English | 简体中文 | 繁体中文 | 日本語 | 한국어 | Bahasa Indonesia | Português (Brasil)

Document | Roadmap | Twitter | Discord | Demo

🍭 RAGFlow 是什么?

RAGFlow 是一款基于深度文档理解构建的开源 RAG（Retrieval-Augmented Generation）引擎。RAGFlow 可以为各种规模的企业及个人提供一套精简的 RAG 工作流程，结合大语言模型（LLM）针对用户各类不同的复杂格式数据提供可靠的问答以及有理有据的引用。

🎮 Demo 试用

请登录网址 https://demo.ragflow.io 试用 demo。

🔥 近期更新

- 2025-02-28 结合互联网搜索（Tavily），对于任意大模型实现类似 Deep Research 的推理功能.
- 2025-02-05 更新硅基流动的模型列表，增加了对 Deepseek-R1/DeepSeek-V3 的支持。
- 2025-01-26 优化知识图谱的提取和应用，提供了多种配置选择。
- 2024-12-18 升级了 DeepDoc 的文档布局分析模型。
- 2024-12-04 支持知识库的 Pagerank 分数。
- 2024-11-22 完善了 Agent 中的变量定义和使用。
- 2024-11-01 对解析后的 chunk 加入关键词抽取和相关问题生成以提高召回的准确度。
- 2024-08-22 支持用 RAG 技术实现从自然语言到 SQL 语句的转换。

🌟 关注项目

🌟 点击右上角的 Star 关注 RAGFlow，可以获取最新发布的实时通知!🌟

🌟 主要功能

🍭 “Quality in, quality out”

- 基于深度文档理解，能够从各类复杂格式的非结构化数据中提取真知灼见。
- 真正在无限上下文（token）的场景下快速完成大海捞针测试。

🍒 基于模板的文本切片

- 不仅仅是智能，更重要的是可控可解释。
- 多种文本模板可供选择

🍒 有理有据、最大程度降低幻觉（hallucination）

- 文本切片过程可视化，支持手动调整。
- 有理有据：答案提供关键引用的快照并支持追根溯源。

🍒 兼容各类异构数据源

- 支持丰富的文件类型，包括 Word 文档、PPT、excel 表格、txt 文件、图片、PDF、影印件、复印件、结构化数据、网页等。

🍒 全程无忧、自动化的 RAG 工作流

- 全面优化的 RAG 工作流可以支持从个人应用乃至超大型企业的各类生态系统。
- 大语言模型 LLM 以及向量模型均支持配置。
- 基于多路召回、融合重排序。
- 提供易用的 API，可以轻松集成到各类企业系统。

🍒 系统架构

🍒 快速开始

🍒 前提条件

- CPU >= 4 核
- RAM >= 16 GB
- Disk >= 50 GB
- Docker >= 24.0.0 & Docker Compose >= v2.26.1 > 如果你并没有在本机安装 Docker（Windows、Mac，或者 Linux）, 可以参考文档 Install Docker Engine 自行安装。

🍒 启动服务器

1. 确保 `vm.max_map_count` 不小于 262144：

> 如需确认 `vm.max_map_count` 的大小：
>
> ```
> $ sysctl vm.max_map_count
> ```
>
> 如果 `vm.max_map_count` 的值小于 262144，可以进行重置：
>
> ```
> # 这里我们设为 262144:
> $ sudo sysctl -w vm.max_map_count=262144
> ```
>
> 你的改动会在下次系统重启时被重置。如果希望做永久改动，还需要在 **/etc/sysctl.conf** 文件里把 `vm.max_map_count` 的值再相应更新一遍：
>
> ```
> vm.max_map_count=262144
> ```

2. 克隆仓库：

```
$ git clone https://github.com/infiniflow/ragflow.git
```

3. 进入 **docker** 文件夹，利用提前编译好的 Docker 镜像启动服务器：

> [!CAUTION] 请注意，目前官方提供的所有 Docker 镜像均基于 x86 架构构建，并不提供基于 ARM64 的 Docker 镜像。如果你的操作系统是 ARM64 架构，请参考这篇文档自行构建 Docker 镜像。

> 运行以下命令会自动下载 RAGFlow slim Docker 镜像 `v0.17.2-slim`。请参考下表查看不同 Docker 发行版的描述。如需下载不同于 `v0.17.2-slim` 的 Docker 镜像，请在运行 `docker compose` 启动服务之前先更新 **docker/.env** 文件内的 `RAGFLOW_IMAGE` 变量。比如，你可以通过设置 `RAGFLOW_IMAGE=infiniflow/ragflow:v0.17.2` 来下载 RAGFlow 镜像的 `v0.17.2` 完整发行版。

```
$ cd ragflow/docker
# Use CPU for embedding and DeepDoc tasks:
$ docker compose -f docker-compose.yml up -d

# To use GPU to accelerate embedding and DeepDoc tasks:
# docker compose -f docker-compose-gpu.yml up -d
```

RAGFlow image tag	Image size (GB)	Has embedding models?	Stable?
v0.17.2	≈9	:heavy_check_mark:	Stable release
v0.17.2-slim	≈2	☒	Stable release
nightly	≈9	:heavy_check_mark:	*Unstable* nightly build
nightly-slim	≈2	☒	*Unstable* nightly build

> [!TIP] 如果你遇到 Docker 镜像拉不下来的问题，可以在 **docker/.env** 文件内根据变量 `RAGFLOW_IMAGE` 的注释提示选择华为云或者阿里云的相应镜像。
>
> - 华为云镜像名：`swr.cn-north-4.myhuaweicloud.com/infiniflow/ragflow`

> - 阿里云镜像名：registry.cn-hangzhou.aliyuncs.com/infiniflow/ragflow

4. 服务器启动成功后再次确认服务器状态：

   ```
   $ docker logs -f ragflow-server
   ```

   出现以下界面提示说明服务器启动成功：

   ```
        ____   ___    ______ ______ __
       / __ \ /   |  / ____// ____// /____  _      __
      / /_/ // /| | / / __ / /_   / // __ \| | /| / /
     / _, _// ___ |/ /_/ // __/  / // /_/ /| |/ |/ /
    /_/ |_|/_/  |_|\____//_/    /_/ \____/ |__/|__/

    * Running on all addresses (0.0.0.0)
   ```

   > 如果您在没有看到上面的提示信息出来之前，就尝试登录 RAGFlow，你的浏览器有可能会提示 network anormal 或 网络异常。

5. 在你的浏览器中输入你的服务器对应的 IP 地址并登录 RAGFlow。> 上面这个例子中，您只需输入 http://IP_OF_YOUR_MACHINE 即可：未改动过配置则无需输入端口（默认的 HTTP 服务端口 80）。

6. 在 service_conf.yaml.template 文件的 user_default_llm 栏配置 LLM factory，并在 API_KEY 栏填写和你选择的大模型相对应的 API key。

   > 详见 llm_api_key_setup。

   好戏开始，接着奏乐接着舞！

⊠ 系统配置

系统配置涉及以下三份文件：

- .env：存放一些基本的系统环境变量，比如 SVR_HTTP_PORT、MYSQL_PASSWORD、MINIO_PASSWORD 等。
- service_conf.yaml.template：配置各类后台服务。
- docker-compose.yml：系统依赖该文件完成启动。

请务必确保 .env 文件中的变量设置与 service_conf.yaml.template 文件中的配置保持一致！

如果不能访问镜像站点 hub.docker.com 或者模型站点 huggingface.co，请按照 .env 注释修改 RAGFLOW_IMAGE 和 HF_ENDPOINT。

> ./docker/README 解释了 service_conf.yaml.template 用到的环境变量设置和服务配置。

如需更新默认的 HTTP 服务端口 (80)，可以在 docker-compose.yml 文件中将配置 80:80 改为 <YOUR_SERVING_PORT>:80。

> 所有系统配置都需要通过系统重启生效：
>
> ```
> $ docker compose -f docker-compose.yml up -d
> ```

把文档引擎从 Elasticsearch 切换成为 Infinity

RAGFlow 默认使用 Elasticsearch 存储文本和向量数据. 如果要切换为 Infinity, 可以按照下面步骤进行:

1. 停止所有容器运行:

   ```
   $ docker compose -f docker/docker-compose.yml down -v
   ```

   Note: `-v` 将会删除 docker 容器的 volumes，已有的数据会被清空。

2. 设置 **docker/.env** 目录中的 `DOC_ENGINE` 为 `infinity`.

3. 启动容器:

   ```
   $ docker compose -f docker-compose.yml up -d
   ```

> [!WARNING] Infinity 目前官方并未正式支持在 Linux/arm64 架构下的机器上运行.

🐬 源码编译 Docker 镜像（不含 embedding 模型）

本 Docker 镜像大小约 2 GB 左右并且依赖外部的大模型和 embedding 服务。

```
git clone https://github.com/infiniflow/ragflow.git
cd ragflow/
docker build --build-arg LIGHTEN=1 --build-arg NEED_MIRROR=1 -f Dockerfile -t infiniflow/ragflow:nightly-slim .
```

🐬 源码编译 Docker 镜像（包含 embedding 模型）

本 Docker 大小约 9 GB 左右。由于已包含 embedding 模型，所以只需依赖外部的大模型服务即可。

```
git clone https://github.com/infiniflow/ragflow.git
cd ragflow/
docker build --build-arg NEED_MIRROR=1 -f Dockerfile -t infiniflow/ragflow:nightly .
```

🐬 以源代码启动服务

1. 安装 uv。如已经安装，可跳过本步骤：

   ```
   pipx install uv
   export UV_INDEX=https://mirrors.aliyun.com/pypi/simple
   ```

2. 下载源代码并安装 Python 依赖：

```
git clone https://github.com/infiniflow/ragflow.git
cd ragflow/
uv sync --python 3.10 --all-extras # install RAGFlow dependent python modules
```

3. 通过 Docker Compose 启动依赖的服务（MinIO, Elasticsearch, Redis, and MySQL）：

```
docker compose -f docker/docker-compose-base.yml up -d
```

在 `/etc/hosts` 中添加以下代码，将 **conf/service_conf.yaml** 文件中的所有 host 地址都解析为 `127.0.0.1`：

```
127.0.0.1       es01 infinity mysql minio redis
```

4. 如果无法访问 HuggingFace，可以把环境变量 `HF_ENDPOINT` 设成相应的镜像站点：

```
export HF_ENDPOINT=https://hf-mirror.com
```

5. 启动后端服务：

```
source .venv/bin/activate
export PYTHONPATH=$(pwd)
bash docker/launch_backend_service.sh
```

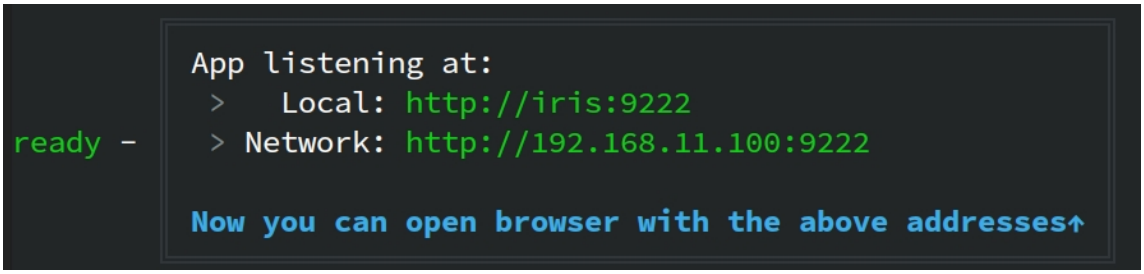
6. 安装前端依赖：

```
cd web
npm install
```

7. 启动前端服务：

```
npm run dev
```

以下界面说明系统已经成功启动：



📚 技术文档

- Quickstart
- User guide
- References
- FAQ

🗺 路线图

详见 RAGFlow Roadmap 2025 。

🏄 开源社区

- Discord
- Twitter
- GitHub Discussions

🙌 贡献指南

RAGFlow 只有通过开源协作才能蓬勃发展。秉持这一精神，我们欢迎来自社区的各种贡献。如果您有意参与其中，请查阅我们的 贡献者指南 。

🤝 商务合作

- 预约咨询

👥 加入社区

扫二维码添加 RAGFlow 小助手，进 RAGFlow 交流群。