**How to Configure SSL on Jenkins Server**

* Last Updated On: April 9, 2020
* [By: devopscube](https://devopscube.com/author/devopscube/)

It is very important to secure Jenkins by enabling SSL which runs in a project environment. This article walks you through the step by step guide for configuring SSL on a Jenkins server.

Following are the steps involved in configuring SSL on Jenkins server.

1. Obtain SSL certificates
2. Convert SSL keys to PKCS12 format
3. Convert PKCS12 to JKS format
4. Add JKS to Jenkins path
5. Configure Jenkins startup to use the JKS file.
6. Validate Jenkins SSL

Lets get started with the setup

Step 1: Obtain Domain & SSL Certificates

You should have a valid domain pointing to Jenkins server IP to configure SSL. The domain can be internal or external based on your organisations infrastructure.

SSL certificate can be obtained using the following methods.

1. In most cases, you will be having Jenkins in a private environment with an internal DNS and you can obtain the internal SSL certificates from the respective organizations.
2. You can also create self-signed SSL certificates using OpenSSL. Follow this blog, [generate SSL certificates](https://scriptcrunch.com/create-ca-tls-ssl-certificates-keys/)to create self-signed certificates using OpenSSL.
3. Also, you can use services as let’s encrypt for valid SSL certificates. But these certificates have to be renewed every three months.

Step 2: Convert SSL keys to PKCS12 format

**Note:** If you already have the certificate in.p12 or.pfx format, you don’t have to do this conversion.

The command given below converts SSL certs to intermediate PKCS12 format named jenkins.p12. Make sure you have the following certs with you before executing the command.

1. ca.crt
2. server.key
3. server.crt

Also,

1. Replace jenkins.devopscube.com in the command with your own alias name
2. Replace your-strong-password with a strong password.

openssl pkcs12 -export -out jenkins.p12 \

-passout 'pass:your-strong-password' -inkey server.key \

-in server.crt -certfile ca.crt -name jenkins.devopscube.com

Step 3: Convert PKCS12 to JKS format

Use the following keytool command to convertjenkins.p12 file to JKS format.

Replace the following with your own values.

1. -srcstorepass – Password used in Step 3
2. -deststorepass – Replace with a strong password.
3. -srcalias– alias name used in step 2
4. -destalias – Replace with a destination alias name.

keytool -importkeystore -srckeystore jenkins.p12 \

-srcstorepass 'your-secret-password' -srcstoretype PKCS12 \

-srcalias jenkins.devopscube.com -deststoretype JKS \

-destkeystore jenkins.jks -deststorepass 'your-secret-password' \

-destalias jenkins.devopscube.com

You should see a file named jenkins.jks in you current location.

Step 4: Add JKS to Jenkins path

jenkins\_keystore.jks file should be saved in a specific location where Jenkins can access it.

Let’s create a folder and move the jenkins\_keystore.jks key to that location.

mkdir -p /etc/jenkins

cp jenkins\_keystore.jks /etc/jenkins/

Change the permissions of the keys and folder.

chown -R jenkins: /etc/jenkins

chmod 700 /etc/jenkins

chmod 600 /etc/jenkins/jenkins.jks

Step 5: Modify Jenkins Configuration for SSL

All the key Jenkins startup configurations are present in /etc/sysconfig/jenkins file. All the SSL based configurations go into this file.

Open the file

sudo vi /etc/sysconfig/jenkins

Find and replace the values in the file as shown below.

**Note:** Replace your-keystore-passwordwith the Keystore password, you set in step 3

JENKINS\_PORT="-1"

JENKINS\_HTTPS\_PORT="8443"

JENKINS\_HTTPS\_KEYSTORE="/etc/jenkins/jenkins.jks"

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

JENKINS\_HTTPS\_LISTEN\_ADDRESS="0.0.0.0"

Save the configuration and restart Jenkins.

sudo systemctl restart jenkins

Check Jenkins status.

sudo systemctl status jenkins

Step 6: Validate SSL

Now you should be able to access Jenkins over https with port 8443

https://<jenkins-dns/ip>:8443

You can also use curl to verify SSL

curl -k https://<jenkins-dns/ip>:8443

Important step to configure ssl in Jenkins:

This is the steps to create certificate…

openssl pkcs12 -export -out jenkins.p12 \

-passout 'pass:DevOpsDev@2020' -inkey devopsdev.statebanktimes.in.key \

-in devopsdev.statebanktimes.in.crt -certfile devopsdev.statebanktimes.in.crt -name jenkins.devopssbi.com

keytool -importkeystore -srckeystore jenkins.p12 \

-srcstorepass 'DevOpsDev@2020' -srcstoretype PKCS12 \

-srcalias jenkins.devopssbi.com -deststoretype JKS \

-destkeystore jenkins.jks -deststorepass 'DevOpsDev@2020' \

-destalias jenkins.devopssbi.com

<VirtualHost jenkinsdev.statebanktimes.in:8081>

SSLEngine on

SSLProtocol all -SSLv2 -SSLv3

SSLCipherSuite HIGH:3DES:!aNULL:!MD5:!SEED:!IDEA

ProxyRequests Off

AllowEncodedSlashes NoDecode

ProxyPreserveHost On

ProxyPass / http://10.191.159.49:8083/ nocanon

ProxyPassReverse / http://10.191.159.49:8083/

SSLCertificateFile /opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.crt

SSLCertificateKeyFile /opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.key

RequestHeader set X-Forwarded-Proto "https"

# RequestHeader set X-Forwarded-Port "8081"

</VirtualHost>

<VirtualHost nexusdev.statebanktimes.in:8081>

SSLEngine on

SSLProtocol all -SSLv2 -SSLv3

SSLCipherSuite HIGH:3DES:!aNULL:!MD5:!SEED:!IDEA

ProxyRequests Off

AllowEncodedSlashes NoDecode

ProxyPreserveHost On

ProxyPass / http://10.191.159.49:8082/ nocanon

ProxyPassReverse / http://10.191.159.49:8082/

SSLCertificateFile /opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.crt

SSLCertificateKeyFile /opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.key

RequestHeader set X-Forwarded-Proto "https"

</VirtualHost>