Hand-on tutorial of Git

Kai Wu, PhD student

Kai.Wu19@student.xjtlu.edu.cn

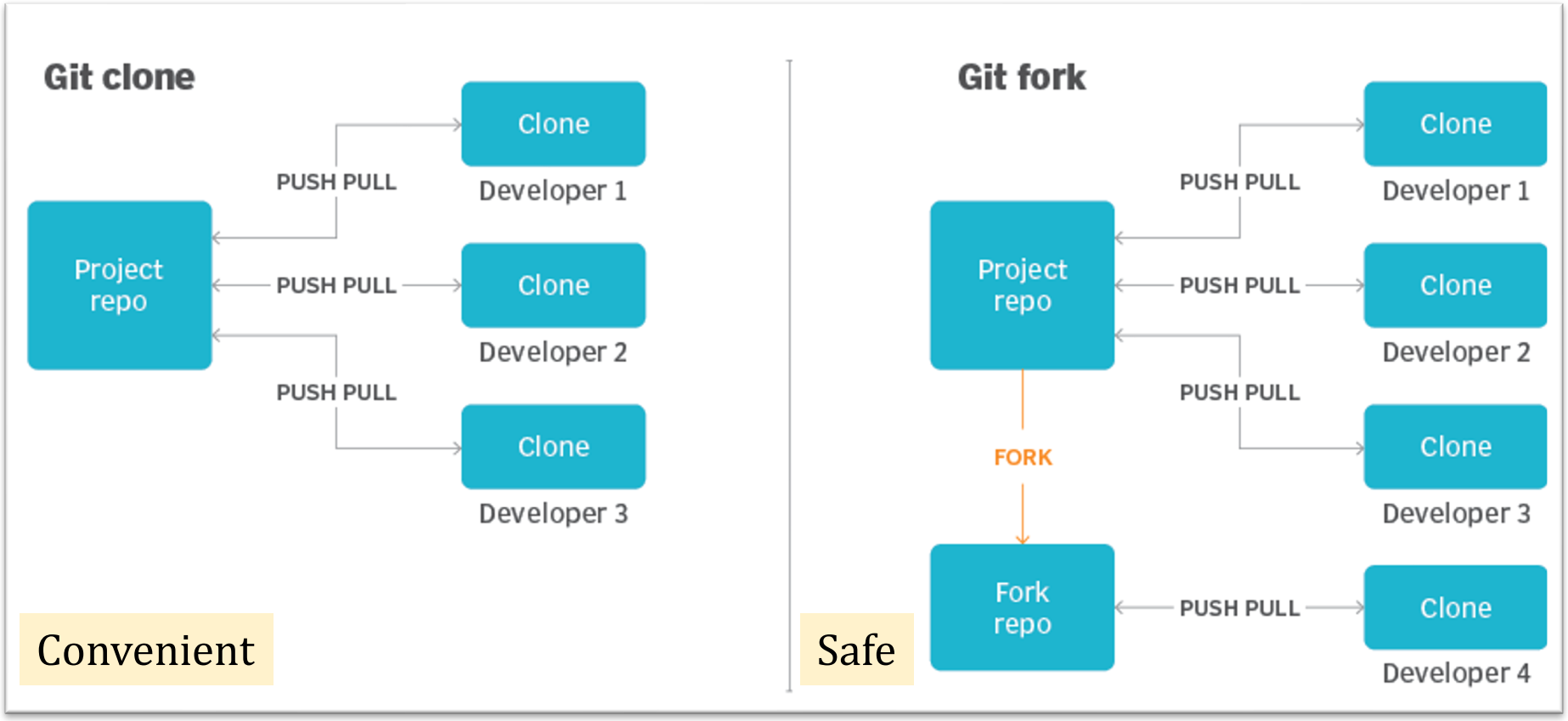
2022-04-22

Before the hand-on tutorial

1. Download or upgrade Git (apt-get / yum / ...)
2. `git --version` , confirm version >=2.23
3. Tell Git who you are

`git config --global user.name “[firstname lastname]”`
set a name that is identifiable for credit when review version history
`git config --global user.email “[valid-email]”` ← your GitHub email
set an email address that will be associated with each history marker
`git config --global color.ui auto`
set automatic command line coloring for Git for easy reviewing

How to get NBODY6++GPU from GitHub?



A fork is a copy of a repository.
Forking a repository allows you to freely experiment with changes without affecting the original project.

How to get NBODY6++GPU from GitHub?

Depends on your role in this project

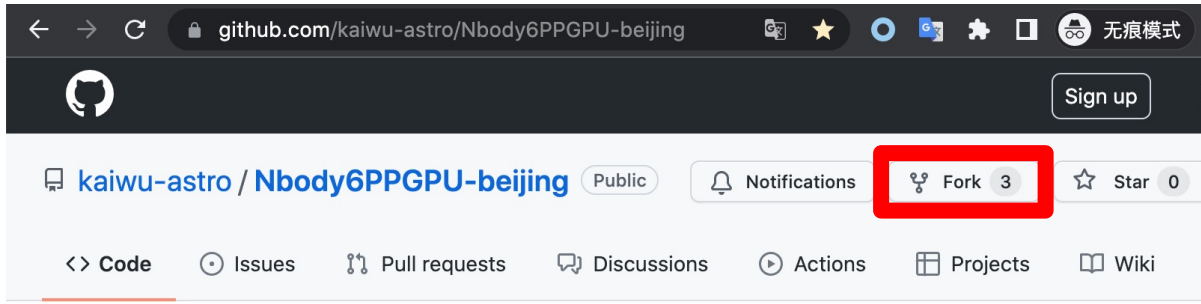
Developer of main versions

Developer of special versions (massless, stardisk, etc.)

Not a developer yet / seldom participate

How to get NBODY6++GPU from GitHub?

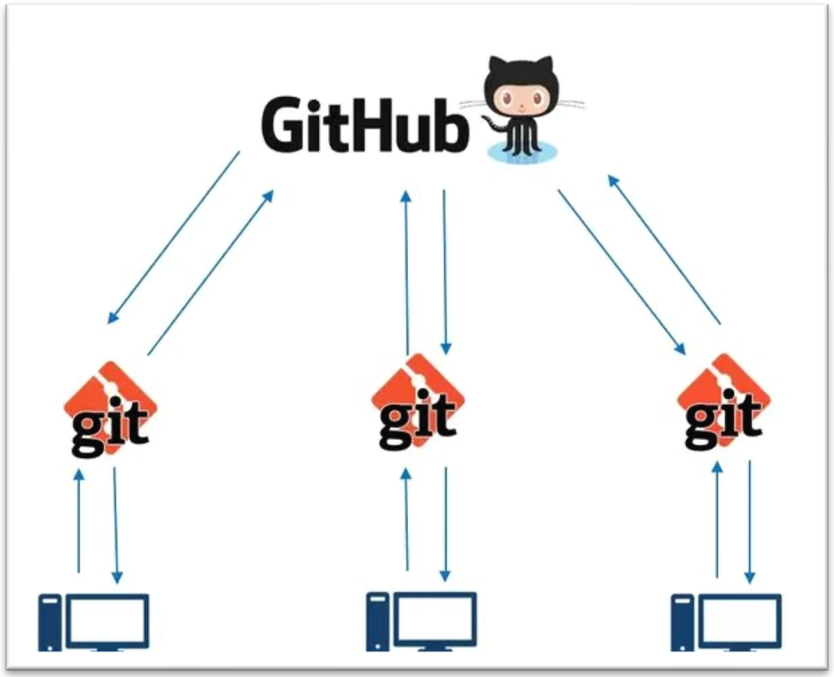
Developer	**Not a developer / seldom develop**
	• Click Fork in our repo's GitHub
• Git clone	• Git clone your folked repo
• Modify code	• Modify code
• Push to GitHub	• Push to GitHub (your folked repo)
	• Pull Request to our repo



https://docs.github.com/en/get-started/quickstart/fork-a-repo

What is Git clone

Difference between Git and GitHub



- Git is a software, for version control
  - You work on your local Git repository
- GitHub stores online repositories (also called remote repository)
- Git clone: download online repo to local
- After work on local, you upload (git push) changes to online, so collaborators can know

[For work] SSH Clone

- Register a GitHub account. Tell GitHub your SSH public key:
  - https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account
- Developers
  - `git clone git@github.com:kaiwu-astro/Nbody6PPGPU-beijing`
- Seldom participate developers
  - first fork our repo (link) on GitHub, and then
  - `git clone git@github.com:`your_username`/Nbody6PPGPU-beijing`

[Only from trail today] HTTPS Clone

Advantage of HTTPS Clone

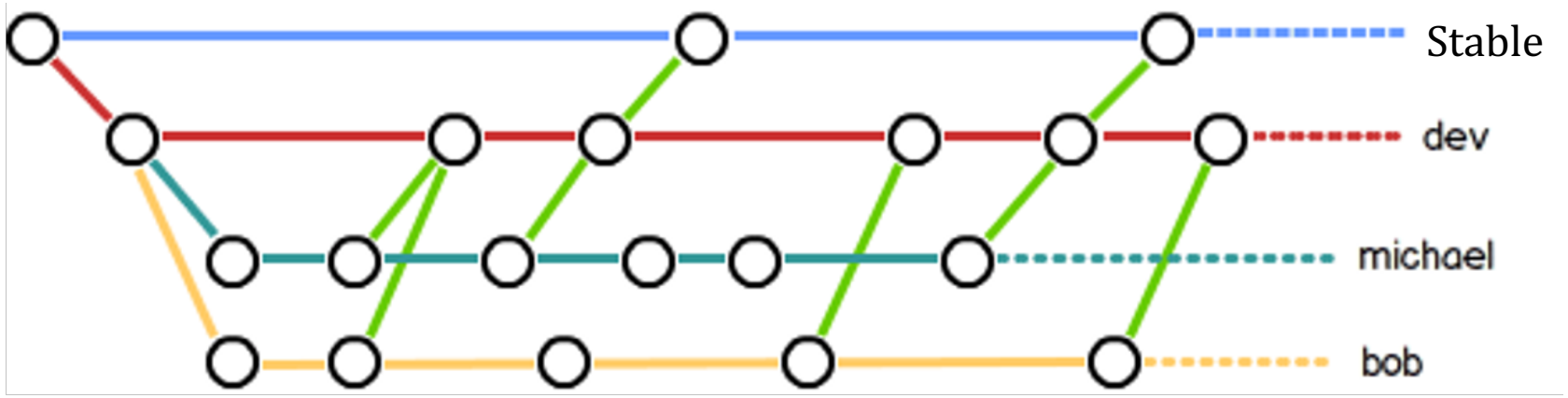
- Free from GitHub account

Disadvantage of HTTPS Clone

- Hard to upload
- Play Git alone locally

1. `git clone https://github.com/kaiwu-astro/NBodyGitHubTutorial`
2. after clone, go to Git directory
   `cd NBodyGitHubTutorial`

How Git works – branch



- Branches allow to
  - separate stable code and developing code
  - different versions (massless, stardisk, etc)

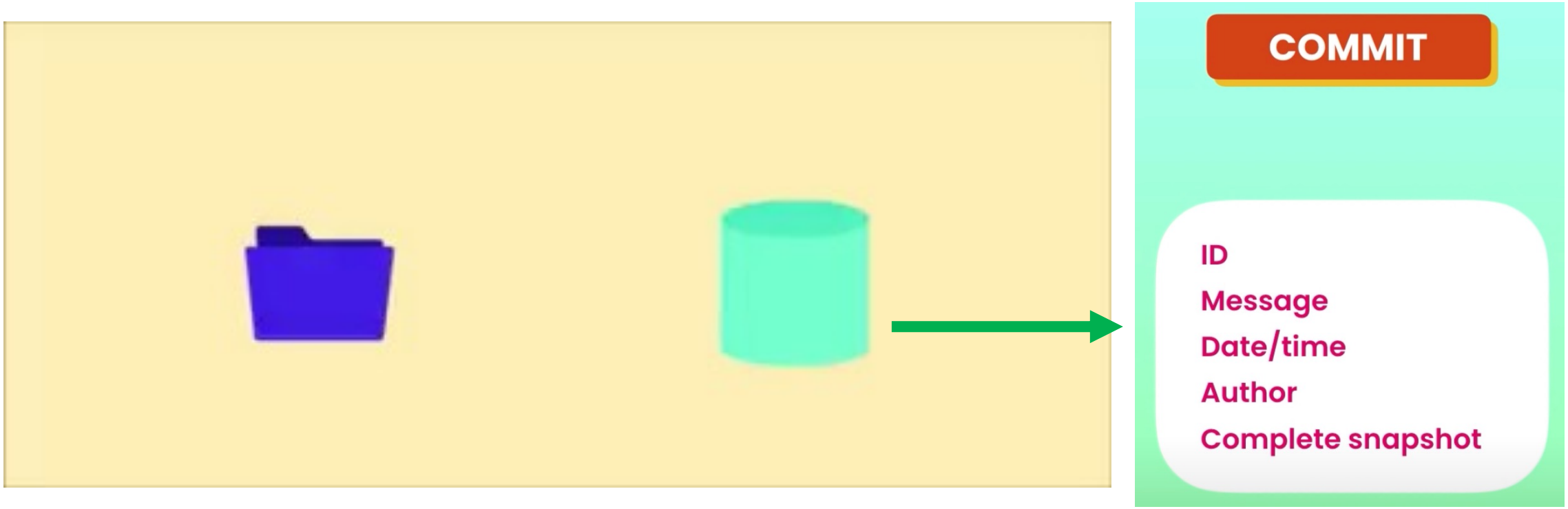
Switch to the correct branch after clone

- For core developers: switch to dev branch
  - `git switch dev`
- Special version developers do
  - `git switch your-branch` `# use git switch –c to create if you didn't`
- List branch
  - `git branch`
  - `git branch --all`
- Different branch may have totally different files. Try:
  - `git switch trash` `#previous repo with no modification time list. Will delete`
  - `ls`
  - `git switch dev`
  - `ls`

4 checks before working

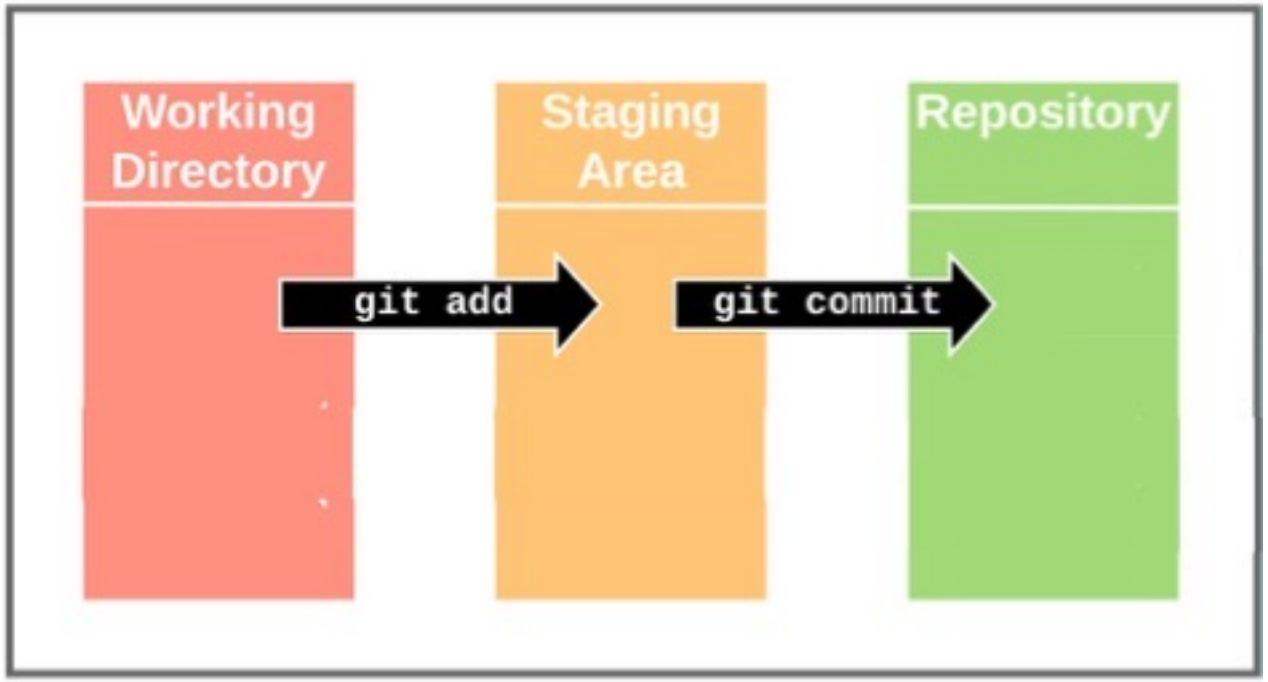
- Right name and email
  - `git config --list`
- Right clone
  - SSH clone for formal work
- Right directory
  - cd to Git directory
- Right branch
  - dev branch for core developers
  - your branch for special version developer

How Git works - snapshots



- The basic idea of version control: take snapshots of the project
- so many snapshots in very small sizes (.git directory, 13MB now)

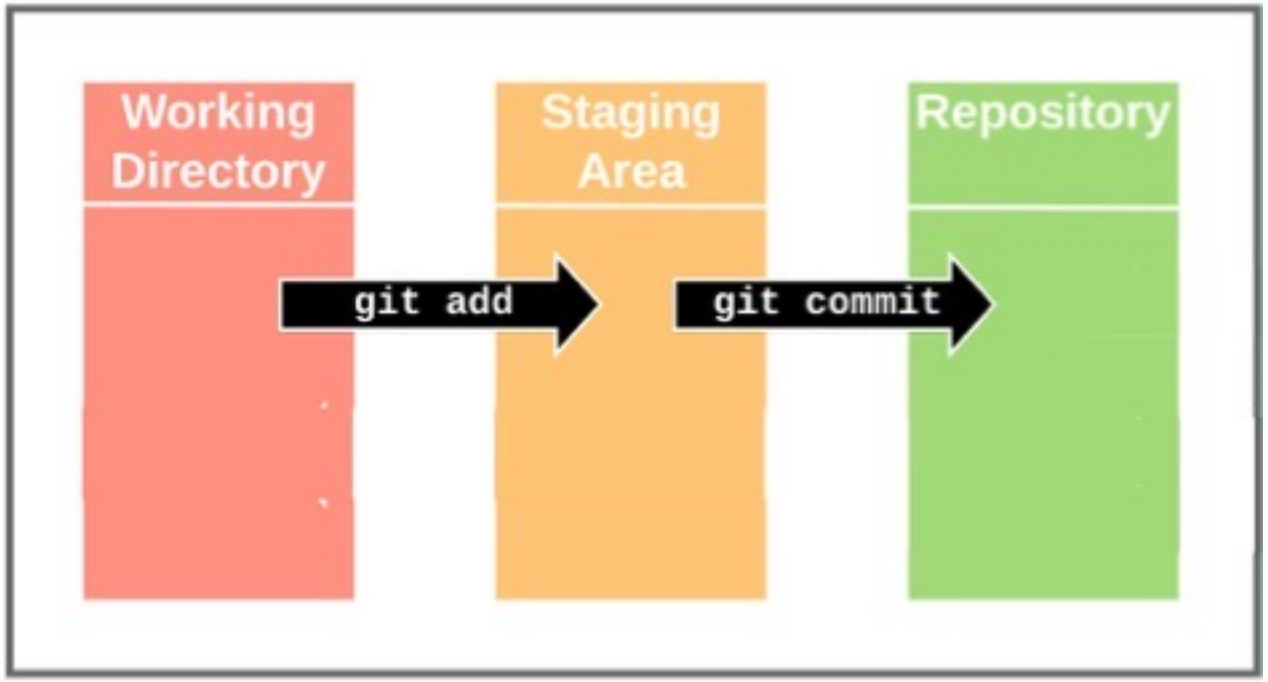
How Git works – stage and commit



- Special of Git: staging area. You first put some of modified files in, and commit

```
1. git status
2. echo hello_from_kai >> README.md
     You can also edit with vi, nano, GUI tools
3. git status
4. git add README.md    #file or dir
5. git status
6. git commit –m “fixed lack of hello”
```

How Git works – stage and commit



- Special of Git: staging area. You first put some of modified files in, and commit
- Example: one day you changed 4 files
  - `git add src/Main/data.F src/Main/input.F`
  - `git commit –m “add feature: accepts .ini input”`
  - `git add src/Main/ksreg.F src/Main/regint.f`
  - `git commit –m “fix a bug in KS regularization”`

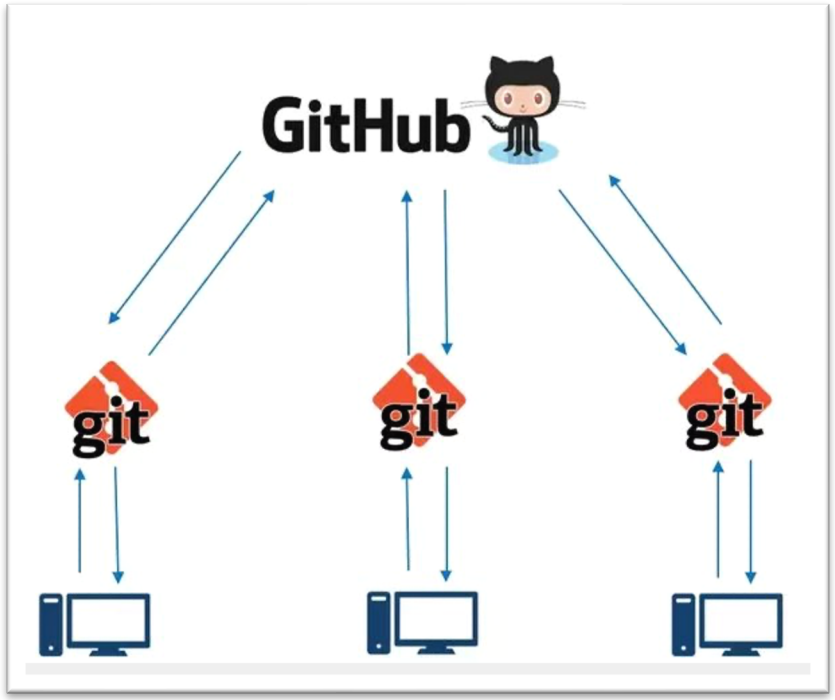
“Who modified this file ??!”

- View commit history
  - git log
  - git log --pretty=oneline
- "Who modified this file"
  - git blame [filename]

Easy and pretty log with `git lg`, once and for all:

```
git config --global alias.lg "log --color --graph --
pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)
%C(bold blue)<%an>%Creset' --abbrev-commit"
```

Sync with remote



Sync with remote

`git push [alias] [branch]`

Transmit local branch commits to the remote repository branch

`git pull`

fetch and merge any commits from the tracking remote branch

```
~/0TempSave/n6togit/repodir  stable  git pull
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 40 (delta 20), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (40/40), done.
From github.com:kaiwu-astro/tmp
   7bde4c8..75d7805  stable     -> origin/stable
Updating 7bde4c8..75d7805
Fast-forward
 .circleci/config.yml | 24 +++++++++++++++++++---------
 httpstest            |  0
 testssh              |  0
 3 files changed, 15 insertions(+), 9 deletions(-)
 create mode 100644 httpstest
 create mode 100644 testssh
```

A good habit: always run `git pull` before push

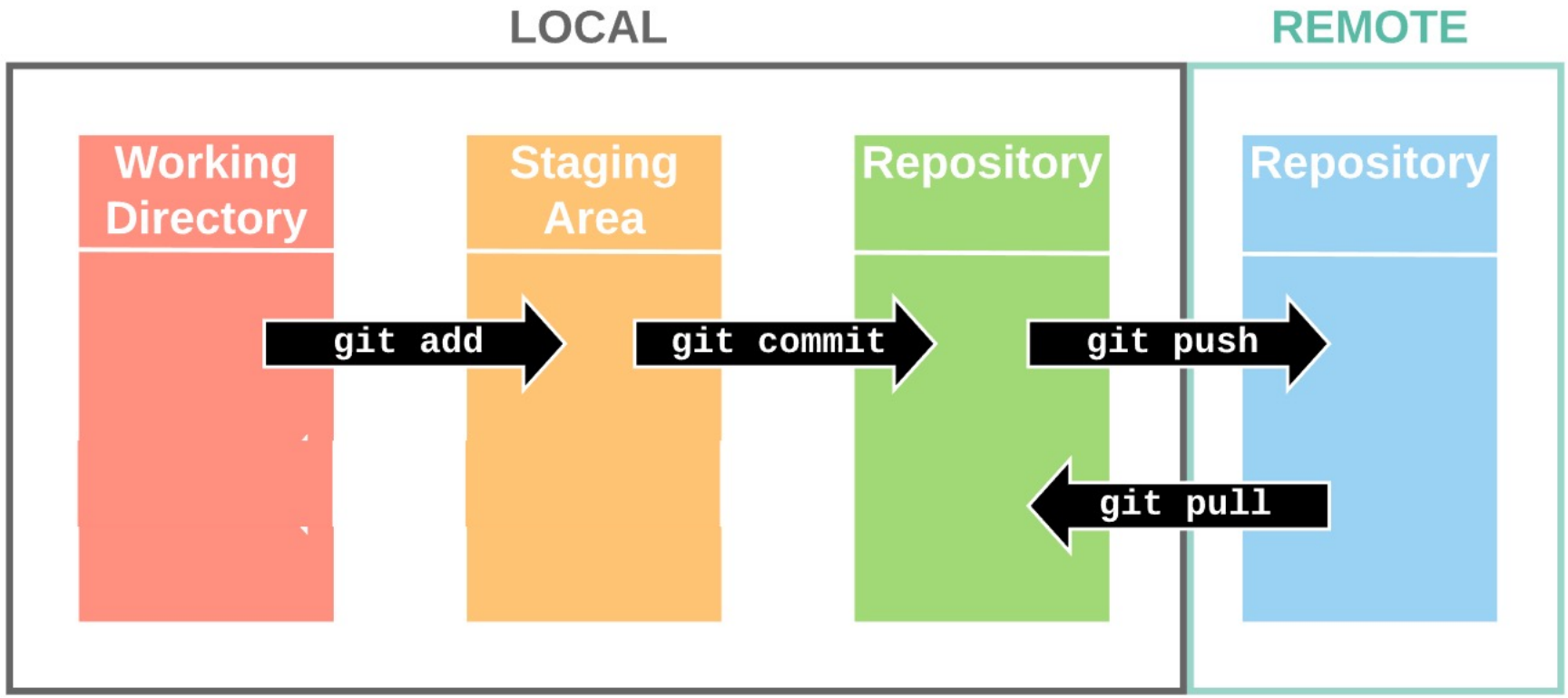
```
~/0TempSave/n6togit/tmp  dev  git status
On branch dev
Your branch is up to date with 'origin/dev'.

nothing to commit, working tree clean
~/0TempSave/n6togit/tmp  dev  echo no_warranty >> LICENSE
~/0TempSave/n6togit/tmp  dev  git status
On branch dev
Your branch is up to date with 'origin/dev'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        LICENSE

nothing added to commit but untracked files present (use "git add" to track)
~/0TempSave/n6togit/tmp  dev  git add .; git commit -m "add license"
[dev bbbccf0] add license
 1 file changed, 1 insertion(+)
 create mode 100644 LICENSE
~/0TempSave/n6togit/tmp  dev  git pull && git push
Already up to date.
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To meow:wukai-meow/tmp.git
   57c4906..bbbccf0  dev -> dev
~/0TempSave/n6togit/tmp  dev
```

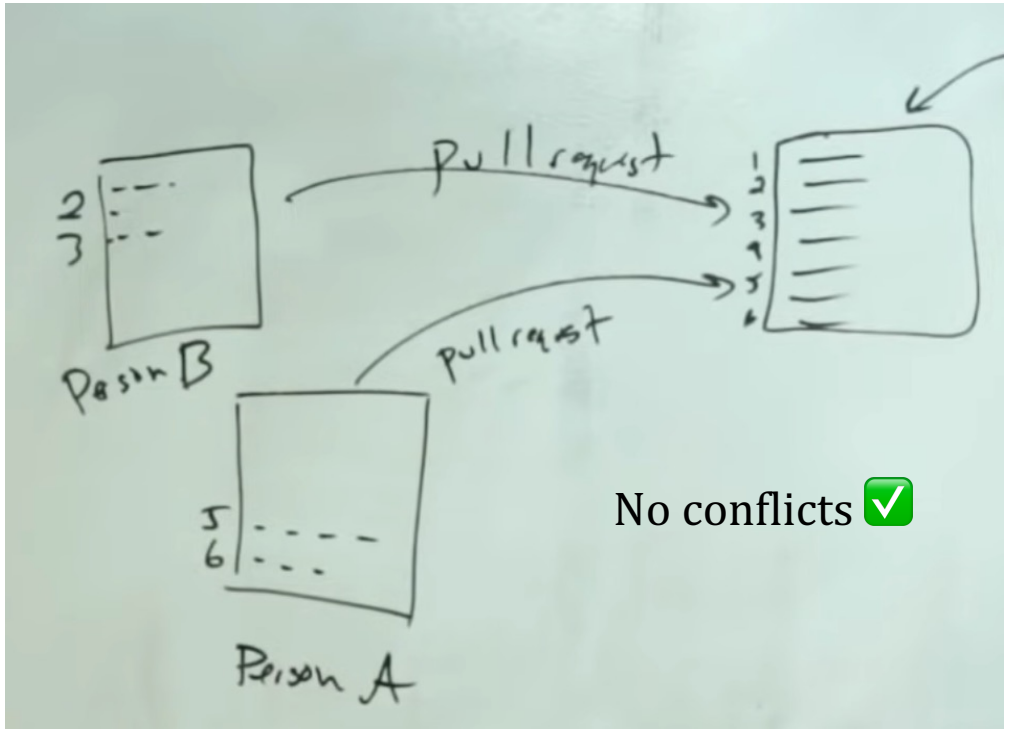
Ideal Git workflow



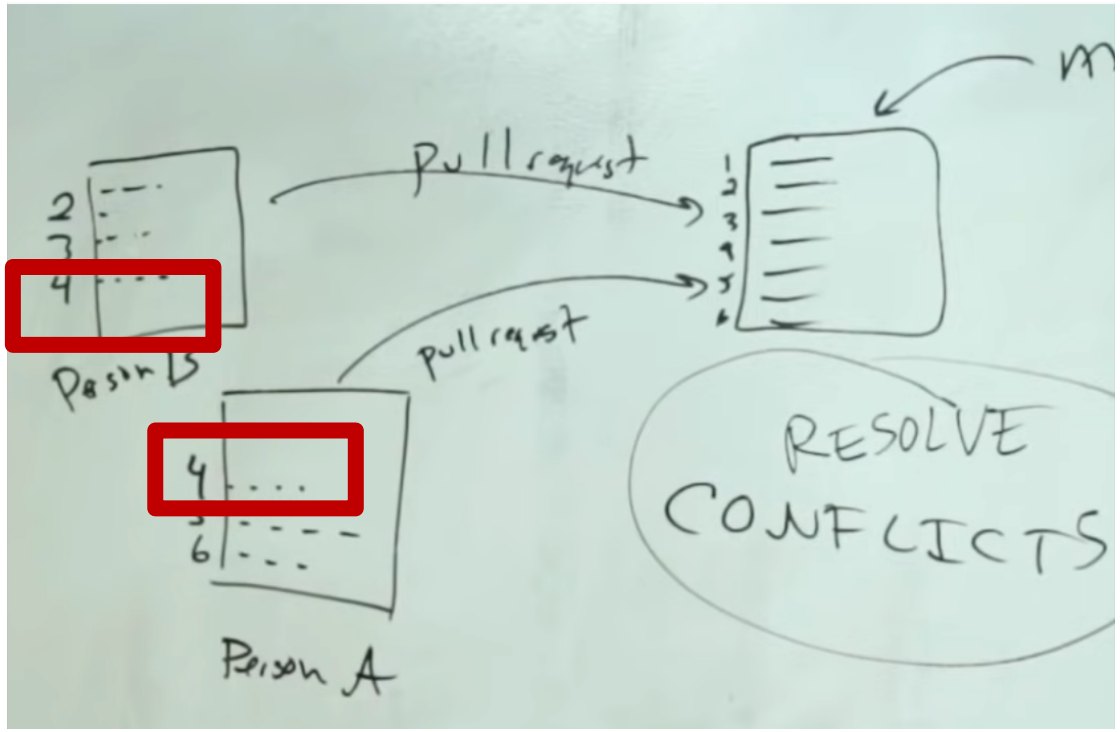
But it’s not always ideal
Conflicts may happen

How can conflict happen

https://www.youtube.com/watch?v=JtIX3HJKwfo



One person edited a file, commited and pushed
You edit the same file but different line, commit
`git pull && git push` , no problem✅



One person modified line 4, commited and pushed
You accidentally are also editing line 4, commit
`git push` says error, and `git pull` says conflict ❌

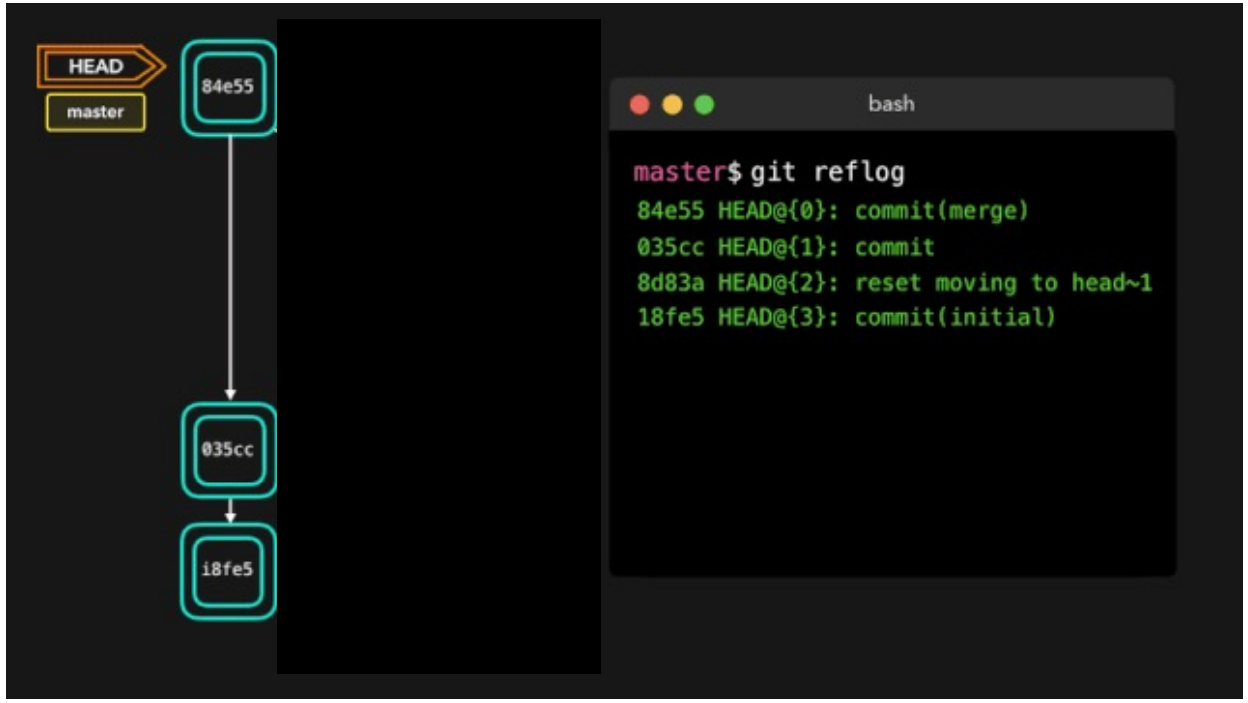
Solve conflicts

- How can conflict happen?
  - Same file
  - Same line
  - Different content (by different people)
  - When you pull

- How to solve?
  - Simply edit the conflicted file, and commit again

Time machine travel!

- `git reset [ID] --hard`
  - travel to a snapshot
  - `git log --pretty=oneline`
- `git reflog`
  - see where I travel from
- Travel to present, in our tutorial
  - `git reset 89ac0d9 --hard`
  - `git log --pretty=oneline`



For core developers

You can directly modify the repo

1. `git clone git@github.com:kaiwu-astro/Nbody6PPGPU-beijing`
2. `cd Nbody6PPGPU-beijing`
3. `git switch dev`

4. do anything. e.g. fix a bug
5. `git add .; git commit -m “briefly introduce what you have done”`
6. `git pull --all; git push`
7. `goto 4`

I do something by accident! No worries

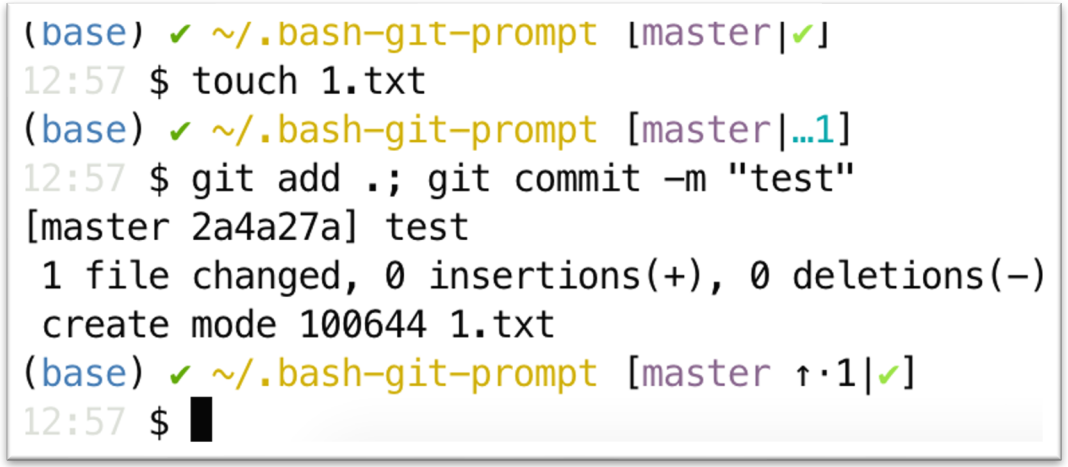
- https://ohshitgit.com/
- Help with
  - I need to change the message on my last commit
  - I accidentally committed something to master that should have been on a brand-new branch
  - I accidentally committed to the wrong branch
  - ...
- Any other trouble: lots of solutions on Google and StackOverflow

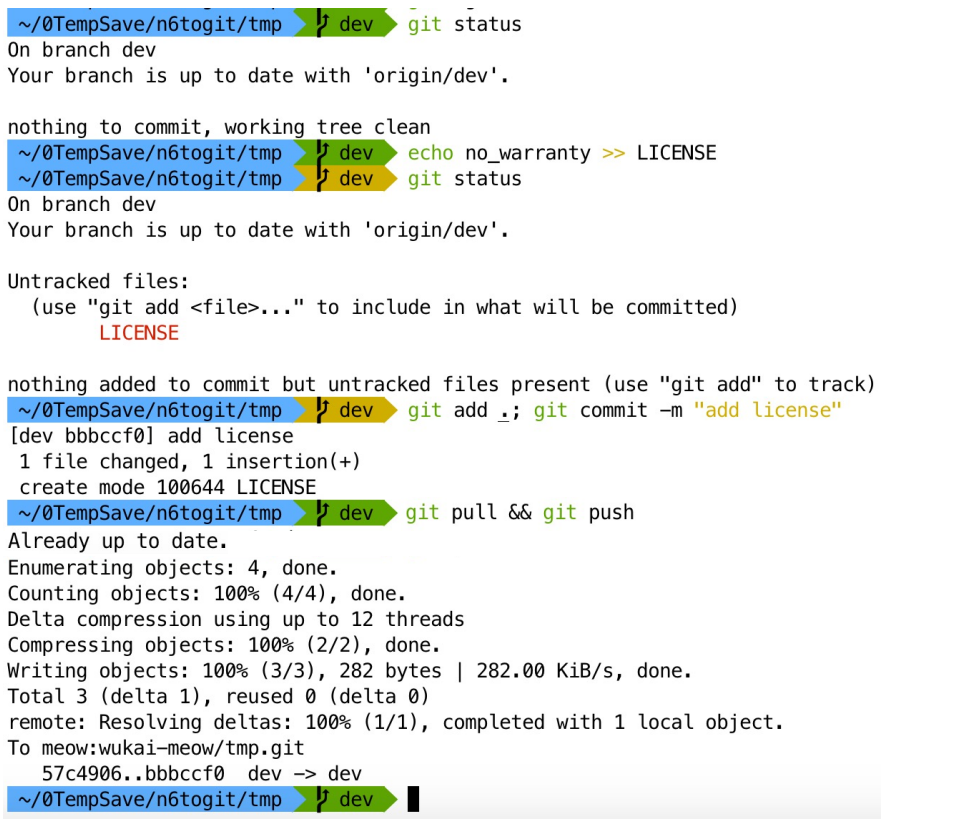
Dos and don’ts

- Dos:
  - often run `git pull`, to avoid conflicts
  - commit timely (but ideally < 10 per day)
- Don’ts
  - Don’t commit to main/master/stable branches directly
  - Don’t use `git push –f` .
    - Use new commit to undo remote change!
  - Don’t use rebase or squash: use them only when you understand

Make Git command line more productive

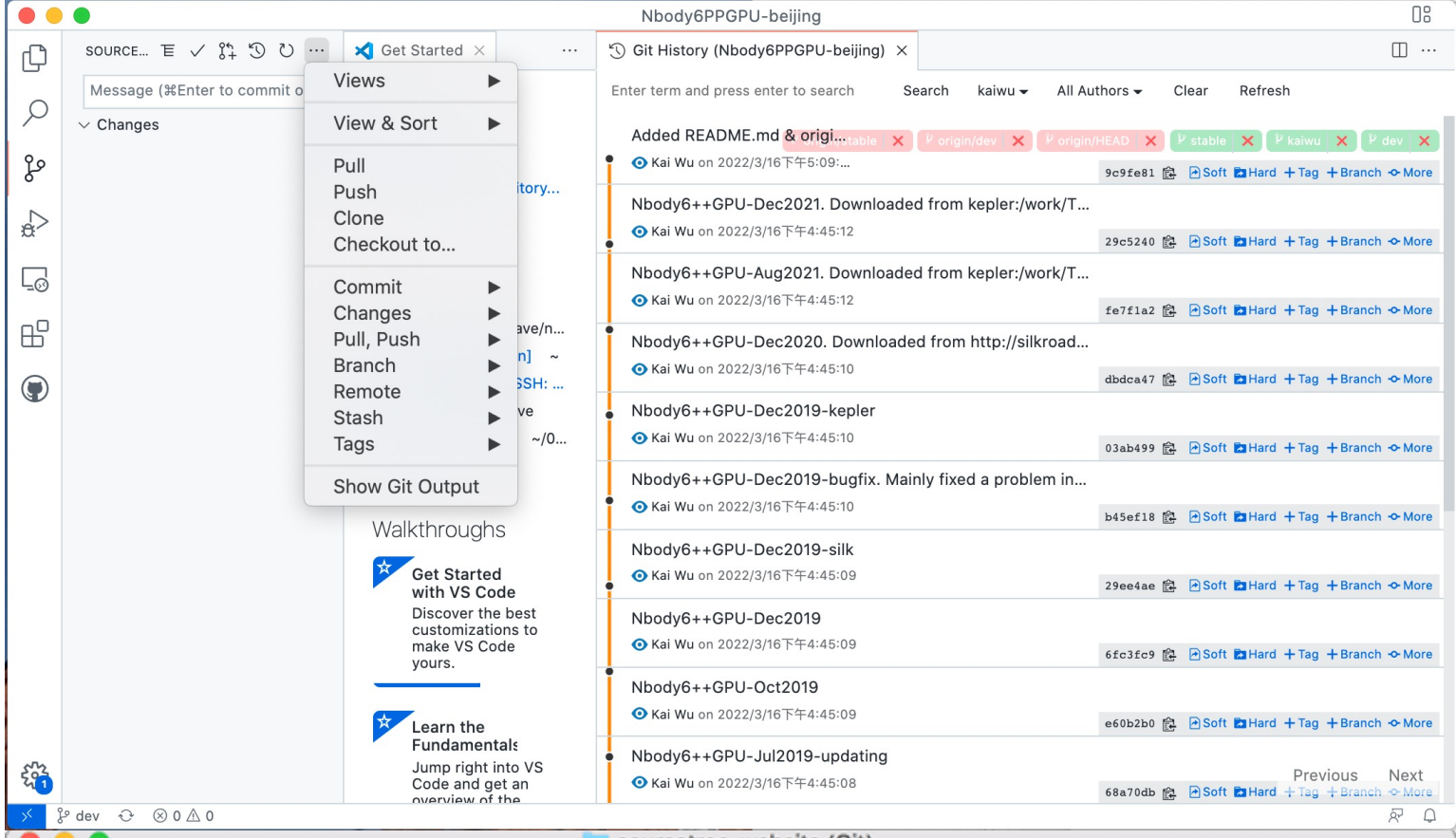
- bash
  - https://github.com/magicmonty/bash-git-prompt
- zsh
  - oh-my-zsh: https://ohmyz.sh/ or https://github.com/ohmyzsh/ohmyzsh
    - guide to install on-my-zsh https://ivanaugustobd.medium.com/your-terminal-can-be-much-much-more-productive-5256424658e8

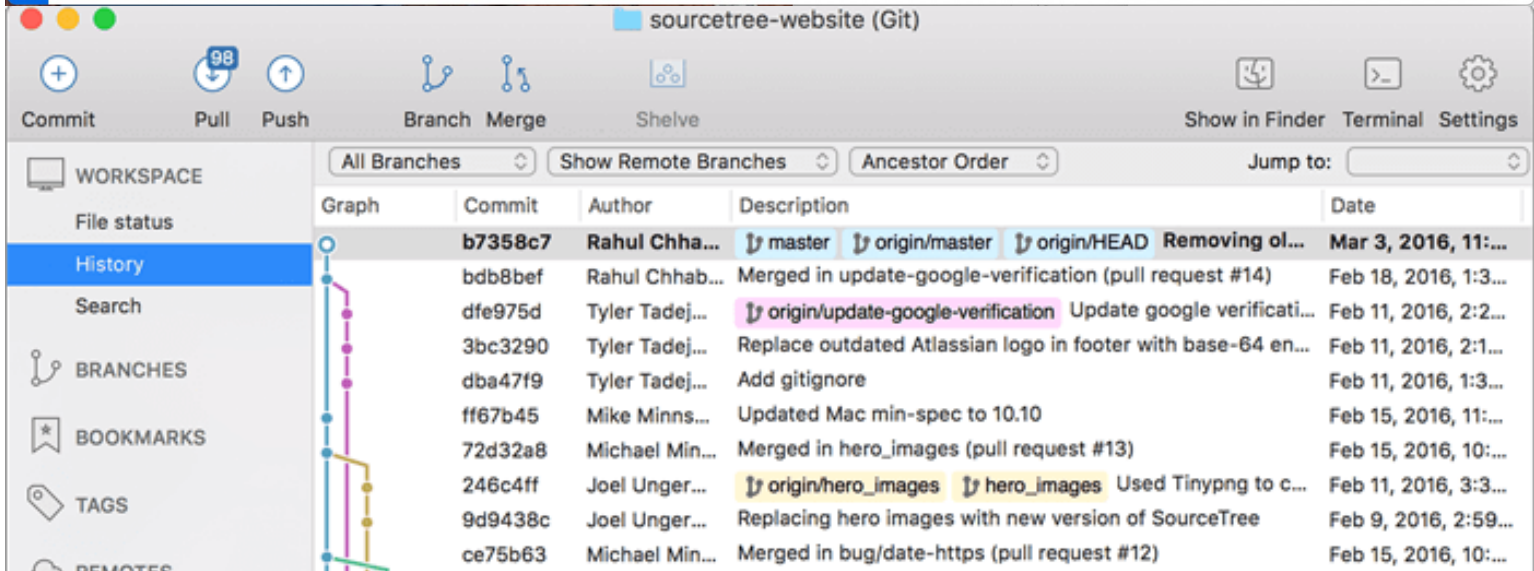




Visualize Git/ GUI tool

- Use VSCode
  - Win, Mac, Linux
  - Built in Git management
  - Download extension: Git History
- Use GitKraken
  - Win, Mac, Linux
- Use Sourcetree
  - Win, Mac





Development

- Track updates:
  - Watch the repo on GitHub. Email notification on every change →
- Be a contributor
  - [For seldom participate developers] fork the repo and use pull request to submit modifications
    - https://docs.github.com/en/get-started/quickstart/fork-a-repo
    - https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request
  - [For developers] send your GitHub username or GitHub account email to me (via the chats of our meeting room now or later by email Kai.Wu19@student.xjtlu.edu.cn)

[lwang-astro/PeTar] typo corrections -nsflag (PR #29)

SaraRastello 3月

至： lwang-astro/PeTar 抄送： Subscribed

You can view, comment on, or merge this pull request online at:

https://github.com/lwang-astro/PeTar/pull/29

Commit Summary

- 72d001e typo corrections -nsflag

File Changes

(1 file)

- **M** bse-interface/bse_interface.h (2)

Patch Links:

- https://github.com/lwang-astro/PeTar/pull/29.patch
- https://github.com/lwang-astro/PeTar/pull/29.diff

—

Reply to this email directly, view it on GitHub, or unsubscribe.
Triage notifications on the go with GitHub Mobile for iOS or Android.
You are receiving this because you are subscribed to this thread.

You are receiving this because you are subscribed to this thread.

End of tutorial. Thank you!

Git command cheatsheets

1. https://github.com/kaiwu-astro/garage/raw/main/GitHub_Education-git-cheat-sheet.pdf
2. https://github.com/kaiwu-astro/garage/raw/main/Atlassian-Git-Cheatsheet.pdf
3. https://github.com/kaiwu-astro/garage/raw/main/Youtube-Git-Cheat-Sheet.pdf

Auto test (continuous integration, CI)

- Automatically configure, make and run test simulations every time when any changes happen on GitHub (when you do `git push` or pull request is accepted)
- Now it's set to check any errors raised during compile or run. If needed, it's also possible to validate the physics by checking output (i.e. we expect the half-mass radius to be between some values)

Time	N	input	OS	Compiler	Platform
0.5Myr	1k	no dat.10	Ubuntu18.04	GCC-9	GitHub Actions 4 GPU, 8G RAM
			Ubuntu20.04	GCC-5	
?	?	Any	other	ideas	?

Public repo or private?

Both: we control who can modify

- Public repo
  - anyone can see it and track changes
- Private repo
  - we control who can see and track

New name?

- Now is https://github.com/kaiwu-astro/Nbody6PPGPU-beijing
  - We can rename it anytime we like. After renaming, old link will be an alias
- Any other idea?

worknb6++-save1
Nbody6++GPU-Maria-2020.6
...

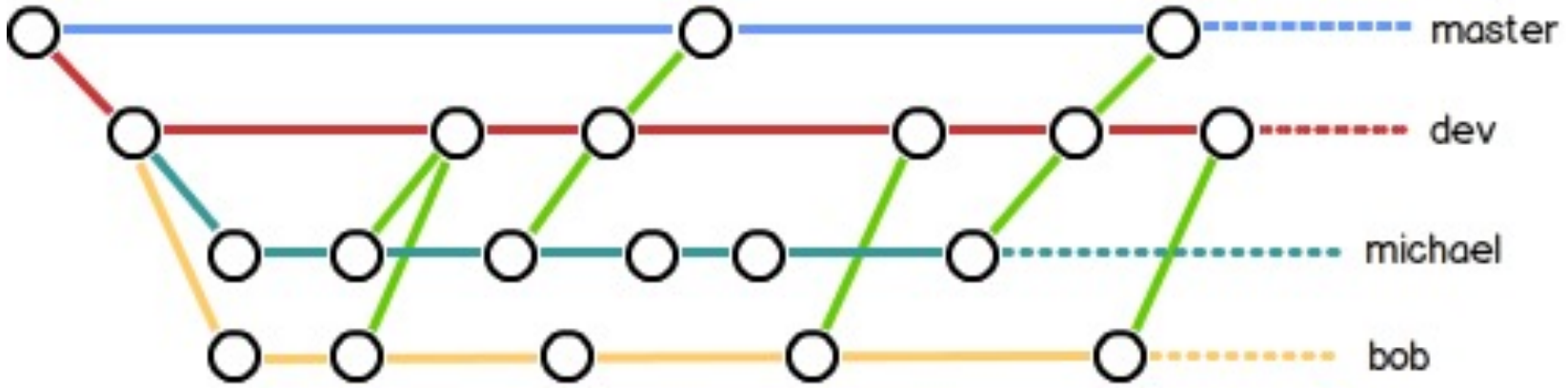
(parallelly upload 110 folders in the archive)
...
Nbody6new
Nbody6++GPU-Oct2021-massless

structure of Qi's repo

Nbody6++GPU-Dec2017
↓
Nbody6++GPU-Sep2018
↓
Nbody6++GPU-Feb2019
↓
...
(Nbody6++GPU main working version after Dec2017 in order of time (total number: 13))
...
↓
Nbody6++GPU-Dec2019bugfix
↓
Nbody6++GPU-Dec2020
↓
Nbody6++GPU-Aug2021
↓
Nbody6++GPU-Dec2021

structure of Kai's repo

New branches



- stable (main branch)
  - Stable versions, like Nbody6++GPU-Dec2021.
  - (the master branch I show last time was renamed ”trash” and will be removed, because it does not include original modification time of each file)
- dev
  - bugfix and new features to stable version should be merged to dev branch first
  - merge to stable after tests by our members
- ”your branch”
  - create based on dev branch
  - work directly on it. Merge to dev when finish
- (to do, if needed) NBody6++
  - this branch is mainly for archive and storytelling, from Sverre Aarseth’s NBody6 to Rainer’s NBody6++, then to first version of NBody6++GPU
  - (I need support to know the sequence of nbody6++ folders, can Rainer or anyone else help?)
- massless? stardisk? author please create?
  - use `git cherry-pick` to apply bugfix from stable or dev