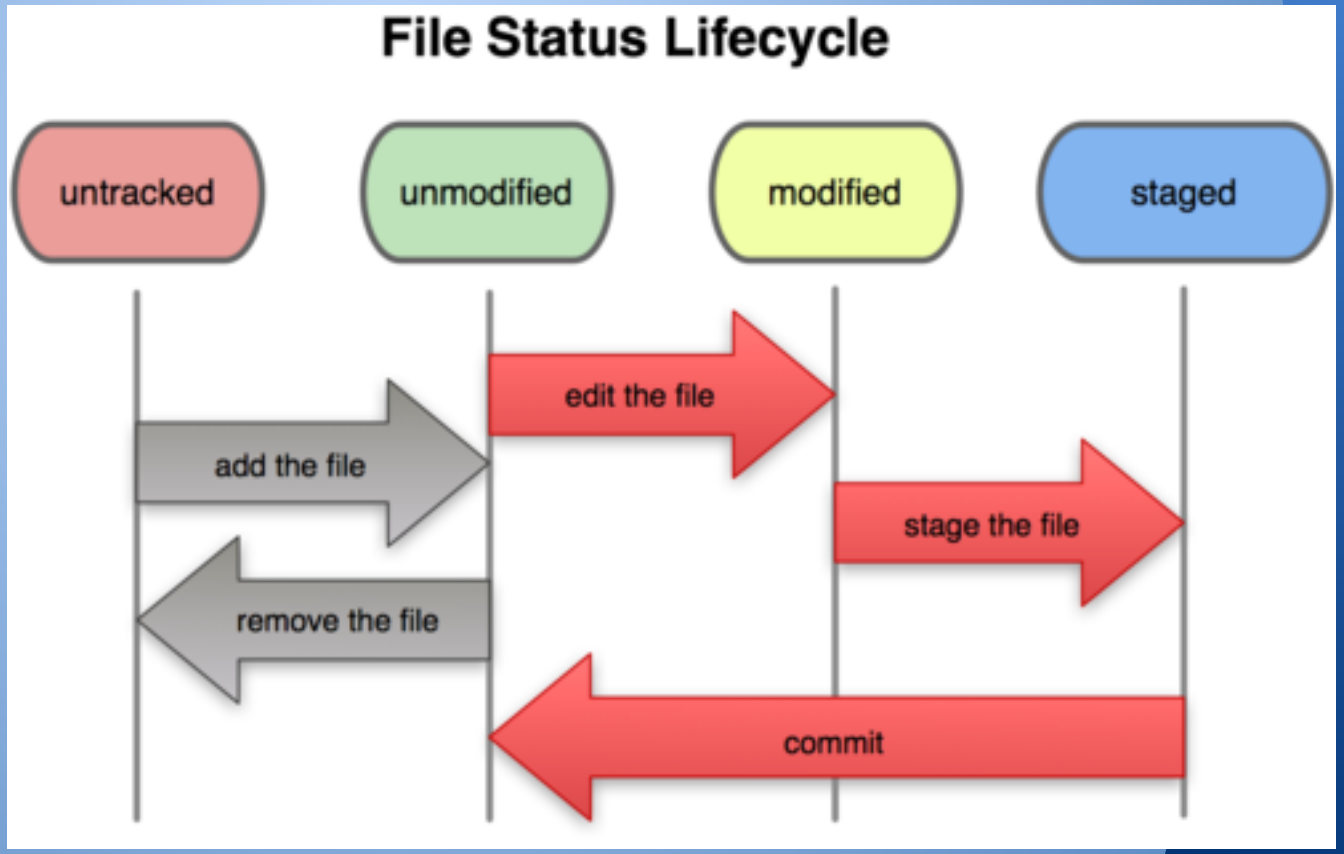
Version Contol 2:

Using git as part of your daily workflow

Does anyone know how to attach a severed head?

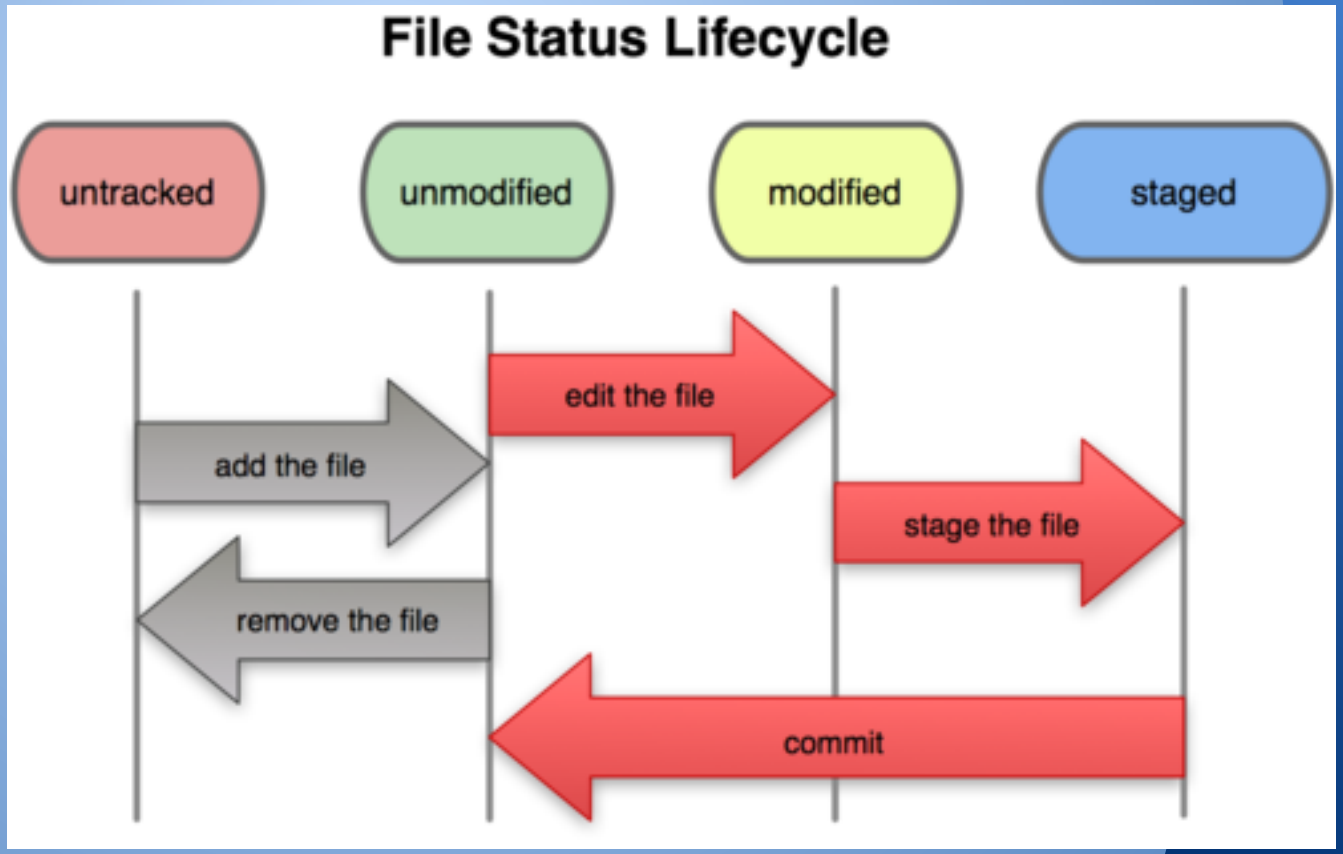
Local workflow basics

- git init - create a repository in a given directory



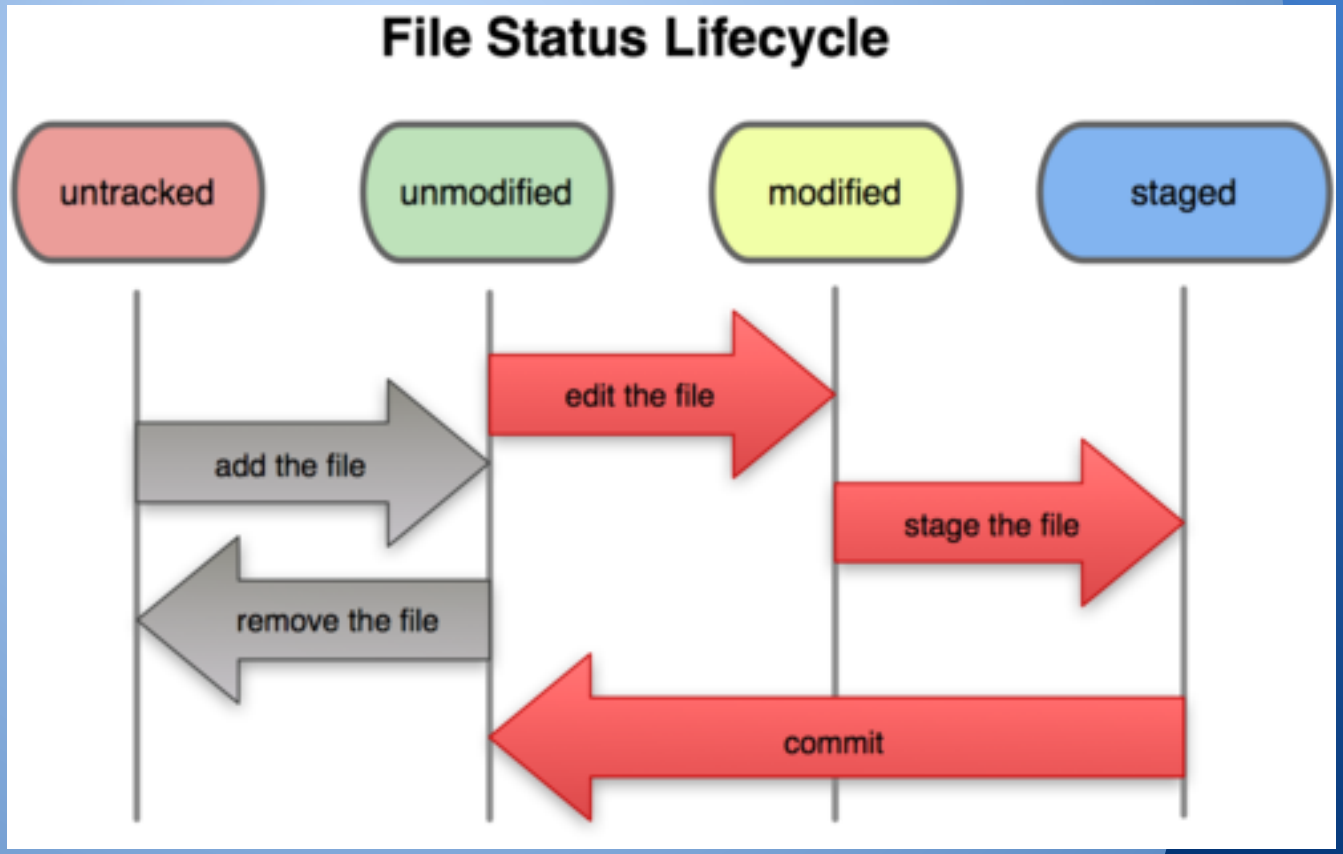
Local workflow basics

- git status - find out what stage every file in a directory is in



Local workflow basics

- git add filename or directory - add file to list of file to be committed to local repository. This is also referred to as staging the files



Getting help

- git help
- git help command (e.g. git help status)

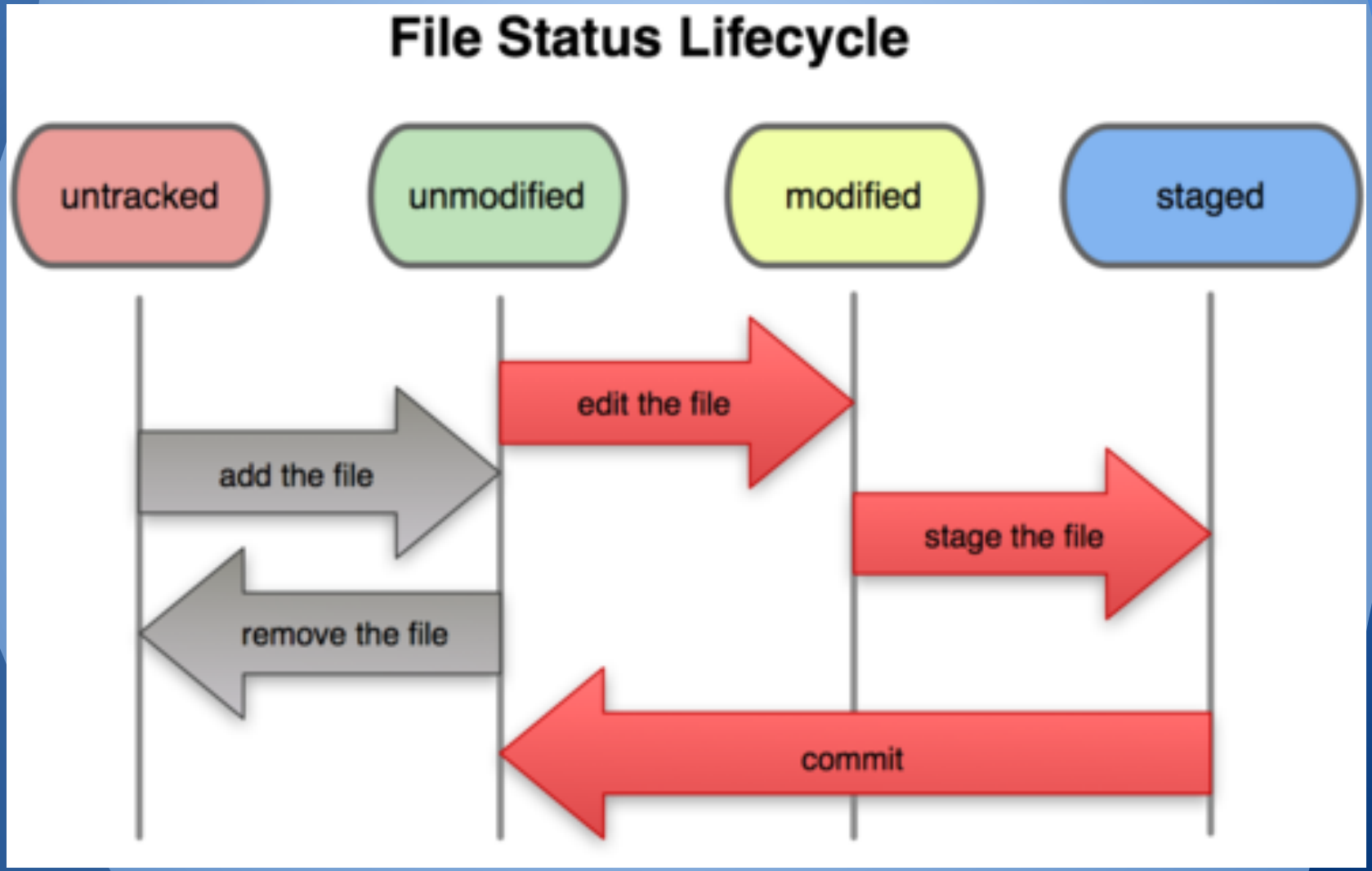
Exercise

1. Copy run19.mod to your repository and add it to your staging area.

Local workflow basics

- git commit -m "detailed commit message"
- What if you forget -m? go to your default editor
  - If your default editor is vi or vim:
    - type i (for insert) to start typing
    - When you are done typing type:
      - ESC
      - :wq (for write, quit)

Local Workflow



Exercise 2:

1. commit your staging area to your local repository (don't forget a commit message)
2. Modify run19.mod then:
   a. save your changes
   b. what is the status of run19.mod?
   c. add run19.mod to your staging area
   d. commit run 19.mod

Undoing Mistakes:

- Un-staging a file
  - git reset HEAD filename
- Un-modifying a filen
  - git checkout -- filename

Exercise 3:

- Modify run19.mod and save your changes
- Add run19.mod to your staging area
- Remove run19.mod from your staging area
- unmodify run19.mod using git

Viewing differences

- Everything: git diff
  - not recommended
- A single file:
  - git diff filename

- + added since last staging
- - removed since last staging

Exercise 4:

1. Modify run19.mod and save your changes
2. Use git diff to find your changes
3. Stage your changes
4. Run git diff again, do you get a different output?
5. Commit your changes (don't forget your commit message)

Viewing History

- git log (all history)
- git log -2 (last 2 entries)

Shell commands in git

- git mv
  - tells git your are renaming (and possibly changing the location of a file) so it can continue to track it
- git rm
  - tells git you are removing a file from a repository

Remote Repositories

Find a partner

Github

- You should already have a github account
- Remote repository server
- Lots of good projects
- Easy to explore code

Create a remote repository

1. sign in
2. Click on the repositories tab
3. Click the new botton (its green)
4. Fill in repository name, description
5. check the initialize this repository with a readme file

Your repository will always be public if you are using the free version of github

Create local copies of your remote repository

1. Choose one person's repository for both of you to clone
2. In github, click on the repository you chose to clone.
3. Copy the url (make sure http is selected)
4. in git bash type:
    a. git clone url
5. This should have created a local copy of your repository

Exporing your local/remote repository

- git remote -v
  - What does git call my remote repositories?
  - Default: origin
- git branch
  - What branch am I on?
  - Default: master

Recall - local version control

1. Have one person copy a file into the repository
2. Add the file to the staging area
3. Commit your file

Saving changes to the remote repository

- git push remote_name branch_name (e.g. git push origin master)
- You will have to enter your github username and password.

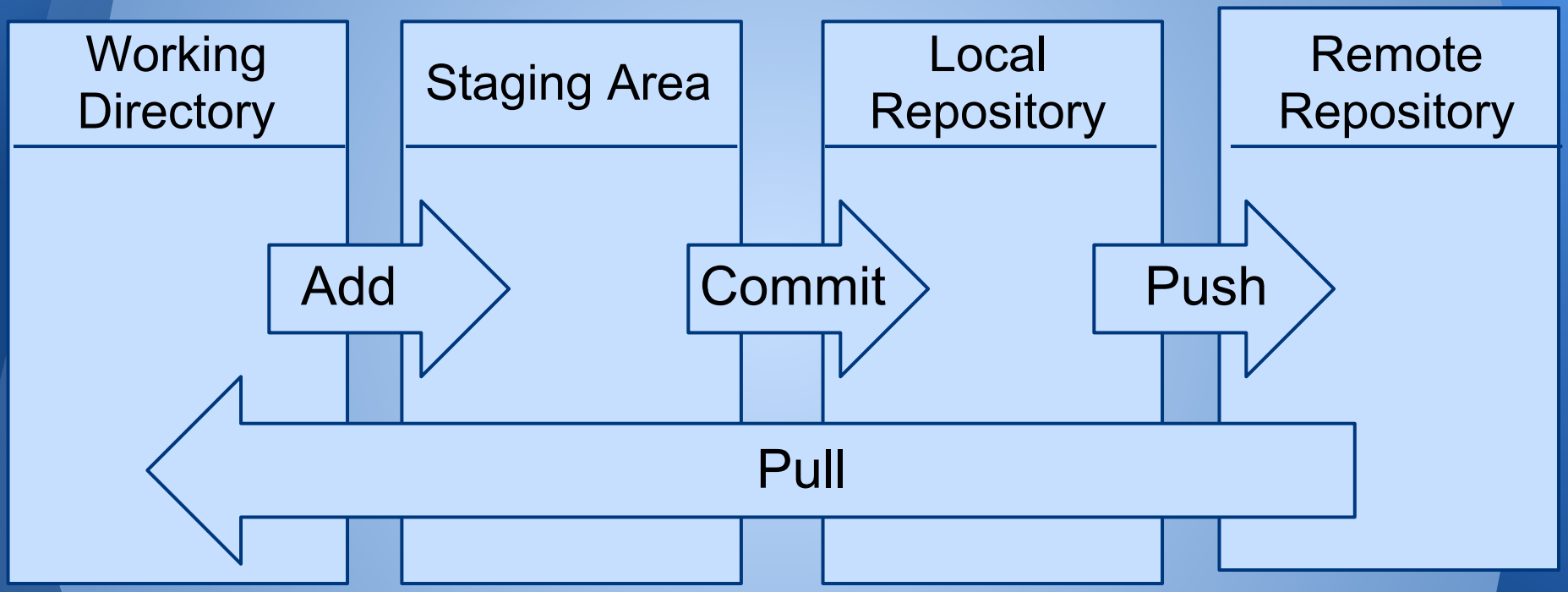
Getting changes from the remote repository

- The person who did not just commit something should type:
  - git pull remote_name branch_name (e.g. git pull origin master)
- Check your repository - you should have the new file

Switch roles and repeat

1. Have one person copy a file into the repository
2. Add the file to the staging area
3. Commit your file
4. Push your changes to the remote repository
5. Have the other person pull the changes

Workflow:



Before starting work, you should always pull to make sure you are modifying the most up to date files

Learn More:

http://git-scm.com/book