What Can I Do With Git?

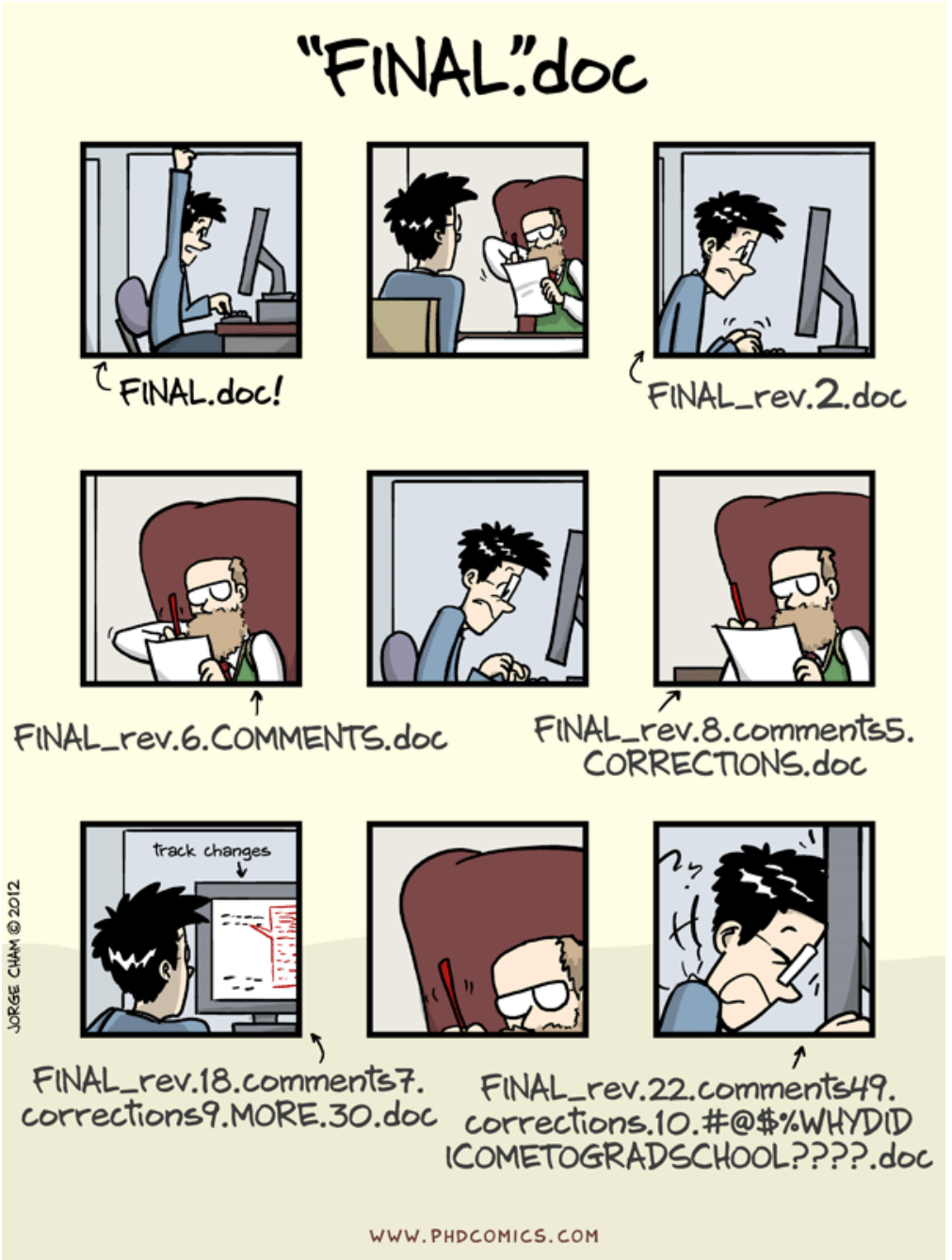
Software carpentry

- Write software for people not computers
- Automate repetitive tasks
- **Use the computer to record history**
- Make incremental changes
- **Use version control**
- Don’t repeat yourself
- Plan for mistakes
- First make it correct, then make it fast
- Document design & purpose not mechanics
- **Conduct code reviews**

Wilson et al. (2014) Best practices for scientific computing. PLoS Biology 12: e1001745

What is version control? What does git do?

- Keeps track of changes made to files
- Lets you go back to old versions
- Only one version in your directory
- Can also be used to…
  - Keep files synced at multiple locations
  - Work with other people
  - Publish your code


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
JORGE CHAM © 2012
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
WWW.PHDCOMICS.COM

More Realistically

script_v1.py
script_v2.py

file-04-05-2015.docx
file-06-08-2016.docx

2015-05-04-file.docx
2016-06-07file_PI.docx
2016-06-08-file.docx

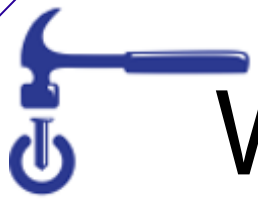
Why Learn Git?

- Reproducible Research
  - For you
    - Your closest collaborator is you 6 months ago, and you don't respond to emails.
  - For publishing your analysis
- For collaboration
  - Code review
  - Working together on a project

Side Note: Git is awesome!

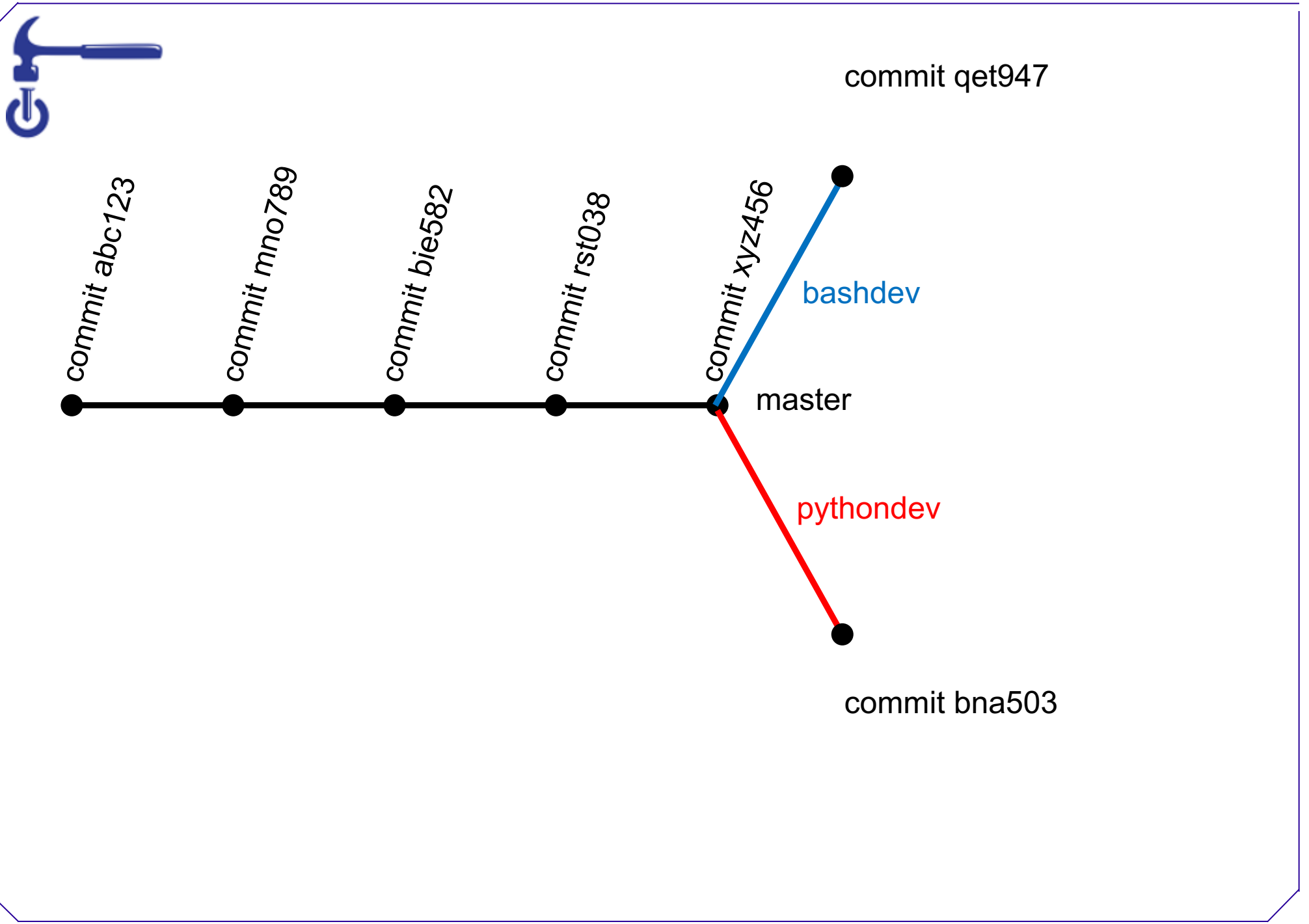
- Lots of flexibility and workflows and pipelines
- Learning the most common ones usages for working by yourself today
- Lots of good google-able resources!

Branches Lesson



Working in Parallel to ‘master’

- You may want to keep the ‘stable’ version as your master, and make changes in parallel
- Then if the changes are good and tested, then merge them into your main/‘master’ branch
- In this case we are going to make two paralell branches from ‘master’ one
- Imagine we are running an analysis and are unsure if python or bash would be faster. Let’s test it! But leave the master branch as is incase neither works out.


commit qet947
commit abc123
commit mno789
commit bie582
commit rst038
commit xyz456
bashdev
master
pythondev
commit bna503

Making pythondev Branch

Making the branch and then checking it out

```
$ git branch pythondev
$ git branch
$ git checkout pythondev
```

‘Write’ Python script/Track It

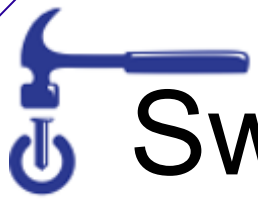
We will only touch the script for this example but imagine you worked a long time on it.

```
$ touch script.py
$ git add script.py
$ git commit –m “wrote and tested
  python script”
```

Checking Our Work

To see that this branch has that commit and that file we will list the directory and look at the log

```
$ ls
$ git log --oneline
```



Switching Back to master Branch

Next we will switch back to the master branch and see that it is unchanged.

```
$ git checkout master
$ git branch
$ ls
$ git log --oneline
```

Making bashdev Branch

Next we will repeat this process working on the bash version of our script. This time we will make and checkout the branch in one step,

```
$ git checkout -b bashdev
$ git branch
$ ls
```

‘Write’ Bash Script & Track It

As before imagine you worked a long time on the script.

```
$ touch script.sh
$ git add script.sh
$ git commit –m “wrote and tested
  bash script”
```

Checking Our Work

To see that this branch has that commit and that file we will list the directory and look at the log

```
$ ls
$ git log --oneline
```

Turns out…

The python one was much more efficient. So lets merge it back into the master branch. You always merge the branch you want into the branch you are in, so first we must switch back to the master branch.

```
$ git checkout master
$ git merge pythondev
```

Deleting the Old Branches

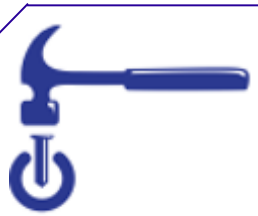
You can keep these old branches but you probably want to clean them up since they might be confusing later.

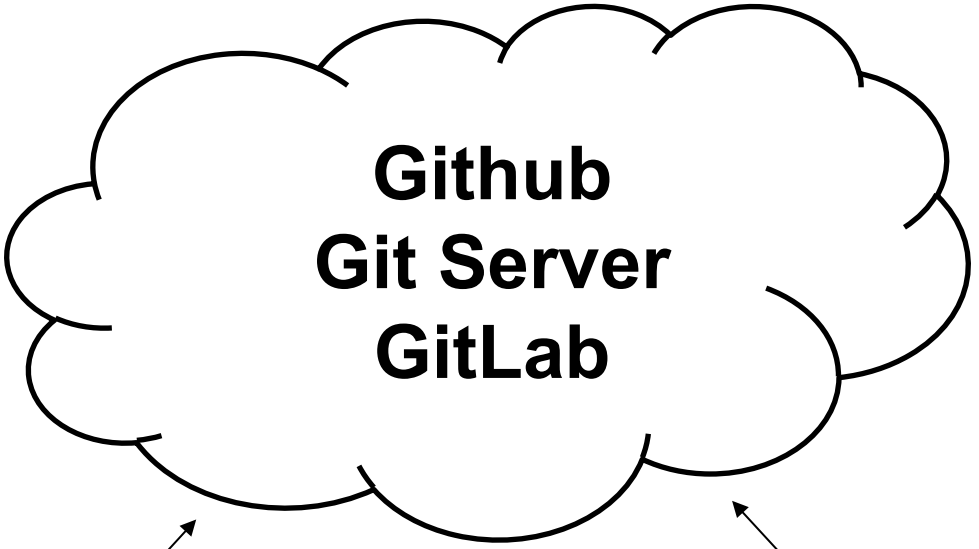
```
$ git branch -d bashdev
```

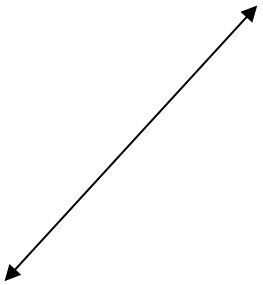
This gives us a warning since we never merged it into our master branch. We will still delete it

```
$ git branch -D bashdev
```

```
$ git branch -d pythondev
```

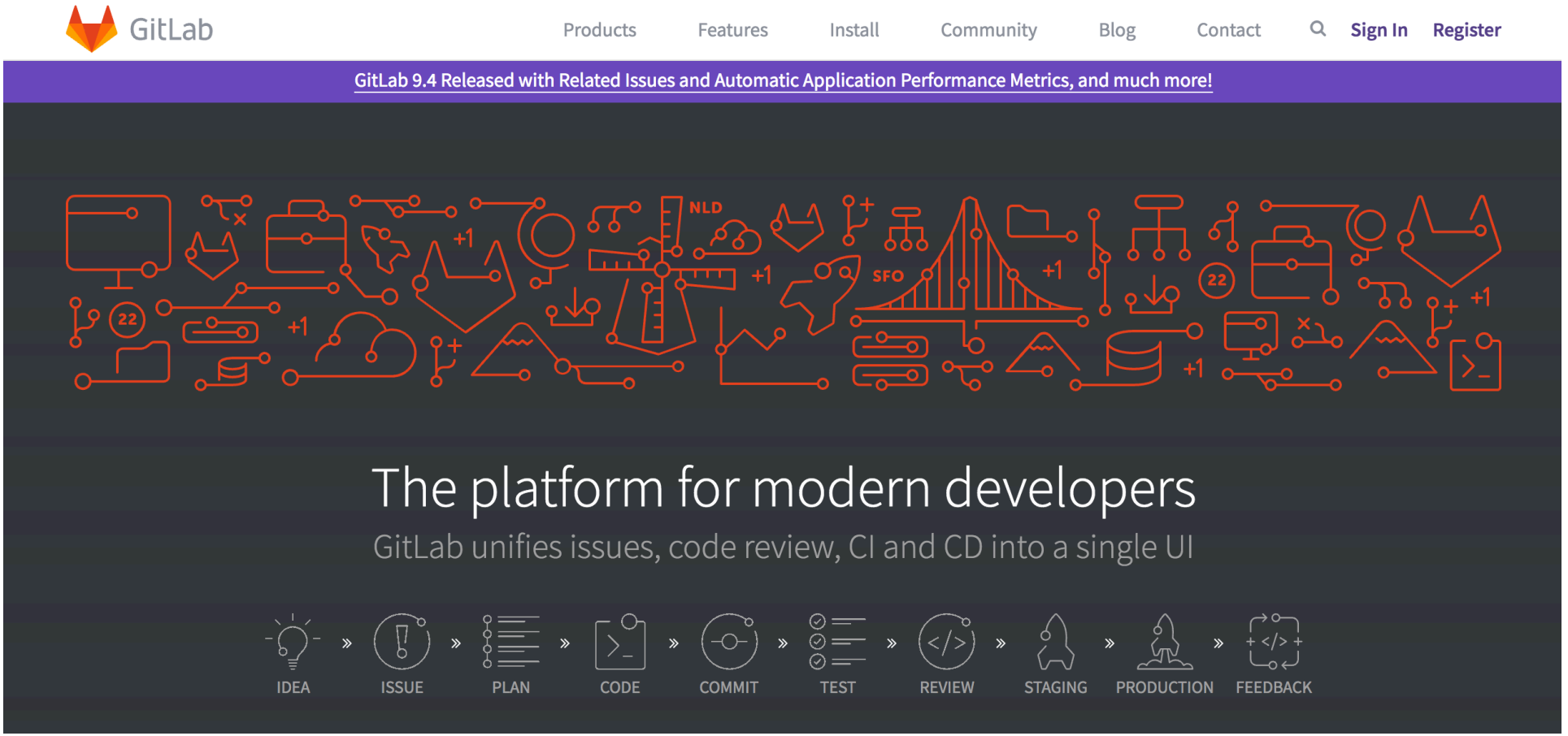



Github
Git Server
GitLab

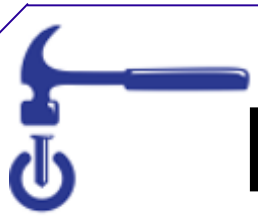


Example Github Repos

- All Software Carpentry Lessons are version controlled with git
  - https://github.com/swcarpentry/git-novice
- The website for this workshop is kept in a git repo
  - https://github.com/sstevens2/2017-08-02-chicago-frb



https://www.linux.com/learn/how-run-your-own-git-server



For those who prefer clicking

There are GUI's for using git!

- Examples:
  - Github Desktop, SourceTree, GitKraken…
- More found at:
  - https://git-scm.com/download/gui/windows
- R users: Rstudio has git integration!

Alt-text:Merge branch 'asdfasjkfdlas/alkdjf' into sdkjfls-final

https://xkcd.com/1296/

Alt-text: If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.

https://xkcd.com/1597/