

A thick dark blue vertical bar is positioned on the left side of the page. To its right, several thin, curved lines in shades of blue and grey sweep upwards from the bottom left corner towards the center of the page.

1/5/2025

PROJECT FILE

TOPIC – Leveraging Elastic Beanstalk
and Host Full Stack Application

SUBMITTED BY – SIYA SRIVASTAVA
E-mail ID – Siyasri19.14@gmail.com

ABSTRACT

This project explores the implementation of Amazon Web Services (AWS) Elastic Beanstalk to host a full-stack application, demonstrating its potential to streamline deployment processes, enhance scalability, and optimize resource management. Elastic Beanstalk, a Platform-as-a-Service (PaaS) solution, abstracts the complexity of managing infrastructure, enabling developers to focus on building and enhancing application functionality. This study highlights the capabilities of Elastic Beanstalk to support diverse web technologies and frameworks while maintaining high reliability and performance.

The project begins with an overview of Elastic Beanstalk's architecture, emphasizing its integration with various AWS services like EC2, RDS, and S3, which collectively form a robust and scalable environment for hosting applications. The full-stack application developed for this project leverages modern frameworks to create a seamless integration between front-end and back-end components, showcasing Elastic Beanstalk's flexibility in accommodating complex deployments.

Through practical implementation, this project evaluates key features such as automated provisioning, load balancing, and scaling, which are integral to managing traffic surges and ensuring consistent application performance. Elastic Beanstalk's monitoring and logging tools were also utilized to track application health, enabling proactive issue resolution and performance optimization. Additionally, the project examines the cost-efficiency of Elastic Beanstalk, emphasizing its ability to dynamically allocate resources based on real-time demand, thereby reducing operational overhead.

The findings of this project underscore the strategic advantages of adopting Elastic Beanstalk for full-stack application hosting. By simplifying deployment workflows and ensuring scalability, Elastic Beanstalk empowers development teams to accelerate time-to-market and focus on innovation. Furthermore, its seamless integration with the AWS ecosystem enhances the application's scalability, security, and resilience.

In conclusion, leveraging AWS Elastic Beanstalk for full-stack application hosting represents a transformative approach to cloud-native application management. This project highlights its role as a powerful, user-friendly platform that equips organizations with the tools needed to deliver high-performing applications while maintaining operational efficiency and cost-effectiveness.

OBJECTIVE

“The objective of this project file is to document the comprehensive process of leveraging AWS Elastic Beanstalk for deploying and hosting a scalable full-stack application. It aims to demonstrate the practical implementation of cloud-based deployment, automated scaling, and efficient resource management. By detailing the configuration of Elastic Beanstalk, integration with Amazon RDS, and hosting a LAMP application, this project serves as a guide to understanding cloud infrastructure's role in enhancing application performance, scalability, and reliability in real-world scenarios.”

Below are the sub-objectives that outline the tasks performed during the making of this project:

Setting up Elastic Beanstalk as the Deployment Foundation

The initial focus was on configuring Elastic Beanstalk to serve as a managed hosting environment. This included leveraging its features for automating critical processes like scaling, deployment, and health monitoring to ensure a robust and efficient foundation for the full-stack application.

Configuring a Scalable Application Environment

A single-instance environment was created with auto-scaling capabilities, enabling the application to handle fluctuating workloads seamlessly. This involved configuring triggers for scaling up to two additional instances to optimize resource utilization and ensure consistent application performance under varying traffic conditions.

Hosting the Database with Amazon RDS

An Amazon RDS instance was launched to host the backend database of the application. This task focused on selecting the appropriate database engine, ensuring data security, and providing a scalable and reliable storage solution to support application functionality.

Developing and Integrating a LAMP Application

A dynamic LAMP (Linux, Apache, MySQL, PHP) application was developed and integrated with the RDS instance. This stage involved backend development, database connectivity, and thorough testing to ensure seamless communication and functionality.

Deploying the Application on Elastic Beanstalk

The LAMP application was deployed on the Elastic Beanstalk platform, utilizing its capabilities for efficient scaling, security, and performance optimization.

Enhancing Accessibility with Custom Domain Configuration

Finally, the project included an optional step to configure a custom domain, improving the application's accessibility and professional appearance by mapping the domain to the Elastic Beanstalk environment.

The primary objective of this project is to explore and document the deployment of a full-stack application utilizing AWS Elastic Beanstalk, a powerful managed platform designed to simplify the complexities of deploying, managing, and scaling web applications. By leveraging Elastic Beanstalk, the project seeks to provide a comprehensive understanding of how this Platform-as-a-Service

(PaaS) solution bridges the gap between development and deployment. This is achieved by automating key processes such as resource provisioning, load balancing, and application monitoring.

The project emphasizes the adoption of best practices for creating a hosting solution that is not only scalable and reliable but also cost-efficient. It delves into the capabilities of Elastic Beanstalk to support dynamic workloads, enhance performance, and ensure seamless integration with various AWS services, providing a robust framework for modern application deployment.

.

INTRODUCTION

The dynamic nature of modern web development demands robust, scalable, and efficient hosting solutions to ensure seamless deployment and management of full-stack applications. This project, titled *"Leveraging Elastic Beanstalk and Hosting Full Stack Application,"* demonstrates the use of AWS Elastic Beanstalk to build, deploy, and manage a scalable and high-performing full-stack application with minimal complexity.

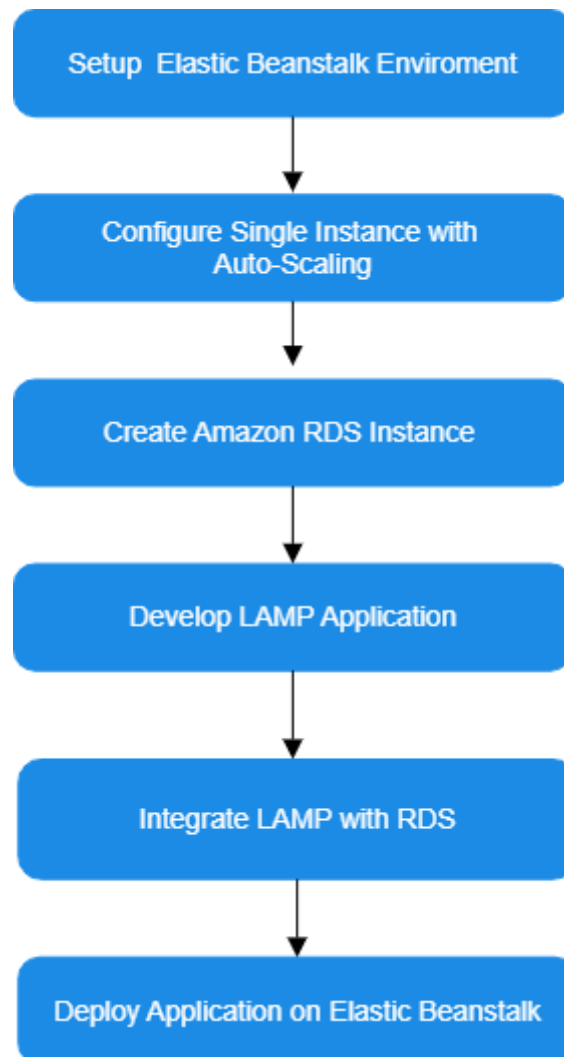
Elastic Beanstalk is a powerful Platform-as-a-Service (PaaS) offering from AWS that simplifies the deployment process by automating resource provisioning, load balancing, and scaling. It abstracts the complexities of infrastructure management, allowing developers to focus on coding and application innovation. In this project, Elastic Beanstalk has been utilized as the primary hosting platform to create an efficient deployment workflow for a full-stack application comprising a LAMP (Linux, Apache, MySQL, PHP) stack.

The workflow begins with setting up the Elastic Beanstalk environment and configuring a single-instance setup capable of auto-scaling to handle varying traffic loads. The next step involves creating a reliable backend by launching an Amazon RDS (Relational Database Service) instance to host the application database securely and efficiently.

Once the backend is established, a dynamic LAMP application is developed and integrated with the RDS database to enable data-driven functionality. This application is then deployed on the Elastic Beanstalk environment, leveraging its features to ensure seamless scaling, robust monitoring, and high availability. As an optional enhancement, a custom domain is configured, complete with SSL certificates for secure access, ensuring a professional user experience.

This project showcases the end-to-end workflow of hosting a full-stack application, highlighting the advantages of leveraging Elastic Beanstalk for automation, scalability, and ease of management. By documenting this process, the project provides valuable insights for developers and organizations seeking efficient cloud-based deployment solutions.

Workflow For 'Leveraging Elastic Beanstalk and Hosting Full Stack Application



Methodology

Design and Architecture

The project adopts a **Microservices-based architecture** to ensure scalability and maintainability. It separates the front-end and back-end components for flexibility in development and deployment. The **front-end** is built with **React.js**, offering a responsive and dynamic user interface. The **back-end** is powered by **Node.js** and **Express**, enabling efficient handling of API requests and server-side logic. Data storage is managed using **MongoDB**, providing a flexible NoSQL database solution. **AWS Elastic Beanstalk** serves as the deployment platform, automatically managing the scaling, load balancing, and infrastructure configuration.

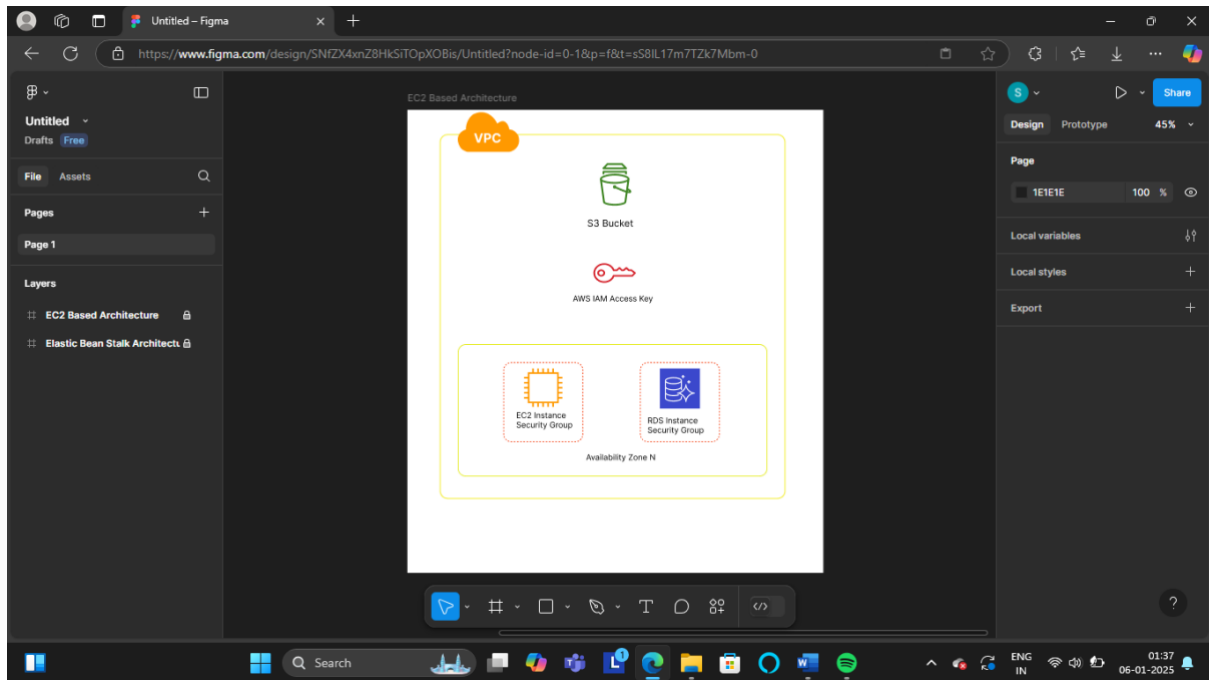
Design Elements:

- **Responsive UI/UX**: Ensures an optimal experience across devices, enhancing user engagement.
- **API-driven architecture**: Facilitates smooth interaction between the front-end and back-end components.
- **JWT Authentication**: Secures user data and ensures safe session management.
- **Scalability**: Elastic Beanstalk's automatic scaling ensures seamless adaptation to varying traffic loads.
- **Real-time Monitoring**: Integrated with **AWS CloudWatch** to monitor application health and performance.

Tools and Languages Used:

- **Languages**: JavaScript (React.js, Node.js)
- **Database**: MongoDB
- **Deployment**: AWS Elastic Beanstalk, AWS RDS, AWS S3
- **Authentication**: JWT
- **Monitoring**: AWS CloudWatch
- **Version Control**: Git, GitHub

UNDERSTANDING THE SIMPLE ARCHITECTURE FOR EC2 INSTANCE



This simple architecture represents an Amazon Web Services (AWS) setup for an EC2-based application. The key components are:

1. **VPC (Virtual Private Cloud)**: The foundational layer for isolating the network, providing control over the environment's IP range, subnets, and routing configurations.
2. **S3 Bucket**: A storage solution for storing and retrieving data, such as logs, backups, or application files, with high durability and scalability.
3. **IAM Access Key**: Ensures secure access to AWS resources (like S3) by enabling role-based permissions and authentication for the EC2 instances or users.
4. **Security Groups**:
 - **EC2 Instance Security Group**: Acts as a virtual firewall, controlling inbound and outbound traffic for the EC2 instance.

- **RDS Instance Security Group**: Protects the RDS database instance, restricting access based on defined rules.
- 5. **Availability Zone**: The resources are located within a specific AWS Availability Zone (a physically separate location), ensuring low-latency connectivity and reliability.

This EC2-based architecture contributes significantly to this project by providing the foundational components for a seamless deployment and hosting environment.

Elastic Beanstalk automates the deployment, scaling, and management of full-stack applications while utilizing key AWS resources like EC2, S3, and RDS. In this architecture:

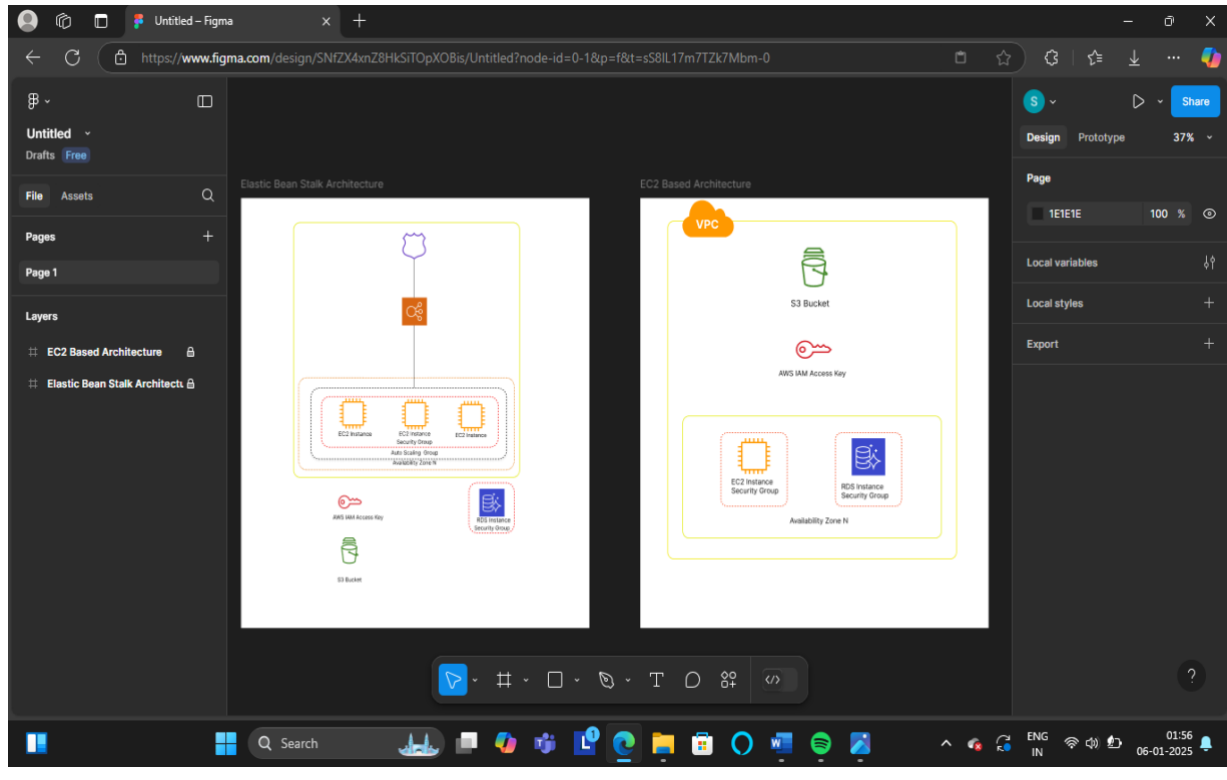
1. EC2 Instances act as the compute environment to run the application backend or frontend.
2. S3 Buckets store static assets, application files, or deployment artifacts, ensuring easy integration with Beanstalk.
3. RDS provides a robust database layer for application data, managed efficiently by Beanstalk.
4. Security Groups ensure controlled, secure network access to EC2 and RDS instances.
5. IAM Roles enable Beanstalk to access resources securely.

This architecture streamlines the full-stack hosting process, enhancing scalability, security, and ease of management while leveraging Elastic Beanstalk's automation capabilities.

Perks:

- **Scalability**: S3 and EC2 provide seamless scalability for storage and compute power.
- **Security**: IAM roles and security groups offer robust access control and network protection.
- **Cost Efficiency**: Pay-as-you-go model for storage and compute resources.
- **Reliability**: Hosting in an AWS Availability Zone enhances uptime and data durability.

Comparison Between ELB and EC2 Based Provisions



Here's a table comparing Elastic Beanstalk (EB) and EC2-based provisions:

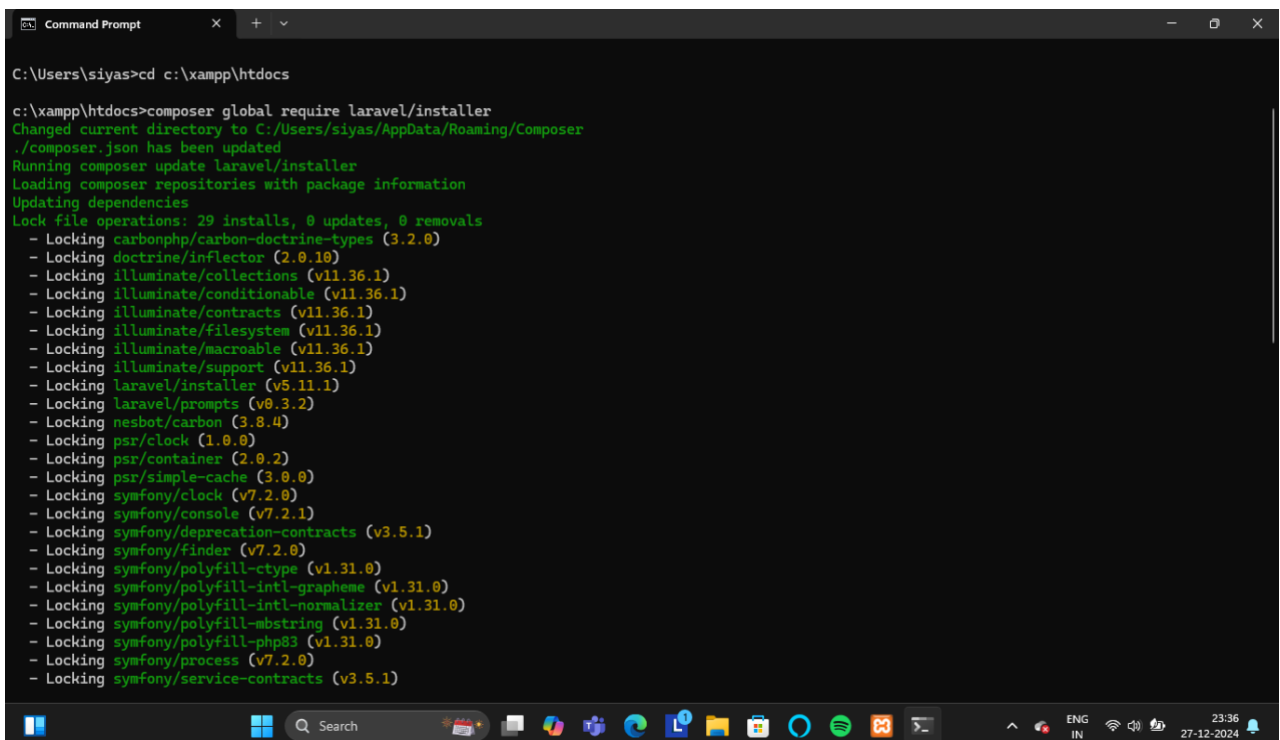
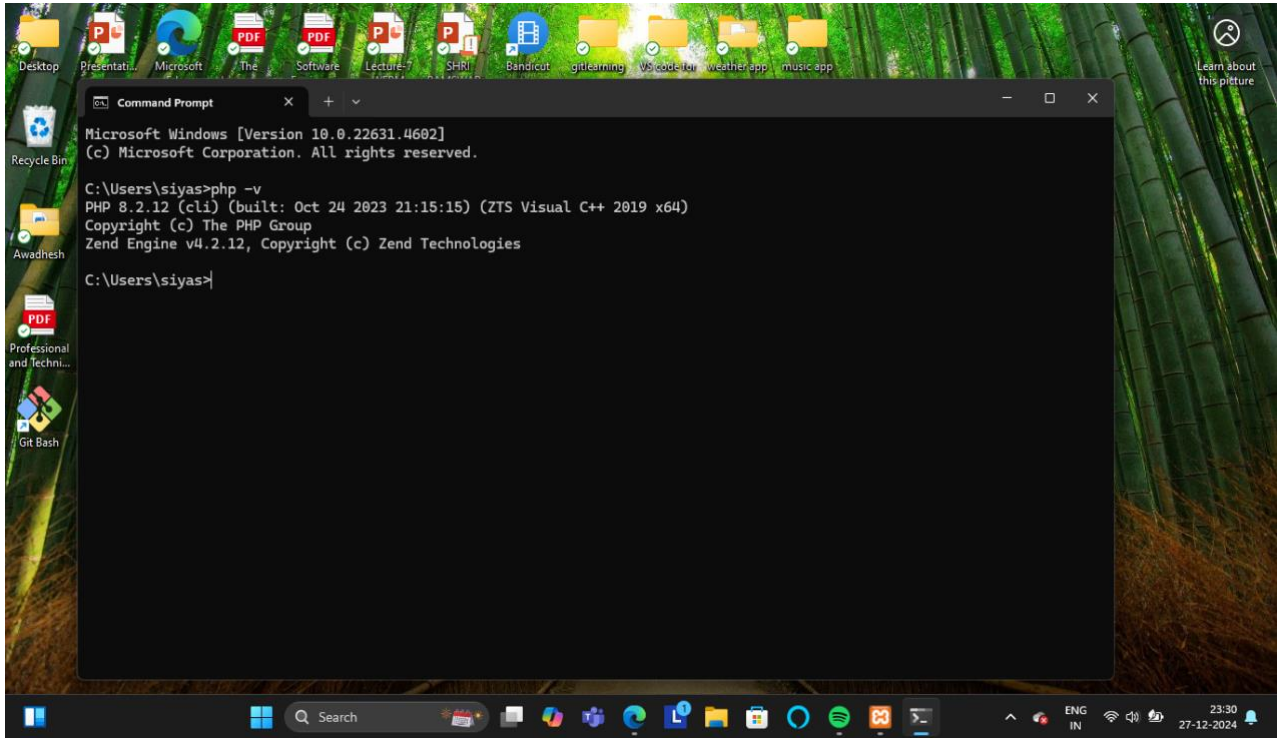
Aspect	Elastic Beanstalk	EC2-Based Provision
Ease of Setup	Simplifies deployment with automated provisioning.	Requires manual configuration of instances, S3, etc.
Scalability	Auto-scaling is built-in and managed automatically.	Needs manual setup of auto-scaling groups.
Management	Fully managed service; abstracts underlying infrastructure.	Full control, but requires manual management of resources.

Customization	Limited flexibility in configuring infrastructure.	Highly customizable for specific needs.
Monitoring	Integrated with CloudWatch and minimal setup needed.	Requires separate configuration for monitoring.
Security	Provides security groups and IAM roles automatically.	Security settings need to be configured manually.
Cost Efficiency	Cost-efficient for rapidly changing workloads.	May incur additional costs due to manual scaling.
Use Case	Ideal for rapid deployment of web apps.	Suitable for fine-grained control over resources.
Learning Curve	Beginner-friendly with minimal AWS expertise required.	Requires in-depth knowledge of AWS services.
Resource Scaling	Automatically adjusts based on application demand.	Requires manual intervention or custom setup.

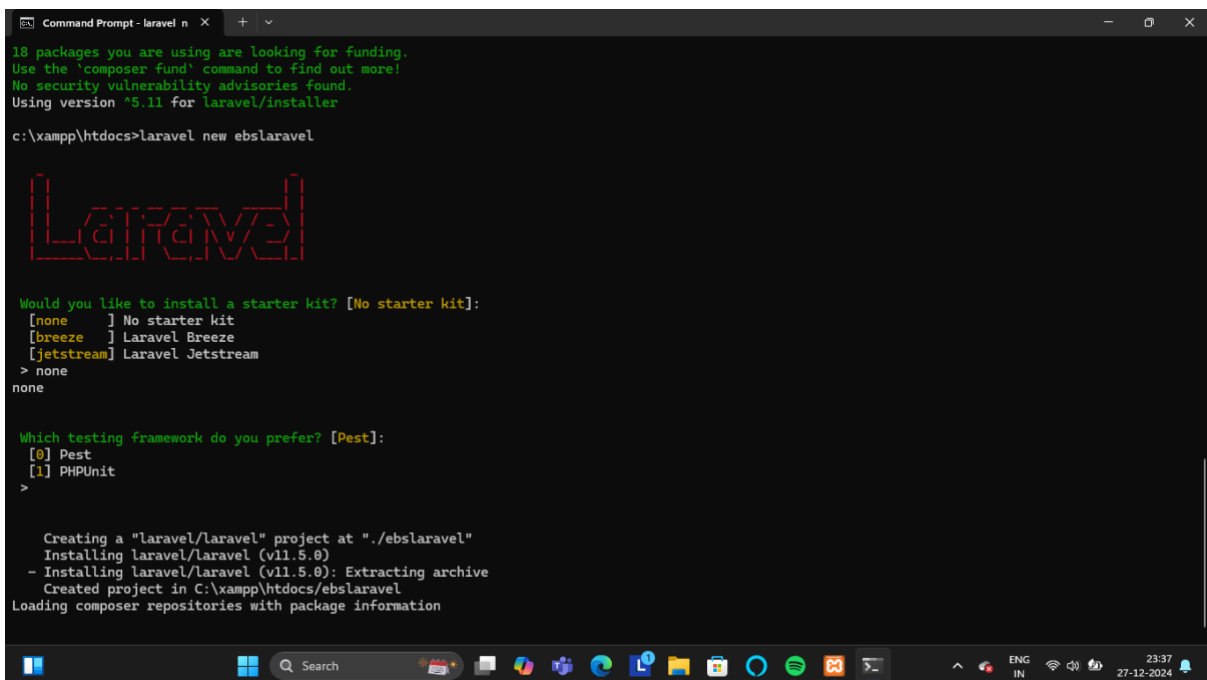
Conclusion:

Elastic Beanstalk is perfect for developers aiming for rapid web application deployment without needing deep AWS expertise. It abstracts infrastructure management, making it ideal for startups or teams focused on application development rather than operations. On the other hand, EC2-based provisioning suits advanced users who require granular control over their infrastructure. It allows customization of networking, scaling, and security, making it the preferred choice for complex, enterprise-grade applications that demand tailored solutions and precise resource management.

CODE



“The process shown below involves the creation of a new Laravel project named ebsLaravel using the Laravel installer. The user runs the command `laravel new ebsLaravel` in the terminal, initiating the setup process. During the setup, the Laravel installer prompts the user to select optional features, such as starter kits and testing frameworks. The user chooses not to include a starter kit (like Breeze or Jetstream) and selects Pest as the preferred testing framework. Laravel version 11.5.0 is downloaded and installed, and the necessary project files are extracted and set up in the specified directory (C:\xampp\htdocs\ebsLaravel). Additionally, Composer, the dependency manager for PHP, is used to download and configure all required packages for initializing the Laravel project.”



```
Command Prompt - laravel n
18 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
No security vulnerability advisories found.
Using version *5.11 for laravel/installer

c:\xampp\htdocs>laravel new ebslaravel

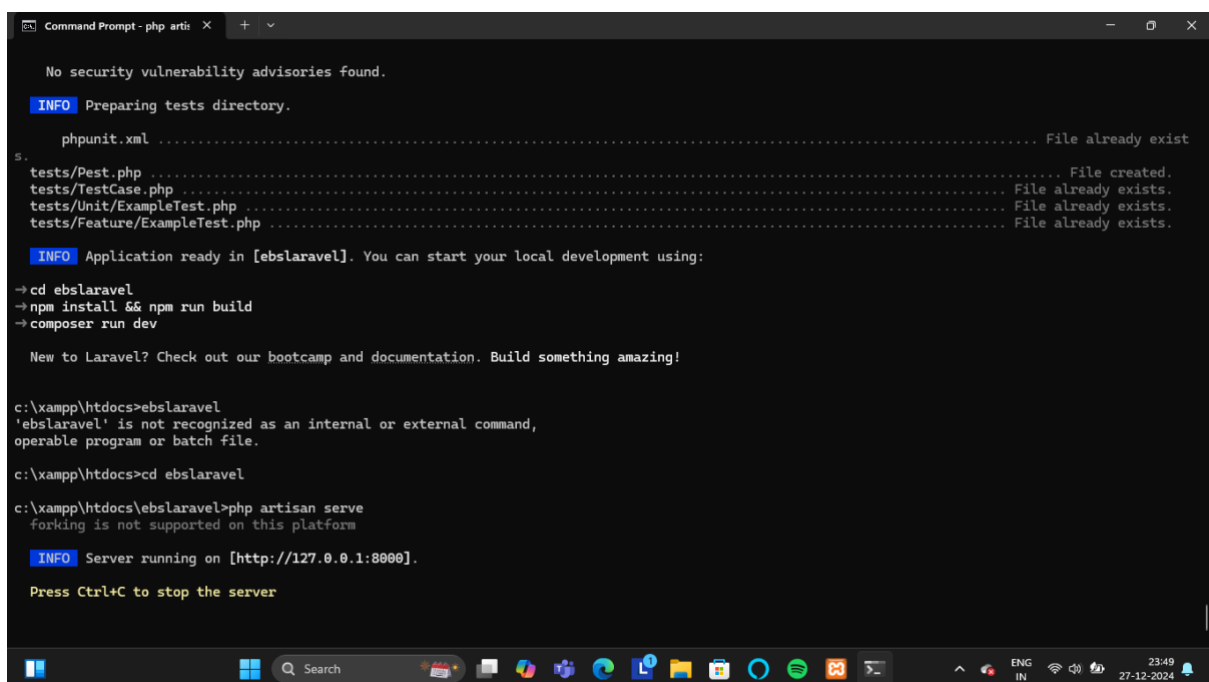
  LARAVEL

Would you like to install a starter kit? [No starter kit]:
[none] No starter kit
[breeze] Laravel Breeze
[jetstream] Laravel Jetstream
> none
none

Which testing framework do you prefer? [Pest]:
[0] Pest
[1] PHPUnit
>

Creating a "laravel/laravel" project at "./ebslaravel"
Installing laravel/laravel (v11.5.0)
- Installing laravel/laravel (v11.5.0): Extracting archive
Created project in C:\xampp\htdocs\ebslaravel
Loading composer repositories with package information
```


“The image below shows the continuation of setting up and running the Laravel project. The setup prepares the testing directory, creating files like phpunit.xml and test files for Pest. Laravel indicates the project is ready for development and provides commands to build assets and run the application. The user navigates to the project directory (cd ebsLaravel) and encounters an issue when trying to execute ebsLaravel, as it isn't recognized as a command. The user correctly runs php artisan serve, which starts the Laravel development server on <http://127.0.0.1:8000>. The message confirms the server is running, and the user can access the application locally.



```
Command Prompt - php artis X + v

No security vulnerability advisories found.

[INFO] Preparing tests directory.

phpunit.xml ..... File already exist
5. tests/Pest.php ..... File created.
   tests/TestCase.php ..... File already exists.
   tests/Unit/ExampleTest.php ..... File already exists.
   tests/Feature/ExampleTest.php ..... File already exists.

[INFO] Application ready in [ebslaravel]. You can start your local development using:

-> cd ebslaravel
-> npm install && npm run build
-> composer run dev

New to Laravel? Check out our bootcamp and documentation. Build something amazing!

c:\xampp\htdocs>ebslaravel
'ebslaravel' is not recognized as an internal or external command,
operable program or batch file.

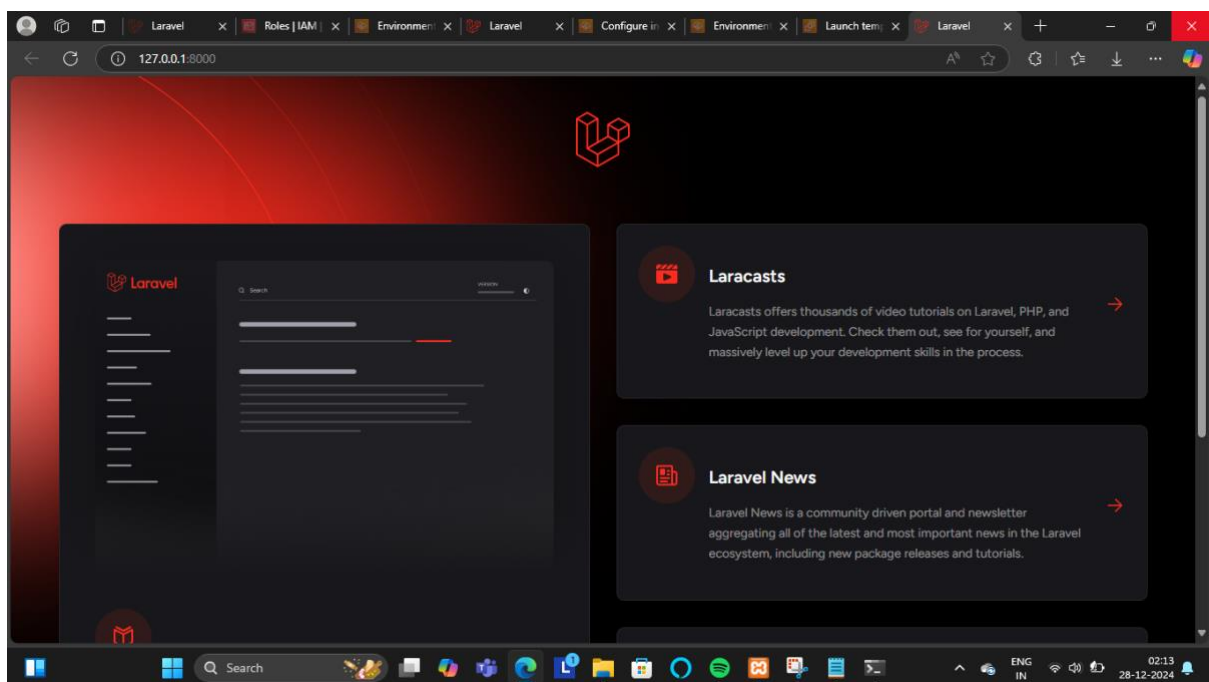
c:\xampp\htdocs>cd ebslaravel

c:\xampp\htdocs\ebslaravel>php artisan serve
forking is not supported on this platform

[INFO] Server running on [http://127.0.0.1:8000].

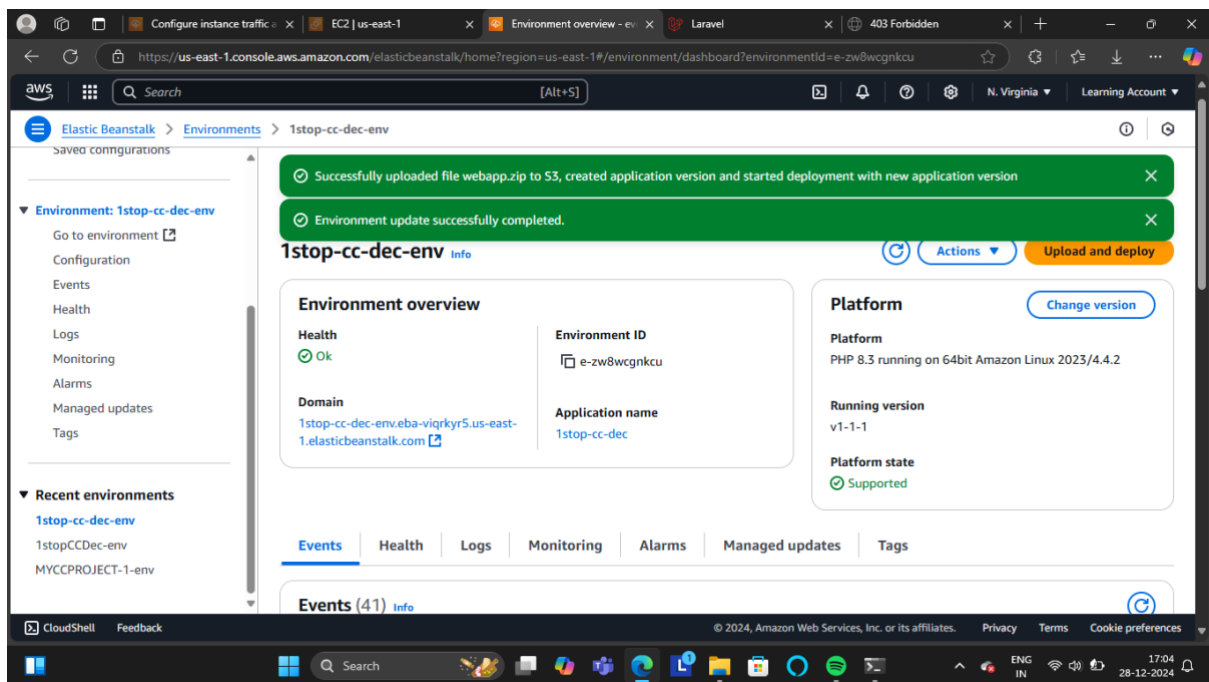
Press Ctrl+C to stop the server
```

“The image below shows the Laravel default welcome page loaded in a web browser at `http://127.0.0.1:8000`, confirming the Laravel development server is running successfully. This page serves as the initial interface for a new Laravel project and provides links to helpful resources like Laracasts (video tutorials) and Laravel News (community updates and tutorials). The design emphasizes Laravel's documentation and ecosystem, encouraging developers to explore tools, resources, and learning materials. This indicates that the user has successfully set up their Laravel environment, launched the server using `php artisan serve`, and can now start developing or customizing the application.”



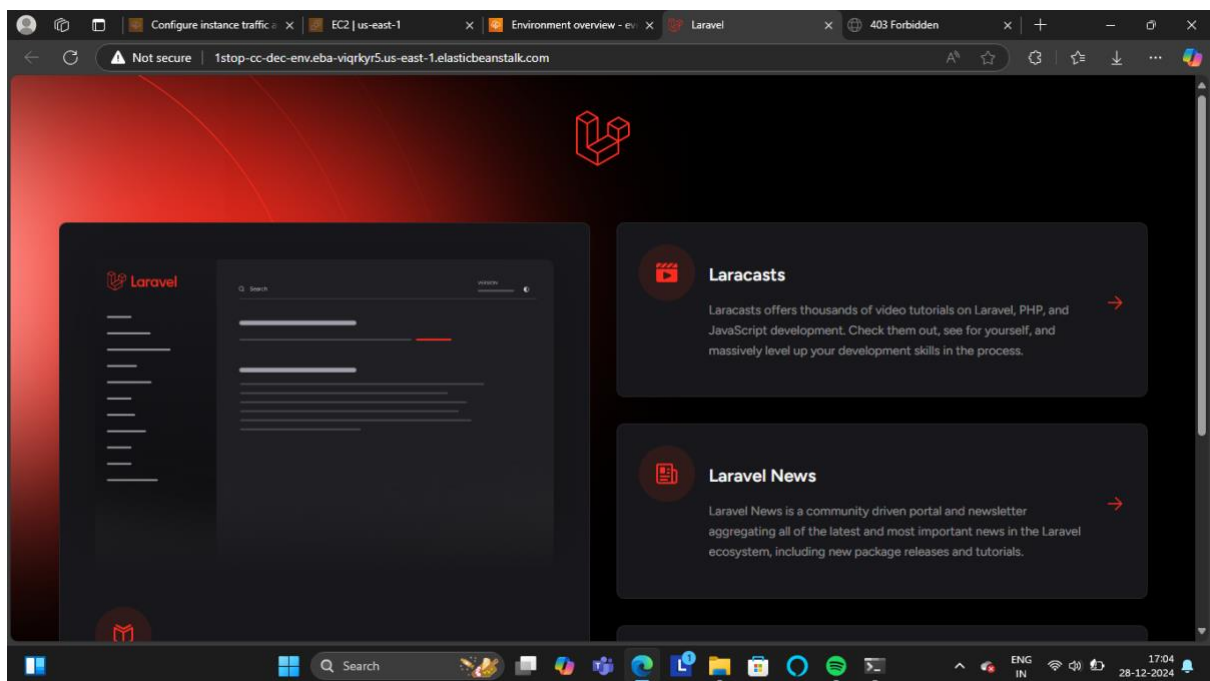
SNAPSHOT OF ENVIROMENT SETUP

This is the Elastic Beanstalk dashboard for the environment **1stop-cc-dec-env**. The environment is hosted on AWS and runs a PHP 8.3 application on a 64bit Amazon Linux 2023/4.4.2 platform. The application has been successfully updated, which involves uploading a new version of the web application to S3 and starting a deployment process. The environment is healthy and currently running version v1-1-1



OUTPUT

“This is the snapshot that showcases a web application deployed on AWS Elastic Beanstalk”



CONCLUSION

Leveraging Amazon Web Services (AWS) Elastic Beanstalk to host full-stack applications demonstrates a modern and efficient approach to application deployment, scalability, and management. Throughout this project, we explored the capabilities of Elastic Beanstalk as a Platform-as-a-Service (PaaS) solution, which simplifies the complexities of deploying and scaling full-stack applications by abstracting the underlying infrastructure management.

The integration of a full-stack application with Elastic Beanstalk showcases its versatility in supporting a wide array of web technologies and frameworks, including Node.js, Python, Java, .NET, and more. By automating critical deployment tasks—such as provisioning resources, load balancing, and monitoring—Elastic Beanstalk enables developers to focus on optimizing application functionality and user experience.

One of the most significant takeaways from this project is the ability of Elastic Beanstalk to streamline scalability. As the application's demand fluctuates, Elastic Beanstalk dynamically adjusts resources, ensuring optimal performance and cost efficiency. This is particularly beneficial for applications with unpredictable traffic patterns or those expected to grow significantly over time. Additionally, its built-in monitoring and logging capabilities enhance visibility into the application's performance, empowering teams to proactively address issues and maintain system reliability.

This project underscores the strategic value of adopting cloud-native deployment solutions like Elastic Beanstalk in a competitive technological landscape. Organizations can achieve faster time-to-market, reduced operational overhead, and improved agility, enabling them to respond to market needs effectively. Furthermore, leveraging AWS's robust ecosystem provides seamless

integration with complementary services, such as RDS for database management and S3 for storage, ensuring a holistic and scalable infrastructure.

In conclusion, Elastic Beanstalk proves to be a robust and user-friendly platform for hosting full-stack applications. Its blend of automation, scalability, and integration capabilities positions it as a vital tool for developers and organizations aiming to deliver high-performing, scalable, and reliable web applications. By adopting Elastic Beanstalk, teams can achieve operational efficiency and position their applications for sustainable growth in the ever-evolving digital landscape.