大厂学苑-大数据&人工智能

Kafka

版本：V1.0

More than 80% of all Fortune 100 companies trust, and use Kafka.

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

第1章 Kafka 安装

1.1 下载软件

从官网下载最新版本软件：http://kafka.apache.org/downloads。截至到 2021 年 5 月 1 日，最新版本为：kafka_2.12-2.8.0.tgz，因为 kafka 是使用 scala 语言开发的，所以 2.12 为 scala 语言版本，2.8.0 为 kafka 最新软件版本

1.2 安装软件

1.2.1 单机模式

1.2.1.1 解压缩

将下载后的软件上传到虚拟机中，并将软件解压缩到无中文，无空格的路径中

```
[root@linux1 software]# tar -zxvf kafka_2.12-2.8.0.tgz -C /opt/module
```

JDK 必须安装 1.8 或以上版本

1.2.1.2 启动 zookeeper 服务

Kafka 计划不久的将来会将对 zookeeper 的依赖从 kafka 中移除，从 2.8.0 版本中已经可以进行无 zookeeper 测试了，但是还不能应用于生产环境中，所以依然需要连接 zookeeper 服务

```
[root@linux1 kafka]# bin/zookeeper-server-start.sh config/zookeeper.properties
```

1.2.1.3 启动 kafka 服务

```
[root@linux1 kafka]# bin/kafka-server-start.sh -daemon config/server.properties
```

成功启动所有服务后，kafka 环境就准备好了

1.2.2 集群模式

1.2.2.1 解压缩

将下载后的软件上传到虚拟机中，并将软件解压缩到无中文，无空格的路径中

```
[root@linux1 software]# tar -zxvf kafka_2.12-2.8.0.tgz -C /opt/module
```

JDK 必须安装 1.8 或以上版本

在解压缩后的目录中创建 logs 目录

```
[root@linux1 kafka]# mkdir logs
```

1.2.2.2 修改配置文件

- server.properties

```
#broker 的全局唯一节点编号，不能重复
```

```
broker.id=0
#删除 topic 功能使能
delete.topic.enable=true
#处理网络请求的线程数量
num.network.threads=3
#用来处理磁盘 IO 的现成数量
num.io.threads=8
#发送套接字的缓冲区大小
socket.send.buffer.bytes=102400
#接收套接字的缓冲区大小
socket.receive.buffer.bytes=102400
#请求套接字的缓冲区大小
socket.request.max.bytes=104857600
#kafka 运行日志存放的路径
log.dirs=/opt/module/kafka/logs
#topic 在当前 broker 上的分区个数
num.partitions=1
#用来恢复和清理 data 下数据的线程数量
num.recovery.threads.per.data.dir=1
#segment 文件保留的最长时间，超时将被删除
log.retention.hours=168
#配置连接 Zookeeper 集群地址
zookeeper.connect=linux1:2181,linux2:2181,linux3:2181/kafka-2.8
```

1.2.2.3 分发安装包

```
[root@linux1 module]$ xsync kafka/
```

注意：分发之后记得配置其他机器的环境变量

7）分别在 linux2 和 linux3 上修改配置文件 /opt/module/kafka/config/server.properties 中的 broker.id=1、broker.id=2

注意：broker.id 不得重复

1.2.2.4 启动 kafka 服务

```
依次在 linux1、linux2、linux3 节点上启动 kafka
[root@linux1 kafka]$ bin/kafka-server-start.sh -daemon
config/server.properties
[root@linux2 kafka]$ bin/kafka-server-start.sh -daemon config/server.properties
[root@linux3 kafka]$ bin/kafka-server-start.sh -daemon config/server.properties
```

1.2.2.4 关闭 kafka 服务

```
依次在 linux1、linux2、linux3 节点上关闭 kafka
[root@linux1 kafka]$ bin/kafka-server-stop.sh
[root@linux2 kafka]$ bin/kafka-server-stop.sh
[root@linux3 kafka]$ bin/kafka-server-stop.sh
```

1.3 安装测试

kafka 环境准备好，可以通过命令行工具发送指令进行简单测试

```
# 查看某一个主题
bin/kafka-topics.sh --bootstrap-server linux1:9092 --describe --topic test
# 查看主题列表
bin/kafka-topics.sh --bootstrap-server linux1:9092 --list
```

9092 为 kafka 启动服务后的默认通信端口

第2章 Kafka 操作

2.1 命令行操作

2.1.1 查看主题

```
[root@linux1 kafka]# bin/kafka-topics.sh --bootstrap-server linux1:9092 --list
```

2.1.2 创建主题

```
[root@linux1 kafka]# bin/kafka-topics.sh --bootstrap-server linux1:9092
--create --topic event-topic --partitions 3 --replication-factor 2
```

选项说明：

--topic 定义 topic 名

--replication-factor 定义副本数（副本数量不能大于服务器节点数量）

--partitions 定义分区数

2.1.3 删除主题

```
[root@linux1 kafka]# bin/kafka-topics.sh --bootstrap-server linux1:9092
--delete --topic event-topic
```

需要 server.properties 中设置 delete.topic.enable=true 否则只是标记删除

2.1.4 生产事件（消息）

```
[root@linux1 kafka]# bin/kafka-console-producer.sh --broker-list linux1:9092
--topic event-topic
```

2.1.5 消费事件（消息）

```
[root@linux1 kafka]# bin/kafka-console-consumer.sh --bootstrap-server
linux1:9092 --from-beginning --topic event-topic

[root@linux1 kafka]# bin/kafka-console-consumer.sh --bootstrap-server
linux1:9092 --topic event-topic
```

选项说明：

--from-beginning 从头消费

2.1.6 查看主题

```
[root@linux1 kafka]# bin/kafka-topics.sh --bootstrap-server linux1:9092
--describe --topic event-topic
```

2.2 Java API

2.2.1 POM 依赖

```
<dependencies>
    <dependency>
        <groupId>org.apache.kafka</groupId>
        <artifactId>kafka_2.12</artifactId>
        <version>2.8.0</version>
        </dependency>
    <dependency>
        <groupId>org.apache.kafka</groupId>
        <artifactId>kafka-clients</artifactId>
        <version>2.8.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-slf4j-impl</artifactId>
        <version>2.12.0</version>
    </dependency>
</dependencies>
```

2.2.2 日志配置文件

在 Resources 目录新建 log4j2.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="error" strict="true" name="XMLConfig">
    <Appenders>
        <!-- 类型名为 Console，名称为必须属性 -->
        <Appender type="Console" name="STDOUT">
            <!-- 布局为 PatternLayout 的方式，
            输出样式为[INFO] [2018-01-22 17:34:01][org.test.Console]I'm here -->
            <Layout type="PatternLayout"
                    pattern="[%p] [%d{yyyy-MM-dd HH:mm:ss}][%c{10}]%m%n" />
        </Appender>

    </Appenders>

    <Loggers>
        <!-- 可加性为 false -->
        <Logger name="test" level="info" additivity="false">
            <AppenderRef ref="STDOUT" />
        </Logger>

        <!-- root loggerConfig 设置 -->
        <Root level="info">
            <AppenderRef ref="STDOUT" />
        </Root>
```

```
    </Loggers>

</Configuration>
```

2.2.3 Java API

- 生产数据

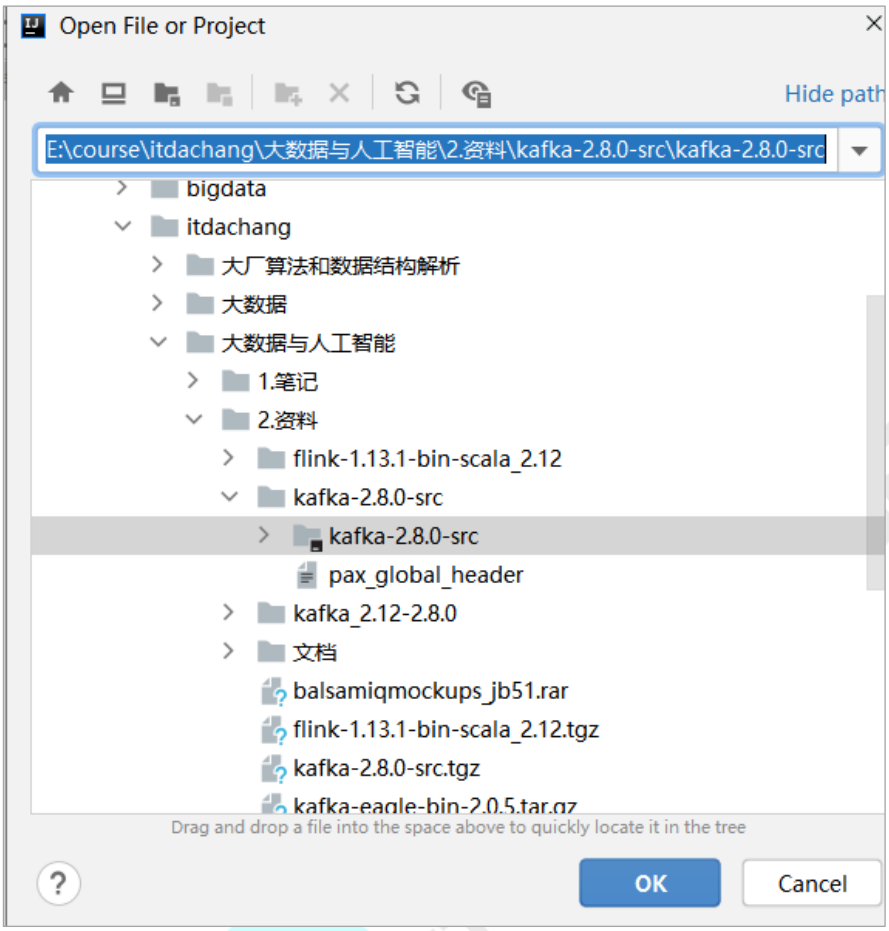
```
Properties props = new Properties();
props.put("bootstrap.servers", "linux1:9092");
props.put("acks", "all");// ACK 应答
props.put("retries", 1);//重试次数
props.put("batch.size", 16384);//批次大小
props.put("linger.ms", 1);//等待时间
props.put("buffer.memory", 33554432);//RecordAccumulator 缓冲区大小
props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

Producer<String, String> producer = new KafkaProducer<>(props);
for (int i = 0; i < 100; i++) {
   producer.send(new ProducerRecord<String, String>("event-topic",
Integer.toString(i), Integer.toString(i)));
}
producer.close();
```

- 消费数据

```
Properties props = new Properties();
props.put("bootstrap.servers", "linux1:9092");
props.put("group.id", "test");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
consumer.subscribe(Arrays.asList("event-topic"));
while (true) {
   ConsumerRecords<String, String> records = consumer.poll(1000);
   System.out.println("cnt = " + records.count());
   for (ConsumerRecord<String, String> record : records)
      System.out.printf("offset = %d, key = %s, value = %s%n", record.offset(),
record.key(), record.value());
}
```

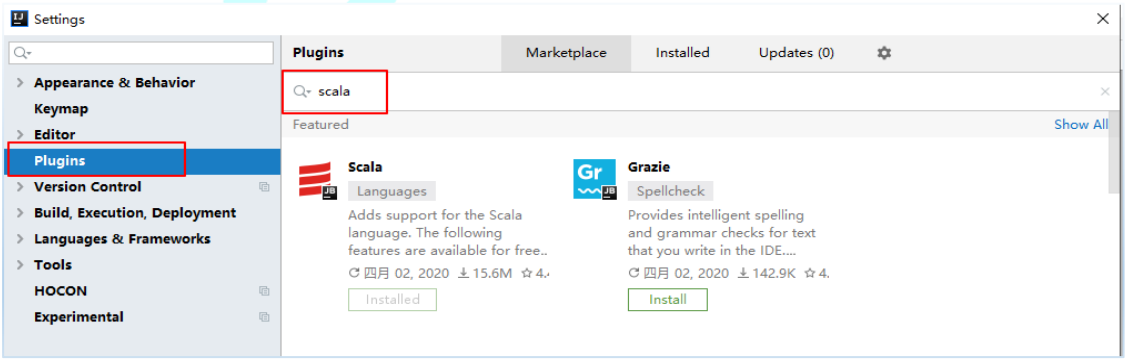
第3章　Kafka 源码

将下载后的源码压缩包文件 kafka-2.8.0-src.tgz 解压缩，然后使用 IDEA 工具直接导入即可。



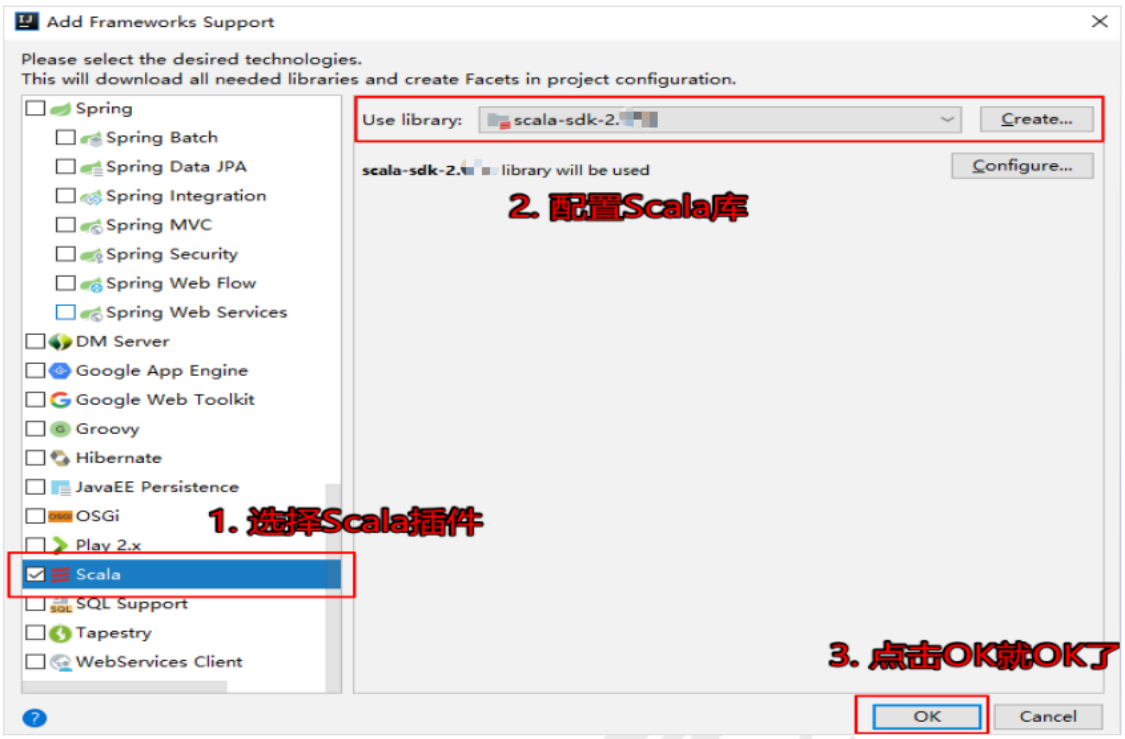
因为 Kafka 的源码是基于 Scala 语言开发的，所以还需要在 IDEA 中配置 Scala 环境.

默认情况下 IDEA 不支持 Scala 的开发，需要安装 Scala 插件。



如果下载慢的，请访问网址：https://plugins.jetbrains.com/plugin/1347-scala/versions

默认情况，IDEA 中创建项目时不支持 Scala 的开发，需要添加 Scala 框架的支持。


Add Frameworks Support
Please select the desired technologies.
This will download all needed libraries and create Facets in project configuration.
Spring
Spring Batch
Spring Data JPA
Spring Integration
Spring MVC
Spring Security
Spring Web Flow
Spring Web Services
DM Server
Google App Engine
Google Web Toolkit
Groovy
Hibernate
JavaEE Persistence
OSGi
Play 2.x
Scala
SQL Support
Tapestry
WebServices Client
Use library:
scala-sdk-2.
Create...
Configure...
library will be used
2. 配置Scala库
1. 选择Scala插件
3. 点击OK就OK了
OK
Cancel