阿里云
应用实时监控服务

自定义监控

文档版本：20190911

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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	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。	`swich` *`{stand | slave}`*

目录

1 自定义监控概述

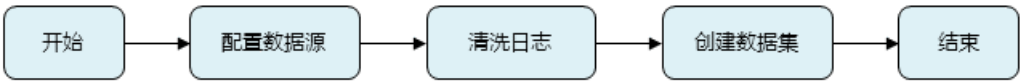
本文介绍 ARMS 自定义监控支持的监控任务类型和创建流程。

ARMS 自定义监控支持以下监控任务类型：

- 完全自定义的监控任务
- 基于自定义模板的监控任务
- 基于标准行业模板的监控任务

以上监控任务的流程相似，均包含配置数据源、清洗日志和创建数据集三个关键步骤，如下图所示。

图 1-1: 自定义监控任务的创建流程



完全自定义的监控任务

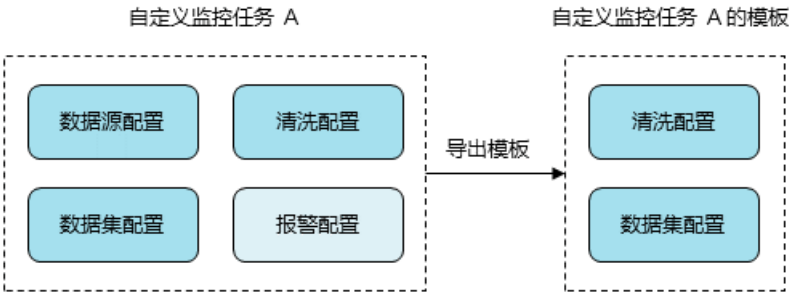
对于高度定制化的业务场景，可以通过创建自定义监控任务来清洗日志，自由统计所需指标，生成需要的数据与报表，灵活地配置报警。

基于自定义模板的监控任务

当需要创建一个与现有自定义监控任务大同小异的新监控任务时，可以从现有自定义监控任务导出模板，然后创建基于自定义模板的监控任务，这样就能省去重复的配置工作。

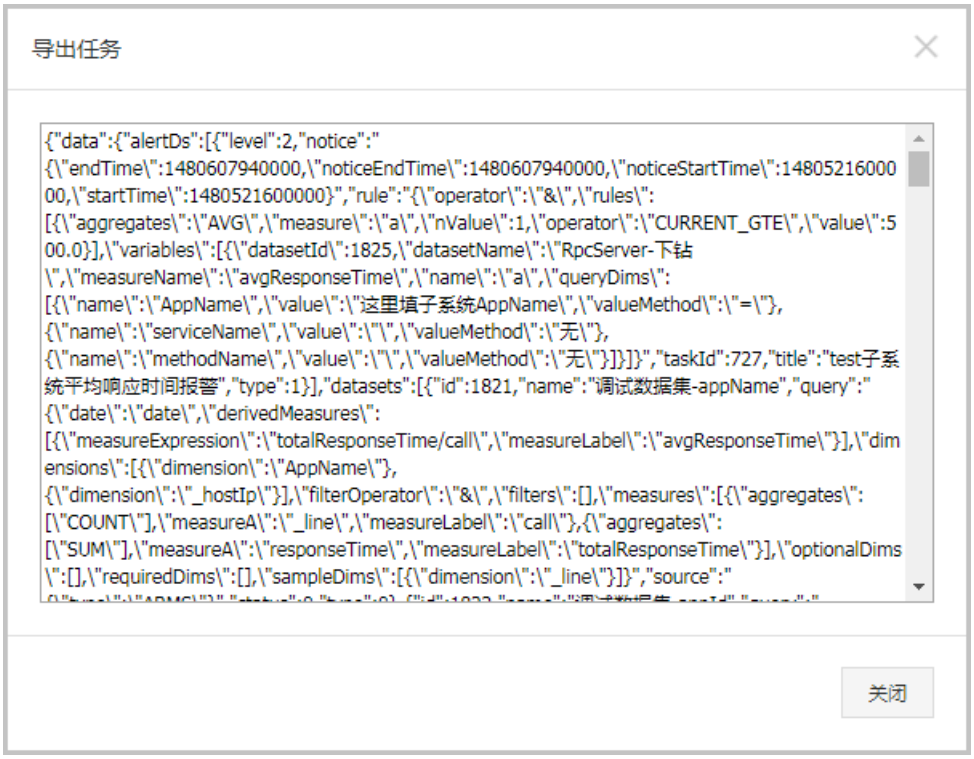
完整的监控任务包含数据源配置、清洗配置、数据集和/或报警配置，而导出的模板中仅包含通用配置，即清洗配置（日志样例与清洗规则）和数据集配置。因此，导入模板后，您需要重新配置数据源。

图 1-2: 监控任务模板



导出的模板为 JSON 文件。

图 1-3: 导出监控任务模板



基于标准行业模板的监控任务

标准行业模板是 ARMS 根据普遍的行业实践制定的监控任务模板，包含针对行业的常用监控字段。通过使用标准行业模板，ARMS 将自动根据模板内容生成日志清洗规则（如有）、数据集配置、报警配置，以及最终的交互大盘，从而快速生成对应任务的整套监控解决方案。

现有标准行业模板包括：

- Nginx 监控模板：通过收集和分析 Nginx 的默认日志，监控服务端的被访问情况。
- 零售行业监控模板：通过收集各区域门店的销售状态，实时反映零售行业的销售现状。

相关文档

#unique_4

#unique_5

2 管理数据源

2.1 数据源概述

数据源（日志源）为 ARMS 提供数据流入，您可以通过多种方式将数据推送到 ARMS 实时计算引擎。

- 云服务器 ECS

  通过 Logtail Agent 完成在 ECS 上的增量推送，例如日志文件。适用场景包括应用运行在阿里云 ECS 上的所有业务监控场景。

- LogHub 数据源

  将阿里云日志服务 LogHub 作为 ARMS 的数据源输入。如果 ECS 上的日志已经被 LogHub 收集，那么可以用此方法让 ARMS 复用 LogHub 上的数据。

- API 数据源

  通过 API SDK 向 ARMS 直接推送日志。适用于不适合装 Agent 但是可以通过 API 集成来直接推送数据收集场景，例如移动终端。

- MQ 数据源

  通过对接 MQ 直接拉取 Topic 内消息并基于其内容进行实时统计，进行业务监控。适用于已用 MQ 处理业务的用户，包括电商、物联网等领域。

2.2 管理 ECS 数据源

ECS 实例是重要的日志产生来源，也是 ARMS 支持的自定义监控数据源。您可以授权 ARMS 将您阿里云账号下的 ECS 实例信息同步至 ARMS 控制台，为这些实例安装 Logtail 日志采集 Agent。您也可以为这些 ECS 实例分组以便管理。

前提条件

执行同步 ECS 操作时必须使用阿里云账号或具备完整权限的 RAM 用户，不可使用不具备完整权限的 RAM 用户，例如仅具备只读权限的 RAM 用户。

同步 ECS 实例

为了进行后续的管理，首先需要将您阿里云账号下的 ECS 实例信息同步至 ARMS 控制台。

1. 登录 ARMS 控制台。
2. 在左侧导航栏中选择自定义监控数据源管理 > 云服务器 ECS 。

3. 在实例列表页面顶部选择目标地域，单击右上角的同步ECS。

> **说明:**
> 同步操作仅针对当前所选地域。

4. 如果您此前没有授权，则在提示框中单击进入RAM进行授权。

5. 在云资源访问授权页面选择所需权限，并单击同意授权。

说明:

如果您使用的是不具备完整权限的 RAM 子账号，则无法授权。请使用阿里云主账号或具备完整权限的 RAM 用户授权。

检查 Logtail Agent 状态

ARMS 系统通过 Logtail Agent 日志收集客户端来收集 ECS 实例上的日志，故需为每台 ECS 实例安装 Logtail Agent。

安装前需要检查 Logtail Agent 状态，可以对单个 ECS 实例执行检查，也可以对选中的多个 ECS 实例执行批量检查。

- 对单个 ECS 实例执行检查：在云服务器ECS页签上，在目标 ECS 实例右侧操作列中单击检查Agent。
- 对多个 ECS 实例执行检查：在云服务器ECS上，勾选所有目标 ECS 实例，并单击页面底部的批量检测Agent

安装 Logtail Agent

若检查发现未安装 Logtail Agent，请按照以下步骤安装：

1. 下载 Agent。请根据 ECS 实例的网络环境和日志服务所在地域替换 <*logtail.sh_path*>。

```
wget <logtail.sh_path> -O logtail.sh
```

网络类型	地域	下载地址
经典网络	华北 2（北京）	http://logtail-release-cn-beijing.oss-cn-beijing-internal.aliyuncs.com/linux64/logtail.sh
	华北 1（青岛）	http://logtail-release-cn-qingdao.oss-cn-qingdao-internal.aliyuncs.com/linux64/logtail.sh
	华东 1（杭州）	http://logtail-release-cn-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/linux64/logtail.sh
	华东 2（上海）	http://logtail-release-cn-shanghai.oss-cn-shanghai-internal.aliyuncs.com/linux64/logtail.sh
	华南 1（深圳）	http://logtail-release-cn-shenzhen.oss-cn-shenzhen-internal.aliyuncs.com/linux64/logtail.sh
VPC	华北 2（北京）	http://logtail-release-bj.vpc100-oss-cn-beijing.aliyuncs.com/linux64/logtail.sh
	华东 1（杭州）	http://logtail-release.vpc100-oss-cn-hangzhou.aliyuncs.com/linux64/logtail.sh
	华东 2（上海）	http://logtail-release-sh.vpc100-oss-cn-shanghai.aliyuncs.com/linux64/logtail.sh
	华南 1（深圳）	http://logtail-release-sz.vpc100-oss-cn-shenzhen.aliyuncs.com/linux64/logtail.sh
公网（自建 IDC 或其他云主机）	华北 2（北京）	http://logtail-release-cn-beijing.oss-cn-beijing.aliyuncs.com/linux64/logtail.sh
	华北 1（青岛）	http://logtail-release-cn-qingdao.oss-cn-qingdao.aliyuncs.com/linux64/logtail.sh
	华东 1（杭州）	http://logtail-release-cn-hangzhou.oss-cn-hangzhou.aliyuncs.com/linux64/logtail.sh
	华东 2（上海）	http://logtail-release-cn-shanghai.oss-cn-shanghai.aliyuncs.com/linux64/logtail.sh
	华南 1（深圳）	http://logtail-release-cn-shenzhen.oss-cn-shenzhen.aliyuncs.com/linux64/logtail.sh

2. 执行授权操作。

```
chmod 755 logtail.sh
```

3. 安装 Agent。请根据 ECS 实例的网络环境和日志服务所在地域替换 <*region_id*>。

```
sudo ./logtail.sh install <region_id>
```

网络类型	地域	地域 ID
经典网络	华北 2（北京）	cn-beijing
	华北 1（青岛）	cn-qingdao
	华东 1（杭州）	cn-hangzhou
	华东 2（上海）	cn-shanghai
	华南 1（深圳）	cn-shenzhen
VPC	华北 2（北京）	cn_beijing_vpc
	华东 1（杭州）	cn_hangzhou_vpc
	华东 2（上海）	cn_shanghai_vpc
	华南 1（深圳）	cn_shenzhen_vpc
公网（自建 IDC 或其他云主机）	华北 2（北京）	cn-beijing-internet
	华北 1（青岛）	cn-qingdao-internet
	华东 1（杭州）	cn-hangzhou-internet
	华东 2（上海）	cn-shanghai-internet
	华南 1（深圳）	cn-shenzhen-internet

4. 创建配置文件。

```
sudo touch /etc/ilogtail/users/1098370038733503
```

执行完上述步骤后，按照检查 Logtail Agent 状态的说明操作。若 Agent 状态变为已安装则表示安装成功。

卸载 Logtail Agent

按照1的说明下载 Logtail Agent，并在 Shell 环境下以管理员身份运行以下命令：

```
sudo sh logtail.sh uninstall
sudo rm -rf /etc/ilogtail/users/1098370038733503
```

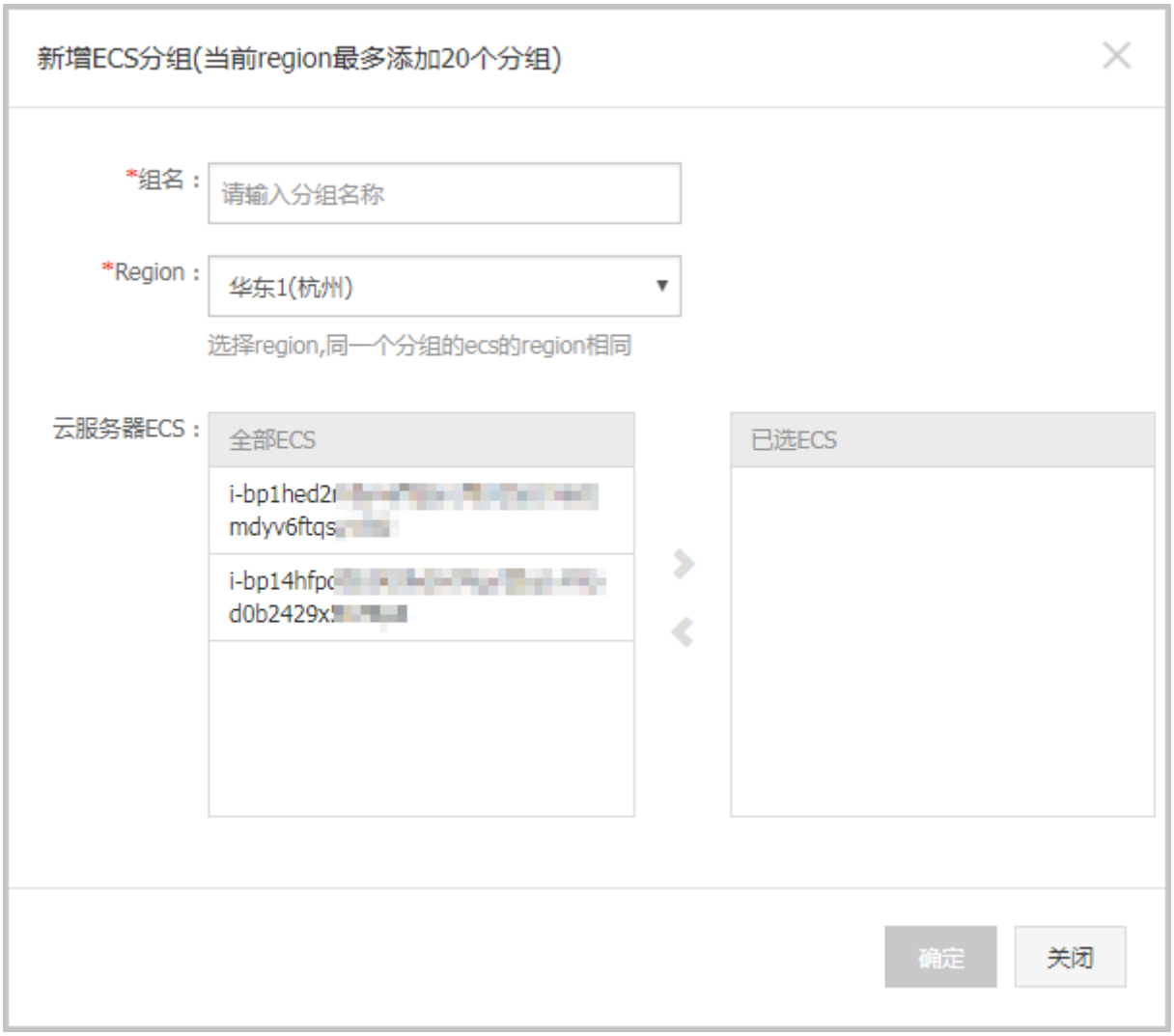
创建 ECS 分组

1.

2. 在控制台左侧导航栏中选择自定义监控数据源管理 > 云服务器 ECS，并单击云服务器ECS分组页签。
3. 在云服务器ECS分组页签单击右上角的创建ECS分组。
4. 在新增ECS分组对话框中，输入组名、选择地域（Region）、添加需要加入当前分组的 ECS 实例，单击确定。

说明：

只能将同一个地域的服务器纳入同一个 ECS 分组。



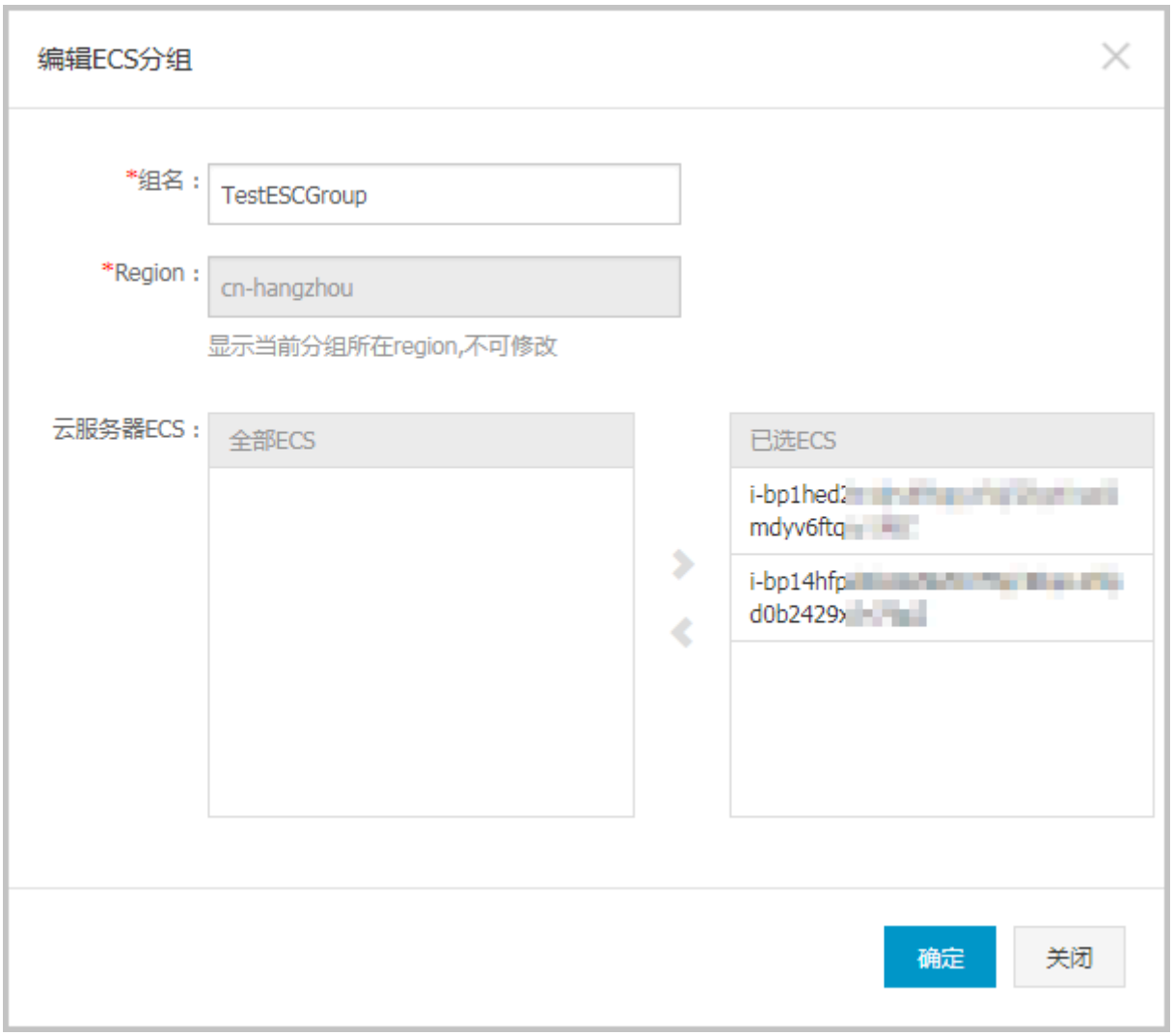
查看 ECS 分组内的实例详情

在云服务器ECS分组页签，单击 ECS 分组的展开按钮，即可查看该分组内的 ECS 实例详细信息，如图所示。

您也可以在此页面将特定 ECS 实例从分组中移除。

编辑 ECS 分组

单击 ECS 分组的铅笔按钮，即可修改该 ECS 分组的信息。例如，您可以修改 ECS 分组名称，添加 ECS 实例到分组，或者将某个 ECS 实例从分组中移除。



删除 ECS 分组

单击 ECS 分组的删除按钮，即可删除当前分组。

更多信息

2.3 同步 LogHub 数据源

如果您已经开通阿里云日志服务（Log Service），则可以在 ARMS 控制台直接同步日志服务 LogHub 中的数据并作为自定义监控的数据源使用。

前提条件

- 您已开通阿里云日志服务。

- 使用主账号或已被授权访问日志服务的 RAM 用户登录 ARMS 控制台。如果使用未被授权访问日志服务的 RAM 用户登录，则无法授权 ARMS 访问您在日志服务中的资源，继而无法同步 LogHub 数据源。

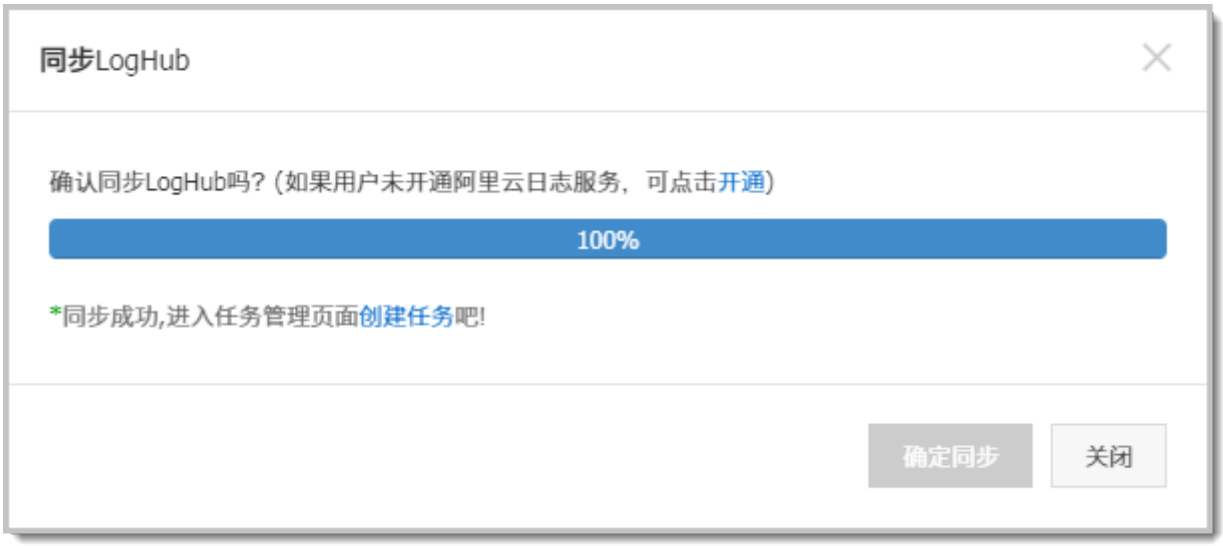
操作步骤

1. 登录 ARMS 控制台。
2. 在左侧导航栏中选择自定义监控数据源管理 > LogHub 数据源。如果此前未授权 ARMS 读取 LogHub 数据，则 ARMS 会给出提示信息。
3. 在提示对话框中单击进入 RAM 进行授权。

4. 在云资源访问授权页面勾选系统创建的 RAM 角色，并单击同意授权。

5. 在 LogHub 数据源页面右上角单击同步 LogHub。
6. 在同步 LogHub对话框中，单击确定同步。

7. 同步完毕后，在同步 LogHub 对话框中，单击关闭。同步后的数据将显示在 LogHub 数据源页面上。



后续操作

LogHub 数据源同步成功后，您可以前往自定义监控 > 监控任务管理页面创建自定义监控任务。在自定义监控任务的数据源配置页面上，当选择 LogHub 数据源作为日志源类型时，同步后的 LogHub 数据源将显示在选择日志源下拉列表中。

更多信息

- #unique_5
- #unique_10

2.4 Logstash SDK 数据源

2.4.1 SDK 数据源概述

ARMS 集成了阿里云日志服务的 Logstash SDK，您可以通过程序 SDK 集成直接推送数据到 ARMS。

接入 SDK 数据源主要分为两个步骤：

1. 创建 SDK 数据源。
2. 通过 SDK 写入数据到创建好的数据源。

创建 SDK 数据源

1. 在 ARMS 控制台左侧导航栏中选择自定义监控数据源管理 > SDK 数据源。
2. 在实例列表页面上，单击右上角的创建 SDK 数据源，并在提示对话框中单击确定创建。

> 说明:
> 一个地域中至多可以创建 20 个 SDK 数据源。

创建的 SDK 数据源显示在实例列表页面的列表中。

创建完毕后，在为自定义监控配置数据源（参见#unique_5）时，只需在日志源类型下拉列表中选择 SDK 数据源，即可在选择日志源下拉列表中看到创建好的 SDK 数据源。

通过 SDK 将数据写入创建好的数据源

通过 ARMS SDK，您可以将数据写入创建的数据源。目前支持 Java 和 Python。ARMS 采用了日志服务的 API，具体使用方法参见日志服务 SDK 文档“#unique_13”。除了填写 Endpoint、Project、LogStore、AK、SK 方法不同以外，其他方法类似。

- AK、SK：在实例列表页面上，单击操作栏中的获取 AccessKeys，即可在获取 AccessKeys 对话框中找到该信息。
- LogStore、Project：在实例列表页面的表格中。
- Endpoint：见下表。

表 2-1: ARMS 支持的 Endpoint

地域	Endpoint
北京	cn-beijing.log.aliyuncs.com
青岛	cn-qingdao.log.aliyuncs.com
上海	cn-shanghai.log.aliyuncs.com
杭州	cn-hangzhou.log.aliyuncs.com
深圳	cn-shenzhen.log.aliyuncs.com

2.4.2 SDK for Java

本文为 Java 示例。

引入 POM 依赖

```
<dependency>
  <groupId>com.aliyun.openservices</groupId>
  <artifactId>aliyun-log</artifactId>
  <version>0.6.6</version>
</dependency>
```

开始一个 Java 程序

```
public class LogstashForJavaDemo {
    public static void main(String[] args) throws LogException {
        DateFormat dateFormat = new java.text.SimpleDateFormat("yyyy-
MM-dd HH:mm:ss");
        /**
         * endpoint: 数据写入存储所在区域
         * project,logstore: 构成基本数据存储目标
         * accessKeyId,accessKeySecret: 构成访问密钥
         *
         * 注意: 请用户根据实际情况填写
         */
        String endpoint = "cn-hangzhou.log.aliyuncs.com";
        String project = "proj-arms-7dd6ecb06d21e02aed9eeb56b79e9f";
        String logstore = "logstore-56f96ec5546fb6555ef97dd057acb4e9";
        String accessKeyId = "xxx";
        String accessKeySecret = "yyy";
        int logGroupSize = 10;// 建议100-2000, 每个batch发送数据上限
        List<String> examples = new ArrayList<String>();
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=男装&区域=杭州&eventTeyp=1&性别=1&价格=2140|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=家居&区域=上海&eventTeyp=3&性别=0&价格=8305|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=食品&区域=深圳&eventTeyp=3&性别=1&价格=7121|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=男装&区域=上海&eventTeyp=3&性别=1&价格=2917|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=食品&区域=上海&eventTeyp=1&性别=1&价格=4285|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=男装&区域=杭州&eventTeyp=3&性别=1&价格=7864|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=女装&区域=杭州&eventTeyp=5&性别=0&价格=2983|");
        examples.add("|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=食品&区域=深圳&eventTeyp=5&性别=1&价格=3201|");
        // 构建一个客户端实例
        Client client = new Client(endpoint, accessKeyId, accessKeyS
ecret);
        // 连续发送10个数据包, 每个数据包有10条日志
        long currentTime = System.currentTimeMillis();
        String formatedTime = dateFormat.format(new Date(currentTime
));
        for (int i = 0; i < 10; i++) {
            Vector<LogItem> logGroup = new Vector<LogItem>();
            for (int j = 0; j < logGroupSize; j++) {
                LogItem logItem = new LogItem();
                logItem.PushBack("content", formatedTime + examples.
get(j % examples.size()) + UUID.randomUUID());
                logGroup.add(logItem);
```

```
            }
            PutLogsRequest req = new PutLogsRequest(project, logstore,
 "", "", logGroup);
            client.PutLogs(req);
        }
        System.out.println("send data success");
    }
}
```

重要参数说明

参数	说明
endpoint	数据写入区域（#unique_15/unique_15_Connect_42_table_ow2_prs_gfb）
accessKeyId	写入数据时的密钥 ID
accessKeySecret	写入数据时的密钥密码
project	写入数据的 Project ID
logstore	写入数据的 Logstore ID

说明:

- ARMS 颁发的 accessKeyId、accessKeySecret 非阿里云 AK/SK，需要从 ARMS 获取，详情参见#unique_16。
- Project ID 和 Logstore ID 标识一个唯一的数据源。

2.4.3 SDK for Python

本文为 Python 示例。

安装 Python 环境

Python SDK 是一个纯 Python 的库，支持所有运行 Python 的操作系统，目前主要为 Linux 和 Windows。

请按以下步骤安装 Python：

1. 下载并安装最新的 Python 2 安装包。

   目前，Python SDK 支持 Python 2.6/2.7环境。可以运行 `python -V` 查询当前 Python 版本。

2. 下载并安装 Python 的包管理工具 pip。

   完成 pip 安装后，你可以运行 `pip -V` 确认 pip 是否安装成功和查看当前 pip 版本。

安装 Python SDK 的依赖库

Python SDK 依赖于一组第三方的 Python 库。在使用该 SDK 前必须安装好以下依赖库：

- Google Protocol Buffer：Python SDK 依赖于 Protocol Buffer 协议向服务端写入日志。执行以下命令进行安装：

  ```
  sudo pip install protobuf==2.5.0
  ```

  pip 安装 Protobuf 需要连接 Python Package Index 网站，请确保机器能够访问该网站。如果因为网络问题无法安装成功，可以考虑从 Protocol Buffer 的 Github 官方网站手动下载安装（具体安装步骤请参考下载包的 Readme 文档）。
- Python-Requests：Python SDK 依赖于 Python-Requests 类进行 HTTP 通信。执行以下命令进行安装：

  ```
  sudo pip install requests
  ```
- SimpleJson：Python SDK 依赖于 SimpleJson 处理 API 的 JSON 格式返回结果。执行以下命令进行安装：

  ```
  sudo pip install simplejson
  ```

安装 Python SDK

安装好上述环境后，需要下载 Python SDK。

1. 从 GitHub 下载最新的 Python SDK 包。
2. 解压完整下载的包到指定目录。
3. 在解压后的目录运行以下命令安装 Python SDK。

   ```
   python setup.py install
   ```

开始一个 Python 程序

```
#!/usr/bin/env python
#encoding: utf-8
import datetime
from aliyun.log.logclient import LogClient
from aliyun.log.logitem import LogItem
from aliyun.log.putlogsrequest import PutLogsRequest
def main():
    #endpoint: 数据写入存储所在区域
    #project,logstore: 构成基本数据存储目标
    #accessKeyId,accessKeySecret: 构成访问密钥
    #注意: 请用户根据实际情况填写
    endpoint = "cn-hangzhou.log.aliyuncs.com"
    project = "proj-arms-7dd6ecb06d21e02aed9eeb56b7****"
    logstore = "logstore-56f96ec5546fb6555ef97dd057ac****"
    accessKeyId = "utmxiro7BYtT****"
    accessKeySecret = "PyjsffdlggBoYcrgpr69w023b9****"
    logGroupSize = 10
```

```
    examples = []
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=男装&区域=杭州&eventTeyp=1&性别=1&价格=2140|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=家居&区域=上海&eventTeyp=3&性别=0&价格=8305|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=食品&区域=深圳&eventTeyp=3&性别=1&价格=7121|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=男装&区域=上海&eventTeyp=3&性别=1&价格=2917|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=食品&区域=上海&eventTeyp=1&性别=1&价格=4285|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=男装&区域=杭州&eventTeyp=3&性别=1&价格=7864|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=女装&区域=杭州&eventTeyp=5&性别=0&价格=2983|')
    examples.append('|c0a895e114526786450161001d1ed9|9|EADS|BIZ-
MONITOR|0|类目=食品&区域=深圳&eventTeyp=5&性别=1&价格=3201|')
    # 构建一个client
    client = LogClient(endpoint, accessKeyId, accessKeySecret)
    for i in range(0, 10):
        logGroup = []
        for j in range(0, 10):
            logItem = LogItem()
            logItem.push_back("content", datetime.datetime.now().
strftime('%Y-%m-%d %H:%M:%S') + examples[j % len(examples)])
            logGroup.append(logItem)
        req = PutLogsRequest(project, logstore, "", "", logGroup)
        client.put_logs(req)
    print "send data success"
if __name__ == '__main__':
    main()
```

重要参数说明

参数	说明
endpoint	数据写入区域（#unique_15/unique_15_Connect_42_table_ow2_prs_gfb）
accessKeyId	写入数据时的密钥 ID
accessKeySecret	写入数据时的密钥密码
project	写入数据的 Project ID
logstore	写入数据的 Logstore ID

说明:

- ARMS 颁发的 accessKeyId、accessKeySecret 非阿里云 AK/SK，需要从 ARMS 获取，详情参见#unique_16。
- Project ID 和 Logstore ID 标识一个唯一的数据源。

2.5 MQ 数据源

如果您已开通消息队列 MQ，ARMS 可直接统计 MQ 消息的内容。关于利用 MQ 数据源进行监控的案例，请参考#unique_19。

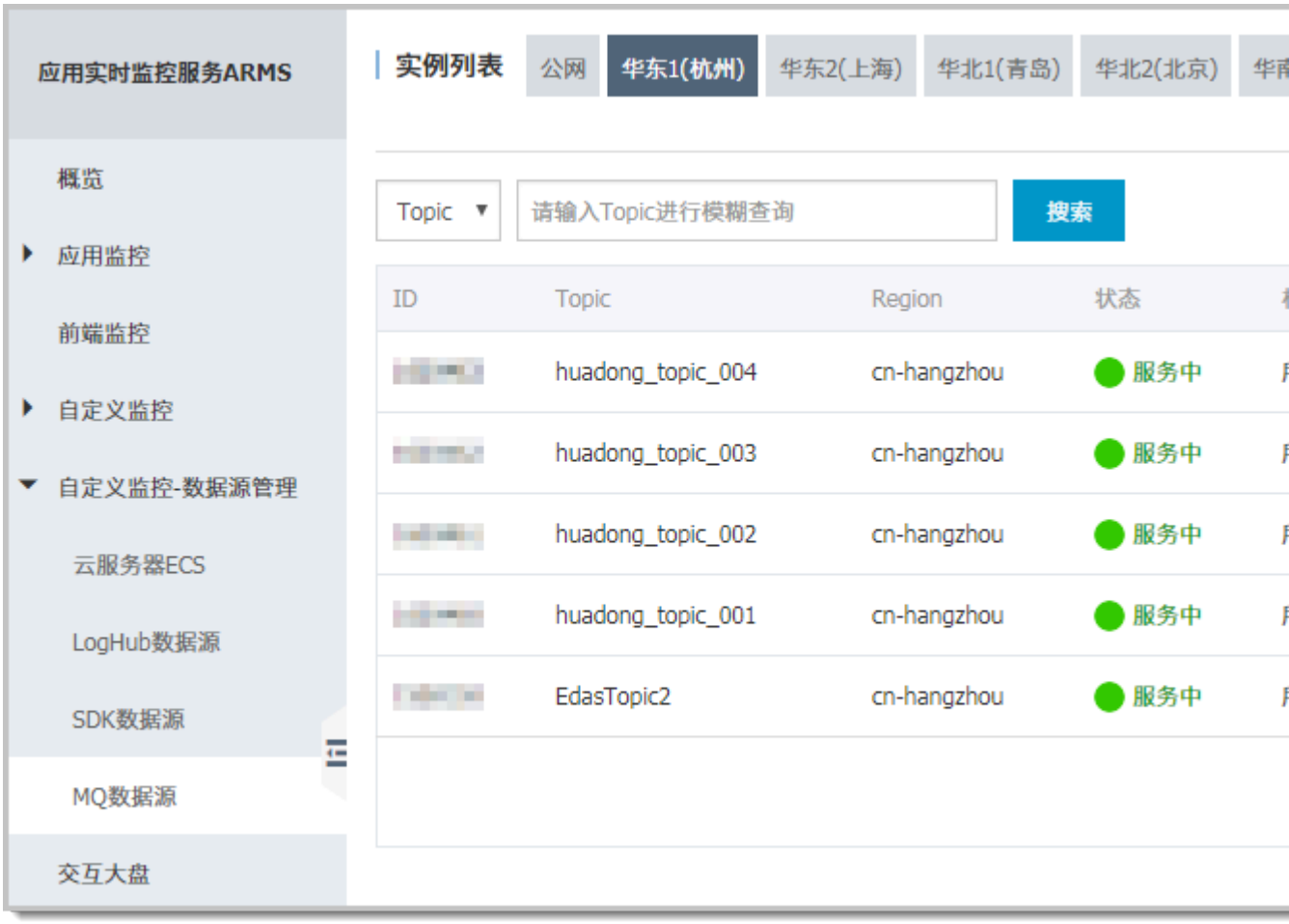
前提条件

- 您已开通阿里云消息队列 RocketMQ，且已创建 Topic。

操作步骤

1. 登录 ARMS 控制台。
2. 在控制台左侧导航栏中选择自定义监控数据源管理 > MQ 数据源。如果此前未授权 ARMS 读取 MQ 数据，则 ARMS 会给出提示信息。
3. 在提示对话框中单击确定授权，然后在确认对话框中单击确定。

4. 在 MQ 数据源页面右上角单击刷新。您在该地域所拥有的 MQ Topic 将显示在该页面上。



后续操作

MQ 数据源刷新成功后，您可以前往自定义监控 > 监控任务管理页面创建自定义监控任务。在自定义监控任务的数据源配置页面上，当选择 MQ 数据源作为日志源类型并选择 Region 后，您在该 Region（地域）的 MQ Topic 将显示在 Topic 下拉列表中。

更多信息

- #unique_5

3 创建监控任务

3.1 配置数据源

本文介绍了创建自定义监控任务的第 1 个步骤：配置数据源。在准备好数据源后，需要在 ARMS 控制台进行简单的配置。

背景信息

创建自定义监控时，您有以下三种选择：

- 创建完全自定义的监控：您需要手工完成所有配置工作，包括配置数据源、清洗日志、创建报警等。
- 创建基于标准行业模板的监控：您只需要选择 ARMS 提供的一种标准行业模板，即可快速创建自定义监控。标准行业模板是指针对某个行业或者某类问题的监控任务解决方案。
- 创建基于自定义模板的监控：您可以将现有的自定义监控导出为模板，并在创建新的自定义监控时导入此模板，以减少手工配置的工作量。

不论选择哪种方式，自定义监控任务的总体流程都是一致的，区别在于基于模板创建时能利用已有的配置，从而减少手工配置工作量。

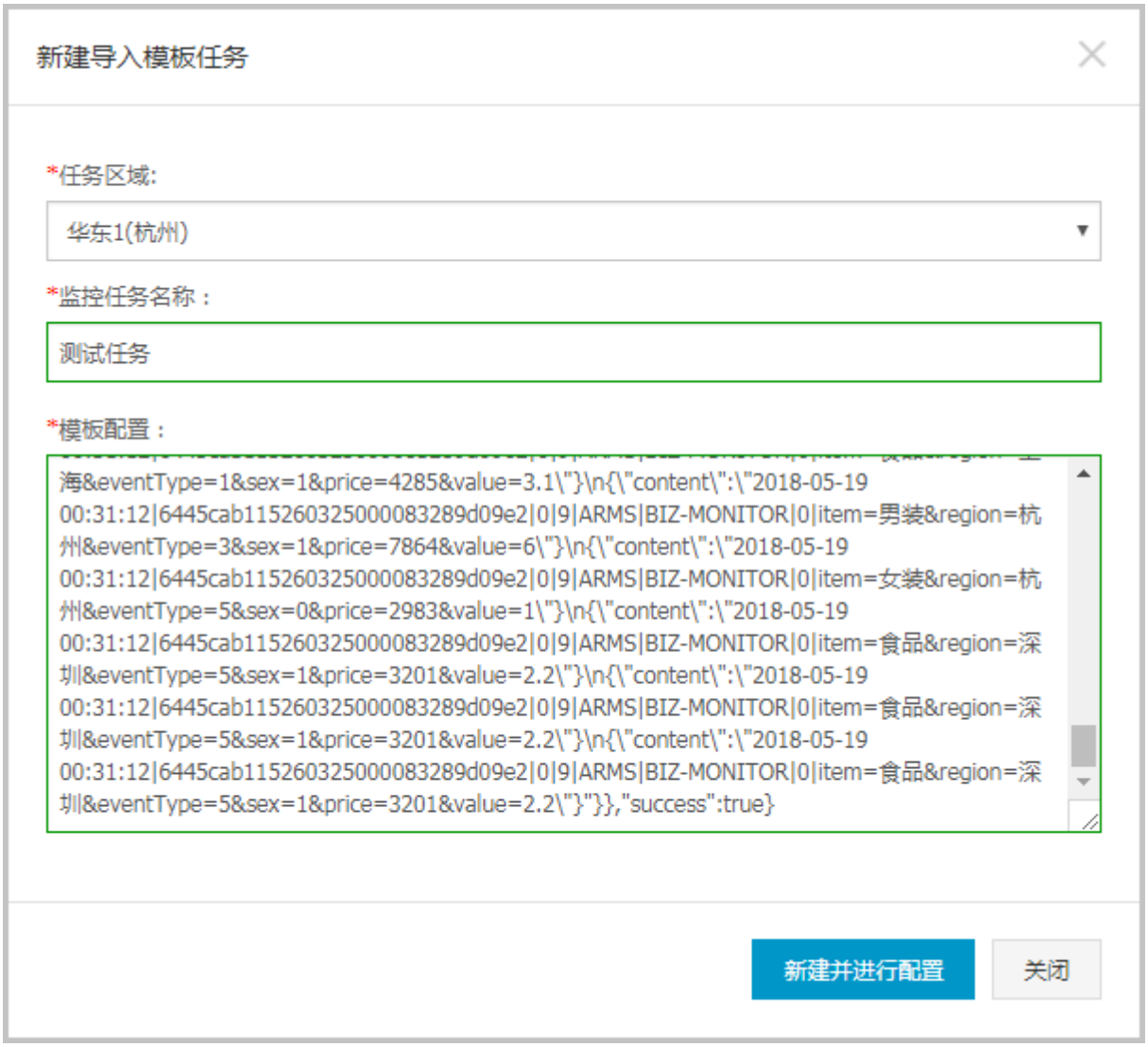
操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 监控任务管理。

2. 在监控任务管理页面上按需执行以下操作，并在弹出对话框中单击新建并进行配置。

   - 如需创建自定义监控任务，在右上角选择新建监控任务 > 新建自定义监控，然后在新建自定义任务对话框中选择任务区域，并输入监控任务名称。
   - 如需创建基于行业模板的监控任务，在右上角选择新建监控任务 > 新建标准模板监控，然后在新建行业模板任务对话框中选择任务区域和模板配置，并修改监控任务名称和交互大盘名称。

- 如需创建基于自定义模板的监控任务，在右上角选择新建监控任务 > 导入自定义模板，然后在新建导入模板任务对话框中选择任务区域，输入监控任务名称，并在模板配置文本框中粘贴准备好的模板文件。



> **说明:**
> 如需将一个现有任务导出为模板，请在监控任务管理页面，单击该任务右侧的更多 > 导出模板，然后在弹出对话框中拷贝任务模板。

3. 在数据源配置页面上的日志源配置区域输入日志源类型、日志源路径、编码设置等信息。

> **说明:**
>
> - ARMS 支持的数据源类型包括云服务器 ECS、LogHub、API 和 MQ。详情参见#unique_23。ECS 数据源有以下限制：一个文件一次只能被一个任务消费，而且被抓取文件所在目录所含的文件数量不能超过 200。

- 在日志路径字段（如有）中，请输入日志的绝对路径，例如 `/apps/arms.log`。如果有多个路径，请以逗号分隔，例如 `/root/arms01.log,/apps/arms02.log`。
- 编码设置：默认为“自动探测”模式，但建议选择特定编码，因为自动探测可能会导致乱码。

4. 在日志抓取结果区域，单击右上角的日志抓取预览。

> 说明:
> ARMS 会从选择的机器日志中抓取部分数据（最多 20 条）。由于需要建立预抓取的临时通道，一般需要 30 秒左右。

日志抓取结果显示在预览窗口中。

> 说明:
> 如果预抓取日志不成功，请检查输入的日志源、日志路径和采集目标等信息是否正确。

5. 在数据源配置页面单击保存和下一步。

参考

- #unique_24
- #unique_22
- #unique_23

3.2 清洗日志

清洗日志是指通过切分、静态 Join 等操作，将日志数据转化为标准 Key-Value（KV）格式的过程。

前提条件

- 已完成自定义监控任务的以下步骤：#unique_5。
- 如果是已经创建完毕的自定义监控任务，且该任务正在运行，请先暂停或停止该任务。

背景信息

日志清洗有两种模式：智能切分和自定义切分。智能切分模式会对抓取的日志数据进行自动切分，将其转化为标准的 KV 格式。您还可以调整方案，包括删除和修改等操作。如果智能切分方案无法满足需求，您可以使用可视化的自定义切分功能。

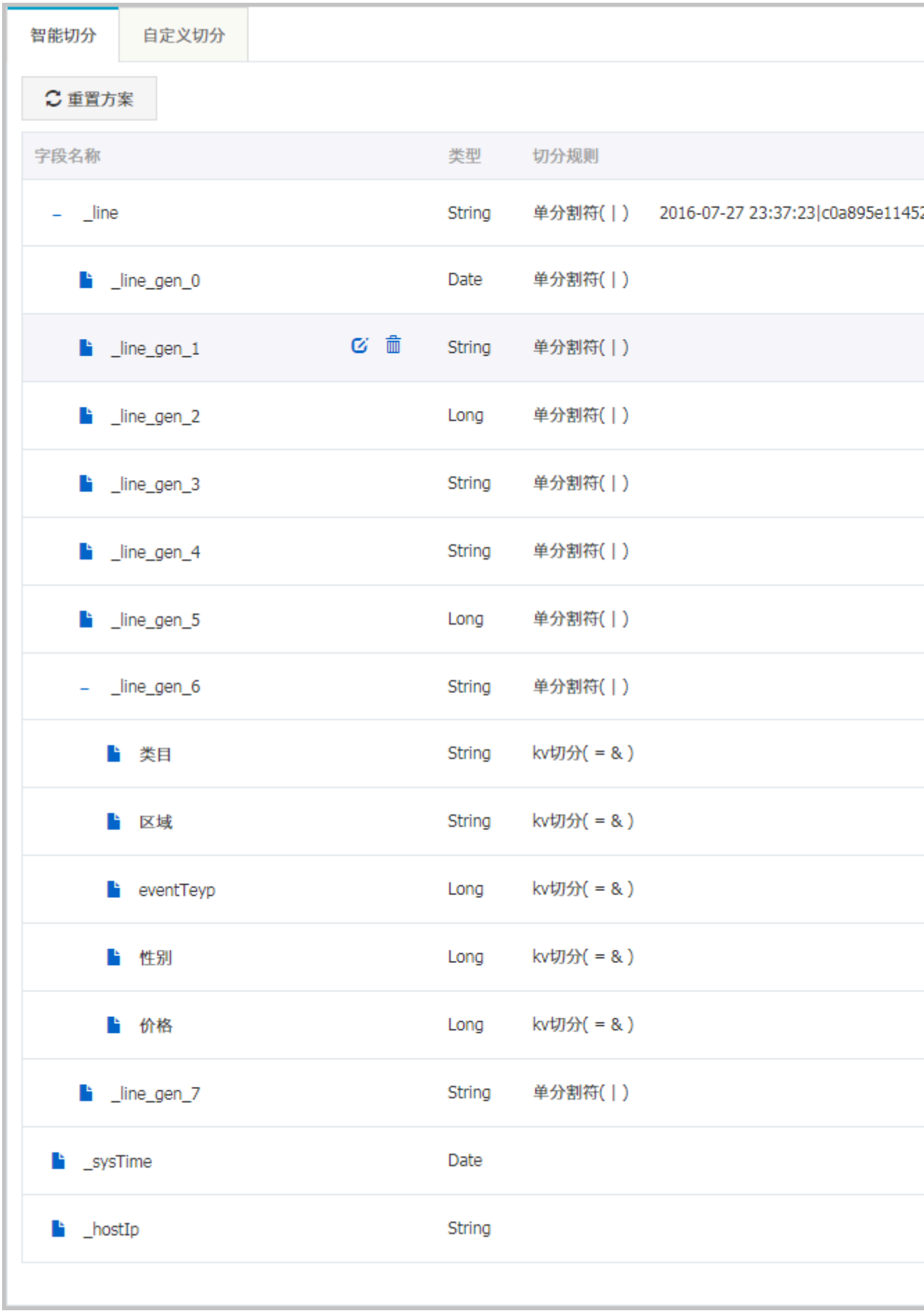
操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 监控任务管理。

2. 在监控任务管理页面上找到目标监控任务，单击右侧的编辑，并单击下一步。

3. 在日志清洗页面单击智能切分页签，并单击获取方案。

   ARMS 会对日志抓取结果区域框中的样例日志进行智能切分，切分方案显示在下方表格中。


智能切分
自定义切分
重置方案
字段名称
类型
切分规则
_line
String
单分割符(|)
2016-07-27 23:37:23|c0a895e11452
_line_gen_0
Date
单分割符(|)
_line_gen_1
String
单分割符(|)
_line_gen_2
Long
单分割符(|)
_line_gen_3
String
单分割符(|)
_line_gen_4
String
单分割符(|)
_line_gen_5
Long
单分割符(|)
_line_gen_6
String
单分割符(|)
类目
String
kv切分(= &)
区域
String
kv切分(= &)
eventTeyp
Long
kv切分(= &)
性别
Long
kv切分(= &)
价格
Long
kv切分(= &)
_line_gen_7
String
单分割符(|)
_sysTime
Date
_hostIp
String

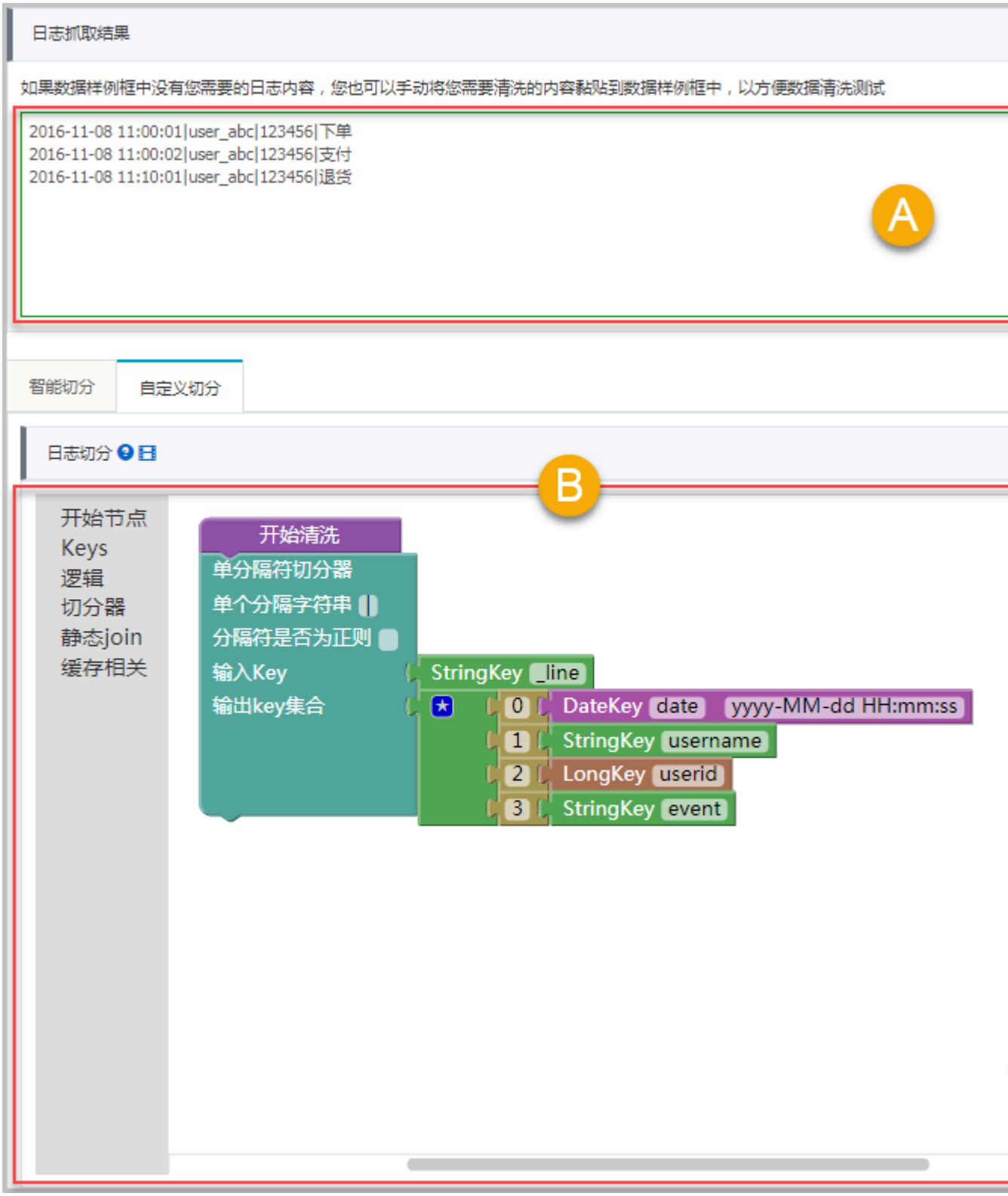
将鼠标悬停在需要修改的字段所在的行，会显示编辑（ ）和删除（ ）图标。

- 如需编辑字段，单击编辑图标，然后在编辑切分方案对话框中修改字段名称或切分类型。
- 如需删除字段，单击删除图标，然后在删除切分字段对话框中单击删除。

说明:

修改方案后，如需恢复为推荐方案，请单击重置方案。

4. 如果智能切分方案无法满足需求，请单击自定义切分页签。



- A：样例日志区域框
- B：可视化清洗流程编辑区域框
- C：清洗结果预览区域框

5. 在可视化清洗流程编辑区域框，按需执行以下操作。

   - 如需插入积木块，请单击编辑区左侧菜单中的积木块类别，然后单击要插入的积木块。选中的积木块会出现在右侧的空白编辑区域框，您可以用鼠标自由拖放。

   - 如需拼接积木块，请按住目标积木块，将其拖移至其他积木块的接口处并释放，积木块将会自动吸附在一起。
   - 如需移除积木块，请按住目标积木块，将其拖移至右下角的垃圾桶图表上并释放。

> 说明:
> 积木块的本质是 XML 代码，如需复用拼装好的积木块，请单击导入/导出，然后在导入/导出切分方案对话框中，拷贝当前代码留待日后使用（即导出），或者将其他可复用的代码粘贴至此处（即导入）。

6. 单击日志切分预览。

   日志切分结果显示在清洗结果预览区域框。

7. 单击保存和下一步。

参考

- #unique_5
- #unique_26
- #unique_27
- #unique_28
- #unique_29

- #unique_30

3.3 创建数据集

数据集定义了监控任务中采集日志的预聚合方式和持久化存储方式。在 ARMS 中，通过简单的交互，就可以获得 ARMS 系统针对多维数据分析优化后的数据组织和存储方式。

前提条件

已有自定义监控任务。

背景信息

创建多维数据集是指将日志中的流式数据转化为结构化的模型数据存储，并提供页面查询方式和 API 查询方式。在 ARMS 中进行简单的操作，就可以获得 ARMS 系统关于多维数据分析优化的数据组织和存储模式。

多维数据集在概念上是一个超立方块（Hypercube），而在物理存储上可以理解为一个多维数组。维的属性值被映射为多维数组的下标值或下标值的范围，而指标数据作为多维数的值存储在数据的单元中。可以将维看作自变量，将指标数据看作因变量。

维度有以下使用限制：

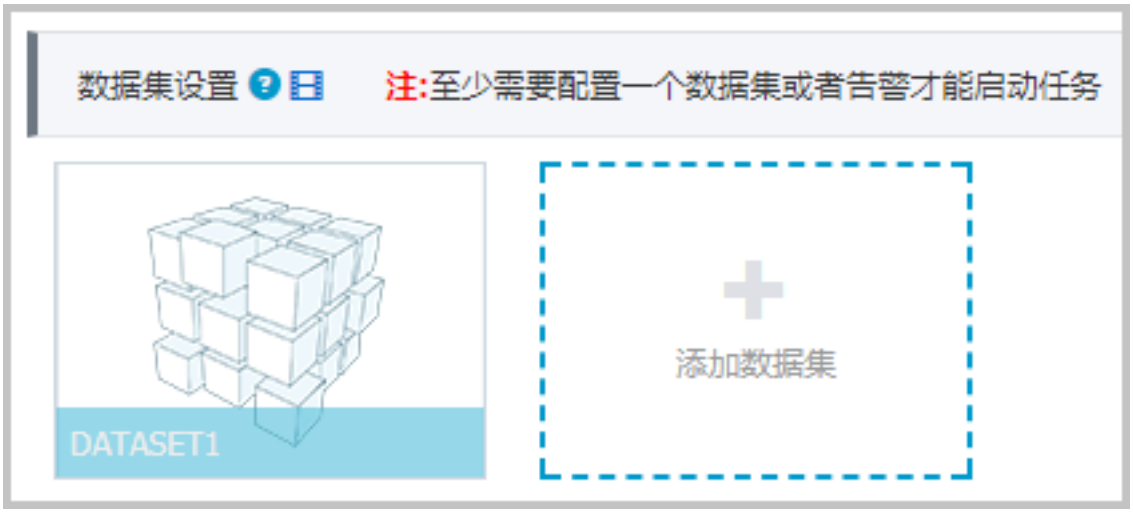
- 最多可以设置三个维度的数据集。
- 维度之间是有层级关系的。例如，要查看第二个维度，必须要先选择第一个维度的属性。维度类似于一个树状结构。维度的定义需要规划，例如，将省作为第一维度，市作为第二维度，区作为第三维度，市民消费情况作为指标数据。
- 除非有特殊场景需求，否则请不要一同定义两个完全不相关的维度，例如“地域”和“物品类型”。
- ARMS 提供下钻功能，可以从汇总数据深入到细节数据进行观察或新增维度。钻取的目的是改变维度层次、变换分析粒度。

操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 监控任务管理。
2. 在监控任务管理页面找到目标监控任务，单击右侧的编辑，并进入数据集与报警配置页面。

> 说明：
> 如果单击编辑之前，自定义监控任务正在运行，则会直接进入数据集与报警配置页面。如果自定义监控任务不在运行，则需要通过单击下一步导航至数据集与报警配置页面。

3. 在数据集设置区域框单击添加数据集。



4. 在添加数据集对话框中输入相关信息，并单击保存。

· 筛选：只有满足该筛选条件的数据集会被计算在内。

> 说明:
> 同时满足下述规则对应且的关系，满足下述一条规则对应或的关系。

- 指标： 一般为数字类型，是衡量目标的度量，类似于多维联机分析处理（Multi-Dimensional OLAP）中的值。ARMS 的指标对应于实时计算后的 Count、Max、Sum、Count Distinct 等值。
- 复合指标：可以对数据集的指标结果进行加减乘除等运算。
- 时间字段：日志切分对应的时间字段，是实时监控的最基础维度。
- 维度：是衡量目标的思维角度。例如按班级统计学生人数，那么班级就是维度。相当于 SQL 语言中的GROUP BY。ARMS 的维度分通用类型和下钻类型，详情参见#unique_32。
- 采样字段：是指这一分钟内对某个字段数据的采样，便于在监控异常时根据当时的样本排查问题。

5. 单击保存和完成配置。
6. 在启动监控任务对话框中，选择从头开始消费或从最新位置消费，并单击确定。 监控任务启动成功。
7. 回到监控任务管理页面，为目标监控任务单击启动。

   设置的报警生效。

参考

- #unique_33
- #unique_32

3.4 创建全息排查事件集

本文介绍了如何为自定义监控任务创建事件集。事件集是对日志的结构化存储，支持全文搜索、指定特殊字段的搜索和全息排查。

前提条件

已有自定义监控任务。

背景信息

事件集是对日志的结构化存储，支持全文搜索、指定特殊字段的搜索和全息排查。您可以在调用链路查询页面进行全息排查事件查询。

操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 监控任务管理。

2. 在实例列表找到上一步创建的监控任务，单击右侧的编辑，并进入数据集与报警配置页面。

> **说明:**
> 如果单击编辑之前，自定义监控任务正在运行，则会直接进入数据集与报警配置页面。如果自定义监控任务不在运行，则需要通过单击下一步导航至数据集与报警配置页面。

3. 在全息排查事件集设置区域框单击添加全息排查事件集。

4. 在添加事件集对话框中输入相关信息，并单击保存。

5. 单击保存和完成配置。
6. 在启动监控任务对话框中，选择从头开始消费或从最新位置消费，并单击确定。 监控任务启动成功。

更多信息

- #unique_35
- #unique_36

3.5 添加模式检测配置

本文介绍了如何添加模式检测配置。该配置是模式检测模块主动发现日志中的异常以及对比不同时间段日志的前提条件。关于模式检测模块的详细信息，请参考#unique_38。

前提条件

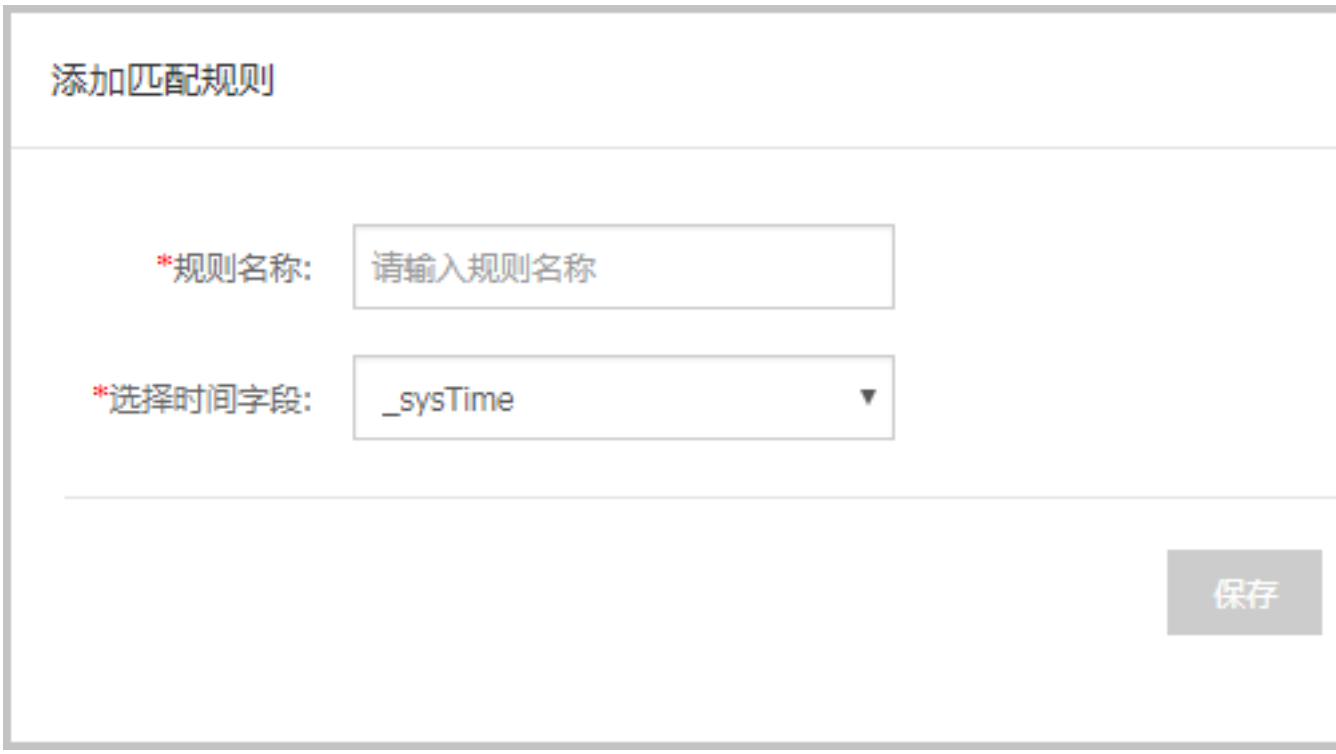
已有自定义监控任务。

操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 监控任务管理。
2. 在监控任务管理页面上找到目标监控任务，单击右侧的编辑，并进入数据集与报警配置页面。

   说明:
   如果单击编辑之前，自定义监控任务正在运行，则会直接进入数据集与报警配置页面。如果自定义监控任务不在运行，则需要通过单击下一步导航至数据集与报警配置页面。
3. 在模式检测区域框单击添加匹配规则。

4. 在添加匹配规则对话框中输入规则名称，选择时间字段（日志中的业务时间或系统时间），并单击保存。



> 说明:
> 在选择时间字段下拉框中，如果未切分日志，可选择系统时间 `_sysTime`（默认选项）。如果已切分日志，可以选择日志中的业务时间。

5. 单击保存和完成配置。
6. 在启动监控任务对话框中，选择从头开始消费或从最新位置消费，并单击确定。 监控任务启动成功。

   设置的模式检测配置生效。

后续操作

在自定义监控任务实例列表页面启动该任务后，即可前往模式检测模块查看日志异常情况或对比不同时间的日志。

> 说明:
> 启动自定义监控任务后，大约 3 分钟后可看到模式检测结果。

更多信息

- #unique_39
- #unique_38

4 管理监控任务

4.1 查看监控任务详情

本文介绍了如何查看监控任务详情、数据集详情和任务运行详情。这些信息有助于进行运维和故障诊断。

查看监控任务详情

1. 登录 ARMS 控制台。
2. 在控制台左侧导航栏中选择自定义监控 > 监控任务管理。
3. 在监控任务管理页面上，单击目标监控任务右侧的浏览。

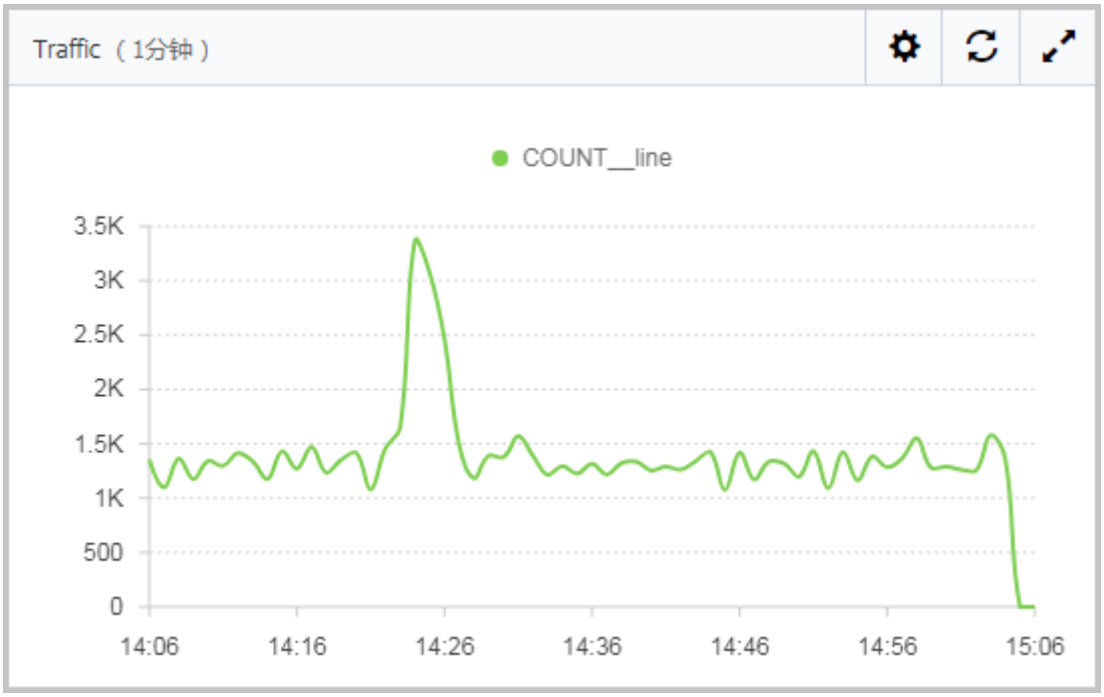
   监控任务详情页面以只读方式展示任务的以下配置信息：

   - 数据源配置
   - 日志切分模型
   - 数据集配置

查看数据集详情

1. 在左侧导航栏中单击数据集详情。

   数据集详情页面以折线图的形式展示该任务的所有数据集。

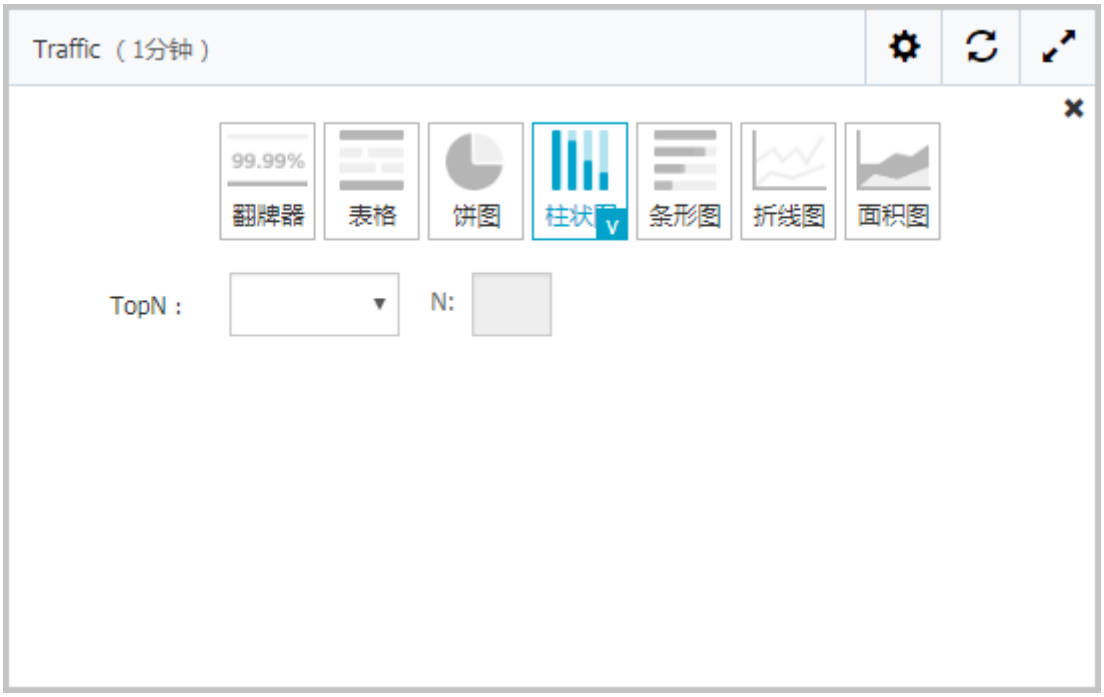


说明:

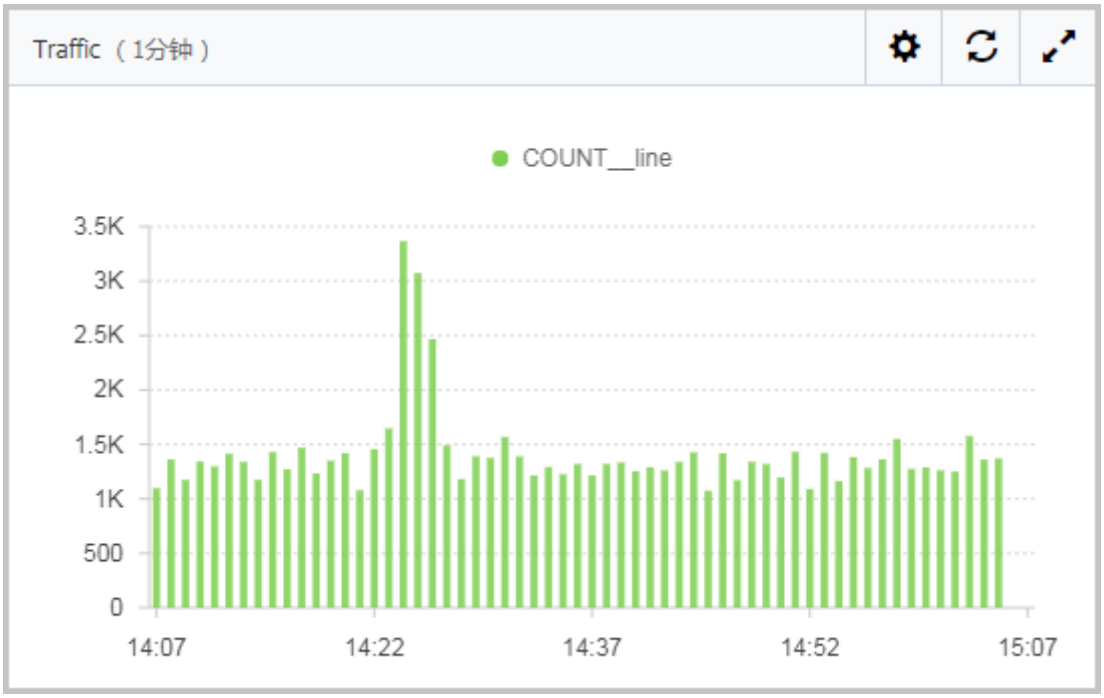
每配置一个报警就会生成一个数据集。

2. 单击右上角的齿轮，显示所有配置选项，并按需更改配置。

   例如，单击柱状图。



下图为配置后的效果。

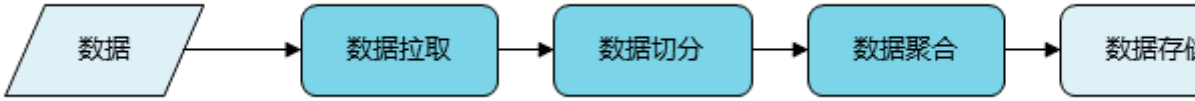


查看任务运行详情

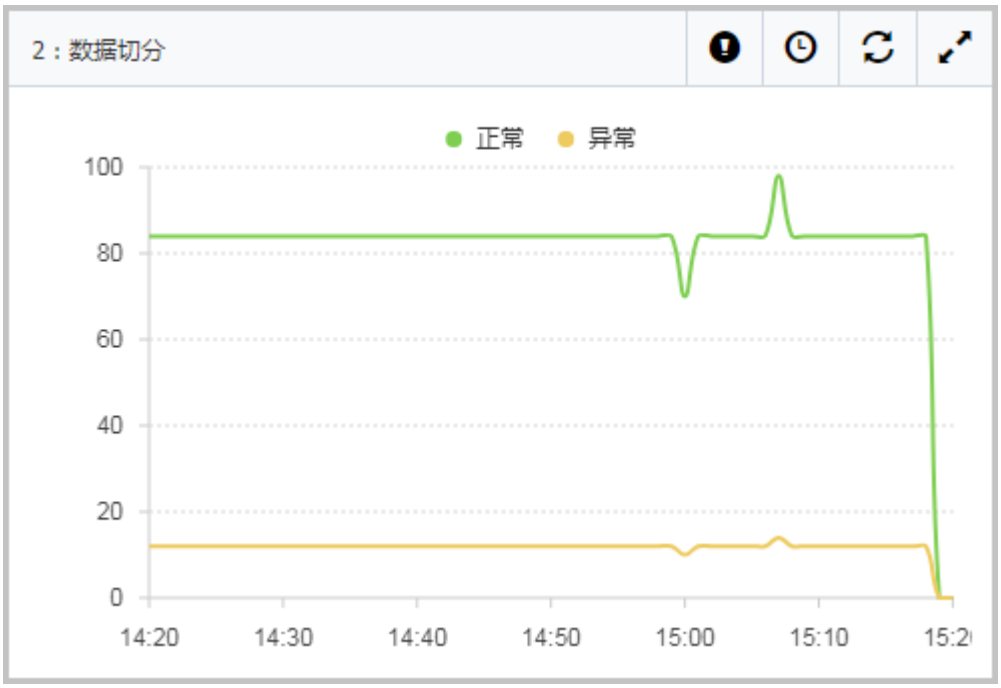
1. 在左侧导航栏中单击任务运行详情。

   任务运行详情页面展示了任务的详细运行过程，包括除数据存储之外的另外三个环节。

   图 4-1: 任务运行详情



   通过数据拉取、数据切分、数据聚合图表，可以详细地看到每分钟处理成功的行数和处理失败的行数。下图为数据切分图表。



2. 单击图表右上角的感叹号图标。

4.2 监控任务的管理

在监控任务管理页面，可以对监控任务进行浏览、编辑、启动、停止、暂停、恢复、复制、导出模板等操作。

在控制台左侧导航栏中选择自定义监控 > 监控任务管理，进入监控任务管理页面。

更改任务状态

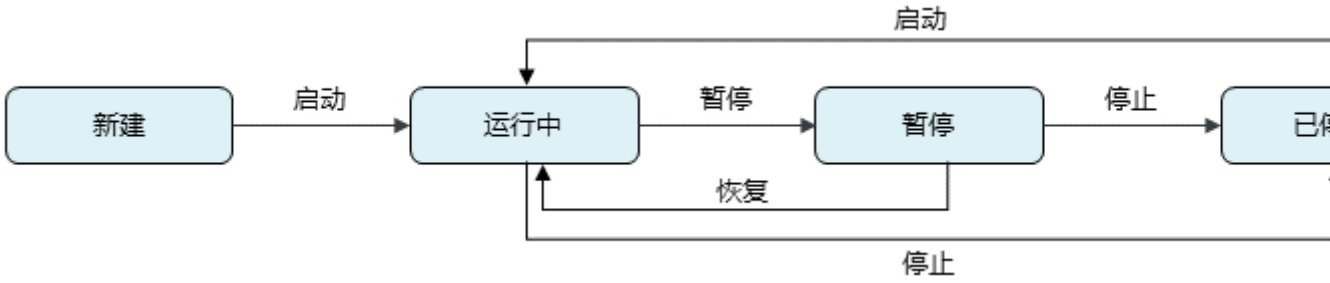
监控任务的状态如下表所示。

表 4-1: 监控任务状态

状态	描述
新建	首次建立任务显示该状态，对 新建 的任务可以单击启动。
运行中	表示任务正在运行，对 运行中 的任务可以单击停止或暂停。
暂停	单击暂停之后任务变为该状态，对 暂停 的任务可以单击恢复或停止。
已停止	表示任务停止运行，对 已停止 的任务可以单击启动。

任务的各种状态之间的转化关系如图所示。

图 4-2: 监控任务状态转化关系

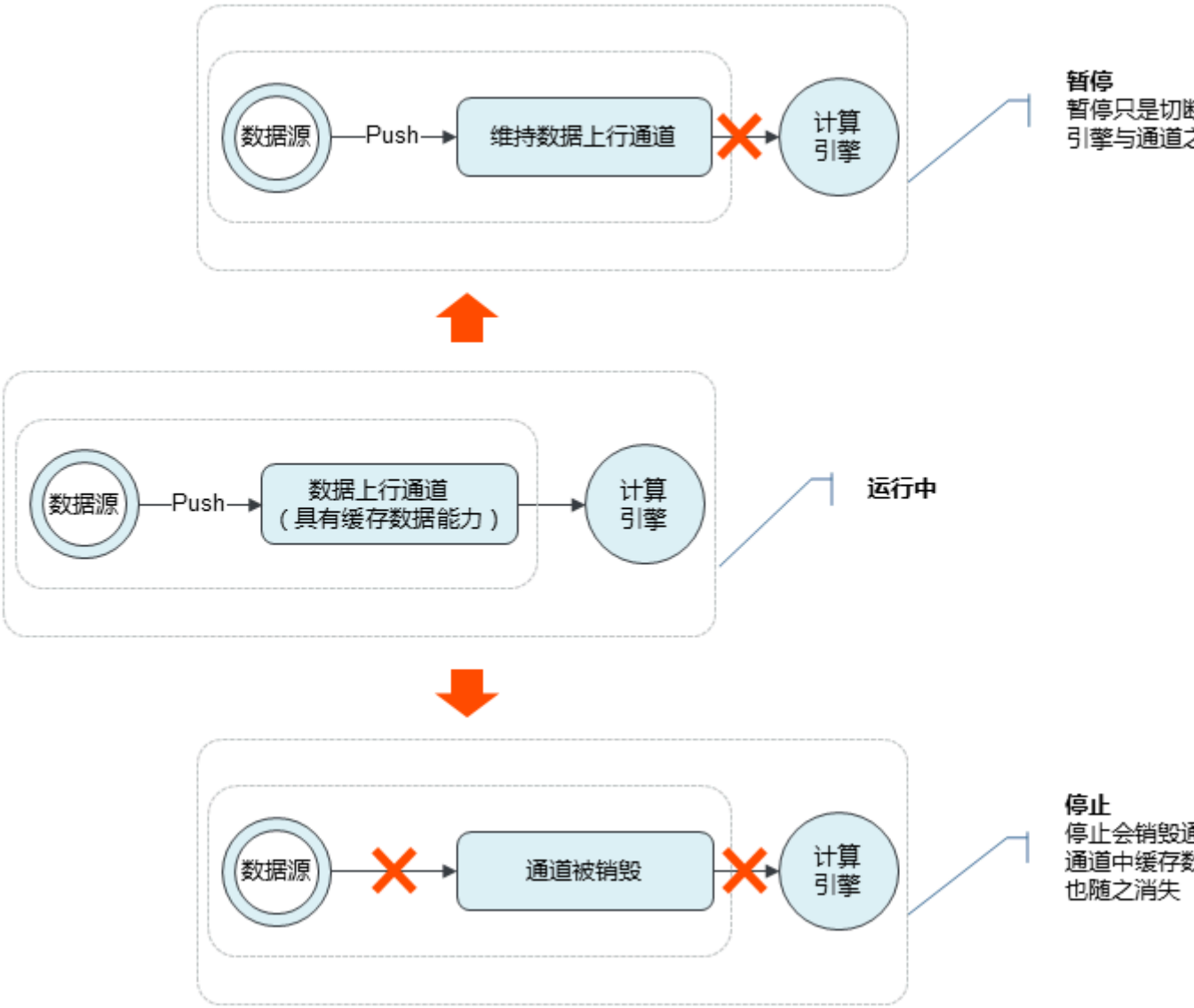


暂停与停止操作的区别

暂停操作将会切断计算引擎与通道之间的连接，单击恢复将会重新建立连接，并从上次的消费位点继续消费数据。

停止操作则会销毁通道。

图 4-3: 暂停和停止操作的区别



复制任务

对目标任务单击更多 > 复制任务，并填写新任务名称，即可制作该任务的复本。

删除任务

对目标任务单击更多 > 删除，并在删除监控任务对话框中单击删除，即可删除该任务。

> **说明:**
>
> · 状态为 运行中 或 暂停 的任务不可删除，状态为 新建 或 已停止 的任务可以删除。

· 删除任务的同时也将删除任务相关的配置、数据集、报表与报警。

导出任务模板

对目标任务单击更多 > 导出模板，并在导出任务对话框中复制全部代码，即可将任务导出为 JSON 格式的模板。

一个完整的任务包含数据源配置、日志清洗配置、数据集和/或报警配置。导出模板中包含清洗配置（日志样例与清洗规则）和数据集配置。

操作错误信息说明

在对任务进行启动、停止、暂停或恢复操作时可能会发生错误。以下为提示信息和解决方法。

提示	描述	解决方法
成功	操作成功	无
数据源配置不能为空	当前任务没有配置数据源，例如未选择数据源类型，或未选择数据源	进行数据源配置
切分配置为空	样例日志不存在，或者没有对样例日志进行切分	检查切分模型并完成配置
数据集或者报警不能为空	用户没有配置任何业务逻辑	为任务配置报警或者数据集
操作频繁	系统默认的操作时间最小间隔为 15 秒	等待 15 秒后重试
数据集中字段存在与切分模型之中	数据集中选择的字段在切分模型中无法找到	检查切分模型或者数据集
无法分配资源	系统无法为该任务分配资源	请联系管理员
错误	其他原因，例如网络原因、系统故障等	请联系管理员

4.3 创建和编辑映射表

映射表可用于对清洗出的字段进行静态 Join 操作来获取需要的目标字段，也可以在查询数据集时用于组合查询。本文介绍如何创建和编辑映射表。

背景信息

映射表用于存储静态数据，其主要功能有：

· 清洗日志后，通过对清洗出的字段进行静态 Join 操作来获取需要的目标字段。

· 查询数据集时，可以结合映射表进行组合查询。

关于如何更好地使用映射表，请参见#unique_44。

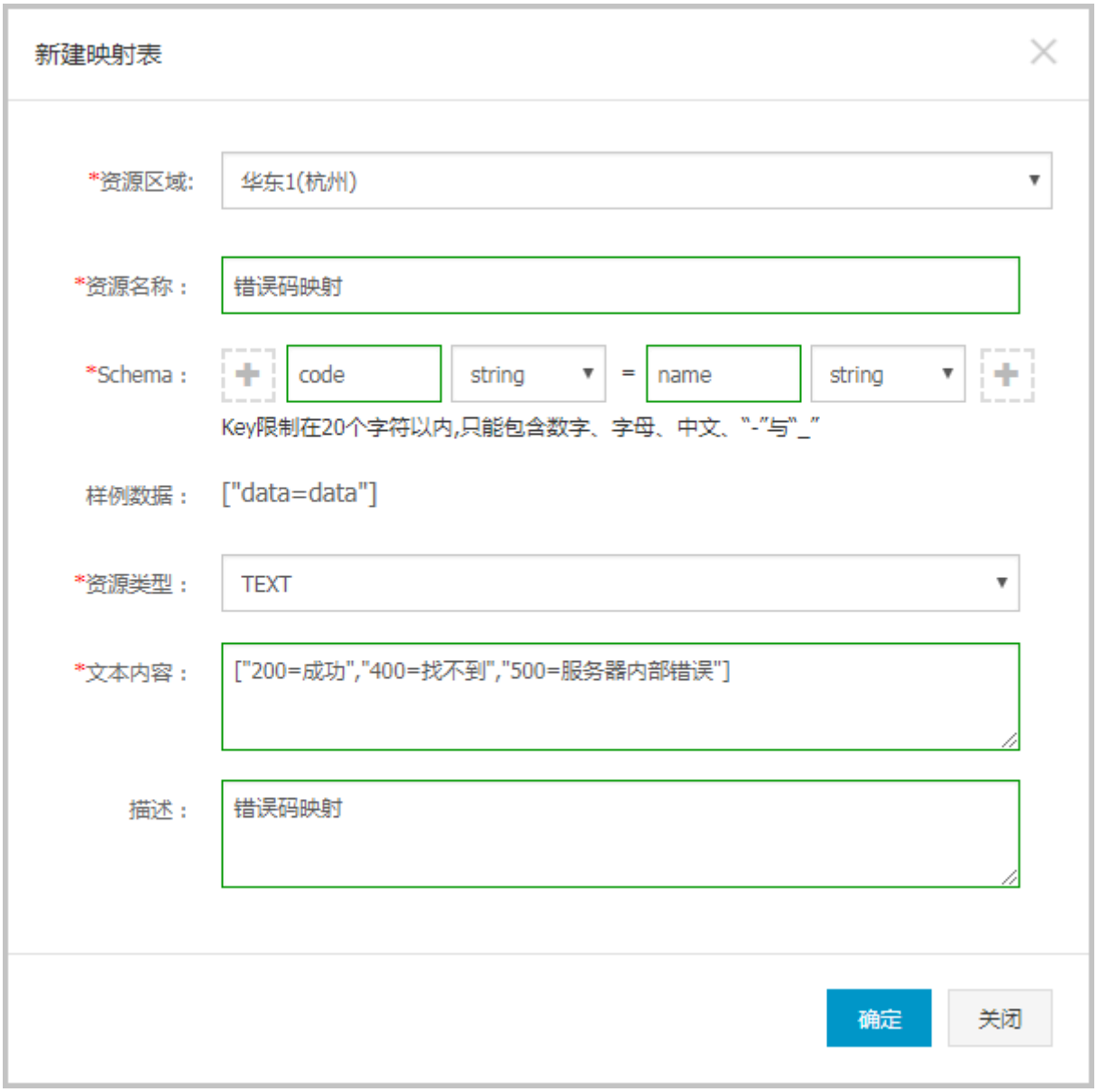
创建映射表

1. 在左侧导航栏中选择自定义监控 > 映射表管理。
2. 在映射表管理页面右上角，单击新建映射表。
3. 在新建映射表对话框中，选择资源地域，输入资源名称，在 Schema 区域定义映射关系，选择资源类型，并在文本内容中按照样例数据的格式输入映射内容，最后单击确定。

一对一映射关系

创建一个名为错误码映射的映射表，其 Schema 为一对一的映射关系，也就是将一个 code 字段转化为一个 name 字段。

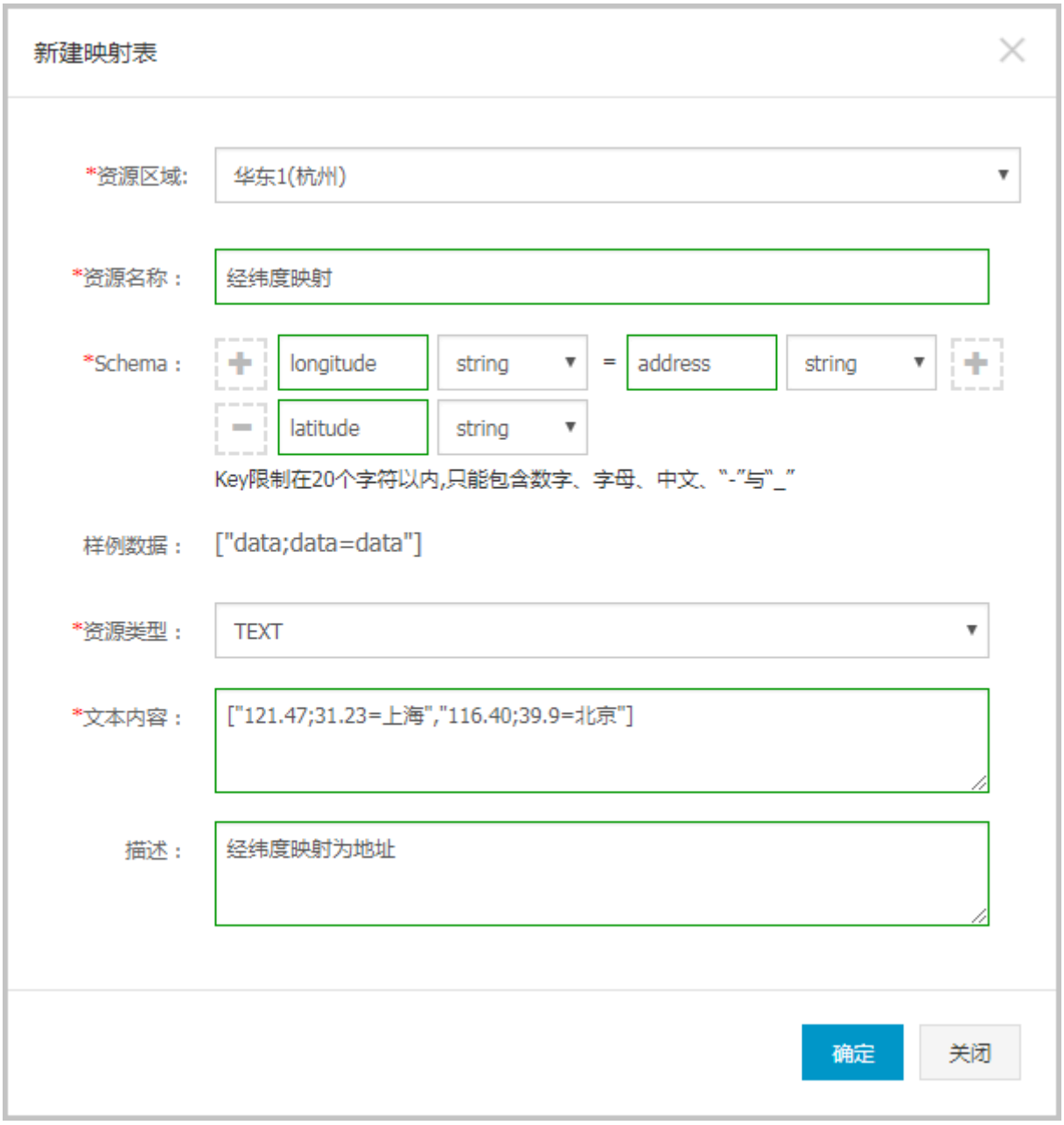
图 4-4: 创建映射表：一对一映射关系



多对一映射关系

下图的映射表为多对一的映射关系，将“经度 + 纬度”转化为地址。

图 4-5: 创建映射表：多对一映射关系



编辑映射表

1. 在左侧导航栏中选择自定义监控 > 映射表管理。

2. 在映射表管理页面的映射表列表中，单击目标映射表右侧操作列中的编辑，并在编辑映射表对话框中按需修改。

说明:

编辑模式下不能修改 Schema 和资源类型。如果资源类型是 TEXT，可以修改文本内容。修改之后，依赖映射表的任务或其他模块将立即生效。

参考

- #unique_44

4.4 配置监控任务

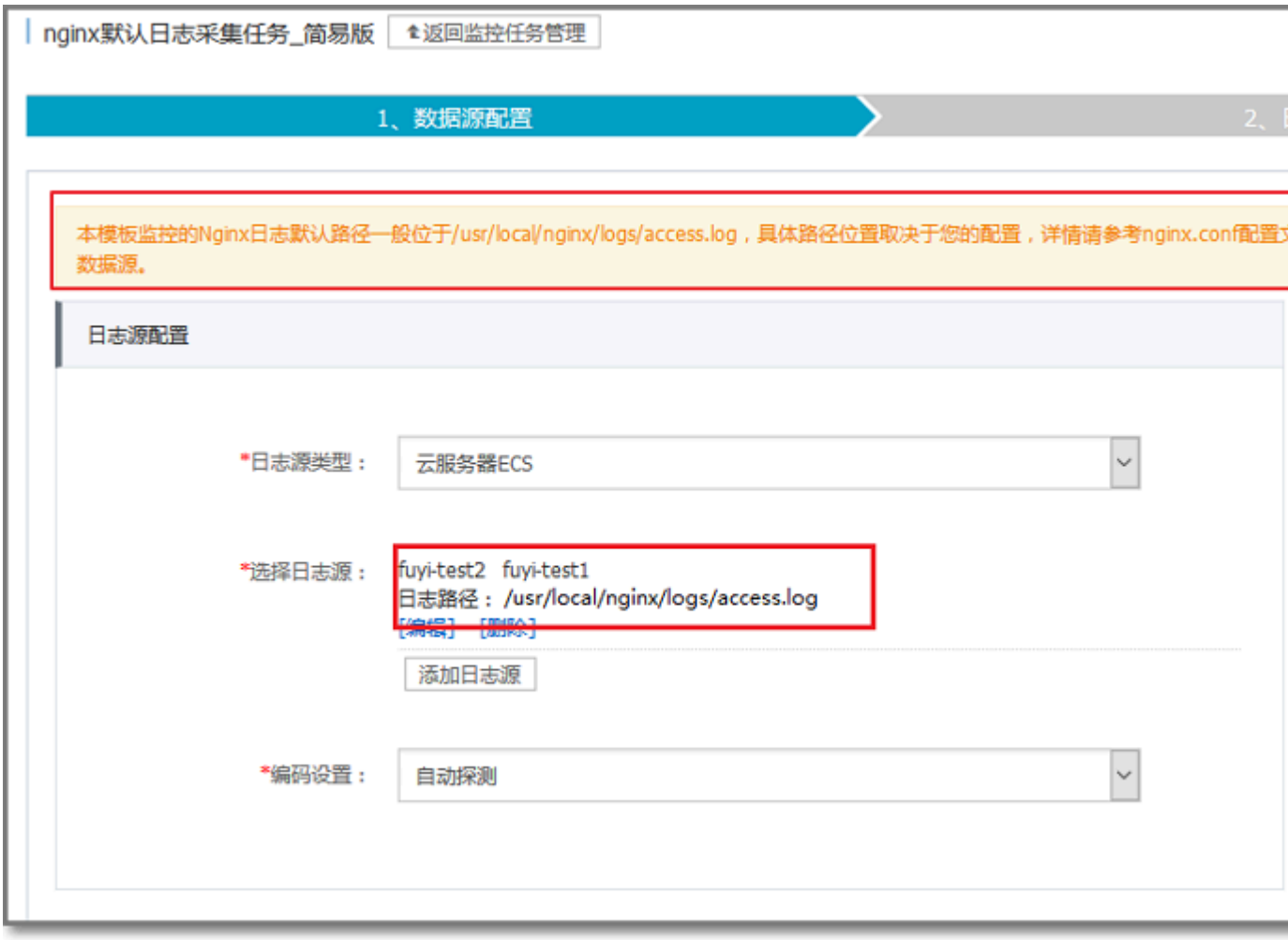
本文以 Nginx 监控任务模板为例，介绍了如何配置监控任务。

编辑数据源

标准方案模板的监控任务和自定义监控任务最大的区别在于，在数据清洗、数据集定义、报警等配置上，标准方案模板已有预先设定。您只需要重点关注数据源和其他定制需求。

在监控任务管理页面，单击监控任务右侧的编辑按钮，进入 Nginx 的标准方案模板任务编辑页面。其中，上部红框内显示的是模板任务相关的数据源说明。您可按照提示来设定数据源。

为了防止真实日志和模板中的日志有出入，建议单击日志抓取预览来确保相关日志数据的准确性。



修正数据清洗格式

在数据清洗页面，系统会根据标准方案模板生成对应的需要切分出的字段。以 Nginx 为例，由于该模板需要监控 PV、UV、调用返回值、客户端等相关信息，因此要求切分出以下字段：

- remote_addr：远程调用地址，在 nginx.conf 中用`$remote_addr`表示。
- time_local：时间戳，在 nginx.conf 中用`$time_local`表示。

- status：调用返回值，在 nginx.conf 中用`$status`表示。
- body_bytes_sent：返回字节长度，在 nginx.conf 中用`$body_bytes_sent`表示。
- http_referrer：被导航的 HTTP 原字段，在 nginx.conf 中用`$http_referrer`表示。
- user_agent：客户端程序版本，在 nginx.conf 中用`$user_agent`表示。

您需要根据自己的日志的实际使用情况，来定制相应字段的清洗切分规则。


nginx模板加强版v1_2
返回监控任务管理
1、数据源配置
日志抓取结果
如果数据样例框中没有您需要的日志内容，您也可以手动将您需要清洗的内容黏贴到数据样例
4288 - [16/Mar/2017:21:47:07 +0800] "POST http://arms.console.aliyun.
"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.4 (KHTML, like Gecko) C
8219 - [16/Mar/2017:21:47:08 +0800] "POST http://arms.console.aliyun.
"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.4 (KHTML, like Gecko) C
0851 - [16/Mar/2017:21:47:08 +0800] "POST http://arms.console.aliyun.
"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.4 (KHTML, like Gecko) C
3798 - [16/Mar/2017:21:47:24 +0800] "POST http://arms.console.aliyun.
"Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.4 (KHTML, like Gecko) C
本监控任务需要您针对nginx日志格式的配置做一些定制，需要确保将"$upstream_response_
以下是完整的nignx.conf中关于log_format的的例子：
log_format proxyformat "$remote_addr $request_time_usec $http_x_readtime [$time_local]
\"$http_user_agent\" \"$upstream_response_time\" \"$request_time\" \"$jsessionid\"";
请确保以下关键本模板的清洗逻辑默认需要切分出以下日志：
- source_ip: 远程调用地址，在nginx.conf 中用 $remote_addr 表示
- date: 时间时间戳，在nginx.conf 中用 $time_local 表示
- status: 调用返回值，在nginx.conf 中用 $status表示
- body_bytes_sent: 调用处理的字节长度，在nginx.conf 中用 $body_bytes_sent表示
- url: 用户调用的链接，在nginx.conf 中用$request_uri表示
- browser_type: 客户端程序版本，通过nginx.conf 中的$user_agent来判断。
- request_time: 代表nginx处理用户请求的相应时间，在nginx.conf中用$request_time表示，
- session_id： 代表session id的字段，需要用户定制。
智能切分
自定义切分

确认配置并启动

在下一步数据集与报警配置页面，可以在看到标准方案模板配置的规则。您可以在此确认以下信息：

- 数据集和报警功能是否存在所需字段缺失的情况。通常情况下，如果在上一步中缺少相应的清洗字段，将在该页面报错。
- 您也可以重点关注报警功能是否满足您的业务需求。例如相关服务调用 500 错误的报警等。

确认无误后，可完成配置并启动任务。任务启动后，即可在交互式大盘中查看数据。

5 使用教程

5.1 查询数据集

报表、报警和数据查询是数据集（Dataset）的展示形式，数据集是上述控件的数据本质。

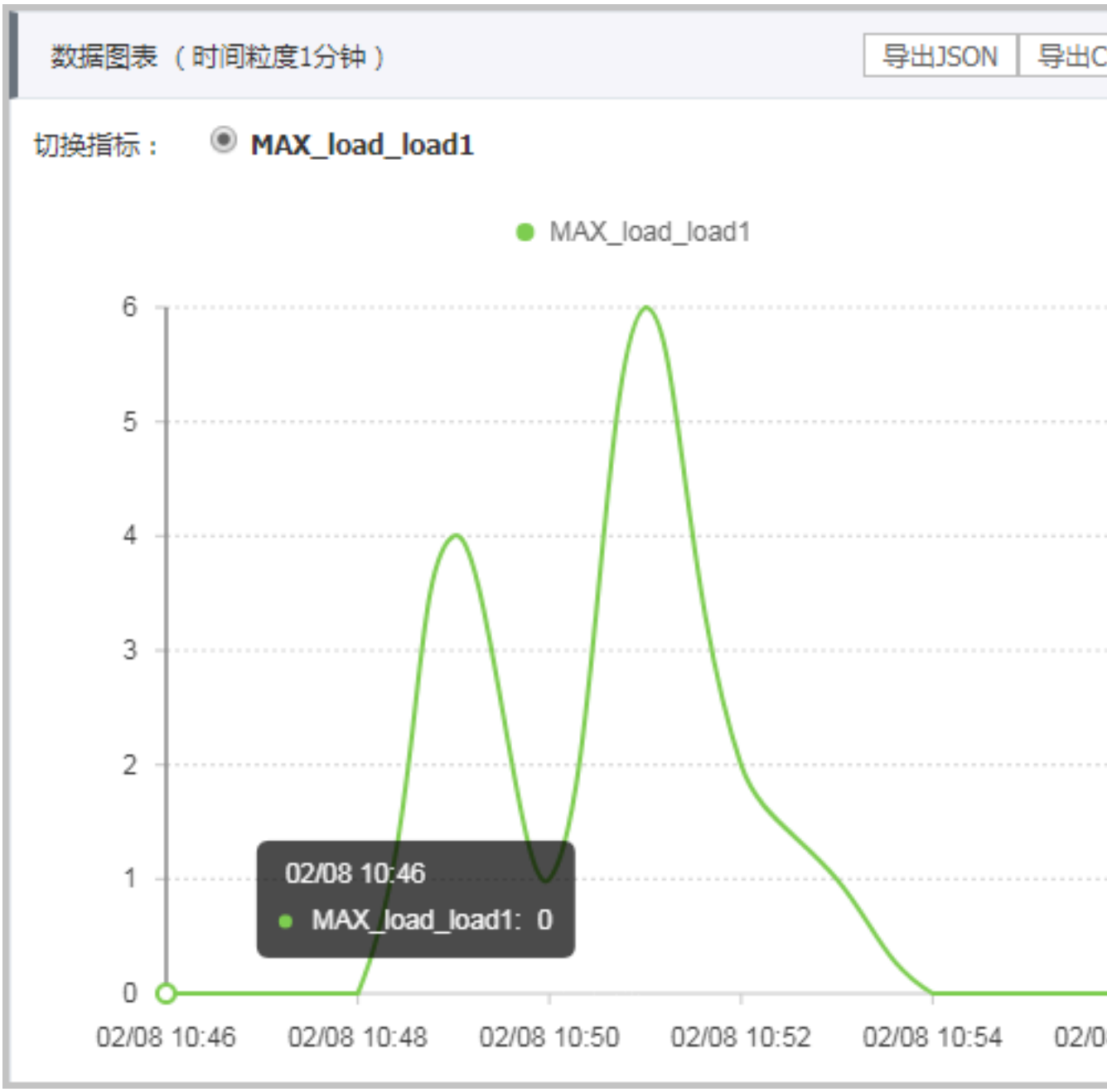
ARMS 以数据集的形式展示数据，屏蔽了底层数据存储结构、压缩存储、数据备份、数据查询机制等细节，因此您无需了解数据是如何存储在介质中的，只需要关注数据本身价值的实现。

1. 在控制台左侧导航栏中选择自定义监控 > 数据集管理。
2. 在数据集管理页面上，对目标数据集单击操作栏中的查询数据。

3. 在查询数据标签页上，输入数据集 ID、时间间隔、开始时间和截止时间，选择查询策略和补零策略（可选），然后单击查询。

   - 查询策略： 实时性优先会实时返回数据，但可能会因数据不齐全而不准确。齐全度优先会在获得准确数据后再展示数据。
   - 补零策略： 为数据中的 Null 值补零。
   - 高级配置： 映射表功能请参考文档创建和编辑映射表。

   数据会以数据图表的方式展示。



5.2 模式检测

本文介绍了如何使用模式检测。

背景信息

我们的应用往往部署在多台机器上，所有这些机器都在不断生成日志。这些日志就像应用的心跳一样，可以反映应用的健康状况。然而，应用的日志大多都是非结构化的，难以分析，而如果不加以分析，这些日志就不具备任何价值。为此，我们推出了模式检测功能，包含日志概览和日志对比模块。

日志概览

日志概览模块可通过模式发现、危险关键字提取、日志等级提取等方式，来主动发现日志中的异常。

前提条件

已在自定义监控任务中添加模式检测配置。

操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 模式检测。
2. 在模式检测页面上，单击规则名称栏中的规则名称，或操作栏中的详情，即可进入模式检测的概览页面。
3. 在时间选择框内选择起止时间，并单击搜索，即可显示该时间段内的日志总行数和异常数量等信息。
4. 在日志详情区域，选择日志的路径、类型，或输入关键字来显示符合条件的日志内容。

示例

下图展示了一个模式检测概览页面的示例。在此示例中，我们先将各类日志进行分类（选中的 PipelineException 出现了 483 次），再从分类后的日志中根据关键字提取出了 PipelineException、SSOException 两个异常。概览模块很直观地体现了这个应用的异常日志和正常日志的分布情况。


arms日常环境的日志
2018-04-12 00:00:00 - 2018-04-12 20:04:22
日志总行数
0
类型数
0
异常日志
0
类型数
0
5千
4千
3千
2千
1千
0
日志详情
已收藏3类 未收藏10类 已监控15类 未监
全部路径
SSOException
PipelineException
总数等级
模式

日志对比

在日志对比模块中可对比两个指定时间段的日志。例如，发布新代码时，可以通过对比日志来发现新代码造成了哪些新的问题。又如，通过对比今天和昨天的日志来发现今天是否出现了新的问题。

前提条件

已在自定义监控任务中添加模式检测配置。

操作步骤

1. 在控制台左侧导航栏中选择自定义监控 > 模式检测。
2. 在模式检测页面上，单击规则名称栏中的规则名称，或操作栏中的详情，即可进入模式检测的概览页面。

3. 在控制台左侧导航栏中选择日志对比，即可进入日志对比页面。


arms日常环境的日志
原始数据
2018-04-12 19:08:22
-
2018-04
关键字搜索
请输入关键字
基本信息
日志详情
次数：739
query: $_JSON_$
次数：447
ip is $_IPS_$ whil
次数：118
ArmsDB-App-$_K
次数：49
[Medusa Error] ca
次数：7
[Medusa Error] ca
次数：3
getEdasAccountB
次数：3
syn Edas app'size

4. 在原始数据和对比数据时间选择框内分别指定起止时间，然后单击执行对比，即可并排对比相同关键字在这两段时间的日志中出现的次数。

6 日志清洗进阶教程

6.1 日志清洗中的系统字段

本文介绍了日志清洗中的主要系统字段_line、_hostIp和_sysTime。

在日志清洗环节中，除了您定义的字段，系统还会添加一些默认字段，主要包括以下字段。

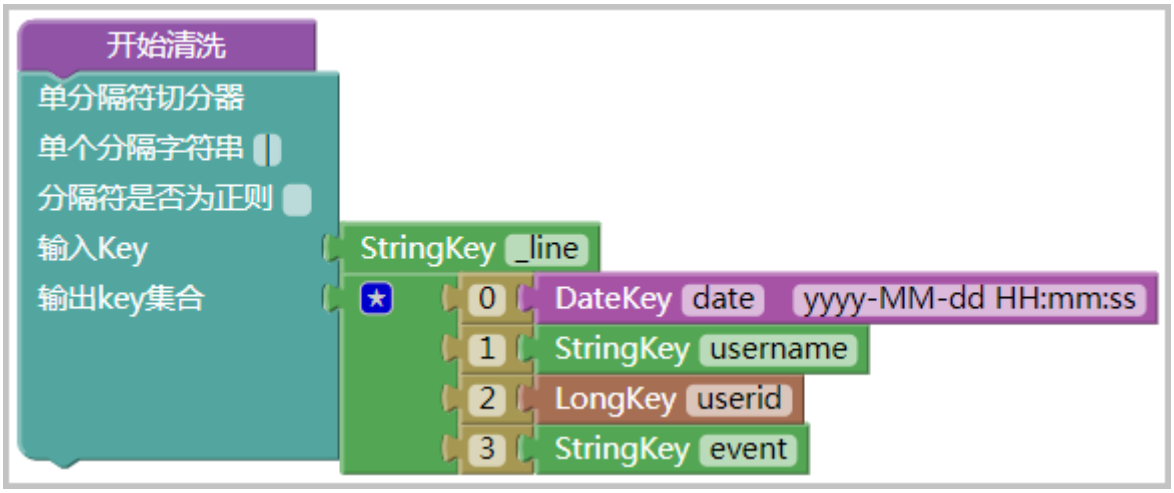
- _line

  _line 字段表示每一行日志。

  对于以下日志：

```
2016-11-08 11:00:01|user_abc|123456|下单
2016-11-08 11:00:02|user_abc|123456|支付
2016-11-08 11:10:01|user_abc|123456|退货
```

  首次切分时，输入 key 为每一行日志的 _line，自定义切分形式如下图所示。



- _hostIp

  _hostIp 字段表示每一行日志的来源 IP。各类型数据源对该字段的支持情况如下表所示。

数据源	是否支持_hostIp
StarAgent 数据源	支持
鹊桥数据源	支持

数据源	是否支持_hostIp
MQ 数据源	不支持

目前仅自定义切分模式提供 _hostIp，智能切分模式不提供。以上方日志为例，单击日志切分预览后如下图所示。

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2016-11-08 11:00:01|user_abc|123...
_sysTime	date	1528791630431
date	date	1478574001000
event	string	下单
userid	long	123456
username	string	user_abc

图中的 _hostIp 字段为 127.0.0.1。

> 说明:
> 单击日志切分预览后，无论是什么数据源，_hostIp 字段均为 127.0.0.1，因为是本地模式，当任务真正运行时会产生真实数据。

- _sysTime

  _sysTime 字段表示日志的处理时间，如果您的日志中没有自己的业务时间，可以选择 _sysTime 时间字段进行聚合计算。

6.2 智能切分可识别的时间格式

智能切分会自动识别特定格式的时间字段，本文详细说明了这些时间格式。

首先，时间必须有精确的“年月日时分秒”字段。此外，您可以按照以下格式添加可选的毫秒字段和时区字段。关于时间格式的规范，请参考 SimpleDateFormat 官方文档说明。

常见时间格式

常见时间格式是指日志中常见的时间格式，通常是以规律分隔符隔开的“年月日时分秒”数据。“年月日”与“时分秒”之间可用分隔符或者大写字母“T”隔开。“秒”字段后可用分隔符“,”或“.”加上“毫秒”字段。最后可加上标准时区字段。

- yyyy-MM-ddTHH:mm:ss

  “T”可替换为空格或其他分隔符，“:”可替换为其他分隔符，“年月日”与“时分秒”之间的分隔符可替换为“/”或“-”等常用分隔符。例如：

```
1963-04-17T23:51:12
1963/04/17 12:12:53
```

- yyyy-MM-dd HH:mm:ss,SSS

  “,”可替换为“.”，其他位置可替换的字符同上。例如：

```
1963/04/17 12:12:53,172
1963/04/17T12:12:53.172
```

- yyyy-MM-dd HH:mm:ssZ

  例如：

```
1963/04/17 12:12:53+0800
1963/04/17T12:12:53-0700
```

- yyyy-MM-dd HH:mm:ss,SSS

  例如：

```
1963/04/17 12:12:53,999+0800
1963/04/17T12:12:53.878-0700
```

固定标准时间格式

固定标准时间格式是指 SimpleDateFormat 中能翻译成日期正则表达式的较常用时间格式，例如“年月日”之间没有分隔符的时间格式，以及“月日年”形式的常用时间格式等。

- 类型一：日期和时间之中均无分隔符。
  - yyyyMMddHHmmss

    ```
    19551020233546
    ```

  - yyyyMMdd HHmmss

    ```
    19551020 233546
    ```

- 类型二：时间格式倒序，“日”在前、“月”在中、“年”在后。

  dd-MM-yyyy HH:mm:ss

  ```
  17-04-1963 23:59:23
  ```

SimpleDateFormat 中的部分常见时间格式

详情请参考 SimpleDateFormat 官方文档说明。

- yyyy.MM.dd G 'at' HH:mm:ss z

```
2001.07.04 AD at 12:08:56 PDT
```

- EEE, MMM d, "yy

```
Wed, Jul 4, '01
```

- yyyyy.MMMMM.dd GGG hh:mm aaa

```
02001.July.04 AD 12:08 PM
```

- EEE, d MMM yyyy HH:mm:ss Z

```
Wed, 4 Jul 2001 12:08:56 -0700
```

- yyMMddHHmmssZ

```
010704120856-0700
```

- yyyy-MM-dd'T'HH:mm:ss.SSSZ

```
2001-07-04T12:08:56.235-0700
```

- yyyy-MM-dd'T'HH:mm:ss.SSSXXX

```
2001-07-04T12:08:56.235-07:00
```

其他特定时间格式

例如 Nginx 时间格式：

```
25/Aug/2012:16:41:07 +0800
```

6.3 自定义切分的使用

除了智能切分器，ARMS 还提供了自定义切分器，用于自定义日志清洗流程来满足复杂的切分需求。

自定义清洗配置器采用所见即所得的可视化配置方式，本文档将详细介绍如何使用自定义清洗配置器。

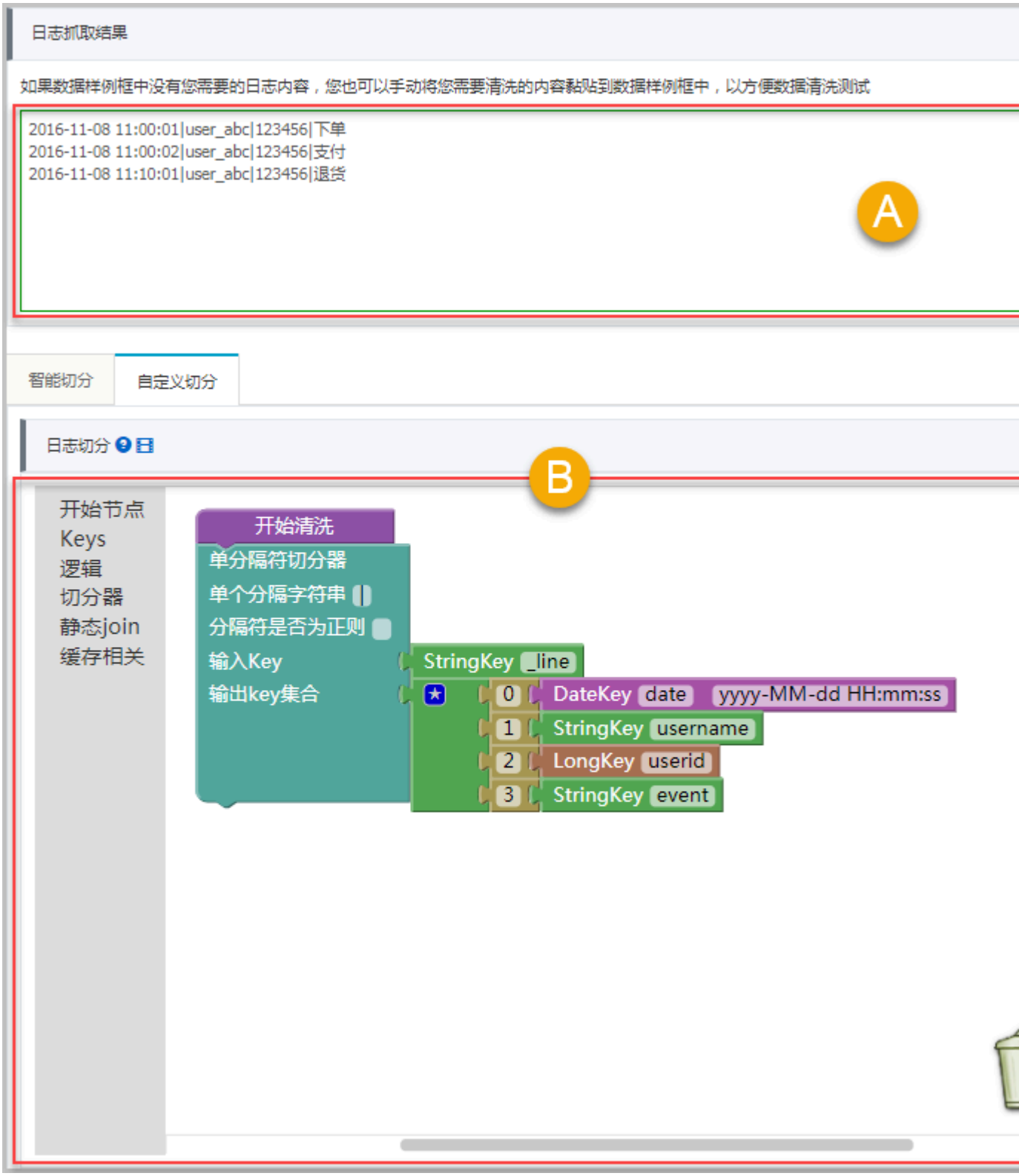
配置界面简介

自定义清洗配置界面包含三个部分：

- A：样例日志区域
- B：可视化清洗流程编辑区域

· C：清洗结果预览区域

图 6-1: 自定义切分配置页面



在可视化流程编辑区域（以下简称“编辑区”），可以使用一个或多个“积木块”组装出日志清洗的逻辑，将原始日志区域中的日志“清洗”为结构化的键值对（Key-Value Pair）。

在编辑区编辑的过程中，可以随时单击清洗结果预览区域上方的日志切分预览按钮，预览每一行原始日志通过当前编辑区的流程清洗后的键值对结果。

在通过预览结果确认清洗流程准确无误后，单击页面下方的下一步按钮启动任务。ARMS 将使用在编辑区保存的流程来处理每一条从数据源中消费的日志。

示例一：简单的日志清洗流程

样例日志：

```
2016-11-08 11:00:01|user_abc|123456|下单
2016-11-08 11:00:02|user_abc|123456|支付
2016-11-08 11:10:01|user_abc|123456|退货
```

以第一行数据为例，假设需要切分出以下键值对：

Key 类型	Key	Value
Date	date	2016-11-08 11:00:01
String	username	user_abc
Long	userid	123456
String	event	下单

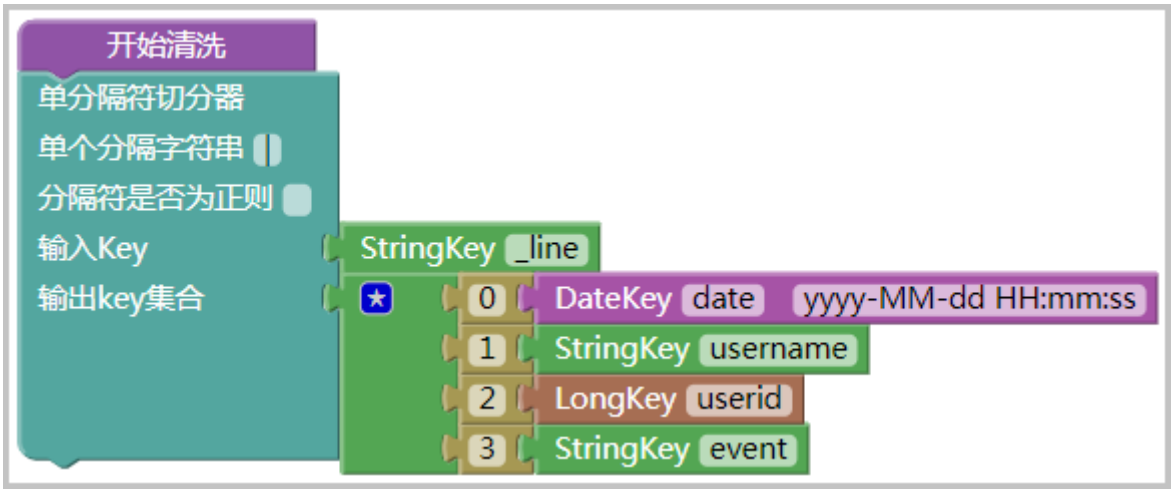
通过观察发现，键值对之间均使用竖线（|）分隔，因此可以使用单分隔符切分器按照以下步骤切分键值对。

1. 在编辑区左侧的工具栏中，单击切分器，将单分隔符切分器拖拽至编辑区。

说明：

切分器积木块必须连接在开始清洗模块上才能生效。

图 6-2: 简单的日志清洗流程



切分逻辑说明：

- **以竖线（|）为分隔符，以 `_line` 作为入参（`_line` 代表原始日志行本身）。**
- **切分后 0 号位置的字符串按照 `yyyy-MM-dd HH:mm:ss` 的格式转为 date （内部实际将以 Long 型保存）。**
- **1 号位置的字符串转为以 username 为 Key 的字符串。**
- **2 号位置转为以 userid 为 Key 的 Long。**
- **3 号位置转为以 event 为 Key 的字符串。**

2. 单击日志切分预览，可以看到每行日志的切分结果如下：

图 6-3: 单分隔符切分器切分结果

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2016-11-08 11:00:01|user_abc|123...
_sysTime	date	1528791630431
date	date	1478574001000
event	string	下单
userid	long	123456
username	string	user_abc

3. 单击保存或下一步，令清洗流程生效。

到此为止，一个简单的日志切分流程就配置完成了。

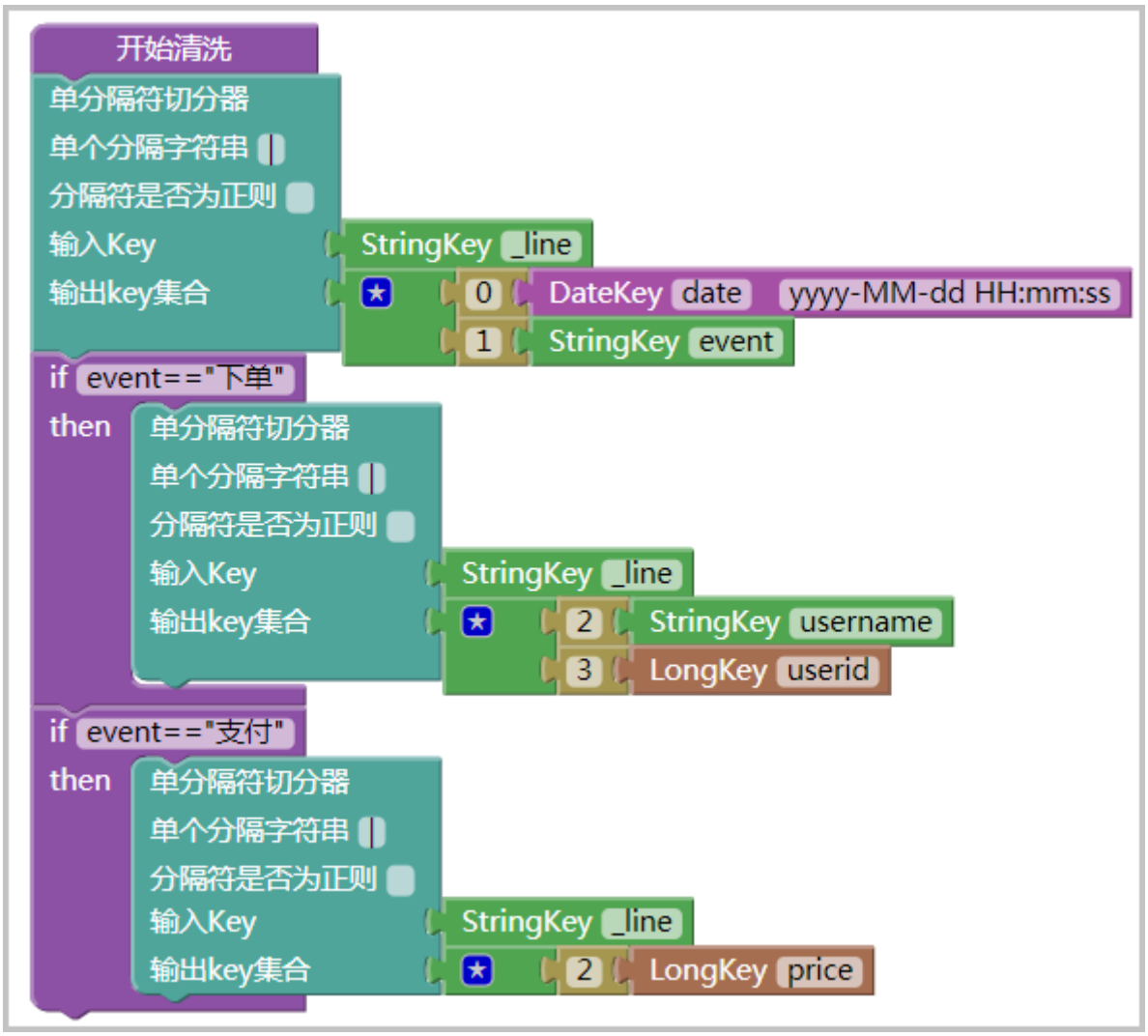
示例二：复杂的日志清洗流程

样例日志：

```
2016-11-08 11:00:01|下单|user_abc|123456
2016-11-08 11:00:02|支付|200
2016-11-08 11:10:01|退货
```

在此示例中，不同行上的日志由于操作类型不一样，字段的个数、同位置各字段的含义都不同（这在业务系统中非常常见）。这时可以通过 if-then/if-else 逻辑区分不同类型的日志。

图 6-4: 复杂的日志清洗流程



切分逻辑说明：

- 首先使用单分隔符切分器切分出 `date` 与 `event` 字段。

- 接着使用逻辑模块中的 if-then 模块，针对不同的 event 使用不同的单分隔符切分器切分出不同的字段。

图 6-5: 第 1 行日志的切分结果

第1行日志切分结果：

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2016-11-08 11:00:01|下单|user_ab...
_sysTime	date	1524894670741
date	date	1478574001000
event	string	下单
userid	long	123456
username	string	user_abc

图 6-6: 第 2 行日志的切分结果

第2行日志切分结果：

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2016-11-08 11:00:02|支付|200
_sysTime	date	1524894670741
date	date	1478574002000
event	string	支付
price	long	200

第一行与第二行切分出的键值对组合是不同的。第一行中有 `username` 和 `userid`，而第二行中有 `price`。

更多信息

- #unique_29
- #unique_53

6.4 内置切分器

本文介绍了 ARMS 的内置切分器，包括单分隔符、多分隔符、顺序、KV、JSON 等多种切分器。您可以针对不同的场景单独或组合使用这些切分器。

单分隔符切分器

说明

将单一字符串作为输入，当匹配到用户指定的分隔字符串时进行切分，切分结果为多个键值对（Key-Value）。

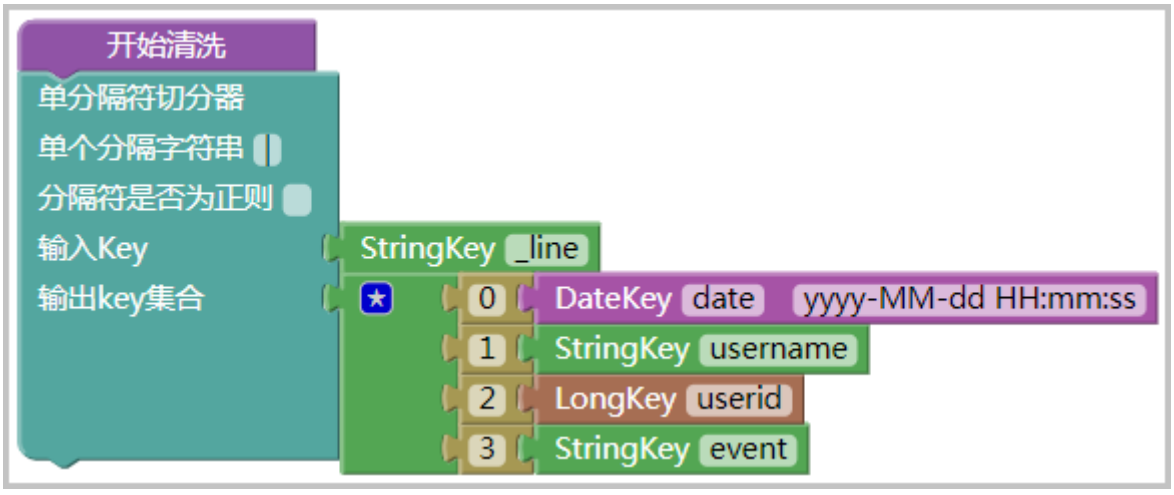
示例

样例日志：

```
2016-11-08 11:00:01|user_abc|123456|下单
2016-11-08 11:00:02|user_abc|123456|支付
2016-11-08 11:10:01|user_abc|123456|退货
```

自定义切分逻辑：

图 6-7: 单分隔符切分器用法示例



切分逻辑说明：

- 以竖线（|）为分隔符，以 `_line` 作为入参（`_line` 代表原始日志行本身）。
- 切分后 0 号位置的字符串按照 `yyyy-MM-dd HH:mm:ss` 的格式转为 date （内部实际将以 Long 型保存）。

- 1 号位置的字符串转为以 username 为 Key 的字符串。
- 2 号位置转为以 userid 为 Key 的 Long。
- 3 号位置转为以 event 为 Key 的字符串。

> 说明:
>
> - 对不可见字符的支持：本切分器支持不可见字符的切分，例如 \t，\u001f（ASCII char=31）。因为无法在浏览器里粘贴不可见字符，所以您只能暂时将测试文本的分隔符替换为可见字符进行测试。全部流程测试通过后，再将分隔符替换为不可见字符进行部署。
> - 多字符分隔符：本切分器分隔符可以是多字符。例如以 `a::b::c` 为输入，以 `::` 为分隔符切分。
> - 切分模型包含 `_line` 字段和 `_hostIp` 字段。详情请参见#unique_55。

多分隔符切分器

说明

将单一字符串作为输入，当匹配到用户指定的任意一个分隔字符串时进行切分，切分结果为多个键值对（Key-Value）。

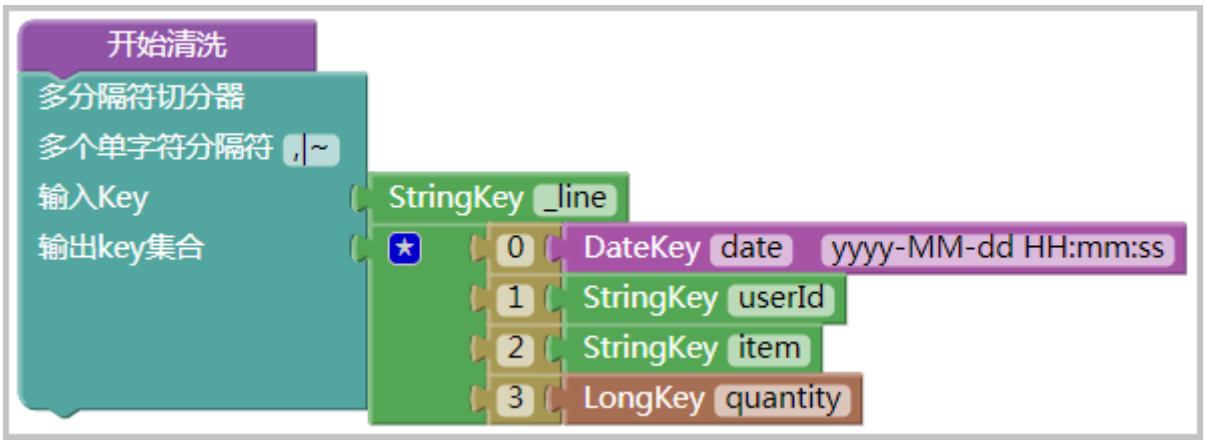
示例

样例日志:

```
2014-07-25 17:25:00,aaa|b~1
2014-07-25 17:26:00,aaa|b~1
```

自定义切分逻辑:

图 6-8: 多分隔符切分器用法示例



切分结果：

图 6-9: 多分隔符切分器切分结果

第1行日志切分结果：

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2014-07-25 17:25:00,aaa|b~1
_sysTime	date	1524822509464
date	date	1406280300000
item	string	b
quantity	long	1
userId	string	aaa

顺序分隔符切分器

说明

对于较复杂的切分场景（例如 accesslog），可以使用 ARMS 的顺序分隔符切分器，以 O(n) 的复杂度切分日志。

示例

样例日志：

```
117.xx.xx.xx 835158 - [26 May 2014:14:05:28 +0800] "GET http://trade
.taobao.com/trade/detail/trade_snap.htm?spm=a1z0f.2.100003.9.8BpicQ&
trade_id=664793864233811" 302 0
117.xx.xx.xx 835158 - [26 May 2014:14:05:28 +0800] "GET http://trade
.taobao.com/trade/detail/trade_snap.htm?spm=a1z0f.2.100003.9.8BpicQ&
trade_id=664793864233811" 302 0
```

需要切分出的字段有：

- **IP：117.xx.xx.xx**
- **平均响应时间：835158**
- **时间：1401084328000**

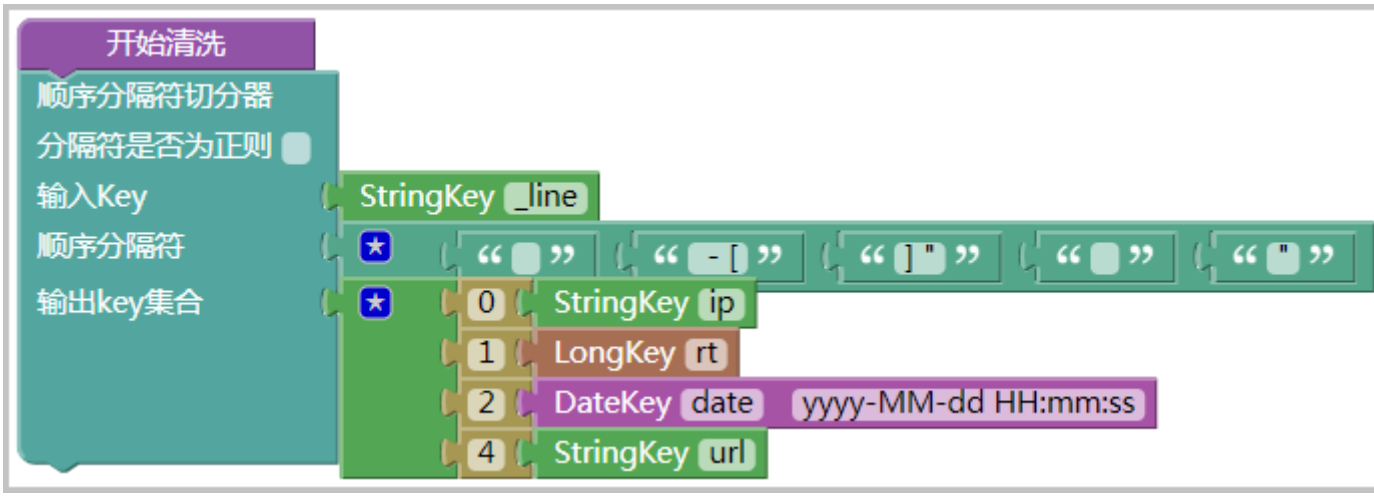
- 请求的 URL：`http://trade.taobao.com/trade/detail/trade_snap.htm?spm=a1z0f.2.100003.9.8BpicQ&trade_id=664793864233811`

我们会发现：

- 单分隔符切分器显然无法满足需求。
- 多分隔符切分器也不能满足需求。如果要切分出 rt=835158 这一对 KV，必须以空格作为切分字符之一，但是这样做会错误地把 `26 May 2014:14:05:28 +0800` 这个字符串切分成几段，因此我们将无法得到 date - 1401084328000。
- 如果使用正则表达式来匹配 KV，则需要遍历多次，效率太低。

ARMS 提供了顺序分隔符切分器，可以满足本示例的需求。切分逻辑：

图 6-10: 顺序分隔符切分器用法示例



切分逻辑说明：

先定义了 5 个分隔符字符串，并依次将输入字符串切分为了 6 段子串，随后将指定的子串分别赋给不同的 key，即可达到目标切分效果。

KV 切分器

说明

如果需要切分含有多组 KV 的日志，且各个 KV 的位置不固定时，可以使用 KV 切分器。

示例 1

样例日志：

```
key1=aaaa;key2=bbbb;key3=cccc;key4=dddd;....
```

```
key1=aaaa;key2=bbbb;key3=cccc;key4=dddd;....
```

切分逻辑：

图 6-11: KV 切分器用法示例 1

说明：

- 如果没有配置文本字符与 Key 的映射，KV 切分器将视所有 Key 为“未定义 key”，所有 Key 对应的 Value 将被切分为 StringKey 类型。
- 如果选择取消勾选是否切分未定义的 Key，那么未定义 Key 将不会被切分。换言之，如果没有配置文本字符与 Key 的映射，并且未勾选是否切分未定义的 Key，则 KV 切分器将不做任何处理。

示例 2

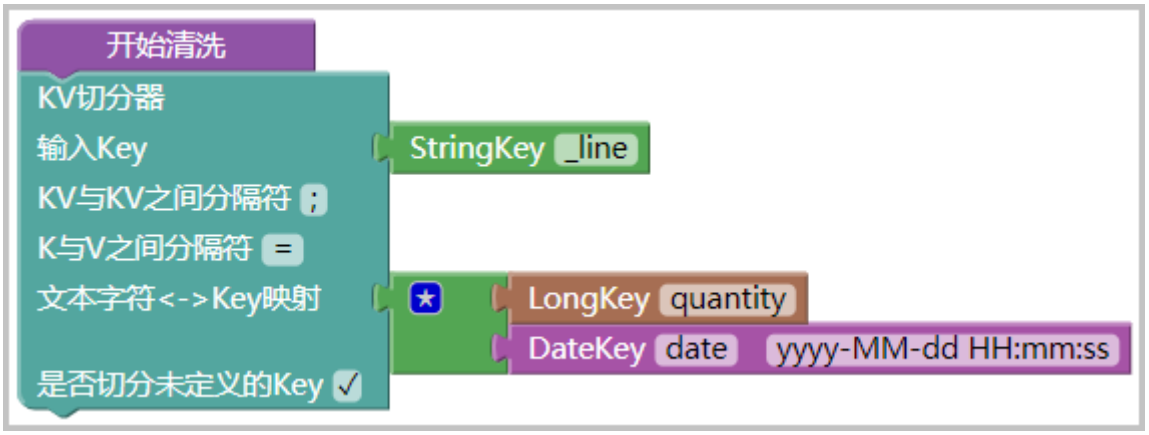
以下是一个需要将指定 Key 转换成对应类型的示例：

样例日志：

```
name=abc;item=iphone6;quantity=15;date=2014-07-25 17:25:00;....
name=abc;item=iphone6;quantity=15;date=2014-07-25 17:25:00;....
```

目标是将 quantity 变为 LongKey，date 变为 DateKey。切分逻辑：

图 6-12: KV 切分器用法示例 2



JSON 切分器

说明

如果日志中包含 JSON 字符串，使用 JSON 切分器可以快速切分出该 JSON 中的叶子节点。

示例 1：全自动切分

样例日志：

```
{
"title": "Example Schema",
"type": "object",
"quantity": 200,
"date":"2015-12-12 12:12:12",
"properties": {
        "firstName": {
          "type": "string"
        },
        "lastName": {
          "type": "string"
        },
        "age":
            "description": "Age in years",
            "type": "integer",
            "minimum": 0
        }
    },
"required": ["firstName", "lastName"]
```

```
}
```

将该 JSON 放在一行文本中，使用 JSON 切分器进行切分。切分逻辑：

图 6-13: JSON 切分器使用示例 1

切分结果：

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	{ "title": "Example Schema", "type"...
_sysTime	date	1524827053782
date	string	2015-12-12 12:12:12
description	string	Age in years
minimum	long	0
quantity	long	200
title	string	Example Schema
type	string	string

切分逻辑说明：

由于勾选了是否切分未定义的 Key，JSON 切分器将自动切出所有非 Array 类型的叶子节点，并自动转换成对应类型（StringKey 或 LongKey）。

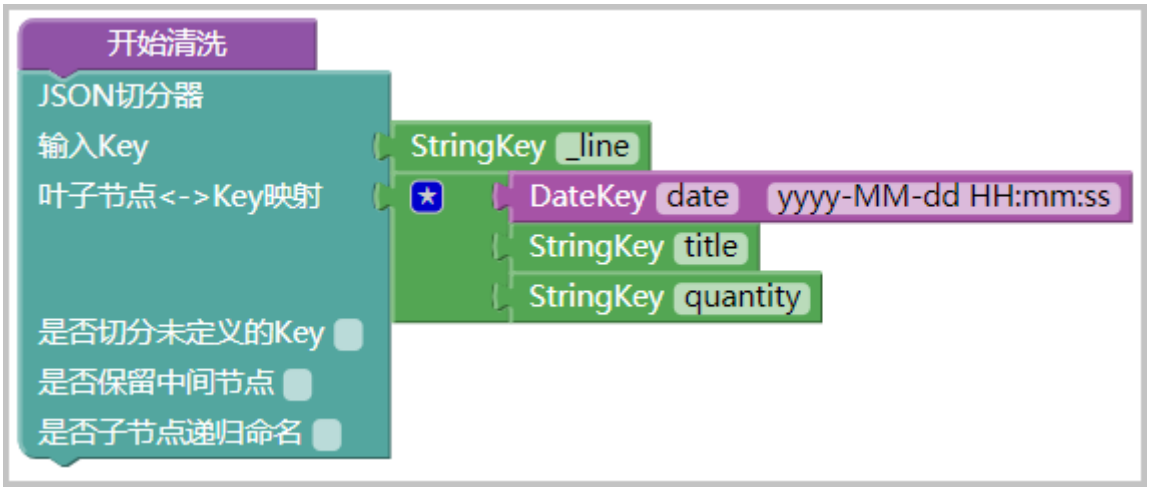
说明：

本例中的 `required` 字段由于是 Array 类型，所以没有切分。另外，`date` 字段默认转化为 String 类型。

示例 2：自定义切分

如果只需要切分 JSON 中的指定字段，可以这样配置切分逻辑：

图 6-14: JSON 切分器使用示例 2



切分结果：

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	{ "title": "Example Schema", "type"...
_sysTime	date	1524828377774
date	date	1449893532000
quantity	string	200
title	string	Example Schema

切分逻辑说明：

本例中未勾选是否切分未定义的 Key，并自定义了 `title`、`date` 和 `quantity` 三个字段，且指定了类型。因此 ARMS 会只切分自定义的字段，并转换成对应的类型（本例中 `date` 字段被转成了 ARMS 内置的 Long 型）。

更多信息

- #unique_26
- #unique_28
- #unique_53

6.5 自定义切分中 if/assign 的语法介绍

ARMS 默认支持 Aviator 表达式引擎，您可以完成对数据处理时的逻辑控制、赋值操作等。

条件判断 if-else

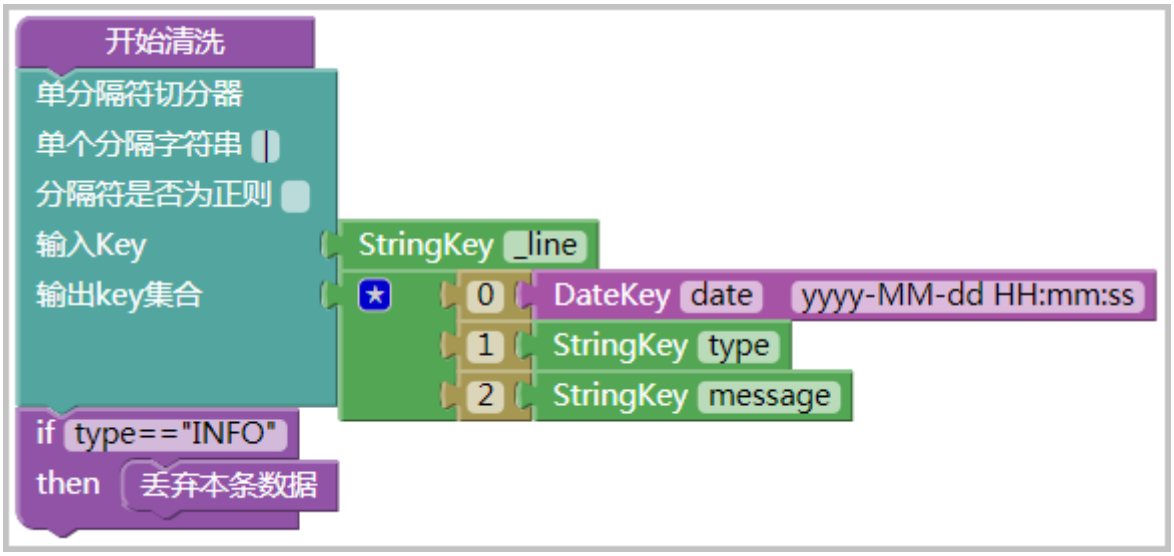
在自定义切分标签页的左侧操作面板中选择逻辑，系统提供两种类型的逻辑判断积木块“if/else”和“if”，如下图所示。

下图演示一些简单的逻辑表达式使用例子，用户日志如下所示。

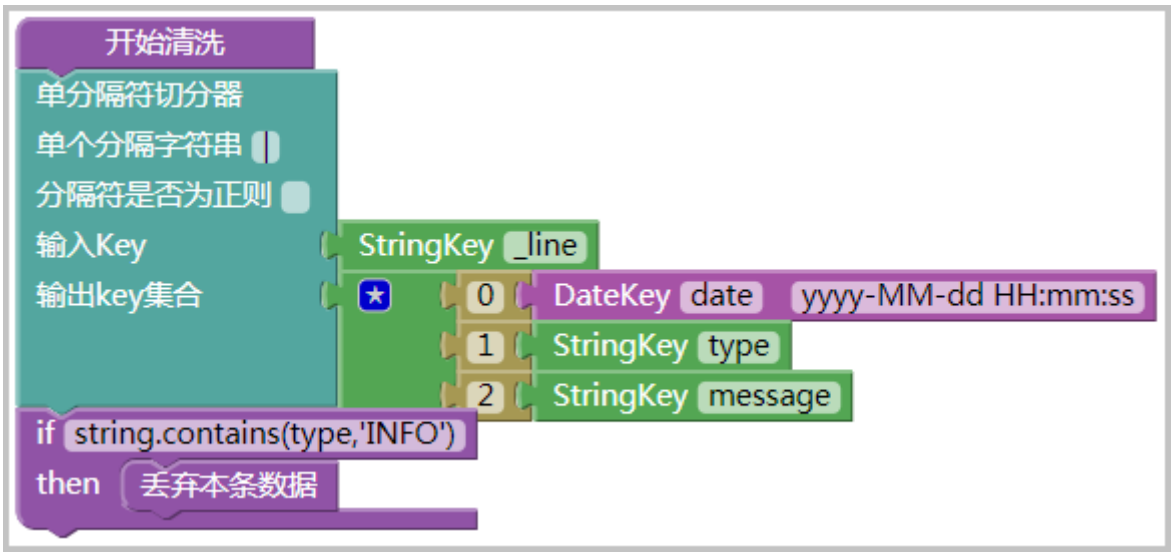
```
2017-01-09 16:02:49|ERROR|it is error...
2017-01-09 16:03:49|INFO|it is ok...
2017-01-09 16:03:49|INFO_1|it is ok...
```

日志按照“|”切分之后的字段分别为`date`、`type`、`message`。

示例一：当`type`为 INFO 时，该行日志直接丢弃；



示例二：当`type`包含 INFO 的时候，该行日志丢弃；

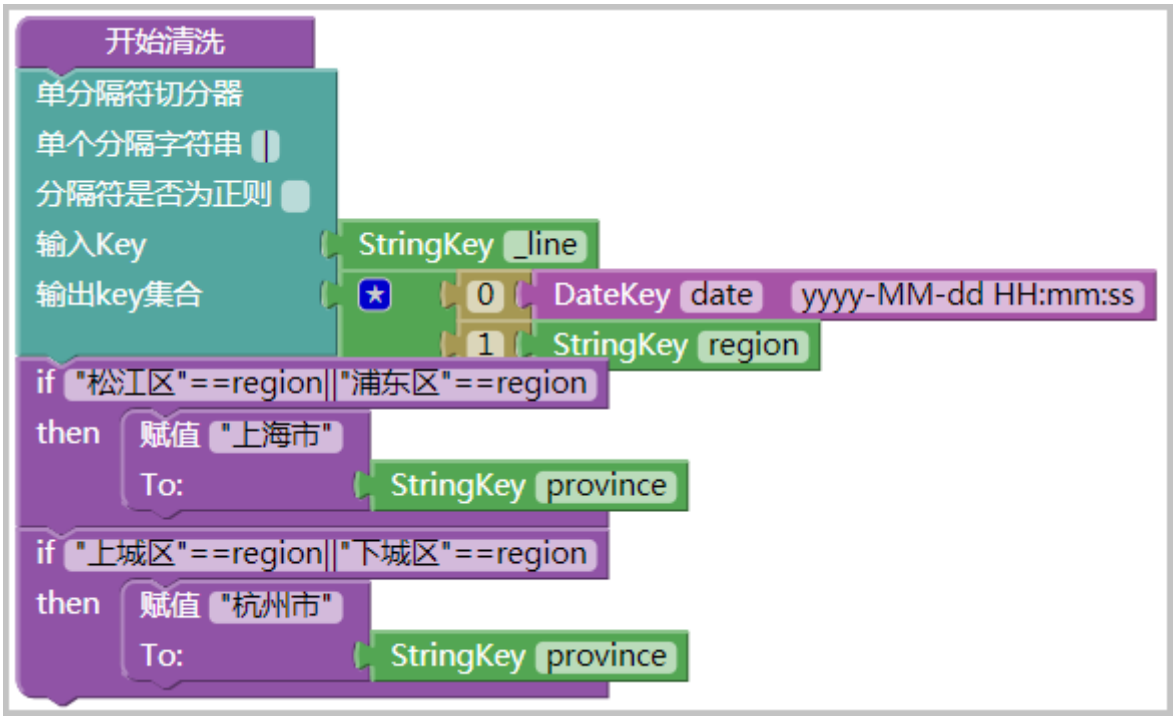


赋值

接下来演示一个简单的赋值使用例子，用户日志如下所示。

```
2017-01-09 16:02:49|松江区
2017-01-09 16:03:49|浦东区
2017-01-09 16:03:49|上城区
2017-01-09 16:03:49|下城区
```

日志按照“|”切分之后的的字段分别为`date`、`region`。如果`region`为“松江区”或者“浦东区”，则添加一个新的字段`province`（String 类型），其值为“上海市”；如果`region`为“上城区”或者“下城区”，则添加一个新的字段`province`（String 类型），其值为“杭州市”。



内置函数（转自 Aviator 官方文档）

Aviator 表达式是一个开源产品（点此查看官方文档）。常见内置函数的说明如下表所示。

函数	描述
sysdate()	返回当前日期对象 java.util.Date
rand()	返回一个介于0-1的随机数，Double 类型
now()	返回 System.currentTimeMillis
long(v)	将值的类型转为 Long
double(v)	将值的类型转为 Double
date_to_string(date,format)	将 Date 对象转化为特定格式的字符串
string_to_date(source,format)	将特定格式的字符串转化为 Date 对象
string.contains(s1,s2)	判断 s1 是否包含 s2，返回 Boolean
string.length(s)	求字符串长度，返回 Long
string.startsWith(s1,s2)	s1 是否以 s2 开始，返回 Boolean
string.endsWith(s1,s2)	s1 是否以 s2 结尾，返回 Boolean
string.substring(s,begin[,end])	截取字符串 s，从 begin 到 end，end 如果忽略的话，将从 begin 到结尾，与 java.util.String.substring 一样
string.indexOf(s1,s2)	Java 中的 s1.indexOf(s2)，求 s2 在 s1 中的起始索引位置，如果不存在为-1
string.split(target,regex,[limit])	Java 里的 String.split 方法一致

函数	描述
string.replace_first(s,regex, replacement)	Java 里的 String.replaceFirst 方法
string.replace_all(s,regex, replacement)	Java 里的 String.replaceAll 方法
math.abs(d)	求 d 的绝对值
math.sqrt(d)	求 d 的平方根
math.pow(d1,d2)	求 d1 的 d2 次方
math.log(d)	求 d 的自然对数
math.log10(d)	求 d 以 10 为底的对数
math.sin(d)	正弦函数
math.cos(d)	余弦函数
math.tan(d)	正切函数

7 最佳实践

7.1 日志清洗最佳实践

本文介绍了关于清洗日志的最佳实践。

为什么要清洗日志

ARMS 为用户提供低学习成本的实时监控解决方案。不同用户的日志格式是不同的，日志的字段信息也是不同的。为了让 ARMS 能使用用户的日志，需要先对用户日志进行清洗工作。

以下是一行示例日志：

```
2016-08-24-13:32:33|itemID=abc|amount=100
```

ARMS 需要知道，在这行日志中，时间为 2016-08-24-13:32:33，商品 ID 为 abc，以及交易额为 100。只有清洗出这些属性以后，才能在后续聚合计算编排中，针对这些属性进行计算。

ARMS 将数据清洗过程抽象为通过切分日志数据形成时间、字符串、数字等字段的过程。

两种日志清洗方式

ARMS 提供两种日志清洗方式：

- 智能切分：快速、便捷、智能

  智能生成数据清洗切分方案。对用户提供的部分日志样例进行智能分析，并采用最优方案切分日志。用户可在拖拽式的可视化界面微调智能切分方案，大幅节省了手动配置切分方式的时间。
- 自定义切分：手动、全面、可控

  用户可在拖拽式的可视化界面配置数据清洗切分逻辑。全图形化的配置流程，让用户不需要编写代码即可完成大部分的监控配置任务。

关于日志格式的建议

- 尽量保证格式一致

  请尽量保证同一日志源的日志格式一致。若同一日志源有多种格式的日志，则需要使用 filter 功能过滤出需要的日志内容。

- 使用可以识别的时间格式

  ARMS 支持多种常用时间格式（需要精确到年月日时分秒，毫秒可选）。

  自定义切分支持所有能转化为常用 SimpleDateFormat 时间正则表达式（例如 yyyy-MM-dd HH:mm:ss）的时间格式。

  智能切分能识别的时间格式有：

  - 能转化为常用 SimpleDateFormat 时间正则表达式（例如 2016-8-21 23:12:53）的常用时间格式。
  - Nginx 时间格式，例如 28/Nov/2014:11:56:09 +0800。
  - 其他 Java SimpleDateFormat 常用格式。详情请参见智能切分可识别的时间格式。

- 使用清晰无歧义的分隔符 （常用分隔符有空格、“|”、“;”、“,”等）

  建议使用同类分隔符，这样更容易被切分器切分。

- 字串子串等建议使用 JSON 格式

  如果日志串中含有 JSON 字串，则使用 JSON 切分器清洗日志更容易。

  日志中的 JSON 串需为有引号的标准 JSON 格式，且不能含有换行符。

7.2 映射表使用最佳实践

本文介绍了关于使用映射表的最佳实践。

映射表用于存储静态数据，其主要功能有：

- 清洗日志后，通过对清洗出的字段进行静态 Join 操作来获取需要的目标字段。
- 查询数据集时，可以结合映射表进行组合查询。

静态 Join 使用案例

假设用户日志格式如下：

```
2017-01-09 16:02:49|42.**.**.**|Kevin.Yang
```

下图为原始切分逻辑，切分后的字段分别为 date、ip 和 username。

图 7-1: 原始切分逻辑

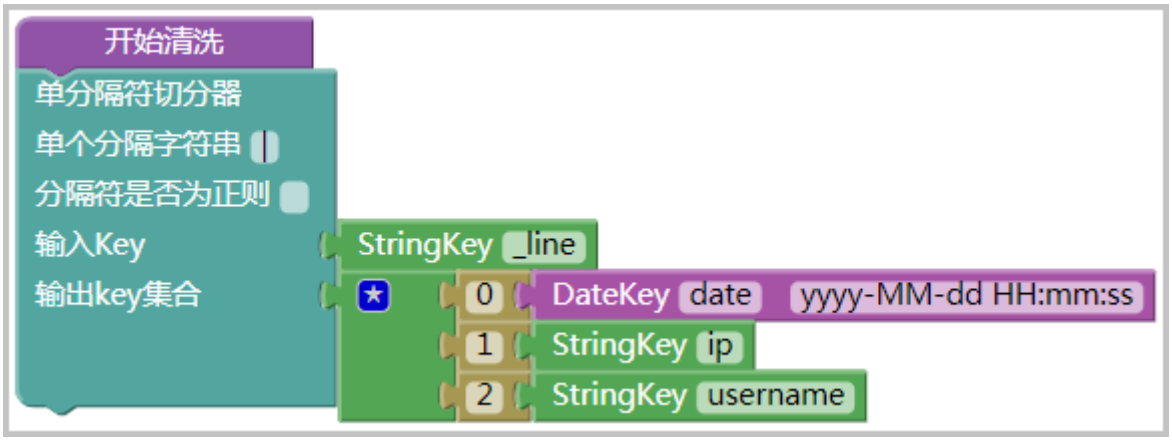


图 7-2: 原始切分结果

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2017-01-09 16:02:49|42.**.**.**|...
_sysTime	date	1528439406469
date	date	1483948969000
ip	string	42.*
username	string	Kevin.Yang

由于切分模型中没有相关的国家字段，如需统计每分钟每个国家的用户访问量，可以通过映射表进行 Join 操作。映射表中保存了 IP 与国家、省份、城市的映射关系。下图为结合映射表之后的切分逻辑。

图 7-3: 结合映射表的切分逻辑

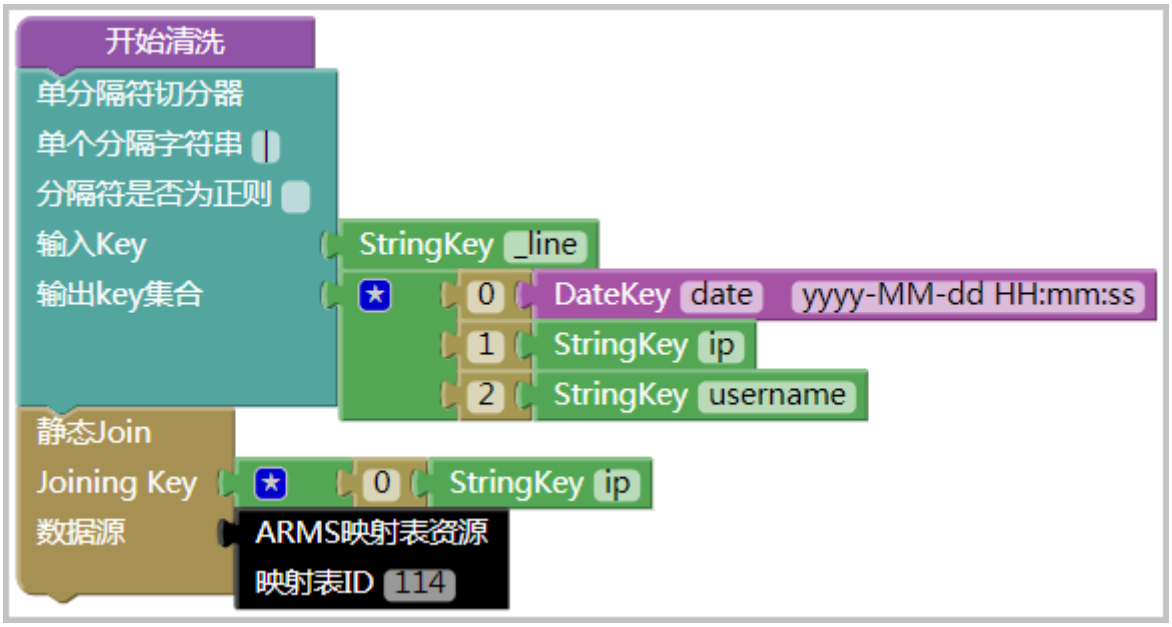


图 7-4: 结合映射表的切分结果

字段名称	类型	样例数据
_hostIp	string	127.0.0.1
_line	string	2017-01-09 16:02:49|42.**.**.**|...
_sysTime	date	1528439526435
city	string	杭州
country	string	中国
date	date	1483948969000
ip	string	42.
province	string	浙江
username	string	Kevin.Yang

可见，国家、省份、城市这三个字段已根据原始的 IP 字段获得，那么图片图 7-3: 结合映射表的切分逻辑中的“映射表 ID”从何而来呢?

以下是为本例新建映射表的步骤。

1. 在控制台左侧导航栏中选择自定义监控 > 映射表管理，进入映射表管理页面。在页面右上角单击新建映射表。

2. 在新建映射表对话框中填写相关信息。

图 7-5: 新建映射表

a. 填写映射表的 Schema 信息：Schema 信息代表源字段信息（名称、类型）与目标字段信息（名称、类型）的映射关系。

注意:

仅支持 String、Long、Double 类型。

b. 按照样例数据的格式，在文本内容中输入您的数据。由于本例的需求是将 IP 转化为地址，上图中的内容是 IP 与地址的映射关系。

说明:

文本内容必须严格按照 Schema 设定的形式，否则无法保存。

3. 在新建映射表对话框中单击确定，系统会为此映射表生成一个唯一的资源 ID 。将此资源 ID 填入“ARMS 映射表资源”积木块中即可。

在以上示例中，映射表中只为 IP 42.**.**.** 配置了映射关系。当监控任务开始运行后，如果实际日志中包含其他 IP，利用上述映射表进行静态 Join 会不会有问题?

不用担心，您无需停止任务，只需要更新映射表里面的文本内容即可。

图 7-6: 编辑映射表

说明:

编辑映射表时，Schema 不能修改。

数据集组合查询使用示例

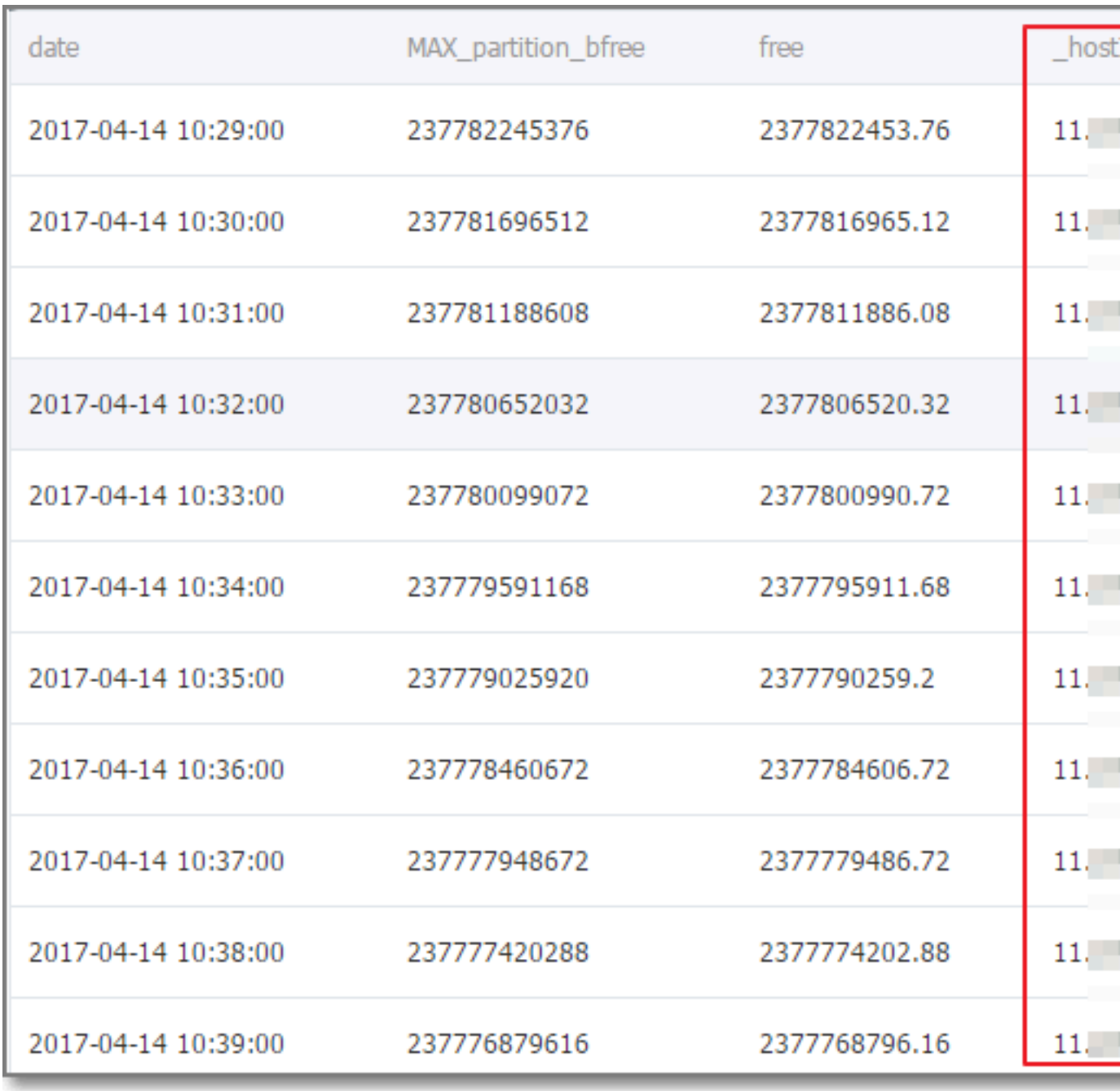
以下是一个利用映射表进行数据集组合查询的简单案例。

在控制台左侧导航栏中选择自定义监控 > 数据集管理，单击右侧的查询数据。

图 7-7: 数据查询

以上数据集存在维度_hostIp，单击查询，结果如下所示。

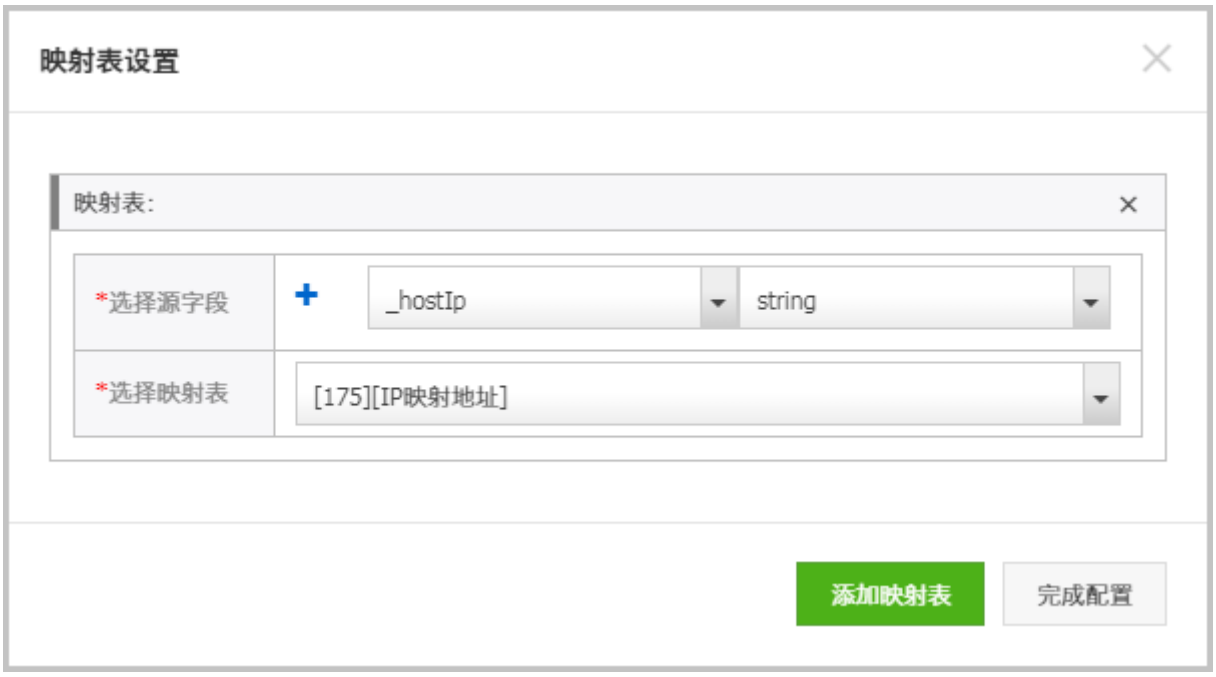
图 7-8: 查询结果



date	MAX_partition_bfree	free	_host
2017-04-14 10:29:00	237782245376	2377822453.76	11.
2017-04-14 10:30:00	237781696512	2377816965.12	11.
2017-04-14 10:31:00	237781188608	2377811886.08	11.
2017-04-14 10:32:00	237780652032	2377806520.32	11.
2017-04-14 10:33:00	237780099072	2377800990.72	11.
2017-04-14 10:34:00	237779591168	2377795911.68	11.
2017-04-14 10:35:00	237779025920	2377790259.2	11.
2017-04-14 10:36:00	237778460672	2377784606.72	11.
2017-04-14 10:37:00	237777948672	2377779486.72	11.
2017-04-14 10:38:00	237777420288	2377774202.88	11.
2017-04-14 10:39:00	237776879616	2377768796.16	11.

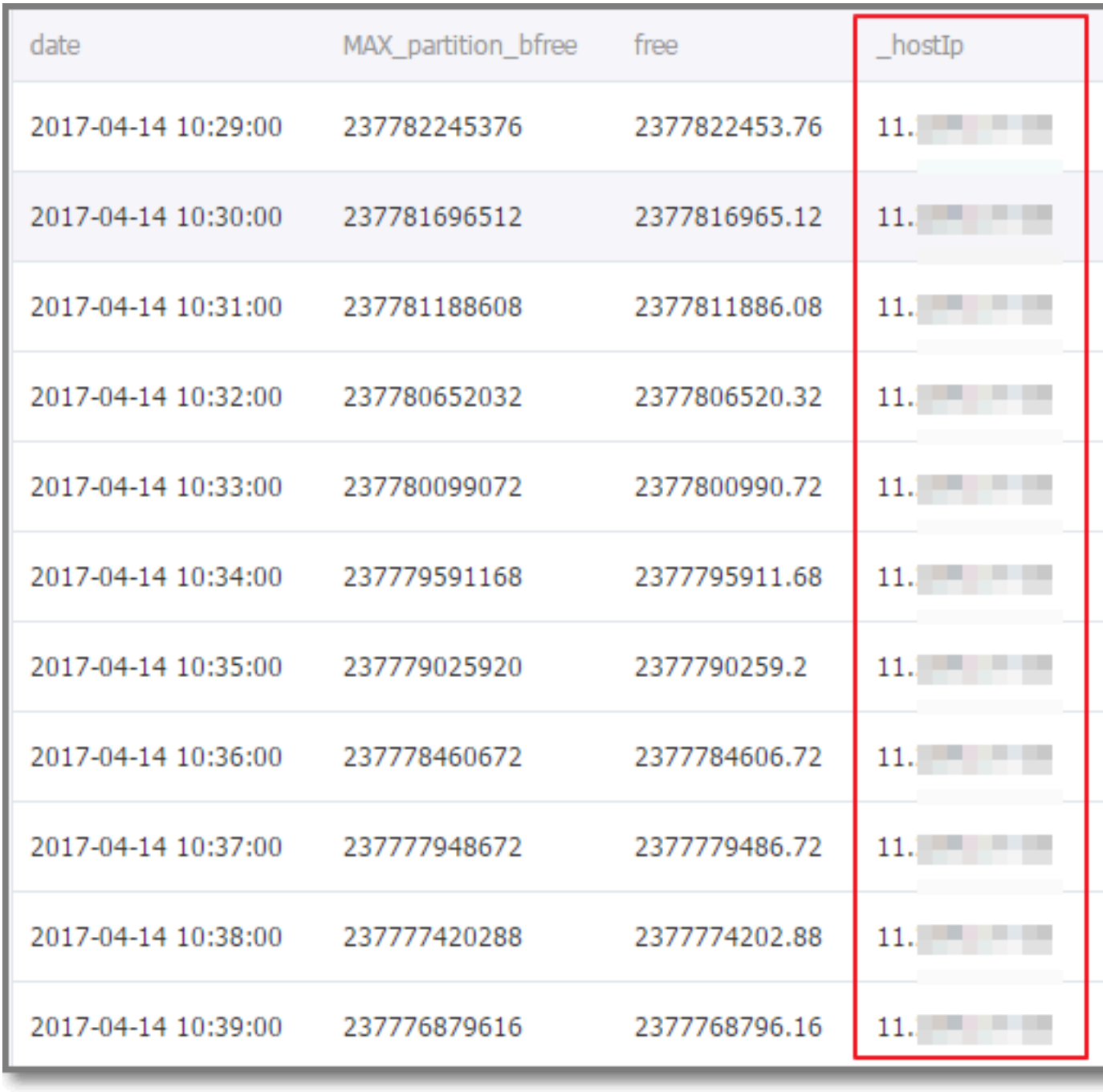
现在需要查询各个省份、地址的详细数据，该怎么做？只需要对_hostIp 进行 Join 查询即可，因为我们已经在自己的映射表中配置 IP 与地区的映射关系。

图 7-9: 映射表配置



查询后的详细数据如下所示。

图 7-10: 详细数据



date	MAX_partition_bfree	free	_hostIp
2017-04-14 10:29:00	237782245376	2377822453.76	11.
2017-04-14 10:30:00	237781696512	2377816965.12	11.
2017-04-14 10:31:00	237781188608	2377811886.08	11.
2017-04-14 10:32:00	237780652032	2377806520.32	11.
2017-04-14 10:33:00	237780099072	2377800990.72	11.
2017-04-14 10:34:00	237779591168	2377795911.68	11.
2017-04-14 10:35:00	237779025920	2377790259.2	11.
2017-04-14 10:36:00	237778460672	2377784606.72	11.
2017-04-14 10:37:00	237777948672	2377779486.72	11.
2017-04-14 10:38:00	237777420288	2377774202.88	11.
2017-04-14 10:39:00	237776879616	2377768796.16	11.

7.3 监控任务故障诊断最佳实践

在日常运维中，监控任务可能会由于各种各样的原因出现异常。本文介绍了出现问题的各种场景、原因和处理方法，旨在帮助用户快速解决问题。

ARMS 的任务处理环节介绍

ARMS 的监控任务主要由三个环节组成：

1. 数据拉取：ARMS 计算集群从 Loghub、ECS Log 等数据源拉取数据。
2. 数据清洗：ARMS 在抓取数据后，成功清洗（切分）了多少条数据。
3. 数据聚合：ARMS 在清洗数据后，在内存中成功进行了多少次数据聚合以及数据持久化操作。

在以上每个步骤中，创建和启动任务以后，ARMS 都有监控显示运行详情。在监控任务管理页面单击目标任务，即可展示任务运行详情，如下图所示。

此页面上各图表展示的是对应系统时间内，ARMS 处理的数据行数。

当任务出现异常时，可以从该页面进行异常诊断。以下详细介绍各种情况。

任务异常诊断的一般过程

任务运行详情页所有图表显示“暂无数据”

此时可以检查以下几点：

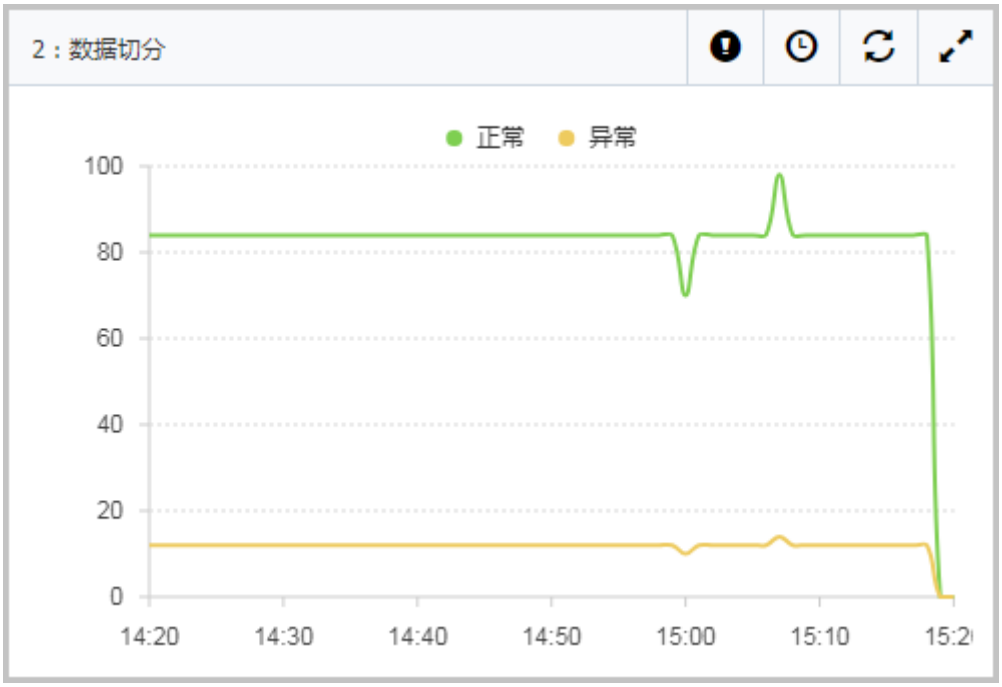
1. 如果监控任务是刚刚启动的，那么请稍等 1-2 分钟，等待数据拉取完毕。
2. 可能这段时间确实没有数据。单击面板上的时钟图标，拉长时间范围，查看一下是否有历史数据。

图表有黄色异常线条

正常情况下，三个图表都只会有一条绿色线条。如果出现异常状况，相应图表里会多出一条黄色线条。任务运行的三个环节都有可能出现异常。单击感叹号图标可以查看异常的抽样详情，用于判断异常原因。以下以数据清洗和数据聚合为例，展示如何诊断异常。

示例 1：数据清洗异常诊断

症状



出现黄色曲线之后，请先单击面板上的感叹号图标，查看异常类型。在下图的示例中，code 为 market2_mtair_et2 的字段出现了类型转换异常。

分析

在此例中，产生问题的原因通常是，使用智能切分时，提供的字段值被判定为 long 型，而实际产生数据的时候，出现了不能转换为 long 的 String 类型。

解决

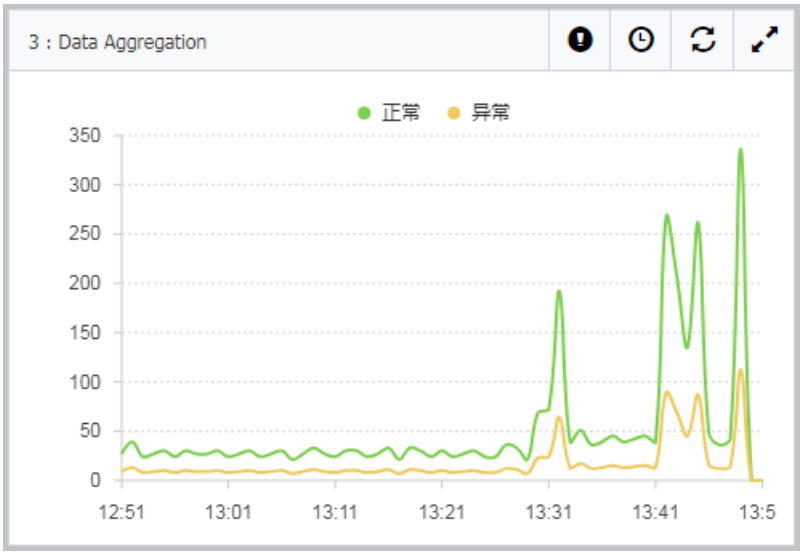
暂停监控任务后，编辑监控任务，在第 2 步日志清洗页面，选择自定义切分，将 LongKey 的 code 换成 StringKey ，保存后恢复监控任务。

重新运行之后，可以发现后续的黄色线条数值变为 0，切分异常得到解决。

示例 2：数据聚合异常诊断

症状

紧接上一个例子，通过调整切分模型，切分异常消失了，但是出现了数据聚合异常。



单击感叹号图标，可以看到一个 String 无法转换为 Number 类型的异常，如下图：

分析

此时需要回忆一下，在建立数据集时是否对刚才 LongKey 的 code 做过一些算术运算。经过排查，发现在某个数据集中，我们对 code 这个字段进行了 SUM 。当时进行 SUM 运算只是为了试验 SUM 的效果。

解决

将 SUM 去除，异常消失。

7.4 进一步了解 ARMS：社区文章参考

以下社区文章能帮助您进一步了解 ARMS。

- 论云时代最经济的 APM 工具的姿势
- 阿里云应用监控（ARMS）过程全解析：我们在乎用户每一秒的体验
- 通过页面埋点做监控却不影响性能？解密 ARMS 前端监控数据上报技术内幕
- ARMS “前端+应用”监控商业化首发，一起聊聊关于监控哪些事儿
- 数十万应用结点全息监控，ARMS 新上线的应用监控神器到底有多牛？
- 用户洞察的秘密武器：ARMS 前端监控功能正式上线
- 波司登全国超千家门店，销售状况如何实时监控？

7.5 全息排查最佳实践

全息排查用于通过业务主键快速定位问题链路，需要和应用监控功能搭配使用。本文介绍了全息排查最佳实践。

前提条件

1. 在 ARMS 控制台上已创建应用监控，并已在 Java 程序中挂载和启动应用监控的 Agent。详情参见#unique_63中关于安装 Java 探针的步骤。
2. 程序中已引入 *arms-sdk-1.7.1.jar*。

```
<dependency>
<groupId>com.alibaba.arms.apm</groupId>
<artifactId>arms-sdk</artifactId>
<version>1.7.1</version>
</dependency>
```

说明:

如果无法获取 Pom，请直接下载 arms-sdk-1.7.1.jar。

获取 TraceId 与 RpcId

满足上述前提条件后，即可通过以下代码获取上下文的 TraceId 与 RpcId。

```
Span span = Tracer.builder().getSpan();
String traceId = span.getTraceId();
```

```
String rpcId = span.getRpcId();
```

打印日志

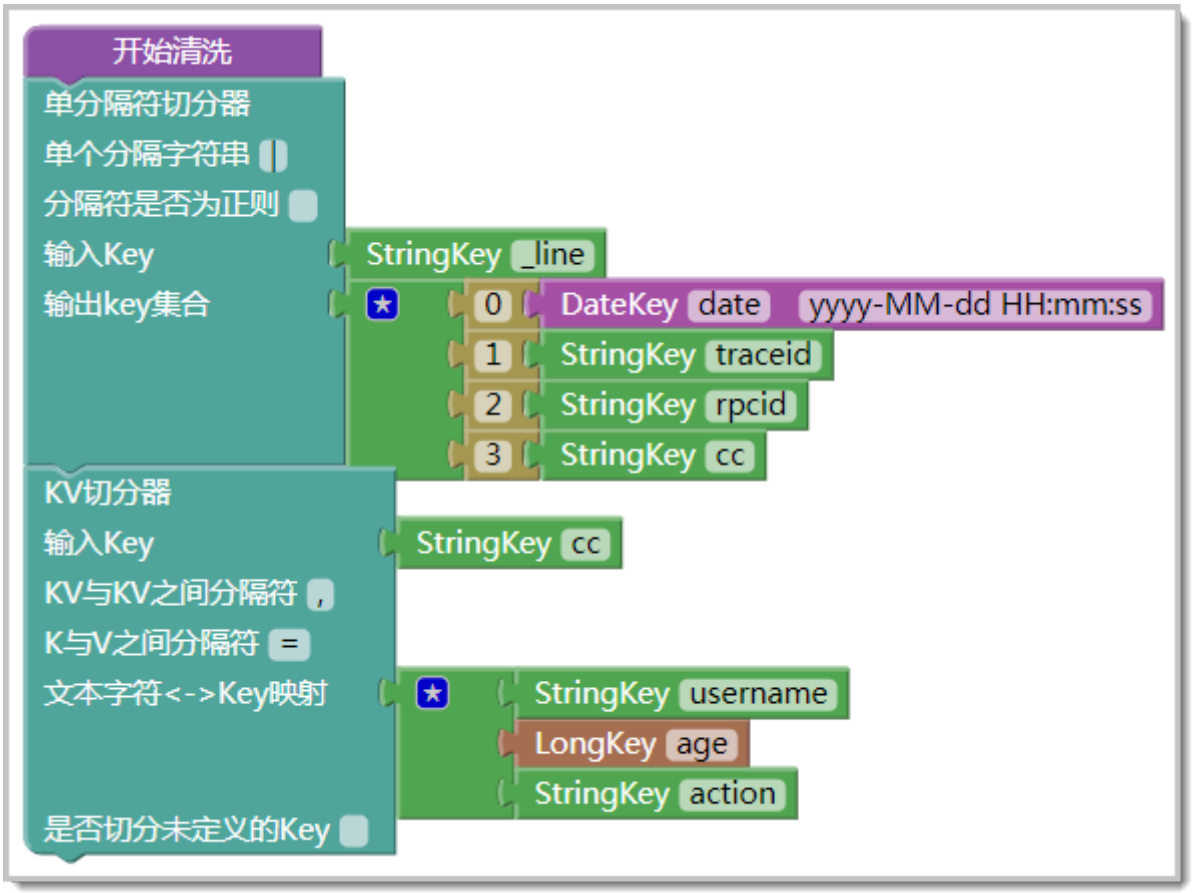
获取 TraceId 与 RpcId 后，您可以根据业务需求打印和输出业务日志。以下是包含 TraceId 与 RpcId 的样例业务日志。该日志输出到文件 */home/admin/logs/example/example.log* 中，但您也可以按需将其输出到 SLS、MQ 等其他通道中。

```
2018-07-12 11:37:40|1e057c401531366659965100 5d1201|0|username=xiao,age
=22,action=login
2018-07-12 11:37:40|1e057c401531366659965100 5d1201|0|username=xiao,age
=22,action=search
2018-07-12 11:37:40|1e057c401531366659965100 5d1201|0|username=xiao,age
=22,action=cart
```

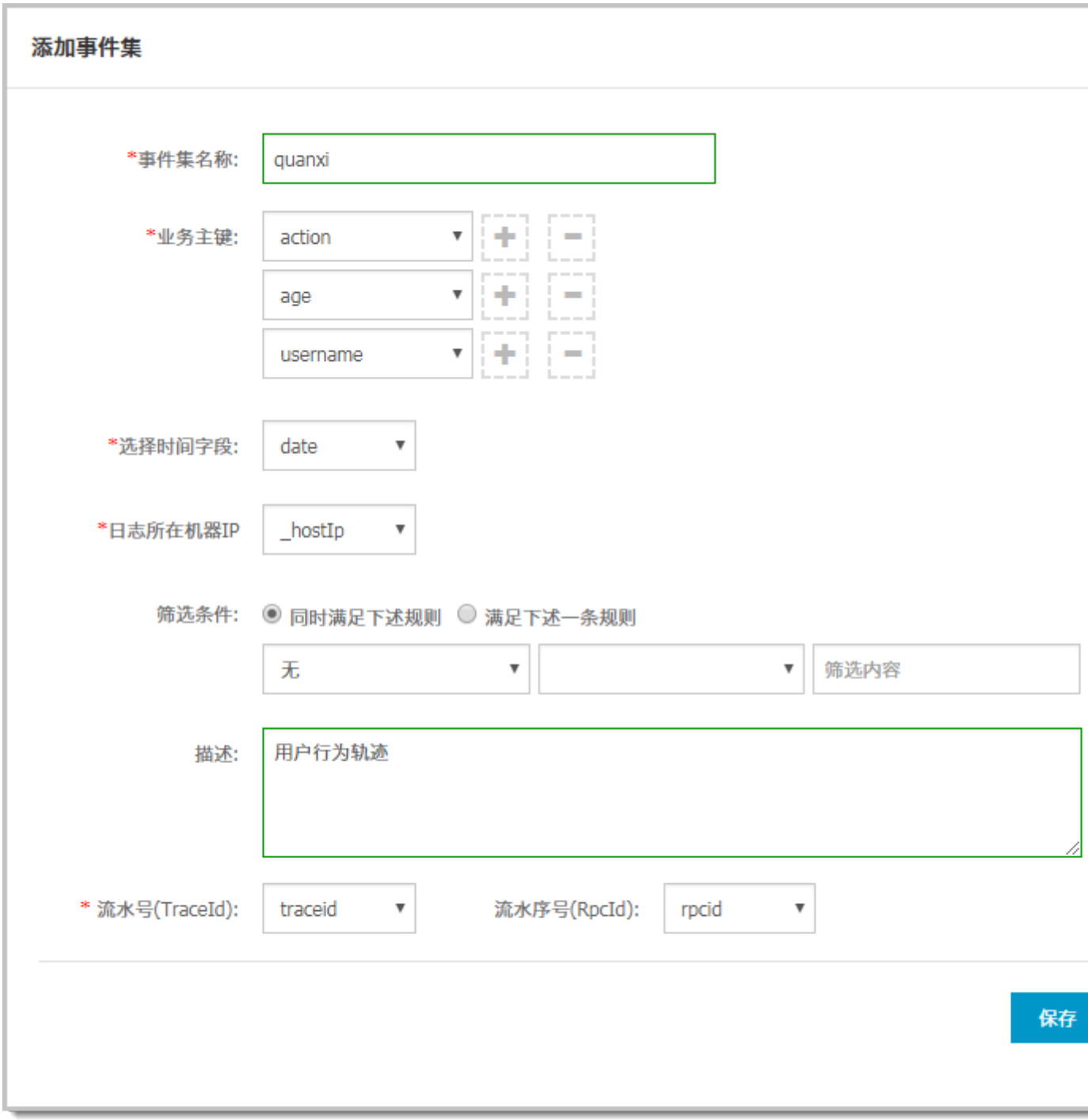
以上每条业务日志均表示用户的一条行为轨迹。

配置全息排查事件集

按照自定义监控章节的说明创建一个自定义监控任务，使用上方的样例业务日志作为数据源，并按照下图方案自定义切分日志。



随后进入创建自定义监控任务的“#unique_39”步骤，并按照下图配置事件集。



- 业务主键：表示搜索业务事件所使用的字段。在本文的示例中，业务主键是行为（action）和用户名称（username）。
- 选择时间字段：必须选择业务时间，不能选择系统时间。
- 流水号：设置 TraceId。
- 流水序号：设置 RpcId。

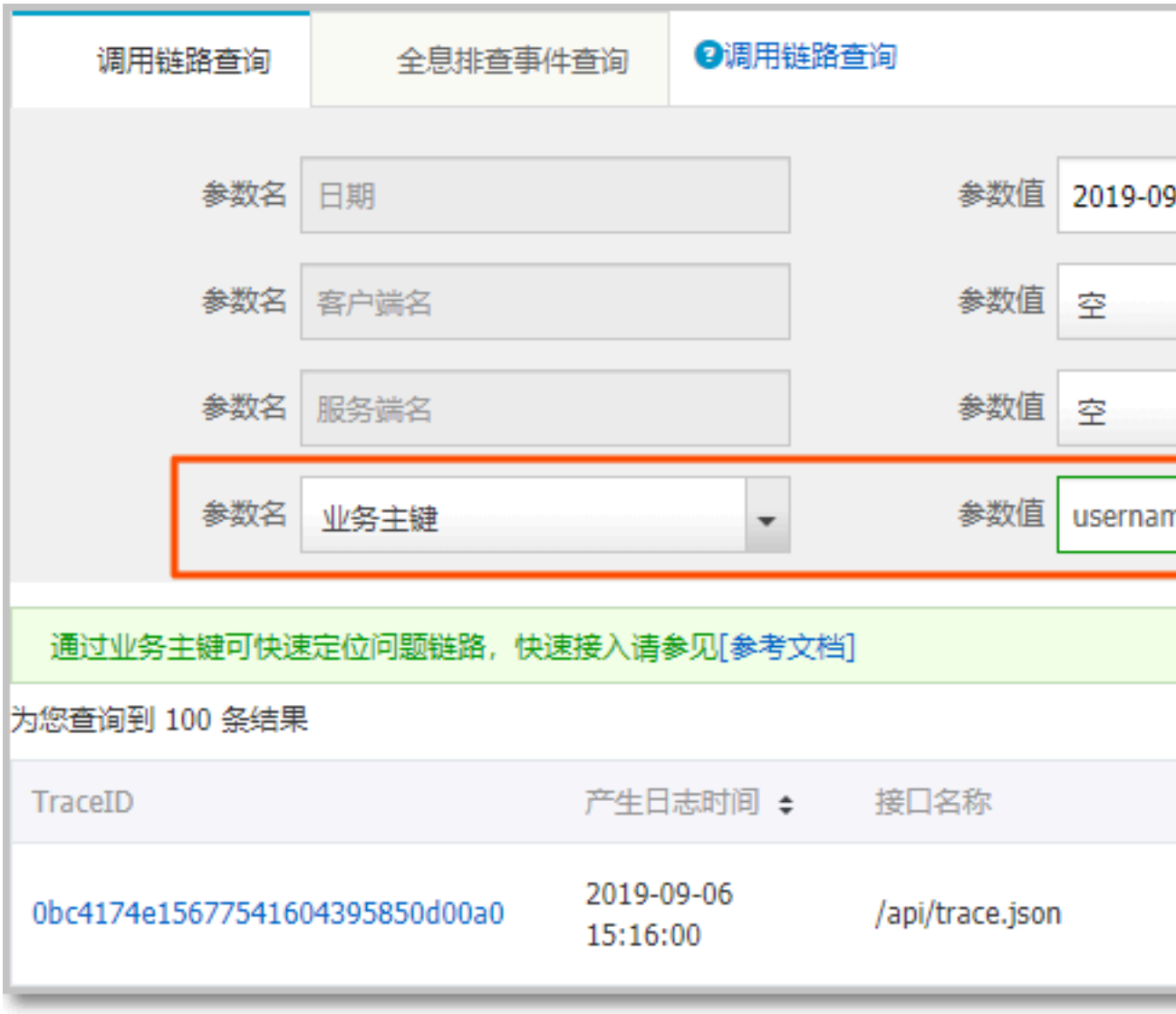
配置好事件集后，请启动自定义监控任务。

全息排查功能的使用案例

本案例的业务日志表示用户的行为轨迹，对应的应用为购物网站。假设用户 kevin.yang 投诉称在 2018-07-12 14:20 以后下单失败，那么您可以通过以下两种方法来定位问题原因。

- 方法 1：调用链路查询。

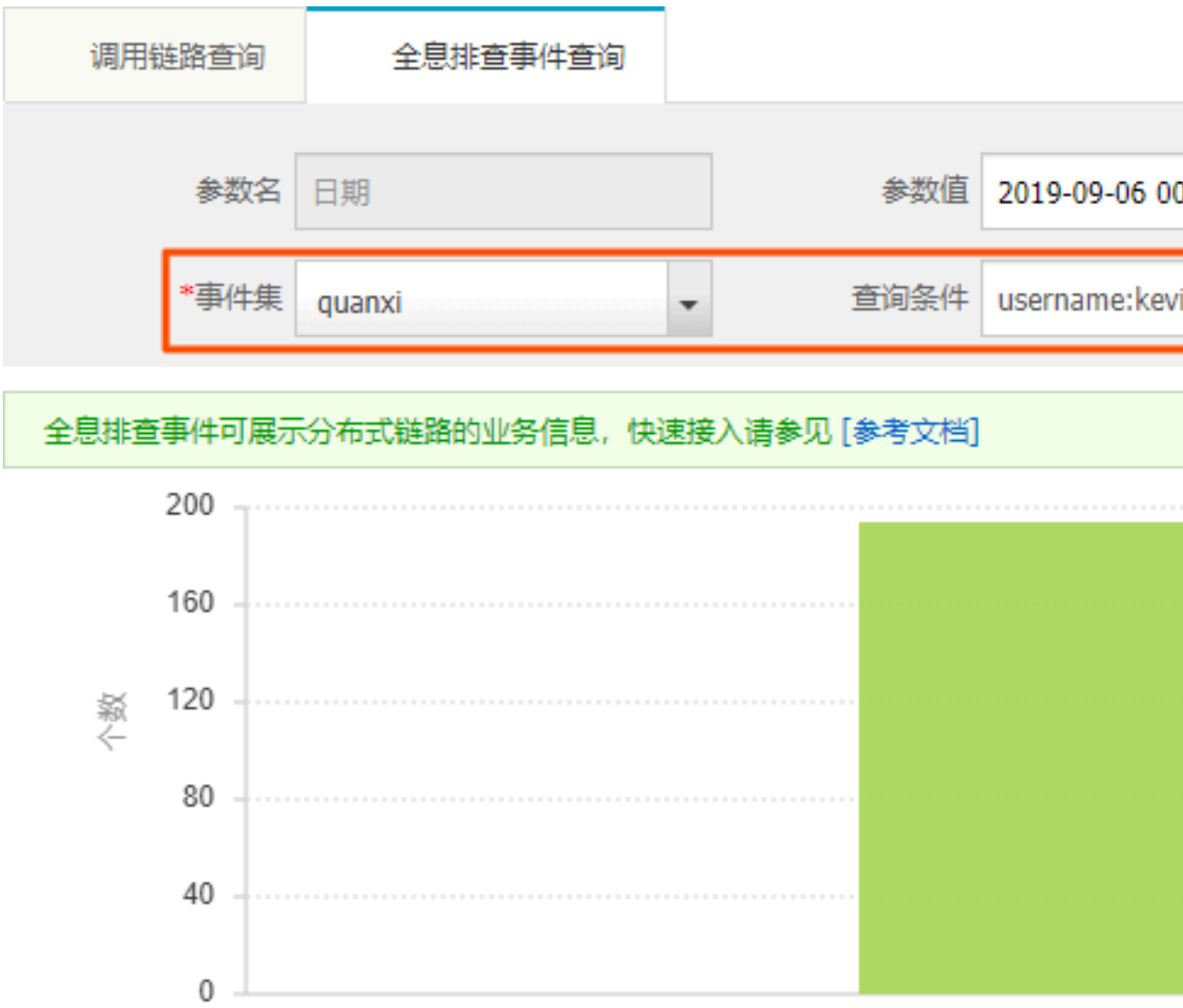
  1. 在左侧导航栏中单击应用监控 > 调用链路查询，进入实例列表页面的调用链路查询页签。
  2. 在标签页上的日期参数值中输入日期范围，在最下方的参数名下拉列表中选择业务主键，并在右侧的参数值中输入业务主键的值，例如本例中的 `username:kevin.yang`，然后单击查询。指定时间区间内的所有调用链路显示在搜索结果中。



  3. 在搜索结果中，单击异常调用链路的 TraceId，然后单击业务轨迹页签，显示该 TraceId 下的所有业务事件，并根据业务事件定位问题原因。

- 方法 2：全息排查事件查询。

  1. 在左侧导航栏中单击应用监控 > 调用链路查询，进入实例列表页面的全息排查事件查询页签。
  2. 在标签页上的日期参数值中输入日期范围，在全息排查事件集下拉框中选择前面配置的事件集，然后单击查询。指定时间区间内的所有调用链路显示在搜索结果中。



时间

2019年9月6日 上午 10:17:34

traceid 流水号(traceId): 1e057c4015313666599651005d120
_hostIp: [redacted]
action: login
rpcid 流水序号(rpcId): 0
age: 22
_line: 2019-09-06 10:17:34|1e057c4015313666599651005d
username: kevin.yang

2019年9月6日 上午 10:17:44

traceid 流水号(traceId): 1e057c4015313666599651005d120
_hostIp: [redacted]
action: login
rpcid 流水序号(rpcId): 0
age: 22
_line: 2019-09-06 10:17:44|1e057c4015313666599651005d
username: kevin.yang

3. 在搜索结果中，单击调用链查询，然后单击业务轨迹页签，显示该 TraceId 下的所有业务事件，并根据业务事件定位问题原因。

更多信息

- #unique_39

8 基本概念

8.1 标准方案模板介绍

从 2017 年初开始，ARMS 开始上线标准行业模板功能，日后还将陆续上线一系列常用的标准行业模板任务，供用户开箱即用。本文介绍了这些已发布或将要发布的行业模板。

关于标准方案模板任务的定义，以及它和自定义监控任务的区别，请参见#unique_66。关于标准方案模板的使用，请参考#unique_5和#unique_67。

现有标准方案模板

- Nginx 监控模板：通过收集和分析 Nginx 的默认日志，来监控服务端被访问情况。
- 零售行业监控模板：通过收集各区域门店的销售状态，来实时反映零售行业的销售现状。

标准方案模板详细介绍

Nginx 监控模板标准版

Nginx 监控建议模板通过收集和分析 Nginx 的默认 Nginx access.log 日志来监控服务端被访问情况。

该模板统计包括以下部分：

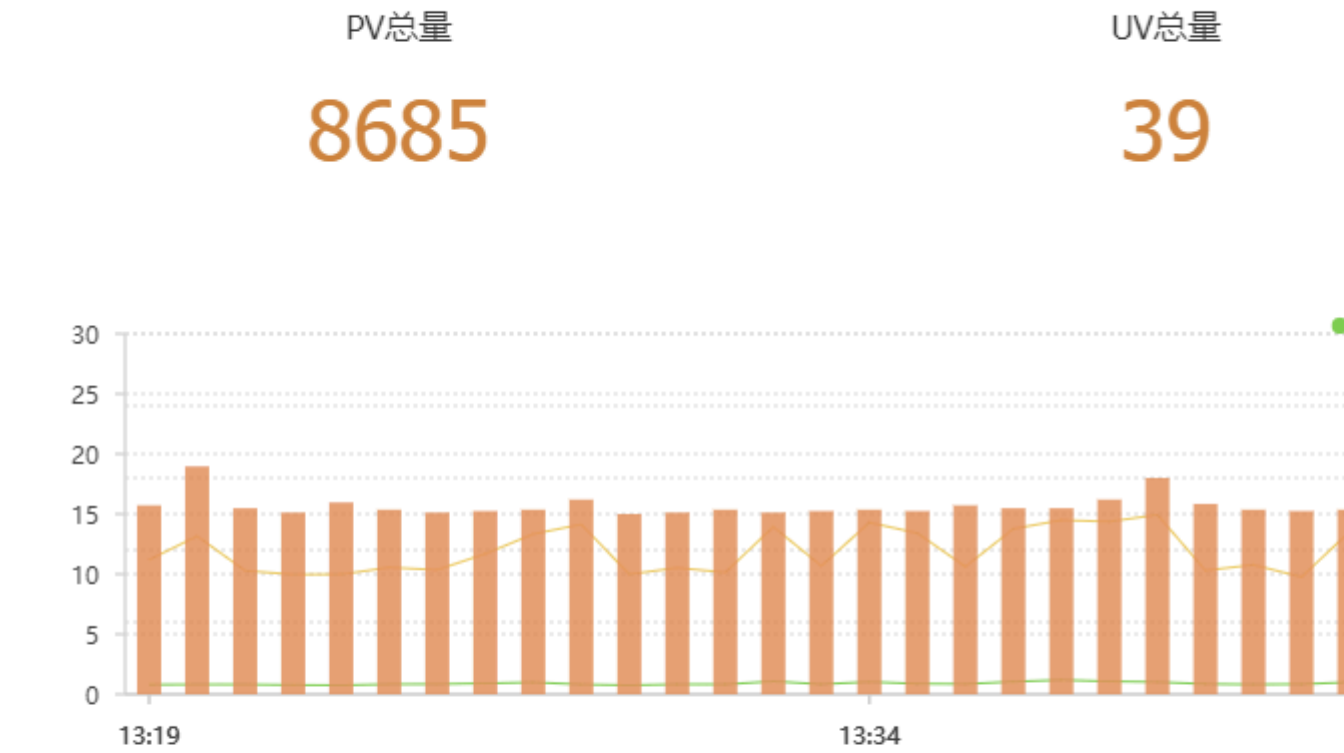
- Nginx 的默认日志字段，包括 $remote_addr、“$request”、“$status”、“$body_bytes_sent”、“$http_referer”、“$http_user_agent”等。
- Nginx 的扩展日志字段（“$upstream_response_time”、“$request_time”），以及代表 Session ID 的对应的 Cookie ID（在本例中，以“$jsessionid”为例）。

以下是本例中 access.log 的 log_format 配置：

```
log_format proxyformat "$remote_addr $request_time_usec $http_x_rea
dtime [$time_local] \"$request_method http://$host$request_uri\" $
status $body_bytes_sent \"$http_referer\" \"$upstream_addr\" \"$
http_user_agent\" \"$upstream_response_time\" \"$request_time\" \"$
jsessionid\"";
```

以下是 Nginx 监控模板的大盘内容首页。大盘版本一直在迭代，最新版本以线上为准。

Nginx 总体访问标准版监控示例：



url
http://arms.console.aliyun.com/shareapi/query.json
http://arms.console.aliyun.com/api/query.json
http://arms.console.aliyun.com/api/index.json

零售行业监控模板

本零售行业监控模板基于某行业领先的服装零售商的真实业务场景所建。该模板提供以下内容的实时监控：

- **基于任意时间维度的全网销售总额、销售商品总件数，和交易笔数的实时监控统计。**
- **基于各销售片区以及各旗下公司，主管的销售额实时监控统计。**
- **基于各片区的商品热度销售排名。**
- **基于历史数据的商品销售历史查询和趋势统计。**

本监控任务需要业务方将每一笔业务交易接入给 ARMS。业务交易数据可以使用但不限于日志、消息或 API 推送等方式。在每一笔交易数据中，需要保证从数据源中准确清洗出以下数据字段：

- 片区名称：公司业务根据业务量下划的片区，例如西南片区、东北片区。
- 公司简称：根据片区下划的公司，例如东北片区的吉林分公司。
- 客户名称：一般指代销的经销商，例如东北片区的吉林分公司所管理的经销商吉林百货大楼。
- 客户类型：根据业务类型，一般可分为直营和加盟两类。
- 货号：商品 ID 号。
- 尺码：商品的尺码。
- 货色：商品的颜色。
- 件数：本次交易所发生的商品件数。
- 零售金额：本次交易所发生的实际零售金额总额。

8.2 通用维度与下钻维度

本文介绍了通用维度与下钻维度的区别。

通用维度：适用于所有场景，但其中的维度是没有加速索引的（除非开启 ID 类维度，详细解释参见下文）。

下钻维度：针对特定场景，当维度之间存在层级关系，如省 > 市 > 区，那么下钻类维度会针对每层的查询加速。

通用维度

通用维度场景解析

以电商日志为例：`2017-01-01 12:00:00`｜类目：男装｜省份：浙江｜市：杭州｜区：西湖区｜性别：男｜身高：L|数量：5｜总价：100｜

切分后的字段为：

- 时间
- 类目
- 省份
- 市
- 区
- 性别
- 身高
- 数量

· 总价

我们需要根据商品的类目、性别、省份属性来分析该数据，则维度依次为类目、性别、省份，指标为单价和数量。预聚合之后数据为：

总价	数量	时间	性别	类目	省份
100	1	2017-01-01 12:00:00	男	男装	浙江
200	2	2017-01-01 12:00:00	女	食品	江苏
300	3	2017-01-01 12:00:00	女	男装	北京

当我们需要查看类目为男装的数据时，需要读取不同类目，不同性别和不同省份对应的所有数据，然后过滤出男装的数据。这里取出数据记录数是大于结果记录数的。

限制和优化方法

假设有 200 万的类目，我们还是查看类目为男装的数据，那么需要读取约 N 个 200 万的数据，然后过滤出男装数据。这里取出数据记录数是大于结果记录数，且太多的读取记录直接影响到获取数据的速度。

此时可以通过创建类目的索引来解决。

在通用维度类型的数据集中，ARMS 提供一种辅助维度，叫 ID 类维度。ID 类维度就相当于索引维度，在查询时要给出明确的维度值，加速数据查询；而维度就是普通非索引维度，如本例最开始的性别、类目、省份维度。

维度和 ID 类维度区别

通用维度包含维度和 ID 类维度。在数据集的查询过程中，ID 类维度不能为空，而维度可为空。目前 ARMS 中最多包含一个 ID 类维度和七个维度。

· 维度
  - 维度可独立使用或组合使用。例如某数据集有维度 A、B、C，您可以仅选择维度 A、B 或 C，也可以使用 BC 组合，或者 ABC 组合等查询。
· ID 类维度
  - ID 类维度类似于对该维度创建索引，在查询时，通过指定 ID 类维度可以快速查询到数据结果。
  - 对于数据中不可枚举或者维度值个数较多的情况，建议使用 ID 类维度。

下钻维度

下钻维度场景解析

以系统监控领域的场景为例，系统日志中包含机房、分组和 IP 三个维度。用户需要从机房运行情况，下钻到某机房的分组，然后是分组的某一台机器进行数据查询。如果使用通用维度处理该问题，则存在查询数据量较大导致查询延迟的问题。下钻维度可用来解决以上这种固定的逐级查询场景。

下钻维度对机房、分组、IP 创建多级索引，索引分别为机房（索引 1），机房-分组（索引 2），机房-分组-IP（索引 3）。查看机房数据时使用索引 1，查看某机房的分组数据时使用索引 2，从该分组下钻到 IP 时使用索引 3。

下钻维度的使用场景还可以包括按照省、区维度进行业务统计，按照学校、年级、班级查询学生分布，或按照厂商、品牌、类目统计售卖情况等。

下钻维度使用限制

- ARMS 中最多可以设置三个下钻维度。
- 下钻维度之间有层级关系。例如，要查看第二个维度，必须先选择第一个维度的属性。维度类似于一个树状结构。维度的定义需要规划，例如，第一维度可以是省，第二维度可以是市，第三个维度可以是区，指标数据则是市民消费情况。
- 除非有特殊场景需求，两个完全不相关的维度最好不要一同定义，比如“地域”和“物品类型”。
- ARMS 提供下钻功能，可以从汇总数据深入到细节数据进行观察或新增维度。钻取的目的是改变维度层次、变换分析粒度。