1. 常用

git init

初始化一个本地仓库

ssh-keygen -t rsa -C "注册github所用的邮箱"

创建一个SSH Key

git config –global user.name "YOUR NAME"

设置用户名

git config --global user.email "YOUR EMAIL ADDRESS"

设置邮箱

git config --global core.editor "editor -w"

设置git默认的编辑器，其中的“editor”替换成你自己的编辑器，如Vim、Emacs、、Atom、gedit、subl等。

git remote add origin [git@github.com:windofme1109/hello\_world.git](mailto:git@github.com:windofme1109/hello_world.git)

配置远端仓库，并设置标识为origin

git status

查看状态

git add filename

添加文档到暂存区

git commit -m ”提交说明”

提交，并附带一行提交说明

git commit

提交，启动文本编辑器，可以输入多行提交说明

格式是：

* 第一行：用一行文字简述提交的更改内容

● 第二行：空行

● 第三行以后：记述更改的原因和详细内容

保存，并关闭编辑器。记住，是关闭编辑器，不是关闭文件。是关闭编辑器，不是关闭文件。是关闭编辑器，不是关闭文件。

这样提交信息才会被保存。

git commit -am "提交说明"

添加并提交

git log

查看日志

git log --pretty=short

程序显示第一行简述信息

git log filename

显示与该文件相关的日志

git log dir

显示与该目录下的相关的日志

git log -p

查看提交所带来的改动，可以加上 -p参数，文件的前后差别就会显示在提交信息之后

git log --graph

以图形化的方式查看提交日志以及分支合并信息

git diff

查看工作树、暂存区、最新提交之间的差别

git diff HEAD

想查看工作树和最新提交的差别

git checkout -b feature-A

创建并切换到feature-A分支

git checkout master

切换到master分支

git branch

查看所有分支，并显示当前所在分支

git branch -a

同时显示本地仓库和远程仓库的分支信息

git merge --no-ff feature-A

将分支feature-A与master分支合并

git reset --hard 哈希值

根据回溯到指定时间节点的状态

git reflog

查看当前仓库的所有的操作日志

git commit --amend

修改上一条提交信息

git rebase -i HEAD~2

选定当前分支中包含HEAD（最新提交）在内的两个最新历史记录为对象，将两次提交记录压缩为一次（需要在编辑器中将pick命令修改为fixup）

git clone <url>

克隆远端仓库到本地

git checkout -b feature-D origin/feature-D

新建本地仓库分支feature-D,并指定分组的来源是origin/feature-D（远端仓库origin的分支feature-D）

git push

推送至远端仓库（在哪个分支下就推送哪个分支的内容）

git push origin master

本地仓库内容推送至远端仓库，分支为master

git push -u origin master

注意：-u参数可以在推送的同时，将origin仓库的master分支设置为本地仓库当前分支的 upstream（上游）。添加了这个参数，将来运行 git pull命令从远程仓库获取内容时，本地仓库的这个分支就可以直接从 origin 的 master 分支获取内容，省去了另外添加参数的麻烦。

**记住：第一次push到远端仓库时，一定要使用这条命令，否则后面再push的时候，会出问题。**

推送到远端仓库的master分支下

git push -u origin feature-D

推送到远端仓库的feature-D分支下

git pull origin master

从远端仓库分支master下载到本地仓库

git pull

从远端仓库下载到本地仓库（从哪个分支下，就从远端仓库的那个分支下载）

1. 别名

git config --global alias.br="branch"

创建/查看本地分支

git config --global alias.co="checkout"

切换分支

git config --global alias.cb="checkout -b"

创建并切换到新分支

git config --global alias.cm="commit -m"

提交

git config --global alias.st="status"

查看状态

git config --global alias.pullm="pull origin master"

拉取分支

git config --global alias.pushm="push origin master"

提交分支

git config --global alias.log="git log --oneline --graph --decorate --color=always"

单行、分颜色显示记录

git config --global alias.logg="git log --graph --all --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(bold white)— %an%C(reset)%C(bold yellow)%d%C(reset)' --abbrev-commit --date=relative"

复杂显示

1. 创建版本库

git clone <url>

克隆远程版本库

git init

初始化本地版本库

1. 提交和修改

git status

查看状态

git diff

查看变更内容

git add .

跟踪所有改动过的文件

git add <file>

跟踪指定的文件

git mv <old> <new>

文件改名

git rm <file>

删除文件

git rm --cached <file>

停止跟踪文件但不删除

git commit -m “commit message”

提交所有更新过的文件

git commit --amend

修改最后一次提交

1. 查看提交历史

git log

查看提交历史

git log -p <file>

查看指定文件的提交历史

git blame <file>

以列表方式查看指定文件的提交历史

1. 撤销

git reset --hard HEAD

撤消工作目录中所有未提交文件的修改内容

git reset --hard <version>

撤销到某个特定版本

git checkout HEAD <file>

撤消指定的未提交文件的修改内容

git checkout -- <file>

同上一个命令

git revert <commit>

撤消指定的提交

1. 分支与标签

git branch

显示所有本地分支

git checkout <branch/tag>

切换到指定分支或标签

git branch <new-branch>

创建新分支

git branch -d <branch>

删除本地分支

git tag

列出所有本地标签

git tag <tagname>

基于最新提交创建标签

git tag -a "v1.0" -m "一些说明"

-a指定标签名称，-m指定标签说明

git tag -d <tagname>

删除标签

git checkout dev

合并特定的commit到dev分支上

git cherry-pick 62ecb3

1. 合并与衍合

git merge <branch>

合并指定分支到当前分支

git merge --abort

取消当前合并，重建合并前状态

git merge dev -Xtheirs

以合并dev分支到当前分支，有冲突则以dev分支为准

git rebase <branch>

衍合指定分支到当前分支

1. 远程操作

git remote -v

查看远程版本库信息

git remote show <remote>

查看指定远程版本库信息

git remote add <remote> <url>

添加远程版本库

git remote remove <remote>

删除指定的远程版本库

git fetch <remote>

从远程库获取代码

git pull <remote> <branch>

下载代码及快速合并

git push <remote> <branch>

上传代码及快速合并

git push <remote> :<branch/tag-name>

删除远程分支或标签

git push –tags

上传所有标签