向量数据库

实践教程

腾讯云

文档目录

实践教程
基于向量数据库混合检索 + Dify 构建知识库

最近更新时间：2025-04-22 17:51:02

背景信息

Dify 是一款面向开发者与企业的 AI 应用开发平台，致力于简化大语言模型（LLM）集成与开发流程，助力用户高效构建智能化应用。腾讯云向量数据库作为其官方支持的向量数据库组件，凭借高效的 向量检索与稀疏向量 能力，为 Dify 平台上的 RAG（检索增强生成）应用提供底层支持，显著提升语义理解与检索精度。用户可将文档、图表等数据向量化存储，结合 Dify 的 LLM 交互逻辑，直接搭建智能知识库，实现精准问答与知识检索。

准备工作

1. 选型并购买向量数据库实例，具体操作，请参见 购买实例 。
2. 在向量数据库中，创建数据库。具体操作，请参见 create 。
3. 客户端运行环境准备。

类别	要求
地域与网络环境	• 使用腾讯云**外网**访问，需手动配置白名单，开通外网功能。具体操作，请参见 开启外网访问 。 • 使用腾讯云**内网**方式访问，确保购买的 CVM 所处地域与向量数据库为同地域，且 VPC 和安全组策略能正常连通向量数据库实例。
客户端	• 使用腾讯云**外网**方式，可使用本地客户端操作环境即可。 • 使用腾讯云**内网**方式，推荐选择 云服务器 CVM ，如 SA3.LARGE8、SA3.2XLARGE16等规格。

4. 登录客户端运行环境，安装 Docker。具体操作，请参见 Docker Compose 部署 。若选择 云服务器 CVM ，请参见 搭建 Docker ，快速安装与 CVM 操作系统版本适配的 Docker 环境。
5. 克隆 Dify 源代码至本地环境。具体方式，请参见 克隆 Dify 代码仓库 。
6. 准备知识库数据源文本。本示例以腾讯云向量数据库的官网文档为例搭建，存放知识库文件 demo_file1.pdf 与 demo_file2.md 于本地。

快速搭建

1. 登录客户端环境，进入克隆的 dify 目录的 docker 文件。
2. 执行 `cp .env.example .env` 拷贝一份配置文件。
3. 使用 vim .env 打开配置文件，将配置文件中的 **VECTOR_STORE** 修改为 **tencent**，并配置向量数据库信息。

```
# tencent vector configurations, only available when VECTOR_STORE is `tencent`
TENCENT_VECTOR_DB_URL=http://172.19.0.8
TENCENT_VECTOR_DB_API_KEY=[redacted]
TENCENT_VECTOR_DB_TIMEOUT=30
TENCENT_VECTOR_DB_USERNAME=root
TENCENT_VECTOR_DB_DATABASE=dify
TENCENT_VECTOR_DB_SHARD=1
TENCENT_VECTOR_DB_REPLICAS=2
TENCENT_VECTOR_DB_ENABLE_HYBRID_SEARCH=true
```

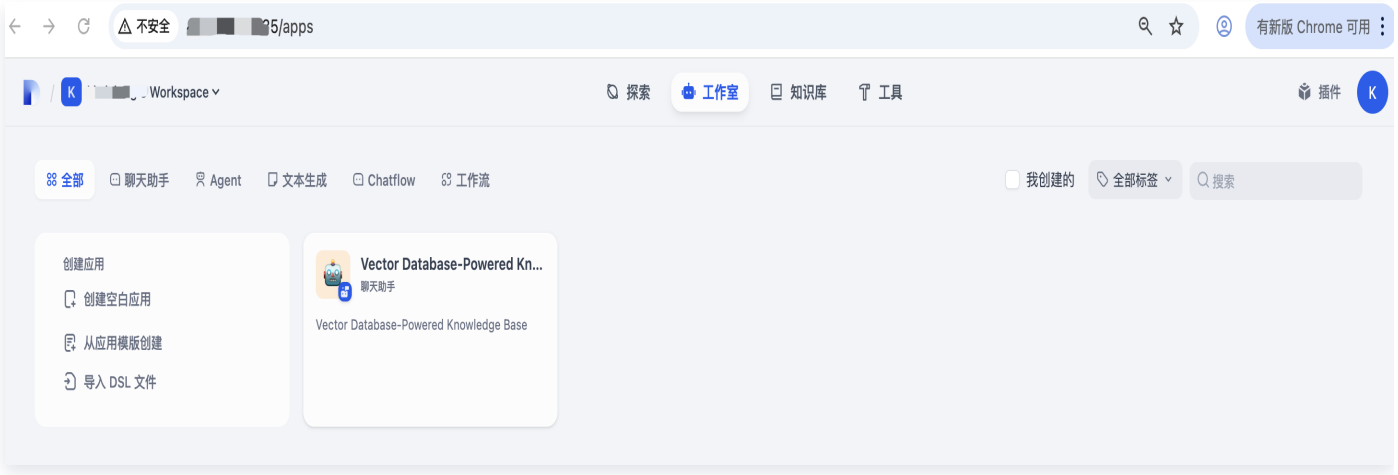
配置参数	参数解释	配置说明
TENCENT_VECTOR_DB_URL	向量数据库实例的内网地址或外网地址。建议使用内网方式。	请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息 。
TENCENT_VECTOR_DB_API_KEY	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理 。
TENCENT_VECTOR_DB_TIMEOUT	连接超时时间。	-
TENCENT_VECTOR_DB_USERNAME	指定访问向量数据库的用户名。	具体信息，请参见 账号与权限管理 。
TENCENT_VECTOR_DB_DATABASE	指定已准备的向量数据库名。	Database 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。 • 长度要求：[1,128]。

TENCENT_VECTOR_DB_SHARD	指定创建集合所需的分片数量。	• 取值类型：uint64。
		• 取值范围：[1,100]。例如：5。
		• 配置建议：在搜索时，全部分片是并发执行的，分片数量越多，平均耗时越低，但是过多的分片会带来额外开销而影响性能。
		  ○ 单分片数据量建议控制在300万以内，例如500万向量，可设置2个分片。
		  ○ 如果数据量小于300万，建议使用1分片。系统对1分片有特定优化，可显著提升性能。

TENCENT_VECTOR_DB_REPLICAS	指定创建集合的副本数量。	• 取值类型：uint64。 • 取值范围如下所示。搜索请求量越高的索引，建议设置越多的副本

		数，避免负载不均衡。 ○ 单可用区实例：0。 ○ 两可用区实例：[1,节点数-1]。 ○ 三可用区实例：[2,节点数-1]。
TENCENT_VECTOR_DB_ENABLE_HYBRID_SEARCH	指定是否开启混合检索。	true：开启。

4. 在浏览器地址栏中，输入部署的服务器的 IP 地址，进入 Dify 平台并完成账号创建，如下图所示。

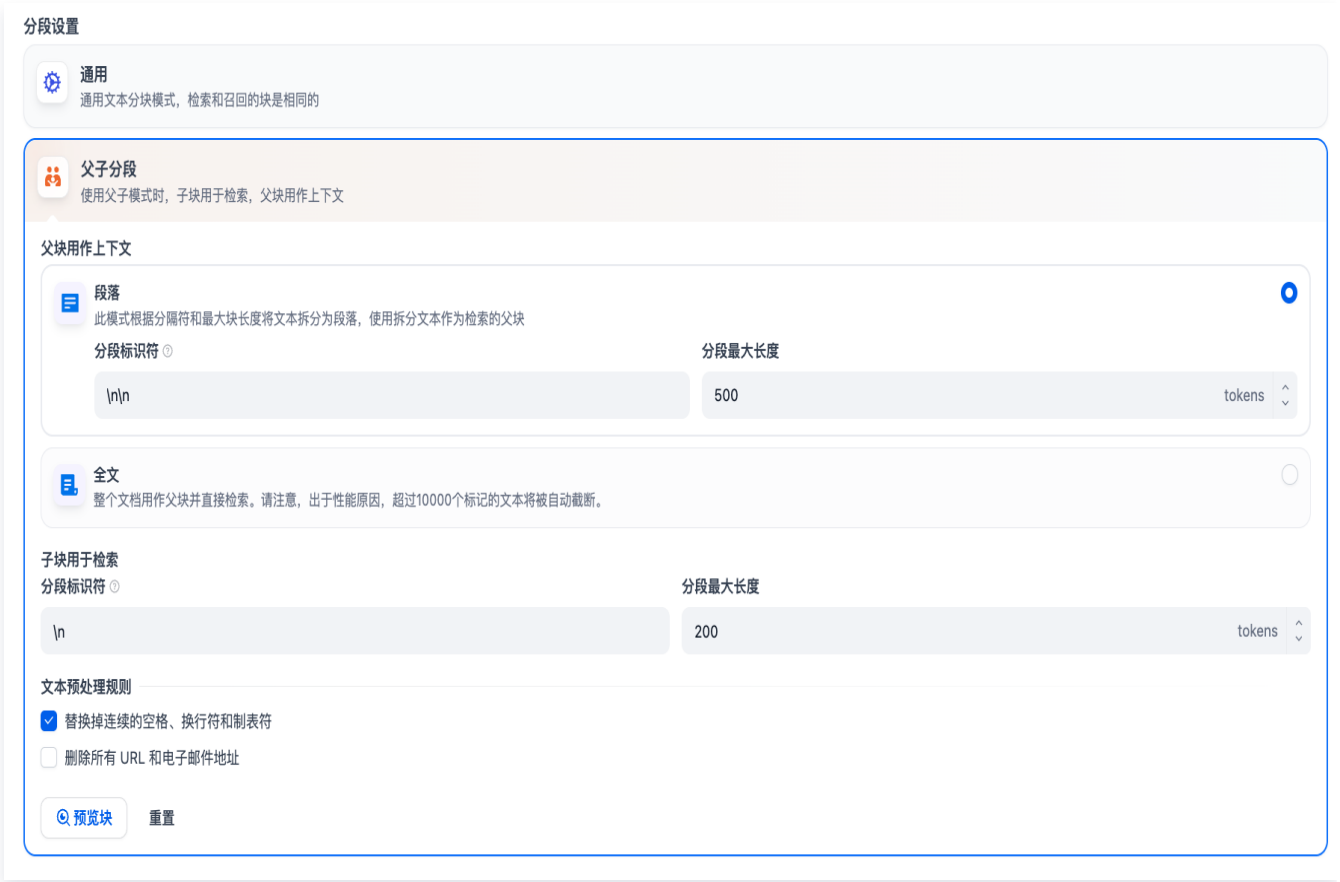


5. 在 Dify 工作空间上方，选择**知识库**，并单击**创建知识库**，如下所示。

6. 在创建知识库的导航页面，上传知识库的数据源文本，单击**下一步**。

7. 在**文本分段与清洗**指引页面，设置分段规则，Embedding 模型以及检索方式。

   7.1 在**分段设置**区域，选择**父子分段**，保持默认配置。



   7.2 在**索引方式**区域，选择**高质量**，在 **Embedding 模型**的下拉列表，选择嵌入模型。

> **说明：**

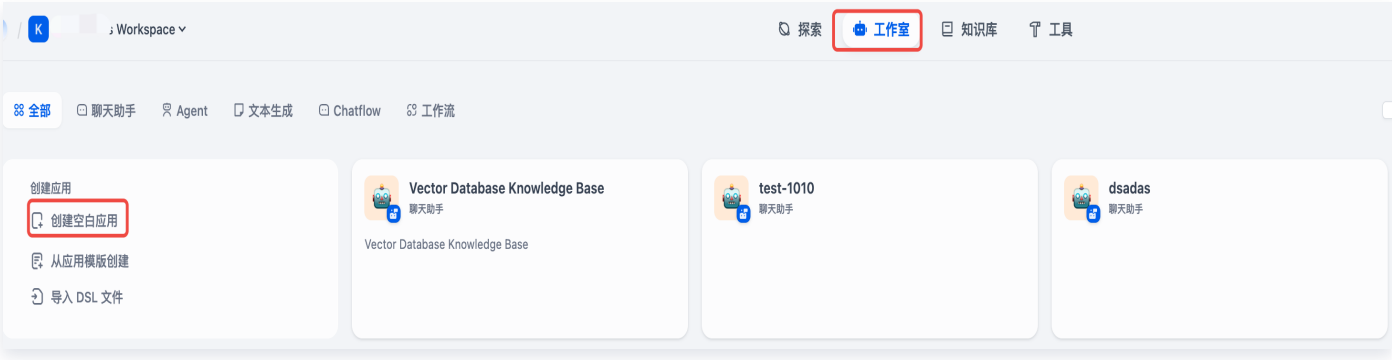
> 第一次使用，在 Embedding 模型的下拉列表，单击**模型设置**，可在模型供应商页面，选择所需的模型并安装。

7.3 在**检索设置**区域，选择**混合检索**，并根据页面指引，选择检索结果排序方式。

> **说明：**
> Dify 已支持使用腾讯云向量数据库自带的稠密向量 + 稀疏向量 实现混合检索，通过双路检索的方式，整体提升 RAG 应用的效果，实现精准问答与知识检索。

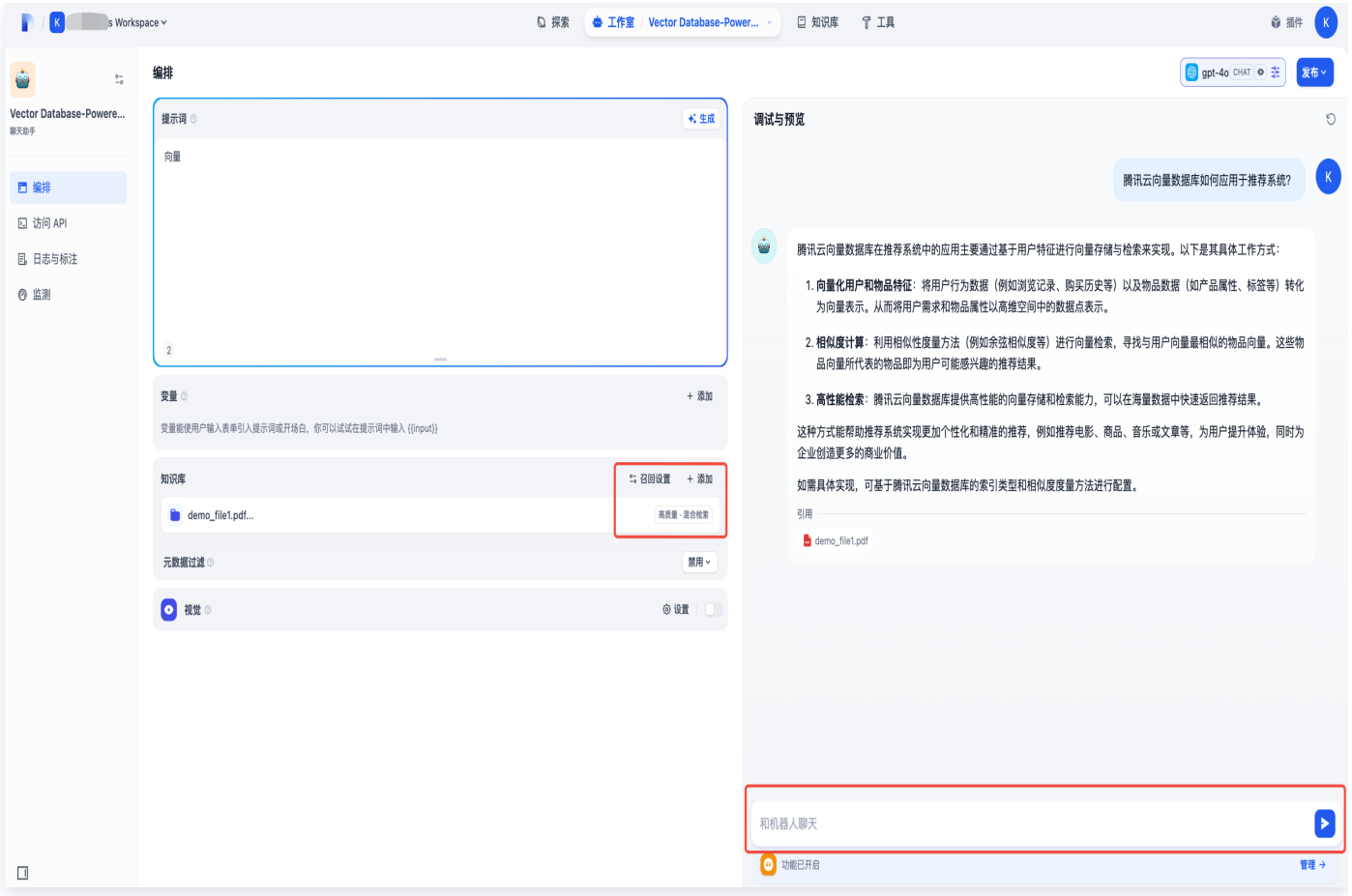
8. 单击**保存并处理**，在如下图所示页面，知识库默认以导入的数据源文件命名，等待数据源文件嵌入完成，则知识库创建成功。

9. 在Dify工作空间上方，选择**工作室**，并单击**创建空白应用**，单击**创建**。如下图所示。



10. 在**创建空白应用**页面，选择应用类型为**聊天助手**，并在**应用名称 & 图标**的输入框，输入应用名称，如下所示。

11. 在创建的工作室中，在**知识库**区域，单击**添加**，导入新建的知识库（本示例以腾讯云向量数据库的官网文档为例搭建，知识库文件 demo_file1.pdf ），便可以在下方输入框，输入具体问题进行知识问答，如下图所示。


Workspace
探索
工作室
Vector Database-Power...
知识库
工具
插件
编排
gpt-4o CHAT
发布
Vector Database-Powere...
聊天助手
编排
访问 API
日志与标注
监测
提示词
生成
向量
变量
+ 添加
变量能使用户输入表单引入提示词或开场白，你可以试试在提示词中输入 {{input}}
知识库
召回设置
+ 添加
demo_file1.pdf...
高质量 · 混合检索
元数据过滤
禁用
视觉
设置
调试与预览
腾讯云向量数据库如何应用于推荐系统?
腾讯云向量数据库在推荐系统中的应用主要通过基于用户特征进行向量存储与检索来实现。以下是其具体工作方式:
1. 向量化用户和物品特征：将用户行为数据（例如浏览记录、购买历史等）以及物品数据（如产品属性、标签等）转化为向量表示。从而将用户需求和物品属性以高维空间中的数据点表示。
2. 相似度计算：利用相似性度量方法（例如余弦相似度等）进行向量检索，寻找与用户向量最相似的物品向量。这些物品向量所代表的物品即为用户可能感兴趣的推荐结果。
3. 高性能检索：腾讯云向量数据库提供高性能的向量存储和检索能力，可以在海量数据中快速返回推荐结果。
这种方式能帮助推荐系统实现更加个性化和精准的推荐，例如推荐电影、商品、音乐或文章等，为用户提升体验，同时为企业创造更多的商业价值。
如需具体实现，可基于腾讯云向量数据库的索引类型和相似度度量方法进行配置。
引用
demo_file1.pdf
和机器人聊天
功能已开启
管理

快速搭建 RAG 应用实践
总览

最近更新时间：2025-02-12 12:02:02

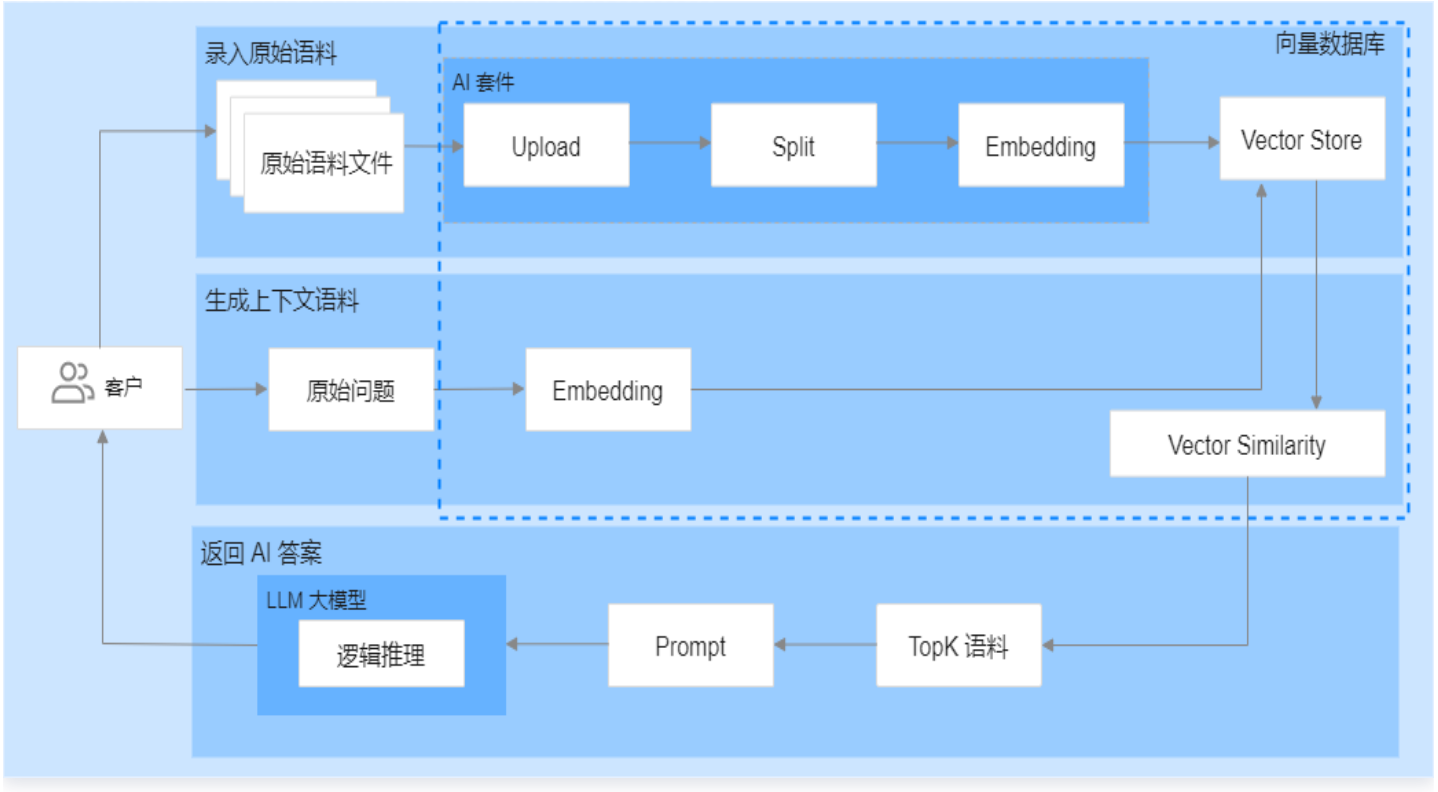
本教程演示如何使用腾讯云向量数据库（Tencent Cloud VectorDB）AI 套件一站式文档检索方案进行相似性检索，结合 LLM 大模型，构建专属知识库问答服务的方法。

背景信息

大语言模型（LLM）作为自然语言处理（NLP）服务领域的核心技术，具有丰富的服务能力，但其训练数据主要涵盖普适知识和一般常识性知识，在处理特定领域（如医疗保健、金融、科技等）的知识时存在局限性。为了扩展 LLM 的知识范畴，使其能够理解并获取训练范围之外的特定领域知识，可以通过特定的提示构造来引导 LLM 在回答特定领域问题时理解意图，并根据注入的领域知识做出回答。
检索增强生成（RAG）技术融合了信息检索和语言生成模型，通过检索外部知识库中的相关信息，并将其作为提示输入给 LLM，以增强模型的逻辑推理和生成能力，从而返回更准确、全面的知识型答案。腾讯云向量数据库的 AI 套件能够解析和检索多种文本文件，包括难以处理的 PDF 图文内容，有效支持构建基于 RAG 的高质量图文知识库应用，帮助客户快速实现信息的深入理解和高效检索。

实现思路

基于腾讯云向量数据库的 AI 套件对知识库文件进行上传、拆分和向量化，将文件和向量化数据存储于数据库中。借助腾讯云向量数据库的 Embedding 功能，将用户提出的问题转化为向量，在向量数据库中进行相似性检索，找出与问题相似度最高的语料。最后，将用户提出的问题和相似性检索的上下文语料进行组装，送入 LLM 大模型，进行逻辑匹配，生成问题答案。实现方案，如下图所示。



腾讯云向量数据库的 AI 套件提供了一套完整的一站式向量检索方案，包括数据切割和 Embedding 服务，无需自行编写拆分和向量化相关代码，减少了算法工程投入，极大简化了整个实现过程，降低了业务接入门槛。同时，相似性检索的上下文语料可以更有效地指导 LLM 大模型生成更精准的答案，进一步提升回答的准确性。并且，腾讯云向量数据库采用了灵活的存储策略，可根据实际变化的需求，及时优化更新知识库，保证了系统的稳定性。

LLM 大模型

腾讯云向量数据库（Tencent Cloud VectorDB）分别结合 DeepSeek、混元和百川等模型可以高效地搭建 RAG 知识问答系统。每种模型都有其独特的优势，用户可以根据具体业务需求选择合适的模型进行集成。

大模型	说明
向量数据库 + DeepSeek	基于 Gradio 框架，依赖 腾讯云 TI 平台 一键部署 DeepSeek 系列模型，结合向量数据库，快速搭建基于 RAG 知识问答系统。
向量数据库 + 混元	基于 Gradio 框架，依赖 腾讯混元大模型 的强大能力，提供一套界面化、直观且完整的知识问答系统搭建方案。
向量数据库 + 百川	基于 Python 代码，通过调用百川的 HTTP API 接口，并结合向量数据库，搭建一套在命令窗口内进行交互的 RAG 知识问答系统。

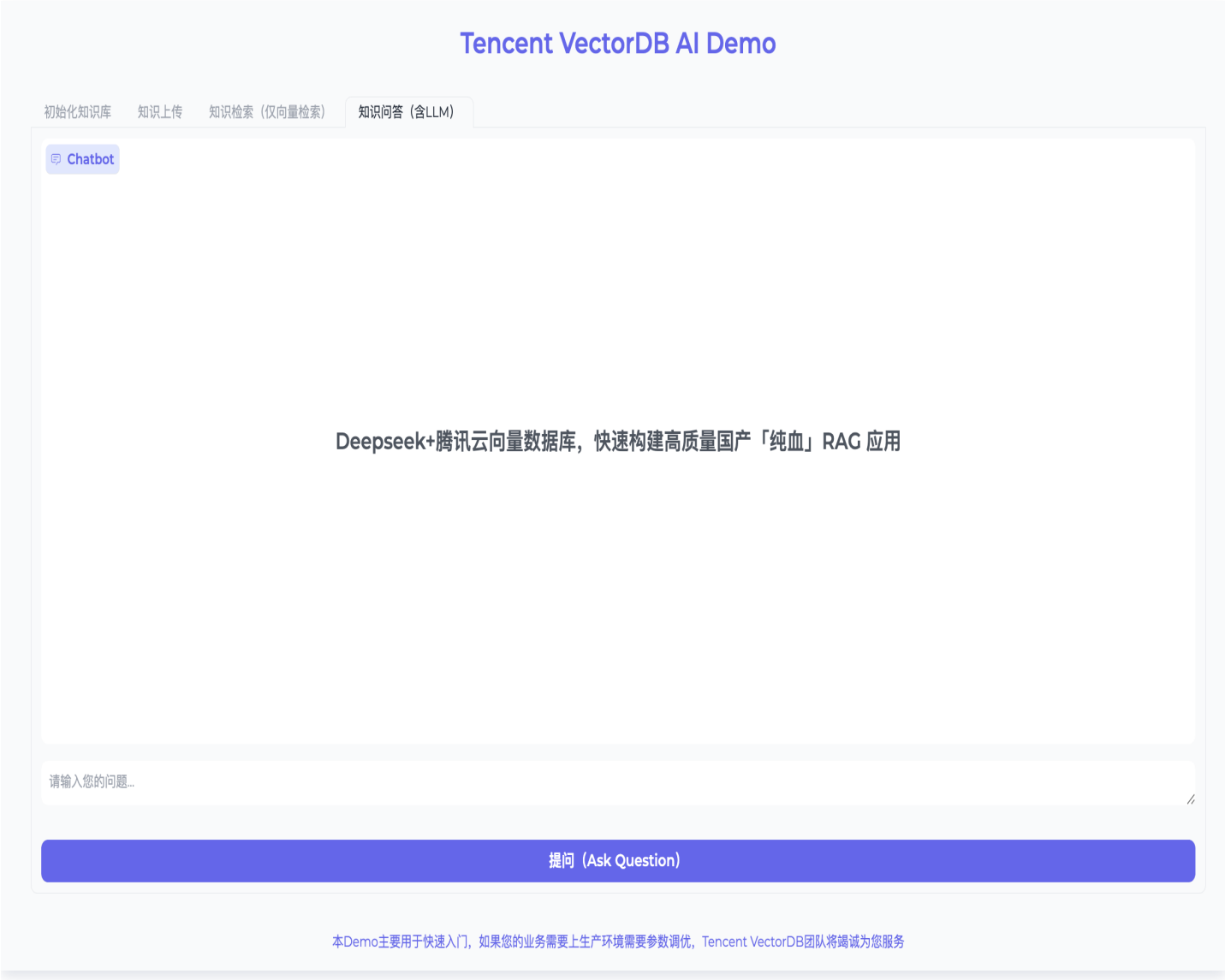
向量数据库 + DeepSeek 大语言模型

最近更新时间：2025-02-21 11:07:12

背景信息

随着 DeepSeek 大语言模型在全球范围内的迅猛发展和广泛应用，其热度持续攀升，成为当前人工智能领域的焦点。为了满足市场对高效部署和应用的需求，腾讯云 TI 平台 提供了快速部署 DeepSeek 系列模型的方案。通过将腾讯云向量数据库与 DeepSeek 深度结合，用户可以高效搭建基于 DeepSeek 的 RAG 知识问答系统，实现高效的检索和生成能力，同时显著降低部署成本和时间。

效果预览



准备工作

1. 选型并购买向量数据库实例，具体操作，请参见 购买实例 。

2. 客户端运行环境准备。

类别	要求
地域与网络环境	• 使用腾讯云**外网**访问，需手动配置白名单，开通外网功能。具体操作，请参见 开启外网访问 。 • 使用腾讯云**内网**方式访问，确保购买的 CVM 所处地域与向量数据库为同地域，且 VPC 和安全组策略能正常连通向量数据库实例。
客户端	• 使用腾讯云**外网**方式，可使用本地客户端操作环境即可。 • 使用腾讯云**内网**方式，推荐选择 云服务器 CVM ，如 SA3.LARGE8、SA3.2XLARGE16等规格。
Python 环境依赖	• 推荐 Python 版本 >= 3.8。 • 安装 Python SDK 执行 `pip3 install tcvectordb` 命令，可直接安装最新版本。

3. 下载 document_search_demo_20250207.zip 压缩包，并将其上传于客户端运行环境。
4. 准备知识库文件，本示例以腾讯云向量数据库的官网文档为例搭建，存放知识库文件 demo_file1.pdf 与 demo_file2.md 于本地。

> **说明：**
> 1. 当前支持导入数据库的**文件类型**包含： Markdown、PDF、Word、PPT。
> 2. Markdown 类型文件最大限制为1MB，其余类型最大限制为10MB。若文件超过10MB，请 提交工单 处理。

5. 在 腾讯云 TI 平台 部署 DeepSeek 模型。具体操作，请参见 快速部署和体验 DeepSeek 系列模型 。部署完成后，待服务状态为**运行中**，在**服务调用**页面，获取**调用地址**与 **AuthToken**。

> **说明：**
> 若不开通 DeepSeek 大模型，不阻塞知识库搭建，仅支持从腾讯云向量数据库搜索到的图文并茂的知识点，而不支持经过大模型润色加工的知识答案。

快速搭建

1. 登录客户端运行环境，执行 `pip3 install tcvectordb` ，安装向量数据库最新的 Python SDK。
2. 使用 unzip 命令，解压 document_search_demo_20250207.zip 压缩包。
3. 进入压缩包解压后的文件夹，执行 `pip3 install -r requirements.txt` 命令，安装 SDK 通用依赖，DeepSeek 大模型 SDK 以及相关依赖组件。
4. 使用 `vim conf/config.ini` 命令，根据参数注释修改相关配置并保存，如下所示。

```
[vector_db]
address=http://bj-vdb-qpvr****.sql.tencentcdb.com:8100
key=****************************
ai_db=test_ai_db
ai_collection=test_ai_collection

[model]
address=https://ms-d6b6rhnl-1000********.gw.ap-
beijing.ti.tencentcs.com/ms-d6b6****

name=ms-d6b6****
# 如果在 TiOne 平台开启了鉴权，则填入key
key***************

[server]
name=127.0.0.1
```

```
port=7869
```

配置项	参数名	参数含义	配置说明
[vector_db]	address	向量数据库实例的内网地址或外网地址。建议使用内网方式。	请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息 。
	key	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理 。
	ai_db	AI 类数据库名。	Database 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。 • 长度要求：[1,128]。
	ai_collection	AI 数据库集合视图名。	CollectionView 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。 • 长度要求：[1,128]。
[model]	address	DeepSeek 模型的 API 服务域名	在 腾讯云 TI 平台，部署 DeepSeek 之后，在**服务调用**页面，获取调用地址。
	name	DeepSeek 模型的 ID。	在 腾讯云 TI 平台，部署 DeepSeek 之后，随机分配的唯一标识 ID，在**服务管理**页面直接复制模型 ID。 
	key	DeepSeek 模型鉴权 Token	在 腾讯云 TI 平台，部署 DeepSeek 之后，在**服务调用**页面，开启鉴权，获取用于身份验证的 Token。
[server]	name	客户端运行环境 IP 地址。	-
	port	运行环境分配的端口	注意避免端口冲突。

5. 执行 `python3 main.py` ，运行脚本，生成知识问答前端访问链接，如下图所示。

```
init configs conf/config.ini
Try to connect vector db http:/                    tencentclb.com
Running on local URL:  http://127.0.0.1:7869
```

6. 复制 **Running on local URL** 后面的访问链接，在浏览器访问链接，显示 **Tencent VectorDB AI Demo** 配置向导，如下图所示。

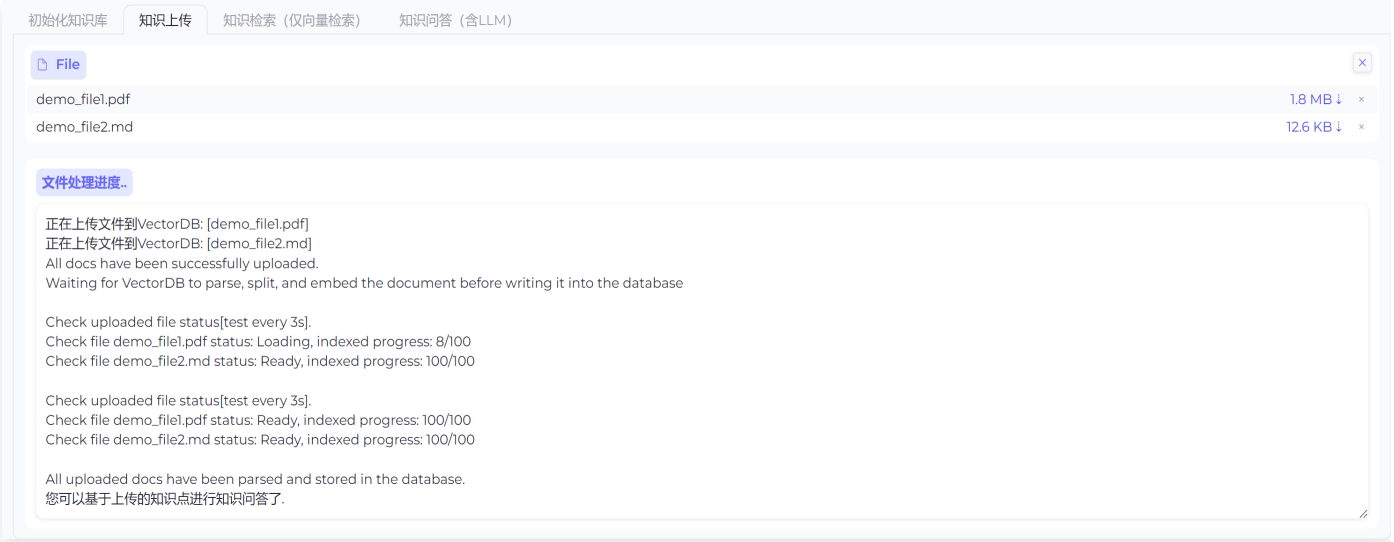
> **注意：**
> 在浏览器访问知识问答应用，切勿关闭后端运行环境正在运行的进程。

7. 单击下方**开始初始化**，脚本将自动创建向量数据库与集合，界面显示初始化进度，等待任务完成，如下图所示。

8. 单击**知识上传**，将已准备的知识库文件 demo_file1.pdf 与 demo_file2.md 一起直接拖放或者上传至应用程序，程序将自动将文件向量化写入向量数据库，如下图所示。

> **说明：**

支持批量上传文件，不支持上传压缩包。



9. 等待文档上传完成，便可以进行知识检索或问答。

说明：

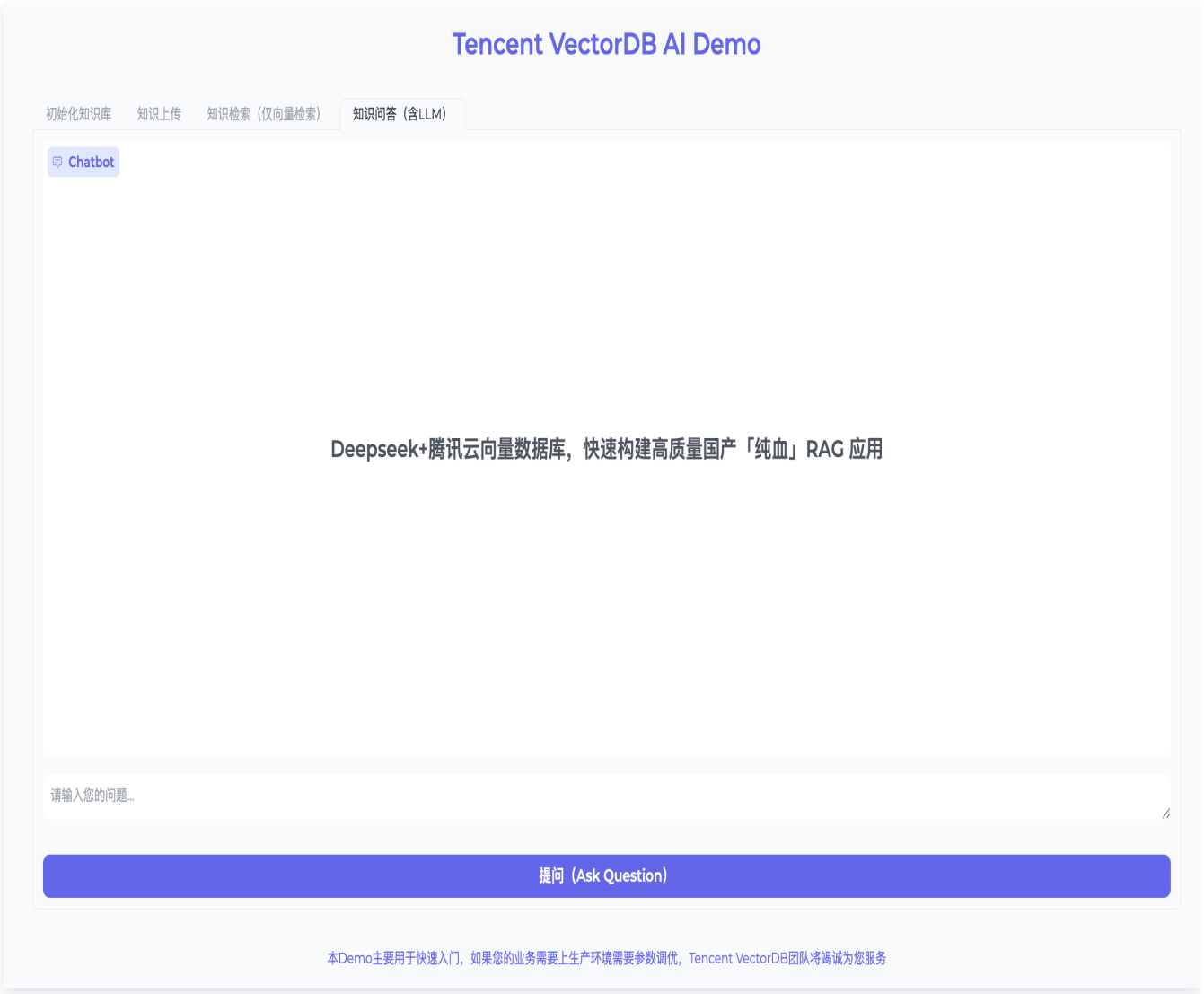
针对知识库文件，可参考如下问题进行问答和检索体验：

- 问题1: 腾讯云向量数据库如何构建大模型知识库?
- 问题2: 腾讯云向量数据库如何应用于推荐系统?
- 问题3: 腾讯云向量数据库如何应用于文本或者图片检索场景?
- 问题4: 腾讯会议如何快速发起会议?
- 问题5: 腾讯会议如何加入会议?
- 问题6: 腾讯会议如何取消会议?

- **知识检索**：单击**知识检索（仅向量检索）**，在下方**请输入您要检索的问题**处，输入需要检索的问题，单击**检索知识**，即可检索到从腾讯云向量数据库中搜索到的图文并茂的知识点。

- **知识问答**：单击**知识问答（含LLM）**，在下方**请输入您的问题**处，输入需要查阅的问题，单击**提问**，即可看到经过 DeepSeek 大模型润色后的答案。

> ⚠ **注意：**
> 若需更换其他系列模型，请在工程 models 文件夹下添加模型，同步修改 conf 下配置文件中模型的信息。


Tencent VectorDB AI Demo
初始化知识库
知识上传
知识检索（仅向量检索）
知识问答（含LLM）
Chatbot
Deepseek+腾讯云向量数据库，快速构建高质量国产「纯血」RAG 应用
请输入您的问题...
提问（Ask Question）
本Demo主要用于快速入门，如果您的业务需要上生产环境需要参数调优，Tencent VectorDB团队将竭诚为您服务

向量数据库 + 混元大模型

最近更新时间：2025-07-09 15:17:22

背景信息

本文基于腾讯云向量数据库与 腾讯混元大模型，提供一套直观且完整的知识问答系统搭建方案。相关 AI 套件具体信息，请参见 AI 套件。

准备工作

1. 选型并购买向量数据库实例，具体操作，请参见 购买实例。
2. 客户端运行环境准备。

类别	要求
地域与网络环境	• 使用腾讯云**外网**访问，需手动配置白名单，开通外网功能。具体操作，请参见 开启外网访问。 • 使用腾讯云**内网**方式访问，确保购买的 CVM 所处地域与向量数据库为同地域，且 VPC 和安全组策略能正常连通向量数据库实例。
客户端	• 使用腾讯云**外网**方式，可使用本地客户端操作环境即可。 • 使用腾讯云**内网**方式，推荐选择 云服务器 CVM，如 SA3.LARGE8、SA3.2XLARGE16 等规格。
Python 环境依赖	• 推荐 Python 版本 >= 3.8。 • 安装 Python SDK 执行 `pip3 install tcvectordb` 命令，可直接安装最新版本。

3. 下载 document_search_demo_20241114.zip 压缩包，并将其上传于客户端运行环境。
4. 准备知识库文件，本示例以腾讯云向量数据库的官网文档为例搭建，存放知识库文件 demo_file1.pdf 与 demo_file2.md 于本地。

> **说明：**
> 1. 当前支持导入数据库的**文件类型**包含： Markdown、PDF、Word、PPT。
> 2. Markdown 类型文件最大限制为1MB，其余类型最大限制为10MB。若文件超过10MB，请 提交工单 处理。

5. （可选）开通腾讯混元大模型 服务，并获取其访问密钥 secret_id 与 secret_key。

> **说明：**

> 若不开通混元大模型，不阻塞知识库搭建，仅支持从腾讯云向量数据库搜索到的图文并茂的知识点，而不支持经过大模型润色加工的知识答案。

快速搭建

1. 登录客户端运行环境，执行 `pip3 install tcvectordb`，安装向量数据库最新的 Python SDK。
2. 使用 unzip 命令，解压 document_search_demo_20241114.zip 压缩包。
3. 进入压缩包解压后的文件夹，执行 `pip3 install -r requirements.txt` 命令，安装 SDK 通用依赖，混元大模型 SDK 以及相关依赖组件。
4. 使用 `vim conf/config.ini` 命令，根据参数注释修改相关配置并保存，如下所示。

```
[vector_db]
# 腾讯云向量数据库访问地址，建议使用内网方式。
address=http://xxx:xxx
# 向量数据库实例密钥
key=xxx
ai_db=test_ai_db
ai_collection=test_ai_collection

[model]
address=hunyuan.tencentcloudapi.com
# 备选模型：hunyuan-standard,hunyuan-pro,hunyuan-turbo
name=hunyuan-turbo
# 开通混元模型api能力账号的secret_id和secret_key
secret_id=xxx
secret_key=xxx

[server]
# 启动服务的地址和端口
name=127.0.0.1
port=7869
```

配置项	参数名	参数含义	配置说明
[vector_db]	address	向量数据库实例的内网地址或外网地址。建议使用内网方式。	请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息。
	key	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理。

	ai_db	AI 类数据库名。	Database 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。 • 长度要求：[1,128]。
	ai_collection	AI 数据库集合视图名。	CollectionView 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。 • 长度要求：[1,128]。
[model]	address	腾讯云混元大模型的 API 服务域名	固定为 `hunyuan.tencentcloudapi.com` 。
	name	腾讯混元大模型的名称	hunyuan-turbo 为模型默认版本，采用全新的混合专家模型（MoE）结构。更多模型信息，请参见 混元大模型产品概述 ，可按需选择。
	secret_id	开通腾讯混元大模型的账号访问密钥 ID。	获取方式具体操作，请参见 开通腾讯混元大模型 。
	secret_key	腾讯混元大模型访问密钥 ID对应的 Key。	为降低密钥泄露的风险，自2023年11月30日起，对所有主账号、子账号的密钥，关闭查询 SecretKey 的功能，仅支持在创建时查看，请及时保存SecretKey。
[server]	name	客户端运行环境 IP 地址。	-
	port	运行环境分配的端口	注意避免端口冲突。

5. 执行 `python3 main.py` ，运行脚本，生成知识问答前端访问链接，如下图所示。

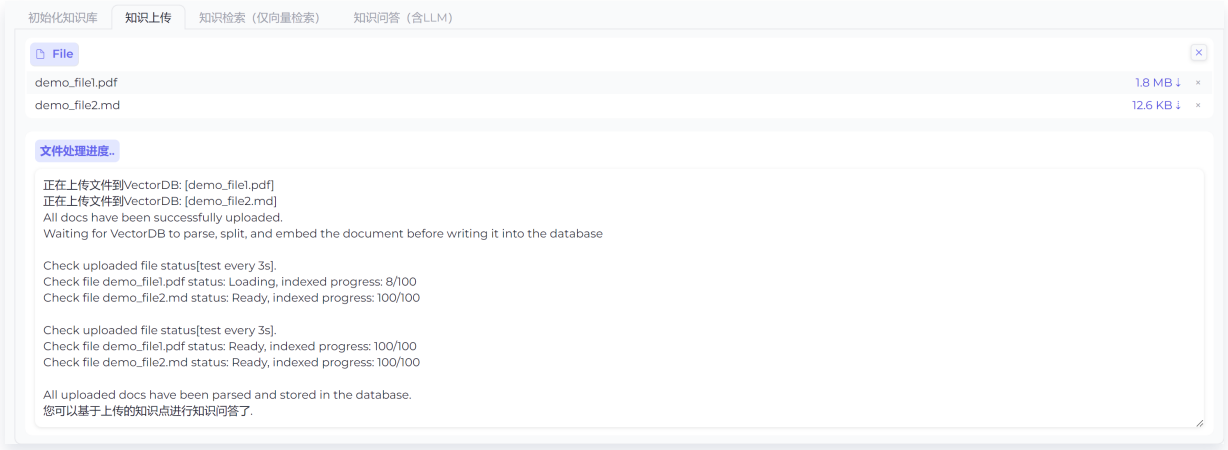
6. 复制 Running on local URL 后面的访问链接，在浏览器访问链接，显示 Tencent VectorDB AI Demo 配置向导，如下图所示。

> **注意：**
> 在浏览器访问知识问答应用，切勿关闭后端运行环境正在运行的进程。

7. 单击下方**开始初始化**，脚本将自动创建向量数据库与集合，界面显示初始化进度，等待任务完成，如下图所示。

8. 单击**知识上传**，将已准备的知识库文件 demo_file1.pdf 与 demo_file2.md 一起直接拖放或者上传至应用程序，程序将自动将文件向量化写入向量数据库，如下图所示。

> **说明：**
> 支持批量上传文件，不支持上传压缩包。



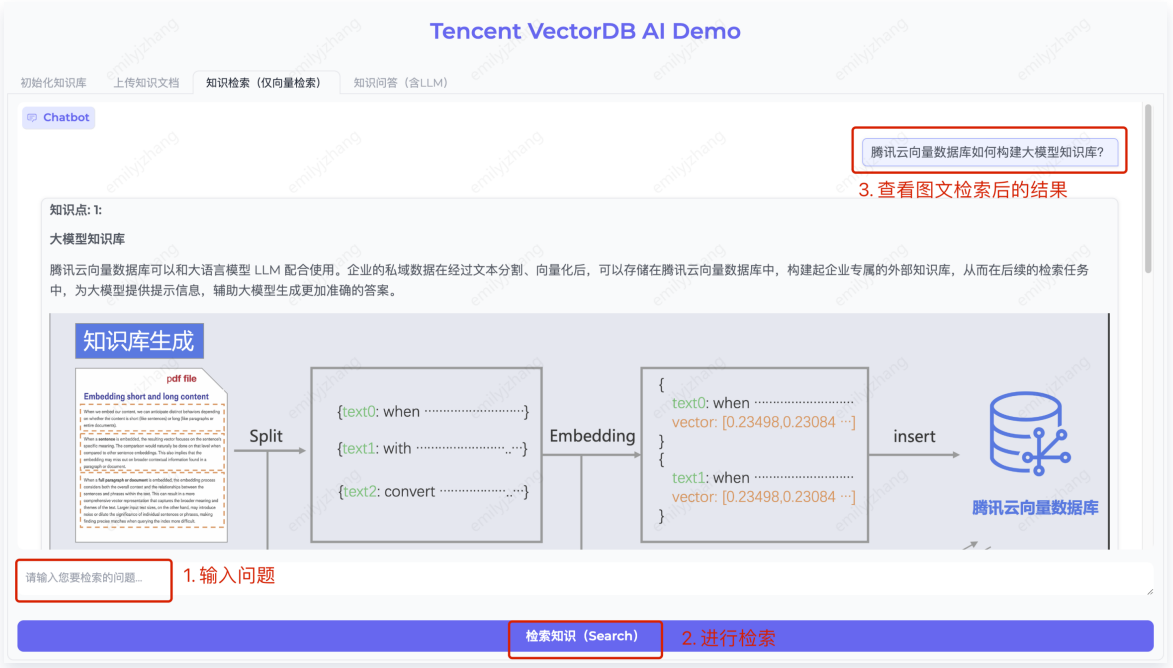
9. 等待文档上传完成，便可以进行知识检索或问答。

说明：

针对知识库文件，可参考如下问题进行问答和检索体验：

- 问题1: 腾讯云向量数据库如何构建大模型知识库?
- 问题2: 腾讯云向量数据库如何应用于推荐系统?
- 问题3: 腾讯云向量数据库如何应用于文本或者图片检索场景?
- 问题4: 腾讯会议如何快速发起会议?
- 问题5: 腾讯会议如何加入会议?
- 问题6: 腾讯会议如何取消会议?

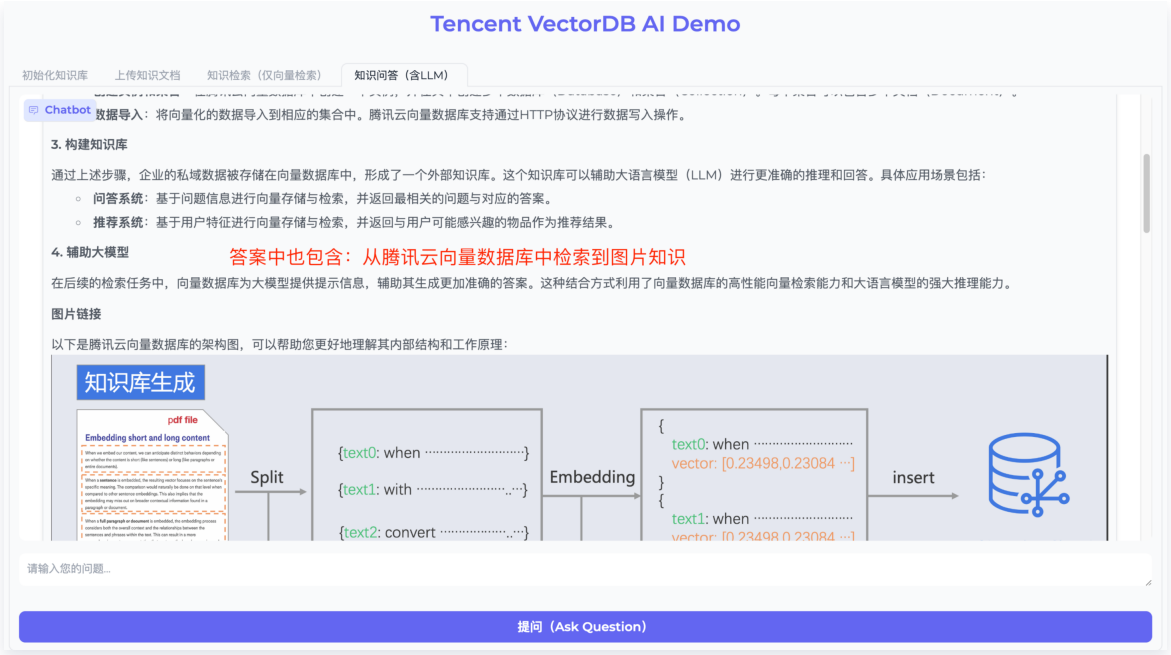
- **知识检索**：单击**知识检索（仅向量检索）**，在下方**请输入您要检索的问题**处，输入需要检索的问题，单击**检索知识**，即可检索到从腾讯云向量数据库中搜索到的图文并茂的知识点。



- **知识问答**：单击**知识问答（含LLM）**，在下方**请输入您的问题**处，输入需要查阅的问题，单击**提问**，即可看到经过腾讯混元大模型润色后的答案。

注意：

- 若未开通腾讯混元大模型，访问该**知识问答（含LLM）**页面将会提示错误。若需开通，请参见 开通混元大模型 。
- 若需更换其他 LLM 模型，请在工程 model 文件夹下添加其他模型，同步修改 conf 下配置文件中模型的信息。



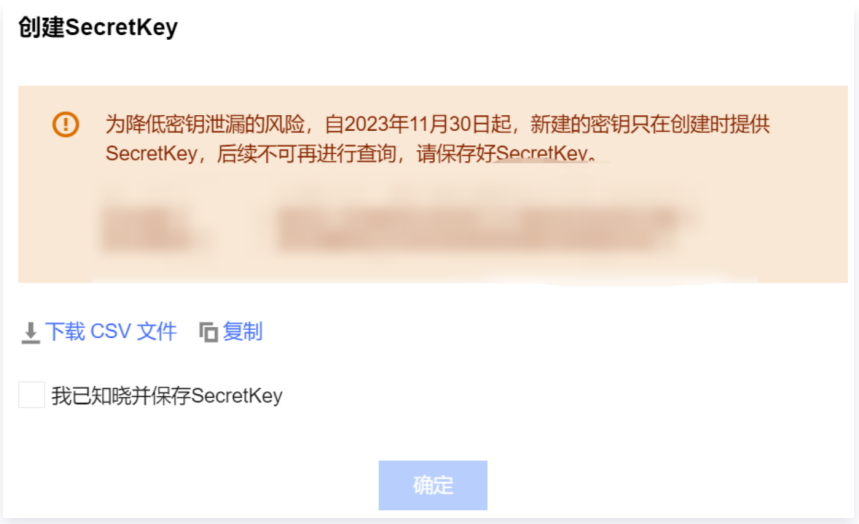
（可选）开通腾讯混元大模型

1. 前往 腾讯混元大模型控制台，如下图所示，在**服务关闭设置**区域，单击 ，立即开通。

2. 在左侧导航，选择**立即接入**，单击**创建密钥**，如下图所示。

3. 在 访问管理 > API 密钥管理 页面，单击**新建密钥**，显示如下图所示弹窗，获取访问密钥 **secret_id** 与 **secret_key**。

> **注意：**
> 为降低密钥泄露的风险，自2023年11月30日起，对所有主账号、子账号的密钥，关闭查询 SecretKey 的功能，仅支持在创建时查看，请及时保存SecretKey。



腾讯云向量数据库 AI 套件，可帮助企业挖掘文本数据的更大价值，加速企业在文本大模型领域的应用创新和发展。如果您在使用 AI 套件过程中有任何疑问，请联系腾讯云向量数据库，感谢您的支持。

向量数据库 + 百川大模型

最近更新时间：2025-02-11 17:29:13

本操作使用 Python SDK 录入 腾讯云向量数据库的产品文档，构建向量数据库的语料库。

步骤1：导入 Python SDK 依赖库

```
import requests
import json
import tcvectordb
import os
from tcvectordb.model.enum import ReadConsistency
from tcvectordb.model.collection_view import Embedding
```

步骤2：创建客户端对象，连接数据库实例

```
vdbclient = tcvectordb.VectorDBClient(url='{your vdb url}',
username='root', key='{your vdb key}')
```

步骤3：知识库初始化

声明 knowledgeInit() 函数，初始化知识库。

1. 为腾讯云向量数据库专有知识库创建一个 AI 类数据库 **testdb**。
2. 在已创建的 AI 类数据库中，创建集合视图 **knowledge**。
3. 上传 腾讯云向量数据库的产品文档 所有 md 文件于数据库集合视图中。

```
def knowledgeInit():
    #创建DB
    db = vdbclient.create_ai_database(database_name='testdb')
    #创建CollectionView
    collView =
db.create_collection_view(name='knowledge',embedding=Embedding(enable_
words_embedding=True))
    #上传文件
    file_path = "{/yourLocalPath/files/}"
    for file_name in os.listdir(file_path):
        if file_name.endswith(".md"):
            print("\n上传:"+file_name)

collView.load_and_split_text(local_file_path=file_path+file_name)
```

```
    print('upload all file sucess')
```

步骤4：传入问题进行知识内容相似性检索

声明 searchKnowlege() 函数，传入用户 question，返回知识库中与用户 question 最相似的内容。

```
def searchKnowlege(question):
    db = vdbclient.database('testdb')
    collView = db.collection_view('knowledge')
    doc_list = collView.search(
            content=question,
            limit=3
        )
    knowledge = ''
    print("查询向量数据库：")
    for count,doc in enumerate(doc_list):
            print("知识条目：", count, "----------------")
            print(doc.data.text)
            knowledge += doc.data.text
    return knowledge
```

步骤5：将用户问题与知识库检索的相似性语料，送入百川大模型，生成问题答案

> **说明：**
> 如下以 Baichuan2-Turbo 大模型为例，检索所获取的相似性语料将更有效地指引大语言模型（LLM）进行逻辑推理，生成更准确的答案。

```
def generate_answer(msg):
    url = "https://api.baichuan-ai.com/v1/chat/completions"
    api_key = "baichuan_api_key"
    data = {
        "model": "Baichuan2-Turbo",
        "messages": [{
                "role": "user",
                "content": msg
            }]
    }
    json_data = json.dumps(data)
    headers = {
```

```
        "Content-Type": "application/json",
        "Authorization": "Bearer " + api_key
    }
    response = requests.post(url, data=json_data,
headers=headers,timeout=60)
    if response.status_code == 200:
        print("=============================================")
        print("知识条目+大模型推理回答:\n", json.loads(response.text)
["choices"][0]["message"]["content"])
    else:
        print(response.text)
        print("请求失败，状态码:", response.status_code)
```

步骤6：构建 main() 函数

1. 调用 knowledgeInit() 初始化知识库。
2. 输入问题，调用 searchKnowledge() ，传入请求问题，在知识库检索与问题相似的知识。
3. 将检索结果的 Topk 条 **knowledges** 与输入的问题 **question** 进行内容组装。
4. 调用 generate_answer() ，传入组装后的内容，送入大模型，生成问题答案。

```
if __name__ == "__main__":
    knowledgeInit()
    question = input("请输入你的问题:\n")
    print("问题: ", question)
    print("=============================================")

    knowledges = searchKnowlege(question)
    content = json.dumps({
            "请回答问题": question,
            "背景知识如下": knowledges
        },ensure_ascii=False)
    generate_answer(content)
```

问答效果体验

腾讯云向量数据库产品文档中的一些内容，通用的大语言模型（LLM）在预训练时若没有相关的数据，出现了幻觉问题，导致在回答时的效果较差。在使用腾讯云向量数据库 AI 套件搭建知识库对大语言模型（LLM）进行知识补充后，如下三个问题都能准确地回答，对比只使用大语言模型（LLM）的效果显著提升。

问题1：什么是腾讯云向量数据库的 Embedding 功能?

- 直接给大语言模型（LLM）输入问题，生成答案如下所示。


```
{
    "index": 0,
    "message": {
        "role": "assistant",
        "content": "腾讯云向量数据库（TencentDB for TensorBase）的Embedding功能是一种针对大规模高维稀疏特征数据设计的存储和检索技术。它主要用于处理深度学习中的嵌入向量，如Word2Vec、GloVe等词嵌入模型生成的向量。\n\nEmbedding功能的主要特点包括：\n\n1. 高效存储：支持将高维稀疏特征压缩为低维稠密向量，并存储在数据库中，降低存储空间需求。\n\n2. 快速检索：提供高效的向量检索功能，支持多种相似度计算方式，如余弦相似度、欧几里得距离等。\n\n3. 分布式扩展：支持水平扩展，通过分片和复制提高系统的吞吐量和可用性。\n\n4. 灵活的数据处理：支持实时更新和查询操作，方便用户对数据进行实时分析和处理。\n\n5. 兼容性强：支持多种编程语言和开发框架，方便用户进行二次开发和集成。"
    },
    "finish_reason": "stop"
}
```


- 经过向量数据库的 AI 套件从知识库中检索出相似的 TopK 条语料，再送入大模型，生成答案如下所示。

```
问题： 什么是腾讯云向量数据库的Embedding功能
==========
查询向量数据库:
知识条目： 0 ----------
Embedding 功能是腾讯云向量数据库（Tencent Cloud VectorDB）提供将非结构化数据转换为向量数据的能力，目前已支持文本 Embedding 模型，能够覆盖多种主流语言的向量转换，包括但不限于中文、英文。开启 Embedding 功能并在创建 Collection 时配置模型，在插入、更新和相似性检索数据时直接传入原始文本，向量数据库会自动将原始文本进行转换，生成对应的向量数据后插入数据库或进行相似性计算，大幅提高业务接入效率。

知识条目： 1 ----------
## Embedding 实现架构
腾讯云向量数据库（Tencent Cloud VectorDB）通过 HTTP API 将这些非结构化文本数据送入向量数据库，向量数据库将原始文本数据转交给 Embedding 模型进行向量化，再将转换后的向量数据以及原始文本一并存储在向量数据库中。其整个实现架构，如下图所示。

知识条目： 2 ----------
## 计费说明
腾讯云向量数据库（Tencent Cloud VectorDB）默认开通 Embedding 功能。在使用 Embedding 功能时，腾讯云向量数据库（Tencent Cloud VectorDB）将会根据输入文本的 **Token **数量进行计费。目前在公测阶段，暂不计费。
> **说明：**
>
> 在 Embedding 模型中，Token 是指文本数据处理的基本单元。通常在文本中，一个 Token 可以是一个字或词，也可以是一个标点符号。
==========
知识条目+大模型推理回答：
 腾讯云向量数据库（Tencent Cloud VectorDB）的Embedding功能是一种将非结构化数据（如文本）转换为向量数据的能力。这种转换是通过预训练的Embedding模型来实现的，这些模型能够将文本数据映射到高维空间中的向量表示。

Embedding功能支持多种主流语言，包括中文和英文。当您启用此功能并在创建Collection时配置相应的模型，您可以直接将原始文本数据插入、更新和用于相似性检索。向量数据库会自动处理这些文本数据，将其转换为向量形式，从而简化了数据处理流程并提高了业务接入效率。

腾讯云向量数据库通过HTTP API接收非结构化文本数据，然后将这些数据传递给Embedding模型进行向量化处理。转换后的向量数据和原始文本一起存储在向量数据库中。

关于计费方面，腾讯云向量数据库默认开通Embedding功能。目前，在使用Embedding功能时，会根据输入文本的Token数量进行计费。在公测阶段，这项服务暂不收费。在Embedding模型中，Token是指文本数据处理的基本单元，它可以是一个字、词或标点符号。
```

问题2：什么是腾讯云向量数据库中的 AI 套件?

- 直接给大语言模型（LLM）输入问题，生成答案如下所示。

```
"choices": [
    {
        "index": 0,
        "message": {
            "role": "assistant",
            "content": "腾讯云向量数据库中的AI套件是一种工具集，它为开发者提供了在向量数据库上运行机器学习算法的能力。这个套件可以帮助用户更轻松地实现图像识别、自然语言处理等AI功能，并优化这些功能的性能和效率。\n\n具体来说，AI套件可能包括以下组件：\n\n1. 预训练模型：提供一些已经训练好的机器学习模型，如图像分类、物体检测等，方便用户直接使用。\n\n2. 模型训练与优化：提供模型训练和优化的工具，帮助用户提高模型的准确性和性能。\n\n3. 数据处理：提供数据清洗、特征提取等功能，帮助用户更好地处理和分析数据。\n\n4. 接口与SDK：提供API接口和软件开发包（SDK），方便用户在自己的应用程序中集成和使用这些AI功能。\n\n5. 监控与分析：提供实时监控和分析工具，帮助用户了解AI功能的运行状况和效果。"
        },
        "finish_reason": "stop"
    }
],
```


- 经过向量数据库的 AI 套件从知识库中检索出相似的 TopK 条语料，再送入大模型，生成答案如下所示。

```
问题： 什么是腾讯云向量数据库中的AI套件
==========
查询向量数据库:
知识条目： 0 ----------
## 什么是 AI 套件?
AI 套件是腾讯云向量数据库（Tencent Cloud VectorDB）提供的一站式文档检索解决方案，包含自动化文档解析、信息补充、向量化、内容检索等能力，并拥有丰富的可配置项，助力显著提升文档检索召回效果。用户仅需上传原始文档，数分钟内即可快速构建专属知识库，大幅提高知识接入效率。

知识条目： 1 ----------
### AI 类 Database
AI 类 Database 是专门用于 AI 套件上传和存储文件的向量数据库系统，可用于构建知识库。用户可以直接将文件上传至 AI 类 Database 下的 CollectionView 中，自动构建个性化的知识库。
> **说明：**
>
> - AI 类 Database 不支持直接对向量数据进行操作，已上传的文件不支持更新文件内容。
> - 为便于区别，腾讯云向量数据库将可直接操作向量数据的数据库称为 **Base 类 Database**。用户可以将向量数据上传至 Base 类 Database 中进行存储和管理，并可以直接对向量数据进行操作和处理。
知识条目： 2 ----------
## 设计思想
AI 套件检索方案提供完整的文档预处理和灵活的内容检索能力。用户只需上传 Markdown 格式的文档文件。腾讯云向量数据库将自动进行文本切分（Split）、信息补充、向量化（Embedding）和索引构建等一系列操作，完成知识库的建立。在进行检索时，会先基于切分后的内容进行相似度计算，并结合词（Words）向量进一步对检索结果进行精排，最终返回排名靠前的 Top K 条数据和其上下文内容。这种综合利用词级别做精排的检索方式，提供了更专业、更精确的内容检索体验。

==========
知识条目+大模型推理回答：
 腾讯云向量数据库中的AI套件是一站式文档检索解决方案，它包括自动化文档解析、信息补充、向量化、内容检索等功能。AI套件可以帮助用户快速构建专属知识库，提高知识接入效率。

AI套件使用AI类Database来存储和管理文件，用户可以将文件上传到AI类Database下的CollectionView中，自动构建个性化的知识库。需要注意的是，AI类Database不支持直接对向量数据进行操作，也不支持更新已上传的文件内容。

在设计思想上，AI套件提供完整的文档预处理和灵活的内容检索能力。用户只需上传Markdown格式的文档文件，腾讯云向量数据库会自动进行文本切分、信息补充、向量化和索引构建等一系列操作，完成知识库的建立。在进行检索时，会先基于切分后的内容进行相似度计算，并结合词向量进一步对检索结果进行精排，最终返回排名靠前的Top K条数据和其上下文内容。这种综合利用词级别做精排的检索方式，提供了更专业、更精确的内容检索体验。
```

问题3：腾讯云向量数据库支持的最大 QPS

- 直接给大语言模型（LLM）输入问题，生成答案如下所示。


```
"choices": [
    {
        "index": 0,
        "message": {
            "role": "assistant",
            "content": "腾讯云向量数据库TencentDB for OLTR（One-Click Low-Latency Time-series）的最大QPS取决于所选实例规格。不同实例规格具有不同的CPU、内存和IO能力，从而影响其处理请求的能力。\n\n具体支持的QPS数值，建议参考腾讯云官方文档或联系腾讯云客服获取详细信息。"
        },
        "finish_reason": "stop"
    }
],
```


- 经过向量数据库的 AI 套件从知识库中检索出相似的 TopK 条语料，再送入大模型，生成答案如下所示。

问题： 腾讯云向量数据库最大支持多大QPS
==========
查询向量数据库:
知识条目： 0 ----------
节点类型
腾讯云向量数据库依据存储节点 CPU 与内存资源分配比例不同，分为**存储型**和**计算型**两类。
- **存储型：**主要用于存储和管理大规模的向量数据，其主要优势在于：提供低查询延迟，能够高效地存储和管理向量数据，特别适用于数据量大、数据增长快、查询 QPS 相对较低的场景，例如：人脸识别、图像搜索等。
- **计算型：**主要用于快速查找和检索向量数据，支持高并发的查询请求，其主要优势在于：提供更高的查询 QPS 和更低的查询延迟，适用于流量大、延迟敏感的场景，例如：实时推荐、广告投放等。

知识条目： 1 ----------
腾讯云向量数据库是什么?
腾讯云向量数据库是一款全托管的自研企业级分布式数据库服务，专用于存储、检索、分析多维向量数据。该数据库支持多种索引类型和相似度计算方法，单索引支持10亿级向量规模，可支持百万级 QPS 及毫秒级查询延迟。腾讯云向量数据库不仅能为大模型提供外部知识库，提高大模型回答的准确性，还可广泛应用于推荐系统、NLP 服务、计算机视觉、智能客服等 AI 领域。

知识条目： 2 ----------
具体信息，请参见 [鉴权方式](https://write.woa.com/document/116658115942817792)。
- **连接方式**：腾讯云向量数据库支持通过 HTTP 协议进行数据写入和查询等操作。具体信息，请参见 [连接方式](https://write.woa.com/document/116551590185791488)。
- **检索方法**：腾讯云向量数据库支持通过精确检索、相似度检索、混合检索的方法。
- 精确查询：基于标量（指一个单独的数值，例如文本字段、数值字段或日期字段，区别于向量等多维数据结构）字段精确查找数据的方式。
==========
知识条目+大模型推理回答：
腾讯云向量数据库的最大支持QPS取决于具体的配置和部署环境。根据您提供的背景知识，腾讯云向量数据库可以支持百万级 QPS 及毫秒级查询延迟。然而，确切的QPS值可能会因实际应用场景、硬件资源、网络条件等因素而有所不同。

如果您需要了解特定场景下的最大QPS，建议您联系腾讯云的技术支持团队，以便他们为您提供个性化的建议和解决方案。

快速搭建以图搜图应用实践

最近更新时间：2025-06-19 15:13:42

背景信息

随着数据智能技术的不断发展，图像搜索技术在各行各业中的应用越来越广泛。传统的图像搜索技术往往依赖于图像的元数据或标签，存在信息更新不及时、垂直领域知识匮乏、搜索精度不高等问题。如何推进图像搜索技术在各行业、各业务场景的落地，成为各方普遍关注的问题。目前，基于向量检索的图像搜索技术正成为解决这些问题的有效方法，并逐渐成为数据智能时代的一个主要趋势。

图片向量检索技术通过将图片转换为高维向量，并在向量空间内计算相似度，实现高效、精准的图像搜索。本文将探讨如何结合腾讯云向量数据库和 CLIP（Contrastive Language-Image Pre-Training）图像处理模型，构建一站式的图搜应用解决方案。腾讯云向量数据库以其高性能、高可用性、大规模数据处理能力、低成本和简单易用性等优势，为用户提供了强大的向量数据存储和检索能力。结合 CLIP 模型，可实现通过向量相似性检索的方式搜索图片，并提升图像搜索的准确性和效率，适用于电商推荐、内容审核、智能相册等多种图像处理任务。

准备工作

1. 选型并购买向量数据库实例。具体操作，请参见 购买实例 。
2. 客户端运行环境准备。

类别	要求
地域与网络环境	• 使用腾讯云**外网**访问，需手动配置白名单，开通外网功能。具体操作，请参见 开启外网访问 。 • 使用腾讯云**内网**方式访问，确保购买的 CVM 所处地域与向量数据库为同地域，且 VPC 和安全组策略能正常连通向量数据库实例。
客户端	• 使用腾讯云**外网**方式，可使用本地客户端操作环境即可。 • 使用腾讯云**内网**方式，推荐选择 云服务器 CVM ，如 SA3.LARGE8、SA3.2XLARGE16等规格。
Python 环境依赖	• 推荐 Python 版本 >= 3.8。 • 安装 Python SDK 执行 `pip3 install tcvectordb` 命令，可直接安装最新版本。

3. 下载 graph_emb_demo1030.zip 压缩包，并将其上传于运行环境。
4. 下载图像文件 vdb_test_graphs.zip ，存放于本地环境。

快速搭建

1. 登录客户端运行环境，执行 `pip3 install tcvectordb` ，安装向量数据库最新的 Python SDK。

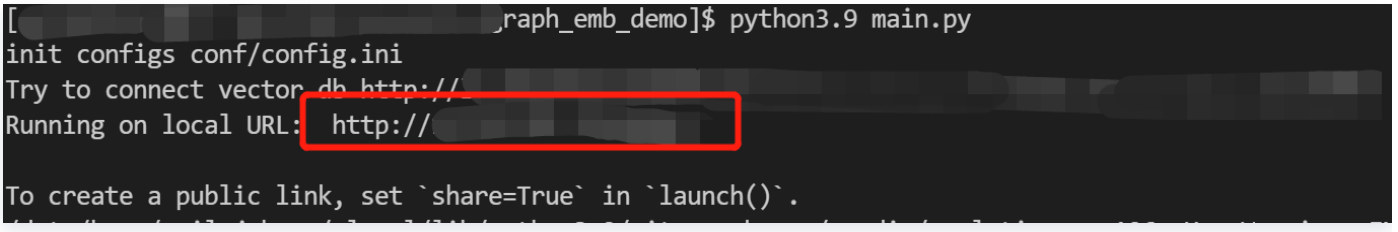
2. 使用 `unzip` 命令，解压 graph_emb_demo1030.zip 压缩包。
3. 进入压缩包解压后的文件夹，执行 `pip3 install -r requirements.txt` 命令，安装 SDK 通用依赖，模型 SDK 以及相关依赖组件。
4. 使用 `vim conf/config.ini` 命令，根据参数注释修改相关配置并保存，如下所示。

```
[vector_db]
# 腾讯云向量数据库访问地址。
address=http://xxx:xxx
# 向量数据库实例密钥
key=xxxxxxxx
# AI类向量数据库
ai_db=test_ai_db
# 向量数据库集合名
ai_collection=test_ai_graph_collection
# CLIP 模型以及存放路径
[download_model]
local_model_path=models/
model_name=openai/clip-vit-base-patch32

[graph_upload]
local_graph_path=graphs/

[server]
# 启动服务的地址和端口
name=127.0.0.1
port=8080
```

配置项	参数名	参数含义	配置说明
[vector_db]	address	向量数据库实例的内网地址或外网地址。建议使用内网方式。	请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息 。
	key	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理 。
	ai_db	AI 类数据库名。	Database 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。

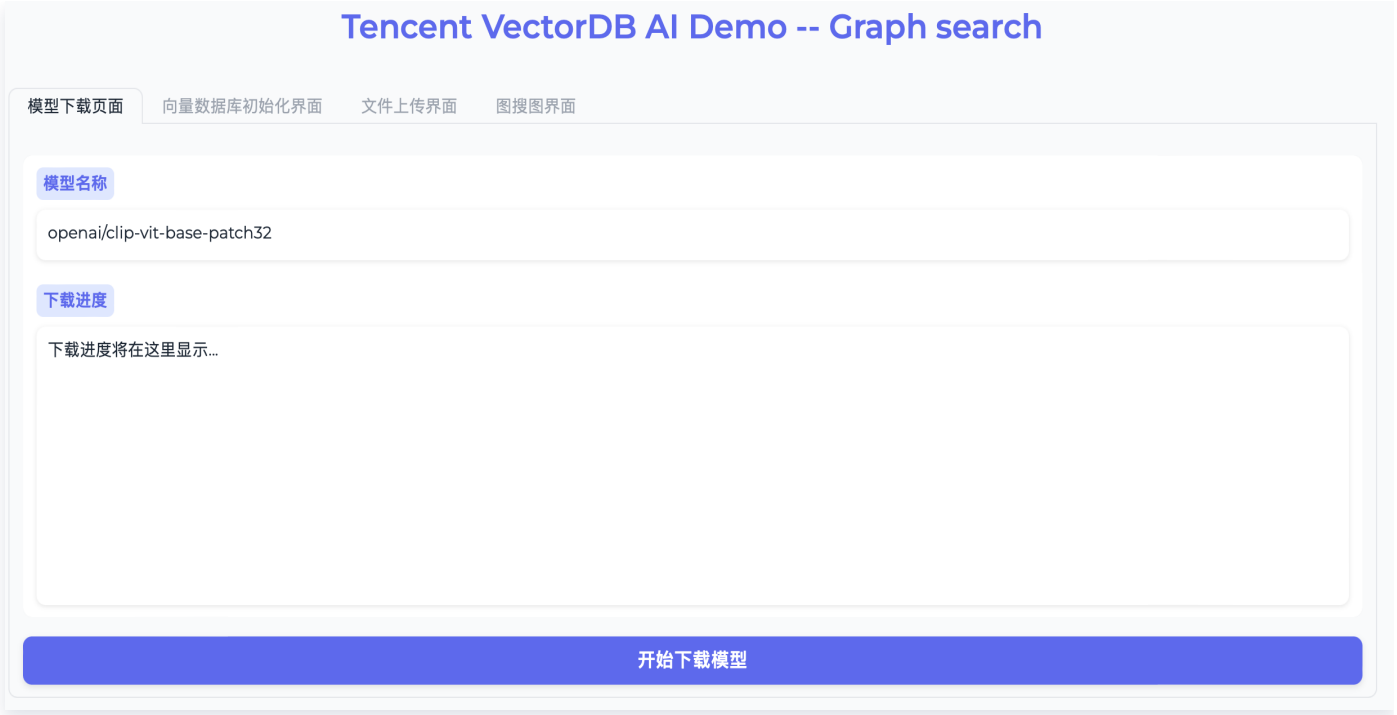
			• 长度要求：[1,128]。
	ai_collection	AI 数据库集合视图名。	CollectionView 命名要求如下： • 只能使用英文字母，数字，下划线_、中划线-，并以英文字母开头。 • 长度要求：[1,128]。
[download_model]	local_model_path	CLIP 模型存放路径。	脚本运行将下载 CLIP 模型，需指定其客户端存放路径。本示例使用相对路径，指定于压缩包所在文件夹 `models` 下。
	model_name	CLIP 模型名称。	默认为 `openai/clip-vit-base-patch32` 模型。如需更换为其他 CLIP 模型，请参见 Huggingface Models 。注意格式：Hugging face 仓库名/模型名称。
[graph_upload]	local_graph_path	图像文件压缩包存放路径。	本示例使用相对路径，指定于压缩包所在文件夹 `graph` 下。
[server]	name	客户端运行环境 IP 地址。	-
	port	运行环境分配的端口。	注意避免端口冲突。

5. 执行 `python3 main.py` ，运行脚本，生成图搜应用前端访问链接，如下图所示。



6. 复制 `Running on local URL` 后面的访问链接，在浏览器访问链接，显示 **Tencent VectorDB AI Demo -- Graph search** 配置向导页面，如下图所示。

> **注意：**
> 在浏览器访问图搜应用，切勿关闭后端运行环境正在运行的进程。

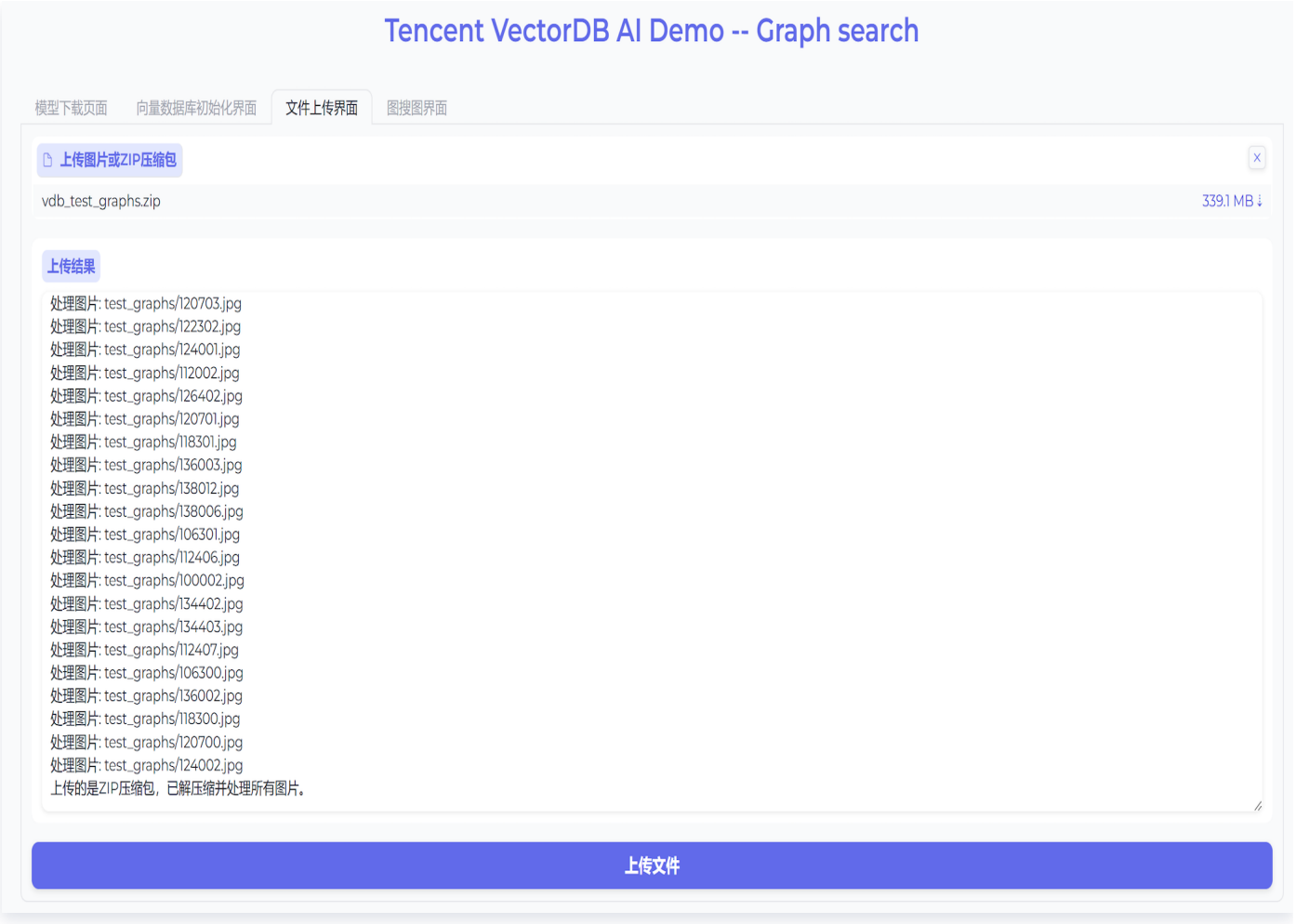


7. 在**模型下载页面**，单击**开始下载模型**，下载 CLIP 模型，模型文件将保存在配置文件中 local_model_path 指定的路径，默认下载至 `models/` 目录。

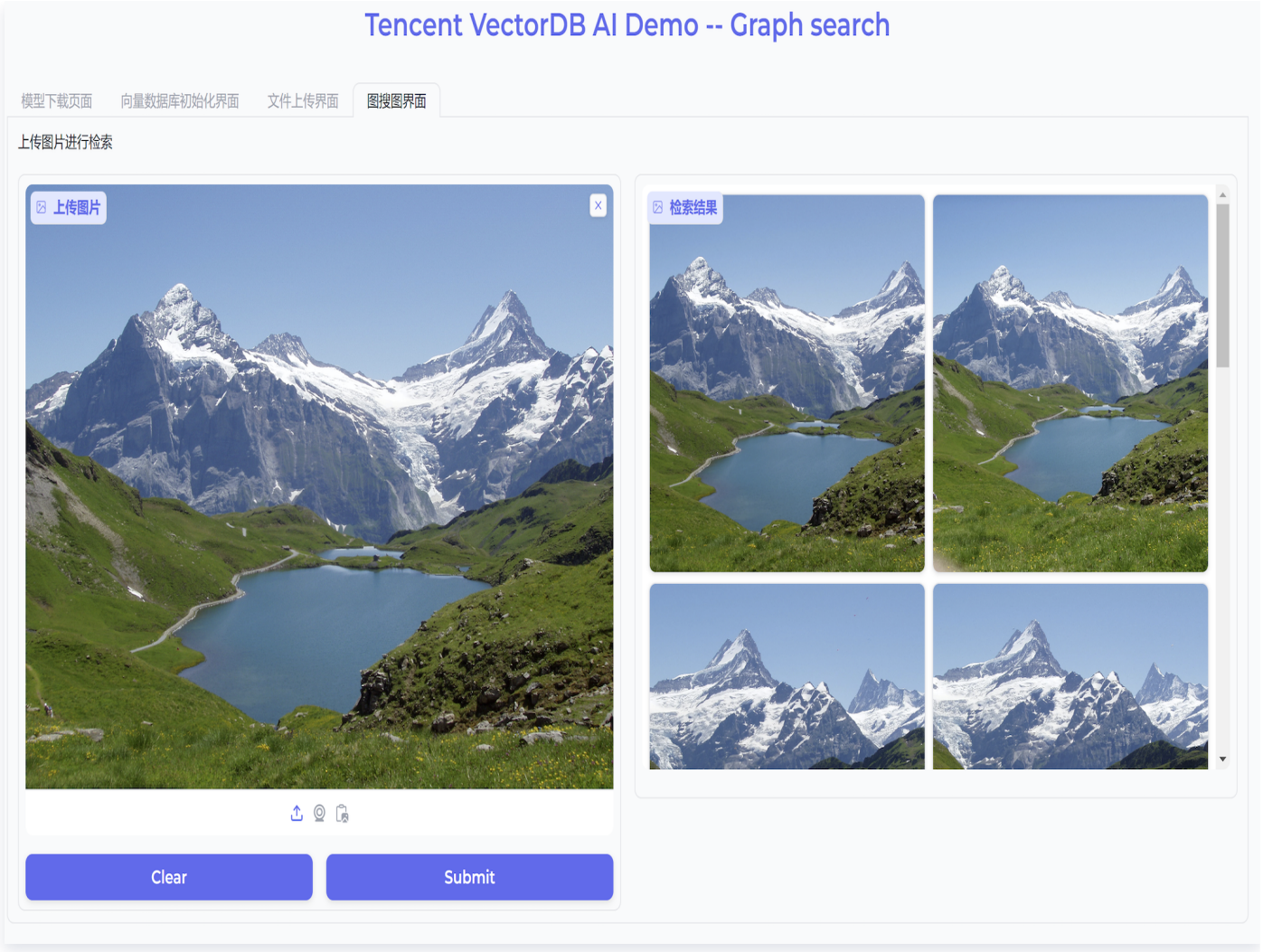
> **说明：**
>
> - 下载模型首次可能需要较长时间，可在后端运行环境查看下载进度，请耐心等待。之后再次下载时，系统会自动检测到模型已存在，可迅速完成操作。
> - 如果在下载模型的过程中遇到网络连接错误，可能是因为服务器与 Hugging Face 网站之间的连接不稳定，可以重新点击下载按钮来继续之前的下载过程。
> - 在**模型名称**输入框，也可以编辑更新为其他 CLIP 模型，配置文件也会同步更新。更多模型信息，请参见 Huggingface Models 。注意格式：Hugging face 仓库名/模型名称。

8. 切换至**向量数据库初始化界面**，单击**初始化向量数据库**，将自动在指定的实例创建向量数据库与集合，进行初始化，界面显示初始化进度，等待任务完成，如下图所示。

9. 切换至**文件上传界面**，在**上传图片或ZIP压缩包**区域，将已准备的图像文件 vdb_test_graphs.zip 直接拖放或者上传至应用程序，等待显示上传完成，单击下方的**上传文件**，程序将自动将图像文件上传于配置文件 config.ini 中 local_graph_path 指定的路径，并将图片向量化写入已创建的向量数据库中，如下图所示。



10. 切换至**图搜图界面**，在左侧**上传图片进行检索**区域，将需检索的图片拖放或上传于此处，单击 **Submit**，程序将自动检索，并在右侧**检索结果**区域显示其检索的结果，如下图所示。



遵循上述步骤，您将能够顺利运行图搜图 Demo 应用，上传图片集合，并利用图搜图功能查找相似图片。如果您有任何其他疑问，欢迎随时联系腾讯云向量数据库团队获取支持。

安装虚拟环境

如果您遇到环境之间依赖包冲突，可通过安装虚拟环境来解决。遵循以下步骤，确保虚拟环境安装所有必需的依赖。

使用 venv 创建虚拟环境

1. 创建虚拟环境。

```
python -m venv img_search_demo
```

2. 激活虚拟环境。
   - 在 windows 操作系统

```
img_search_demo\Scripts\activate
```

- 在 macOS 或者 Linux 系统

```
source img_search_demo/bin/activate
```

3. 安装所需的依赖。

```
pip install -r requirements.txt
```

使用 conda 创建虚拟环境

1. 创建虚拟环境。

```
conda create --name img_search_demo python=3.9
```

2. 激活虚拟环境。

```
conda activate img_search_demo
```

3. 安装所需的依赖。

```
pip install -r requirements.txt
```

将文件数据导入数据库
导入 CSV 文件

最近更新时间：2024-12-10 17:28:22

工具介绍

工具脚本	说明
import_csv_into_vdb.py	用于导入 CSV 类型的数据文件于向量数据库

CSV 文件每一列已指定名称

```
python3 import_csv_into_vdb.py \
    --url="http://xx.xx.xx.xx:80"\
    --key="xB2iQyVVFy9AtEFswF4ohQ*****************"\
    --db="db-test-XX" --collection="collection-XX"\
    --file="./test-csv-*.csv"\
    --field_mappings="pk=id,title=title2"
```

CSV 文件未指定列名称

```
python3 import_csv_into_vdb.py\
     --url="http://xx.xx.xx.xx:80"\
     --key="xB2iQyVVFy9AtEFswF4ohQ*****************"\
     --db="db-test-XX" --collection="collection-XX"\
     --file="./test-csv-*.csv"\
     --
fields="paragraphs,vector,spider_time,comment,share,source,sentence,
like,quality_medium,title2,pub_time,collect,quality_poor,quality_goo
d,id"\
     --header=N
```

参数	参数含义	说明
url	向量数据库实例的内网地址或外网地址。建议使用内网方式。	获取向量数据库实例内网 IP 地址与端口，请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息 。
key	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理 。
db	数据库名。	–
collection	数据库集合名。	–
file	预导入的 CSV 文件的路径。	• 可以指定为**绝对地址**，也可以为相应程序运行的**相对地址**。 • 支持使用**通配符**，例如：--file="./test_*.csv"\，将文件名包含 test_ 的CSV文件均导入数据库。
field_mappings	CSV 文件中已指定了每一列的名称，需通过该参数指定 CSV 文件每一列名称与导入数据库字段的映射关系。	• CSV 列名称与数据库字段名不一致需配置映射关系，一致则无需配置。 • key = value，key 为 Json 字段，value 为向量数据库字段名，切勿写反。
header	若 CSV 文件未指定每一个列的名称，需配置该参数为 N 进行标识。	默认值为：Y。 **注意：** 若 CSV 文件未指定每一个列的名称，请务必设置该参数，否则写入数据失败。
fields	若 CSV 文件未指定每一个列的名称，需通过该参数指定 CSV 每一列的名称对应写入向量数据库集合的字段名。	–
confirm_work	标识是否通过 nohub 命令在后台运行数据导入任务。 • Y：默认值，可在终端会话确认参数配置是否正确。	如果导入海量数据，建议先设置 confirm_work=Y 确认配置参数是否正确，提示 `please check the parameters and field mappings [yes no]:`

	• N：可以通过 nohub 的方式来在后台运行程序。	输入 no 退出，再设置 confirm_work=N 配合 nohub 后台运行程序。
parallel	用于控制客户端并发数。	默认值：30。
batch_size	用于控制单次写入操作的 batch 数量。	根据维度等因素动态计算。

示例

1. 准备如下工作。

   a. 准备数据库实例。具体操作，请参见 购买实例 。

   b. 执行 `pip3 install tcvectordb` ，获取最新版本 Python SDK，并应用 Python SDK 连接数据库实例。具体信息，请参见 新建 Client 。

   c. 创建 Base 类数据库以及相应集合。具体操作，请参见 新建数据库 与 新建集合 。

   > **说明：**
   > • 该脚本工具当前不支持 Embedding 文本向量化，需创建直接写入向量数据的集合。
   > • 创建集合时，请注意向量数据维度与 CSV 文件的向量数据维度一致，并根据业务需要指定必要的 Filter 字段。

2. 将数据导入工具的 Python 脚本上传于运行的操作环境，同时上传 CSV 文件。

   > **说明：**
   > 向量字段在 CSV 文件中只能以字符串的形式存在，否则会被认为 CSV 的多个列。正确示例："[0.11, 0.23,...]"。

3. 根据 CSV 文件业务数据与文件路径，在文本编辑器配置工具脚本参数，如下所示。

CSV 文件每一列已指定名称

```
python3 import_csv_into_vdb.py \
  --url="http://10.0.x.x:80"\
  --key="xB2iQyVVFy9AtEFswF4ohQPoxUYok0**********"\
  --db="db-test-0905" --collection="coll-vector-01"\
  --file="./test_*.csv"\
  --field_mappings="pk=id,title=tag"
```

CSV 文件未指定列名称

```
python3 import_csv_into_vdb.py\
      --url="http://10.0.x.x:80"\
      --key="xB2iQyVVFy9AtEFswF4ohQPoxUYok0**********"\
      --db="db-test-0905" --collection="coll-vector-01"\
      --file="./test_*.csv"\
      --
fields="paragraphs,vector,spider_time,comment,share,source,sentence,
like,quality_medium,title2,pub_time,collect,quality_poor,quality_goo
d,id"\
      --header=N
```

4. 在操作环境运行脚本，显示如下所示，以 CSV 指定列名称为例。
   - 显示初始化参数：

   - 显示索引的字段，及其对应的索引方式。

```
----------------------------------------
[All indexes]
 id -> {'fieldName': 'id', 'fieldType': 'string', 'indexType': 'primaryKey'}
 spider_time -> {'fieldName': 'spider_time', 'fieldType': 'string', 'indexType': 'filter'}
 tag -> {'fieldName': 'tag', 'fieldType': 'uint64', 'indexType': 'filter'}
 source -> {'fieldName': 'source', 'fieldType': 'string', 'indexType': 'filter'}
 vector -> {'fieldName': 'vector', 'fieldType': 'vector', 'indexType': 'HNSW', 'metricType': 'COSINE', 'dimension': 768, 'params': {'M': 16, 'efConstruction':
200}, 'indexedCount': 0}
----------------------------------------
```

- 显示 CSV 文件列名称 confirm_work 库字段的映射关系。

```
--------------------------------------------
[csv columns count]: 15
[fields mapping] csv_column_name -> vdb_field_name:
  paragraphs -> paragraphs
  vector -> vector
  spider_time -> spider_time
  comment -> comment
  share -> share
  source -> source
  sentence -> sentence
  like -> like
  quality_medium -> quality_medium
  title -> tag
  pub_time -> pub_time
  collect -> collect
  quality_poor -> quality_poor
  quality_good -> quality_good
  pk -> id
```

5. 提示 `please check the parameters and field mappings [yes no]:`，输入 `yes`，启动数据导入任务。

> **说明：**
> 当导入数据量大时，推荐先使用默认值（confirm_work=Y）运行，在终端会话确认索引参数、映射字段设置正确，提示 `please check the parameters and field mappings [yes no]:`，输入 no，退出之后再指定 `confirm_work=N`，重新启动程序，在后台运行数据导入任务。

```
please check the parameters and field mappings [yes|no]: yes
Try to write the first row of data from CSV to VDB
Successfully inserted the first piece of data, the remaining data will be inserted concurrently next.
----------------------------------------
start processes.
producer process, queue data size: 665
需处理文件列表:
['./test_01.csv', './test_02.csv']
开始处理文件: ./test_01.csv
process 0 ready, dimensions = 768, batch_size = 133
process 1 ready, dimensions = 768, batch_size = 133
process 2 ready, dimensions = 768, batch_size = 133
process 3 ready, dimensions = 768, batch_size = 133
process 4 ready, dimensions = 768, batch_size = 133
process 5 ready, dimensions = 768, batch_size = 133
process 6 ready, dimensions = 768, batch_size = 133
process 7 ready, dimensions = 768, batch_size = 133
process 8 ready, dimensions = 768, batch_size = 133
process 9 ready, dimensions = 768, batch_size = 133
process 10 ready, dimensions = 768, batch_size = 133
process 11 ready, dimensions = 768, batch_size = 133
process 12 ready, dimensions = 768, batch_size = 133
process 13 ready, dimensions = 768, batch_size = 133
process 14 ready, dimensions = 768, batch_size = 133
process 15 ready, dimensions = 768, batch_size = 133
process 16 ready, dimensions = 768, batch_size = 133
process 17 ready, dimensions = 768, batch_size = 133
process 18 ready, dimensions = 768, batch_size = 133
process 19 ready, dimensions = 768, batch_size = 133
process 20 ready, dimensions = 768, batch_size = 133
process 21 ready, dimensions = 768, batch_size = 133
```

6. 等待写入任务完成，提示类似如下信息，说明执行成功。

```
2956 rows were successfully inserted into the VDB.
avg batch latency: 933.55ms, batch size: 133
TPS: 591.131, dimensions: 768, parallel: 30
```

导入 Jsonl 文件

最近更新时间：2024-12-10 17:28:22

工具介绍

工具脚本	说明
import_json_into_vdb.py	用于导入 Jsonl 类型的数据文件于向量数据库

```
python3.9 import_json_into_vdb.py\
    --url="http://xx.xx.xx.xx:80"\
    --key="xB2iQyVVFy9AtEFswF4ohQ*****************"\
    --db="db-test-0905" --collection="test-jsonl"\
    --file="./test_json_XX.jsonl"\
    --field_mappings="pk=id"
```

参数	参数含义	说明
url	向量数据库实例的内网地址或外网地址。建议使用内网方式。	获取向量数据库实例内网 IP 地址与端口，请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息 。
key	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理 。
db	数据库名。	-
collection	数据库集合名。	-
file	预导入的 jsonl 文件的路径。	• 可以指定为**绝对地址**，也可以为相应程序运行的**相对地址**。 • 支持使用**通配符**，例如：--file="./test_*.jsonl"\，将文件名包含 test_ 的 Jsonl 文件批量导入数据库。
field_mappings	指定Jsonl 字段名与导入数据库字段的映射关系。	• Jsonl 字段与数据库字段名不一致需配置映射关系，一致则无需配置。 • key = value，key 为 Jsonl 字段，value 为向量数据库字段名，切勿写反。

confirm_work	标识是否通过 nohub 命令在后台运行数据导入任务。 • Y：默认值，可在终端会话确认参数配置是否正确。 • N：可以通过 nohub 的方式来在后台运行程序。	如果导入海量数据，建议先设置 confirm_work=Y 确认配置参数是否正确，提示 `please check the parameters and field mappings [yes no]:` 输入 no 退出，再设置 confirm_work=N 配合 nohub 后台运行程序。

使用示例

1. 准备如下工作。

   a. 准备数据库实例。具体操作，请参见 购买实例 。

   b. 执行 `pip3 install tcvectordb` ，获取最新版本 Python SDK，并应用 Python SDK 连接数据库实例。具体信息，请参见 新建 Client 。

   c. 创建 Base 类数据库以及相应集合。具体操作，请参见 新建数据库 与 新建集合 。

   > **说明：**
   > - 该脚本工具当前不支持 Embedding 文本向量化，需创建直接写入向量数据的集合。
   > - 创建集合时，请注意向量数据维度与 JSON 文件的向量数据维度一致，并根据业务需要指定必要的 Filter 字段。

2. 将数据导入工具的 Python 脚本上传于运行的操作环境，同时上传 Jsonl 文件。

   > **说明：**
   > - JSON 文件格式，要求每一行一个 JSON 对象，例如：`{"pk":1,"name":"张三","vector":[0.11592275, -0.13876367, ...]}` 。
   > - 向量数据字段支持如下两种表达格式。
   >   - float数组：[0.11, 0.23,...]。
   >   - float数组字符串："[0.11, 0.23,...]"。

3. 根据 jsonl 文件业务数据与文件路径，在文本编辑器配置工具脚本参数，如下所示。

```
python3.9 import_json_into_vdb.py\
 --url="http://10.0.xx.xx:80"\
 --key="xB2iQyVVFy9AtEFswF4ohQP******************"\
 --db="db-test-0905" --collection="test-jsonl"\
 --file="./test_json*.jsonl"\
 --field_mappings="pk=id"
```

4. 在操作环境运行脚本，显示如下信息，逐一确认相关参数，索引字段以及字段映射关系。

```
[ ]$ python3.9 import_json_into_vdb.py\
>    --url="http:// '\
>    --key="xB2iQyVVFy9AtEFsv "\
>    --db="db-test-0905" --collection="test-jsonl"\
>    --file="./test_json*.jsonl"\
>    --field_mappings="pk=id"
----------------------------------------
[init args]
 {
  "url": "http:// ",
  "key": "******",
  "parallel": 30,
  "file": "./test_json*.jsonl",
  "db": "db-test-0905",
  "collection": "test-jsonl",
  "batch_size": -1,
  "field_mappings": "pk=id",
  "confirm_work": "Y"
}
upsert collection db-test-0905.test-jsonl.
dimensions: 768, batch_size: 133, build index=True
----------------------------------------
[All indexes]
 vector -> {'fieldName': 'vector', 'fieldType': 'vector', 'indexType': 'HNSW', 'metricType': 'COSINE', 'dimension': 768, 'params': {'M': 16, 'efConstruction':
200}, 'indexedCount': 4}
 id -> {'fieldName': 'id', 'fieldType': 'string', 'indexType': 'primaryKey'}
 name -> {'fieldName': 'name', 'fieldType': 'string', 'indexType': 'filter'}
----------------------------------------
[json columns count]: 3
[fields mapping] json_column_name -> vdb_field_name:
  pk -> id
  name -> name
  vector -> vector
----------------------------------------
```

5. 提示 `please check the parameters and field mappings [yes no]:` ，输入 `yes` ，启动数据导入任务。

> **说明：**
> 当导入数据量大时，推荐先使用默认值（confirm_work=Y）运行，在终端会话确认索引参数、映射字段设置正确，提示 `please check the parameters and field mappings [yes no]:` ，输入 no，退出之后再指定 `confirm_work=N` ，重新启动程序，在后台运行数据导入任务。

```
please check the parameters and field mappings [yes|no]: yes
Try to write the first row of data from jsonl to VDB
Successfully inserted the first piece of data, the remaining data will be inserted concurrently next.
----------------------------------------
start processes.
producer process, queue data size: 665
需处理文件列表:
['./test_json_02.jsonl', './test_json.jsonl', './test_json_01.jsonl']
开始处理文件: ./test_json_02.jsonl
成功处理文件: ./test_json_02.jsonl
开始处理文件: ./test_json.jsonl
process 0 ready, dimensions = 768, batch_size = 133
process 1 ready, dimensions = 768, batch_size = 133
process 2 ready, dimensions = 768, batch_size = 133
process 3 ready, dimensions = 768, batch_size = 133
process 4 ready, dimensions = 768, batch_size = 133
process 5 ready, dimensions = 768, batch_size = 133
process 6 ready, dimensions = 768, batch_size = 133
process 7 ready, dimensions = 768, batch_size = 133
process 8 ready, dimensions = 768, batch_size = 133
process 9 ready, dimensions = 768, batch_size = 133
process 10 ready, dimensions = 768, batch_size = 133
成功处理文件: ./test_json.jsonl
开始处理文件: ./test_json_01.jsonl
process 11 ready, dimensions = 768, batch_size = 133
成功处理文件: ./test_json_01.jsonl
process 12 ready, dimensions = 768, batch_size = 133
```

6. 等待提示类似如下信息，说明数据导入完成。

```
12 rows were successfully inserted into the VDB.
avg batch latency: 4651.70ms, batch size: 133
TPS: 40.056, dimensions: 768, parallel: 30
```

导入 Parquet 文件

最近更新时间：2024-12-10 17:28:22

工具介绍

工具脚本	说明
import_parquet_into_vdb.py	用于导入 Parquet 类型的数据文件于向量数据库

```
python3.9 import_parquet_into_vdb.py\
    --url="http://xx.xx.xx.xx:80"\
    --key="xB2iQyVVFy9AtEFswF4ohQ*****************"\
    --db="db-test-0905" --collection="test-parquet-new"\
    --file="./test-parquet*.parquet"\
    --field_mapping="text_id=id,vecs=vector"
```

参数	参数含义	说明
url	向量数据库实例的内网地址或外网地址。建议使用内网方式。	获取向量数据库实例内网 IP 地址与端口，请登录 向量数据库控制台，在**实例详情**页面**网络信息**区域直接复制**访问地址**。具体操作，请参见 查看实例信息 。
key	向量数据库实例 API 密钥，用于进行身份认证。	请登录 向量数据库控制台，在**密钥管理**页面直接复制密钥。具体操作，请参见 密钥管理 。
db	数据库名。	-
collection	数据库集合名。	-
file	预导入的 Parquet 文件的路径。	• 可以指定为**绝对地址**，也可以为相应程序运行的**相对地址**。 • 支持使用**通配符**，例如：--file="./test_*.parquet"\，将文件名包含 test_ 的 Parquet 文件批量导入数据库。
field_mappings	指定 Parquet 文件字段与导入数据库字段的映射关系。	• Parquet 文件字段与数据库字段名不一致需配置映射关系，一致则无需配置。

		• key = value，key 为 Parquet 文件字段，value 为向量数据库字段名，切勿写反。并且，**一个 key，可以映射到多个 Value**，例如：text_id=id|doc_id。 • 若源端与目标端字段数据类型不一致，支持**数据类型自动转换写入**。例如：text_id 是 unit64，数据库中的 id 和 doc_id 字段类型为 string，也可以建立映射关系，脚本会强制转换写入。
confirm_work	标识是否通过 nohub 命令在后台运行数据导入任务。 • Y：默认值，可在终端会话确认参数配置是否正确。 • N：可以通过 nohub 的方式来在后台运行程序。	如果导入海量数据，建议先设置 **confirm_work=Y** 确认配置参数是否正确，提示 `please check the parameters and field mappings [yes no]:` 输入 no 退出，再设置 **confirm_work=N** 配合 nohub 后台运行程序。

使用示例

1. 准备如下工作。

   a. 准备数据库实例。具体操作，请参见 购买实例 。

   b. 执行 `pip3 install tcvectordb` ，获取最新版本 Python SDK，并应用 Python SDK 连接数据库实例。具体信息，请参见 新建 Client 。

   c. 创建 Base 类数据库以及相应集合。具体操作，请参见 新建数据库 与 新建集合 。

   > **说明：**
   > • 该脚本工具当前不支持 Embedding 文本向量化，需创建直接写入向量数据的集合。
   > • 创建集合时，请注意向量数据维度与 Parquet 文件的向量数据维度一致，并根据业务需要指定必要的 Filter 字段。

2. 将数据导入工具的 Python 脚本上传于运行的操作环境，同时上传 Parquet 文件。

   > **说明：**
   > Parquet 文件的 Vector 字段可处理如下三种类型。
   > • 数组格式(list)：[1.2, 2.2, 0.38]。
   > • numpy格式(numpy.ndarray): numpy.array([1.2, 2.2, 0.38], np.float32)。
   > • 字符串格式(str)："[1.2, 2.2, 0.38]"。

3. 根据 Parquet 文件业务数据与文件路径，在文本编辑器配置工具脚本参数，如下所示。

```
python3.9 import_parquet_into_vdb.py\
    --url="http://xx.xx.xx.xx:80"\
    --key="xB2iQyVVFy9AtEFswF4ohQ*****************"\
    --db="db-test-0905" --collection="test-parquet-new"\
    --file="./test_*.parquet"\
    --field_mapping="text_id=id,vecs=vector"
```

4. 在操作环境运行脚本，显示如下信息，逐一确认相关参数，索引字段以及字段映射关系。

```
[ python3.9 import_parquet_into_vdb.py\
>    --url="http://  '\
>    --key="xB2iQyVVFy9AtEFswF4o  \
>    --db="db-test-0905" --collection="test-parquet-new"\
>    --file="./test_*.parquet"\
>    --field_mapping="text_id=id,vecs=vector"
----------------------------------------
[init args]
 {
  "url": "http:  ,
  "key": "******",
  "parallel": 30,
  "file": "./test_*.parquet",
  "db": "db-test-0905",
  "collection": "test-parquet-new",
  "batch_size": -1,
  "field_mappings": "text_id=id,vecs=vector",
  "confirm_work": "Y"
}
upsert collection db-test-0905.test-parquet-new.
dimensions: 128, batch_size: 800, build index=True
----------------------------------------
[All indexes]
 id -> {'fieldName': 'id', 'fieldType': 'string', 'indexType': 'primaryKey'}
 name -> {'fieldName': 'name', 'fieldType': 'string', 'indexType': 'filter'}
 vector -> {'fieldName': 'vector', 'fieldType': 'vector', 'indexType': 'HNSW', 'metricType': 'COSINE', 'dimension': 128, 'params': {'M': 16, 'efConstruction':
200}, 'indexedCount': 100000}
----------------------------------------
[parquet columns count]: 2
[fields mapping] parquet_column_name -> vdb_field_name:
  text_id -> id
  vecs -> vector
----------------------------------------
```

5. 提示 `please check the parameters and field mappings [yes no]:` ，输入 `yes` ，启动数据导入任务。

> **说明：**
> 当导入数据量大时，推荐先使用默认值（confirm_work=Y）运行，在终端会话确认索引参数、映射字段设置正确，提示 `please check the parameters and field mappings [yes no]:` ，输入 no，退出之后再指定 `confirm_work=N` ，重新启动程序，在后台运行数据导入任务。

```
please check the parameters and field mappings [yes|no]: yes
Try to write the first row of data from parquet to VDB
Successfully inserted the first piece of data, the remaining data will be inserted concurrently next.
----------------------------------------
start processes.
producer process, queue data size: 4000
需处理文件列表:
['./test_parquet_01.parquet', './test_parquet_02.parquet']
开始处理文件: ./test_parquet_01.parquet
process 0 ready, dimensions = 128, batch_size = 800
process 1 ready, dimensions = 128, batch_size = 800
process 2 ready, dimensions = 128, batch_size = 800
process 3 ready, dimensions = 128, batch_size = 800
process 4 ready, dimensions = 128, batch_size = 800
process 5 ready, dimensions = 128, batch_size = 800
process 6 ready, dimensions = 128, batch_size = 800
process 7 ready, dimensions = 128, batch_size = 800
process 8 ready, dimensions = 128, batch_size = 800
```

6. 等待提示类似如下信息，说明数据导入完成。

```
200000 rows were successfully inserted into the VDB.
avg batch latency: 11658.52ms, batch size: 800
TPS: 1504.685, dimensions: 128, parallel: 30
```

IVF 系列索引应用指南

最近更新时间：2024-09-08 17:20:01

IVF 索引方式数据导入速度快，且内存空间占用低，特别**适合于亿级大规模**高维向量数据的相似性检索，本文介绍通过 Python SDK 应用 IVF 索引的操作流程。

IVF 索引介绍

IVF 索引的基本原理是将向量数据集划分为多个子集，每个子集称为一个聚类中心或一个簇。每个簇都有一个代表性的向量，称为聚类中心向量。通过构建一个倒排表，将聚类中心向量与属于该簇的向量进行关联。
在进行搜索时，首先根据查询向量找到与之最相似的聚类中心向量，然后在该聚类中心对应的倒排表中查找更接近查询向量的具体向量。这种两级索引结构可以极大地减少搜索的计算量，提高搜索效率。

使用限制

- IVF 索引需要在插入一定量的数据后才能开始训练。因此，在插入数据时无需构建索引，等待数据插入完成后需重建 IVF 索引。
- 创建集合时，需配置参数 nlist， 聚类中心的数量，建议取值范围为 `[sqrt(单分片数据量)*4, sqrt(单分片数据量)*16]` 。每个分片内至少需要写入 30*nlist 条数据，最多选取 256*nlist 条数据进行模型训练。
- 重建索引过程中，不允许数据库写入，无法停止任务。

应用示例

导入 SDK

```
import tcvectordb
from tcvectordb.model.document import Document, SearchParams, Filter
from tcvectordb.model.enum import FieldType, IndexType, MetricType,
ReadConsistency
from tcvectordb.model.index import Index, VectorIndex, FilterIndex,
IVFPQParams
```

创建客户端连接

如下示例可直接复制，运行之前，您需在文本编辑器将 api_key=A5VOgsMpGWJhUI0WmUbY********** 与 10.0.X.X 依据实际情况进行替换。

```
# 1. 创建客户端连接
client = tcvectordb.VectorDBClient(url='http://10.x.x.x',
username='root',
```

```
key='A5VOgsMpGWJhUI0WmUbY********************', timeout=30,

read_consistency=ReadConsistency.STRONG_CONSISTENCY)
```

创建 Database

```
# 2. 创建 Database
db = client.create_database(database_name='db_test_ivf')
```

创建 Collection

如下列出创建集合时，应用 IVF 系列索引需特别设置的关键参数。

参数	参数含义	取值说明
indexType	索引类型，目前支持的 IVF_FLAT、IVF_PQ、IVF_SQ4, IVF_SQ8, IVF_SQ16。	本文以 IVF_PQ 为例介绍。
nlist	在 IVF_FLAT 算法中，向量空间被划分为 nlist 个聚类中心。	建议取值范围为：[sqrt(单分片数据量)*4, sqrt(单分片数据量)*16]
M	指乘积量化中原始数据被拆分的子向量的数量。将原始数据向量拆分为 M 个子向量。每个子向量的维度为 D/M，其中 D 是原始向量的维度。然后，对每个子向量进行独立的量化，得到 M 个码本（codebook），每个码本对应一个子向量的离散化表示。最终，将 M 个码本拼接起来，得到原始向量的 PQ 编码（code）。	• 仅 IVF_PQ 类型涉及该参数，IVF 其他系列无需设置。 • M 必须能被 D（原始向量的维度）整除。

```
# 3. 创建Collection，并使用IVF_PQ索引
# 3.1  定义索引，其中向量索引为IVF_PQ
index = Index(
    FilterIndex(name='id', field_type=FieldType.String,
index_type=IndexType.PRIMARY_KEY),
    FilterIndex(name='name', field_type=FieldType.String,
index_type=IndexType.FILTER),
    VectorIndex(name='vector', dimension=128,
index_type=IndexType.IVF_PQ,
                metric_type=MetricType.COSINE, params=IVFPQParams(m=2,
nlist=10))     # nlist建议取值范围为[sqrt(单分片数据量)*4, sqrt(单分片数据
量)*16]
)
```

```
# 3.2 创建集合
coll = db.create_collection(
    name='test_ivf',
    shard=1,
    replicas=2,
    description='this is a collection of test IVF_PQ',
    index=index
)
```

写入数据

> **注意：**
>
> 如果创建 Collection 选择的索引类型为 IVF 系列：
>
> - 当第一次写入时，当前集合还没有向量索引，此时 **buildIndex** 必须为 **false**。插入原始数据之后，需通过 rebuild 训练数据并重建索引。
> - 当集合已经调用过 rebuild 创建索引后，集合已经存在向量索引，此时：
>   - 如果 **buildIndex = true**，表示新写入的数据会加入到已有的 IVF 索引中，但不会更新索引结构，此时新写入的数据可以被检索到。
>   - 如果 **buildIndex = false**，表示新写入的数据不会加入到已有的 IVF 索引中，此时新写入的数据无法被检索到。

```
# 4. 写入数据（此处写入的数据，均为随机生成的测试数据）
# 注意：IVF索引需要写入一定数据量后，才能开始训练，
#       每个分片内至少需要写入 30*nlist 条数据，最多能训练 256*nlist 条数据
data_num = 300
for i in range(data_num):
    tmp_id = str(i)
    tmp_name = ''.join(random.sample(string.ascii_lowercase, 5))
    tmp_vector = np.random.randn(1, 128)[0].tolist()
    res = coll.upsert(
        documents=[
            Document(id=tmp_id, vector=tmp_vector, name=tmp_name)
        ],
        build_index=False    # IVF索引在首次写入数据时，需要设置build_index为
False
    )
```

Rebuild 索引

> **注意：**
> - 单分片内行数少于 30*nlist（nlist 为聚类中心数量）不支持训练。
> - 重建索引过程中，不允许数据库写入，无法停止任务。

如下列出 Rebuild 时需配置的关键参数，请参见下表。

参数	参数含义	配置方法及要求
drop_before_rebuild	标识在重建索引时，是否需先删除旧索引再重建新索引。 **说明：** 重建索引需要占用额外的内存空间，数据量越大，消耗的内存空间越大。在重建索引之前，您需根据实际资源情况选择是否需先删除旧索引再重建，避免引起内存占满而阻塞业务正常运行。	取值如下所示： • **True**：重建之前，先删除旧索引再重建索引。 **说明：** 内存资源不足时，可先删除旧索引，在新索引还没有创建完成之前，该表无法正常读写。 • **False**：重建之前，不删除旧索引，创建新索引完成之后再删除旧索引。默认为 False。 **说明：** 内存资源足够的情况下，可不删除旧索引。在新索引还没有创建完成之前，该表可读，禁止写入。
throttle	标识是否限制构建索引的单节点 CPU 核数。 **说明：** 重建索引会消耗 CPU 资源，为防止资源打满影响写入或者检索等操作，请根据业务实际配置重建索引的 CPU 核数。	取值如下所示： • **0**：不限制 CPU 核数。在模型训练期间，会消耗大量的 CPU 资源。重建索引任务将会尽快执行，但可能会对其他集合的读写操作产生影响。 • **1**：CPU 核数为 1，即仅使用 CPU 1 核进行模型训练，可避免构建索引期间对其他集合产生影响，但任务执行较慢。

```
# 5. 重建索引
# dropBeforeRebuild：标识重建索引之前，是否需要先删除旧索引，再创建新索引。
```

```
# throttle：标识是否限制构建索引的单节点 CPU 核数。默认为 1，即使用1核进行训练；可设置为 0，表示不限制 CPU 核数。
coll.rebuild_index(drop_before_rebuild=False, throttle=1, timeout=30)
```

查看索引状态

```
# 6. 查询索引是否重建完成
db = client.database('db_test_ivf')
res = db.describe_collection('test_ivf')
print(vars(res))
```

返回参数 **indexStatus** 中的 **status** 标识当前 Collection 重建索引的状态，**startTime** 显示重建索引开始的时间。

- **ready**：表示当前 Collection 准备就绪，可正常使用。
- **training data**：IVF 索引下特定状态，表示当前 Collection 正在进行数据训练，即训练模型已生成向量数据。
- **building index**：表示当前 Collection 正在重建索引，即将生成的向量数据存储到新的索引中。
- **failed**: 重建索引失败，可能会影响集合读写操作。

> ⚠ **注意：**
> **training data** 与 **building index** 状态期间不可写入数据。若重建索引之前先删除旧索引，则集合不可进行相似性查询，只能进行精确查询。

```
{
    "code": 0,
    "msg": "operation success",
    "collection": {
        "database": "db_test_ivf",
        "collection": "test_ivf",
        "documentCount": 4,
        "indexes": [
            ......
        ],
        "indexStatus": {
            "status": "ready",
            "startTime": ""
        }
    }
}
```

核心告警指标预设建议

最近更新时间：2025-03-14 16:18:43

实践背景

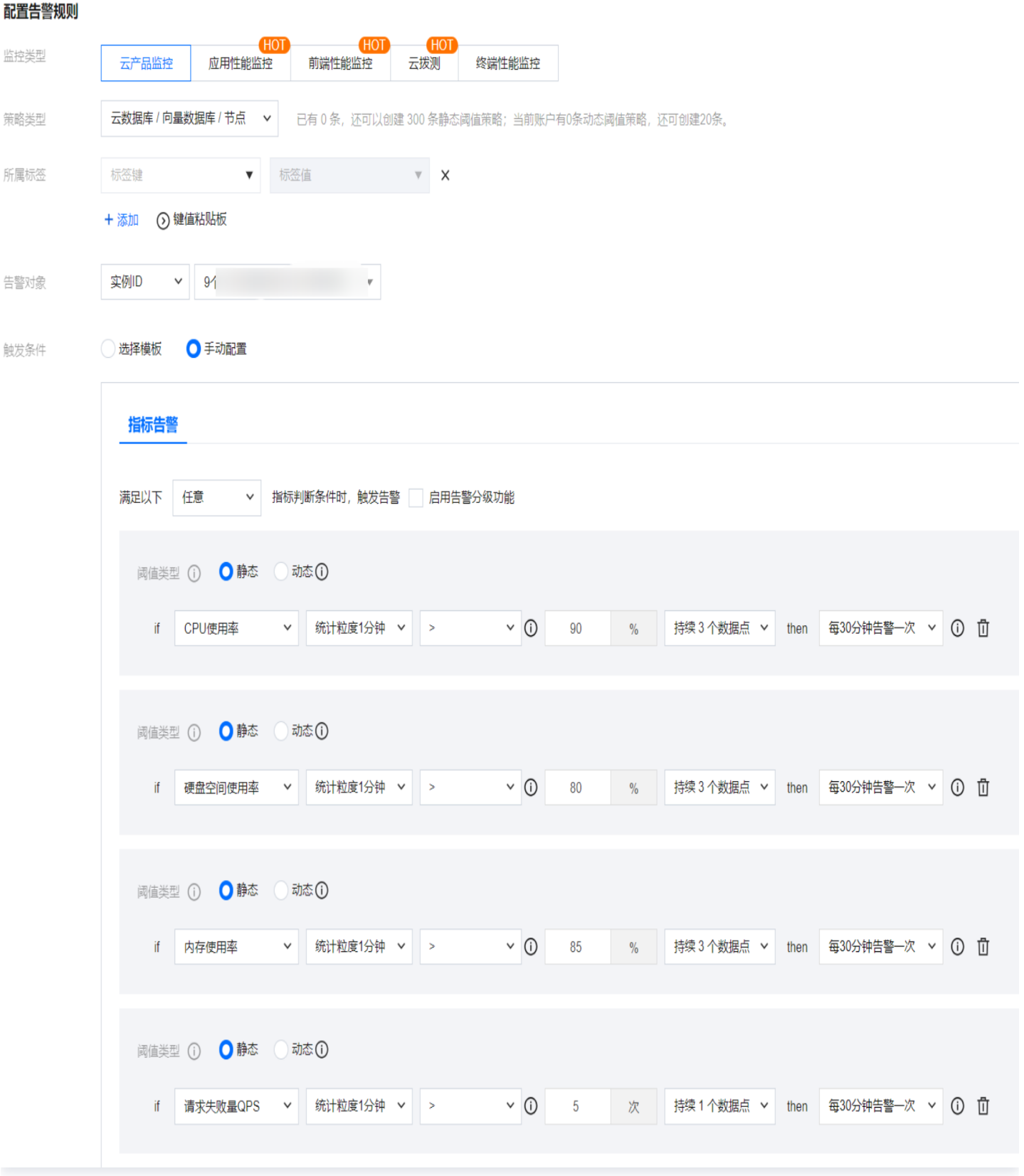
专家建议的核心告警指标通常包括但不限于：CPU 使用率、内存使用率、磁盘使用率等。通过这些指标，可以全面了解系统的运行状况，并采取适当的措施以确保系统的稳定和高效运行。

专家建议核心告警指标

告警指标	统计粒度	判断条件	阈值	单位	持续周期	告警方式
磁盘空间使用率（节点）	统计粒度1分钟	>	80%	%	持续3个数据点	每三十分钟告警一次
CPU 使用率（节点）	统计粒度1分钟	>	90%	%	持续3个数据点	每三十分钟告警一次
内存使用率（节点）	统计粒度1分钟	>	85%	%	持续3个数据点	每三十分钟告警一次
请求失败量 QPS（节点）	统计粒度1分钟	>	5	count/s	持续1个数据点	每三十分钟告警一次

配置告警策略

具体操作，请参见 配置监控告警 。配置专家建议的核心告警指标，如下图所示。



告警处理策略

告警原因分析及排查、资源问题等，请 提交工单 申请。