

MTLS IMPLEMENT FOR KAFKA IN DOCKER-COMPOSE

1. CREATE CERTIFICATES OF EACH BROKER IN LOCALLY :

A. Using openssl command create ca certificate of certification authority

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

B. Create each client's trust store and keystore JDK files and import the certificates of ca authority

- `keytool -keystore client.keystore.jks -alias kafkauser -validity 365 -genkey -keyalg RSA -dname "CN=kafka"`
- Client trust already created during TLS

C. Create certificate request file of client using keytool command

- `keytool -keystore client.keystore.jks -alias kafkauser -certreq -file client-cert-file`

D. Request to CA to sign the certificates of brokers

- `openssl x509 -req -CA ca-cert -CAkey ca-key -in client-cert-file -out client-cert-signed -days 365 -CAcreateserial`

E. Importing the signed certificates of each broker into the respected broker's keystore jdk files

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

- `keytool -keystore client.keystore.jks -alias localhost -import -file client-cert-signed`

2. Configuring client.properties file :

`security.protocol=SSL`

`ssl.truststore.location=/etc/kafka/secrets/client.truststore.jks`

`ssl.truststore.password=allkeys`

`ssl.keystore.location=/etc/kafka/secrets/client.keystore.jks`

`ssl.keystore.password=allkeys`

`ssl.key.password=allkeys`

3. Volume Mounting :

A). - `/home/charan/stores:/etc/kafka/secrets` :

Already we have mounted some files for broker side that I have already mentioned in TLS document those files are :

- 1). `server.truststore.jks`
- 2). `kafka1.keystore.jks`
- 3). `Kafka2.keystore.jks`
- 4). `Kafka3.keystore.jks`
- 5). `Client.truststore.jks`
- 6). `ssl.creds`

Then we have to add another file `Client.keystore.jks` to local directory

`/home/charan/stores`

- 7) `Client.keystore.jks`

B). - /home/charan/configs:/home/appuser/client

Mounting another volume of client.properties file on each broker containers to for certification validation of client

In home/charan/configs directory the client.properties is there and we are mounting it to /home/appuser/client directory inside each broker's container

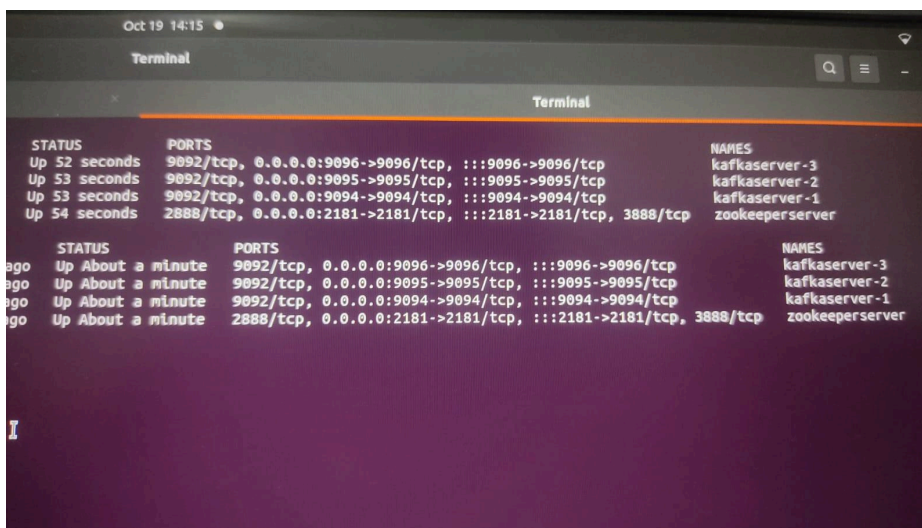
4. Run containers :

Inside the directory where docker-compose.yml present .Navigate into that Directory and run the all containers by below command :

> [Docker-compose up](#)

And confirm that all the containers are running up to check that >>>>

> [docker ps](#)

A terminal window screenshot showing the output of the 'docker ps' command. The output is a table with three columns: STATUS, PORTS, and NAMES. It lists four containers: kafka-3, kafka-2, kafka-1, and zookeeper. All containers are in the 'Up' state. The zookeeper container is up for 54 seconds, while the kafka containers are up for 52, 53, and 53 seconds respectively. The terminal has a dark background with light-colored text.

STATUS	PORTS	NAMES
Up 52 seconds	9092/tcp, 0.0.0.0:9096->9096/tcp, :::9096->9096/tcp	kafka-3
Up 53 seconds	9092/tcp, 0.0.0.0:9095->9095/tcp, :::9095->9095/tcp	kafka-2
Up 53 seconds	9092/tcp, 0.0.0.0:9094->9094/tcp, :::9094->9094/tcp	kafka-1
Up 54 seconds	2888/tcp, 0.0.0.0:2181->2181/tcp, :::2181->2181/tcp, 3888/tcp	zookeeper

5. Navigate to go inside the one of container in bash mode:

```
> Docker exec -it container id bash
```

(Exec it with container-id or with container name)

6. Creating topic and producing messages with unsecure port 9092:

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

Which will creates the tls topic with 3 partitions with 3 replication factors of each partitions

7. Produce message with secure port of 9097:

```
Kafka-console-producer --broker-list kafka-1:9097 --topic  
tls-topic
```

It fails

If we are producing the messages **with** secure port **without** client properties that contains the certificates for secure encryption of messages. Then It will fails.

8. Produce messages with client.props :

```
> Kafka-console-producer --broker-list kafka-1:9097 --topic  
tls-topic --producer.config client.properties
```

IT works

Then we prompted to produce messages

