文档中心

OpenSDK Javascript

产品文档

目录

UCloud SDK Javascript

快速开始

安装

使用 npm 安装(推荐):

```
$ npm install @ucloud-sdks/ucloud-sdk-js
```

初次使用

目前,SDK 使用 PublicKey/PrivateKey 作为唯一的鉴权方式,该公私钥可以从以下途径获取:

- UAPI 密钥管理

下面提供一个简单的示例:

```
const {Client} = require("@ucloud-sdks/ucloud-sdk-js");

function sleep(ms) {
return new Promise(resolve => setTimeout(resolve, ms))
}
```

```
async function main() {
// Build client
const client = new Client({
config: {
region: 'cn-bj2',
projectId: process.env.UCLOUD_PROJECT_ID || '',
},
credential: {
publicKey: process.env.UCLOUD_PUBLIC_KEY || '',
privateKey: process.env.UCLOUD_PRIVATE_KEY || '',
}
});
const zone = "cn-bj2-05";

let resp = null;

// Describe Image
try {
resp = await client.uhost().describeImage();
} catch (e) {
throw e;
}
const image = resp["ImageSet"][0];

// Create Instance
```

```
try {
resp = await client.uhost().createUHostInstance({
Name: "sdk-js-example",
Zone: zone,
ImageId: image["ImageId"],
LoginMode: "Password",
Password: new Buffer("UCloud1234!").toString('base64'),
CPU: 1,
Memory: 1024,
Disks: [{
Size: image["ImageSize"],
Type: "CLOUD_SSD",
IsBoot: "true",
}],
});
} catch (e) {
throw e;
}
}

main().catch(e => { console.error(e) })
```

将上述代码中 client 相关配置，以及主机的 image id 等，替换成自己的配置，即可创建一台云主机。

在该示例中，使用 SDK 完成了一个创建云主机的请求。至此，已经涵盖了 SDK 的基本核心用法，可以构建自己的脚本啦！

SDK 中的每一个 api 调用都有详细的注释文档，可以通过 Editor/IDE 跳转到具体的方法中查看(也可以 查看接口文档)，并根据 IDE 自动补全和报错信息继续探索 SDK 的用法。

如果需要了解这段代码提及但未完全覆盖的使用技巧，请参考：

- 通用配置，了解如何配置 SDK，如日志、重试、服务访问端点(公有云、专有云)等
- 错误处理，了解如何处理不同类型的 SDK 异常，包括参数错误，RetCode 不为 0 的业务异常等
- 请求中间件，了解如何拦截 SDK 发起的请求，并统一添加额外的逻辑。
- 泛化调用，如何调用 SDK 尚未支持的 API(不建议使用此类 API，因为没有兼容性保证)

获取更多示例

基于场景的示例

SDK 提供了部分基于场景的示例，并提供了对应的资源销毁逻辑，可以点击以下链接查看源码：

- 批量创建云主机

通用配置

了解如何配置 SDK，如日志、重试、服务访问端点（公有云、专有云）等。

配置项清单

配置	类型	描述
region	string	（必填）服务所在地域，可参考 地域和可用区列表
projectId	string	（选填）项目的唯一标识，用于组织资源，大多数资源都需要 ProjectId，如果是主账号或财务账号，默认值为默认 ProjectId，如果是子账号非财务账号，则该字段必须填写。
baseUrl	string	（选填）API 服务的访问端点，默认是 https://api.ucloud.cn
userAgent	string	（选填）UserAgent 是 SDK 客户端特有的属性，用于区分使用 SDK 的版本。User-Agent 的定义请参考 MDN。用户自定义的 UserAgent 将追加到 SDK 版本号的末尾。例如，“MyAPP/0.10.1” -> “Node/16.4.2 JS-SDK/0.1.0 MyAPP/0.10.1”
timeout	number	（选填）请求超时时间，默认 30s
maxRetries	number	（选填）最大重试次数. 默认重试 3 次。设置该值大于 0 将对网络和服务可用性问题进行自动重试，使用指数退避的重试间隔，并自动跳过资源创建类的接口。
logger	logger	（选填）自定义 Logger

进阶配置选项

修改默认日志

关闭日志：

```
const client = new Client({
config: {
// ...
logger: null,
},
// ...
});
```

访问专有云/渠道云或其它网关

```
const client = new Client({
config: {
// ...
region: "专有云地域",
baseUrl: "foo.api.ucloud.cn", // 替换成专有云网关
},
// ...
});
```

错误处理

了解如何处理不同类型的 SDK 异常，包括参数错误，RetCode 不为 0 的业务异常等。

```
try {
resp = await client.uhost().describeImage();
} catch (e) {
console.log(e.typ);
console.log(e.retCode);
console.log(e.requestId);
throw e;
}
```

请求中间件

了解如何拦截 SDK 发起的请求，并统一添加额外的逻辑。UCloud SDK 为请求提供了请求中间件的特性。

该特性允许在 请求/响应 的生命周期中添加自定义的逻辑。例如，Client 级别的中间件，可以拦截参数/响应字典:

```
client.useMiddleware({
request: function (ctx: Context) {
ctx.config.logger.info(ctx.request.toObject());
return ctx.request;
},
response: function (ctx: Context) {
ctx.config.logger.info(ctx.response.toObject());
return ctx.response;
},
error: function (ctx: Context) {
ctx.config.logger.error(ctx.exception?.message);
},
})
```

泛化调用

如何调用 SDK 尚未支持的 API ?可以使用泛化调用方式。

NOTE 如果没有必须使用的理由，不建议使用泛化方式调用 API，因为无法享受 OpenAPI 提供的兼容性保证。

调用方式

```
const {Client, Request} = require("@ucloud-sdks/ucloud-sdk-js");

async function main() {
// Build client
const client = new Client({
config: {
region: 'cn-bj2',
projectId: process.env.UCLOUD_PROJECT_ID || '',
},
credential: {
publicKey: process.env.UCLOUD_PUBLIC_KEY || '',
privateKey: process.env.UCLOUD_PRIVATE_KEY || '',
}
});
```

```
let resp = null;

try {
resp = await client.invoke(new Request({
Action: "DescribeImage",
}));
} catch (e) {
throw e;
}
const image = resp.toObject()["ImageSet"][0];
}

main().catch(e => { console.error(e) })
```