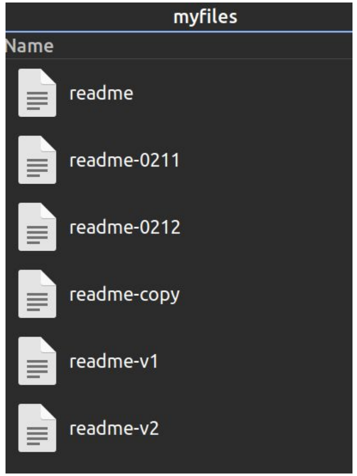
CS 35L
Discussion 1A
Week 5

Git

Contents

1. Version Control

- Why version control? Suppose...
    - I'm editing a file -- readme.
    - I want to delete some paragraphs, but I'm afraid that someday I may need these paragraphs! 😥
    - So I created a copy of the current file.
    - One week later... I'm having something like this:
    - Now I want to find the paragraph I deleted last Mon.
    - 😱 Which file should I look into!!!???
    - Another day, I want my friend to help me to edit the file.
    - I copy the file to my friend.
    - After the editing, I need to merge the file by myself! 😫



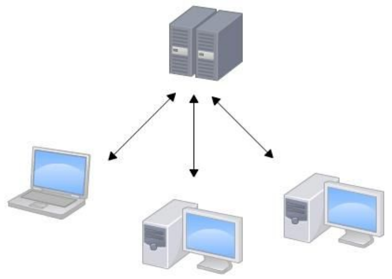
myfiles

Name

readme
readme-0211
readme-0212
readme-copy
readme-v1
readme-v2

- Why not using an automatic tool to do version control!?

1. Git History



- Linux: *Linus Torvalds* released the first version in 1991
  - Open-source => Large community => Volunteers coding for Linux
  - How to control these source codes?
  - Before 2002: People sent diff file to *Linus*, and he merged by himself
  - At that time, people had some version control tools like CVS, SVN
  - *Linus* said NO! These **centralized version control** tools need Internet to work
  - After 2002: the source code repo is too large to be manually handled
  - *BitKeeper* (*BitMover*): authorized free version control tool to Linux community
  - 2005: Someone tried to reverse engineering the *BitKeeper* protocols, so the copyright holder of the *BitKeeper* withdraw the free use of the product
  - *Linus*: spent two weeks and wrote a **distributed version control** system with C -- Git
    - **Distributed**: everyone has a complete repo copy on their own devices; we can push a bunch of modifications to others when needed; no internet requirement; other pros...

Compare Centralized and Distributed: https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs

Contents

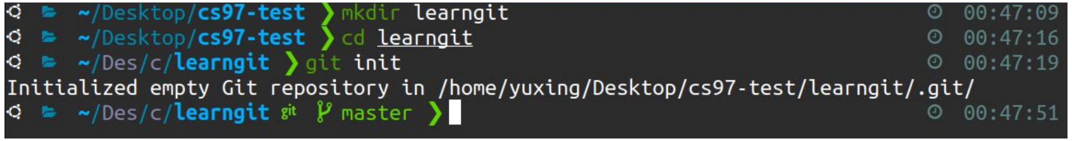
2. Prepare -- Install

- Linux
  - `git`
  - `sudo apt-get install git`
- Mac OS:
  - Install *homebrew*, use *homebrew* to install git (https://brew.sh/)
  - More easy way:
    - AppStore - Xcode; run Xcode
    - Xcode->Preferences->Download tab->Command Line Tools (click “install”)
- Windows:
  - https://git-scm.com/downloads

Contents

3. Git Basis

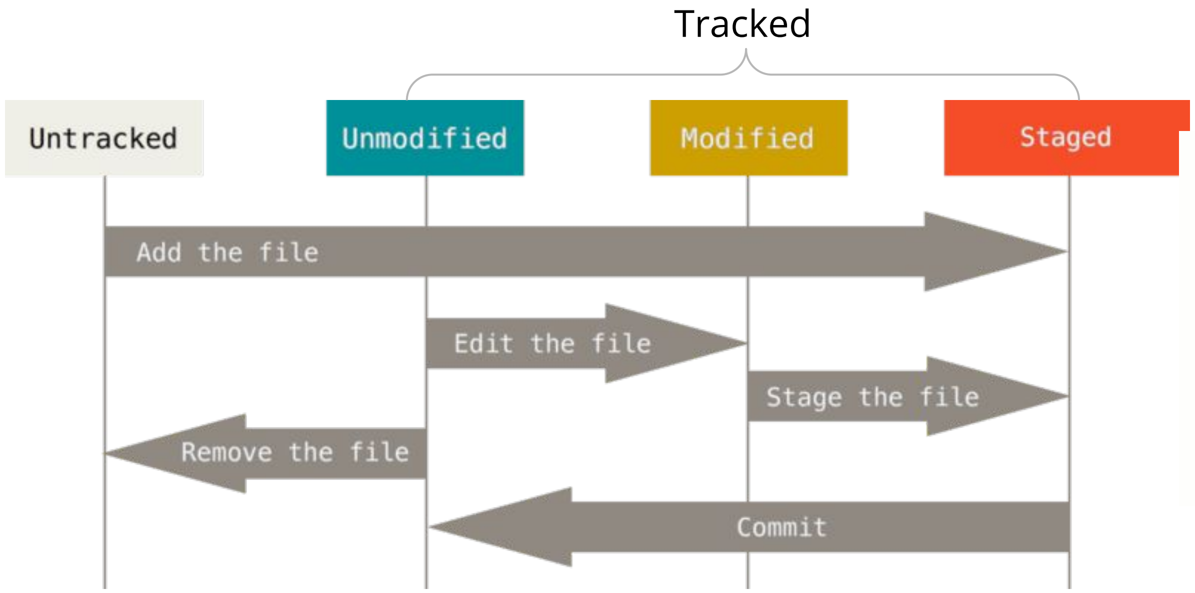
- Create a Git Repo
    - `git init`
        - Initialize an empty git repo in current directory
        - Everything inside this repo(directory) can be tracked if you add them to git
        - A new folder `.git` : contain version control information; do not modify



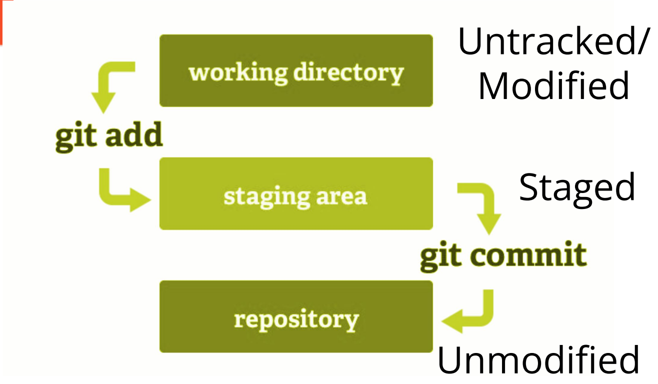
```
~/Desktop/cs97-test ❯ mkdir learngit                         00:47:09
~/Desktop/cs97-test ❯ cd learngit                            00:47:16
~/Des/c/learngit ❯ git init                                  00:47:19
Initialized empty Git repository in /home/yuxing/Desktop/cs97-test/learngit/.git/
~/Des/c/learngit git  master ❯                               00:47:51
```

3. Git Basis

- Manage files in the Git Repo

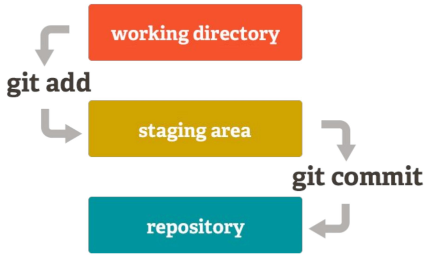


The lifecycle of the status of your files

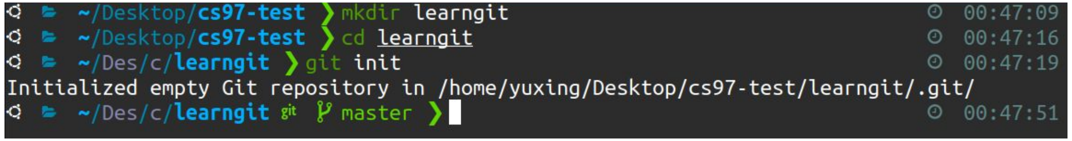


Git areas

3. Git Basis

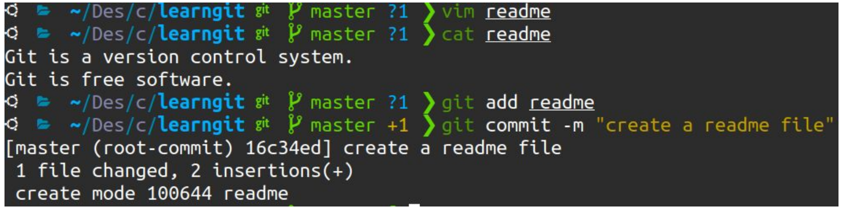


- Create a Git Repo
  - `git init`
    - Initialize an empty git repo in current directory
    - Everything inside this repo(directory) can be tracked if you add them to git
    - A new folder `.git` : contain version control information; do not modify



```
~/Desktop/cs97-test ❯ mkdir learngit                                   00:47:09
~/Desktop/cs97-test ❯ cd learngit                                      00:47:16
~/Des/c/learngit ❯ git init                                            00:47:19
Initialized empty Git repository in /home/yuxing/Desktop/cs97-test/learngit/.git/
~/Des/c/learngit git  master ❯                                         00:47:51
```

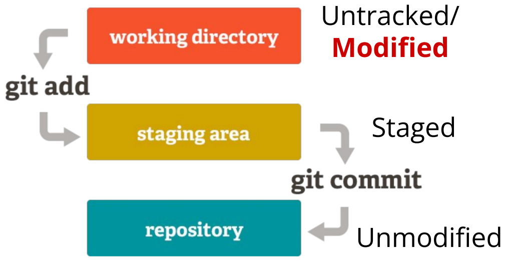
  - `git add <file>`
    - If `<file>` is untracked by Git: track this file
    - // (not this case) If `<file>` is already tracked: Stage changes in `<file>`
  - `git commit -m "some message"`
    - Commit the file into Git Repo
    - with some description

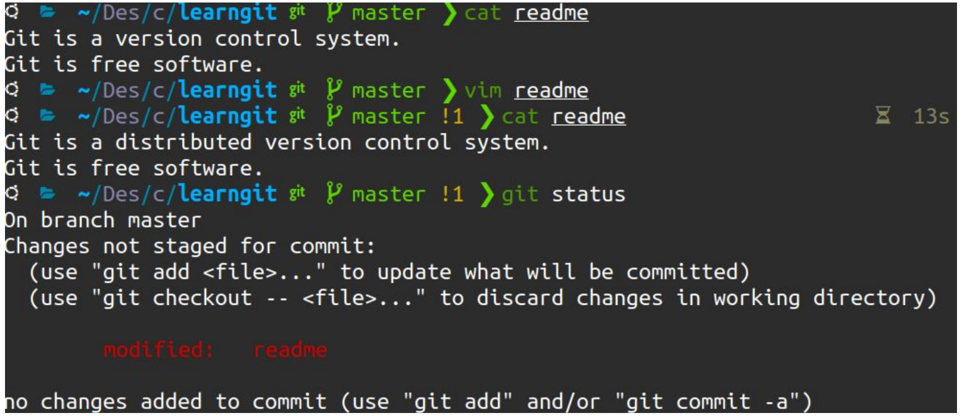


```
~/Des/c/learngit git  master ?1 ❯ vim readme
~/Des/c/learngit git  master ?1 ❯ cat readme
Git is a version control system.
Git is free software.
~/Des/c/learngit git  master ?1 ❯ git add readme
~/Des/c/learngit git  master +1 ❯ git commit -m "create a readme file"
[master (root-commit) 16c34ed] create a readme file
 1 file changed, 2 insertions(+)
 create mode 100644 readme
```

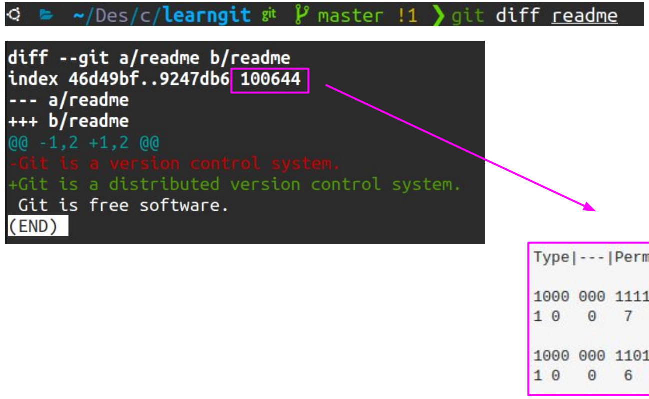
3. Git Basis

- Record Changes - 1
  - Modify our "**readme**" file
  - **git status**
    - Show the current tracking status
      - Modified files
      - Untracked files
  - **git diff <file>**
    - Show the the changes between commits





```
~/Des/c/learngit  master ❯ cat readme
Git is a version control system.
Git is free software.
~/Des/c/learngit  master ❯ vim readme
~/Des/c/learngit  master !1 ❯ cat readme                                    13s
Git is a distributed version control system.
Git is free software.
~/Des/c/learngit  master !1 ❯ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme

no changes added to commit (use "git add" and/or "git commit -a")
```



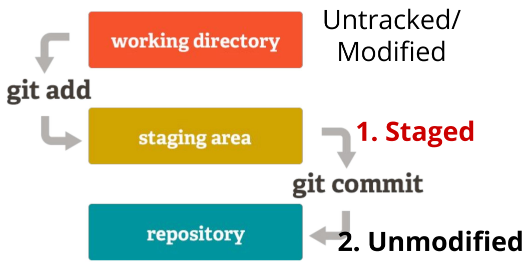
```
~/Des/c/learngit  master !1 ❯ git diff readme
```

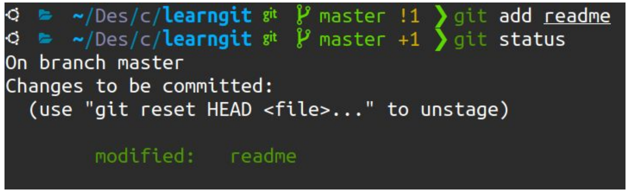
```
diff --git a/readme b/readme
index 46d49bf..9247db6 100644
--- a/readme
+++ b/readme
@@ -1,2 +1,2 @@
-Git is a version control system.
+Git is a distributed version control system.
 Git is free software.
(END)
```

```
Type|---|Perm bits

1000 000 111101101
1 0    0    7  5  5

1000 000 110100100
1 0    0    6  4  4
```

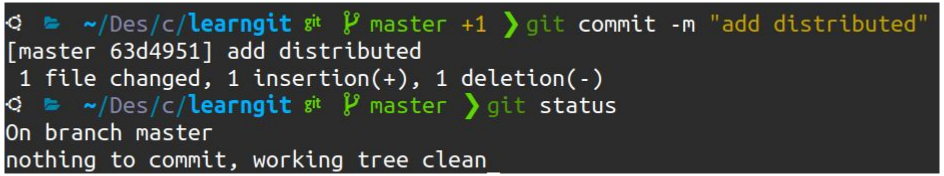
3. Git Basis

- Record Changes - 2
  - Move our **readme** to staging area :
  - **git add <file>**
    - // (not this case) If **<file>** is untracked by Git: track this file
    - If **<file>** is already tracked: Stage changes in **<file>**
  - Commit our **readme** to Git Repo
  - **git commit -m "some message"**
    - Commit the file into Git Repo
    - with some description





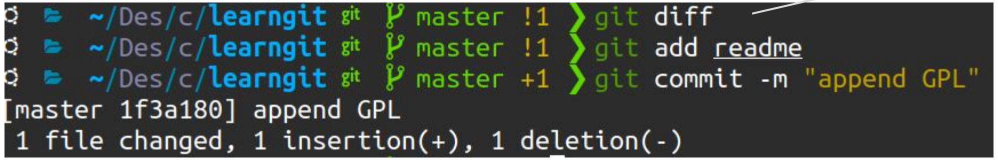
```
~/Des/c/learngit git  master !1 ❯ git add readme
~/Des/c/learngit git  master +1 ❯ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme
```



```
~/Des/c/learngit git  master +1 ❯ git commit -m "add distributed"
[master 63d4951] add distributed
 1 file changed, 1 insertion(+), 1 deletion(-)
~/Des/c/learngit git  master ❯ git status
On branch master
nothing to commit, working tree clean
```

3. Git Basis

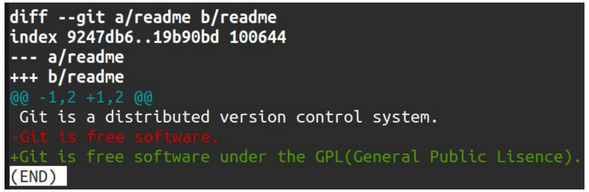
- View History
  - One more time: change **readme** and commit to our Git repo
  - Move our **readme** to staging area
  - Commit our **readme** to Git Repo



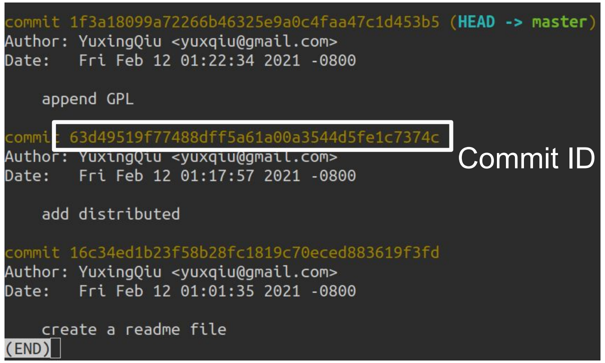
```
~/Des/c/learngit git  master !1 ❯ git diff
~/Des/c/learngit git  master !1 ❯ git add readme
~/Des/c/learngit git  master +1 ❯ git commit -m "append GPL"
[master 1f3a180] append GPL
 1 file changed, 1 insertion(+), 1 deletion(-)
```

  - **git log**
    - Show the commit logs
  - **git log --pretty=oneline**
    - Show with less info
  - **git log --graph --pretty=format:"%h %s"**

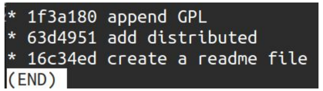
- Commit ID
  - HEAD -- current version
  - HEAD^ -- the parent of HEAD
  - HEAD^^ -- the parent of HEAD^



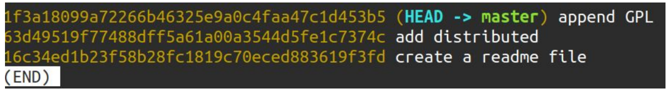
```
diff --git a/readme b/readme
index 9247db6..19b90bd 100644
--- a/readme
+++ b/readme
@@ -1,2 +1,2 @@
 Git is a distributed version control system.
-Git is free software.
+Git is free software under the GPL(General Public Lisence).
(END)
```



```
commit 1f3a18099a72266b46325e9a0c4faa47c1d453b5 (HEAD -> master)
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 01:22:34 2021 -0800

    append GPL

commit 63d49519f77488dff5a61a00a3544d5fe1c7374c
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 01:17:57 2021 -0800

    add distributed

commit 16c34ed1b23f58b28fc1819c70eced883619f3fd
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 01:01:35 2021 -0800

    create a readme file
(END)
```

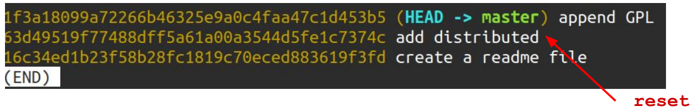


```
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```



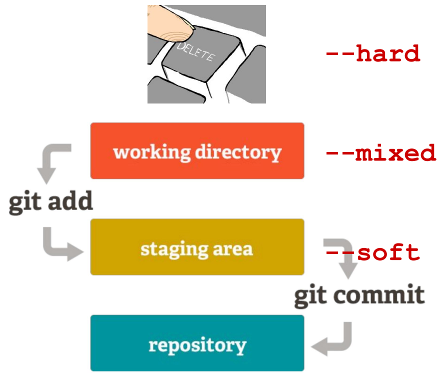
```
1f3a18099a72266b46325e9a0c4faa47c1d453b5 (HEAD -> master) append GPL
63d49519f77488dff5a61a00a3544d5fe1c7374c add distributed
16c34ed1b23f58b28fc1819c70eced883619f3fd create a readme file
(END)
```

3. Git Basis



```
1f3a18099a72266b46325e9a0c4faa47c1d453b5 (HEAD -> master) append GPL
63d49519f77488dff5a61a00a3544d5fe1c7374c add distributed
16c34ed1b23f58b28fc1819c70eced883619f3fd create a readme file
(END)
```

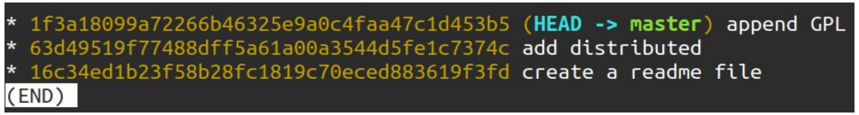
New changes after our "reset" place



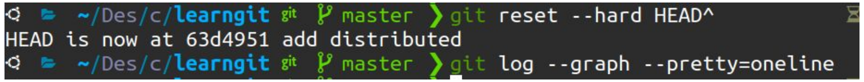
- Rollback
  - We want to rollback to the version `"add distributed"`
  - `git reset`
    - Reset current `HEAD` to the specified state
    - `git reset --mixed <commit>` (default option, ⇔ `git reset <commit>`)
      - `master` and `head` point to `<commit>`, the index is also modified to match the modified files (modifications are push back to the working directory, we need to do `git add`, then `git commit` to save changes after `<commit>`)
    - `git reset --soft <commit>`
      - Same as `--mixed`, but the index still has later changes saving in the staging area (we need to do `git commit` to save changes after `<commit>`)
    - `git reset --hard <commit>`
      - Same as `--mixed`, but remove all changes after `<commit>`

3. Git Basis

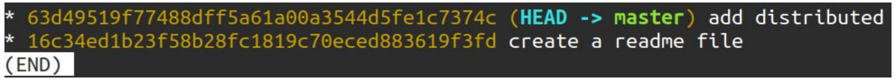
- Rollback
  - We want to rollback to previous versions
  - `git reset --hard HEAD^`
    - Reset current **HEAD** to its parent



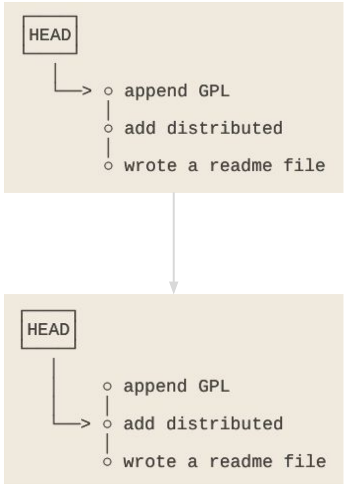
```
* 1f3a18099a72266b46325e9a0c4faa47c1d453b5 (HEAD -> master) append GPL
* 63d49519f77488dff5a61a00a3544d5fe1c7374c add distributed
* 16c34ed1b23f58b28fc1819c70eced883619f3fd create a readme file
(END)
```



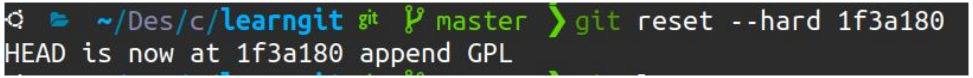
```
~/Des/c/learngit git  master  git reset --hard HEAD^
HEAD is now at 63d4951 add distributed
~/Des/c/learngit git  master  git log --graph --pretty=oneline
```



```
* 63d49519f77488dff5a61a00a3544d5fe1c7374c (HEAD -> master) add distributed
* 16c34ed1b23f58b28fc1819c70eced883619f3fd create a readme file
(END)
```



  - `git reset --hard <commit_id>`



```
~/Des/c/learngit git  master  git reset --hard 1f3a180
HEAD is now at 1f3a180 append GPL
```

  - `git reflog` ## checkout your previous command and commit ids

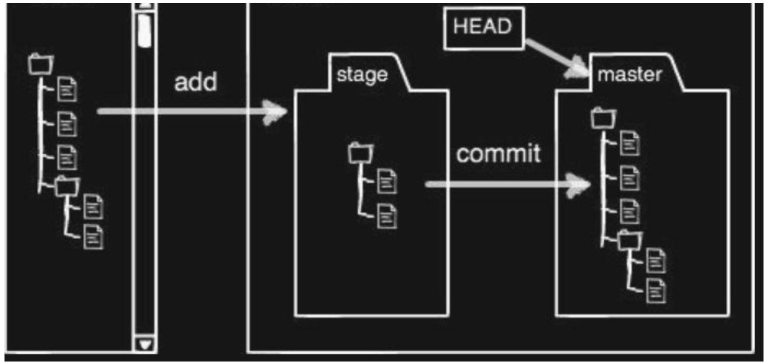
3. Git Basis

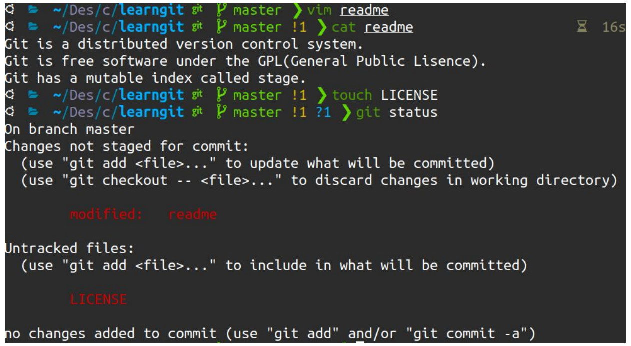
- Working Directory, Staging area and Repo
  - Local development environment
    - Working directory
    - Staging area
    - Local Repository

Working Directory

Staging Area

Git Repository

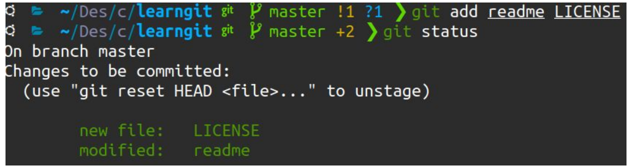




```
~/Des/c/learngit git  master ❯ vim readme
~/Des/c/learngit git  master !1 ❯ cat readme                                  16s
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
~/Des/c/learngit git  master !1 ❯ touch LICENSE
~/Des/c/learngit git  master !1 ?1 ❯ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme

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

        LICENSE

no changes added to commit (use "git add" and/or "git commit -a")
```

Edit files in our working directory

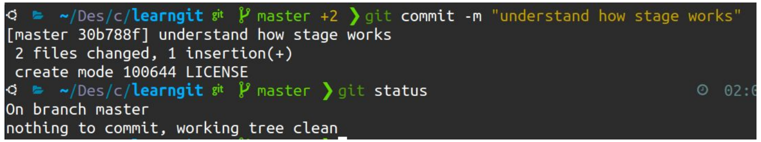
↓



```
~/Des/c/learngit git  master !1 ?1 ❯ git add readme LICENSE
~/Des/c/learngit git  master +2 ❯ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   LICENSE
        modified:   readme
```

`git add`: move all the changes to the stage

↓



```
~/Des/c/learngit git  master +2 ❯ git commit -m "understand how stage works"
[master 30b788f] understand how stage works
 2 files changed, 1 insertion(+)
 create mode 100644 LICENSE
~/Des/c/learngit git  master ❯ git status                                    02:0
On branch master
nothing to commit, working tree clean
```

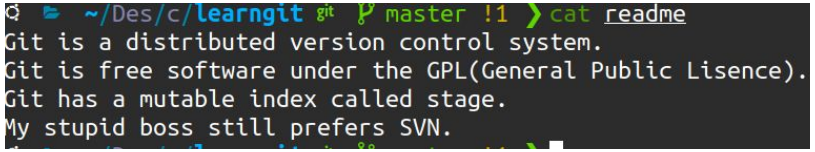
`git commit`: commit changes to the branch

3. Git Basis

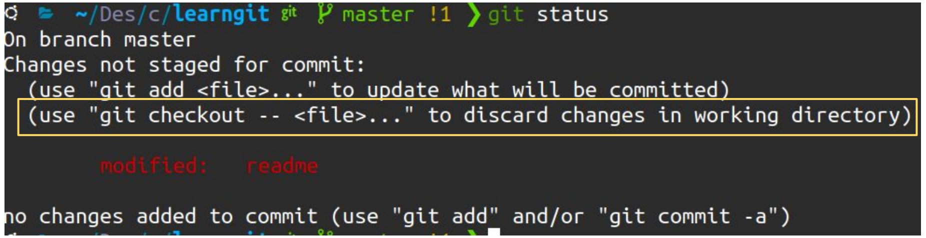
- Discard Changes
  - Modify our **readme**
  - We want discard changes in **readme**

1. **Readme** is in the current working directory
   - **git status**
   - **git checkout -- readme**
     - Discard the changes in the current working directory

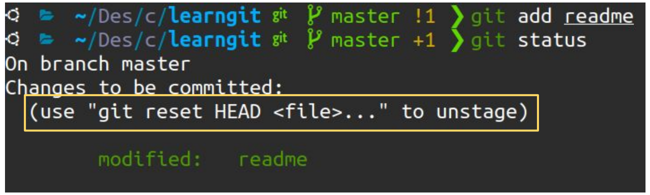
2. **Readme** is in the staging area
   - **git status**
   - **git reset HEAD readme**
     - Unstage the current file, push it back to the working directory



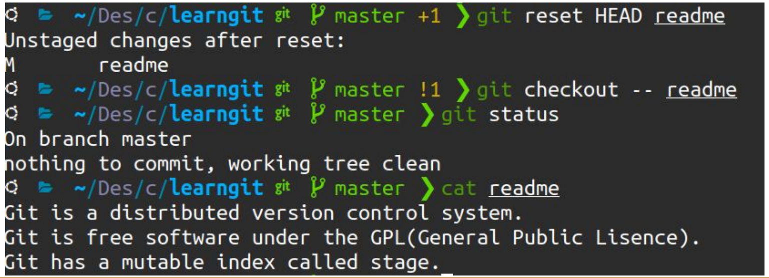
```
~/Des/c/learngit git  master !1 ❯ cat readme
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
My stupid boss still prefers SVN.
```



```
~/Des/c/learngit git  master !1 ❯ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   readme

no changes added to commit (use "git add" and/or "git commit -a")
```



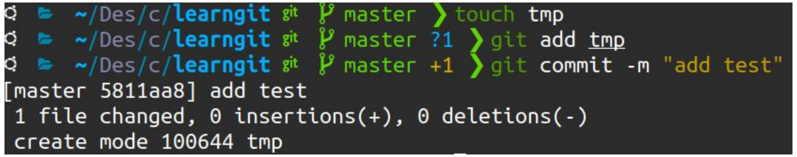
```
~/Des/c/learngit git  master !1 ❯ git add readme
~/Des/c/learngit git  master +1 ❯ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme
```



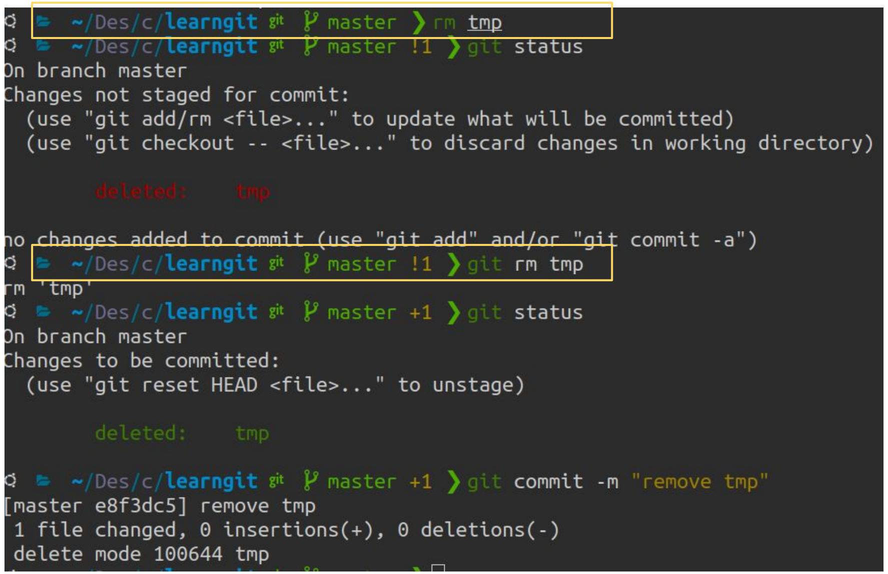
```
~/Des/c/learngit git  master +1 ❯ git reset HEAD readme
Unstaged changes after reset:
M       readme
~/Des/c/learngit git  master !1 ❯ git checkout -- readme
~/Des/c/learngit git  master ❯ git status
On branch master
nothing to commit, working tree clean
~/Des/c/learngit git  master ❯ cat readme
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
```

3. Git Basis

- Delete File
  - Create a `tmp`
  - Add `tmp` to the git repo
  - We want remove this `tmp` file
  - `rm tmp`
  - `git rm tmp`
  - `git add tmp` (also works)



```
~/Des/c/learngit git  master ❯ touch tmp
~/Des/c/learngit git  master ?1 ❯ git add tmp
~/Des/c/learngit git  master +1 ❯ git commit -m "add test"
[master 5811aa8] add test
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 tmp
```

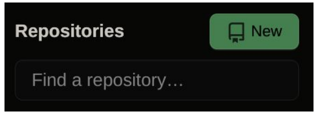


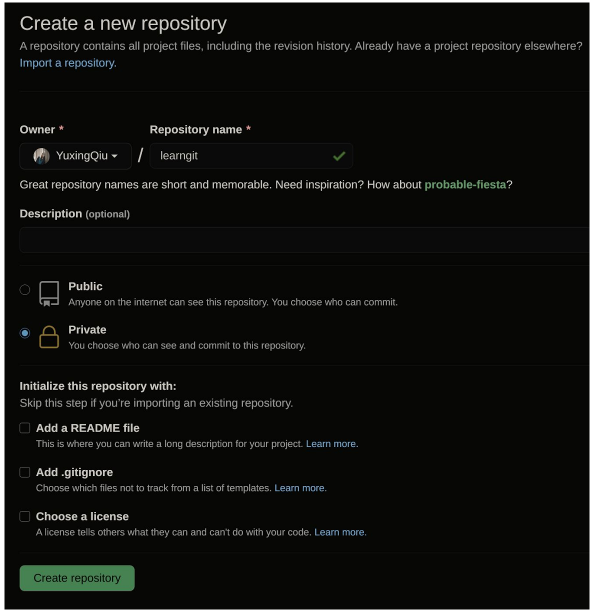
```
~/Des/c/learngit git  master ❯ rm tmp
~/Des/c/learngit git  master !1 ❯ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    tmp

no changes added to commit (use "git add" and/or "git commit -a")
~/Des/c/learngit git  master !1 ❯ git rm tmp
rm 'tmp'
~/Des/c/learngit git  master +1 ❯ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    tmp

~/Des/c/learngit git  master +1 ❯ git commit -m "remove tmp"
[master e8f3dc5] remove tmp
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 tmp
```

Contents

4. Remote Repo

- Add Remote Repo - 1
  - Create a Repo on Github (or GitLab, BitBucket ...)





Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * YuxingQiu / Repository name * learngit

Great repository names are short and memorable. Need inspiration? How about probable-fiesta?

Description (optional)

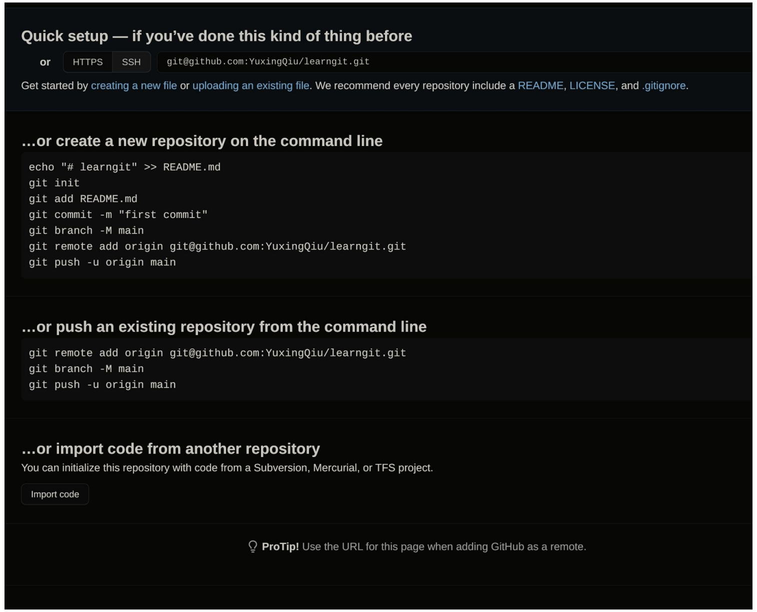
Public
Anyone on the internet can see this repository. You choose who commits.

[✓ Private]
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

- [ ] Add a README file
  This is where you can write a long description for your project. Learn more.
- [ ] Add .gitignore
  Choose which files not to track from a list of templates. Learn more.
- [ ] Choose a license
  A license tells others what they can and can't do with your code. Learn more.

Create repository



Quick setup — if you've done this kind of thing before

or HTTPS SSH `git@github.com:YuxingQiu/learngit.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# learngit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:YuxingQiu/learngit.git
git push -u origin main
```

…or push an existing repository from the command line

```
git remote add origin git@github.com:YuxingQiu/learngit.git
git branch -M main
git push -u origin main
```

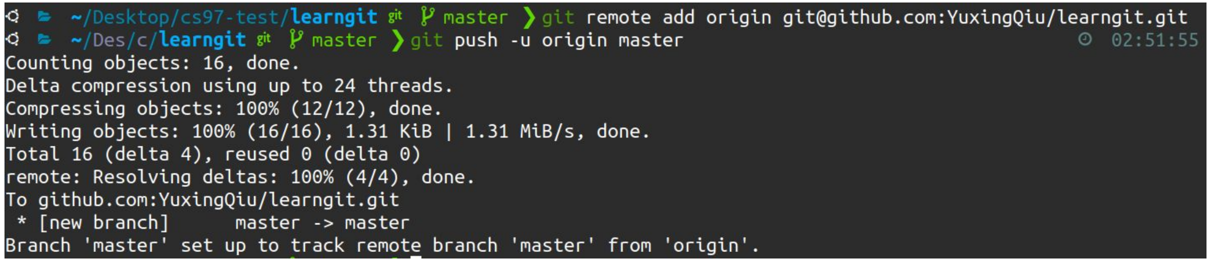
…or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

4. Remote Repo



```
~/Desktop/cs97-test/learngit git  master ❯ git remote add origin git@github.com:YuxingQiu/learngit.git
~/Des/c/learngit git  master ❯ git push -u origin master                                   02:51:55
Counting objects: 16, done.
Delta compression using up to 24 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (16/16), 1.31 KiB | 1.31 MiB/s, done.
Total 16 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
To github.com:YuxingQiu/learngit.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

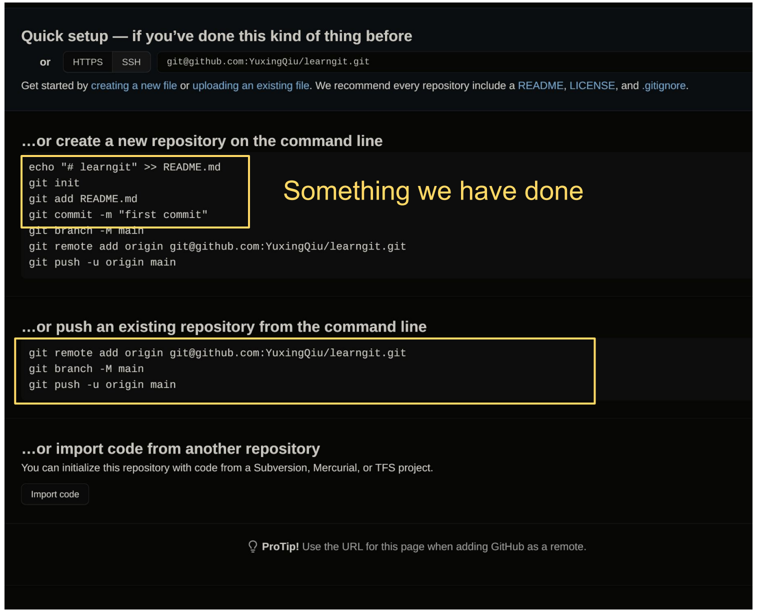
- Add Remote Repo - 2
  - Push our local repo to remote
    - Follow the instructions
  - Connecting through ssh protocol
    - https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh

- Connect your local repo to a remote repo

`git remote add origin git@github.com:YuxingQiu/learngit.git`

- Push your local repo to remote

`git push -u origin master`



Quick setup — if you've done this kind of thing before

or HTTPS SSH `git@github.com:YuxingQiu/learngit.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# learngit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:YuxingQiu/learngit.git
git push -u origin main
```

Something we have done

...or push an existing repository from the command line

```
git remote add origin git@github.com:YuxingQiu/learngit.git
git branch -M main
git push -u origin main
```

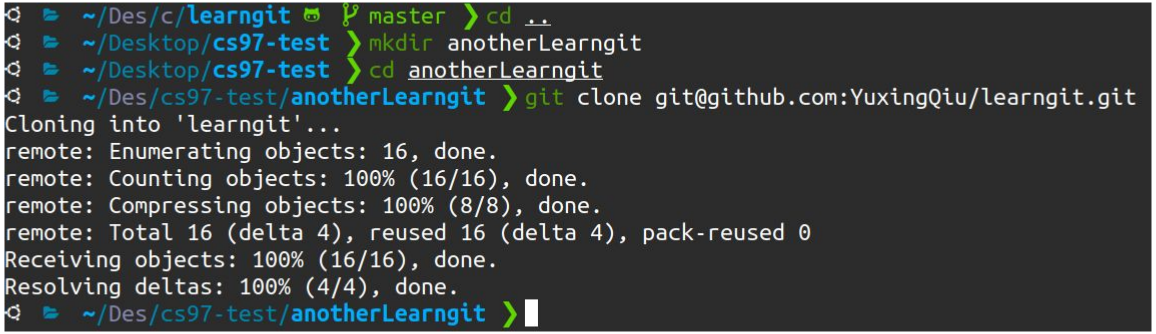
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

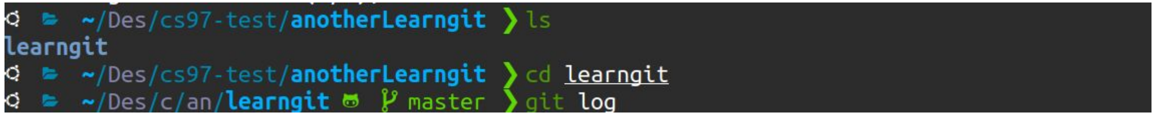
ProTip! Use the URL for this page when adding GitHub as a remote.

4. Remote Repo

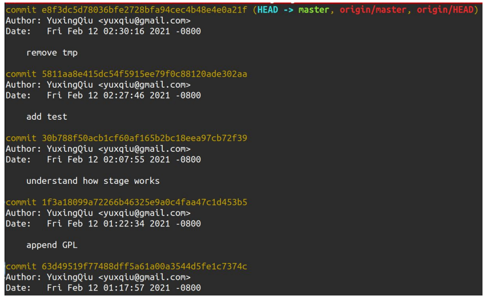
- Clone From the Remote Repo
  - Copy an existing repo
    - **`git clone <repo-url>`**



```
~/Des/c/learngit  master ❯ cd ..
~/Desktop/cs97-test ❯ mkdir anotherLearngit
~/Desktop/cs97-test ❯ cd anotherLearngit
~/Des/cs97-test/anotherLearngit ❯ git clone git@github.com:YuxingQiu/learngit.git
Cloning into 'learngit'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 16 (delta 4), reused 16 (delta 4), pack-reused 0
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (4/4), done.
~/Des/cs97-test/anotherLearngit ❯
```



```
~/Des/cs97-test/anotherLearngit ❯ ls
learngit
~/Des/cs97-test/anotherLearngit ❯ cd learngit
~/Des/c/an/learngit  master ❯ git log
```



```
commit e8f3dc5d78036bfe2728bfa94cec4b48e4e0a21f (HEAD -> master, origin/master, origin/HEAD)
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 02:30:16 2021 -0800

    remove tmp

commit 5811aa8e415dc54f5915ee79f0c88120ade302aa
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 02:27:46 2021 -0800

    add test

commit 30b788f50acb1cf60af165b2bc18eea97cb72f39
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 02:07:55 2021 -0800

    understand how stage works

commit 1f3a18099a72266b46325e9a0c4faa47c1d453b5
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 01:22:34 2021 -0800

    append GPL

commit 63d49519f77488dff5a61a00a3544d5fe1c7374c
Author: YuxingQiu <yuxqiu@gmail.com>
Date:   Fri Feb 12 01:17:57 2021 -0800
```

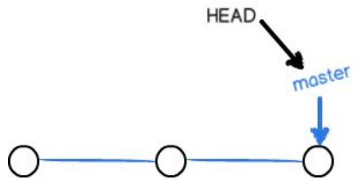
4. Remote Repo

- Pull/Fetch From the Remote Repo
    - Pull information from the remote repo to the local repo
        - `git fetch`
            - Retrieves metadata information only (what is changed)
        - `git pull`
            - Retrieves metadata information and copies any changed files
            - Runs two commands
                - `git fetch`
                - `git merge`

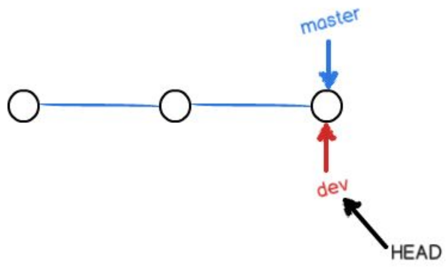
Contents

5. Branch

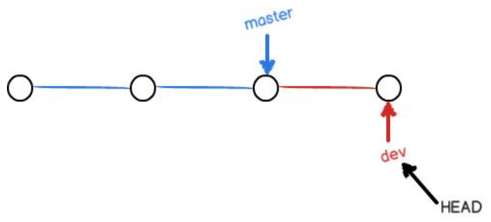
- Create and Merge - 1
  - Before: we have a **`master`** branch, and a pointer **`HEAD`** to indicate where we are
  - Every commit will increase the length of master by one step



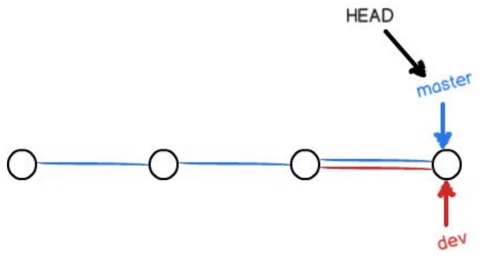
  - When we create a new branch **`dev`**, Git will
    - create a new pointer **`dev`**
    - let **`dev`** point to the same commit in **`master`**
    - Let **`HEAD`** point to **`dev`**

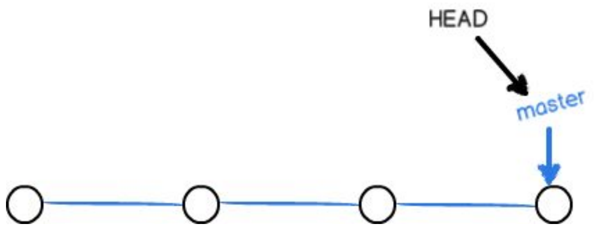


5. Branch



- Create and Merge - 2
    - Then we will work on this new branch `dev`
        - Every `git add` and `git commit` will operate on this branch
        - Pointer `dev` will move forward
        - Pointer `master` stay unchanged
    - When we finish working on `dev` and want to merge it back to `master`
        - Let `master` point to the current `dev` commit
        - Done

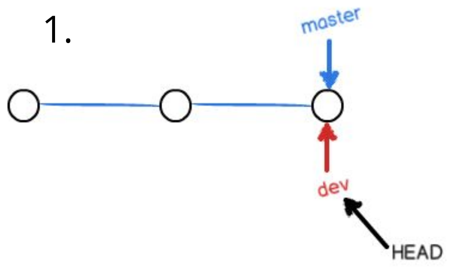




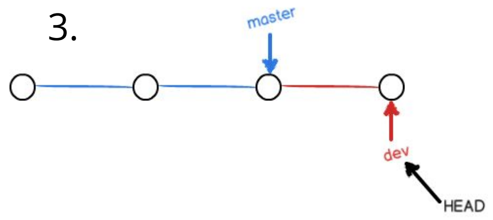
5. Branch

- Create and Merge - 3
  - create a new branch **dev**
    1. `git checkout -b dev` ## create a new branch **dev** and switch to **dev**
       - Equivalent to
         - `git branch dev` ## create a new branch **dev**
         - `git checkout dev` ## switch to branch **dev**
    2. `git branch` ## show current branches in local repo
    3. Make changes in **dev** (`git add, git commit`)
    4. `git checkout master` ## switch to branch **master**
    5. `git merge dev` ## merge changes in **dev** to **master**
    6. `git branch -d dev` ## remove branch **dev**

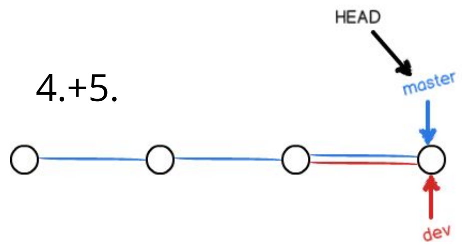
1.



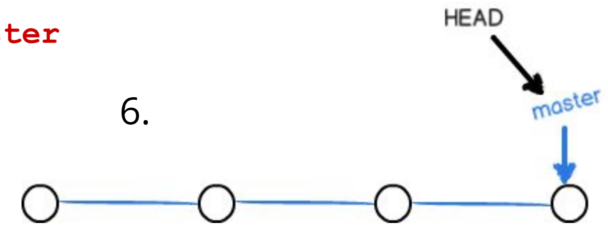
3.



4.+5.



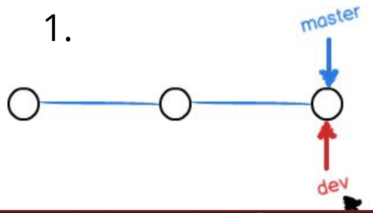
6.



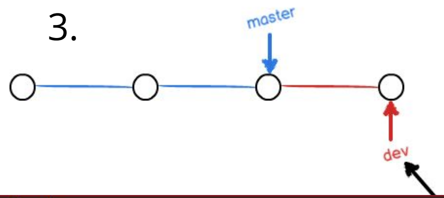
5. Branch

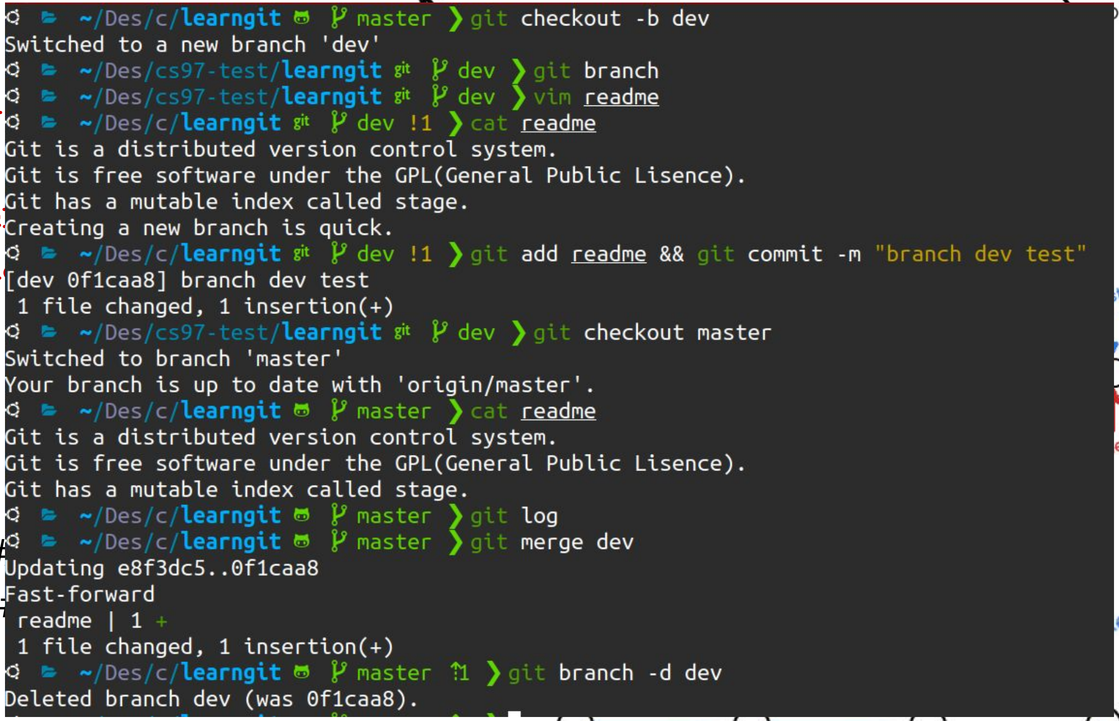
- Create and Merge - 3
  - create a new branch **dev**
    1. **git checkout -b dev**
       - Equivalent to
         - **git branc**
         - **git check**
    2. **git branch** ## show
    3. Make changes in **dev**
    4. **git checkout master**
    5. **git merge dev** #
    6. **git branch -d dev** #

1.

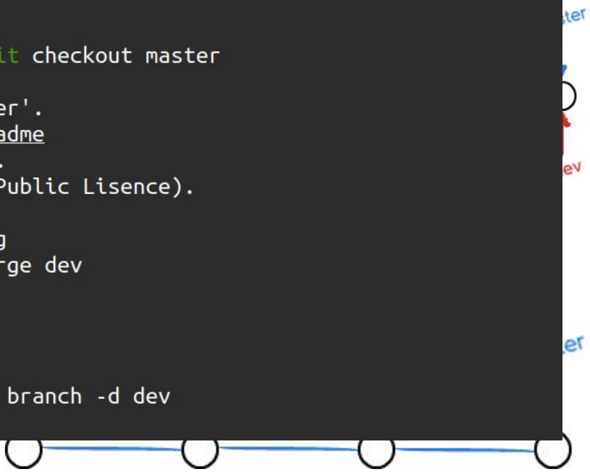


3.



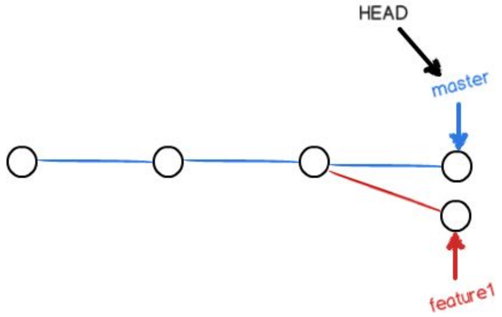


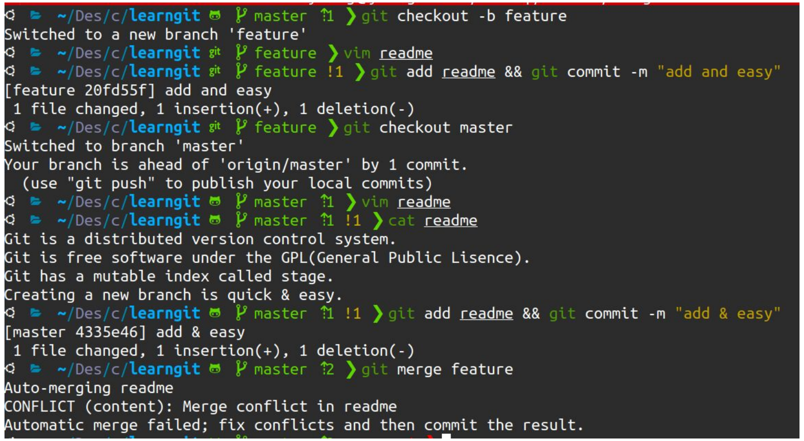
```
~/Des/c/learngit  master ❯ git checkout -b dev
Switched to a new branch 'dev'
~/Des/cs97-test/learngit git  dev ❯ git branch
~/Des/cs97-test/learngit git  dev ❯ vim readme
~/Des/c/learngit git  dev !1 ❯ cat readme
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
Creating a new branch is quick.
~/Des/c/learngit git  dev !1 ❯ git add readme && git commit -m "branch dev test"
[dev 0f1caa8] branch dev test
 1 file changed, 1 insertion(+)
~/Des/cs97-test/learngit git  dev ❯ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
~/Des/c/learngit  master ❯ cat readme
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
~/Des/c/learngit  master ❯ git log
~/Des/c/learngit  master ❯ git merge dev
Updating e8f3dc5..0f1caa8
Fast-forward
 readme | 1 +
 1 file changed, 1 insertion(+)
~/Des/c/learngit  master ⇡1 ❯ git branch -d dev
Deleted branch dev (was 0f1caa8).
```



5. Branch

- Solve Conflicts - 1
  - What if we have conflicts when merging two branches
    - Create a new branch **`feature`**
    - Make changes in **`feature`** (**`git add`**, **`git commit`**)
    - Switch to branch **`master`**
    - Make (conflict) changes in **`master`** (**`git add`**, **`git commit`**)
    - Merge => conflict!

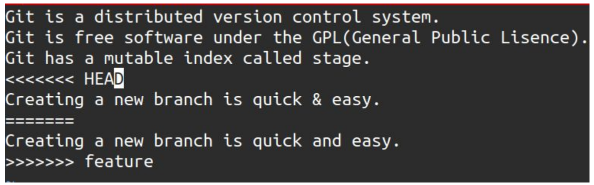




```
~/Des/c/learngit  master ⇡1 ❯ git checkout -b feature
Switched to a new branch 'feature'
~/Des/c/learngit  feature ❯ vim readme
~/Des/c/learngit  feature !1 ❯ git add readme && git commit -m "add and easy"
[feature 20fd55f] add and easy
 1 file changed, 1 insertion(+), 1 deletion(-)
~/Des/c/learngit  feature ❯ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
~/Des/c/learngit  master ⇡1 ❯ vim readme
~/Des/c/learngit  master ⇡1 !1 ❯ cat readme
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
Creating a new branch is quick & easy.
~/Des/c/learngit  master ⇡1 !1 ❯ git add readme && git commit -m "add & easy"
[master 4335e46] add & easy
 1 file changed, 1 insertion(+), 1 deletion(-)
~/Des/c/learngit  master ⇡2 ❯ git merge feature
Auto-merging readme
CONFLICT (content): Merge conflict in readme
Automatic merge failed; fix conflicts and then commit the result.
```

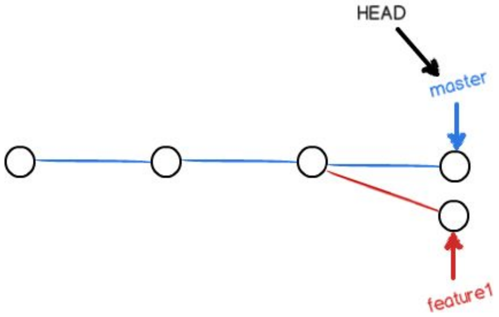
5. Branch

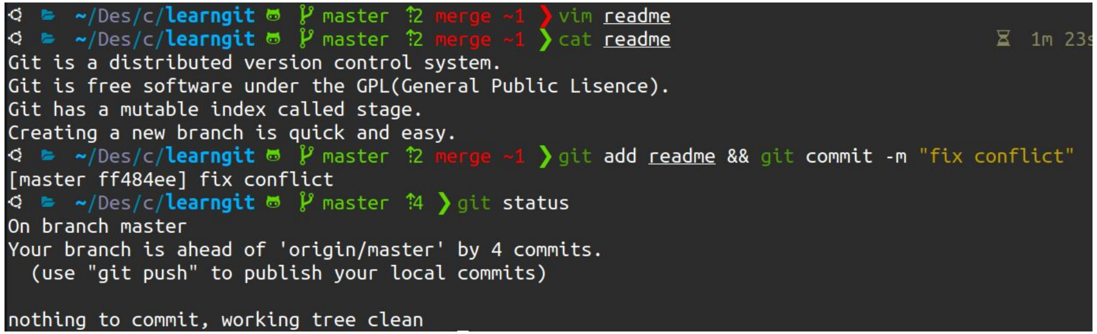
- Solve Conflicts - 2
  - What if we have conflicts when merging two branches
    - Checkout the content of **readme**



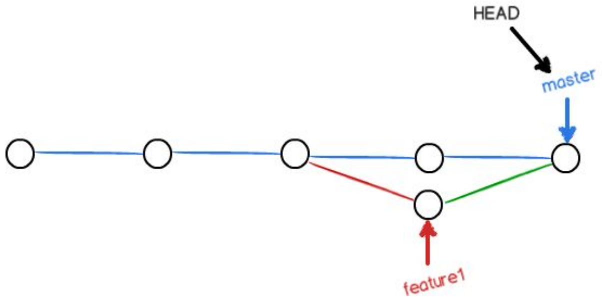
```
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
<<<<<<< HEAD
Creating a new branch is quick & easy.
=======
Creating a new branch is quick and easy.
>>>>>>> feature
```

    - Modify **readme**, add, commit



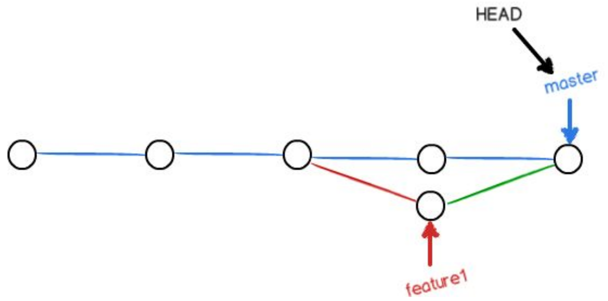


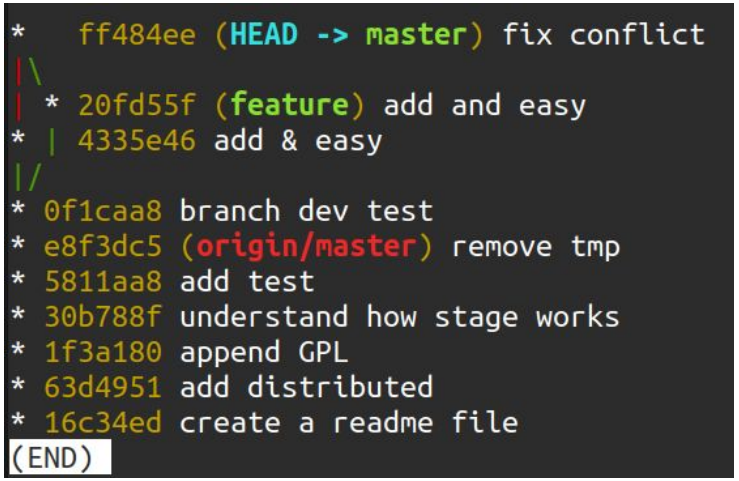
```
~/Des/c/learngit  master ↑2 merge ~1 ❯ vim readme
~/Des/c/learngit  master ↑2 merge ~1 ❯ cat readme                     ⌛ 1m 23s
Git is a distributed version control system.
Git is free software under the GPL(General Public Lisence).
Git has a mutable index called stage.
Creating a new branch is quick and easy.
~/Des/c/learngit  master ↑2 merge ~1 ❯ git add readme && git commit -m "fix conflict"
[master ff484ee] fix conflict
~/Des/c/learngit  master ↑4 ❯ git status
On branch master
Your branch is ahead of 'origin/master' by 4 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```



5. Branch

- Solve Conflicts - 3
  - What if we have conflicts when merging two branches
    - Use `git log --graph --pretty=oneline --abbrev-commit` to see the branch merging status

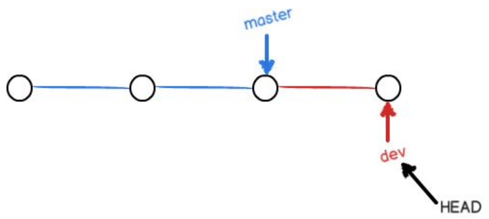


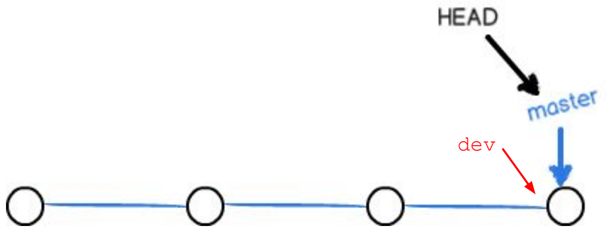


```
*   ff484ee (HEAD -> master) fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 (origin/master) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

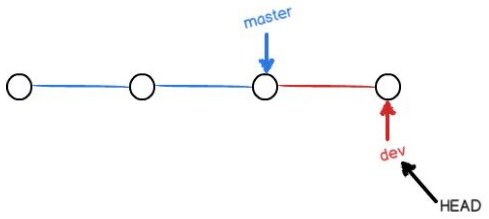
5. Branch

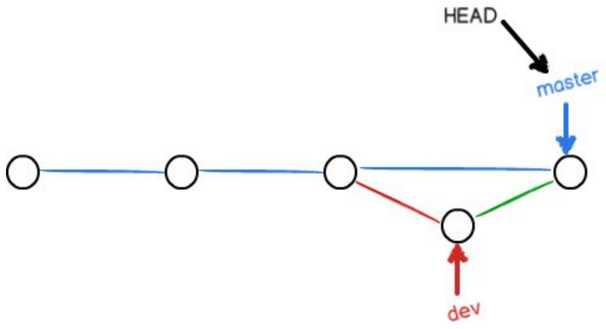
- Merging Strategy
  - Fast-forward Merge



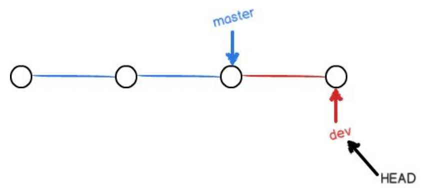


  - Recursive Merge

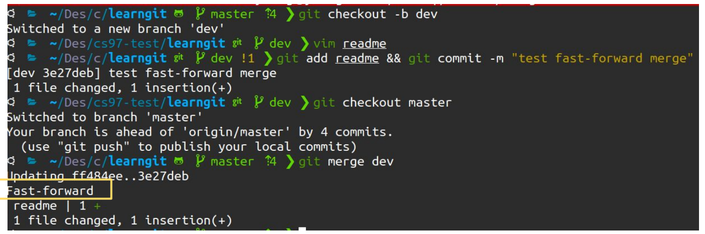




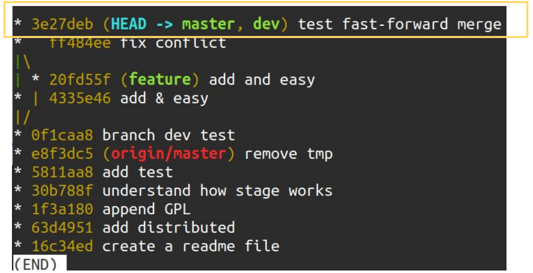
5. Branch



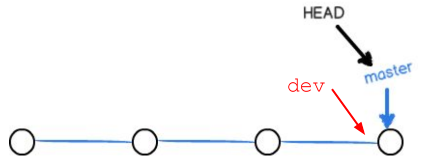
- Merging Strategy
  - Fast-forward Merge
    - **git merge <branch>**



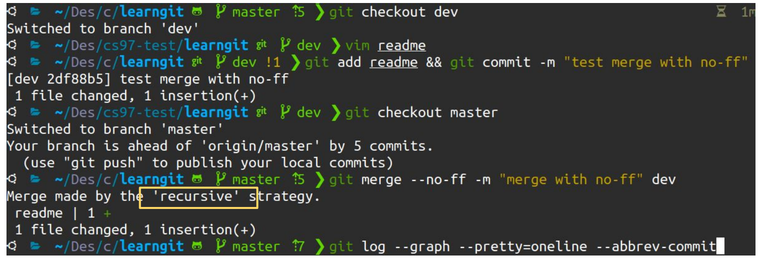
```
~/Des/c/learngit  master ↑4 ❯ git checkout -b dev
Switched to a new branch 'dev'
~/Des/cs97-test/learngit git  dev ❯ vim readme
~/Des/c/learngit git  dev !1 ❯ git add readme && git commit -m "test fast-forward merge"
[dev 3e27deb] test fast-forward merge
 1 file changed, 1 insertion(+)
~/Des/cs97-test/learngit git  dev ❯ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 4 commits.
  (use "git push" to publish your local commits)
~/Des/c/learngit  master ↑4 ❯ git merge dev
Updating ff484ee..3e27deb
Fast-forward
 readme | 1 +
 1 file changed, 1 insertion(+)
```



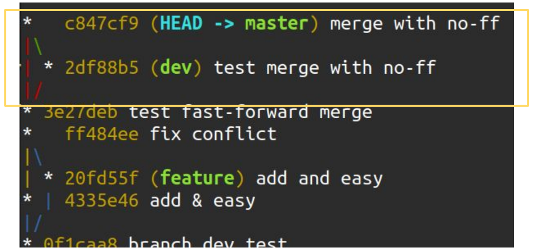
```
* 3e27deb (HEAD -> master, dev) test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 (origin/master) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```



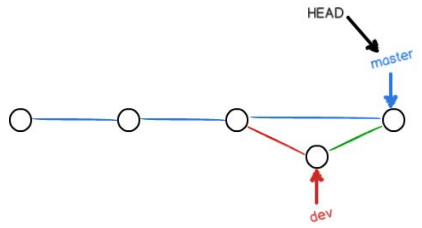
  - Recursive Merge
    - **git merge --no-ff -m "message" <branch>**



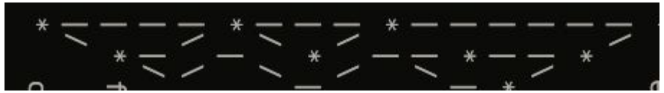
```
~/Des/c/learngit  master ↑5 ❯ git checkout dev                                  1m
Switched to branch 'dev'
~/Des/cs97-test/learngit git  dev ❯ vim readme
~/Des/c/learngit git  dev !1 ❯ git add readme && git commit -m "test merge with no-ff"
[dev 2df88b5] test merge with no-ff
 1 file changed, 1 insertion(+)
~/Des/cs97-test/learngit git  dev ❯ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 5 commits.
  (use "git push" to publish your local commits)
~/Des/c/learngit  master ↑5 ❯ git merge --no-ff -m "merge with no-ff" dev
Merge made by the 'recursive' strategy.
 readme | 1 +
 1 file changed, 1 insertion(+)
~/Des/c/learngit  master ↑7 ❯ git log --graph --pretty=oneline --abbrev-commit
```



```
*   c847cf9 (HEAD -> master) merge with no-ff
|\
| * 2df88b5 (dev) test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
```



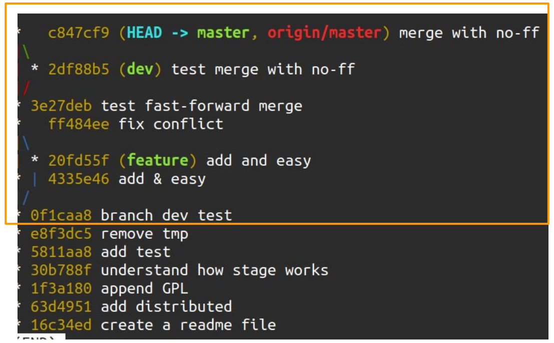
5. Branch

- **Rebase** v.s. Merge
  - Both designed to integrate two branches
  - With merge, we probably get: 
  - Someone may say:
    - I don’t like this, I want to make my history a straight timeline (much cleaner)
  - Use Rebase: `git rebase`
    - Rewrite the history
    - Address all conflicts at a time

5. Branch

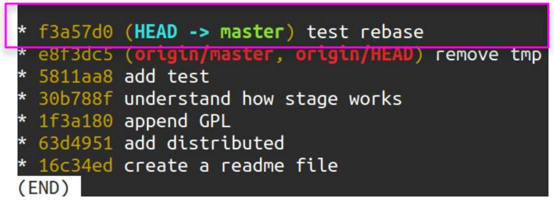
- Rebase v.s. Merge
    - E.g.:
    - Suppose my friend and I are working on the same remote repo
    - We clone/pull the repo/changes to local
    - My friend push his/her updates to remote
    - I also made some local changes
    - Now I want to push my changes to remote
    - Conflict!

My friend's changes (in remote repo)



```
*   c847cf9 (HEAD -> master, origin/master) merge with no-ff
|\
| * 2df88b5 (dev) test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
```

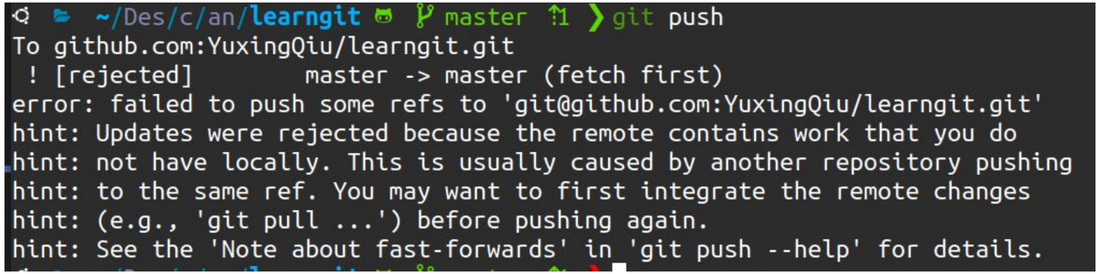
My changes (in local repo)



```
* f3a57d0 (HEAD -> master) test rebase
* e8f3dc5 (origin/master, origin/HEAD) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

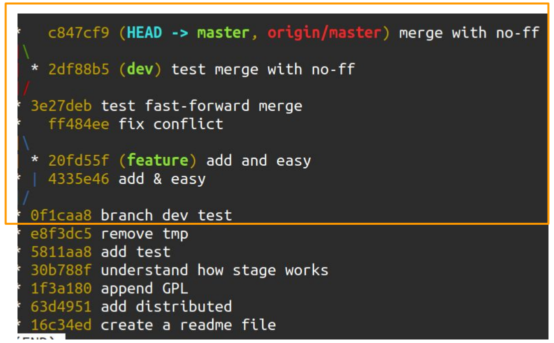
5. Branch

- Rebase v.s. Merge
  - E.g.:
  - Suppose my friend and I are working on the same remote repo
  - We clone/pull the repo/changes to local
  - My friend push his/her updates to remote
  - I also made some local changes
  - Now I want to push my changes to remote
  - Conflict!
  - If I push:



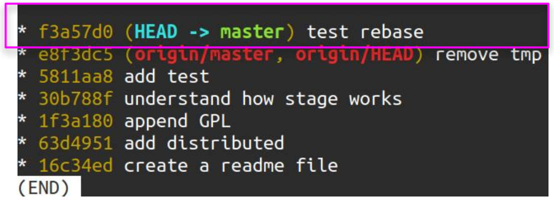
```
~/Des/c/an/learngit  master ⇡1 ❯ git push
To github.com:YuxingQiu/learngit.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:YuxingQiu/learngit.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

My friend's changes (in remote repo)



```
*   c847cf9 (HEAD -> master, origin/master) merge with no-ff
|\
| * 2df88b5 (dev) test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
```

My changes (in local repo)

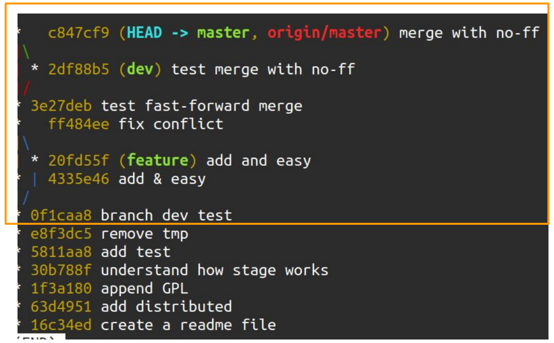


```
* f3a57d0 (HEAD -> master) test rebase
* e8f3dc5 (origin/master, origin/HEAD) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

5. Branch

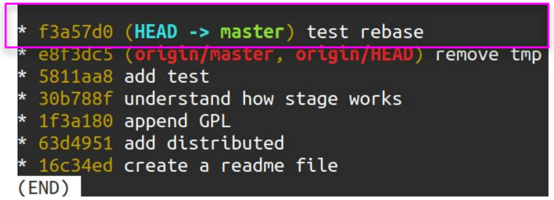
- Rebase v.s. Merge
  - E.g.:
  - Suppose my friend and I are working on the same remote repo
  - We clone/pull the repo/changes to local
  - My friend push his/her updates to remote
  - I also made some local changes
  - Now I want to push my changes to remote
  - Conflict!
  - I will pull and deal with the conflicts
    - `git pull` (`git fetch && git merge`)
    - We have discussed before

My friend's changes (in remote repo)



```
*   c847cf9 (HEAD -> master, origin/master) merge with no-ff
|\
| * 2df88b5 (dev) test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
```

My changes (in local repo)

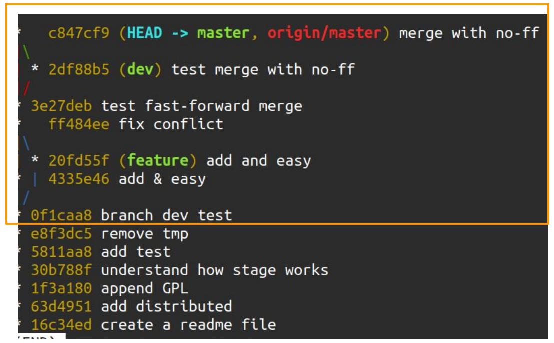


```
* f3a57d0 (HEAD -> master) test rebase
* e8f3dc5 (origin/master, origin/HEAD) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

5. Branch

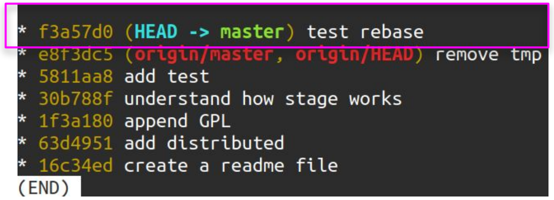
- Rebase v.s. Merge
  - E.g.:
  - Suppose my friend and I are working on the same remote repo
  - We clone/pull the repo/changes to local
  - My friend push his/her updates to remote
  - I also made some local changes
  - Now I want to push my changes to remote
  - Conflict!
  - I will do rebase
    - `git rebase`

My friend's changes (in remote repo)



```
*   c847cf9 (HEAD -> master, origin/master) merge with no-ff
|\
| * 2df88b5 (dev) test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
```

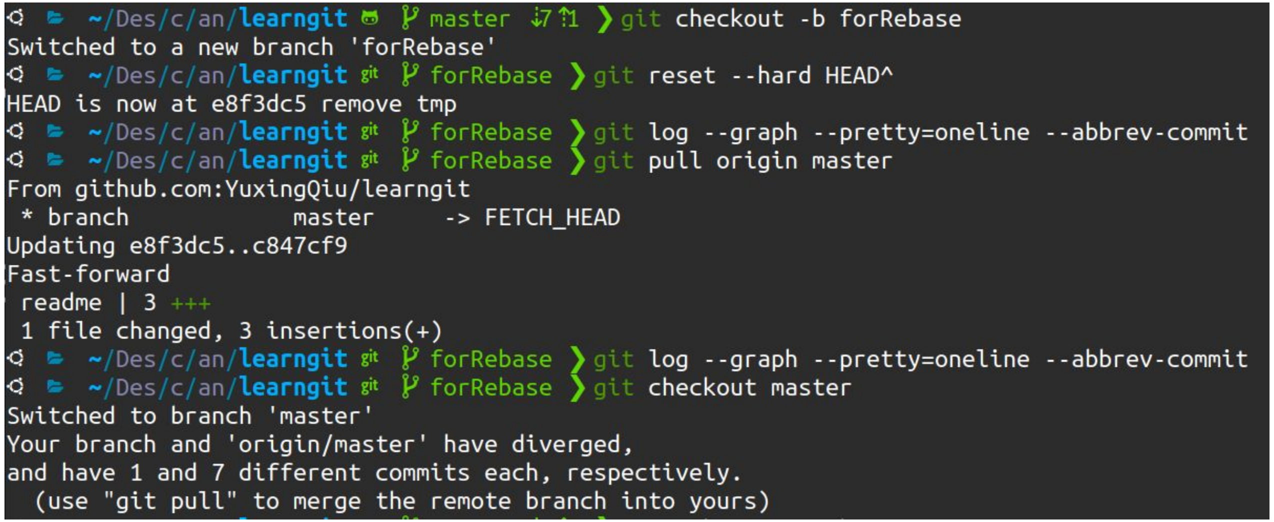
My changes (in local repo)



```
* f3a57d0 (HEAD -> master) test rebase
* e8f3dc5 (origin/master, origin/HEAD) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

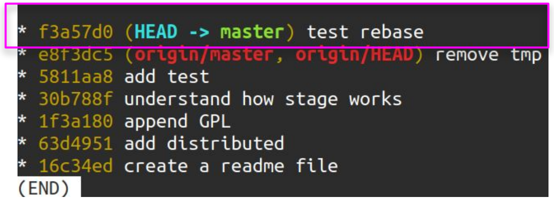
5. Branch

- Rebase v.s. Merge
  - I will do rebase
    - **git rebase**

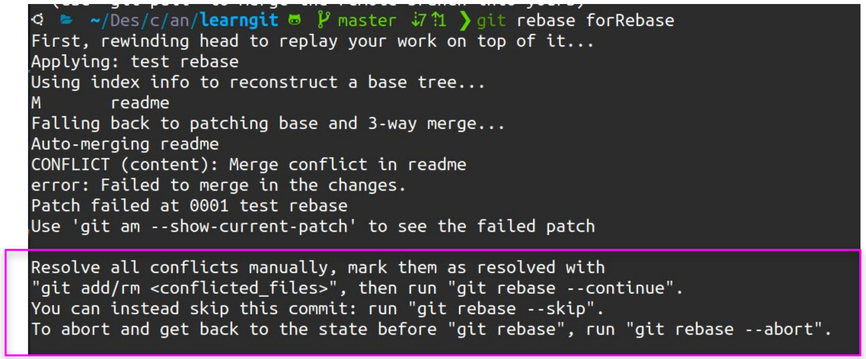


```
~/Des/c/an/learngit  master ⇣7⇡1 ❯ git checkout -b forRebase
Switched to a new branch 'forRebase'
~/Des/c/an/learngit git  forRebase ❯ git reset --hard HEAD^
HEAD is now at e8f3dc5 remove tmp
~/Des/c/an/learngit git  forRebase ❯ git log --graph --pretty=oneline --abbrev-commit
~/Des/c/an/learngit git  forRebase ❯ git pull origin master
From github.com:YuxingQiu/learngit
 * branch            master     -> FETCH_HEAD
Updating e8f3dc5..c847cf9
Fast-forward
 readme | 3 +++
 1 file changed, 3 insertions(+)
~/Des/c/an/learngit git  forRebase ❯ git log --graph --pretty=oneline --abbrev-commit
~/Des/c/an/learngit git  forRebase ❯ git checkout master
Switched to branch 'master'
Your branch and 'origin/master' have diverged,
and have 1 and 7 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
```

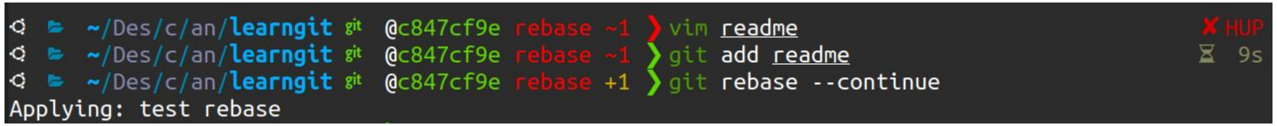
My changes (in local repo)



```
* f3a57d0 (HEAD -> master) test rebase
* e8f3dc5 (origin/master, origin/HEAD) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```



```
~/Des/c/an/learngit  master ⇣7⇡1 ❯ git rebase forRebase
First, rewinding head to replay your work on top of it...
Applying: test rebase
Using index info to reconstruct a base tree...
M       readme
Falling back to patching base and 3-way merge...
Auto-merging readme
CONFLICT (content): Merge conflict in readme
error: Failed to merge in the changes.
Patch failed at 0001 test rebase
Use 'git am --show-current-patch' to see the failed patch

Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
```

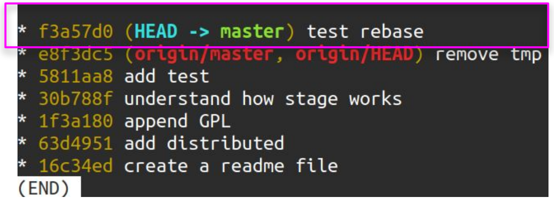


```
~/Des/c/an/learngit git  @c847cf9e rebase ~1 ❯ vim readme                      ✘ HUP
~/Des/c/an/learngit git  @c847cf9e rebase ~1 ❯ git add readme                  ⌛ 9s
~/Des/c/an/learngit git  @c847cf9e rebase +1 ❯ git rebase --continue
Applying: test rebase
```

5. Branch

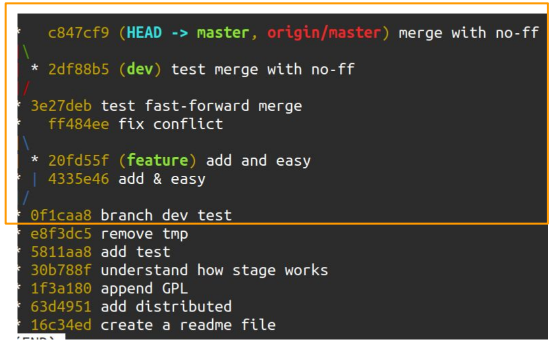
- Rebase v.s. Merge
  - Compare **rebase** and **merge**

My changes (in local repo)



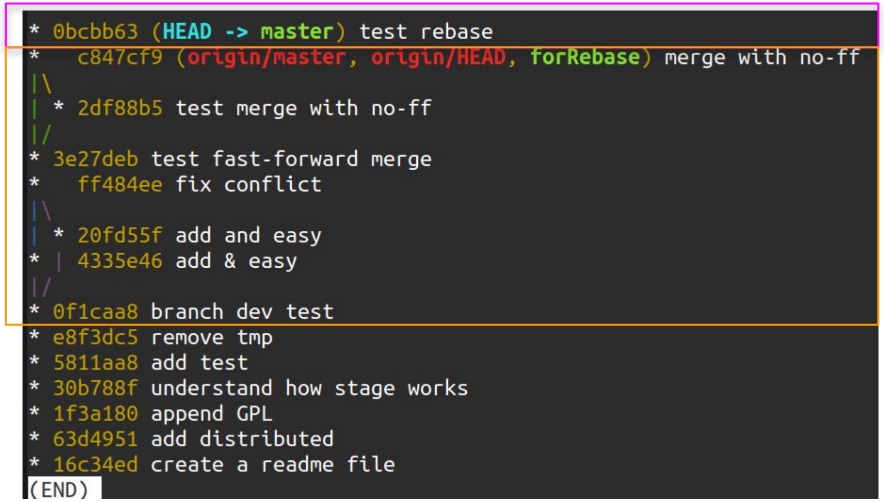
```
* f3a57d0 (HEAD -> master) test rebase
* e8f3dc5 (origin/master, origin/HEAD) remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

My friend's changes (in remote repo)



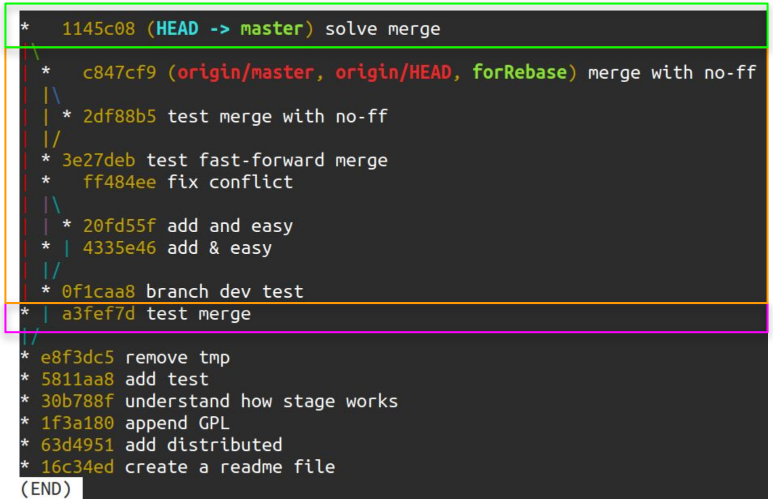
```
*   c847cf9 (HEAD -> master, origin/master) merge with no-ff
|\
| * 2df88b5 (dev) test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f (feature) add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
```

rebase



```
* 0bcbb63 (HEAD -> master) test rebase
*   c847cf9 (origin/master, origin/HEAD, forRebase) merge with no-ff
|\
| * 2df88b5 test merge with no-ff
|/
* 3e27deb test fast-forward merge
*   ff484ee fix conflict
|\
| * 20fd55f add and easy
* | 4335e46 add & easy
|/
* 0f1caa8 branch dev test
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

merge Extra commit to solve merge



```
*   1145c08 (HEAD -> master) solve merge
|\
| *   c847cf9 (origin/master, origin/HEAD, forRebase) merge with no-ff
| |\
| | * 2df88b5 test merge with no-ff
| |/
| * 3e27deb test fast-forward merge
| *   ff484ee fix conflict
| |\
| | * 20fd55f add and easy
| * | 4335e46 add & easy
| |/
| * 0f1caa8 branch dev test
* | a3fef7d test merge
|/
* e8f3dc5 remove tmp
* 5811aa8 add test
* 30b788f understand how stage works
* 1f3a180 append GPL
* 63d4951 add distributed
* 16c34ed create a readme file
(END)
```

Contents

5. Tag

- Give your commit a nickname
- Create a tag
  - `git tag <tag_name>` ## default commit: HEAD
  - `git tag <tag_name> <commit_id>`
  - `git tag <tag_name> -m "tag message"`
- List all tags
  - `git tag`
- List tag info
  - `git show <tag_name>`
- Delete a tag
  - `git tag -d <tag_name>`
- Push a tag
  - `git push origin <tag_name>`
- Push all tags
  - `git push origin --tags`
- Delete tag in remote repo
  - `git push origin :refs/tags/<tagname>`