1. Generate and Sign a Client Certificate Using the Certificate Authority (CA)

```
cd ~/certs
keytool -keystore client.keystore.jks -alias kafkauser -validity 365 -
genkey -keyalg RSA -dname "CN=kafka"
keytool -keystore client.keystore.jks -alias kafkauser -certreq -file
client-cert-file
openssl x509 -req -CA ca-cert -CAkey ca-key -in client-cert-file -out
client-cert-signed -days 365 -CAcre
keytool -keystore client.keystore.jks -alias CARoot -import -file ca-cert
keytool -keystore client.keystore.jks -alias kafkauser -import -file
client-cert-signed
```

- `keytool -genkey`: This generates a new key pair and stores it in a keystore (`client.keystore.jks`). The key is for the Kafka client, with the alias `kafkauser`.
- `-validity 365`: Sets the certificate validity period to 365 days.
- `-keyalg RSA`: Specifies the key algorithm (RSA) to use.
- `keytool -certreq`: Generates a certificate signing request (CSR) for the key (`client-cert-file`).
- `openssl x509 -req`: Signs the CSR using the CA's private key (`ca-key`) and generates a client certificate (`client-cert-signed`).
- `keytool -import`: The CA certificate (`ca-cert`) and the signed client certificate (`client-cert-signed`) are imported into the keystore.+++

First command : In this command we are goining to create new client.keystore.jks in which certificate and private key CN is refferd as name of user which has to be mentioned in certificate. RSA is ky algorithm used to encryption

Second command : in this command we are creating requsting file of certificate as client-cert-file

Third command : we are requestng to that reqsting file to CA (certificate authority) to sign certificate request file by using ca-key and give outputfile as clint-cert-signed

Fourth and fifth cmmand : In this command we are gooing to import ca-cert and client-cert-signed to client.keystore.jks

At this stage, the client keystore (`client.keystore.jks`) contains the private key for the Kafka client and the signed client certificate.

2. Move the Client Keystore to an Appropriate Location

```
sudo cp client.keystore.jks /var/private/ssl/
sudo chown root:root /var/private/ssl/client.keystore.jks
```

The keystore containing the client certificate (`client.keystore.jks`) is moved to a secure directory `/var/private/ssl/`, which ensures it's only accessible to root.

`chown root:root`: Changes the owner of the keystore to `root`, further restricting access.

```
sudo vi /etc/kafka/server.properties
```

Open the `server.properties` file on each Kafka broker. This file contains the server's configuration.

4. Configure SSL Client Authentication in

```
sudo vi /etc/kafka/server.properties
```

Open the `server.properties` file on each Kafka broker. This file contains the server's configuration

`server.properties``ssl.client.auth=required`

`ssl.client.auth=required`: This forces the Kafka brokers to require clients to authenticate using SSL/TLS certificates. Without this line, client authentication via certificates would not be mandatory.

5. Restart Kafka and Check its Status

```
sudo systemctl restart confluent-kafka
sudo systemctl status confluent-kafka
```

`systemctl restart confluent-kafka`: Restarts the Kafka service to apply the changes made in the `server.properties` file.

`systemctl status confluent-kafka`: Checks whether Kafka restarted successfully and is running correctly.

6. Attempt to Connect a Client to the Secure Port

At this point, try to connect a Kafka client without configuring SSL. This step is to ensure that the client authentication is enforced.

The connection attempt should **fail** because the client is not yet configured to use the client certificate.

This verifies that the server is correctly requiring SSL client certificates.

7. Configure Client Authentication with Client Certificate

```
Cd ~
vi client-ssl.properties
```

set these values in file ↓

```
ssl.keystore.location=/var/private/ssl/client.keystore.jks
ssl.keystore.password=<client keystore password>
ssl.key.password=<client key password>
```

As we have created client-ssl.properties in TLS we just change directory to home where client-ssl.properties file is located and then we configure it

`ssl.keystore.location`: Specifies the location of the client keystore containing the client certificate.

`ssl.keystore.password`: The password for the client keystore.

`ssl.key.password`: The password for the private key within the keystore.

This configuration tells Kafka that the client should use SSL/TLS for authentication using its client certificate.

8. Test the Connection with the Configured Client

```
kafka-console-consumer --bootstrap-server zoo1:9093 --topic tls-test --
from-beginning --consumer.config client-ssl.properties
```

This command runs a Kafka console consumer to connect to the `tls-test` topic using the secure broker (port 9093).

The `--consumer.config` option points to the `client-ssl.properties` file that you configured in the previous step.