Internship Project 2
Create user Login System using AWS Cognito and LAMP Stack
A Comprehensive Guide

Report By: Keshav Kumar
Date: 4th May 2025

ABSTRACT

This report presents the complete design, configuration, and testing of a secure user authentication system for a Laravel web application using Amazon Cognito, an identity management service offered by AWS. The solution leverages AWS Cognito to manage user sign-up and sign-in processes via OAuth2-based authentication and a customizable Hosted UI. This integration eliminates the need to build a custom authentication system from scratch, reducing development effort while enhancing security and scalability.

The Laravel application is configured to initiate the OAuth2 login flow and process callbacks from Cognito to authenticate users. The project also demonstrates environment-specific configurations, enabling seamless deployment in local development and optional deployment to a cloud-hosted EC2 instance running a LAMP stack (Linux, Apache, MySQL, PHP). Key processes, including user pool creation, application registration, route handling, and token exchange, are validated through successful login redirections and user session management.
These results confirm the effectiveness of using AWS Cognito as a centralized authentication mechanism for modern web applications. The project exemplifies the advantages of integrating managed cloud services with open-source frameworks to deliver secure, scalable, and maintainable user authentication for real-world production environments.

Key Highlights :

1. **OAuth2-Based Secure Authentication -**
   Integrated AWS Cognito with Laravel to provide secure login functionality using industry-standard OAuth2 protocols and a customizable Hosted UI.

2. **User Pool and App Client Configuration –**
   Configured Cognito User Pools and App Clients through the AWS Console to manage user identities, enable authorization flows, and support callback redirection for login success.

3. **Laravel Integration with AWS Cognito -**
   Implemented authentication flow in Laravel using pre-built packages and manual configuration, enabling seamless login, token parsing, and user session management.

4. **Tested Login via Hosted UI -**
   Verified functionality by logging into the Hosted UI, which successfully redirected back to the Laravel application with authentication tokens.

5. **EC2 Deployment with LAMP Stack -**
   Provisioned an EC2 instance, installed LAMP stack, configured DNS and SSL, demonstrating cloud-hosted deployment readiness for the Laravel app.

OBJECTIVE

Task 1: Create a user Login System using AWS Cognito and LAMP Stack

- **1.1 Project Setup:**
  - **Create Project Directory -**
    - Start by creating a dedicated folder for the project to organize source code and configuration files.
  - **Create AWS Cognito User Pool -**
    - Begin by setting up a Cognito User Pool, which acts as a secure and scalable user directory for managing registration, login, and user profiles.
    - Define required attributes like email, username, and phone_number, and configure verification via email or SMS.
  - **Set Up App Client and OAuth2 Flow -**
    - Create a new App Client within the User Pool to integrate with the Laravel application.
    - Enable OAuth 2.0 flows like **Authorization Code Grant** and **Implicit Grant** for secure authentication.
    - Configure the callback URLs (e.g., http://localhost:8000/callback) and select scopes like openid, email, and profile.
  - **Enable and Customize Hosted UI –**
    - Use the Cognito Hosted UI to simplify the login, registration, and account recovery pages.
    - Hosted UI handles the authentication interface, redirecting users back to the Laravel app after login.
  - **Install Laravel Project –**
    - Use Composer to create a Laravel Project.
    - Configure .env for database and base URL.
  - **Configure Apache for Laravel Deployment –**
    - Update Apache `VirtualHost` configuration to point to Laravel's `public/` folder
    - Enable mod_rewrite and ensure proper permissions.

- ## 1.2 Define AWS Cognito Resources:
  - **Create User Pool Schema** –
    - Define key attributes like `email`, `username`, and `phone_number`.
    - Enable multi-factor authentication (MFA) and account recovery settings.
    - Configure password policy to enhance security (e.g., min length, complexity).
  - **Configure App Client Settings** –
    - Create an App Client without generating a client secret (for public applications).
    - Enable OAuth2 flows (Authorization Code Grant, Implicit Grant).
    - Specify callback and sign-out URLs to your Laravel application endpoints.
  - **Create Domain for Hosted UI** –
    - Configure a unique Cognito domain (e.g., `yourapp.auth.region.amazoncognito.com`).
    - Optionally use a custom domain if branding is required.

- ## 1.3 Laravel Application Local Setup:
  - **Install Laravel Framework** –
    - Use Composer to install Laravel and required packages.
    - Verify PHP version compatibility and configure `.env` variables.
  - **Install AWS SDK and Socialite Packages** –
    - Install `aws/aws-sdk-php` for backend AWS interactions.
    - Use Laravel Socialite or `league/oauth2-client` for Cognito integration.
  - **Configure OAuth2 Environment Settings** –
    - Add Cognito client ID, client secret, redirect URLs to `.env` file.
    - Ensure values match Cognito App Client settings.

- ## 1.4 Integration and Deployment:
  - **Integrate Cognito with Laravel** –
    - Set up login, callback, and logout routes in Laravel.
    - Use OAuth2 client or Laravel Socialite for authentication.
    - Parse ID token to retrieve user details and manage sessions.
  - **Deploy on EC2 with LAMP Stack** –
    - Launch an EC2 instance and install Apache, PHP, and MySQL.
    - Upload Laravel app to `/var/www/html/` and set permissions.
  - **Configure DNS and SSL** –
    - Link EC2 public IP with domain via Route 53 or DNS provider.
    - Use Let's Encrypt (Certbot) to install SSL and enable HTTPS.

Create user Login System using AWS Cognito and LAMP Stack
A Comprehensive Guide

Introduction

Title - Exploring the world of Cloud Computing .

1. Understanding Cloud Computing -

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of traditional infrastructure management, users access services like computing power, storage, and databases from a cloud provider as needed.

Features :

- **On-demand delivery of IT resources:** This means you can access computing resources (like servers, storage, software) whenever you need them, without needing to wait for a long setup process.
- **Over the Internet:** The resources are accessed through the internet, allowing you to use them from anywhere with an internet connection.
- **Pay-as-you-go pricing:** You only pay for the resources you actually use. This is similar to how you pay for electricity or water.
- **Instead of traditional infrastructure management:** Cloud computing eliminates the need for businesses to buy and maintain their own physical servers, data centers, and IT staff.
- **Users access services...from a cloud provider:** Companies like Amazon (AWS), Microsoft (Azure), and Google (Google Cloud) provide these cloud computing services.

Service Models:

- **IaaS (Infrastructure as a Service):** Provides virtualized resources (VMs, storage, networks). Users manage OS and applications.

  Features:
  - **Virtualized resources:** Instead of physical hardware, you get virtualized resources. For example, a virtual machine (VM) simulates a physical server.
  - **VMs, storage, networks:** IaaS provides the basic building blocks of IT infrastructure. You get virtual servers, storage space, and networking capabilities.
  -
  - **Cloud provider manages the infrastructure:** The cloud provider is responsible for maintaining the physical servers, data centers, and network hardware.
- **PaaS (Platform as a Service):** Offers development platforms and tools. Developers build and deploy applications without managing underlying infrastructure.
-
  - **Development platforms and tools:** PaaS provides a complete environment for developers to build, test, and deploy software applications. This includes tools, libraries, and services.

- **Developers build and deploy applications:** PaaS is designed to make it easier for developers to create applications quickly.
- **Without managing underlying infrastructure:** Developers don't have to worry about the servers, operating systems, or network infrastructure. The cloud provider handles all of that.

- **SaaS (Software as a Service):** Delivers complete software solutions accessible via a web browser (e.g., email, CRM).

  Features:

  - **Complete software solutions:** SaaS provides fully functional applications that you can use without installing anything on your computer.
  - **Accessible via a web browser:** You can access the software through a web browser, from any device with an internet connection.
  - **Examples (e.g., email, CRM):** Examples of SaaS applications include Gmail, Salesforce (CRM), and Dropbox.

Deployment Models:

Public Cloud: Shared infrastructure operated by third-party providers.

Features:

- **Shared infrastructure:** In a public cloud, the cloud provider's hardware and resources are shared among multiple customers.
- **Third-party providers:** Companies like AWS, Azure, and Google Cloud own and operate public clouds.
- **Examples:** AWS, Microsoft Azure, Google Cloud Platform.

- **Private Cloud:** Dedicated infrastructure for a single organization.
  Features:

  - **Dedicated infrastructure:** A private cloud is set up for the exclusive use of a single company or organization.
  - **Single organization:** The resources are not shared with other customers.
  - **On-premises or hosted:** A private cloud can be located in the company's own data center or hosted by a third-party provider.

- **Hybrid Cloud:** A combination of public and private cloud environments.
  Features:

  - **Combination of public and private:** A hybrid cloud uses both public and private cloud resources.
  - **Data and application sharing:** Data and applications can be moved between the public and private clouds.
  - **Flexibility and options:** Hybrid clouds offer greater flexibility and allow organizations to optimize their infrastructure based on their specific needs.

Features:

- **Easily adjust resources:** Cloud computing allows you to quickly increase or decrease the amount of resources you are using (e.g., computing power, storage) as your needs change.
- **Changing demands:** If your website traffic increases, you can easily scale up your resources to handle the extra load. If traffic decreases, you can scale down to save money.
- **Cost Efficiency:** Pay-as-you-go pricing eliminates upfront investments and reduces operational expenses.
- **Pay-as-you-go pricing:** You only pay for the resources you use, which can be more cost-effective than buying and maintaining your own hardware.
- **Eliminates upfront investments:** You don't have to spend a lot of money upfront to buy servers and other infrastructure.
- **Reduces operational expenses:** Cloud computing can reduce the costs associated with running your own IT infrastructure, such as power, cooling, and IT staff.
- **High Availability:** Ensures continuous service availability through redundancy and fault tolerance.
- **Continuous service availability:** Cloud computing helps to ensure that your applications and data are always available.
- **Redundancy and fault tolerance:** Cloud providers use multiple data centers and redundant systems to minimize downtime in case of failures.
- **Agility and Speed:** Quickly provision and deploy resources, accelerating development and innovation.
- **Quickly provision and deploy resources:** You can quickly get the resources you need to set up and run applications, without long delays.
- **Accelerating development and innovation:** Cloud computing makes it easier and faster to develop and deploy new applications, which can help businesses innovate more quickly.
- **Global Reach:** Access resources and services from anywhere in the world.
- **Access resources...from anywhere:** Cloud computing allows you to access your applications and data from anywhere with an internet connection.
- **Global network of data centers:** Cloud providers have data centers around the world, which allows them to offer services to users globally.

3.2 AWS Cognito –

AWS Cognito is a service that provides authentication, authorization, and user management for your web and mobile apps. It supports sign-in with user name and password as well as with social identity providers like Google, Facebook, and Amazon, and enterprise identity providers via SAML.

You only pay for active users and the authentication operations they perform.

Features of AWS Cognito:

• User Pools:

- A User Pool is a user directory in Amazon Cognito where users can sign up and sign in.
- Cognito handles user registration, authentication, and account recovery.
- Supports multi-factor authentication (MFA), password policies, and account verification via email or SMS.
- Users can sign in through a customizable Hosted UI or through your app directly using the SDK.

• Identity Pools:

- Identity Pools provide temporary AWS credentials to access other AWS services.
- Can be used to authenticate users via Cognito User Pools, social identity providers (Google, Facebook), or SAML-based identity providers.
- Enables fine-grained access control to AWS services using IAM roles.

• Secure Authentication:

- Built-in support for securing authentication with OAuth 2.0, OpenID Connect (OIDC), and SAML.
- Token-based authentication using Access, ID, and Refresh Tokens.
- Ensures secure backend communication and resource access management.

• Federation:

- Users can sign in using social identity providers (Google, Facebook, Apple), SAML-based identity providers, or your own custom identity system.
- Supports single sign-on (SSO) across applications.

• Fine-Grained Access Control:

- Use AWS IAM policies to control which users can access which AWS resources.
- Supports condition-based access policies, including by user group, token claims, or device metadata.

Examples of Authentication Flows –

AWS Cognito supports a variety of authentication flows that cater to diverse application needs and security requirements. These flows provide flexibility to integrate with both traditional and modern identity systems, ensuring secure and scalable user access management. Below are key examples of supported authentication mechanisms:

1.3.1 Username/Password Sign-In

This is the most traditional and widely-used authentication method, where users register and authenticate using their credentials directly with an Amazon Cognito User Pool.

- Flow: Users sign up or sign in using a username (or email/phone) and a password via Cognito's hosted UI or application interface.
- Security: Cognito provides built-in support for password policies, multi-factor authentication (MFA), and account recovery options.
- Use Case: Ideal for custom applications requiring in-app registration and login without external identity providers.
- Example: A mobile app that requires users to create accounts directly within the app interface.

1.3.2 Federated Identity Login (Social Providers)

Cognito supports federated authentication through social identity providers like Google, Facebook, Apple, and Amazon.

- Flow: Users choose a social login option, are redirected to the provider's authentication page, and upon successful login, are returned to the application with a Cognito-generated token.
- Integration: Easily configured via Cognito’s Identity Pools and Federated Identities section.
- Use Case: Applications that aim to provide convenience to users by letting them sign in with accounts they already trust.
- Example: A web app where users can log in with their Google account instead of creating a new account.

1.3.3 Enterprise Login via SAML (Security Assertion Markup Language)

Cognito can connect to enterprise identity providers using SAML 2.0, enabling single sign-on (SSO) with corporate credentials such as Microsoft Active Directory or Okta.

- Flow: Users are redirected to their organization's SSO login portal. Upon successful authentication, a SAML assertion is sent to Cognito to establish the session.
- Compliance & Security: Aligns with enterprise security policies and compliance requirements such as SAML assertions, session lifetimes, and audit trails.
- Use Case: Business applications that need to provide secure access to internal users or partners using their organization’s credentials.
- Example: An internal dashboard accessed by employees via Active Directory SSO.

1.3.4 Custom Authentication Flow

For unique or advanced use cases, AWS Cognito supports custom authentication workflows using AWS Lambda triggers.

- Flow: Developers define custom logic at various stages such as Pre Sign-Up, Pre Authentication, Post Authentication, Define Auth Challenge, etc.
- Use Cases:
    - User Migration: Seamlessly import users from an existing identity store during their first login.
    - Progressive Profiling: Prompt users for additional information after authentication.
    - One-time Passcodes (OTP): Replace standard password flow with email or SMS OTP.
    - Post-Login Actions: Implement additional validations or log events after successful login.
- Example: A platform that wants to validate users against an external API before allowing access, or perform fraud checks after login.

Use Cases:

• Web and Mobile App Authentication:

- AWS Cognito provides a secure, scalable, and fully managed user directory for web and mobile apps.
- Developers can authenticate users for access to front-end interfaces and protect APIs using industry-standard authentication protocols.
- Cognito integrates with frameworks like React, Angular, Android, and iOS through SDKs, enabling seamless user sign-up/sign-in experiences.
- It also supports custom attributes and user metadata for richer user profile management.

• Enterprise Single Sign-On (SSO) Integration:

- Cognito supports integration with enterprise identity providers using SAML 2.0 and OIDC, allowing employees to use their corporate credentials to log in.
- This makes it ideal for internal enterprise applications that require centralized identity management and SSO across multiple services.
- Cognito also supports Role Mapping, which allows different user groups in a corporate directory to be assigned varying levels of access within your application.

• Secure API Gateway Access:

- You can protect backend APIs (e.g., hosted on Amazon API Gateway or Lambda) by validating JWT tokens issued by Cognito.
- The token contains user identity and claims, which can be used for fine-grained authorization.
- This ensures that only authenticated users with valid roles and permissions can access protected endpoints or perform specific actions.

• Custom Authentication Flows:

- AWS Cognito can be extended using AWS Lambda triggers to create tailored authentication workflows.
- Examples include:
  - CAPTCHA validation before sign-up to prevent bot abuse.
  - Sending custom One-Time Passwords (OTPs) via email or SMS.
  - Enforcing age verification or terms acceptance before account creation.
  - Integration with third-party risk assessment or fraud detection services.
- Custom flows provide flexibility to meet compliance, user experience, and security requirements not available in default authentication flows.

• Multi-Tenant SaaS Applications:

- Cognito allows user segregation and group-level access control, making it ideal for Software-as-a-Service (SaaS) platforms.
- Each tenant (customer) can have their users logically grouped with specific roles and access rights.

• Progressive Profiling and User Lifecycle Management:

- Applications can start with minimal user data (e.g., email).
- Developers can use Cognito APIs to update user attributes, manage sessions, revoke tokens, and perform account deletions.
- Supports account recovery features like email or SMS-based password reset.

Create user Login System using AWS Cognito and LAMP Stack

A Comprehensive Guide

METHODOLOGY

This section outlines the detailed, step-by-step methodology adopted to design and implement a secure user login system by integrating Amazon Web Services (AWS) Cognito with a traditional LAMP (Linux, Apache, MySQL, PHP) stack. The approach focuses on leveraging AWS Cognito's robust identity and access management features to handle user authentication and authorization securely, while utilizing the reliability and flexibility of the LAMP stack for hosting the application backend. The methodology ensures that best practices in cloud security, system architecture, and web application development are followed. Each phase of the implementation—from infrastructure setup and identity management configuration to domain integration, deployment, and testing—has been carefully executed to build a seamless, secure, and scalable user authentication system.

Requirements Analysis and System Design-

3.1 Identity and Access Control Strategy -

- IAM (Identity and Access Management):
  - Used to manage AWS resources securely.
  - Roles and policies were defined for administrators to access services like Cognito and EC2.

- **Customer IAM (cIAM) via AWS Cognito:**
  - Enables user-level identity and access management.
  - Used for user registration, login, and authentication flows.

- **Cognito Functionalities:**
  - Allows creation and management of users.
  - Supports Single Sign-On (SSO) based on OAuth2 standards.
  - Ensures secure authorization and token management**.**

3.2 Architecture Planning –

- Designed a layered architecture to integrate identity management, backend logic, and domain services.
- Identified and mapped out all required AWS services and LAMP components.
- Ensured separation of admin-level control (IAM) from user-level authentication (Cognito).
- Planned secure communication between all system modules.

3.3 User Interface (UI) Design -

- Laravel-based PHP UI was developed to handle authentication states.
- UI contained links or buttons to redirect users to Cognito Hosted UI.
- Provided user feedback post-login such as token status or welcome messages.
- Kept interface clean and focused on demonstrating authentication.

3.4 LAMP Stack Setup on EC2 –

- Linux Server: Deployed Ubuntu instance via EC2.
- Apache Web Server: Installed and configured to host PHP apps.
- MySQL: Installed for database support (not central to Cognito but included).
- PHP: Configured with required extensions to support web application logic.
- IP Access: Public IP was used during early testing before domain configuration.

3.5 AWS Cognito Configuration -

- **User Pool Creation**:
  - Set up to manage user lifecycle (registration, login, profile).
- **App Client Configuration**:
  - Defined callback URLs for redirection post-login.
  - Enabled OAuth2 flows and scopes (e.g., email, profile).
  - Client secret was generated for secure token exchanges.
- **App Integration**:
  - Linked Cognito with the application via endpoints.
  - Configured redirect URIs and token handling mechanisms.

3.6 Domain and DNS Setup –

- **Domain Registration**:
  - Registered cognito.fsdevtutor.shop via Godaddy.com or you can use any other external provider.
- **DNS Configuration**:
  - Pointed domain A-record to EC2 instance's IP address.
  - Enabled clean URL access to the Laravel application.
- **Hosted UI Integration**:
  - Domain was used as a trusted redirect endpoint in Cognito settings.

3.7 PHP Application for Cognito Token Handling-

- **OAuth2 Flow Integration**:
    - Application redirected users to Hosted UI for login.
    - Captured authorization code from redirect URI.

- **Token Exchange**:
    - Authorization code was exchanged for access and ID tokens.
    - Tokens were stored temporarily for session use.

- **User Info Retrieval**:
    - Used access tokens to fetch user profile details from Cognito.
    - Ensured secure token validation before granting access.

3.8 Git Configuration for Deployment –

- **Version Control Setup**:
    - Local repository managed using Git.
    - EC2 server configured with SSH access for Git operations.

- **Code Deployment**:
    - Code was pushed to the EC2 instance using git pull.
    - Enabled continuous updates without manual file transfers.

3.9 Application Deployment on LAMP Server –

- **File Structure**:
    - Laravel project placed in Apache root directory (/var/www/html).

- **Server Configuration**:
    - Apache settings updated to serve Laravel routes.
    - Environment variables set for production mode.
- **Permissions**:
    - Correct file permissions assigned to avoid access issues.

- **Security Enhancements**:
    - .env file secured.
    - Access restricted using firewalls and SSL configuration.

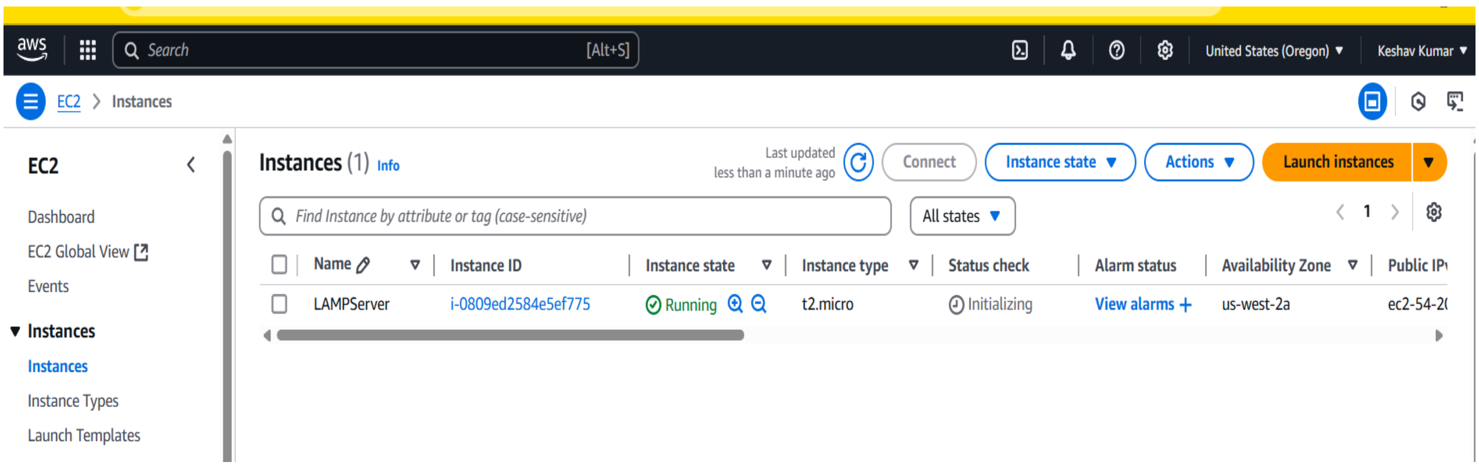
3.10 System Testing and Validation-

- Verified user authentication by testing and token-based access through AWS Cognito.
- Confirmed secure login workflows by simulating both successful and failed login attempts.

Create user Login System using AWS Cognito and LAMP Stack

A Comprehensive Guide

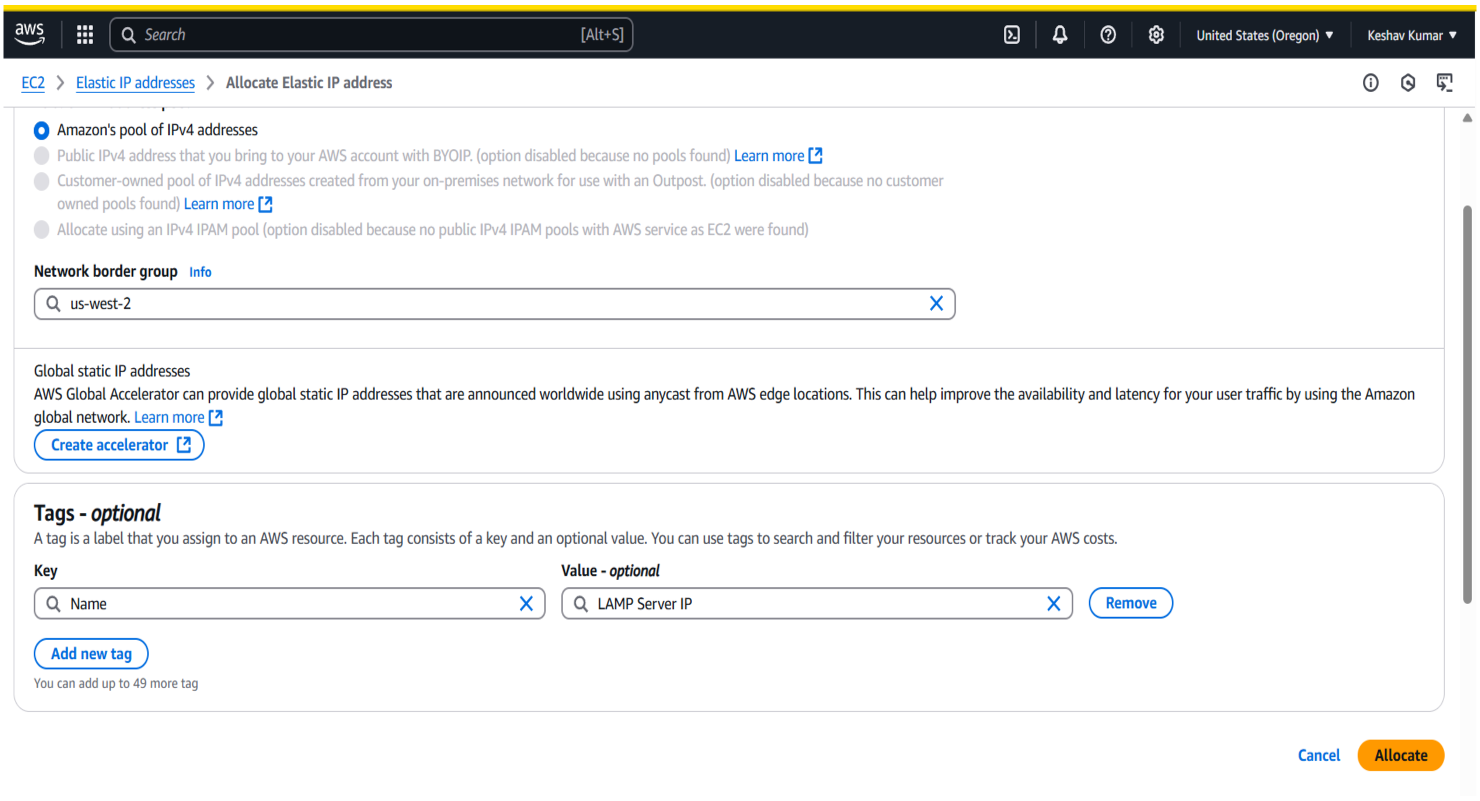
SCREENSHOTS


EC2 > Instances

Instances (1) Info — Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive) — All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
LAMPServer	i-0809ed2584e5ef775	Running	t2.micro	Initializing	View alarms +	us-west-2a	ec2-54-2(


EC2 > Elastic IP addresses > Allocate Elastic IP address

[✓ Amazon's pool of IPv4 addresses]
Public IPv4 address that you bring to your AWS account with BYOIP. (option disabled because no pools found) Learn more
Customer-owned pool of IPv4 addresses created from your on-premises network for use with an Outpost. (option disabled because no customer owned pools found) Learn more
Allocate using an IPv4 IPAM pool (option disabled because no public IPv4 IPAM pools with AWS service as EC2 were found)

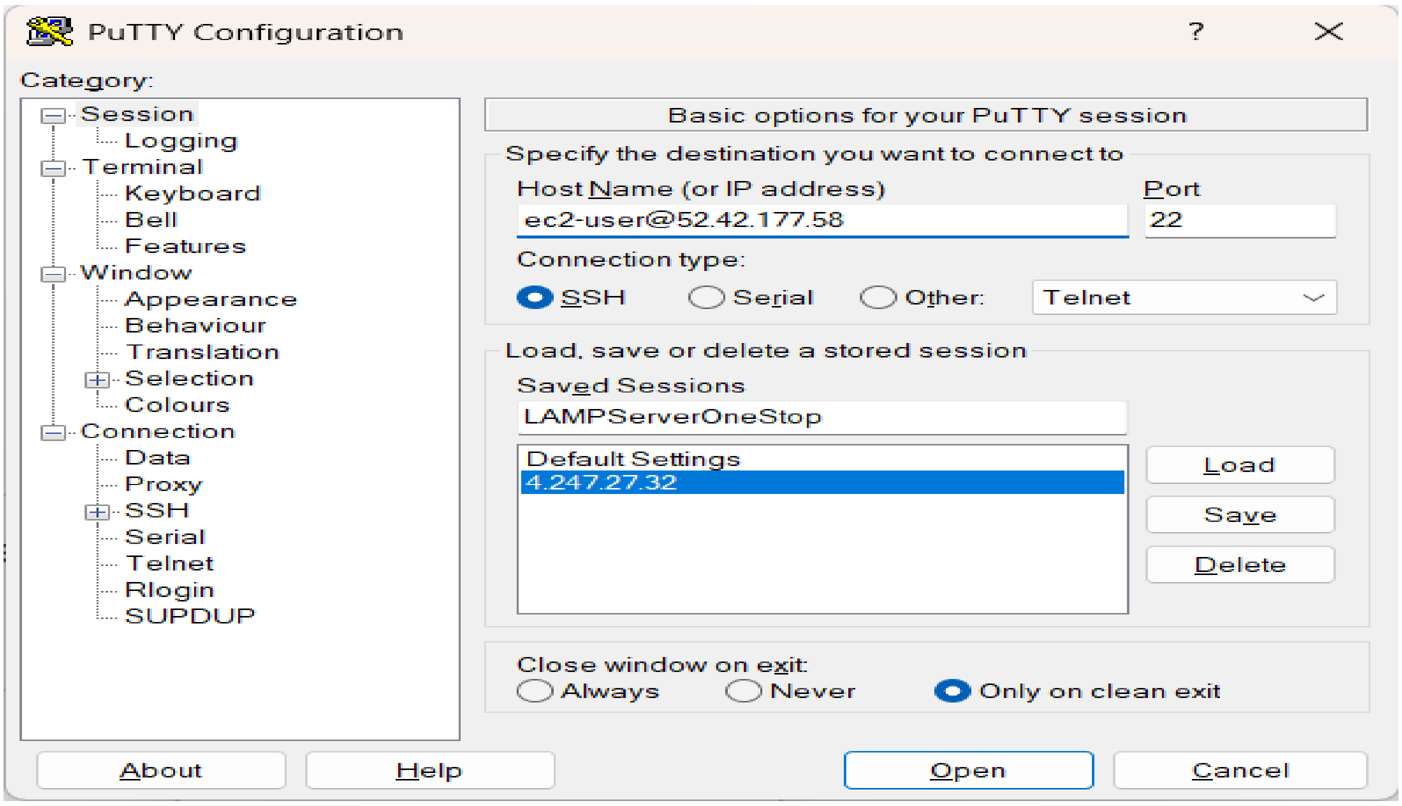
Network border group Info: us-west-2

Global static IP addresses
AWS Global Accelerator can provide global static IP addresses that are announced worldwide using anycast from AWS edge locations. This can help improve the availability and latency for your user traffic by using the Amazon global network. Learn more

Tags - *optional*
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - *optional*
Name	LAMP Server IP

You can add up to 49 more tag



PuTTY Configuration

Category:

- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - SSH
  - Serial
  - Telnet
  - Rlogin
  - SUPDUP

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-user@52.42.177.58

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

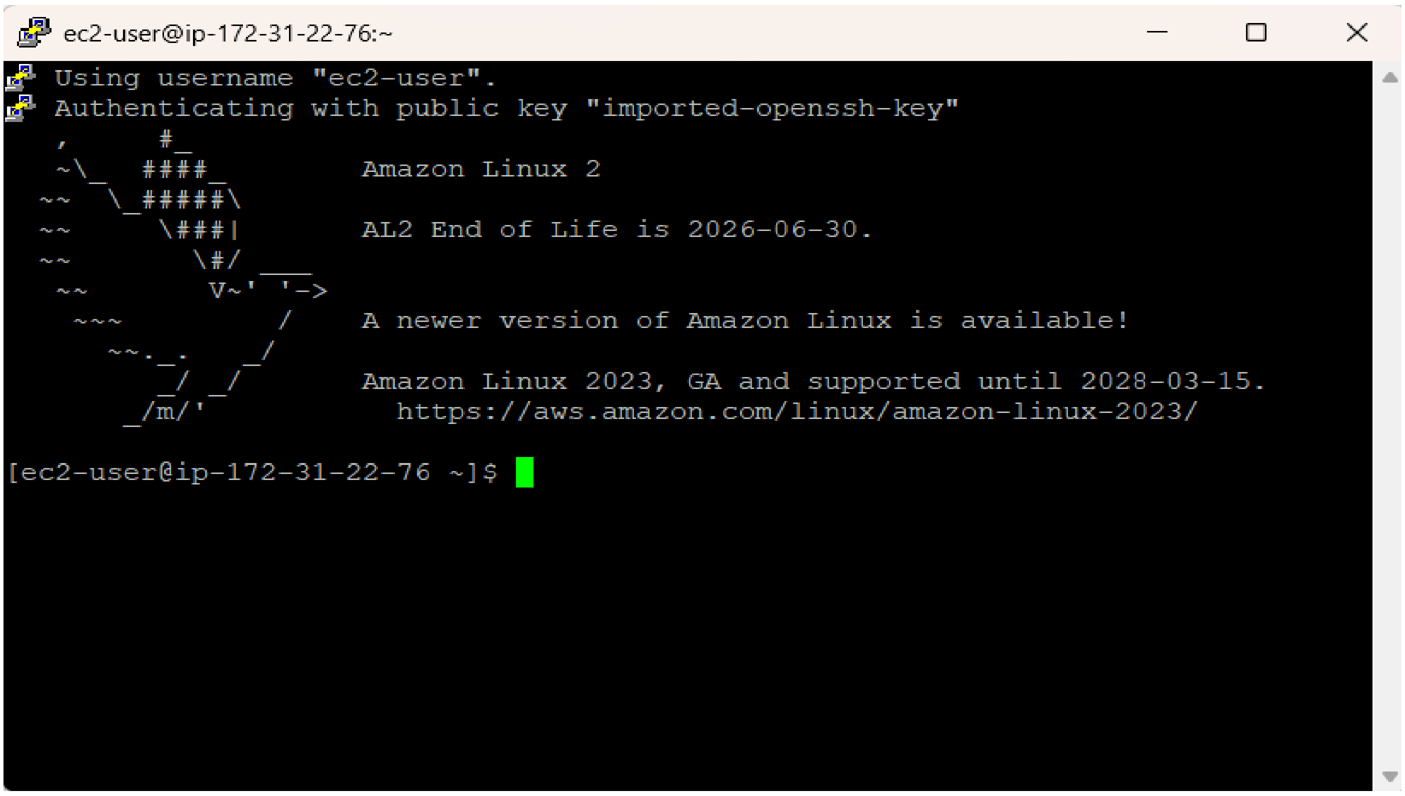
Saved Sessions: LAMPServerOneStop

Default Settings
4.247.27.32

Load | Save | Delete

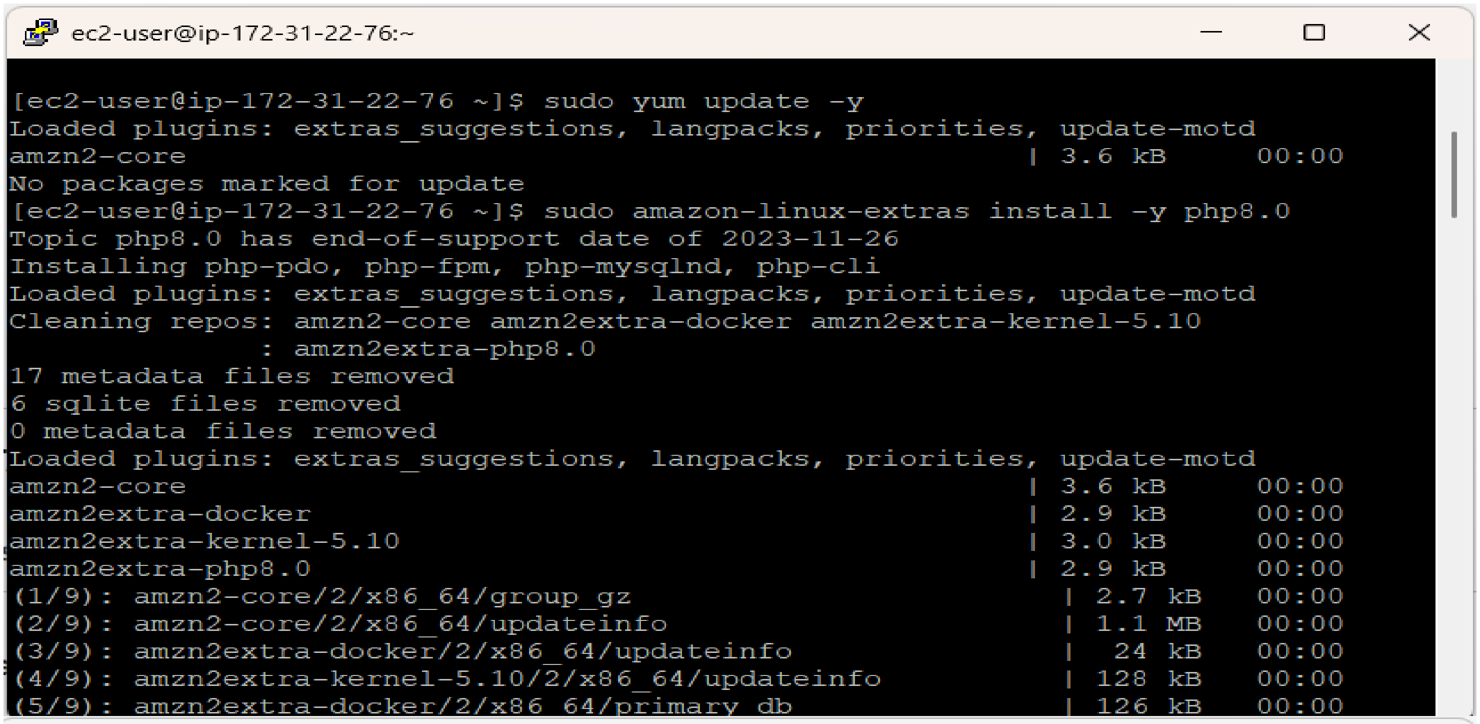
Close window on exit: Always Never [✓ Only on clean exit]

About | Help | Open | Cancel



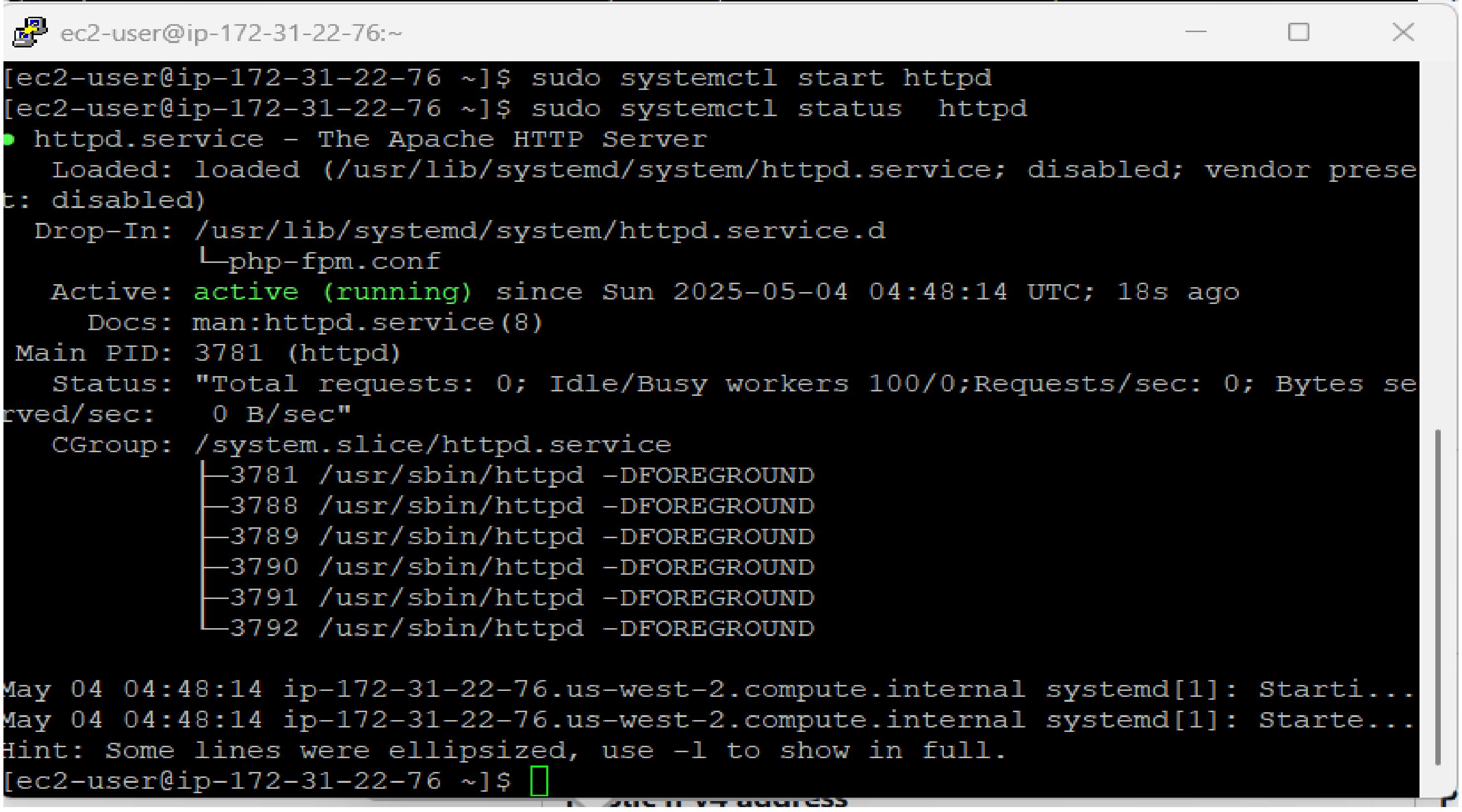
ec2-user@ip-172-31-22-76:~

```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-22-76 ~]$
```



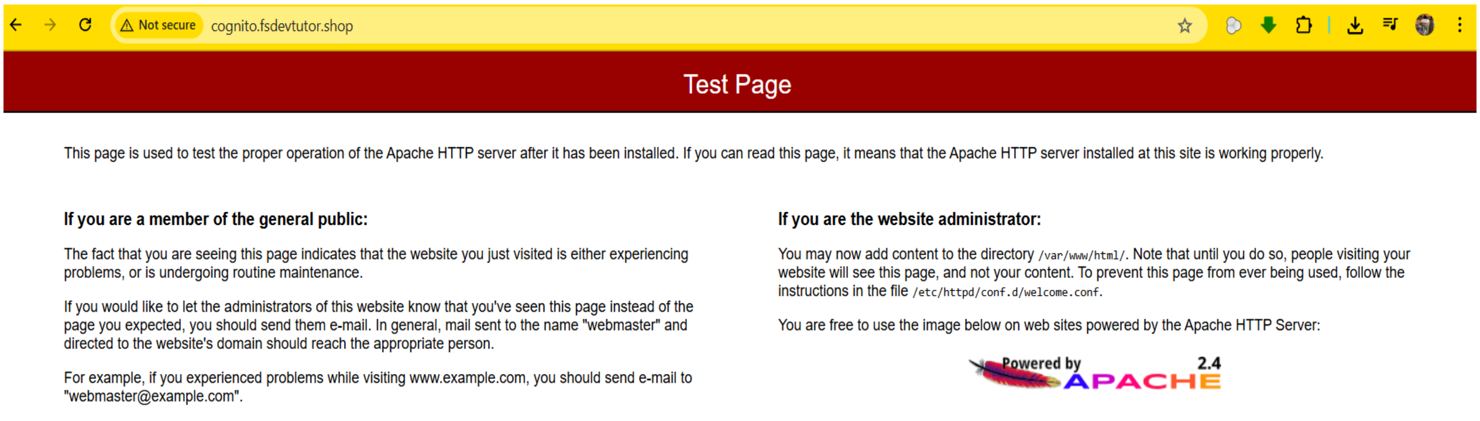
ec2-user@ip-172-31-22-76:~

```
[ec2-user@ip-172-31-22-76 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.6 kB     00:00
No packages marked for update
[ec2-user@ip-172-31-22-76 ~]$ sudo amazon-linux-extras install -y php8.0
Topic php8.0 has end-of-support date of 2023-11-26
Installing php-pdo, php-fpm, php-mysqlnd, php-cli
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
              : amzn2extra-php8.0
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 3.6 kB     00:00
amzn2extra-docker                                        | 2.9 kB     00:00
amzn2extra-kernel-5.10                                   | 3.0 kB     00:00
amzn2extra-php8.0                                        | 2.9 kB     00:00
(1/9): amzn2-core/2/x86_64/group_gz                        | 2.7 kB   00:00
(2/9): amzn2-core/2/x86_64/updateinfo                      | 1.1 MB   00:00
(3/9): amzn2extra-docker/2/x86_64/updateinfo               |  24 kB   00:00
(4/9): amzn2extra-kernel-5.10/2/x86_64/updateinfo          | 128 kB   00:00
(5/9): amzn2extra-docker/2/x86_64/primary_db               | 126 kB   00:00
```



ec2-user@ip-172-31-22-76:~

```
[ec2-user@ip-172-31-22-76 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-22-76 ~]$ sudo systemctl status  httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor prese
t: disabled)
  Drop-In: /usr/lib/systemd/system/httpd.service.d
           └─php-fpm.conf
   Active: active (running) since Sun 2025-05-04 04:48:14 UTC; 18s ago
     Docs: man:httpd.service(8)
 Main PID: 3781 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes se
rved/sec:   0 B/sec"
   CGroup: /system.slice/httpd.service
           ├─3781 /usr/sbin/httpd -DFOREGROUND
           ├─3788 /usr/sbin/httpd -DFOREGROUND
           ├─3789 /usr/sbin/httpd -DFOREGROUND
           ├─3790 /usr/sbin/httpd -DFOREGROUND
           ├─3791 /usr/sbin/httpd -DFOREGROUND
           └─3792 /usr/sbin/httpd -DFOREGROUND

May 04 04:48:14 ip-172-31-22-76.us-west-2.compute.internal systemd[1]: Starti...
May 04 04:48:14 ip-172-31-22-76.us-west-2.compute.internal systemd[1]: Starte...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-22-76 ~]$
```



Test Page

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

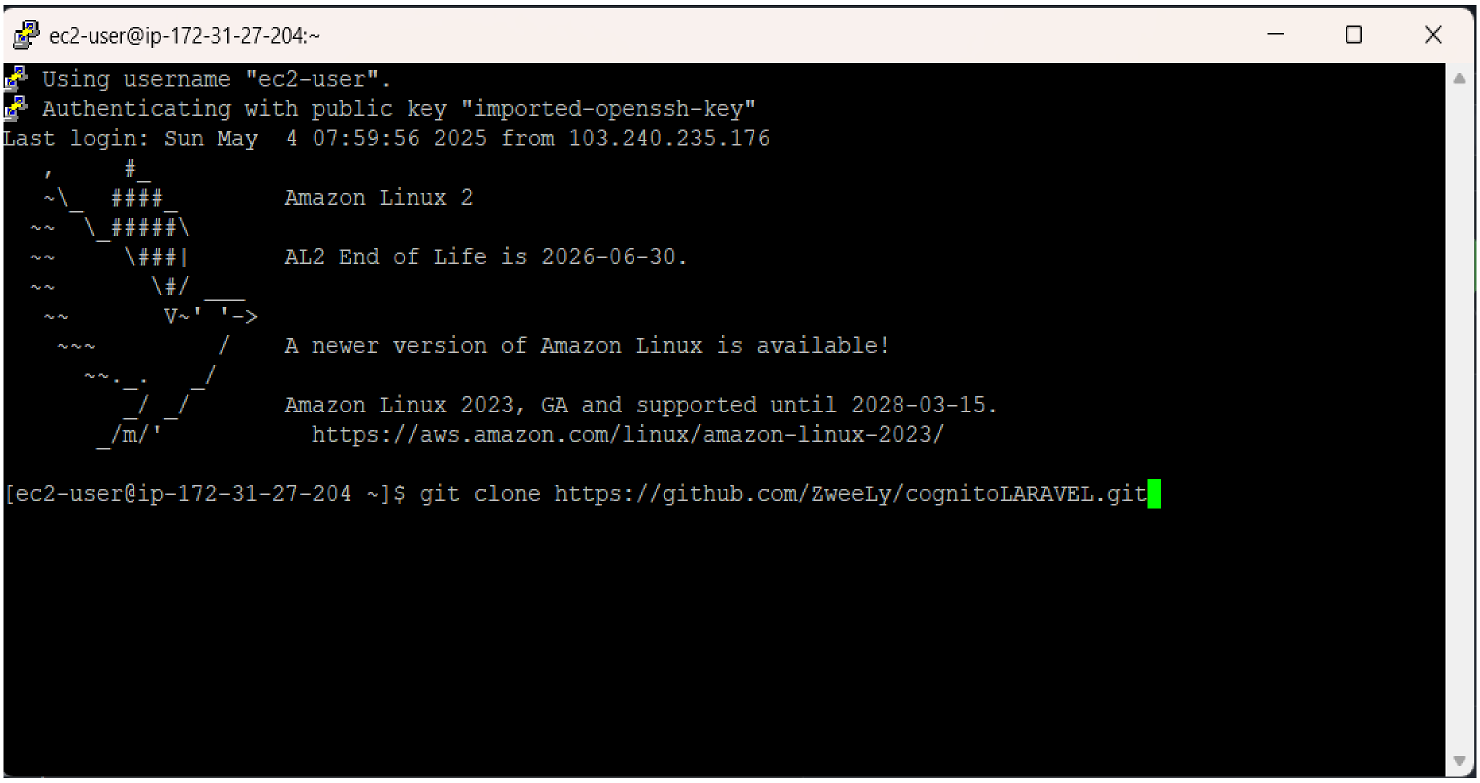
If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

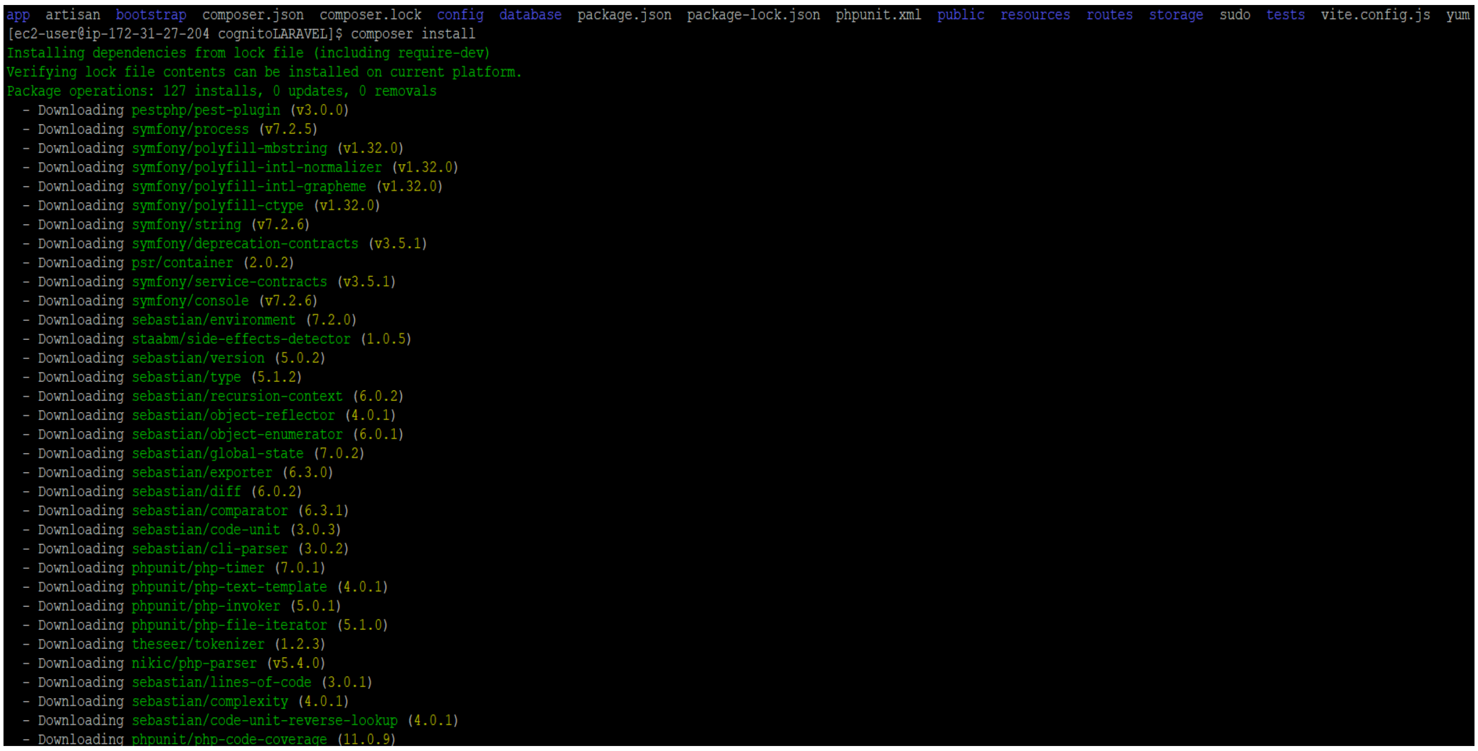
If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

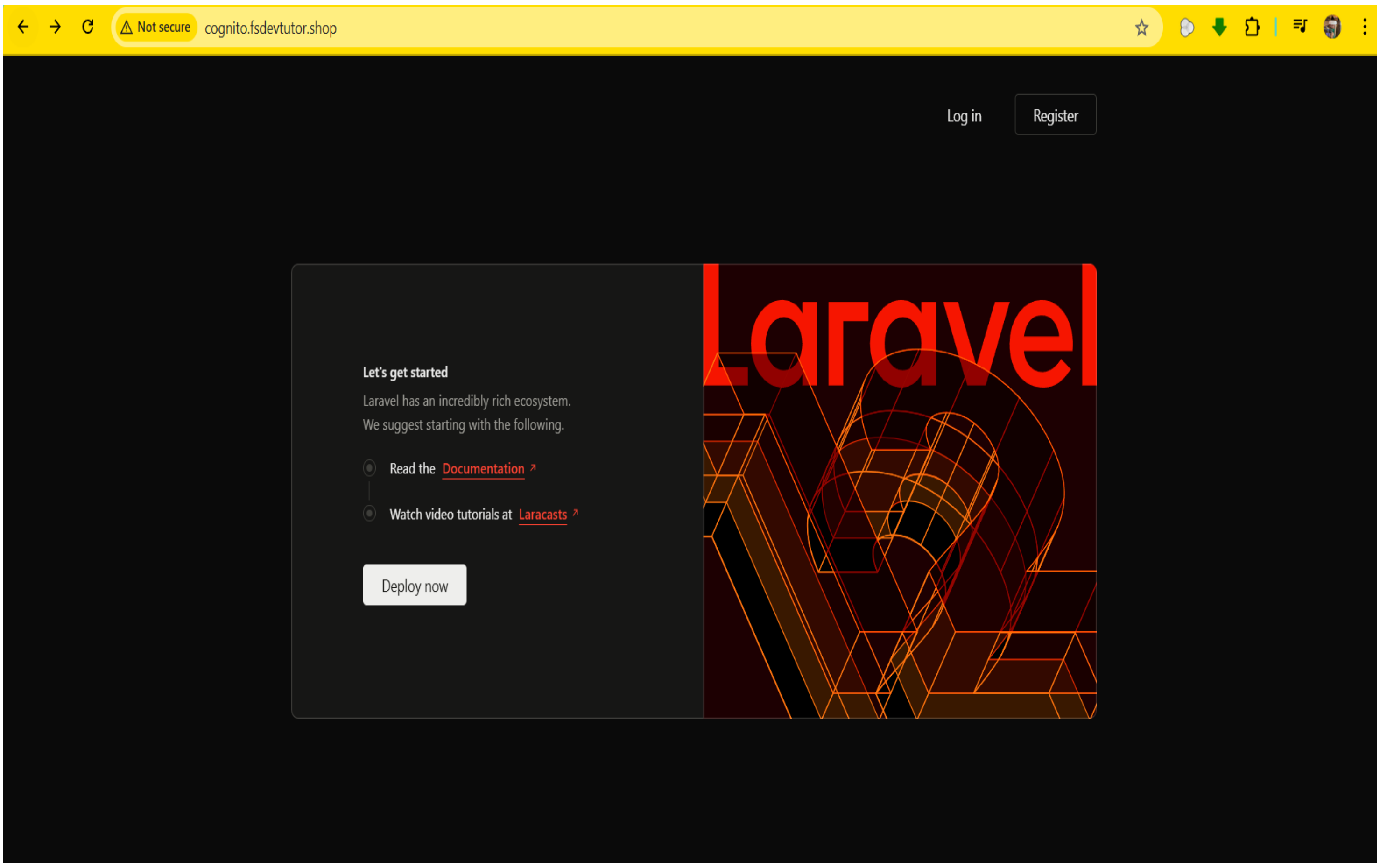
You are free to use the image below on web sites powered by the Apache HTTP Server:



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Sun May  4 07:59:56 2025 from 103.240.235.176
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-27-204 ~]$ git clone https://github.com/ZweeLy/cognitoLARAVEL.git
```



```
app  artisan  bootstrap  composer.json  composer.lock  config  database  package.json  package-lock.json  phpunit.xml  public  resources  routes  storage  sudo  tests  vite.config.js  yum
[ec2-user@ip-172-31-27-204 cognitoLARAVEL]$ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 127 installs, 0 updates, 0 removals
  - Downloading pestphp/pest-plugin (v3.0.0)
  - Downloading symfony/process (v7.2.5)
  - Downloading symfony/polyfill-mbstring (v1.32.0)
  - Downloading symfony/polyfill-intl-normalizer (v1.32.0)
  - Downloading symfony/polyfill-intl-grapheme (v1.32.0)
  - Downloading symfony/polyfill-ctype (v1.32.0)
  - Downloading symfony/string (v7.2.6)
  - Downloading symfony/deprecation-contracts (v3.5.1)
  - Downloading psr/container (2.0.2)
  - Downloading symfony/service-contracts (v3.5.1)
  - Downloading symfony/console (v7.2.6)
  - Downloading sebastian/environment (7.2.0)
  - Downloading staabm/side-effects-detector (1.0.5)
  - Downloading sebastian/version (5.0.2)
  - Downloading sebastian/type (5.1.2)
  - Downloading sebastian/recursion-context (6.0.2)
  - Downloading sebastian/object-reflector (4.0.1)
  - Downloading sebastian/object-enumerator (6.0.1)
  - Downloading sebastian/global-state (7.0.2)
  - Downloading sebastian/exporter (6.3.0)
  - Downloading sebastian/diff (6.0.2)
  - Downloading sebastian/comparator (6.3.1)
  - Downloading sebastian/code-unit (3.0.3)
  - Downloading sebastian/cli-parser (3.0.2)
  - Downloading phpunit/php-timer (7.0.1)
  - Downloading phpunit/php-text-template (4.0.1)
  - Downloading phpunit/php-invoker (5.0.1)
  - Downloading phpunit/php-file-iterator (5.1.0)
  - Downloading theseer/tokenizer (1.2.3)
  - Downloading nikic/php-parser (v5.4.0)
  - Downloading sebastian/lines-of-code (3.0.1)
  - Downloading sebastian/complexity (4.0.1)
  - Downloading sebastian/code-unit-reverse-lookup (4.0.1)
  - Downloading phpunit/php-code-coverage (11.0.9)
```



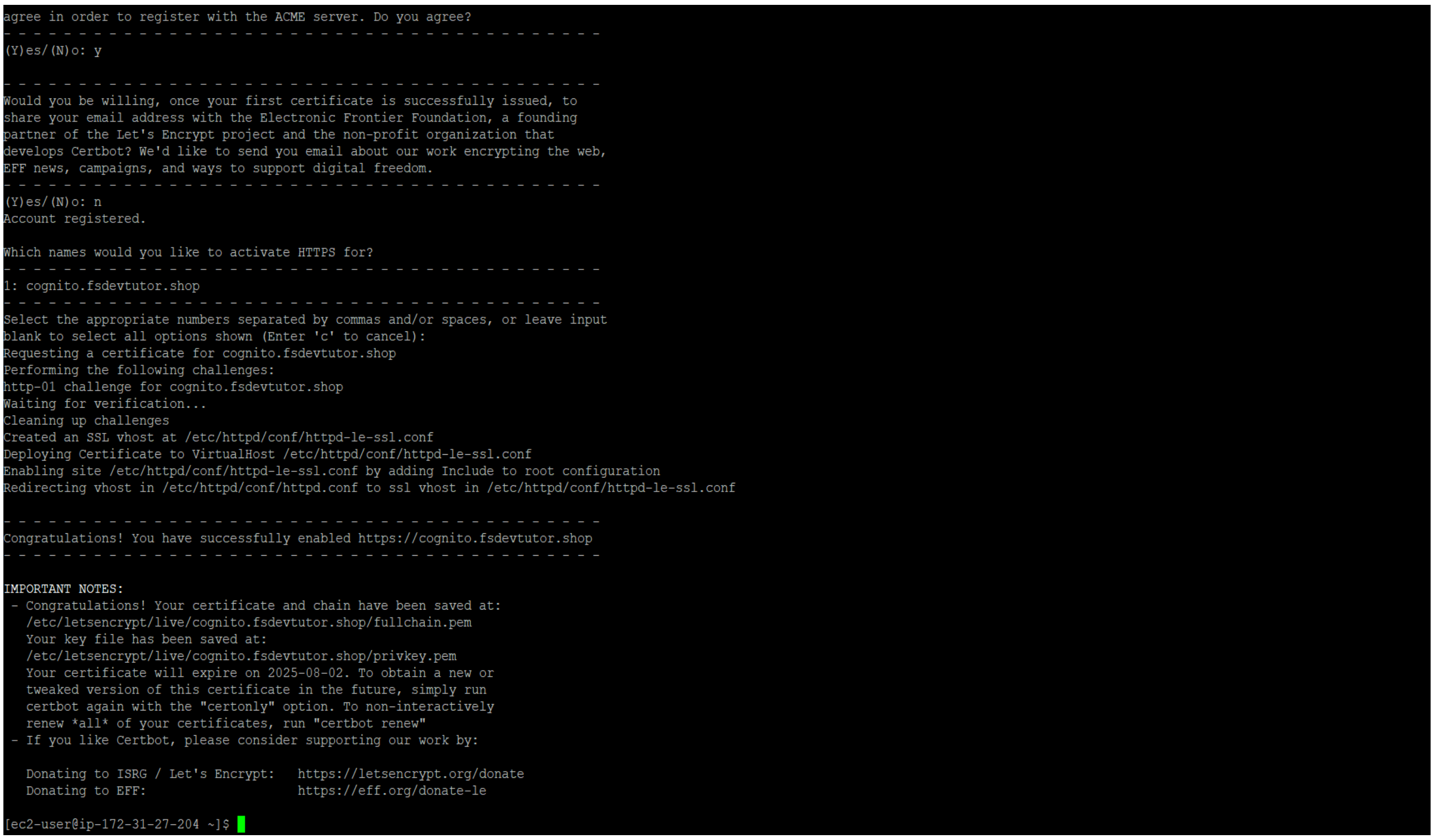
Let's get started

Laravel has an incredibly rich ecosystem.
We suggest starting with the following.

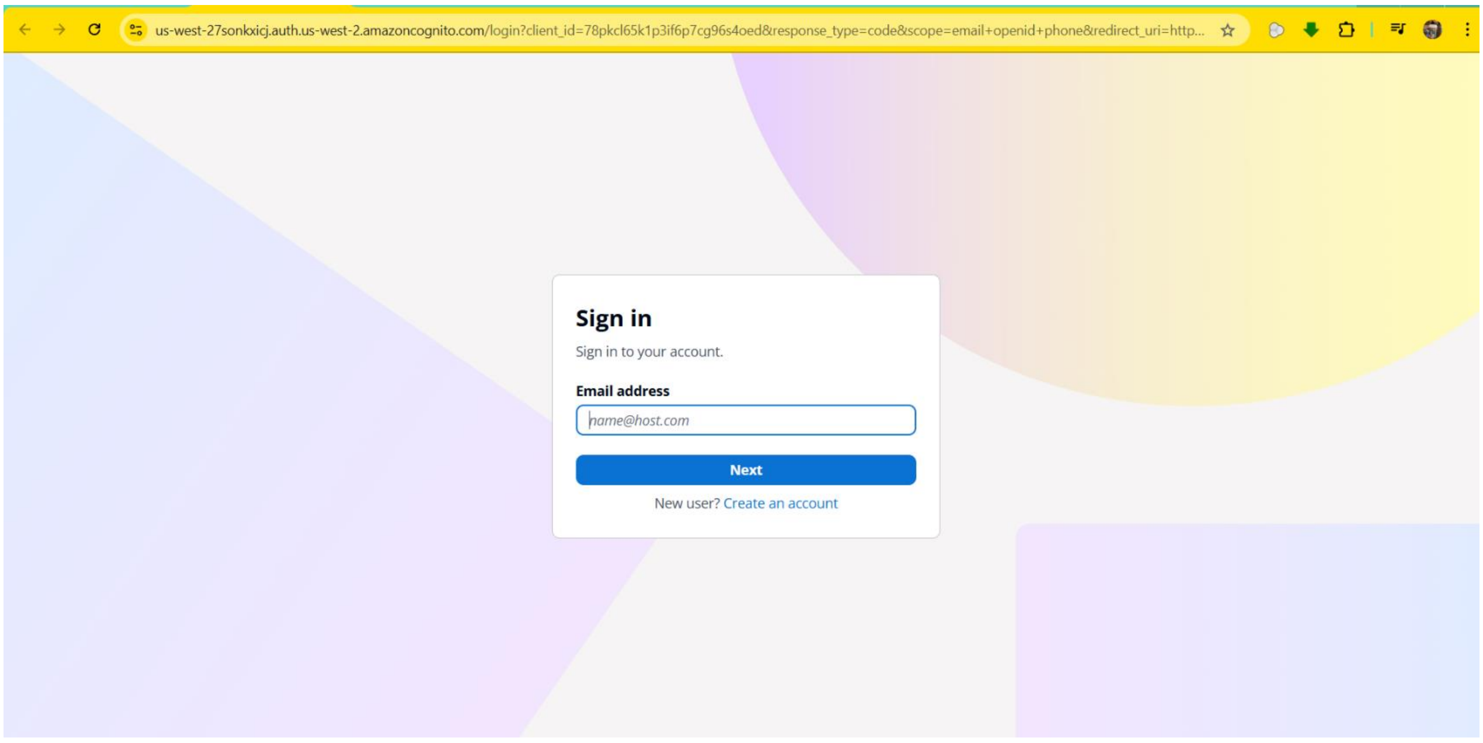
Read the Documentation

Watch video tutorials at Laracasts

Deploy now



```
agree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: n
Account registered.

Which names would you like to activate HTTPS for?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: cognito.fsdevtutor.shop
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel):
Requesting a certificate for cognito.fsdevtutor.shop
Performing the following challenges:
http-01 challenge for cognito.fsdevtutor.shop
Waiting for verification...
Cleaning up challenges
Created an SSL vhost at /etc/httpd/conf/httpd-le-ssl.conf
Deploying Certificate to VirtualHost /etc/httpd/conf/httpd-le-ssl.conf
Enabling site /etc/httpd/conf/httpd-le-ssl.conf by adding Include to root configuration
Redirecting vhost in /etc/httpd/conf/httpd.conf to ssl vhost in /etc/httpd/conf/httpd-le-ssl.conf

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations! You have successfully enabled https://cognito.fsdevtutor.shop
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/cognito.fsdevtutor.shop/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/cognito.fsdevtutor.shop/privkey.pem
   Your certificate will expire on 2025-08-02. To obtain a new or
   tweaked version of this certificate in the future, simply run
   certbot again with the "certonly" option. To non-interactively
   renew *all* of your certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

[ec2-user@ip-172-31-27-204 ~]$
```



us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/login?client_id=78pkcl65k1p3if6p7cg96s4oed&response_type=code&scope=email+openid+phone&redirect_uri=http...

Sign in

Sign in to your account.

Email address

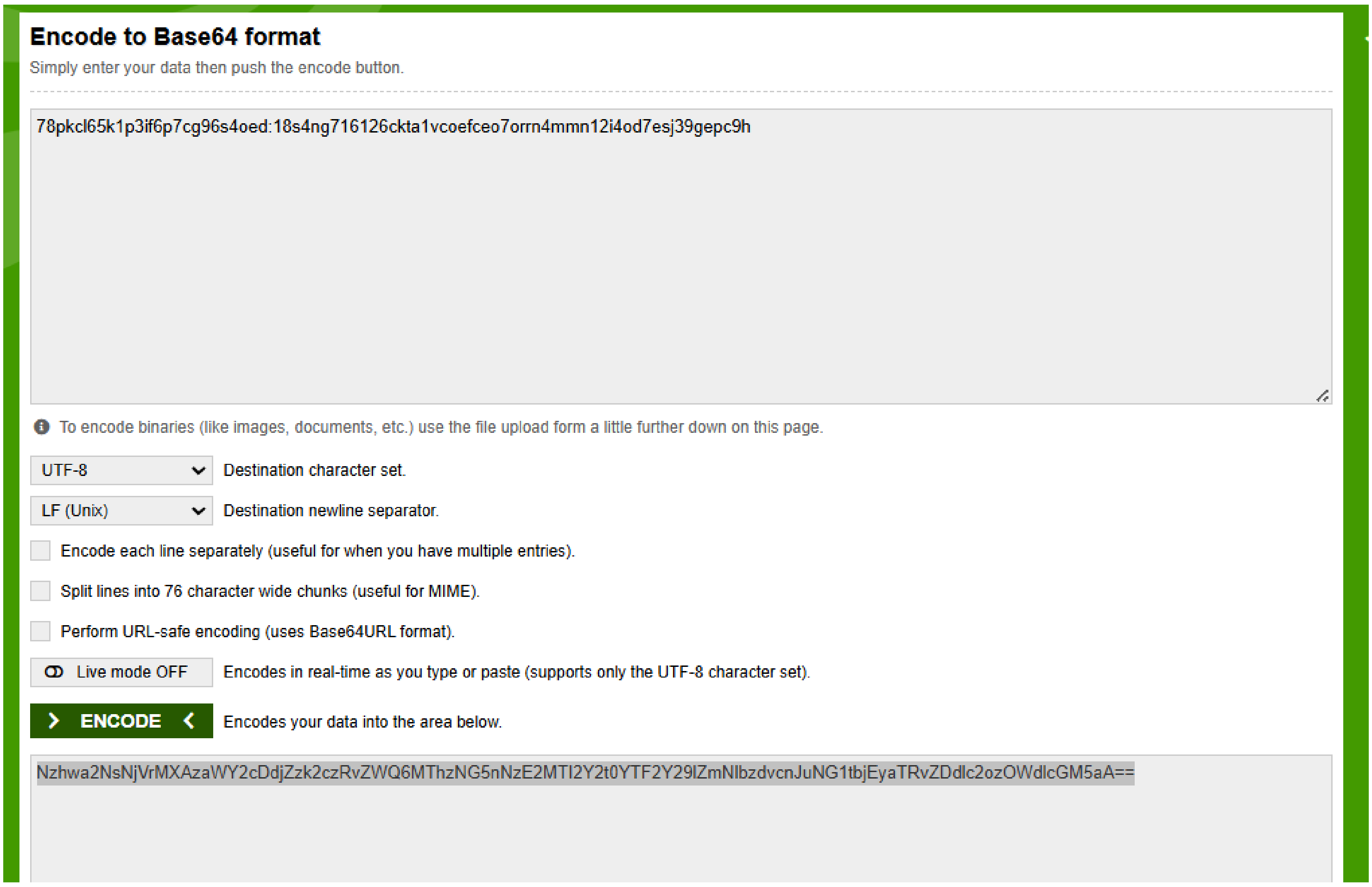
name@host.com

Next

New user? Create an account

Create user Login System using AWS Cognito and LAMP Stack

A Comprehensive Guide



Encode to Base64 format

Simply enter your data then push the encode button.

```
78pkcl65k1p3if6p7cg96s4oed:18s4ng716126ckta1vcoefceo7orrn4mmn12i4od7esj39gepc9h
```

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

☐ Encode each line separately (useful for when you have multiple entries).

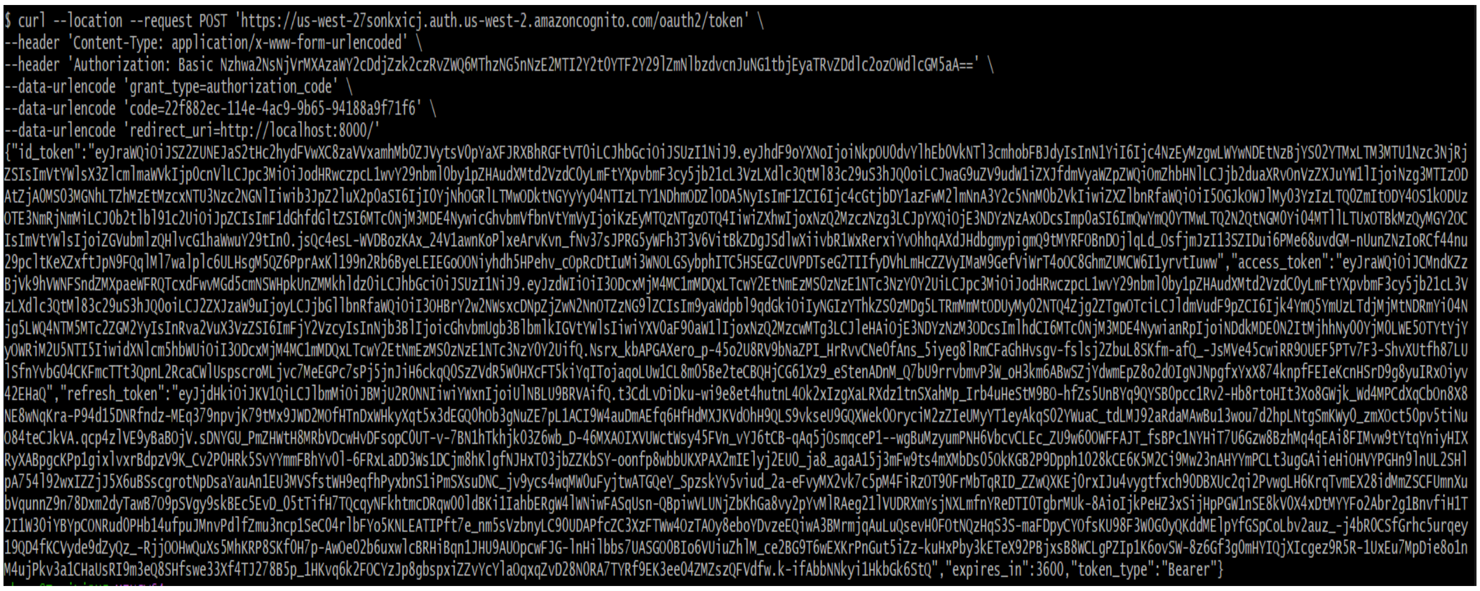
☐ Split lines into 76 character wide chunks (useful for MIME).

☐ Perform URL-safe encoding (uses Base64URL format).

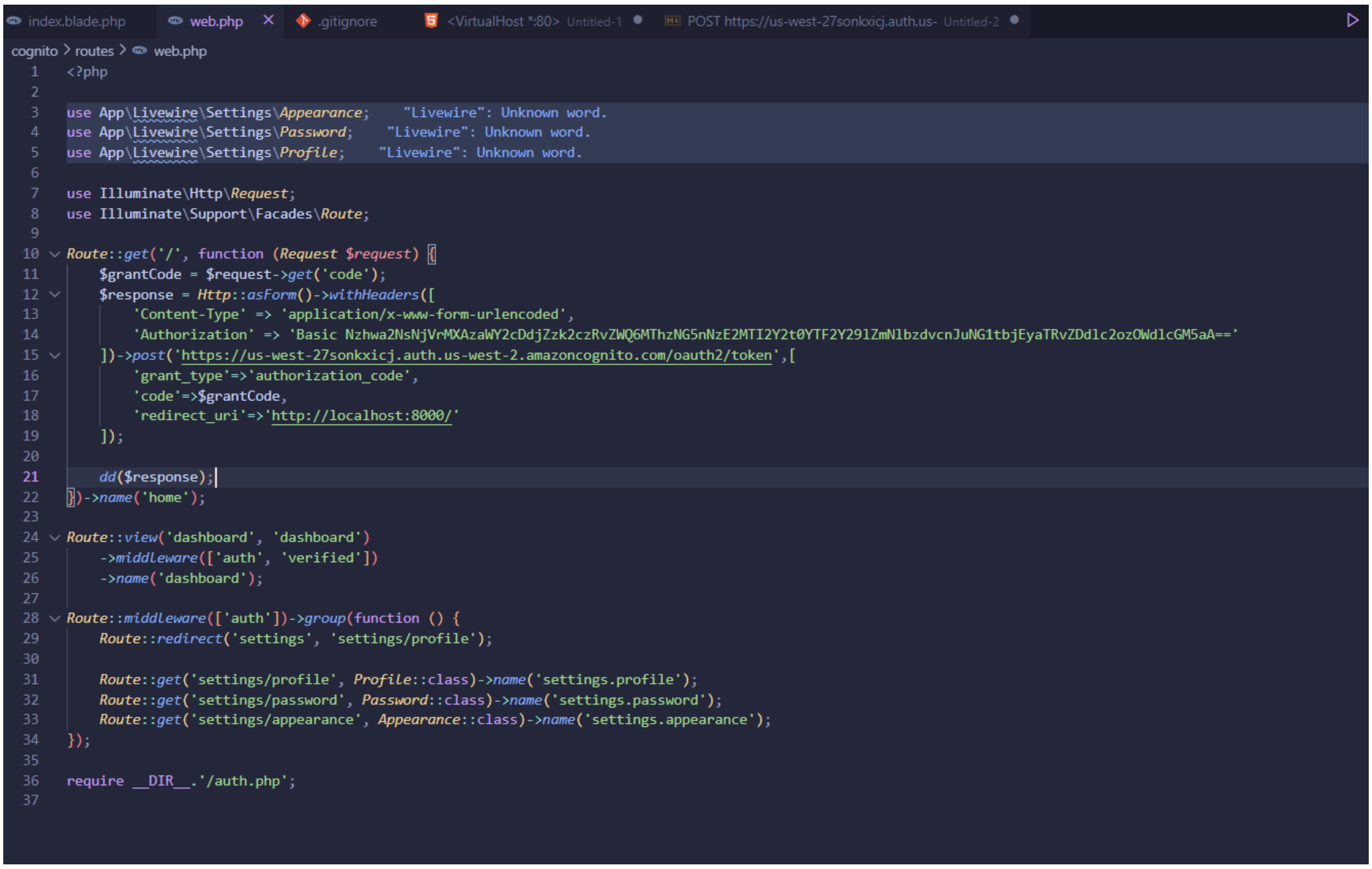
Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

Encodes your data into the area below.

```
Nzhwa2NsNjVrMXAzaWY2cDdjZzk2czRvZWQ6MThzNG5nNzE2MTI2Y2t0YTF2Y29lZmNlbzdvcnJuNG1tbjEyaTRvZDdlc2ozOWdlcGM5aA==
```



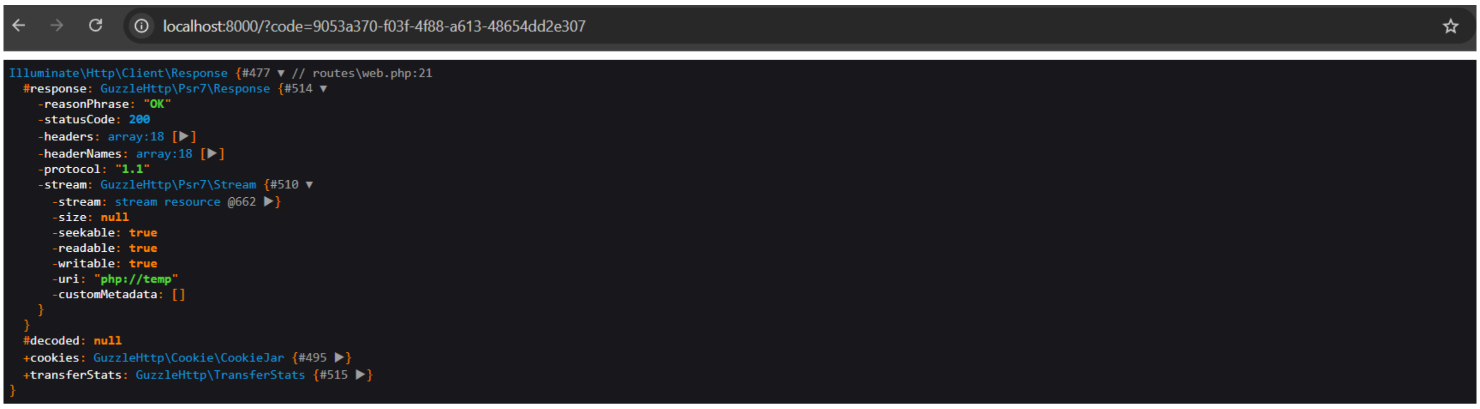
```
$ curl --location --request POST 'https://us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/oauth2/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Nzhwa2NsNjVrMXAzaWY2cDdjZzk2czRvZWQ6MThzNG5nNzE2MTI2Y2t0YTF2Y29lZmNlbzdvcnJuNG1tbjEyaTRvZDdlc2ozOWdlcGM5aA==' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'code=22f882ec-114e-4ac9-9b65-94188a9f71f6' \
--data-urlencode 'redirect_uri=http://localhost:8000/'
{"id_token":"eyJraWQiOiJSZ2ZUNEJaS2tHc2hydFVwXC8zaVVxamhMb0ZJVytsV0pYaXFJRXBhRGFtVT0iLCJhbGciOiJSUzI1NiJ9.eyJhdF9oYXNoIjoiNkp0U0dvYlhEb0VkNTl3cmhobFBJdyIsInN1YiI6Ijc4NzEyMzgwLWYwNDEtNzBjYS02YTMxLTM3MTU1Nzc3NjRj
ZSIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cy5jb21cL3VzLXdlc3QtMl83c29uS3hJQ0oiLCJwaG9uZV9udW1iZXJfdmVyaWZpZWQiOmZhbHNlLCJjb2duaXRvOnVzZXJuYW1lIjoiNzg3MTIzOD
AtZjA0MS03MGNhLTZhMzEtMzcxNTU3Nzc2NGNlIiwib3JpZ2luX2p0aSI6IjI0YjNhOGRlLTMwODktNGYyYy04NTIzLTY1NDhmODZlODA5NyIsImF1ZCI6Ijc4cGtjbDY1azFwM2lmNnA3Y2c5NnM0b2VkIiwiZXZlbnRfaWQiOiI5OGJkOWJlMy03YzIzLTQ0ZmItODY4OS1kODUz
OTE3NmRjNmMiLCJ0b2tlbl91c2UiOiJpZCIsImF1dGhfdGltZSI6MTc0NjM3MDE4NywicGhvbmVfbnVtYmVyIjoiKzEyMTQzNTgzOTQ4IiwiZXhwIjoxNzQ2MzczNzg3LCJpYXQiOjE3NDYzNzAxODcsImp0aSI6ImQwYmQ0YTMwLTQ2N2QtNGM0Yi04MTllLTUxOTBkMzQyMGY2OC
IsImVtYWlsIjoiZGVubmlzQHlvcG1haWwuY29tIn0.jsQc4esL-WVDBozKAx_24V1awnKoPlxeArvKvn_fNv37sJPRG5yWFh3T3V6VitBkZDgJSdlwXiivbR1WxRerxiYvOhhqAXdJHdbgmypigmQ9tMYRFOBnDOjlqLd_0sfjmJzI13SZIDui6PMe68uvdGM-nUunZNzIoRCf44nu
29pcltKeXZxftJpN9FQqlMl7walplc6ULHsgM5QZ6PprAxKl199n2Rb6ByeLEIEGoOONiyhdh5HPehv_cOpRcDtIuMi3WNOLGSybphITC5HSEGZcUVPDTseG2TIIfyDVhLmHcZZVyIMaM9GefViWrT4oOC8GhmZUMCW6I1yrvtIuww","access_token":"eyJraWQiOiJCMndKZz
BjVk9hVWNFSndZMXpaeWFRQTcxdFwvMGd5cmNSWHpkUnZMMkhldz0iLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI3ODcxMjM4MC1mMDQxLTcwY2EtNmEzMS0zNzE1NTc3NzY0Y2UiLCJpc3MiOiJodHRwczpcL1wvY29nbml0by1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cy5jb21cL3V
zLXdlc3QtMl83c29uS3hJQ0oiLCJ2ZXJzaW9uIjoyLCJjbGllbnRfaWQiOiI3OHBrY2w2NWsxcDNpZjZwN2NnOTZzNG9lZCIsIm9yaWdpbl9qdGkiOiIyNGIzYThkZS0zMDg5LTRmMmMtODUyMy02NTQ4Zjg2ZTgwOTciLCJldmVudF9pZCI6Ijk4YmQ5YmUzLTdjMjMtNDRmYi04N
jg5LWQ4NTM5MTc2ZGM2YyIsInRva2VuX3VzZSI6ImFjY2VzcyIsInNjb3BlIjoicGhvbmUgb3BlbmlkIGVtYWlsIiwiYXV0aF90aW1lIjoxNzQ2MzcwMTg3LCJleHAiOjE3NDYzNzM3ODcsImlhdCI6MTc0NjM3MDE4NywianRpIjoiNDdkMDE0N2ItMjhhNy00YjM0LWE5OTYtYjY
yOWRiM2U5NTI5IiwidXNlcm5hbWUiOiI3ODcxMjM4MC1mMDQxLTcwY2EtNmEzMS0zNzE1NTc3NzY0Y2UifQ.Nsrx_kbAPGAXero_p-45o2U8RV9bNaZPI_HrRvvCNe0fAns_5iyeg8lRmCFaGhHvsgv-fslsj2ZbuL8SKfm-afQ_-JsMVe45cwiRR9OUEF5PTv7F3-ShvXUtfh87LU
lSfnYvbG04CKFmcTTt3QpnL2RcaCWlUspscroMLjvc7MeEGPc7sPj5jnJiH6ckqQ0SzZVdR5WOHXcFT5kiYqITojaqoLUw1CL8m05Be2teCBQHjCG61Xz9_eStenADnM_Q7bU9rrvbmvP3W_oH3km6ABwSZjYdwmEpZ8o2dOIgNJNpgfxYxX874knpfFEIeKcnHSrD9g8yuIRxOiyv
42EHaQ","refresh_token":"eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAifQ.t3CdLvDiDku-wi9e8et4hutnL4Ok2xIzgXaLRXdz1tnSXahMp_Irb4uHeStM9BO-hfZs5UnBYq9QYSBOpcc1Rv2-Hb8rtoHIt3Xo8GWjk_Wd4MPCdXqCbOn8X8
NE8wNqKra-P94d15DNRfndz-MEq379npvjK79tMx9JWD2MOfHTnDxWHkyXqt5x3dEGQ0hOb3gNuZE7pL1ACI9W4auDmAEfq6HfHdMXJKVdOhH9QLS9vkseU9GQXWekOOryciM2zZIeUMyYT1eyAkqS02YWuaC_tdLMJ92aRdaMAwBu13wou7d2hpLNtgSmKWy0_zmXOct5Opv5tiNu
O84teCJkVA.qcp4zlVE9yBaBOjV.sDNYGU_PmZHWtH8MRbVDcwHvDFsopCOUT-v-7BN1hTkhjk03Z6wb_D-46MXAOIXVUWctWsy45FVn_vYJ6tCB-qAq5jOsmqceP1--wgBuMzyumPNH6VbcvCLEc_ZU9w6OOWFFAJT_fsBPc1NYHiT7U6Gzw8BzhMq4qEAi8FIMvw9tYtqYniyHIX
RyXABpgcKPp1gixlvxrBdpzV9K_Cv2POHRk5SvYYmmFBhYvOl-6FRxLaDD3Ws1DCjm8hKlgfNJHxTO3jbZZKbSY-oonfp8wbbUKXPAX2mIElyj2EUO_ja8_agaA15j3mFw9ts4mXMbDs05OkKGB2P9Dpph1028kCE6K5M2Ci9Mw23nAHYYmPCLt3ugGAiieHiOHVYPGHn9lnUL2SHl
pA754l92wxIZZjJ5X6uBSscgrotNpDsaYauAn1EU3MVSfstWH9eqfhPyxbnS1iPmSXsuDNC_jv9ycs4wqMWOuFyjtwATGQeY_SpzskYv5viud_2a-eFvyMX2vk7c5pM4FiRzOT90FrMbTqRID_ZZwQXKEj0rxIJu4vygtfxch90DBXUc2qi2PvwgLH6KrqTvmEX28idMmZSCFUmnXu
bVqunnZ9n78Dxm2dyTawB7O9pSVgy9skBEc5EvD_05tTifH7TQcqyNFkhtmcDRqw00ldBKi1IahbERgW4lWNiwFASqUsn-QBpiwVLUNjZbKhGa8vy2pYvMlRAeg21lVUDRXmYsjNXLmfnYReDTI0TgbrMUk-8AioIjkPeHZ3xSijHpPGW1nSE8kVOX4xDtMYYFo2Abr2g1BnvfiH1T
2I1W30iYBYpCONRud0PHb14ufpuJMnvPdlfZmu3ncp1SeC04rlbFYo5KNLEATIPft7e_nm5sVzbnyLC90UDAPfcZC3XzFTWw4OzTAOy8eboYDvzeEQiwA3BMrmjqAuLuQsevHOFOtNQzHqS3S-maFDpyCYOfsKU98F3WOGOyQKddMElpYfGSpCoLbv2auz_-j4bROCSfGrhc5urqey
19QD4fKCVyde9dZyQz_-RjjOOHwQuXs5MhKRP8SKf0H7p-AwOe02b6uxwlcBRHiBqn1JHU9AUOpcwFJG-lnHilbbs7UASGOOBIo6VUiuZhlM_ce2BG9T6wEXKrPnGut5iZz-kuHxPby3kETeX92PBjxsB8WCLgPZIp1K6ovSW-8z6Gf3gOmHYIQjXIcgez9R5R-1UxEu7MpDie8o1n
M4ujPkv3a1CHaUsRI9m3eQ8SHfswe33Xf4TJ278B5p_1HKvq6k2FOCYzJp8gbspxiZZvYcYlaOqxqZvD28NORA7TYRf9EK3ee04ZMZszQFVdfw.k-ifAbbNNkyi1HkbGk6StQ","expires_in":3600,"token_type":"Bearer"}
```



index.blade.php | web.php | .gitignore | <VirtualHost *:80> Untitled-1 | POST https://us-west-27sonkxicj.auth.us- Untitled-2

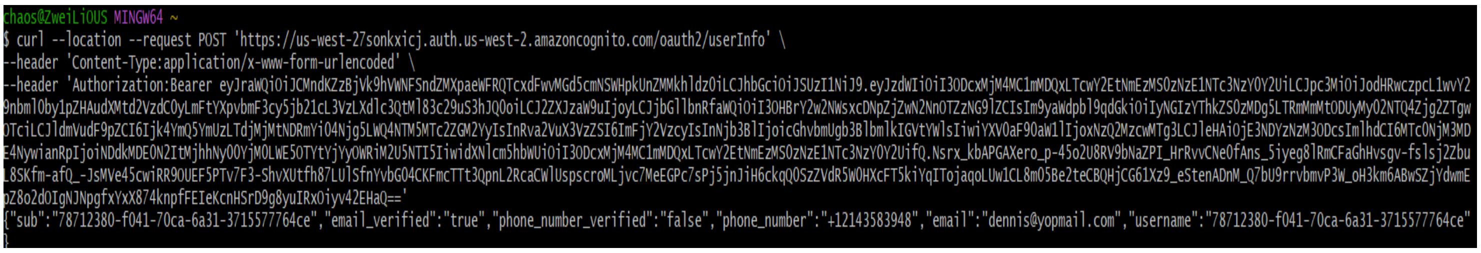
cognito > routes > web.php

```php
<?php

use App\Livewire\Settings\Appearance;    "Livewire": Unknown word.
use App\Livewire\Settings\Password;    "Livewire": Unknown word.
use App\Livewire\Settings\Profile;    "Livewire": Unknown word.

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::get('/', function (Request $request) {
    $grantCode = $request->get('code');
    $response = Http::asForm()->withHeaders([
        'Content-Type' => 'application/x-www-form-urlencoded',
        'Authorization' => 'Basic Nzhwa2NsNjVrMXAzaWY2cDdjZzk2czRvZWQ6MThzNG5nNzE2MTI2Y2t0YTF2Y291ZmNlbzdvcnJuNG1tbjEyaTRvZDdlc2ozOWdlcGM5aA=='
    ])->post('https://us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/oauth2/token',[
        'grant_type'=>'authorization_code',
        'code'=>$grantCode,
        'redirect_uri'=>'http://localhost:8000/'
    ]);

    dd($response);
})->name('home');

Route::view('dashboard', 'dashboard')
    ->middleware(['auth', 'verified'])
    ->name('dashboard');

Route::middleware(['auth'])->group(function () {
    Route::redirect('settings', 'settings/profile');

    Route::get('settings/profile', Profile::class)->name('settings.profile');
    Route::get('settings/password', Password::class)->name('settings.password');
    Route::get('settings/appearance', Appearance::class)->name('settings.appearance');
});

require __DIR__.'/auth.php';
```

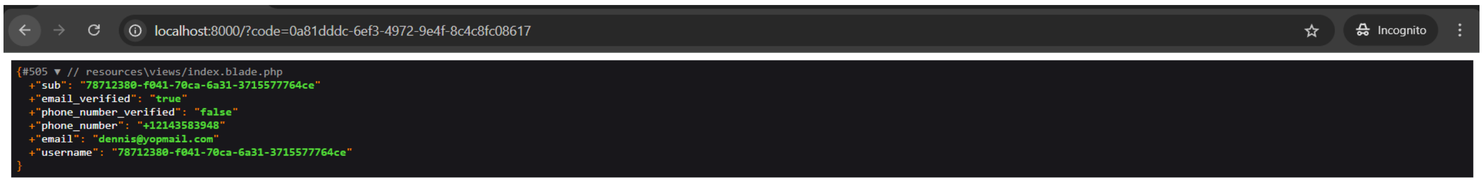


localhost:8000/?code=9053a370-f03f-4f88-a613-48654dd2e307

```
Illuminate\Http\Client\Response {#477 ▼ // routes\web.php:21
  #response: GuzzleHttp\Psr7\Response {#514 ▼
    -reasonPhrase: "OK"
    -statusCode: 200
    -headers: array:18 [▶]
    -headerNames: array:18 [▶]
    -protocol: "1.1"
    -stream: GuzzleHttp\Psr7\Stream {#510 ▼
      -stream: stream resource @662 ▶}
      -size: null
      -seekable: true
      -readable: true
      -writable: true
      -uri: "php://temp"
      -customMetadata: []
    }
  }
  #decoded: null
  +cookies: GuzzleHttp\Cookie\CookieJar {#495 ▶}
  +transferStats: GuzzleHttp\TransferStats {#515 ▶}
}
```

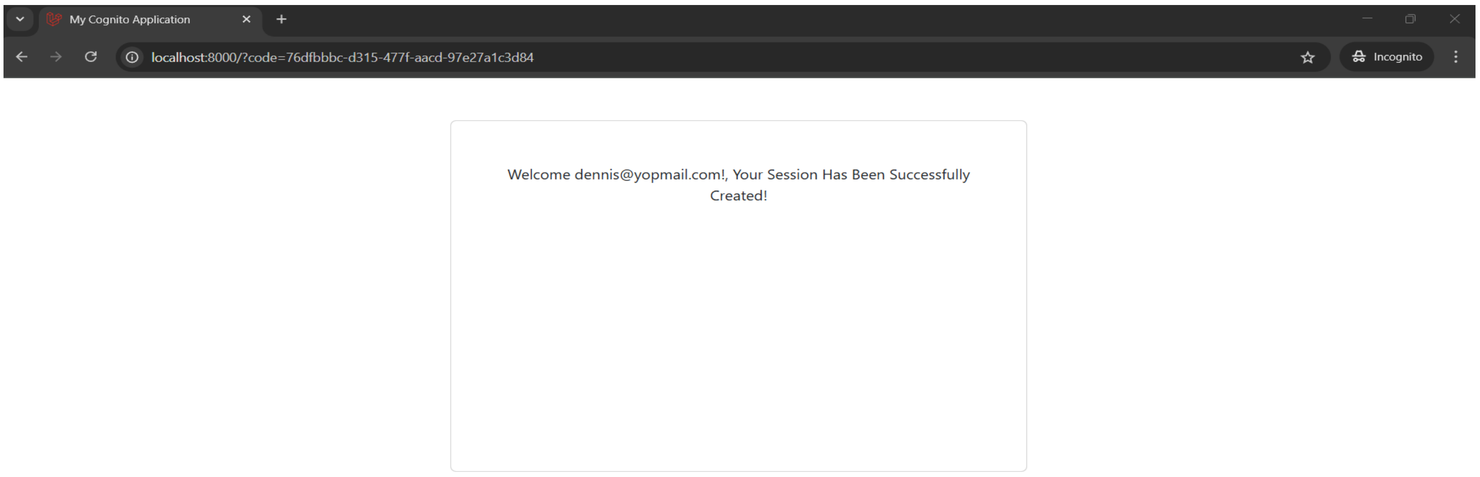


```
chaos@ZweiLiOUS MINGW64 ~
$ curl --location --request POST 'https://us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/oauth2/userInfo' \
--header 'Content-Type:application/x-www-form-urlencoded' \
--header 'Authorization:Bearer eyJraWQiOiJCMndKZzBjVk9hVWNFSndZMXpaeWFRQTcxdFwvMGd5cmNSWHpkUnZMMkhldz0iLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI3ODcxMjM4MC1mMDQxLTcwY2EtNmEzMS0zNzE1NTc3NzY0Y2UiLCJpc3MiOiJodHRwczpcL1wvY2
9nbml0by1pZHAudXMtd2VzdC0yLmFtYXpvbmF3cy5jb21cL3VzLXdlc3QtMl83c29uS3hJQ0oiLCJ2ZXJzaW9uIjoyLCJjbGllbnRfaWQiOiI3OHBrY2w2NWsxcDNpZjZwN2NnOTZzNG9lZCIsIm9yaWdpbl9qdGkiOiIyNGIzYThkZS0zMDg5LTRmMmMtODUyMy02NTQ4Zjg2ZTgw
0TciLCJldmVudF9pZCI6Ijk4YmQ5YmUzLTdjMjMtNDRmYi04Njg5LWQ4NTM5MTc2ZGM2YyIsInRva2VuX3VzZSI6ImFjY2VzcyIsInNjb3BlIjoicGhvbmUgb3BlbmlkIGVtYWlsIiwiYXV0aF90aW1lIjoxNzQ2MzcwMTg3LCJleHAiOjE3NDYzNzM3ODcsImlhdCI6MTc0NjM3MD
E4NywianRpIjoiNDdkMDE0N2ItMjhhNy00YjM0LWE5OTYtYjYyOWRiM2U5NTI5IiwidXNlcm5hbWUiOiI3ODcxMjM4MC1mMDQxLTcwY2EtNmEzMS0zNzE1NTc3NzY0Y2UifQ.Nsrx_kbAPGAXero_p-45o2U8RV9bNaZPI_HrRvvCNe0fAns_5iyeg8lRmCFaGhHvsgv-fslsj2Zbu
L8SKfm-afQ_-JsMVe45cwiRR9OUEF5PTv7F3-ShvXUtfh87LUlSfnYvbG04CKFmcTTt3QpnL2RcaCWlUspscroMLjvc7MeEGPc7sPj5jnJiH6ckqQ0SzZVdR5W0HXcFT5kiYqITojaqoLUw1CL8m05Be2teCBQHjCG61Xz9_eStenADnM_Q7bU9rrvbmvP3W_oH3km6ABwSZjYdwmE
pZ8o2dOIgNJNpgfxYxX874knpfFEIeKcnHSrD9g8yuIRxOiyv42EHaQ=='
{"sub":"78712380-f041-70ca-6a31-371557764ce","email_verified":"true","phone_number_verified":"false","phone_number":"+12143583948","email":"dennis@yopmail.com","username":"78712380-f041-70ca-6a31-371557764ce"
}
```



localhost:8000/?code=0a81dddc-6ef3-4972-9e4f-8c4c8fc08617

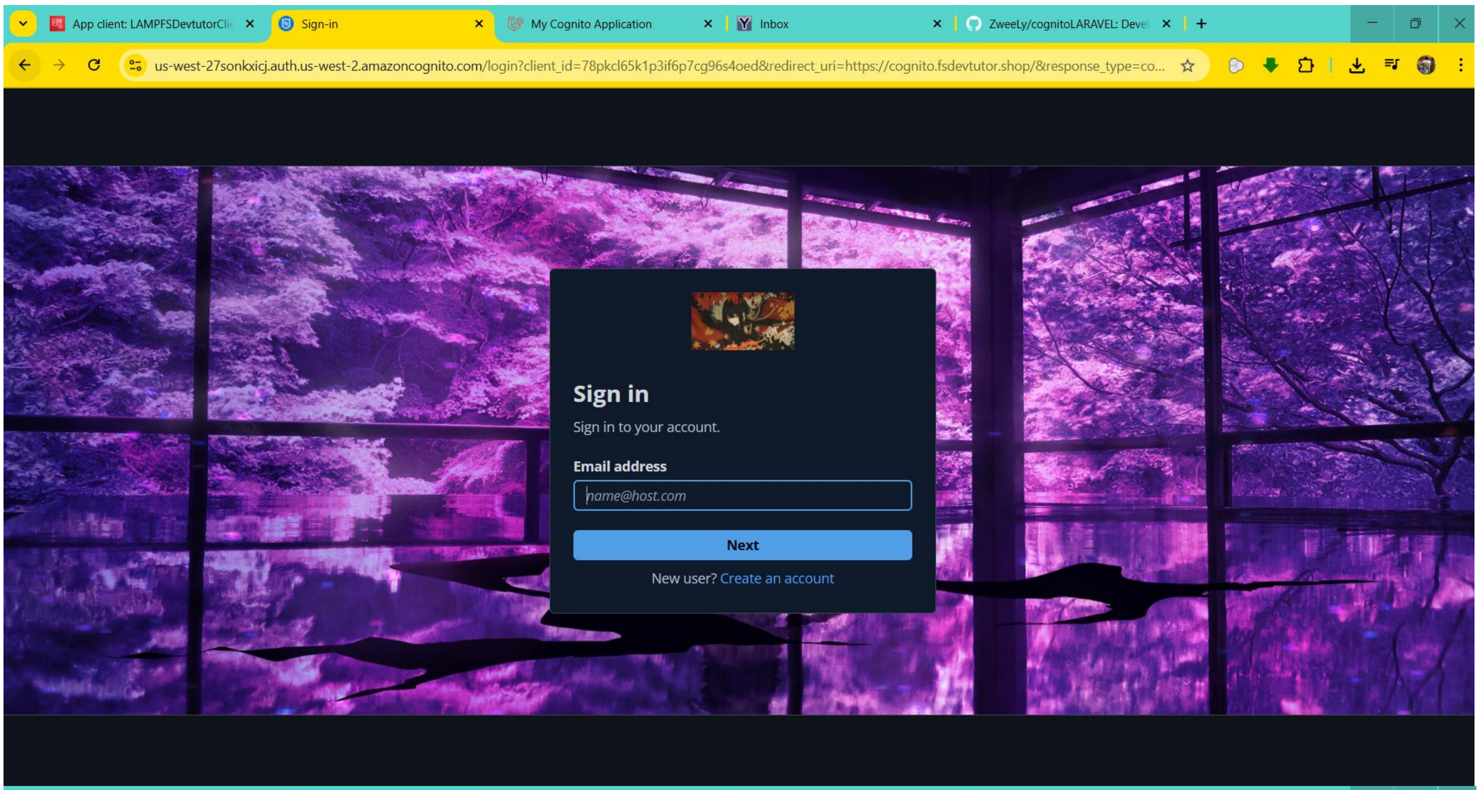
```
{#505 ▼ // resources\views/index.blade.php
  +"sub": "78712380-f041-70ca-6a31-3715577764ce"
  +"email_verified": "true"
  +"phone_number_verified": "false"
  +"phone_number": "+12143583948"
  +"email": "dennis@yopmail.com"
  +"username": "78712380-f041-70ca-6a31-3715577764ce"
}
```



localhost:8000/?code=76dfbbbc-d315-477f-aacd-97e27a1c3d84

Welcome dennis@yopmail.com!, Your Session Has Been Successfully Created!

Final Outputs -



App client: LAMPFSDevtutorCli
Sign-in
My Cognito Application
Inbox
ZweeLy/cognitoLARAVEL: Deve
us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/login?client_id=78pkcl65k1p3if6p7cg96s4oed&redirect_uri=https://cognito.fsdevtutor.shop/&response_type=co...

Sign in

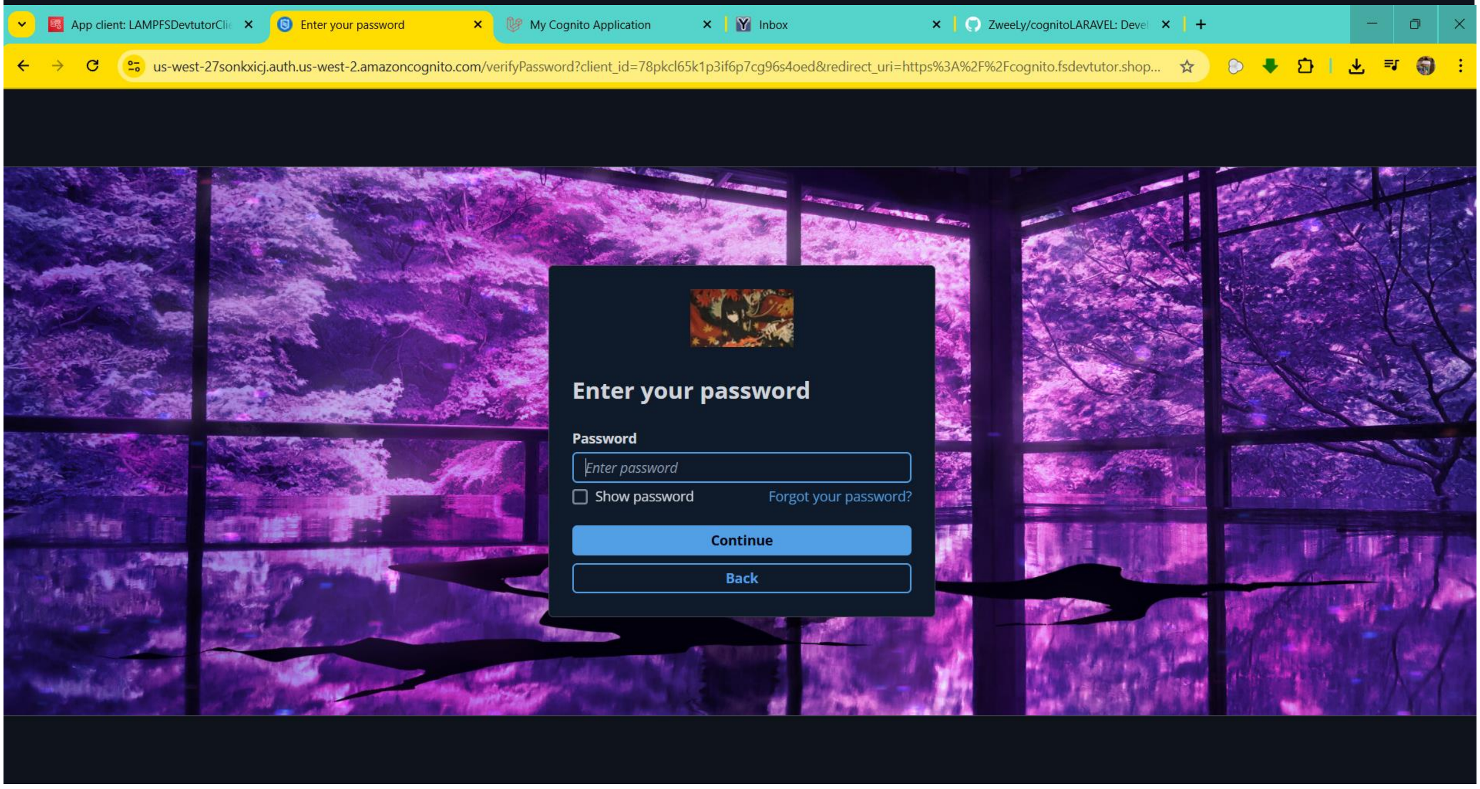
Sign in to your account.

Email address

name@host.com

Next

New user? Create an account



App client: LAMPFSDevtutorCli
Enter your password
My Cognito Application
Inbox
ZweeLy/cognitoLARAVEL: Deve
us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/verifyPassword?client_id=78pkcl65k1p3if6p7cg96s4oed&redirect_uri=https%3A%2F%2Fcognito.fsdevtutor.shop...

Enter your password

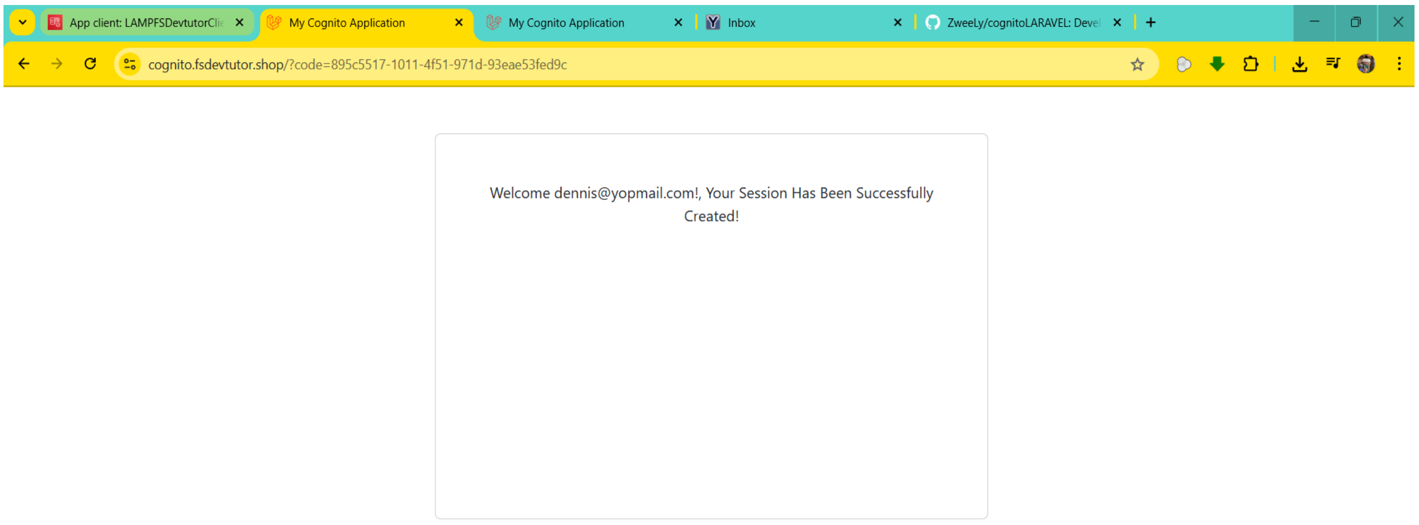
Password

Enter password

Show password　Forgot your password?

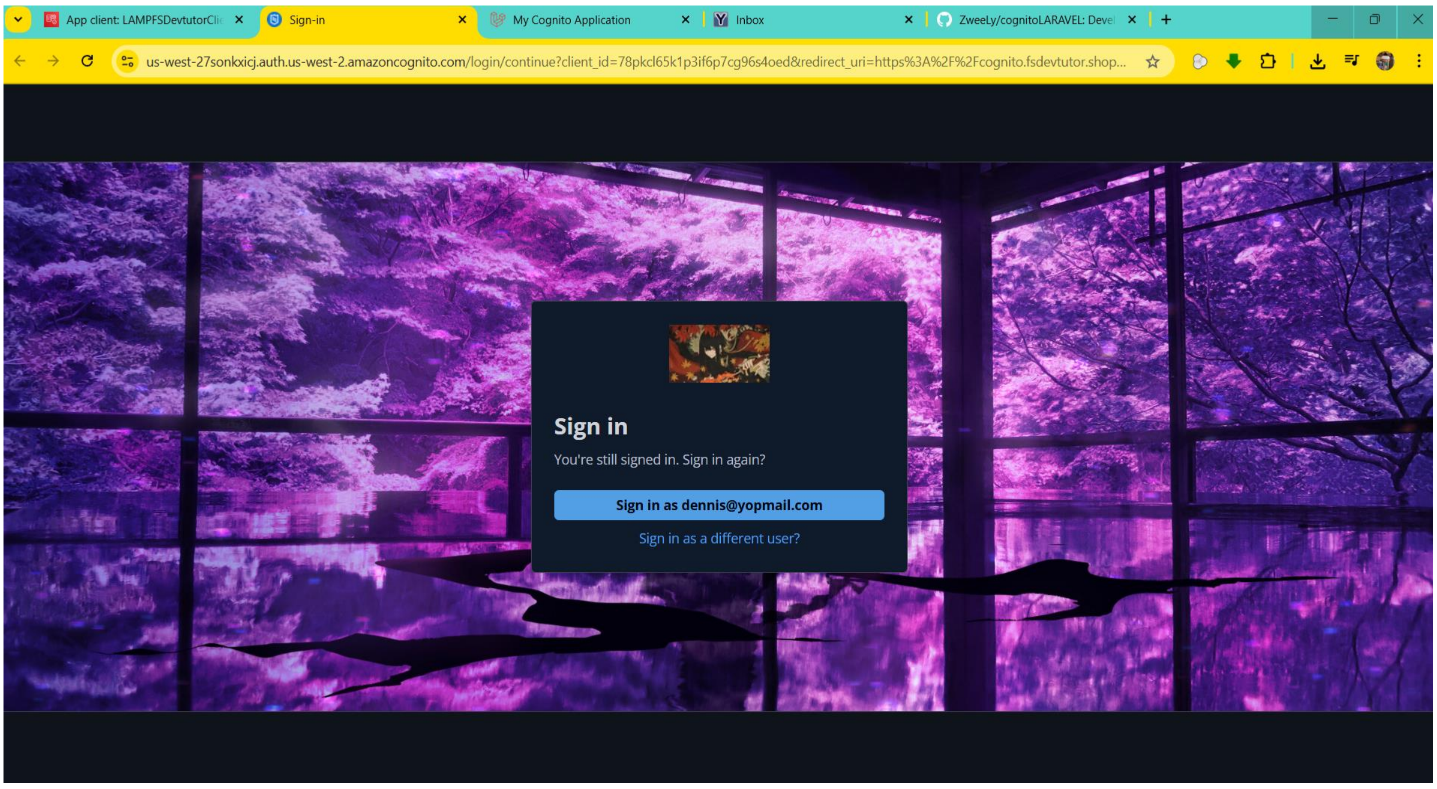
Continue

Back



App client: LAMPFSDevtutorCli
My Cognito Application
My Cognito Application
Inbox
Zweely/cognitoLARAVEL: Devel
cognito.fsdevtutor.shop/?code=895c5517-1011-4f51-971d-93eae53fed9c

Welcome dennis@yopmail.com!, Your Session Has Been Successfully Created!



App client: LAMPFSDevtutorCli
Sign-in
My Cognito Application
Inbox
Zweely/cognitoLARAVEL: Devel
us-west-27sonkxicj.auth.us-west-2.amazoncognito.com/login/continue?client_id=78pkcl65k1p3if6p7cg96s4oed&redirect_uri=https%3A%2F%2Fcognito.fsdevtutor.shop...

Sign in

You're still signed in. Sign in again?

Sign in as dennis@yopmail.com
Sign in as a different user?

Conclusion -

In conclusion, the project on implementing a secure user login system using AWS Cognito and the LAMP stack demonstrates a practical and efficient approach to integrating modern authentication services into traditional web applications. The use of AWS Cognito enabled robust user management, secure authorization workflows, and seamless integration with web applications through OAuth 2.0 protocols. Combining this with a reliable LAMP (Linux, Apache, MySQL, PHP) environment facilitated a cost-effective and scalable deployment infrastructure.
Throughout the project, it became clear that AWS Cognito greatly simplifies the process of building secure authentication flows while maintaining high standards of security, reliability, and user experience. The successful implementation showcases how legacy application stacks can be modernized by incorporating cloud-based identity services. This work not only provides a foundational system for user authentication but also sets the stage for integrating additional cloud-native services, ensuring future scalability and flexibility.

Concluding Points -

- Demonstrates the integration of cloud-based authentication with traditional web application stacks.
- Leverages AWS Cognito for secure user creation, sign-in, and token-based access control.
- Highlights OAuth 2.0-based workflows using Hosted UI and application callbacks.
- Implements the LAMP stack on an EC2 instance for reliable, scalable hosting.
- Utilizes DNS and SSL configuration for secure, domain-specific access.
- Emphasizes best practices in authentication, authorization, and user session management.
- Validates the system through successful user testing and API token flow verification.
- Provides a foundation for integrating additional AWS services such as Lambda, CloudWatch, or S3.
- Demonstrates cost-effective cloud adoption using open-source and pay-as-you-go services.
- Serves as a reference for developers combining legacy infrastructure with modern identity solutions.

More Affiliated Links –
Github Project link- https://github.com/ZweeLy/cognitoLARAVEL