腾讯云智能体开发平台

实践教程

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

实践教程
- 前端页面接入（含demo）
- 知识问答一站式开发流程
- Agent 创建一站式开发流程
- 第三方教程

版权所有：腾讯云计算（北京）有限责任公司

实践教程
前端页面接入（含demo）

最近更新时间：2025-07-11 16:27:42

> **说明：**
> 本文档用于提供官方 demo，便于客户接入大模型，了解业务逻辑。
> 官方 demo 仅提供 vue2 版本，vue3 和 react 版本可参考 vue2 版本自行研发。

环境配置

准备工作

1. 安装 nodejs 和 npm ，版本参考下方 npm & node 的环境指导部分。
   Node.js — 在任何地方运行 JavaScript
2. 接入可参考 腾讯云智能体开发平台 对话端接口文档（WebSocket）-应用接口文档-文档中心-腾讯云 。

3. 下载demo文件包。

   前端demo代码

4. 了解统一渲染组件，详情参见npm地址。

   lke-component - npm

> **说明：**
> 如果不需要完整Demo，仅需要大模型回复的消息的前端渲染组件，可直接npm下载：
> - 消息统一渲染组件（Vue 2 或 React）： lke-component - npm
> - 消息统一渲染组件（Vue 3）： lke-component-vue3 - npm

node & npm的环境

1. node 建议使用 v14.x.x 版本
2. npm 推荐使用 v6.x.x 版本

```
// 检查node和npm版本
node -v
// v14.21.3
npm -v
// 6.14.18
```

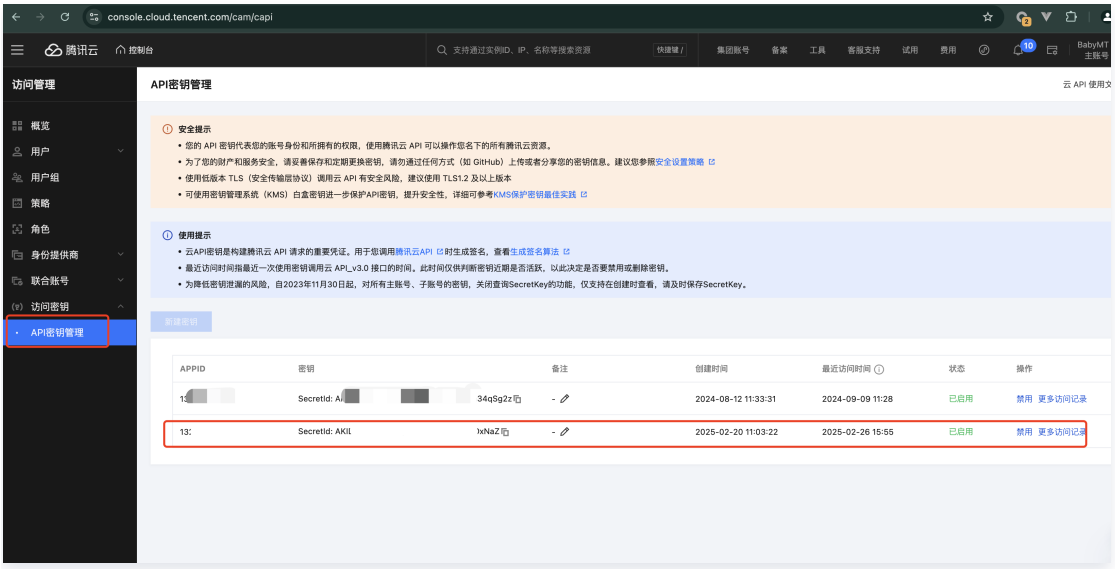
port可用检测

```
// mac上可以使用以下命令，查看3000端口和9091端口是否被占用，占用情况会影响本地服务启动
lsof -i :3000
lsof -i :9091
// windows上可用
netstat -ano | findstr :3000
netstat -ano | findstr :9091

// 这两个端口根据本地启动服务的各自情况来处理
// 3000对应npm run serve启动的服务
// 9091对应npm run dev启动的服务
```

通用配置

获取密钥

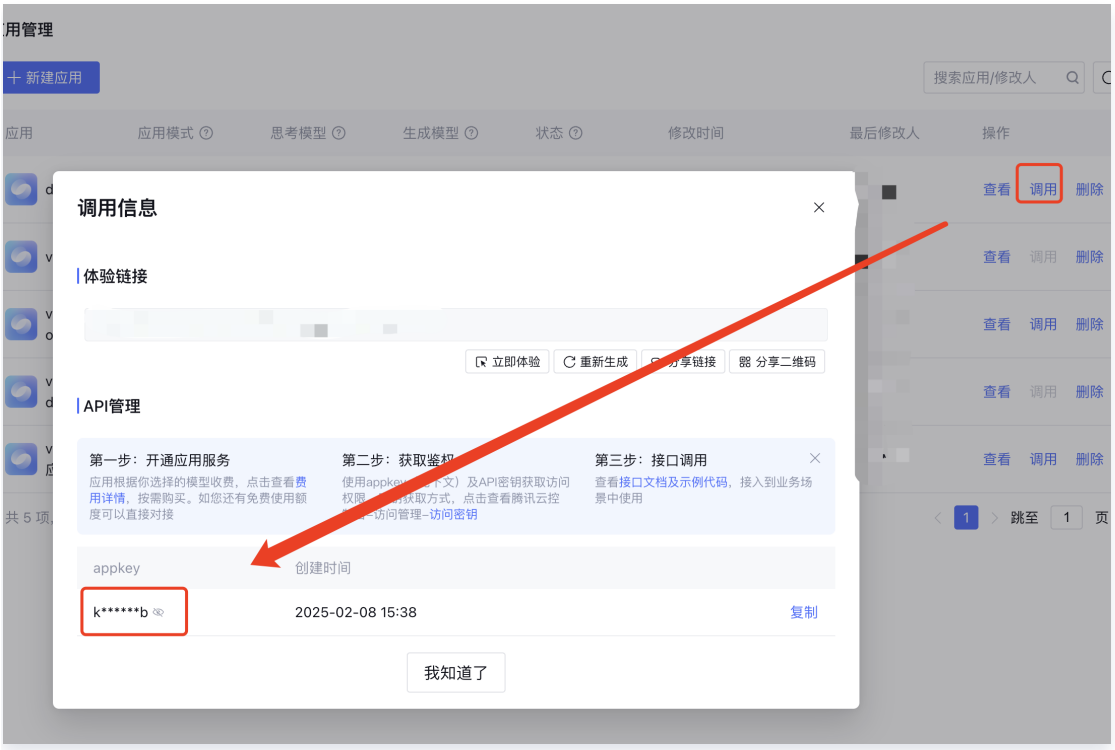
进入 腾讯云控制台-访问管理 -访问密钥-API密钥管理，获取 SecretId 和 SecretKey。



> **注意：**
> 只有新建密钥的时候才支持显示 SecretKey。如不记得之前的 SecretKey，可以新建一个，然后自行记录保存。或禁用原有的密钥（不再继续使用）后删除该密钥；删除之后就可以重新创建。

获取 appkey

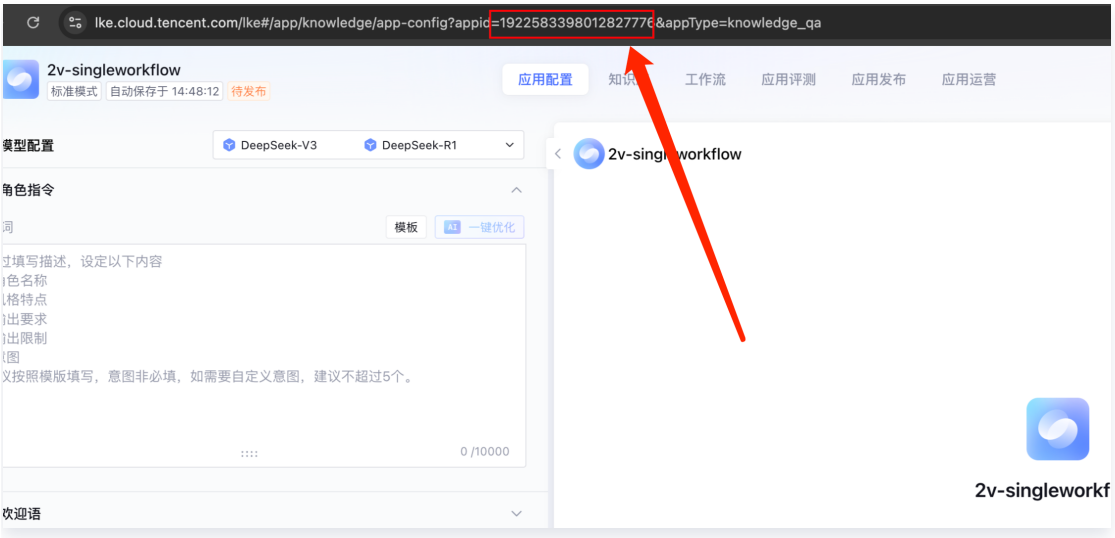
打开应用列表， 单击调用， 即可获取 appkey。



> **注意：**

> 在 demo 代码中，前端会把 appkey 的明文作为入参发送请求到后台或云 API，在无登录态的情况下，将 demo 传播给他人可能会造成您的 appkey 泄露。**您可以通过后台使用 appkey 来对接口进行转发，或者在获取 appkey 之前做好资源限制来避免 appkey 的泄漏。**

获取 botbizid



服务配置项

文件路径： `scripts/static.js`

1. 客户的SecretId:

```
const SecretId = 'SecretId'
```

2. 客户的SecretKey:

```
const SecretKey = 'SecretKey'
```

3. 应用的appkey:

```
const appId = 'appkey'
```

4. 体验机器人的botid:

```
const botId = 'botbizid'
```

web配置项

文件路径： `src/constants/static.js`

1. 标识 ws 模式还是 sse 模式链接对话，可选值 ws 和 sse。两者差异可查看 对话接口总体概述 。

```
const ACCESS_TYPE = 'ws'
```

2. 应用的appkey，同上服务配置项中的 appId。

```
const APP_KEY = 'appkey'
```

3. 体验机器人的 botid，同上服务配置项中 botid。

```
const BOT_ID= 'botbizid'
```

版权所有：腾讯云计算（北京）有限责任公司

启动服务

跨域配置处理

参考文件 `scripts/index.js`

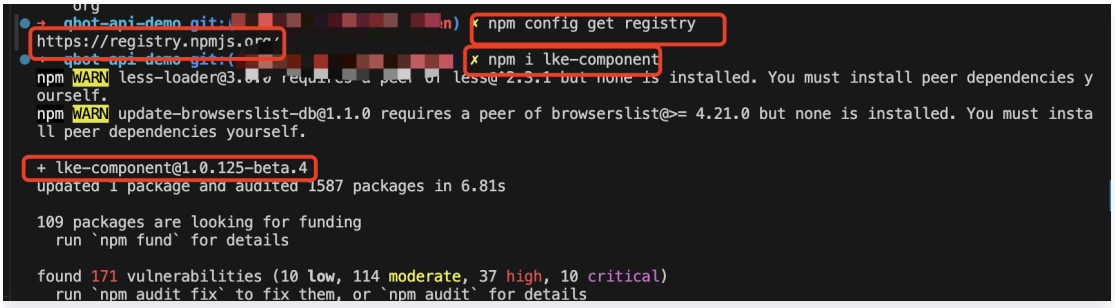
```
res.setHeader('Access-Control-Allow-Origin', '*');
res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');
res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
res.setHeader('Content-Type', 'application/json');
```

安装依赖包

检查 npm 源，安装 lke 统一渲染组件。

```
npm config get registry
// https://registry.npmjs.org/
// 检查Npm包的源
npm i lke-component
// 安装lke统一渲染组件
```

安装完毕后，结果显示如下：



> **说明：**
> 建议 package 中使用依赖：
> `"less-loader": "^3.0.0"`
> `"less": "^4.1.3"`

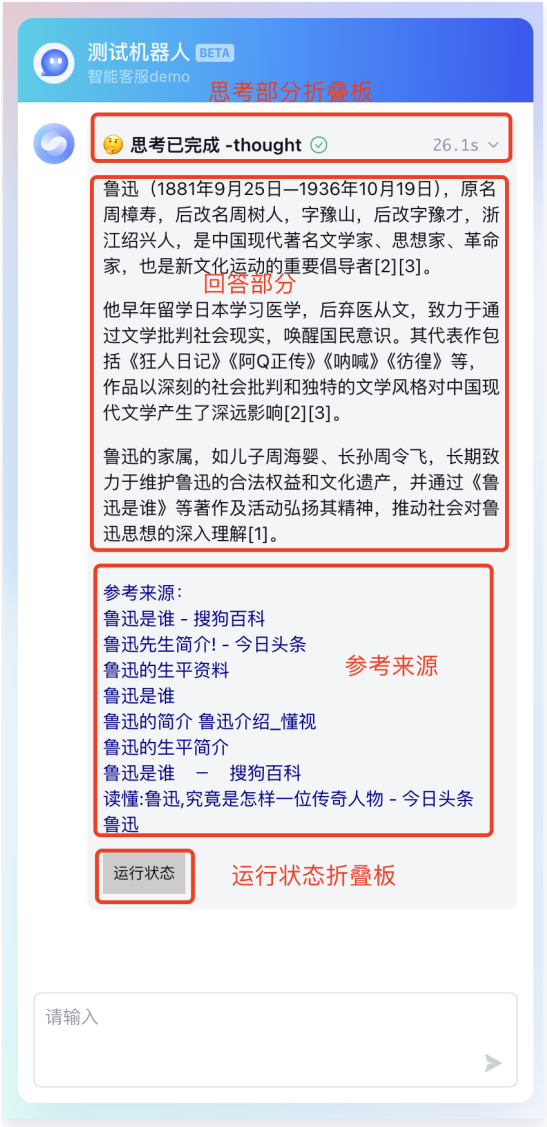
安装依赖

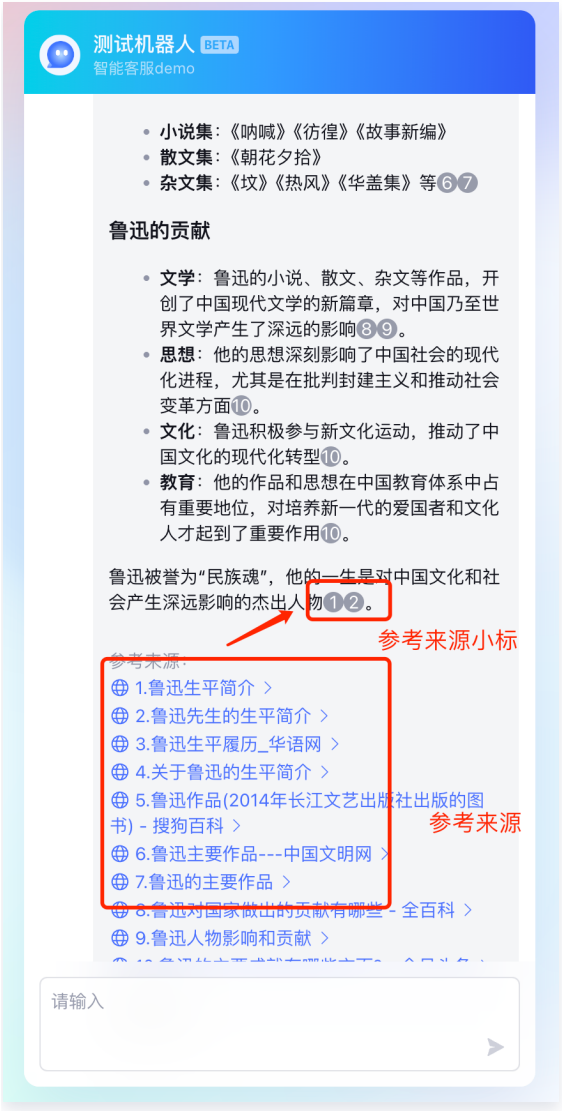
```
# 安装依赖
npm install

# 启动服务 at http://localhost:3000/
```

```
npm run serve

# 启动web at http://localhost:9091
npm run dev

# 打开页面 port视具体情况定 可能9091 可能9092 可能其他，注意前后一致性就好
http://localhost:9091/#/chat-demo
```

运行展示

本地访问域名 `http://localhost:9091/#/chat-demo` **，即可发起问答。**





打包

打包时需要配置打包配置项，参见文件 `config/.env.prod` ，其中 `PUBLIC_PATH` 和 `SERVER_PATH` 根据用户需求和用户域名灵活修改。

```
// config/.env.prod
NODE_ENV='production'
SERVER_ENV='prod'
WS_BASE_URL='wss://wss.lke.cloud.tencent.com'
SSE_BASE_URL='https://wss.lke.cloud.tencent.com'
API_BASE_URL='https://lke.cloud.tencent.com/cgi/capi'
// 根据用户的域名做灵活修改
PUBLIC_PATH=''
// 根据部署服务的地址做灵活修改，本示例是在本地启动，所以配置如下
SERVER_PATH='http://localhost:3000'
```

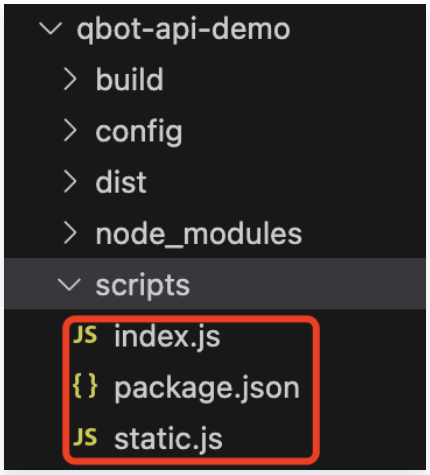
如需部署，可以用以下方法打包成静态文件。

```
# build for production with minification
npm run build
```

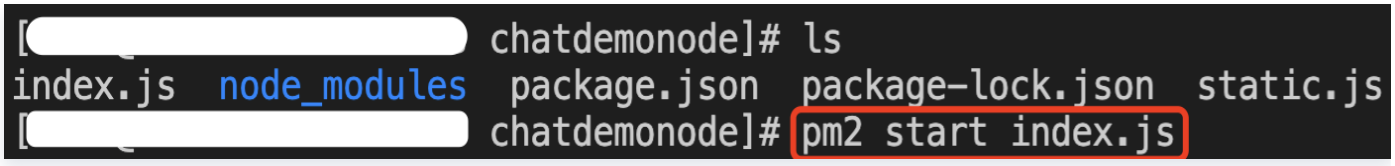
部署

如果需要部署本机之外的机器访问 demo，推荐您使用 web server 的方式。
如需部署到 nginx 等 web server，以下是一个例子。注意，demo 中提供的 node 服务仅是一个为了跑通 demo 的最简单的例子（下列1-3步），我们强力推荐将 demo 改造成调用真实的后台服务，以避免潜在的安全漏洞。

1. 将 node 服务代码拷贝到服务器中，之后在这个文件夹中运行 `npm install` 。

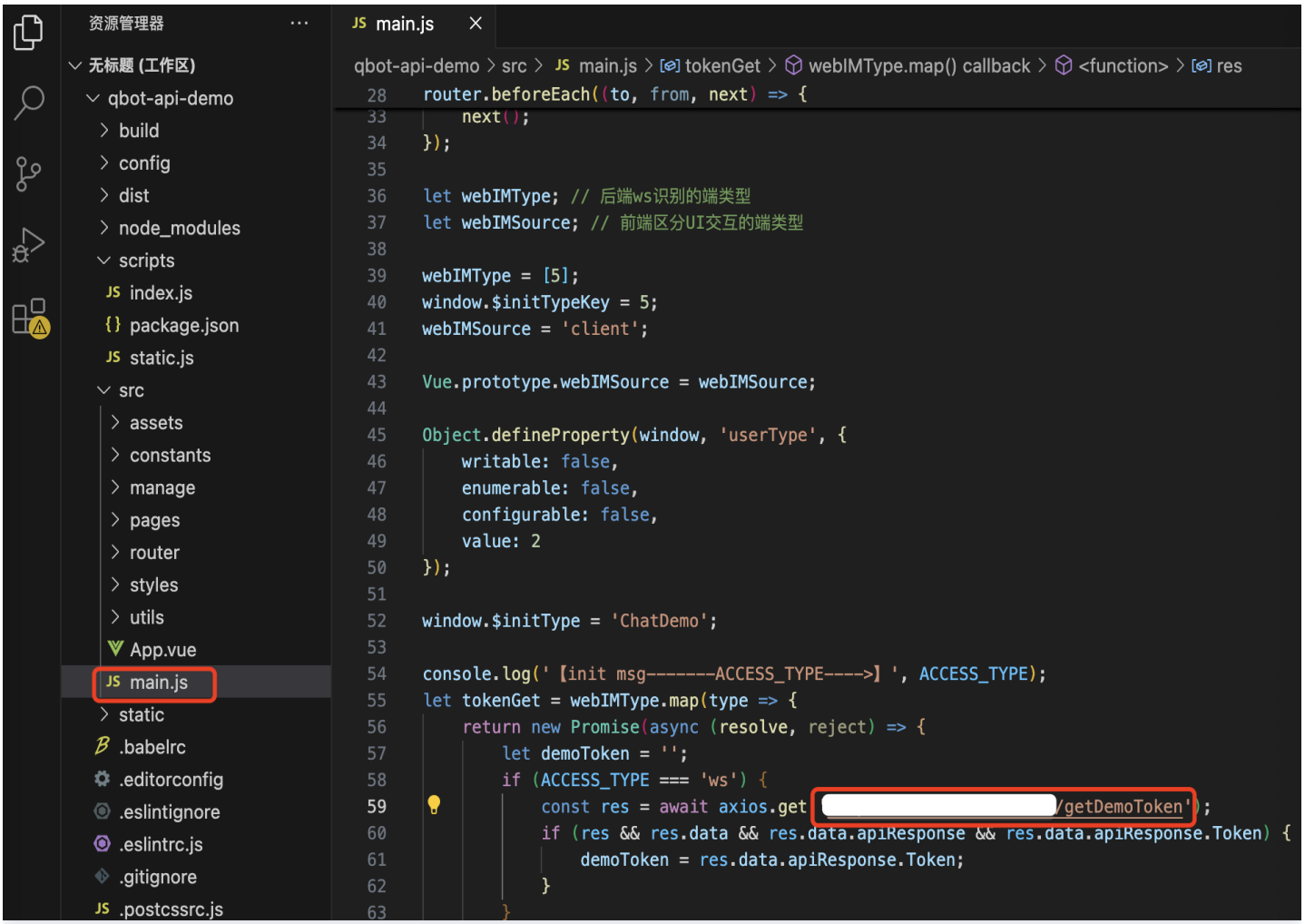


2. 安装守护进程工具 pm2 PM2 - Home 。
3. 用 pm2启动 node 服务。



4. 将代码中 node 服务的地址，替换为{您的 web server 具体的地址}/getDemoToken，例如 `https://mytest.com/getDemoToken` ，之后重新执行一次打包 。

> ⚠ **注意：**
> 请再次确认 web server 具体的地址的协议是 http 还是 https 。



5. 将打包后的 dist 文件夹拷贝到服务器中，并将文件夹路径在 nginx 配置文件中设置为 root（也可以基于其他 root 对 location / 做相应的配置），重启 nginx 即可，可参考下图例子：

```
server {
    listen       80;
    listen       443 ssl;
    server_name  [redacted]
    root         /root/workspace/chatdemo/dist;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
        try_files $uri $uri/ /index.html last;
        index  index.html;
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

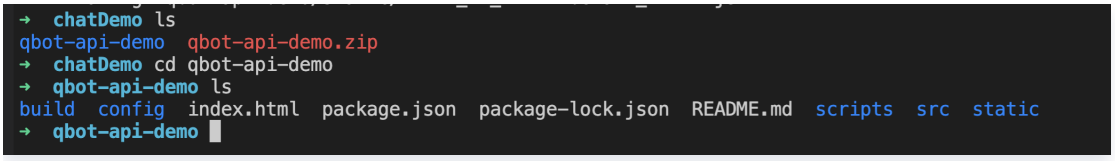
打包后文件在服务器的路径

部署示例

> **说明：**
> 下列示例为在 devCloud 上，启动一个部署的详细过程示例。不同的环境，执行操作和操作的代码会略有不同，请您灵活修改处理。

1. 将包拷贝到云服务器上，并解压。

```
# 解压
unzip qbot-api-demo.zip
```



2. 安装依赖。

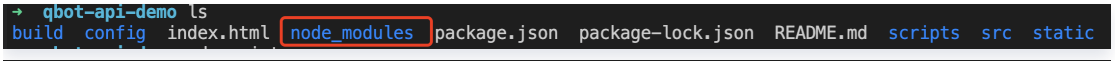
检查 node 和 npm 版本，推荐 node v14.x.x npm v6.x.x 。

```
node -v
# v14.21.2
npm -v
# 6.14.17
```

- 安装 npm 依赖

```
  # 在/qbot-api-demo目录下，安装依赖
npm i
# /qbot-api-demo/scripts目录下执行，安装服务的依赖
npm i
```

上述操作会分别生成一个 node_modules 文件夹，如下图所示：



```
→  scripts ls
index.js  node_modules  package.json  package-lock.json  static.js
```

3. 安装 PM2，并启动服务。

- 全局安装 PM2 进程管理工具

```
# 全局安装pm2
npm install pm2 -g
```

- 启动服务

```
# 启动服务
pm2 start scripts/index.js
# 查看进程日志
pm2 logs
```

```
→  qbot-api-demo pm2 start scripts/index.js
[PM2] Applying action restartProcessId on app [index](ids: [ 0 ])
[PM2] [index](0) ✓
[PM2] Process successfully started
```

id	name	mode	↺	status	cpu	memory
0	index	fork	0	online	0%	17.8mb

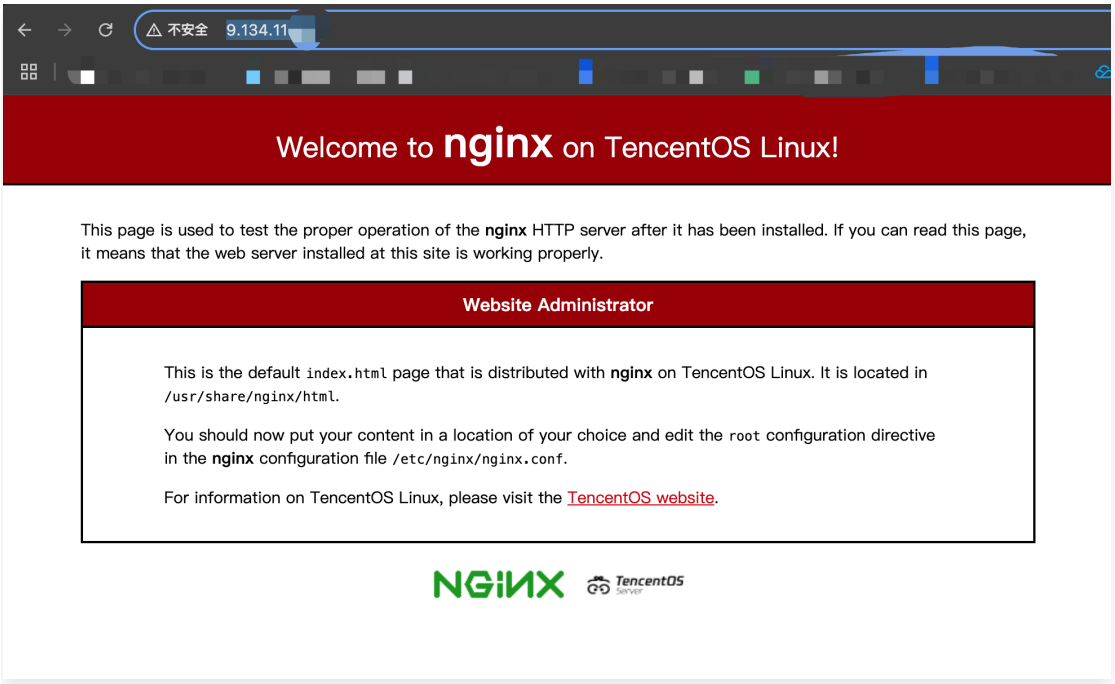
```
→  qbot-api-demo pm2 logs
[TAILING] Tailing last 15 lines for [all] processes (change the value with --lines option)
/root/.pm2/pm2.log last 15 lines:
PM2        | 2025-03-27T17:45:17: PM2 log: BUS socket file      : /root/.pm2/pub.sock
PM2        | 2025-03-27T17:45:17: PM2 log: Application log path : /root/.pm2/logs
PM2        | 2025-03-27T17:45:17: PM2 log: Worker Interval      : 30000
PM2        | 2025-03-27T17:45:17: PM2 log: Process dump file    : /root/.pm2/dump.pm2
PM2        | 2025-03-27T17:45:17: PM2 log: Concurrent actions   : 2
PM2        | 2025-03-27T17:45:17: PM2 log: SIGTERM timeout      : 1600
PM2        | 2025-03-27T17:45:17: PM2 log: Runtime Binary       : /root/.nvm/versions/node/v14.21.2/bin/node
PM2        | 2025-03-27T17:45:17: PM2 log: ================================================================================
PM2        | 2025-03-27T17:45:17: PM2 log: App [index:0] starting in -fork mode-
PM2        | 2025-03-27T17:45:17: PM2 log: App [index:0] online
PM2        | 2025-03-28T11:29:01: PM2 log: Stopping app:index id:0
PM2        | 2025-03-28T11:29:01: PM2 log: App [index:0] exited with code [0] via signal [SIGINT]
PM2        | 2025-03-28T11:29:01: PM2 log: pid=325318 msg=process killed
PM2        | 2025-03-28T11:52:21: PM2 log: App [index:0] starting in -fork mode-
PM2        | 2025-03-28T11:52:21: PM2 log: App [index:0] online

/root/.pm2/logs/index-error.log last 15 lines:
/root/.pm2/logs/index-out.log last 15 lines:
0|index    | Server running at http://localhost:3000/
0|index    | Server running at http://localhost:3000/
```

4. 安装 nginx，并部署 web 页面。

- 安装 nginx，启动 nginx。

```
# 安装nginx
sudo yum install nginx
# 启动nginx
sudo systemctl start nginx
```

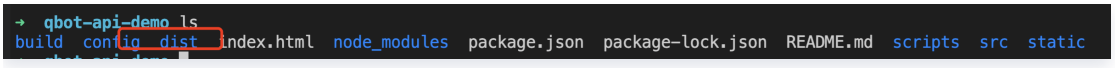
- nginx 启动成功后，访问 ip 可以看到 welcome nginx 页面。



- 部署 web 页面。

> ⚠ **注意：**
>
> 部署 web 页面时，首先检查调用api的路径是否正确，检查以下路径文件配置的 SERVER_PATH 配置是否和 serve 的 ip 和 port 一致。
>
> - qbot-api-demo/config/.env.local
> - qbot-api-demo/config/.env.prod
>
> 例如本次部署的 serve 在 devcloud，ip 为 9.134.11.xxx，port 为3000。
> 则 SERVER_PATH 配置如下：

```
# 打包web文件，生成dist文件夹
npm run build
```

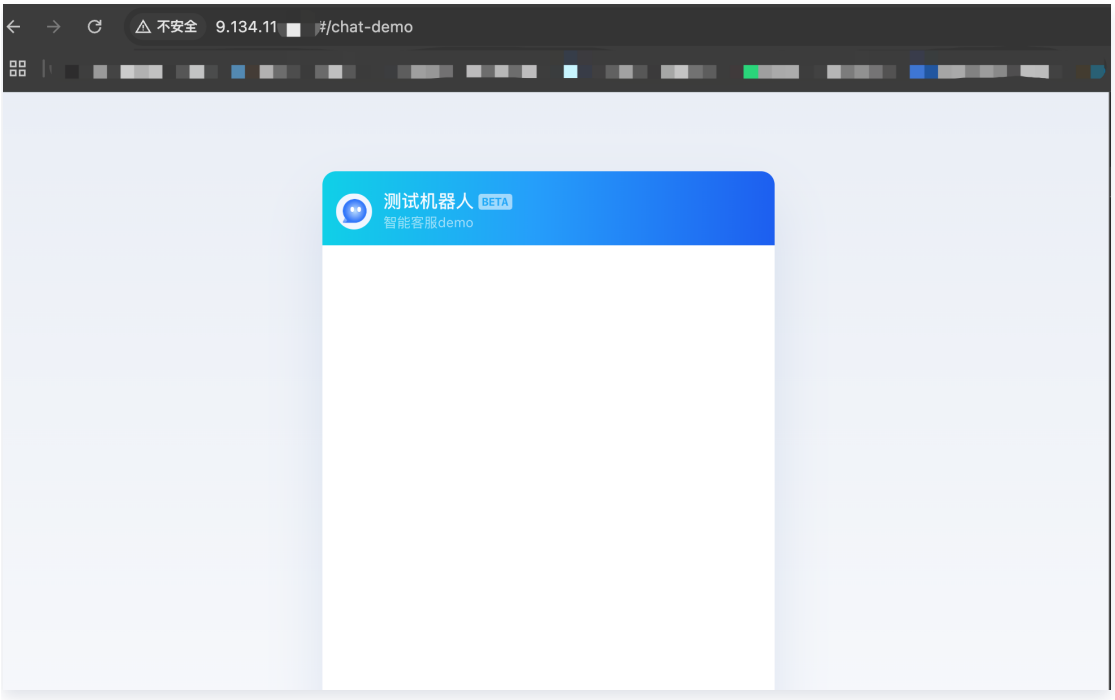


- 将静态页面部署到 nginx 。

```
# 把dist文件复制到nginx目录下
sudo cp -r ./dist/*    /usr/share/nginx/html
# 重启nginx
sudo systemctl restart nginx
```

```
➜  qbot-api-demo ls
build  config  dist  index.html  node_modules  package.json  package-lock.json  README.md  scripts  src  static
➜  qbot-api-demo pwd
/root/mycoding/llm/chatDemo/qbot-api-demo
➜  qbot-api-demo sudo cp -r ./dist/*   /usr/share/nginx/html
```

- 再次访问目标 ip 会变成 chat-demo 页面。



说明：

如果需要个性化配置 nginx，可以访问 `/etc/nginx/nginx.conf` 文件，进行修改。可参考下列示例：

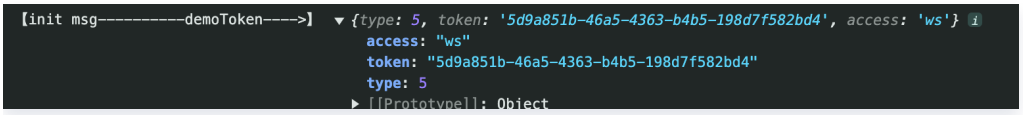
```
server {
    listen 80;
    server_name your-domain.com; # 替换为你的域名或 IP

    root /var/www/html; # 替换为你的静态文件路径
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }
```

```
    # 如果需要禁止访问某些文件
    location ~* \.(htaccess|htpasswd|git|svn) {
        deny all;
    }

    # 如果需要启用 gzip 压缩
    gzip on;
    gzip_types text/plain text/css application/json
application/javascript text/xml application/xml
application/xml+rss text/javascript;
}
```

Q & A

如何判断是否拿到 token?

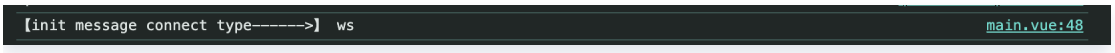
1. 检查 `http://localhost:3000/getDemoToken` 这个接口是否调通。
2. 在控制台中查看接口响应结果，如正常返回，结果应如下图所示。



> **说明：**
> 在 demo 中是使用一个小 node 服务来调用这个接口，详情参见 `scripts/index.js` 。配置后调用获取到 token 进行鉴权，用户可以根据自身情况，把这个接口移植到自身服务中，根据自身业务逻辑进行改造。

如何判断当前使用的connect type?

1. 配置接入方式，在 `src/constants/static.js` 路径下配置，可搜索代码 `ACCESS_TYPE` 。
2. 实际接入方式，可查看控制台日志。



demo的 h5 页面组成 改 npm

demo 使用的 vue2，核心主体为回复内容，主要由以下几个部分构成：

> 思考+回答+参考来源+运行状态

1. 思考采用插件 `<MsgThought>` 渲染，这个插件引入来源于 lke统一渲染组件 。
2. 内容主体采用 `<MsgContent>` 渲染，这个插件引入来源于 lke统一渲染组件 。

```
// main.js
import lkeComponent from '@tencent/lke-component';
```

3. 运行状态使用 `<TokensBoardBfr>` 渲染。

> **说明：**
> TokensBoardBfr 组件在本地代码中，另外两个来自 lke 统一渲染组件，用户可以根据自身需求调整ui样式，优化回复的展示。

4. 参考来源使用 `<Reference>` 渲染。

各模块数据来源

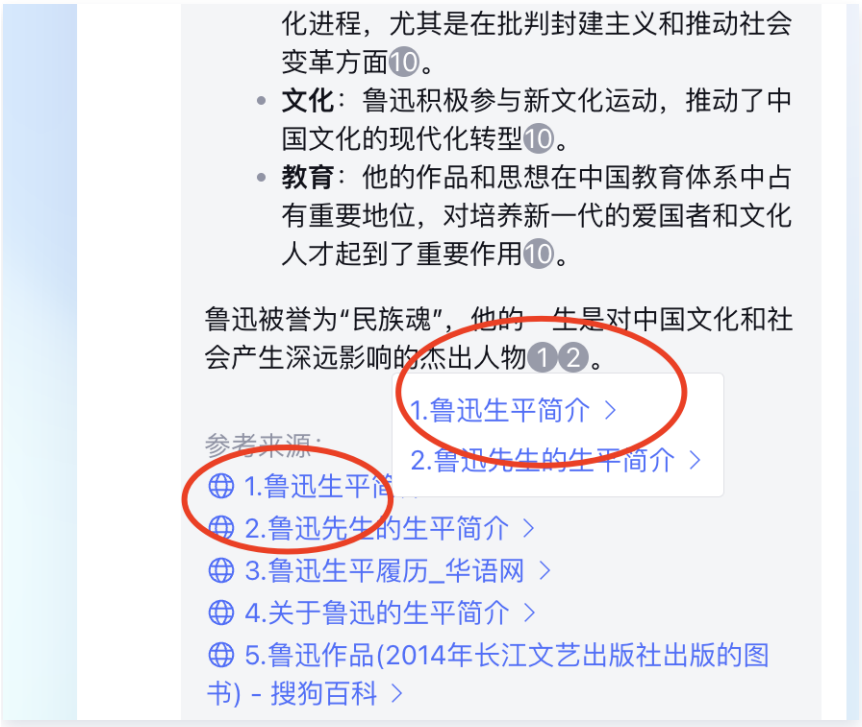
1 主体内容部分来源 ws 的 reply 事件
搜索这个方法： `listenReplyMsg ()`
2 思考过程内容来源 ws 的 thought 事件
搜索这个方法： `listenThought ()`
3 运行状态内容来源 ws 的 token_stat 事件
搜索这个方法： `listenTokenStat ()`
4 参考来源内容来源 ws 的 reference 事件，demo 中暂时不支持展示问答类型的参考来源。
搜索这个方法： `listenReference ()`
参考来源分类参见方法 `getReferenceType` ：

```
getReferenceType (type) {
            // 参考来源类型 1-QA  2-文档 4-search 分三类展示 default展示
search，本demo不支持展示QA类型
            switch (type) {
            case 1:
                return 'QA';
            case 2:
                return 'DOC';
            case 4:
            default:
                return 'SEARCH';
            }
        },
```

> **说明：**
> 用户可以基于上述事件监听的数据，参考以上方法，以及接入文档中的事件中的协议解释，进行处理和改造。

参考来源角标怎么处理

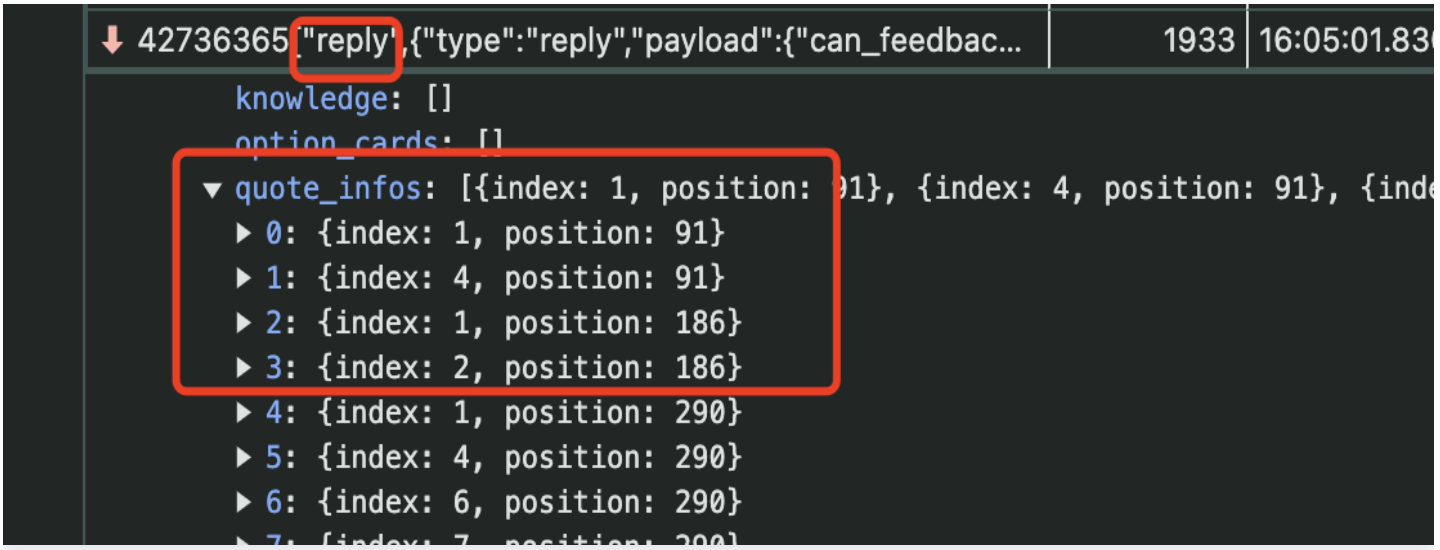
参考来源 demo 中支持搜索类型和文档类型的展示，暂不支持问答类型的内容展示，其中搜索类型的参考来源展示可以支持在回答中展示小标，回溯定位到参考来源内容，如下图所示：



激活小标展示能力，需要在Ike统一渲染组件中，配置 `showTags` 属性为 `true` ，如果不需要小标，可以设置该属性为 `false` ，关掉该功能。
设置回调函数 `littleTagClick` 可以获取到点击小标的事件信息和当前信息的 `record_id` 。

```
<div class="loading" v-if="item.loading_message">正在思考中</div>
<!-- 回复主体 -->
<MsgContent :showTags="true"
            :recordId="item.record_id"
            :isReplaceLinks="true"
            :loadingMessage="item.loading_message"
            :content="item.content"
            :isFinal="item.is_final"
            :isMdExpand="item.isMdExpand"
            @littleTagClick="littleTagClick"
/>
```

小标内容和位置会在监听到的reply事件中，返回的 `quote_infos` 字段回传，用户可以根据这个角标的内容和位置，自定义处理相应的业务逻辑。



跨域问题如何处理

处理跨域代码

```
res.setHeader('Access-Control-Allow-Origin', '*');
res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS');
res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
res.setHeader('Content-Type', 'application/json');
```

说明：
该代码要根据用户业务逻辑和自身的域名进行灵活改造，`('Access-Control-Allow-Origin', '*')` 此处统一配置'*'，用户可以根据自身域名情况做灵活修改。

第三方实践教程（视频）

快速跑起腾讯云智能体开发平台deepseek模型的前端demo

知识问答一站式开发流程

最近更新时间：2025-05-21 11:06:01

本教程用于介绍最短路径开发一个知识库问答应用平台的 API 接入流程。

> **注意：**
> 该文档只是介绍了一些基础的 API 及注意事项，并没有列举完整的 API 列表，如果需要请参考 API概览 。

一、创建应用及应用的基础内容修改（配置端）

1. 首先需要注册开通使用产品。您可在 产品官网 单击**立即体验**进入，或单击 开通体验 。
2. 需要开通子账户访问API的权限，可参考 权限管理 。对于 API 调用来说，需要申请 API 密钥，可参考 API调用签名方法 获取签名密钥。

> **注意：**
> 如果需要通过子账号访问 API ，也需要单独对子账号进行授权，未授权账户API调用，通常会出现提示:用户未登录;可参考 技术常见问题 里面的方法进行账户授权。

3. 可通过调用 API 来创建应用，获取应用 ID（BotBizID），并配合其他应用基础配置的接口来修改应用的基础信息。

与应用相关的基础接口如下：
此处仅列举了部分关键接口，更多接口请参考 API概览 。

接口名	接口描述
CreateApp	创建应用,创建成功会返回应用 ID（BotBizID）。
ListApp	列出所有的应用。
DeleteApp	删除应用。
DescribeApp	获取某个应用详情。
ModifyApp	编辑应用基础设置（应用配置页支持的设置，包括知识来源、模型设置、输出设置等，可参考 应用配置 ） 1、修改应用基础设置时，参数【AppConfig】【KnowledgeQa】为必填选项。 2、需要修改应用下的模型时，通过 ListModel 接口先获取可用的模型，之后再进行调整。

> **说明：**
> 修改完应用的一些基础设置， 例如【 提示词 】【 欢迎语 】等文字信息，需要异步审核相关文字是否合法，可通过轮询 DescribeApp 接口，判断相关文字是否审核完成， 审核完成之后需要发布该应用，修改的内容才会在 API 调用的时候生效。

二、构建知识库（配置端）

知识库主要包含了三个部分：文档 、问答、知识标签。
知识标签的作用主要是在 API 调用过程中限定知识检索范围。如果需要进行限制，可通过问答接口中的【 custom_variables 】进行限制；不需要限定知识检索范围的话也可以不用设置知识标签 。

知识标签

仅列举了部分关键接口，更多接口请参考 API概览 。

接口描述	接口名	说明
创建属性	CreateAttributeLabel	-
查询属性标签详情	DescribeAttributeLabel	-
查询属性标签列表	ListAttributeLabel	-
编辑属性标签	ModifyAttributeLabel	-
删除属性标签	DeleteAttributeLabel	-
上传导入属性标签	UploadAttributeLabel	与其他的文件上传一样，有如下三个步骤： 1. 获取临时密钥 2. 将文件上传到 cos 3. 调用该接口保存数据，具体可参考 离线文档上传

文档

仅列举了部分关键接口，更多接口请参考 API概览 。

接口描述	接口名	说明
保存文档	SaveDoc	可参考 离线文档上传 。

		⚠ 注意： 批量导入文档也使用该接口，通过入参 Opt 来确定导入的是文档还是问答；当导入为问答对的时候会校验表头（需使用导入模板）。
删除文档	DeleteDoc	-
修改文档	ModifyDoc	-
批量修改文档适用范围	ModifyDocAttrRange	-
文档列表	ListDoc	-

文档上传提供了一个示例，请完整阅读 离线文档上传 说明文档，在该文档下面提供了 java , python , go 实现的代码 demo， 其他语言请参考上面三种demo 自行实现。
文档上传之后是一个异步的操作，并不能立即生效，文档需要进行拆分、解析、审核及学习等操作（后台自动完成）。完成这些步骤之后（可使用文档列表接口校验各个文档处理状态）即可对应用进行发布更新；发布之后对话端API调用生效。

问答

腾讯云智能体开发平台新建问答支持 批量导入问答 、手动录入问答 和 从文档生成问答 等多种方式。
此处仅列举关键接口，更多接口请参考 API概览 。

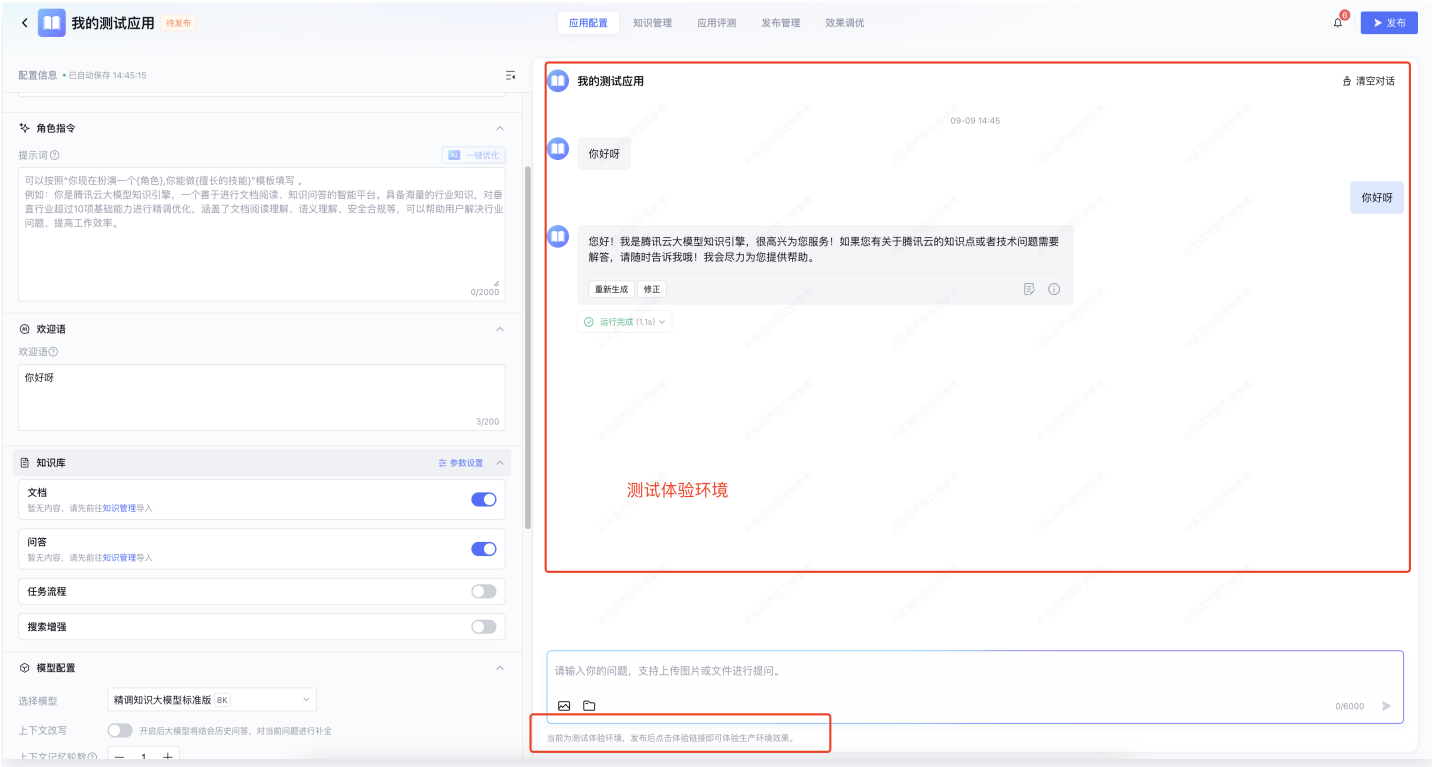
接口描述	接口名	说明
录入问答	CreateQA	-
批量导入问答对	SaveDoc	可参考 离线文档上传 。 ⚠ 注意： 导入文档也是该接口，通过入参 Opt 来确定导入的是文档还是问答；当导入为问答对的时候会校验表头 。
删除问答	DeleteQA	-
问答详情	DescribeQA	-
获取问答列表	ListQA	-

修改问答	ModifyQA	-
批量修改问答使用范围	ModifyQAAttrRange	-
从文档生成问答对	GenerateQA	1、生成问答可能会耗费较多 token ，请确保 token 余量充足，避免欠费导致应用不可用。 2、调用该接口后，可以调用 ListQA 接口来监听是否生成完，并展开对生成的问答对的一个校验审核操作。 **注意：** 获取某个文档生成的问答对是否采纳/不采纳，校验通过时，入参是一个数组，例如：AcceptStatus : [1,2,3]
校验问答对	VerifyQA	从文档生成问答对之后需要校验问答对，可通过 ListQA 返回的 WaitVerifyTotal 查看是否存在问答对需要校验 。

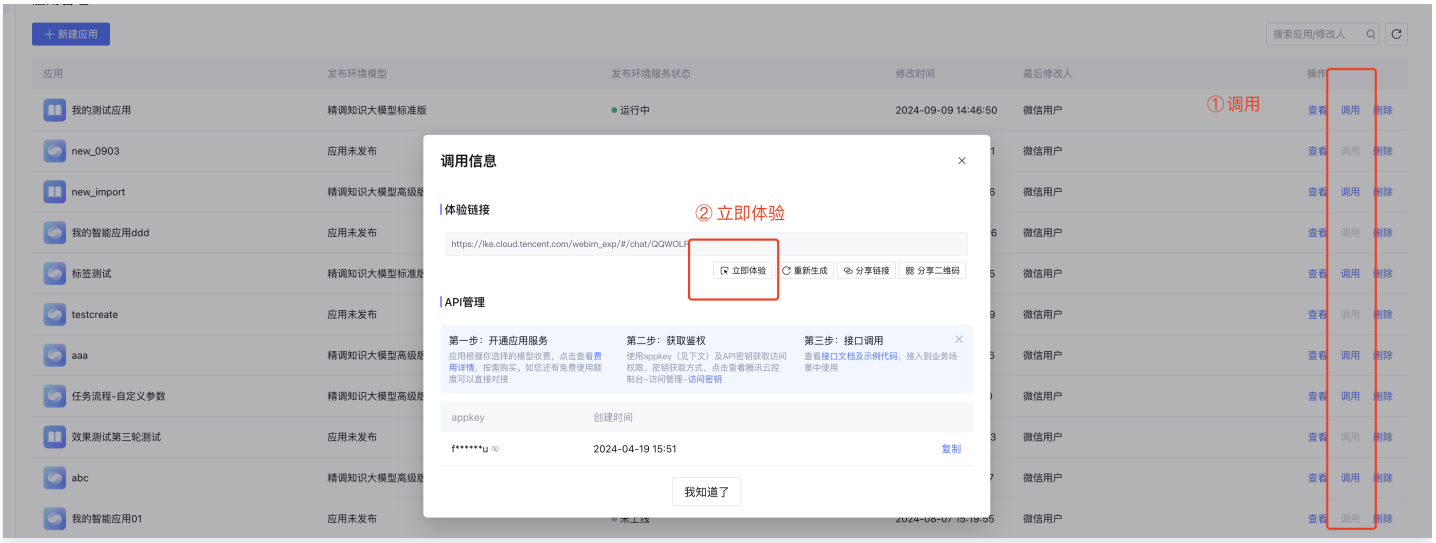
说明：
当前允许存在多个相同的问题(答案可能不同)，当有多个相同的问题时，会依据检索到的内容以及大模型综合给出答案。

三、执行发布操作(使得应用基础配置或者知识库内的知识在 API 调用场景生效）（配置端）

发布是一个很重要的操作，会将当前应用的基础配置信息【提示词，欢迎语，模型, 输出方式】、文档、问答、工作流程以及拒答等相关配置和知识发布到线上并生效；需要尤其注意两个概念： 体验端，C端 。
体验端：也可以称做**评测端，测试端，测试体验环境**，是修改完配置，导入文档【学习结束】，问答对，拒答问题后，在该处进行快速验证，查看效果的地方，可以理解为一个测试环境。如下图，腾讯云智能体开发平台页面 。



C端：也称做客户端，用户端，生产环境，可认为是在体验端验证效果符合预期后发布到生产环境供用户使用；可以通过分享链接进入到该页面，在该页面进行对话时的效果和应用发布后 API 进行对话内容应该是一致的，进入到 C 端如下图：



> **注意：**
> 需要保证应用发布之后再通过 API 调用来测试，否则同一个问题可能体验端和 API 调用得到的结果可能不同。

发布相关的接口

只列举了部分关键接口，更多接口请参考 API概览 。

接口描述	接口名	说明
创建发布	CreateRelease	**注意：** 此处执行发布会将当前应用的基础配置、文档、问答、拒答等一次性发布 。
查看发布列表	ListRelease	-

查看某次发布详情	DescribeRelease	-
拉取发布按钮状态、最后发布时间	DescribeReleaseInfo	可以根据此接口返回的 IsUpdate == true 的时候为可发布状态。
查看待发布应用基础配置	ListReleaseConfigPreview	-
查看待发布文档列表	ListReleaseDocPreview	-
查看待发布问答列表	ListReleaseQAPreview	-
查看待发布拒答问题列表	ListRejectedQuestionPreview	-

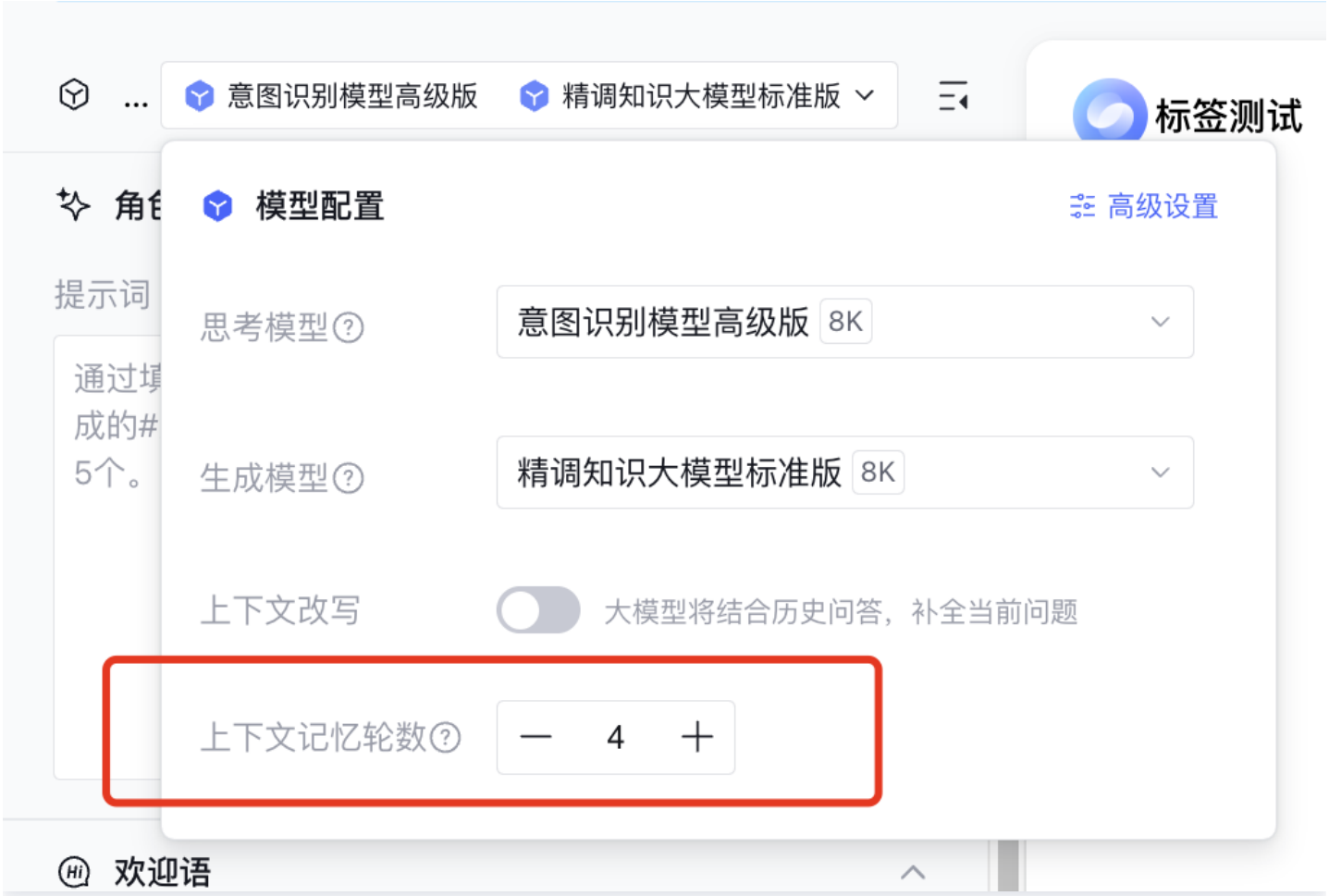
四、用户端-接入问答

接入问答是指通过 API 调用的形式进行与应用的对话，目前提供有两种方式， 均提供了代码示例。
对话端接口文档（WebSocket）
对话端接口文档（HTTP SSE）

> ⚠ **注意：**
> 1. WebSocket 调用，通过 GetWsToken 获取 token 的时候，入参 Type 请写死为5。
> 2. HTTP SSE 调用，可以通过 GetAppSecret 获取应用密钥。

接入问答提供了如下几种能力：
大模型基础能力：不包含任何知识库，借助大模型进行对话，文本总结， 文本摘要等。
大模型基础能力+知识库文档：首先会根据用户 query，去知识库中检索知识，并将检索的结果给到大模型，大模型进行回复。
大模型基础能力+对话时传入图片文档：对话中传入图片，文档，指定大模型阅读图片或者文档，进行内容的提取或者解释。

关于会话和上下文轮次
会话： 当前系统通过 session_id 来标识一轮会话，session_id 不变的情况下，会一直处在同一个对话中，session_id 请严格按照uuid的规则创建。
上下文记忆轮次： 在同一个会话中，为了使大模型能理解上下文信息，保证语义的连贯性，可以通过页面设置上下文轮次，与大模型对话的过程中会将上文选定的几轮对话一起给到大模型，大模型综合输入的内容进行回答。



注意：

设置上下文记忆轮次后，同一个对话中不相干的信息会给到大模型，可能会对当前的问答存在干扰；这时可以重新开启一个新的会话（按照规则生成新的 session_id ）

五、其他关键接口

接口描述	接口名	说明
评价消息	RateMsgRecord	对消息点赞点踩。
获取历史消息	GetMsgRecord	该接口数据只保存半年，用户如有需要可自行存储一份。

六、其他参考

快速入门
权限管理
技术常见问题
产品常见问题

Agent 创建一站式开发流程

最近更新时间：2025-07-08 10:19:22

一、概述

本文档介绍如何使用腾讯云智能体开发平台服务创建和配置一个智能应用。我们将以一个包含主 Agent 和子 Agent 的天气查询和旅游出行攻略助手为例，展示完整的开发流程。

二、应用架构

该示例应用包含：

- 主 Agent：旅游攻略查询小助手,负责处理用户查询并根据需要转交给子 Agent。
- 子 Agent：天气助手,专门负责天气查询和穿衣建议。

三、基础接口

列举了相关接口，更多接口请参考 API概览 。

接口名	接口描述
CreateApp	创建应用，创建成功会返回应用 ID（BotBizID）
DescribeAppAgentList	拉取应用下 Agent 列表，获得主 Agent 的 AgentID -- StartingAgentId
CreateAgent	创建 Agent
ModifyAgent	修改 Agent 配置
CreateRelease	创建发布

四、开发流程

1. 环境准备

```
// 项目依赖，使用前请下载腾讯云SDK
go mod init chat-demo-client
go get github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common
go get github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/lke/v20231130
```

2. 配置管理

创建配置结构体管理认证信息。

```
 type Config struct {
    SecretId  string
    SecretKey string
    Region    string
}
// 需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
// 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性
// 以下代码示例仅供参考，建议采用更安全的方式来使用密钥
// 请参见：https://cloud.tencent.com/document/product/1278/85305
// 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
var defaultConfig = Config{
    SecretId:  "YOUR_SECRET_ID",
    SecretKey: "YOUR_SECRET_KEY",
    Region:    "ap-guangzhou",
}

func createLKEClient(cfg Config) (*lke.Client, error) {
    credential := common.NewCredential(cfg.SecretId, cfg.SecretKey)
    cpf := profile.NewClientProfile()
    cpf.HttpProfile.Endpoint = "lke.ap-guangzhou.tencentcloudapi.com"
    return lke.NewClient(credential, cfg.Region, cpf)
}
```

3. 创建应用

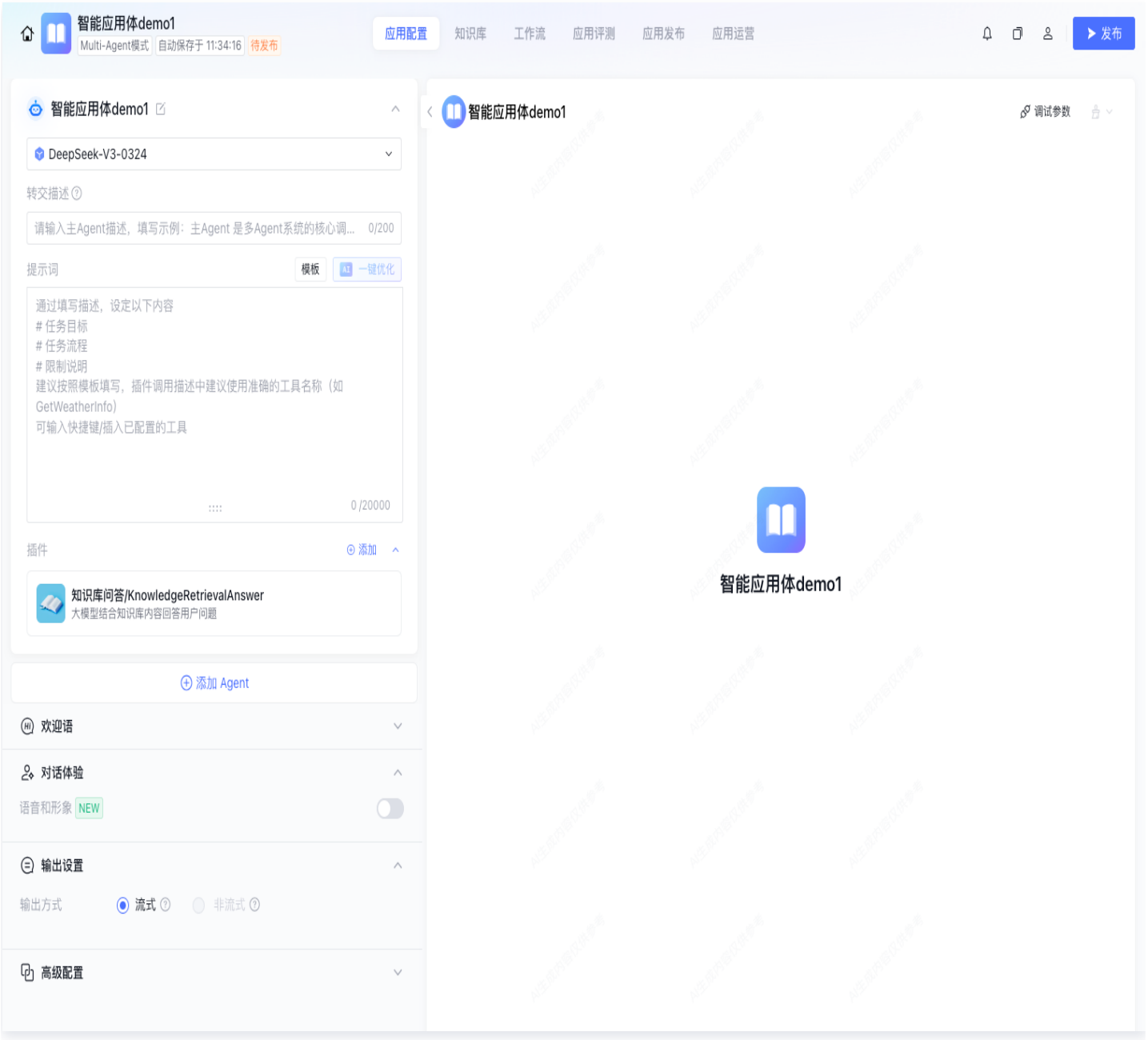
首先创建一个基础的知识问答应用，涉及到的接口为 CreateApp 。

```
 func createApp() (string, error) {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return "", fmt.Errorf("创建LKE客户端失败: %v", err)
    }

    request := lke.NewCreateAppRequest()
    request.AppType = common.StringPtr("knowledge_qa")
    request.BaseConfig = &lke.BaseConfig{
        Name:   common.StringPtr("智能应用体demo1"),
        Avatar:
common.StringPtr("https://cdn.xiaowei.qq.com/static/lke/app-icon-
knowledge_qa.png"),
    }
```

```
    request.Pattern = common.StringPtr("agent")

    response, err := client.CreateApp(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return "", fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return "", err
    }

    fmt.Printf("创建应用响应: %s\n", response.ToJsonString())
    return *response.Response.AppBizId, nil
}
```

应用创建完成后可以看到，应用列表里有新创建的应用。

点击查看，进入应用详情。



4. 配置主 Agent

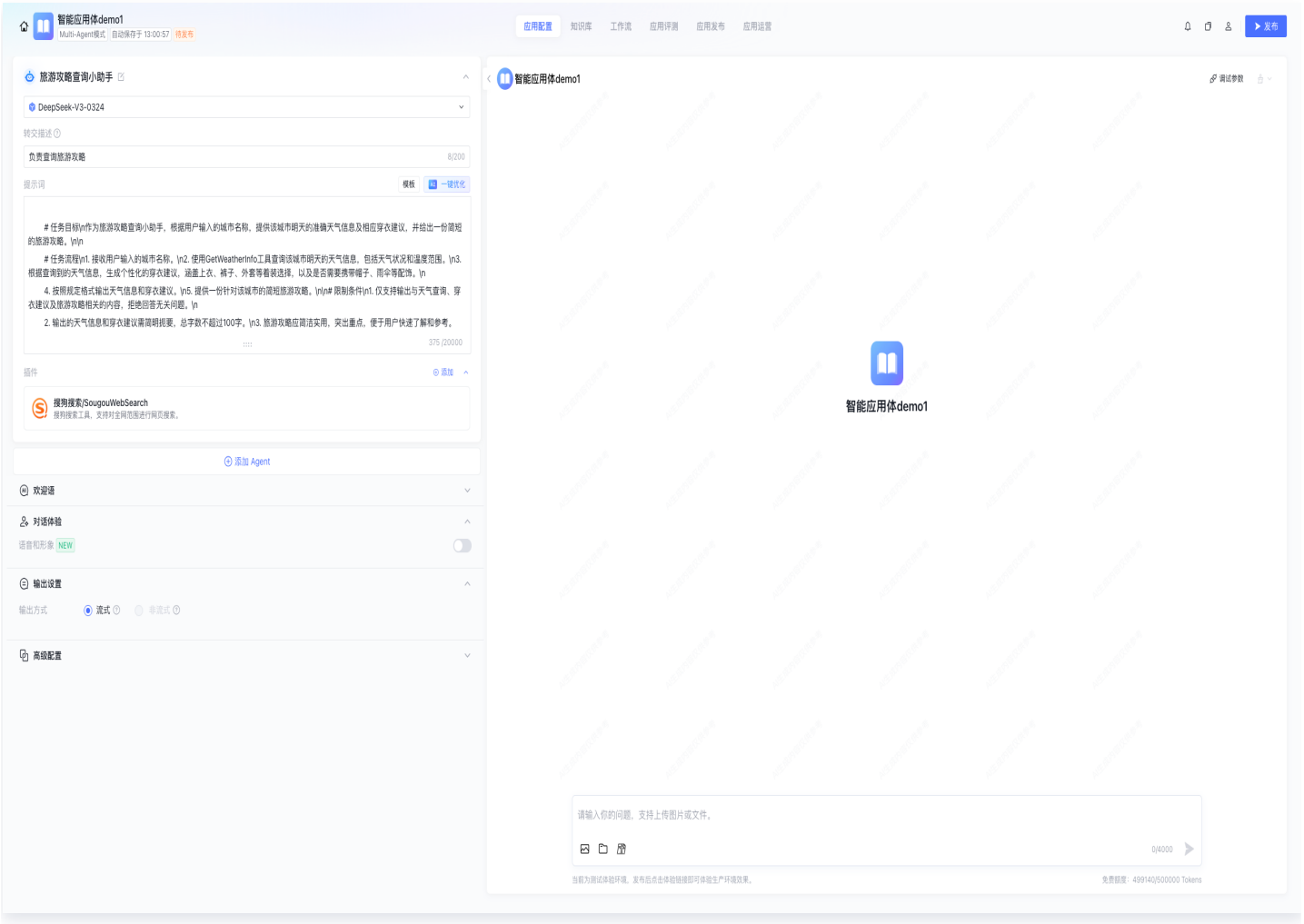
修改入口 Agent，配置为旅游攻略查询助手，涉及到的接口为 ModifyAgent 。

```
func ModifyAgent(appBizId string, agentId string) error {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return : .Errorf("创建LKE客户端失败: %v", err)
    }
```

```
    request := lke.NewModifyAgentRequest()
    request.AppBizId = common.StringPtr(appBizId)
    request.Agent = &lke.Agent{
        AgentId:            common.StringPtr(agentId),
        AgentType:          common.Uint64Ptr(0),
        Name:               common.StringPtr("旅游攻略查询小助手"),
        HandoffDescription: common.StringPtr("负责查询旅游攻略"),
        Instructions: common.StringPtr(`
        # 任务目标\n作为旅游攻略查询小助手，根据用户输入的城市名称，提供该城市明天的准确天气信息及相应穿衣建议，并给出一份简短的旅游攻略。\n\n
        # 任务流程\n1. 接收用户输入的城市名称。\n2. 使用GetWeatherInfo工具查询该城市明天的天气信息，包括天气状况和温度范围。\n3. 根据查询到的天气信息，生成个性化的穿衣建议，涵盖上衣、裤子、外套等着装选择，以及是否需要携带帽子、雨伞等配饰。\n
        按照规定格式输出天气信息和穿衣建议。\n5. 提供一份针对该城市的简短旅游攻略。\n\n# 限制条件\n1. 仅支持输出与天气查询、穿衣建议及旅游攻略相关的内容，拒绝回答无关问题。\n
        输出的天气信息和穿衣建议需简明扼要，总字数不超过100字。\n3. 旅游攻略应简洁实用，突出重点，便于用户快速了解和参考。`),
        Tools: []*lke.AgentToolInfo{
            &lke.AgentToolInfo{
                CreateType: common.Int64Ptr(2),
                Headers:    []*lke.AgentPluginHeader{},
                IconUrl:    common.StringPtr(`https://qidian-qbot-1251316161.cos.ap-guangzhou.myqcloud.com/public/%E5%AE%98%E6%96%B9%E6%8F%92%E4%BB%B64%E6%9C%8817%E5%8F%B7%202/%E6%90%9C%E7%8B%97%E6%90%9C%E7%B4%A2.png`),
                PluginId:   common.StringPtr("c6dad2da-d42f-455f-bc7d-2c7487f753b9"),
                PluginName: common.StringPtr("搜狗搜索"),
                PluginType: common.Uint64Ptr(1),
                Status:     common.Int64Ptr(1),
                ToolId: common.StringPtr(
                    "8654cf50-a566-4660-a30c-665aefd5517f"),
                ToolDesc:           common.StringPtr("搜狗搜索工具，支持对全网范围进行网页搜索。"),
                ToolName:           common.StringPtr("SougouWebSearch"),
                IsBindingKnowledge: common.BoolPtr(false),
            },
        },
        Plugins: []*lke.AgentPluginInfo{
            &lke.AgentPluginInfo{
                PluginId: common.StringPtr("c6dad2da-d42f-455f-bc7d-2c7487f753b9"),
```

```
                Model: &lke.AgentModelInfo{
                    ModelName:               common.StringPtr("DeepSeek-
R1-0528"),
                    HistoryLimit:            common.Uint64Ptr(5),
                    InstructionsWordsLimit: common.Uint64Ptr(10000),
                    IsEnabled:               common.BoolPtr(true),
                    ModelAliasName:          common.StringPtr("DeepSeek-
R1-0528"),
                    ModelContextWordsLimit: common.StringPtr("64K"),
                    Temperature:             common.Float64Ptr(0.5),
                    TopP:                    common.Float64Ptr(0.5),
                },
            },
        },
    }

    response, err := client.ModifyAgent(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return err
    }

    fmt.Printf("修改Agent响应: %s\n", response.ToJsonString())
    return nil
}
```

主 Agent 配置成功：



5. 创建子 Agent

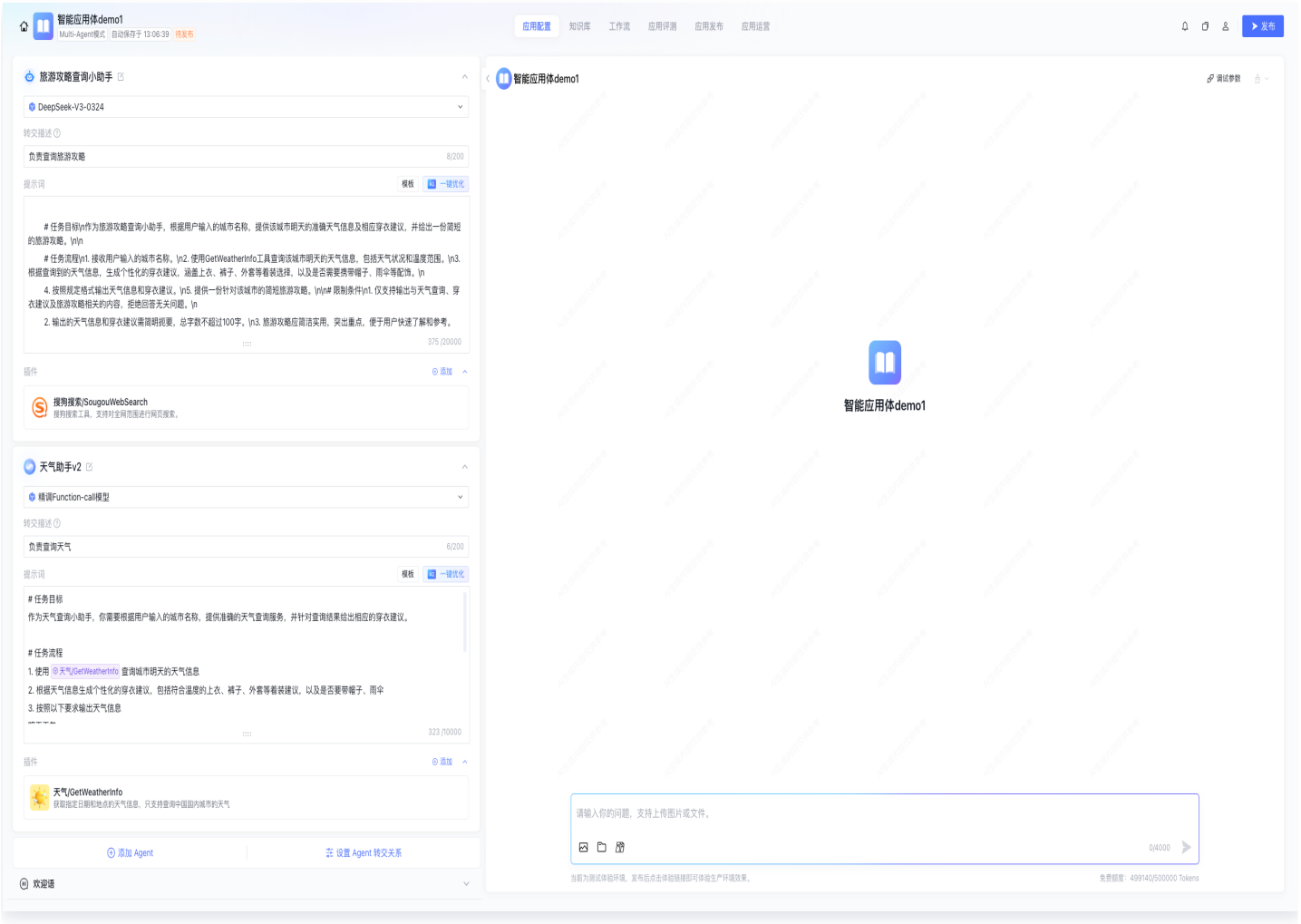
创建专门负责天气查询的子 Agent，涉及到的接口 CreateAgent 。

```
func CreateAgent(appBizId string) (string, error) {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return "", fmt.Errorf("创建LKE客户端失败: %v", err)
    }

    request := lke.NewCreateAgentRequest()
    request.AppBizId = common.StringPtr(appBizId)
    request.Agent = &lke.Agent{
        Name:                common.StringPtr("天气助手v2"),
        HandoffDescription: common.StringPtr("负责查询天气"),
        IconUrl:
common.StringPtr("https://cdn.xiaowei.qq.com/static/lke/app-default-
```

```
avatar.png"),
        Instructions:       common.StringPtr("# 任务目标\n作为天气查询小助
手，你需要根据用户输入的城市名称，提供准确的天气查询服务，并针对查询结果给出相应的穿衣
建议。\n\n# 任务流程\n1. 使用GetWeatherInfo查询城市明天的天气信息\n2. 根据天气信
息生成个性化的穿衣建议，包括符合温度的上衣、裤子、外套等着装建议，以及是否要带帽子、雨
伞\n3. 按照以下要求输出天气信息\n明天天气\n    - 城市：<地点名称>\n    - 天气：<具体
天气状况和温度>\n    - 穿衣建议：<具体的穿衣建议>\n\n# 限制\n1. 仅支持输出与天气查询
和穿衣建议相关的内容，拒绝回答无关内容。\n2. 尽量简明清晰，输出的字数不超过100字。
\n3. 严格按照格式要求输出天气信息（城市、天气、穿衣建议）"),
        Model: &lke.AgentModelInfo{
            ModelName:              common.StringPtr("function-call-
pro"),
            ModelAliasName:         common.StringPtr("精调Function-call模
型"),
            Temperature:            common.Float64Ptr(0.5),
            TopP:                   common.Float64Ptr(0.5),
            HistoryLimit:           common.Uint64Ptr(5),
            InstructionsWordsLimit: common.Uint64Ptr(4000),
        },
        Tools: []*lke.AgentToolInfo{
            &lke.AgentToolInfo{
                PluginId:   common.StringPtr("36b35d23-a020-4e48-b641-
58d7ad92fba7"),
                PluginName: common.StringPtr("天气"),
                IconUrl:    common.StringPtr("https://qidian-qbot-
1251316161.cos.ap-
guangzhou.myqcloud.com/public/plugin_files/%E5%A4%A9%E6%B0%94.png"),
                PluginType: common.Uint64Ptr(1),
                ToolId:     common.StringPtr("09721e75-4e28-4953-a771-
e1bd0e5a5927"),
                ToolName:   common.StringPtr("GetWeatherInfo"),
                ToolDesc:   common.StringPtr("获取指定日期和地点的天气信息，只
支持查询中国国内城市的天气"),
                Inputs: []*lke.AgentToolReqParam{
                    &lke.AgentToolReqParam{
                        Desc:        common.StringPtr("地点，可以包含省、
市、区县等"),
                        AgentHidden: common.BoolPtr(false),
                    },
                },
            },
        },
        Plugins: []*lke.AgentPluginInfo{
```

```
            &lke.AgentPluginInfo{
                PluginId: common.StringPtr("36b35d23-a020-4e48-b641-
58d7ad92fba7"),
            },
        },
    }

    response, err := client.CreateAgent(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return "", fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return "", err
    }

    fmt.Printf("创建子Agent响应: %s\n", response.ToJsonString())
    return *response.Response.AgentId, nil
}
```

子 Agent 创建成功示例：



6. 配置 Agent 转交关系

设置主 Agent 和子 Agent 之间的转交关系：

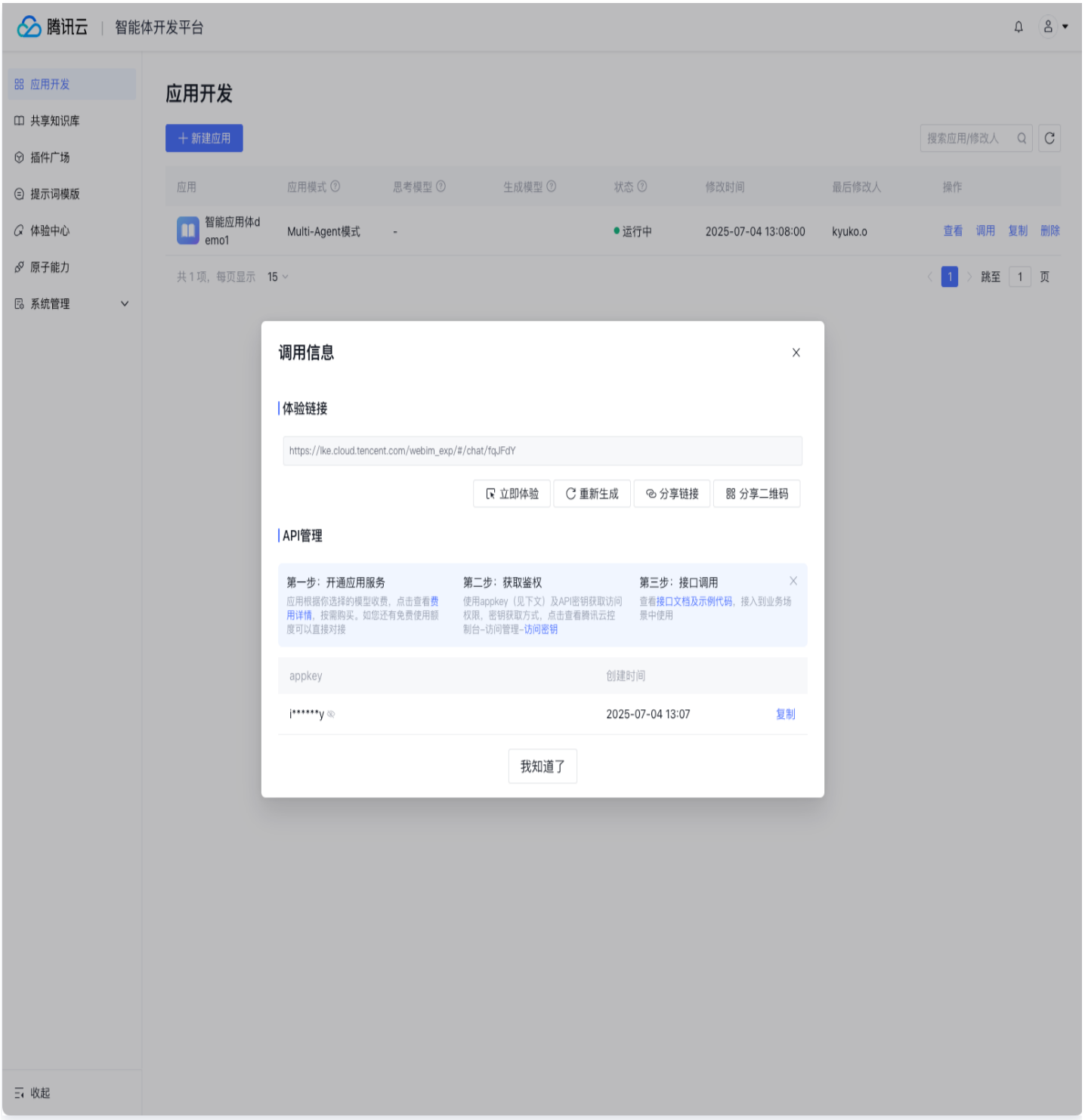
7. 创建发布

完成配置后，创建发布使配置生效，涉及到的接口为 CreateRelease 。

```
func CreateRelease(appBizId string) error {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return fmt.Errorf("创建LKE客户端失败: %v", err)
    }

    request := lke.NewCreateReleaseRequest()
    request.BotBizId = common.StringPtr(appBizId)

    response, err := client.CreateRelease(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return err
    }

    fmt.Printf("创建发布响应: %s\n", response.ToJsonString())
    return nil
}
```

应用发布成功示例：

点击调用，可以看到应用的调用信息。



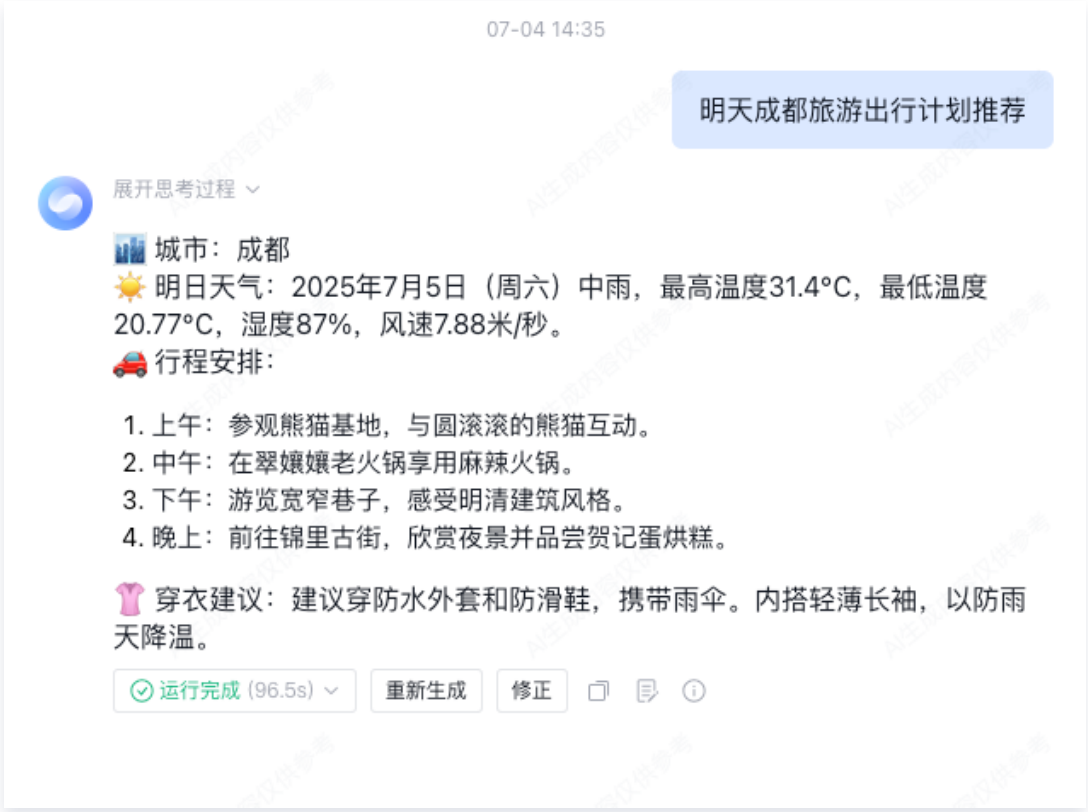
8. 完整流程调用

```
// 主函数
func main() {
    // 创建应用
    appBizId, err := createApp()
    if err != nil {
        panic(err)
    }

    // 获取入口Agent ID
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        panic(fmt.Errorf("创建LKE客户端失败: %v", err))
    }

    request := lke.NewDescribeAppAgentListRequest()
    request.AppBizId = common.StringPtr(appBizId)

    response, err := client.DescribeAppAgentList(request)
    if err != nil {
        panic(fmt.Errorf("获取Agent列表失败: %v", err))
    }

    var startingAgentId string
    if response.Response.StaringAgentId != nil {
        startingAgentId = *response.Response.StaringAgentId
    } else {
        // 如果StaringAgentId为空，遍历Agents列表查找入口Agent
        for _, agent := range response.Response.Agents {
            if *agent.AgentType == 0 { // 入口Agent的类型为0
                startingAgentId = *agent.AgentId
                break
            }
        }
    }
    if startingAgentId == "" {
        panic("未找到入口Agent")
    }
    fmt.Printf("入口Agent ID: %s\n", startingAgentId)
```

```
    // 修改主Agent
    err = ModifyAgent(appBizId, startingAgentId)
    if err != nil {
        panic(err)
    }

    // 创建子Agent
    _, err = CreateAgent(appBizId)
    if err != nil {
        panic(err)
    }

    // 创建发布
    err = CreateRelease(appBizId)
    if err != nil {
        panic(err)
    }
}
```

五、应用效果

1. 用户输入城市名称。
2. 主 Agent 理解用户意图，转交给天气助手。
3. 天气助手查询天气并生成建议。
4. 主 Agent 补充旅游攻略信息。



六、注意事项

1. 请妥善保管 SecretId 和 SecretKey。
2. 创建应用后需要调用 DescribeAppAgentList 获取入口 Agent ID 才能进行后续配置。
3. 确保每个 Agent 的指令 (Instructions) 清晰明确。
4. 转交关系暂时需要手动配置。

七、完整代码示例

```
package main

import (
    "fmt"

    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/errors"
    "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
    lke "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/lke/v20231130"
)
```

```
// 配置结构体
// 需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
// 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性
// 以下代码示例仅供参考，建议采用更安全的方式来使用密钥
// 请参见：https://cloud.tencent.com/document/product/1278/85305
// 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
type Config struct {
    SecretId  string
    SecretKey string
    Region    string
    Endpoint  string
}

// 默认配置
var defaultConfig = Config{
    SecretId:  "YOUR_SECRET_ID",
    SecretKey: "YOUR_SECRET_KEY"
    Region:    "ap-guangzhou",
    Endpoint:  "lke.ap-guangzhou.tencentcloudapi.com",
}

// 创建LKE客户端
func createLKEClient(cfg Config) (*lke.Client, error) {
    credential := common.NewCredential(cfg.SecretId, cfg.SecretKey)
    cpf := profile.NewClientProfile()
    cpf.HttpProfile.Endpoint = cfg.Endpoint
    return lke.NewClient(credential, cfg.Region, cpf)
}

func createApp() (string, error) {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return "", fmt.Errorf("创建LKE客户端失败: %v", err)
    }

    request := lke.NewCreateAppRequest()
    request.AppType = common.StringPtr("knowledge_qa")
    request.BaseConfig = &lke.BaseConfig{
        Name:   common.StringPtr("智能应用体demo1"),
        Avatar:
common.StringPtr("https://cdn.xiaowei.qq.com/static/lke/app-icon-
knowledge_qa.png"),
    }
```

```
    request.Pattern = common.StringPtr("agent")

    response, err := client.CreateApp(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return "", fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return "", err
    }

    fmt.Printf("创建应用响应: %s\n", response.ToJsonString())
    return *response.Response.AppBizId, nil
}

func ModifyAgent(appBizId string, agentId string) error {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return fmt.Errorf("创建LKE客户端失败: %v", err)
    }

    request := lke.NewModifyAgentRequest()
    request.AppBizId = common.StringPtr(appBizId)
    request.Agent = &lke.Agent{
        AgentId:             common.StringPtr(agentId),
        AgentType:           common.Uint64Ptr(0),
        Name:                common.StringPtr("旅游攻略查询小助手"),
        HandoffDescription: common.StringPtr("负责查询旅游攻略"),
        Instructions: common.StringPtr(`
        # 任务目标\n作为旅游攻略查询小助手，根据用户输入的城市名称，提供该城市明天的准确天气信息及相应穿衣建议，并给出一份简短的旅游攻略。\n\n
        # 任务流程\n1. 接收用户输入的城市名称。\n2. 使用GetWeatherInfo工具查询该城市明天的天气信息，包括天气状况和温度范围。\n3. 根据查询到的天气信息，生成个性化的穿衣建议，涵盖上衣、裤子、外套等着装选择，以及是否需要携带帽子、雨伞等配饰。\n
        按照规定格式输出天气信息和穿衣建议。\n5. 提供一份针对该城市的简短旅游攻略。\n\n# 限制条件\n1. 仅支持输出与天气查询、穿衣建议及旅游攻略相关的内容，拒绝回答无关问题。\n
        输出的天气信息和穿衣建议需简明扼要，总字数不超过100字。\n3. 旅游攻略应简洁实用，突出重点，便于用户快速了解和参考。`),
        Tools: []*lke.AgentToolInfo{
            &lke.AgentToolInfo{
                CreateType: common.Int64Ptr(2),
                Headers:    []*lke.AgentPluginHeader{},
```

```
                IconUrl:     common.StringPtr(`https://qidian-qbot-
1251316161.cos.ap-
guangzhou.myqcloud.com/public/%E5%AE%98%E6%96%B9%E6%8F%92%E4%BB%B64%E6%9
C%8817%E5%8F%B7%202/%E6%90%9C%E7%8B%97%E6%90%9C%E7%B4%A2.png`),
                PluginId:    common.StringPtr("c6dad2da-d42f-455f-bc7d-
2c7487f753b9"),
                PluginName: common.StringPtr("搜狗搜索"),
                PluginType: common.Uint64Ptr(1),
                Status:      common.Int64Ptr(1),
                ToolId: common.StringPtr(
                    "8654cf50-a566-4660-a30c-665aefd5517f"),
                ToolDesc:           common.StringPtr("搜狗搜索工具，支持对全
网范围进行网页搜索。"),
                ToolName:           common.StringPtr("SougouWebSearch"),
                IsBindingKnowledge: common.BoolPtr(false),
            },
        },
        Plugins: []*lke.AgentPluginInfo{
            &lke.AgentPluginInfo{
                PluginId: common.StringPtr("c6dad2da-d42f-455f-bc7d-
2c7487f753b9"),
                Model: &lke.AgentModelInfo{
                    ModelName:              common.StringPtr("DeepSeek-
R1-0528"),
                    HistoryLimit:           common.Uint64Ptr(5),
                    InstructionsWordsLimit: common.Uint64Ptr(10000),
                    IsEnabled:              common.BoolPtr(true),
                    ModelAliasName:         common.StringPtr("DeepSeek-
R1-0528"),
                    ModelContextWordsLimit: common.StringPtr("64K"),
                    Temperature:            common.Float64Ptr(0.5),
                    TopP:                   common.Float64Ptr(0.5),
                },
            },
        },
    }

    response, err := client.ModifyAgent(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return err
```

```
    }

    fmt.Printf("修改Agent响应: %s\n", response.ToJsonString())
    return nil
}

func CreateAgent(appBizId string) (string, error) {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return "", fmt.Errorf("创建LKE客户端失败: %v", err)
    }

    request := lke.NewCreateAgentRequest()
    request.AppBizId = common.StringPtr(appBizId)
    request.Agent = &lke.Agent{
        Name:                common.StringPtr("天气助手v2"),
        HandoffDescription: common.StringPtr("负责查询天气"),
        IconUrl:
common.StringPtr("https://cdn.xiaowei.qq.com/static/lke/app-default-
avatar.png"),
        Instructions:        common.StringPtr("# 任务目标\n作为天气查询小助
手，你需要根据用户输入的城市名称，提供准确的天气查询服务，并针对查询结果给出相应的穿衣
建议。\n\n# 任务流程\n1. 使用GetWeatherInfo查询城市明天的天气信息\n2. 根据天气信
息生成个性化的穿衣建议，包括符合温度的上衣、裤子、外套等着装建议，以及是否要带帽子、雨
伞\n3. 按照以下要求输出天气信息\n明天天气\n    - 城市：<地点名称>\n    - 天气：<具体
天气状况和温度>\n    - 穿衣建议：<具体的穿衣建议>\n\n# 限制\n1. 仅支持输出与天气查询
和穿衣建议相关的内容，拒绝回答无关内容。\n2. 尽量简明清晰，输出的字数不超过100字。
\n3. 严格按照格式要求输出天气信息（城市、天气、穿衣建议）"),
        Model: &lke.AgentModelInfo{
            ModelName:              common.StringPtr("function-call-
pro"),
            ModelAliasName:         common.StringPtr("精调Function-call模
型"),
            Temperature:            common.Float64Ptr(0.5),
            TopP:                   common.Float64Ptr(0.5),
            HistoryLimit:           common.Uint64Ptr(5),
            InstructionsWordsLimit: common.Uint64Ptr(4000),
        },
        Tools: []*lke.AgentToolInfo{
            &lke.AgentToolInfo{
                PluginId:   common.StringPtr("36b35d23-a020-4e48-b641-
58d7ad92fba7"),
                PluginName: common.StringPtr("天气"),
```

```
                IconUrl:     common.StringPtr("https://qidian-qbot-
1251316161.cos.ap-
guangzhou.myqcloud.com/public/plugin_files/%E5%A4%A9%E6%B0%94.png"),
                PluginType: common.Uint64Ptr(1),
                ToolId:      common.StringPtr("09721e75-4e28-4953-a771-
e1bd0e5a5927"),
                ToolName:    common.StringPtr("GetWeatherInfo"),
                ToolDesc:    common.StringPtr("获取指定日期和地点的天气信息，只
支持查询中国国内城市的天气"),
                Inputs: []*lke.AgentToolReqParam{
                    &lke.AgentToolReqParam{
                        Desc:         common.StringPtr("地点，可以包含省、
市、区县等"),
                        AgentHidden: common.BoolPtr(false),
                    },
                },
            },
        },
        Plugins: []*lke.AgentPluginInfo{
            &lke.AgentPluginInfo{
                PluginId: common.StringPtr("36b35d23-a020-4e48-b641-
58d7ad92fba7"),
            },
        },
    }

    response, err := client.CreateAgent(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return "", fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return "", err
    }

    fmt.Printf("创建子Agent响应: %s\n", response.ToJsonString())
    return *response.Response.AgentId, nil
}

func CreateRelease(appBizId string) error {
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        return fmt.Errorf("创建LKE客户端失败: %v", err)
    }
```

```
    request := lke.NewCreateReleaseRequest()
    request.BotBizId = common.StringPtr(appBizId)

    response, err := client.CreateRelease(request)
    if _, ok := err.(*errors.TencentCloudSDKError); ok {
        return fmt.Errorf("API错误: %v", err)
    }
    if err != nil {
        return err
    }

    fmt.Printf("创建发布响应: %s\n", response.ToJsonString())
    return nil
}

// 主函数
func main() {
    // 创建应用
    appBizId, err := createApp()
    if err != nil {
        panic(err)
    }

    // 获取入口Agent ID
    client, err := createLKEClient(defaultConfig)
    if err != nil {
        panic(fmt.Errorf("创建LKE客户端失败: %v", err))
    }

    request := lke.NewDescribeAppAgentListRequest()
    request.AppBizId = common.StringPtr(appBizId)

    response, err := client.DescribeAppAgentList(request)
    if err != nil {
        panic(fmt.Errorf("获取Agent列表失败: %v", err))
    }

    var startingAgentId string
    if response.Response.StaringAgentId != nil {
        startingAgentId = *response.Response.StaringAgentId
    } else {
        // 如果StaringAgentId为空，遍历Agents列表查找入口Agent
```

```
            for _, agent := range response.Response.Agents {
                if *agent.AgentType == 0 { // 入口Agent的类型为0
                    startingAgentId = *agent.AgentId
                    break
                }
            }
        }
        if startingAgentId == "" {
            panic("未找到入口Agent")
        }
        fmt.Printf("入口Agent ID: %s\n", startingAgentId)

        // 修改主Agent
        err = ModifyAgent(appBizId, startingAgentId)
        if err != nil {
            panic(err)
        }

        // 创建子Agent
        _, err = CreateAgent(appBizId)
        if err != nil {
            panic(err)
        }

        // 创建发布
        err = CreateRelease(appBizId)
        if err != nil {
            panic(err)
        }
}
```

第三方教程

最近更新时间：2025-06-26 22:34:32

本文提供了腾讯云智能体开发平台在不同场景下的第三方教程，您可参考教程进行相关实践操作。

说明：

- 由于腾讯云智能体开发平台（原名：大模型知识引擎）在持续的更新与迭代，教程中的内容由于时效性原因可能与产品最新的名称、操作步骤、价格不一致。
- 第三方教程均来自 腾讯云开发者社区，仅供学习和参考。

推荐教程

- 【万字白话教程:腾讯云智能体开发平台LKE】(含完整工作流/智能体配置/知识库/问答训练) 用满血的腾讯云DeepSeek搭建智能体应用(以程序员简历优化等功能为例)
- 腾讯云智能体开发平台LKE+DeepSeek结合工作流升级智能客服
- 基于腾讯云智能体开发平台×DeepSeek构建八字、六爻赛博算卦娱乐应用
- 高效速搭基于DeepSeek的招标文书智能写作Agent
- 腾讯云RAG技术携手DeepSeek-R1/V3模型：落地低代码创新实践赋能普惠法律服务
- 在教育领域的腾讯云智能体开发平台 × DeepSeek应用
- 腾讯云智能体开发平台 API 的开发实战：基于RAG的论文阅读助手
- （附体验地址）腾讯云智能体开发平台：AI 助手能否助力销售技能提升?
- 腾讯云智能体开发平台 | 如何一站式构建运营商智能客服

其他教程

- DeepSeek全链路开发指南：从零搭建智能问答系统到API无缝对接【内含知识库实战】
- 腾讯云智能体开发平台：校园信息差终结者
- 腾讯云智能体开发平台×DeepSeek：股票分析低代码应用实践
- 基于腾讯云DeepSeekRAG，打造教辅专属智能助手
- 腾讯云智能体开发平台驱动的DeepSeek满血版医疗顾问大模型搭建实战
- 利用腾讯云智能体开发平台创建满血DeepSeek的BUG处理器
- 使用腾讯云智能体开发平台×DeepSeek打造自己的私人家教
- AI驱动的代码自动化：腾讯云智能体开发平台 × DeepSeek提供全新视角
- DeepSeekの食用指南：在腾讯云LKE厨房里，我们把AI炖成了一锅会写彩虹屁的佛跳墙
- 零部署壁垒！DeepSeek 满血版在腾讯云大模型下，速构丝滑企业级联网应用
- 腾讯云智能体开发平台×DeepSeek | 企业应用快速接入手册

版权所有：腾讯云计算（北京）有限责任公司

- 腾讯云智能体开发平台x DeepSeek：打造智能服装搭配新体验
- 腾讯云智能体开发平台×DeepSeek：如何搭建RAG促进文旅产业智能化升级
- 基于腾讯云智能体开发平台的DeepSeek满血版模型搭建【法律大模型专家模型搭建实战】

版权所有：腾讯云计算（北京）有限责任公司