TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Git

PRESENTED BY:

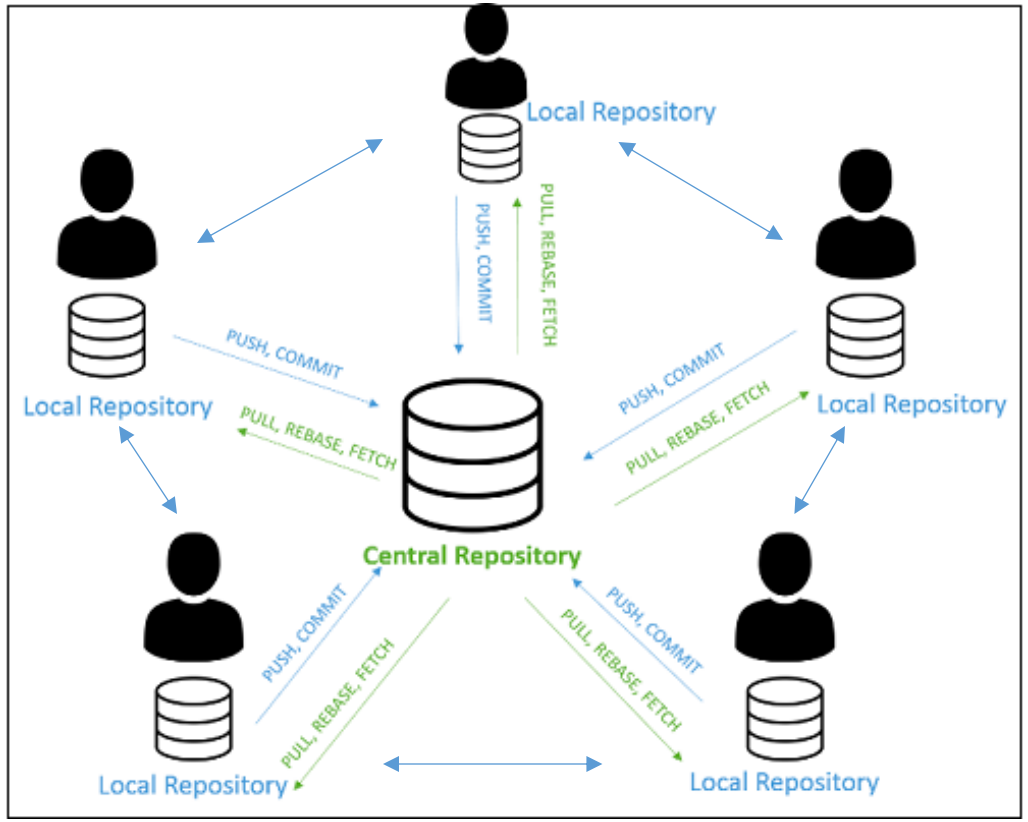
Brandi Kuritz

bkuritz@tacc.utexas.edu

Software Developer @ TACC

What is Git?

- **Version control for tracking revisions and changes in a set of files**
  - This set of files are referred to as a “repository”
- Distributed, non-linear model
- Aids in resolving code differences between different developers



Git: Repositories

- **Repository: The files and their Git history**
  - Remote repository: Housed in a Git server where everyone can access it.
    - Common examples: Bitbucket, Github, Gitlab
    - “ Upstream”
  - Local repository: Copy of the remote repository on a different machine (usually in reference to your local copy)
    - Will point to **one** remote repository
    - “Downstream”
- Designated by a /.git directory within the top directory

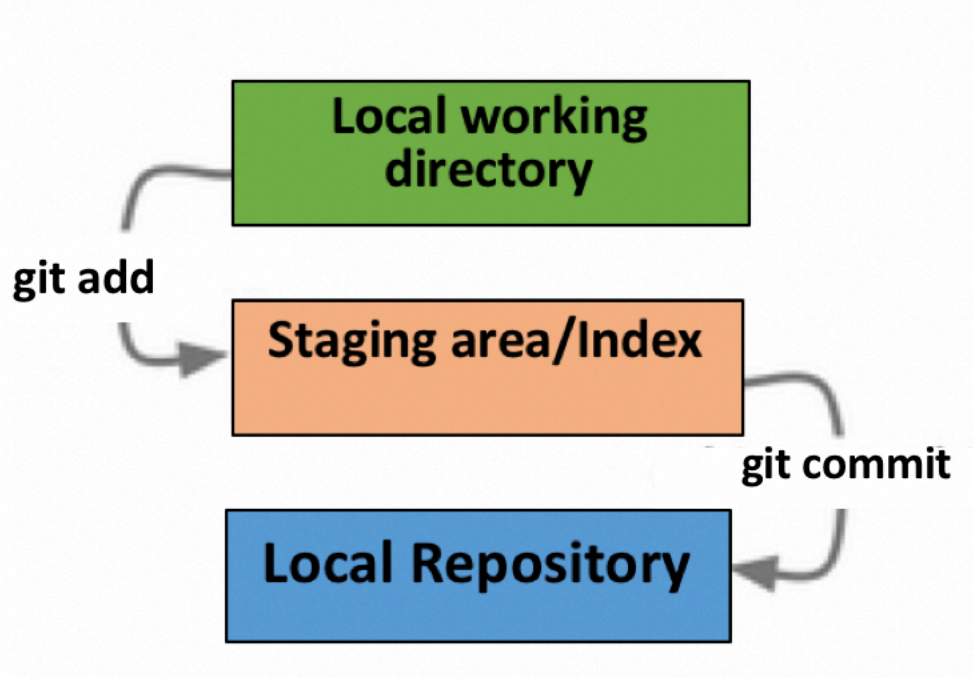
Git: Clone & Init

- Both are used to create a new Git repository
- `git init`: Used to create a new, local **repository from a local directory.**
  - git remote add origin <repo url>
- `git clone <repo url>`: Used to create a new, local repository **from an existing remote repository**
  - https://github.com/BranRitz/PracticeRepo

```
bkuritz:~ $ mkdir new_git
bkuritz:~ $ cd new_git/
bkuritz:~/new_git $ ll
total 0
drwxr-xr-x    2 bkuritz  staff    64 Sep 28 15:42 ./
drwxr-xr-x+ 134 bkuritz  staff  4288 Sep 28 15:42 ../
bkuritz:~/new_git $ git init
Initialized empty Git repository in /Users/bkuritz/new_git/.git/
bkuritz:~/new_git $ ll
total 0
drwxr-xr-x    3 bkuritz  staff    96 Sep 28 15:42 ./
drwxr-xr-x+ 134 bkuritz  staff  4288 Sep 28 15:42 ../
drwxr-xr-x   10 bkuritz  staff   320 Sep 28 15:42 .git/
bkuritz:~/new_git $
```

Git: Add & Commit

- **Add: Adds modified files to a staging area that will compose a commit**
- **Commit: A snapshot of changes between your repository and the remote repository.**
  - A commit is stored as a diff with a unique SHA-1 hash 40 character string of hex digits identifier.
  - The git history is composed of these commits
    - https://github.com/TACC-Cloud/tapis-cli/commits/main



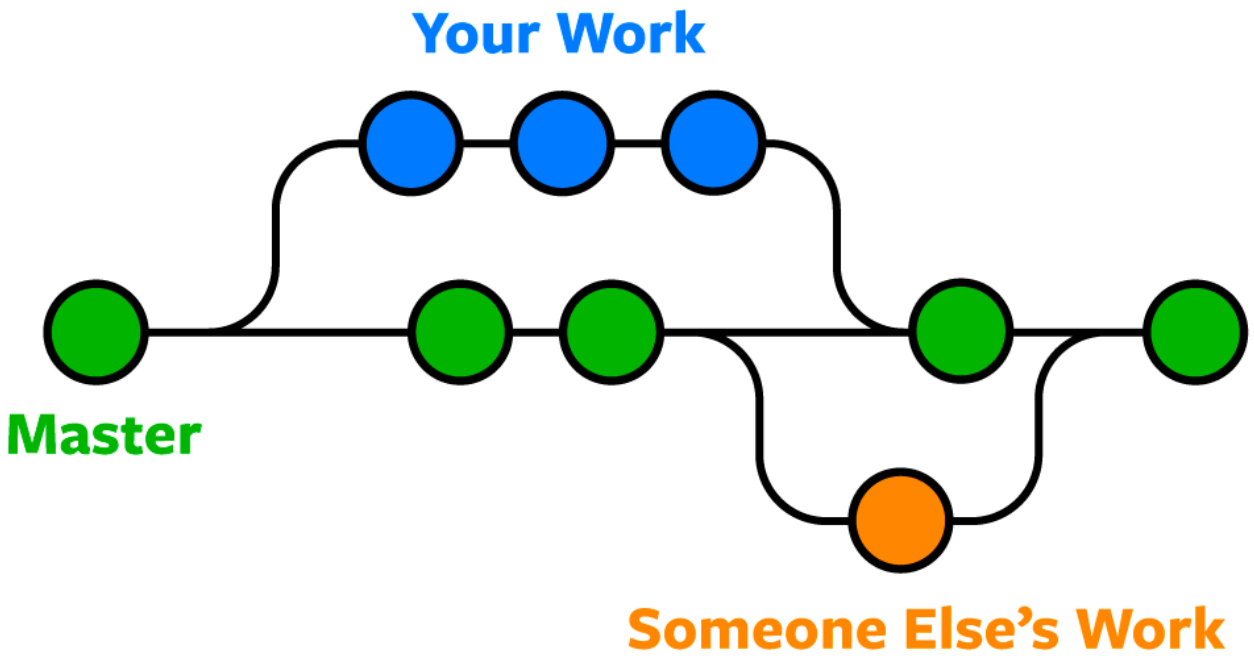
Git: Status & Diff

- `git status`: displays the state of the working directory and staging area
  - You can see which files are staged for the next commit, and which aren't.
- `git diff`: displays changes between local working directory and staging area

```
bkuritz:~/git/PycharmProjects/ConstructionAPI $ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   cii_api/db_scripts/csv_files/q_c.csv

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   cii_api/db_scripts/csv_files/q_c.csv

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        cii_api/__pycache__/
        cii_api/cii_api/__pycache__/
        cii_api/construction/__pycache__/
        cii_api/construction/migrations/__pycache__/
        cii_api/construction/question_data.py
        cii_api/construction/tests_bak.py
        cii_api/construction/urls_bak.py
        cii_api/construction/views_bak.py
        cii_api/db_scripts/csv_files/quartile_data.csv
        cii_api/db_scripts/csv_files/question_data.csv
        cii_api/db_scripts/quartile_data.py
```

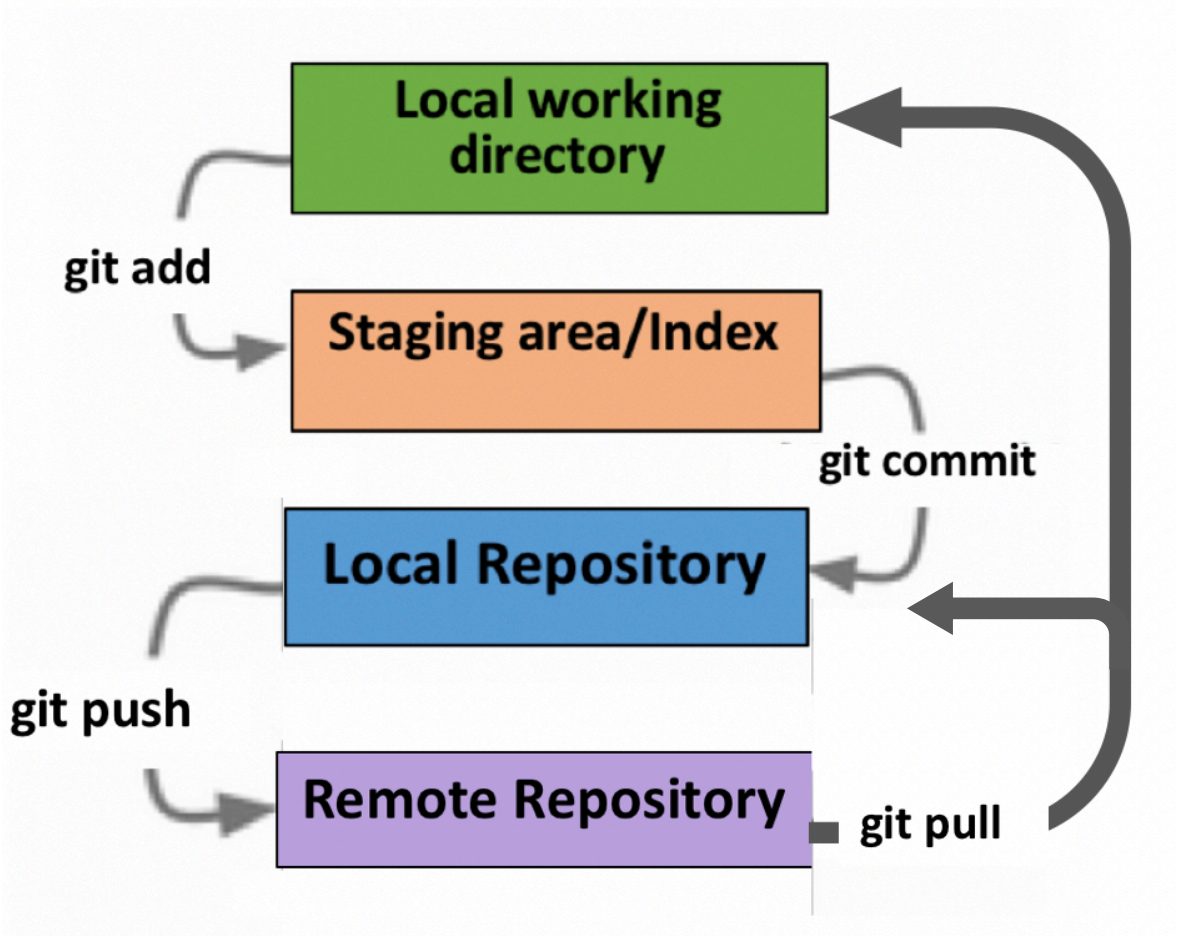
Git: Branches

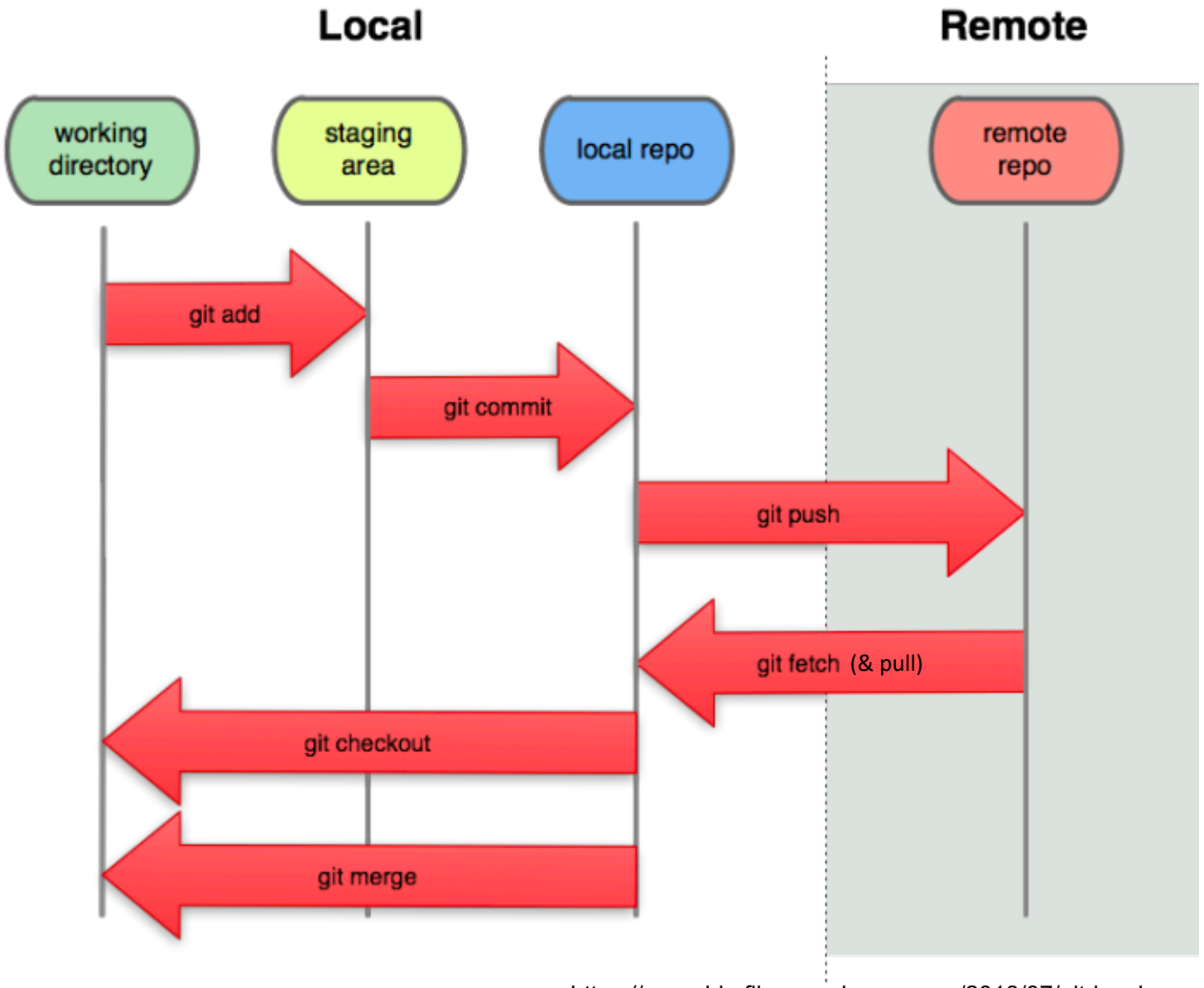
- **Branch: A pointer to a chain of commits**
- **Main/Master: The main branch that all changes get merged back to**
  - Created on repo initialization
  - Keeps questionable code out of the main source code
- **Fetch: Downloads data from remote repository without integrating changes to your local code**
  - You can do this when you want to see what other people are working on



Git: Push, Merge & Pull

- `git push`: Transfers your commits from your local repository to the remote repository
- `git pull`: Transfers commits made to the remote repository to your local repository and local working directory
- `git merge <branch name>`: Merges the specified branch into current branch
  - Can merge from local repository or remote repository: master vs origin/master

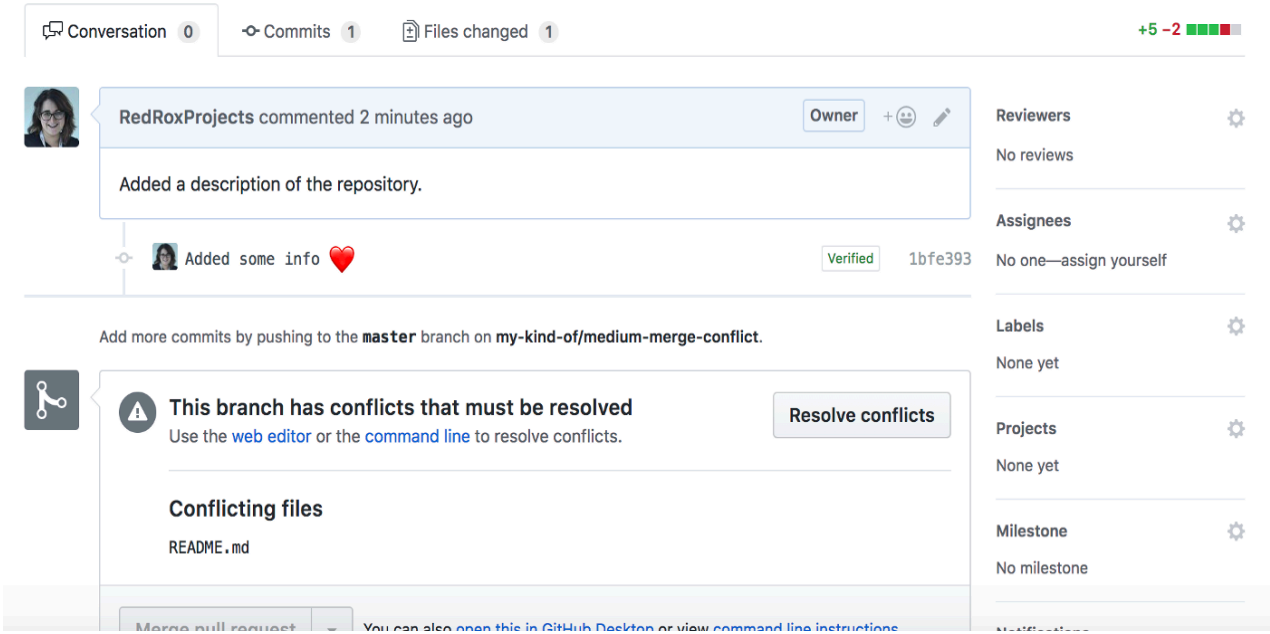


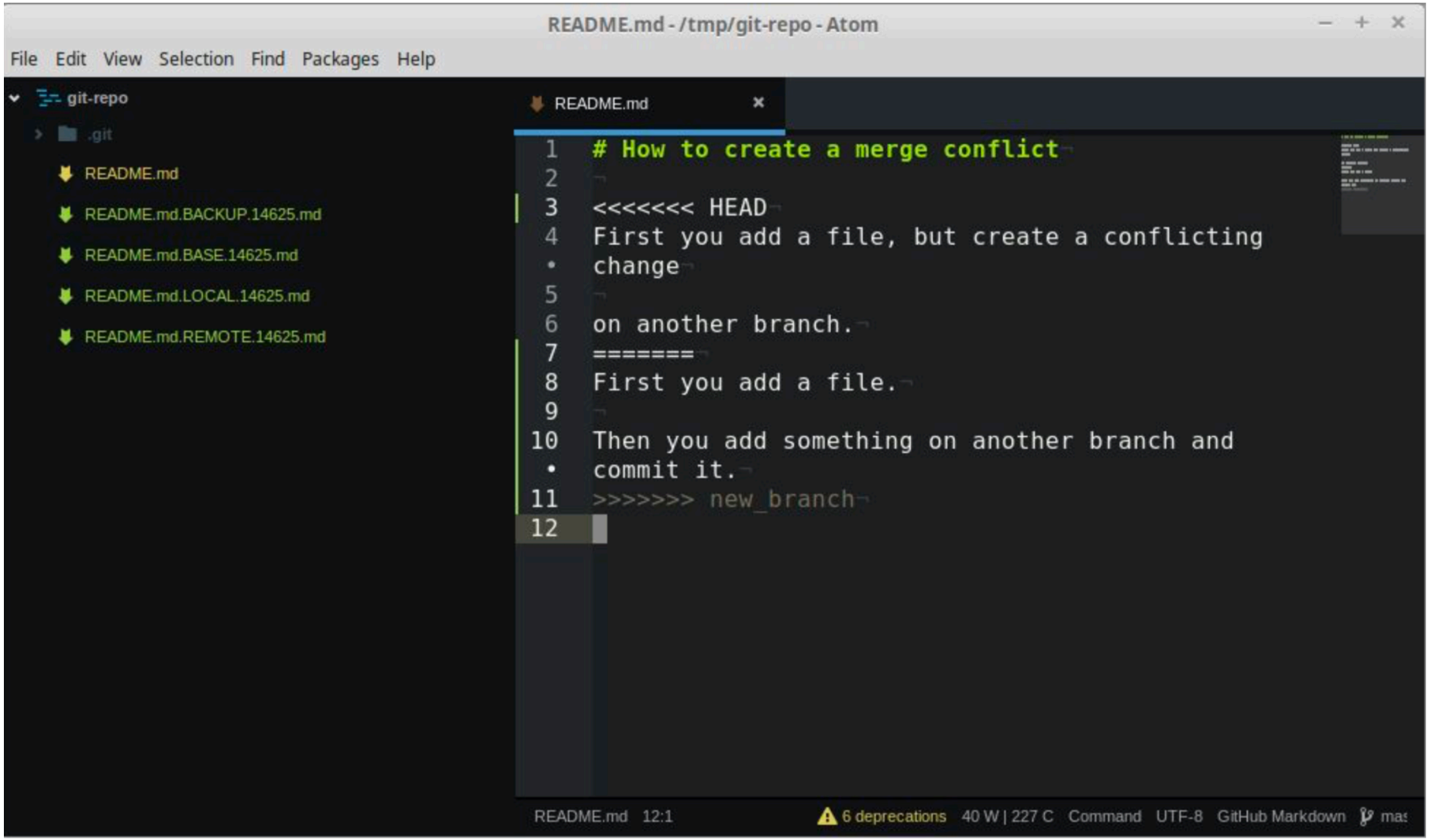


https://greenido.files.wordpress.com/2013/07/git-local-remote.png?w=696&h=570

Git: Merge Conflicts

- Merge conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it.
- Work in isolated branches to avoid this, but have to deal with it eventually when integrating code back into the main code




README.md - /tmp/git-repo - Atom
File Edit View Selection Find Packages Help
git-repo
.git
README.md
README.md.BACKUP.14625.md
README.md.BASE.14625.md
README.md.LOCAL.14625.md
README.md.REMOTE.14625.md
README.md
1 # How to create a merge conflict
2
3 <<<<<<< HEAD
4 First you add a file, but create a conflicting change
5
6 on another branch.
7 =======
8 First you add a file.
9
10 Then you add something on another branch and commit it.
11 >>>>>>> new_branch
12
README.md 12:1
6 deprecations
40 W | 227 C
Command
UTF-8
GitHub Markdown

Git: Making Mistakes

- `git log`: allows you to see what changes you've made locally
- `git checkout <filename>`: reverts the specified file to the last commit
- `git reset --hard HEAD`: Can revert everything locally to the previous commit and deletes the most recent commit.
  - Can specify how many commits to back track. `git reset --hard HEAD~3` backtracks by 3 commits.

Git: Help & Resources

- Git cheat sheet: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- Official docs: https://git-scm.com/docs
- Google ☺

Git: Example of Basic Workflow

Basic Git workflow:

1. Pull from the remote repository: `git pull`
2. Modify files in your working directory.
3. Stage files, adding snapshots of them to your staging area: `git add <file name>`
4. Do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory: `git commit –m "<message>"`
5. Push commit(s) to the remote repository: `git push`
   - Deal with any merge conflicts that may come up
6. If on a separate branch, you can now merge your branch to a new branch (not needed if you're not using branches): `git merge <branch to merge in>`

Git: Getting Started

- Verify you have Git installed: `$ git version`
- Set the name and email for Git to use when you commit:

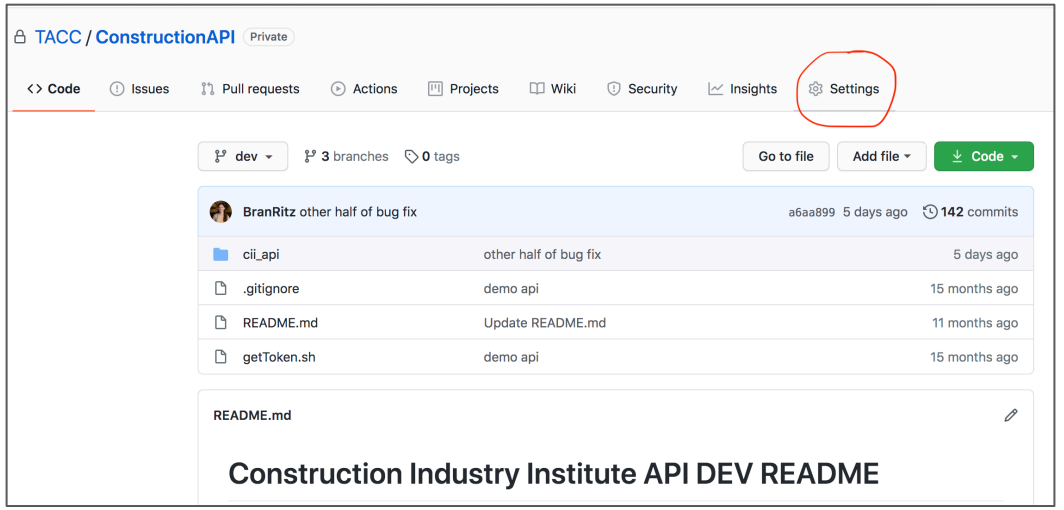
  `$ git config --global user.name Brandi`

  `$ git config --global user.email bkuritz@tacc.utexas.edu`
  - Use `git config --list` to see the set variables. These will be set globally for all Git projects you work with. You will be prompted for your git password when you access the remote repository.
  - You can also set variables on a project-only basis by not using the --global flag.

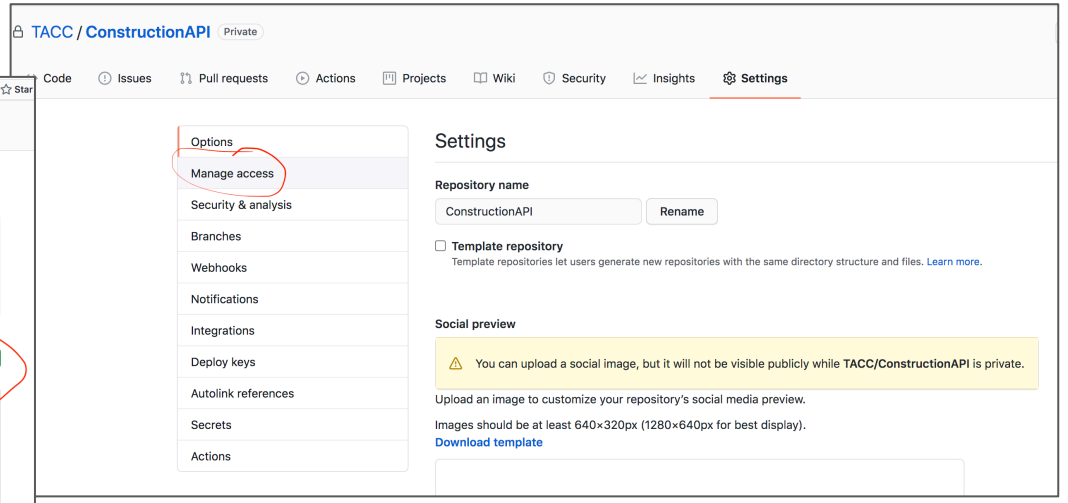
Git: Create Group Repo

1) Select one student to create a remote repository in Github
2) Have this student add all the other students in the group

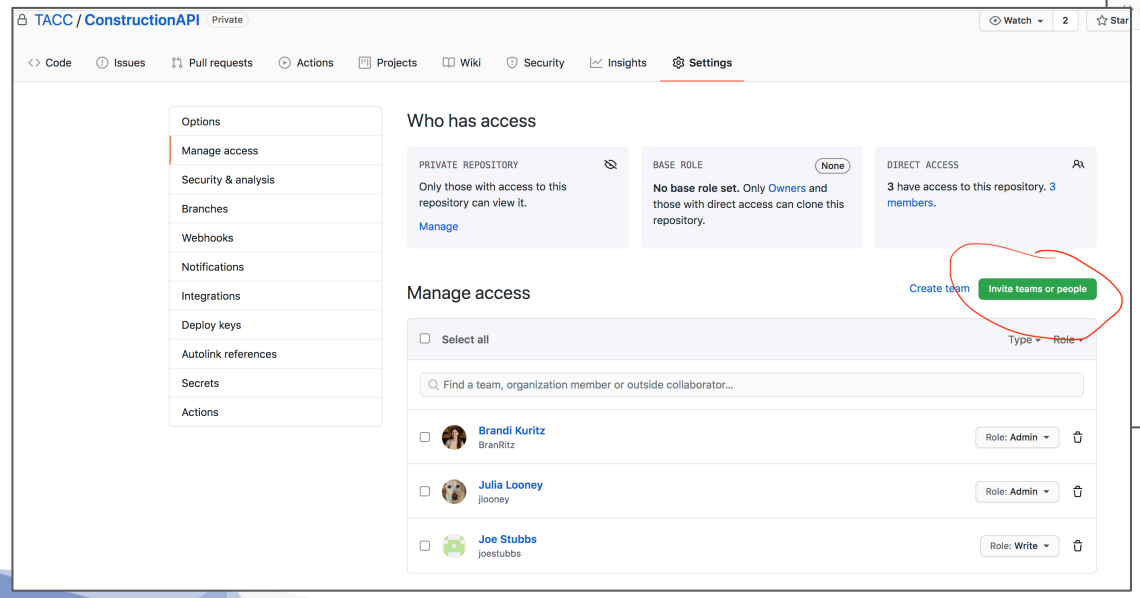
1.



2.



3.



Git: In Class Exercise

1) Clone your group's repository
2) Open the repository in your editor of choice
3) Create a new file `<your name>.txt` and add a fun fact about yourself to this file
4) Add the file to the remote repository main branch using the workflow we discussed

Workflow recap:

- `git status` & `git pull` to see what files have/haven't been staged
- Edit files locally
- `git add <file name>` to stage your file
- `git commit` to create a snapshot
- `git push` to add it to the remote repository
- `git checkout <branch name>` & `git merge <branch name>` to merge to branch (if applicable)

Git: In Class Exercise

5) Everyone pull down your teammates' changes: `git pull`

6) Delete your file from your local repository and push the change

7) Without doing a pull first, open someone else's file and edit it. Try to push this in. You will likely encounter a merge conflict. Work through this conflict and then push it in.

Git: Assignment, Due Wednesday 10/7

- Continue playing around in your local repository. Try the following:
  1. Create a new directory: `<your UTEID>`
  2. Create three files in this directory:
     - `<your initials>_thoughts.txt`
     - `<your initials>_status.txt`
     - `<your initials>_noadd.txt`
  3. In `<your initials>_thoughts.txt`, write 1-2 paragraphs about your experience so far with git. Have you used it before? Did you like it? Hit any hurdles?
  4. Only push `<your initials>_thoughts.txt` and `<your initials>_status.txt` to the remote repo. Do not push `<your initials>_noadd.txt`
  5. Once that is all pushed in, do a `git status`. Copy the text and paste it into `<your initials>_status.txt`. Provide a brief explanation of what the text you pasted means. Push this change into the repo.
- Have one group member upload your git log for the repository created to canvas. Key points that we will be looking for:
  - `<your initials>_thoughts.txt` is added and contains the requested paragraph(s)
  - `<your initials>_practice.txt` was added and edited in two different pushes, and contains a meaningful explanation of the `git status` text
  - `<your initials>_noadd.txt` does not appear in the remote repository
  - Each group member has at least 2 commits, pushed in 2 separate pull requests