1. **Create Test Data (TLS Test Topic):**

```
kafka-topics --bootstrap-server localhost:9092 --create --topic tls-test --
partitions 1 --replication-factor 3
```

Produce some sample data to this topic:

```
kafka-console-producer --broker-list localhost:9092 --topic tls-test
```

```
On those above commands the first command will creates topic tls-test of
partition 1 and replicatioin factor value of 3
And second command will cerates produceer to publish the values to topic we
have just created tls-test
```

2. **Generate Certificate Files:**
   -

```
cd ~/
mkdir certs
cd certs
```

Generate a Certificate Authority (CA**)**: This step involves creating a root CA certificate that will be used to sign other certificates.

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days 365 -subj
"/C=US/ST=Texas/L=Keller/O=Linux Academy"
```

```
The above command will used generate certificate of CERTIFICATE AUTHORITY
it self to trust creation between both client and server that above cmmand
generates ca-cert and ca-key this key will be used to sign the other
certificates of brokers (in this case zoo1,zoo2,zoo3)
```

3. **Set Up Truststores for Clients and Servers:**

The truststore will hold the public key of the CA so that both servers and clients can verify certificates.Import the CA certificate into both client and server truststores:

```
keytool -keystore client.truststore.jks -alias CARoot -import -file ca-cert
keytool -keystore server.truststore.jks -alias CARoot -import -file ca-cert
```

```
The above cmmand will cerates truststres for both client and server and
then c ommand import used to import ca-cert certificate of CA-AUTHORITY TO
create trust between both clint and server
```

4. **Generate Broker Certificates:**

Each Kafka broker needs its own certificate to establish secure connections.

Create keys and certificates for all brokers using the CA. This example shows certificates for three brokers.

```
keytool -keystore zoo1.keystore.jks -alias localhost -validity 365 -genkey
-keyalg RSA -dname "CN=broker1, OU=Unknown, O=Unknown, L=Unknown,
ST=Unknown, C=Unknown" -ext"SAN=dns:zoo3,dns:localhost,ip:127.0.0.1,ip:172.31.97.202"

keytool -keystore zoo2.keystore.jks -alias localhost -validity 365 -genkey
-keyalg RSA -dname "CN=broker2, OU=Unknown, O=Unknown, L=Unknown,
ST=Unknown, C=Unknown" -ext "san=dns:zoo3,dns:localhost,ip:127.0.0.1,ip:172.31.97.202"

keytool -keystore zoo3.keystore.jks -alias localhost -validity 365 -genkey
-keyalg RSA -dname "CN=broker3OU=Unknown, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown" -ext "san=dns:zoo3,dns:localhost,ip:127.0.0.1,ip:172.31.97.202"
```

```
these commands will beusd to create keystores of every broker in kafka
cluster using keytool command as zoo1,zoo2,zoo3.keystores.jks and validity
command specifies the validity of certificate and genkey command will
generates private key and certificate  with encryption algorithm RSA
and we specifies the distingush the identity of broker
```

5. **Export and Sign Broker Certificates:**
   - Each broker certificate must be signed by the CA.

Export the certificate requests for each broker:

```
keytool -keystore zoo1.keystore.jks -alias localhost -certreq -file zoo1-
cert-file
keytool -keystore zoo2.keystore.jks -alias localhost -certreq -file zoo2-
cert-file
keytool -keystore zoo3.keystore.jks -alias localhost -certreq -file zoo3-
cert-file

this will creates certification request file for eaach broker and then that
will be exported to be signed by ca-cert
```

Sign the certificates using the CA:

```
openssl x509 -req -CA ca-cert -CAkey ca-key -in zoo1-cert-file -out zoo1-
cert-signed -days 365 -CAcreateserial
openssl x509 -req -CA ca-cert -CAkey ca-key -in zoo2-cert-file -out zoo2-
cert-signed -days 365 -CAcreateserial
openssl x509 -req -CA ca-cert -CAkey ca-key -in zoo3-cert-file -out zoo3-
cert-signed -days 365 -Cacreateserial


above commands are used to sign the requested files (zoo1,zoo2,zoo3-cert-
file) using ca-key and then it will out the signed certificate as in file
that zoo1,zoo2,zoo3-signed-cert
```

6. **Import the Signed Certificates:**

After signing, the broker's keystore needs to include both the CA certificate and the signed broker certificate.

Import the CA and signed certificates back into each broker's keystore:

```
keytool -keystore zoo1.keystore.jks -alias CARoot -import -file ca-cert
keytool -keystore zoo1.keystore.jks -alias localhost -import -file zoo1-
cert-signed
same for all brokers (zoo2, zoo3)

here we are importing signed certificate into brokers keystore.jks which
will stores certificates and private key
and aldo we are importing ca-cert to broker keystore.jks
```

7. **Copy Keystores and Truststores to Brokers:**

Each broker needs its own keystore and truststore files.

Copy the necessary files to each Kafka broker:

```
scp zoo2.keystore.jks server.truststore.jks
cloud_user@zoo2:/home/cloud_user
```

```
scp zoo3.keystore.jks server.truststore.jks
cloud_user@zoo3:/home/cloud_user
```

Here we are copying the other broker's keystore.jks using scp command into their servers

8. **Configure Brokers for SSL:**

Kafka needs to know where the keystores are and what passwords to use.
`server.properties` for each broker to enable SSL:

`sudo vi /etc/kafka/server.properties`. Add the following lines to configure SSL listeners:

```
properties
listeners=PLAINTEXT://broker1:9092,SSL://broker1:9093
ssl.keystore.location=/var/private/ssl/server.keystore.jks
ssl.keystore.password=<keystore password>
ssl.key.password=<broker key password>
ssl.truststore.location=/var/private/ssl/server.truststore.jks
ssl.truststore.password=<trust store password>
ssl.client.auth=none
```

Comment the `advertised.listeners` line

```
#advertised.listeners=PLAINTEXT://broker1:9092
```

Here we are configring the server.proprties (/etc/kafka/server.properties)
Kafka needs to know that where the brokers keystore.jks located and we have to mention the password of kyestore of evry broker

Also needs to know that where is the server truststore and we have to mention the password of that

We have to uncomment and set listners with SSL network protocol as aove shown for secure port and comment the advertiesed.listners

9. **Restart Kafka Brokers:**

After making configuration changes, you need to restart Kafka on all brokers.

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

10. **Connect Clients Using SSL:**

Once the brokers are configured for SSL, you can connect clients securely using the certificates.

Copy the client truststore to an appropriate location on the client machine:

```
sudo cp ~/certs/client.truststore.jks /var/private/ssl/
```

```
vi ~/client-ssl.properties
security.protocol=SSL
ssl.truststore.location=/var/private/ssl/client.truststore.jks
ssl.truststore.password=<client trust store password>
```

```
kafka-console-consumer --bootstrap-server broker1:9093 --topic tls-test --
consumer.config client-ssl.properties
```

In clint machine have to cretae client-SSL.properties for secure connection to server and there we have mentioned that client truststorelocation and its password and choose to mention SSL network protocol

Files and Directories Created:

1. **Directories:**
   - `certs`: This directory contains the CA key and certificates, and it is where certificates for brokers and clients are generated.
   - `/var/private/ssl/`: This directory holds the keystore and truststore files needed for each broker.
2. **Files:**
   - `ca-key`: The private key of the certificate authority.
   - `ca-cert`: The public certificate of the CA.
   - `client.truststore.jks`: Truststore for clients to authenticate brokers.
   - `server.truststore.jks`: Truststore for brokers to authenticate other brokers.
   - `zoo1.keystore.jks`, `zoo2.keystore.jks`, `zoo3.keystore.jks`: Keystore files for the three Kafka brokers.
   - `zoo1-cert-file`, `zoo2-cert-file`, `zoo3-cert-file`: Certificate requests generated for each broker.
   - `zoo1-cert-signed`, `zoo2-cert-signed`, `zoo3-cert-signed`: Broker certificates signed by the CA.