Engineering能力提升-01-git

包括：原理、日志提交、代码提交、撤销修改、创建tag、创建branch、合并分支、commit规范等等

☆ `Git` is not an acronym (/'ækrənɪm/首字母缩略词) but rather an expression of intent. 由于Linux本人想用分布式的代码管理工具而不是集中式的(SVN,i.e. Apache/əˈpæʃ/ Subversion)，he decided to develop a distributed software versioning and revision control system.
☆ `snapshot` 快照是对当前文件(夹)状态信息的保存， `copy` 备份是对当前文件(夹)复制。

1. Git Installation

- 下一项 | 返回top

1.1 Windows安装

- 访问地址:https://git-scm.com/ (https://git-scm.com/)
- 点击下载:https://git-scm.com/download/gui/win (https://git-scm.com/download/gui/win)
- 开始安装，下一步…

1.2 Linux安装

- `yum install git`
- 或者下载源码:https://github.com/git/git.git (https://github.com/git/git.git) 编译安装。

1.3 Mac安装

- 打开终端，执行命令 `brew install git@2.21.0` (BW:后面为版本号)

2. Git Config

- 上一项 下一项 | 返回top

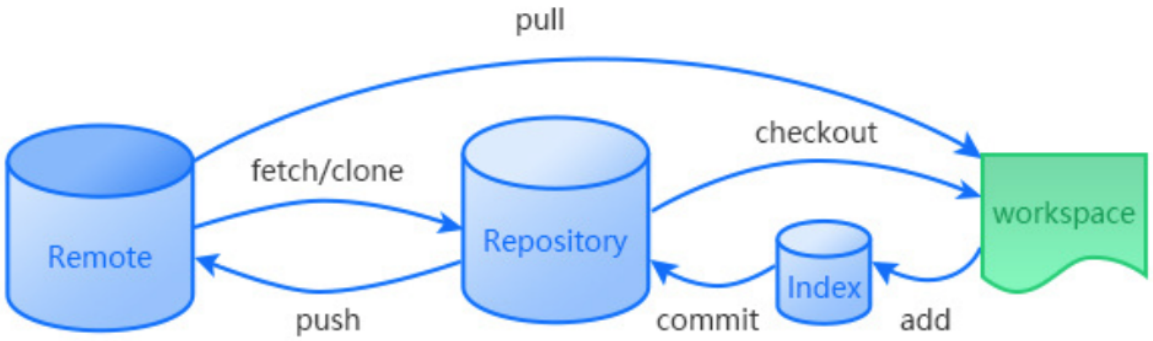
- 配置用户名: `git config --global user.name “xxx”`
- 配置邮箱: `git config --global user.email “xxx"`
- 配置大小写敏感: `git config --global core.ignorecase false`
- 查看配置信息: `git config --list` (BW:my local git account与github.com account不一样)

3. Structure

3. 基本原理

- 上一项 | 下一项 | 返回top
- ☆ remote github 远程仓库 (e.g. github repository).
- ☆ repository 本地仓库，也就是.git文件夹.
- ☆ index/stage 本地暂存区，用于临时存放你的改动，事实上，它只是一个文件，保存即将提交到文件列表的改动信息。

- ☆ workspace 本地工作区，就是你平时看到和编辑文件夹区域，也就是terminal工作区。

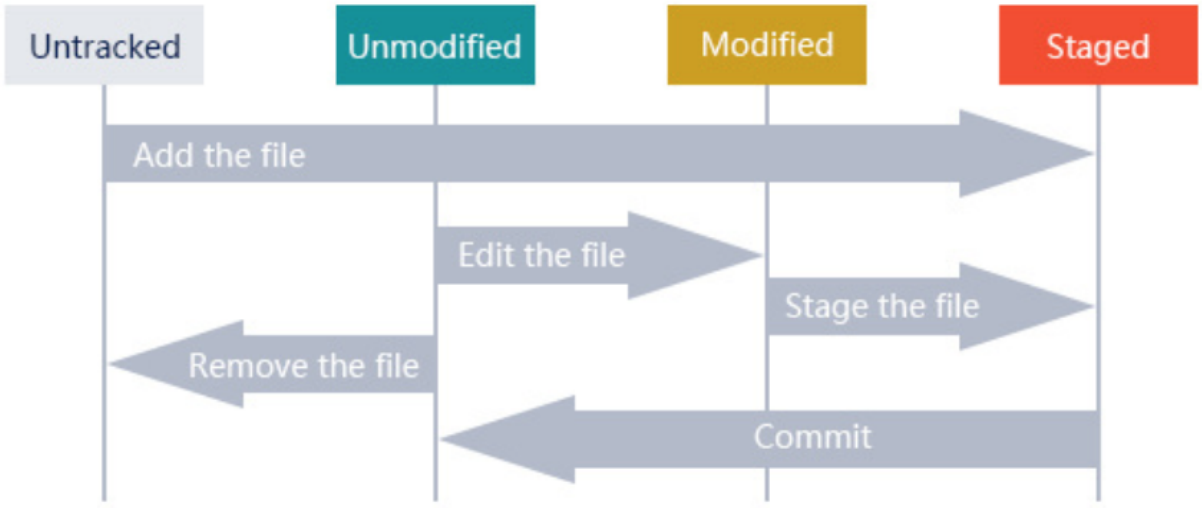


- ☆ BW:注意它们之间的相互转换及指令，e.g. clone, pull, add, commit, push等。

4. Workflow

4. 工作流程

- 上一项 | 下一项 | 返回top



Git文件状态(版本控制就是对文件的版本控制)

- ☆ Untracked:未跟踪，在此文件夹中，但**没有加入到git库**，不参与版本控制，通过git add filenameBW 状态变为staged (&track状态)。
- ☆ Unmodified:文件已经入库，未修改，即版本库中的文件快照内容与文件夹一致，这种类型的文件有两种去处，如果被修改，变为modified，如果被移除版本库git rm

filenameBW，则被删除也就变为Untracked。

- ☆ Modified:文件已修改，仅仅是修改，并没有进行其他操作，这个文件有两个去处，第一个是staged，第二个是unmodified。
- ☆ Staged:执行git commit -m “comments here BW” 则将修改同步到库(repository)中，这时库中的文件和本地文件虽为一致，文件为Unmodified。执行git reset HEAD filename取消暂存，文件状态为modified。
- ☆ 查看git状态： `git status .`

example

```
$vim README.md
$git add README.md
$git commit –m "add README.md file"
```

5. Git Init

- 上一项 | 下一项 | 返回top

5.1 (Mac)

1. `cd work_directory`
2. `git init` #☆ 在当前工作目录下便会产生.git的文件夹。

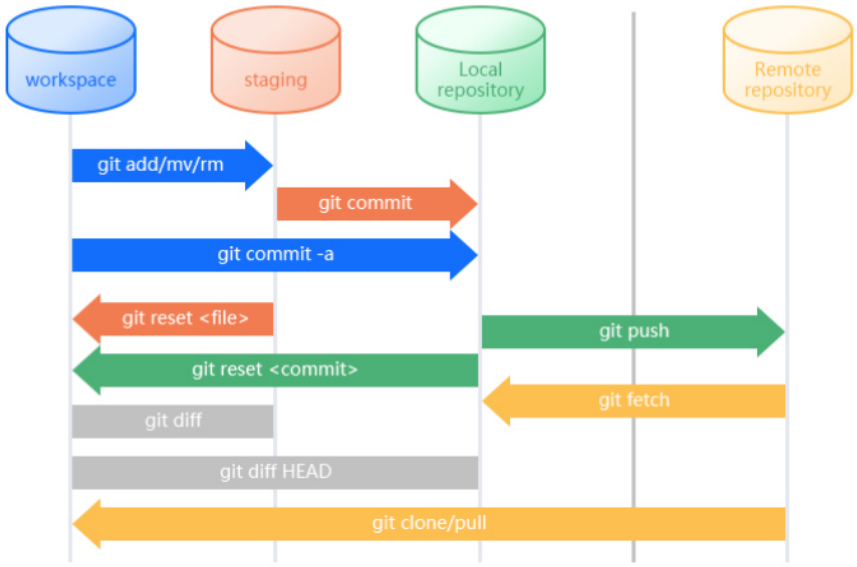
5.2 (Windows)

- 右击工作文件夹，选择git bash即可进入命令编辑界面。

6. Git Commands

6. git基本操作

- 上一项 | 下一项 | 返回top



git add	添加文件	git mv	移动或重命名文件，目录或符号链接
git rm	从工作区和索引中删除文件	git commit	添加文件到本地仓库
git reset	回退版本/提交	git push	将本地修改推送到远程库
git fetch	更新远程库信息	git diff	显示提交和工作树等之间的更改
git clone	拉取远程仓库到本地	git pull	将远程库最新修改更新到本地
git status	显示工作目录和暂存区的状态		

6.1 Commands for Basic Operations

① 选择一个合适的地方，创建(mkdir)一个空目录(版本库)
② 初始化版本库 (if not done yet)

```
$ git init
$ ll –ah 或 ls –ah
drwxr–xr–x 7 root root 4.0K Jan 11 17:32 .git
```

注: init之后会生成一个.git隐藏文件夹，可以查看里面的信息:

```
$cd .git
$cd logs
$cd refs/heads
$cat master
```

③ 新增文件

在创建并编辑好文件后，用 `git add FileName.XXX` 变为staged状态，然后用 `git commit -m "XXXXXXX"` 变为unmodified状态。

```
$git add readme.txt
$git commit –m "add readme.txt file"
[master (root-commit) 3289742] add readme 1 file changed, 2 insertions(+)
create mode 100644 readme
```

6.2 Deal with Bugs – switch to desired historical versions

☆☆提交了版本，发现有bug想撤销最近一次commit怎么办!!! 或者发现还是之前某历史版本的结果好想回滚到那个版本怎么办!!!

🍒 使用 `git reset --hard HEAD^` 强制回滚至上个版本或指定 `commit id` 。(可followed by `git status` 查看reset结果)

☆ 注: git的HEAD指针总是指向最新的commit，次新的commit的指针地址是HEAD^.

☆☆☆ 注: **如果想强制回滚至上上一个状态或者任意指定状态:**

```
$cat readme.txt
line 1. A
line 2. B
line 3. C
line 4. D
$git reflog
6d0f19f (HEAD -> master) HEAD@{0}: commit: readme-D
21160f6 HEAD@{1}: commit: readme-C
b671cec HEAD@{2}: commit: readme-B
7369df0 HEAD@{3}: commit (initial): readme-A
$git reset --hard (+上面显示在最前面的id,i.e. 7369df0)
$cat readme.txt
line 1. A
$git reflog
7369df0 (HEAD -> master) HEAD@{0}: reset: moving to 7369df0
6d0f19f HEAD@{1}: commit: readme-D
21160f6 HEAD@{2}: commit: readme-C
b671cec HEAD@{3}: commit: readme-B
7369df0 (HEAD -> master) HEAD@{4}: commit (initial): readme-A
```

🍒 使用 `git log` 查看”提交到本地仓库”的日志; 使用 `history` 查看历史命令。

小结：**HEAD**指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭。要重返未来，用 **`git reflog`** 查看命令历史，以便确定要回到未来的哪个版本。

6.3 Deal with Bugs – remove content edition

☆☆ 内容有错误，但是已经git add到了暂存区想撤销add怎么办!!!

🍒 `git reset HEAD FileName.xxx` ，撤销暂存区(index)的修改。

🍒 `git checkout -- FileName.xxx` ，撤销工作区(workspace)的修改。

6.4 Deal with Bugs – retrieve deleted files

☆☆ 本地文件误删了怎么办by `rm FileName.xxx` ，还能找回来吗!!! 实际上是删除了本地工作区(workspace)的文件，但是可以从已提交本文件的本地仓库(repository)里找回来。

- 别着急，可以继续使用 `git checkout -- FileName.xxx` 命令找回来文件
- 如果不是误删，就是不想要这个文件了，可以使用如下命令:

```
$ git rm FileName.xxx    #☆本地工作区没有了本地仓库也没有了，所以是永久删除!!!所以一般用rm filename.xxx指令还可以找回。
$ git commit –m “remove FileName.xxx”
[master d46f35e] remove FileName.xxx
1 file changed, 1 deletion(–)
delete mode 100644 readme
```

+ [x] 场景1:当你改乱了工作区某个文件的内容，想撤销工作区的修改时，用命令git checkout – FileName.xxx。 + [x] 场景2:当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想撤销修改，分两步，第一步用命令git reset HEAD ，就回到了场景1，第二步按场景1操作。 + [x] 场景3:已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。

7. Remote Github

7. 远程仓库

- 上一项 | 下一项 | 返回top

7.1 Github Account and Login Management

- 注册github账号from github.com
- 设置SSH Keys， `ssh-keygen -t rsa -C "youremail@exaple.com"` ,然后 `cd ~` ,然后 `cd .ssh` ,然后复制下 `cat id_rsa.pub` 出的内容。

- 在github个人account下面的settings页面里面的SSH Keys然后点击new SSH然后将上面复制的内容粘贴到Key里面即可。这样，以后再用当前电脑往这个GitHub推送东西就不用再输入密码(免密)了。【不一定非要操作】
- Github支持设置多个SSH Key。

7.2 Set Remote Repository

- 在github.com新建名为learngit的新版本库。
- 根据github的提示，如果本地仓库还没有被新建(即init)那么要先 `git init` 新建本地仓库，然后在本地仓库运行命令: `git remote add origin https://github.com/wonderclouds/learngit.git` 建立本地仓库与远程仓库的连接/关联。
- 连接/关联后，使用命令 `git push -u origin master` 第一次推送 `master分支` 的所有内容。【☆☆☆BW:master指本地仓库,origin指远程仓库】
- 此后，每次本地提交若还是到master分支就可以使用简短命令 `git push origin master` 甚至是 `git push` 推送最新修改。

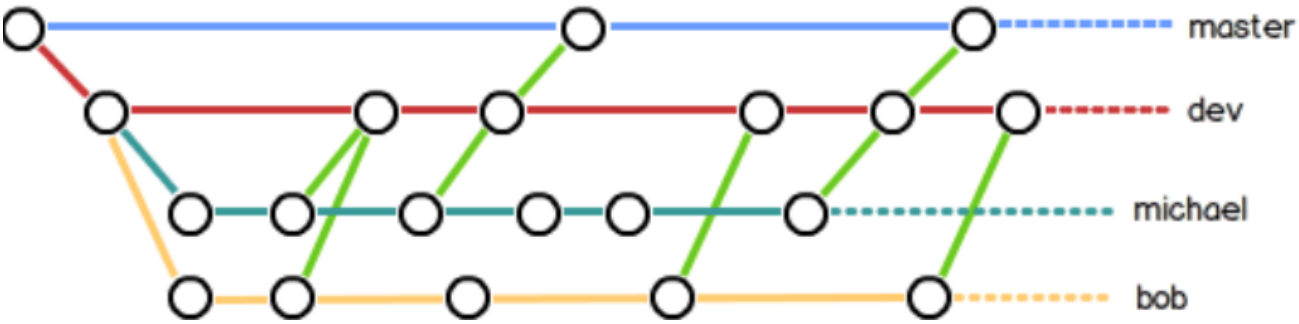
8. Branch and Tag Management

☆☆☆ 8. 协作开发时的分支与标签管理

- 上一项 | 下一项 | 返回top

8.0 下面所有的brach和tag都可以同步在github.com远程仓库上看到才对。

8.1 Brach



1. 首先，master分支应该是测试后非常稳定的，也就是仅用来发布新版本，平时不能在上面干活; 2. 那在哪干活呢?干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时， 再把dev分支合并到master上，在master分支发布1.0版本; 3. 你和你的

小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。 4. 合并分支时，加上–no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

8.1.1 brach cmd

- 查看所有分支及当前分支: `git branch`

结果 e.g. `* master #表示当前指针指向的分支。`

- *创建dev分支: `git checkout -b dev` 或者切换至已有的分支: `git switch -c dev`*
- *查看所有分支及当前分支: `git branch`*

*结果 e.g. * dev，这里*表示当前指针指向的分支。

- 分支内容提交: `git commit -a -m “update file“`
- 切换至master分支(BW:才能git push -u origin master): `git checkout master`或者`git switch master`
- 合并分支: `git merge dev`
- 删除已结束使命的dev分支: `git branch -d dev`

8.2 Tag

☆☆ 通过tag来管理不同的版本,e.g.beta版或发行版。

- 创建标签: `git tag v1.0`
- 查看所有标签: `git tag`
- 创建带有描述信息的标签: `git tag -a v0.1 -m "version 0.1 released"`
- 用命令 `git show <tagname>` 可以看到说明文字
- 如果打错了，可以(本地)删除: `git tag -d v0.1`
- 还可以将标签推到远程仓库: `git push origin v1.0` 或完整的 `git push -u origin v1.0`
- 删除远程标签需要先删除本地标签: `git tag -d v0.9` ,然后 `git push origin :refs/tags/v0.9` 这样远程仓库的v0.9也被删除了。

9. Commit Keywords

- 上一项 | 下一项 | 返回top

开发Commit规范		
Type 用于说明 commit 的类别， 只允许使用下面8个标识	feat	新增 feature
	fix	修复 bug
	docs	仅修改文档，比如 README, CHANGELOG, CONTRIBUTE等等
	refactor	代码重构，没有加新功能或者修复 bug
	chore	改变构建流程、或者增加依赖库、工具等
	revert	回滚到上一个版本
	style	仅修改空格、格式缩进、逗号等等，不改变代码逻辑
	test	测试用例，包括单元测试、集成测试等
scope 用于说明 commit 影响的范围，比如数据层、控制层、视图层等等，视项目不同而不同。		
subject 是 commit 目的的简短描述，不超过50个字符，中文字符描述		

e.g. `git commit -m "docs: update README.md"`

10. Optional HW

- 上一项 | 返回top

- 熟悉github操作及使用
- 基于hexo和githubpage搭建个人博客
- 在自己的博客中发布一篇git相关的技术文章
- 思考上述博客搭建方法的优势和缺点，并调研其他搭建方法