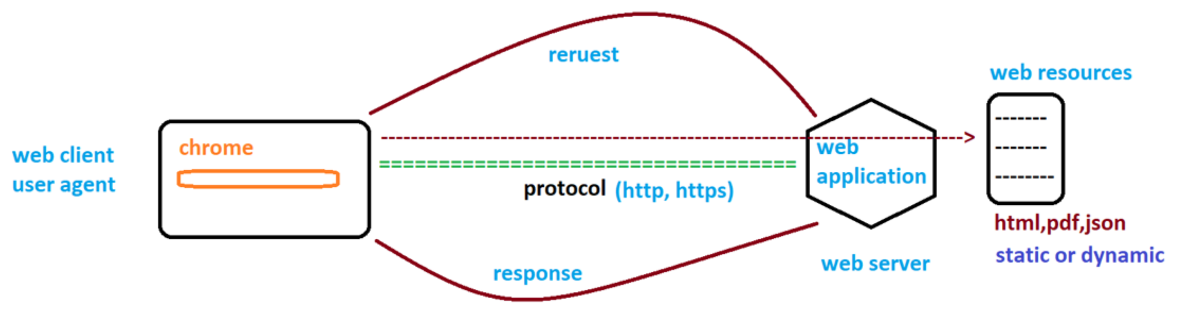
Apache Web-Server



1. What is a Web Server?

A web server is a software that processes requests via HTTP (Hypertext Transfer Protocol) and serves web pages to users. It handles client requests and delivers content like HTML, images, and other web resources.

2. What is Web Hosting?

Web hosting is a service that allows individuals and organizations to make their websites accessible via the internet. Hosting providers offer server space, security, and bandwidth to store and serve website content.

3. Types of Web Hosting

- **Shared Hosting**: Multiple websites share the same server resources.
- **VPS Hosting**: Virtual Private Server with dedicated resources for a website.
- **Dedicated Hosting**: A full server dedicated to a single website.
- **Cloud Hosting**: Resources distributed across multiple servers for scalability.
- **Managed Hosting**: Hosting with administrative support and maintenance.

4. Different Web Server Tools

- **Apache HTTP Server** (Most widely used open-source web server)
- **Nginx** (High-performance, event-driven architecture)
- **Microsoft IIS** (Windows-based web server)

- **LiteSpeed** (Optimized for speed and performance)
- **Tomcat** (For Java-based applications)

5. Difference Between Apache and Nginx

Feature	Apache	Nginx
Architecture	Process-based	Event-driven
Performance	Slower for high traffic	High performance for static files
Configuration	.htaccess support	Uses config files
Load Balancing	Limited	Built-in

6. Ports of Web Server

- **Port 80**: Default HTTP port
- **Port 443**: Default HTTPS (SSL) port

7. What is DocumentRoot?

DocumentRoot is the directory where website files are stored and served by Apache. The default location is:

```
/var/www/html
```

8. Important Files of Web Server

Configuration Files:

- `/etc/httpd/conf/httpd.conf` (Main Apache configuration file)
- `/etc/httpd/conf.d/` (Additional configuration files)
- `/etc/httpd/conf.modules.d/` (Modules configuration)

Log Files:

- `/var/log/httpd/access_log` (Records all access requests)
- `/var/log/httpd/error_log` (Records all error messages)

9. Installing Apache (httpd) on Linux (rhel9)

```
yum install httpd -y
```

10. Start and Enable Apache Service

```
systemctl start httpd
systemctl enable httpd
```

11. Allow HTTP and HTTPS Ports in Firewall

```
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https
firewall-cmd --reload
```

12. Basic Explanation of httpd Config Files

- `httpd.conf`: Core configuration file
- vhost.conf: Virtual hosts configuration
- ssl.conf: SSL settings for secure connections

13. Create a Test index.html File

```
echo "<h1>Apache Web Server is Working</h1>" > /var/www/html/index.html
```

Restart the Apache service:

```
systemctl restart httpd
```

Access the test page using the server's IP:

```
http://your-server-ip
```

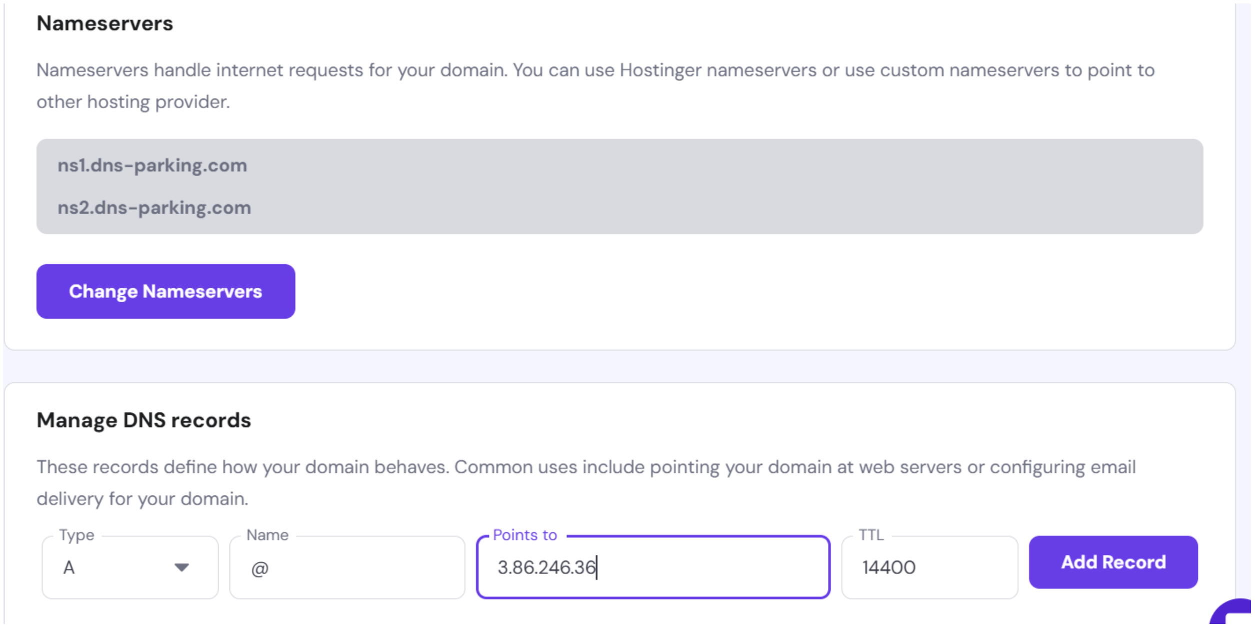
14. Map Server Public IP with Domain (Hostinger)

1. Log in to **Hostinger**.
2. Navigate to **DNS Management**.

3. Update the **A Record** with your public server IP.
4. Wait for DNS propagation (may take up to 24 hours).
5. Verify by running:

Get Public IP via Command line

```
curl -s ifconfig.me
```



Nameservers

Nameservers handle internet requests for your domain. You can use Hostinger nameservers or use custom nameservers to point to other hosting provider.

ns1.dns-parking.com
ns2.dns-parking.com

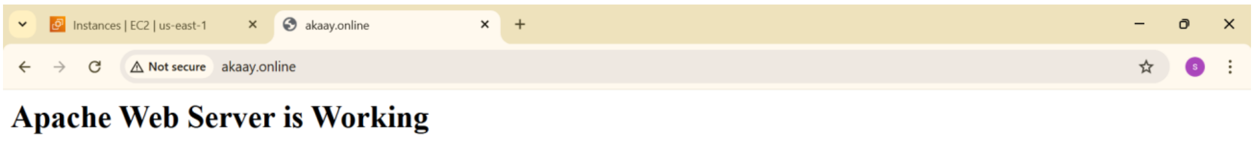
Manage DNS records

These records define how your domain behaves. Common uses include pointing your domain at web servers or configuring email delivery for your domain.

Type	Name	Points to	TTL
A	@	3.86.246.36	14400

```
dig akaay.online
```

Your Apache Web Server is now ready to serve websites!



Apache Web Server is Working

Virtual Hosting

Okay, now we will set up virtual hosting on our server.

Here, we will configure virtual hosting for the following three websites:

- www.akaay.online
- udaipur.akaay.online
- jaipur.akaay.online

To achieve this, we need to create **three custom configuration files** and set up **three different DocumentRoot directories**.

We will create the following configuration files in the `/etc/httpd/conf.d/` directory:

- `akaay.conf`
- `udaipur_akaay.conf`
- `jaipur_akaay.conf`

16. Create akaay.conf

```
vim /etc/httpd/conf.d/akaay.conf
```

```
<VirtualHost *:80>

  servername www.akaay.online

  serveradmin root@localhost

  documentroot /var/www/html

</VirtualHost>
```

17. Create udaipur_akaay.conf

```
vim /etc/httpd/conf.d/udaipur_akaay.conf
```

```
<VirtualHost *:80>

  servername udaipur.akaay.online

  serveradmin root@localhost

  documentroot /var/www/udaipur

</VirtualHost>
```

17. Create jaipur_akaay.conf

```
vim /etc/httpd/conf.d/jaipur_akaay.conf
```

```
<VirtualHost *:80>

   servername jaipur.akaay.online

   serveradmin root@localhost

   documentroot /var/www/jaipur

</VirtualHost>
```

A DocumentRoot **/var/www/html** has already been created. So now, we need to create **/var/www/udaipur** and **/var/www/jaipur.**

```
mkdir /var/www/{udaipur,jaipur}
```

18. Download free css templates

Okay, now we will prepare the web content for all three websites.

We are going to host **static websites**, so we will use **three free website templates**. You can download them from www.free-css.com.

Download css template for www.akaay.online

```
wget https://www.free-css.com/assets/files/free-css-templates/download/page296/oxer.zip
```

Next we need extract it.

we don't have **zip and unzip** command so 1st we need to install zip package.

```
yum install -y zip
```

```
unzip oxer.zip

cp -rf oxer-html/* /var/www/html/

[root@web-server ~]# ls /var/www/html/

about.html  blog.html  class.html  css  images  index.html  js

[root@web-server ~]#
```

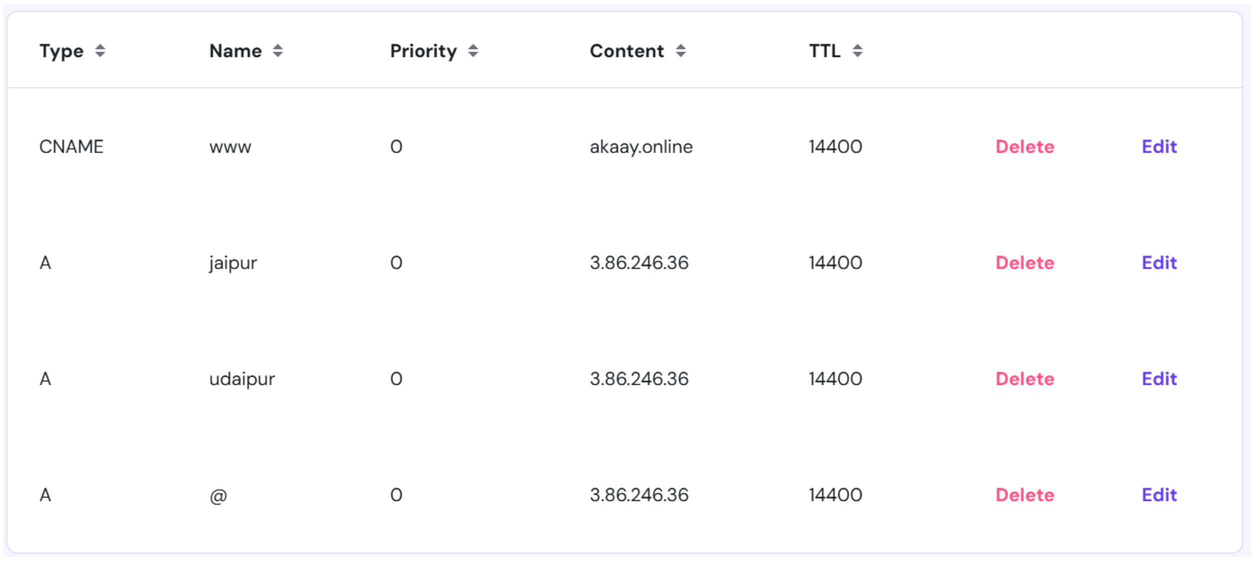
We will perform the same this steps for others domains or documentroot. Download free css template , extract it and copy all content to the documentroot (/var/www/udaipur & /var/www/jaipur)

19. Restart httpd service

```
systemctl restart httpd
```

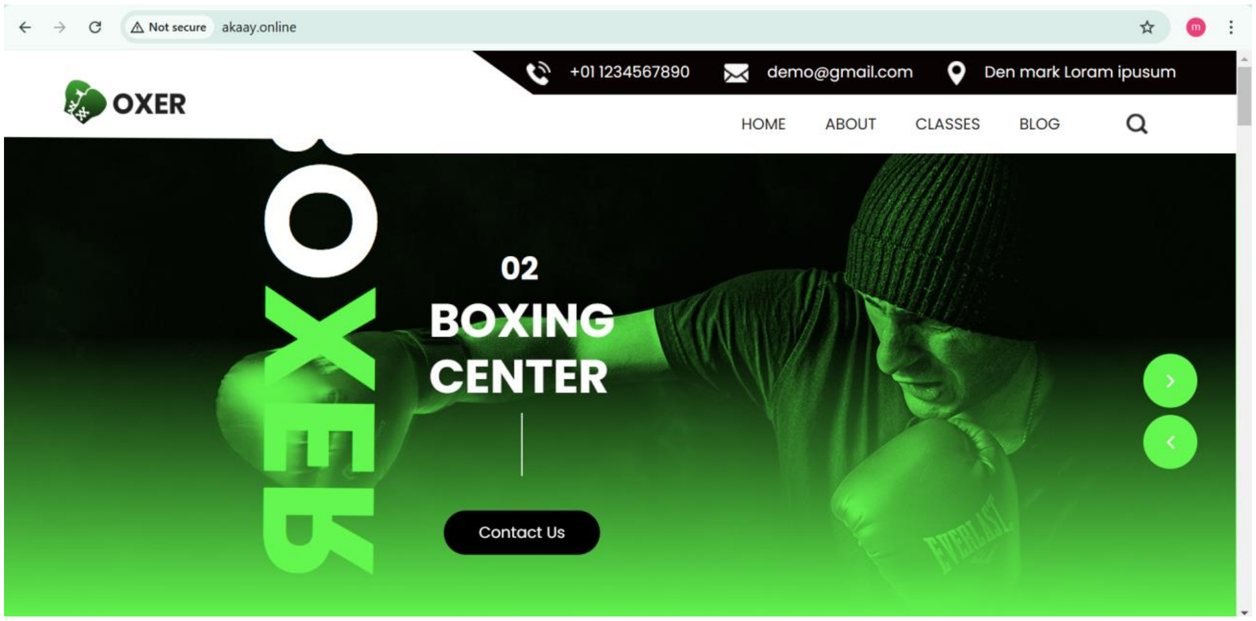
20. Create A and CNAME record on hostinger

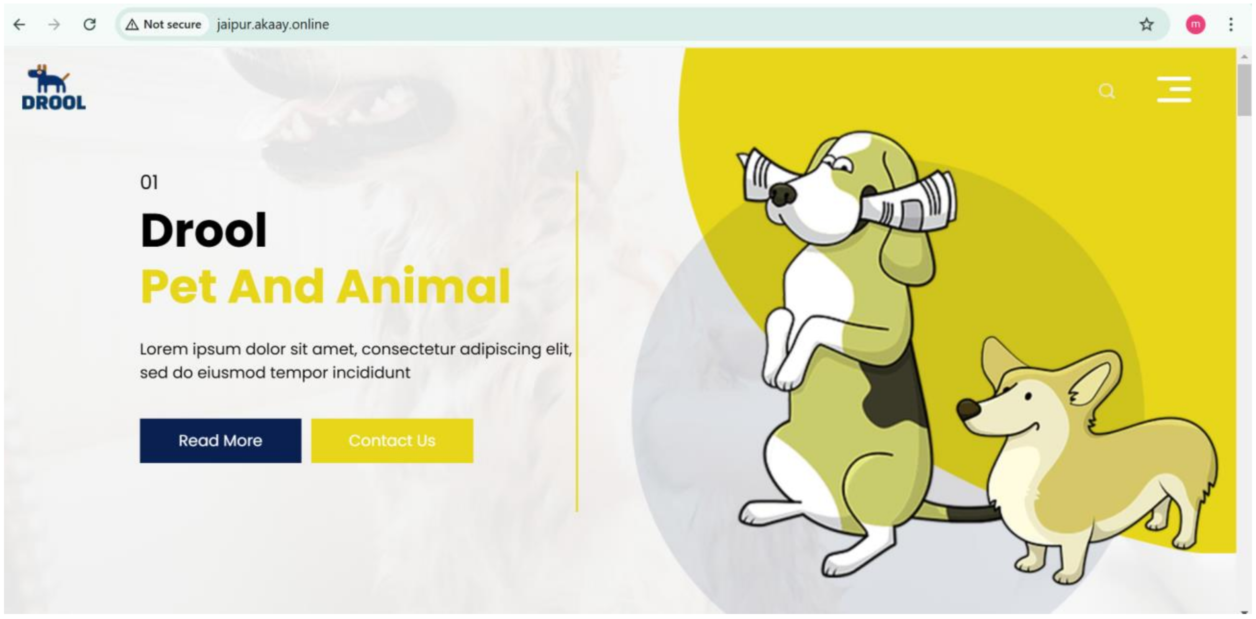
Okay, now we will create a **CNAME** record for **www.akaay.online** on Hostinger and create **A** records for **udaipur.akaay.online and jaipur.akaay.online**, mapping them to our web server's public IP.



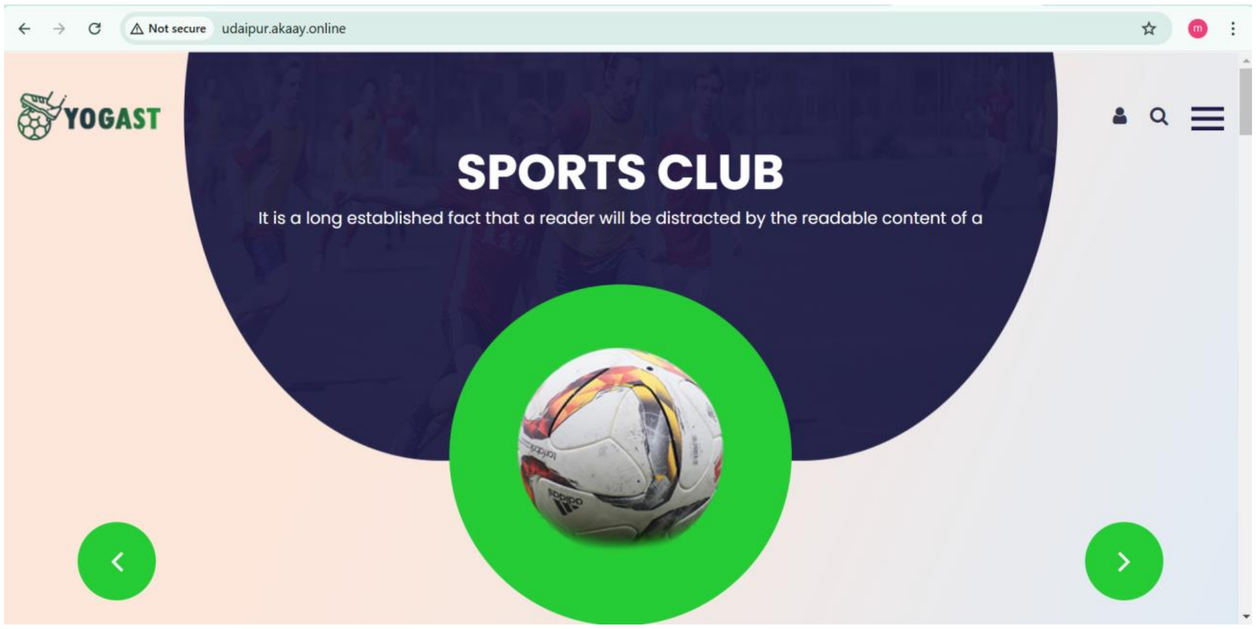
Type	Name	Priority	Content	TTL		
CNAME	www	0	akaay.online	14400	Delete	Edit
A	jaipur	0	3.86.246.36	14400	Delete	Edit
A	udaipur	0	3.86.246.36	14400	Delete	Edit
A	@	0	3.86.246.36	14400	Delete	Edit

21. Okay, now let's open the browser and check all three websites.





Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt


Not secure
udaipur.akaay.online
YOGAST
SPORTS CLUB
It is a long established fact that a reader will be distracted by the readable content of a

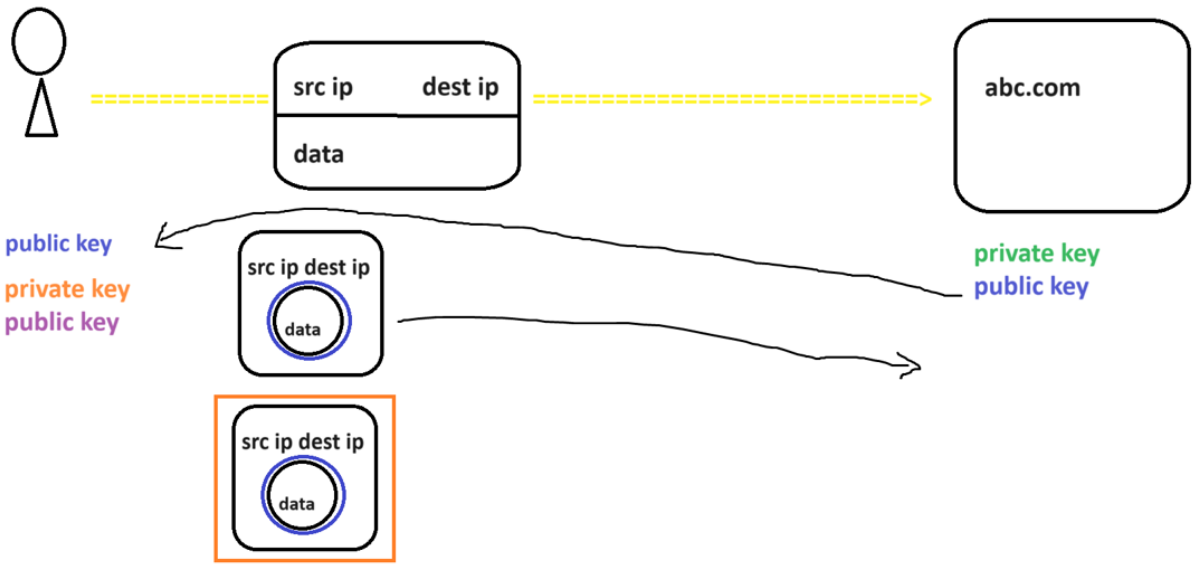
Using HTTPS For the client-server communication

1. How HTTPS Works

HTTPS (HyperText Transfer Protocol Secure) is an encrypted version of HTTP that ensures secure communication between the client and the server using **SSL/TLS (Secure Sockets Layer / Transport Layer Security).**

How it Works?

1. **Client Request** → A user visits an HTTPS website.
2. **SSL Handshake** → The server responds with its SSL certificate.
3. **Certificate Verification** → The browser checks if the certificate is valid and issued by a trusted Certificate Authority (CA).
4. **Encryption Established** → A secure connection is established using asymmetric and symmetric encryption.
5. **Secure Data Transfer** → All communication between the client and server is now encrypted.



2. Configuring HTTPS on Apache Web Server

To enable HTTPS on Apache, follow these steps:

Step 1: Install OpenSSL

Ensure that Apache and OpenSSL are installed on your system.

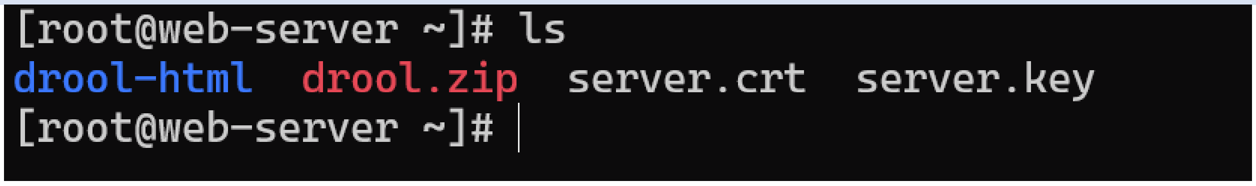
For **RHEL-based systems (CentOS, Rocky Linux, AlmaLinux, etc.):**

```
dnf install httpd mod_ssl openssl –y
```

Step 2: Create a Self-Signed Certificate

If you don't want to use Let's Encrypt, create a self-signed certificate:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
```



```
[root@web-server ~]# ls
drool-html  drool.zip  server.crt  server.key
[root@web-server ~]#
```

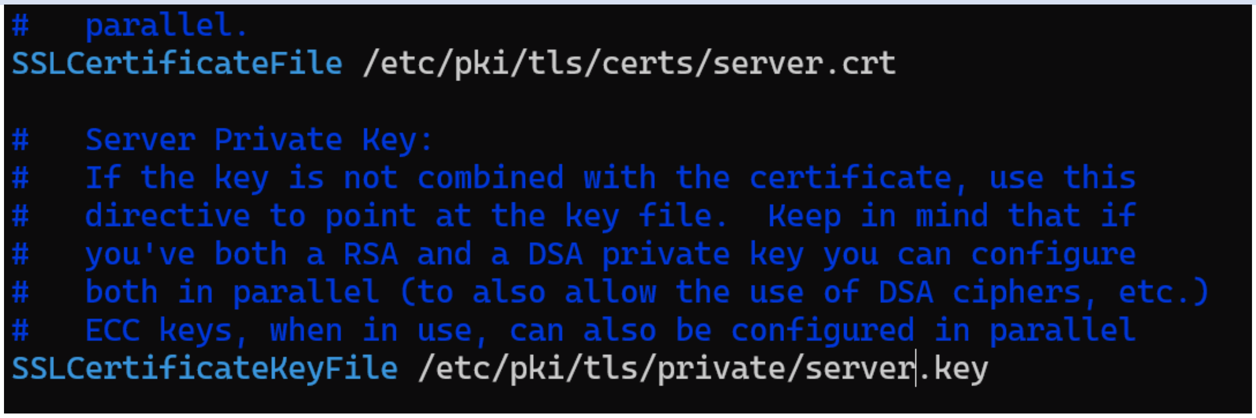
Step3: Copy crt file to /etc/pki/tls/certs/ and key file to /etc/pki/tls/private/ directory

```
cp server.crt  /etc/pki/tls/certs/
cp server.key /etc/pki/tls/private/
```

Step4: Modify ssl.conf file

Define the path of key and cert file into the ssl.conf file

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



```
#   parallel.
SSLCertificateFile /etc/pki/tls/certs/server.crt

#   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.)
#   ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile /etc/pki/tls/private/server.key
```

Step5: Modify akaay.conf (custom conf file) file

```
vim /etc/httpd/conf.d/akaay.conf
```

```
<VirtualHost *:80>

SSLEngine on

SSLCertificateFile /etc/pki/tls/certs/server.crt

SSLCertificateKeyFile /etc/pki/tls/private/server.key

  servername www.akaay.online

  serveradmin root@localhost

  documentroot /var/www/html

</VirtualHost>
```

Step6: Restart httpd service

```
systemctl restart httpd
```

Now Get SSL Certificate from Let’s Encrypt jaipur.akaay.online using certbot

Install **Certbot** to generate a free SSL certificate:

```
dnf install certbot python3-certbot-apache –y

certbot –apache

rm -f /etc/httpd/conf.d/ssl.conf

certbot --apache -d akaay.online
```

Step7: Go to browser and search https://www.akaay.online

Now let’s run host website on non standard port

Right now, we have hosted the website on port 80/TCP. Now, we will make **udaipur.akaay.online** live on port 85/TCP and see what issues we face and how we can fix them.