

目录

Git 基本使用	2
1. Git Configurations	2
2. Creating New Repos	2
2.1 git init	2
2.2 git clone.....	3
2.3 git status	3
3. Review a Repo's History	4
3.1 git log	4
3.2 git show	6
4. Add Commits To A Repo	7

Git 基本使用

1. Git Configurations

基本配置命令：

```
git config --global user.name "xxxxx"
```

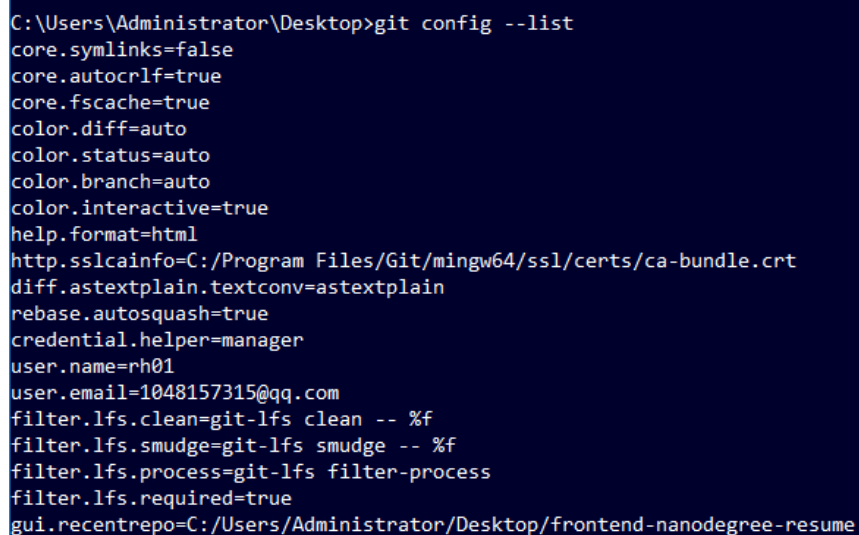
```
git config --global user.email xxx@xx.com
```

使用上述命令时，`user.name`和`user.email`均为 GitHub 注册的用户名和邮箱。

查看 Git 的配置内容：

```
git config --list
```

一般情况下打印的内容如下图所示：



```
C:\Users\Administrator\Desktop>git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
credential.helper=manager
user.name=rh01
user.email=1048157315@qq.com
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
gui.recentrepo=C:/Users/Administrator/Desktop/frontend-nanodegree-resume
```

图 1 Git 配置内容

2. Creating New Repos

主要学习的命令为：`git init`，`git clone`，`git status`

`git init`为在本地计算机创建一个新的仓库。

`git clone`为从远程或者本地 Git 服务器上 copy 一个 repo 到本地计算机。

`git status`为检查 repo 的状态信息，比如那些文件发生改变，那些需要 commit 等等。

2.1 git init

运行 `git init` 命令后，`Git`将跟踪当前文件夹下的所有内容，包括文件和目录。

所有这些文件信息都存储在名为`.git`的目录中（注意在开头`.`，这意味着它将是 Mac / Linux 上的隐藏目录）。这个`.git`目录是 `Git` 记录“repo”跟踪所有变化和提交的地方！

警告 - 不要直接编辑`.git`目录中的任何文件。这是版本库（repo）的核心。如果更改文件名和/或文件内容，git 可能会丢失在 repo 中保留的文件，并且可能会失去很多工作！可以查看这些文件，但不要编辑或删除它们。

以下是`.git`目录中每个文件夹的简要描述：

config file - 存储所有特定的项目的配置。

From the [Git Book](#):

Git looks for configuration values in the configuration file in the Git directory (`.git/config`) of whatever repository you're currently using. These values are specific to that single repository.

Git 在您当前使用的任何存储库的 Git 目录（`.git / config`）中的配置文件中查找配置值。这些值特定于该单个存储库。

例如，假设您在 Git 的全局配置（global）使用您的个人电子邮件地址。但是如果您希望将工作电子邮件用于特定项目而不是您的个人电子邮件，那么就应该修改该文件中的邮件设置。

description file - 此文件仅由 GitWeb 程序使用，可以忽略它。

hooks directory - 我们可以在该目录中放置客户端或服务器端脚本，这些脚本用来挂钩 Git 不同的生命周期事件。

info directory - 包含全局排除文件 objects 目录 - 此目录将存储我们所做的所有提交。

refs directory - 此目录包含提交的指针（基本上是“分支”和“标签”）。

2.2 git clone

目的：在本地构建一个副本

usage:

```
git clone https://github.com/xxx/xxxx.git
```

```
git clone git@github.com:xxx/xxx.git
```

2.3 git status

`git status` 会告诉我们 Git 正在考虑什么，以及 Git 看到的版本库的状态。当你是第一次使用它，你应该每一步都要使用 `git status` 命令！在执行完其他命令后，都要很习惯性地运行该命令。这将帮助您了解 Git 的工作原理，并帮助您对文件/版本库的状态做出正确的判断。

Git Status Explanation

比如下列一段输出：

```
On branch master  
  
Your branch is up-to-date with 'origin/master'.  
  
nothing to commit, working directory clean
```

The output tells us two things:

1. `On branch master` – this tells us that Git is on the master branch. You've got a description of a branch on your terms sheet so this is the "master" branch (which is the default branch).
2. `Your branch is up-to-date with 'origin/master'.` – Because git clone was used to copy this repository from another computer, this is telling us if our project is in sync with the one we copied from. We won't be dealing with the project on the other computer, so this line can be ignored.
3. `nothing to commit, working directory clean` – this is saying that there are no pending changes.

Explanation Of Git Status In A New Repo

```
$ git status  
  
On branch master  
  
Initial commit  
  
nothing to commit (create/copy files and use "git add" to track)
```

3. Review a Repo's History

学习一下两个命令：

`git log`, `git show`

3.1 git log

`git log` 命令用于显示版本库的所有提交。

默认情况下，此命令显示：SHA、作者、日期和消息。如下图所示：

```
commit 94d305ee8f81ccaf866a76222f10357940beb5de
Author: 申恒恒(Shine) <1048157315@qq.com>
Date:   Mon May 8 14:41:30 2017 +0800

    Add files via upload

commit 589415654d071a7d5fe7ff903e18743f4dac45eb
Author: 申恒恒(Shine) <1048157315@qq.com>
Date:   Mon May 8 14:37:20 2017 +0800

    Delete db.sqlite3

commit dabcb3b169e12b5bda1b0c51e96877246048b5ab6
Author: rh01 <1048157315@qq.com>
Date:   Fri Mar 31 21:46:30 2017 +0800

    add publish file

commit edd7309147531b5289a37e38eb4c3ed261600947
Author: rh01 <1048157315@qq.com>
Date:   Fri Mar 31 20:27:43 2017 +0800

    first commit
```

图 2 git log 输出信息

the SHA - git log will display the complete SHA for every single commit. Each SHA is unique, so we don't really need to see the *entire* SHA. We could get by perfectly fine with knowing just the first 6-8 characters. Wouldn't it be great if we could save some space and show just the first 5 or so characters of the SHA?

the author - the git log output displays the commit author for *every single commit*! It could be different for other repositories that have multiple people collaborating together, but for this one, there's only one person making all of the commits, so the commit author will be identical for all of them. Do we need to see the author for each one? What if we wanted to hide that information?

the date - By default, git log will display the date for each commit. But do we really care about the commit's date? Knowing the date might be important occasionally, but typically knowing the date isn't vitally important and can be ignored in a lot of cases. Is there a way we could hide that to save space?

the commit message - this is one of the most important parts of a commit message...we usually always want to see this

学习一个新的命令：

```
$ git log --oneline
```

`git log --oneline` 这个命令：

1. 列出每行一个提交
2. 显示提交的 SHA 的前 7 个字符
3. 显示提交的消息

```
$ git log --stat
```

`git log --stat` 这个命令：

1. 显示已修改的文件
2. 显示已添加/删除的行数
3. 显示已添加/删除的行总数和已修改文件数的总数目的摘要行

```
$ git log -p
```

The git log command has a flag that can be used to display the actual changes made to a file. The flag is --patch which can be shortened to just -p:

```
$ git log -p
```

`git log -p` 命令用于显示对文件所做的实际更改。

1. 显示已修改的文件
2. 显示已添加/删除的行的位置
3. 显示所做的实际更改

3.2 git show

```
$ git show
```

`git show` 它只会显示最近的提交。通常，提供 SHA 作为参数：

```
$ git show fdf5493
```

git show 命令只显示一个提交。

git show 命令的输出与 git log -p 命令输出完全相同。所以默认情况下，git show 显示：

1. 提交

2. 作者日期
3. 提交消息
4. 补丁信息

4. Add Commits To A Repo

介绍三个命令：`git add`，`git commit`，`git diff`

4.1 git add

`git add` 命令用于将文件从工作目录添加到分段索引。

```
$ git add <file1> <file2> ... <fileN>
```

这个命令：

1. 采用空格分隔的文件名列表
2. 或者，`.` 可以使用代替文件列表来告诉 Git 添加当前目录（和所有嵌套文件）

4.2 git commit

一般情况下，只输入 `git commit` 指令，将会弹出没有设置编辑器路径等类似的错误信息。因此需要在第 1 节中配置时添加上默认的编辑器路径或者链接。

```
$ git config --global core.editor <your-editor's-config-went-here>
```

接下来，如果要提交时，就会弹出编辑器窗口，在里面添加提交信息，保存并关闭，即可提交上去。

```
git commit -m "xxxx"
```

这条命令绕过了编辑器。如果您正在撰写的提交消息很短，并且您不想等待代码编辑器打开再输入，可以使用 `-m` 标志直接在命令行中传递消息。

比如：

```
$ git commit -m "Initial commit"
```

输入完成后，可以通过 `git log` 命令查看提交信息。