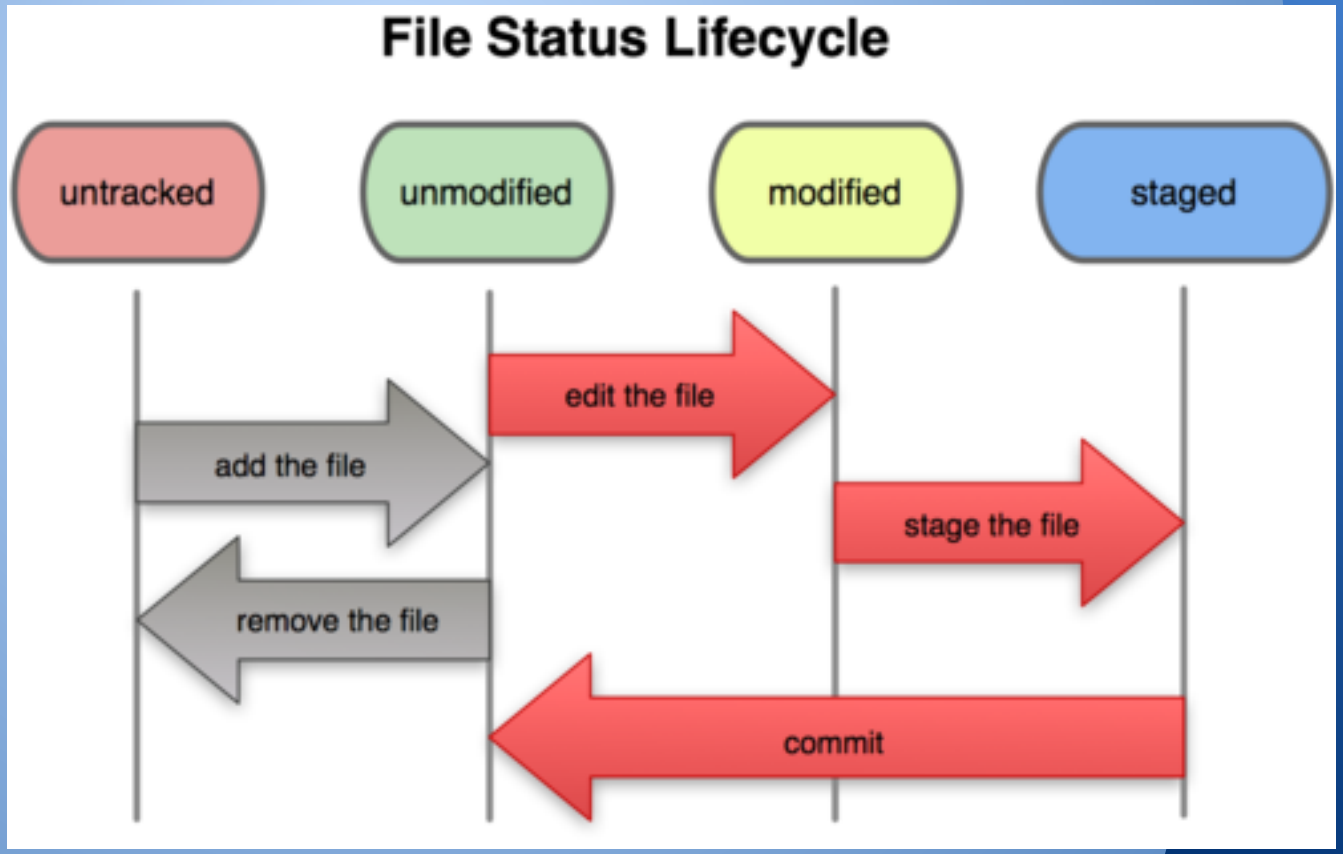
Version Contol 2:

Using git as part of your daily workflow

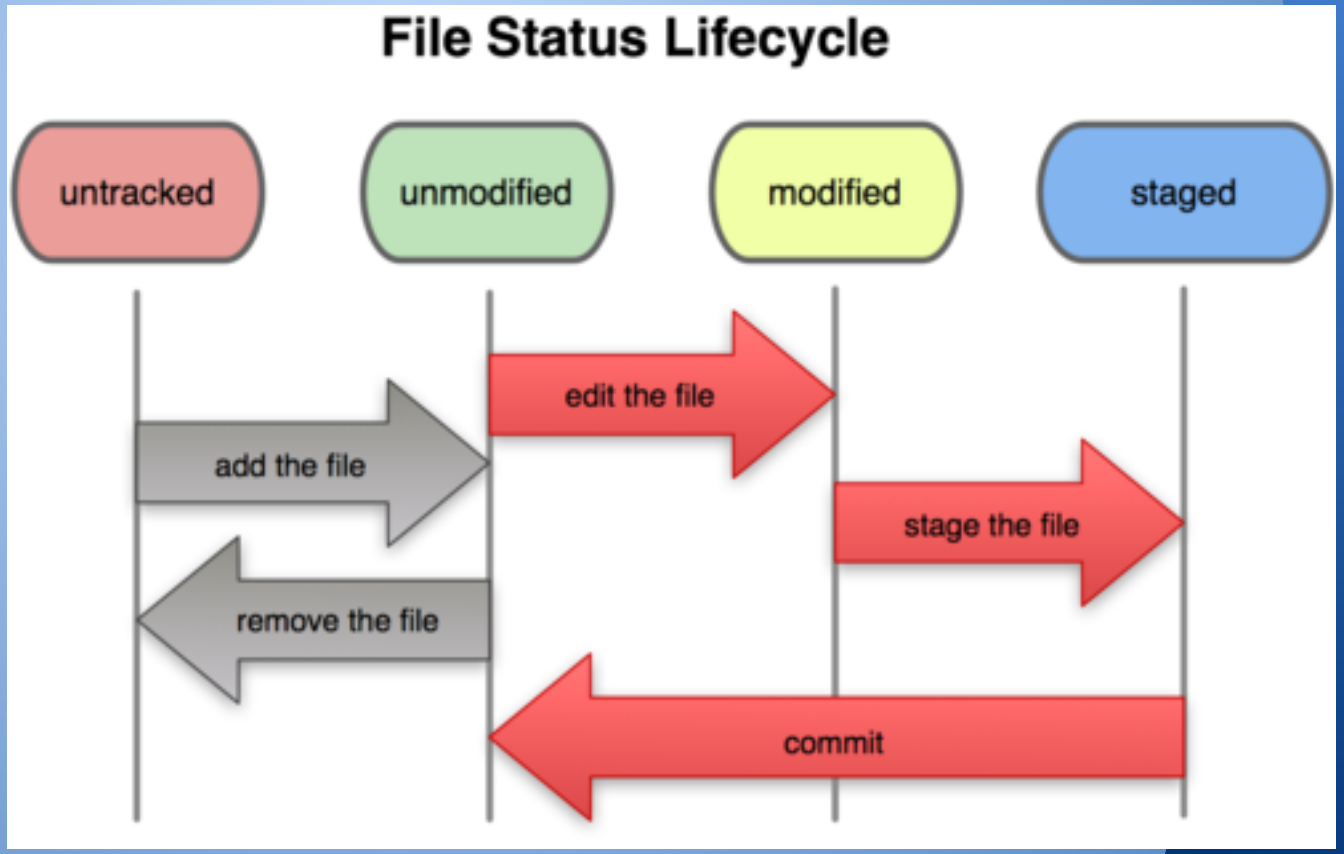
Local workflow basics

- **git init** - create a repository in a given directory



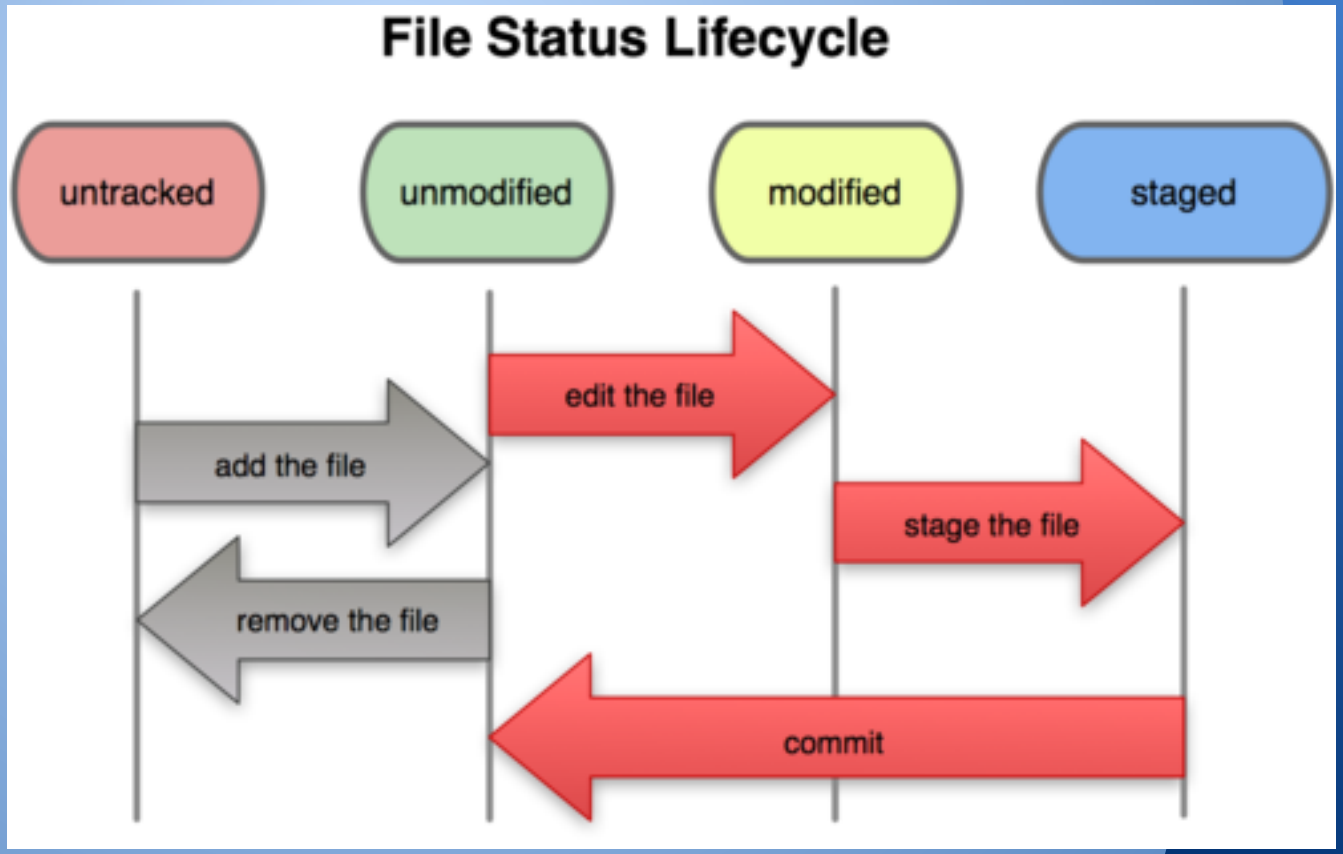
Local workflow basics

- **git status** - find the state of every file



Local workflow basics

- **git add** filename or directory - add file to list of files 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)

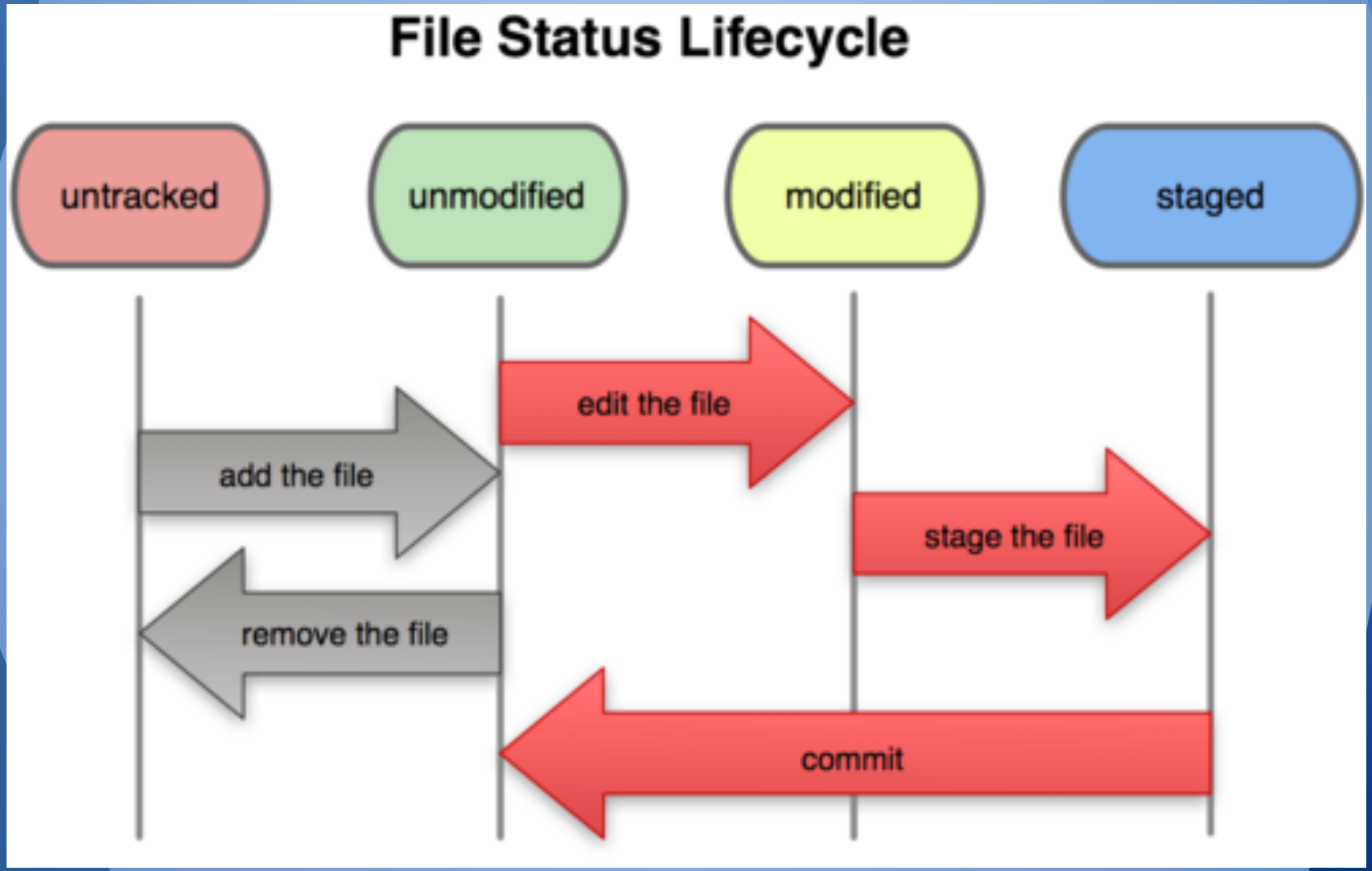
Exercise 1

1. Copy audioresult-00215 from boot-camps/shell/data/Bert 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 (often vi or vim)
  - remember:
    - i to insert
    - esc to exit insert
    - :wq to write and quit

Local Workflow



Exercise 2:

1. if you haven't already done so, commit your staging area to your local repository (don't forget a commit message)
2. Modify audioresult-00215 then:
   a. save your changes
   b. what is the status of audioresult-00215?
   c. add audioresult-00215 to your staging area
   d. commit audioresult-00215

Undoing Mistakes:

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

Exercise 3:

- Modify audioresult-00215 and save your changes
- Add audioresult-00215 to your staging area
- Remove audioresult-00215 from your staging area
- unmodify audioresult-00215 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 audioresult-00215 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

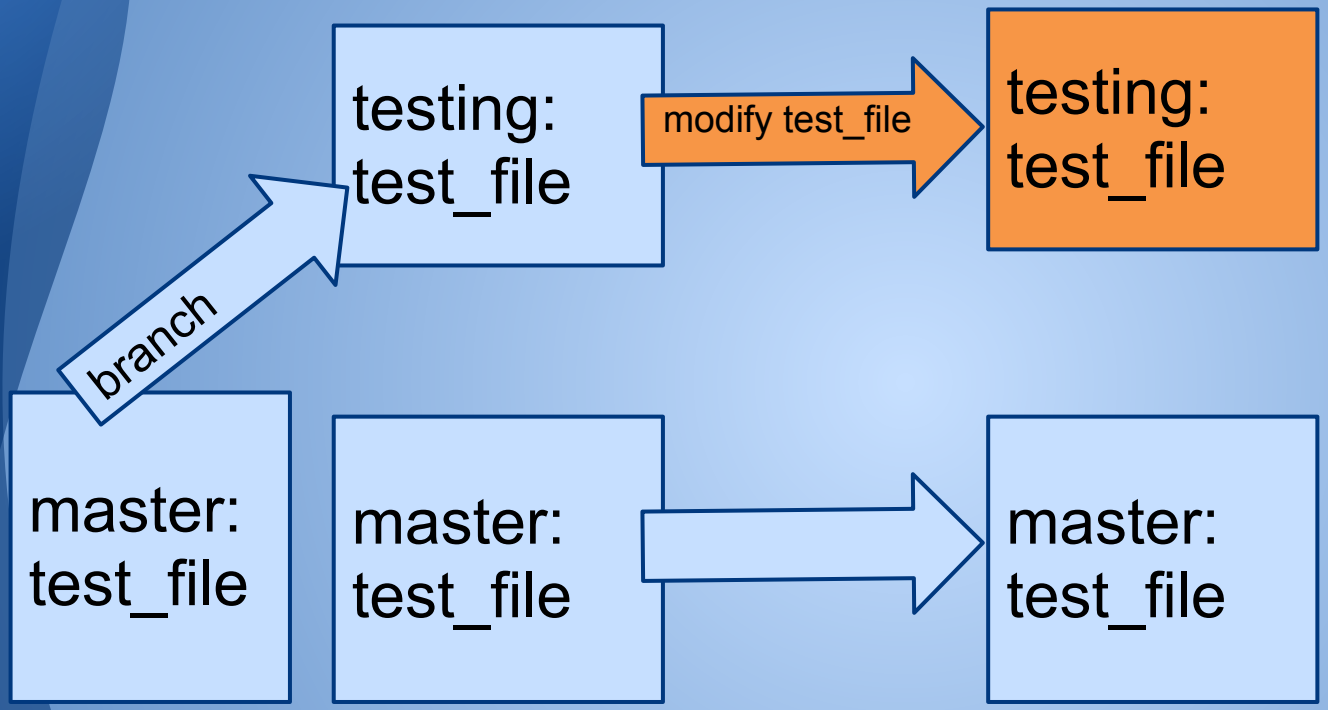
Branching

- Why: try new code without messing up working version
- branch: version of file that can be modified without affecting the working version of the code
- HEAD: points to current branch
- master: default branch name

Basic branching

- git branch testing: create a branch called testing
- git branch: tells you which branch you are on using *
- git checkout testing: switch to testing branch

branching:



http://pcottle.github.io/learnGitBranching/?NODEMO

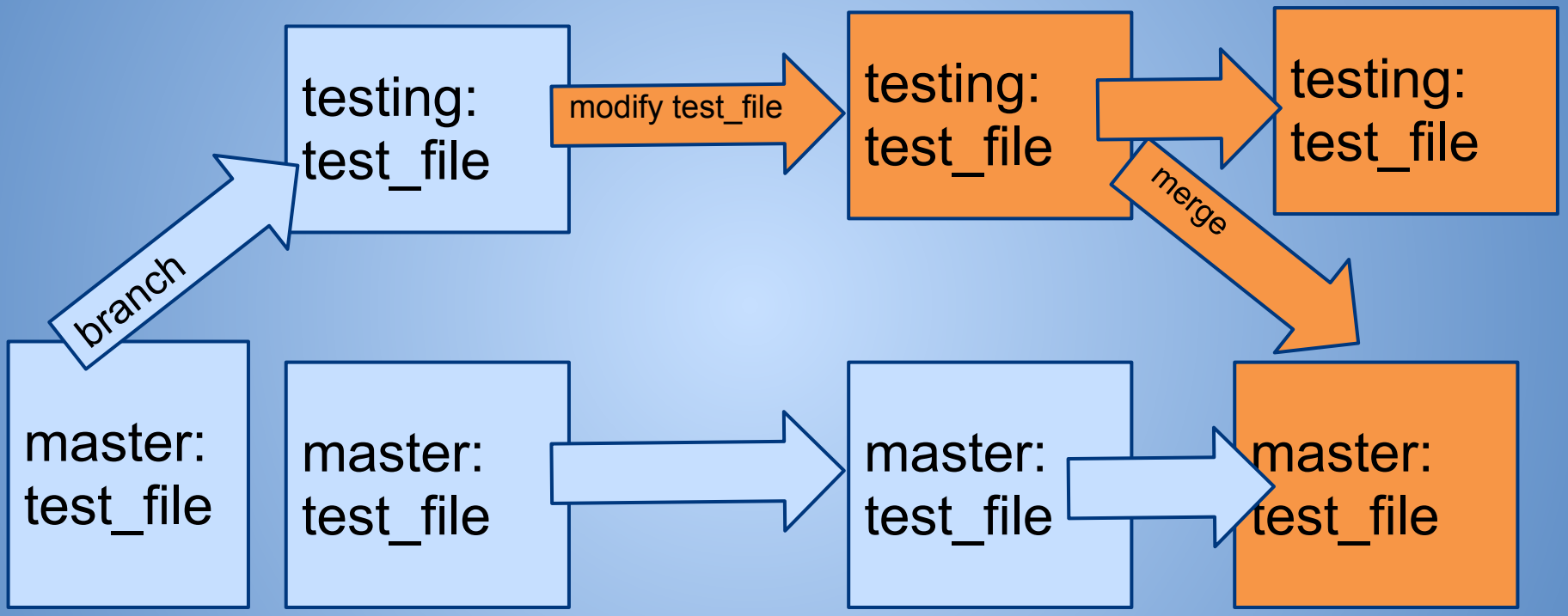
Exercise 5:

1. create a new file called exercise5.txt
2. Before you add and commit the file, check is it visible in your file system on the master branch? What about the testing branch?
3. add and commit your file to your master branch
4. Now is it visible in your file system on the master branch? What about the testing branch? Why is this different from #2?
5. create a branch called exercise_branch from master
6. move to exercise_branch and modify exercise5.txt
7. add and commit exercise5.txt to exercise_branch
8. go to your master branch. Are your modifications to exercise5.txt visible on master?
9. modify exercise5.txt on your master branch. Before you and commit your changes, try to switch to exercise_branch. Can you? What error message do you get?
10. Add and commit your changes to the master branch

Merging:

- go to branch you want to merge into
- git merge branch_you_want_to_merge
- No conflicts? awesome
- Conflicts?
  - resolve, add, commit
- merging doesn't delete or modify merged branch
- git branch -d branch_name - deletes branch

merging:



http://pcottle.github.io/learnGitBranching/?NODEMO

Remote Repositories

Find a partner

Github

- create 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 "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. Add the other person as a collaborator
3. In github, click on the repository you chose to clone.
4. Copy the url (make sure http is selected)
5. in git bash type:
    a. git clone url
6. 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)
    - master is the remote and local branch name
- Check your repository - you should have the new file
- pull = fetch + merge

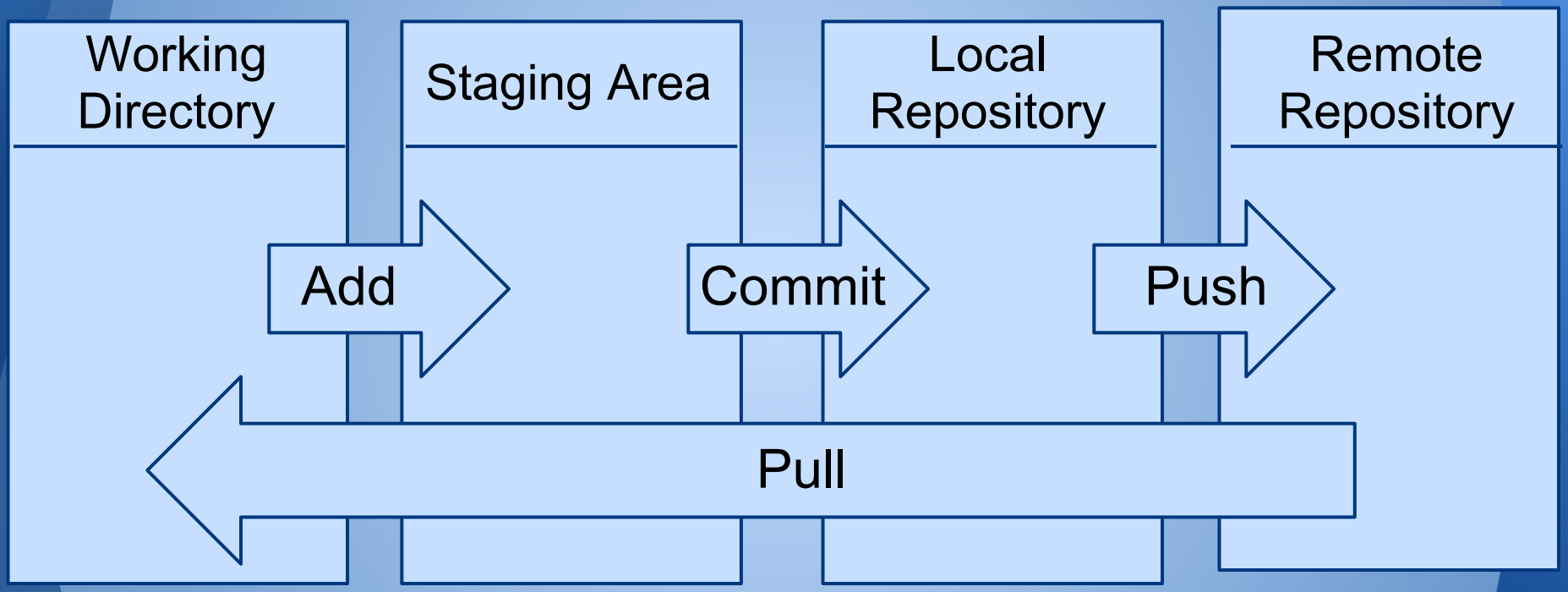
Switch roles and repeat

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

Resolving conflicts:

1. Both modify the same line in a file.
2. Have one person add, commit, and push their change
3. Have the other person add, commit, and push their change
4. Resolve the conflict, add, commit, and push
5. Repeat and switch who commits first

Workflow:



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

Not covered (incomplete list)

- remote branching
- tags

Learn More:

- http://git-scm.com/book