_ общее 1

30 ноября 2020 г. 22:23

The need for encryption, authentication & authorization in Kafka

- Currently, any client can access your Kafka cluster (authentication)
- The clients can publish / consume any topic data (authorisation)
- All the data being sent is fully visible on the network (encryption)

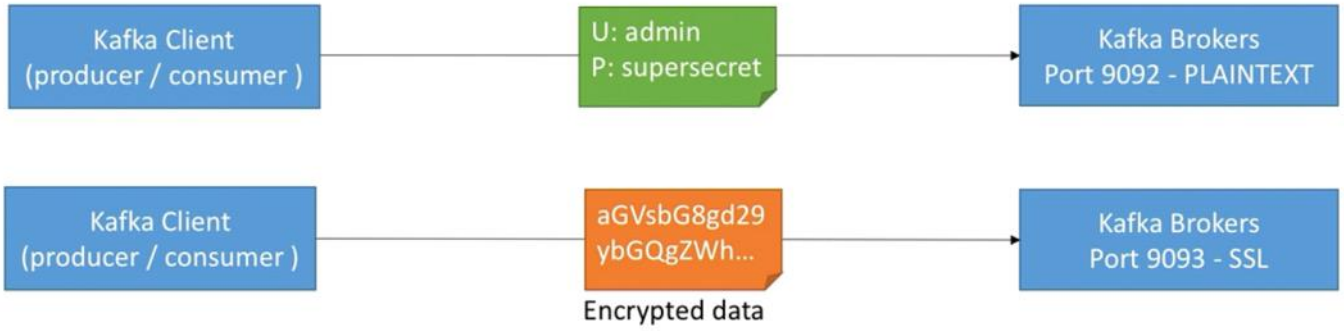
- Someone could intercept data being sent
- Someone could publish bad data / steal data
- Someone could delete topics

- All these reasons push for more security and an authentication model

f

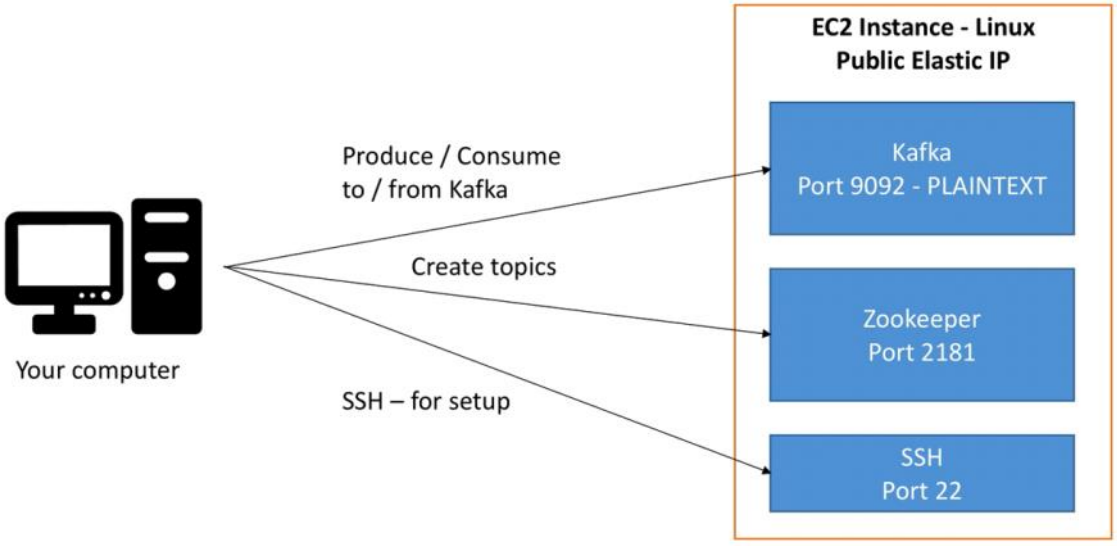
Encryption in Kafka

- Encryption in Kafka ensures that the data exchanged between clients and brokers is secret to routers on the way
- This is similar concept to an https website



Authentication in Kafka

- Authentication in Kafka can take a few forms
- SSL Authentication: clients authenticate to Kafka using SSL certificates
- SASL Authentication:
  - PLAIN: clients authenticate using username / password (weak – easy to setup)
  - Kerberos: such as Microsoft Active Directory (strong – hard to setup)
  - SCRAM: username / password (strong – medium to setup)



SASL Simple Authentication and Security Layer

преимущество в том что он не должен менять протокол

- SASL stands for Simple Authentication and Security Layer
- It's becoming a standard in big data applications for security (Hadoop, Kafka, etc…)
- The reason it's popular is that it does not change the application protocol and theoretically one SASL ticket can give you access to many systems
- SASL is usually combined with SSL/TLS for adding an encryption layer

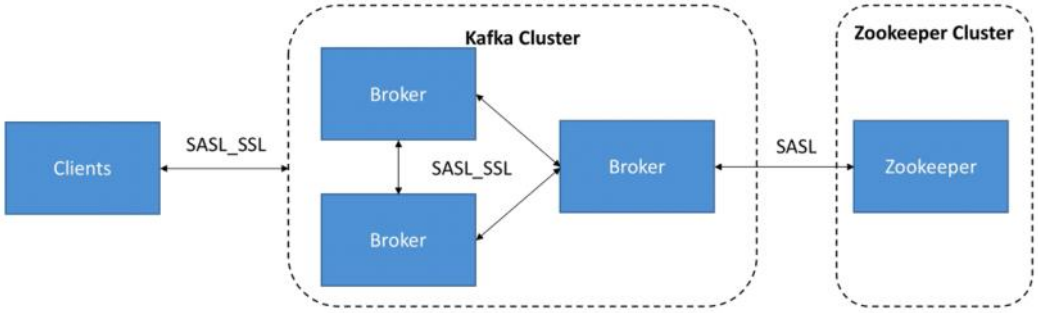
типы SASL в Kafka

- SASL in Kafka currently implements the following protocols:
  - PLAIN (simple username / password)
  - SCRAM (modern username / password with challenge)
  - GSSAPI (Kerberos authentication / Active Directory authentication)
- Kafka is slowly extending the SASL protocols it will support
  - OAUTHBEARER (WIP – KIP 255) for OAuth 2 support
  - + other custom mechanisms (WIP – KIP 86)

- **SASL/GSSAPI** is the most frequent one today and the most complicated – we will visit that one in this course

Cluster Security

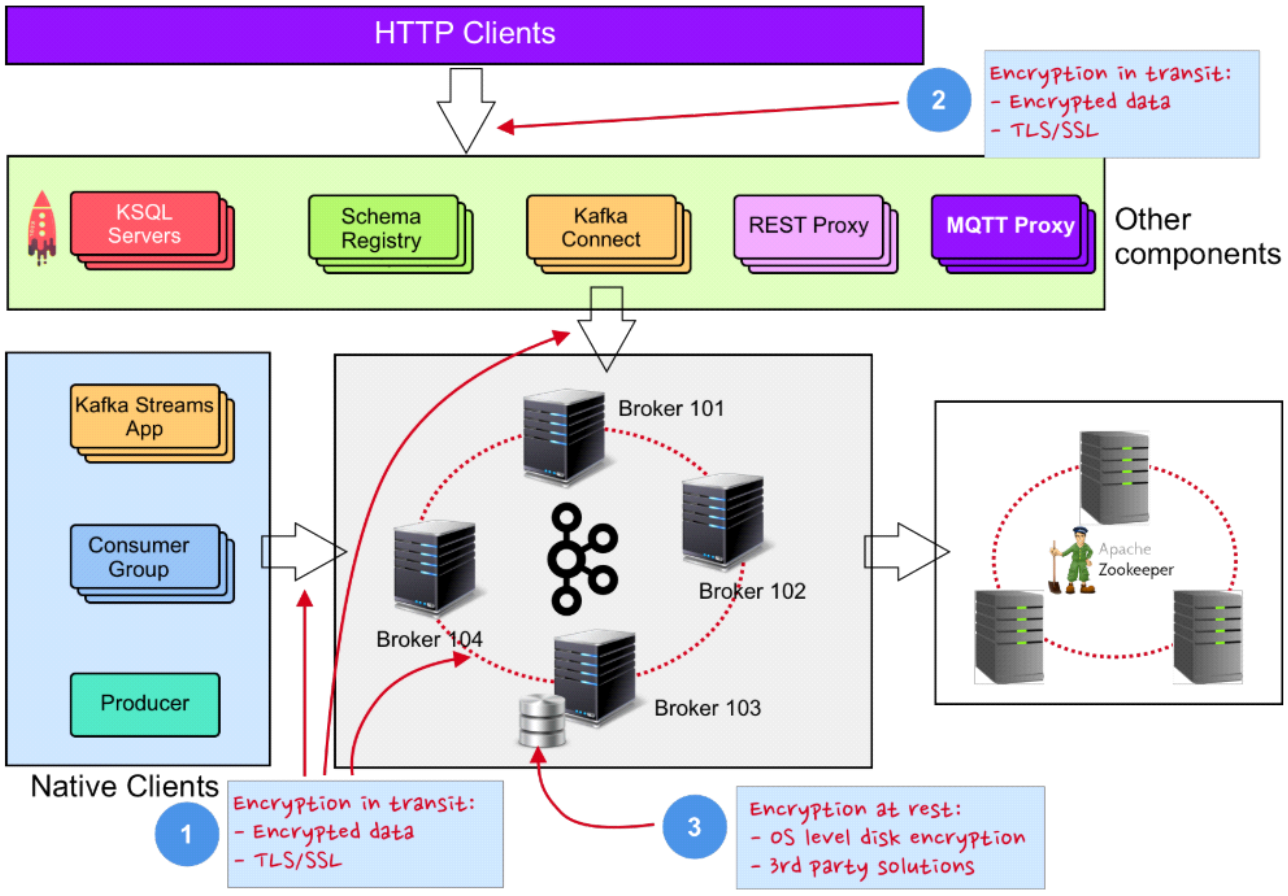
- Cluster security is an advanced topic.
- You can secure Broker to Broker connections using SASL / SSL
- You can secure Broker to Zookeeper connections using SASL



_ общее 2

13 декабря 2020 г. 14:18

Security - Encryption at Rest & in Transit

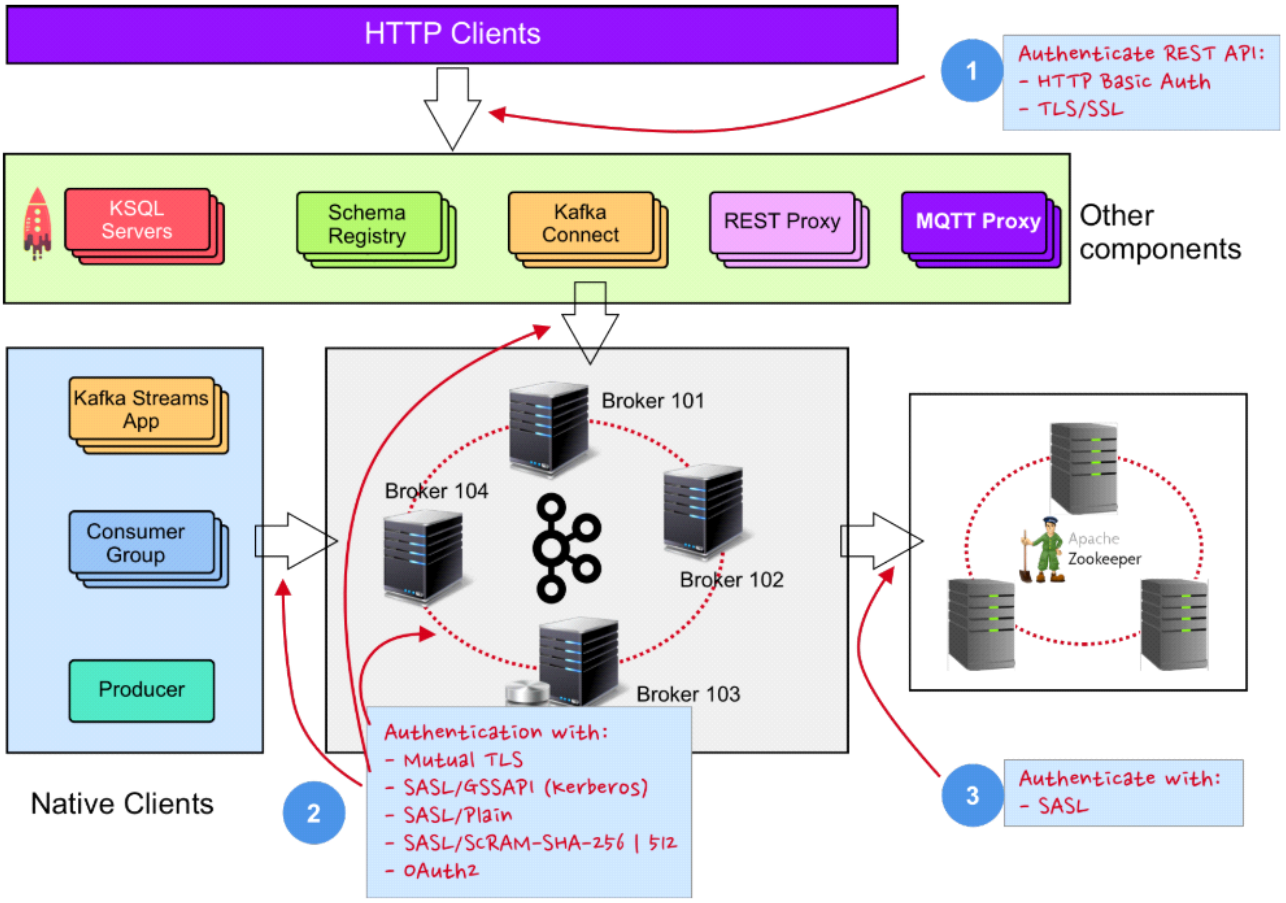


The Kafka cluster, including the ZooKeeper ensemble can be secured as follows:

1. Encryption **in-transit** (or **in-flight**) is mainly achieved in the following 2 ways:
   - the producer encrypts the data before sending it to the broker. The consumer decrypts this data
   - using (mutual) TLS

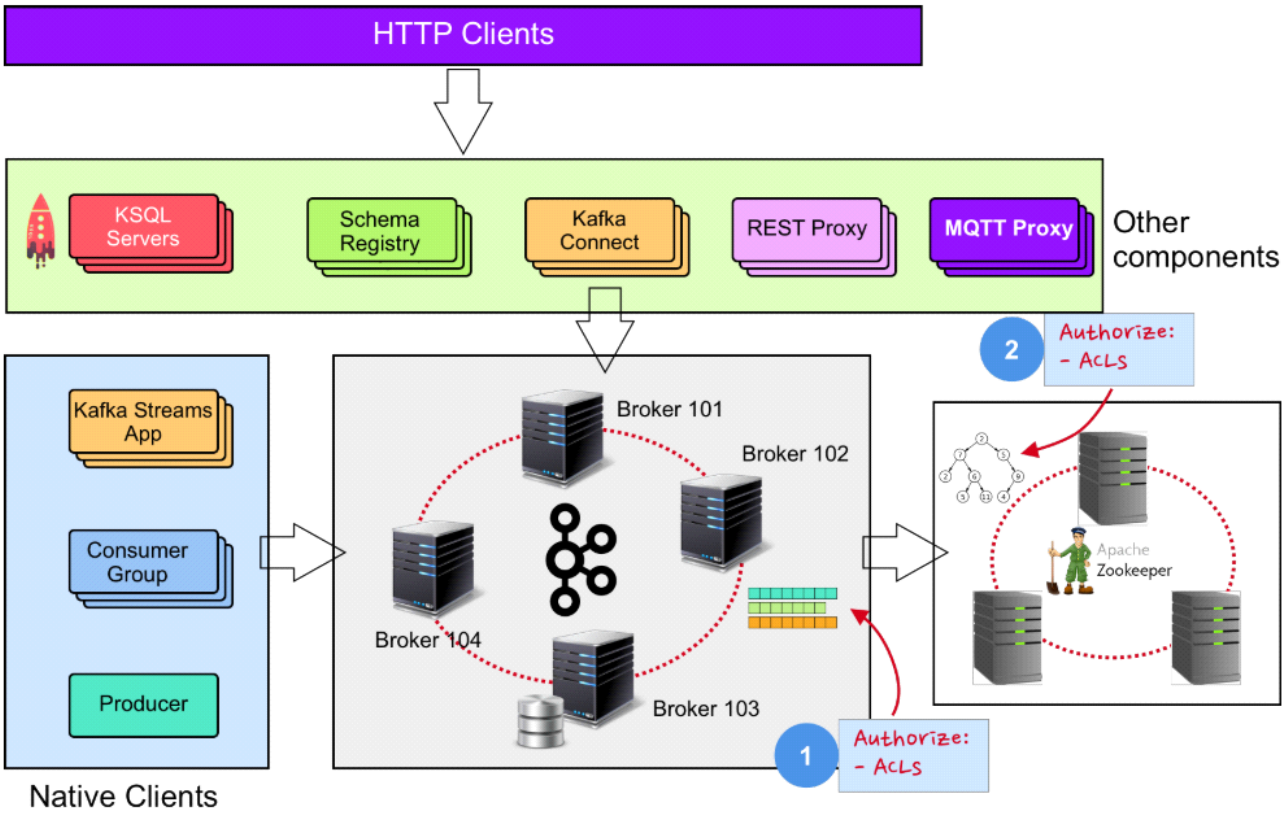
   The above applies for both situations:
   - client → broker
   - broker → broker
2. Encryption between HTTP clients and Confluent Platform services such as Schema Registry happens in the following ways
   - Using TLS/SSL
   - Using the Confluent Enterprise security plugin
3. Encryption at rest is implemented either by using OS level disk partition encryption or by using 3rd party services such as the Vorometric Data Security manager

Security - Authentication



1. HTTP Clients can authenticate themselves to the REST APIs of the Schema Registry, Kafka Connect Workers, REST Proxy or KSQL Server via:
    - HTTP Basic Auth
    - SASL
2. Intra-broker and client to broker authentication happens via:
    - **MTLS:** Mutual TLS/SSL
    - SASL/Plain:
    - SASL/GSSAPI (Kerberos):
    - SASL/SCRAM-SHA-256 and SASL/SCRAM-SHA-512:
    - OAuth2
3. Brokers authenticate themselves to the ZooKeeper ensemble via SASL

Security - Authorization



1. All relevant resources on the Kafka cluster, such as topics can be protected by Access Control Lists (ACLs). All clients accessing the Kafka cluster need the corresponding rights to execute `create`, `read` and/or `write` operations on those resources.

   > Impersonation is not possible at this time, that is, e.g. the identity of a client accessing the KSQL API is not passed through to the Kafka cluster, but rather the KSQL Server identifies itself (with its technical account) to the Kafka cluster.

2. All nodes containing meta data about the Kafka cluster, stored in ZooKeeper, can be protected by ACLs. The brokers need the corresponding rights to read and/or write from and to those nodes

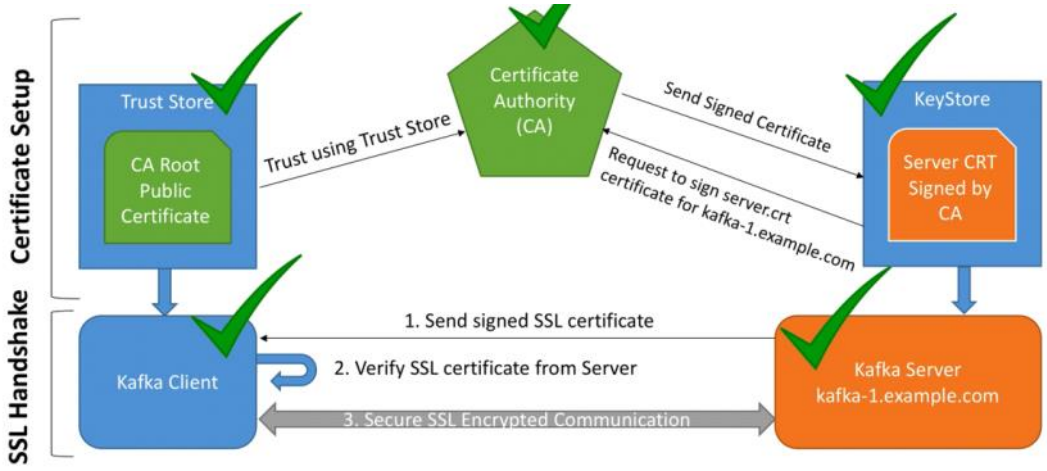
encryption: PLAINTEXT

30 ноября 2020 г. 22:26

.. encryption: SSL

30 ноября 2020 г. 22:27

- We'll setup a certificate authority
- We'll setup a broker certificate
- We'll sign a broker certificate
- We'll setup a key store for Kafka Broker
- We'll setup a trust store for Kafka Client
- We'll reboot Kafka Broker with SSL mode (Port 9093)
- We'll test our setup using a secure SSL producer and consumer



(1) setup CA authority

```
ubuntu@ec2-18-196-169-2:~/ssl$ openssl req -new -newkey rsa:4096 -days 365 -x509 -subj "/CN=Kafka-Security-CA" -ke
yout ca-key -out ca-cert -nodes
Generating a 4096 bit RSA private key
.....................................................++
.....................++
writing new private key to 'ca-key'
-----
ubuntu@ec2-18-196-169-2:~/ssl$ ll
total 16
drwxrwxr-x 2 ubuntu ubuntu 4096 Feb 15 11:16 ./
drwxr-xr-x 6 ubuntu ubuntu 4096 Feb 15 11:15 ../
-rw-rw-r-- 1 ubuntu ubuntu 1809 Feb 15 11:16 ca-cert
-rw-rw-r-- 1 ubuntu ubuntu 3272 Feb 15 11:16 ca-key
```

(2) установить SSL на кафку

- We will configure a keystore & truststore for Kafka broker
- Using the certificates we have generated, we will setup Kafka broker to use SSL on the Port 9093
- Once Kafka is rebooted, we will test the SSL certificates once again to make sure they are correctly applied

создадим keystore и заложим в него URL доменное имя кафки

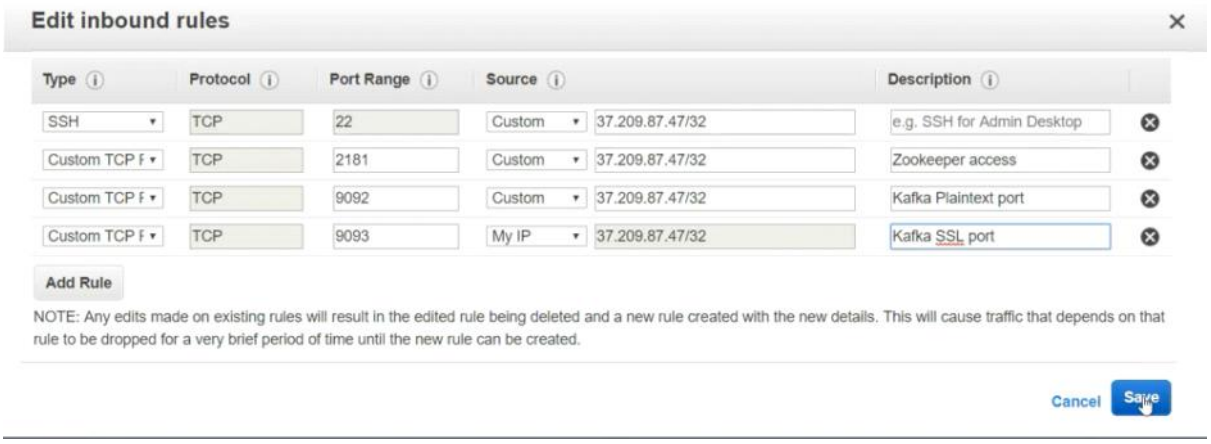
```
ubuntu@ec2-18-196-169-2:~/ssl$ keytool -genkey -keystore kafka.server.keystore.jks -validity 365 -storepass $SRVPA
SS -keypass $SRVPASS  -dname "CN=ec2-18-196-169-2.eu-central-1.compute.amazonaws.com" -storetype pkcs12
ubuntu@ec2-18-196-169-2:~/ssl$ ll
total 20
drwxrwxr-x 2 ubuntu ubuntu 4096 Feb 15 12:41 ./
drwxr-xr-x 6 ubuntu ubuntu 4096 Feb 15 12:29 ../
-rw-rw-r-- 1 ubuntu ubuntu 1809 Feb 15 11:16 ca-cert
-rw-rw-r-- 1 ubuntu ubuntu 3272 Feb 15 11:16 ca-key
-rw-rw-r-- 1 ubuntu ubuntu 2229 Feb 15 12:41 kafka.server.keystore.jks
```

```
ubuntu@ec2-18-196-169-2:~/ssl$ ll
total 20
drwxrwxr-x 2 ubuntu ubuntu 4096 Feb 15 12:41 ./
drwxr-xr-x 6 ubuntu ubuntu 4096 Feb 15 12:29 ../
-rw-rw-r-- 1 ubuntu ubuntu 1809 Feb 15 11:16 ca-cert
-rw-rw-r-- 1 ubuntu ubuntu 3272 Feb 15 11:16 ca-key
-rw-rw-r-- 1 ubuntu ubuntu 2229 Feb 15 12:41 kafka.server.keystore.jks
```

подпишем сертификат ключем из первого шага (на выходе получим cert-file)

```
ubuntu@ec2-18-196-169-2:~/ssl$ keytool -keystore kafka.server.keystore.jks -certreq -file cert-file -storepass $SR
VPASS -keypass $SRVPASS
ubuntu@ec2-18-196-169-2:~/ssl$ ll
total 24
drwxrwxr-x 2 ubuntu ubuntu 4096 Feb 15 12:47 ./
drwxr-xr-x 6 ubuntu ubuntu 4096 Feb 15 12:29 ../
-rw-rw-r-- 1 ubuntu ubuntu 1809 Feb 15 11:16 ca-cert
-rw-rw-r-- 1 ubuntu ubuntu 3272 Feb 15 11:16 ca-key
-rw-rw-r-- 1 ubuntu ubuntu 1517 Feb 15 12:47 cert-file
-rw-rw-r-- 1 ubuntu ubuntu 2229 Feb 15 12:41 kafka.server.keystore.jks
```

```
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# see kafka.server.KafkaConfig for additional details and defaults

############################ Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################ Socket Server Settings ############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://0.0.0.0:9092,SSL://0.0.0.0:9093
advertised.listeners=PLAINTEXT://ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9092,SSL://ec2-18-196-169-2.e
u-central-1.compute.amazonaws.com:9093
zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:2181

ssl.keystore.location=/home/ubuntu/ssl/kafka.server.keystore.jks
ssl.keystore.password=serversecret
ssl.key.password=serversecret
ssl.truststore.location=/home/ubuntu/ssl/kafka.server.truststore.jks
ssl.truststore.password=serversecret
```



клиентом(консьюмер, продьюсер) присоеденимся к кафке по TLS

```
gerd ~ $ openssl s_client -connect ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9093
```

скаачем на ноут сертификат с сервера

```
gerd ~ ssl $ scp -i ~/kafka-security.pem ubuntu@ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:/h
ome/ubuntu/ssl/ca-cert .
```

```
-rw-rw-r-- 1 gerd gerd 1809 Feb 18 12:21 ca-cert
gerd ~ ssl $ keytool -keystore kafka.client.truststore.jks -alias CARoot -import -file ca-cert -store
pass $CLIPASS -keypass $CLIPASS -noprompt
```

```
gerd ~ ssl $ keytool -list -v -keystore kafka.client.truststore.jks
```

```
gerd ~ ssl $ vi client.properties
```

```
security.protocol=SSL
ssl.truststore.location=/home/gerd/ssl/kafka.client.truststore.jks
ssl.truststore.password=clientpass
```

```
gerd ~ ssl $ ~/kafka/bin/kafka-console-producer.sh --broker-list ec2-18-196-169-2.eu-central-1.compute
.amazonaws.com:9093 --topic kafka-security-topic --producer.config ~/ssl/client.properties
```

authentication: SSL

30 ноября 2020 г. 22:29

- Currently, only the servers have certificates.
- These certificates are used to enable encryption

- With SSL, clients can also have certificates!

- If the certificate is validated by the broker, the client is authenticated and has an identity
- This is a very common authentication scheme in Kafka, mostly found around managed Kafka Solutions

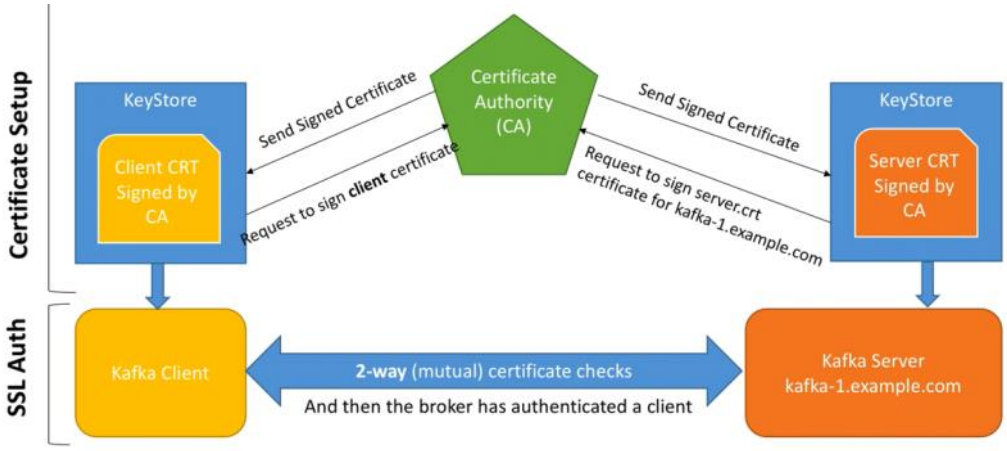
Аутентификация - это процесс подтверждения того, что клиент (человек или служба) является тем, кем он является

- (обычно с использованием имени пользователя и пароля), в то время как авторизация - это процесс разрешения или запрета пользователю доступа к определенному ресурс.

клиент имеет подписанный сертификат, потому что он не анонимный а имеет идентичность

- In the encryption case:
  - Only the brokers has signed servers certificates
  - The client were verifying the broker certificates to establish a SSL connection
  - The client is "anonymous" to the broker (no identity)

- In the authentication case:
  - The clients AND the brokers have signed server certificates
  - The client and the brokers verify each other's certificates
  - The client now has an IDENTITY to the broker (we can apply ACLs)



сгенерим сертификат для ноута

```
gerd ~ $   export CLIPASS=clientpass
gerd ~ $   cd ssl
gerd ~ ssl   $   keytool -genkey -keystore kafka.client.keystore.jks -validity 365 -storepass $CLIPASS -keypass
$CLIPASS  -dname "CN=mylaptop" -alias my-local-pc -storetype pkcs12
gerd ~ ssl   $   ll
total 12
drwxrwxrwx 0 gerd gerd  512 Feb 19 19:53 ./
drwxr-xr-x 0 gerd gerd  512 Feb 19 11:07 ../
-rw-rw-r-- 1 gerd gerd 1809 Feb 18 12:21 ca-cert
-rw-rw-rw- 1 gerd gerd  125 Feb 18 12:24 client.properties
-rw-rw-rw- 1 gerd gerd 2169 Feb 19 19:53 kafka.client.keystore.jks
-rw-rw-rw- 1 gerd gerd 1358 Feb 18 12:22 kafka.client.truststore.jks
```

подпишем(signing request) сертификат для ноута в CA

```
gerd ~ ssl   $   keytool -keystore kafka.client.keystore.jks -certreq -file client-cert-sign-request -alias my-
local-pc -storepass $CLIPASS -keypass $CLIPASS
```

```
gerd ~ ssl   $   scp -i ~/kafka-security.pem client-cert-sign-request ubuntu@ec2-18-196-169-2.eu-central-1.comp
ute.amazonaws.com:/tmp/
```

на CA сервере подпишем

```
gerd~$ ssh -i Downloads/kafka-security.pem ubuntu@ec2-18-196-169-2.eu-central-1.compute.amazonaws.com
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-1049-aws x86_64)
```

```
ubuntu@ip-172-31-26-230:~$ openssl x509 -req -CA ca-cert -CAkey ca-key -in /tmp/client-cert-sign-reque
st -out /tmp/client-cert-signed -days 365 -CAcreateserial -passin pass:serversecret
```

```
ubuntu@ip-172-31-26-230:~/ssl$ ll /tmp/client-cert-sign*
-rw-rw-r-- 1 ubuntu ubuntu 2069 Feb 19 19:10 /tmp/client-cert-signed
-rw-rw-r-- 1 ubuntu ubuntu 1455 Feb 19 18:57 /tmp/client-cert-sign-request
```

загрузим на ноут сертификат с CA

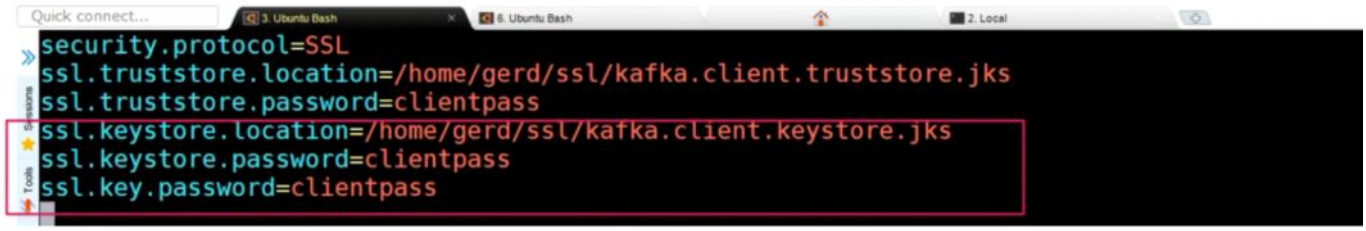
```
gerd ~ ssl   $   scp -i ~/kafka-security.pem ubuntu@ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:
/tmp/client-cert-signed  .
```

```
gerd ~ ssl   $   keytool -keystore kafka.client.keystore.jks -alias CARoot -import -file ca-cert -storep
ass $CLIPASS -keypass $CLIPASS -noprompt
```

```
gerd ~ ssl   $   keytool -keystore kafka.client.keystore.jks -import -file client-cert-signed -alias my-
local-pc -storepass $CLIPASS -keypass $CLIPASS -noprompt
Certificate reply was installed in keystore
```

в настройках кафки нужна одна строчка

```
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://0.0.0.0:9092,SSL://0.0.0.0:9093
advertised.listeners=PLAINTEXT://ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9092,SSL://ec2-18
-196-169-2.eu-central-1.compute.amazonaws.com:9093
zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:2181

ssl.keystore.location=/home/ubuntu/ssl/kafka.server.keystore.jks
ssl.keystore.password=serversecret
ssl.key.password=serversecret
ssl.truststore.location=/home/ubuntu/ssl/kafka.server.truststore.jks
ssl.truststore.password=serversecret

ssl.client.auth=required
```


Quick connect...
3. Ubuntu Bash
6. Ubuntu Bash
2. Local
security.protocol=SSL
ssl.truststore.location=/home/gerd/ssl/kafka.client.truststore.jks
ssl.truststore.password=clientpass
ssl.keystore.location=/home/gerd/ssl/kafka.client.keystore.jks
ssl.keystore.password=clientpass
ssl.key.password=clientpass
Sessions
Tools

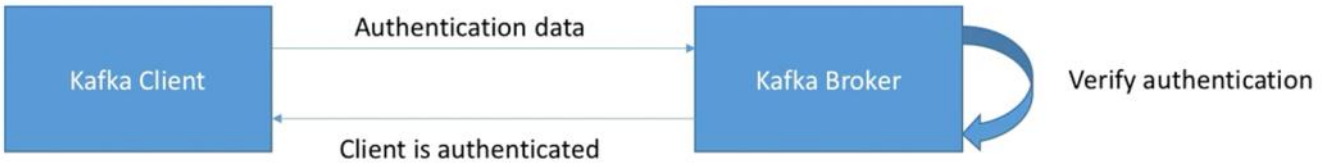

gerd ~ $
gerd ~ $ ~/kafka/bin/kafka-console-producer.sh --broker-list ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9093 --topic kafka-security-topic --producer.config ~/ssl/client-ssl-auth.properties

.. authentication: SASL: PLAIN (user/pass)

30 ноября 2020 г. 22:27

Authentication in Kafka

- Authentication in Kafka ensures that only clients that can prove their identity can connect to our Kafka Cluster
- This is similar concept to a login (username / password)



.. authentication: SASL: Kerberos (MS AD)

30 ноября 2020 г. 22:29

What is Kerberos?

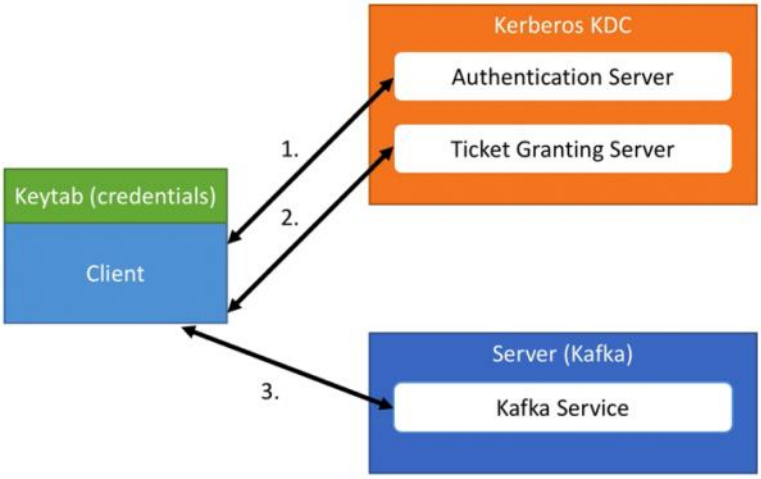
- A protocol for authentication over (unsecure) networks
- Data exchange is encrypted
- Based on tickets issued to *Service Principals* (also a user is a *service* in Kerberos)
- Involves a trusted 3rd-party (KDC, Key Distribution Center)
  - maintains the Service Principal Database
- Created at MIT

- Microsoft Active Directory is the most popular implementation of Kerberos
- The free implementation is the one provided by the MIT (this course)
- Other commercial implementations may exist

клиенту выдается временный тикет на доступ к кафке

Authentication Workflow

1. Client uses Keytab to authenticate to KDC
2. Client requests a new ticket to the Ticket Granting Server
3. Client access to *Kafka* via temporary ticket obtained in 2)

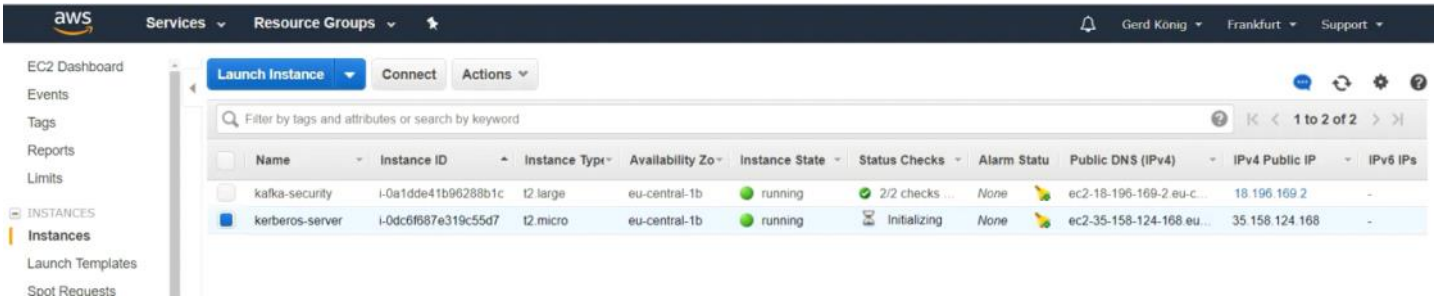


Key Characteristics

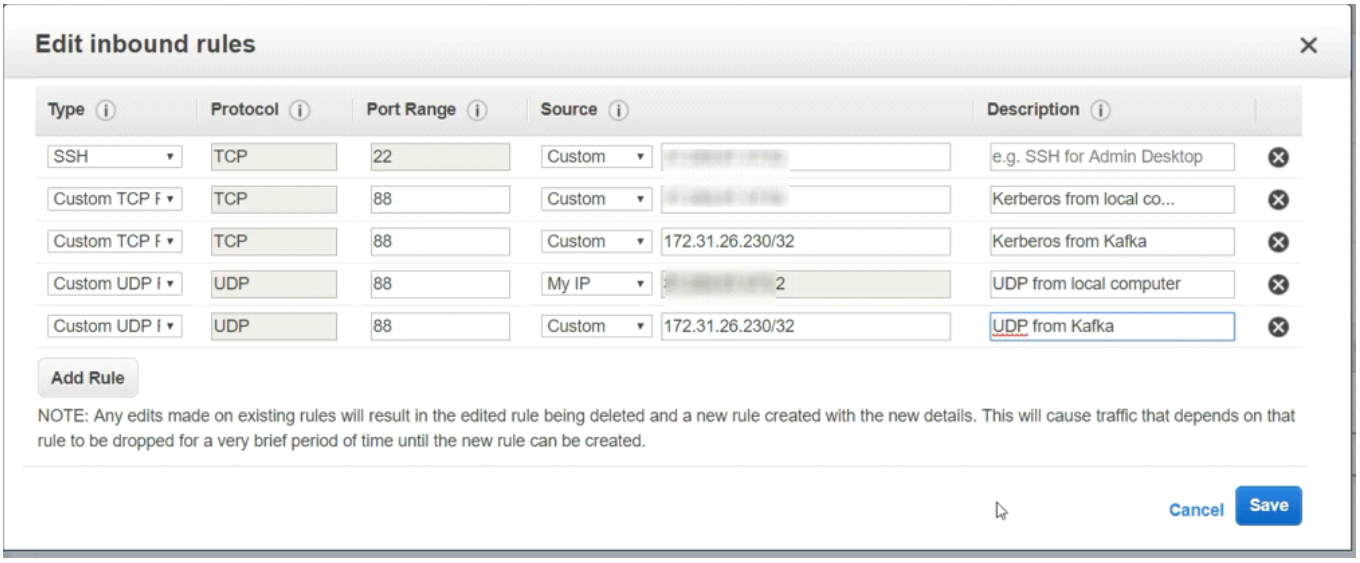
- All communication is encrypted
- No passwords are being sent over the wire
- Tickets do expire, hence keep your servers time in sync
- Your clients will automatically renew tickets as long as their credentials are valid

- Only clients interact with the KDC, the target service/server NEVER talks to the KDC

нужен отдельный kerberos server



d




```
[centos@ip-172-31-21-208 ~]$ sudo yum install -y krb5-server
```

```
[centos@ip-172-31-21-208 ~]$ sudo vi /var/kerberos/krb5kdc/kdc.conf
```

```
[kdcdefaults]
  kdc_ports = 88
  kdc_tcp_ports = 88
  default_realm=KAFKA.SECURE
[realms]
  KAFKA.SECURE = {
    acl_file = /var/kerberos/krb5kdc/kadm5.acl
    dict_file = /usr/share/dict/words
    admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
    supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfou
r-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-sha1:normal des-cbc-
md5:normal des-cbc-crc:normal
  }
```

```
[logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log

[libdefaults]
    default_realm = KAFKA.SECURE
    kdc_timesync = 1
    ticket_lifetime = 24h

[realms]
    KAFKA.SECURE = {
      admin_server = ec2-54-93-111-92.eu-central-1.compute.amazonaws.com
      kdc  = ec2-54-93-111-92.eu-central-1.compute.amazonaws.com
      }
```

```
[centos@ip-172-31-21-208 ~]$ sudo /usr/sbin/kdb5_util create -s -r KAFKA.SECURE -P this-i
s-unsecure
```

```
[centos@ip-172-31-21-208 ~]$ sudo systemctl restart krb5kdc
```

Setup the Kerberos environment

We will

- create Kerberos principals
- create keytabs
- test grabbing a Kerberos ticket

клиентом законектимся к kerberos-серверу

импортируем ключи

```
gerd ~ $ scp -i ~/kafka-security.pem centos@ec2-18-195-215-129.eu-central-1.compute.amazonaw
s.com:/tmp/*.keytab /tmp/
```

поставим на ноут kerberos-client packae

```
gerd ~ $   export DEBIAN_FRONTEND=noninteractive && sudo apt-get install -y krb5-user
```

```
[logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log

[libdefaults]
    default_realm = KAFKA.SECURE
    kdc_timesync = 1
    ticket_lifetime = 24h

[realms]
    KAFKA.SECURE = {
      admin_server = ec2-18-195-215-129.eu-central-1.compute.amazonaws.com
      kdc  = ec2-18-195-215-129.eu-central-1.compute.amazonaws.com
      }
```

проверим, заберем kerberos тикет

```
gerd ~ $   kinit -kt /tmp/admin.user.keytab admin
```

проверим что получили тикет

```
ubuntu@ip-172-31-26-230:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: kafka/ec2-18-196-169-2.eu-central-1.compute.amazonaws.com@KAFKA.SECURE

Valid starting       Expires              Service principal
02/22/2018 13:15:30  02/23/2018 13:15:28  krbtgt/KAFKA.SECURE@KAFKA.SECURE
```

настроим кафка -сервер

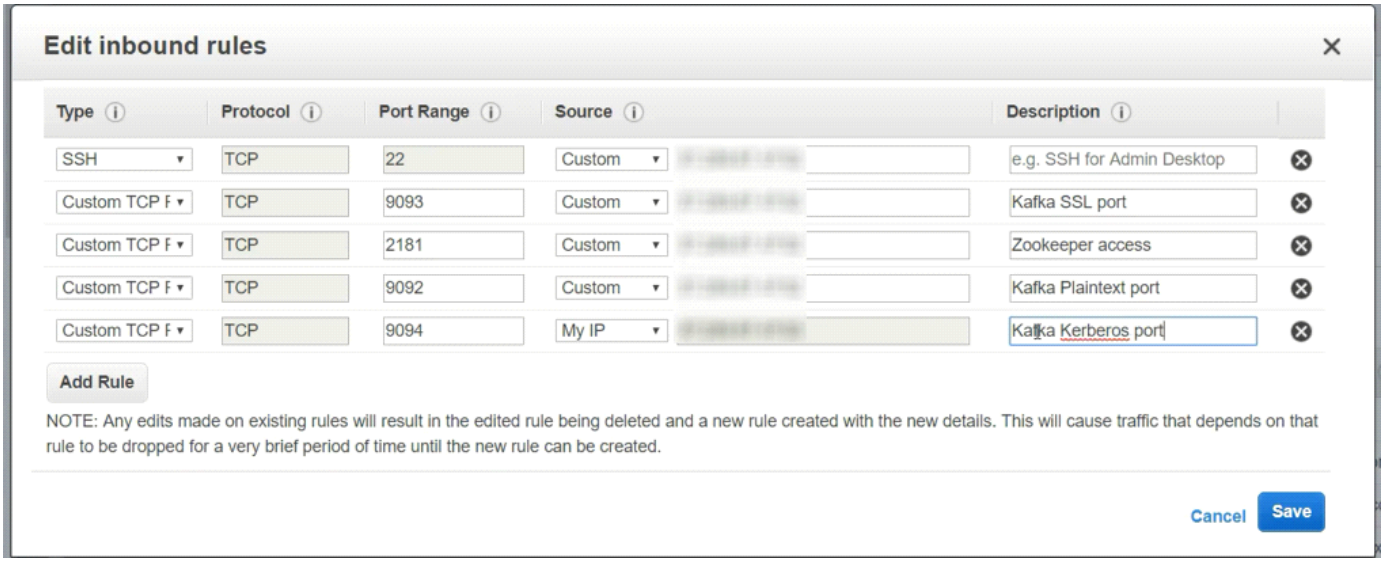
```
# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://0.0.0.0:9092,SSL://0.0.0.0:9093,SASL_SSL://0.0.0.0:9094
advertised.listeners=PLAINTEXT://ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9092,SSL://ec2-1
8-196-169-2.eu-central-1.compute.amazonaws.com:9093,SASL_SSL://ec2-18-196-169-2.eu-central-1.compute.
amazonaws.com:9094
zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:2181

sasl.enabled.mechanisms=GSSAPI
sasl.kerberos.service.name=kafka

ssl.keystore.location=/home/ubuntu/ssl/kafka.server.keystore.jks
ssl.keystore.password=serversecret
ssl.key.password=serversecret
ssl.truststore.location=/home/ubuntu/ssl/kafka.server.truststore.jks
-- INSERT --                                                    19,1          14%
```

конфигурация SASL AUTHENITICATION

```
ubuntu@ip-172-31-26-230:~$ sudo vi kafka/config/kafka_server_jaas.conf
```

```
KafkaServer {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/tmp/kafka.service.keytab"
    principal="kafka/ec2-18-196-169-2.eu-central-1.compute.amazonaws.com@KAFKA.SECURE";
};
```

```
[Unit]
Description=Apache Kafka server (broker)
Documentation=http://kafka.apache.org/documentation.html
Requires=zookeeper.service

[Service]
Type=simple
Environment="KAFKA_OPTS=-Djava.security.auth.login.config=/home/ubuntu/kafka/config/kafka_server_jaas
.conf"
ExecStart=/home/ubuntu/kafka/bin/kafka-server-start.sh /home/ubuntu/kafka/config/server.properties
ExecStop=/home/ubuntu/kafka/bin/kafka-server-stop.sh

[Install]
WantedBy=multi-user.target

ubuntu@ip-172-31-26-230:~$ sudo vi /etc/systemd/system/kafka.service
ubuntu@ip-172-31-26-230:~$
ubuntu@ip-172-31-26-230:~$
ubuntu@ip-172-31-26-230:~$ sudo systemctl daemon-reload
ubuntu@ip-172-31-26-230:~$
```



kafka consumer / producer settings

сначала нужно создать клиентский jaas файл , для SASL AUTHENTICATION

```
gerd ~ $   vi /tmp/kafka_client_jaas.conf
```

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

```
gerd ~ $   export KAFKA_OPTS="-Djava.security.auth.login.config=/tmp/kafka_client_jaas.conf"
```

обычный prop файл

```
gerd ~ $   vi /tmp/kafka_client_kerberos.properties
```

```
security.protocol=SASL_SSL
sasl.kerberos.service.name=kafka
ssl.truststore.location=/home/gerd/ssl/kafka.client.truststore.jks
ssl.truststore.password=clientpass
```

проверим что получили тикет

```
gerd ~ $ kdestroy
gerd ~ $   klist
```

```
gerd ~ $   klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: reader@KAFKA.SECURE

Valid starting       Expires              Service principal
02/22/2018 18:11:05  02/23/2018 18:11:05  krbtgt/KAFKA.SECURE@KAFKA.SECURE
```

проверим консьюмер/продьюсер

```
gerd ~ $   ~/kafka/bin/kafka-console-producer.sh --broker-list ec2-18-196-169-2.eu-central-1.comput
e.amazonaws.com:9094 --topic kafka-security-topic --producer.config /tmp/kafka_client_kerberos.propertie
s
```

.. authentication: SASL: OAuth

2 декабря 2020 г. 10:37

https://docs.confluent.io/platform/current/kafka/authentication_sasl/authentication_sasl_oauth.html

authorisation: ACL

30 ноября 2020 г. 22:31

Authorisation in Kafka

- Once a client is authenticated, Kafka can verify its identity
- It still needs to be combined with authorisation, so that Kafka knows that
  - "User alice can view topic finance"
  - "User bob cannot view topic trucks"
- ACL (Access Control Lists) have to be maintained by administration and onboard new users

What are ACLs?

- Now that our clients are authenticated to our brokers (using either SSL or SASL), we need to define permissions!
- ACLs (Access Control Lists) come in two flavours:
  - **Topics:** restrict which client can read / write data
  - **Consumer Groups:** which client can use a specific consumer group
  - **Cluster:** which client can create / delete topics or apply settings
- Finally, there is a concept of "Super User" in Kafka that can do everything without any special kind of ACL
- There is no concept of User Groups so far in Kafka. Each ACL have to be written for each client / user

Where are ACLs stored?

- ACLs are stored in Zookeeper, and added through the usage of the command line.
- Therefore, you need to absolutely restrict who can access your Zookeeper Cluster (through security or network rules)
- **Only Kafka admins should have the right to create topics, and create ACLs.**

настроим кафка сервер на поддержку ACL

```
ubuntu@ip-172-31-26-230:~$ sudo vi kafka/config/server.properties
```

```
# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://0.0.0.0:9092,SSL://0.0.0.0:9093,SASL_SSL://0.0.0.0:9094
advertised.listeners=PLAINTEXT://ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9092,
SSL://ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9093,SASL_SSL://ec2-18-196-169-2
.eu-central-1.compute.amazonaws.com:9094
zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:2181

authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
super.users=User:admin;User:kafka
allow.everyone.if.no.acl.found=false
security.inter.broker.protocol=SASL_SSL

sasl.enabled.mechanisms=GSSAPI
sasl.kerberos.service.name=kafka

ssl.keystore.location=/home/ubuntu/ssl/kafka.server.keystore.jks
ssl.keystore.password=serversecret
ssl.key.password=serversecret
ssl.truststore.location=/home/ubuntu/ssl/kafka.server.truststore.jks
-- INSERT --                                                    36,1          18%
```

Authorization

Sample scenario

- Scenario to demonstrate granted and denied access to Kafka
- Users: *reader*, *writer*, *admin*
- Topic: *acl-test*
- Who should be able to do what ?
  - user *reader*: consume from topic acl-test
  - user *writer*: produce to and read from acl-test
  - user *admin*: all privileges to topic acl-test

Authorization

Sample scenario

- Scenario to demonstrate granted and denied access to Kafka
- Users: *reader*, *writer*, *admin*
- Topic: *acl-test*
- Who should be able to do what ?
  - user *reader*: consume from topic acl-test
  - user *writer*: produce to and read from acl-test
  - user *admin*: all privileges to topic acl-test

для юзеров reader и writer дадим права на чтение(operation Read) к конкретному топику

```
gerd ~ $   ~/kafka/bin/kafka-acls.sh \
> --authorizer-properties zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amaz
onaws.com:2181 --add \
> --allow-principal User:reader --allow-principal User:writer \
> --operation Read \
> --group=* \
> --topic acl-test
Adding ACLs for resource `Topic:acl-test`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *

Adding ACLs for resource `Group:*`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *

Current ACLs for resource `Topic:acl-test`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *

Current ACLs for resource `Group:*`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *
```

для юзеров reader и writer дадим права на запись (operation Write) к конкретному топику

```
gerd ~ $   ~/kafka/bin/kafka-acls.sh \
> --authorizer-properties zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amaz
onaws.com:2181 --add \
> --allow-principal User:writer \
> --operation Write \
> --topic acl-test
Adding ACLs for resource `Topic:acl-test`:
        User:writer has Allow permission for operations: Write from hosts: *

Current ACLs for resource `Topic:acl-test`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Write from hosts: *
```

посмотрим получившиеся права на топике

```
gerd ~ $   ~/kafka/bin/kafka-acls.sh \
> --authorizer-properties zookeeper.connect=ec2-18-196-169-2.eu-central-1.compute.amaz
onaws.com:2181 \
> --list \
> --topic acl-test

Current ACLs for resource `Topic:acl-test`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Write from hosts: *
```

```
Current ACLs for resource `Topic:acl-test`:
        User:reader has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Read from hosts: *
        User:writer has Allow permission for operations: Write from hosts: *
```

как смотреть секьюрити лог

```
ubuntu@ip-172-31-26-230:~$ ll kafka/logs/kafka-authorizer.log
-rw-r--r-- 1 root root 324 Feb 23 17:05 kafka/logs/kafka-authorizer.log
ubuntu@ip-172-31-26-230:~$ cat kafka/logs/kafka-authorizer.log
[2018-02-23 17:05:33,470] INFO Principal = User:reader is Denied Operation = Read from ho
st = 37.209.87.47 on resource = Topic:acl-test (kafka.authorizer.logger)
[2018-02-23 17:05:33,509] INFO Principal = User:reader is Denied Operation = Read from ho
st = 37.209.87.47 on resource = Topic:acl-test (kafka.authorizer.logger)
```

но для этого должен быть настроен log4j

```
log4j.logger.kafka.request.logger=WARN, requestAppender
log4j.additivity.kafka.request.logger=false

# Uncomment the lines below and change log4j.logger.kafka.network.RequestChannel$ to TRAC
E for additional output
# related to the handling of requests
log4j.logger.kafka.network.Processor=INFO, requestAppender
log4j.logger.kafka.server.KafkaApis=INFO, requestAppender
#log4j.additivity.kafka.server.KafkaApis=false
log4j.logger.kafka.network.RequestChannel$=WARN, requestAppender
log4j.additivity.kafka.network.RequestChannel$=false

log4j.logger.kafka.controller=INFO, controllerAppender
log4j.additivity.kafka.controller=false

log4j.logger.kafka.log.LogCleaner=INFO, cleanerAppender
log4j.additivity.kafka.log.LogCleaner=false

log4j.logger.state.change.logger=WARN, stateChangeAppender
log4j.additivity.state.change.logger=false

# Access denials are logged at INFO level, change to DEBUG to also log allowed accesses
log4j.logger.kafka.authorizer.logger=INFO, authorizerAppender
log4j.additivity.kafka.authorizer.logger=false
```

настроим консьюмер/продьюсер

```
gerd ~ $   export KAFKA_OPTS="-Djava.security.auth.login.config=/tmp/kafka_clien
t_jaas.conf"
gerd ~ $   kdestroy
gerd ~ $   klist
gerd ~ $   kinit -kt /tmp/writer.user.keytab writer
gerd ~ $   klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: writer@KAFKA.SECURE

Valid starting       Expires              Service principal
02/23/2018 17:57:17  02/24/2018 17:57:17  krbtgt/KAFKA.SECURE@KAFKA.SECURE
```

```
gerd ~ $   ~/kafka/bin/kafka-console-producer.sh \
> --broker-list ec2-18-196-169-2.eu-central-1.compute.amazonaws.com:9094 \
> --topic acl-test \
> --producer.config /tmp/kafka_client_kerberos.properties
```

в команде kinit указываем юзера для которого получаем токен

properties config files

1 декабря 2020 г. 19:57

- Best support for Kafka Security for applications is with **Java**.
- This course solely focuses on showing and demonstrating examples with the command line (+ properties config files)

ZOOKEEPER SECURITY

2 декабря 2020 г. 13:34

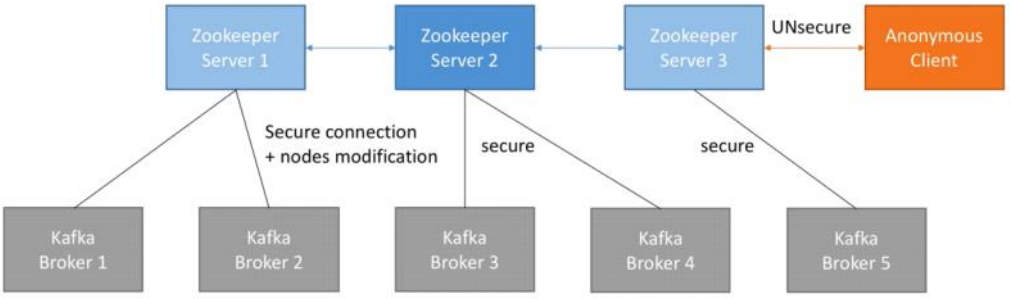
Why Securing Zookeeper?

- By default, in Zookeeper any user that's connected has all the priviledges
- The Kafka brokers store metadata in Zookeeper. For example:
  - Topic list: /brokers/topics
  - Broker list /brokers/ids
  - Topic Configurations /config/topics
  - Kafka ACL (Access Control List) /kafka-acl
- It's important to secure Zookeeper to prevent unauthorized users from modifying the metadata in Zookeeper (especially the Kafka ACL).

Zookeeper Security Models

- Zookeeper has two security models:
  - Default: anyone can read, write data etc…
  - Secure: authenticated users have rights to do certain operations
- Zookeeper has the following ACL Schemes (authentication):
  - "world": no authentication (default)
  - "digest": username / password
  - "sasl": Kerberos for example

Zookeeper Security Diagram



Note on Zookeeper Nodes security

- Zookeeper has its own ACL for each node.
- Each node has ACL in the form of cdwra:
  - Create: create a child node
  - Read: read data from node and list its children
  - Write: set data for a node
  - Delete: delete a child node
  - Admin: set permissions
- We want Kafka to have "cdwra" on the Kafka nodes and the world (anyone else) to have "r"

What are we going to secure?

- We need to enable Security on every Zookeeper node
- We need to have each Kafka broker use the same security credentials to Zookeeper
- We need Zookeeper to use the "secure znode" by default
- *Note: It's a lot of work. An simpler security model is to use network rules to only allow Kafka to talk to Zookeeper.*

Zookeeper Security

Summary | Key takeaways

- Zookeeper configuration
  - JAAS file section for ZK connection called `Server`
  - adjust handling of Kerberos principal
    - `kerberos.removeHostFromPrincipal`
    - `kerberos.removeRealmFromPrincipal`
- Kafka configuration
  - JAAS file section for ZK connection called `Client`
  - enforce setting ACLs in ZK (`zookeeper.set.acl=true`)
  - use the same Kerberos **principal name** throughout **ALL** brokers
- Zookeeper SuperUser comes to your rescue
- Use `zookeeper-security-migration` tool to transform non-secured topic znodes into secured ones

настроим ZK сервер (kerberos)

```
ubuntu@ip-172-31-28-118:~$ sudo vi ~/kafka/config/zookeeper.properties
Windowed Mode
dataDir=/home/ubuntu/zookeeper
clientPort=2181
maxClientCnxns=0

authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000
```

```
ubuntu@ip-172-31-28-118:~$ sudo vi ~/kafka/config/zookeeper_jaas.conf
Server {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
storeKey=true
keyTab="/tmp/zookeeper.service.keytab"
principal="zookeeper/ip-172-31-28-118.eu-central-1.compute.internal@KAFKA.SECURE";
};
```

```
ubuntu@ip-172-31-28-118:~$ sudo vi ~/kafka/config/kafka_server_jaas.conf
5 SecsForward (by keyframe) / 00:02:58(40%)
KafkaServer {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/tmp/kafka.service.keytab"
    principal="kafka/ec2-18-197-48-199.eu-central-1.compute.amazonaws.com@KAFKA.SECURE";
};

Client {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/tmp/zookeeper.service.keytab"
    principal="zookeeper/ip-172-31-28-118.eu-central-1.compute.internal@KAFKA.SECURE";
};
```

```
ubuntu@ip-172-31-28-118:~$ sudo vi /etc/systemd/system/zookeeper.service
[Unit]
Description=Apache Zookeeper server
Documentation=http://zookeeper.apache.org
Requires=network.target remote-fs.target
After=network.target remote-fs.target

[Service]
Type=simple
Environment="KAFKA_OPTS=-Djava.security.auth.login.config=/home/ubuntu/kafka/config/zookeeper_jaas.conf"
ExecStart=/home/ubuntu/kafka/bin/zookeeper-server-start.sh /home/ubuntu/kafka/config/zookeeper.properties
ExecStop=/home/ubuntu/kafka/bin/zookeeper-server-stop.sh

[Install]
WantedBy=multi-user.target
```

кодключимся к ZK клиентом

```
Windowed Mode
ubuntu@ip-172-31-28-118:~$ export KAFKA_OPTS=-Djava.security.auth.login.config=/home/ubuntu/kafka/config/kafka_server_jaas.conf
ubuntu@ip-172-31-28-118:~$ ~/kafka/bin/zookeeper-shell.sh ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:2181
Connecting to ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:2181
Welcome to ZooKeeper!
JLine support is disabled

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

WATCHER::

WatchedEvent state:SaslAuthenticated type:None path:null
```

подключимся к zk секторно

```
3. kafka-node
ubuntu@ip-172-31-28-118:~$ export KAFKA_OPTS=""
ubuntu@ip-172-31-28-118:~$ kafka/bin/zookeeper-shell.sh ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:2181
Connecting to ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:2181
Welcome to ZooKeeper!
JLine support is disabled

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
```

подключимся к zk анонимно

настроим кафку

```
Windowed Mode
# see the License for the specific language governing permissions and
# limitations under the License.

# see kafka.server.KafkaConfig for additional details and defaults

############################ Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################ Socket Server Settings ############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
listeners=PLAINTEXT://0.0.0.0:9092,SSL://0.0.0.0:9093,SASL_SSL://0.0.0.0:9094
advertised.listeners=PLAINTEXT://ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:9092,SSL://ec2-18-197-48-199.eu-central-
.compute.amazonaws.com:9093,SASL_SSL://ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:9094
zookeeper.connect=ec2-18-197-48-199.eu-central-1.compute.amazonaws.com:2181

authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
super.users=User:admin;User:kafka
allow.everyone.if.no.acl.found=false
security.inter.broker.protocol=SASL_SSL

zookeeper.set.acl=true
sasl.enabled.mechanisms=GSSAPI
sasl.kerberos.service.name=kafka

ssl.keystore.location=/home/ubuntu/ssl/kafka.server.keystore.jks
ssl.keystore.password=serversecret
-- INSERT --                                                    40,23         10%
```

```
ubuntu@ip-172-31-28-118:~$ sudo vi kafka/config/zookeeper.properties
```

```
dataDir=/home/ubuntu/zookeeper
clientPort=2181
maxClientCnxns=0

authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000

kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true

~
~
```