origin $branch
```



# Merging

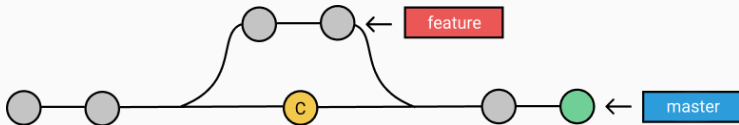
To bring down any changes made by someone or somewhere else

```
git pull
```

To merge another branch into your current branch (e.g. master)

```
git merge $branch
```

This will attempt to merge any changes from \$branch into the branch your current HEAD is on



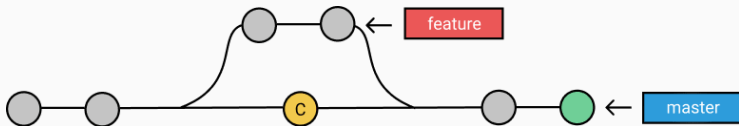
## Merge Conflicts

In both cases `git` tries to auto-merge changes. **Conflicts** — two different changes to the same part of the same file — need to be resolved manually and added before trying again.

```
git add $filename
```

One can also **preview** changes (to see if conflicts will occur ☹️)

```
git diff $source_branch $target_branch
```



# Tagging

When you hit a major milestone 🎉 (it works, we've submitted etc.) you can tag a particular commit

```
git tag 1.0.0 2b331ie545
```

The 2b... is the first 10 characters of the commit ID you want to reference. You can find a commit ID by looking at the [log](#).

You can study a repository history using `git log`. There are many nice options

```
git log --pretty=oneline
```

Or maybe you'd like to see some ASCII art

```
git log --graph --oneline --decorate --all
```

See only which files have changed

```
git log --name-status  
git log --help
```

# Ooops

Halp, I destroyed **everything**.

All the files are gone, I've replaced them with photos of Nicholas Cage, I'm tired and in need of a hug.

*I wish I could go back in time* 🕒



## No problem!

*I wish I could go back in time 🕒*

You can! You can replace local changes with

```
git checkout -- $filename
```

Or if you've committed a bunch of stuff that's wrong and just want to copy from the main repository — you could `git clone` a new copy, or you can

```
git fetch origin  
git reset --hard origin/master
```

## Summary

---

## Summary

- Git tracks changes to files
  - Changes are stored as commits (cheap and plentiful)
  - A group of commits can be pushed (setting them in stone — kinda)
- Branches allow you to develop new things in isolation
  - Can be merged back together again when finished
  - You can move through time and code together

There's a lot more you can recover from



GitHub is a popular online host — software forge — for Git. Plus some (a lot) of stuff

Alternatives include:

- [GitLab](#)
- [BitBucket](#)
- [SourceHut](#)

These take care of backups and many, many fancy things you will likely learn in good time