Set up your Conda Environment

```
# do these steps if not already done:
module load python-miniconda3/4.12.0
conda create -y –n edgar python
source activate edgar
edgar2data --help

cd
git clone https://github.com/rs-kellogg/edgar2data.git
cd edgar2data
pip install .
edgar2data --help

cd ~/edgar2data
conda install pytest
pytest

cd ~/workshop_2022/
```

Using Git and Github on KLC

Fellows Workshop

Summer 2022

Northwestern | Kellogg

Version Control without Git

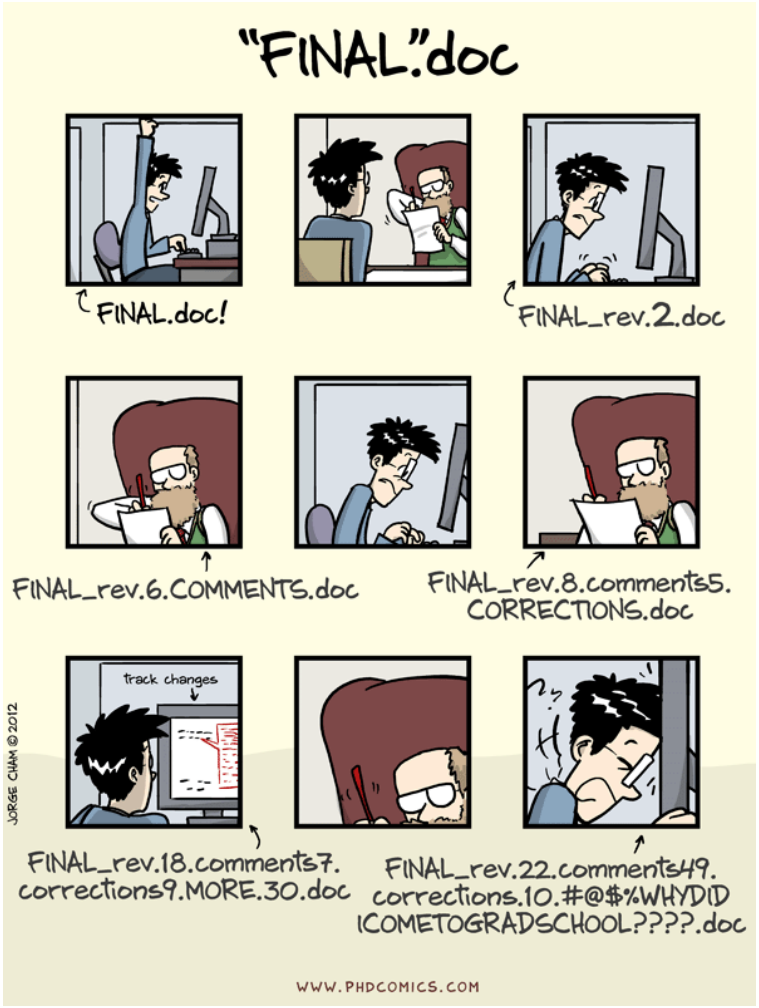


Image taken from: http://phdcomics.com/comics/archive_print.php?comicid=1531

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

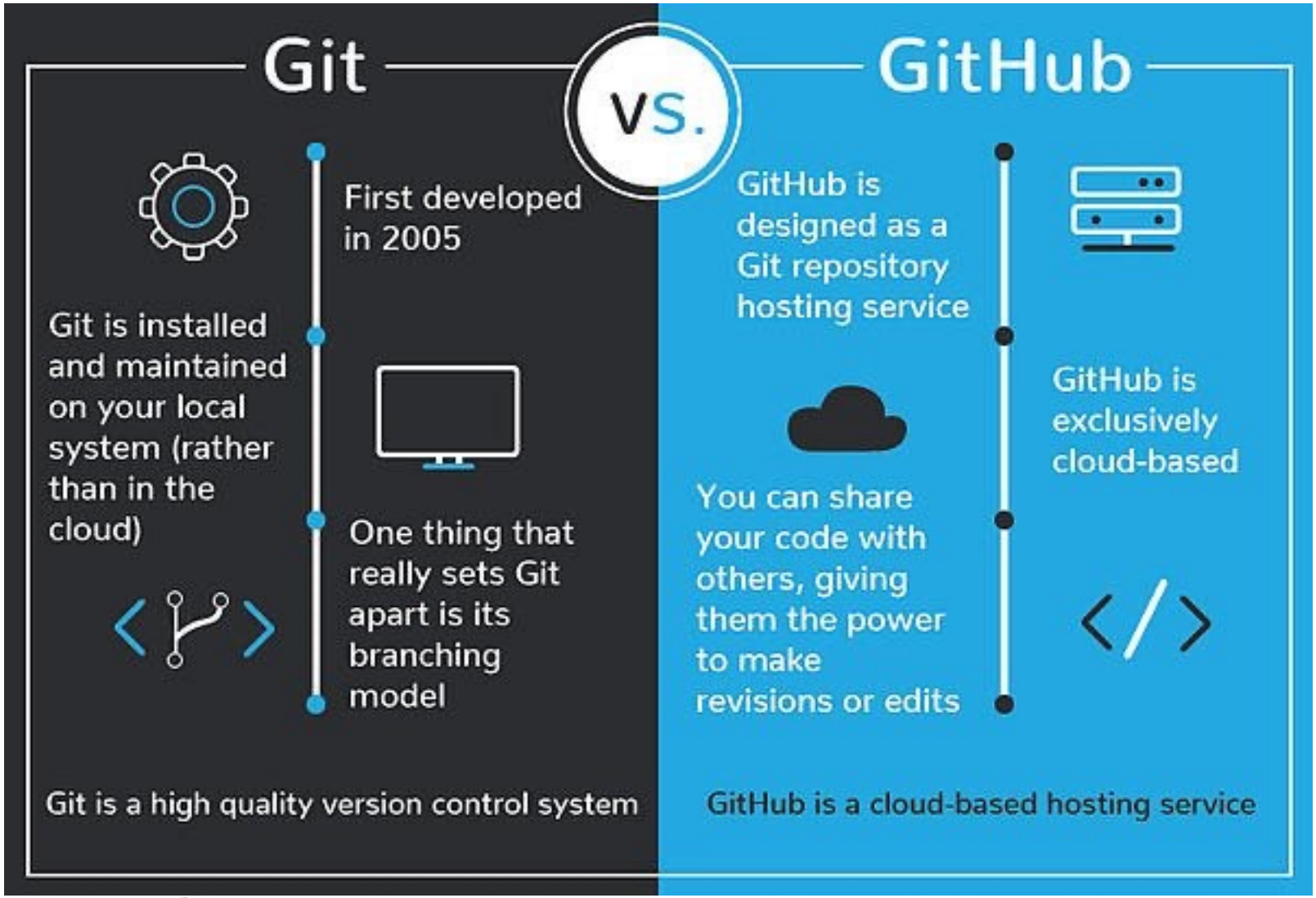


Image taken from: https://blog.devmountain.com/git-vs-github-whats-the-difference/

Git and Github Overview

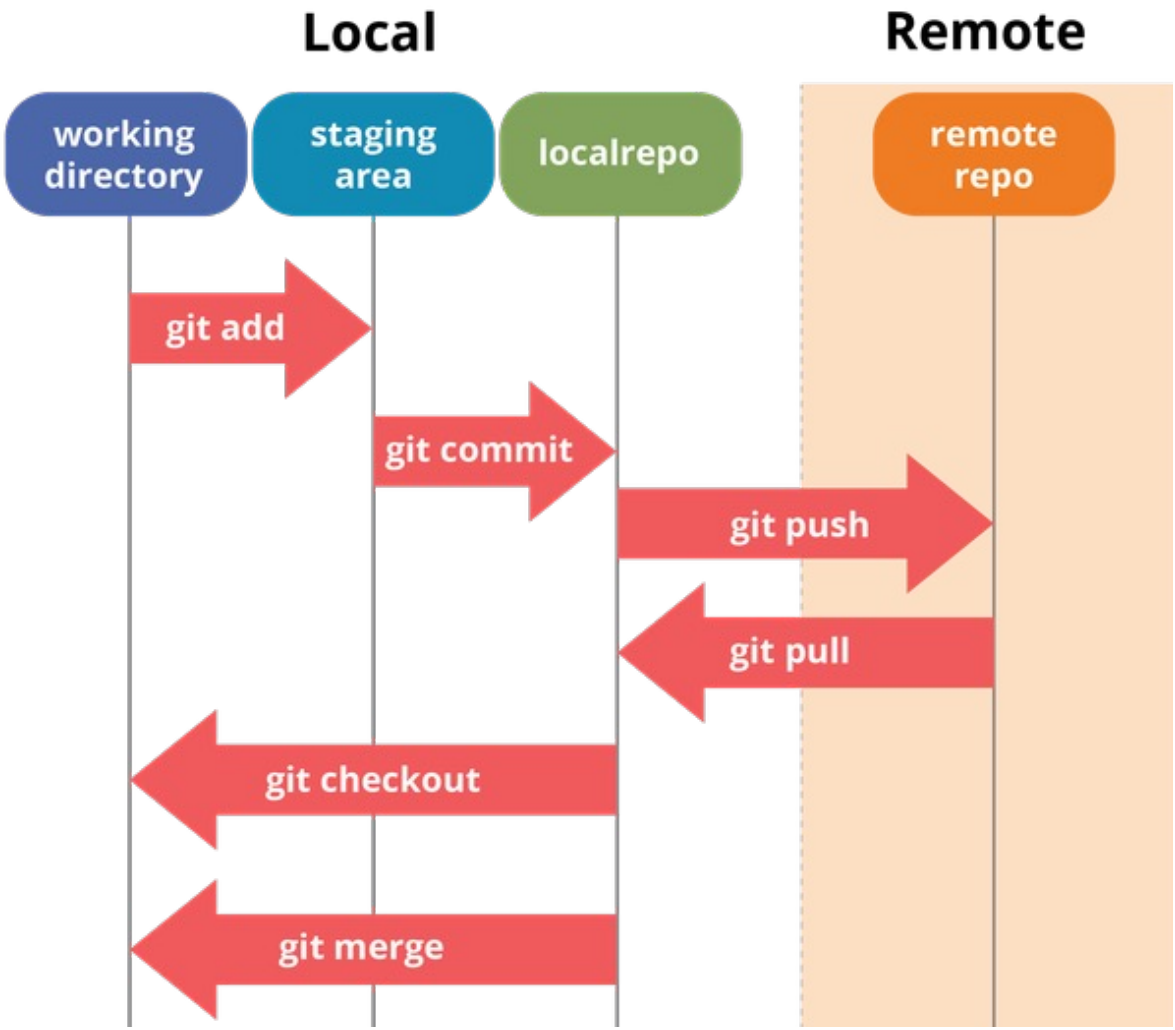


Image taken from: https://dev.to/mollynem/git-github--workflow-fundamentals-5496

Loading git on KLC

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

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

Cloning a repo and initializing a local repo

A Git **repository** (”repo”) is a **container** for your files along with a history of all changes made to them. You can create a Git repo on KLC from scratch or you can download an existing public repository from github:

```
cd
git clone https://github.com/rs-kellogg/workshop_2022.git
```

It should create a new directory called: `workshop_2022.`
Now let’s create a new repository in a directory called `practice.`

```
cd
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.

```
cp -R ~/workshop_2022/Session3_Test/python/ .
```

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 requies two steps:
1.) an “add” and 2.) a “commit”.

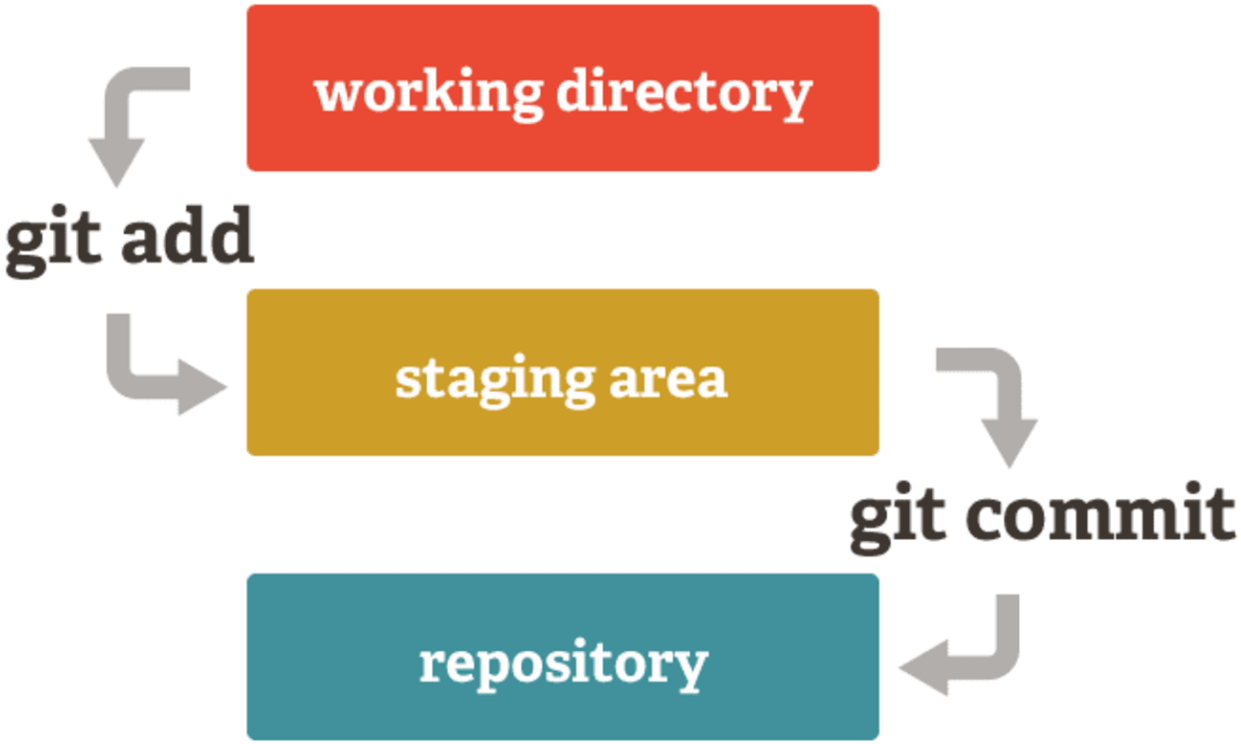


Image taken from:
https://medium.com/tech-and-the-city/changing-a-super-old-git-commit-history-20346f709ca9

Saving Changes to a local git repo

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

```
git add .
```

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

```
git commit -m "initial commit"
```

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:

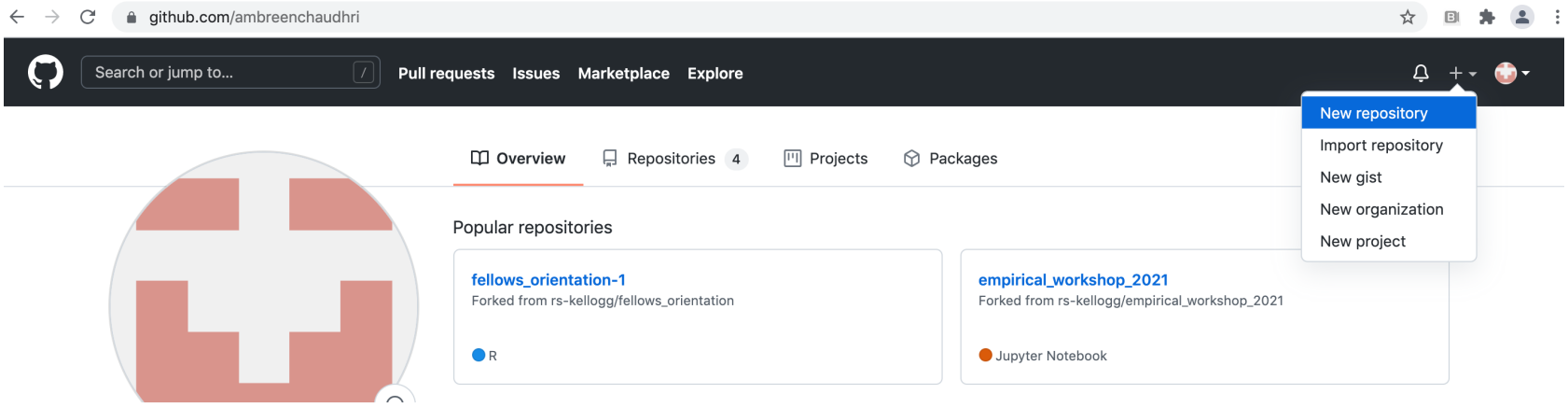
```
git checkout <commit hash>
```

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

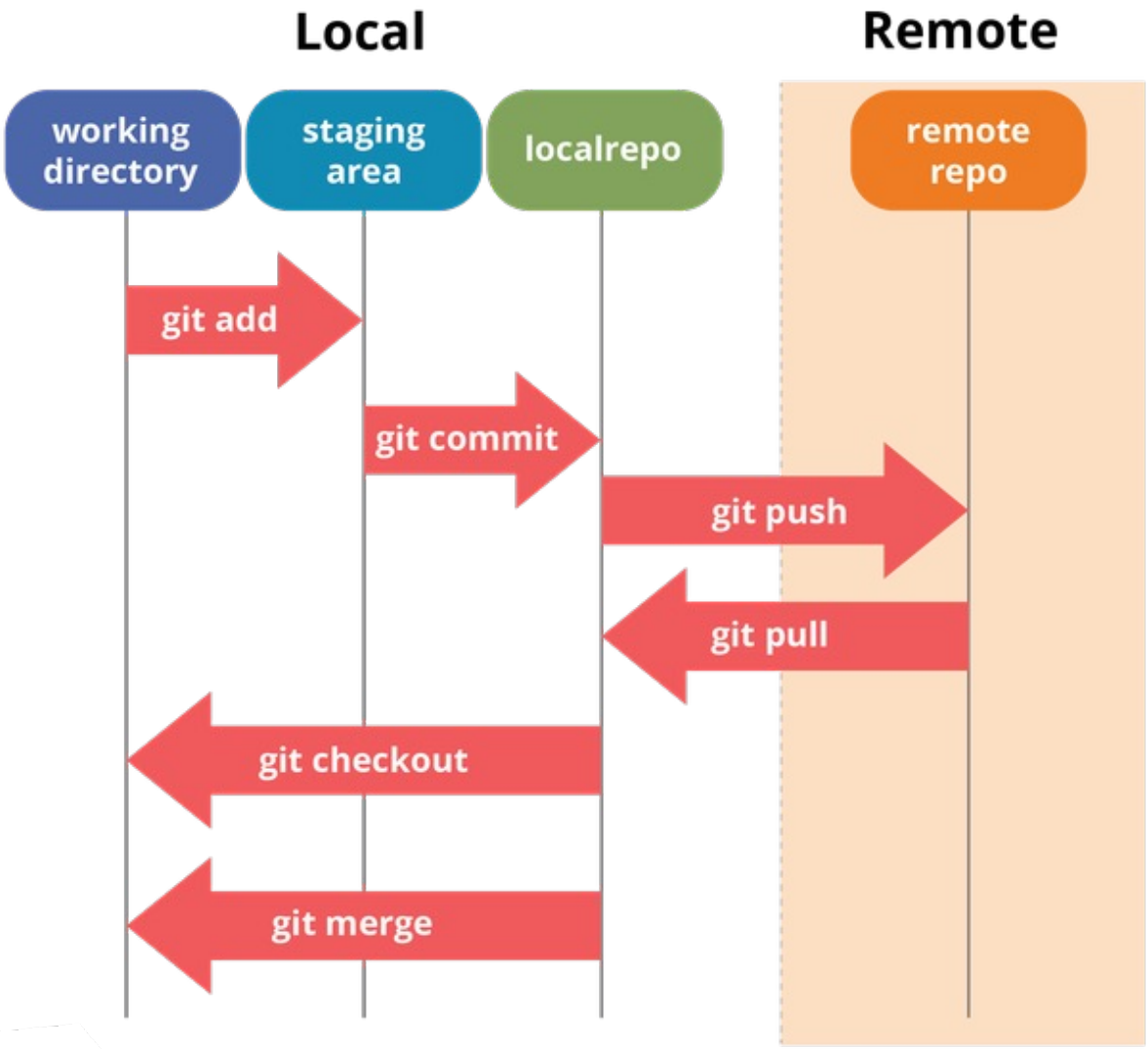
The instructions should list how to add a ”remote” to your Github repo, which is equivalent to a label for your Github url.

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

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

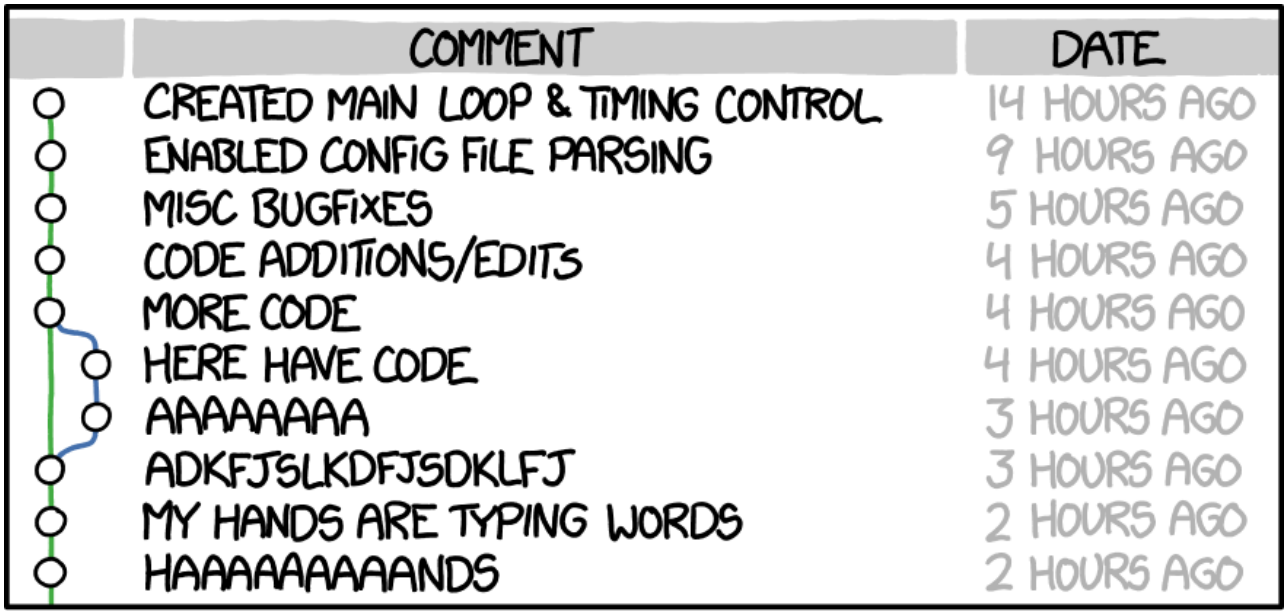
The steps are described here: https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token

Saving Github changes to local repo

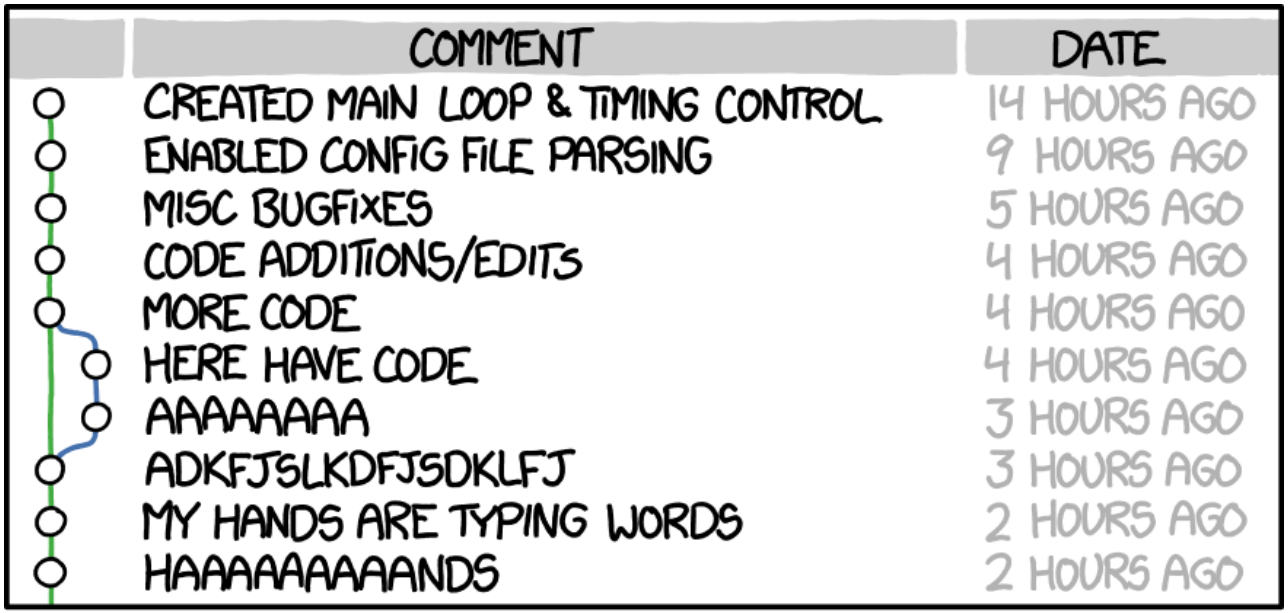
If you manually add files or make other changes on Github, you can sync those changes to your local git repo on KLC by doing the following:

```
git pull origin main
```

Discussion: Using Git and Github Smartly



Discussion: Using Git and Github Smartly



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 – 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
```