workflow使用手册

1. 功能介绍
   1.1. 创建数据源
      1.1.1. 数据源探活规则
   1.2. 创建画布
   1.3. 画布功能
   1.4. 组件功能
   1.5. 编排规则
   1.6. 发布API
   1.7. 创建密钥
      1.7.1. 点击 API 名称跳转到 API 详情页面
      1.7.2. 新增密钥
   1.8. 调用API
2. 组件介绍
   2.1. 开始组件
   2.2. 结束组件
   2.3. 条件组件
   2.4. 迭代组件
   2.5. 执行代码组件
   2.6. 并发组件
   2.7. 聚合组件
   2.8. HTTP组件
   2.9. HTTP-XML组件
   2.10. 数据库组件
   2.11. 文件服务器组件
3. 环境变量

1. 功能介绍

1.1. 创建数据源

MySQL规则 Go

```
1 {"host": "10.99.29.9", "port": 3306, "user": "root", "database": "wkflow", "password": "Root@123"}
```

Oracle规则 Go

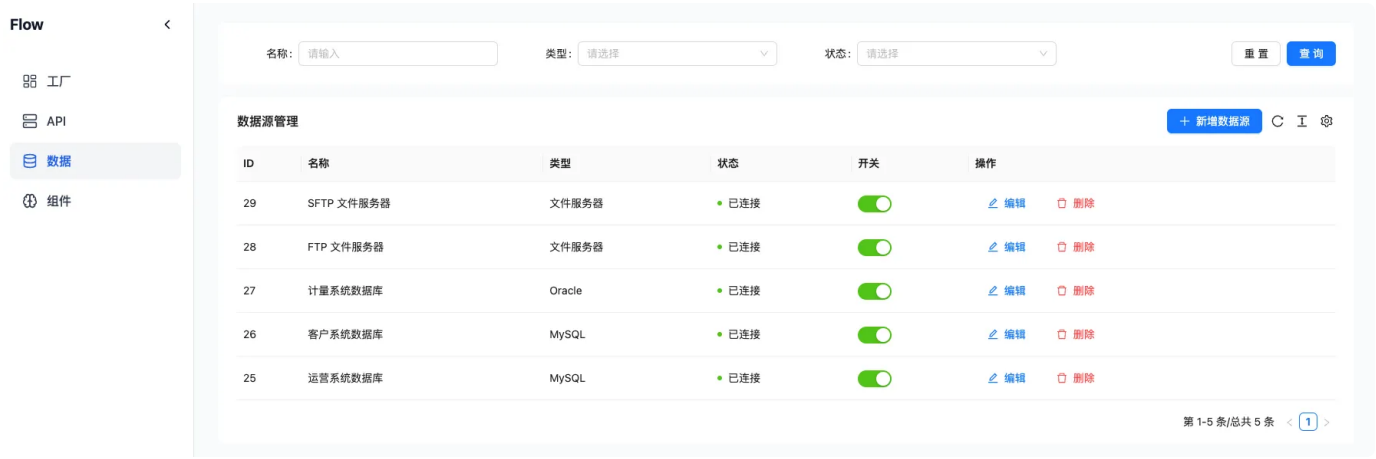
```
1 {"host": "10.99.220.223", "port": 1521, "user": "test3", "database": "helowin", "password": "test3"}
```

FTP Go

```
1 {"host": "10.99.113.114", "port": 21, "passive": true, "password": "test",
  "protocol": "ftp", "username": "test"}
```

SFTP Go

```
1 {"host": "10.99.29.7", "port": 2233, "passive": true, "password": "Bepasswo
  rd@123", "protocol": "sftp", "username": "beuser"}
```

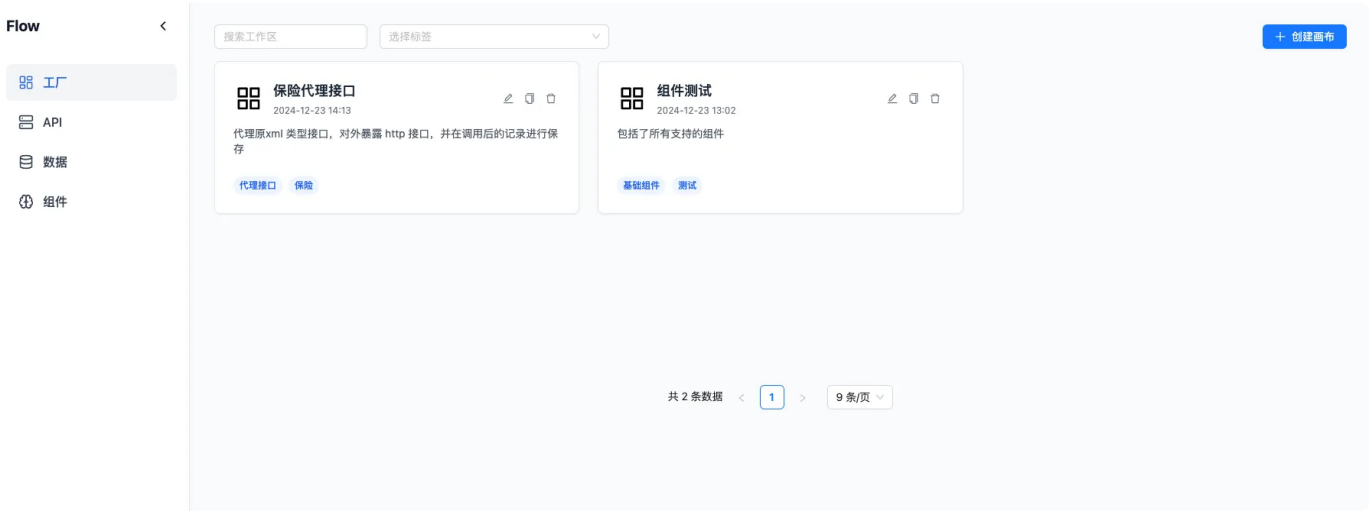


1.1.1. 数据源探活规则

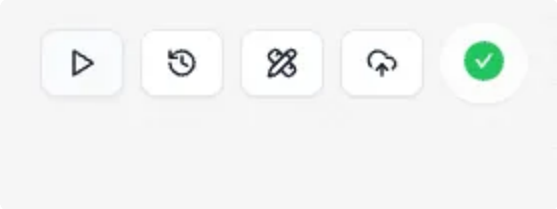
1. 每10分钟自动扫描一次数据库的连接状态【文件服务器不参与扫描】
2. 开关关闭后不继续扫描，清理数据源连接池 【规则引擎里的对应数据源配置也不可用】

1.2. 创建画布


创建画布
* 画布名称
保险代理接口
* 画布描述
代理原xml 类型接口，对外暴露 http 接口，并在调用后的记录进行保存
* 画布图标
应用
标签
代理接口
保险
取 消
确 认

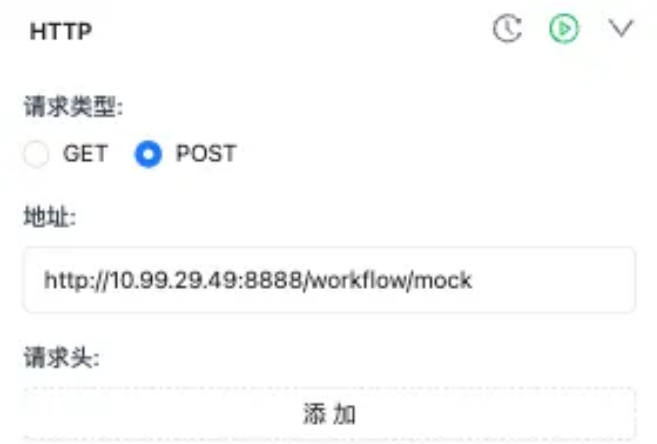

Flow
工厂
API
数据
组件
搜索工作区
选择标签
创建画布
保险代理接口
2024-12-23 14:13
代理原xml 类型接口，对外暴露 http 接口，并在调用后的记录进行保存
代理接口
保险
组件测试
2024-12-23 13:02
包括了所有支持的组件
基础组件
测试
共 2 条数据
1
9 条/页

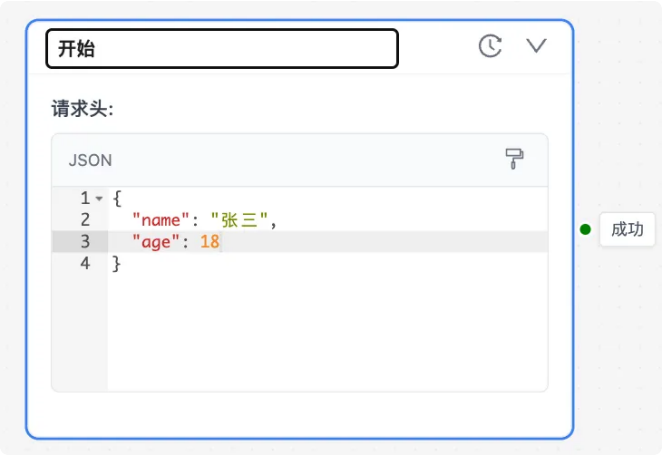
1.3. 画布功能



1. 全部运行
2. 画布运行记录
3. 格式化
4. 发布 API
5. 画布内容报错状态【运行时要保证这个标准时绿色的，否则还是运行的老数据】

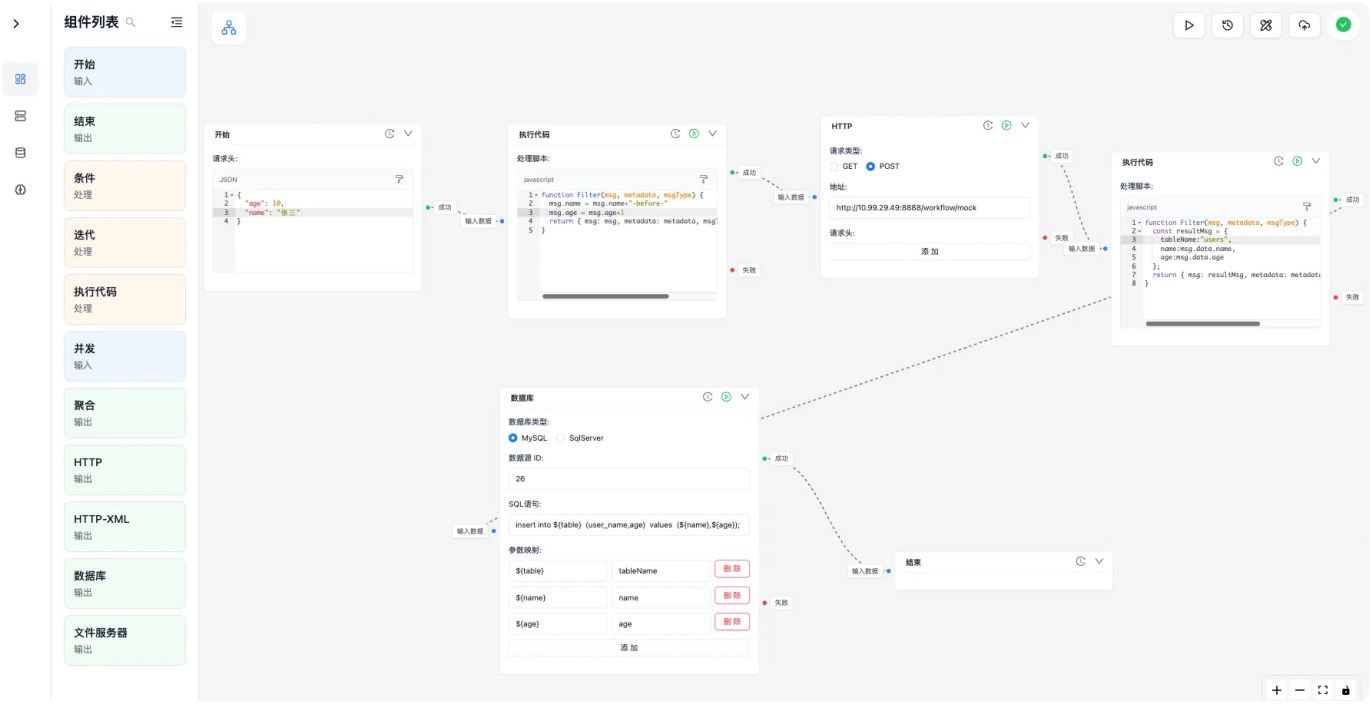
1.4. 组件功能





1. 运行记录
2. 从这里开始运行
3. 组件折叠
4. 编辑节点名称

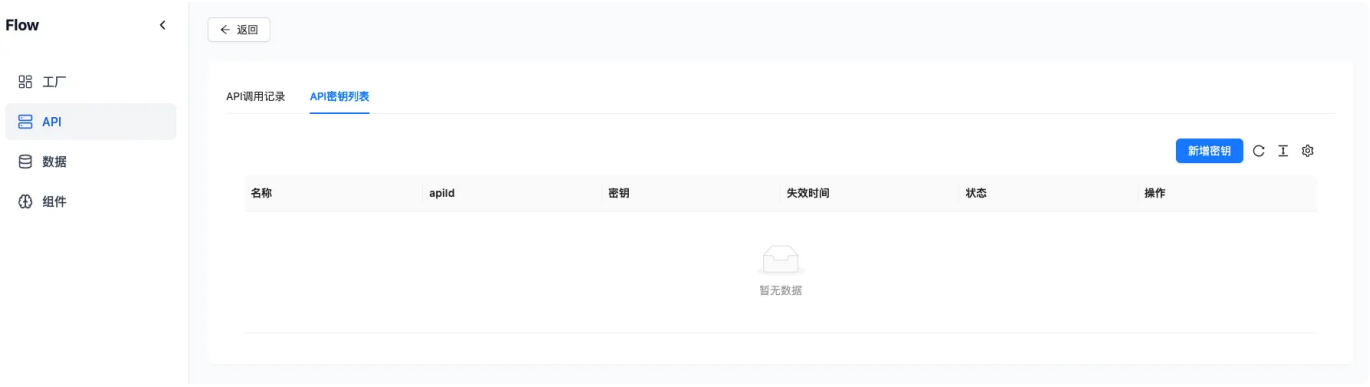
1.5. 编排规则



1.6. 发布API

1.7. 创建密钥

1.7.1. 点击 API 名称跳转到 API 详情页面



1.7.2. 新增密钥

1.8. 调用API

`POST http://10.99.29.49:8889/api/role/v1/ctkgflqflvkqiagjhqrg`

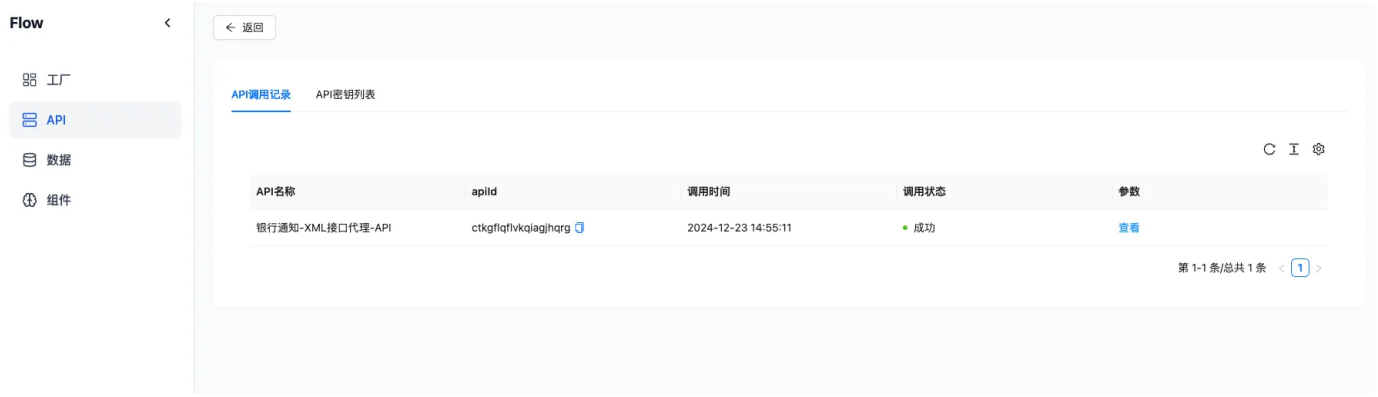
Authorization: Bearer 2fd2ad5b718f4a5a86106d8991a06707

四部分组成

1. API 服务地址
2. API 前缀
3. API ID
4. Authorization请求头 为创建的密钥

调用模板 Go

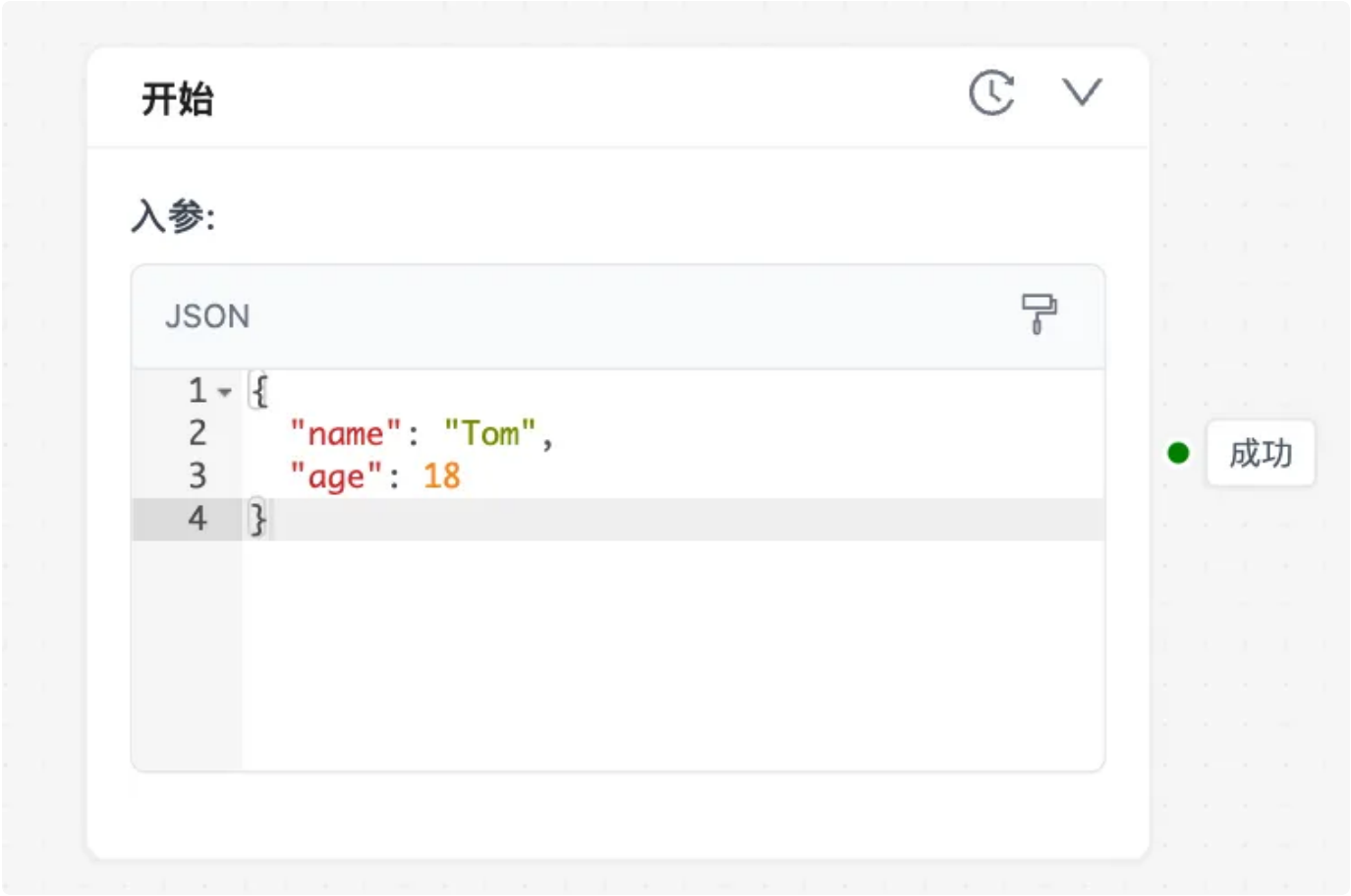
```
POST http://10.99.29.49:8889/api/role/v1/ctkgflqflvkqiagjhqrg
User-Agent: Apifox/1.0.0 (https://apifox.com)
Content-Type: application/json
Authorization: Bearer 2fd2ad5b718f4a5a86106d8991a06707

{
  "name": "测试名称",
  "age": 10
}
```



2. 组件介绍

> 核心规则: 组件的标准输入都是msg、msgType、metadata，个别组件除外
>
> msg：组件之间传递的消息
>
> msgType: 消息类型
>
> metadata：单次执行时的环境变量

2.1. 开始组件



功能说明: **"开始"** 节点是每个工作空间必备的预设节点，为后续工作流节点以及应用的正常流转提供必要的初始信息

使用场景: 画布、API 发布的必要节点

属性参数: json 输入框

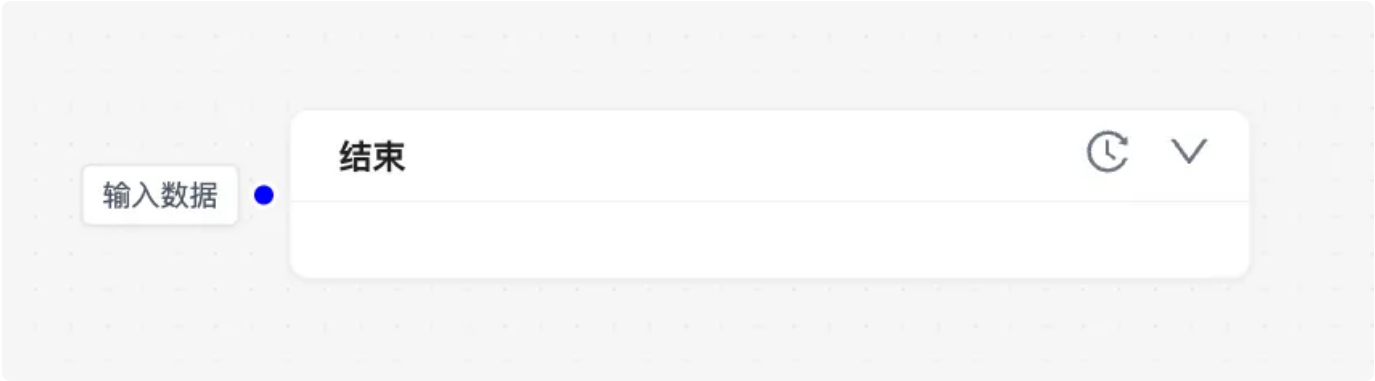
操作指南: 输入标准json

示例:

常见问题：

注意填写后点格式化 json 按钮，完成 json 检查，如果输入非法，后续组件读取数据会失败

2.2. 结束组件



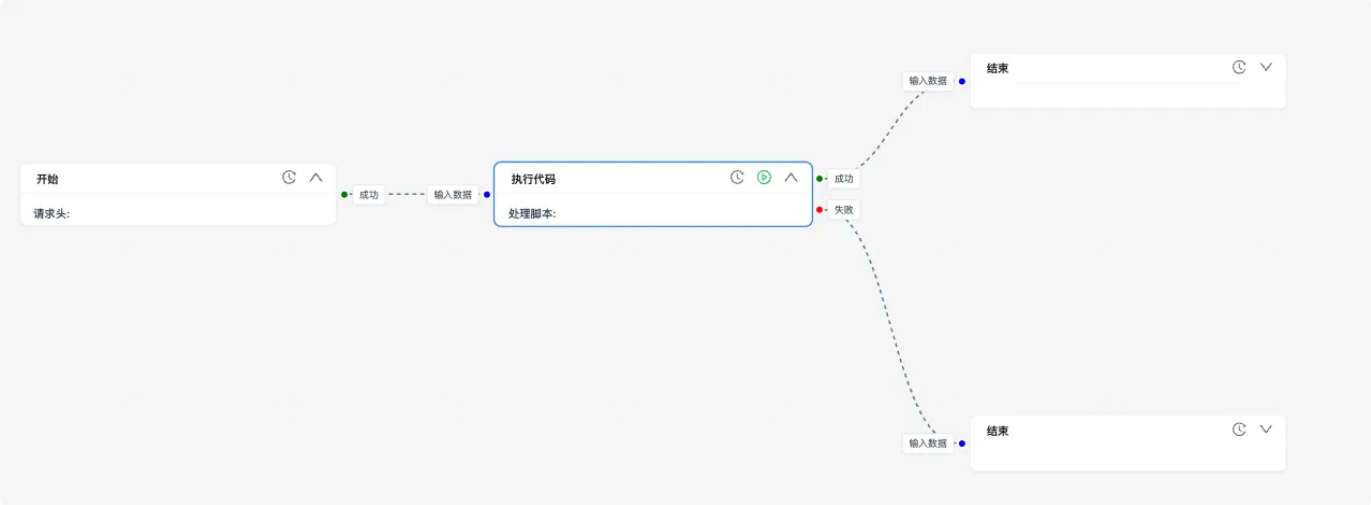
功能说明: "结束" 节点是每个工作空间必备的预设节点，工作流发布成 API 后，结束节点为请求结果的输出节点

使用场景: API 发布的必要节点

属性参数:

操作指南: 输出标准json

示例:

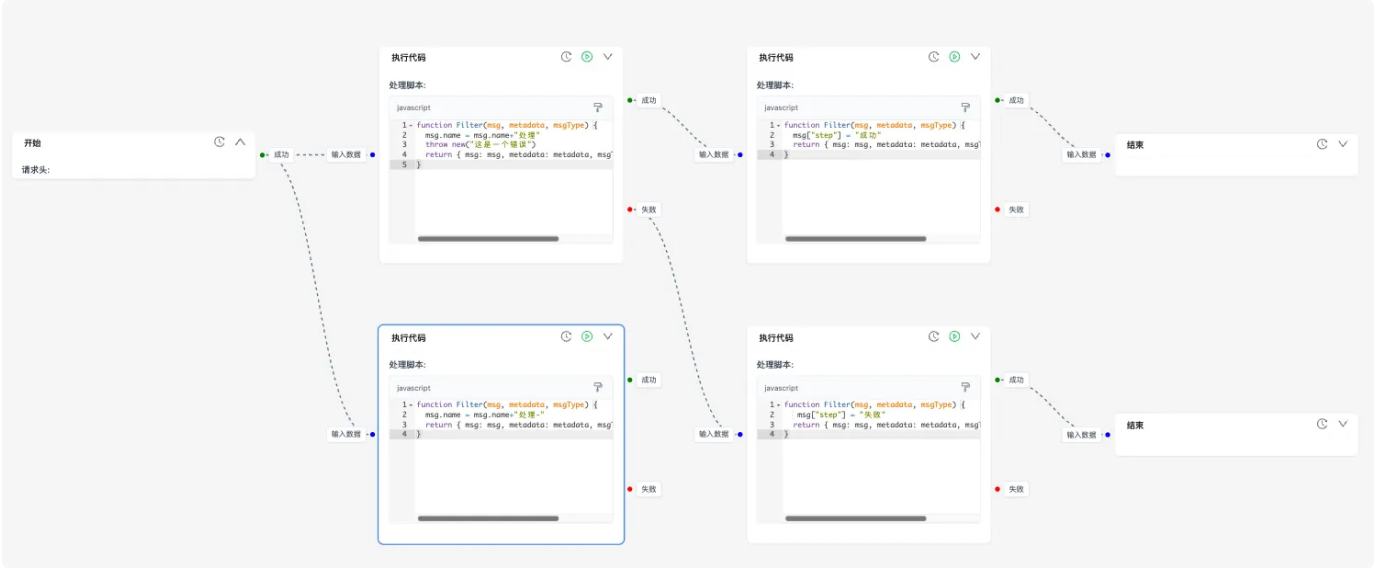


常见问题：

发布 API 后，注意如果存在多个叶子节点，则要在希望返回结果的节点后连接结束节点，否则调用接口后不会有任何返回值

如图：下面这种情况只会返回一个结果

```
POST http://127.0.0.1:8889/api/role/v1/ctl4m0t3sjtkr9cvi3eg
User-Agent: Apifox/1.0.0 (https://apifox.com)
Content-Type: application/json
Authorization: Bearer 051d029fcb6241deb9a7a32c41e07e39

{
    "name": "张三",
    "age": 10
}

HTTP/1.1 200 OK
Content-Type: application/json
Date: 2024 GMT
Content-Length: 42

{
  "age": 10,
  "name": "张三",
  "step": "失败"
}
```



2.3. 条件组件

功能说明: **"条件"** 节点根据判断条件将流程拆分成多个分支。

使用场景: 希望通过单条件或者多重条件判断来使

属性参数: 条件表达式，读取参数时前面要 msg[**组件直接传递信息**] 或者 metadata[**单次执行规则时的环境变量**]

操作指南:

条件判断是一段JavaScript脚本，支持ECMAScript 5.1(+) 语法规范

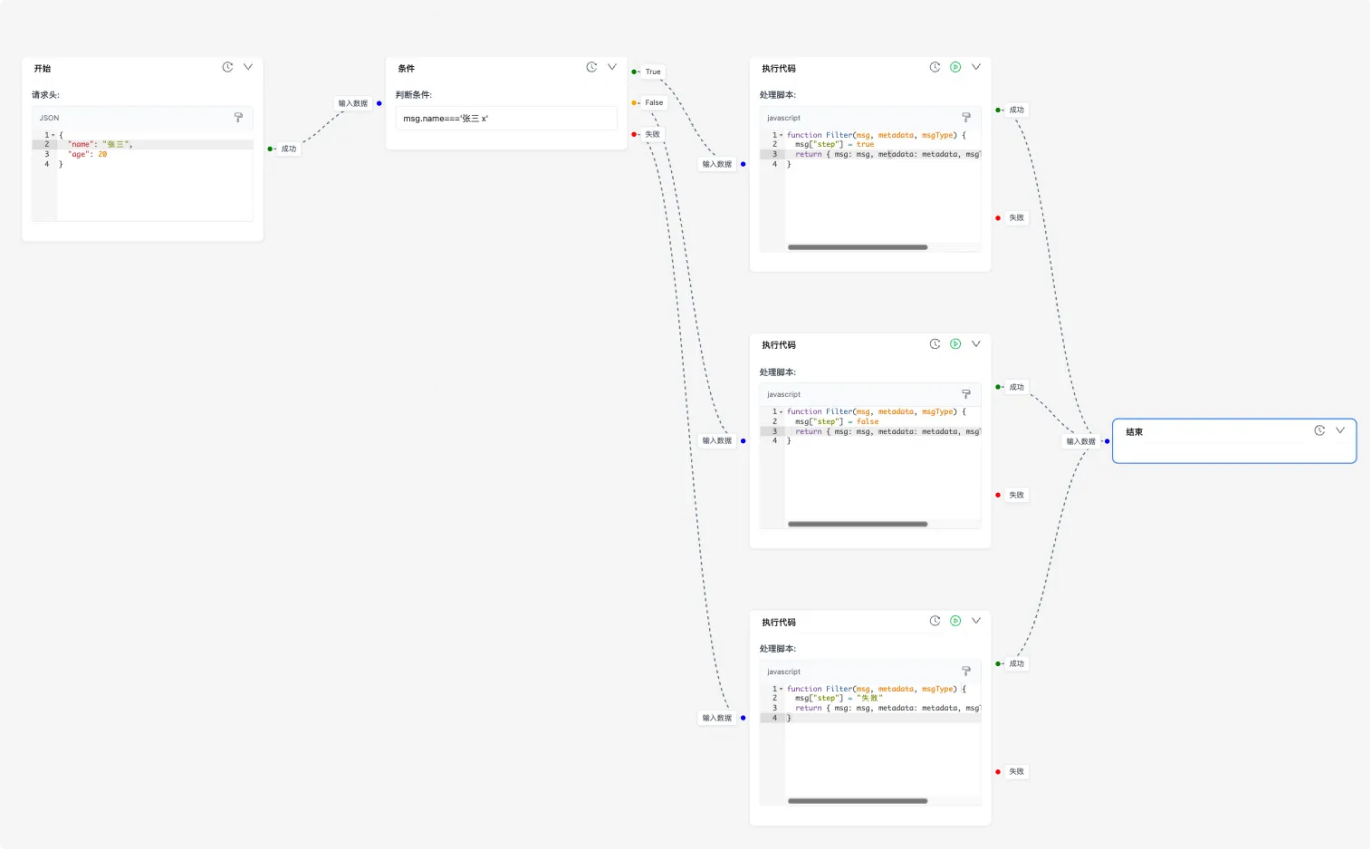
复杂表达式 Go

```
msg.name === '张三' || msg.age === 21
```

包含 Go

```
['张三','李四','王五'].some(element => element === msg.name);
```

示例:



常见问题：

什么情况下会走失败逻辑?

1. 表达式语法错误
2. 表达式中有命名错误
3. 表达式比较类型错误

2.4. 迭代组件

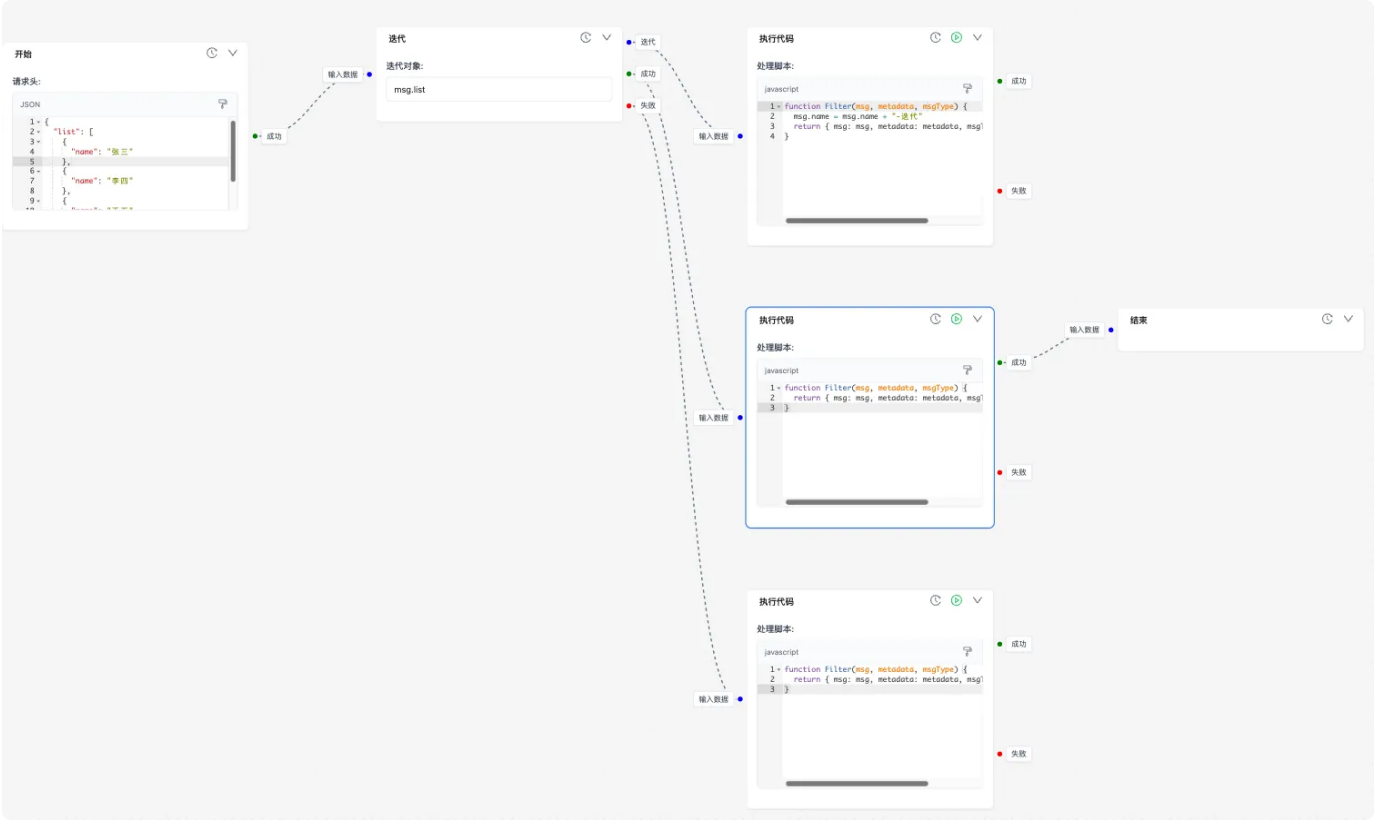
功能说明: "迭代" 节点依次执行迭代桩后相同的规则步骤，全部遍历完成后会走到成功后的逻辑，如果失败则走到失败的分支，可以理解为任务批处理器。

使用场景: 遍历数组和结构体

属性参数: 迭代字段，读取参数时前面要 msg[**组件直接传递信息**] 或者 metadata[**单次执行规则时的环境变量**]

操作指南:

示例:



常见问题:

迭代的结果汇总逻辑是什么?

会把迭代后的 msg 汇总到一个数组里面，在成功的输出 msg 会变成数组

什么情况下会失败?

1. 表达式所选字段不存在
2. 表达式不合法
3. 遍历执行失败(如迭代的过程中有错误)，消息会发送到失败分支
    a. 比如数组里一共 10 条数据，当遍历到第5条时报错，那么后续的 5 条就不会继续遍历，会直接走到迭代失败的分支
    b. 迭代失败分支的入参是数组中迭代失败的那条记录，而不是整个数组

2.5. 执行代码组件

功能说明: **"代码执行"** 节点使用JavaScript脚本对消息进行转换和处理

使用场景: 可以灵活地修改msg、metadata和msgType的内容，实现数据转换、格式转换、数据增强等功能

属性参数: JavaScript脚本

操作指南:

1. 当 dataType=JSON 时为 `json` 类型,可通过 `msg.field` 方式访问字段

```Go
function Filter(msg, metadata, msgType) {
  msg.name = msg.name+"处理"
  return { msg: msg, metadata: metadata, msgType: msgType };
}
```

2. 可以通过 throw new("这是一个错误")来使组件走到失败分支
3. 可以通过上一个组件的输出来查看组件中的脚本如何编写

上一个组件的输出 Go


```
{
  "metadata": {
    "_loopIndex": "2",
    "_loopItem": "{\"age\":20,\"name\":\"王五\"}",
    "relationType": "Success",
    "startTime": "2024-12-24 20:13:58",
    "traceId": "0f723959-0296-44e4-80f9-893d71bb9e96"
  },
  "msg": [
    {
      "age": 10,
      "name": "张三-迭代-"
    },
    {
      "age": 15,
      "name": "李四-迭代-"
    },
    {
      "age": 20,
      "name": "王五-迭代-"
    }
  ],
  "msgType": "CANVAS_MSG"
}
```


示例:

```
function Filter(msg, metadata, msgType) {
  // 用于聚合的结果对象
  const aggregatedData = {
    age: 0,
    names: [],
    count: 0,
  };

  // 遍历 msg 数组
  msg.forEach((item) => {
    const message = item.msg; // 获取 msg 对象
    const data = JSON.parse(message.data); // 获取具体的数据

    // 聚合年龄
    aggregatedData.age += data.age;

    // 收集名字
    aggregatedData.names.push(data.name);

    // 计数
    aggregatedData.count++;
  });

  // 计算平均年龄
  const averageAge =
    aggregatedData.count > 0 ? aggregatedData.age / aggregatedData.count
: 0;

  // 重新定义 msg 对象
  const resultMsg = {
    averageAge: averageAge,
    names: aggregatedData.names,
    totalCount: aggregatedData.count,
  };

  return { msg: resultMsg, metadata: metadata, msgType: msgType };
}

```

常见问题：

什么情况下会失败?

- 脚本语法错误
- 脚本执行异常
- 脚本执行超时
- 返回值格式错误

2.6. 并发组件

功能说明: "并发" 用于将消息流分成多个并行执行的路径，实现消息的并行处理。每个输出路径都会收到相同的消息副本，并可以独立执行不同的处理逻辑。

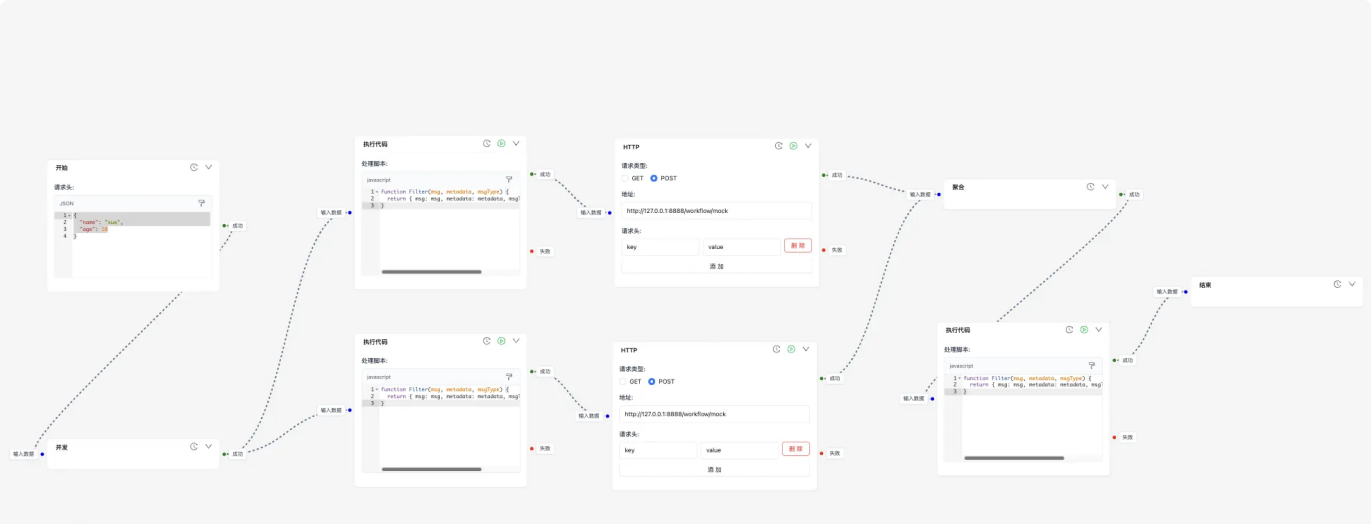
使用场景: 从多个数据源(如不同数据库)获取数据后合并、并行调用多个API后合并结果等

属性参数: 无

操作指南:

1. 该组件是纯路由组件，不会修改传入的 `msg` 、 `metadata` 和 `msgType` 内容。仅负责将消息复制并发送到多个输出路径
2. 通常和聚合组件成对出现

示例:



常见问题:

如果后面不连聚合组件会怎么样?

如果不连聚合的话，会全部异步执行，API 的输出是第一个先执行到结束节点的内容

2.7. 聚合组件

功能说明: "聚合" 节点用于汇聚并合并多个异步并行执行节点的结果

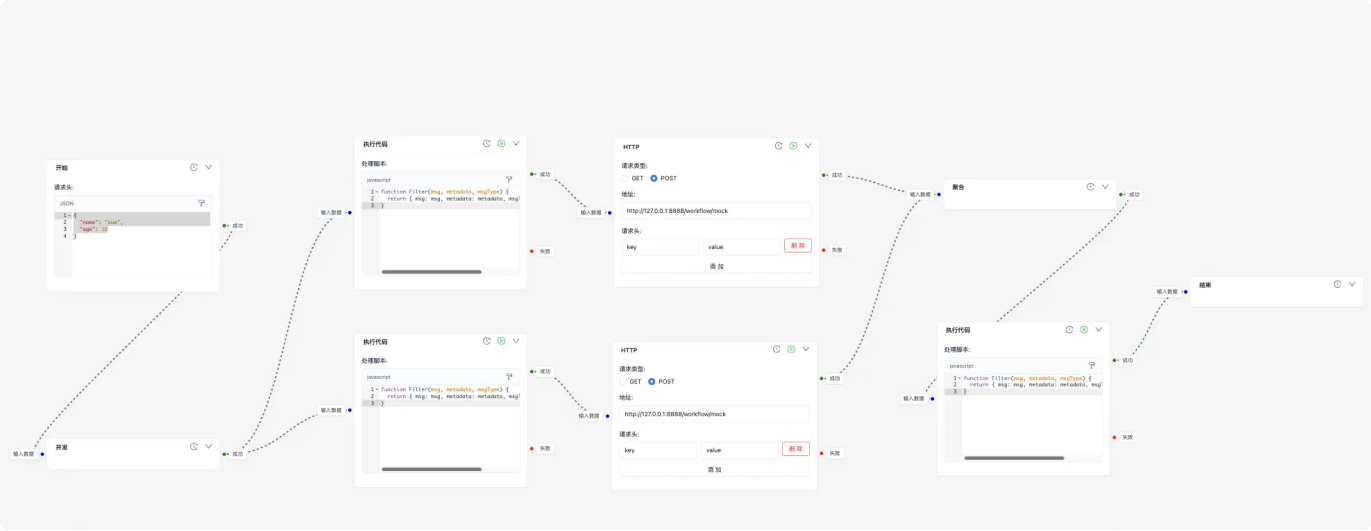
使用场景: 从多个数据源(如不同数据库)获取数据后合并、并行调用多个API后合并结果等

属性参数: 无

操作指南:

1. 聚合组件会等待所有前置异步节点执行完成
2. 合并所有节点的metadata，相同key时后执行的节点会覆盖先执行节点的值
3. 将所有节点处理后的消息封装成msg数组
4. 聚合组件的输入桩不要连接执行不到的链路，否则会一直等待⏳直到超时(10秒)

示例:



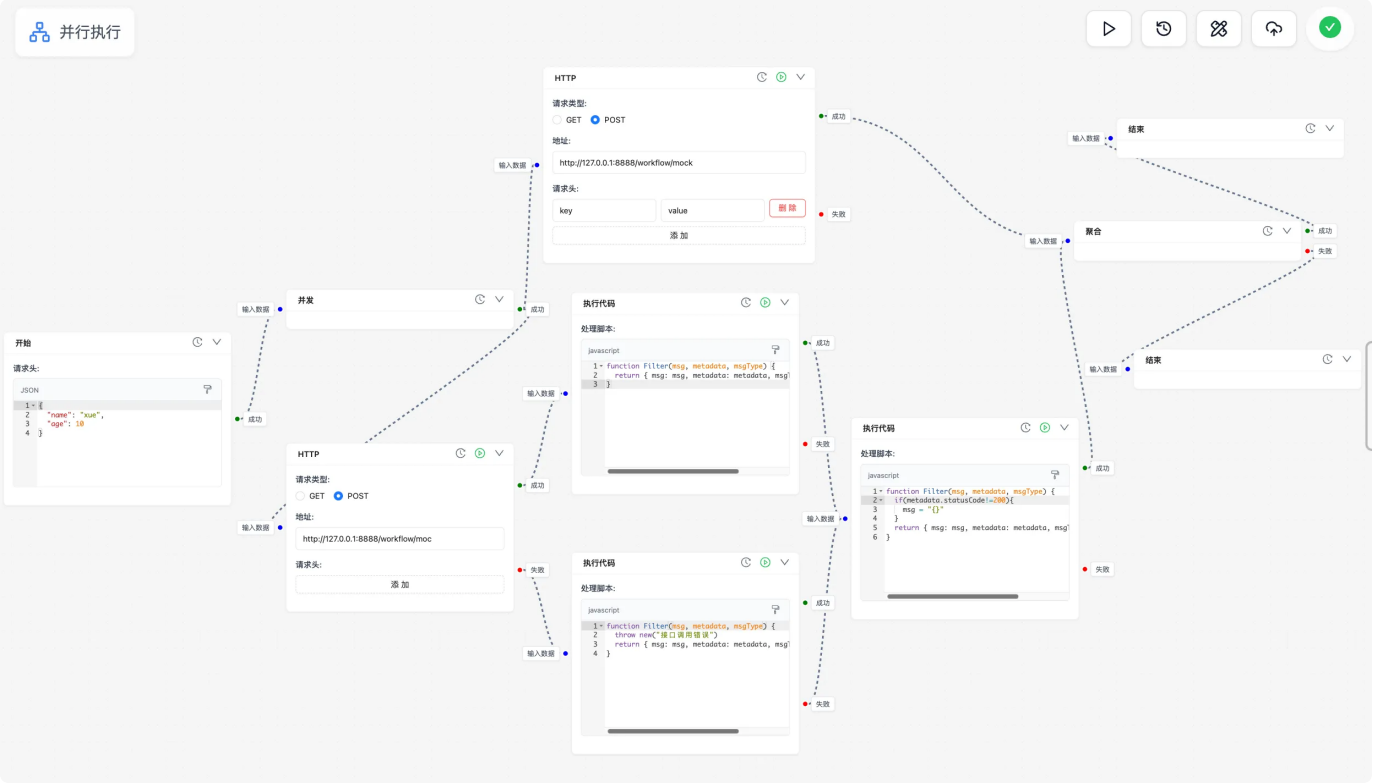
常见问题：

什么情况下会失败?

1. 执行超时 10s

如果涉及到接口请求失败的处理怎么办?

由于一个组件只能对下一个组件连接一条逻辑，所以需要在聚合组件之前，先把错误处理的逻辑都聚合到一个组件中



2.8. HTTP组件

功能说明: **"HTTP"** 节点用于调用外部REST API服务，支持常见的HTTP方法、自定义请求头

使用场景: 将msg作为请求体发送给目标服务,并将响应内容回填到msg中

属性参数:

1. 请求类型GET、POST
2. URL 支持变量读取 [msg或 metadata]
3. 请求头 支持变量读取 [msg或 metadata]
4. 默认超时时间 2s

执行成功时:

- msg.Data: 更新为HTTP响应体内容
- msg.Metadata.status: 响应状态描述
- msg.Metadata.statusCode: HTTP响应状态码

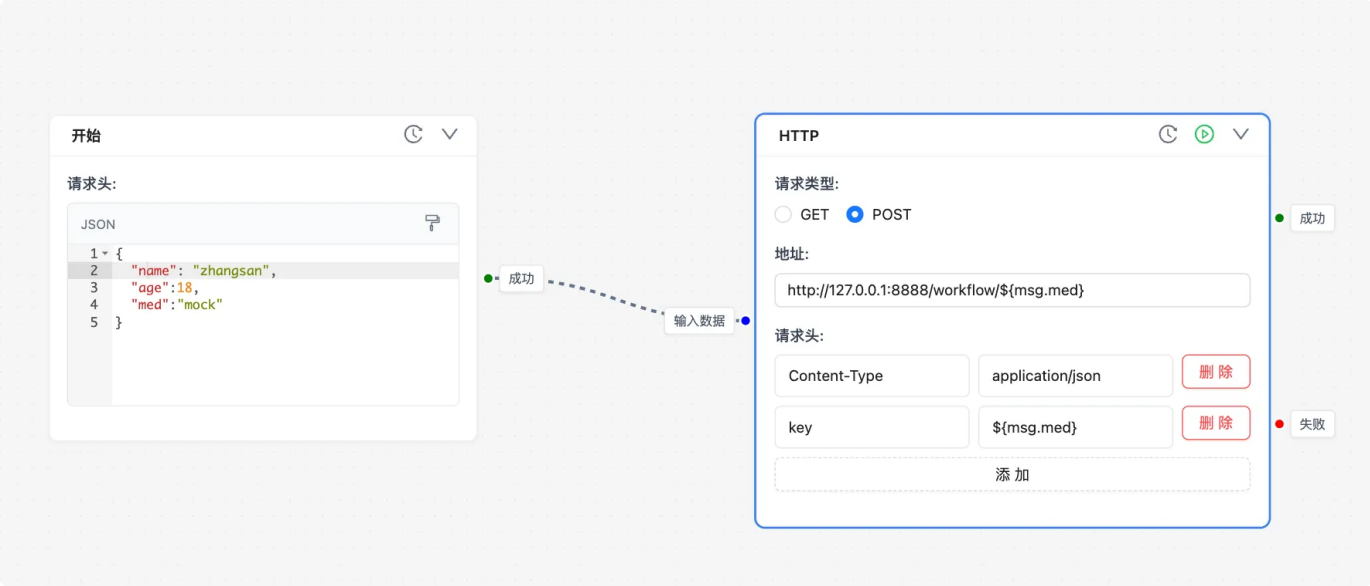
执行失败时:

- msg.Metadata.status: 错误状态描述
- msg.Metadata.statusCode: HTTP错误码
- msg.Metadata.errorBody: 错误响应内容

操作指南:

将msg作为请求体发送给目标服务,并将响应内容回填到msg中

示例:



请求后的输出 JSON

```
{
  "code": 0,
  "data": {
    "age": 28,
    "name": "zhangsanmock"
  },
  "msg": "OK"
}
```

常见问题：

什么情况下会失败?

1. 请求执行失败
2. 响应状态码非2xx
3. 请求超时
4. URL解析错误

2.9. HTTP–XML组件

功能说明: "HTTP–XML" 节点用于调用外部 XML 类型的 API服务，支持常见的HTTP方法、自定义请求头

使用场景: 代理 XML 类型接口转 json 类型

属性参数:

1. 请求类型GET、POST
2. URL 支持变量读取 [msg或 metadata]
3. 请求头 支持变量读取 [msg或 metadata]

4. 请求参数，按照指定协议拼接

soap 协议 JSON

```
<?xml version="1.0" encoding="UTF-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
  <Add xmlns="http://tempuri.org/">
  <intA>${intA}</intA>
  <intB>${intB}</intB>
</Add>
</soap:Body>
</soap:Envelope>
```

5. 默认超时时间 2s

执行成功时:

- msg.Data: 更新为HTTP响应体内容
- msg.Metadata.status: 响应状态描述
- msg.Metadata.statusCode: HTTP响应状态码

执行失败时:

- msg.Metadata.status: 错误状态描述
- msg.Metadata.statusCode: HTTP错误码
- msg.Metadata.errorBody: 错误响应内容

操作指南:

1. 根据变量替换后，将请求参数作为请求体发送给目标服务并将响应内容回填到msg中，请求结果会转成json
2. 注意请求头 xml 有多种类型 text/xml、application/xml，默认 application/xml

示例:

http://www.dneonline.com/calculator.asmx

入参 JSON

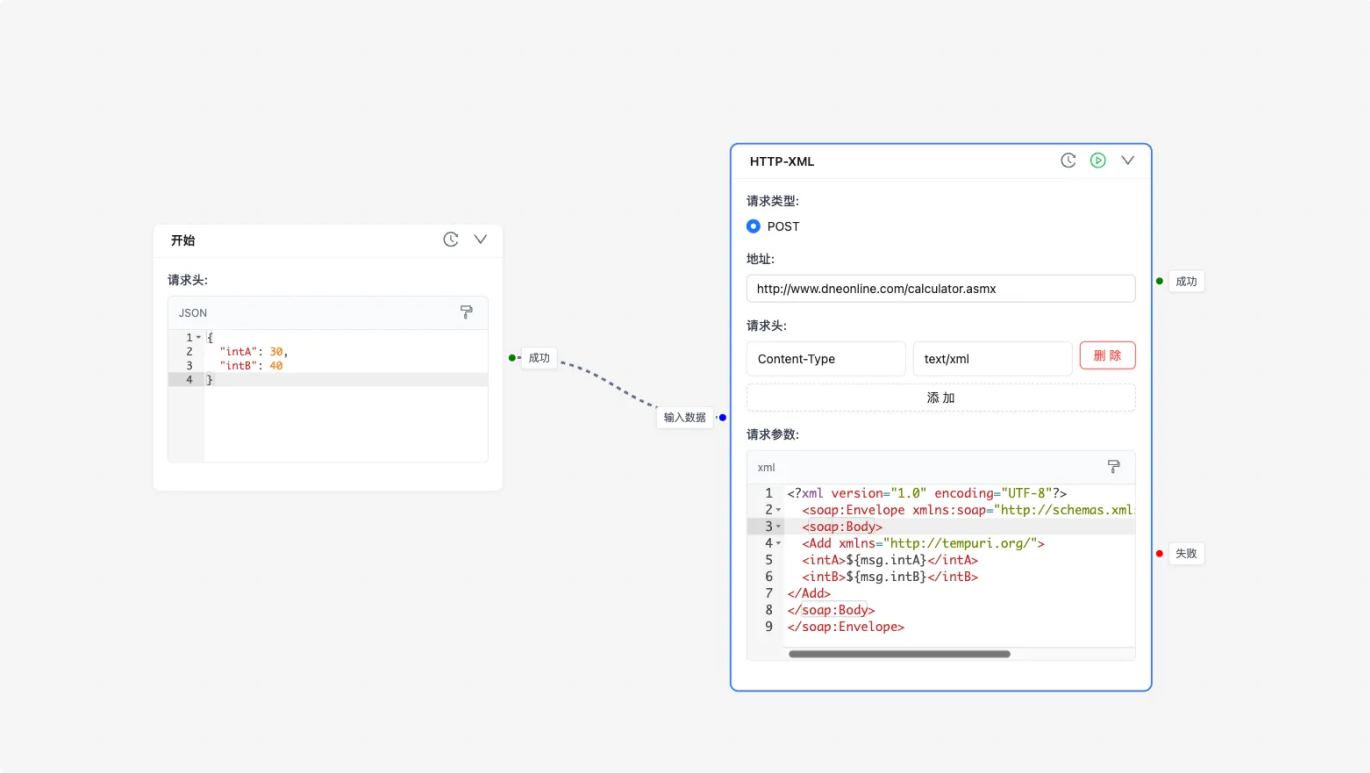
```json
{
  "intA": 30,
  "intB": 40
}
```

请求参数 JSON

```
<?xml version="1.0" encoding="UTF-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
  <Add xmlns="http://tempuri.org/">
  <intA>${intA}</intA>
  <intB>${intB}</intB>
</Add>
</soap:Body>
</soap:Envelope>
```

返回结果 JSON

```json
{
    "Envelope": {
      "-soap": "http://schemas.xmlsoap.org/soap/envelope/",
      "-xsd": "http://www.w3.org/2001/XMLSchema",
      "-xsi": "http://www.w3.org/2001/XMLSchema-instance",
      "Body": {
        "AddResponse": {
          "-xmlns": "http://tempuri.org/",
          "AddResult": "70"
        }
      }
    }
  }
```



常见问题：

请求不通怎么办?

1. 注意请求头是否与目标接口一致
2. 注意 xml 解析协议是否与目标接口一致

2.10. 数据库组件

功能说明: "数据库" 节点通过标准sql对数据库进行增删修改查操作。内置支持 `mysql` 和 `oracle` 数据库。

使用场景: 可以执行SQL查询、更新、插入和删除操作

属性参数:

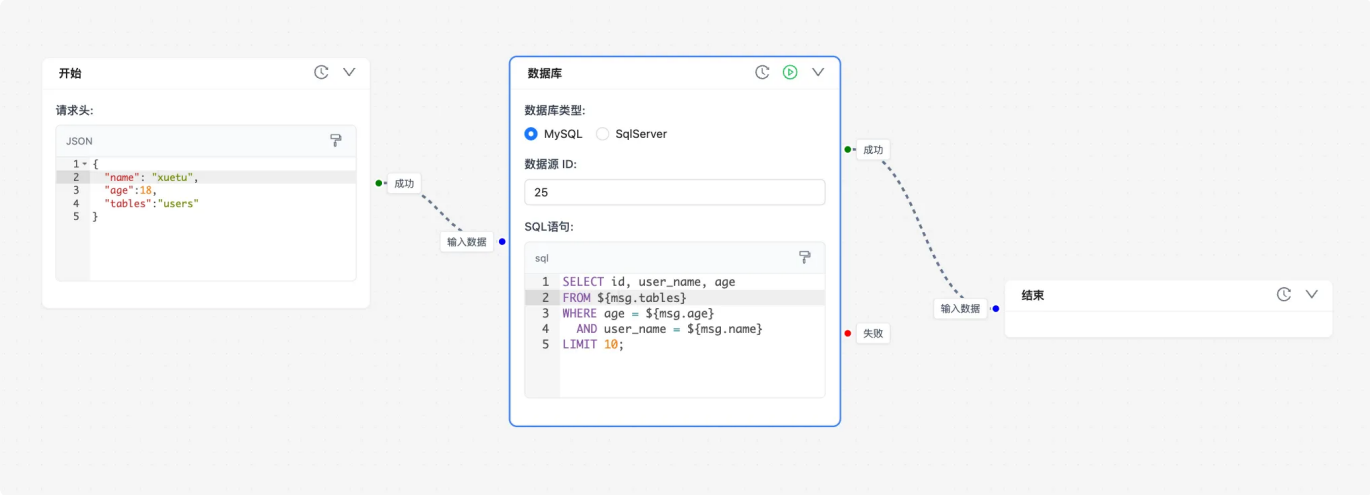
1. 数据源类型
2. 数据源 ID
    a. 在数据目录下创建

3. SQL语句，支持变量替换

操作指南:

1. 支持变量读取 [msg 或 metadata]
2. 数据库组件标准输出都是数组格式
3. 可以通过 as 来对输出字段进行重命名

示例:



输出 JSON

```
[
    {
      "age": 18,
      "id": 24334,
      "user_name": "xuetu"
    },
    {
      "age": 18,
      "id": 24335,
      "user_name": "xuetu"
    }
  ]
```

常见问题:

2.11. 文件服务器组件

功能说明: "文件服务器" 节点支持 FTP、SFTP 协议，支持文件下载、上传、删除功能

使用场景: 从 A 文件服务器下载文件，上传到 B 服务器，然后删除 A 服务器对应文件内容

属性参数:

路径统一为 / 开头

下载

1. 文件系统类型
2. 操作模式 上传
3. 数据源 ID
4. 路径–下载文件的具体地址 支持变量替换

上传

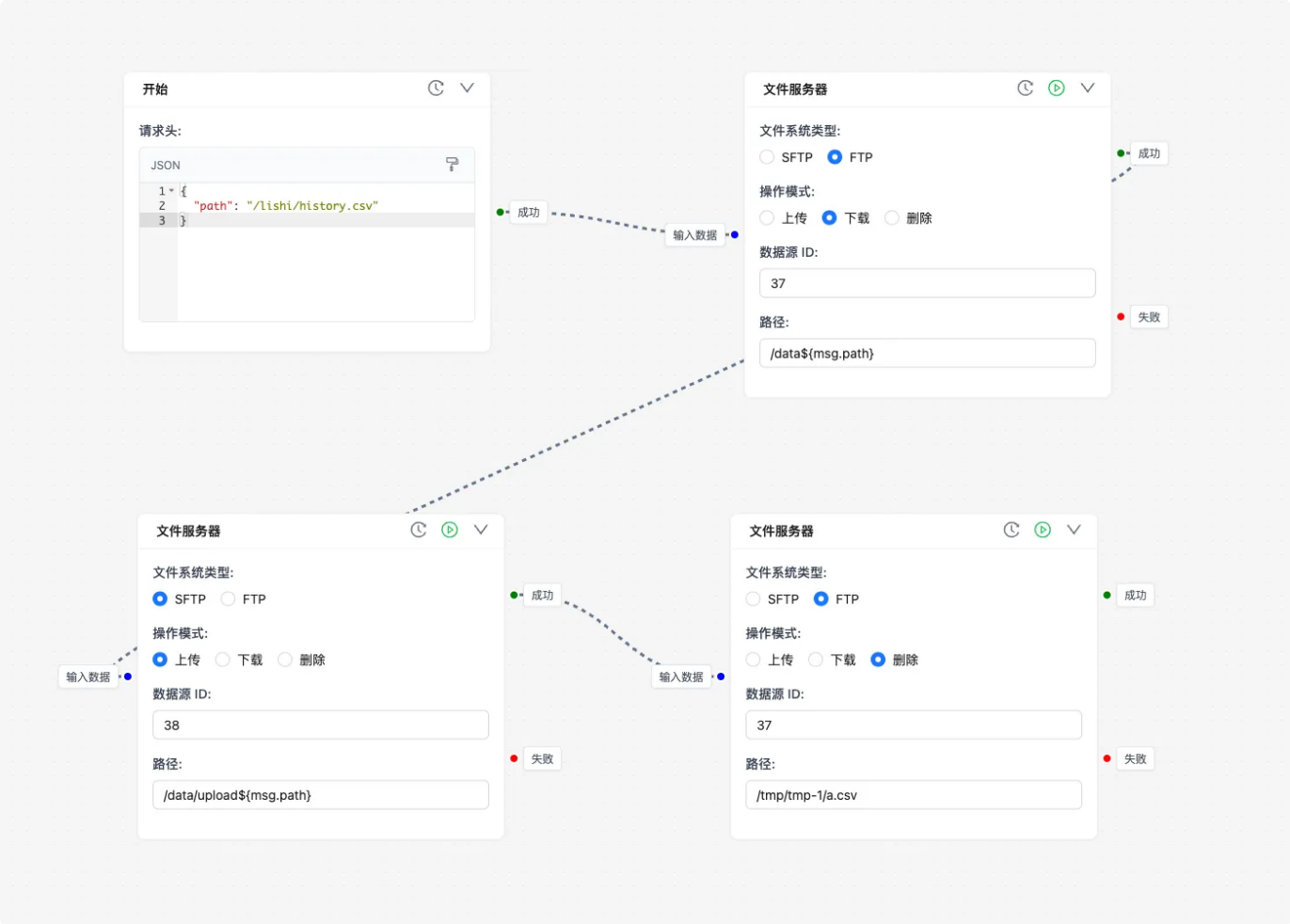
1. 文件系统类型
2. 操作模式 下载
3. 数据源 ID
4. 路径–上传文件服务器文件的具体地址，将上一个下载组件输出的 msg.tmpPath 临时文件上传到这里

删除

1. 文件系统类型
2. 操作模式 删除
3. 数据源 ID
4. 路径–文件服务器中要删除文件的地址

操作指南:

示例:



常见问题：

报错 550 Delete operation failed.

目录没有权限，chmod a+w 文件

3. 环境变量

k8s 环境变量配置 Go

```
- name: PORT
  value: '8888'
- name: API_PORT
  value: '8889'
- name: RULE_SERVER_TRACE
  value: 'true'
- name: RULE_SERVER_LIMIT_SIZE
  value: '4'
- name: LOG_MODE
  value: file
- name: LOG_LEVEL
  value: info
- name: DSN
  value: >-
    wkflow:Gaia@123@tcp(ido-mysql-headless:3306)/wkflow?charset=utf8mb4&parseTime=True&loc=Local
- name: REDIS_HOST
  value: 'ido-redis-headless:6379'
- name: REDIS_PASSWORD
  value: redis123
- name: REDIS_DB
  value: '10'
```

序号	配置	描述
1	PORT	后台服务端口
2	API_PORT	发布的API调用端口
3	RULE_SERVER_TRACE	API服务链路追踪 开启后方便排错 关闭后提高性能
4	RULE_SERVER_LIMIT_SIZE	API服务请求体大小限制单位 M 兆
5	LOG_MODE	日志模式(保存文件)
6	LOG_LEVEL	日志级别默认 info

7	LOG_LEVEL	日志级别
8	DSN	mysql 数据库配置
9	REDIS_HOST	redis 配置
10	REDIS_PASSWORD	redis 配置
11	REDIS_DB	redis 配置