6502 words | 17 minutes

Introduction to Large Language Model (LLM) Theory

1. What is a Large Language Model (LLM)

1.1 Concept of Large Language Model (LLM)

A large language model (LLM) is an artificial intelligence model designed to understand and generate human language .

LLM usually refers to **language models containing tens of billions (or more) parameters** , which are trained on massive amounts of text data to gain a deep understanding of the language. At present, well-known LLMs abroad include GPT-3.5, GPT-4, PaLM, Claude and LLaMA, and domestic ones include Wenxin Yiyan, iFlytek Spark, Tongyi Qianwen, ChatGLM, Baichuan, etc.

In order to explore the limits of performance, many researchers began to train increasingly large language models, such as with `1750 亿` parameters `GPT-3` and `5400 亿` with parameters `PaLM` . Although these large language models use similar architectures and pre-training tasks as small language models (such as `3.3 亿` with parameters `BERT` and `15 亿` with parameters `GPT-2`), they exhibit completely different capabilities, especially showing amazing potential in solving complex tasks, which is called " **emergent ability** ". Taking GPT-3 and GPT-2 as examples, GPT-3 can solve few-sample tasks by learning context, while GPT-2 performs poorly in this regard. Therefore, the scientific research community has given these huge language models a name, calling them "large language models (LLM)". An outstanding application of LLM is **ChatGPT** , which is a bold attempt to use the GPT series LLM for conversational applications with humans, showing a very smooth and natural performance.

1.2 Development History of LLM

The study of language modeling can be traced back to `20 世纪 90 年代` the time when the research focused on using **statistical learning methods** to predict words, predicting the next word by analyzing the previous words. However, it has certain limitations in understanding complex language rules.

Subsequently, researchers continued to try to improve it. **Bengio** 2003 年 , a pioneer in deep learning , first incorporated the idea of deep learning into the language model in his classic paper . The powerful **neural network model** is equivalent to providing a powerful "brain" for computers to understand language, allowing the model to better capture and understand the complex relationships in language. 《A Neural Probabilistic Language Model》

2018 年 Around this time, **neural network models with Transformer architecture** began to emerge. These models were trained with large amounts of text data, enabling them to deeply understand language rules and patterns by reading large amounts of text, just like letting computers read the entire Internet. This gave them a deeper understanding of language and greatly improved the performance of the models on various natural language processing tasks.

At the same time, researchers found that as **the size of the language model increases (increasing the model size or using more data)** , the model shows some amazing capabilities and significantly improves performance in various tasks. This discovery marks the beginning of the era of large language models (LLMs).

1.3 Common LLM models

Although the development of large language models has only been less than five years, the development speed is quite amazing. As of June 2023, more than 100 large models have been released at home and abroad. The following figure shows the influential large language models with more than 10 billion model parameters from 2019 to June 2023 according to the timeline:

(This figure is from reference [1])

Next, we will mainly introduce several common large models at home and abroad (including open source and closed source)

1.3.1 Closed-source LLM (undisclosed source code)

1.3.1.1 GPT series

> OpenAI Model Introduction

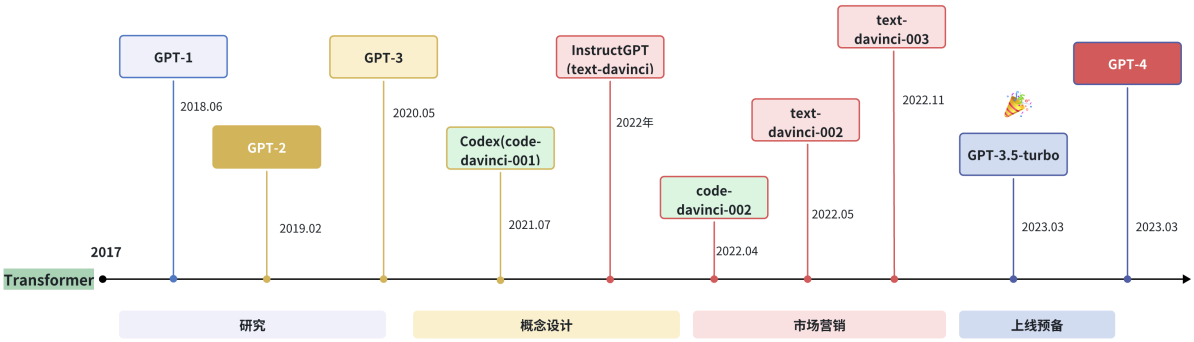
The GPT (Generative Pre-Training) 2018 年 model proposed by **OpenAI** is a typical one. Generative pre-trained language model

The basic principle of the GPT model is **to compress world knowledge into a decoder-only Transformer model through language modeling** , so that it can recover (or remember) the

semantics of world knowledge and act as a general task solver. There are two key points to its success:

- Train a decoder-only Transformer language model that can accurately predict the next word
- Scaling the size of language models

OpenAI's research on LLM can be roughly divided into the following stages:



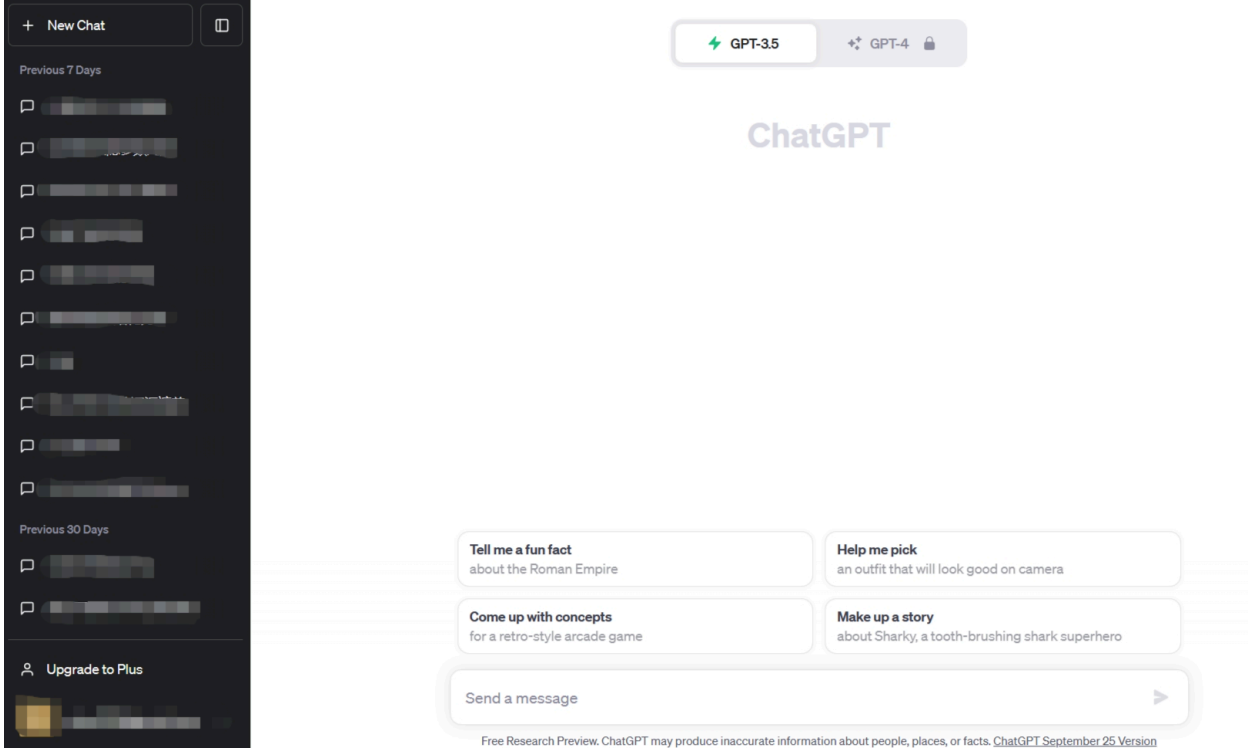
Next, we will introduce the well-known ChatGPT and GPT4 from the aspects of model scale and characteristics:

1.3.1.1.1 ChatGPT

> [ChatGPT usage address]

2022 年 11 月 , **OpenAI** released **ChatGPT, a conversational application** based on the GPT model (GPT-3.5 and GPT-4) . ChatGPT has sparked excitement in the AI community since its release due to its outstanding ability to communicate with humans. ChatGPT is developed based on the powerful GPT model with specially optimized conversational capabilities.

ChatGPT is essentially an LLM application, which is developed based on the base model and is fundamentally different from the base model. It supports two versions: GPT-3.5 and GPT-4.



Today's ChatGPT supports up to 32,000 characters, with a knowledge deadline of September 2021. It can perform a variety of tasks, including **code writing, math problem solving, writing suggestions** , and more. ChatGPT has demonstrated outstanding ability to communicate with humans: it has a rich knowledge reserve, the skill to reason about math problems, accurately tracks context in multi-turn conversations, and is very consistent with the values of human safety use. Later, ChatGPT supported a plug-in mechanism, which further expanded the capabilities of ChatGPT with existing tools or applications. So far, it seems to be the most powerful chatbot in the history of artificial intelligence. The launch of ChatGPT has a significant impact on future artificial intelligence research, and it provides inspiration for exploring human-like artificial intelligence systems.

1.3.1.1.2 GPT-4

2023 年 3 月 GPT-4 was released, which **expands text input to multimodal signals** . GPT3.5 has 175 billion parameters, and the number of parameters of GPT4 has not been officially announced, but relevant personnel have speculated that GPT-4 contains a total of 1.8 trillion parameters in 120 layers, which means that the scale of GPT-4 is more than 10 times that of GPT-3. Therefore, GPT-4 **is more capable of solving complex tasks than GPT-3.5, and has shown significant performance improvements on many evaluation tasks** .

A recent study investigated the capabilities of GPT-4 by qualitatively testing it on artificially generated questions covering a wide variety of difficult tasks and showed that GPT-4 can achieve superior performance than previous GPT models such as GPT3.5. In addition, thanks to six months of iterative calibration (with additional safety reward signals in RLHF training), GPT-4

responds more safely to malicious or provocative queries and applies some intervention strategies to mitigate issues that may arise with LLM, such as hallucinations, privacy, and over-reliance.

> **Note: On November 7, 2023, OpenAI held its first developer conference, where it launched its latest large language model, GPT-4 Turbo, which is equivalent to an advanced version. It extends the context length to 128k, equivalent to 300 pages of text, and updates the training knowledge to April 2023.**

GPT3.5 is free, but GPT-4 is paid. You need to subscribe to the plus membership for $20/month.

2024 年 5 月 14 日 , the new generation flagship generative model **GPT-4o** was officially released. GPT-4o has the ability to deeply understand the three modalities of text, voice, and image, and is quick to respond, emotional, and extremely human. Moreover, GPT-4o is completely free, although the number of free uses per day is limited.

Usually we can call the model API to develop our own applications. **The comparison of mainstream model APIs** is as follows:

Language model name	Context length	Features	Input Fee ($/million tokens)	Output Fee ($/ 1M tokens)	Knowledge Deadline
GPT-3.5-turbo-0125	16k	Economy, specialized dialogue	0.5	1.5	September 2021
GPT-3.5-turbo-instruct	4k	Instruction Model	1.5	2	September 2021
GPT-4	8k	Better performance	30	60	September 2021
GPT-4-32k	32k	Strong performance, long context	60	120	September 2021
GPT-4-turbo	128k	Better performance	10	30	December 2023
GPT-4o	128k	Highest performance, faster speed	5	15	October 2023

Embedding model name	Dimensions	Features	Fees ($/ 1M tokens)
text-embedding-3-small	512/1536	Smaller	0.02
text-embedding-3-large	256/1024/3072	Larger	0.13
ada v2	1536	Tradition	0.1

1.3.1.2 Claude Series

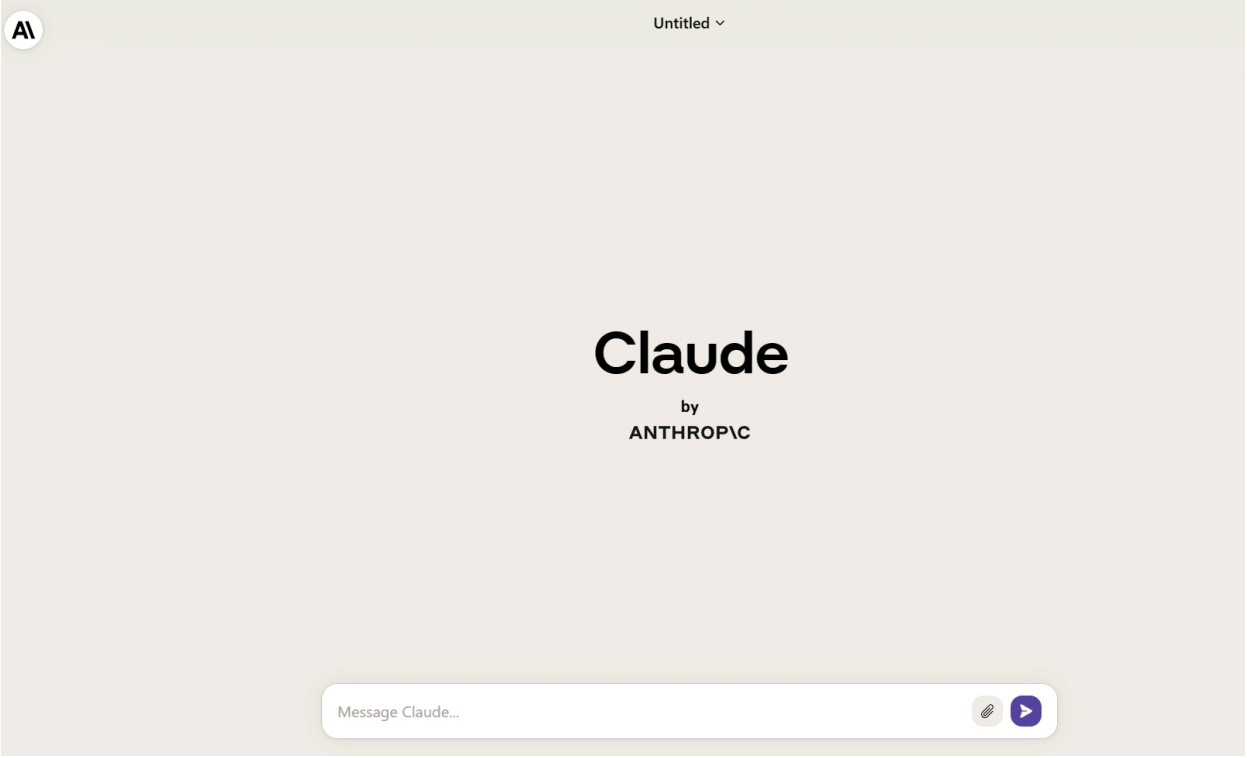
The Claude series of models are large closed-source language models developed by **Anthropic,** a company founded by former OpenAI employees .

> Claude uses the address

The earliest **Claude** was released on March 15, 2023 , and on July 11, 2023, it was updated to **Claude-2** , and then March 4, 2024 to **Claude-3** .

The Claude 3 series includes three different models, namely Claude 3 Haiku, Claude 3 Sonnet and Claude 3 Opus, with increasing capabilities to meet the needs of different users and application scenarios.

Model Name	Context length	Features	Input Fee ($/1M tokens)	Output Fee ($/1M tokens)
Claude 3 Haiku	200k	Fastest	0.25	1.25
Claude 3 Sonnet	200k	balance	3	15
Claude 3 Opus	200k	Highest performance	15	75



1.3.1.1.3 PaLM/Gemini Series

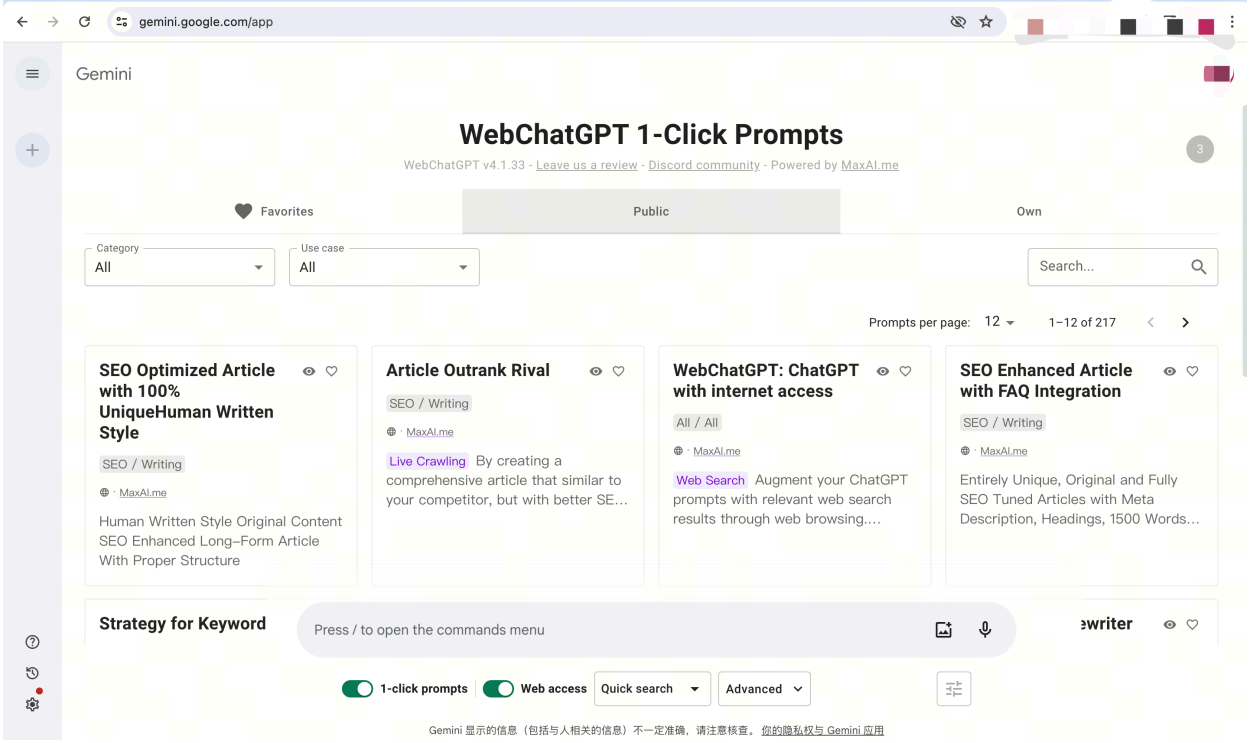
The PaLM series of language models was developed by **Google** . Its initial version was 2022 年 4 月 released in 2020, and its API was made public in March 2023. In May 2023, Google released **PaLM 2.** Google 2024 年 2 月 1 日 changed the underlying model driver of Bard (a previously released conversational application) from PaLM2 to **Gemini** , and also renamed the original Bard to **Gemini** .

> PaLM official website

> Gemini usage address

The current Gemini is the first version, Gemini 1.0, which is divided into three versions: Ultra, Pro and Nano according to different parameter quantities.

The following window is the Gemini interface:



1.3.1.1.4 Wen Xin Yi Yan

> [Wenxinyiyan usage address]

Wenxinyiyan is a knowledge-enhanced language big model based on Baidu's Wenxin big model . It 2023 年 3 月 was the first to be launched in China. Wenxinyiyan's basic model, Wenxin big model, released version 1.0 in 2019 and has now been updated to version **4.0** . Further classification, Wenxin big model includes NLP big model, CV big model, cross-modal big model, biocomputing big model, and industry big model. It is a closed-source model with relatively good Chinese capabilities.

The web version of Wenxin Yiyan is divided into **free version** and **professional version** .

- The free version uses Wenxin 3.5, which can already meet most of the needs of individual users or small businesses.
- The professional version uses Wenxin 4.0. The price is 59.9 yuan/month, and the monthly discount price is 49.9 yuan/month.

You can also use the API to make calls (**billing details**).

The following is the user interface of Wenxinyiyan:

1.3.1.1.5 Spark Large Model

> [Spark Large Model Usage Address]

iFlytek Spark Cognitive Big Model is a language big model released by **iFlytek** 2023 年 5 月 , which supports a variety of natural language processing tasks. The model was first released and has been upgraded many times. 2023 年 10 月 iFlytek released **iFlytek Spark Cognitive Big Model V3.0** . 2024 年 1 月 iFlytek released **iFlytek Spark Cognitive Big Model V3.5** , which has been upgraded in seven aspects including language understanding, text generation, knowledge question and answer, and supports multiple functions such as system instructions and plug-in calls.

The following is the user interface of iFlytek Spark:

1.3.2. Open Source LLM

1.3.2.1 LLaMA series

> [LLaMA official website]

> [LLaMA open source address]

The LLaMA series of models is a set of basic language models open sourced by Meta with parameter sizes ranging from 7B to 70B. LLaMA was released in February 2023, the LLaMA2 model was released in July 2023, and the LLaMA3 model was released on April 18, 2024. They are both trained on trillions of characters, demonstrating how to train state-of-the-art models using only publicly available datasets, without relying on proprietary or inaccessible datasets. These datasets include Common Crawl, Wikipedia, OpenWebText2, RealNews, Books, etc. The LLaMA model uses large-scale data filtering and cleaning techniques to improve data quality and diversity and reduce noise and bias. The LLaMA model also uses efficient data parallelism and pipeline parallelism technology to accelerate model training and expansion. In particular, LLaMA 13B surpasses GPT-3 (175B) on 9 benchmarks such as CommonsenseQA, while LLaMA 65B is comparable to the state-of-the-art models Chinchilla-70B and PaLM-540B. LLaMA achieves optimal performance by using fewer characters, giving it advantages under a variety of inference budgets.
Like the GPT series, the LLaMA model also adopts a decoder-only architecture and combines some improvements from previous work:

- Pre-normalization regularization: In order to improve training stability, LLaMA performs RMSNorm normalization on the input of each Transformer sub-layer. This normalization method can avoid the problem of gradient explosion and disappearance, and improve the convergence speed and performance of the model. ;
- SwiGLU activation function: Replace the ReLU nonlinearity with the SwiGLU activation function to increase the expressive power and nonlinearity of the network while reducing the amount of parameters and calculations;
- Rotary Position Embedding (RoPE, Rotary Position Embedding): The input of the model no longer uses position encoding, but adds position encoding to each layer of the network. RoPE position encoding can effectively capture the relative position information in the input sequence, and has Better generalization ability.

LLaMA3 improves upon the LLaMA series of models to increase performance and efficiency:

- More training data: LLaMA3 is pre-trained on 15 trillion tokens of data, which is 7 times more than LLaMA2, and 4 times more code data. LLaMA3 is exposed to more text information, which improves its ability to understand and generate text.
- Longer context length: LLaMA3's context length has doubled from 4096 tokens in LLaMA2 to 8192. This enables LLaMA3 to process longer text sequences and improves its ability to understand and generate long texts.
- Grouped-Query Attention (GQA): By grouping queries and sharing keys and values within the group, the amount of computation is reduced while maintaining model performance, improving the inference efficiency of large models (LLaMA2 only uses 70B).
- Larger vocabulary: LLaMA3 has been upgraded to a 128K tokenizer, which is four times the 32K of the previous two generations. This greatly enhances its semantic encoding capabilities, thereby significantly improving the performance of the model.

1.3.2.2 Thousand Questions on Tongyi

> **Tongyi Qianwen usage address**

Tongyi Qianwen open source address

Tongyi Qianwen was developed by Alibaba based on the "Tongyi" large model and officially released in April 2023. In September 2023, Alibaba Cloud open-sourced the Qwen (Tongyi Qianwen) series of work. On February 5, 2024, Qwen1.5 (beta version of Qwen2) was open-sourced. And on June 6, 2024, Qwen2 was officially open-sourced. Qwen2 is a decoder-only model that uses the architecture of SwiGLU activation, RoPE, and GQA. It is an open-source model with relatively good Chinese capabilities.
Currently, 5 model sizes have been open-sourced: 0.5B, 1.5B, 7B, 72B Dense models and 57B (A14B) MoE models; all models support contexts with a length of 32768 tokens. And the context length of Qwen2-7B-Instruct and Qwen2-72B-Instruct is extended to 128K tokens.
The following is the user interface of Tongyi Qianwen:

1.3.2.3 GLM series

ChatGLM Use address

ChatGLM Open source address

The GLM series of models are large language models developed jointly by Tsinghua University and Zhipu AI. ChatGLM was released in March 2023. ChatGLM 2 was released in June. ChatGLM3 was launched in October. GLM4 was released on January 16, 2024, and officially open sourced on June 6, 2024.

GLM-4-9B-Chat supports multi-round conversations, web browsing, code execution, custom tool calls (Function Call), and long text reasoning (supporting up to 128K context).

The dialogue model GLM-4-9B-Chat, the basic model GLM-4-9B, the long text dialogue model GLM-4-9B-Chat-1M (supporting 1M context length), and the multimodal model GLM-4V-9B are open sourced to fully benchmark OpenAI:

以下是智谱清言的使用界面：

1.3.2.4 Baichuan series

> [Baichuan Use Address]

> [Baichuan Open Source Address]

Baichuan is an open-source commercial language model developed by Baichuan Intelligence. It is based on the Transformer decoder architecture (decoder-only).

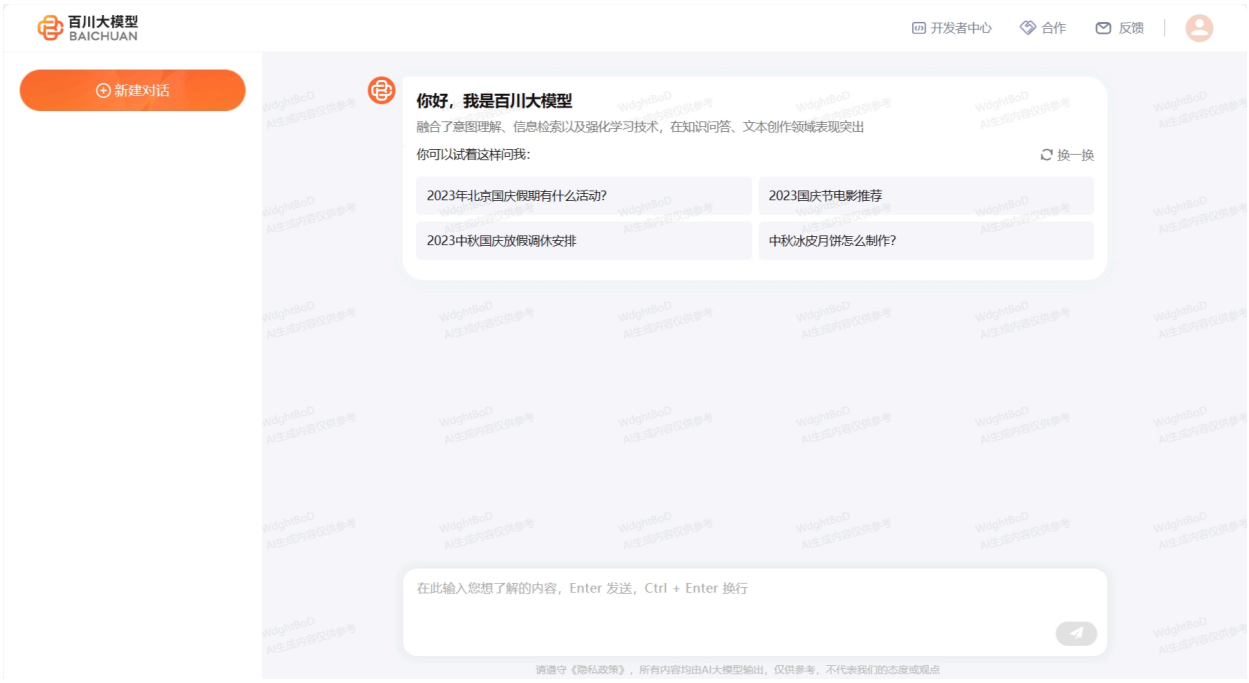
Baichuan-7B and Baichuan-13B were released on June 15, 2023. Baichuan also open-sourced the pre-training and alignment models.

The pre-trained model is the "foundation" for developers, while the aligned model is for ordinary users who need conversational functions.

Baichuan2 was launched in 2011. **7B and 13B** Base **and** Chat **versions** 2023年 9 月 6 日 were released , and **4bits quantization** was provided for the Chat version .

2024 年 1 月 29 日 **Baichuan 3** has been released . However, **it is not open source yet** .

The following is the user interface of Baichuan model:



II. LLM's Capabilities and Characteristics

2.1 LLM Capabilities

2.1.1 Emergent abilities

One of the most notable features that distinguish large language models (LLMs) from previous pre-trained language models (PLMs) is their Emergent Capabilities emergence ability, a surprising ability that is not obvious in small models but is particularly prominent in large models. Similar to the phase transition phenomenon in physics, the emergence ability is like the model performance rapidly improving as the scale increases, exceeding the random level, which is what we often call **quantitative change leading to qualitative change** .

Emergent capabilities can be related to certain complex tasks, but we are more concerned with its general capabilities. Next, we briefly introduce three typical emergent capabilities of LLM:

1. **Contextual learning** : The contextual learning capability was first introduced by GPT-3. This capability allows the language model to perform tasks by understanding the context and generating corresponding outputs when provided with natural language instructions or multiple task examples, without the need for additional training or parameter updates.

2. **Instruction Following** : Fine-tuned on multi-task data using natural language descriptions, so-called Instruction fine-tuning LLMs are shown to perform well on unseen tasks formally described using instructions. This means that LLMs are able to perform tasks based on task instructions without having seen specific examples beforehand, demonstrating their strong generalization capabilities.

3. **Step-by-step reasoning** : Small language models often have difficulty solving complex tasks that involve multiple reasoning steps, such as math problems. However, LLMs Thinking Chain (CoT, Chain of Thought) solve these tasks by adopting a reasoning strategy that uses a hint mechanism that includes intermediate reasoning steps to arrive at the final answer. It is speculated that this ability may be acquired through training on the code.

These emergent capabilities allow LLMs to excel in a variety of tasks, making them powerful tools for solving complex problems and applying them in multiple fields.

2.1.2 Ability to support multiple applications as a base model

In 2021, researchers from Stanford University and other universities proposed the concept of foundation model, clarifying the role of pre-trained models. This is a new AI technology paradigm that uses training on massive amounts of unlabeled data to obtain large models (single or multimodal) that can be applied to a large number of downstream tasks. In this way, **multiple applications can rely on only one or a few large models for unified construction** .

The large language model is a typical example of this new model. Using a unified large model can greatly improve R&D efficiency. Compared with the way of developing a single model each time, this is an essential improvement. Large models can not only shorten the development cycle of each specific application and reduce the required manpower investment, but also achieve better application results based on the reasoning, common sense and writing ability of large models. Therefore, large models can become a unified base model for AI application development. This is a new paradigm that achieves multiple goals at one stroke and deserves to be vigorously promoted.

2.1.3 Supporting the ability to use dialogue as a unified entry point

The opportunity that made the large language model really popular was **ChatGPT** , which is based on conversational chat. The industry has long discovered users' special preference for

conversational interaction. When Lu Qi was at Microsoft, he promoted the strategy of "conversation as a platform" in 2016. In addition, products based on voice conversations such as Apple Siri and Amazon Echo are also very popular, reflecting the preference of Internet users for chat and conversation as interaction modes. Although there were various problems with previous chatbots, the emergence of large language models has once again allowed chatbots as an interaction mode to re-emerge. Users are increasingly looking forward to artificial intelligence like "Jarvis" in Iron Man, who is omnipotent and omniscient. This has triggered our Agent (Agent) thinking about the prospects of type applications. Projects such as Auto-GPT and Microsoft Jarvis have already appeared and received attention. I believe that many similar projects will emerge in the future to allow assistants to complete various specific tasks in the form of conversations.

2.2 Characteristics of LLM

Large language models have several notable features that have made them a popular area of interest and research in natural language processing and other fields. Here are some of the main features of large language models:

1. **Huge scale:** LLMs usually have huge parameter scale, which can reach billions or even hundreds of billions of parameters. This allows them to capture more linguistic knowledge and complex grammatical structures.

2. **Pre-training and fine-tuning:** LLM adopts a pre-training and fine-tuning learning method. First, it is pre-trained on large-scale text data (unlabeled data) to learn general language representation and knowledge. Then it is adapted to specific tasks through fine-tuning (labeled data), so that it performs well in various NLP tasks.

3. **Context-aware:** LLMs have strong context-awareness when processing text, and are able to understand and generate text content that depends on previous text. This makes them excellent in conversation, article generation, and situational understanding.

4. **Multi-language support:** LLMs can be used in multiple languages, not just English. Their multi-lingual capabilities make cross-cultural and cross-linguistic applications easier.

5. **Multimodal support:** Some LLMs have been extended to support multimodal data, including text, images, and sound, allowing them to understand and generate content of different media types and achieve more diverse applications.

6. **Ethical and risk issues:** Although LLMs have excellent capabilities, they also raise ethical and risk issues, including the generation of harmful content, privacy issues, cognitive biases, etc. Therefore, research and application of LLMs require caution.

7. **High computing resource requirements:** LLM parameters are large in scale and require a lot of computing resources for training and reasoning. Usually, high-performance GPU or TPU clusters are required to implement it.

Large language models are a technology with powerful language processing capabilities that have demonstrated potential in many fields. They provide powerful tools for natural language understanding and generation tasks, but also raise concerns about their ethical and risk issues. These characteristics make LLM an important research and application direction in computer science and artificial intelligence today.

III. Application and Impact of LLM

LLM has had a profound impact in many fields. In the field of **natural language processing** , it can help computers better understand and generate text, including writing articles, answering questions, translating languages, etc. In the field of **information retrieval** , it can improve search engines and make it easier for us to find the information we need. In the field of **computer vision** , researchers are also working to make computers understand images and text to improve multimedia interactions.

Most importantly, the emergence of LLM has made people rethink the possibility of **artificial general intelligence (AGI)** . AGI is artificial intelligence that thinks and learns like humans. LLM is considered an early form of AGI, which has triggered many thoughts and plans for the future development of artificial intelligence.

In summary, LLM is an exciting technology that allows computers to better understand and use language, is changing the way we interact with technology, and is also triggering endless exploration of the future of artificial intelligence.

> **In the next chapter we will introduce RAG, an important technology in the LLM period.**

【Reference content】：

1. A Survey of Large Language Models
2. Zhou Feng: When we talk about big models, what new capabilities should we focus on?

Next Chapter >

2. Introduction to Retrieval Enhancement Generation RAG

Retrieval Enhancement Generation RAG Introduction

1. What is RAG

Large language models (LLMs) are more powerful than traditional language models, but in some cases they may still not provide accurate answers. In order to address a series of challenges faced by large language models when generating text and improve the performance and output quality of the models, researchers proposed a new model architecture: **Retrieval-Augmented Generation (RAG)** . This architecture cleverly **integrates relevant information retrieved from a huge knowledge base, and based on this, guides large language models to generate more accurate answers** , thereby significantly improving the accuracy and depth of the answers.

The main problems currently faced by LLM are:

- **Information bias/illusion:** LLM sometimes generates information that is inconsistent with objective facts, resulting in inaccurate information received by users. RAG assists the model generation process by retrieving data sources to ensure the accuracy and credibility of the output content and reduce information bias.
- **Knowledge update lag:** LLM is trained based on static data sets, which may cause the model's knowledge update to lag and fail to reflect the latest information dynamics in a timely manner. RAG retrieves the latest data in real time to maintain the timeliness of the content and ensure the continuous updating and accuracy of information.
- **Content is not traceable:** LLM-generated content often lacks a clear source of information, which affects the credibility of the content. RAG links the generated content with the retrieved original materials, enhancing the traceability of the content and thus increasing users' trust in the generated content.
- **Lack of domain expertise:** LLM may not be very effective in handling domain-specific expertise, which may affect the quality of its answers in related fields. RAG retrieves relevant documents in a specific field and provides the model with rich contextual information, thereby improving the quality and depth of answering questions in the domain of expertise.

- **Reasoning ability limitations:** When faced with complex questions, LLM may lack the necessary reasoning ability, which affects its understanding and answering of the questions. RAG combines the retrieved information with the generative ability of the model, and enhances the model's reasoning and understanding ability by providing additional background knowledge and data support.

- **Limited adaptability to application scenarios:** LLM needs to be efficient and accurate in a variety of application scenarios, but a single model may not be able to fully adapt to all scenarios. RAG enables LLM to flexibly adapt to various application scenarios such as question-answering systems and recommendation systems by retrieving data for the corresponding application scenarios.

- **Weak ability to process long texts:** LLM is limited to a limited context window when understanding and generating long content, and must process content sequentially. The longer the input, the slower the speed. RAG strengthens the model's understanding and generation of long contexts by retrieving and integrating long text information, effectively breaking through the limitation of input length, while reducing call costs and improving overall processing efficiency.

2. RAG Workflow

RAG is a complete system whose workflow can be simply divided into four stages: data processing, retrieval, enhancement, and generation:

1. **Data processing stage**
    1. Clean and process the raw data.
    2. Convert the processed data into a format that can be used by the retrieval model.
    3. The processed data is stored in the corresponding database.
2. **Retrieval stage**
    1. The user's question is input into the retrieval system and relevant information is retrieved from the database.
3. **Enhancement Phase**
    1. The retrieved information is processed and enhanced so that it can be better understood and used by the generative model.
4. **Generation phase**
    1. The augmented information is fed into the generative model, which generates answers based on the information.

3. RAG VS Finetune

RAG and fine-tuning (Finetune) are two mainstream methods for improving the performance of large language models.

Fine-tuning : Further training a large language model on a specific dataset to improve the model's performance on a specific task.

The comparison between RAG and fine-tuning can be found in the following table (source: [1][2])

Feature Comparison	RAG	Fine-tuning
Knowledge Update	Directly update the retrieval knowledge base without retraining. The cost of information update is low and suitable for dynamically changing data.	Retraining is usually required to keep knowledge and data updated. The update cost is high and it is suitable for static data.
External knowledge	Skilled in leveraging external resources, particularly suited to working with documents or other structured/unstructured databases.	Learn external knowledge into LLM.
data processing	The requirements for data processing and operation are extremely low.	Relying on building a high-quality dataset, a limited dataset may not significantly improve the performance.
Model customization	Focuses on information retrieval and incorporating external knowledge, but may not adequately customize model behavior or writing style.	LLM behavior, writing style, or specific domain knowledge can be tailored to a particular style or terminology.
Explainability	It can be traced back to the specific data source and has good explainability and traceability.	Black box, relatively low interpretability.
Computing resources	Additional resources are required to support the search mechanism and maintenance of the database.	Relying on high-quality training datasets and fine-tuning targets, it places high demands on computing resources.
Inference Latency	Increased the time consumption of the retrieval step	Time consumption for simple LLM generation

Feature Comparison	RAG	Fine-tuning
Reduce hallucinations	The answers are generated by retrieving real information, which reduces the probability of hallucination.	Models that learn from domain-specific data can help reduce hallucinations, but they may still hallucinate when faced with unseen inputs.
Ethical Privacy	Retrieving and using external data may raise ethical and privacy issues.	Sensitive information in training data needs to be properly handled to prevent leakage.

IV. RAG's Success Stories

RAG has achieved success in many fields, including question-answering systems, dialogue systems, document summarization, document generation, etc.

We will introduce the application of RAG in detail in the third part. We will disassemble the existing mature RAG cases and have a deeper understanding of RAG with you.

1. **The Datawhale Knowledge Base Assistant** is a combination of the content of this course. It is based on **ChatWithDatawhale** , a Datawhale content learning assistant created by **Sanbu** , and adjusts the architecture to the LangChain architecture that is easy for beginners to learn. It is an LLM application that encapsulates large model APIs from different sources with reference to the content of Chapter 2. It can help users communicate smoothly with DataWhale's existing warehouses and learning content, thereby helping users quickly find the content they want to learn and the content they can contribute.
2. **Tianji** is a free, non-commercial artificial intelligence system developed by **SocialAI** . You can use it to perform tasks involving traditional social skills, such as how to toast, how to say nice things, how to be good at dealing with people, etc., to improve your emotional intelligence and core competitiveness. We firmly believe that only social skills are the core technology of future AI, and only AI that is good at dealing with people has the opportunity to move towards AGI. Let us work together to witness the advent of general artificial intelligence. —— "Tianji cannot be leaked."

> **In this chapter, we have a brief understanding of RAG. In the next chapter, we will introduce a commonly used RAG development framework LangChain.**

【Reference content】：

1. Retrieval-Augmented Generation for Large Language Models: A Survey
2. Retrieval-augmented generation techniques for large language models: A review

Previous chapter
1. Introduction to Large Language Model (LLM) Theory

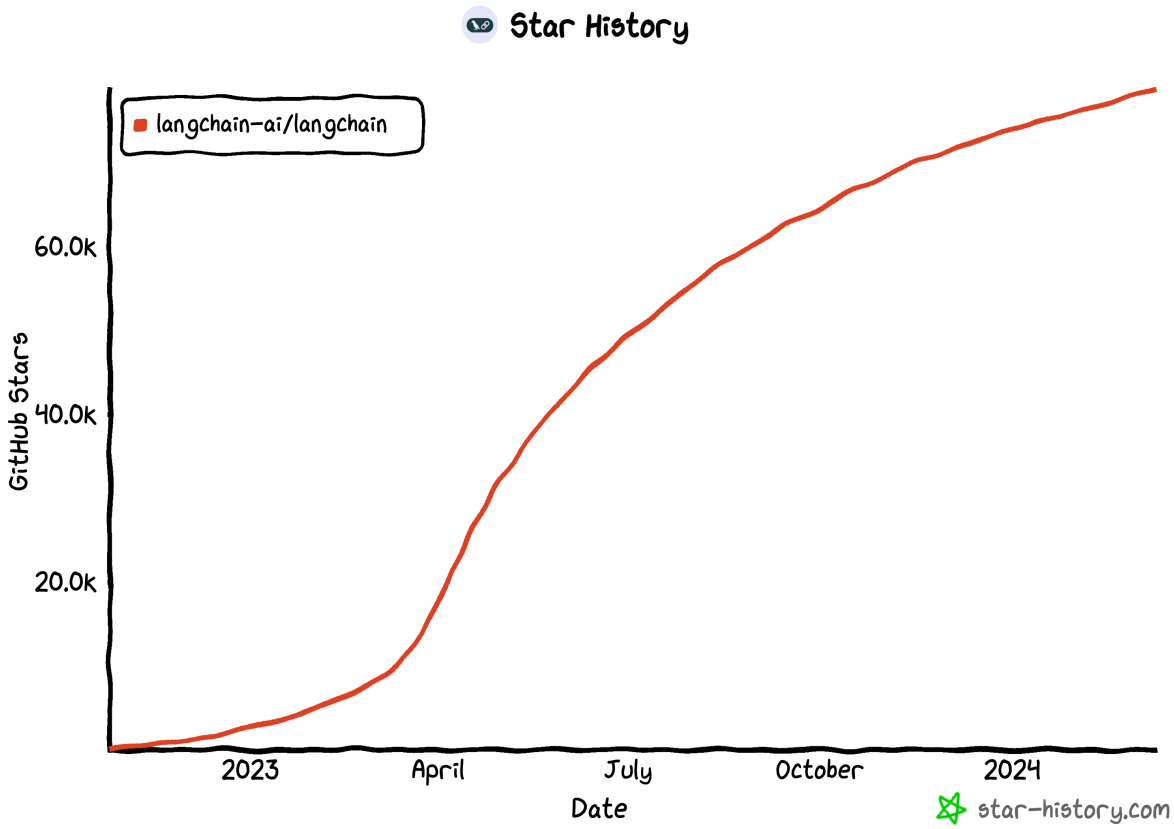
Next Chapter
3. Introduction to LangChain

LangChain

1. What is LangChain

The huge success of ChatGPT has inspired more and more developers to develop applications based on large language models using the API or private models provided by OpenAI. Although the call of large language models is relatively simple, creating a complete application still requires a lot of custom development work, including API integration, interaction logic, data storage, and so on.

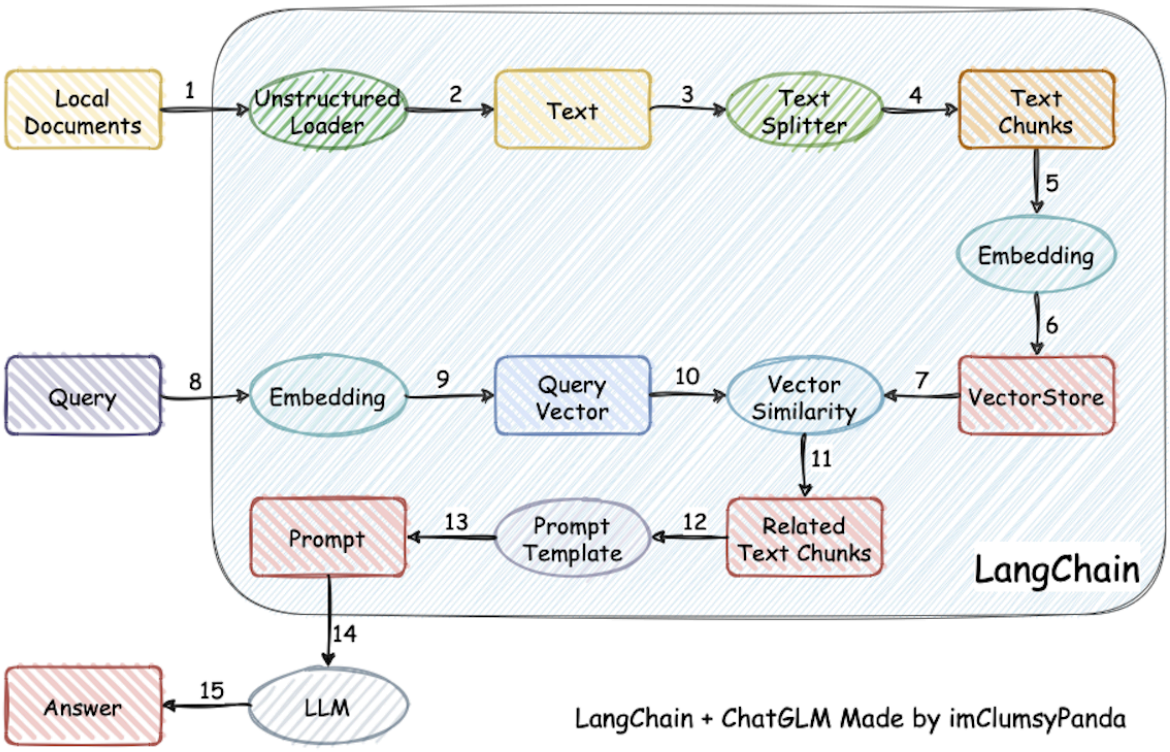
To solve this problem, many institutions and individuals have launched a number of open source projects since 2022 to **help developers quickly build end-to-end applications or workflows based on large language models** . One of the most popular projects is the LangChain framework.



The LangChain framework is an open source tool that leverages the power of large language models to develop a variety of downstream applications. Its goal is to provide a common interface for a variety of large language model applications, thereby simplifying the

application development process . Specifically, the LangChain framework enables data awareness and environmental interaction, that is, it enables language models to connect with other data sources and allows language models to interact with their environment.

Using the LangChain framework, we can easily build a RAG application as shown below (**Image source**). In the figure below, Each oval represents a module of LangChain , such as a data collection module or a preprocessing module. Each rectangle represents a data state , such as raw data or preprocessed data. The arrows indicate the direction of data flow, from one module to another. At each step, LangChain can provide corresponding solutions to help us handle various tasks.



2. Core Components of LangChain

As a large language model development framework, LangChain can integrate LLM models (dialogue models, embedding models, etc.), vector databases, interactive layer prompts, external knowledge, and external agent tools, and can freely build LLM applications. LangChain mainly consists of the following 6 core components:

- **Model I/O** : Interface for interacting with language models
- **Data connection** : An interface for interacting with application-specific data.
- **Chains** : Combine components to achieve end-to-end applications. For example, we will build it Search question and answer chainto complete search question and answer later.
- **Memory** : used to persist application state between multiple runs of the chain;

- **Agents** : Extend the reasoning capabilities of the model. Used for complex application call sequences;
- **Callbacks** : Extend the reasoning capabilities of the model. Used for complex application call sequences;

During the development process, we can flexibly combine according to our own needs.

3. Stable version of LangChain

In the rapid development of LLM technology, LangChain, as an evolving innovation platform, continues to push the boundaries of technology. `2024 年 1 月 9 日` LangChain officially released its stable version **v0.1.0** . This milestone update brings comprehensive and powerful functional support to developers. It covers key components such as model input and output processing, data connection, chain operation, memory mechanism, proxy service, and callback processing, providing a solid foundation for the development and deployment of LLM applications. At the same time, LangChain's continuous optimization and functional iteration will bring more innovative features and performance improvements in the future.

- **Compatibility and support** : LangChain v0.1.0 takes into account `Python and JavaScript` the support for while maintaining backward compatibility, ensuring that developers can seamlessly transition during the upgrade process and enjoy a more secure and stable development experience.

- **Architecture Improvement** : By e ectively separating the core component langchain-core from the partner package, LangChain's architecture design has become more organized and stable, laying a solid foundation for future systematic expansion and security improvement.

- **Observability** : LangChain provides industry-leading debugging and observation capabilities through deep integration with LangSmith. This enables developers to have a clear understanding of each operation and its input and output in the LLM application, greatly simplifying the debugging and troubleshooting process.

- **Wide range of integrations** : LangChain has nearly **700** integrations, covering multiple technical areas from LLM to vector storage, tools, and agents, greatly reducing the complexity of building LLM applications on various technology stacks.

- **Composability** : With `LangChain Expression Language (LCEL)` , developers can easily build and customize chains and take full advantage of the data orchestration framework, including advanced features such as batch processing, parallel operations, and alternatives.

- **Streaming processing** : LangChain has deeply optimized streaming processing to ensure that all chains created using LCEL can support streaming processing, including data streaming transmission in intermediate steps, thereby providing users with a smoother experience.

- **Output parsing** : LangChain provides a series of powerful output parsing tools to ensure that LLM can return information in a structured format, which is crucial for LLM to execute specific action plans.

- **Retrieval capabilities** : LangChain introduces advanced retrieval technologies suitable for production environments, including text segmentation, retrieval mechanisms, and indexing pipelines, allowing developers to easily combine private data with the capabilities of LLM.

- **Tool Usage and Agents** : LangChain provides a rich collection of agents and tools, and provides a simple way to define tools. It supports agent workloads, including allowing LLM to call functions or tools, and how to efficiently make multiple calls and inferences, greatly improving development efficiency and application performance.

4. LangChain Ecosystem

- **LangChain Community** : Focusing on third-party integration, it greatly enriches the LangChain ecosystem, making it easier for developers to build complex and powerful applications, while also promoting community cooperation and sharing.

- **LangChain Core** : The core library and core component of the LangChain framework, which provides basic abstractions and the LangChain Expression Language (LCEL), and provides infrastructure and tools for building, running, and interacting with LLM applications, providing a solid foundation for the development of LangChain applications. The document processing, formatting prompt, output parsing, etc. that we will use later all come from this library.

- **LangChain CLI** : A command line tool that enables developers to interact with the LangChain framework through the terminal to perform tasks such as project initialization, testing, and deployment. It improves development efficiency and allows developers to manage the entire application lifecycle through simple commands.

- **LangServe** : A deployment service for deploying LangChain applications to the cloud, providing a scalable, highly available hosting solution with monitoring and logging capabilities. It simplifies the deployment process so that developers can focus on application development without having to worry about the underlying infrastructure and operations.

- **LangSmith** : A developer platform that focuses on the development, debugging, and testing of LangChain applications, providing a visual interface and performance analysis tools

designed to help developers improve the quality of their applications and ensure that they meet the expected performance and stability standards before deployment.

> **In this chapter, we briefly introduced the development framework LangChain. In the next chapter, we will introduce the overall process of developing LLM applications.**

Previous chapter
2. Introduction to Retrieval Enhancement Generation RAG

Next Chapter
4. Overall process of developing LLM application

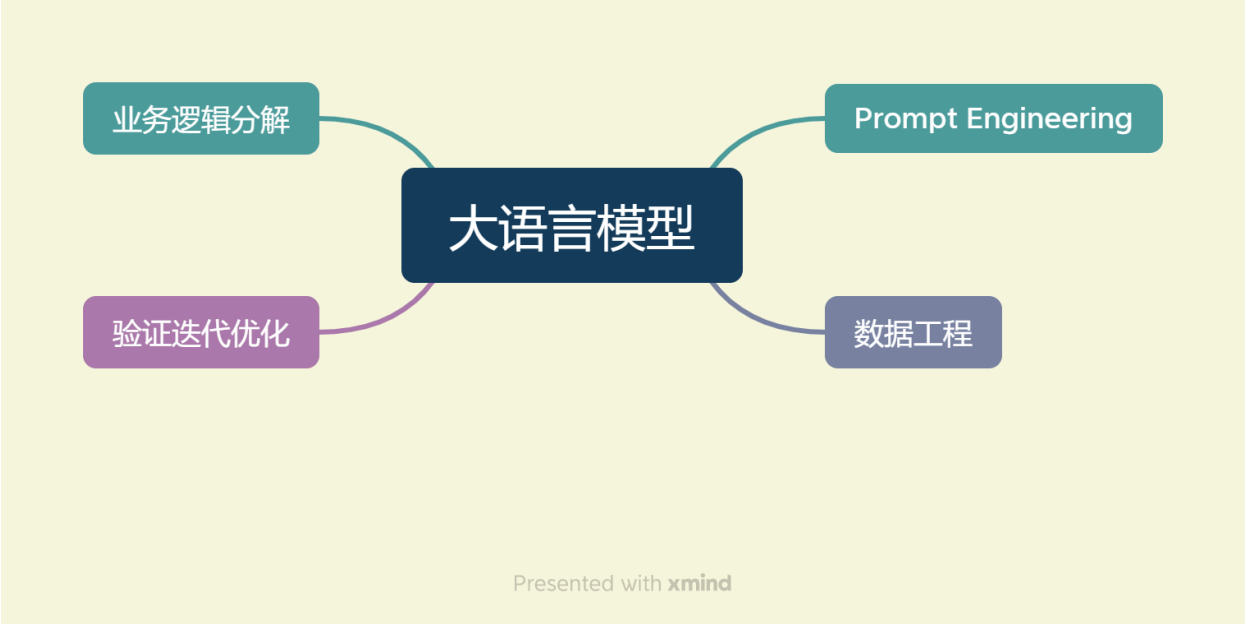
3316 words | 9 minutes

The overall process of developing LLM applications

1. What is large model development?

We call **the development of applications that use big language models as the core functions, use the powerful understanding and generation capabilities of big language models, and combine special data or business logic to provide unique functions** as big model development . Although the core technology of developing big model-related applications is on big language models, they are generally achieved by calling APIs or open source models to achieve core understanding and generation, and by prompt engineering to achieve control of big language models. Therefore, although big models are the culmination of deep learning, big model development is more of an **engineering problem** .

In the development of large models, we generally do not make major changes to the model, but **use the large model as a calling tool, and use prompt engineering, data engineering, business logic decomposition and other means to give full play to the capabilities of the large model and adapt to application tasks** , rather than focusing on optimizing the model itself. Therefore, as beginners in the development of large models, we do not need to deeply study the internal principles of the large model, but rather need to master the practical skills of using the large model.



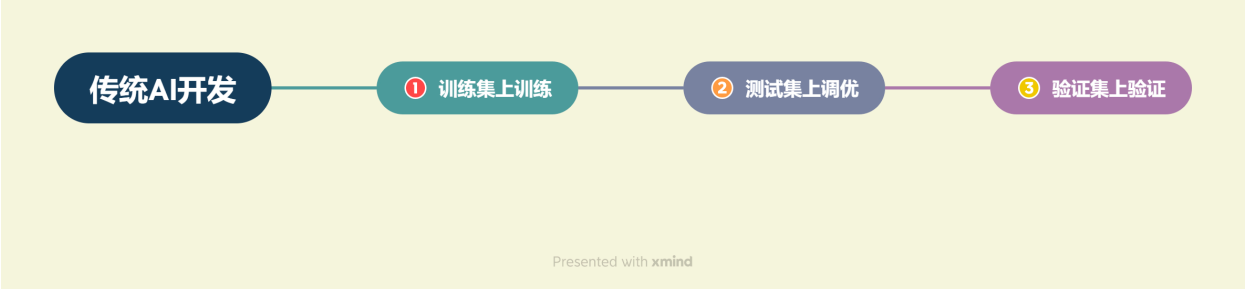
Large Model Development Elements

At the same time, the overall idea of large model development, which focuses on calling and utilizing large models , is quite different from that of traditional AI development . The two core capabilities of large language models are: Instructions to follow providing Text Generation a simple alternative to complex business logic.

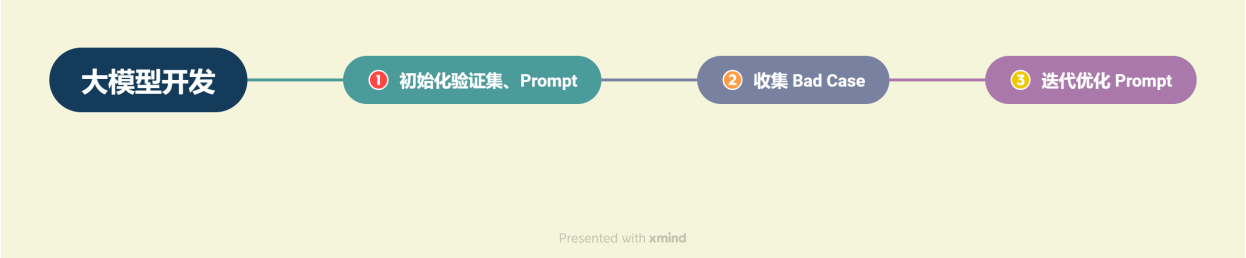
- Traditional AI Development : First, you need to disassemble the very complex business logic one by one, construct training data and verification data for each sub-business, train and optimize the model for each sub-business, and finally form a complete model chain to solve the entire business logic.
- Large model development : Use Prompt Engineering to replace sub-model training and tuning, implement business logic through Prompt link combination, use a general large model + several business prompts to solve the task, thereby transforming traditional model training and tuning into a simpler, easier and lower-cost Prompt design and tuning.

At the same time, in terms of **evaluation ideas** , there are qualitative differences between large model development and traditional AI development.

- Traditional AI Development : You need to first construct a training set, a test set, and a validation set, and then evaluate the performance by training the model on the training set, fine-tuning the model on the test set, and finally verifying the model effect on the validation set.
- Large model development: The process is more flexible and agile. Construct a small batch validation set based on actual business needs, and design a reasonable prompt to meet the validation set effect. Then, continuously collect the bad cases of the current prompt from the business logic, add the bad cases to the validation set, optimize the prompt in a targeted manner, and finally achieve a better generalization effect.



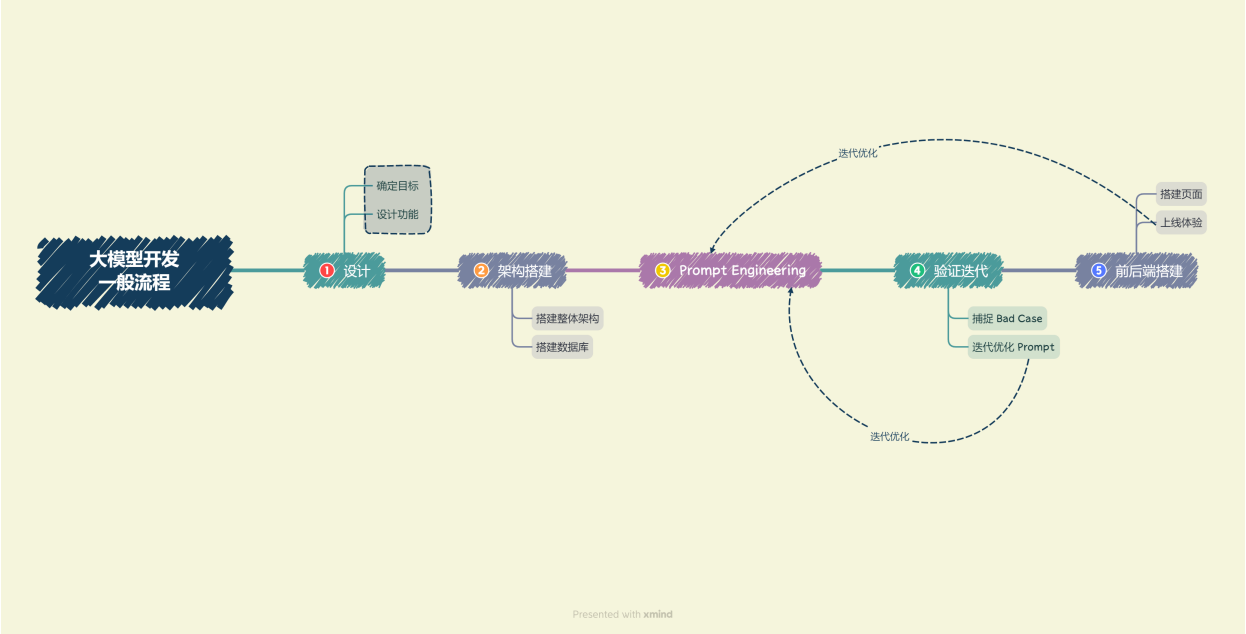
Traditional AI Evaluation



In this chapter, we will briefly describe the general process of large model development, and combine it with the actual needs of the project to gradually analyze the work and steps to complete the project development.

2. General process of large model development

Combined with the above analysis, we can generally decompose large model development into the following processes:



1. **Determine the goal** . Before development, we first need to determine the development goal, that is, the application scenario, target population, and core value of the application to be developed. For individual developers or small development teams, they should generally set a minimum goal first, starting with building an MVP (minimum viable product), and gradually improve and optimize.

2. **Design functions** . After determining the development goals, it is necessary to design the functions that the application will provide, as well as the general implementation logic of each function. Although we have simplified the decomposition of business logic by using large models, clearer and deeper understanding of business logic can often lead to better Prompt effects. Similarly, for individual developers or small development teams, the first thing to do is to determine the core functions of the application, and then extend the design of upstream and downstream functions of the core functions; for example, if we want to create a personal knowledge base assistant, then the core function is to answer questions based on the content of the personal knowledge base, then the upstream function of users

uploading knowledge bases and the downstream function of users manually correcting model answers are sub-functions that we must also design and implement.

3. **Build the overall architecture** . At present, most large model applications use the architecture of specific database + prompt + general large model. We need to build the overall architecture of the project for the functions we designed, and realize the whole process from user input to application output. Generally speaking, we recommend development based on the LangChain framework. LangChain provides the implementation of Chain, Tool and other architectures. We can customize it based on LangChain to realize the overall architecture connection from user input to database to large model and final output.

4. **Build a database** . Personalized large model applications need to be supported by a personalized database. Since large model applications require vector semantic retrieval, vector databases such as Chroma are generally used. In this step, we need to collect data and preprocess it, and then vectorize and store it in the database. Data preprocessing generally includes conversion from multiple formats to plain text, such as PDF, MarkDown, HTML, audio and video, as well as cleaning of erroneous data, abnormal data, and dirty data. After preprocessing, it is necessary to slice and vectorize to build a personalized database.

5. **Prompt Engineering** . High-quality prompts have a great impact on the capabilities of large models. We need to gradually and iteratively build high-quality prompt engineering to improve application performance. In this step, we should first clarify the general principles and techniques of prompt design, build a small validation set from actual business, and design a prompt based on the small validation set that meets basic requirements and has basic capabilities.

6. **Verification iteration** . Verification iteration is an extremely important step in large model development. It generally refers to improving system effects and dealing with boundary conditions by constantly discovering bad cases and improving prompt engineering in a targeted manner. After completing the initial prompt design in the previous step, we should conduct actual business tests, explore boundary conditions, find bad cases, and analyze the problems of prompts in a targeted manner, so as to continuously iterate and optimize until a relatively stable prompt version that can basically achieve the goal is reached.

7. **Build the front-end and back-end** . After completing Prompt Engineering and its iterative optimization, we have completed the core functions of the application and can give full play to the powerful capabilities of the large language model. Next, we need to build the front-end and back-end, design the product page, and make our application go online as a product. Front-end and back-end development is a very classic and mature field, so I will not go into details here. We use Gradio and Streamlit to help individual developers quickly build visual pages to realize the Demo online.

8. **Experience optimization** . After completing the front-end and back-end construction, the application can be put online for experience. Next, it is necessary to conduct long-term user experience tracking, record bad cases and user negative feedback, and then make targeted optimizations.

3. Brief analysis of the process of building an LLM project (taking the knowledge base assistant as an example)

Below we will combine this practical project with the overall process introduction above to briefly analyze **the knowledge base assistant project** development process:

Step 1: Project planning and demand analysis

1. Project goal : Question-answering assistant based on personal knowledge base

2. Core Functions

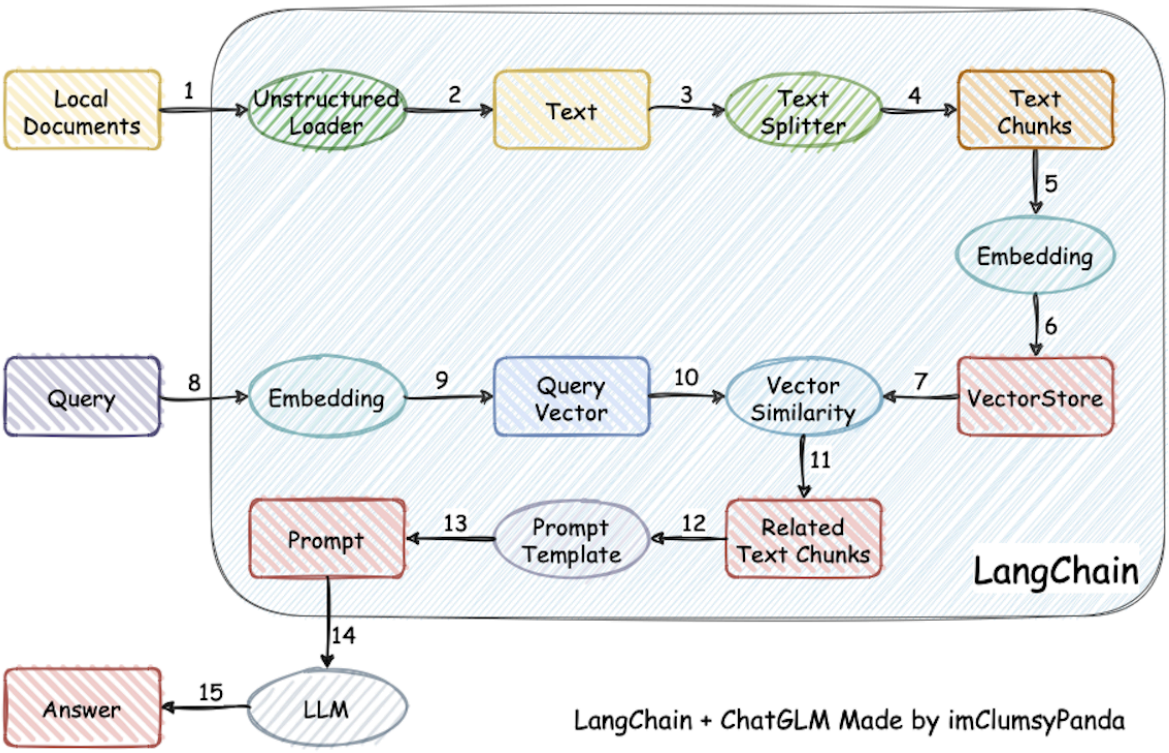
1. Vectorize the crawled and summarized MarkDown files and user-uploaded documents, and create a knowledge base;
2. Select a knowledge base and retrieve the knowledge fragments asked by the user;
3. Provide knowledge snippets and questions to get answers from the big model;
4. Streaming reply;
5. Historical conversation records

3. Determine technical architecture and tools

1. **Framework** : LangChain
2. **Embedding models** : GPT, Zhipu, **M3E**
3. **Database** : Chroma
4. **Large models** : GPT, iFlytek Spark, Wenxin Yiyan, GLM, etc.
5. **Front and back end** : Gradio and Streamlit

Step 2: Data preparation and vector knowledge base construction

The implementation principle of this project is shown in the figure below (image source): load local document -> read text -> text segmentation -> text vectorization -> question vectorization -> match the top k most similar to the question vector in the text vector -> add the matched text as context and question to the prompt -> submit to LLM to generate an answer.



1. Collect and organize documents provided by users

Common document formats used by users include PDF, TXT, MD, etc. First, we can use LangChain's document loader module to easily load documents provided by users, or use some mature Python packages to read them.

Due to the limitation of using tokens in current large models, we need to segment the read text and divide longer text into smaller texts. At this time, a piece of text is a unit of knowledge.

2. Vectorize document words

Text Embeddings Technology The segmented documents are vectorized so that semantically similar text segments have close vector representations. Then, they are stored in the vector database to complete the `(index)` creation of .

By using the vector database to index each document fragment, fast retrieval can be achieved.

3. Import the vectorized documents into the Chroma knowledge base and create a knowledge base index

Langchain integrates with over 30 different vector databases. The Chroma database is lightweight and the data is stored in memory, which makes it very easy to launch and start using.

The user's knowledge base content is stored in the vector database through Embedding, and then every time the user asks a question, it will also go through Embedding. The vector correlation algorithm (such as the cosine algorithm) is used to find the most matching knowledge base fragments. These knowledge base fragments are used as context and submitted to the LLM as a prompt together with the user's question for answer.

Step 3: Large model integration and API connection

1. Integrate large models such as GPT, Spark, Wenxin, GLM, etc., and configure API connections.
2. Write code to interact with the big model API to get answers to your questions.

Step 4: Core Function Implementation

1. Build Prompt Engineering to implement large-model answering capabilities and generate answers based on user questions and knowledge base content.
2. Implement streaming replies, allowing users to have multiple rounds of conversations.
3. Add the historical conversation record function to save the interaction history between the user and the assistant.

Step 5: Iterative optimization of core functions

1. Conduct verification and evaluation and collect Bad Cases.
2. Iterate and optimize the core function implementation according to Bad Case.

Step 6: Front-end and user interface development

1. Use Gradio and Streamlit to build the front-end interface.
2. Realize the function of users uploading documents and creating knowledge base.
3. Design the user interface, including question input, knowledge base selection, history display, etc.

Step 7: Deployment, testing and launch

1. Deploy the Q&A Assistant to a server or cloud platform and ensure that it is accessible on the Internet.
2. Conduct production environment testing to ensure system stability.

3. Go online and release to users.

Step 8: Maintenance and Continuous Improvement

1. Monitor system performance and user feedback, and handle issues in a timely manner.
2. The knowledge base is updated regularly with new documents and information.
3. Collect user needs and make system improvements and function expansions.

The entire process will ensure that the project can proceed smoothly from planning, development, testing to launch and maintenance, providing users with high-quality question-and-answer assistants based on personal knowledge bases.

> **Now that we have a preliminary understanding of the general process of large model development, we will introduce the entire development environment to ensure that everyone can carry out project development smoothly.**
>
> - If you are an old hand, you can skip the subsequent content of this chapter and go directly to the second part.
> - The next two chapters mainly introduce the construction of two development environments for students who do not have a suitable development environment. You can read them as needed.
>   - Chapter 5 mainly introduces Basic usage of Alibaba Cloud Server, remote connection to the server via SSH and Use of Jupyter Notebook.
>   - Chapter 6 mainly introduces `GitHub CodeSpace` the use of GitHub and how to `GitHub CodeSpace` build a development environment in GitHub. (First, make sure you have a network environment that can smoothly access GitHub. Otherwise, it is recommended to use Alibaba Cloud.)
> - If you already have a suitable development machine, you can jump directly to `7.Environment Configuration` the chapter and start configuring the development environment.

Previous chapter
3. Introduction to LangChain

5. Alibaba Cloud Server Usage Guide

Basic usage of Alibaba Cloud Server

1. Why choose Alibaba Cloud Server?

Alibaba Cloud is a leading global cloud computing service provider, providing cloud computing, big data, artificial intelligence, security, enterprise applications, digital entertainment and other services to millions of customers in more than 200 countries and regions around the world. Alibaba Cloud's servers are stable in performance and low in price, making them the first choice for many beginners. In particular, Alibaba Cloud's university plan allows users to receive cloud servers for free, which is very suitable for students. For new users, Alibaba Cloud also provides a free trial opportunity, allowing users to use cloud servers for free for one year.

2. Obtaining the Alibaba Cloud University Plan

2.1. Introduction to the University Program

The general benefits for college students are open to all Chinese college students, including students enrolled in colleges and universities at the junior college, undergraduate, master's, doctoral, and postgraduate levels in mainland China, Hong Kong, Macao and Taiwan. On this basis, students from Alibaba Cloud partner universities can enjoy an additional 30% off exclusive benefits.

2.2. Application for University Student Program

Application link: **https://university.aliyun.com/mobile?clubTaskBiz=subTask..11337012..10212..&userCode=1h9ofupt**

1. Benefit 1: Chinese college students who have passed the student certification can receive a 300 yuan no-threshold coupon.

2. Benefit 2: Chinese university students who have passed the student certification and whose universities are Alibaba Cloud partner universities can receive a 30% discount on Alibaba Cloud

public cloud products (excluding special products) on top of a 300 yuan no-threshold coupon, with the original order price not exceeding 5,000 yuan.

The current partner universities include: Tsinghua University, Peking University, Zhejiang University, Shanghai Jiao Tong University, University of Science and Technology of China, South China University of Technology and Hong Kong University of Science and Technology (Guangzhou). More universities are being negotiated, so stay tuned!

Click on the Get It Now button in the picture, and scan the QR code with your real-name authenticated Alipay account to log in and verify.

2.3. Rights and interests of college students and teachers

The user is a teacher (including postdoctoral fellows) of the cooperating university and has completed identity authentication in accordance with the activity requirements.

The current partner universities include: Tsinghua University, Peking University, Zhejiang University, Shanghai Jiao Tong University, University of Science and Technology of China, South China University of Technology and Hong Kong University of Science and Technology (Guangzhou). More universities are being negotiated, so stay tuned!

- Alibaba Cloud offers an exclusive 50% discount on all public cloud products (excluding special products) and sets up exclusive service channels to accelerate scientific research and teaching.

Note: It is best to purchase the product only after confirming that the coupon has been received; 用户中心-卡券管理-优惠券管理 check the coupon information.

https://developer.aliyun.com/plan/student

3. New users can receive a trial of the cloud server

Application link: https://free.aliyun.com/?crowd=personal . I recommend you to apply here 云服务器 ECS . The free quota is 280 yuan per month and is valid for 3 months. The configuration is as follows:

- e series 2 cores 2GB or 2 cores 4GB (200 yuan free quota per month);
- 80 yuan free quota for public network traffic per month (can be used to deduct 100GB of domestic traffic)

4. Guidelines for Creating a Cloud Server

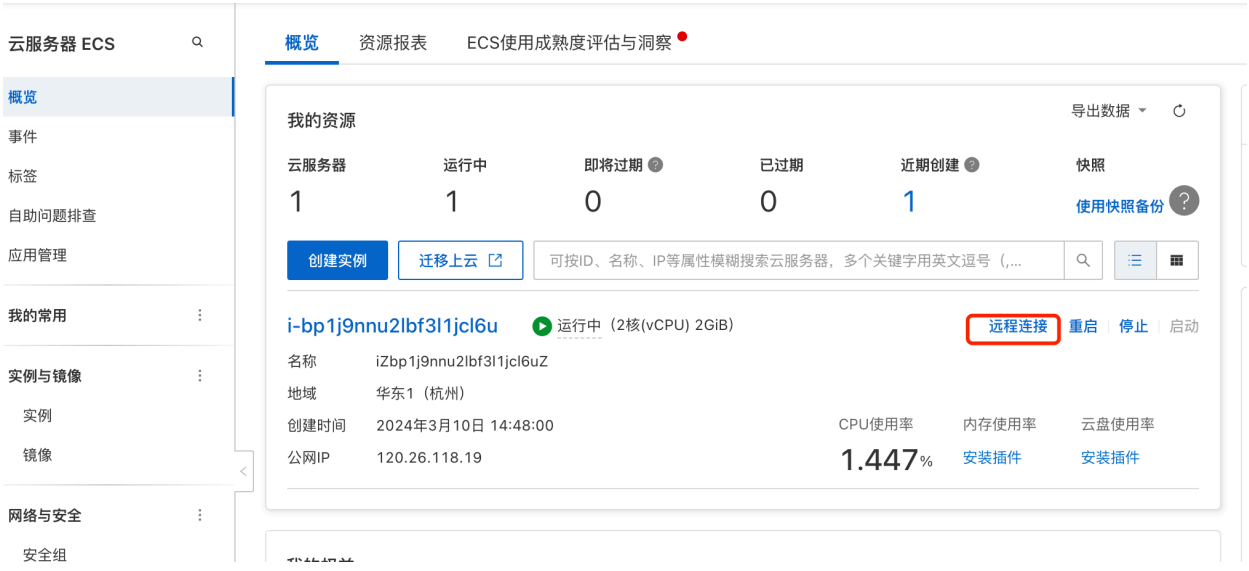
Here we 云服务器 ECS take selection as an example for configuration. Select the minimum configuration for system selection Ubuntu .

云服务器 ECS

During the trial, fill in "Automatically release the instance" in the expiration release setting, so

that no fees will be incurred after the expiration.

After the creation is complete, click Trial or Link to see the instance we just created.



Click Remote Connection and click Login Now.

The default password is root;

if you are not allowed to log in, follow **the link** to modify the server configuration.

[Simple solution: On the cloud server management console page, click the three dots near

'Remote Connection' and select Reset Instance Password. Log in again and you are done.]

After that, you can enter the environment to study! ! !

5. VSCode connects to a remote server

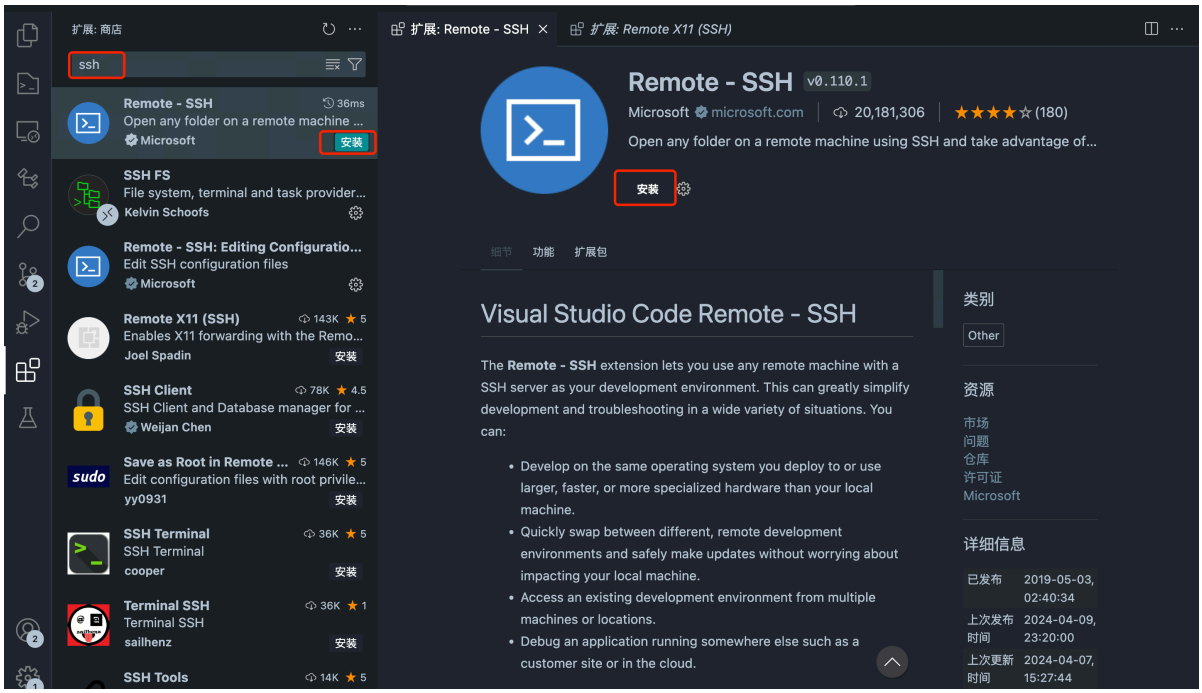
`Visual Studio Code (VSCode)` VSCode is a free, open source, modern code editor developed by Microsoft. It is favored by developers for its lightweight, high performance, and wide range of programming language support. The core features of VSCode include:

1. **Cross-platform** : Supports Windows, macOS, and Linux operating systems.
2. **Extension Market** : Provides a wide range of extension plug-ins to meet different development needs.
3. **Built-in Git support** : convenient for version control operations.
4. **Debugging tools** : built-in powerful debugging functions, supporting multiple programming languages.
5. **Smart Sense** : Provides intelligent prompt functions such as code completion and parameter information.
6. **Integrated Terminal** : Built-in terminal, you can perform command line operations without switching.
7. **Customization** : Supports personalized settings such as themes and key bindings.

VSCode's flexibility and ease of use make it one of the preferred code editing tools for developers.

Here we choose VSCode to connect to the remote server, which makes it convenient for us to operate the remote server directly in the local editor.

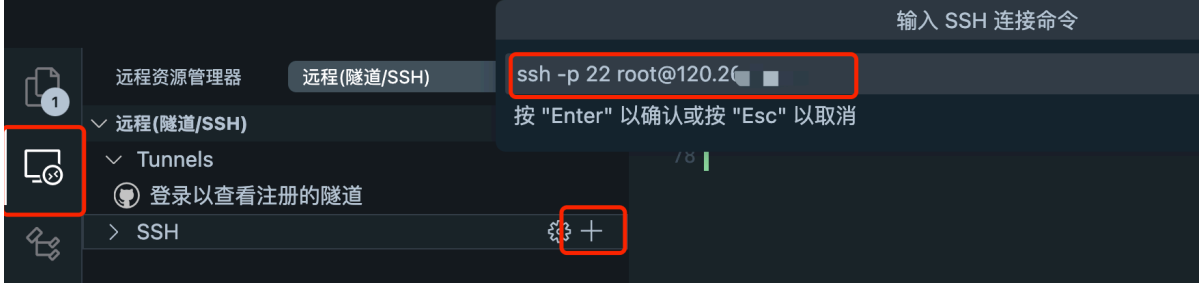
1. Install the SSH plug-in. Open the VSCODE plug-in market, search for SSH, find `Remote - SSH` the plug-in and install it.



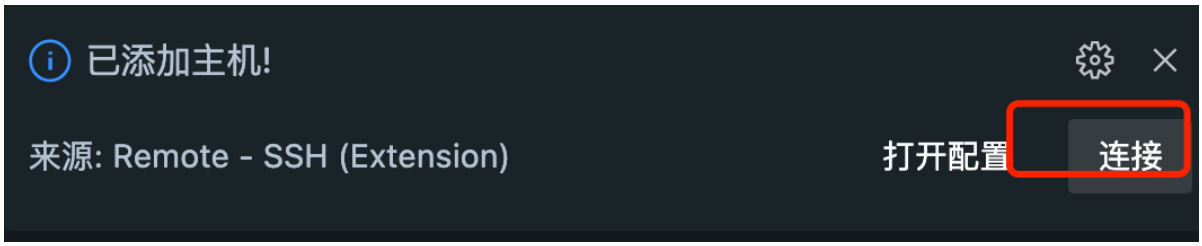
2. Get the server IP Open the instance list of Alibaba Cloud Server Find the public IP address of the server we need to connect to and copy it.

Open the editor of the remote server that can be connected, here we take VSCODE as an example.

3. Configure SSH. Open the plugin you just downloaded 远程资源管理器 and add the server's SSH. `ssh -p port username@ip` The port is usually set to 22. The username can be replaced with the server's IP address using root or a custom username. Select the local SSH configuration file.



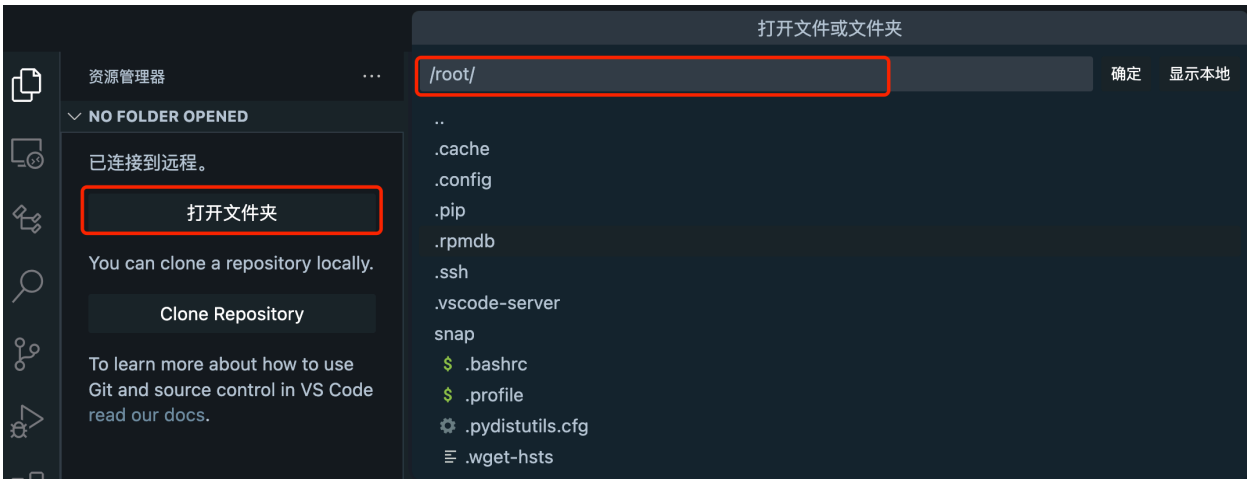
Click the link in the lower right corner to enter the server.



4. After connecting, when we are connected, we can continue to click on the left 远程资源管理器 to find our server, and there are two options on the right.

- The arrow is the window that opens
- The plus sign in the upper left corner means a new window is opened

5. After opening the directory, click Open Folder and enter the required directory to open it.



After that, you can start programming happily!!!

6. Use Jupyter Notebook

Jupyter Notebook is an open source library 交互式计算环境 that allows users to create and share documents containing live code, equations, visualizations, and text. Its name comes from the three core programming languages it supports: Julia, Python, and R, which is also the origin of the name "Ju-pyt-er". Files written in Jupyter Notebook have the suffix `.ipynb`

Key features of Jupyter Notebook include:

1. **Interactive Programming** : Users can write code in separate cells and execute it, Immediately see the results of running the code which is very useful for fields such as data analysis, machine learning, and scientific computing.

2. **Multi-language support** : Although originally designed for Julia, Python, and R, Jupyter now supports more than 40 programming languages by using corresponding kernels.

3. **Rich presentation capabilities** : Jupyter Notebook supports Markdown, allowing users to add rich media content such as formatted text, images, videos, HTML, LaTeX, etc., making documents more vivid and informative.

4. **Data visualization** : Jupyter Notebook is seamlessly integrated with many data visualization libraries (such as Matplotlib, Plotly, Bokeh, etc.), and you can generate charts and visualize data directly in Notebook.

5. **Easy to share** : Notebook files can be easily shared via email, cloud services, or Jupyter Notebook Viewer, allowing others to view the content, run the code, and even leave comments.

6. **Extensibility** : Jupyter has a large number of extension plug-ins that can enhance its functionality, such as interactive widgets, code auto-completion, theme replacement, etc.

7. **Integration of scientific computing tools** : Jupyter Notebook can be integrated with many scientific computing and data analysis tools, such as Python libraries such as NumPy, Pandas, and SciPy, making data processing and analysis more convenient.

Jupyter Notebook is a widely used tool for data scientists, researchers, educators, and students. It promotes the development of open science and education, making it easier to share and reproduce research results.

This tutorial uses Jupyter Notebook to write and run code, which makes it easier for us to write and debug code.

`VSCODE` Currently, you can open Jupyter Notebook files directly without installing any plugins. (You can also follow the next chapter to install and configure plugins)

A Notebook document consists of a series of cells, which are mainly in the following two forms.

- **Code cell** : Enter code in a code cell and press `Shift + Enter` to run the code in the cell and display the output result below.
- **Markdown cell** : Use `Markdown` the syntax to write text in the cell. You can create headings, lists, links, formatted text, etc., and use `Ctrl + Enter` to render the current Markdown cell.

We usually use code cells to write code and run it in time to view the results. And use the following shortcut keys to improve efficiency:

Cell Editing

- `Enter` : Enter edit mode.
- `Esc` : Exit edit mode.

Cell Operations

- `A` : Insert a new cell above the current cell.
- `B` : Insert a new cell below the current cell.
- `D` (Press twice): Delete the current cell.
- `Z` : Undo the delete operation.
- `C` : Copy the current cell.
- `V` : Paste previously copied cells.
- `X` : Cut the current cell.
- `Y` : Convert the current cell to a code cell.
- `M` : Convert the current cell to a Markdown cell.
- `Shift + M` : Toggle Markdown rendering of the cell.

Code execution and debugging

- `Shift + Enter` : Run the current cell and jump to the next cell.
- `Ctrl + Enter` : Run the current cell but do not jump to the next cell.
- `Alt + Enter` : Run the current cell and insert a new cell below it.
- `Esc` : Enter command mode.
- `Enter` : Enter edit mode.
- `Ctrl + Shift + -` : Split the current cell into two cells.
- `Ctrl + Shift + P` : Open the command palette, where you can search and execute various commands.

Navigation and window management

- `Up` / `Down` or `K` / `J` : Move up or down between cells.
- `Home` / `End` : Jump to the beginning or end of the Notebook.
- `Ctrl + Home` / `Ctrl + End` : Jump to the first or last cell of the current Notebook.

- `Tab` : Switch to the next panel in the Notebook view (for example, from the Editor to the Output or Metadata panel).
- `Shift + Tab` : Switch to the previous panel in Notebook view.

Other useful shortcuts

- `H` : Show or hide the Notebook sidebar.
- `M` : Convert the current cell to a Markdown cell.
- `Y` : Convert the current cell to a code cell.

> **Now that we have the necessary foundation for development, we can proceed directly `7. Environment Configuration` to environment configuration.**

Previous chapter
4. Overall process of developing LLM application

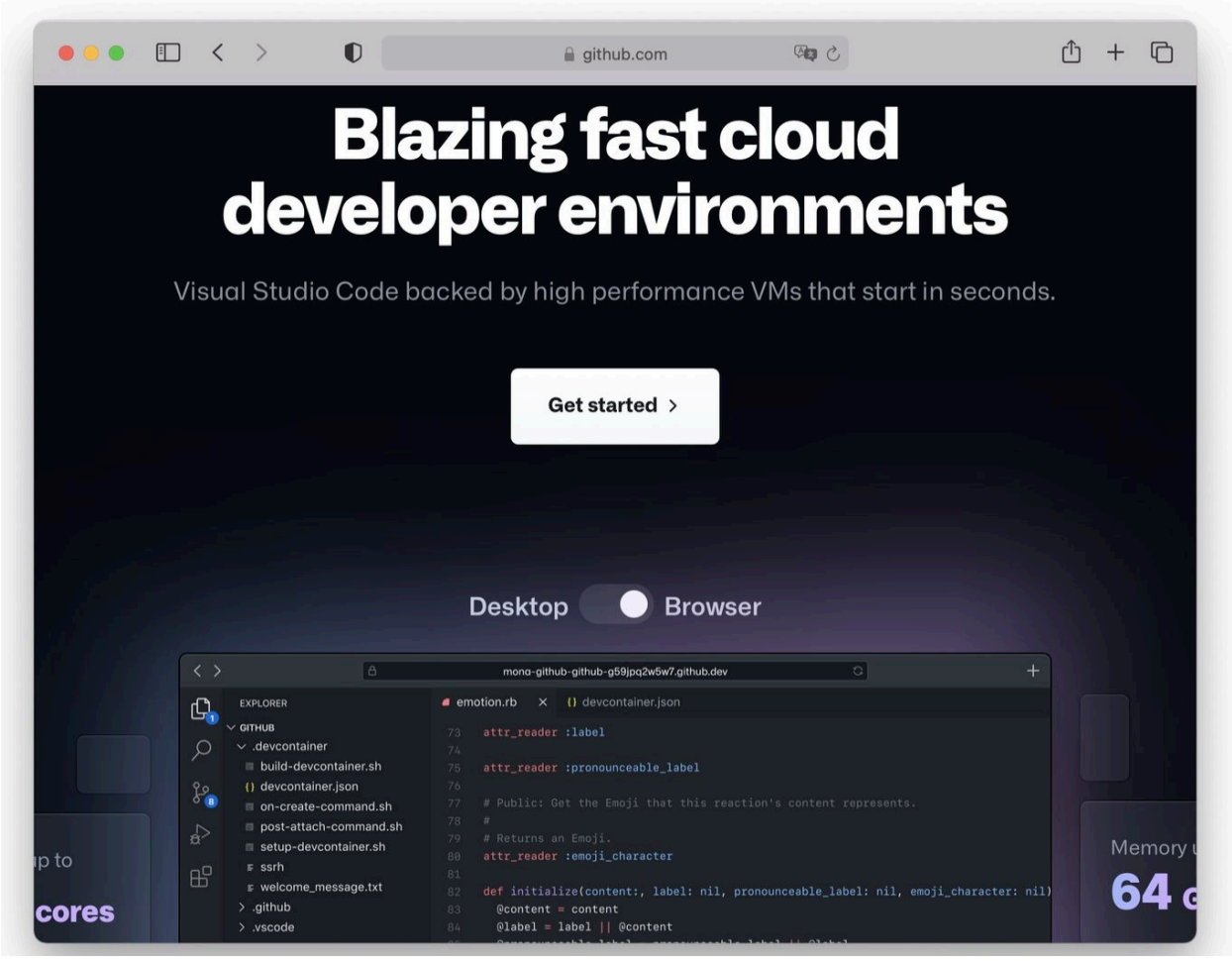
Next Chapter
6. Basic use of GitHub Codespaces (optional)

GitHub Codespaces Overview & Environment Configuration (Optional)

> **First, make sure you have a network environment that can access GitHub smoothly. Otherwise, it is recommended to use Alibaba Cloud.**

1. What is a codespace?

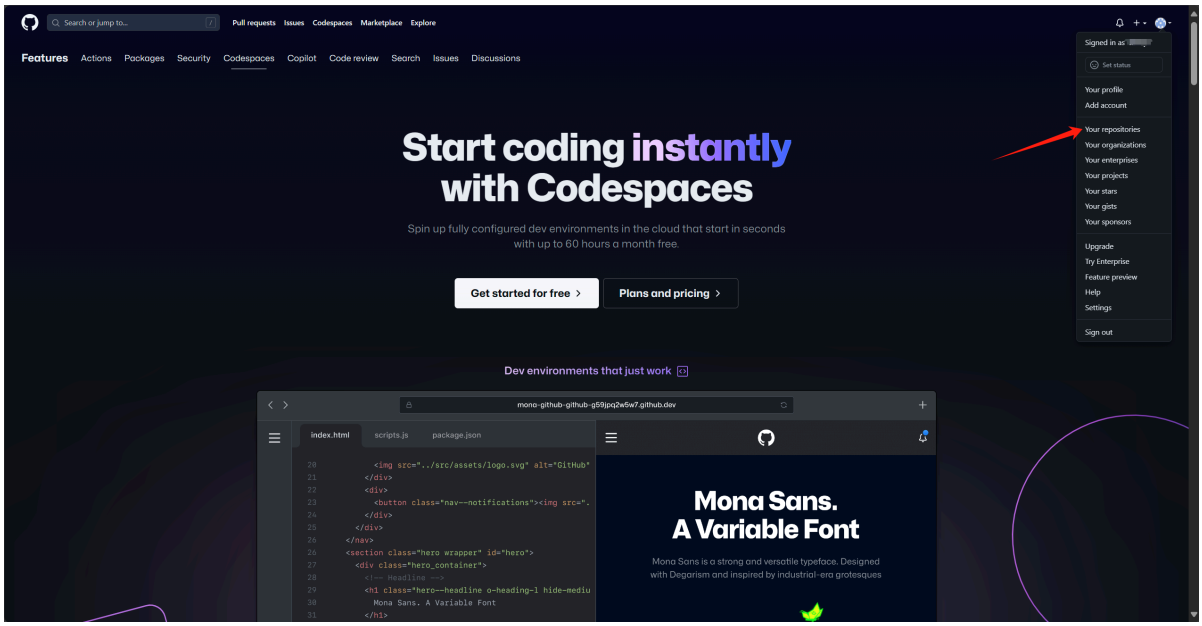
Codespaces are development environments hosted in the cloud. You can customize your projects for GitHub Codespaces by committing configuration files to a repository (often referred to as "configuration as code"), which creates a repeatable codespace configuration for all users of the project. For more information, see " **Introduction to development containers** ."



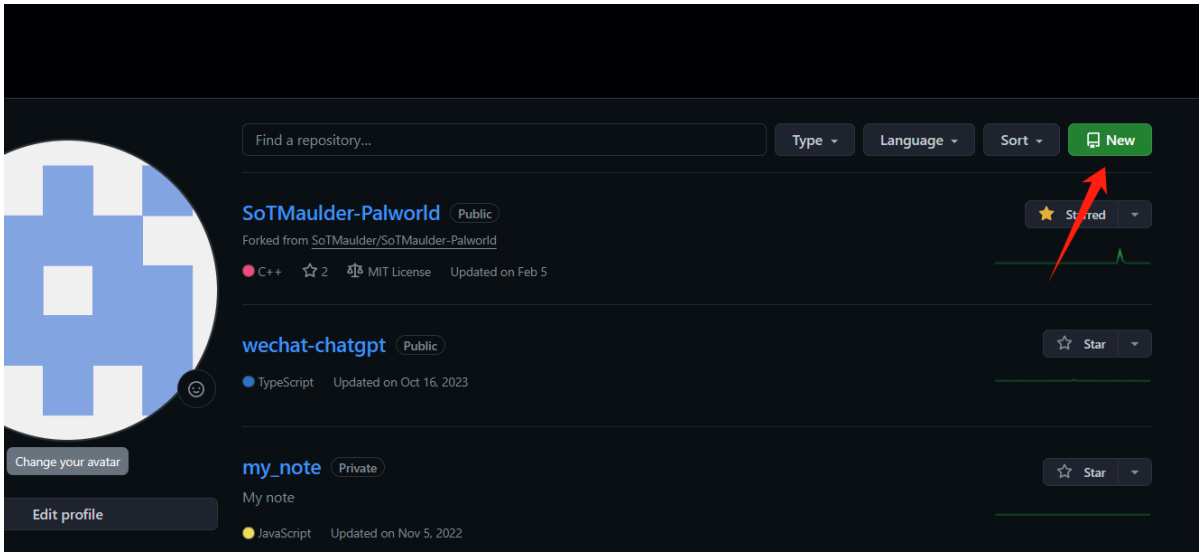
Official Documentation

2. Create the first codespace

1. Open the URL: https://github.com/features/codespaces
2. Log in to your GitHub account
3. Click the icon **Your repositories**

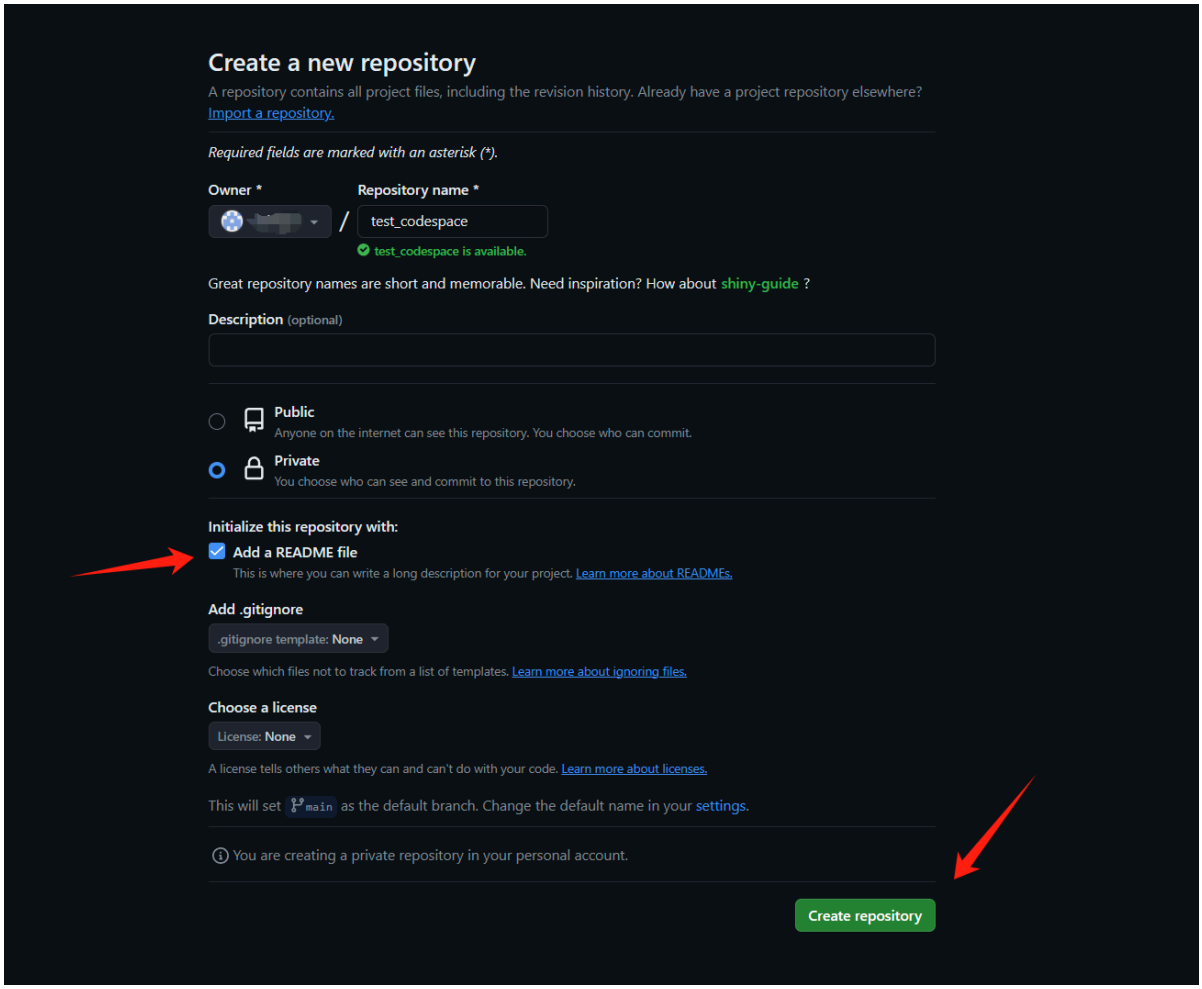


4. After entering your own repository list, click the **New** icon to create a new repository

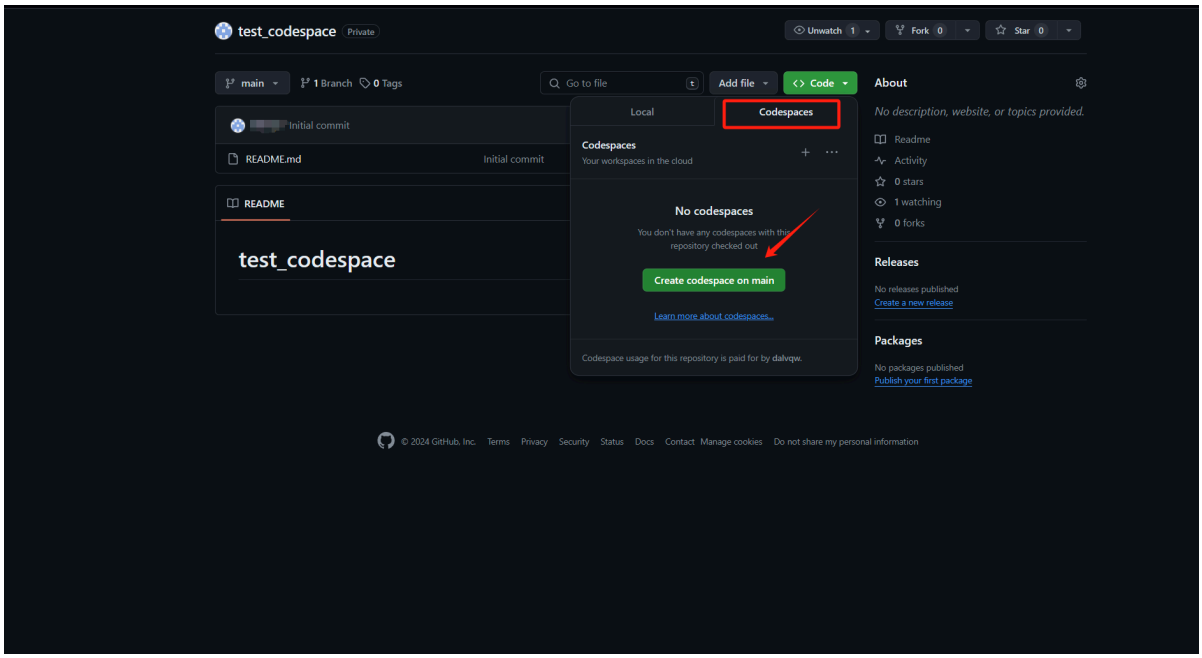


5. You can set it up as you like. For the sake of convenience and security, it is recommended to check **Add a README file** and select **Private** (because the API key is used in the course, pay

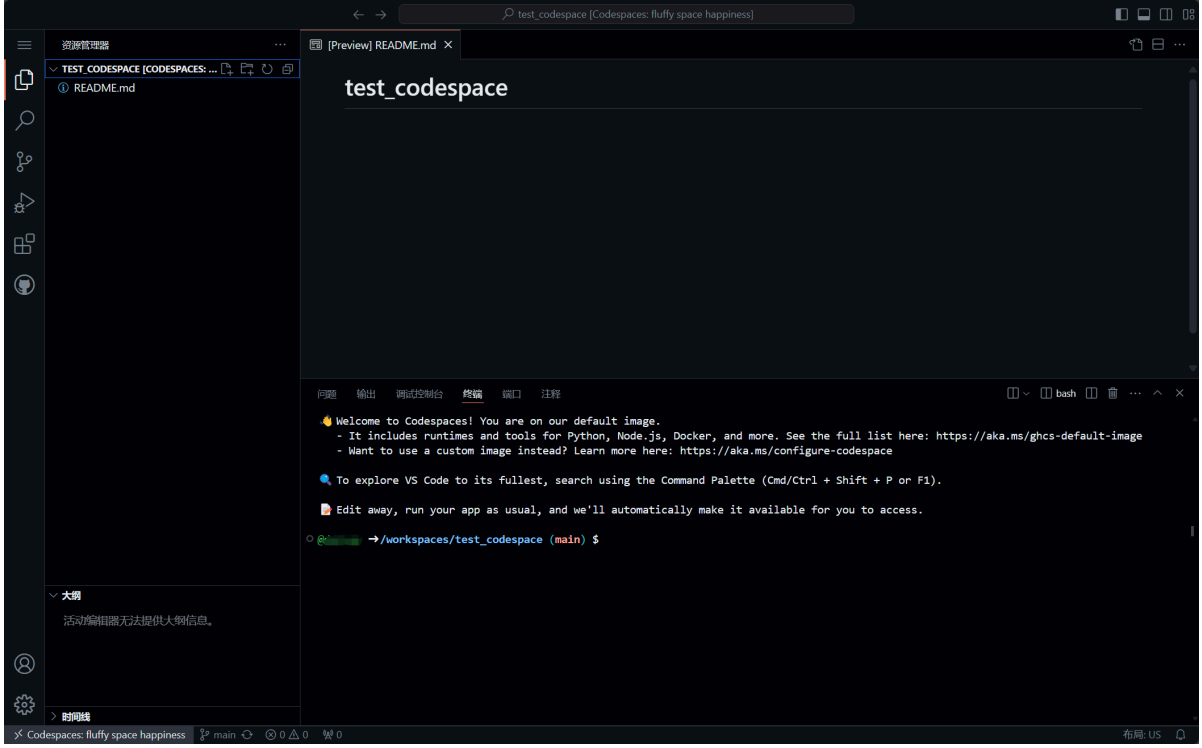
attention to privacy protection). After the settings are completed, click **Create repository**



6. After creating the repository, click **code** and select **Codespaces** . Click the icon **Create codespace on main**



7. After waiting for a while, the following interface will appear. The following operations are the same as VSCode. You can install plug-ins and adjust settings as needed.



3. Environment Configuration

Just refer 7. Environment configuration to 1.2 General environment configuration the configuration environment and skip the first two steps.

> **Since each repository can set up an independent codespace, we don't need to install the conda environment here. And because the GitHub server is overseas, there is no need to configure a domestic mirror source.**

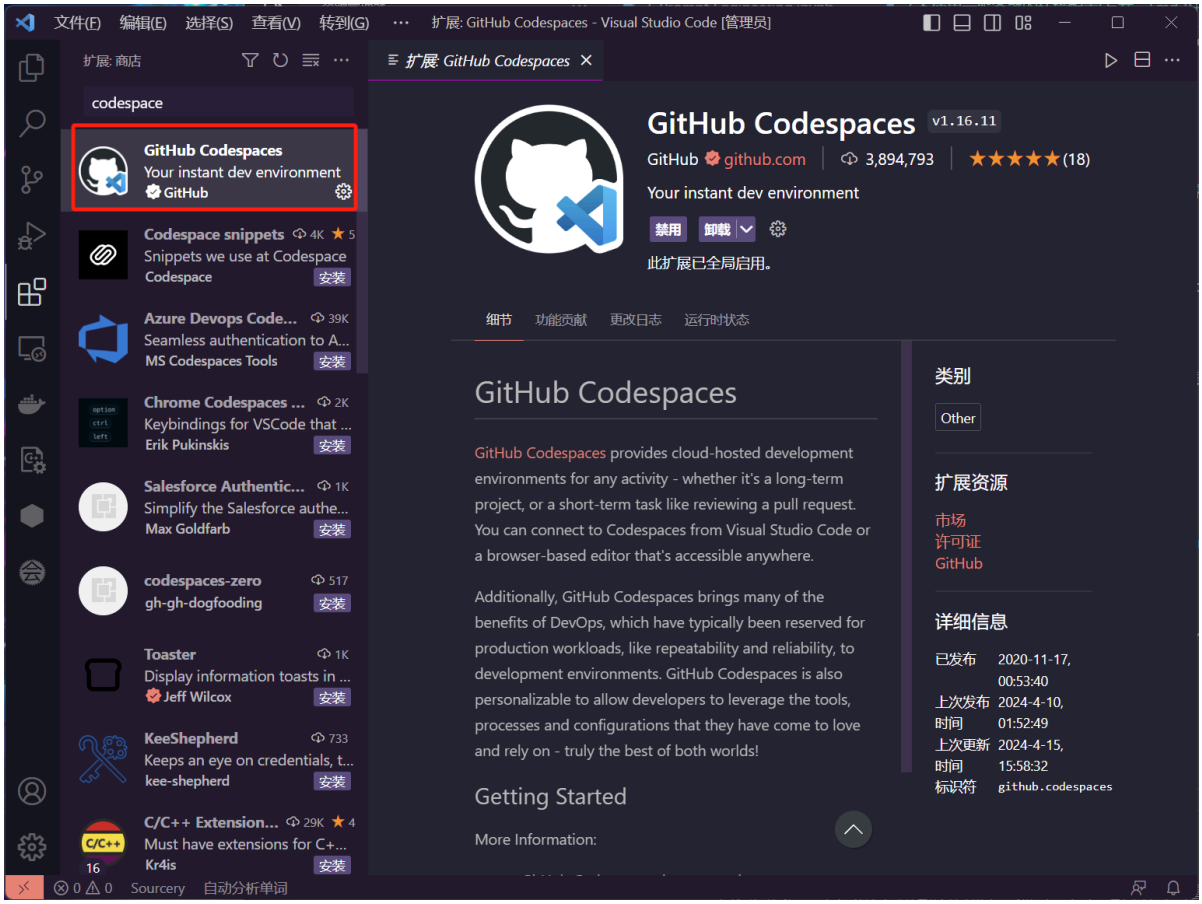
4. VSCode configures Python environment

Refer 7. Environment configuration to 2. VSCode configures Python environment the configuration environment

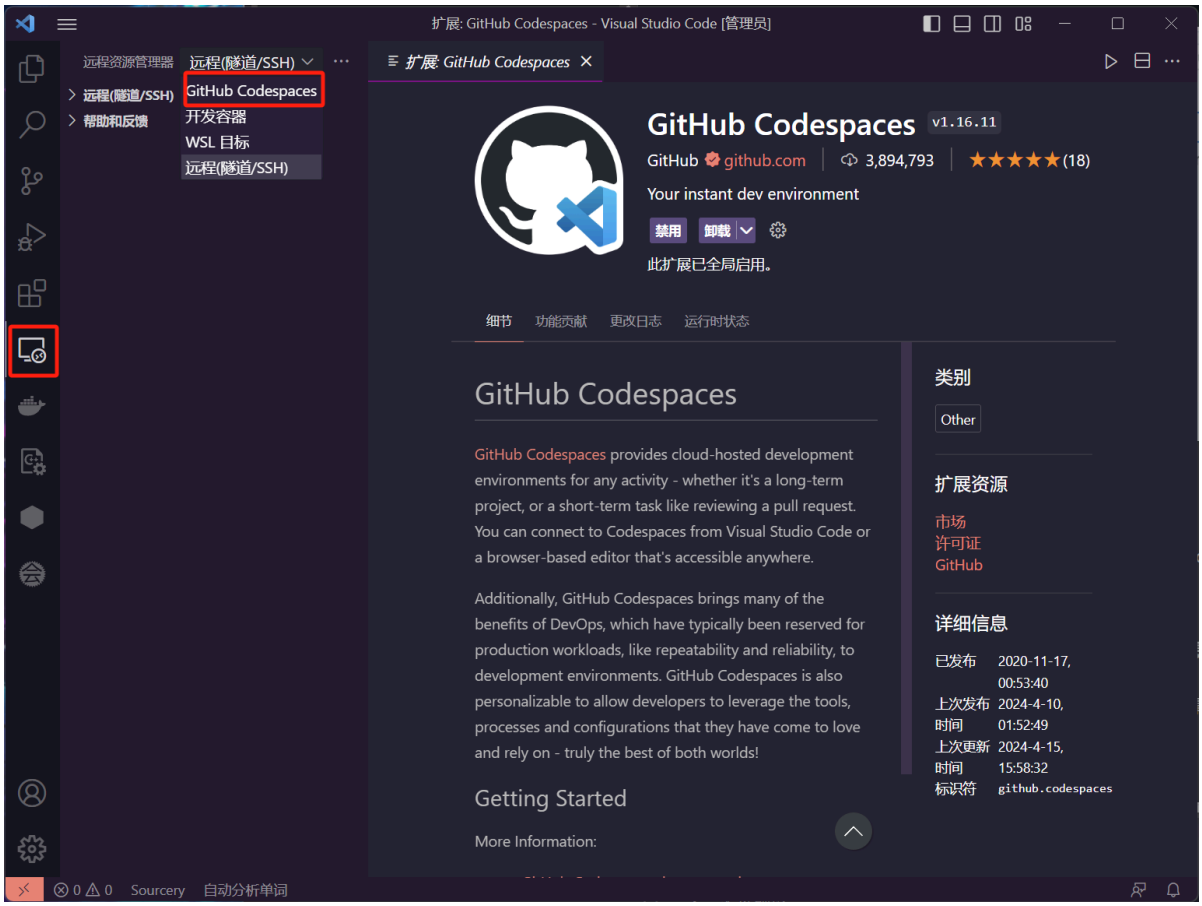
> **Note: After installing all configurations for the first time, you need to restart the codespace**

5. Connecting Local VSCode to Codespace (optional)

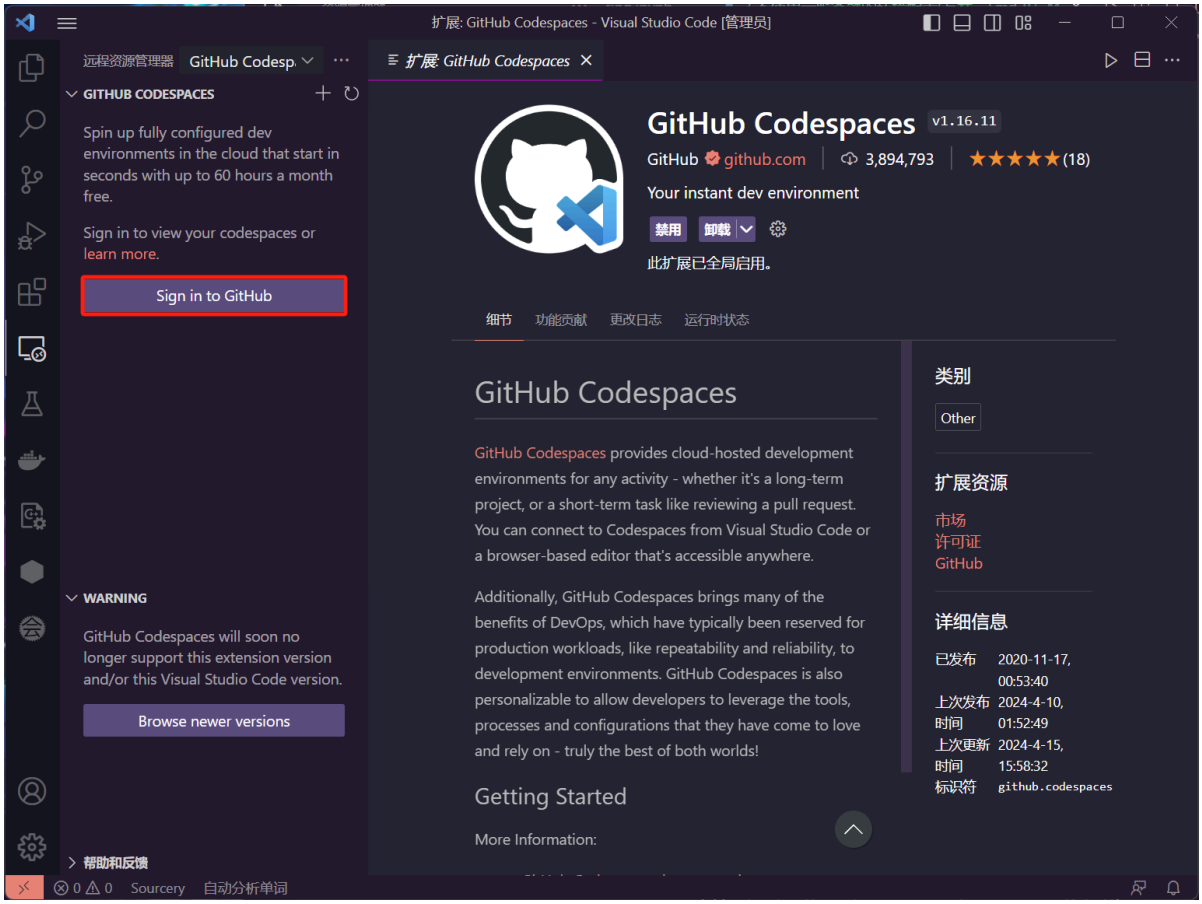
1. Open VSCode, search for codespace and install the plugin



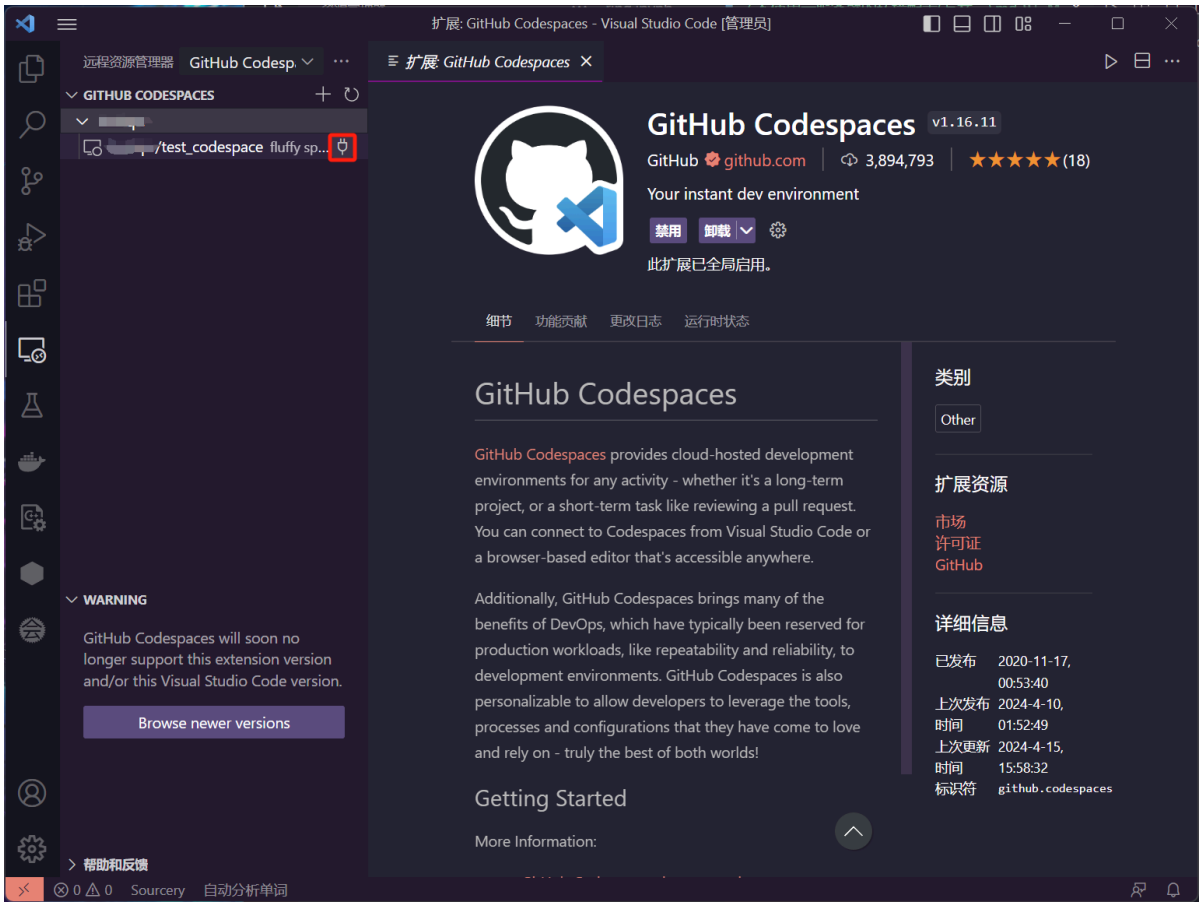
2. In the VS Code activity bar, click **the Remote Explorer** icon



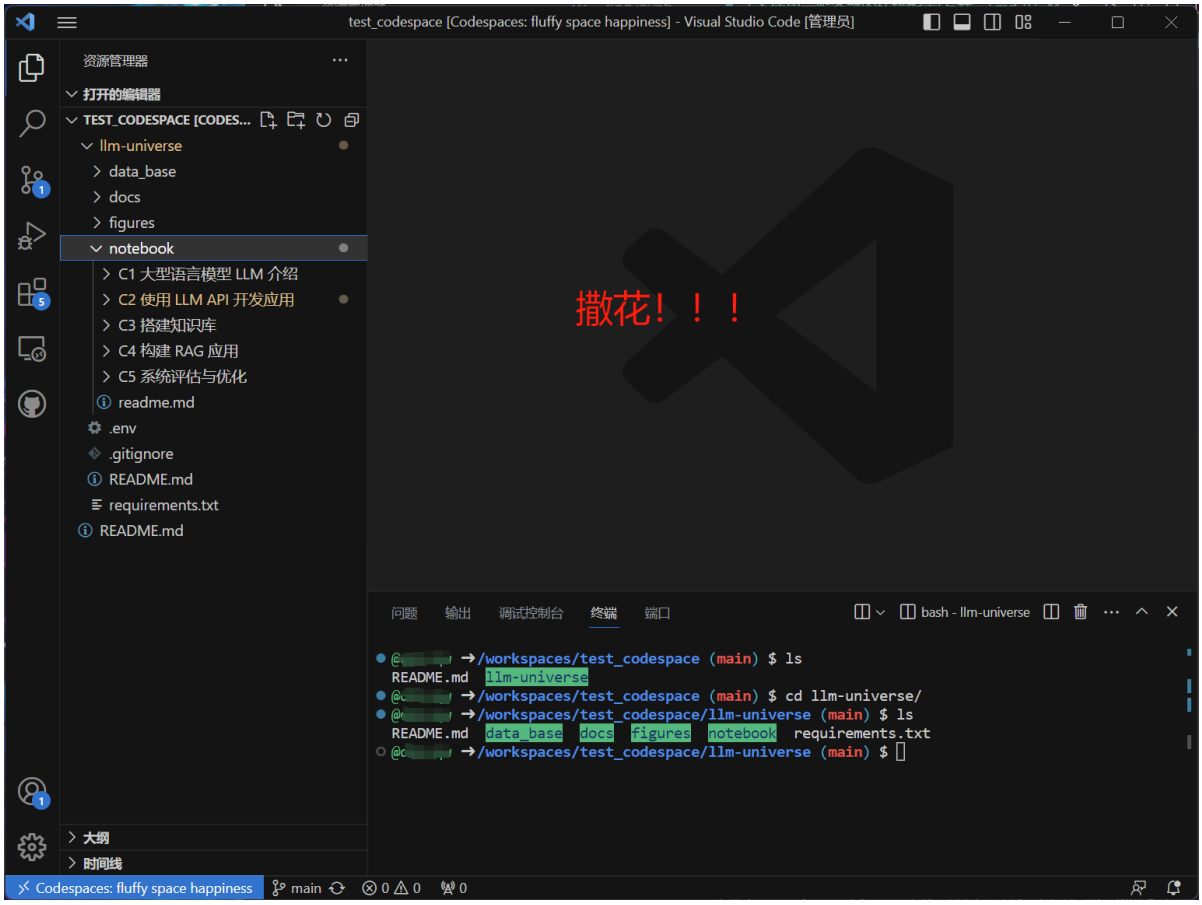
3. Log in to GitHub and log in according to the prompts



4. You can see the codespace we just created here. Click the red box connection icon



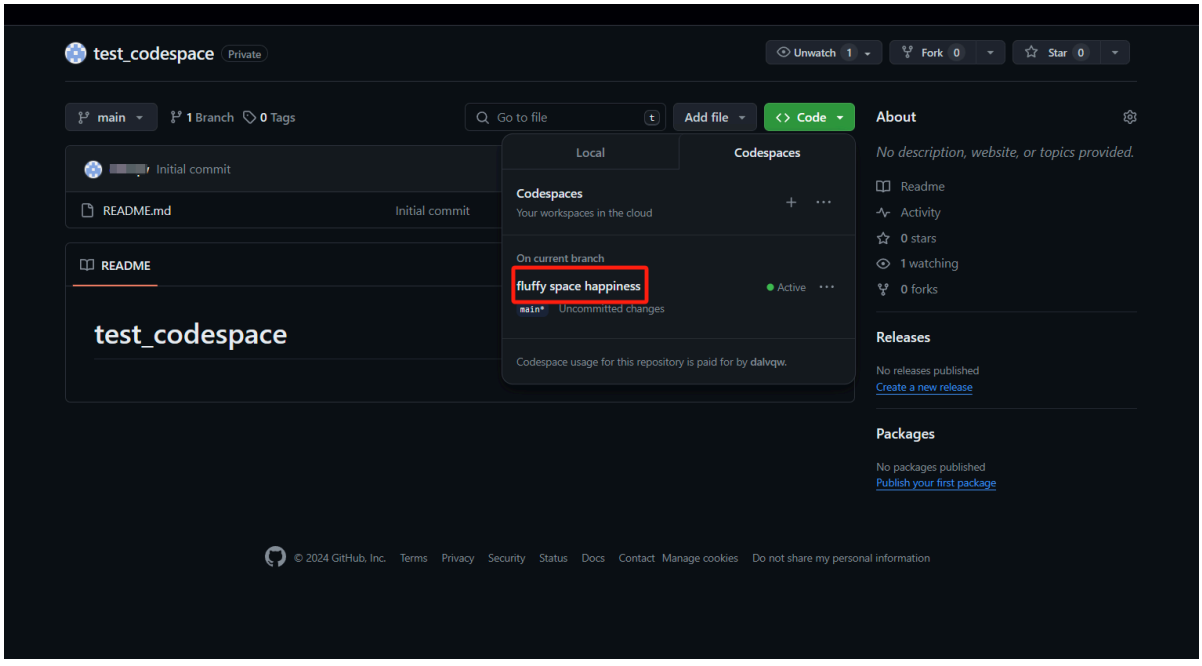
5. Successfully connected to codespace



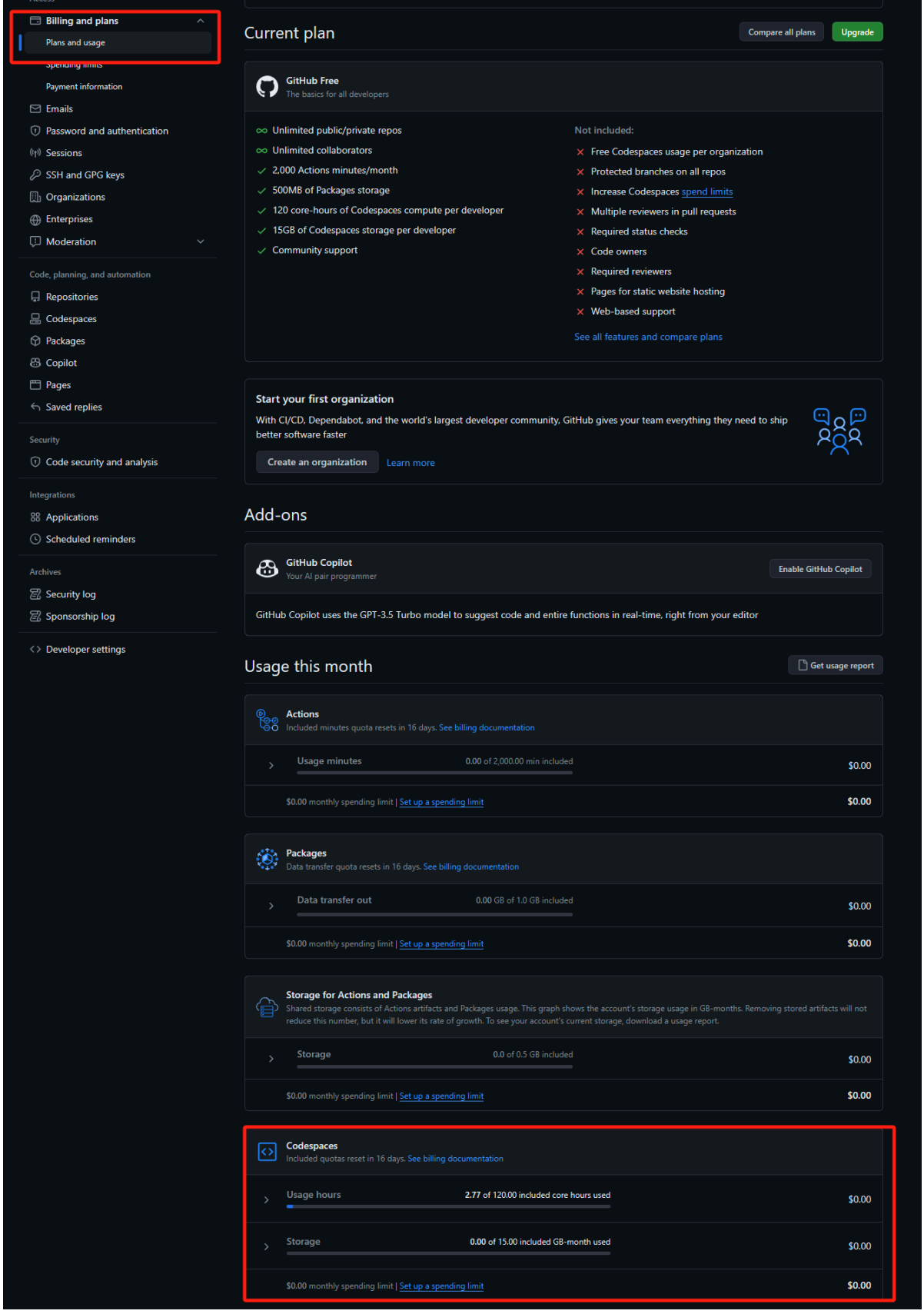
6. [VSCode official configuration document]

6. Others

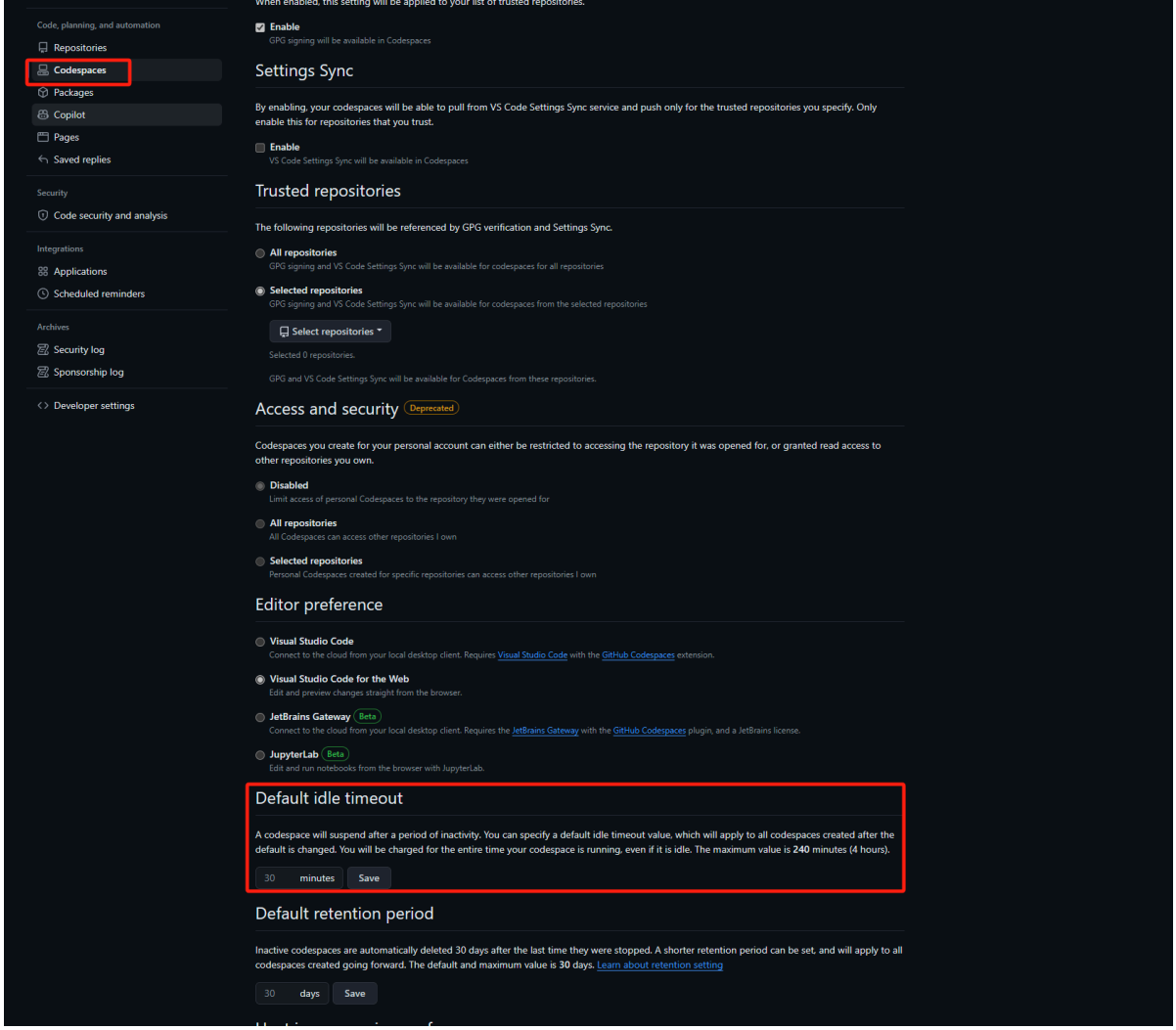
1. After closing the webpage, find the newly created repository and click the red box to select the content to re-enter the codespace



2. After finding the GitHub account settings, you can see the remaining free quota in **Plans and usage.**



3. Codespace settings, it is recommended to adjust the suspension time (too long a time will waste credits)



4. Because codespace can be accessed through the web, the most important thing is of course that you can **carry a tablet with you to access the web for programming learning.**

> **We now have the necessary foundation for development. In the next chapter, we will introduce the required environment configuration in detail.**

Previous chapter
5. Alibaba Cloud Server Usage Guide

Next Chapter
7. Environment Configuration

Environment Configuration

This chapter mainly provides some necessary environment configuration guides, including code environment configuration, Python environment configuration of VSCODE code editor, and some other resource configurations used.

1. Code Environment Configuration Guide

Here we introduce each step of code environment configuration in detail, which is divided into two parts: basic environment configuration and general environment configuration to meet the needs of different users and environments.

- **Basic environment configuration** section: Suitable for **beginners** of environment configuration or **new server environments (such as Alibaba Cloud)** . This section introduces how to generate an SSH key and add it to GitHub, as well as how to install and initialize the conda environment.

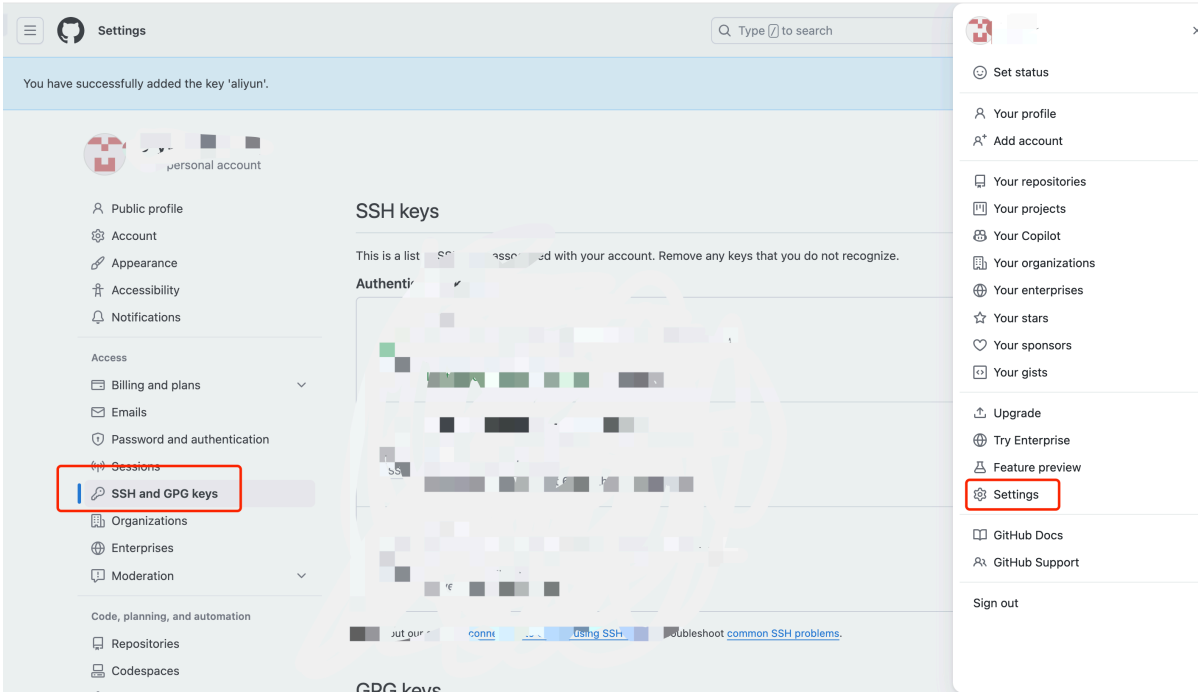
- **General environment configuration** section: suitable for **users with some experience** , local installations with **existing environment foundations , or completely independent environments (such as GitHub Codespace)** . This section describes how to create and activate a conda virtual environment, clone a project repository, switch to the project directory, and install the required Python packages. To speed up the installation of Python packages, we also provide some domestic mirror sources. *For completely independent environments, you can skip the first two steps about virtual environment (conda) configuration* .

1.1 Basic environment configuration (configure git and conda)

1. Generate SSH key `ssh-keygen -t rsa -C "youremail@example.com"`

2. Add the public key to GitHub `cat ~/.ssh/id_rsa.pub` Copy the output content, open GitHub, click the avatar in the upper right corner, select `settings` -> `SSH and GPG keys` -

> `New SSH key` , paste the copied content into key, and click `Add SSH key` .



3. Install conda environment

    1. Linux environment (usually using Linux environment)

        1. Install:

```shell
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

        2. initialization:

```shell
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

        3. Create a new terminal and check whether conda is installed successfully `conda --version`

    2. macOS environment

        1. Install

shell

```
mkdir -p ~/miniconda3
curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOS
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

2. initialization:

shell

```
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

3. Create a new terminal and check whether conda is installed successfully `conda --version`

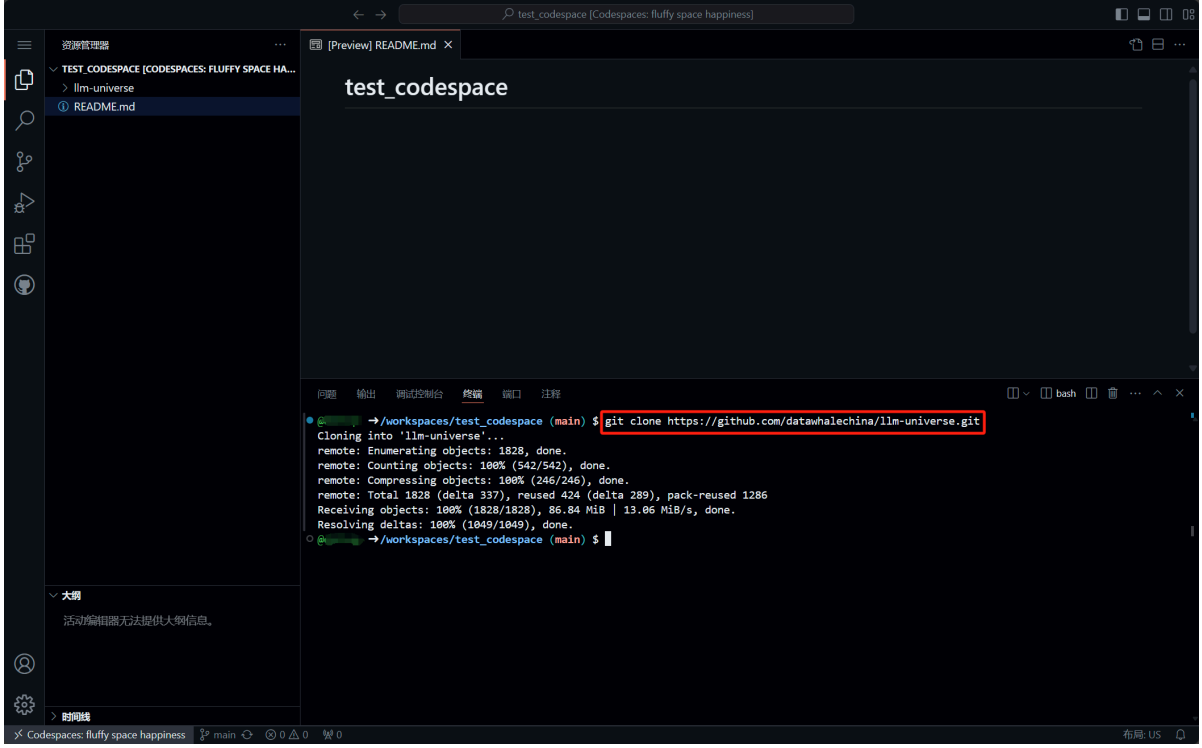
3. Windows environment

1. download: `curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe -o miniconda.exe`
2. Installation: Click on the downloaded file `miniconda.exe` and follow the installation instructions to install it
3. Open Anaconda Prompt in the menu to check whether conda is installed successfully `conda --version`
4. Delete the installation package: `del miniconda.exe`

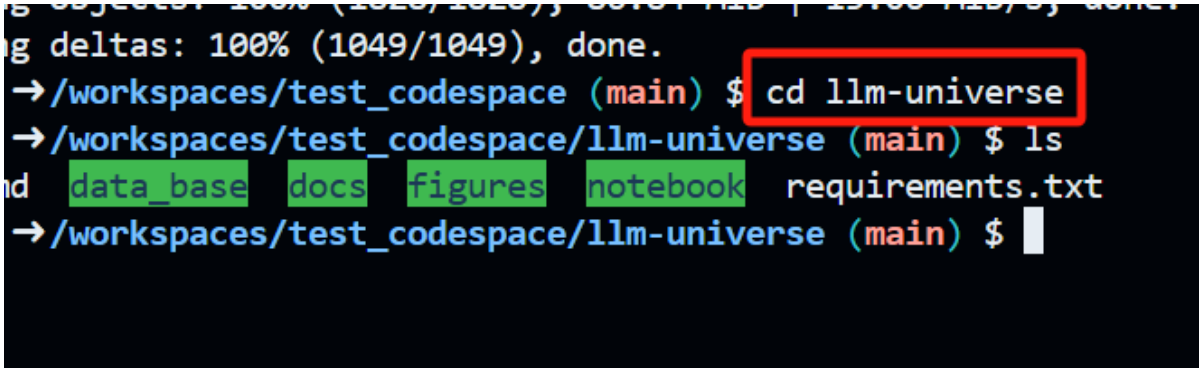
4. Please refer to the following `通用环境配置` section for subsequent configuration

1.2 General environment configuration

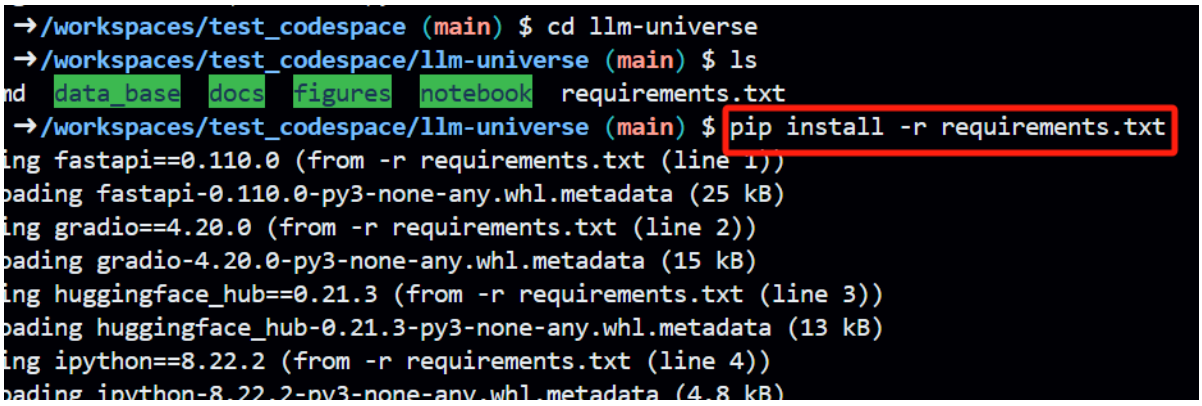
1. Create a new virtual environment `conda create -n llm-universe python=3.10`
2. Activate the virtual environment `conda activate llm-universe`
3. Clone the current repository in the path where you want to store the project `git clone git@github.com:datawhalechina/llm-universe.git`



4. Change directory to llm-universe `cd llm-universe`



5. Install the required packages. `pip install -r requirements.txt`



Usually you can speed up the installation through Tsinghua source `pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple`

> **Here is a list of commonly used domestic mirror sources. When the mirror source is not stable, you can switch it as needed: Tsinghua University: https://pypi.tuna.tsinghua.edu.cn/simple/Alibaba Cloud: http://mirrors.aliyun.com/pypi/simple/University of Science and Technology of China: https://pypi.mirrors.ustc.edu.cn/simple/Huazhong University of Science and Technology:**

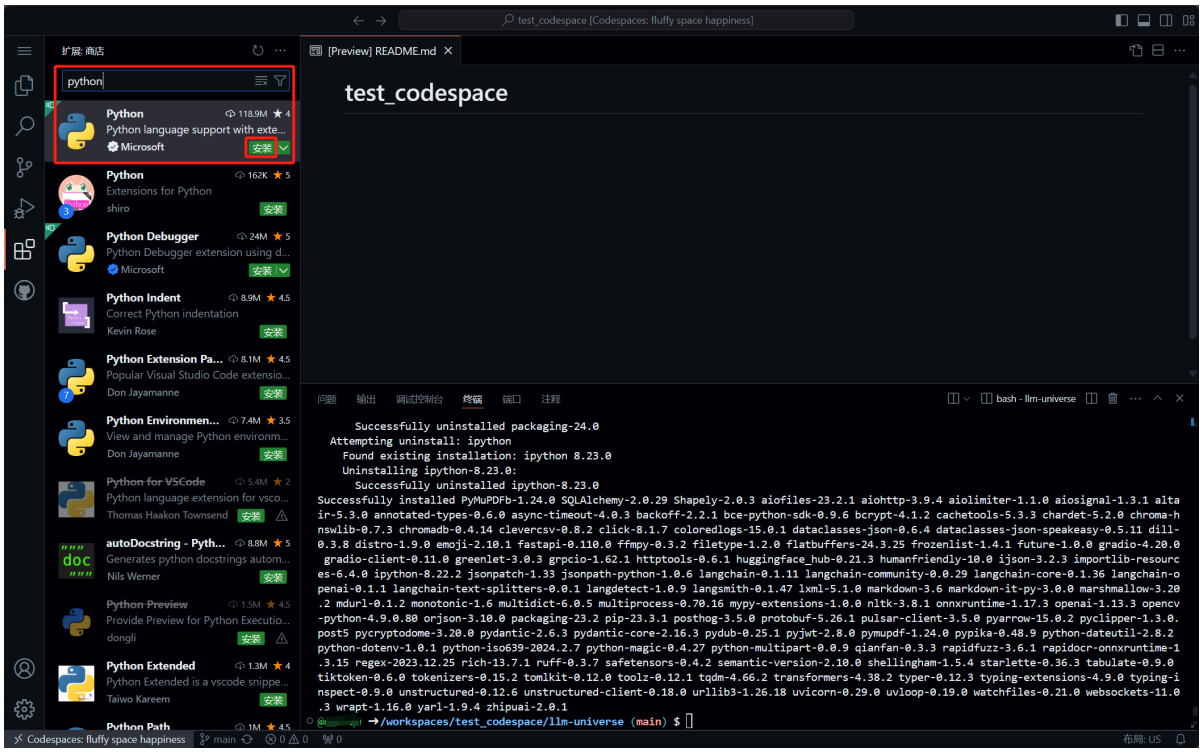
> http://pypi.hustunique.com/simple/Shanghai Jiaotong University: https://mirror.sjtu.edu.cn/pypi/web/simple/Douban : http://pypi.douban.com/simple

2. VSCode configures Python environment

1. Installing the Python plugin

   This tutorial is developed based on Python language. For a better development experience, we need to install the Python plug-in.

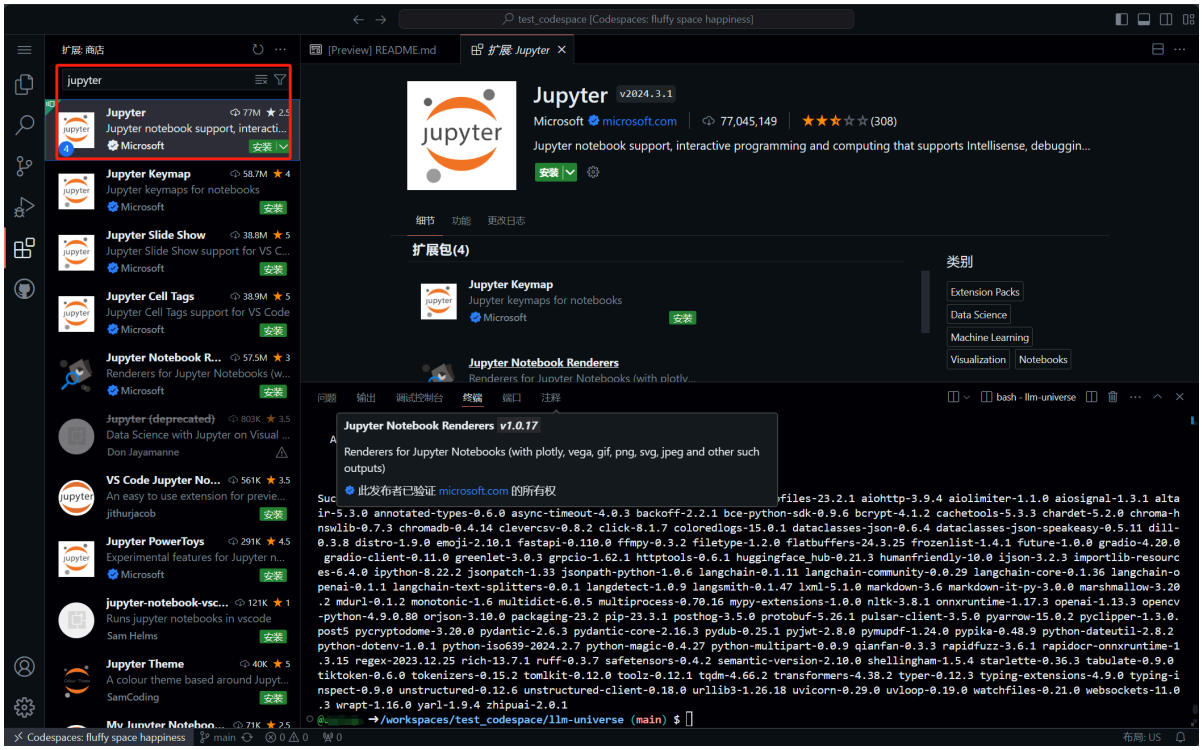
   Search in the plugin market `Python` , find `Python` the plugin and install it.



   When we execute Python code, it will automatically recognize our Python environment and provide functions such as code completion to facilitate our development.

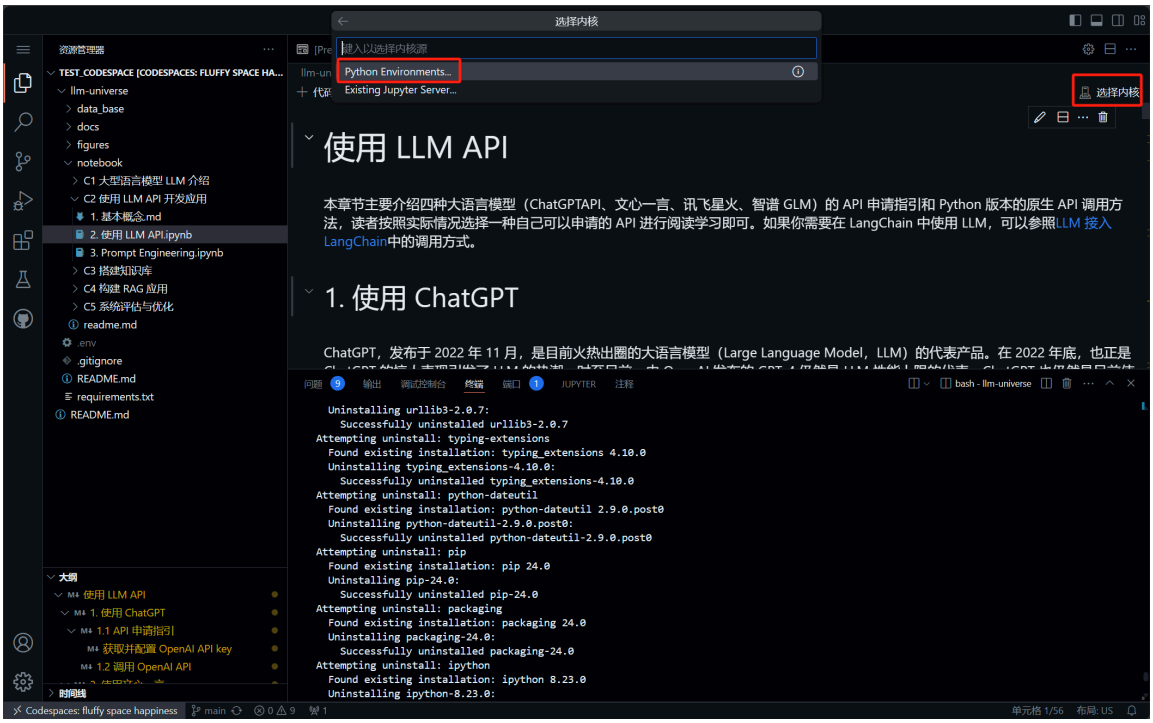
2. Install Jupyter plugin In this tutorial, we use Jupyter Notebook for development, so we need to install the Jupyter plugin. Search in the plugin market `Jupyter` , find `Jupyter` the

plugin and install it.



3. Configuring the Python environment for Jupyter Notebook

    1. Open a Jupyter Notebook
    2. Click in the upper right corner Select the Python interpreter (the display will change depending on the name of the environment you selected) to select the Python environment for the current Jupyter Notebook.



    3. Click choose Python to enter the environment list and select the environment we configured `llm-universe` .

After that we can use our Python environment for development in Jupyter Notebook.

3. Download other resources

3.1 Download NLTK related resources

When we use the open source word vector model to build open source word vectors, we need to use some resources of the third-party library nltk. Under normal circumstances, it will be automatically downloaded from the Internet, but the download may be interrupted due to network reasons. When we use nltk, an error will be reported. Here we download relevant resources from the domestic warehouse mirror address.

We use the following command to download the nltk resource and decompress it:

```shell
cd /root
git clone https://gitee.com/yzy0612/nltk_data.git  --branch gh-pages
cd nltk_data
mv packages/*  ./
cd tokenizers
unzip punkt.zip
cd ../taggers
unzip averaged_perceptron_tagger.zip
```

6. Basic use of GitHub Codespaces (optional)