

**SSL Configuration**

|  |
| --- |
|  |

# Version history

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Name** | **Version** | **Prepared by** | **Reviewed by** |
| SSL Configuration | v.01 | Abhishek Kushwaha | Sumit Malhotra |
|  |  |  |  |

Contents

[1 Version history 2](#_Toc61974175)

[3 Install Apache webserver with SSL/TLS support 4](#_Toc61974176)

[4 Verify installation and status: 5](#_Toc61974177)

[5 Verify loaded as a module into httpd server: 6](#_Toc61974178)

[apachectl -M | grep ssl 6](#_Toc61974179)

[ssl\_module (shared) 6](#_Toc61974180)

[6 Redirecting http traffic to https: 7](#_Toc61974181)

[6.1 Jenkins: 8](#_Toc61974182)

[6.2 JIRA: 8](#_Toc61974183)

[6.3 Keycloak: 9](#_Toc61974184)

# Install Apache webserver with SSL/TLS support

If the server already have httpd installed, you only need to install mod\_ssl, all the required configuration is done by the installer. Note however that in this case you need to restart httpd, so it can load the ssl module. By using the packages shipped with the distribution, we can make our life much easier, as Red Hat will provide properly tested updates for both the operating system and the webserver, of course, you need a subscription to receieve the updates - but updates are needed for the operating system anyway to stay up to date.

yum install httpd mod\_ssl –y

Enable and start httpd server:

systemctl enable httpd

systemctl start httpd

# Verify installation and status:

You can check status of the webserver using systemd:

systemctl status httpd

To Check that mod\_ssl is properly installed:

rpm -q mod\_ssl

mod\_ssl-2.4.6-80.el7.x86\_64

# Verify loaded as a module into httpd server:

## apachectl -M | grep ssl

## ssl\_module (shared)

To find out when the self-signed (or any other) certificate will expire, we have to find it on the filesystem by consulting the ssl module's configuration file:

grep SSLCertificateFile /etc/httpd/conf.d/ssl.conf | grep -v "#"

SSLCertificateFile /etc/pki/tls/certs/ devopsdev.statebanktimes.in.crt

And then use openssl to get the expiration date:

openssl x509 -enddate -noout -in /etc/pki/tls/certs/ devopsdev.statebanktimes.in.crt notAfter=Jul 10 07:06:17 2019 GMT

After (or rather, before) the certificate expires, you have to renew or replace it with a certificate the clients trust. A more elegant approach in contrast to self-signed certificates is requesting and using a certificate from a CA (Certificate Authority) your clients already trust, either from your internal CA (which in turn can have a globally trusted root CA above it), or directly from a globally-trusted CA.  
  
To use the obtained certificate instead of the default, the below parameters must point to the certificate file, the certificate key, and the certificate of the CA that signed the SSL certificate, respectively. The files must be copied on the webserver, and must be readable by the operating system user running the webserver - in case of a Red Hat default install, the apache user. These parameters can be found in the above mentioned ssl.conf.

SSLCertificateFile /etc/httpd/custom-cert/ devopsdev.statebanktimes.in.crt

SSLCertificateKeyFile /etc/httpd/custom-cert/ devopsdev.statebanktimes.in.key

SSLCACertificateFile /etc/httpd/custom-cert/ca.crt

# Redirecting http traffic to https:

Now that we serve over https, we can enforce the usage of https while serving all or part of our content. In our example, we are very secure, and use http only to redirect incoming clients to https.  
  
A question may arise, if we want to speak https only, why do we listen to http at all? Suppose an end user, who just heard of our site, and got an URL from a friend not containing the protocol. To this day, most browsers default to http protocol, if one is not specified explicitly. If we stop serving over http, the user typing the URL without https will receive an error message if his/her browser tries to reach our server over http.  
  
To redirect all incoming http requests to https, we create a file under **/etc/httpd/conf.d** with a descriptive name, say, **redirect\_http.conf** with the following content (where web.foobar.com is the DNS name of the site):

<VirtualHost devopsdev.statebanktimes.in:443>

Servername devopsdev.statebanktimes.in

Redirect permanent / https://devopsdev.statebanktimes.in/

</VirtualHost>

And restart the webserver. We can test if the redirection works correctly from the command line with wget (from a host that trusts the SSL certificate of the webserver):

**Makesure whatever the domain you are using that domain entry should by there in hosts file like below.**

**vi /etc/hosts**

10.191.159.48 ETADEVOPSAPP01DEV.sbi.co.in ETADEVOPSAPP01DEV jiradev.statebanktimes.in devopsdev.statebanktimes.in sonarqubedev.statebanktimes.in

10.191.159.49 ETADEVOPSAPP02DEV.sbi.co.in ETADEVOPSAPP02DEV jenkinsdev.statebanktimes.in jmeterdev.statebanktimes.in nexusdev.statebanktimes.in

10.191.159.50 ETADEVOPSAPP03DEV.sbi.co.in ETADEVOPSAPP03DEV gitlabdev.statebanktimes.in elkapmdev.statebanktimes.in kibanadev.statebanktimes.in nagiosdev.statebanktimes.in

**go to below path and do some changes.**

vim /etc/sysconfig/selinux

SELINUX=permissive

GITLAB:

Gitlab ssl conifuguration:

vim /etc/gitlab/gitlab.rb

981 line number only change below entry

nginx['ssl\_certificate'] = "/opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.crt"

nginx['ssl\_certificate\_key'] = "/opt/ssl\_cert/29122020/devopsdev.statebanktimes.in.key"

gitlab-ctl reconfigure

gitlab-ctl restart

gitlab-ctl stop

check domain is added in cat/etc/hosts file required domain is added or not

check httpd install or not

check mod\_ssl is install or not

**Enable x-frame in gitlab through adopt.**

vim /opt/gitlab/embedded/service/gitlab-rails/app/controllers/application\_controller.rb

Add below enty inside given file.

238 def default\_headers

239 headers['X-Frame-Options'] = 'ALLOW-FROM \*.statebanktime.in'

## Jenkins:

For Jenkins go to below path and change port:

vim /etc/sysconfig/Jenkins

JENKINS\_HTTPS\_PORT="8083"

Go to below path and put below entry:

**/etc/httpd/conf.d/ssl.conf**

<VirtualHost jenkinsdev.statebanktimes.in:8081>

SSLEngine on

SSLProtocol all -SSLv2 -SSLv3

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

Header set X-Frame-Options "ALLOW-FROM devopsdev.statebanktime.in:9090"

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"

## JIRA:

**Step1: go to below path and enable below configurations:**

**vi /opt/atlassian/jira/conf/server.xml**

**Setp1:**

<Connector port="8085" relaxedPathChars="[]|" relaxedQueryChars="[]|{}^&#x5c;&#x60;&quot;&lt;&gt;"

maxThreads="150" minSpareThreads="25" connectionTimeout="20000" enableLookups="false"

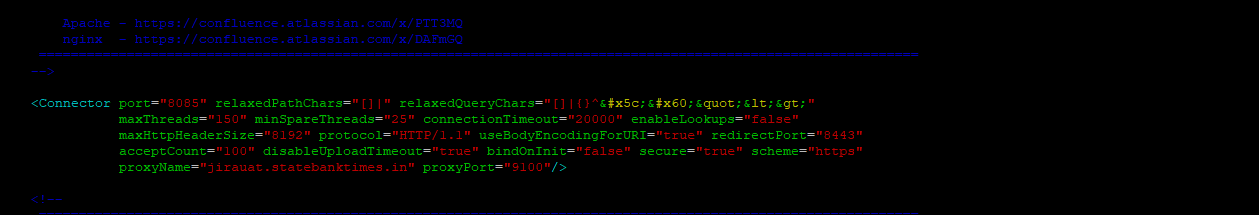
maxHttpHeaderSize="8192" protocol="HTTP/1.1" useBodyEncodingForURI="true" redirectPort="8443"

acceptCount="100" disableUploadTimeout="true" bindOnInit="false" secure="true" scheme="https"

proxyName="jirauat.statebanktimes.in" proxyPort="443"/>

/etc/rc.d/init.d/jira start

/etc/rc.d/init.d/jira stop



**Step2: Go to below path and do below configuration**

**vi /etc/httpd/conf.d/ssl.conf**

<VirtualHost jirauat.statebanktimes.in:9100>

SSLEngine on

SSLProtocol all -SSLv2 -SSLv3

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

Header set X-Frame-Options "ALLOW-FROM devopsuat.statebanktime.in:9090"

AllowEncodedSlashes NoDecode

ProxyPreserveHost On

ProxyPass / http://10.191.159.51:8085/ nocanon

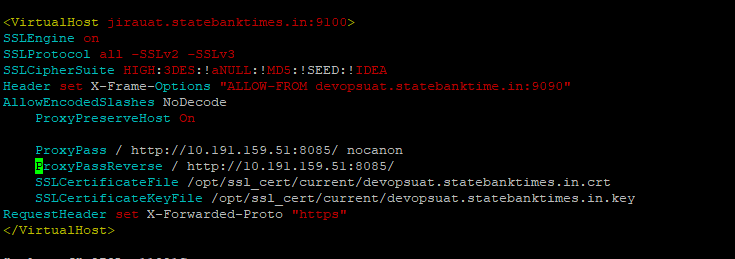
ProxyPassReverse / http://10.191.159.51:8085/

SSLCertificateFile /opt/ssl\_cert/current/devopsuat.statebanktimes.in.crt

SSLCertificateKeyFile /opt/ssl\_cert/current/devopsuat.statebanktimes.in.key

RequestHeader set X-Forwarded-Proto "https"

</VirtualHost>



https://confluence.atlassian.com/jirakb/security-headers-in-jira-939919914.html

**To enable X-fram in jira follow the below steps:**

1. From <jira-install>/bin open setenv.bat (for Windows) or setenv.sh (for Linux).
2. Find the section set JVM\_SUPPORT\_RECOMMENDED\_ARGS=
3. Add the following code into to the section "-Dcom.atlassian.jira.clickjacking.protection.disabled=true"
   1. The full argument should look as follows:

set JVM\_SUPPORT\_RECOMMENDED\_ARGS="-Dcom.atlassian.jira.clickjacking.protection.disabled=true"

## Keycloak:

vi /etc/httpd/conf.d/ssl.conf



**AFTER CONFIGURING SSL WITH REVERSE PROXY SIMPLY START THE HTTPD SERVICE.**

Systemctl restart httpd

Systemctl stop httpd

Systemctl status httpd

**Go to below configuration**

**vi /opt/keycloak-12.0.1/standalone/configuration/** **standalone.xml**

<http-listener name="default" socket-binding="http" redirect-socket="proxy-https" proxy-address-forwarding="true" enable-http2="true"/>

---

> <http-listener name="default" socket-binding="http" redirect-socket="https" enable-http2="true"/>

606d605

< <socket-binding name="proxy-https" port="9100"/>



**Add below entry in keycloak standalone.xml file:**

<keystore path="application.keystore" relative-to="jboss.server.config.dir" keystore-password="password" alias="server" key-password="password" generate-self-signed-certificate-host="keycloakuat.statebanktimes.in"/>

**Service for keycloak:**

This is the path were we can write service :

/etc/systemd/system

[Unit]

Description=Keycloak

After=syslog.target network.target

[Service]

Type=idle

User=root

Group=root

ExecStart=/opt/keycloak-12.0.1/bin/standalone.sh

TimeoutStartSec=600

TimeoutStopSec=600

[Install]

WantedBy=multi-user.target

**TESTLINKE:**

**After installation make sure stop the firewalld**

**systemctl stop firewalld**

**SSL CONFIGURATION**

**Setp: go to below path and chance only certificate path**

**vim /opt/testlink-1.9.20-9/apache2/conf/bitnami/binami.conf**

inside virtualhost put only certificate and key

SSLCertificateFile "/opt/ssl\_cert/current/ devopspreprod.statebanktimes.in.crt"

SSLCertificateKeyFile "/opt/ssl\_cert/current/devopspreprod.statebanktimes.in.key"

**Go to below directory and restart the testlink**

**/opt/testlink-1.9.20-9/**

**./** **ctlscript.sh start**

**./ctlscript.sh restart**

**./ctlscript.sh stop**

**Enable x-fram in testlink:**

**/opt/testlink-1.9.20-9/apache2/conf/httpd.conf**

**Add below entry in that file.**

**#Header set X-Frame-Options: "SAMEORIGIN"**

**Header set X-Frame-Options "ALLOW-FROM \*.statebanktime.in"**

**Coment below entry as given below.**

**# <IfVersion >= 2.4.7 >**

**# Header always setifempty X-Frame-Options SAMEORIGIN**

**# </IfVersion>**

**# <IfVersion < 2.4.7 >**

**# Header always merge X-Frame-Options SAMEORIGIN**

**# </IfVersion>**

**SONARQUBE:**

**Go to below path and put first line as java which version you want to use.**

**vim /opt/sonarqube/conf/** **wrapper.conf**

wrapper.java.command=/usr/java/jdk-11.0.9/bin/java

**Create sonarqube services:**

**Vim /etc/systemd/systemc/sonarqube.service**

[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=idle

User=sonarqube

Group=sonarqube

PermissionsStartOnly=true

ExecStart=/bin/nohup java -Xms32m -Xmx32m -Djava.net.preferIPv4Stack=true -jar /opt/sonarqube/lib/sonar-application-7.6.jar

StandardOutput=syslog

LimitNOFILE=65536

LimitNPROC=8192

TimeoutStartSec=5

Restart=always

[Install]

WantedBy=multi-user.target

ADD CA-CERTIFICATE:

Cd /opt/ssl-cert/current/keytool -importcert -trustcacerts -file /opt/ssl\_cert/current/cacert.cer -alias sbi -keystore "/opt/atlassian/jira/jre/lib/security/cacerts"

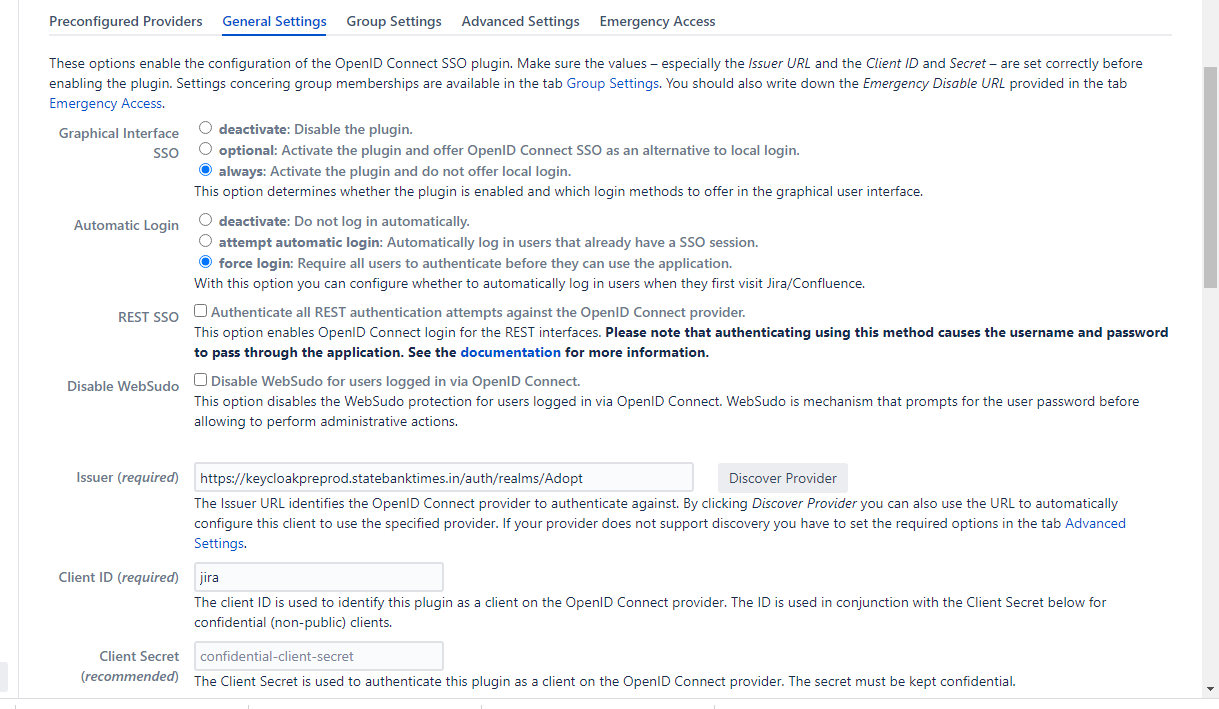
ADD ca-certificate: in linux where generated certificate is there

Cd /opt/ssl-cert/current /keytool -importcert -trustcacerts -file /opt/ssl\_cert/current/cacert.cer -alias sbi -keystore "/opt/jdk1.8.0\_271/jre/lib/security/cacerts"

**KEYCLOAK JIRA CONFIGURATION:**

**Manage app -> manage-app-> upload plugin-> configuration**

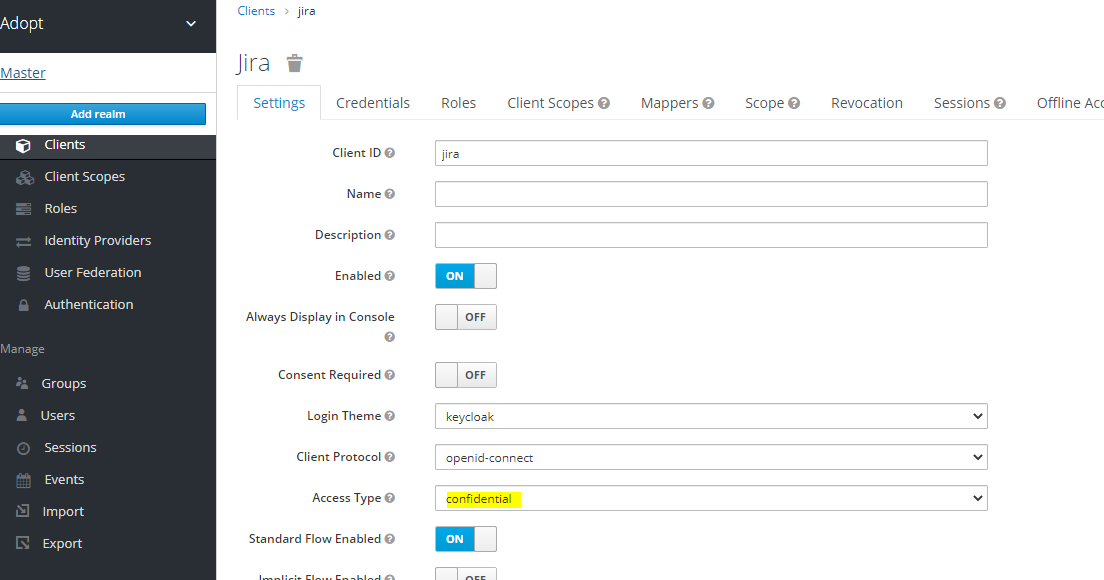
**Then do the below configuration**



**Jira keycloak configuration link:**

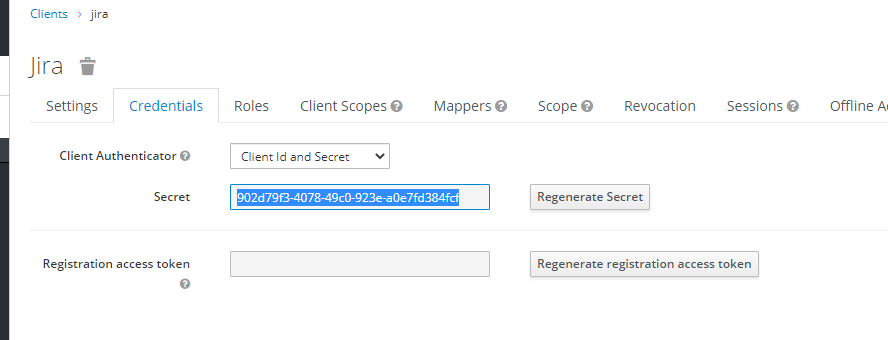
**https://plugins.miniorange.com/oauth-openid-single-sign-on-sso-jira-using-keycloak**

**go to keycloak:**



**Access Type:confidential**

**Then go to credential then secret code will will be shown below**



**After that create the same user in keycaloak by which we have create in jira**

**Tomcat: do below configuration for server.xml inside the tomcat**

**cd /opt/tomcat/conf/server.xml**

**<Connector port="9090" protocol="HTTP/1.1"**

**connectionTimeout="20000" scheme="https"**

**redirectPort="8443" proxyName="devopspreprod.statebanktimes.in"**

**proxyPort="443" />**

**ADOPT properties files changes.**

**Go to below direstory and do some changes:**

**/opt/tomcat/webapps/adopt-services/WEB-INF/classes**

**application.properties**

**crowd.properties**

**epm.properties**

**jenkins.properties**

**jira.properties**

**pipeline.propertie**

**planning.propertie**

**scm.properties**

**sonar.properties**

**IN ROLE TABEL:**

**Assign a role which we have given in keycloak role like admin,dev,test ets ….**

**NEXUS:**

**IN ADOPT-PORTAL:**

**/opt/tomcat/webapps/adopt-portal/WEB-INF/pages/blankprograms.jsp**

**Nexus url is hardcoded.**

**IN adopt-services**

**vim /opt/tomcat/webapps/adopt-services/WEB-INF/classes/scripts/stages/binary/binary.groovy**