**How to setup SSL/TLS with Apache httpd on Red Hat**

## Install Apache webserver with SSL/TLS support

# yum install httpd mod\_ssl -y

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.

## 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

And is 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 \_default\_:80>

Servername web.foobar.com

Redirect permanent / https://web.foobar.com/

</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):

$ wget http://devopsdev.statebanktimes.in/

**This is the exact configuration for reverse proxy**

**Step1:Go to the below configuration for below files.**

vim /etc/httpd/conf.d/ssl.conf

</VirtualHost>

<VirtualHost \*:9100>

SSLEngine on

SSLProtocol all -SSLv2 -SSLv3

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

    ProxyPreserveHost On

    ProxyPass / http://10.191.159.48:9101/

    ProxyPassReverse / http://10.191.159.48:9101/

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

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

</VirtualHost>

**Step2: /opt/sonarqube-7.9.1/conf/sonar.properties**

#sonar.web.host=0.0.0.0

sonar.web.host=10.191.159.48

#sonar.web.host=0.0.0.0

sonar.web.port=9101

**Step1:go to below path and do some changes.**

vim /etc/sysconfig/selinux

SELINUX=permissive

**Step2:go to host file and enter the host entry.**

Cd /etc/hosts

Put all domain name and host entry like below

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

**Jenkins:**

For Jenkins go to below path and change port:

vim /etc/sysconfig/Jenkins

JENKINS\_HTTPS\_PORT="8083"

<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**

<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="9100"/>

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

<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://linuxconfig.org/how-to-setup-ssl-tls-with-apache-httpd-on-red-hat>

To check the logs for reverse proxy:

cd /etc/httpd/logs/

access\_log

access\_log-20201216

error\_log

error\_log-20201129

error\_log-20201206

error\_log-20201213

ssl\_access\_log

ssl\_error\_log

ssl\_request\_log