**How To Create CA and Generate SSL/TLS Certificates & Keys**

**Terminologies used in this article:**

PKI – Public key infrastructure

CA – Certificate Authority

CSR – Certificate signing request

SSL – Secure Socket Layer

TLS – Transport Layer Security

**Certificate Creation Workflow:**

Following are the steps involved in creating CA, SSL/TLS certificates.

* CA Key and Certificate Creation
* Generate a CA private key file using a utility (OpenSSL, cfssl etc)
* Create the CA root certificate using the CA private key.

**Server Certificate Creation Process**

* Generate a server private key using a utility (OpenSSL, cfssl etc)
* Create a CSR using the server private key.
* Generate the server certificate using CA key, CA cert and Server CSR.

**Generating Certificates Using OpenSSL**

Openssl utility is present by default on all Linux and Unix based systems.

**Generate CA Certificate and Key**

Step 1: Create a openssl directory and CD in to it.

cd /etc/ssl/

mkdir certificate

sudo yum install openssl

openssl genrsa -out jenkins.key 2048

openssl req -x509 -new -nodes \

-key jenkins.key -sha256 \

-days 1825 -out jenkins.crt

**You can enter below details:**

Country=US

State=Pennsylvania

City/locality=West Chester

Organization=Resolution Life US

Org unit/Department=IT

Name/CN:  %computername%.resolutionlifeus.com

Email: [XXXXXXX@resolutionlife.us](mailto:XXXXXXX@resolutionlife.us)

Enter command: ls -lart

Now you can see Jenkins.key and Jenkins.cert

**Generate SSL/TLS Certificates:**

**Step 1:** Create a server private key

**openssl genrsa -out server.key 2048**

**Step 2:** Create a configuration file named **csr.conf** for generating the Certificate Signing Request (CSR) as shown below. Replace the values as per your needs.

Note: **alt\_names should contain your servers DNS where you want to use the SSL**. Also, add all the IPs associated with the server if clients use the IP to connect to the server over SSL.

[ req ]

default\_bits = 2048

prompt = no

default\_md = sha256

req\_extensions = req\_ext

distinguished\_name = dn

[ dn ]

C = US

ST = California

L = San Fransisco

O = Scriptcrunch

OU = Scriptcrunch Dev

CN = scriptcrunch.com

[ req\_ext ]

subjectAltName = @alt\_names

[ alt\_names ]

DNS.1 = scriptcrunch

DNS.2 = scriptcrunch.com

IP.1 = 10.34.12.5

IP.2 = 10.34.12.5

**Step 3:** **Generate the CSR using the private key and config file.**

openssl req -new -key server.key -out server.csr -config csr.conf

**Step 4: Generate the server SSL certificate using ca.key, ca.crt and server.csr**

openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key \

-CAcreateserial -out server.crt -days 10000 \

-extfile csr.conf

Enter the **ls -lart**



**Reference**:

* https://devopscube.com/configure-ssl-jenkins/
* [**https://scriptcrunch.com/create-ca-tls-ssl-certificates-keys/**](https://scriptcrunch.com/create-ca-tls-ssl-certificates-keys/)
* <https://www.digicert.com/order/order-2.php> --🡪 **Digi certificate**
* <https://www.digicert.com/support/tools/certificate-utility-for-windows>
* https://sopblog.com/how-to-enable-ssl-in-jenkins-server/

**How to Generate a CSR (Certificate Signing Request) on Linux:**

$ sudo apt install openssl [On Debian/Ubuntu]

$ sudo yum install openssl [On CentOS/RHEL]

$ sudo dnf install openssl [On Fedora]

openssl req -new -newkey rsa:2048 -nodes -keyout **example.com**.key -out **example.com**.csr

**You can enter below details:**

Country=US

State=Pennsylvania

City/locality=West Chester

Organization=Resolution Life US

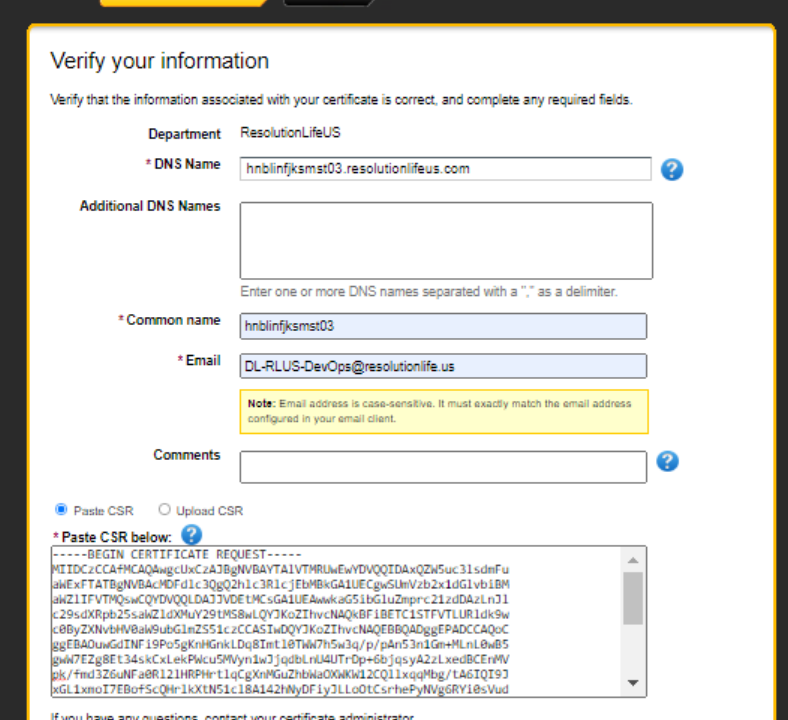
Org unit/Department=IT

Name/CN:  %computername%.resolutionlifeus.com

Email: [XXXXXXX@resolutionlife.us](mailto:XXXXXXX@resolutionlife.us)

Cat (ServerName.com).csr

Upload details into what you have digi cert link portal



**Reference:** <https://www.tecmint.com/generate-csr-certificate-signing-request-in-linux/>

**Validate CSR Certificate:** search google and validate

https://www.sslshopper.com/csr-decoder.html

**Default Password**: changeit

**Add a certificate to a trust store using keytool:**

**Default command:** Keytool -import -cacerts -alias rootcert -file ca.p7b -keystore cacerts

**Example:** Keytool -import -trustcerts -alias certAlias -file ca.p7b -keystore cacerts

**How to enable HTTP to HTTPS for Jenkins:**

**Step 1: Create pkcs12 file**

Now, use following command to create an intermediate pkcs12 file and define following parameters.

Give name of pkcs12 file name, example- jenkins\_demo.p12

Set strong password for pkcs12 file

Give FQDN or alias name, example- sopblog.com

**openssl pkcs12 -export -out jenkins\_demo.p12 -passout 'pass:password' \**

**-inkey demo.cert.key -in demo.cert.cert -name sopblog.com**

**Step 2: Create Java Keystore file (JKS)**

Now we will use Ketstore command-line tool to generate a new key “jenkins\_demo\_jks” and will set a password in deststorepass field.

**keytool -importkeystore -srckeystore jenkins\_demo.p12 \**

**-srcstorepass 'password' -srcstoretype PKCS12 \**

**-srcalias sopblog.com -deststoretype JKS \**

**-destkeystore jenkins\_demo.jks -deststorepass 'password' \**

**-destalias sopblog.com**

**Step 3: Copy keystore file to Jenkins**

Execute following commands to create keystore directory and add keystore file to Jenkins at default location as well as change directory permissions.

**cd /var/lib/jenkins**

**mkdir keystore**

**cp ~/jenkins\_demo.jks /var/lib/jenkins/keystore/**

**chmod 700 keystore/**

**Step 4: Change in Jenkins file**

Make few changes in the Jenkins’s file property. Open the file **/etc/sysconfig/jenkins file.**

**sudo vi /etc/sysconfig/jenkins**

Now find and replace keystore-password with the Keystore password, as you set in step 5 and set port no. Here in my case I used port 8080 but you can use any other port as you want.

**JENKINS\_PORT="-1"**

**JENKINS\_HTTPS\_PORT="8080"**

**JENKINS\_HTTPS\_KEYSTORE="/var/lib/jenkins/keystore/jenkins\_demo.jks"**

**JENKINS\_HTTPS\_KEYSTORE\_PASSWORD="<keystore-password>"**

**JENKINS\_HTTPS\_LISTEN\_ADDRESS="0.0.0.0"**

**Save the configuration file and restart Jenkins service and check status.**

**sudo systemctl restart jenkins**

**sudo systemctl status Jenkins**

sopblog@ubuntu:~$ sudo systemctl status jenkins

jenkins.service – LSB: Start Jenkins at boot time

Loaded: loaded (/etc/init.d/jenkins; generated)

Active: active (exited) since Sat 2020-09-05 16:10:55 UTC; 2h 48min ago

Docs: man:systemd-sysv-generator(8)

Tasks: 0 (limit: 2332)

CGroup: /system.slice/jenkins.service

**Step 5: Validate the configuration**

Congratulation, we’re done with all steps. Now this should redirect your custom name from http: / / localhost: 8080 to https:// in your newly secured Jenkins. You should be able to access your Jenkins server over https at port 8080.

**https://<dns\_name/ip>:8080**

**Extra commands:** keytool -v -list -keystore caerts

**Reference:**

* <https://sopblog.com/how-to-enable-ssl-in-jenkins-server/>
* <https://devopscube.com/configure-ssl-jenkins/>
* <https://support.code42.com/Administrator/6/Configuring/Use_OpenSSL_to_install_a_keystore#Step_2:_Request_a_CA-signed_certificate>
* <https://sam.gleske.net/blog/engineering/2016/05/04/jenkins-with-ssl.html#assumptions>
* https://jansipke.nl/enable-https-jenkins/

**How to convert a certificate into the appropriate format:**

https://knowledge.digicert.com/solution/SO26449.html