

GitHub

Aiichiro Nakano

*Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
Department of Biological Sciences
University of Southern California*

Email: anakano@usc.edu

Goal: Use GitHub for version control, collaborative software development & dissemination (digital business card)



GitHub

- **GitHub is a code hosting platform for version control & collaboration. Please sign up at:** <https://github.com/>
- **Read “Chapter 15—Local version control” & “Chapter 16—Remote version control” of *Effective Computation in Physics* by Scopatz and Huff; USC students have free access to the book through Safari Online:** <https://libraries.usc.edu/databases/safari-books>
- **Software Carpentry has a good tutorial on “Version control with Git”:** <http://swcarpentry.github.io/git-novice>
- **How to create a README file for your repository using the Markdown language:** <https://www.markdownguide.org/cheat-sheet>

Version Control with Git

- **Git:** Distributed version-control system software
- While your own computer may have Git installed, here we will use Discovery cluster as an example

- **Getting help**

```
~$ git --help
```

- **Configuring Git**

```
~$ git config -global user.name "XXX"
```

Choose your username

```
~$ git config -global user.email "your_ID@usc.edu"
```

Your USC ID

```
~$ git config -global core.editor "vim"
```

Prompt, indicating you are working
in your home directory (~)

Text editor to be used, like
nano, emacs, vim

Create a Local Repository

- As an example, let us create a directory named SimpleMD and populate it with simple molecular-dynamics (MD) files: md.c, md.h, md.in

- Enter the SimpleMD directory, and type:

```
~/SimpleMD $ git init
```

Create an empty Git repository

```
~/SimpleMD $ git add *
```

Stage all files in the directory to be tracked by Git

```
~/SimpleMD $ git commit
```

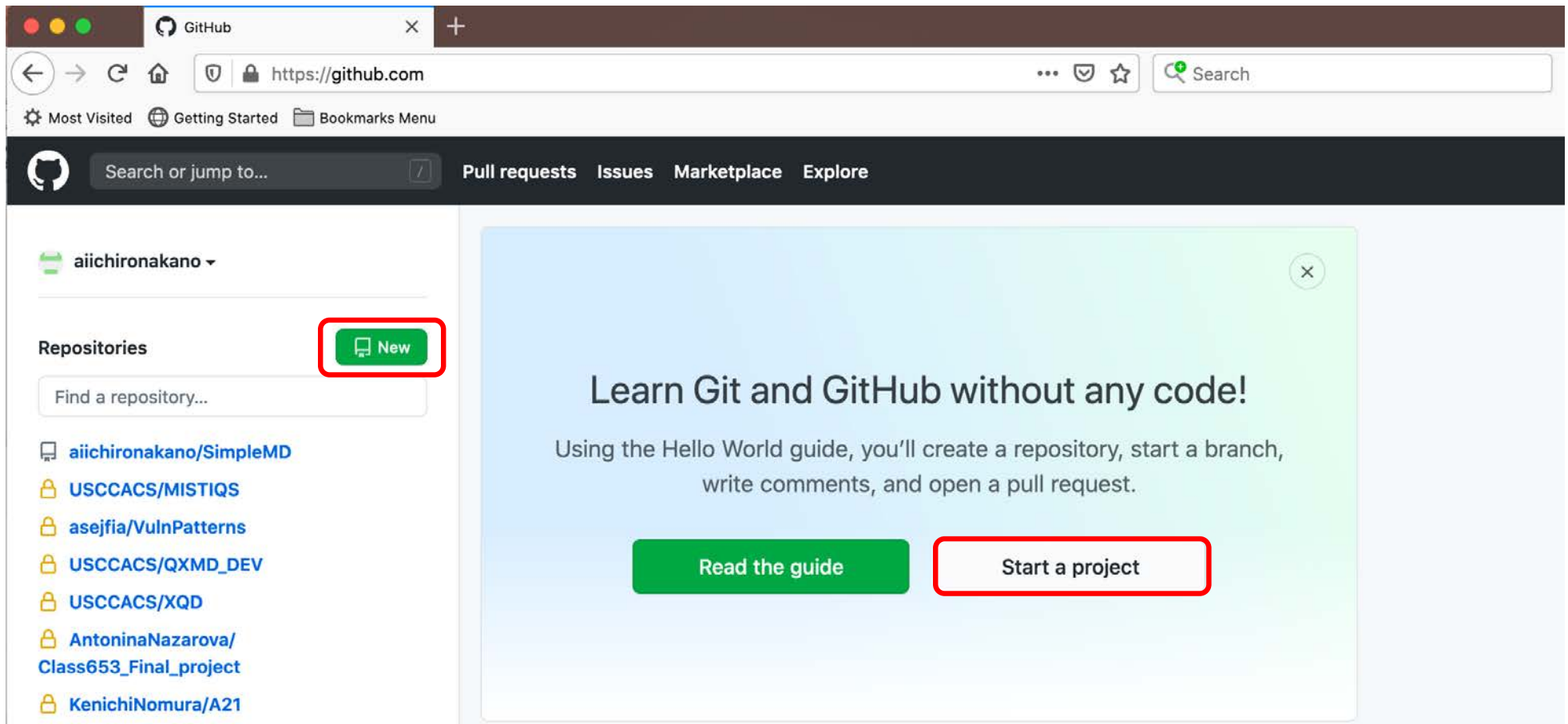
Record changes to the repository

You will be asked to enter a comment in the text editor you have configured earlier; add an informative comment like “repository created” or “file XXX modified with a feature YYY”

- **Branch:** Git allows you to create multiple branches of the repository to be tracked in parallel, using the `git branch` command. When a repository is initialized, a single branch called “master” is automatically created.

Create a GitHub Repository

- **Repository:** Used to organize a project; it can contain folders, files, data, images, *etc.*
- **Create a GitHub repository:** Click “New” repository or “Start a project” button after signing in github.com.



Declare a Remote

- Suppose you have created a GitHub repository named SimpleMD under your GitHub account (in my case aiichironakano), the URL (uniform resource locator) is `https://github.com/aiichironakano/SimpleMD.git`
- In the previous example of the SimpleMD directory on Discovery, type:

```
~/SimpleMD $ git remote add origin https://github.com/aiichironakano/SimpleMD.git
```

— Create an alias named “origin” for the remote repository, with which the local copy to be synchronized

```
~/SimpleMD $ git push origin master
```

Push commits in the master branch of the local repository to the remote repository
- Other GitHub users can now clone (*i.e.*, create a local copy of) your online repository by typing:


```
~/ $ git clone https://github.com/aiichironakano/SimpleMD.git
```
- For the other user to retrieve updated commits into his/her local master branch in later times, that user should type (origin is the automatically-created alias):

```
~/ $ git pull origin master
```

Use version control for team final projects

Create a README File

- Each repository has a README file that concisely summarizes the software
- Here is a sample README.md file written in the Markdown language:

```
# SimpleMD: Simple Molecular Dynamics  heading level 1
This is a readme file for a simple molecular dynamics (MD) program
for Lennard-Jones potential.
## 0. Prerequisites  heading level 2
Only needed is C compiler.
## 1. How to compile and run
If the C compiler on your computer is cc (also common is gcc for Gnu C
  compiler), type:
cc -O -o md md.c -lm
This will create an executable named md. To run the executable, type:
./md < md.in
## 2. Files
The following files are included in this folder, in addition to this readme
  file, readme.md.
<ul>  unordered list
<li>md.c: Main C program</li> List item
<li>md.h: Header file for md.c</li>
<li>md.in: Input parameter file (to be redirected to the standard input)</li>
</ul>
! [Screen shot of MD simulation](ScreenShot.png)

The image file, ScreenShot.png, needs be
placed in the repository along with
README.md
```

Every final project needs to create a README file

See <https://www.markdownguide.org/cheat-sheet>

README in Web Browser

SimpleMD: Simple Molecular Dynamics # (heading level 1)

This is a readme file for a simple molecular dynamics (MD) program for Lennard-Jones potential.

0. Prerequisites ## (heading level 2)

Only needed is C compiler.

1. How to compile and run

If the C compiler on your computer is cc (also common is gcc for Gnu C compiler), type:

```
cc -O -o md md.c -lm
```

This will create an executable named md. To run the executable, type:

```
./md < md.in
```

2. Files

The following files are included in this folder, in addition to this readme file, readme.md.

- md.c: Main C program
- md.h: Header file for md.c
- md.in: Input parameter file (to be redirected to the standard input)

} (unordered list)



! (image)

Fancier example at <https://github.com/USCCACS/QXMD>, including user's manual

Git Cheat Sheet (1)

Install

GitHub for Windows

<https://windows.github.com>

GitHub for Mac

<https://mac.github.com>

Git for All Platforms

<http://git-scm.com>

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Configure tooling

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
  Sets the name you want attached to your commit transactions

$ git config --global user.email "[email address]"
  Sets the email you want attached to your commit transactions

$ git config --global color.ui auto
  Enables helpful colorization of command line output
```

Branches

Branches are an important part of working with Git. Any commits you make will be made on the branch you're currently "checked out" to. Use `git status` to see which branch that is.

```
$ git branch [branch-name]
  Creates a new branch

$ git checkout [branch-name]
  Switches to the specified branch and updates the
  working directory

$ git merge [branch]
  Combines the specified branch's history into the
  current branch. This is usually done in pull requests,
  but is an important Git operation.

$ git branch -d [branch-name]
  Deletes the specified branch
```

Create repositories

When starting out with a new repository, you only need to do it once; either locally, then push to GitHub, or by cloning an existing repository.

```
$ git init
  Turn an existing directory into a git repository

$ git clone [url]
  Clone (download) a repository that already exists on
  GitHub, including all of the files, branches, and commits
```

The .gitignore file

Sometimes it may be a good idea to exclude files from being tracked with Git. This is typically done in a special file named `.gitignore`. You can find helpful templates for `.gitignore` files at github.com/github/gitignore.

Synchronize changes

Synchronize your local repository with the remote repository on GitHub.com

```
$ git fetch
  Downloads all history from the remote tracking branches

$ git merge
  Combines remote tracking branch into current local branch

$ git push
  Uploads all local branch commits to GitHub

$ git pull
  Updates your current local working branch with all new
  commits from the corresponding remote branch on GitHub.
  git pull is a combination of git fetch and git merge
```

Git Cheat Sheet (2)

Make changes

Browse and inspect the evolution of project files

```
$ git log
Lists version history for the current branch
```

```
$ git log --follow [file]
Lists version history for a file, including renames
```

```
$ git diff [first-branch]...[second-branch]
Shows content differences between two branches
```

```
$ git show [commit]
Outputs metadata and content changes of the specified commit
```

```
$ git add [file]
Snapshots the file in preparation for versioning
```

```
$ git commit -m "[descriptive message]"
Records file snapshots permanently in version history
```

Redo commits

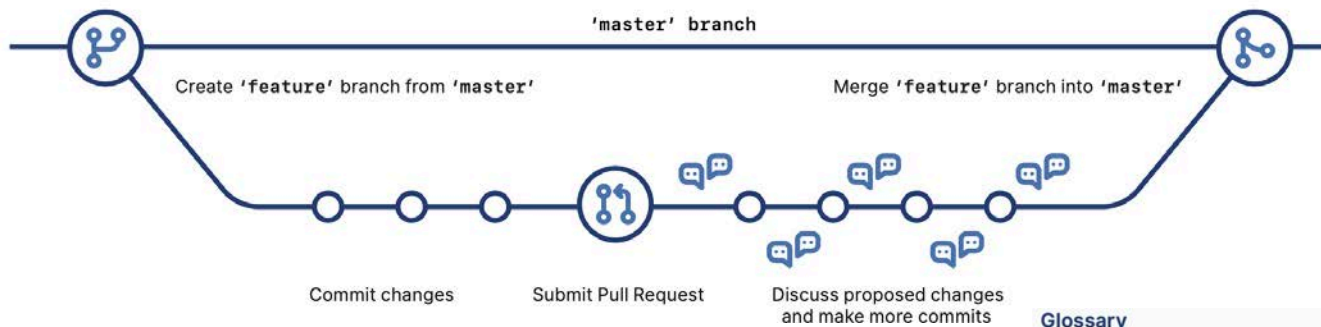
Erase mistakes and craft replacement history

```
$ git reset [commit]
Undoes all commits after [commit], preserving changes locally
```

```
$ git reset --hard [commit]
Discards all history and changes back to the specified commit
```

CAUTION! Changing history can have nasty side effects. If you need to change commits that exist on GitHub (the remote), proceed with caution. If you need help, reach out at [github.community](https://github.com/community) or contact support.

GitHub Flow



Glossary

git: an open source, distributed version-control system

GitHub: a platform for hosting and collaborating on Git repositories

commit: a Git object, a snapshot of your entire repository compressed into a SHA

branch: a lightweight movable pointer to a commit

clone: a local version of a repository, including all commits and branches

remote: a common repository on GitHub that all team member use to exchange their changes

fork: a copy of a repository on GitHub owned by a different user

pull request: a place to compare and discuss the differences introduced on a branch with reviews, comments, integrated tests, and more

HEAD: representing your current working directory, the HEAD pointer can be moved to different branches, tags, or commits when using `git checkout`