

# ENPM818N - Mid Term Project Report

## Group 4:

Aditi Lnu, Rishabh Goel, Solomon Bezabih and Tanvi Kanchan

## Table of Contents

SR NO.	TITLE	PAGE NO
1	Introduction	1
2	Architecture of the Project	2
	2.1 Virtual Private Cloud (VPC)	3
	2.2 Relational Database Services (RDS)	4
	2.3 Auto Scaling Groups	7
	2.4 Load Balancer	9
	2.5 Availability Zones	12
	2.6 Security Groups	12
	2.7 Route53	14
3	Cloud Formation Templates	15
4	Security Measures	28
5	Optimization Measures	30
6	Testing and Monitoring	34
7	Cost Analysis	36
8	Conclusion	37
9	Resources	38

## **1. INTRODUCTION:**

The group project report details the designing, developing and deployment phases of building our scalable and secure e-commerce platform site (**enpmprojectgroup4.shop**) using Amazon Web Services (AWS). The project is developed in a way that the system can handle fluctuating user traffic, protect sensitive customer data and also deliver content of our e-commerce website efficiently to users globally.

After thorough research and exploring AWS resources, we have implemented the best practices of several AWS services, each of which contribute to the e-commerce website's reliability, security and performance. The services incorporated for our project are AutoScaling, Load Balancer, Amazon RDS, Amazon WAF (Web Application Firewall), CloudFront for CDN, Amazon KMS and Cloud Watch to name a few.

Auto-scaling is used to dynamically adjust the EC2 instances based on the traffic spike to maintain optimal performance. Load balancers distribute traffic amongst the EC2 instances thus enhancing fault tolerance. Amazon RDS hosts our MySQL database which helps with efficient customer and product management. WAF is set up to protect the website against common web threats like Cross-Site Scripting and SQL injection.

Further, to decrease the latency in delivering static assets to users worldwide, Amazon's CDN service, Cloudfront, is implemented. Encryption is one of the core features of the project. To encrypt data at rest, Amazon KMS feature is used and SSL is enabled to secure connections between the e-commerce platform and the RDS MySQL instance.

So to summarize, the objectives we accomplished through the project are as follows:

- Deployed a secure and scalable e-commerce platform using AWS services.
- Implemented auto-scaling to efficiently handle varying levels of traffic.
- Ensured data security through encryption and secure database management.
- Optimized content delivery for global users.
- Implemented robust security measures to protect against common web vulnerabilities.

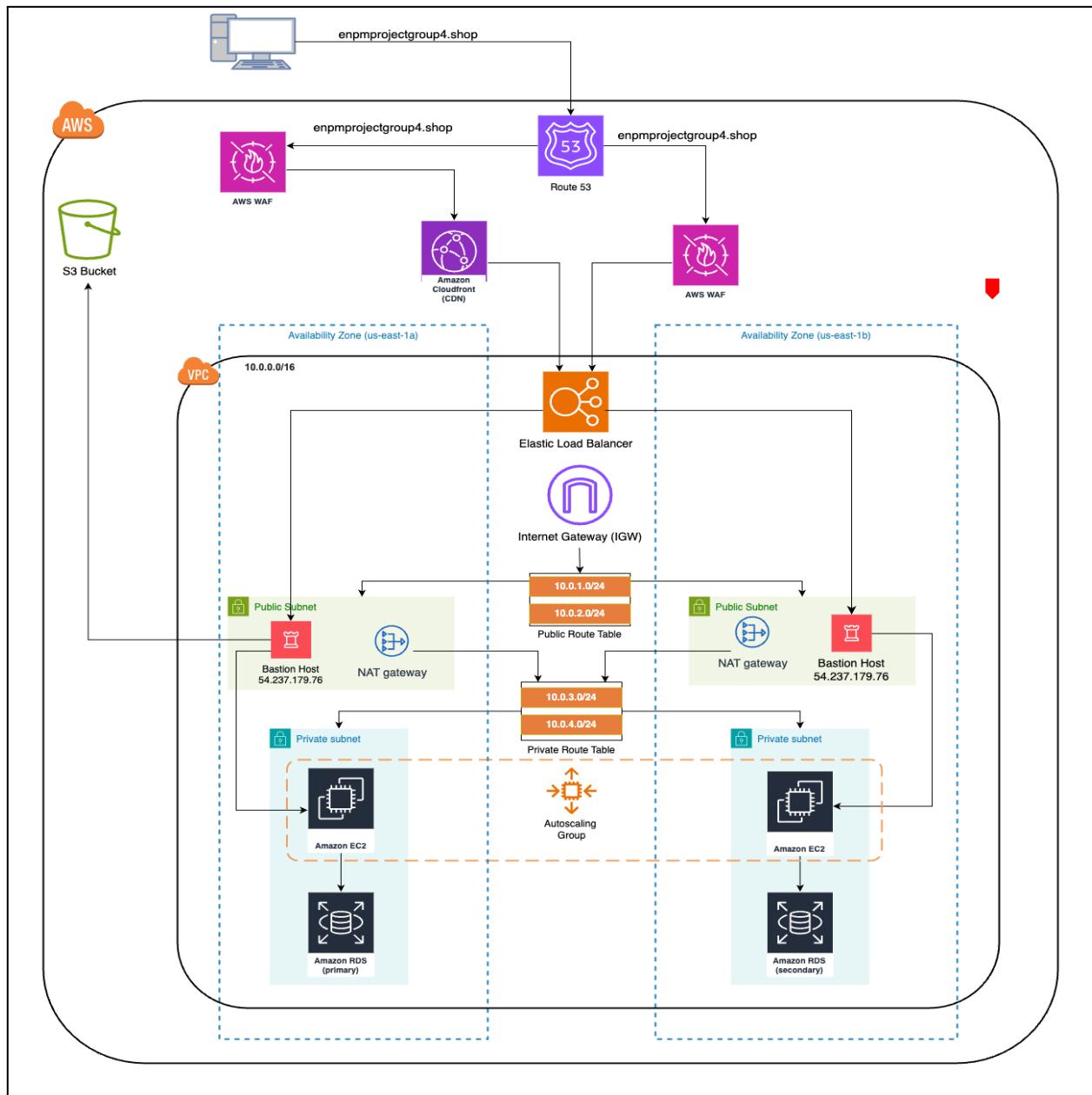
Through rigorous testing using Apache Jmeter, it can be verified that our platform fulfills the project objectives, ensuring reliability, security, and optimized performance.

## 2. ARCHITECTURE OF THE PROJECT:

The cloud-based architecture is designed to support a highly available, secure and scalable application on AWS for our e-commerce platform: *enpmprojectgroup4.shop*. The architecture makes use of various foundational components like Virtual Private Cloud (VPC), Elastic Load Balancing, Auto Scaling, Route 53, Web Application Firewall (WAF), and a Cloudfront Content Delivery Network (CDN) for optimized global performance.

The VPC provides an isolated network resource, the Load Balancing allows for even traffic distribution across the available resources to maintain availability, an Auto Scaling group for dynamic resource management, and a relational database service (RDS) for data storage. Resources are distributed across public and private subnets within multiple Availability Zones (AZs) for fault tolerance and effective resource management. The Route53 service is also enabled to manage the DNS settings for *enpmprojectgroup4.shop*. Additionally, the WAF and Cloudfront services are also utilized for security and optimization purposes.

The following diagram illustrates how the above mentioned services are integrated to form our architecture.



As seen in the diagram above, the workflow of our architecture can be explained as follows:  
An internet-user will search for **enpmprojectgroup4.shop** on their web browser, triggering a DNS lookup through **Route53**. Route53 then directs the request to **CloudFront**, which, integrated with **WAF**, analyzes the traffic for any malicious pattern before allowing it to proceed. Requests for static assets are directly served through CloudFront's cache, while dynamic requests are forwarded to the **Load Balancer** (LB). The LB distributes these incoming requests to the EC2 instances. These instances scale up or down based on demand. For any data-related operations required by the backend EC2 instances, the requests are processed through **Amazon RDS**. The instance is configured to handle primary database operations in the primary availability zone, with a secondary availability zone instance for failover support. **CloudWatch** monitors instance CPU utilization and triggers scaling policies in the **Auto Scaling Group** as needed.

## 2.1 Virtual Private Cloud (VPC):

The screenshot shows the AWS VPC Details page for a VPC named 'vpc-097fa1faab70e4ea9 / MyVPC'. The 'Resource map' section is highlighted with a red box. It displays the following components:

- VPC:** Your AWS virtual network (MyVPC)
- Subnets (4):** Subnets within this VPC, split across us-east-1a and us-east-1b, including PublicSubnet1, PrivateSubnet1, PublicSubnet2, and PrivateSubnet2.
- Route tables (3):** Route network traffic to resources, including rtb-0c7f4b465a0529f00, PrivateRouteTable, and PublicRouteTable.
- Network connections (2):** Connections to other networks, including igw-0ab4a2d3fe13f15fc and nat-058d7c79876136c6c.

The VPC has been allocated the IPv4 CIDR block of **10.0.0.0/16**, which provides a large address space for assigning IP addresses to various subnets within the VPC. The DNS hostname and resolution options are enabled to facilitate smooth internal name resolution, enhancing connectivity among instances.

VPC includes four subnets, split across two Availability Zones (us-east-1a and us-east-1b). This approach provides high availability and fault tolerance:

- Public Subnet 1 (**10.0.1.0/24**) and Public Subnet 2 (**10.0.2.0/24**): These subnets are intended for resources that require internet access, such as the Application Load Balancer (ALB) and the Bastion Host. The public subnets are configured to automatically assign public IP addresses to resources upon launch.
- Private Subnet 1 (**10.0.3.0/24**) and Private Subnet 2 (**10.0.4.0/24**): These are internal subnets that have sensitive resources such as EC2 instances within the Auto Scaling Group (ASG) and the RDS databases. These subnets are isolated from direct internet access to ensure data security.

The VPC consists mainly of three route tables to manage traffic routing within the network:

- Public Route Table: This table routes outbound traffic from public subnets to the internet through the Internet Gateway (**igw-0ab4a2d3fe13f15fc**). This allows resources in public subnets to communicate with external networks.

- **Private Route Table:** This table directs outbound internet traffic from public subnets through a Network Address Translation (**nat-058d7c79876136c6c**) Gateway, which provides a secure way for internal resources to access the internet without exposing them to public traffic.
- **Main Route Table (**rtb-0c7f4b465a0529f00**):** This route table handles default network routing for the VPC and is linked with the primary network ACL (Access Control List) for additional security control.

## 2.2 Relational Database Service (RDS):

The screenshot shows the AWS RDS console for an e-commerce MySQL database instance. The instance is currently available and has a db.t3.medium configuration. It is running MySQL Community engine in the N. Virginia region (us-east-1a). The configuration tab is selected, displaying detailed settings such as instance class, storage, and performance insights. Key configuration details include:

- Configuration:** DB instance ID: e-commerce, Engine version: 8.0.39, RDS Extended Support: Enabled, DB name: ecommerce\_1, License model: General Public License, Option groups: default.mysql8.0 (In sync), Amazon Resource Name (ARN): arnawsrdsus-east-1:314147830127:db:e-commerce, Resource ID: db-OYG7TSARM76V72KMUF3JG5LU4, Created time: October 28, 2024, 23:19 (UTC-04:00).
- Storage:** Storage type: General Purpose SSD (gp2), Storage: 20 GiB, Provisioned IOPS: -, Storage throughput: -, Storage autoscaling: Enabled, Maximum storage threshold: 100 GiB.
- Performance Insights:** Performance Insights enabled: Turned on, AWS KMS key: aws/rds, Retention period: 7 days.

The image illustrates the configuration details for our RDS instance, tailored for our e-commerce platform. With features like autoscaling, multi-AZ deployment, and performance insights enabled, our database is optimized for high availability, reliability, and efficient performance under varying load conditions.

### **2.2.1 RDS Configuration Details:**

- DB Instance ID: e-commerce
- Engine and Version: MySQL engine, version 8.0.39
- Instance Class: db.t3.medium, this instance has 2 vCPUs and 4 GB of RAM

### **2.2.2 Storage and Scalability:**

- Storage Type and Size: General Purpose SSD (gp2) storage with an allocated size of 20 GiB
- Storage Autoscaling: Autoscaling is enabled, allowing the storage capacity to increase automatically up to a maximum threshold of 100 GiB
- Encryption: AWS Key Management Service (KMS)

### **2.2.3 High Availability and Reliability:**

- Multi-AZ Deployment: primary instance in us-east-1a and a secondary standby instance in us-east-1b
- Backup and Retention: Backups are enabled with a default retention period of 7 days, safeguarding data recovery options in case of unexpected data loss.

### **2.2.4 Access and Monitoring:**

- Master User: Accessible by admin user
- Performance Insights: enabled

The above image outlines the connectivity and security settings for our RDS instance, designed to ensure secure and controlled access within our ecommerce platform's infrastructure. It highlights critical components like the VPC configuration, endpoint accessibility, and security measures, ensuring robust protection and optimal performance.

## 2.2.5 Connectivity:

- Endpoint and Port: The database can be accessed using the endpoint (**e-commerce.cchwcgordfrx.us-east-1.rds.amazonaws.com**) on port **3306**.
- Networking: The RDS instance is hosted within the VPC (**vpc-097fa1faab70e4ea9**) in the us-east-1a Availability Zone.
- Subnet Group: The database is associated with a subnet group (**rds-28-10-2024-dbsubnetgroup-34yj4j1aexwg**), which includes private subnets.

## 2.2.6 Security:

- VPC Security Groups: RDS instance is protected by a dedicated security group (**SecurityGroup-28-10-2024-RDSSecurityGroup-wMooEJPvW6p**), which strictly controls inbound and outbound traffic.
- Access Control: The database is configured as "**Not Publicly Accessible**", which implies that it is only accessible to resources within the VPC.
- Encryption: This layer supports secure data transmission, with the endpoint being managed within the VPC and leveraging security groups.

The screenshot shows the AWS RDS console with the following details:

- Instance Type:** MySQL 8.0
- Storage:** 1000 MB
- Engine:** MySQL 8.0.39
- Region:** us-east-1a
- Security Group:** SecurityGroup-28-10-2024-RDSSecurityGroup-wMooEJPYw6p (sg-02a5617a9bf4fe78f)
- Replication:** None (1 entry: e-commerce, Instance, us-east-1a)
- Proxies:** None (0 entries)
- Security Group Rules:** 4 entries (CIDR/IP - Inbound for 0.0.0.0/0)

The above image shows that the instance has been configured in us-east-1a as a standalone instance for this setup, without cross-region replication.

```
[root@ip-10-0-3-137:/var/www/html/includes# mysql -h e-commerce.cchcwgordfrx.us-east-1.rds.amazonaws.com -u admin -p
[Enter password]:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42454
Server version: 8.0.39 Source distribution

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

[mysql> show databases;
+-----+
| Database      |
+-----+
| ecommerce_1   |
| information_schema |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.00 sec)

[mysql> use ecommerce_1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
[mysql> show tables;
+-----+
| Tables_in_ecommerce_1 |
+-----+
| admin_table           |
| brands                |
| card_details          |
| categories             |
| orders_pending         |
| products               |
| user_orders            |
| user_payments          |
| user_table              |
+-----+
9 rows in set (0.01 sec)

mysql>
```

The above image provides an overview of the database structure within the Amazon RDS instance for the **ecommerce\_1** database, demonstrating the tables that form the core of the e-commerce application. This database schema includes essential tables such as **admin\_table**, **brands**, **card\_details**, **categories**, **orders\_pending**, **products**, **user\_orders**, **user\_payments**, and **user\_table**.

## New User Registration

Username

Email

User Image

Password

Confirm Password

Address

Mobile

Already have an account? [Login](#)

user_id	username	user_email	user_password	user_image	user_ip	user_address	user_mobile
1	abdo	abdo@gmail.com	\$2y\$10\$5synbyf6t7/r2Zm1lkvghu.JGBKG7z2LULzuJ9jI5P03rGZDH..	new logo after Edit1920.png	11	Cairo	120456789
2	test	test@test.com	\$2y\$10\$1fparM1.EFZD70Fhmxu02L3mXb5P8YAK0KrwuxanVxK2jy0	Screenshot 2024-10-13 at 7:03:11 PM.png	50..237.13.62	21123121	2404678423
3	testuser	user@test.com	\$2y\$10\$5iusb40LFtnlbcNtV6NUHq0xkElC1u4Bz2Fpjpj1nhj20xGc.Gt/HW	Screenshot 2024-10-13 at 7:03:11 PM.png	50..237.13.62	adsasd	121232121
4	tamvittest	test@gmail.com	\$2y\$10\$hGWvS7w1jSYRQ..62Mz/30w1h7o7a79noplem.unS/7IVqdF.8PP0	image.jpg	69.137.233.24	College Park, MD	1234567890
5	test1234	raviv86796@ruhtan.com	\$2y\$10\$MSpb0/eBF1plLn7jGYx3z0mlPKz7lszk4bjCDzo0BKwtLouKSwT6	image.jpg	69.137.233.24	College Park, MD	1234567890
6	test1212	sdhdsjh@dhjsids.com	\$2y\$10\$eQNUYT50b4rClFRardeGerrkHTB4oJ6Fomp12JyOTthQkSDG3rC	bg.png	50..237.13.62	dasd	123123123
7	Aditi	aditi@gmail.com	\$2y\$10\$6DEKv.GM0t8u4NDRxxSCzeUrzzED..60..JTrkgheNyRVY1wV02vEcC	test.png	50..237.13.57	College Park, MD	240 110 1100

Basic create, read, update and delete (CRUD) operations on the database via the e-commerce platform.

### 2.3 Auto Scaling Groups:

The Auto-Scaling group for the private instances. Our application is based on the private instances and we route it through Bastion Host for added security, hence the private instances need to be scaled based on the traffic spike. Based on our requirements, we have defined the parameters as following:

- **Desired capacity - 2**
- **Minimum Capacity - 1**
- **Maximum Capacity - 5**

The screenshot shows the AWS EC2 Auto Scaling Groups console. The main page displays the details for an Auto Scaling group named 'autoscale-PrivateAutoScalingGroup-hhEslv9Ef3el'. Key information includes:

- Group details:** Desired capacity is set to 2, with a minimum of 1 and a maximum of 5. The status is 'Updating capacity'.
- Launch template:** The launch template used is 'lt-001e9846757927bde' (EcommercePrivateLaunchTemplate). It specifies an AMI ID ('ami-06a9cf407e797e782'), instance type ('t3.micro'), and security group ('sg-0e990e7782038824f'). The owner is 'arn:aws:iam::314147830127:root'.
- Network:** The availability zones are 'us-east-1a, us-east-1b', and the subnet ID is 'subnet-067c8ab2de0d93fc4, subnet-0ffe04c1985c6fa0f'.

The Auto-scaling group follows two simple dynamic scaling policies. If the CPU utilization **exceeds the 50% threshold for 1 consecutive period of 60 seconds**, that is 1 minute, and an **instance is added** to the available resources. Similarly, if the CPU utilization is **less than 10%** for 1 minute, an **instance is removed** from the pool of available resources. We have ensured a cooldown period of 2 minutes after each scaling action to prevent flapping[].

The screenshot shows the AWS EC2 Auto Scaling Groups console. The main page displays the configuration for an Auto Scaling group named 'Autoscale-PrivateAutoScalingGroup-wEhQNoUAprVD'. Key sections include:

- Dynamic scaling policies:** Two policies are listed:
  - Autoscale-PrivateScaleDownPolicy-NYlv5xkqkV5i**: Policy type is Step scaling. It triggers when CPUUtilization >= 10 for 60 seconds. The action is 'Remove 1 capacity units when 10 >= CPUUtilization > -infinity'.
  - Autoscale-PrivateScaleUpPolicy-DZnlBmUPj5de**: Policy type is Step scaling. It triggers when CPUUtilization <= 50 for 60 seconds. The action is 'Add 1 capacity units when 50 <= CPUUtilization < +infinity'.
- Predictive scaling policies:** One policy is listed:
  - Autoscale-ScaleDownAlarm-bibu0eVl2n3o**: Triggers when CPUUtilization >= 10 for 60 seconds. The metric dimensions are 'AutoScalingGroupName = Autoscale-PrivateAutoScalingGroup-wEhQNoUAprVD'.
  - Autoscale-ScaleUpAlarm-zXewcsJTvIG**: Triggers when CPUUtilization <= 50 for 60 seconds. The metric dimensions are 'AutoScalingGroupName = Autoscale-PrivateAutoScalingGroup-wEhQNoUAprVD'.

The scaling policies can be checked in real-time through the activity history. The instances scale up when an unhealthy instance is detected and it scales down when traffic detected is low on the instance.

The screenshot shows the AWS CloudWatch Metrics interface. The top navigation bar includes 'Services' (selected), 'Activity' (highlighted in blue), 'Automatic scaling', 'Instance management', and 'Monitoring'. The top right shows 'N. Virginia' and a timestamp 'rgoel22-ENPM665-0201-2024'. Below the navigation is a search bar 'Filter notifications' and a 'Send to' dropdown set to 'On instance action' with 'Email1' selected. The 'Launch, Terminate, Fail to launch, Fail to terminate' option is also visible. The main area is divided into 'Activity notifications (1)' and 'Activity history (7)'. The 'Activity notifications' section has a single entry: 'Terminating EC2 instance: i-0ebdf29cf1b54ea9 - Waiting For ELB Connection Draining.' The 'Activity history' section lists seven entries from October 29, 2024, detailing various EC2 instance events like launching, terminating, and health checks.

## 2.4 Load Balancer:

The image below presents the EcommerceLoadBalancer, designed to increase the availability and performance of our e-commerce platform by evenly distributing incoming traffic. It operates across two Availability Zones (**us-east-1a** and **us-east-1b**), further enhancing availability and resilience by routing traffic between these zones.

The screenshot shows the AWS CloudWatch Metrics interface. The top navigation bar includes 'Services' (selected), 'Search' (highlighted in blue), and '[Option+S]'. The top right shows 'N. Virginia' and a timestamp 'rgoel22-ENPM665-0201-2024'. Below the navigation is a search bar 'Filter listeners' and a table for 'Listeners and rules (2)'. The table has columns for 'Protocol/Port', 'Default action', 'Rules', 'ARN', 'Security policy', 'Default SSL/TLS certificate', 'mTLS', and 'Trust store'. The first rule for 'HTTP:80' is 'Redirect to HTTPS://#[host]:443/#[path]?#[query]' with a status of '1 rule'. The second rule for 'HTTPS:443' is 'Forward to target group' with a status of '1 rule'. At the bottom of the page are links for 'CloudShell' and 'Feedback', and a copyright notice '© 2024, Amazon Web Services, Inc. or its affiliates.' along with 'Privacy', 'Terms', and 'Cookie preferences'.

The load balancer has two listeners: an HTTP listener on **port 80** and an HTTPS listener on **port 443**, respectively.

- HTTP listener captures unencrypted requests and permanently redirects them to HTTPS using a 301 status code, ensuring all traffic is secure.
- HTTPS listener handles encrypted requests, forwarding them to the **EcommerceTargetGroup**, where they are distributed across EC2 instances in the Auto Scaling Group. Configured with SSL/TLS certificates for the **enpmprojectgroup4.shop** domain, the HTTPS listener ensures data privacy and integrity. Additionally, a security policy (**ELBSecurityPolicy-TLS-1-2-2021-07**) applies secure protocols supporting a secure connection for all user traffic.

The load balancer, with the dual listener configuration and security policies, provides a robust approach to handling user traffic. It ensures that users access the application through secure connections, while efficiently balancing the load across multiple backend instances.

The screenshot shows the AWS CloudFormation console for the 'EcommerceLoadBalancer' stack. The 'Details' tab is selected, displaying information about the VPC, subnets, and security groups. The 'Network mapping' tab is also visible, showing the VPC configuration and subnet mappings. The 'Mappings' section lists two subnets: 'subnet-0e9d61c9771fcdf19' (us-east-1a) and 'subnet-006275fc6a80f2647' (us-east-1b), each assigned from CIDR blocks 10.0.1.0/24 and 10.0.2.0/24 respectively.

The load balancer operates in an internet-facing scheme within the **vpc-097fa1faab70e4ea9** VPC, which has an IPv4 CIDR block of **10.0.0.0/16**. This setup utilizes two Availability Zones—us-east-1a and us-east-1b—each associated with a designated public subnet (**10.0.1.0/24** for us-east-1a and **10.0.2.0/24** for us-east-1b).

The screenshot shows the AWS EC2 Load Balancer console. On the left, the navigation menu includes options like EC2 Dashboard, EC2 Global View, Events, Instances, Images, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and CloudShell. The main content area displays the details of the 'EcommerceLoadBalancer'. Key information includes:

- Load balancer type:** Application
- Status:** Active
- VPC:** vpc-097fa1faab70e4ea9
- Hosted zone:** Z35SXDOTRQ7X7K
- Availability Zones:** subnet-0e9d61c9771fcdf19 us-east-1a (use1-az2) and subnet-006275fc6a80f2647 us-east-1b (use1-az4)
- DNS name:** EcommerceLoadBalancer-1135684291.us-east-1.elb.amazonaws.com (A Record)

The 'Resource map' tab is selected, showing the architecture of the load balancer. It consists of two listeners (HTTPS:443 and HTTP:80), two rules (Priority default: Forward to target group and Priority default: Redirect), one target group (EcommerceTargetGroup), and one target (i-0592f93410ee9fa0e on port 80, marked as Healthy).

The HTTPS listener forwards the secure traffic to a designated target group, **EcommerceTargetGroup**, which contains one registered EC2 instance on port 80. This instance is marked as "**Healthy**", indicating it is ready to handle incoming traffic.

The screenshot shows the AWS EC2 Load Balancer console. The navigation menu is identical to the previous screenshot. The main content area displays the details of the 'EcommerceLoadBalancer'. Key information includes:

- Load balancer type:** Application
- Status:** Active
- VPC:** vpc-097fa1faab70e4ea9
- Hosted zone:** Z35SXDOTRQ7X7K
- Availability Zones:** subnet-0e9d61c9771fcdf19 us-east-1a (use1-az2) and subnet-006275fc6a80f2647 us-east-1b (use1-az4)
- DNS name:** EcommerceLoadBalancer-1135684291.us-east-1.elb.amazonaws.com (A Record)

The 'Security' tab is selected, showing the security groups associated with the load balancer. There is one security group listed:

Security Group ID	Name	Description
sg-052b49f23899e402d	SecurityGroup...	Allow HTTP and HTTPS traffic to the ALB

The **EcommerceLoadBalancer** is secured by a dedicated security group (**sg-052b49f23899e402d**), which is configured to allow HTTP and HTTPS traffic. This security group plays the role in managing and restricting inbound traffic, ensuring that only web traffic on **port 80** (HTTP) and **port 443** (HTTPS) can reach the load balancer.

## 2.5 Availability Zones:

The Availability Zones in the US East (*N. Virginia*) region provide a reliable, distributed infrastructure for our application. We use zones ***us-east-1a*** and ***us-east-1b*** to enhance fault tolerance and ensure high availability.

The screenshot shows the AWS EC2 Free Tier page. It displays various resource counts: Instances (running) 2, Auto Scaling Groups 1, Capacity Reservations 0, Dedicated Hosts 0, Elastic IPs 1, Instances 6, Key pairs 3, Load balancers 1, Placement groups 0, Security groups 10, Snapshots 7, Volumes 2. On the right, it shows EC2 Free Tier offers in use (3), end-of-month forecast, and offers exceeded. It also includes sections for Service health, Offer usage (monthly), and Account attributes.

## 2.6 Security Groups:

### **2.6.1 ALB**

As the load balancer has to be internet-facing, in the Application Load Balancer security group, we allow ***port 80*** (HTTP) and ***port 443*** (HTTPS) traffic for any IP address (***0.0.0.0/0***). In this way, our application is available globally.

The screenshot shows the AWS Security Groups page for the security group **sg-052b49f23899e402d**. It displays details like Security group name, ID, Description (Allow HTTP and HTTPS traffic to the ALB), and VPC ID. Under Inbound rules, there are two entries: one for port 443 (HTTPS) and another for port 80 (HTTP). The Outbound rules section shows 1 permission entry.

## 2.6.2 RDS:

In the RDS Security group, we only allow database connections from within the VPC. The inbound rule shown in the image indicates the same, all TCP connections from the VPC will be directed to **port 3306**.

The screenshot shows the AWS EC2 Security Groups console. The selected security group is "sg-02a5617a9bf4fe78f - SecurityGroup-28-10-2024-RDSSecurityGroup-wMoolEJPYw6p". The "Details" tab is active, displaying the following information:

Security group name	sg-02a5617a9bf4fe78f - SecurityGroup-28-10-2024-RDSSecurityGroup-wMoolEJPYw6p	Security group ID	sg-02a5617a9bf4fe78f	Description	VPC ID
Owner	314147830127	Inbound rules count	2 Permission entries	Outbound rules count	1 Permission entry

The "Inbound rules" tab is selected, showing one rule:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-00e0d55d561c7db...	IPv4	MySQL/Aurora	TCP	3306	10.0.0.0/16	-

## 2.6.3 Auto-Scaling:

In the Auto Scaling security group, we allow all **port 80** (HTTP) and **port 443** (HTTPS) traffic from the VPC CIDR **10.0.0.0/16** that typically originates from the Application Load Balancer. Additionally, we only allow SSH (port 22) requests specifically from the Bastion Host subnet (CIDR **10.0.1.0/24**) for secure administrative access.

The screenshot shows the AWS EC2 Security Groups console. The selected security group is "sg-0e990e7782038824f - SecurityGroup-28-10-2024-AutoscalingSecurityGroup-bctQtmCcdjai". The "Details" tab is active, displaying the following information:

Security group name	sg-0e990e7782038824f - SecurityGroup-28-10-2024-AutoscalingSecurityGroup-bctQtmCcdjai	Security group ID	sg-0e990e7782038824f	Description	VPC ID
Owner	314147830127	Inbound rules count	4 Permission entries	Outbound rules count	1 Permission entry

The "Inbound rules" tab is selected, showing four rules:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-01479ab950960...	IPv4	SSH	TCP	22	10.0.1.0/24	-
-	sgr-0ad5464b8bfeff1a02	IPv4	MySQL/Aurora	TCP	3306	10.0.0.0/16	-
-	sgr-0ed549e8238595...	IPv4	HTTPS	TCP	443	10.0.0.0/16	-
-	sgr-036869db914000...	IPv4	HTTP	TCP	80	10.0.0.0/16	-

## 2.6.4 Bastion Host:

Bastion Host is used for secure remote management, hence only **port 22** (SSH) access is given to specific IPs which belong to the team members working on the project. In this way, even if there is any unauthorized SSH pem file access, the security group will still restrict access only to those four IPs.

**Details**

Security group name sg-Ofa5ae192296c3ff5 - SecurityGroup-28-10-2024-BastionSecurityGroup-4k2asyBd2bof	Security group ID sg-Ofa5ae192296c3ff5	Description Allow SSH access to Bastion Host	VPC ID vpc-097fa1fab70e4ea9
Owner 314147830127	Inbound rules count 4 Permission entries	Outbound rules count 1 Permission entry	

**Inbound rules (4)**

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
-	sgr-045f3fe5c1cc95ad	IPv4	SSH	TCP	22	98.192.33.217/32	-
-	sgr-0f0bd328bf74fd30b	IPv4	SSH	TCP	22	50.237.13.62/32	-
-	sgr-06d32056b13ba7...	IPv4	SSH	TCP	22	69.137.233.24/32	-
-	sgr-0aa5cb29d8e1c7226	IPv4	SSH	TCP	22	50.237.13.57/32	-

## 2.7 Route 53:

Route53 is the service provided by AWS which allows us to manage DNS for our domain. This is achieved by simply pointing the domain to the AWS nameservers as mentioned in the hosted zone.

<https://dnschecker.org/#NS/enpmprojectgroup4.shop>

**Hosted zone details**

**Records (4) Info**

Record name	Type	Routing policy	Alias	Value/Route traffic to	TTL (s...)	Health ...	Evalua...	Record ID
enpmprojectgroup4.shop	A	Simple	-	duallstack.ecommerceceloadbal...	-	-	Yes	-
enpmprojectgroup4.shop	NS	Simple	-	ns-750.awsdns-29.net. ns-1565.awsdns-03.co.uk. ns-1287.awsdns-32.org. ns-83.awsdns-10.com.	172800	-	-	-
enpmprojectgroup4.shop	SOA	Simple	-	ns-750.awsdns-29.net. awsd...	900	-	-	-
_fd9b2a2280dab2cdf9ee75bcc566567.en...	CNAME	Simple	-	_cf6b54cc89c8df2450e09fb...	300	-	-	-

Additionally an A record (refer record 1) pointing to the Load Balancer and a CNAME record (refer record 4) needed to issue the SSL certificate for the domain.

### 3. CLOUD FORMATION TEMPLATES:

To ensure automated provisioning of infrastructure, consistent and repeatable deployment, we have made use of Cloud Formation features available at AWS. The cloud formation templates were deployed in the order in which there are listed below:

#### **3.1 VPC Template:**

```
AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template to create a VPC with public and private subnets, route tables, and a NAT gateway.

Resources:
# VPC
VPC:
  Type: AWS::EC2::VPC
  Properties:
    CidrBlock: 10.0.0.0/16
    EnableDnsSupport: true
    EnableDnsHostnames: true
  Tags:
    - Key: Name
      Value: "MyVPC"

# Public Subnet 1
PublicSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.1.0/24
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
    MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
      Value: "PublicSubnet1"

# Public Subnet 2
PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.2.0/24
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
    MapPublicIpOnLaunch: true
  Tags:
    - Key: Name
      Value: "PublicSubnet2"
```

```

# Private Subnet 1
PrivateSubnet1:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.3.0/24
    AvailabilityZone: !Select [ 0, !GetAZs '' ]
  Tags:
    - Key: Name
      Value: "PrivateSubnet1"

# Private Subnet 2
PrivateSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref VPC
    CidrBlock: 10.0.4.0/24
    AvailabilityZone: !Select [ 1, !GetAZs '' ]
  Tags:
    - Key: Name
      Value: "PrivateSubnet2"

# Internet Gateway
InternetGateway:
  Type: AWS::EC2::InternetGateway
  Properties: {}

# Attach Internet Gateway to VPC
AttachGateway:
  Type: AWS::EC2::VPCGatewayAttachment
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway

# NAT Gateway EIP
NatEIP:
  Type: AWS::EC2::EIP
  Properties:
    Domain: vpc

# NAT Gateway
NatGateway:
  Type: AWS::EC2::NatGateway
  Properties:
    AllocationId: !GetAtt NatEIP.AllocationId
    SubnetId: !Ref PublicSubnet1 # Deploy in a public subnet for internet access

# Public Route Table

```

```

PublicRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: "PublicRouteTable"

# Private Route Table
PrivateRouteTable:
  Type: AWS::EC2::RouteTable
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: "PrivateRouteTable"

# Route for Public Subnet Internet Access
PublicRoute:
  Type: AWS::EC2::Route
  DependsOn: AttachGateway
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

# Route for Private Subnet Internet Access via NAT
PrivateRoute:
  Type: AWS::EC2::Route
  Properties:
    RouteTableId: !Ref PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NatGateway

# Associate Public Subnet 1 with Public Route Table
PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet1
    RouteTableId: !Ref PublicRouteTable

# Associate Public Subnet 2 with Public Route Table
PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable

```

```

# Associate Private Subnet 1 with Private Route Table
PrivateSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet1
    RouteTableId: !Ref PrivateRouteTable

# Associate Private Subnet 2 with Private Route Table
PrivateSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PrivateSubnet2
    RouteTableId: !Ref PrivateRouteTable

Outputs:
VPCId:
  Description: "VPC ID"
  Value: !Ref VPC
  Export:
    Name: VPCId

PublicSubnet1:
  Description: "Public Subnet 1 ID"
  Value: !Ref PublicSubnet1
  Export:
    Name: PublicSubnet1

PublicSubnet2:
  Description: "Public Subnet 2 ID"
  Value: !Ref PublicSubnet2
  Export:
    Name: PublicSubnet2

PrivateSubnet1:
  Description: "Private Subnet 1 ID"
  Value: !Ref PrivateSubnet1
  Export:
    Name: PrivateSubnet1

PrivateSubnet2:
  Description: "Private Subnet 2 ID"
  Value: !Ref PrivateSubnet2
  Export:
    Name: PrivateSubnet2

NatGatewayId:
  Description: "NAT Gateway ID"
  Value: !Ref NatGateway

```

```

Export:
  Name: NatGatewayId

InternetGatewayId:
  Description: "Internet Gateway ID"
  Value: !Ref InternetGateway
  Export:
    Name: InternetGatewayId

```

### **3.2 Security Group Template:**

```

AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template for security groups for the VPC, including ALB, Auto Scaling instances, RDS, and Bastion Host.

Resources:
# Security Group for RDS (allows MySQL traffic on port 3306)
RDSSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Allow MySQL access to RDS"
    VpcId: !ImportValue VPCId
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 3306
        ToPort: 3306
        CidrIp: 10.0.0.0/16 # Use the VPC CIDR block for internal traffic
    Tags:
      - Key: Name
        Value: "RDS Security Group"

# Security Group for Auto Scaling instances (allow HTTP/HTTPS traffic and SSH from Bastion Host)
AutoscalingSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Allow HTTP, HTTPS, and SSH access to EC2 instances"
    VpcId: !ImportValue VPCId
    SecurityGroupIngress:
      # Allow HTTP traffic from ALB
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 10.0.0.0/16 # Internal traffic from ALB within the VPC
      # Allow HTTPS traffic from ALB
      - IpProtocol: tcp
        FromPort: 443

```

```

    ToPort: 443
    CidrIp: 10.0.0.0/16 # Internal HTTPS traffic from ALB within the VPC
    # Allow SSH traffic from the Bastion Host's subnet or private IP
    - IpProtocol: tcp
      FromPort: 22
      ToPort: 22
      CidrIp: 10.0.1.0/24 # Replace this with the Bastion Host subnet CIDR or specific IP

    Tags:
      - Key: Name
        Value: "Autoscaling Security Group"

# Security Group for ALB (allow HTTP/HTTPS traffic)
ALBSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Allow HTTP and HTTPS traffic to the ALB"
    VpcId: !ImportValue VPCID
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0 # Allow public HTTP traffic from the internet
      - IpProtocol: tcp
        FromPort: 443
        ToPort: 443
        CidrIp: 0.0.0.0/0 # Allow public HTTPS traffic from the internet
    Tags:
      - Key: Name
        Value: "ALB Security Group"

# Security Group for Bastion Host (allow SSH access from the team's IPs)
BastionSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: "Allow SSH access to Bastion Host"
    VpcId: !ImportValue VPCID
    SecurityGroupIngress:
      # Your IP 1
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 50.237.13.57/32 # IP 1
      # Your IP 2
      - IpProtocol: tcp
        FromPort: 22
        ToPort: 22
        CidrIp: 50.237.13.62/32 # IP 2
      # Teammate's IP 3

```

```

- IpProtocol: tcp
  FromPort: 22
  ToPort: 22
  CidrIp: 69.137.233.24/32 # IP 3
# Teammate's IP 4
- IpProtocol: tcp
  FromPort: 22
  ToPort: 22
  CidrIp: 98.192.33.217/32 # IP 4

Tags:
- Key: Name
  Value: "Bastion Security Group"

Outputs:
RDSSecurityGroupId:
  Description: "Security Group ID for RDS"
  Value: !Ref RDSSecurityGroup
  Export:
    Name: RDSSecurityGroupId

AutoscalingSecurityGroupId:
  Description: "Security Group ID for Auto Scaling instances"
  Value: !Ref AutoscalingSecurityGroup
  Export:
    Name: AutoscalingSecurityGroupId

ALBSecurityGroupId:
  Description: "Security Group ID for ALB"
  Value: !Ref ALBSecurityGroup
  Export:
    Name: ALBSecurityGroupId

BastionSecurityGroupId:
  Description: "Security Group ID for Bastion Host"
  Value: !Ref BastionSecurityGroup
  Export:
    Name: BastionSecurityGroupId

```

### **3.3 RDS Template:**

```

AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template to create a highly available, encrypted MySQL RDS
instance with storage auto-scaling and default KMS encryption at rest.

Parameters:
DBInstanceIdentifier:
  Description: "The name of the RDS instance"

```

```

Type: String
Default: "e-commerce"

DBPassword:
  Description: "The password for the RDS instance master user."
  Type: String
  NoEcho: true
  MinLength: 8
  MaxLength: 41
  AllowedPattern: "[a-zA-Z0-9]*"
  ConstraintDescription: "Must contain only alphanumeric characters."

DBAllocatedStorage:
  Description: "The allocated storage for the RDS instance in GB."
  Type: Number
  Default: 20

DBInstanceClass:
  Description: "The compute and memory capacity of the DB instance."
  Type: String
  Default: db.t3.medium

Resources:
# Create a DB Subnet Group for RDS (for Multi-AZ support)
DBSubnetGroup:
  Type: AWS::RDS::DBSubnetGroup
  Properties:
    DBSubnetGroupDescription: "Subnet group for RDS instance in private subnets"
    SubnetIds:
      - !ImportValue PrivateSubnet1 # Use imported value from VPC template
      - !ImportValue PrivateSubnet2 # Use imported value from VPC template
    Tags:
      - Key: Name
        Value: "RDS Subnet Group"

MyDBInstance:
  Type: AWS::RDS::DBInstance
  Properties:
    DBInstanceIdentifier: !Ref DBInstanceIdentifier
    AllocatedStorage: !Ref DBAllocatedStorage
    MaxAllocatedStorage: 100
    DBInstanceClass: !Ref DBInstanceClass
    Engine: MySQL
    EngineVersion: '8.0'
    MasterUsername: admin
    MasterUserPassword: !Ref DBPassword
    DBName: ecommerce_1
    MultiAZ: true

```

```

StorageType: gp2
StorageEncrypted: true
BackupRetentionPeriod: 7
VPCSecurityGroups:
  - !ImportValue RDSecurityGroupId # Imported RDS Security Group
DBSubnetGroupName: !Ref DBSubnetGroup
PubliclyAccessible: false
DeletionPolicy: Snapshot

Outputs:
DBInstanceEndpoint:
  Description: "RDS MySQL Endpoint"
  Value: !GetAtt MyDBInstance.Endpoint.Address

DBInstancePort:
  Description: "RDS MySQL Port"
  Value: !GetAtt MyDBInstance.Endpoint.Port

```

### 3.4 AutoScaling and Load Balancer Template:

```

AWSTemplateFormatVersion: '2010-09-09'
Description: CloudFormation template for an E-commerce platform with autoscaling support for instances in private subnets and a bastion host.

Parameters:
KeyName:
  Type: AWS::EC2::KeyPair::KeyName
  Default: enpm818n
  Description: "The name of the EC2 Key Pair to enable SSH access"

AMIId:
  Type: AWS::EC2::Image::Id
  Description: "The AMI ID for the EC2 instances"

Resources:
# Bastion Host in Public Subnet
BastionHost:
  Type: AWS::EC2::Instance
  Properties:
    InstanceType: t3.micro
    KeyName: !Ref KeyName
    ImageId: ami-08c40ec9ead489470 # Ubuntu 20.04 LTS AMI ID for us-east-1, replace for your region

```

```

SubnetId: !ImportValue "PublicSubnet1" # Bastion Host in Public Subnet 1
SecurityGroupIds:
  - !ImportValue "BastionSecurityGroupId" # Use the Bastion Security Group
Tags:
  - Key: Name
    Value: "BastionHost"

# Application Load Balancer
EcommerceLoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    Name: "EcommerceLoadBalancer"
    Subnets:
      - !ImportValue "PublicSubnet1" # Import Public Subnet 1 from the VPC template
      - !ImportValue "PublicSubnet2" # Import Public Subnet 2 from the VPC template
    SecurityGroups:
      - !ImportValue "ALBSecurityGroupId" # Import Load Balancer Security Group
    Scheme: "internet-facing"
    LoadBalancerAttributes:
      - Key: idle_timeout.timeout_seconds
        Value: '60'

# Target Group for Load Balancer
EcommerceTargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: "EcommerceTargetGroup"
    VpcId: !ImportValue "VPCId" # Ensure you import the correct VPC
    Port: 80
    Protocol: HTTP # Specify the protocol
    TargetType: instance
    HealthCheckPath: "/"
    HealthCheckProtocol: HTTP
    HealthCheckIntervalSeconds: 30
    HealthyThresholdCount: 3
    UnhealthyThresholdCount: 3
    HealthCheckPort: "80"
    HealthCheckTimeoutSeconds: 5
    Matcher:
      HttpCode: "200"

# Listener for Load Balancer
EcommerceListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    LoadBalancerArn: !Ref EcommerceLoadBalancer
    Port: 80
    Protocol: HTTP

```

```

DefaultActions:
  - Type: forward
    TargetGroupArn: !Ref EcommerceTargetGroup

# Launch Template for Private Instances
PrivateLaunchTemplate:
  Type: AWS::EC2::LaunchTemplate
  Properties:
    LaunchTemplateName: "EcommercePrivateLaunchTemplate"
    LaunchTemplateData:
      ImageId: !Ref AMIID # Use the AMI parameter
      InstanceType: t3.micro
      SecurityGroupIds:
        - !ImportValue "AutoscalingSecurityGroupId" # Import Auto Scaling Security Group
      KeyName: !Ref KeyName # Shared key name
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          set -e # Exit on any error

          # Go to the application directory
          cd /var/www/html
          sudo su
          git checkout -- .
          git pull origin main

          # Ensure permissions are correct for the web server
          chown -R www-data:www-data /var/www/html

          # Restart Apache to apply the latest code
          systemctl restart apache2

    TagSpecifications:
      - ResourceType: instance
        Tags:
          - Key: Name
            Value: "EcommercePrivateInstance"

# Auto Scaling Group for Private Instances
PrivateAutoScalingGroup:
  Type: AWS::AutoScaling::AutoScalingGroup
  DependsOn: EcommerceLoadBalancer
  Properties:
    VPCZoneIdentifier:
      - !ImportValue "PrivateSubnet1"
      - !ImportValue "PrivateSubnet2"
    LaunchTemplate:
      LaunchTemplateId: !Ref PrivateLaunchTemplate

```

```

Version: !GetAtt PrivateLaunchTemplate.LatestVersionNumber
MinSize: 1
MaxSize: 5
DesiredCapacity: 2
TargetGroupARNs:
  - !Ref EcommerceTargetGroup
HealthCheckType: ELB
HealthCheckGracePeriod: 300

# Scale-Up Policy for Private Instances
PrivateScaleUpPolicy:
  Type: AWS::AutoScaling::ScalingPolicy
  DependsOn: PrivateAutoScalingGroup # Ensure Auto Scaling Group is created first
  Properties:
    AutoScalingGroupName: !Ref PrivateAutoScalingGroup
    PolicyType: StepScaling
    StepAdjustments:
      - MetricIntervalLowerBound: 0
        scalingAdjustment: 1
    AdjustmentType: ChangeInCapacity
    Cooldown: 120
    EstimatedInstanceWarmup: 120

# Scale-Down Policy for Private Instances
PrivateScaleDownPolicy:
  Type: AWS::AutoScaling::ScalingPolicy
  DependsOn: PrivateAutoScalingGroup
  Properties:
    AutoScalingGroupName: !Ref PrivateAutoScalingGroup
    PolicyType: StepScaling
    StepAdjustments:
      - MetricIntervalUpperBound: 0
        scalingAdjustment: -1
    AdjustmentType: ChangeInCapacity
    Cooldown: 120
    EstimatedInstanceWarmup: 120

# Scale-Up Alarm (>= 30% CPU)
ScaleUpAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: "Scale up when CPU Utilization is >= 50% for 50 seconds"
    MetricName: "CPUUtilization"
    Namespace: "AWS/EC2"
    Statistic: "Average"
    Period: 60
    EvaluationPeriods: 1
    Threshold: 50

```

```

ComparisonOperator: "GreaterThanOrEqualToThreshold"
AlarmActions:
  - !Ref PrivateScaleUpPolicy
Dimensions:
  - Name: "AutoScalingGroupName"
    Value: !Ref PrivateAutoScalingGroup

# Scale-Down Alarm (<= 10% CPU)
ScaleDownAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: "Scale down when CPU Utilization is <= 10% for 60 seconds"
    MetricName: "CPUUtilization"
    Namespace: "AWS/EC2"
    Statistic: "Average"
    Period: 60
    EvaluationPeriods: 1
    Threshold: 10
    ComparisonOperator: "LessThanOrEqualToThreshold"
  AlarmActions:
    - !Ref PrivateScaleDownPolicy
  Dimensions:
    - Name: "AutoScalingGroupName"
      Value: !Ref PrivateAutoScalingGroup

Outputs:
  PrivateAutoScalingGroupName:
    Description: "Name of the Private Auto Scaling Group"
    Value: !Ref PrivateAutoScalingGroup

  LoadBalancerDNSName:
    Description: "DNS Name of the Load Balancer"
    Value: !GetAtt EcommerceLoadBalancer.DNSName

  BastionHostPublicIP:
    Description: "Public IP of the Bastion Host"
    Value: !GetAtt BastionHost.PublicIp

```

All these templates can also be easily accessible on our github repository:  
<https://github.com/tan-vi-k/enpm818NyamlFiles.git>

## 4. SECURITY MEASURES:

### 4.1 Amazon Key Management Service (KMS):

Amazon KMS is a fully managed service provided by AWS, which makes it easy to manage and use. We have implemented it to enhance the security of our database infrastructure. This provides robust encryption capabilities, ensuring the confidentiality and integrity of our sensitive data. The advantage of using the KMS service is that it is highly scalable, so if the instances need to be scaled up or down, it automatically does it for us. Also, KMS integrates with AWS CloudTrail, providing a comprehensive audit trail of all key usage for monitoring and compliance purposes.

Instance			
Configuration	Instance class	Storage	Performance Insights
DB instance ID e-commerce	Instance class db.t3.medium	Encryption Enabled	Performance Insights enabled Turned on
Engine version 8.0.39	vCPU 2	AWS KMS key <a href="#">aws/rds</a>	AWS KMS key <a href="#">aws/rds</a>
RDS Extended Support Enabled	RAM 4 GB	Storage type General Purpose SSD (gp2)	Retention period 7 days
DB name	Availability	Storage	

Using KMS, the data encryption is satisfied at rest.

### 4.2 SSL for domain:

As per the workflow of our architecture, once the our domain enpmprojectgroup4.shop is requested, Route53 directs a request through Cloudfront and WAF, and sends it to the Load Balancer. As our website is an e-commerce site, sensitive data needs to be transferred to and fro, such as login details, order details, payment details, etc, so it is important for the data in transit to be encrypted.

To achieve this, we have enabled a SSL certificate from AWS Certificate Manager (ACM) on our domain, which is also associated with the Load balancer and CloudFront services that we have used.

The screenshot shows the AWS Certificate Manager (ACM) console with the following details:

- Certificate status:** Identifier: b27396c9-bec3-4692-befc-bc9d06eed2f1, ARN: arn:aws:acm:us-east-1:314147830127:certificate/b27396c9-bec3-4692-befc-bc9d06eed2f1, Type: Amazon Issued, Status: Issued.
- Domains (1):** enpmprojectgroup4.shop, Status: Success, CNAME value: fd9b2a2280dab2cdf9ee75bec566567.enpmprojectgroup4.shop.
- Details:**
  - In use: Yes
  - Domain name: enpmprojectgroup4.shop
  - Number of additional names: 0
  - Can be used with: CloudFront, Elastic Load Balancing, API Gateway and other integrated services.
- Associated resources (2):** arn:aws:cloudfront:314147830127:distribution/E31IDNRUU9T944, arn:aws:elasticloadbalancing:us-east-1:314147830127:loadbalancer/app/EcommerceLoadBalancer/76a5f327091ce21

## 4.3 SSL between database and web connections:

The image below shows the connect.php file in which an SSL certificate, located at `/var/www/html/global-bundle.pem`, is specified to authenticate the connection. The `mysqli_ssl_set` function is used to enforce SSL, ensuring that all communication between the web server and the RDS instance is encrypted.

```
GNU nano 7.2
<?php
// Database connection parameters
$host = 'e-commerce.cchcwgordfrx.us-east-1.rds.amazonaws.com'; // RDS endpoint
$user = 'admin'; // Database admin user
$password = 'rootemp818n'; // Admin password
$dbname = 'ecommerce_1'; // Database name

// Create connection

$ssl_cert_path = '/var/www/html/global-bundle.pem';
$con = new mysqli($host, $user, $password, $dbname);
$con->ssl_set(NULL, NULL, $ssl_cert_path, NULL, NULL);

// Check connection and provide detailed error reporting
if ($con->connect_error) {
    die("Connection failed: " . $con->connect_error . "<br>Error Number: " . $con->connect_errno);
}

?>
```

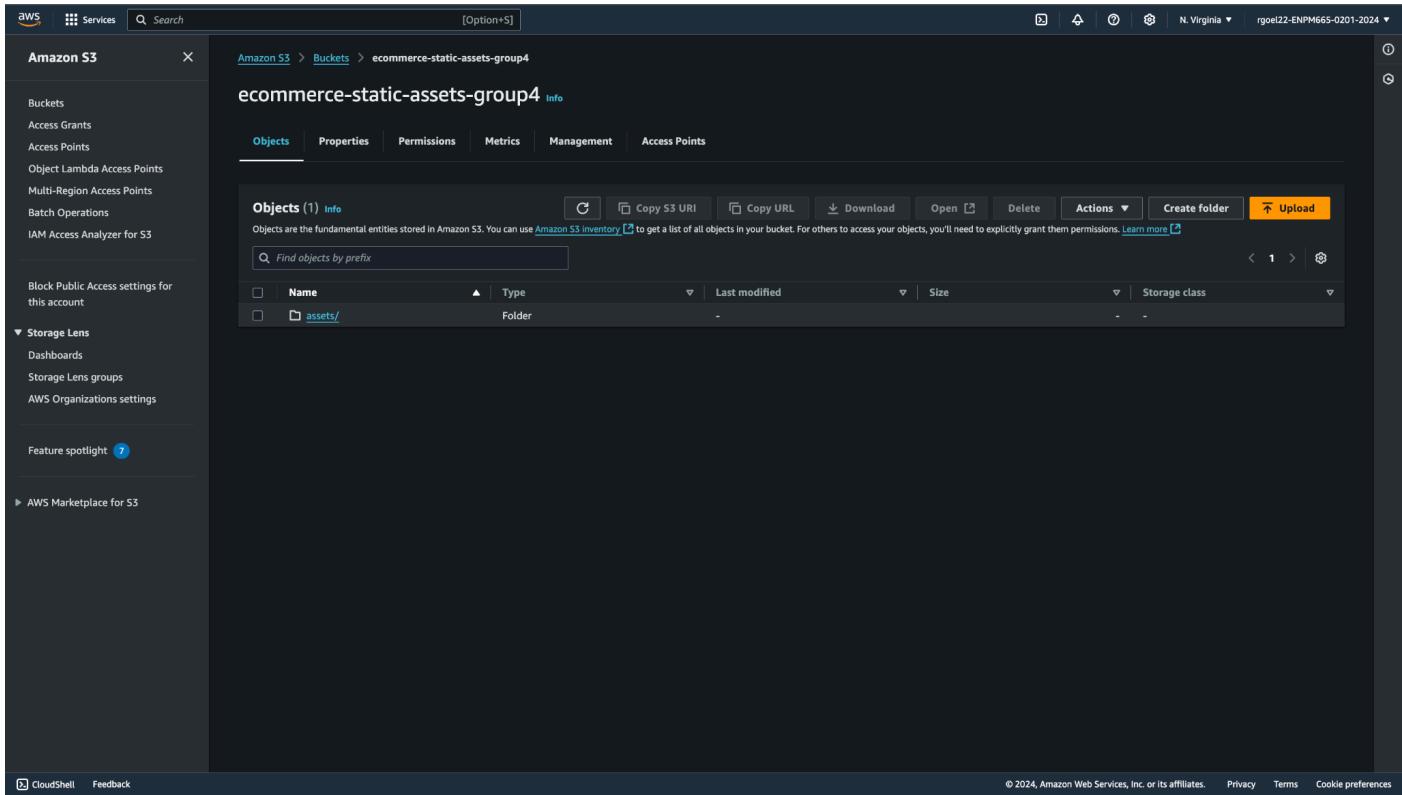
## 4.4 WAF rules configured:

The screenshot shows the AWS WAF & Shield interface. On the left, the navigation menu includes 'AWS WAF' (selected), 'Getting started', 'Web ACLs' (selected), 'Bot control dashboard', 'Application integration', 'IP sets', 'Regex pattern sets', 'Rule groups', 'AWS Marketplace managed rules', and 'Switch to AWS WAF Classic'. Under 'AWS Shield', there are 'Getting started' and 'Overview'. The main panel displays the 'ecommerce-waf' Web ACL. It has tabs for 'Traffic overview', 'Rules' (selected), 'Associated AWS resources', 'Custom response bodies', 'Logging and metrics', 'Sampled requests', and 'CloudWatch Log Insights'. A 'Download web ACL as JSON' button is also present. The 'Rules' section lists six managed rule groups: 'AWS-AWSManagedRulesAmazonIpReputationList', 'AWS-AWSManagedRulesSQLiRuleSet', 'AWS-AWSManagedRulesCommonRuleSet', 'AWS-AWSManagedRulesPHPRuleSet', and 'AWS-AWSManagedRulesKnownBadInputsRuleSet'. Each rule is listed with its name, action (Use rule actions), priority (0-4), and custom response.

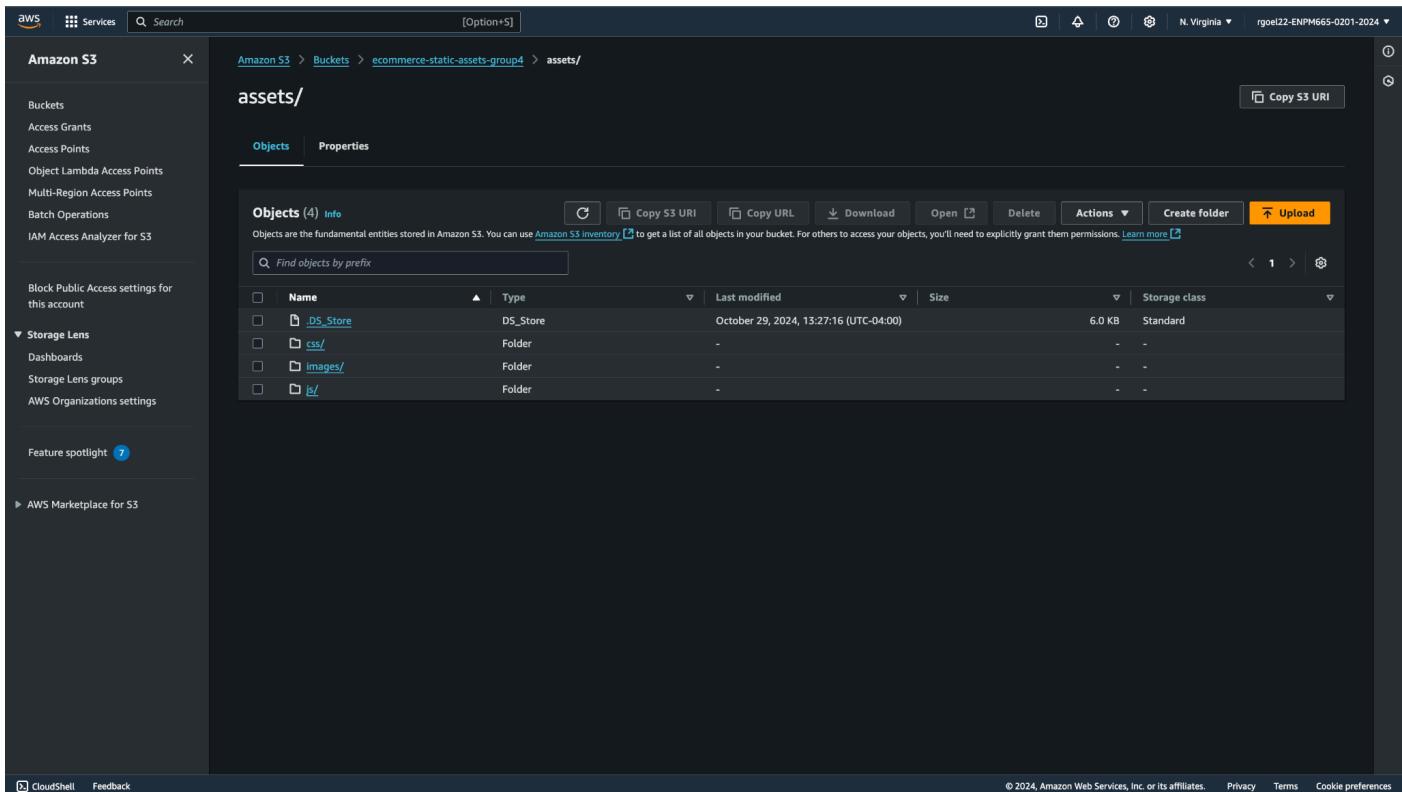
The WAF configuration for the "*ecommerce-waf*" includes a comprehensive set of managed rule groups, such as the **AWSManagedRulesAmazonIpReputationList**, **SQLiRuleSet**, **CommonRuleSet**, **PHPRuleSet**, and **KnownBadInputsRuleSet**. With a Web ACL capacity usage of **1225** out of the allowable 5000 WCUs, the setup is well within limits, optimizing security without compromising performance. The default action is set to "**Allow**" for requests that do not match any specific rules, maintaining a balanced security approach while permitting necessary traffic.

## 5. OPTIMIZATION MEASURES:

### 5.1 CloudFront:



The screenshot shows the AWS S3 console interface. On the left, the navigation pane is visible with sections like Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Storage Lens, Dashboards, Storage Lens groups, AWS Organizations settings, Feature spotlight, and AWS Marketplace for S3. The main content area displays the details for the bucket 'ecommerce-static-assets-group4'. The 'Objects' tab is selected, showing one object named 'assets/'. A table lists the object's details: Name (assets/), Type (Folder), Last modified (October 29, 2024, 13:27:16 (UTC-04:00)), Size (6.0 KB), and Storage class (Standard). Below the table, there are buttons for Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, and Upload.



This screenshot is identical to the one above, showing the contents of the 'assets/' folder in the 'ecommerce-static-assets-group4' bucket. The table shows the same four objects: DS\_Store, css/, images/, and js/. The DS\_Store object is a folder with a size of 6.0 KB and a last modified date of October 29, 2024, 13:27:16 (UTC-04:00). The other three are empty folders.

The S3 bucket, named **ecommerce-static-assets-group4**, was configured to serve as the origin for CloudFront, providing a centralized storage for files like CSS, JavaScript, and images. By organizing assets within designated folders—css, js, and images—this structure ensures easier file management and retrieval, as seen in the organization under the "**assets**" directory in the S3 configuration.

**Permissions overview**

**Block public access (bucket settings)**

Block all public access  On Individual Block Public Access settings for this bucket

**Bucket policy**

Public access is blocked because Block Public Access settings are turned on for this bucket

```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      ...
    }
  ]
}
```

**Bucket policy**

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Public access is blocked because Block Public Access settings are turned on for this bucket

```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3::ecommerce-static-assets-group4/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::314147830127:distribution/E31IDNRUU9T944"
        }
      }
    }
  ]
}
```

**Object Ownership**

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Object Ownership  
Bucket owner enforced  
ACLs are disabled. All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

**Access control list (ACL)**

Grant basic read/write permissions to other AWS accounts. [Learn more](#)

The permissions configuration for the **ecommerce-static-assets-group4** S3 bucket prioritizes security and controlled access, with public access fully blocked to prevent unauthorized access. To enable secure integration with CloudFront, a specific bucket policy was applied, allowing only the designated CloudFront distribution to retrieve objects from the bucket. This policy authorizes the CloudFront service principal (`cloudfront.amazonaws.com`) to perform the `s3:GetObject` action, restricted by a condition that ensures access is granted exclusively through the specified CloudFront distribution.

The screenshot shows the AWS CloudFront Distribution Details page for distribution ID **E31IDNRUU9T944**. The left sidebar contains navigation links for CloudFront services like Distributions, Telemetry, Reports & analytics, Key management, and Savings Bundle. The main content area displays the distribution's details, including its distribution domain name (`d2n9uhffspp0.cloudfront.net`), ARN (`arn:aws:cloudfront::314147830127:distribution/E31IDNRUU9T944`), and last modified date (October 29, 2024). The 'Settings' section includes fields for Description, Price class (set to 'Use all edge locations (best performance)'), Supported HTTP versions (HTTP/2, HTTP/1.1, HTTP/1.0), Alternate domain names, Custom SSL certificate (linked to 'enpmprojectgroup4.shop'), Security policy (TLSv1.2\_2021), Standard logging (Off), Cookie logging (Off), and Default root object. A 'Continuous deployment' section is also present.

The screenshot shows a GitHub pull request titled "Changes for cloudfront url" by user **rgoel22**. The code editor displays the `index.php` file from the `master` branch. The code includes logic to check if the session variable `admin_username` is set, and if so, it constructs a SQL query to fetch the admin's name from a database table. It then uses `window.open` to redirect the user to an admin login page. The GitHub Copilot icon is visible in the top right of the code editor. A specific line of code is highlighted in yellow, which contains the CloudFront URL `https://d2n9uhffspp0.cloudfront.net/assets/css/bootstrap.css`.

CloudFront distribution (**E31IDNRUU9T944**) was created to cache assets across its global network, enhancing load times by delivering content from the edge location closest to the user. The CloudFront URL (`d2n9uhffspp0.cloudfront.net`) was then integrated in the code, replacing direct links to S3.

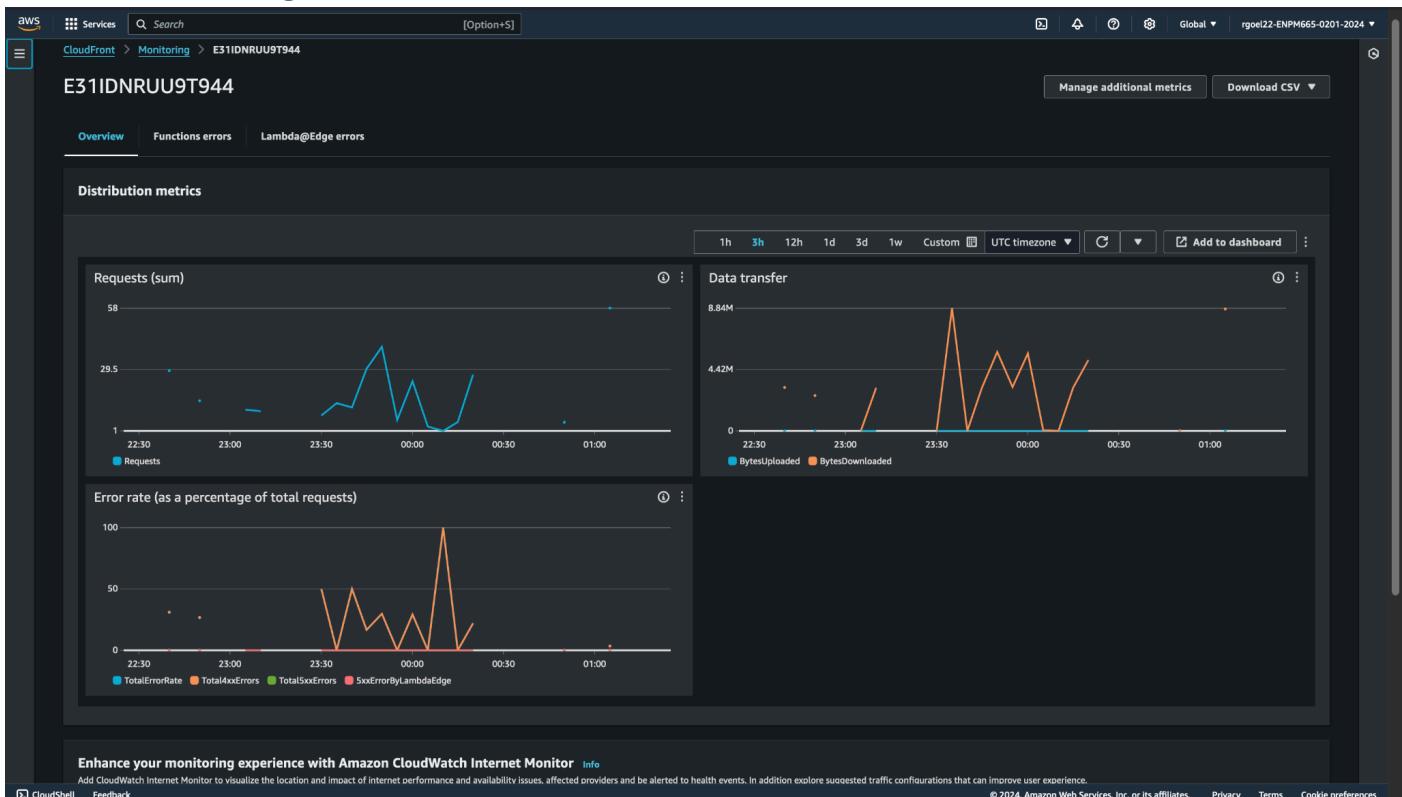
## 5.2 Gzip compression:

The screenshot shows the Network tab in DevTools for a request to index.php. The main.css file is selected. Headers show CloudFront delivery with Gzip compression activated. Request Headers include Accept-Encoding: gzip, deflate, br, zstd.

Name	Headers
index.php	Server: AmazonS3 Vary: Accept-Encoding Via: 1.1 6260617092a905727637e669f8f39e2.cloudfront.net (CloudFront) X-Amz-Cf-Id: wxeMs8alyVwH0JddNlbBg1RoCfc1Rodjjjd2U4BFSSjydGoUahrNw== X-Amz-Cf-Pop: IAD55-P6 X-Amz-Server-Side-Encryption: AES256 X-Cache: Hit from cloudfront
main.css	:authority: d2n9uhffsp0.cloudfront.net :method: GET :path: /assets/css/main.css :scheme: https Accept: text/css,*/*;q=0.1 Accept-Encoding: gzip, deflate, br, zstd Accept-Language: en-US,en;q=0.9

The image displays the effective deployment of GZIP compression on static assets in our web application, focusing on the **main.css** file distributed via CloudFront with compression activated. As visible, **Content-Encoding: gzip** response header, CloudFront delivers the CSS file with GZIP compression, which is vital for optimizing web performance.

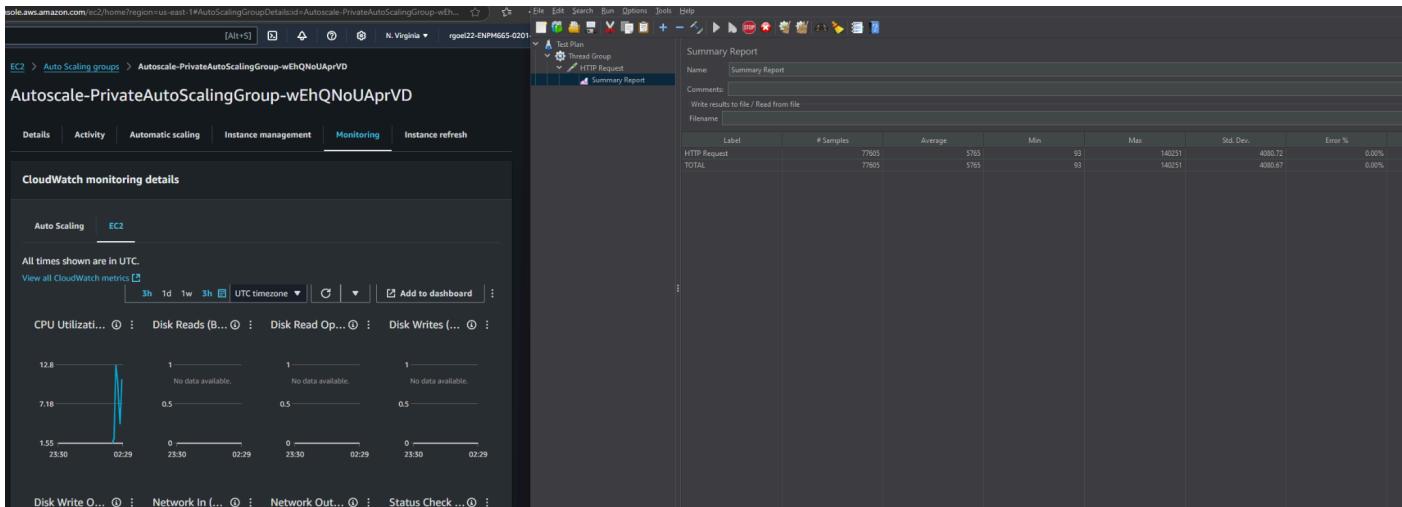
## 5.3 Cloudwatch Logs:



## 6. TESTING AND MONITORING:

### 6.1 Stress Testing:

The stress testing was simulated using Apache JMeter (5.6.3). To see auto scaling in action, we needed to temporarily dial down the upscale CPU utilization threshold to 10 % from the production's 50%. A total of 1000 individual users/threads with a ramp-up period of 20 secs accomplished that.



<b>Successful</b>	Launching a new EC2 instance: i-02ca2835fa20aa410	At 2024-10-30T02:31:06Z a monitor alarm Autoscale-ScaleUpAlarm-zxEwcxsjTvtG in state ALARM triggered policy Autoscale-PrivateScaleUpPolicy-DZnL8mUPj5de changing the desired capacity from 1 to 2. At 2024-10-30T02:31:16Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.	2024 October 29, 10:31:18 PM -04:00	2024 October 29, 10:33:24 PM -04:00
-------------------	---	--	-------------------------------------	-------------------------------------

### 6.2 Monitoring:

This screenshot shows the AWS CloudTrail Trails console. It lists a single trail named 'management-events'. The trail is configured with the Home region as US East (N. Virginia), Multi-region trail as Yes, Insights as Disabled, Organization trail as No, S3 bucket as 'aws-cloudtrail-logs-314147830127-577872fd', Log file prefix as '577872fd', CloudWatch Logs log group as 'aws-cloudtrail-logs-314147830127-577872fd', and Status as Logging.

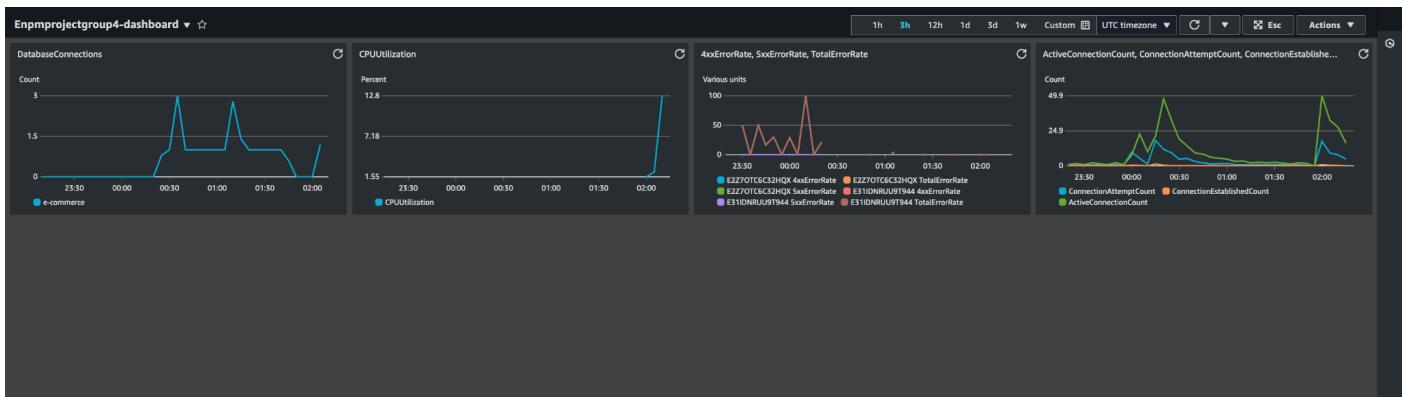
This screenshot shows the AWS Amazon S3 Buckets console. It displays the contents of the 'aws-cloudtrail-logs' bucket. There is a single folder named 'AWSLogs/'. The objects table shows one item: 'AWSLogs/'.

Screenshot of the AWS S3 console showing the CloudTrail logs bucket. The left sidebar shows navigation links like Buckets, IAM Access Analyzer for S3, Storage Lens, and AWS Marketplace for S3. The main content area displays a list of objects in the 'CloudTrail\_us-east-1' folder, all named with a timestamp and suffix (gz). The objects are standard storage class files.

The CloudTrail, includes a "**management-events**" trail to log security and management events across all regions. The logs are stored in the S3 bucket "***aws-cloudtrail-logs-314147830127-577872fd***", as shown in the above images. This setup centralizes event records securely, ensuring that all critical activities are logged and stored for future analysis and auditing.

## 6.3 Cloudwatch Dashboard:

The following is the custom dashboard created to monitor database connections, CPU utilization, error rates and active connections to the website.



## 7. COST ANALYSIS:

Given that the increase in AWS costs is due to recent usage for the assignment, it's crucial to focus on optimizing future expenses. The current month-to-date cost is \$7.28, with a forecasted total of \$9.70, compared to last month's minimal usage of \$1.23. The primary costs stem from Amazon RDS and EC2 services. To manage and potentially reduce costs, we recommend adopting Reserved Instances (RIs) or Compute Savings Plans, which can provide up to 72-75% savings for long-term, consistent usage. Specifically, for Amazon RDS, using RDS Reserved Instances or Savings Plans would significantly lower ongoing database costs. Additionally, optimizing VPC usage by reviewing and eliminating unnecessary resources like unused Elastic IPs or NAT gateways can further reduce costs. Regularly tracking AWS usage and costs using AWS Cost Explorer will help identify any unexpected spikes and allow for timely resource adjustments. These measures will ensure efficient use of AWS services while minimizing long-term expenses.

The screenshot shows the AWS Billing and Cost Management Bills page. At the top, there are navigation links for 'Billing and Cost Management' and 'Bills'. Below that is a 'Bills' section with a 'Info' link. On the right, there are buttons for 'Download all to CSV', 'Print', and a dropdown for 'Billing period: October 2024'. A refresh button and a help icon are also present. The main content area displays the 'AWS estimated bill summary' with the following details:

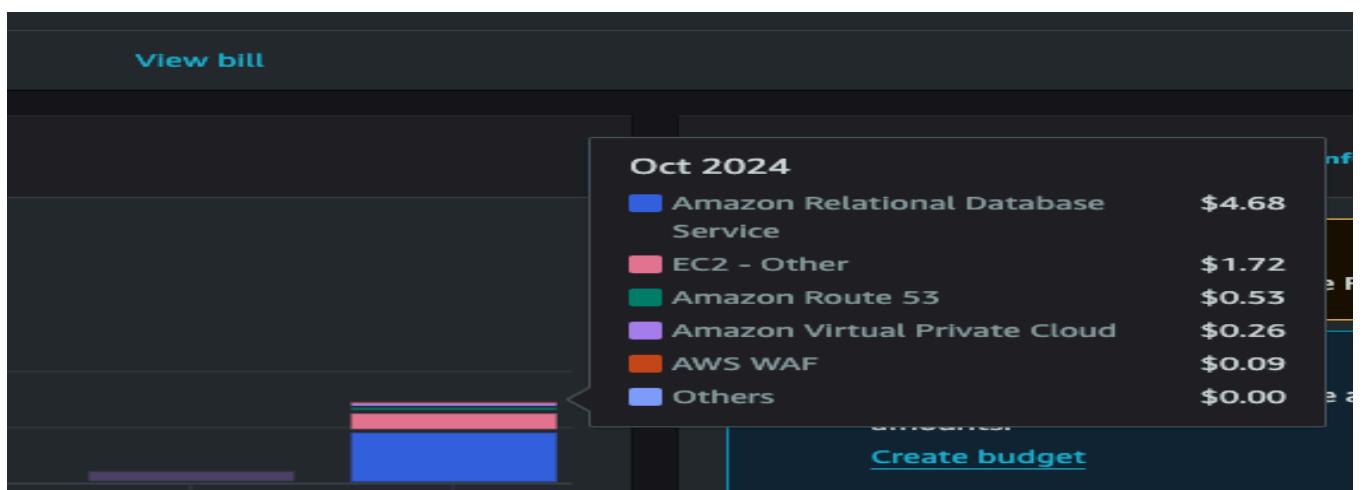
AWS estimated bill summary <a href="#">Info</a>	
Total charges and payment information	
Account ID 314147830127	Billing period <a href="#">Info</a> October 1 – October 31, 2024
Service provider Amazon Web Services, Inc.	Bill status <a href="#">Info</a> Pending
Total in USD <b>USD 0.82</b>	
Estimated grand total: <b>USD 0.82</b>	

The screenshot shows the AWS Billing and Cost Management home page. It features two main sections: 'Cost summary' and 'Cost monitor'.

**Cost summary:** This section displays month-to-date and total forecasted costs, along with a comparison to the previous month. It includes a 'View bill' button.

Month-to-date cost <b>\$7.28</b>	Last month's cost for same time period <b>\$1.23</b> Sep 1 – 29
Total forecasted cost for current month <b>\$9.70</b>	Last month's total cost <b>\$1.23</b>

**Cost monitor:** This section shows budgets and cost anomalies. It indicates 'Setup required' for budgets and '3 detected' for cost anomalies.



## **8. CONCLUSION:**

In conclusion, our project successfully met its core objectives of designing and deploying a secure, scalable, and high-performance e-commerce platform on AWS. By leveraging AWS's cloud infrastructure and managed services, including Auto Scaling, Load Balancers, RDS, WAF, and CloudFront, we created an architecture that supports fluctuating user demands, optimizes performance, and secures sensitive data. The implementation of encryption for data at rest and in transit, along with strict access controls through security groups and WAF rules, has ensured a high level of data security and compliance with industry standards. CloudFront's CDN capabilities further enhanced user experience by providing low-latency content delivery across global locations.

Our cost analysis highlights the value of AWS's Reserved Instances and Savings Plans for long-term cost efficiency. The adoption of these financial strategies, along with continuous monitoring using AWS Cost Explorer, will help manage the platform's operational costs effectively. Rigorous testing through tools like Apache JMeter validated our system's scalability and resilience under various load conditions, affirming that the platform can adapt to real-world traffic variations while maintaining stable performance.

This project not only achieved its technical goals but also served as a practical demonstration of AWS best practices for cloud-based e-commerce platforms. The foundational architecture and configurations developed here provide a strong basis for future enhancements, including the potential integration of advanced analytics, AI-driven insights, and additional security layers. This report showcases the thoughtful design and diligent implementation process undertaken to ensure both immediate functionality and long-term sustainability of our e-commerce solution on AWS.

## **9. RESOURCES:**

- [1] <https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html>
- [2] [https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER\\_CreateDBInstance.html](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateDBInstance.html)
- [3] <https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html>
- [4] <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html>
- [5] <https://docs.aws.amazon.com/awscloudtrail/latest/userguide/best-practices-security.html>
- [6] <https://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html>
- [7] <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/ServingCompressedFiles.html>
- [8] <https://aws.amazon.com/premiumsupport/support-cloud-cost-optimization/>