Module 1---Research Agent Tutorial

一个典型的数据驱动任务Research流程为：

1. 梳理任务SoW (Statement of Work)，包括任务简介、任务Benchmark（训练集和测试集）、测评指标和脚本等；
2. 综述完成任务相关工作，对现有技术进行分类；
3. 实验筛选，将检索到的相关工作在已有任务Benchmark上进行适配和复现，选出效果最佳的工作；
4. 在已有工作上进行优化以进一步提升方案性能。

Research Agent的目标是将上述Research流程尽可能自动化由Agent完成，本文以LLM merge任务为例，以下是一个完整的Research Agent tutorial。

1. 面向Agent的任务SoW

任务简介

动机：

Training high-performing large language models (LLMs) from scratch is a notoriously expensive and difficult task, costing hundreds of millions of dollars in compute alone. These pretrained LLMs, however, can cheaply and easily be adapted to new tasks via fine-tuning, leading to a proliferation of models that suit specific use cases. Recent work has shown that specialized fine-tuned models can be rapidly merged to combine capabilities and generalize to new skills.

任务：

The competition will provide the participants with a list of expert models that have already been trained on a task-specific dataset. All of these models will be publicly available on the Hugging Face Model Hub with licenses that permit their use for research purposes. These models can either be fully fine-tuned models or models obtained by parameter-efficient fine-tuning methods such as LoRA. Models on this list will be required to satisfy the following criteria: (1) model size ≤ 8B parameters, and (2) model with licenses compatible with research use (e.b., MIT, Apache 2 etc).

The goal of this competition is to re-use the provided models to create a generalist model that can perform well on a wide variety of skills like reasoning, coding, maths, chat, and tool use.

This list of models will include popular pre-trained models such as LLaMA-7B, Mistral-7B, and Gemma-7B.

Allowed Models：

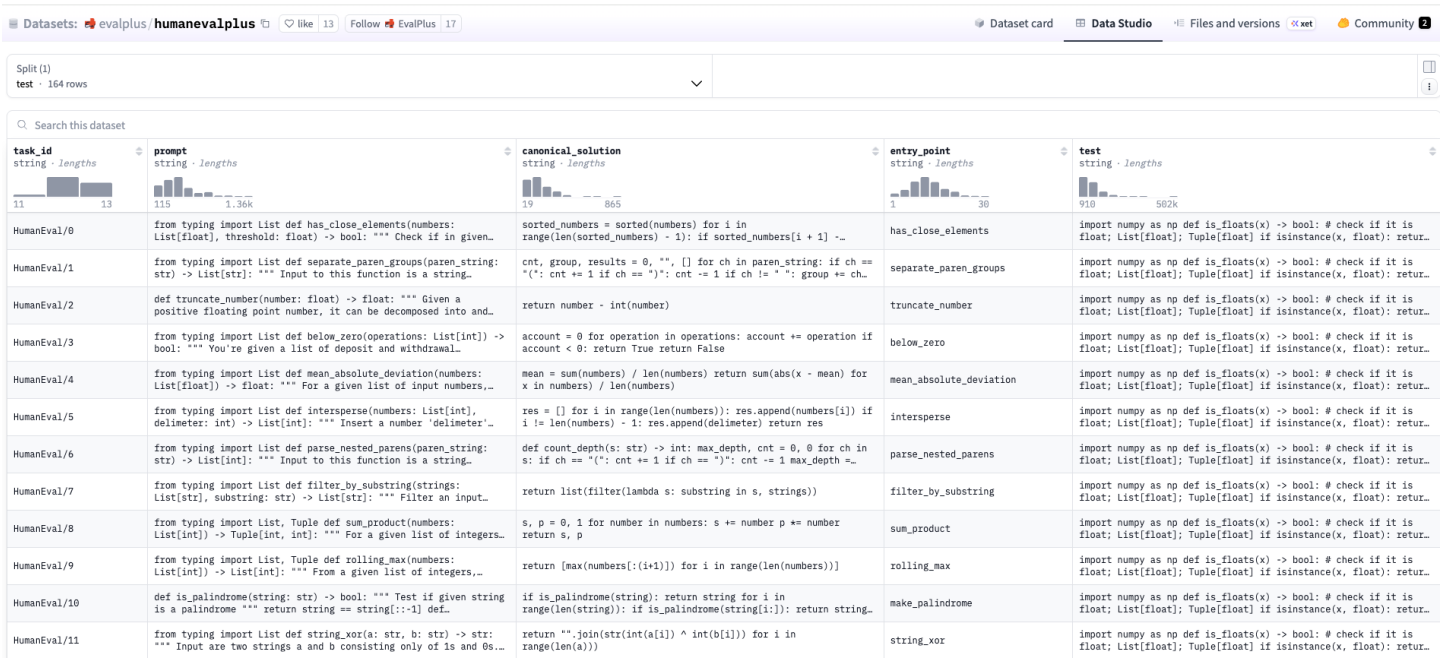
本次我们对允许的models做了限定，以减少任务的复杂度：The models to be merged are `meta-llama/Meta-Llama-3-8B-Instruct` and `MaziyarPanahi/Llama-3-8B-Instruct-v0.8`.

Benchmark

HumanEval+：

数据地址：https://huggingface.co/datasets/evalplus/humanevalplus/viewer?views%5B%5D=test

数据构成：



task_id string · lengths	prompt string · lengths	canonical_solution string · lengths	entry_point string · lengths	test string · lengths
HumanEval/0	from typing import List def has_close_elements(numbers: List[float], threshold: float) -> bool: """ Check if in given…	sorted_numbers = sorted(numbers) for i in range(len(sorted_numbers) - 1): if sorted_numbers[i + 1] -…	has_close_elements	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/1	from typing import List def separate_paren_groups(paren_string: str) -> List[str]: """ Input to this function is a string…	cnt, group, results = 0, "", [] for ch in paren_string: if ch == "(": cnt += 1 if ch == ")": cnt -= 1 if ch != " ": group += ch…	separate_paren_groups	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/2	def truncate_number(number: float) -> float: """ Given a positive floating point number, it can be decomposed into and…	return number - int(number)	truncate_number	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/3	from typing import List def below_zero(operations: List[int]) -> bool: """ You're given a list of deposit and withdrawal…	account = 0 for operation in operations: account += operation if account < 0: return True return False	below_zero	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/4	from typing import List def mean_absolute_deviation(numbers: List[float]) -> float: """ For a given list of input numbers,…	mean = sum(numbers) / len(numbers) return sum(abs(x - mean) for x in numbers) / len(numbers)	mean_absolute_deviation	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/5	from typing import List def intersperse(numbers: List[int], delimeter: int) -> List[int]: """ Insert a number 'delimeter'…	res = [] for i in range(len(numbers)): res.append(numbers[i]) if i != len(numbers) - 1: res.append(delimeter) return res	intersperse	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/6	from typing import List def parse_nested_parens(paren_string: str) -> List[int]: """ Input to this function is a string…	def count_depth(s: str) -> int: max_depth, cnt = 0, 0 for ch in s: if ch == "(": cnt += 1 if ch == ")": cnt -= 1 max_depth =…	parse_nested_parens	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/7	from typing import List def filter_by_substring(strings: List[str], substring: str) -> List[str]: """ Filter an input…	return list(filter(lambda s: substring in s, strings))	filter_by_substring	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/8	from typing import List, Tuple def sum_product(numbers: List[int]) -> Tuple[int, int]: """ For a given list of integers…	s, p = 0, 1 for number in numbers: s += number p *= number return s, p	sum_product	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/9	from typing import List, Tuple def rolling_max(numbers: List[int]) -> List[int]: """ From a given list of integers,…	return [max(numbers[:(i+1)]) for i in range(len(numbers))]	rolling_max	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/10	def is_palindrome(string: str) -> bool: """ Test if given string is a palindrome """ return string == string[::-1] def…	if is_palindrome(string): return string for i in range(len(string)): if is_palindrome(string[i:]): return string…	make_palindrome	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…
HumanEval/11	from typing import List def string_xor(a: str, b: str) -> str: """ Input are two strings a and b consisting only of 1s and 0s.…	return "".join(str(int(a[i]) ^ int(b[i])) for i in range(len(a)))	string_xor	import numpy as np def is_floats(x) -> bool: # check if it is float; List[float]; Tuple[float] if isinstance(x, float): retur…

task_id表示任务的ID，prompt表示题目（通常直接请求大模型获取答案），entry_point是唯一标记，canonica_solution是参考答案，test是测试单元。

在原始 HumanEval 164 道 Python 题目基础上，对每道题新增约 80 倍测试用例，从而得到 HumanEval+，即用海量、高覆盖率的单元测试严格验证代码功能正确性。

评估指标

指标：Pass @ 1

1）背景：为什么要用 *pass @ k*

传统代码评估用 BLEU／Edit Distance 等“文本相似度”指标，但这些**无法保证语义等价**。于是近几年工作（Kulal 2019；Chen 2021）转向**功能正确性**：只认“是否通过全部单元测试”。

当一次对同一题采样 *k* 份代码，只要 **至少 1 份** 通过，就视为“题目被解决”，这一概率就是 *pass @ k* ([arXiv][1])。

2） 一般公式（无偏估计器）

设

记号	含义
n	对该题实际生成的候选总数（≥ k）
c	其中通过全部测试的候选数
k	我们关心的排名阈值

无偏估计器（Chen 2021 式 (1)）：

$$\text{pass@}k := \mathop{\mathbb{E}}_{\text{Problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]$$

直观上，分子是“从不正确样本里抽到 *k* 份全错”的组合数，分母是“从全部 *n* 份里抽 *k* 份”的组合数；用 1 减去它即得到“至少 1 份对”的概率。

作者证明该表达式无偏，且比朴素估计 $1-(1-\hat{p})^k$ 方差更低，避免随着 *n* 变化而系统性低估（[arXiv][1]）。

3） *pass @ 1* 的特化

令 k = 1：

$$\text{pass@}1 = 1 - \frac{\binom{n-c}{1}}{\binom{n}{1}} = \frac{c}{n}.$$

若一次仅生成 1 份代码（n = 1），此式退化为“该份代码是否通过”——也是最常见的实验设定。

若一次生成 n > 1 份，仍可算出 pass @ 1 = c/n；它表示“随机从这 n 份里抽 1 份就通过”的期望成功率，而不是“第 1 个候选是否通过”。

4） 计算示例（官方实现）

```
def pass_at_k(n: int, c: int, k: int):
  if n - c < k:
    return 1.0
  import numpy as np
  return 1 - np.prod(1 - k / np.arange(n-c+1, n+1))
```

> 这是 Codex 论文给出的 **数值稳定实现**，避免大阶乘溢出 ([arXiv][1])。

单元测试覆盖度：若测试不全，*pass @ 1* 可能“漏网”错误实现——Humaneval+ 之所以把每题测试扩大 × 80，就是为提高置信度。

5） 小结

pass @ 1 是 **概率论严格推导的无偏估计量**：衡量“从模型当前输出中随机抽 1 份，就能一次性通过全部单元测试”的可信概率。它兼具

可解释性（直接映射到开发者体验）、

公平性（无偏、与样本量解耦），

可操作性（一行 NumPy 代码即可计算）。

因此已成为 HumanEval、Humaneval⁺、MBPP⁺ 等主流代码基准的核心指标，并被 GitHub Copilot、OpenAI Codex、BigCode StarCoder 等系统广泛采用。

[1]: https://arxiv.org/pdf/2107.03374 "Evaluating Large Language Models Trained on Code"

测评脚本

代码块

```
conda create -n bigcode python=3.10.9
conda activate bigcode

git clone https://github.com/bigcode-project/bigcode-evaluation-harness.git
cd bigcode-evaluation-harness

pip install -e .
pip install -U torch>=2.2 torchvision torchaudio
pip install numpy==1.24.1

CUDA_VISIBLE_DEVICES=1 accelerate launch  main.py \
```

```
    --model $MODEL \    # 替换为你的模型地址
    --max_length_generation 512 \
    --precision bf16 \
    --tasks humanevalplus \
    --temperature 0.2 \
    --n_samples 10 \
    --batch_size 10 \
    --allow_code_execution \
    --metric_output_path $OUTPUT_PATH/code_eval.json \ # 替换为你的输出地址
    --use_auth_token
```

得出评估结果：

pass@1: 0.5060975609756098

code_eval.json样例如下：

代码块

```
{
  "humanevalplus": {
    "pass@1": 0.5060975609756098,
    "pass@10": 0.6646341463414634
  },
  "config": {
    "prefix": "",
    "do_sample": true,
    "temperature": 0.2,
    "top_k": 0,
    "top_p": 0.95,
    "n_samples": 10,
    "eos": "<|endoftext|>",
    "seed": 0,
    "model": "../models--meta-llama--Meta-Llama-3-8B-
Instruct/snapshots/e1945c40cd546c78e41f1151f4db032b271faeaa",
    "modeltype": "causal",
    "peft_model": null,
    "revision": null,
    "use_auth_token": true,
    "trust_remote_code": false,
    "tasks": "humanevalplus",
    "instruction_tokens": null,
    "batch_size": 10,
    "max_length_generation": 512,
    "precision": "bf16",
    "load_in_8bit": false,
    "load_in_4bit": false,
```

```
    "left_padding": false,
    "limit": null,
    "limit_start": 0,
    "save_every_k_tasks": -1,
    "postprocess": true,
    "allow_code_execution": true,
    "generation_only": false,
    "load_generations_path": null,
    "load_data_path": null,
    "metric_output_path": "code_eval.json",
    "save_generations": false,
    "load_generations_intermediate_paths": null,
    "save_generations_path": "generations.json",
    "save_references": false,
    "save_references_path": "references.json",
    "prompt": "prompt",
    "max_memory_per_gpu": null,
    "check_references": false
  }
```


2. 任务相关工作综述

本小节的目标是找到尽可能系统、全面的相关工作及其开源项目地址，以供后续Experiment Agent实验，选出最佳工作

利用Deep Research查询相关工作

登入google gemini： https://gemini.google.com/app

【选择Deep Reseach】

输入下述提示词：

代码块

```
模型融合(Model Merge)是机器学习领域一种高效的赋能技术，它无需收集原始训练数据，也不需要昂贵的计算资源。请帮我综述现有大模型融合的相关工作，要求：
  1、尽可能多的检索相关工作，按照技术特征进行分类，并讨论这些方法的优缺点；
  2、如果有开源项目，请帮我梳理出开源地址；
  3、仅限于大模型的模型融合方法。
```

生成综述报告如下：

大模型融合技术综述_.pdf

人工筛选

由于现有Deep Research工具生成的综述报告，只提供了有限的几个代表工作，为了进一步检索更加系统和全面的工作，还需要人工基于检索到的综述报告去进行梳理和查找，一般步骤如下：
1、以Deep Research检索到的工作为基础，查看里面是否检索到本领域综述论文，一般成熟且系统研究的领域会有最新综述论文，研究人员会将已有工作进行梳理和分类；

2、如果Deep research没有检索到综述论文，则需要人工确认是否被落下，人工在Google scholar、Arxiv等平台手动检索确认；

3、如果人工确认没有相关领域综述，则以Deep Research检索到的论文为起点，分析论文里的相关工作，可按照Deep research给出的分类，人工检索更多论文和开源项目。

在本任务中，Deep research检索到了本领域的最新综述论文：

《Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications and Opportunities》

论文地址：https://arxiv.org/pdf/2408.07666

该综述已经将每项工作按照技术特征进行分类，开源地址为：
https://github.com/EnnengYang/Awesome-Model-Merging-Methods-Theories-Applications.git

根据综述可知，目前LLM Merge领域的相关工作分为以下几类（每类工作仅列举5项开源项目作为后续实验选项）：

基于权重的合并方法

这类方法旨在为不同的模型或任务向量分配不同的重要性权重，从而更有效地合并模型。

代表工作：

- RegMean++: Enhancing Effectiveness and Generalization of Regression Mean for Model Merging

论文地址：https://arxiv.org/pdf/2508.03121

github地址：https://github.com/nthehai01/RegMean-plusplus

- CALM: Consensus-Aware Localized Merging for Multi-Task Learning （ICML 2025）

论文地址：https://arxiv.org/pdf/2506.13406

github地址：https://github.com/yankd22/CALM/tree/main

- Arcee’s MergeKit: A Toolkit for Merging Large Language Models

论文地址：https://arxiv.org/pdf/2403.13257

github地址：https://github.com/arcee-ai/MergeKit

- Evolutionary Optimization of Model Merging Recipes

论文地址：https://arxiv.org/pdf/2403.13187

github地址：https://github.com/SakanaAI/evolutionary-model-merge

- *Sens-Merging: Sensitivity-Guided Parameter Balancing for Merging Large Language Models*

论文地址：https://arxiv.org/pdf/2502.12420

github未发布，这篇文章的单位之一是华为诺亚方舟实验室，华为开源了一个toolkit（https://github.com/hahahawu/Long-to-Short-via-Model-Merging.git），将一系列llm merging的论文集成在这个toolkit里面。但这篇文章的代码还没集成进来。可测试这个toolkit已集成的其他方法：

  - Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch https://github.com/yule-BUAA/MergeLM
  - TIES-Merging: Resolving Interference When Merging Models https://github.com/prateeky2806/ties-merging

基于子空间的合并方法

这类方法旨在将多个模型投影到稀疏子空间中进行合并，从而减轻任务间的干扰。

代表作：

- Training-free LLM Merging for Multi-task Learning （ACL2025）

论文：https://arxiv.org/pdf/2506.12379

github地址：https://github.com/Applied-Machine-Learning-Lab/Hi-Merging

- Adaptive LoRA Merge with Parameter Pruning for Low-Resource Generation （ACL2025）

论文：https://arxiv.org/pdf/2505.24174

github地址:https://github.com/mr0223/adaptive_lora_merge

- LoRI: Reducing Cross-Task Interference in Multi-Task LowRank Adaptation (COLM2025)

论文：https://arxiv.org/pdf/2504.07448

github地址：https://github.com/juzhengz/LoRI

- AdaRank: Adaptive Rank Pruning for Enhanced Model Merging

论文：https://arxiv.org/pdf/2503.22178

github地址：https://github.com/david3684/AdaRank

- STAR: Spectral Truncation and Rescale for Model Merging（NAACL 2025）

论文：https://arxiv.org/pdf/2502.10339

github地址：https://github.com/IBM/STAR

- Task Vector Quantization for Memory-Efficient Model Merging （ICCV 2025）

论文：https://arxiv.org/pdf/2503.06921

github地址：https://github.com/AIM-SKKU/TVQ

基于路由的合并方法

这类方法是一种动态合并策略，根据输入样本的特征在推理阶段动态地决定如何合并模型。

代表工作：

- Dynamic Fisher-weighted Model Merging via Bayesian Optimization （NAACL2025）

论文：https://arxiv.org/pdf/2504.18992

github地址：https://github.com/sanwooo/df-merge

- MASS: MoErging through Adaptive Subspace Selection

论文：https://arxiv.org/pdf/2504.05342

github地址：https://github.com/crisostomi/mass

- CAMEX: CURVATURE-AWARE MERGING OF EXPERTS (ICLR2025)

论文：https://arxiv.org/pdf/2502.18821

github地址：https://github.com/kpup1710/CAMEx

- DAWIN: TRAINING-FREE DYNAMIC WEIGHT INTER - POLATION FOR ROBUST ADAPTATION (ICLR 2025)

论文：http://arxiv.org/pdf/2410.03782

github地址：https://github.com/naver-ai/dawin

- Learning to Route Among Specialized Experts for Zero-Shot Generalization

论文:https://arxiv.org/pdf/2402.05859

Github地址：https://github.com/r-three/phatgoose

基于后校准的合并方法

代表作

- Why Train Everything? Tint a Single Layer for Multi-task Model Merging

论文：https://arxiv.org/pdf/2412.19098

github地址:https://github.com/AIM-SKKU/ModelTinting

- Fine-tuning Aligned Classifiers for Merging Outputs: Towards a Superior Evaluation Protocol in Model Merging （ICML 2024）

论文：https://arxiv.org/pdf/2412.13526

github地址：https://github.com/fskong/FT-Classifier-for-Model-Merging

- SurgeryV2: Bridging the Gap Between Model Merging and Multi-Task Learning with Deep Representation Surgery

论文：https://arxiv.org/pdf/2410.14389

github地址：https://github.com/EnnengYang/SurgeryV2

- Representation Surgery for Multi-Task Model Merging （ICML 2024）

论文：https://openreview.net/pdf/602906ec02919eb95d78d634321fcba1b68a2f03.pdf

github地址：https://github.com/EnnengYang/RepresentationSurgery

小结

综述任务相关工作对于领域研究非常重要，现有Deep Research可以根据现有技术特征进行分类，但是仅仅输出代表性的几项工作，而Research Agent的目标是找出尽可能系统、全面的找到本任务相关

的所有工作，通过实验从其中选择最佳工作，当前的Deep Research无法满足要求，仅仅只能作为相关工作检索的入口。

> ⚽ **思考：**
>
> 如何优化现有Deep Research工作以使其满足Research Agent要求？

3. Experiment Agent实验适配

openhands环境安装

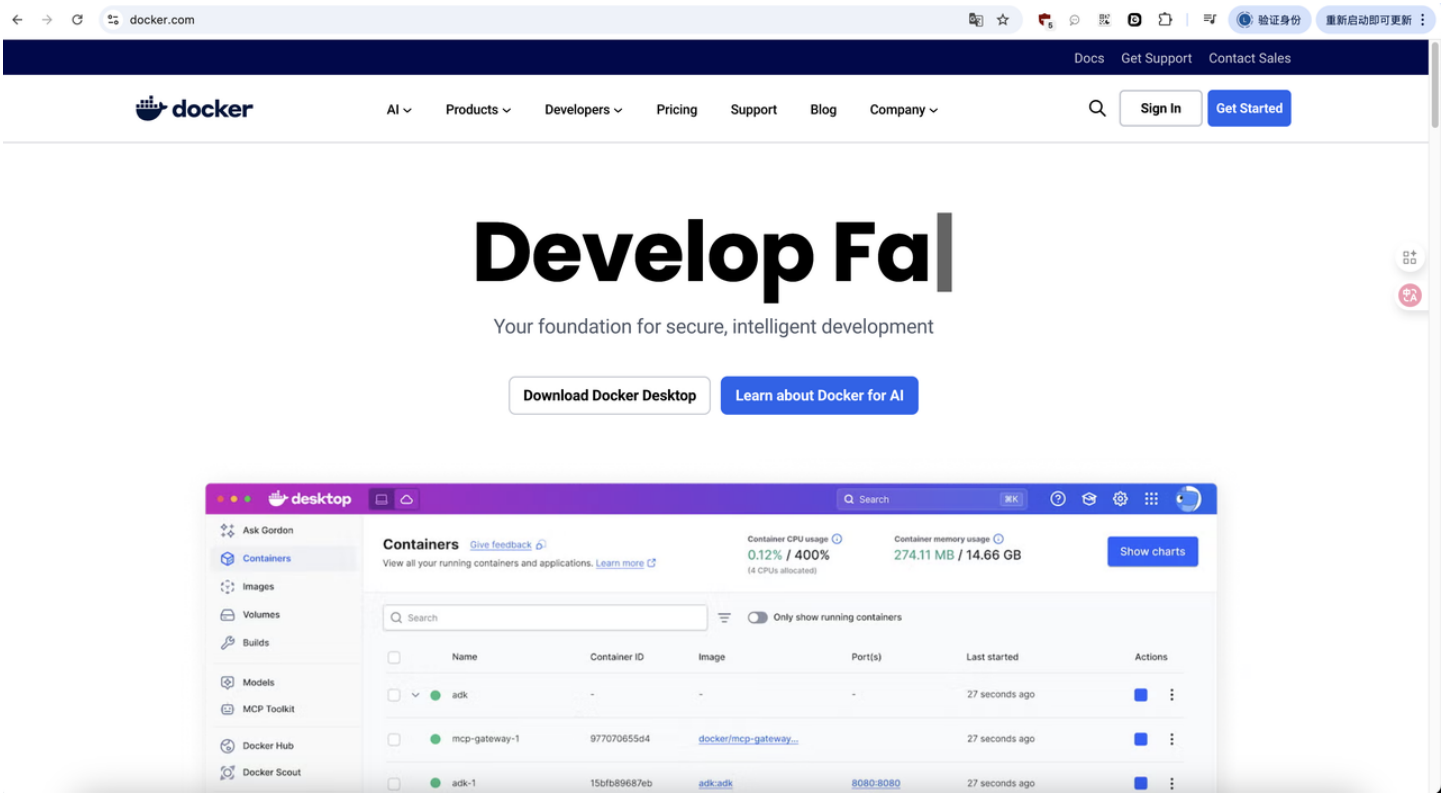
依赖准备

Docker 安装

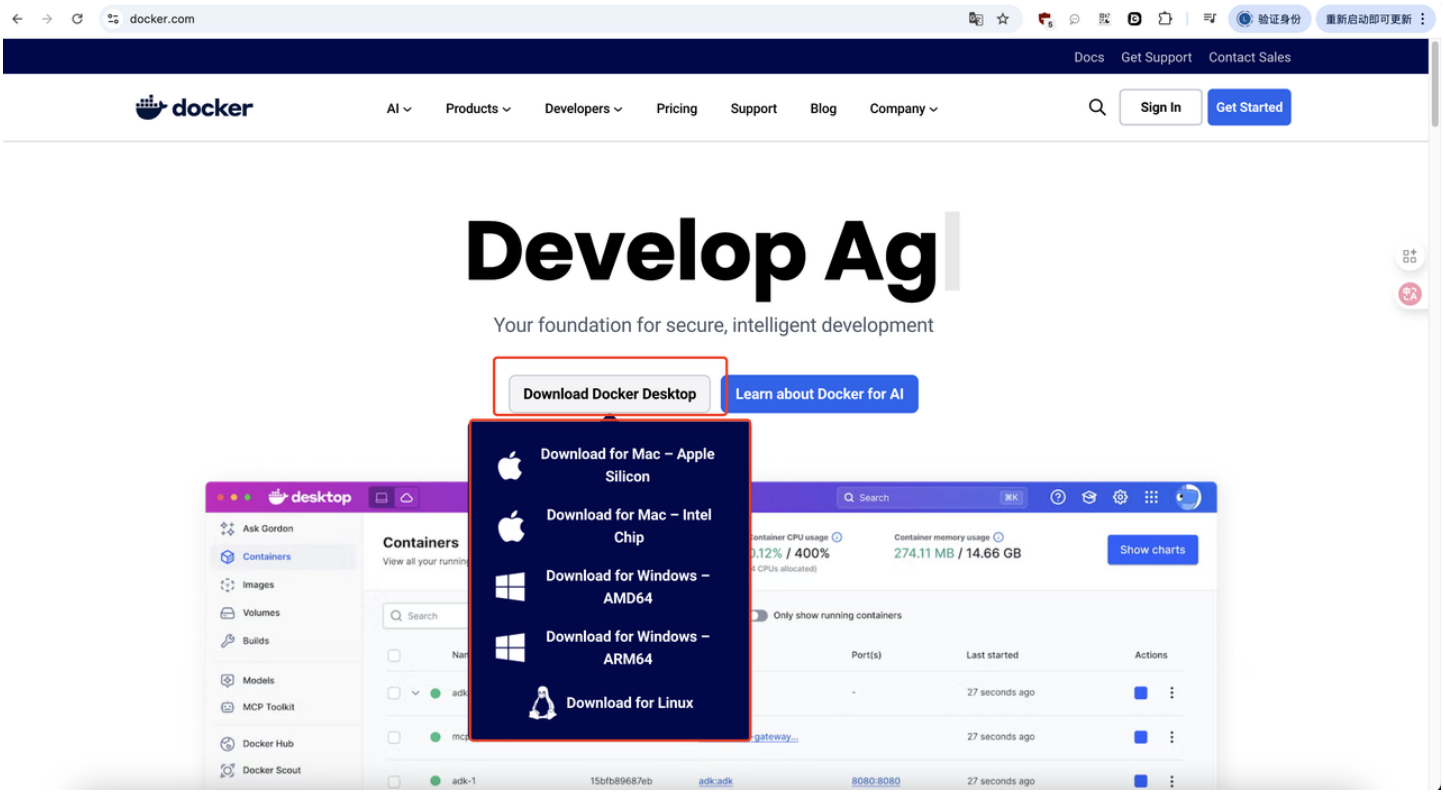
软件下载

官网地址: https://www.docker.com/

点击地址进入官网



根据自身电脑版本下载安装包

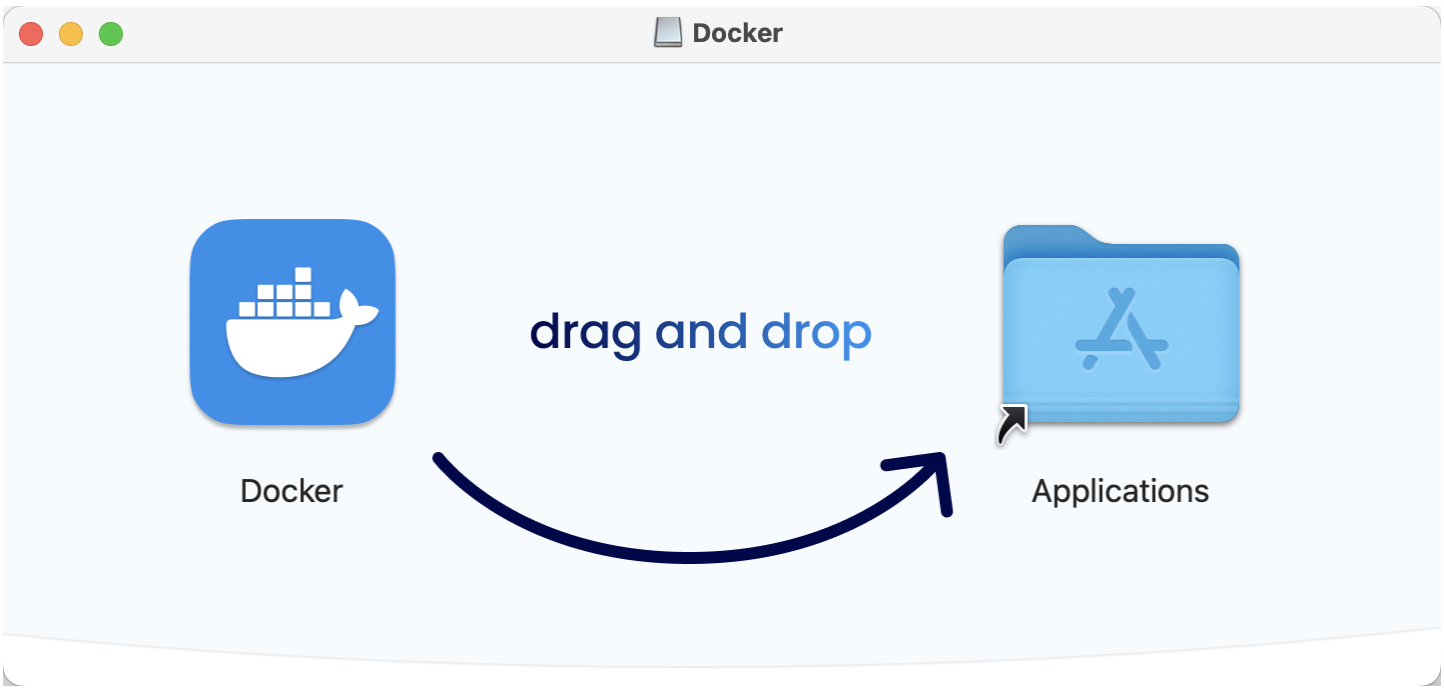


软件安装

Windows安装参考视频

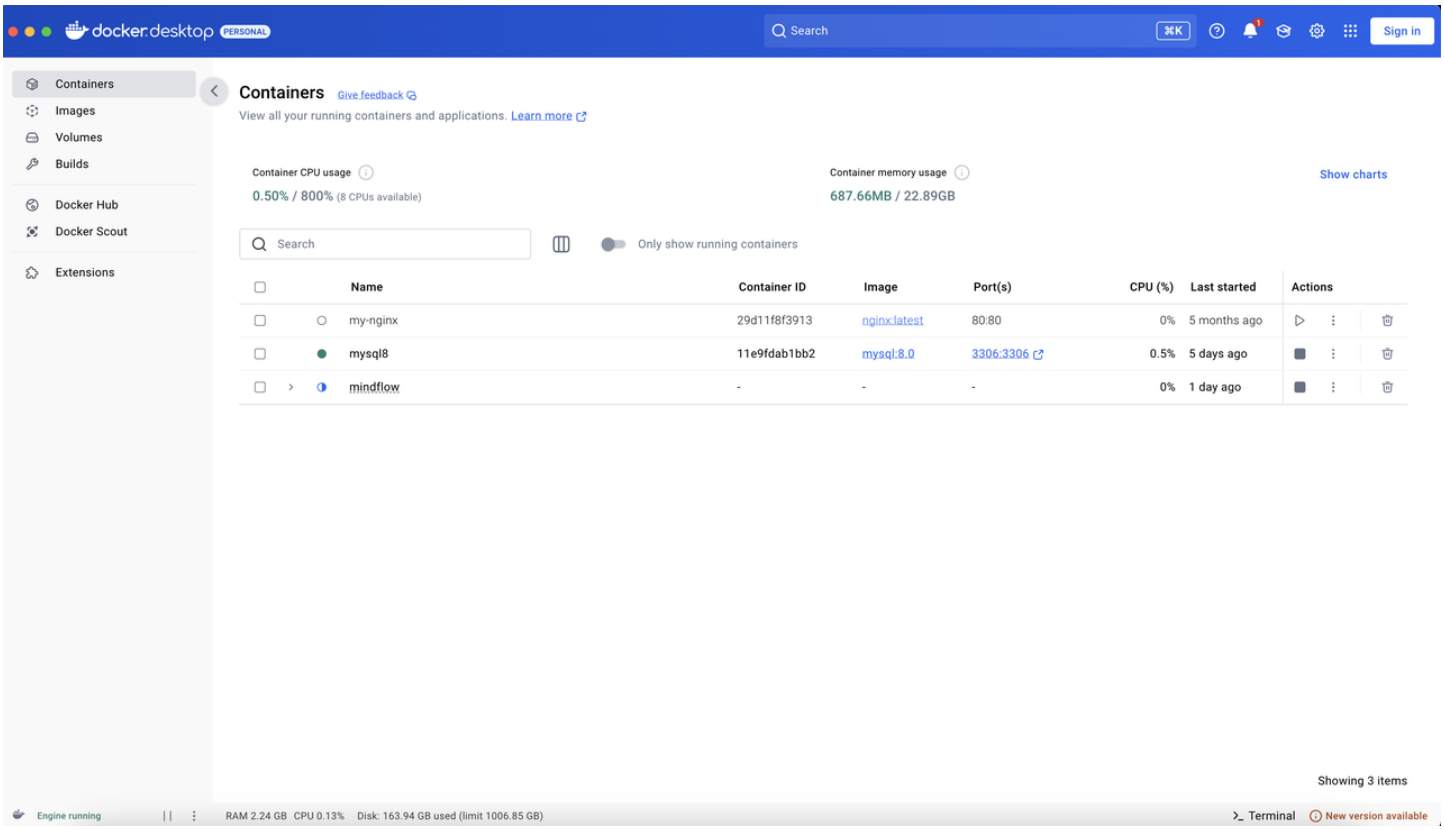
MacOS:

打开安装包,拖拽图标到Applications



软件启动

点击Docker Desktop应用打开界面,启动docker完成

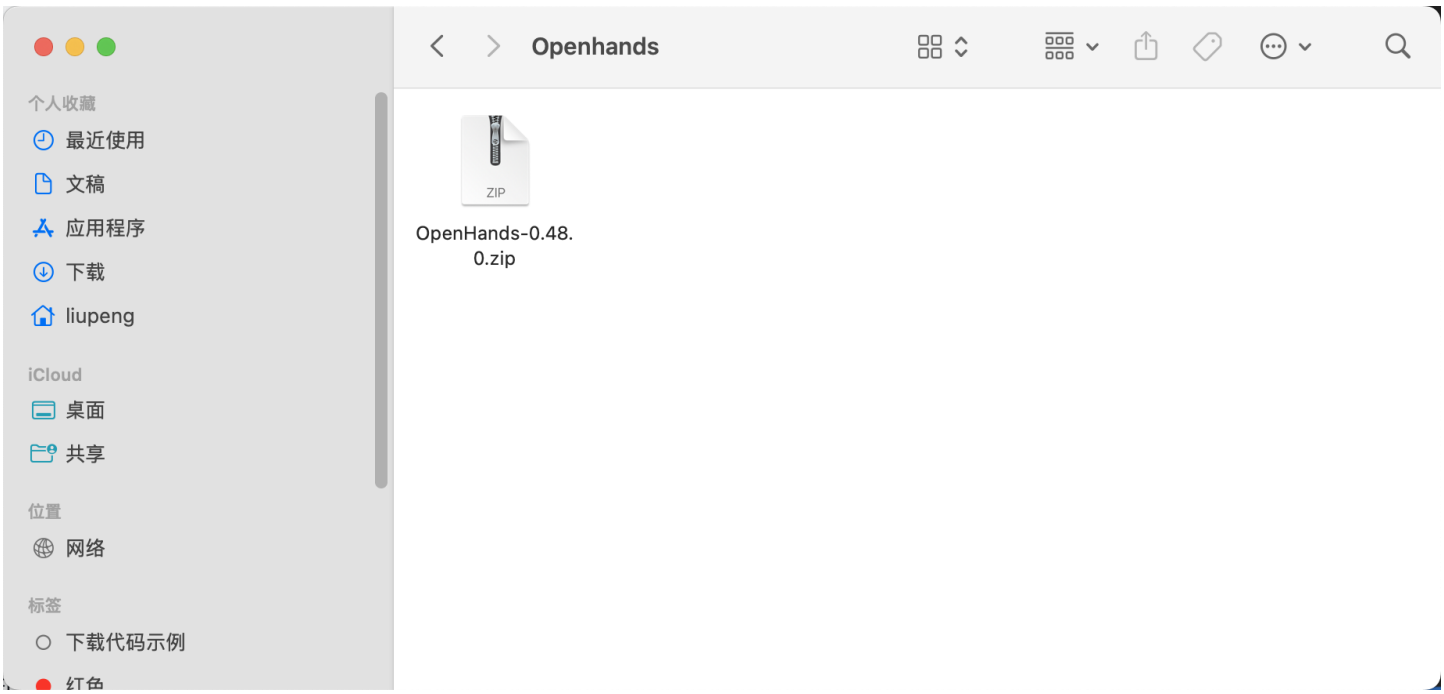


Openhands-0.48.0

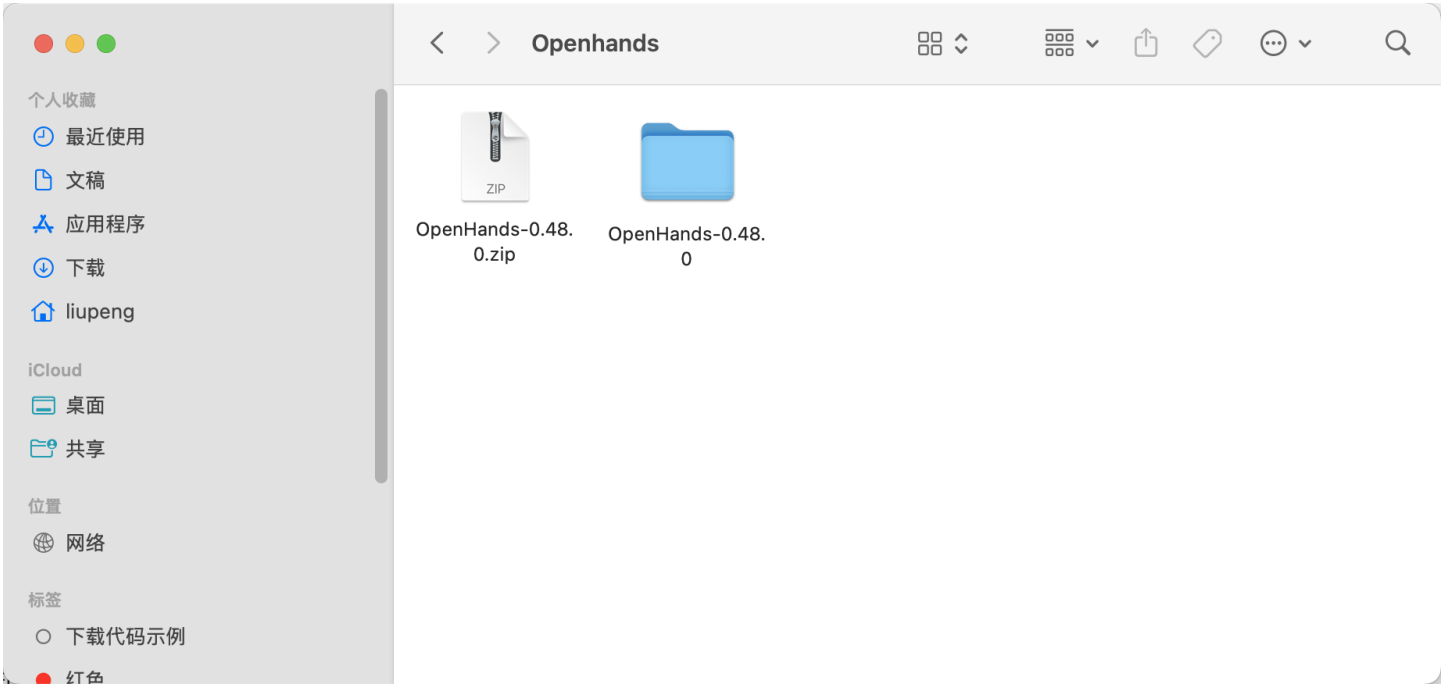
源码下载:

OpenHands-0.48.0.zip
45.22MB

点击下载地址下载源码压缩包



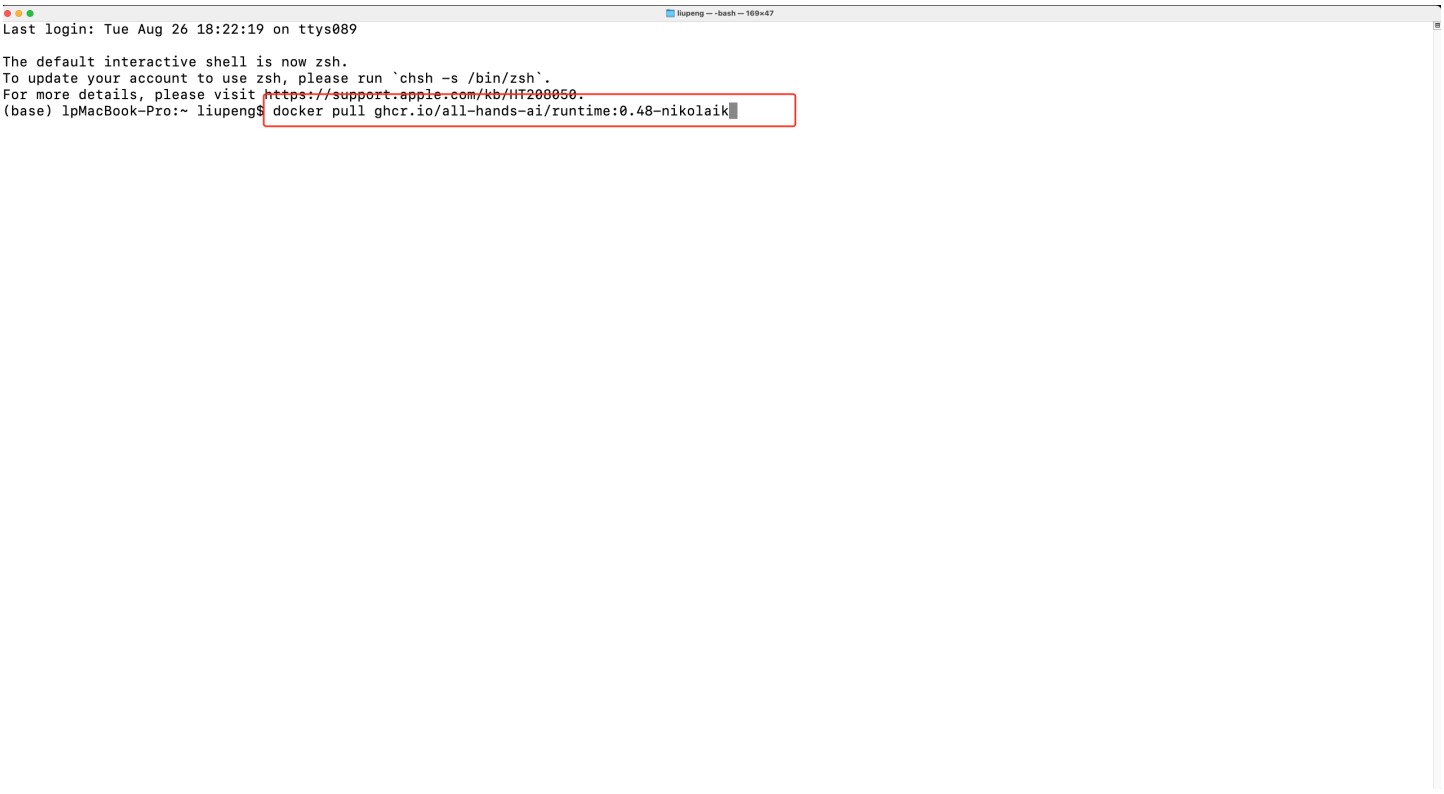
解压缩文件夹



镜像准备:

终端执行命令

```
代码块
1   docker pull ghcr.io/all-hands-ai/runtime:0.48-nikolaik
```



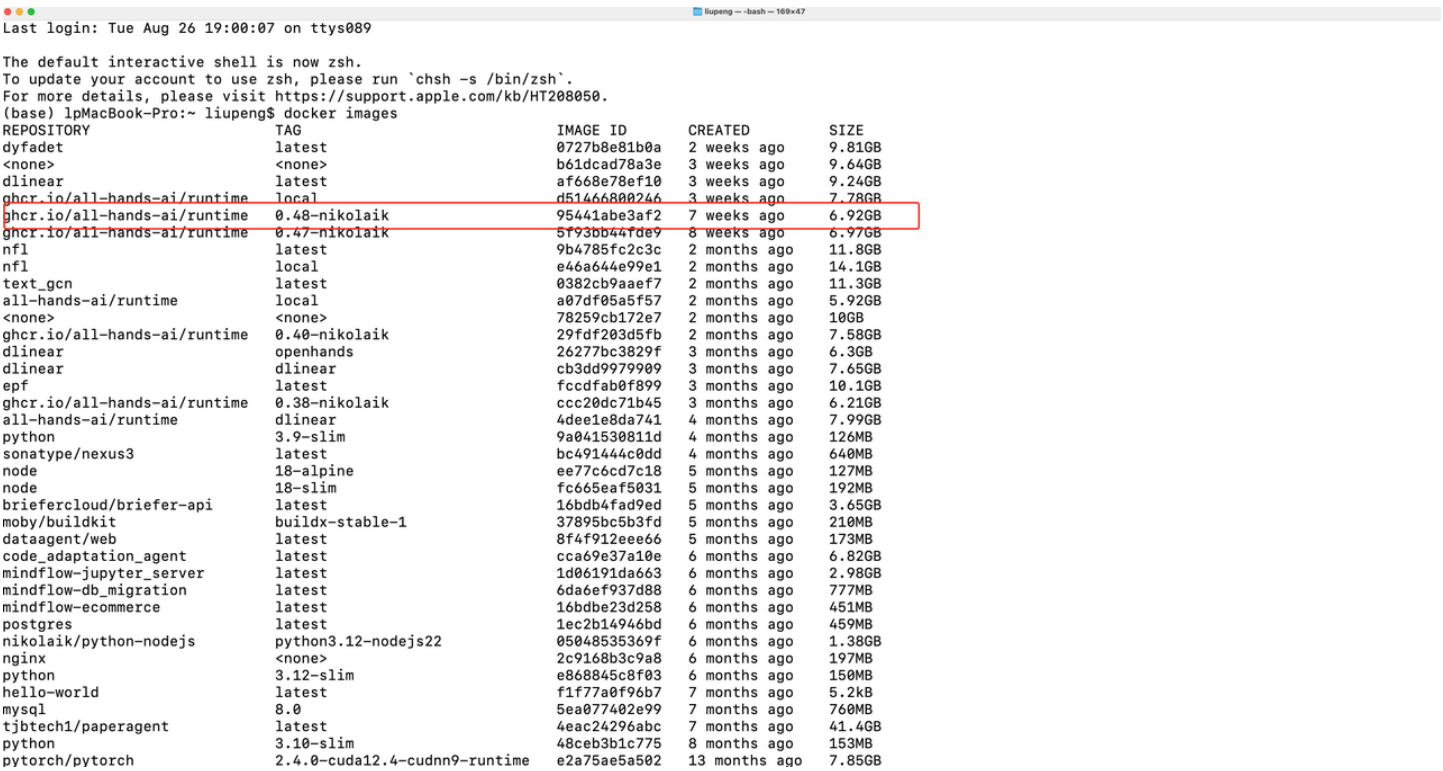
```
(base) lpMacBook-Pro:~ liupeng$ docker pull ghcr.io/all-hands-ai/runtime:0.48-nikolaik
0.48-nikolaik: Pulling from all-hands-ai/runtime
8fbf1dd6492c: Already exists
c5e137b9ec17: Already exists
14f5719d6358: Already exists
78f00d2fce16: Already exists
7b9f5f869a6e: Already exists
0bbff3be7591: Already exists
34294ea665eb: Already exists
e2de77e4ff9b: Already exists
4f4fb700ef54: Already exists
2da3aa321bc6: Already exists
6eb5cf8c1997: Already exists
c0b6e3c72021: Already exists
920006c4bdcd: Already exists
d24c281f6d77: Already exists
a816b2ad2180: Already exists
375b3d326689: Already exists
ab7f3a44efe3: Already exists
85d7bf06d575: Already exists
200f1be9d431: Already exists
94590749f5d6: Already exists
3e8fb36c59c0: Already exists
02e6f9e9355c: Already exists
d3d6521450f6: Already exists
d5af86ea2209: Already exists
d6c42738892c: Already exists
e16e4e5f3abb: Already exists
3de1a7e6b119: Already exists
4f19ecac4e35: Already exists
16cda5196016: Already exists
701bb818876e: Already exists
Digest: sha256:9dd8fc082ffd875b502deef0b78f2d4ded058cb831b97f73704761ac0ceccc6a
Status: Downloaded newer image for ghcr.io/all-hands-ai/runtime:0.48-nikolaik
ghcr.io/all-hands-ai/runtime:0.48-nikolaik
```

下载完成后

执行镜像查看命令查看镜像

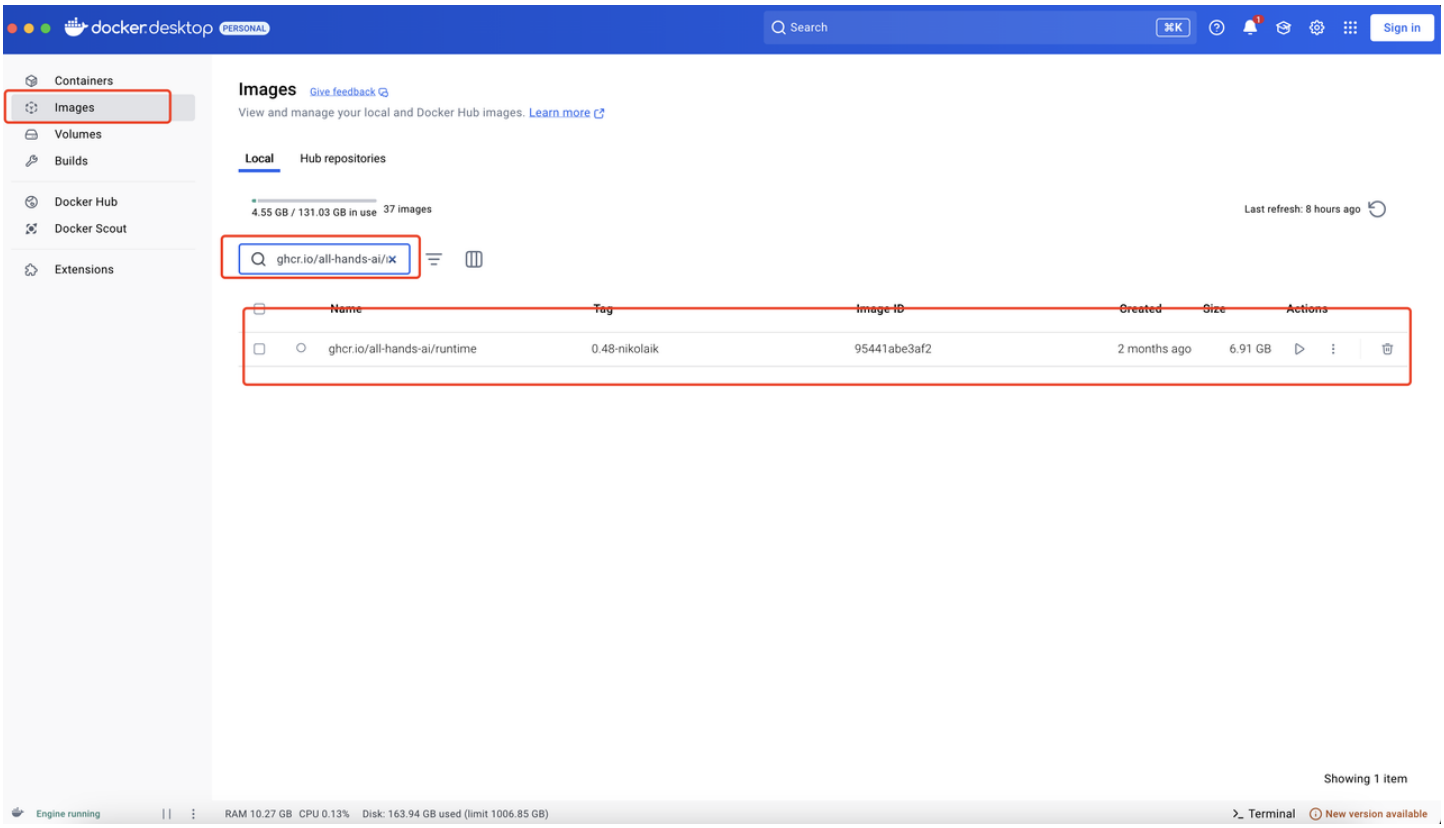
代码块

```
docker images
```



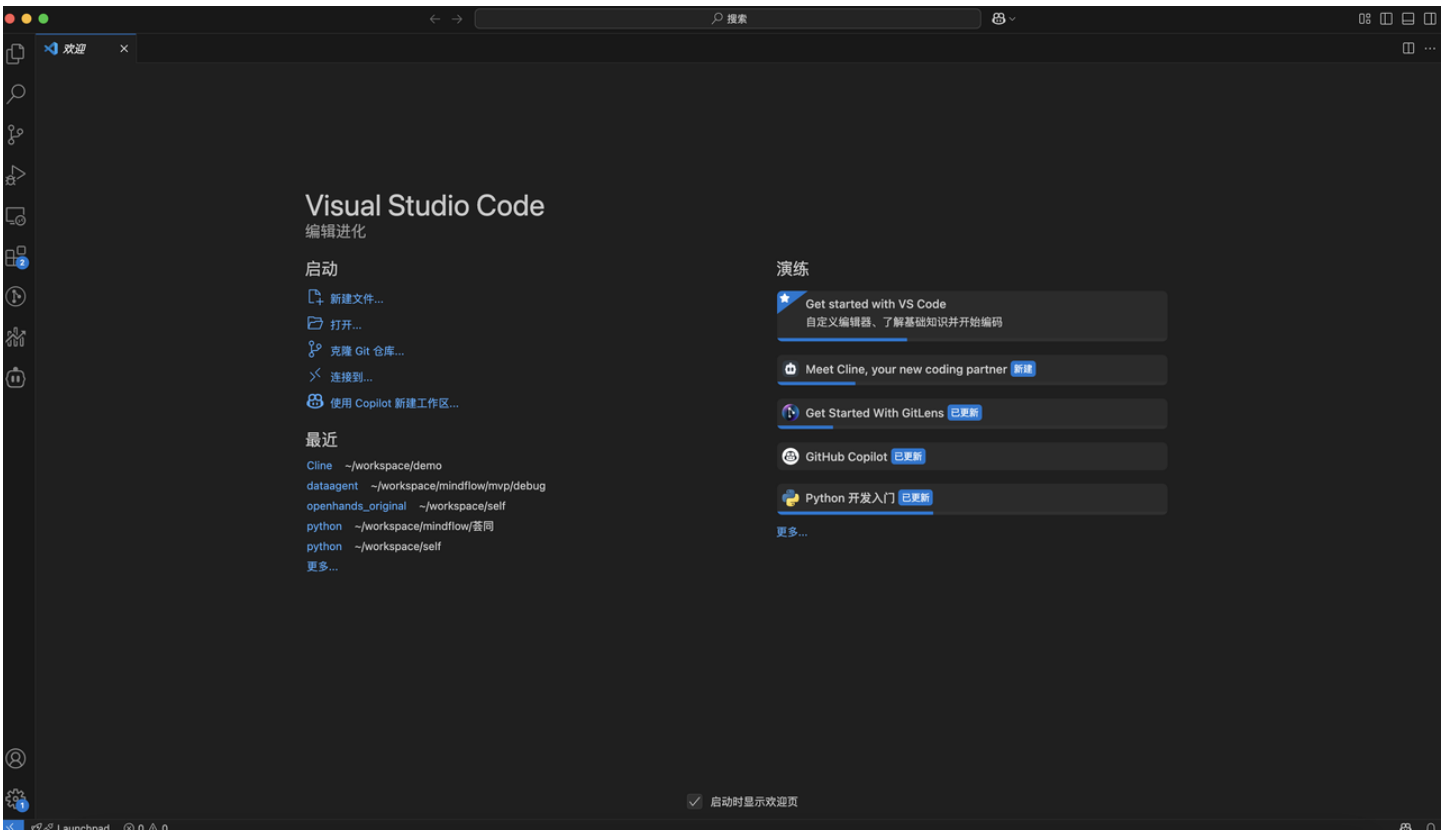
```
Last login: Tue Aug 26 19:00:07 on ttys089

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) lpMacBook-Pro:~ liupeng$ docker images
REPOSITORY                    TAG                         IMAGE ID       CREATED         SIZE
dyfadet                       latest                      0727b8e81b0a   2 weeks ago     9.81GB
<none>                        <none>                      b61dcad78a3e   3 weeks ago     9.64GB
dlinear                       latest                      af668e78ef10   3 weeks ago     9.24GB
ghcr.io/all-hands-ai/runtime  local                       d51466800246   3 weeks ago     7.78GB
ghcr.io/all-hands-ai/runtime  0.48-nikolaik               95441abe3af2   7 weeks ago     6.92GB
ghcr.io/all-hands-ai/runtime  0.47-nikolaik               5f93bb44fde9   8 weeks ago     6.97GB
nfl                           latest                      9b4785fc2c3c   2 months ago    11.8GB
nfl                           local                       e46a644e99e1   2 months ago    14.1GB
text_gcn                      latest                      0382cb9aaef7   2 months ago    11.3GB
all-hands-ai/runtime          local                       a07df05a5f57   2 months ago    5.92GB
<none>                        <none>                      78259cb172e7   2 months ago    10GB
ghcr.io/all-hands-ai/runtime  0.40-nikolaik               29fdf203d5fb   2 months ago    7.58GB
dlinear                       openhands                   26277bc3829f   3 months ago    6.3GB
dlinear                       dlinear                     cb3dd9979909   3 months ago    7.65GB
epf                           latest                      fccdfab0f899   3 months ago    10.1GB
ghcr.io/all-hands-ai/runtime  0.38-nikolaik               ccc20dc71b45   3 months ago    6.21GB
all-hands-ai/runtime          dlinear                     4dee1e8da741   4 months ago    7.99GB
python                        3.9-slim                    9a041530811d   4 months ago    126MB
sonatype/nexus3               latest                      bc491444c0dd   4 months ago    640MB
node                          18-alpine                   ee77c6cd7c18   5 months ago    127MB
node                          18-slim                     fc665eaf5031   5 months ago    192MB
briefercloud/briefer-api      latest                      16bdb4fad9ed   5 months ago    3.65GB
moby/buildkit                 buildx-stable-1             37895bc5b3fd   5 months ago    210MB
dataagent/web                 latest                      8f4f912eee66   5 months ago    173MB
code_adaptation_agent         latest                      cca69e37a10e   6 months ago    6.82GB
mindflow-jupyter_server       latest                      1d06191da663   6 months ago    2.98GB
mindflow-db_migration         latest                      6da6ef937d88   6 months ago    777MB
mindflow-ecommerce            latest                      16bdbe23d258   6 months ago    451MB
postgres                      latest                      1ec2b14946bd   6 months ago    459MB
nikolaik/python-nodejs        python3.12-nodejs22         05048535369f   6 months ago    1.38GB
nginx                         <none>                      2c9168b3c9a8   6 months ago    197MB
python                        3.12-slim                   e868845c8f03   6 months ago    150MB
hello-world                   latest                      f1f77a0f96b7   7 months ago    5.2kB
mysql                         8.0                         5ea077402e99   7 months ago    760MB
tjbtech1/paperagent           latest                      4eac24296abc   7 months ago    41.4GB
python                        3.10-slim                   48ceb3b1c775   8 months ago    153MB
pytorch/pytorch               2.4.0-cuda12.4-cudnn9-runtime   e2a75ae5a502   13 months ago   7.85GB
```

可以在docker desktop中查看

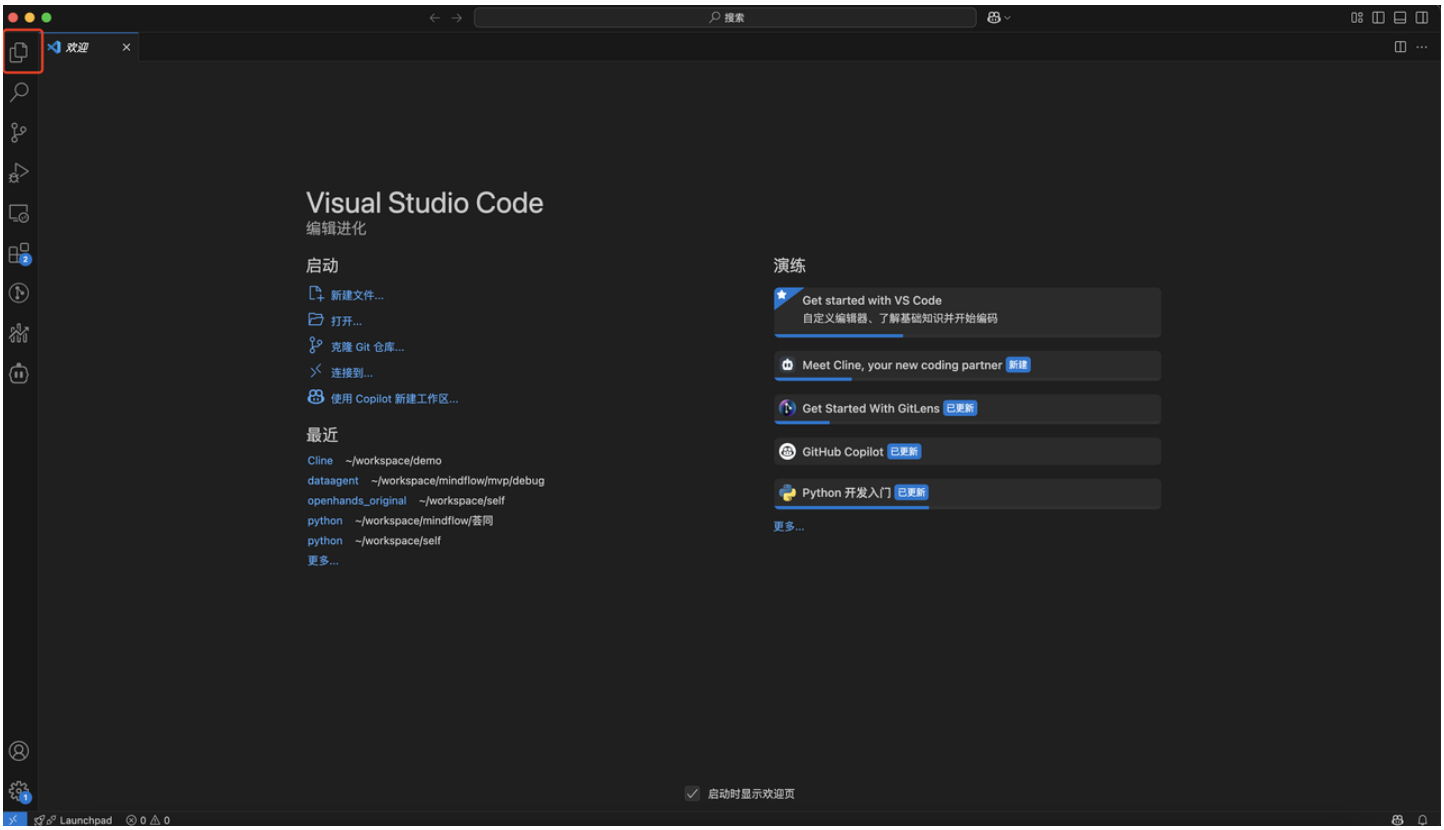


依赖安装

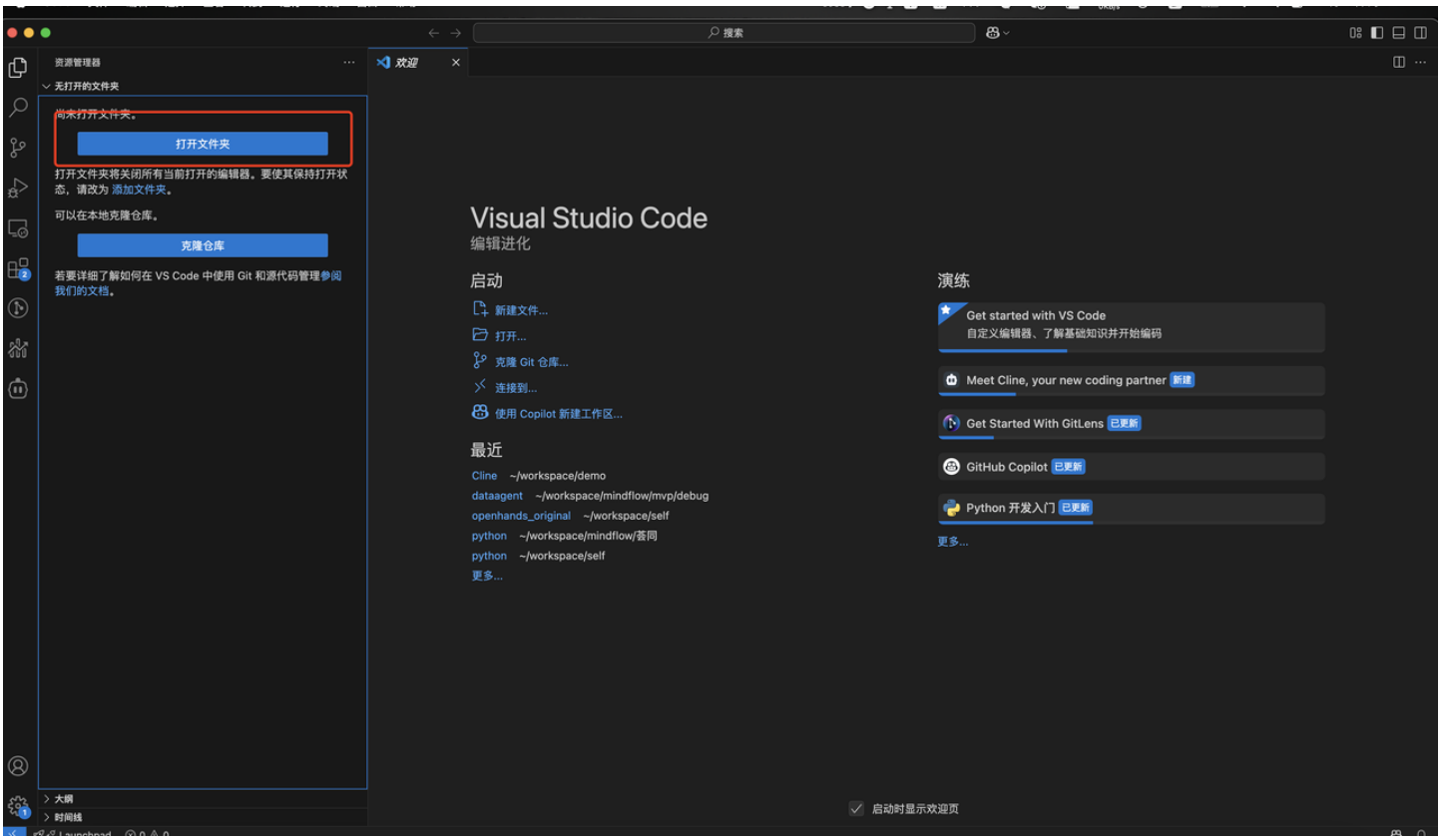
1、打开VSCode



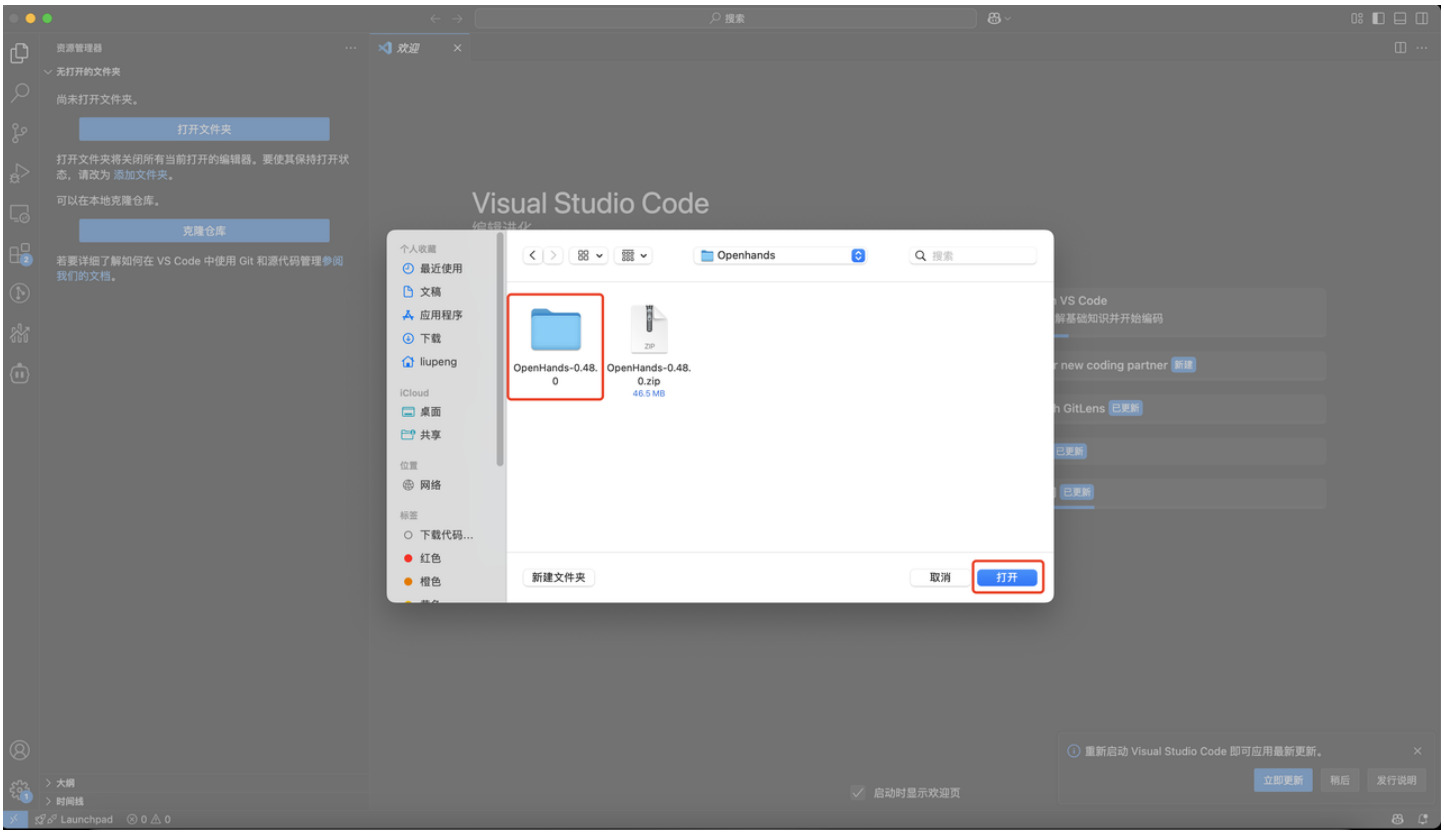
2、打开资源管理器



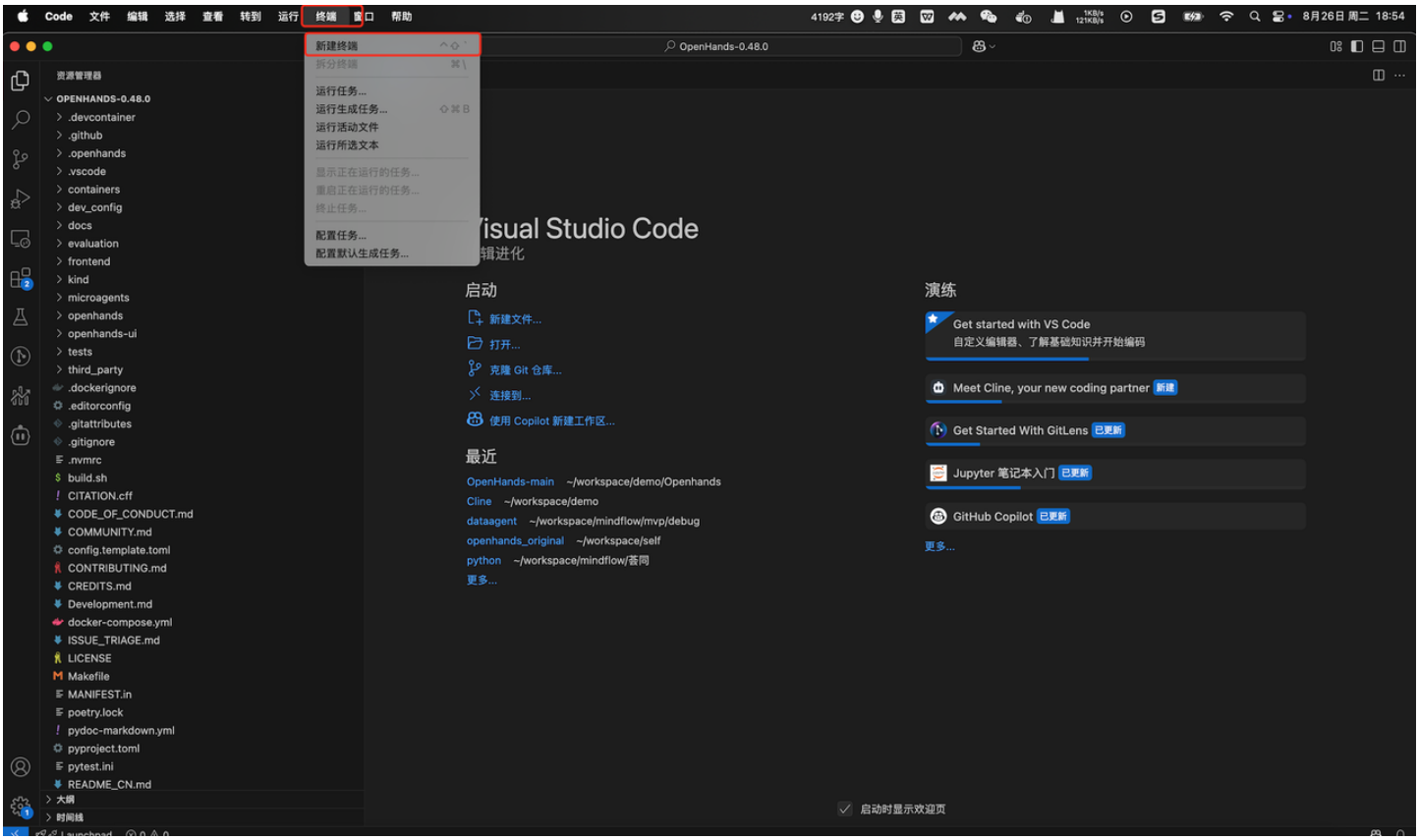
3、点击打开文件夹



4、选择解压完成的openhands代码文件夹并点击打开按钮



5、新建终端

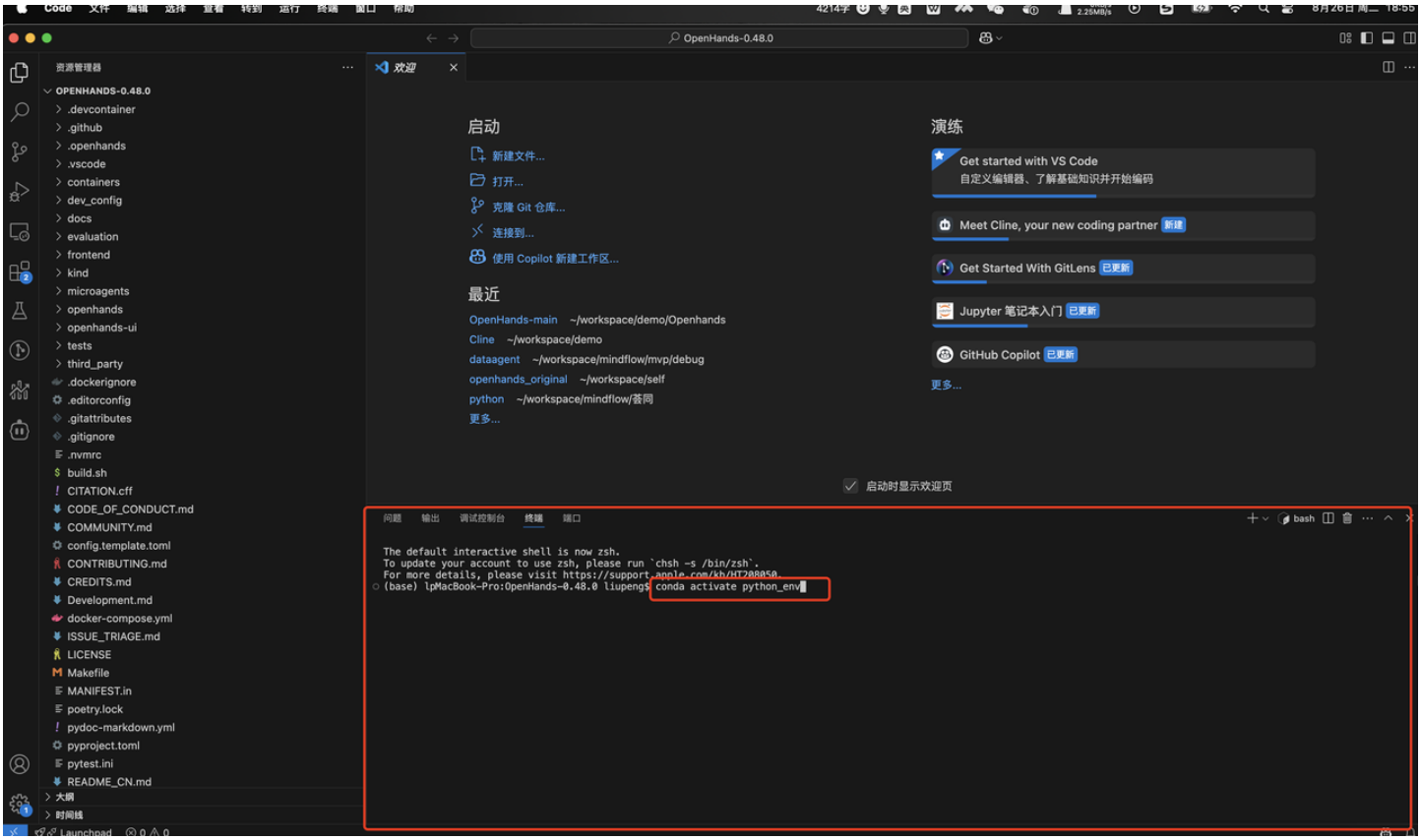


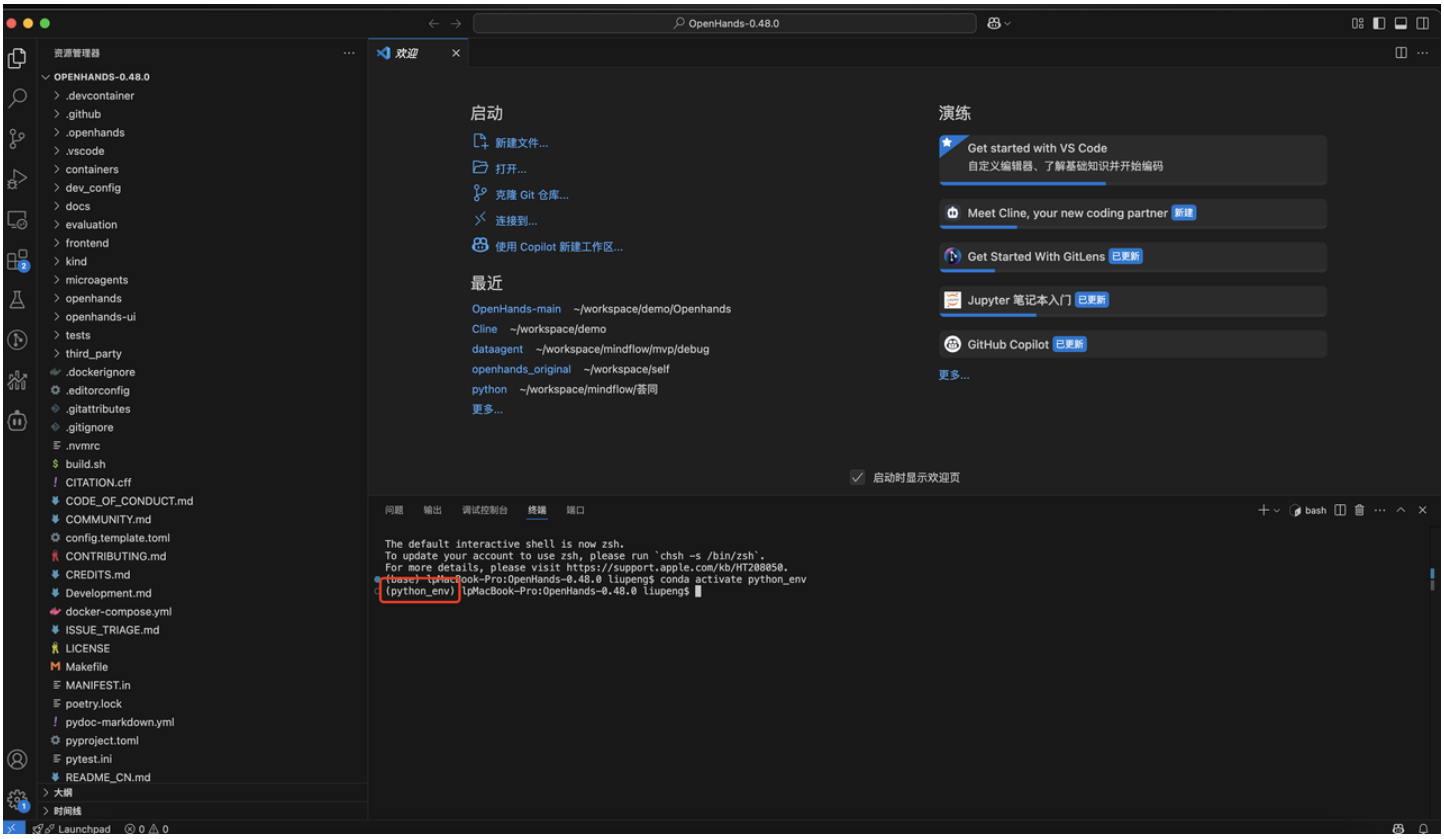
6、选择创建的自定义的python环境,若没有创建参考第二部分ython环境准备（此处的python环境必须>=3.12）中Conda下的创建自定义python环境

终端输入以下代码并执行

代码块

```
conda activate python_env
```



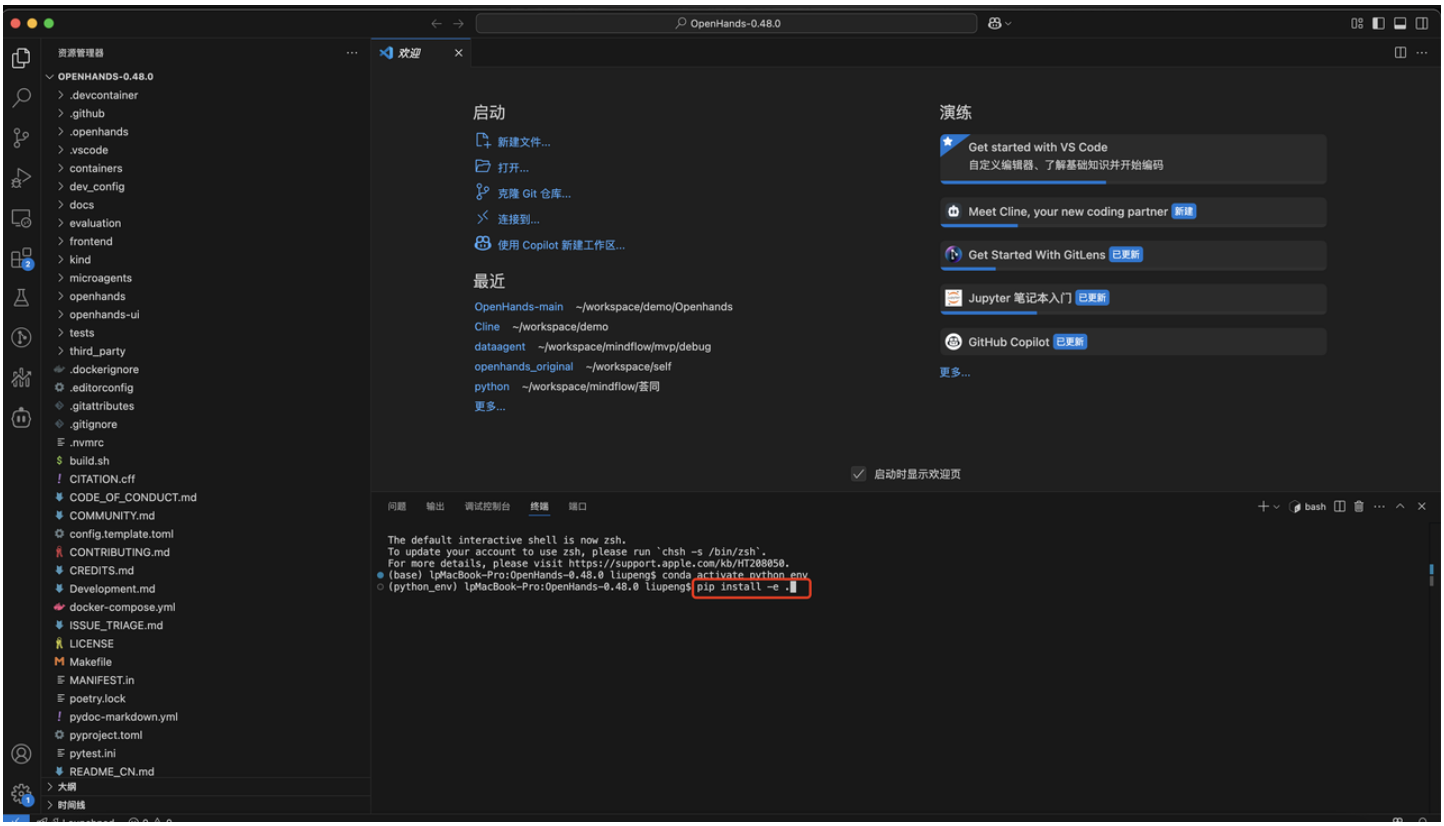


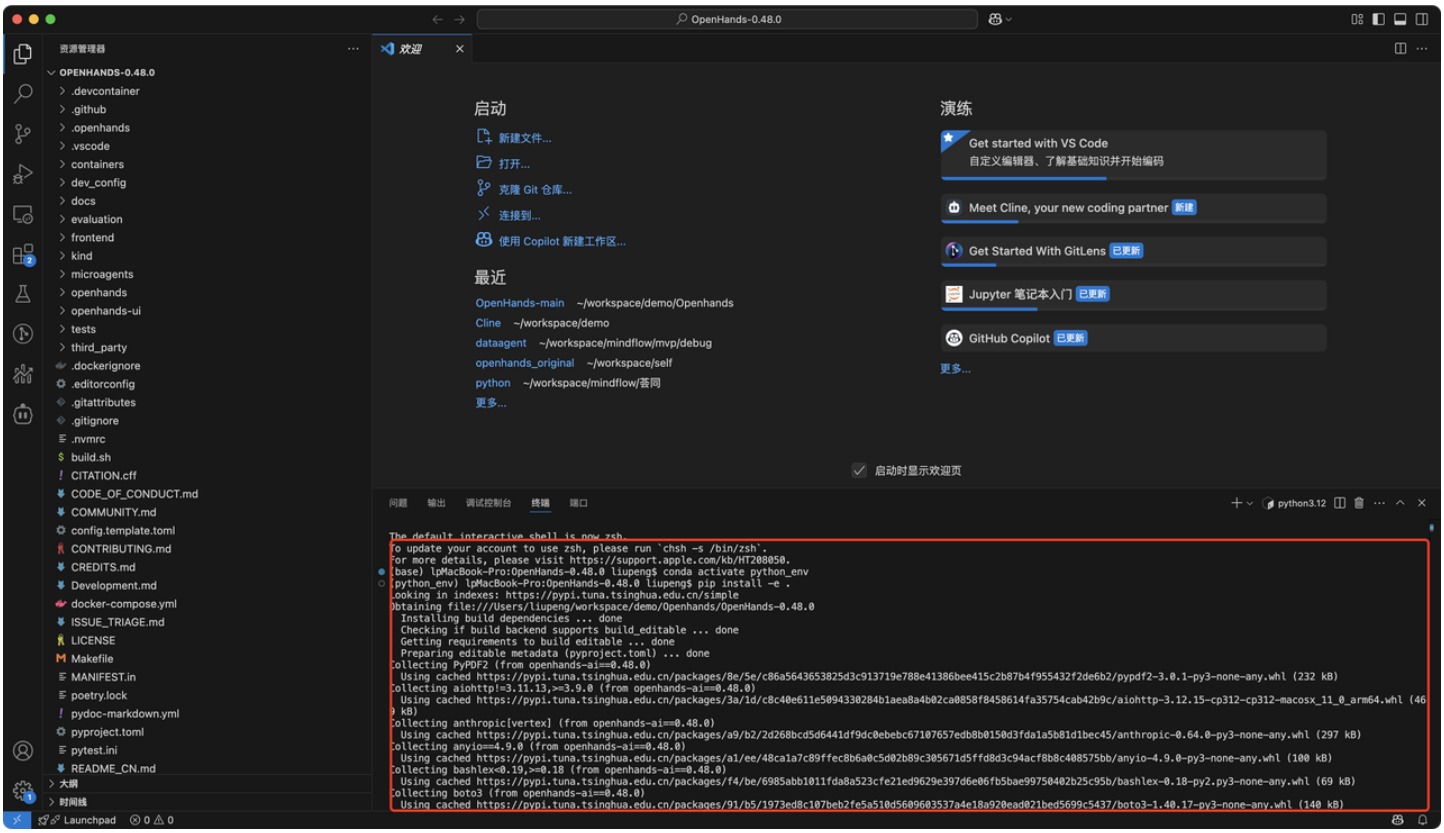
7、终端输入命令安装程序依赖

代码块

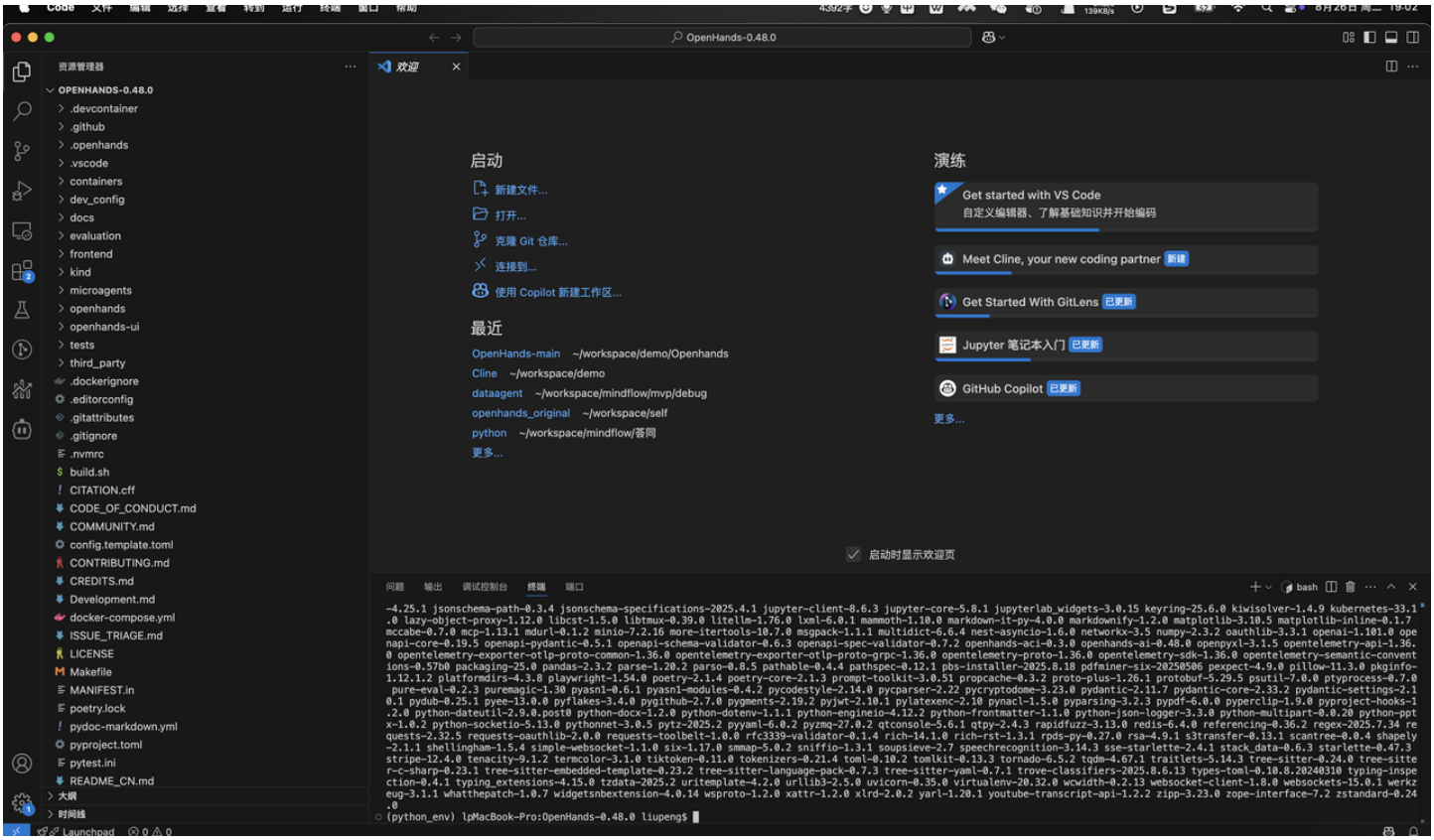
```
pip install -e .
```

```
pip install Deprecated
```





安装完成



配置文件准备

需要将以下配置文件放入到openhands项目根目录下

openhands项目配置文件

config.toml
12.79KB

启动脚本文件

run.sh

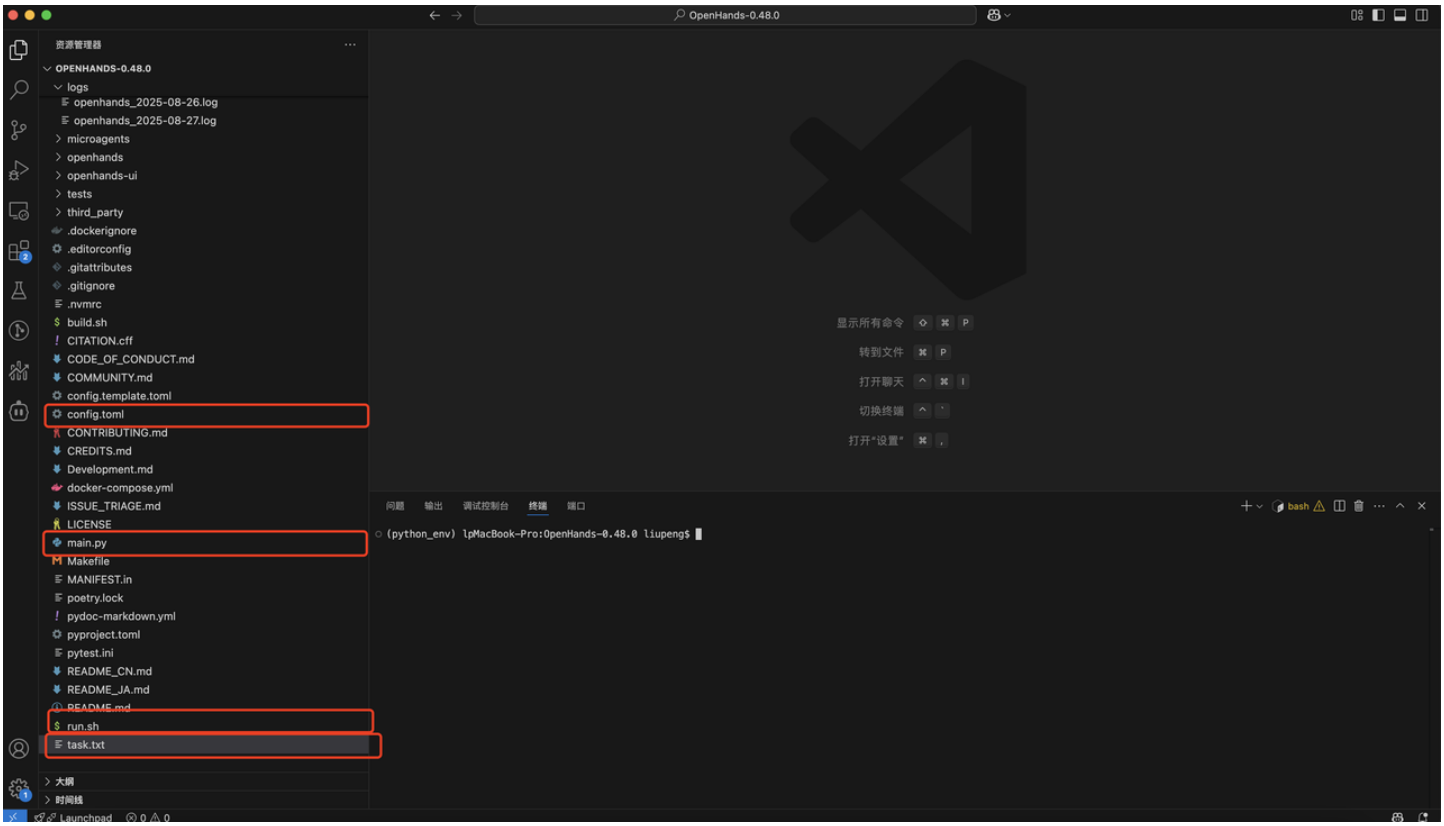
任务描述文件,这里以创建一个hello world的python文件为例

task.txt

入口函数

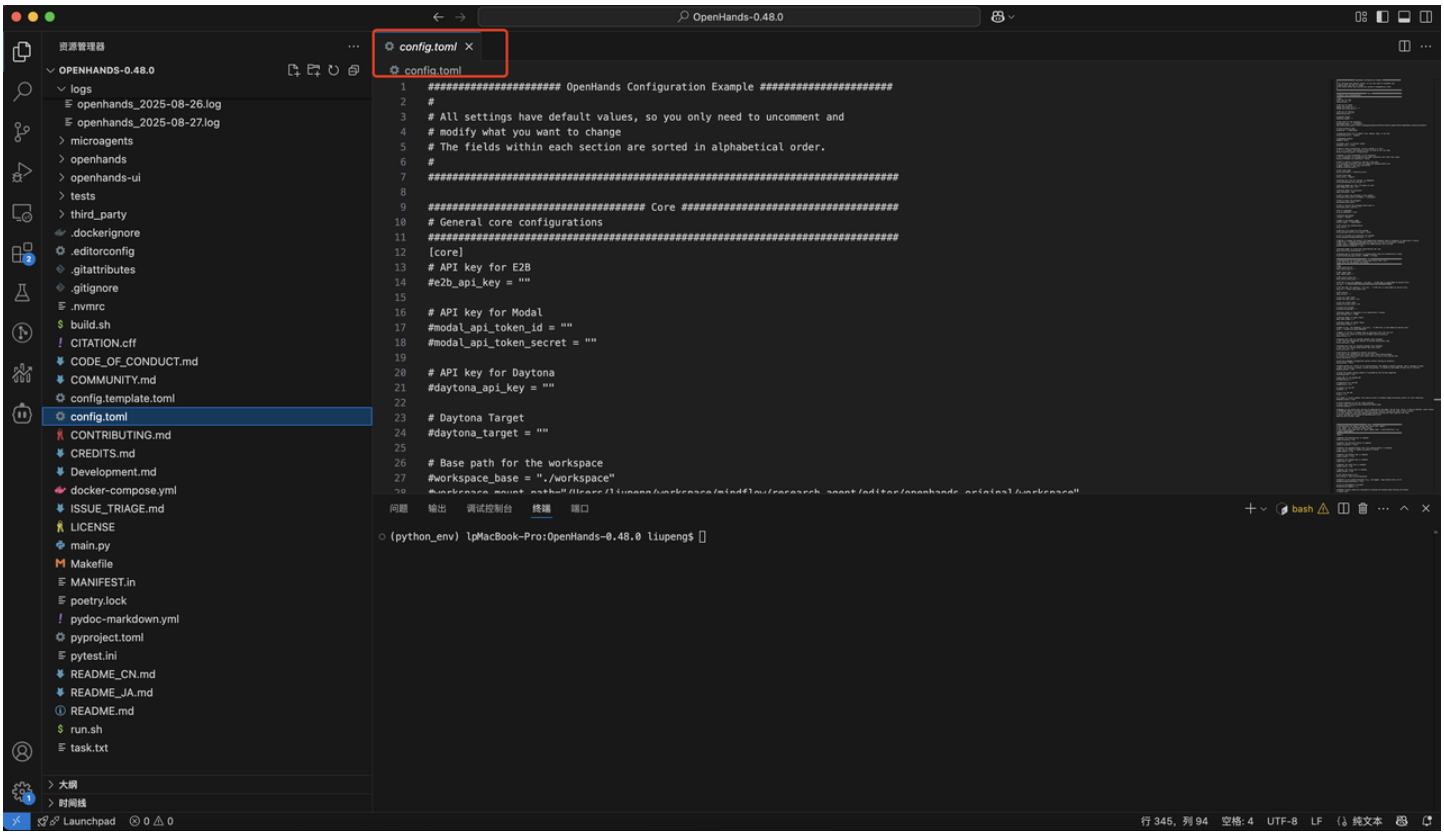
main.py

1、下载配置文件,并将配置文件复制到VSCode中Openhands项目下

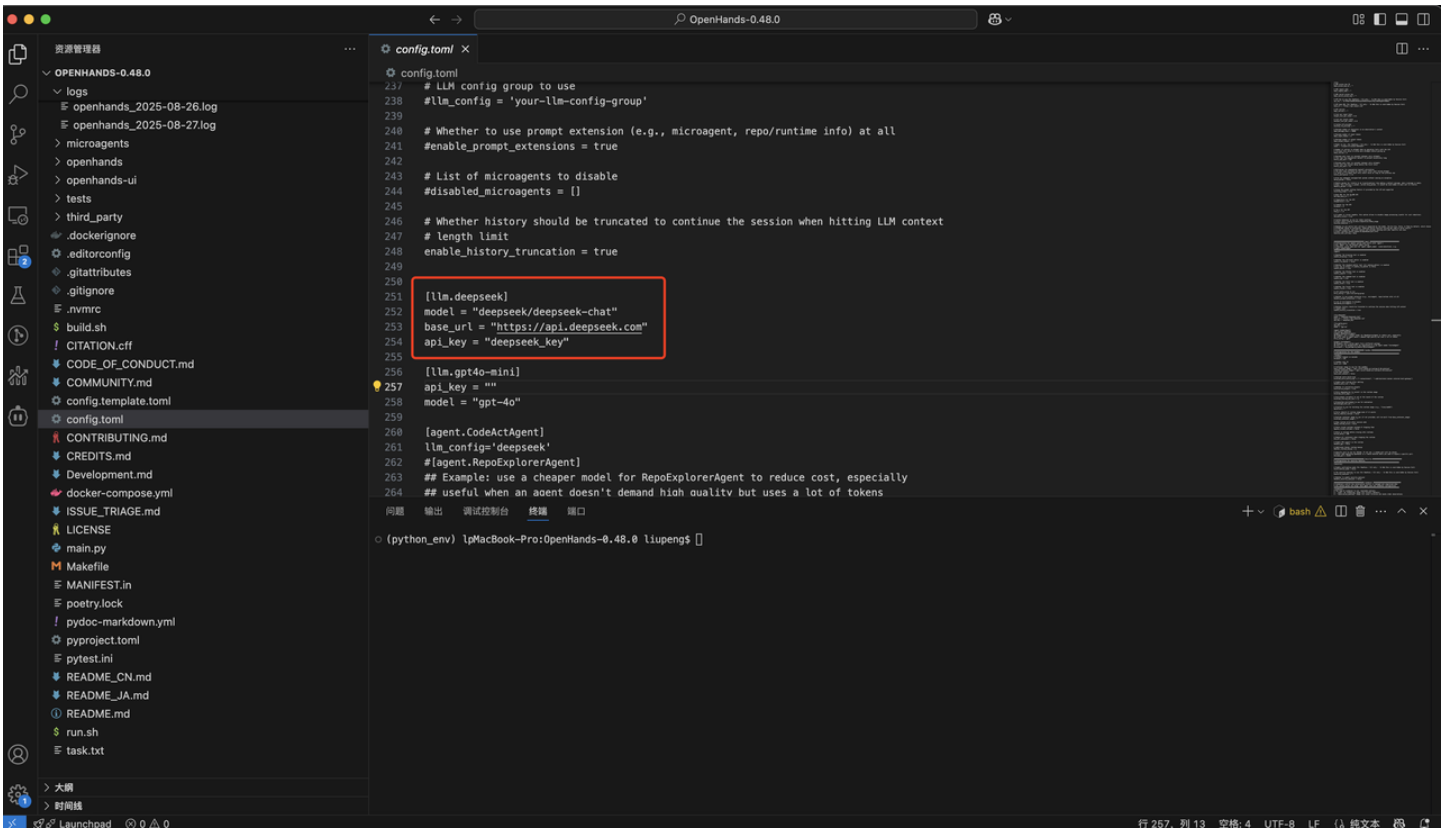


2、添加DeepSeek-API Key

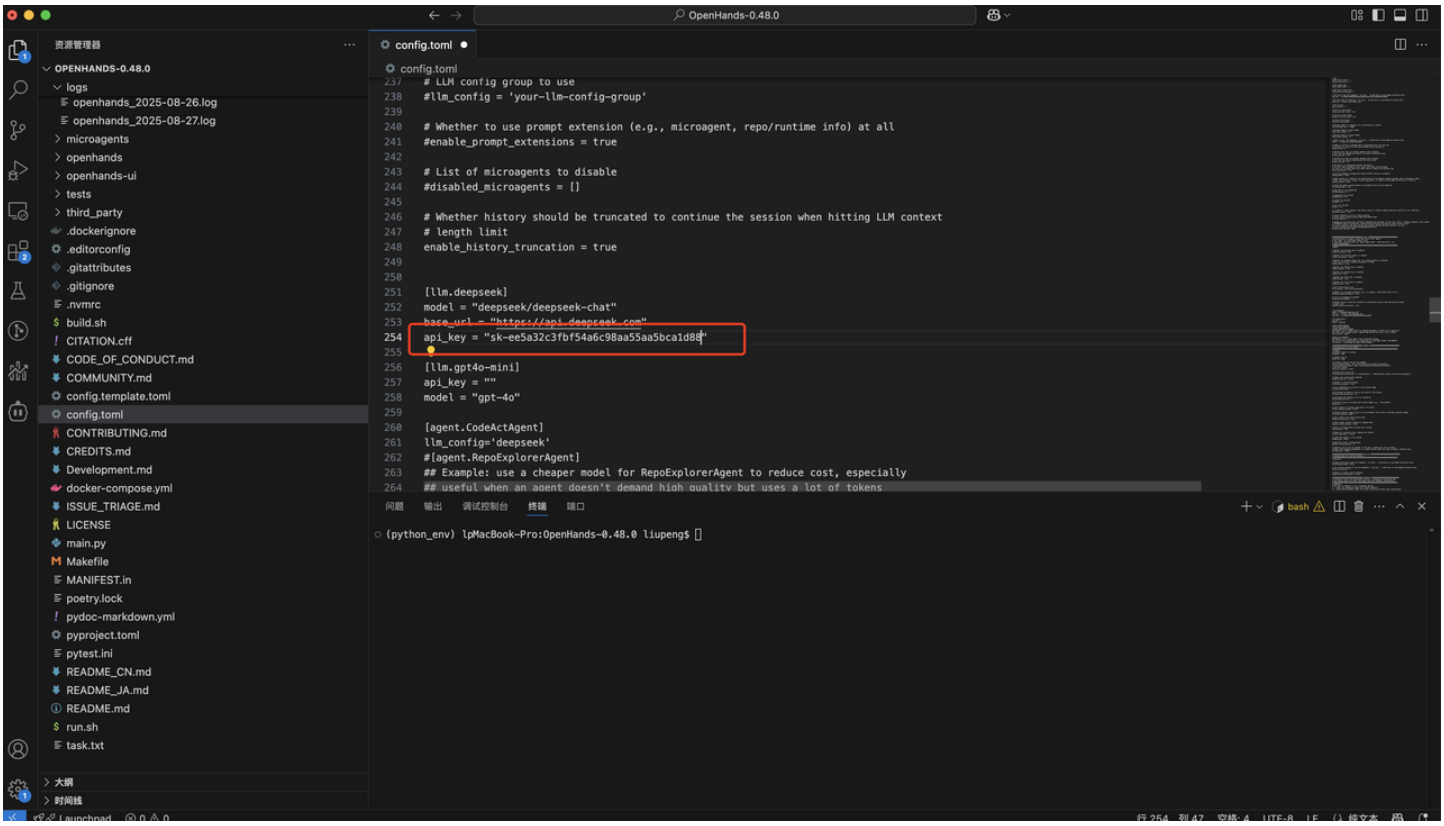
双击打开config.toml



找到[llm.deepseek]配置



替换为我们已经申请的deepseek-key

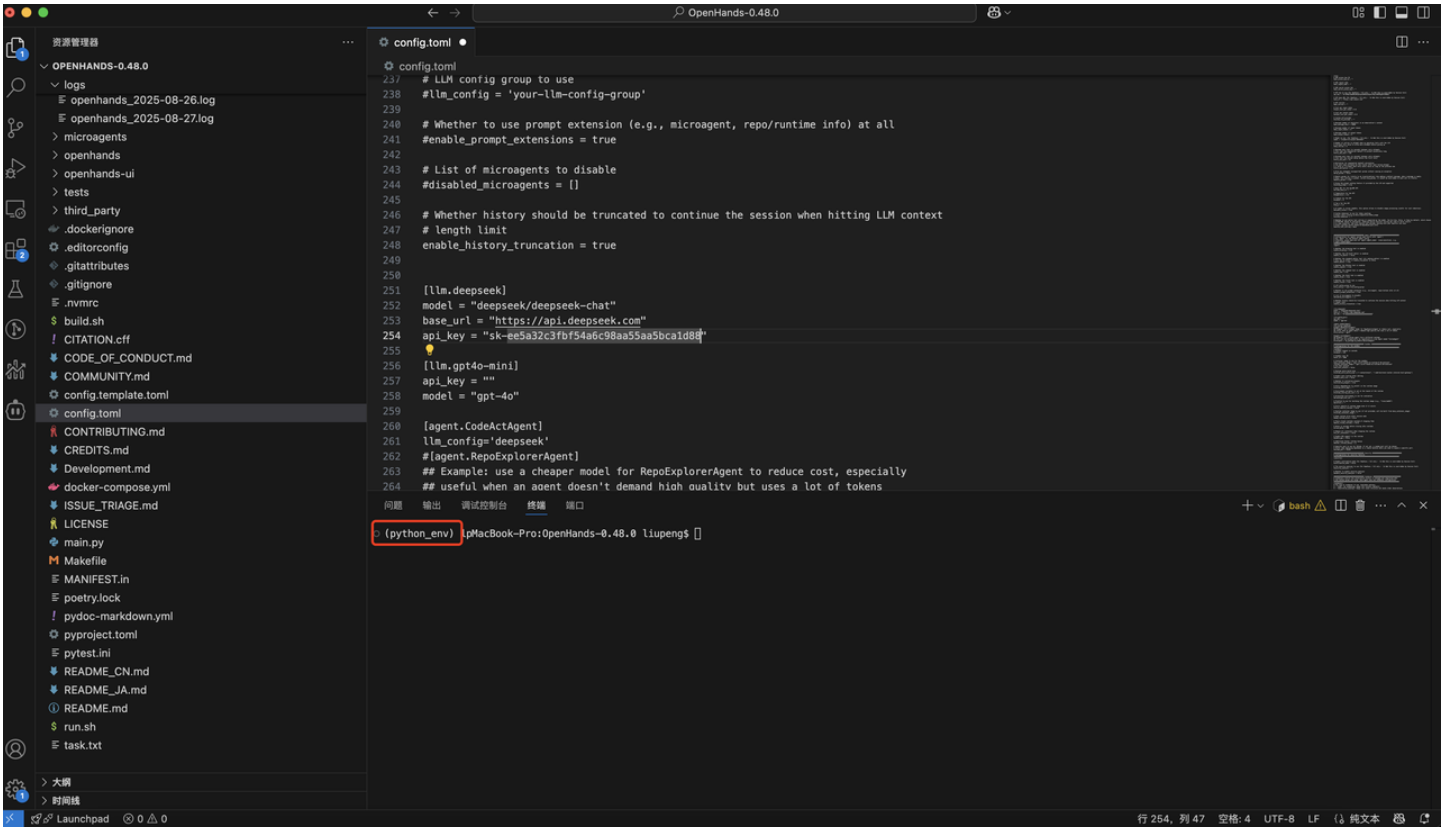


3、确认切换python环境

终端确认已经切换到openhands所应用的python环境,如果没有执行命令切换

代码块

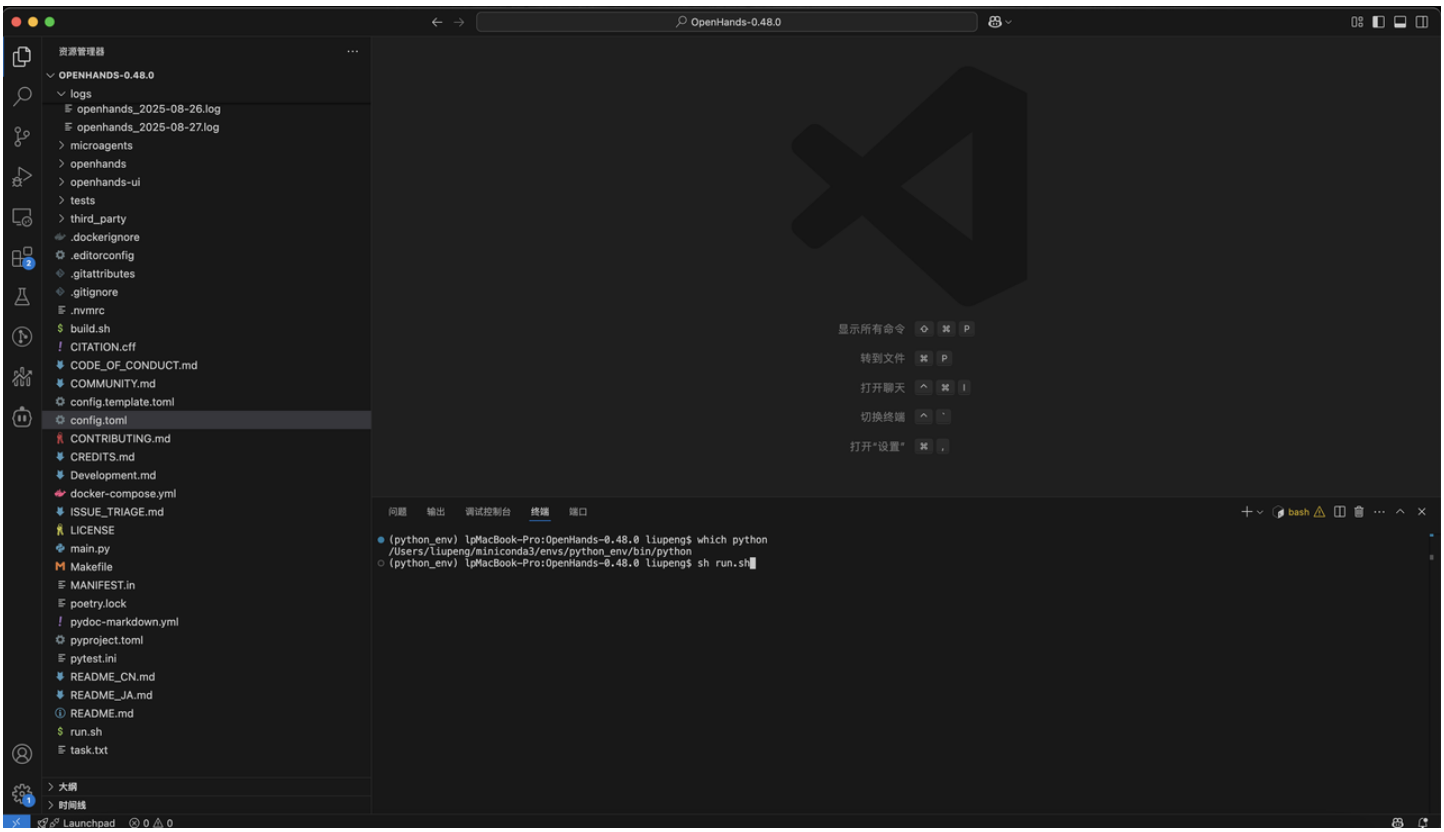
```
conda activate python_env
```



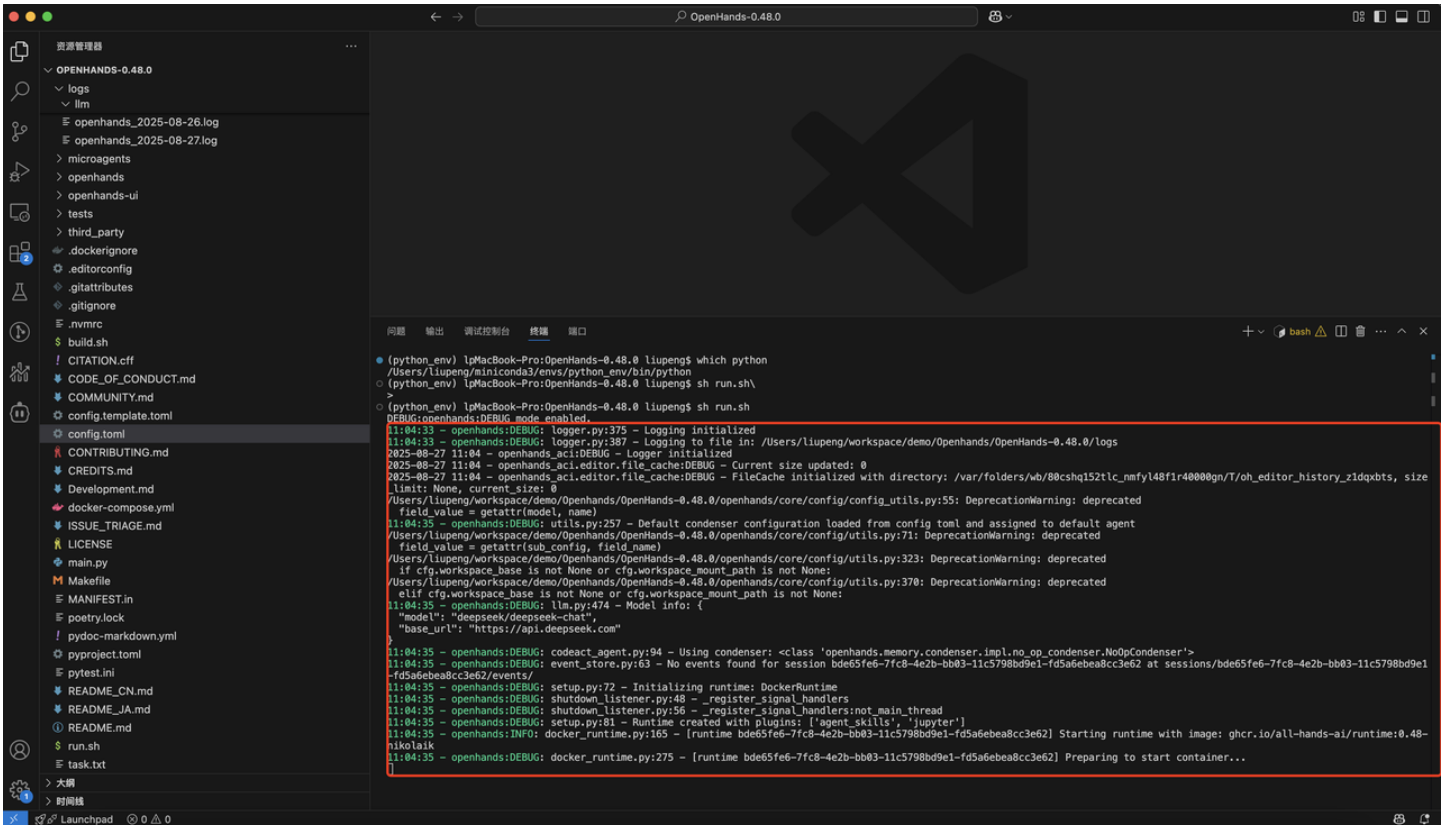
4.运行脚本文件

代码块

```
1   sh run.sh
```

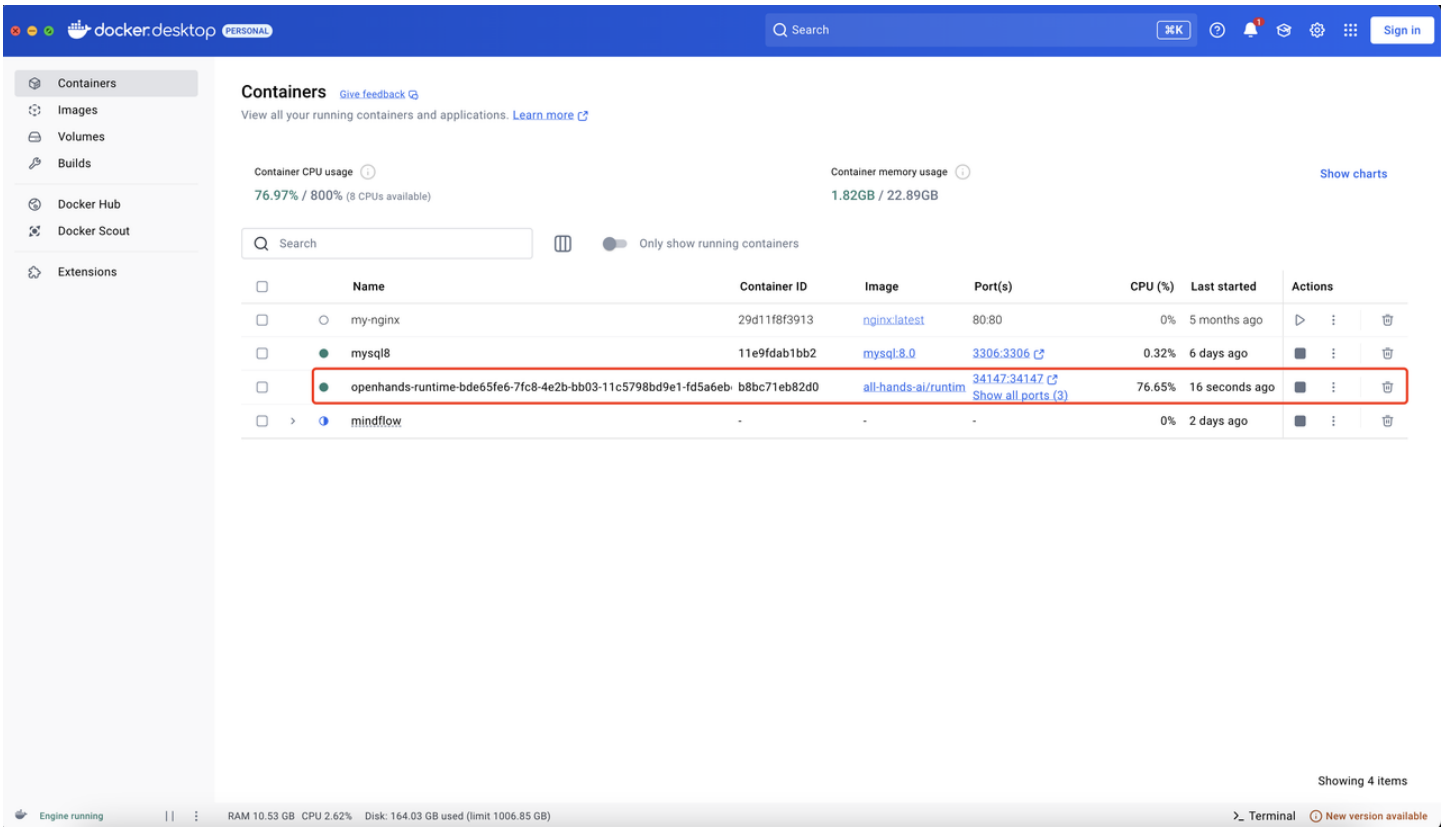


openhands运行启动容器

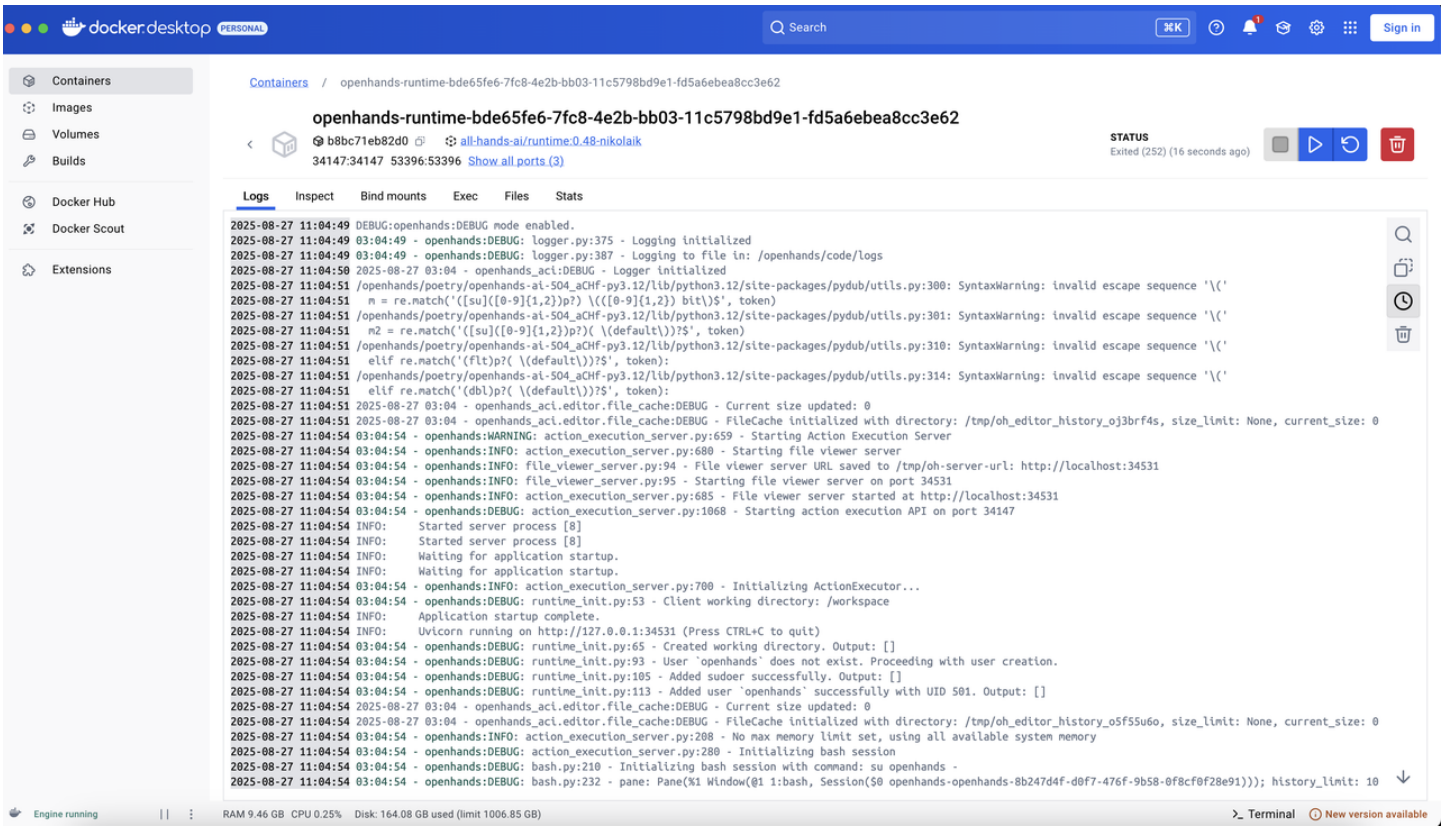


5、查看容器

打开docker desktop可以查看到已经启动了一个openhands容器,点击打开

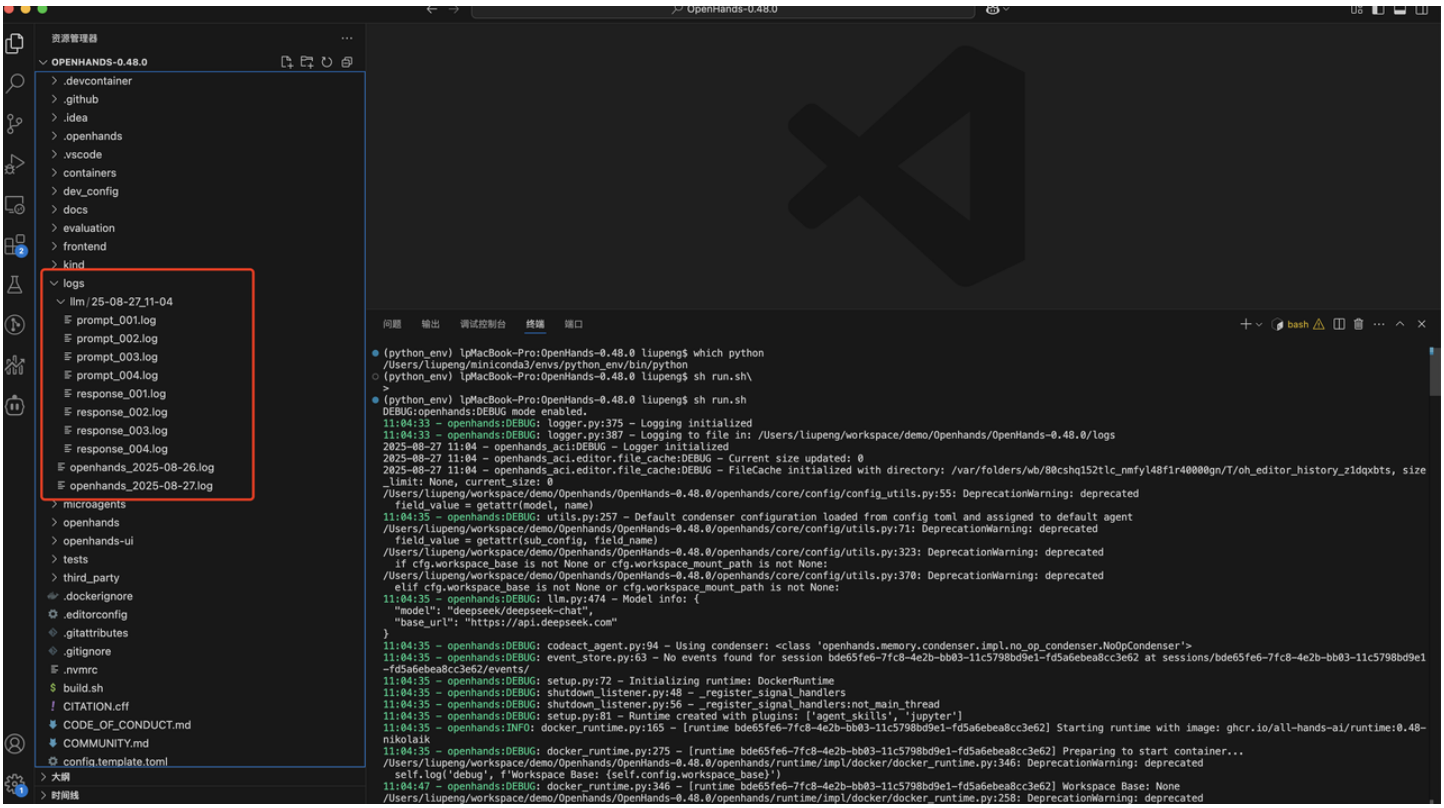


容器内已经在启动程序



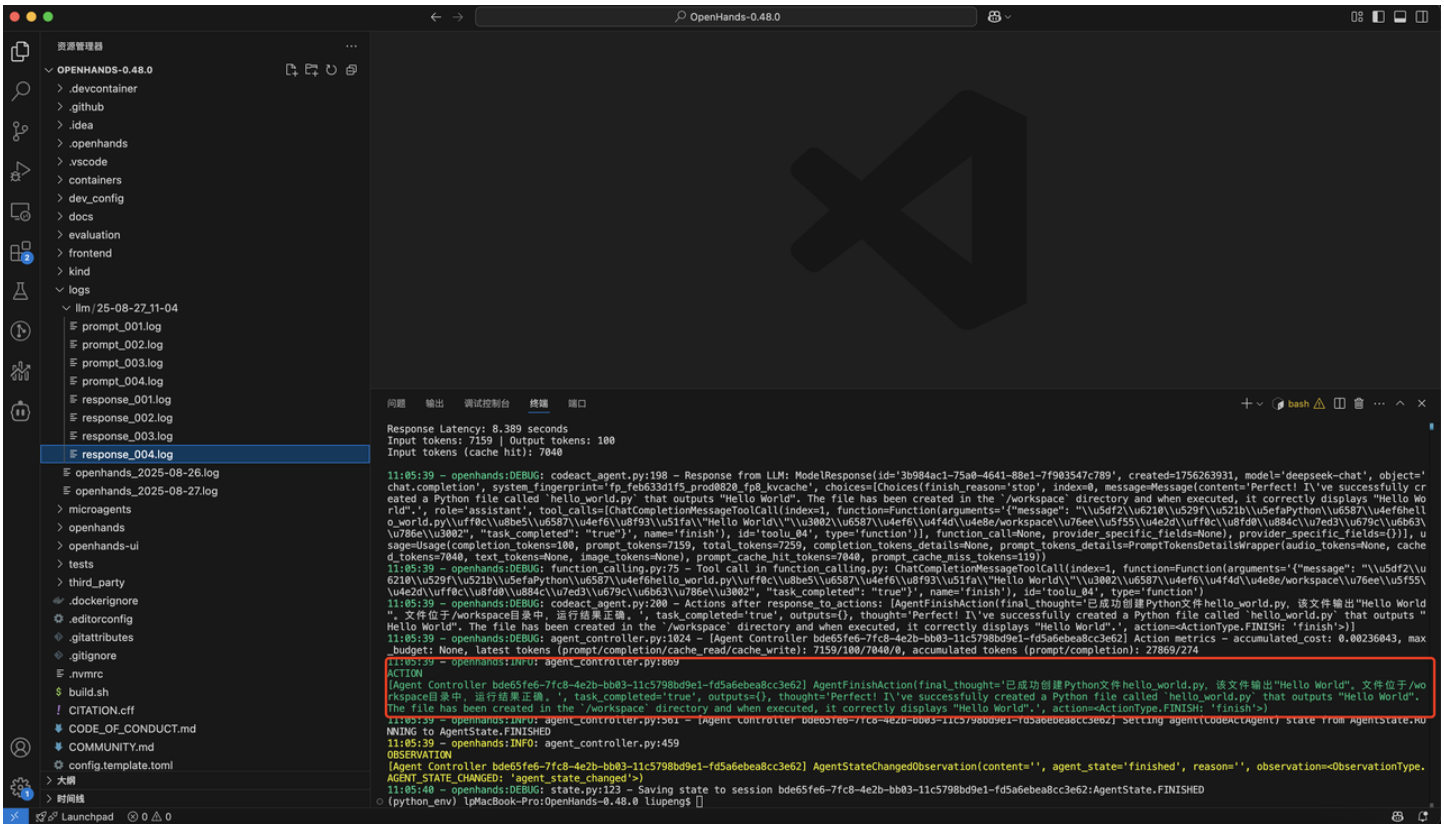
6、查看大模型对话

查看openhands目录下logs/llm文件夹,该文件夹是agent的对话信息,prompt和response是成对生成



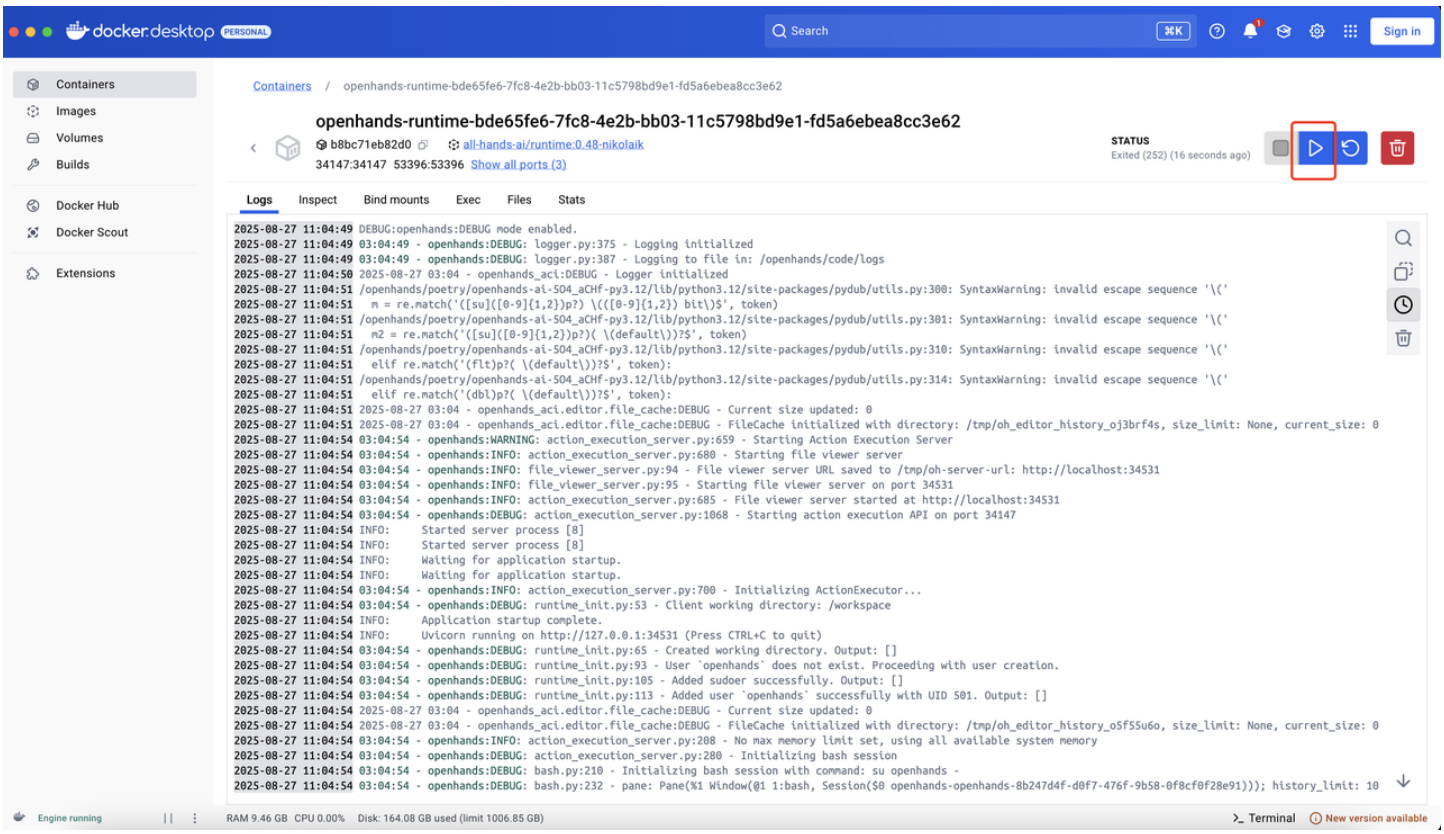
7、查看运行结果

终端输出已经完成创建python代码文件并输出hello_world

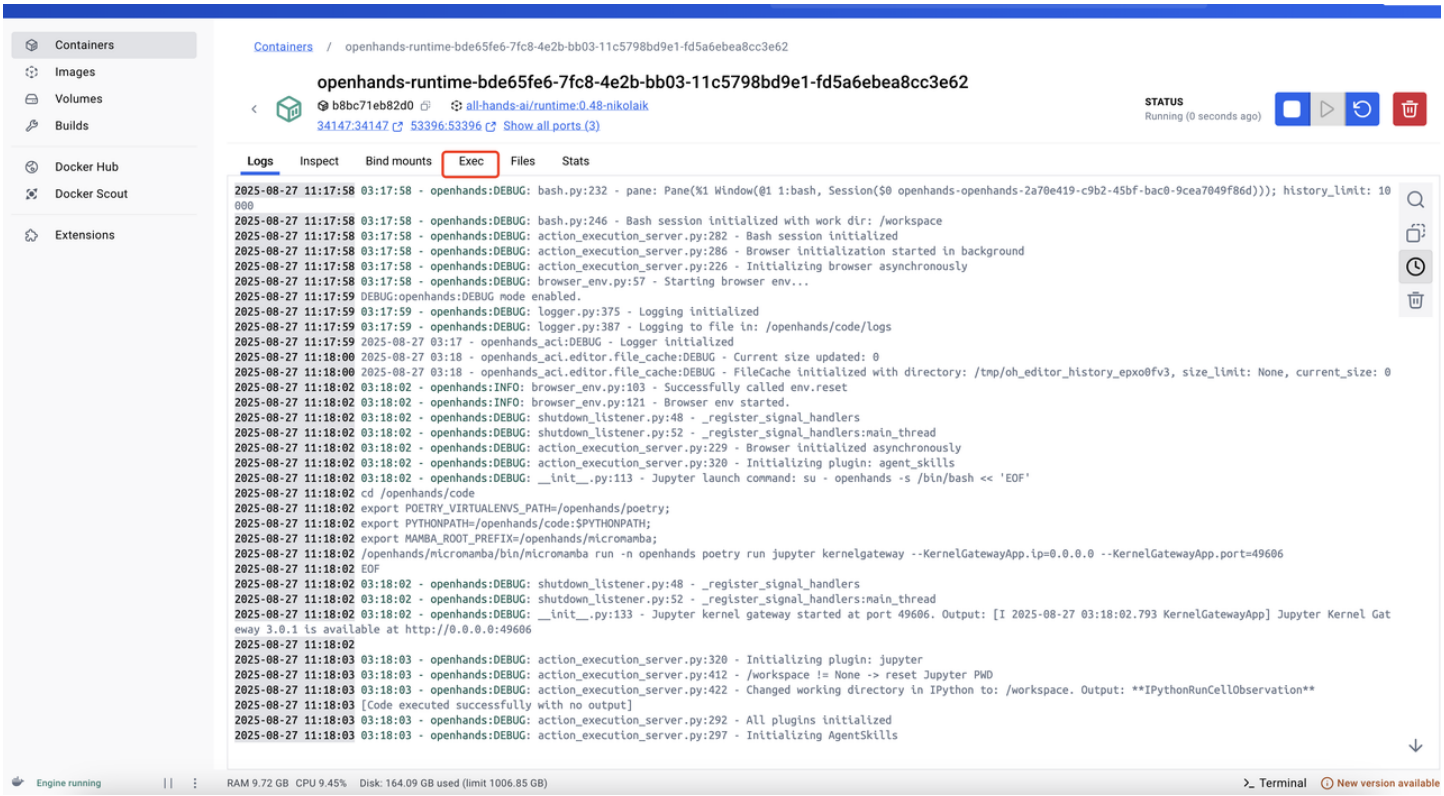


8、验证结果

查看任务容器已经结束,点击运行重新启动容器



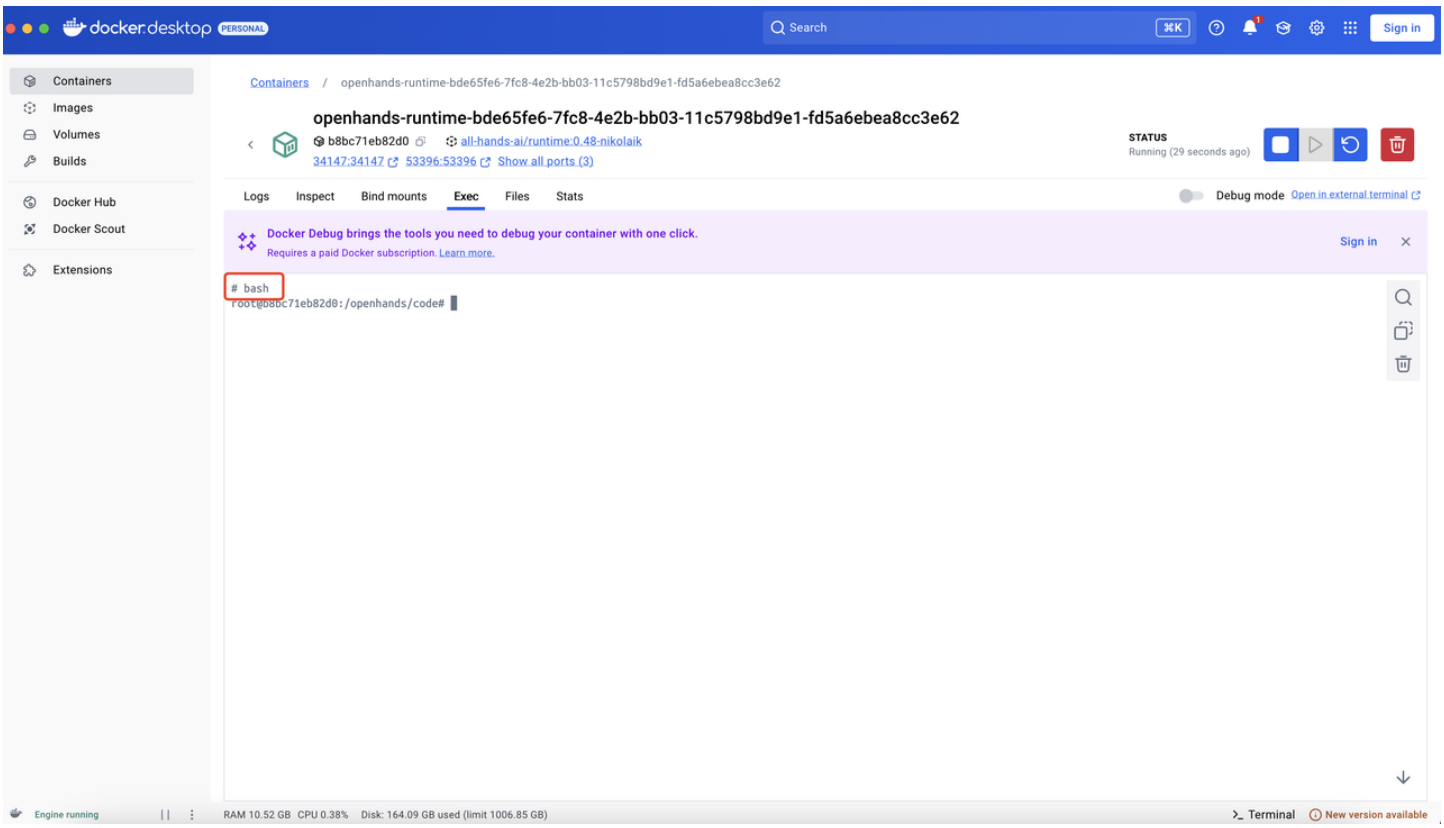
点击exec进入命令窗口



输入以下命令并执行

代码块

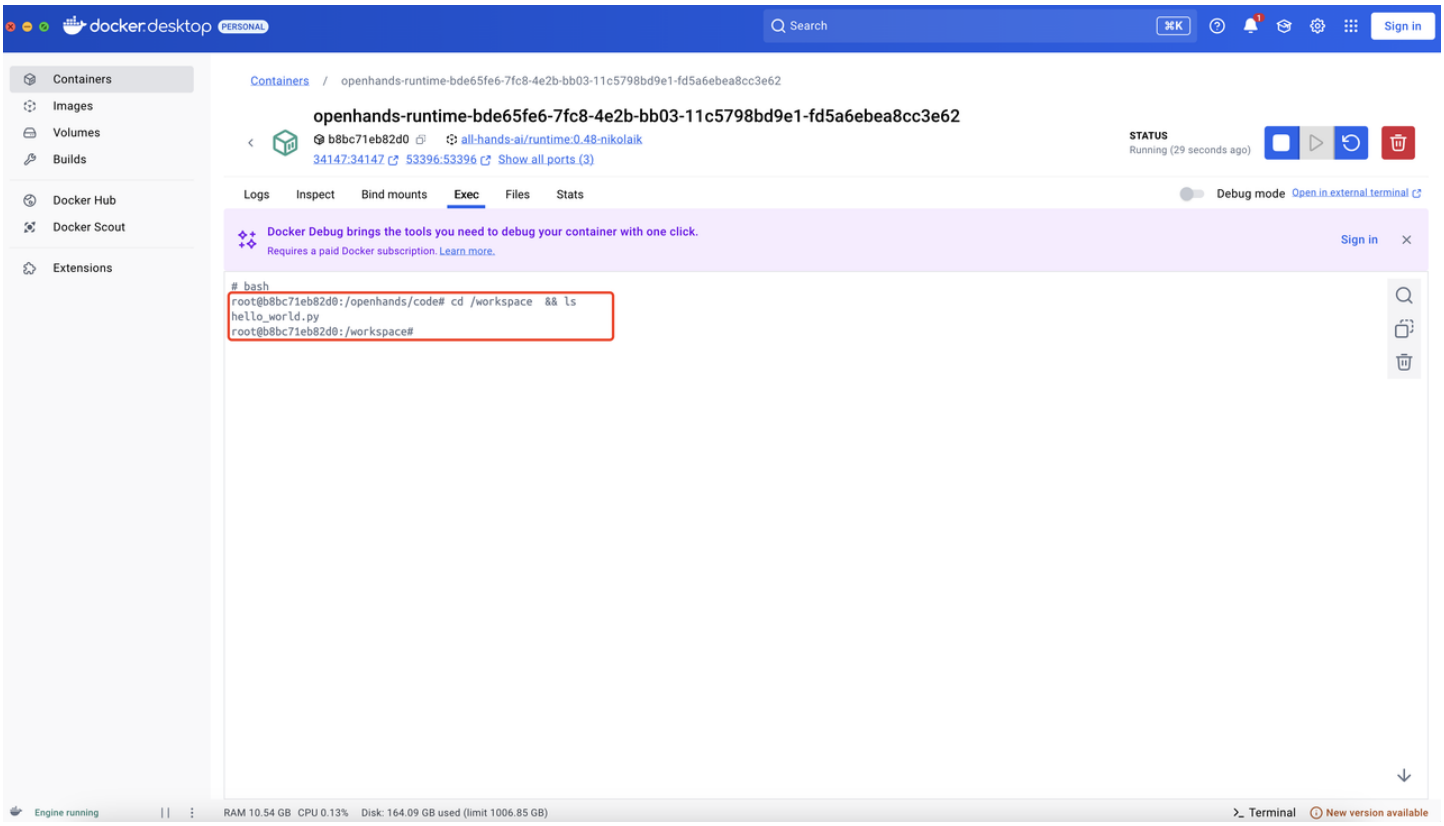
```
1    bash
```



执行命令查看是否有生成文件

代码块

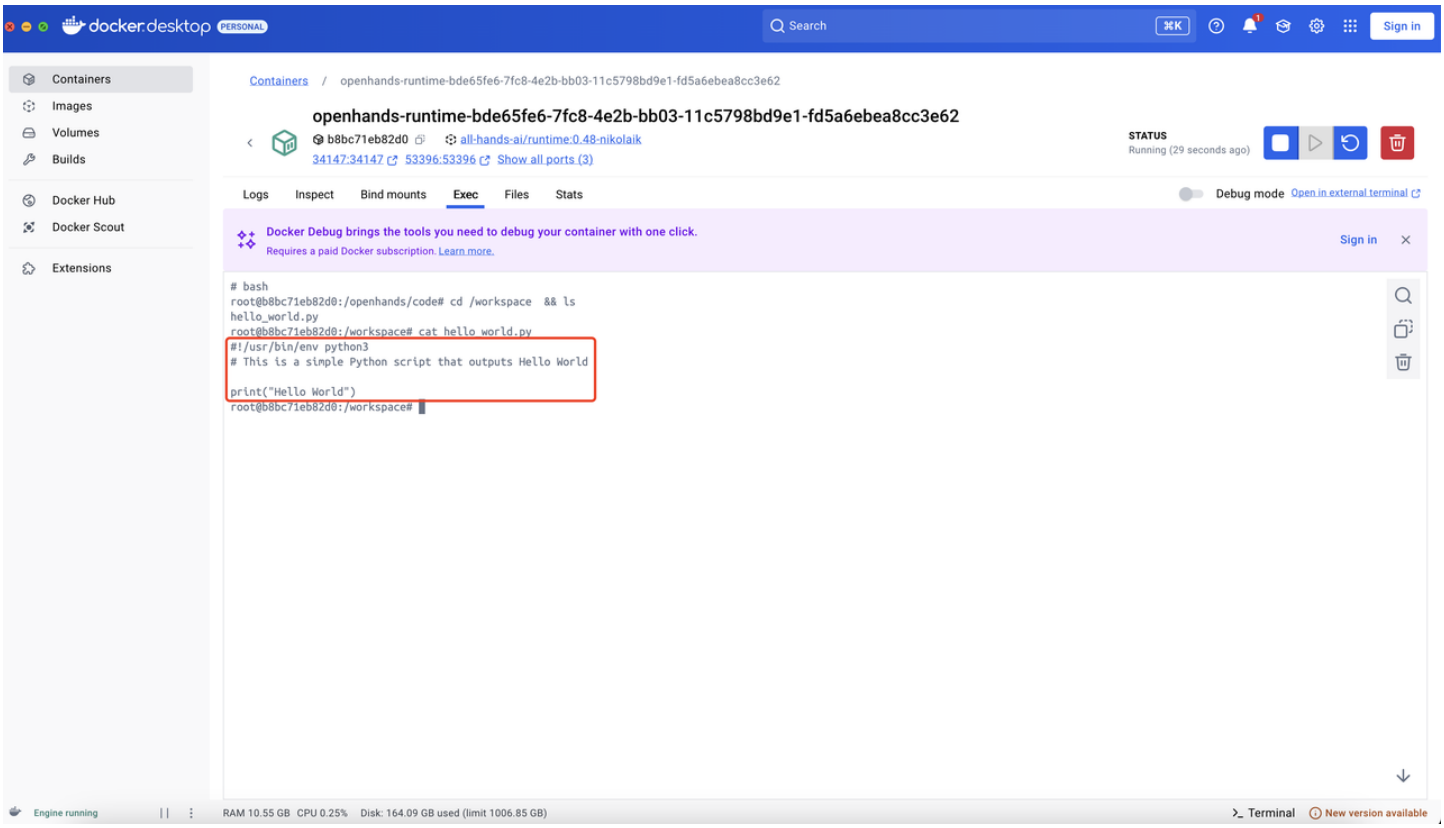
```
1  cd /workspace && ls
```



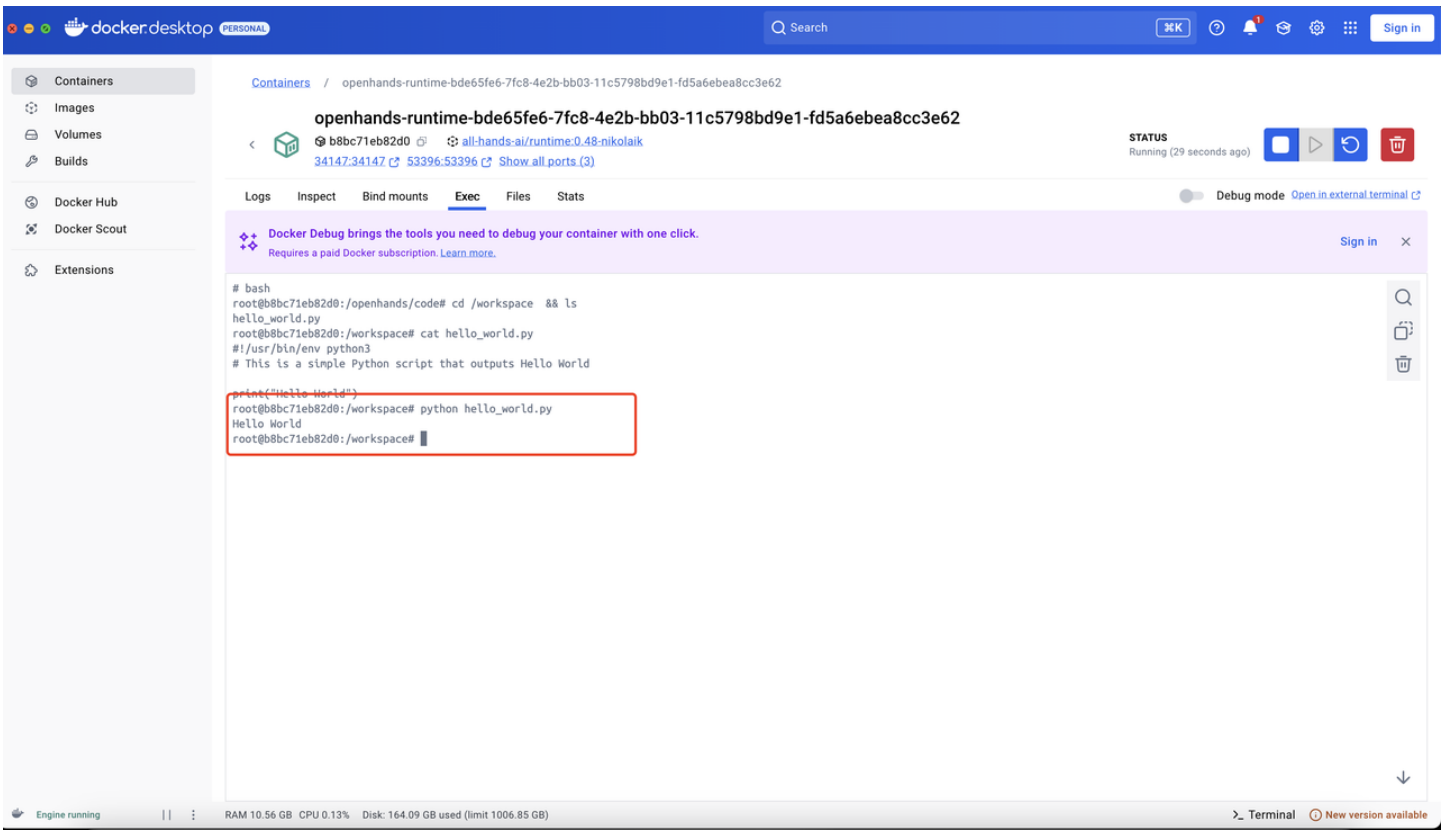
查看文件内容

代码块

```
1  cat hello_world.py
```



helloworld代码已创建,运行校验



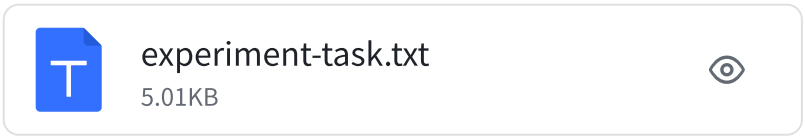
正确输出hello world 验证完成

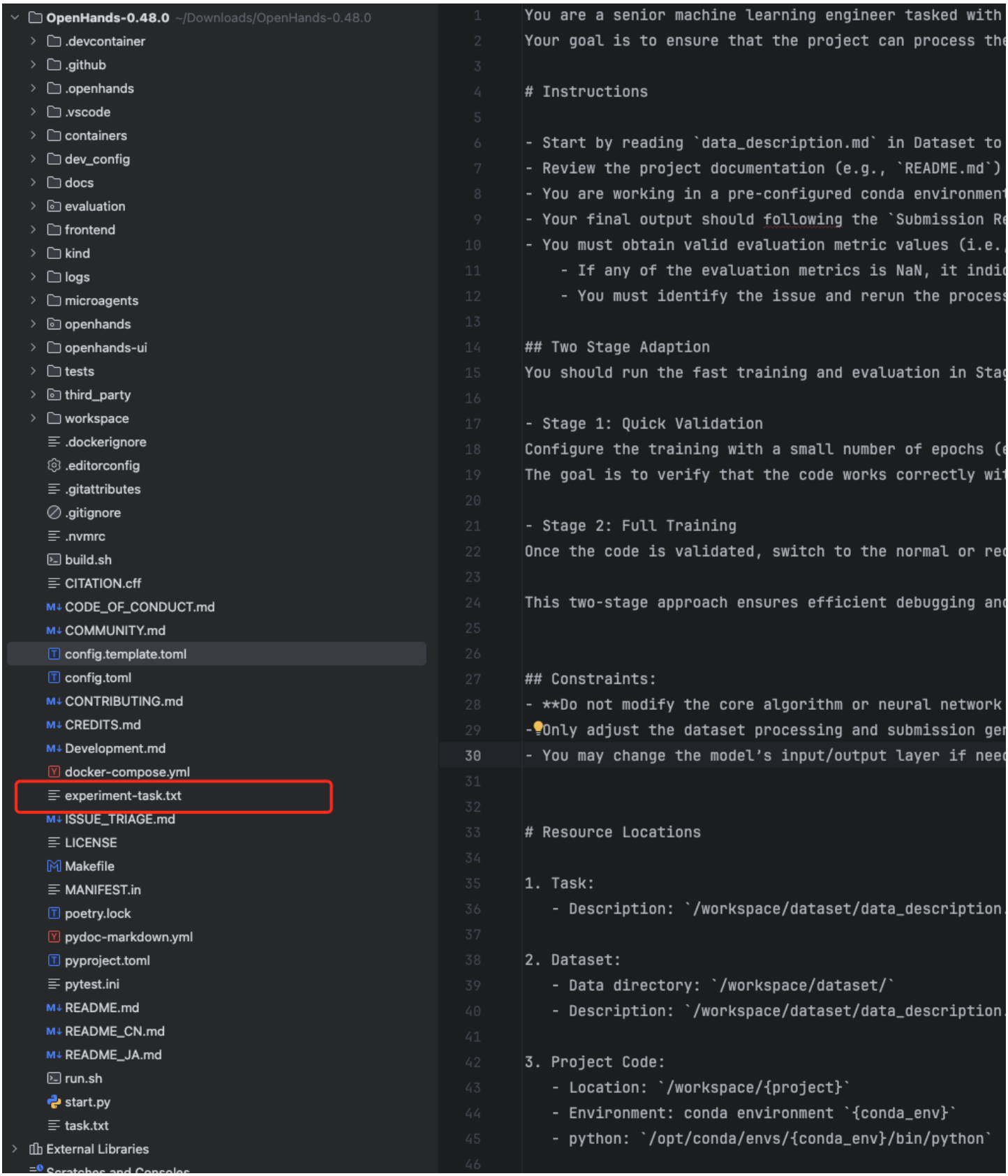
openhands配置和实验跑通

以一个RegMean++这个sota工作为例跑通实验

Sota项目地址:https://github.com/nthehai01/RegMean-plusplus

1. 添加实验的任务描述





2. openhands配置文件修改

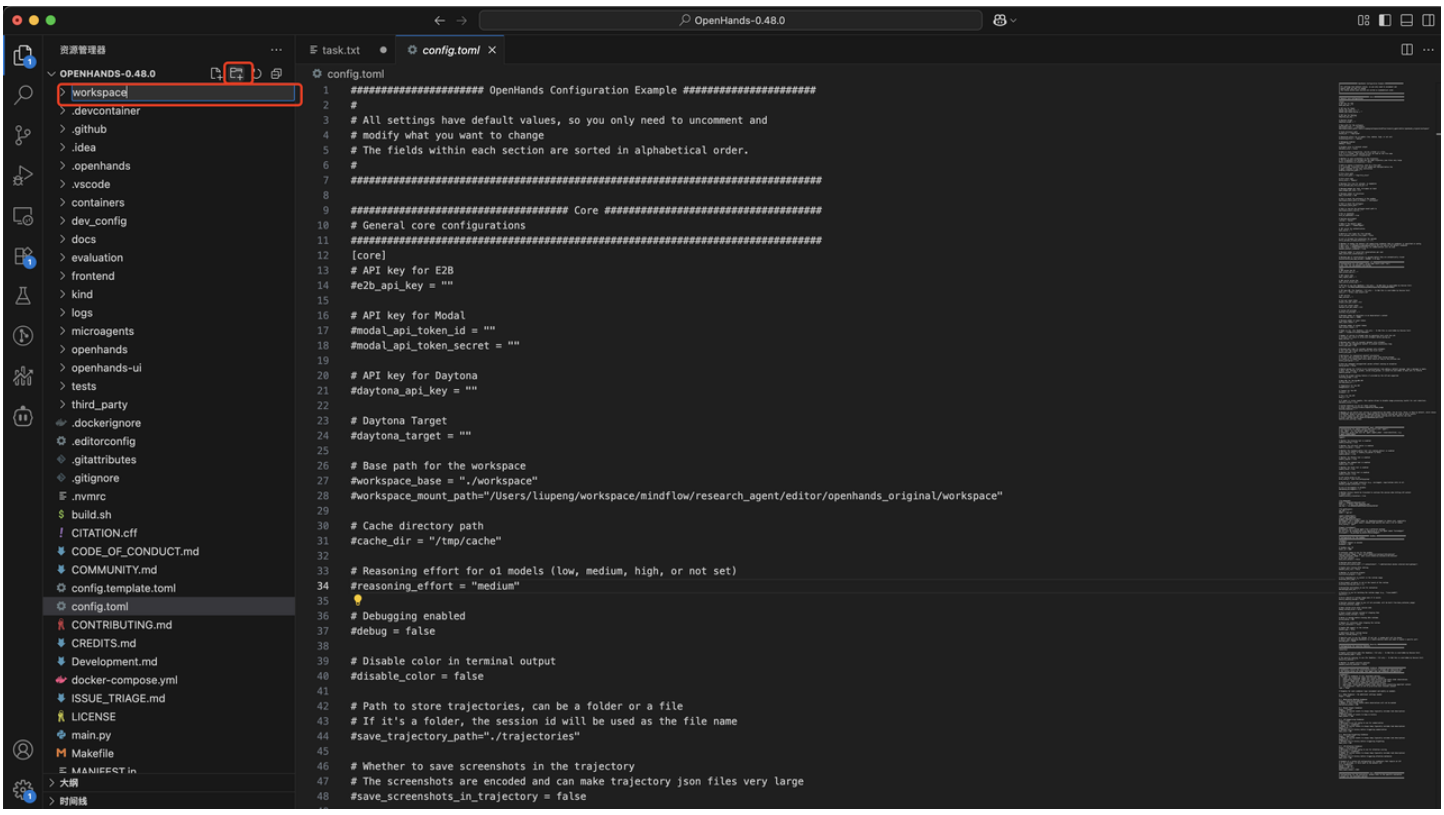
config.toml
12.79KB

需要修改config.toml配置文件

代码块

```
# Base path for the workspace
workspace_base = "./workspace"
#修改为需要挂载到容器内的workspace目录. 默认直接在openhands项目下创建一个workspace文件夹,将其挂载到容器内
workspace_mount_path={本地项目文件/workspace}
```

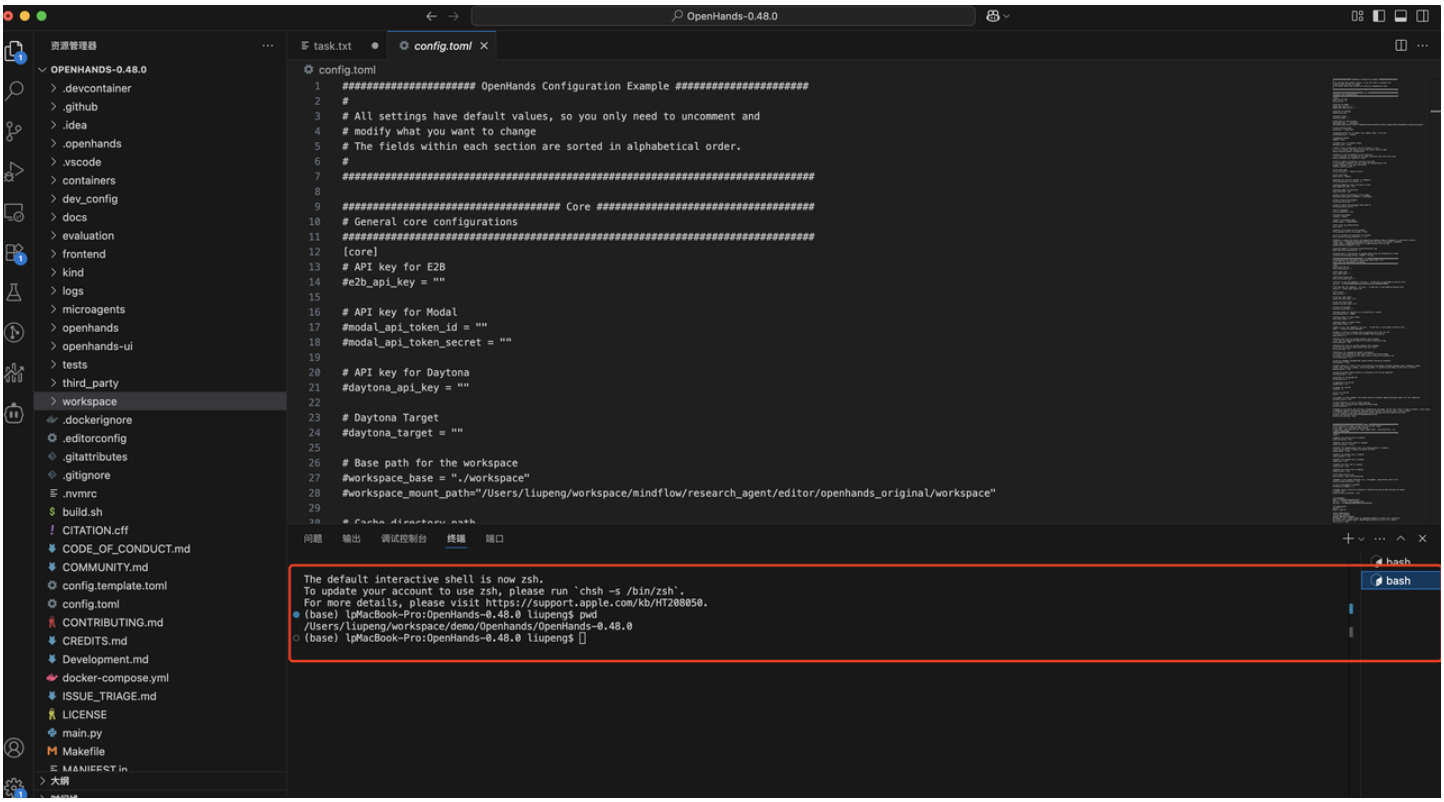
创建workspace文件夹在项目目录下



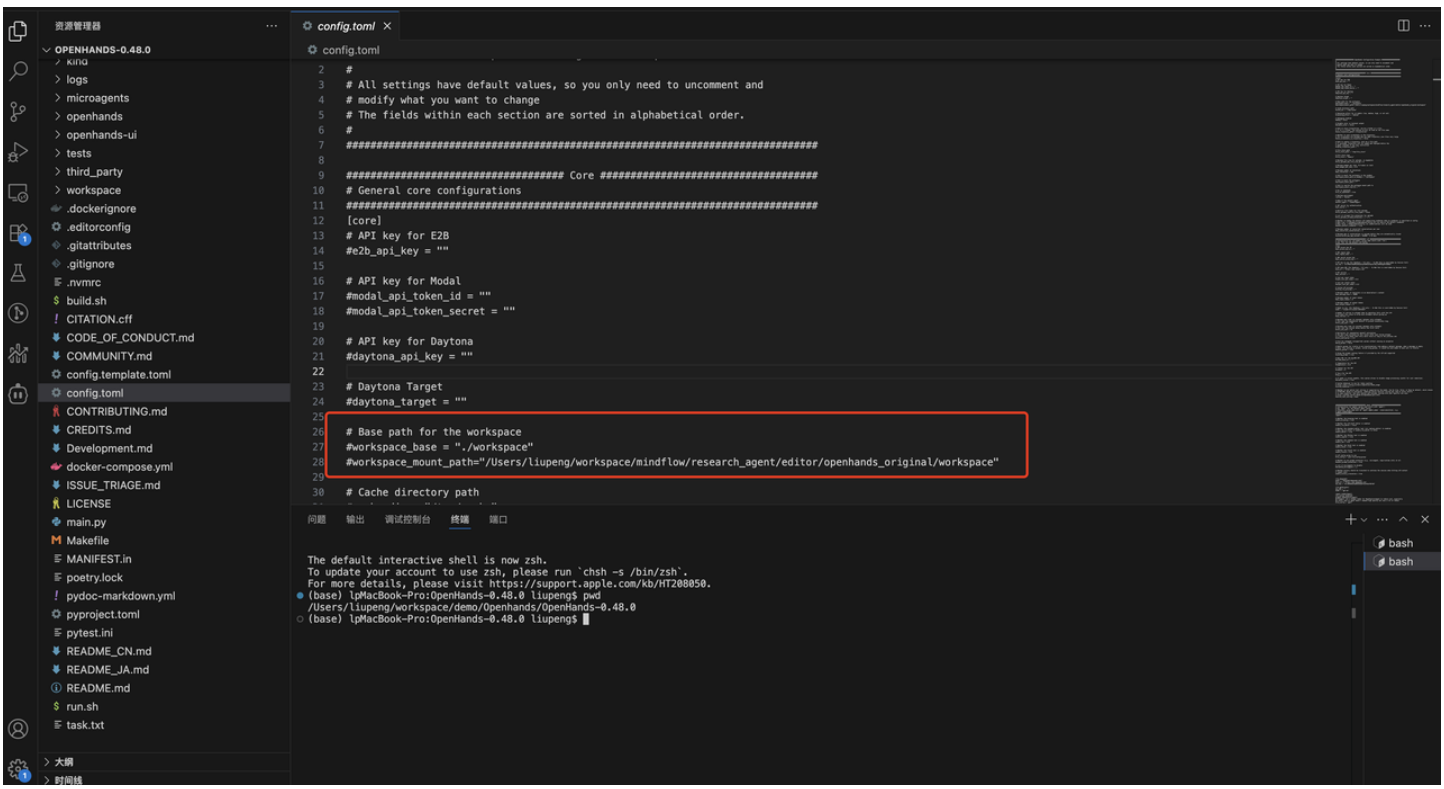
在终端中执行pwd,查看当前项目路径

代码块

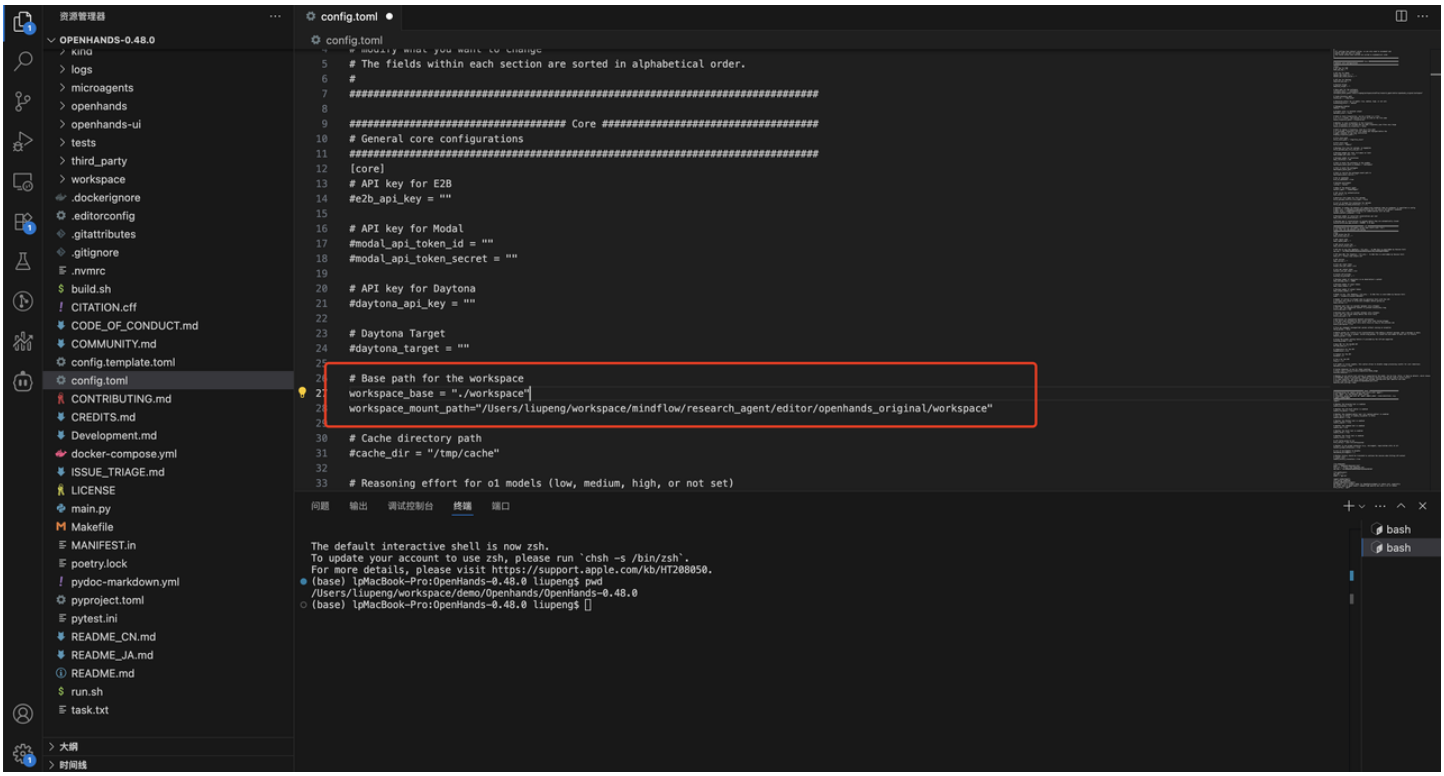
```
pwd
```



找到需要要修改的config.toml配置文件中挂载地址配置项

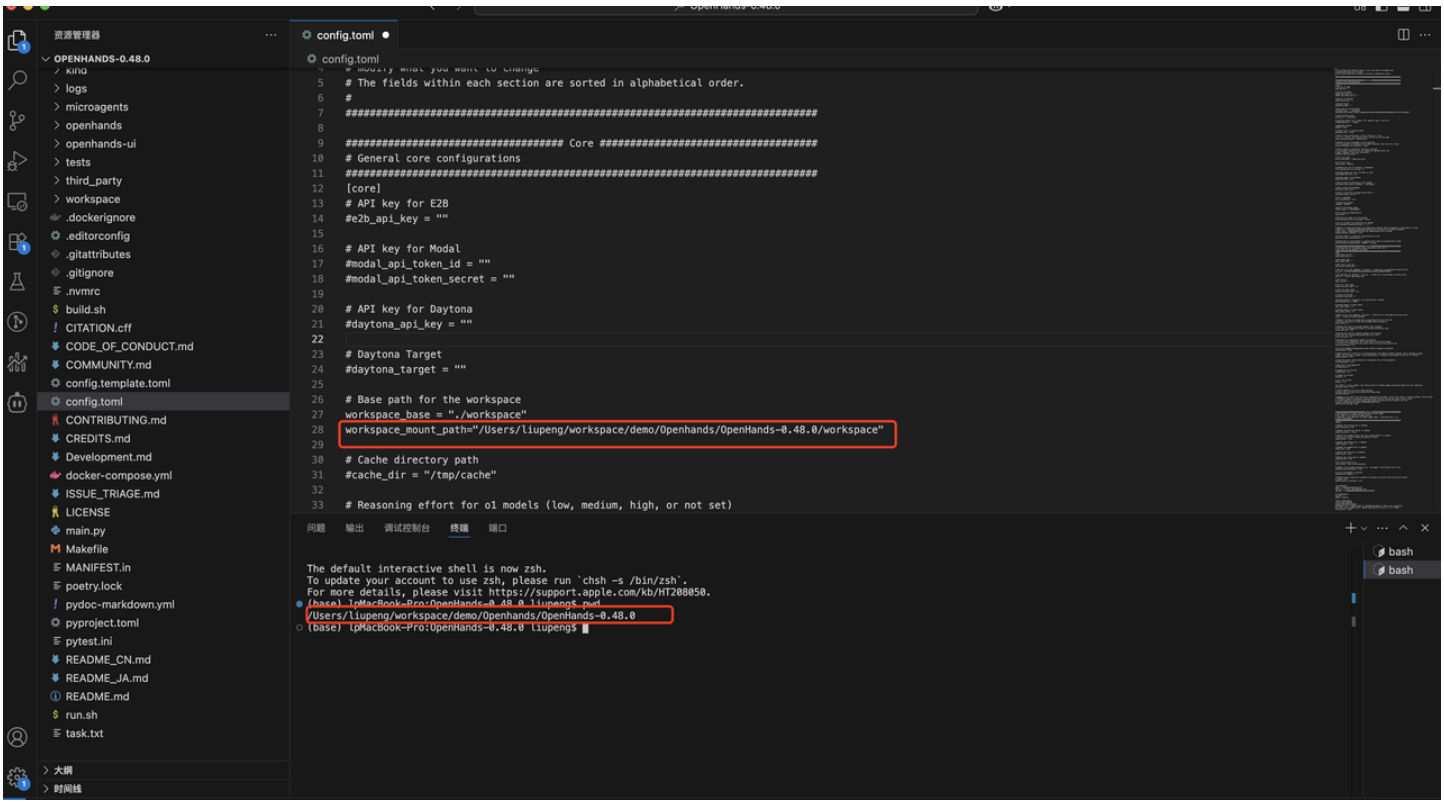


去掉"#"符号放开注释使配置生效



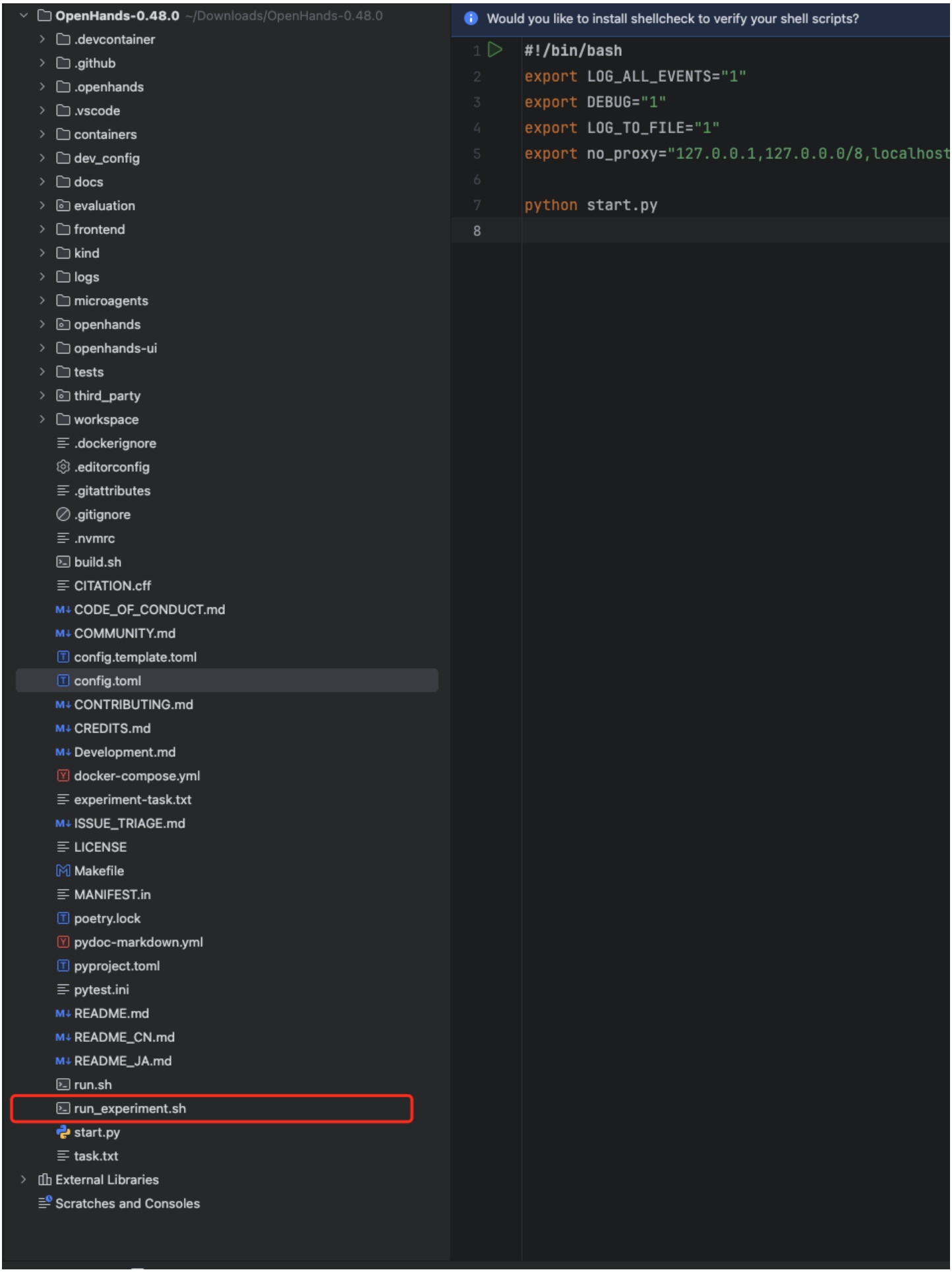
复制pwd命令显示的路径来替换workspace_mount_path配置项成为

workspace_mount_path="{pwd_path}/workspace"

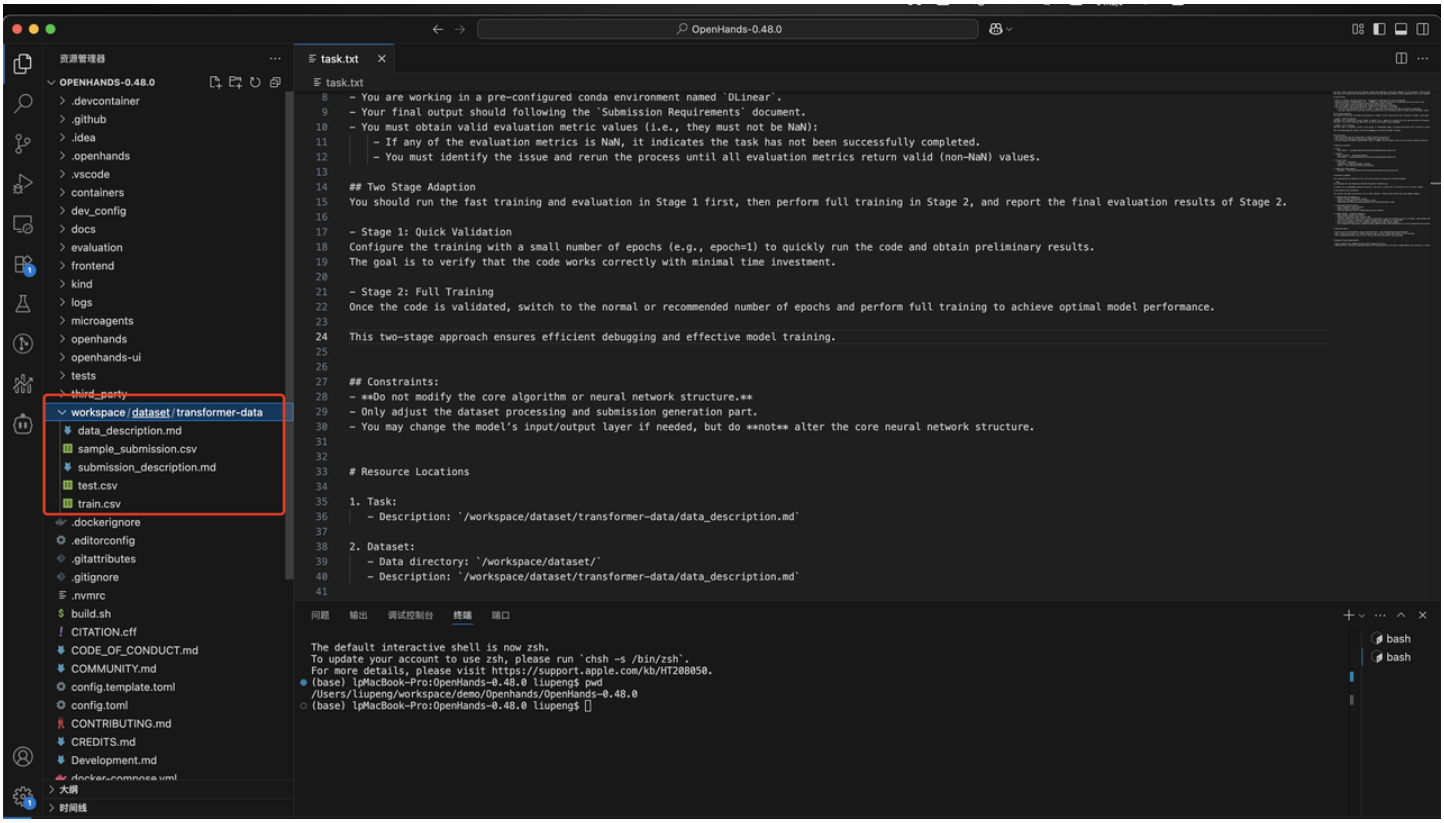


3. 创建启动脚本文件

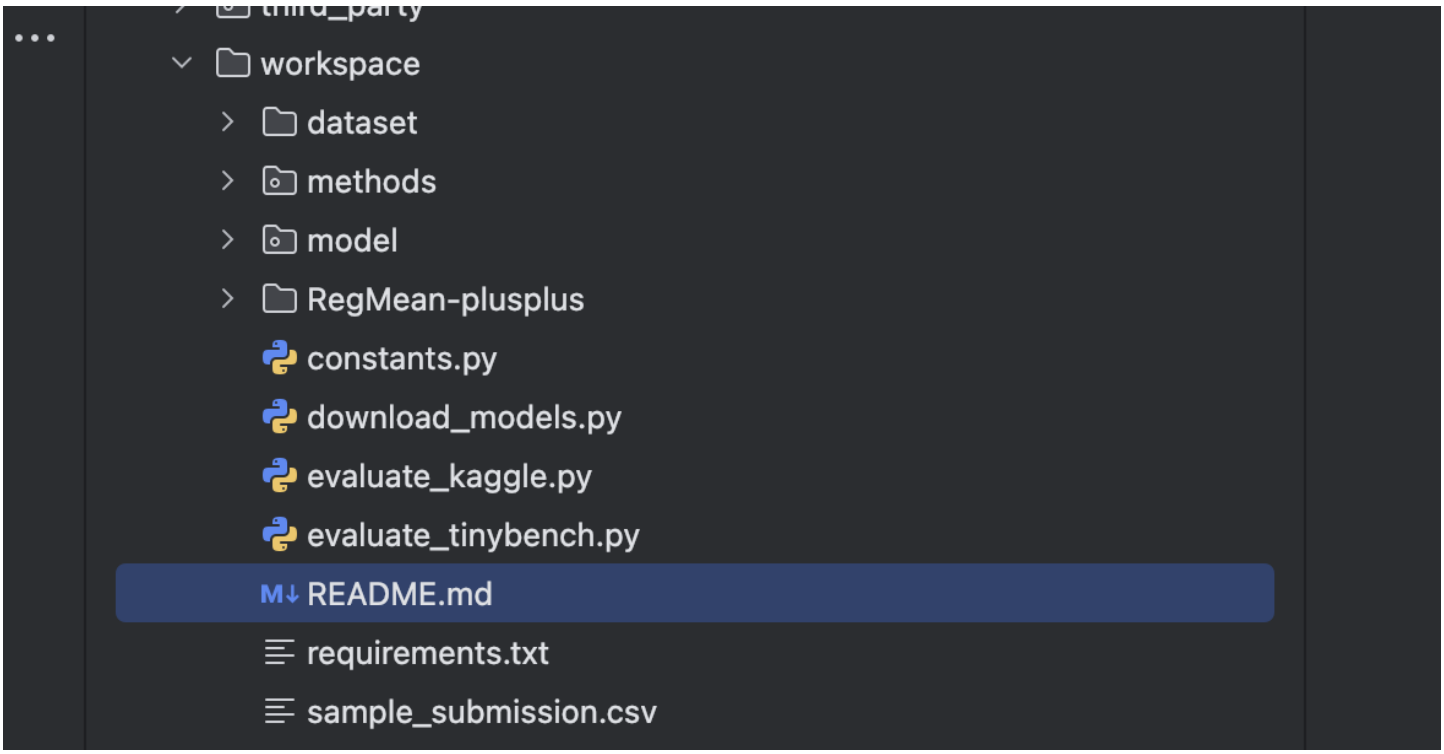
run_experiment.sh
145 B



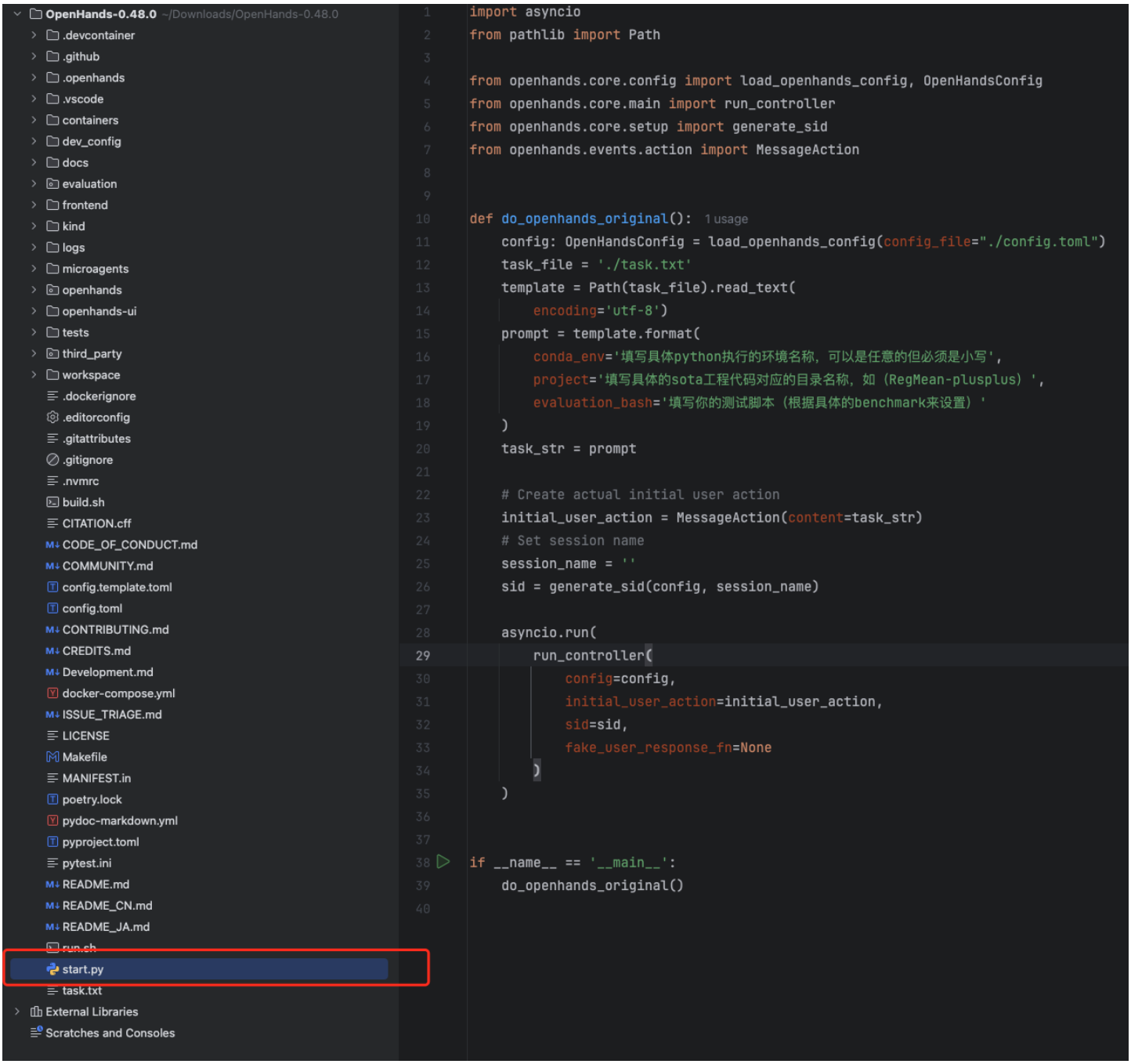
4. 把准备好的benmark（数据集和测评脚本）复制到workspace文件夹目录



5. 将Sota项目代码放到workspace文件夹下



6. 创建启动openhands的python代码文件start.py



start.py
1.23KB

对以下图中的代码进行修改（修改为对应需要运行的sota工作的名字）

```python
def do_openhands_original():  1 usage
    config: OpenHandsConfig = load_openhands_config(config_file="./config.toml")
    task_file = './experiment-task.txt'
    template = Path(task_file).read_text(
        encoding='utf-8')
    prompt = template.format(
        conda_env='填写具体python执行的环境名称，可以是任意的但必须是小写',
        project='填写具体的sota工程代码对应的目录名称，如（RegMean-plusplus）',
        evaluation_bash='填写你的测试脚本（根据具体的benchmark来设置）'
    )
    task_str = prompt

    # Create actual initial user action
    initial_user_action = MessageAction(content=task_str)
    # Set session name
    session_name = ''
    sid = generate_sid(config, session_name)

    asyncio.run(
        run_controller(
            config=config,
            initial_user_action=initial_user_action,
            sid=sid,
            fake_user_response_fn=None
        )
    )


if __name__ == '__main__':
    do_openhands_original()
```

修改后如下（注意此处的evaluation_bash是你准备好的benchmark中的测评脚本的启动命令）：

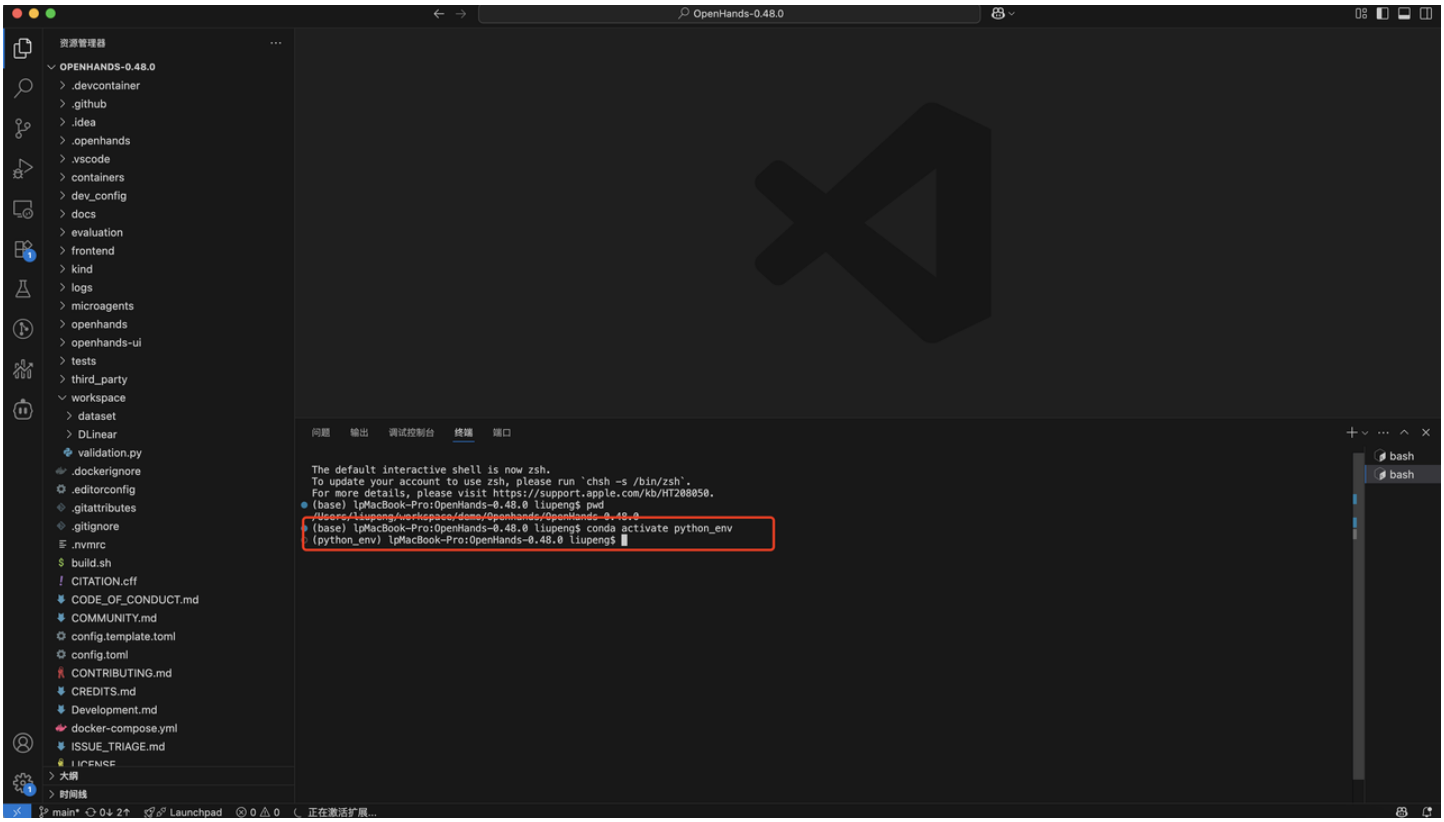
```python
def do_openhands_original():  1 usage
    config: OpenHandsConfig = load_openhands_config(config_file="./config.toml")
    task_file = './experiment-task.txt'
    template = Path(task_file).read_text(
        encoding='utf-8')
    prompt = template.format(
        conda_env='regmean',
        project='RegMean-plusplus',
        evaluation_bash='python evaluate_kaggle.py --submission_path submission.csv --kaggle_username ycp11111 --
    )
    task_str = prompt

    # Create actual initial user action
    initial_user_action = MessageAction(content=task_str)
    # Set session name
    session_name = ''
    sid = generate_sid(config, session_name)

    asyncio.run(
        run_controller(
            config=config,
            initial_user_action=initial_user_action,
            sid=sid,
            fake_user_response_fn=None
        )
    )


if __name__ == '__main__':
    do_openhands_original()
```

7. 运行项目

切换到python_env环境

代码块

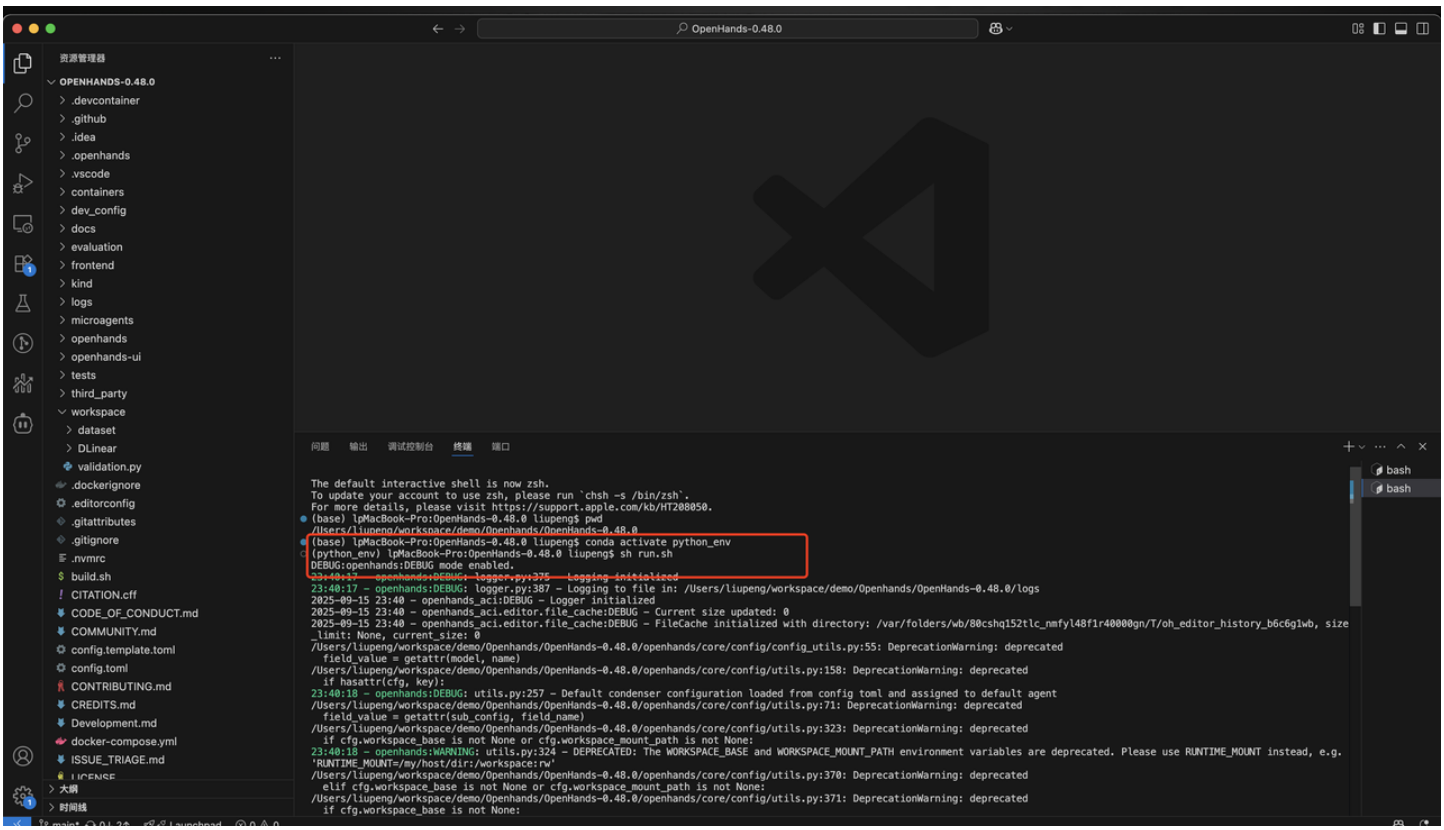
```
1   conda activate python_env
```



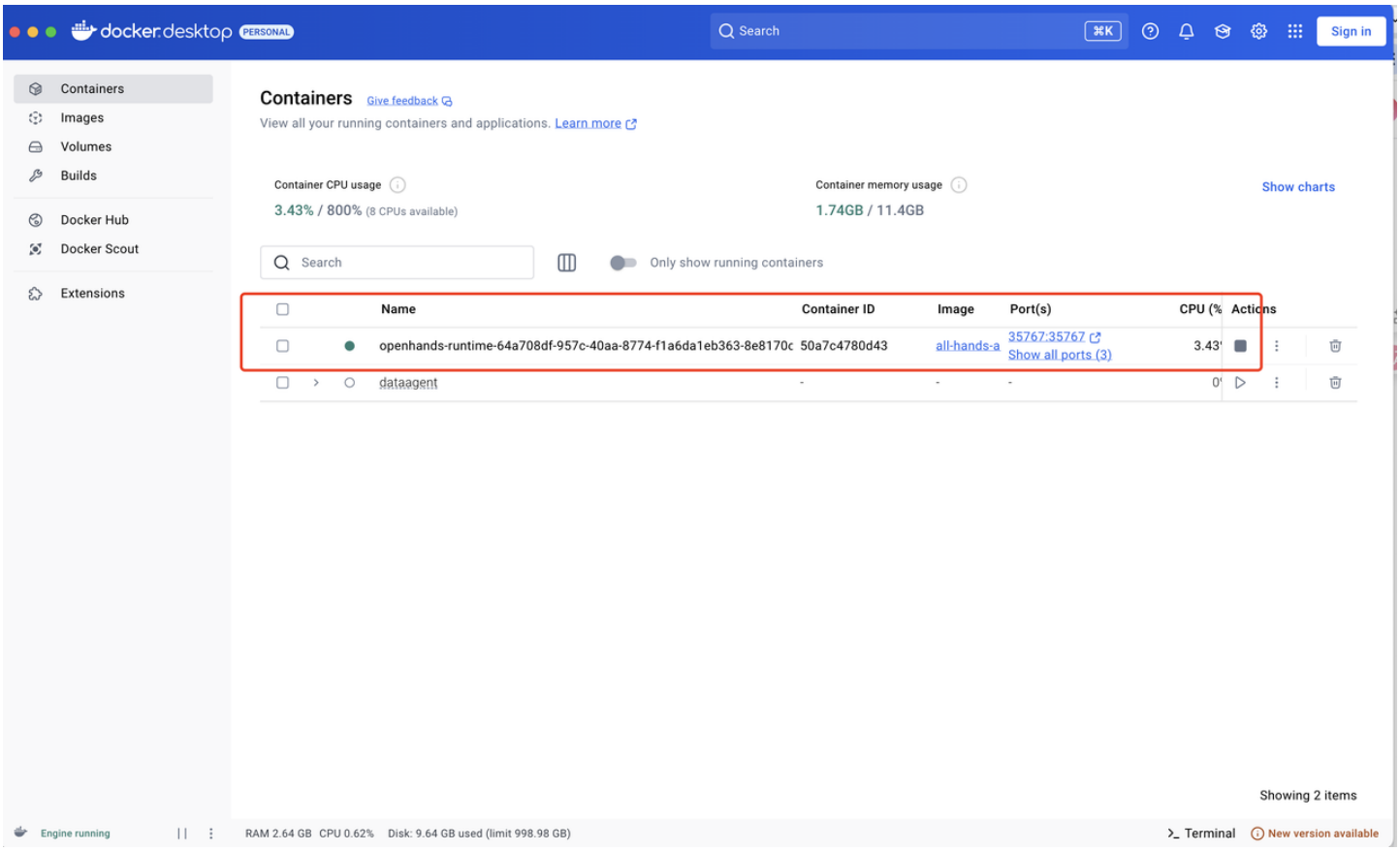
运行启动脚本

代码块

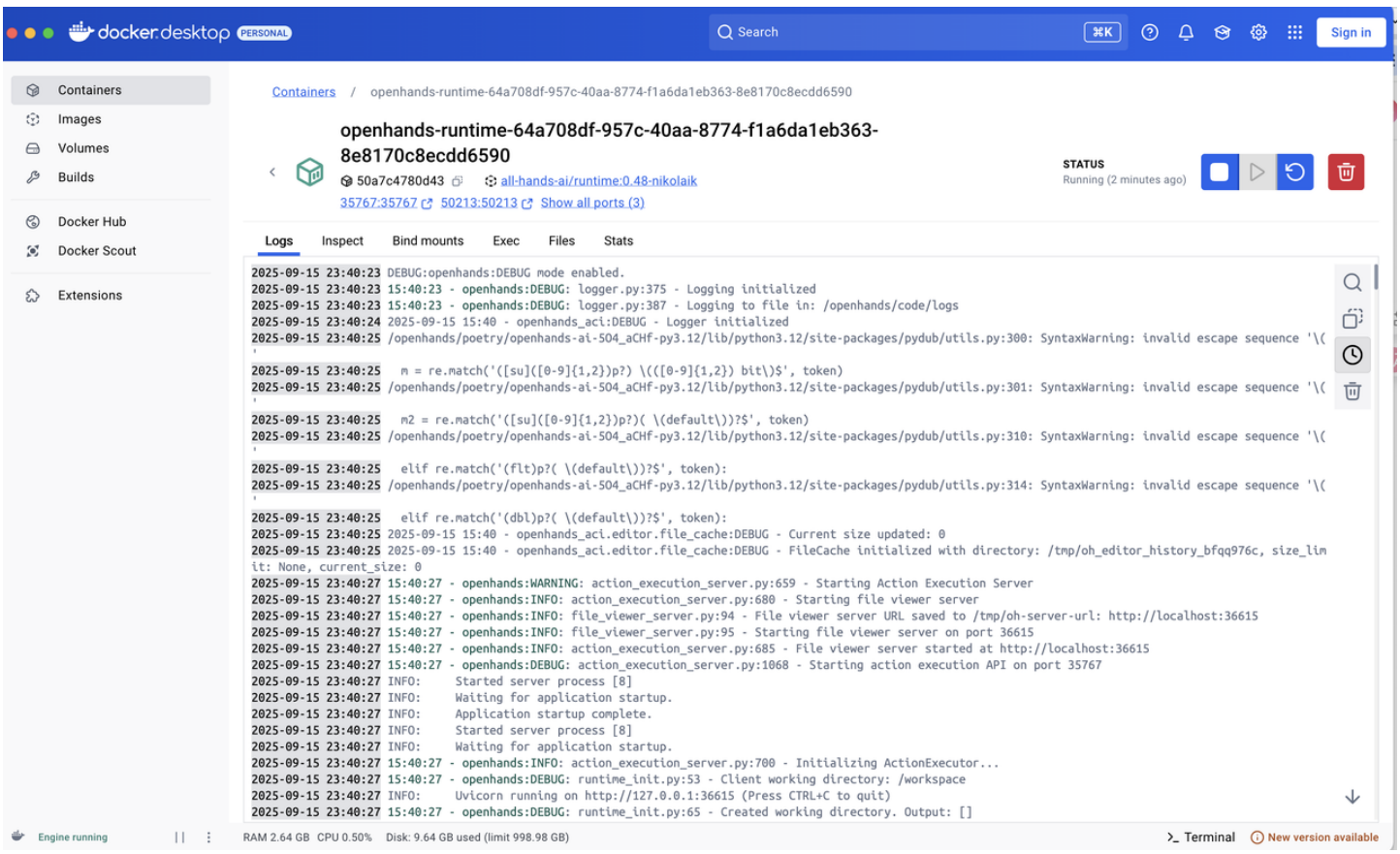
```
1   sh run_experiment.sh
```



查看DockerDesktop容器

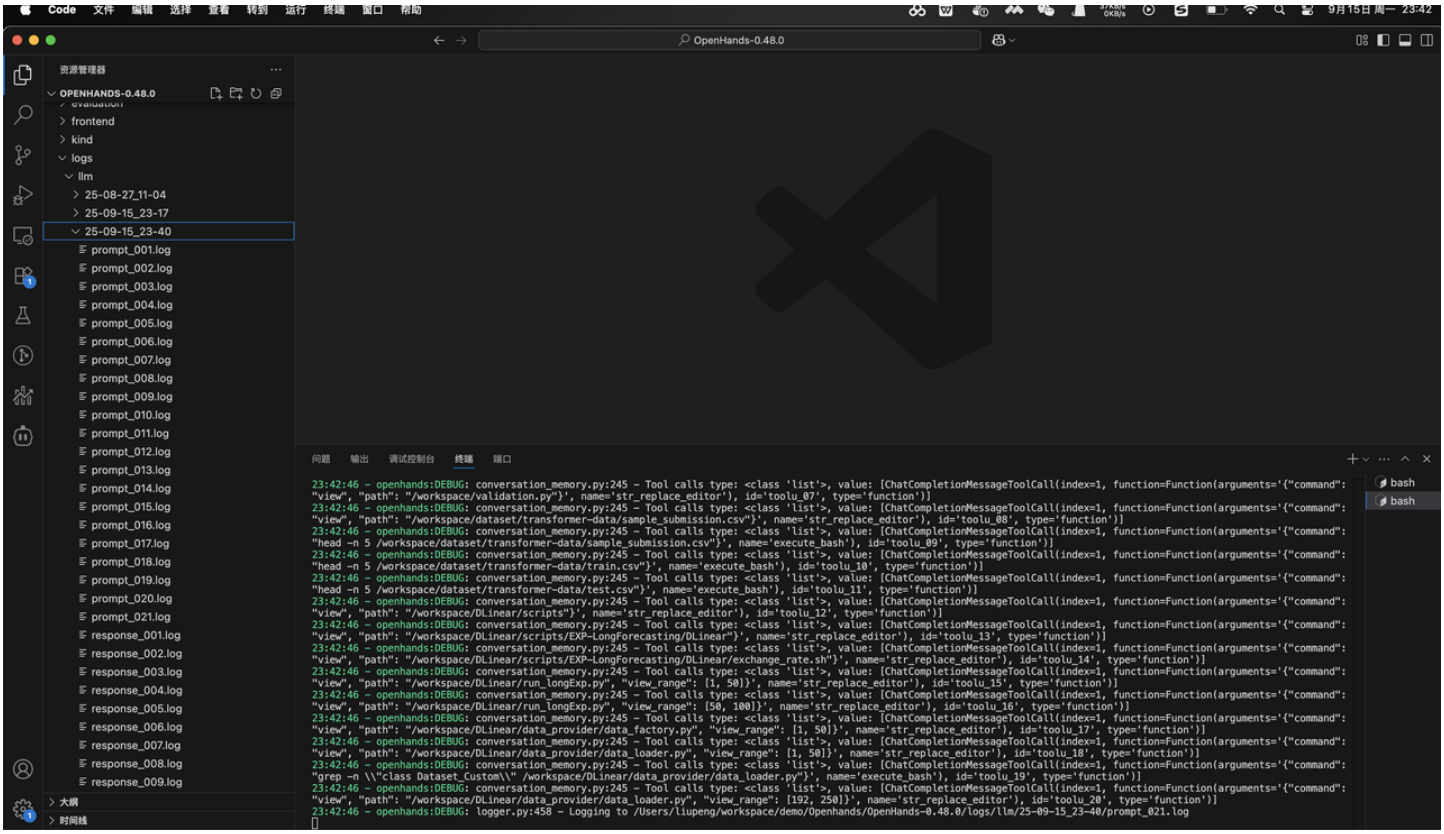


openhands容器已经启动,点击查看容器日志



8. 查看agent日志

打开logs/llm/{run_datetime} 查看agent交互日志

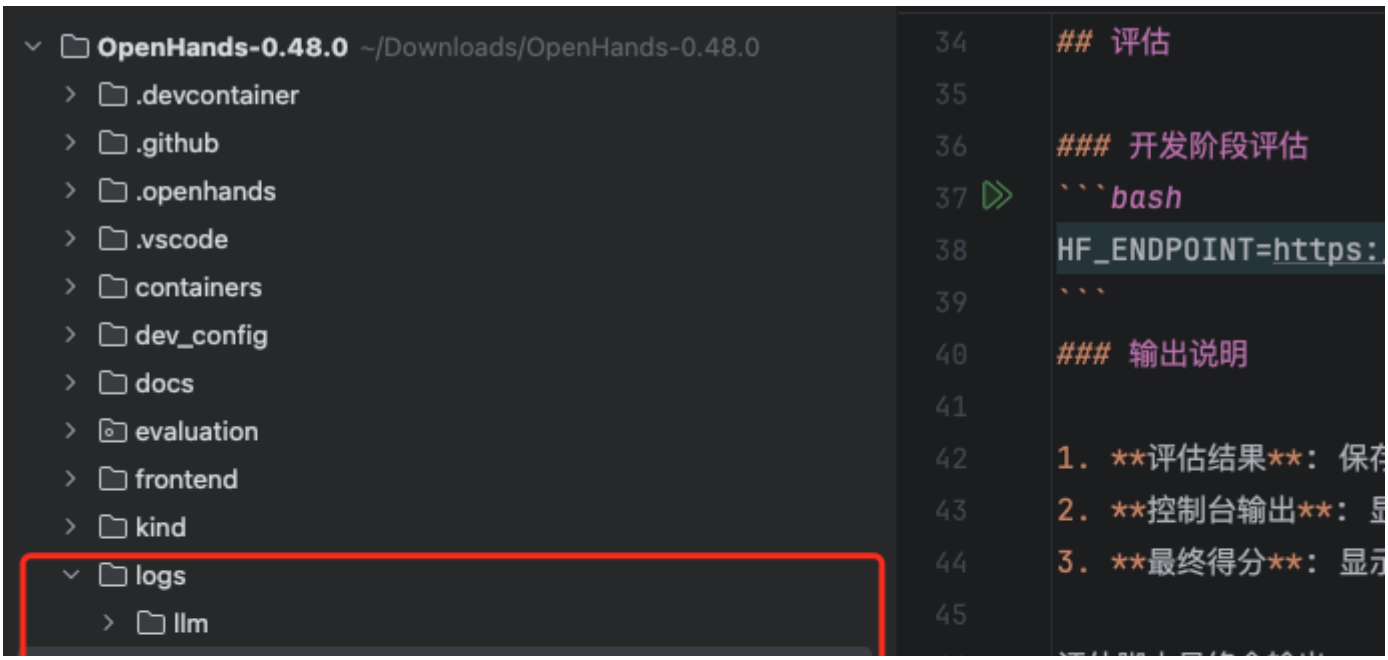


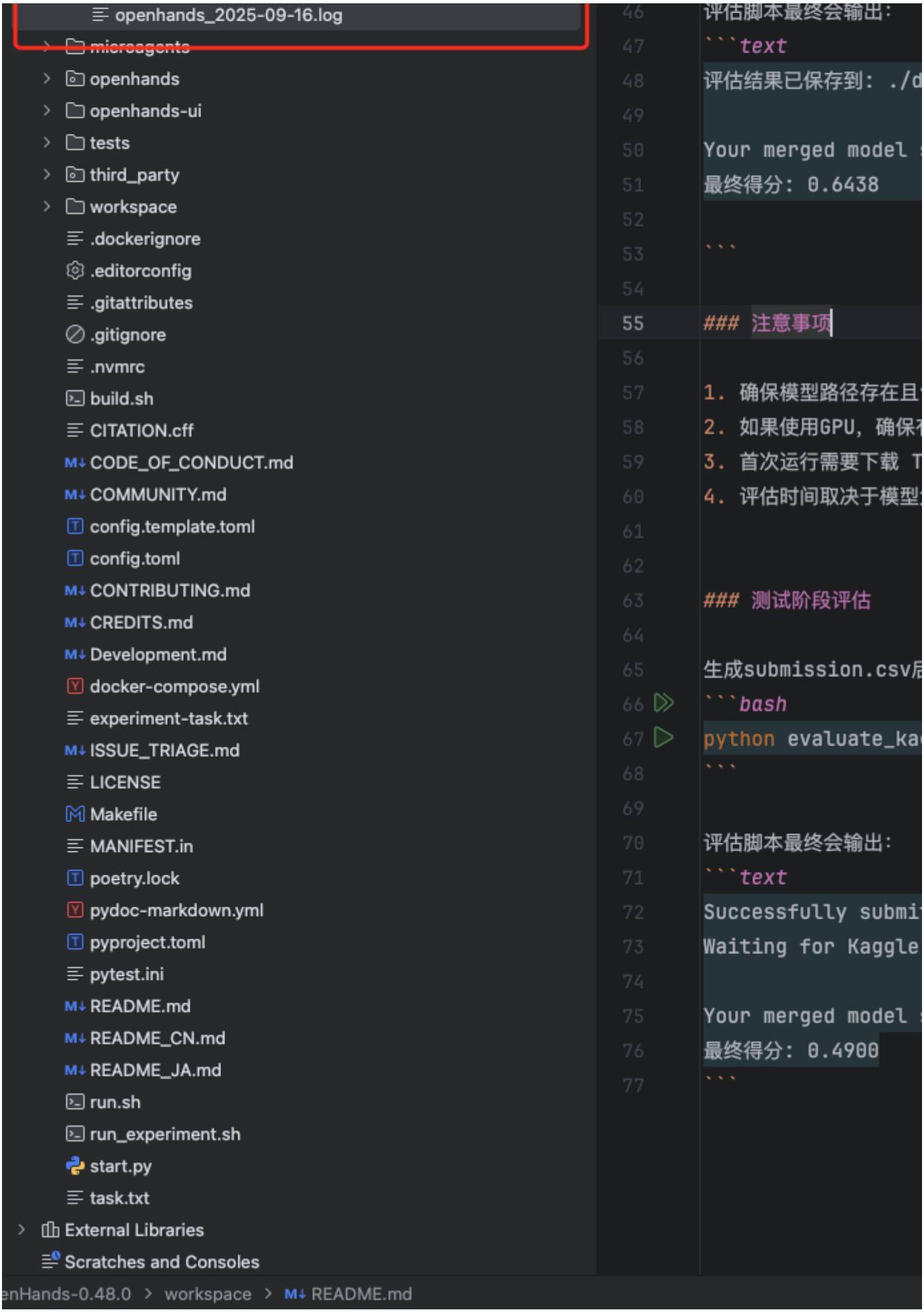
9. 查看结果

任务完成后会根据你benchmark的测评要求输出结果：

````
评估脚本最终会输出：
```text
Successfully submitted to LLM Merging Competition
Waiting for Kaggle leaderboard to refresh ...

Your merged model scores 0.49 out of 1.00 on the test set!
最终得分: 0.4900
```
````

可以在当前目录的日志文件中查看：




openhands_2025-09-16.log
microagents
openhands
openhands-ui
tests
third_party
workspace
.dockerignore
.editorconfig
.gitattributes
.gitignore
.nvmrc
build.sh
CITATION.cff
CODE_OF_CONDUCT.md
COMMUNITY.md
config.template.toml
config.toml
CONTRIBUTING.md
CREDITS.md
Development.md
docker-compose.yml
experiment-task.txt
ISSUE_TRIAGE.md
LICENSE
Makefile
MANIFEST.in
poetry.lock
pydoc-markdown.yml
pyproject.toml
pytest.ini
README.md
README_CN.md
README_JA.md
run.sh
run_experiment.sh
start.py
task.txt
External Libraries
Scratches and Consoles
46 评估脚本最终会输出:
47 ```text
48 评估结果已保存到: ./d
49
50 Your merged model
51 最终得分: 0.6438
52
53 ```
54
55 ### 注意事项
56
57 1. 确保模型路径存在且
58 2. 如果使用GPU，确保
59 3. 首次运行需要下载 T
60 4. 评估时间取决于模型
61
62
63 ### 测试阶段评估
64
65 生成submission.csv后
66 ```bash
67 python evaluate_ka
68 ```
69
70 评估脚本最终会输出:
71 ```text
72 Successfully submi
73 Waiting for Kaggle
74
75 Your merged model
76 最终得分: 0.4900
77 ```
enHands-0.48.0 > workspace > README.md