

Git 101:

Git and GitHub for beginners

Overview

1. Install git and create a Github account
2. What is git?
3. How does git work?
4. What is GitHub?
5. Quick example using git and GitHub

Install git and a create GitHub
account

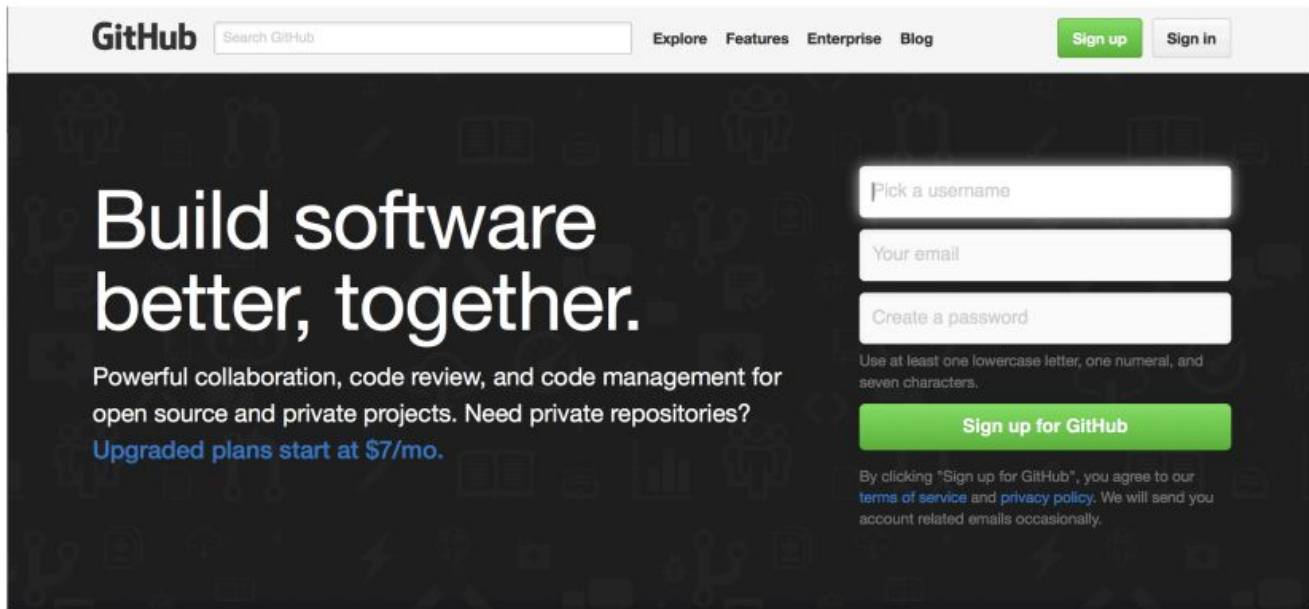
Install git

- Linux (Debian)
 - Command: `sudo apt-get install git`
- Linux (Fedora)
 - Command: `sudo yum install git`
- Mac
 - <http://git-scm.com/download/mac>
- Windows
 - <http://git-scm.com/download/win>

Create Github account

www.github.com

Free for public repositories

A screenshot of the GitHub website's sign-up page. The header features the GitHub logo, a search bar, and navigation links for Explore, Features, Enterprise, and Blog. There are 'Sign up' and 'Sign in' buttons in the top right. The main content area has a dark background with the text 'Build software better, together.' and a description of GitHub's services. On the right, there are three input fields for 'Pick a username', 'Your email', and 'Create a password', followed by a green 'Sign up for GitHub' button. A disclaimer at the bottom states that by signing up, the user agrees to the terms of service and privacy policy.

What is version control?

- A system that keeps records of your changes
- Allows for collaborative development
- Allows you to know who made what changes and when
- Allows you to revert any changes and go back to a previous state

What is git?

What is version control?

- Distributed version control
- Users keep entire code and history on their location machines
 - Users can make any changes without internet access
 - (Except pushing and pulling changes from a remote server)

What is git?

- Started in 2005
- Created by Linus Torvald to aid in Linux kernel development



What is git?

Git isn't the only version control system



But (we think) it's the best

How does git work?

How does git work?

- Can be complicated at first, but there are a few key concepts
- Important git terminology in following slides are blue

Key Concepts: Snapshots

- The way git keeps track of your code history
- Essentially records what all your files look like at a given point in time
- You decide when to take a snapshot, and of what files
- Have the ability to go back to visit any snapshot
- Your snapshots from later on will stay around, too

Key Concepts: Commit

- The act of creating a snapshot
- Can be a noun or verb
 - “I committed code”
 - “I just made a new commit”
- Essentially, a project is made up of a bunch of commits

Key Concepts: Commit

- Commits contain three pieces of information:
 1. Information about how the files changed from previously
 2. A reference to the commit that came before it
 - Called the “parent commit”
 3. A hash code name
 - Will look something like:

f2d2ec5069fc6776c80b3ad6b7cbde3cade4e

Key Concepts: Repositories

- Often shortened to 'repo'
- A collection of all the files and the history of those files
- Consists of all your commits
- Place where all your hard work is stored

Key Concepts: Repositories

- Can live on a local machine or on a remote server (GitHub!)
- The act of copying a repository from a remote server is called [cloning](#)
- Cloning from a remote server allows teams to work together

Key Concepts: Repositories

- The process of downloading commits that don't exist on your machine from a remote repository is called **pulling** changes
- The process of adding your local changes to the remote repository is called **pushing** changes

Key Concepts: Branches

- All commits in git live on some branch
- But there can be many, many branches
- The main branch in a project is called the **master** branch

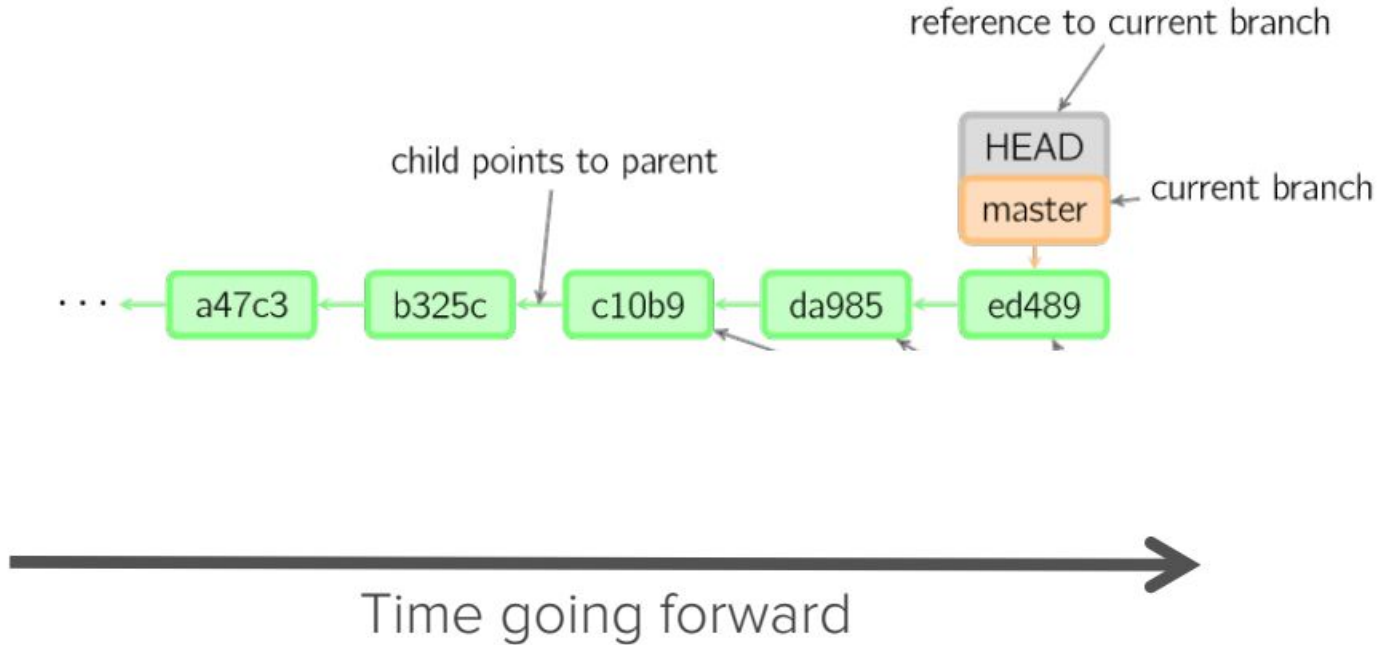
So, what does a typical project look like?

- A bunch of commits linked together that live on some branch, contained in a repository
- Following images taken and modified from:

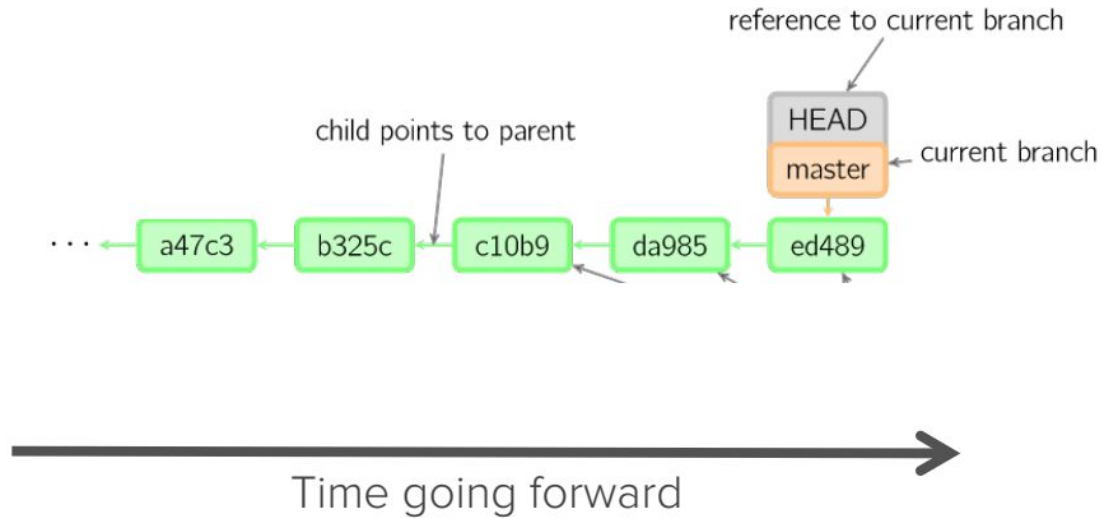
<http://marklodato.github.io/visual-gitguide/index-en.html>

- Also a good tutorial!

So, what does a typical project look like?

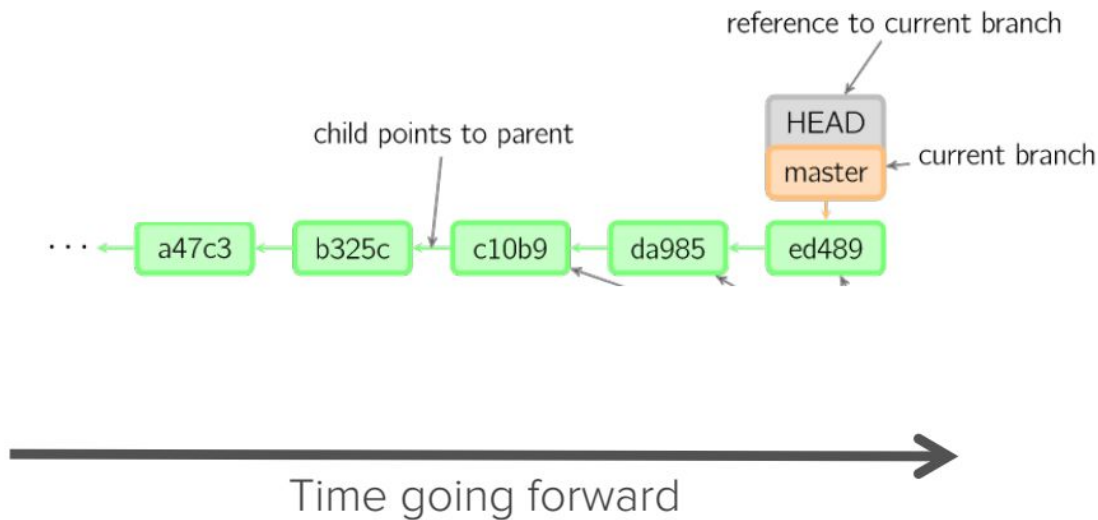


So, what is **HEAD**?



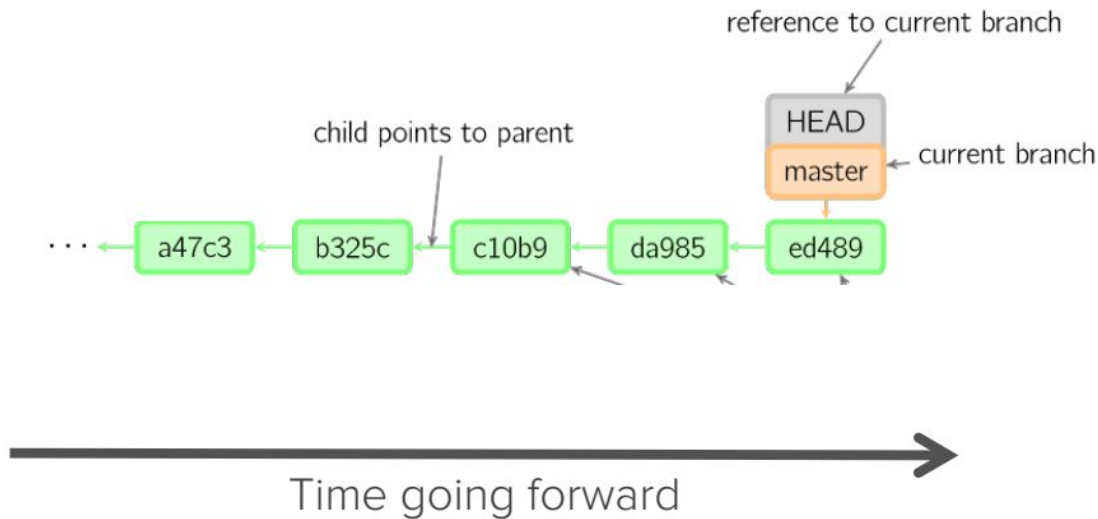
So, what is HEAD?

- A reference to the most recent commit
 - (in most cases – not always true!)



So, what is **MASTER**?

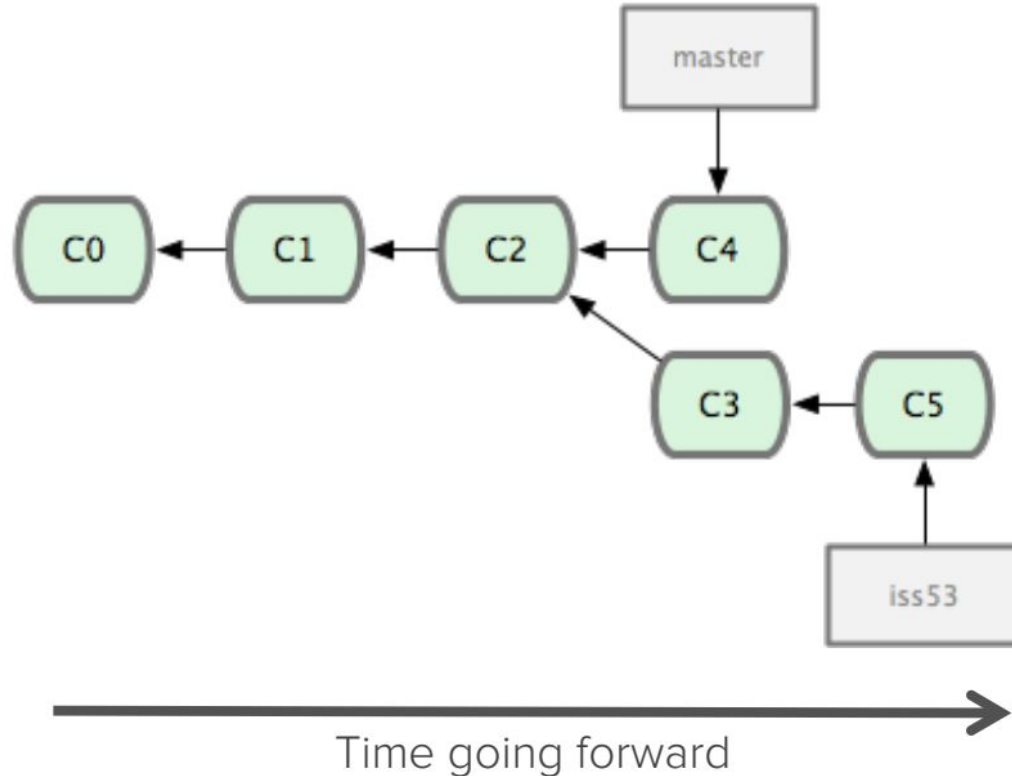
- The main branch in your project
- Doesn't *have* to be called master, but almost always is!



Key Concepts: Branching off of the **master** branch

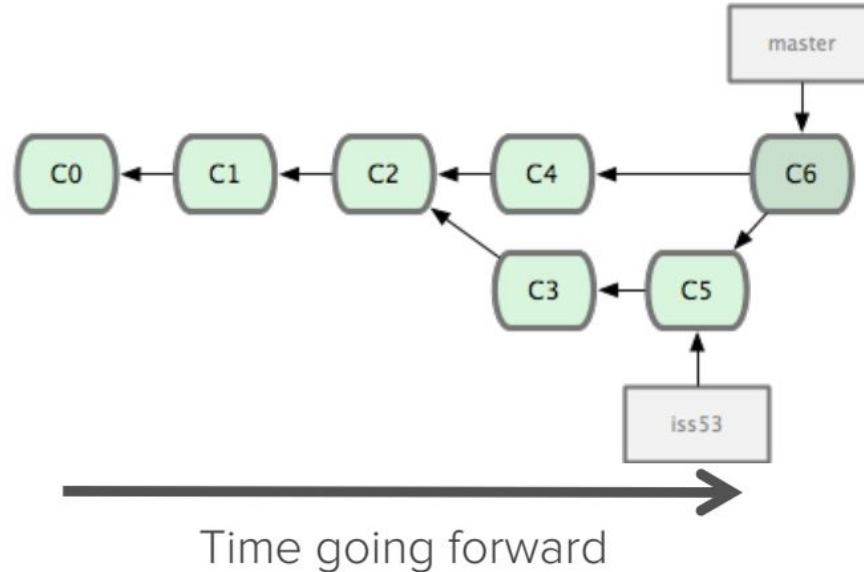
- The start of a branch points to a specific commit
- When you want to make any changes to your project you make a new branch based on a commit

Key Concepts: Branching off of the **master** branch



Key Concepts: Merging

Once you're done with your feature, you merge it back into **master**



Key Concepts: How do you make a commit anyway?

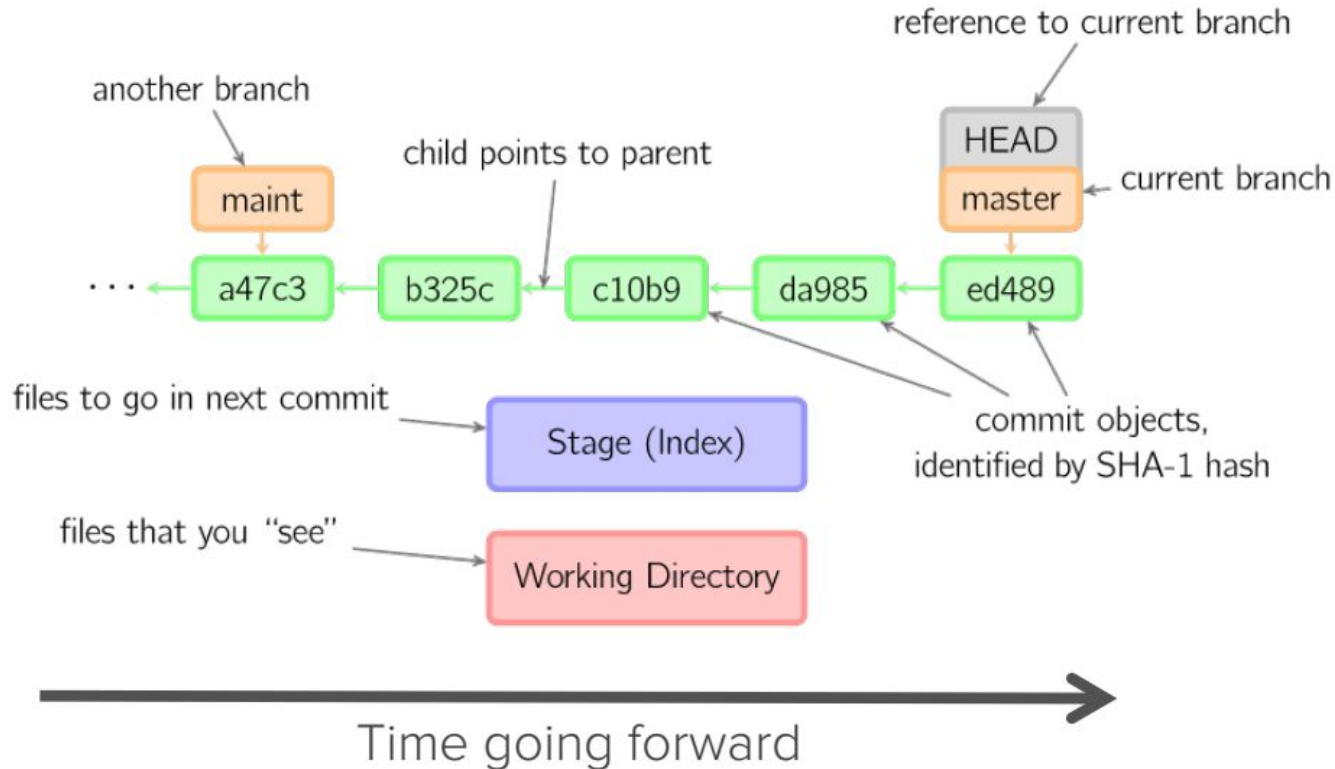
- There are a lot of 'states' and 'places' a file can be
- Local on your computer: the 'working directory'
- When a file is ready to be put in a commit you add it onto the 'index' or 'staging'
- Staging is the new preferred term – but you can see both 'index' and 'staging' being used

Key Concepts: How do you make a commit anyway?

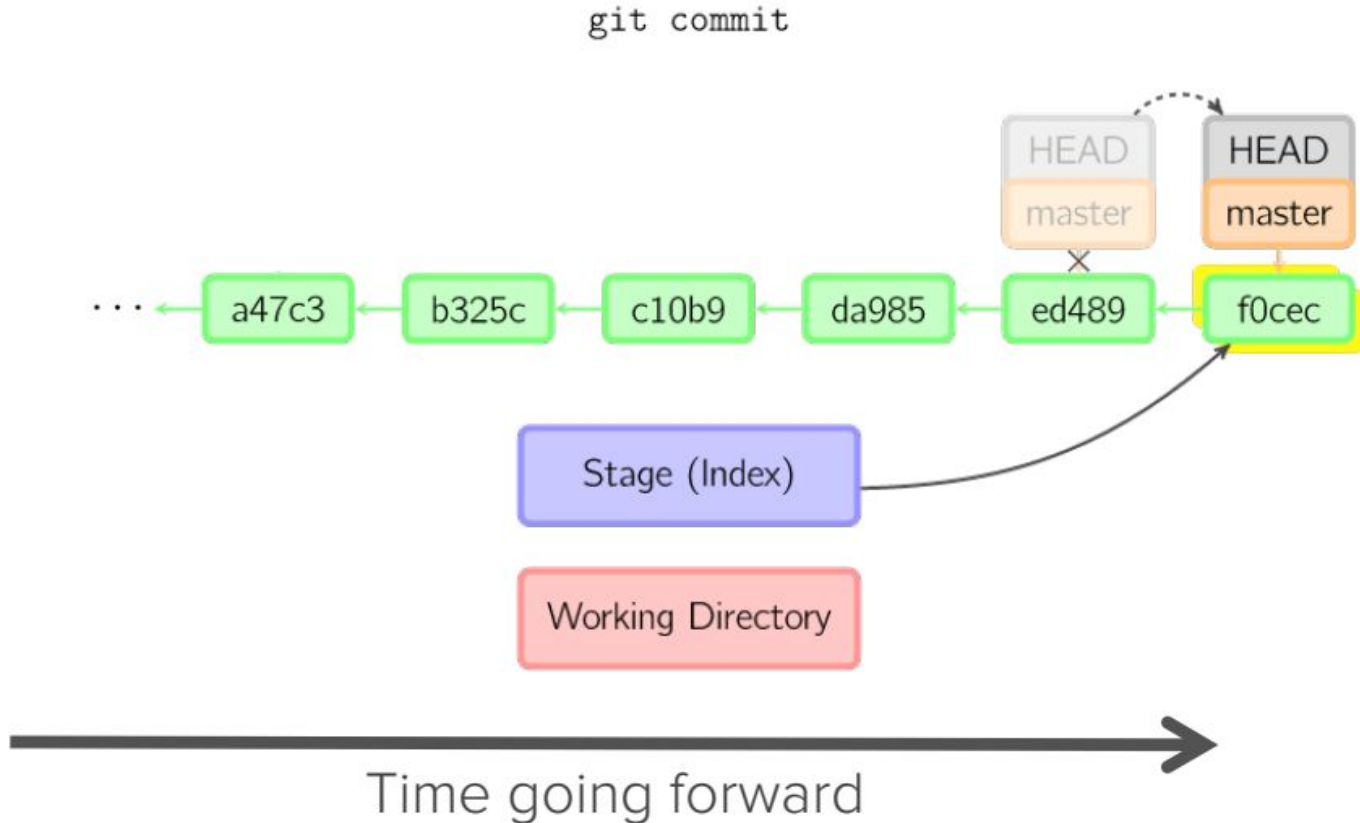
The process:

- Make some changes to a file
- Use the 'git add' command to put the file onto the staging environment
- Use the 'git commit' command to create a new commit

Key Concepts: How do you make a commit anyway?



Key Concepts: How do you make a commit anyway?



What is GitHub?

What is GitHub?

- www.github.com
- Largest web-based git repository hosting service
 - Aka, hosts 'remote repositories'
- Allows for code collaboration with anyone online
- Adds extra functionality on top of git
 - UI, documentation, bug tracking, feature requests, pull requests, and more!



What is GitHub?

- Founded in 2008
- Also has an Enterprise edition for businesses



Additional Resources

- Official git site and tutorial:
 - <https://git-scm.com/>
- GitHub guides:
 - <https://guides.github.com/>
- Command cheatsheet:
 - <https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf>
- Interactive git tutorial:
 - <https://try.github.io/levels/1/challenges/1>
- Visual/interactive cheatsheet:
 - <http://ndpsoftware.com/git-cheatsheet.html>