AWS Solutions Architect Course-end Project 1

Problem Statement :: Configure and Connect a MySQL Database Instance with a Web Server and Set up the Monitoring of the Solution

Description:

Your organization wants to deploy a new multi-tier application. The application will take live inputs from the employees, and it will be hosted on a web server running on the AWS cloud. The development team has asked you to set up the web server and configure it to scale automatically in cases of a traffic surge, to make the application highly available. They have also asked you to take the inputs from the employees and store them securely in the database.

Real-World Scenario:

Your organization wants to deploy a new multi-tier application. The application will take live inputs from the employees, and it will be hosted on a web server running on the AWS cloud. The development team has asked you to set up the web server and configure it to scale automatically in cases of a traffic surge, to make the application highly available. They have also asked you to take the inputs from the employees and store them securely in the database.

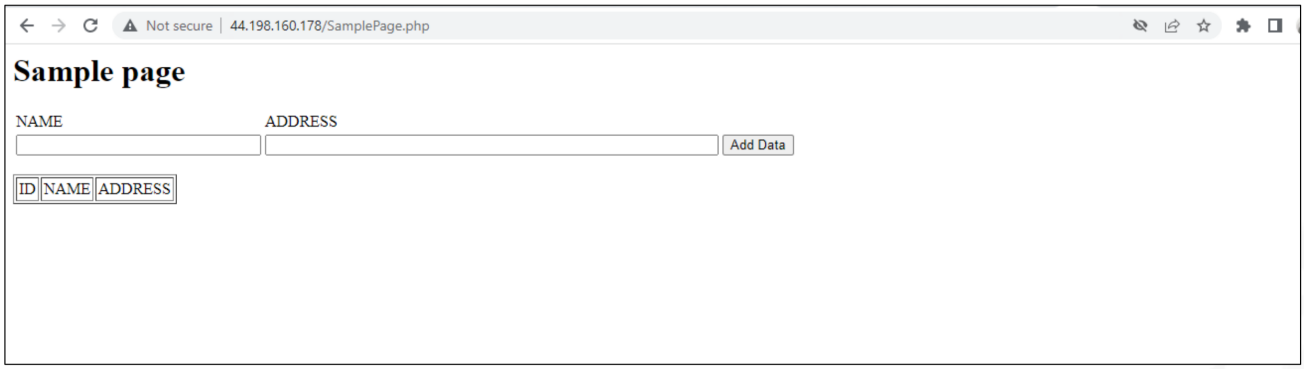
Skills used in the project and their usage in the industry are given below:

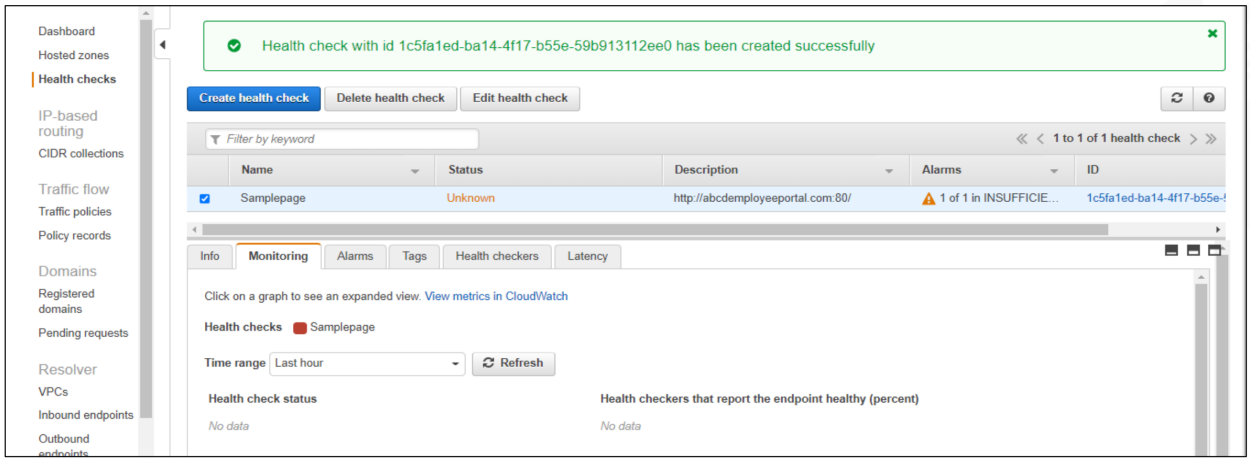
• AWS console - The AWS Management Console is a web application that includes and references several service consoles for managing AWS services.

• EC2 Instance - Amazon EC2 provides a large set of instance types that are customized to certain use cases.

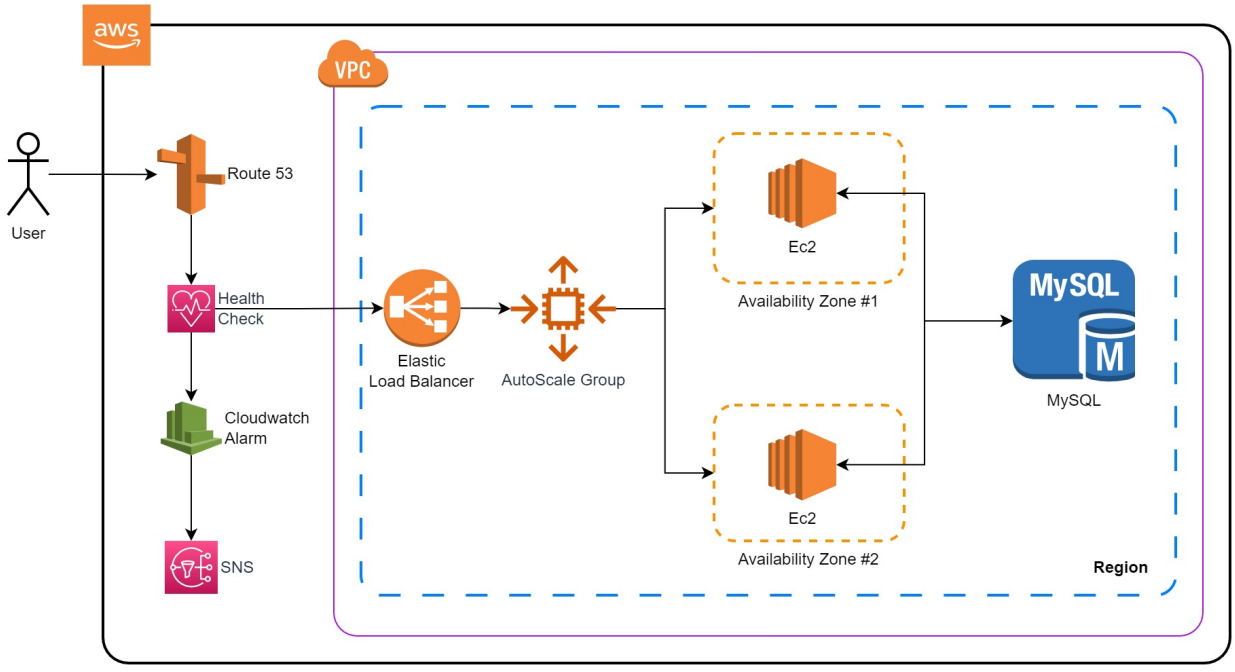
• Apache Web Server - As a Web server, Apache is in charge of accepting directory (HTTP) requests from Internet users and delivering the requested data in the form of files and Web pages.

Expected Output:





Solution Architecture:



1. Architecture Overview: The architecture will consist of an EC2 instance running Apache web server, an RDS MySQL database for secure data storage, and additional services for scalability and availability.

2. EC2 Instances: Set up one or more EC2 instances to host the Apache web server. These instances will handle incoming requests from employees and serve the application. The instances should be launched in an Auto Scaling group to enable automatic scaling based on traffic demand.

3. Auto Scaling: Configure the Auto Scaling group to automatically add or remove EC2 instances based on predefined scaling policies. The policies can be based on metrics such as CPU utilization, network traffic, or request count. This ensures that the application can handle traffic surges by scaling horizontally.

4. Load Balancer: Integrate an Elastic Load Balancer (ELB) with the Auto Scaling group to distribute incoming traffic across the EC2 instances. The ELB acts as a single entry point for employees and improves the availability of the application by distributing requests evenly.

5. RDS MySQL Database: Create an RDS instance running MySQL to securely store employee inputs. Ensure that the database is provisioned with appropriate compute and storage resources based on the expected workload.

6. Security: Implement security measures such as securing the EC2 instances with appropriate firewall rules, using SSL certificates for secure communication, and applying necessary access controls to the RDS instance. Follow AWS security best practices to protect the application and data.

7. Monitoring and Logging: Configure CloudWatch to monitor key metrics of the EC2 instances, RDS database, and load balancer. Set up alarms to notify you in case of any issues or breaches. Enable logging to capture relevant logs for troubleshooting and auditing purposes.

8. Backup and Disaster Recovery: Implement regular automated backups for the RDS database to ensure data durability. Consider configuring a Multi-AZ deployment for RDS to provide high availability and automatic failover in case of a failure in the primary database.

9. Scaling Considerations: Analyze the application's performance and determine the optimal scaling thresholds based on expected traffic patterns. Regularly review and adjust the scaling policies to ensure efficient resource utilization and cost optimization.

10. Route 53: Integrate Route 53, the DNS service provided by AWS, into the architecture. Route 53 will allow you to route incoming traffic to the application's Elastic Load Balancer (ELB) and provide health checks for improved availability.

11. DNS Configuration: Create a Route 53 hosted zone for your domain and configure the necessary DNS records. Set up an "A" record or a "CNAME" record to point to the ELB's DNS name. This ensures that when employees access the application using the domain, their requests are directed to the ELB.

12. Health Checks: Configure Route 53 health checks to monitor the health of the ELB and the EC2 instances. Define health check settings to periodically send requests to the ELB and evaluate its responses. If the health checks fail for an instance or the ELB, Route 53 can automatically stop routing traffic to the unhealthy resource.

13. Routing Policies: Define routing policies in Route 53 to control the distribution of traffic among healthy resources. There are several routing policies available, such as simple, weighted, latency-based, geolocation-based, and failover. Choose the appropriate routing policy based on your requirements.

14. Simple Routing Policy: In the context of this solution, you can use the "Simple" routing policy in Route 53. With the Simple routing policy, Route 53 responds to DNS queries with all the IP addresses

associated with the ELB. The client's DNS resolver chooses one of the IP addresses randomly. This provides a basic load balancing capability across the available EC2 instances.

15. Health-Based Routing: Additionally, you can combine the Simple routing policy with health checks. Route 53 will only include IP addresses of the ELB associated with healthy EC2 instances in the DNS response. This further enhances the availability and reliability of the application by ensuring that traffic is only directed to healthy instances.

16. Deployment Automation: Utilize AWS CloudFormation or other deployment automation tools to define the infrastructure as code, enabling easy replication and updates of the application stack.

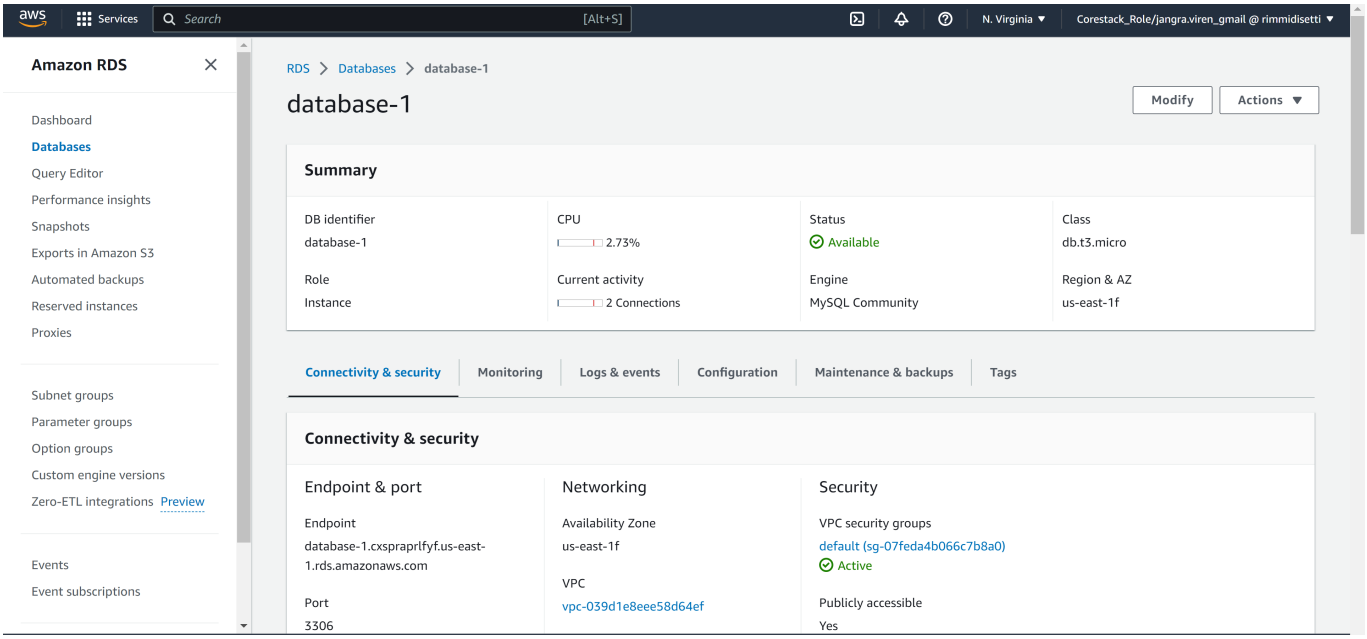
Posible ways of achieving the same:

1. Using Web Console and manually configure resources

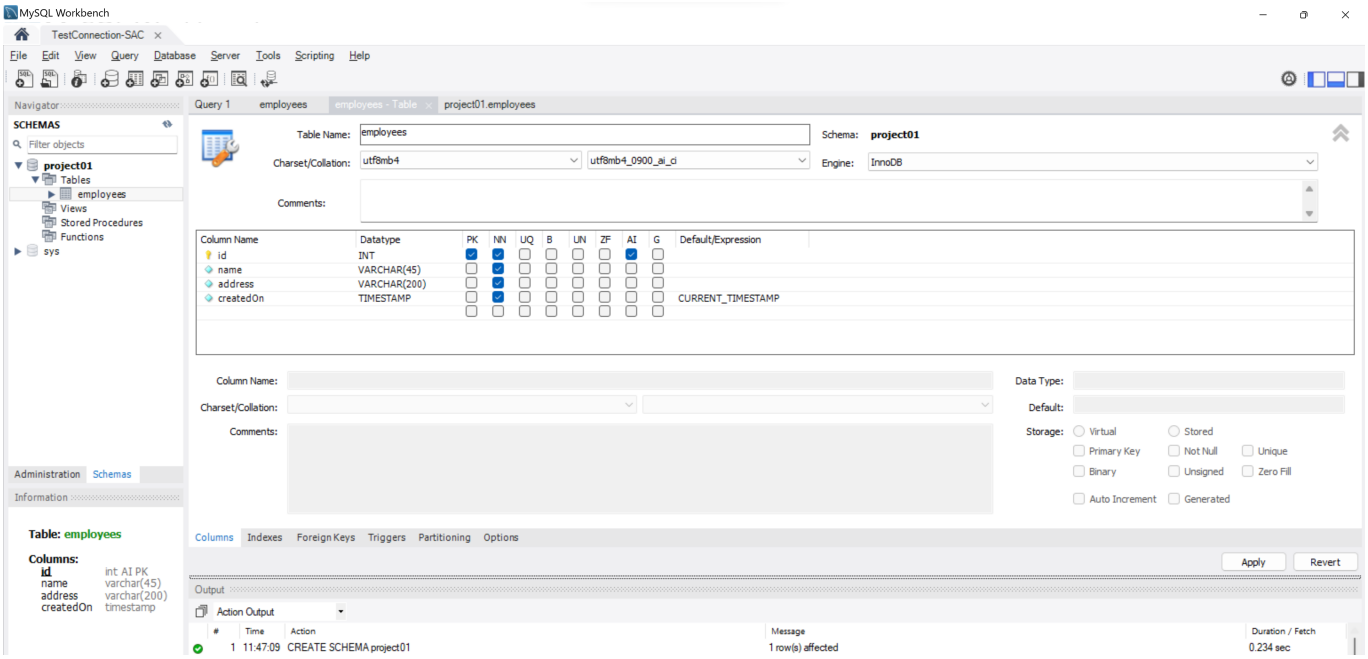
2. Write CloudFormation templates, use AWS CLI and deploy through cloudformation templates.

Project Implementation:

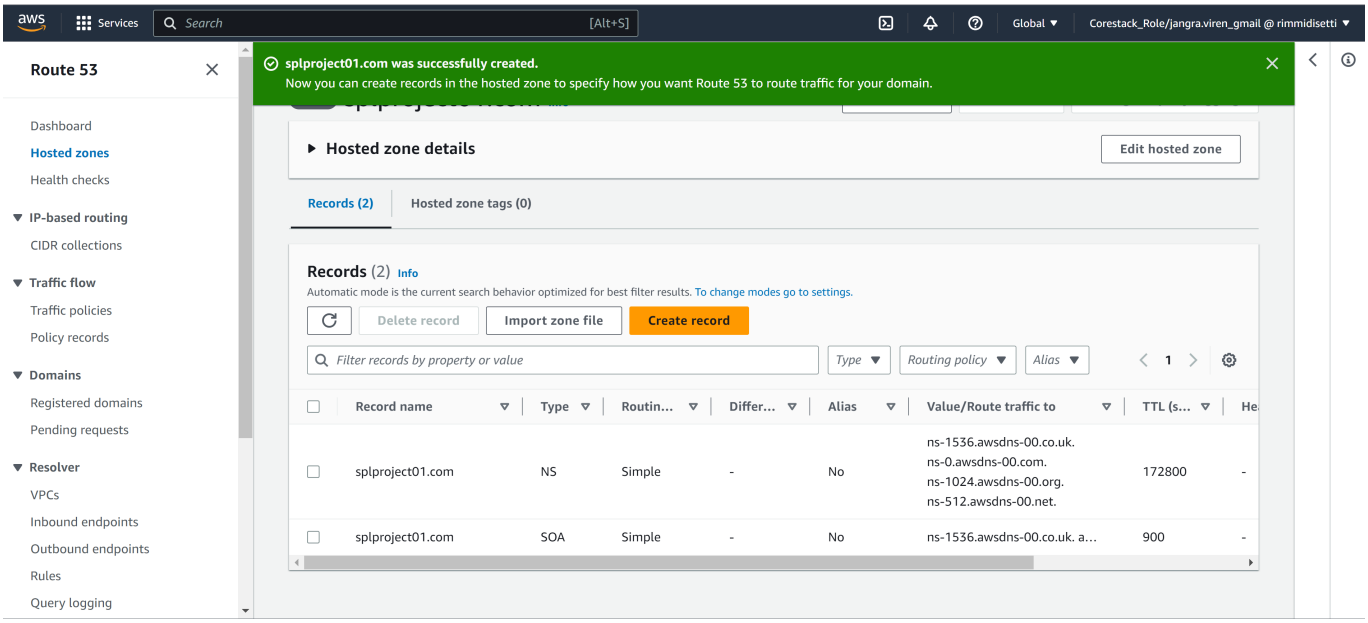
1. RDS MySQL Instance is created opening the port 3306. In real scenarios we can have backups, snapshots and additional users for RDS access. Direct opening of the 3306 port from public won't be a good option and we should go for a Jump Server which would be inside the same VPC/Network.



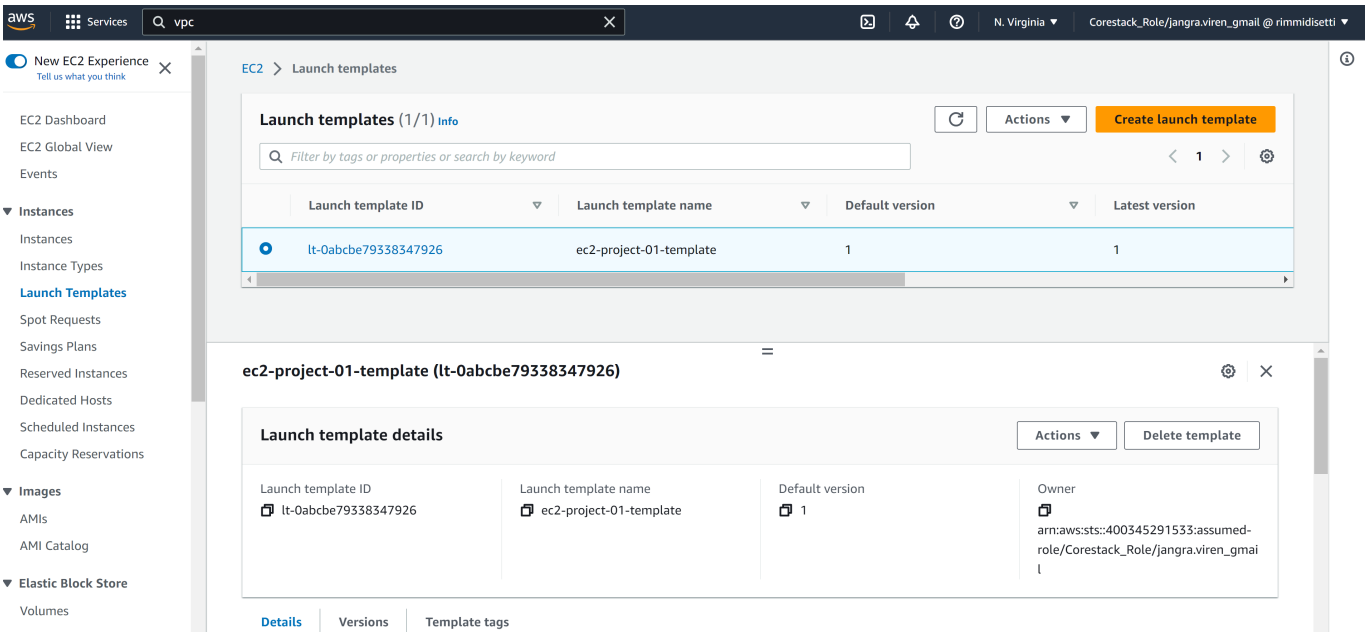
2. Create a project schema/database and the required table with required columns.



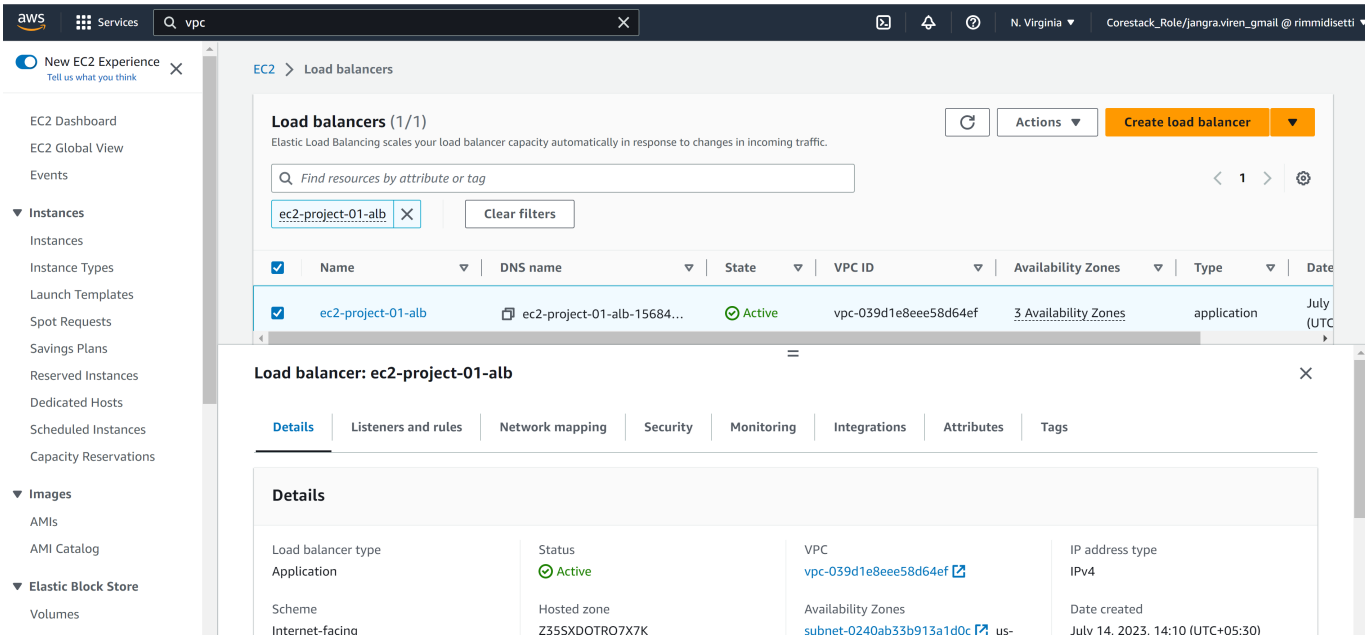
3. Hosted Zone Created: Actual hosted zone can be public and would be actual domain based. Also health check points are created for the DNS which can check the



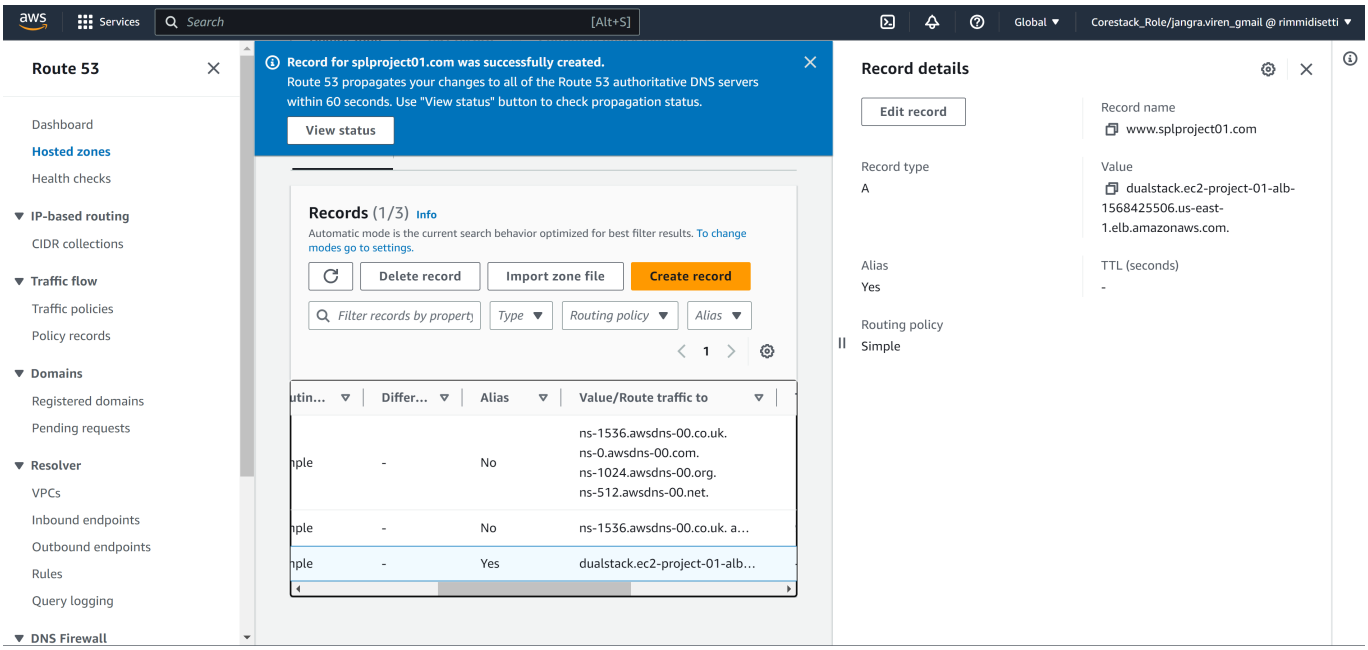
4. Ec2 launch template created with user data script for the required files. Script includes installing apache, php and required package to connect php application to mysql database. Apart from that application files like index.html, save_data.php, get_data.php and a config.php is included in the script. Although in real world scenarios and environment based scenarios, we would use AWS Secrets manager and Database proxy for the db connectivity. But just for this project, we store credentials in plain config file.



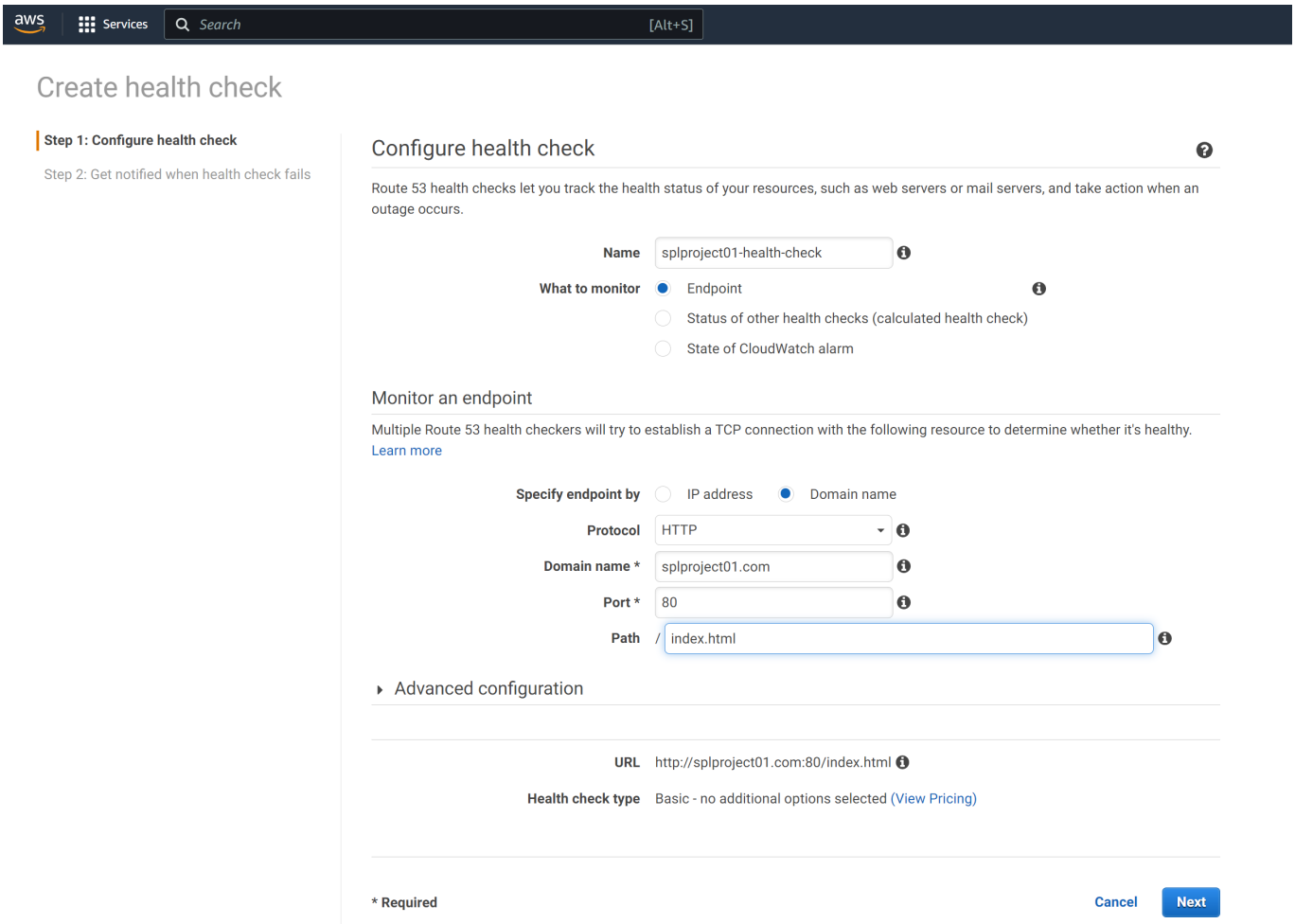
5. Application load balancer is created with target group on port 80.

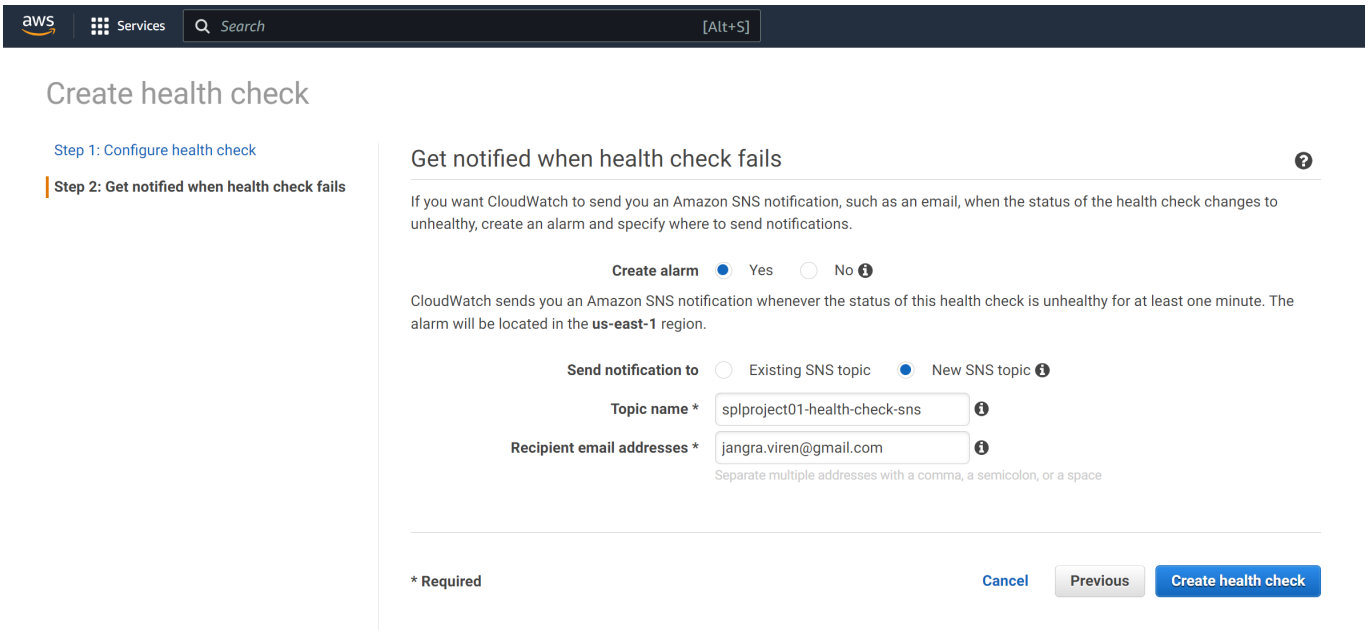


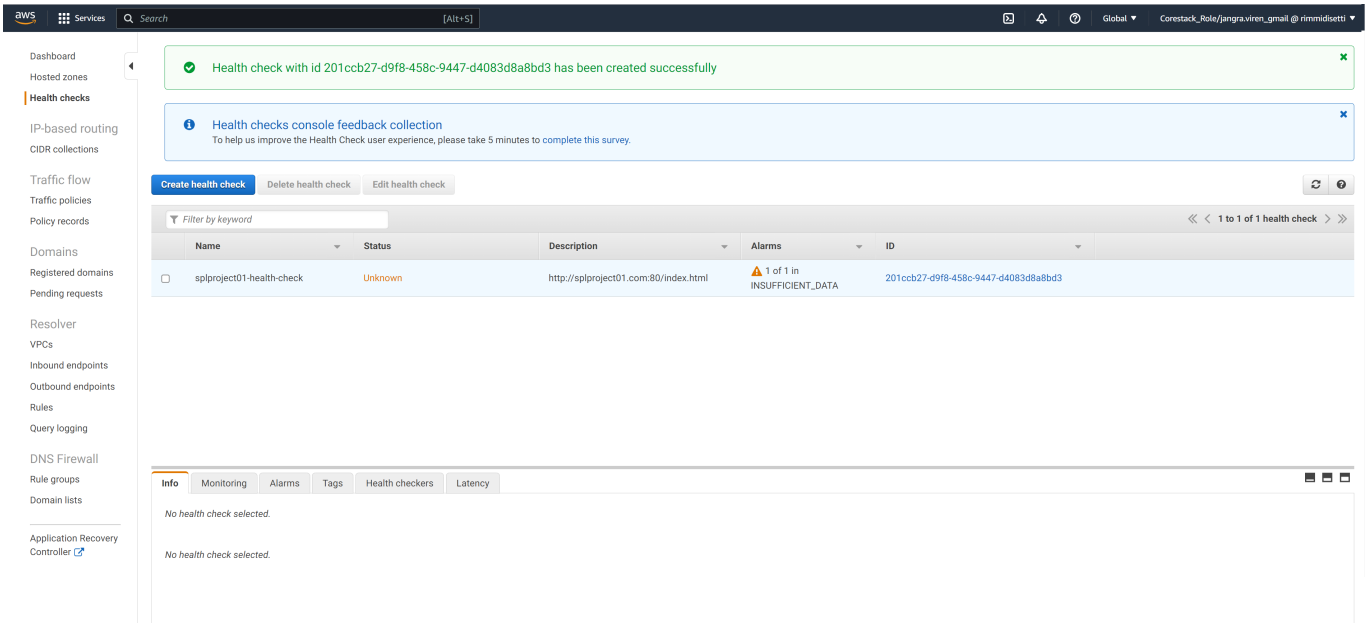
6. Alias record for the load balancer is created in the hosted zone. So that this domain route all the traffic to the load balancer.



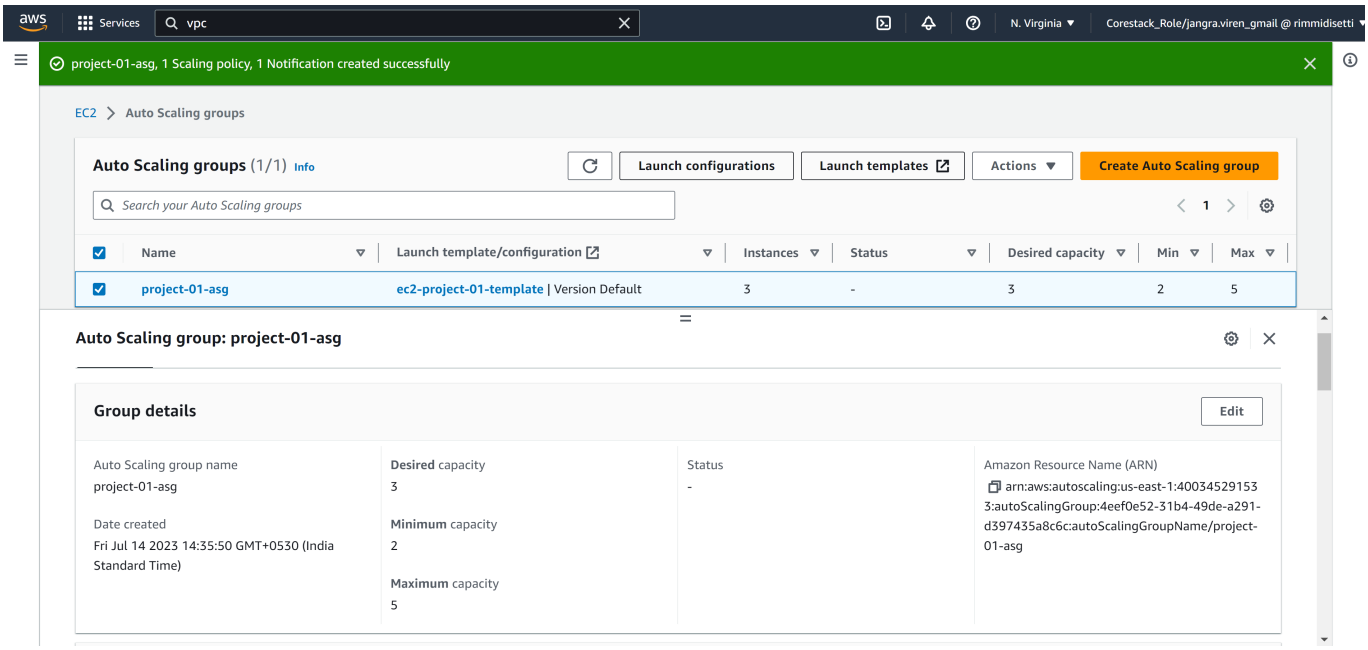
7. DNS Routing Policy created with health check. So that Route 53 will only include healthy EC2 instances in the Load Balancer in the DNS response. This further enhances the availability and reliability of the application by ensuring that traffic is only directed to healthy instances



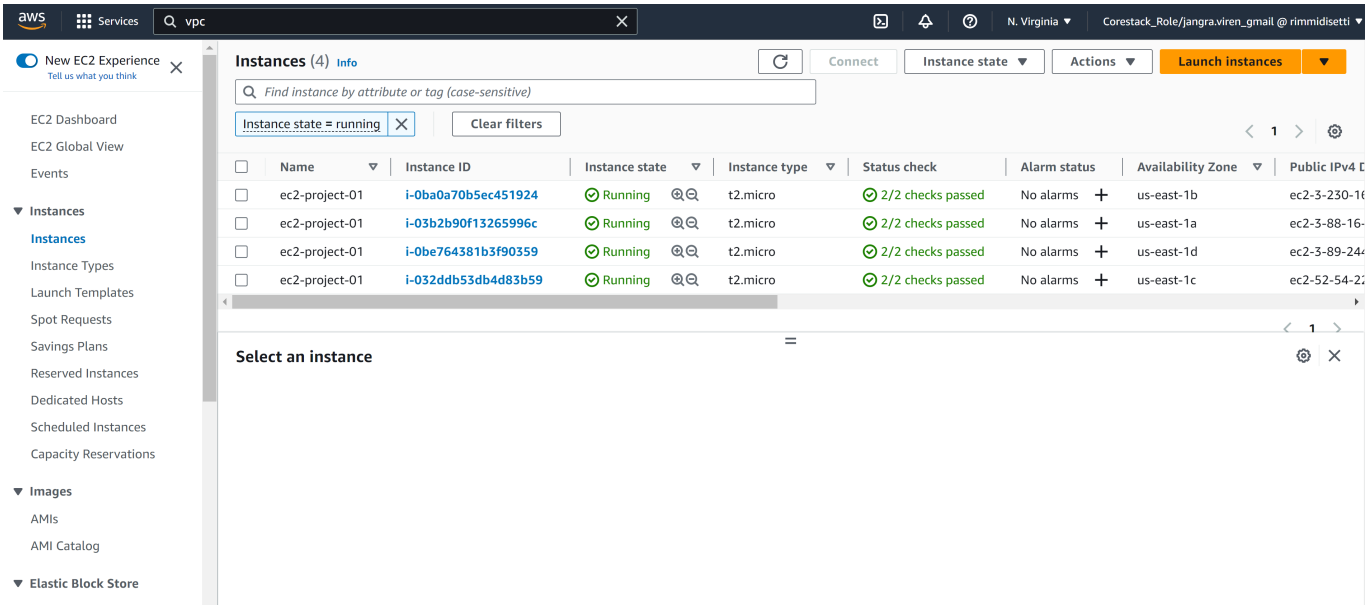


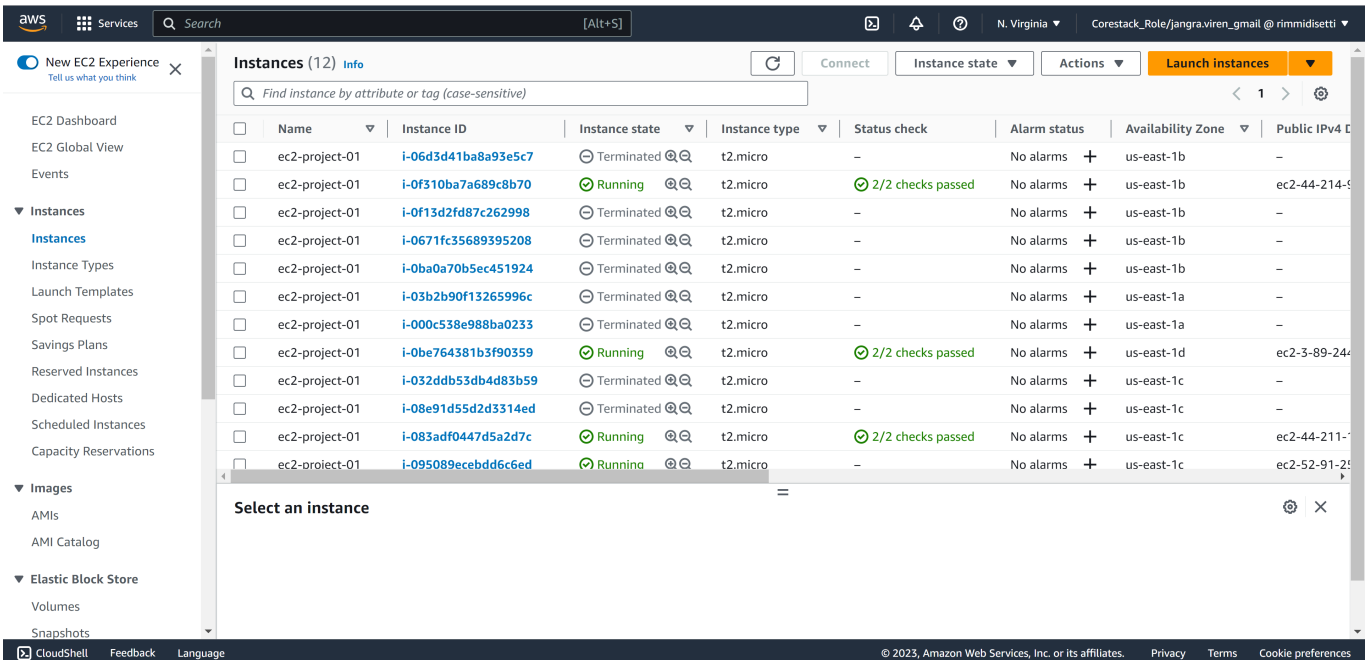


8. AutoScaling group is created with the required configuration. Desired capacity, min, max instances and policy could be any based on the real world situation.

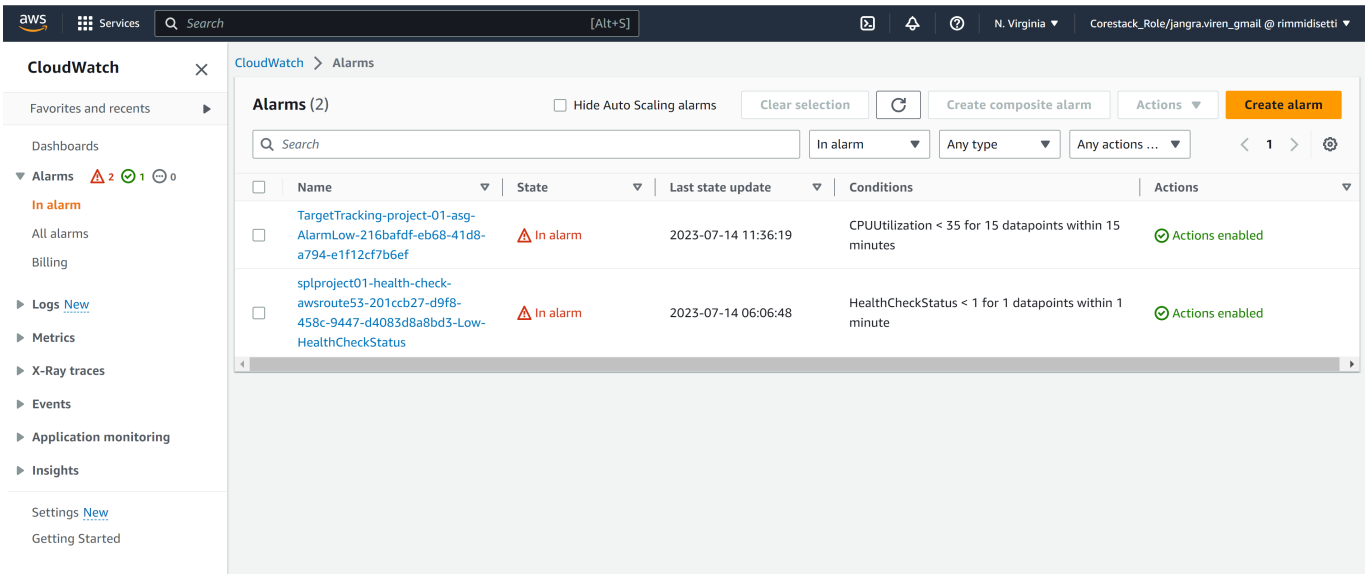


9. Required instances automatically created and deleted based on the auto scaling configuration.

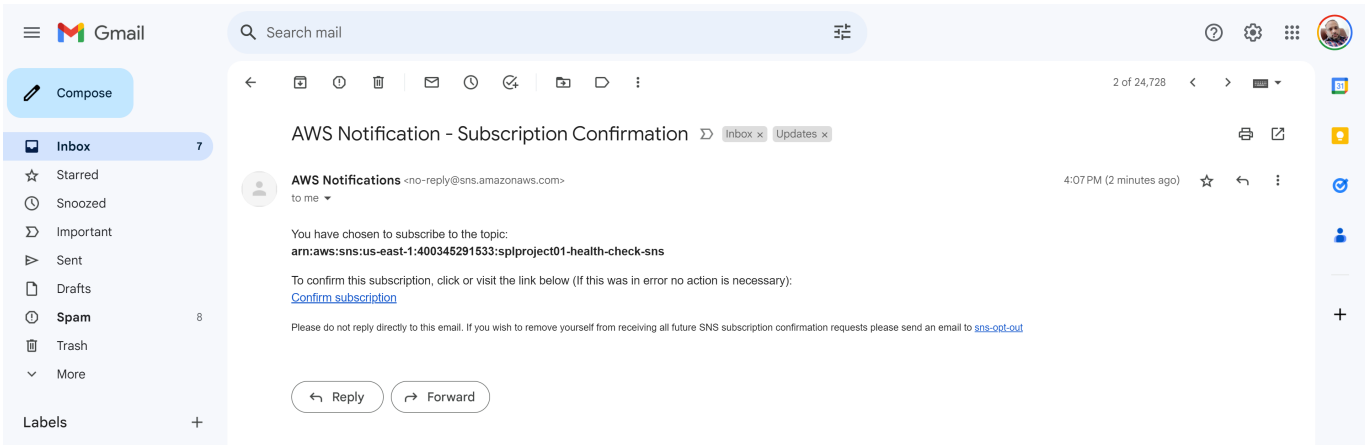




10. Cloudwatch Alarm take relevant action whenever policy configuration/rule executed.

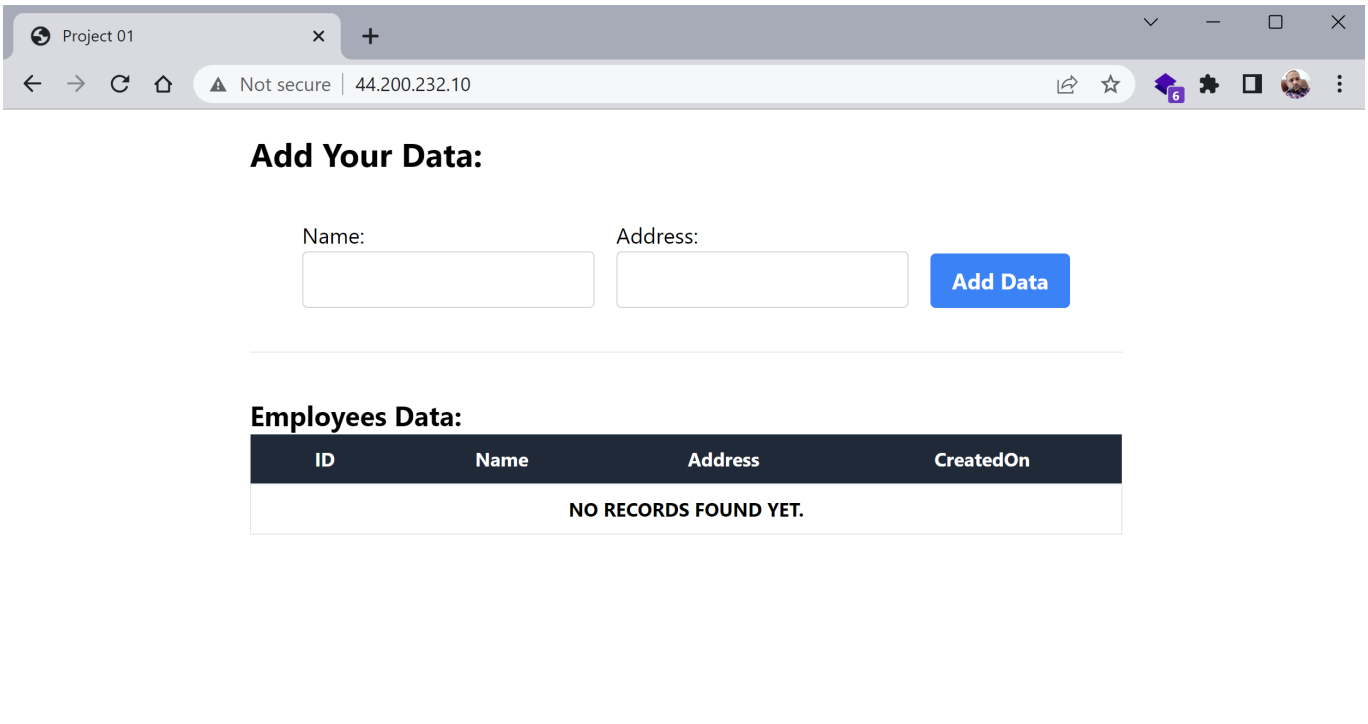


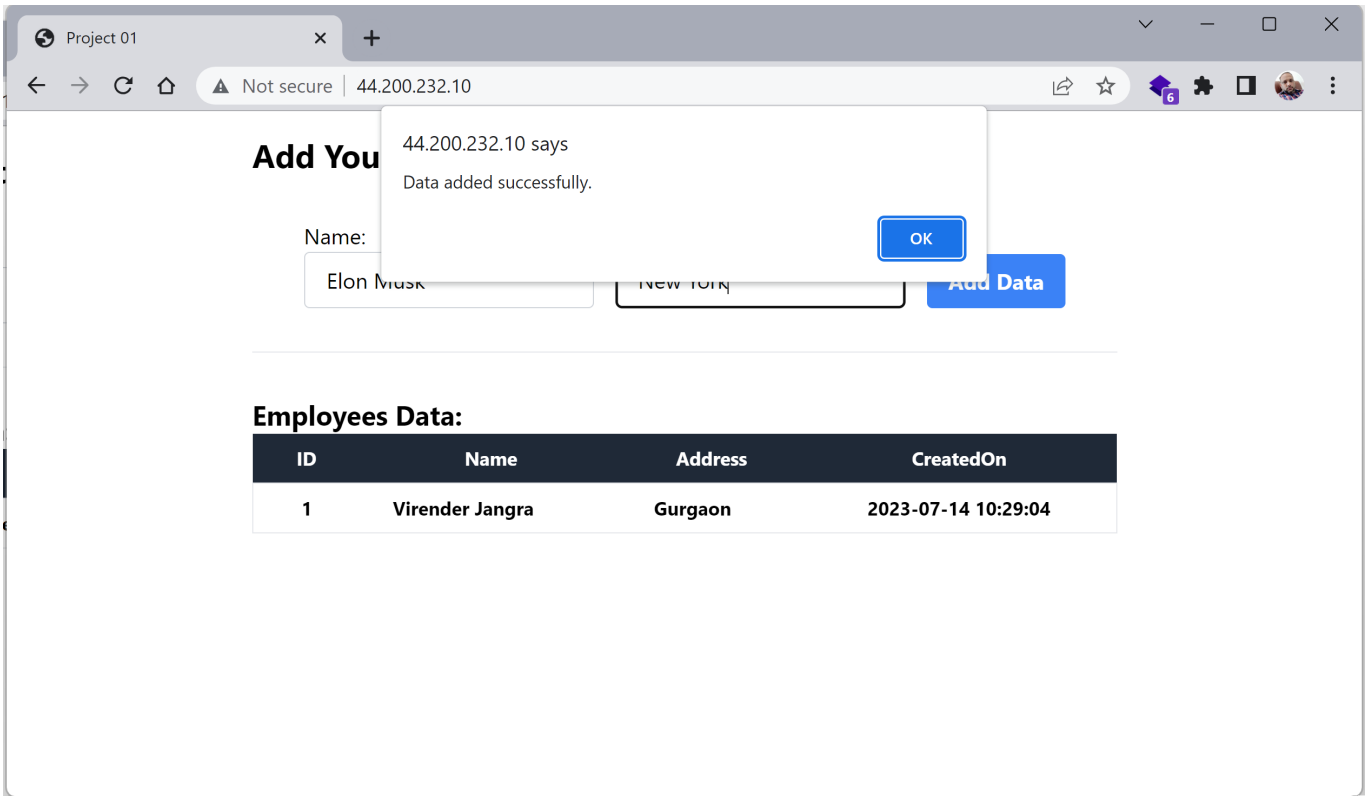
11. SNS Topic suscription email goes to the required consumer.

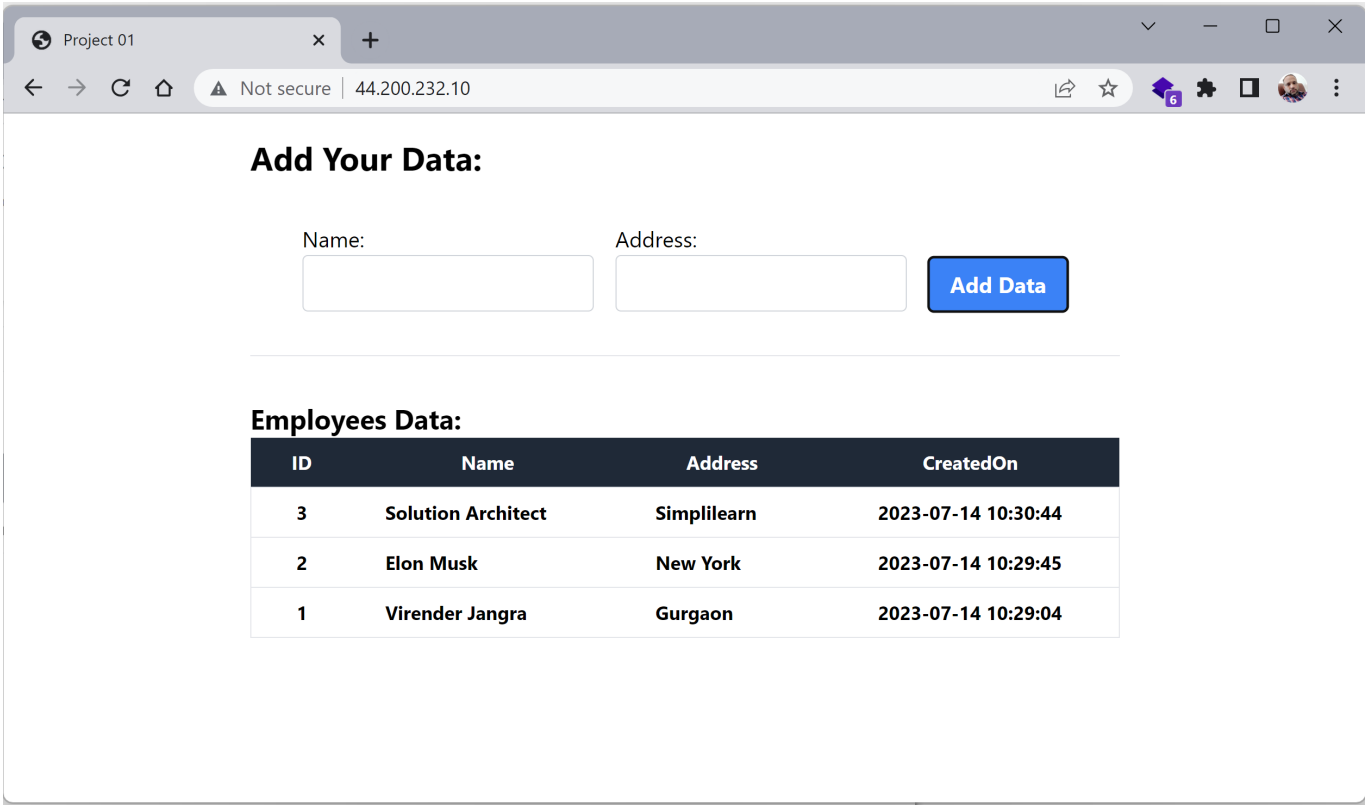


12. Application is created as per the required logic. Name and Address taking post form is included. Ajax is used to submit the form data to the database. Employees datatable is created to list the submitted

database. Different instances showing the same data inserted.






Project 01
Not secure
44.200.232.10
Add Your Data:
Name:
Address:
Add Data
Employees Data:
ID
Name
Address
CreatedOn
3
Solution Architect
Simplilearn
2023-07-14 10:30:44
2
Elon Musk
New York
2023-07-14 10:29:45
1
Virender Jangra
Gurgaon
2023-07-14 10:29:04