Git，世界上最先进的分布式版本控制管理工具。

我们已经在电脑上安装了Git。安装过程省略，下面开始介绍Git的基本用法。

注意，我们使用的是git bash这个终端工具进行git命令操作。不使用windows自带的命令提示符。

1. 配置Git
2. 设置姓名和邮箱地址

git config --global user.name "github用户名"

git config --global user.email "注册github所用的邮箱"

用自己的github用户名和注册邮箱其替换。

这个命令，会在“~/.gitconfig”中以如下形式输出设置文件。这个文件的在C:\Users\Administrator.USER-20181123NM目录下。

我们可以打开看一下：

[filter "lfs"]

required = true

clean = git-lfs clean -- %f

smudge = git-lfs smudge -- %f

process = git-lfs filter-process

[user]

name = windofme1109

email = wuzengyan1314@163.com

[core]

editor = \"C:\\Users\\Administrator.USER-20181123NM\\AppData\\Local\\atom\\app-1.36.1\\atom.exe\" --wait

[color]

ui = auto

想更改这些信息时，可以直接编辑这个设置文件。这里设置的姓名和邮箱地址会用在 Git 的提交日志中。

1. 设置SSH Key

GitHub 上连接已有仓库时的认证，是通过使用了 SSH 的公开密钥认证方式进行的。现在让我们来创建公开密钥认证所需的 SSH Key，并将其添加至 GitHub。

1. 输入：

ssh-keygen -t rsa -C "注册github所用的邮箱"

1. 然后会出现：

Generating public/private rsa key pair.

Enter file in which to save the key (/c/Users/Administrator.USER-20181123NM/.ssh/id\_rsa):

此时按下回车键，会将SSH Key保存在C:/Users/Administrator.USER-20181123NM/.ssh这个目录下，同时提示：

Created directory '/c/Users/Administrator.USER-20181123NM/.ssh'.

1. 紧接着会提示：

Enter passphrase (empty for no passphrase):

这一步告诉我们要设置一个密码，我们可以不用设置，直接按回车键即可。

Enter same passphrase again:

同样，让我们确认密码，由于我们没有输入，所在也是直接按下回车键。

1. 完成上面三步以后，会出现一下结果：

Your identification has been saved in /c/Users/Administrator.USER-20181123NM/.ssh/id\_rsa.

Your public key has been saved in /c/Users/Administrator.USER-20181123NM/.ssh/id\_rsa.pub.

The key fingerprint is:

**fingerprint值**

The key's randomart image is:

+---[RSA 3072]----+

|.o ..+. |

|B o. +. . |

|oX.. oE. . |

|+o.+..o. o . |

| + + +.S.o+ . |

| o =..+.B |

| o + .+.+ |

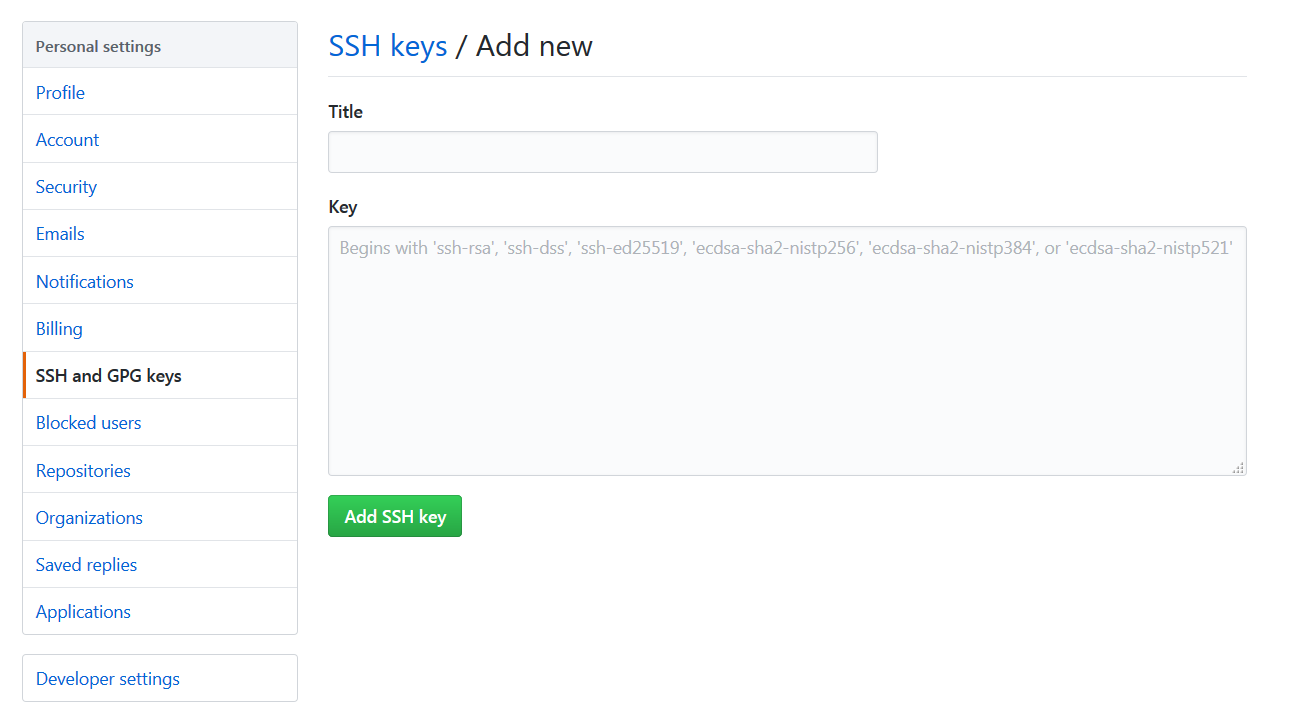
| . o o o... |

| +o... o. |

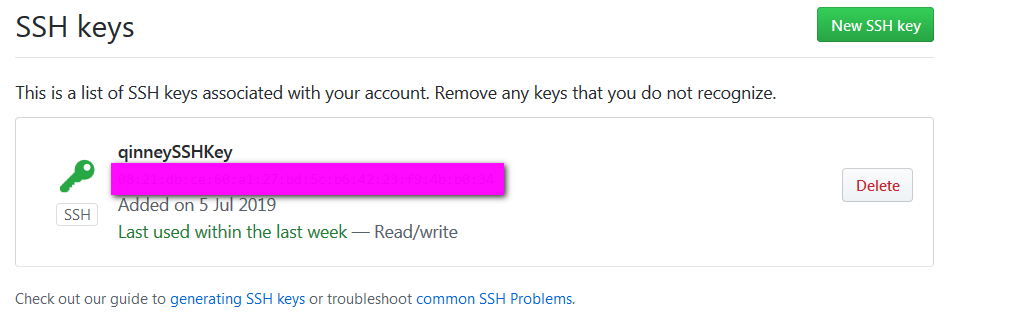
+----[SHA256]-----+

1. 添加公开秘钥。在 GitHub 中添加公开密钥，今后就可以用私有密钥进行认证了。

点击右上角的账户设定按钮（Account Settings），选择 SSH Keys 菜单。点击 Add SSH Key 之后，会出现如下图的输入框。在 Title 中输入适当的密钥名称。Key 部分请粘贴 id\_rsa.pub 文件里的内容。这个文件保存在C:/Users/Administrator.USER-20181123NM/.ssh路径下。



然后点击Add SSH key这个按钮。就完成了秘钥的添加。如下图所示：



添加成功之后，创建账户时所用的邮箱会接到一封提示“公共密钥添加完成”的邮件。

1. 使用手中的私人密钥与 GitHub 进行认证和通信。在git bash中输入：

ssh -T git@github.com

提示：

The authenticity of host 'github.com (13.250.177.223)' can't be established.

RSA key fingerprint is fingerprint值

Are you sure you want to continue connecting (yes/no/[fingerprint])?

输入：

yes

出现如下内容即为成功：

Hi windofme1109! You've successfully authenticated, but GitHub does not provide shell access.

二、Git实际操作

首先，我们要在本地建立一个仓库。路径是：E:\GitForCoding\hello\_world。hello\_world是仓库名称。我们在GitHub已经建立了同名的仓库。方便后面的同步。

在git bash中，输入：

cd E:\GitForCoding

mkdir hello\_world

然后，输入：

git init

这个条命令的作用是初始化仓库。执行了 git init命令的目录下就会生成 .git 目录。这个 .git 目录里存储着管理当前目录内容所需的仓库数据。

当然，这个目录不可见。我们可以输入：

ls -al

进行查看。显示结果如下：

drwxr-xr-x 1 Administrator 197121 0 七月 5 11:49 ./

drwxr-xr-x 1 Administrator 197121 0 七月 5 11:49 ../

drwxr-xr-x 1 Administrator 197121 0 七月 5 11:49 .git/

输入：

git status

这条命令的作用是用于显示 Git 仓库的状态。此时会显示：

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

第一条表示我们处在master分支下，第二条表示我们还没有进行提交（commit）。提交（Commit），是指“记录工作树中所有文件的当前状态”。第三条表示没有内容可以提交。就是说当前我们建立的这个仓库中还没有记录任何文件的任何状态。

我们在仓库中新建一个文件，输入：

touch readme.md

这条命令的意思是新建一个名字为readme的md文件。

然后输入：

git status

显示的内容是：

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

readme.md

nothing added to commit but untracked files present (use "git add" to track)

在Untracked files中显示了readme.md文件。表示还没对这个文件进行追踪。

如果只是用 Git 仓库的工作树创建了文件，那么该文件并不会被记入 Git 仓库的版本管理对象当中。因此我们用 git status命令查看README.md 文件时，它会显示在 Untracked files 里。

要想让文件成为 Git 仓库的管理对象，就需要用 git add命令将其加入暂存区（Stage 或者 Index）中。暂存区是提交之前的一个临时区域。

在git bash中，输入：

git add readme.md

这条命令的作用是将readme.md这个文件添加到暂存区中。

此时我们查看一下状态：

git status

提示：

On branch master

Your branch is up to date with 'origin/master'.

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: readme.md

我的理解是，暂存区是本地文件与git仓库之间的一个中转区，我们通过 git add命令，将文件添加到这个暂存区，暂存区保存的都是等待提交到仓库的文件（或者是文件的状态）。

接下来，我们要把暂存区中文件保存到仓库的历史记录中，这就是所谓的提交（commit）过程。

输入：

git commit -m"add hello world"

git commit表示提交。-m后面的"add hello world"表示对这次提交的一个说明。

推荐每次提交加上-m这个参数。

-m参数一般用于添加一行说明信息，如果我们想添加多行信息，就不使用-m参数，直接使用git commit命令，然后就会启动编辑器（安装时指定的），并显示：

$ hint: Waiting for your editor to close the file...

编辑器里面显示的内容是：

# Please enter the commit message for your changes. Lines starting

# with '#' will be ignored, and an empty message aborts the commit.

#

# On branch master

# Your branch is ahead of 'origin/master' by 1 commit.

# (use "git push" to publish your local commits)

#

# Changes to be committed:

# modified: readme.md

#

在编辑器中记述提交信息的格式如下。

● 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

只要按照上面的格式输入，今后便可以通过确认日志的命令或工具

看到这些记录。

在以 #（井号）标为注释的 Changes to be committed（要提交的更改）栏中，可以查看本次提交中包含的文件。将提交信息按格式记述完毕后，请保存并关闭编辑器，以 #（井号）标为注释的行不必删除。随后，刚才记述的提交信息就会被提交。

如果在编辑器启动后想中止提交，请将提交信息留空并直接关闭编辑器，随后提交就会被中止。

此时使用git status查看状态，输入：

git status

提示：

On branch master

nothing to commit, working tree clean

如果我们想查看一下提交日志，包括可以查看什么人在什么时候进行了提交或合并，以及操作前后有怎样的差别，我们可以使用git log命令。

输入：

git log

提示：

commit 5c1c086a2fe61281cbffbc361f6ae622f8a410de (HEAD -> master)

Author: windofme1109 <wuzengyan1314@163.com>

Date: Fri Jul 5 12:05:31 2019 +0800

create new file

commit 栏旁边显示的“5c1c0……”是指向这个提交的哈希值。Git 的其他命令中，在指向提交时会用到这个哈希值。

Author 栏中显示我们给 Git 设置的用户名和邮箱地址。Date 栏中显示提交执行的日期和时间。再往下就是该提交的提交信息。

git log命令的参数：

1. 只显示提交信息的第一行

如果只想让程序显示第一行简述信息，可以在 git log命令后加上 --pretty=short。这样一来开发人员就能够更轻松地把握多个提交。

输入：

git log --pretty=short

提示：

commit 5c1c086a2fe61281cbffbc361f6ae622f8a410de

Author: windofme1109 <wuzengyan1314@163.com>

create new file

1. 只显示指定目录、文件的日志

只要在 git log命令后加上目录名，便会只显示该目录下的日志。如果加的是文件名，就会只显示与该文件相关的日志。

输入：

git log readme.md

提示：

commit 5c1c086a2fe61281cbffbc361f6ae622f8a410de

Author: windofme1109 <wuzengyan1314@163.com>

Date: Fri Jul 5 12:05:31 2019 +0800

create new file

1. 显示文件的改动

如果想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。我在readme.md加了一句话：hello world，此时可以使用git log -p命令查看。

git log -p

提示：

commit 757378fe76e80337679e448374796d1c75fec3fd

Author: windofme1109 <wuzengyan1314@163.com>

Date: Fri Jul 5 12:12:30 2019 +0800

add hello world

**diff --git a/readme.md b/readme.md**

**index e69de29..95d09f2 100644**

**--- a/readme.md**

**+++ b/readme.md**

@@ -0,0 +1 @@

+hello world

\ No newline at end of file

commit 5c1c086a2fe61281cbffbc361f6ae622f8a410de

Author: windofme1109 <wuzengyan1314@163.com>

Date: Fri Jul 5 12:05:31 2019 +0800

create new file

**diff --git a/readme.md b/readme.md**

**new file mode 100644**

**index 0000000..e69de29**

执行下面的命令，就可以只查看 README.md 文件的提交日志以及提交前后的差别。

git log -p readme.md

git diff命令可以查看工作树、暂存区、最新提交之间的差别。

我们刚刚向readme.md添加了内容：eeeeeeeeeeeeeeeeeeee，现在执行 git diff命令，查看当前工作树与暂存区的差别。

输入：

git diff

提示：

**diff --git a/readme.md b/readme.md**

**index aee217d..2ff89dd 100644**

**--- a/readme.md**

**+++ b/readme.md**

@@ -8,4 +8,6 @@ hello world

where amazing happens

-aaaaaaaaaaaaaaaaaaaa

\ No newline at end of file

+aaaaaaaaaaaaaaaaaaaa

+

+eeeeeeeeeeeeeeeeeeee

\ No newline at end of file

由于我们尚未用 git add命令向暂存区添加任何东西，所以程序只会显示工作树与最新提交状态之间的差别。

+”号标出的是新添加的行，被删除的行则用“-”号标出。

如果我们想查看工作树和最新提交的差别，可以使用git diff HEAD命令。

输入：

git diff HEAD

提示：

**diff --git a/readme.md b/readme.md**

**index 5032852..2ff89dd 100644**

**--- a/readme.md**

**+++ b/readme.md**

@@ -5,4 +5,9 @@ hello world

22222222222222

-where amazing happens

\ No newline at end of file

+where amazing happens

+

+

+aaaaaaaaaaaaaaaaaaaa

+

+eeeeeeeeeeeeeeeeeeee

\ No newline at end of file

使用这个命令可以让我们在真正的提交之前，检查一次最新的提交内容与上一次提交的差别，等确认完毕后再进行提交。这里的 HEAD 是指向当前分支中最新一次提交的指针。