Cluster **AMBARI**

Autenticación **KERBEROS**

Autorización **KAFKA**

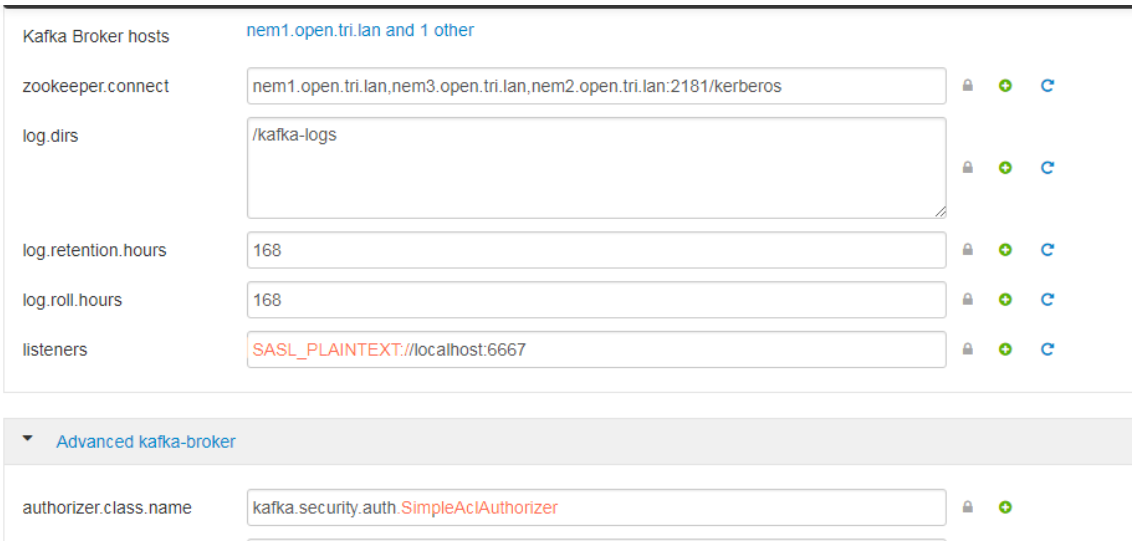
Pasos para configurar el entorno seguro Kafka en un cluster Ambari mediante autenticación por Kerberos.

Configuración de Brokers

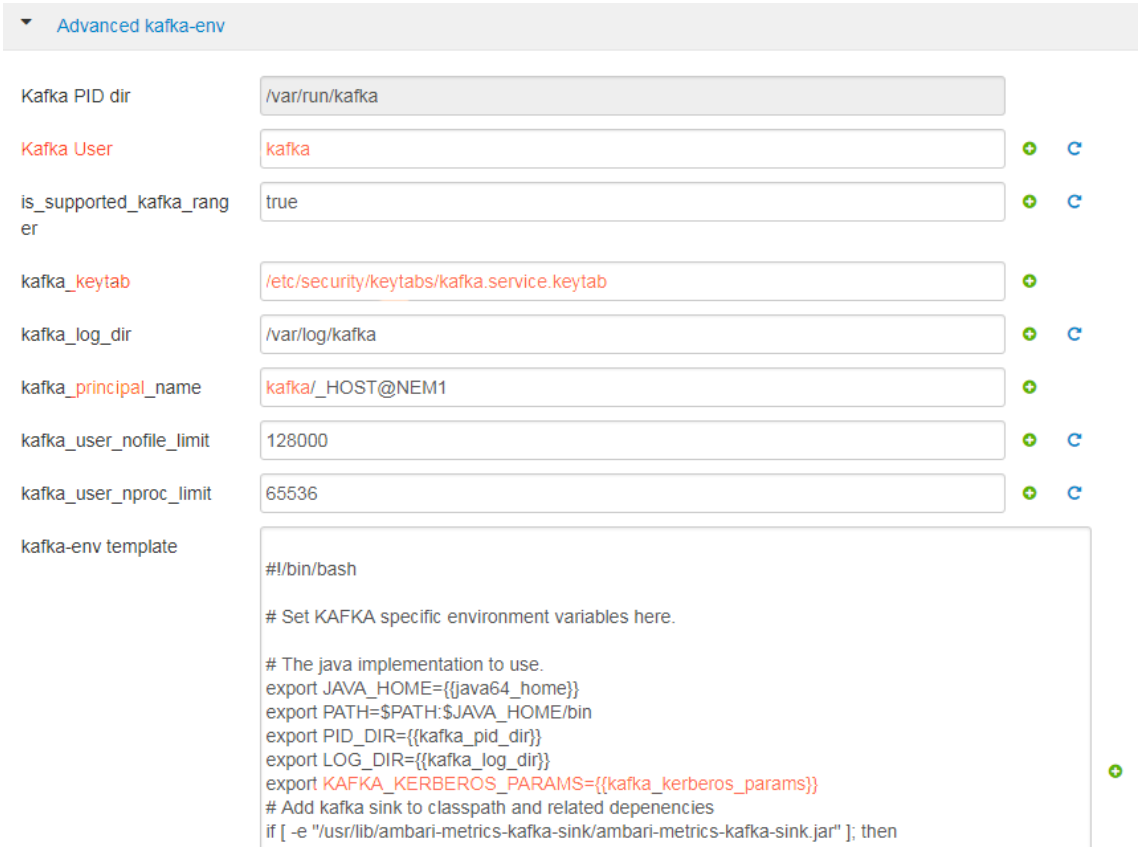
Algunas de las modificaciones necesarias para soportar la autorización las realiza Ambari automáticamente al kerberizar el cluster:

- Crear fichero JAAS.
- Configurar JVM.
- Establecer protocolo inter-broker y con Zookeeper.

Es necesario definir el protocolo **SASL** en los puertos de los brokers. Se especifica también la clase administradora de las **ACL**. Comprobar que el parámetro `zookeeper.set.acl` es **`true`.**

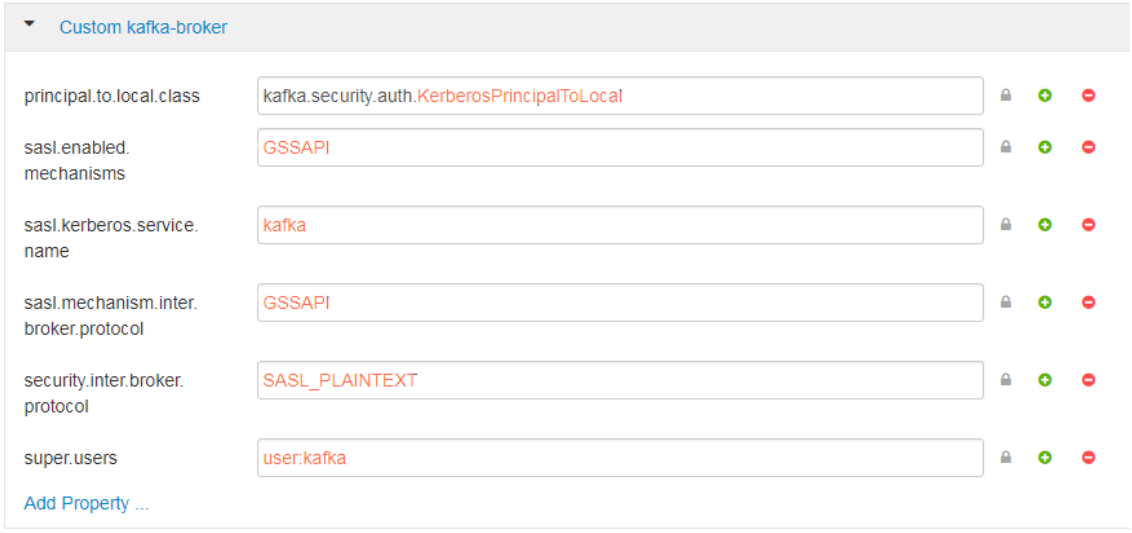


Ambari automatiza la configuración del apartado *Advanced kafka-env,* donde se define el entorno Kerberos del broker. Comprobar que tanto los principales, realms, keytabs y la configuración de la JVM (*KAFKA_KERBEROS_PARAMS)* son las correctas:



En el apartado *Custom kafka-broker* se establecen las políticas restantes para configurar el broker mediante Kerberos. Comprobar que se utiliza el mecanismo GSSAPI (propio de Kerberos para SASL).

Comprobar que el valor del protocolo en los parámetros ***listeners*** y ***security.inter.broker.protocol*** es idéntico: Ambari permite SASL_PLAINTEXT y PLAINTEXTSASL como String para realizar autenticación por SASL y envío no cifrado, configurando automáticamente ***security.inter.broker.protocol*** pero no ***listeners***. Si uno de ellos tiene como valor SASL_PLAINTEXT y el otro PLAINTEXTSASL, el broker no arrancará.



En cuanto a la configuración JAAS, Ambari se encarga de generar las secciones *KafkaServer* y *Client* y no se recomienda realizar cambios en cuanto al fichero ni a la configuración JVM.

```
KafkaServer {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   keyTab="/etc/security/keytabs/kafka.service.keytab"
   storeKey=true
   useTicketCache=false
   serviceName="kafka"
   principal="kafka/nem1.open.tri.lan@NEM1";
};
```

El server se autentica en el cluster como ***kafka*** indicando la ubicación de su keytab. Este apartado se utiliza también para la comunicación entre los brokers (en este caso, el jaas es de un broker del host *nem1.open.tri.lan*). Los demás parámetros se utilizan para indicar a Kerberos que para este servicio no se utiliza la caché de tickets (kinit), sino que se requiere un fichero keytab. Es la opción recomendada para servicios *"long-term"*.

```
Client {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   keyTab="/etc/security/keytabs/kafka.service.keytab"
   storeKey=true
   useTicketCache=false
   serviceName="zookeeper"
   principal="kafka/nem1.open.tri.lan@NEM1";
};
```

Para comunicación con el zookeeper. Al igual que con la sección anterior, indicamos que utilizamos autenticación mediante keytab.

Configuración de Clientes (Producers/Consumers)

Los ficheros de properties de los clientes deben establecer los siguientes valores:

```
security_protocol = SASL_PLAINTEXT
sasl.mechanism = GSSAPI
```

En cuanto a la configuración ***JAAS,*** existen dos enfoques para autenticar a los clientes:

1. **Caché de tickets**.

El consumer/producer que solicite conectarse al broker utilizará el ticket kerberos en caché (el del último *kinit* requerido al servidor Kerberos). Ejemplo de JAAS de un cliente basado en tickets. En este modo, antes de conectarse al broker el cliente debe hacer *kinit* con el usuario habilitado.

```
KafkaClient {
   com.sun.security.auth.module.Krb5LoginModule required
   useTicketCache=true
   renewTicket=true
   serviceName="kafka";
};
```

Ejemplo de fichero JAAS de cliente basado en tickets.

2. **Keytabs.**

Al principal de Kerberos se le asocia un keytab, el cual se transfiere a las máquinas que realizarán trabajos de cliente, como consumir o producir en una cola. La diferencia con respecto al método anterior es que no se utiliza la caché de tickets, sino que la autenticación se realiza comprobando el fichero .keytab. En este modo, se libera al usuario de la necesidad de introducir la contraseña del principal.

```
KafkaClient{
        com.sun.security.auth.module.Krb5LoginModule required
        useKeyTab=true
        storeKey=true
        keyTab="/etc/security/keytabs/kafka_client.keytab"
        principal="kafka_client@REALM;
};
```

Ejemplo de fichero JAAS de cliente basado en Keytab.

Configuración de Zookeeper

Ambari realiza todas las modificaciones necesarias de manera automática y no se recomienda ningún cambio manual.

ESCENARIO y CASOS DE USO

Escenario de pruebas y principales implicados:

BROKER

Conexión -> **SASL_PLAINTEXT**://*brokerHost*:6667 (SASL)
Autenticación –> GSSAPI
Autorización -> Kafka ACL
Kerberos Principal -> **kafka**

ZOOKEEPER

Conexión -> *zooHost*:2181/znodeA (SASL / no Auth)
Autenticación –> GSSAPI
Autorización -> Zookeeper ACL
Kerberos Principal -> **zookeeper**

CONSUMERs

Conexión -> *zooHost*:2181/znodeA (Console-Consumer)
Autenticación –> GSSAPI
Kerberos Principals -> **consumerKafka, consumerKafkaInvitado**

PRODUCERs

Conexión -> *brokerHost:6667*
Autenticación –> GSSAPI
Kerberos Principals -> **producerKafka, producerKafkaInvitado**

TOPICS

- TopicPúblico
- TopicPrivado

Tabla de principales y permisos requeridos por Topic:

Topic/Permisos	READ	WRITE
TopicPrivado	consumerKafka	producerKafka
TopicPúblico	consumerKafkaInvitado, consumerKafka	producerKafkaInvitado, producerKafka

1. Crear principales

Crear los principales desde ***kadmin.local***:

En el ejemplo, creación de los principales Producers. En este caso, no utilizan un keytab, sino que habría que introducir la password cada vez que quisiéramos usar alguno de los principales.

```
kadmin.local:  add_principal producerKafka@NEM1
WARNING: no policy specified for producerKafka@NEM1; defaulting to no policy
Enter password for principal "producerKafka@NEM1":
Re-enter password for principal "producerKafka@NEM1":
Principal "producerKafka@NEM1" created.
kadmin.local:  add_principal producerKafkaInvitado@NEM1
WARNING: no policy specified for producerKafkaInvitado@NEM1; defaulting to no policy
Enter password for principal "producerKafkaInvitado@NEM1":
Re-enter password for principal "producerKafkaInvitado@NEM1":
Principal "producerKafkaInvitado@NEM1" created.
```

2. Levantar broker

En Ambari , los brokers se inician y paran desde la propia interfaz, donde se utiliza el principal Kerberos asociado al servicio Kafka : ***kafka.***
No habría que realizar ninguna modificación en cuanto a permisos/ACLs para esta función.
En la imagen, el movimiento de tickets al lanzar la operación de levantar el broker desde Ambari.

```
AS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506322892, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tri.lan@NEM1 for krbtgt/NEM1@NEM1
TGS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506322892, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tri.lan@NEM1 for zookeeper/nem1.open.tri.

TGS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506322892, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tri.lan@NEM1 for zookeeper/nem2.open.tri.

AS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506322893, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tri.lan@NEM1 for krbtgt/NEM1@NEM1
TGS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506322892, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tri.lan@NEM1 for zookeeper/nem3.open.tri.

TGS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506322893, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tri.lan@NEM1 for kafka/nem1.open.tri.lan@
```

3. Crear topic sin modificar zookeper ACLs (kafka)

Al ejecutar la operación créate topic desde el superUser ***kafka***, la operación es satisfactoria:

```
rtit[root@NEM1 kafka-broker]# bin/kafka-topics.sh --create --topic topicSuperUser --replication-factor 1 \
> --partitions 2 --zookeeper nem1.open.tri.lan:2181/znodeB
Created topic "topicSuperUser".
```

Kerberos:

- El usuario ***kafka*** intenta autenticarse contra el servidor Kerberos. Si el keytab del usuario ***kafka*** es el adecuado, la autenticación es correcta.
- Una vez autenticado, kafka pide al *Ticket Granting Service* un ticket para acceder al servicio zookeeper.

AS_REQ kafka → kerberos
TGS_REQ kafka → zookeeper

```
Sep 25 09:11:11 NEM1 krb5kdc[34506](info): AS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506323471, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.tr
i.lan@NEM1 for krbtgt/NEM1@NEM1
Sep 25 09:11:11 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.26.251.70: ISSUE: authtime 1506323471, etypes {rep=18 tkt=18 ses=18}, kafka/nem1.open.t
ri.lan@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
```

4. Crear topic modificando zookeper ACLs (Kafka)

En este caso, se modifican las ACLs del zookeeper, dando únicamente permisos de lectura **(r)** al usuario ***kafka***:

```
[zk: localhost:2181(CONNECTED) 1] getAcl /znodeF/brokers/topics
'sasl,'kafka
: r
```

Al intentar crear el topic, Zookeeper avisa que no tenemos permisos, por lo que la creación falla.

```
[root@NEM2 kafka-broker]# bin/kafka-topics.sh --replication-factor 1 --partitions 1 --zookeeper nem1.open.tri.lan:2181 --create --topic testSinAuth
Error while executing topic command : replication factor: 1 larger than available brokers: 0
[2017-10-02 11:29:01,247] ERROR kafka.admin.AdminOperationException: replication factor: 1 larger than available brokers: 0
        at kafka.admin.AdminUtils$.assignReplicasToBrokers(AdminUtils.scala:77)
        at kafka.admin.AdminUtils$.createTopic(AdminUtils.scala:236)
        at kafka.admin.TopicCommand$.createTopic(TopicCommand.scala:105)
        at kafka.admin.TopicCommand$.main(TopicCommand.scala:60)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
```

5. Crear topic con usuario no kafka (consumerKafka)

Tras realizar el kinit, el usuario ***consumerKafka*** trata de crear un topic pero da un error de autorización. Viendo el log de Kerberos, vemos que ***consumerKafka*** se autentica correctamente y le es entregado un ticket para acudir al zookeeper.

```
TGS_REQ (4 etypes {18 17 16 23}) 172.26.251.73: ISSUE: authtime 1506941262, etypes {rep=18 tkt=18 ses=18}, consumerKafka@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
```

Al intentar el acceso, es el zookeeper el encargado de suspender la operación por falta de privilegios. Sólo Kafka está habilitado para crear los topics.

```
[root@NEM4 kafka_2.11-0.10.1.0]# kinit consumerKafka@NEM1
Password for consumerKafka@NEM1:
[root@NEM4 kafka_2.11-0.10.1.0]# bin/kafka-topics.sh --create --topic testSinAuth --zookeeper nem1.open.tri.lan:2181/znodeE --replication-factor 1 --partitions 4
Error while executing topic command : org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth for /config/topics
[2017-10-02 12:01:01,383] ERROR org.I0Itec.zkclient.exception.ZkException: org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth for /config/topics
        at org.I0Itec.zkclient.exception.ZkException.create(ZkException.java:68)
        at org.I0Itec.zkclient.ZkClient.retryUntilConnected(ZkClient.java:1000)
        at org.I0Itec.zkclient.ZkClient.create(ZkClient.java:527)
        at org.I0Itec.zkclient.ZkClient.createPersistent(ZkClient.java:293)
        at kafka.utils.ZkPath$.createPersistent(ZkUtils.scala:965)
        at kafka.utils.ZkUtils.createParentPath(ZkUtils.scala:437)
        at kafka.utils.ZkUtils.updatePersistentPath(ZkUtils.scala:507)
        at kafka.admin.AdminUtils$.writeEntityConfig(AdminUtils.scala:568)
        at kafka.admin.AdminUtils$.createOrUpdateTopicPartitionAssignmentPathInZK(AdminUtils.scala:453)
        at kafka.admin.AdminUtils$.createTopic(AdminUtils.scala:415)
        at kafka.admin.TopicCommand$.createTopic(TopicCommand.scala:107)
        at kafka.admin.TopicCommand$.main(TopicCommand.scala:60)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
Caused by: org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth for /config/topics
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:113)
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:51)
        at org.apache.zookeeper.ZooKeeper.create(ZooKeeper.java:783)
        at org.I0Itec.zkclient.ZkConnection.create(ZkConnection.java:99)
        at org.I0Itec.zkclient.ZkClient$3.call(ZkClient.java:530)
        at org.I0Itec.zkclient.ZkClient$3.call(ZkClient.java:527)
        at org.I0Itec.zkclient.ZkClient.retryUntilConnected(ZkClient.java:990)
        ... 11 more
 (kafka.admin.TopicCommand$)
```

6. Producir en topic (producerKafka)

Primero nos autenticamos como el usuario "*kafka*" para evitar que nos de errores al crear el topic:

```
[root@NEM1 kafka-broker]# sudo bin/kafka-topics.sh --create --topic pruebaProducer --replication-factor 1 --partitions 1 --zookeeper nem1.open.tri.lan:2181/znodeG
Error while executing topic command : org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth for /config/topics
[2017-10-03 12:47:50,077] ERROR org.I0Itec.zkclient.exception.ZkException: org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth for /config/topics
        at org.I0Itec.zkclient.exception.ZkException.create(ZkException.java:68)
        at org.I0Itec.zkclient.ZkClient.retryUntilConnected(ZkClient.java:995)
        at org.I0Itec.zkclient.ZkClient.create(ZkClient.java:526)
        at org.I0Itec.zkclient.ZkClient.createPersistent(ZkClient.java:292)
        at kafka.utils.ZkPath$.createPersistent(ZkUtils.scala:916)
        at kafka.utils.ZkUtils.createParentPath(ZkUtils.scala:339)
        at kafka.utils.ZkUtils.updatePersistentPath(ZkUtils.scala:414)
        at kafka.admin.AdminUtils$.writeEntityConfig(AdminUtils.scala:346)
        at kafka.admin.AdminUtils$.createOrUpdateTopicPartitionAssignmentPathInZK(AdminUtils.scala:269)
        at kafka.admin.AdminUtils$.createTopic(AdminUtils.scala:237)
        at kafka.admin.TopicCommand$.createTopic(TopicCommand.scala:105)
        at kafka.admin.TopicCommand$.main(TopicCommand.scala:60)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
Caused by: org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth for /config/topics
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:113)
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:51)
        at org.apache.zookeeper.ZooKeeper.create(ZooKeeper.java:783)
        at org.I0Itec.zkclient.ZkConnection.create(ZkConnection.java:99)
        at org.I0Itec.zkclient.ZkClient$3.call(ZkClient.java:529)
        at org.I0Itec.zkclient.ZkClient$3.call(ZkClient.java:526)
        at org.I0Itec.zkclient.ZkClient.retryUntilConnected(ZkClient.java:985)
        ... 11 more
 (kafka.admin.TopicCommand$)
[root@NEM1 kafka-broker]# kinit kafka/nem1.open.tri.lan@NEM1 -kt /etc/security/keytabs/kafka.service.keytab
[root@NEM1 kafka-broker]# sudo bin/kafka-topics.sh --create --topic pruebaProducer --replication-factor 1 --partitions 1 --zookeeper nem1.open.tri.lan:2181/znodeG
Created topic "pruebaProducer".
```

Nos autenticamos como **producerKafka**, indicando en este caso la contraseña del principal (no tiene keytab):

```
[root@NEM1 kafka-broker]# kinit producerKafka@NEM1
Password for producerKafka@NEM1:
[root@NEM1 kafka-broker]#
```

Si intentamos producir sin realizar modificaciones en los ACL de Kafka, tenemos un error de falta de autorización:

```
# bin/kafka-console-producer.sh --broker-list 172.26.251.71:6667 --topic pruebaProducer --security-protocol SASL_PLAINTEXT

WARN Error while fetching metadata [{TopicMetadata for topic pruebaProducer ->
topic pruebaProducer due to kafka.common.TopicAuthorizationException}] for topic [pruebaProducer]: class kafka.common.TopicAuthorizationException

WARN Error while fetching metadata [{TopicMetadata for topic pruebaProducer ->
topic pruebaProducer due to kafka.common.TopicAuthorizationException}] for topic [pruebaProducer]: class kafka.common.TopicAuthorizationException
```

Modificamos las ACLs para darle permisos de escritura en el Topic *pruebaProducer*.

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties \
> zookeeper.connect=nem1.open.tri.lan:2181/znodeG --add --allow-principal User:producerKafka \
> --producer --topic pruebaProducer
```

```
Following is list of acls for resource: Topic:pruebaProducer
        User:producerKafka has Allow permission for operations: Write from hosts: *
        User:producerKafka has Allow permission for operations: Describe from hosts: *
```

De nuevo probamos a producir; Esta vez la operación es satisfactoria:

```
[root@NEM1 kafka-broker]# bin/kafka-console-producer.sh --broker-list 172.26.251.71:6667 --topic pruebaProducer --security-protocol SASL_PLAINTEXT
estoy escribiendo!!
y sin errores!!!
es porque tengo privilegios de producer en este topic
```

7. Leer de un topic (consumerKafka)

Tratamos de leer del topic *pruebaProducer* sin modificar los ACL de Kafka, falla la autorización:

```
[root@NEM1 kafka-broker]# kinit consumerKafka@NEM1
Password for consumerKafka@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --zookeeper nem1.open.tri.lan:2181/znodeG --topic pruebaProducer --security-protocol SASL_PLAINTEXT
[2017-10-03 13:33:54,301] WARN ZooKeeper event while creating registration node: /consumers NOAUTH (kafka.utils.ZKCheckedEphemeral)
[2017-10-03 13:33:54,324] ERROR Unknown error when running consumer:  (kafka.tools.ConsoleConsumer$)
org.I0Itec.zkclient.exception.ZkException: org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth
        at org.I0Itec.zkclient.exception.ZkException.create(ZkException.java:68)
        at kafka.utils.ZKCheckedEphemeral.create(ZkUtils.scala:1090)
        at kafka.consumer.ZookeeperConsumerConnector.kafka$consumer$ZookeeperConsumerConnector$$registerConsumerInZK(ZookeeperConsumerConnector.scala:289)
        at kafka.consumer.ZookeeperConsumerConnector$WildcardStreamsHandler.<init>(ZookeeperConsumerConnector.scala:1017)
        at kafka.consumer.ZookeeperConsumerConnector.createMessageStreamsByFilter(ZookeeperConsumerConnector.scala:179)
        at kafka.consumer.OldConsumer.<init>(BaseConsumer.scala:76)
        at kafka.tools.ConsoleConsumer$.run(ConsoleConsumer.scala:63)
        at kafka.tools.ConsoleConsumer$.main(ConsoleConsumer.scala:47)
        at kafka.tools.ConsoleConsumer.main(ConsoleConsumer.scala)
Caused by: org.apache.zookeeper.KeeperException$NoAuthException: KeeperErrorCode = NoAuth
        at org.apache.zookeeper.KeeperException.create(KeeperException.java:113)
```

Es necesario otorgarle permisos de lectura para el topic:

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --autho
-allow-principal User:consumerKafka --operation Read --topic pruebaProducer
Adding following acls for resource: Topic:pruebaProducer
        User:consumerKafka has Allow permission for operations: Read from hosts: *

Following is list of acls for resource: Topic:pruebaProducer
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
        User:producerKafka has Allow permission for operations: Describe from hosts: *
```

Lanzamos el consumer:

```
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --zookeeper nem1.open.t
opic pruebaProducer
[2017-10-03 17:02:45,986] WARN ZooKeeper event while creating registration node
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.ti
y.protocol=SASL_PLAINTEXT}
hola, soy el producer
si te llega esto, te han configurado bien
=)
```

Autorizaciones – casos de uso

✓ **Topics**:

1. TopicPúblico
2. TopicPrivado

✓ **Clientes** y sus **ACLs**:

1. **producerKafka** (autorización de escritura para los dos topics)

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer \
> --authorizer-properties zookeeper.connect=nem1.open.tri.lan:2181/znodeG --add --allow-principal \
> User:producerKafka --operation Write --topic TopicPublico
Adding following acls for resource: Topic:TopicPublico
        User:producerKafka has Allow permission for operations: Write from hosts: *

Following is list of acls for resource: Topic:TopicPublico
        User:producerKafka has Allow permission for operations: Write from hosts: *

[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties
ct=nem1.open.tri.lan:2181/znodeG --add --allow-principal User:producerKafka --operation Write --topic TopicPrivado
Adding following acls for resource: Topic:TopicPrivado
        User:producerKafka has Allow permission for operations: Write from hosts: *

Following is list of acls for resource: Topic:TopicPrivado
        User:producerKafka has Allow permission for operations: Write from hosts: *
```

2. **consumerKafka** (autorización de lectura para los dos topics)

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties
ct=nem1.open.tri.lan:2181/znodeG --add --allow-principal User:consumerKafka --operation Read --topic TopicPrivado
Adding following acls for resource: Topic:TopicPrivado
        User:consumerKafka has Allow permission for operations: Read from hosts: *

Following is list of acls for resource: Topic:TopicPrivado
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *

[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties
ct=nem1.open.tri.lan:2181/znodeG --add --allow-principal User:consumerKafka --operation Read --topic TopicPublico
Adding following acls for resource: Topic:TopicPublico
        User:consumerKafka has Allow permission for operations: Read from hosts: *

Following is list of acls for resource: Topic:TopicPublico
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
```

3. **consumerKafkaInvitado** (autorización de lectura para TopicPúblico)

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties zookeeper.
ct=nem1.open.tri.lan:2181/znodeG --add --allow-principal User:consumerKafkaInvitado --operation Read --topic TopicPublico
Adding following acls for resource: Topic:TopicPublico
        User:consumerKafkaInvitado has Allow permission for operations: Read from hosts: *

Following is list of acls for resource: Topic:TopicPublico
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
        User:consumerKafkaInvitado has Allow permission for operations: Read from hosts: *
```

4. **producerKafkaInvitado** (autorización de escritura para TopicPúblico)

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorizer-properties z
ct=nem1.open.tri.lan:2181/znodeG --add --allow-principal User:producerKafkaInvitado --operation Write --topic TopicPublico
Adding following acls for resource: Topic:TopicPublico
        User:producerKafkaInvitado has Allow permission for operations: Write from hosts: *

Following is list of acls for resource: Topic:TopicPublico
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
        User:consumerKafkaInvitado has Allow permission for operations: Read from hosts: *
        User:producerKafkaInvitado has Allow permission for operations: Write from hosts: *
```

CASOS DE USO

❖ **producerKafkaInvitado** escribe en *TopicPúblico*

Primero realizamos la carga en caché del ticket Kerberos, autenticándonos como *producerKafkaInvitado*:

```
[root@NEM1 kafka-broker]# kinit producerKafkaInvitado@NEM1
Password for producerKafkaInvitado@NEM1:
```

AS_REQ producerKafkaInvitado → kerberos

```
Oct 04 13:07:27 NEM1 krb5kdc[34506](info): AS_REQ (6 etypes {18 17 16 23 25 26})
 172.26.251.70: ISSUE: authtime 1507115247, etypes {rep=18 tkt=18 ses=18}, produ
cerKafkaInvitado@NEM1 for krbtgt/NEM1@NEM1
```

El *producerKafkaInvitado* no tiene problemas para escribir en el *TopicPúblico*:

```
Password for producerKafkaInvitado@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-producer.sh --broker-list 172.26.251
.71:6667 --topic TopicPublico --security-protocol SASL_PLAINTEXT
hola
hoola
```

Mirando el log de Kerberos, observamos que se la ha asignado correctamente un ticket para acceder al broker ubicado en *nem2.open.tri.lan*.

TGS_REQ producerKafkaInvitado → kafka (broker nem2)

```
26.251.70: ISSUE: authtime 1507115247, etypes {rep=18 tkt=18 ses=18}, producerKa
fkaInvitado@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ consumerKafka y consumerKafkaInvitado leen de TopicPúblico

- **consumerKafka**

Lanzamos el consumer habiéndonos autenticado anteriormente en Kerberos como *consumerKafka*.

```
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --zookeeper nem1.open.tr
i.lan:2181/znodeG --topic TopicPublico --security-protocol SASL_PLAINTEXT
[2017-10-04 13:43:11,342] WARN ZooKeeper event while creating registration node:
 /consumers NOAUTH (kafka.utils.ZKCheckedEphemeral)
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.tim
eout.ms=30000, client.id=console-consumer-83140, security.protocol=SASL_PLAINTEX
T}
mensaje desde el producer para consumerKafka en TopicPublico
```

ConsumerKafka no tiene problemas para leer del topic público. Mirando el log de Kerberos vemos que se le ha autorizado la entrada tanto en zookeeper como en los brokers (nem2,nem3).

```
AS_REQ   consumerKafka → Kerberos
TGS_REQ  consumerKafka → zookeeper
TGS_REQ  consumerKafka → broker(s)
```

```
26.251.70: ISSUE: authtime 1507117314, etypes {rep=18 tkt=18 ses=18}, consumerKa
fka@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
Oct 04 13:42:06 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507117314, etypes {rep=18 tkt=18 ses=18}, consumerKa
fka@NEM1 for kafka/nem3.open.tri.lan@NEM1
Oct 04 13:42:06 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507117314, etypes {rep=18 tkt=18 ses=18}, consumerKa
fka@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

- **consumerKafkaInvitado**

Lanzamos el consumer habiéndonos autenticado anteriormente en Kerberos como *consumerKafkaInvitado.* Dispone de permisos de lectura, por lo que no tiene problemas para leer del *TopicPublico:*

```
[root@NEM1 kafka-broker]# kdestroy
[root@NEM1 kafka-broker]# kinit consumerKafkaInvitado@NEM1
Password for consumerKafkaInvitado@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --zookeeper nem1.open.tri.lan:2181/znodeG --topic TopicPublico --security-protocol SASL
[2017-10-04 13:38:29,377] WARN ZooKeeper event while creating registration node: /consumers NOAUTH (kafka.utils.ZKCheckedEphemeral)
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-41847, security.proto
mensaje desde el producer para consumerKafkaInvitado en TopicPublico
```

```
AS_REQ   consumerKafkaInvitado → Kerberos
TGS_REQ  consumerKafkaInvitado → Zookeeper
TGS_REQ  consumerKafkaInvitado → Broker(s)
```

```
26.251.70: ISSUE: authtime 1507117716, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
Oct 04 13:48:41 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507117716, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for kafka/nem3.open.tri.lan@NEM1
Oct 04 13:48:41 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507117716, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ **producerKafka** escribe en TopicPrivado

El *producerKafka* tiene permisos para escribir en el *TopicPrivado:*

```
[root@NEM1 kafka-broker]# kdestroy
[root@NEM1 kafka-broker]# kinit producerKafka@NEM1
Password for producerKafka@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-producer.sh --broker-list 172.26.251
.71:6667 --topic TopicPrivado --security-protocol SASL_PLAINTEXT
hola
soy el producerKafka, escribiendo en el topic privado
```

AS_REQ producerKafka → Kerberos
TGS_REQ producerKafka → Broker(nem2) → (1 partición)

```
Oct 04 13:49:53 NEM1 krb5kdc[34506](info): AS_REQ (6 etypes {18 17 16 23 25 26})
 172.26.251.70: ISSUE: authtime 1507117793, etypes {rep=18 tkt=18 ses=18}, produ
cerKafka@NEM1 for krbtgt/NEM1@NEM1
Oct 04 13:50:14 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507117793, etypes {rep=18 tkt=18 ses=18}, producerKa
fka@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ **consumerKafka** lee de TopicPrivado

El *consumerKafka* tiene permisos para escribir en el *TopicPrivado:*

```
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.tim
eout.ms=30000, client.id=console-consumer-95267, security.protocol=SASL_PLAINTEX
T}
hola, si lees esto estás autorizado para leer del TopicPrivado
```

AS_REQ consumerKafka → Kerberos
TGS_REQ consumerKafka → Zookeeper
TGS_REQ consumerKafka → Broker(nem2)

```
Oct 04 16:16:00 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507126546, etypes {rep=18 tkt=18 ses=18}, consumerKa
fka@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
Oct 04 16:16:01 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507126546, etypes {rep=18 tkt=18 ses=18}, consumerKa
fka@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ **consumerKafkaInvitado** intenta leer de TopicPrivado

El *consumerKafkaInvitado* no tiene los ACL necesarios para leer de la cola privada:

```
[root@NEM1 kafka-broker]# kinit consumerKafkaInvitado@NEM1
Password for consumerKafkaInvitado@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --zookeeper nem1.open.tr
i.lan:2181/znodeG --topic TopicPrivado --security-protocol SASL_PLAINTEXT
[2017-10-04 15:11:29,310] WARN ZooKeeper event while creating registration node:
 /consumers NOAUTH (kafka.utils.ZKCheckedEphemeral)
[2017-10-04 15:11:29,320] ERROR Unknown error when running consumer:  (kafka.too
ls.ConsoleConsumer$)
org.I0Itec.zkclient.exception.ZkException: org.apache.zookeeper.KeeperException$
NoAuthException: KeeperErrorCode = NoAuth
        at org.I0Itec.zkclient.exception.ZkException.create(ZkException.java:68)
        at kafka.utils.ZKCheckedEphemeral.create(ZkUtils.scala:1090)
        at kafka.consumer.ZookeeperConsumerConnector.kafka$consumer$ZookeeperCon
```

Mirando el log de Kerberos, observamos que se le ha asignado el ticket para acceder al *broker*, pero una vez dentro éste le ha denegado la lectura ya que no tiene permisos para el *TopicPrivado*:

```
AS_REQ   consumerKafkaInvitado → Kerberos
TGS_REQ  consumerKafkaInvitado → Zookeeper
TGS_REQ  consumerKafkaInvitado → Broker(nem2)
```

```
26.251.70: ISSUE: authtime 1507127426, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
Oct 04 16:31:13 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507127426, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ **producerKafkaInvitado** intenta escribir en TopicPrivado

El *producerKafkaInvitado* no tiene los ACL necesarios para escribir de la cola privada:

```
[root@NEM1 kafka-broker]# kinit producerKafkaInvitado@NEM1
Password for producerKafkaInvitado@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-producer.sh --broker-list 172.26.251
.71:6667 --topic TopicPrivado --security-protocol SASL_PLAINTEXT
creo que va a dar error..
[2017-10-04 16:36:49,941] WARN Error while fetching metadata [{TopicMetadata for
 topic TopicPrivado ->
No partition metadata for topic TopicPrivado due to kafka.common.TopicAuthorizat
ionException}] for topic [TopicPrivado]: class kafka.common.TopicAuthorizationEx
ception  (kafka.producer.BrokerPartitionInfo)
```

Mirando el log de Kerberos, observamos que se le ha asignado el ticket para acceder al *broker*, pero una vez dentro éste le ha denegado la escritura ya que no tiene permisos para el *TopicPrivado*:

```
AS_REQ   producerKafkaInvitado → Kerberos
TGS_REQ  consumerKafkaInvitado → Broker(nem2)
```

```
Oct 04 16:38:21 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507127745, etypes {rep=18 tkt=18 ses=18}, producerKa
fkaInvitado@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ permitir lectura de **consumerKafkaInvitado** en TopicPúblic

Mediante las ACL de Kafka, le otorgamos al *consumerKafkaInvitado* permisos temporales para consumir de la cola Privada:

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.Sim
pleAclAuthorizer --authorizer-properties zookeeper.connect=nem1.open.tri.lan:218
1/znodeG --add --allow-principal User:consumerKafkaInvitado --operation Read --t
opic TopicPrivado
Adding following acls for resource: Topic:TopicPrivado
        User:consumerKafkaInvitado has Allow permission for operations: Read fro
m hosts: *

Following is list of acls for resource: Topic:TopicPrivado
        User:consumerKafka has Allow permission for operations: Read from hosts:
 *
        User:producerKafka has Allow permission for operations: Write from hosts
: *
        User:consumerKafkaInvitado has Allow permission for operations: Read fro
m hosts: *
```

Lanzamos de nuevo el *consumerKafkaInvitado* contra el *TopicPrivado:*

```
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --zookeeper nem1.open.tr
i.lan:2181/znodeG --topic TopicPrivado --security-protocol SASL_PLAINTEXT
[2017-10-04 15:17:31,063] WARN ZooKeeper event while creating registration node:
 /consumers NOAUTH (kafka.utils.ZKCheckedEphemeral)
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.tim
eout.ms=30000, client.id=console-consumer-37021, security.protocol=SASL_PLAINTEX
T}
hola invitado, si lees esto te hemos dado permisos correctamente
```

En este caso, la operación ha sido satisfactoria. Mirando las operaciones de Kerberos observamos que el trueque de tickets es idéntico a cuando no tenía permisos:

AS_REQ consumerKafkaInvitado → Kerberos
TGS_REQ consumerKafkaInvitado → Zookeeper
TGS_REQ consumerKafkaInvitado → Broker(nem2)

```
Oct 04 15:17:30 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507123041, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for zookeeper/nem1.open.tri.lan@NEM1
Oct 04 15:17:31 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.
26.251.70: ISSUE: authtime 1507123041, etypes {rep=18 tkt=18 ses=18}, consumerKa
fkaInvitado@NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ permitir escritura a **producerKafkaInvitado** en TopicPrivado

En este caso, se quiere dar permiso tempOral al Producer Invitado, de manera que pueda escribir en la cola privada. Para ello, al igual que con el *consumerKafkaInvitado,* utilizamos las ACL de Kafka para otorgarle permisos de escritura en *TopicPrivado.*

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAcl
Authorizer --authorizer-properties zookeeper.connect=nem1.open.tri.lan:2181/znodeG --a
dd --allow-principal User:producerKafkaInvitado --operation Write --topic TopicPrivado

Adding following acls for resource: Topic:TopicPrivado
        User:producerKafkaInvitado has Allow permission for operations: Write from hos
ts: *

Following is list of acls for resource: Topic:TopicPrivado
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
        User:consumerKafkaInvitado has Allow permission for operations: Read from host
s: *
        User:producerKafkaInvitado has Allow permission for operations: Write from hos
ts: *
```

Intentamos producir y observamos que ya no tenemos errores:

```
[root@NEM1 kafka-broker]# kinit producerKafkaInvitado@NEM1
Password for producerKafkaInvitado@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-producer.sh --broker-list 172.26.251.71:66
67 --topic TopicPrivado --security-protocol SASL_PLAINTEXT
soy el producer invitado, escribiendo en la cola privada!
```

AS_REQ producerKafkaInvitado → Kerberos
TGS_REQ producerKafkaInvitado → Broker

```
Oct 04 15:24:00 NEM1 krb5kdc[34506](info): TGS_REQ (4 etypes {18 17 16 23}) 172.26.251
.70: ISSUE: authtime 1507123390, etypes {rep=18 tkt=18 ses=18}, producerKafkaInvitado@
NEM1 for kafka/nem2.open.tri.lan@NEM1
```

❖ **Revocar permisos** a los dos clientes invitados en TopicPrivado

Ahora se requiere revocar los permisos de la cola privada, de manera que únicamente *consumerKafka* y *producerKafka* estén autorizados para utilizar el *TopicPrivado*.

- **ProducerInvitado**

Utilizando las ACL de Kafka, se eliminan los permisos de escritura en el *TopicPrivado.*

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorize
r-properties zookeeper.connect=nem1.open.tri.lan:2181/znodeG --remove --allow-principal User:producerKafkaIn
vitado --operation Write --topic TopicPrivado
Are you sure you want to remove acls:
        User:producerKafkaInvitado has Allow permission for operations: Write from hosts: *
 from resource Topic:TopicPrivado y/n?
y
Following is list of acls for resource: Topic:TopicPrivado
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
        User:consumerKafkaInvitado has Allow permission for operations: Read from hosts: *
```

```
[root@NEM1 kafka-broker]# kinit producerKafkaInvitado@NEM1
Password for producerKafkaInvitado@NEM1:
[root@NEM1 kafka-broker]# bin/kafka-console-producer.sh --broker-list 172.26.251.71:6667 --topic TopicPriv
o --security-protocol SASL_PLAINTEXT
hola
[2017-10-04 15:29:22,934] WARN Error while fetching metadata [{TopicMetadata for topic TopicPrivado ->
No partition metadata for topic TopicPrivado due to kafka.common.TopicAuthorizationException}] for topic [T
picPrivado]: class kafka.common.TopicAuthorizationException  (kafka.producer.BrokerPartitionInfo)
```

El *producerKafkaInvitado* , al intentar producir sobre el *TopicPrivado,* se encuentra con un error de autorización sobre el Topic: ya no tiene acceso a la cola.

- **ConsumerInvitado**

Al igual que con el *producerKafkaInvitado*, al consumer se le revocan los permisos de lectura mediante las ACL de Kafka; En este momento, los permisos sobre el *TopicPrivado* vuelven a ser los originales (sin permisos para invitados):

```
[root@NEM1 kafka-broker]# bin/kafka-acls.sh --authorizer kafka.security.auth.SimpleAclAuthorizer --authorize
r-properties zookeeper.connect=nem1.open.tri.lan:2181/znodeG --remove --allow-principal User:consumerKafkaIn
vitado --operation Read --topic TopicPrivado
Are you sure you want to remove acls:
        User:consumerKafkaInvitado has Allow permission for operations: Read from hosts: *
 from resource Topic:TopicPrivado y/n?
y
Following is list of acls for resource: Topic:TopicPrivado
        User:consumerKafka has Allow permission for operations: Read from hosts: *
        User:producerKafka has Allow permission for operations: Write from hosts: *
```

En el momento en el que le son revocados los permisos de lectura, y aunque se encuentre consumiendo en ese mismo momento, el *consumerKafkaInvitado* se encuentra con un error de autorización sobre el *TopicPrivado*.

```
[root@NEM1 kafka-broker]# bin/kafka-console-consumer.sh --topic TopicPrivado --zookeeper nem1.open.tri.lan:2181/znodeG --security-protocol SASL_PLAI
[2017-10-04 15:35:14,691] WARN ZooKeeper event while creating registration node: /consumers NOAUTH (kafka.utils.ZKCheckedEphemeral)
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-59904, security.protocol=S
hola
este es el ultimo mensaje que vas a recibir, ya que te voy a cortar los permisos

[2017-10-04 15:36:13,172] ERROR [ConsumerFetcherThread-console-consumer-59904_NEM1.open.tri.lan-1507124114515-b61d83bf-0-1003], Error for partition
03:kafka.common.TopicAuthorizationException (kafka.consumer.ConsumerFetcherThread)
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-59904, security.protocol=S
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-59904, security.protocol=S
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-59904, security.protocol=S
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-59904, security.protocol=S
{metadata.broker.list=nem3.open.tri.lan:6667,nem2.open.tri.lan:6667, request.timeout.ms=30000, client.id=console-consumer-59904, security.protocol=S
```