

# Git

Git属于分散性版本管理系统，是为版本管理而设计的软件。

Linux--Linus在05年开发了git原型程序

SVN，CVS集中式的版本控制系统，集中存放中央服务器的

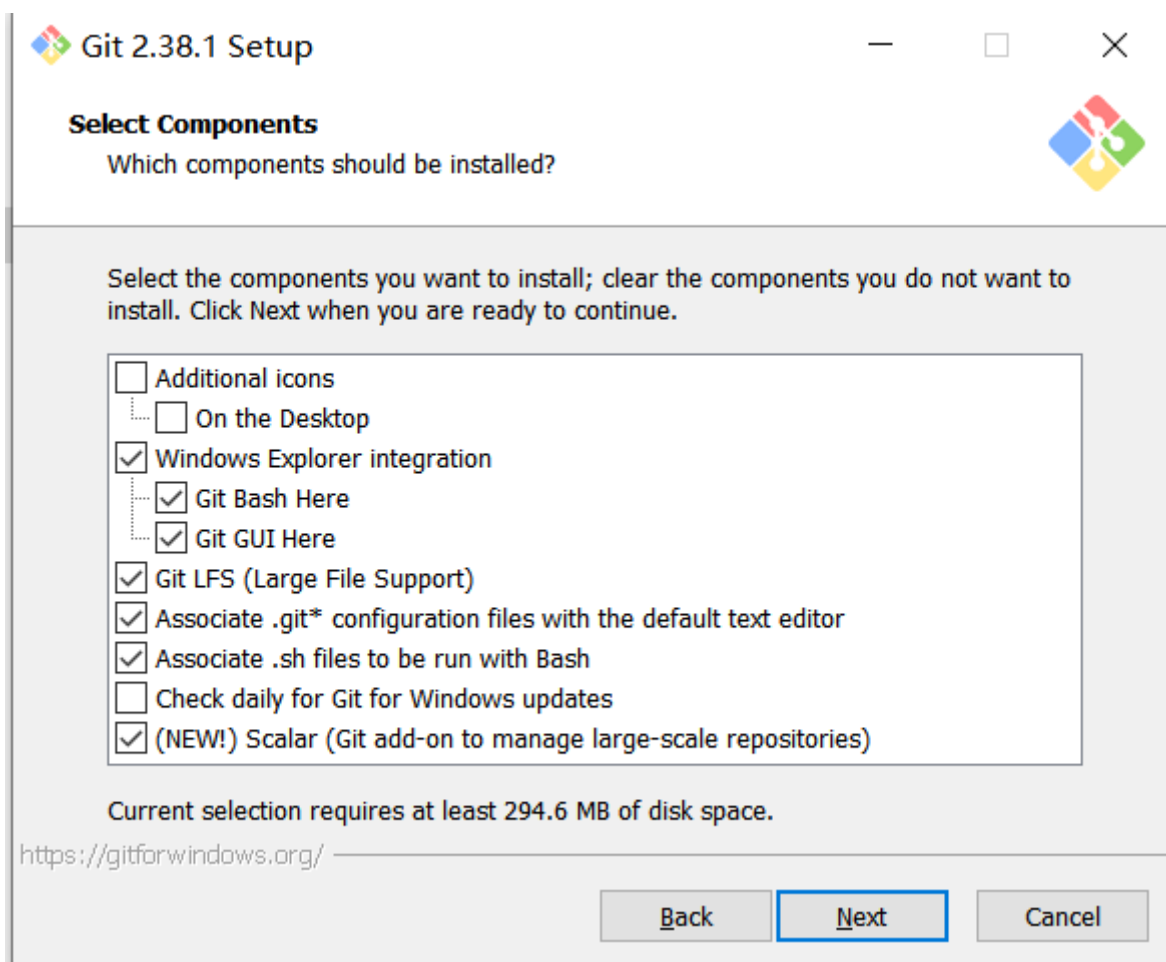
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Git 是一个免费的开源分布式版本控制系统，旨在快速高效地处理从小型到超大型项目的所有内容。Git 易于学习，占用空间很小，性能快如闪电。它超越了Subversion，CVS，Perforce和ClearCase等SCM工具，具有廉价的本地分支，方便的暂存区域和多个工作流程等功能。

## 安装

组件的选择，默认就可以



注册账号

[Gitee - 基于 Git 的代码托管和研发协作平台](#)

<https://github.com/>

# 设置

## 设置姓名和邮箱

```
git config --global user.name "你的名字"  
git config --global user.email "你的邮箱"
```

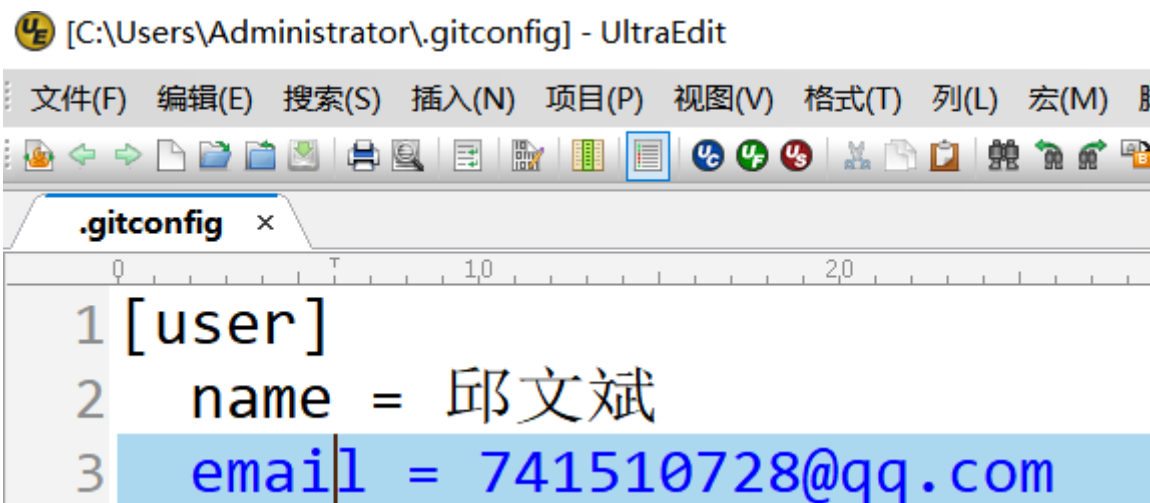
git config命令指定与git相关的配置

config配置级别：

- system：系统
- global：用户
- local：当前仓库

范围system>global>local

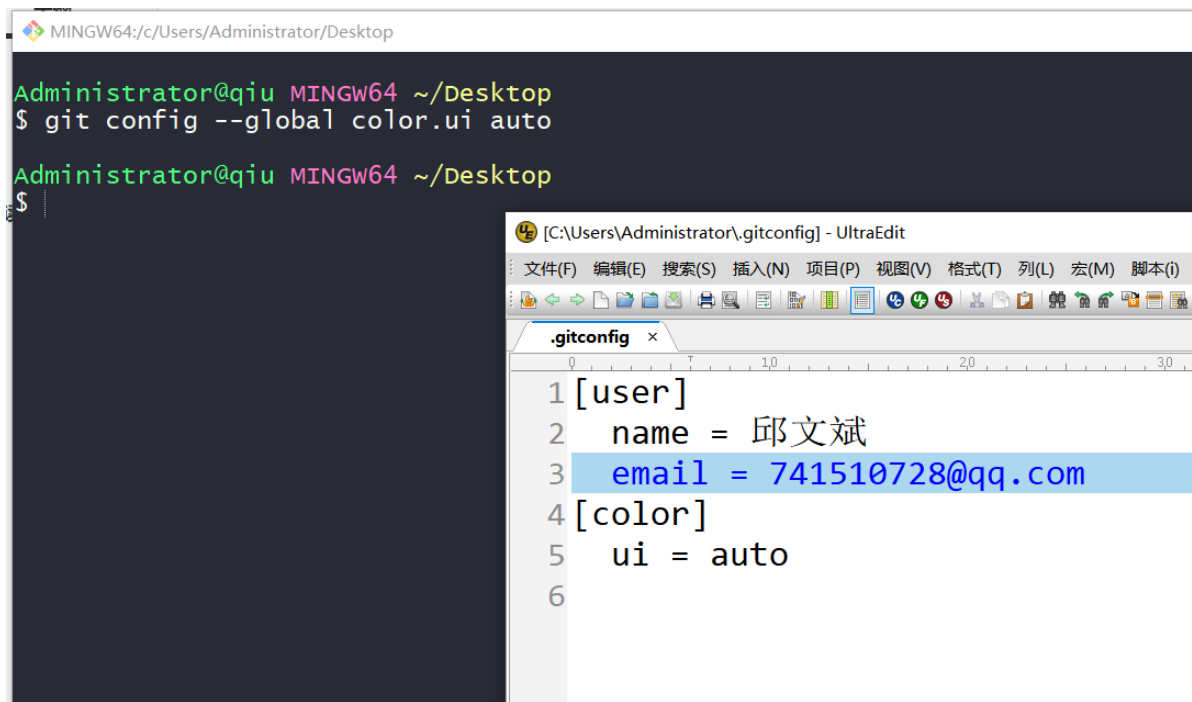
```
# 查看git配置  
git config --list
```



## 提高命令输出的可读性

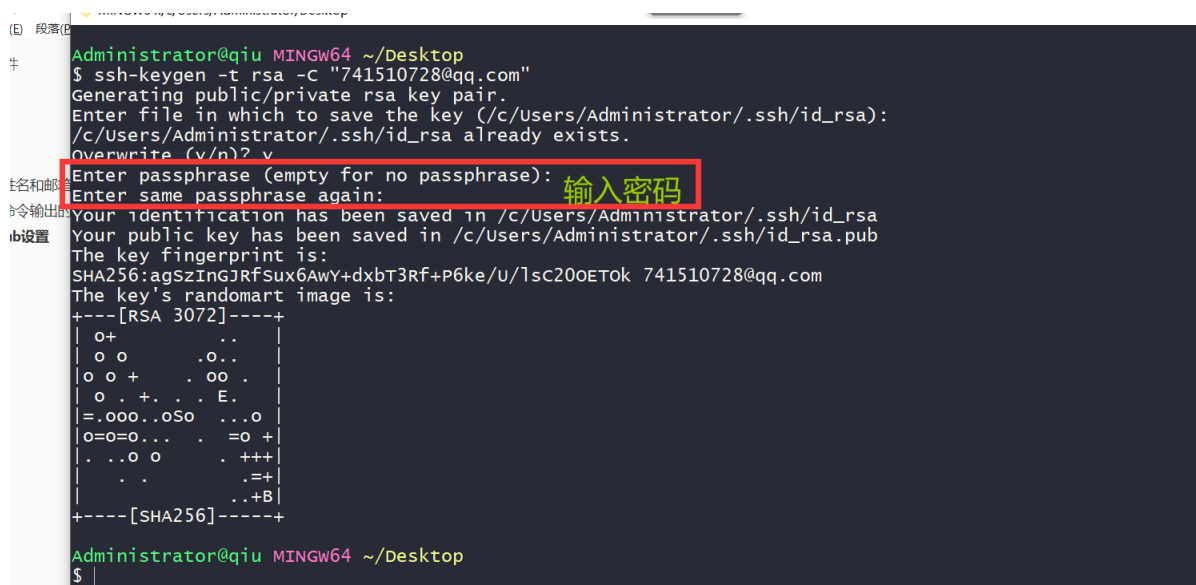
设置color.ui为auto，让输出有更高的可读性

```
git config --global color.ui auto
```



## GitHub设置

```
# 1、打开git bash
# 2、生成sshkey
ssh-keygen -t rsa -C "741510728@qq.com"
```



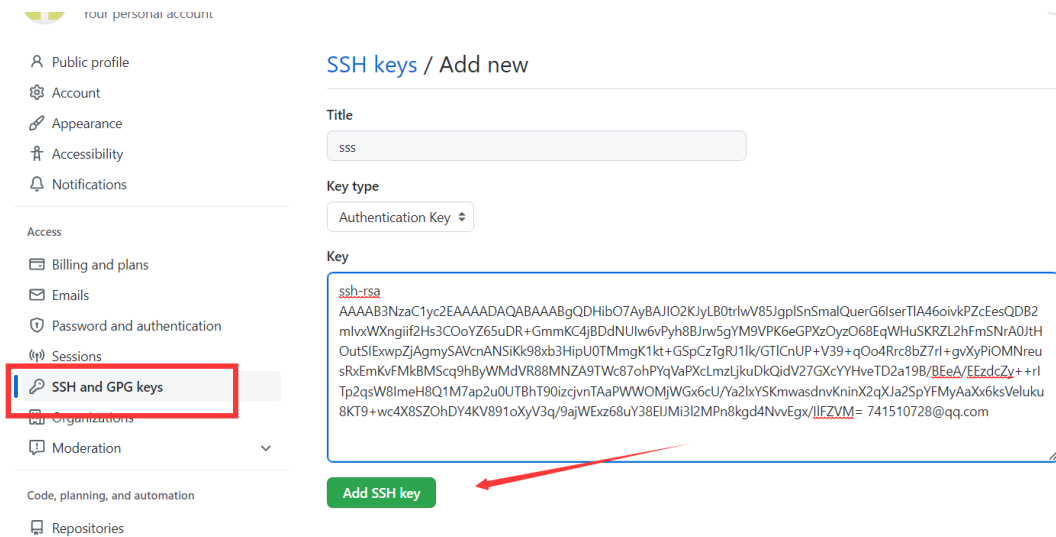
然后打开路径

```
C:\Users\Administrator\.ssh
```

Acer (C:) > 用户 > Administrator > .ssh			
名称	修改日期	类型	大小
<input type="checkbox"/> id_rsa	2022/11/26 10:17	文件	3 KB
<input checked="" type="checkbox"/> id_rsa.pub	2022/11/26 10:17	PUB 文件	1 KB

分别是私钥和公钥

我们用记事本打开公钥id\_rsa.pub，然后全选复制所有内容



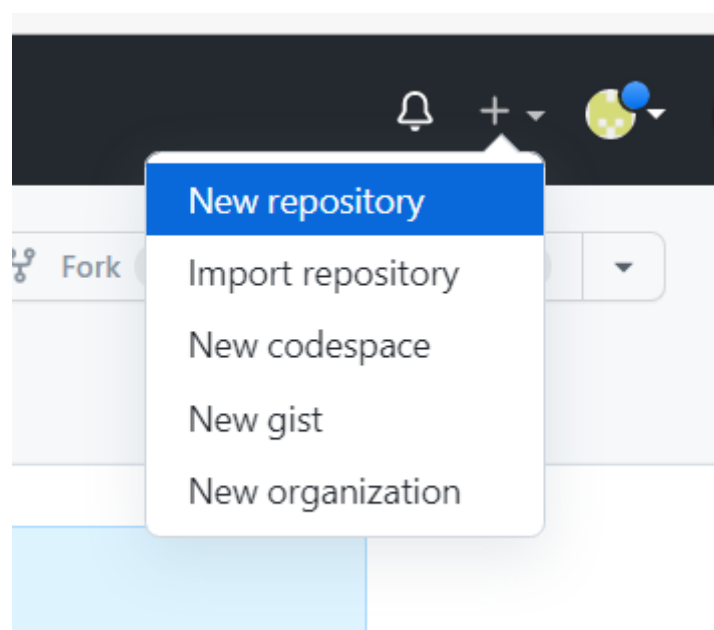
完成设置后，邮箱会收到提示邮件，这个时候，我们就可以使用私钥和GitHub进行认证和通信了。

执行命令验证

```
ssh -T git@github.com
```

```
Administrator@qiu MINGW64 ~/Desktop
$ ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3vVVV6TuJJhpZisF/zLDA0zPMSvHdkr4UvCoQu.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enter passphrase for key '/c/Users/Administrator/.ssh/id_rsa':
Hi waqwb! You've successfully authenticated, but GitHub does not provide shell access.
```

## 创建仓库



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*

waqwb

Repository name \*

Hello-World

Great repository names are short and memorable. Need inspiration? How about [probable-giggle?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None

You are creating a public repository in your personal account.

Create repository

```
cd /f/tarena/授课/枣庄/VIP创新课程（行业）/code
# 把github上的仓库，克隆到本地
git clone https://github.com/waqwb/HelloWorld.git
```

Doc (F:) > tarena > 授课 > 枣庄 > VIP创新课程（行业） > code > HelloWorld >

名称	修改日期	类型	大小
.git	2022/11/26 10:43	文件夹	

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/code
$ cd HelloWorld/

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/code/HelloWorld
(main)
$ pwd
/f/tarena/授课/枣庄/VIP创新课程（行业）/code/HelloWorld



Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/code/HelloWorld
(main)
$ echo "# HelloWorld" >> README.md
```

布局

当前视图

显示/隐藏

此电脑 > Doc (F:) > tarena > 授课 > 枣庄 > VIP创新课程 (行业) > code > HelloWorld

<div><input type="checkbox"/></div> 名称	修改日期	类型	大小
<div><input type="checkbox"/></div>  .git	2022/11/26 10:43	文件夹	
<div></div> README.md	2022/11/26 10:51	MD 文件	11 B

```
git add README.md
git commit -m "first commit"
```

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Hello.java
```

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程
d (main)
$ git commit -m "second commit"
[main 7e8b8cf] second commit
1 file changed, 5 insertions(+)
create mode 100644 Hello.java

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程
d (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

此时只是提交操作，并不会把我们目录中的内容上传至网站

```
git push -u origin main
```

第一次push的时候，会在git bash中有一个登录github的验证，大家验证即可

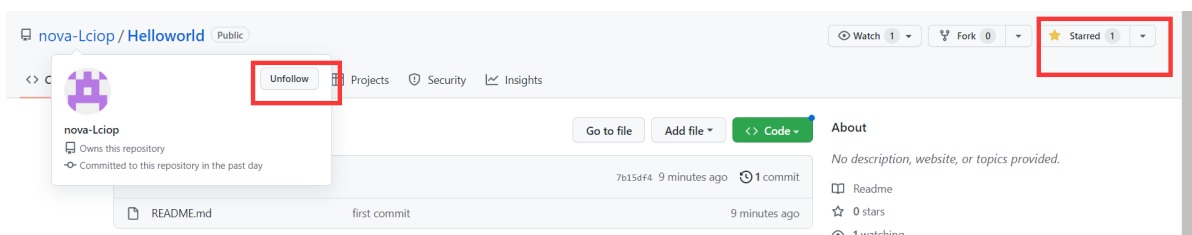
```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/code/HelloWorld (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 376 bytes | 376.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/waqwb/HelloWorld.git
   39246d3..7e8b8cf  main -> main
branch 'main' set up to track 'origin/main'.

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/code/HelloWorld (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

出现以下内容证明成功了，在浏览器中刷新GitHub中的仓库

## 番外

关注作者和star仓库



```
cd HelloWorld
echo "# HelloWorld" >> README.md
# 当仓库没有HelloWorld仓库的时候 .git
git init
git add README.md
git commit -m "first commit"
git branch -M main
# 指定我们要上传到的仓库的地址
git remote add origin https://github.com/waqwb/HelloWorld.git
git push -u origin main
```

## 练习

新建一个hello.java,写一段代码添加到文件中

上传至GitHub中

## Git基本操作

## 初始化仓库

```
git init
```

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note
$ git init
Initialized empty Git repository in F:/tarena/授课/枣庄/VIP创新课程（行业）/note/.git/
```

如果初始化成功，执行git init命令后会在目录下生成一个.git目录，目录中存储着管理当前目录内容所需的仓库数据。

电脑 > Doc (F:) > tarena > 授课 > 枣庄 > VIP创新课程（行业） > note

<input type="checkbox"/> 名称	修改日期	类型	大小
 .git	2022/11/26 14:04	文件夹	
 Git.assets	2022/11/26 14:05	文件夹	
 Git.md	2022/11/26 14:06	MD 文件	5 KB
 Git.pdf	2022/11/26 11:36	WPS PDF 文档	562 KB

这个目录的内容称之为“附属于该仓库的工作树”

文件的编辑等操作在工作树中进行，然后记录到仓库中，以此来管理文件的历史快照。

如果想将文件恢复到原先的状态，可以从仓库中调取之前的快照，在工作树中打开。

开发者可以通过这种方式获取以往的文件。

## 查看仓库的状态

```
git status
```

显示git仓库的状态。

工作树和仓库被操作的过程中，状态会不断地发生变化，可以使用这个命令来查看当前的状态。

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Git.assets/
    Git.md
    Git.pdf

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

提交commit，记录工作树中所有文件的当前状态

## 向暂存区中添加文件

想让文件称为git仓库的管理对象，那么我们可以用git add命令将其加入暂存区，暂存区是一个临时区域。

```
git add .
```



```

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git add .

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git status
On branch master

No commits yet

changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Git.assets/image-20221126083030535.png
    new file:   Git.assets/image-20221126092855114.png
    new file:   Git.assets/image-20221126092904395.png
    new file:   Git.assets/image-20221126101755047.png
    new file:   Git.assets/image-20221126102007932.png
    new file:   Git.assets/image-20221126102109323.png
    new file:   Git.assets/image-20221126102454497.png
    new file:   Git.assets/image-20221126103422353.png
    new file:   Git.assets/image-20221126103510905.png
    new file:   Git.assets/image-20221126105352918.png
    new file:   Git.assets/image-20221126105425085.png
    new file:   Git.assets/image-20221126111421660.png
    new file:   Git.assets/image-20221126112155748.png
    new file:   Git.assets/image-20221126113329134.png
    new file:   Git.assets/image-20221126113408179.png
    new file:   Git.assets/image-20221126113504543.png
    new file:   Git.assets/image-20221126140521073.png
    new file:   Git.assets/image-20221126140621265.png
    new file:   Git.assets/image-20221126141042324.png
    new file:   Git.md
    new file:   Git.pdf

```

如果想把文件移出暂存区可以使用以下命令

```
git rm --cached ...
```

## 保存仓库的历史记录

```
git commit -m "FirstCommit"
```

可以将暂存区中的文件实际保存到仓库的历史记录中。通过这些记录，我们可以在工作数中复原文件。

其中 `-m "FirstCommit"` 称作提交信息，是对提交的概述

```

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git commit -m "firstcommit"
[master (root-commit) cf517d3] firstcommit
22 files changed, 228 insertions(+)
create mode 100644 Git.assets/image-20221126083030535.png
create mode 100644 Git.assets/image-20221126092855114.png
create mode 100644 Git.assets/image-20221126092904395.png
create mode 100644 Git.assets/image-20221126101755047.png
create mode 100644 Git.assets/image-20221126102007932.png
create mode 100644 Git.assets/image-20221126102109323.png
create mode 100644 Git.assets/image-20221126102454497.png
create mode 100644 Git.assets/image-20221126103422353.png
create mode 100644 Git.assets/image-20221126103510905.png
create mode 100644 Git.assets/image-20221126105352918.png
create mode 100644 Git.assets/image-20221126105425085.png
create mode 100644 Git.assets/image-20221126111421660.png
create mode 100644 Git.assets/image-20221126112155748.png
create mode 100644 Git.assets/image-20221126113329134.png
create mode 100644 Git.assets/image-20221126113408179.png
create mode 100644 Git.assets/image-20221126113504543.png
create mode 100644 Git.assets/image-20221126140521073.png
create mode 100644 Git.assets/image-20221126140621265.png
create mode 100644 Git.assets/image-20221126141042324.png
create mode 100644 Git.assets/image-20221126141659103.png
create mode 100644 Git.md
create mode 100644 Git.pdf

```

如果直接用git commit命令,会启动一个编辑器,在编辑器中记录要提交的信息。

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git status
On branch master
nothing to commit, working tree clean
```

当前工作树处于刚刚完成提交的状态, 结果显示没有更改

## 查看提交日志

git log可以查看以往仓库中提交的日志

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

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git log
commit 4e695916c4e3e0ad5c5df1075d9044d3e39ceae1 (HEAD -> master)
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 14:28:49 2022 +0800

    新增文件

commit cf517d33dacf9ef2432bc04ea36105c06e07b87f
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 14:24:39 2022 +0800

    firstcommit

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git log --pretty=short
commit 4e695916c4e3e0ad5c5df1075d9044d3e39ceae1 (HEAD -> master)
Author: 邱文斌 <741510728@qq.com>

    新增文件

commit cf517d33dacf9ef2432bc04ea36105c06e07b87f
Author: 邱文斌 <741510728@qq.com>

    firstcommit

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$
```

只显示指定目录或文件的日志

```
git log 文件名/目录名
```

显示文件的改动

```
git log -p 文件名/目录名
```

## 查看更改前后的差别

git diff

查看工作树、暂存区、最新提交之间的差别。

```
$ git diff
diff --git a/Git.md b/Git.md
index ba299c3..b341732 100644
--- a/Git.md
+++ b/Git.md
@@ -256,3 +256,9 @@ git log 文件名/目录名
git log -p 文件名/目录名
```

+### 查看更改前后的差别

```
+
+git diff
+
+查看工作树、暂存区、最新提交之间的差别。
+
```

执行git add之后，工作树和暂存区的状态并无差别，结果什么都不会显示。

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git diff
diff --git a/Git.md b/Git.md
index b341732..052a767 100644
--- a/Git.md
+++ b/Git.md
@@ -262,3 +262,4 @@ git diff
```

查看工作树、暂存区、最新提交之间的差别。

+执行git add之后，工作树和暂存区的状态并无差别，结果什么都不会显示。  
\ No newline at end of file

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git add .
```

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git diff
```

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
```

可以使用以下命令

```
git diff HEAD
```

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git diff HEAD
diff --git a/Git.assets/image-20221126150209605.png b/Git.assets/image-20221126150209605.png
new file mode 100644
index 0000000..e04b7b2
Binary files /dev/null and b/Git.assets/image-20221126150209605.png differ
diff --git a/Git.assets/image-20221126150243902.png b/Git.assets/image-20221126150243902.png
new file mode 100644
index 0000000..d1a2041
Binary files /dev/null and b/Git.assets/image-20221126150243902.png differ
diff --git a/Git.md b/Git.md
index b341732..223d7cf 100644
--- a/Git.md
+++ b/Git.md
@@ -262,3 +262,8 @@ git diff
```

查看工作树、暂存区、最新提交之间的差别。

```
+![image-20221126150209605](Git.assets/image-20221126150209605.png)
+
+执行git add之后，工作树和暂存区的状态并无差别，结果什么都不会显示。
+
+![image-20221126150243902](Git.assets/image-20221126150243902.png)
\ No newline at end of file
```

查看本次提交和上一次提交之间有什么差别，等确认完毕之后，再进行提交（在执行git commit之前，要执行下git diff HEAD）

HEAD指当前分支中最新的依次提交的指针。

在提交之前，多查看下，提交之后，可以查看下日志，确认是否提交成功。

```
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Git.assets/image-20221126150209605.png
    new file:   Git.assets/image-20221126150243902.png
    new file:   Git.assets/image-20221126150401815.png
    modified:   Git.md

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git commit -m "add index"
[master 3500456] add index
 4 files changed, 17 insertions(+)
 create mode 100644 Git.assets/image-20221126150209605.png
 create mode 100644 Git.assets/image-20221126150243902.png
 create mode 100644 Git.assets/image-20221126150401815.png

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git log
commit 3500456fd1b2207d4f22dcbf7a364b6eedef29a9 (HEAD -> master)
Author: 邱文斌 <741510728@qq.com>
Date:   Sat Nov 26 15:06:30 2022 +0800

    add index

commit 824825503232a83bc4f0454a3e1af227ec55da10
Author: 邱文斌 <741510728@qq.com>
Date:   Sat Nov 26 15:00:53 2022 +0800

    git diff

commit 7425dc3f8448b766dbb506d7291a7ab505c730da
Author: 邱文斌 <741510728@qq.com>
```

## <<<<<< HEAD

## 分支操作

master分支是git默认创建的分支，基本上所有的开发都是以master为中心进行的。

可以在不同的分支中，同时进行完全不同的作业。等各个分支的作业完成之后，再与master分支进行合并。

使用分支可以让多人同时高效的进行并行开发。

## 显示分支

```
git branch
```

将分支名的列表显示，可以确认当前所在分支。

```
Administrator@qiu MINGW64 /f/ta
$ git branch
* master
```

master左面的`*`，表示我们当前所在的分支。

## 创建、切换分支

```
git checkout -b 分支名
```

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (feature-A)
$ git checkout -b feature-A
Switched to a new branch 'feature-A'

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (feature-A)
$ git branch
* feature-A
  master
```

执行下面两条命令也可以达到同样的效果

```
git branch 分支名
```

```
git checkout 分支名
```

不断对一个分支进行提交操作，称之为：培育分支

```
Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (feature-A)
$ git commit -m "feature"
[feature-A 71fb375] feature
4 files changed, 38 insertions(+), 1 deletion(-)
create mode 100644 Git.assets/image-20221126151851573.png
create mode 100644 Git.assets/image-20221126152046361.png
create mode 100644 Git.assets/image-20221126152108292.png

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (feature-A)
$ git log
commit 71fb37535d59465c154019c0d035cede2fcb0b76 (HEAD -> feature-A)
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 15:24:28 2022 +0800

    feature

commit 72c4c1ba931ae9f975c6fe991fdaeeea797fe3da (master)
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 15:11:09 2022 +0800

    pdf
```

如果想切换回master分支，可以执行 `git checkout master`

可以切换回master分支，查看master分支下的文件是否有变化

以下图片是切换回master分支之后，查看的git日志情况

```

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (feature-A)
$ git checkout master
Switched to branch 'master'

Administrator@qiu MINGW64 /f/tarena/授课/枣庄/VIP创新课程（行业）/note (master)
$ git log
commit 72c4c1ba931ae9f975c6fe991fdaeeea797fe3da (HEAD -> master)
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 15:11:09 2022 +0800

    pdf

commit 1dffeaa15de8a4c9f4ab9289a07382afe23f18e8f
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 15:08:12 2022 +0800

    git basic operation

commit 3500456fd1b2207d4f22dcbf7a364b6eedef29a9
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 15:06:30 2022 +0800

    add index

commit 824825503232a83bc4f0454a3e1af227ec55da10
Author: 邱文斌 <741510728@qq.com>
Date: Sat Nov 26 15:00:53 2022 +0800

    git diff

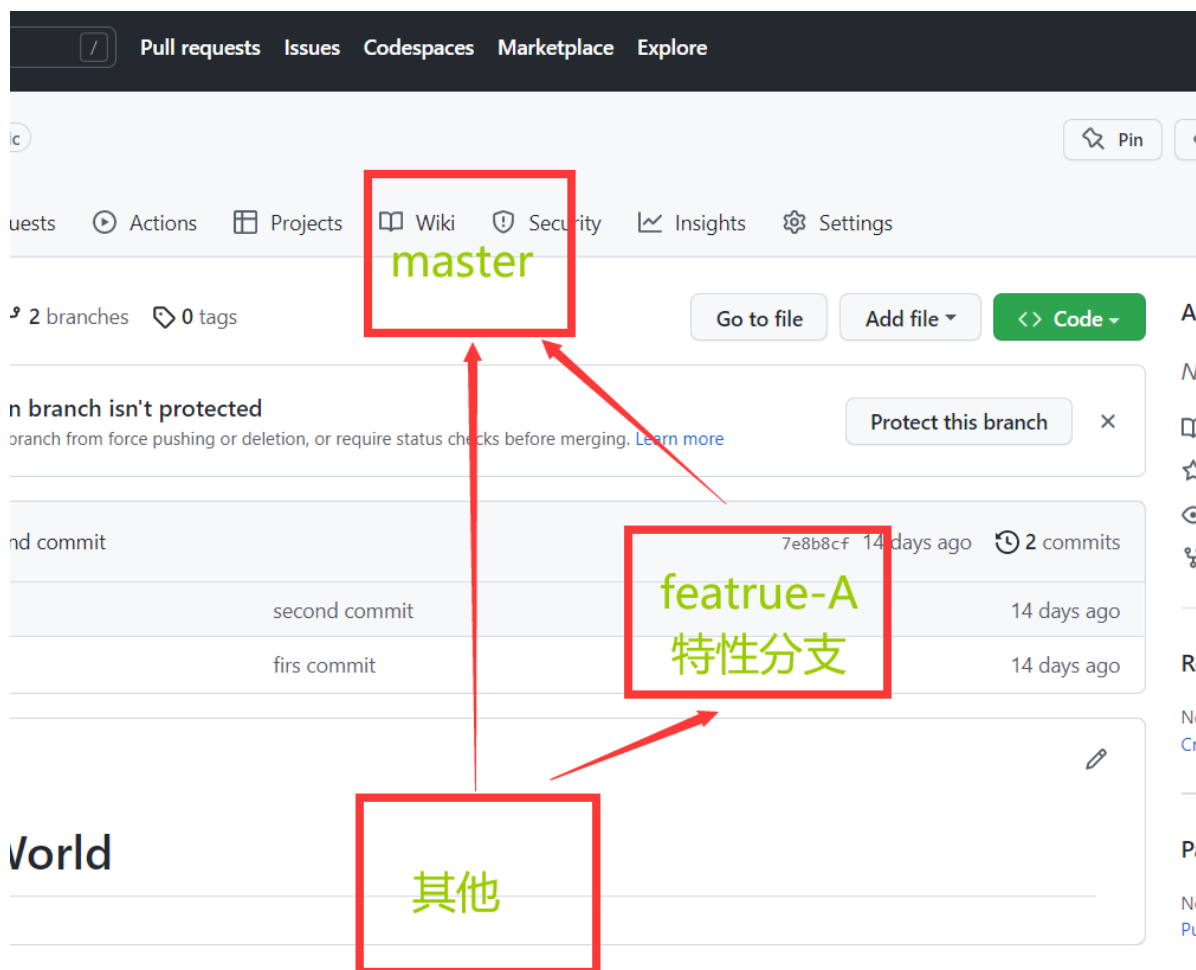
```

## 特性分支

Topic

集中实现单一特性，除此之外，不进行任何作业的分支。

日常开发中，会保留一个稳定分支做发布用，通常是master分支。



## 主干分支

是特性分支的原点，也就是合并的终点。

master

Tag创建版本信息，管理多个版本的发布。

## 合并分支

```
git merge
```

想要实现合并分支到主干分支master中，首先切换到master分支

```
git checkout master
# 为了在历史记录中明确记录本次的分支合并，我们要创建合并提交，参数 --no-ff
git merge --no-ff feature-A
```

## >>>>>> feature-A

## 以图表形式查看分支

```
git log --graph
```

```
$ git log --graph
* commit 21b4d1c1e632c0c0aabe66177acf652c9972721e (HEAD -> master)
  Merge: 21eeb46 34bcf7c
  Author: 邱文斌 <741510728@qq.com>
  Date: Sat Dec 10 08:36:43 2022 +0800

    合并到master

* commit 34bcf7ca6e90ab94cc560df139440849c49d0344 (feature-A)
  Author: 邱文斌 <741510728@qq.com>
  Date: Sat Dec 10 08:33:20 2022 +0800

    分支合并

* commit adad1fb639b1b6f10b903c4bd83b9ca9ebf1260d
  Author: 邱文斌 <741510728@qq.com>
  Date: Sat Dec 10 08:29:56 2022 +0800

    合并分支

* commit e124b4e946e287ed9c955bed405722b14f598846
  Author: 邱文斌 <741510728@qq.com>
  Date: Sat Nov 26 15:32:53 2022 +0800

...skipping...
* commit 21b4d1c1e632c0c0aabe66177acf652c9972721e (HEAD -> master)
  Merge: 21eeb46 34bcf7c
  Author: 邱文斌 <741510728@qq.com>
  Date: Sat Dec 10 08:36:43 2022 +0800

    合并到master

* commit 34bcf7ca6e90ab94cc560df139440849c49d0344 (feature-A)
  Author: 邱文斌 <741510728@qq.com>
  Date: Sat Dec 10 08:33:20 2022 +0800

    分支合并
```

## 更改提交的操作



Git可以灵活操作历史版本

```
git reset --hard 哈希值
```

## 创建fix-B分支

```
# 特性分支
git checkout -b fix-B
echo "fix-B" >> readme.txt
git add .
git commit -m "fix-B"
git reflog # 查看操作日志，可以看到所有的git命令执行记录，但是不要用git的GC，可以使用git
reglout恢复到原先的状态
git checkout master
git reset --hard f078ada # 以图表形式查看分支
git reset --hard 21b4d1c #合并到master
```

## 消除冲突

```
# 合并fix-B
```

```
image 2022120100715100jcrkdsccomage 2
<<<<<< HEAD
=====
## 分支操作
790
>>>>>> feature-A
### 以图表形式查看分支
```

用编辑器打开文件的时候，会发现出现如上图代码，如果是md文件，需要通过查看源码模式查看。

=====以上的部分是当前HEAD的内容，以下的部分是要合并的分支中的内容。

实际开发中，可能需要删除其中之一，所以在处理冲突的时候，要仔细分析冲突部分的内容或代码，再进行修改。

## 修改提交信息

```
git commit --amend
```