Git Tutorial

Version Control on Projects

Weidao Lee
Data Science Degree Program
National Taiwan University and Sinica

Junting Chen
Data Science Degree Program
National Taiwan University and Sinica

Why to Use git?

- Version control
- History of each modification (checkpoint)

```
unicorngundam@festivo ~/P/myProj> ls -1
train_v0.py
train_v1.py
train_v2.py
train_v3.py
train_v4.py
train_v5.py
utils
```

- Collaboration
- Developing new features

```
unicorngundam@festivo ~/P/myProj> ls -1
main_JT.py
main_WD.py
main_newFT.py
main_old.py
utils
utils_JT
utils_WD
```

Outline

1. Tracking Files & Making Checkpoints (commit)
2. Remote Repository Management
3. Branches

Tracking Files &

Making Checkpoint (commit)

On local Repository

Tracking Files

To use git to manage your project, you need to initialize a **git repository**

```
$ git init
```

After executing the command, a folder named `.git/` will be created

```
unicorngundam@festivo ~/P/myProj (master)> ls -1 .git/
HEAD
branches
config
description
hooks
info
objects
refs
```

Tracking Files

Tell git who you are

```
$ git config --local user.name "<user_name>"
$ git config --local user.email "<email_id>"
```

You can also make these setting global,

```
$ git config --global user.name "<user_name>"
$ git config --global user.email "<email_id>"
```

Tracking Files

Now you can check the status of the **git repository**

```
$ git status
```

```
unicorngundam@festivo ~/P/myProj (master)> ls -A1
.env
.git
.gitignore
.tmp
dataloader.py
main.py
models.py
notebooks
requirements.txt
utils
```

```
unicorngundam@festivo ~/P/myProj (master)> git status
On branch master

No commits yet

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

        .gitignore
        .tmp/
        dataloader.py
        main.py
        models.py
        notebooks/
        requirements.txt

nothing added to commit but untracked files present (use "git add"
to track)
```

Tracking Files

Ignoring some files or directories from tracking

```
$ vi .gitignore
```

Adding paths

```
# vim caches
**/*~
**/*.swp
**/*.swo
```

https://github.com/github/gitignore/blob/master/Python.gitignore

Tracking Files

Now you can track some important files

```
$ git add main.py
```

Note that git track the **modifications** (or changes) of the file instead, hence you need to execute the `git add` each time after editing.

```
unicorngundam@festivo ~/P/myProj (master)> git add main.py
unicorngundam@festivo ~/P/myProj (master)> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   main.py
```

Making Checkpoints

Now you need to make the checkpoint of current status, or say **commit**

```
$ git commit -m “add main.py”
```

Now check the **commit history** by

```
$ git log
```

```
unicorngundam@festivo ~/P/myProj (master)> git commit -m "add main.py"
[master (root-commit) 854e7be] add main.py
 1 file changed, 5 insertions(+)
 create mode 100644 main.py
unicorngundam@festivo ~/P/myProj (master)> git log
commit 854e7beabfb64e8711d03c61a18e34e7bd137fcd (HEAD -> master)
Author: hankchen1728 <hankchen1728@gmail.com>
Date:   Wed Dec 9 07:03:28 2020 +0800

    add main.py
```

Making Checkpoints

Untrack some files

```
$ git rm --cached main.py
```

Warning: If the flag `--cached` is not attached, the file will be deleted directly

```
unicorngundam@festivo ~/P/myProj (master)> git rm --cached main.py
rm 'main.py'
unicorngundam@festivo ~/P/myProj (master)> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    main.py
```

Tracking Files

Show all files in repo

```
$ git ls-tree --full-tree -r --name-only <commit hash>
```

Or

```
$ git ls-files
```

```
unicorngundam@festivo ~/P/myProj (master)>
git ls-tree --full-tree -r --name-only HEAD
.gitignore
dataloader.py
main.py
models.py
```

Tracking Files

Show all files (or files with modification) added in certain **commit**

```
$ git diff-tree -r --name-only <commit hash>
```

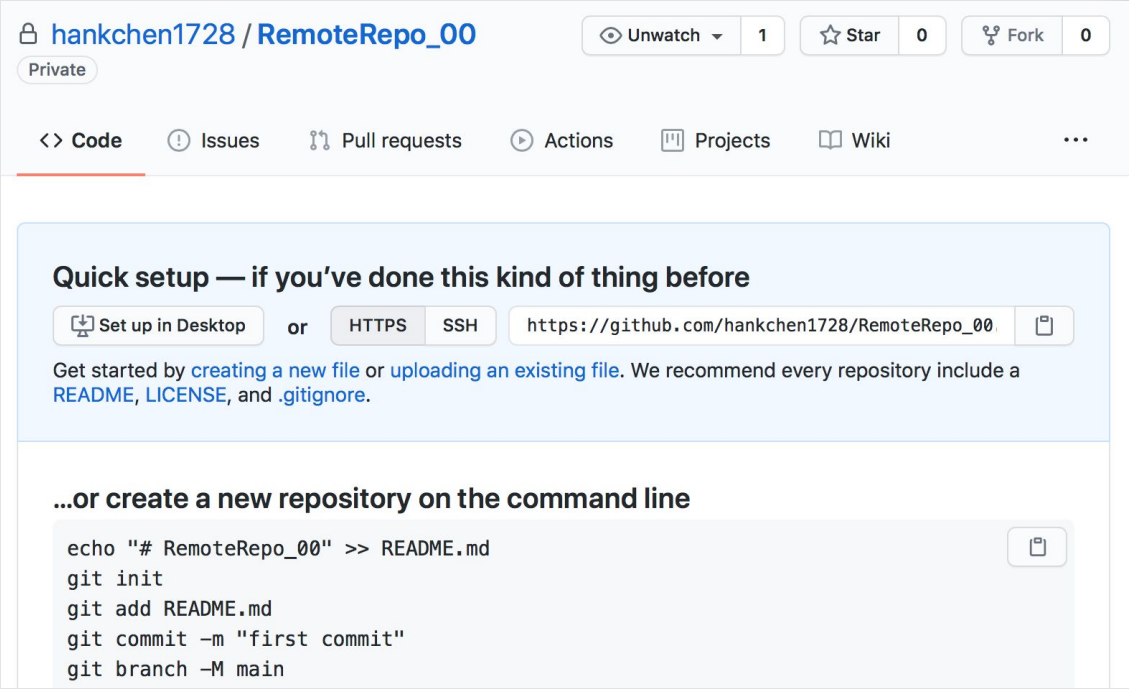
```
unicorngundam@festivo ~/P/myProj (master)> vim main.py
unicorngundam@festivo ~/P/myProj (master)> git add main.py
unicorngundam@festivo ~/P/myProj (master)> git commit -m "edit main.py"
[master df0b495] edit main.py
 1 file changed, 1 insertion(+)
unicorngundam@festivo ~/P/myProj (master)>
git ls-tree --full-tree -r --name-only HEAD
.gitignore
dataloader.py
main.py
models.py
unicorngundam@festivo ~/P/myProj (master)>
git diff-tree --no-commit-id --name-only -r df0b495
main.py
```

Remote Repository Connection

Connection with Github

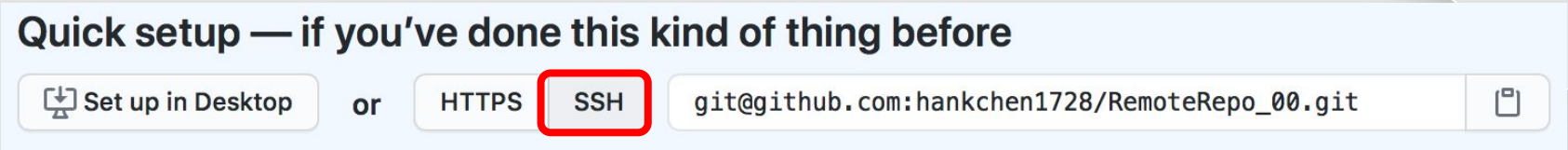
Remote Repository

Now consider that you want to push your repo to **Github**, and you have created new repository



Remote Repository

Take a look at the url link



Remote Repository

Add the **url** of the remote repo to git

```
$ git remote add <name> <remote_url>
```

```
unicorngundam@festivo ~/P/myProj (master)>
git remote add amazing git@github.com:hankchen1728/RemoteRepo_00.git
unicorngundam@festivo ~/P/myProj (master)> git remote -v
amazing git@github.com:hankchen1728/RemoteRepo_00.git (fetch)
amazing git@github.com:hankchen1728/RemoteRepo_00.git (push)
```

Remote Repository

You can also information of remote repository

```
$ git remote show <name>
```

```
unicorngundam@festivo ~/P/myProj (master)> git remote show amazing
Enter passphrase for key '/home/u/unicorngundam/.ssh/id_rsa_github':
* remote amazing
  Fetch URL: git@github.com:hankchen1728/RemoteRepo_00.git
  Push  URL: git@github.com:hankchen1728/RemoteRepo_00.git
  HEAD branch: (unknown)
```

Remote Repository

Reset the url of exist remote name

```
$ git remote set-url <name> <new url>
```

Get the url of exist remote

```
$ git remote get-url <name>
```

```
unicorngundam@festivo ~/P/myProj (master)>
git remote set-url amazing https://github.com/hankchen1728/RemoteRepo_00.git
unicorngundam@festivo ~/P/myProj (master)> git remote get-url amazing
https://github.com/hankchen1728/RemoteRepo_00.git
```

Remote Repository

Push the commit to remote repo

```
$ git push <remote name> <branch>
```

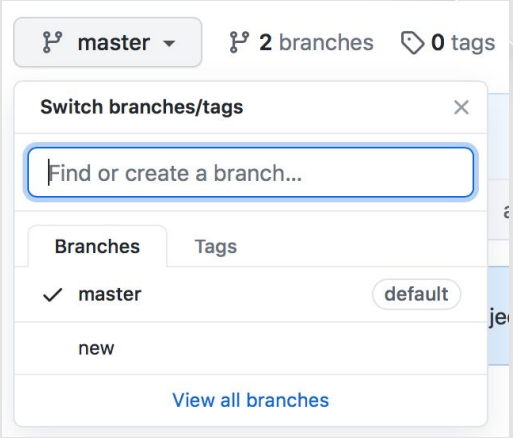
```
unicorngundam@festivo ~/P/myProj (master)> git push amazing master
Enter passphrase for key '/home/u/unicorngundam/.ssh/id_rsa_github':
Counting objects: 3, done.
Delta compression using up to 72 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 263 bytes | 263.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:hankchen1728/RemoteRepo_00.git
 * [new branch]      master -> master
```

Remote Repository

If we want the new created branch on remote has different name to local branch

```
$ git push <remote name> <local branch>:<remote branch>
```

```
unicorngundam@festivo ~/P/myProj (master)> git push amazing master:new
Enter passphrase for key '/home/u/unicorngundam/.ssh/id_rsa_github':
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'new' on GitHub by visiting:
remote:      https://github.com/hankchen1728/RemoteRepo_00/pull/new/new
remote:
To github.com:hankchen1728/RemoteRepo_00.git
 * [new branch]      master -> new
```



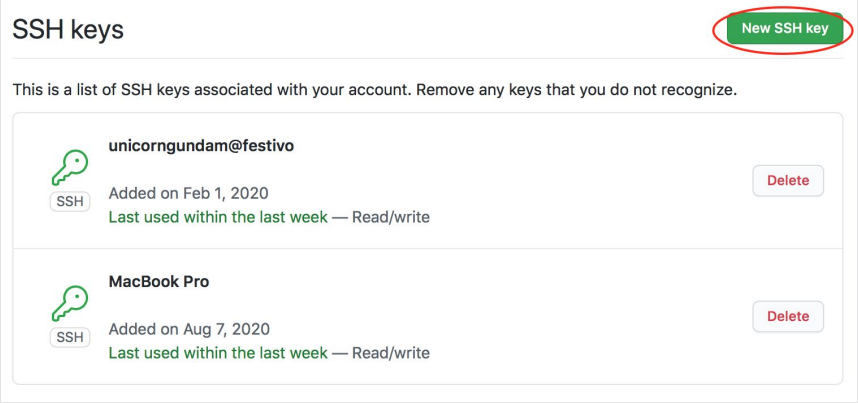
(Optional) Setup SSH Connection to Github

1. Generating a ssh key

```
$ ssh-keygen -t rsa -b 4096 -C [email]
```

2. Copy the content of your public key to setting page

```
$ cat ~/.ssh/id_rsa.pub
```



(Optional) Setup SSH Connection to Github

3. Add the SSH key to the **`ssh-agent`**

```
$ eval "$(ssh-agent -s)"
$ ssh-add ~/.ssh/id_rsa
```

4. [Optional] Configure the **`ssh`**

   If you use several keys, then you need to config your ssh as follow:

```
$ vi ~/.ssh/config
```

   Then paste the following content to the config file

```
Host github.com
    IdentityFile ~/.ssh/id_rsa
```

(Optional) Setup SSH Connection to Github

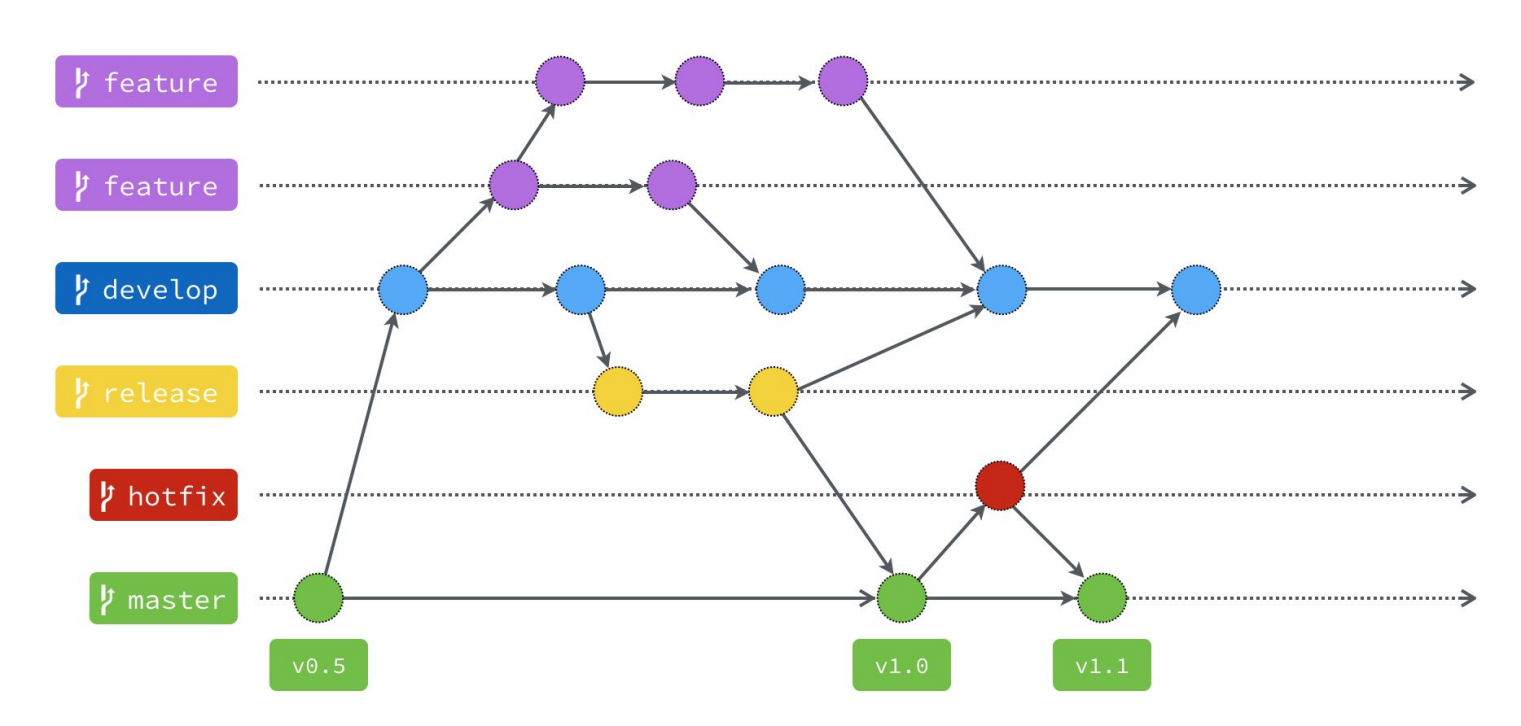
5. Testing the ssh connection

```
$ ssh -T git@github.com
```

```
unicorngundam@festivo ~/P/myProj (master)> ssh -T git@github.com
Enter passphrase for key '/home/u/unicorngundam/.ssh/id_rsa_github'
:
Hi hankchen1728! You've successfully authenticated, but GitHub does
 not provide shell access.
```

Branch Management

Branches



https://gitbook.tw/chapters/gitflow/why-need-git-flow.html

Branches

Show All the branches

```
$ git branch -a
```

```
unicorngundam@festivo ~/P/myProj (master)> git branch -a
* master
  remotes/amazing/master
  remotes/amazing/new
```

Branches

Create new branch

```
$ git branch dev
```

```
unicorngundam@festivo ~/P/myProj (master)> git branch dev
unicorngundam@festivo ~/P/myProj (master)> git branch -a
  dev
* master
  remotes/amazing/master
  remotes/amazing/new
```

Branches

Switch to new created branch

```
$ git checkout dev
```

```
unicorngundam@festivo ~/P/myProj (master)> git checkout dev
Switched to branch 'dev'
unicorngundam@festivo ~/P/myProj (dev)> git branch -a
* dev
  master
  remotes/amazing/master
  remotes/amazing/new
```

Show Current Branch in Shell Prompt

- Fish, zsh

```
unicorngundam@festivo ~/P/myProj (dev)> git branch -a
* dev
  master
  remotes/amazing/master
  remotes/amazing/new
```

- For Bash
  Paste the following code in your `.bashrc`

```
parse_git_branch() {
     git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/*
\(.*\)/(\1)/'
}
export PS1="\u@\h \[\e[32m\]\w
\[\e[91m\]\$(parse_git_branch)\[\e[00m\]$ "
```

Future Work

- Merge, Conflict
- Detached Head