For MTLS

Create CA

```
$ openssl req -new x509 -newkey rsa:4096 -days 365 -keyout ca.key -out ca.crt -node
```

Create key for Server

```
$ openssl genrsa -out server.key 4096
```

Create key for Client

```
$ openssl genrsa -out client.key 4096
```

Create openssl.cnf file {to add extensions}

```
[ req ]
distinguished_name  = req_distinguished_name
policy              = policy_match
x509_extensions     = user_crt
req_extensions      = v3_req
[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = IN
countryName_min                 = 2
countryName_max                 = 2
stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = UNKNOWN
localityName                    = Locality Name (eg, city)
localityName_default            = UNKNOWN
0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = UNKNOWN
organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = UNKNOWN
commonName                      = Common Name
commonName_max                  = 64
emailAddress                    = Email Address
emailAddress_max                = 64
[ user_crt ]
nsCertType              = client, server, email
nsComment               = "OpenSSL Generated Certificate"
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer
[ v3_req ]
basicConstraints        = CA:FALSE
extendedKeyUsage        = serverAuth, clientAuth <ClientAuth- for client, serverAuth- for serverAuth>
subjectAltName          = @alt_names
[alt_names]
DNS.1 = example.com
DNS.2 = arunlocalhostinternal
IP.1 = 192.168.0.1
DNS.3 = <ec2-link></>
```

Create CSR for Client

```
$ openssl req -config openssl.cnf -new -key client.key -out client.csr
```

Create CSR for Server

```
$ openssl req -config openssl.cnf -new -key server.key -out server.csr
```

Sign client CSR with CA

```
$ openssl x509 -req -CA ca.crt -CAkey ca.key -in client.csr -out client.crt -days 365 -CAcreateserial
```

Sigh server CSR with CA

```
$ openssl x509 -req -CA ca.crt -CAkey ca.key -in server.csr -out server.crt -days 365 -CAcreateserial -extensions v3_req -extfile ./ssl-ext
```

`x509`: This option specifies that you want to work with X.509 certificates, which are widely used in SSL/TLS protocols

`-req`: This option indicates that the input file (`server.csr`) is a certificate signing request (CSR)

`-CAcreateserial`: This option tells OpenSSL to create a serial number file (`ca.srl`) if it doesn't already exist. The serial number is used to uniquely identify the signed certificate and keep track of the CA's issued certificates.

`-extensions v3_req`: This option specifies that you want to include X.509 extensions in the generated certificate. In this case, the extension is named `v3_req`. Extensions provide additional attributes or capabilities to a certificate beyond the basic information.

`-extfile ./ssl-extension-x509.cnf`: This option specifies the path to a configuration file (`ssl-extension-x509.cnf`) that contains the details of the X.509 extensions to be included in the certificate. The file is in OpenSSL configuration format and defines the specific extensions and their values.

Sign Client CSR with CA

```
$ openssl x509 -req -CA ca.crt -CAkey ca.key -in client.csr -out client.crt -days 365 -CAcreateserial -extensions v3_req -extfile ./ssl-ext
```

Create pkcs12 Server keystore

```
$ openssl pkcs12 -export -in server.crt -inkey server.key -out server_keystore.p12
```

Create pkcs12 Client keystore

```
$ openssl pkcs12 -export -in client.crt -inkey client.key -out client_keystore.p12
```

Create jks server keystore

```
$ keytool -keystore server.keystore.jks -alias server -import -file ca.crt
```

Create jks client keystore

```
$ keytool -keystore client.keystore.jks - alias client -import -file ca.crt
```