

Using .htaccess File

Let's Encrypt SSL Certificate

Change Apache HTTP Port

Check Apache Server Status and Uptime

How to Password Protect Web Directories in Apache Using .htaccess File

Marin Todorov March 25, 2019

Categories

Apache 7 Comments

When you manage online projects, you often need to limit access to that project in order to protect it against the outside world. There are might be different reasons for that – for example you want to prevent search engine crawlers from accessing your site while it is still in development phase.

Apache Password Protect Directories

Password Protect Apache Web Directories

In this tutorial, I am going to show you how to password protect different web sites directories in **Apache** web server. There are many ways you can achieve this, but we will review two of them which are most commonly used.

The first method configures password protection directly in Apache's configuration file, while the second one uses **.htaccess** file.

Requirements

In order to setup password protection for your web directories, you will need to have:

- A working Apache web server

- The **AllowOverride AuthConfig** directive must be enabled in Apache configuration file.

Setup Apache Password Protected Directory

1. For this tutorial, we will be protecting the main web root directory `/var/www/html`. To protect that directory, open your Apache's configuration:

```
----- On RedHat/CentOS based systems -----  
# vi /etc/httpd/conf/httpd.conf
```

```
----- On Debian/Ubuntu based systems -----  
# nano /etc/apache2/sites-available/000-default.conf
```

2. Find the Apache Document directory root for `/var/www/html` and add the following things as suggested:

On Apache 2.2 Version

```
<Directory /var/www/html>  
Options Indexes Includes FollowSymLinks MultiViews  
AllowOverride All  
Order allow,deny  
Allow from all
```

```
</Directory>
```

On Apache 2.4 Version

```
<Directory /var/www/html>
```

```
Options Indexes Includes FollowSymLinks MultiViews
```

```
AllowOverride All
```

```
Require all granted
```

```
</Directory>
```

```

# Further relax access to the default document root:
<Directory "/var/www/html">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks MultiViews

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride All

#
# Controls who can get stuff from this server.
#
Require all granted
</Directory>

```

Apache 2.4: Enable AllowOverride All

3. Save the file and restart Apache by using the following command:

```

----- On Systemd -----
# systemctl restart httpd          [On RedHat based systems]
# systemctl restart apache2        [On Debian based systems]

```

```
----- On SysV init -----  
# service httpd restart      [On RedHat based systems]  
# service apache2 restart    [On Debian based systems]
```

4. Now we will use the **htpasswd** command to generate username and password for our protected directory. This command is used to manage user files for basic authentication.

The general syntax of the command is:

```
# htpasswd -c filename username
```

The **-c** option specifies the file that will keep the encrypted password and **username** specifies the user for the authentication.

5. Our password file needs to be located out of the Apache's web accessible directory so that it is well protected. For that purpose, we will create new directory:

```
# mkdir /home/tecmint
```

6. After that we will generate our username and password that will be stored in that directory:

```
# htpasswd -c /home/tecmint/webpass tecmint
```

Once you execute this command you will have to enter a password for our new user "tecmint" twice:

```
[root@www html]# htpasswd -c /home/tecmint/webpass tecmint
New password:
Re-type new password:
Adding password for user tecmint
[root@www html]# _
```

Create Apache User Password

After that we will need to make sure that Apache is able to read the “**webpass**” file. For that purpose, you will need to change the ownership of that file with the following command:

```
----- On RedHat/CentOS based systems -----
# chown apache: /home/tecmint/webpass
# chmod 640 /home/tecmint/webpass
```



```
----- On Debian/Ubuntu based systems -----  
# chown www-data /home/tecmint/webpass  
# chmod 640 /home/tecmint/webpass
```

7. At this point our new user and password are ready. Now we need to tell Apache to request password when accessing our targeted directory. For that purpose, create file called **.htaccess** in **/var/www/html**:

```
# vi /var/www/html/.htaccess
```

Add the following code in it:

```
AuthType Basic  
AuthName "Restricted Access"  
AuthUserFile /home/tecmint/webpass  
Require user tecmint
```

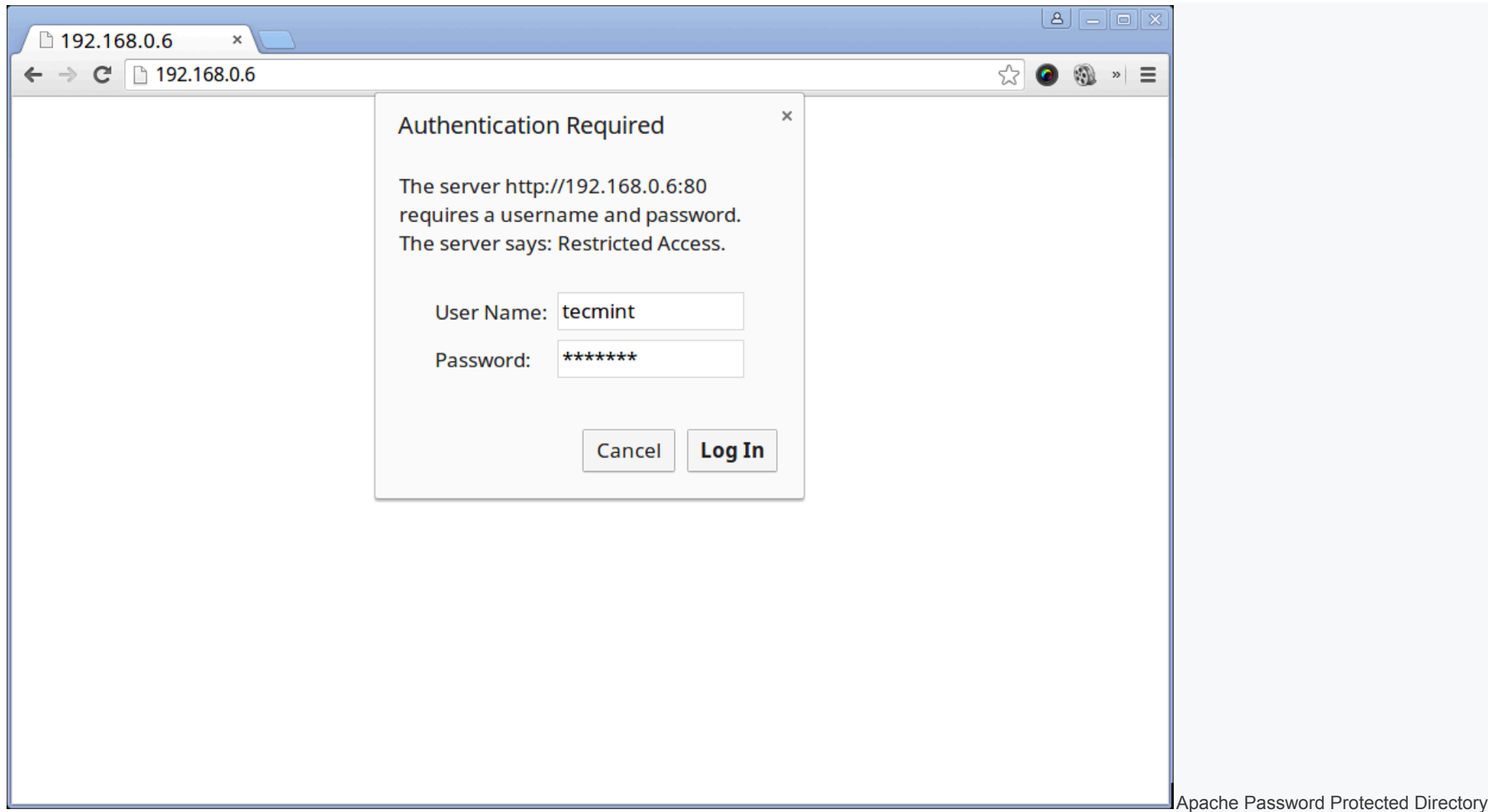
```
[root@www html]# cat /var/www/html/.htaccess
AuthType Basic
AuthName "Restricted Access"
AuthUserFile /home/tecmin/webpass
Require user tecmint
[root@www html]# _
```

Create Apache Restricted Access

8. Now save the file and put your setup to the test. Open your browser and enter your IP address or domain name in the web browser, for example:

```
http://ip-address
```

You should be prompted for username and password:



Authentication

Enter the username and password that you set to proceed to your page.

How to Install Let's Encrypt SSL Certificate to Secure Apache on RHEL/CentOS 7/6

Matei Cezar March 18, 2016

Categories

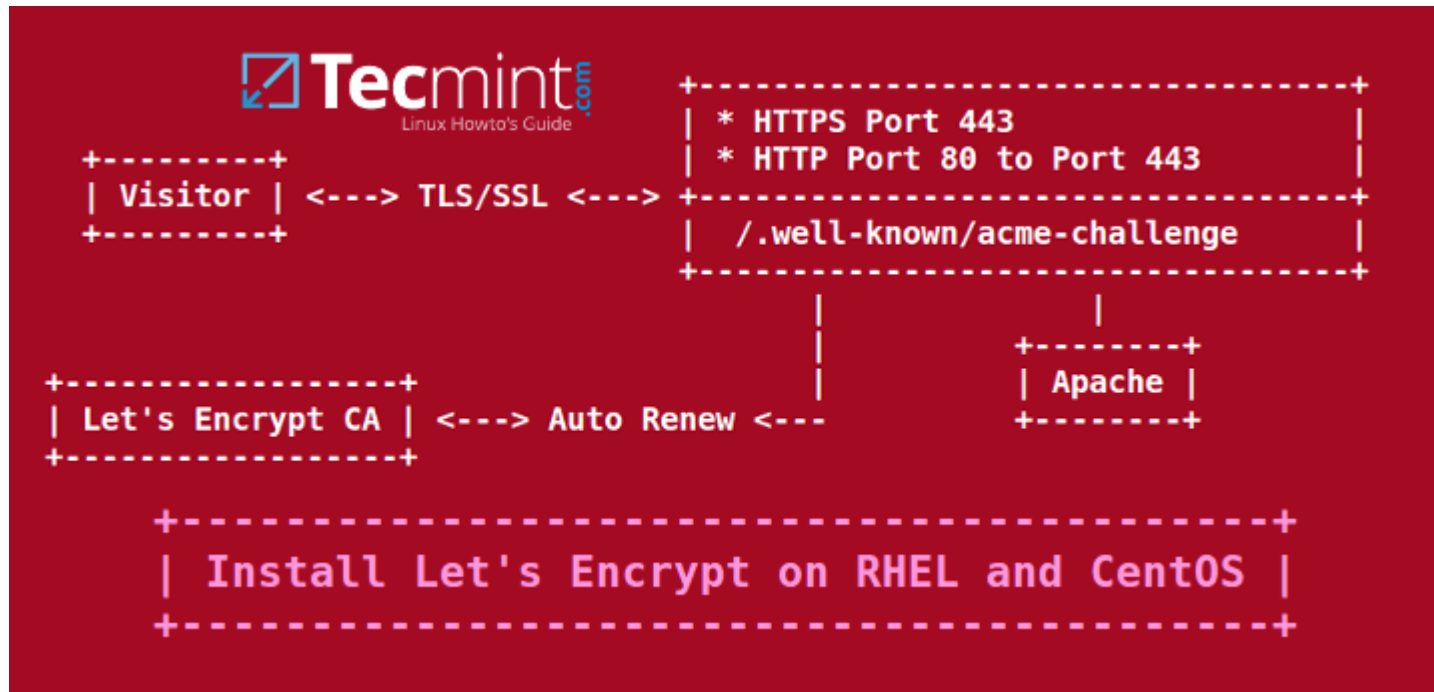
Apache, Let's Encrypt 11 Comments

Extending the last [Let's Encrypt tutorial](#) regarding SSL/TLS free certificates, in this article we are going to demonstrate how to obtain and install free SSL/TLS certificates issued by **Let's Encrypt Certificate Authority** for **Apache** web server on **CentOS/RHEL 7/6** and Fedora distributions too.

If you're looking to install Let's Encrypt for Apache on Debian and Ubuntu, follow this guide below:

[Setup Let's Encrypt to Secure Apache on Debian and Ubuntu](#)

Testing Sample Environment



Install Lets Encrypt for Apache on CentOS and RHEL

Requirements

1. A registered domain name with valid [A](#) records to point back to your server public IP Address.
2. Apache server installed with SSL module enabled and Virtual Hosting enabled in case you're hosting multiple domains or subdomains.

Step 1: Install Apache Web Server

1. If not already installed, httpd daemon can be installed by issuing the below command:

```
# yum install httpd
```

2. In order for Let's encrypt software to work with Apache, assure that the SSL/TLS module is installed by issuing the command below:

```
# yum -y install mod_ssl
```

3. Finally, start Apache server with the following command:

```
# systemctl start httpd.service [On RHEL/CentOS 7]
```

```
# service httpd start [On RHEL/CentOS 6]
```

Step 2: Install Let's Encrypt SSL Certificate

4. The simplest method of installing **Let's Encrypt** client is by cloning github repository in your filesystem. To install git on your system you must enable Epeel repositories with the following command.

```
# yum install epel-release
```

5. Once Epeel repos are added in your system, go ahead and install git client by running the command below:

```
# yum install git
```

6. Now , once you have installed all the required dependencies in order to deal with Let's Encrypt, go to `/usr/local/` directory and start pulling the Let's Encrypt client form its official github repository with the following command:

```
# cd /usr/local/  
# git clone https://github.com/letsencrypt/letsencrypt
```

Step 3: Obtain a Free Let's Encrypt SSL Certificate for Apache

7. The process of obtaining a free Let's Encrypt Certificate for Apache is automated for **CentOS/RHEL** thanks to the apache plugin.

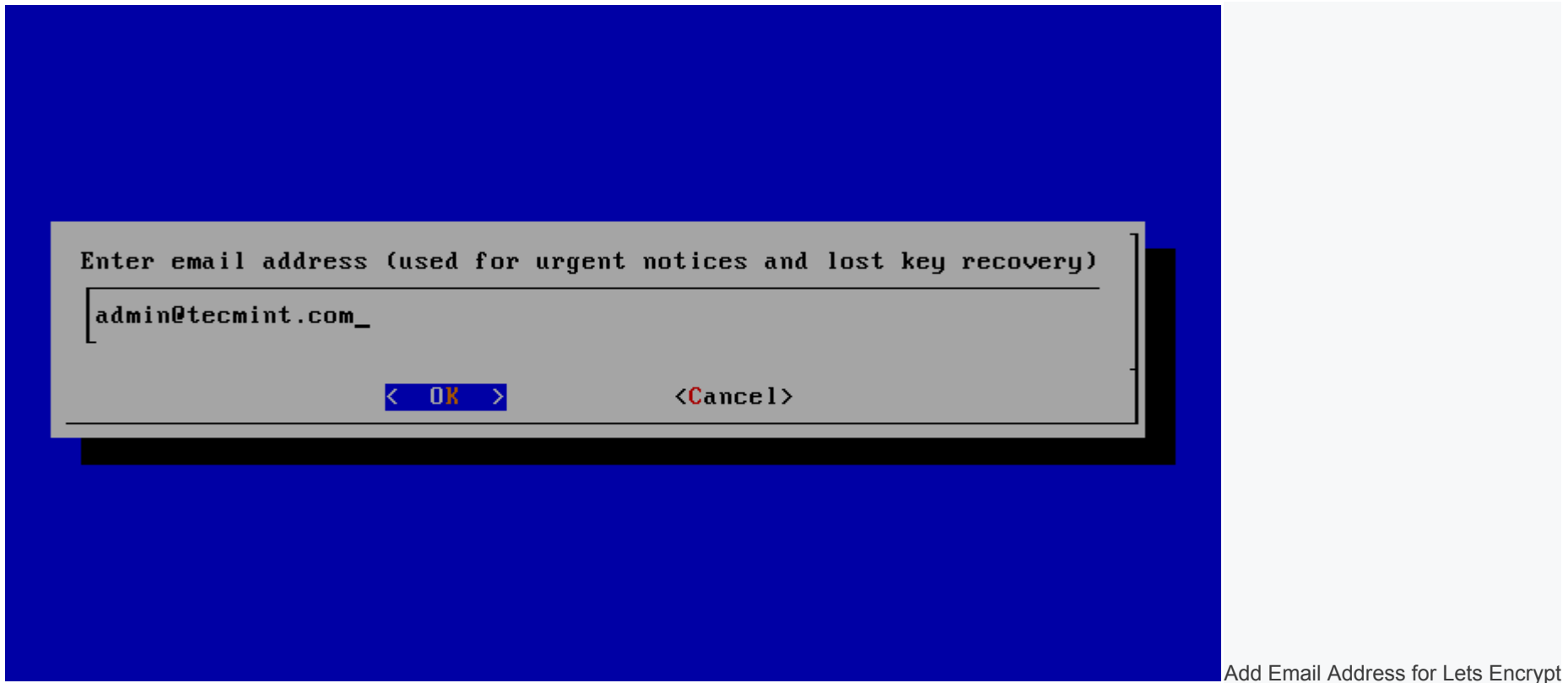
Let's run **Let's Encrypt** script command in order to obtain a SSL Certificate. Go to Let's Encrypt installation directory from `/usr/local/letsencrypt` and run the `letsencrypt-auto` command by providing `--apache` option and the `-d` flag for every subdomain you need a certificate.

```
# cd /usr/local/letsencrypt
# ./letsencrypt-auto --apache -d your.domain.tld
```

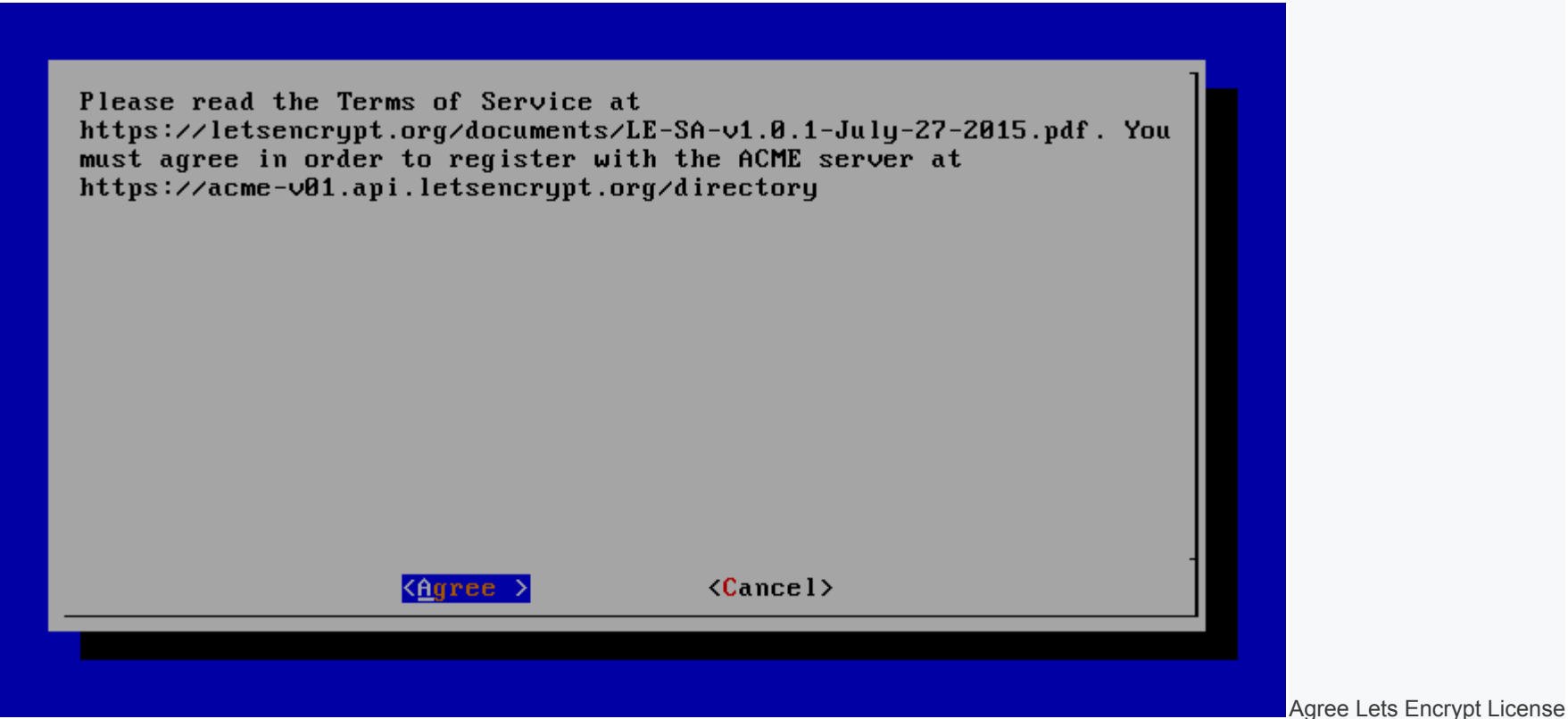
```
[root@localhost local]# cd /usr/local/letsencrypt/
[root@localhost letsencrypt]# ./letsencrypt-auto --apache -d caesar.tk
Bootstrapping dependencies for RedHat-based OSes...
yum is /bin/yum
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
-
```

Create Lets Encrypt SSL Certificate for Apache

8. Supply the email address that will be used by Let's Encrypt to recover your lost key or for urgent notices and press **Enter** to continue.



9. Agree the terms of the license by hitting Enter key.



Please read the Terms of Service at
<https://letsencrypt.org/documents/LE-SA-v1.0.1-July-27-2015.pdf>. You
must agree in order to register with the ACME server at
<https://acme-v01.api.letsencrypt.org/directory>

<Agree >

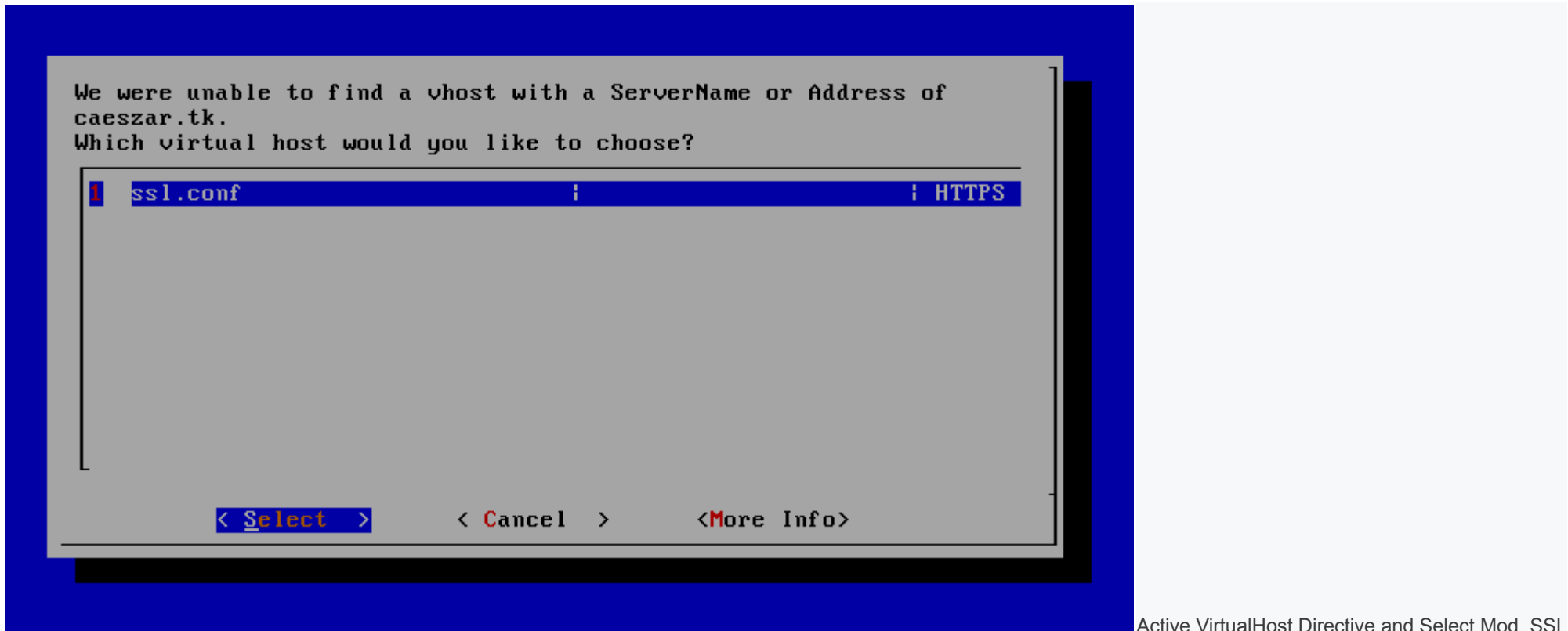
<Cancel>

Agree Lets Encrypt License

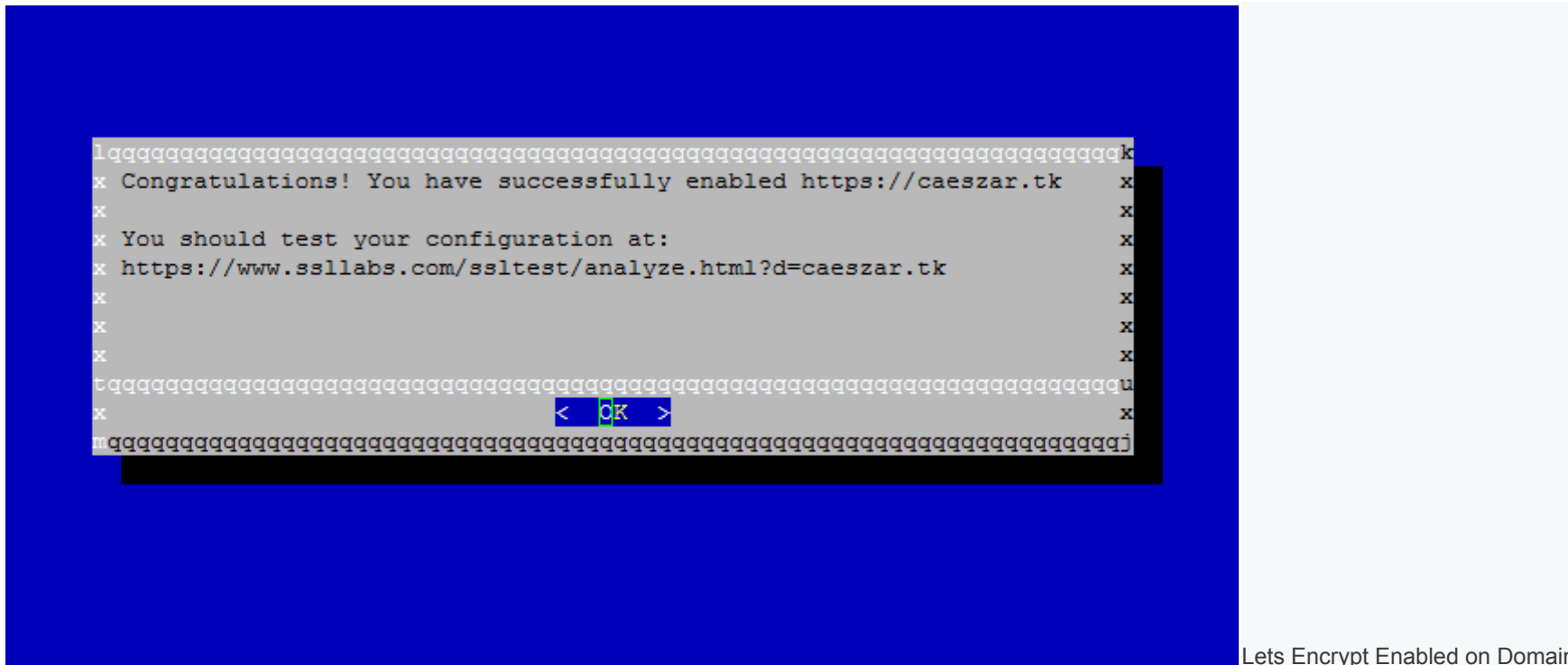
10. On **CentOS/RHEL**, by default, Apache server does not use the concept of separating directories for enabled hosts from available (inactive) hosts as **Debian** based distribution do.

Also, virtual hosting is disabled by default. The Apache statement which specifies the name of the server (**ServerName**) it's not present on SSL configuration file.

To activate this directive, Let's Encrypt will prompt you to select a virtual host. Because it does not find any Vhost available, select the `ssl.conf` file to be automatically modified by Let's Encrypt client and press **Enter** to continue.



11. Next, choose the **Easy** method for **HTTP** requests and press **Enter** to move forward.



That's it! You have successfully issued a **SSL/TLS** certificate for your domain. Now you can start browsing your website using **HTTPS** protocol.

Step 4: Test Free Let's Encrypt Encryption on Domain

13. In order to test the straightness of your domain SSL/TLS handshake visit the below link and test your certificate on your domain.

<https://www.ssllabs.com/ssltest/analyze.html>

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > caesar.tk

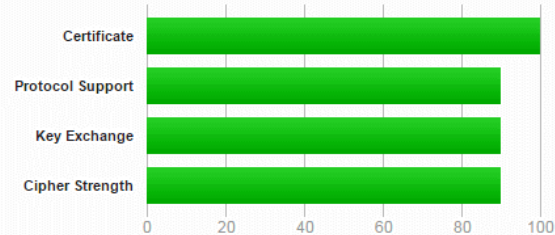
SSL Report: caesar.tk ()

Assessed on: Thu, 10 Mar 2016 21:26:59 UTC | [Hide](#) | [Clear cache](#)

[Scan Another »](#)

Summary

Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This server is vulnerable to the POODLE attack. If possible, disable SSL 3 to mitigate. Grade capped to C. [MORE INFO »](#)

This server accepts RC4 cipher, but only with older protocol versions. Grade capped to B. [MORE INFO »](#)

The server does not support Forward Secrecy with the reference browsers. [MORE INFO »](#)

Authentication



[Server Key and Certificate #1](#)



Domain

Verify Lets Encrypt Certificate on

14. If you receive a series of reports concerning your domain vulnerability in the conducted tests, then you need to fix those security holes urgently.

An overall rating of **C** class makes your domain very insecure. To fix these security problems, open Apache SSL configuration file and make the following changes:

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

Search for line with `SSLProtocol` statement and add `-SSLv3` at the end of the line.


```

GNU nano 2.3.1                                File: /etc/httpd/conf.d/ssl.conf

# Use "SSLCryptoDevice" to enable any supported hardware
# accelerators. Use "openssl engine -v" to list supported
# engine names. NOTE: If you enable an accelerator and the
# server does not start, consult the error logs and ensure
# your accelerator is functioning properly.
#
SSLCryptoDevice builtin
#SSLCryptoDevice ubsec

##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>

# General setup for the virtual host, inherited from global configuration
#DocumentRoot "/var/www/html"
#ServerName www.example.com:443

# Use separate log files for the SSL virtual host; note that LogLevel
# is not inherited from httpd.conf.
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel warn

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   SSL Protocol support:
#   List the enable protocol levels with which clients will be able to
#   connect.  Disable SSLv2 access by default:
SSLProtocol all -SSLv2 -SSLv3

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5:!SEED:!IDEA

#   Speed-optimized SSL Cipher configuration:
#   If speed is your main concern (on busy HTTPS servers e.g.),

^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text       ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is     ^V Next Page     ^U UnCut Text    ^T To Spell

```

Fix Apache SSL Configuration

Go deeper in the file, search and comment the line with `SSLCipherSuite` by placing a `#` in front of it and add the following content under this line:

```
SSLCipherSuite
ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AE
S256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA
256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:EC
DHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AE
S128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GC
M-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-S
HA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB
5-DES-CBC3-SHA
SSLHonorCipherOrder on
SSLOptions +StrictRequire
```

```
GNU nano 2.3.1                               File: /etc/httpd/conf.d/ssl.conf

#  SSL Protocol support:
#  List the enable protocol levels with which clients will be able to
#  connect.  Disable SSLv2 access by default:
SSLProtocol all -SSLv2 -SSLv3

#  SSL Cipher Suite:
#  List the ciphers that the client is permitted to negotiate.
#  See the mod_ssl documentation for a complete list.
#SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5:!SEED:!IDEA

SSLCipherSuite      ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDS
SSLHonorCipherOrder  on
SSLOptions +StrictRequire

# Speed-optimized SSL Cipher configuration:
# If speed is your main concern (on busy HTTPS servers e.g.),
# you might want to force clients to specific, performance
# optimized ciphers. In this case, prepend those ciphers
# to the SSLCipherSuite list, and enable SSLHonorCipherOrder.
# Caveat: by giving precedence to RC4-SHA and AES128-SHA
# (as in the example below), most connections will no longer
# have perfect forward secrecy - if the server's key is
# compromised, captures of past or future traffic must be
# considered compromised, too.
#SSLCipherSuite RC4-SHA:AES128-SHA:HIGH:MEDIUM:!aNULL:!MD5
#SSLHonorCipherOrder on

#  Server Certificate:
#  Point SSLCertificateFile at a PEM encoded certificate.  If
#  the certificate is encrypted, then you will be prompted for a
#  pass phrase.  Note that a kill -HUP will prompt again.  A new
#  certificate can be generated using the genkey(1) command.
SSLCertificateFile /etc/letsencrypt/live/caesar.tk/cert.pem

#  Server Private Key:
#  If the key is not combined with the certificate, use this
#  directive to point at the key file.  Keep in mind that if
#  you've both a RSA and a DSA private key you can configure
#  both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile /etc/letsencrypt/live/caesar.tk/privkey.pem

^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page     ^K Cut Text      ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page     ^U UnCut Text    ^T To Spell
```

Configure SSL Configuration

15. After you've made all the above changes, save and close the file, then restart Apache daemon to apply changes.

```
# systemctl restart httpd.service [On RHEL/CentOS 7]
```

```
# service httpd restart [On RHEL/CentOS 6]
```

16. Now, tests the status of your domain encryption again, by visiting the same link as above. To perform retests hit the Clear cache link from the website.

```
https://www.ssllabs.com/ssltest/analyze.html
```

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > caesar.tk

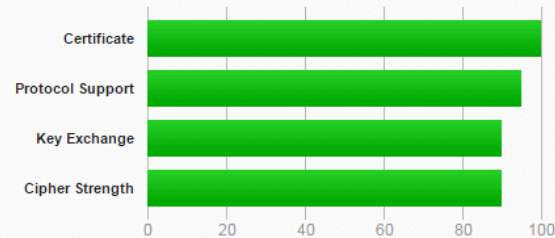
SSL Report: caesar.tk

Assessed on: Thu, 10 Mar 2016 21:45:10 UTC | [Hide](#) [Clear cache](#)

[Scan Another »](#)

Summary

Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

Authentication



Server Key and Certificate #1

Subject	caesar.tk Fingerprint SHA1: 57c8e0900eef8b518807b389ef8fe361de2486b9 Pin SHA256: w8Kvk8qIQhAwTUBulbtsB+TtAlJvCQnCBPQRQHh2h0w=
Common names	caesar.tk
Alternative names	caesar.tk
Valid from	Thu, 10 Mar 2016 20:22:00 UTC
Valid until	Wed, 08 Jun 2016 20:22:00 UTC (expires in 2 months and 28 days)

Test Lets Encrypt SSL Certificate on

Now you should get a class **A** overall rating, which means your domain is highly secured.

Step 4: Auto Renew Let's Encrypt Certificates on Apache

17. This beta version of Let's Encrypt software releases certificates with expiration date after **90** days. So, in order to renew the SSL certificate, you must execute the `letsencrypt-auto` command again before expiration date, with the same options and flags used to obtain the initial certificate.

An example on how to manually renew the certificate is presented below.

```
# cd /usr/local/letsencrypt
# ./letsencrypt-auto certonly --apache --renew-by-default -d your domain.tld
```

18. To automate this process, create the following bash script provided by [github erikaheidi](#), in `/usr/local/bin/` directory with the following content. (the script is slightly modified to reflect our letsencrypt installation directory).

```
# vi /usr/local/bin/le-renew-centos
```

Add the following content to `le-renew-centos` file:

```
#!/bin/bash
```

```
domain=$1
```

```
le_path='/usr/local/letsencrypt'
```

```
le_conf='/etc/letsencrypt'
```

```
exp_limit=30;
```

```
get_domain_list(){
```

```
    certdomain=$1
```

```
    config_file="$le_conf/renewal/$certdomain.conf"
```

```
    if [ ! -f $config_file ] ; then
```

```
        echo "[ERROR] The config file for the certificate $certdomain was not found."
```

```
        exit 1;
```

```
    fi
```

```
    domains=$(grep --only-matching --perl-regex "(?<=domains \= ).*" "${config_file}")
```

```
    last_char=$(echo "${domains}" | awk '{print substr($0,length,1)}')
```

```
    if [ "${last_char}" = "," ] ; then
```

```
        domains=$(echo "${domains}" | awk '{print substr($0, 1, length-1)}')
```

```
    fi
```

```
    echo $domains;
```

```
}
```

```
if [ -z "$domain" ] ; then
```

```
echo "[ERROR] you must provide the domain name for the certificate renewal."
```

```
exit 1;
```

```
fi
```

```
cert file="/etc/letsencrypt/live/$domain/fullchain.pem"
```

```
if [ ! -f $cert file ]; then
```

```
echo "[ERROR] certificate file not found for domain $domain."
```

```
exit 1;
```

```
fi
```

```
exp=$(date -d "`openssl x509 -in $cert file -text -noout|grep "Not After"|cut -c 25-`" +%s)
```

```
datenow=$(date -d "now" +%s)
```

```
days exp=$(echo \( $exp - $datenow \) / 86400 |bc)
```

```
echo "Checking expiration date for $domain..."
```

```
if [ "$days exp" -gt "$exp limit" ] ; then
```

```
echo "The certificate is up to date, no need for renewal ($days exp days left)."
```

```
exit 0;
```

```
else
```

```
echo "The certificate for $domain is about to expire soon. Starting renewal request..."
```

```
domain list=$( get domain list $domain )
```

```
"$le path"/letsencrypt-auto certonly --apache --renew-by-default --domains "${domain list}"
```

```
echo "Restarting Apache..."
```

```
/usr/bin/systemctl restart httpd
```

```
echo "Renewal process finished for domain $domain"
```

```
exit 0;
```

```
fi
```


19. Grant execution permissions for the script, install **bc** package and run the script in order to test it. Use your domain name as a positional parameter for the script. Issue the below commands to accomplish this step:

```
# yum install bc
# chmod +x /usr/local/bin/le-renew-centos
# /usr/local/bin/le-renew-centos your domain.tld
```

20. Finally, using Linux scheduling, add a new cron job in order to run the script every two months, assuring that your certificate will be updated before expiration date.

```
# crontab -e
```

Add the following line at the bottom of the file.

```
0 1 1 */2 * /usr/local/bin/le-renew-centos your domain.tld >> /var/log/your domain.tld-renew.log
2>&1
```

That's it! Your Apache server running on top of **CentOS/RHEL** system is now serving SSL content using a free Let's Encrypt SSL certificate.

How to Change Apache HTTP Port in Linux

Matei Cezar January 31, 2018

Categories

Apache Leave a comment

Apache HTTP server is one of the most used web server in internet today, do to its flexibility, stability and a pleiad of features, some of which are not for the moment present in other web servers, such a rival **Nginx**.

Some of the most important features of Apache include the ability to load and run different types of modules and special configurations at runtime, without actually stopping the server or, worse, compiling the software each time a new module must be added and the special role played by [.htaccess files](#), which can alter web server configurations specific to webroot directories.

By default, Apache web server is instructed to listen for incoming connection and bind on port **80**. If you opt for the TLS configuration, the server will listen for secure connections on port **443**.

In order to instruct Apache web server to bind and listen for web traffic on other ports than the standard web ports, you need to add a new statement containing the newly port for future bindings.

In **Debian/Ubuntu** based system, the configuration file that needs modified is **/etc/apache2/ports.conf** file and on **RHEL/CentOS** based distributions edit **/etc/httpd/conf/httpd.conf** file.

Open the file specific to your own distribution with a console text editor and add the new port statement as shown in the below excerpt.

```
# nano /etc/apache2/ports.conf      [On Debian/Ubuntu]
# nano /etc/httpd/conf/httpd.conf  [On RHEL/CentOS]
```

In this example we'll configure **Apache HTTP** server to listen on connections on port **8081**. Make sure you add the below statement in this file, after the directive that instructs the web server to listen on port **80**, as illustrated in the below image.

```
Listen 8081
```

```
GNU nano 2.7.4 File: /etc/apache2/ports.conf

# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80
Listen 8081

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Change Apache Port on Debian and Ubuntu

```
GNU nano 2.3.1 File: /etc/httpd/conf/httpd.conf

# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 80
Listen 8081
```

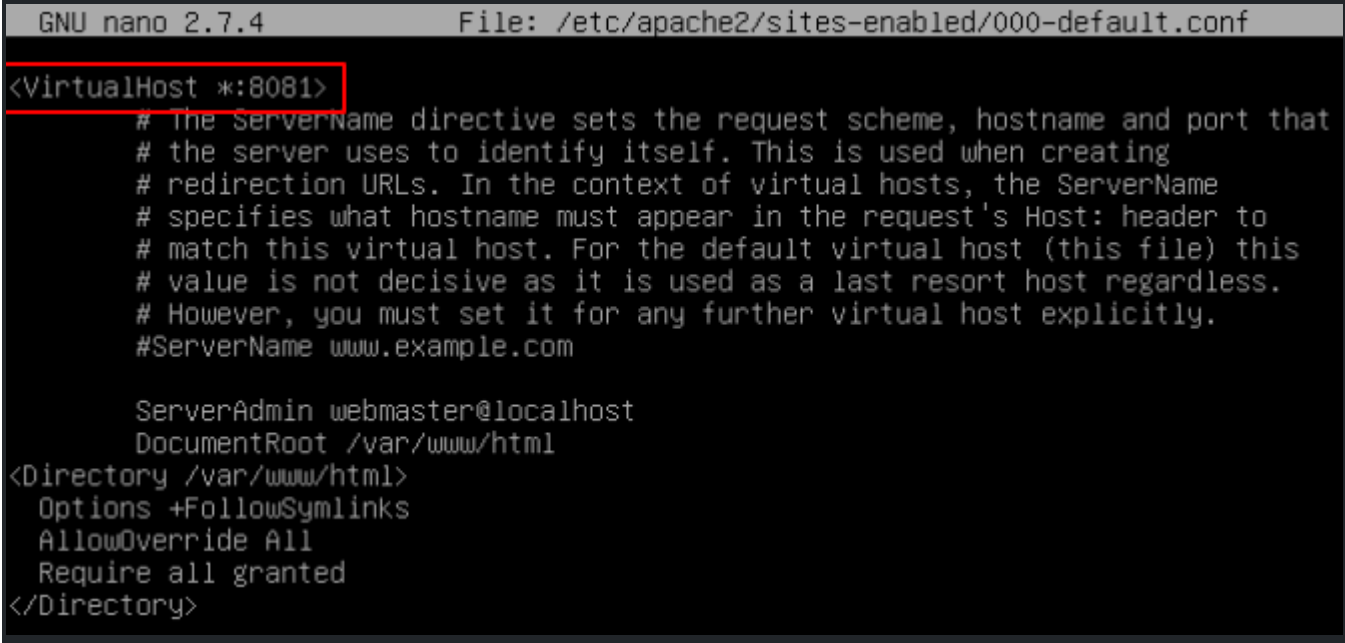
Change Apache Port on CentOS and RHEL

After you've added the above line, you need to create or alter an Apache virtual host in **Debian/Ubuntu** based distribution in order to start the binding process, specific to your own vhost requirements.

In **CentOS/RHEL** distributions, the change is applied directly into default virtual host. In the below sample, we'll modify the default virtual host of the web server and instruct Apache to listen for web traffic from **80** port to **8081** port.

Open and edit **000-default.conf** file and change the port to **8081** as shown in the below image.

```
# nano /etc/apache2/sites-enabled/000-default.conf
```



```
GNU nano 2.7.4      File: /etc/apache2/sites-enabled/000-default.conf
<VirtualHost *:8081>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
<Directory /var/www/html>
    Options +FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

Change Apache Port on Virtualhost

Finally, to apply changes and make Apache bind on the new port, restart the daemon and check local network sockets table using [netstat](#) or **ss command**. Port **8081** in listening should be displayed in your server network table.

```
# systemctl restart apache2
# netstat -tlnp| grep apache
# ss -tlnp| grep apache
```

```

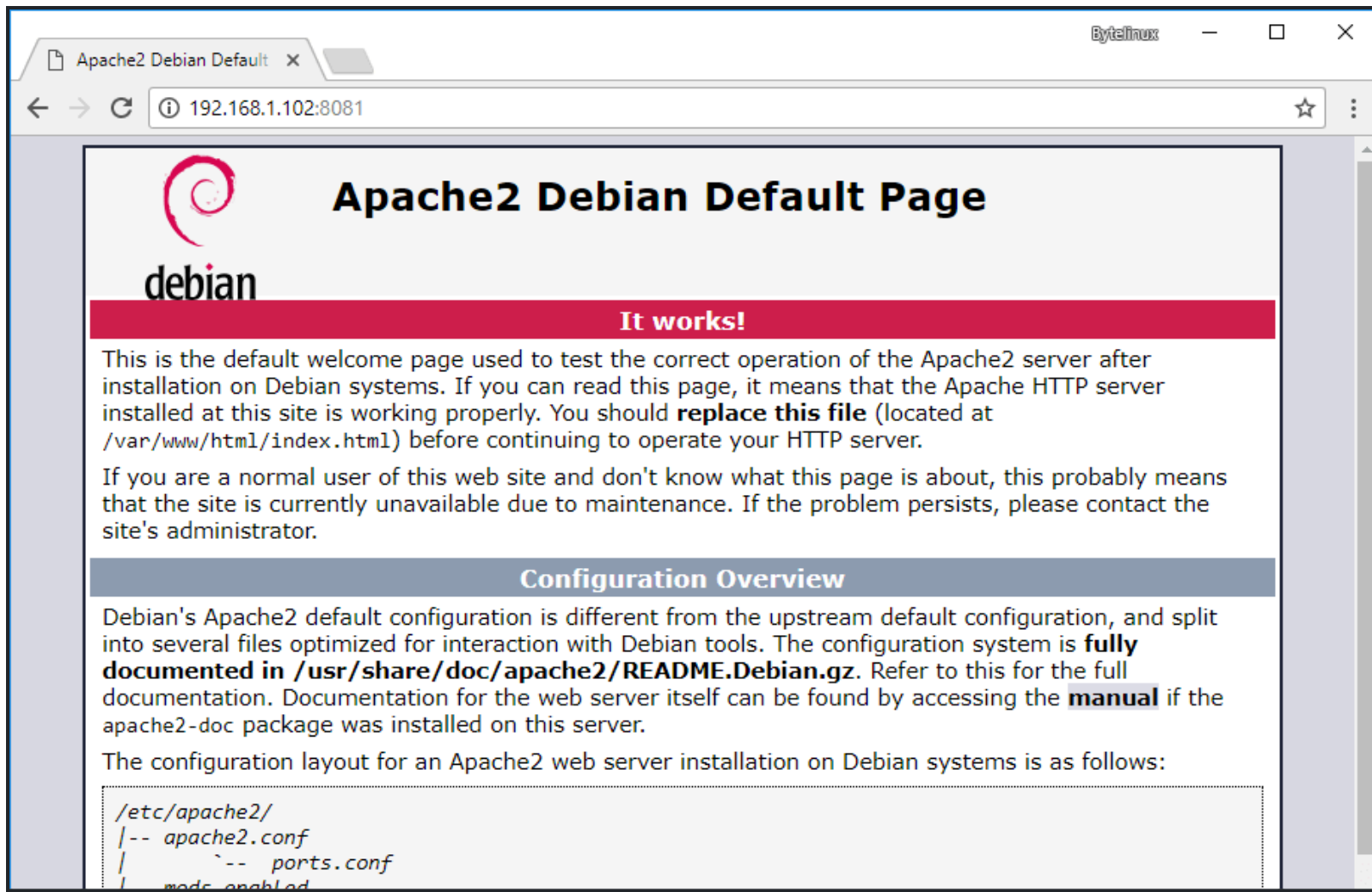
root@www:~# systemctl restart apache2
root@www:~# netstat -tlnp| grep apache
tcp6      0      0 :::80          :::*           LISTEN      21195/apache2
tcp6      0      0 :::8081        :::*           LISTEN      21195/apache2
tcp6      0      0 :::443         :::*           LISTEN      21195/apache2
root@www:~# ss -tlnp| grep apache
LISTEN    0      128          :::80          :::*           users:(("apache2
d=21200,fd=4),("apache2",pid=21199,fd=4),("apache2",pid=21198,fd=4),("apache2",pid=21197,fd=4),("
che2",pid=21196,fd=4),("apache2",pid=21195,fd=4))
LISTEN    0      128          :::8081        :::*           users:(("apache2
d=21200,fd=6),("apache2",pid=21199,fd=6),("apache2",pid=21198,fd=6),("apache2",pid=21197,fd=6),("
che2",pid=21196,fd=6),("apache2",pid=21195,fd=6))
LISTEN    0      128          :::443         :::*           users:(("apache2
d=21200,fd=8),("apache2",pid=21199,fd=8),("apache2",pid=21198,fd=8),("apache2",pid=21197,fd=8),("
che2",pid=21196,fd=8),("apache2",pid=21195,fd=8))
root@www:~# _

```

Verify Apache Port

You can also, open a browser and navigate to your server IP address or domain name on port **8081**. The Apache default page should be displayed in browser. However, if you cannot browse the webpage, return to server console and make sure the proper firewall rules are setup to allow the port traffic.

```
http://server.ip:8081
```



Apache Default Page on Debian and

Ubuntu

On **CentOS/RHEL** based Linux distribution install **policycoreutils** package in order to add the required SELinux rules for Apache to bind on the new port and restart Apache HTTP server to apply changes.

```
# yum install policycoreutils
```


Add Selinux rules for port **8081**.

```
# semanage port -a -t http_port_t -p tcp 8081
# semanage port -m -t http_port_t -p tcp 8081
```

Restart Apache web server

```
# systemctl restart httpd.service
```

Execute netstat or **ss command** to check if the new port successfully binds and listen for incoming traffic.

```
# netstat -tlnp | grep httpd
# ss -tlnp | grep httpd
```

```

[root@centos ~]# semanage port -a -t http_port_t -p tcp 8081
ValueError: Port tcp/8081 already defined
[root@centos ~]#
[root@centos ~]# semanage port -m -t http_port_t -p tcp 8081
[root@centos ~]#
[root@centos ~]#
[root@centos ~]# systemctl restart httpd.service
[root@centos ~]#
[root@centos ~]#
[root@centos ~]# netstat -tlnpi grep httpd
tcp6      0      0 :::80          :::*           LISTEN      1631/httpd
tcp6      0      0 :::8081        :::*           LISTEN      1631/httpd
[root@centos ~]#
[root@centos ~]#
[root@centos ~]# ss -tlnpi grep httpd
LISTEN    0      128          :::80          :::*           users: (("http
1636,fd=4), ("httpd",pid=1635,fd=4), ("httpd",pid=1634,fd=4), ("httpd",pid=1633,fd=4), ("httpd",p
,fd=4), ("httpd",pid=1631,fd=4))
LISTEN    0      128          :::8081        :::*           users: (("http
1636,fd=6), ("httpd",pid=1635,fd=6), ("httpd",pid=1634,fd=6), ("httpd",pid=1633,fd=6), ("httpd",p
,fd=6), ("httpd",pid=1631,fd=6))
[root@centos ~]# _

```

Check Apache Port on CentOS and RHEL

Open a browser and navigate to your server IP address or domain name on port **8081** to check is the new web port is reachable in your network. The Apache default page should be displayed in browser

```
http://server.ip:8081
```

If you cannot navigate to the above address, make sure you add the proper firewall rules in your server Firewall table.

3 Ways to Check Apache Server Status and Uptime in Linux

Aaron KiliSeptember 5, 2017

Categories

Apache, Monitoring Tools 1 Comment

Apache is a world's most popular, cross platform HTTP web server that is commonly used in Linux and Unix platforms to deploy and run web applications or websites. Importantly, it's easy to install and has a simple configuration as well.

Read Also: [How to Hide Apache Version Number and Other Sensitive Info](#)

In this article, we will show how to check Apache web server uptime on a Linux system using different methods/commands explained below.

1. Systemctl Utility

Systemctl is a utility for controlling the systemd system and service manager; it is used to start, restart, stop services and beyond. The systemctl status sub-command, as the name states is used to view the status of a service, you can use it for the above purpose like so:

```
$ sudo systemctl status apache2      #Debian/Ubuntu
# systemctl status httpd              #RHEL/CentOS/Fedora
```

```
aaronkilik@tecmin ~ $ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese
   Active: active (running) since Mon 2017-09-04 10:05:51 EAT; 4h 8min ago
     Process: 3030 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=
     Process: 1182 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SU
   Main PID: 1395 (apache2)
      Tasks: 7 (limit: 512)
    CGroup: /system.slice/apache2.service
           └─1395 /usr/sbin/apache2 -k start
              └─3037 /usr/sbin/apache2 -k start
                 └─3038 /usr/sbin/apache2 -k start
                    └─3039 /usr/sbin/apache2 -k start
                       └─3040 /usr/sbin/apache2 -k start
                          └─3041 /usr/sbin/apache2 -k start
                             └─4232 /usr/sbin/apache2 -k start

Sep 04 10:05:39 tecmint systemd[1]: Starting The Apache HTTP Server...
Sep 04 10:05:50 tecmint apachectl[1182]: AH00558: apache2: Could not reliably
Sep 04 10:05:51 tecmint systemd[1]: Started The Apache HTTP Server.
Sep 04 10:10:43 tecmint systemd[1]: Reloading The Apache HTTP Server.
Sep 04 10:10:43 tecmint apachectl[3030]: AH00558: apache2: Could not reliably
Sep 04 10:10:43 tecmint systemd[1]: Reloaded The Apache HTTP Server.
lines 1-22/22 (END)
```

Check Apache Status Using Systemctl

2. Apachectl Utilities

Apachectl is a control interface for Apache HTTP server. This method requires the **mod_status** (which displays info about the server is performing including its uptime) module installed and enabled (which is the default setting).

On Debian/Ubuntu

The **server-status** component is enabled by default using the file **/etc/apache2/mods-enabled/status.conf**.

```
$ sudo vi /etc/apache2/mods-enabled/status.conf
```

```

<IfModule mod_status.c>
    # Allow server status reports generated by mod_status,
    # with the URL of http://servername/server-status
    # Uncomment and change the "192.0.2.0/24" to allow access from other hosts.

    <Location /server-status>
        SetHandler server-status
        Require local
        #Require ip 192.0.2.0/24
    </Location>

    # Keep track of extended status information for each request
    ExtendedStatus On

    # Determine if mod_status displays the first 63 characters of a request or
    # the last 63, assuming the request itself is greater than 63 chars.
    # Default: Off
    #SeeRequestTail On

    <IfModule mod_proxy.c>
        # Show Proxy LoadBalancer status in mod_status
        ProxyStatus On
    </IfModule>

</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~
~

```

1,1

All

Apache Mod_Status Configuration

On RHEL/CentOS

To enable **server-status** component, create a file below.

```
# vi /etc/httpd/conf.d/server-status.conf
```

and add the following configuration.

```
<Location "/server-status">  
    SetHandler server-status  
    #Require host localhost      #uncomment to only allow requests from localhost  
</Location>
```

Save the file and close it. Then restart the web server.

```
# systemctl restart httpd
```

If you are primarily using a terminal, then you also need a [command line web browser such as lynx or links](#).


```
$ sudo apt install lynx      #Debian/Ubuntu
# yum install links          #RHEL/CentOS
```

Then run the command below to check the Apache service uptime:

```
$ apachectl status
```

```
aaronkilik@tecmint ~ $ apache2ctl status
Apache Server Status for localhost (via 127.0.0.1)
```

```
Server Version: Apache/2.4.27 (Ubuntu) OpenSSL/1.1.0f
Server MPM: prefork
Server Built: 2017-07-12T05:42:18
```

```
-----
Current Time: Monday, 04-Sep-2017 14:15:07 EAT
Restart Time: Monday, 04-Sep-2017 10:06:01 EAT
Parent Server Config. Generation: 2
Parent Server MPM Generation: 1
Server uptime: 4 hours 9 minutes 6 seconds
Server load: 1.04 0.99 1.01
Total accesses: 58 - Total Traffic: 120 kB
CPU Usage: u.03 s.03 cu0 cs0 - .000401% CPU load
.00388 requests/sec - 8 B/second - 2118 B/request
1 requests currently being processed, 5 idle workers
```

```
W_____.....
.....
.....
```

Scoreboard Key:

"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process

```
aaronkilik@tecmint ~ $
```

Check Apache Status Using Apache2ctl

Alternatively, use the URL below to view the Apache web server status information from a graphical web browser:

<http://localhost/server-status>

OR

```
http:SERVER IP/server-status
```

3. ps Utility

ps is a utility which shows information concerning a selection of the active processes running on a Linux system, you can use it with [grep command](#) to check Apache service uptime as follows.

Here, the flag:

- **-e** – enables selection of every processes on the system.
- **-o** – is used to specify output (comm – command, etime – process execution time and user – process owner).

```
# ps -eo comm,etime,user | grep apache2
```

```
# ps -eo comm,etime,user | grep root | grep apache2
```

OR

```
# ps -eo comm,etime,user | grep httpd
```

```
# ps -eo comm,etime,user | grep root | grep httpd
```

The sample output below shows that **apache2** service has been running for 4 hours, 10 minutes and 28 seconds (only consider the one started by root).

```
aaronkilik@tecmint ~ $ ps -eo comm,etime,user | grep apache2
apache2      04:10:28 root
apache2      04:05:36 www-data
apache2      04:05:36 www-data
apache2      04:05:36 www-data
apache2      04:05:36 www-data
apache2      04:05:36 www-data
apache2      04:02:58 www-data
aaronkilik@tecmint ~ $
aaronkilik@tecmint ~ $ ps -eo comm,etime,user | grep root | grep  apache2
apache2      04:10:36 root
aaronkilik@tecmint ~ $
```

Check Apache Uptime

Lastly, check out more useful Apache web server guides: