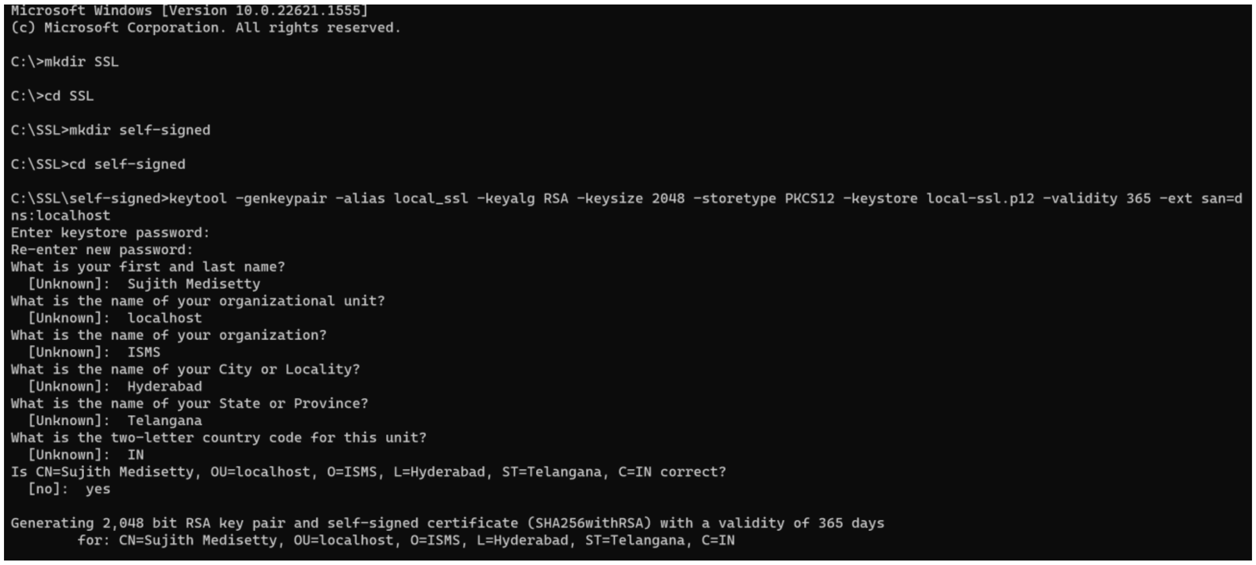

```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\>mkdir SSL

C:\>cd SSL

C:\SSL>mkdir self-signed

C:\SSL>cd self-signed

C:\SSL\self-signed>keytool -genkeypair -alias local_ssl -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore local-ssl.p12 -validity 365 -ext san=d
ns:localhost
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Sujith Medisetty
What is the name of your organizational unit?
  [Unknown]:  localhost
What is the name of your organization?
  [Unknown]:  ISMS
What is the name of your City or Locality?
  [Unknown]:  Hyderabad
What is the name of your State or Province?
  [Unknown]:  Telangana
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=Sujith Medisetty, OU=localhost, O=ISMS, L=Hyderabad, ST=Telangana, C=IN correct?
  [no]:  yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 365 days
        for: CN=Sujith Medisetty, OU=localhost, O=ISMS, L=Hyderabad, ST=Telangana, C=IN
```

Alias is defining the keypair name here it is local_ssl

Key algo is RSA with keysize of 2048 and the storage type is
PKCS12, key store is the file name where the key is going to store and it has extension .p12, validity of the key is 365 days and run from localhost

Enable ssl for springboot application

Self-signed certificate is not charted by the web browser, so we need to install the self-signed certificate on this system

Steps to make angular app from http to https

1. Open a command prompt or terminal window.
2. Navigate to a directory where you would like to store the certificate and private key files.
3. Generate a new private key using the following command:

openssl genrsa -out localhost.key 2048

This will create a new private key file called `localhost.key` with a length of 2048 bits.

4. Generate a new certificate signing request (CSR) using the following command:

openssl req -new -key localhost.key -out localhost.csr -config "C:\Program Files\OpenSSL-Win64\bin\openssl.cfg"

This will generate a new CSR file called `localhost.csr`. You will be prompted to enter information about the certificate, such as the common name, organization name, and country name.

5. Generate a new self-signed certificate using the following command:

```
openssl x509 -req -days 365 -in localhost.csr -signkey localhost.key -out localhost.crt
```

This will generate a new self-signed certificate file called `localhost.crt` that is valid for 365 days.

6. Copy the `localhost.key` and `localhost.crt` files to your Angular application's directory.

8. Run the following command to start the Angular development server with SSL/TLS encryption:

```
ng serve --ssl true --ssl-key localhost.key --ssl-cert localhost.crt
```

These steps should generate a new self-signed certificate and private key using OpenSSL, and configure SSL/TLS encryption for your Angular application using the new certificate and private key files.

In the fourth step, you generate a new certificate signing request (CSR) using the `openssl req` command. A CSR is a message that you send to a certificate authority (CA) to request a new SSL/TLS certificate. The CSR contains information about the entity requesting the certificate (such as the organization name, country, and common name) and the public key that will be used for the SSL/TLS encryption.

In the fifth step, you generate a new self-signed certificate using the `openssl x509` command. A self-signed certificate is a type of SSL/TLS certificate that is signed by the entity that created it (rather than by a trusted third-party CA). Self-signed certificates are typically used for testing or internal use, as they are not trusted by web browsers and can result in security warnings for end-users.

So, the difference between the fourth and fifth steps is that in the fourth step you create a CSR, which can be used to request a certificate from a trusted CA, while in the fifth step you create a self-signed certificate that is signed by the entity that created it.

why we have used keytool for spring boot cretification and openssl for angular certification?

The reason for using different tools for generating certificates for Spring Boot and Angular applications is due to the fact that Spring Boot and Angular are different frameworks and use different technologies.

Spring Boot is a Java-based framework that runs on the Java Virtual Machine (JVM). It provides a built-in server that supports SSL/TLS encryption, and it uses the Java KeyStore (JKS) format for storing certificates and keys. Keytool is a tool provided by Java that can be used to create, manipulate, and manage keystores and certificates in JKS format. Hence, we use Keytool to generate and manage SSL/TLS certificates for Spring Boot applications.

On the other hand, Angular is a client-side framework that runs in a web browser and communicates with a server over HTTP or HTTPS. To enable HTTPS for an Angular application, we need to use a server that supports SSL/TLS encryption and provide it with a valid SSL/TLS certificate. OpenSSL is a widely used and versatile tool that provides a command-line interface for generating and managing SSL/TLS certificates. It can generate certificates in various formats, including PEM, which is compatible with most web servers and is the format used by Angular.

Therefore, we use Keytool for generating certificates for Spring Boot applications as it uses JKS format, while we use OpenSSL for generating certificates for Angular applications as it generates certificates in PEM format which is compatible with web servers and client-side frameworks like Angular.