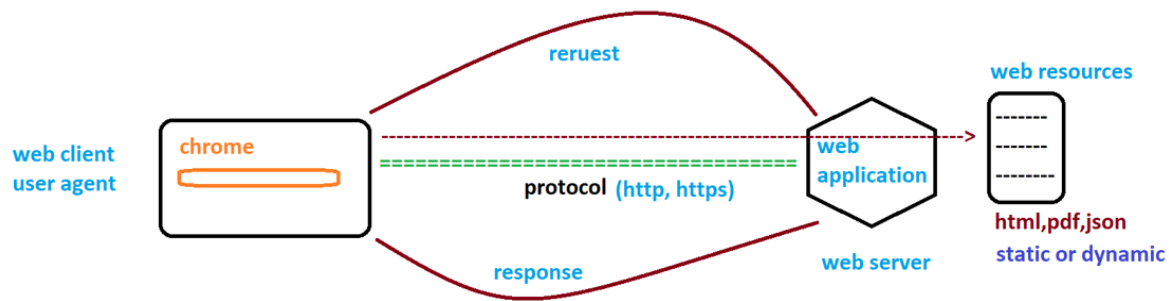


# Apache Web-Server



## 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

*web web server*

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

✓  
*h*

- **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

Change Nameservers

#### 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

A

Name

@

Points to

3.86.246.36

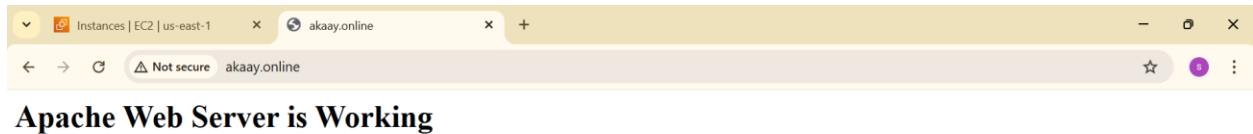
TTL

14400

Add Record

```
dig akaay.online
```

Your Apache Web Server is now ready to serve websites!



## 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](http://www.akaay.online)
- [udaipur.akaay.online](http://udaipur.akaay.online)
- [jaipur.akaay.online](http://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>
```

Handwritten annotations in blue ink: A bracket on the right side of the configuration block is labeled with the IP address '172.31.0.25'. Above the 'servername' line, there is a note '\*-80' with an arrow pointing to the port '80' in the VirtualHost tag. A checkmark is drawn next to the 'serveradmin' line.

## 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](http://www.free-css.com).

## Download css template for [www.akaay.online](http://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 1<sup>st</sup> we need to install zip package.

```
yum install -y zip

unzip oxeer.zip

cp -rf oxeer-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
```

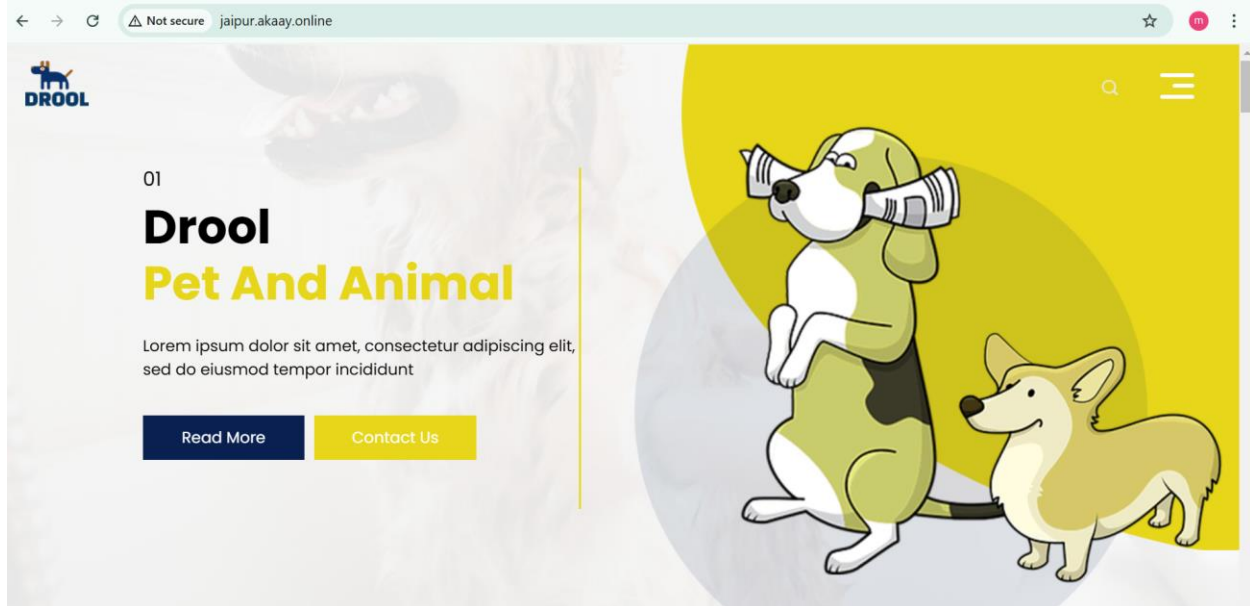
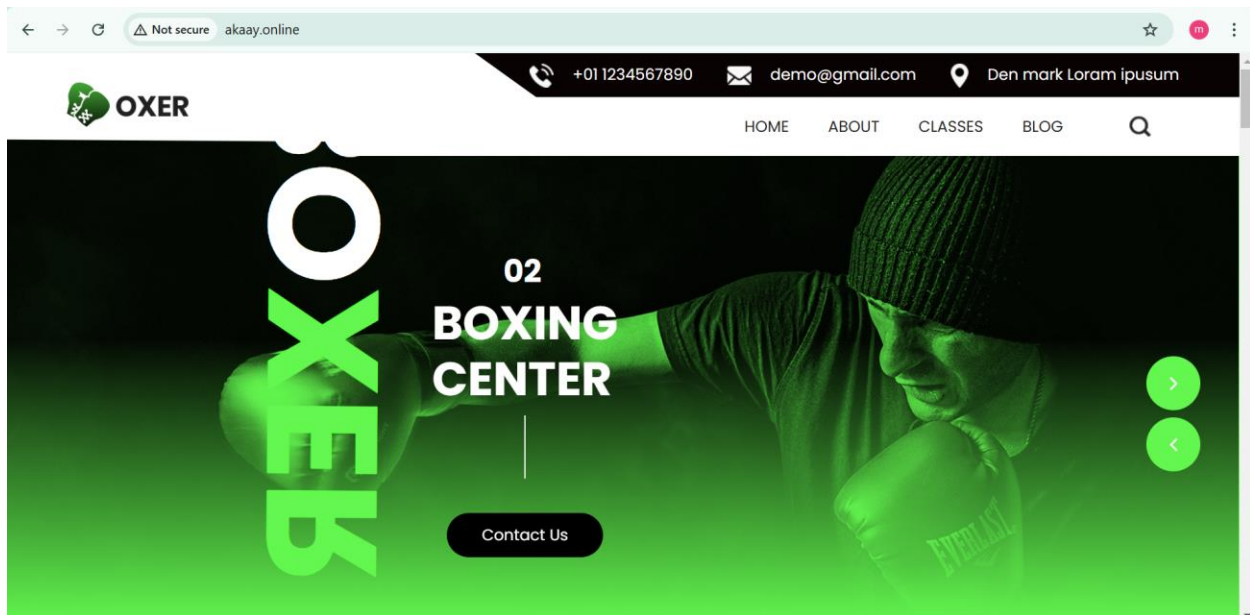
## 20. Create A and CNAME record on hostinger

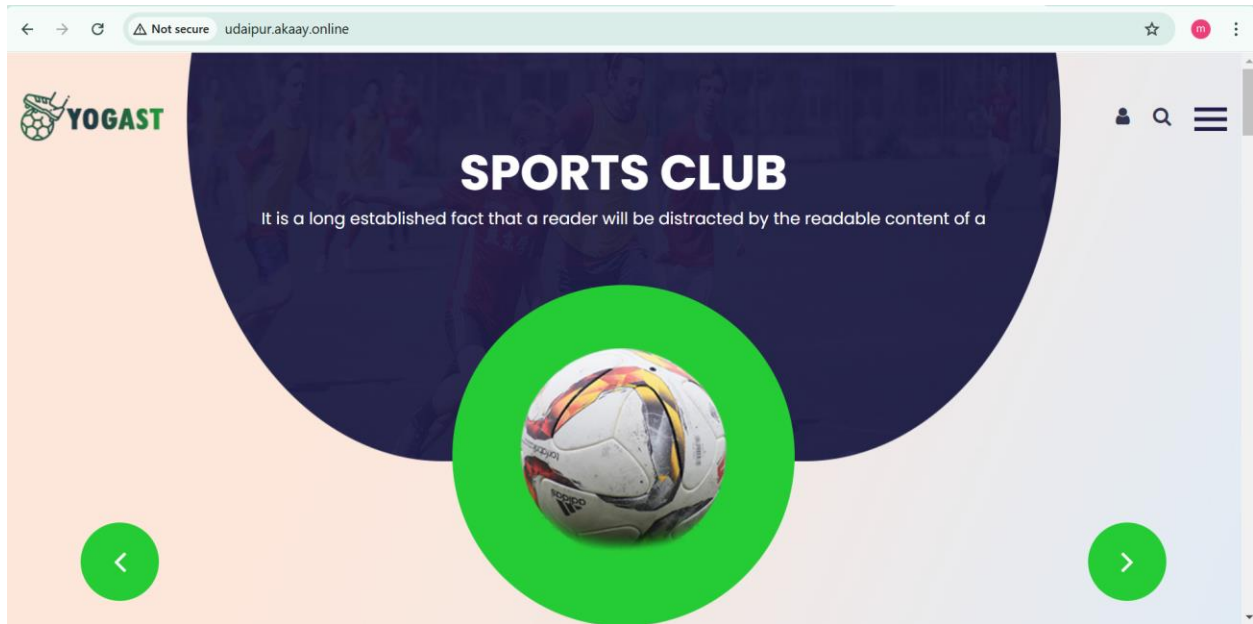
Okay, now we will create a **CNAME** record for **www.akaay.online** on Hostinger and create **A** records for **udaipur.akaay.online** and **jaipur.akaay.online**, mapping them to our web server's public IP.

Type ↕	Name ↕	Priority ↕	Content ↕	TTL ↕		
CNAME	www	0	akaay.online	14400	<a href="#">Delete</a>	<a href="#">Edit</a>
A	jaipur	0	3.86.246.36	14400	<a href="#">Delete</a>	<a href="#">Edit</a>
A	udaipur	0	3.86.246.36	14400	<a href="#">Delete</a>	<a href="#">Edit</a>
A	@	0	3.86.246.36	14400	<a href="#">Delete</a>	<a href="#">Edit</a>

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







Okay, now let's change the document root of **jaipur.akaay.online** from **/var/www/jaipur** to **/jaipur** and see what issues we face and how we can fix them.

```
mkdir /jaipur ✓
```

```
rm -rf /var/www/jaipur/ ✓
```

```
wget https://www.free-css.com/assets/files/free-css-templates/download/page291/drool.zip ✓
```

```
unzip drool.zip ✓
```

```
cp -rf drool-html/* /jaipur/
```

```
[root@web-server ~]# ls /jaipur/  
about.html  contact.html  css  images  index.html  js  
[root@web-server ~]# |
```

Let's change documentroot location into configuration file

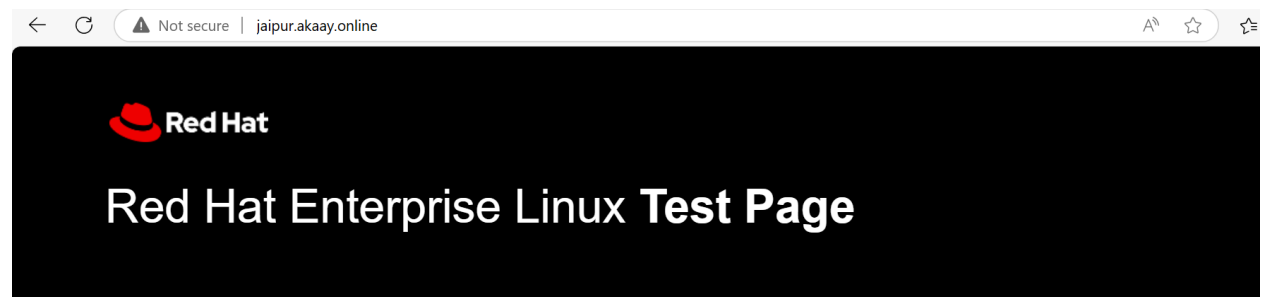
```
vim /etc/httpd/conf.d/jaipur_akaay.conf
```

```
<VirtualHost *:80>
    servername jaipur.akaay.online
    serveradmin root@localhost
    documentroot /jaipur
</VirtualHost>
~
```

Restart httpd service

```
systemctl restart httpd
```

Let's access jaipur.akaay.online



As we can see that it is showing Test Page.

So now let's check the httpd logs to find out the issues.

```
tail -f /var/log/httpd/error_log
```

```
[root@web-server ~]# tail -f /var/log/httpd/error_log
[Wed Feb 12 18:06:33.919889 2025] [authz_core:error] [pid 2076:tid 2191] [client 170.39.218.109:45412] AH01630: client denied by server configuration: /jaipur/config.env
[Wed Feb 12 18:06:34.085188 2025] [authz_core:error] [pid 2076:tid 2192] [client 170.39.218.109:45412] AH01630: client denied by server configuration: /jaipur/production
[Wed Feb 12 18:06:34.297164 2025] [authz_core:error] [pid 2076:tid 2193] [client 170.39.218.109:45412] AH01630: client denied by server configuration: /jaipur/dev
[Wed Feb 12 18:06:34.520695 2025] [authz_core:error] [pid 2076:tid 2194] [client 170.39.218.109:45412] AH01630: client denied by server configuration: /jaipur/api
```

This issue looks like that it is permission related issues.

To fix the issue, we need to allow the `/jaipur` directory in the configuration file. This is because directories created inside `/var/www/html` and `/var/www/` are allowed by default.

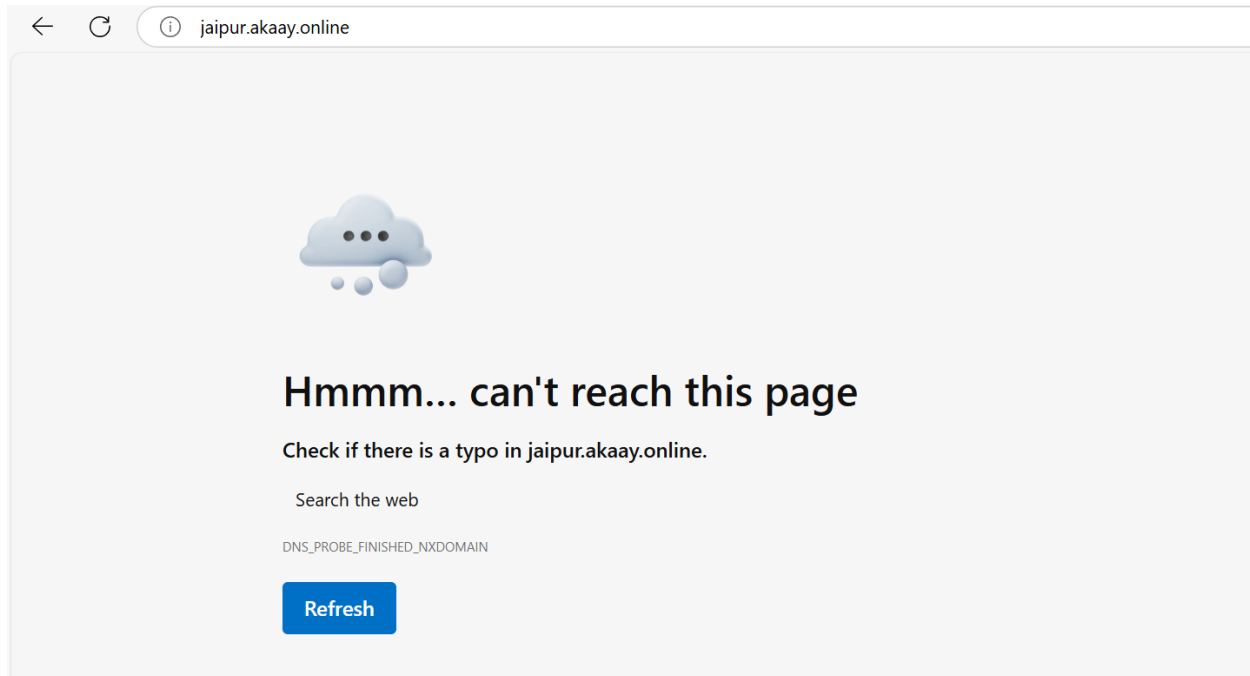
Here's how you can modify your Apache configuration to allow `/jaipur`:

```
<VirtualHost *:80>
    servername jaipur.akaay.online
    serveradmin root@localhost
    documentroot /jaipur
</VirtualHost>

<Directory "/jaipur">
    Require all granted
</Directory>

~
```

```
systemctl restart httpd
```



Still we are facing issue to access webpage but this time problem is different. Let's check logs.

```
[root@web-server ~]# tail -f /var/log/httpd/error_log
[Wed Feb 12 18:28:30.749727 2025] [core:notice] [pid 2477:tid 2477] SELinux policy enabled; httpd running as context system_u:system_r:httpd_t:s0
[Wed Feb 12 18:28:30.750641 2025] [suexec:notice] [pid 2477:tid 2477] AH01232: suEXEC mechanism enabled (wrapper: /usr/sbin/suexec)
[Wed Feb 12 18:28:30.753099 2025] [ssl:warn] [pid 2477:tid 2477] AH01906: www.akaay.online:443:0 server certificate is a CA certificate (BasicConstraints: CA == TRUE !?)
[Wed Feb 12 18:28:30.753357 2025] [ssl:warn] [pid 2477:tid 2477] AH01909: www.akaay.online:443:0 server certificate does NOT include an ID which matches the server name
[Wed Feb 12 18:28:30.767344 2025] [ssl:warn] [pid 2477:tid 2477] AH01906: www.akaay.online:443:0 server certificate is a CA certificate (BasicConstraints: CA == TRUE !?)
[Wed Feb 12 18:28:30.767441 2025] [ssl:warn] [pid 2477:tid 2477] AH01909: www.akaay.online:443:0 server certificate does NOT include an ID which matches the server name
[Wed Feb 12 18:28:30.767596 2025] [lbmethod_heartbeat:notice] [pid 2477:tid 2477] AH02282: No slotmem from mod_heartbeat
[Wed Feb 12 18:28:30.774131 2025] [mpm_event:notice] [pid 2477:tid 2477] AH00489: Apache/2.4.62 (Red Hat Enterprise Linux) OpenSSL/3.2.2 configured -- resuming normal operations
[Wed Feb 12 18:28:30.774228 2025] [core:notice] [pid 2477:tid 2477] AH00094: Command line: '/usr/sbin/httpd -D FOREGROUND'
[Wed Feb 12 18:36:42.785408 2025] [core:error] [pid 2665:tid 2699] (13)Permission denied: [client 43.157.22.57:59548] AH00035: access to /index.html denied (filesystem path '/jaipur/index.html') because search permissions are missing on a component of the path
```

Ok got this is selinux related issues. To fully track this issue, you can also check the **SELinux logs** for detailed information.

```
cat /var/log/audit/audit.log | grep AVC
```

```
[root@web-server ~]# cat /var/log/audit/audit.log | grep AVC
type=AVC msg=audit(1739385402.783:249): avc: denied { getattr } for pid=2665 comm="httpd"
path="/jaipur/index.html" dev="xvda4" ino=17707506 scontext=system_u:system_r:httpd_t:s0 tcon
text=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
type=AVC msg=audit(1739385402.783:250): avc: denied { getattr } for pid=2665 comm="httpd"
path="/jaipur/index.html" dev="xvda4" ino=17707506 scontext=system_u:system_r:httpd_t:s0 tcon
text=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
[root@web-server ~]#
```

**NOTE - AVC (Access Vector Cache)** is a logging mechanism in SELinux that tracks denied access attempts.

If SELinux is enabled, you need to set the **httpd\_sys\_content\_t** label on **/jaipur** and its content.

```
[root@web-server ~]# ls -lZ /jaipur/
total 48
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 9553 Feb 12 18:00 about.html
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 6940 Feb 12 18:00 contact.html
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0 105 Feb 12 18:00 css
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0 4096 Feb 12 18:00 images
-rw-r--r--. 1 root root unconfined_u:object_r:default_t:s0 22948 Feb 12 18:00 index.html
drwxr-xr-x. 2 root root unconfined_u:object_r:default_t:s0 53 Feb 12 18:00 js
[root@web-server ~]#
```

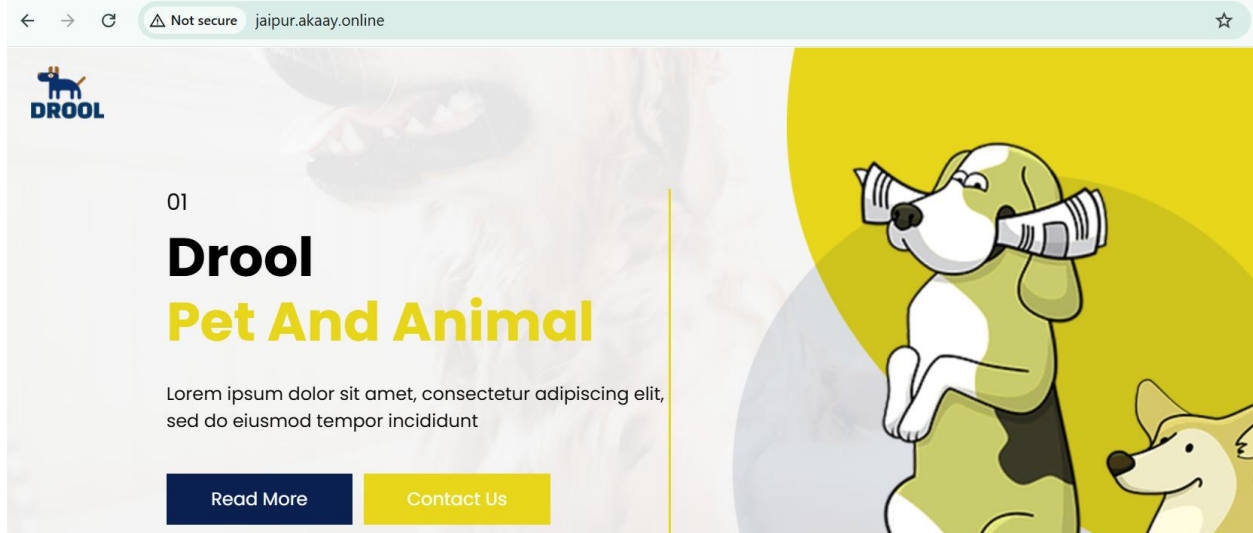
If you want to set the label permanently, use **semanage** and **restorecon**.

```
semanage fcontext -a -t httpd_sys_content_t "/jaipur(/.*)?"
```

```
restorecon -Rv /jaipur
```

```
systemctl restart httpd
```

```
[root@web-server ~]# ls -lZ /jaipur/
total 48
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 9553 Feb 12 18:00 about
.html
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 6940 Feb 12 18:00 conta
ct.html
drwxr-xr-x. 2 root root unconfined_u:object_r:httpd_sys_content_t:s0 105 Feb 12 18:00 css
drwxr-xr-x. 2 root root unconfined_u:object_r:httpd_sys_content_t:s0 4096 Feb 12 18:00 image
s
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_content_t:s0 22948 Feb 12 18:00 index
.html
drwxr-xr-x. 2 root root unconfined_u:object_r:httpd_sys_content_t:s0 53 Feb 12 18:00 js
[root@web-server ~]#
```



Okay, now let's create a directory named "data" inside the document roots of both websites:

- For [www.akaay.online](http://www.akaay.online) → /var/www/html/data
- For [jaipur.akaay.online](http://jaipur.akaay.online) → /jaipur/data







And then, we will try to access the content inside these directories.

```
[root@web-server ~]# mkdir /var/www/html/data
[root@web-server ~]#
[root@web-server ~]# mkdir /jaipur/data
[root@web-server ~]#
[root@web-server ~]# touch /var/www/html/data/file{1..5}.txt
[root@web-server ~]#
[root@web-server ~]# touch /jaipur/data/file{1..5}.txt
[root@web-server ~]#
[root@web-server ~]# |
```



← ↻ Not secure | <https://akaay.online/data/>

## Index of /data

<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
 <a href="#">Parent Directory</a>		-	
 <a href="#">file1.txt</a>	2025-02-12 18:54	0	
 <a href="#">file2.txt</a>	2025-02-12 18:54	0	
 <a href="#">file3.txt</a>	2025-02-12 18:54	0	
 <a href="#">file4.txt</a>	2025-02-12 18:54	0	
 <a href="#">file5.txt</a>	2025-02-12 18:54	0	

← → ↻ Not secure | [jaipur.akaay.online/data/](http://jaipur.akaay.online/data/)

## Forbidden

You don't have permission to access this resource.

As we can see, we are able to access the content at <https://akaay.online/data/>, but we are getting a **403 Forbidden** error at <http://jaipur.akaay.online/data/>. Let's check the logs.

```
tail -f /var/log/httpd/error_log
```

```
49:32535] AH01276: Cannot serve directory /jaipur/data/: No matching DirectoryIndex (index.html) found, and server-generated directory index forbidden by Options directive
[Wed Feb 12 19:06:38.075394 2025] [autoindex:error] [pid 3028:tid 3078] [client 223.184.132.1
49:32520] AH01276: Cannot serve directory /jaipur/data/: No matching DirectoryIndex (index.html) found, and server-generated directory index forbidden by Options directive
[Wed Feb 12 19:06:38.429821 2025] [autoindex:error] [pid 3028:tid 3079] [client 223.184.132.1
49:32520] AH01276: Cannot serve directory /jaipur/data/: No matching DirectoryIndex (index.html) found, and server-generated directory index forbidden by Options directive
```

Okay, got it. This problem is occurring because `/var/www/html` allows **index files by default**, but if we set a different **DocumentRoot**, indexing is **denied by default**. To fix this, we need to allow indexing in the configuration file.

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



```

<VirtualHost *:80>
    servername jaipur.akaay.online
    serveradmin root@localhost
    documentroot /jaipur
</VirtualHost>







<Directory "/jaipur">
    Require all granted
    Options Indexes FollowSymLinks
</Directory>
~

```

systemctl restart httpd

← → ↻ ⚠ Not secure jaipur.akaay.online/data/

## Index of /data

<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
 <a href="#">Parent Directory</a>		-	
 <a href="#">file1.txt</a>	2025-02-12 18:54	0	
 <a href="#">file2.txt</a>	2025-02-12 18:54	0	
 <a href="#">file3.txt</a>	2025-02-12 18:54	0	
 <a href="#">file4.txt</a>	2025-02-12 18:54	0	
 <a href="#">file5.txt</a>	2025-02-12 18:54	0	

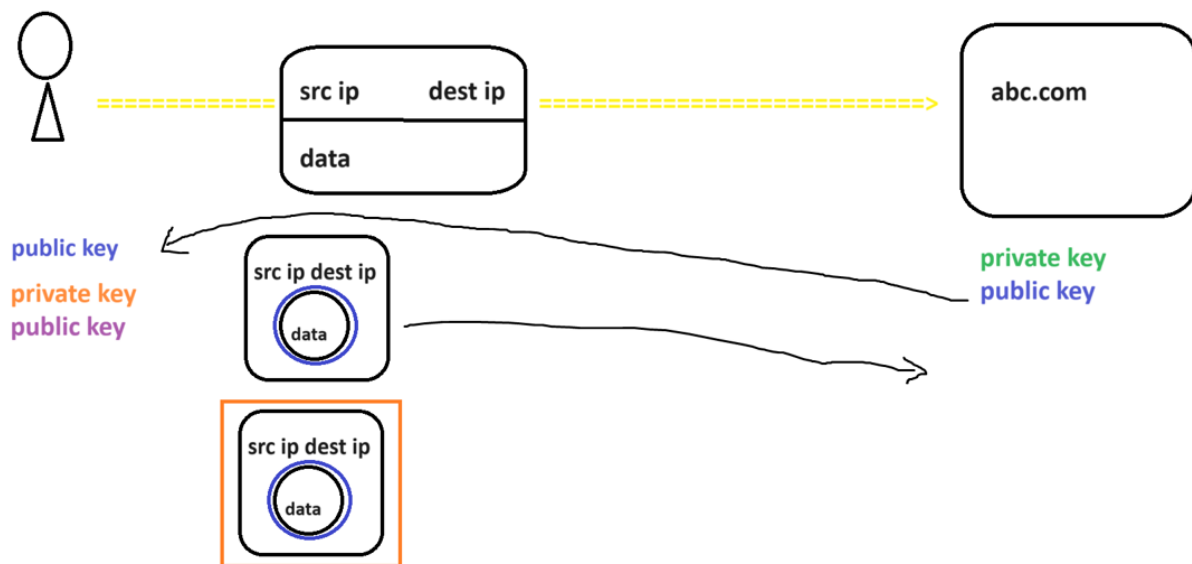
## Using HTTPS For the client-server communication

### 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
dr00l-html  dr00l.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>**

**[Set Up Password Authentication in Apache on CentOS/RedHat](#)**  
( Click here and will be diverted on google docs page. )