Git & Github

Sunnerversets Studio, Jan 1
Qiu Yuheng

日不落开发工作室
sunneversets.studio

Outline

1. What/Why git
2. What is github
3. Git workflow & Structure
4. Git in Studio
5. Education

What, Why & How Git?
Why Github?

What is Git?

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people.

It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

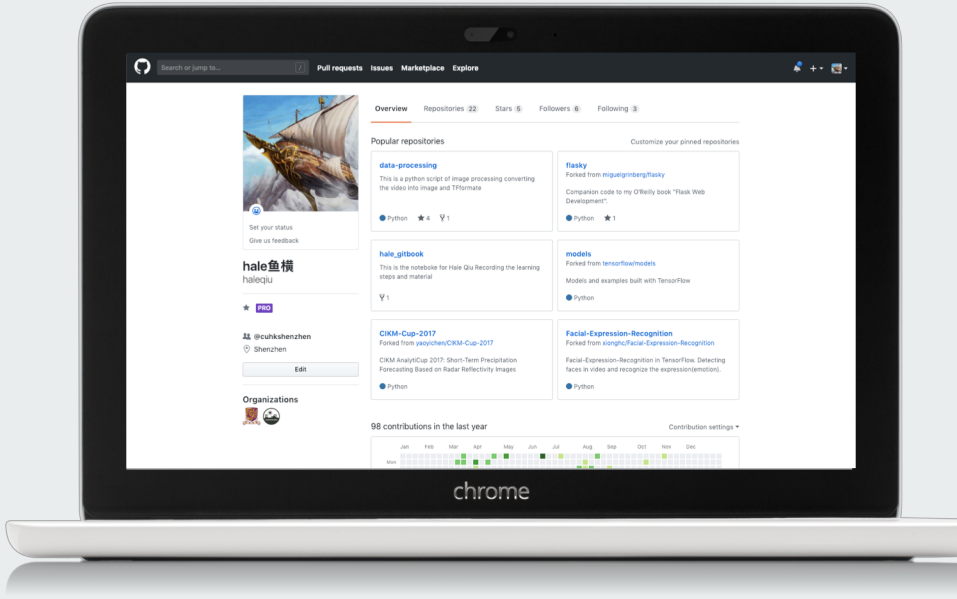
Why is Git?

It gives every developer their own local copy of the entire project. This isolated environment lets each developer work independently of all other changes to a project—they can add commits to their local repository and completely forget about upstream developments

It gives you access to Git’s robust branching and merging model. Git branches are designed to be a fail-safe mechanism for integrating code and sharing changes between repositories.

Why Github?

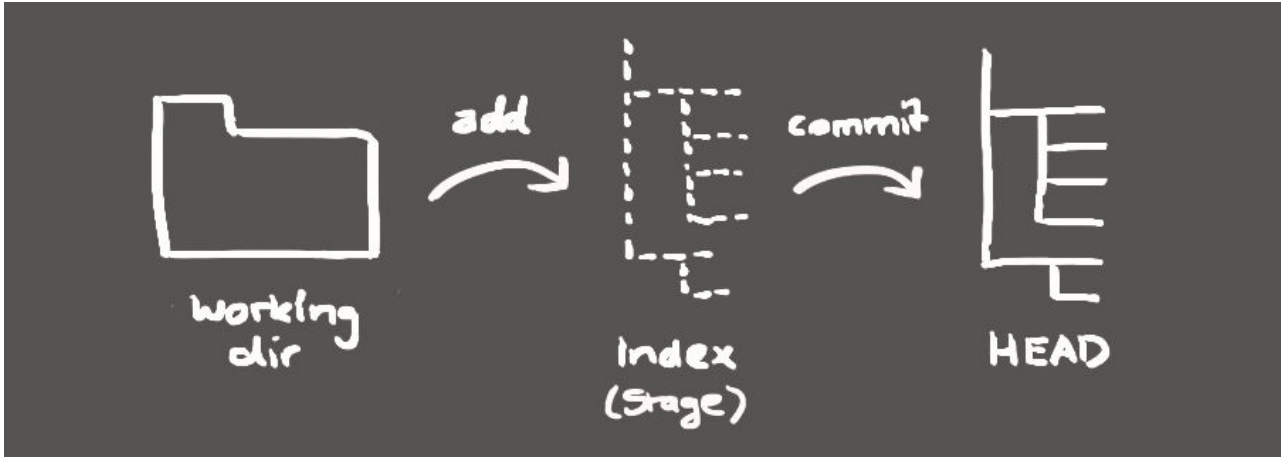
- **GitHub** is a **Git** repository hosting service, but it adds many of its own features.
- While **Git** is a command line tool, **GitHub** provides a Web-based graphical interface.
- It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.



Git Workflow & Structure

Workflow

Your local repository consists of three "trees" maintained by git. the first one is your Working Directory which holds the actual files. the second one is the Index which acts as a staging area and finally the HEAD which points to the last commit you've made.



Create a new repository

You can do it in GitHub or your local computer!

In GitHub: https://github.com/new

Local Computer: create a new directory, open it and perform a `git init` to create a new git repository.

Checkout a repository

Create a working copy of a repository by running the command

```
git clone /path/to/repository
```

For example, `git clone git@github.com:Sunneversets-Studio/resources.git`

Add & Commit

You can propose changes (add it to the Index) using

```
git add <filename>
```

```
git add *
```

```
git add -A  # add changes from all tracked and untracked files
```

This is the first step in the basic git workflow.
To actually commit these changes use

```
git commit -m "Commit message"
```

Now the file is committed to the HEAD, but not in your remote repository yet.

Replace Local Changes

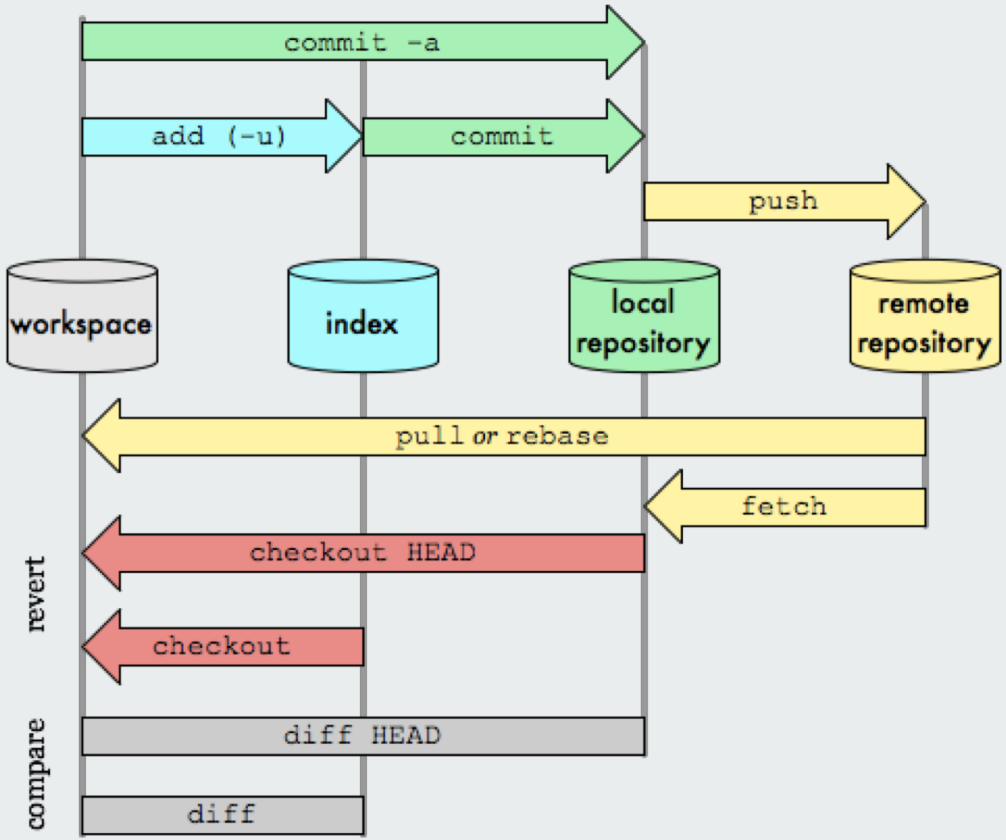
In case you did something wrong, which for sure never happens ;), you can replace local changes using the command `git checkout -- <filename>`
This replaces the changes in your working tree with the last content in HEAD. Changes already added to the index, as well as new files, will be kept.

If you instead want to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it like this

```
git fetch origin
git reset --hard origin/master
```

Git Data Transport Commands

http://osteele.com



Pushing Changes to Remote

Your changes are now in the HEAD of your local working copy.

To send those changes to your remote repository, execute `git push`

Branching

Branches are used to develop features isolated from each other. The master branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.

Create a New Branch

Create a new branch named "feature_x" and switch to it using

```
git checkout -b feature_x
```

Switch back to master

```
git checkout master
```

Delete the branch

```
git branch -d feature_x
```

A branch is not available to others unless you push the branch to your remote repository

```
git push origin <branch>
```

Update & Merge

To update your local repository to the newest commit, execute `git pull` in your working directory to fetch and merge remote changes.
To merge another branch into your active branch (e.g. master), use `git merge <branch>`

In both cases git tries to auto-merge changes. Unfortunately, this is not always possible and results in conflicts. You are responsible to merge those conflicts manually by editing the files shown by git. After changing, you need to mark them as merged with `git add <filename>` before merging changes.

You can also preview them by using `git diff <source_branch> <target_branch>`

Tagging

It's recommended to create tags for software releases. You can create a new tag named 1.0.0 by executing `git tag 1.0.0`

Git Log

In its simplest form, you can study repository history using `git log`
You can add a lot of parameters to make the log look like what you want.
To see only the commits of a certain author: `git log --author=bob`
To see a very compressed log where each commit is one line: `git log --pretty=oneline`
Or maybe you want to see an ASCII art tree of all the branches, decorated with the names of tags and branches: `git log --graph --oneline --decorate --all`

These are just a few of the possible parameters you can use. For more, see `git log --help`

How we use Git in Studio?

Forking Workflow -- How it works?

The Centralized Workflow uses a central repository to serve as the single point-of-entry for all changes to the project. The default development branch is called master and all changes are committed into this branch.

The Forking Workflow is fundamentally different than other popular Git workflows. Instead of using a single server-side repository to act as the “central” codebase, it gives every developer their own server-side repository. This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one.

You start by forking the central repository, then clone your own repo to your disk. In your own local copies of the project, you can edit files and commit changes to your own repo. If you are happy about what has been done, you can “push” local branch to your repo, then issue a pull request waiting for administrator to merge your repo into the central repository.

Code Review

All projects in the Studio will be forced to use Code Review. Once there is a pull request, another person will review the changes and leave inline comments.

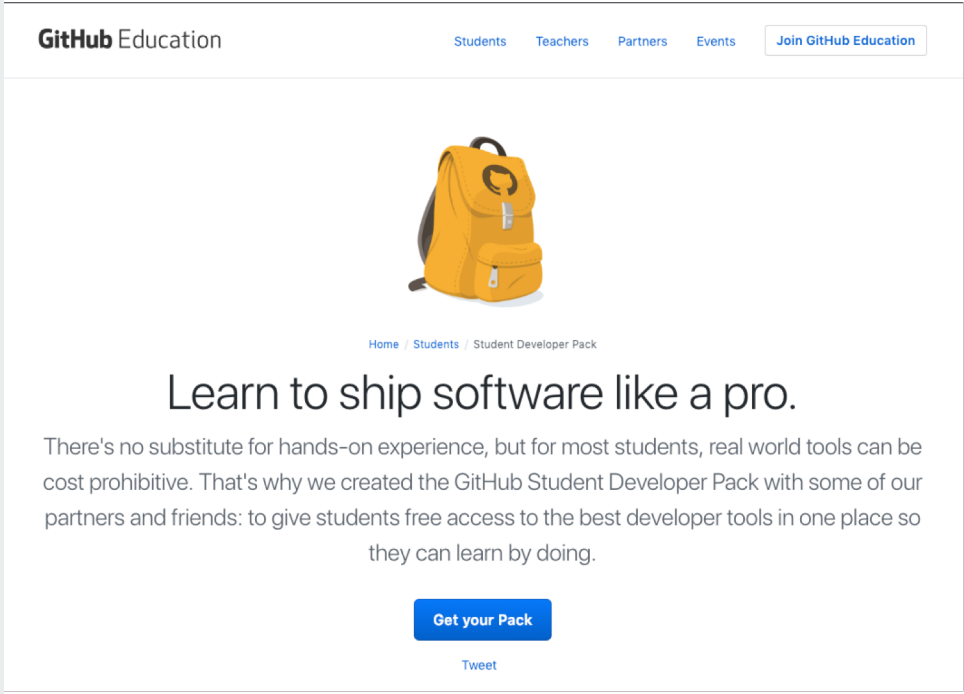
The developer will see the comments and make changes and additional commits (these new commits will show in the pull request automatically, one pull request maps to one branch)

The changes are reviewed and merged. Some merges might result in conflicts, which must be resolved by using the command line. Once merged and pushed, the pull request will be closed automatically.

Github Education pack

https://education.github.com/pack

Wait for several days, will inform you through email.



Enjoy your Journey in Studio

Join us?
Emile us with your CV
sunneversets.studio@gmail.com

Questions?

References

1. git - the simple guide: http://rogerdudler.github.io/git-guide/index.html
2. git- Wikipedia: https://en.wikipedia.org/wiki/Git
3. Forking Workflow: https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow
4. Structure: http://crackfree.github.io/2017/03/10/Depth-comprehension-of-git-structure-and-conception-md/
5.https://www.liaoxuefeng.com/wiki/001373951630592960 6dd18361248578c67b8067c8c017b000/001373962845513aefd77a99f4145f0a2c7a7ca057e7570000