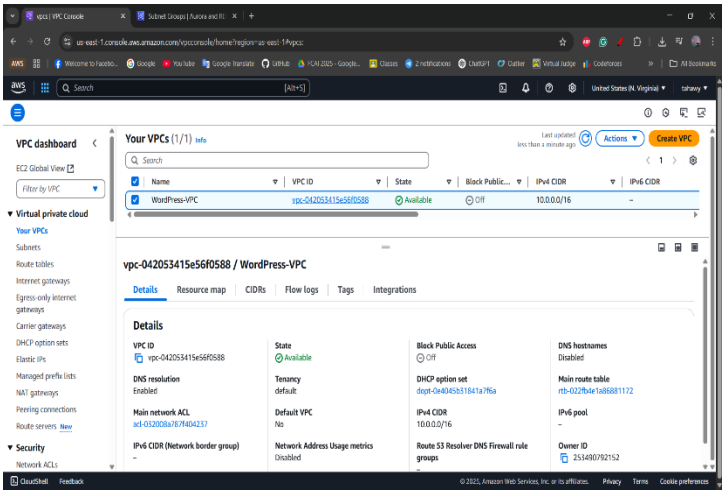
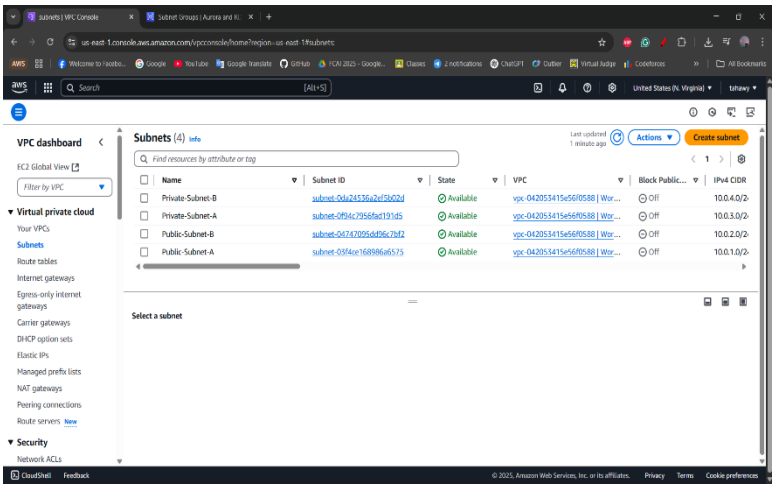
Task 1: Deploy a Scalable WordPress Website Using EC2, RDS, ALB, and ASG (Dynamic Site)

GitHub Repo | CFN ALB | NO CFN ALB

VPC and Subnets:

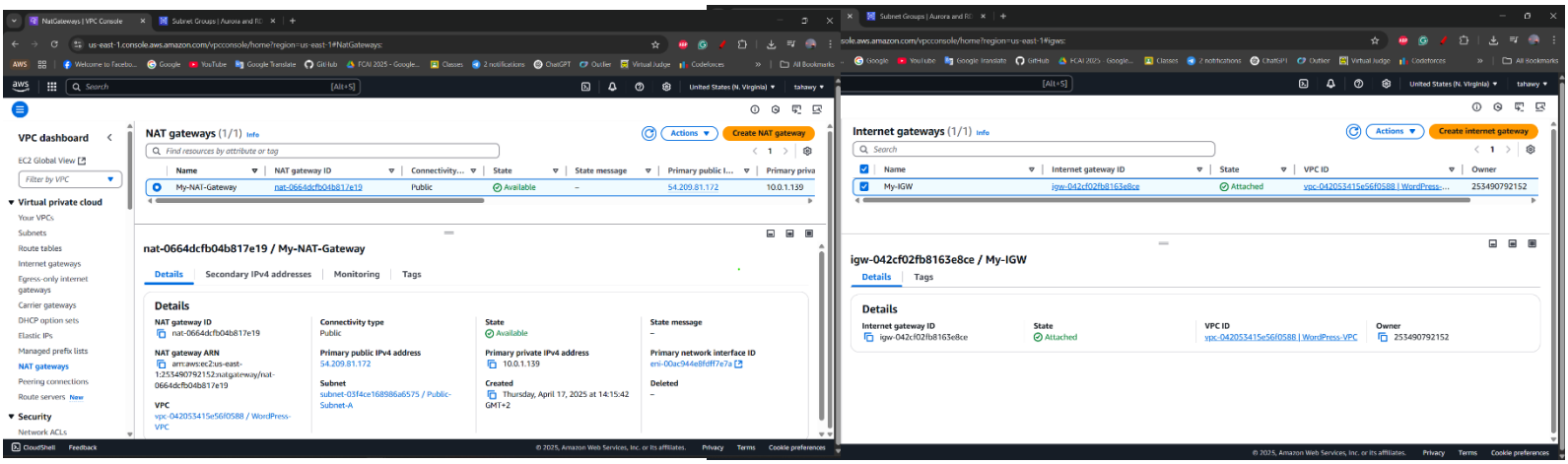
Creating a VPC to have our own private network inside the cloud for our WordPress website and four subnets (two public and two private) by placing subnets in **two different AZs** to ensure that if one AZ experiences and outage, the application remains available in the other AZ.





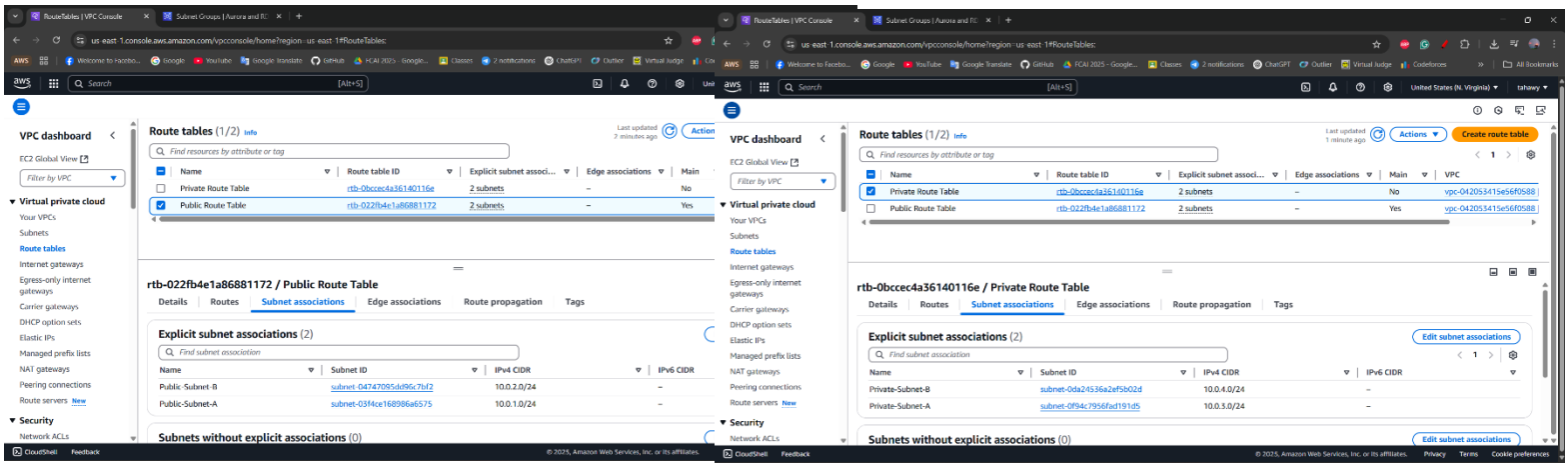
IGW and NGW:

The IGW is to allow resources in a **public subnet** to communicate directly with the internet. The NGW is to allow resources in **private subnets** to initiate outbound traffic without exposing them to inbound internet traffic.



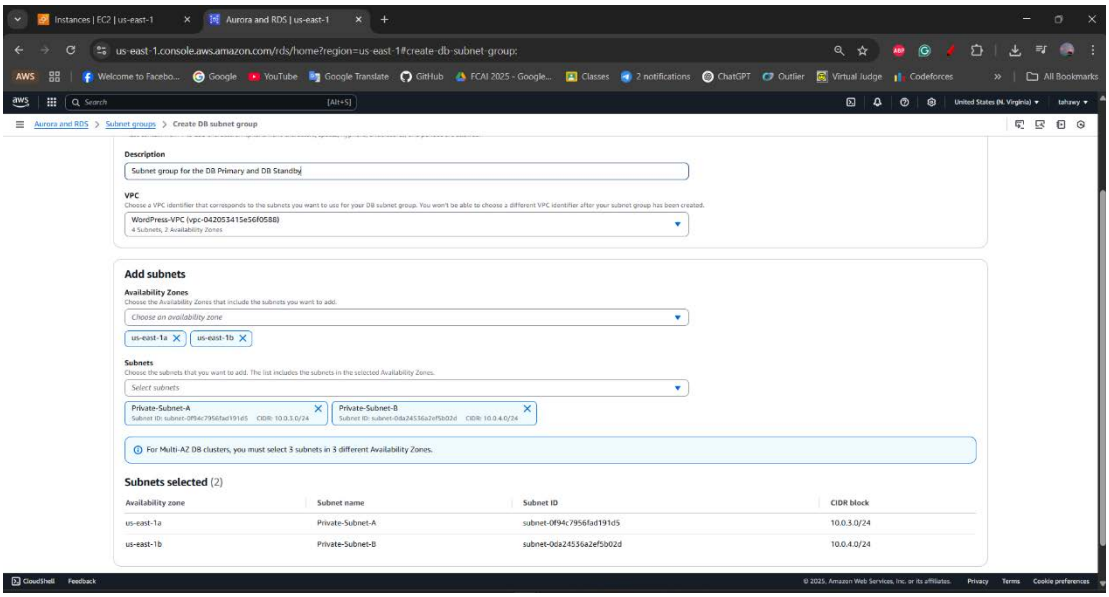
Route Tables:

Route Tables are used to control the routing of network traffic within VPC. They determine how packets are directed between subnets, the internet, and other network resources.
Added a route to direct traffic destined for the internet (0.0.0.0/0) to the Internet Gateway (IGW). This allows resources in the public subnet to access the internet. Also, there was added a route for internet-bound traffic (0.0.0.0/0) to the NAT Gateway. This allows instances in private subnets to access the internet for updates and patches, but they remain inaccessible from the internet.



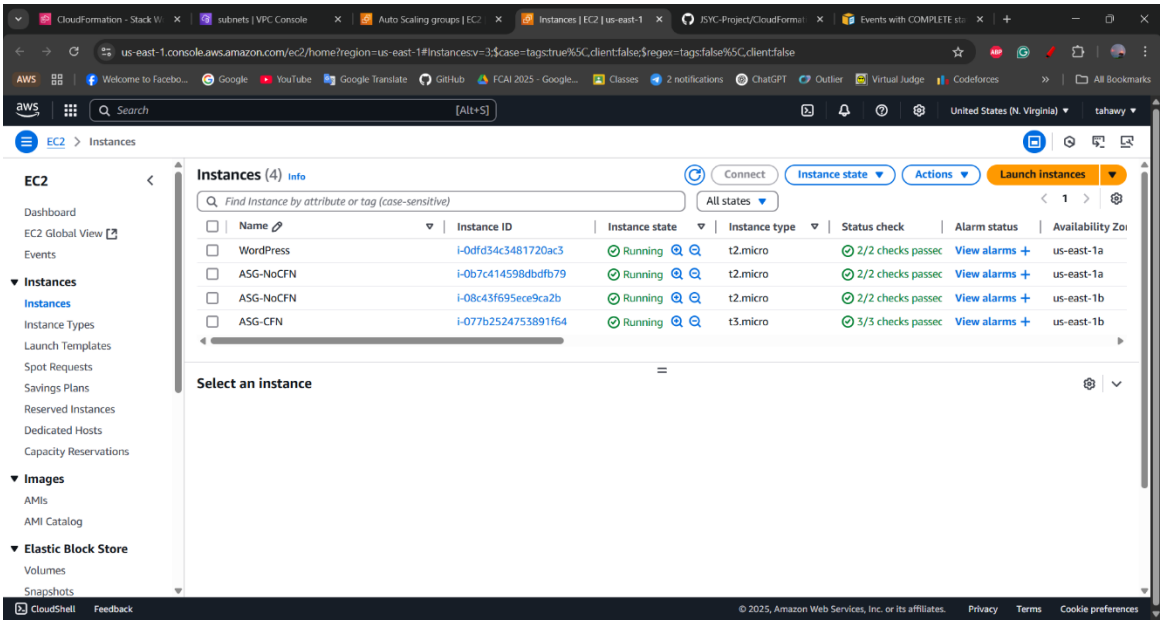
RDS Subnet Group:

This is a subnet group for RDS that has two private subnets, they are two for high availability and security, and when setting up **Amazon RDS** for WordPress project, AWS requires you to create a **DB Subnet Group** with **at least two private subnets in different Availability Zones (AZs)**.



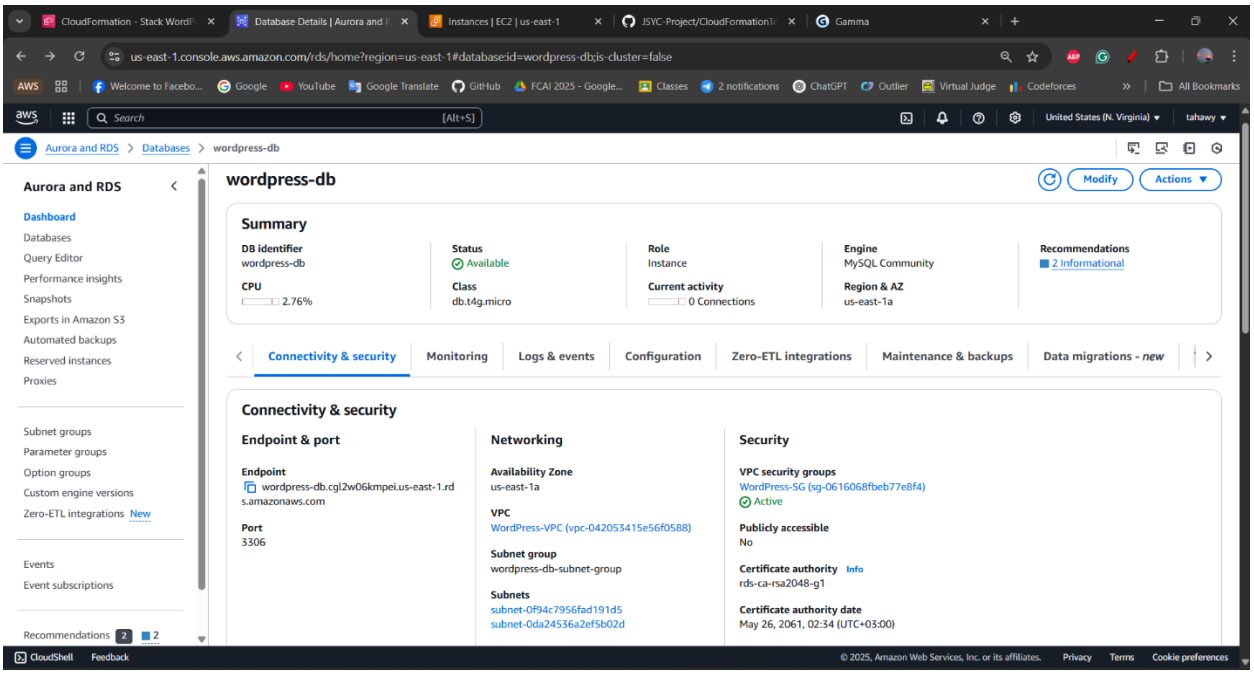
Instances:

The instances created that the WordPress Site has deployed in them either manually or using CloudFormation.



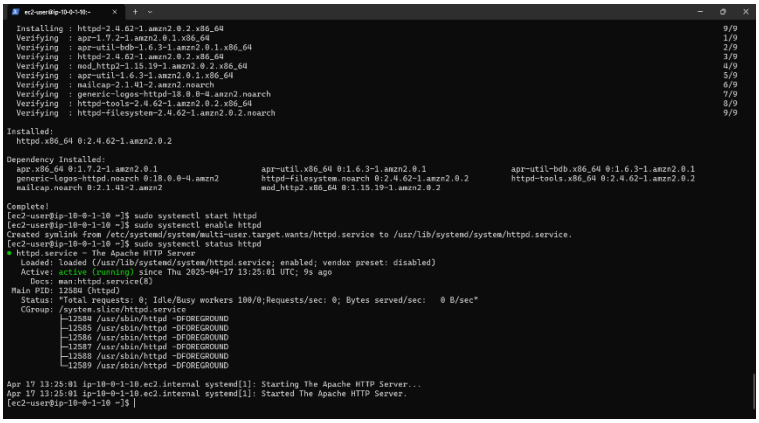
RDS:

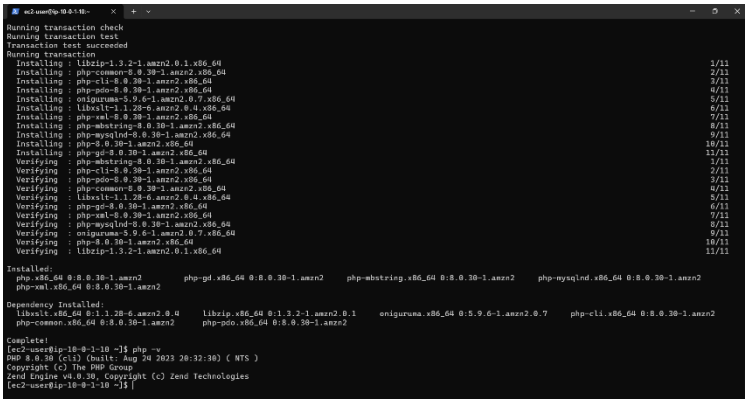
This is the database that was created for WordPress because it is a dynamic content management system that stores and retrieves content from a database. Without a database, WordPress cannot manage content or maintain user data, making it non-functional. It’s connected to the two private subnets.

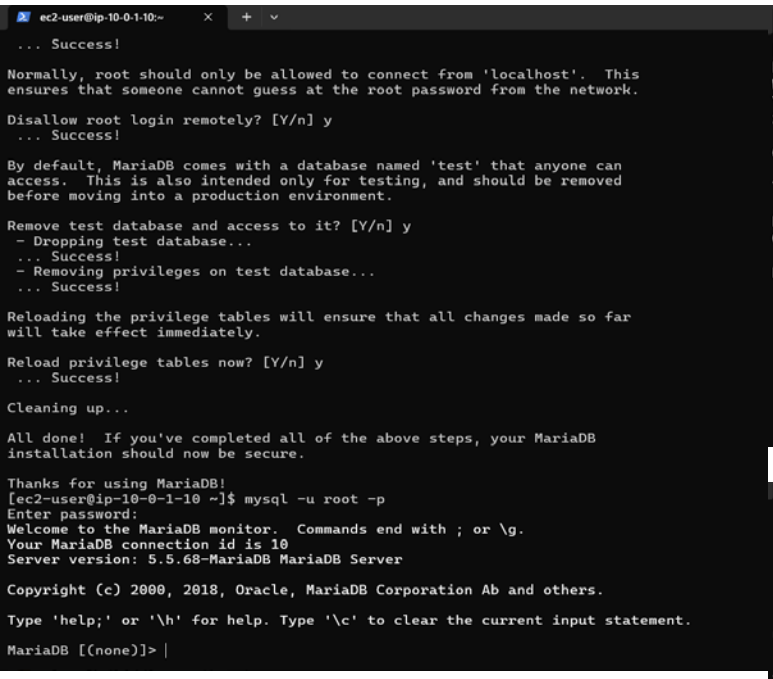


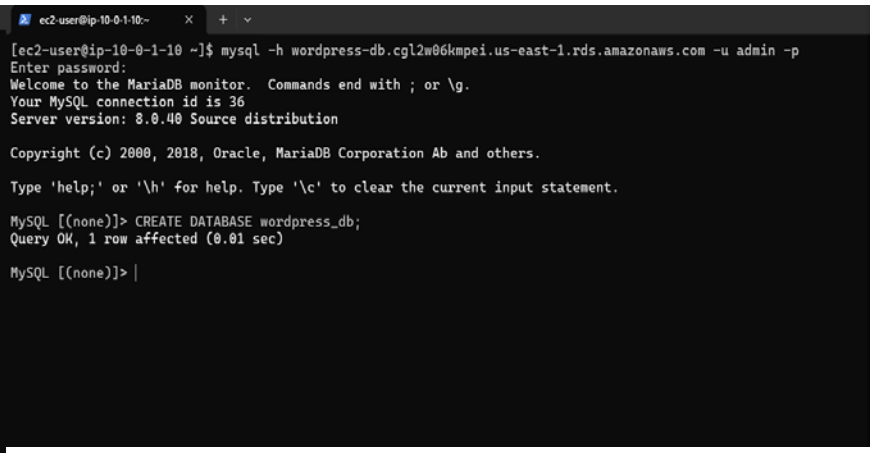
Installation of WordPress in Instance Manually:

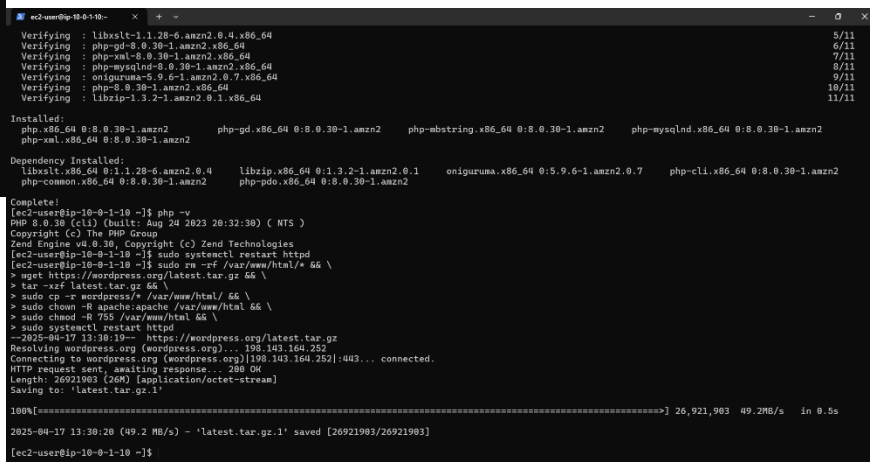
Those are screenshots of installing Apache and PHP which are required first to install WordPress later. Also, the installation of the DB which is required for WordPress.



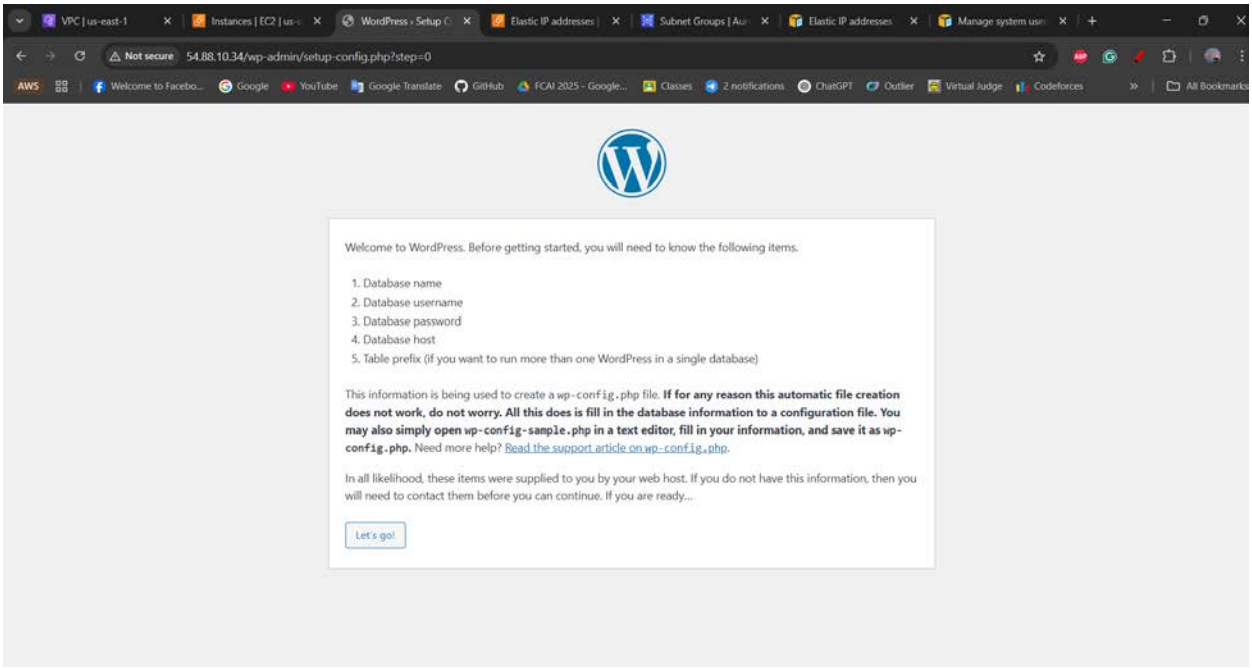


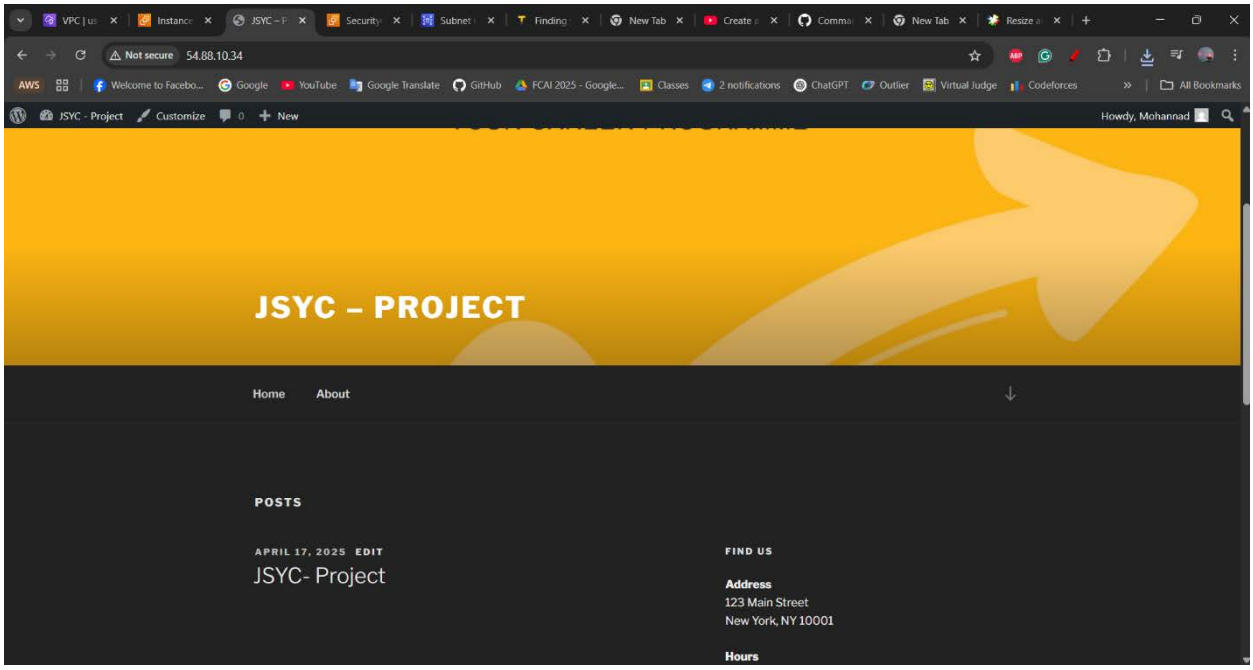

ec2-user@ip-10-0-1-10:~
... Success!
Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n] y
... Success!
By default, MariaDB comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.
Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
Reloading the privilege tables will ensure that all changes made so far will take effect immediately.
Reload privilege tables now? [Y/n] y
... Success!
Cleaning up...
All done! If you've completed all of the above steps, your MariaDB installation should now be secure.
Thanks for using MariaDB!
[ec2-user@ip-10-0-1-10 ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 5.5.68-MariaDB MariaDB Server
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>


ec2-user@ip-10-0-1-10:~
[ec2-user@ip-10-0-1-10 ~]$ mysql -h wordpress-db.cgl2m06kmpei.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 8.0.40 Source distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MySQL [(none)]> CREATE DATABASE wordpress_db;
Query OK, 1 row affected (0.01 sec)
MySQL [(none)]>


ec2-user@ip-10-0-1-10:~
Verifying : libxslt-1.1.28-6.amzn2.0.4.x86_64 5/11
Verifying : php-gd-8.0.30-1.amzn2.x86_64 6/11
Verifying : php-xml-8.0.30-1.amzn2.x86_64 7/11
Verifying : php-mysqlnd-8.0.30-1.amzn2.x86_64 8/11
Verifying : oniguruma-5.9.6-1.amzn2.0.7.x86_64 9/11
Verifying : php-8.0.30-1.amzn2.x86_64 10/11
Verifying : libzip-1.3.2-1.amzn2.0.1.x86_64 11/11
Installed:
php.x86_64 0:8.0.30-1.amzn2 php-gd.x86_64 0:8.0.30-1.amzn2 php-mbstring.x86_64 0:8.0.30-1.amzn2 php-mysqlnd.x86_64 0:8.0.30-1.amzn2
php-xml.x86_64 0:8.0.30-1.amzn2
Dependency Installed:
libxslt.x86_64 0:1.1.28-6.amzn2.0.4 libzip.x86_64 0:1.3.2-1.amzn2.0.1 oniguruma.x86_64 0:5.9.6-1.amzn2.0.7 php-cli.x86_64 0:8.0.30-1.amzn2
php-common.x86_64 0:8.0.30-1.amzn2 php-pdo.x86_64 0:8.0.30-1.amzn2
Complete!
[ec2-user@ip-10-0-1-10 ~]$ php -v
PHP 8.0.30 (cli) (built: Aug 24 2023 20:32:30) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.0.30, Copyright (c) Zend Technologies
[ec2-user@ip-10-0-1-10 ~]$ sudo systemctl restart httpd
[ec2-user@ip-10-0-1-10 ~]$ sudo rm -rf /var/www/html/* && \
> wget https://wordpress.org/latest.tar.gz && \
> tar -xzf latest.tar.gz && \
> sudo cp -r wordpress/* /var/www/html/ && \
> sudo chown -R apache:apache /var/www/html && \
> sudo chmod -R 755 /var/www/html && \
> sudo systemctl restart httpd
--2025-04-17 13:30:19-- https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 26921903 (26M) [application/octet-stream]
Saving to: 'latest.tar.gz.1'
100%[=>] 26,921,903 49.2MB/s in 0.5s
2025-04-17 13:30:20 (49.2 MB/s) - 'latest.tar.gz.1' saved [26921903/26921903]
[ec2-user@ip-10-0-1-10 ~]$

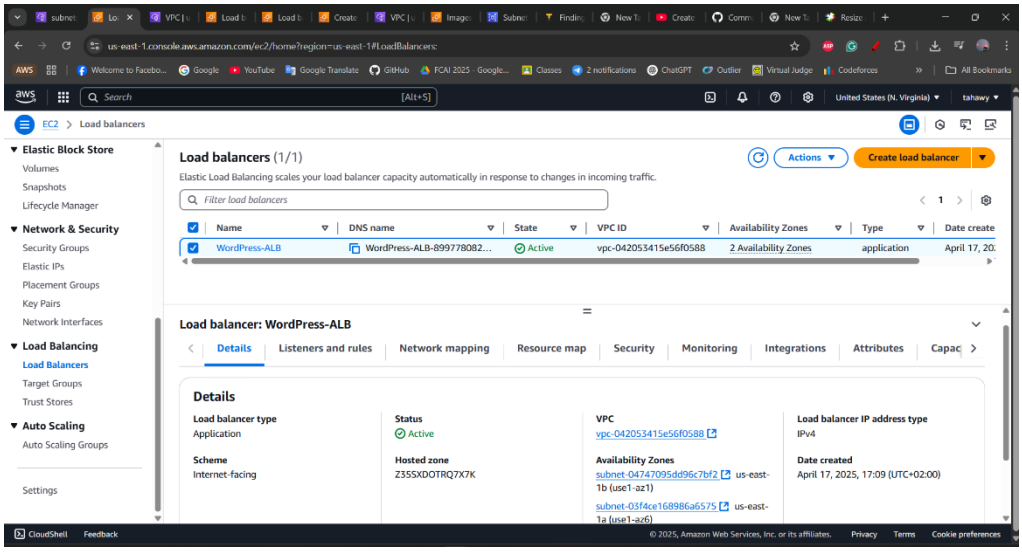
Installation of WordPress from Website:

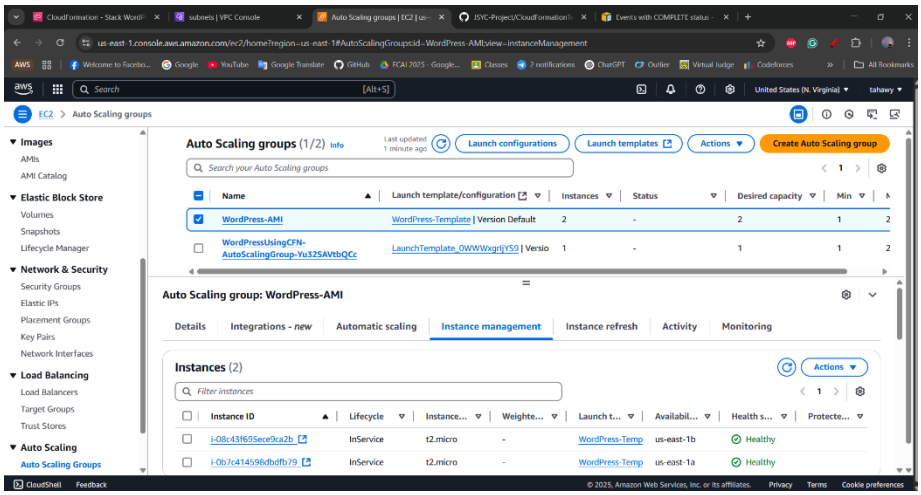


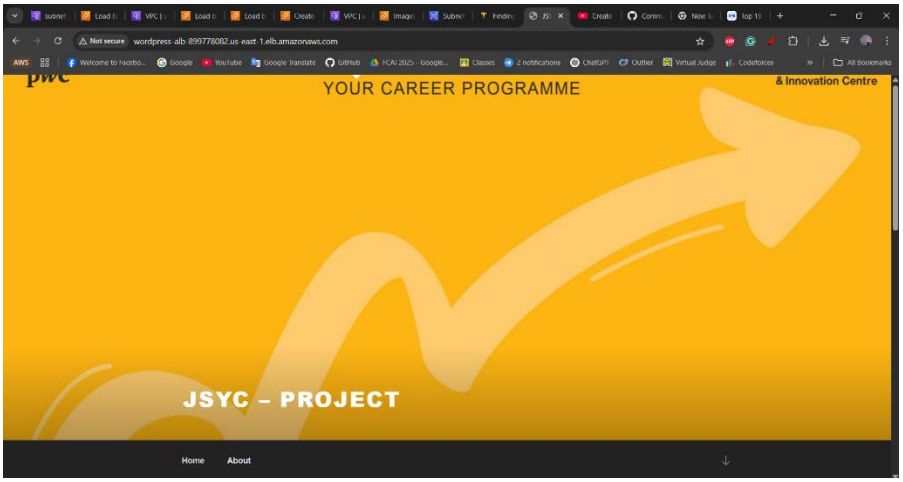


ALB & ASG:

ALB helps in distributing incoming traffic across multiple EC2 instances, Ensuring high availability and fault tolerance. While ASG automatically adds or removes EC2 instances based on demand, Ensuring the right number of instances are always running.

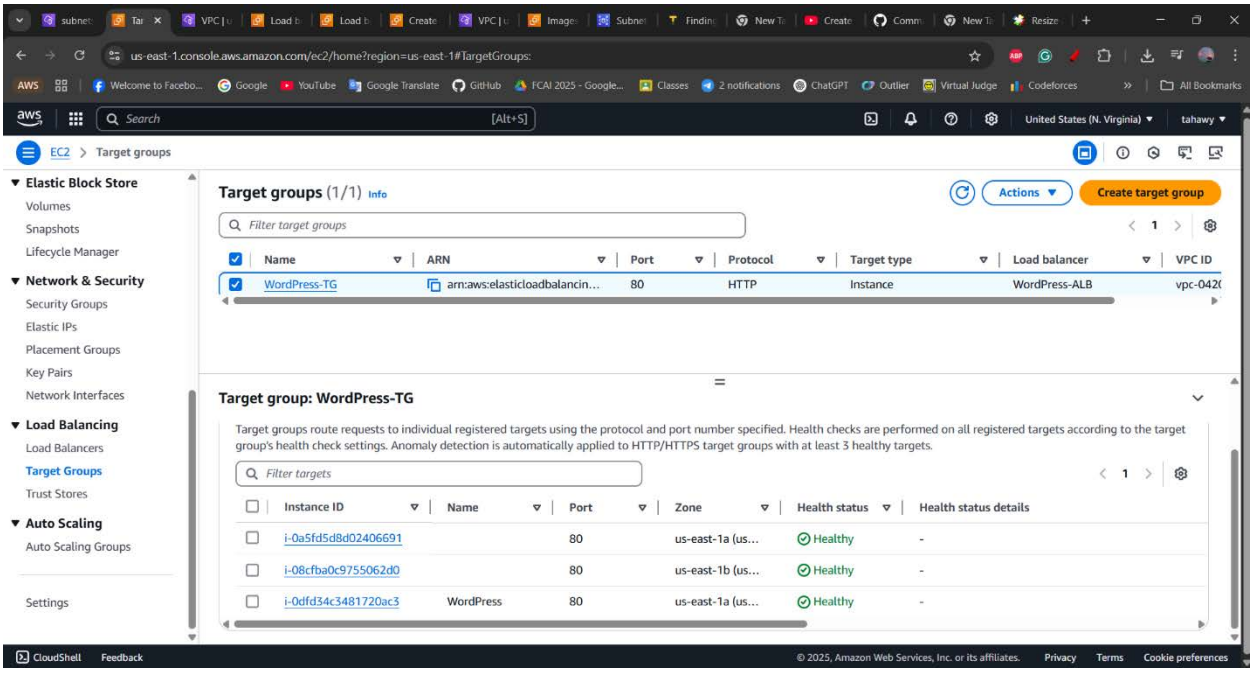






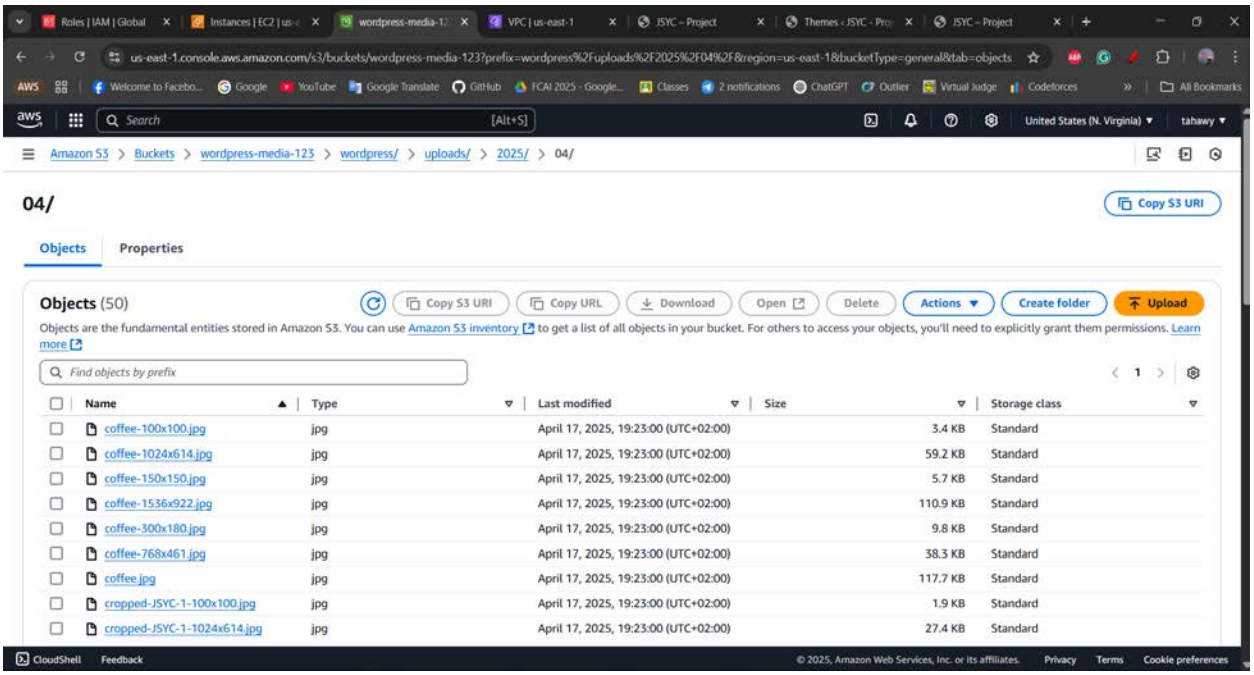
Accessing Website from ALB DNS

This is a Target Group screenshot, Target Group routes traffic from the ALB to registered EC2 instances or other targets.



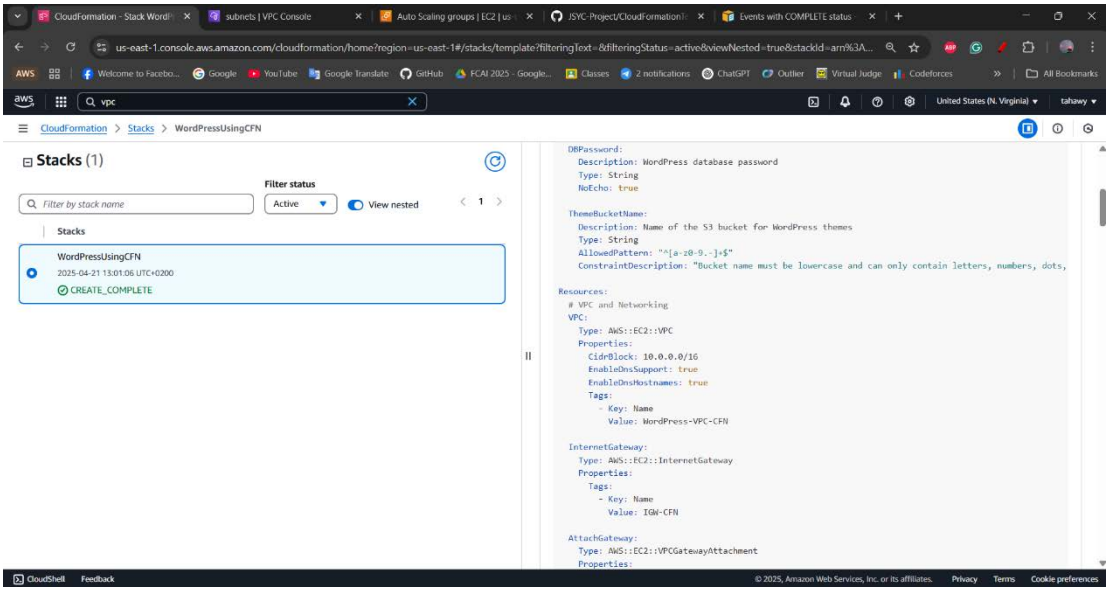
S3 Bucket:

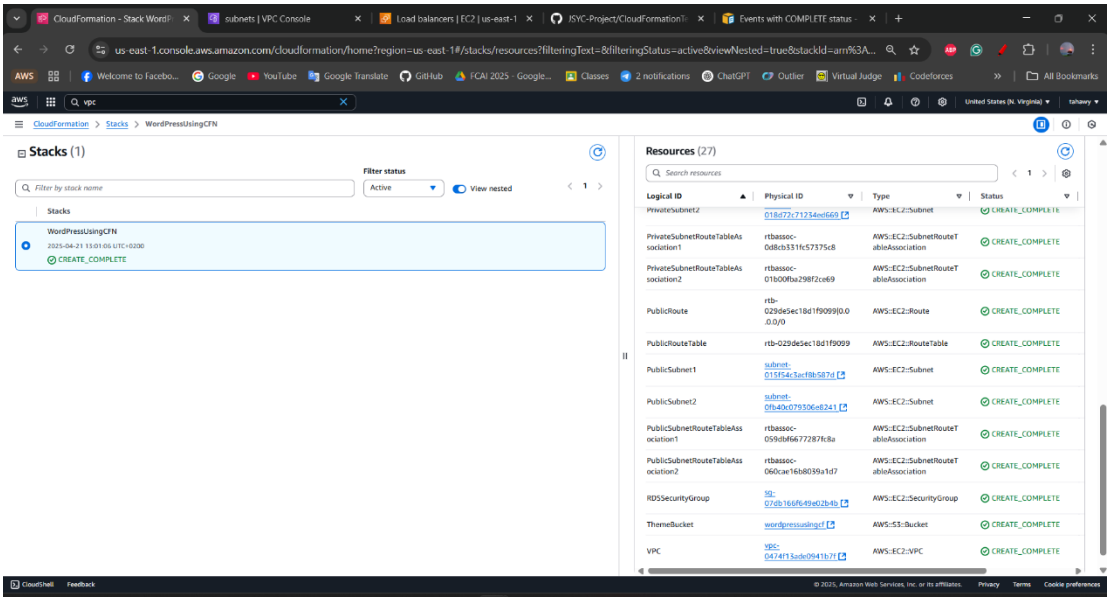
Helpful in Storing static website content like HTML, CSS, JavaScript, and media files. But here I saved all the website photos.



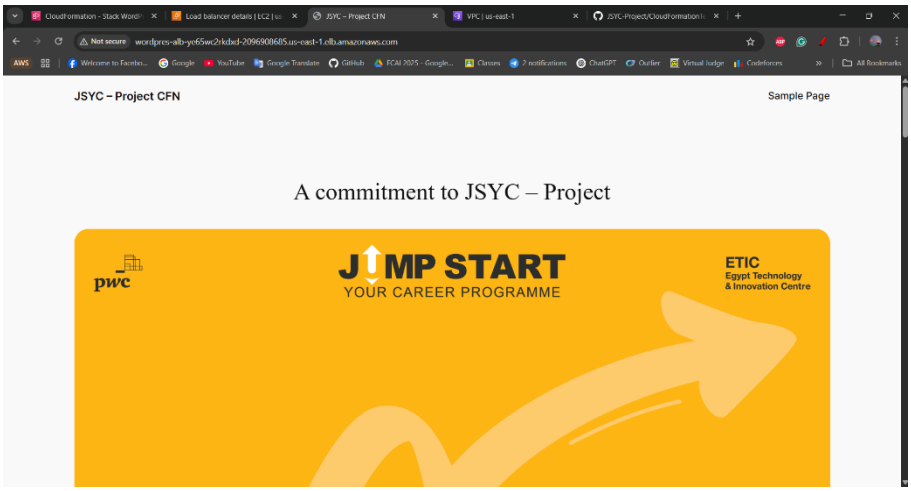
CloudFormation:

I switched to using CloudFormation after realizing that creating and deleting AWS resources manually was time-consuming and error prone. With CloudFormation, I can define all my infrastructure in a single template, which automates the entire process whether it's launching or tearing down a complete stack. This not only saves a lot of time but also ensures consistency and reduces the chances of missing any resources.





Using CFN ALB DNS:



Using CFN Instance IP Address: