

ubuntu本地部署dify

一、安装docker

- 检验docker是否安装成功：

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

- 安装的docker方法：

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install docker.io
sudo apt-get install docker-compose
```

二、下载dify及配置端口号

1. 下载仓库

```
git clone https://github.com/langgenius/dify.git
```

```
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL$ cd dify/
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL/dify$ ls
AGENTS.md  AUTHORS  CONTRIBUTING.md  docker  images  Makefile  scripts  web
api        CLAUDE.md  dev             docs    LICENSE  README.md  sdks
```

2. 进入dify目录下的docker文件夹，将.env.example文件复制到.env

```
cd dify
cd docker
ls
```

```
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL/dify$ cd docker
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL/dify/docker$ ls
certbot                    docker-compose.yaml      README.md
couchbase-server           elasticsearch             ssrf_proxy
docker-compose.middleware.yaml generate_docker_compose  startupscripts
docker-compose.override.yml middleware.env.example    tidb
docker-compose.png         nginx                    volumes
docker-compose-template.yaml pgvector
```

这里我们需要找到dify的模板文件，但是在ubuntu系统里属于隐藏文件，可以使用快捷键`Ctrl + H`或者在终端里输入`ls -a`查看

```
ls -a
```

```
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL/dify/docker$ ls -a
.          docker-compose-template.yaml  pgvector
..         docker-compose.yaml      README.md
certbot    elasticsearch                     ssrf_proxy
couchbase-server .env.example                      startupscripts
docker-compose.middleware.yaml generate_docker_compose           tidb
docker-compose.override.yml  middleware.env.example           volumes
docker-compose.png          nginx
```

```
cp .env.example .env
```

3. 修改 .env 文件中的端口号

```
nano .env
```

在 nano 中，使用 **Ctrl + W** 来查找文本。使用 **Ctrl + W** 后输入文本并按 **Enter** 查找下一个匹配项。

- 查找 **EXPOSE_NGINX**

```
# -----
# Docker Compose Service Expose Host Port Configurations
# -----
EXPOSE_NGINX_PORT=80
EXPOSE_NGINX_SSL_PORT=443
```

- 修改端口号，防止冲突

端口冲突：端口 80 和 443 是 HTTP 和 HTTPS 协议的默认端口，很多系统或服务（如 Apache、其他 Web 服务、甚至系统级别的服务）都可能使用这些端口。如果你在同一台机器上运行多个 Web 服务或应用，可能会发生端口冲突，导致一个应用无法启动或无法访问。

通过将 Nginx 的默认端口改为非标准端口（如 8099 和 4433），可以避免与其他已经占用 80 和 443 端口的服务发生冲突。例如，若你的服务器同时运行着多个 Web 服务，你可以为每个服务分配不同的端口，避免冲突。

然后保存 **Ctrl + O** 并确定 **Enter**，然后退出 nano **Ctrl + X**

三、docker 换源

因为 DockerHub 服务器在国外，所以直接下载依赖可能会出现网络连接问题，所以需要给 Docker 配置。在 docker 中直接部署会出现网络问题，因此先配置使用镜像加速器，提高 Docker Hub 镜像拉取速度。

- 打开配置文件

```
sudo nano /etc/docker/daemon.json
```

- 修改配置文件

```
{  
    "registry-mirrors": ["https://docker.m.daocloud.io"]  
}
```

四、启动Docker和dify

1. 换源后重启Docker

真正的 docker-compose 文件在dify/docker/目录里，因此需要确保在docker目录下

```
cd docker
```

💡 第一次运行前建议先确保 .env 文件已配置好：确认数据库、端口、镜像源等参数都设置正确。

```
nano .env
```

- 重启docker

```
systemctl restart docker
```

配置好以后可以直接使用`sudo systemctl start docker`启动docker

🔍 检查 Docker 状态

```
sudo systemctl status docker
```

正常状态应该显示：`Active: active (running)`，如果显示 `inactive`或`failed`，则可以尝试重启

2. 确定docker compose版本

```
docker compose version
```

3. 启动docker compose（首次较慢...）

- 如果是v1.xx.x, 那么使用

```
docker-compose up -d
```

- 如果是v2.xx.x, 那么使用

```
docker compose up -d
```

```
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL/dify/docker$ systemctl restart docker
docker compose up -d
[+] Running 87/87
✔ api Pulled 770.3s
✔ worker_beat Pulled 770.3s
✔ weaviate Pulled 192.7s
✔ worker Pulled 770.3s
✔ ssrf_proxy Pulled 238.4s
✔ web Pulled 618.9s
✔ plugin_daemon Pulled 1022.6s
✔ sandbox Pulled 381.5s
✔ nginx Pulled 222.3s
✔ db Pulled 407.3s
✔ redis Pulled 304.6s
```

拉取所有必要的Docker镜像

```
[+] Running 13/13
✔ Network docker_default Created 0.0s
✔ Network docker_ssrf_proxy_network Created 0.0s
✔ Container docker-ssrf_proxy-1 Started 0.8s
✔ Container docker-weaviate-1 Started 0.8s
✔ Container docker-db-1 Healthy 3.2s
✔ Container docker-sandbox-1 Started 0.7s
✔ Container docker-web-1 Started 0.7s
✔ Container docker-redis-1 Started 0.8s
✔ Container docker-worker-1 Started 3.0s
✔ Container docker-plugin_daemon-1 Started 3.0s
✔ Container docker-api-1 Started 3.1s
✔ Container docker-worker_beat-1 Started 3.1s
✔ Container docker-nginx-1 Started 3.2s
```

创建Docker网络 启动所有服务容器 配置服务间的通信

4. 验证dify

```
docker compose ps
```

```
yuuki@yuuki-JiguangX-Series-GM6IR0C:~/Documents/AppsL/dify/docker$ docker compose ps
NAME                IMAGE                                COMMAND                                SERVICE    CREATED      STATUS      PORTS
docker-api-1        langgenius/dify-api:1.9.1          "/bin/bash /entrypoi..."          api        24 minutes ago Up 24 minutes 5801/tcp
docker-db-1         postgres:15-alpine                "docker-entrypoint.s..."          db         24 minutes ago Up 24 minutes (healthy) 5432/tcp
docker-nginx-1      nginx:latest                        "sh -c 'cp /docker-e..."          nginx      24 minutes ago Up 24 minutes 0.0.0.0:8099->80/tcp, [::]:8099->80/tcp, 0.0.0.0:443->443/tcp, [::]:443->443/tcp
docker-plugin_daemon-1 langgenius/dify-plugin-daemon:0.3.0-local "/bin/bash -c /app/e..." plugin_daemon 24 minutes ago Up 24 minutes 0.0.0.0:5803->5803/tcp, [::]:5803->5803/tcp
docker-redis-1      redis:6-alpine                     "docker-entrypoint.s..."          redis      24 minutes ago Up 24 minutes (healthy) 6379/tcp
docker-sandbox-1    langgenius/dify-sandbox:0.2.12     "/"main"                             sandbox     24 minutes ago Up 24 minutes (healthy)
docker-ssrf_proxy-1 ubuntu/squid:latest                "sh -c 'cp /docker-e..."          ssrf_proxy 24 minutes ago Up 24 minutes 3128/tcp
docker-weaviate-1    semitechnologies/weaviate:1.27.0   "/bin/weaviate --hos..."          weaviate   24 minutes ago Up 24 minutes 8080/tcp
docker-web-1        langgenius/dify-web:1.9.1          "/bin/sh -/entrypoi..."          web        24 minutes ago Up 24 minutes 5801/tcp
docker-worker-1     langgenius/dify-api:1.9.1          "/bin/bash /entrypoi..."          worker     24 minutes ago Up 24 minutes
docker-worker_beat-1 langgenius/dify-api:1.9.1          "/bin/bash /entrypoi..."          worker_beat 24 minutes ago Up 24 minutes 5801/tcp
```

可以看到：3个业务服务 api / worker / web；6个基础组件 weaviate / db / redis / nginx / ssrf_proxy / sandbox

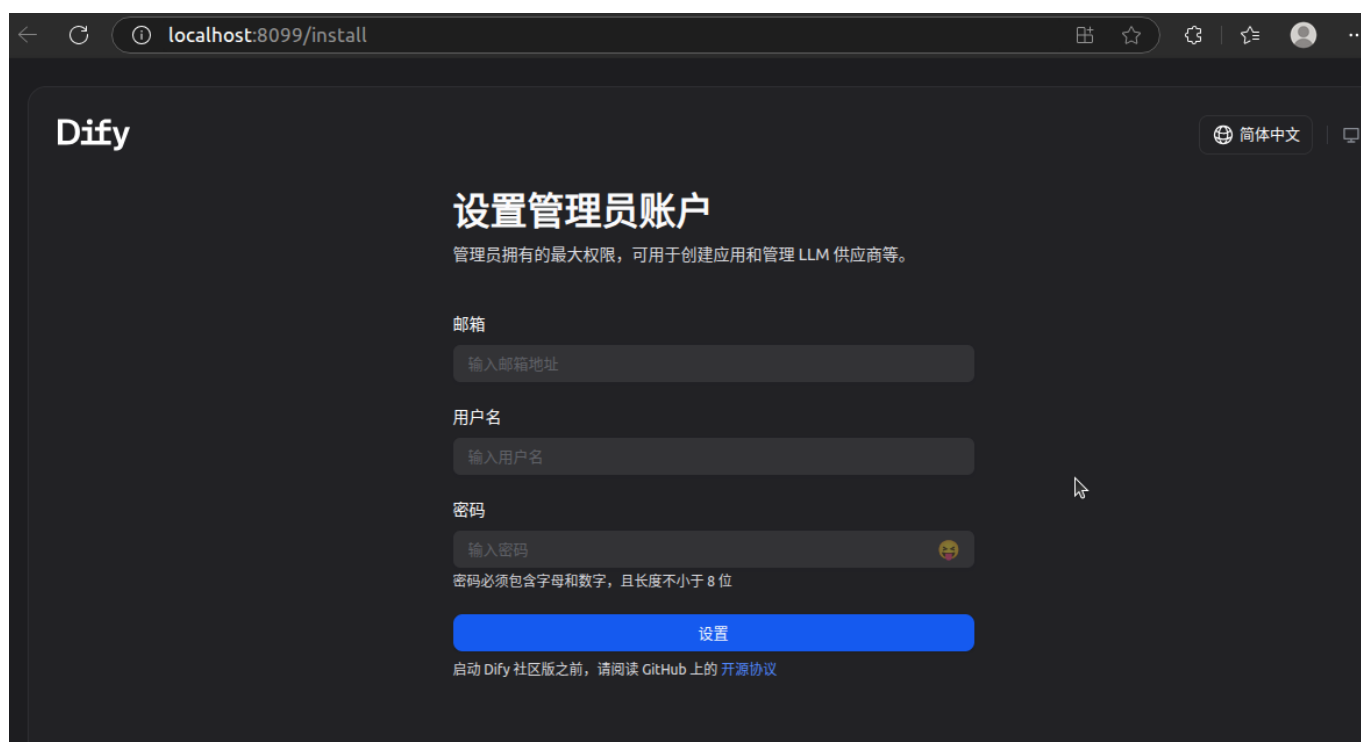
五、使用dify

1. 打开dify网页

根据之前修改的EXPOSE_NGINX_PORT=8099，可以在浏览器里打开以下地址

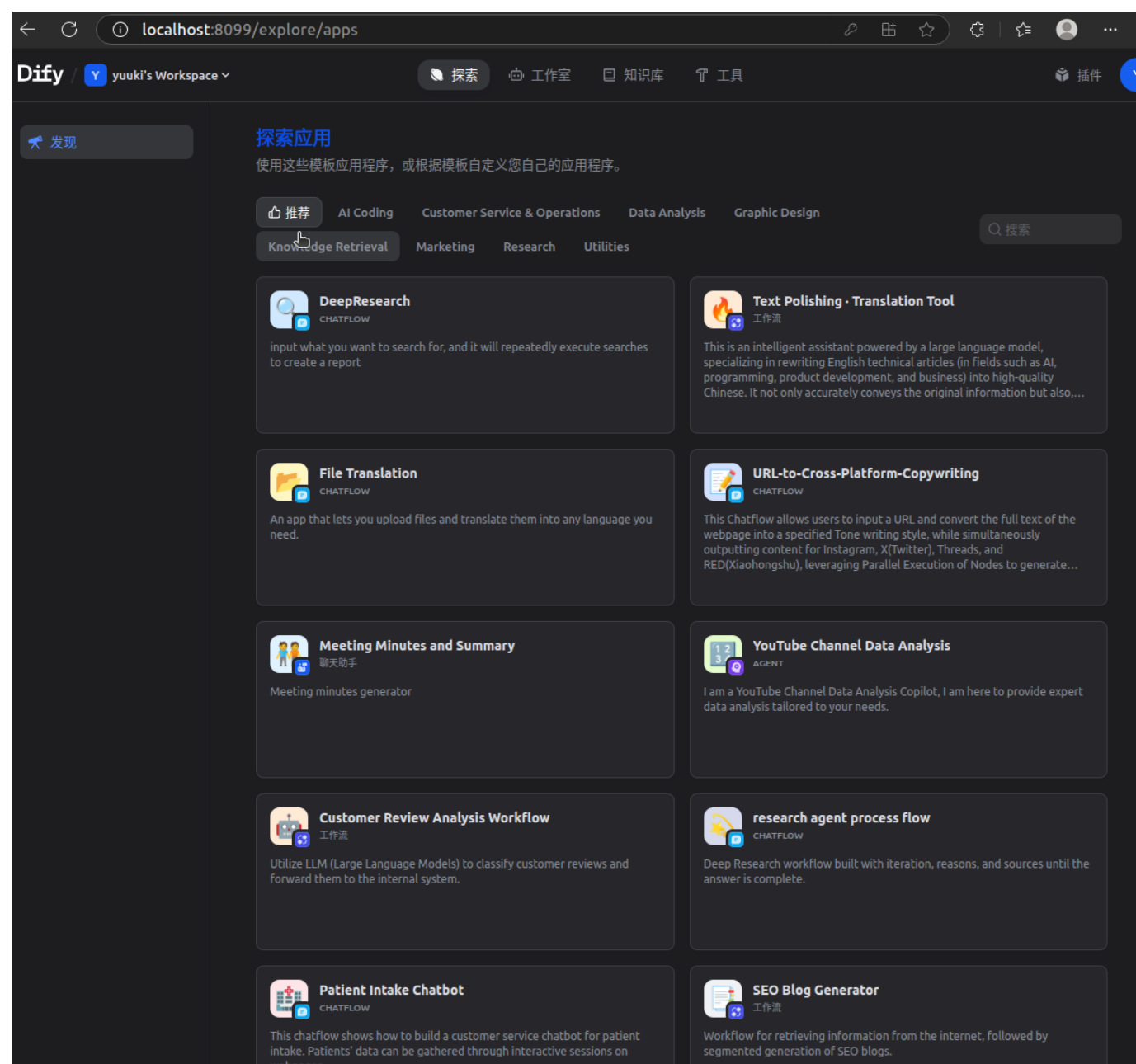
- 如果没有修改端口号打开这个，默认使用80端口号：<http://localhost/install>
- 如果修改了端口号，需要对应调整，比如修改为8099，那么地址为：<http://localhost:8099/install>
- 如果在远程服务器不是在本机，那么将localhost修改为服务器的IP地址

2. 首次启动需要初始化设置



The screenshot shows a web browser window with the address bar displaying 'localhost:8099/install'. The page title is 'Dify' and the main heading is '设置管理员账户' (Set Admin Account). Below the heading, there is a subtext: '管理员拥有的最大权限，可用于创建应用和管理 LLM 供应商等。' (Maximum permissions for the administrator, can be used to create applications and manage LLM providers, etc.). The form contains three input fields: '邮箱' (Email) with placeholder '输入邮箱地址', '用户名' (Username) with placeholder '输入用户名', and '密码' (Password) with placeholder '输入密码' and a password strength indicator. Below the password field, there is a note: '密码必须包含字母和数字，且长度不小于 8 位' (Password must contain letters and numbers, and length must be not less than 8 bits). A blue '设置' (Set) button is at the bottom of the form. At the very bottom, there is a link: '启动 Dify 社区版之前，请阅读 GitHub 上的 开源协议' (Before starting Dify Community Edition, please read the Open Source License on GitHub).

3. 与dify玩耍🎮



六、关闭服务

```
docker compose down
```

```
✓ Container docker-ssrf_proxy-1      Removed      11.2
✓ Container docker-worker-1         Remo...      5
3[+] Running 7/9
✓ Container docker-worker_beat-1     Removed      0.8s
✓ Container docker-ssrf_proxy-1     Removed      11.2
+ ] Running 13/13 docker-worker-1    Rem...      5.
✓ Container docker-worker_beat-1     Removed      0.8s
✓ Container docker-ssrf_proxy-1     Removed      11.2s
✓ Container docker-worker-1         R...        5.3s
✓ Container docker-weaviate-1        Removed      4.5s
✓ Container docker-plugin_daemon-1   Removed      11.1s
✓ Container docker-nginx-1           Re...       11.3s
✓ Container docker-sandbox-1         Removed      0.2s
✓ Container docker-web-1             Remo...     10.3s
✓ Container docker-api-1             Remo...     2.5s
✓ Container docker-redis-1           Re...       1.7s
✓ Container docker-db-1              Remov...    2.4s
✓ Network docker_ssrf_proxy_network  Removed      0.1s
✓ Network docker_default             Remo...     0.2s
```

七、其他

7.1 dify个性化设置

举个栗子：修改上传文件大小上限，默认是100MB。

- 在.env里的修改，在docker目录下打开，`nano .env`
- 使用快捷键`Ctrl + W`查找`NGINX_CLIENT_MAX_BODY_SIZE UPLOAD_FILE_SIZE_LIMIT`
- 修改后面的数值为自己需要的，比如1000MB，然后保存退出。

```
# Nginx performance tuning
NGINX_WORKER_PROCESSES=auto
NGINX_CLIENT_MAX_BODY_SIZE=100M
NGINX_KEEPALIVE_TIMEOUT=65
```

- 同样要在`docker-compose.yaml`里对应配置处修改。

.env.example 是 Docker 环境中提供的示例配置文件，可以按需修改。 .env 文件中的配置可以用来控制应用的行为，但它本身不会直接改变 docker-compose.yaml 中的容器配置。

docker-compose.yaml 文件用来配置和管理多容器 Docker 应用。在 docker-compose.yaml 文件中，我们通常会指定容器环境变量的值。虽然 .env 文件也定义了环境变量，但这些变量在 Docker Compose 中并不会自动生效，除非你在 docker-compose.yaml 文件中引用它们。

7.2 常见错误与处理

问题描述	可能原因	解决方案
80 端口被占用	系统中已有服务使用该端口	修改 .env 中的 EXPOSE_NGINX_PORT 为其他可用端口
容器无法启动	资源不足或配置错误	检查系统资源和 Docker 日志，调整 Docker 资源限制
数据库连接失败	数据库配置错误或服务未启动	检查数据库容器状态和连接配置
模型连接失败	API 密钥错误或网络问题	验证 API 密钥，检查网络连接和代理设置
文档上传失败	文件格式不支持或存储配置错误	检查文件格式，验证存储配置
向量检索不工作	Weaviate 服务问题或配置错误	检查 Weaviate 容器状态，验证向量库配置
访问页面 404 错误	Nginx 配置问题或服务未正确启动	检查 Nginx 配置和容器日志

7.3 dify升级教程

1. 备份现有配置与数据

```
cd docker
cp docker-compose.yaml docker-compose.yaml.$(date +%s).bak
```

2. dify文件夹拉取最新代码

```
git checkout main
git pull origin main
```

3. 关闭服务

```
docker compose down
```

4. 启动最新版本

```
docker compose up -d
```


5. 备份数据

```
tar -cvf volumes-$(date +%s).tgz volumes
```

这条命令会创建一个名为 `volumes-<时间戳>.tgz` 的压缩文件，其中 `<时间戳>` 会根据执行命令时的时间戳动态生成。压缩文件包含了 `volumes` 目录中的所有内容。

6. 验证运行状态

- 运行 `docker compose ps` 检查容器状态
- 访问网络界面，触发一次文档解析，尝试上传文件，调用知识库 API，确认新特性生效

参考教程：

[1] <https://cloud.tencent.com/developer/article/2514801>

[2] <https://zhuanlan.zhihu.com/p/30071849251>

[3] https://blog.csdn.net/Luo_Daimeng/article/details/148709871