Kafka应用开发

www.huawei.com

HUAWEI

目标

- 学完本课程后，您将能够:
  - 了解**Kafka**应用开发适用场景
  - 熟悉**Kafka**应用开发流程
  - 熟悉并使用**Kafka**常用**API**
  - 进行**Kafka**应用开发

目录

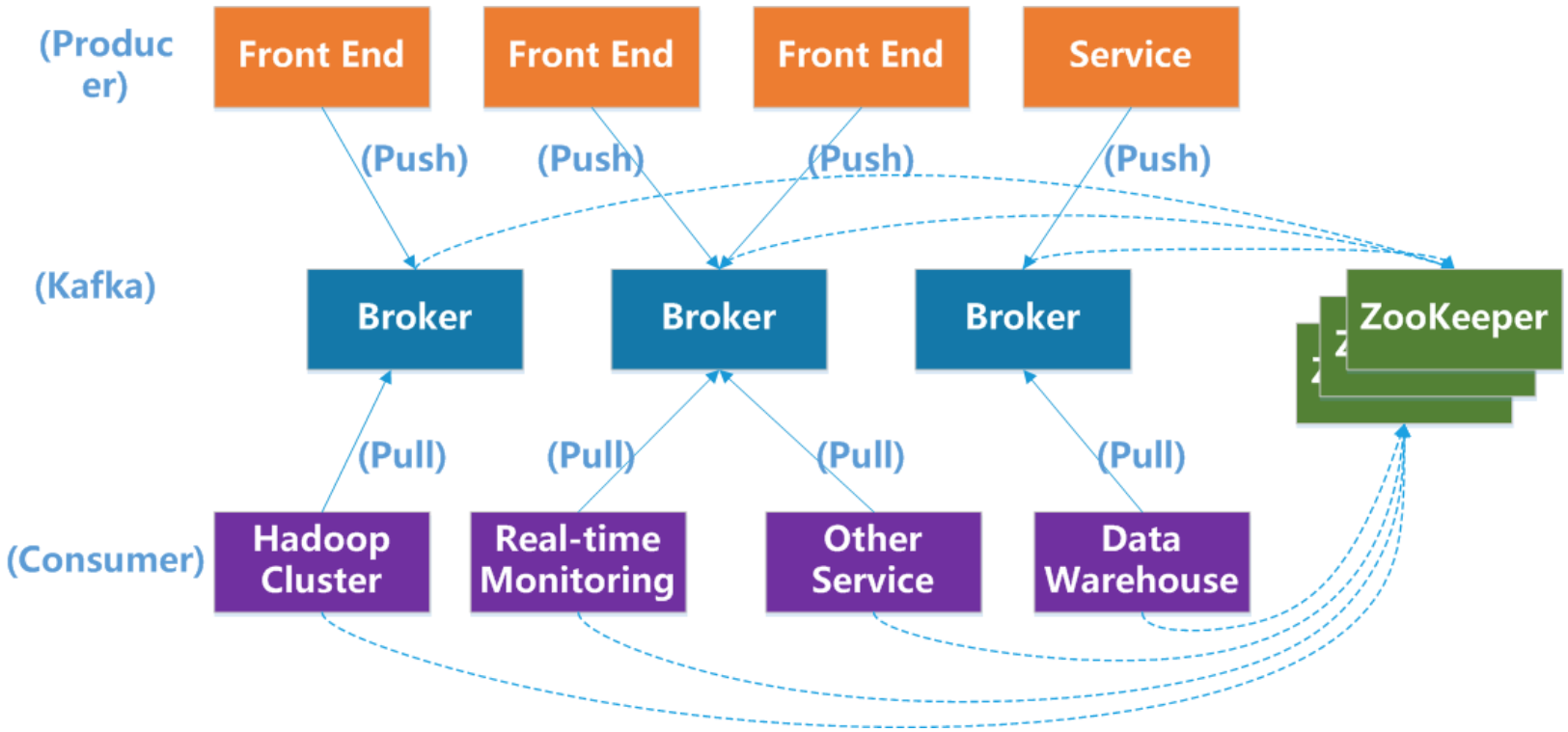
Kafka的定义

Kafka是一个高吞吐、分布式、基于发布订阅的消息系统。

Kafka有如下几个特点：

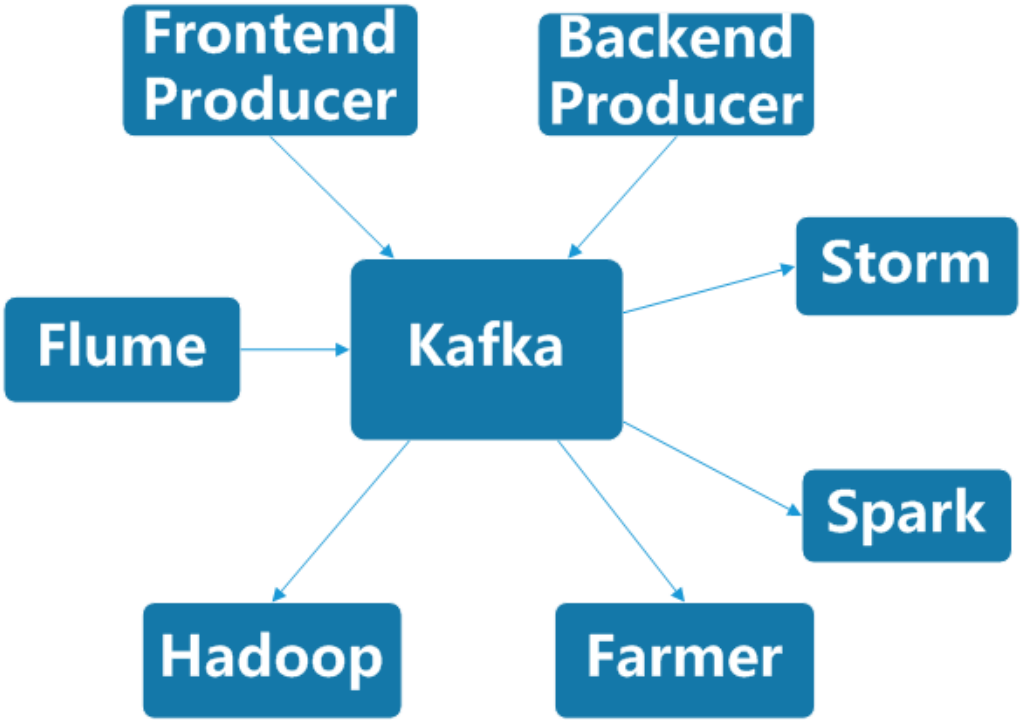
- 高吞吐量
- 消息持久化到磁盘
- 分布式系统易扩展
- 容错性好

Kafka架构回顾



Kafka的适用场景

- **Kafka**和其他组件比较，具有消息持久化、高吞吐、分布式、多客户端支持、实时等特性，适用于离线和在线的消息消费，如常规的消息收集、网站活性跟踪、聚合统计系统运营数据（监控数据）、日志收集等大量数据的互联网服务的数据收集场景。

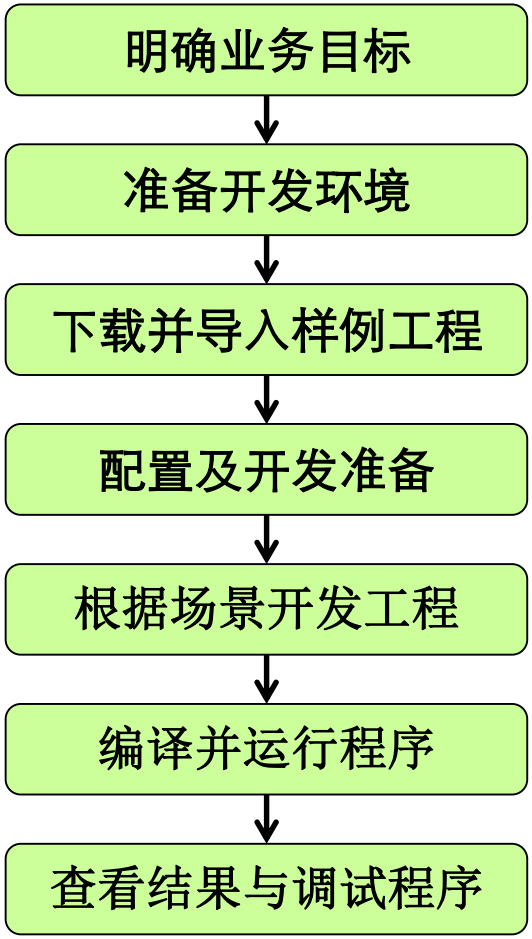


Kafka的适用场景

- 已对接组件
  - **Streaming、Spark、Flume**
- 使用了**Kafka**的好处
  - 解耦——使得消息生产、消费系统能够独立变更。
  - 可靠——有效解决单点故障引发系统不可用问题。
  - 易扩展——生产、消费系统扩展简单。
  - 可恢复——消息缓存下来，支持故障后从故障点读取。
  - 异步通信——生产系统无需关心消费系统的消费时间。

目录

Kafka应用开发流程



明确业务目标

- 预开发客户端属于哪种角色？**Producer**？**Consumer**？
- 目标**Topic**是否是已经在使用？正在使用？还是需要新创建？
- 如果需要创建**Topic**，那么该**Topic**和**Partition**的划分关系如何确定？
- 目标**Topic**是否为安全**Topic**？
- 预开发客户端性能指标？可靠性？实时性？

准备开发环境

准备项	说明
操作系统	**Windows**系统，推荐**Windows 7**以上版本。
安装**JDK**	开发环境的基本配置。版本要求：**1.7**或者**1.8**。
安装和配置**Eclipse**	用于开发**Kafka**应用程序的工具。
网络	确保客户端与**Kafka**服务主机的业务**IP**互通。

下载并导入Kafka样例工程

1、从**FusionInsight Manager**界面下载并解压**Kafka**客户端压缩包。

2、执行样例工程中**install.bat**文件（该批处理文件完成配置文件和运行依赖**jar**的拷贝）。

3、导入样例工程到**Eclipse**开发环境。

配置及开发准备

1、在FusionInsight Manager页面新建机机用户，并加入相应权限的Kafka用户组（具体权限介绍参考下一页）。

2、下载用户的keytab文件。

3、配置keytab文件到Kafka客户端样例工程（具体参考下下页）。

4、向管理员申请Topic访问权限，如果Topic尚未创建，需要联系Kafka管理员用户先创建该Topic。

配置及开发准备—Kafka用户组权限介绍

Kafka用户组权限介绍

- **kafkaadmin组：**

  Kafka管理员用户组。添加入本组的用户，拥有所有**Topic**的创建、删除、授权及读写权限。

- **kafkasuperuser组**

  Kafka超级用户组。添加入本组的用户，拥有所有**Topic**的读写权限。

- **kafka组**

  Kafka普通用户组。添加入本组的用户，需要被**kafkaadmin**组用户授予特定**Topic**的读写权限，才能访问对应**Topic**。

配置及开发准备—配置keytab文件到样例工程

配置Keytab到工程

```
public final static String topic
 // Topic名称，安全模式下，需要以管理员用户添加当前用户的访问权限
 = "example-metric1";
private static final String USER_KEYTAB_FILE = "用户自己申请的机机账号keytab文件名称";
private static final String USER_PRINCIPAL = "用户自己申请的机机账号名称";
```

安全认证

```
// 调用认证接口
LoginUtil.setKrb5Config(krbFile);
LoginUtil.setZookeeperServerPrincipal("zookeeper/hadoop.hadoop.com");
LoginUtil.setJaasFile(USER_PRINCIPAL, userKeyTableFile);
```

Topic订阅

```
// 订阅 consumer.subscribe(Collections.singletonList(this.topic));
```

配置及开发准备—配置keytab文件到样例工程

根据场景开发工程

1. 梳理业务场景流程
2. 设计各模块接口
3. 如果使用的是安全集群，需要进行安全认证
4. 熟悉**Kafka**提供的相应**API**
5. 调用业务需要的**API**实现各功能

编译并运行程序

方式一：在开发环境**Eclipse**中，右击配置的**Producer**或者**Consumer**对应的样例代码，单击“**Run as > Java Application**”运行对应的应用程序工程。

方式二：导出**jar**包到**Linux**下运行，具体请参考产品文档的《应用开发指南》中**Kafka**相关章节。

查看结果与调试程序

- 如果是生产数据，那么可以通过控制台打印信息，及**Consumer**消费的方式来查看发送结果。

  Consumer消费命令参考如下：

  bin/kafka-console-consumer.sh --topic *<Topic名称>* --bootstrap-server *<Kafka集群IP:21007>* --new-consumer --consumer.config config/consumer.properties

- 如果是消费数据，可以通过控制台打印信息查看结果。
- 也可以同时验证生产者和消费者，需要配置同一个**Topic**，优先启动消费者，然后启动生产者，查看两边信息是否能够对应。

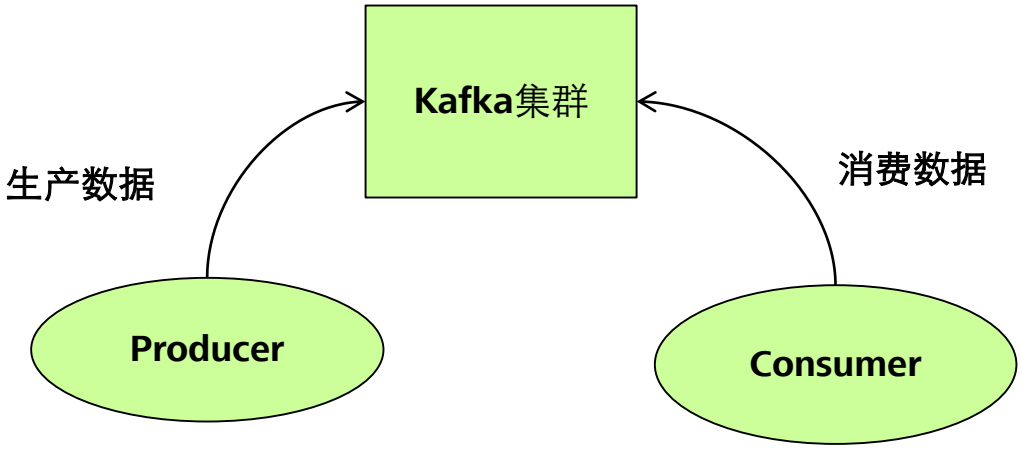
  Producer的生产命令参考如下：

  bin/kafka-console-producer.sh --broker-list *<Kafka集群IP:21007>* --topic *<Topic名称>* --producer.config config/producer.properties

目录

应用开发案例分析

- 预开发一个**Producer**客户端和一个**Consumer**客户端。
  - **Topic**要求：需要增加权限控制，支持每天输入数据量**100GB**，保存**7**天。
  - **Producer**要求：数据需要按顺序发送，上条数据发送完成后，再发送下一条。
  - **Consumer**要求：消费该**Producer**生产的数据，支持离线消费和在线消费，不重复消费。



应用开发案例分析

- 应用需求解析：
  - **Topic**要求增加权限控制。

    在当前**FusionInsight HD V100R002C60SPC200**版本中，**Kafka**的内核版本是**0.9.0.0**，该版本已经支持安全，只要向管理员申请**ACL**赋权限，那么该**Topic**就已经是安全**Topic**了，其他未授权的普通用户将无法访问。
  - **Topic**要求支持每天输入数据量**100GB**，保存**7**天。

    创建**Topic**时需要需要结合当前配置的磁盘情况，来规划**Partition**的个数和副本个数，根据要求磁盘需要满足（**700 *** 副本数）**GB**的存储容量，具体的**Partition**划分模型可参考产品文档中《**Kafka业务规格说明**》章节。

应用开发案例分析

- 应用需求解析：

Producer要求数据需要按顺序发送，上条数据发送完成后，再发送下一条。

调用**Producer**同步发送接口“**producer.send(record).get()**”，以保证数据按顺序发送。

例如：在样例工程的**com.huawei.bigdata.kafka.example.NewProducer**类中**run**方法中当**isAsync**变量为**false**时的代码。

应用开发案例分析

NewProducer.java

```
91      /**
92       * 生产者线程执行函数，循环发送消息。
93       */
94      public void run()
95      {
96          LOG.info("New Producer: start.");
97          int messageNo = 1;
98          // 指定发送多少条消息后sleep1秒
99          int intervalMessages=10;
100
101         while (messageNo <= messageNumToSend)
102         {
103             String messageStr = "Message_" + messageNo;
104             long startTime = System.currentTimeMillis();
105
106             // 构造消息记录
107             ProducerRecord<Integer, String> record = new ProducerRecord<Integer, String>(topic, messageNo, messageStr);
108
109             if (isAsync)
110             {
111                 // 异步发送
112                 producer.send(record, new DemoCallBack(startTime, messageNo, messageStr));
113             }
114             else
115             {
116                 try
117                 {
118                     // 同步发送
119                     producer.send(record).get();
120                 }
121                 catch (InterruptedException ie)
122                 {
123                     LOG.info("The InterruptedException occured : {}.", ie);
124                 }
125                 catch (ExecutionException ee)
126                 {
127                     LOG.info("The ExecutionException occured : {}.", ee);
128                 }
129             }
130             messageNo++;
131
132             if (messageNo % intervalMessages == 0)
133             {
134                 // 每发送intervalMessage条消息sleep1秒
```

应用开发案例分析

- 应用需求解析：
  - **Consumer**要求消费该**Producer**生产的数据，支持离线消费和在线消费，不重复消费。
  - 首先需要配置与**Producer**相同的**Topic**，然后需要配置一个固定的**group.id**，例如在**com.huawei.bigdata.kafka.example.NewConsumer**类中的构造方法中**props.put(groupId, “DemoConsumer”)**。
  - 及时提交消费的**offset**来记录当前消费位置，以便停掉**Consumer**或者是起多个**Consumer**时，能够继续从此位置进行消费数据，以保证离线数据消费和在线数据消费的均不会出现重复消费。默认情况下可以自动提交，并可设置自动提交**offset**的时间间隔。

应用开发案例分析

NewConsumer.java

```
63      */
64     public NewConsumer(String topic)
65     {
66         super("KafkaConsumerExample", false);
67         Properties props = new Properties();
68
69         KafkaProperties kafkaProc = KafkaProperties.getInstance();
70         // Broker连接地址
71         props.put(bootstrapServers,
72             kafkaProc.getValues(bootstrapServers, "localhost:21007"));
73         // Group id
74         props.put(groupId, "DemoConsumer");
75         // 是否自动提交offset
76         props.put(enableAutoCommit, "true");
77         // 自动提交offset的时间间隔
78         props.put(autoCommitIntervalMs, "1000");
79         // 会话超时时间
80         props.put(sessionTimeoutMs, "30000");
81         // 消息Key值使用的反序列化类
82         props.put(keyDeserializer,
83             "org.apache.kafka.common.serialization.IntegerDeserializer");
84         // 消息内容使用的反序列化类
85         props.put(valueDeserializer,
86             "org.apache.kafka.common.serialization.StringDeserializer");
87         // 安全协议类型
88         props.put(securityProtocol, kafkaProc.getValues(securityProtocol, "SASL_PLAINTEXT"));
89         // 服务名
90         props.put(saslKerberosServiceName, "kafka");
91         consumer = new KafkaConsumer<Integer, String>(props);
92         this.topic = topic;
93     }
94
95     /**
96      * 订阅Topic的消息处理函数
97      */
98     public void doWork()
99     {
100        // 订阅
101        consumer.subscribe(Collections.singletonList(this.topic));
102        // 消息消费请求
103        ConsumerRecords<Integer, String> records = consumer.poll(waitTime);
104        // 消息处理
105        for (ConsumerRecord<Integer, String> record : records)
106        {
```

目录

Kafka常用接口示例

- 新**Producer**重要参数：

参数	描述	备注
bootstrap.servers	**Broker**地址列表	生产者通过此参数值，创建与**Broker**之间的连接。
security.protocol	安全协议类型	生产者使用的安全协议类型，当前安全模式下仅支持**SASL**协议，需要配置为**SASL_PLAINTEXT**。
sasl.kerberos.service.name	服务名	**Kafka**集群运行，所使用的**Kerberos**用户名（需配置为**kafka**）。
key.serializer	消息**Key**值序列化类	指定消息**Key**值序列化方式。
value.serializer	消息序列化类	指定所发送消息的序列化方式。

Kafka常用接口示例

- 新**Producer**重要接口函数：

返回值类型	接口函数	描述
java.util.concurrent.Future<RecordMetadata>	**send(ProducerRecord<K,V> record)**	不带回调函数的发送接口，通常使用**Future**的**get()**函数阻塞发送，实现同步发送。
java.util.concurrent.Future<RecordMetadata>	**send(ProducerRecord<K,V> record, Callback callback)**	带回调函数的发送接口，通常用于异步发送后，通过回调函数实现对发送结果的处理。
void	**onCompletion(RecordMetadata metadata, Exception exception)**	回调函数接口方法，通过实现**Callback**中的此方法来进行异步发送结果的处理。

Kafka常用接口示例

- 新**Consumer**重要参数：

参数	描述	备注
bootstrap.servers	**Broker**地址列表	消费者通过此参数值，创建与**Broker**之间的连接。
security.protocol	安全协议类型	消费者使用的安全协议类型，当前安全模式下仅支持**SASL**协议，需要配置为**SASL_PLAINTEXT**。
sasl.kerberos.service.name	服务名	**Kafka**集群运行，所使用的**Kerberos**用户名（需配置为**kafka**）。
key.deserializer	消息**Key**值反序列化类	反序列化消息**Key**值。
value.deserializer	消息反序列化类	反序列化所接收的消息。

Kafka常用接口示例

- 新**Consumer**重要接口函数：

返回值类型	接口函数	描述
void	**close()**	关闭**Consumer**接口方法。
void	**subscribe(java.util.List<java.lang.String> topics)**	**Topic**订阅接口方法。
ConsumerRecords<K,V>	**poll(long timeout)**	请求获取消息接口方法。

Kafka常用接口示例

- 关于安全端口和非安全端口说明：
  - **Kafka**集群安全访问端口默认为**21007**，非安全访问端口默认为**21005**；
  - 旧**API**仅支持访问**21005**端口；新**API**兼容访问非安全端口**21005**和安全端口**21007**。
  - 服务端参数**allow.everyone.if.no.acl.found**设置为**true**，则允许旧**API**通过**21005**端口访问未设置**ACL**的**Topic**。
  - 非**kafka/kafkaadmin/kafkasuperuser**组的用户访问安全**Kafka**接口，鉴权不通过（除系统管理员组用户）。
  - 访问安全**Topic**，必须要使用新**API**。

Kafka Producer接口(新API使用示例)

- 向管理员申请目标**Topic**的生产者权限。
- 根据业务需求，配置发送相关配置参数。
- 调用新**Producer API**接口发送数据。

示例：

```
// Broker连接地址
props.put(bootstrapServers,kafkaProc.getValues(bootstrapServers,
"localhost:21007"));
// 消息Key值使用的反序列化类
props.put(keyDeserializer,

"org.apache.kafka.common.serialization.IntegerDeserializer");
// 消息内容使用的反序列化类
props.put(valueDeserializer,

"org.apache.kafka.common.serialization.StringDeserializer");
// 安全协议类型
props.put(securityProtocol,kafkaProc.getValues(securityProtocol,
"SASL_PLAINTEXT"));
// 服务名
props.put(saslKerberosServiceName, "kafka");
 // 构造消息记录
ProducerRecord<Integer, String> record = new ProducerRecord<Integer,
String>(topic, messageNo, messageStr);
// 异步发送
producer.send(record, new DemoCallBack(startTime, messageNo,
messageStr));
// 同步发送
producer.send(record).get();
```

Kafka Producer接口(旧API使用示例)

- 旧**Producer API**仅支持通过非安全端口访问未设置**ACL**的**Topic**，需要将服务端配置项**allow.everyone.if.no.acl.found**配置为**true**。
- 根据业务需求，配置发送相关配置参数。
- 调用旧**Producer API**接口发送数据。

示例：

```
// 配置同步发送模式
props.put(producerType, "sync");
// 配置SimplePartitioner为解析key值，返回PartitionId.
props.put(partitionerClass,
"com.huawei.bigdata.kafka.example.SimplePartitioner");
// 序列化类
props.put(serializerClass,
"kafka.serializer.StringEncoder");
// Broker列表
props.put(metadataBrokerList,
KafkaProperties.getInstance().getValues(bootstrapServers, "localhost:9092"));
// 创建生产者对象
producer = new
kafka.javaapi.producer.Producer<String,
String>(new ProducerConfig(props));
// 指定消息序号作为key值
String key = String.valueOf(messageNo);
producer.send(new KeyedMessage<String,
String>(topic, key, messageStr));
```

Kafka Consumer接口(新API使用样例)

示例：

```
// Broker连接地址
props.put(bootstrapServers,kafkaProc.getValues(bootstrapServers,
"localhost:21007"));
// Group id
props.put(groupId, "DemoConsumer");
// 消息Key值使用的反序列化类
props.put(keyDeserializer,
"org.apache.kafka.common.serialization.IntegerDeserializer");
// 消息内容使用的反序列化类
props.put(valueDeserializer,
"org.apache.kafka.common.serialization.StringDeserializer");
// 安全协议类型
props.put(securityProtocol, kafkaProc.getValues(securityProtocol,
"SASL_PLAINTEXT"));
// 服务名
props.put(saslKerberosServiceName, "kafka");
// 构造消费者对象
consumer = new KafkaConsumer<Integer, String>(props);
//订阅
consumer.subscribe(Collections.singletonList(this.topic));
//消息消费请求
ConsumerRecords<Integer, String> records = consumer.poll(waitTime);
```

- 向管理员申请目标**Topic**的消费者权限。
- 根据业务需求，配置消费者相关配置参数。
- 调用新**Consumer API**接口进行消息消费。

Kafka Consumer接口(旧API使用样例)

- 旧**Consumer API**仅支持访问未设置**ACL**的**Topic**，需要将服务端配置项**allow.everyone.if.no.acl.found**配置为**true**。
- 根据业务需求，配置消费者相关配置参数。
- 调用旧**Consumer API**接口进行消息消费。

示例：

```
// 配置消费者组ID
props.put("group.id",
kafkaPros.getValues("group.id", "example-
group1"));
//配置ZooKeeper相关参数
props.put("zookeeper.connect",
kafkaPros.getValues("zookeeper.connect",
"localhost:2181"));
//配置offset提交间隔参数
props.put("auto.commit.interval.ms",
kafkaPros.getValues("auto.commit.interval.ms",
"10000"));
// 消费接口调用
Map<String, List<KafkaStream<byte[], byte[]>>>
consumerMap =
consumer.createMessageStreams(topicCountMap);

List<KafkaStream<byte[], byte[]>> streams =
consumerMap.get(topic);
```

总结

- **Kafka**应用开发适用场景
- 应用开发流程
- 权限控制
- 常用**API**
- 总结与思考

思考题

1. Kafka相对其他消息队列的优势是什么?
2. Kafka是如何保障数据可靠的?

习题

- 单选题

1. 安全**Kafka**集群中，关于**Kafka**组的说法错误的是？（）

A kafkaadmin组用户具有**Topic**的所有权限。

B kafka组用户被授权**Topic**相关权限后，只有使用新**API**才能访问。

C kafkasuper组用户默认具有所有**Topic**的读写权限。

D kafka组用户被授权**Topic**相关权限后，一定可以访问成功。

习题

- 多选题

1. 下面哪些关键词是**Kafka**的特点？( )

A 高吞吐

B 分布式

C 消息持久化

D 支持消息随机读取

Thank you

www.huawei.com