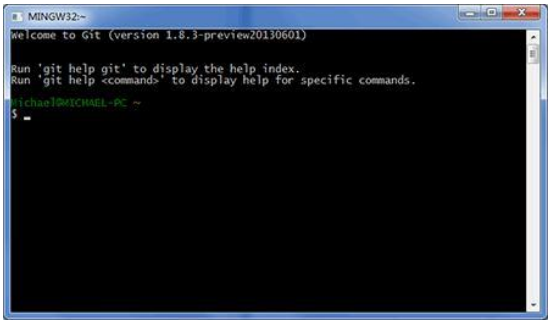
一、Git 初步学习

1、 安装 git

安装后，找到目录下 git bash 点击运行，如图安装成功：



2、 设置你的名字和 Email 地址：

$ git config --global user.name "Your Name"

$ git config --global user.email "email@example.com"

3、 布置一个新仓库

选择一个文件夹，

运行 git bash，

输入$ git init，可以发现当前目录下多了一个.git 的目录

4、 给仓库添加并提交一个文件

（提交修改和提交新文件是一样的两步）

添加（到：暂存区） $ git add xxx.txt

提交（到：当前分支） $ git commit -m "wrote a readme file（这是备注）"

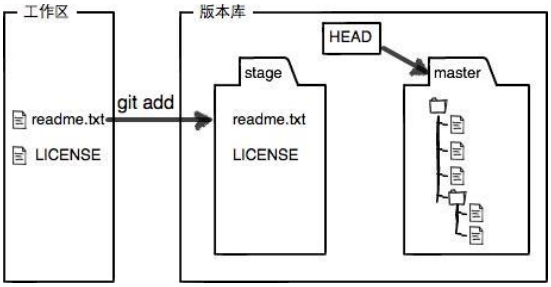
5、 时刻查看仓库当前状态

$ git status

6、 查看提交前，修改了什么

$ git diff xxx.txt

7、 关于“工作区”“版本库”



- “版本库”：工作区的一个隐藏目录.git

其中，最重要的是：

“暂存区”stage、

自动创建的第一个分支“master”、

指向“master”的指针“head”

- “工作区”：电脑我们通常编辑的目录文件

8、 版本回退

上版本 $ git reset --hard HEAD^

上上版本 $ git reset --hard HEAD^^

上 100 个版本 $ git reset --hard HEAD~100

指定版本　　$ git reset --hard id

9、查看日志

查看提交日志（近=>远），可以获取版本 id 和提交明细：

提交日志详细　　$ git log

最近一次提交　　$ git log -1

提交日志简化　　$ git log --pretty=oneline

分支合并图　　$ git log --graph

分支合并图　　$ git log --graph --pretty=oneline --abbrev-commit

分支合并表　　$ git log --pretty=oneline --abbrev-commit

命令日志　　$ git reflog

10、撤销修改

$ git checkout -- xxx.txt　　撤销“工作区”修改

若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；

若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。

总之，就是让这个文件回到最近一次 git commit 或 git add 时的状态。

$ git reset HEAD xxx.txt　撤销“暂存区”修改

该命令既可以回退版本，也可以把暂存区的修改回退到工作区，当我们用 HEAD 时，表示最新的版本。

11、删除文件

删除　$ git rm xxx.txt

提交　$ git commit -m "remove xxx.txt"

12、配置别名

如，用 st 表示 status　　$ git config --global alias.st status（加上--global 是针对当前用户起作用的，如果不加，那只针对当前的仓库起作用）

<很多人都用 co 表示 checkout，ci 表示 commit，br 表示 branch>

二、Git 连接远程仓库

1、 注册 GitHub 账号

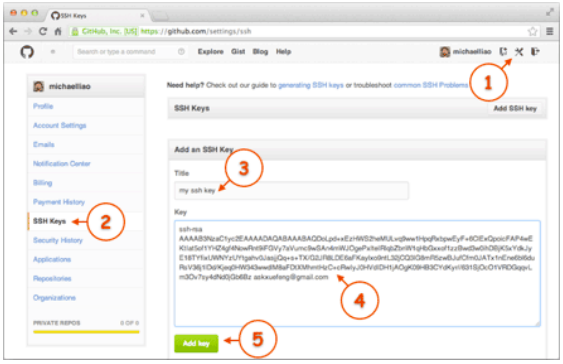
2、 本地创建 SSH Key：

$ ssh-keygen -t rsa -C "youremail@example.com"

找到 id_rsa（私钥）和 id_rsa.pub（公钥）这两个文件

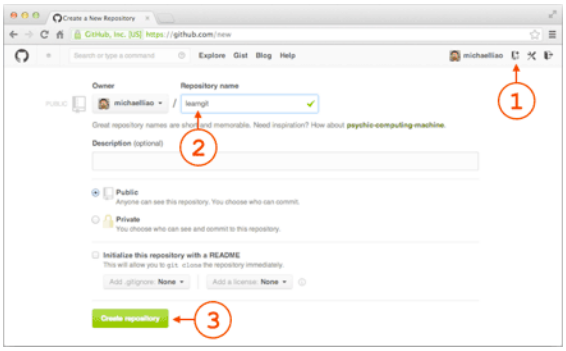
3、 登录 GitHub，打开“Account settings”，“SSH Keys”页面：点“Add SSH Key”，填上任意 Title，在 Key 文本框里粘贴 id_rsa.pub 文件的内容：

GitHub 只要知道了你的公钥，就可以确认只有你自己才能推送。



4、 创建 GitHub 个人仓库

在右上角找到“Create a new repo”按钮，创建一个新的仓库：



5、 关连本地仓库与远程仓库

把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到 GitHub 仓库：

$ git remote add origin git@github.com:账号名/仓库名.git

6、 推送本地分支到远程分支

第一次推送： $ git push -u origin master（推送分支名）

我们第一次推送 master 分支时，加上了-u 参数，Git 不但会把本地的 master 分支内容推送的远程新的 master 分支，还会把本地的 master 分支和远程的 master 分支关联起来，在以后的推送或者拉取时就可以简化命令。

日后推送： $ git push origin master（推送分支名）

如有必要，建立与远程分支关联：

$ git branch --set-upstream-to=远程分支名 本地分支名

抓取远程分支最新版本 $ git pull

解决冲突，提交，再 push 推送分支名

7、 从远程库拷贝至本地库

从零开发，最好先创建远程库，然后再克隆至本地。

克隆至本地： $ git clone git@github.com:账号名/仓库名.git

8、 查看远程仓库信息

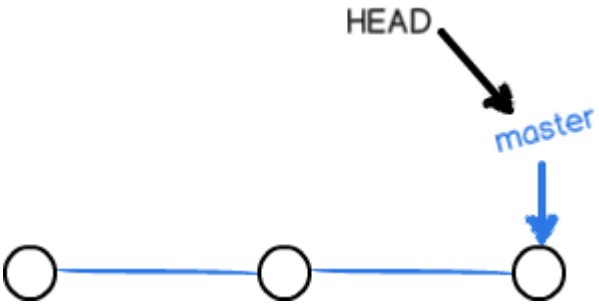
查看： $ git remote

详细查看： $ git remote -v

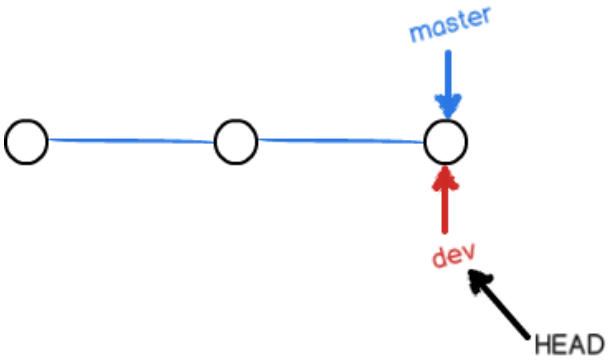
三、Git 使用分支

1、分支创建与合并图示

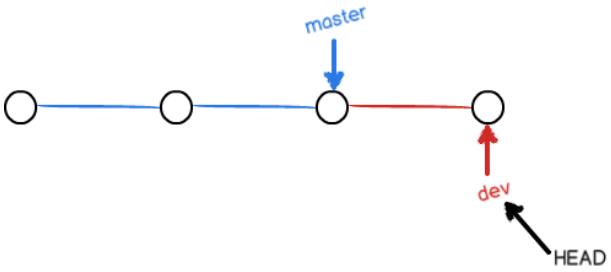
- HEAD 指向当前分支，master 指向主分支



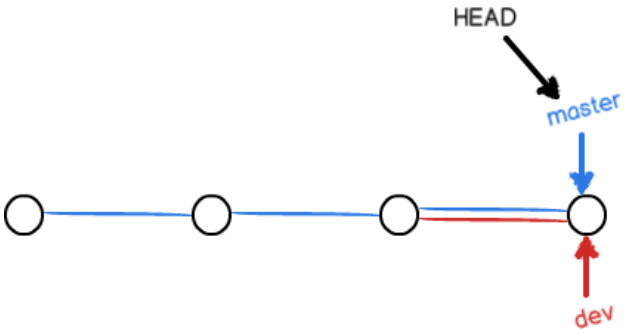
- 创建并切换新分支 dev，HEAD 则指向当前分支 dev



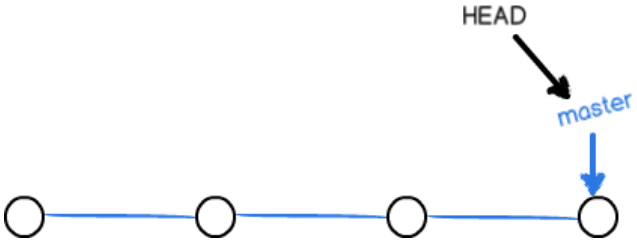
- 对当前新分支 dev 修改并提交，dev 指针前移一步



- dev 合并到 master，需将 master 指向 dev 的提交



- 删除 dev 分支，HEAD 指向 master 分支为当前分支



2、查看当前分支

$ git branch

3、创建、切换、合并、删除分支

创建并切换	$ git checkout -b 分支名 （已过时）
创建并切换	$ git switch -c 分支名
切换分支	$ git switch 分支名
创建分支	$ git branch 分支名
合并分支	$ git merge 分支名（fast forward 模式）
合并分支	$ git merge --no-ff -m "备注" 分支名（普通模式，非 fast forward 模式）

删除分支　　$ git branch -d 分支名
强删未合并分支　$ git branch -D 分支名

4、合并遇到冲突

查看冲突文件，发现：

```
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 of files.
<<<<<<< HEAD
Creating a new branch is quick & simple.
=======
Creating a new branch is quick AND simple.
>>>>>>> feature1
```

Git 用<<<<<<<，=======，>>>>>>>标记出不同分支的内容

5、BUG 分支

修复 bug 流程：

- 储存当前工作现场副本：　$ git status （清空当前工作区）
- 获取 master 最新版本：　$ git checkout master （恢复工作区至 master）
- 创建并切换 bug 分支：　$ git switch -c 分支名
- 修复 bug 文件，提交：　$ add、$ commit
- 切换至 master 分支：　$ git switch master
- 合并 bug 分支：　$ git merge --no-ff -m "备注" bug 分支名
- 切换至工作分支：　$ git switch 分支名
- 还原工作现场副本：

  查看所有工作现场　$ git stash list

  恢复工作现场{0}　$ git stash apply stash@{0} （恢复的 stash 不被删除）

  $ git stash pop （恢复的 stash 同时删除）

  手动删除工作现场{0}　$ git stash drop stash@{0}
- 若需要，可应用修改到当前分支：$ git cherry-pick 修改 bug 提交的 id

6、分支标签

查看所有标签：　$ git tag

查看标签信息：　$ git show 标签名

给当前分支当前版本，打标签：$ git tag 标签名

给当前分支指定版本，打标签：$ git tag 标签名 版本 id

打标签+备注：　$ git tag -a 标签名 -m "备注文字" 版本 id

删除本地标签：　$ git tag -d 标签名

推送单标签至远程：　$ git push origin 标签名

一次性推送全部：　$ git push origin --tags

删除标签：

先删除本地标签　……

再删除远程标签　$ git push origin :refs/tags/标签名