阿里云
日志服务

SDK 参考

文档版本：20180929

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。
斜体	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 基本介绍

1.1 概述

为方便开发人员更高效地使用日志服务，日志服务提供了多个语言版本（Java、.NET、Python、PHP、C）的 SDK（Software Development Kit），您可以根据自己需求选择合适版本使用。

日志服务 SDK 基于日志服务 API 实现，且提供和日志服务 API 同样的能力。如果您需要了解日志服务 API 的更多细节，参见概览。

类似于使用日志服务 API，您首先需要拥有一个处于 Active 状态的阿里云访问秘钥（AccessKeyId/AccessKeySecurity），详情参见 访问秘钥。

为使用日志服务 SDK，您需要了解日志服务在各个阿里云区域（Region）的服务入口。具体如何在 SDK 中指定这个根服务入口，请参考 *SDK* 配置。

尽管不同语言的日志服务 SDK 具体实现细节会有所不同，但是它们都是日志服务 API 在不同语言上的封装，实现的功能也基本一致，具体包括如下几个方面：

- 实现对日志服务 API 接口的 统一封装，让您不需要关心具体的 API 请求构建和响应解析。而且各个不同语言的接口也非常接近，方面您在不同语言间切换。
- 实现日志服务 API 的 数字签名 逻辑，让您不需要关心 API 的签名逻辑细节，大大降低使用日志服务 API 的难度。
- 实现日志服务日志的 *ProtoBuffer* 格式 封装，让您在写入日志时不需要关心 ProtoBuffer 格式的具体细节。
- 实现日志服务 API 中定义的压缩方法，让您不用关心压缩实现的细节。部分语言的 SDK 可以让您指定是否启用压缩模式写入日志（默认为使用压缩方式）。
- 提供统一的 错误处理机制，让您可以使用语言所熟悉的方式处理请求异常。
- 目前所有语言实现的 SDK 仅提供同步请求方式。

各个不同语言的 SDK 的下载地址、详细使用说明及完整的编程参考请见如下表格：

SDK 语言	相关文档	源码
Java	*Java SDK*，用户指南	*GitHub*
.NET	*.NET SDK*，用户指南	*GitHub*
PHP	*PHP SDK*	*GitHub*

SDK 语言	相关文档	源码
Node.js		GitHub
Python	Python SDK，用户指南	GitHub
C	请参考 README	GitHub
GO	参考 README	GitHub
iOS	iOS SDK	GitHub
Android		GitHub

1.2 配置

就如同使用 API 和日志服务服务端交互一样，使用 SDK 也需要指定一些基本配置。目前，所有语言的 SDK 都定义了一个 Client 类作为入口类，这些基本配置信息在该入口类的构造时指定。

具体包括如下几项：

- 服务入口（Endpoint）：确认 Client 需要访问的服务入口。
- 阿里云访问秘钥（AccessKeyId/AccessKeySecret）：指定 Client 访问日志服务时使用的访问秘钥。

下面详细说明这两个配置的使用方式。

服务入口（Endpoint）

当使用 SDK 时，首先需要明确访问的日志服务 Project 所在 Region（如“华东 1 (杭州)”、“华北 1 (青岛)”等），然后选择与其匹配的日志服务入口初始化 Client。该服务入口与 API 中的 服务入口 定义一致。

- 当选择 Client 的 Endpoint 时，必须要保证您需要访问的 Project 的 Region 和 Endpoint 对应的 Region 一致，否则 SDK 将无法访问您指定的 Project。
- 由于 Client 实例只能在构造时指定该服务入口，如果需要访问不同 Region 里的 Project，则需要用不同的 Endpoint 构建不同的 Client 实例。
- 目前，所有 API 的服务入口仅支持 HTTP 协议。
- 如果在阿里云 ECS 虚拟机内使用 SDK，您还可以使用内网 Endpoint 避免公网带宽开销，具体请参考 服务入口。

访问秘钥（AccessKey）

正如 访问秘钥 所述，所有和日志服务端交互的请求都必须经过安全验证，而访问秘钥就是用来对请求进行安全验证的关键因子，且以 AccessKeyId 和 AccessKeySecret 方式成对出现。在 Client 构造时需要指定两个参数（AccessKeyId，AccessKeySecret）即为该访问秘钥对。所以，在使用 SDK 前，请在阿里云控制台 秘钥管理页面获取（或者创建）合适的秘钥对。

说明：

- 您的账号下可以拥有多组访问秘钥对，但在构造 Client 时指定的 AccessKeyId 和 AccessKeySecret 必须成对，否则无法通过服务端的安全验证。
- 指定的访问秘钥对必须处于“启用”状态，否则会被服务端拒绝请求。同样，您也可以到云控制台查看访问秘钥的状态。

示例

如果您需要访问某个 Project，且当前已经拥有一对处于“启用”状态的访问秘钥对。如下：

```
AccessKeyId = "bq2sjzesjmo**************"
AccessKeySecret = "4fdO2fTDDnZPU/*************"
```

则可以如下实例化对应的 Client 实例：

Java：

```
String endpoint = "regionid.example.com";       //在实际使用中，请按照您实际的服务入口和接入方式编写。
String accessKeyId = "bq2sjzesjmo**************";        //用户访问秘钥对中的 AccessKeyId。
String accessKeySecret = "4fdO2fTDDnZPU/*************";//用户访问秘钥对中的 AccessKeySecret。
Client client = new Client(endpoint, accessKeyId, acccessKeySecret);
//use client to operate log service project.....
```

.NET(C#)：

```
String endpoint = "regionid.example.com";       // 在实际使用中，请按照您实际的服务入口和接入方式编写。
String accessKeyId = "bq2sjzesjmo**************";        //用户访问秘钥对中的 AccessKeyId。
String accessKeySecret = "4fdO2fTDDnZPU/*************";//用户访问秘钥对中的 AccessKeySecret。
SLSClient client = new SLSClient(endpoint, accessKeyId, accessKeySecret);
```

```
//use client to operate sls project......
```

PHP：

```
$endpoint = 'regionid.example.com'; //此处以hangzhou Region为例，在实际使用中，请按照您实际的服务入口和接入方式编写。
$accessKeyId = 'bq2sjzesjmo**************';  //用户访问秘钥对中的AccessKeyId。
$accessKey = '4fdO2fTDDnZPU/*************';//用户访问秘钥对中的AccessKeySecret。
$client = new Aliyun_Sls_Client($endpoint, $accessKeyId, $accessKey);
//use client to operate sls project......
```

Python：

```
# // 在实际使用中，请按照您实际的服务入口和接入方式编写。
endpoint = 'regionid.example.com'
# 用户访问秘钥对中的 AccessKeyId。
accessKeyId = 'bq2sjzesjmo*************'
# 用户访问秘钥对中的 AccessKeySecret。
accessKey = '4fdO2fTDDnZPU/*************'
client = LogClient(endpoint, accessKeyId, accessKey)
#use client to operate log project......
```

1.3 错误处理

SDK 可能出现的异常错误可以分成如下几类：

- 由日志服务端返回的错误。这类错误由日志服务端返回并由 SDK 处理。关于这类错误的详细细节可以参考日志服务 API 的通用错误码和各个 API 接口的具体说明。
- 由 SDK 在向服务端发出请求时出现的网络错误。这类错误包括网络连接不通，服务端返回超时等。
- 由 SDK 自身产生的、与平台及语言相关的错误，如内存溢出等。

目前，各个语言 SDK 的实现都采取抛出异常的方式处理错误。具体原则如下：

- 由如上第一或者第二类错误将会被 SDK 处理并包装在统一的 LogException 类抛出给用户处理。
- 由如上第三类错误不会被 SDK 处理，而是直接抛出平台及语言的 Native Exception 类给用户处理。

LogException

LogException 类是 SDK 定义的、用于处理日志服务自身逻辑错误的异常类。它继承自各个语言的异常基类，提供如下异常信息：

- 错误代码（Error Code）：标示错误类型。如果是来自服务端的返回错误，则这个错误代码 API 返回的错误代码一致。如果是 SDK 网络请求错误，则其错误代码为 “RequestError”。具体请参考各个语言的完整 API 参考。
- 错误消息（Error Message）：标示错误消息。如果是来自服务端的响应错误，则这个错误消息 API 返回的错误消息一致。如果是 SDK 网络请求错误，则其错误消息为“request is failed.”。具体请参考各个语言的完整 API 参考。
- 错误请求 ID（Request Id）：标示当前错误对应于服务端的请求 ID。该 ID 只有在服务端返回错误消息时有效，否则为空字符串。用户可以在遇到错误请求时记下该请求 ID 并提供给日志服务团队进行问题追踪和定位。

请求失败与重试

在使用 SDK 访问日志服务端时，有可能会因为网络临时中断、传输延时过程、服务端处理过慢等一系列原因导致请求失败。目前，这类错误都直接以异常抛出，日志服务内部并未对此做任何重试逻辑。所以，您在使用 SDK 时需要自己定义相应的处理逻辑（重试请求或者直接报错等）。

示例

假设您需要访问“华东 1 (杭州)”Region 下名字为 **big-game** 的 Project，且在出现网络异常时主动重试指定次数。各语言的代码片段如下：

Java：

```
//其他代码......
String accessId = "your_access_id";   //TODO：用您的真实阿里云 AccessKeyI
d 替代。
String accessKey = "your_access_key"; //TODO：用您的真实阿里云 AccessKeyS
ecret 替代。
String project = "big-game";
String endpoint = "cn-hangzhou.sls.aliyuncs.com";
int max_retries = 3;
/*
 * 构建一个 client
 */
Client client = new Client(accessId, accessKey, endpoint);
ListLogStoresRequest lsRequest = new ListLogStoresRequest(project);
for (int i = 0; i < max_retries; i++)
{
    try
    {
        ListLogStoresResponse res = client.ListLogStores(lsRequest)
        //TODO: 处理返回的 response......
        break;
    }
    catch(LogException ex)
    {
        if (e.GetErrorCode() == "RequestError")
```

```
        {
            if ( i == max_retries - 1)
            {
                System.out.println("request is still failed after all
retries.");
                break;
            }
            else
                System.out.println("request error happens, retry it
!");
        }
        else
        {
            System.out.println("error code :" + e.GetErrorCode());
            System.out.println("error message :" + e.GetErrorMessage
());
            System.out.println("error requestId :" + e.GetRequestId
());
            break;
        }
    }
    catch(...)
    {
        System.out.println("unrecoverable exception when listing
logstores.");
        break;
    }
}
//其他代码......
```

.NET(C#)：

```
//其他代码......
String accessId = "your_access_id";     //TODO：用您的真实阿里云
AccessKeyId 替代。
String accessKey = "your_access_key";    //TODO：用您的真实阿里云
AccessKeySecret 替代。
String project = "big-game";
String endpoint = "cn-hangzhou.sls.aliyuncs.com";
int max_retries = 3;
//创建一个 Client
SLSClient client = new SLSClient(endpoint, accessId, accessKey);
ListLogstoresRequest request = new ListLogstoresRequest();
request.Project = project;
for (int i = 0; i < max_retries; i++)
{
    try
    {
        ListLogstoresResponse response = client.ListLogstores(request
);
        //TODO: 处理返回的 response......
        break;
    }
    catch(LogException ex)
    {
        if (e.errorCode == "SLSRequestError")
        {
            if ( i == max_retries - 1)
            {
```

```
                Console.Writeline(“request is still failed after all
retries.”);
                break;
            }
            else
            {
                Console.Writeline("request error happens, retry it!");
            }
        }
        else
        {
            Console.Writeline("error code :" + e.errorCode;
            Console.Writeline("error message :" + e.Message;
            Console.Writeline("error requestId :" + e.RequestId;
            break;
        }
    }
    catch(...)
    {
        Console.Writeline("unrecoverable exception when listing
logstores.");
        break;
    }
}
//其他代码......
```

PHP：

```
<?php
//其他代码......
$endpoint = 'cn-hangzhou.sls.aliyuncs.com';
$accessId = 'your_access_id'; // TODO：用你的真实阿里云 AccessKeyId 替代
$accessKey = 'your_access_key'; //TODO：用你的真实阿里云 AccessKeySecret
 替代
$maxRetries = 3;
// 构建一个 sls client
$client = new Aliyun_Sls_Client($endpoint, $accessId, $accessKey);
$project = 'big-game';
$request = new Aliyun_Sls_Models_ListLogstoresRequest($project);
for($i = 0; $i < $maxRetries; ++$i)
{
    try
    {
        $response = $client->ListLogstores($request);
        //TODO: 处理返回的 response......
        break;
    }
    catch (Aliyun_Sls_Exception $e)
    {
        if ($e->getErrorCode()=='RequestError')
        {
            if ($i+1 == $maxRetries)
            {
                echo "error code :" . $e->getErrorCode() . PHP_EOL;
                echo "error message :" . $e->getErrorMessage() .
PHP_EOL;
                break;
            }
            echo 'request error happens, retry it!' . PHP_EOL;
```

```
        }
        else
        {
            echo "error code :" . $e->getErrorCode() . PHP_EOL;
            echo "error message :" . $e->getErrorMessage() . PHP_EOL;
            echo "error requestId :" . $e->getRequestId() . PHP_EOL;
            break;
        }
    }
    catch (Exception $ex)
    {
        echo 'unrecoverable exception when listing logstores.' .
PHP_EOL;
        var_dump($ex);
        break;
    }
}
//其他代码......
```

Python：

```
//其他代码......
endpoint = 'cn-hangzhou.sls.aliyuncs.com'
accessId = 'your_access_id' # TODO：用你的真实阿里云 AccessKeyId 替代
accessKey = 'your_access_key' # TODO：用你的真实阿里云 AccessKeySecret 替
代
maxRetries = 3
# 构建一个 client
client = Client(endpoint, accessId, accessKey)
project = 'big-game'
lsRequest = ListLogstoresRequest(project)
for i in xrange(maxRetries):
    try:
        res = client.ListLogstores(lsRequest)
        # TODO: 处理返回的 response......
        break
    except LogException as e:
        if e.getErrorCode() == "RequestError":
            if i+1 == maxRetries:
                print "error code :" + e.getErrorCode()
                print "error message :" + e.getErrorMessage()
                break
            else:
                print "request error happens, retry it!"
        else:
            print "error code :" + e.getErrorCode()
            print "error message :" + e.getErrorMessage()
            print "error requestId :" + e.getRequestId()
            break
    except Exception as e:
        print 'unrecoverable exception when listing logstores.'
        break
```

```
//其他代码......
```

1.4 接口规范

尽管不同语言的 SDK 实现有所不同，但其接口都遵循 Request-Response 原则，即对 API 的调用按照如下方式进行：

1. 利用请求参数构建相应的 Request 实例。
2. 调用 SDK 中的相应接口并传入上一步的 Request 实例。
3. SDK 接口的返回结果以相应的 Response 实例返回给用户。

如下代码片段解释如何按照上面流程获取一个 Project 下的所有 Logstore 的名称。

Java

```
// 其他代码......
String accessId = "your_access_id";     //TODO：用您的真实阿里云
AccessKeyId 替代。
String accessKey = "your_access_key";   //TODO：用您的真实阿里云
AccessKeySecret 替代。
String project = "your_project";        //TODO: 用您的真实 project 名称替
代。
String endpoint = "region_endpoint";//TODO: 此处以hangzhou Region为
例，在实际使用中，请按照您实际的服务入口和接入方式编写。
//构建一个 Client 实例。
Client client = new Client(endpoint, accessId, accessKey);
//用请求参数“project”初始化 ListLogstores 的请求类。
ListLogStoresRequest lsRequest = new ListLogStoresRequest(project);
//使用 request 实例调用 ListLogstores 接口，且返回参数为对应的 Response 实例
ListLogStoresResponse res = client.ListLogStores(lsRequest);
//访问 Response 实例获取请求结果
ArrayList<String> names = res.GetLogStores();
// 其他代码......
```

.NET(C#)

```
// 其他代码......
String accessId = "your_access_id";     //TODO：用您的真实阿里云
AccessKeyId 替代。
String accessKey = "your_access_key";   //TODO：用您的真实阿里云
AccessKeySecret 替代。
String project = "your_project";        //TODO: 用您的真实 project 名称替
代。
String endpoint = "region_endpoint";////TODO: 此处以hangzhou Region为
例，在实际使用中，请按照您实际的服务入口和接入方式编写。
//构建一个 Client 实例。
SLSClient client = new SLSClient(endpoint, accessId, accessKey);
//用请求参数“project”初始化 ListLogstores 的请求类。
ListLogStoresRequest lsRequest = new ListLogStoresRequest();
```

```
lsRequest.Project = project;
//使用 request 实例调用 ListLogstores 接口，且返回参数为对应的 Response 实例
ListLogStoresResponse res = client.ListLogStores(lsRequest);
//访问 Response 实例获取请求结果
List<String> names = res.Logstores;
// 其他代码......
```

PHP

```
// 其他代码......
$accessId = "your_access_id";     //TODO：用您的真实阿里云 AccessKeyId 替代。
$accessKey = "your_access_key";   //TODO：用您的真实阿里云 AccessKeySecret 替代。
$project = "your_project";        //TODO: 用您的真实 project 名称替代。
$endpoint = "region_endpoint";////TODO: 此处以hangzhou Region为例，在实际使用中，请按照您实际的服务入口和接入方式编写。
//构建一个 SLS Client 实例。
$client = new Aliyun_Sls_Client($endpoint, $accessId, $accessKey);
//用请求参数“project”初始化 ListLogstores 的请求类。
$request = new Aliyun_Sls_Models_ListLogstoresRequest($project);
//使用 request 实例调用 ListLogstores 接口，且返回参数为对应的 Response 实例
$response = $client->listLogstores($request);
//访问 Response 实例获取请求结果
$names = $response->getLogstores();
// 其他代码......
```

Python

```
// 其他代码......
accessId = 'your_access_id';     //TODO：用您的真实阿里云 AccessKeyId 替代。
accessKey = 'your_access_key';   //TODO：用您的真实阿里云 AccessKeySecret 替代。
project = 'your_project';        //TODO: 用您的真实 project 名称替代。
endpoint = 'region_endpoint';////TODO: 此处以hangzhou Region为例，在实际使用中，请按照您实际的服务入口和接入方式编写。
# 构建一个 client
client = LogClient(endpoint, accessId, accessKey)
# 用请求参数“project”初始化 ListLogstores 的请求类。
lsRequest = ListLogstoresRequest(project)
# 使用 request 实例调用 ListLogstores 接口，且返回参数为对应的 Response 实例
res = client.list_logstores(lsRequest)
# 访问 Response 实例获取请求结果
names = res.get_logstores();
// 其他代码......
```

SDK 实现了多组类似 ListLogStores 的接口，也定义了相应的 Request 和 Response 类。除去 Request-Response 风格的基础接口外，各个不同语言的 SDK 还会提供一些包装了这些基础接口的

辅助接口，让您无需自己构建 Request 及解析最终 Reponse 内容。这类接口的细节请见各 SDK 的 API 参考。

2 Java SDK

下载地址

Log Service 的 Java SDK 让 Java 开发人员可以非常方便地使用 Java 程序操作阿里云日志服务。开发者可以直接使用Maven依赖添加SDK，也可以下载包到本地。目前，SDK 支持 J2SE 6.0 及以上版本，单击此处下载最新版完整SDK。

操作步骤

为快速开始使用 Log Service Java SDK，请按照如下步骤进行。

步骤 1 创建阿里云账号

具体方法请参考 阿里云账号注册流程。

步骤 2 获取阿里云访问密钥

为了使用 Log Service Java SDK，您必须申请阿里云的访问秘钥。

登录阿里云 秘钥管理页面。选择一对用于 SDK 的访问密钥对。如果没有，请创建一对新访问密钥，且保证它处于启用状态。有关如何创建访问密钥，参见 准备流程。

该密钥对会在下面的步骤使用，且需要保管好，不能对外泄露。另外，您可以参考 配置 了解更多 SDK 如何使用访问密钥的信息。

步骤 3 创建日志服务项目和日志库

在使用日志服务Java SDK之前，请先在控制台上创建好项目（Project）和日志库（Logstore）。

有关如何创建Project和Logstore，参见准备流程。

说明：

- 请确保使用同一阿里云账号获取阿里云访问密钥和创建日志项目及日志库。
- 关于日志的项目、日志库等概念请参考 Log 基本概念。
- Log 的 Project 名称为日志服务全局唯一，而 Logstore 名称在一个 Project 下面唯一。
- Log 的 Project 一旦创建则无法更改它的所属区域。目前也不支持在不同阿里云 Region 间迁移 Log Project。

步骤 4 安装 Java 开发环境

目前，Log Java SDK 支持 J2SE 6.0 及以上的 Java 运行环境，您可以从 *Java* 官方网站 下载并按说明安装 Java 开发环境。

步骤 5 安装 Log Service Java SDK

在安装完 Java 开发环境后，您需要安装 Log Service Java SDK。目前，我们提供两种方式安装日志服务的 Java SDK：

1. 建议使用 *Apache Maven* 获取最新版本的 SDK，您可以添加如下配置到您的 Maven 项目。

```
<dependency>
         <groupId>com.google.protobuf</groupId>
         <artifactId>protobuf-java</artifactId>
         <version>2.5.0</version>
</dependency>
<dependency>
<groupId>com.aliyun.openservices</groupId>
<artifactId>aliyun-log</artifactId>
<version>0.6.7</version>
<exclusions>
        <exclusion>
            <groupId>com.google.protobuf</groupId>
            <artifactId>protobuf-java</artifactId>
       </exclusion>
</exclusions>
</dependency>
```

2. 您也可以完整下载 Java SDK 软件包，然后在自己的 Java 项目中直接引用本地软件包。

   a. 从 这里 克隆 Java SDK 包（版本会定期更新，如需使用最新版本请使用 Maven）。

   b. 解压完整下载的包到指定的目录即可。Java SDK 是一个软件开发包，不需要额外的安装操作。

   c. 把 SDK 包中的所有 Jar 包（包括依赖的第三方包）添加到您的 Java 工程（具体操作请参照不同的 IDE 文档）。

步骤 6 开始一个新的 Java 项目

现在，您可以开始使用 SDK Java SDK。使用任何文本编辑器或者 Java IDE，运行如下示例代码即可与 Log Service 服务端交互并得到相应输出。Java SDK使用上的一些注意事项请参考注意事项章节。

```
package sdksample;
import java.util.ArrayList;
import java.util.List;
import java.util.Vector;
import java.util.Date;
import com.aliyun.openservices.log.Client;
import com.aliyun.openservices.log.common.*;
```

```
import com.aliyun.openservices.log.exception.*;
import com.aliyun.openservices.log.request.*;
import com.aliyun.openservices.log.response.*;
import com.aliyun.openservices.log.common.LogGroupData;
import com.aliyun.openservices.log.common.LogItem;
import com.aliyun.openservices.log.common.Logs.Log;
import com.aliyun.openservices.log.common.Logs.Log.Content;
import com.aliyun.openservices.log.common.Logs.LogGroup;
import com.aliyun.openservices.log.common.Consts.CursorMode;
public class sdksample {
    public static void main(String args[]) throws LogException,
InterruptedException {
        String endpoint = "<log_service_endpoint>"; // 选择与上面步骤创
建 project 所属区域匹配的
        // Endpoint
        String accessKeyId = "<your_access_key_id>"; // 使用您的阿里云访
问密钥 AccessKeyId
        String accessKeySecret = "<your_access_key_secret>"; // 使用您
的阿里云访问密钥
        // AccessKeySecret
        String project = "<project_name>"; // 上面步骤创建的项目名称
        String logstore = "<logstore_name>"; // 上面步骤创建的日志库名称
        // 构建一个客户端实例
        Client client = new Client(endpoint, accessKeyId, accessKeyS
ecret);
        // 列出当前 project 下的所有日志库名称
        int offset = 0;
        int size = 100;
        String logStoreSubName = "";
        ListLogStoresRequest req1 = new ListLogStoresRequest(project,
offset, size, logStoreSubName);
        ArrayList<String> logStores = client.ListLogStores(req1).
GetLogStores();
        System.out.println("ListLogs:" + logStores.toString() + "\n");
        // 写入日志
        String topic = "";
        String source = "";
        // 连续发送 10 个数据包，每个数据包有 10 条日志
        for (int i = 0; i < 10; i++) {
            Vector<LogItem> logGroup = new Vector<LogItem>();
            for (int j = 0; j < 10; j++) {
                LogItem logItem = new LogItem((int) (new Date().
getTime() / 1000));
                logItem.PushBack("index"+String.valueOf(j), String.
valueOf(i * 10 + j));
                logGroup.add(logItem);
            }
            PutLogsRequest req2 = new PutLogsRequest(project, logstore
, topic, source, logGroup);
            client.PutLogs(req2);
            /*
             * 发送的时候也可以指定将数据发送至有一个特定的 shard，只要设置
shard 的 hashkey，则数据会写入包含该
             * hashkey 的 range 所对应的 shard，具体 API 参考以下接口：
public PutLogsResponse
             * PutLogs( String project, String logStore, String topic,
             * List<LogItem> logItems, String source, String shardHash
 // 根据
```

```
             * hashkey 确定写入 shard，hashkey 可以是 MD5(ip) 或 MD5(id
) 等 ) throws
             * LogException;
             */
        }
        // 把 0 号 shard 中，最近 1 分钟写入的数据都读取出来。
        int shard_id = 0;
        long curTimeInSec = System.currentTimeMillis() / 1000;
        GetCursorResponse cursorRes = client.GetCursor(project,
logstore, shard_id, curTimeInSec - 60);
        String beginCursor = cursorRes.GetCursor();
        cursorRes = client.GetCursor(project, logstore, shard_id,
CursorMode.END);
        String endCursor = cursorRes.GetCursor();
        String curCursor = beginCursor;
        while (curCursor.equals(endCursor) == false) {
            int loggroup_count = 2; // 每次读取两个 loggroup
            BatchGetLogResponse logDataRes = client.BatchGetLog(
project, logstore, shard_id, loggroup_count, curCursor,
                    endCursor);
            // 读取LogGroup的List
            List<LogGroupData> logGroups = logDataRes.GetLogGroups();
            for(LogGroupData logGroup: logGroups){
                FastLogGroup flg = logGroup.GetFastLogGroup();
                System.out.println(String.format("\tcategory\t:\t%s\n\
tsource\t:\t%s\n\ttopic\t:\t%s\n\tmachineUUID\t:\t%s",
                        flg.getCategory(), flg.getSource(), flg.
getTopic(), flg.getMachineUUID()));
                System.out.println("Tags");
                for (int tagIdx = 0; tagIdx < flg.getLogTagsCount(); +
+tagIdx) {
                    FastLogTag logtag = flg.getLogTags(tagIdx);
                    System.out.println(String.format("\t%s\t:\t%s",
logtag.getKey(), logtag.getValue()));
                }
                for (int lIdx = 0; lIdx < flg.getLogsCount(); ++lIdx)
 {
                    FastLog log = flg.getLogs(lIdx);
                    System.out.println("--------\nLog: " + lIdx + ",
time: " + log.getTime() + ", GetContentCount: " + log.getContentsCount
());
                    for (int cIdx = 0; cIdx < log.getContentsCount();
 ++cIdx) {
                        FastLogContent content = log.getContents(cIdx
);
                        System.out.println(content.getKey() + "\t:\t"
 + content.getValue());
                    }
                }
            }
            String next_cursor = logDataRes.GetNextCursor();
            System.out.println("The Next cursor:" + next_cursor);
            curCursor = next_cursor;
        }
        // ！！！重要提示 ： 只有打开索引功能，才能调用以下接口 ！！！
        // 等待 1 分钟让日志可查询
        try {
            Thread.sleep(60 * 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
        // 查询日志分布情况
        String query = "<此处为需要查询的关键词，如果查询全部内容设置为空字符串
即可>";
        int from = (int) (new Date().getTime() / 1000 - 300);
        int to = (int) (new Date().getTime() / 1000);
        GetHistogramsResponse res3 = null;
        while (true) {
            GetHistogramsRequest req3 = new GetHistogramsRequest(
project, logstore, topic, query, from, to);
            res3 = client.GetHistograms(req3);
            if (res3 != null && res3.IsCompleted()) // IsCompleted
() 返回
            // true，表示查询结果是准确的，如果返回
            // false，则重复查询
            {
                break;
            }
            Thread.sleep(200);
        }
        System.out.println("Total count of logs is " + res3.GetTotalCo
unt());
        for (Histogram ht : res3.GetHistograms()) {
            System.out.printf("from %d, to %d, count %d.\n", ht.
GetFrom(), ht.GetTo(), ht.GetCount());
        }
        // 查询日志数据
        long total_log_lines = res3.GetTotalCount();
        int log_offset = 0;
        int log_line = 10;//log_line 最大值为100，每次获取100行数据。若需
要读取更多数据，请使用offset翻页。offset和lines只对关键字查询有效，若使用SQL查
询，则无效。在SQL查询中返回更多数据，请使用limit语法。
        while (log_offset <= total_log_lines) {
            GetLogsResponse res4 = null;
            // 对于每个 log offset,一次读取 10 行 log，如果读取失败，最多重复
读取 3 次。
            for (int retry_time = 0; retry_time < 3; retry_time++) {
                GetLogsRequest req4 = new GetLogsRequest(project,
logstore, from, to, topic, query, log_offset,
                        log_line, false);
                res4 = client.GetLogs(req4);
                if (res4 != null && res4.IsCompleted()) {
                    break;
                }
                Thread.sleep(200);
            }
            System.out.println("Read log count:" + String.valueOf(res4
.GetCount()));
            log_offset += log_line;
        }
        //打开分析功能,只有打开分析功能，才能使用SQL 功能。 可以在控制台开通分析
功能，也可以使用SDK开启分析功能
        IndexKeys indexKeys = new IndexKeys();
        ArrayList<String> tokens = new ArrayList<String>();
        tokens.add(",");
        tokens.add(".");
        tokens.add("#");
        IndexKey keyContent = new IndexKey(tokens,false,"text");
        indexKeys.AddKey("index0",keyContent);
```

```
        keyContent = new IndexKey(new ArrayList<String>(),false,"long
");
        indexKeys.AddKey("index1",keyContent);
        keyContent = new IndexKey(new ArrayList<String>(),false,"
double");
        indexKeys.AddKey("index2",keyContent);
        IndexLine indexLine = new IndexLine(new ArrayList<String>(),
false);
        Index index = new Index(7,indexKeys,indexLine);
        CreateIndexRequest createIndexRequest = new CreateIndexRequest
(project,logstore,index);
        client.CreateIndex(createIndexRequest);
        //使用分析功能
        GetLogsRequest req4 = new GetLogsRequest(project, logstore,
from, to, "", " index0:value | select avg(index1) as v1,sum(index2) as
 v2, index0 group by index0");
        GetLogsResponse res4 = client.GetLogs(req4);
        if(res4 != null && res4.IsCompleted()){
            for (QueriedLog log : res4.GetLogs()){
                LogItem item = log.GetLogItem();
                for(LogContent content : item.GetLogContents()){
                    System.out.print(content.GetKey()+":"+content.
GetValue());
                }
                System.out.println();
            }
        }
    }
}
```

注意事项

1. 为了提高您的系统的IO效率，请尽量不要直接使用SDK往日志服务中写数据，写数据标准做法参考文章*Producer Library*。
2. 要消费日志服务中的数据，请尽量不要直接使用SDK的拉数据接口，我们提供了一个高级消费库消费组消费，该库屏蔽了日志服务的实现细节，并且提供了负载均衡、按序消费等高级功能。

3 .NET SDK

下载地址

日志服务的 .NET SDK让Windows 平台上的开发人员可以非常方便地使用 .NET 平台操作阿里云日志服务。目前，SDK 支持 .NET Framework 3.5/4.0/4.5 版本。尽管针对不同 .NET Framework 版本的 SDK 文件不同，但是其接口和实现功能完全一致。

该 SDK GitHub 地址如下：单击此处进入*GitHub*

操作步骤

为快速开始使用 Log Service .NET SDK，请按照如下步骤进行。

步骤 **1** 创建阿里云账号

具体方法请参考 阿里云账号注册流程。

步骤 **2** 获取阿里云访问密钥

为了使用 Log Service .NET SDK，您必须申请阿里云的 访问秘钥。

登录阿里云 秘钥管理页面。选择一对用于 SDK 的访问密钥对。如果没有，请创建一对新访问密钥，且保证它处于启用状态。有关如何创建访问密钥，参见 准备流程。

该密钥对会在下面的步骤使用，且需要保管好，不能对外泄露。另外，您可以参考 配置 了解更多 SDK 如何使用访问密钥的信息。

步骤 **3** 创建日志服务项目和日志库

在使用日志服务.NET SDK之前，请先在控制台上创建好项目（Project）和日志库（Logstore）。

有关如何创建Project和Logstore，参见准备流程。

说明：

- 请确保使用同一阿里云账号获取阿里云访问密钥和创建日志项目及日志库。
- 关于日志的项目、日志库等概念请参考 Log 基本概念。
- Log 的 Project 名称为日志服务全局唯一，而 Logstore 名称在一个 Project 下面唯一。
- Log 的 Project 一旦创建则无法更改它的所属区域。目前也不支持在不同阿里云 Region 间迁移 Log Project。

步骤 4 安装 .NET 开发环境

目前，日志服务 SDK 支持 .NET 3.5 和 .NET 4.0/4.5 运行环境。为支持日志服务 SDK 开发，建议安装：

- Microsoft .NET Framework 3.5/4.0/4.5 （具体版本依赖于您的程序需要运行的目标环境）
- Visual Studio 2010 及其以后版本

步骤 5下载并安装日志服务 .NET SDK

搭建好 .NET 开发环境后，您需要安装日志服务的 .NET SDK。具体如下：

1. 下载。
   - 通过 Github 下载：*https://github.com/aliyun/aliyun-log-csharp-sdk*
   - 历史版本下载：从 这里 下载最新的日志服务 .NET SDK 包
2. 解压完整下载的包到指定的目录即可。日志服务 .NET SDK 是一个软件开发包，不需要额外的安装操作。您可以按照下面的步骤直接在自己的 Visual Studio 项目中使用。

步骤 6 开始一个新的日志服务 .NET 项目

安装好 .NET 开发环境及日志服务 .NET SDK 后，您就可以开始创建日志服务的 .NET 项目。具体请参见 *Github* SLSSDK40 解决方案中的LOGSDKSample项目。

4 .NET Core SDK

背景信息

日志服务的 .NET Core SDK让跨平台的开发人员可以非常方便地使用.NET Core框架操作阿里云日志服务。

.NET Core SDK GitHub 地址：单击进入*GitHub*。

操作步骤

1. 创建阿里云账号。

   具体方法请参考 阿里云账号注册流程。

2. 获取阿里云访问密钥。

   为了使用 Log Service .NET Core SDK，您必须申请阿里云的 访问秘钥。

   登录阿里云 秘钥管理页面。选择一对用于 SDK 的访问密钥对。如果没有，请创建一对新访问密钥，且保证它处于**启用**状态。有关如何创建访问密钥，参见 准备流程。

   该密钥对会在下面的步骤使用，且需要保管好，不能对外泄露。另外，您可以参考 配置 了解更多 SDK 如何使用访问密钥的信息。

3. 创建日志服务项目和日志库。

   在使用日志服务.NET Core SDK之前，请先在控制台上创建好项目（Project）和日志库（Logstore）。

   有关如何创建Project和Logstore，参见准备流程。

   > **说明：**
   >
   > - 请确保使用同一阿里云账号获取阿里云访问密钥、创建日志项目及日志库。
   > - 关于日志的项目、日志库等概念请参考基本概念。
   > - Log 的 Project 名称为日志服务全局唯一，而 Logstore 名称在一个 Project 下唯一。
   > - Log 的 Project 一旦创建则无法更改它的所属区域。目前也不支持在不同阿里云 Region 间迁移 Log Project。

4. 安装 .NET Core开发环境。

   目前，日志服务.NET Core SDK 支持以下平台版本：

- .NET Core 2.0
- .NET Framework (with .NET Core 1.x SDK) 4.6.2
- .NET Framework (with .NET Core 2.0 SDK) 4.6.1

支持的全部版本请参考*GitHub*。

5. 下载并安装日志服务.NET Core SDK。

   搭建好.NET Core开发环境后，您需要安装日志服务的 .NET Core SDK。

   a) 下载.NET Core SDK。

      单击此处进入Github下载。

   b) 解压安装包到指定的目录。

      日志服务.NET Core SDK是一个软件开发包，不需要额外的安装操作。您可以直接在自己的 Visual Studio 项目中使用。

6. 开始一个新的日志服务.NET Core项目。

   安装好 .NET 开发环境及日志服务.NET Core SDK后，您就可以开始创建日志服务的.NET Core项目。具体请参见*Github Wiki*。

5 PHP SDK

下载地址

该 SDK GitHub 地址如下：*https://github.com/aliyun/aliyun-log-php-sdk*。

操作步骤

为快速开始使用 Log Service PHP SDK，请按照如下步骤进行。

步骤 **1** 创建阿里云账号

具体方法请参考 阿里云账号注册流程。

步骤 **2** 获取阿里云访问密钥

为了使用 Log Service PHP SDK，您必须申请阿里云的 访问秘钥。

登录阿里云 秘钥管理页面。选择一对用于 SDK 的访问密钥对。如果没有，请创建一对新访问密钥，且保证它处于启用状态。有关如何创建访问密钥，参见 准备流程。

该密钥对会在下面的步骤使用，且需要保管好，不能对外泄露。另外，您可以参考 配置 了解更多 SDK 如何使用访问密钥的信息。

步骤 **3** 创建日志服务项目和日志库

在使用日志服务PHP SDK之前，请先在控制台上创建好项目（Project）和日志库（Logstore）。

有关如何创建Project和Logstore，参见准备流程。

说明：

- 请确保使用同一阿里云账号获取阿里云访问密钥和创建日志项目及日志库。
- 关于日志的项目、日志库等概念请参考 Log 基本概念。
- Log 的 Project 名称为日志服务全局唯一，而 Logstore 名称在一个 Project 下面唯一。
- Log 的 Project 一旦创建则无法更改它的所属区域。目前也不支持在不同阿里云 Region 间迁移 Log Project。

步骤 **4** 安装 **PHP** 开发环境

PHP SDK 支持 PHP 5.2.1 及以上版本，您可以在本地安装 SDK 所支持的任何 PHP 版本并搭建好相应的 PHP 开发环境。

步骤 5 下载并安装 PHP SDK

搭建好 PHP 开发环境后，您需要安装 PHP SDK。具体如下：

1. 从 *GitHub* 下载最新的 PHP SDK 包。
2. 解压完整下载的包到指定的目录即可。PHP SDK 是一个软件开发包，不需要额外的安装操作。在个软件开发包除了包括 SDK 自身的代码外，还有一组第三方依赖包和一个 autoloader 类帮助你简化使用流程。您可以按照下面的步骤直接在自己的 PHP 项目中使用。

步骤 6 开始一个新的 PHP 项目

在，您可以开始使用 SDK PHP SDK。使用任何文本编辑器或者 PHP IDE，运行如下示例代码即可与日志服务端交互并得到相应输出。

```
<?php
/* 使用 autoloader 类自动加载所有需要的 PHP 模块。注意使用合适的路径指向
autoloader 类所在文件*/
require_once realpath(dirname(__FILE__) . '/../Log_Autoload.php');
$endpoint = 'cn-hangzhou.sls.aliyuncs.com'; // 选择与上面步骤创建 project
 所属区域匹配的 Endpoint
$accessKeyId = 'your_access_key_id';        // 使用你的阿里云访问秘钥
AccessKeyId
$accessKey = 'your_access_key';             // 使用你的阿里云访问秘钥
AccessKeySecret
$project = 'your_project';                  // 上面步骤创建的项目名称
$logstore = 'your_logstore';                // 上面步骤创建的日志库名称
$client = new Aliyun_Log_Client($endpoint, $accessKeyId, $accessKey);
#列出当前 project 下的所有日志库名称
$req1 = new Aliyun_Log_Models_ListLogstoresRequest($project);
$res1 = $client->listLogstores($req1);
var_dump($res1);
#创建 logstore
$req2 = new Aliyun_Log_Models_CreateLogstoreRequest($project,$logstore
,3,2);
$res2 = $client -> createLogstore($req2);
#等待 logstore 生效
sleep(60);
#写入日志
$topic = "";
$source = "";
$logitems = array();
for ($i = 0; $i < 5; $i++)
{
    $contents = array('index1'=> strval($i));
    $logItem = new Aliyun_Log_Models_LogItem();
    $logItem->setTime(time());
    $logItem->setContents($contents);
    array_push($logitems, $logItem);
}
$req2 = new Aliyun_Log_Models_PutLogsRequest($project, $logstore, $
topic, $source, $logitems);
$res2 = $client->putLogs($req2);
var_dump($res2);
```

```
#不必等待，立即拖数据
#首先遍历有哪些 shardId
$listShardRequest = new Aliyun_Log_Models_ListShardsRequest($project,$
logstore);
$listShardResponse = $client -> listShards($listShardRequest);
foreach($listShardResponse-> getShardIds()  as $shardId)
{
    #对每一个 ShardId，先获取 Cursor
    $getCursorRequest = new Aliyun_Log_Models_GetCursorRequest($
project,$logstore,$shardId,null, time() - 60);
    $response = $client -> getCursor($getCursorRequest);
    $cursor = $response-> getCursor();
    $count = 100;
    while(true)
    {
        #从 cursor 开始读数据
        $batchGetDataRequest = new Aliyun_Log_Models_BatchGetLogs
Request($project,$logstore,$shardId,$count,$cursor);
        var_dump($batchGetDataRequest);
        $response = $client -> batchGetLogs($batchGetDataRequest);
        if($cursor == $response -> getNextCursor())
        {
            break;
        }
        $logGroupList = $response -> getLogGroupList();
        foreach($logGroupList as $logGroup)
        {
            print ($logGroup->getCategory());
            foreach($logGroup -> getLogsArray() as $log)
            {
                foreach($log -> getContentsArray() as $content)
                {
                    print($content-> getKey().":".$content->getValue
().”\t”);
                }
                print("\n");
            }
        }
        $cursor = $response -> getNextCursor();
    }
}
#等待 1 分钟让日志可查询
sleep(60);
#查询日志分布情况询（注意，要查询日志，必须保证已经创建了索引，PHP SDK 不提供该接
口，请在控制台创建）
$topic = "";
$query='';
$from = time()-3600;
$to = time();
$res3 = NULL;
while (is_null($res3) || (! $res3->isCompleted()))
{
    $req3 = new Aliyun_Log_Models_GetHistogramsRequest($project, $
logstore, $from, $to, $topic, $query);
    $res3 = $client->getHistograms($req3);
}
var_dump($res3);
#查询日志数据
$res4 = NULL;
while (is_null($res4) || (! $res4->isCompleted()))
{
```

```
    $req4 = new Aliyun_Log_Models_GetLogsRequest($project, $logstore,
 $from, $to, $topic, $query, 5, 0, False);
    $res4 = $client->getLogs($req4);
}
var_dump($res4);
```

6 Python SDK

下载地址

该 SDK GitHub 地址如下：

https://github.com/aliyun/aliyun-log-python-sdk

操作步骤

为快速开始使用 Log Service Python SDK，请按照如下步骤进行。

步骤 1 创建阿里云账号

具体方法请参考 阿里云账号注册流程。

步骤 2 获取阿里云访问密钥

为了使用 Log Service Python SDK，您必须申请阿里云的 访问秘钥。

登录阿里云 秘钥管理页面。选择一对用于 SDK 的访问密钥对。如果没有，请创建一对新访问密钥，且保证它处于启用状态。有关如何创建访问密钥，参见 准备流程。

该密钥对会在下面的步骤使用，且需要保管好，不能对外泄露。另外，您可以参考 配置 了解更多 SDK 如何使用访问密钥的信息。

步骤 3 创建日志服务项目和日志库

在使用日志服务Python SDK之前，请先在控制台上创建好项目（Project）和日志库（Logstore）。

在进行SDK操作之前，请先在控制台上创建Project和Logstore。

1. 登录日志服务管理控制台。
2. 单击右上角的创建**Project**。
3. 填写**Project**名称和所属地域，单击确认。
4. 在**Project**列表页面，单击项目的名称，然后单击创建 创建日志库。

   或者在创建完项目后，根据系统提示创建日志库，单击创建。
5. 填写日志库的配置信息并单击确认。

   您需要在此处填写Logstore名称、数据保存时间、Shard数目等。请按照您的需求合理设置*Shard*数目，例如本文实例中，您需要设置4个Shard。

说明：

- 请确保使用同一阿里云账号获取阿里云访问密钥和创建日志项目及日志库。
- 关于日志的项目、日志库等概念请参考 Log 基本概念。
- Log 的 Project 名称为日志服务全局唯一，而 Logstore 名称在一个 Project 下面唯一。
- Log 的 Project 一旦创建则无法更改它的所属区域。目前也不支持在不同阿里云 Region 间迁移 Log Project。

步骤 4 安装 Python 环境

Python SDK是一个纯Python的库，支持在所有运行Python的操作系统。目前主要为Linux、Mac OS X和Windows。请如下安装Python：

1. 下载并安装最新的Python 安装包。

   

   说明：

   - 目前，Python SDK支持Python 2.6/2.7和Python 3.3/3.4/3.5/3.6环境。您可以运行`python -V`查询当前的Python运行版本。
   - Python官方不再支持Python 2.6和Python 3.3，推荐您使用Python 2.7、Python 3.4及以后版本。

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，并查看当前pip版本。

步骤 5 安装Python SDK

Shell下以管理员权限执以下命令，安装Python SDK。

```
pip install -U aliyun-log-python-sdk
```

步骤 6 开始一个 Python 程序

现在，你可以开始使用 SDK Python SDK。使用任何文本编辑器或者 Python IDE，运行如下示例代码即可与服务端交互并得到相应输出。

更多信息请参考*Github* 项目地址以及更多详细说明。

```
# encoding: utf-8
import time
from aliyun.log.logitem import LogItem
from aliyun.log.logclient import LogClient
from aliyun.log.getlogsrequest import GetLogsRequest
from aliyun.log.putlogsrequest import PutLogsRequest
from aliyun.log.listlogstoresrequest import ListLogstoresRequest
from aliyun.log.gethistogramsrequest import GetHistogramsRequest
```

```
def main():
    endpoint = ''       # 选择与上面步骤创建Project所属区域匹配的Endpoint
    accessKeyId = ''    # 使用您的阿里云访问密钥AccessKeyId
    accessKey = ''      # 使用您的阿里云访问密钥AccessKeySecret
    project = ''        # 上面步骤创建的项目名称
    logstore = ''       # 上面步骤创建的日志库名称
    # 重要提示：创建的logstore请配置为4个shard以便于后面测试通过
    # 构建一个client
    client = LogClient(endpoint, accessKeyId, accessKey)
    # list 所有的logstore
    req1 = ListLogstoresRequest(project)
    res1 = client.list_logstores(req1)
    res1.log_print()
    topic = ""
    source = ""
    # 发送10个数据包，每个数据包有10条log
    for i in range(10):
        logitemList = []  # LogItem list
        for j in range(10):
            contents = [('index', str(i * 10 + j))]
            logItem = LogItem()
            logItem.set_time(int(time.time()))
            logItem.set_contents(contents)
            logitemList.append(logItem)
        req2 = PutLogsRequest(project, logstore, topic, source,
logitemList)
        res2 = client.put_logs(req2)
        res2.log_print()
    # list所有的shard，读取上1分钟写入的数据全部读取出来
    listShardRes = client.list_shards(project, logstore)
    for shard in listShardRes.get_shards_info():
        shard_id = shard["shardID"]
        start_time = int(time.time() - 60)
        end_time = start_time + 60
        res = client.get_cursor(project, logstore, shard_id,
start_time)
        res.log_print()
        start_cursor = res.get_cursor()
        res = client.get_cursor(project, logstore, shard_id, end_time)
        end_cursor = res.get_cursor()
        while True:
            loggroup_count = 100  # 每次读取100个包
            res = client.pull_logs(project, logstore, shard_id,
start_cursor, loggroup_count, end_cursor)
            res.log_print()
            next_cursor = res.get_next_cursor()
            if next_cursor == start_cursor:
                break
            start_cursor = next_cursor
    # 重要提示： 只有打开索引功能，才可以使用以下接口来查询数据
    time.sleep(60)
    topic = ""
    query = "index"
    From = int(time.time()) - 600
    To = int(time.time())
    res3 = None
    # 查询最近10分钟内，满足query条件的日志条数，如果执行结果不是完全正确，则进行
重试
    while (res3 is None) or (not res3.is_completed()):
```

```
        req3 = GetHistogramsRequest(project, logstore, From, To, topic
, query)
        res3 = client.get_histograms(req3)
    res3.log_print()
    # 获取满足query的日志条数
    total_log_count = res3.get_total_count()
    log_line = 10
    # 每次读取10条日志，将日志数据查询完，对于每一次查询，如果查询结果不是完全准
确，则重试3次
    for offset in range(0, total_log_count, log_line):
        res4 = None
        for retry_time in range(0, 3):
            req4 = GetLogsRequest(project, logstore, From, To, topic,
query, log_line, offset, False)
            res4 = client.get_logs(req4)
            if res4 is not None and res4.is_completed():
                break
            time.sleep(1)
        if res4 is not None:
            res4.log_print()
    listShardRes = client.list_shards(project, logstore)
    shard = listShardRes.get_shards_info()[0]
    # 分裂shard
    if shard["status"] == "readwrite":
        shard_id = shard["shardID"]
        inclusiveBeginKey = shard["inclusiveBeginKey"]
        midKey = inclusiveBeginKey[:-1] + str((int(inclusiveBeginKey[-
1:])) + 1)
        client.split_shard(project, logstore, shard_id, midKey)
    # 合并shard
    shard = listShardRes.get_shards_info()[1]
    if shard["status"] == "readwrite":
        shard_id = shard["shardID"]
        client.merge_shard(project, logstore, shard_id)
    # 删除shard
    shard = listShardRes.get_shards_info()[-1]
    if shard["status"] == "readonly":
        shard_id = shard["shardID"]
        client.delete_shard(project, logstore, shard_id)
   # 创建外部数据源
    res = client.create_external_store(project,ExternalSt
oreConfig("rds_store","cn-qingdao","rds-vpc","vpc-************","
i***********","*.*.*.*","3306","root","sfdsfldsfksflsdfs","meta","
join_meta"));
    res.log_print()
    res = client.update_external_store(project,ExternalStoreConfig
("rds_store","cn-qingdao","rds-vp,"rds-vpc","vpc-************","i
************","*.*.*.*","3306","root","sfdsfldsfksflsdfs","meta","
join_meta"));
    res.log_print()
    res = client.get_external_store(project,"rds_store");
    res.log_print()
    res = client.list_external_store(project,"");
    res.log_print();
    res = client.delete_external_store(project,"rds_store")
    res.log_print();
    # 使用python sdk进行查询分析
    req4 = GetLogsRequest(project, logstore, From, To, topic, "* |
select count(1)", 10,0, False)
    res4 = client.get_logs(req4)
```

```
    # 使用python sdk进行join rds查询
    req4 = GetLogsRequest(project, logstore, From, To, topic, "* |
select count(1) from "+logstore +"  l  join  rds_store  r on  l.ikey =
r.ekey", 10,0, False)
    res4 = client.get_logs(req4)
    # 使用python sdk把查询结果写入rds
    req4 = GetLogsRequest(project, logstore, From, To, topic, "* |
insert into rds_store select count(1) ", 10,0, False)
    res4 = client.get_logs(req4)
if __name__ == '__main__':
    main()
```

7 Android SDK

阿里云日志服务Android SDK主要解决Android平台用户数据采集问题，目前提供写入日志数据的功能。

通过**Github**

https://github.com/aliyun/aliyun-log-android-sdk

8 C SDK

阿里云日志服务C SDK主要解决各种平台的日志接入问题，比如兼容MIPS芯片、OpenWrt系统等。

C SDK 使用 cURL 作为网络库，*apr/apr-util* 库解决内存管理以及跨平台问题。您只需要基于源码进行简单编译便可以使用。

此外我们还有C版本的Producer Library和Producer Lite Library，为您提供跨平台、简洁、高性能、低资源消耗的一站式日志采集方案。

Github项目地址以及更多详细说明参见：

- *C Producer Library#*推荐服务端使用*#*
- *C Producer Lite Library#*推荐*IOT*、智能设备使用*#*
- *C* 原生*API#*推荐二次开发使用*#*

9 Go

阿里云日志服务Go SDK支持以下功能：

- 批量写入、消费数据
- 通过关键词查询索引
- 操作Logtail采集配置、机器组

GitHub地址：*https://github.com/aliyun/aliyun-log-go-sdk*

使用帮助与代码示例请参考项目README。

10 iOS SDK

阿里云日志服务 SDK 基于概览实现，目前提供写入日志的功能。

GitHub 地址：*https://github.com/aliyun/aliyun-log-ios-sdk*

Swift

```
 /*
    通过EndPoint、accessKeyID、accessKeySecret 构建日志服务客户端
    @endPoint: 服务访问入口，参见 https://help.aliyun.com/document_detail
/29008.html
 */
let myClient = try! LOGClient(endPoint: "",
                              accessKeyID: "",
                              accessKeySecret: "",
                              projectName:"")
/* 创建logGroup */
let logGroup = try! LogGroup(topic: "mTopic",source: "mSource")
   /* 存入一条log */
    let log1 = Log()
          try! log1.PutContent("K11", value: "V11")
         try! log1.PutContent("K12", value: "V12")
         try! log1.PutContent("K13", value: "V13")
    logGroup.PutLog(log1)
   /* 存入一条log */
    let log2 = Log()
          try! log2.PutContent("K21", value: "V21")
         try! log2.PutContent("K22", value: "V22")
         try! log2.PutContent("K23", value: "V23")
    logGroup.PutLog(log2)
 /* 发送 log */
 myClient.PostLog(logGroup,logStoreName: ""){ response, error in
        // handle response however you want
        if error?.domain == NSURLErrorDomain && error?.code ==
NSURLErrorTimedOut {
            print("timed out") // note, `response` is likely `nil` if
it timed out
        }
    }
```

Objective-C

参见 Github：*https://github.com/lujiajing1126/AliyunLogObjc*