Version Control Systems

When You're Creating

- Backups
- Ability to see previous versions
- Mark code that works / is stable
- Access to your code from anywhere
- Synchronize changes to code across multiple computers
- Share your code with others.

Doing these things by hand is hard, time consuming, and most often incomplete.

File Versioning Tools

- Most popular tools:
  - Git – may be most popular now
  - Mercurial – a good, simpler alternative to Git
  - Subversion (SVN) – still common, easy to use
  - CVS – older, hard to use, pretty much obsolete
- All have web interfaces for easy access anywhere
- Some editors or IDEs integrate other tools (Xcode, Eclipse)

Integrate into Your Workflow

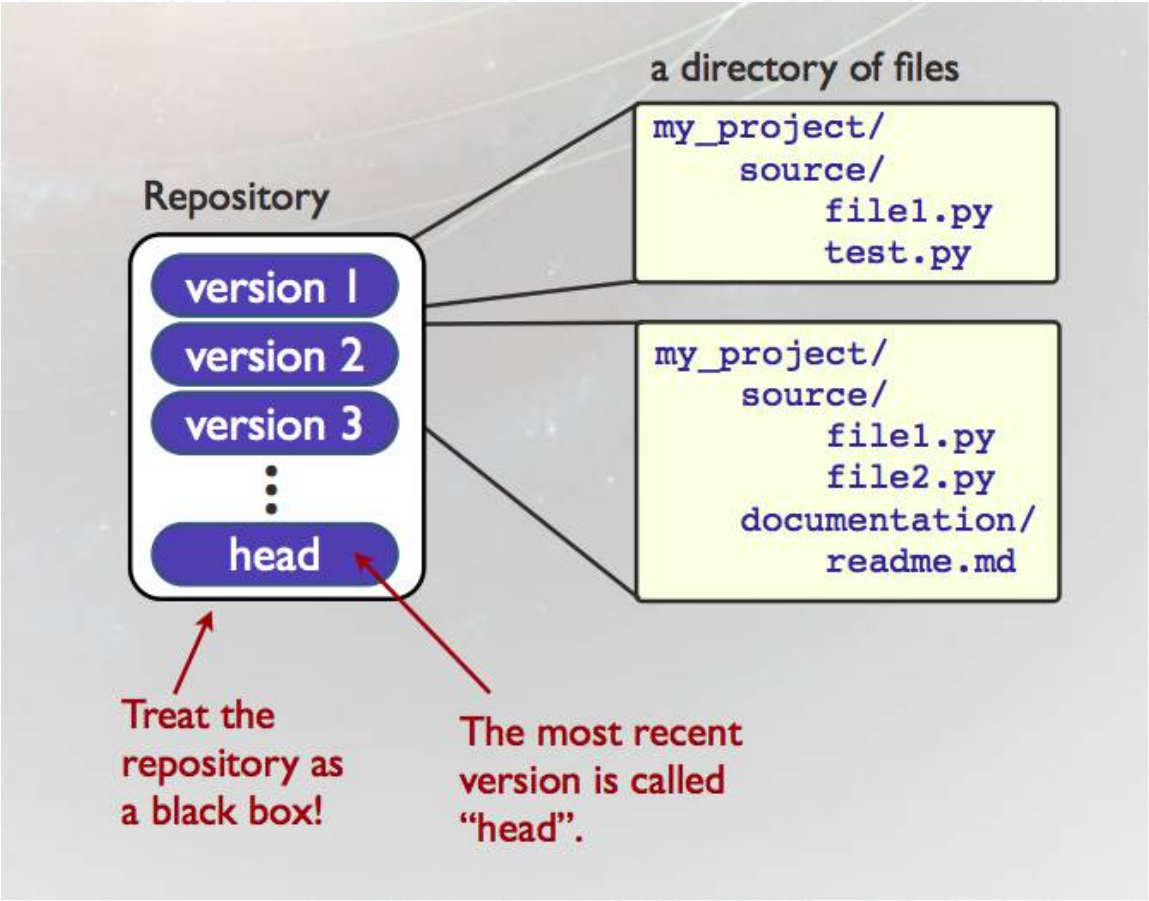
- Many ways to organize your repositories
- Most important: Use It!
- Most any file can be saved into a repository.

(text, images, mp3s,... nearly anything)

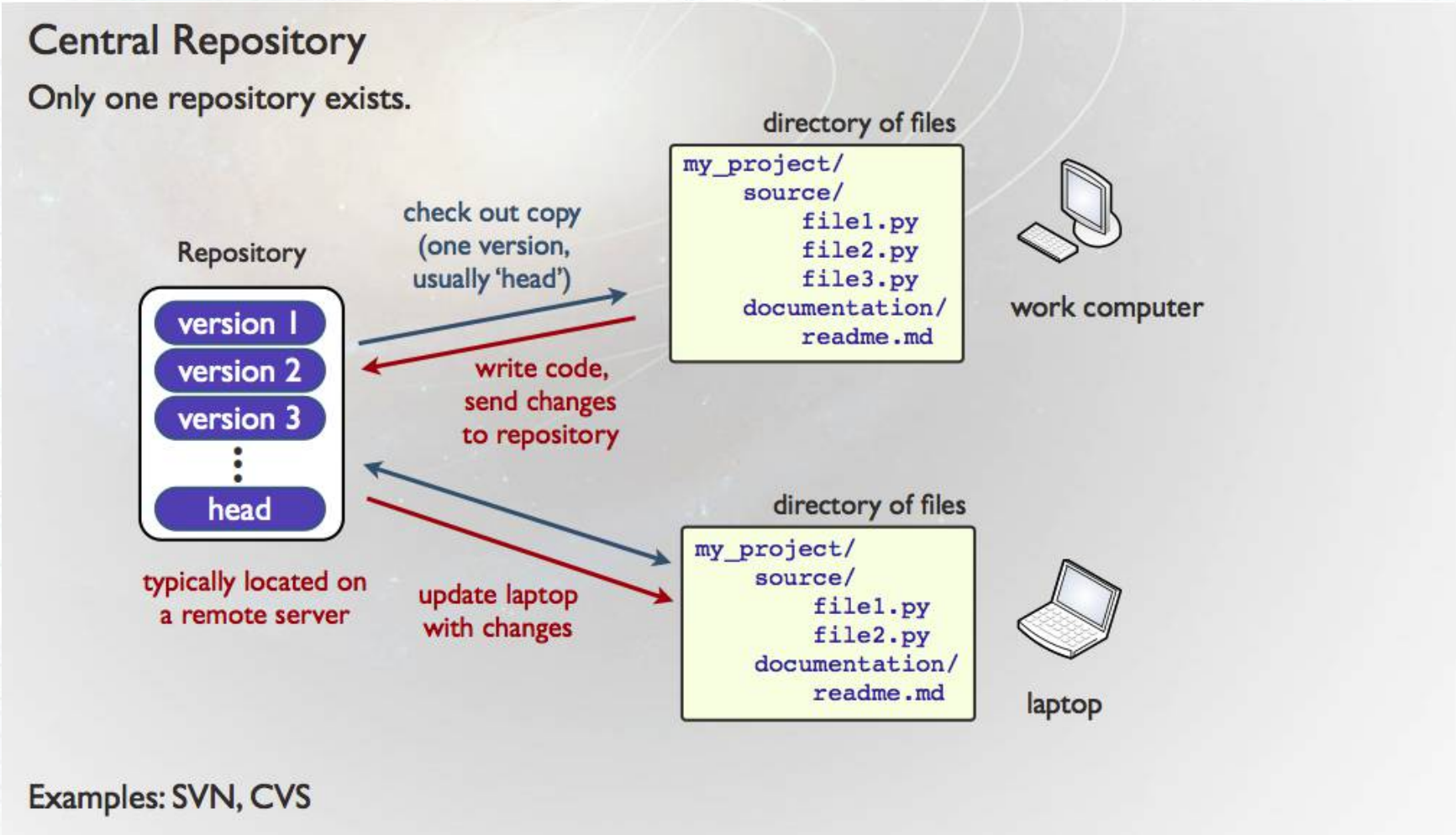
Integrate
Version Control

The Repository

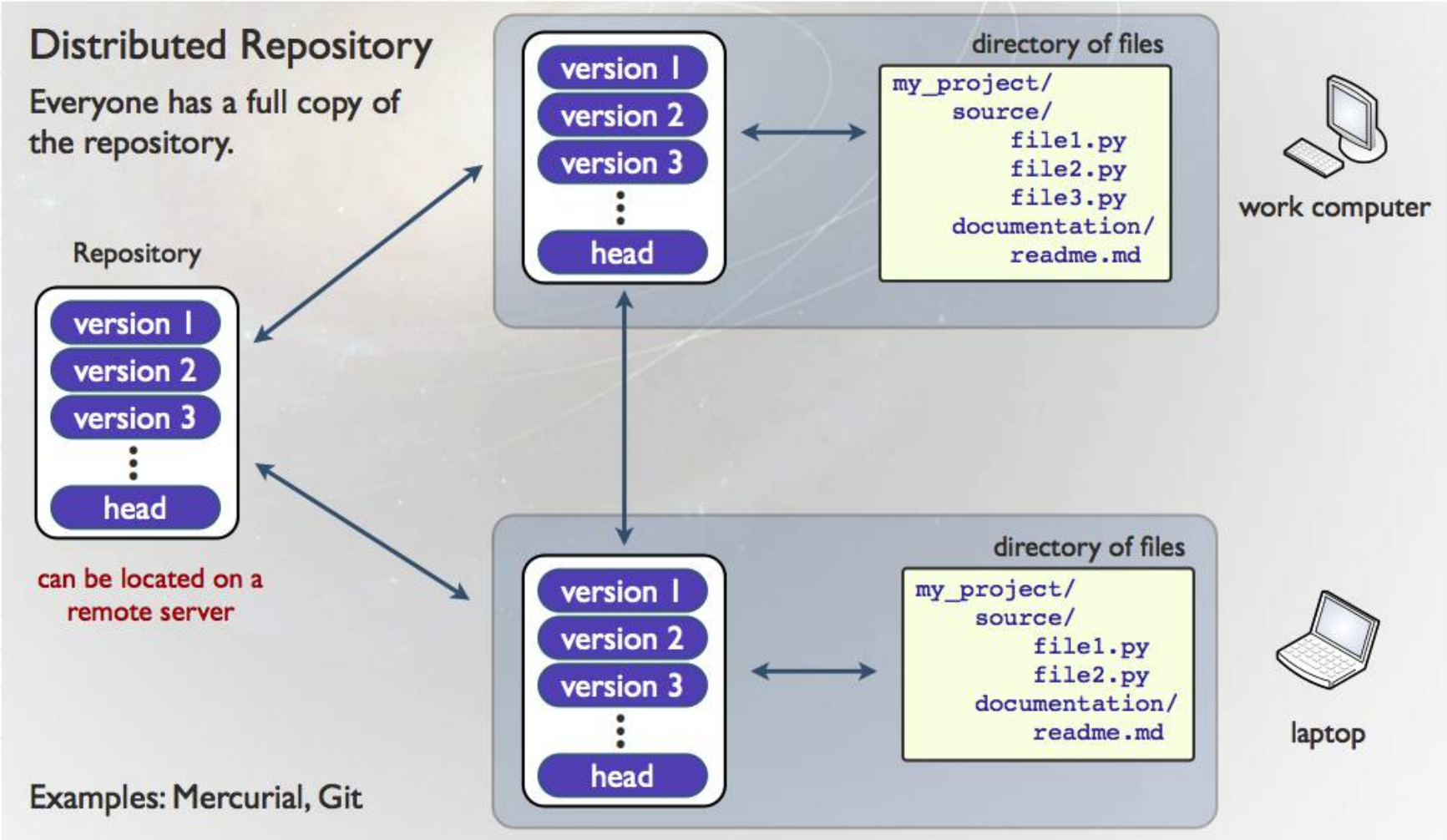
- Where all the versions of the files are stored. Can be local or remote (If local only then it won't be a good backup in the case of computer failure)



Kinds of Repositories



Kinds of Repositories



How Many Repositories?

Typically:

- One for every major project (e.g. abundances of NGC 1261). Includes individual projects but especially collaborative projects.
- A repository to contain bits of code not otherwise contained in another repository

Notes:

- Version Control is for files which are going to change (e.g. code, documents, analysis) not static files (e.g. data)
- Avoid keeping very large data files (images, data) in your repository. Small data files are appropriate.

Basic Structure and Commands

- Each of the arrows on the figure is a git command

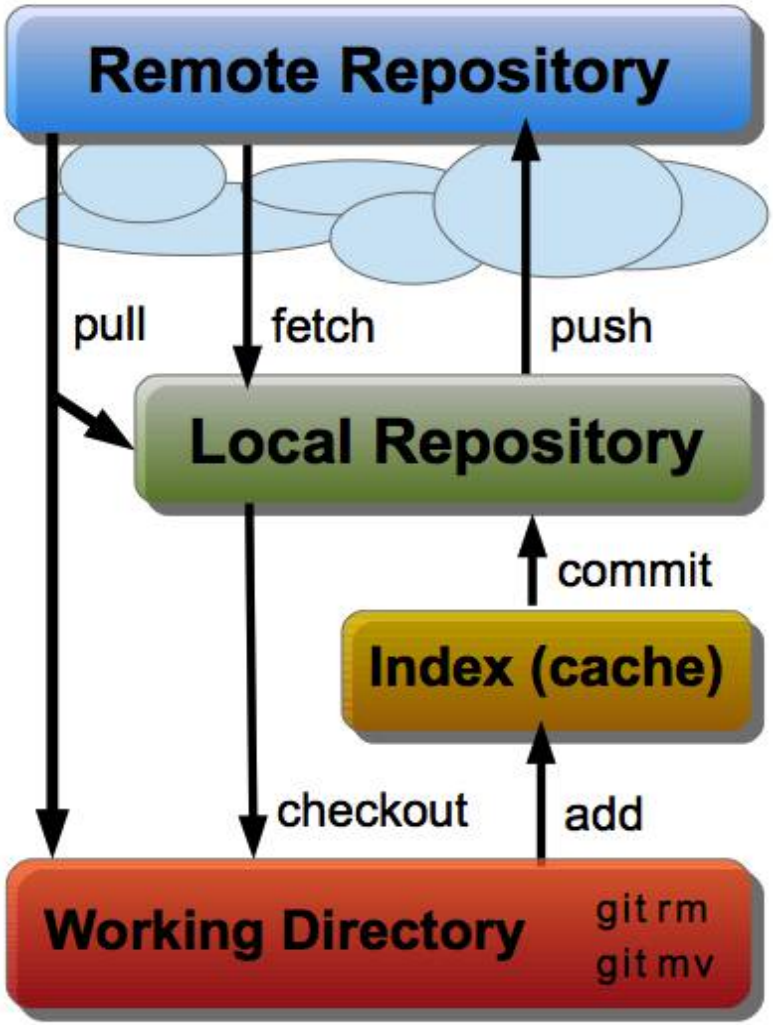
git add <filename>

- commits are like saving your changes to the files
- If you want you can do a commit and add with

git commit -a

- On every commit you should add a message

git commit -a -m “spell checked my presentation”



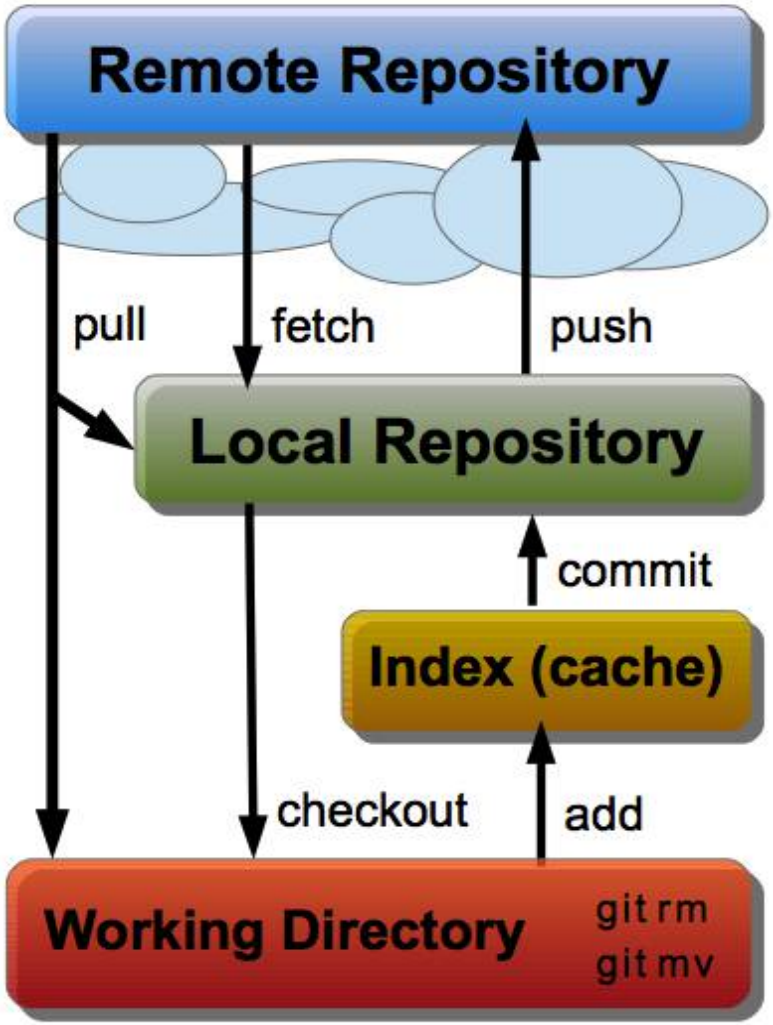
Basic Structure and Commands

- The commands pull and push are used to match up the loca and remote repositories

git push

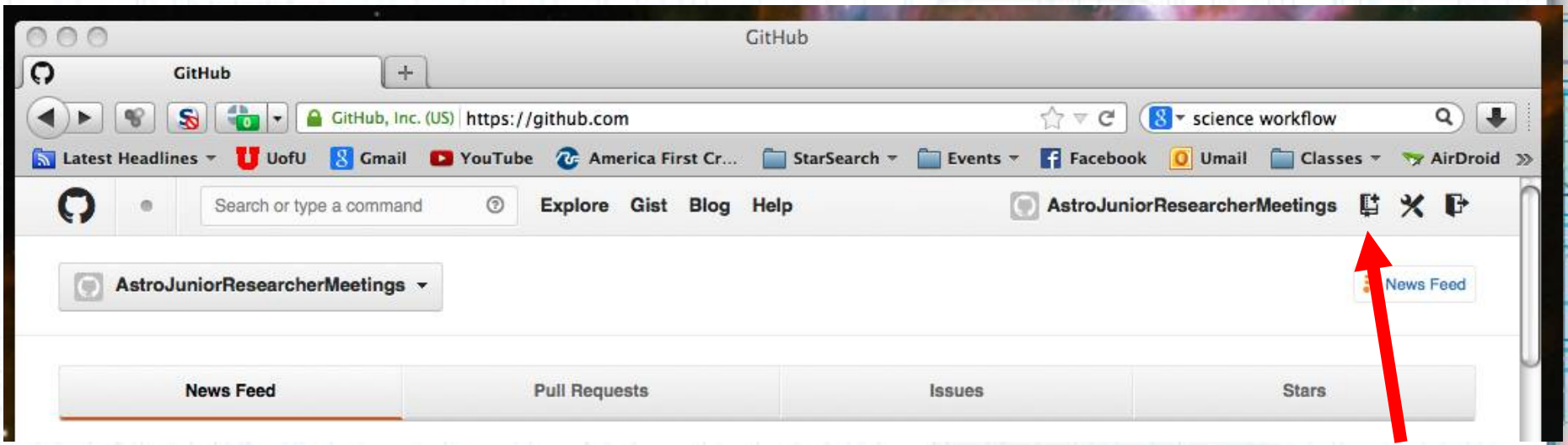
git pull = git fetch & git checkout head

- Merging and combining the differences must be done my hand (software like SourceTree can help)



Create an Online Repository

- Two online repositories which are easy to use
  - GitHub : pubic, good for collaboration
  - BitBucket : private, good for propitiatory projects



And follow the instructions

- The online tool make it easy to create repositories
- From the online you will need to to a clone

Clone Online Repository

- This is a simple process

```
[jrm]$ git clone https://github.com/AstroJuniorResearcherMeetings/test_repositor
y
Cloning into 'test_repository'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
[jrm]$ cd test_repository/
[test_repository]$ git status
# On branch master
nothing to commit (working directory clean)
[test_repository]$
```

- Implicitly you're putting the url into a variable “origin”

```
[jrm]$ git clone --origin origin https://github.com/AstroJuniorResearcherMeeting
s/test_repository
Cloning into 'test_repository'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
[jrm]$
```

Start A Local Repository

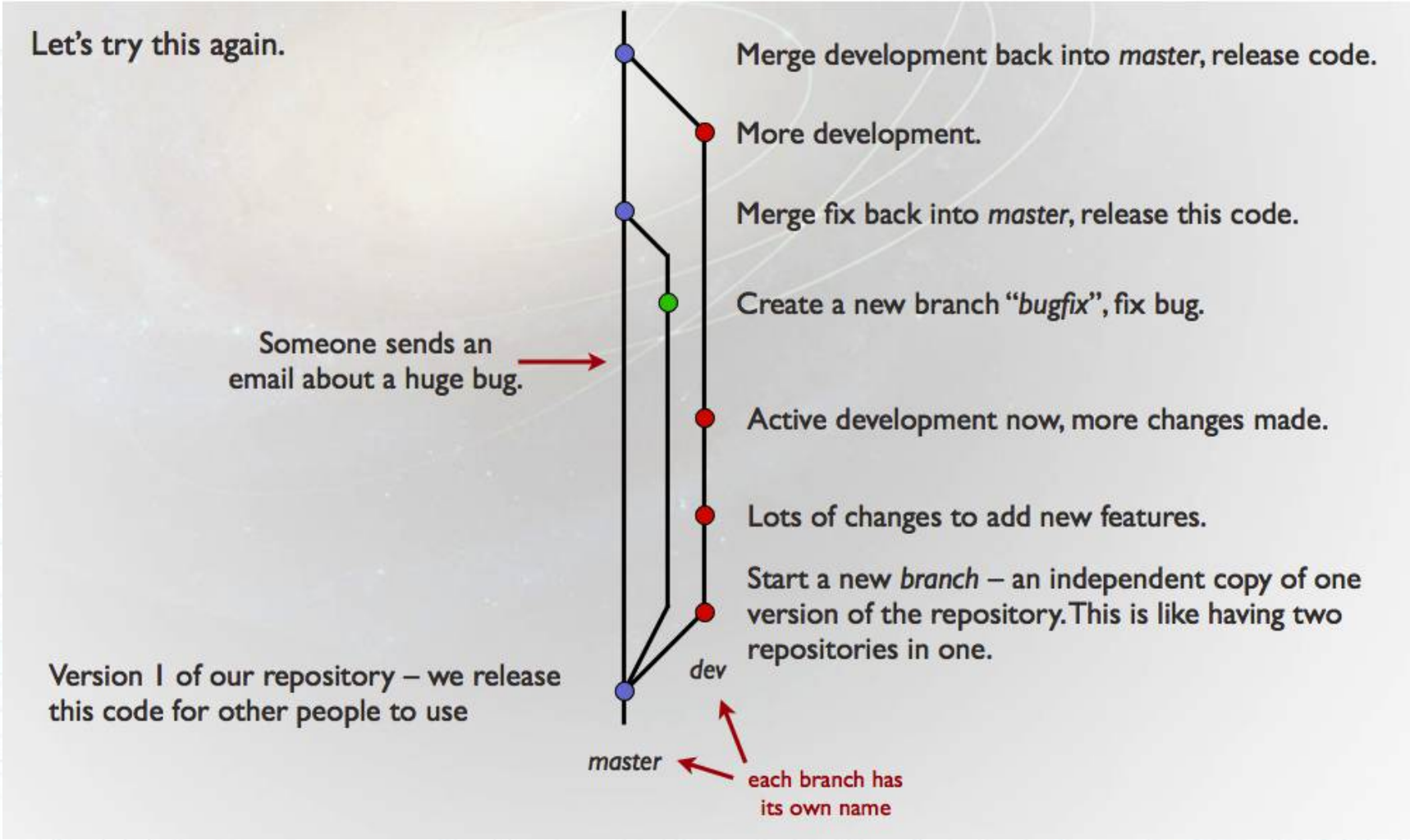
```
/Users/dylangregersen/Desktop/Astrophysics/documents/jrm/test_repo
[test_repo]$ git init
Initialized empty Git repository in /Users/dylangregersen/Desktop/Astrophysics/d
ocuments/jrm/test_repo/.git/
[test_repo]$ touch new_file.txt
[test_repo]$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       new_file.txt
nothing added to commit but untracked files present (use "git add" to track)
[test_repo]$
```

- From a blank directory use git init
- To check the status of a directory use git status
- You can add a remote location

git remote add origin https://github.com/AstroJuniorResearcherMeetings/test_repository.git

git push -u origin master

Branching


Let's try this again.
Merge development back into *master*, release code.
More development.
Merge fix back into *master*, release this code.
Create a new branch "*bugfix*", fix bug.
Someone sends an email about a huge bug.
Active development now, more changes made.
Lots of changes to add new features.
Start a new *branch* – an independent copy of one version of the repository. This is like having two repositories in one.
Version 1 of our repository – we release this code for other people to use
dev
master
each branch has its own name

Deleting and Moving

- If a file is being tracked by git (i.e. you did git add on it) then you need to use a special rm and mv to use those commands because of the tracking

git rm <filename>

git mv <filename> <newfile>

- If you need to remove the file from tracking but not the file system you can use

git rm –cached <filename>

- Then remember to use git commit after

- Note: the GUI git handlers know how to do this so you can just use rm and mv like normal and then handle the git stuff in the GUI

Quick .gitignore

- The file .gitignore list special files which you want to never track. Such as compiled files or emacs saves

```
# emacs
*~

# python
*.pyc

# fortran
*.o
```

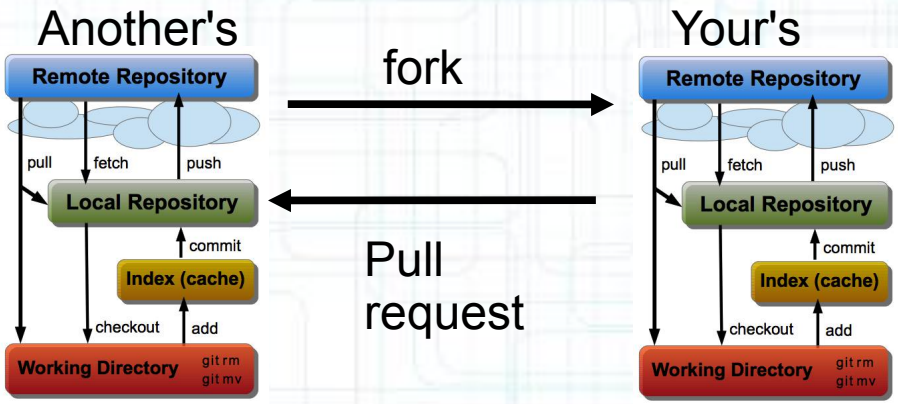
- Read more at: https://help.github.com/articles/ignoring-files

More...

- Go back to points in your code with checkout

https://www.kernel.org/pub/software/scm/git/docs/git-checkout.html

- Fork another repository



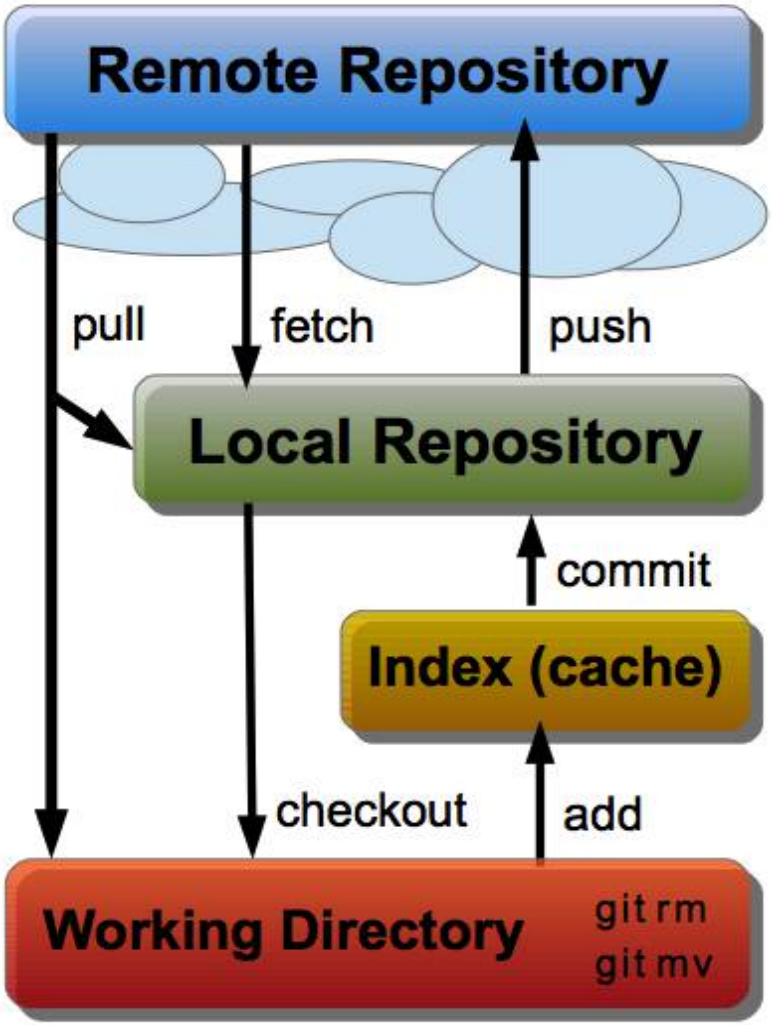
- Use –help to get help on git command

git –help

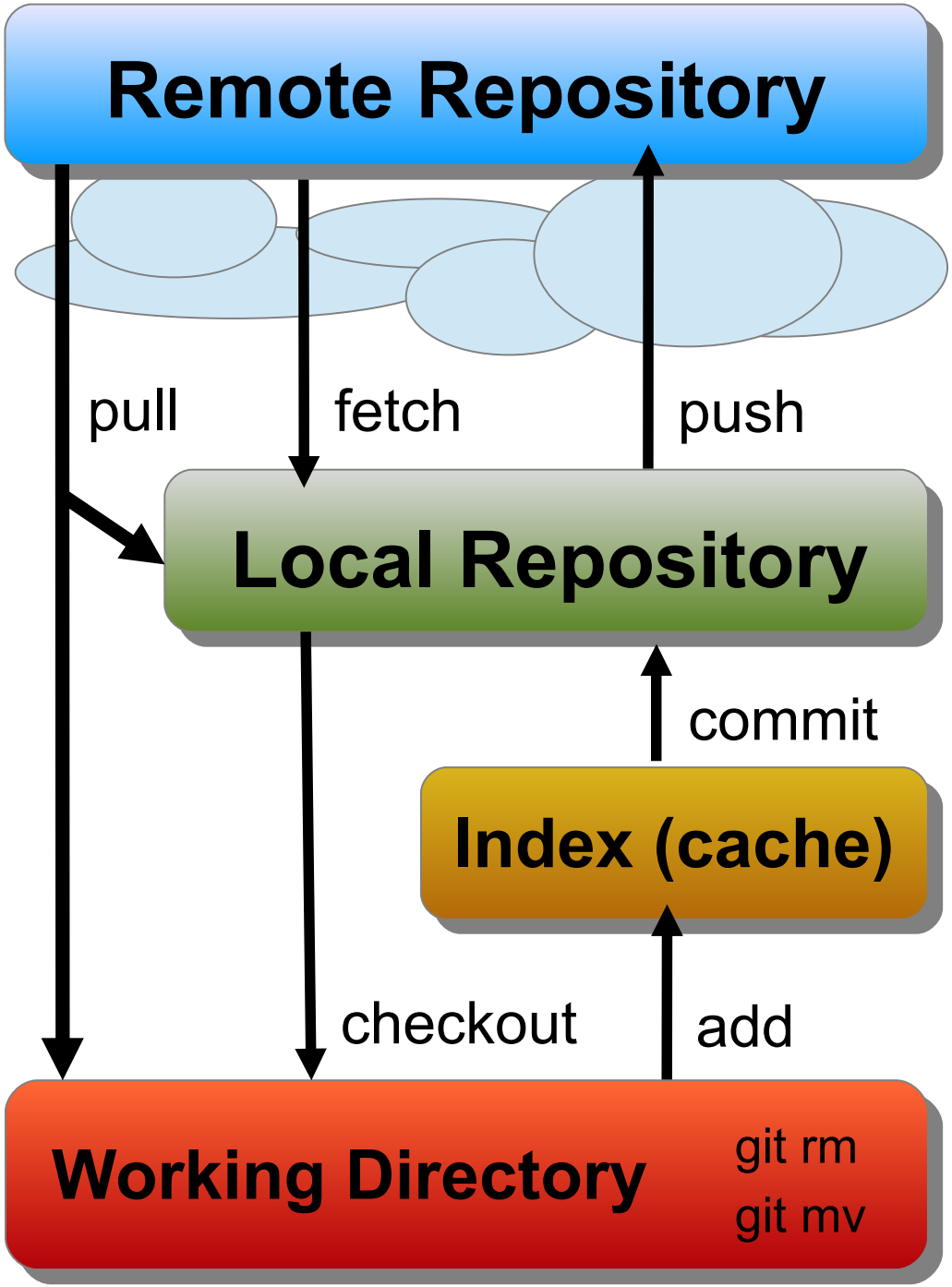
git add –help

- See a log of past commits with git log
- See the status of a git repository git status

Now Get To Work!




```
[test_repository]$ touch new_file.txt
[test_repository]$ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       new_file.txt
nothing added to commit but untracked files present (use "git add" to track)
[test_repository]$ git add new_file.txt
[test_repository]$ git commit -m "added a new file"
[master b3c0b22] added a new file
 0 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 new_file.txt
[test_repository]$ git push
Username for 'https://github.com': astrojuniorresearchermeetings
Password for 'https://astrojuniorresearchermeetings@github.com':
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/AstroJuniorResearcherMeetings/test_repository
   1f51357..b3c0b22  master -> master
[test_repository]$ git pull
Already up-to-date.
[test_repository]$
```


Remote Repository
pull
fetch
push
Local Repository
commit
Index (cache)
checkout
add
Working Directory
git rm
git mv