# 环境要求

1. 系统环境

Linux redhat6.5

1. Kafka版本

Kafka-0.10.2

1. JDK版本

JDK1.7+

# 本次验证集群环境信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Node86** | **Node99** | **Node101** | **Node181** | **Node183** |
| **NameNode** |  | **√** |  |  |  |
| **DataNode** | **√** | **√** | **√** |  | **√** |
| **Kafka** |  |  |  |  | **√** |
| **Sentry** |  |  | **√** |  |  |
| **Zookeeper** | **√** | **√** |  |  | **√** |
| **KDC Server** |  |  |  | **√** |  |

# Kafka ACL权限认证

1. server.properties文件配置

kafka目录config下面的server.properties文件增加如下内容：

|  |
| --- |
| authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer  allow.everyone.if.no.acl.found=true  principal.builder.class=org.apache.kafka.common.security.auth.DefaultPrincipalBuilder |

配置好后重启kafka

1. IP认证权限命令

配置主题ip访问权限示例

配置主题为foo的写权限ip为192.168.80.56

./kafka-acls.sh --authorizer-properties zookeeper.connect=192.168.80.56:2181 --operation Write --operation Read --allow-principal User:\* --allow-host 192.168.80.56 --add --topic foo

配置主题为foo的读权限ip为\*，但是192.168.80.27除外

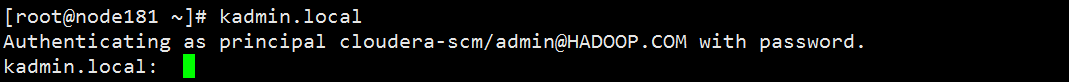
./kafka-acls.sh --authorizer-properties zookeeper.connect=192.168.80.56:2181 --add --allow-principal User:\* --allow-host '\*' --deny-principal User:\* --deny-host 192.168.80.27 --operation Read --topic foo

# Kafka Kerberos + Sentry权限认证

1. 服务器端生成Kerberos凭证（需要root权限）
2. 在KDC Server （即Node181）中执行命令：

kadmin.local

如图所示：



1. 在kadmin.local命令行中执行命令：

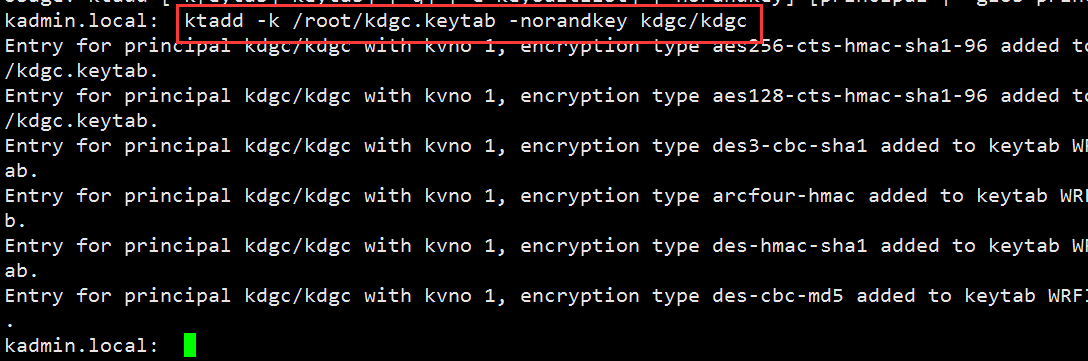
add\_principal kdgc/kdgc

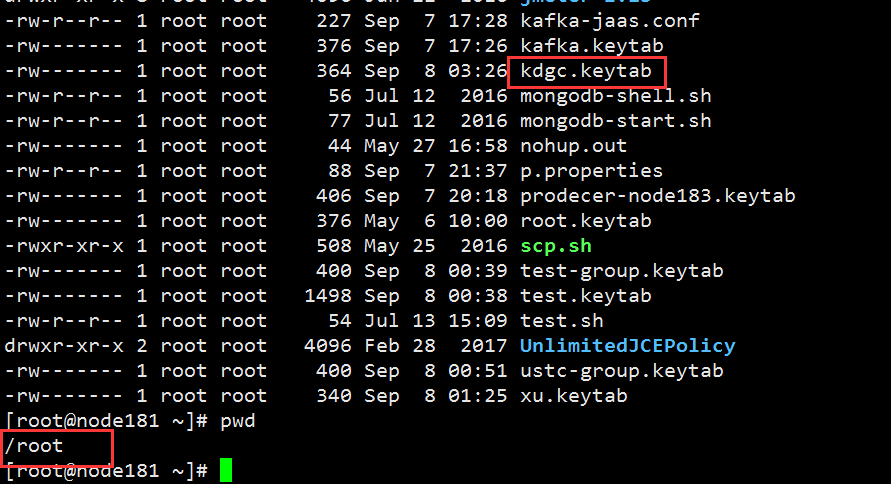
如图所示，需要输入两次密码，该命令表示创建了一个凭证： [kdgc/kdgc@HADOOP.COM](mailto:kdgc/kdgc@HADOOP.COM)，且它的密码是由你自己指定的，此处的kdgc/kdgc可以随意指定，比如指定为abc/xyz等等。



1. 将凭证[kdgc/kdgc@HADOOP.COM](mailto:kdgc/kdgc@HADOOP.COM) 导出为keytab文件，导出路径为/root/kdgc.keytab,命令如下：

ktadd -k /root/kdgc.keytab -norandkey kdgc/kdgc



如图所示，在/root目录下生成了kdgc.keytab文件 

1. Kafka客户端Kerberos认证配置
2. Kafka客户端需要安装Kerberos
3. 从KDC服务器端即node181获取Kerberos票据
4. keytab票据文件，该文件需要KDC 服务器生成;
5. krb5.conf配置文件，需要从KDC服务器拷贝，默认位置为/etc/krb5.conf，文件内容如下，其中关键配置参数都已标红

[logging]

default = FILE:/var/log/krb5libs.log

kdc = FILE:/var/log/krb5kdc.log

admin\_server = FILE:/var/log/kadmind.log

[libdefaults]

default\_realm = HADOOP.COM //Kerberos 域名

dns\_lookup\_realm = false

dns\_lookup\_kdc = false

ticket\_lifetime = 7d

renew\_lifetime = 7d

forwardable = true

[realms]

HADOOP.COM = {

kdc = node181:88 //KDC服务器IP地址及端口号

admin\_server = node181:749 //管理服务的IP地址及端口号

}

[domain\_realm]

.hadoop.com = HADOOP.COM

hadoop.com = HADOOP.COM

1. 创建JAAS配置文件kafka-jaas.conf

配置文件内容如下，其中[kdgc/kdgc@HADOOP.COM](mailto:kdgc/kdgc@HADOOP.COM)为Kerberos凭证，kdgc.keytab为票据，

KafkaClient {

com.sun.security.auth.module.Krb5LoginModule required

doNotPrompt=true

useTicketCache=true

principal="kdgc/kdgc@HADOOP.COM"

useKeyTab=true

serviceName="kafka"

keyTab="/root/kdgc.keytab"

client=true;

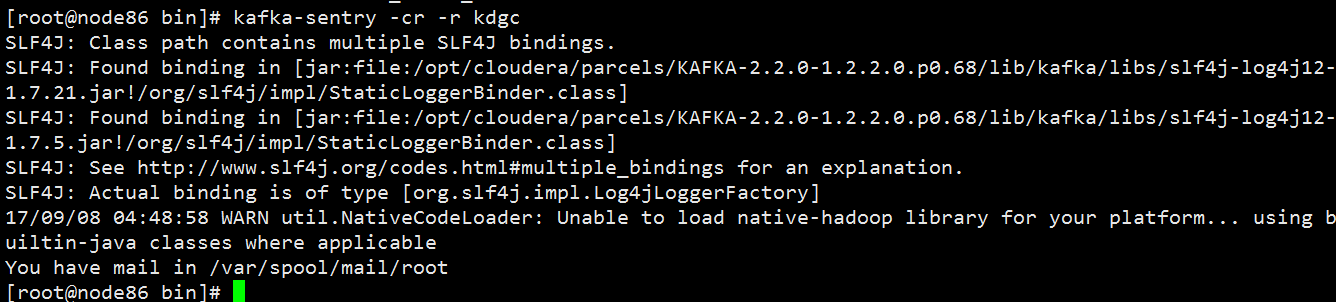
};

1. Sentry 权限配置

说明：Sentry权限配置只需要在服务器端进行，客户端无需配置，具体的权限配置方法请参考文档：<https://cwiki.apache.org/confluence/display/SENTRY/Apache+Kafka+Authorization+with+Apache+Sentry>

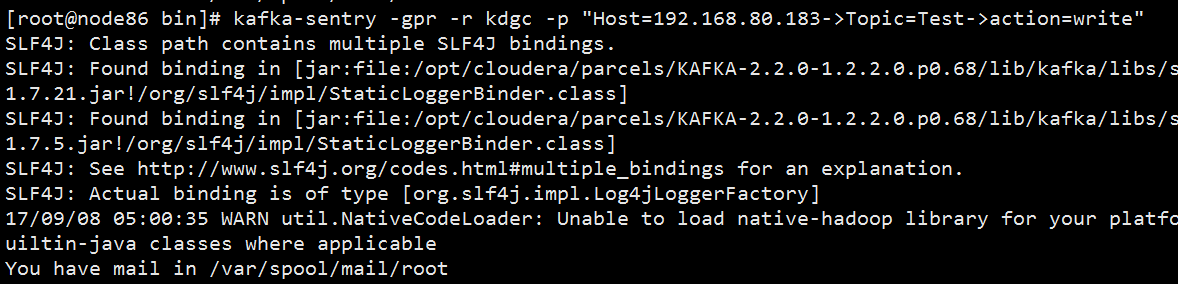
1. 创建角色kdgc，在node86上执行命令：

kafka-sentry -cr -r kdgc



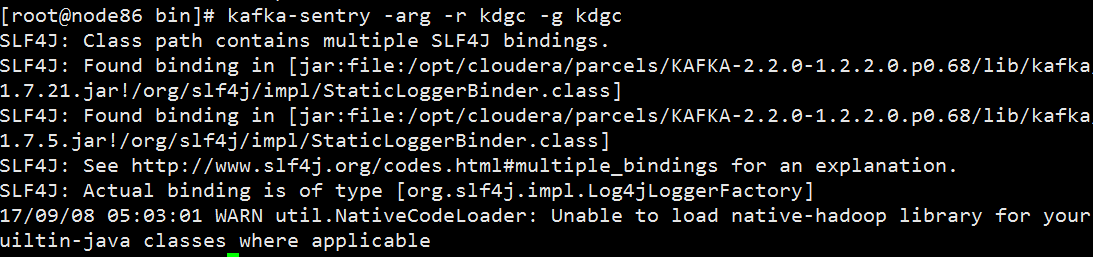
1. 给角色kdgc赋予如下权限: 可以在node183(192.168.80.183)上对Test主题发送消息，命令如下：

kafka-sentry -gpr -r kdgc -p "Host=192.168.80.183->Topic=Test->action=write"



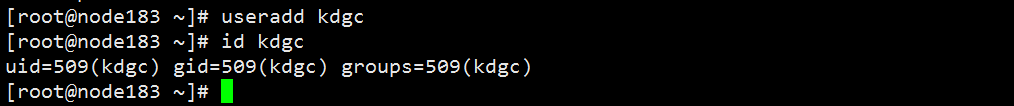
1. 将角色kdgc加入到kdgc组中，命令如下：

kafka-sentry -arg -r kdgc -g kdgc



1. 添加Linux用户kdgc，命令如下：

useradd kdgc



1. 用相同的方法创建一个消费者用户consumer
2. 创建sentry角色

kafka-sentry -cr -r consumer

1. 分配权限

kafka-sentry -gpr -r consumer -p "Host=192.168.80.183->Consumergroup= consumer ->action=read"

1. 加入consumer组

kafka-sentry -arg -r consumer -g consumer

1. 创建Linux用户consumer

useradd consumer

1. 通过Java代码进行生产和消费
2. 基于ACL认证
3. 客户端生产者代码

|  |
| --- |
| **public** **class** KafkaProducerTestCase {  **public** **static** **void** main(String[] args) {  Properties properties = **new** Properties();  /\*\*kafka集群地址及端口号，多台机器用逗号分隔\*\*/  properties.put("bootstrap.servers", "192.168.80.183:9092");  properties.put("acks", "all");  properties.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");  properties.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");  properties.put("kafka.retries", "3");  properties.put("batch.size", "16880");  properties.put("linger.ms", 1);  properties.put("buffer.memory", 33554432);  KafkaProducer<String, String> producer = **new** KafkaProducer<String, String>(properties);  /\*\*Test为Topic,message为发送的消息\*\*/  ProducerRecord<String, String> record1 = **new** ProducerRecord<String, String>("Test", "message");  producer.send(record1);  producer.close();    }  } |

1. 客户端消费者代码

|  |
| --- |
| **public** **class** KafkaConsumerTest {  **public** **static** **void** main(String[] args) {  Properties properties = **new** Properties();  /\*\*kafka集群地址及端口号，多台机器用逗号分隔\*\*/  properties.put("bootstrap.servers", "192.168.80.183:9092");  properties.put("acks", "all");  properties.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  properties.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  properties.put("session.timeout.ms", "180000");  properties.put("max.poll.records", "100");  properties.put("auto.offset.reset", "earliest");  properties.put("auto.commit.interval.ms", "500");  properties.put("max.poll.interval.ms", "5000");  properties.put("enable.auto.commit", "false");  properties.put("heartbeat.interval.ms", "3000");  properties.put("request.timeout.ms", "300000");  properties.put("max.poll.interval.ms", "300000");  KafkaConsumer<String, String> consumer = **new** KafkaConsumer<String, String>(properties);  /\*\*订阅的Topic\*\*/  consumer.subscribe(Arrays.*asList*("Test"));  **while** (**true**) {  /\*\*消费者订阅topic后,调用poll方法,加入到组,要留在组中,必须持续调用poll方法\*\*/  **try** {  Thread.*sleep*(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  ConsumerRecords<String, String> records = consumer.poll(100);  **for** (ConsumerRecord<String, String> record : records) {  System.***out***.println(record.topic()+" --- "+ record.partition());  System.***out***.printf("offset = %d, key = %s, value = %s", record.offset(),record.key(),record.value()+" \r\n");  }  }  }  } |

1. 基于Kerberos和Sentry认证
2. 客户端生产者代码

|  |
| --- |
| **import** org.apache.kafka.clients.producer.ProducerRecord;  **import** org.apache.kafka.clients.producer.KafkaProducer;  **public** **class** KafkaProducerTestCase {  **public** **static** **void** main(String[] args) {  Properties properties = **new** Properties();  /\*\*kafka集群地址及端口号，多台机器用逗号分隔\*\*/  properties.put("bootstrap.servers", "192.168.80.183:9092");  properties.put("acks", "all");  properties.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");  properties.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");  properties.put("kafka.retries", "3");  properties.put("batch.size", "16880");  properties.put("linger.ms", 1);  properties.put("buffer.memory", 33554432);  /\*\*Kerberos服务名,对应于kafka-jaas.conf中的serviceName\*\*/  properties.put("sasl.kerberos.service.name", "kafka");  /\*\*服务器端的安全协议,此处是Kerberos认证,所以是SASL\_PLAINTEXT\*\*/  properties.put("security.protocol", "SASL\_PLAINTEXT");  /\*\*添加Kerberos认证所需的JAAS配置文件到运行时环境\*\*/  System.*setProperty*("java.security.auth.login.config","/root/kafka-jaas.conf");  /\*\*添加krb5配置文件到运行时环境\*\*/  System.*setProperty*("java.security.krb5.conf", "/etc/krb5.conf");  KafkaProducer<String, String> producer = **new** KafkaProducer<String, String>(properties);  /\*\*Test为Topic,message为发送的消息\*\*/  ProducerRecord<String, String> record1 = **new** ProducerRecord<String, String>("Test", "message");  producer.send(record1);  producer.close();  }  } |

1. 客户端消费者代码

|  |
| --- |
| **public** **class** KafkaConsumerTest {  **public** **static** **void** main(String[] args) {  Properties properties = **new** Properties();  /\*\*kafka集群地址及端口号，多台机器用逗号分隔\*\*/  properties.put("bootstrap.servers", "192.168.80.183:9092");  properties.put("acks", "all");  properties.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  properties.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");  properties.put("session.timeout.ms", "180000");  properties.put("max.poll.records", "100");  properties.put("auto.offset.reset", "earliest");  properties.put("auto.commit.interval.ms", "500");  properties.put("max.poll.interval.ms", "5000");  properties.put("enable.auto.commit", "false");  properties.put("heartbeat.interval.ms", "3000");  properties.put("request.timeout.ms", "300000");  properties.put("max.poll.interval.ms", "300000");  /\*\*Kerberos服务名,对应于kafka-jaas.conf中的serviceName\*\*/  properties.put("sasl.kerberos.service.name", "kafka");  /\*\*服务器端的安全协议,此处是Kerberos认证,所以是SASL\_PLAINTEXT\*\*/  properties.put("security.protocol", "SASL\_PLAINTEXT");  /\*\*消费者组名\*\*/  properties.put("group.id", "Test");  /\*\*添加Kerberos认证所需的JAAS配置文件到运行时环境\*\*/  System.*setProperty*("java.security.auth.login.config","/root/kafka-jaas.conf");  /\*\*添加krb5配置文件到运行时环境\*\*/  System.*setProperty*("java.security.krb5.conf", "/etc/krb5.conf");  KafkaConsumer<String, String> consumer = **new** KafkaConsumer<String, String>(properties);  /\*\*订阅的Topic\*\*/  consumer.subscribe(Arrays.*asList*("Test"));  **while** (**true**) {  /\*\*消费者订阅topic后,调用poll方法,加入到组,要留在组中,必须持续调用poll方法\*\*/  **try** {  Thread.*sleep*(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  ConsumerRecords<String, String> records = consumer.poll(100);  **for** (ConsumerRecord<String, String> record : records) {  System.***out***.println(record.topic()+" --- "+ record.partition());  System.***out***.printf("offset = %d, key = %s, value = %s", record.offset(),record.key(),record.value()+" \r\n");  }  }  }  } |