# Git配置

Git 提供了一个叫做 git config 的工具，专门用来配置或读取相应的工作环境变量。

这些环境变量，决定了 Git 在各个环节的具体工作方式和行为。这些变量可以存放在以下三个不同的地方：

（1）/etc/gitconfig 文件：系统中对所有用户都普遍适用的配置。若使用 git config 时用 --system 选项，读写的就是这个文件。

（2）~/.gitconfig 文件：用户目录下的配置文件只适用于该用户。若使用 git config 时用 --global 选项，读写的就是这个文件。

（3）当前项目的 Git 目录中的配置文件（也就是工作目录中的 .git/config 文件）：这里的配置仅仅针对当前项目有效。每一个级别的配置都会覆盖上层的相同配置，所以 .git/config 里的配置会覆盖 /etc/gitconfig 中的同名变量。

# 用户信息

git config --global user.name "runoob"

git config --global user.email test@runoob.com

# 差异分析工具

git config --global merge.tool vimdiff

# 查看配置信息

git config --list

# Git 工作流程



# Git 工作区、暂存区和版本库

1. 工作区：就是你在电脑里能看到的目录。
2. 暂存区：英文叫stage, 或index。一般存放在 ".git目录下" 下的index文件（.git/index）中，所以我们把暂存区有时也叫作索引（index）。
3. 版本库：工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

# Git 创建仓库

初始化（当前目录）：

git init

初始化（指定目录）：

git init newrepo

初始化后会出现一个名为.git的目录，所有的Git需要的数据和资源都放到这个目录当中。

如果当前目录下有几个文件想要纳入版本控制，需要先用git add命令告诉git开始对这些文件进行跟踪然后提交。

$ git add \*.c

$ git add README

$ git commit -m '初始化项目版本'

以上命令将目录下以 .c 结尾及 README 文件提交到仓库中。

克隆仓库的命令格式为：

git clone <repo>

如果我们需要克隆到指定的目录，可以使用以下命令格式：

git clone <repo> <directory>

其中repo指的是Git仓库。

git clone 时，可以所用不同的协议，包括 ssh, git, https 等，其中最常用的是 ssh，因为速度较快，还可以配置公钥免输入密码。各种写法如下：

git clone git@github.com:fsliurujie/test.git --SSH协议

git clone git://github.com/fsliurujie/test.git --GIT协议

git clone https://github.com/fsliurujie/test.git --HTTPS协议

git add（git add 命令可将该文件添加到缓存）

新项目中，添加所有文件很普遍，我们可以使用 git add . 命令来添加当前项目的所有文件。

git status 显示你上次提交更新后的更改或者写入缓存的改动， 而 git diff 一行一行地显示这些改动具体是啥。

git commit

使用 git add 命令将想要快照的内容写入缓存区， 而执行 git commit 将缓存区内容添加到仓库中。

git reset HEAD

git reset HEAD 命令用于取消已缓存的内容。

git commit、git push、git pull、 git fetch、git merge 的含义与区别

git commit：是将本地修改过的文件提交到本地库中；

git push：是将本地库中的最新信息发送给远程库；

git pull：是从远程获取最新版本到本地，并自动merge；

git fetch：是从远程获取最新版本到本地，不会自动merge；

git merge：是用于从指定的commit(s)合并到当前分支，用来合并两个分支；

git pull 相当于 git fetch + git merge。

创建分支命令：

git branch (branchname)

切换分支命令:

git checkout (branchname)

列出分支基本命令：

git branch

Git 查看提交历史

$ git log