Step 1: Configure Nginx or Apache to require client certificates

For Nginx

1. **Install Nginx** (if not already installed): On Debian/Ubuntu:
   bash
   CopyEdit

```
sudo apt update
sudo apt install nginx
```

2. **Copy your Root CA and Intermediate CA certificates** to a directory (e.g., `/etc/ssl/certs/`):
   bash
   CopyEdit

```
sudo cp rootCA.crt /etc/ssl/certs/
sudo cp intermediateCA.crt /etc/ssl/certs/
```

3. **Edit your Nginx configuration** to enable SSL and require client certificates: Open the configuration file for your site (usually located in `/etc/nginx/sites-available/default` or `/etc/nginx/nginx.conf`):
   bash
   CopyEdit

```
sudo nano /etc/nginx/sites-available/default
```

   Modify it to include the SSL configuration and client certificate verification:
   nginx
   CopyEdit

```
server {
    listen 443 ssl;
```

```
    server_name example.com;

    ssl_certificate /etc/ssl/certs/server.crt;
    ssl_certificate_key /etc/ssl/private/server.key;

    ssl_client_certificate /etc/ssl/certs/rootCA.crt;
    ssl_verify_client on;

    location / {
        proxy_pass http://127.0.0.1:5000;  # For Flask app or PHP
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}


```

1 **Restart Nginx** to apply the changes:

bash
CopyEdit

```
sudo systemctl restart nginx.    brew services restart nginx
```

2

For Apache

1 **Install Apache** (if not already installed): On Debian/Ubuntu:

bash
CopyEdit

```
sudo apt update
```

2 `sudo apt install apache2.  brew install apache2`

3

4 **Copy the certificates to Apache's directory**:

bash
CopyEdit

```
sudo cp rootCA.crt /etc/ssl/certs/
```

5
```
sudo cp intermediateCA.crt /etc/ssl/certs/
```
6

7 **Enable SSL and configure Apache**: Edit the SSL configuration for your site (usually in `/etc/apache2/sites-available/default-ssl.conf` or `/etc/apache2/apache2.conf`):

bash
CopyEdit

```
sudo nano /etc/apache2/sites-available/default-ssl.conf
```

8

Add the following configurations to enable client certificate verification:

apache
CopyEdit

```
<VirtualHost _default_:443>
ServerAdmin webmaster@localhost
ServerName example.com
DocumentRoot /var/www/html

SSLEngine on
SSLCertificateFile /etc/ssl/certs/server.crt
SSLCertificateKeyFile /etc/ssl/private/server.key
SSLCACertificateFile /etc/ssl/certs/rootCA.crt
SSLVerifyClient require
SSLVerifyDepth 1

<Location />
    ProxyPass http://127.0.0.1:5000  # Flask app or other backend
    ProxyPassReverse http://127.0.0.1:5000
</Location>
```

```
</VirtualHost>
```

- `SSLCACertificateFile`: Specifies the root CA certificate.
- `SSLVerifyClient require`: Forces client certificate verification.

26 **Enable the SSL module and site**:

bash
CopyEdit

```
sudo a2enmod ssl
```

27
```
sudo a2ensite default-ssl.conf
```
28

29 **Restart Apache** to apply the changes:

bash
CopyEdit

```
sudo systemctl restart apache2
```

30

Step 2: Deploy the Python Flask App or PHP App behind the reverse proxy

For Python Flask App

1 **Install Flask** if it's not already installed:

bash
CopyEdit

```
pip install flask
```

2

3 **Create a simple Flask app** (e.g., `app.py`):

python
CopyEdit

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def index():
    client_cert = request.headers.get('X-Client-Cert')
    if client_cert:
        return f"Client certificate: {client_cert}"
    else:
        return "No client certificate presented", 403

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

18 **Run the Flask app**:

bash
CopyEdit

```
python app.py
```

19

20 The app will now run on `http://127.0.0.1:5000`, and the reverse proxy will forward requests from Nginx or Apache to this port.

For PHP App

1 **Install PHP** (if it's not already installed): On Debian/Ubuntu:

bash
CopyEdit

```
sudo apt update
sudo apt install php libapache2-mod-php
```

4 **Create a simple PHP file** (e.g., `index.php`) in `/var/www/html/`:

```
php
CopyEdit


<?php

if ($_SERVER['SSL_CLIENT_CERT']) {
    echo "Client certificate: " .
$_SERVER['SSL_CLIENT_CERT'];
} else {
    echo "No client certificate presented";
}
?>

```

12 **Restart Apache**:

```
bash
CopyEdit


sudo systemctl restart apache2

```

13

Now, the PHP app will be accessible, and the reverse proxy (Nginx or Apache) will forward client certificate information to the PHP app.

Step 3: Test the Setup

1 **Test with a Client Certificate**: When accessing the site, the client (browser or cURL) should provide a valid client certificate. You can use a tool like `curl` to simulate this:

```
bash
CopyEdit


curl --cert user.crt --key user.key https://
example.com

```

2

3 **Verify the Response**: The server should either return a successful response or display information about the client certificate, depending on whether the certificate is valid.

Summary

You've now set up a system where:

- Nginx or Apache acts as a reverse proxy.
- SSL/TLS client certificate authentication is required for accessing the app.
- A Flask or PHP app handles requests behind the reverse proxy.

This setup is useful for scenarios where you need to authenticate users or services based on their client certificates, ensuring a high level of security for your application.