# Git & Markdown

2021.09.22

LI WANG
WANGLI03@CAAS.CN

# What will you learn through this class?

- Understand what is git, why we use git, and how it works
- Copy the github repository of our course material to your local computer, and update it by "git pull origin master"
- Create your own GitHub/bitbucket repository for your group, share with group members and begin to add files into it (**readme.md**).
- Learn how to write Markdown files

Optional： set up a personal webpage with GitHub template

# Tracking Your Science

**Saving revisions is very important.**
**What kind of versioning do most of you use?**

Many biologists use the "multiple-file" system with cloud-based file sharing (e.g., Dropbox, Google Drive, Box)

```
Wangli-NCFC_2020面上.doc
Wangli-NSFC_2020面上-V1.docx
Wangli-NSFC_2020面上-V2.docx
Wangli-NSFC_2020面上-V3.docx
Wangli-NSFC_2020面上-V4.docx
Wangli-NSFC_2020面上.docx
lvyaqing简历.docx
lvyaqing简历.pdf
paper
sFig2SNPintersection.pdf
sFig7GeneIntersection.pdf
wangli-nsfc_2020面上-v5-zqwu.docx
wangli-nsfc_2020面上-v5-李诚.docx
wangli-nsfc_2020面上-v5-王海洋.docx
wangli-nsfc_2020面上-v5.docx
wangli-nsfc_2020面上-v5_BBWang.docx
wangli-nsfc_2020面上-v6.docx
wangli-nsfc_2020面上-v6_LL edits.docx
wangli-nsfc_2020面上-v6_hgj.docx
wangli-nsfc_2020面上-v7-孙成.docx
wangli-nsfc_2020面上-v7.docx
wangli-nsfc_2020面上-v8.docx
wangli-nsfc_2020面上-v8.pdf
wangli-nsfc_2020面上-v9.pdf
wangli-nsfc_2020面上.pdf
```

# Tracking Your Science

**What are the potential shortcomings of this approach?**

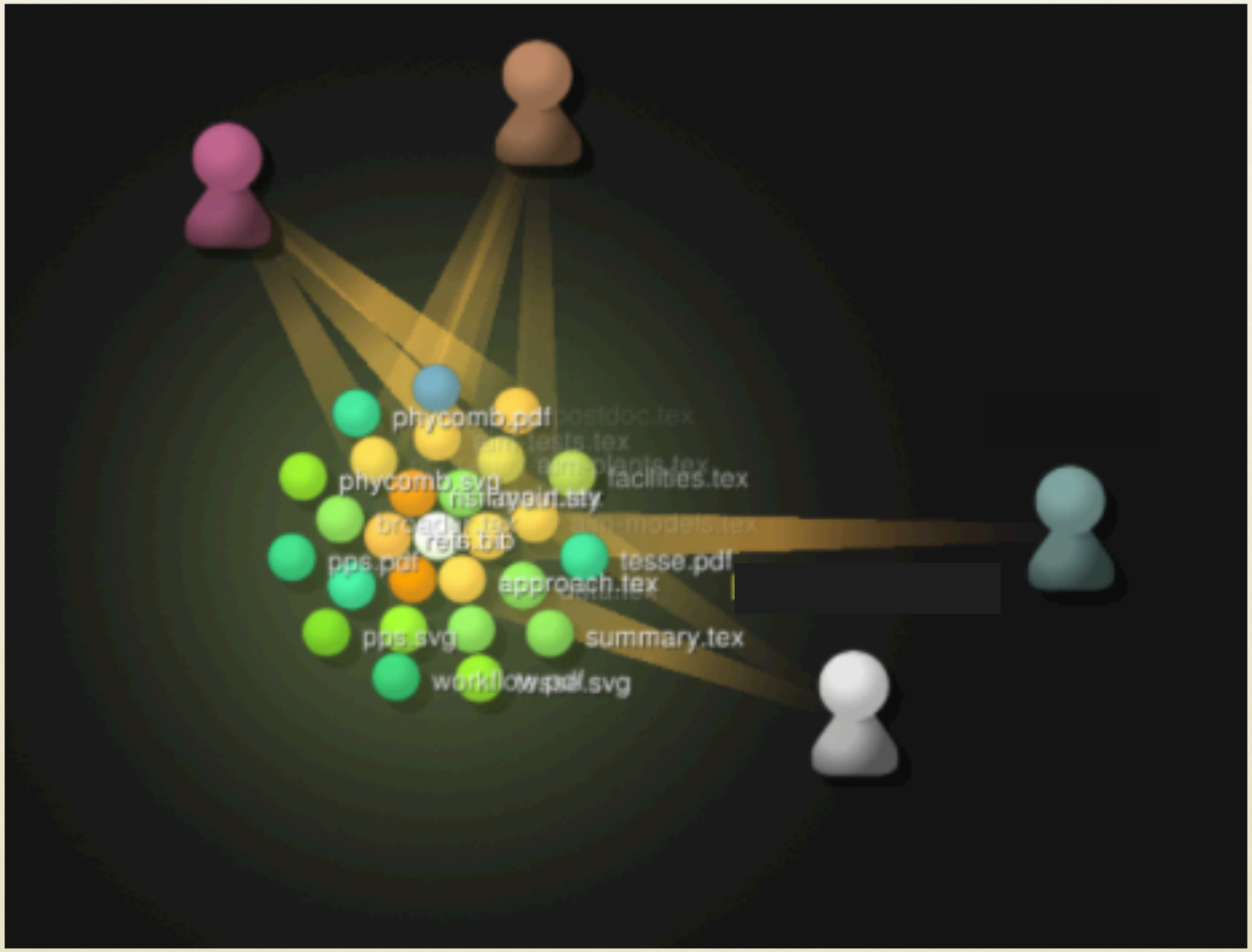
# Tracking Your Science

With a version control system, the file history looks a lot different.

```
lwang at Benjamins-MacBook-Air in ~/Wang_Private/demography/manuscript/GB on mas
ter [!$]
$ ls
Cover_letter_Genome_Biology.pdf     bmc_article_nofiguresTwoAddFile.pdf
Reference.bib                       bmc_article_nofiguresTwoAddFile.tex
bioRxivRevision.pdf                 bmcart-biblio.sty
bioRxivRevision.tex                 bmcart.cls
bmc-mathphys.bst                    figures
bmc_article.bib                     revision
bmc_article.pdf                     spbasic.bst
bmc_article.tex                     supplements
bmc_article_nofigures.pdf           vancouver.bst
bmc_article_nofigures.tex
```

# VERSION CONTROL

A version control system improves organization and collaboration



# Version Control Systems

**What options are there for versioning projects?**

- The most common in biology are Git and SVN (and historically CVS).
- Many other options: https://en.wikipedia.org/wiki/Comparison_of_version-control_software

# Version Control Systems

## What do they allow you to do?

- Track changes made to each file
- Revert the entire project or a single file to a previous version
- Review changes made over time
- View who modified the file (and `blame` them for something if necessary)
- Collaborate with others without overwriting their work or risk file corruption, etc.
- Have multiple independent *branches* of the same repository and make changes without effecting others' work.
- And more...

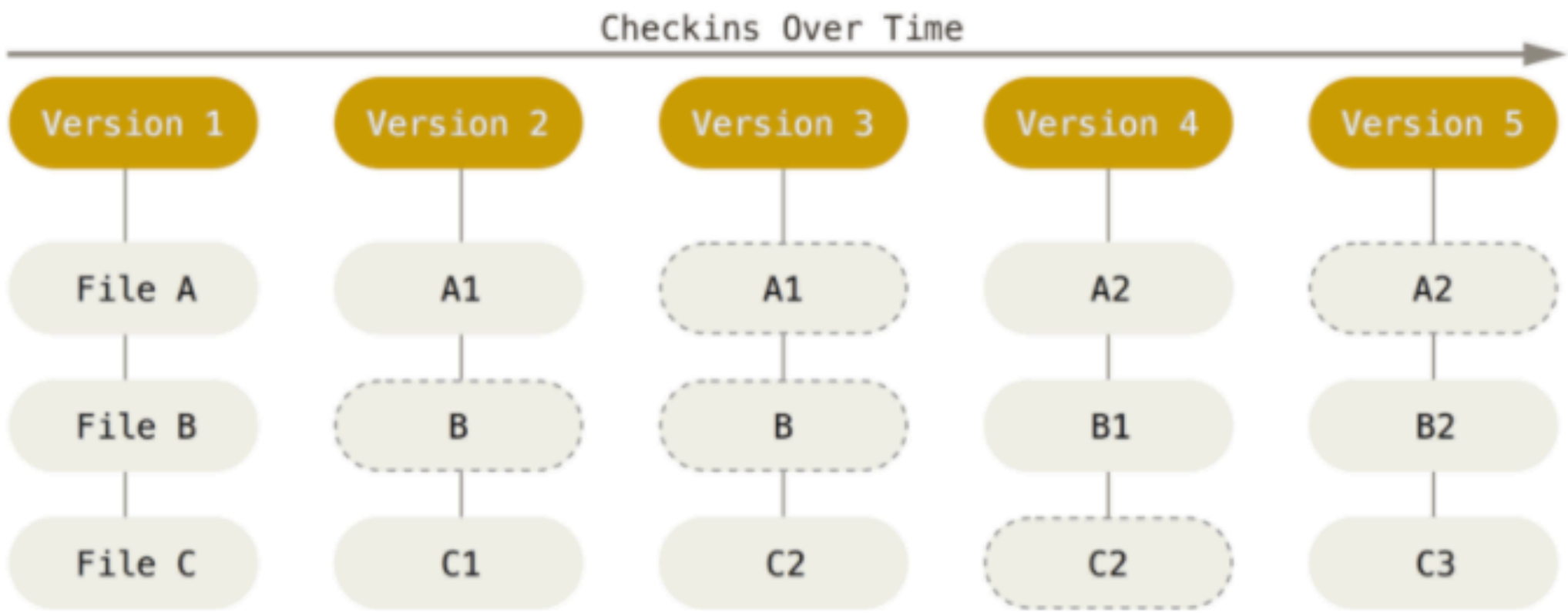
# WHY GIT IS NECESSARY?

- Git Allows You to Keep Snapshots of Your Project
- Git Helps You Keep Track of Important Changes to Code
- Git Helps Keep Software Organized and Available After People Leave

# ♥ Git ♥

## Git manages a filesystem as a set of snapshots

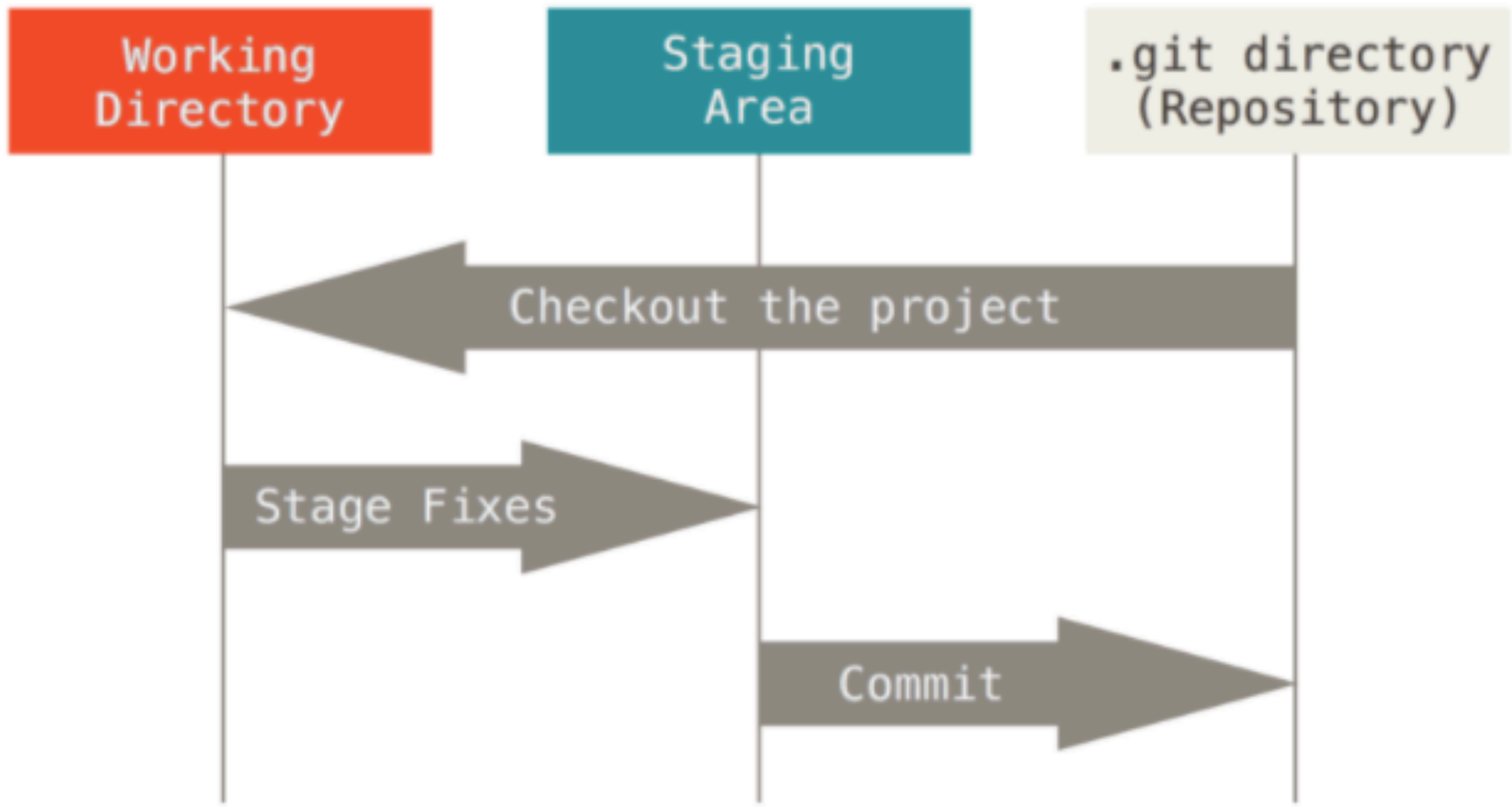
Snapshots are called *commits*




(image source https://git-scm.com)

# ♥ Git ♥

## Almost every interaction with Git happens locally

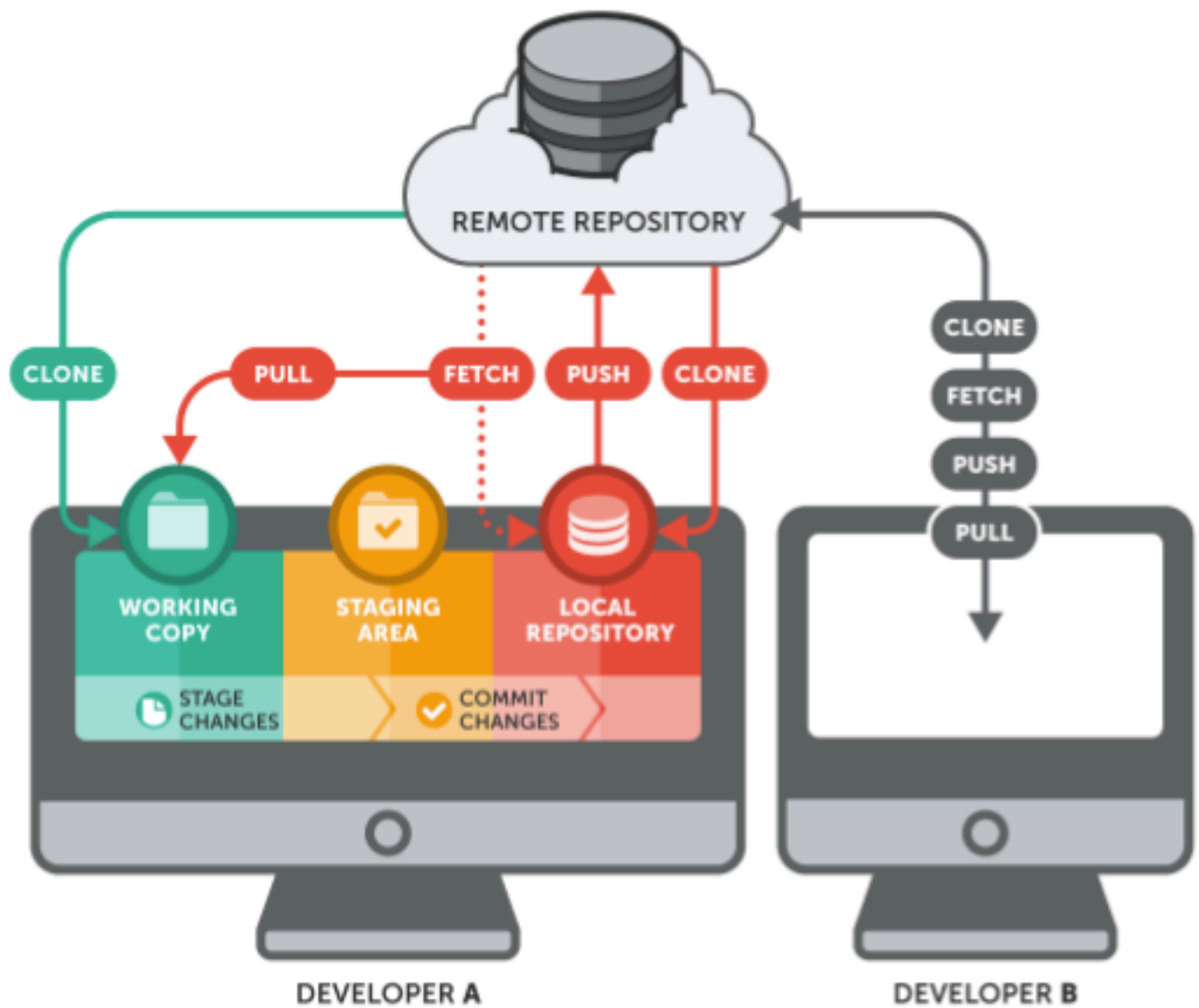



(image source https://git-scm.com)

# ♥ Git ♥

## A remote host adds an additional level to a Git repository

Also, allows for collaboration and back-up.



# USAGE OF GIT

- Sharing of bioinformatics scripts
- Bioinformatic Program development
- Documentation of bioinformatic analyses
- Collaborating on a manuscript

# ♥ Git ♥

## Remote Git repository hosting services

### There are several options for remote hosts

- You can set up your own server and host all of your repositories privately using Gitolite or Gitosis (not recommended)
- You can use a web-based Git host
  - GitHub Drawing: free public repositories & paid private repositories, with repositories over 1 GB discouraged
  - Bitbucket : unlimited free public & free private repositories, limited at 1-2 GB/repo (register with your *.edu* email to get unlimited collaborators on private repositories)

# ♥ Git ♥

## Despite the ♥, Git does have some limitations

Most relevant to this course and our fields are:

- *Repository size*: If your repository gets very large, working within it can be a problem. The network speed will be the main bottleneck. This is why the online Git hosts discourage repos over 1-2 GB.

- *File size*: A single large file can be problematic, particularly if it is frequently being modified. This can also lead to swollen repositories. GitHub will not allow any file over 100 MB.

- *File type*: Git works best with text files, you can have binary files in your repository, but you lose some functionality of version control (like `diff`). Binary files are also often very large. Thus, it is recommended that you keep binary files to a minimum. (This means that it is not practical to use Git to collaborate on MS Word documents.)

# ♥ Git ♥

## Demo: Clone a Repository

- You become familiar with the concepts by using Git. We will start by cloning the GitHub repository of our course.

- Start by going to https://github.com/lepisorus/agisPracticalBioinformatics2021 to get the URL.

```
lwang at Benjamins-MacBook-Air in ~/new
$ git clone https://github.com/WangliLab/CAAS_PracticalBioinformat
Cloning into 'CAAS_PracticalBioinformatics_2020Aut'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 5), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (20/20), done.
```

- Note that using the url git@github... instead of https://github...

# ♥ Git ♥

## Some helpful commands for this cloned repository

Always `pull` from the repository before doing anything with the contents

```
$ git pull origin master
```

Check the `status` of your local files

```
$ git status
```

See the `log` of the snapshots and their commit messages

```
$ git log
```

Compare the `differences` a file you have modified and the last commit

```
$ git diff README.rst
```

# ♥ Git ♥

## Some helpful commands for this cloned repository

Replace a file you modified with the most recent commit using `checkout`

```
$ git checkout README.rst
```

Find out which `branch` you're on

```
$ git branch
```

Change to a different branch

```
$ git checkout h5step7
```

Pull to update from `h5step7`

```
$ git pull origin h5step7
```

# ♥ Git ♥

## What can you do with someone else's GitHub repository?

### If you do not have *push rights*

- You can only clone the repository and make changes locally
- You can ***fork*** their repository and develop it independently
- You can submit a ***pull request*** to their repository if you want to contribute to the original project
- You can contact the owner of the repository and ask them to include you as a contributor and give you push rights (it is recommended that you discuss the nature of your collaboration with them first)

# ♥ Git ♥

## Demo: Clone a Repository

Let's create a new Git repository and host it on GitHub using your own accounts.

But first, let's tell Git who you are

```
lwang at Benjamins-MacBook-Air in ~
$ git config --global user.name "lepisorus"

lwang at Benjamins-MacBook-Air in ~
$ git config --global user.email "lilepisorus@gmail.com"
```

# ♥ Git ♥

## Some helpful commands for your new repository

Initialize a new Git repository

```
$ git init
```

After a file has been added or modified, you can stage the file

```
$ git add README.md
```

Commit the file to your local repository and write a message

```
$ git commit -m "initial commit (README.md)"
```

# ♥ Git ♥

## Some helpful commands for your new repository

After you have made your commit, the repository is up-to-date locally. Next you need to connect your local repo to the remote.

Add the remote

```
$ git remote add origin git@github.com:username/repo-name.git
```

Push your snapshot to the remote

```
$ git push -u origin master
```

# ♥ Git ♥

## Git can be challenging

- What do you find confusing?
- What did you struggle with when creating a repository?
- What do you think would be helpful to overcome these challenges?

# Git Best Practices

- Commit often. Keep commits small and frequent. This also helps you have informative commit messages
- Make sure every commit "works". Never commit if it doesn't compile, runs with errors, or requires files that only exist in your workspace. Since commits are snapshots, this should mean a working snapshot.
- Write commit messages that will be readable and useful to others and *future-you*. (This is really difficult.) Also, remember when your repository is public, anyone can see your commit messages.

# Git Best Practices

- Pull from the remote before you do anything in your repository. This reduces potential merge conflicts. So before you make changes to any files `git pull`, after you make a commit and before you push `git pull`.
- Don't commit unnecessary files. Often these are files that may be generated by your project and can lead to merge conflicts. You can keep these files in your working directory without being tracked using the `.gitignore` file (or the `~/.gitignore_global` file in your home directory).
- Review your changes before committing.
- Use aliases. You can add specific aliases for Git commands in the `.gitconfig` file that lives in your home directory.

```
[alias]
hist = log --graph --pretty=format:'%h %ad | %s%d [%an]' --date=short
last = log -1 HEAD
ci = commit
st = status
```

# Markdown

Markdown is a text-based mark-up language that is easily rendered into HTML

- It has become a staple of reproducible science:
  - it can be rendered on GitHub (also Bitbucket & GitLab) and makes online repositories more readable and accessable
  - it is used as a notebook interface for R (Rmarkdown)
  - it is used as the markup language for Jupyter notebooks, which provide a notebook interface for many languages, including Python and R
- There are some freely available editors that make writing in Markdown pretty easy
  - Windows and Linux: TYPORA https://typora.io/#download
  - Mac OSX: MacDown

# Markdown

## Let's see some Markdown

- It is recommended that all README files on web-based Git hosts be written using Markdown
- The file extension for a Markdown file is `.md`

```
# Welcome to `Practical Bioinformatics` course

A graduate course covering critical computational skills and practical bioinformatic packages.

## AGIS, CAAS

A graduate course covering _**critical computational skills**_ for working with biological data

**Instructors:** [Yuwen Liu](http://www.), [Li Wang](wanglilab.github.io)

**Time/Location:** Wednesday evenings 6.30--9:10pm; D104

**Web View:** [Practical Bioinfomatics 2020 Autumn](https://github.com/WangliLab/CAAS_PracticalBioinformatics_2020Aut)

## Reading Material

* [*Practical Computing for Biologists*](http://practicalcomputing.org/)

## Course Schedule

**[Week 1](https://github.com/WangliLab/CAAS_PracticalBioinformatics_2020Aut/week1)**
```

### Welcome to `Practical Bioinformatics` course

A graduate course covering critical computational skills and practical bioinformatic packages.

## AGIS, CAAS

A graduate course covering ***critical computational skills*** for working with biological data

**Instructors:** Yuwen Liu, Li Wang

**Time/Location:** Wednesday evenings 6.30--9:10pm; D104

**Web View:** Practical Bioinfomatics 2020 Autumn

## Reading Material

- *Practical Computing for Biologists*

## Course Schedule

**Week 1**

**Week 2**

# Get Help

## No matter the problem (with Git or anything else in this class), someone else has encountered it already.

- Google is an immensely powerful tool for troubleshooting computational problems.
- If you can articulate your problem in the form of a Google search query, you will likely find the answer online.
- If your problem is unique, you can always submit a question on **Stack Overflow**, such as **“github markdown cheat sheet”.**

# Git Collaboration

## Now that we all have repositories on GitHub, let's collaborate!

- Get into groups
- Give your collaborators access (push rights) to your repository
- Clone your collaborator's repository
- Add and edit files in your collaborator's repository
- Commit and push those changes to the remote

# How to add collaborators to your GitHub repository?

**GitHub Help**

1. Ask for **the** username of **the** person you're **inviting** as **a collaborator**.
2. Under **your repository** name, click Settings.
3. In **the** left sidebar, click **Collaborators**. **manage access**
4. Under "**Collaborators**", start typing **the collaborator's** username.
5. Select **the collaborator's** username from **the** drop-down menu.
6. Click Add **collaborator**.

# GIT GUI (GRAPHICAL USER INTERFACES)

- GUI tools: https://git-scm.com/download/gui/linux
- These tools may be very helpful for doing things that you don't do every day (when it's difficult to remember the command)
- They also provide nice ways to visualize your repository tree and diff commits, etc.

# SUMMARY

**create the repository on the cloud:**
git clone ***
git pull origin master
git add .
git commit -m “edits to the file”
git pull origin master
git push origin master

# SUMMARY

**create the repository in the local:**

- git init
- git add README.md
- git commit -m “first commit”
- git remote add origin git@github.com:username/IamGreat.git
- git push origin master

# What will you learn through this class?

- Understand what is git, why we use git, and how
- Copy the Github repository of our course material to your local computer, and update it by “git pull origin master”
- Create your own GitHub repository for your group, share with group members and add at least one markdown file into it.

Optional： set up a personal webpage with GitHub template

# Assignments

Chapters 3 & 7

Next Wednesday