Arenadata™ Streaming

Версия - v1.02

Руководство разработчика приложений для сервиса Kafka

2018 Arenadata™

Оглавление

1 Kafka Clients **3**
1.1 Java . . . 4
1.2 C/C++ . . . 5
1.3 JMS . . . 5
1.4 Python . . . 6
1.5 Go . . . 6
1.6 .NET . . . 7

В руководстве приведены сведения необходимые разработчику приложений для сервиса Kafka по работе с платформой ADS.

Руководство может быть полезно разработчикам, администраторам, программистам и сотрудникам подразделений информационных технологий, осуществляющих сопровождение платформы.

Important: Контактная информация службы поддержки – e-mail: info@arenadata.io

Глава 1

Kafka Clients

Клиенты для работы с Apache Kafka доступны на многих языках программирования (Clients).

В **ADS** предусмотрена возможность установки клиентских библиотек для различных языков, которые обеспечивают как низкоуровневый доступ к **Kafka**, так и потоковую обработку более высокого уровня.

Таблица1.1.: Клиентские библиотки для различных языков

Ссылка на установку	Ссылка на документацию
C/C++	github.com/edenhill/librdkafka
Go	github.com/confluentinc/confluent-kafka-go
Java	Kafka Java Consumer и Kafka Java Producer
JMS	JMS Client
.NET	github.com/confluentinc/confluent-kafka-dotnet
Python	github.com/confluentinc/confluent-kafka-python

В следующей таблице описана поддержка клиента по различным функциям платформы **ADS**.

Таблица1.2.: Поддержка клиента по различным функциям платформы

Функция	C/C++	Go	Java	.NET	Python
Admin API	Да	Да	Да	Нет	Да
Control Center metrics integration	Да	Да	Да	Да	Да
Custom partitioner	Да	Нет	Да	Нет	Нет
Exactly Once Semantics	Нет	Нет	Да	Нет	Нет
Idempotent Producer	Нет	Нет	Да	Нет	Нет
Kafka Streams	Нет	Нет	Да	Нет	Нет
Record Headers	Да	Да	Да	Да	Да
SASL Kerberos/GSSAPI	Да	Да	Да	Да	Да
SASL PLAIN	Да	Да	Да	Да	Да
SASL SCRAM	Да	Да	Да	Да	Да
Simplified installation	Да	Нет	Да	Да	Да
Schema Registry	Да	Нет	Да	Да	Да
Topic Metadata API	Да	Да	Да	Нет	Да

1.1 Java

Все JAR-файлы, включенные в пакеты, также доступны в репозитории. Далее приведен пример файла *POM*, показывающий, как добавить репозиторий:

```
<repositories>

  <repository>
    <id>confluent</id>
    <url>https://packages.confluent.io/maven/</url>
  </repository>

  <!-- further repository entries here -->

</repositories>
```

Репозиторий включает в себя скомпилированные версии **Kafka**. Чтобы сослаться на версию **Kafka 2.0** необходимо использовать в файле *pom.xml* следующее:

```
<dependencies>

  <dependency>
    <groupId>org.apache.kafka</groupId>
    <artifactId>kafka_2.11</artifactId>
    <version>2.0.1-cp1</version>
  </dependency>

  <!-- further dependency entries here -->

</dependencies>
```

Important: Arenadata всегда вносит исправления в проект с открытым исходным кодом Apache Kafka. Однако точные версии (и их имена), включаемые в платформу ADS, могут отличаться от артефактов Apache при не совпадении релизов. Если они отличаются, Arenadata сохраняет идентификаторы *groupId* и *artifactId*, но добавляет суффикс *-cpX* (где *X* – цифра) к идентификатору версии ADS для видимого отличия от артефактов Apache

Есть возможность ссылаться на артефакты для всех библиотек **Java**, которые включены в **ADS**. Например, чтобы использовать сериализаторы с открытым исходным кодом **Arenadata**, которые интегрируются с остальной частью платформы **ADS**, необходимо в *pom.xml* включить следующее:

```
<dependencies>

  <dependency>
    <groupId>io.confluent</groupId>
    <artifactId>kafka-avro-serializer</artifactId>
    <!-- For Confluent Platform 5.0.1 -->
    <version>5.0.1</version>
  </dependency>

  <!-- further dependency entries here -->

</dependencies>
```

1.2 C/C++

Клиент **C/C++**, называемый *librdkafka*, доступен в виде исходного кода и в виде предварительно скомпилированных двоичных файлов для дистрибутивов **Linux** на основе **Debian** и **Red Hat**, а также **macOS**. Большинство пользователей используют скомпилированные двоичные файлы.

Для дистрибутивов **Linux** необходимо следовать инструкциям для **Debian** или **Red Hat**, а затем использовать *yum* или *apt-get* для установки соответствующих пакетов. Например, разработчику, создающему приложение **C** на дистрибутиве **Red Hat**, рекомендуется использовать пакет *librdkafka-devel*:

```
sudo yum install librdkafka-devel
```

В дистрибутиве **Debian** используется пакет *librdkafka-dev*:

```
sudo apt-get install librdkafka-dev
```

В **macOS** последняя версия доступна через Homebrew:

```
brew install librdkafka
```

Исходный код доступен в архивах *ZIP* и *TAR* в каталоге *src/*.

1.3 JMS

Клиент **JMS** – это библиотека, используемая в приложениях **Java**. Чтобы сослаться на *kafka-jms-client* в проекте для начала необходимо добавить репозиторий в файл *pom.xml*:

```
<repositories>
    <repository>
        <id>confluent</id>
        <url>http://packages.confluent.io/maven/</url>
    </repository>
</repositories>
```

Затем добавить зависимость от клиента **JMS**, а также спецификацию API JMS (при этом заменив *[version]* на требуемую):

```
<dependency>
    <groupId>io.confluent</groupId>
    <artifactId>kafka-jms-client</artifactId>
    <version>[version]</version>
</dependency>
<dependency>
    <groupId>org.apache.geronimo.specs</groupId>
    <artifactId>geronimo-jms_1.1_spec</artifactId>
    <version>1.1</version>
</dependency>
```

Можно загрузить JAR-файл JMS-клиента напрямую, перейдя по следующему URL-адресу (при этом заменив *[version]* на требуемую):

```
http://packages.confluent.io/maven/io/confluent/kafka-jms-client/[version]/kafka-jms-client
-[version].jar
```

1.4 Python

Клиент **Python**, именуемый *confluent-kafka-python*, доступен в PyPI. Клиент **Python** использует *librdkafka* клиента **C**. Поэтому для установки **Python** сначала необходимо инсталлировать C, включая его пакет разработки, а затем установить библиотеку с помощью *pip* (как для **Linux**, так и для **macOS**):

```
pip install confluent-kafka
```

При этом осуществляется глобальная установка пакета для среды **Python**. Для инсталляции клиента только под конкретный проект можно использовать *virtualenv*.

После чего в **Python** можно импортировать библиотеку:

```
from confluent_kafka import Producer

conf = {'bootstrap.servers': 'localhost:9092', 'client.id': 'test', 'default.topic.config': {'acks
↪': 'all'}}
producer = Producer(conf)
producer.produce(topic, key='key', value='value')
```

Исходный код доступен в архивах *ZIP* и *TAR* в каталоге *src/*.

1.5 Go

Клиент **Go**, именуемый *confluent-kafka-go*, распространяется через GitHub и gopkg.in с привязкой к конкретным версиям. Клиент **Go** использует *librdkafka* клиента **C** и представляет его как библиотеку **Go**, используя cgo. Для установки клиента **Go** сначала необходимо инсталлировать клиент **C**, включая его пакет разработки, а также набор инструментов для сборки с *pkg-config*. В дистрибутивах **Linux** на основе **Red Hat** в дополнение к *librdkafka* следует установить следующие пакеты:

```
sudo yum groupinstall "Development Tools"
```

В дистрибутивах на основе **Debian**, помимо *librdkafka*, необходимо установить:

```
sudo apt-get install build-essential pkg-config git
```

В **macOS** с помощью Homebrew установить:

```
brew install pkg-config git
```

Далее использовать *go get* для установки библиотеки:

```
go get gopkg.in/confluentinc/confluent-kafka-go.v0/kafka
```

Код **Go** теперь может импортировать и использовать клиент. Также можно собрать и запустить небольшую утилиту командной строки **go-kafkacat**, чтобы убедиться, что установка прошла успешно:

```
go get gopkg.in/confluentinc/confluent-kafka-go.v0/examples/go-kafkacat
$GOPATH/bin/go-kafkacat --help
```

Для настройки статической ссылки к *librdkafka* необходимо добавть флаг *-tags static* к командам *go get*. Это позволяет статически связать саму *librdkafka*, чтобы ее динамическая библиотека не требовалась в целевой системе развертывания. Однако при этом статически связанные зависимости *librdkafka* (такие как *ssl*, *sasl2*, *lz4* и пр.), остаются по-прежнему динамически связанными и требуются в целевой системе. Экспериментальная опция для создания полностью статически связанного двоичного файла также доступна – использование флага *-tags static_all*. При этом требуется, чтобы все зависимости были доступны как статические библиотеки (например, *libsasl2.a*). Статические библиотеки обычно не устанавливаются по умолчанию, но доступны в соответствующих пакетах *-dev* или *-devel* (например, *libsasl2-dev*).

Исходный код доступен в архивах *ZIP* и *TAR* в каталоге *src/*.

1.6 .NET

Клиент **.NET**, именуемый *confluent-kafka-dotnet*, доступен в NuGet. Клиент **.NET** использует *librdkafka* клиента **C**. Предварительно скомпилированные двоичные файлы для *librdkafka* предоставляются через зависимый пакет **NuGet** librdkafka.redist для ряда популярных платформ (win-x64, win-x86, debian-x64, rhel-x64 и osx).

Для того, чтобы сослаться на *confluent-kafka-dotnet* из проекта, необходимо выполнить следующую команду в консоли диспетчера пакетов:

```
PM> Install-Package Confluent.Kafka
```

Important: Зависимый пакет *librdkafka.redist* устанавливается автоматически

Для того, чтобы сослаться на *confluent-kafka-dotnet* в файле *project.json*, необходимо включить следующую ссылку в раздел зависимостей:

```
"dependencies": {
    ...
    "Confluent.Kafka": "0.9.4"
    ...
}
```

И затем выполнить команду `dotnet restore`, чтобы восстановить зависимости проекта через **NuGet**.

Клиент *confluent-kafka-dotnet* предназначен для платформ **net451** и **netstandard1.3** и поддерживается в **.NET Framework** версии *4.5.1* и выше и **.NET Core** версии *1.0* (в **Windows**, **Linux** и **Mac**) и выше. Не поддерживается на **Mono**.