



Git和Github

新手入门必备

作者：晨曦中的暮雨

Github账号：foreverfanvy

Github

- **GitHub**是一个在线软件源代码托管服务平台，用于公开程序或软件的代码。使用Git作为版本控制软件，由开发者克里斯·汪斯崔斯、P·J·海特和汤姆·普雷斯顿·沃纳使用Ruby on Rails编写而成。在2018年，GitHub被微软公司收购。
- GitHub是最流行的Git访问站点^[5]。除了允许个人和组织建立和访问保管中的代码以外，它也提供了一些方便社会化共同软件开发的功能，即一般人口中的社群功能，包括允许用户追踪其他用户、组织、软件库的动态，对软件代码的改动和bug提出评论等。GitHub也提供了图表功能，用于概观显示开发者们怎样在代码库上工作以及软件的开发活跃程度。

Github的用途

- GitHub通常用于软件开发。GitHub还支持以下格式和功能：
- 文档：包括自动生成的、采用类[Markdown](#)语言的[Readme](#)文件（称作GitHub Flavored Markdown, GFM）。
- 问题追踪系统（同时可用于功能需求）
- [Wiki](#)
- [GitHub Pages](#)支持用户通过软件仓库建立静态网站或静态博客（通过一个名为Jekyll的软件实现，但是也支持采用诸如Hexo等其他博客引擎搭建）。
- [任务列表](#)
- [甘特图](#)
- 可视化的地理位置分析
- 预览3D渲染文件^[16]。预览功能通过[WebGL](#)和[Three.js](#)实现。
- 预览[Adobe Photoshop](#)的PSD文件，甚至可以比较同一文件的不同版本。

GitHub Pages（可以建立自己的网站）

- **GitHub Pages**
- 主条目：[GitHub Pages](#)
- GitHub Pages是GitHub提供的一个[网页托管服务](#)，可以用于存放静态网页，包括博客、项目文档甚至整本书^[18]。一般GitHub Pages的网站使用github.io的子域名，但是用户也可以使用第三方域名。Github Pages以开源仓库公开静态网页源代码，可在仓库->设置->Code and automation里设置，<https://github.com/>（[页面存档备份](#)，存于[互联网档案馆](#)）<用户名>/<仓库名>/settings/pages。

创建一个账号

- 注册参考CSDN文章: [注册Github账号详细教程【超详细篇 适合新手入门】_github注册-CSDN博客](#)
- [注册链接: GitHub](#)



Top repositories



Find a repository...

foreverfanvy/location_git_code

code50/145102356

foreverfanvy/Data_struature-learning

foreverfanvy/cs61b

Home

[Send feedback](#)[Filter](#)

<> Start writing code

Start a new repository for foreverfanvy

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

name your new repository...

☐ Public

Anyone on the internet can see this repository

☒ Private

You choose who can see and commit to this repository

[Create a new repository](#)

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

foreverfanvy / README.md

[Create](#)

```
1 - 🦋 Hi, I'm @foreverfanvy
2 - 📺 I'm interested in ...
3 - 🌱 I'm currently learning ...
4 - ❤️ I'm looking to collaborate on ...
5 - 💬 How to reach me ...
6 - 🗨️ Pronouns: ...
7 - ⚡ Fun fact: ...
8
```

Use tools of the trade

Write code in your web browser



Use [the github.dev web-based editor](#) from your repository or pull request to create and commit changes.

Get AI-based coding suggestions



Try [GitHub Copilot free for 30 days](#), which suggests entire functions in real time, right from your editor.

Get started on GitHub

UNIVERSE'24

Less than one week left to get free virtual tickets to GitHub Universe, our global developer event on Oct. 29-30.

[Get tickets](#)

Latest changes

- Yesterday
Improving GitHub Copilot C++ completions in Visual Studio
- 2 days ago
Secret scanning supports delegated bypass for push protection on file uploads (GA)
- 3 days ago
Secret scanning support for public leak and multi-repository indicators in webhook and...
- 4 days ago
Repository deploy keys are controlled by enterprise and organization policy [GA]

[View changelog](#)

Explore repositories

AzatAI / cs_books



Computer science books Recommended by AzatAI.

Git初步

- 了解基础应用
- 下载链接win系统: <https://github.com/git-for-windows/git/releases/download/v2.47.0.windows.2/Git-2.47.0.2-64-bit.exe>
- Git原理简介:
 - **快照 (Snapshot):** Git 将代码的每一个版本保存为一个快照, 而不是差异 (差异是其他版本控制系统常用的方式)。每次提交 (commit) 都保存了文件的完整状态, 但对于没有变化的文件, Git 会存储指向之前快照的引用。
 - **Blob、Tree、Commit 对象:**
 - **Blob** 对象存储文件的内容。
 - **Tree** 对象描述文件和文件夹的结构。
 - **Commit** 对象保存快照的引用、提交信息、以及指向上一个提交的指针。
 - [Using Git | CS 61B Spring 2024](#) **分布式系统介绍原理**

普通Git Bash按照比较麻烦

- 参考文档:
 - [Lab 01: Setup | CS 61B Spring 2024](#)

两大主要操作

- 分支 (Branch): Git 的分支操作非常轻量化, 可以帮助开发者快速地从主分支分离出一个新分支, 用于新功能的开发或修复问题。
- 合并 (Merge): 开发完成后, 可以将分支合并回主分支, Git 通过其合并算法来解决不同分支之间的冲突。



Git初步

- **Git 的三个区域**

- **工作区 (Working Directory):** 开发者日常编辑的工作区域。
- **暂存区 (Staging Area):** 保存即将提交的修改，但尚未提交。
- **本地仓库 (Local Repository):** 提交后的文件状态和历史记录。

- **分布式工作流程**

- **推送 (Push):** 将本地仓库的更改推送到远程仓库（例如 GitHub）。
- **拉取 (Pull):** 将远程仓库的更改拉取到本地仓库，保持同步。
- **克隆 (Clone):** 从远程仓库创建一个本地副本。



指令作用 - 用于git bash

- **Git 的基本操作**

- **git init**: 初始化一个新的 Git 仓库。
- **git add**: 将文件添加到暂存区。
- **git commit**: 提交暂存区中的文件到本地仓库。
- **git push**: 推送本地更改到远程仓库。
- **git pull**: 拉取远程仓库的最新更改并合并到本地分支。





命令提示符



Windows PowerShell



MINGW64:/c/Users/yaoji



yaoji@aomu MINGW64 ~


\$ |

非常难使用，建议不要用这个，
用图形化工具或者像JetBrain和
Vscode对应的Git同步程序！！！！

git init: 初始化一个新的 Git 仓库

1. git init: 初始化一个新的 Git 仓库

bash

 复制代码

创建一个新文件夹并进入该文件夹

mkdir my_project

cd my_project

初始化一个新的 Git 仓库

git init

结果：这个命令会在当前目录中创建一个 `.git` 文件夹，用于管理 Git 仓库。

git add: 将文件添加到暂存区

2. git add: 将文件添加到暂存区

bash

 复制代码

创建一个新的文件

```
echo "Hello, Git!" > hello.txt
```

将文件添加到暂存区


```
git add hello.txt
```

结果: `hello.txt` 文件现在在暂存区中, 可以随时提交。

git commit: 提交暂存区中的文件到本地仓库

3. git commit: 提交暂存区中的文件到本地仓库

bash

 复制代码

提交到本地仓库并添加提交信息

```
git commit -m "Add hello.txt with greeting message"
```

结果：此命令会把暂存区中的所有文件提交到本地仓库，并记录提交信息 "Add hello.txt with greeting message"。



git push: 推送本地更改到远程仓库

4. git push: 推送本地更改到远程仓库

bash

 复制代码

首次推送时, 设置远程仓库的 URL

```
git remote add origin https://github.com/username/my_project.git
```

推送更改到远程仓库的主分支


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

结果: 你的本地更改会被推送到远程仓库的 `main` 分支。 `-u` 参数是为了设置默认的上游分支, 以后可以直接使用 `git push` 推送。

git pull: 拉取远程仓库的最新更改并合并到本地分支

5. git pull: 拉取远程仓库的最新更改并合并到本地分支

bash

 复制代码

从远程仓库拉取并合并最新更改

```
git pull origin main
```

结果：此命令会从远程仓库的 `main` 分支拉取最新的更改，并将其合并到你的本地 `main` 分支上。如果有冲突，Git 会提示并允许你解决冲突。

假设你从头创建一个新项目，并将其推送到远程仓库的整个过程如下：

```
bash
```

 复制代码

```
# 创建项目文件夹并进入
```

```
mkdir my_project
```

```
cd my_project
```

```
# 初始化 Git 仓库
```

```
git init
```

```
# 创建一个文件并添加内容
```

```
echo "This is my first Git project!" > README.md
```

```
# 将文件添加到暂存区
```

```
git add README.md
```

```
# 提交到本地仓库
```

```
git commit -m "Initial commit with README file"
```

```
# 添加远程仓库（使用你的 GitHub 仓库 URL）
```

```
git remote add origin https://github.com/username/my_project.git
```

```
# 推送到远程仓库的 main 分支
```

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



图形化管理git仓库-《Github Desktop

- 优点:

- 可视化操作简化复杂流程: GitHub Desktop 为常用操作提供了可视化界面, 使用户无需掌握复杂的 Git 命令即可管理仓库。
- 适合团队协作: GitHub Desktop 与 GitHub 的 Pull Request 流程无缝集成, 便于团队协作开发。
- 易于解决冲突: 内置的冲突解决工具让用户在图形界面下快速处理合并冲突。



GitHub Desktop介绍

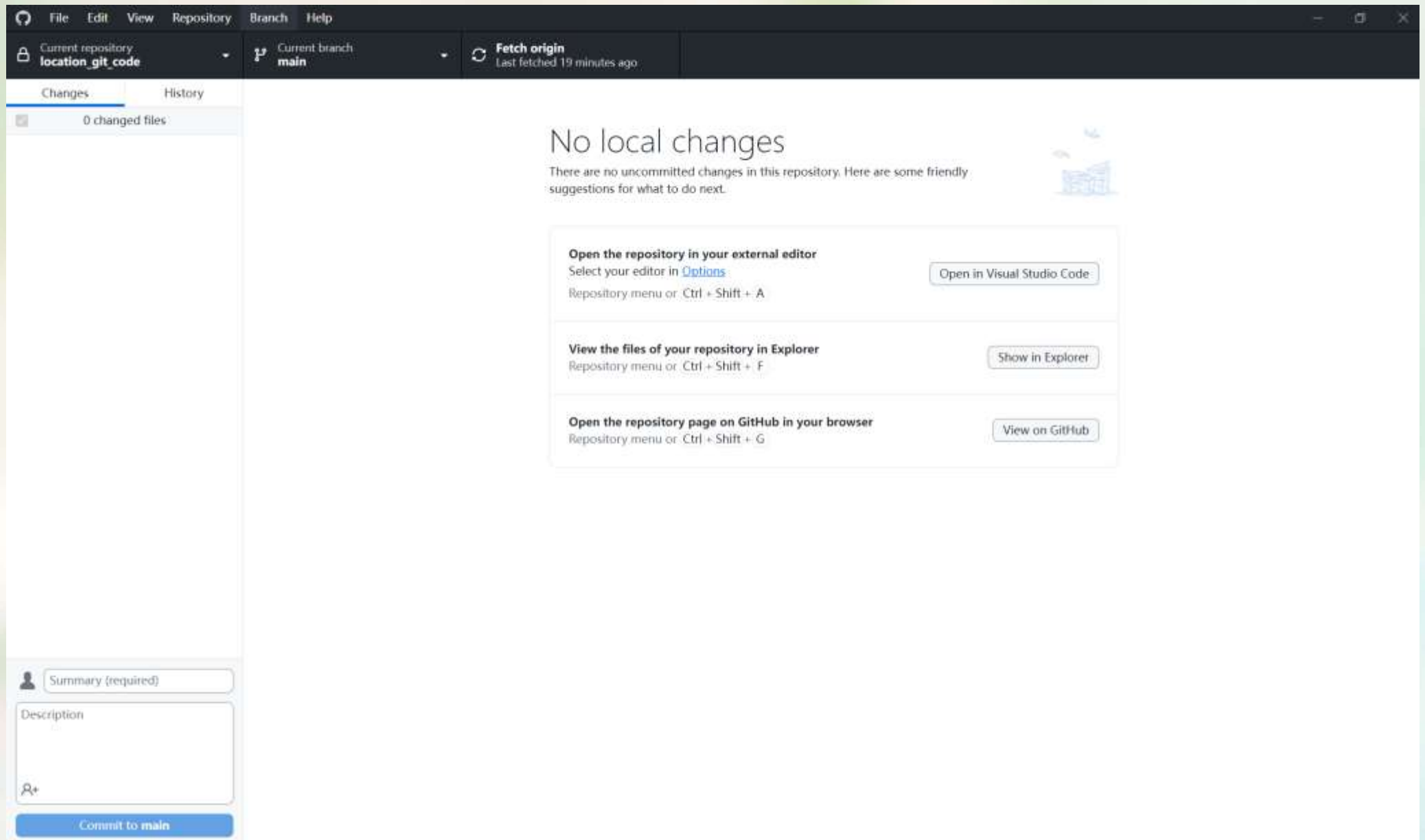
- **GitHub Desktop 的界面和功能**

- **Changes（变更）选项卡**：显示未提交的文件变更，包括新增、修改和删除的文件。您可以选择将文件的更改加入暂存区。
- **History（历史）选项卡**：查看仓库的提交历史，了解各个版本的变动。
- **Branch（分支）管理**：可以方便地创建、切换和删除分支。点击仓库名称旁边的下拉菜单，可以选择分支或创建新分支。
- **Pull Requests**：与 GitHub 网站集成，可以在桌面应用中查看、创建和管理 Pull Requests，但需要在 GitHub 网站上处理详细的 Review 工作。

- **推送到远程仓库**

- 提交后，可以点击右上角的 **Push origin** 按钮，将本地提交推送到 GitHub 上的远程仓库。
- 若团队其他成员在远程仓库有新提交，可以使用 **Fetch origin** 来更新本地仓库，然后选择 **Pull origin** 拉取更改并合并到当前分支





在终端中输入指令来生成SSH Key

通过命令 `ssh-keygen` 生成 SSH Key

```
$ ssh-keygen -t rsa
```

中间通过三次回车键确定。

查看生成的 SSH 公钥和私钥

```
$ ls ~/.ssh
```

输出：

```
id_rsa      id_rsa.pub
```

- 私钥文件 `id_rsa`
- 公钥文件 `id_rsa.pub`

读取公钥文件

```
$ cat ~/.ssh/id_rsa.pub
```

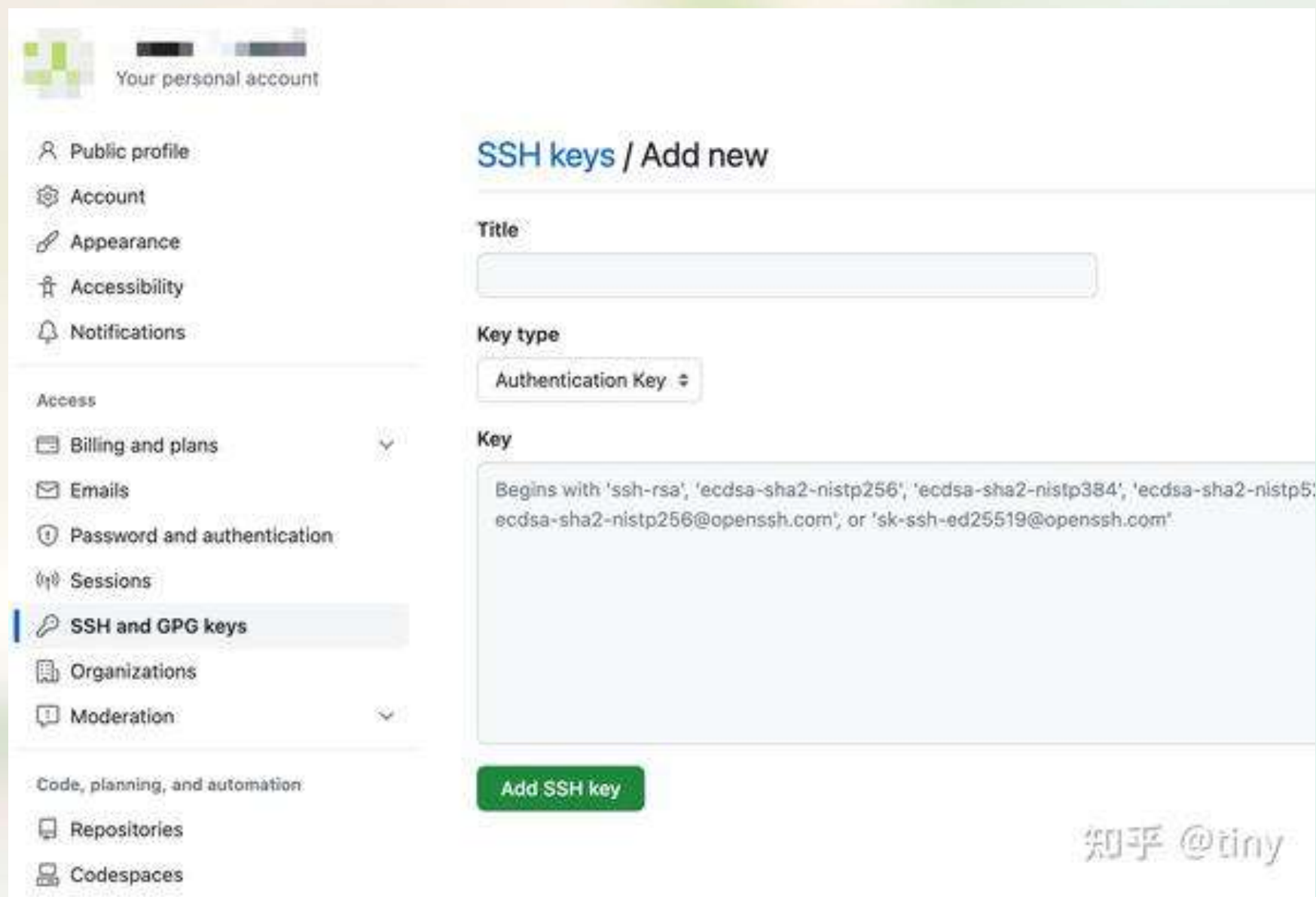
配置SSH来获取访问权限

参考文档: [本地 git 服务使用 SSH 协议连接远程仓库 - 知乎](#)

- PS W:\CCode\学校课程\untitled> **ssh-keygen -t rsa**
- Generating public/private rsa key pair.Enter file in which to save the key (C:\Users\lenovo\.ssh/id_rsa):
Created directory 'C:\Users\lenovo\.ssh'.Enter passphrase (empty for no passphrase): Enter same
passphrase again: Your identification has been saved in C:\Users\lenovo\.ssh/id_rsaYour public key has
been saved in C:\Users\lenovo\.ssh/id_rsa.pubThe key fingerprint
is:SHA256:t0OKAmEZEpvA/rGSAtzpBi8K+ShsiCGlePwd7zyRikI lenovo@YaoyaoThe key's randomart
image is:+---[RSA 3072]----+|=.. ||.= o ||.= =. ||.= = ||=oB o S.o
||X+EB ..o+ . ||O*+..o.+..o ||*oo o.o.o . ||o . .o. |+----[SHA256]-----+
- PS W:\CCode\学校课程\untitled> **ls ~/.ssh**
- 目录: C:\Users\lenovo\.sshMode LastWriteTime Length Name-----
-----a----- 2024/10/27 15:16 2602 id_rsa-a----- 2024/10/27
15:16 568 id_rsa.pub
- PS W:\CCode\学校课程\untitled> **cat ~/.ssh/id_rsa.pubssh-rsa**
AAAAB3Nz/FLBmPtl/T6FhapDfodifF5F7b2E1o+LFNAMKHArbPs6X4R0xHf6pl/vvRXEemIAV4ELXxl8XAyof
gbMgK23dkSkR7B+K2TLQGdB0SU2n3XbQBP8bA8kangRubvZqb7yBwpGzBFsyusJyVREZXTCSR5ZcvmT
Cp0E1q9i5hD5J27ETbPVgU1NHNhjfW5WawmMd5cVwDW66Yw8pT/KcHf8HihlufTBQPJl1QnuEUHiYZU
= lenovo@Yaoyao
- 把这种红色的私钥复制下来到github上绑定



通过头像的下拉菜单
「Settings」->「Access」
->「SSH and GPG keys」-
>「[New SSH key](#)」，添加
生成的 public key 添加到
当前账户中。



Your personal account

- Public profile
- Account
- Appearance
- Accessibility
- Notifications

Access

- Billing and plans
- Emails
- Password and authentication
- Sessions
- SSH and GPG keys**
- Organizations
- Moderation

Code, planning, and automation

- Repositories
- Codespaces

SSH keys / Add new

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ssh-ed25519@openssh.com', or 'sk-ecdsa-sha2-nistp256@openssh.com'

Add SSH key

测试验证（Github版，不同web不一样）

- `$ ssh -T git@github.com`
- Hi *****! You've successfully authenticated, but GitHub does not provide shell access.

