# Email alerts for ssl expiry

Problem :- We have to manually check the certificate expiry.
Solution :- Setting up of email alerts for the certificate expiry before due date
Task :- Email alert when is ssl certificate expired on server

Prerequisite :

1. Os (Rhel, Ubuntu , etc)
2. Root privileges

To be installed :

1. Apache (For testing ssl certificates )
2. Openssl (For creating ssl certificates )
3. Smtp postfix (For sharing the email alerts )
4. Send mail (For sharing mails )
5. Script (For creating email alerts for ssl expiry )
6. Script (To update the new certificate as soon we get and deploy to location)
7. Crontab (For scheduling the tasks to run script )

1. **Setting up of apache server to check the ssl**

kd@127:~$ sudo apt install apache2

```
kd@127:~$ sudo apt install apache2
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.41-4ubuntu3.15).
The following package was automatically installed and is no longer required:
  gir1.2-goa-1.0
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 17 not upgraded.
```

kd@127:/var/log$ sudo systemctl status apache2

```
kd@127:/var/log$ sudo systemctl status apache2
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2024-01-28 00:12:30 IST; 6s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 71748 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 71752 (apache2)
      Tasks: 55 (limit: 18943)
     Memory: 5.1M
     CGroup: /system.slice/apache2.service
             ├─71752 /usr/sbin/apache2 -k start
             ├─71753 /usr/sbin/apache2 -k start
             └─71754 /usr/sbin/apache2 -k start

Jan 28 00:12:29 127.0.0.1abc.local.com systemd[1]: Starting The Apache HTTP Server...
Jan 28 00:12:30 127.0.0.1abc.local.com apachectl[71751]: AH00557: apache2: apr_sockaddr_info_get() failed for 127.0.0.1abc.local.com
Jan 28 00:12:30 127.0.0.1abc.local.com apachectl[71751]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, usi
Jan 28 00:12:30 127.0.0.1abc.local.com systemd[1]: Started The Apache HTTP Server
```

2. **Setting up of openssl and creating of self-signed certificate**

sudo apt-get update
sudo apt-get install openssl

**Generate a Private Key:** Use OpenSSL to generate a private key:
kd@127:/etc/apache2/sites-available$ sudo openssl genpkey -algorithm RSA -out private.key

```
kd@127:/etc/apache2/sites-available$ sudo openssl genpkey -algorithm RSA -out private.key
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
.....................+++++
.......................................................................+++++
```

**Create a Certificate Signing Request (CSR):** Generate a CSR, which is a file that contains information about the organization and the public key:

kd@127:/etc/apache2/sites-available$ sudo openssl req -new -key private.key -out server.csr

```
kd@127:/etc/apache2/sites-available$ sudo openssl req -new -key private.key -out server.csr
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
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]:IN
State or Province Name (full name) [Some-State]:DELHI
Locality Name (eg, city) []:DELHI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:KEENABLE
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:ABC

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:KD
```

**Generate a Self-Signed Certificate:** Use the private key to generate a self-signed certificate:
kd@127:/etc/apache2/sites-available$ sudo openssl req -x509 -key private.key -in server.csr -out server.crt

```
kd@127:/etc/apache2/sites-available$ sudo openssl req -x509 -key private.key -in server.csr -out server.crt
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
```

**Verify the Generated Certificate:**
kd@127:/etc/apache2/sites-available$ sudo openssl x509 -text -noout -in server.crt

```
kd@127:/etc/apache2/sites-available$ sudo openssl x509 -text -noout -in server.crt
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0b:4f:95:f3:bc:e6:cc:e9:c8:08:ce:89:79:75:28:e5:ad:78:04:60
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = IN, ST = DELHI, L = DELHI, O = KEENABLE, OU = " ", CN = " ", emailAddress = ABC
        Validity
            Not Before: Jan 27 19:10:03 2024 GMT
            Not After : Feb 26 19:10:03 2024 GMT
        Subject: C = IN, ST = DELHI, L = DELHI, O = KEENABLE, OU = " ", CN = " ", emailAddress = ABC
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:ef:18:78:ac:ac:ee:ab:5c:32:e4:0c:8d:3f:f6:
                    27:8d:10:f4:49:8e:5a:c8:68:a4:56:7b:4e:67:eb:
                    57:a1:18:d4:40:c3:a7:86:2d:d2:5e:fc:4c:f1:12:
                    ff:bc:50:5a:38:cb:49:11:20:55:61:32:9d:f6:dc:
                    bf:59:1e:6e:03:76:30:7c:32:19:ef:6b:d5:f9:3f:
                    db:74:36:c5:e4:c5:81:3a:ab:31:74:77:e9:6c:27:
                    b2:4f:98:ff:35:7a:ae:44:40:ff:ce:f5:5a:74:35:
                    c9:5c:9e:28:c1:2c:18:78:d5:8f:b1:7f:bc:4c:38:
                    d0:92:96:b8:a7:e6:ca:c6:96:3d:fe:ce:38:a8:75:
                    71:69:7e:47:a6:13:22:f3:96:3b:7e:d2:b8:69:5c:
                    60:2d:a9:c8:e7:f7:35:5e:8c:3d:f5:1f:83:8a:8d:
                    8d:71:2b:5f:6f:e5:1c:bf:12:c2:d1:e1:6e:88:a8:
                    17:83:36:a2:89:c4:45:33:87:af:6a:81:32:5b:7e:
                    8f:39:f9:a8:14:75:13:de:3d:2b:31:bb:e3:41:55:
                    8c:cd:fc:1b:b9:b0:c9:a9:9b:80:3a:d1:0f:59:da:
```

**Update the VirtualHost configuration to include the SSL settings:**

Change the configuration as required

kd@127:/etc/apache2/sites-available$ cat 000-default.conf

```
<VirtualHost *:80>
      ServerAdmin webmaster@localhost
      ServerName localhost

      DocumentRoot /var/www/html

      ErrorLog ${APACHE_LOG_DIR}/error.log
      CustomLog ${APACHE_LOG_DIR}/access.log combined

      Redirect "/" "https://localhost/"
</VirtualHost>

<VirtualHost *:443>
      ServerAdmin webmaster@localhost
      ServerName your_domain_or_ip

      DocumentRoot /var/www/html

      ErrorLog ${APACHE_LOG_DIR}/error.log
```

```
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        SSLEngine on
        SSLCertificateFile /etc/apache2/sites-available/server.crt
        SSLCertificateKeyFile /etc/apache2/sites-available/private.key
        # Optional: SSLCertificateChainFile /path/to/your/chainfile.pem

        <FilesMatch "\.(cgi|shtml|phtml|php)$">
        SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
        SSLOptions +StdEnvVars
        </Directory>

        BrowserMatch "MSIE [2-6]" \
        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown

</VirtualHost>
```

**Enable the SSL module and the new site configuration:**

kd@127:/etc/apache2/sites-available$ sudo a2enmod ssl

```
kd@127:/etc/apache2/sites-available$ sudo a2enmod ssl
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
```

kd@127:/etc/apache2/sites-available$ sudo a2ensite 000-default.conf

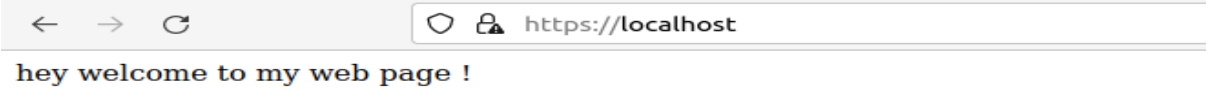
**Restart Apache to apply the changes:**

kd@127:/etc/apache2/sites-available$ sudo systemctl restart apache2

```
kd@127:/etc/apache2/sites-available$ sudo a2ensite 000-default.conf
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
Site 000-default already enabled
kd@127:/etc/apache2/sites-available$ sudo systemctl restart apache2
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
```

Now navigate to https://localhost

You will see that our page is now secured N




Now lets see our self signed certificate -

| | |
|---|---|
| **Subject Name** | |
| Country | IN |
| State/Province | DELHI |
| Locality | DELHI |
| Organization | KEENABLE |
| Organizational Unit | |
| Common Name | |
| Email Address | ABC |
| **Issuer Name** | |
| Country | IN |
| State/Province | DELHI |
| Locality | DELHI |
| Organization | KEENABLE |
| Organizational Unit | |
| Common Name | |
| Email Address | ABC |
| **Validity** | |
| Not Before | Sat, 27 Jan 2024 19:10:03 GMT |
| Not After | Mon, 26 Feb 2024 19:10:03 GMT |
| **Public Key Info** | |
| Algorithm | RSA |
| Key Size | 2048 |
| Exponent | 65537 |
| Modulus | EF:18:78:AC:AC:EE:AB:5C:32:E4:0C:8D:3F:F6:27:8D:10:F4:49:8E:5A:C8:68:A4:... |

Now our certificate is set up now we need to create a script for creating alerts for ssl

For that we need postfix (smtp) for sharing mail alerts and crontab and mutt or mailx for sharing mail .

## 3. Setting up of postfix :-

For that first we had to set up hostname using the following command

kd@127:~$ sudo hostnamectl set-hostname abc.local.com

```
kd@127:~$ sudo hostnamectl set-hostname abc.local.com
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
kd@127:~$ hostnamectl
   Static hostname: abc.local.com
         Icon name: computer-laptop
           Chassis: laptop
        Machine ID: 57eff2f89c044e6691d515e354024e72
           Boot ID: 4730e309fa3d42399e4b6e8a3ef5679c
  Operating System: Ubuntu 20.04.6 LTS
            Kernel: Linux 5.15.0-91-generic
      Architecture: x86-64
```

**Now install postfix using the below command**

kd@127:~$ sudo apt install postfix

```
kd@127:~$ sudo apt install postfix
sudo: unable to resolve host 127.0.0.1abc.local.com: Name or service not known
[sudo] password for kd:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  gir1.2-goa-1.0
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  procmail postfix-mysql postfix-pgsql postfix-ldap postfix-pcre postfix-lmdb postfix-sqlite sasl2-bin | dovecot-common resolvconf postfix-cdb
  postfix-doc
The following NEW packages will be installed:
  postfix
0 upgraded, 1 newly installed, 0 to remove and 17 not upgraded.
Need to get 1,200 kB of archives.
After this operation, 4,578 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 postfix amd64 3.4.13-0ubuntu1.3 [1,200 kB]
Fetched 1,200 kB in 4s (295 kB/s)
Preconfiguring packages ...
Selecting previously unselected package postfix.
(Reading database ... 280292 files and directories currently installed.)
Preparing to unpack .../postfix_3.4.13-0ubuntu1.3_amd64.deb ...
Unpacking postfix (3.4.13-0ubuntu1.3) ...
Setting up postfix (3.4.13-0ubuntu1.3) ...
Adding group `postfix' (GID 142) ...
Done.
Adding system user `postfix' (UID 131) ...
Adding new user `postfix' (UID 131) with group `postfix' ...
Not creating home directory `/var/spool/postfix'.
Creating /etc/postfix/dynamicmaps.cf
Adding group `postdrop' (GID 143) ...
```

Go to main.cf file in /etc/postfix/main.cf

**Add the below configuration**

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete
version


# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
```

```
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html -- default to 2
on
# fresh installs.
compatibility_level = 2



# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_security_level=may

smtp_tls_CApath=/etc/ssl/certs
smtp_tls_security_level=may
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache


smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated
defer_unauth_destination
myhostname = abc.local.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = $myhostname, abc.local.com, localhost,
localhost.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
```

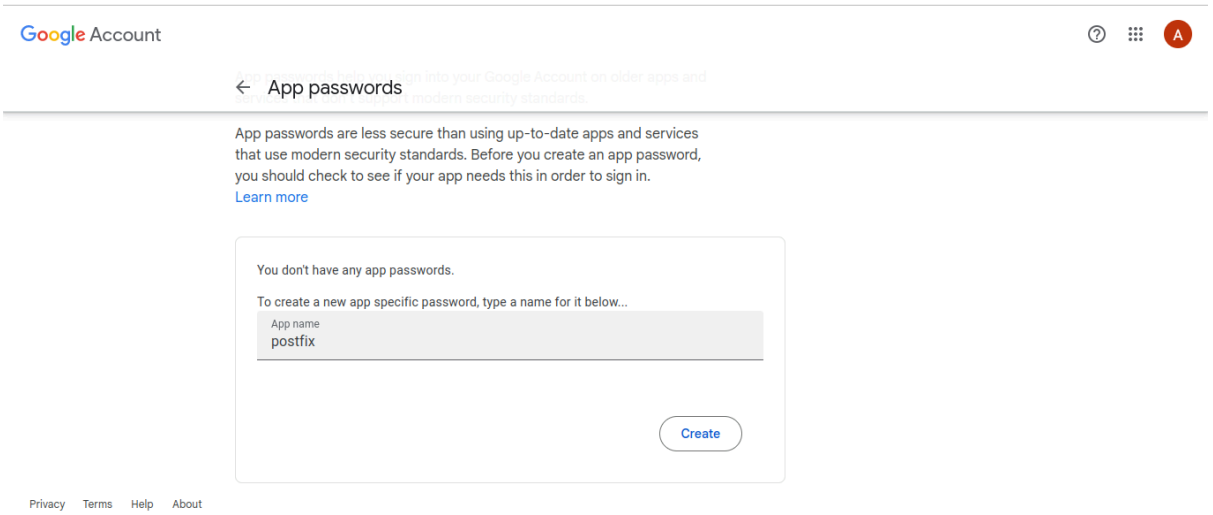
# Generate Google App Password for Postfix

We need to generate an App password

Log in to your email, then click the following link: [Manage your account access and security settings.]

Scroll down to “Signing into Google” and click 2-Step Verification. You may be asked for your password and a verification code before continuing. Ensure that 2-Step Verification is enabled.

Click the following link to [Generate an App password] for Postfix:




# Add Gmail Username and App Password to Postfix configuration

You need to add your username and password in this file sasl_passwd to this directory /etc/postfix/sasl/

Create /etc/postfix/sasl/sasl_passwd file and add your gmail ID and password we have just created using below command

root@ip-172-31-0-205:/etc/postfix/sasl# sudo nano sasl_passwd

And add your gmail ID and password as shown

```
kd@127:/etc/postfix/sasl$ cat sasl_passwd
[smtp.gmail.com]:587 alertbot01@gmail.com:wqvygnuphajwssrt
kd@127:/etc/postfix/sasl$
```

**create the hash file for Postfix using the postmap command**

postmap /etc/postfix/sasl/sasl_passwd

```
kd@127:/etc/postfix/sasl$ sudo postmap /etc/postfix/sasl/sasl_passwd
kd@127:/etc/postfix/sasl$
```

After execute postmap command you should have a new file named sasl_passwd.db in the /etc/postfix/.

```
kd@127:/etc/postfix/sasl$ ls -ltr
total 12
-rw-r--r-- 1 root root    59 Jan 28 01:40 sasl_passwd
-rw-r--r-- 1 root root 12288 Jan 28 01:42 sasl_passwd.db
kd@127:/etc/postfix/sasl$
```

# Secure Your Postfix Hash Database and Email Password Files

chown root:root /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db

chmod 0600 /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db

```
kd@127:/etc/postfix/sasl$ sudo chown root:root /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db
kd@127:/etc/postfix/sasl$ sudo chmod 0600 /etc/postfix/sasl/sasl_passwd /etc/postfix/sasl/sasl_passwd.db
kd@127:/etc/postfix/sasl$
```

# Configure Relay Host postfix with gmail

Modify the main.cf file using below command:

sudo vim /etc/postfix/main.cf

Set the relayhost

relayhost = [smtp.gmail.com]:587

If you want to check your relayhost set or not then run the below command and you will get output like this:

```
kd@127:/etc/postfix/sasl$ cat /etc/postfix/main.cf | grep -i ^relayhost
relayhost = [smtp.gmail.com]:587
kd@127:/etc/postfix/sasl$
```

## Add Custom Configuration

Open main.cf file and this below line end on the file

sudo vim /etc/postfix/main.cf

```
# Enable SASL authentication
smtp_sasl_auth_enable = yes
# Disallow methods that allow anonymous authentication
smtp_sasl_security_options = noanonymous
# Location of sasl_passwd
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
# Enable STARTTLS encryption
smtp_tls_security_level = encrypt
# Location of CA certificates
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

```
# Enable SASL authentication
smtp_sasl_auth_enable = yes
# Disallow methods that allow anonymous authentication
smtp_sasl_security_options = noanonymous
# Location of sasl_passwd
smtp_sasl_password_maps = hash:/etc/postfix/sasl/sasl_passwd
# Enable STARTTLS encryption
smtp_tls_security_level = encrypt
# Location of CA certificates
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

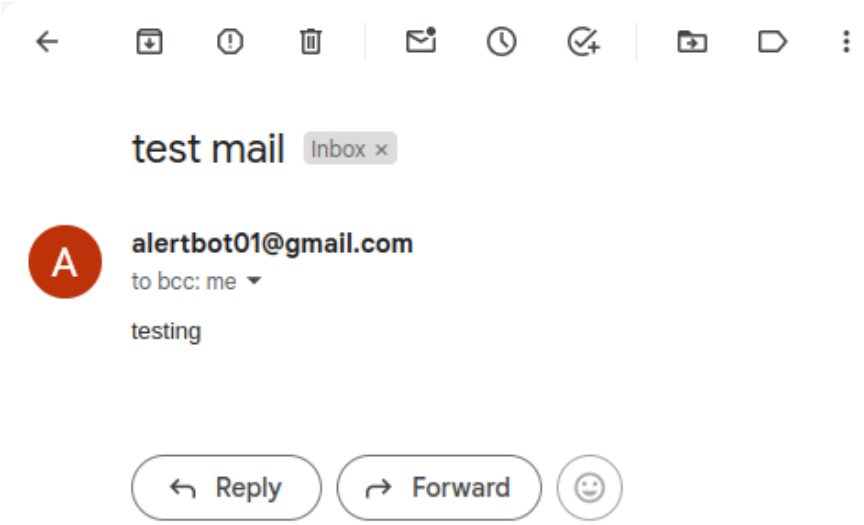
# 4. Send Email using sendmail

Lets test whether our SMTP server

```
kd@127:~$ sendmail alertbot01@gmail.com
From: root@gmail.com
Subject: test mail
testing
.
```

```
kd@127:~$ sendmail alertbot01@gmail.com
From: root@gmail.com
Subject: test mail
testing
.
kd@127:~$
```

Received our first mail successfully




Now lets create a script which will check **For ssl certificate expiring --> email msg should be sent 7 days before onwards, at the same time once the New Certificate Received, should be deployed Automatically and confirmation Message must be sent to all Members.**

## 5. Steps to create script :

1. Create a script using vim command

```
kd@127:~$ sudo vim ssl.sh
```

2. Paste the content inside the file

In this file we have defined the alert threshold as 200 for getting alert we can change it as our requirement for that i had set to 7 days .

```
#!/bin/bash

SSL_HOSTNAME="www.localhost"
SSL_PORT=443
ALERT_THRESHOLD=200
RECIPIENT_EMAIL="alertbot01@gmail.com"

cert_info=$(openssl s_client -showcerts -connect
"${SSL_HOSTNAME}:${SSL_PORT}" </dev/null 2>/dev/null)

if [[ $? -eq 0 ]]; then
    expiration_date=$(echo "${cert_info}" | openssl x509 -noout -enddate
| cut -d= -f2)
    if [ -n "${expiration_date}" ]; then
        expiration_epoch=$(date -d "${expiration_date}" +%s)
        current_epoch=$(date +%s)
        days_until_expiry=$(( (${expiration_epoch} - ${current_epoch}) /
86400 ))

        if [ ${days_until_expiry} -gt 0 ]; then
            echo "The SSL certificate is valid for ${days_until_expiry}
days."

            if [ ${days_until_expiry} -le ${ALERT_THRESHOLD} ]; then
                echo "Sending email alert..."
                {
                    echo "To: ${RECIPIENT_EMAIL}"
                    echo "Subject: SSL Certificate Expiry Alert"
                    echo
                    echo "The SSL certificate for ${SSL_HOSTNAME} is
about to expire in ${days_until_expiry} days."
                    echo "Please take appropriate action to renew the
certificate."
                } | sendmail -t
                echo "Email alert sent."
            fi
```

```
        else
            echo "The SSL certificate has expired or is about to
expire."
        fi
    else
        echo "Error extracting expiration date from SSL certificate."
    fi
else
    echo "Error connecting to the SSL endpoint."
fi
```

3. Change the permission of the script using chmod command
   chmod + x ssl.sh

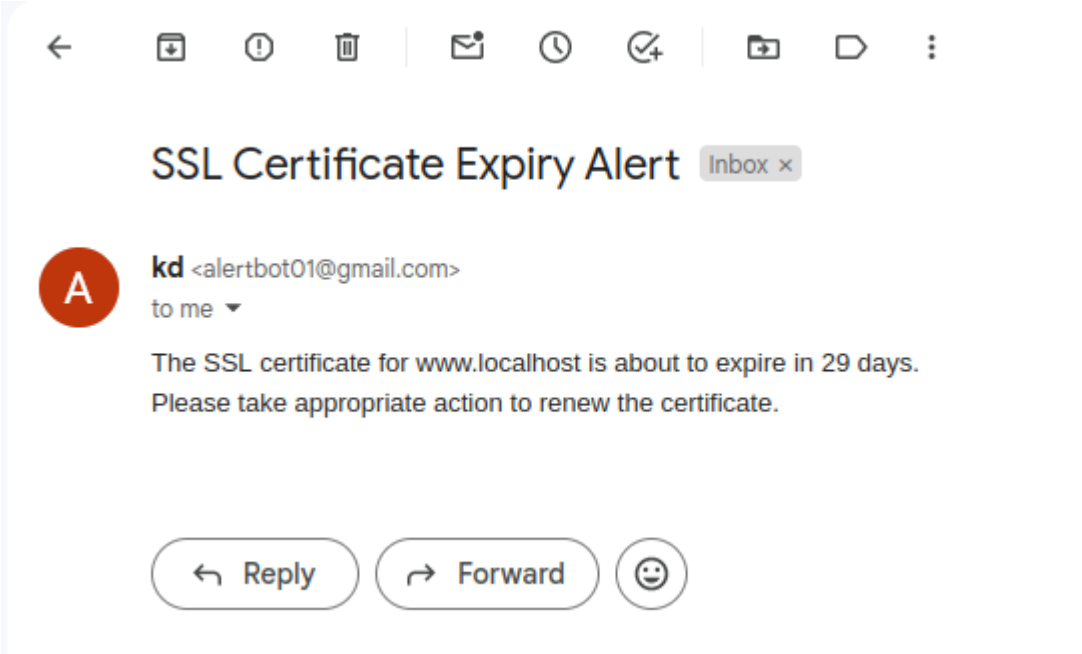
```
kd@127:~$ sudo chmod +x ssl.sh
```

Testing :

1. Test the script using sh or ./ command

```
kd@127:~$ ./ssl.sh
The SSL certificate is valid for 29 days.
Sending email alert...
Email alert sent.
kd@127:~$
```

2. Checking the mail box for the related mail .




**Now lets create another script for automatic deployment of ssl certificate whenever we get.**

**Scenario :- For this we assume that we will get certificate in /tmp directory and when our certificate will expire it will automatically deploy to the location like in apache we have**

I have created ssl certificate in tmp for the testing purpose

kd@127:/tmp$ openssl genpkey -algorithm RSA -out private.key

```
kd@127:/tmp$ openssl genpkey -algorithm RSA -out private.key
.....................................................................+++++
.............................................+++++
```

kd@127:/tmp$ openssl req -new -key private.key -out server.csr

```
kd@127:/tmp$ openssl req -new -key private.key -out server.csr
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]:IN
string is too long, it needs to be no more than 2 bytes long
Country Name (2 letter code) [AU]:INDIA
string is too long, it needs to be no more than 2 bytes long
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:DELHI
Locality Name (eg, city) []:DELHI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:KEENABLE
Organizational Unit Name (eg, section) []:KDORG
Common Name (e.g. server FQDN or YOUR name) []:KRISH
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:KDORG
```

kd@127:/tmp$ openssl x509 -req -days 90 -in server.csr -signkey private.key -out server.crt

```
kd@127:/tmp$ openssl x509 -req -days 90 -in server.csr -signkey private.key -out server.crt
Signature ok
subject=C = IN, ST = DELHI, L = DELHI, O = KEENABLE, OU = KDORG, CN = KRISH
Getting Private key
```

### 6. Now lets create a script for automatic deployment of this certificate and then sharing of notifications to the team members

Create a file in tmp using vim command

kd@127:/tmp$ cat ssl-automation.sh

```
#!/bin/bash

# SSL certificate expiration check
SSL_HOSTNAME="www.localhost"
SSL_PORT=443
ALERT_THRESHOLD=7  # Number of days before expiration to send an alert
RECIPIENT_EMAIL="alertbot01@gmail.com"

cert_info=$(openssl s_client -showcerts -connect
"${SSL_HOSTNAME}:${SSL_PORT}" </dev/null 2>/dev/null)

if [[ $? -eq 0 ]]; then
```

```
	expiration_date=$(echo "${cert_info}" | openssl x509 -noout
-enddate | cut -d= -f2)
	if [ -n "${expiration_date}" ]; then
	expiration_epoch=$(date -d "${expiration_date}" +%s)
	current_epoch=$(date +%s)
	days_until_expiry=$(( (${expiration_epoch} - ${current_epoch}) /
86400 ))

	if [ ${days_until_expiry} -le ${ALERT_THRESHOLD} ]; then
		echo "Sending SSL certificate alert..."
		{
			echo "To: ${RECIPIENT_EMAIL}"
			echo "Subject: SSL Certificate Expiry Alert"
			echo
			echo "The SSL certificate for ${SSL_HOSTNAME} is about
to expire in ${days_until_expiry} days."
			echo "Please take appropriate action to renew the
certificate."
		} | sendmail -t
		echo "SSL certificate alert email sent."

		# Check if certificate files are in /tmp
		if [ -e /tmp/private.key ] && [ -e /tmp/server.csr ] && [ -e
/tmp/server.crt ]; then
			# Set the path to the Apache sites-available directory
			APACHE_DIR="/etc/apache2/sites-available"

			# Move the files to the Apache directory
			sudo mv /tmp/private.key /tmp/server.csr
/tmp/server.crt "$APACHE_DIR/"

			# Reload Apache
			sudo systemctl reload apache2

			if [ $? -eq 0 ]; then
			echo "New SSL certificate deployed. Apache reloaded."
			echo "Sending deployment alert..."
			{
				echo "To: ${RECIPIENT_EMAIL}"
				echo "Subject: SSL Certificate Deployment Alert"
				echo
				echo "A new SSL certificate has been successfully
deployed for ${SSL_HOSTNAME}."
			} | sendmail -t
			echo "Deployment alert email sent."
			else
```

```
                  echo "Error reloading Apache. Please check the Apache
configuration."
                  fi
            else
                  echo "Certificate files not found in /tmp. Email
notification sent."
            fi
      else
            echo "The SSL certificate is valid for ${days_until_expiry}
days."
      fi
      else
      echo "Error extracting expiration date from SSL certificate."
      fi
else
      echo "Error connecting to the SSL endpoint."
fi
```

**Change the permission of the file using chmod**

kd@127:/tmp$ chmod +x ssl-automation.sh

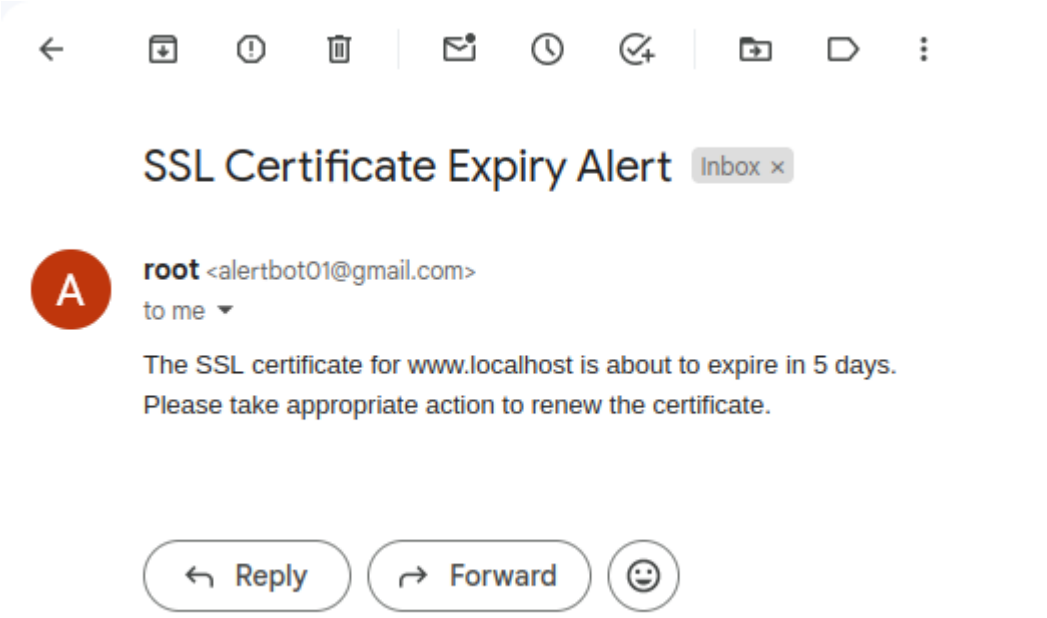
```
kd@127:/tmp$ chmod +x ssl-automation.sh
```

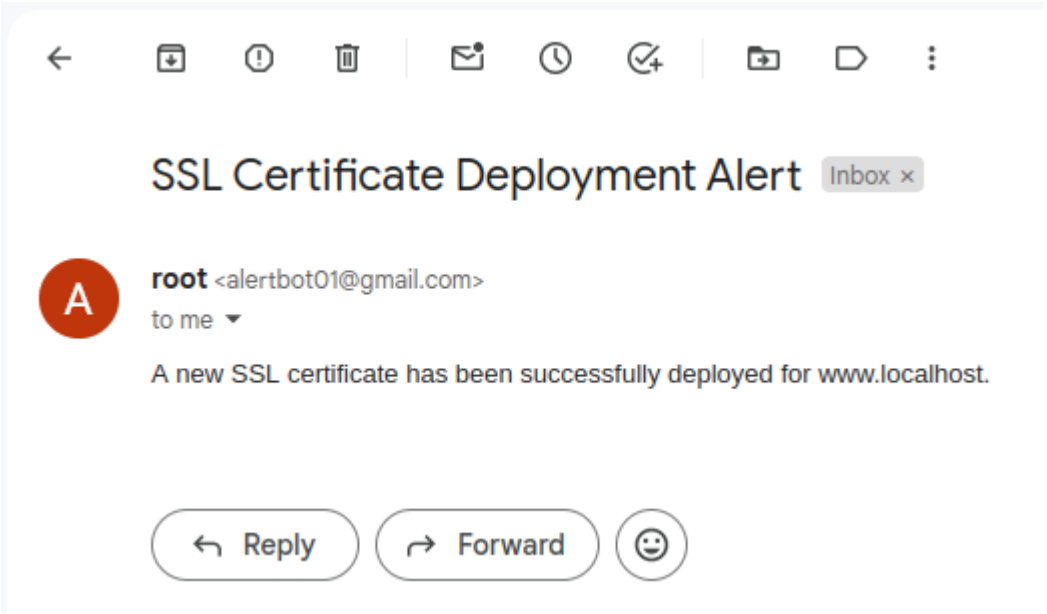
**Now run the script**

kd@127:/tmp$ sudo ./ssl-automation.sh

```
kd@127:/tmp$ sudo ./ssl-automation.sh
Sending SSL certificate alert...
SSL certificate alert email sent.
New SSL certificate deployed. Apache reloaded.
Sending deployment alert...
Deployment alert email sent.
kd@127:/tmp$
```

Mail what we got :




**7. Now setting up of cronjob to run this script everyday so that we dont miss any alert**

```
kd@127:~$ crontab -e
no crontab for kd - using an empty one
crontab: installing new crontab
```

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command


0 9 * * * /tmp/ssl-automation.sh
~
~
~
~
```

As per server requirement i have created new script for ssl alerts

```
#!/bin/bash

# Array of URLs to check
URLS=("www.kibana-openshift-logging.apps.upiprod.finopaymentbank.in"
"www.kiali-istio-system.apps.upiprod.finopaymentbank.in"
"www.10.71.87.48:8080")
EXPIRED_THRESHOLD=0  # Number of days after expiration to trigger alert
ALERT_THRESHOLD=7 # Number of days before expiration to trigger alert
RECIPIENT_EMAIL="krish_01@fosteringlinux.com"
SENDER_EMAIL="vikas_dhumale@finobank.com"

# Function to send email alert
send_email_alert() {
      local subject=$1
      local body=$2
      echo -e "${body}" | mailx -s "${subject}" -r "${SENDER_EMAIL}" -S
smtp="10.71.87.201:25" "${RECIPIENT_EMAIL}"
      echo "Email alert sent."
}

# Function to check SSL certificate for a given URL
check_ssl_certificate() {
      local SSL_HOSTNAME=$1
      local SSL_PORT=443

      cert_info=$(openssl s_client -showcerts -connect
"${SSL_HOSTNAME}:${SSL_PORT}" </dev/null 2>/dev/null)

      if [[ $? -eq 0 ]]; then
      expiration_date=$(echo "${cert_info}" | openssl x509 -noout
-enddate | cut -d= -f2)
      if [ -n "${expiration_date}" ]; then
            expiration_epoch=$(date -d "${expiration_date}" +%s)
            current_epoch=$(date +%s)
            days_until_expiry=$(( (${expiration_epoch} -
${current_epoch}) / 86400 ))

            if [ ${days_until_expiry} -gt ${EXPIRED_THRESHOLD} ]; then
                  echo "The SSL certificate for ${SSL_HOSTNAME} is valid
for ${days_until_expiry} days."

                  if [ ${days_until_expiry} -le ${ALERT_THRESHOLD} ];
```

```
then
                  echo "Sending email alert for SSL certificate
expiry..."
                  subject="SSL Certificate Expiry Alert for
${SSL_HOSTNAME}"
                  email_body="The SSL certificate for ${SSL_HOSTNAME} is
about to expire in ${days_until_expiry} days. Please take appropriate
action to renew the certificate."
                  send_email_alert "${subject}" "${email_body}"
                  fi

            else
                  echo "The SSL certificate for ${SSL_HOSTNAME} has
expired."
                  echo "Sending email alert for expired SSL
certificate..."
                  subject="Expired SSL Certificate Alert for
${SSL_HOSTNAME}"
                  email_body="The SSL certificate for ${SSL_HOSTNAME}
has expired. Please take immediate action to renew the certificate."
                  send_email_alert "${subject}" "${email_body}"
            fi
      else
            echo "Error extracting expiration date from SSL certificate
for ${SSL_HOSTNAME}."
            echo "Sending email alert for expiration date extraction
error..."
            subject="SSL Certificate Expiry Alert Error for
${SSL_HOSTNAME}"
            email_body="Error extracting expiration date from SSL
certificate for ${SSL_HOSTNAME}. Please check the SSL configuration."
            send_email_alert "${subject}" "${email_body}"
      fi
      else
      echo "Error connecting to the SSL endpoint for ${SSL_HOSTNAME}."
      echo "Sending email alert for SSL connection error..."
      subject="SSL Connection Error for ${SSL_HOSTNAME}"
      email_body="Error connecting to the SSL endpoint for
${SSL_HOSTNAME}. Please check the SSL configuration and server
availability."
      send_email_alert "${subject}" "${email_body}"
      fi
}

# Loop through each URL and check SSL certificate
for URL in "${URLS[@]}"; do
```

```
    check_ssl_certificate "$URL"
done
```

What the script do :-

1. An array named `URLS` is defined containing the URLs to be checked for SSL certificate validity.
2. Two threshold values are set:
   - `EXPIRED_THRESHOLD`: Number of days after expiration to trigger an alert.
   - `ALERT_THRESHOLD`: Number of days before expiration to trigger an alert.
3. Email addresses for sending alerts are defined:
   - `RECIPIENT_EMAIL`: Email address where alerts will be sent.
   - `SENDER_EMAIL`: Email address from which alerts will be sent.
4. The function `send_email_alert()` is defined to send email alerts. It takes two parameters: subject and body of the email.
5. The function `check_ssl_certificate()` is defined to check the SSL certificate for a given URL. It takes the hostname as a parameter.
6. Inside the `check_ssl_certificate()` function:
   - It attempts to establish a connection to the SSL endpoint of the given URL.
   - If the connection is successful:
     - It extracts the expiration date of the SSL certificate.
     - Calculates the number of days until the certificate expires.
     - Checks if the certificate is expired or about to expire based on the threshold values.
     - Sends an email alert if the certificate is about to expire or has expired.
   - If the connection fails:
     - Sends an email alert for SSL connection error.
7. The script then loops through each URL in the `URLS` array and calls the `check_ssl_certificate()` function for each URL.
8. For each URL, the script checks the SSL certificate validity and sends appropriate email alerts based on the results.
9. Finally, the script execution completes after checking all URLs in the array.