ALIBABA CLOUD

Alibaba Cloud

函数计算

开发工具

文档版本：20200925

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.Aliyun Serverless VSCode Extension插件

Aliyun Serverless VSCode Extension是一款VSCode图形化开发调试函数计算以及操作函数计算资源的插件。本文介绍了如何通过该插件创建函数以及该插件的常见功能。

前提条件

如果您期望使用Aliyun Serverless VSCode Extension的所有功能，那么您需要确保系统中有以下组件：

- VSCode：可以在Visual Studio Code官网中下载安装。
- Docker：可以在aliyun/fun中根据教程安装配置Docker。

背景信息

Aliyun Serverless VSCode Extension是函数计算提供的VSCode插件，该插件结合了函数计算命令行工具Fun和函数计算SDK的功能，是基于VSCode的开发、调试、部署工具。通过该插件，您可以：

- 快速地在本地初始化项目、创建函数。
- 运行、调试本地函数，以及部署服务函数至云端。
- 拉取云端的服务函数列表、查看服务函数配置信息、调用云端函数。
- 获得模版文件的语法提示：自动补全、Schema校验、悬浮提示。

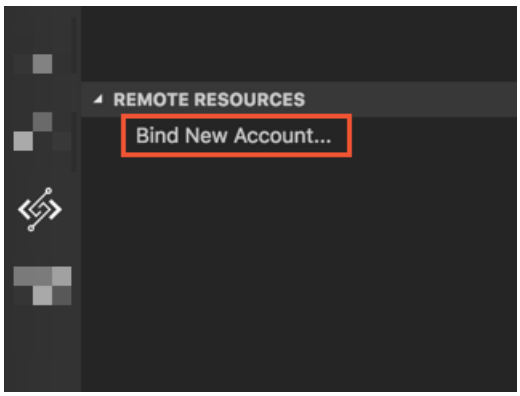
安装插件

1. 打开VSCode并进入插件市场。
2. 在插件市场中搜索Aliyun Serverless，查看详情并安装。
3. 重启VSCode，左侧导航栏中会展示已安装的Aliyun Serverless VSCode Extension插件图标。

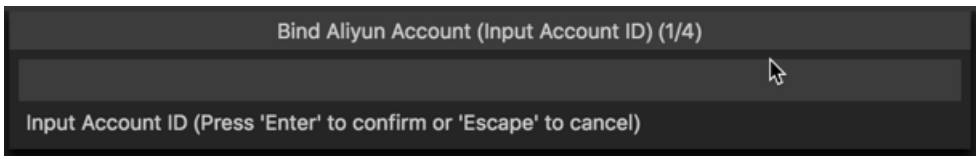
快速入门

1. 绑定阿里云账户。

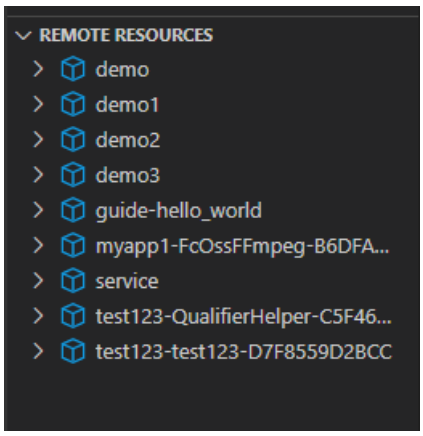
   i. 在左侧导航栏，单击Aliyun Serverless VSCode Extension图标，然后单击Bind New Account。



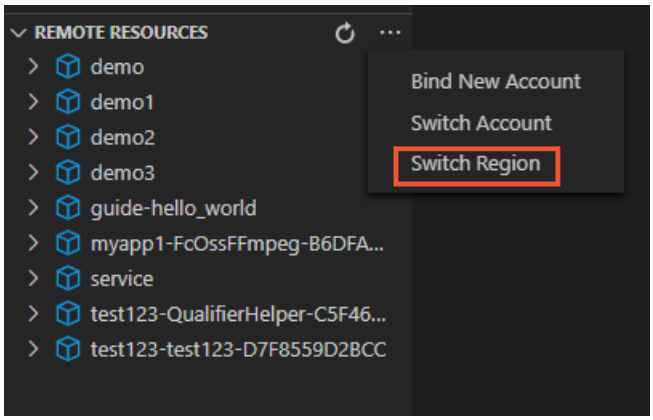
ii. 依次输入阿里云Account ID、阿里云AccessKeyId、阿里云AccessKeySecret、自定义的账户别名（即账户本地名称）。
您可以在账号管理中获取账号的Account ID，用户信息管理中获取AccessKeyID和SecretAccessKey。



绑定完成后，可以看到所绑定的阿里云账户的云端服务与函数列表。

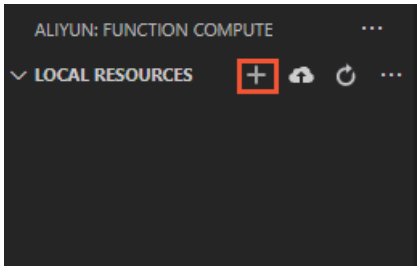


您还可以在REMOTE RESOURCES面板中，单击右上角的更多信息图标，在下拉菜单中，选择Switch Region来查看不同地域的服务与函数。

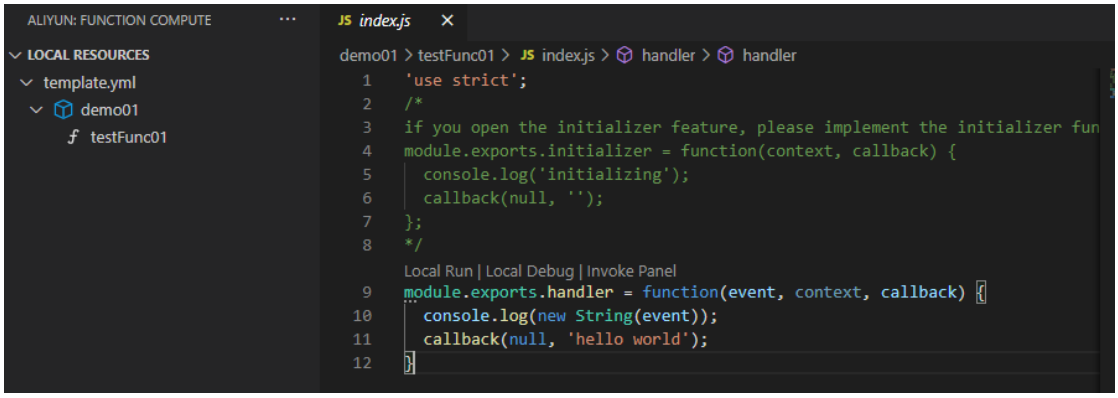


2. 创建函数。

i. 通过VSCode，打开一个空的目录文件。单击LOCAL RESOURCES中的创建函数图标，可以在本地初始化一个函数计算项目。



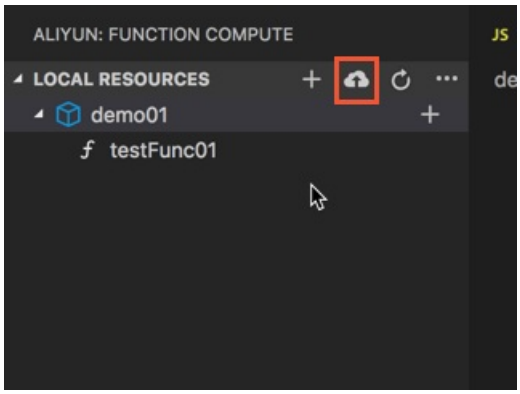
ii. 按照导航依次输入或选择服务名称、函数名称、函数运行环境、函数类型。填写完毕后，插件会自动创建函数并在LOCAL RESOURCES面板中会展示新建的本地服务与函数。



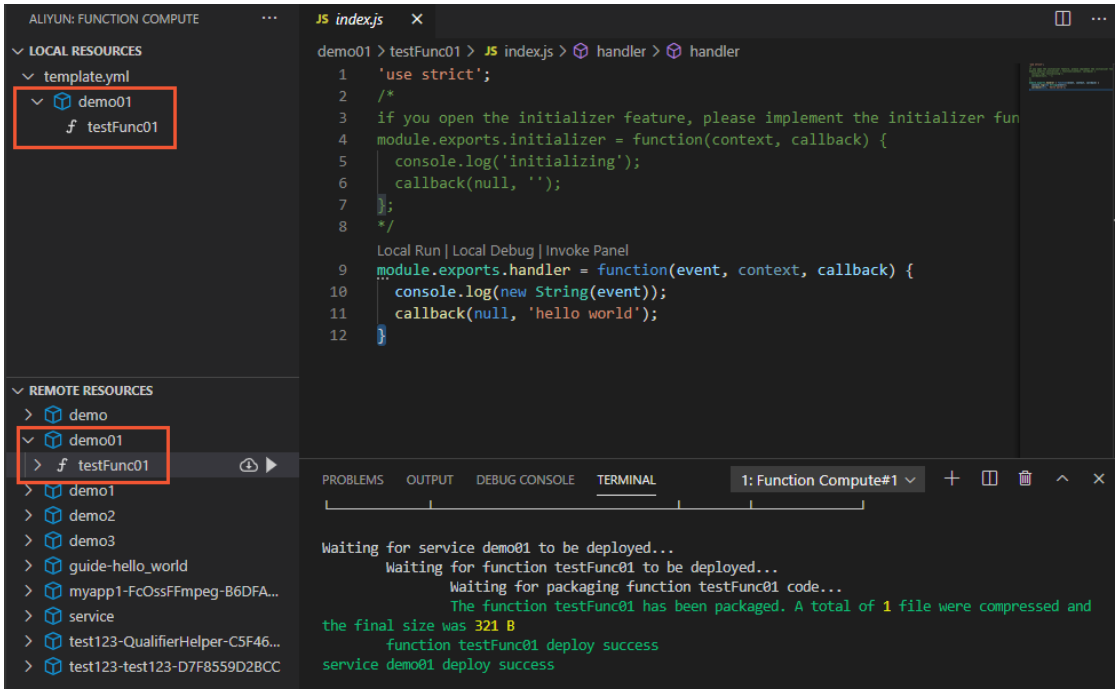
您也可以直接单击LOCAL RESOURCES中服务名右侧的创建函数图标，来为该服务创建函数。按照导航依次输入或选择函数名称、函数运行时、函数类型即可。

3. 部署服务以及函数。

i. 单击LOCAL RESOURCES面板中的部署图标，可以将本地的服务与函数部署到云端。

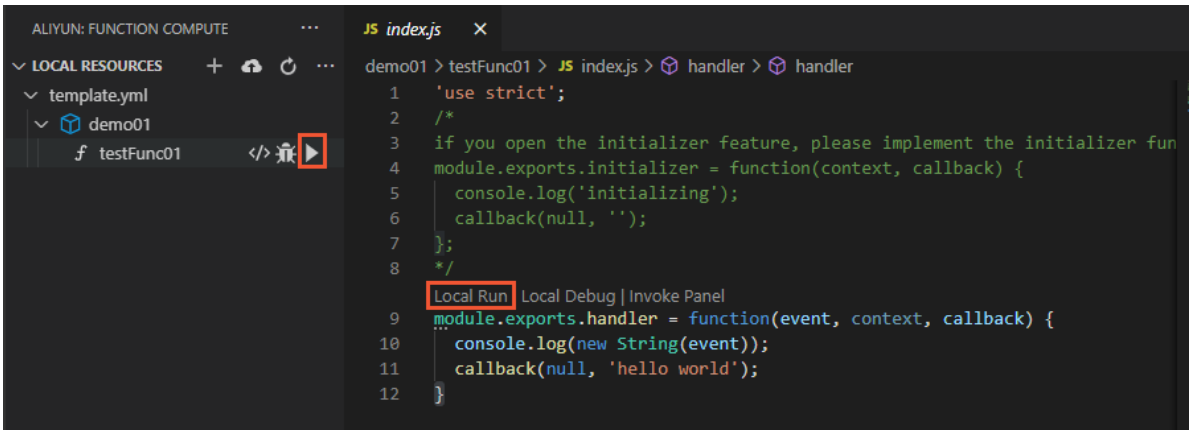


部署完成后，单击REMOTE RESOURCES面板中的刷新图标，可以查看部署到云端的服务与函数。

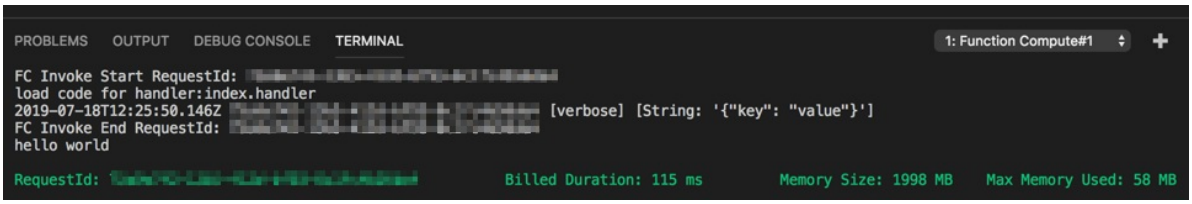


其余功能介绍

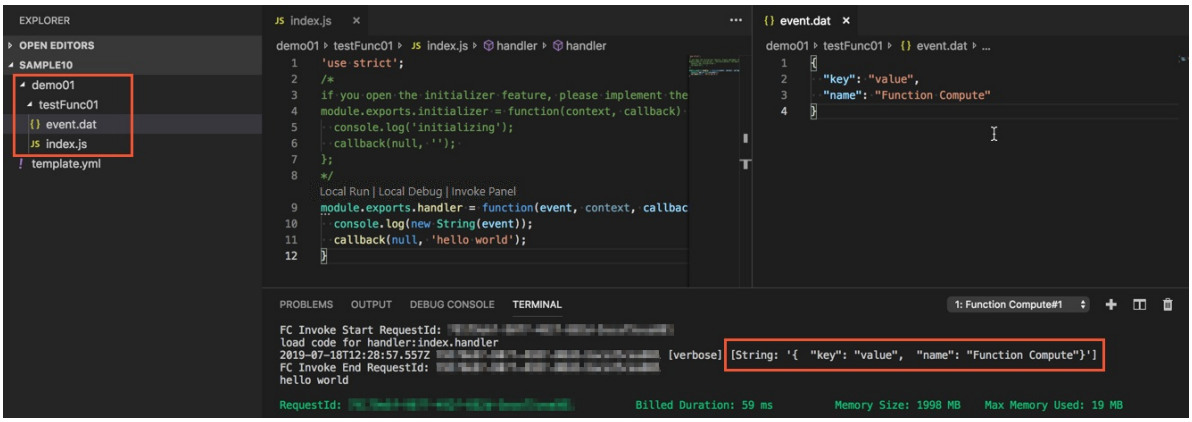
- 本地调用函数
  在LOCAL RESOURCES面板中，单击函数名称右侧的执行图标或*Handler*文件中的执行链接，可以在本地调用该函数。



函数的日志以及结果会输出在TERMINAL中。



插件会为您在函数入口文件同目录下创建*event.dat*文件，您可以通过修改该文件设置每次调用函数时触发的事件信息。

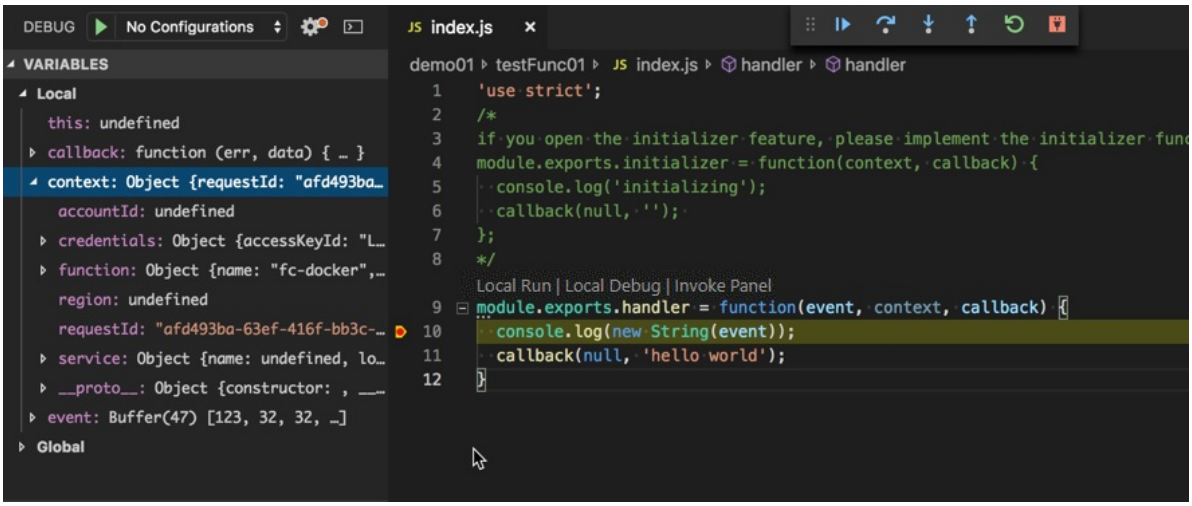


- 本地调试函数

> **注意**
>
> - 若您想要调试Python 2.7或Python 3 runtime的函数，需要事先在插件安装Python插件。
> - 若您想调试PHP runtime的函数，需要事先在插件安装PHP Debug插件。

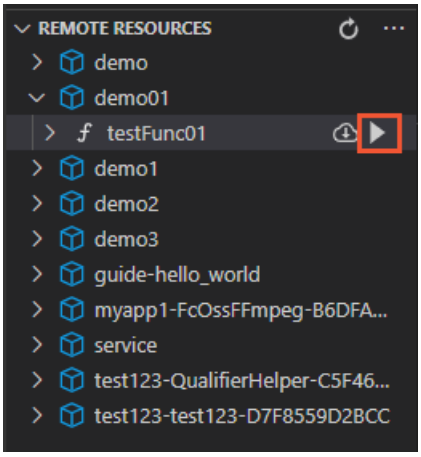
在LOCAL RESOURCES面板中，单击函数名称右侧的调试图标或*Handler*文件中的调试链接，可以在本地调试该函数。

在代码文件中插入断点，启动调试后即可看到调试信息。

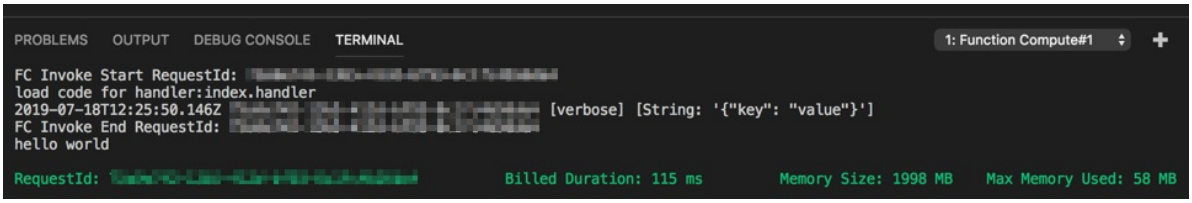


插件会为您在函数入口文件同目录下创建*event.dat*文件，您可以通过修改该文件设置每次调试函数时触发的事件信息。

- 执行云端函数
  单击**REMOTE RESOURCES**面板中函数右侧的执行图标，可以执行云端函数。

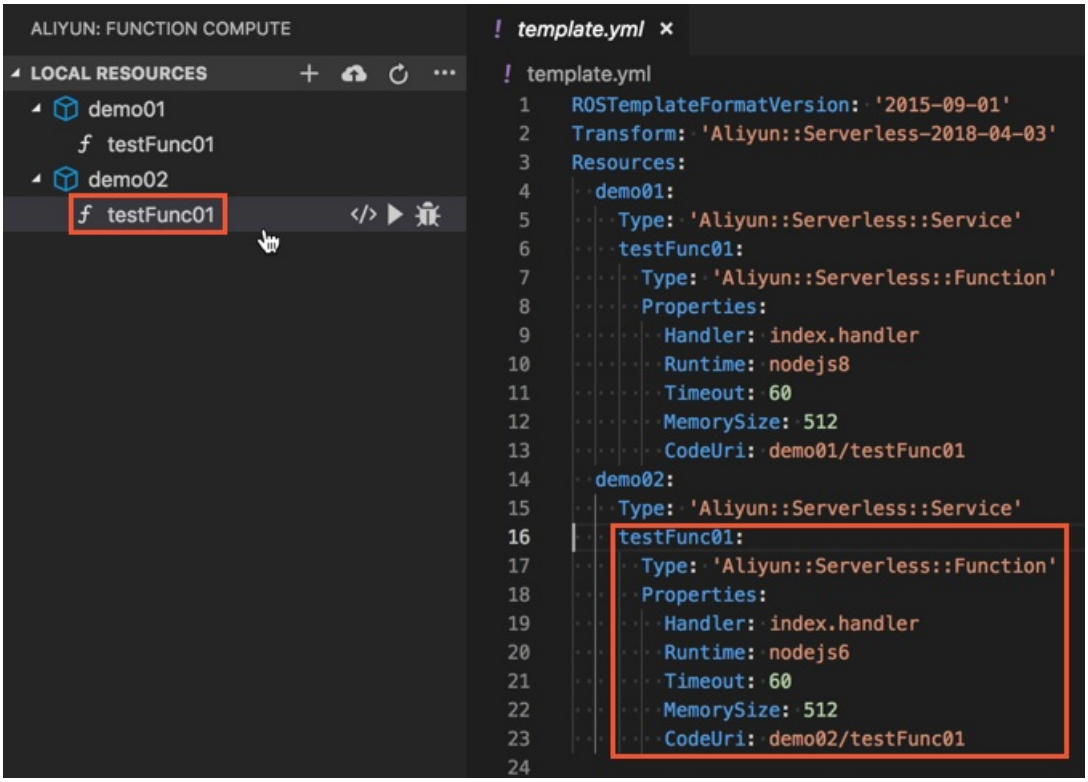


函数的日志以及结果会输出在**TERMINAL**中。



插件会为您在项目根目录下创建*event.dat*文件，您可以通过修改该文件设置每次调用云端函数时触发的事件信息。

- 跳转到模版文件定义
  函数计算Fun工具通过YAML格式的模板文件来描述Serverless应用。通过Aliyun Serverless VSCode Extension创建函数时，会使用默认值自动填充模版文件。若您想修改本地服务或函数的配置，可以通过单击**LOCAL RESOURCES**面板中的服务或函数名，跳转到模版文件中的相关描述，所选择资源在模板文件中的相关描述块会高亮并逐渐褪去。



- 模版文件提示
  - 自动补全
    支持模版文件*template.yml*内所有资源配置属性的自动补全。自动补全会依据缩进层级给出精准的提示选项。
  - Schema校验
    支持模版文件*template.yml*内所有资源配置信息的校验。在*template.yml*中会检测资源的配置信息是否符合规格说明。
  - 悬浮提示
    提供模版文件*template.yml*内所有资源配置的上下文帮助。在*template.yml*中，将鼠标悬浮在相关资源的键名上，会出现关于该键下可配置字段的悬浮信息展示（字段名、字段类型）。

反馈

如果您在使用中遇到问题，欢迎扫描以下二维码加入函数计算官方客户群或在github中反馈。

2.fcli

2.1. 初次使用fcli

fcli是阿里云函数计算的命令行工具，帮助您便捷地管理函数计算中的资源。若您是初次使用fcli，您需要下载并配置fcli。

下载fcli

下载fcli安装包至本地，然后解压安装包。

配置fcli

如果您是第一次使用，需要配置fcli。函数计算提供了以下几种配置方式：

使用 `fcli shell` 命令配置。

1. 在fcli可执行文件所在的文件夹下，执行 `fcli shell` 。
2. 根据界面提示，依次输入阿里云主账号的账号ID、AccessKey ID和AccessKey Secret，然后选择地域。
   您可以在账号管理中获取账号的Account ID，在用户信息管理中获取AccessKeyID和AccessKeySecret。

   ```
   ? Alibaba Cloud Account ID <your account ID>
   ? Alibaba Cloud Access Key ID <your AccessKey ID>
   ? Alibaba Cloud Access Key Secret <your Accesskey Secret>
   ? Default region name cn-shanghai
   Store the configuration in: .fcli
   Welcome to the function compute world. Have fun!
   >>>
   ```

   配置完成后，在fcli可执行文件所在的文件夹下生成*config.yaml*文件。

使用 `fcli config` 命令配置。

1. 在fcli可执行文件所在文件夹下，执行 `fcli config` 。
2. 根据界面提示，依次输入阿里云主账号的账号ID、AccessKey ID和AccessKey Secret，然后选择地域。
   您可以在账号管理中获取账号的Account ID，在用户信息管理中获取AccessKeyID和AccessKeySecret。

   ```
   ? Alibaba Cloud Account ID <your account ID>
   ? Alibaba Cloud Access Key ID <your AccessKey ID>
   ? Alibaba Cloud Access Key Secret <your Accesskey Secret>
   ? Default region name cn-shanghai
   Store the configuration in: .fcli
   ```

   配置完成后，在fcli可执行文件所在的文件夹下生成*config.yaml*文件。

直接配置yaml文件。

1. 打开*~/.fcli/config.yaml*文件。
2. 将对应的参数配置为您自己的值，并保存配置。

```
endpoint: https://<account ID>.<region ID>.fc.aliyuncs.com:8080
api_version: 2016-08-15
access_key_id: <your AccessKeyID>
access_key_secret: <your AccessKeySecret>
security_token: ""
user_agent: fcli-0.1
debug: false
timeout: 60
sls_endpoint: <region ID>.log.aliyuncs.com
```

> 说明 account ID、AccessKeyID、AccessKeySecret的所属账号必须为主账号。您可以在账号管理中获取账号的Account ID，在用户信息管理中获取AccessKeyID和AccessKeySecret。

方法一

方法二

方法三

fcli命令集

函数计算为您提供了fcli的操作命令集以便开发者需要时查阅。开发者也可以通过在可执行文件所在文件夹下执行 `fcli --help` 查看命令的详细信息。

- 通用操作
  - 目录跳转（cd）
  - 列出当前目录下的文件（ls）
  - 查看当前所在目录（pwd）
  - 查看资源的详细信息（info）
  - 配置fcli（config）
  - 删除资源（rm）
  - 沙盒环境（sbox）
  - 帮助（help）
  - 清屏（clear）
  - 退出fcli（exit）
- 服务相关命令
  - 创建服务（mks）
  - 更新服务（ups）
- 函数相关命令
  - 创建函数（mkf）

- 更新函数（upf）
- 执行函数（invk）

- 触发器相关命令
  - 创建触发器（mkt）
  - 更新触发器（upt）

- 日志相关命令
  - 创建日志项目和日志库（mkl）
  - 查看日志（logs）

- 角色授权相关命令
  - 创建RAM权限策略（mkrp）
  - 创建角色（mksr）
  - 赋予角色RAM策略（attach）
  - 为角色解除权限策略（detach）
  - 为函数计算服务授权（grant）

2.2. 通用操作

本文介绍fcli的通用操作命令。

前提条件

在可执行文件所在文件夹下执行 `fcli shell` ，进入交互模式。

目录跳转（cd）

cd命令用于切换函数计算的实体的层次关系，而不是切换本地目录。

```
>>> cd serviceName //进入相应serviceName服务的目录。
>>> ls //在service目录下ls，列出当前serviceName服务下所有函数的名称。
>>> cd functionName //进入相应functionName函数的目录。
>>> ls //在function目录下ls，列出当前functionName函数下所有触发器的名称。
```

列出当前目录下的文件（ls）

- `-l int32` 或 `--limit int32` ：指定列出资源的最大数目（默认最大数目为100）。
- `-t string` 或 `--next-token string` ：列出从NextToken开始的资源。
- `-p string` 或 `--prefix string` ：列出指定前缀的资源。
- `-k string` 或 `--start-key string` ：列出从此资源开始的资源。

```
>>> cd myService
>>> ls
aFunction
bFunction
cFunction
cFuncion2
dFunction
eFunction
>>> ls -l 4 //指定列出文件的最大数目为4。
aFunction
bFunction
cFunction
cFuncion2
NextToken: dFunction
>>> ls -t dFunction //列出从NextToken（dFunction）开始的文件。
dFunction
eFunction
>>> ls -p c //列出前缀为c的文件。
cFunction
cFunction2
>>> ls -k dFunction //列出名字从dFunction开始的文件。
dFunction
eFunction
```

查看当前所在目录（pwd）

pwd命令用于查看当前所在目录。

查看资源的详细信息（info）

info命令用于查看函数计算资源的详细信息，其参数可以是服务名称、函数名称以及触发器名称等。

```
>>> info <your serviceName> //查看此服务的详细信息。
>>> info <your functionName> //查看此函数的详细信息。
>>> info <your triggerName> //查看此触发器的详细信息。
```

配置fcli（config）

config命令用于更改*config.yaml*中的配置信息。

```
>>> config --access-key-id 12345678 //修改access-key-id为12345678。
```

删除资源（rm）

- `rm`：删除资源 。

> **说明** 使用该命令前需要保证目标函数下没有触发器，目标服务下没有函数。

- `-f` 或 `--forced`：删除资源，且删除前无需确认。

```
>>> rm myFunction
Do you want to remove the resource /fc/myService/myFunction [y/n]: //删除资源前需要确认。
>>> rm -f myFunction //删除资源前无需确认，直接删除。
```

沙盒环境（sbox）

fcli为您提供了一个本地的沙盒环境，和函数计算服务中的函数运行环境保持一致。在沙盒环境中，您可以方便地安装第三方依赖库，进行本地调试等操作。

- `-d string` 或 `--code-dir string`：指定代码所在目录，他将被挂载到沙盒环境的*/code*位置。
- `-t string` 或 `--runtime string`：指定运行环境。

```
>>> sbox -d code -t nodejs6 //在code目录下安装依赖，语言为Node.js6。
Entering the container. Your code is in the /code direcotry.
root@df9fc****:/code# npm init -f //Node.js6需要生成package.json。
npm info it worked if it ends with ok
npm info using npm@3.10.10
npm info using node@v6.10.3
npm WARN using --force I sure hope you know what you are doing.
Wrote to /code/package.json:
{
"name": "code",
"version": "1.0.0",
"description": "",
"main": "index.js",
"dependencies": {
 "jimp": "^0.2.28"
},
"devDependencies": {},
"scripts": {
 "test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC"
}
npm info init written successfully
npm info ok
root@df9fc****:/code# npm install jimp //安装jimp依赖库。
npm info it worked if it ends with ok
...
npm info lifecycle jimp@0.2.28~postinstall: jimp@0.2.28
code@1.0.0 /code
-- jimp@0.2.28
npm WARN code@1.0.0 No description
npm WARN code@1.0.0 No repository field.
npm info ok
root@df9fc****:/code# exit //退出sbox环境。
```

帮助（help）

help命令用于列出帮助信息。

- `help`：可以列出所有的命令。
- `[command] --help`：可以列出此操作下的所有参数的介绍。
  例如 `ls --help`。

清屏（clear）
clear命令用于清屏。

退出fcli（exit）
exit命令用于退出fcli工具。

2.3. 服务相关命令
本文介绍fcli中服务相关的命令。

前提条件
在可执行文件所在文件夹下执行 `fcli shell`，进入交互模式。

创建服务（mks）
- `-d string` 或 `--description string`：服务中的描述信息。
- `-p string` 或 `--log-project string`：指定服务对应的日志项目Project。
- `-l string` 或 `--log-store string`：指定服务对应的日志库Logstore。
- `-r string` 或 `--role string`：指定服务对应的角色。

```
>>> mks myService //新建一个服务，不带任何高级配置内容。
>>> mks myService -d 'my description' -p my-log-project -l my-log-store -r acs:ram::myID:role/myRoleName //新建服务，其中描述信息为my description，日志项目为my-log-project，日志存储my-log-store，角色为myRoleName。
```

更新服务（ups）
该命令的参数与mks命令的参数相同。

- `-d string` 或 `--description string`：服务中的描述信息。
- `-p string` 或 `--log-project string`：指定服务对应的日志项目Project。
- `-l string` 或 `--log-store string`：指定服务对应的日志库Logstore。
- `-r string` 或 `--role string`：指定服务对应的角色。
-

```
>>> ups myService -d 'my description' -p my-log-project -l my-log-store -r acs:ram::myID:role/myRoleName //更新服务，其中描述信息为my description，日志项目为my-log-project，日志存储my-log-store，角色为myRoleName。
```

更多信息
- 列出此账户下的所有服务

- 查看服务具体信息
- 删除服务

2.4. 函数相关命令

本文介绍fcli中函数相关的命令。

前提条件

在可执行文件所在文件夹下执行 `fcli shell` ，进入交互模式。

创建函数（mkf）

- `-b string` 或 `--code-bucket string` ：指定代码所在的OSS Bucket。
- `-o string` 或 `--code-object string` ：指定代码所在的Bucket中的Object Key。
- `-d string` 或 `--code-dir string` ：指定代码所在的目录。
- `-f string` 或 `--code-file string` ：指定压缩的代码文件。
- `-h string` 或 `--handler string` ：设置函数handler，handler的格式为“文件名.函数名”。例如hello_world.handler指定了函数的调用入口为*hello_world.js*文件中的handler函数。
- `-e int32` 或 `--initializationTimeout int32` ：设置初始化函数超时时间（默认30s）。
- `-i string` 或 `--initializer string` ：设置初始化函数。
- `-m int32` 或 `--memory int32` ：设置函数执行的内存大小。
- `-t string` 或 `--runtime string` ：指定运行环境。
- `--timeout int32` ：设置函数超时时间（默认30s）。

```
// 在相应service目录下
>>> mkf myFunction -t nodejs6 -h myFunction.handler -b ossBucketName -o objectKey //代码存储在OSS上，-t指定Runtime为Node.js6，-h指定函数入口，-b指定代码所在的OSS Bucket，-o指定了代码在Bucket中的Object Key。
>>> mkf myFunction -t nodejs6 -h myFunction.handler -d codeDir/myFunction -m 512 //代码存储在本地，-d指定了代码所在目录，-m设置函数执行的内存大小。
>>> mkf myFunction -h myFunction.handler -f code.zip -t nodejs6 //代码在本地的code.zip中
```

更新函数（upf）

- `-b string` 或 `--code-bucket string` ：指定代码所在的OSS Bucket。
- `-o string` 或 `--code-object string` ：指定代码所在的Bucket中的Object Key。
- `-d string` 或 `--code-dir string` ：指定代码所在的目录。
- `-f string` 或 `--code-file string` ：指定压缩的代码文件。
- `-h string` 或 `--handler string` ：设置函数handler，handler的格式为“文件名.函数名”。例如hello_world.handler指定了函数的调用入口为*hello_world.js*文件中的handler函数。
- `-m int32` 或 `--memory int32` ：设置函数执行的内存大小。

- `-t string` 或 `--runtime string`：指定运行环境。

```
// 在相应service目录下
>>> upf myFunction -t nodejs6 -h myFunction.handler -b ossBucketName -o objectKey //代码存储在OSS上，-t指定Runtime为Node.js6，-b指定代码所在的OSS Bucket，-o指定了代码在Bucket中的Object Key。
>>> upf myFunction -t nodejs6 -h myFunction.handler -d codeDir/myFunction -m 512 //代码存储在本地，-d指定了代码所在目录，-m设置函数执行的内存大小。
>>> upf myFunction -h myFunction.handler -f code.zip -t nodejs6 //代码在本地的code.zip中。
>>> upf myFunction -t nodejs6 -i myFunction.newInitializer -e 30  -b ossBucketName -o objectKey // 将initializer从myFunction.initializer更新至myFunction.newInitializer。代码存储在OSS上且包含initializer函数，-t指定Runtime，-i指定initializer入口，-e指定initializer超时时间，-b指定代码所在的OSS Bucket，-o指定了代码在Bucket中的Object Key。
>>> upf myFunction -t nodejs6  -i  "" -b ossBucketName -o objectKey // 将initializer从myFunction.newInitializer更新至空，即关闭函数initializer功能。
>>> upf myFunction -t nodejs6 -i  myFunction.newInitializer -e 30 -b ossBucketName -o objectKey // 将initializer从空更新至myFunction.newInitializer。
```

执行函数（invk）

- `-e` 或 `--encode`：对函数的返回值进行base64编码。
- `-f string` 或 `--event-file string`：从文件中读取触发事件内容。
- `-s string` 或 `--event-str string`：从字符串中读取触发事件内容。
- `-o string` 或 `--output-file string`：将返回结果写入文件的文件名称。
- `-t string` 或 `--type string`：触发类型，取值：
  - Sync：同步触发（默认值）
  - Async：异步触发

```
>>> invk myFunction //如果不需要输入参数，不需要触发事件的话，则直接调用。
>>> invk myFunction -e //对函数的返回值进行base64编码。
>>> invk myFunction -s 'hello,world'//从字符串中读取触发事件内容。
>>> invk myFunction -f event.json //从文件中读取触发事件内容。
>>> invk myFunction -o code/result.txt //将返回结果写入result.txt文件中。
>>> invk myFunction -t Async //设置触发类型为异步触发。
```

更多信息

- 列出指定服务下的所有函数
- 查看函数具体信息
- 删除函数

2.5. 触发器相关命令

本文介绍fcli中触发器相关的命令。

前提条件

在可执行文件所在文件夹下执行 `fcli shell` ，进入交互模式。

创建触发器（mkt）

- `-r string` 或 `--invocation-role string` ：设置触发角色。
- `-s string` 或 `--source-arn string` ：事件源的资源符号，例如acs:oss:cn-shanghai:12345678:myBucketName。
- `-c string` 或 `--trigger-config string` ：设置触发器配置文件。
- `-t string` 或 `--type string` ：触发器类型，默认值为oss。
- 创建OSS触发器

```
>>> mkt myFunction/myFunctionTrigger -t oss -r acs:ram::12345678:role/AliyunOSSEventNotificatio
nRole -s acs:oss:cn-shanghai:12345678:myOssBucket -c code/ossTrigger.yaml
//其中yaml的文件内容如下triggerConfig。
  events:
    - oss:ObjectCreated:PutObject
    - oss:ObjectRemoved:DeleteObject
  filter:
    key:
      prefix: myPrefix
      suffix: mySuffix
```

- 创建HTTP触发器

```
>>> mkt myFunction/myFunctionTrigger -t http  -c code/httpTrigger.yaml
//其中yaml的文件内容如下triggerConfig:
 authType: anonymous
 methods:
  - GET
```

命令

示例

更新触发器（upt）

- `-r string` 或 `--invocation-role string` ：设置触发角色。
- `-s string` 或 `--source-arn string` ：事件源的资源符号，例如acs:oss:cn-shanghai:12345678:myBucketName。
- `-c string` 或 `--trigger-config string` ：设置触发器配置文件。
- `-t string` 或 `--type string` ：触发器类型，默认值为oss。

```
>>> upt myFunction/myFunctionTrigger -t oss -r acs:ram::account_id:role/AliyunOSSEventNotification
Role -s acs:oss:cn-region:account_id:bucketName -c code/trigger.yaml
```

更多信息

- 列出当前目录下的文件（ls）
- 查看资源的详细信息（info）
- 删除资源（rm）

2.6. 日志相关命令

本文介绍fcli中的日志相关命令。

前提条件

在可执行文件所在文件夹下执行 `fcli shell` ，进入交互模式。

创建日志项目和日志库（mkl）

mkl命令用于创建函数服务对应的日志项目Project和日志库Logstore。

- `-p string` ：创建日志项目（Log Project）。
- `-s string` ：创建日志库（Log Store）。

```
>>> mkl -p my-log-project -s my-log-store myService
// 为服务新建日志项目和日志库。
// Log Project的名称全局唯一，如果名称已被占用，那么会创建失败。
```

查看日志（logs）

logs命令用于查看日志相关信息。

- `-c int` 或 `--count int` ：设置返回日志数目的最大行数（默认是1000行）。
- `-d int` 或 `--duration int` ：返回从这段时间之前一直到现在的函数日志，单位秒，默认86400秒（24小时）。
- `-e string` 或 `--end string` ：设置查看日志的截止时间，格式为UTC RFC3339，例如2017-01-01T01:02:03Z。
- `-s string` 或 `--start string` ：设置查看日志的起始时间，格式为UTC RFC3339，例如2017-01-01T01:02:03Z
- `-t` 或 `--tail` ：设置从倒数第i行开始打印日志。

```
// 在相应service目录下。
logs myFunction //默认打印一天内的前1000行日志。
logs -d 60 -c 5000 myFunction //打印一分钟内执行的日志，最多打印5000条。
logs -t -c 100 myFunction // 打印倒数100行日志。
logs -s 2018-01-22T18:00:00Z -e 2018-01-22T19:00:00Z myFunction //打印从2018-01-22T18:00:00Z到2018-01-22T19:00:00Z的函数日志信息。
```

2.7. 角色授权相关命令

本文介绍fcli中的角色授权相关命令。

前提条件

在可执行文件所在文件夹下执行 `fcli shell` ，进入交互模式。

创建RAM权限策略（mkrp）

mkrp命令用于创建RAM权限策略。

- `-a string` 或 `--action string` ：设置策略的操作名称。
- `-r string` 或 `--resource string` ：设置策略的操作对象。

> 说明 权限策略的详细内容请参见Policy语法结构。

创建角色（mksr）

mksr命令用于创建角色，函数计算可以使用此角色访问云资源。

```
mksr roleName
```

赋予角色RAM策略（attach）

attach命令用于将RAM策略赋予指定角色。

- `-p string` 或 `--policy string` ：指定RAM策略。
- `-r string` 或 `--role string` ：指定RAM角色。

```
attach -p /ram/policies/myPolicy -r /ram/roles/myRole //将myPolicy策略赋予myRole角色。
```

为角色解除权限策略（detach）

detach命令用于为指定角色解除指定策略。

- `-a string` 或 `--action string` ：设置策略的操作名称。
- `-r string` 或 `--resource string` ：设置策略的操作对象。
-

```
detach -p /ram/policies/myPolicy -r /ram/roles/myRole //为myRole角色解除myPolicy授权。
```

为函数计算服务授权（grant）

grant命令用于为函数计算服务授予指定权限。

```
grant myService
Please input the role name: myRole
Please input the policy name: myPolicy
Permission grant scenarios:
 1. Allow FC write function logs to your log store.
 2. Allow FC copy code from your OSS location.
Please input your choice [1-2]: 1
Please input the log project: my-log-project
Please input the log store: my-log-store
```