Secure Communication (TLS) in Datastax Enterprise Cluster

Assumptions:

- All nodes have FQDNs resolvable via DNS and matching certificate CNs.
- DSE 6.8 is installed and the cluster is healthy (verify with nodetool status).

On a admin machine, create required certificates.

Generate Certificates, Keystores, and Truststores

Create a certificate Repository:

```
mkdir -p /tmp/ssl-ca
cd /tmp/ssl-ca
```

Create rootca.conf:

```
[ req ]
distinguished_name = CA_DN
prompt             = no
output_password    = cassandra
default_bits       = 2048

[ CA_DN ]
C   = US
O   = ExampleOrg
OU = ExampleCluster
CN = RootCA
```

Generate root key and cert:

```
openssl req -config rootca.conf -new -x509 -nodes -keyout rootca.key -out rootca.crt -
days 3650
```

Create Shared Truststore:

```
keytool -keystore dse-truststore.jks -storetype JKS -importcert -file rootca.crt -
keypass cassandra -storepass cassandra -alias RootCA -noprompt
```

Verify:

keytool -list -keystore dse-truststore.jks -storepass Cassandra

Create a Keystore repository for all nodes:

```
mkdir -p /tmp/keystores/node{1..3}
```

Generate Per-Node Keystores and CSRs

For node1:

```
cd /tmp/keystores/node1
```

Generate keypair and keystore:

```
keytool -genkeypair -keyalg RSA -alias node1.example.com -keystore node1-keystore.jks
-storepass cassandra -keypass cassandra -validity 730 -keysize 2048 -dname
"CN=node1.example.com, OU=ExampleCluster, O=ExampleOrg, C=US" -ext
"SAN=ip:192.168.1.101"
```

Generate CSR:

```
keytool -keystore node1-keystore.jks -alias node1.example.com -certreq -file node1.csr
-keypass cassandra -storepass Cassandra
```

For node2:

```
cd /tmp/keystores/node2
```

Generate keypair and keystore:

```
keytool -genkeypair -keyalg RSA -alias node2.example.com -keystore node2-keystore.jks
-storepass cassandra -keypass cassandra -validity 730 -keysize 2048 -dname
"CN=node2.example.com, OU=ExampleCluster, O=ExampleOrg, C=US" -ext
"SAN=ip:192.168.1.102"
```

Generate CSR:

```
keytool -keystore node1-keystore.jks -alias node2.example.com -certreq -file node2.csr
-keypass cassandra -storepass Cassandra
```

For node3:

```
cd /tmp/keystores/node3
```

Generate keypair and keystore:

```
keytool -genkeypair -keyalg RSA -alias node3.example.com -keystore node3-keystore.jks
-storepass cassandra -keypass cassandra -validity 730 -keysize 2048 -dname
"CN=node3.example.com, OU=ExampleCluster, O=ExampleOrg, C=US" -ext
"SAN=ip:192.168.1.103"
```

Generate CSR:

```
keytool -keystore node3-keystore.jks -alias node3.example.com -certreq -file node3.csr
-keypass cassandra -storepass Cassandra
```

Sign CSRs with Root CA

```
cd /tmp/ssl-ca
```

Sign certificate for node1:

```
vi node1-san.conf
```

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = node1.example.com
IP.1 = 192.168.1.101
```

Sign:

```
openssl x509 -req -CA rootca.crt -CAkey rootca.key -in /tmp/keystores/node1/node1.csr
-out /tmp/keystores/node1/node1.crt_signed -days 3650 -CAcreateserial -passin
pass:cassandra -extfile node1-san.conf
```

Verify:

```
openssl verify -CAfile rootca.crt /tmp/keystores/node1/node1.crt_signed
```

Sign certificate for node2:

```
vi node2-san.conf
```

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = node2.example.com
IP.1 = 192.168.1.102
```

Sign:

```
openssl x509 -req -CA rootca.crt -CAkey rootca.key -in /tmp/keystores/node2/node2.csr
-out /tmp/keystores/node2/node2.crt_signed -days 3650 -CAcreateserial -passin
pass:cassandra -extfile node2-san.conf
```

Verify:

```
openssl verify -CAfile rootca.crt /tmp/keystores/node2/node2.crt_signed
```

Sign certificate for node3:

```
vi node3-san.conf
```

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = node3.example.com
IP.1 = 192.168.1.103
```

Sign:

```
openssl x509 -req -CA rootca.crt -CAkey rootca.key -in /tmp/keystores/node3/node3.csr
-out /tmp/keystores/node3/node3.crt_signed -days 3650 -CAcreateserial -passin
pass:cassandra -extfile node3-san.conf
```

Verify:

```
openssl verify -CAfile rootca.crt /tmp/keystores/node3/node3.crt_signed
```

Import Signed Certs and Root CA into Per-Node Keystores

For node1

```
cd /tmp/keystores/node1

keytool -keystore node1-keystore.jks -alias RootCA -importcert -file /tmp/ssl-
ca/rootca.crt -keypass cassandra -storepass cassandra -noprompt


keytool -keystore node1-keystore.jks -alias node1.example.com -importcert -file
node1.crt_signed -keypass cassandra -storepass cassandra -noprompt


keytool -list -keystore node1-keystore.jks -storepass Cassandra
```

For node2

```
cd /tmp/keystores/node2

keytool -keystore node2-keystore.jks -alias RootCA -importcert -file /tmp/ssl-
ca/rootca.crt -keypass cassandra -storepass cassandra -noprompt


keytool -keystore node2-keystore.jks -alias node2.example.com -importcert -file
node2.crt_signed -keypass cassandra -storepass cassandra -noprompt


keytool -list -keystore node2-keystore.jks -storepass Cassandra
```

For node3

```
cd /tmp/keystores/node3

keytool -keystore node3-keystore.jks -alias RootCA -importcert -file /tmp/ssl-
ca/rootca.crt -keypass cassandra -storepass cassandra -noprompt
```

```
keytool -keystore node3-keystore.jks -alias node3.example.com -importcert -file
node3.crt_signed -keypass cassandra -storepass cassandra -noprompt
```

```
keytool -list -keystore node3-keystore.jks -storepass Cassandra
```

On Each DSE Cluster Nodes create a directory for saving certificates

```
mkdir  -p /etc/des/keystores
```

Copy certificates the above created directory

```
scp /tmp/ssl-ca/dse-truststore.jks user@node1:/etc/dse/
scp /tmp/keystores/node1/node1-keystore.jks user@node1.example.com:/etc/dse/keystores/

scp /tmp/ssl-ca/dse-truststore.jks user@node2:/etc/dse/
scp /tmp/keystores/node2/node2-keystore.jks user@node2.example.com:/etc/dse/keystores/

scp /tmp/ssl-ca/dse-truststore.jks user@node3:/etc/dse/
scp /tmp/keystores/node3/node3-keystore.jks user@node3.example.com:/etc/dse/keystores/
```

Secure Certificate Directories (on each node repeat)

```
chmod 600 /etc/dse/dse-truststore.jks
chmod 600 /etc/dse/keystores/*.jks
chown cassandra:cassandra /etc/dse/dse-truststore.jks /etc/dse/keystores/*.jks
```

Configure Node-to-Node Encryption

Edit `cassandra.yaml` file and change following options on each node

```
vi $DSE_HOME/resources/cassandra/cassandra.yaml
```

```
server_encryption_options:
  internode_encryption: all
  keystore: /etc/dse/keystores/node1-keystore.jks
  keystore_password: cassandra
  truststore: /etc/dse/dse-truststore.jks
  truststore_password: cassandra
  require_client_auth: true  # Mutual authentication
  require_endpoint_verification: true
```

Note: `node1-keystore.jks` should be changed to respective node keystore

Restart nodes one after another (rolling restart):

```
dse cassandra-stop
dse cassandra
```

Client connections Encryption

Same certificates generated for Node to Node encryption can be used for client to Node encryption.

Modify Cassandra.yaml file as shown below on all nodes.

```
client_encryption_options:
  enabled: true
  optional: false
  keystore: /etc/dse/keystores/node1-keystore.jks
  keystore_password: cassandra
  require_client_auth: false
  truststore: /etc/dse/dse-truststore.jks
  truststore_password: cassandra
```

Perform rolling restart of all nodes one after another.

Send the `/etc/dse/dse-truststore.jks` file to clients