Cloud Studio（云端 IDE）

代码编辑

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

代码编辑
基本操作
布局介绍

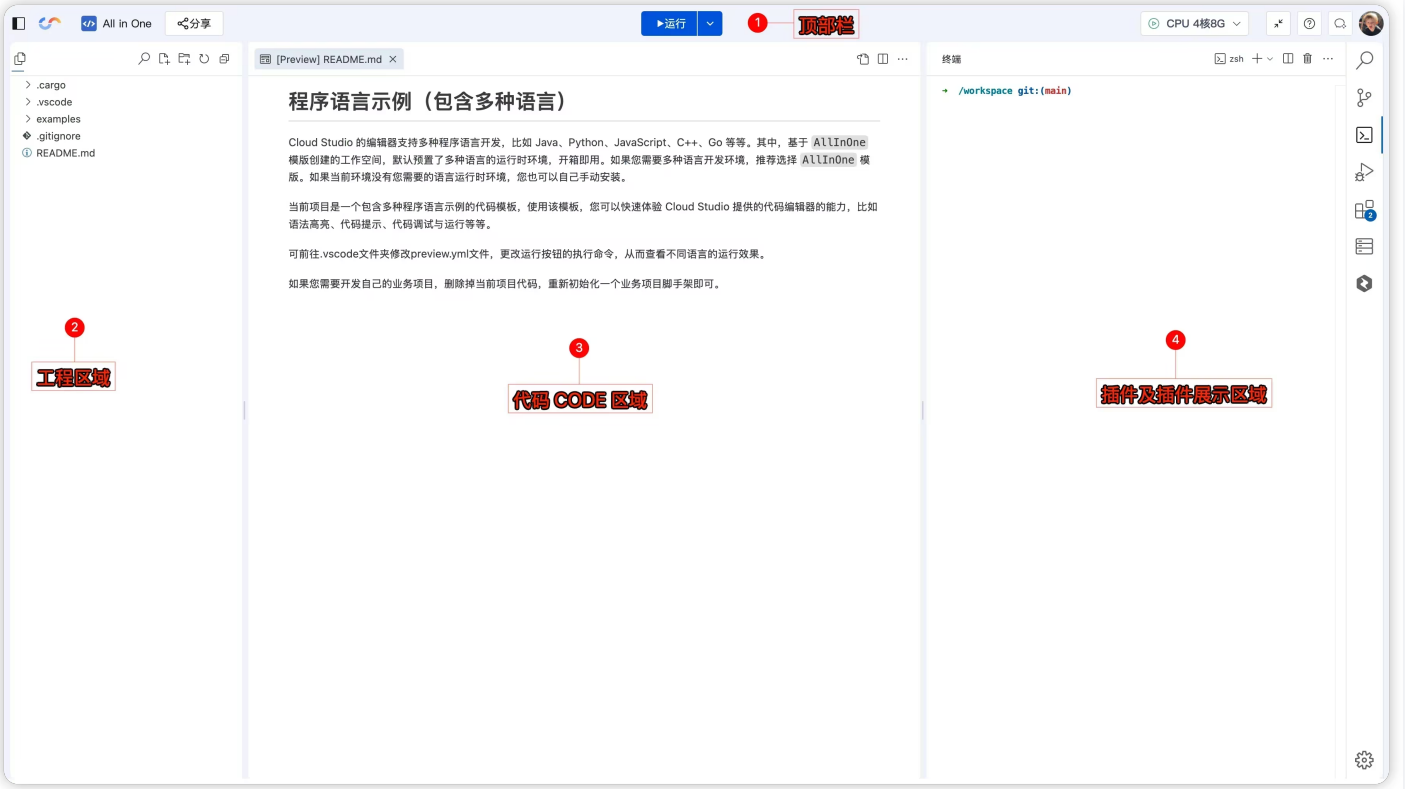
最近更新时间：2026-05-19 18:07:22

概述

CloudStudio 常用两种界面：**应用模式（开发）**与 **教学模式（学习）**。两者布局相近，但教学模式会多一个“教案”区域，方便按课程学习。

一、应用模式（开发）

适合日常项目开发，主要包含 4 个区域：

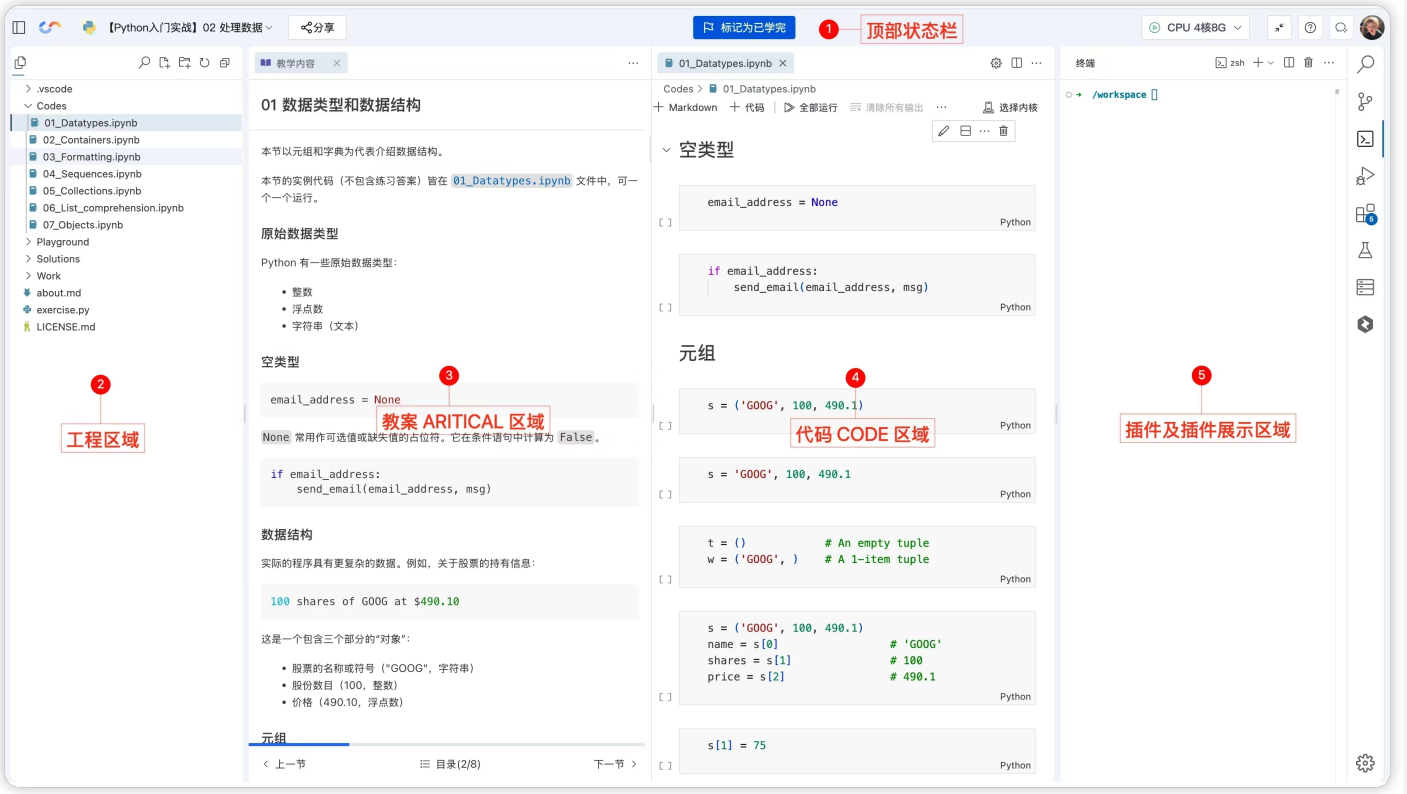


区域	主要用途
顶部栏	全局操作：菜单、分享、帮助、窗口/算力等
工程区域	管理项目文件与资源（常用筛选/同步入口）
代码区域	编写与编辑代码

插件区域	使用 Git、终端、调试、扩展、AI 助手等工具

二、教学模式（学习）

适合跟课程学习，主要包含 5 个区域：



区域	主要用途
顶部栏	章节切换、进度查看、帮助、窗口/算力等
工程区域	查看学习用代码文件（通常按语言/章节组织）
教案区域	阅读讲解与示例代码（支持目录跳转/复制）
代码区域	完成练习、运行与验证
插件区域	使用 Web 预览、测试、AI 助手等辅助工具

三、通用：调整布局大小

1. 鼠标移到任意两个区域之间的分隔线。
2. 指针变为「⟷」后，按住鼠标左键拖动。
3. 松开鼠标完成调整。


模块五：建立你的AI助手
CPU 4核8G
标记为已学完
分享
.devcontainer
module-0
module-1
module-2
module-3
module-4
module-5
module-6
.gitignore
all.md
README.md
requirements.txt
start.md
教学内容
5.2 并行化
右侧Jupyter Notebook互动教学指引
基础环境认知
教程由可执行代码与文字教程组成，点击运行单元格即可在下方查看代码运行的结果
1. 界面布局认知
右侧代码区示例（建议同步操作）
例:
# 点击▶运行查看直方图
import matplotlib.pyplot as plt
plt.hist([1,5,3,7,2,8])
上一节
目录(2/5)
下一节
parallelization.ipynb M
module-4 > parallelization.ipynb
代码
Markdown
全部运行
清除所有输出
大纲
选择内核
Open in Colab
LangChain Academy
并行节点执行
回顾
在模块 3 中，我们深入讨论了 "人在循环中"，展示了 3 种常见的使用案例：
(1) "批准"--我们可以中断智能体，将状态显示给用户，并允许用户接受某个操作
(2) "调试"--我们可以倒退图形以重现或避免问题的出现
(3) "编辑"--可以修改状态
目标
本模块将以模块 2 中讨论的 "人在回路中 "和 "记忆 "概念为基础。
我们将深入研究 "多智能体 "工作流程，并建立一个多智能体研究助手，将本课程的所有模块串联起来。
为了建立这个多智能体研究助手，我们将首先讨论一些 LangGraph 可控性主题。
我们将从并行化开始。
扇出和扇入
让我们构建一个简单的线性图，每一步都会重写状态。
通过DeepL.com（免费版）翻译
%%capture --no-stderr
%pip install -U langgraph tavily-python wikipedia langchain_openai langchain_c
Python
import os, getpass
终端
zsh
/workspace git:(main) X

算力切换

最近更新时间：2026-05-19 18:07:22

概述

CloudStudio 支持在工作空间内切换 CPU / GPU，以适配不同任务。

一、前提：进入工作空间

先进入需要操作的工作空间。

- 还没有工作空间？可参考：应用创建

二、切换 CPU / GPU

1. 在工作空间 **顶部栏右侧** 找到「**算力**」按钮并单击。

2. 在弹窗中选择需要的类型：**CPU** 或 **GPU**。

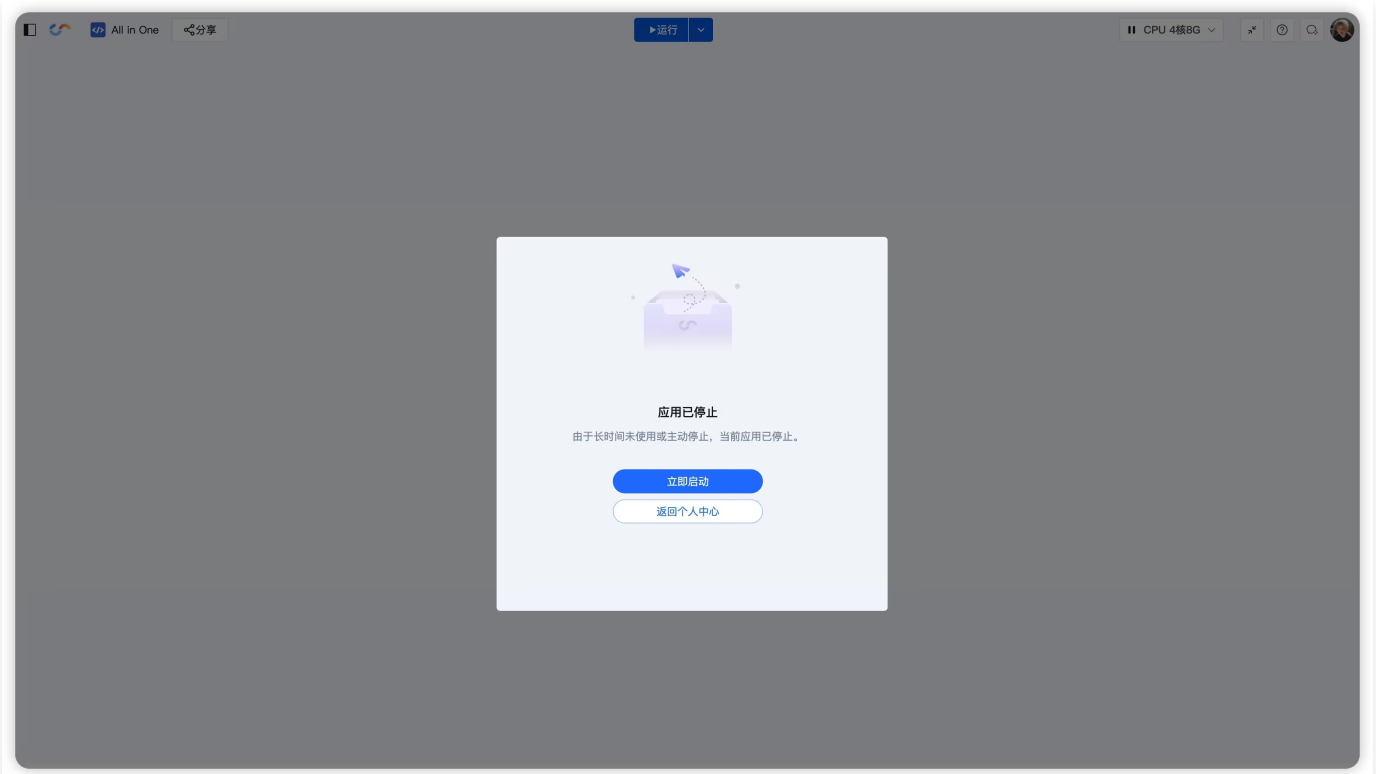
3. 等待切换完成（一般十秒左右）。

选择建议	适合场景
CPU	日常开发、前端/后端、普通脚本与编译构建
GPU	深度学习训练、需要 GPU 加速的计算任务

三、「停止」按钮

在「算力」按钮右侧的「停止」用于 关闭当前工作空间算力。

- **关闭后，工作空间将停止运行，并出现提示。**



- 需要继续使用时，按提示单击「**立即启动**」即可。

使用提示

- **避免连点**：切换过程中不要重复单击，防止触发多次请求。
- **短暂加载属正常**：切换时页面可能短暂刷新/加载。

配置运行文件

最近更新时间：2026-05-19 18:07:22

概述

本页为您介绍如何通过 `preview.yml` 配置项目的**启动命令与预览端口**。

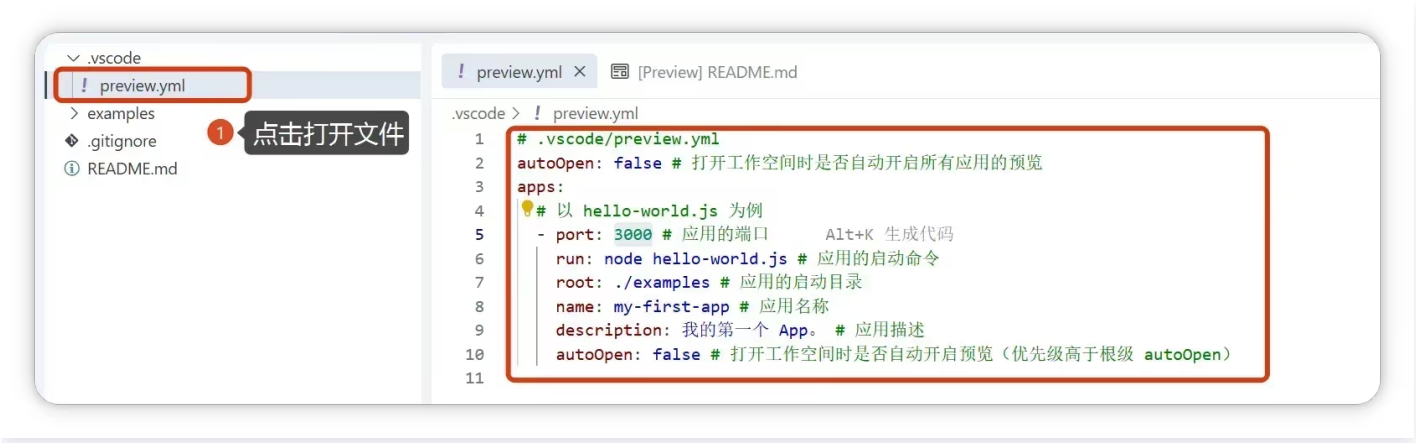
一、`preview.yml` 在哪？做什么？

- **路径**：项目根目录的 `.vscode/preview.yml`（需要显示隐藏文件）。
- **作用**：定义项目启动命令（ `run` ）、工作目录（ `root` ）、端口（ `port` ）以及多服务配置。

二、如何打开

方式 1

左侧工程区 > `.vscode` > `preview.yml`



方式 2

顶部「运行」下拉 >「查看配置文件」

三、port 规则

- 在 `apps` 中 **必须保留** `port:`，不要删除。

- `port:` **可以留空：系统会自动识别项目默认端口；若端口冲突会自动分配新端口。**

> **正确写法示例**
> 保留 `port:`（冒号后不填数字）。

四、常用配置示例

前端项目

```
# 指定端口
apps:
  - port: 8080
    run: npm run dev
    root: ./frontend
    name: 前端应用

# 自动端口（留空）
apps:
  - port:
    run: npm run dev
    root: ./frontend
    name: 前端应用
```

后端项目

```
apps:
  - port:
    run: flask run
    root: ./backend
    name: 后端服务
```

五、设置主预览端口

> **注意**
> 仅在"应用主页预览"场景下生效。

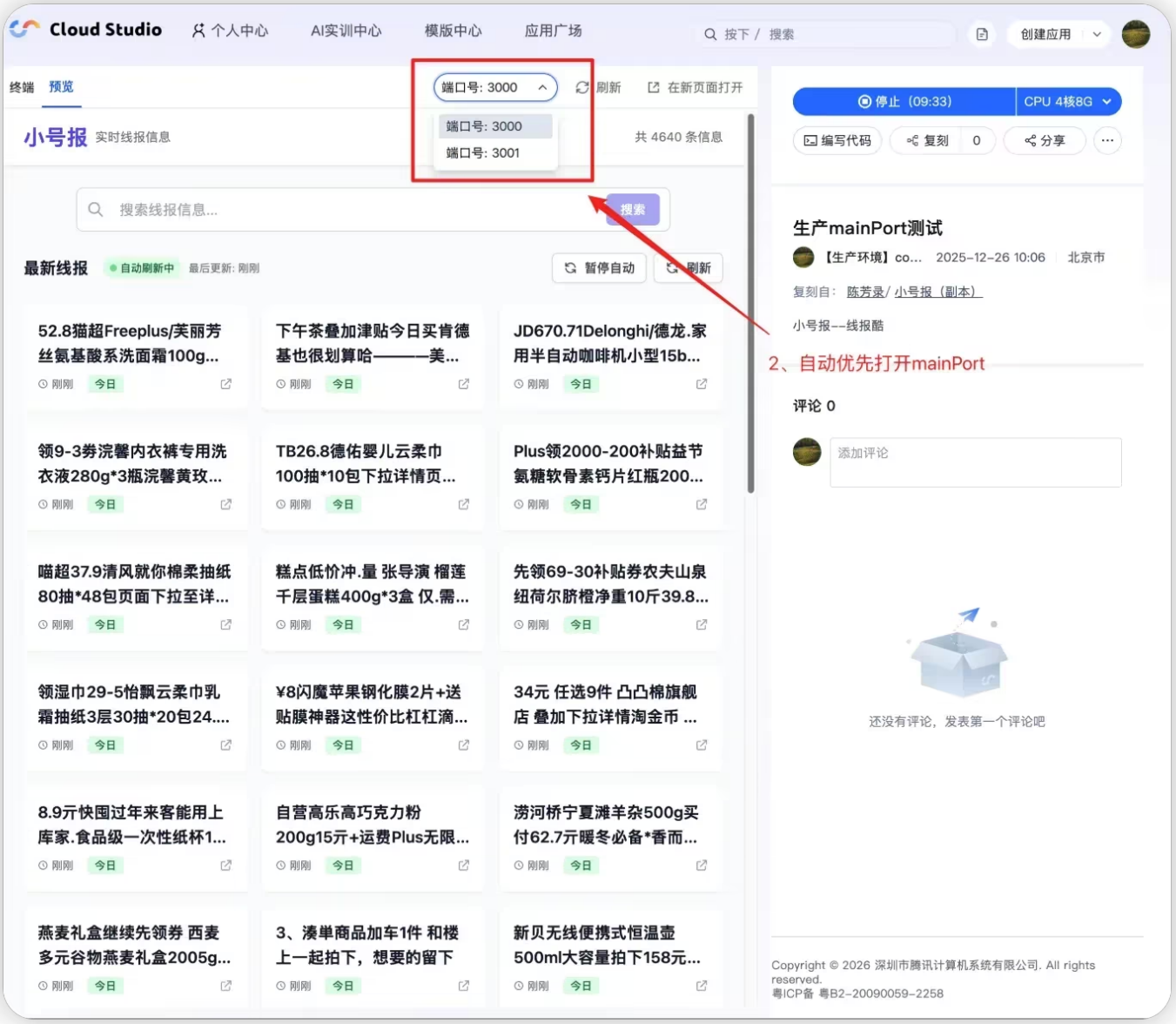
1. 打开 `preview.yml`
2. 在目标服务配置里加上 `mainPort: true`

[Preview] README.md　! preview.yml M ×

.vscode > ! preview.yml

```yaml
autoOpen: true # 打开工作空间时是否自动开启所有应用的预览
apps:
  - port: 3001 # 应用的端口
    run: echo
    root: ./ # 应用的启动目录
    name: leo的基础模版 # 应用名称
    description: leo的基础模版。 # 应用描述
    autoOpen: false # 打开工作空间时是否自动开启预览（优先级高于根级 autoOpen）
    autoPreview: false
  - port: 3000 # 应用的端口
    run: | # 应用的启动命令
      kill -9 $(lsof -t -i :3001)
      kill -9 $(lsof -t -i :3000)
      pkill -f "node.*server.js" && pkill -f "vite"
      # uv venv --python 3.12 --seed
      # chmod +x ./update && ./update
      # git fetch origin main && git reset --hard origin/main
      cd /workspace/xiaohao-bao && ./run-cloudstudio.sh
      # git pull origin main
      # source .venv/bin/activate
      # uv pip install vllm --torch-backend=auto
      # pip install --upgrade vllm --torch-backend=auto && pip install open-webui && pip --upgrade open-webui
      # uv pip install open-webui && uv pip --upgrade open-webui
      # vllm serve tencent/Hunyuan-MT-7B
      # 延时10秒 sleep 10
      # 设置环境变量，获取当前部署url的地址 export LLM_API_BASE="https://${X_IDE_SPACE_KEY}--11434.${X_IDE_SPACE_REGION}.cloudstudio.club/"
      # open-webui serve
    root: ./ # 应用的启动目录
    name: leo的基础模版 # 应用名称
    description: leo的基础模版。 # 应用描述
    autoOpen: false # 打开工作空间时是否自动开启预览（优先级高于根级 autoOpen）
    autoPreview: true
    mainPort: true
```

3. 回到应用浏览页点击「启动」，会优先打开设置为主端口的页面



六、参数速查

参数	说明	示例
`port`	必须保留；留空自动分配端口	`8080` / 留空
`run`	启动命令	`npm run dev`
`root`	启动命令工作目录	`./frontend`
`autoOpen`	是否自动运行服务	`true` / `false`
`autoPreview`	是否自动打开预览窗口	`true` / `false`
`mainPort`	是否为主预览端口	`true` / `false`

常见问题

- **运行失败：**确认 `apps` 中存在 `port:` 字段（不可删除）。
- **端口冲突：**把 `port:` 留空让系统自动分配，或手动改为其他端口（如 8081）。
- **启动失败/找不到目录：**检查 `run` 命令与 `root` 目录是否与项目实际一致。

文件与Git管理
文件上传/下载

最近更新时间：2026-05-19 18:07:22

概述

本页介绍 CloudStudio 中**上传文件/文件夹**与**下载文件**的常用方法。

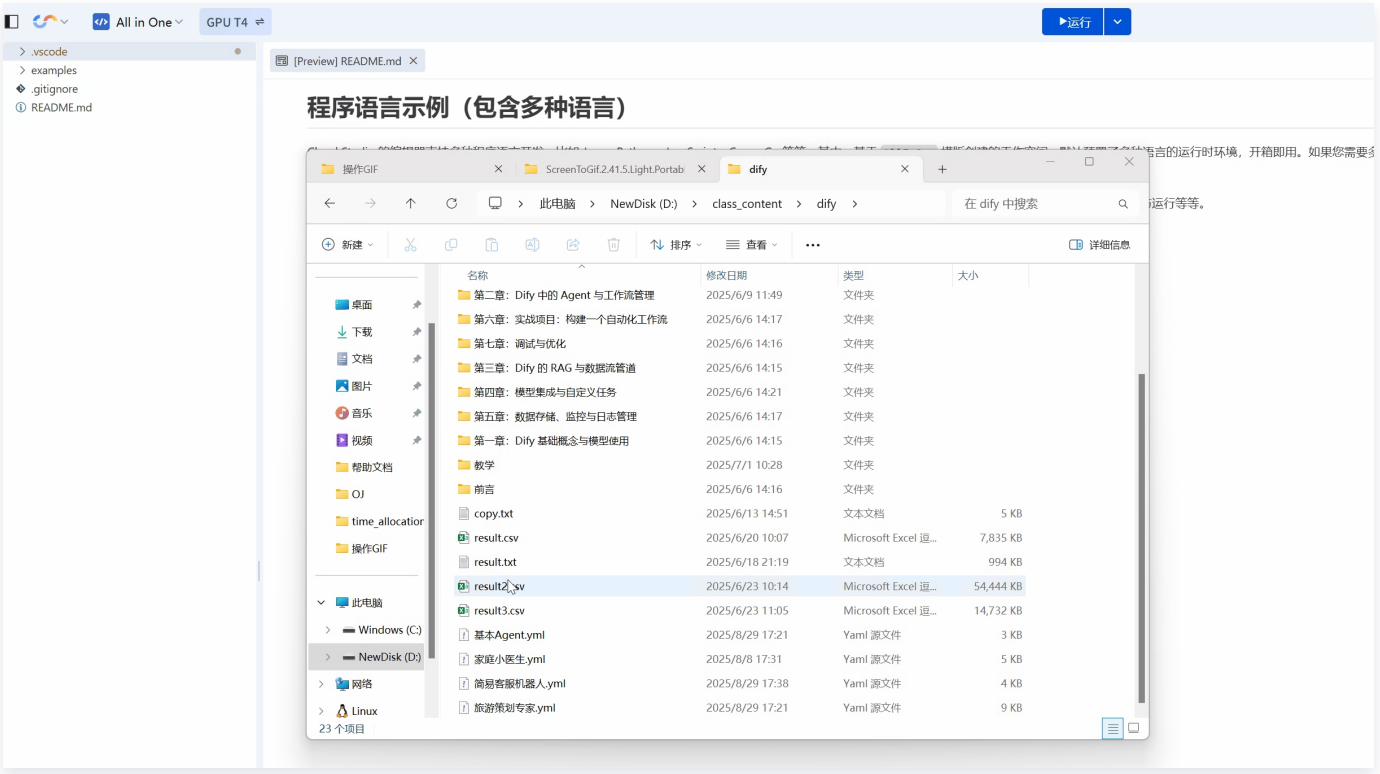
一、上传文件

支持两种方式：

- **拖拽上传**：支持文件与文件夹（推荐）
- **复制粘贴**：仅支持文件

方式 1：拖拽上传（推荐）

1. 在本地找到要上传的文件/文件夹。
2. 将其拖到 CloudStudio 左侧「**工程区域**」的目标目录。
3. 松开鼠标后自动开始上传，上传完成即可在目录中看到文件。



方式 2：复制粘贴上传（仅文件）

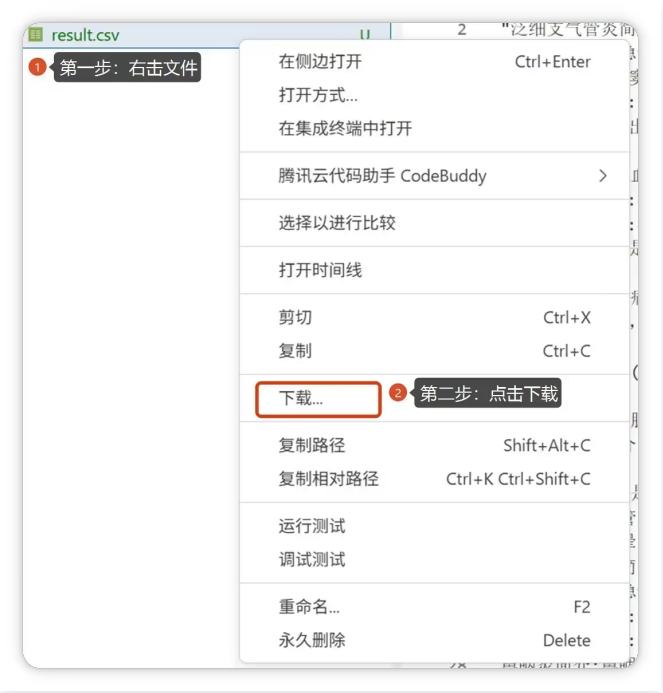
1. 在本地复制文件：`Ctrl+C`（Windows）/ `Command+C`（Mac）。

2. 在 CloudStudio「**工程区域**」选中目标目录。
3. 粘贴上传：`Ctrl+V`（Windows）/ `Command+V`（Mac）。

二、下载文件

目前仅支持**单个文件下载**（不支持直接下载文件夹）。

1. 在「**工程区域**」找到目标文件并右键。
2. 单击「**下载**」。
3. 浏览器开始下载，默认保存到本地下载目录。



下载文件夹（打包为 ZIP）

如需下载文件夹，建议先在工作空间内把文件夹打包成 ZIP，再下载 ZIP 文件。

- **示例**：在终端执行（将 `my-folder` 替换为你的文件夹名）：
  - `zip -r my-folder.zip my-folder`
- 打包完成后，在工程区域右键 `my-folder.zip` >「**下载**」。

三、注意事项

- **大文件上传**：建议使用稳定网络；必要时先压缩后上传（再在工作空间内解压）。若文件可公网访问，也可在终端使用 `wget` / `curl` 下载。
- **大文件下载**：目前已支持大文件下载。如下载失败，建议先压缩/拆分文件。

效率插件
代码搜索

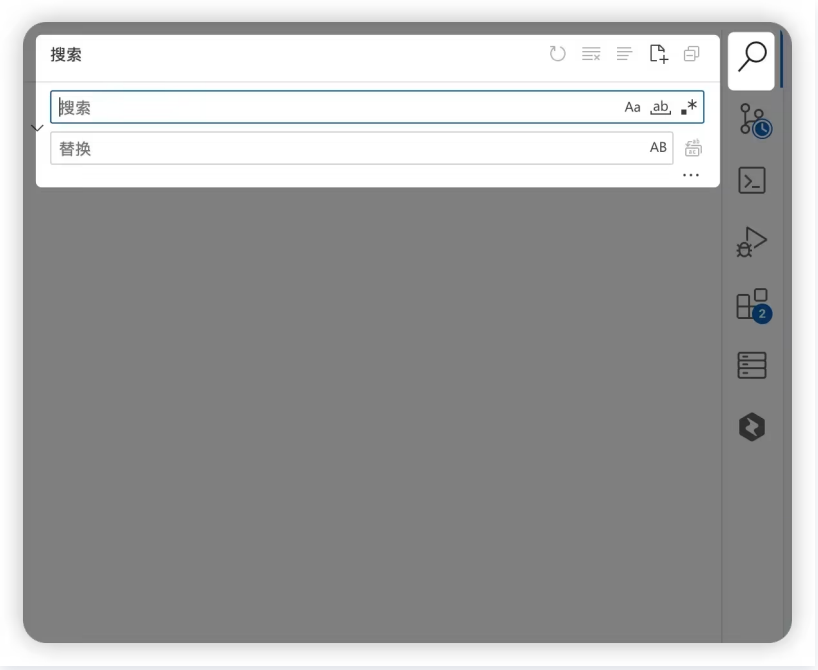
最近更新时间：2026-05-19 18:07:22

概述

CloudStudio 支持在**当前文件**或**整个项目**中搜索内容，并可配合替换与正则提升效率。

一、入口与快捷键

- **打开全局搜索面板**：左侧插件栏 >「**代码搜索**」



- **常用快捷键**：

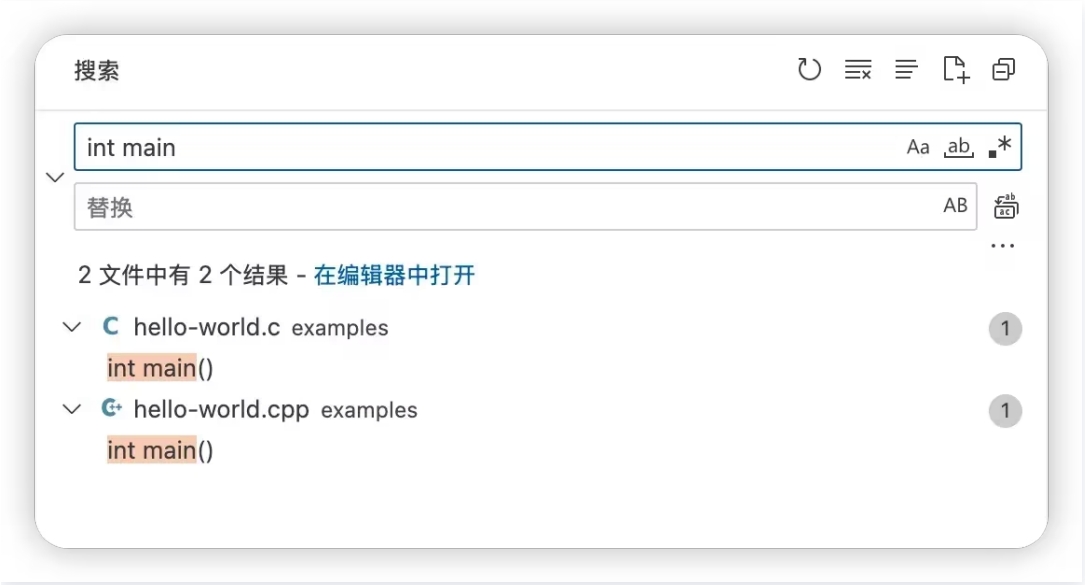
场景	Windows / Linux	Mac
全局搜索（全项目）	`Ctrl + Shift + F`	`Command + Shift + F`
当前文件搜索	`Ctrl + F`	`Command + F`
当前文件替换	`Ctrl + H`	`Command + H`

二、怎么用（按需求选）

1）全局搜索：查整个项目

适合：找函数/变量的引用位置、定位配置项、排查重复代码。

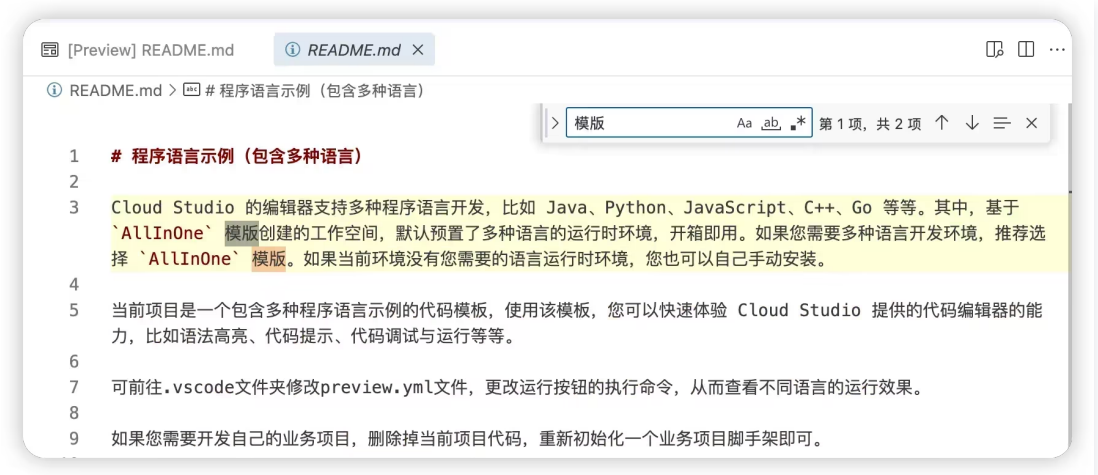
- 打开全局搜索面板
- 输入关键字（例如变量名/接口路径/报错信息）
- 点击结果，跳转到对应文件与行



2）当前文件/选中区域：查局部

适合：在单个文件里快速定位某个字段或片段。

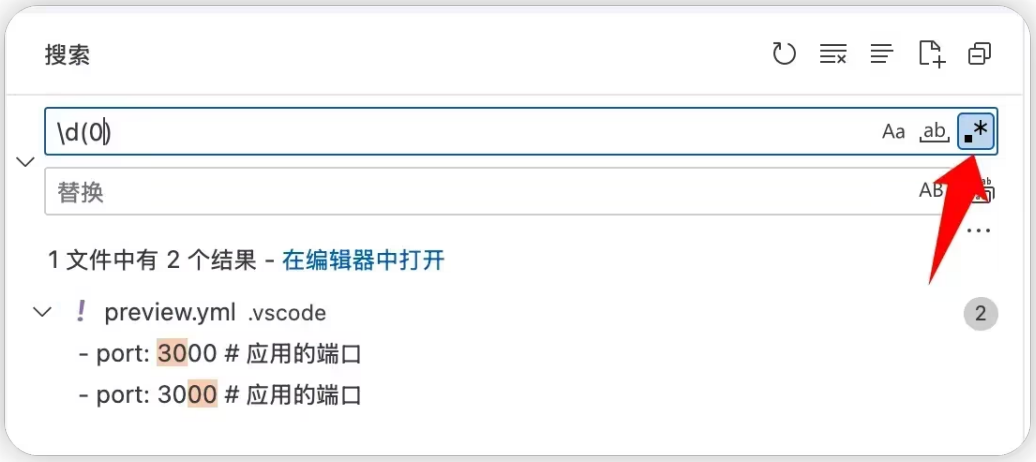
- 当前文件：按 `Ctrl + F` / `Command + F`
- 选中区域：先选中一段内容，再按 `Ctrl + F` / `Command + F`（范围会限定在选中内容内）



3）正则搜索：按规则匹配

适合：查找特定格式（如手机号/日期/函数定义等）。

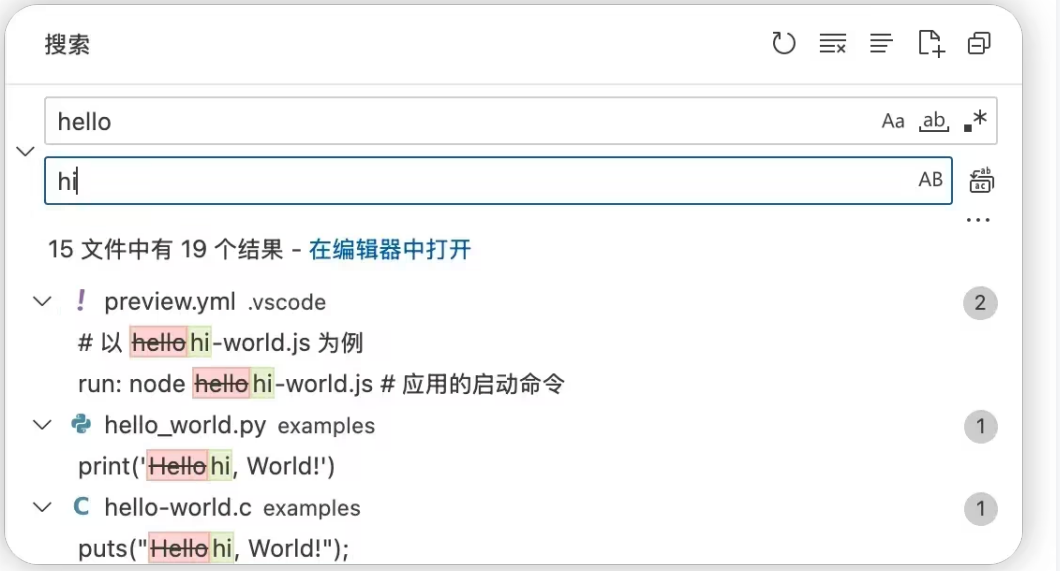
- 打开搜索框（全局或当前文件均可）
- 点击 `.*` 图标启用正则
- 输入正则表达式进行匹配



4）替换：查到就改

适合：批量改变量名、替换旧 API、统一文案/路径。

- **当前文件：在搜索框中打开替换，支持“替换当前/替换全部”**
- **全项目：在全局搜索面板启用替换，建议先预览结果再执行**



三、实用技巧

- **先缩小范围再替换：全局替换前先确认匹配结果，避免误改。**
- **排除无关目录：全局搜索可排除 `node_modules` 、 `dist` 等目录，减少噪音。**
- **正则先验证：规则复杂时建议先在小范围测试，确认匹配正确再扩大范围。**

代码助手 CodeBuddy

最近更新时间：2026-05-19 18:07:22

概述

CloudStudio 内置 **腾讯云代码助手 CodeBuddy**，可在多模型（如 GLM、混元、DeepSeek）间切换，用于**解释代码、生成注释、修复问题、生成单元测试**等；也支持 **Craft** 用自然语言进行多文件生成。

> **重要**
> CloudStudio 里面的 CodeBuddy 配额是每月 **500 credit**。

一、唤醒代码助手 CodeBuddy

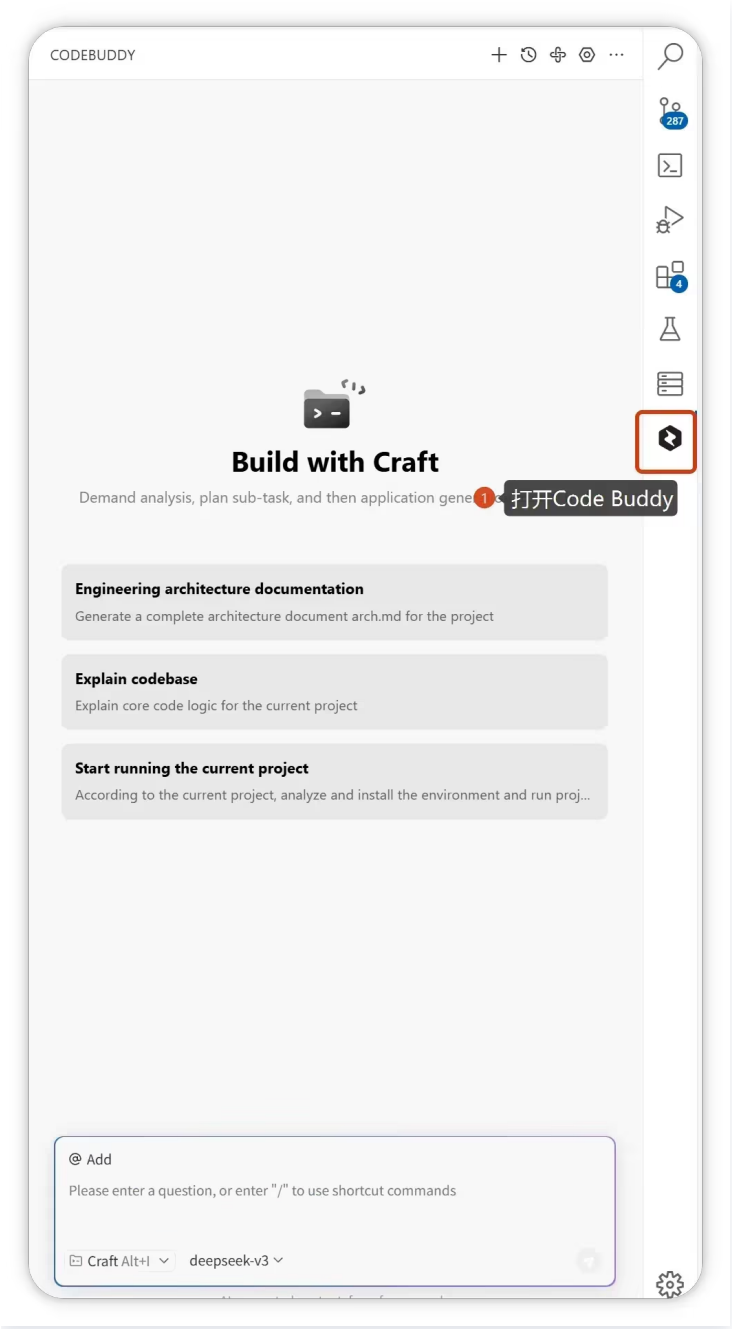
CodeBuddy 提供多种唤醒路径，覆盖不同操作场景，灵活选择即可：

方式 1：通过右侧工具栏唤醒（全局调用）

适用于主动发起 AI 交互（如无选中代码时提问）的场景。

操作步骤：

1. 在工作空间右侧找到并点击「**腾讯云代码助手**」**图标**（通常为机器人或 AI 相关图标）；
2. 弹出 CodeBuddy 功能面板，可直接输入需求（如“写一个 Python 冒泡排序函数”），或后续选中代码使用针对性功能。



方式 2：选中代码自动触发（精准适配）

适用于对特定代码片段操作（如解释、修复）的场景，无需手动打开面板。

操作步骤：

1. 在代码编辑区（CODE 区域）选中目标代码片段（如函数、循环逻辑）；
2. 选中后，代码上方自动弹出功能选项栏（含“解释代码”“修复代码”“添加注释”等）；
3. 点击对应选项，触发 AI 处理（如点击“解释代码”生成逻辑说明）。

方式 3：右键菜单唤醒（便捷操作）

适用于习惯右键菜单调用的场景。

操作步骤：

1. 选中目标代码片段后右键点击；
2. 在右键菜单中选择「**腾讯云 AI 代码助手**」选项；
3. 在子菜单中按需选择功能（如“生成单元测试”“优化代码”“变量重命名建议”），触发 AI 服务。

核心基础能力：代码答疑与解析

无论通过哪种方式唤醒，CodeBuddy 均能针对选中代码提供深度解析，包括：

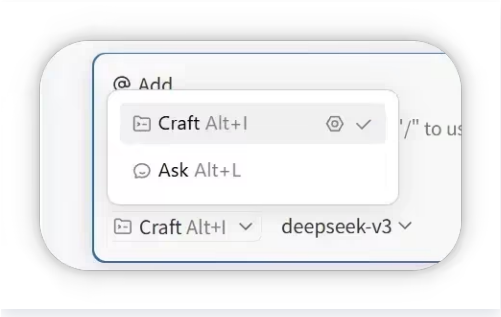
- 代码核心用途（如“处理用户登录请求的参数验证”）；
- 关键逻辑含义（如“if 条件判断 token 是否过期，过期返回 401 状态码”）；
- 代码在文件中的角色（如“后端接口核心处理函数，接收请求并调用数据库服务”）。

二、进阶功能：Craft 开发智能体

Craft 是 CodeBuddy 的进阶能力，支持通过自然语言指令或已有代码文件，自动生成多文件应用（如完整前端页面、后端接口服务），无需手动编写基础代码。

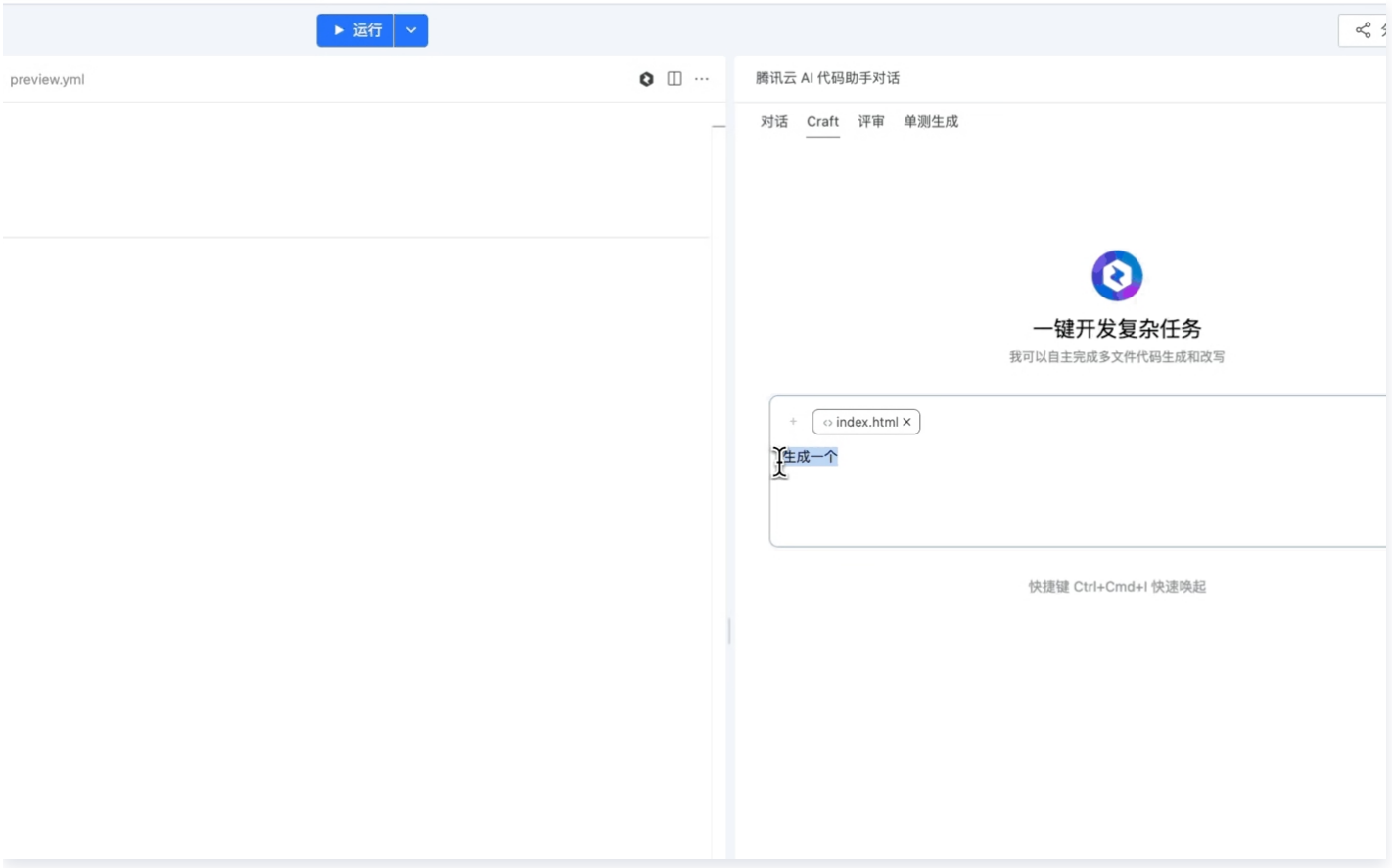
进入 Craft 对话框

1. 打开 CodeBuddy 功能面板（通过右侧工具栏唤醒）；
2. 点击面板中的「Craft」选项，进入智能体专属对话框（界面提示“自然语言生成代码”）。



自然语言指令生成应用（从 0 到 1）

用于需快速搭建基础项目框架的场景，只需输入清晰的自然语言需求，AI 会自动生成对应代码文件及目录结构。


运行
preview.yml
腾讯云 AI 代码助手对话
对话
Craft
评审
单测生成
一键开发复杂任务
我可以自主完成多文件代码生成和改写
index.html
生成一个
快捷键 Ctrl+Cmd+I 快速唤起

端口与 Web 预览插件

最近更新时间：2026-05-19 18:07:22

概述

端口插件用于查看与管理工作空间内的端口，并提供预览入口与访问权限设置。

> **静态网页如何预览**
> 可使用 Live Server 插件，请参考 快速预览静态网页 。

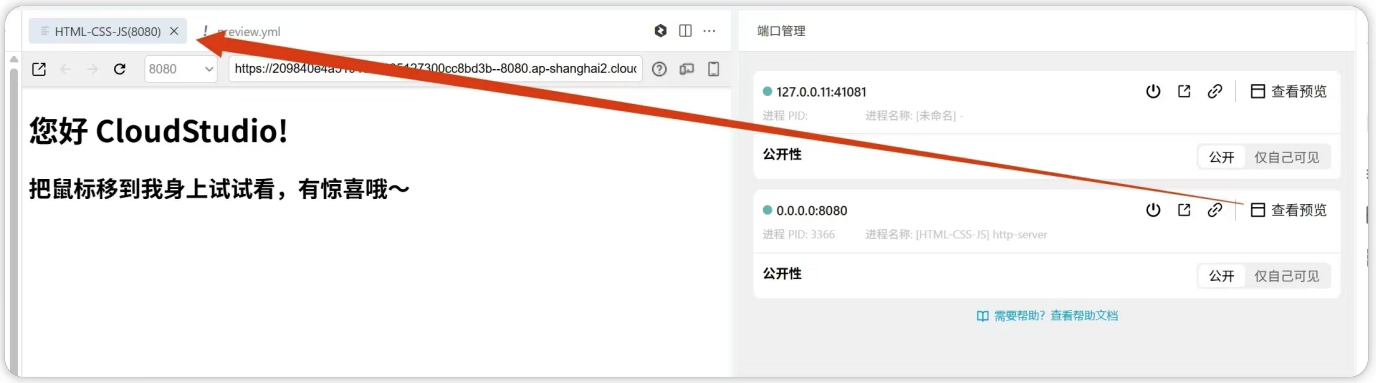
一、如何打开

1）打开端口插件

在左侧工具栏点击「**端口插件**」图标，面板会展示当前已识别的端口卡片（端口号、状态、进程信息等）。

2）打开 Web 预览

在端口卡片中点击「**查看预览**」，会自动打开 Web 预览窗口并加载该端口页面。



二、端口卡片信息说明

端口卡片会展示以下信息，通常无需手动查询：

- **端口号**：服务监听的端口。
- **状态**：端口是否正在使用。
- **PID**：关联进程 ID，用于定位与排查。
- **进程名称**：显示规则见 **八、插件按钮功能提示。**

三、端口常用操作

端口卡片右侧提供常用操作按钮：

1）关闭端口

用于终止关联进程并释放端口（如端口冲突、需要重启服务）。

2）在浏览器中打开

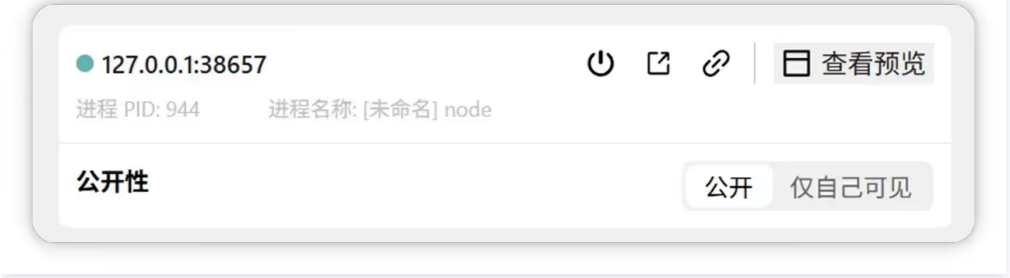
在本地默认浏览器打开该端口页面，便于使用浏览器开发者工具调试。

3）复制分享链接

复制端口访问链接（请确保对应服务已启动）。

4）查看预览

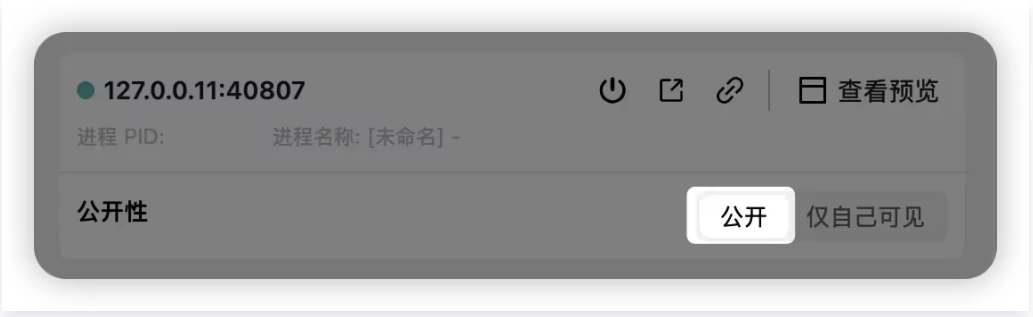
在右侧打开 Web 预览窗口并加载该端口页面。



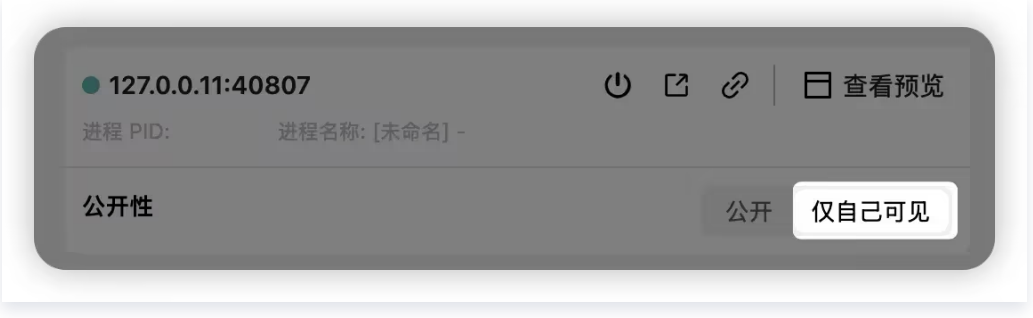
四、端口访问权限设置

您可以切换端口链接的可访问范围：

- **公开（默认）**：链接可被外部访问（需服务已启动），适合对外展示。



- **仅自己可见**：仅当前账号可访问（需服务已启动），适合隐私场景。



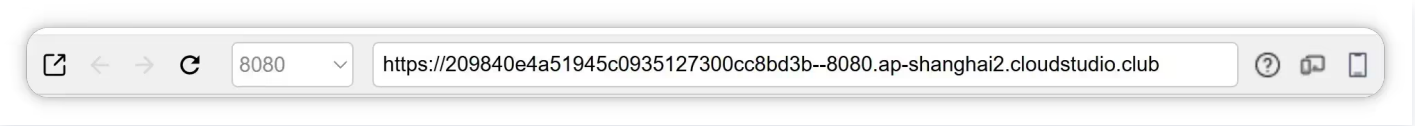
五、Web 预览

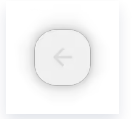
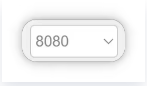
Web 预览用于在工作空间内快速查看页面效果。

1）切换预览端口

在 Web 预览窗口顶部点击「端口」下拉框，选择目标端口即可切换。

2）工具栏按钮说明



按钮	作用
	在本地浏览器打开预览（可配合开发者工具）
	后退（上一页）
	前进（下一页）
	刷新预览
	显示当前预览端口号
	显示预览地址（可复制）

	打开帮助文档
	切换桌面/手机视图
	生成二维码，在手机查看

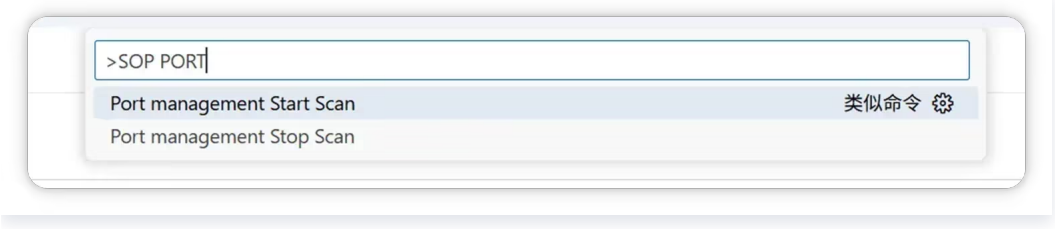
> **提示**
> 需要深入排查问题时，优先使用「**在本地浏览器打开**」，结合 Network/Console 等工具分析请求与报错。

六、插件显示端口范围

端口插件可识别并展示 0~80000 范围内的端口。

七、停止端口插件扫描

1. 打开命令面板：
   - Mac：`Command + Shift + P`
   - Windows：`Ctrl + Shift + P`
2. 输入 `SOP PORT`，选择 `Port management Stop Scan`。



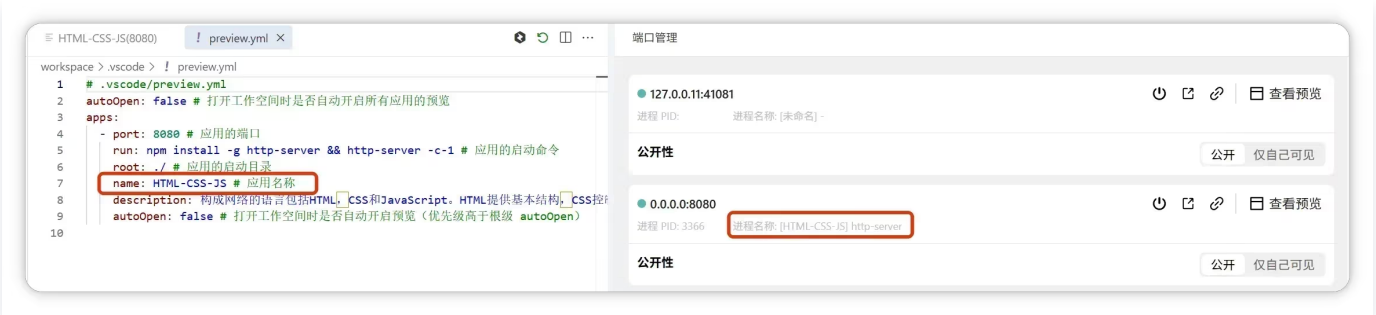
八、插件按钮功能提示

将鼠标悬停在插件任意按钮上约 3 秒，会显示该按钮说明。

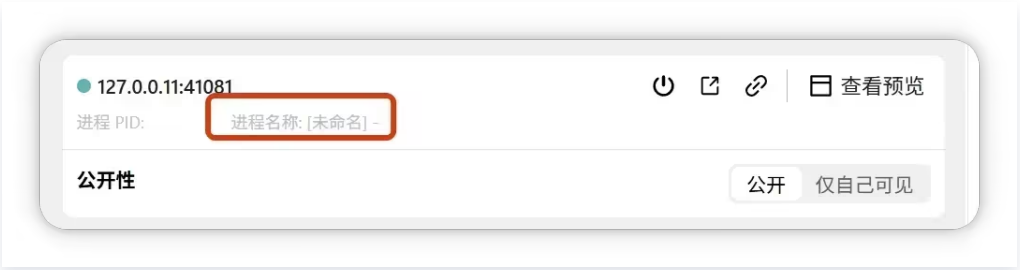
九、插件进程名称组成规则

端口卡片中的“进程名称”由以下规则生成：

- **已配置预览配置文件（`preview.yml`）：显示为 `[preview.yml 的 name]` 进程名称。**



- **未配置 `preview.yml`：显示为 `[未命名]` 进程名称。**



CodeBuddy Code（终端版）

最近更新时间：2026-05-19 18:07:22

概述

CodeBuddy Code 是 CloudStudio 内置的智能编码工具，可在**终端**中通过自然语言完成常见开发任务（解释/优化/修复代码、生成测试、代码审查等）。

> **重要**
> CloudStudio 里面的 CodeBuddy 配额是每月 **500 credit**。

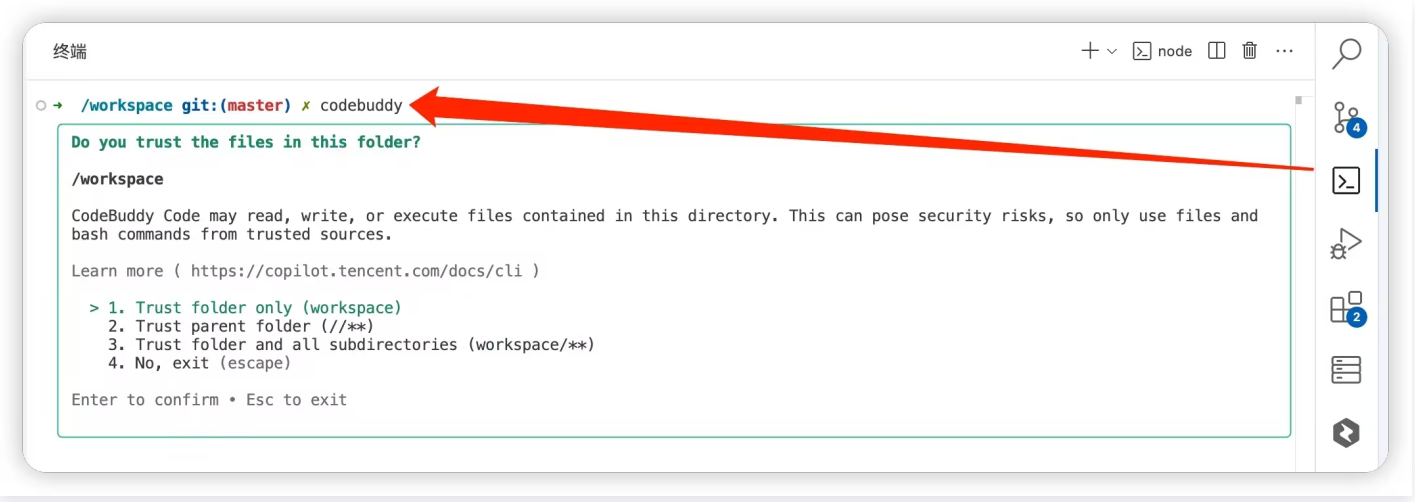
一、快速开始

1）进入交互模式（默认）

在任意项目目录运行：

```
codebuddy
```

启动后在输入框直接输入需求即可。



2）执行单次任务（非交互）

```
codebuddy "修复这个 bug"
```

二、常用场景（示例）

1）日常开发

```
codebuddy "帮我优化这个函数的性能"
codebuddy "为这个组件添加错误处理"
codebuddy "生成这个 API 的单元测试"
```

2）代码审查

```
codebuddy "检查这次提交的代码质量"
codebuddy "分析潜在的安全问题"
codebuddy "建议代码改进方案"
```

3）学习与探索

```
codebuddy "解释这个算法的实现原理"
codebuddy "这个设计模式在项目中如何使用"
codebuddy "推荐适合的第三方库"
```

三、命令速查

1）基本用法

```
codebuddy [选项] [命令] [提示词]
```

```
● → /workspace git:(master) ✗ codebuddy -p "解释一下迪杰斯特拉算法的实现原理"
我来为您详细解释迪杰斯特拉算法的实现原理。

## 迪杰斯特拉算法概述

迪杰斯特拉算法是一种用于在加权图中找到从源顶点到所有其他顶点的最短路径的算法。它只适用于边权重非负的图。

## 核心原理

### 1. 基本思想
- **贪心策略**：每次从未处理的顶点中选择距离源点最近的顶点
- **逐步扩展**：将选中的顶点标记为已处理，并用它来松弛到其他顶点的距离
- **优先队列优化**：使用最小堆来高效获取当前最近的顶点

### 2. 数据结构
- `dist[]`：存储源点到每个顶点的当前最短距离
- `visited[]`：标记顶点是否已被处理
- `priority_queue`：最小堆，存储未处理顶点及其距离

### 3. 算法步骤

**初始化：**
- 将源点到自己的距离设为0
- 将源点到其他所有顶点的距离设为无穷大
- 将所有顶点加入优先队列
```

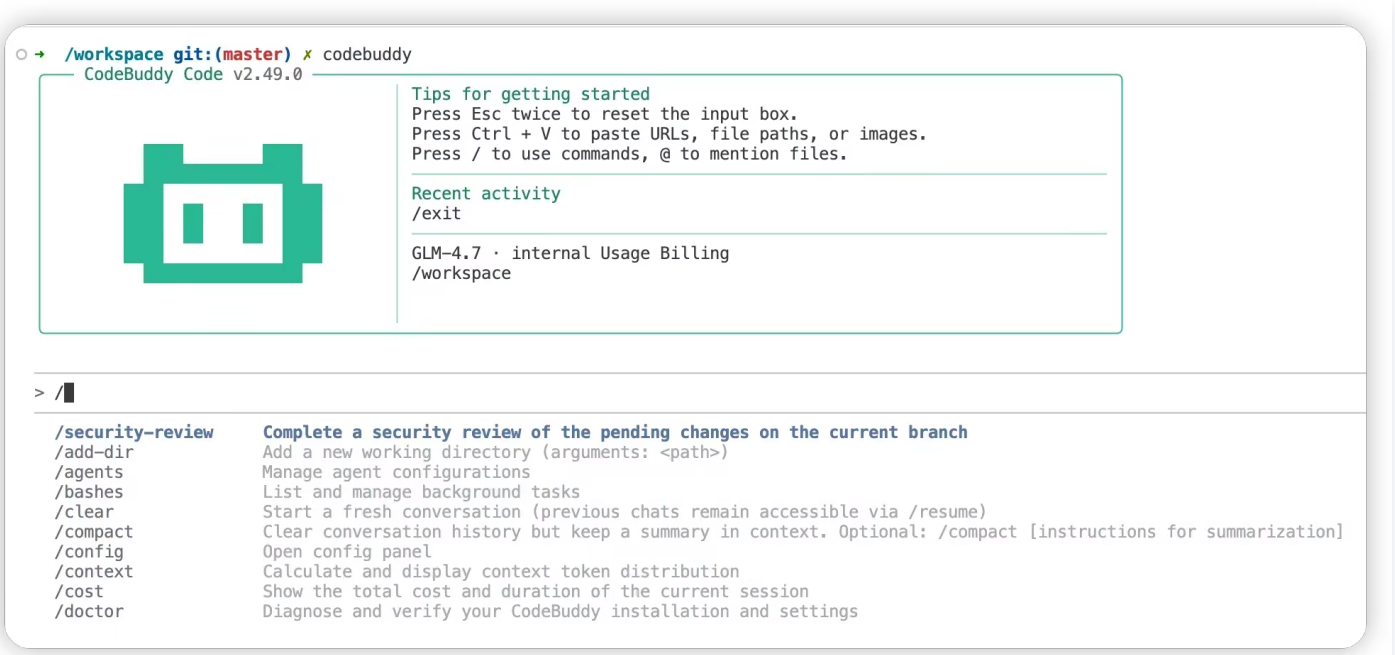
2）常用选项

> **说明**
> 下面只保留高频选项；更多参数可使用 `codebuddy -h` 查看。

选项	作用	示例
`codebuddy`	启动交互式会话（默认）	`codebuddy`
`codebuddy "..."`	执行单次任务	`codebuddy "解释这个函数"`
`-p, --print`	打印响应并退出（非交互）	`codebuddy -p "总结这段代码"`
`-c, --continue`	继续最近的对话	`codebuddy -c`
`-r, --resume [会话ID]`	恢复指定对话	`codebuddy -r 12345`
`--model <模型>`	指定使用的模型	`codebuddy --model sonnet`
`--ide`	启动时自动连接 IDE	`codebuddy --ide`
`-d, --debug`	开启调试输出	`codebuddy -d`
`-h, --help`	显示帮助信息	`codebuddy -h`
`-V, --version`	显示版本信息	`codebuddy -V`

四、交互式会话指令（以 `/` 开头）

在交互模式输入 `/` 可看到可用指令，例如 `/model` 用于切换模型。



指令	用途
`/add-dir`	添加新的工作目录
`/agents`	管理 Agent 配置
`/bashes`	列出和管理后台任务
`/clear`	清除对话历史并释放上下文
`/compact`	清理历史但保留摘要（可选：`/compact [摘要指令]`）
`/config`	打开配置面板
`/cost`	显示当前会话的总成本和持续时间
`/doctor`	诊断并验证安装与设置
`/exit`	退出 CodeBuddy
`/export`	导出当前对话到文件或剪贴板

五、子命令

子命令	用途	示例
`config`	管理配置	`codebuddy config set -g theme dark`

`mcp`	配置与管理 MCP 服务器	`codebuddy mcp`
`doctor`	检查系统健康状态	`codebuddy doctor`
`update`	检查并安装更新	`codebuddy update`
`install`	安装原生构建版本	`codebuddy install stable`

常见问题
快捷键

最近更新时间：2026-05-19 18:07:22

通用快捷键

快捷键	作用
Ctrl+Shift+P,F1	展示全局命令面板
Ctrl+P	快速打开最近打开的文件
Ctrl+Shift+N	打开新的编辑器窗口
Ctrl+Shift+W	关闭编辑器

基础编辑

快捷键	作用
Ctrl/Command + X	剪切
Ctrl/Command+ C	复制
Alt + up/down	移动行上下
Shift + Alt up/down	在当前行上下复制当前行
Ctrl/Command+ Shift + K	删除行
Ctrl/Command+ Enter	在当前行下插入新的一行
Ctrl/Command+ Shift + Enter	在当前行上插入新的一行
Ctrl/Command+ Shift +	匹配花括号的闭合处，跳转
Ctrl/Command+] / [	行缩进
Home	光标跳转到行头
End	光标跳转到行尾
Ctrl/Command + Home	跳转到页头
Ctrl/Command + End	跳转到页尾

Ctrl/Command + up/down	行视图上下偏移
Alt + PgUp/PgDown	屏视图上下偏移
Ctrl/Command + Shift + [	折叠区域代码
Ctrl/Command + Shift +]	展开区域代码
Ctrl/Command+ K Ctrl/Command+ [	折叠所有子区域代码
Ctrl/Command+ k Ctrl/Command+]	展开所有折叠的子区域代码
Ctrl/Command+ K Ctrl/Command+ 0	折叠所有区域代码
Ctrl/Command+ K Ctrl/Command+ J	展开所有折叠区域代码
Ctrl/Command+ K Ctrl/Command+ C	添加行注释
Ctrl/Command+ K Ctrl/Command+ U	删除行注释
Ctrl/Command+ /	添加关闭行注释
Shift + Alt +A	块区域注释
Alt + Z	添加关闭词汇包含

导航

快捷键	作用
Ctrl/Command+ T	列出所有符号
Ctrl/Command+ G	跳转行
Ctrl/Command+ P	跳转文件
Ctrl/Command+ Shift + O	跳转到符号处
Ctrl/Command+ Shift + M	打开问题展示面板

F8	跳转到下一个错误或者警告
Shift + F8	跳转到上一个错误或者警告
Ctrl/Command+ Shift + Tab	切换到最近打开的文件
Alt + left / right	向后、向前
Ctrl/Command+ M	用 Tab 来移动焦点

查询与替换

快捷键	作用
Ctrl/Command+ F	查询
Ctrl/Command+ H	替换
F3 / Shift + F3	查询下一个/上一个
Alt + Enter	选中所有出现在查询中的
Ctrl/Command+ D	匹配当前选中的词汇或者行，再次选中-可操作
Ctrl/Command+ K Ctrl/Command+ D	移动当前选择到下个匹配选择的位置(光标选定)
Alt + C / R / W	不分大小写/使用正则/全字匹配

多行光标操作与选择

快捷键	作用
Alt + Click	插入光标-支持多个
Ctrl/Command+ Alt + up/down	上下插入光标-支持多个
Ctrl/Command+ U	撤销最后一次光标操作
Shift + Alt + I	插入光标到选中范围内所有行结束符
Ctrl/Command+ I	选中当前行
Ctrl/Command+ Shift + L	选择所有出现在当前选中的行-操作
Ctrl/Command+ F2	选择所有出现在当前选中的词汇-操作

Shift + Alt + right	从光标处扩展选中全行
Shift + Alt + left	收缩选择区域
Shift + Alt + (drag mouse)	鼠标拖动区域，同时在多个行结束符插入光标
Ctrl/Command+ Shift + Alt + (Arrow Key)	插入多行光标的[方向键控制]
Ctrl/Command+ Shift + Alt + PgUp/PgDown	插入多行光标的[整屏生效]

丰富的语言操作

快捷键	作用
Ctrl/Command+ Space	输入建议[智能提示]
Ctrl/Command+ Shift + Space	参数提示
Tab	Emmet 指令触发/缩进
Shift + Alt + F	格式化代码
Ctrl/Command+ K Ctrl/Command+ F	格式化选中部分的代码
F12	跳转到定义处
Alt + F12	代码片段显示定义
Ctrl/Command+ K F12	在其他窗口打开定义处
Ctrl/Command+ .	快速修复部分可以修复的语法错误
Shift + F12	显示所有引用
F2	重命名符号
Ctrl/Command+ Shift + . / ,	替换下个值
Ctrl/Command+ K Ctrl/Command+ X	移除空白字符
Ctrl/Command+ K M	更改页面文档格式

编辑器管理

快捷键	作用
Ctrl/Command+ F4, Ctrl/Command+ W	关闭编辑器
Ctrl/Command+ k F	关闭当前打开的文件夹
Ctrl/Command+	切割编辑窗口
Ctrl/Command+ 1/2/3	切换焦点在不同的切割窗口
Ctrl/Command+ K Ctrl/Command	切换焦点在不同的切割窗口
Ctrl/Command+ Shift + PgUp/PgDown	切换标签页的位置
Ctrl/Command+ K	切割窗口位置调换

文件管理

快捷键	作用
Ctrl/Command+ N	新建文件
Ctrl/Command+ O	打开文件
Ctrl/Command+ S	保存文件
Ctrl/Command+ Shift + S	另存为
Ctrl/Command+ K S	保存所有当前已经打开的文件
Ctrl/Command+ F4	关闭当前编辑窗口
Ctrl/Command+ K Ctrl/Command+ W	关闭所有编辑窗口
Ctrl/Command+ Shift + T	撤销最近关闭的一个文件编辑窗口
Ctrl/Command+ K Enter	保持开启
Ctrl/Command+ Shift + Tab	调出最近打开的文件列表，重复按会切换
Ctrl/Command+ Tab	与上面一致，顺序不一致
Ctrl/Command+ K P	复制当前打开文件的存放路径
Ctrl/Command+ K R	打开当前编辑文件存放位置【 文件管理器 】

Ctrl/Command+ K O	在新的编辑器中打开当前编辑的文件

显示

快捷键	作用
F11	切换全屏模式
Ctrl/Command+ =/−	放大 / 缩小
Ctrl/Command+ B	侧边栏显示隐藏
Ctrl/Command+ Shift + E	资源视图和编辑视图的焦点切换
Ctrl/Command+ Shift + F	打开全局搜索
Ctrl/Command+ Shift + G	打开 Git 可视管理
Ctrl/Command+ Shift + D	打开 Debug 面板
Ctrl/Command+ Shift + X	打开插件市场面板
Ctrl/Command+ Shift + H	在当前文件替换查询替换
Ctrl/Command+ Shift + J	开启详细查询
Ctrl/Command+ Shift + V	预览 Markdown 文件【编译后】
Ctrl/Command+ K v	在边栏打开渲染后的视图【新建】

调试

快捷键	作用
F9	添加解除断点
F5	启动调试、继续
F11 / Shift + F11	单步进入 / 单步跳出
F10	单步跳过
Ctrl/Command+ K Ctrl/Command+ I	显示悬浮

集成终端

快捷键	作用
Ctrl/Command+ `	打开集成终端
Ctrl/Command + Shift + `	创建一个新的终端
Ctrl/Command+ Shift + C	复制所选
Ctrl/Command+ Shift + V	复制到当前激活的终端
Shift + PgUp / PgDown	页面上下翻屏
Ctrl/Command+ Home / End	滚动到页面头部或尾部

浏览器兼容性

最近更新时间：2026-05-19 18:07:22

概述

CloudStudio 目前内置的编辑器为 VS Code 1.89 版本，有浏览器最低版本的限制。如您在打开编辑器的过程中出现提示，则需要升级或者更换浏览器以获得更好的体验。

浏览器兼容要求

目前 CloudStudio 内置的 VS Code 编辑器可用于以下版本的浏览器：

- Mozilla Firefox 100及以上
- Safari 15及以上
- Google Chrome 90及以上
- Microsoft Edge 90及以上

在 Firefox 和 Safari 中，Web 视图可能会显示不同或出现一些意外行为。您可以在 VS Code GitHub 存储库中查看问题查询，以跟踪与特定浏览器相关的问题，例如 Safari 标签和 Firefox 标签。

MySQL 常见问题

最近更新时间：2026-05-19 18:07:22

`systemctl` 启动失败

现象

执行 `sudo systemctl start mysql` 后提示：

```
System has not been booted with systemd as init system (PID 1). Can't
operate.
Failed to connect to bus: Host is down
```

原因

CloudStudio 环境不使用 `systemd`，因此 `systemctl` 不可用。

解决方法

改用 `service` 启动：

```
sudo service mysql start
```

后续启动报“权限相关错误”

现象

MySQL 初次安装可启动，但后续执行 `sudo service mysql start` 报权限错误。

```
→ /workspace git:(main) X sudo service mysql start
 * Starting MySQL database server mysqld
su: warning: cannot change directory to /nonexistent: No such file or directory
                                                                    [fail]
```

解决方法

把 MySQL 的常用目录权限修复为 `mysql:mysql`，再重新启动。

1. 修复目录权限

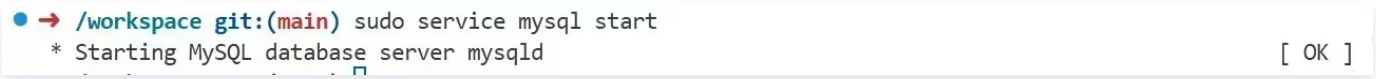
```
# 如果目录不存在，可先创建（可选）
sudo mkdir -p /var/run/mysqld

# 修复数据目录权限
```

```
sudo chown -R mysql:mysql /var/lib/mysql

# 修复日志目录权限
sudo chown -R mysql:mysql /var/log/mysql

# 修复运行时 PID 目录权限
sudo chown -R mysql:mysql /var/run/mysqld
```

2. 启动并检查状态

```
sudo service mysql start
sudo service mysql status
```

3. 确认启动成功



> **提示**
> 如果仍启动失败，优先查看错误日志定位具体原因（例如权限不足、端口占用、配置错误等）。

URL 访问异常排查

最近更新时间：2026-05-19 18:07:22

> **问题**
> 当您遇到**服务启动了但页面打不开、或前后端请求 URL 不通**，通常按下面三步就能解决。

一、先确认：服务监听地址是否正确

很多服务默认只监听 `127.0.0.1`，会导致外部（预览/浏览器）访问不到。

- **建议：让服务监听 `0.0.0.0` 并指定端口。**

```
0.0.0.0:${PORT}
```

- **示例（端口 9000）：** `0.0.0.0:9000`

> **说明：**
> 不同框架的写法不同，但目标一致：**host 绑定到 0.0.0.0，并明确端口。**

二、在 CloudStudio 里生成可访问的预览 URL

在 CloudStudio 环境中，可以用预置环境变量拼出完整访问地址：

```
https://${X_IDE_SPACE_KEY}-
-${PORT}.${X_IDE_SPACE_REGION}.${X_IDE_SPACE_HOST}
```

变量说明

变量	含义	示例
`${X_IDE_SPACE_KEY}`	工作空间唯一标识	`5adb8439bf8147658b86f063097fa479`
`${PORT}`	服务端口号（由您的服务决定）	`9000`
`${X_IDE_SPACE_REGION}`	工作空间所在区域	`ap-shanghai2`

`${X_IDE_SPACE_HOST}`	平台域名	`cloudstudio.club`

生成示例（端口 9000）

```
https://5adb8439bf8147658b86f063097fa479--9000.ap-shanghai2.cloudstudio.club
```

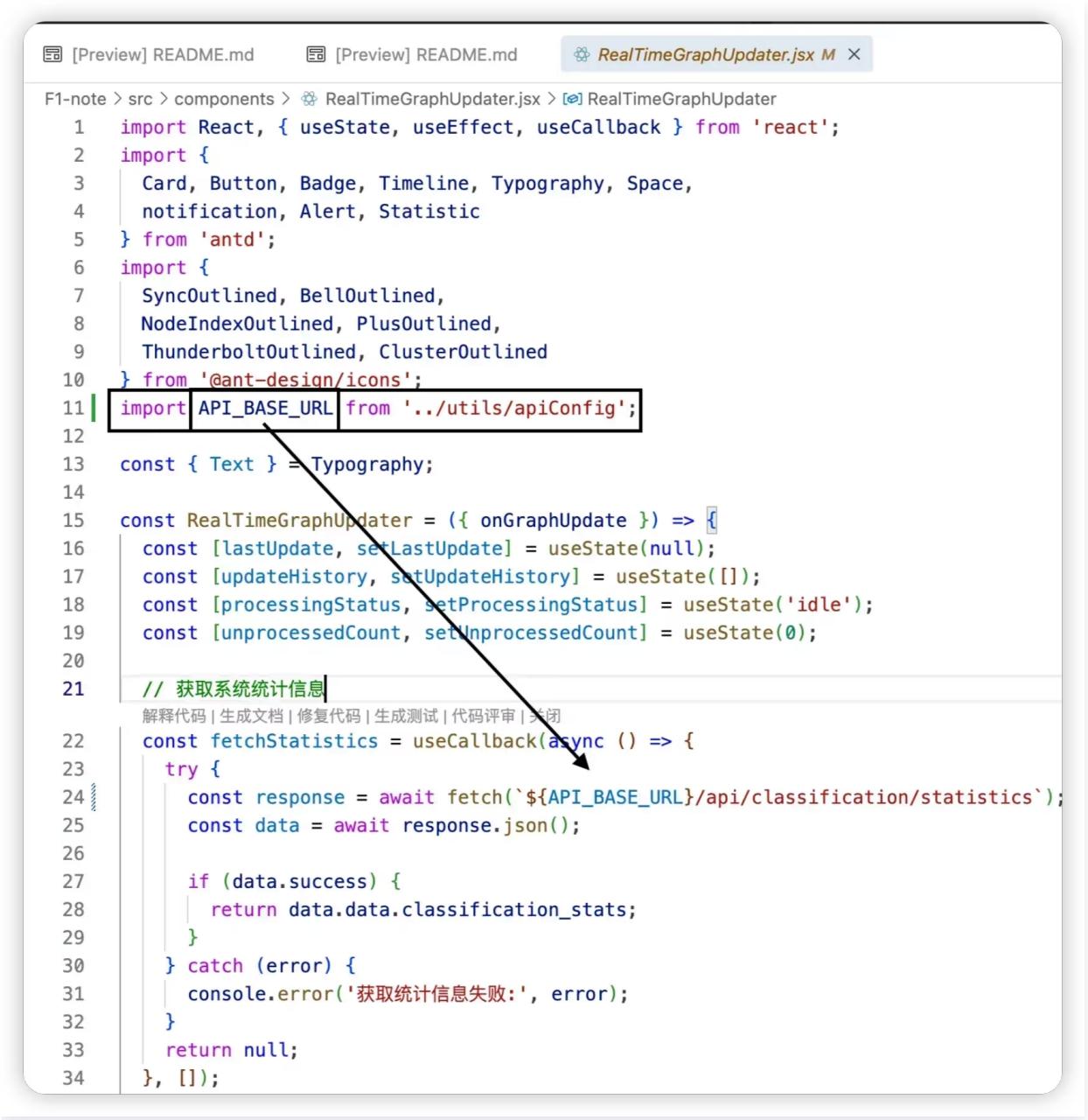
> **注意：**
> 预览 URL 里的端口必须与服务实际监听端口一致，否则会访问失败。

三、最小配置示例（前端调用后端）

您可以把上面的 URL 作为 `API_BASE_URL`，统一在项目里使用。

F1-note > src > utils > JS apiConfig.js > default

```
1  // API URL 配置
2  const X_IDE_SPACE_KEY = process.env.X_IDE_SPACE_KEY;
3  const X_IDE_SPACE_REGION = process.env.X_IDE_SPACE_REGION;
4  const X_IDE_SPACE_HOST = process.env.X_IDE_SPACE_HOST;
5  const host = X_IDE_SPACE_KEY ? `${X_IDE_SPACE_KEY}--9000.${X_IDE_SPACE_REGION}.${X_IDE_SPACE_HOST}` : '0.0.0.0:9000';
6  const API_BASE_URL = process.env.REACT_APP_API_URL || `https://${host}`;
7
8  export default API_BASE_URL;
```


[Preview] README.md
[Preview] README.md
RealTimeGraphUpdater.jsx M
F1-note > src > components > RealTimeGraphUpdater.jsx > RealTimeGraphUpdater
1 import React, { useState, useEffect, useCallback } from 'react';
2 import {
3 Card, Button, Badge, Timeline, Typography, Space,
4 notification, Alert, Statistic
5 } from 'antd';
6 import {
7 SyncOutlined, BellOutlined,
8 NodeIndexOutlined, PlusOutlined,
9 ThunderboltOutlined, ClusterOutlined
10 } from '@ant-design/icons';
11 import API_BASE_URL from '../utils/apiConfig';
12
13 const { Text } = Typography;
14
15 const RealTimeGraphUpdater = ({ onGraphUpdate }) => {
16 const [lastUpdate, setLastUpdate] = useState(null);
17 const [updateHistory, setUpdateHistory] = useState([]);
18 const [processingStatus, setProcessingStatus] = useState('idle');
19 const [unprocessedCount, setUnprocessedCount] = useState(0);
20
21 // 获取系统统计信息
解释代码 | 生成文档 | 修复代码 | 生成测试 | 代码评审 | 关闭
22 const fetchStatistics = useCallback(async () => {
23 try {
24 const response = await fetch(`${API_BASE_URL}/api/classification/statistics`);
25 const data = await response.json();
26
27 if (data.success) {
28 return data.data.classification_stats;
29 }
30 } catch (error) {
31 console.error('获取统计信息失败:', error);
32 }
33 return null;
34 }, []);

Matplotlib 中文显示问题

最近更新时间：2026-05-19 18:07:22

问题

常见表现：中文变成方块/乱码，或提示“找不到字体”，以及负号 `-` 显示异常。

先判断您遇到哪一种

- **A：报错找不到字体**（提示缺少中文字体）→ 看 安装中文字体 。
- **B：不报错但中文仍是方块/乱码** → 看 让 Matplotlib 使用中文字体 。

安装中文字体

适用于

提示找不到字体

当 Matplotlib 提示缺少字体时，需要把中文字体文件放入它的字体目录。

```python
import matplotlib.pyplot as plt
import numpy as np

# 设置字体，支持中文显示
plt.rcParams["font.family"] = ["SimHei", "SimSun", "Microsoft-YaHei"]
# 解决负号显示问题（可选）
plt.rcParams['axes.unicode_minus'] = False  # 正确显示负号

# 示例：绘制包含中文的图表
x = np.linspace(0, 10, 100)
y = np.sin(x)

plt.plot(x, y)
plt.title("正弦曲线示例")  # 中文标题
plt.xlabel("横轴")         # 中文x轴标签
plt.ylabel("纵轴")         # 中文y轴标签
plt.show()
```

[3] ✓ 0.1s Python

```
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
...
findfont: Font family 'Microsoft-YaHei' not found.
findfont: Font family 'SimHei' not found.
findfont: Font family 'SimSun' not found.
findfont: Font family 'Microsoft-YaHei' not found.
```

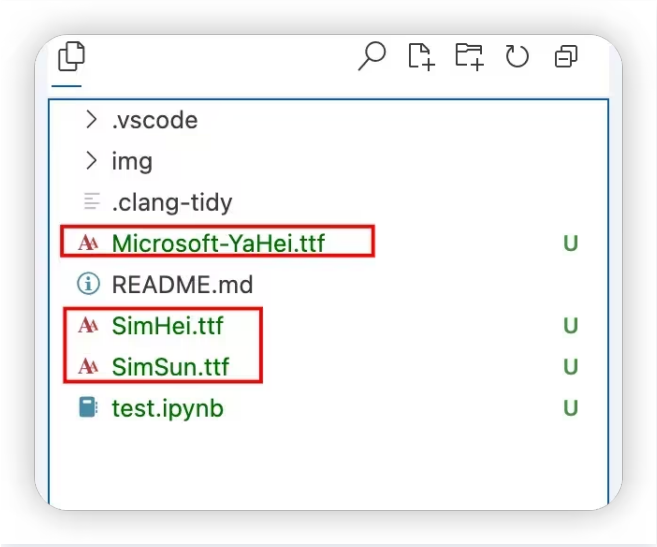
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

1）准备字体文件

建议使用常见 TrueType 字体（`.ttf`）：

- Microsoft-YaHei.ttf
- SimHei.ttf
- SimSun.ttf

把字体文件放到工作空间（例如放在 `/workspace/` 下）。



2）安装字体到 Matplotlib 字体库

1. 首先进入 Matplotlib 的字体目录，在终端中执行：

```
cd /root/.pyenv/versions/3.11.1/lib/python3.11/site-packages/matplotlib/mpl-data/fonts/ttf
```

> **注意：**
> 这里修改的 3.11.1 Python 版本，如需修改其他版本的需根据类似目录找到字体路径

2. 将工作空间中的字体文件移动到该目录，使用 mv 命令：

```
mv /workspace/字体文件名.ttf .
```

例如移动黑体文件：

```
mv /workspace/SimHei.ttf .
```

```
● → /workspace git:(master) cd /root/.pyenv/versions/3.11.1/lib/python3.11/site-packages/matplot
  lib/mpl-data/fonts/ttf
● → ttf git:(ff93c58b) ✗ mv /workspace/SimHei.ttf .
● → ttf git:(ff93c58b) ✗ mv /workspace/SimSun.ttf .
● → ttf git:(ff93c58b) ✗ mv /workspace/Microsoft-YaHei.ttf .
```

3）验证字体安装

在终端中执行 `ll` 命令，查看字体文件是否已成功移动到字体目录中，确认文件名正确显示。

```
● → ttf git:(ff93c58b) ✗ ll
total 41M
-rw-r--r-- 1 root root  26K May 13 15:34 cmb10.ttf
-rw-r--r-- 1 root root  21K May 13 15:34 cmex10.ttf
-rw-r--r-- 1 root root  32K May 13 15:34 cmmi10.ttf
-rw-r--r-- 1 root root  26K May 13 15:34 cmr10.ttf
-rw-r--r-- 1 root root  20K May 13 15:34 cmss10.ttf
-rw-r--r-- 1 root root  29K May 13 15:34 cmsy10.ttf
-rw-r--r-- 1 root root  28K May 13 15:34 cmtt10.ttf
-rw-r--r-- 1 root root 627K May 13 15:34 DejaVuSans-BoldOblique.ttf
-rw-r--r-- 1 root root 688K May 13 15:34 DejaVuSans-Bold.ttf
-rw-r--r-- 1 root root  26K May 13 15:34 DejaVuSansDisplay.ttf
-rw-r--r-- 1 root root 248K May 13 15:34 DejaVuSansMono-BoldOblique.ttf
-rw-r--r-- 1 root root 324K May 13 15:34 DejaVuSansMono-Bold.ttf
-rw-r--r-- 1 root root 246K May 13 15:34 DejaVuSansMono-Oblique.ttf
-rw-r--r-- 1 root root 333K May 13 15:34 DejaVuSansMono.ttf
-rw-r--r-- 1 root root 619K May 13 15:34 DejaVuSans-Oblique.ttf
-rw-r--r-- 1 root root 739K May 13 15:34 DejaVuSans.ttf
-rw-r--r-- 1 root root 339K May 13 15:34 DejaVuSerif-BoldItalic.ttf
-rw-r--r-- 1 root root 348K May 13 15:34 DejaVuSerif-Bold.ttf
-rw-r--r-- 1 root root  14K May 13 15:34 DejaVuSerifDisplay.ttf
-rw-r--r-- 1 root root 338K May 13 15:34 DejaVuSerif-Italic.ttf
-rw-r--r-- 1 root root 371K May 13 15:34 DejaVuSerif.ttf
-rw-r--r-- 1 root root 4.8K May 13 15:34 LICENSE_DEJAVU
-rw-r--r-- 1 root root 5.4K May 13 15:34 LICENSE_STIX
-rw-r--r-- 1 root root  15M Oct  9 08:04 Microsoft-YaHei.ttf
-rw-r--r-- 1 root root 9.4M Oct  9 08:04 SimHei.ttf
-rw-r--r-- 1 root root  11M Oct  9 08:04 SimSun.ttf
-rw-r--r-- 1 root root 177K May 13 15:34 STIXGeneralBolIta.ttf
-rw-r--r-- 1 root root 232K May 13 15:34 STIXGeneralBol.ttf
-rw-r--r-- 1 root root 171K May 13 15:34 STIXGeneralItalic.ttf
-rw-r--r-- 1 root root 438K May 13 15:34 STIXGeneral.ttf
-rw-r--r-- 1 root root  41K May 13 15:34 STIXNonUniBolIta.ttf
-rw-r--r-- 1 root root  30K May 13 15:34 STIXNonUniBol.ttf
-rw-r--r-- 1 root root  46K May 13 15:34 STIXNonUniIta.ttf
-rw-r--r-- 1 root root  58K May 13 15:34 STIXNonUni.ttf
-rw-r--r-- 1 root root  14K May 13 15:34 STIXSizFiveSymReg.ttf
-rw-r--r-- 1 root root  12K May 13 15:34 STIXSizFourSymBol.ttf
-rw-r--r-- 1 root root  16K May 13 15:34 STIXSizFourSymReg.ttf
-rw-r--r-- 1 root root  13K May 13 15:34 STIXSizOneSymBol.ttf
-rw-r--r-- 1 root root  20K May 13 15:34 STIXSizOneSymReg.ttf
-rw-r--r-- 1 root root  12K May 13 15:34 STIXSizThreeSymBol.ttf
-rw-r--r-- 1 root root  16K May 13 15:34 STIXSizThreeSymReg.ttf
-rw-r--r-- 1 root root  12K May 13 15:34 STIXSizTwoSymBol.ttf
-rw-r--r-- 1 root root  16K May 13 15:34 STIXSizTwoSymReg.ttf
```

让 Matplotlib 使用中文字体

> **适用于**
> 字体已存在但仍乱码/方块

通常是默认字体没有包含中文字体，按下面两种方式配置即可。

方案 1：临时配置（只对当前代码生效）

在代码开头加入：

```
import matplotlib.pyplot as plt

# 让 Matplotlib 优先使用中文字体（按可用性自动选择）
plt.rcParams["font.family"] = ["SimHei", "SimSun", "Microsoft YaHei",
"Microsoft-YaHei"]
# 负号显示异常时打开
plt.rcParams["axes.unicode_minus"] = False
```

方案 2：永久配置（修改 `matplotlibrc`）

1. 找到配置文件路径：

```
import matplotlib
```

```
print(matplotlib.matplotlib_fname())
```

2. 打开 `matplotlibrc`，设置字体与负号：

```
font.family         : sans-serif
font.sans-serif     : SimHei, SimSun, Microsoft YaHei, Microsoft-YaHei
axes.unicode_minus  : False
```

[Preview] README.md | test.ipynb U | matplotlibrc | SimHei.ttf

root > .pyenv > versions > 3.11.1 > lib > python3.11 > site-packages > matplotlib > mpl-data > matplotlibrc

```
## The font.variant property has two values: normal or small-caps.  For
## TrueType fonts, which are scalable fonts, small-caps is equivalent
## to using a font size of 'smaller', or about 83 % of the current font
## size.
##
## The font.weight property has effectively 13 values: normal, bold,
## bolder, lighter, 100, 200, 300, ..., 900.  Normal is the same as
## 400, and bold is 700.  bolder and lighter are relative values with
## respect to the current weight.
##
## The font.stretch property has 11 values: ultra-condensed,
## extra-condensed, condensed, semi-condensed, normal, semi-expanded,
## expanded, extra-expanded, ultra-expanded, wider, and narrower.  This
## property is not currently implemented.
##
## The font.size property is the default font size for text, given in points.
## 10 pt is the standard value.
##
## Note that font.size controls default text sizes.  To configure
## special text sizes tick labels, axes, labels, title, etc., see the rc
## settings for axes and ticks.  Special text sizes can be defined
## relative to font.size, using the following values: xx-small, x-small,
## small, medium, large, x-large, xx-large, larger, or smaller

#font.family:  sans-serif
#font.style:   normal
#font.variant: normal
#font.weight:  normal
#font.stretch: normal
#font.size:    10.0

# 字体族设置，添加中文字体
font.family         : sans-serif
# 字体列表，优先使用前面的字体
font.sans-serif     : SimHei, SimSun, Microsoft-YaHei
# 解决负号显示问题
axes.unicode_minus  : False
```

3. 保存后 **重启 Jupyter 内核** 使配置生效。

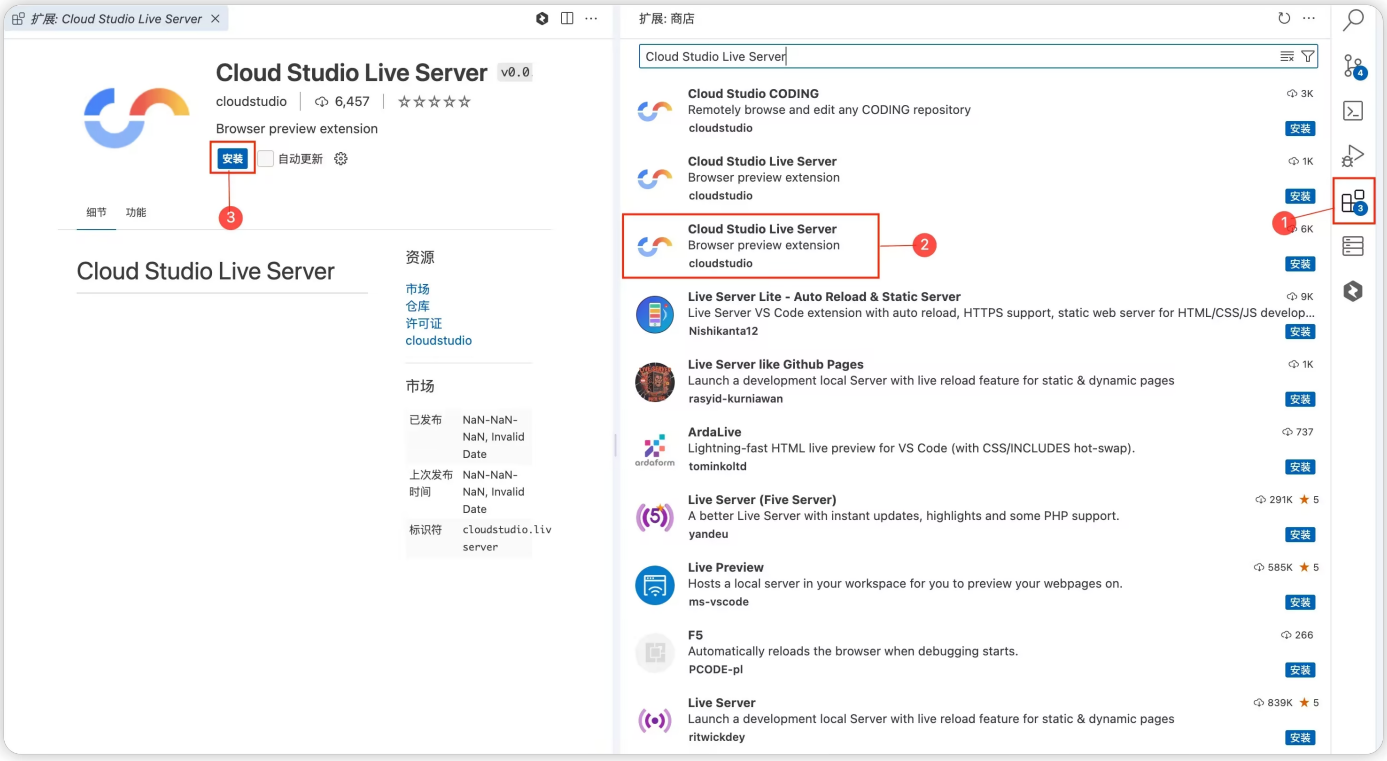
如何使用 Live Server 插件

最近更新时间：2026-05-19 18:07:22

> **这是一个 FAQ**
>
> - 本文适用于 **预览纯静态页面**（HTML/CSS/JS）。
> - 如果您在跑 React/Vue/Next.js 等项目，请优先使用对应框架的开发服务器（例如 `npm run dev`）。启动后的端口和预览地址的配置，请参考 端口与 Web 预览插件。

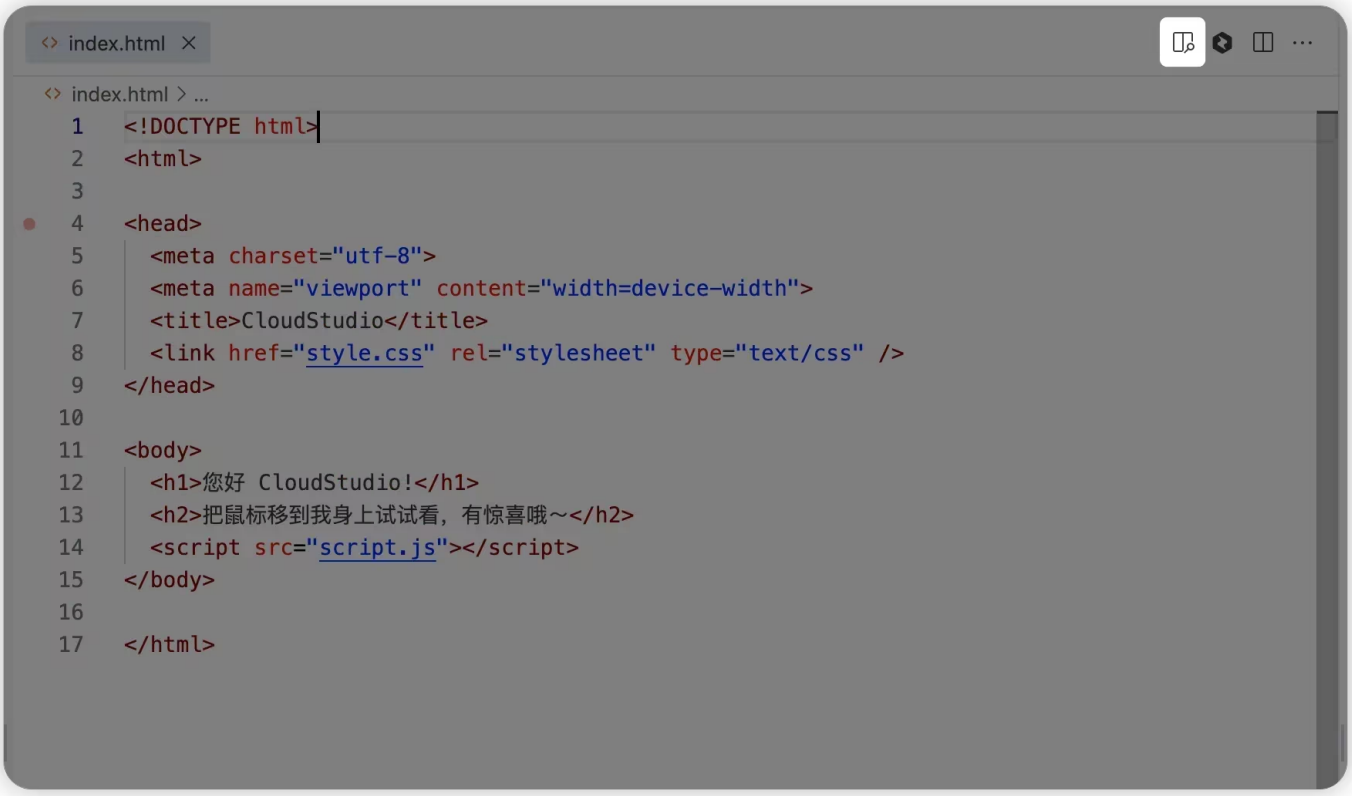
一、准备：安装 Live Server 插件

1. 打开左侧 **扩展（Extensions）**。
2. 搜索并安装：`Cloud Studio Live Server`。
3. 安装完成后，重载窗口（如有提示）。

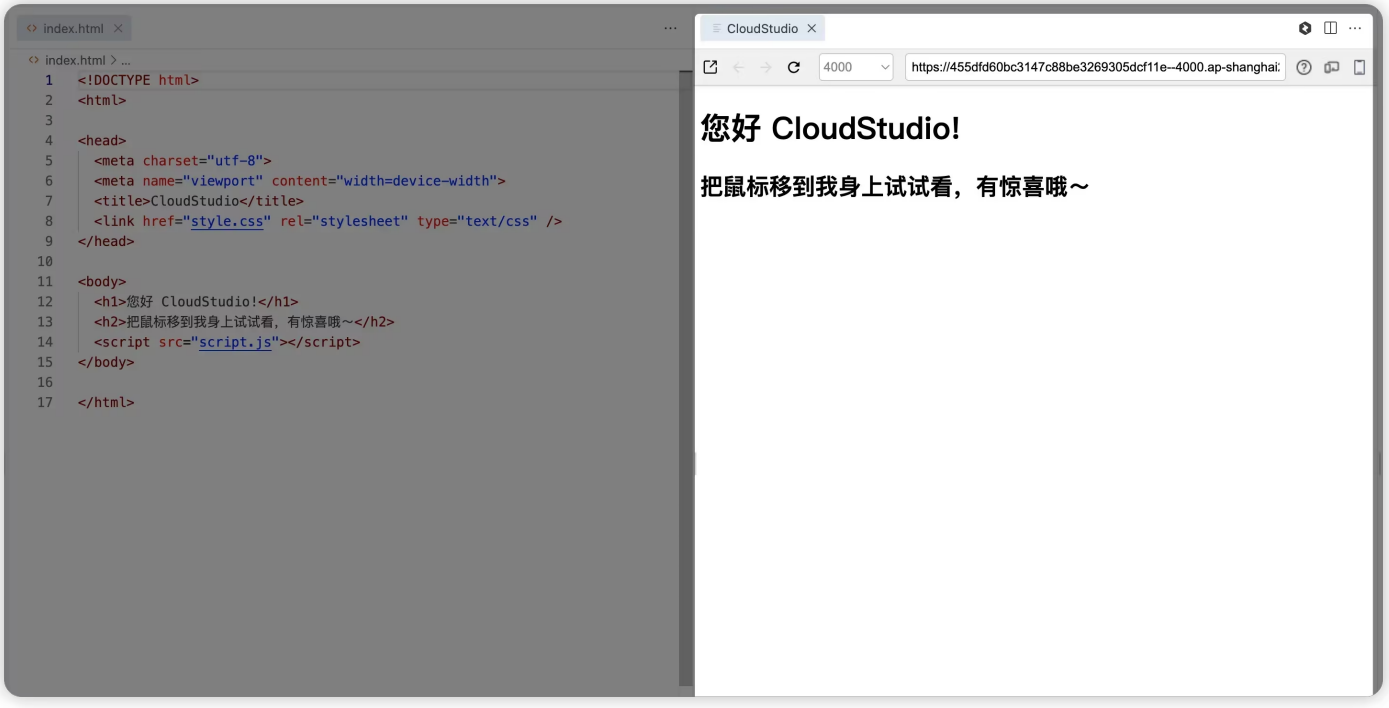


二、启动与打开

在 HTML 文件的右上角点击 `Show Preview` 图标后即可。



启动成功后，通常会弹出一个新的标签页 `CloudStudio` 。



常见问题

页面能打开，但刷新后资源 404 / 路径不对？

- **检查资源引用路径**：静态站点更推荐使用相对路径（如 `./assets/app.css` ）。

- **确认打开的目录**：尽量从项目根目录打开工作区，并确保 HTML 文件与资源路径一致。

为什么没有 `Show Preview` 图标？

- 目前测试时发现只有 HTML 文件才会出现这个图标，其他文件不会出现。
- 并且标签页也只能有一个 HTML 文件，如果有其他标签页，那么这个图标就不会出现。

> **说明**
> 如果您使用了 Live Server 插件，那么在您保存文件时，Live Server 会自动启动一个本地服务器，并在浏览器中打开您的 HTML 文件。

其他问题

最近更新时间：2026-05-19 18:07:22

turtle 为什么不能用？

`turtle` 依赖交互式图形界面渲染，在 CloudStudio 环境中**暂不支持**。

如何拼出预览访问地址？

预览地址可用环境变量拼接（把 `${PORT}` 换成您的服务端口）：

```
https://${X_IDE_SPACE_KEY}-
-${PORT}.${X_IDE_SPACE_REGION}.${X_IDE_SPACE_HOST}
```

需要查看当前空间变量时，可在终端输出：

```
echo "https://${X_IDE_SPACE_KEY}-
-${PORT}.${X_IDE_SPACE_REGION}.${X_IDE_SPACE_HOST}"
```

端口转发支持哪些？

- **TCP / UDP**：暂不支持。
- **WebSocket**：支持，客户端请使用 `wss://`。
- **HTTPS 服务**：平台 HTTPS 网关通常只代理工作空间内的**非 HTTPS 服务**；如果您的服务本身是 HTTPS，可能无法通过预览转发。

是否支持 SSH 直连？

目前**不支持直接 SSH 连接**。

Vite 页面报 “Blocked request” 怎么办？

通常是访问校验导致。建议在 `vite.config.js` 使用：

```
server: {
```

```
  host: true,
  port: 3000,
  strictPort: false
  // 通常无需配置 allowedHosts
}
```

> **提示**
> 如果您手动配置了 `allowedHosts`，它会覆盖默认行为；建议优先使用默认配置。

内置 CodeBuddy 提示“本月免费额度已用完”怎么办?

当出现如下提示时，说明本月的 CodeBuddy 免费配额（每月 500 credit）已耗尽：

解决方式：

- 等待下月额度自动刷新；
- 或升级套餐以获取更多配额。