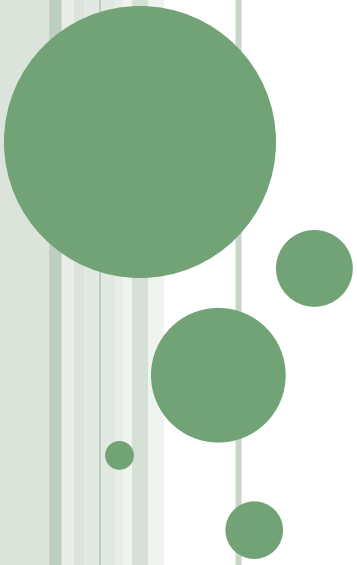


Version Control: git

A complete history of your work

What is version control?

- A tool to:
  - Back-up files
  - Save history of changes
  - Collaborate and combine changes
- But it isn't magic…
  - You have to know how to use it
  - You have to make it part of your work flow

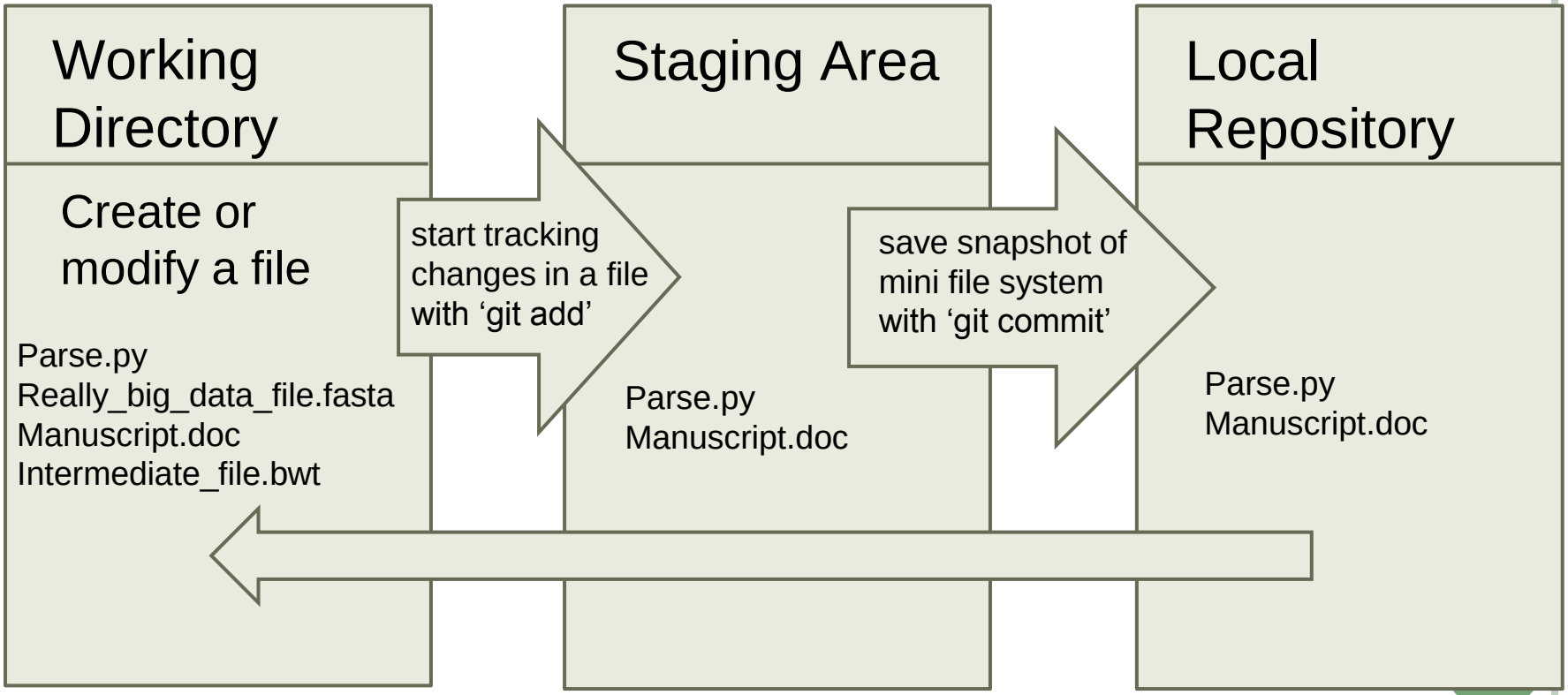
Why version control?

Case Studies:

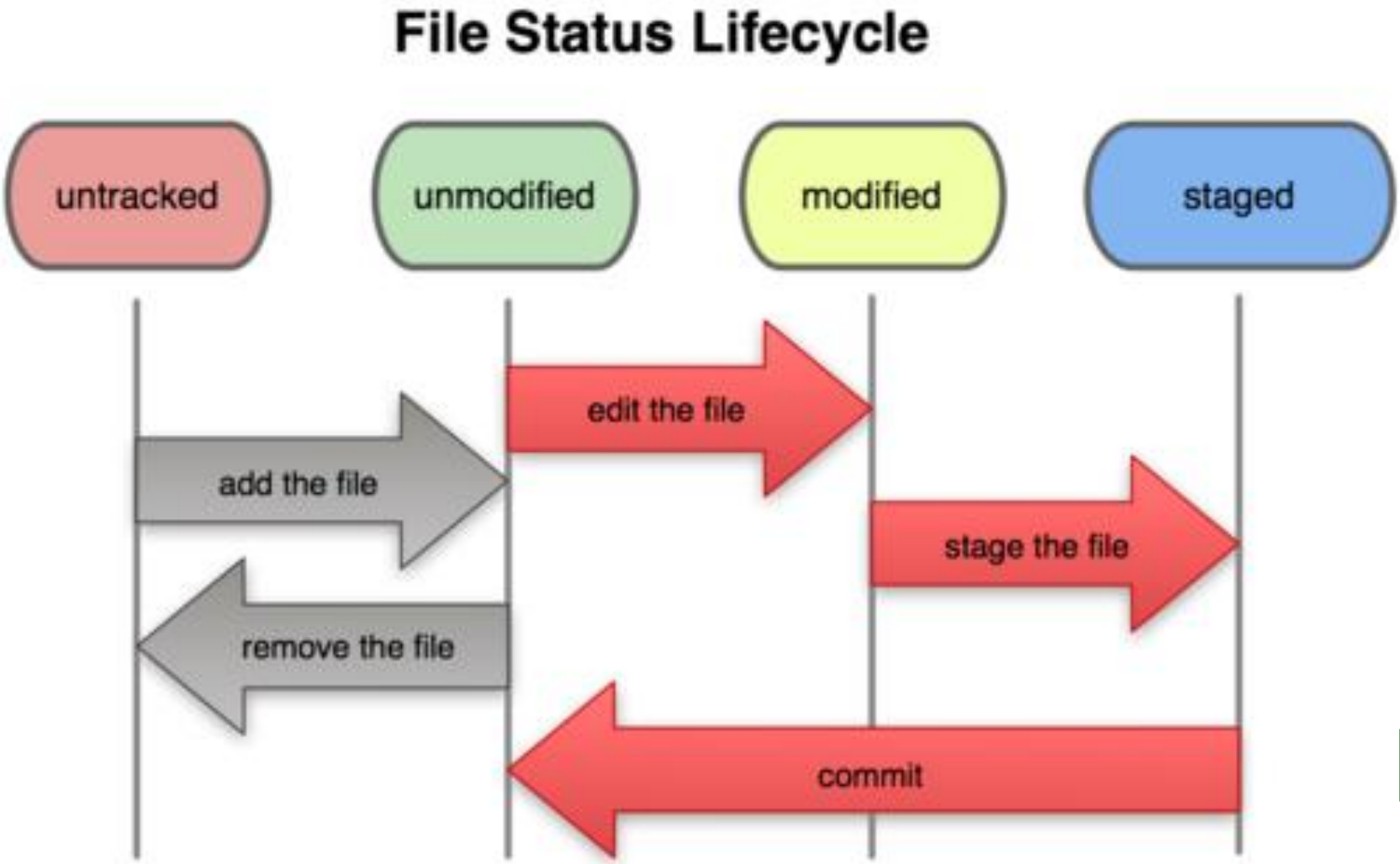
1. Know what changes you made when
2. Avoid files named ..._final_final2
3. Your code used to work and now it doesn‘t
4. Make your process transparent to others
5. Make your code easy

How? The Structure of Git

Local repository - track your own changes

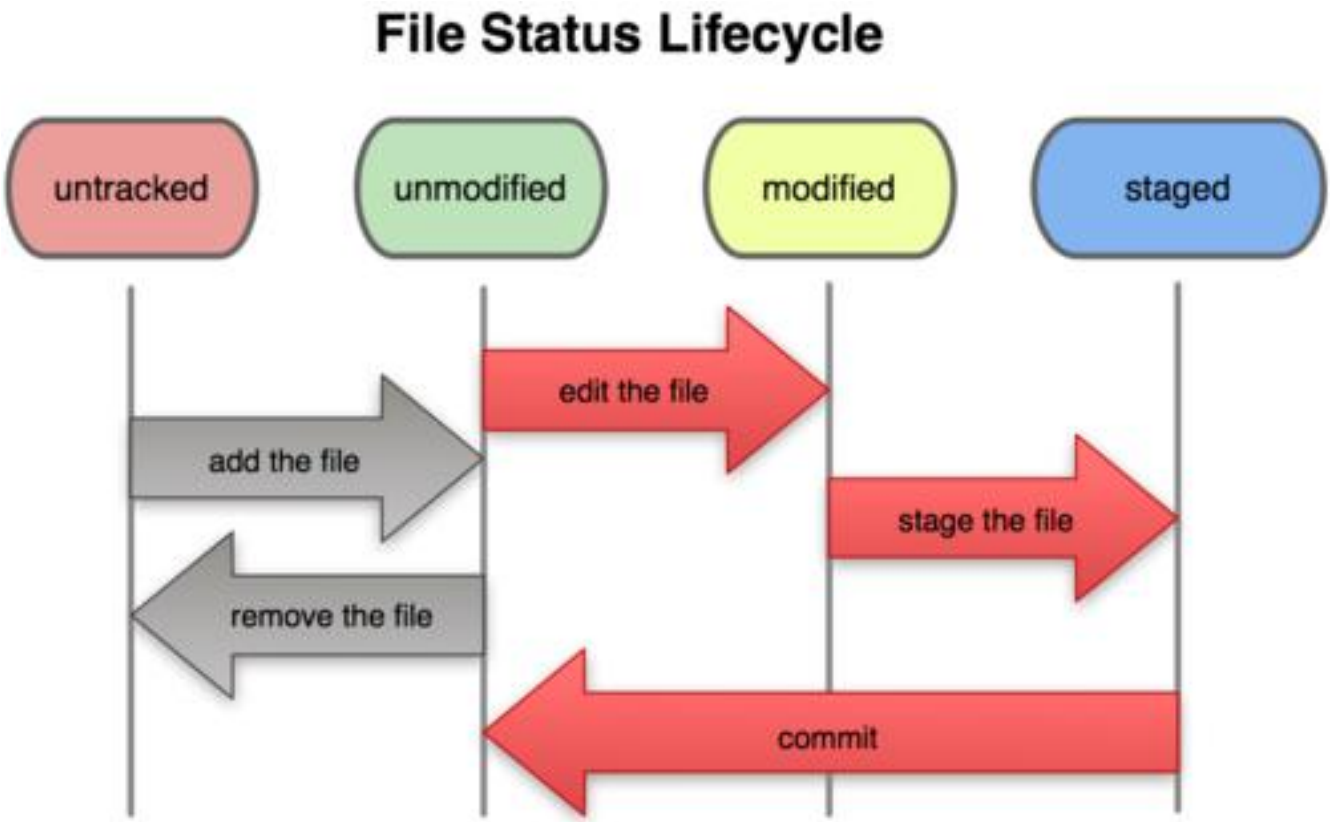


Local Workflow



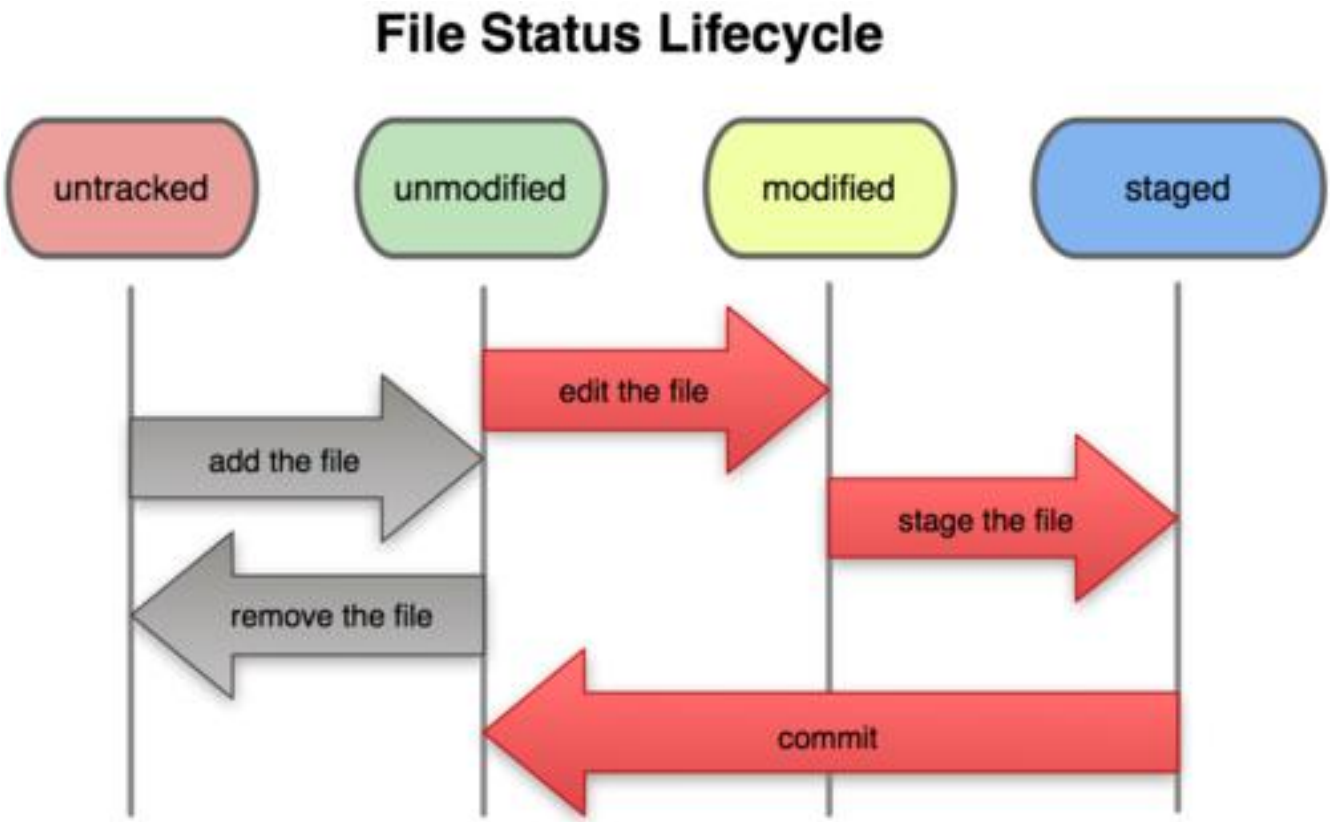
Basic git workflow

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



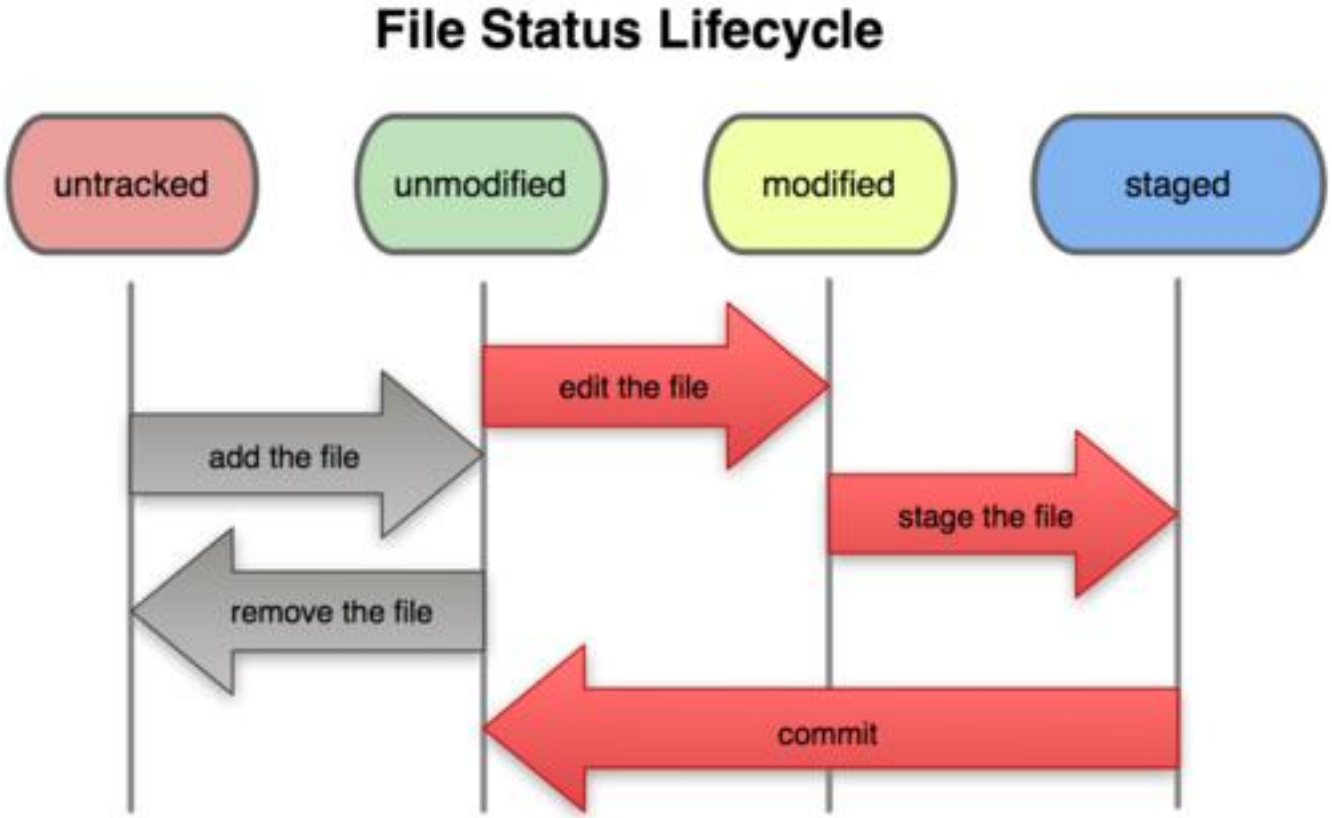
Basic git workflow

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



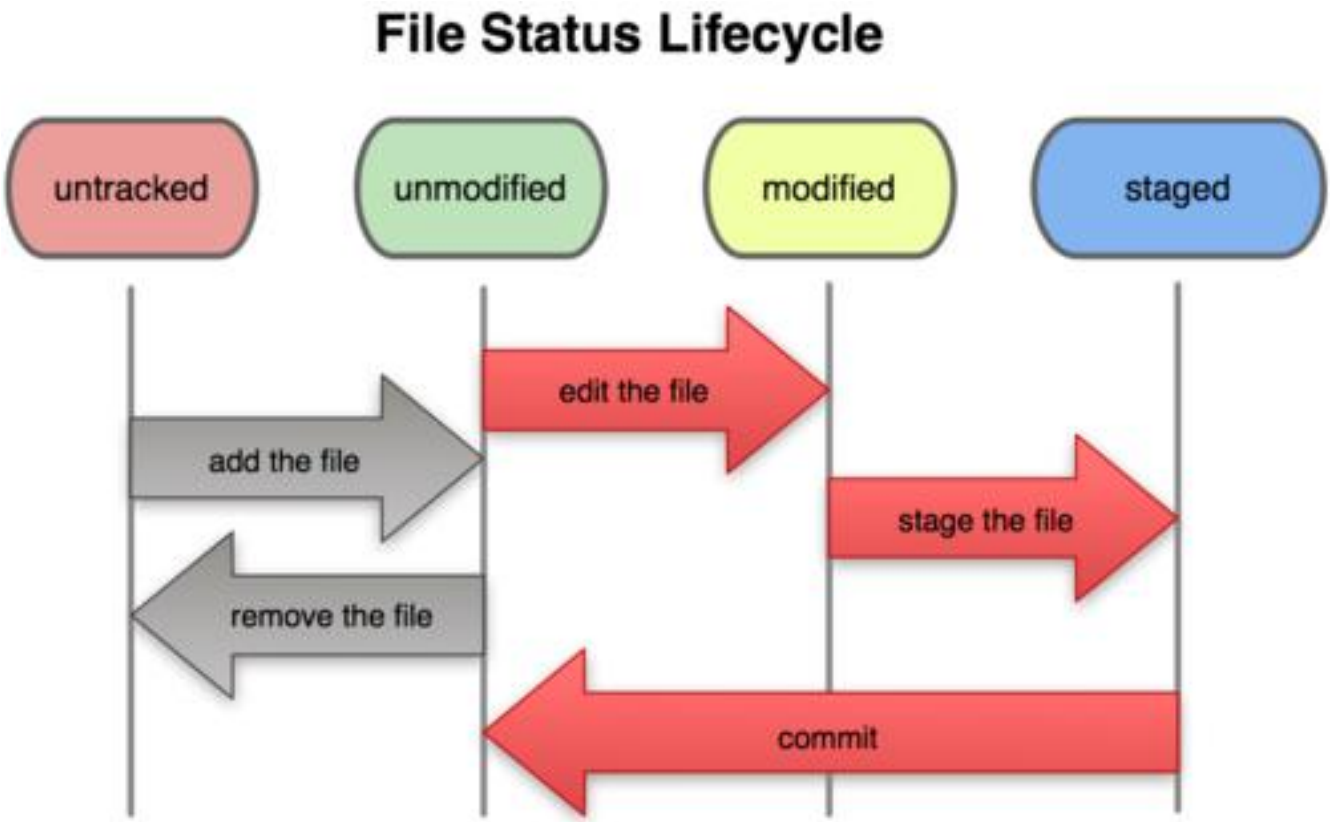
Basic git workflow

- **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



Basic git workflow

- **git commit** commit all stage changes to the local repository



Step 0: Configuring Git

git config --global user.name "User Name“

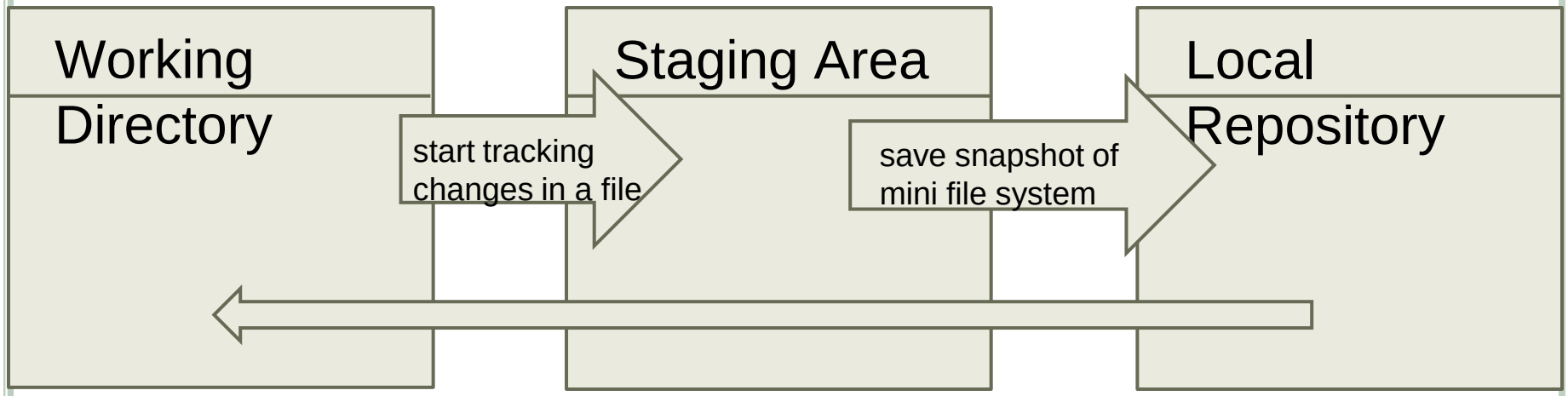
git config --global user.email "user@email.com"

- Selecting the default editor
  - Windows
    - git config --global core.editor "'C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
  - Mac
    - git config --global core.editor "edit -w"
  - Linux
    - git config --global core.editor <your favorite editor here (nano, emacs, etc etc)>

Exercise 1: Our first Repo

- Create a new repository (don't forget to make a new directory!)
- Create a file named README containing the text:

  My first git repository!
- Commit README to your local repository



More git commands

- git checkout <object>
  - Can be a
    - Branch
    - Tag
  - Can also be used to revert changes to a file
- git mv
  - Tell git you're moving a file
- git rm <file>
  - Remove a file
  - Use with –cached to untrack a file

Exercise 2: Dealing with accidents

- Copy 01-shell/data/Bert/audioresult-00215 to your repository and “accidently” add it
- Unstage it from the commit
- “Accidently” overwrite/delete README
- Restore it from the latest version

(Even) more git commands

- git log
  - View the commit history
- git diff
  - View differences
    - Between files
    - Between revisions
    - Between branches

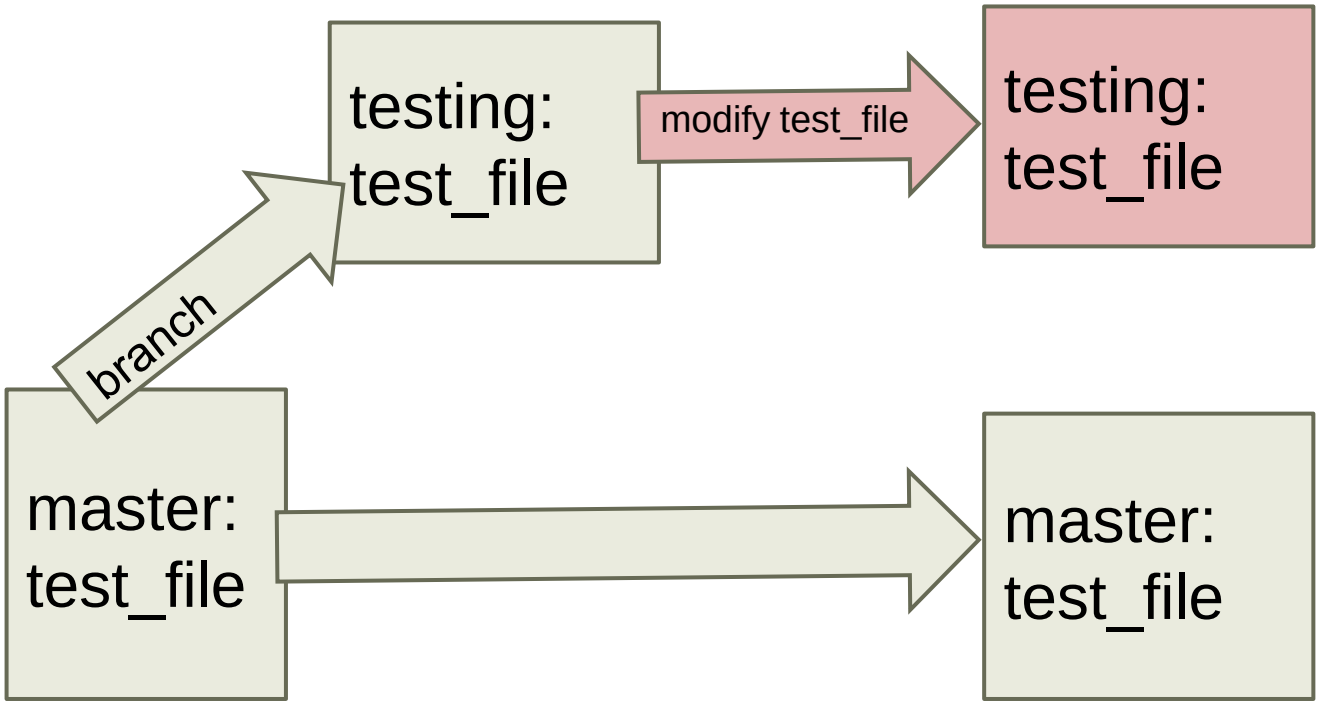
Exercise 3: Seeing the history

- Add some text to README
- Look at the difference between your current README and the latest version in your repo
- Copy the hello.py script from 02-python-variables to your repository
- Commit both files
- View the history for your repository
- Move hello.py to 02_python_solution.py and commit
- Look at the log again
- Delete hello.py from the repository and commit the changes
- Look at the log again to see what's changed

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

Branching



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

Git (branching) commands

- git branch <name>
  - create a branch called <name>
- git branch
  - tells you which branch you are on using *
- git checkout <name>
  - switch to <name> branch

Exercise 6: Branching

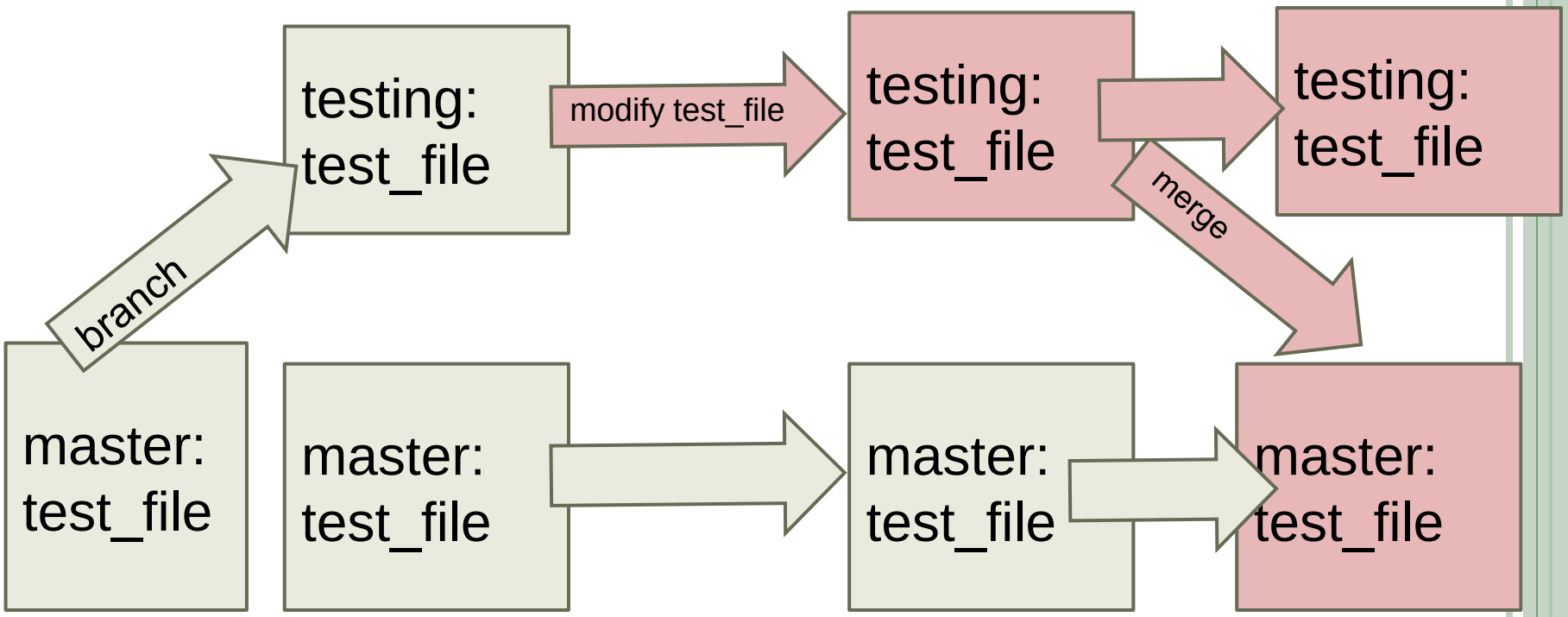
- Create a branch and switch to it
- Make a change to an EXISTING LINE in README
- Commit the change
- Compare the two version of README
  - Via checking out master
  - Extra credit: Via git diff

Merging

- Checkout the branch you want to merge into
- git merge <other branch>
- No conflicts? awesome
- Conflicts?
  - Edit files to resolve conflicts
  - Add
  - Commit
- git branch –d <branch name>
  - Deletes the branch

- git branch -d branch_name - deletes branch

Merging



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

Exercise 7

- Merge the branch you created in exercise 6 in to the master branch
- Resolve the conflict