Workshop Git & GitHub

@araramaker
Twitter: MtChicao
Blog: araramakerspace.github.io/blog
Discord : https://discord.gg/fS66Kk

ARARA maker space

Goals

- Create a maker culture
- Create a git & github culture
- Learn about git & github

What is version control

- Version control system are a category of software tools that help a software team manage changes to source code over time.
- Software developers working in teams are continually writing new source code and changing existing source code.
- Version control systems are all about managing contributions between multiple distributed authors (usually developers).

Git

is a distributed version control system, used mainly in software development, but can be used to record the history of edits of any file type . Git was originally designed and developed by Linus Torvalds for Linux kernel development, but was adopted by many other projects.

What is Git and why use it?

- Why use Git & GitHub?
  - Centralized cloud storage of your code.
  - Version Control.
  - Working in teams.
  - Get involved / Open Source.
  - Bettering your code.
  - Show off
  - You're gonna need it anyway

ARARA maker space

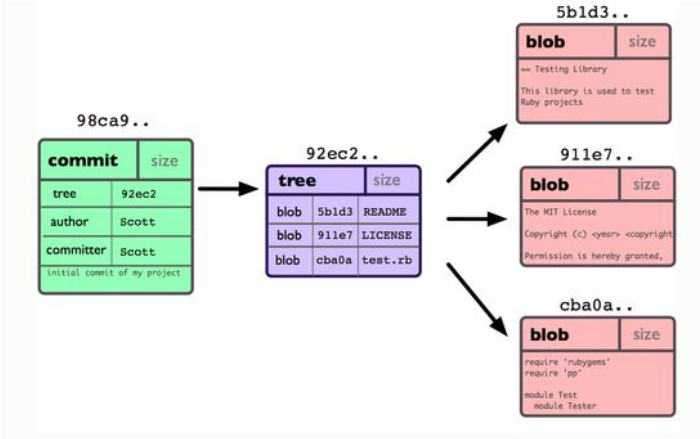
What is repository?

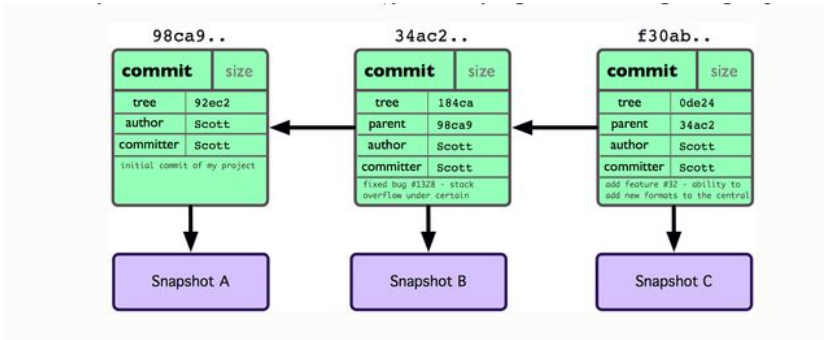
- .git/
  - Added
  - Changed
  - Renamed
  - Removed

ARARA maker space

Branching

- Branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.





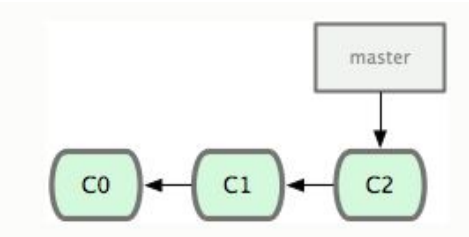
ARARA maker space

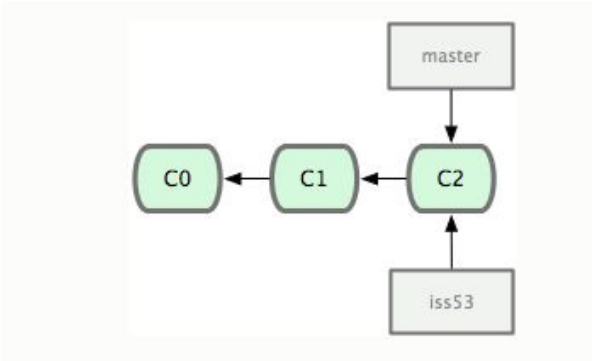
Branching

```git checkout -b iss53```

Add something and commit

```git commit -a -m ‘add a new file [issue 53]```





Merging

Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

Note that all of the commands presented below merge into the current branch. The current branch will be updated to reflect the merge, but the target branch will be completely unaffected. Again, this means that git merge is often used in conjunction with git checkout for selecting the current branch and git branch -d for deleting the obsolete target branch.

Merge conflicts

Merging and conflicts are a common part of the Git experience. Conflicts in other version control tools like SVN can be costly and time-consuming. Git makes merging super easy. Most of the time, Git will figure out how to automatically integrate new changes.

Create a merge conflicts

```
$ mkdir git-merge-test
$ cd git-merge-test
$ git init .
$ echo "this is some content to mess with" > merge.txt
$ git add merge.txt
$ git commit -am"we are commiting the inital content"
[master (root-commit) d48e74c] we are commiting the inital content
1 file changed, 1 insertion(+)
create mode 100644 merge.txt
```

```
$ git checkout -b new_branch_to_merge_later
$ echo "totally different content to merge later" > merge.txt
$ git commit -am"edited the content of merge.txt to cause a conflict"
[new_branch_to_merge_later 6282319] edited the content of merge.txt to cause a conflict
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
git checkout master
Switched to branch 'master'
echo "content to append" >> merge.txt
git commit -am"appended content to merge.txt"
[master 24fbe3c] appended content to merge.tx
1 file changed, 1 insertion(+)
```

```
git merge new_branch_to_merge_later
Auto-merging merge.txt
CONFLICT (content): Merge conflict in merge.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Git states : Modified, Staging, Committed

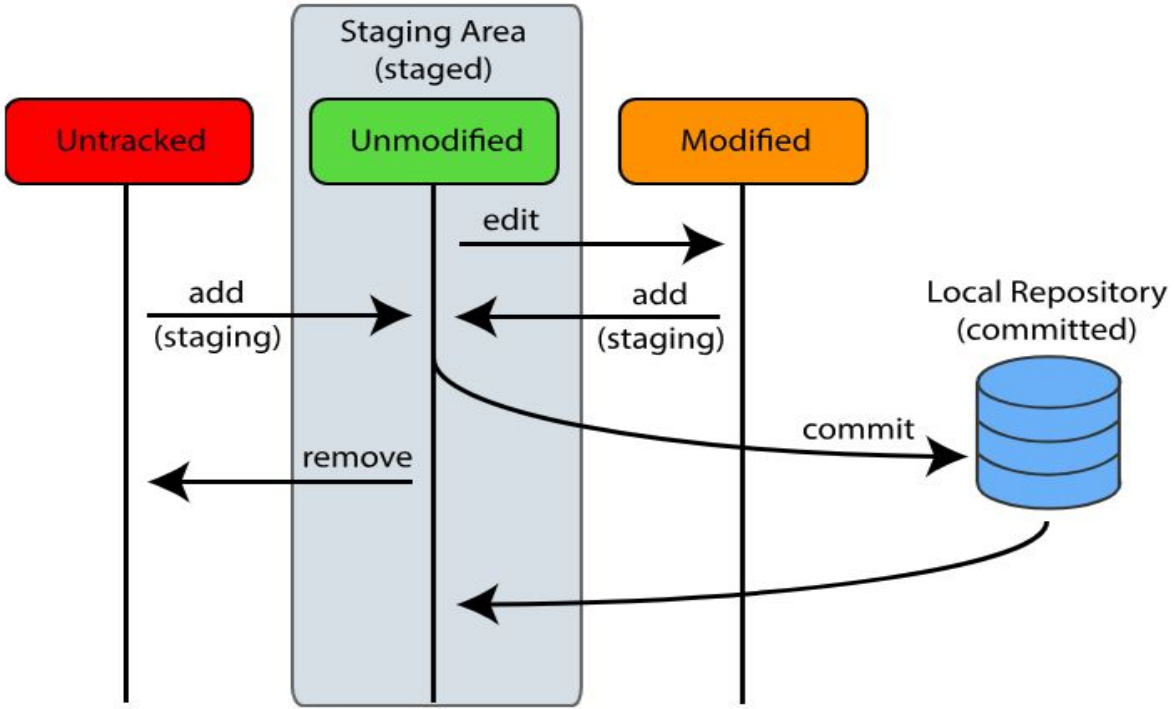
In order to fully understand Git, we have view our files how Git does. Envisioning what states your files are in will allow you to quickly pick up on the Git commands. Git views all files in three ways:

committed

modified/untracked

staged

Git states : Modified, Staging, Committed



ARARA maker space

Modified/Untracked and your Working Directory

Git views untracked and modified files similarly. Untracked means that the file is new to your Git project. Modified means that the file has been seen before, but has been changed, so is not ready to be snapshotted by Git. Modification of a file occurs in your working directory.

Staged and Staging Area

When a file becomes staged, it's taken into the staging area. This is where Git is able to take a snapshot of it and store its current state to your local repository. This area is also known as the Index.

Committed and the Git Directory

Committed means that Git has officially taken a snapshot of the files in the staging area, and stored a unique index in the Git directory. The terms snapshotted and committed are very similar. The significance of being committed is that you can now revert back to this project's current state at any time in the future.

The term for the very last snapshot you've made for commitment is known as the HEAD.

It's very important to understand the three states of a file, and the three areas they live in! If you have a good handle on these concepts, the rest of Git fundamentals should be a cinch!

Git workflow

- Centralized workflow
- Feature Branch Workflow
- Gitflow Workflow
- Forking Workflow

Feature workflow

The core idea behind the Feature Branch Workflow is that all feature development should take place in a dedicated branch instead of the `master` branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the `master` branch will never contain broken code, which is a huge advantage for continuous integration environments

Let’s go….

- Let’s go to create some repo and add some files….

Thanks ...