ALIBABA CLOUD

阿里云

小程序云

基础操作

文档版本：20220209

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.服务空间管理 08
1.1. 创建服务空间 08
1.2. 删除服务空间 08
2.云函数 10
2.1. 概述 10
2.2. 使用云函数 10
2.3. 传递参数 12
2.4. 使用环境参数 13
2.5. 打印日志 14
2.6. 调用小程序Serverless基础服务 14
2.7. 调用HTTP服务 15
2.8. 开启云函数HTTP触发功能 15
2.9. 使用云函数定时触发功能 18
2.10. 小程序Serverless服务关联角色 19
3.云数据库 22
3.1. 数据结构设计 22
3.2. 权限设置 24
3.2.1. 数据权限管理 24
3.2.2. 设置数据表权限 26
3.3. 创建数据表 27
3.4. 添加记录 27
3.5. 删除记录 29
3.6. 更新记录 30
3.7. 查询记录 31
3.8. 使用高级模式 34
3.9. 设置数据库索引 35

3.10. 数据管理最佳实践 36
3.10.1. 云数据库查询优化 36
3.10.2. 管理数据集合 38
3.10.3. 云数据库备份与恢复 42
3.10.4. 云数据库的导入与导出 45
3.10.5. 数据库事务 48
3.10.6. 使用地理位置数据 49
4.云存储 55
4.1. 文件权限管理 55
4.1.1. 文件权限管理 55
4.1.2. 设置文件权限 57
4.2. 上传文件 57
4.3. 删除文件 60
4.4. 文件管理最佳实践 61
5.静态网站托管 64
5.1. 静态网站托管功能介绍 64
5.2. 开通静态网站托管功能 64
5.3. 文件管理 65
5.3.1. 添加文件夹 65
5.3.2. 上传文件 65
5.3.3. 删除文件 67
5.4. 域名管理 67
5.4.1. 添加域名 67
5.4.2. 删除域名 68
5.4.3. 验证域名归属权 68
5.5. 证书管理 71
5.5.1. 绑定证书 71
5.5.2. 解绑证书 72

5.6. 更新网站配置 73
5.7. 静态网站托管最佳实践 73
6.统计分析 79
6.1. 集成统计分析服务 79
6.1.1. 支付宝小程序集成 79
6.1.2. 微信小程序集成 82
6.2. 查看统计数据 85
6.3. 基本统计指标说明 86
6.4. 概况分析 86
6.4.1. 整体趋势 86
6.4.2. 实时统计 89
6.5. 用户分析 90
6.5.1. 新增用户 90
6.5.2. 活跃用户 91
6.5.3. 启动次数 93
6.6. 留存分析 94
6.6.1. 留存用户 94
6.7. 用户参与度 95
6.7.1. 使用时长 95
6.7.2. 使用频率 96
6.8. 渠道 97
6.8.1. 时段详情 97
6.8.2. 渠道列表 97
7.扩展能力 99
7.1. 云调用概述 99
7.2. 使用云调用 100
7.3. 公共数据结构 100
8.设置 102

8.1. 添加支付宝小程序 102
8.2. 添加钉钉小程序 103
8.3. 添加微信小程序 104

1.服务空间管理

1.1. 创建服务空间

在使用小程序Serverless进行开发前，您需要先创建一个服务空间。小程序开发中创建的云资源会和服务空间进行关联。每个服务空间之间相互独立。

背景信息

所有的资源都是通过服务空间来进行分配和管理的。每个服务空间都有一个全局唯一的space ID。在使用云资源时，都通过这个space ID进行关联。

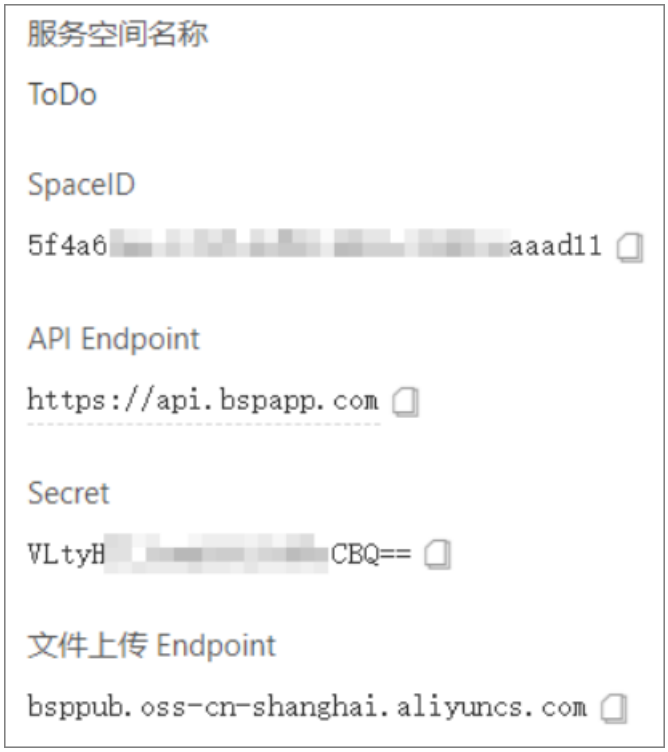
操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 服务空间管理**。
3. 单击**创建服务空间**。
4. 输入服务空间名称和描述，然后单击**确定**。

   服务空间名称可包含3~15个字符，不能以数字和特殊字符开头。

执行结果

服务空间创建成功后，在**服务空间管理**页面，单击**详情**获取space ID、endpoint和secret等信息。



1.2. 删除服务空间

当您不需要一个服务空间时，可以将其删除。服务空间关联的资源（云数据库、云存储等）也会随之删除。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 服务空间管理**。
3. 找到要删除的服务空间，然后单击**删除**。
4. 在弹出的对话框，单击**确定**。

2.云函数

2.1. 概述

云函数（FaaS）是一段运行在云端的、轻量的、无关联的、并且可重用的代码。无需管理服务器，只需编写和上传代码，即可获得对应的数据结果。使用云函数可以使企业和开发者不需要担心服务器或底层运维设施，可以更专注代码和业务本身，也可以使代码进一步解耦，增加其重用性。

在客户端只需依赖 `mpserverless-sdk` ，在应用的上下文中进行简单的鉴权配置，即可调用云函数。

云函数的入参只有一个 `ctx` 对象，该对象可获取调用的云函数时传入的参数，也可获取小程序serverless服务空间的云数据库，云存储等服务。

例如，我们定义一个云函数，名为hello。代码如下所示：

```
//云函数定义在index.js中，此文件为云函数的入口文件
module.exports = ctx => {
    const args = ctx.args;
    return {
        "text": args.hello,
    };
};
```

2.2. 使用云函数

云函数支持使用Node.js进行开发。您可将代码提交到云端运行，在客户端使用小程序Serverless提供的API进行调用。您还可以在云函数中直接通过API调用数据存储和文件存储的服务资源。

> **说明** 您可以通过ctx.mpserverless.function.invoke调用其他云函数，详细说明请参见invoke。

步骤一：创建云函数

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云函数**。
3. 如果您尚未开通函数计算，单击**前往开通**，然后选择**服务协议**并单击**确认开通**。
4. 返回云函数页面，在页面右上角选择一个服务空间。
5. 单击**新建云函数**。
6. 输入函数名称。

   函数名称长度在1-30个字符间，只能包含字母、数字、下划线和中划线，不能以数字、中划线开头。

   > **注意** 云函数的名称必须和要上传的Node.js代码包名称一致。

7. 选择运行环境。目前支持Node.js8。
8. 输入描述。
9. 单击**确定**。

步骤二：定义云函数

1. 在支付宝小程序开发工具中关联对应的后端云服务。
2. 在小程序的*server/functions*目录下创建云函数目录。

   其中*index.js*是云函数 `getImageList` 的入口文件。

```
└── server/
    └── functions
        └── getImageList
            └── index.js
```

3. 编写云函数代码。

   以下代码示例展示了如何从数据库 `images` 里面查出特定用户上传的图片记录。

```
module.exports = async (ctx) => {
  const images = await ctx.mpserverless.db.collection('images').find({ owner: ctx.args.
username });
  return { images };
};
```

> **说明** 云函数的返回值类型仅支持JSON格式。

步骤三：部署云函数

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云函数**。
3. 单击已创建的函数名称链接。
4. 在**发布管理**页签，单击**上传js包**或**更新js包**，然后选择要上传的代码包。

> **说明** 上传的代码包必须满足以下要求：
> - 代码包的名称必须和在控制台上创建的函数名称一致。
> - 代码包必须是.zip文件。
> - 上传的代码包必须包含index.js文件。

5. 上传成功后，单击**代码部署**。

步骤四：调用云函数

云函数代码发布后，在客户端您可以使用 `mpserverless.function.invoke` 方法调用上述定义的云函数。

以下代码示例展示了如何获取用户上传的图片记录，然后更新页面数据。

```
// 引入MPServerless模块
const MPServerless = require('@alicloud/mpserverless-sdk');
// 初始化MPServerless对象
const mpServerless = new MPServerless({
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode,
}, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://api.bspapp.com', // 服务空间地址，从小程序 serverless 控制台处获得
});
// 调用云函数getImageList
mpServerless.function.invoke('getImageList', {
  username: 'Vincent',
}).then((res) => {
  if (res.success && res.result) {
    this.setData({ imageList: res.result.images });
  }
}).catch(console.error);
```

步骤五：查看日志

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云函数**。
3. 单击已创建的函数名称链接。
4. 单击**日志**。
5. 根据执行时间、状态和Request ID等信息过滤您要查看的日志。

2.3. 传递参数

云函数是一个后端服务，有时需要获得从客户端传来的参数。通过云函数内置的 `ctx.args` 方法，您可以获得从客户端传来的参数。

客户端代码

以下是客户端代码示例：

```
await mpserverless.function.invoke('hello', { hello: 'world' });
```

云函数代码

以下是云函数代码示例：

```
module.exports = async ctx => {
  ctx.logger.info('hello %s', ctx.args.hello);
};
```

2.4. 使用环境参数

您可以在云函数内通过ctx.env来获取环境参数，例如SpaceId、调用来源、客户端源IP和客户端UserAgent等信息。

ctx.env是一个object，它包含以下内容：

key值	含义	备注
MP_SPACE_ID	SpaceId	服务空间ID
MP_SOURCE	调用来源	取值说明： • server：服务端触发 • function：云函数触发 • client：客户端触发 • http：HTTP触发 • timing：定时触发
MP_USER_AGENT	客户端User-agent	仅来自客户端的调用包含该字段
MP_CLIENT_IP	客户端IP	仅来自客户端的调用包含该字段
MP_APP_ID	小程序AppId	客户端非匿名授权后调用云函数包含该字段

下面是一个ctx.env的内容示例：

```
{
  "MP_SPACE_ID": "d37f2911-9ac8-4eca-ad75-53f23871b079",
  "MP_SOURCE": "client",
  "MP_USER_AGENT": "Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like Mac OS X) AppleWebKit/605.
1.15 (KHTML, like Gecko) Mobile/15E148 ChannelId(0) LyraVM Nebula  AlipayDefined() AliApp(A
P/10.1.80) AlipayClient/10.1.80 Language/en AlipayIDE",
  "MP_CLIENT_IP": "30.43.101.240",
  "MP_APP_ID": "2019091167276458"
}
```

2.5. 打印日志

为了便于开发调试、问题排查和日常监控，云函数提供日志服务。您可以在云函数中使用 `ctx.logger` 方法打印不同类型的日志信息，然后在云函数控制台中查看执行日志。

目前支持以下类型日志：

- info
- warn
- error
- debug

以 `info` 日志为例，打印客户端传来的参数内容如下：

```
module.exports = async (ctx) => {
  ctx.logger.info('invoke args: %j', ctx.args);
};
```

2.6. 调用小程序Serverless基础服务

在云函数中，您可以直接调用同一环境（空间）的数据存储和文件存储服务以及获取当前请求用户的基本信息。ctx.mpserverless在云函数封装了小程序的基础服务， API使用方式和客户端一致。

数据存储

例如，要从数据存储获取数据，您可以直接调用 `find` 方法：

```
module.exports = async ctx => {
  const args = ctx.args;
  return await ctx.mpserverless.db.collection('user').find({ uid: args.uid });
};
```

文件存储

例如，要从文件存储中删除一个文件，您可以直接调用 `deleteFile` 方法：

```
module.exports = async ctx => {
  const args = ctx.args;
  return await ctx.mpserverless.file.deleteFile(args.filePathUrl);
};
```

用户管理

例如，要获取当前访问用户的基本信息，您可以直接调用 `getInfo` 方法：

```
module.exports = async ctx => {
  return await ctx.mpserverless.user.getInfo();
};
```

相关文档

- find
- deleteFile
- getInfo

2.7. 调用HTTP服务

云函数提供了 ctx.httpclient HTTP 请求方法。无需额外依赖，您就可以请求任何 HTTP 和 HTTPS 协议的 Web 服务。

例如，请求一个网页：

```
module.exports = async ctx => {
  const res = await ctx.httpclient.request('https://www.alipay.com/x/notFound.htm');
  return {
    html: res.status === 200 ? res.data : '',
  };
};
```

其中，ctx.httpclient返回的是一个urllib实例。

2.8. 开启云函数HTTP触发功能

小程序Severless云函数支持由HTTP请求触发云函数调用的功能，本文介绍如何开启云函数HTTP触发功能。

> **注意** 当开启某一函数的 HTTP触发功能后，只要使用 HTTP请求就可以访问到函数，开发者需要关注业务安全和资源安全。
>
> - 业务安全：开发者需在云函数代码中做好权限控制和安全防护，避免未授权访问触发敏感操作。
> - 资源安全：云函数开启了 HTTP 访问后，如果遇到大量恶意访问，消耗云函数资源，开发者可以将云函数的触发路径设置为空或关掉该Space的HTTP触发功能，停止HTTP触发支持。

开启方法

开启云函数HTTP触发功能，您需要开启Space的触发功能且配置云函数的HTTP触发路径。

- 开启Space的HTTP触发功能

  调用接口 UpdateHttpTriggerConfig接口来开启Space的HTTP触发功能，注意需要设置EnableService为*true*。

  系统会为开启HTTP触发功能的Space分配一个默认域名 `(DefaultEndpoint)` ，供HTTP触发使用。

- 开启云函数的HTTP触发功能

  调用接口 UpdateFunction接口来为云函数设置HTTP触发路径 `HttpTriggerPath` ，只有设置了触发路径的云函数才能使用该功能。

- HTTP触发函数执行

  访问 `https://${DefaultEndpoint}${HttpTriggerPath}` ，即可触发对应的云函数执行。

限制说明

1. 请求路径必须以 `/http` 开头，不能以 `/` 结尾，同一个Space下面的云函数的请求路径不能重复，请求路径只支持（/）、（-）、（_）、（.）、字母和数字组合，请求路径最长不超过128个字符。
2. 请求方法仅支持以下四种，其余方法会被忽略。
   - GET
   - POST
   - PUT
   - DELETE
3. 请求时Body的Content-Type仅支持以下三类，允许没有Body的请求，即没有Content-Type。其余Content-Type类型会被拒绝。
   - *application/json*
   - *application/x-www-form-urlencoded*
   - *text/**
4. 通过HTTP触发的响应，都会添加固定的Header，即 `Content-Disposition: attachment` ，此字段会使得返回结果在浏览器中以附件的方式下载。当前此字段无法覆盖，后续支持使用自定义域名之后，可以避免该问题。
5. 请求Body大小限制，不能超过1M。
6. 响应Body大小限制，不能超过1M。

云函数入参说明

在云函数中，通过 `ctx.args` 来获取请求时的内容。各参数的详细解释如下：

```
{
    path: 'HTTP请求路径，如 /hello',
    httpMethod: 'HTTP请求方法，如 GET',
    headers: {HTTP请求头，键值对形式},
    queryStringParameters: {HTTP请求的Query，键值对形式},
    body: 'HTTP请求体',
    isBase64Encoded: 'true or false，表示body是否为Base64编码，当前只会有false'
}
```

云函数返回值说明

云函数返回时，可以返回*string*、*object*、*number*、*boolean*等类型的数据，或者返回自定义响应，随后小程序Serverless会将返回值转化为正常的HTTP响应。

- 返回*string*

  当服务端返回的Content-Type是*text/plain*。

  例如云函数返回：

  ```
  'hello world!'
  ```

  则最终的响应为：

```
HTTP/2 200
date: Mon, 23 Mar 2020 10:12:41 GMT
content-type: text/plain
content-length: 12
content-disposition: attachment
request-id: ac14000d1584958361033101870
hello world!
```

- 返回*object*、*number*或*boolean*

  当服务端返回的Content-Type是*application/json*。

  例如云函数返回：

```
{
    "keyString": "value1",
    "keyNumber": 1234
}
```

  则最终的响应为：

```
HTTP/1.1 200 OK
Date: Mon, 23 Mar 2020 10:18:48 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 39
request-id: ac14000d1584958728277106170
content-disposition: attachment
{"keyString":"value1","keyNumber":1234}
```

- 返回自定义响应

  服务端返回的Content-Type，HttpStatusCode，Headers都由云函数来指定。

  要求云函数返回一个*object*，同时要设定mpserverlessComposedResponse为*true*。

  云函数的返回格式如下：

```
{
    "mpserverlessComposedResponse": true,
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headerName": "headerValue", ... },
    "body": "..."
}
```

  例如云函数返回：

```
{
  mpserverlessComposedResponse: true,
  isBase64Encoded : false,
  statusCode: 200,
  headers: {
    'content-type': 'text/html'
  },
  body: '<h1>Hello World</h1>'
}
```

则最终的响应为：

```
HTTP/2 200
date: Mon, 23 Mar 2020 10:40:09 GMT
content-type: text/html; charset=utf-8
content-length: 20
request-id: ac14000d1584960009970137070
content-disposition: attachment
<h1>Hello World</h1>
```

2.9. 使用云函数定时触发功能

小程序Severless云函数支持定时触发云函数的功能，本文介绍如何使用云函数定时触发功能。

使用方法

调用UpdateFunction接口，通过设置参数TimingTriggerConfig来为云函数设置定时触发配置，若您的云函数执行时间较长，还需要同时设置Timeout来为云函数设置合适的超时时间。

TimingTriggerConfig参数的值为一个cron表达式，例如 `cron:0 0 * * * *` 表示每个整点触发。

一般情况下云函数的超时时间最高支持10秒，云函数执行超过10秒会中断，但是定时触发情况下云函数的最高执行时间上限为600秒。

配置规则说明

配置规则仅支持设置cron表达式。

- cron表达式使用限制：
  - cron表达式必须以 `cron:` 开头，中间不能包含多余的空格。
  - 当前限制云函数定时触发的频率为每小时最多触发一次，要求cron表达式中的秒和分仅支持配置固定的数字，不支持特殊字符。
  - cron表达式不支持配置年。

> **说明** 在cron表达式中不支持同时指定日和星期，两个字段中至少有一个为 * 或者 ？ 。

- cron表达式语法说明：

  cron表达式有六个必选字段，以空格分隔。其中，每个字段都有相应的取值范围。

位置	字段	取值	特殊字符
1	秒	0~59。	不支持。
2	分	0~59。	不支持。
3	时	0~23。	逗号（,）、连接号（-）、星号（*）、正斜线（/）。
4	日	1~31。（根据具体每月天数而定）	逗号（,）、连接号（-）、星号（*）、正斜线（/）。

位置	字段	取值	特殊字符
5	月	可填数字1~12或月份对应英文单词前三个字母的大写形式：{JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC}。	逗号（,）、连接号（-）、星号（*）、正斜线（/）。
6	星期	可填数字0~7或星期对应英文单词的前三个字母的大写形式：{MON|TUE|WED|THU|FRI|SAT|SUN}，其中数字0和7都表示周日。	逗号（,）、连接号（-）、星号（*）、正斜线（/）。

- 特殊字符说明：

特殊字符	含义
逗号（,）	表示列出枚举值。例如：在时字段中 `2,3,4` 表示2点、3点和4点。
连接号（-）	表示指定范围的所有值。例如：在日字段中，`1-5` 包含指定月份的1号到5号。
星号（*）	表示所有值。例如：在时字段中，`*` 表示每个小时。
问号（？）	表示不指定。可以日或者星期字段中使用。
正斜线（/）	表示起始时间开始触发，然后每隔固定时间触发一次。 例如：在分字段中，输入 `1/10` 以指定从第一分钟开始的每隔十分钟触发一次，第11分钟、第21分钟和第31分钟，以此类推。

云函数入参说明

在云函数中，通过 `ctx.args` 来获取执行参数。以下示例展示了定时触发时获取的参数：

```
{
  "timingTriggerConfig": "cron:0 0 * * * *",
  "timestamp": 1585670400006
}
```

其中：

- timingTriggerConfig与用户设置的定时触发配置相同。
- timestamp是触发执行的时间戳（毫秒），实际可能会略晚于cron表达式中设置的时间。

2.10. 小程序Serverless服务关联角色

本文为您介绍小程序Serverless服务关联角色（AliyunServiceRoleForMPServerless）的应用场景以及如何删除服务关联角色。

背景信息

小程序Serverless服务关联角色（AliyunServiceRoleForMPServerless）是在某些情况下，为了完成小程序Serverless自身的某些功能，需要获取其他云服务的访问权限，而提供的RAM角色。更多关于服务关联角色的信息请参见服务关联角色。

应用场景

小程序Serverless的云函数功能需要访问函数计算的资源，通过服务关联角色功能获取访问权限。

AliyunServiceRoleForMPServerless介绍

角色名称：AliyunServiceRoleForMPServerless

角色权限策略：AliyunServiceRolePolicyForMPServerless

权限说明：

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": [
                "fc:CreateService",
                "fc:ListServices",
                "fc:GetService",
                "fc:UpdateService",
                "fc:DeleteService",
                "fc:CreateFunction",
                "fc:ListFunctions",
                "fc:GetFunction",
                "fc:UpdateFunction",
                "fc:DeleteFunction",
                "fc:GetFunctionCode",
                "fc:InvokeFunction",
                "fc:CreateTrigger",
                "fc:ListTriggers",
                "fc:GetTrigger",
                "fc:UpdateTrigger",
                "fc:DeleteTrigger",
                "fc:PublishVersion",
                "fc:ListVersions",
                "fc:DeleteVersion",
                "fc:CreateAlias",
                "fc:ListAliases",
                "fc:GetAlias",
                "fc:UpdateAlias",
                "fc:DeleteAlias",
                "fc:PutProvisionConfig",
                "fc:GetProvisionConfig",
                "fc:ListProvisionConfigs"
            ],
            "Resource": "*",
```

```
            "Effect": "Allow"
        },
        {
            "Action": "ram:PassRole",
            "Resource": "acs:ram:*:*:role/aliyunfcaccessingmpserverlessrole-*",
            "Effect": "Allow"
        },
        {
            "Action": "ram:DeleteServiceLinkedRole",
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ram:ServiceName": "mpserverless.aliyuncs.com"
                }
            }
        }
    ]
}
```

删除服务关联角色

如果您需要删除AliyunServiceRoleForMPServerless（服务关联角色），需要先删除所有的服务空间。

- 删除服务空间具体操作请参见删除服务空间。
- 删除服务关联角色具体操作请参见删除服务关联角色。

3.云数据库

3.1. 数据结构设计

数据存储服务是基于MongoDB托管在云端的数据库，数据以JSON格式存储。作为开发者，您可以在客户端内直接操作数据，也可以在云函数中读写数据。

数据库结构

小程序Serverless服务使用的是MongoDB，以JSON格式存储数据。数据库中的每条记录都是一个JSON格式的对象，一个数据库可以包含多个集合（相当于关系型数据库中的表），每个集合可看做一个JSON数组。

小程序Serverless MongoDB数据库和关系型数据库MySQL的对比，如下表所示。

分布式文件存储数据库 (MongoDB)	关系型数据库 (MySQL)
数据库（database）	数据库（database）
表（collection）	表（table）
记录（document）	行（row）
域（field）	列（column）
索引（index）	索引（index）
自动将 `_id` 字段作为主键	主键（primary key）

数据结构设计策略

MongoDB是一个基于分布式文件存储的NoSQL数据库，旨在为WEB应用提供可扩展的高性能数据存储解决方案。由于数据的灵活性以及通用性，在设计数据结构时也需要根据数据之间的关系来决定如何存储。

以电商平台为例。通常电商平台的核心数据包含产品、用户、购物车和订单。订单是由用户购买产品产生的，订单就可以认为是产品和用户之间的关联关系产生的。在订单产生之前，产品和用户之间的关联关系是通过购物车来维持的。

针对上述案例，可以设计一个名称为 `products` 的产品表存储以下信息：

- 产品基础信息：包含展示信息、商品规格等。
- 属性信息：属性都归属于一个产品，属于N对1的关系。因此作为子文档存储在 `products` 集合中较合适。
- 库存及价格信息：库存不仅仅跟产品关联，更直接对应产品属性。因此也应该放在 `products` 集合中。

最终的数据库设计如下：

```
{
    "id": 5573,
    "name": "Egg T恤",
    "desc": {
        "short": "限量版 Egg T恤，穿上它你离极客也不远了",
        "long": "这是一段非常长的描述",
        "category": {
            "_id": "48bf43a..29e90bc",
```

```
            "name": "上衣"
        }
    },
    "attributes": [
        {
            "id": 1151,
            "name": "尺寸",
            "values": [
                {
                    "id": 3871,
                    "value": "S"
                },
                {
                    "id": 3874,
                    "value": "M"
                },
                {
                    "id": 3875,
                    "value": "L"
                }
            ]
        },
        {
            "id": 1152,
            "name": "性别",
            "values": [
                {
                    "id": 3872,
                    "value": "男"
                },
                {
                    "id": 3873,
                    "value": "女"
                }
            ]
        }
    ],
    "sku": [
        {
            "id": 1153,
            "stock": 30,
            "attributeIds": [
                3871,
                3872
            ],
            "attributes": [
                {
                    "key": "尺寸",
                    "value": "S"
                },
                {
                    "key": "性别",
                    "value": "女"
                }
```

```
                ]
            }
        ]
    }
```

3.2. 权限设置

3.2.1. 数据权限管理

小程序云Serverless提供了一套简明易懂的安全规则方案用来控制小程序端侧用户对资源的访问，保障数据安全，概念上近似于Linux的文件权限体系。对于云数据库，您可以在控制台上通过配置权限规则来管理数据集合的读写权限。

> **说明** 从控制台、云函数或者管控SDK发起的数据库操作，不受此安全规则限制。

规则语法

目前数据库的访问权限是在数据库集合粒度上配置的，通过一个JSON规则来表达集合对不同操作的访问限制，例如。

```
{
  ".read": true,
  ".write": "request.auth.userId==resource.auth.userId"
}
```

用JSON来表达权限规则的语法约定为：

- 文本内容必须是合法的JSON。
- key代表操作类型：
  - `.read` ：读操作。
  - `.write` ：写操作。
  - `*` ：所有操作。
- 操作类型对应的取值支持：
  - `true` ：可操作。
  - `false` ：不可操作。
  - `request.auth.userId==resource.auth.userId` ：即校验用户身份，只有数据归属人能操作。

匹配的优先级和限制：

- 目前明确的操作类型只支持 `.read` 和 `.write` 两种。
- 对于操作类型，需要明确指定规则的优先，否则再进行通配符匹配。
- 若未配置通配符，则 `.read` 和 `.write` 都要明确配置权限规则。

预设权限说明

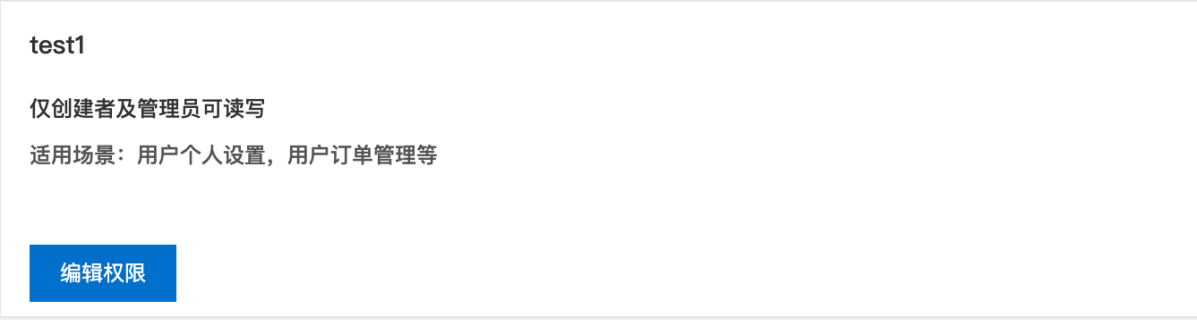
下表列举了小程序云预设的4种常用数据权限，和通过JSON自定义权限的示例。

规则名称	对应规则详情	说明
开放所有权限（需要通过自定义规则模式编写，建议仅在调试时使用）	`{ ".read": true, ".write": true }`	所有用户都对该数据表拥有全部读写权限。 **说明** 该权限规则会放开数据库操作权限，请谨慎设置。
所有用户可读，仅创建者及管理员可写	`{ ".read": true, ".write": "request.auth.userId==resource.auth.userId" }`	公开读，创建者可读写。 适用场景：如用户评论、公开信息等。
仅创建者及管理员可读写	`{ ".read": "request.auth.userId==resource.auth.userId", ".write": "request.auth.userId==resource.auth.userId" }`	仅创建者可读写。 适用场景：商品信息等。
所有用户可读，仅管理员可写	`{ ".read": true, ".write": false }`	所有用户只读。 适用场景：商品信息等。
仅管理员可读写	`{ ".read": false, ".write": false }`	所有用户不可读写。 适用场景：仅从业务后端写入后台数据流水、日志等。

关于如何修改数据库权限，请参见设置数据表权限。

默认规则

对于新创建的数据库，小程序Serverless会自动提供一个默认规则。默认所有人有读权限，仅管理员有写权限，接入测试前，需要先调整权限规则，请参见设置数据表权限。



该权限对应JSON规则表达式为：

```
{
  ".read": true,
  ".write": false
}
```

仅创建者可访问

如果期望某个集合中的数据，只有该数据的创建者可访问，那么需要指明权限规则为校验用户身份：`request.auth.userId==resource.auth.userId`。对于预设权限，以下两种权限会使用此规则：

- 所有用户可读，仅创建者及管理员可写。
- 仅创建者及管理员可读写。

> **注意**
>
> - 当 `.write` 权限（包括通配符匹配）为**仅创建者可写**时，创建者发起的写操作都会自动插入字段auth来存储创建者的userId，无需手动处理，且该字段不可篡改。
> - 当 `.read` 权限（包括通配符匹配）为**仅创建者可读**时，访问者只能获取到数据携带了auth字段，且auth.userId 字段和自己userId一致的数据。
> - 若用户之前写入的数据中未包含auth字段，而在某个时刻才把集合的权限调整为**仅创建者可读**后，用户可能将无法拉取之前上传的数据。

3.2.2. 设置数据表权限

小程序Serverless会为每个新建的数据表自动提供一个默认权限规则。默认所有人有读权限，仅管理员有写权限，您可以通过修改权限规则管理数据表的权限。

前提条件

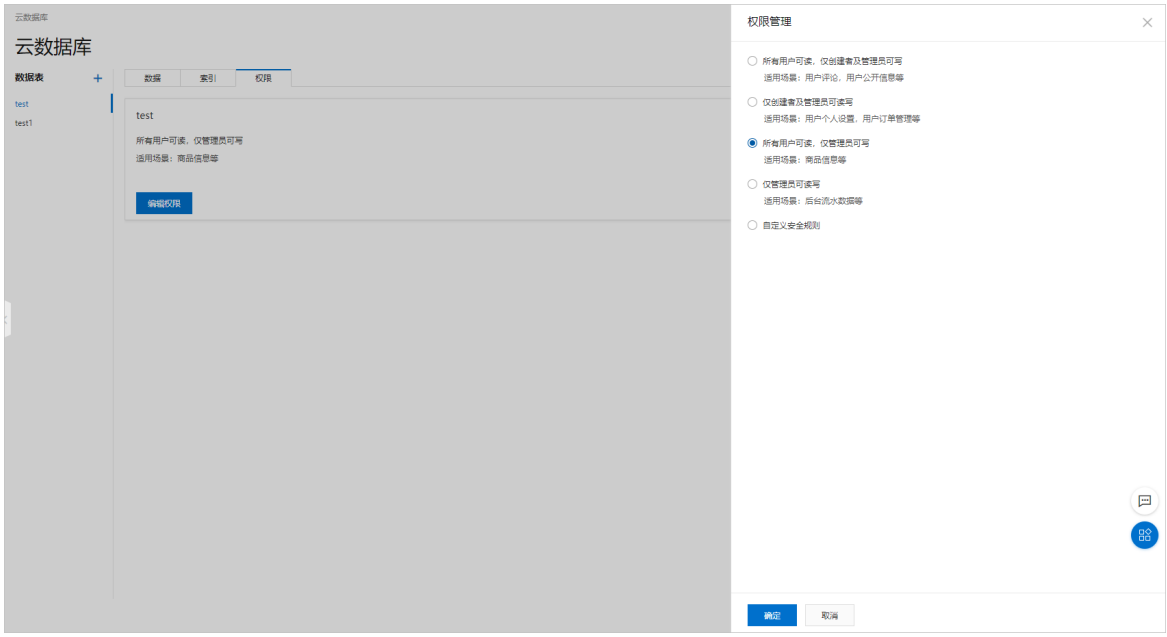
确保您已经了解小程序云Serverless服务的数据库权限设置规则。

小程序Serverless在集合粒度上提供了5种模式来配置权限策略，其中包含4种不同的预设模式，并提供自定义模式通过编辑JSON的方式自由控制数据权限。更多信息，请参见数据权限管理。

操作步骤

1. 在云数据库页面，单击您要设置的数据表。
2. 在**数据表设置**页面，单击**权限 > 编辑权限**，然后选择权限策略。

   关于预设权限模式和自定义JSON权限规则语法要求，请参见数据权限管理。
3. 修改完成后，单击**确定**。



3.3. 创建数据表

小程序Serverless服务使用的是分布式文件存储数据库MongoDB，以JSON格式存储数据。数据库中的每条记录都是一个JSON格式的对象，一个数据库可以包含多个集合（相当于关系型数据库中的表）。您可以在控制台上创建数据表。

前提条件

创建服务空间

控制台操作

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云数据库**。
3. 在页面右上角，选择要关联的环境。
4. 单击添加图标 +，输入数据表名称，最后单击**确定**。

相关文档

- 数据结构设计

3.4. 添加记录

数据库中的每条记录都是一个JSON格式的对象，您可以直接在控制台上添加一条记录。

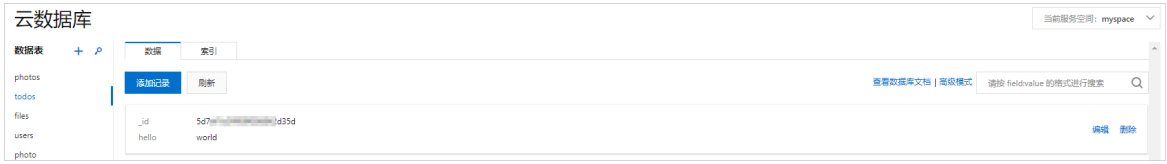
前提条件

创建数据表

控制台操作

1. 在云数据库页面，单击目标数据表，然后单击**添加记录**。
2. 在**添加记录**页面，以JSON格式输入记录内容，然后单击**确定**。

   每条记录都有一个唯一的ID。添加成功后您可以删除或修改该条记录。



客户端调用

- 添加一条记录。

```
mpserverless.db.collection('users').insertOne({
    name: 'tom',
    age: 1
})
.then(res => {})
.catch(console.error);
```

- 添加多条记录。

```
mpserverless.db.collection('users').insertMany([{
    name: 'tom',
    age: 1
},{
    name: 'jerry',
    age: 2
}])
.then(res => {})
.catch(console.error);
```

云函数调用

- 添加一条记录。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users').insertOne({
     name: 'tom',
      age: 1
  });
};
```

- 添加多条记录。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users').insertMany([{
    name: 'tom',
    age: 1
  },
  {
    name: 'jerry',
    age: 2
  }]);
};
```

相关文档

- insertOne
- insertMany

3.5. 删除记录

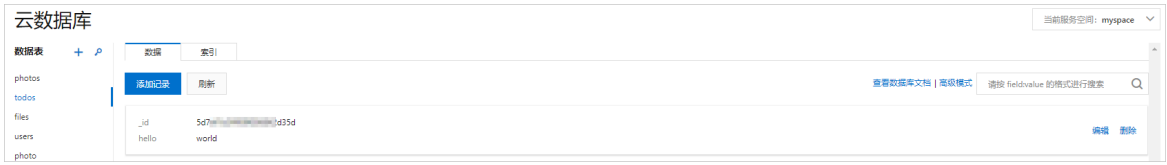
您可以通过阿里云控制台或调用API删除数据库记录。

前提条件

创建数据表

控制台操作

1. 在云数据库页面，单击目标数据表。
2. 在**数据**页签，单击目标记录的**删除**选项，然后单击**确定**。



云函数调用

- 删除一条记录。

```
mpserverless.db.collection('users')
  .deleteOne({
    name: 'Tom'
  })
  .then((res) => {
    const hasDeleted = res.affectedDocs > 0;
  })
  .catch(console.error);
```

- 删除多条记录。

```
mpserverless.db.collection('users')
  .deleteMany({
     age: {$lt: 18}
  })
  .then((res) => {
    const hasDeleted = res.affectedDocs > 0;
  })
  .catch(console.error);
```

相关文档

- deleteOne
- deleteMany

3.6. 更新记录

本文介绍如何在小程序端和云函数中更新一条或多条记录。

控制台操作

1. 在云数据库页面，单击目标数据表。
2. 在**数据**页签，单击目标记录的**编辑**修改记录。

客户端调用

- 更新集合中的一条记录。

```
mpserverless.db.collection('users').updateOne({
    name: 'jerry'
}, {
    $set: {
        age: 10
    }
})
.then(res => {})
.catch(console.error);
```

- 更新集合中的多条记录。

```
mpserverless.db.collection('users').updateMany({
    name: 'jerry'
}, {
    $set: {
        age: 10
    }
})
.then(res => {})
.catch(console.error);
```

云函数调用

- 更新集合中的一条记录。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .updateOne({
      name: 'jerry'
    }, {
      $set: {
        age: 10
      }
    });
};
```

- 更新集合中的多条记录。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .updateMany({
      name: 'jerry'
    }, {
      $set: {
        age: 10
      }
    });
};
```

3.7. 查询记录

本文介绍如何使用数据存储服务查询数据集中的记录。

客户端调用

- 查询一条记录。

  调用findOne方法查询大于且最接近18岁的一条记录的name。

  `projection: { name: 1 }` 表示只返回name字段，其中1表示过滤，0表示排除。

  `sort: { age: 1 }` 表示将查询后的结果按升序排列，其中1表示升序，-1表示降序。

```
mpserverless.db.collection('users')
  .findOne({
    age: { $gt: 18 }
  }, {
    projection: { name: 1 },
    sort: { age: 1 }
  })
  .then(res => { })
  .catch(console.error);
```

- 查询所有记录。

  调用find方法查询所有大于18岁的name，并将查询结果按升序返回。

```
mpserverless.db.collection('users')
  .find({
    age: { $gt: 18 }
  }, {
    projection: { name: 1 },
    sort: { age: 1 }
  })
  .then(res => { })
  .catch(console.error);
```

- 查询并删除一条记录。

  调用findOneAndDelete方法查询并删除小于且最接近18岁的一条数据。

```
mpserverless.db.collection('users')
  .findOneAndDelete({
    age: { $lt: 18 }
  }, {
    sort: { age: -1 }
  })
  .then(res => { })
  .catch(console.error);
```

- 查询并整体替换一条记录。

  调用findOneAndReplace方法查询并替换name为zhangsan的一条数据。

```
mpserverless.db.collection('users')
  .findOneAndReplace({
    name: "zhangsan"
  }, {
    $set: {
      name: "lisi",
      age: 20
    }
  })
  .then(res => { })
  .catch(console.error);
```

- 查询并更新记录。

  调用findOneAndUpdate查询并更新name为zhangsan的一条记录的age字段。

```
mpserverless.db.collection('users')
    .findOneAndUpdate({
        name: "zhangsan"
    },
    {
    $set: {
        age: 18
        }
    })
    .then(res => {})
    .catch(console.error);
```

云函数调用

- 查询一条记录。

  调用findOne方法查询大于且最接近18岁的一条记录的name。

  `projection: { name: 1 }` 表示只返回name字段，其中1表示过滤，0表示排除。

  `sort: { age: 1 }` 表示将查询后的结果按升序排列，其中1表示升序，-1表示降序。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .findOne({
      age: { $gt: 18 }
    }, {
      projection: { name: 1 },
      sort: { age: 1 }
    });
};
```

- 查询所有记录。

  调用find方法查询所有大于18岁的name，并将查询结果按升序返回。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .find({
      age: { $gt: 18 }
    }, {
      projection: { name: 1 },
      sort: { age: 1 }
    });
};
```

- 查询并删除一条记录。

  调用findOneAndDelete方法查询并删除小于且最接近18岁的一条数据。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .findOneAndDelete({
      age: { $lt: 18 }
    }, {
      sort: { age: -1 }
    });
};
```

- 查询并整体替换一条记录。

  调用findOneAndReplace方法查询并替换name为zhangsan的一条数据。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .findOneAndReplace({
      name: "zhangsan"
    },{
      $set: {
        name: "lisi",
        age: 20
      }
    });
};
```

- 查询并更新记录。

  调用findOneAndUpdate查询并更新name为zhangsan的一条记录的age字段。

```
'use strict';
module.exports = async function (ctx) {
  return await ctx.mpserverless.db.collection('users')
    .findOneAndUpdate({
      username: "zhangsan"
    },
      {
        $set: {
          age: 18
        }
      });
};
```

3.8. 使用高级模式

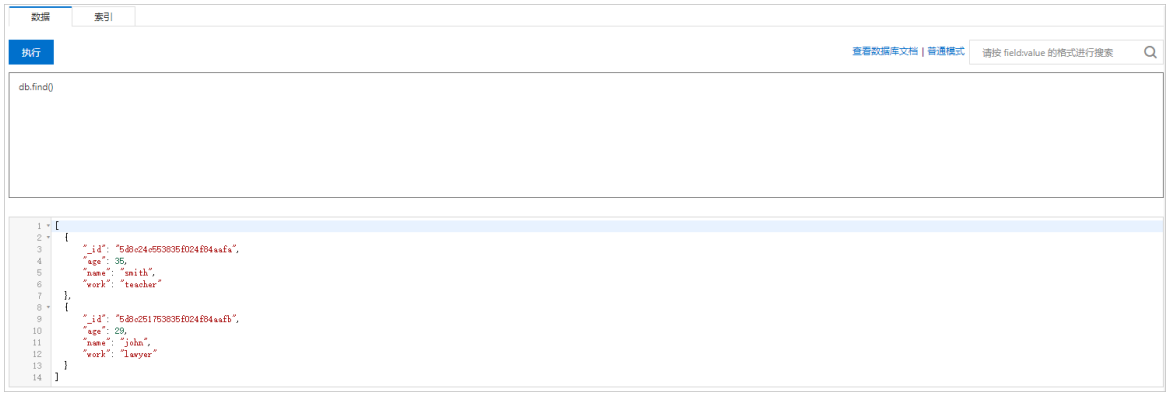
在控制台的云数据库管理页面中，可以切换到高级模式直接编写数据库语句的执行脚本，对数据库进行增删查改操作。

前提条件

创建数据集合

操作步骤

1. 在数据库页面，单击目标数据表，然后选择页面右上角的**高级模式**。
2. 在输入框中输入执行脚本。
3. 单击**执行**按钮，输出执行结果。



> **说明** 高级模式支持的操作同客户端，请参见数据库API。

3.9. 设置数据库索引

使用数据库时，对成为查询条件的字段设置索引可以有效提高查询效率，更快的获取信息。缺省情况下会对_id_字段默认创建一个非唯一索引。

前提条件

- 创建数据集合
- 添加记录

操作步骤

1. 在数据库页面，单击目标数据表，然后单击**索引**页签进入索引设置页面。
2. 单击**添加索引**进入索引设置页面。
3. 填写索引名称，设置索引属性、索引字段。
   - 索引属性：可以设置为唯一索引或非唯一索引。

     将字段设置为唯一索引后，可以防止不同记录的被索引键上存储相同值。

     > **注意** 如果一个记录在一个唯一索引中没有索引字段的值，则该索引将为该记录存储一个空值。由于唯一约束限制，只会允许一个记录缺少索引字段。对多于一个以上的记录没有索引字段的值或缺少索引字段，索引构建将失败，提示重复键错误。

   - 索引字段：支持单键索引和复合索引设置，字段按升序或降序排列。

     单键索引是最常见的索引形式，针对一个指定字段建立索引。对于单字段索引，升序、降序的查询效果一样。

     复合索引是单键索引的升级版，针对多个字段联合创建索引，先按照第一个字段排列，第一个字段相同的记录按第二个字段排列，依次类推。
4. 单击**保存**按钮，保存索引设置。

3.10. 数据管理最佳实践

3.10.1. 云数据库查询优化

小程序Serverless云数据库使用的是Mongo。在数据量比较大的情况下，有时候查询操作会报错 `operation exceeded time limit` 。本文介绍如何进行查询优化以避免此类问题。

设置合适的索引

如果您的查询操作包含了过滤条件（包含等值测试和范围过滤）或者是排序功能，则要考虑给集合添加索引。

> **说明**
>
> 在创建索引时，可以去掉那些没有选择性的等值测试字段或范围过滤字段，以减少索引空间的占用。

针对复杂的查询语句，包含了等值测试、范围过滤以及排序功能，这里提供一个建立索引的方法。

- 索引中字段的顺序需要按照等值测试字段、排序字段、范围过滤字段的方式排列。
- 若等值测试包含多个字段，它们之间的顺序可以任意互换，索引中的升序降序也不影响。
- 若排序字段包含多个字段，添加到索引中顺序需要按照排序中的顺序，升序降序也要一致。
- 若范围过滤包含多个字段，则优先将基数值（集合中字段不同值的数量）少的字段放到前面。

> **说明**
>
> 如果您的集合上有多条索引，尤其是您的查询语句较复杂时，Mongo不一定能为您选择正确的索引，建议您在查询语句中指定此次查询过程中使用的索引。

如下所示，您可以创建一个复合索引，依次包含 `course` ， `sex` ， `class` ， `name` ， `birthmonth` ， `score` 。 `course` 和 `sex` 属于等值测试字段，它们需要放在最前面，顺序也可以互换。 `birthmonth` 和 `score` 属于范围过滤字段，需要放到最后，由于 `birthmonth` 取值范围是1-12，score的取值范围是0-100，所以需要把 `birthmonth` 放到 `score` 前面。 `class` 和 `name` 属于排序字段，应该放在中间，顺序和升降序都和排序时一致。

```
//在分数表中，查询数学成绩大于80分并且下半年出生的男生的记录，并且按照班级和姓名排序。
mpserverless.db.collection('score').find(
  {
    course: "Math",
    sex: "male",
    birthmonth: { $gt: 6 },
    score: { $gt: 80 },
  },
  {
    sort: { class: 1, name: 1 },
  },
);
```

大量数据查询优化

如果您的数据量非常大，在设置合适的索引之后仍然会查询超时，您要考虑以下优化方案。

1. 尽量避免使用skip，至少不应该skip比较大的值，因为skip操作Mongo服务端依然会扫描被skip的数据，带skip操作的耗时和skip的数量线性相关。您可以考虑使用排序和范围查询功能来替代直接使用skip。
2. 对于非常大的数据可以分段来查询，即通过一定的条件将一次查询拆分为多次查询操作。

分段计算count

您可以通过findOne+RangeQuery+Skip+Sort的方式来分段计算Count，注意使用该方法需要添加合适的索引。

如下面的代码示例所示，以_id作为分段列，我们先调用findOne接口，依次查询第100001，200001，300001....个_id值，当剩余数据条数不足100001条时，findOne返回的result为空，这时可以通过count获取剩余的记录数。注意在调用findOne时需要指定查询条件_id大于等于特定值，需要指定排序规则为按_id升序排列，需要指定查询跳过100000条记录；在调用count查询时需要指定_id值大于最后一次findOne返回的_id。

```
module.exports = async (ctx) => {
    const skip = 100000;
    let count = 0;
    let minId = { $minKey: 1 };
    const collection = ctx.mpserverless.db.collection('collectionName');
    while (true) {
        const query = { _id: { $gte: minId } };
        const options = {
            skip,
            projection: { _id: 1 },
            sort: { _id: 1 },
        };
        const findOneResult = await collection.findOne(query, options);
        const { affectedDocs, success } = findOneResult;
        if (!success) {
            throw new Error("findOne return success false");
        }
        if (affectedDocs > 0) {
            const newId = findOneResult.result._id;
            minId = newId;
            count += skip;
        } else {
            break;
        }
    }
    const query = { _id: { $gte: minId } };
    const countResult = await collection.count(query);
    const { affectedDocs, success } = countResult;
    if (!success) {
        throw new Error("count return success false");
    }
    count += affectedDocs;
    return count;
};
```

遍历整个集合

您应该避免使用find+Skip+Limit的查询方式来遍历整个集合，因为这种方式随着Skip数量的增长响应时间会越来越慢，还可能会造成请求超时。您可以改为使用find+RangeQuery+Limit+Sort的方式。

下面的代码给出了一个示例。将数据按_id排列，每次查询时都指定查询条件大于上次查询结果中的最后一条记录的_id，依次遍历到最后。

```
module.exports = async (ctx) => {
    const pageSize = 100;
    let minId = { $minKey: 1 };
    const collection = ctx.mpserverless.db.collection('collectionName');
    while (true) {
        const query = { _id: { $gt: minId } };
        const options = {
            limit: pageSize,
            sort: { _id: 1 },
        };
        const findResult = await collection.find(query, options);
        const { affectedDocs, success } = findResult;
        if (!success) {
            throw new Error("find return success false");
        }
        if (affectedDocs > 0) {
            const data = findResult.data;
            const lastDoc = data[data.length -1];
            minId = lastDoc._id;
            //todo 在这里添加处理data列表的逻辑。
        } else {
            break;
        }
    }
    return 'success';
};
```

分页查询优化

您应该使用上一页、下一页、上n页（其中n是一个比较小的数字）、下n页的翻页功能来替换随机翻页。您可以参考百度或者谷歌的搜索结果的分页功能，当结果页数非常多时，不展示共有多少页，仅支持在前10页中支持随机翻页；再往下翻页的过程中，不再支持随机翻页，仅支持向下翻一个较小的页数，这样就可以在已经查询出结果的基础上再使用find+RangeQuery+Skip（少量）+Limit+Sort的方式来快速查询到结果。例如当您已经查询出第13页的数据后，再次查询第17页的数据时，就可以把第13页的最后一条数据作为RangeQuery的判断条件，跳过3页的数据行数，再查询一页的数据就是第17页的数据。

3.10.2. 管理数据集合

您可以通过小程序Serverless提供的数据存储服务API对集合（相当于MySQL中的表）进行数据读取、写入、更新、删除等操作。

读取数据

以下代码示例展示了适用于不同场景的数据读取方法：

- 从集合中读取所有数据。

例如，查询所有图片：

```
mpserverless.db.collection('images').find();
```

- 根据查询条件从集合中读取指定数据。

  例如，查询特定用户添加的图片：

```
mpserverless.db.collection('images').find({ 'userId': userId });
```

- 读取数据并对获取的数据进行排序。

  例如，将获取的图片按照添加时间进行排序：

```
mpserverless.db.collection('images').find({ 'userId': userId }, {sort: {'uploadTime': -1
}});
```

- 读取数据并对获取的数据进行操作。

  例如，读取用户的图片列表，并显示出来。其中，`_getImages(user)` 是封装好的方法，用来获取用户对应的图片列表，并更新小程序中的数据，此时页面内容也会刷新。在 `onShow()` 方法中，当用户信息获取成功，便去获取用户的图片列表并显示到页面上。

```
// client/index/index.js
// 显示指定当前用户的图片列表
onShow() {
  mpserverless.user.getInfo().then((user) => {
    this._getImages(user);
  }).catch(console.log);
},
// 获取特定用户的图片列表
_getImages(user) {
  mpserverless.db.collection('images')
  .find({ userId: user.userId }, {sort: {uploadTime: -1 }})
  .then(({ result: images }) => {
    images.map((item) => {
      item.uploadTime = new Date(item.uploadTime).toDateString();
      return item;
    });
    this.setData({ images });
  })
  .catch(console.log);
},
```

写入数据

您可以通过以下两种方式写入数据：

- 通过 `insertOne()` 方法逐条向集合写入数据。

  例如，使用 `insertOne()` 方法为用户添加一个文档：

```
mpserverless.db.collection('images').insertOne({
  text: inputText,
  url: imageUrl,
});
```

- 通过 `insertMany()` 方法批量写入数据。

  例如，使用 `insertMany()` 方法为用户添加一个文档：

```
// client/add-image/add-image.js
// 将新的任务添加到当前用户的图片列表中
add() {
  mpserverless.user.getInfo().then((user) => {
    mpserverless.db.collection('images').insertOne({
      text: this.data.inputValue,
      url: this.data.imageUrl ? this.data.imageUrl : false,
      userId: user.userId,
      uploadTime: new Date(),
    }).then(() => {
      my.navigateBack();
    }).catch(console.log);
  }).catch(console.log);
},
```

更新数据

您可以通过以下两种方式更新数据：

- 通过 `updateOne()` 方法更新符合条件的第一条数据。

  例如，使用 `updateOne()` 方法更新一个文档：

```
mpserverless.db.collection('images').updateOne(
  { _id: imageId },
  {
    $set: {
      text: result.inputValue,
    }
  }
);
```

  其中， `$set` 是更新记录的处理方式。

- 通过 `updateMany()` 方法更新符合条件的所有数据。

  以一个图片管理的小程序为例，用户触发重命名图片名称操作时，显示对话框让用户输入新的图片名称，执行更新文档中的名称字段：

```
// client/index/index.js
// 变更图片名称的事件处理
rename(e) {
  const dataset = e.target.dataset;
  my.prompt({
    title: '修改名称',
    message: '请输入新的图片名称：',
    placeholder: '',
    okButtonText: '确定',
    cancelButtonText: '取消',
    success: (result) => {
      if (result.ok) {
        mpserverless.db.collection('images').updateOne(
          { _id: dataset.itemId },
          {
            $set: {
              text: result.inputValue,
            }
          }
        ).then(() => {
          this._getImages(this.data.user);
        }).catch(console.log);
      }
    },
  });
},
```

删除数据

您可以通过以下两种方式删除数据：

- 通过 `deleteOne()` 方法删除第一个符合条件的数据。

  例如，使用 `deleteOne()` 方法删除一个文档：

  ```
  mpserverless.db.collection('image').deleteOne({ _id: imageId });
  ```

- 通过 `deleteMany()` 方法删除所有符合条件的数据。

  以一个图片管理的小程序为例，当用户单击示例小程序图片删除按钮时，唤起删除对话框，确认后执行删除和刷新图片列表：

```
// client/index/index.js
// 删除图片的事件处理
delete(e) {
  const dataset = e.target.dataset;
  // 确认和删除图片
  my.confirm({
    title: '删除图片',
    content: '是否确认删除该图片?',
    confirmButtonText: '删除',
    cancelButtonText: '取消',
    success: (result) => {
      if (result.confirm) {
        mpserverless.db.collection('images').deleteOne({
          '_id': dataset.itemId,
          'userId': this.data.user.userId,
        }).then(() => {
          // 刷新任务列表
          this._getImages(this.data.user);
        }).catch(console.log);
      }
    },
  });
},
```

3.10.3. 云数据库备份与恢复

本文介绍小程序Serverless云数据库的回档和重命名操作，以及如何进行备份与恢复。

备份

小程序Serverless的云数据库会自动在每天的凌晨开始进行备份，并保留最近7天的备份记录。小程序Serverless云数据库暂不支持手动备份。

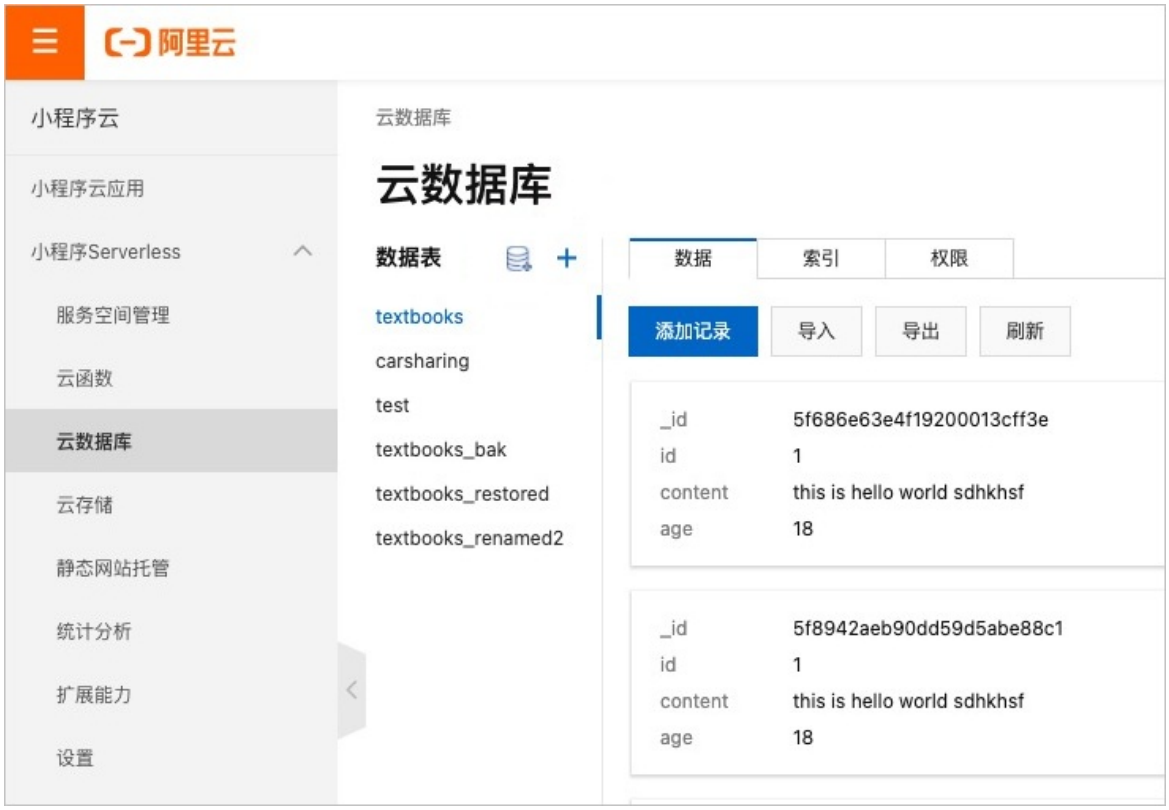
恢复

本节以test集合为例介绍如何将数据库恢复到指定时间点。

1. 将原集合test回档为新集合test_bak，具体操作，请参见回档。
2. 删除原集合test。
3. 检查并修改新集合test_bak的权限策略。
4. 将新集合test_bak重命名为test，具体操作，请参见集合重命名。
5. 再次确认新集合test中的权限策略是否正确。

回档

1. 在云数据库管理页面单击图标。



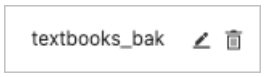
2. 在右侧弹出的新建回档页面下拉选择**回档的时间**，选中要回档的集合（支持多选），然后在集合右侧确认回档后的**集合名称**，最后单击页面下方的**确定**。

> **注意**
>
> 由于备份需要时间，这里显示的**可选回档时间**是定时备份任务开始的时间，而不是云数据库这个时间点的快照，存在时间差。

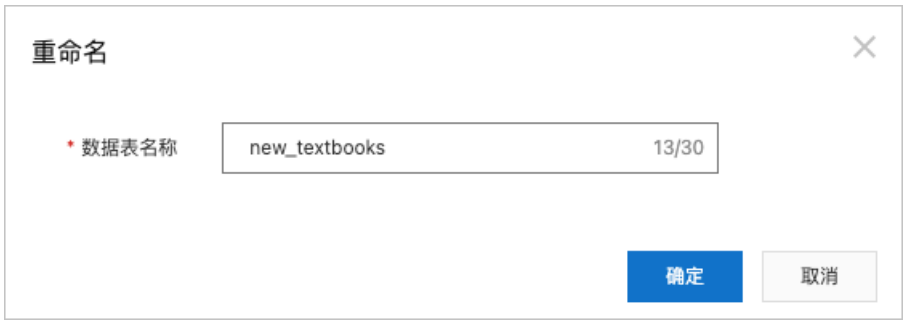
回档执行成功后会提示**执行回档任务成功**。

集合重命名

1. 将鼠标悬浮在集合名称上，然后单击图标。



2. 填入重命名后的集合表名称，然后单击**确定**。



说明

- 新集合名称不能和已有集合重名。
- 重命名前后集合的权限策略不变，建议您在重命名集合前后都要检查集合的权限策略是否符合预期。

3.10.4. 云数据库的导入与导出

本文介绍如何对云数据库进行导入与导出操作。

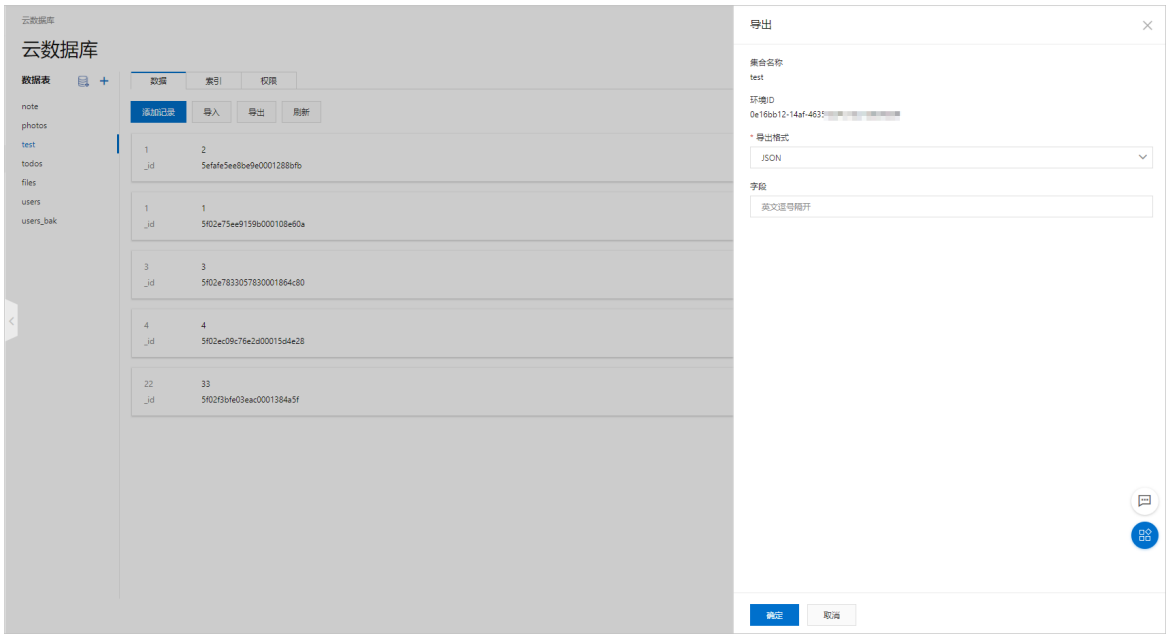
数据导出

您可以将数据库中的内容导出为JSON或者CSV格式的文件，导出结果中的每行数据都对应数据库中的一条记录。

导出格式	是否必须指定导出字段	默认导出内容	建议使用场景
JSON格式	不必须	默认导出全部内容，指定字段时仍会导出_id字段	备份数据
CSV格式	必须	仅导出指定字段	分析数据

控制台操作步骤：

1. 在云数据库管理页面，单击您的目标数据库。
2. 在**数据**页签单击**导出**。
3. 在导出页面，下拉选择**导出格式**，然后填入要**导出的字段**，最后单击**确定**。



注意

- 云数据库导出时会丢失部分类型信息，不能保证导出后再导入后的内容和原有的数据完全一致。
- 若您的集合数据量比较大，导出需要较长的时间，建议您耐心等待，导出完成后浏览器会自动下载导出后的文件。

数据导入

您可以将JSON或者CSV格式的文件导入进数据库，导入文件中的每行数据都对应数据库中的一条记录。

JSON文件格式要求如下。

- 每行都是一个完整的JSON，并且类型是一个JSON对象，不能是JSON数组或者JSON的基本类型。
- JSON 数据每个键值对的键名首尾不能是英文句号（.），并且不能包含多个连续的英文句号（.）。
- 键名不能重复。

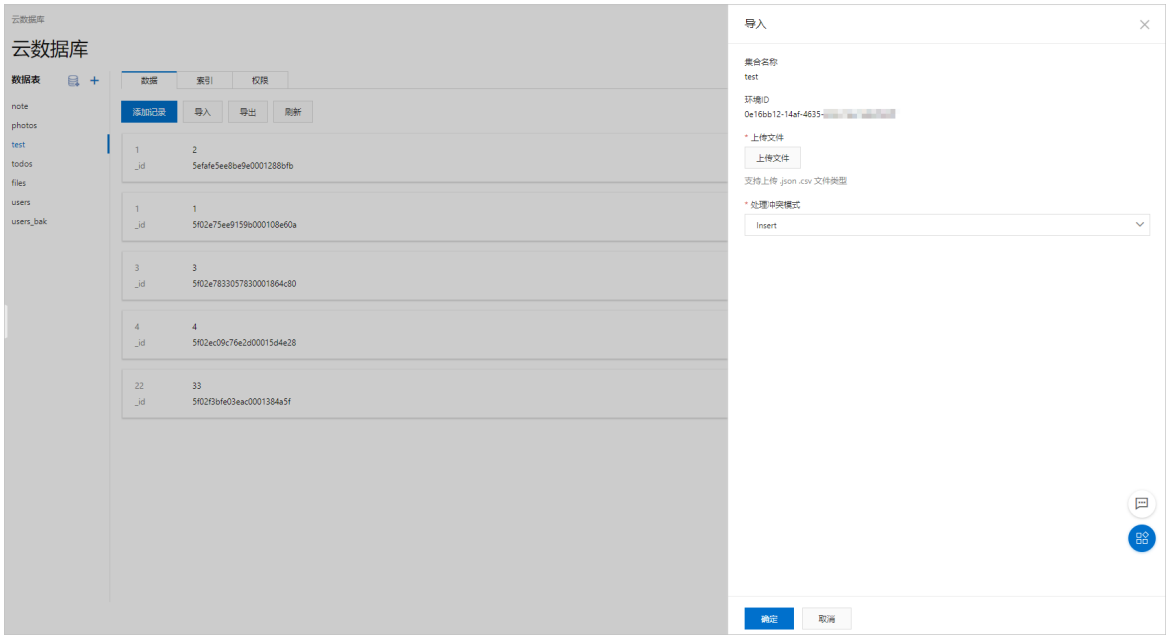
CSV文件格式要求首行作为导入后的键名，用英文逗号（,） 分隔，其余为键值记录，与首行键名一一对应。

导入支持两种**冲突处理模式**。

- Insert模式：会尝试插入文件中的数据，若导入过程中出现了主键冲突或者惟一索引冲突，则忽略该记录，后面的数据不受影响。
- Upsert模式：会判断该记录是否存在，若已存在则更新记录，若不存在则插入记录，若出现了惟一索引冲突则会直接中止导入操作。

控制台操作步骤：

1. 在云数据库管理页面，单击**目标数据库**。
2. 在**数据**页签单击**导入**。
3. 在导入页面，单击**上传文件**，选择本地文件完成上传后，选择冲突处理模式，最后单击**确定**。



说明

- 若文件中没有_id字段，导入过程中会自动生成_id。
- 导入文件的内容格式与导出文件的格式相同，要求必须是UTF-8编码格式。导入文件最大支持50M。
- 导入操作不是原子操作，可能会出现部分数据导入成功，部分数据导入失败的情况。
- 若您导入的数据量比较大，导入需要较长的时间，建议您耐心等待导入结果。

错误码说明

错误码	描述信息
SERVER_BUSY	服务端繁忙，建议您稍后再试。
INVALID_FILE_FORMAT	导入时，文件格式错误，建议您检查文件格式。
DUPLICATE_KEY_ERROR	导入时，出现了惟一索引冲突，建议您检查原有数据和导入数据，检查惟一索引设置。
INTERNAL_ERROR	服务端错误，建议您重试，如果仍然失败请提交工单。

3.10.5. 数据库事务

在云数据库操作中，对单个Document的操作是原子的，您应该尽量使用嵌入式Document设计来避免跨Document或者跨Collection事务。如果原子操作和嵌入式结构设计无法满足您的需求，您可以使用事务相关API来实现多Document事务。

云数据库事务说明

1. 事务要么完全执行，要么不执行，不会存在部分成功的现象。
2. 在事务提交之前，事务中的数据更新在事务外完全不可见。在提交事务之后，如果事务操作了多条数据，有可能会出现一部分先可见、另一部分后可见的情况，但是会保证最终一致性。如果事务回滚了，事务中的修改在事务外一直都是不可见的。
3. 在事务过程采用的是快照隔离，在快照隔离中：
   - 读操作返回的是对象的快照，而非实际数据。
   - 写操作会改变对象的快照，保证接下来事务内读到的数据是相同的。
   - 写操作会给对象添加事务锁，如果一个对象上有事务锁，其它的事务对该对象的写入会操作失败，非事务对该对象的写入会阻塞，直到事务锁释放。
4. 为避免大量的锁冲突，目前仅支持在事务中对单个Document的读写操作。

   支持事务的命令有：

   ```
   insertOne, findOne, updateOne, deleteOne, replaceOne,
   findOneAndUpdate,findOneAndReplace,findOneAndDelete
   ```

   。
5. 仅支持在云函数里面使用事务，不支持客户端直接访问云数据库时使用事务。
6. 事务的有效期为10s，若一个事务在有效期内没有提交，则会自动回滚。

事务API

1. 开启事务。

   ```
   const transaction = await ctx.mpserverless.db.startTransaction();
   ```

2. 在事务中操作单行数据。

```
const findOneResult = await transaction.collection('user').insertOne(
    {name: 'xxxx', "age": 19},
);
```

3. 提交事务。

```
await transaction.commit();
```

4. 回滚事务。

```
await transaction.rollback();
```

3.10.6. 使用地理位置数据

本文介绍如何在小程序Serverless的数据库中插入、创建索引和查询地理位置数据。

插入地理位置数据

小程序Serverless支持以下几种地理位置数据类型。

地理位置数据类型	描述	示例
Point	点	`{ type: "Point", coordinates: [ 40, 5 ] }`
LineString	线	`{ type: "LineString", coordinates: [ [ 40, 5 ], [ 41, 6 ] ] }`
Polygon	多边形	`{ type: "Polygon", coordinates: [ [ [ 0 , 0 ] , [ 3 , 6 ] , [ 6 , 1 ] , [ 0 , 0 ] ] ] }`
MultiPoint	点集合	`{ type: "MultiPoint", coordinates: [ [ -73.9580, 40.8003 ], [ -73.9498, 40.7968 ], [ -73.9737, 40.7648 ], [ -73.9814, 40.7681 ] ] }`

地理位置数据类型	描述	示例
MultiLineString	线段集合	`{ type: "MultiLineString", coordinates: [ [ [ -73.96943, 40.78519 ], [ -73.96082, 40.78095 ] ], [ [ -73.96415, 40.79229 ], [ -73.95544, 40.78854 ] ], [ [ -73.97162, 40.78205 ], [ -73.96374, 40.77715 ] ], [ [ -73.97880, 40.77247 ], [ -73.97036, 40.76811 ] ] ] }`
MultiPolygon	多边形集合	`{ type: "MultiPolygon", coordinates: [ [ [ [ -73.958, 40.8003 ], [ -73.9498, 40.7968 ], [ -73.9737, 40.7648 ], [ -73.9814, 40.7681 ], [ -73.958, 40.8003 ] ] ], [ [ [ -73.958, 40.8003 ], [ -73.9498, 40.7968 ], [ -73.9737, 40.7648 ], [ -73.958, 40.8003 ] ] ] ] }`

插入地理位置数据示例：

```
mpserverless.db.collection('places').insertMany([
      {
        location: { type: "Point", coordinates: [-73.88, 40.78] },
        name: "La Guardia Airport",
        category: "Airport"
      },
      {
        location: { type: "LineString", coordinates: [[113, 23], [120, 50]] },
        name: "ss Airport",
        category: "Airport"
      }
]}
```

创建地理位置索引

注意 给一个字段创建了**地理位置**类型的索引之后，该字段只能存储地理位置数据类型的数据。

您可以通过控制台或者调用RunDBComand接口来创建**地理位置**类型的索引。以下为使用RunDBComand接口创建地理位置索引的Body参数示例。

```
{
    "command": "createIndex",
    "collection": "places",
    "field": {
        "location": "2dsphere"
    },
    "options": {
        "name": "location_2dsphere",
        "unique": false
    }
}
```

查询地理位置数据

您可以在小程序端或者云函数中来使用地理位置查询，两者的查询语法是相同的。支持以下几种查询方法。

- **nearSphere**

  按从近到远的顺序，找出字段值在给定地点附近的记录。

  > **说明** 使用nearSphere必须给查询的字段添加地理位置索引，否则会查询失败。

  示例代码：

```
mpserverless.db.collection('places').find(
  {
    location: {
      $nearSphere: {
        $geometry: {
          type: "Point",
          coordinates: [-73.9667, 40.78]
        },
        $minDistance: 1000,
        $maxDistance: 5000
      }
    }
  }
);
```

  nearSphere查询方法相关参数说明

属性	类型	必填	说明
$geometry	Point	是	给定的地理位置点
$minDistance	number	否	最小距离，单位米
$maxDistance	number	否	最大距离，单位米

- **geoWithin**

找出字段值在给定区域内的记录。给定区域可以是Polygon、MultiPolygon或者CenterSphere。

> **说明** CenterSphere表示一个圆形，对应的值的定义是：`[ [经度，纬度]，半径 ]`。半径需以弧度计，例如需要10km的半径，则取距离除以地球半径6378.1km得出的数字。

示例代码1：

```
mpserverless.db.collection('places').find(
  {
    location: {
      $geoWithin: {
        $geometry: {
          type: "Polygon",
          coordinates: [
            [[0, 0], [30, 20], [20, 30], [0, 0]],
            [[10, 10], [16, 14], [14, 16], [10, 10]]
          ]
        }
      }
    }
  }
);
```

geometry参数说明

属性	类型	必填	说明
$geometry	Polygon或MultiPolygon	是	给定的区域

示例代码2：

```
mpserverless.db.collection('places').find(
  {
    location: {
      $geoWithin: {
        $centerSphere: [ [ -88, 30 ], 10/6378.1 ] }
      }
    }
  }
);
```

centerSphere参数说明

属性	类型	必填	说明
$centerSphere	CenterSphere	是	给定的区域

- **geoIntersects**

  找出字段值与给定的地理位置相交的记录。

  示例代码：

```
mpserverless.db.collection('places').findOne(
  {
    location: {
      $geoIntersects: {
        $geometry: {
          type: "Point",
          coordinates: [-73.88, 40.78]
        }
      }
    }
  }
);
```

geoIntersects查询方法相关参数说明

属性	类型	必填	说明
$geometry	地理位置类型	是	给定的位置

- **geoNear（聚合查询）**

  适用于聚合阶段，将记录按照给定点从近到远输出。

  示例代码：

```
mpserverless.db.collection('places').aggregate(
  [
    {
      $geoNear: {
        near: { type: "Point", coordinates: [113.323809, 23.097732] },
        distanceField: "distance",
        maxDistance: 2,
        query: { category: "Parks" },
        includeLocs: "location",
        spherical: true
      }
    }
  ]
);
```

聚合查询相关参数说明

属性	类型	必填	说明
near	Point	是	给定的地理位置点
spherical	boolean	是	必填，值为true
distanceField	string	是	存放距离的输出字段名，可以用点表示法表示一个嵌套字段
maxDistance	number	否	最大距离，单位米

属性	类型	必填	说明
query	document	否	要求记录必须同时满足该条件
distanceMultiplier	number	否	返回时在距离上乘以该数字
includeLocs	string	否	列出要用于距离计算的字段，如果记录中有多个字段都是地理位置时有用
key	string	否	指定使用的索引字段，当集合中有多个地理位置索引时使用

注意

1. geoNear只能出现在pipeline的第一个阶段。
2. 必须设定distanceField，distanceField用于指定包含计算距离的字段。
3. 使用geoNear必须给集合添加地理位置索引，如果集合包含了多个地理位置索引，则要使用key属性来指定使用哪个字段。
4. 在geoNear中的query里无法使用nearSphere（或者near）断言。

4.云存储

4.1. 文件权限管理

4.1.1. 文件权限管理

小程序Serverless提供了一套简明易懂的JSON语法用来控制用户对资源的访问，类似于身份验证体系里的IAM或者网络权限中的ACL。您可以在控制台上修改文件权限。

规则语法

一组权限规则由多个子规则组成，每个子规则包含以下信息：

- 范围：用于定义规则生效范围。

  在数据存储的安全规则中，每一个规则的范围是一个文件（路径）。您可以通过文件路径指定一个特定的文件， 也可以通过JavaScript的正则表达式指代一批文件或使用星号（*）指代所有文件。以下是三种范围书写方式示例：

  - 文件路径： `xxx/yyy/zzz.jpg` 。范围是这个特定文件。
  - 正则表达式： `/.*\\.png/` 。范围是所有后缀为*.png*的文件。
  - 所有集合： `*` 指代所有集合。

- 操作：用于定义生效范围内的特定操作。所有人都具有文件的读权限。

  文件存储提供以下操作定义规则：

  - `.write` ：写操作
  - `*` ：所有操作

- 策略：用于确定生效范围内的特定操作是否允许。默认没有标记为允许的操作，都是不允许的。

  策略是操作是否被允许的标记，可以是一个布尔值，或者是一个表达式。当策略冲突时，以最先出现的规则策略为准。

  表达式通用对象

顶级对象	字段名称	描述
request.auth	userId	请求用户ID。
resource.auth	userId	资源所属用户ID。

  `userId` 指的是通过 `mpserverless.user.getInfo()` 获取的userId字段。

  表达式运算符

操作符	描述	匹配类型
==	等于	string number

操作符	描述	匹配类型
!=	不等于	string number
>	大于	number
>=	大于等于	number
<	小于	number
<=	小于等于	number
!	非	string number

默认规则

对于新创建的服务，小程序Serverless会自动提供一个默认规则。默认规则规定所有文件都只有资源所有者可以进行写操作，所有人可读。

```
{
  "file": {
    "*": {
      ".write": "request.auth.userId == resource.auth.userId"
    }
  }
}
```

规则示例

例如，您可以创建如下安全规则，满足不同的图片访问需求：

- 对于*.png*格式的图片只有图片所有者可以写，所有人可读。
- 某个特殊路径的图片只有所有者可以读写。
- 其他文件所有人都可读写。

> **注意** JSON结构的安全规则配置暂不支持//注释 。为例方便阅读，在文档中使用//进行说明，在实际使用时请去掉//注释。

```
{
    "file": {
        "/.*\\.png/": {
            ".write": "request.auth.userId == resource.auth.userId",
            "*": true
        },
        "xxx/yyy/zzz.jpg": {
            "*": "request.auth.userId == resource.auth.userId"
        },
        "*": {
            "*": true
        }
    }
}
```

其中：

- 第一个规则是针对*.png*格式的图片文件。 `/.*\\.png/` 是一个正则，匹配所*.png*图片作为规则范围。 `.read` 是具体操作描述， `*` 是全部其他操作的描述。规则内容为*.png*格式的图片只有资源所有者可以进行文件写操作，所有人可以进行读取操作。
- 第二个规则是针对 `xxx/yyy/zzz.jpg` 这个特定路径的图片，只有资源所有者可以进行文件读写操作。
- 第三个规则是针对除了前两个规则匹配到的文件之外的所有文件，所有人都可以进行读写操作。

修改文件权限

详细说明，请参见设置文件权限。

4.1.2. 设置文件权限

小程序Serverless会为每个新建的文件自动提供一个默认权限规则。默认规则规定所有文件都只有资源所有者可以进行写操作，所有人可读。您可以在控制台上修改文件权限。

前提条件

确保您已经了解小程序云Serverless服务的文件权限设置规则。

小程序Serverless提供了一套简明易懂的JSON语法用来控制用户对资源的访问，类似于身份验证体系里的IAM或者网络安全中的ACL。您可以通过修改权限规则来管理文件权限。更多详细信息，请参见文件权限管理。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云存储**。
3. 单击**权限**页签。
4. 修改权限规则，然后单击**保存**。

4.2. 上传文件

您可以直接在控制台上上传小程序所需的文件。

前提条件

上传文件前，确保您以满足以下条件：

- 创建服务空间。
- 支持上传图片、音频和视频文件，格式要求如下：
  - 图片：.jpg .jpeg .png .gif .bmp .wbmp .webp .svg .image
  - 音频：.audio .au .mp2 .mp3
  - 视频：.3gpp .mp4 .mpeg .flv .avi .wmv .mpg .mpga
- 要上传的图片名称必须满足以下条件：
  - 使用 UTF-8 编码。
  - 图片名称区分大小写。
  - 长度必须在 1-1023 字节之间。
  - 不能以 / 或者 \ 字符开头。

控制台操作

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云存储**。
3. 单击**上传文件**。
4. 单击**直接上传**或将文件拖拽至上传区域。

   文件上传过程中，请勿刷新或关闭页面，否则上传任务会被中断且列表会被清空。
5. 上传成功后，您可以单击**详情**查看图片下载地址。

客户端调用

示例代码如下：

```
// 选择并上传图片，获得图片 URL
attach() {
  my.chooseImage({
    chooseImage: 1,
    success: res => {
      const path = res.apFilePaths[0];
      const options = {
        filePath: path,
        headers: {
          contentDisposition: 'attachment',
        },
      };
      mpserverless.file.uploadFile(options).then((image) => {
        console.log(image);
        this.setData({
          imageUrl: image.fileUrl,
        });
      }).catch(console.log);
    },
  });
}
```

相关文档

- uploadFile

4.3. 删除文件

您可以直接在控制台上删除不需要的文件。

控制台操作

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 云存储**。
3. 找到目标文件，然后单击**删除**。
4. 在弹出的对话框中，单击**确定**。

客户端调用

示例代码如下：

```
const options = {
  filePath: path,
  headers: {
    contentDisposition: 'attachment',
  },
};
mpserverless.file.uploadFile(options).then((image) => {
  mpserverless.file.deleteFile({ fileUrl: image.fileUrl });
});
```

云函数调用

示例代码如下：

```
module.exports = async (ctx) => {
  // ctx.args 是从客户端传过来的参数
  const res = await ctx.mpserverless.file.deleteFile({ fileUrl: ctx.args.fileUrl });
  return res;
};
```

相关文档

- deleteFile

4.4. 文件管理最佳实践

`mpserverless.file` 对象提供 `uploadFile` 和 `deleteFile` 方法管理文件。上传的文件将通过CDN进行网络加速。

文件格式

文件存储服务目前支持上传图片、音频和视频文件。文件格式要求如下表所示。

文件类型	文件格式
图片	jpg/jpeg、png、gif、bmp、wbmp、webp、svg
音频	mp3
视频	mp4、ogg、webm

上传文件

以下代码展示了如何通过 `uploadFile` 方法以指定路径的方式上传一张公开的图片。

```
const options = {
  // env: public, // 不指定存储方式 env 的情况下，默认为公开（public）
  filePath: path,
  headers: {
    contentDisposition: 'attachment',
  },
};
mpserverless.file.uploadFile(options);
```

删除文件

以下代码展示了如何通过 `deleteFile` 方法根据文件的URL删除一个文件。

```
const options = {
  filePath: path,
  headers: {
    contentDisposition: 'attachment',
  },
};
mpserverless.file.uploadFile(options).then((image) => {
  mpserverless.file.deleteFile(image.fileUrl);
});
```

示例教程

以一个图片管理小程序为例，当用户新增一个图片条目的时候，需要上传一个图片作为附件。那么您需要让用户在添加图片时，在客户端选择一个图片上传到Serverless文件存储服务，并将得到的图片路径记录在数据存储中。

首先，增加 `attach()` 方法，作为上传图片的事件处理，用于上传图片和获取图片路径。

```
// client/add-image/add-image.js
// 选择并上传图片，获得图片 URL
attach() {
  my.chooseImage({
    chooseImage: 1,
    success: res => {
      const path = res.apFilePaths[0];
      const options = {
        filePath: path,
        headers: {
          contentDisposition: 'attachment',
        },
      };
      mpserverless.file.uploadFile(options).then((image) => {
        console.log(image);
        this.setData({
          imageUrl: image.fileUrl,
        });
      }).catch(console.log);
    },
  });
},
```

然后，提交新增图片条目时，将图片URL作为数据一并存储到数据存储中。

```
// client/add-image/add-image.js
// 将新的图片内容添加到当前用户的图片列表中
add() {
  // 如果图片名称没有填，或者没有上传图片，则进行提示
  if (this.data.inputValue == '' || !this.data.imageUrl) {
    my.alert({
      title: '添加失败',
      content: '请填写图片名称和上传图片。',
      buttonText: '我知道了',
    });
  // 正常情况，写入数据存储
  } else {
    mpserverless.user.getInfo().then((user) => {
      mpserverless.db.collection('images').insertOne({
        text: this.data.inputValue,
        url: this.data.imageUrl ? this.data.imageUrl : false,
        userId: user.userId,
        uploadTime: new Date(),
      }).then(() => {
        my.navigateBack();
      }).catch(console.log);
    }).catch(console.log);
  }
},
```

5.静态网站托管

5.1. 静态网站托管功能介绍

小程序Serverless提供静态网站托管服务，您可以将小程序中的H5页面等静态资源放在阿里云中托管。

静态网站即为只包含静态内容（例如图片、视频、音频、HTML、CSS、JS、字体）的网站，托管在小程序Serverless之后，这些静态内容的分发由对象存储OSS和内容分发网络CDN进行支持，提供给您的用户访问。

静态网站托管的使用流程如下：

1. 开通小程序云服务。
2. 创建服务空间。
3. 开通静态网站托管功能。
4. 上传文件。
5. 绑定自定义域名。
6. 为自定义域名绑定证书。
7. 更新网站配置。
8. 访问静态网站。

用量限制

小程序Serverless公测期间，我们提供H5静态网站托管功能的免费体验资源，默认限制单个服务空间在H5静态网站托管空间中上传的文件总存储量最大为2GB。如果您在此期间有业务上的合理诉求，需要调整该限制，请提交工单以获得技术支持。

5.2. 开通静态网站托管功能

在您首次使用静态网站托管功能前，您需要开通静态网站托管功能。

前提条件

创建服务空间

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 选择**当前服务空间**，单击**立即开通**。

4. 请等待3~5分钟至开通完成。

相关文档

- OpenWebHostingService
- GetWebHostingStatus

5.3. 文件管理

5.3.1. 添加文件夹

静态文件托管也支持创建多级路径，本文介绍如何在控制台添加文件夹。

前提条件

开通静态网站托管功能

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击文件夹名称，进入对应路径下，单击**新建文件夹**。

5. 输入文件夹名称，然后单击**确定**。

5.3.2. 上传文件

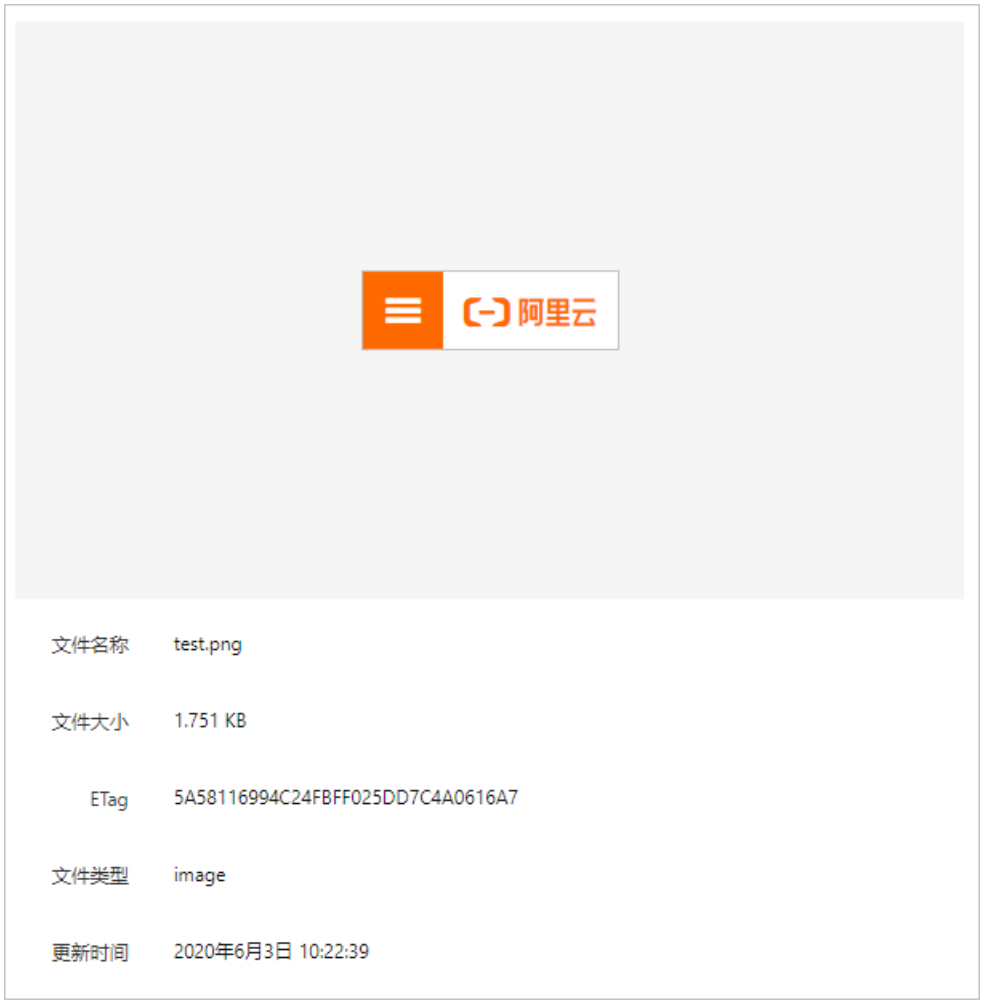
您可以直接在控制台上上传托管网站所需的静态文件。

前提条件

开通静态网站托管功能

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击文件夹名称，进入对应路径下，单击**上传文件**。
5. 单击**直接上传**或将文件（夹）拖拽至上传区域。
6. 上传成功后，您可以单击**详情**查看文件的基本信息。



> **说明** 小程序云Serverless公测期间，我们提供H5静态网站托管功能的免费体验资源，默认限制单个服务空间在H5静态网站托管空间中上传的文件总存储量最大为2GB。如果您在此期间有业务上的合理诉求，需要调整该限制，请提交工单以获得技术支持。

相关文档

- GetWebHostingUploadCredential

5.3.3. 删除文件

您可以直接在控制台上删除不需要的文件或文件夹。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 找到目标文件（夹），然后单击**删除**。

5. 在弹出的对话框中，单击**确定**。

相关文档

- DeleteWebHostingFile

5.4. 域名管理

5.4.1. 添加域名

本文介绍如何在阿里云控制台上为您的静态网站添加自定义访问域名。

前提条件

开通静态网站托管功能

背景信息

静态网站托管功能给您分配了一个默认域名，仅供测试使用，请求速度会受到限制。如您需要对外正式提供网站服务，请绑定您已备案的自定义域名。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击**设置 > 添加域名**。
5. 输入您的自定义域名，然后单击**确定**。

> 说明
> - 加速区域含中国内地时，加速域名必须备案。由于工信部备案系统有数据延迟，刚完成备案的域名请在8个小时以后再配置。
> - 添加成功后，立即刷新可能无法看到新增域名。更新需要1~2分钟的同步时间，请稍后刷新页面查看。
> - 待审核通过后，需要在域名解析服务商处，配置为您分配的CName记录，才能通过您的自定义域名访问托管的静态网站。

相关文档

- BindWebHostingCustomDomain
- ListWebHostingCustomDomains

5.4.2. 删除域名

本文介绍如何在阿里云控制台删除您的自定义域名。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击**设置**。
5. 找到要删除的域名，单击**删除**。
6. 在弹出的对话框，单击**确定**。

> 说明　域名删除后，请到域名解析服务商处删除该域名相关的解析记录。

相关文档

- UnbindWebHostingCustomDomain

5.4.3. 验证域名归属权

本文介绍如何验证域名归属权。

背景信息

当您首次将一个新域名添加至小程序云静态网站托管时，需要您完成验证域名的归属权。验证通过后，小程序云视为您拥有该域名。您在小程序云控制台再次添加该域名或子域名时，不需要再进行验证。小程序云为您提供DNS解析验证和文件验证两种方法，您可以任选其一。本文以添加 a.com 为例，为您介绍验证域名归属权的操作方法。

方法一：DNS解析验证（推荐）

添加新的加速域名时，DNS验证方法如下：

1. 获取加速域名的记录类型、主机记录及记录值。

i. 登录小程序云控制台。

ii. 在左侧导航栏，选择**静态网站托管 > 设置 > 添加域名**。当您首次添加新域名时，需要按页面提示添加指定的TXT记录。

2. 添加记录。

i. 登录 阿里云域名解析控制台

ii. 进入域名解析页面，在a.com域名的右侧，单击解析设置。

iii. 单击添加记录。填写步骤1中查看的记录类型、主机记录及记录值。

3. 验证。

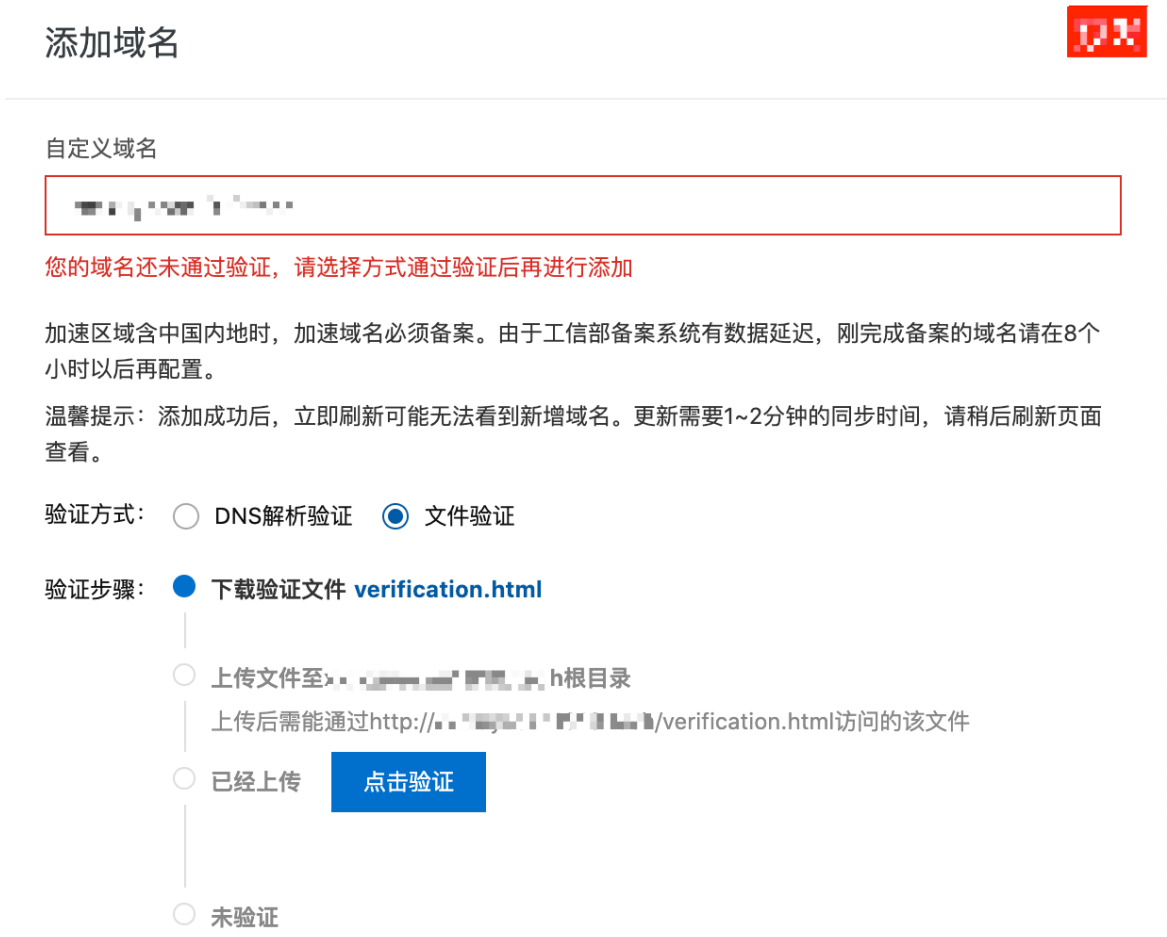
i. 返回小程序云控制台。

ii. 单击**点击验证**，完成验证。

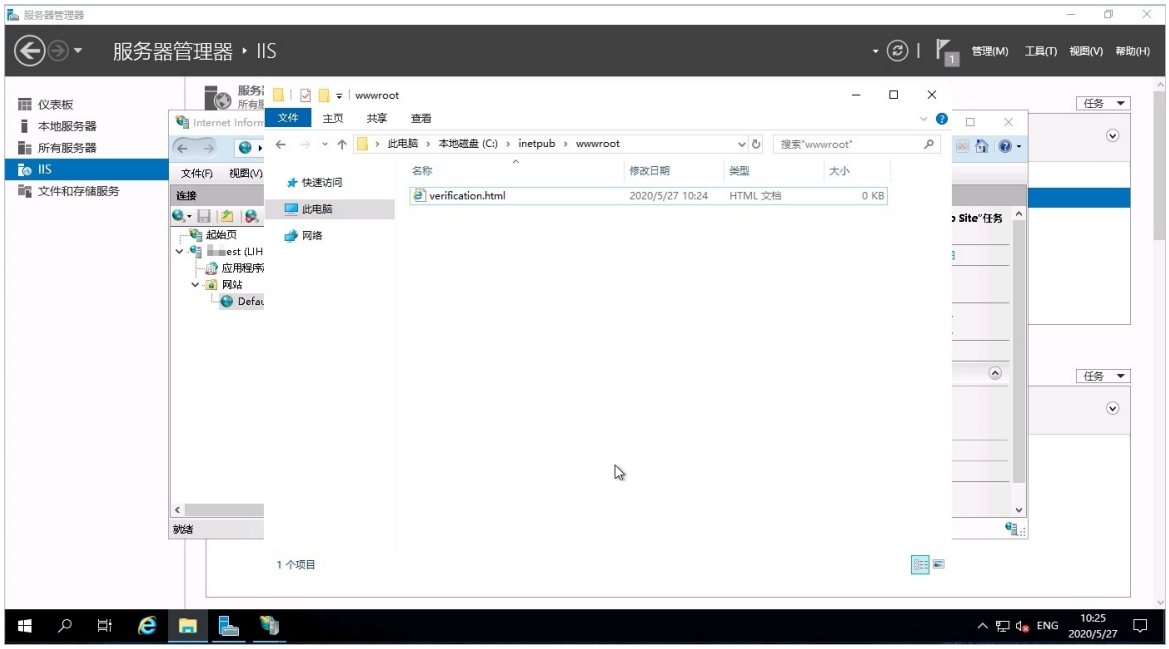
方法二：文件验证

添加新的加速域名时，文件验证方法如下：

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**静态网站托管 > 设置 > 添加域名**，验证方式选择**文件验证**。



3. 单击verification.html下载验证文件。
4. 将验证文件上传到您的域名源站服务器（例如您的ECS、OSS、CVM、COS、EC2等）的根目录。



> **说明** 小程序云系统后台将访问您的源站 `http://a.com/verification.html` 获取校验文件，判断您是否按要求上传指定的验证文件。请确保该文件可被访问。

5. 单击**点击验证**，完成验证。

常见问题

在进行添加新的加速域名时，您可能会遇到如下问题：

- Q：为什么要做域名归属权验证？

  A：为了确保域名只被真正的拥有者添加，避免出现用户A的域名被用户B添加导致的域名冲突及安全隐患。

- Q：我有多个阿里云账号，每个账号首次添加新域名时都需要做归属权验证吗？

  A：是的。多个账号视为多个不同的独立用户，每个账号都需要对新域名进行一次归属权验证。

- Q：我已完成DNS验证或文件验证，是否可以删除用作验证的DNS记录或文件？

  A：可以。要求您添加的DNS解析或文件，只用作添加域名时的归属权验证，验证通过后，您可以删除记录或文件。

- Q：我无法完成DNS验证或文件验证，怎么办？

  A：您可以提交工单

5.5. 证书管理

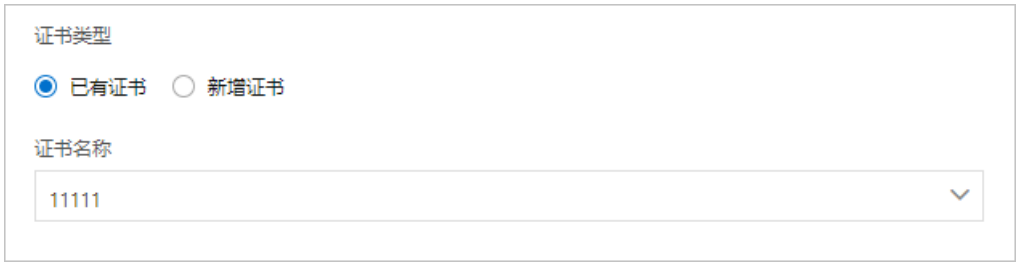
5.5.1. 绑定证书

本文介绍在阿里云控制台上为自定义域名绑定证书。

操作步骤

1. 登录小程序云控制台。

2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击**设置**。
5. 选择需要绑定证书的域名，单击**证书 > 绑定证书**。
6. 根据以下步骤填写证书信息，然后单击**确定**。
   i. 根据您的需要选择证书类型。
      - 已有证书：阿里云SSL证书。
      - 新增证书：第三方证书。
   ii. 配置证书信息。
      - 如果选择已有证书，请选择您的阿里云SSL证书名称。



      - 如果选择新增证书，请填写您的第三方证书内容和私钥。

相关文档

- AttachWebHostingCertificate

5.5.2. 解绑证书

本文介绍在阿里云控制台上解绑自定义域名的证书。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击**设置**。
5. 找到要解绑证书的域名，单击**证书 > 解绑证书**。
6. 在弹出的对话框中，单击**确定**。

相关文档

- DeleteWebHostingCertificate

5.6. 更新网站配置

本文介绍如何在阿里云控制台上更改网站的默认索引文档和错误文档配置。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 在页面右上角，选择要关联的环境。
4. 单击**设置 > 更新配置**。
5. 修改索引文档和错误文档名称，然后单击**确定**。

相关文档

- ModifyWebHostingConfig
- GetWebHostingConfig

5.7. 静态网站托管最佳实践

本文将介绍如何使用静态网站托管功能搭建一个静态网站。

步骤一：开通小程序云服务

在您首次使用小程序Serverless服务前，您需要开通小程序云服务。

1. 登录小程序云控制台。
2. 在**云产品开通**页面，单击《小程序云服务协议》链接。

3. 在阅读后，勾选**我已阅读并同意《小程序云服务协议》**复选框，然后单击**立即开通**。
4. 单击**前往RAM进行授权**。

5. 单击**同意授权**完成访问授权。

步骤二：创建服务空间

在使用静态网站托管功能前，您需要先创建一个服务空间。静态网站中的资源会和服务空间进行关联。每个服务空间之间相互独立。

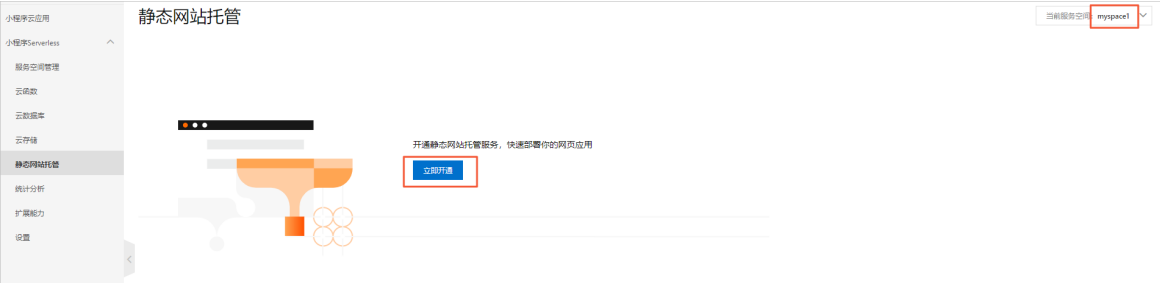
1. 在左侧导航栏，选择**小程序Serverless > 服务空间管理**。
2. 单击**创建服务空间**。
3. 输入服务空间名称和描述，然后单击**确定**。

   服务空间名称可包含3~15个字符，不能以数字和特殊字符开头。

步骤三：开通静态网站托管功能

在一个服务空间下首次使用静态网站托管功能前，您需要开通该服务空间的静态网站托管功能。

1. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
2. 选择**当前服务空间**，单击**立即开通**。



3. 请等待3~5分钟至开通完成。

步骤四：上传文件

1. 准备文件，本教程使用如下两个简单的HTML文件。
   - *index.html*文件内容如下：

```
<html>
<title>Index page</title>
<body>
<h2>Hello World!</h2>
</body>
</html>
```

   - *error.html*文件内容如下：

```
<html>
<title>Error page</title>
<body>
<h2>404 error</h2>
</body>
</html>
```

2. 在小程序云控制台左侧导航栏，选择**小程序Serverless > 静态网站托管**。
3. 单击**上传文件**。
4. 单击**直接上传**，选择*index.html*和*error.html*文件上传。

步骤五：更新网站配置

1. 在左侧导航栏，选择**小程序Serverless > 静态网站托管**。
2. 单击**设置 > 更新配置**。

3. 修改索引文档和错误文档名称，然后单击**确定**。

步骤六：绑定自定义域名

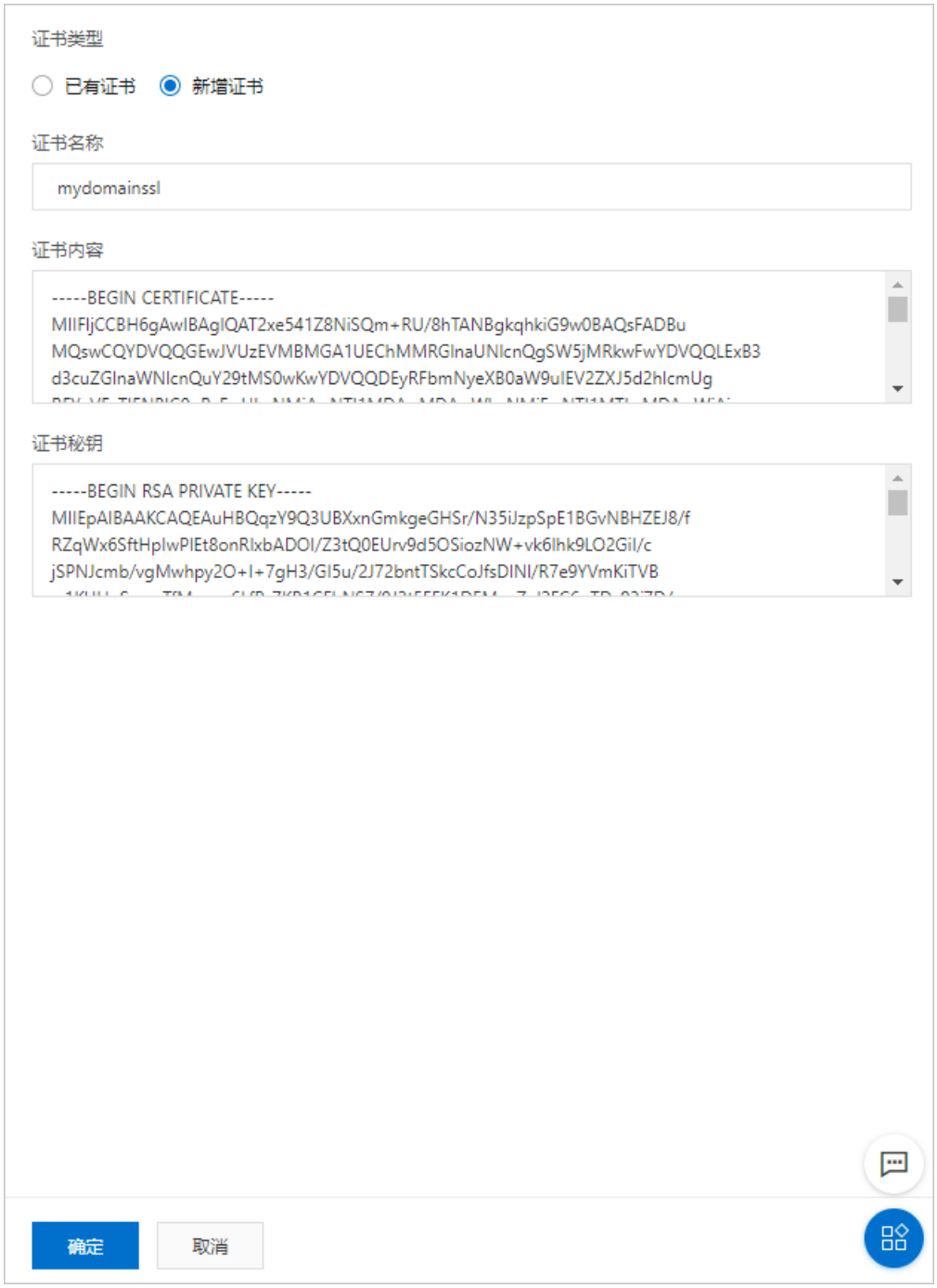
1. 在小程序云控制台左侧导航栏，选择**小程序Serverless > 静态网站托管**。
2. 单击**设置 > 添加域名**。
3. 输入您的自定义域名，然后单击**确定**。

> **说明**
>
> - 您输入的自定义域名必须是已备案的。由于工信部备案系统有数据延迟，刚完成备案的域名请在8个小时以后再配置。
> - 添加成功后，立即刷新可能无法看到新增域名。更新需要1~2分钟的同步时间，请稍后刷新页面查看。

4. 在域名解析服务商处，配置为您分配的CName记录。使用阿里云云解析添加CName解析记录，请参见添加解析记录。

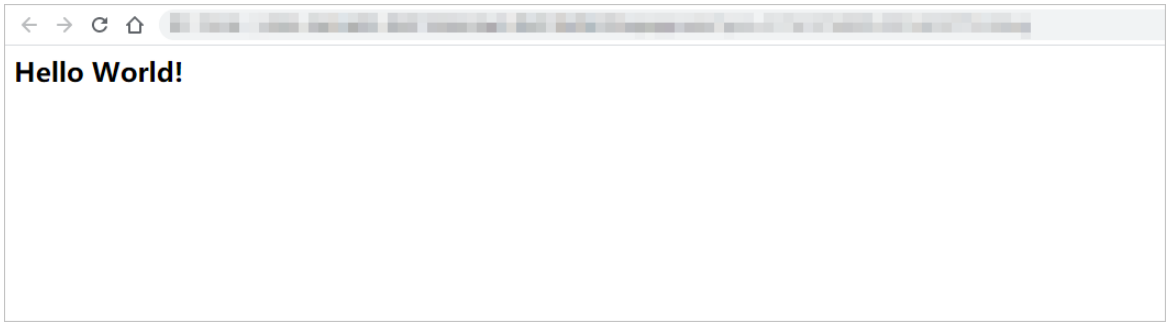
步骤七：为自定义域名绑定证书

1. 在小程序云控制台左侧导航栏，选择**小程序Serverless > 静态网站托管**。
2. 单击**设置**。
3. 选择需要绑定证书的域名，单击**证书 > 绑定证书**。
4. 本教程中使用第三方证书。证书类型选择*新增证书*，填写证书名称、证书内容和证书私钥，单击**确定**。

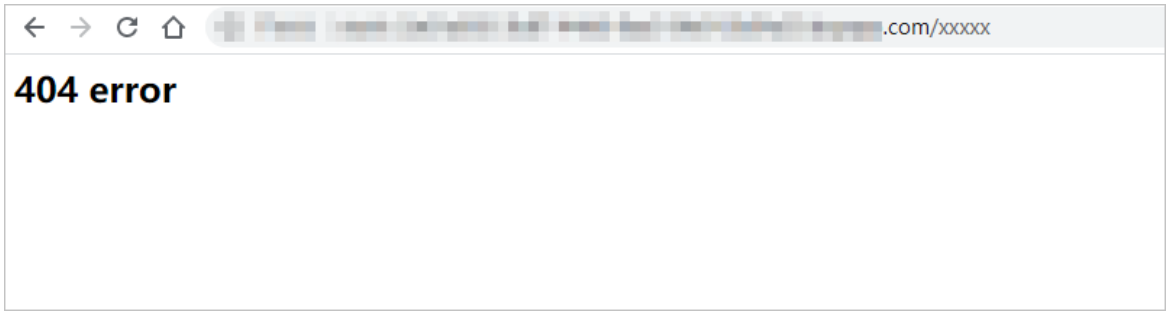


步骤八：访问静态网站

1. 在浏览器地址栏输入绑定的自定义域名，访问网站主页。



2. 在浏览器地址栏输入网站下任意不存在路径（例如 `www.example.com/xxxxx` ），将会显示*error.html*页面内容。



6.统计分析

6.1. 集成统计分析服务

6.1.1. 支付宝小程序集成

要为支付宝小程序接入统计分析服务，您需要先在控制台创建应用，然后在小程序项目中安装统计分析服务SDK。

前提条件

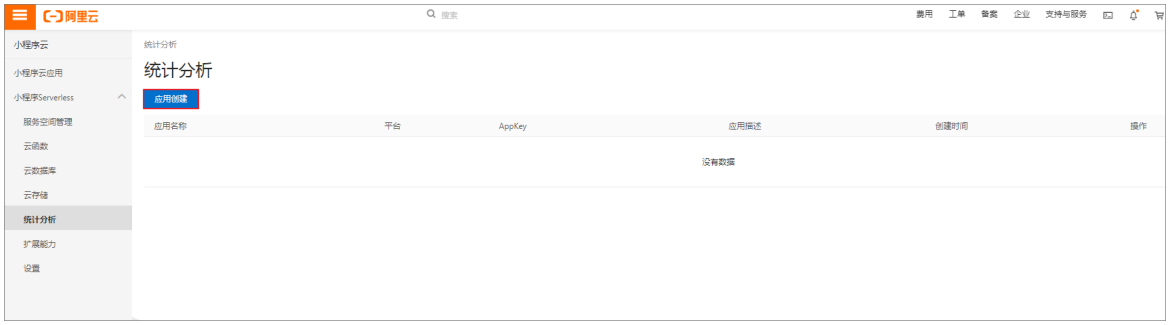
- 确保支付宝小程序版本不低于10.1.52，低版本将获取不到统计信息。
- 创建服务空间。

背景信息

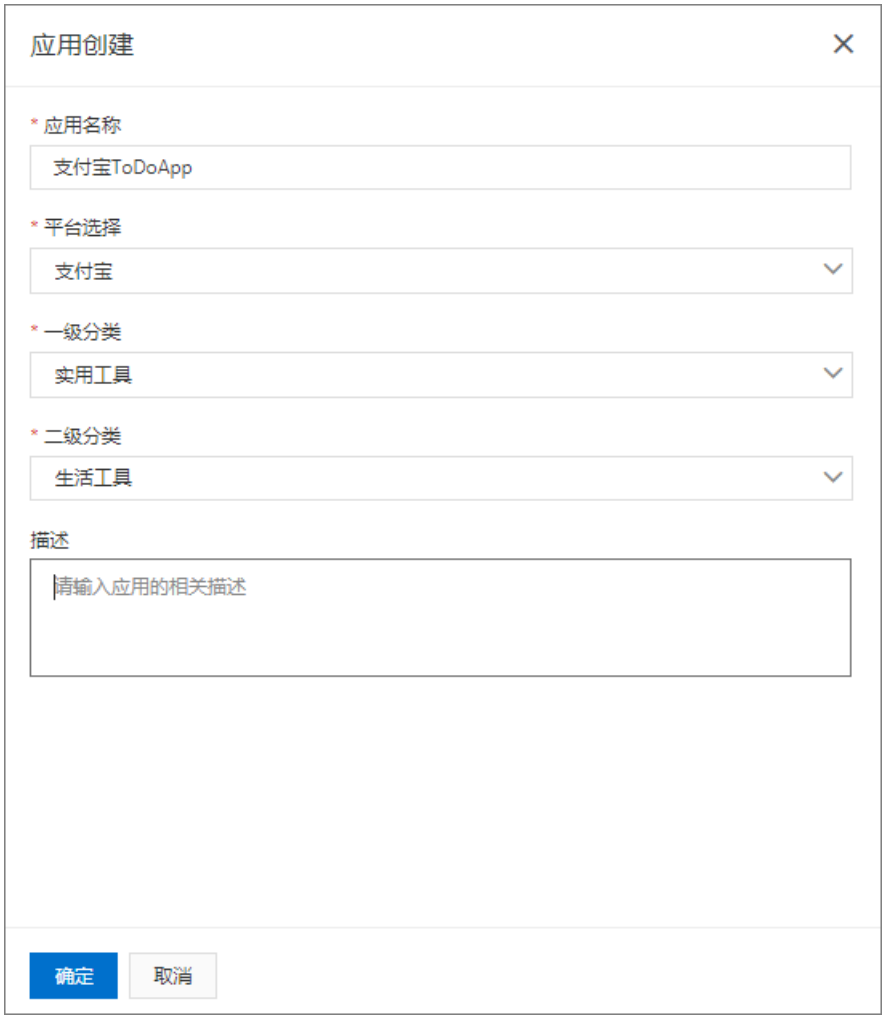
要接入支付宝小程序进行用户使用数据统计分析，您需要完成以下操作：

步骤一：创建统计分析应用

1. 登录小程序云管理控制台。
2. 选择服务空间。
3. 在左侧导航栏，单击**统计分析**，然后单击**创建应用**。



4. 根据提示配置应用信息，然后单击**确定**。



5. 记录生成的AppKey。

步骤二：安装SDK

1. 在小程序项目的根目录执行以下命令安装统计分析SDK。

```
npm i @alicloud/mpserverless-analyse-alipaysdk -- save
```

2. 执行以下命令安装小程序Serverless 客户端SDK。

```
npm install @alicloud/mpserverless-sdk@2.2.0 --save
```

> **说明** 客户端SDK版本不能低于2.2.0。

3. 在支付宝小程序项目的*app.js*文件中添加如下代码：

   在使用该代码时请注意：

说明

- 示例中的3个方法是SDK最基本的统计方法，请务必都引入。
- 请在小程序最初初始化时调用 `MPAnalyse.init()` 方法，请勿出现任何异步方法阻塞int方法的调用。
- 请在init方法中填入在统计分析页面中生成的AppKey，否则无法在统计分析页面中看到正确数据。
- 慎重调用 `clearStorage()` 以及 `clearStorageSync()` 接口。SDK会将用户相关操作数据缓存在客户端数据存储模块，在特定时间启动发送策略。若调用该接口可导致数据统计不准确的问题。
- 若用户在使用小程序过程强制关闭支付宝有可能会造成统计数据丢失的情况。

```
import MPAnalyse from '@alicloud/mpserverless-analyse-alipaysdk';
import MPServerless from '@alicloud/mpserverless-sdk'; ///SDK版本必须为2.2.0及以上
const serverless = new MPServerless({
  uploadFile: my.uploadFile,
  request: my.request,
  getAuthCode: my.getAuthCode
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint' // 服务空间地址，从小程序Serverless控制台处获得
});
App({
  onLaunch(options) {
    MPAnalyse.init('YOUR_APP_KEY', my, {      // 务必填入已注册的统计分析appKey，不然将无法统计
      mpServerless: serverless,              // 务必传入实例化MPServerless后的serverless
      platformParam: options                 // 若不需要渠道分析功能，可以暂不传入
    });
  },
  onShow() {
    MPAnalyse.resume();                       // 请务必引入
  },
  onHide() {
    MPAnalyse.pause();                        // 请务必引入
  }
});
```

其中：

- appId：小程序的ID。您可以在支付宝小程序开放平台获取小程序的App ID。
- spaceIdclientSecret和endpoint：在小程序Serverless控制台创建服务空间后可以获得。详情参见创建服务空间。
- MPAnalyse.init ：YOUR_APP_KEY是步骤一：创建统计分析应用中创建应用生成的AppKey。

- platformParam: options：如果需要进行渠道分析功能，必须引入该参数。

> **注意** 渠道分析将会统计用户首次进入小程序页面时带的渠道参数，请保证首次进入的url参数中带有sdk_channel字段。例如首次进入的URL为：pages/index/index?sdk_channel=test，*test*将会作为此次渠道的值。

4. 打开调试工具中的console选项，若看到有 `[SDK] -- 集成SDK成功` 提示。即表示SDK集成成功。

> **说明** 该方案仅提供验证是否集成成功的方法，由于在开发环境下可能会存在数据不完整的情况，请您正式发布小程序后在验证数据正确性。

步骤三：查看统计数据

1. 登录小程序云控制台。
2. 在左侧导航栏，单击**统计分析**。
3. 单击目标统计分析应用的**查看数据**链接进入统计分析页面。

6.1.2. 微信小程序集成

要为微信小程序接入统计分析服务，您需要先在控制台创建应用，然后在小程序项目中安装统计分析服务SDK。

前提条件

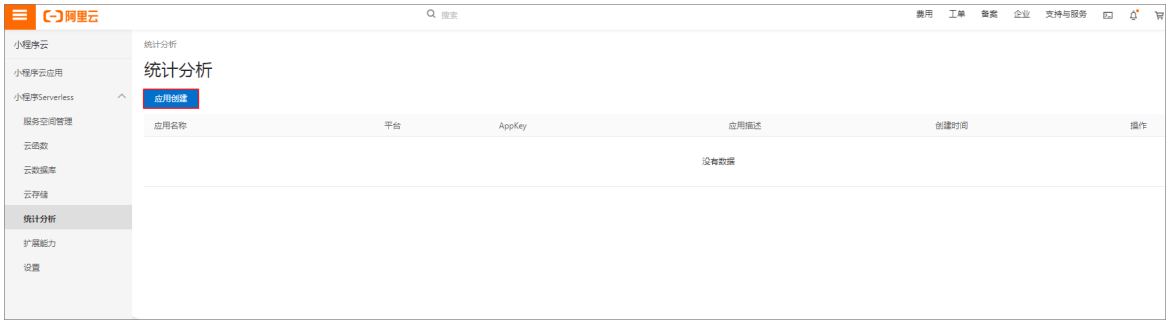
创建服务空间。

背景信息

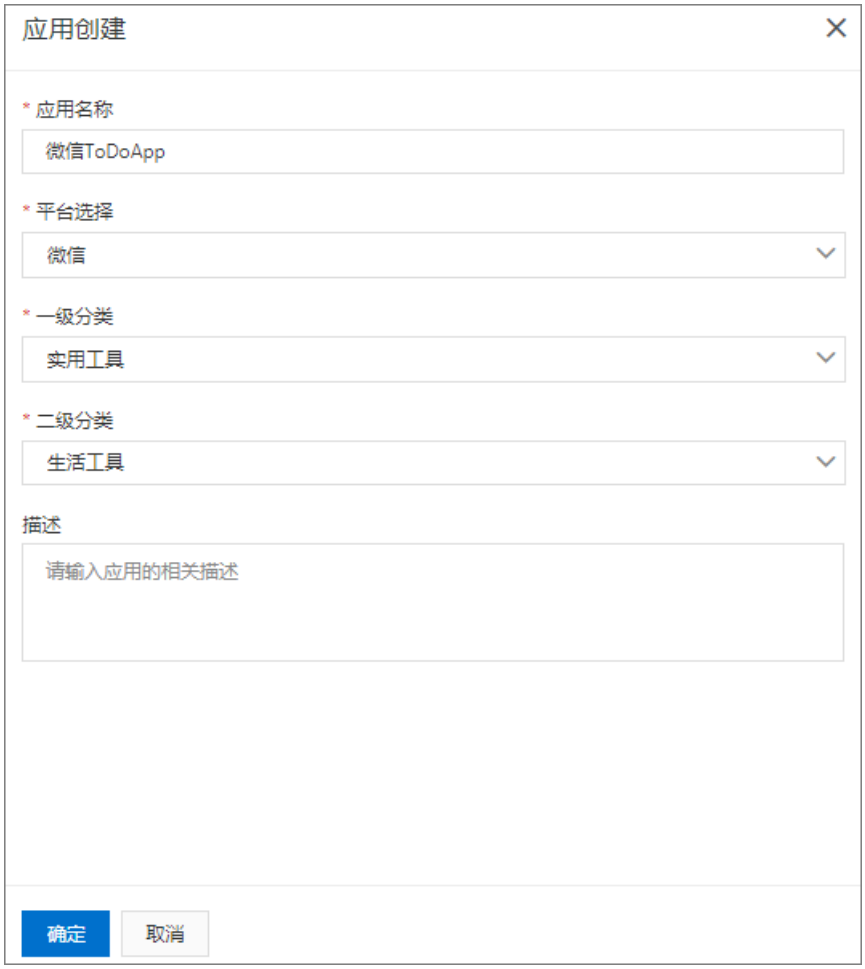
要接入微信小程序进行用户使用数据统计分析，您需要完成以下操作：

步骤一：创建统计分析应用

1. 登录小程序云管理控制台。
2. 在左侧导航栏，单击**统计分析**，然后单击**创建应用**。



3. 根据提示配置应用信息，然后单击**确定**。



4. 记录生成的AppKey。

步骤二：安装SDK

1. 在小程序项目的根目录执行以下命令安装统计分析SDK。

```
npm i @alicloud/mpserverless-analyse-wxsdk -- save
```

2. 单击这里下载小程序Serverless客户端SDK文件，将下载后的*mpserverless.js*文件添加到小程序项目目录下。

> **说明** 客户端SDK版本不能低于2.2.0。

3. 在微信小程序项目的*app.js*文件中添加如下代码：

   在使用该代码时请注意：

> **说明**
> - 示例中的3个方法是SDK最基本的统计方法，请务必都引入。
> - 请在小程序最初初始化时调用 `MPAnalyse.init()` 方法，请勿出现任何异步方法阻塞int方法的调用。
> - 请在init方法中填入在统计分析页面中生成的AppKey，否则无法在统计分析页面中看到正确数据。
> - 慎重调用 `clearStorage()` 以及 `clearStorageSync()` 接口。SDK会将用户相关操作数据缓存在客户端数据存储模块，在特定时间启动发送策略。若调用该接口可导致数据统计不准确的问题。
> - 若用户在使用小程序过程强制关闭微信有可能会造成统计数据丢失的情况。

```
const MPAnalyse = require('@alicloud/mpserverless-analyse-wxSdk');
const MPServerless = require('/sdk/mpserverless.js'); //客户端SDK版本必须为2.2.0及以上
const serverless = new MPServerless({
  uploadFile: wx.uploadFile,
  request: wx.request,
  getAuthCode: wx.login,
  getFileInfo: wx.getFileInfo,
  getImageInfo: wx.getImageInfo
  }, {
  appId: '1234456789', // 小程序应用标识
  spaceId: 'db4dd657-7041-470a-90xxxxx', // 服务空间标识
  clientSecret: '6c3c86xxxx6', // 服务空间 secret key
  endpoint: 'https://endpoint', // 服务空间地址，从小程序Serverless控制台处获得
});
App({
  onLaunch: function (options) {
    MPAnalyse.init('YOUR_APP_KEY', {            // 务必填入已注册的统计分析appKey，不然将无法
统计
      mpServerless: serverless,                 // 务必传入实例化MPServerless后的serverless
      platformParam: options                   // 若不需要渠道分析功能，可以暂不传入
    });
  },
  onShow (options) {
    MPAnalyse.resume();                         // 请务必引入
  },
  onHide () {
    MPAnalyse.pause();                          // 请务必引入
  }
});
```

其中：

- appId：小程序的ID。您可以在微信小程序开放平台获取小程序的App ID。

- spaceIdclientSecret和endpoint：在小程序Serverless控制台创建服务空间后可以获得。详情参见创建服务空间。
- MPAnalyse.init ：YOUR_APP_KEY是步骤一中创建统计分析应用生成的AppKey。

- platformParam: options：如果需要进行渠道分析功能，必须引入参数。

> **注意** 渠道分析将会统计用户首次进入小程序页面时带的渠道参数，请保证首次进入的url参数中带有sdk_channel字段。例如首次进入的URL为：pages/index/index?sdk_channel=test，*test*将会作为此次渠道的值。

4. 打开调试工具中的console选项，若看到有 `[SDK] -- 集成SDK成功` 提示。即表示SDK集成成功。

> **说明** 该方案仅提供验证是否集成成功的方法，由于在开发环境下可能会存在数据不完整的情况，请您正式发布小程序后在验证数据正确性。

步骤三：查看统计数据

1. 登录小程序云控制台。
2. 在左侧导航栏，单击**统计分析**。
3. 单击目标统计分析应用的**查看数据**链接进入统计分析页面。

6.2. 查看统计数据

接入统计分析服务后，您可以在小程序控制台中查看小程序运行的用户、渠道等数据，帮助您制定小程序运营策略。

前提条件

在查看统计数据前，确保您已经完成小程序的接入：

- 支付宝小程序集成
- 微信小程序集成

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，单击**统计分析**。
3. 单击目标统计分析应用的**查看数据**链接进入统计分析页面。

6.3. 基本统计指标说明

指标	说明
新增用户	第一次启动小程序的用户（以设备为判断标准）。
启动次数	打开小程序视为启动，完全退出或退至后台即视为启动结束。
活跃用户	启动过小程序的用户（去重），启动过一次的用户即视为活跃用户，包括新用户与老用户。
累计用户	截止到当前时间，启动过小程序的所有独立用户（去重，以设备为判断标准）。
平均日使用时长	日使用时长的均值，即全部用户的日使用时长/总活跃用户。
平均日启动次数	用户一天内启动小程序的次数，即总启动次数/活跃用户。
活跃用户构成	每日活跃用户中新用户与老用户的分布。
新增用户次日留存率	该日的新增用户在次日仍启动小程序的用户占该日新增的比例。
平均单次使用时长	单次使用时长的均值，即全部用户的日使用时长/总启动次数。

6.4. 概况分析

6.4.1. 整体趋势

整体趋势页面通过展示累计用户、活跃用户等基本指标的趋势图和TOP渠道数据，帮助您了解小程序使用的整体趋势和规模。

基本指标说明

指标	说明
新增用户	第一次启动小程序的用户（以设备为判断标准）。
启动次数	打开小程序视为启动，完全退出或退至后台即视为启动结束。
活跃用户	启动过小程序的用户（去重），启动过一次的用户即视为活跃用户，包括新用户与老用户。
累计用户	截止到当前时间，启动过小程序的所有独立用户（去重，以设备为判断标准）。
平均日使用时长	日使用时长的均值，即全部用户的日使用时长/总活跃用户。

指标	说明
平均日启动次数	用户一天内启动小程序的次数，即总启动次数/活跃用户。
活跃用户构成	每日活跃用户中新用户与老用户的分布。
新增用户次日留存率	该日的新增用户在次日仍启动小程序的用户占该日新增的比例。
平均单次使用时长	单次使用时长的均值，即全部用户的日使用时长/总启动次数。

整体趋势概览

提供截止至昨日小程序整体情况的7个指标数据，并计算这些指标的同比变化。

> **说明** 同比变化指和上一个周期（上个7日内）的数据变化。

整体趋势指标说明

指标	说明
新增用户（7日平均）	最近7日（不含今日）每日新增用户的平均值。
新用户次日留存率（7日平均）	最近7日次日留存率的平均值。
活跃用户（7日平均）	最近7日（不含今日）每日活跃用户的平均值。
使用时长（7日平均）	最近7日（不含今日）用户每日使用时长的平均值。
近7日总活跃用户数	最近7日（不含今日）活跃用户的总数（去重）。
累计用户数	截止到当前时间，启动过应用的所有独立用户（去重，以设备为判断标准）。
近30日总活跃用户数	单最近30日（不含今日）活跃用户的总数（去重）。

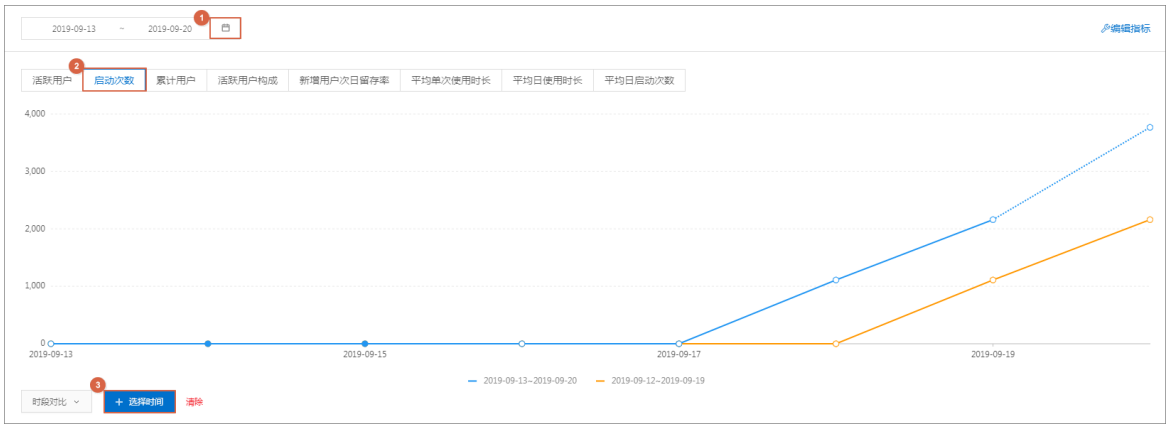
趋势变化与对比

提供新增用户、活跃用户、启动次数等9个常用指标的线图变化趋势，支持时间对比。默认只展示8个指标的趋势图，您可以单击**编辑指标**选择要查看的指标趋势。

整体趋势图支持添加时间对比。时间对比可以帮助您准确的获取对比时间内的各项指标的变化趋势，分析用户使用行为。

1. 单击趋势图上方的日历图标，选择趋势图的时间粒度。例如选择展示过去7天的数据趋势。
2. 选择要查看的指标。
3. 单击**+选择时间**然后选择对比时间。

> **说明** 对比时间的时间粒度和您步骤 1 中选择的时间粒度相同。

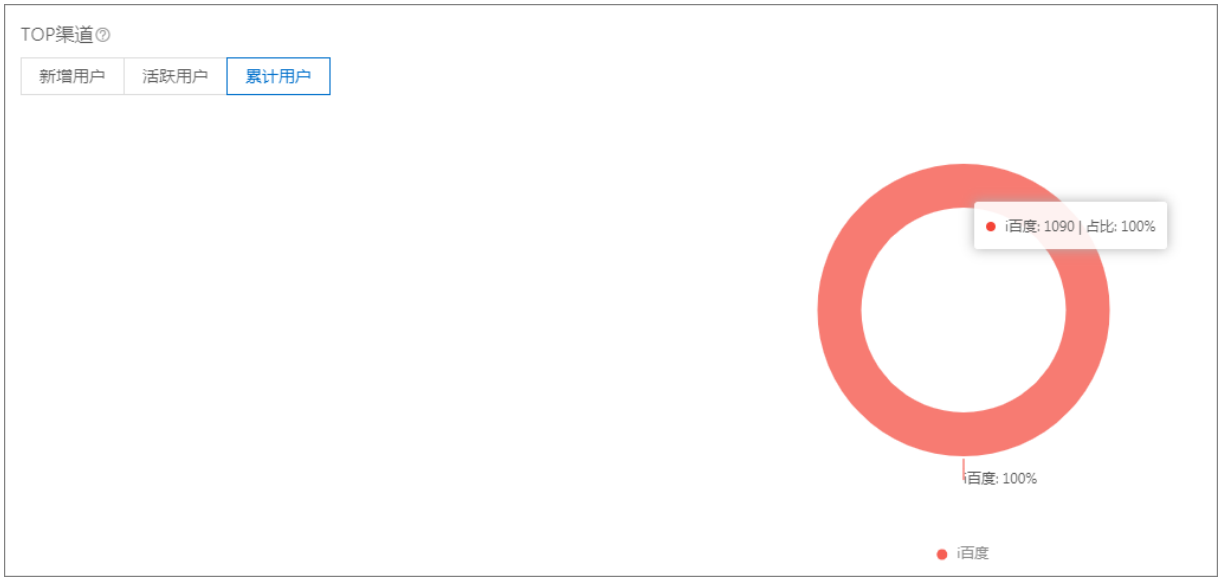


对比趋势指标说明

指标	说明
新增用户	第一次启动小程序的用户（以设备为判断标准）。
启动次数	打开小程序视为启动，完全退出或退至后台即视为启动结束。
活跃用户	启动过小程序的用户（去重），启动过一次的用户即视为活跃用户，包括新用户与老用户。
累计用户	截止到当前时间，启动过小程序的所有独立用户（去重，以设备为判断标准）。
平均日使用时长	日使用时长的均值，即全部用户的日使用时长/总活跃用户。
平均日启动次数	用户一天内启动小程序的次数，即总启动次数/活跃用户。
活跃用户构成	每日活跃用户中新用户与老用户的分布。
新增用户次日留存率	该日的新增用户在次日仍启动小程序的用户占该日新增的比例。
平均单次使用时长	单次使用时长的均值，即全部用户的日使用时长/总启动次数。

TOP渠道

提供昨日新增、活跃、累计用户排名Top 10渠道和其它渠道的占比情况。



- 新增用户：按照今日新增排名的Top 10渠道的今日与昨日的新增用户数量的占比。
- 活跃用户：按照今日活跃排名的Top 10渠道的今日与昨日的活跃用户数量的占比。
- 累计用户：按照累计用户排名的Top 10渠道的累计用户数量的占比。

6.4.2. 实时统计

实时统计页面主要展示今日实时的新增用户、启动次数等数据。

基本指标

展示截止至今日上个整点时刻的各指标的当日总数据，以及相对于昨日相同整点时刻的数据的变化率。

指标	说明
新增用户	第一次启动小程序的用户（以设备为判断标准）。
启动次数	打开小程序视为启动，完全退出或退至后台运行超过30秒后再次进入小程序，视为一次新启动。
时段累计日活	启动过小程序的用户（去重），启动过一次的用户即视为活跃用户，包括新用户与老用户。

指标	说明
分时活跃用户	活跃用户在24小时中的分布情况（每小时去重）。

6.5. 用户分析

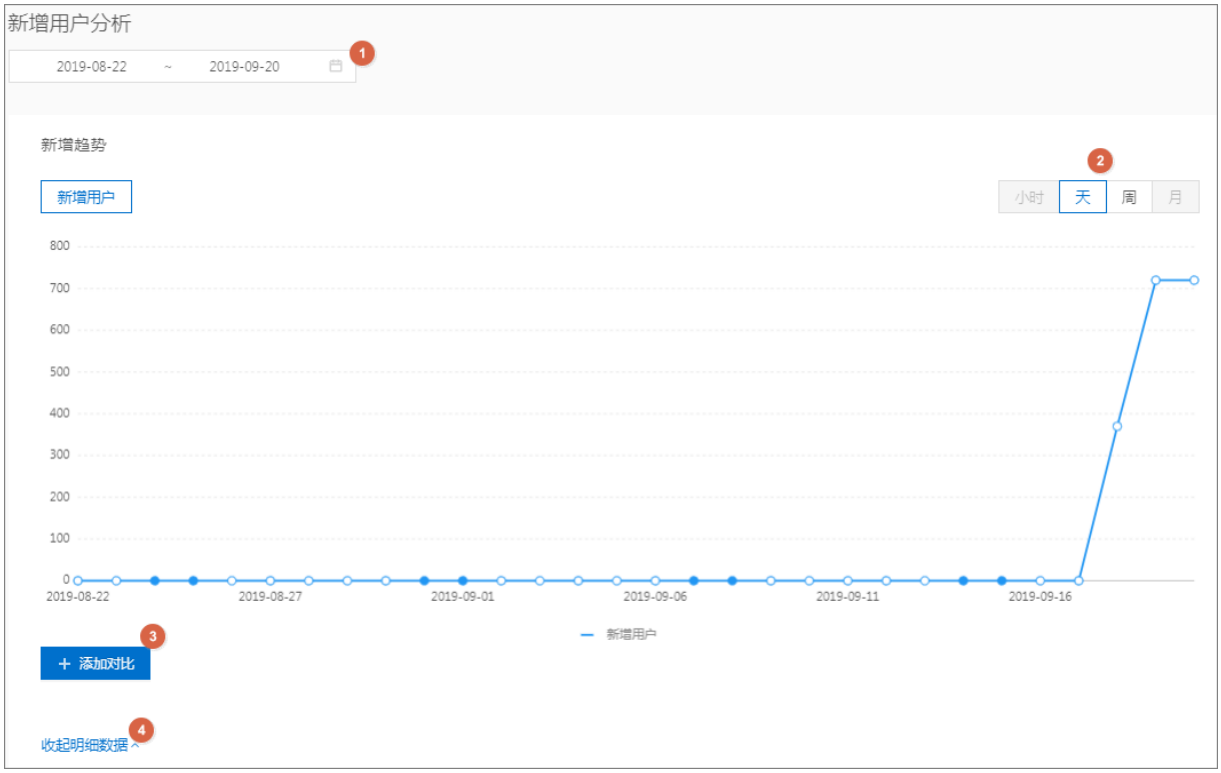
6.5.1. 新增用户

新增用户页面展示实时的新增用户数据和次日留存率，帮助您了解新增用户质量。

新增用户分析

统计口径：首次启动进入小程序的设备视为一个新增用户。

您可以筛选查看不同时间段的新增用户，并添加时间对比数据。



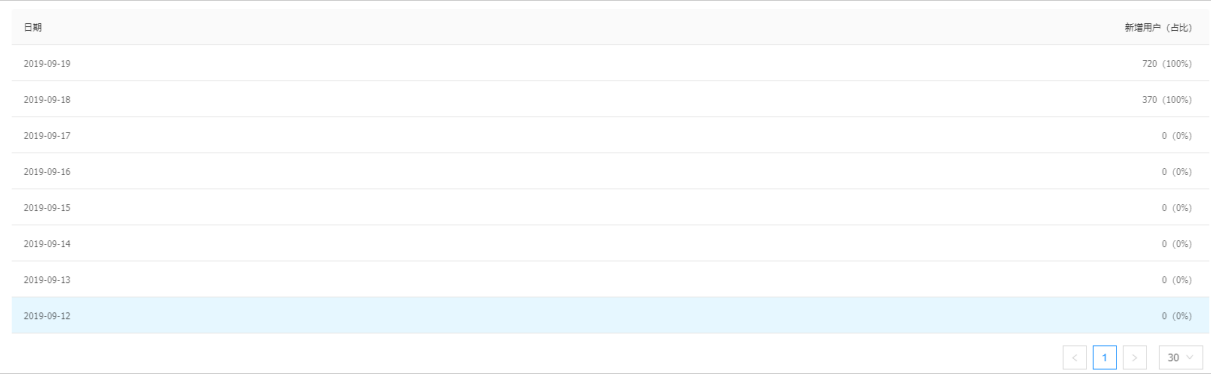
配置	说明
1. 时段选择	选择特定时间维度，目前支持选择最大时间筛选为732天。

配置	说明
2. 时间颗粒度选择	按照不同颗粒度查看数据： ● 当时间维度在0~7天时，可按小时查看。 小时数据最多展示7天并暂时不支持筛选。 ● 当时间维度大于8天时，可按周查看。 按周显示新增用户时，界面上用每周的周日来代表该周。 ● 当时间维度大于30天时，可按月查看。 按月显示新增用户时，界面上用每个月的第一日来代表该月。
3. 添加对比	支持渠道、时段对比。 选择**时间对比**可对所选时段的数据做上周同期以及上月同期数据对比。
4. 展开明细数据	单击此处可查看所选时间范围内的明细数据包括新增用户总数和新增用户占比。 新增用户占比指时段内新增用户占该时段内活跃用户的比例。

次日留存率

统计口径：该日的新增用户在次日仍启动小程序的用户占该日新增的比例。

您可以通过观测次日留存率了解新增用户质量。



日期	新增用户（占比）
2019-09-19	720（100%）
2019-09-18	370（100%）
2019-09-17	0（0%）
2019-09-16	0（0%）
2019-09-15	0（0%）
2019-09-14	0（0%）
2019-09-13	0（0%）
2019-09-12	0（0%）

6.5.2. 活跃用户

活跃用户页面展示活跃用户趋势、活跃用户构成等数据。

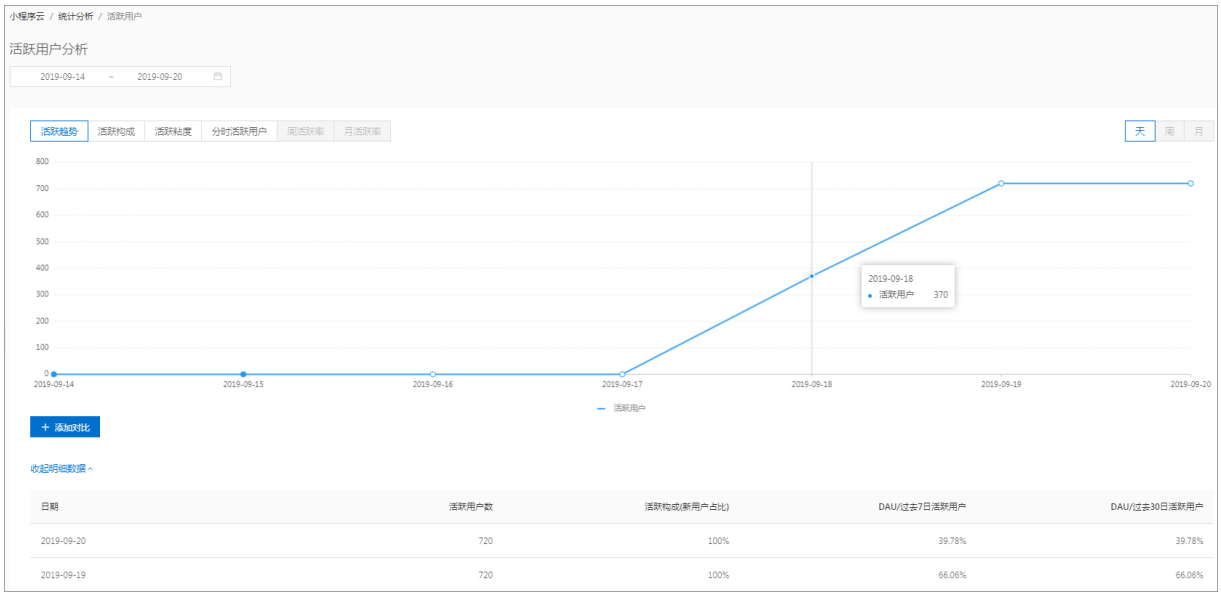
统计口径

所选时间段内，启动过小程序的用户（去重），启动过一次的用户即被视为活跃用户，包括新用户和老用户。

更新周期

- 日活跃用户实时更新。
- 周活跃用户每周日更新上周数据。
- 月活跃用户每月1日更新上月数据。

详细说明



配置	说明
1. 时段选择	选择特定时间维度，目前支持选择最大时间筛选为732天。
2. 指标选择	提供活跃用户、活跃用户构成、分时活跃用户、活跃粘度、用户周活跃率、用户月活跃率指标。您可以单击不同指标进行切换。 **说明** • 分时活跃用户：只有所选时间段小于或等于7天时可查看整点的活跃用户数据。 • 周活跃率：只有所选时间段大于或等于8天时可查看。 • 月活跃率：只有所选时间段大于30天时可查看。
3. 时间颗粒度选择	按照不同颗粒度查看数据： • 当时间维度在0~7天时，可按小时查看。 小时数据最多展示7天并暂时不支持筛选。 • 当时间维度大于8天时，可按周查看。 按周显示新增用户时，界面上用每周的周日来代表该周。 • 当时间维度大于30天时，可按月查看。 按月显示新增用户时，界面上用每个月的第一日来代表该月。
4. 添加对比	支持渠道、时段对比。 选择**时间对比**可对所选时段的数据做上周同期以及上月同期数据对比。

配置	说明
5. 展开明细数据	单击此处可查看所选时间范围内的明细数据。 活跃用户占比指该日（周/月）活跃用户占所查看时段内每日（周/月）活跃用户加和的比例。

常见问题

1. 为什么活跃用户概况中的过去7天活跃用户数少于活跃用户明细中，每天活跃用户的加和总数？

   7天活跃用户数是对这一周的活跃用户进行去重。而活跃用户明细中，是按天进行去重的。如果一个用户每天都使用小程序，那么每天加和的数量就要比7天活跃用户数多6个。

2. 我昨天夜里测试的小程序，但是为什么算在了今天的活跃中？

   我们按零点划分日期的。在零点之后使用，则会被算为第二天的活跃用户。

6.5.3. 启动次数

启动次数页面展示启动次数的数据实时更新。

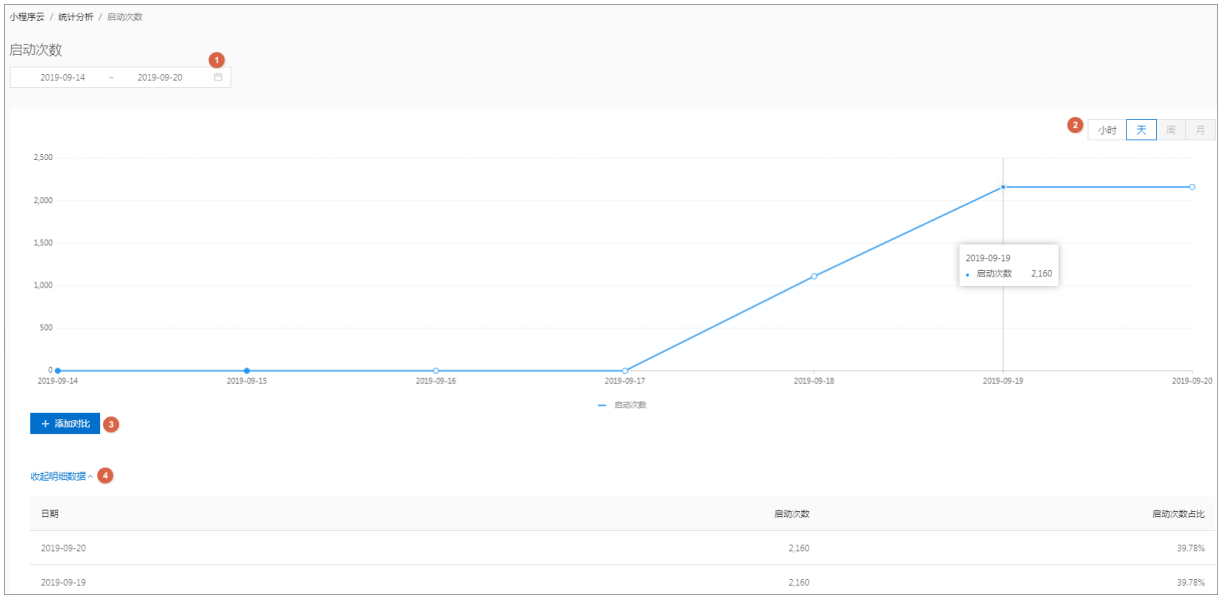
统计口径

打开小程序视为启动，完全退出或退至后台即视为启动结束。

更新周期

启动次数的数据实时更新。

详细说明



配置	说明
1. 时段选择	选择特定时间维度，目前支持选择最大时间筛选为732天。

配置	说明
2. 时间颗粒度选择	按照不同颗粒度查看数据： • 当时间维度在0~7天时，可按小时查看。 小时数据最多展示7天并暂时不支持筛选。 • 当时间维度大于8天时，可按周查看。 按周显示新增用户时，界面上用每周的周日来代表该周。 • 当时间维度大于30天时，可按月查看。 按月显示新增用户时，界面上用每个月的第一日来代表该月。
3. 添加对比	支持渠道、时段对比。 选择**时间对比**可对所选时段的数据做上周同期以及上月同期数据对比。
4. 展开明细数据	单击此处可查看所选时间范围内的明细数据。 启动次数占比指该日（周/月）启动次数占所查看时段内每日（周/月）启动次数加和的比例。

常见问题

1. 只要打开就算启动吗，没有时长的限制吗？

   没有，只要打开即算为一次启动。

2. 我可以设置需要使用一定的时长才算一次真正的启动码？

   不可以。

6.6. 留存分析

6.6.1. 留存用户

留存用户页面展示新用户留存和活跃用户留存数据。

更新周期

留存用户的数据每日更新一次。

详细说明

配置	说明
1. 时段选择	选择特定时间维度，目前支持选择最大时间筛选为732天。
2. 留存用户选择	• 新用户留存：某段时间内的新增用户，经过一段时间后，又继续使用小程序的被认作是留存用户。 • 活跃用户留存：某段时间内的活跃用户，经过一段时间后，又继续使用小程序的被认作是留存用户。
3. 时间颗粒度选择	按照不同颗粒度查看数据。选择周（月）时，展示的是该自然周（月）的新增用户，在之后的自然周（月）的留存情况。
4. 选择数据粒度	• 留存率：某段时间内的新增用户（活跃用户），经过一段时间后，又继续使用小程序的比例。 • 留存数：某段时间内的新增用户（活跃用户），经过一段时间后，又继续使用小程序的用户数。 例如，5月份新增用户200，这200人在6月份启动过小程序的有100人，7月份启动过小程序的有80人，8月份启动过小程序的有50人；则5月新增用户一个月后的留存率是50%，两个月后的留存率是40%，三个月后的留存率是25%。

常见问题

1. 留存用户列表里不同的颜色表示什么？

   留存率越高，标注的颜色越深，以此直观区分留存的变化情况。

6.7. 用户参与度

6.7.1. 使用时长

使用时长页面展示用户在任意1天内单次、单日使用时长的分布情况。

统计口径

- 单次使用时长是指一次启动内小程序在前台的持续时长。
- 日使用时长是指单设备在一天内使用时长的总和。

更新周期

本页面内的数据每日更新一次，无当日数据。

详细说明

配置	说明
1. 时段选择	选择特定日期查看。
2. 平均使用时长	所查看日期的平均单次使用时长（总时长/总启动次数）。
3. 时间对比	可对所选时段的数据做上周同期、上月同期或自选日期的数据对比。
4. 启动次数占比	该使用时长区间内的启动次数占总启动次数的比例。

6.7.2. 使用频率

使用频率页面展示用户在任意某天、某周、某月内日启动次数的分布情况。

统计口径

- 日启动次数展示的是在查看日期内的用户每日启动次数的分布。
- 周启动次数展示的是查看日期前一个自然周内的用户每周启动次数的分布。
- 月启动次数展示的是查看日期前一个自然月内的用户每月启动次数的分布。

更新周期

本页面内的数据每日更新一次，无当日数据。

详细说明

配置	说明
1. 时段选择	选择时间段，支持选择任意一天。
2. 平均启动次数	所有用户的平均每日启动次数，此Tab默认隐藏，点击后展开。

配置	说明
3. 明细数据	该使用频率区间内的用户数占总用户数的比例。
4. 时间对比	对任意天数据进行选择对比，支持多天对比。

6.8. 渠道

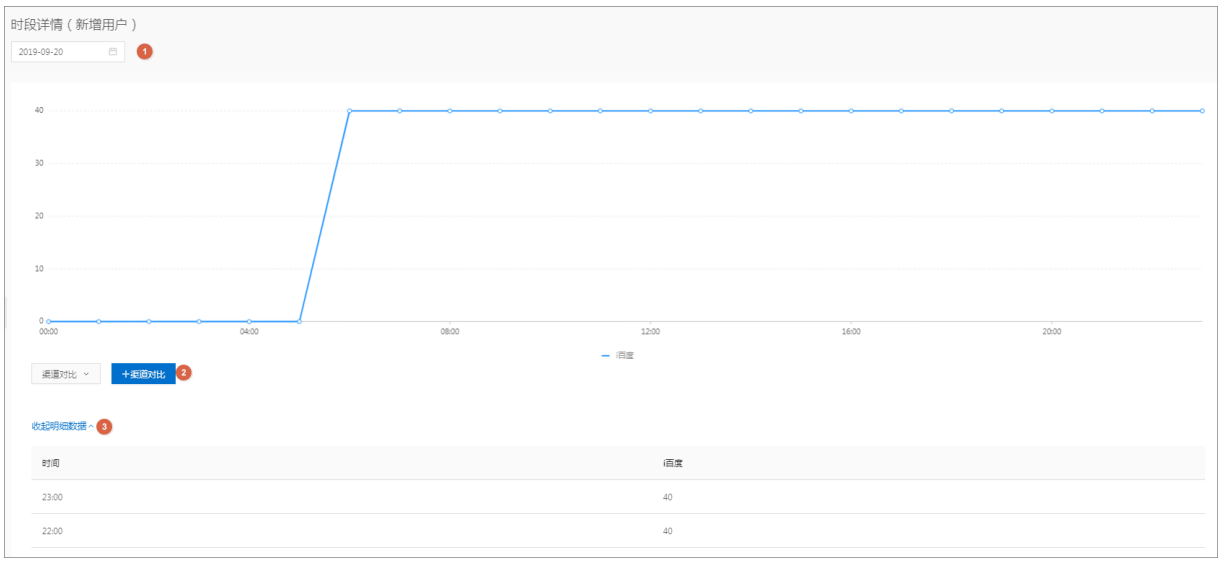
6.8.1. 时段详情

时段详情页面展示不同渠道在选定时间内的新增用户数据。

统计口径

首次启动进入小程序的设备视为一个新增用户。

详细说明



配置	说明
1. 时段选择	选择时间段，支持选择任意一天。
2. 渠道对比	对比任意所选渠道在特定时间内的新增用户数据。
3. 明细数据	各渠道的新增用户明细。

6.8.2. 渠道列表

渠道列表页面展示Top 10渠道的新增、活跃等数据，帮助您了解该渠道用户的粘着度、忠诚度，进而了解渠道推广的质量，为您调整渠道推广策略提供依据。

详细说明

配置	说明
1. 时段选择	选择特定时间维度，目前支持选择最大时间筛选为732天。
2. 指标选择	• 新用户：第一次启动小程序的用户（以设备为判断标准）。 • 活跃用户：启动过小程序的用户（去重），启动过一次的用户即视为活跃用户，包括新用户与老用户。 • 平均单次使用时长：单次使用时长的均值，即全部用户的日使用时长/总启动次数。 • 平均日使用时长：日使用时长的均值，即全部用户的日使用时长/总活跃用户。 • 新增用户次日留存率：该日的新增用户在次日仍启动小程序的用户占该日新增的比例。
3. 渠道对比	可选择其他渠道来进行数据对比。

7.扩展能力

7.1. 云调用概述

云调用是基于小程序Serverless的云函数来使用支付宝小程序开放接口的能力，可以方便的让开发者在小程序中直接调用支付宝的后端开放接口，不需要关注服务端的相关配置，极大的减小了接入的流程，进一步降低了支付宝小程序的开发门槛。

目前云调用涵盖了支付宝开放的基础能力、营销能力、会员能力、支付能力、安全能力、资金能力以及通用接口能力。各项能力涉及的API列表如下所示。

> **说明** 目前，仅支持在支付宝小程序中使用如下扩展能力。

基础能力

API	描述
cloud.base.oauth.getToken	获取授权访问令牌。
cloud.base.oauth.refreshToken	刷新授权访问令牌。
cloud.base.qrcode.create	创建小程序二维码。

营销能力

API	描述
cloud.marketing.templateMessge.send	发送模板消息。

会员能力

API	描述
cloud.member.identification.init	身份认证初始化。
cloud.member.identification.certify	生成认证链接。
cloud.member.identification.query	身份认证记录查询。

支付能力

API	描述
cloud.payment.common.create	创建交易。
cloud.payment.common.query	查询交易。
cloud.payment.common.refund	交易退款。

API	描述
cloud.payment.common.close	关闭交易。
cloud.payment.common.cancel	撤销交易。
cloud.payment.huabei.create	创建花呗分期交易。
cloud.payment.faceToFace.pay	当面付交易付款。

安全能力

API	描述
cloud.security.textRisk.detect	检测内容风险。

资金能力

API	描述
cloud.fund.transferAccount.transfer	资金转账到支付宝账户。
cloud.fund.transferAccount.query	资金转入查询。
cloud.fund.transferBankcard,transfer	资金转账到银行卡。
cloud.fund.transferBankcard.query	资金转出查询。
cloud.fund.b2cRedPacket.send	发放B2C现金红包。
cloud.fund.b2cRedPacket.query	红包转账查询。

通用接口能力

API	描述
cloud.util.generic.execute	通用接口。

7.2. 使用云调用

要使用云调用，您需要首先在控制台开通云调用功能。若您需要在小程序端直接使用云调用，您需要在小程序中安装云调用SDK并进行初始化。若您需要在云函数端使用云调用，无需安装SDK。详细的使用方式请参考支付宝开放平台的文档：云调用的使用。

> **说明** 目前，仅支持在支付宝小程序中使用扩展能力。

7.3. 公共数据结构

返回数据结构

下表列举了alipay-cloud-sdk的公共返回数据。

字段名	数据类型	说明
requestId	String	请求ID。
success	Boolean	请求是否成功。
result	Any	请求响应数据，结构与具体 API 有关。

result公共数据如下表所示。

字段名	数据类型	说明
code	String	响应码。
msg	String	响应主消息。
subCode	String	响应子码，一般是子系统的响应码。
subMsg	String	响应子消息，一般是子系统的返回消息

公共错误码

下表列举了公共错误码。

错误码	说明
SYS_INIT	SDK未初始化。 使用SDK之前需要调用 cloud.init 方法初始化。详情请参见使用云调用。
FEATURE_NOT_SUPPORT	API暂未支持。
INVALID_JSON_OBJECT	JSON解析失败。
NOT_FOUND_PARAM	缺少必要参数，请检查必填参数。
INVALID_PARAM	参数不合法，请检查参数类型等是否正确。

8.设置

8.1. 添加支付宝小程序

在使用小程序Serverless开发支付宝小程序前，您需要在控制台配置支付宝小程序信息。

前提条件

您已经在蚂蚁金服开放平台创建了小程序，并为支付宝小程序配置了公钥。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，单击**小程序Serverless > 设置**。
3. 单击**支付宝**页签，然后单击**添加**。
4. 输入您的私钥。

   您可以通过RSA密钥生成工具生成密钥，详情请参见生成支付宝RSA密钥。
5. 输入支付宝小程序配置的公钥。参考以下步骤查看公钥：
   i. 登录蚂蚁金服开放平台。
   ii. 在支付宝小程序菜单栏单击**设置**，然后单击**开发设置**。
   iii. 在**接口加签方式**区域，单击**查看**。

iv. 复制支付宝公钥信息，然后粘贴到阿里云支付宝小程序的配置中。

6. 单击**确定**完成添加。

8.2. 添加钉钉小程序

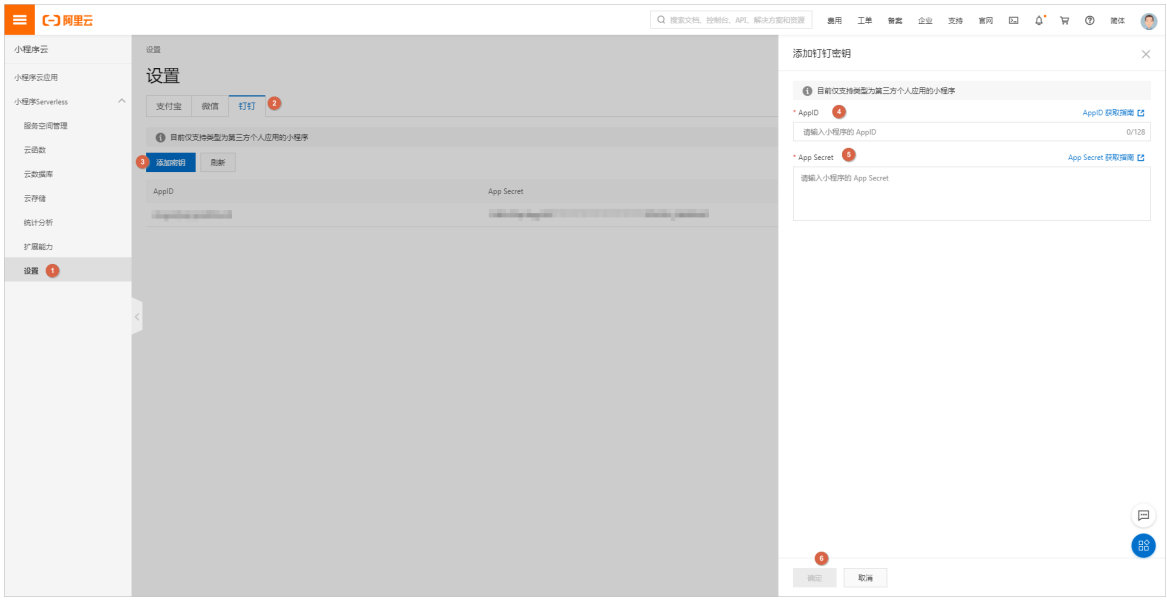
在使用阿里云小程序Serverless开发钉钉小程序前，您需要在控制台配置钉钉小程序信息。

前提条件

您已经在钉钉开放平台创建了第三方个人小程序。

操作步骤

1. 登录小程序云控制台。
2. 在左侧导航栏，单击**小程序Serverless > 设置**。
3. 单击**钉钉**页签，然后单击**添加密钥**。
4. 在**添加钉钉密钥**页面，输入以下信息，然后单击**确定**。
   - AppID：输入钉钉小程序的App ID。
   - App Secret：输入钉钉小程序的密钥。



参考以下步骤获取钉钉小程序的App ID和密钥：

i. 登录钉钉开放平台，并选择您的组织。

ii. 选择已创建的小程序，在**应用首页**页签，单击**应用信息**列的查看详情获取小程序的App ID及App Secret。

8.3. 添加微信小程序

在使用小程序Serverless开发微信小程序前，您需要在控制台配置微信小程序信息。

前提条件

已完成微信小程序注册并完成认证。详细信息请参见注册微信小程序。

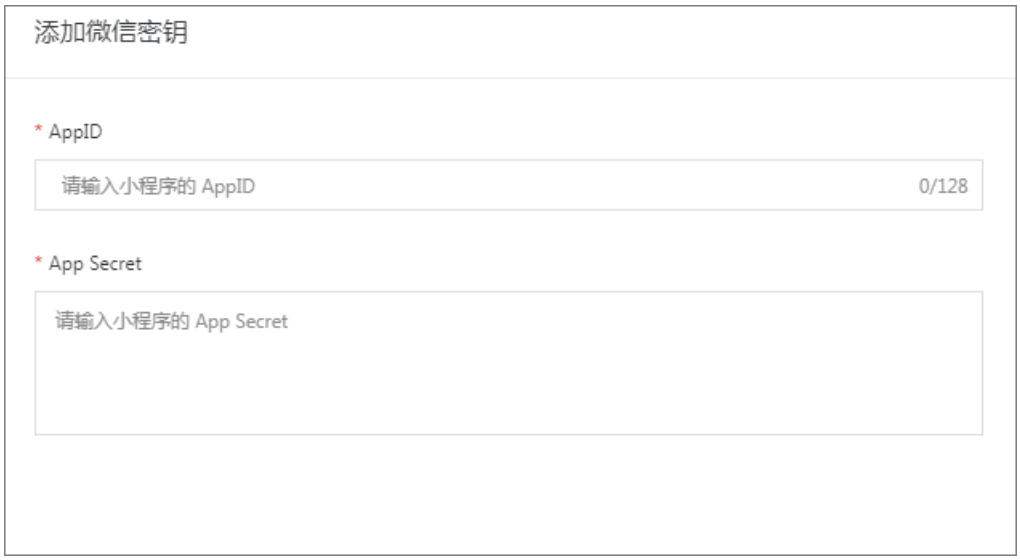
操作步骤

1. 登录小程序云控制台。

2. 在左侧导航栏，单击**小程序Serverless > 设置**。
3. 单击**微信**页签，然后单击**添加密钥**。
4. 在**添加微信密钥**页面，输入以下信息，然后单击**确定**。
   - **AppID**：输入微信小程序的App ID。

     登录微信公众平台，在**开发设置**页面查看App ID。

   - **App Secret**：输入微信小程序的密钥。



后续步骤

在微信小程序中使用SDK