Kafka Settings

How to setup Apache Kafka

Using CP-Ansible to deploy a cluster with SASL PLAIN security

1. Clone the CP-Ansible repository to the host you wish to deploy from:

```
git clone git@github.com:confluentinc/cp-ansible.git
```

2. Copy `hosts_example.yml` to `hosts.yml`:

```
cp hosts_example.yml hosts.yml
```

3. Edit `hosts.yml` using the editor of your choice. In this instance, we are using `vi`:

```
vi hosts.yml
```

4. Set the `ansible-user:` property to be a user who has sudo access on the hosts you wish to deploy to:

```
all:
  vars:
     ansible_connection: ssh
     ansible_user:
     ansible_become: true
```

5. Uncommment this variable to enable `sasl_plain` and make sure it aligns with the `all` group:

```
 # sasl_protocol: plain
```

6. Under each component name, modify the `hosts:` section to reflect the hostnames or IP addresses of the hosts you wish to deploy to:

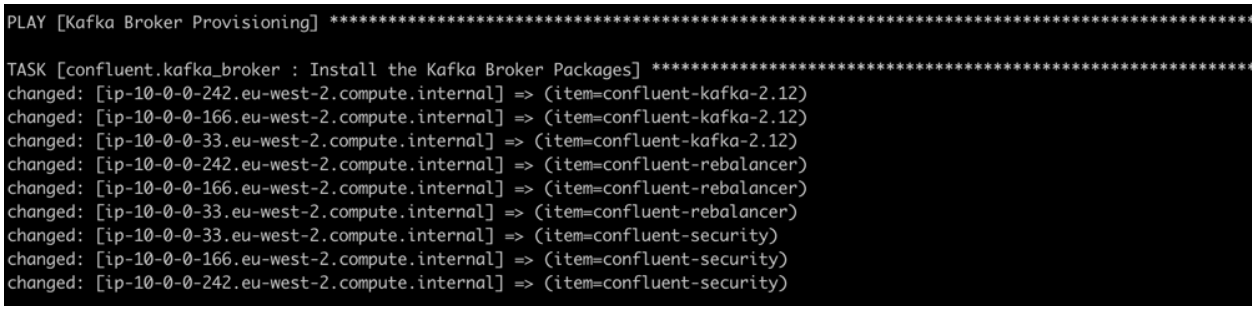
```
zookeeper:
     hosts:
           Ip-10-0-0-34.eu-west-2.compute.internal:
```

7. Save your edits to the `hosts.yml` file.

8. Execute the following command:

```
ansible-playbook --private-key=  -i hosts.yml all.yml
```

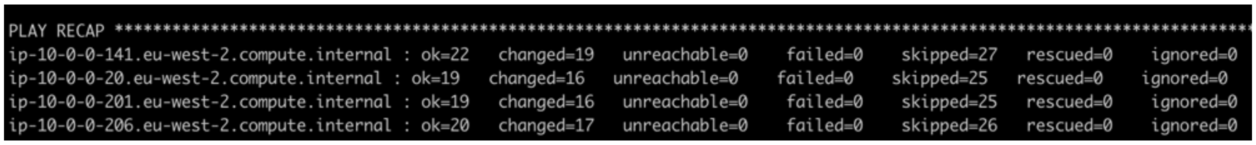
9. You will now see the playbooks execute:



```
PLAY [Kafka Broker Provisioning] ***************************************************************************

TASK [confluent.kafka_broker : Install the Kafka Broker Packages] ******************************************
changed: [ip-10-0-0-242.eu-west-2.compute.internal] => (item=confluent-kafka-2.12)
changed: [ip-10-0-0-166.eu-west-2.compute.internal] => (item=confluent-kafka-2.12)
changed: [ip-10-0-0-33.eu-west-2.compute.internal] => (item=confluent-kafka-2.12)
changed: [ip-10-0-0-242.eu-west-2.compute.internal] => (item=confluent-rebalancer)
changed: [ip-10-0-0-166.eu-west-2.compute.internal] => (item=confluent-rebalancer)
changed: [ip-10-0-0-33.eu-west-2.compute.internal] => (item=confluent-rebalancer)
changed: [ip-10-0-0-33.eu-west-2.compute.internal] => (item=confluent-security)
changed: [ip-10-0-0-166.eu-west-2.compute.internal] => (item=confluent-security)
changed: [ip-10-0-0-242.eu-west-2.compute.internal] => (item=confluent-security)
```

10. When completed successfully, you should see output similar to the following:

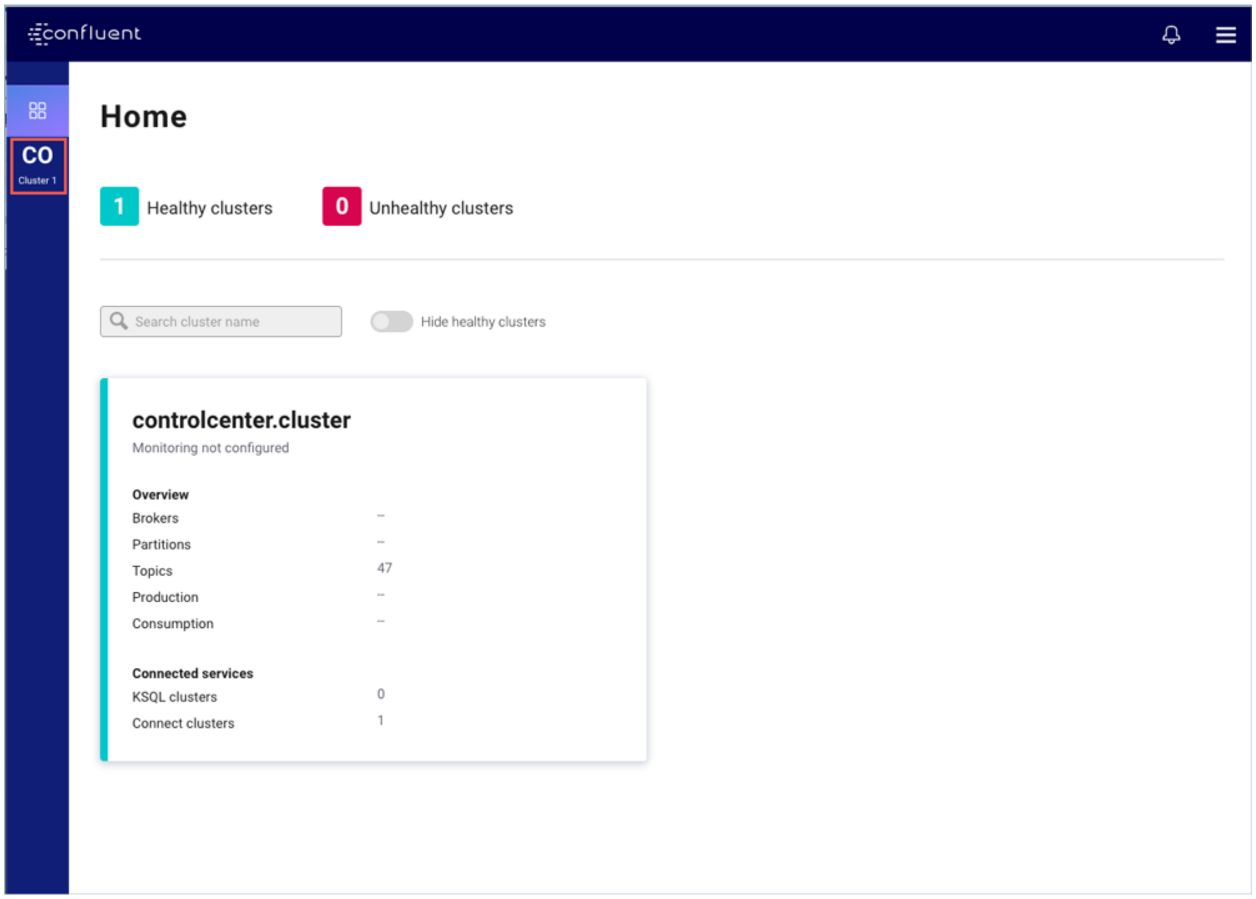


```
PLAY RECAP *************************************************************************************************
ip-10-0-0-141.eu-west-2.compute.internal : ok=22   changed=19   unreachable=0    failed=0    skipped=27   rescued=0    ignored=0
ip-10-0-0-20.eu-west-2.compute.internal : ok=19   changed=16   unreachable=0    failed=0    skipped=25   rescued=0    ignored=0
ip-10-0-0-201.eu-west-2.compute.internal : ok=19   changed=16   unreachable=0    failed=0    skipped=25   rescued=0    ignored=0
ip-10-0-0-206.eu-west-2.compute.internal : ok=20   changed=17   unreachable=0    failed=0    skipped=26   rescued=0    ignored=0
```

Validate installation

To validate your installation, access the Confluent Control Center interface by opening your web browser and navigate to the host and port where Control Center is running. Control Center runs at `http://localhost:9021/` by default.

When you first open Control Center, the homepage for your clusters appears. The homepage provides a global overview of all clusters managed by the Control Center instance. Each cluster tile displays its running status, Kafka overview statistics, and connected services.



controlcenter.cluster
Monitoring not configured

Overview	
Brokers	–
Partitions	–
Topics	47
Production	–
Consumption	–
Connected services	
KSQL clusters	0
Connect clusters	1

You should see something similar to the image above showing one healthy cluster and a list of components. You can drill into individual clusters by clicking on the cluster name for more details.

The Easiest Way to Install Apache Kafka and Confluent Platform - With Ansible

With Confluent Platform 5.3, we are actively embracing the rising DevOps movement by introducing CP-Ansible, our very own open source Ansible playbooks for deployment of Apache Kafka® and the Confluent [...]

https://www.confluent.io/blog/confluent-platform-installation-with-cp-ansible/

Documentation

Add security to running cluster

Adding SASL security to ZooKeeper

1. In `zookeeper.properties`, add the authentication provider to enable ZooKeeper security:

   ```
   authProvider.sasl=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
   ```

2. Export the ZooKeeper JAAS file before restarting ZooKeeper:

   ```
   KAFKA_OPTS=-Djava.security.auth.login.config=<path>/zookeeper_server_jaas.conf
   ```

   The content of `zookeeper_server_jaas.conf` should look like the following:

   ```
   Server {
      org.apache.zookeeper.server.auth.DigestLoginModule required
      user_super="adminsecret"
      user_bob="bobsecret";
   };
   ```

3. Perform ZooKeeper rolling restarts with the JAAS login file shown above; this enables brokers to authenticate.

4. Create the JAAS file for the brokers to authenticate with ZooKeeper. Add the Client information to the broker JAAS configuration and then export the JAAS file:

   ```
   export KAFKA_OPTS=-Djava.security.auth.login.config=<path>/kafka_server_jaas.conf
   ```

   The content of the broker JAAS file (`kafka_server_jaas.conf`) should look like the following:

   ```
   Client {
      org.apache.zookeeper.server.auth.DigestLoginModule required
      username="bob"
      password="bobsecret";
   };
   ```

5. Perform a rolling restart of brokers, this time setting the configuration parameter `zookeeper.set.acl` to true, which enables the use of secure ACLs when creating znodes.
6. Execute the ZkSecurityMigrator tool using the script: `bin/zookeeper-security-migration` with `zookeeper.acl` set to `secure`. This tool traverses the corresponding sub-trees, changing the ACLs of the znodes.

If you wish to validate that security has been enabled between the broker and ZooKeeper:

1. Export the broker JAAS configuration:

```
export KAFKA_OPTS=-Djava.security.auth.login.config=<path>/kafka_server_jaas.conf
```

2. Create a new topic named `test` using the `kafka-topic` command:

```
bin/kafka-topics --bootstrap-server localhost:9092 --create --topic test --partitions 2 --rep
lication-factor 2
```

3. Log in to the `zookeeper-shell` and check the ACL of the newly-created znode, which should have the ACL enabled:

```
bin/zookeeper-shell <zk_host>:<zk_port>

[zk: localhost:12181(CONNECTED) 9] getAcl /config/topics/test
'world,'anyone
: r
'sasl,'bob
: cdrwa
```

If you want to turn off authentication in a secure cluster:

1. Perform a rolling restart of brokers setting the JAAS login file, which enables brokers to authenticate, but setting `zookeeper.set.acl` to `false`. At the end of the rolling restart, brokers stop creating znodes with secure ACLs, but are still able to authenticate and manipulate all znodes.
2. Run the ZkSecurityMigrator tool using the script `bin/zookeeper-security-migration` with `zookeeper.acl` set to `unsecure`. This tool traverses the corresponding sub-trees, changing the ACLs of the znodes.
3. Perform a second rolling restart of brokers, this time omitting the system property that sets the JAAS login file.

Run this command to see the full list of parameters:

`bin/zookeeper-security-migration --help`

Configure TLS encryption for SASL security

1. In `zookeeper.properties`, add the authentication provider to enable ZooKeeper security:

   ```
   authProvider.sasl=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
   ```

   Optionally, to enable TLS encryption, specify the following configurations (be sure to use camel case for `keyStore` and `trustStore`). You will have both a `clientPort` and a `secureClientPort` at this point:

   ```
   authProvider.x509=org.apache.zookeeper.server.auth.X509AuthenticationProvider
   secureClientPort=2182
   serverCnxnFactory=org.apache.zookeeper.server.NettyServerCnxnFactory
   ssl.keyStore.location=<path-to-zookeeper-keystore>
   ssl.keyStore.password=<zookeeper-keystore-password>
   ssl.trustStore.location=<path-to-zookeeper-truststore>
   ssl.trustStore.password=<zookeeper-truststore-password>
   ssl.clientAuth=none
   ```

2. Export the ZooKeeper JAAS file before restarting ZooKeeper:

   ```
   KAFKA_OPTS=-Djava.security.auth.login.config=<path>/zookeeper_server_jaas.conf
   ```

   The content of `zookeeper_server_jaas.conf` should look like the following:

   ```
   Server {
      org.apache.zookeeper.server.auth.DigestLoginModule required
      user_super="adminsecret"
      user_bob="bobsecret";
   };
   ```

3. Perform ZooKeeper rolling restarts with the JAAS login file shown above; this enables brokers to authenticate.

4. Create the JAAS file for the brokers to authenticate with ZooKeeper. Add the Client information to the broker JAAS configuration and then export the JAAS file:

   ```
   export KAFKA_OPTS=-Djava.security.auth.login.config=<path-to-kafka_server_jaas.conf>
   ```

The content of the broker JAAS file (`kafka_server_jaas.conf`) should look like the following:

```
Client {
   org.apache.zookeeper.server.auth.DigestLoginModule required
   username="bob"
   password="bobsecret";
};
```

Optionally, if enabling TLS encryption to ZooKeeper, add these broker configurations (do not use camel case in `truststore`):

```
# Connect to the ZooKeeper port configured for TLS
zookeeper.connect=zk1:2182,zk2:2182,zk3:2182
# Required to use TLS to ZooKeeper (default is false)
zookeeper.ssl.client.enable=true
# Required to use TLS to ZooKeeper
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
# Define trust store to use TLS to ZooKeeper; ignored unless zookeeper.ssl.client.enable=true
zookeeper.ssl.truststore.location=<path-to-kafka-truststore>
zookeeper.ssl.truststore.password=<kafka-truststore-password>
```

1. Perform a rolling restart of brokers, this time setting `zookeeper.set.acl=true`, which enables the use of secure ACLs when creating znodes.
2. Execute the ZkSecurityMigrator tool using the scriptv `bin/zookeeper-security-migration` with `zookeeper.acl` set to `secure`. This tool traverses the corresponding sub-trees, changing the ACLs of the znodes.

Optionally, if enabling TLS encryption to ZooKeeper, connect to the TLS-encrypted port on ZooKeeper (for example, 2182) and specify `--zk-tls-config-file <path-to-tls-configs>` (note the double-dash) with this content for the file (do not use camel case in `truststore`):

```
zookeeper.ssl.client.enable=true
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
zookeeper.ssl.truststore.location=<path-to-kafka-truststore>
zookeeper.ssl.truststore.password=<kafka-truststore-password>
```

To validate that security has been enabled between the broker and ZooKeeper:

1. Export the broker JAAS configuration:

```
export KAFKA_OPTS=-Djava.security.auth.login.config=<path-to-kafka_server_jaas.conf>
```

2. Create a new topic named `test`:

```
bin/kafka-topics --bootstrap-server <kafka-host>:<kafka-port> --create --topic test --partiti
ons 2 --replication-factor 2
```

3. Log in to the `zookeeper-shell` and check the ACL of the newly-created znode, which should have the ACL enabled:

```
bin/zookeeper-shell <zk_host>:<zk_port>

[zk: localhost:12181(CONNECTED) 9] getAcl /config/topics/test
'world,'anyone
: r
'sasl,'bob
: cdrwa
```

Optionally, if enabling TLS encryption to ZooKeeper you can connect to the TLS port with `zookeeper-shell` by including the command line flags `-zk-tls-config-file <path-to-tls-configs>` (use a single dash) with this content for the file (do not use camel case in `truststore`):

```
zookeeper.ssl.client.enable=true
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
zookeeper.ssl.truststore.location=<path-to-kafka-truststore>
zookeeper.ssl.truststore.password=<kafka-truststore-password>
```

If you have enabled TLS encryption, then you can now remove the `clientPort` configuration in ZooKeeper.

Adding mTLS security to ZooKeeper

Enable SASL and/or mTLS authentication on ZooKeeper. If enabling mTLS, you would have both a non-TLS port and a TLS port, as shown here:

```
clientPort**=**2181
secureClientPort**=**2182
serverCnxnFactory**=**org.apache.zookeeper.server.NettyServerCnxnFactory
authProvider.x509**=**org.apache.zookeeper.server.auth.X509AuthenticationProvider
ssl.keyStore.location**=**<path-to-zookeeper-keystore>
ssl.keyStore.password**=**<zookeeper-keystore-password>
ssl.trustStore.location**=**<path-to-zookeeper-truststore>
ssl.trustStore.password**=**<zookeeper-truststore-password>
```

1. Perform a rolling restart of brokers.

Specify the JAAS login file and/or defining ZooKeeper mTLS configurations (including connections to the TLS-enabled ZooKeeper port) as required. This enables brokers to authenticate to ZooKeeper.

At the end of the rolling restart, brokers can manipulate znodes with strict ACLs, but will not create znodes with those ACLs. Note that unlike ZooKeeper, Kafka does not use camel case names for TLS-related configurations (for example, Kafka uses `zookeeper.ssl.keystore.location`, while ZooKeeper uses `ssl.keyStore.location`.

```
# Connect to the ZooKeeper port configured for TLS
zookeeper.connect=zk1:2182,zk2:2182,zk3:2182
# Required to use TLS to ZooKeeper (default is false)
zookeeper.ssl.client.enable=true
# Required to use TLS to ZooKeeper
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
# Define key/trust stores to use TLS to ZooKeeper; ignored unless zookeeper.ssl.client.enable
=true
zookeeper.ssl.keystore.location=<path-to-kafka-keystore>
zookeeper.ssl.keystore.password=<kafka-keystore-password>
zookeeper.ssl.truststore.location=<path-to-kafka-truststore>
zookeeper.ssl.truststore.password=<kafka-truststore-password>
```

2. If you enabled mTLS, disable the non-TLS port in ZooKeeper:

```
secureClientPort=2182
serverCnxnFactory=org.apache.zookeeper.server.NettyServerCnxnFactory
authProvider.x509=org.apache.zookeeper.server.auth.X509AuthenticationProvider
ssl.keyStore.location=<path-to-zookeeper-keystore>
ssl.keyStore.password=<zookeeper-keystore-password>
ssl.trustStore.location=<path-to-zookeeper-truststore>
ssl.trustStore.password=<zookeeper-truststore-password>
```

3. Perform a second rolling restart of brokers, this time setting `zookeeper.set.acl=true`, which enables the use of secure ACLs when creating znodes.

```
# Connect to the ZooKeeper port configured for TLS
zookeeper.connect=zk1:2182,zk2:2182,zk3:2182
# Required to use TLS to ZooKeeper (default is false)
zookeeper.ssl.client.enable=true
# Required to use TLS to ZooKeeper
zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
# Define key/trust stores to use TLS to ZooKeeper; ignored unless zookeeper.ssl.client.enable
=true
zookeeper.ssl.keystore.location=<path-to-kafka-keystore>
zookeeper.ssl.keystore.password=<kafka-keystore-password>
zookeeper.ssl.truststore.location=<path-to-kafka-truststore>
zookeeper.ssl.truststore.password=<kafka-truststore-password>
```

```
# Tell broker to create ACLs on znodes
zookeeper.set.acl=true
```

4. Run the ZkSecurityMigrator tool using the script: `bin/zookeeper-security-migration` with `zookeeper.acl` set to `secure`. This tool traverses the corresponding sub-trees, changing the ACLs of the znodes. Because you are enabling mTLS, specify the `-zk-tls-config-file <file>` option.

   ```
   ./bin/zookeeper-security-migration.sh --zookeeper.acl=secure --zookeeper.connect=localhost:21
   82 --zk-tls-config-file <path-to-tls-config-file.properties>
   ```

   The TLS configuration should look like this:

   ```
   zookeeper.ssl.client.enable=true
   zookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
   zookeeper.ssl.truststore.location=<path-to-kafka-truststore>
   zookeeper.ssl.truststore.password=<kafka-truststore-password>
   zookeeper.ssl.keystore.password-<encrypted-keystore-password>
   ```

To turn off mTLS authentication in this secure cluster:

1. Perform a rolling restart of brokers setting the JAAS login file and/or defining ZooKeeper mTLS configurations, which enables brokers to authenticate, but setting `zookeeper.set.acl` to `false`. At the end of the rolling restart, brokers stop creating znodes with secure ACLs, but are still able to authenticate and manipulate all znodes.
2. Run the ZkSecurityMigrator tool using the script `bin/zookeeper-security-migration` with `zookeeper.acl` set to `unsecure`. This tool traverses the corresponding sub-trees, changing the ACLs of the znodes. Specify `-zk-tls-config-file <file>` if you need to set TLS configuration.

   ```
   --zookeeper.connect=localhost:2182
   ```

3. If you are disabling mTLS, enable the non-TLS port in ZooKeeper:

   ```
   clientPort=2181
   secureClientPort=2182
   serverCnxnFactory=org.apache.zookeeper.server.NettyServerCnxnFactory
   authProvider.x509=org.apache.zookeeper.server.auth.X509AuthenticationProvider
   ssl.keyStore.location=<path-to-zookeeper-keystore>
   ssl.keyStore.password=<zookeeper-keystore-password>
   ```

```
ssl.trustStore.location=<path-to-zookeeper-truststore>
ssl.trustStore.password=<zookeeper-truststore-password>
```

4. Perform a second rolling restart of brokers, this time omitting the system property that sets the JAAS login file and/or removes ZooKeeper mTLS configuration (including connection to the non-TLS-enabled ZooKeeper port) as required.
5. If you are disabling mTLS, disable the TLS port in Kafka:
   `clientPort = 2181`

Adding security to a running cluster | Confluent Documentation
Use the Cloud quick start to get up and running with Confluent Cloud using a basic cluster
https://docs.confluent.io/platform/current/kafka/incremental-security-upgrade.html#adding-security-to-a-running-zk-cluster

Kafka Connect Security Basics

Check the following link to redirect you in the step you are currently in

Kafka Connect Security Basics | Confluent Documentation
If you have enabled SSL encryption in your Apache Kafka® cluster, then you must make sure that Kafka Connect is also configured for security. Click on the section to configure encryption in Kafka Connect: If you have enabled authentication in your Kafka cluster, then you must make sure that Kafka Connect is also configured for security.
https://docs.confluent.io/platform/current/connect/security.html#kconnect-long-security-basics

Kafka Consumer Configurations for Confluent Platform

Check the following link for all the settings values and meaning

Kafka Consumer Configurations for Confluent Platform | Confluent Documentation
Video courses covering Apache Kafka basics, advanced concepts, setup and use cases, and everything in between.
https://docs.confluent.io/platform/current/installation/configuration/consumer-configs.html#ak-consumer-configurations-for-cp

Kafka Producer Configurations for Confluent Platform

Check the following link for all the settings values and meaning

Kafka Producer Configurations for Confluent Platform | Confluent Documentation
Video courses covering Apache Kafka basics, advanced concepts, setup and use cases, and everything in between.
https://docs.confluent.io/platform/current/installation/configuration/producer-configs.html#ak-producer-configurations-for-cp

ZooKeeper Security

When authenticating brokers with ZooKeeper

- Set `zookeeper.set.acl=true` for all brokers.
    - If accept default (zookeper.set.acl=false), no ACLs are created and the info in ZooKeeper will be writeable by everyone.

When using mTLS alone

- Every broker and/or CLI tool must identify itself using the same Distinguished Name (DN).
    - DN included in the ZooKeeper ACL and will only authorize that DN, all connections to ZooKeeper must provide the DN.
    - If want to choose what gets put into the ACL from the DN, write and deploy a custom authentication provider (https://docs.confluent.io/platform/current/security/zk-security.html#zk-mtls).
- Each CA certificate should use the same DN
- Specify a different Subject Alternative Name (SAN)

Enable ZooKeeper authentication with SASL

Ass the following to zookeper.properties

```
authProvider.sasl=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
```

💡 Before starting ZooKeeper, check the JAAS syntax and keytab permissions. The most common errors that prevent the server from starting are JAAS syntax errors or permissions set incorrectly on the keytab file. Refer to Encrypting communication to ZooKeeper with TLS
for details.

mTLS authentication

💡 If using mTLS only (without SASL) and specifying `zookeeper.set.acl=true`, do not use certificates with `CN=hostname` where hostname differs based on the location from which the request originates as a means to satisfy hostname verification. If you do, you may find that brokers cannot access ZooKeeper nodes. Note that the full DN is included in the ZooKeeper ACL, and ZooKeeper only authorizes what is in the ACL.

- To illustrate the DN and SAN requirement, consider the scenario where you browse the web site `https://meeting.bigdata.us`.
- The CA certificate for the site must include the DN `CN=meeting.bigdata.us` (or a wildcard certificate with `CN=*.bigdata.us`)
  - If not, then your browser will reject the connection due to a failed hostname verification.
  - If a client makes an mTLS connection to ZooKeeper, and your client hostname is `foo.example.org`.
    - Present a certificate with DN `CN=foo.example.org` (or a wildcard certificate using `CN=*.example.org`).
      - If you don't present a certificate with this DN, then ZooKeeper will reject the connection unless you also define a Subject Alternative Name (SAN) containing `foo.example.org`
      - In the absence of SASL authentication, you can have different brokers and CLI tools connect to ZooKeeper with mTLS from different hosts only if they all use the same DN, in which case they must also specify a SAN that matches their respective hostnames.
- As an alternative to using the DN, you can specify the identity of mTLS clients by writing a class that extends `org.apache.zookeeper.server.auth.X509AuthenticationProvider` and overrides the method `protected String getClientId(X509Certificate clientCert)`
- Choose a scheme name and set `authProvider.[scheme]` in ZooKeeper to be the fully-qualified class name of the custom implementation.
  - Then configure `ssl.authProvider=[scheme]` to use it.

ZooKeeper Security | Confluent Documentation

When using SASL and mTLS authentication simultaneously with ZooKeeper, the SASL identity and either the DN that created the znode (the creating broker's CA certificate) or the DN of the security migration tool (if migration was performed after the znode was created) are included in ACLs.

https://docs.confluent.io/platform/current/security/zk-security.html#zk-security