Apache Configuration Step By Step

Table of Contents

Copy

How To Install Apache From Source Code

```
Step by Step - Apache installation from source code

Step 1: Lets create a directory for storing all installable that apache needs:
mkdir -p /middleware/installables

Step 2: Download all prerequisite softwares/packages for apache under middleware/installables

Prerequisite packages/softwares:
1. Gcc and development tool
2. perl-IPC-Cmd
3. Apr
4. Apr-util
5. Openssl
6. Pcre
7. Apache-httpd
8. Expect

Out of these eight prerequisites, first two we can install using “yum install” or “apt-get install” and
remaining 5 packages we need to download from internet.
        yum groupinstall -y 'Development Tools'
        yum install perl-IPC-Cmd -y

Download below mentioned package from the URL given below

1. Apr –> https://apr.apache.org/download.cgi
2. Apr-util –> https://apr.apache.org/download.cgi
3. Apache-httpd –> https://httpd.apache.org/download.cgi
4. Pcre –> https://sourceforge.net/projects/pcre/files/pcre/
5. Openssl –> http://artfiles.org/openssl.org/source/
6. Expat –> https://github.com/libexpat/libexpat/releases
```

Once you have download all the above listed packages, you will have files similar to files mentioned below:

```
[root@apache-node1 installables]# ls -ltr
total 28028
        -rw-r--r--. 1 root root  1093896 Jan  9 17:32 apr-1.7.0.tar.gz
        -rw-r--r--. 1 root root   554301 Jan  9 17:32 apr-util-1.6.1.tar.gz
        -rw-r--r--. 1 root root  9719976 Jan  9 17:32 httpd-2.4.52.tar.gz
        -rw-r--r--. 1 root root 15011207 Jan  9 17:32 openssl-3.0.1.tar.gz
        -rw-r--r--. 1 root root  2309964 Jan  9 17:32 pcre-8.45.tar.gz
        -rw-r--r--. 1 root root  2309964 Jan  9 17:32 expat-2.4.2.tar.gz
```

Step 3: Lets extract all these packages using “tar -xvf” command

Example: tar -xvf apr-1.7.0.tar.gz

Once all the packages are extracted, our directory will look as shown below:

```
        drwxr-xr-x. 20 1001  1001 4096 Oct 18  2017 apr-util-1.6.1
        drwxr-xr-x. 27 1001  1001 4096 Apr  1  2019 apr-1.7.0
        drwxr-xr-x.  7 1169  1169 4096 Oct 29 16:05 pcre-8.45
        drwxrwxr-x. 20 root root  4096 Dec 14 16:16 openssl-3.0.1
        drwxr-xr-x. 12  504 games 4096 Dec 16 13:49 httpd-2.4.52
```

Step 4: Now lets start compiling all source codes required for apache, one by one.

A. Let’s set some environment variables first, that will help in compiling our source codes.

```
export LC_ALL=en_US.UTF-8
export LC_CTYPE=en_US.UTF-8
```

I will be installing apache under /middleware/apache2.4 directory, so let’s create a directory

```
mkdir -p /middleware/apache2.4
```

B. Will start with packages that we can easily install using yum module.

a. Compile & Install apr-1.7.0

cd /middleware/installables/apr-1.7.0

vi configure. (open this file in vi editor and search for “cfg-file")

change $RM "$cfgfile" to $RM -f “$cfgfile"

execute below mentioned command one by one:

```
./configure --prefix=/middleware/apache2.4/apr
make && make install
```

Once the above commands are executed, we will file that a new filesystem got created under our apache2.4 directory.

```
[root@apache-node1 apr-1.7.0]# ls -ltr /middleware/apache2.4/apr/
drwxr-xr-x. 3 root root  19 Jan  9 18:24 include
drwxr-xr-x. 3 root root 140 Jan  9 18:24 lib
drwxr-xr-x. 2 root root 108 Jan  9 18:24 build-1
drwxr-xr-x. 2 root root  26 Jan  9 18:24 bin
```

b. Comple & Install expat-2.4.2

```
cd /middleware/installables/expat-2.4.2
```

execute below mentioned command one by one:

```
./configure --prefix=/middleware/apache2.4/expat
make && make install
```

c. Comple & Install apr-util-1.6.1

```
cd /middleware/installables/apr-util-1.6.1
```

execute below mentioned command one by one:

```
./configure --prefix=/middleware/apache2.4/apr-util --with-apr=/middleware/apache2.4/apr --with-expat=/middleware/apache2.4/expat
make && make install
```

d. Compile and Install pcre-8.45

```
cd /middleware/installables/pcre-8.45
```

execute below mentioned command one by one:

```
./configure --prefix=/middleware/apache2.4/pcre
make && make install
```

e. Compile and Install penssl-3.0.1

```
cd /middleware/installables/openssl-3.0.1
```

execute below mentioned command one by one:

```
./configure -fPIC --prefix=/middleware/apache2.4/openssl
make && make install
```

f. Compile and Install httpd-2.4.52

```
cd /middleware/installables/httpd-2.4.52
```

Execute the below mentioned commands one by one.

```
./configure --prefix=/middleware/apache2.4/ --with-apr=/middleware/apache2.4/apr --with-apr-util=/middleware/apache2.4/apr-util --with-pcre=/middleware/apache2.4/pcre --with-ssl=/middleware/apache2.4/openssl --with-expat=/middleware/apache2.4/expat
make && make install
```

Step 5: It’s time to start apache.

```
cd middleware/apache2.4/bin
./apachectl -k start
```

Check the apache process by executing “ ps -ef | grep httpd”

```
root@apache-node1 bin]# ps -ef | grep httpd
root      175745       1  0 19:47 ?        00:00:00 /middleware/apache2.4//bin/httpd -k start
daemon    175746  175745  0 19:47 ?        00:00:00 /middleware/apache2.4//bin/httpd -k start
daemon    175747  175745  0 19:47 ?        00:00:00 /middleware/apache2.4//bin/httpd -k start
daemon    175748  175745  0 19:47 ?        00:00:00 /middleware/apache2.4//bin/httpd -k start
daemon    175833  175745  0 19:48 ?        00:00:00 /middleware/apache2.4//bin/httpd -k start
```

The directory structure under/middleware/apache2.4 will look like as shown below:

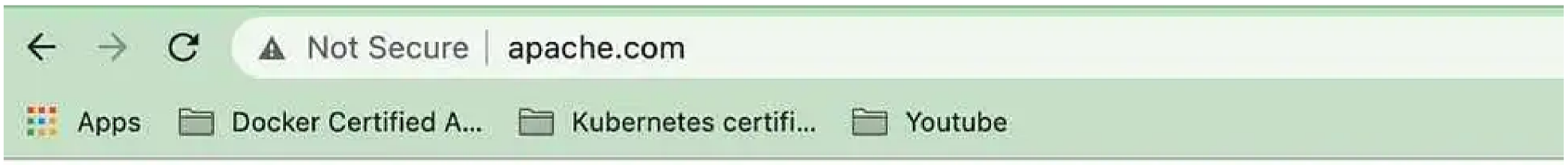
```
[root@apache-node1 apache2.4]# pwd
/middleware/apache2.4
[root@apache-node1 apache2.4]# ls -ltr
total 36
drwxr-xr-x.  2 root root   24 Jan  9 17:46 htdocs
drwxr-xr-x. 14 root root 8192 Jan  9 17:46 manual
drwxr-xr-x.  6 root root   58 Jan  9 18:24 apr
drwxr-xr-x.  6 root root   56 Jan  9 18:48 expat
drwxr-xr-x.  5 root root   43 Jan  9 18:49 apr-util
drwxr-xr-x.  7 root root   69 Jan  9 19:07 openssl
```

```
drwxr-xr-x.  6 root root   56 Jan  9 19:40 pcre
drwxr-xr-x.  2 root root 4096 Jan  9 19:44 modules
drwxr-xr-x.  2 root root  262 Jan  9 19:44 bin
drwxr-xr-x.  4 root root   84 Jan  9 19:44 conf
drwxr-xr-x.  3 root root 4096 Jan  9 19:44 error
drwxr-xr-x.  3 root root 8192 Jan  9 19:44 icons
drwxr-xr-x.  2 root root   78 Jan  9 19:44 cgi-bin
drwxr-xr-x.  2 root root 4096 Jan  9 19:44 include
drwxr-xr-x.  2 root root  167 Jan  9 19:44 build
drwxr-xr-x.  4 root root   30 Jan  9 19:44 man
drwxr-xr-x.  2 root root   58 Jan  9 19:47 logs
[root@apache-node1 apache2.4]#
```

How To Define Document Root

Copy

```
If we have followed the previous step ( Installing Apache from Source code and starting apache instance
), we will get default page on accessing apache over default port 80.
```



Now lets see

Importance of MPM module – Prefork vs Worker

By default, Apache is configured in preforked mode, non-threaded pre-forking web server. That means that each Apache child process contains a single thread and handles one request at a time. Because of that, it consumes more resources.

Prefork

With the Prefork module installed, Apache is a non-threaded, pre-forking web server. That means that each Apache child process contains a single thread and handles one request at a time. Because of that, it consumes more resources than the threaded MPMs: Worker and Event.

Prefork is the default MPM, so if no MPM is selected in EasyApache, Prefork will be selected. It still is the best choice if Apache has to use non-thread safe libraries such as mod_php (DSO), and is ideal if isolation of processes is important.

Worker

The Worker MPM turns Apache into a multi-process, multi-threaded web server. Unlike Prefork, each child process under Worker can have multiple threads. As such, Worker can handle more requests with fewer resources than Prefork. Worker generally is recommended for high-traffic servers

Apache Modules

There are around 125 modules in Apache 2.4, some most popular and commonly used modules are highlighted below. I will try to cover practical on most of these modules based on my experience.

How to configure Port, Domain & IP under virtual host

PFB Screenshot from a working environment, it’s simple to configure.

```
Listen 147.182.134.166:80
Listen 147.182.134.167:81
Listen 147.182.134.168:82
Listen apache.com:83

<VirtualHost 147.182.134.166:80>
ServerName 147.182.134.167
        DocumentRoot "/middleware/htdocs/server1"
        ErrorLog /middleware/logs/server1_error.log
        CustomLog /middleware/logs/server1_access.log common

<Directory "/middleware/htdocs/server1">
        Require all granted
</Directory>
</VirtualHost>

<VirtualHost 147.182.134.167:81>
ServerName 147.182.134.168
        DocumentRoot "/middleware/htdocs/server2"
        ErrorLog /middleware/logs/server2_error.log
        CustomLog /middleware/logs/server2_access.log common

<Directory "/middleware/htdocs/server2">
        Require all granted
</Directory>
</VirtualHost>

<VirtualHost 147.182.134.168:82>
ServerName 147.182.134.168
        DocumentRoot "/middleware/htdocs/server3"
        ErrorLog /middleware/logs/server3_error.log
        CustomLog /middleware/logs/server3_access.log common

<Directory "/middleware/htdocs/server3">
        Require all granted
</Directory>
</VirtualHost>

<VirtualHost apache.com:83>
ServerName apache.com
        DocumentRoot "/middleware/htdocs/server4"
        ErrorLog /middleware/logs/server4_error.log
```

How to configure ALIAS for URL/Page

Copy

```
<VirtualHost apache.com:83>
ServerName apache.com
        DocumentRoot "/middleware/htdocs/server4"
        ErrorLog /middleware/logs/server4_error.log
        CustomLog /middleware/logs/server4_access.log common

<Directory "/middleware/htdocs/server4">
        Require all granted
</Directory>

Alias /alias-demo /middleware/apache2.4/conf/website1/johny.html
<Directory "/middleware/apache2.4/conf/website1/">
        Require all granted
</Directory>
```

```
</VirtualHost>
```

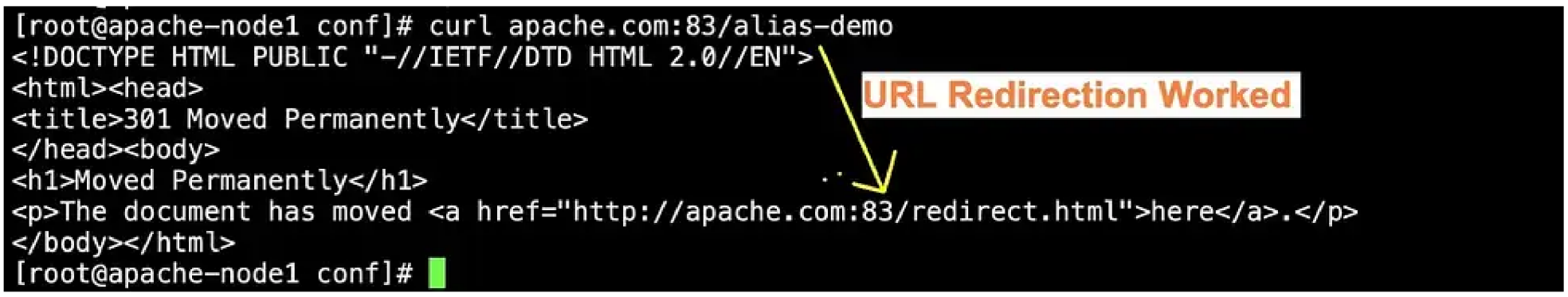
In the above post, I have set alias for /alias-demo context root to serve the page from a custom directory which is not part of document root.
I updated the cose as shown above and restarted the apache server. PFB the output.

How to redirect URL permanently

Copy

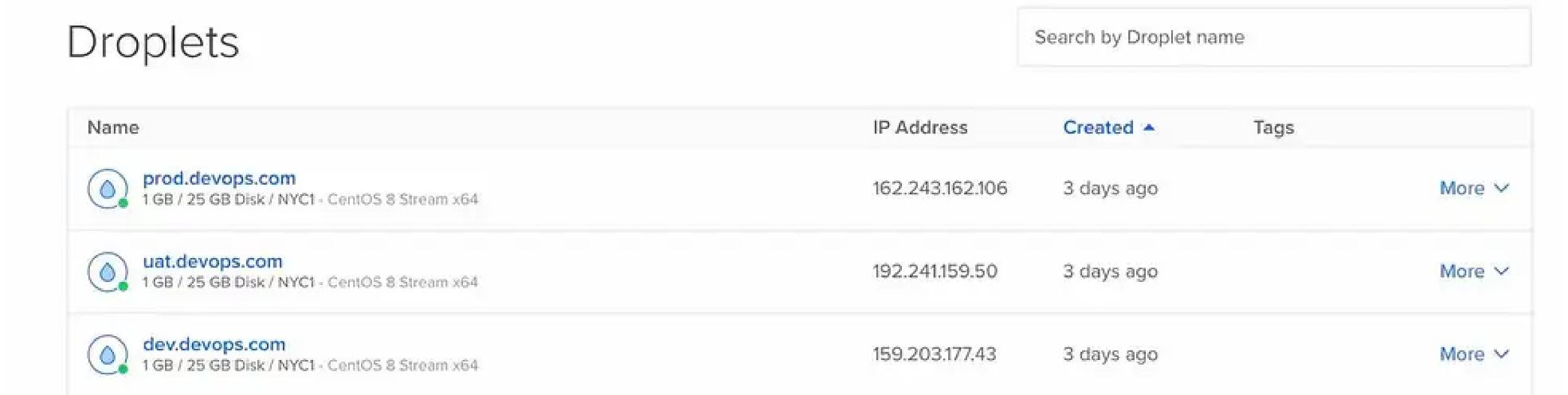
```
</VirtualHost>

<VirtualHost apache.com:83>
ServerName apache.com
        DocumentRoot "/middleware/htdocs/server4"
        ErrorLog /middleware/logs/server4_error.log
        CustomLog /middleware/logs/server4_access.log common

<Directory "/middleware/htdocs/server4">
        Require all granted
</Directory>
Alias /alias-demo /middleware/apache2.4/conf/website1/johny.html
<Directory "/middleware/apache2.4/conf/website1/">
        Require all granted
</Directory>

Redirect permanent "/alias-demo" "http://apache.com:83/redirect.html"

</VirtualHost>
```

After adding the above code, just restart apache server and test.

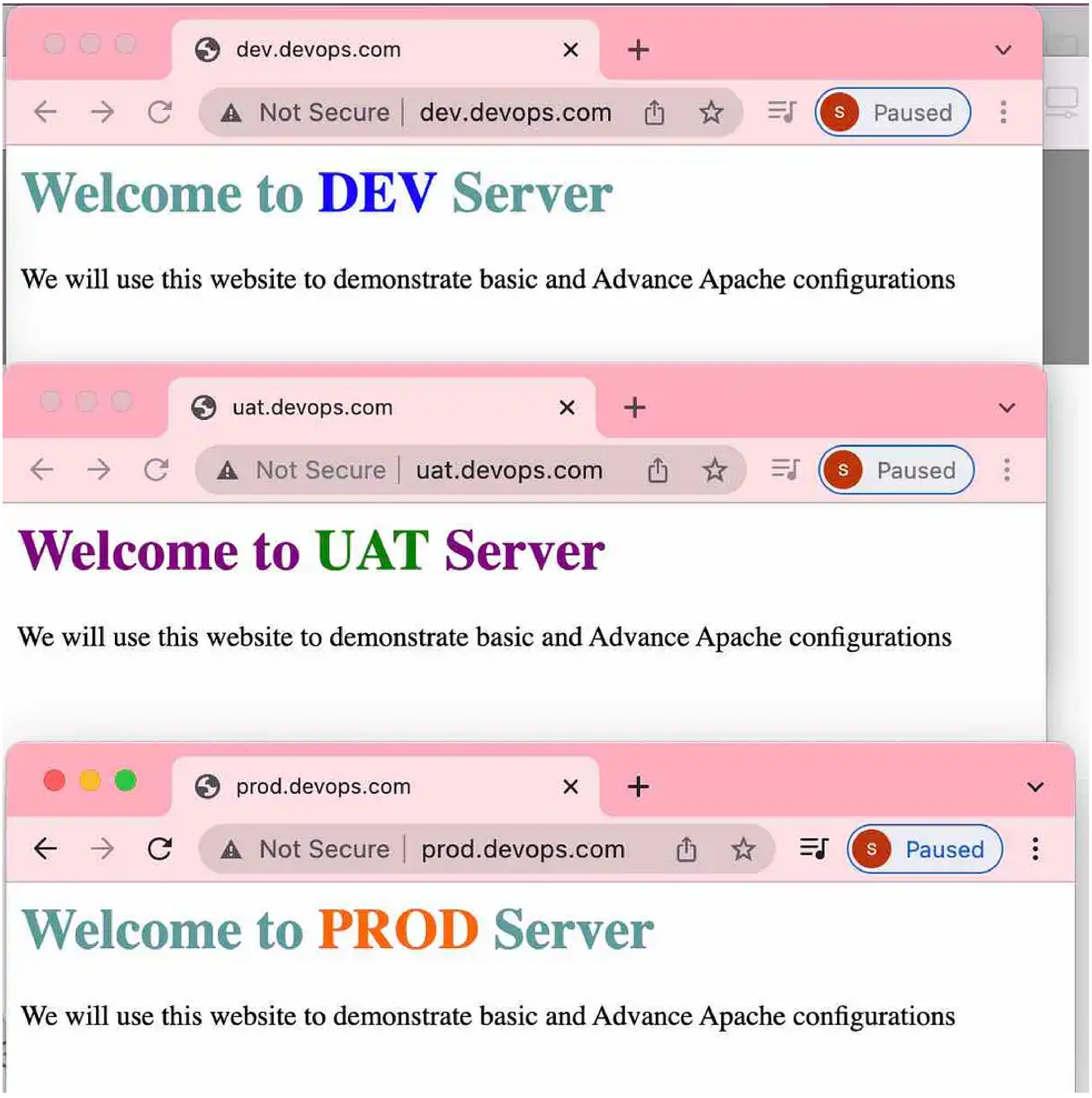


How to Configure ProxyPass and ProxyPassReverse in Apache

To demonstrate this exercise, I have setup DEV, UAT and PROD environment running on three different servers. Each server has apache installed and service a simple http page.



I am able to access all my environments, as shown in picture below:



As part of this exercise, I was to configure my dev environment to serve pages (content) from UAT server. Let's say, if anyone tries to access http://dev.devops.com/ then apache should get the content from UAT server and serve to the users.

Note: We are not redirecting URL here, URL redirection can be identified by user but ProxyPass & ProxyPassreverse will not be noticed by the application user. They will feel as if content is actually being served by dev server only.

Step1: Enable below mentioned two modules under httpd.conf file:

1- mod_proxy – it is the main module responsible for redirecting the connections,

2- mod_proxy_http – add the support for proxying HTTP connections,

```
#LoadModule remoteip_module modules/mod_remoteip.so
LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
```

Step 2: Update the apache configuration as highlighted below:

Copy

```
Listen 80
ServerName dev.devops.com:80

<VirtualHost dev.devops.com:80>
ServerName dev.devops.com

        DocumentRoot "/middleware/htdocs/dev"
        ErrorLog /middleware/logs/dev_error.log
        CustomLog /middleware/logs/dev_access.log common

<Directory "/middleware/htdocs/dev">
        Require all granted
</Directory>

ProxyPass / http://uat.devops.com:80/
ProxyPassReverse / http://uat.devops.com:80/

</VirtualHost>
```

Step 3: Restart the apache server and test the dev server url (dev.devops.com).
The below mentioned image shows that the request coming for dev.devops.com is being served by uat.devops.com.

A funny example –:> It is just like, when a teacher calls name of student-1 (while making attendance) the response is provided by student-2. We use to call that proxy attendance.

How To List Enable & Disable Directory browsing

Let’s say you have a requirement, to list files and directories of any given path on the server. For example, let’s display all files and directories of /tmp. Once directory listing is enabled, we will also see how to disable it.

Copy

```
ServerRoot "/middleware/apache2.4/"
Include conf/modules.conf
Listen 80
ServerName dev.devops.com:80
```

```
<VirtualHost dev.devops.com:80>
ServerName dev.devops.com

        DocumentRoot "/middleware/htdocs/"
        ErrorLog /middleware/logs/error.log
        CustomLog /middleware/logs/access.log common


        <Directory "/middleware/htdocs/">
        Require all granted
        </Directory>

Alias /list-directory /tmp/
        <Directory "/tmp/">
        Options Indexes
        AllowOverride none
        Require all Granted
        </Directory>
</VirtualHost>
```

Lets restart Apache and try to access the URL with context root “**list-directory**“

Index of /list-directory

- Parent Directory
- .ICE-unix/
- .Test-unix/
- .X11-unix/
- .XIM-unix/
- .font-unix/
- apache2.4.tar
- apache2.4_uat.tar
- snap.lxd/
- systemd-private-3c624e18e6d84d4d92896885ebdadc30-systemd-logind.service-hrJxei/
- systemd-private-3c624e18e6d84d4d92896885ebdadc30-systemd-resolved.service-0TgZFf/
- systemd-private-3c624e18e6d84d4d92896885ebdadc30-systemd-timesyncd.service-oP1NQh/
- test1/
- tmptr9zv28x/

To Disable the directory listing replace “Indexes” by “-Indexes”

Copy

```
Alias /list-directory /tmp/
        <Directory "/tmp/">
        Options -Indexes
        AllowOverride none
        Require all Granted
        </Directory>
```

How to Deny access to some directories to the user.

Lets say I have three websites (three directories having index.html) on the server and i want to hide one of the directory. Lets say I want userver not to access app2 directory.

```
root@kube-worker-01:/middleware/htdocs# pwd
/middleware/htdocs
root@kube-worker-01:/middleware/htdocs# tree
.
├── app1
│   └── index.html
├── app2
│   └── index.html
├── app3
│   └── index.html
└── index.html

3 directories, 4 files
root@kube-worker-01:/middleware/htdocs#
```

Lets say,I do not users to access app2.
Add below mentioned three lines of code and restart the apache.

```
<VirtualHost dev.devops.com:80>
ServerName dev.devops.com

        DocumentRoot "/middleware/htdocs/"
        ErrorLog /middleware/logs/error.log
        CustomLog /middleware/logs/access.log common


        <Directory "/middleware/htdocs/">
        Require all granted
        </Directory>

 <DirectoryMatch /app2>
 Require all Denied
 </DirectoryMatch>

Alias /list-directory /tmp/
        <Directory "/tmp/">
        Options Indexes
        AllowOverride none
        Require all Granted
        </Directory>
</VirtualHost>
```

Forbidden

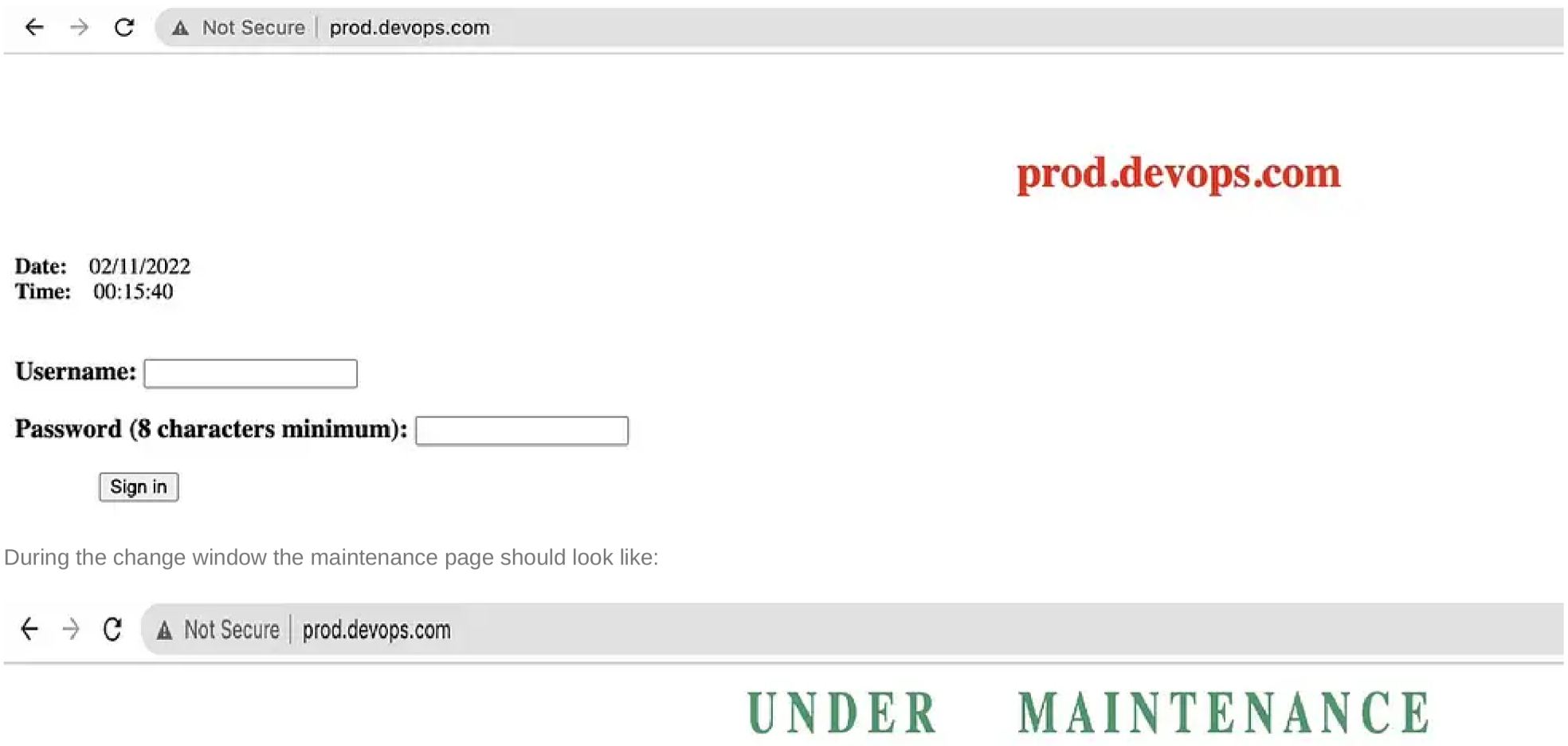
You don't have permission to access this resource.

How To Display a static Maintenance Page during a change window

Scenario : On the upcoming weekend, you have a change in the production environment and there will be downtime for a few minutes. The project manager wants you to configure your apache server to meet below mentioned two scenarios:

A. The end-users should not get 503 error page instead, they should get a maintenance page when accessing the URL (http://prod.devops.com), while change implementation and internal testing are in progress.
B. The same URL (http://prod.devops.com) should show the actual website to selected developers and testers during the change window.

Before Change window the website looks like this:



During the change window the maintenance page should look like:

To Achieve this goal, we need to follow below mentioned approach:

Step 1: Add the **highlighted** code under the virtual host section in httpd.conf file.

```
Listen 80
ServerName prod.devops.com:80

<VirtualHost prod.devops.com:80>
ServerName prod.devops.com

        DocumentRoot "/middleware/htdocs/"
        ErrorLog /middleware/logs/error.log
        CustomLog /middleware/logs/access.log common

<Directory "/middleware/htdocs/">
        Require all granted
</Directory>
LoadModule rewrite_module modules/mod_rewrite.so
RewriteEngine On
RewriteCond %{REMOTE_ADDR} !^147\.182\.172\.234
RewriteCond %{DOCUMENT_ROOT}/maintenance.html -f
RewriteCond %{DOCUMENT_ROOT}/maintenance.enable -f
```

```
RewriteCond %{SCRIPT_FILENAME} !maintenance.html
RewriteRule ^.*$ /maintenance.html [R=503,L]
ErrorDocument 503 /maintenance.html
Header Set Cache-Control "max-age=0, no-store"

</VirtualHost>
```

Step 2: Under your document root (In my case document root is “/middleware/htdocs/”) , create two below mentioned three files
index.html : You might already have this page for your application
maintenance.html: Developer should provide (You can also create a simple HTML page) HTML page with the content that they want to display during maintenance window.
maintenance.enable: This is just an empty file, until this file exists under the document root directory, users will not be able to access the actual website. So, once the change window is over, remove/delete this maintenance.enable file.

Step 3: Restart apache, we should get the maintenance page now. So, our first goal is achieved. End users will get a custom maintenance page during the change window.
Step 4: Under httpd.conf file, we have added a host entry (shown below), here the IP address (147.182.172.234) belongs to a developer’s computer. During change window, he will not get the maintenance page. Once the change is completed and testing is in progress, only that particular developer will get actual prod application when accessing **http://p**rod.devops.com/ .

```
RewriteCond %{REMOTE_ADDR} !^147\.182\.172\.234
```

Copy

If you want to add more IPs (multiple developer or testers) then just multiple the above entry in the httpd.conf file.

```
RewriteCond %{REMOTE_ADDR} !^147\.182\.172\.234
RewriteCond %{REMOTE_ADDR} !^147\.182\.172\.235
RewriteCond %{REMOTE_ADDR} !^147\.182\.172\.236
```

Copy

So, our second goal is also completed

Explanation

1. Turn on the Rewrite Engine.

```
RewriteEngine On
```

Copy

2. (optional) Don’t match your IP address. Use this directive to prevent redirection to maintenance.html for traffic from listed IP address.

```
RewriteCond %{REMOTE_ADDR} !^123\.456\.789\.000
```

Copy

You can set several IP addresses line by line, as follows:

```
RewriteCond %{REMOTE_ADDR} !^192\.168\.0\.1
RewriteCond %{REMOTE_ADDR} !^192\.168\.1\.100
RewriteCond %{REMOTE_ADDR} !^172\.16\.10\.15
```

Copy

3. Make sure the maintenance.html page exists.

```
RewriteCond %{DOCUMENT_ROOT}/maintenance.html -f
```

Copy

4. Check for the maintenance.enable file (this is how you turn the maintenance page on and off).

```
RewriteCond %{DOCUMENT_ROOT}/maintenance.enable -f
```

Copy

5. Don’t apply the rule when serving the maintenance page (avoids circular rewrites).

```
RewriteCond %{SCRIPT_FILENAME} !maintenance.html
```

Copy

6. The 503 redirect to maintenance page itself.

```
RewriteRule ^.*$ /maintenance.html [R=503,L]
ErrorDocument 503 /maintenance.html
```

Copy

503 Service Unavailable – HTTP status code, which means that the server is currently unavailable (because it is overloaded or down for maintenance).

7. Avoid caching.

```
Header Set Cache-Control "max-age=0, no-store"
```

Copy

How to generate Self-signed certificate and configure SSL in Apache

TLS/SSL works by using a combination of a public certificate and a private key. The SSL key is kept secret on the server. It is used to encrypt content sent to clients. The SSL certificate is publicly shared with anyone requesting the content. It can be used to decrypt the content signed by the associated SSL key.

We can create a self-signed key and certificate pair with OpenSSL in a single command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /middleware/certs/prod_devops_com.key -out
/middleware/certs/prod_devops_com.crt
```

Copy

Explanation:

- **openssl**: This is the basic command line tool for creating and managing OpenSSL certificates, keys, and other files.
- **req**: This subcommand specifies that we want to use X.509 certificate signing request (CSR) management. The “X.509” is a public key infrastructure standard that SSL and TLS adheres to for its key and certificate management. We want to create a new X.509 cert, so we are using this subcommand.
- **–nodes**: This tells OpenSSL to skip the option to secure our certificate with a passphrase. We need Apache to be able to read the file, without user intervention, when the server starts up. A passphrase would prevent this from happening because we would have to enter it after every restart.
- **–days 365**: This option sets the length of time that the certificate will be considered valid. We set it for one year here.
- **-newkey rsa:2048**: This specifies that we want to generate a new certificate and a new key at the same time. We did not create the key that is required to sign the certificate in a previous step, so we need to create it along with the certificate. The `rsa:2048` portion tells it to make an RSA key that is 2048 bits long.
- **-keyout**: This line tells OpenSSL where to place the generated private key file that we are creating.
- **-out**: This tells OpenSSL where to place the certificate that we are creating.

```
root@kube-master:/middleware/certs# pwd
/middleware/certs
root@kube-master:/middleware/certs# openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /middleware/certs/prod_devops_com.key -out /middleware/c
erts/prod_devops_com.crt
Generating a RSA private key
........+++++
......+++++
writing new private key to '/middleware/certs/prod_devops_com.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:prod.devops.com
Email Address []:
root@kube-master:/middleware/certs# ls -ltr
total 8
-rw------- 1 root root 1704 Feb 11 16:05 prod_devops_com.key
-rw-r--r-- 1 root root 1318 Feb 11 16:05 prod_devops_com.crt
root@kube-master:/middleware/certs#
```

Step 2: Update httpd.conf file to receive requests on port 443. We can either add configuration in httpd.conf file or create a separate configuration file and include the new file in httpd.conf file.

I will prefer to create a file by name SSL;conf under conf directory.

```
/middleware/apache2.4/conf
root@kube-master:/middleware/apache2.4/conf# ls -ltr
total 116
drwxr-xr-x 3 root root  4096 Jan  9 19:44 original
-rw-r--r-- 1 root root 60847 Jan  9 19:44 mime.types
-rw-r--r-- 1 root root 13064 Jan  9 19:44 magic
drwxr-xr-x 2 root root  4096 Jan  9 19:44 extra
-rw-r--r-- 1 root root  4769 Feb 10 17:12 modules.conf
-rwxr-xr-x 1 root root   672 Feb 11 19:34 ssl.conf
-rw-r--r-- 1 root root 17608 Feb 11 19:35 httpd.conf
root@kube-master:/middleware/apache2.4/conf#
```

Under ssl.conf, I have updated SSL configuration.

```
root@kube-master:/middleware/apache2.4/conf# cat ssl.conf
<VirtualHost prod.devops.com:443>
ServerName prod.devops.com


SSLEngine on

SSLCertificateFile     /middleware/certs/prod_devops_com.crt
SSLCertificateKeyFile /middleware/certs/prod_devops_com.key


  DocumentRoot "/middleware/htdocs/"
  ErrorLog /middleware/logs/error.log
  CustomLog /middleware/logs/access.log common

 <Directory "/middleware/htdocs/">
    Require all granted
 </Directory>
```

Step 3: Now include ssl.conf under httpd.conf file.

```
#Listen 12.34.56.78:80
Listen 443
ServerName prod.devops.com
Include conf/ssl.conf
```

Step 4: Restart apache and validate.

I was getting this error during the restart. To resolve this we need to export open SSL library.

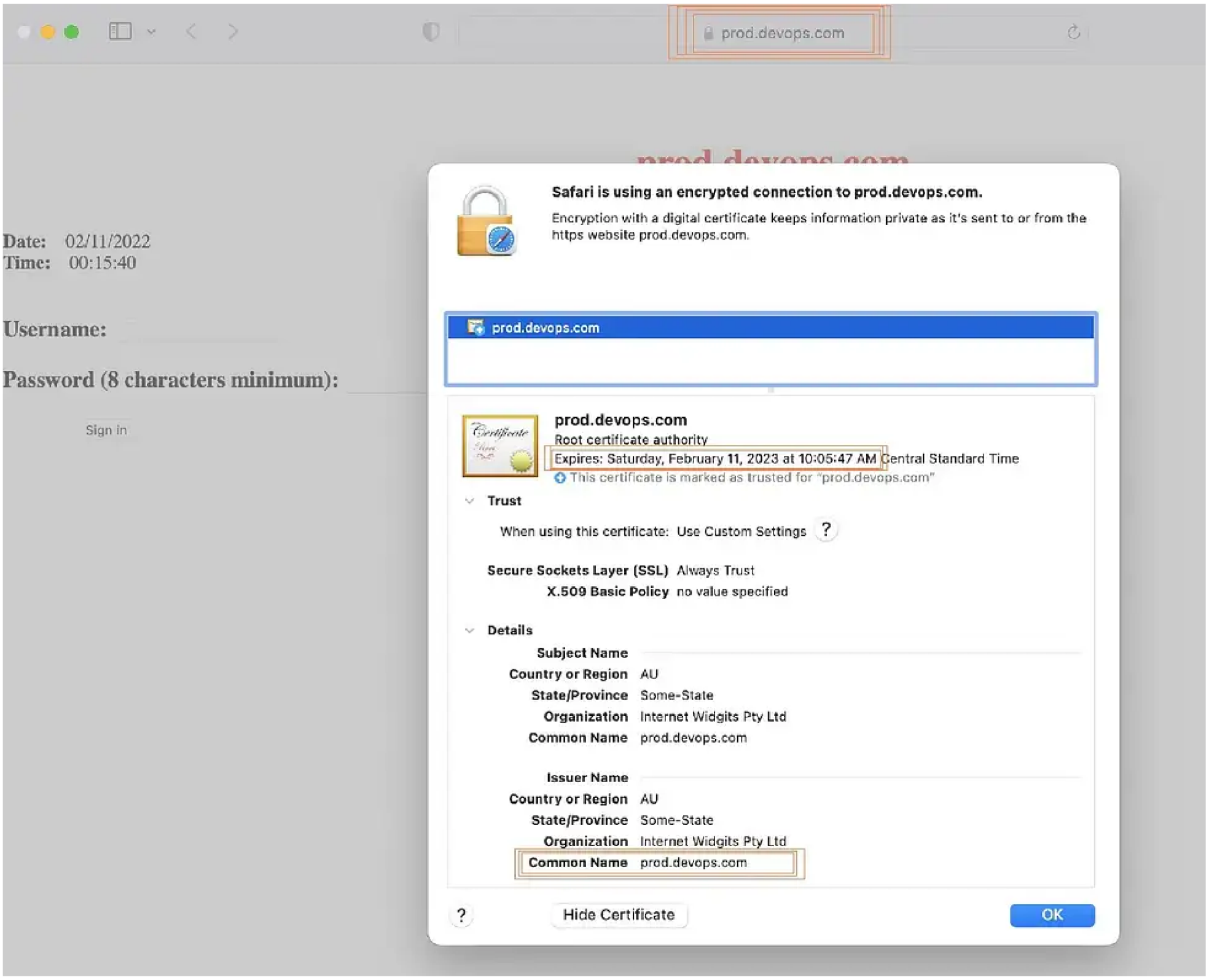
```
root@kube-master:/middleware/apache2.4/bin# ./apachectl -k start
httpd: Syntax error on line 137 of /middleware/apache2.4/conf/httpd.conf: Cannot load /middleware/apache2.4/modules/mod_ssl.so into server: libssl.so.3:
 cannot open shared object file: No such file or directory
root@kube-master:/middleware/apache2.4/bin#
```

export LD_LIBRARY_PATH=”/middleware/apache2.4/openssl/lib64”

Now the error that we were getting has resolved now.

```
root@kube-master:/middleware/apache2.4/bin# export LD_LIBRARY_PATH="/middleware/apache2.4/openssl/lib64"
root@kube-master:/middleware/apache2.4/bin# ./apachectl -k start
root@kube-master:/middleware/apache2.4/bin#
```

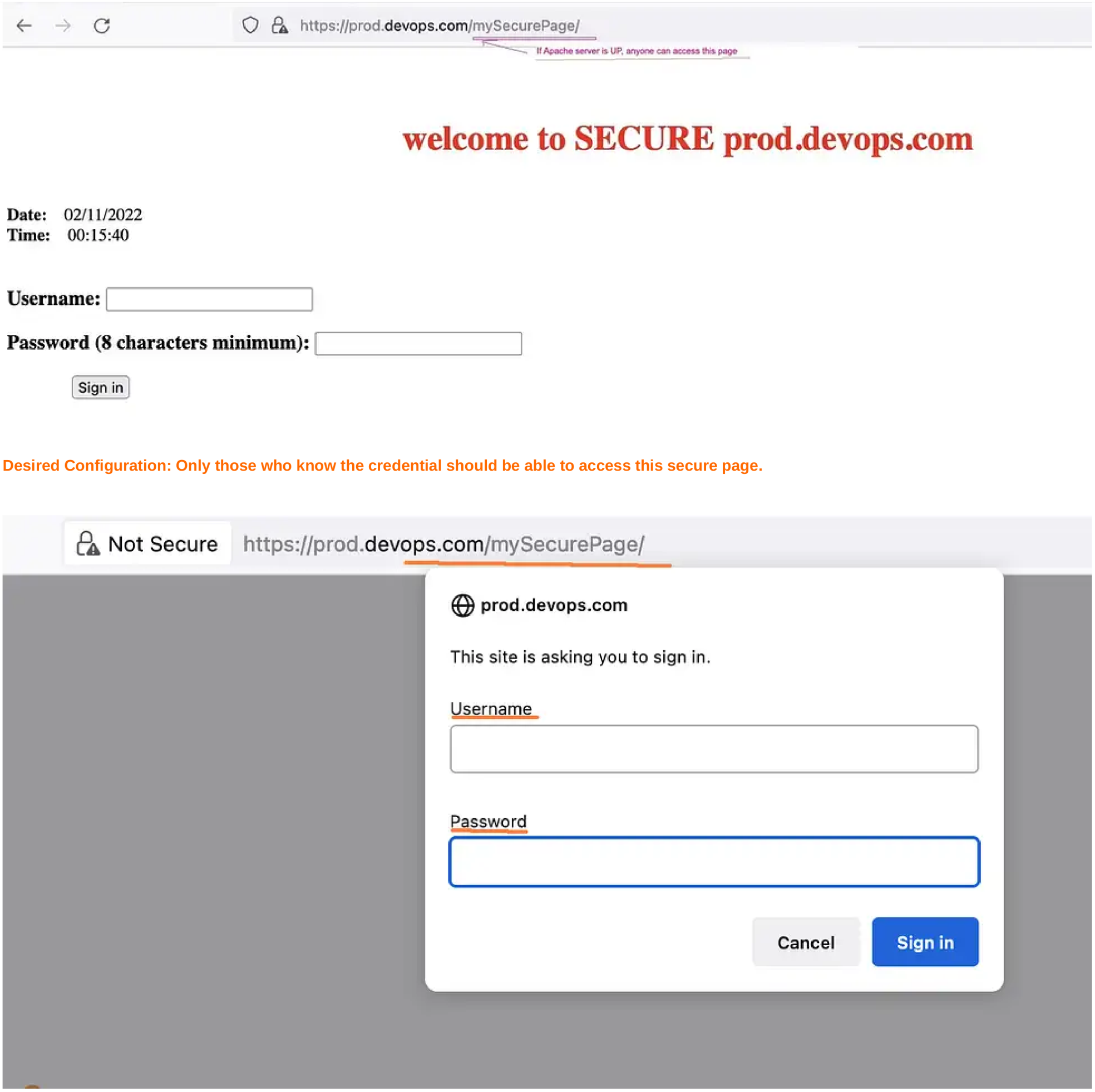
Lets validate the URL and certificate.



How to lock any WebPage with User id and Password

Scenario: You are configuring apache for one application. The application has some secure page, whose access is limited to some selected persons only. Those secure pages are not protected by any authentication mechanism. App Team wants you to configure apache in such a way that when a user try to access those pages then apache should challenge the user to authenticate.

Existing configuration: No authentication is configured. Everyone who knows the URL can access this page



Desired Configuration: Only those who know the credential should be able to access this secure page.

Configuration **Steps:**

Step 1: Create a separate file under the apache conf directory by name of your secure website. In my case I have created mySecurePage.conf (You can choose any name you want):

```
cat mySecurePage.conf
 LoadModule auth_basic_module modules/mod_auth_basic.so
 Alias /mySecurePage /middleware/htdocs/secureWebsite

 <Directory "/middleware/htdocs/secureWebsite/">
   AuthType Basic
   AuthName "Restricted content"
   AuthUserFile "conf/htpasswd.user"
   Require valid-user
 </Directory>
```

Copy

Step 2: Include mySecurePage.conf under the apache httpd.conf file

```
Listen 443
ServerName prod.devops.com
<VirtualHost prod.devops.com:443>
Include conf/ssl.conf
Include conf/mySecurePage.conf
</VirtualHost>
```

Note: It's a good practice to keep httpd.conf file as clean and small as possible. Create additional conf file (for example mySecurePage.conf) and include them under httpd.conf

Step 3: Now let's create a file on the apache server (OS) which will have an encrypted user id and password.
Under apache bid directory there an executable file "htpasswd" that can create encrypted credentials under htpasswd.user.

In the below example, prod-user2 is my username

```
/middleware/apache2.4/bin/htpasswd -c /middleware/apache2.4/conf/htpasswd.user prod-user2
```

Step 3: Restart the apache server and validate your change. You will see your secure website is asking you for credentials (user name and password).

How to check Server-Status page in Apache or how much load Apache is handling

Step 1: Create a file by name serverStatus.conf (you can choose any file name) with below-mentioned code.

```
root@kube-master:/middleware/apache2.4/conf# cat serverStatus.conf
LoadModule status_module modules/mod_status.so
<Location /server-status>
  SetHandler server-status
  Require local
  Require ip 76.184.214.252
</Location>
```

Step 2: Include serverStatus.conf under httpd.conf and restart the apache.

```
Include conf/serverStatus.conf
```

Step 3: Try to access server-status page (https://prod.devops.com/server-status)

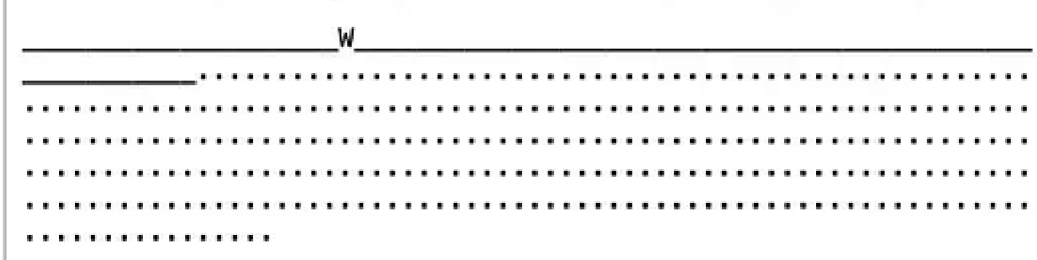
https://prod.devops.com/server-status

Apache Server Status for prod.devops.com (via 137.184.149.102)

Server Version: Apache/2.4.52 (Unix) OpenSSL/3.0.1
Server MPM: event
Server Built: Jan 9 2022 19:42:10

Current Time: Saturday, 12-Feb-2022 20:33:40 UTC
Restart Time: Saturday, 12-Feb-2022 20:31:22 UTC
Parent Server Config. Generation: 1
Parent Server MPM Generation: 0
Server uptime: 2 minutes 17 seconds
Server load: 0.13 0.17 0.17
Total accesses: 2 - Total Traffic: 0 kB - Total Duration: 0
CPU Usage: u.01 s0 cu0 cs0 - .0073% CPU load
.0146 requests/sec - 0 B/second - 0 B/request - 0 ms/request
1 requests currently being processed, 74 idle workers

Slot	PID	Stopping	Connections		Threads		Async connections		
			total	accepting	busy	idle	writing	keep-alive	closing
0	1130253	no	0	yes	1	24	0	0	0
1	1130254	no	0	yes	0	25	0	0	0
2	1130255	no	0	yes	0	25	0	0	0
Sum	3	0	0		1	74	0	0	0



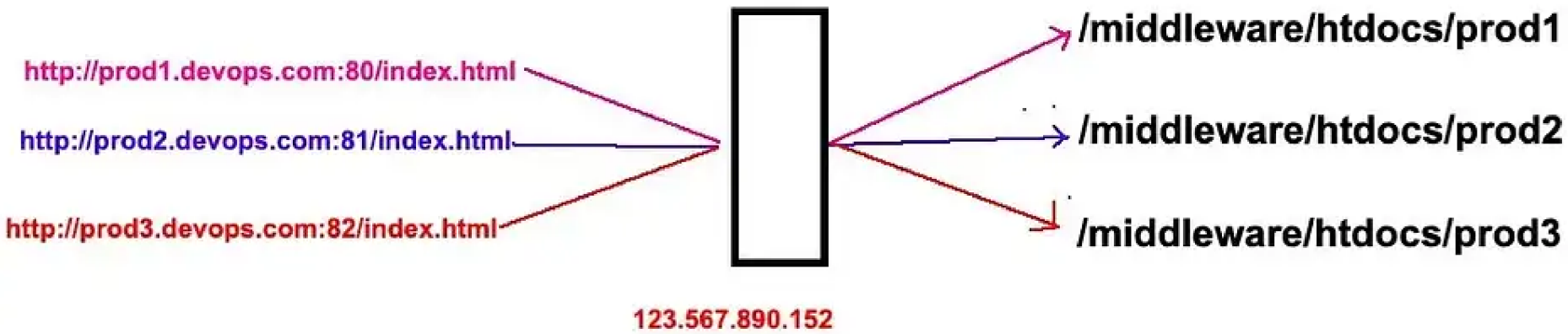
Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process

Srv	PID	Acc	M	CPU	SS	Req	Dur	Conn	Child	Slot	Client	Protocol	VHost	Request
0-0	1130253	0/1/1	_	0.00	124	0	0	0.0	0.00	0.00	76.184.214.252	http/1.1	prod.devops.com:443	GET /favicon.ico HTTP/1.1
0-0	1130253	0/1/1	_	0.00	9	0	0	0.0	0.00	0.00	76.184.214.252	http/1.1	prod.devops.com:443	GET /mySecurePage/ HTTP/1.1
0-0	1130253	1/0/0	W	0.00	0	0	0	0.0	0.00	0.00	76.184.214.252	http/1.1	prod.devops.com:443	GET /server-status HTTP/1.1

How to make your apache configuration super simple and short, using macro.

Apache Macro is like a function in a shell script or a programming language. If you are working as devops engineer or Middleware engineer, it's recommended to make use of micro and create a blueprint (reusable configuration) of apache configuration. Once the blueprint is created, you can reuse it for multiple projects with just changing the values of variables.

Magic of Apache Macro




```
root@kube-master:/middleware/apache2.4/conf# cat apacheMacro.conf
LoadModule macro_module modules/mod_macro.so

<Macro apacheVariable $host $port $dir>
  Listen $port
  <VirtualHost *:$port>

    ServerName $host
    DocumentRoot $dir


    # limit access to intranet subdir.
    <Directory $dir/>
#       order deny,allow
#       deny from all
#       allow from 10.0.0.0/8
        Require all granted
    </Directory>
  </VirtualHost>
</Macro>

Use apacheVariable prod1.devop.com 80 /middleware/htdocs/prod1
Use apacheVariable prod2.devop.com 81 /middleware/htdocs/prod2
Use apacheVariable prod3.devop.com 82 /middleware/htdocs/prod3
```

BluePrint

Step 1: Create a configuration file by name apacheMacro.conf

Copy

```
LoadModule macro_module modules/mod_macro.so

<Macro apacheVariable $host $port $dir>
  Listen $port
  <VirtualHost *:$port>

    ServerName $host
    DocumentRoot $dir

    <Directory $dir/>
       Require all granted
    </Directory>
  </VirtualHost>
</Macro>

## Define values for the fariables
```

```
Use apacheVariable prod1.devop.com 80 /middleware/htdocs/prod1
Use apacheVariable prod2.devop.com 81 /middleware/htdocs/prod2
Use apacheVariable prod3.devop.com 82 /middleware/htdocs/prod3
```

Step 2: Include apacheMacro.conf under apache httpd.conf file and rester the server. You should be able to access your website.

Copy

```
Listen 443
ServerName prod.devops.com
<VirtualHost prod.devops.com:443>
Include conf/ssl.conf
Include conf/secure-webpage.conf
</VirtualHost>
Include conf/serverStatus.conf
Include conf/apacheMacro.conf
=
```

Related Posts:

- Tomcat Server - Configuration & Administration
- Ansible - Configuration & Administration
- How I Setup my Git Repo | Step by Step Approach

admin January 11, 2022 Uncategorized No Comments

← Tomcat – Administration

Next Post →

Subscribe

Login

Notify of

new follow-up comments

Email

›

Be the First to Comment!

0 Comments

Inline Feedbacks

View all comments

RECENT POSTS

CONTACT INFORMATION

ABOUT ME

March 30, 2022

Certified Kubernetes Administrator

February 20, 2022

Docker Certified Administrator

January 15, 2022

Email : santoshkumar.mwa@gmail.com

Experienced Middleware Engineer, skilled in Enterprise Automation with Ansible, Linux shell,Python scripting, management of Enterprise web and app servers including WebSphere Application Server, Jboss, Tomcat, Apache, IHS Web server. Experience on Cloud Technologies like Amazon Webservices, Kubernetes , Docker etc. Strong information technology professional with Bachelor degree in Engineering and Master degree in Financial management

0

Would love your thoughts, please comment.x

()

x

| Reply

Insert