1. 配置用户名和邮箱

$ git config --global user.name "qiuyunliang"

$ git config --global user.email "qiuyunliang@star-net.cn"

1. 创建版本库

$ mkdir learngit

$ cd learngit

$ pwd

/Users/michael/learngit

$ git init

Initialized empty Git repository in /Users/michael/learngit/.git/

1. 把文件添加到仓库

$ git add readme.txt

1. 把文件提交到仓库

$ git commit -m "wrote a readme file"

$ git commit –amend // recommit

[master (root-commit) cb926e7] wrote a readme file

1 file changed, 2 insertions(+)

create mode 100644 readme.txt

1. 查看仓库当前的状态

$ 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.txt

#

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

1. 查看文件修改情况

$ git diff readme.txt

1. 查看提交记录

$ git log

$ git log --pretty=oneline

1. 回退到上一个版本

$ git reset --soft HEAD^ // 将版本库指向HEAD^，删除撤销的提交记录，撤销的全部退到暂存区

$ git reset --mixed HEAD^ // 将版本库和暂存区指向HEAD^

$ git reset --hard HEAD^ // 将版本库、暂存区和工作区都指向HEAD^

$ git reset HEAD <file> // 取消已经暂存的文件

$ git checkout -- <file> // 取消对文件的修改

$ git revert HEAD // 撤销前一次 commit

$ git revert HEAD^               // 撤销前前一次 commit

$ git revert <commit> // 撤销指定的commit，版本继续增加

1. 记录你的每一次命令

$ git reflog

1. 查看工作区和版本库里面最新版本的区别

$ git diff HEAD -- readme.txt

diff --git a/readme.txt b/readme.txt

index 76d770f..a9c5755 100644

--- a/readme.txt

+++ b/readme.txt

@@ -1,4 +1,4 @@

Git is a distributed version control system.

Git is free software distributed under the GPL.

Git has a mutable index called stage.

-Git tracks changes.

+Git tracks changes of files.

1. 撤销工作区的修改

$ git checkout -- readme.txt

说明：

1. readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；
2. readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。
3. 把暂存区的修改撤销掉（unstage），重新放回工作区

$ git reset HEAD readme.txt

Unstaged changes after reset:

M readme.txt

1. 直接在文件管理器中把没用的文件删除，或者用rm命令删除

$ rm test.txt

1. 从版本库中删除该文件

$ git rm test.txt

rm 'test.txt'

$ git commit -m "remove test.txt"

[master d17efd8] remove test.txt

1 file changed, 1 deletion(-)

delete mode 100644 test.txt

1. 创建dev分支，然后切换到dev分支

$ git checkout -b dev

Switched to a new branch 'dev'

$ git branch dev

$ git checkout dev

Switched to branch 'dev'

1. 把dev分支的工作成果合并到master分支上

$ git merge dev

Updating d17efd8..fec145a

Fast-forward

readme.txt | 1 +

1 file changed, 1 insertion(+)

1. 删除dev分支

$ git branch -d dev

Deleted branch dev (was fec145a).

1. 分支的合并情况

$ git log --graph --pretty=oneline --abbrev-commit

\* 59bc1cb conflict fixed

|\

| \* 75a857c AND simple

\* | 400b400 & simple

|/

\* fec145a branch test

…

1. 把当前工作现场暂存，等以后恢复现场后继续工作

$ git stash

1. 使用git stash list命令查看工作现场存放位置

$ git stash list

stash@{0}: WIP on dev: 6224937 add merge

1. 恢复工作现场

$ git stash pop

# On branch dev

# Changes to be committed:

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

#

# new file: hello.py

#

# 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.txt

#

Dropped refs/stash@{0} (f624f8e5f082f2df2bed8a4e09c12fd2943bdd40)

$ git stash list

$ git stash apply stash@{0}

说明：

1. 用git stash apply恢复，但是恢复后，stash内容并不删除，需要用git stash drop来删除；
2. 用git stash pop，恢复的同时把stash内容删除。
3. 强行删除分支

$ git branch -D feature-vulcan

Deleted branch feature-vulcan (was 756d4af).

1. 查看远程库的信息

$ git remote -v

1. 推送分支

$ git push origin master

$ git push origin dev

说明：

把该分支上的所有本地提交推送到远程库。推送时，要指定本地分支，Git就会把该分支推送到远程库对应的远程分支上。

1. 在另一台电脑（注意要把SSH Key添加到GitHub）或者同一台电脑的另一个目录下克隆

$ git clone git@github.com:michaelliao/learngit.git

Cloning into 'learngit'...

remote: Counting objects: 46, done.

remote: Compressing objects: 100% (26/26), done.

remote: Total 46 (delta 16), reused 45 (delta 15)

Receiving objects: 100% (46/46), 15.69 KiB | 6 KiB/s, done.

Resolving deltas: 100% (16/16), done.

1. 要在dev分支上开发，必须创建远程origin的dev分支到本地

$ git checkout -b dev origin/dev

1. 现在，某人就可以在dev上继续修改，然后时不时地把dev分支push到远程

$ git commit -m "add /usr/bin/env"

[dev 291bea8] add /usr/bin/env

1 file changed, 1 insertion(+)

$ git push origin dev

Counting objects: 5, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 349 bytes, done.

Total 3 (delta 0), reused 0 (delta 0)

To git@github.com:michaelliao/learngit.git

fc38031..291bea8 dev -> dev

1. 某人已经向origin/dev分支推送了他的提交，而碰巧自己也对同样的文件作了修改，并试图推送

$ git add hello.py

$ git commit -m "add coding: utf-8"

[dev bd6ae48] add coding: utf-8

1 file changed, 1 insertion(+)

$ git push origin dev

To git@github.com:michaelliao/learngit.git

! [rejected] dev -> dev (non-fast-forward)

error: failed to push some refs to 'git@github.com:michaelliao/learngit.git'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')

hint: before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

1. 推送失败，某人的最新提交和自己试图推送的提交有冲突，先用git pull把最新的提交从origin/dev抓下来，然后在本地合并解决冲突，再推送

$ git pull

remote: Counting objects: 5, done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 3 (delta 0)

Unpacking objects: 100% (3/3), done.

From github.com:michaelliao/learngit

fc38031..291bea8 dev -> origin/dev

There is no tracking information for the current branch.

Please specify which branch you want to merge with.

See git-pull(1) for details

git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

git branch --set-upstream dev origin/<branch>

1. 此时，git pull也失败了，原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示设置dev和origin/dev的链接

$ git branch --set-upstream dev origin/dev

Branch dev set up to track remote branch dev from origin.

1. 再次尝试pull

$ git pull

Auto-merging hello.py

CONFLICT (content): Merge conflict in hello.py

Automatic merge failed; fix conflicts and then commit the result.

1. 最后git pull成功，但是合并有冲突，需要手动按分支管理中的解决冲突方法解决，解决后提交再次push

$ git commit -m "merge & fix hello.py"

[dev adca45d] merge & fix hello.py

$ git push origin dev

Counting objects: 10, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (5/5), done.

Writing objects: 100% (6/6), 747 bytes, done.

Total 6 (delta 0), reused 0 (delta 0)

To git@github.com:michaelliao/learngit.git

291bea8..adca45d dev -> dev

总结：

1. 查看远程库信息，使用git remote -v；
2. 本地新建的分支如果不推送到远程，对其他人就是不可见的；
3. 从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；
4. 在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地和远程分支的名称最好一致；
5. 建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；
6. 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。
7. 打一个新标签

$ git tag v1.0

$ git tag v0.9 6224937

$ git tag

v0.9

v1.0

$ git tag -a v0.1 -m "version 0.1 released" 3628164

1. 删除标签

$ git tag -d v0.1

1. 推送某个标签到远程

$ git push origin v1.0

Total 0 (delta 0), reused 0 (delta 0)

To git@github.com:michaelliao/learngit.git

\* [new tag] v1.0 -> v1.0

1. 一次性推送全部尚未推送到远程的本地标签

$ git push origin --tags

Counting objects: 1, done.

Writing objects: 100% (1/1), 554 bytes, done.

Total 1 (delta 0), reused 0 (delta 0)

To git@github.com:michaelliao/learngit.git

\* [new tag] v0.2 -> v0.2

\* [new tag] v0.9 -> v0.9

1. 如果标签已经推送到远程，要删除远程标签需先从本地删除

$ git tag -d v0.9

Deleted tag 'v0.9' (was 6224937)

1. 然后，从远程删除。删除命令也是push，但格式不同

$ git push origin :refs/tags/v0.9

To git@github.com:michaelliao/learngit.git

- [deleted] v0.9

总结：

1. 命令git push origin <tagname>可以推送一个本地标签；
2. 命令git push origin --tags可以推送全部未推送过的本地标签；
3. 命令git tag -d <tagname>可以删除一个本地标签；
4. 命令git push origin :refs/tags/<tagname>可以删除一个远程标签。
5. 一个完整的.gitignore文件

# Windows:

Thumbs.db

ehthumbs.db

Desktop.ini

# Python:

\*.py[cod]

\*.so

\*.egg

\*.egg-info

dist

build

# My configurations:

db.ini

deploy\_key\_rsa

1. 需要添加一个文件到Git，但发现添加不了，原因是这个文件被.gitignore忽略

$ git add App.class

The following paths are ignored by one of your .gitignore files:

App.class

Use -f if you really want to add them.

1. 如果确实想添加该文件，可以用-f强制添加到Git

$ git add -f App.class

1. 或者可能是.gitignore写得有问题，需要找出来规则写错，可以用git check-ignore命令检查

$ git check-ignore -v App.class

.gitignore:3:\*.class App.class

1. 配置别名

$ git config --global alias.co checkout

$ git config --global alias.ci commit

$ git config --global alias.br branch

$ git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d

%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

说明：

1. 每个仓库的Git配置文件都放在.git/config文件中；
2. 当前用户的Git配置文件放在用户主目录下的一个隐藏文件.gitconfig中。
3. 克隆远程版本库

$ git clone –o <主机名origin> <website> <local path>

1. 查看远程库信息

$ git remote –v

$ git remote show <主机名>

$ git remote add <主机名> <网址>

$ git remote rm <主机名>

$ git remote rename <原主机名> <新主机名>

1. 拉取版本库更新

$ git fetch <远程主机名> <分支名>

1. 取回远程主机某个分支的更新

$ git pull <远程主机名> <远程分支名>:<本地分支名>

$ git fetch origin

$ git merge origin/next

$ git branch --set-upstream master origin/next

1. 本地分支合并远程分支

$ git merge origin/master

$ git rebase origin/master

1. 推送本地分支至远程分支

$ git push <远程主机名> <本地分支名>:<远程分支名>

$ git push origin :master

# 等同于

$ git push origin --delete master

1. 删除远程

git remote remove origin

1. 添加远程主机

$ git remote add origin https://github.com/yunliangqiu/practice.git

1. 将本地分支与远程分支建立连接

$ git push --set-upstream origin master

1. 删除远程主机

$ git remote remove "dev"

1. 设置区分文件名大小写

$ git config core.ignorecase false