Version Control

Git vs CVS

<http://stackoverflow.com/questions/802573/difference-between-git-and-cvs>

Git vs GitHub

Git is a revision control system, a tool to manage your source code history. GitHub is a hosting service for Git repositories. So they are not the same thing: Git the tool, GitHub the service for projects that use Git. Most notably, GitHub is a consequence of the existance of git and not the only hosting service. Also, you don't *have* to use a remote service like github if all you want is version control - local git is just fine for that. Remote repositories are for backup and collaboration. Note that you can create a remote git repository in many web servers like webfaction vs heroku. It's a bit harder to configure that github, but cheaper for big teams.

Github provides access control and several collaboration features such as wikis, task management, and bug tracking and feature requests for every project.

You do not need GitHub to use Git.

Git = **Local** (on you computer), GitHub = **Remote** (web).

Github allows you to:

* Share your repositories with others.
* Access other user's repositories.
* Store remote copies of your repositories (github servers) as backup of your local copies.

GitHub, On Mac and Windows

On Linux, someone suggests git-cola, gitk. <http://stackoverflow.com/questions/1516720/git-gui-client-for-linux>

* Create and use a repository

Problem: how to create a repository

Resolve: find the folder in mac, add folders and files. And then you can see it in GitHub desktop app. Commit(Add to the local repository) and click publish(to cloud). Add more folders to the local folder, then check on GitHub desktop app. Commit and Sync(ONLY the first time needs publish).

* Start and manage a new branch

A new branch is a flow separated from your default branch (Master). You can try any new features on the new branch. The master branch must keep deployable.

Problem: fatal error: complains about default identification.

Resolve: this is because GitHub tries to find a global(on PC) user information, like email and name. On my mac, it finds jscobbie@.... It should be my email.

Do this in terminal: git config --global user.email [guyingjie@gmail.com](mailto:guyingjie@gmail.com)

git config --global user.name “caroline”

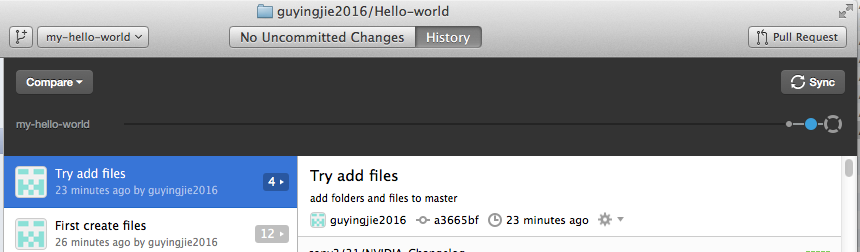
* Make changes to a file and push them to GitHub as commits
* Open and merge a pull request

To commit your changes to master branch, you need to do Pull request. Pull request publicize your changes and give you or anyone you’re working with a changce to review them.

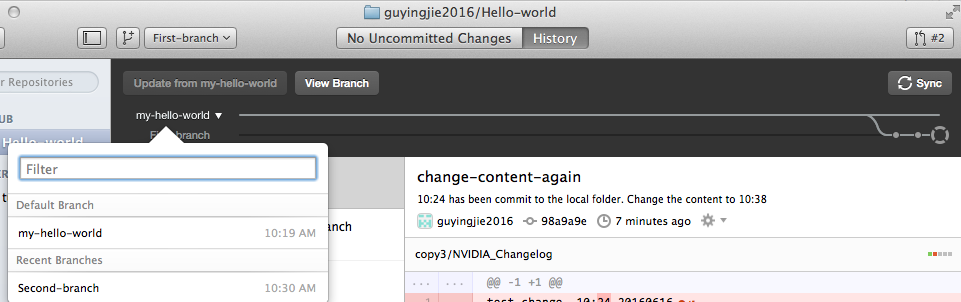
Problem: understand the buttons on GitHub Desktop app.

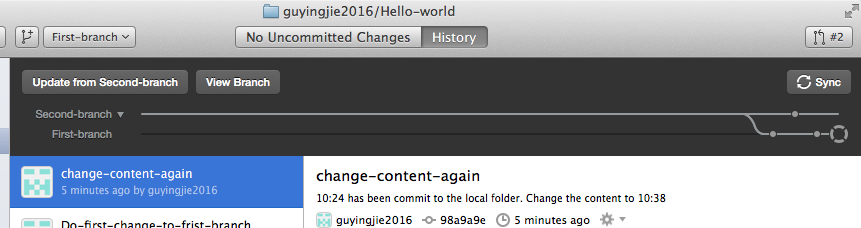
Resolve: After make branches and changes to branches. You need to know how to show and update changes(usually from branch/master to branch) and commit (from branch to master.)

Click “view branch”, then choose branch name on top of the app. The black block will show the graph of specific branch.

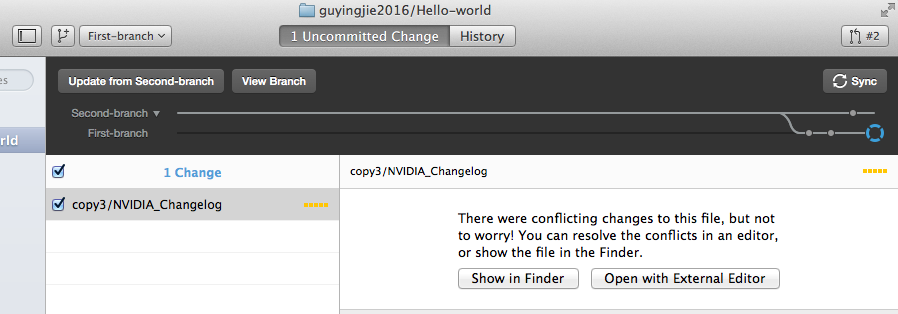


Click “Compare” to choose a branch (including the master branch) to compare with the pre-selected branch. Following pictures show select First-branch and compare from the pull down menus. If the compare branch has something changed after the split of two branches, the “Update from xxx” will be selectable.

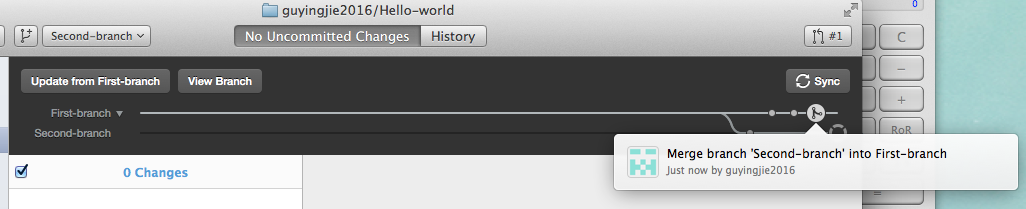




If there are conflicts between the to be merged branches, you will see this.



You can check by editor to see the conflicts. And if its intended to merge the conflicts, just commit the merge. Then compare the two branches, you will see.



Merge between branches doesn’t need to pull. But merge from branches to master (update from branches)or commit(locally change files) to master need pull request.

Working copy(in your work folder) > staging area(when you did &git add)>repository(after commit)

Command: <http://www.oxox.work/web/git/git/>

$git config –list

$git config user.name

$git help /git help commit (opens a browser to show the details)

Cd into your repository folder.

$git init

.git will be created. Ls –la.

Create files in the folder.

$git add . //add all files in . to repository. $git add filename //add a file

$git commit –m “This is xxx” //commit(saving) with a message.

[master (root-commit) 521c114] This is our first commit.

1 file changed, 1 insertion(+)

create mode 100644 first.txt

How to view your git commit history.

$git log (--author=”caroline”)

commit 521c114e7a4fab97ab0db406f4874fea274fac01

Author: caroline <guyingjie@gmail.com>

Date: Fri Jun 17 12:36:24 2016 -0400

This is our first commit.

$ git status

On branch master

nothing to commit, working directory clean

$ git status //if there is non committed changeds.

On branch master

Untracked files:

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

second.txt

third.txt

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

add some files

$git status

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: second.txt

Untracked files:

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

third.txt

$git checkout – filename…

If you change a file, add, commit it. It actually a deletion and a add.

[master 3629d33] Changed first.txt

1 file changed, 1 insertion(+), 1 deletion(-)

$git diff //shows the difference between working directory and repository.

Git diff only shows the diff between working directory and repository. If you add but not commit, git diff won’t show the diff any more. What you should do is to

$git diff --staged

How to delete files from repository? It also delete a file in the working repository.

$git rm filename

$git commit –m ”We delete third file.”

How to rename files.//If you rename a file, it is first a deletion and then a add. But if you add the renamed file and rm the previous filename, git will regard it as a rename. A easier way:

jmac:trygit jscobbie$ git mv first.txt home.txt

jmac:trygit jscobbie$ git diff

jmac:trygit jscobbie$ git diff --staged

**diff --git a/first.txt b/first.txt**

**deleted file mode 100644**

**index 4d4c072..0000000**

**--- a/first.txt**

**+++ /dev/null**

@@ -1 +0,0 @@

-dddddddThis is first file in trygit.

**diff --git a/home.txt b/home.txt**

**new file mode 100644**

**index 0000000..4d4c072**

**--- /dev/null**

**+++ b/home.txt**

@@ -0,0 +1 @@

+dddddddThis is first file in trygit.

$git commit –m “”

How to move a file to another folder.

jmac:trygit jscobbie$ git mv second.txt honeyhouse/putty.txt

jmac:trygit jscobbie$ git diff

jmac:trygit jscobbie$ git diff --staged

**diff --git a/honeyhouse/putty.txt b/honeyhouse/putty.txt**

**new file mode 100644**

**index 0000000..76942e4**

**--- /dev/null**

**+++ b/honeyhouse/putty.txt**

@@ -0,0 +1 @@

+aaaaaaaaaffffffaaaaThis is the second file.

**diff --git a/second.txt b/second.txt**

**deleted file mode 100644**

**index 76942e4..0000000**

**--- a/second.txt**

**+++ /dev/null**

@@ -1 +0,0 @@

-aaaaaaaaaffffffaaaaThis is the second file.

$git commit –m “”

$git commit –am “” // add and commit meanwhile from working directory to repository. But be careful, this is only for very simple change, it will commit all changes in the folder to repository. And don’t do this for rename files.

$git checkout -- filename //recover files from repository to working directory, no need to do commit.

How to unstage files. //the file is moved out of stage area, but keep changed in the working area.

$ git reset HEAD index.html

Unstaged changes after reset:

M index.html

What if we want to goback to some point? Do not delete the current point, just copy the previous point to the next point.

First >second >third>second.

$git log //to find out the commit id.

$git checkout first-digits-of-commit-id -- filename

$git commit –am “”

Using github //push is sending files to github, fetch is taking back files.

Setup remote server//github

$git remote add githubRepo <https://github.com/guyingjie2016/tutorials.git>

$git push –u githubRepo master

Github desktop commit is a real commit on local repository, need to sync(github) or &git push –u

Create branch, changing branch and then commit to master.

**Step 1:** From your project repository, bring in the changes and test.

git fetch origin

git checkout -b frogs origin/frogs

git merge master

**Step 2:** Merge the changes and update on GitHub.

git checkout master

git merge --no-ff frogs

git push origin master

$git branch -D frogs //remove a branch, -D is force, -d is not force.

$git branch branchname //create a branch.

$git branch //shows branches.

$git checkout branchname //switch branch.

$git push origin branchname //commit new branch to repository. Not commit to master.

Tips:

If you want to create a new repository.

On pc: mkdir, git init, add and commit files. Git remote add ahubname aURL.git,,git push –u ahubname master.

If there is a repository on github,

$mkdir, git init, git remote add ahubname aURL.git, git fetch ahubname repositoryname. Git checkout master.

If there is a repository on pc and you want to publish to github.

Create a repository on github and copy the URL.

$git remote add ahubname aURL.git

$git push –u ahubname master

If you make changes on local. First do add,commit. Then $git push ahubname master. Ahubname is the remote you add for this repository.

Add a branch $git branch abranchname $git checkout abranchname

Whenever you make changes, you make to the checkout branch. So remember to checkout the branch you want to work.