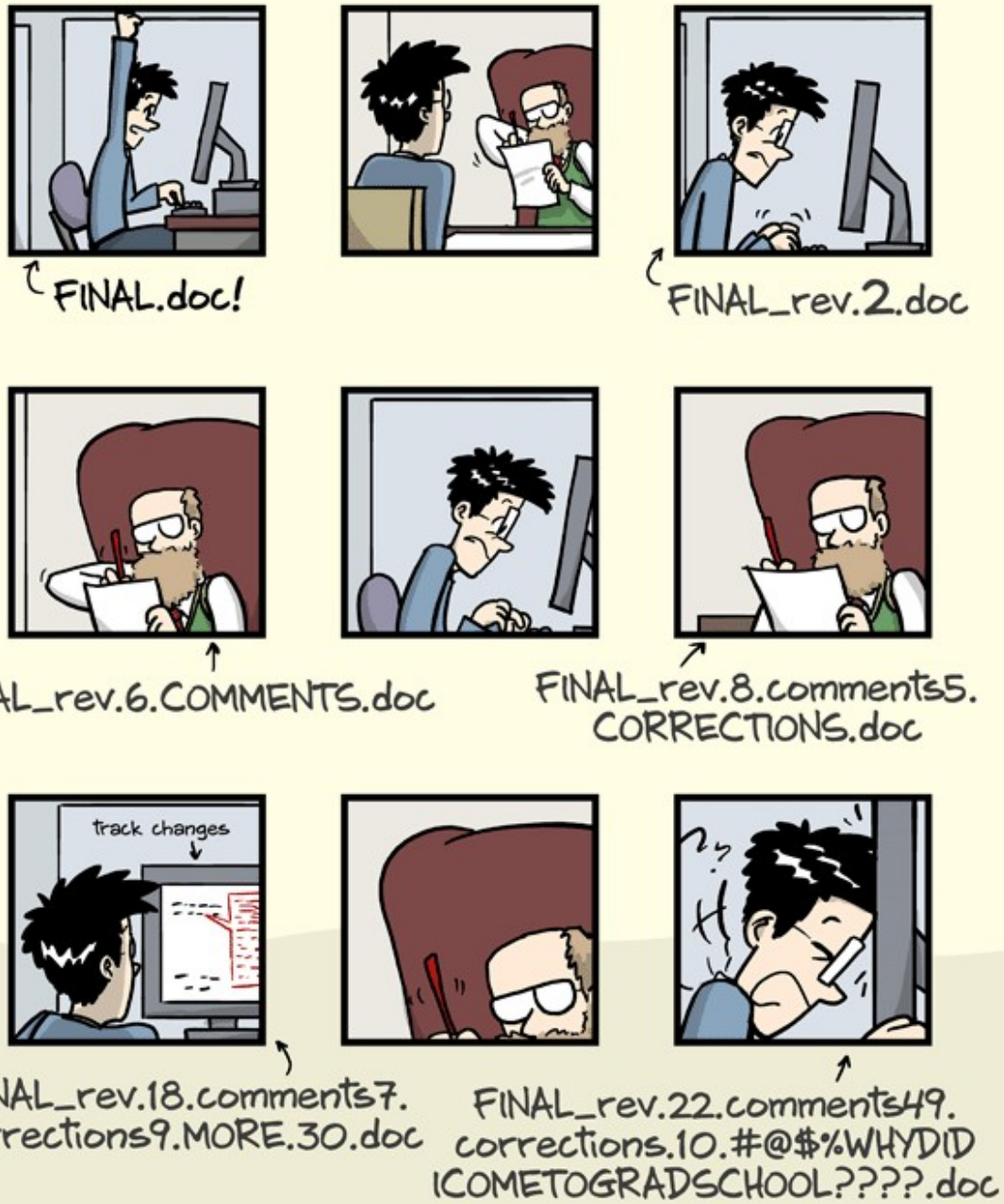
Introduction to git

Learning Outcomes

On successful completion of this module, students should be able to:

- **Use Docker, (bio)conda, and git to create reproducible analysis environments and generate reproducible results**
- Use the Linux command line environment including access/use of a High-Performance Compute (HPC) cluster
- Write Rmarkdown documents to generate reproducible research reports
- Analyse gene expression microarrays in order to identify differentially expressed genes, enriched GO terms, pathways, and gene sets
- Develop simple Shiny applications

"FINAL".doc



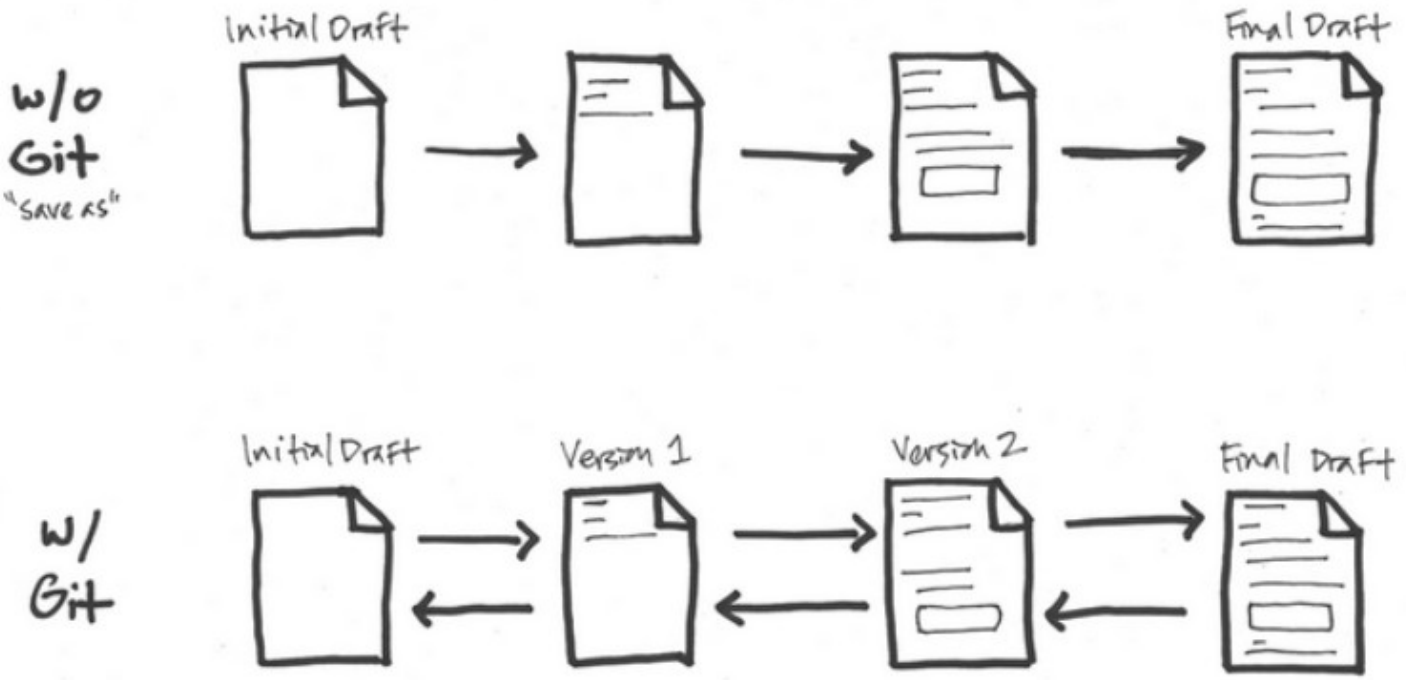
WWW.PHDCOMICS.COM

What is version control?

- Methodology in software development that ensures all changes to a software project (and code) are tracked in time.
  - Advantages
    - you can revert back to specific ‘versions’ of your code
    - collaboration becomes practical, as specific changes and associated contributors are tracked

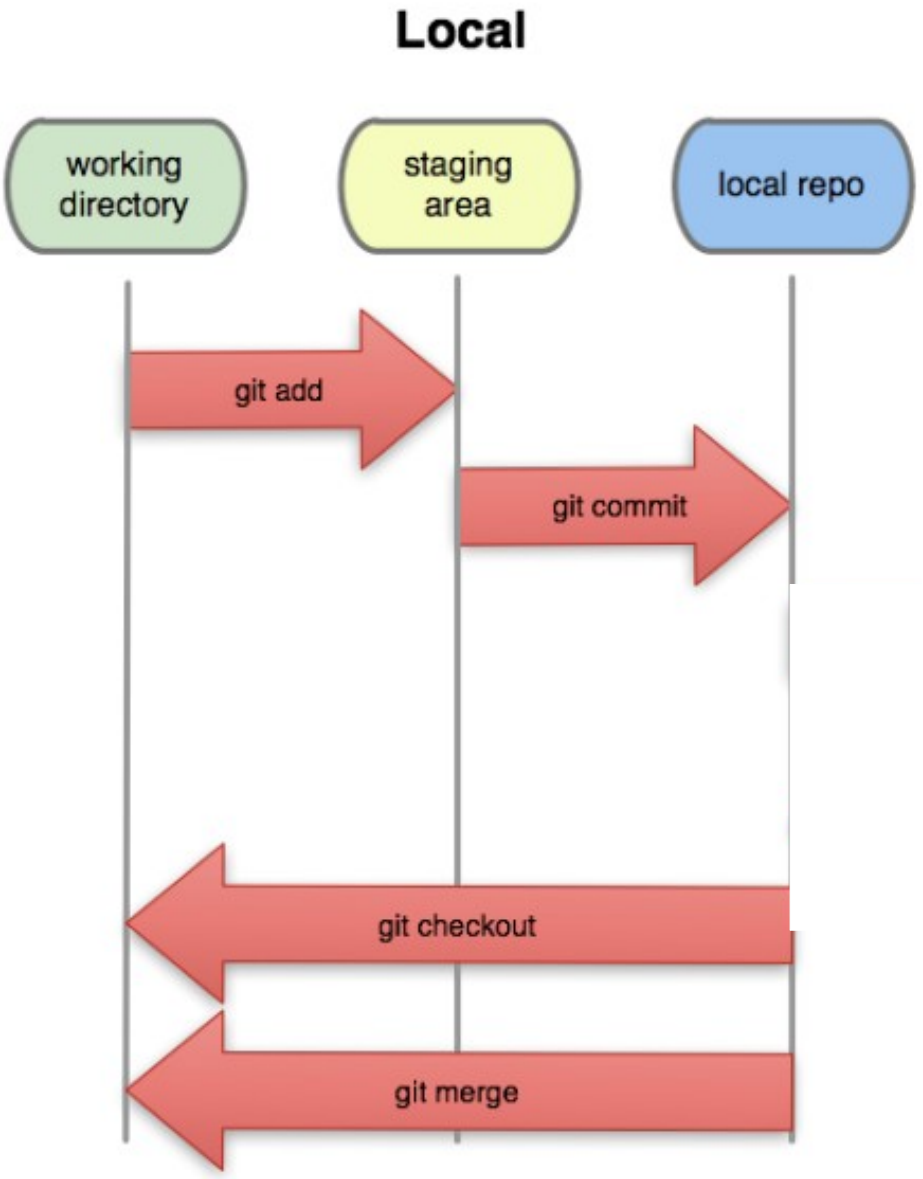
- The most commonly used version control systems is Git

Version control with Git



'Edits' etc. are easily forgotten - with *git* all changes are logged

Version control with Git



What is Github?

- Github (github.com) is a web-based hosting service for version control using Git.

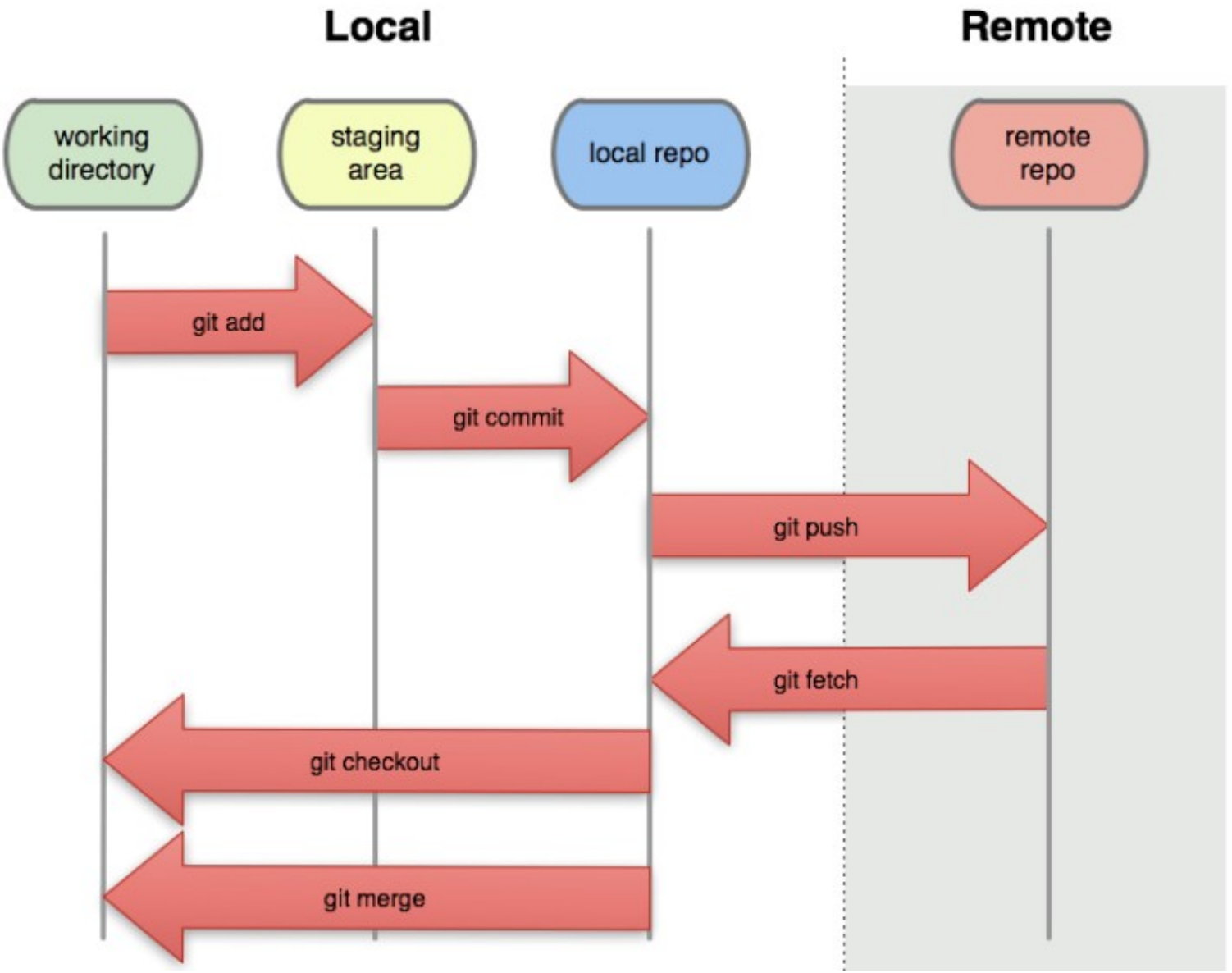
- It offers free and private accounts it forms the backbone for the open source movement.

- You will need to have a Github account to submit you MSc project

- Whilst it is primarily a means of managing & curating software

  projects, it is also an excellent 'shop front'. (https://github.com/BarryDigby)

Version control with Git and github



How do I Install Git?

On Ubuntu (if not already installed):

```
sudo apt-get update
sudo apt-get install git -y
```

For Mac/Windows look here:

https://github.com/git-guides/install-git

Getting Started

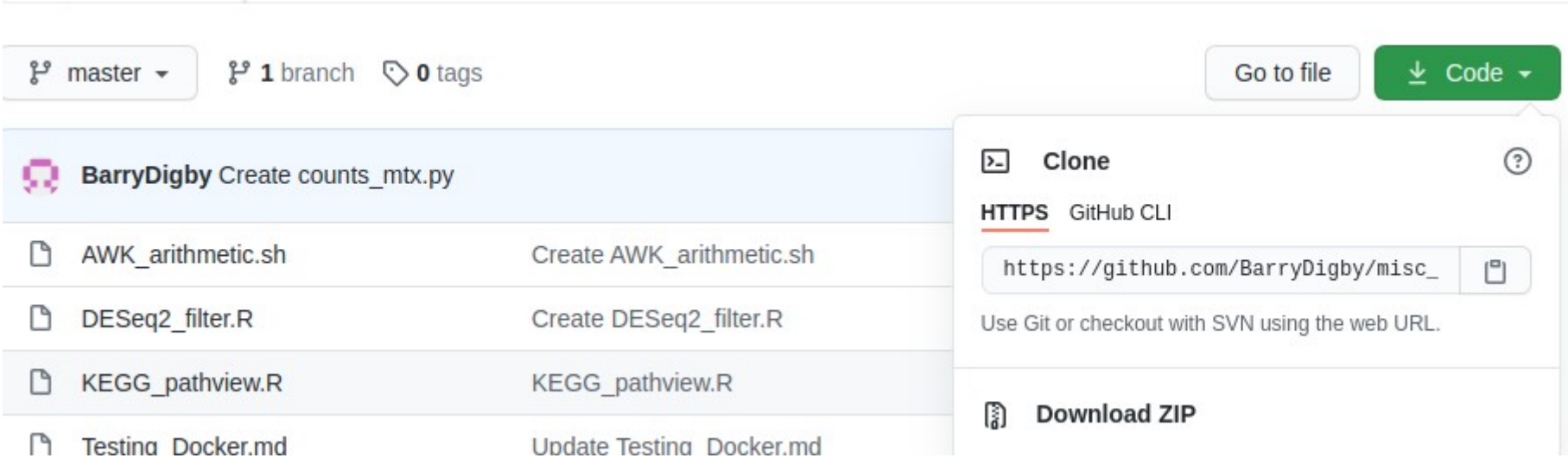
Before we get going, you're going to need to provide

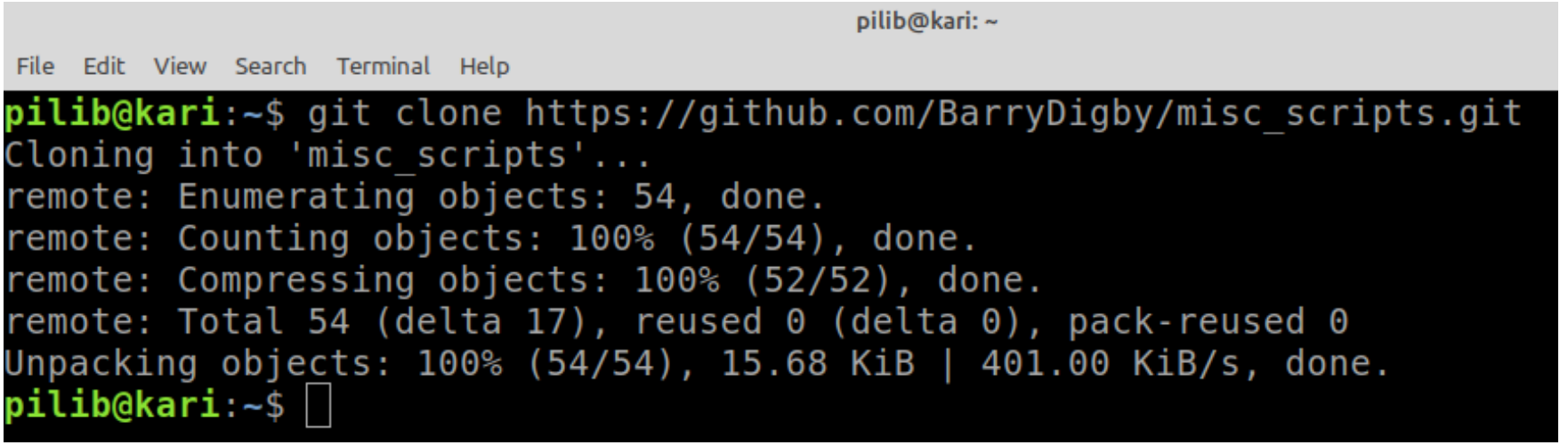
- a user name
  - doesn't necessarily have to be your actual name
- an e-mail address
  - suggest you use an e-mail that isn't your NUIG one (you could lose it!)

```
git config --global user.name "First Last"
git config --global user.email "example@example.com"
```

Creating a Local Repository

Option 1 - *clone* an existing one from github using **git clone**





Creating a Local Repository

Option 2 – create a new project directory and put it under version control using **git init**

We're going to use the example from the paper on Blackboard

PLOS | COMPUTATIONAL BIOLOGY

EDUCATION

A Quick Introduction to Version Control with Git and GitHub

John D. Blischak[1]*, Emily R. Davenport[2], Greg Wilson[3]

1 Committee on Genetics, Genomics, and Systems Biology, University of Chicago, Chicago, Illinois, United States of America, **2** Department of Molecular Biology and Genetics, Cornell University, Ithaca, New York, United States of America, **3** Software Carpentry Foundation, Toronto, Ontario, Canada

* jdblischak@gmail.com

"This is part of the PLOS Computational Biology Education collection."

Introduction to Version Control

Many scientists write code as part of their research. Just as experiments are logged in laboratory notebooks, it is important to document the code you use for analysis. However, a few key problems can arise when iteratively developing code that make it difficult to document and track

Creating a Local Repository

Hypothetical situation

Imagine that, as part of your thesis, you are studying the transcription factor CTCF, and you want to identify high-confidence binding sites in kidney epithelial cells. To do this, you will utilize publicly available ChIP-seq data produced by the ENCODE consortium [3]. ChIP-seq is a method for finding the sites in the genome where a transcription factor is bound, and these sites are referred to as peaks [4]. `process.sh` *downloads the ENCODE CTCF ChIP-seq data from multiple types of kidney samples and calls peaks (S1 Data);* `clean.py` *filters peaks with a fold change cutoff and merges peaks from the different kidney samples (S2 Data); and* `analyze.R` *creates diagnostic plots on the length of the peaks and their distribution across the genome (S3 Data).*

Creating a Local Repository

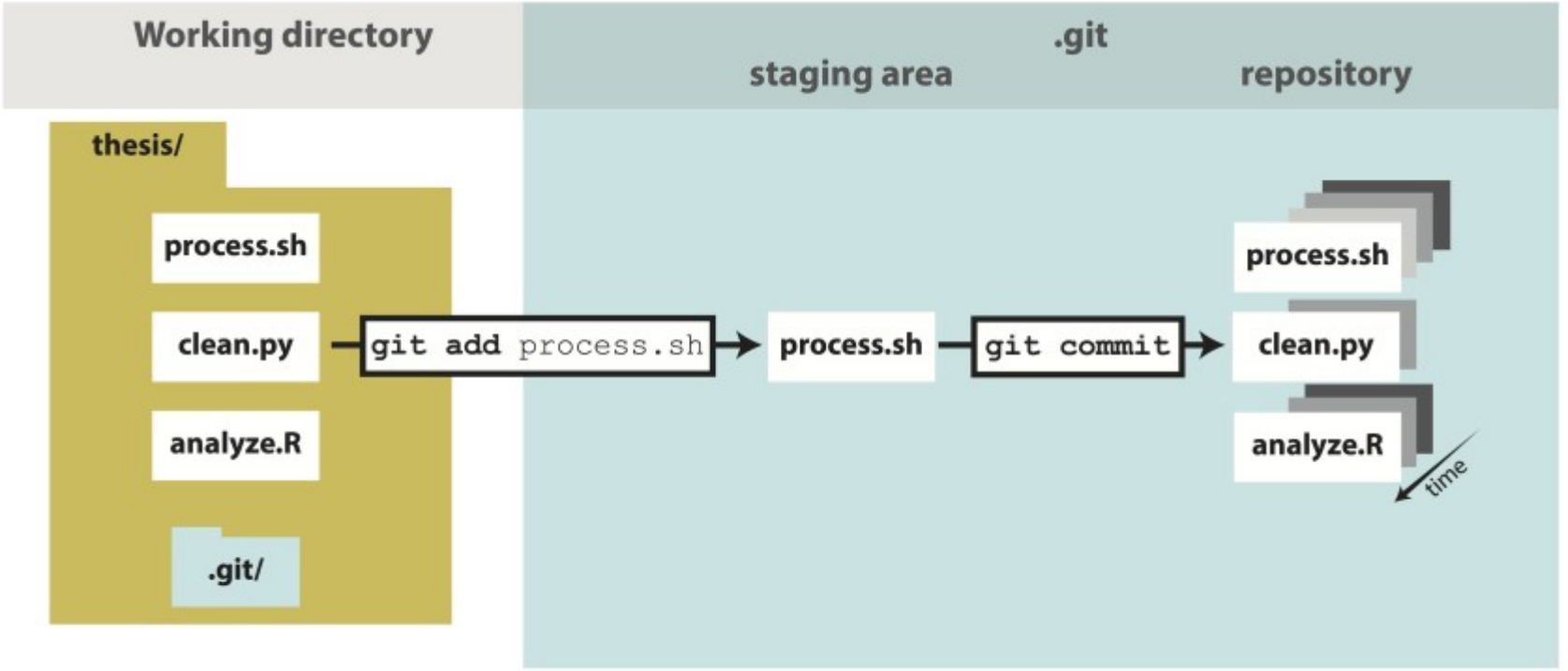
Download and untar (tar xvf codes.tar) the scripts from Blackboard then move the three files to a new directory call thesis

cd into the thesis directory and use nano or any other text editor to examine the files. Then type **git init** to initialise your new repository

Now type **ls -la**, you should see a new **.git** folder. This is where Git will track all of the changes you make to the files

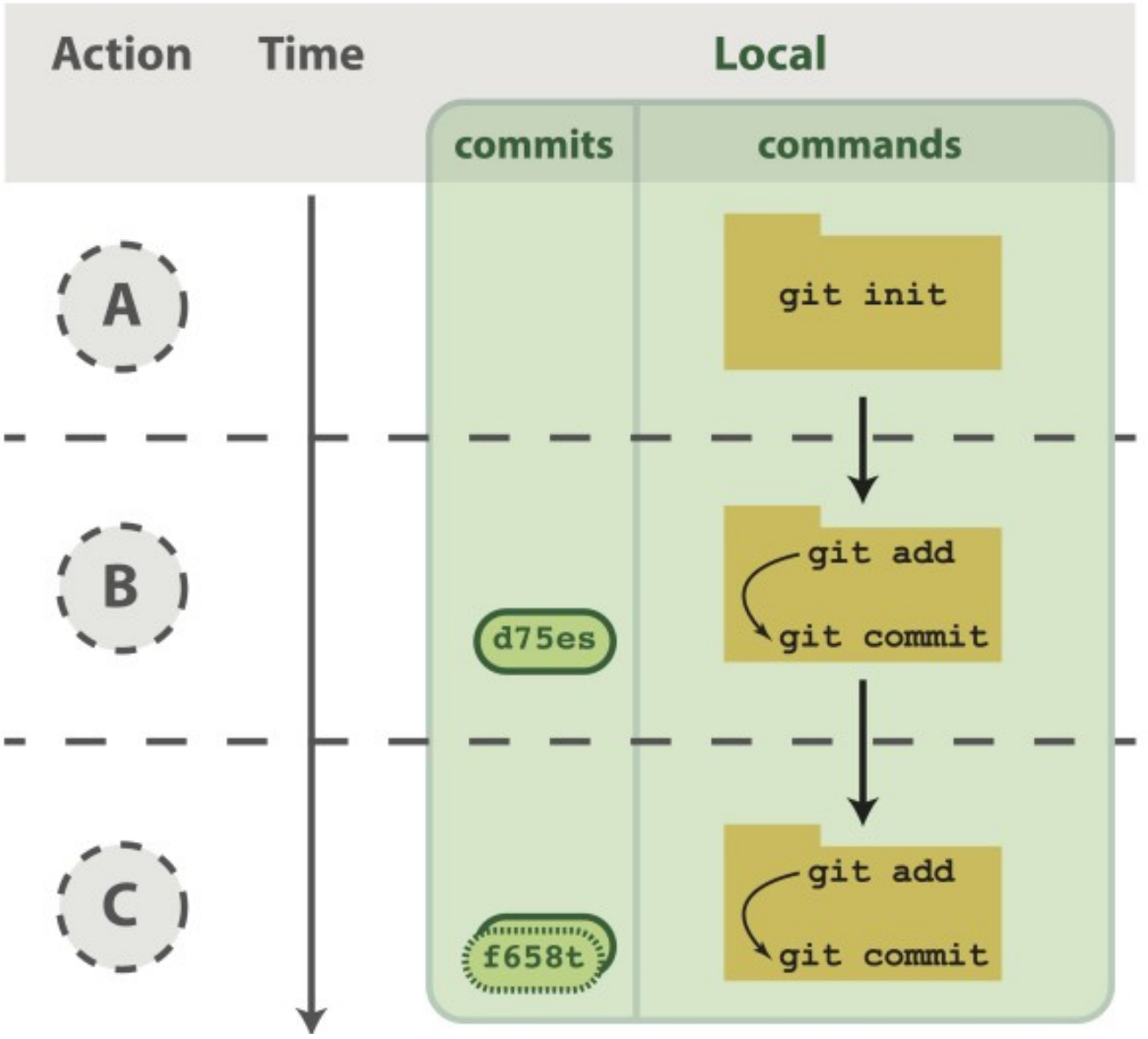
Next, type **git status** to see what the current status of the repository is

Staging & Committing Files




```
git add process.sh
git status
git add clean.py analyze.R
git commit -m "Add initial version of thesis code."
git log
```

How Git Timestamps Commits



Checking What Changes Were Made

We can use **git diff** to see what changes were made to the files that have not yet been staged. **HEAD** points to the latest commit, you can also use e.g. **git diff HEAD~1** or **git diff HEAD~2** to see differences to previous versions

```
pilib@kari:~/Desktop/git$ git diff
diff --git a/process.sh b/process.sh
index 71b00b9..0b3aaa5 100644
--- a/process.sh
+++ b/process.sh
@@ -1,7 +1,7 @@
 #!/bin/bash

 # Download bam files and call peaks.
-
+#Adding a comment here
 ENCODE=https://www.encodeproject.org/files
 OUTDIR=../data
 MACS2="macs2 callpeak -f BAM -g hs"
```

Committing or Discarding Changes

If we want to keep the changes in the working directory that we see with **git diff**, we can **git add** and then **git commit** the file. If we don't want to keep them, we can type

git restore process.sh

to restore the file to the latest commit (pre-modification) version. If we had already staged the modified S1_data.sh file with git add but then decided not to keep the change, we can use

git restore - - staged process.sh

Note that this is different from the paper as there has been a Git update since the article was published

See: https://medium.com/blue-harvest-tech-blog/git-2-23-0-forget-about-checkout-and-switch-to-restore-ac2682b737b3

Restoring a Previous Version of a File

If we realise after committing a file that we need to restore a previous version, we can use

git restore - - source commitID process.sh

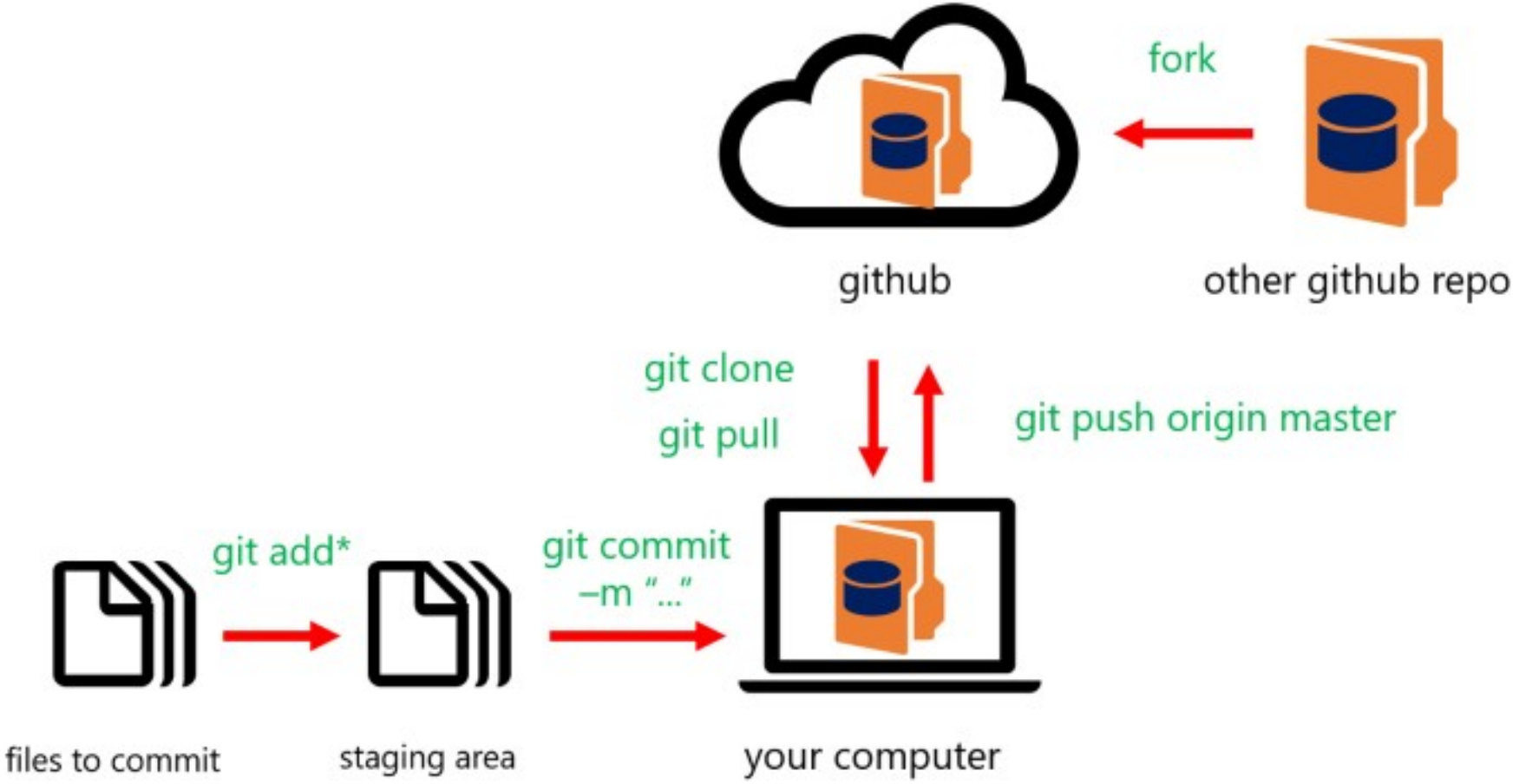
to restore the file to a specific commit (shortened versions of commit IDs can be found using **git log --oneline**)

Branching Out

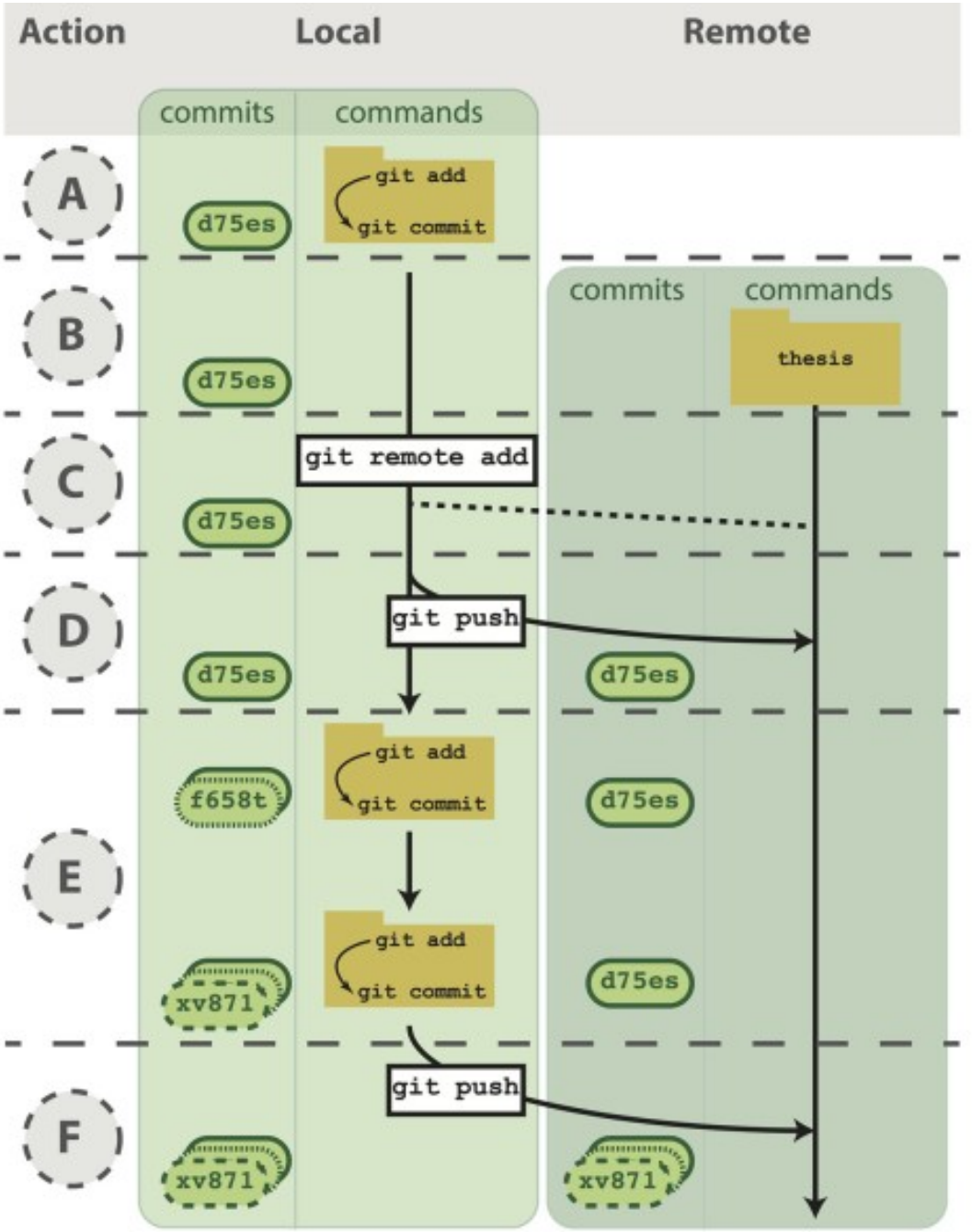
For example, suppose we want to work on fixing a particular bug in our code:

git branch to list all branches
git branch bugfix to create a new branch (re-list branches to confirm)
git switch bugfix to move to the bugfix branch to make & commit changes
git switch master to return to the master branch
git merge bugfix to merge changes in the bugfix branch back into the master branch

Working With a Remote Repository



Working With a Remote Repository



Adding a Remote Repository

Quick setup — if you've done this kind of thing before

or HTTPS SSH https://github.com/pilib/test.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/pilib/test.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/pilib/test.git
git branch -M master
git push -u origin master
```

Adding a Remote Repository

```
pilib@kari:~/Documents/work/2020_21/MA5111/git/test$ git remote add origin https://github.com/pilib/test.git
pilib@kari:~/Documents/work/2020_21/MA5111/git/test$ git push origin master
Username for 'https://github.com': pilib
Password for 'https://pilib@github.com':
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 2.45 KiB | 837.00 KiB/s, done.
Total 12 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/pilib/test.git
 * [new branch]      master -> master
pilib@kari:~/Documents/work/2020_21/MA5111/git/test$
```

Forking a Remote Repository