



```
Entities.java — copilot-demo  
欢迎 J Entities.java 1 x  
src > main > java > com > example > copilotdemo > J Entities.java > User  
1 package com.example.copilotdemo;  
2  
3 public class User {  
4  
5 }  
6  
资源管理器  
COPILOT-DEMO  
> .mvn  
> .vscode  
src  
main  
java/com/example/copilotdemo  
CopilotDemoApplication.java  
Entities.java 1  
resources  
test  
target  
.gitignore  
HELP.md  
mvnw  
mvnw.cmd  
pom.xml  
大纲  
时间线  
JAVA PROJECTS  
MAVEN  
行 3, 列 20 空格: 4 UTF-8 LF {} Java Prettier
```

GitHub Copilot 智能辅助编程实践

AI不会取代你，会用AI的人会取代你



 **OpenAI**
 **GitHub Copilot**

AI不会取代你，会用AI的人会取代你



GitHub Copilot

AI人工智能结对编程助手

拒绝单飞

copilot.github.com

日程

- 概述 – AI碾压人类的10个编程场景
 - GitHub Copilot 账号开通、安装/配置
 - AI辅助编程实操演示



AI 碾压人类的10个编程场景 AI=100xH

正则表达式编写

编写测试代码，包含各类边界条件验证

使用难以记忆关键字编写代码，比如：HTML/CSS编写

编写/不熟悉的复杂算法

使用/学习不熟悉的编程语言

按常识完善对象字段

示例/测试数据生成

复杂参数填写和上下文匹配

*理解复杂代码并编写文档，注释

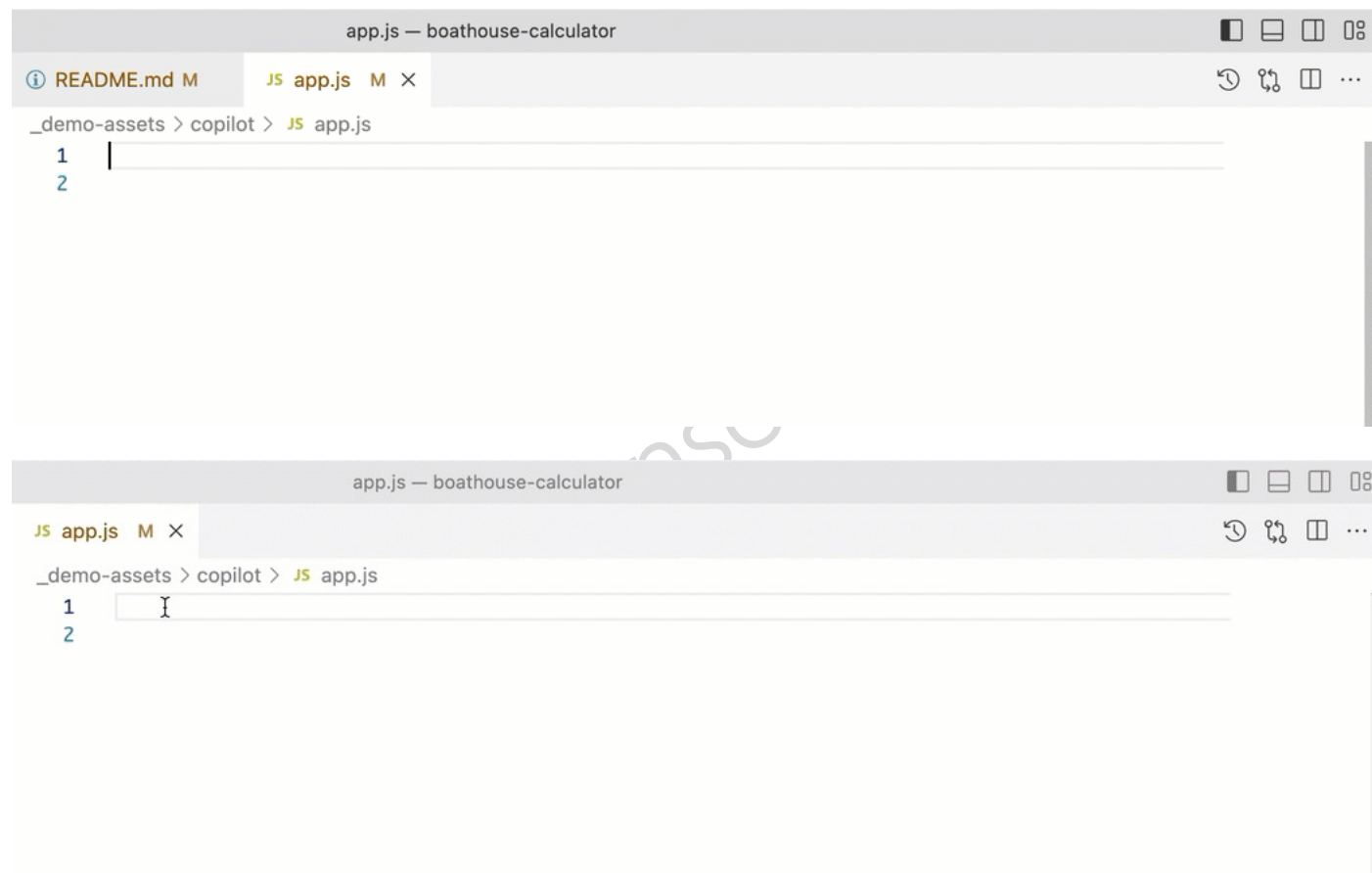
*评审代码，提出改进意见

* GitHub Copilot X 新特性

AI=100xH 1. 正则表达式编写

验证IP地址格式

验证密码复杂度



AI=100xH 2. 编写测试代码，边界条件验证

自动感知业务逻辑

生成各类边界条件测试

```
app_test.js — boathouse-calculator

JS app_test.js M ×

_demo-assets > copilot > JS app_test.js > calculator

1 function calculator(str) {
2   var reg = /^(\d+)(\+|-|\*|\/)(\d+)$/;
3   var result = reg.exec(str);
4   if (result) {
5     var num1 = parseInt(result[1]);
6     var num2 = parseInt(result[3]);
7     var operator = result[2];
8     var res = 0;
9     switch (operator) {
10      case "+":
11        res = num1 + num2;
12        break;
13      case "-":
14        res = num1 - num2;
15        break;
16      case "*":
17        res = num1 * num2;
18        break;
19      case "/":
20        res = num1 / num2;
21        break;
22      }
23     return res;
24   } else {
25     return "输入的字符串不符合要求";
26   }
27 }
28
```

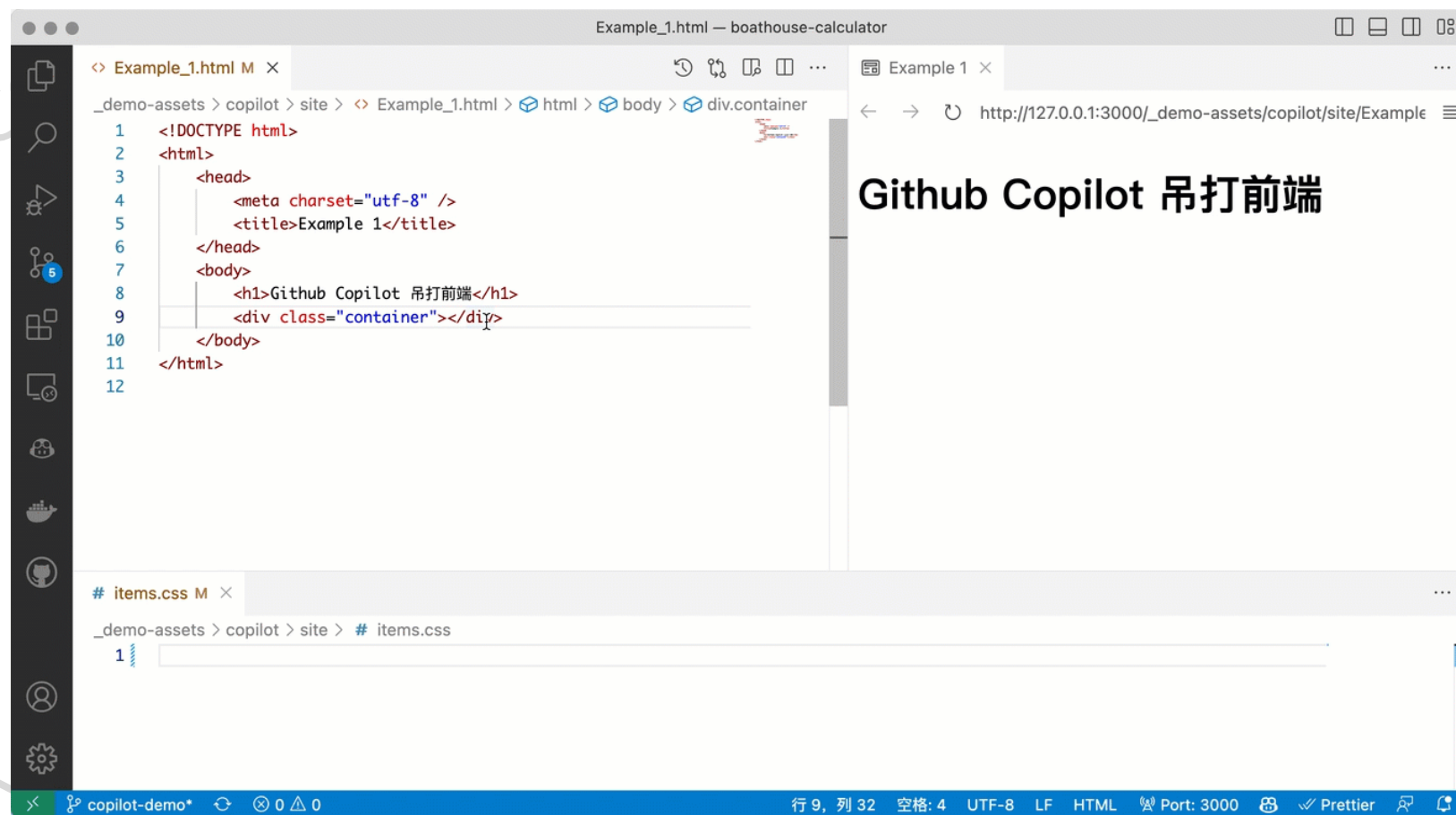

AI=100xH 3. 使用难以记忆关键字编写代码， 比如：HTML/CSS编写

HTML语法

CSS语法

引用外部资源

使用常见库资源



The screenshot shows a VS Code editor with two files open: `Example_1.html` and `items.css`. The `Example_1.html` file contains the following HTML code:

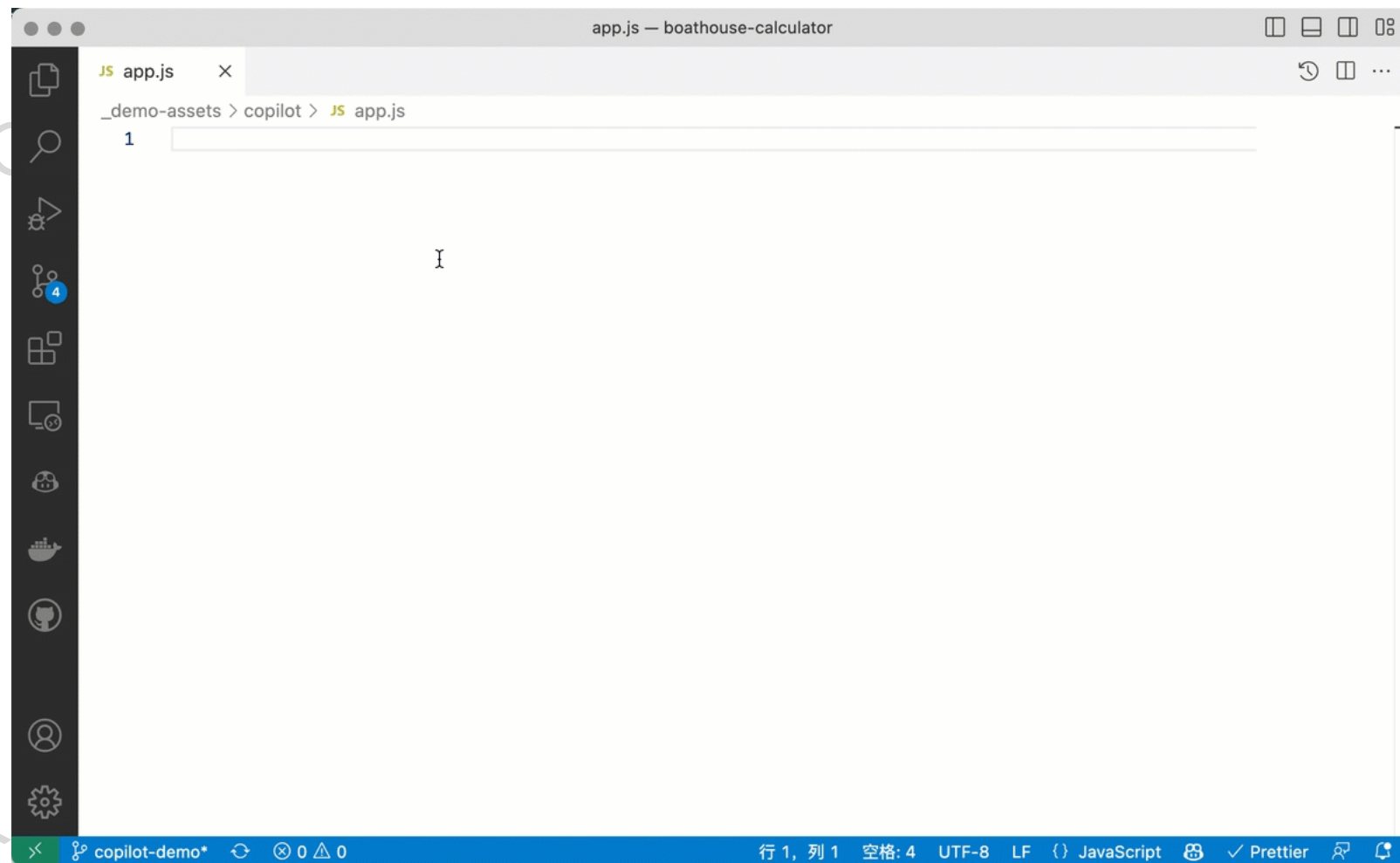
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <title>Example 1</title>
6   </head>
7   <body>
8     <h1>Github Copilot 吊打前端</h1>
9     <div class="container"></div>
10  </body>
11 </html>
```

The `items.css` file is currently empty. The browser preview on the right shows the rendered HTML, displaying the text "Github Copilot 吊打前端". The status bar at the bottom indicates the current file is `copilot-demo`, with 9 lines and 32 columns, using UTF-8 encoding and LF line endings.

Github Copilot 吊打前端

AI=100xH 4.编写/不熟悉的复杂算法

你知道什么是夏普比率吗？

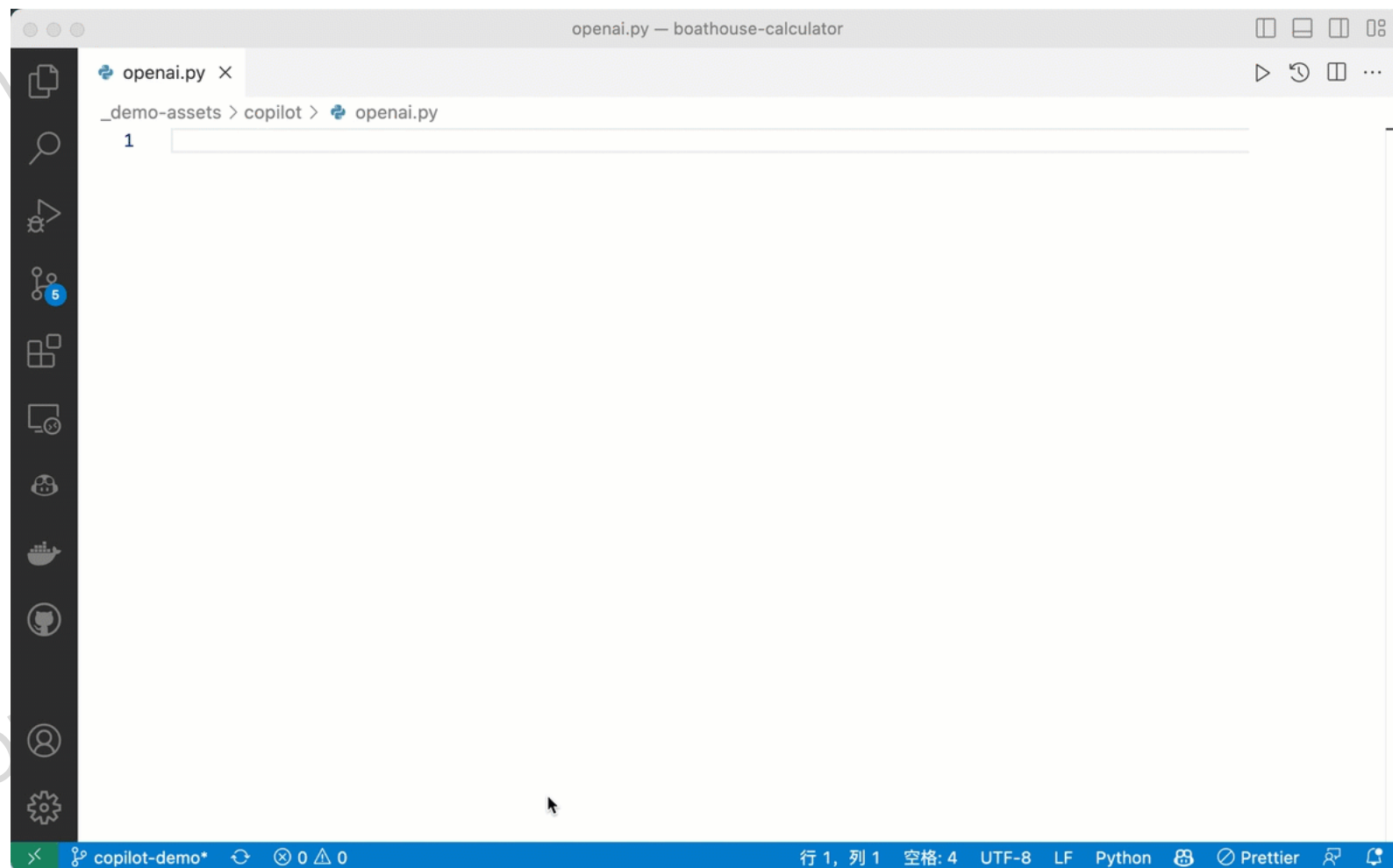


AI=100xH 5. 使用/学习不熟悉的编程语言



没学过Python

怎样用Python调用OpenAI进行
文本生成？

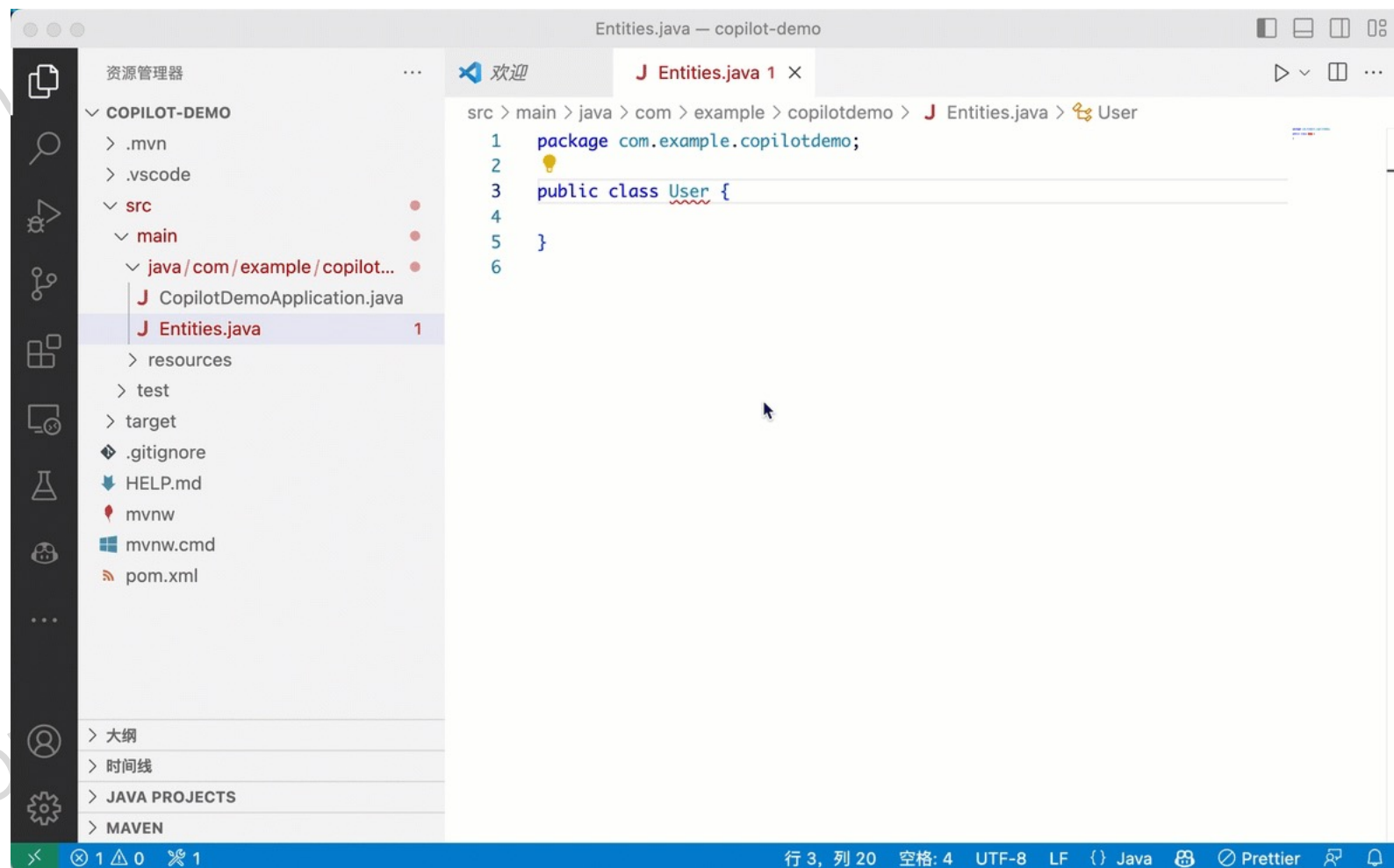


AI=100xH 6.按常识完善对象字段

User 用户

Book 书籍

Car 汽车

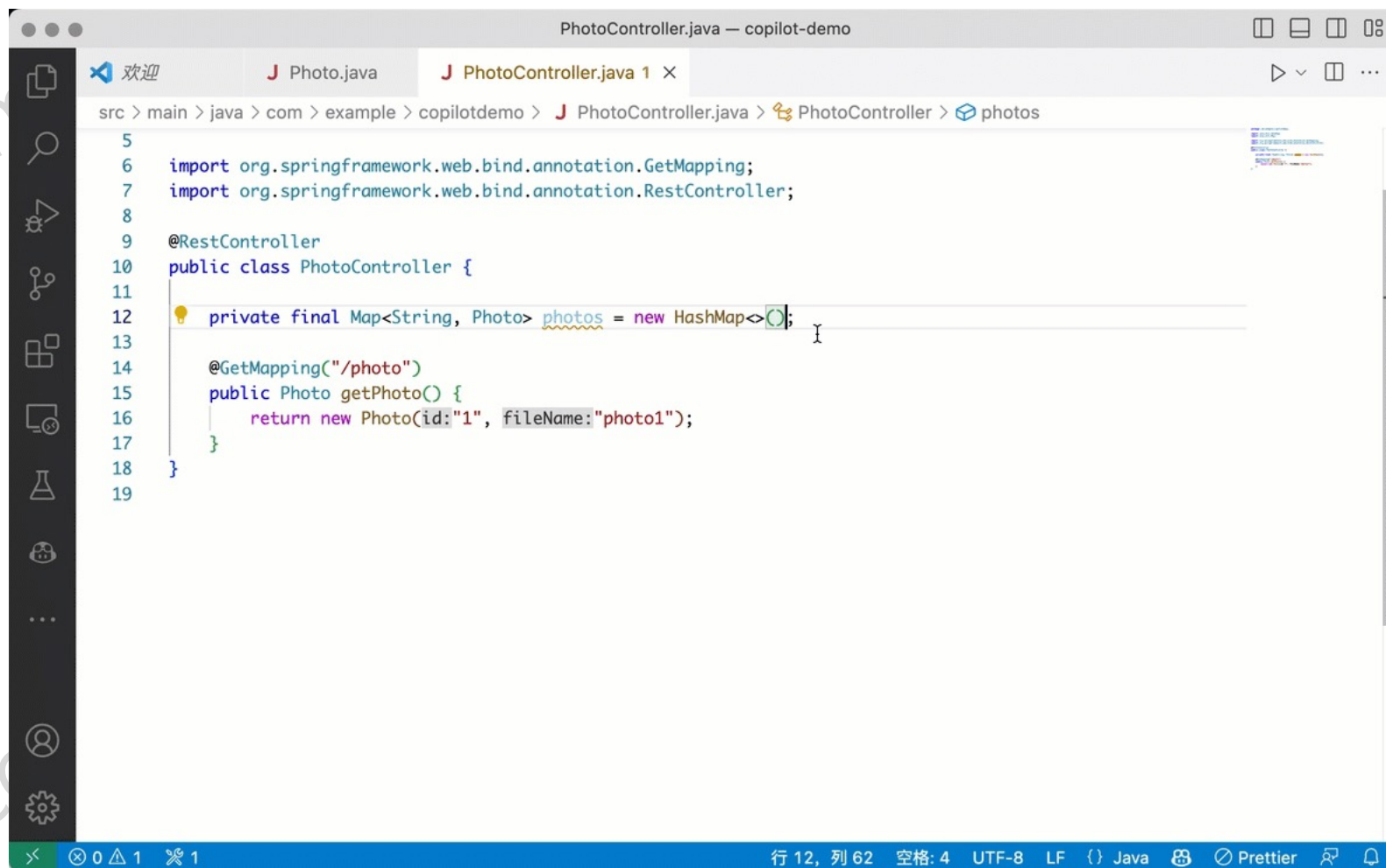


```
Entities.java — copilot-demo  
欢迎 J Entities.java 1 x  
src > main > java > com > example > copilotdemo > J Entities.java > User  
1 package com.example.copilotdemo;  
2  
3 public class User {  
4  
5 }  
6  
资源管理器  
COPILOT-DEMO  
  .mvn  
  .vscode  
  src  
    main  
      java/com/example/copilot...  
        CopilotDemoApplication.java  
        Entities.java 1  
      resources  
      test  
      target  
  .gitignore  
  HELP.md  
  mvnw  
  mvnw.cmd  
  pom.xml  
大纲  
时间线  
JAVA PROJECTS  
MAVEN  
行 3, 列 20 空格: 4 UTF-8 LF Java Prettier
```

AI=100xH 7. 示例/测试数据生成

起名字是个技术活儿

根据提示快速创建示例数据



```
PhotoController.java — copilot-demo
src > main > java > com > example > copilotdemo > PhotoController.java > PhotoController > photos

5
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @RestController
10 public class PhotoController {
11
12     private final Map<String, Photo> photos = new HashMap<>();
13
14     @GetMapping("/photo")
15     public Photo getPhoto() {
16         return new Photo(id:"1", fileName:"photo1");
17     }
18 }
19
```

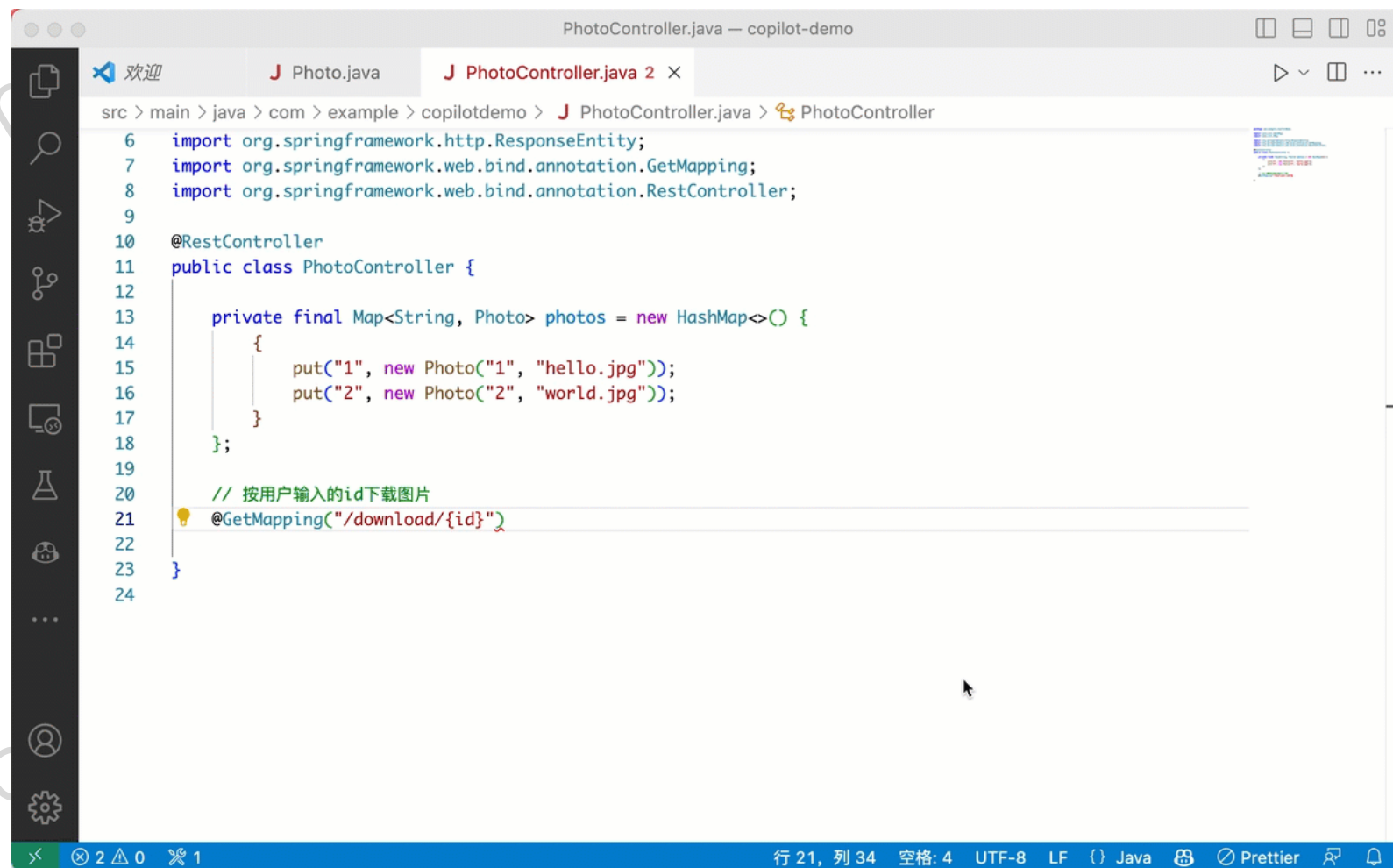
行 12, 列 62 空格: 4 UTF-8 LF {} Java Prettier

AI=100xH 8.复杂参数填写和上下文匹配

使用库函数的时候

不用再记忆复杂的参数格式

自动完成参数填充



```
PhotoController.java — copilot-demo
欢迎 | Photo.java | PhotoController.java 2 x
src > main > java > com > example > copilotdemo > PhotoController.java > PhotoController

6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.RestController;
9
10 @RestController
11 public class PhotoController {
12
13     private final Map<String, Photo> photos = new HashMap<>() {
14         {
15             put("1", new Photo("1", "hello.jpg"));
16             put("2", new Photo("2", "world.jpg"));
17         }
18     };
19
20     // 按用户输入的id下载图片
21     @GetMapping("/download/{id}")
22
23 }
24

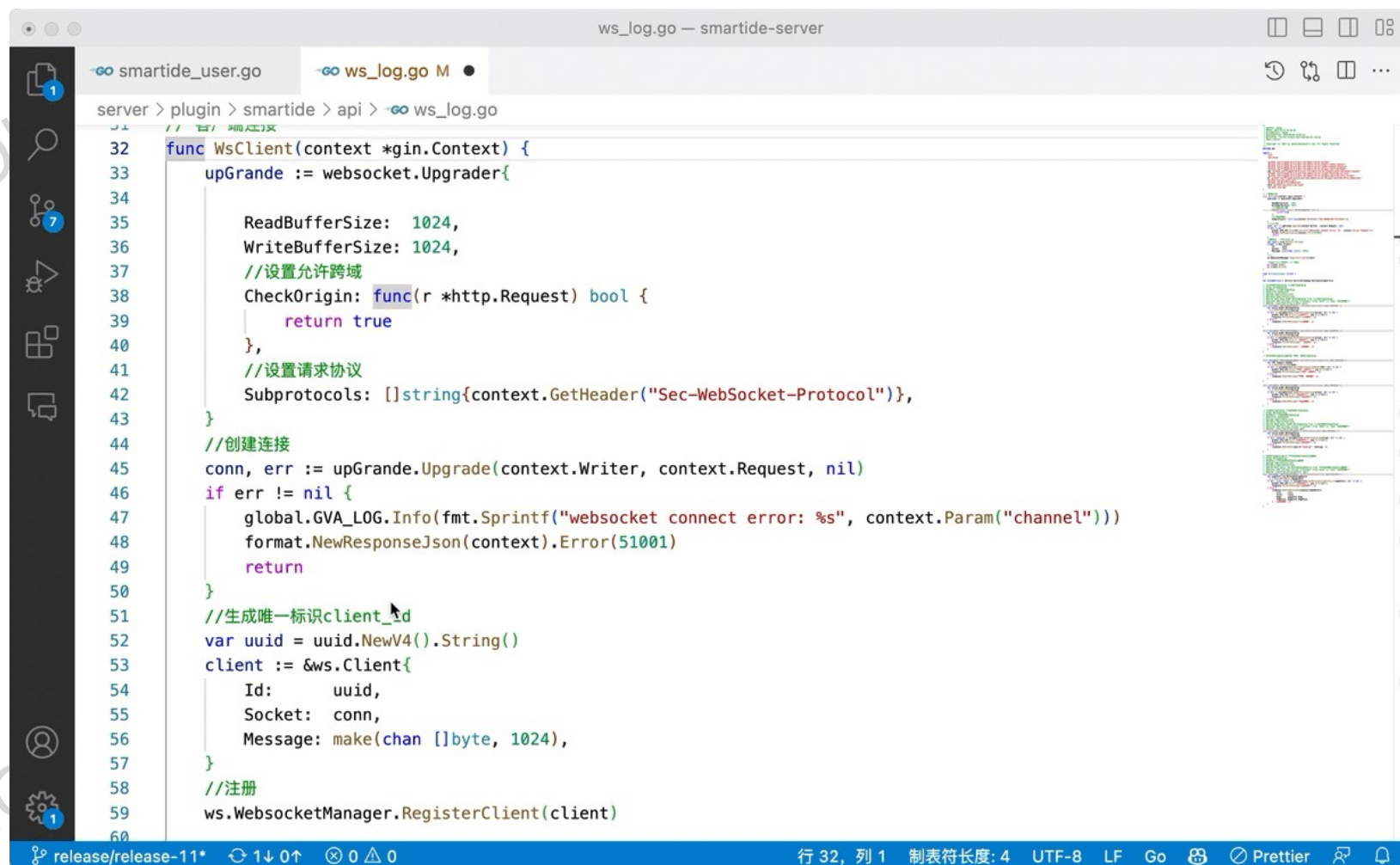
行 21, 列 34  空格: 4  UTF-8  LF  {} Java  Prettier
```


AI=100xH 9.理解复杂代码并编写文档，注释

不熟悉的语言

不熟悉的框架

自然语言读代码

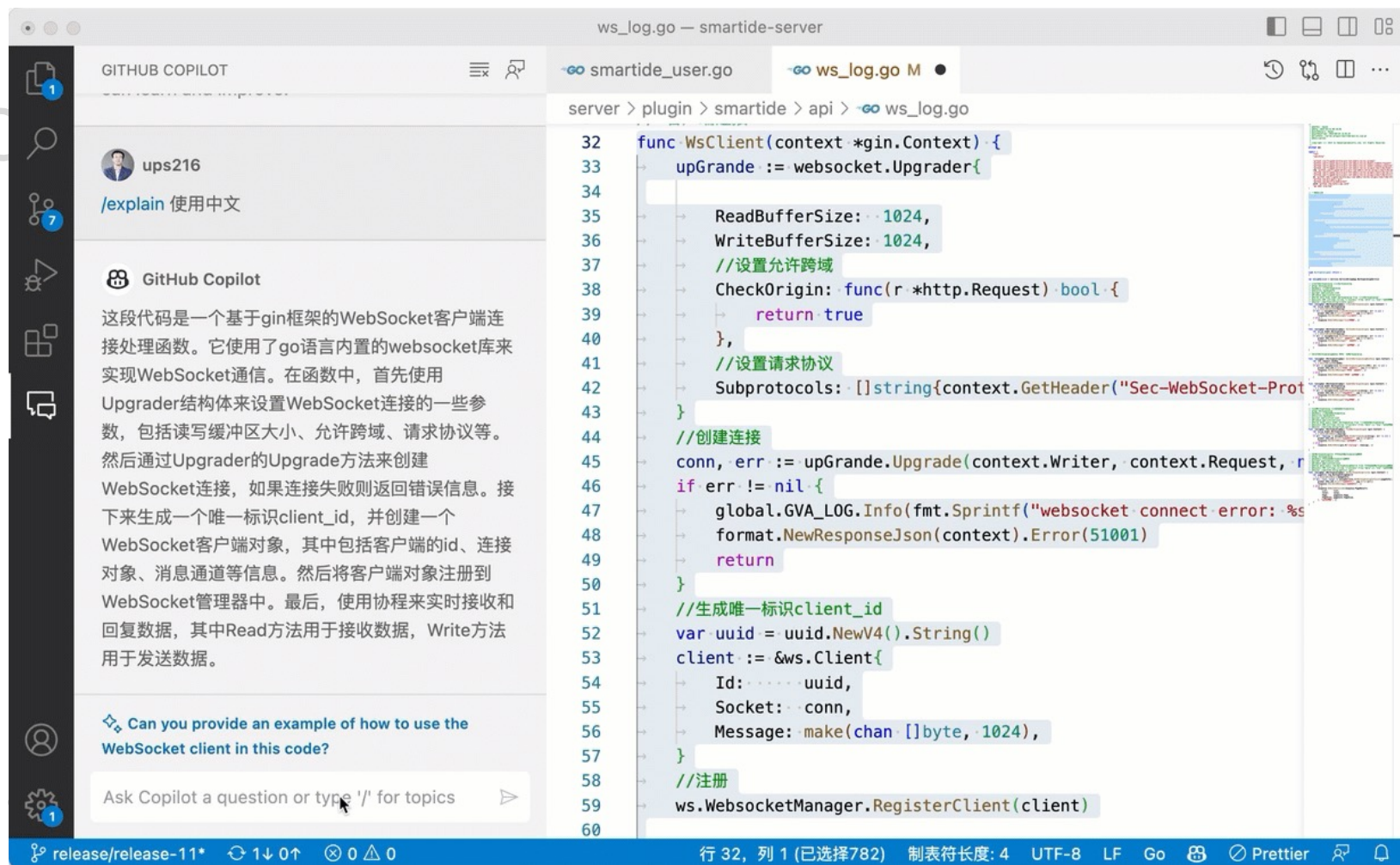


```
server > plugin > smartide > api > -go ws_log.go
32 func WsClient(context *gin.Context) {
33     upGrande := websocket.Upgrader{
34
35         ReadBufferSize: 1024,
36         WriteBufferSize: 1024,
37         //设置允许跨域
38         CheckOrigin: func(r *http.Request) bool {
39             return true
40         },
41         //设置请求协议
42         Subprotocols: []string{context.GetHeader("Sec-WebSocket-Protocol")},
43     }
44     //创建连接
45     conn, err := upGrande.Upgrade(context.Writer, context.Request, nil)
46     if err != nil {
47         global.GVA_LOG.Info(fmt.Sprintf("websocket connect error: %s", context.Param("channel")))
48         format.NewResponseJson(context).Error(51001)
49         return
50     }
51     //生成唯一标识client_id
52     var uuid = uuid.NewV4().String()
53     client := &ws.Client{
54         Id:      uuid,
55         Socket:  conn,
56         Message: make(chan []byte, 1024),
57     }
58     //注册
59     ws.WebsocketManager.RegisterClient(client)
60 }
```

AI=100xH 10.评审代码，提出改进意见

代码改进

代码评审



The screenshot shows the VS Code interface with the GitHub Copilot chat panel on the left and a Go source file on the right.

GitHub Copilot Chat:

- Header: ups216, /explain 使用中文
- Content:

GitHub Copilot

这段代码是一个基于gin框架的WebSocket客户端连接处理函数。它使用了go语言内置的websocket库来实现WebSocket通信。在函数中，首先使用Upgrader结构体来设置WebSocket连接的一些参数，包括读写缓冲区大小、允许跨域、请求协议等。然后通过Upgrader的Upgrade方法来创建WebSocket连接，如果连接失败则返回错误信息。接下来生成一个唯一标识client_id，并创建一个WebSocket客户端对象，其中包括客户端的id、连接对象、消息通道等信息。然后将客户端对象注册到WebSocket管理器中。最后，使用协程来实时接收和回复数据，其中Read方法用于接收数据，Write方法用于发送数据。
- Input: Can you provide an example of how to use the WebSocket client in this code?
- Footer: Ask Copilot a question or type '/' for topics

Go Code (ws_log.go):

```
32 func WsClient(context *gin.Context) {
33     upGrande := websocket.Upgrader{
34
35         ReadBufferSize: 1024,
36         WriteBufferSize: 1024,
37         //设置允许跨域
38         CheckOrigin: func(r *http.Request) bool {
39             return true
40         },
41         //设置请求协议
42         Subprotocols: []string{context.GetHeader("Sec-WebSocket-Prot
43     }
44     //创建连接
45     conn, err := upGrande.Upgrade(context.Writer, context.Request, r
46     if err != nil {
47         global.GVA_LOG.Info(fmt.Sprintf("websocket connect error: %s",
48             format.NewResponseJson(context).Error(51001)
49     }
50     }
51     //生成唯一标识client_id
52     var uuid = uuid.NewV4().String()
53     client := &ws.Client{
54         Id:      uuid,
55         Socket:  conn,
56         Message: make(chan []byte, 1024),
57     }
58     //注册
59     ws.WebsocketManager.RegisterClient(client)
60 }
```

AI 碾压人类的10个编程场景 AI=100xH

- ✓ 正则表达式编写
 - ✓ 编写测试代码，包含各类边界条件验证
 - ✓ 使用难以记忆关键字编写代码，比如：HTML/CSS编写
 - ✓ 编写/不熟悉的复杂算法
 - ✓ 使用/学习不熟悉的编程语言
 - ✓ 按常识完善对象字段
 - ✓ 示例/测试数据生成
 - ✓ 复杂参数填写和上下文匹配
 - ✓ *理解复杂代码并编写文档，注释
 - ✓ *评审代码，提出改进意见
- * GitHub Copilot X 新特性



使用 GitHub Copilot 的开发者发现 ...

88%

... 提高编码效率

87%

... 减少浪费在编写重复性和胶水代码上的时间

77%

... 减少用于搜索问题解决方案的时间

73%

... 更加专注与主线任务

60%

... 更加享受自己的工作

59%

... 编码变得更加有趣

<https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/>

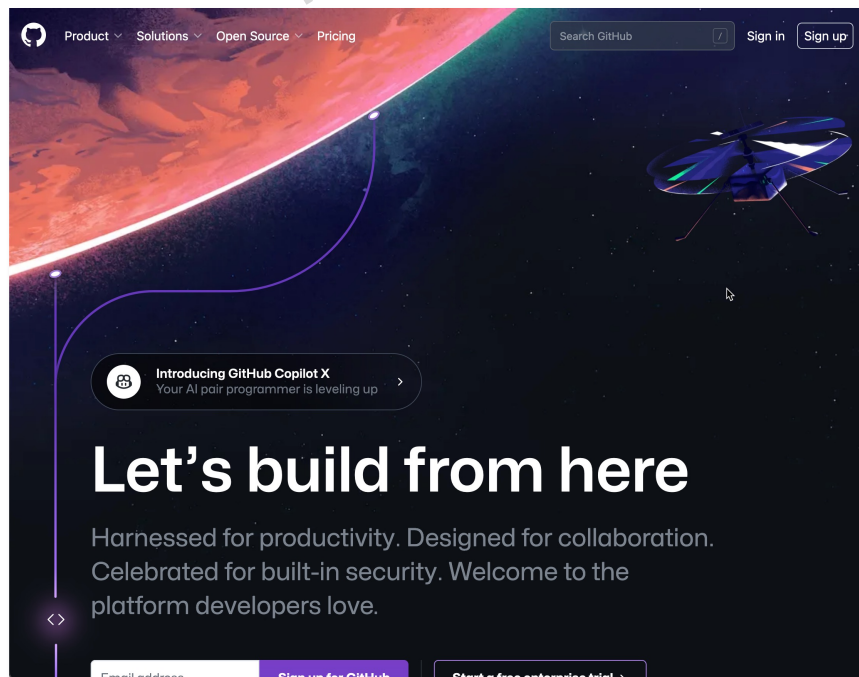


日程

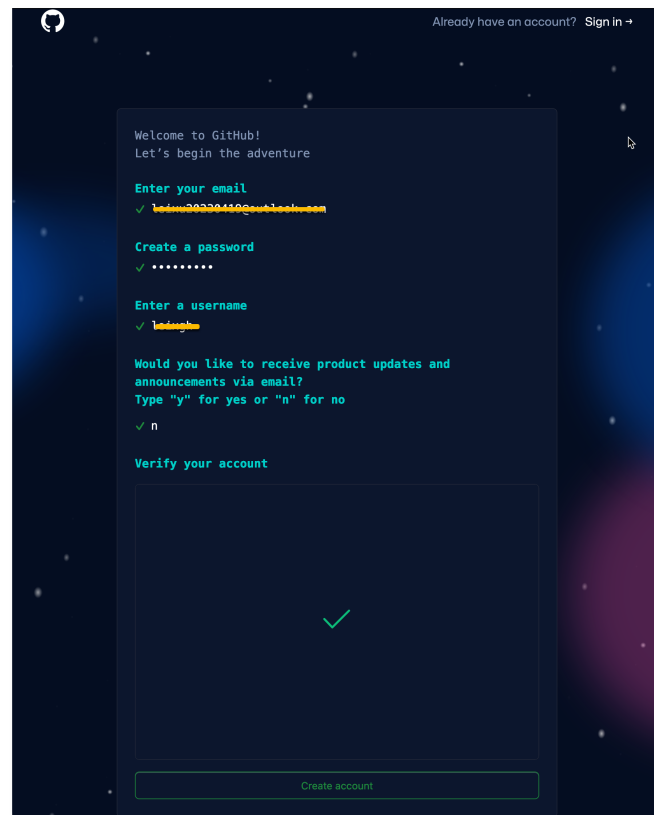
- 概述 – AI碾压人类的10个编程场景
 - GitHub Copilot 账号开通、安装/配置
- AI辅助编程实操演示



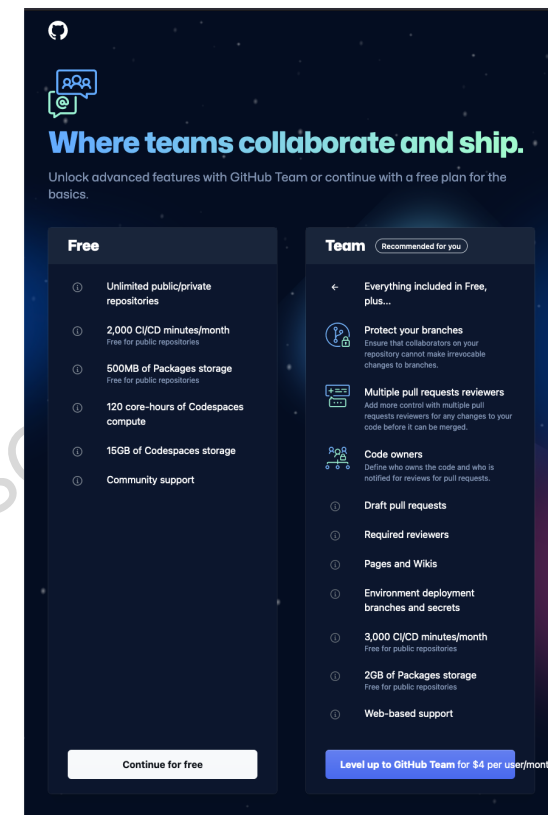
步骤1 – 注册/登录GitHub账号



进入GitHub首页，点击 Sign up（注册）
<https://github.com>

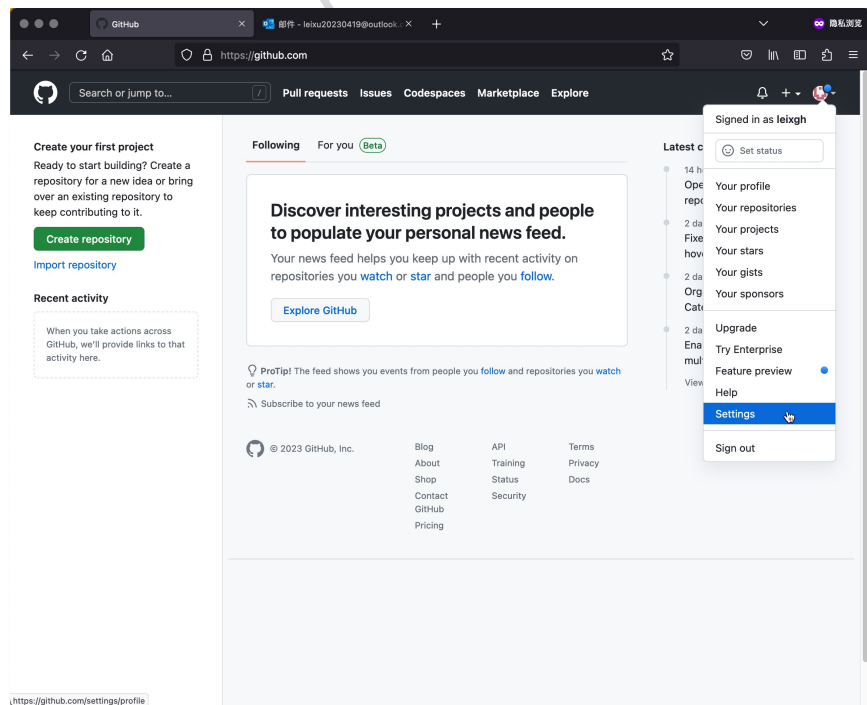


输入邮件地址和密码
 用户名

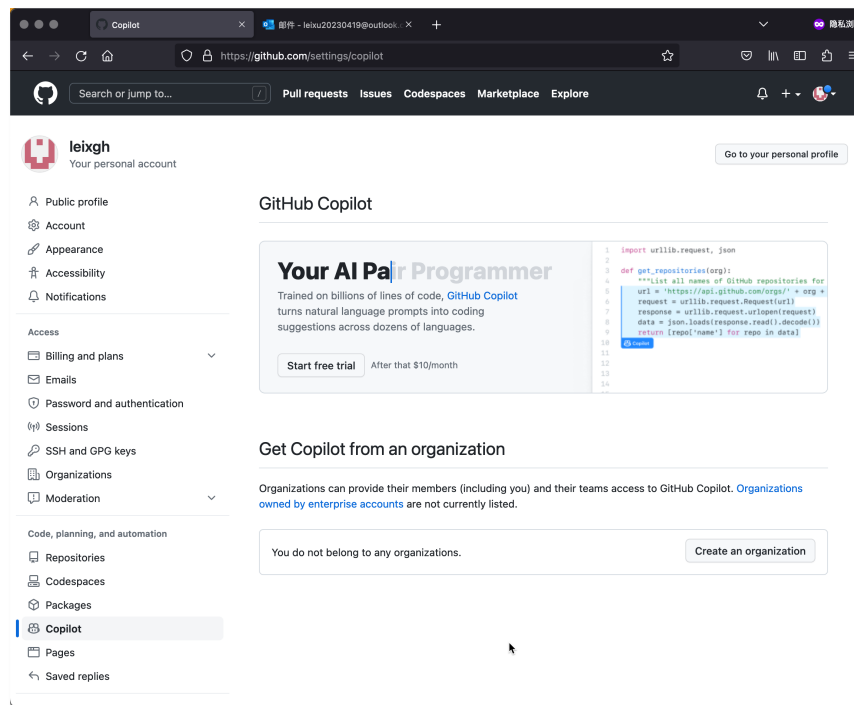


点击 Continue for free

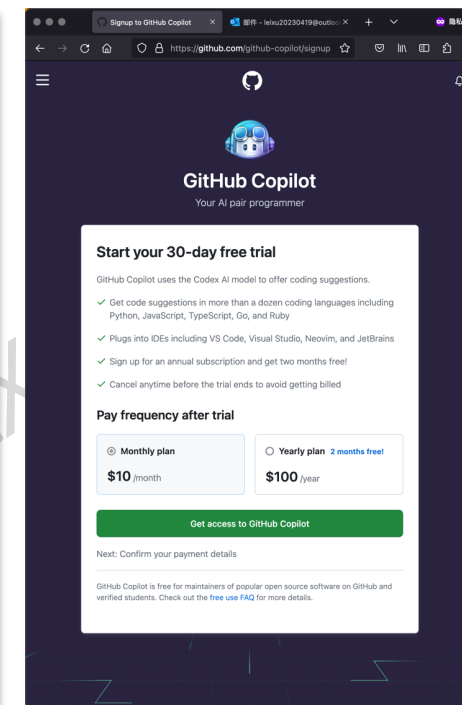
步骤2 – 开通GitHub Copilot



点击 用户名 | Settings

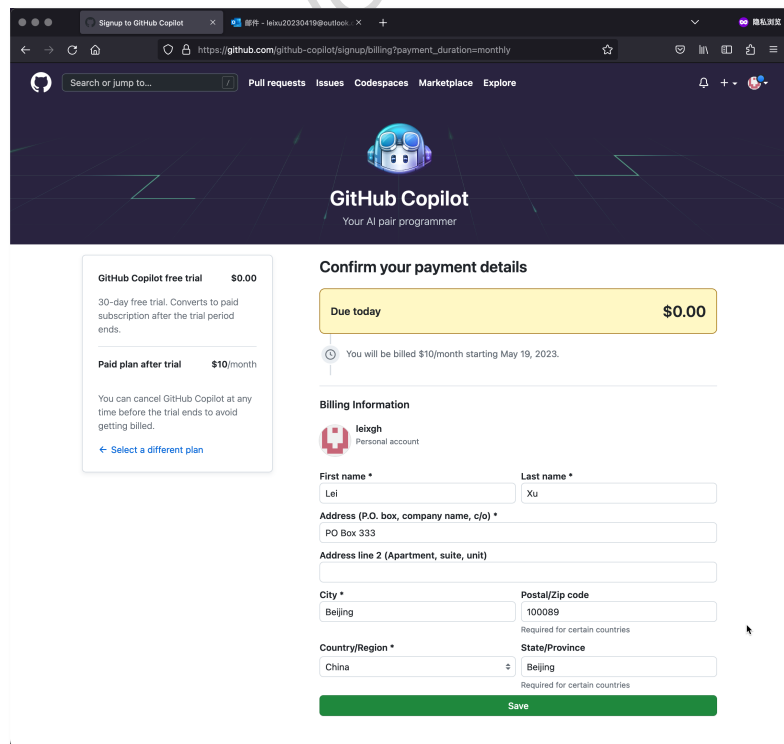


左侧菜单选择 Copilot
点击 Start free trial



选择月付/年付
注册以后免费试用30天

步骤2 – 开通GitHub Copilot



GitHub Copilot free trial \$0.00
30-day free trial. Converts to paid subscription after the trial period ends.

Paid plan after trial \$10/month
You can cancel GitHub Copilot at any time before the trial ends to avoid getting billed.
[Select a different plan](#)

Confirm your payment details

Due today \$0.00

You will be billed \$10/month starting May 19, 2023.

Billing information

lelgh Personal account

First name * Last name *

Lei Xu

Address (P.O. box, company name, c/o) *

PO Box 333

Address line 2 (Apartment, suite, unit)

City * Postal/Zip code

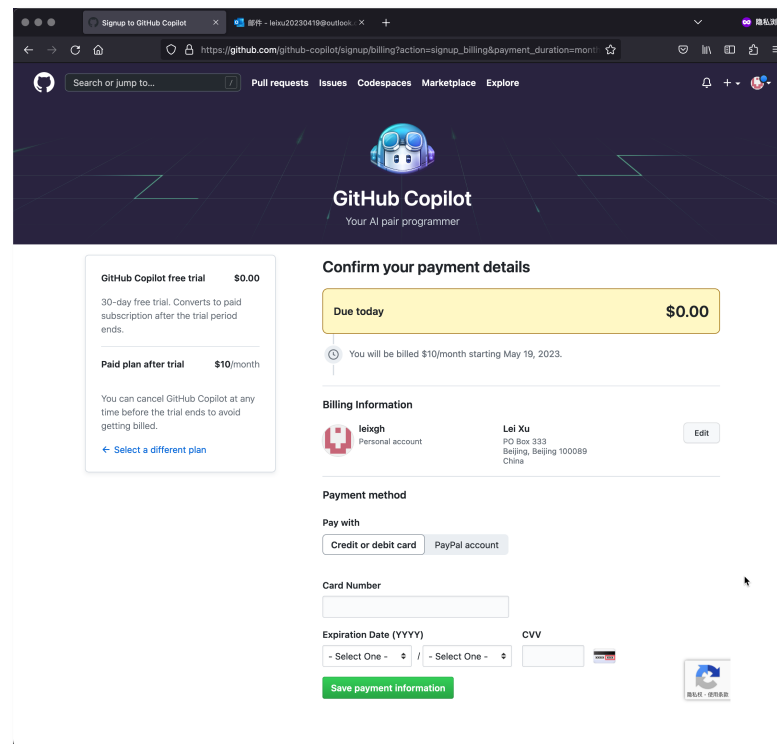
Beijing 100089

Country/Region * State/Province

China Beijing

[Save](#)

填写账单地址



GitHub Copilot free trial \$0.00
30-day free trial. Converts to paid subscription after the trial period ends.

Paid plan after trial \$10/month
You can cancel GitHub Copilot at any time before the trial ends to avoid getting billed.
[Select a different plan](#)

Confirm your payment details

Due today \$0.00

You will be billed \$10/month starting May 19, 2023.

Billing information

lelgh Personal account

Lei Xu
PO Box 333
Beijing, Beijing 100089
China

Payment method

Pay with

☒ Credit or debit card ☐ PayPal account

Card Number

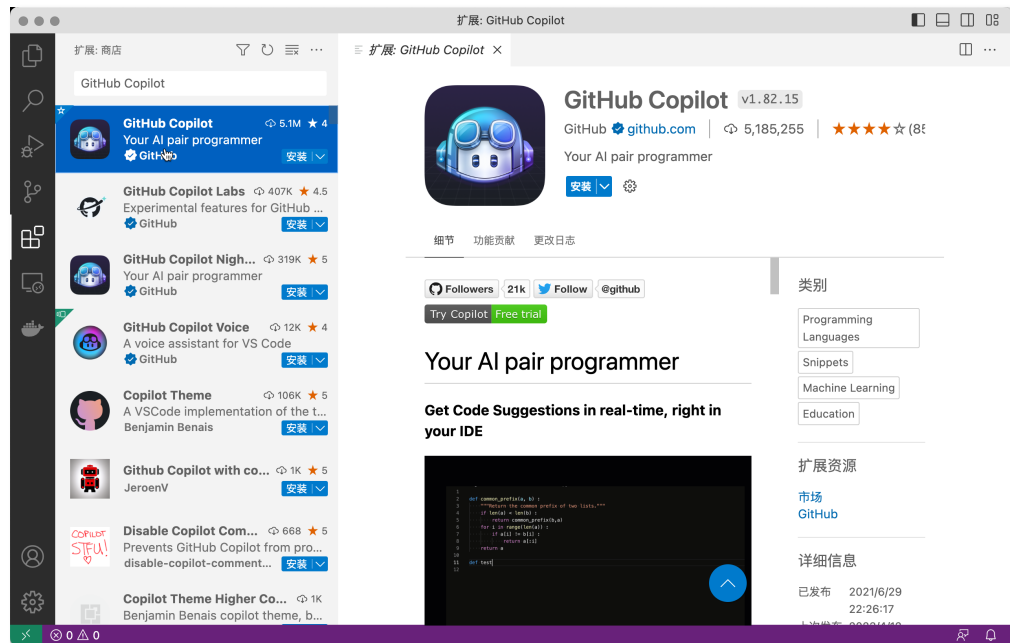
Expiration Date (YYYY) CVV

- Select One - / - Select One -

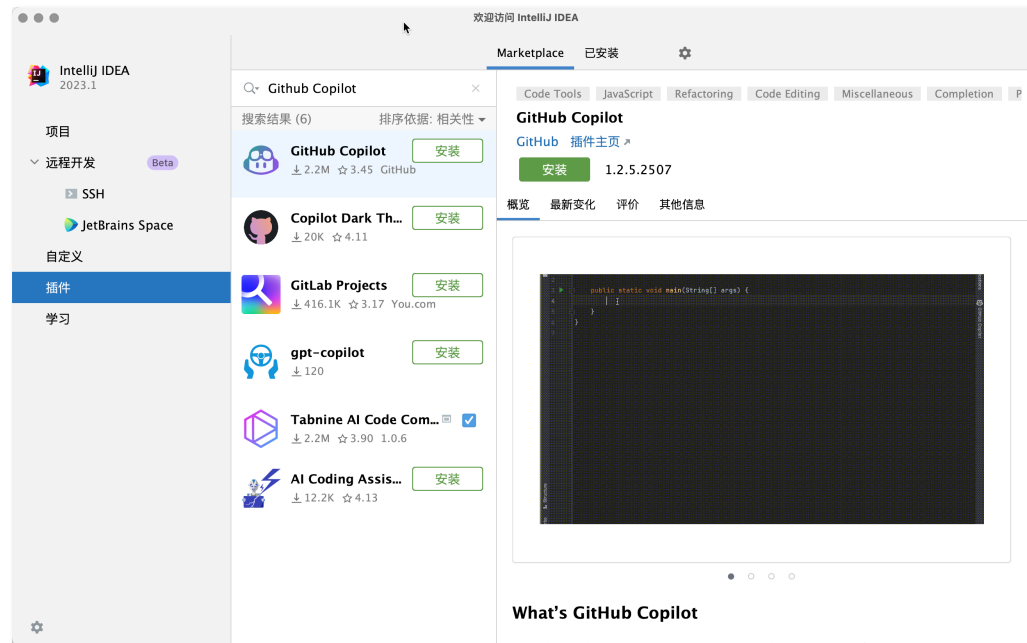
[Save payment information](#)

填写付款方式
信用卡/PayPal
国内信用卡可用

步骤3 - 安装插件

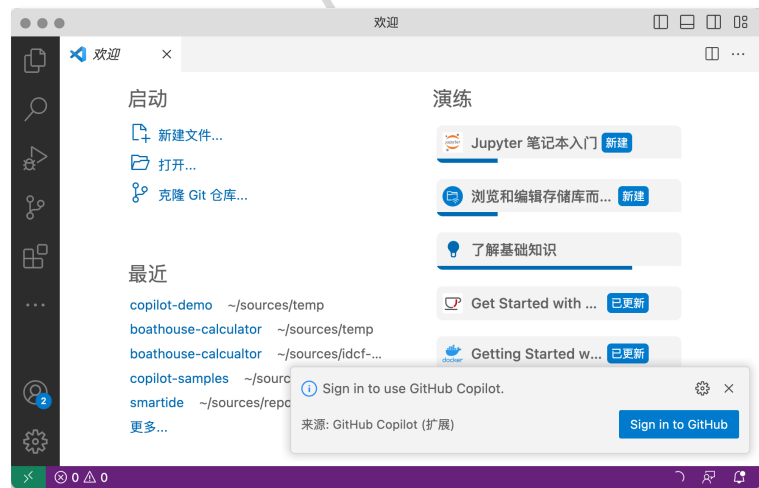


Visual Studio Code

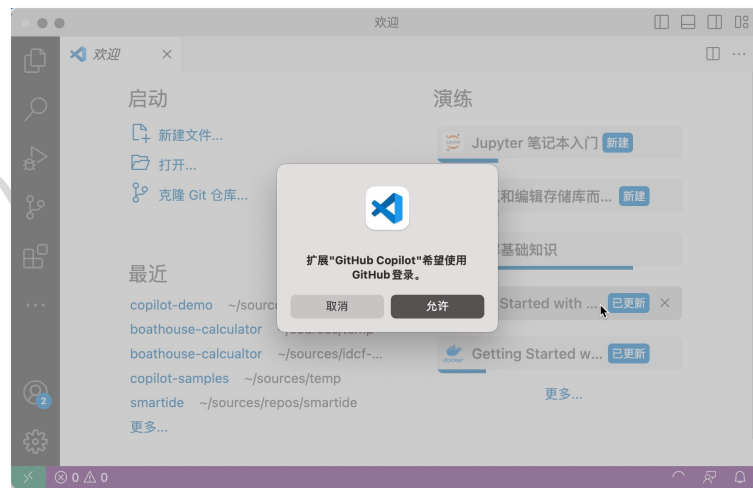


IntelliJ

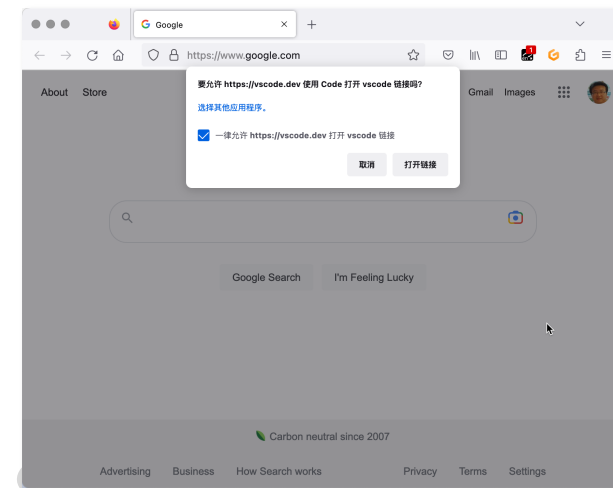
步骤4 – 在VSCode上激活GitHub Copilot



点击 Sign in to GitHub

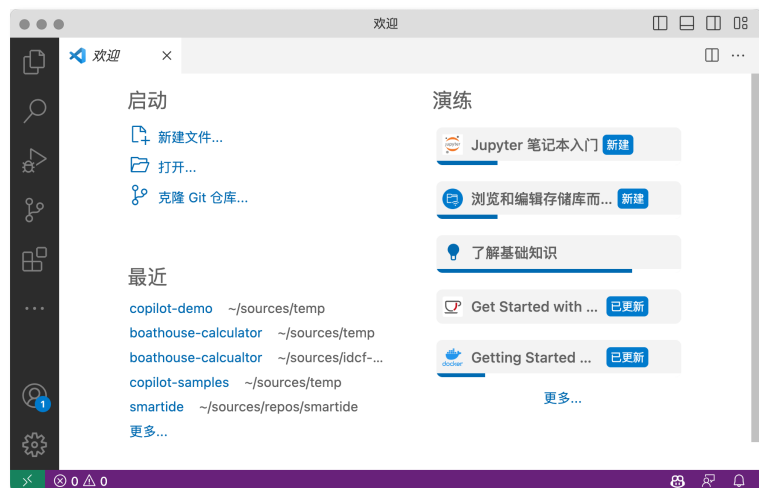


点击 允许



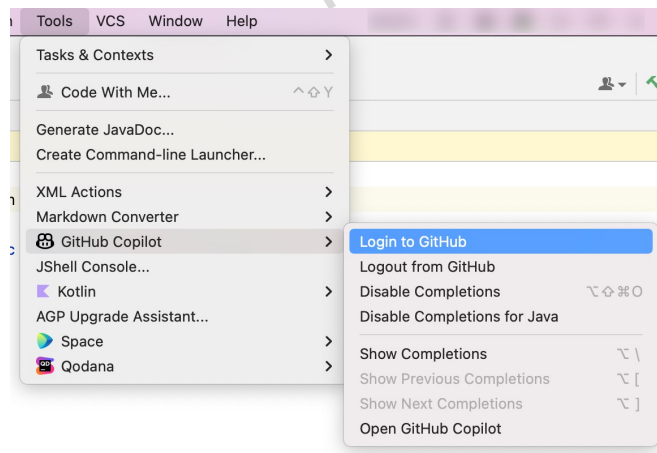
点击 打开链接

备注：某些情况下，这个登录用的浏览器需要科学上网
后续 Github Copilot 试用过程中一般无需科学上网

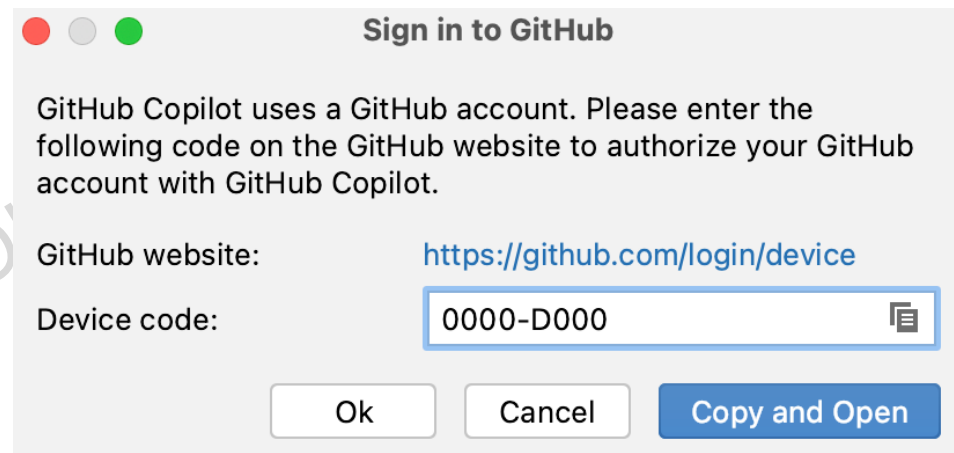


激活成功

步骤4 – 在JetBrains上激活GitHub Copilot



Tools | Github Copilot | Login

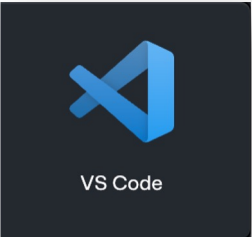


点击 打开链接 <https://github.com/login/device>
从以上界面复制Code到登录页面中

备注：某些情况下，这个登录用的浏览器需要科学上网
后续 Github Copilot 试用过程中一般无需科学上网

快捷键

Tab键接受建议



	下一个/上一个建议	10个建议	逐个Token接受	强制触发
Windows	Alt + []	Ctrl + Enter	Alt + 左右箭头	Alt + \
MacOS	Option/Alt + []	Ctrl + Enter	Alt + 左右箭头	Option (⌘)+\



	下一个/上一个建议	10个建议	逐个Token接受	强制触发
Windows	Alt + []	Ctrl + Enter	Alt + 左右箭头	Alt + \
MacOS	Option/Alt + []	Command+Shift+\	Alt + 左右箭头	Option (⌘)+\

```
// 按用户输入的id下载图片
< 1/2 > 接受 Tab 接受 Word %> → ...
public ResponseEntity<Byte[]> downloadPhoto(String id) {
    Photo photo = photos.get(id);
    return ResponseEntity.ok().header("Content-Disposition", "attachment; filename=\"" +
        .body(photo.getImage());
}
```

Would you like to disable Copilot?

Source: GitHub Copilot (E... Disable Globally Disable for markdown

移动鼠标到代码建议区域，可以激活浮动控制栏

点击图标：禁用Copilot

配置

是否允许推荐和公共代码一致的代码

* 避免开源代码协议问题

VSCode 配置文件：
控制激活状态

Suggestions matching public code

GitHub Copilot can allow or block suggestions matching public code. See [the GitHub Copilot documentation](#) to learn more.

Allowed ▾

Blocked

For this organization, GitHub Copilot will block suggestions matching public code.

✓ Allowed

For this organization, GitHub Copilot will show suggestions matching public code.

```
{
  "editor.inlineSuggest.enabled": true,
  "github.copilot.enable": {
    "*": true,
    "yaml": false,
    "plaintext": false,
    "markdown": true,
    "javascript": true,
    "python": true
  }
}
```

http Proxy 配置 (Business & Enterprise)

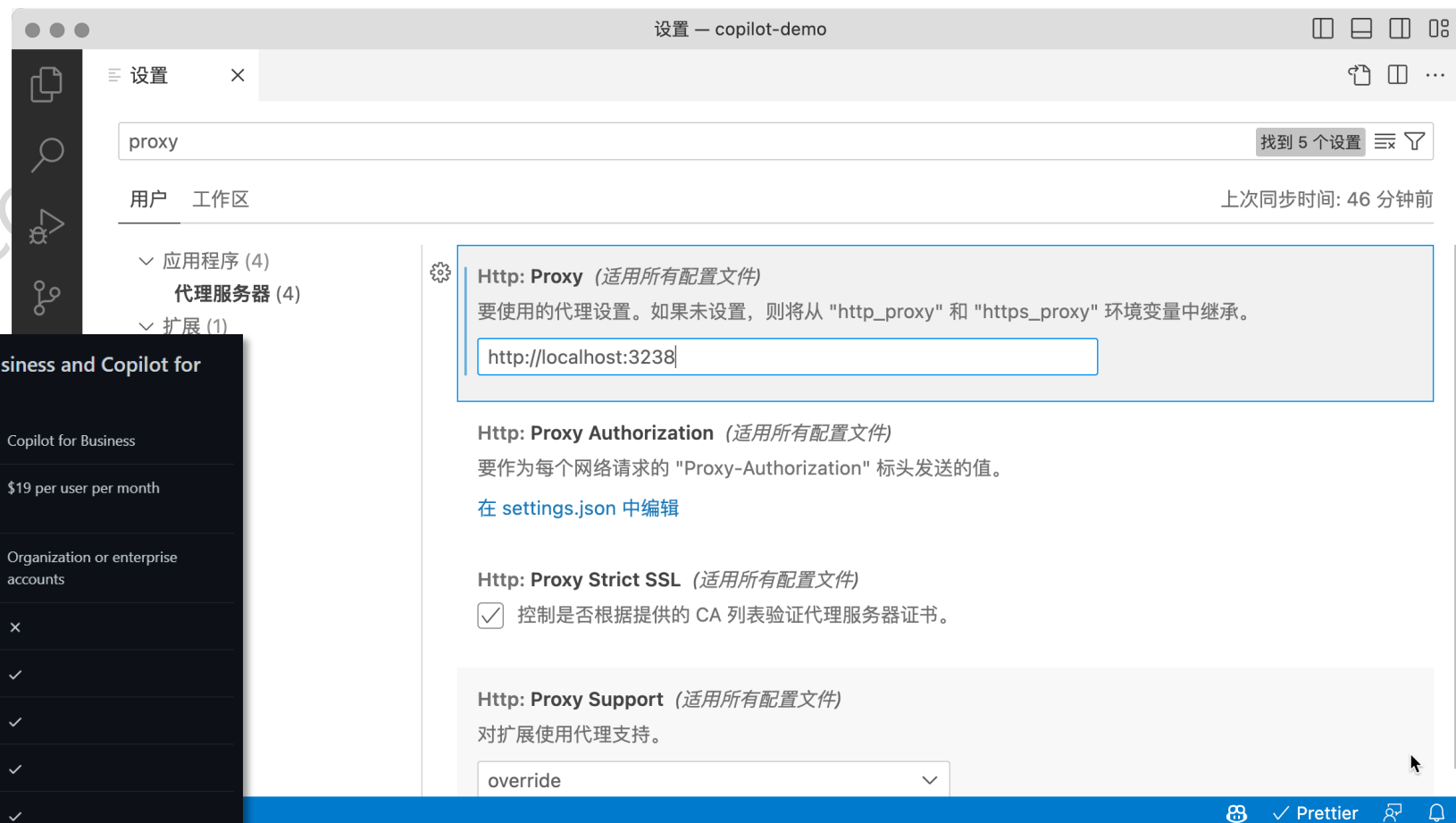
GitHub Copilot
支持使用http proxy中转连接

备注：不支持个人版用户

Understanding the differences between Copilot for Business and Copilot for Individuals

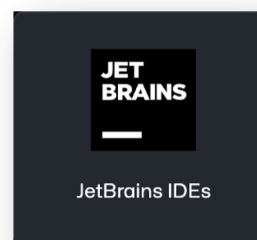
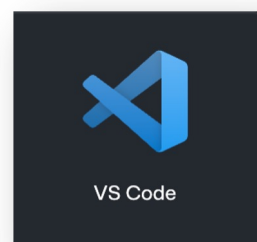
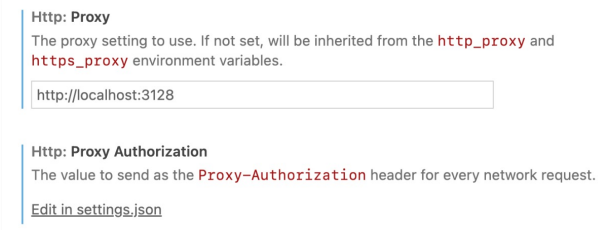
	Copilot for Individuals	Copilot for Business
Pricing	\$10 per month/\$100 per year	\$19 per user per month
Types of GitHub accounts	Personal accounts	Organization or enterprise accounts
Telemetry	✓	✗
Blocks suggestions matching public code	✓	✓
Plugs right into your editor	✓	✓
Offers multi-line function suggestions	✓	✓
Organization-wide policy management	✗	✓
VPN Proxy support via self-signed certificates [1]	✗	✓

[1] For more information, see "[Configuring GitHub Copilot in Visual Studio Code](#)" and "[Configuring GitHub Copilot in a JetBrains IDE](#)."



隔离通过代理服务网络访问Copilot

隔离网络



PROXY

Github 服务白名单

<https://github.com/login/device/code>
https://github.com/login/oauth/access_token
<https://api.github.com/user>
https://api.github.com/copilot_internal/v2/token
https://api.github.com/copilot_internal/notification
<https://default.exp-tas.com>
<https://copilot-telemetry.githubusercontent.com/telemetry>
<https://copilot-proxy.githubusercontent.com>

GitHub Copilot的数据保护策略



[Site policy](#) / [Privacy Policies](#) /

GitHub Copilot for Business Privacy Statement

Effective Date: December 7, 2022

Use of GitHub Copilot for Business is also subject to the [GitHub Privacy Statement](#), the [Acceptable Use Policies](#) and the [GitHub Copilot for Business license terms](#).

What data does Copilot for Business collect?

Copilot for Business relies on file content and additional data to work. It collects data to provide the service, some of which is then saved for further analysis and product improvements. Copilot for Business collects data as described below:

User Engagement Data

When you use Copilot for Business it will collect usage information about events generated when interacting with the IDE or editor. These events include user edit actions like completions accepted and dismissed, and error and general usage data to identify metrics like latency and features engagement. This information may include personal data, such as pseudonymous identifiers.

Code Snippets Data

GitHub Copilot transmits snippets of your code from your IDE to GitHub to provide Suggestions to you. Code snippets data is only transmitted in real-time to return Suggestions, and is discarded once a Suggestion is returned. Copilot for Business does not retain any Code Snippets Data.

How is data in Copilot for Business used and shared?

User Engagement Data is used by GitHub, Microsoft, and OpenAI to provide the service and to enable improvements.

Such uses may include:

- Evaluating GitHub Copilot, for example, by measuring the positive impact it has on the user
- Fine tuning ranking and sorting algorithms and prompt crafting
- Detecting potential abuse of GitHub Copilot or violation of [Acceptable Use Policies](#).
- Conducting experiments and research related to developers and their use of developer tools and services.

How can users of Copilot for Business control use of their data?

User engagement data (which includes pseudonymous identifiers and general usage data), is required for the use of GitHub Copilot and will continue to be collected, processed, and shared with Microsoft and OpenAI when you use GitHub Copilot.

For more information on how GitHub processes and uses personal data, please see the [GitHub Privacy Statement](#).

<https://github.com/features/copilot/#faq-privacy-copilot-for-business>

<https://docs.github.com/en/site-policy/privacy-policies/github-copilot-for-business-privacy-statement>

日程

- 概述 – AI碾压人类的10个编程场景
- GitHub Copilot 账号开通、安装/配置
- AI辅助编程实操演示

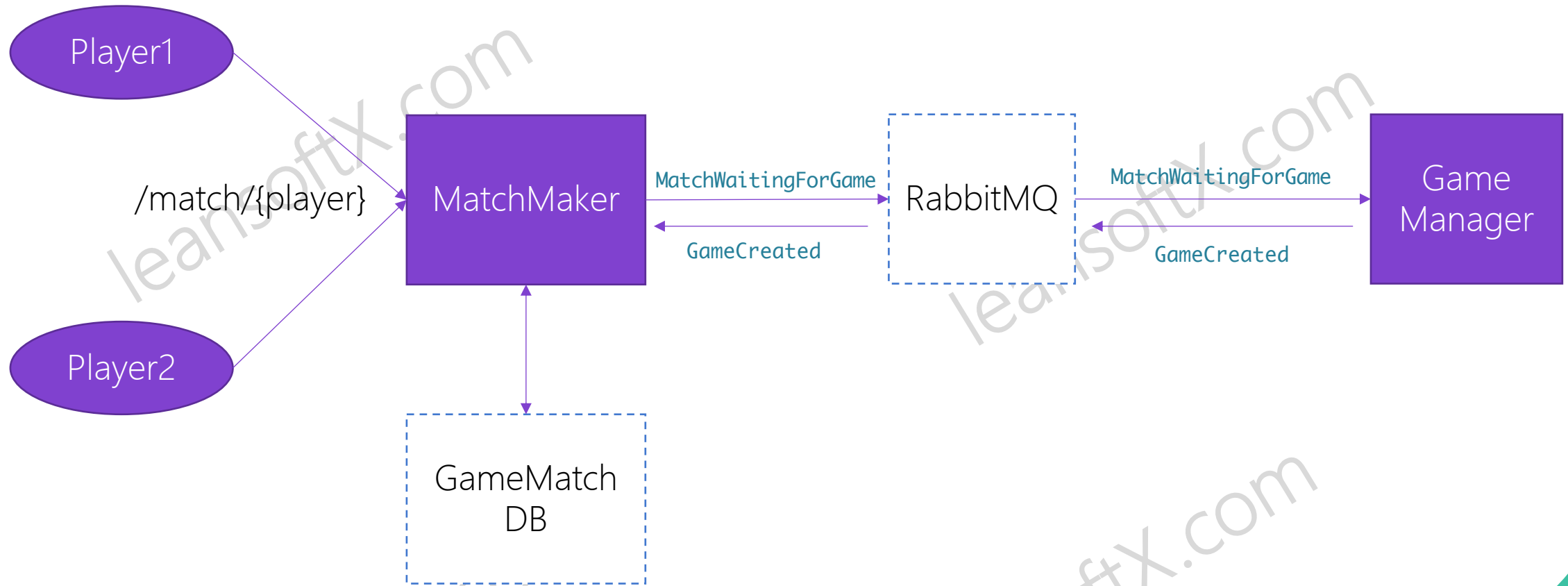




GitHub Copilot完成一个 Java SpringBoot API服务开发



使用ChatGPT和GitHub Copilot完成一个微服务架构的游戏匹配服务开发



leansoftX

谢谢

版权所有 leansoftX.com

leansoftX

