配置用户信息

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

git config --global user.email 2501204164@qq.com 用户邮箱

git config --global credentiak.helper store 保存密码

git config –-global -–list 展示已经保存的用户信息

新建一个仓库

mkdir learn-git 创建一个 learn-git 目录

cd learn-git 进入 learn-git 文件夹

git init 本地创建一个仓库

    Initialized empty Git repository in D:/APPs/APP_Programme/Git/learn-git/.git/

git clone repo-url 从 GitHub 上克隆一个仓库

ls 显示目录下的文件，不包括隐藏文件

ls-a 显示目录下的文件，包括隐藏文件

ls -altr 列表的形式展示目录下的文件信息

git ls-files 显示暂存区的文件，或仓库内的文件

\rm -rf filename 删除 filename 文件夹

cp -rf repoName 复制仓库

cd .. 返回上一级目录

工作区、文件状态

工作区 working directory：工作目录或者本地目录，自己电脑上的目录

暂存区 staging area：临时存储区域，用于保存即将提交到本地仓库的文件

本地仓库 local repository：自己创建的仓库，用于存放版本信息和代码的主要位置

文件状态：

未跟踪 untrack 新创建还没有被 git 管理起来

未修改 unmodified 已经被 git 管理但是内容还没有修改的文件

已修改 modified 内容修改后但是没有 add 到暂存区中

已暂存 staged 已经 add 到暂存区

添加和提交文件

git status 查看仓库状态，可以查看当前仓库处在哪个分支，以及文件的状态

echo “content” > filename 创建一个 filename 文件，并将 content 写入到文件中

echo “content” >> filename 向 filename 中追加 content

cat filename 查看 filename 文件的内容

git add filename 将 filename 提交到暂存区

git rm –cached filename 将提交到暂存区的文件撤回暂存区

git add . 将所有未提交到暂存区的文件提交到暂存区

git add *.txt 将所有的 txt 文件添加到暂存区里，txt 可以换成其他的文件

git restore --staged filename 将已经 add 到暂存区的文件撤销 add

git commit -m“information”将暂存区的文件提交到仓库 -m 可以省略，commit 只会提交暂存区的文件

git commit -a -m“information” 直接将文件添加到暂存区和仓库

git log 展示提交历史

git log --oneline 展示简洁的版本信息

git reset 回退版本

git reset --soft HEAD^ 回退到某个版本，保留工作区和暂存区的内容

git reset --hard HEAD^ 回退到某个版本，删除工作区和暂存区的内容

git reset --mix HEAD^ 回退到某个版本，只保留工作区的内容，删除暂存区的内容

git reflog 查看操作记录，再使用 git rese 回退到需要的版本

git diff 查看差异

git diff 可以用来查看工作区、暂存区、本地仓库之间的差异，以及两个版本之间的差异

git diff 查看工作区和暂存区的差异

git diff --cached 比较暂存区和仓库之间的差异

git diff HEAD 查看暂存区和版本库的差异

git diff 版本号 1 版本号 2 查看两个特定的版本之间的差异

```
Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ git log --oneline
d423180 (HEAD -> master) add all
2ba6c4a add all
ee11d89 第一次提交

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ git diff d423180 ee11d89
diff --git a/file.txt b/file.txt
index cb856a4..e1cfd5c 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1 @@
-444
\ No newline at end of file
+这是第一个文件
\ No newline at end of file
```

git diff HEAD~ HEAD 查看上一个版本和现在的版本之间的区别

git diff HEAD~n HEAD 查看上 n 个版本和现在的版本之间的区别

git diff HEAD~n HEAD filename 查看上 n 个版本和现在的版本之间 filename 的区别

删除文件

rm filename 删除本地工作区中的文件，但暂存区中的文件还在，需要 git add filename 来删除暂存区的文件

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    file.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

git rm filename 直接删除暂存区的文件，然后 git commit 删除版本库中的文件

.gitignore 忽略文件

系统或者软件自动生成的文件

编译产生的中间文件和结果文件

运行时生成日志文件、缓存文件、临时文件

涉及身份、密码、口令、密钥的隐私文件

echo filename > .gitignore 将文件添加到忽略文件名单中

```
Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ echo access.log > .gitignore

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ cat .gitigore
cat: .gitigore: No such file or directory

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ cat .gitignore
access.log

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        other.log

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

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$
```

vi .gitignore 修改.gitignore 文件，将*.log 添加进去，即忽略所有的 log 文件，此时再创建.log 文件也不会被识别出来，但是已经提交到版本库的文件不会被忽略，若产生变动，仍需要提交文件，向.gitignore 中添加 temp\，则 temp 文件夹内的文件都不会被提交到版本库中 github.com/github/gitignore

*匹配多个字符

？匹配单个字符

[]匹配列表中的单个字符。例如[abc]表示 a/b/c

表示匹配任意的中间目录，例如 doc//*.pdf 表示忽略 doc 目录下所有子目录内的 pdf 文件

中括号的内容可以使用短中线连接。例如[1-9]匹配 1-9 的数字

感叹号!表示取反，例如忽略匹配模式以外的所有文件可以再匹配模式前加!

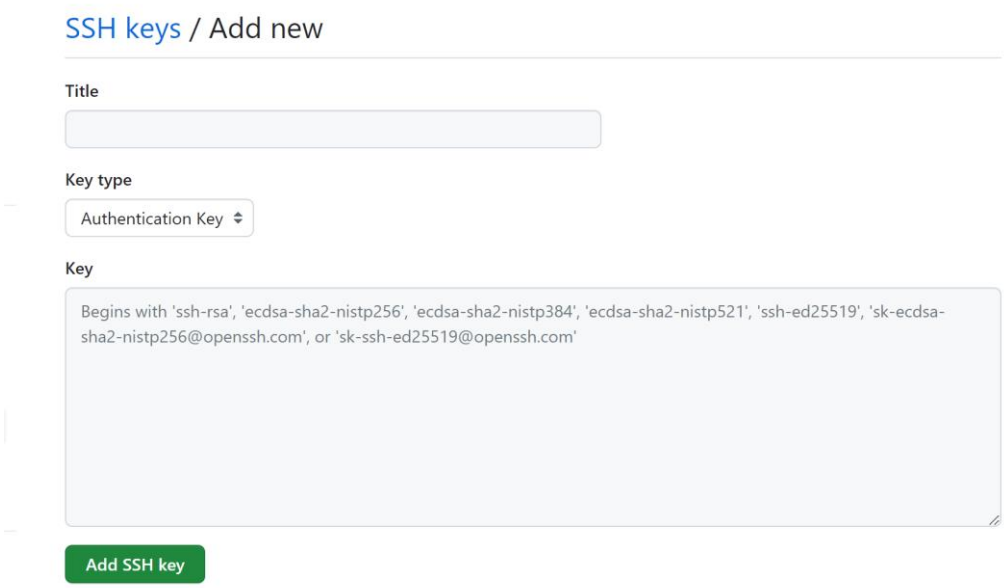
```
Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ vi .gitignore

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
```

SSH 配置和克隆仓库

回到仓库的根目录

ssh-keygen -t rsa -b 4096 生成 ssh 文件，首次配置一路 enter 就行

复制公钥文件的内容，打开 github，点击头像>settings>左侧 SSH and GPG keys>new ssh key



关联本地仓库和远程仓库

在 github 上创建一个新的仓库

Git remote add<shortname> <url>

git remote add origin git@github.com:raojj/GoLang.git 这句代码新建仓库后会自动生成，复制即可

git remote -v 查看当前仓库对应的远程仓库的别名和地址

git branch -M main 将分支的名称改成 main

git push -u origin main:main 将本地的仓库和别名为 origin 的远程仓库关联起来

git pull <远程仓库名> <远程分支名>:<本地分支名> 将远程仓库的内容 pull 到本地以及合并

git pull git@github.com:raojj/Golang.git

```
Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ git remote add origin git@github.com:raojj/GoLang.git

Joe@DESKTOP-UF4O3SB MINGW64 /d/APPs/APP_Programme/Git/learn-git/my-repo (master)
$ git remote -v
origin  git@github.com:raojj/GoLang.git (fetch)
origin  git@github.com:raojj/GoLang.git (push)
```

分支简介和基本操作

团队开发和功能管理，分支开发最后合并到主分支

git branch 查看当前仓库的所有分支

git branch branchName 创建一个叫 branchName 的分支

git switch branchName 切换到 branchName 分支上

git checkout filename 将 fileName 还原到前一个版本

git merge branchName 先切换到 main 分支，执行语句，将 branchName 合并到 main 分支中

git log --graph --oneline --decorate --all 查看分支图

git branch -d branchName 删除 branchName（已经合并后的分支）

git branch -D branchName 强制删除 branchName(还没有合并的分支)

解决合并冲突

若两个分支对同一个文件做出了修改，那么就会产生冲突

git status 查看产生冲突的文件列表

git diff 查看产生冲突文件的区别

vi filename 修改 filename 的内容

修改好冲突的文件后，提交一下，提交后会自动合并分支

若想中止分支合并可以使用

git merge --abort

回退或 rebase

可以在任意分支上执行 rebase 操作

git rebase main 那么另外一个分支的记录就会拼接到 main 分支上

git rebase otherBranch 那么开始分叉后的 main 分支就会拼接到 otherBranch 上，但最后都是形成一条直线

只是中间的顺序有所不同