


```
docker --version
docker compose version
```

配置 docker

```
mkdir -p /etc/docker
cat > /etc/docker/daemon.json <<EOF
{
  "registry-mirrors": [
    "https://docker.mirrors.ustc.edu.cn",
    "https://hub-mirror.c.163.com"
  ]
}
EOF
systemctl daemon-reload
systemctl restart docke
```

Docker

```
docker --version
docker compose version
```

版本

```
Docker >= 24
Docker Compose >= 2.x
```



目录

```
mkdir -p /opt/litellm
cd /opt/litellm
```

安装

```
/opt/litellm
├─ docker-compose.yml
├─ .env
└─ config.yaml
```



LiteLLM



config.yaml

```
vi config.yaml
```



```
model_list:

# OpenAI
- model_name: gpt-5
  litellm_params:
    model: openai/gpt-5
    api_key: os.environ/OPENAI_API_KEY

# Gemini
- model_name: gemini-3.5-flash
  litellm_params:
    model: vertex_ai/gemini-3.5-flash
    api_key: os.environ/GEMINI_API_KEY

general_settings:

  master_key: os.environ/LITELLM_MASTER_KEY

#
store_model_in_db: true
```



.env

```
vi .env
```



```
LITELLM_MASTER_KEY=sk-litellm-admin-123456

OPENAI_API_KEY=sk-xxxx

GEMINI_API_KEY=xxxx
```

Docker Compose



```
vi docker-compose.yml
```



```
services:

  litellm:
    image: ghcr.io/berriai/litellm:1.87.0

    container_name: litellm

    restart: always
```

ports:

- "4000:4000"

environment:

STORE_MODEL_IN_DB: "True"

DATABASE_URL: postgresql://litellm:litellm@postgres:5432/litellm

LITELLM_MASTER_KEY: \${LITELLM_MASTER_KEY}

OPENAI_API_KEY: \${OPENAI_API_KEY}

GEMINI_API_KEY: \${GEMINI_API_KEY}

volumes:

- ./config.yaml:/app/config.yaml

command:

--config /app/config.yaml

depends_on:

- postgres

postgres:

image: postgres:16

container_name: litellm-postgres

restart: always

```
environment:
```

```
    POSTGRES_DB: litellm
```

```
    POSTGRES_USER: litellm
```

```
    POSTGRES_PASSWORD: litellm
```

```
volumes:
```

```
  - ./postgres:/var/lib/postgresql/data
```



```
# 拉取ghcr镜像
```

```
docker pull ghcr.nju.edu.cn/berriai/litellm:1.87.0
```

```
# 使用compose
```

```
docker tag ghcr.nju.edu.cn/berriai/litellm:1.87.0 ghcr.io/berriai/litellm:1.87.0
```

```
docker rmi ghcr.nju.edu.cn/berriai/litellm:1.87.0
```

```
docker compose up -d
```



```
docker ps
```



```
litellm
```

```
litellm-postgres
```



```
docker logs -f litellm
```



LiteLLM Proxy Server Started
Server running on port 4000



API

LiteLLM  OpenAI API 



```
curl http://127.0.0.1:4000/v1/models \  
-H "Authorization: Bearer sk-litellm-admin-123456"
```



```
{  
  "data": [  
    {  
      "id": "gpt-5"  
    }  
  ]  
}
```



```
curl http://127.0.0.1:4000/v1/chat/completions \  
-H "Authorization: Bearer sk-litellm-admin-123456" \  
-H "Content-Type: application/json" \  
-d '  
{  
  "model": "gpt-5",  
  "messages": [  
    {  
      "role": "user",  
      "content": "hello"  
    }  
  ]  
}'
```

```
]
}'
```



LiteLLM v1.87 [Dashboard](#)



`http://IP:4000/ui`



`http://192.168.4.21:4000/ui`



Master Key:
`sk-litellm-admin-123456`



[PostgreSQL](#)

```
docker run \
-d \
--name litellm \
-p 4000:4000 \
-e STORE_MODEL_IN_DB=True \
docker.litellm.ai/berriai/litellm:1.87.0
```



- 部署
- 部署
- 部署
- Key

部署

部署 RAGFlow

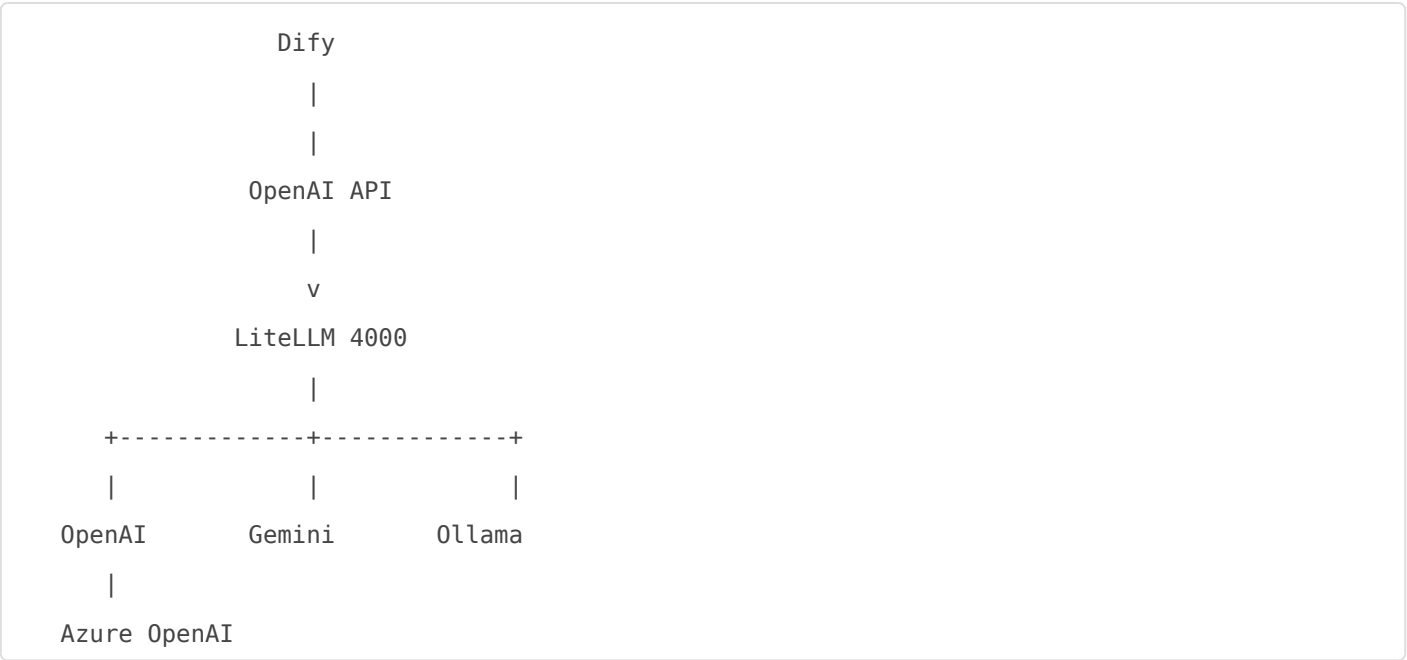
Dify /

部署

- Dify
- RAGFlow
- Keycloak
- PostgreSQL
- Docker Compose

LiteLLM 部署 AI Gateway

部署



Dify 部署

□□□□

OpenAI Compatible

Base URL:

http://litellm:4000/v1

API Key:

sk-litellm-admin-123456

□□

□□□□□□□□

□□□□□□□□□□

Ubuntu 24.04

|

Docker

|

Nginx HTTPS

|

LiteLLM 1.87

|

PostgreSQL 16

|

Keycloak OAuth

□□

- Redis□□□
- Prometheus□□□
- Grafana□ Dashboard□

□□□□

LITELLM_MASTER_KEY=sk-litellm-admin-123456

🏠🏠🏠🏠🏠

1. 🏠 `.env`

🏠🏠

```
cd /opt/litellm
vi .env
```

🏠🏠🏠🏠

```
LITELLM_MASTER_KEY=sk-litellm-admin-9f83a7d2c1e54b6a

OPENAI_API_KEY=sk-xxxx

GEMINI_API_KEY=xxxx
```

🏠🏠🏠🏠🏠🏠🏠

```
openssl rand -hex 32
```

🏠🏠

```
a9d7f3e8b4c1d2e6f7a8b9c0d1e2f3456789abcd
```

🏠🏠

```
LITELLM_MASTER_KEY=sk-a9d7f3e8b4c1d2e6f7a8b9c0d1e2f3456789abcd
```

2. 🏠 LiteLLM

🏠🏠🏠🏠🏠🏠🏠🏠🏠🏠🏠

```
docker compose down
```

```
docker compose up -d
```

□□□

```
docker compose restart litellm
```

□□ `.env` □ `compose` □□□□□

```
docker compose up -d --force-recreate litellm
```

□□□□□□□□□□

3. □□□ Key

□□□□

```
docker exec litellm env | grep LITELLM_MASTER_KEY
```

□□□□

```
LITELLM_MASTER_KEY=sk-litellm-admin-9f83a7d2c1e54b6a
```

4. Dashboard□□□□

□□□

```
http://□□□IP:4000/ui
```

□□□

Master Key:

```
sk-litellm-admin-9f83a7d2c1e54b6a
```

□□□

□□□□□□□□

.env

□

restart

□□□□□□□□

□□□

```
vi .env
docker compose restart litellm
```

□□□□□□□□□□□□□□□□

□□□

```
docker inspect litellm | grep LITELLM_MASTER_KEY
```

□□□□□□□□

```
docker compose up -d --force-recreate litellm
```

□□□□□□□□

□□□□

```
LITELLM_MASTER_KEY=sk-litellm-admin-123456
```

□□□□□□□□

□□□

```
LITELLM_MASTER_KEY=sk-prod-$(openssl rand -hex 32)
```

□□□

- Master Key □□□□□ Dashboard
- Dify/RAGFlow □□□□□□□ API Key
- □□□ Master Key □□□□□□□

□□□□□ Dify□□□□□ LiteLLM Dashboard □□□

Keys → Generate Key

1. 创建密钥

sk-dify-prod-xxxx

2. 配置 Dify 主密钥

3. 配置 LiteLLM 模型 ID 为 DashScope / Model Studio 的 qwen3.7-max

4. 配置 LiteLLM 模型 ID 为 OpenAI Compatible 的 OpenAI 模型 ID

- model
- api_base
- api_key

5. 配置 LiteLLM 模型 ID 为 qwen3.7-max 的 OpenAI 模型 ID

qwen3.7-max

6. 配置 LiteLLM 模型 ID 为 OpenAI Compatible Endpoint (AlibabaCloud)

1. 创建 .env

1. 创建

```
cat /opt/litellm/.env
```

2. 配置

```
LITELLM_MASTER_KEY=sk-litellm-admin-123456

OPENAI_API_KEY=sk-xxxx

GEMINI_API_KEY=xxxx
```

3. 配置

```
DASHSCOPE_API_KEY=sk-xxxxxxxxxxxxxxxxxx
```

'''

```
LITELLM_MASTER_KEY=sk-litellm-admin-123456

OPENAI_API_KEY=sk-xxxx

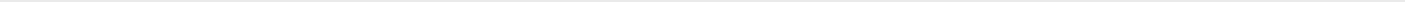
GEMINI_API_KEY=xxxx

DASHSCOPE_API_KEY=sk-abcd123456789
```

''' Key '''

'''''''''' → API Key '''

'''



2. ''' LiteLLM config.yaml

''''''''

```
/opt/litellm/config.yaml
```

'''

```
model_list:

- model_name: gpt-5
  litellm_params:
    model: openai/gpt-5
    api_key: os.environ/OPENAI_API_KEY

- model_name: gemini-3.5-flash
  litellm_params:
    model: gemini/gemini-3.5-flash
    api_key: os.environ/GEMINI_API_KEY

# '''' qwen3.7-max
```

```
- model_name: qwen3.7-max
  litellm_params:
    model: openai/qwen3.7-max
    api_base: https://dashscope.aliyuncs.com/compatible-mode/v1
    api_key: os.environ/DASHSCOPE_API_KEY
```

LiteLLM `model_name` Dify/RAGFlow `litellm_params.model` (docs.litellm.com.cn)

3. LiteLLM

`.env`

`restart`

```
cd /opt/litellm
```

```
docker compose up -d --force-recreate litellm
```

```
docker compose logs -f litellm
```

4.

```
curl http://127.0.0.1:4000/v1/models \
-H "Authorization: Bearer sk-litellm-admin-123456"
```

```
{
  "data": [
    {
```



```
    "id": "gpt-5"
  },
  {
    "id": "gemini-3.5-flash"
  },
  {
    "id": "qwen3.7-max"
  }
]
}
```

5. qwen3.7-max

```
curl http://127.0.0.1:4000/v1/chat/completions \
-H "Authorization: Bearer sk-litellm-admin-123456" \
-H "Content-Type: application/json" \
-d '{
{
  "model": "qwen3.7-max",
  "messages": [
    {
      "role": "user",
      "content": "你好，你是谁？"
    }
  ]
}'
```

你好

```
{
  "choices": [
    {
      "message": {
        "content": "..."
      }
    }
  ]
}
```

```
}
```

6. 使用 Dify 部署

使用 Dify 部署 LiteLLM

配置

```
- model_name: qwen-max
  litellm_params:
    model: openai/qwen3.7-max
    api_base: https://dashscope.aliyuncs.com/compatible-mode/v1
    api_key: os.environ/DASHSCOPE_API_KEY
```

部署 Dify 模型

qwen-max

模型

qwen3.7-max
↓
qwen4-max
↓
DeepSeek

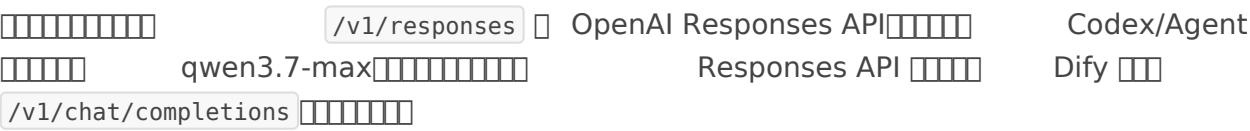
Dify 部署

7. 部署流程

```
graph TD
    Dify --> OpenAI[OpenAI API]
    OpenAI --> DeepSeek
```



阿里云 Dify + RAGFlow + 阿里云



Revision #5
Created 2026-07-21 09:36:37 UTC by Admin
Updated 2026-07-21 09:53:36 UTC by Admin