

An intro to Git and GitHub - Contributor Role

DIME Analytics

November 30, 2020

DIME - The World Bank - <https://www.worldbank.org/en/research/dime>

Before the session starts:

1. Do you have a GitHub.com account? If not, got to <https://github.com/join> and sign up
2. Have you sent your GitHub username to the organizer or to kbjarkefur@worldbank.org ?
3. Have you installed GitHub Desktop? If not got to <https://desktop.github.com/> and download it.
4. Have you logged in at least once on GitHub Desktop? If not open GitHub Desktop and log in using your GitHub account.
5. Have you been invited to github.com/kbjarkefur/lyrics ?
6. And have you accepted at github.com/kbjarkefur/lyrics/invitations ?

An intro to Git and GitHub - Contributor Role

DIME Analytics

November 30, 2020

DIME - The World Bank - <https://www.worldbank.org/en/research/dime>

Three common GitHub roles

The objective of this training is to make you able to take the role of a Contributor. See DIME Analytics GitHub Roles for full details. (link at the end of the presentation)

Observer

- Browse code in GitHub
- Provide feedback through GitHub
- **Who?** Typically a PI that does not code

Contributor

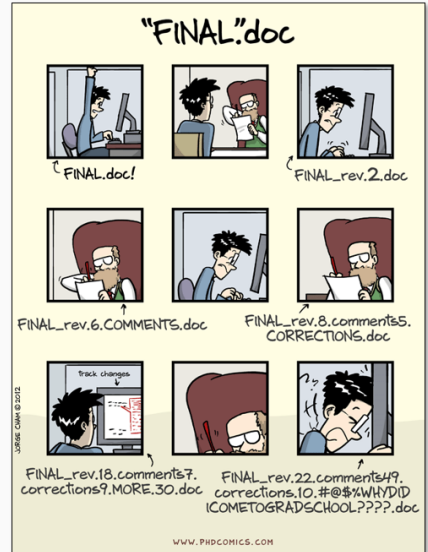
- Contribute to code in GitHub
- Understand and follow instructions from a Repo Maintainer
- **Who?** Typically an RA, or a PI that codes

Repo Maintainer

- Make sure that best practices and standards are followed in the repository
- Guide new contributors
- **Who?** Typically the most senior RA. Takes too much time for a PI.

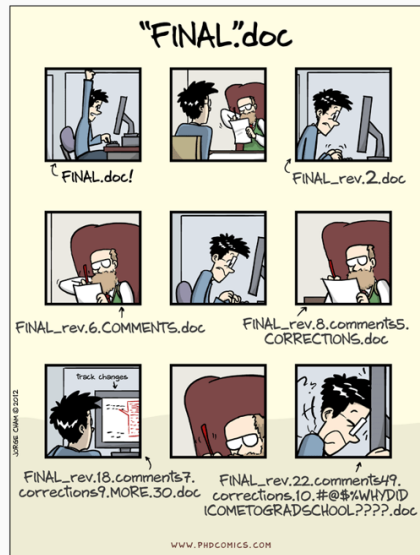
What is Git used for?

- Git solves the *Final.doc* problem



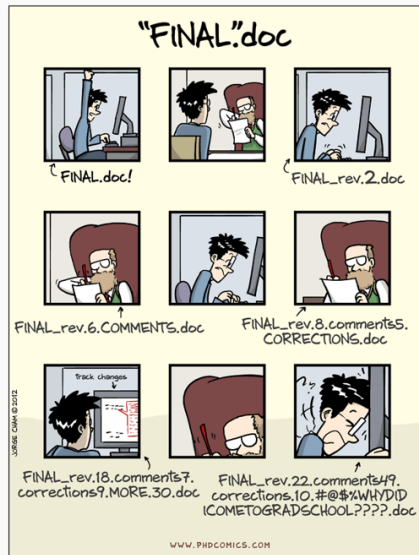
What is Git used for?

- Git solves the *Final.doc* problem
- Common solution to the *Final.doc* problem:
Name all your docs like
YYMMDD_docname_INITIALS.doc



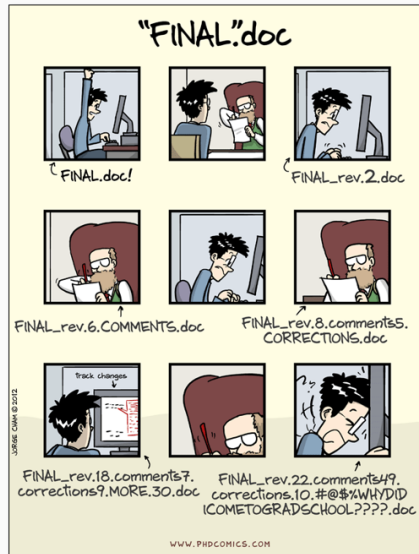
What is Git used for?

- Git solves the *Final.doc* problem
- Common solution to the *Final.doc* problem:
Name all your docs like
YYMMDD_docname_INITIALS.doc
- Git tracks *YYMMDD* and *INITIALS* for all
edits without the user having to remember it

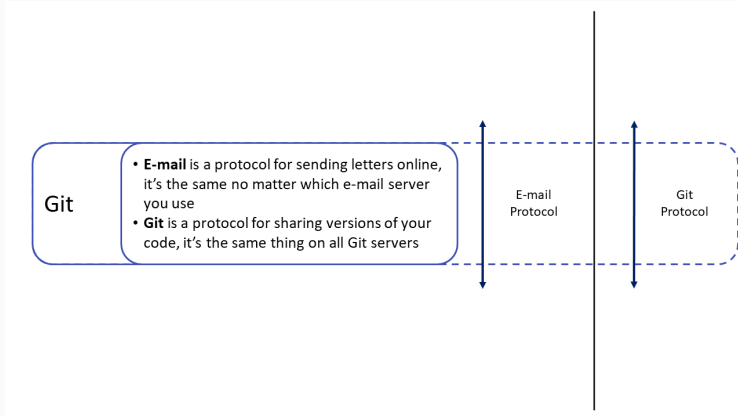


What is Git used for?

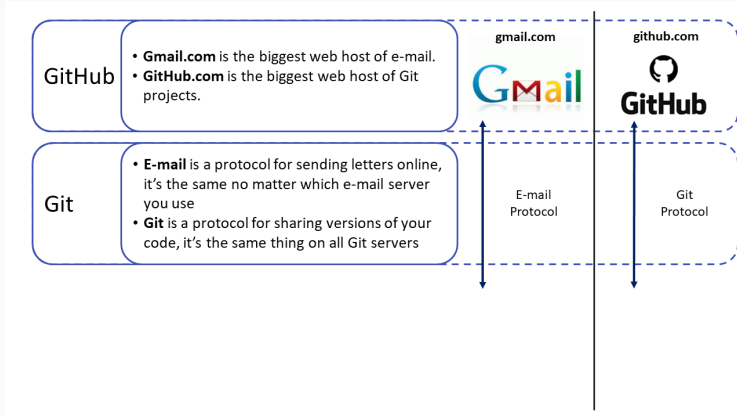
- Git solves the *Final.doc* problem
- Common solution to the *Final.doc* problem:
Name all your docs like
YYMMDD_docname_INITIALS.doc
- Git tracks *YYMMDD* and *INITIALS* for all edits without the user having to remember it
- That's far from everything, Git also solves:
 - Conflicting copy problem (DropBox etc.)
 - I can't re-produce my Baseline report problem
 - Who wrote this code 4 years ago and why?
 - And much much more...



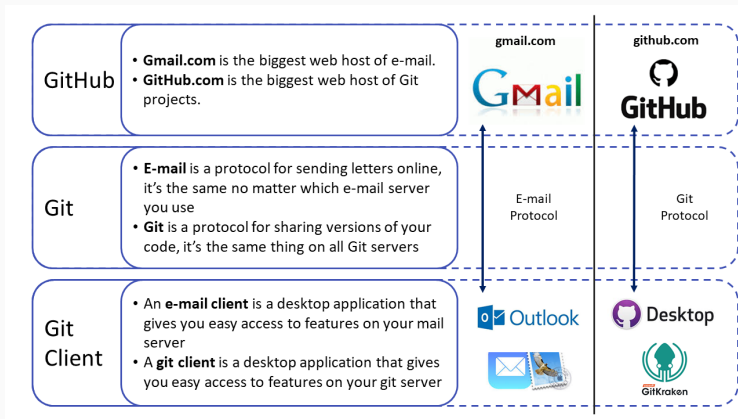
What is Git, GitHub and GitHub Desktop?



What is Git, GitHub and GitHub Desktop?



What is Git, GitHub and GitHub Desktop?



What will we learn?

In **An intro to Git and GitHub - Contributor Role** you will learn to:

- Explore the history of a project folder in GitHub and see what different team members are currently working on
- Download a project folder from GitHub so you can work on it
- Create a space in the project folder where you can make your edits
- Make edits and share those versions with your team. When you are ready, request that your edits are included in the main version

Three Git concepts needed to do this:

- Clone
- Commit
- Branch

Code free training!

No code today!

We will not work with code today.

Code tends to distract people if, for example, they see a command they do not understand.

Instead we will work with lyrics in .txt files that works exactly the same as code files in Git.

How to browse GitHub.Com

Your project folder is called a **repository** in Git, often **repo** for short. When you enter github.com/kbjarkefur/lyrics you get to what we will call the **repository landing page**.

Go from **GitHub.com** to **repo**

1. From anywhere on github.com click the *octocat* icon in the top left corner.
2. In the menu to your left you see the repositories you are invited to
3. Click any repo to get to the landing page of that repo.

Go from **repo** to **landing page**

1. Click the repo name in [kbjarkefur/lyrics](https://github.com/kbjarkefur/lyrics) at the top of any page within the repo

Clone

What is cloning?

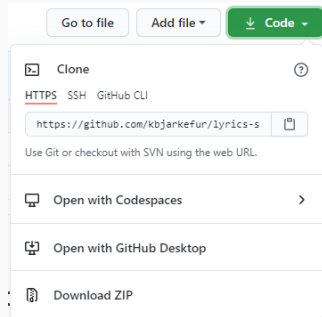
Cloning is similar to downloading a **repository** to your computer.

The difference between cloning and downloading is that **when Git clones a repository it remembers where you downloaded it from**. This is necessary so that Git knows where send any edits you make to the files when sharing them with your team.

How do I clone a repository from GitHub?

How to clone a repository:

1. Go to the **landing page** of github.com/kbjarkefur/lyrics
2. Click the green *Code* button (see image)
3. Click *Open with GitHub Desktop*
4. Select where on your computer to clone the repository.
Do **NOT** clone in a shared folder, like a network drive or in DropBox. Create a *GitHub* folder in non-synced location and clone there.



Explore the clone!

Compare the files and folders you cloned to your computer with those in the repository on github.com/kbjarkefur/lyrics

Collaboration on a Repository

In order to collaborate on a repository we need to introduce two topics:

Commits

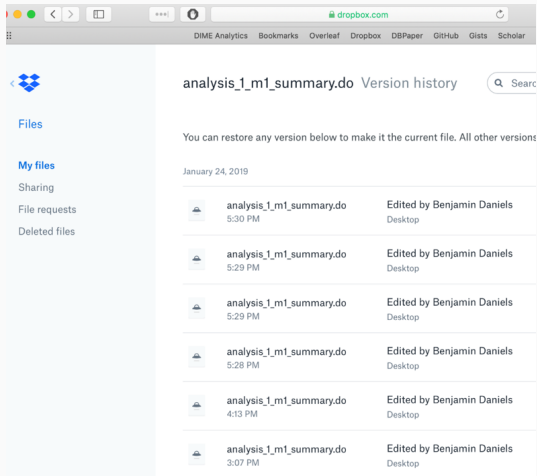
Branches

Commit

What is a version in version control?

First look at how version control works in another software you might be familiar with.

In DropBox each saved version of a file is saved to the version history. This is the only way to do it automatically, but are all these versions meaningful differences?



The screenshot shows the Dropbox web interface. On the left is a sidebar with the Dropbox logo and navigation links: Files, My files, Sharing, File requests, and Deleted files. The main content area is titled 'analysis_1_m1_summary.do Version history' and includes a search bar. Below the title, a message states: 'You can restore any version below to make it the current file. All other versions will be deleted.' The date 'January 24, 2019' is displayed. A table lists six versions of the file, all edited by Benjamin Daniels from the Desktop. The versions are listed in descending order of time.

Icon	File Name	Edited by	Location
	analysis_1_m1_summary.do	Benjamin Daniels	Desktop
	analysis_1_m1_summary.do	Benjamin Daniels	Desktop
	analysis_1_m1_summary.do	Benjamin Daniels	Desktop
	analysis_1_m1_summary.do	Benjamin Daniels	Desktop
	analysis_1_m1_summary.do	Benjamin Daniels	Desktop
	analysis_1_m1_summary.do	Benjamin Daniels	Desktop

What is a commit?

Instead of having a list of each saved version of a file, in Git we use **commits to indicate what is each meaningful difference between two versions of our project folder.**

Each commit is a snap shot of all files in the project folder, and lists how that snap shot differ from the previous snap shot (the previous commit).

Each commit has a time stamp and tracks who did the commit. This is very similar to the *YYMMDD_docname_INITIALS.doc* solution to the *Final.doc* problem.

How to make a commit

We need to introduce *branches* before we can all commit to the same repository, so for now, let me show you how to make a commit:

1. I add a new lyrics .txt file in the clone
2. I use GitHub desktop to commit the new file to the repository
3. Can you see the new file on your computer?
4. Can you see it if you sync in GitHub Desktop?

Now when we know what a commit is, we can start exploring how the github.com/kbjarkefur/lyrics repository was created.

We will see a list of commits, that at first sight is similar to the the version history in DropBox, but **in Git the version list is more meaningful, as it is a list of only meaningful differences.**

- github.com/kbjarkefur/lyrics/commits
- This list can also be found in GitHub Desktop in the *History* tab

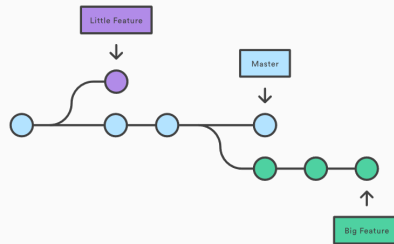
Branch

Introducing branches

Branches is the "killer feature" of Git. This is where Git becomes really powerful as a collaboration tool and not just as version control.

Branches allows you to **create a copy of the code where you can experiment**, if you like the result, **you can very easily merge your experiment with the main version of your code.**

This non-linear version control is much more similar to how we actually work than the strictly linear version control in, for example, DropBox



Looking at branches

One more way to explore the repository

- github.com/kbjarkefur/lyrics/commits <- Linear progression
- github.com/kbjarkefur/lyrics/network <- Non-linear progression

Exploring branches

- You can change branch in `/commits`. What happens when you change branch?
- Go to the landing page, what happens if you change branch here?
- Which version is in the clone on your computer? They are all actually in your clone, but only one is shown - **checked out** - at the time
- What happens to the content of the folder on your computer when you check out another branch in GitHub Desktop?

Working with branches

A typical Git work flow involves multiple branches and there are tools in GitHub to makes that work flow easy, but that is not within today's scope. Although, what you should know after this training is only how to create your own branch and how to commit to it.

Create a branch:

- Go to github.com/kbjarkefur/lyrics and click the button where it says *Branch: main*.
- Write your name in the field and click *Create branch: your_name*. Make sure it says *from 'main'*.
- See how the button now says *Branch: your_name*
- Go to github.com/kbjarkefur/lyrics/network to check that your branch is there.

Combining Commit & Branch

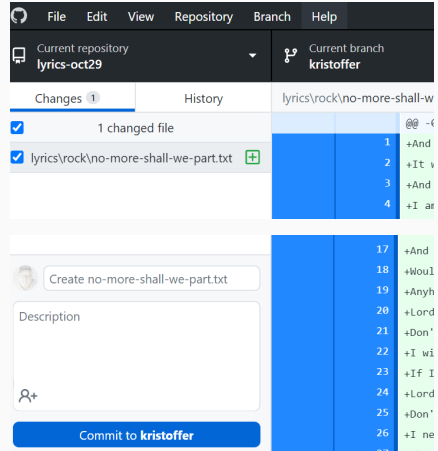
Now it is time to collaborate

Now it is time for you to collaborate:

1. Make sure your branch is checked out in GitHub Desktop.
2. Open a text editor. It could be *Notepad* if you are using Windows, *TextEdit* if you are using Mac, or any other code editor like *Atom*, or *Notepad++* etc.
3. Google the lyrics of your favorite song, and copy the lyrics to a new file in the text editor you just opened
4. Save the lyrics in your local clone according to these instructions:
 - Save the file in *.txt* format (Especially Mac users, this is not always the default)
 - Name the file after the title of the song
 - Save it in the appropriate genre folder (create a new genre folder if needed)

Do your first commit

1. Open the changes tab in GitHub Desktop
2. GitHub Desktop tracks your clone and has noticed that you changed something in it
3. Then you need to do the three steps required to commit a file to the repository:
 - 3.1 Make sure the file you want to add is checked
 - 3.2 Write a commit message
 - 3.3 Click *Commit to your_name* - check out your branch if it says *main*
 - 3.4 Click the sync button



Check your commit on GitHub:

- Go to github.com/kbjarkefur/lyrics/network
 - Can you find your commit?
- Go to github.com/kbjarkefur/lyrics/commits
 - Can you find your commit?
 - If you cannot see your commit, make sure that you are looking in the correct branch

Pull Requests

A feature related to branches is a **pull request**. When the edits you have done in your branch are ready to be merged with the main version of the code, you make a pull request, i.e. you are requesting that your edits are pulled into the main branch.

A pull request can be made either by the person that edited the branch or the repo maintainer.

It is common in GitHub repositories that only the repo maintainer have access to the main branch, and pull requests are then the only way to contribute to the main branch.

To make a pull request for your branch:

- Go to github.com/kbjarkefur/lyrics/pulls and click *New pull request*
- Make sure that the *main* branch is selected as the *base:* branch, and then select your branch as the *compare:* branch
- Scroll down to check the edits you are requesting to be pulled in to the main branch. If it looks ok, then click *Create pull request*
- Then you have the chance to add more instructions if you want, then click *Create pull request* again

Merge your contribution

Can you see your lyrics file in the main branch now? Why not?

Your contribution will not be included until the branch is merged. We have more trainings on the details of merging branches but for now this is all you need to know:

- Always have someone to review your PR before merging it
- Always delete your branch after it is merged - you can always recreate a new branch with the same name if you want

What have we learned?

In **An intro to Git and GitHub - Contributor Role** you have learned to:

- Explore the history of a project folder in GitHub and see what different team members are currently working on
- Download a project folder from GitHub so you can work on it
- Create a space in the project folder where you can make your edits
- Make edits and share those versions with your team. When you are ready, request that your edits are included in the main version

What have we learned?

In **An intro to Git and GitHub - Contributor Role** you have learned to:

- Explore the history of **repository** and see what different team members are currently working on
- **Clone** a **repository** so you can work on it
- Create a **branch** in the **repository** where you can make your edits
- Make edits and share **commits** with your team. When you are ready, make a **pull request** to the **main branch**

Magic Trick

Magic Trick

All great training ends with a magic trick! This is a pedagogical magic trick, don't worry, just bear with me!

I need a volunteer for a simple task from the audience!

Next steps for the research team

Three common GitHub Roles

Observer

- Browse code in GitHub
- Provide feedback through GitHub
- **Who?** Typically a PI that does not code

Contributor

- Contribute to code in GitHub
- Understand and follow instructions from a Repo Maintainer
- **Who?** Typically an RA, or a PI that codes

Repo Maintainer

- Make sure that best practices and standards are followed in the repository
- Guide new contributors
- **Who?** Typically the most senior RA. Takes too much time for a PI.

Next steps for the research team

Before adopting Git the research team should discuss the following things:

- Who will be repo maintainer
- Agree on a work flow
- Public/private repository
- External collaborators
- Where to put data and where to put code
- Request a World Bank repo to be set up

Useful links

- All DIME Analytics GitHub trainings: <https://osf.io/e54gy/>
- Other DIME Analytics GitHub resources:
<https://github.com/worldbank/dime-github-trainings>. For example:
 - DIME Analytics GitHub Templates (for example .gitignore):
<https://github.com/worldbank/dime-github-trainings/tree/master/GitHub-resources/DIME-GitHub-Templates>
 - DIME Analytics GitHub Roles:
<https://github.com/worldbank/dime-github-trainings/blob/master/GitHub-resources/DIME-GitHub-Roles/DIME-GitHub-roles.md>
- Markdown cheat sheet (how to format text on GitHub.com):
<https://www.markdownguide.org/cheat-sheet/>
- DIME GitHub Account admin info and instructions:
<https://github.com/dime-worldbank/dime-account-admin>

Most recent commit at the point of compiling this \LaTeX Beamer presentation:

`https://github.com/worldbank/dime-github-trainings/commit/9b9db49773c778d5e139cb2bc69adee437d85c6b`

For instructions on how to include URLs to most recent commit used to generate document in Git, see: <https://gist.github.com/kbjarkefur/88820e5a5365b3f707b6b20aee57cf8a>