



用户手册

目录

1. 前言	5
2. 进入系统	6
2.1. 初始化系统	6
2.2. 登录系统	7
3. 脚本编辑器	8
3.1. 脚本库	9
3.1.1. 添加/配置脚本集	10
3.1.2. 添加/配置脚本	11
3.1.3. 代码编辑器	12
3.1.4. 从帮助按钮获取帮助	17
3.1.5. 代码快速查看面板	18
3.2. 数据源	19
3.2.1. 添加/配置数据源	19
3.2.2. 数据源简易调试面板	24
3.3. 环境变量	27
3.3.1. 添加/配置环境变量	27
4. 管理	28
4.1. 授权链接	28
4.2. 自动触发配置	29
4.3. 批处理	29
4.4. 脚本包导出	30
4.5. 脚本包导入	31
4.6. 脚本库还原	32
4.7. 操作记录	32
4.8. 实验性功能	32
5. 函数文档、授权链接文档	33
6. 设置	34
6.1. 清除缓存	34
6.2. 代码编辑器配置	34

6.3. 用户信息	34
6.4. 修改密码	34
6.5. 成员管理	34
7. 脚本编写初步	35
7.1. 最简单的函数	35
7.2. 编写函数的要求	37
7.2.1. 谨慎编写涉及系统层面的代码以及猴子补丁	37
7.2.3. 函数名称	37
7.2.5. 函数标题	37
7.2.4. 函数定义	38
7.2.6. 函数文档	38
7.2.7. 函数参数	40
7.2.8. 函数返回值	40
8. FAQ	42
为什么使用 Python 作为脚本语言 ?	42
为什么选用 Python 3.7 版本 ?	42
我能选择不同的 Python 版本来运行脚本吗 ?	42
DataFlux Func 的性能如何 ?	42
函数执行时间过长会怎么样 ?	42
脚本调试阶段执行了慢查询, 是否会对系统有影响 ?	42
可以在脚本中使用类、装饰器吗 ?	42
可以从 DataFlux Func 连接到已有业务系统的数据库吗 ?	42
我不想对外开放业务系统数据库, 可以通过 API 方式访问吗 ?	42
DataFlux Func 如何控制函数被调用的权限 ?	43
脚本的运行环境是否绝对安全 ?	43
数据源是否绝对安全 ?	43
我在使用时遇到了问题, 应该怎么办 ?	43
我可以将自己编写的脚本分发给其他人吗 ?	43
我无法通过 API 调用函数, 为什么 ?	43
通过 API 调用函数返回了 JSON 字符串, 但 HTTP 请求头中没有 Content-Type: application/json, 为什么 ?	44
脚本集、脚本的 ID, 函数名长度有限制吗 ?	44
我是否能够在脚本中使用全局变量 ?	44

DataFlux Func 是否是 Flink、Hadoop 等的替代品？	44
DataFlux Func 的编辑器为什么没有 VS Code 好用？	44
<i>附录</i>	<i>45</i>
网络资源	45

本文档基于 v1.0.0 版本编写第初版

新版本更新后会同步更新本文档，并会标注新增内容属于哪个版本

由于本系统会持续更新升级，因此文档内容、截图等可能与实际略有出入

1. 前言

DataFlux Func 前身是 DataFlux 的一个组成部分，读作「DataFlux Function」，是一个类 Serverless 平台的多功能函数计算平台。用户只需编写具体业务处理函数脚本，即可自动生成标准 HTTP API 接口，供外部系统调用完成具体任务。

DataFlux Func 包含以下基本概念：

- 脚本库：DataFlux Func 所有脚本内容
- 脚本集：多个脚本的集合
(注：脚本集并不是 Python 模块，两者无任何关系)
- 脚本：即 Python 脚本文件，一个脚本集包含多个脚本
- 顶层函数：脚本中最外层的函数，这些函数可通过内建装饰器开放为 HTTP API
- 函数导出装饰器：将「顶层函数」生成 HTTP API 的@DFF.API(...)装饰器
- 数据源：脚本中可调用的数据源/数据库。
- 环境变量：脚本中可调用的环境变量，可用作配置、常量等
- 脚本编辑器：即脚本编辑页面
- 授权链接：用于控制外部访问函数的 URL 别名，允许从外部访问函数
- 自动触发：基于 Crontab 语法的自动执行函数的功能
- 函数文档：包含所有函数信息的动态在线文档
- 授权链接函数文档：包含所有授权链接信息的动态在线文档

安装及部署文档，请参考：

官方网站：<https://function.dataflux.cn>

- 对于首次使用的人员，请继续阅读本文
- 对于运维人员，请关注官方网站「维护手册」
- 对于开发人员，请关注官方网站「开发手册」


2. 进入系统

当全新安装 DataFlux Func，首次访问系统时，会展示初始化配置界面，允许指定系统的各项配置。

配置完成后再次进入系统，则直接进入登录界面。

2.1. 初始化系统

无论何种部署安装方式，首次进入系统都会展示配置界面：



注意：如果使用自动部署脚本或官方提供的部署配置文件进行部署，那么本页面绝大多数项目已经自动填写，请勿修改！

在初始化配置界面最后，需要填写初始管理员账号密码，作为全新安装的系统第一个用户。

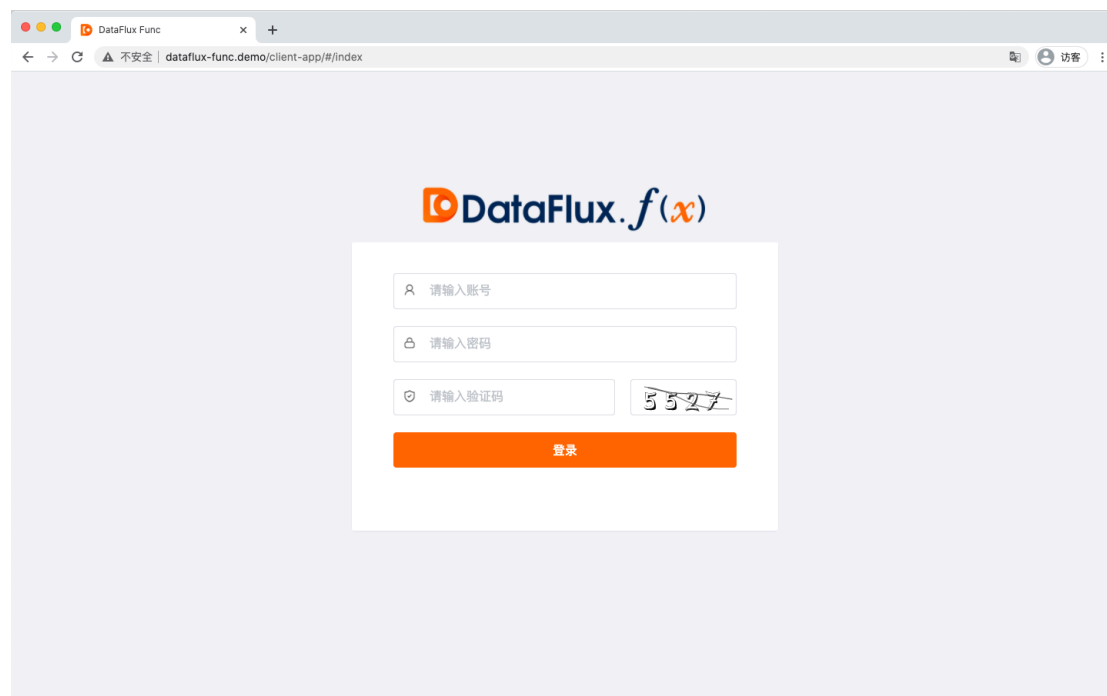


最后，点击「保存配置并初始化数据库」按钮，系统会自动创建数据库并进行初始化等处理。处理完成后，页面自动跳转到登录界面。

如需要进一步修改详细配置，请直接修改配置文件后重启系统。配置详表见「附录」

2.2. 登录系统

访问 DataFlux Func 首页后，如果没有登录，则自动跳转到登录界面。
登录界面如下图所示：



DataFlux Func 在初次安装完成后第一次运行会提示输入管理员账号密码。管理员可以在设置中继续添加其他用户的账号。

3. 脚本编辑器

登录成功后，默认进入简介画面。

简介画面提供了各画面的进入方式和简单使用方式的快速手册。

画面左边有包含脚本编辑器与脚本编写有关的所有相关项目的侧栏，包含以下内容：

- 脚本库：所有的脚本集、脚本，以及脚本中已发布的导出函数列表
- 数据源：所有脚本需要连接的数据库列表
- 环境变量：所有脚本需要的环境变量列表

当位于其他模块时，点击顶部的「脚本编辑器」即可回到脚本编辑器



3.1. 脚本库

脚本库以树状菜单显示，最顶层到最底层依次是：

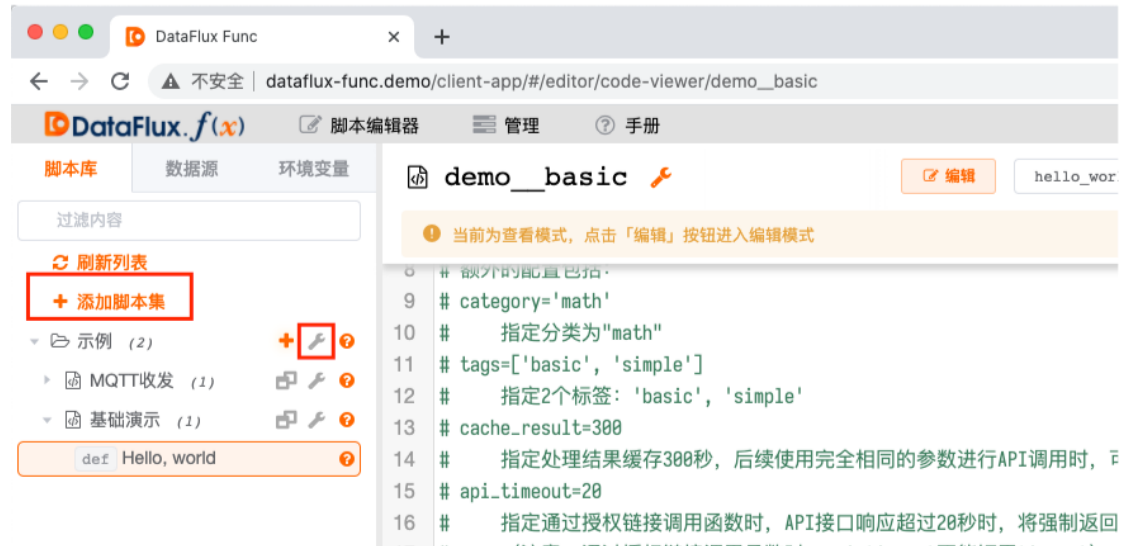
1. 脚本集
2. 脚本
3. 脚本中已发布的导出函数

在左侧边栏选择脚本后，点击「开始编辑」即可进入在线代码编辑器，对代码进行修改。



3.1.1. 添加/配置脚本集

在脚本集列表最后，点击「（添加脚本集）」可进入新脚本集添加界面。
点击列表中脚本集右侧的扳手按钮，可进入现有脚本集的配置界面。



添加、配置脚本集时，需要填写如下字段：

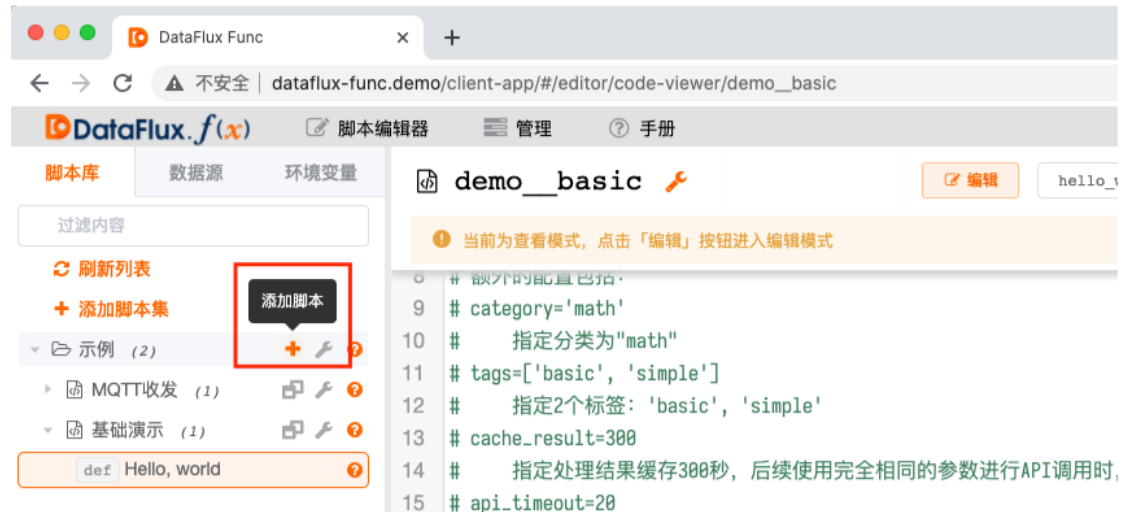
- ID：关键字段，用于后续编写 Python 脚本时和外部调用时的引用名
- 标题：单纯展示用字段，填写后作为新建的脚本集名称显示在左侧栏中
- 描述：单纯展示用字段，填写后作为新建的脚本集详情展示在左侧栏帮助提示中

配置脚本集页面除了可以修改脚本集，也可以将脚本集删除。

3.1.2. 添加/配置脚本

每个脚本集下，在脚本列表最后，点击「（添加脚本）」可进入新脚本添加界面，添加的脚本归入所属的脚本集。

点击列表中脚本右侧的扳手按钮，可进入既存脚本的配置界面。



添加脚本时，需要填写如下字段：

- ID：关键字段，用于后续编写 Python 脚本时和外部调用时的引用名
- 标题：单纯展示用字段，填写后作为新建的脚本名称显示在左侧栏中
- 描述：单纯展示用字段，填写后作为新建的脚本详情展示在左侧栏帮助提示中

配置脚本页面除了可以修改脚本，也可以将脚本删除。

3.1.3. 代码编辑器

点击任意脚本，或脚本下属的函数，右侧主面板会进入对应脚本的在线代码查看器。

此时，页面为只读状态。可以切换查看草稿、已发布的代码和草稿与已发布代码之间的差异。对于存在编辑但未发布的脚本，左侧栏会出现相应标记。



当需要修改脚本时，点击「开始编辑」，即可进入在线脚本编辑器。

在线脚本编辑器具有「草稿」状态，单纯保存仅仅为保存草稿，只有发布之后，脚本内容才会生效。（见下文）

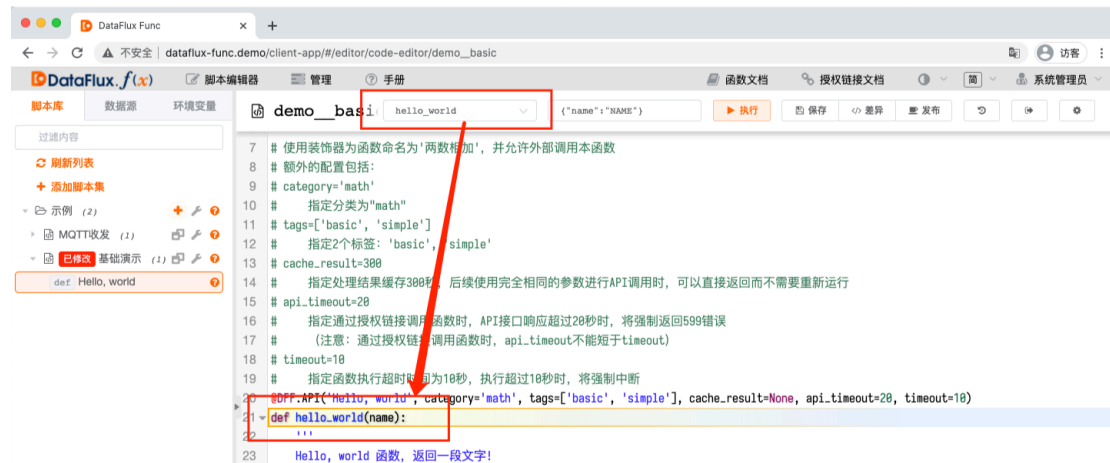
在线脚本编辑器左上方为当前脚本引用名。

右上方为操作区，从左到右依次为：

1. 当前脚本所有函数列表（包括未导出的函数）
2. 函数调试执行参数：按 JSON 格式填写的函数参数
3. 执行按钮：以填写的参数执行所选函数，并输出返回值（见下文）
4. 保存按钮：保存当前脚本为草稿
5. 发布按钮：保存并发布当前脚本
6. 复原按钮：将草稿恢复到上次发布时的状态
7. 退出编辑：进入脚本查看页面
8. 配置编辑器：修改编辑器的主题、文字大小、行距



函数选择列表中会列出所有非下划线开头的顶层函数，无论是否导出。因此，在在线脚本编辑器中，可以直接运行任何顶层脚本，方便调试。选择某个函数后，编辑器会跳转到函数所在位置，如下图：



编辑器随时可以指定函数执行，确认脚本状态。

对于附带参数的函数，需要按 JSON 格式输入参数列表，如：

```
{"x": 1, "y": 2}
```

虽然可以通过选择函数和填入参数运行任意函数

但从便捷角度，另外编写专用的测试函数更为方便，如：

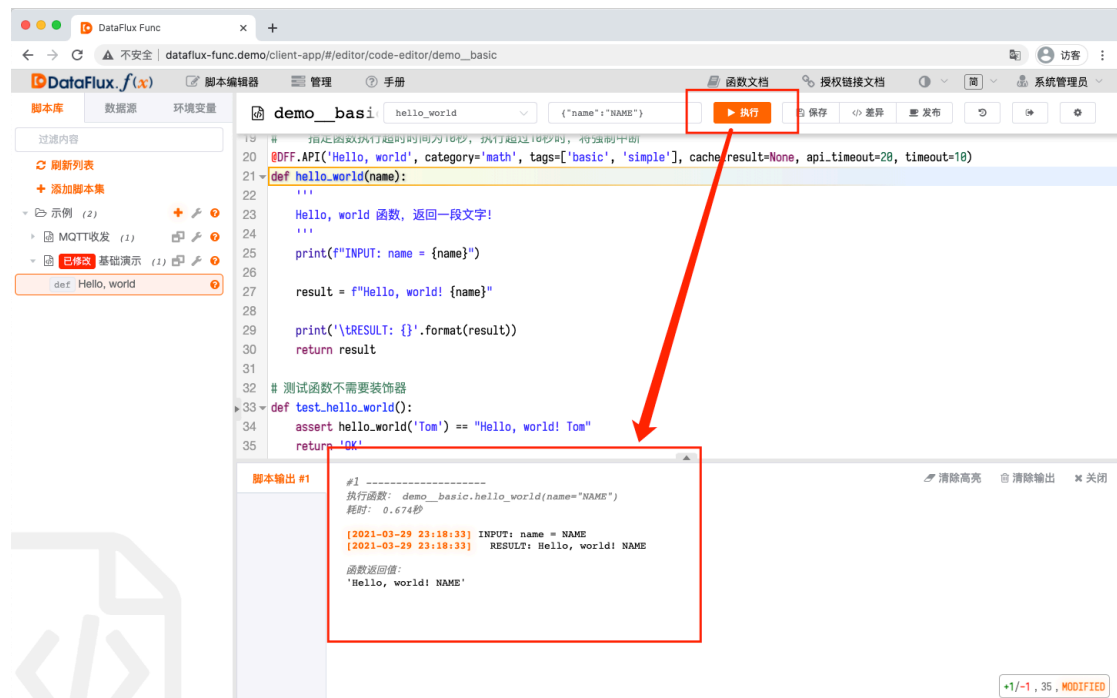
```
def my_func(a, b):  
    return a + b
```

测试函数中可以一次运行多个测试用例

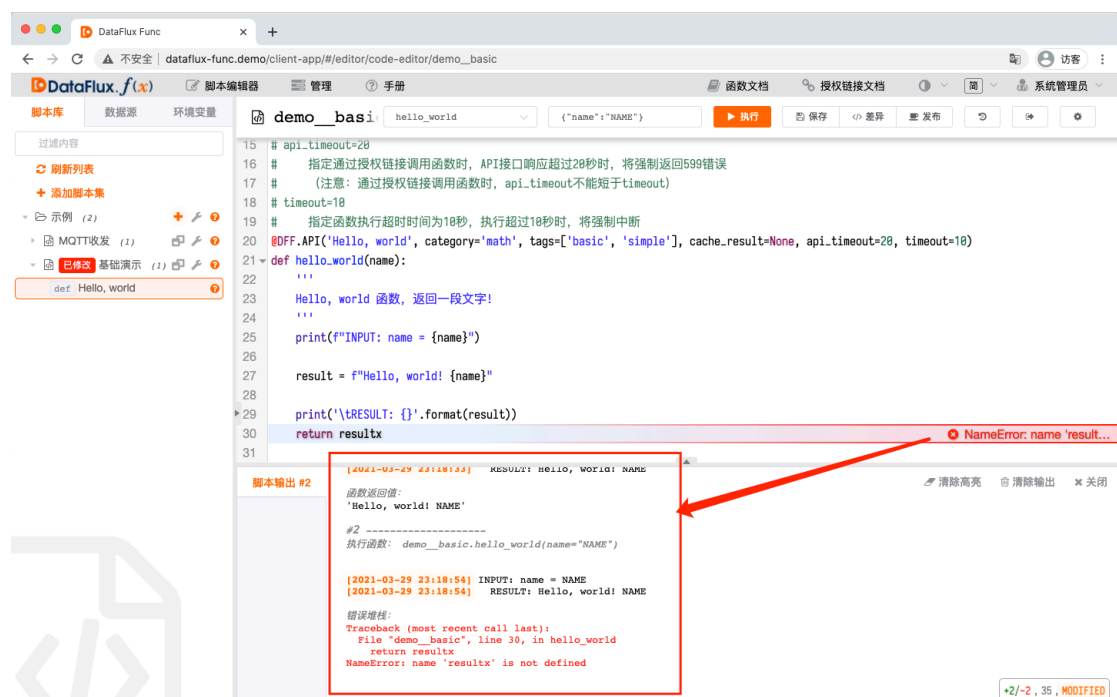
且选择函数后不用另外填写复杂的参数，直接运行即可

```
def test_my_func():  
    print(my_func(1, 2))  
    print(my_func(1, None))
```

函数执行后，编辑器底部会弹出输出窗格，包含函数执行时间、print 输出以及函数返回值，如下图：



如脚本执行出错，编辑器会标注错误行，并在底部输出窗格展示详细错误堆栈，如下图：



注意！在编辑脚本时，务必随时保存，防止误关浏览器导致内容丢失！

单纯保存脚本并不会影响系统对外的函数接口。

当确认脚本编辑完毕后，点击发布按钮，即可更新系统对外的函数接口。

在编辑页面，可以使用 Command + S (Mac) 或 Ctrl + S (Windows) 快捷键执行保存

由于缓存策略，发布后可能有一小段时间的延迟才会实际体现

脚本发布时，会进行简单的基础检查。

如果存在问题，发布会失败。同时会标注第一个错误行，并在最底下弹出的输出窗格展示详细错误信息。

当脚本编辑完毕后，请勿继续停留在编辑页面，应当点击退出编辑按钮。

同一个脚本仅允许单个登录账号处于编辑状态，其他登录账号只能阅读，因此**请勿多人共享账号**



在线代码编辑器快捷键

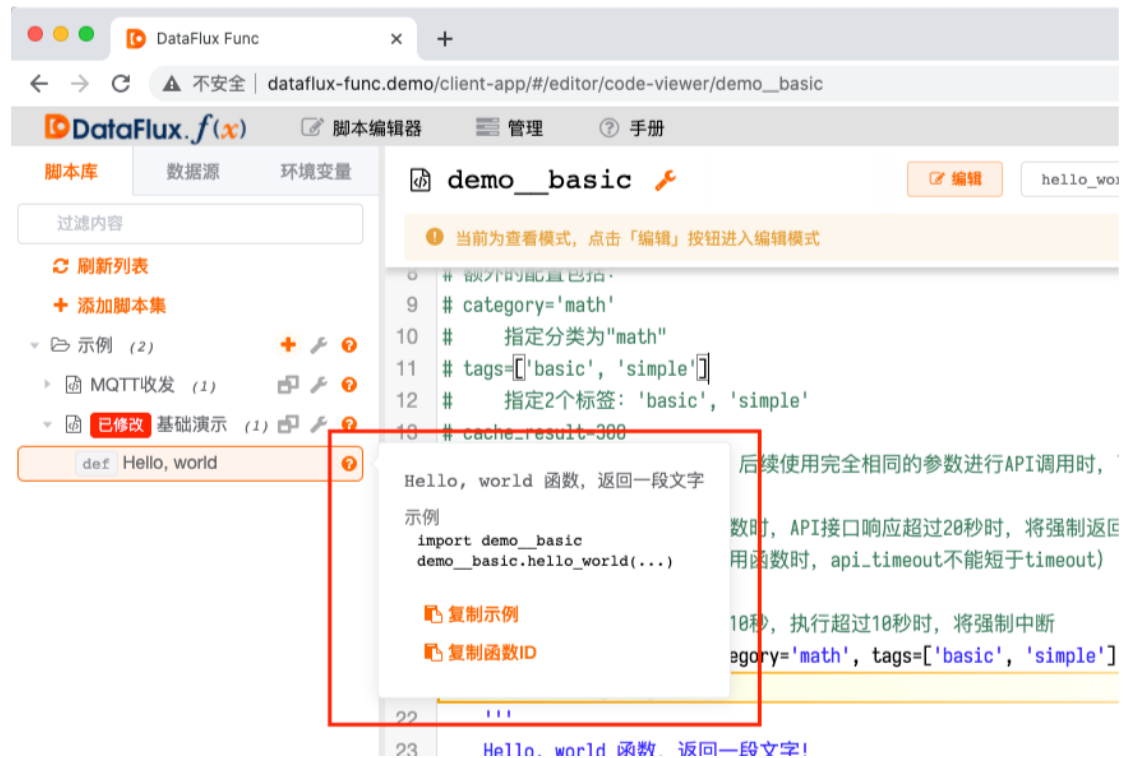
在线代码编辑器提供了一些类 Sublime Text 的快捷键。

随键输入出现提示后 TAB 或 ENTER	提示上屏
COMMAND + / (Mac) CTRL + / (Windows)	注释/取消注释 支持选中多行进行操作
光标未选中字符时 COMMAND + X (Mac) 或 CTRL + X (Windows)	剪切整行
光标未选中字符时 COMMAND + C (Mac) 或 CTRL + C (Windows)	复制整行
剪切或复制整行后 COMMAND + V (Mac) 或 CTRL + V (Windows)	粘贴整行
选中多行后 TAB	增加多行缩进
选中多行后 SHIFT + TAB	减少多行缩进
COMMAND +] (Mac) 或 CTRL +] (Windows)	增加缩进 支持选中多行进行操作
COMMAND + [(Mac) 或 CTRL + [(Windows)	减少缩进 支持选中多行进行操作
COMMAND + D (Mac) 或 CTRL + D (Windows)	选中整个单词
选中整个单词后 连续按下 COMMAND + D (Mac) 或连续按下 CTRL + D (Windows)	继续选中相同的单词 并支持多点同时编辑

3.1.4. 从帮助按钮获取帮助

左侧栏中的脚本集、脚本、数据源、环境变量右侧都有一个问号图标。点击这个问号图标，即可显示相关的信息：

- 脚本集：描述信息
- 脚本：描述信息、示例代码
- 函数：描述信息、示例代码
- 数据源：描述信息、示例代码、简易调试面板入口（见下文）
- 环境变量：描述信息、示例代码

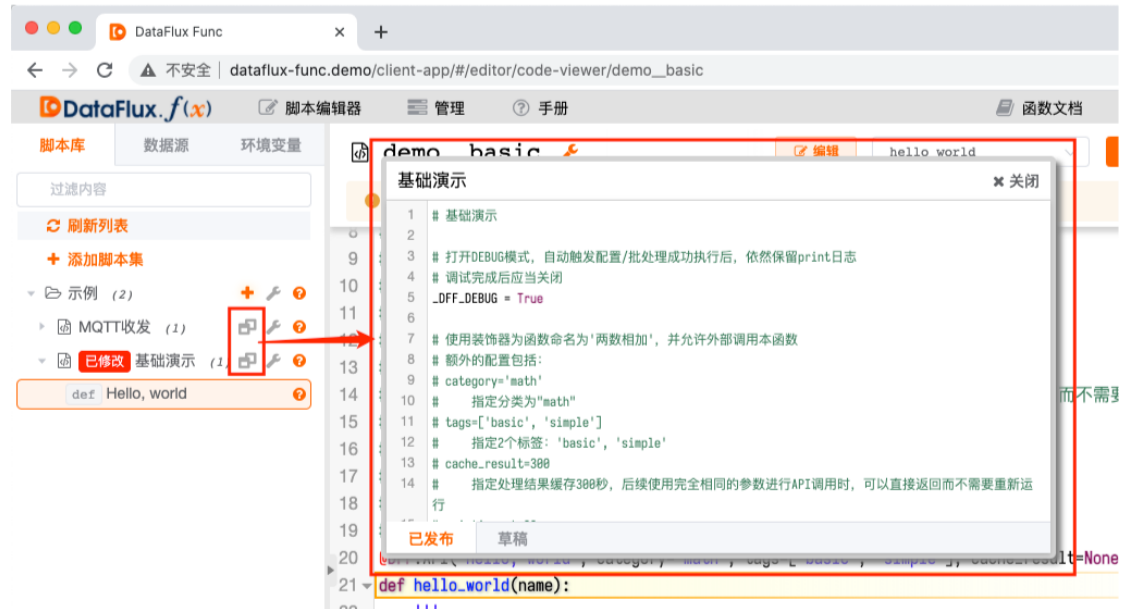


3.1.5. 代码快速查看面板

脚本库中的脚本，可以使用快速查看面板随时查看内容，且支持切换查看已发布的代码和草稿代码。点击脚本项目右侧的窗口图标，即可显示快速查看面板。

快速查看面板可以拖动到页面的任意位置。

注：快速查看面板仅在左侧栏选中脚本库时才会显示



3.2. 数据源

数据源以列表方式展示，包含了所有脚本可以访问的数据库等资源。



3.2.1. 添加/配置数据源

在数据源列表最后，点击「（添加数据源）」可进入新数据源添加界面。
点击列表中数据源右侧的扳手按钮，可进入既存数据源的配置界面。

添加数据源时，需要填写如下基本字段：

- ID：关键字段，用于后续编写 Python 脚本时和外部调用时的引用名
- 类型：关键字段，数据源所属类型（见下文）
- 标题：单纯展示用字段，填写后作为新建的数据源名称显示在左侧栏中
- 描述：单纯展示用字段，填写后作为新建的数据源详情展示在左侧栏帮助提示中
- 其他数据源所需的连接字段，如访问地址、端口等信息

在选择数据源类型后，会出现额外需要填写的数据源相关字段。不同种类的数据源所需填写的字段会有不同。

配置数据源页面除了可以修改数据源，也可以将数据源删除。

注意：数据源一旦创建，则不允许修改类型

添加数据源时，系统会自动对填写的配置做简单的连通性检查，只有能够连通的数据源才能够被添加成功。

此外，修改数据源配置时，密码栏不会显示之前填写的密码，且必须重新填写正确的密码才能够进行修改。

添加/配置 InfluxDB 数据源

InfluxDB 数据源的连接字段包括：

- 主机
- 端口
- 协议（HTTP/HTTPS）
- 数据库
- 用户名
- 密码

本数据源也可用于兼容 InfluxDB 的数据库，如：

- 阿里云时序数据库 InfluxDB 版

添加/配置 MySQL 数据源

MySQL 数据源的连接字段包括：

- 主机
- 端口
- 数据库
- 用户名
- 密码
- 编码（如：utf8mb4、utf8）

本数据源也可用于兼容 MySQL 的数据库，如：

- MariaDB
- Percona Server for MySQL
- 阿里云 PolarDB MySQL
- 阿里云 OceanBase
- 阿里云分析型数据库 (ADB) MySQL 版

添加/配置 Redis 数据源

Redis 数据源的连接字段包括：

- 主机
- 端口
- 数据库
- 密码

添加/配置 Memcached 数据源

Memcached 数据源的连接字段包括：

- 服务器列表

多个服务器使用英文逗号分隔，如：

host1:11211, host2:11211, host3:11211

添加/配置 ClickHouse (TCP) 数据源

ClickHouse (TCP) 数据源的连接字段包括：

- 主机
- 端口
- 数据库
- 用户名
- 密码

ClickHouse 会开放 2 个端口，一个用于原生 TCP 接口（默认为 9000），另一个用于 HTTP 接口（默认为 8123）。此处的端口需要填写用于原生 TCP 接口的端口。

对于使用阿里云提供的 ClickHouse 时，默认 TCP 端口可能为 3306。

具体请以控制台配置为准。

此外，各个云平台提供的数据库产品一般都会有 IP 白名单限制。添加数据源之前，请确保连通性。

添加/配置 Oracle 数据源

Oracle 数据源的连接字段包括：

- 主机
- 端口
- 数据库（服务名）
- 用户名
- 密码

添加/配置 SQL Server 数据源

SQL Server 数据源的连接字段包括：

- 主机
- 端口
- 数据库
- 用户名
- 密码

添加/配置 PostgreSQL 数据源

PostgreSQL 数据源的连接字段包括：

- 主机
- 端口
- 数据库
- 用户名
- 密码

本数据源也可用于兼容 PostgreSQL 的数据库，如：

- Greenplum Database
- 阿里云 PolarDB MySQL
- 阿里云分析型数据库 (ADB) PostgreSQL 版

添加/配置 MongoDB 数据源

mongodb 数据源的连接字段包括：

- 主机
- 端口
- 数据库
- 用户名
- 密码

添加/配置 Elasticsearch 数据源

Elasticsearch 数据源的连接字段包括：

- 主机
- 端口
- 协议 (HTTP/HTTPS)
- 数据库
- 用户名
- 密码

添加/配置 NSQ 数据源

NSQ 数据源的连接字段包括：

- 主机
- 端口
- 服务器列表
- 协议 (HTTP/HTTPS)

多个服务器使用英文逗号分隔，如：

host1:81, host2:82, host3:83

添加/配置 MQTT Broker 数据源

MQTT Broker 数据源的连接字段包括：

- 主机
- 端口
- 用户
- 密码
- 客户端 ID
- 主题/处理函数

订阅主题简易使用共享订阅，防止重复消费，如：

MQTTv5 的 \$share/GROUP/TOPIC

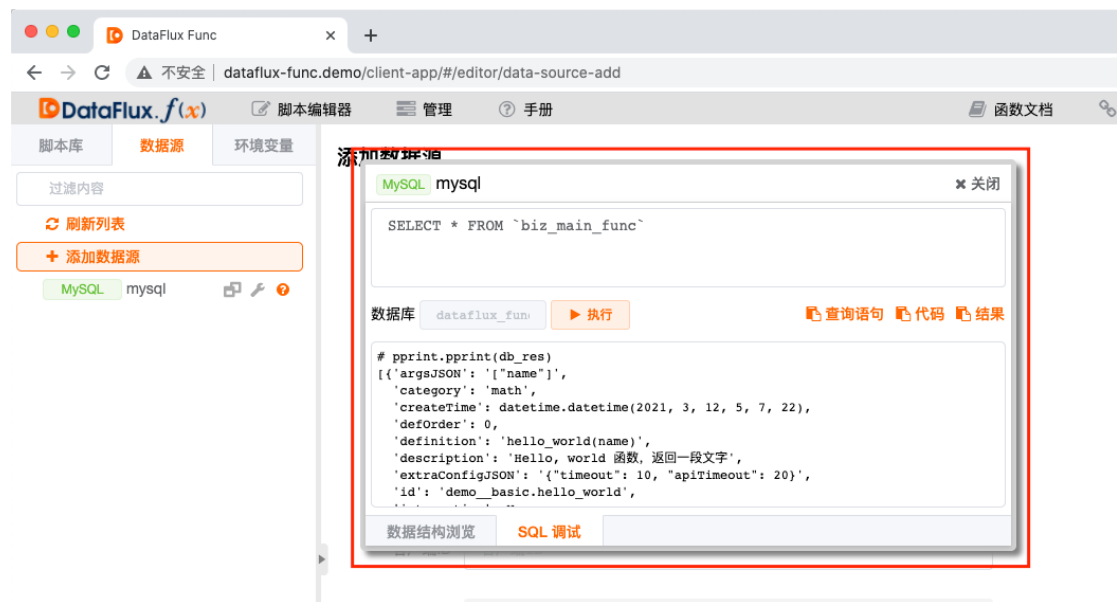
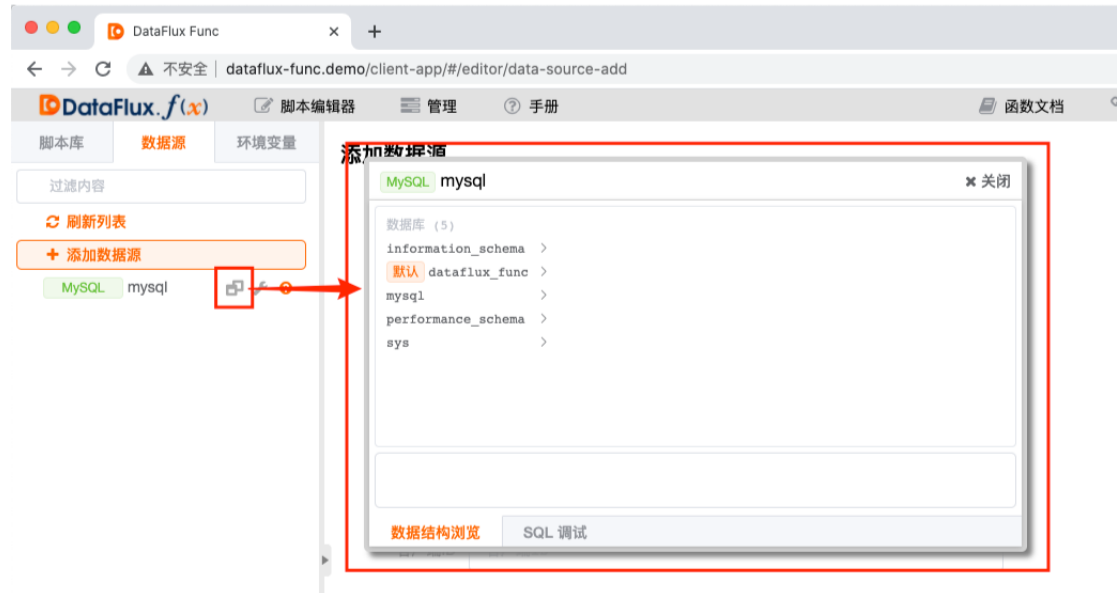
EMQX 的 \$queue/TOPIC

3.2.2. 数据源简易调试面板

已添加的数据源，可以使用简易调试面板执行查询语句，且支持执行 SQL 语句或命令，以及浏览数据结构。点击数据源项目右侧的窗口图表，即可显示简易调试面板。

简易调试面板可以拖动到页面的任意位置。

注：简易调试面板仅在左侧栏选中数据源时才会显示



此外，在调试面板中，可以直接执行 SQL 语句/命令并展示结果。如查询执行成功，面板右侧还会出现 3 个方便按钮，分别是：

- 复制查询语句
- 复制示例代码
- 复制查询结果

用户可以在确定查询语句后，直接使用上述复制功能方便后续工作。

InfluxDB 查询语句

InfluxDB 查询使用 InfluxDB Query Language (Influx QL) 语法, 如:

```
SELECT * FROM some_measurement LIMIT 10
```

详细请参考 InfluxQL 文档:

https://docs.influxdata.com/influxdb/v1.7/query_language/

MySQL 查询语句

MySQL 查询使用标准 SQL 语法, 如:

```
SELECT * FROM some_table LIMIT 10
```

详细请参考 MySQL SQL 文档:

<https://dev.mysql.com/doc/refman/5.7/en/sql-statements.html>

Redis 查询语句

Redis 查询直接使用命令行格式语法, 如:

```
GET some_key
```

详细请参考 Redis 命令文档:

<https://redis.io/commands>

Memcached 查询语句

Memcached 查询直接使用命令行格式语法, 如:

```
get some_key
```

详细请参考 Memcached 命令文档:

<https://github.com/memcached/memcached/wiki/Commands>

ClickHouse 查询语句

ClickHouse 查询使用 SQL 语法, 如:

```
SELECT * FROM some_table LIMIT 10
```

详细请参考 ClickHouse 查询语句文档:

https://clickhouse.tech/docs/zh/query_language/select/

Oracle 查询语句

Oracle 查询使用 SQL 语法，如：

```
SELECT * FROM some_table WHERE ROWNUM <= 10
```

详细请参考 Oracle 查询语句文档：

<https://www.oracle.com/technetwork/cn/database/database-technologies/sql/documentation/index.html>

SQL Server 查询语句

SQL Server 查询使用 SQL 语法，如：

```
SELECT TOP 10 * FROM some_table
```

详细请参考 SQL Server 相关文档：

<https://docs.microsoft.com/zh-cn/sql/sql-server>

PostgreSQL 查询语句

PostgreSQL 查询使用 SQL 语法，如：

```
SELECT * FROM some_table LIMIT 10
```

详细请参考 PostgreSQL 相关文档：

<https://www.postgresql.org/docs/>

mongoDB 查询方法

mongoDB 直接通过数据源操作对象提供的方法进行操作，如：

```
# 获取默认数据库对象
db = helper.db()
# 获取指定数据库对象
db = helper.db('some_database')
# 获取集合对象
collection = db['some_collection']
# 查询处理
data = collection.find_one()
# 写成一行
data = helper.db('some_database')['some_collection'].find_one()
```

详细请参考 mongoDB 相关文档：

<https://pymongo.readthedocs.io/en/stable/api/pymongo/database.html>

<https://pymongo.readthedocs.io/en/stable/api/pymongo/collection.html>

elasticsearch 查询方法

elasticsearch 直接通过 HTTP 请求来操作，可使用数据源操作对象提供的 query 方法进行操作，如：

```
data = helper.query('GET', '/some_index/_search?q=some_field:something')
```

详细请参考 elasticsearch 相关文档：

<https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html>

3.3. 环境变量

环境变量以列表方式展示，包含了所有脚本可以访问的环境变量。

3.3.1. 添加/配置环境变量

在环境变量列表最后，点击「（添加环境变量）」可进入新环境变量添加界面。

点击列表中环境变量右侧的扳手按钮，可进入既存环境变量的配置界面。

添加环境变量时，需要填写如下字段：

- ID：关键字段，用于后续编写 Python 脚本时和外部调用时的引用名
- 标题：单纯展示用字段，填写后作为新建的环境变量名称显示在左侧栏中
- 描述：单纯展示用字段，填写后作为新建的环境变量详情展示在左侧栏帮助提示中
- 值：关键字段，环境变量内容
- 自动转换类型：在取出环境变量值时自动转换类型

自动转换类型支持：

- 字符串：默认，即不转换
- 整数：取出时使用 `int()` 函数转换
- 浮点数：取出时使用 `float()` 函数转换
- 布尔值：取出时尝试转换为 `True` 或 `False`
 - `true, yes, on` 会被转换为 `True`（无论大小写）
 - `false, no, off` 会被转换为 `False`（无论大小写）
- JSON：取出时使用 `simplejson.loads()` 转换
- 逗号分隔数组：取出时使用英文逗号分隔，转换为字符串数组

配置环境变量页面除了可以修改环境变量，也可以将环境变量删除。

4. 管理

点击顶部的「管理」即可进入管理模块

管理模块包含了有关本系统所有的管理项目，包含：

- 总览：展示系统当前使用状况的信息
- 关于：展示系统版本等信息
- 授权链接：允许函数可以从外部访问
- 自动触发配置：允许函数以特定频率自动被调用
- 批处理：允许函数以异步方式运行较长时间
- 脚本包导出/导入：导出/导入脚本集
- 脚本库还原：允许将整个脚本库（所有脚本集和所有脚本）还原至特定的时间点
- 成员管理：添加/修改 DataFlux Func 操作账户
- 操作记录：可以查看近期较为关键的操作日志

注意：由于本系统会持续更新，管理模块会不断增加新的管理项目

4.1. 授权链接

点击「管理」模块左侧栏的「授权链接」，即可打开授权链接列表。列表包含了当前已经创建的所有授权链接以及相关信息。

点击「API 调用示例」，可以查看对应授权链接的可用参数以及调用方式示例。

授权链接支持多种调用方式，包括：

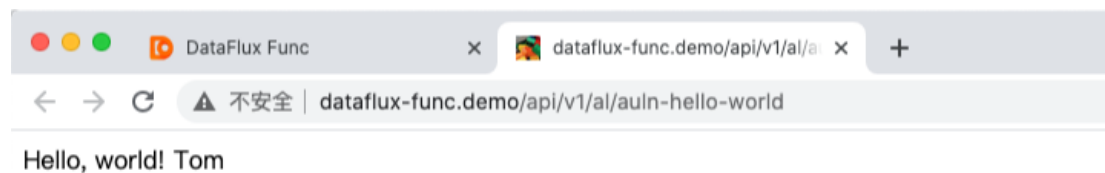
- POST 方式：参数、选项包含在 body 中，以 json 方式发送请求
- POST 方式（简化形式）：参数包含在 body 中，以 form 方式发送请求。**支持文件上传（需要 form-data/multipart 方式）**
- GET 方式：参数、选项按照 JSON 序列化后包含在 URL 中
- GET 方式（简化形式）：参数包含在 URL 中

各种调用方式存在一些各自的限制：

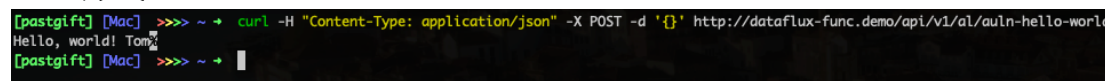
调用方式	传递 kwargs	kwargs 中参数类型	传递 options	文件上传
POST 标准	支持	支持 JSON 中的数据类型	支持	不支持
POST 简化	支持	只能传递字符串	不支持	支持
GET 标准	支持	支持 JSON 中的数据类型	支持	不支持
GET 简化	支持	只能传递字符串	不支持	不支持

任何**同步**调用方式都能得到相同的返回结果，可根据需要自行选择。

GET 方式:



POST 方式:



注意：使用异步方式调用时，系统不会立即返回函数结果，而是返回结果查询 URL 地址。调用方需要后续根据此 URL 地址查询结果。大多数情况下，使用同步方式即可。长耗时异步任务请使用「批处理」功能

在授权链接列表，点击「新建授权链接」，即可进入添加授权链接页面。
新建授权链接需要填写以下内容：

- 自定义 ID【可选】
- 执行函数
- 调用参数（参数会在选择函数后自动生成模板）
- 是否显示在「授权链接函数文档」中（见下文）
- 有效期
- 限流策略（允许按年、月、日、时、分、秒限制调用次数）
- 备注

4.2. 自动触发配置

点击「管理」模块左侧栏的「自动触发配置」，即可打开自动触发配置列表。
列表包含了当前已经创建的所有自动触发配置以及相关信息。

在自动触发配置列表，点击「新建自动触发配置」，即可进入添加自动触发配置页面。

新建自动触发配置需要填写以下内容：

- 执行函数
- 调用参数（参数会在选择函数后自动生成模板）
- 重复规则（Crontab 语法格式）
- 备注

4.3. 批处理

点击「管理」模块左侧栏的「批处理」，即可打开批处理列表。列表包含了当前已经创建的所有批处理以及相关信息。

在批处理列表，点击「新建批处理」，即可进入添加批处理页面。

新建批处理需要填写以下内容：

- 自定义 ID 【可选】
- 执行函数
- 调用参数（参数会在选择函数后自动生成模板）
- 备注

批处理的使用方式和授权链接基本相同，但面向场景不同。批处理主要用于处理长耗时的异步任务。因此投递任务（即调用 API）后，系统会立即返回，而不会等待处理结束。

4.4. 脚本包导出

点击「管理」模块左侧的「脚本包导出」，即可打开脚本包导出界面。

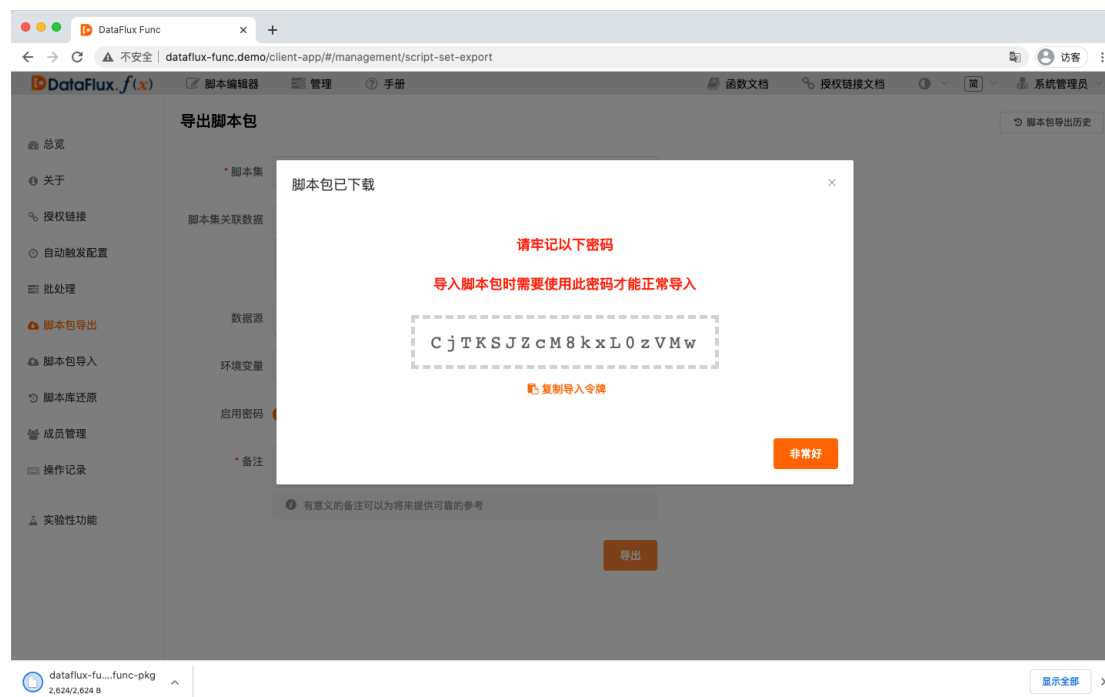
导出脚本包需要填写以下内容：

- 脚本集
- 是否同时导出「授权链接」、「自动触发配置」、「批处理」
- 需要同时导出的数据源、环境变量
- 是否启用密码（启用后会在导出时自动生成）
- 操作人（仅用于记录说明）
- 操作备注（仅用于记录说明）

成功导出后的脚本包文件会自动下载，文件名为固定格式如下：

```
dataflux-func-<导出年月日>_<导出时分秒>.func-pkg
```

当启用密码后，系统会自动生成区分大小写的密码显示在页面。此密码需要在导入脚本包时填写才能正常导入，用于防止代码中途泄漏。（见下文）



4.5. 脚本包导入

点击「管理」模块左侧的「脚本包导入」，即可打开脚本包导入历史列表。列表包含了已执行过的脚本包导入历史。

在脚本包导入历史列表，点击「导入脚本包」，即可进入脚本包导入页面。导入脚本包需要填写以下内容：

- 脚本包文件（扩展名为 **.func-pkg**）
- 导入令牌（仅当导出时开启密码时需要，否则留空即可）

脚本包上传并解析成功后，会出现确认导入对话框。提示新增脚本集，或某个脚本集在导入后会被替换。

注意：如当前脚本库中已经存在相同 ID 的脚本集，则会完整替换为导入的脚本集。此外脚本包导入成功后立刻生效，不需要额外进行发布操作

此外，如果导入的脚本包包含授权链接、自动触发配置、批处理、数据源、环境变量，那么导入后具有相同 ID 的数据将会被替换。

4.6. 脚本库还原

点击「管理」模块左侧的「脚本库还原」，即可打开还原点列表。列表包含了所有可以恢复的脚本库还原点。需要还原脚本库到某个时间点时，可点击列表右侧「还原至此状态」。

脚本库还原会立刻生效，不需要额外进行发布操作

还原成功后，还原操作本身也会产生还原点。因此即使误操作执行了还原，也能够还原到未还原之前的状态。

在脚本库还原点列表，点击「创建还原点」，即可进入创建脚本库还原点页面。

创建还原点需要填写以下内容：

- 说明（用于对即将创建的还原点添加描述）

4.7 操作记录

点击「管理」模块左侧的「操作记录」，即可打开可操作记录列表。所有关键操作（修改、删除等）都会记录再此但查询类操作不会记录。

4.8 实验性功能

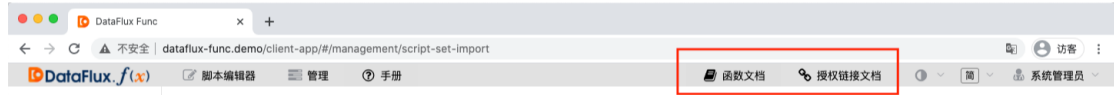
点击「管理」模块左侧的「实验性功能」，即可进入实验性功能开关配置页面。用户可以在此体验一些尚未提升为正式发布的前瞻性功能。具体功能详见实际页面说明

注意：实验性功能可能存在一些问题，不保证一定可用。因此请勿过分依赖实验性功能。此外，实验性功能也会不定期新增、修改、删除，驻云官方不会另行告知。

5. 函数文档、授权链接文档

点击导航栏的「函数文档」、「授权链接文档」，会在新页面打开文档页面。「函数文档」包含了所有当前已发布脚本中，通过 `DFF.API()` 装饰器修饰的所有函数。

「授权链接文档」包含了所有标记为显示在文档中的授权链接。



文档页面也可以通过点击「API 调用示例」，查看对应函数的可用参数以及调用方式示例。

授权链接函数文档包含 JSON 版，供第三方应用程序获取，访问以下地址即可：

```
GET /api/v1/auth-link-func-list
```

此外，授权链接函数文档 JSON 版支持查询参数，如：

```
GET /api/v1/auth-link-func-list?scriptSetId=demo&scriptId=demo__basic
```

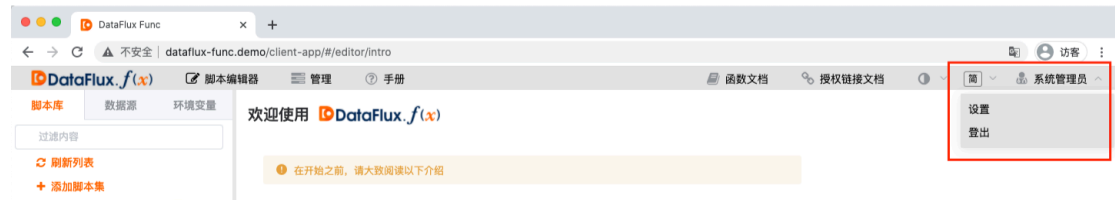
完整参数如下：

参数	说明	可选值
funcId	函数 ID	如: demo__basic.plus
funcName	函数名	如: plus
funcCategory	函数分类 (即 category)	代码中 <code>@DFF.API()</code> 装饰器指定的 <code>category</code> 参数值 可以使用英文逗号方式传递多个，表示过滤包含任意指定分类的函数 如: <code>category=a,b</code> 可以匹配分类为 <code>a</code> 和分类为 <code>b</code> 的函数，但无法匹配分类为 <code>c</code> 的函数
funcTags	函数标签 (即 tags)	代码中 <code>@DFF.API()</code> 装饰器指定的 <code>tags</code> 参数 可以使用英文逗号方式传递多个，表示过滤包含所有指定标签的函数 如: <code>tags=a,b</code> 可以匹配具有标签 <code>a</code> 、 <code>b</code> 、 <code>c</code> 的函数，但无法匹配具有标签 <code>a</code> 的函数
scriptId	脚本 ID	如: demo__basic
scriptSetId	脚本集 ID	如: demo

6. 设置

点击右上角当前登录用户信息下拉框，选择「设置」，即可进入设置页面。
在设置页面，可以进行以下操作：

- 清除缓存
- 代码编辑器配置
- 用户信息
- 修改密码



6.1. 清除缓存

点击「设置」模块左侧的「清除缓存」，即可进入缓存清除页面。一般来说，只有在页面操作时发生无法恢复的问题时，才需要尝试清除缓存。

6.2. 代码编辑器配置

点击「设置」模块左侧的「代码编辑器配置」，即可进入代码编辑器配置页面。用户可在此页面调整代码编辑器的着色主题，文字大小、行距等。右侧的预览可以看到实际效果。

6.3. 用户信息

点击「设置」模块左侧的「用户信息」，即可进入当前用户信息页面。用户可在此页面修改自己的用户信息。

6.4. 修改密码

点击「设置」模块左侧的「修改密码」，即可进入密码修改页面。用户可在此页面修改自己的登录密码。

6.5. 成员管理

点击「设置」模块左侧的「成员管理」，即可进入成员管理页面。系统管理员可在此页面管理其他用户。

注意：只有初始安装时创建的系统管理员才可用此选项。此外，为了保证操作记录完整性，用户一旦创建只能禁用，无法删除。

7. 脚本编写初步

DataFlux Func 使用 Python 3.7 作为运行环境，并对脚本的执行上下文进行了一些定制，与原版的 Python 有一定区别。

根据 DataFlux Func 要求，本系统中的 Python 脚本都应该使用 **4 个空格** 作为缩进单位，请勿使用 tab 等其他缩进方式。

此外，由于脚本中的函数返回值最终都会通过 HTTP 协议，以 JSON 数据返回给调用方。因此，函数返回值要求能够被序列化为 JSON 数据，一些复杂对象实例无法直接返回。

7.1. 最简单的函数

一个最简单的函数如下：

```
def hello_world():  
    return 'Hello, World!'
```

可以看到，这个函数和原版 Python 中的函数写法并没有任何区别。大部分情况下，在 DataFlux Func 中编写函数也与和原版 Python 没有什么区别。

此时，hello_world() 函数已经可以使用，但仅限 DataFlux Func 内部被其他函数调用。如果希望函数能够在 DataFlux Func 外部被调用，则需要添加 @DFF.API() 装饰器。

典型示例如下：

```
@DFF.API('一个最简单的函数')  
def hello_world():  
    return 'Hello, World!'
```

除了直接返回字符串，也可以返回能够序列化为 JSON 的对象。函数本身不需要手工进行序列化操作，如：

```
@DFF.API('一个最简单的函数')  
def hello_world():  
    ret = { 'message': 'Hello, World!' }  
    return ret
```

如果确实没有返回值需要返回，返回 None 即可（Python 没有 return 语句时，同样返回 None），如：

```
@DFF.API('一个没有返回值的函数')  
def hello_world():  
    return None
```

函数可以附带参数、函数文档，同样按照正常 Python 编写即可，如：

```
@DFF.API('问候')
def hello_world(you_name=None):
    """
    本函数返回一句问候语
    参数：
        your_name 可选，字符串。问候者姓名
    返回：
        字符串
    """
    if you_name is None:
        your_name = 'My friend'
    return 'Hello, World! ' + your_name
```

在发布脚本后，DataFlux Func 会提取函数的标题、名称、参数列表、函数文档等，供外部系统使用或通过函数文档对外展示。因此请务必正确填写。

7.2. 编写函数的要求

在 DataFlux Func 中编写函数，存在一些限制或要求。

在编写函数时，请务必遵守这些规定，否则可能会带来意想不到的问题

7.2.1. 谨慎编写涉及系统层面的代码以及猴子补丁

虽然 DataFlux Func 已经在用户脚本运行的上下文环境隔离上做了一些努力。但由于 Python 目前并没有一个较为可靠的沙箱系统，而每次执行函数都启动一个 Docker 容器也无法满足性能要求（大多数函数仅仅只需要运行不到 1 秒）。因此用户代码的实际运行环境在同一个进程中。

虽然 DataFlux Func 再每次运行函数时，会将用户代码的上下文（globals）初始化，但系统级对象（引入的库、磁盘文件等）依然是共享的。**用户在编写代码时应当注意有关系统级对象的操作，以及各类猴子补丁的副作用。**

7.2.3. 函数名称

函数名称即 def 后面的函数名称，在 DataFlux Func 中也被称为函数 ID。除了需要满足 Python 本身的要求外，在 DataFlux Func 中额外要求通过 @DFF.API() 开放的函数，其所在的脚本集 ID、脚本 ID 以及本身函数名称总长度不超过 253 个字符。

不用担心长度不够用，超过 100 个字符长度的命名我也只在 Java 里见过！

为了方便函数 ID 出现在 URL 中的情况，**通过 @DFF.API() 开放的函数建议只使用小写英文、数字及下划线**。对于其他普通函数，支持使用中文命名，以方便处理一些在行业中难以用英文表示的晦涩术语。

```
def 获取金秋活动蜜饼代金券积分兑换券早鸟码():  
    return 'code-001'  
  
@DFF.API('获取活动数据')  
def get_event_data():  
    return 获取金秋活动蜜饼代金券积分兑换券早鸟码()
```

7.2.5. 函数标题

函数标题为 @DFF.API() 装饰器的第一个参数 title，在 DataFlux Func 中要求长度不超过 256 个字符。

7.2.4. 函数定义

为了便于代码解析，函数定义**无论多长都务必写在一行里**，如：

```
def hello_world(arg1=None, arg2=None, arg3=None, args4=None):  
    pass
```

以下为**错误示范**：

```
def hello_world(arg1=None,  
                arg2=None,  
                arg3=None,  
                args4=None):  
    pass
```

7.2.6. 函数文档

函数文档为函数定义行（def 行）之后，第一个代码块，如：

```
@DDF.API('问候')  
def hello_world(you_name=None):  
    '''  
    本函数返回一句问候语  
    参数：  
        your_name 可选，字符串。问候者姓名  
    返回：  
        字符串  
    '''  
    if you_name is None:  
        your_name = 'My friend'  
    return 'Hello, World! ' + your_name
```

编写时，缩进请勿小于三引号，并保证三引号都各自单独占有一行。

以下为**错误示范**：

```
def hello_world():  
    '''这是函数文档的错误示范'''  
    pass
```

```
def hello_world():  
    '''  
    这是函数文档的错误示范'''  
    pass
```

```
def hello_world():  
    '''这是函数文档的错误示范  
    '''  
  
    pass
```

```
def hello_world():  
    '''  
这是函数文档的错误示范  
    '''  
  
    pass
```

如内容比较多，需要分节展示，可以使用悬挂缩进方式：

```
def hello_world():  
    '''  
    内容比较多需要分节时，可以悬挂缩进  
    第一节标题  
        第一节内容  
    第二节标题  
        第二节内容  
    '''  
  
    pass
```

标准模板：

```
def hello_world(you_name=None):  
    '''  
    本函数返回一句问候语  
    参数：  
        your_name 可选，字符串。问候者姓名  
    返回：  
        字符串  
    '''  
  
    pass
```


7.2.7. 函数参数

函数参数本质上与原版 Python 并没有什么区别，但是考虑到函数作为 HTTP API 调用时，无法明确类型（如 GET 请求的 query 参数），所以接收到的参数值可能是字符串类型，因此需要函数内进行判断、转换类型。

```
@DFF.API('错误示范')
def get_sqrt(number=None):
    """
    本函数执行可能会由于 number 参数为字符串而失败
    """
    return number * number
```

```
@DFF.API('正确示范')
def get_sqrt(number=None):
    """
    应当进行判断/类型转换
    """
    try:
        number = float(number)
    except Exception as e:
        raise Exception('Parameter `number` is not a number')

    return number * number
```

7.2.8. 函数返回值

函数返回值本质上与原版 Python 并没有什么区别，但是考虑到面向外部系统的易用性，返回值需要可以被外部系统正常解析。

总是返回能够被序列化为 JSON 的数据

在 Python，可以被正常序列化的数据为 `dict`、`list`、`tuple`、`int`、`float`、`str`、`unicode`、`bool`、`None` 等基本内置数据类型。

此外，对于第三方库特有的对象实例，如果实现了对应的序列化方法（即虽然不是内置数据类型，但依然可以序列化为 JSON）也可以返回。

```
@DFF.API('正确示范')
def hello_world():
    ret = {
        'list': [1, 1.1, True, False, None],
        'dict': {'int': 1, 'float': 1.1, 'bool': True, 'null': None},
        'int': 1,
        'float': 1.1,
        'bool': True,
        'null': None,
    }
```

不要返回多个返回值

由于函数返回值在 JSON 序列化后会包裹在 `result` 字段中，方便程序获取，而返回多个返回值会导致 `result` 变成数组。

```
@DFF.API('错误示范')
def hello_world():
    """
    请勿返回多个返回值
    """
    return True, [1, 2, 3]
```

```
@DFF.API('正确示范')
def hello_world():
    """
    一次只返回一个返回值
    """
    ret = {
        'ok': True,
        'data': [1, 2, 3]
    }
    return ret
```

错误应当使用 `raise` 语句抛出，而不是 `return`

由于 DataFlux Func 本身需要监控函数执行成功与否。因此，当函数执行发生错误时，应当使用 `raise` 语句将问题抛出，而不是 `return`。否则 DataFlux Func 无法获知函数是否正确执行。

```
@DFF.API('错误示范')
def hello_world():
    """
    请勿使用 return 代替 raise
    """
    return False
```

```
@DFF.API('正确示范')
def hello_world():
    """
    出错应当使用 raise 语句
    """
    raise Exception('Some error message')
```

8. FAQ

为什么使用 Python 作为脚本语言？

Python 这门语言较为简单易上手。此外，Python 生态已经大量存在了专业的数学、机器学习等第三方库，可以方便地满足业务需要。

为什么选用 Python 3.7 版本？

首先，Python 2.x 版本已于 2020 年 1 月 1 日停止维护，Python 3.x 是大势所趋。其次，Python 3.7 版本为目前的稳定版本，各第三方库对 3.7 的支持也最全面。

我能选择不同的 Python 版本来运行脚本吗？

不可以，DataFlux Func 是服务端应用，且为了性能，并没有使用 Docker 等技术来运行用户脚本，因此用户脚本都运行在同一个 Python 3.7 环境下。

DataFlux Func 的性能如何？

很大程度上与函数处理内容有关，过分复杂的处理必然拖慢系统。根据测试，单个 Pod 执行 Hello World 函数的性能大概在 200 次/秒左右。可以根据需要，增加 Worker 的数量以提高处理能力。

函数执行时间过长会怎么样？

一般来说，一个被外部系统同步调用，并将返回数据应用于 UI 界面的函数，建议执行时长为 3 秒以内，以确保外部系统和 UI 界面操作的连贯性。但 DataFlux Func 最长允许一个同步方式调用的函数执行 30 秒。超过此时长，执行进程会被强制停止。对于批处理方式执行的函数，最长允许执行 3600 秒（即 1 小时）

脚本调试阶段执行了慢查询，是否会对系统有影响？

DataFlux Func 调试执行与正式被 API 调用执行分属两个不同的执行单元，本身不会相互影响。但执行慢查询的数据源会受到影响。

可以在脚本中使用类、装饰器吗？

可以，类和装饰器的使用与原版 Python 没有任何区别。

可以从 DataFlux Func 连接到已有业务系统的数据库吗？

可以，但需要保证 DataFlux Func 能够连通业务系统的数据库。此外，为了避免可能存在的慢查询影响线上业务系统，建议对业务系统数据库做读写分离，并让 DataFlux Func 连接到读库上。

我不想对外开放业务系统数据库，可以通过 API 方式访问吗？

可以，DataFlux Func 本身预装了 `requests` 库，可以向外发送 HTTP 请求。

DataFlux Func 如何控制函数被调用的权限？

DataFlux Func 本身不提供权限系统，编写的函数默认不能被外部直接调用，只有创建对应的授权链接、批处理才能被外网间接调用。授权链接可以设置限流，也可以随时禁用；批处理也可以随时禁用。因此开放函数时，只需要控制授权链接和批处理即可。

脚本的运行环境是否绝对安全？

不是绝对安全，Python 是动态脚本语言，虽然 DataFlux Func 已经做了一定限制，但无法 100% 杜绝猴子补丁等越狱手段。

此外，目前 DataFlux Func 也没有对 import 的库进行限制，脚本实际可以对 Worker 的运行环境做任何事情，也因此我们也建议使用 Docker 方式运行 DataFlux Func。

切忌将 DataFlux Func 的访问权限交给不可信任的第三方使用。

数据源是否绝对安全？

数据源安全性主要基于配置和使用方式。

由于函数可以由用户自行编写，因此类似 SQL 注入等问题依然需要脚本编写者注意规避。DataFlux Func 的数据源操作对象支持防止 SQL 注入的传参功能，请勿手工在 SQL 语句中手工拼接外部输入的参数。

因此，除非有特定业务需求，数据源建议使用只读库、只读用户、在数据库中为用户指定明确的权限等方式减少安全隐患。

我在使用时遇到了问题，应该怎么办？

如果是 DataFlux Func 本身的故障，请联系官方反馈问题。

如果是使用方面的疑问，请先阅读相关文档。如依然无法解决问题，请联系官方反馈问题。

我可以将自己编写的脚本分发给其他人吗？

可以，DataFlux Func 并不会限制用户分发代码。

我无法通过 API 调用函数，为什么？

无法调用函数的问题有很多种

- 请求返回 4xx 或 5xx：
网络正常，错误原因请参考返回内容。大多数情况为函数本身的问题。
- 请求超时，服务器无响应：
网络不通，这并不是 DataFlux Func 的问题，请检查自身网络、代理等配置是否正确。
- 请求返回未登录或需要登录：
您可能尝试直接调用了函数而不是调用授权链接或批处理。根据 DataFlux Func 要求，从外部调用函数需要通过授权链接或批处理调用

通过 API 调用函数返回了 JSON 字符串，但 HTTP 请求头中没有 Content-Type: application/json，为什么？

您所调用的函数可能直接返回了序列化后的 JSON 字符串。当函数返回 JSON 数据时，请直接返回整个 dict 对象，不需要手动进行序列化操作。而直接返回字符会被当作 plain/text 处理。

脚本集、脚本的 ID，函数名长度有限制吗？

有，脚本集、脚本的 ID 与函数名总长度不能超过 253 个字符。过长的命名的可能会导致函数无法正常调用。

我是否能够在脚本中使用全局变量？

可以，但仅限只读方式访问定义的全局变量。
由于每次函数运行都会初始化执行上下文，因此函数中对全局变量的修改，不会保留到下次函数执行。

DataFlux Func 是否是 Flink、Hadoop 等的替代品？

不是，DataFlux Func 前身是 DataFlux 产品的一个组成部分，仅仅是涉及到「大数据」，但与目前市面上的大数据处理系统没有什么特定的关系，仅仅是功能上有共同点。而目前 DataFlux Func 则以独立系统方向发展，本身定位为一个轻量级函数计算平台。

使用 DataFlux Func 请按照其本身的逻辑使用，而不是简单替换已有的系统。

DataFlux Func 的编辑器为什么没有 VS Code 好用？

VS Code 是非常成功并且先进的 IDE，一个很重要的原因是项目经理是两个写了 20 年 Eclipse 的「老伙计」。我们会始终将其视为榜样，为用户提供更好的「简易」编辑器。

附录

网络资源

DataFlux 官方网站:

<https://www.dataflux.cn/>

DataFlux Func 官方网站:

<https://function.dataflux.cn/>