

# 数据挖掘-服务

在进行机器学习实验的过程中，为了简化和加速模型的构建、训练和部署，使用自动化机器学习功能更快地识别合适的算法并优化超参数。

- 服务 workflow 示例
- 部署服务
  - 概述
  - 部署方法
    - 数据输入
    - 部署方式
    - 应用服务
  - 示例
- 服务监控
- 服务调用示例

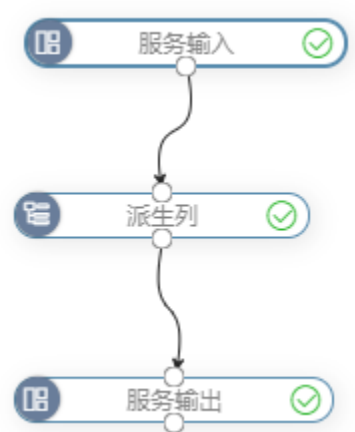
## 服务 workflow 示例

服务 workflow 是将数据挖掘以服务的方式进行发布。

要求：输入层必须是“服务输入”节点，输出层必须是“服务输出”节点。

通过部署服务后，通常用于数据预测的应用。“服务输入”的处理层可以实现数据来源于其它接口。

下图是一个简单的服务 workflow 示例：



在这个 workflow 中，实现的是对输入的数据进行派生列处理后输出。

## 部署服务

### 概述

服务部署指的是当用户用实验训练好了一个预测模型，并将此预测模型部署成一个 web 的服务。当前发布的 web 服务就可以作为预测模型应用于类似场景的预测，当用户要预测一个新样本，只需要把样本特征信息用 `restful api` 传给服务，服务就可以依据训练好的预测模型返回预测结果。

前提条件：服务 workflow 完成搭建且执行成功。workflow 中每个节点旁边显示绿色的勾即表示该节点是执行成功的。

### 部署方法

#### 数据输入

服务输入中的数据支持“手工输入”和“选择节点数据”两种方式：

设置输入数据

手工输入

选择节点数据

输入关键字搜索

特征选择

预测

拆分

服务输出

[{"id":327,"area":1.29093,"activity":256.521,"lv\_total":1,"near\_dista":9311.43,"Type\_CN":"专营型商业区"}, {"id":991,"area":1.1085,"activity":92.0348,"lv\_total":2,"near\_dista":14157.8,"Type\_CN":"购物中心型商业区"}, {"id":176,"area":1.08434,"activity":241.87,"lv\_total":1,"near\_dista":7890.56,"Type\_CN":"专营型商业区"}, {"id":718,"area":1.07388,"activity":131.676,"lv\_total":2,"near\_dista":3584.83,"Type\_CN":"购物中心型商业区"}, {"id":24,"area":0.956711,"activity":100.659,"lv\_total":2,"near\_dista":27959,"Type\_CN":"购物中心型商业区"}, {"id":641,"area":0.902957,"activity":102.756,"lv\_total":2,"near\_dista":2285.33,"Type\_CN":"专营型商业区"}, {"id":582,"area":0.900749,"activity":322.326,"lv\_total":1,"near\_dista":811.803,"Type\_CN":"购物中心型商业区"}, {"id":412,"area":0.897021,"activity":187.591,"lv\_total":1,"near\_dista":10741.8,"Type\_CN":"专营型商业区"}, {"id":572,"area":0.822336,"activity":228.195,"lv\_total":1,"near\_dista":4120.3,"Type\_CN":"购物中心型商业区"}, {"id":254,"area":0.78733,"activity":69.1679,"lv\_total":2,"near\_dista":5488.47,"Type\_CN":"专营型商业区"}, {"id":759,"area":0.786447,"activity":92.0876,"lv\_total":2,"near\_dista":6940.78,"Type\_CN":"购物中心型商业区"}, {"id":690,"area":0.66808,"activity":118.871,"lv\_total":2,"near\_dista":2995,"Type\_CN":"购物中心型商业区"}, {"id":185,"area":0.611637,"activity":48.9516,"lv\_total":2,"near\_dista":6392.31,"Type\_CN":"购物中心型商业区"}, {"id":797,"area":0.611319,"activity":143.611,"lv\_total":1,"near\_dista":6984.32,"Type\_CN":"专营型商业区"}, {"id":443,"area":0.61074,"activity":165.093,"lv\_total":1,"near\_dista":701.685,"Type\_CN":"购物中心型商业区"}, {"id":1060,"area":0.60593,"activity":55.2552,"lv\_total":2,"near\_dista":4456.44,"Type\_CN":"购物中心型商业区"}, {"id":1007,"area":0.570801,"activity":51.2594,"lv\_total":2,"near\_dista":17546.6,"Type\_CN":"购物中心型商业区"}, {"id":25,"area":0.564122,"activity":100.818,"lv\_total":2,"near\_dista":18563,"Type\_CN":"专营型商业区"}, {"id":677,"area":0.550328,"activity":116.418,"lv\_total":2,"near\_dista":4773.78,"Type\_CN":"购物中心型商业区"}, {"id":601,"area":0.543132,"activity":139.786,"lv\_total":1,"near\_dista":3311.66,"Type\_CN":"购物中心型商业区"}, {"id":691,"area":0.542341,"activity":53.4881,"lv\_total":2,"near\_dista":3520.85,"Type\_CN":"综合型商业区"}, {"id":905,"area":0.536716,"activity":39.0215,"lv\_total":2,"near\_dista":9692.97,"Type\_CN":"专营型商业区"}, {"id":892,"area":0.531509,"activity":193.143,"lv\_total":1,"near\_dista":10513.7,"Type\_CN":"购物中心型商业区"}, {"id":525,"area":0.529527,"activity":145.826,"lv\_total":1,"near\_dista":6102.97,"Type\_CN":"购物中心型商业区"}]

确定

取消

- 手工输入：是手工输入json列数据。
- 选择节点数据：是选择运行通过后的某节点输出结果数据。

部署方式

部署服务包括以下三种方式：

名称	说明
新增服务	<div>输入服务名称新增一个服务：</div> <div><div><div>部署服务</div><div><div>新增服务</div><div>更新服务</div><div>灰度部署</div></div><div><div>服务名称 *</div><div></div></div><div><div>确定</div><div>取消</div></div></div></div> <div>一个服务实验可以生成多个服务。</div>

更新服务

选择一个原服务更新其实验节点：

灰度部署

灰度部署用于使多版本模型并行运行，并保持多版本模型运行的结果，通过结果比较灰度测试的模型版本的准确性和稳定性。



- 灰度测试：又名A/B测试、灰度发布，一种在黑白之间发布平滑过渡的方式。可以对其执行A/B测试，一些用户继续使用产品功能A，一些用户开始使用产品功能B，如果用户不反对B，则逐渐扩大范围并迁移所有用户到B来。

灰度测试可以确保整个系统的稳定性，并且可以在初始灰度级找到并调整问题以确保其影响度。

- 灰度部署是基于某个服务（主服务/A服务），创建一个新的服务（B服务），此服务与主服务按照“加权采样”进行分流处理。

例如：A服务流量设置60，则B服务流量为40，将会有40%的概率调用B服务，将会有60%的概率执行A服务（主服务）。生成的B服务名称是以A服务名称加此时的时间戳，一个服务只能产生一个B服务。

在灰度部署中，已经生成过灰度服务的服务将不能被选择。

- 关联服务：基于当前服务再创建一个关联的服务；
- 新增服务流量/关联服务流量：用于分流处理中分配服务的流量，流量之和为100%，即设置新增服务流量值为50，将会有50%的流量分配给新增服务，50%的流量分配给关联服务；

生成的服务将会在服务管理中显示，如图灰度部署生成两个服务，上面的为A服务，下面的为B服务。

实验管理

服务管理

模型管理

模型自主学习

模型性能预测

自定义模块管理

部署服务

部署服务-1630291...

便捷简单的对服务进行监控，随时了解服务运营状态

全部 4

应用 4

异常 0

已下线 0

刷新

名称	描述	修改时间	状态	流量控制	常用操作
部署服务		2021-08-30 10:41:32	应用	30%	
部署服务-16302914899...		2021-08-30 10:41:32	应用	70%	
服务-名称		2021-08-27 16:35:00	应用		
fw1		2021-08-27 16:30:39	应用		

其中“流量控制”列显示的是对应服务的流量，点击对应服务的流量值，可进行对应A/B服务的流量配置。

流量控制[部署服务]

当前服务名称

部署服务

当前服务流量

30

^

v

?

关联服务名称

部署服务-1630291489985

关联服务流量

70

^

v

?

确定(O)

取消(C)

## 应用服务

目前服务工作流的应用通常为：对于已创建好的分类模型、回归模型、聚类模型，我们将新的数据输入后，通过模型计算预测出结果。

选择部署服务方式或成功部署服务后，可进行服务的应用。

应用入口：在“服务监控”界面的服务列表中，双击“名称”列中的服务，进入到“服务配置”界面，再单击 **服务测试** 页签，显示“服务测试”界面：

服务配置

服务测试

服务ID18a8aefa3017b86ba86ba455c017b95d2cc4f0042

测试数据

```
[{"data": [{"animal_name": "aardvark", "hair": 1, "feathers": 0, "eggs": 0, "milk": 1, "airborne": 0, "aquatic": 0, "predator": 1, "toothed": 1, "backbone": 1, "breathes": 1, "venomous": 0, "fins": 0, "legs": 4, "tail": 0, "domestic": 0, "catsize": 1, "type": 1}, {"animal_name": "antelope", "hair": 1, "feathers": 0, "eggs": 0, "milk": 1, "airborne": 0, "aquatic": 0, "predator": 0}]}]
```

示例: [{"data":[{"id":1,"name":"cl","age":24}]}]

测试结果

测试

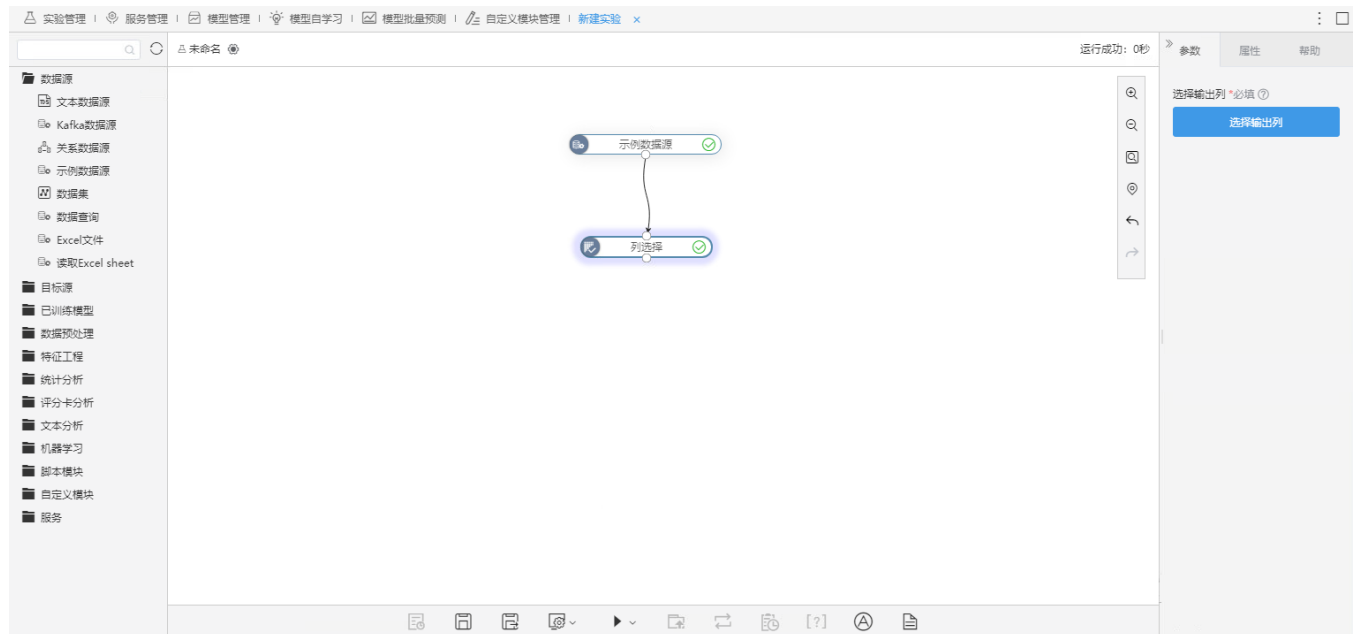
在“测试数据”文本框中按照示例填写数据，单击 **测试** 按钮，这些数据进入到当前服务部署的模型中执行后显示测试结果。



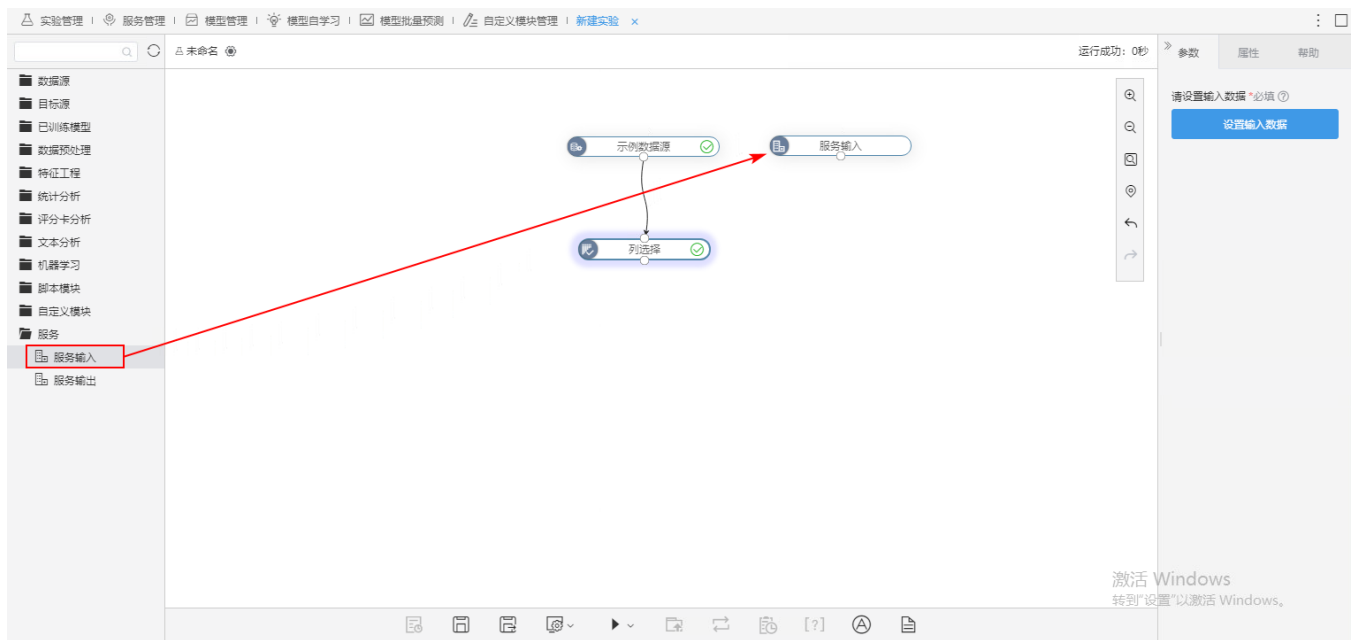
对于灰度部署，测试数据将按照流量分配选择A服务、B服务中的一列进行运行，返回最后的测试结果，测试结果即算法选择服务输出的结果。

## 示例

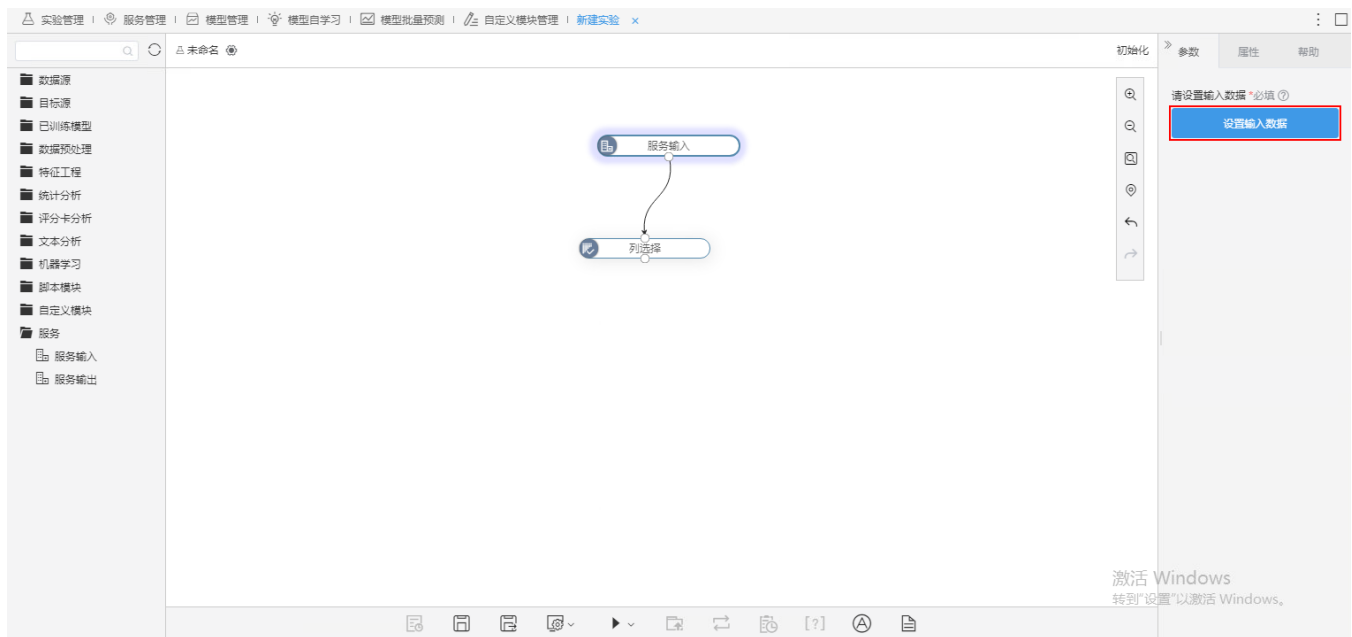
1、通过数据源或数据预处理定义实验模型，并运行该实验模型。



2、拖拽“服务输入”节点到画布，并选择节点，该节点的数据结果是用于服务发布的节点。



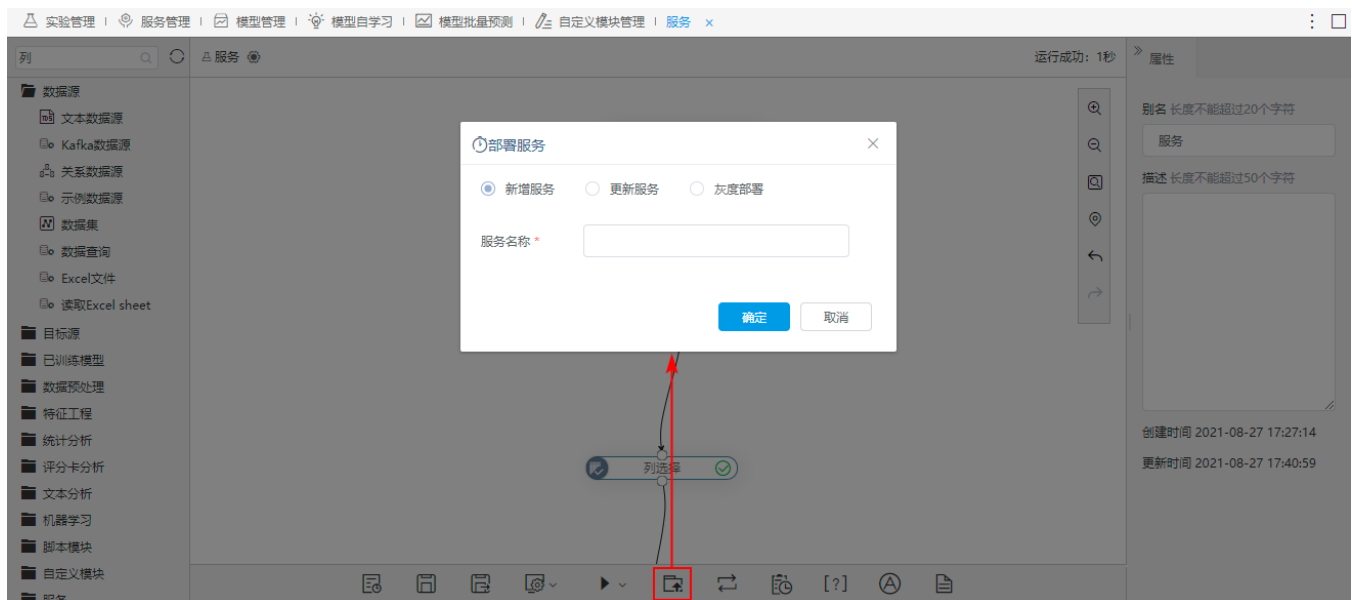
3、将“服务输入”节点在实验模型中替换所选的节点，并将所选节点前的所有节点删除，点击 **设置输入数据** 按钮。



输入数据默认为“选择节点数据”。



4、拖拽“服务输出”节点到流程节点最末，流程执行通过后单击工具栏的 **部署服务** 按钮。



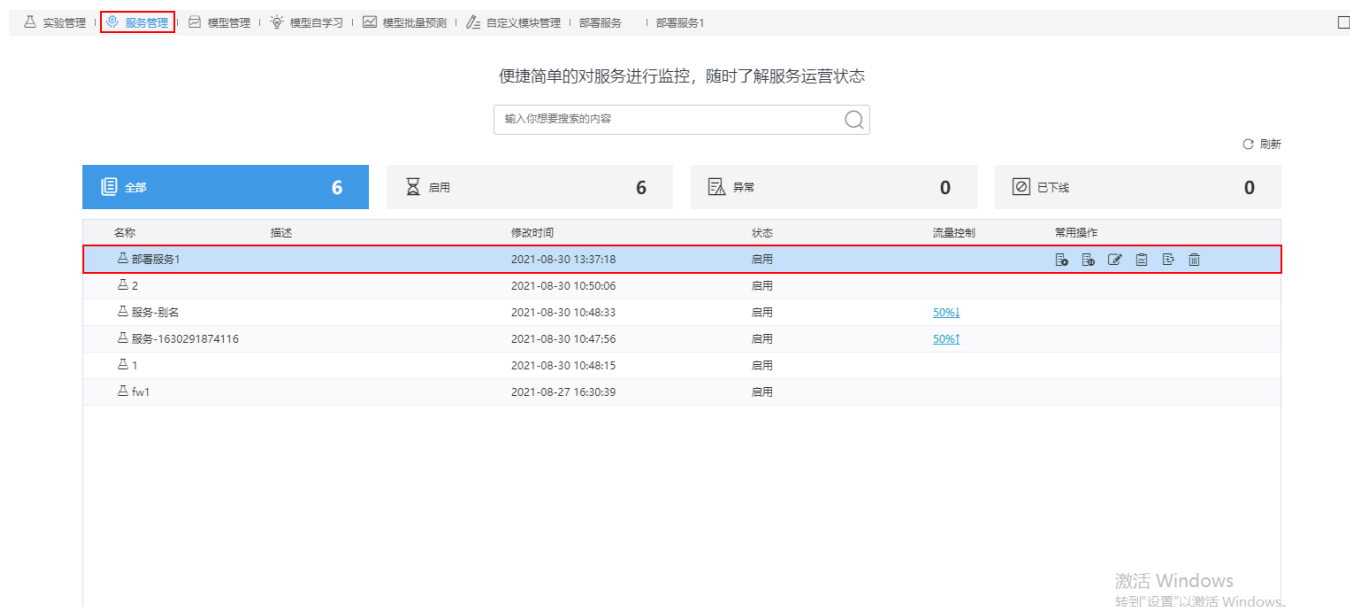
5、选择新增服务并输入服务名称。



6、选择“服务测试”页面，点击 **测试** 按钮对服务进行应用。



7、选择服务管理界面，点击 **刷新** 按钮，可对新建的服务进行启用、下线、编辑等操作，详情请参考 [服务监控](#) 。

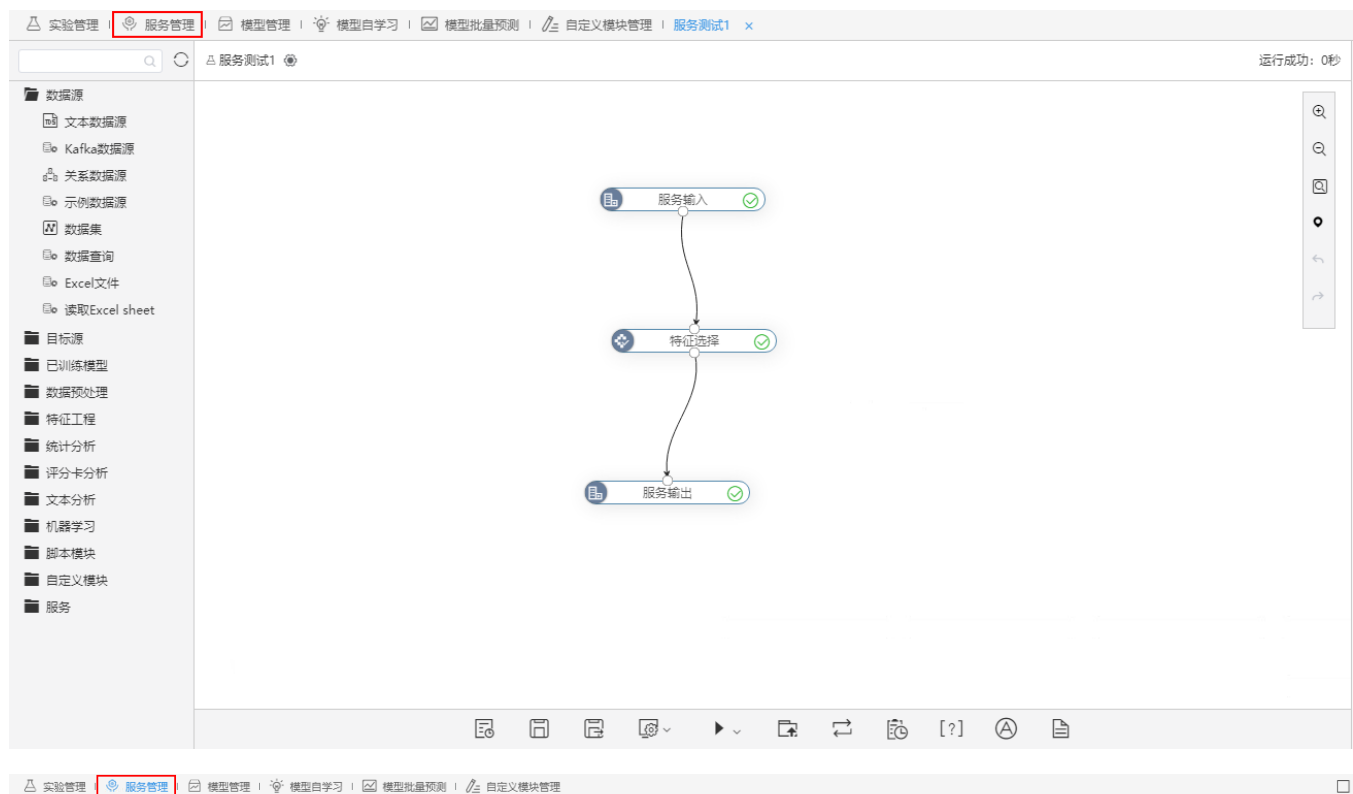


## 服务监控

服务监控是指对所有已经搭建的服务工作流程进行管理。

功能入口：在“数据挖掘”界面中选择 **服务管理** 页签，显示“服务管理”界面，如下图所示：





便捷简单的对服务进行监控，随时了解服务运营状态

输入你想要搜索的内容

功能区

刷新

全部2

启用

状态栏2

异常0

已下线0

名称	描述	修改时间	状态	流量控制	常用操作
部署服务		2021-08-27 14:52:20	启用		
服务测试-1		2021-08-27 14:49:37	启用		<div>更多</div>



列表区

该界面分为如下几个区：

- 功能区：用于手动刷新列表和搜索。

操作	图标	说明
下载SDK	下载SDK	用于将服务提供给其他系统调用。
手动刷新	刷新	手动刷新服务列表。
搜索服务	输入你想要搜索的内容	服务流程名称关键字模糊匹配搜索结果。

- 状态栏：显示所有服务流程不同状态的统计数据。
- 列表区：显示服务流程列表，支持如下操作：

操作	图标	说明
服务启用		用于启用当前服务流程。 若当前服务呈启用状态，该按钮置灰。
服务下线		用于设置当前服务下线。 若当前服务呈下线状态，该按钮置灰。
编辑		用于编辑当前服务，进入“服务配置”界面（如下图所示），支持修改服务的别名和描述。 <div><div>服务配置 服务测试</div><div><div>服务ID</div><div>18a8aefa3017b86ba86ba455c017b958dc730041</div></div><div><div>服务名称</div><div>部署服务1</div></div><div><div>服务别名 *</div><div>部署服务1</div></div><div><div>内部调用地址</div><div>https://10.10.111.35:8901/api/v1/services/18a8aefa3017b86ba86ba455c017b958dc730041</div></div><div><div>实例ID</div><div>18a8aefa3017b86ba86ba455c017b86d4d180008</div></div><div><div>实例ID</div><div>18a8aefa3017b86ba86ba455c017b86d4d199w000a</div></div><div><div>服务描述</div><div></div></div><div>保存</div></div>

服务调用记录

服务调用记录[名称: 部署服务]

服务器名	开始时间	结束时间	执行时间	运行结果	错误日志
host-10-10-111-36	2021-08-27 14:48:53	2021-08-27 14:48:53	0秒	执行成功	---
host-10-10-111-36	2021-08-27 14:48:51	2021-08-27 14:48:51	0秒	执行成功	---
host-10-10-111-36	2021-08-27 14:41:18	2021-08-27 14:41:18	0秒	失败	<a href="#">错误日志</a>
host-10-10-111-36	2021-08-27 14:40:53	2021-08-27 14:40:53	0秒	失败	<a href="#">错误日志</a>
host-10-10-111-36	2021-08-27 14:40:50	2021-08-27 14:40:50	0秒	失败	<a href="#">错误日志</a>
host-10-10-111-36	2021-08-27 14:39:39	2021-08-27 14:39:39	0秒	失败	<a href="#">错误日志</a>
host-10-10-111-36	2021-08-27 14:39:14	2021-08-27 14:39:14	0秒	执行成功	---
host-10-10-111-36	2021-08-27 14:39:03	2021-08-27 14:39:04	1秒	执行成功	---

共 8 条

10条/页

< >

前往

1

/ 1页

确定(O)

取消(C)

错误日志：在资源有限的情况下，用于排查某个模型服务卡死。只有执行失败的服务记录才会有错误日志，点击错误日志即可下载日志到本地查看。

删除

用于删除当前服务流程。

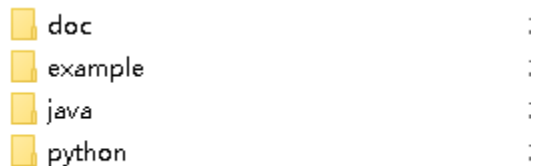


服务监控只能使用ClickHouse高速缓存库, 如果更改了错误的高速缓存库, 需要重启服务引擎后才能使用。

## 服务调用示例

调用流程说明:

1. 示例附件 ([ServiceInvokeDemo.zip](#)) 解压后, 得到Demo目录结构如下:



2. 各目录说明

- 1) doc——文档目录, demo说明文档在该目录下。
- 2) example——服务示例目录, 自带一个可运行的服务Dag示例, 可以在平台中进行流程导入, 并部署成服务。
- 3) java——java语言编写的demo工程, 该项目为maven工程, 可以使用开发工具以maven项目的形式进行导入, maven环境自行安装。
- 4) python——python语言编写的demo工程, 开发工具以python项目的形式进行导入, python环境自行安装 (推荐使用anaconda3和python3.6及其以上版本)。

3. 调用执行

- 1) java语言, 成功导入工程后, 在ServiceInvokeDemo.java文件中根据实际调用的服务环境, 修改对应的ip、端口、服务ID和输入数据, 然后运行。运行成功返回如图所示:

```
Result: {"data":[{"a":1.0,"b":"a","c":1.0}, {"a":2.0,"b":"b","c":3.1}], "type": "dataTable"}
```

备注: 该图片返回数据是自带服务Dag示例的返回数据。如果不是使用该服务示例, 可能返回数据有一定差异。

- 2) python语言, 成功导入工程后, 在ServiceInvokeDemo.py文件中根据实际调用的服务环境, 修改对应的ip、端口、服务ID和输入数据, 然后运行。运行成功返回如图所示:

```
{"data":[{"a":1.0,"b":"a","c":1.0}, {"a":2.0,"b":"b","c":3.1}], "type": "dataTable"}
```

备注: 该图片返回数据是自带服务Dag示例的返回数据。如果不是使用该服务示例, 可能返回数据有一定差异。

