Background : You are recently promoted from a Cloud Engineer to a Cloud Architect and assigned a project to prepare a new environment in the cloud, to which your team will later migrate their applications

Requirements:

•They are concerned about the security of the environment, so they have decided to virtually isolate their network from the rest of the customers and the rest of the environments in the same AWS Cloud Account

•Due to the budget issue, the company cannot afford a dedicated DB engineer, so they are willing to outsource the DB management from a Cloud provider, to store and maintain the customer information received by PHP application. You must pick the right solution from AWS, which should be a Platform as a Service. It should also provide high availability, patching, and back-ups. (hint: Create DB subnet group)

•And about disaster recovery, you should have enough backups for both the Web and Database server, so if in case the environment crashes, we can launch a new environment from the disaster recovery backups

•Design a dynamic website where the customers can enter their details, which should be stored in a database

•They are uncertain about the traffic pattern that how low or high it can be, so they have a requirement that the environment should be running at least two EC2 servers all time, and when there is a high load, they must burst up to four servers in total

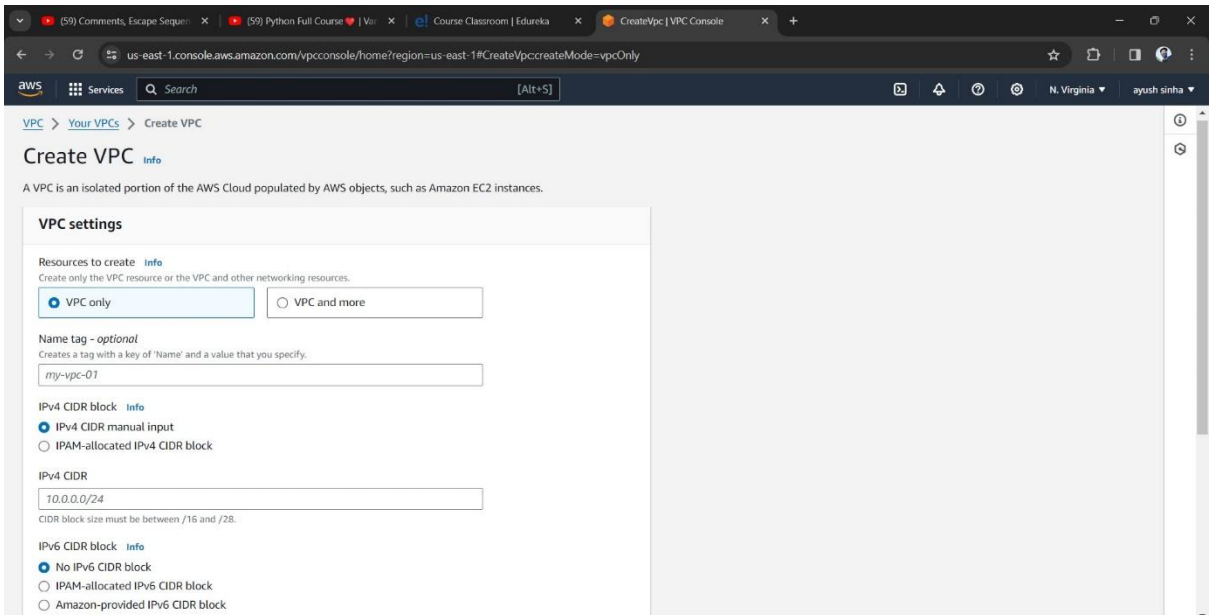
•Now the company cannot afford a dedicated engineer for monitoring, so you must automate the incident management through an event notification. Anytime there is an increase and decrease in the VM's due to high or low traffic, you must receive a notification via email

•The application should be highly available, even if a VM fails to respond to queries, there should be a mechanism to shift the connection to another healthy VM automatically

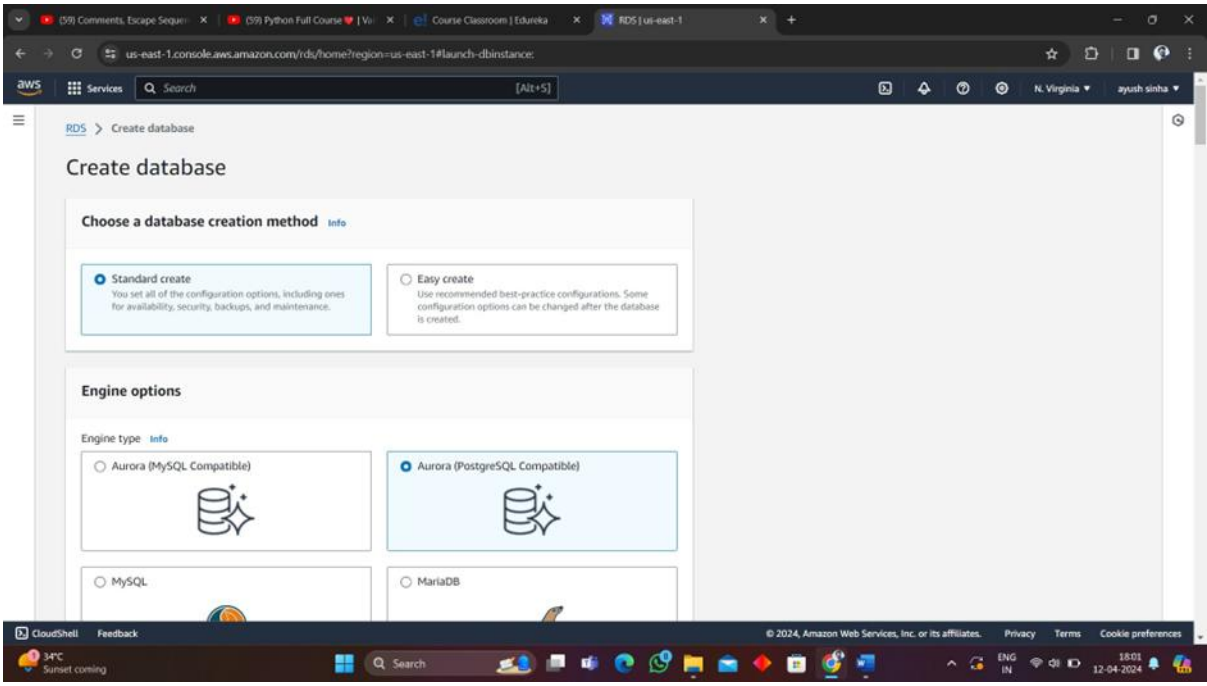
•Your Dynamic Website should also be cached globally, so users worldwide can access it with less latency. The customer is okay if we get an unfriendly AWS generated URL for accessing the website

the requirements and design a solution that meets all of them using AWS services:

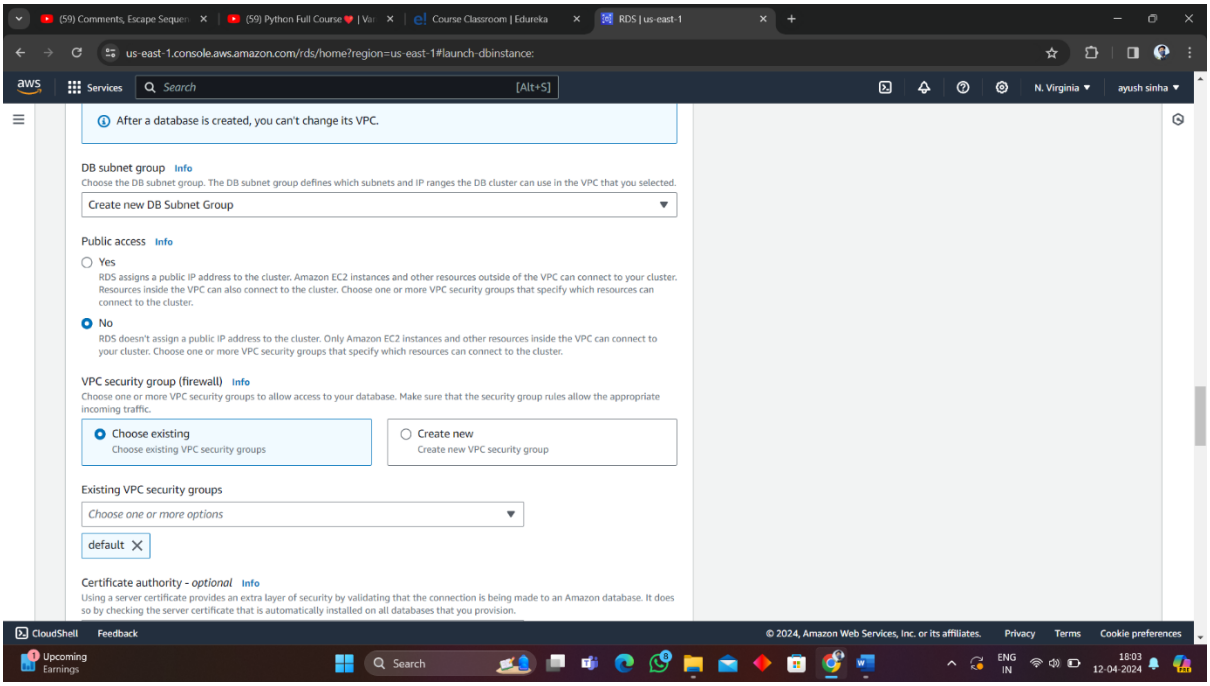
1. **Isolated Network Environment**:
    - Use Virtual Private Cloud (VPC) to create a private network for your resources.
    - Configure Network Access Control Lists (NACLs) and Security Groups to control inbound and outbound traffic.



2. **Database Management**:
    - Choose Amazon RDS (Relational Database Service) as a Platform as a Service (PaaS) for your database.



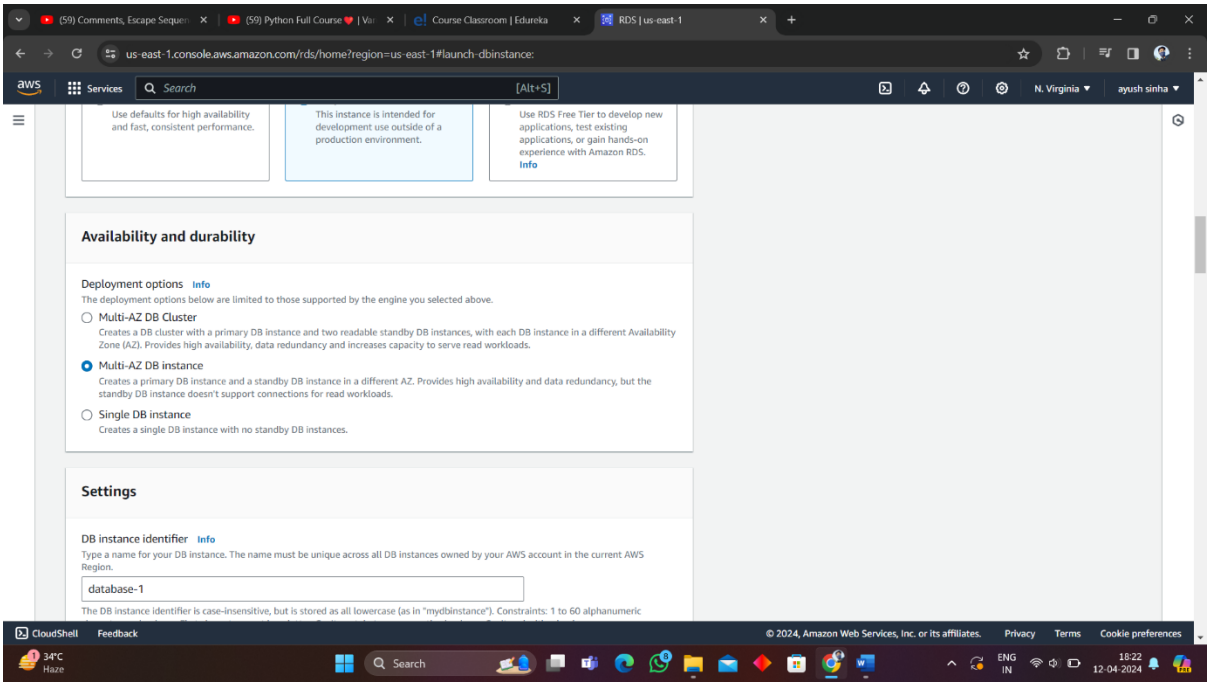
- Create a DB subnet group within your VPC for the RDS instance to ensure it's in a private subnet.



- Enable Multi-AZ deployment for high availability, automatic failover, and backups.
- RDS handles patching and backups automatically.

3. **Disaster Recovery**:

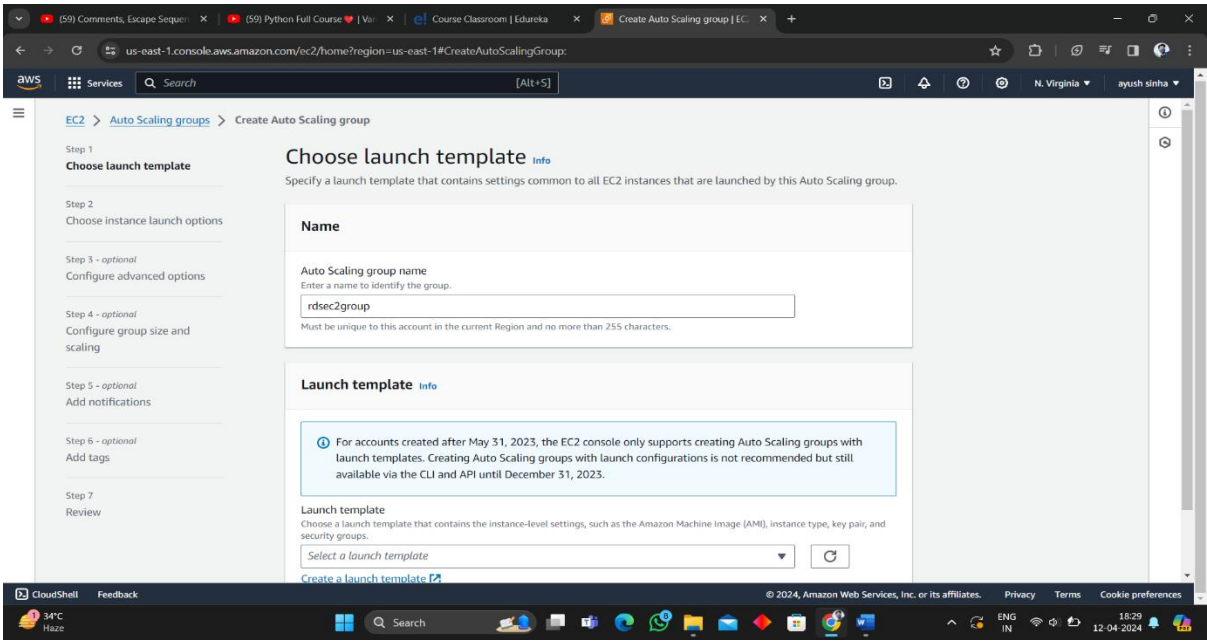
    - Enable automated backups for both the web servers (EC2 instances) and the RDS database.
    - Configure RDS Multi-AZ deployment for automatic failover in case of a primary instance failure.



    - Use AWS Backup for additional backup management and retention policies.

4. **Dynamic Website and Database**:

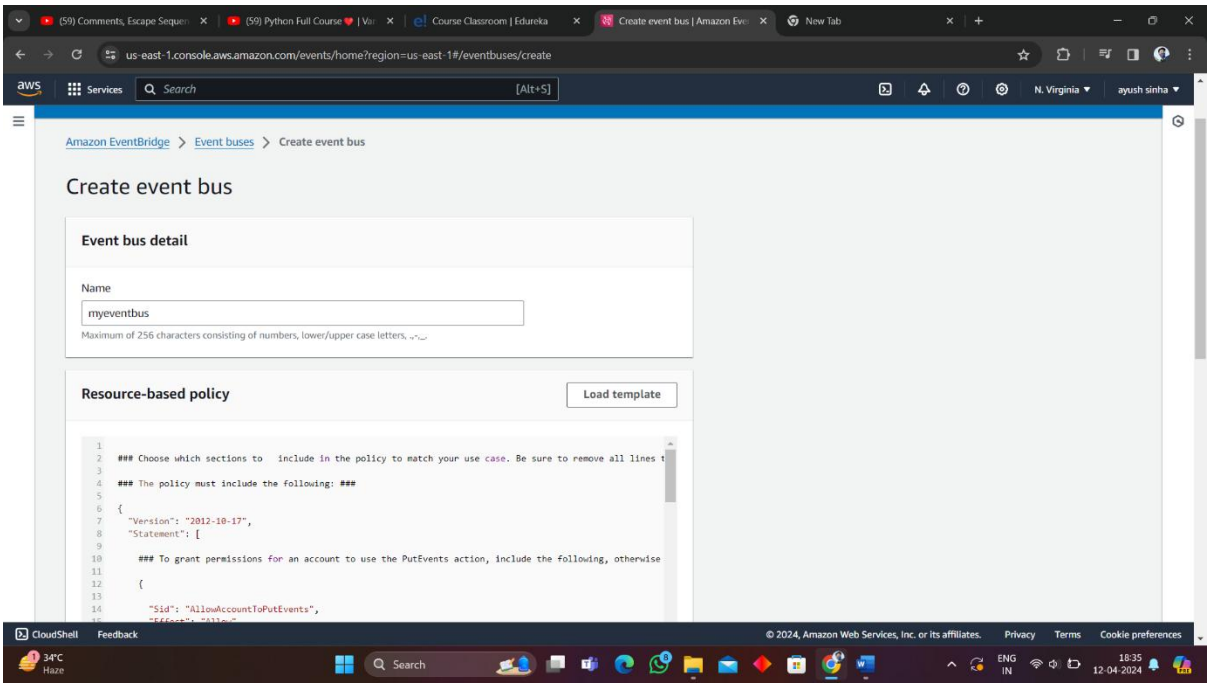
    - Set up an Auto Scaling Group for your EC2 instances to maintain at least two servers and scale up to four based on demand.



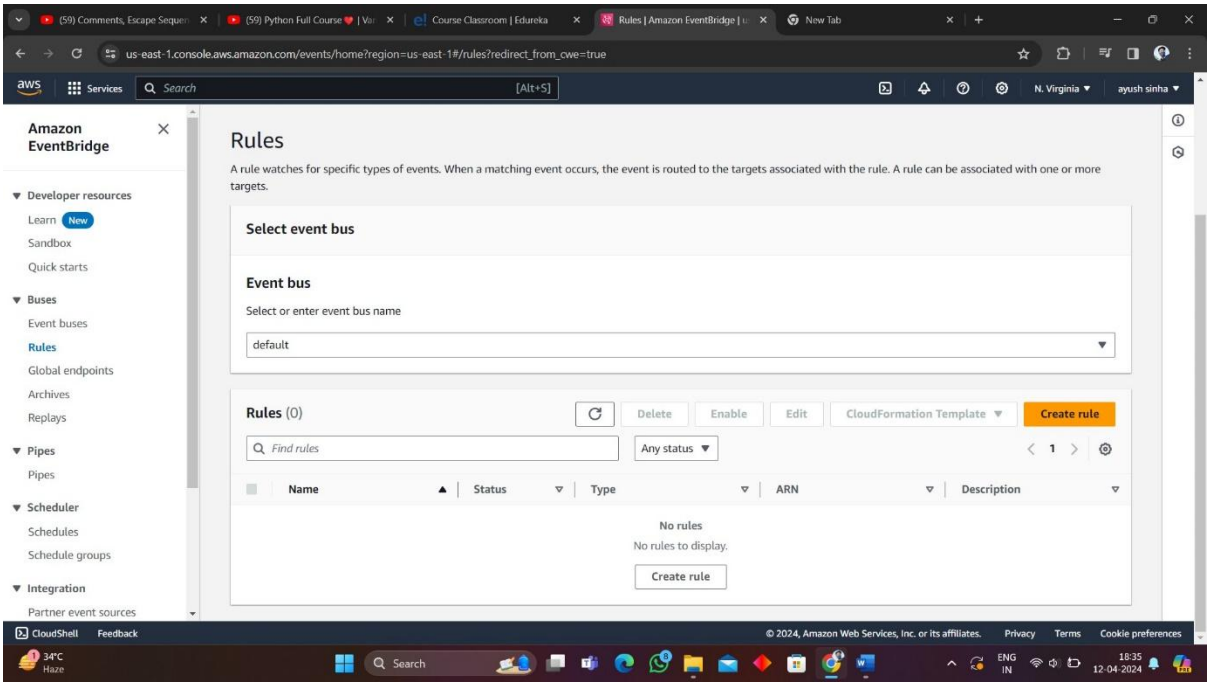
- Develop your dynamic website using PHP and connect it to the RDS database for storing customer information.

5. **Automated Incident Management**:

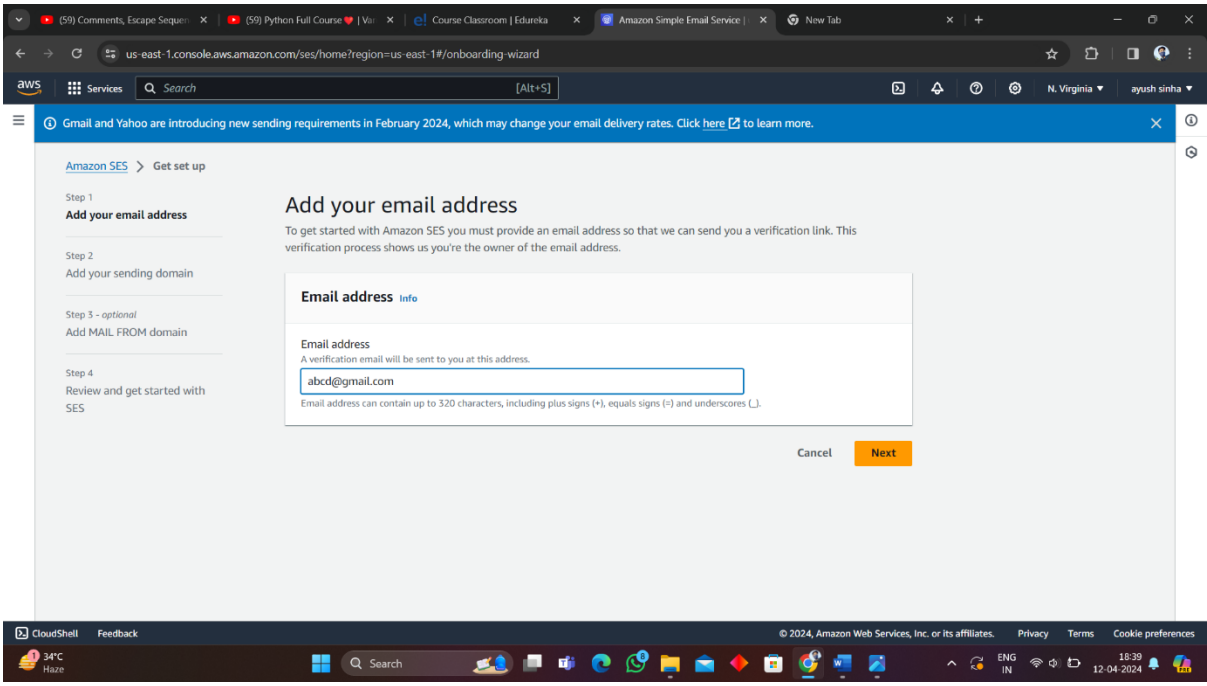
- Use AWS CloudWatch Events to monitor EC2 instance state changes (scaling up/down).



- Create a CloudWatch Event Rule to trigger an AWS Lambda function when EC2 instances scale up/down.

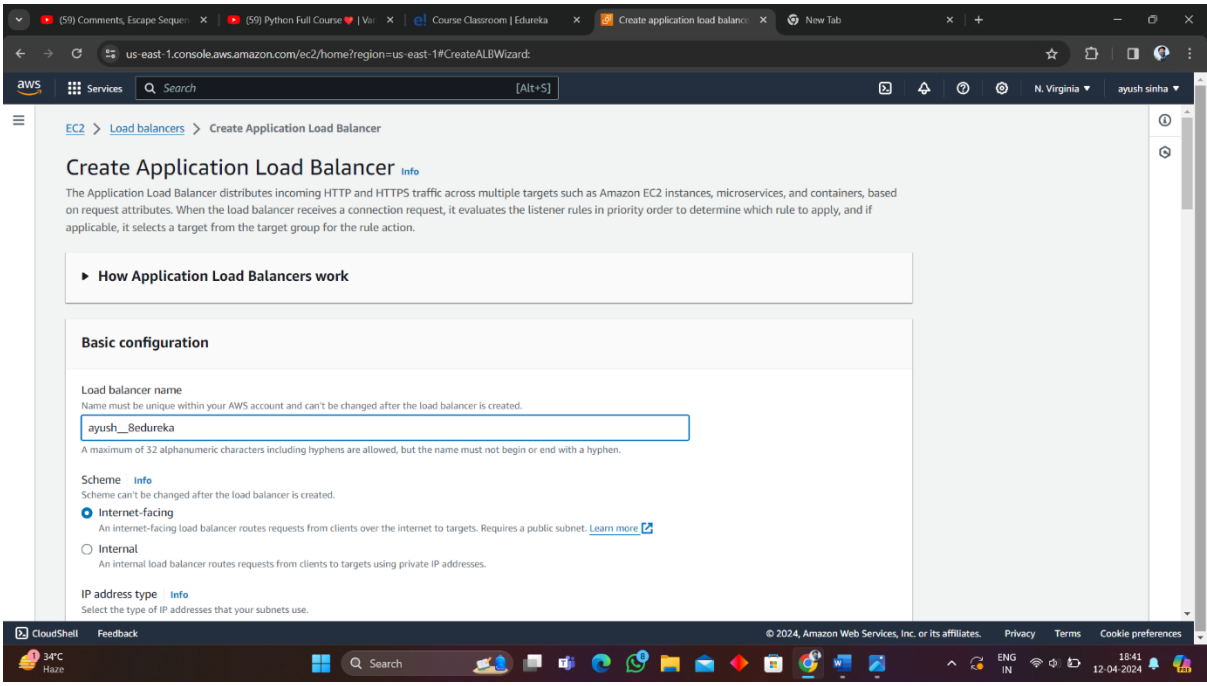


- The Lambda function can send an email notification using Amazon SES (Simple Email Service).



6. **High Availability**:

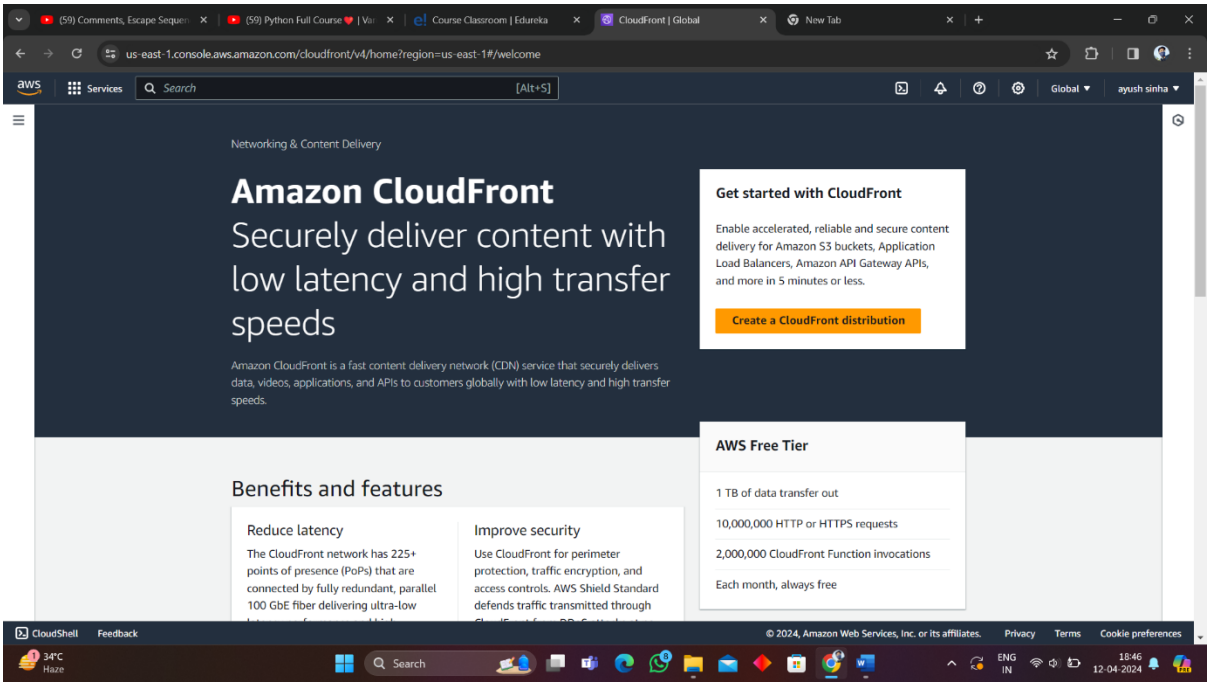
- Configure an Application Load Balancer (ALB) in front of your EC2 instances for load balancing and fault tolerance.



- Enable health checks on the ALB to detect unhealthy instances and route traffic only to healthy instances.

7. **Global Caching**:

- Use Amazon CloudFront as a content delivery network (CDN) to cache your dynamic website globally.



- Configure CloudFront to distribute your website content to edge locations worldwide, reducing latency for users.