

Version Control

git + GitHub

Marco Morales
marco.morales@columbia.edu

GR50672
Modern Data Structures
Spring 2022
Columbia University

A Data Science project...

Three **aims** of a Data Science project

a) **reproducibility**

- ▶ anyone should be able to arrive to your **same results**

b) **portability**

- ▶ anyone should be able to **pick up where you left off** on any machine

c) **scalability**

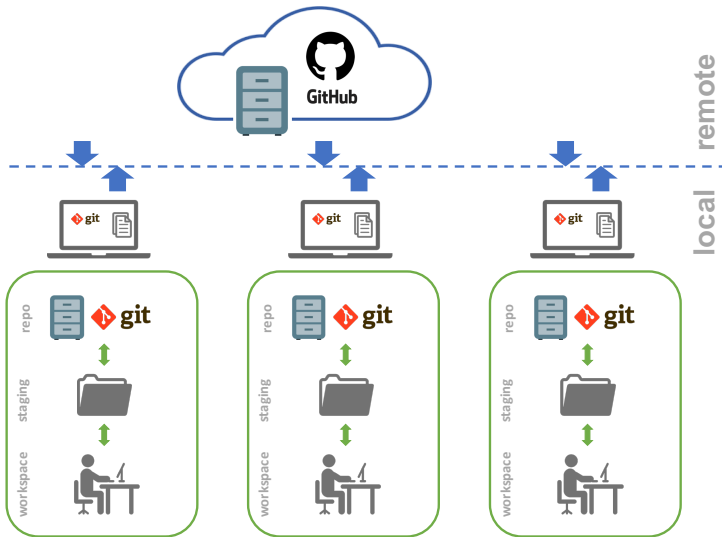
- ▶ your project should also work for **larger data sets** and/or be on the path of **automation**

a) and b) crucial for **collaborative work**

Why version control?

- ▶ **version control** allows you to keep track of changes / progress in your code
 - ▶ keeps “**snapshots**” of your code over time
 - ▶ helpful to **debug**, and to enhance **reproducibility**
 - ▶ also great for **team collaboration** (everyone can see who changed what!) and **portability**
- ▶ **git** is a version control software
- ▶ **GitHub** is an online open source repository

An ideal version control setup for collaboration

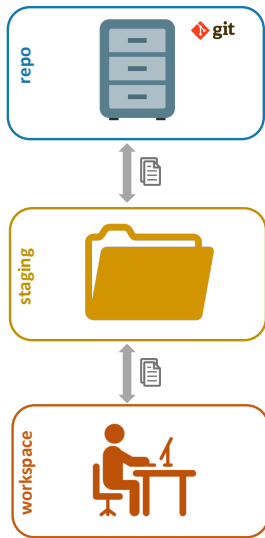


git locally

recap: what was this **git** thing?

- ▶ **git** is a version control software
 - ▶ installed “locally” on your computer (or virtual machine)
 - ▶ keeps snapshots of your (coding) work
- ▶ helps with
 - ▶ “time travel” (insert your favorite “Back to the future” gif here)
 - ▶ keep collaboration organized when multiple people are working on the same project
- ▶ a vehicle to be nice to your fellow collaborators (and to the you of the future)

git: a mental model



Introduce yourselves: git, meet your new user!

from the command line:

- ▶ set your **user name** and **email address**

```
$ git config --global user.name "John Doe"
```

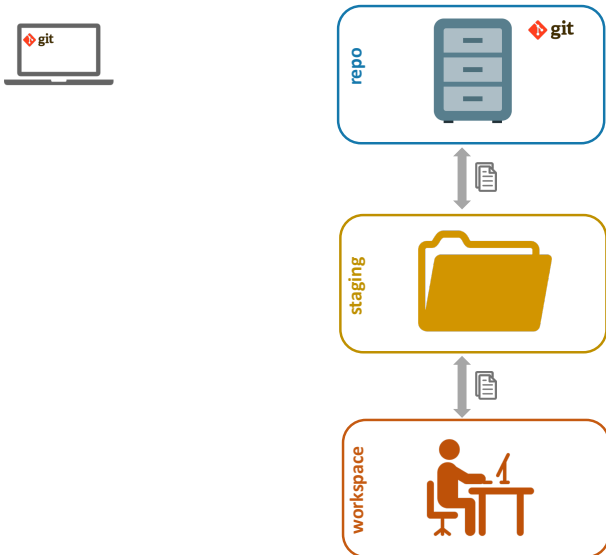
```
$ git config --global user.email johndoe@example.com
```

- ▶ **verify** that information was successfully entered

```
$ git config --list
```

- ▶ this information gets baked in your commits
- ▶ **ProTip:** other useful information (e.g. proxy settings) also goes on `git config`

now, turn your folder structure into a git repo



now, turn your folder structure into a git repo

from the command line:

- ▶ go to the **root** of your project and **initialize** the repo

```
$ git init
```

- ▶ there are **files you never want tracked** by git (e.g. log files, access keys), even by mistake
- ▶ from the root of your local repository, create a `.gitignore` file

```
$ touch .gitignore    (Mac)
```

```
$ echo > .gitignore   (Windows)
```

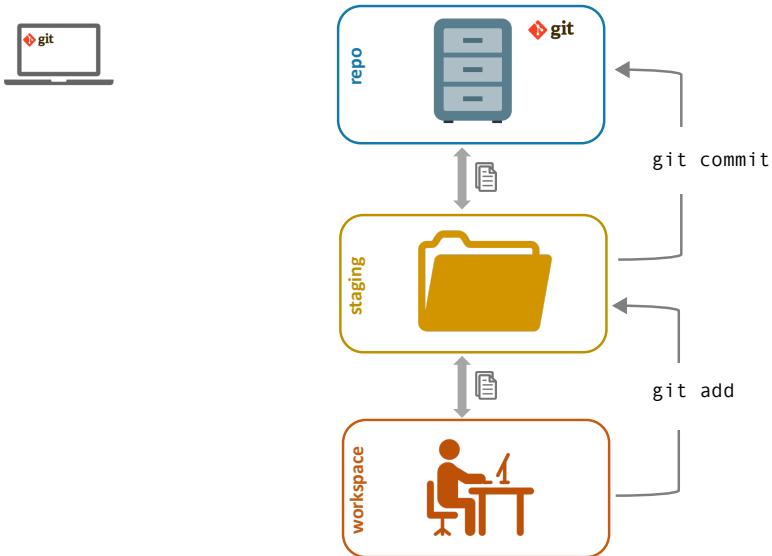
- ▶ add files you want git to ignore in the `.gitignore` file

what could go into a .gitignore file ?

```
# OS generated files #  
*.DS_Store  
  
# Jupyter Notebook  
.ipynb_checkpoints  
  
# RStudio files  
*.Rproj.user/  
  
# all data folders  
data/
```

- ▶ **ProTip:** further info/templates:
<https://github.com/github/gitignore>

your basic git workflow



your basic git workflow

from the command line:

- ▶ indicate a file to be tracked by git

```
$ git add samplefile.R
```

- ▶ verify what's being tracked

```
$ git status
```

- ▶ commit your tracked files (with an informative message)

```
$ git commit -m "Commit initial files"
```

a few confusing things about git

- ▶ a file will be committed **exactly** as it was when you `git add-ed` it
- ▶ if you change the file **after** you `git add` it, and want to commit the new changes, you need to `git add` again **before the** `git commit`
- ▶ use `git status` to assess what's being staged and will be committed

git workflow **ProTips**

- ▶ **NEVER** use `git add .`
- ▶ use `git status` often as **validation**
- ▶ only `add` and `commit` **source files**
 - ▶ omit files you can reproduce using source files
- ▶ `commit` **small chunks of logically grouped changes**
 - ▶ you may want to undo a change, and only that change
- ▶ `commit` with **informative** (imperative mood) **messages**
 - ▶ *[this commit will]* `Rename income variable`

git workflow ProTips

A quick detour: `master` vs `main` branch

- ▶ **Pro Tip:** current best practice is to use `main` for your default branch; used to be `master`
- ▶ by default, git will create a `master` branch after your first commit
- ▶ easy to rename your branch to `main`

```
$ git branch -M main
```

- ▶ for a permanent solution (in git \geq 2.28)

```
$ git config --global init.defaultBranch main
```


**push globally
(to GitHub)**

recap: what was this **GitHub** thing?


- ▶ **GitHub** is a cloud service that hosts **git** repositories
 - ▶ lives in the cloud
 - ▶ understands the git dialect!
 - ▶ can speak with multiple git users simultaneously
- ▶ helps with
 - ▶ persisting repository storage (your dog cannot eat your repo!)
 - ▶ synchronizing work
 - ▶ minimizing risk of people stepping on each other's toes (while working on the same project)
 - ▶ seamless transition between environments (dev > staging > prod)

first, create a GitHub repo to store/share in the cloud

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 marco-morales ▾

Repository name

/ testrepo ✓

Great repository names are short and memorable. Need inspiration? How about [friendly-octo-guide](#).

Description (optional)

a test repository

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

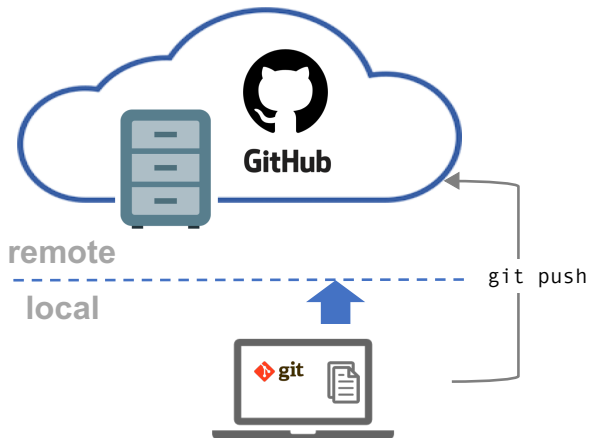
Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

then, `push` to that GitHub repo



then, push to that GitHub repo

from the command line:

- ▶ tell git the **location** of the remote GitHub repo you just created (typically nicknamed “origin”)

```
$ git remote add origin  
https://github.com/marco-morales/testrepo.git
```

- ▶ send **committed files** to your GitHub (“origin”) repo from your local git branch (“main”)

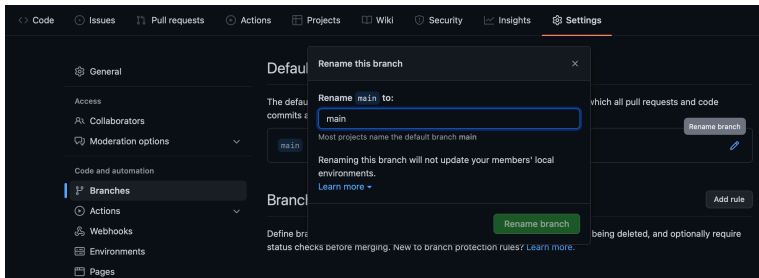
```
$ git push -u origin main
```

- ▶ **ProTip:** current best practice is to use `main` for your default branch. Default used to be `master`

GitHub workflow ProTips

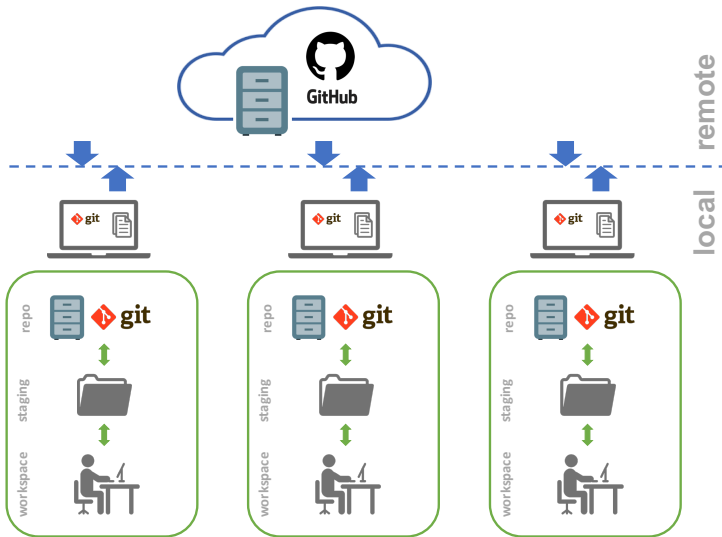
A quick detour: `master` vs `main` branch

- ▶ **Pro Tip:** current best practice is to use `main` for your default branch; used to be `master`
- ▶ by default, GitHub will create a `master` branch after your first create a repo if you do not change defaults
- ▶ easy to change permanently in your GitHub settings




git+GitHub for team collaboration

all the building blocks are now in place



now, enable collaborators in your GitHub repo

 marco-morales / testrepo

Unwatch ▾2

★ Star0

🍴 Fork0

<> Code

🔔 Issues0

🔗 Pull requests1

📁 Projects1

📖 Wiki

📊 Insights

⚙️ Settings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services


Deploy keys

Moderation

Interaction limits

Collaborators

Push access to the repository

 gulbzhnh

×

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Add collaborator

important to know what each role can do

- ▶ add **collaborators** to your repo
 - ▶ as a repo **owner** you have control over what gets changed
 - ▶ **collaborators** will be able to `push` to the repo

a) **Collaborators:**

- ▶ work on a branch on the repo and create code
- ▶ send a `pull` request to add that code to the master repo

b) **Owner:**

- ▶ comment on the `pull` request
- ▶ accept the `pull` request and/or `merge` the code

(1) a collaborator creates a branch to work on, that will eventually be merged back to the main branch

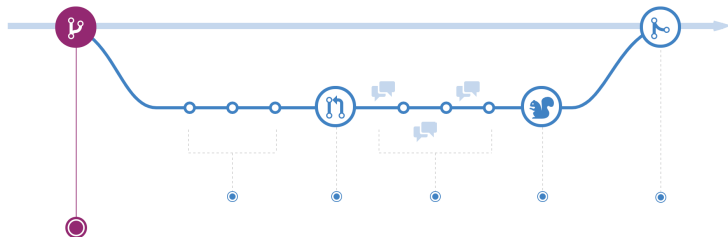


Figure: Understanding the GitHub flow

- ▶ changes in a branch do not affect the `main` branch
- ▶ **ProTips**
 - ▶ anything in the `main` branch is deployment-ready
 - ▶ the branch should always be created off of the `main` branch

(2) a collaborator works and `commits` on that branch

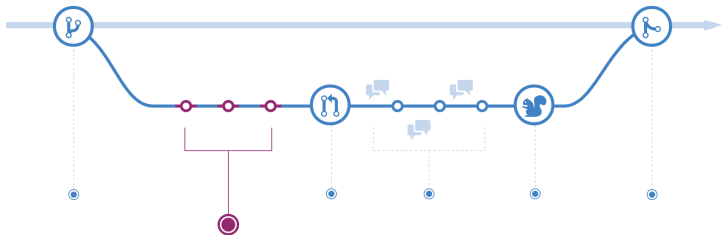


Figure: Understanding the GitHub flow

- ▶ use the same workflow in a branch: `git add`, `git status`, `git commit`
- ▶ **ProTip**
 - ▶ use informative messages in your branch `commits`

(3) a collaborator `pushes` to create a pull request

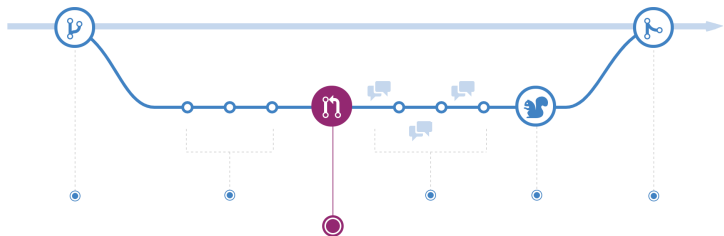



Figure: Understanding the GitHub flow

- ▶ a pull request notifies that your changes are ready to be reviewed and merged back to the `main` branch
- ▶ the review will validate that the changes do not create problems in the `main` branch and incorporate other members' comments

(3) a collaborator pushes to create a pull request

 **marco-morales / testrepo**

Unwatch ▾ 2

★ Star 0

🍴 Fork 0

<> Code

🕒 Issues 0

🔗 Pull requests 1

📁 Projects 1

📖 Wiki

📊 Insights

⚙️ Settings

Label issues and pull requests for new contributors

Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with **help wanted** or **good first issue**

Filters ▾

🔍 is:pr is:open

Labels

Milestones

New pull request

☐ 1 Open ✓ 1 Closed

Author ▾

Projects ▾


Labels ▾

Milestones ▾



Reviews ▾

Assignee ▾

Sort ▾

☐  **Melissabbranch**

#2 opened on Feb 7, 2018 by marco-morales

💡 ProTip! Type   on any issue or pull request to go back to the issue listing page.

(4) an owner reviews changes, resolves conflicts, and approves the merge

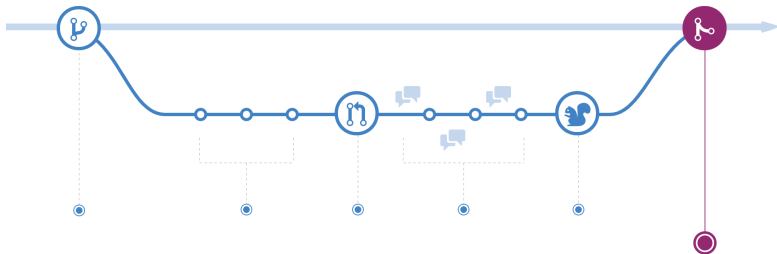



Figure: Understanding the GitHub flow

- ▶ once the proposed changes have been validated they are merged back into the `main` branch
- ▶ the merge preserves records of changes made on the branch

(4) an owner reviews changes, resolves conflicts, and approves the merge

 marco-morales / testrepo

Unwatch 2

Star 0

Fork 0

<> Code

Issues 0

Pull requests 1

Projects 1

Wiki

Insights

Settings

Melissabranch #2

Edit

Open

 marco-morales wants to merge 3 commits into master from melissabranch

Conversation 0

Commits 3

Checks 0

Files changed 1

Changes from all commits File filter... Jump to... +1 -1

Diff settings

Review changes

2 testfile.R

2	@@ -3,4 +3,4 @@
3	### this is a test fi
4	
5	print("Hello World!")
6	- print(";")
6	+ print("Hello World!")

Write

Preview

AA B i

« > ↺

⋮ ⋮ ⋮

@ 🔖 ↶

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

☒ Comment

Submit general feedback without explicit approval.

☐ Approve

Submit feedback and approve merging these changes.

☐ Request changes

Submit feedback that must be addressed before merging.

Submit review

© 2019 GitHub, Inc. Terms Privacy Security

rinse and repeat



**Though this be madness,
yet there's method in't**

Recap: the method to this version control madness...

- ▶ basic **actions** to master in git
 - ▶ `git init`: initializes git, and indicates that the folder should be tracked
 - ▶ `git add`: brings new files to the attention of git to be tracked as well
 - ▶ `git commit`: takes a snapshot of alerted files
 - ▶ `git push`: sends changes committed in your branch (of your local repo) to the remote branch (of the GitHub repo)

Recap: the method to this version control madness...

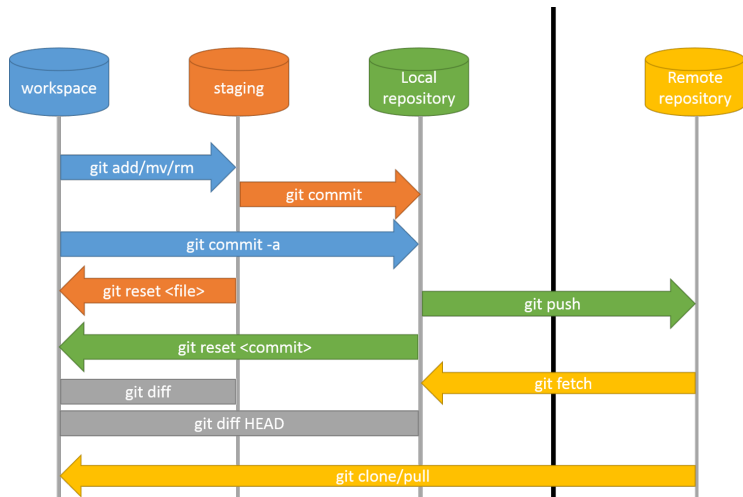


Figure: Pro Git, 2nd Edition

Version Control

git + GitHub

Marco Morales
marco.morales@columbia.edu

GR50672
Modern Data Structures
Spring 2022
Columbia University