

# An intro to Git and GitHub - Observer Role

---

DIME Analytics

November 13, 2019

DECIE - The World Bank

## Before the session starts:

1. Do you have a GitHub.com account? If not, got to <https://github.com/join> and sign up
2. Open [github.com/kbjarkefur/country-facts](https://github.com/kbjarkefur/country-facts) in a browser

# An intro to Git and GitHub - Observer Role

---

DIME Analytics

November 13, 2019

DECIE - The World Bank

# Three common GitHub roles

The objective of this training is to make you able to take the role of an **Observer**. See DIME Analytics GitHub Roles for full details (link in end of presentation).

## Observer

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

## Contributor

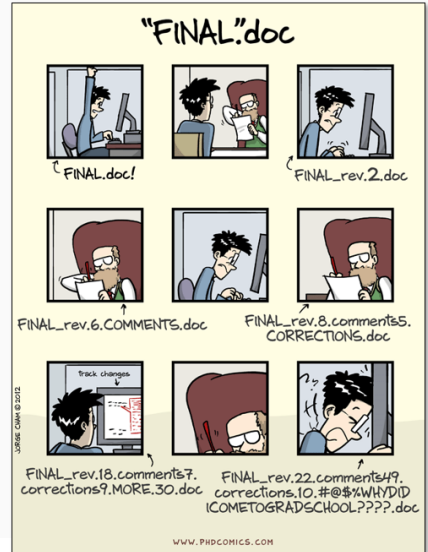
- Contribute to code in GitHub
- Understand and follow instructions from a Repo Maintainer
- **Who?** Typically an RA, or a PI that codes

## 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

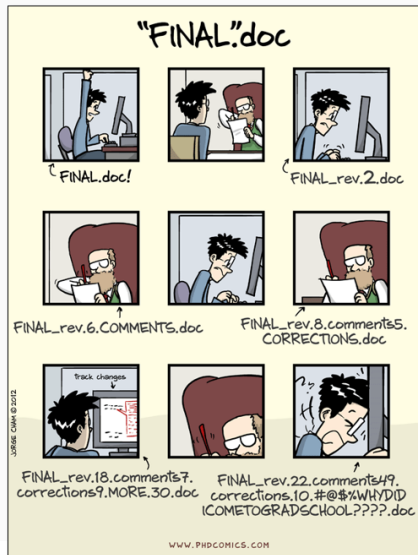


# What is Git used for?

- Git solves the *Final.doc* problem
- Common solution to the *Final.doc* problem.

Name all your docs like

*YYMMDD\_docname\_INITIALS.doc*



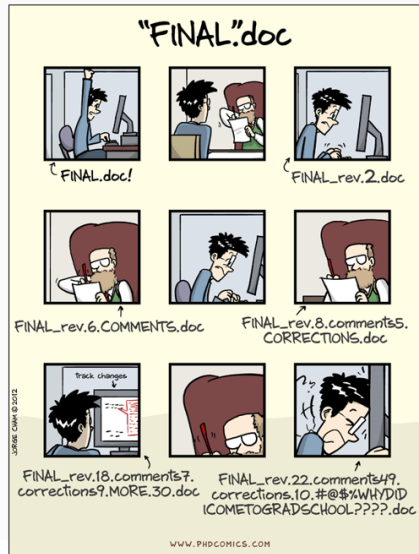
# What is Git used for?

- 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...

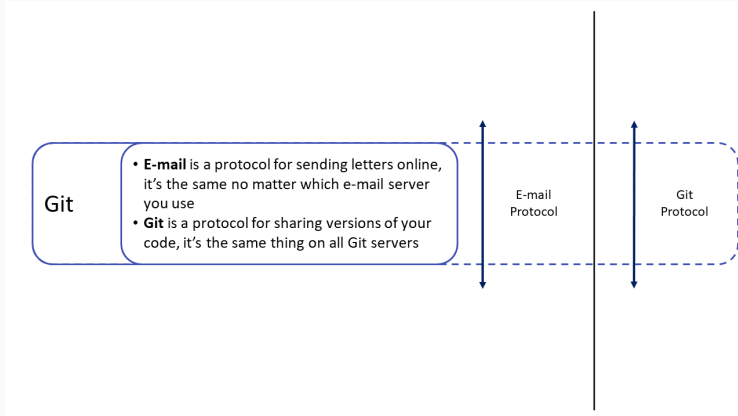




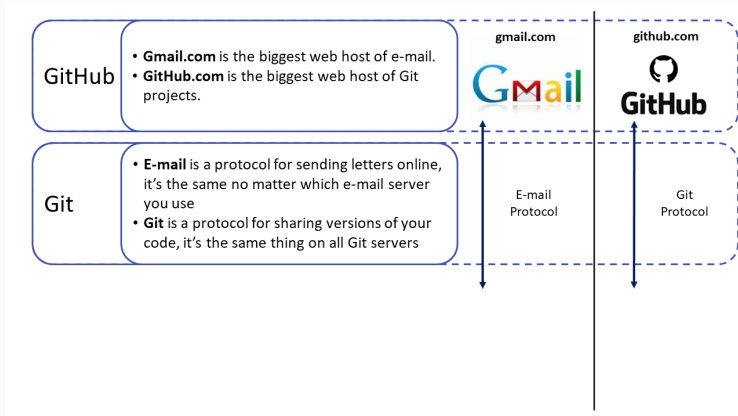
# Git vs. GitHub vs. GitHub Desktop

---

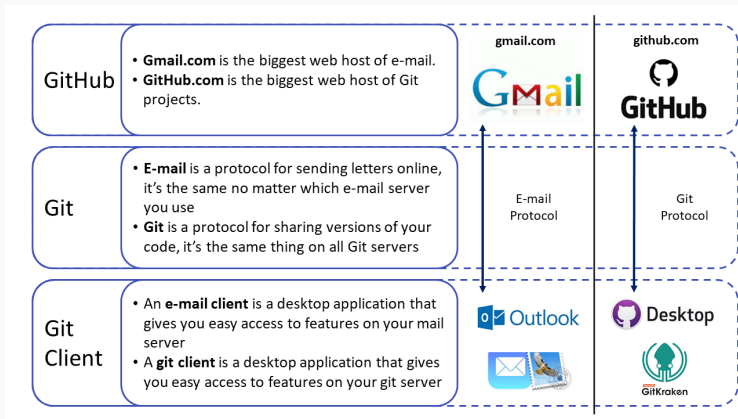
# What is Git, GitHub and GitHub Desktop?



# What is Git, GitHub and GitHub Desktop?



# What is Git, GitHub and GitHub Desktop?



## Exploring a Repository

---

What we will cover today:

- Browse code on GitHub
  - **Commits** - very briefly
  - **Branches** - very briefly
- Provide feedback through GitHub
  - **Issues**
- Discuss some basic GitHub best practices

# How to browse GitHub.com

Your project folder is called a **repository** in Git. We will call the page you land on when you enter [github.com/kbjarkefur/country-facts](https://github.com/kbjarkefur/country-facts) the **main page**.

## GitHub.com -> Repo

1. From anywhere on [github.com](https://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 main page of that repo.

## Repo -> Main page

1. Click the repo name in [kbjarkefur/country-facts](https://github.com/kbjarkefur/country-facts) at the top of any page within the repo
2. Click the tab that says **Code** below the repo name at any page in your repo

## No focus on code today!

Code tends to distract people if, for example, they see a command they do not understand. Instead we will work on a *Country Fact Sheet* in a file called *countries.do*.

It is a .do file, so the content will behave just as code, but there is no code in it.

We will look at some examples that uses actual code once we are familiar with the features we will cover today.

[github.com/kbjarkefur/country-facts/blob/master/facts/countries.do](https://github.com/kbjarkefur/country-facts/blob/master/facts/countries.do)



You are not expected to get a full understanding of **Commits** and **Branches** and how they are used today, that is what the *Contributor training* is for.

But after I have briefly covered these two topics and also introduced **Issues**, we will look at an example that will give you the intuition for what **Commits** and **Branches** are.

# Commits

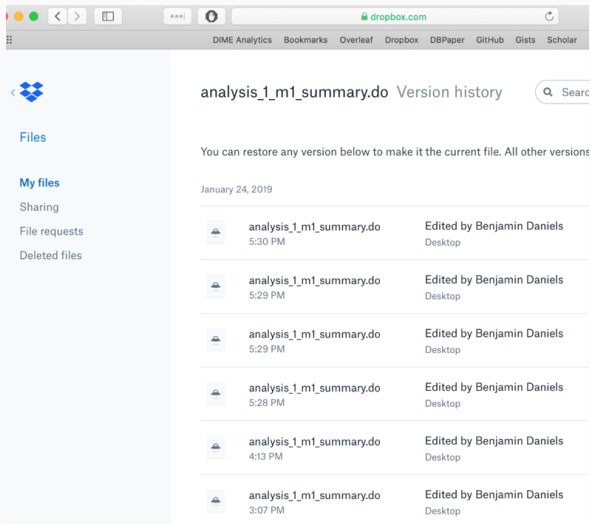
---

# What is a commit? - 12 min

**In DropBox**, each saved version of a file is saved to the version history. Are all these versions meaningful differences?

**In GitHub**, commits are used to indicate what is a meaningful difference.

One meaningful difference can be large or small, can be an edit to one, some or all files at the same time.



The screenshot shows the Dropbox web interface. The left sidebar contains navigation links: Files, My files, Sharing, File requests, and Deleted files. The main content area is titled 'analysis\_1\_m1\_summary.do Version history'. Below the title, it states: 'You can restore any version below to make it the current file. All other versions'. A date separator 'January 24, 2019' is shown. A table lists six versions of the file, all edited by Benjamin Daniels on the Desktop.

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

Now when have a basic understanding of what a *Commit* is, we can start exploring how the [github.com/kbjarkefur/country-facts](https://github.com/kbjarkefur/country-facts) 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/country-facts/commits](https://github.com/kbjarkefur/country-facts/commits)

# Branches

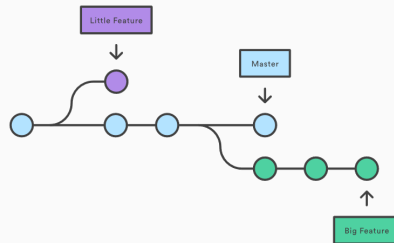
---

# Introducing branches - 16 min

**Branches is the "killer feature" of Git.** This is where Git becomes really powerful as a collaboration tool and 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



## One more way to explore the repository

- [github.com/kbjarkefur/country-facts/commits](https://github.com/kbjarkefur/country-facts/commits) <- Linear progression
- [github.com/kbjarkefur/country-facts/network](https://github.com/kbjarkefur/country-facts/network) <- Non-linear progression

# Issues

---



## Issues are feedback - 20 min

**Issues are optional** in Git/GitHub, you do not have to use them

**Issues** in GitHub is where you discuss **feedback**. It is like a **task management tool integrated with your code**

**Issues and edits to the code can be linked** so that anyone with access to the repo, can **link a discussion about an edit to the code to the Commit with that edit**

The link can also goes the other way, so **if Commits and Issues are linked**, then anyone in the future with access to the repo, can **link edits to the code in a Commit with the discussions that lead up to that edit**

Resolved **Issues are archived and searchable on GitHub** as long as the repo exists

## Issues are feedback

For each *Issue* a **discussion thread** will be opened where anyone with access to the repo can contribute to the discussion

Issues can be **assigned** to GitHub users similarly to a task management tool

Specific **lines of code can be referenced and previewed**, so the discussion can be tied to specific lines of code, making it clear to everyone what is being discussed

In addition to reference specific lines of code, there are tools to **label Issues, cross reference Issues, tag people** relevant to the discussion etc. so that collectively the *Issues* becomes a the **documentation** or the meta-information on how the code for your project was developed

An example that ties commits, branches and issues together:

Go to the issues tab, then click the issue with the title **Capital of India is incorrect**.

Or type this URL in your browser:

[github.com/kbjarkefur/country-facts/issues/1](https://github.com/kbjarkefur/country-facts/issues/1)

## Practice issues

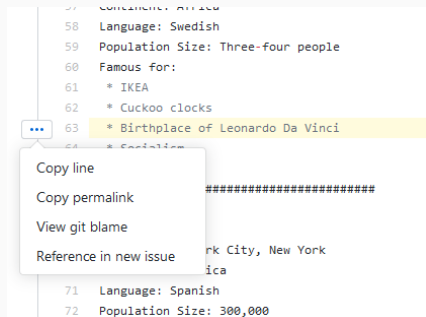
---

# How to create an issue - 27 min

Time to create issues! Go to:

[github.com/kbjarkefur/country-facts/blob/master/facts/countries.do](https://github.com/kbjarkefur/country-facts/blob/master/facts/countries.do)

1. Find an incorrect fact to create an issue for
2. Click the line number of the line with the incorrect fact you want corrected
3. Click the three-dot menu that appears
4. Click *Reference in new issue*
5. Give the issue a title, describe what you want fixed, use the *preview* as needed, and then click *Submit new issue*



1. Assign me (*kbjarkefur*) or someone else to the *Issue* you just created
2. See if you find a label that fit your *Issue*
3. Write a comment on an *Issue* someone else created
4. Click the **Subscribe** button on an issue that you have not interacted with. This will give you notification on activities in this issue.

# Multi-line code reference

How to reference multiple lines of code:

## Using menus:

1. Click the line number of the first line you want to reference
2. Before clicking the three-dot menu that appears, hold shift and click the last line you want to reference
3. Then click the three-dot menu and click *Reference in new issue*

## Manually:

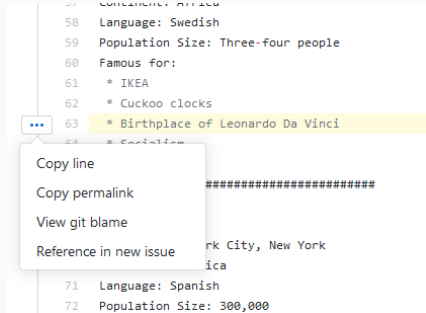
1. Create a one line reference and edit the link from: <https://github.com/kbjarkefur/country-facts/blob/random-sha/facts/countries.do#L18>
2. Modify the end of the URL to:  
<https://github.com/kbjarkefur/country-facts/blob/random-sha/facts/countries.do#L18-L22>

# Multiple code references

How to reference multiple lines of code:

## No way to do this in menus

1. When you click *Reference in new issue* GitHub copies a permalink to the code line and put that in an issue
2. In the menu in the three-dot menu you can instead click *Copy permalink*. This creates the same link as when clicking *Reference in new issue*.
3. Create an issue and copy permalinks to all code lines that you want to reference and copy them to the issue





**Where are files when using  
Git/GitHub?**

---

Your project is now using GitHub, where should I look for:

- code?
- data?
- outputs?

# Where should I look for code?

Where should I look for **code**?

- Code is now in Git/GitHub.
  - It is error prone to also keep it on DropBox
  - No problem to have archived copies of code on DropBox, just don't do work on them
- How to access code:
  - **Browser:** Go to the URL of your repository and explore code using your browser
  - **Download for collaboration:** Clone the repository. It is covered in the collaborator training.

# Where should I look for data?

Where should I look for **data**?

- Data is where it was before (ex. DropBox).
  - Github is secure, but not secure enough for project data
  - Most data formats is not suited for GitHub and makes GitHub slow
- How to access data:
  - Same as before.
  - Code reads data from DropBox, and reads code from the clone.

# Where should I look for outputs?

Where should I look for **outputs**?

- Outputs can go both on Git/GitHub and on DropBox. It is up to the project team!
- GitHub/Git:
  - **Pro:** Tables in .txt/.tex and other raw text formats can be tracked in Git and differences between version will be displayed
  - **Con:** Outputs are not available on DropBox without an extra step
- DropBox
  - **Pro:** Outputs become available immediately for everyone
  - **Con:** If two branches creates different outputs, they will overwrite each other's output, with no documentation on which branch the output came from

**Other tools good for management**

---

Have you ever looked at code you or someone in your project wrote asking yourself these questions:

- Who wrote this line of code?
- When did someone write this line of code?
- Why did someone write this line of code?

Git has a feature for this "*tactfully*" called **Blame**.

- [github.com/kbjarkefur/country-facts/blame/master/facts/countries.do](https://github.com/kbjarkefur/country-facts/blame/master/facts/countries.do)

## Useful links:

- DIME Analytics GitHub Templates (for example .gitignore):  
<https://github.com/worldbank/DIMEwiki/tree/master/Topics/GitHub>
- DIME Analytics GitHub Roles:  
<https://paper.dropbox.com/doc/GitHub-Roles-IXdy2rnRN0Nt917h0nkI7>
- Markdown cheat sheet (how to format text on GitHub.com):  
<https://www.markdownguide.org/cheat-sheet/>