

Using Git and Github on KLC

Fellows Workshop

Summer 2021

Northwestern | Kellogg

The bottom right corner of the slide features a decorative graphic composed of several overlapping triangles. The triangles are in various shades of purple, ranging from a very light lavender to a deep, dark indigo. The shapes are arranged in a way that creates a sense of depth and modern design.

Version Control without Git

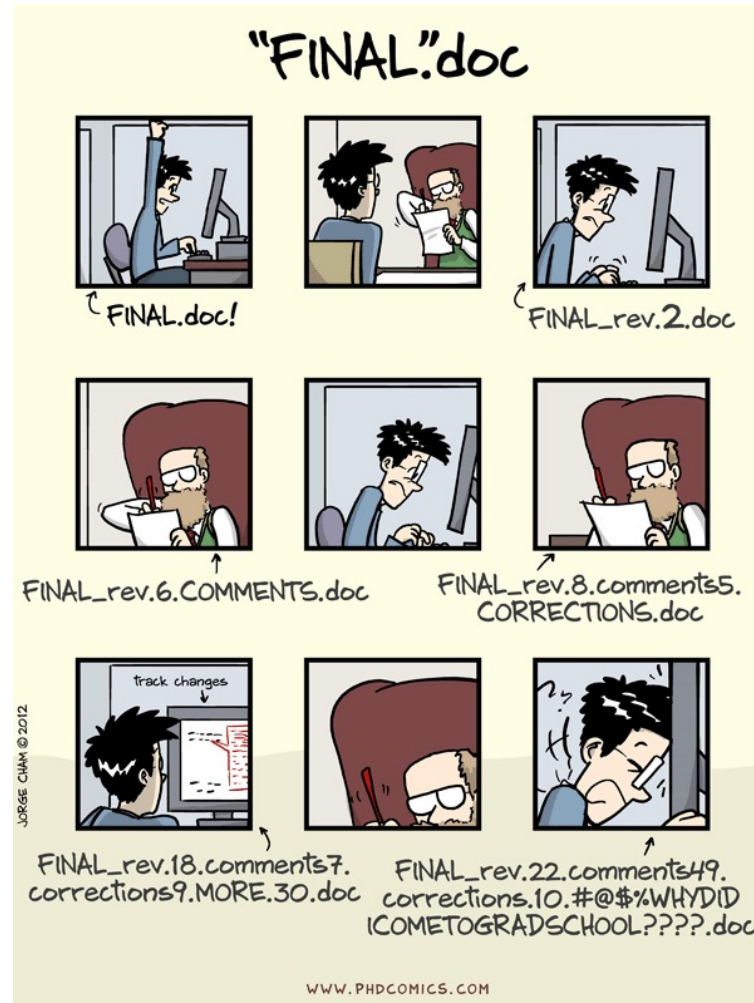


Image taken from: http://phdcomics.com/comics/archive_print.php?comid=1531

Today we will Cover

- The difference between Git and Github
- Using Git locally as a version control tool on KLC
- Syncing your local Git directory to a Github repository
- Discussion: Using Git and Github Smartly

Git vs. GitHub

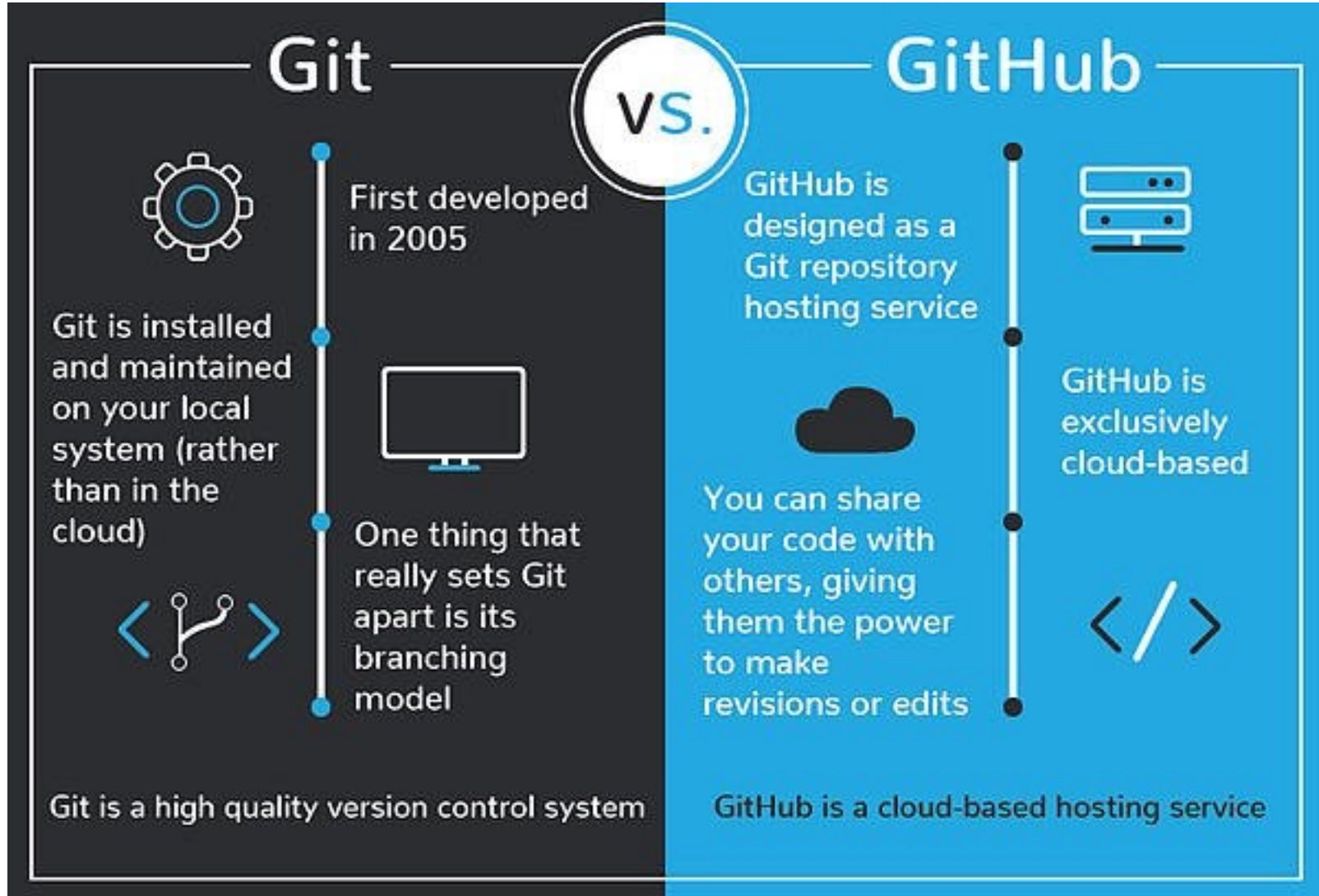


Image taken from: <https://blog.devmountain.com/git-vs-github-whats-the-difference/>

Git and Github Overview

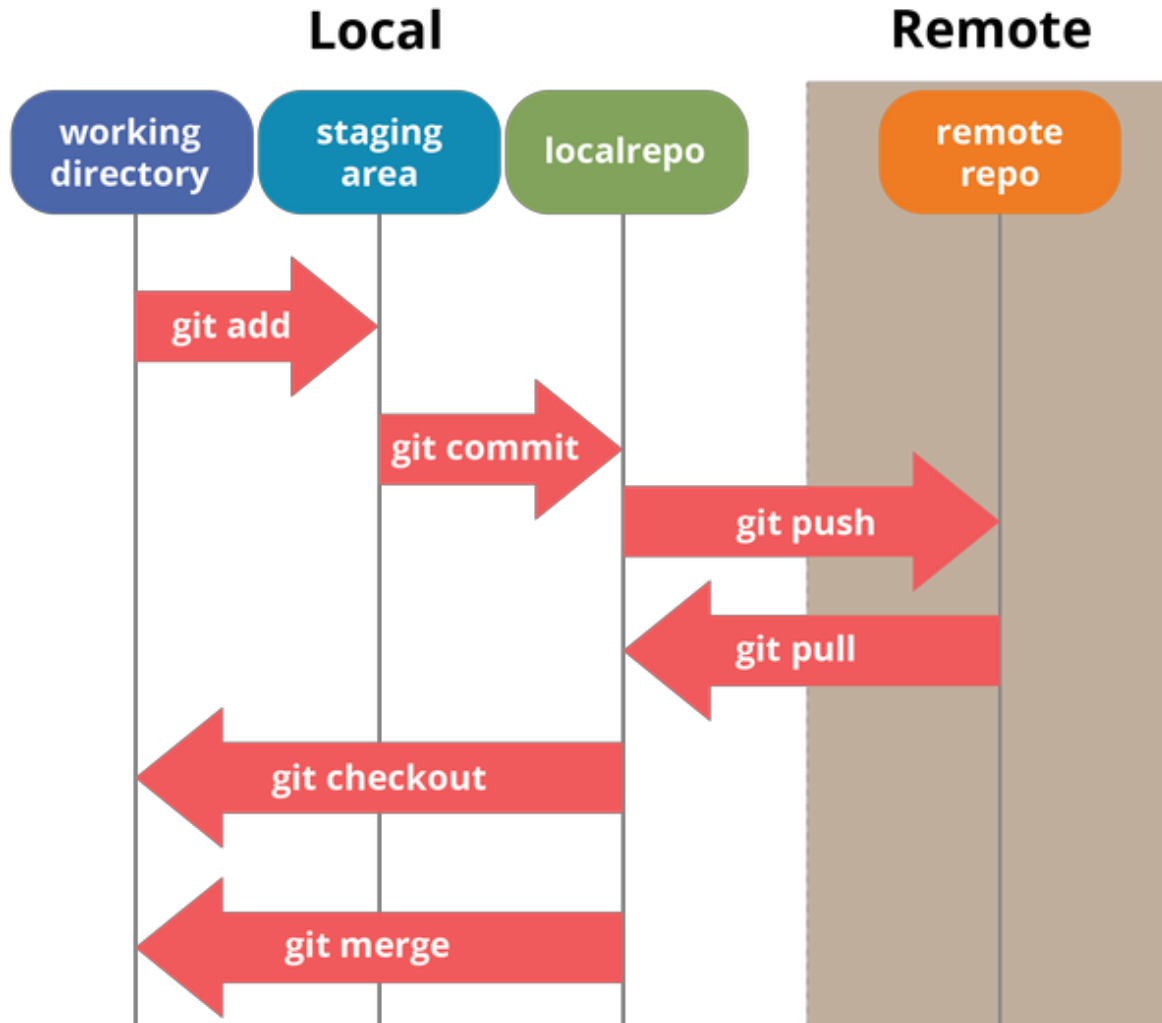


Image taken from: <https://dev.to/mollynem/git-github--workflow-fundamentals-5496>

Loading git on KLC

Recall that no modules are preloaded in a new KLC session. You will need to load everything you use.

To see what version of a software package are available type:

```
module load git
```

The first time you load git on KLC, you need to set the username and email it will use when it records your commits. Note that this will have no bearing on your Github login later on.

```
git config --global user.name "John Doe"  
git config --global user.email johndoe@northwestern.edu  
git config --list
```

Resources:

NU IT page: <https://kb.northwestern.edu/page.php?id=78598>

NU IT github: <https://github.com/nuitrcs/gitworkshop>

Cloning a repo and initializing a local repo

A Git **repository** ("repo") is essentially a **container** for your Git code. You can create a Git repo on KLC from scratch or you can download an existing public Github repo and work on the code locally on KLC.

Let's first clone a public Github repo to your home directory:

```
git clone https://github.com/ambreenchaudhri/fellows_git_intro
```

It should download a directory called: `fellows_git_intro`.
Let's create a subfolder in this directory called `practice`.

```
cd fellows_git_intro  
mkdir practice  
cd practice
```

The first step to start version control with Git is to initialize your (empty) directory:

```
git_init
```

Make Changes to your repo

Note that when you initialize a repo a new hidden folder will be saved here:

```
ls -a
```

If you type the following, you notice there is no activity in your Git repo:

```
git status
```

Let's make some basic changes to the Git repo we initialized. For instance, let's start by moving in all the FEC files we cloned from Github.

```
mv ~/fellows_git_intro/fec* ~/fellows_git_intro/practice/
```

Now running `git status` will show untracked changes.

Saving Changes to a local git repo

Since a Git repo consists of two spaces: a “staging area” and the actual “repository”, saving changes requires two steps:

1.) an “add” and 2.) a “commit”.

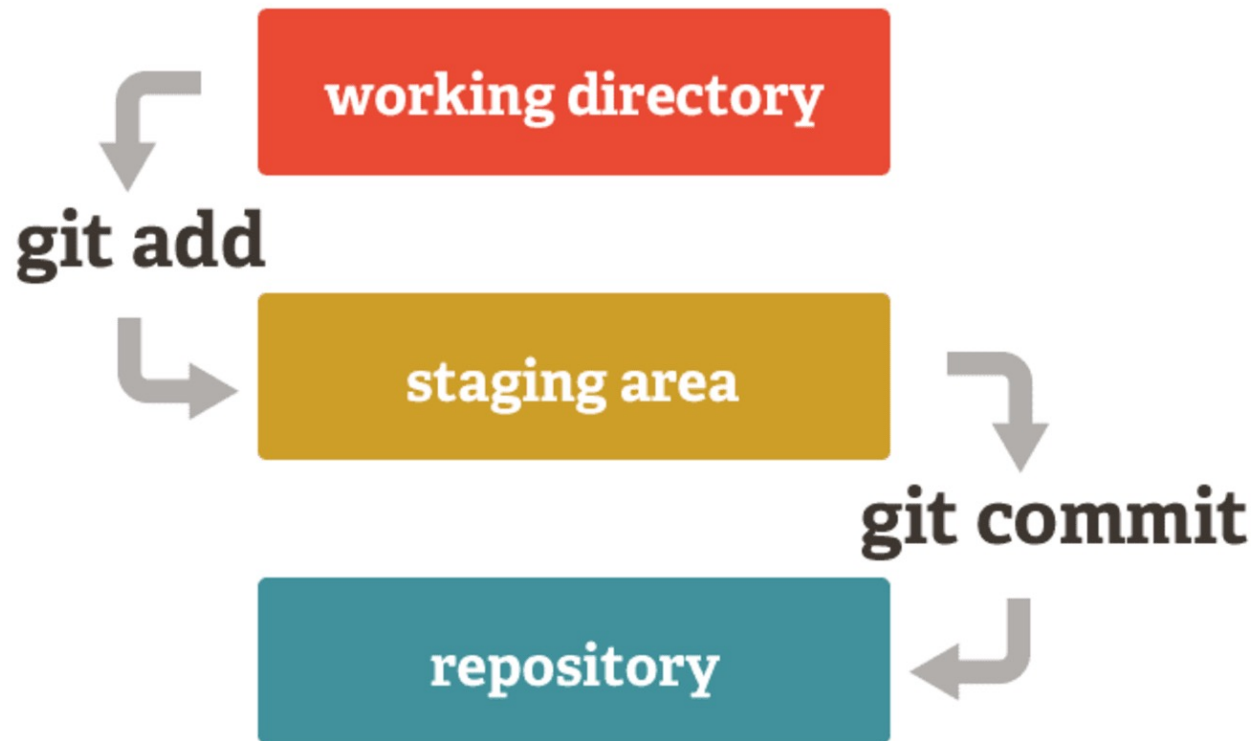


Image taken from:

<https://medium.com/tech-and-the-city/changing-a-super-old-git-commit-history-20346f709ca9>

Saving Changes to a local git repo

We can add the files we moved to the staging area with the following:

```
git add fec_all.sh fec_cron.sh fec_extract.py fec_process.R
```

OR

```
git add .
```

We can move the files from staging area to the local Git `practice` repo with:

```
git commit -m "Add all FEC files to repo"
```

After making these changes, we can retrieve a log:

```
git log
```

Retrieving a Previous Version

If we go back to our `git log`, there is an alpha-numeric sequence associated with each commit. These are commit-hashes.

To retrieve a previous version of a document or set of documents, use the commit-hash:

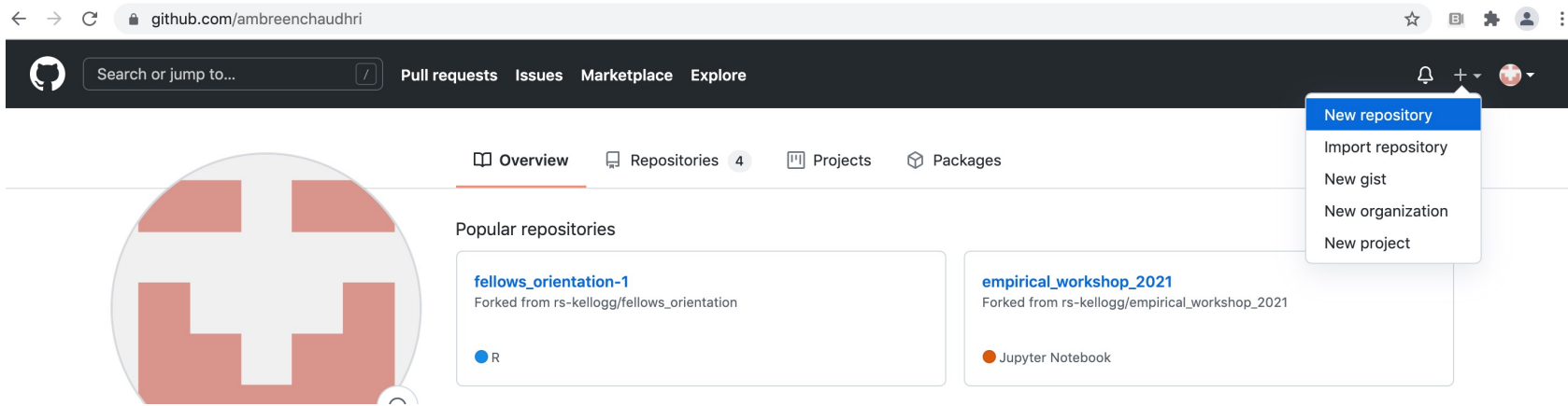
```
git checkout <commit hash>
```

Files in a git repo are saved in **branches**. Branches are essentially timelines on Git. For now, we have been working in the main branch called “master”. You can return to the main timeline with:

```
git checkout master
```

Creating and accessing a Github Repo

We can go to a Github account to create a new private repo to save our local changes.



Follow the steps to create a repo and you will receive a page with instructions to access it.

Accessing a github repo on KLC

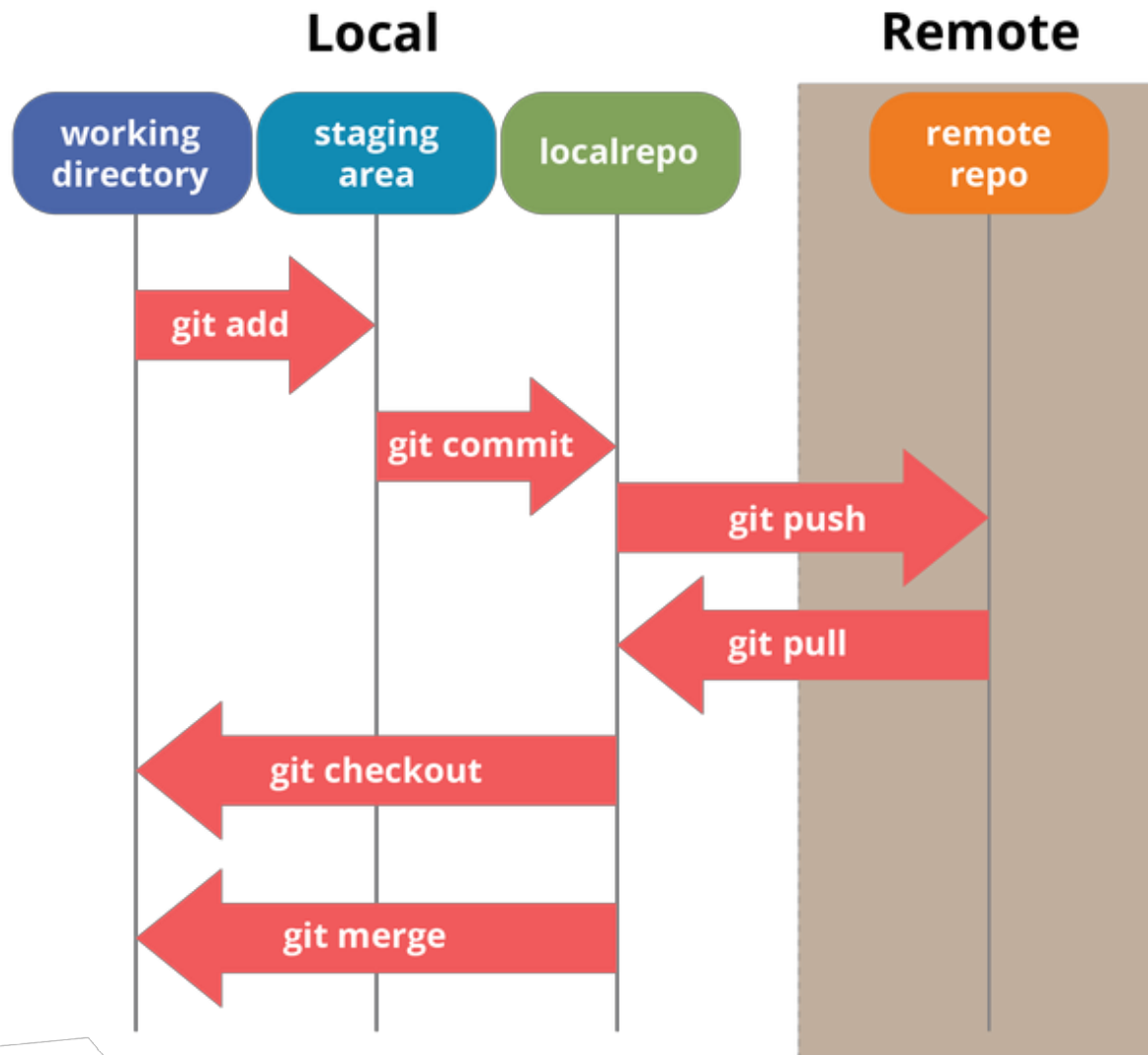
The instructions should list how to add a "remote" to your Github repo, which is equivalent to a label for your Github url.

```
git remote add origin https://github.com/<user>/git_practice.git
```

Note that "origin" is just a generic name. You can use any name you like. The following command will show you the remote that is set.

```
git remote -v
```

Saving Changes to a Github repo



Saving local Git repo to Github

To push the changes in your local repo to Github, do the following:

```
git branch -M main  
git push -u origin main
```

The push request will require you to sign into your Github account on KLC. In order to do so you will need a **personal access token**. You can acquire one by going to:

Settings → Developer Settings → Personal Access Token → Generate New Token

The steps are described here: <https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token>

Saving Github changes to local repo

If you manually add files or make other changes on Github, you can sync those changes to your local git repo on KLC by doing the following:

```
git pull origin main
```


Discussion: Using Git and Github Smartly

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Discussion: Using Git and Github Smartly

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFI	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Git tools are not a substitute for good documentation. You are still responsible for making decisions about whether the code is tested, what version is saved, how descriptive the comments are. Placing bad code in Git/Github is still bad code!

Appendix – Text Editors on KLC

KLC/Quest has a number of text editors available for you to use. The links below provide the basic syntax for other text editors available to you:

Nano: From any terminal session on KLC, type:

```
nano <file_name>
```

<https://www.redhat.com/sysadmin/getting-started-nano>

Emacs: From any terminal session on KLC, type:

```
emacs <file_name>
```

<https://www.redhat.com/sysadmin/beginners-guide-emacs>

Vi: From any terminal session on KLC, type:

```
vi <file_name>
```

<https://www.redhat.com/sysadmin/introduction-vi-editor>

Gedit: From a terminal session with graphics enabled, type:

```
gedit <file_name>
```

<https://www.lifewire.com/gedit-linux-command-unix-command-4097153>

Appendix – Branching on Git

A branch is an independent line of development. You could think of a branch as a timeline of a project that could be created in parallel with the main branch. To see a list of branches a project contains, type:

```
git branch
```

If you'd like to make an experimental branch of a project and start working on it, type the following:

```
git branch experimental  
git checkout experimental
```

The changes you make in a new branch will remain independent of the master branch you are working on. Once you are comfortable with these changes, you can merge them back to the master branch by doing the following:

```
git checkout master  
git merge experimental
```

Appendix – Forking a Github Repo

Much like cloning, forking enables a user to copy all the contents of a Github repository. However, forking a repo also enables a user to suggest and possibly implement changes to the original repository with the creator's approval. Forking is a useful tool to collaborate with others to improve open-source code.

There is no command line tool in the Git module on KLC to fork a repository. Instead, navigate to the main page of a repository. On the right side, select the “Fork” button. The selected repository will be copied to your list of Github repos.

The screenshot shows the GitHub interface for the repository 'ambreenchaudhri/fellows_git_intro'. The 'Fork' button is highlighted with a red box, and a red arrow points to it from the text below. The repository page includes a file list with the following details:

File Name	Commit Message	Commit Hash	Time Ago
fec_all.sh	Add files via upload	546ddd5	6 days ago
fec_cron.sh	Add files via upload		6 days ago
fec_extract.py	Rename 1_fec_extract.py to fec_extract.py		5 days ago
fec_process.R	Update fec_process.R		5 days ago