PROJECT 2

Leveraging Elastic Beanstalk and Configuring and Hosting Full Stack Application

RUPAM PATWARI

Abstract

This project aims to deploy a full-stack web application on Amazon Web Services (AWS) by leveraging AWS Elastic Beanstalk for application hosting, Amazon RDS for database management, and a LAMP (Linux, Apache, MySQL, PHP) stack for web development. By implementing these AWS services, the project demonstrates how to create a scalable and reliable environment for hosting a dynamic web application. The utilization of Elastic Beanstalk simplifies application deployment and scaling, while RDS ensures secure and managed database operations. The report outlines the steps taken to achieve this deployment and highlights the benefits of using AWS services in modern web application development.

Objective

The primary objective of this project is to demonstrate the end-to-end process of deploying a full-stack web application on Amazon Web Services (AWS) while harnessing the power of AWS Elastic Beanstalk for streamlined application hosting, Amazon RDS for efficient and managed database services, and the LAMP (Linux, Apache, MySQL, PHP) stack for robust web development.

Through this project, we aim to showcase the following key goals:

- Scalability: AWS Elastic Beanstalk create an environment that can seamlessly scale with varying levels of traffic, ensuring optimal performance and resource utilization.
- Reliability: Employ Amazon RDS to host the application's database, ensuring data integrity, automated backups, and high availability.
- Security: Implement best practices for securing both the application and the database within AWS, including proper IAM (Identity and Access Management) configurations and security groups.
- Efficiency: Demonstrate the efficiency of the LAMP stack for web development, emphasizing its role in building dynamic and interactive web applications.

Introduction

In today's digital landscape, the ability to develop, deploy, and manage web applications with speed, scalability, and security is paramount. Amazon Web Services (AWS) stands as a formidable platform that empowers developers and businesses to achieve these goals effortlessly. In this project, we embark on a journey to harness the potential of AWS by deploying a full-stack web application that showcases the synergy of AWS services.

Our project focuses on three core pillars:

- **Elastic Beanstalk for Application Hosting**: AWS Elastic Beanstalk is a Platform as a Service (PaaS) offering that simplifies the deployment and management of web applications. It enables developers to focus on their code, leaving the infrastructure management to AWS. We leverage Elastic Beanstalk to effortlessly deploy, scale, and maintain our web application.

- **Amazon RDS for Database Management**: Databases are the lifeblood of many applications, and Amazon RDS provides a reliable, scalable, and fully managed database solution. We utilize Amazon RDS to host our application's database, ensuring data integrity, high availability, and automated backups.

- **LAMP Stack for Web Development:** The LAMP stack, comprising Linux, Apache, MySQL, and PHP, is a time-tested and versatile choice for web development. Our project demonstrates the power of the LAMP stack in building dynamic, feature-rich web applications.

By seamlessly integrating these components, we aim to create a secure, scalable, and highly available environment for hosting our web application. Throughout this project, we will navigate the intricacies of AWS services..

Overall, this project aims to provide a comprehensive understanding of how AWS services can be leveraged to create a robust, scalable, and secure web application infrastructure, showcasing the benefits and advantages of AWS for modern web development and deployment.

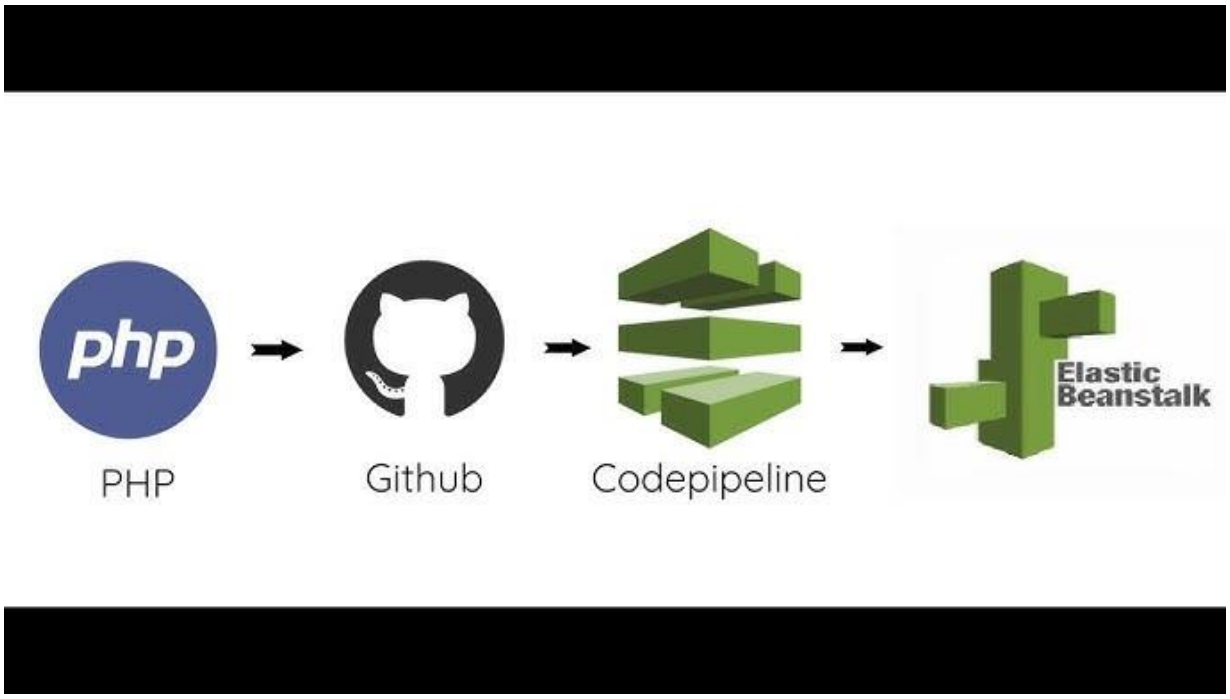
Steps followed to do this project are as follows :

A. Web application setup.
B. GitHub setup.
C. Creation of Ec2 instance.
D. Installation of PHP, GIT and Composer in Ec2 instance.
E. RDS database setup
F. Elastic Beanstalk setup.

In this project, I have attached the screenshots of my working platforms along with the text for better understanding.

Each step consists of those data that I have utilized in the project. We can put the data according to our needs. Not necessarily my data.

Sample diagram to visualize the architecture:



A. Web Application Setup: Steps are as follows:

- Download PHP from google by following the below steps:
  - PHP → Download → Windows download → download the zip file.
- Extract PHP file to the Local Disk (C:) by following the steps:
  - Extract to → Local Disk C: → Create new folder (PHP8211) → successfully extracted to PHP8211 folder.
- Now to run the Laravel (PHP web app framework) we need a composer.

  So to download composer we need to follow the below steps:
  - Google.com → Composer → Download → Composer-Setup.exe → Install
- Move to the command promt of our local device and type the following command in order to check whether PHP is working or not : **php –v**

  It will show ouput as :

```
Command Prompt
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\patwa>php -v
PHP 8.2.11 (cli) (built: Sep 26 2023 15:25:14) (NTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.11, Copyright (c) Zend Technologies

C:\Users\patwa>
```

- Now create a new Laravel project via composer using 'Create-project' command :

  Composer create-project laravel/laravel Project2

  But we will see that an error message is popped out showing the message as :

  "Your requirement couldnot be resolved to an installable set of packages"

```
Command Prompt
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Your requirements could not be resolved to an installable set of packages.

  Problem 1
    - laravel/framework[v10.10.0, ..., v10.26.2] require league/flysystem ^3.8.0 -> satisfiable by league/flysystem[3.8.
0, ..., 3.16.0].
    - league/flysystem[3.3.0, ..., 3.14.0] require league/mime-type-detection ^1.0.0 -> satisfiable by league/mime-type-
detection[1.0.0, ..., 1.13.0].
    - league/flysystem[3.15.0, ..., 3.16.0] require league/flysystem-local ^3.0.0 -> satisfiable by league/flysystem-loc
al[3.15.0, 3.16.0].
    - league/mime-type-detection[1.0.0, ..., 1.3.0] require php ^7.2 -> your php version (8.2.11) does not satisfy that
requirement.
    - league/mime-type-detection[1.4.0, ..., 1.13.0] require ext-fileinfo * -> it is missing from your system. Install o
r enable PHP's fileinfo extension.
    - league/flysystem-local[3.15.0, ..., 3.16.0] require ext-fileinfo * -> it is missing from your system. Install or e
nable PHP's fileinfo extension.
    - Root composer.json requires laravel/framework ^10.10 -> satisfiable by laravel/framework[v10.10.0, ..., v10.26.2].

To enable extensions, verify that they are enabled in your .ini files:
    - C:\php8211\php.ini
You can also run `php --ini` in a terminal to see which files are used by PHP in CLI mode.
Alternatively, you can run Composer with `--ignore-platform-req=ext-fileinfo` to temporarily ignore these required exten
sions.

C:\Users\patwa>
```

- Move to Local Disk C: → php8211 folder(already created) → open “php configuration setting” in notepad for better view → Search for ‘Extension’ → uncheck those ‘;’ from “extension=fileinfo” → save and exit.

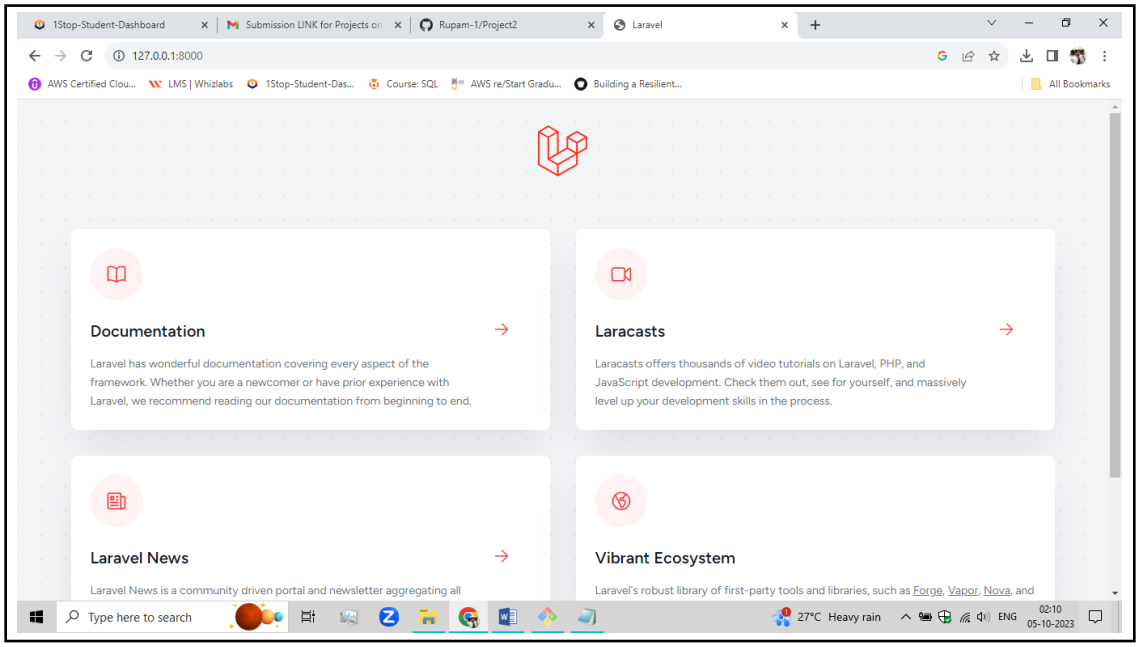
```
;extension=bz2

; The ldap extension must be before curl if OpenSSL 1.0.2 and OpenLDAP is used
; otherwise it results in segfault when unloading after using SASL.
; See https://github.com/php/php-src/issues/8620 for more info.
;extension=ldap
extension=curl
;extension=ffi
;extension=ftp
extension=fileinfo
;extension=gd
;extension=gettext
;extension=gmp
;extension=intl
;extension=imap
extension=mbstring
;extension=exif      ; Must be after mbstring as it depends on it
;extension=mysqli
;extension=oci8_12c  ; Use with Oracle Database 12c Instant Client
;extension=oci8_19  ; Use with Oracle Database 19 Instant Client
;extension=odbc
extension=openssl
;extension=pdo_firebird
;extension=pdo_mysql
;extension=pdo_oci
```

- Now move to cmd promt and create a folder in Local Disk C named “xampp” and within it create another folder named “cloud” by using the commands :

```
cd c:\
mkdir xampp
cd xampp
mkdir cloud
cd c:\xampp\cloud
```

- Now create the project using the command :

  `composer create-project laravel/laravel Project2` → Successfully created project.
- Now enter the command `cd Project2`
- `php artisan server` This will give a ip address which we can open to get a preview of our application.



B. GitHub setup:

- ✓ Create/Login to GitHub account
- ✓ Create a new repository keeping Repository name as ‘Project2-1stop’ and public.
- ✓ In the dashboard, we will get our repository.

  Code → copy the httpd (https://github.com/Rupam-1/Project2-1stop.git)
- ✓ Now we need to clone our repository.

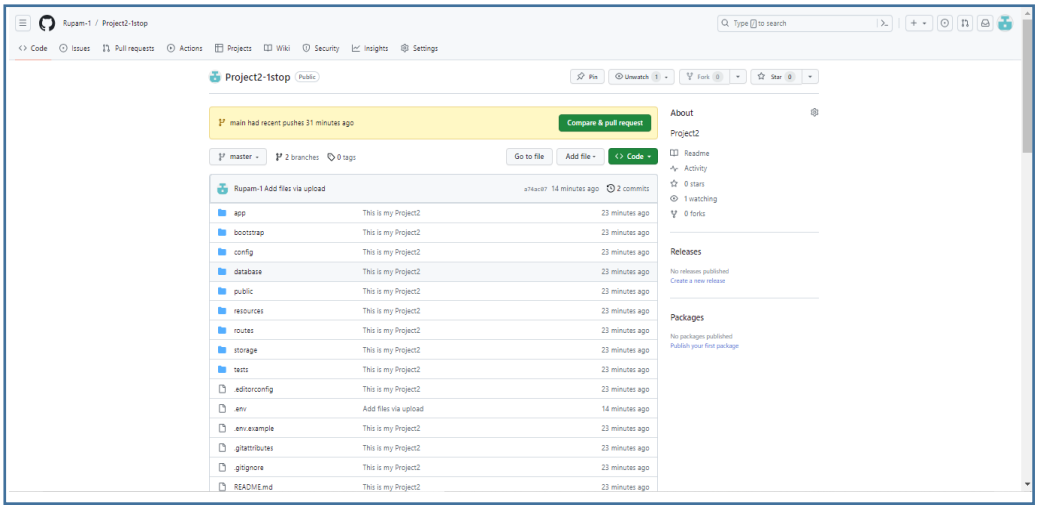
  But we will have only gitignore in our repository. So we need to upload our Project2 files in our repository (Without these files we cannot proceed further)

  For that, for easy access we can download GitBash and follow the steps:

```
cd c:/xampp/cloud
git init
git add .
git status
git commit –m “This is my Project2”
git remote add origin https://github.com/Rupam-1/Project2-1stop.git
git push -u origin master
```

```
 create mode 100644 tests/TestCase.php
 create mode 100644 tests/Unit/ExampleTest.php
 create mode 100644 vite.config.js

patwa@RUPAM MINGW64 /c/xampp/cloud/Project2 (master)
$ git remote add origin https://github.com/Rupam-1/Project2-1stop.git

patwa@RUPAM MINGW64 /c/xampp/cloud/Project2 (master)
$ git push -u origin master
Enumerating objects: 102, done.
Counting objects: 100% (102/102), done.
Delta compression using up to 4 threads
Compressing objects: 100% (84/84), done.
Writing objects: 100% (102/102), 72.19 KiB | 1.57 MiB/s, done.
Total 102 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/Rupam-1/Project2-1stop/pull/new/master
remote:
To https://github.com/Rupam-1/Project2-1stop.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

- ✓ Now we can see our entire file is uploaded in our github repository as below diagram:



C. Creation of Ec2 :

a) Sign in to AWS Console

b) Once you're logged in, navigate to the EC2 service by clicking on "Services" in the top-left corner and selecting "EC2" under the "Compute" section.

c) Click the "Launch Instance" button and name as “Project2-ec2”

d) Choose an Amazon Machine Image (AMI) → Amazon Linux 2

e) Choose an Instance Type → t3.micro

f) Create or Use an Existing Key Pair → Create a new key pair or use an existing one and store the private key file (.pem) because we'll need it to access your instance.

g) Configure Instance Details → We can configure various settings for your instance, such as the number of instances, network settings, IAM roles, and user data. Here we are leaving all these.

h) Add Storage → We can specify the amount and type of storage for your instance. The default settings typically include an Elastic Block Store (EBS) volume (8GB). We can adjust the size and type of this volume as needed.

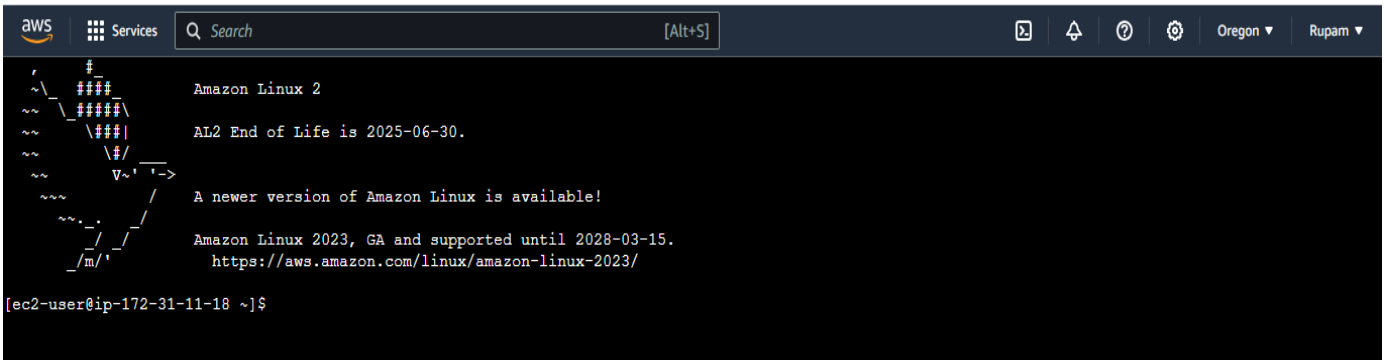
i) Add Tags (Optional) → Add tags to your instance for better organization and identification.

j) Configure Security Groups → Security groups act as firewalls for your instance, controlling inbound and outbound traffic. You can create a new security group or use an existing one.

k) Review and Launch → Review your instance's configuration to ensure everything is set up correctly. Click the "Launch" button when you're ready to proceed.

l) Launch the Instance → Click the "Launch Instances" button. Your EC2 instance will now be created.

m) Now connect to the Ec2 terminal as shown below:

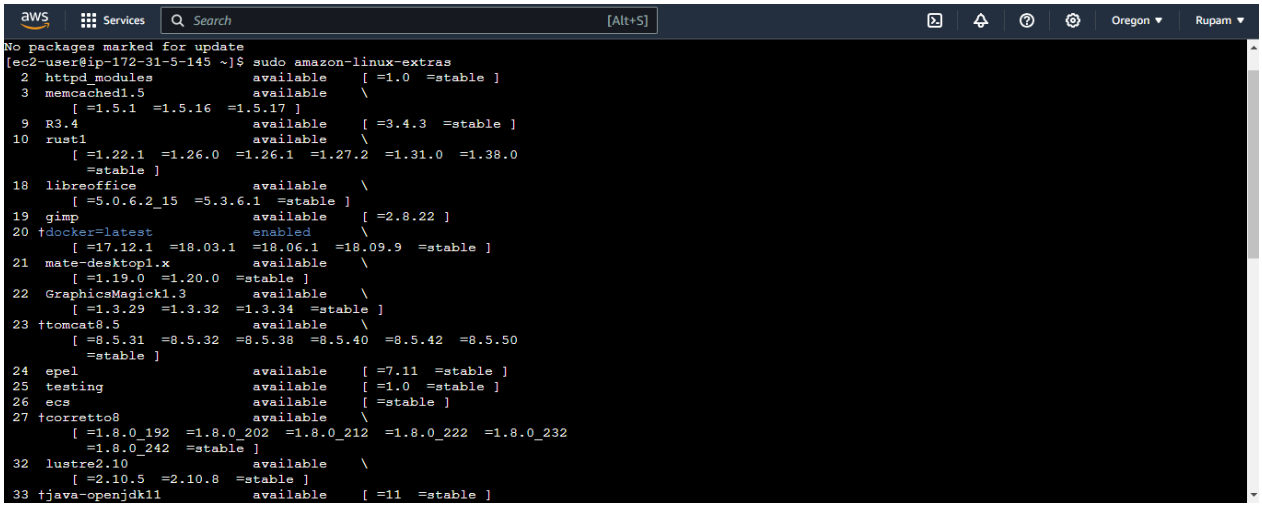


D. Installation of PHP,GIT & Composer in EC2 instance:

This will require a bunch of linux commands as shown below with screenshots:

☑ **sudo yum update –y** (checks whether system is updated or not)

☑ **sudo amazon-linux-extras** (it will show all the available softwares. We will select php8.1)



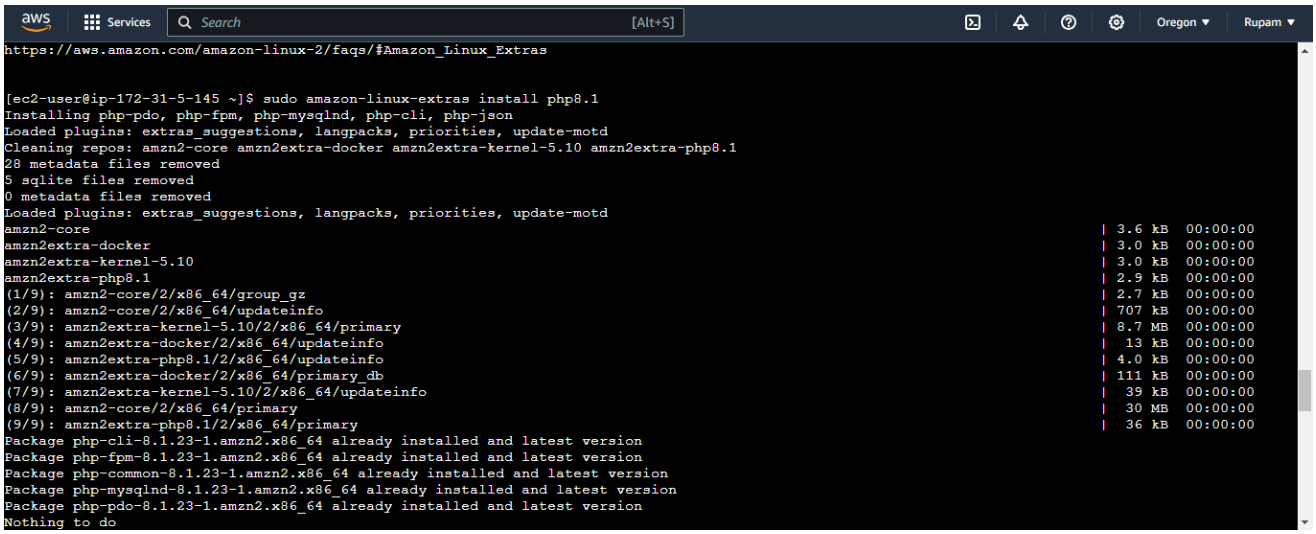
☑ **sudo amazon-linux-extras enable php8.1**

This command will make available php8.1 and linux2 will present 2 commands to install all dependencies of php8.1:

- 'Yum clean metadata'
- 'yum install php-cli php-pdo php-fpm php-json php-mysqlnd'

☑ **Sudo yum clean metadata**

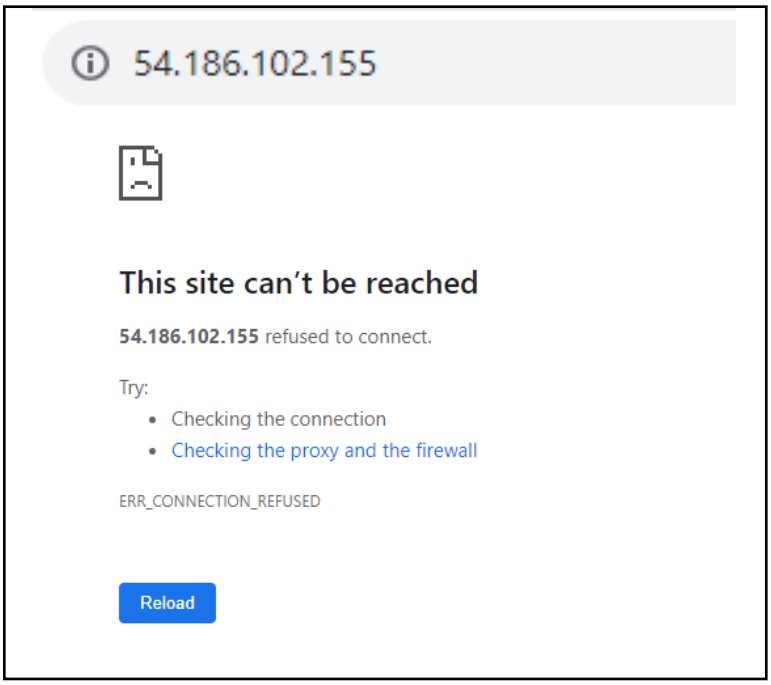
☑ **Sudo yum install php-cli php-pdo php-fpm php-json php-mysqlnd**



☑ **Sudo amazon-linux-extras install php8.1** (will install php8.1 in terminal)

☑ **Sudo yum install –y httpd** (installs Apache server)

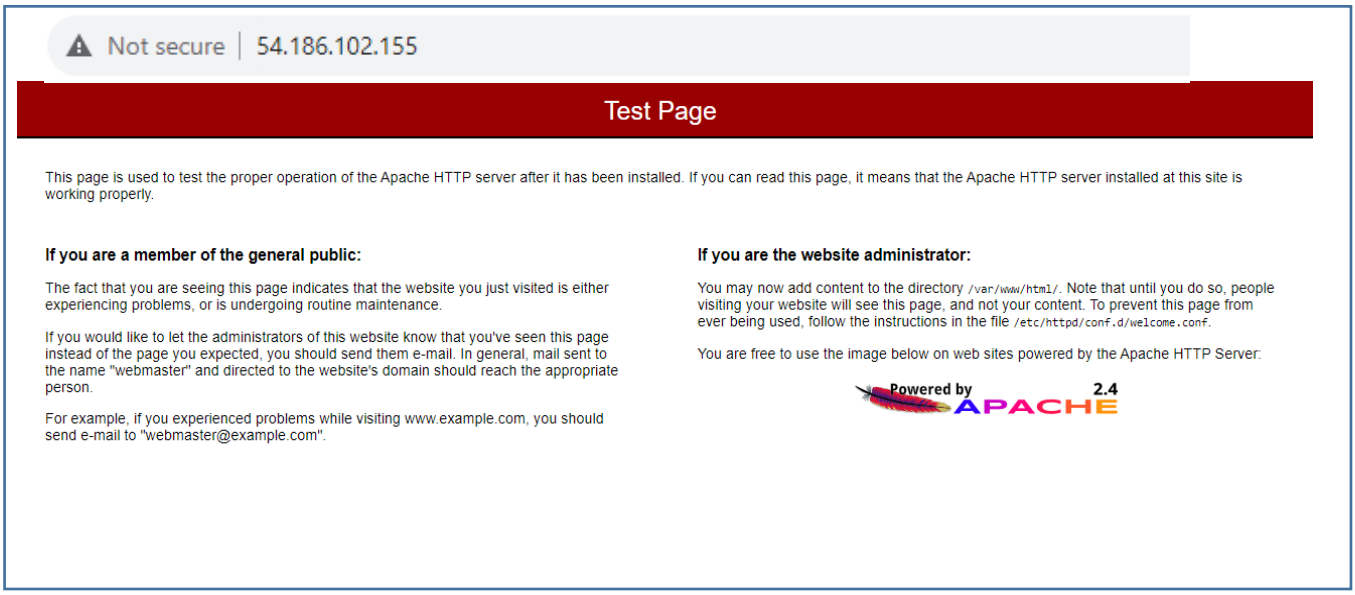
Now if we copy the ip address and open it in new tab. It will not show the content as apache is installed but not yet enabled.



☑ **sudo systemctl start httpd** (starts Apache server)

☑ **sudo systemctl enable httpd**

Now if we again open that tab, we can see the Apache server is working fine.



☑ **sudo usermod -a -G apache ec2-user** (adds ec2-user to the Apache group)

☑ **sudo chown –R ec2-user:apache /var/www**

(Changes the ownership of apache group from root user to ec2-user)

☑ **sudo chmod 2775 /var/www && find /var/www –type d –exec sudo chmod 2775 {} \;**

(changes the permission of directory and sub-directories in /var/www)

☑ **find /var/www/ -type f –exec sudo chmod 0664 {} \;**

```
[ec2-user@ip-172-31-5-145 ~]$ sudo chown -R ec2-user:apache /var/www
[ec2-user@ip-172-31-5-145 ~]$ sudo chown -R ec2-user:apache /var/www^C
[ec2-user@ip-172-31-5-145 ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;
[ec2-user@ip-172-31-5-145 ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;^C
[ec2-user@ip-172-31-5-145 ~]$ find /var/www/ -type f -exec sudo chmod 0664 {} \;
[ec2-user@ip-172-31-5-145 ~]$
```

☑ **sudo yum install –y git**

```
[ec2-user@ip-172-31-5-145 ~]$ sudo yum install -y git
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                                               | 3.6 kB  00:00:00
Resolving Dependencies
--> Running transaction check
---> Package git.x86_64 0:2.40.1-1.amzn2.0.1 will be installed
--> Processing Dependency: perl-Git = 2.40.1-1.amzn2.0.1 for package: git-2.40.1-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core-doc = 2.40.1-1.amzn2.0.1 for package: git-2.40.1-1.amzn2.0.1.x86_64
--> Processing Dependency: git-core = 2.40.1-1.amzn2.0.1 for package: git-2.40.1-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Term::ReadKey) for package: git-2.40.1-1.amzn2.0.1.x86_64
--> Processing Dependency: perl(Git) for package: git-2.40.1-1.amzn2.0.1.x86_64
--> Running transaction check
---> Package git-core.x86_64 0:2.40.1-1.amzn2.0.1 will be installed
---> Package git-core-doc.noarch 0:2.40.1-1.amzn2.0.1 will be installed
---> Package perl-Git.noarch 0:2.40.1-1.amzn2.0.1 will be installed
--> Processing Dependency: perl(Error) for package: perl-Git-2.40.1-1.amzn2.0.1.noarch
---> Package perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2 will be installed
--> Running transaction check
---> Package perl-Error.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
================================================================================================================================
 Package                    Arch                    Version                    Repository                    Size
================================================================================================================================
Installing:
 git                        x86_64                  2.40.1-1.amzn2.0.1         amzn2-core                    54 k
Installing for dependencies:
 git-core                   x86_64                  2.40.1-1.amzn2.0.1         amzn2-core                    10 M
```

☑ **sudo curl -sS https://getcomposer.org/installer | sudo php** (installs composer)

☑ **sudo mv composer.phar /usr/local/bin/composer**

(Moves composer.phar to /usr/local/bin and renames it to ‘composer’)

☑ **sudo ln –s /usr/local/bin/composer /usr/bin/composer**

(Links the composer so that it comes at first)

☑ **cd /var/www/html** (moving inside /html)

☑ **ls** (it will show an empty space)

☑ **git clone https://github.com/Rupam-1/Project2-1stop.git**

(It will clone my github repo inside /var/www/html which we can check)

```
[ec2-user@ip-172-31-5-145 html]$ git clone https://github.com/Rupam-1/Project2-1stop.git
Cloning into 'Project2-1stop'...
remote: Enumerating objects: 108, done.
remote: Counting objects: 100% (108/108), done.
remote: Compressing objects: 100% (83/83), done.
remote: Total 108 (delta 7), reused 101 (delta 6), pack-reused 0
Receiving objects: 100% (108/108), 73.79 KiB | 4.92 MiB/s, done.
Resolving deltas: 100% (7/7), done.
```

☑ **ls** (we can see our Project2-1stop directory successfully cloned inside)

☑ **cd Project2-1stop**

☑ **composer install**

(will install composer but some errors will pop up as all the packages will not be downloaded by default like .xml and .dom files which we will separately download)

```
[ec2-user@ip-172-31-5-145 html]$ ls
Project2-1stop
[ec2-user@ip-172-31-5-145 html]$ cd Project2-1stop
[ec2-user@ip-172-31-5-145 Project2-1stop]$ composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Your lock file does not contain a compatible set of packages. Please run composer update.

  Problem 1
    - tijsverkoyen/css-to-inline-styles is locked to version 2.2.6 and an update of this package was not requested.
    - tijsverkoyen/css-to-inline-styles 2.2.6 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
  Problem 2
    - laravel/pint is locked to version v1.13.2 and an update of this package was not requested.
    - laravel/pint v1.13.2 requires ext-xml * -> it is missing from your system. Install or enable PHP's xml extension.
  Problem 3
    - phar-io/manifest is locked to version 2.0.3 and an update of this package was not requested.
    - phar-io/manifest 2.0.3 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
  Problem 4
    - phpunit/php-code-coverage is locked to version 10.1.7 and an update of this package was not requested.
    - phpunit/php-code-coverage 10.1.7 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
  Problem 5
    - phpunit/phpunit is locked to version 10.3.5 and an update of this package was not requested.
    - phpunit/phpunit 10.3.5 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
  Problem 6
    - sebastian/comparator is locked to version 5.0.1 and an update of this package was not requested.
    - sebastian/comparator 5.0.1 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
  Problem 7
    - theseer/tokenizer is locked to version 1.2.1 and an update of this package was not requested.
    - theseer/tokenizer 1.2.1 requires ext-dom * -> it is missing from your system. Install or enable PHP's dom extension.
  Problem 8
```

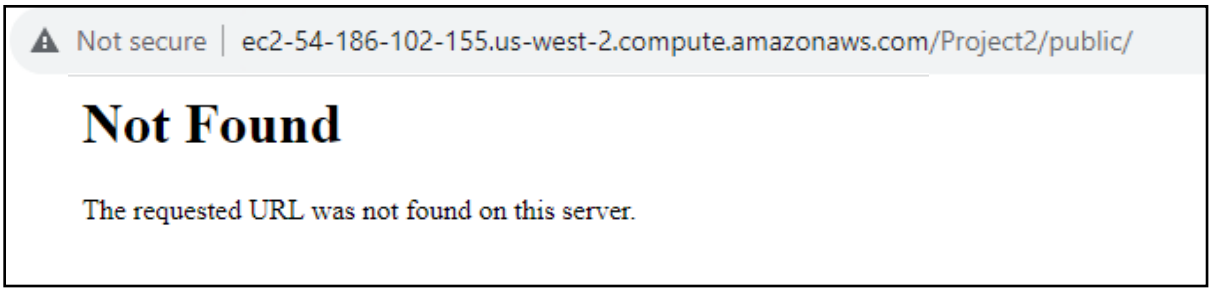
☑ **sudo yum install –y php-xml**

☑ **sudo yum install –y php-dom**

☑ **composer install**

```
  - Installing myclabs/deep-copy (1.11.1): Extracting archive
  - Installing phpunit/phpunit (10.3.5): Extracting archive
  - Installing spatie/backtrace (1.5.3): Extracting archive
  - Installing spatie/flare-client-php (1.4.2): Extracting archive
  - Installing spatie/ignition (1.11.2): Extracting archive
  - Installing spatie/laravel-ignition (2.3.0): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

  INFO  Discovering packages.

  laravel/sail ..................................................... DONE
  laravel/sanctum .................................................. DONE
  laravel/tinker ................................................... DONE
  nesbot/carbon .................................................... DONE
  nunomaduro/collision ............................................. DONE
  nunomaduro/termwind .............................................. DONE
  spatie/laravel-ignition .......................................... DONE

82 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
[ec2-user@ip-172-31-5-145 Project2-1stop]$
```

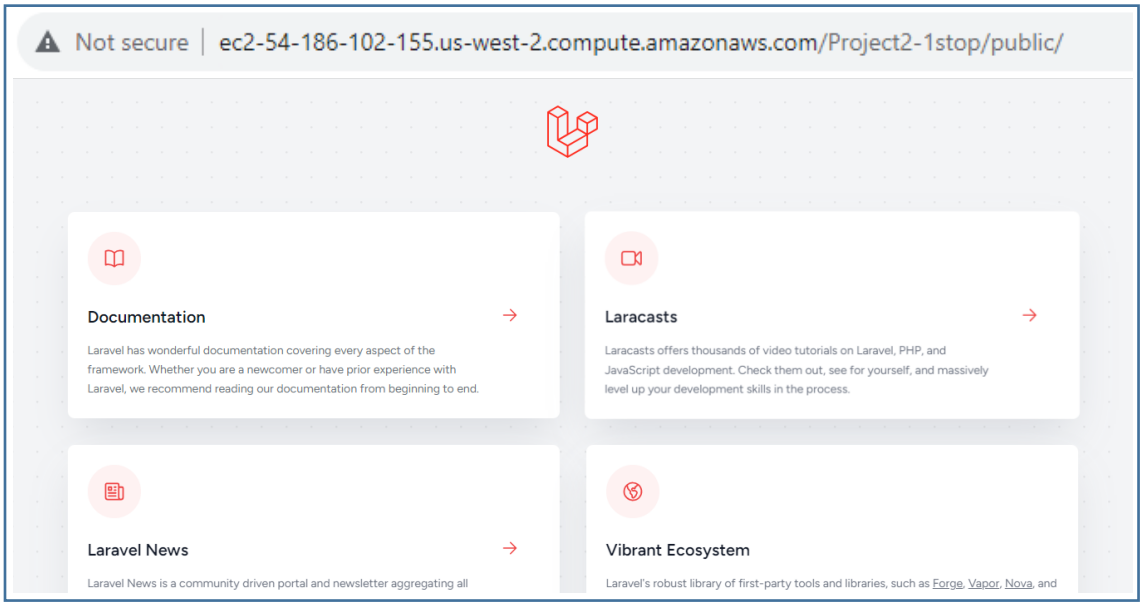
Now, we can move to the previously opened tab and search for ‘http://ec2-54-186-102-155.us-west-2.compute.amazonaws.com/Project2/public/’. It will show some error.



- ☑ **cp .env.example .env**
- ☑ **php artisan key:generate**

```
[ec2-user@ip-172-31-5-145 Project2-1stop]$ php artisan key:generate

  INFO  Application key set successfully.

[ec2-user@ip-172-31-5-145 Project2-1stop]$
```

Now we can see Laravel webpage is running fine

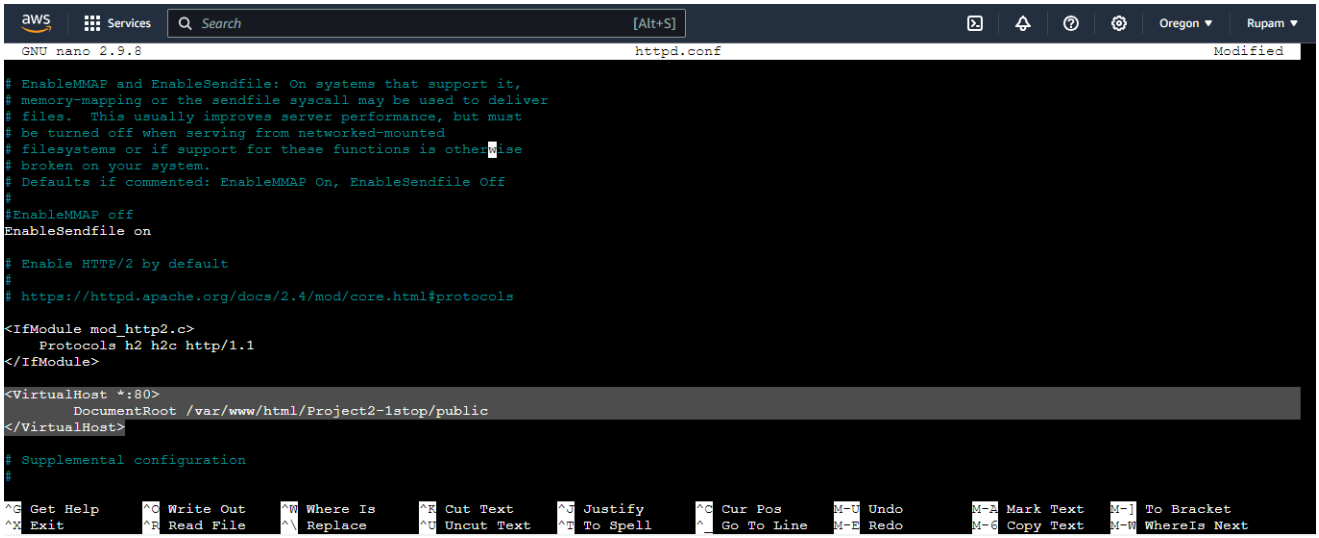


But some changes we have to make.

- ☑ **cd /etc/httpd**
- ☑ **ls**
- ☑ **cd conf**

```
[ec2-user@ip-172-31-5-145 Project2-1stop]$ cd /etc/httpd/
[ec2-user@ip-172-31-5-145 httpd]$ ls
conf  conf.d  conf.modules.d  logs  modules  run  state
```

☑ **Sudo nano httpd.conf**



Here we need to add some commands:

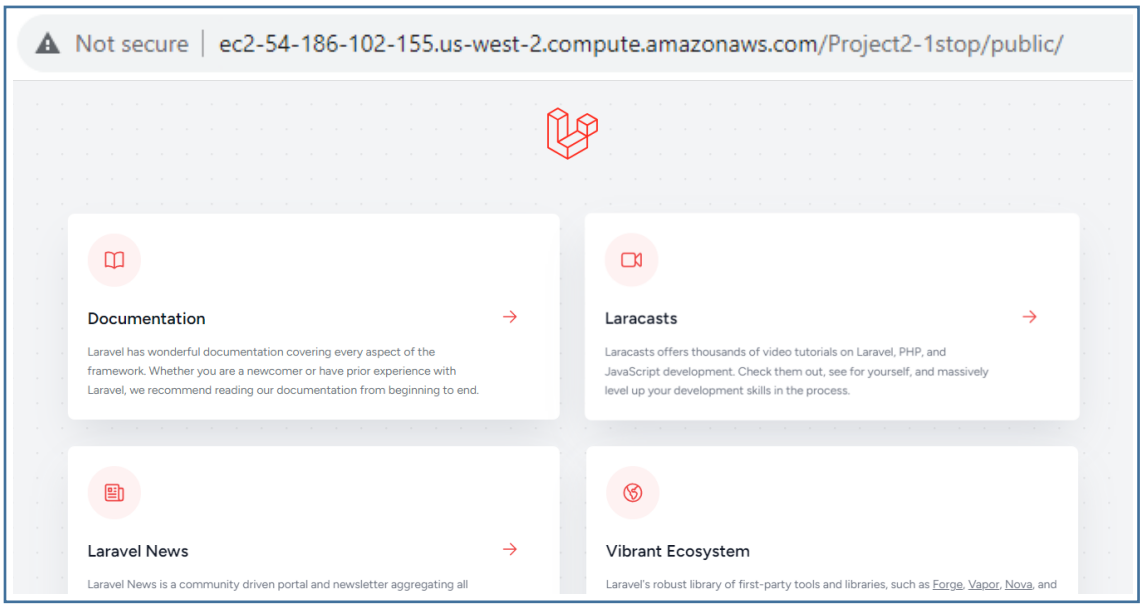
<VirtualHost *:80>

DocumentRoot /var/www/html/Project2-1stop/public

</VirtualHost>

☑ **Sudo systemctl restart httpd**. (restarts our webpage)

Now open the webpage and we can see its working fine.



E. RDS Database setup:

Steps as follows:

- Sign in to AWS Console.
- Navigate to Amazon RDS: In the AWS Console, go to the "Services" menu, select "Database," and then choose "RDS."
- Click the "Create database" button.
- Select the database engine we want to use → MariaDB
- Choose template that suits our needs → Free Tier
- Database identifier (Name) → Project2-DB
- Configure the instance details, including DB instance size, storage, and other settings.
- Specify the master username → Rupam
- Master password → Rupam
- Instance type → db.t2.micro
- Configure Advanced Settings: Configure additional settings like VPC, security groups, backups, and maintenance preferences according to your requirements.
- Review your configuration settings and click "Create database."

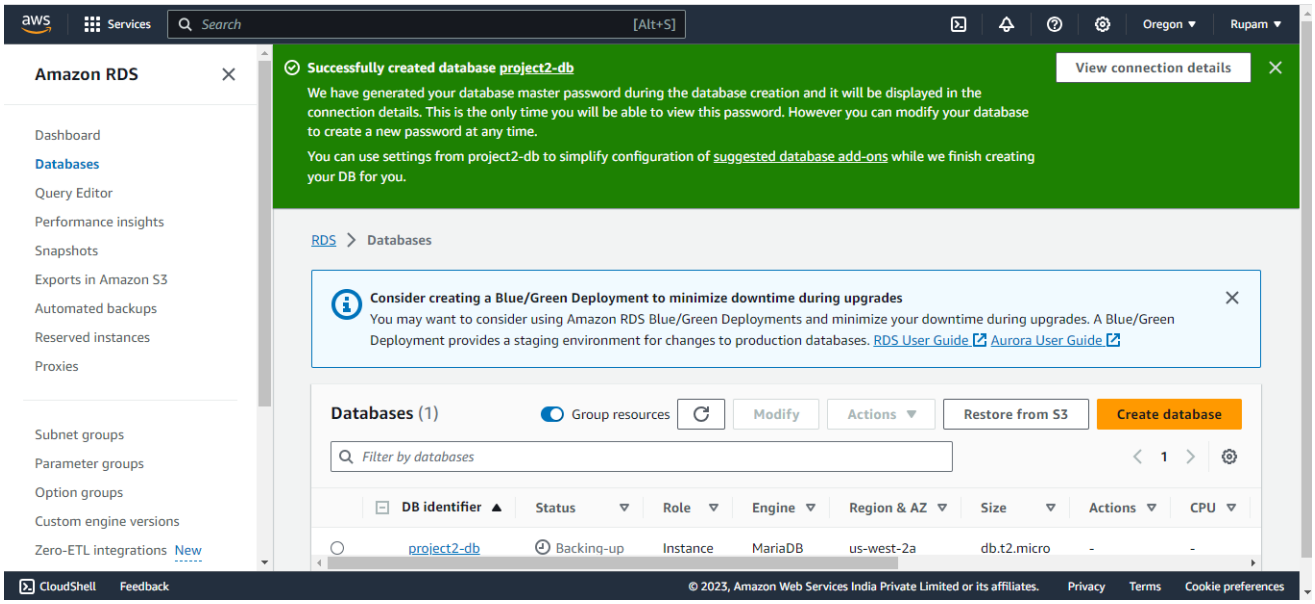
AWS will provision the RDS instance, and it may take a few minutes to complete.

Once the database is created, we can access it using the endpoint provided in the RDS dashboard.

Note down the credentials of our Database i.e. Master password: **********.

Endpoint: **********.

Master Username: Rupam.

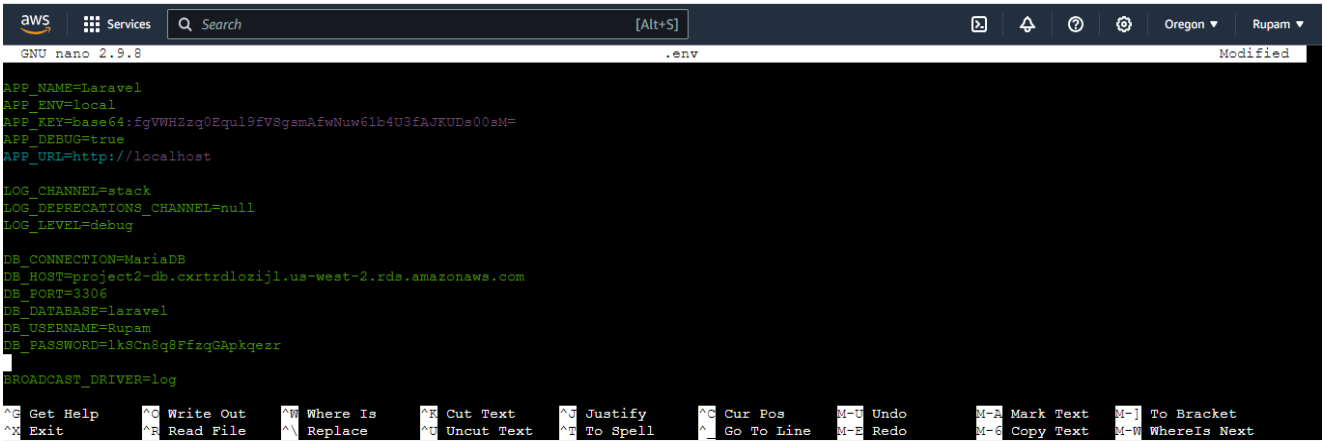


Now get back to our Ec2 terminal to attach our RDS to our terminal. For that we need to enter the credentials inside the nano editor of .env and make the changes under the database portion.

- ☑ **Cd /var/ww/html**
- ☑ **Cd Project2-1stop**
- ☑ **Nano .env**

  Make the changes as below:

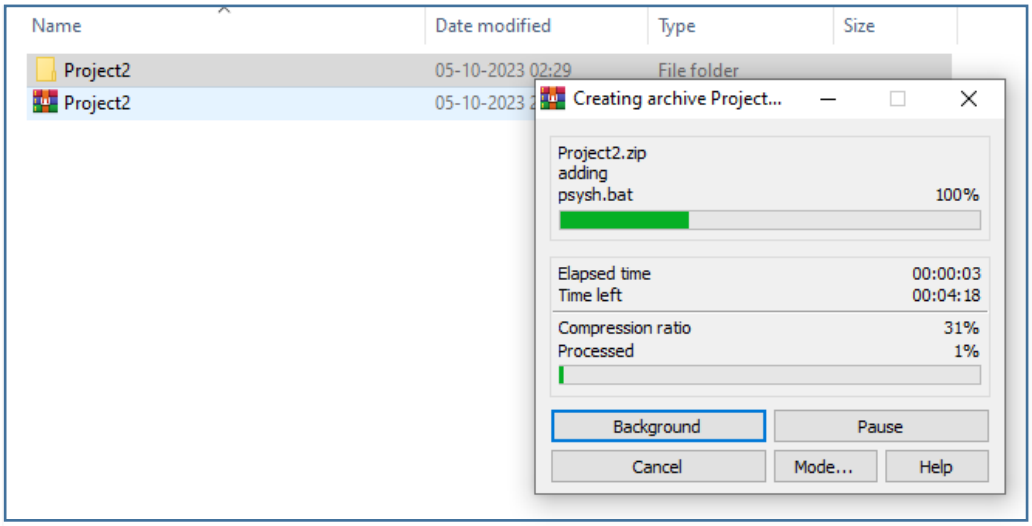
  - DB_CONNECTION = MariaDB
  - DB_HOST = project2-db.cxrtrdlozijl.us-west-2.rds.amazonaws.com
  - DB_PORT=3306
  - DB_DATABASE = laravel
  - DB_USERNAME = Rupam
  - DB_PASSWORD = lkSCn8q8FfzqGApkqezr



Hence we are done with the configuration of RDS with our instance.

Now moving to our Local Disk C → xampp → cloud

Here, we will zip our Project2-1stop so as to put in our Elastic Beanstalk.



F. Elastic Beanstalk setup :

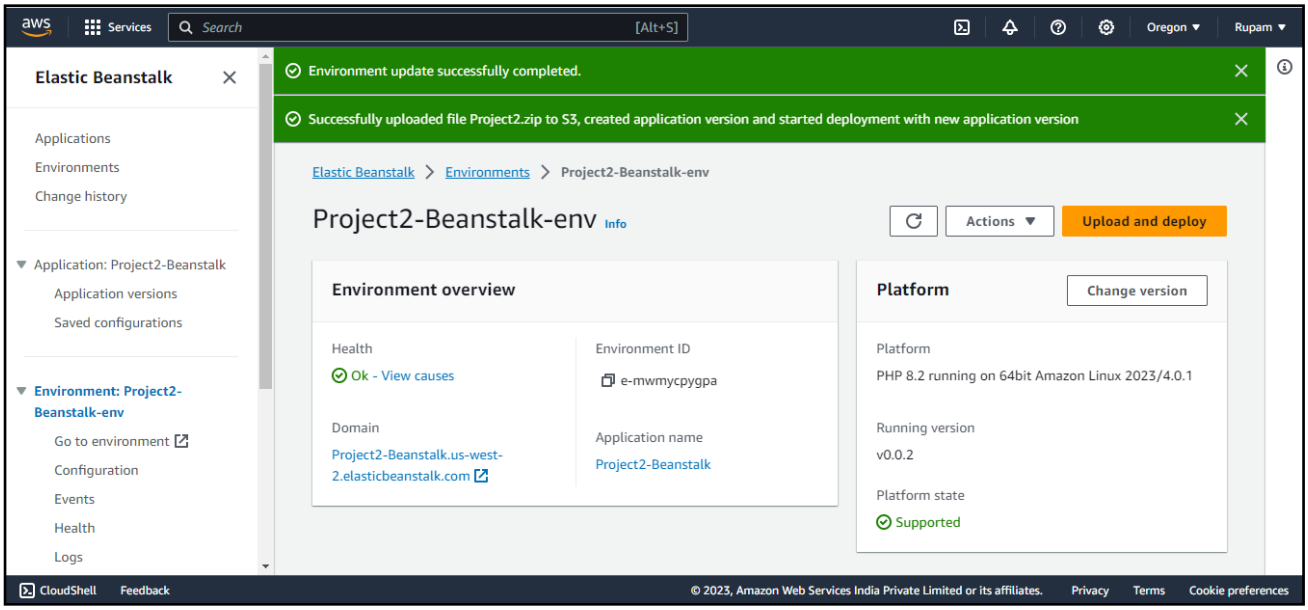
At first we have to create IAM user and role to get access to beanstalk and attach the IAM role to the IAM user.

- Sign in to AWS Console
- IAM → User → create user → User name : beanstalk user
  Attach policies : Administrator access
  Beanstalk MultiContainer Docker
  Beanstalk Web Tier
  Beanstalk Worker Tier
- After user created. Copy the user ARN.
- IAM Roles → Create Role → policies : Administrator access
  Beanstalk Web Tier
  Beanstalk Worker Tier
  Beanstalk MultiContainer Docker
  EC2 Full Access
- Role created → Trust Relations → Edit Trust policy → Add principal → Paste the copied IAM user ARN → update policy

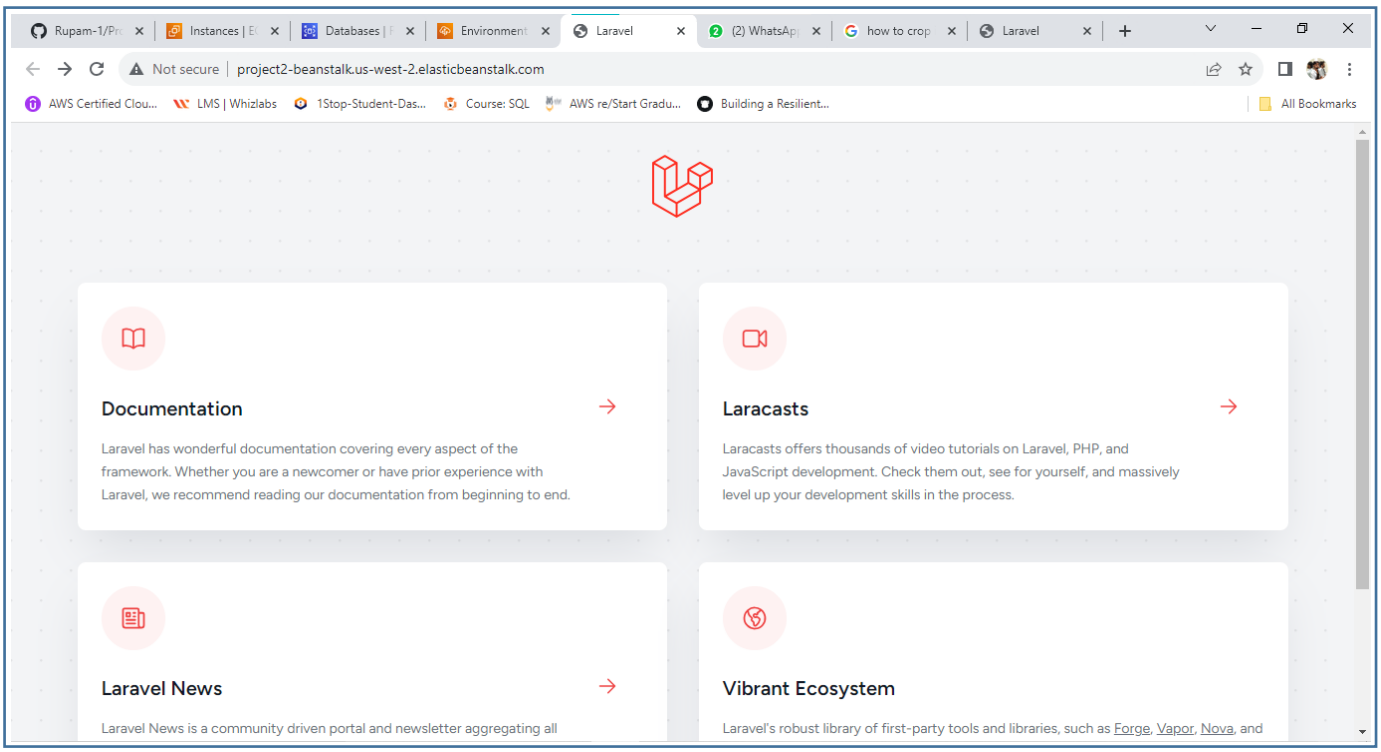
Now,

- Create an Elastic Beanstalk Application → In the AWS Console, navigate to the "Services" menu and select "Elastic Beanstalk."
- Click the "Create Application" button.
- Configure Environment → select Web Server environment
  Name → Project2-beanstalk
  Domain name → Project2-Beanstalk.us-west-2.elasticbeanstalk.com
  Platform → PHP
  Upload code → Local
  Here we will upload that zipped file of our Project2
- Existing service roles → aws-elasticbeanstalk-service-role
- Ec2 instance profile → beanstalk-ec2-role
- Setup networking, database and tags → keep as it is
- Configure instance traffic & scaling → instance type : t2.micro
- Review and create

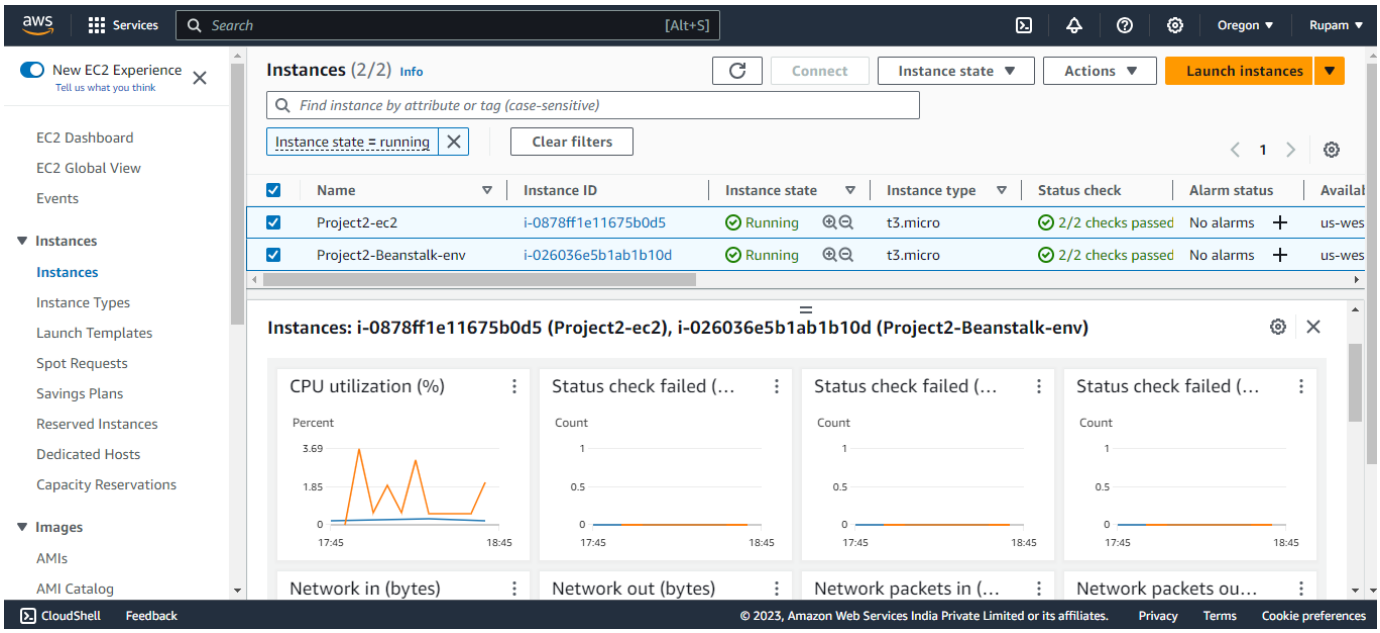
Hence, completed Beanstalk configuration. It will take some time to configure.



Final view of our webpage after successful deployment of Benastalk:



EC2 Dashboard after successful creation of Beanstalk



Here, we have deployed the Project2-ec2 instance at the initial steps.

Project2-Beanstalk-env is auto created by the beanstalk itself as we have deployed maximum of 3 instances.

Conclusion

In conclusion, this project has successfully demonstrated the capabilities of Amazon Web Services (AWS) in hosting a full-stack web application with efficiency, scalability, and security. Leveraging AWS Elastic Beanstalk for simplified deployment, Amazon RDS for robust database management, and the versatile LAMP stack for web development, we've showcased the power of cloud computing in modern application hosting.

In today's competitive digital landscape, the ability to swiftly deploy, scale, and secure web applications is indispensable. By embracing AWS services, this project exemplifies the advantages of cloud computing, enabling developers and businesses to focus on innovation while AWS takes care of the underlying infrastructure.

As we conclude this project, we encourage further exploration of AWS services and their endless possibilities in modern web development and deployment. Whether we are a seasoned developer or new to the cloud, AWS provides the tools and resources to bring your web applications to life, securely and seamlessly.