गुरुकुल काँगड़ी (समविश्वविद्यालय), हरिद्वार, उत्तराखण्ड

Gurukula Kangri (Deemed to be University), Haridwar, Uttarakhand

(Formerly Gurukula Kangri Vishwavidyalaya)

Mini-Project
Report File

Submitted By

Ankit Raj
Reg No. 226303007
CSE VIth Sem

Submitted To

Head of Department
Dr. Mayank Aggarwal
(CS&Engg Department)

Mini-Project

On

“The cloud deployment architecture on AWS”

Computer Science Engineering

(Batch 2021 – 2025)

Acknowledgement

- I would like to express my sincere gratitude to my university for providing the necessary resources and facilities for the completion of this project.
- I am deeply grateful to my classmates/peers for their assistance, feedback, and collaborative efforts during the various stages of this project.
- I extend my appreciation to the online community and forums for providing helpful insights, troubleshooting tips, and code snippets that aided in overcoming technical challenges.
- I am thankful to my friends and family for their understanding, encouragement, and patience during the time devoted to this project.
- Special acknowledgment goes to Visual Studio Code for facilitating the development process and enhancing the efficiency of the project.
- I am grateful to the authors of the textbooks, online tutorials, and documentation that served as valuable resources for learning and implementing the necessary concepts and techniques.
- Finally, I would like to thank everyone who contributed directly or indirectly to the successful completion of this project, your support has been instrumental in achieving the desired outcomes.

Table of Content

Project Details

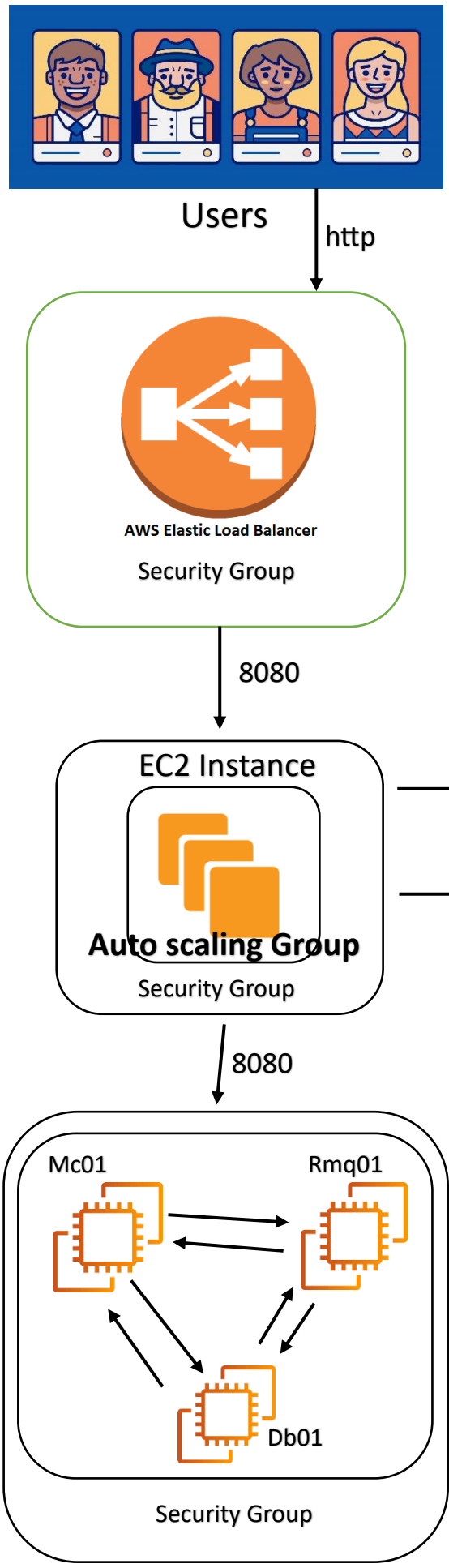
The cloud deployment architecture

- The objective of this project is to deploy a multi-service application on Amazon Web Services (AWS) using EC2 instances. The application comprises four EC2 instances With Elastic Load Balancer and autoscaling facility, each hosting a specific service

    - **app01**: Hosting the application and responsible for deployment and interaction.
    - **mc01**: Hosting Memcached service.
    - **db01**: Hosting MariaDB service.
    - **rmq01**: Hosting RabbitMQ service.
    - **Elastic Load Balancer (ELB).** : for handling incoming traffic and distributing it across multiple instances.
    - **Auto Scaling :** Auto Scaling for the instance to automatically adjust the number of instances based on predefined metrics such as CPU utilization

Software Requirement

- AWS Account
- Create 4 instance in EC2 Service
- Elastic Load balancer
- Route 53
- Auto Scaling Service
- Mariadb, Tomcat9, RabbitMQ, Memcached
- A demo project for deployment


Users
http
AWS Elastic Load Balancer
Security Group
8080
EC2 Instance
Auto scaling Group
Security Group
S3 Bucket
Route 53
DNS
DNS Private Zone
Mc01 = ip
Rmq01 = ip
Db01 = ip
8080
Mc01
Rmq01
Db01
Security Group

Work Plan

There are certain tasks we must perform

There are following steps

Step 1: - Create a Key pair

Step 2: - Create 3 Security Group for connecting the following server mention the following details
Allow the ports and protocol

Step 3: - Create 4 instance with following Specification on EC2

- App01 With Ubuntu AMI or OS
- Db01 with almalinux
- Mc01 with almalinux
- Rmq01 with almalinux

Add userdata for provision
Note: I attach userdata below

Step 4: - Goto : Route S3 Services
Create Host Zone
Connect db01 Mc01 Rmq01 with Private ip

Step 5: - Deploy Project on App01 and start the Service

Step 6: - Create ELB and target the app01 instances

Step 7: - Configure autoscaling
By create AMI of app01
And launch templates

After few minutes its live over the internet

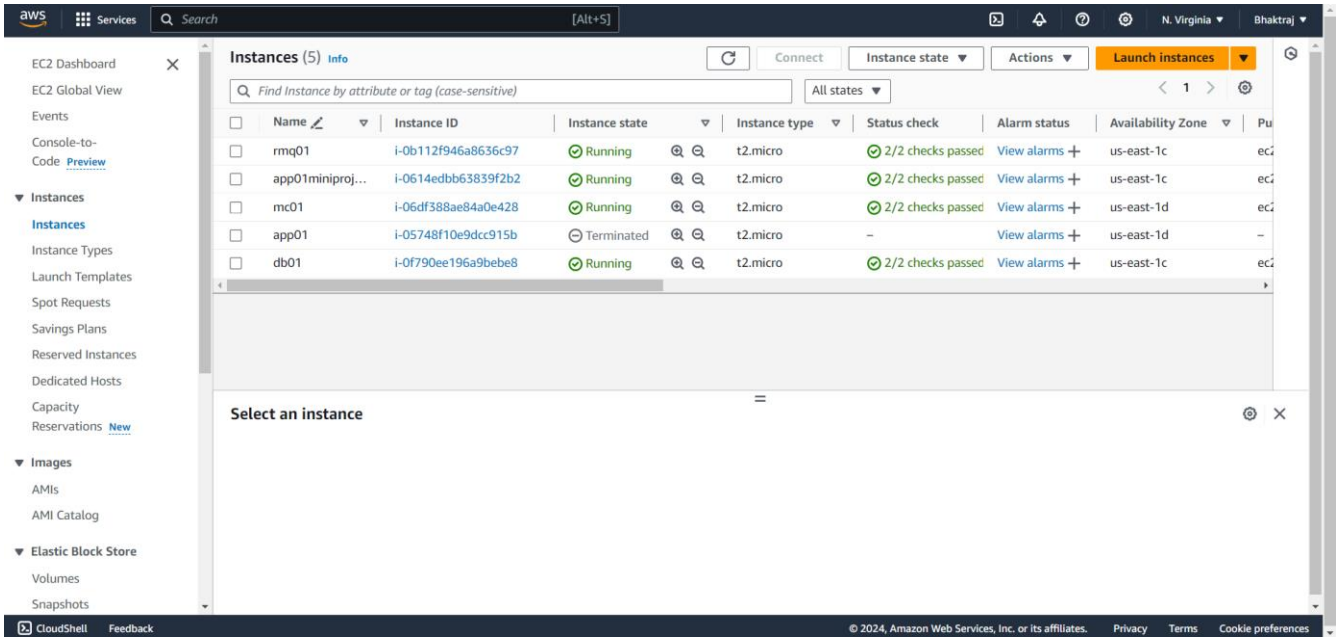
Components and Technologies Used:

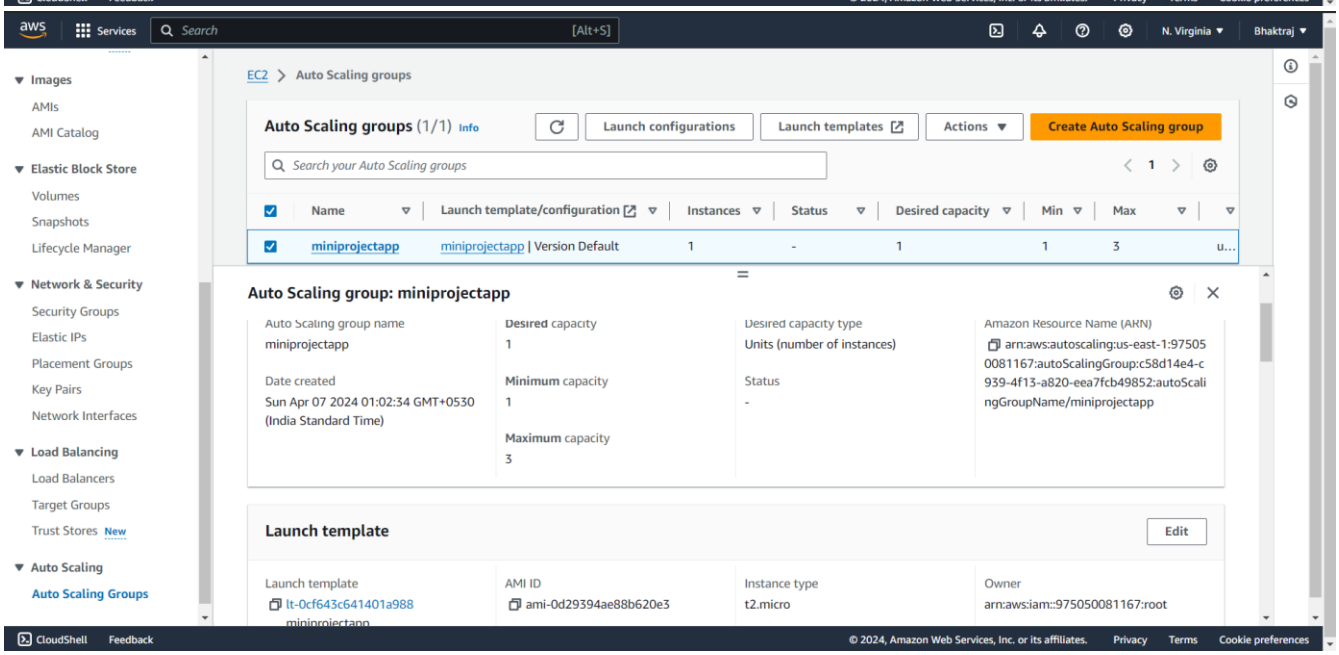
Amazon EC2: Utilized for hosting the application and services.
Tomcat 9: Hosts the application on the app01 instance.
Memcached: Utilized for caching purposes, hosted on the mc01 instance.
MariaDB: Used as the backend database service, hosted on the db01 instance.
RabbitMQ: Enables message queue functionality for asynchronous communication, hosted on the rmq01 instance.
Route 53: Facilitates DNS management and the creation of hosted zones for inter-instance communication.
Elastic Load Balancer (ELB): Handles incoming traffic and distributes it across multiple instances, providing high availability and scalability.
Auto Scaling : Implement Auto Scaling for the app01 instance to automatically adjust the number of instances based on predefined metrics such as CPU utilization, incoming traffic, or custom application-specific metrics

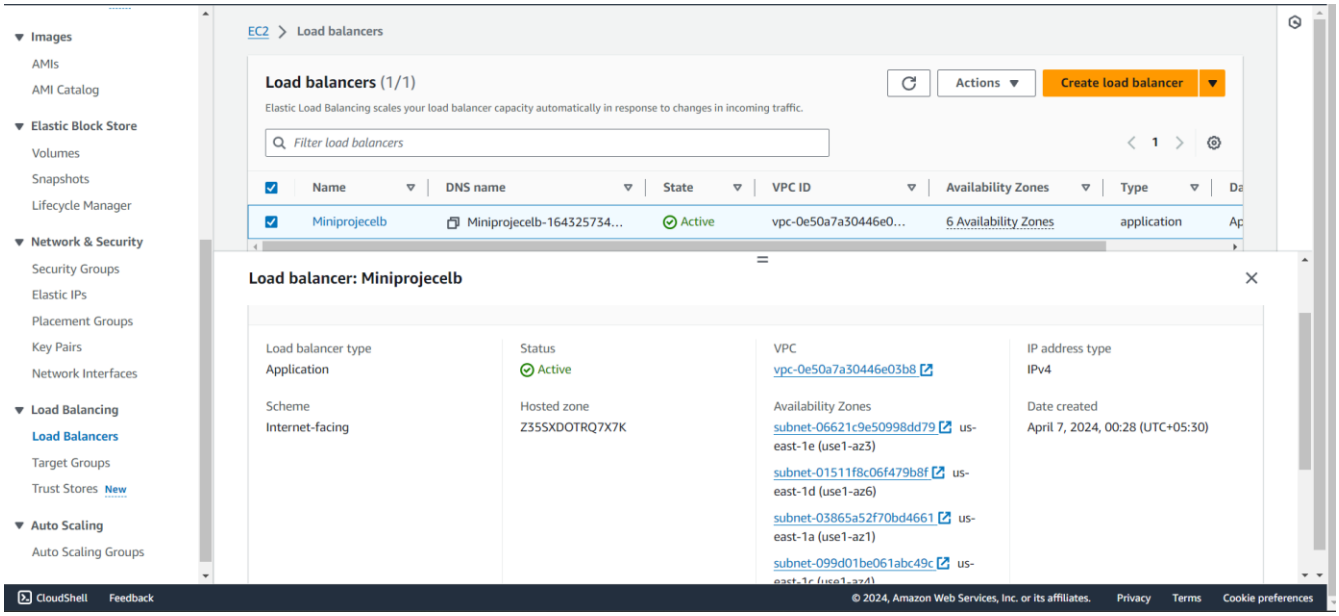
Userdata file for create instance file:

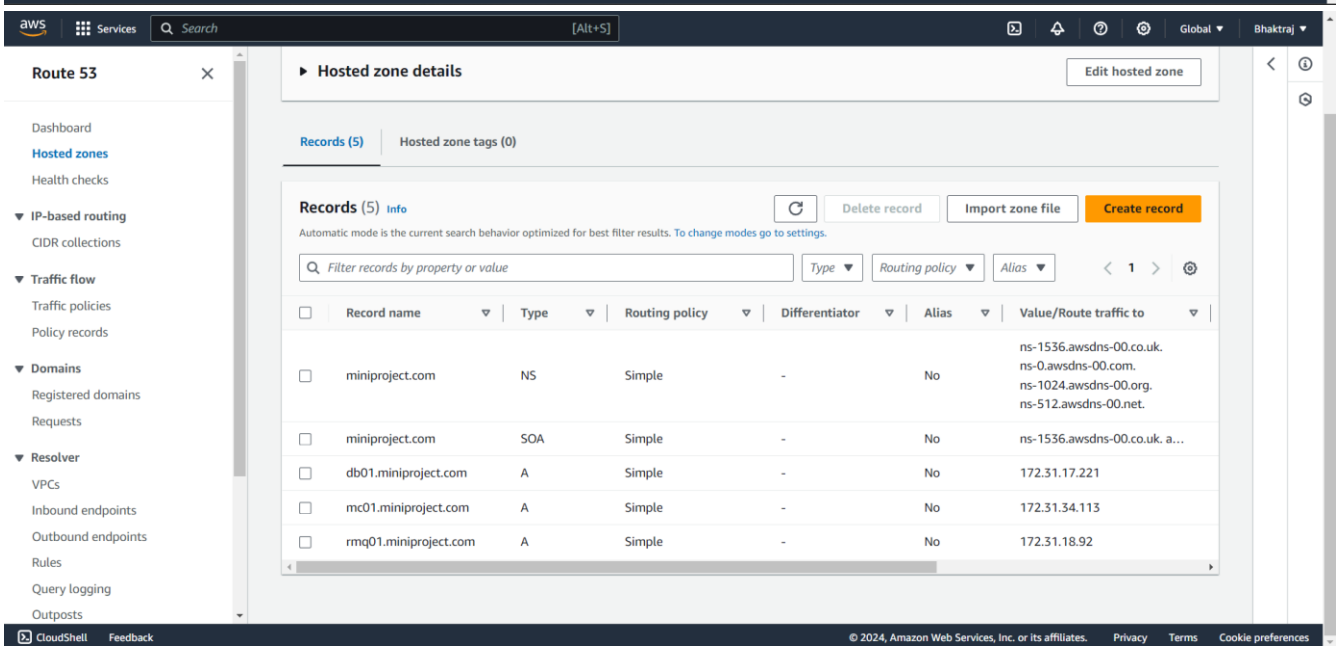
Qr code :

Running Screenshot:-






Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores New
Auto Scaling
Auto Scaling Groups
EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
Miniprojecelb
Miniprojecelb-164325734...
Active
vpc-0e50a7a30446e0...
6 Availability Zones
application
Load balancer: Miniprojecelb
Load balancer type
Application
Status
Active
VPC
vpc-0e50a7a30446e03b8
IP address type
IPv4
Scheme
Internet-facing
Hosted zone
Z35SXDOTRQ7X7K
Availability Zones
subnet-06621c9e50998dd79 us-east-1e (use1-az3)
subnet-01511f8c06f479b8f us-east-1d (use1-az6)
subnet-03865a52f70bd4661 us-east-1a (use1-az1)
subnet-099d01be061abc49c us-
Date created
April 7, 2024, 00:28 (UTC+05:30)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
Global
Bhaktraj
Route 53
Dashboard
Hosted zones
Health checks
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
Outposts
Hosted zone details
Edit hosted zone
Records (5)
Hosted zone tags (0)
Records (5) Info
Delete record
Import zone file
Create record
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.
Filter records by property or value
Type
Routing policy
Alias
Record name
Type
Routing policy
Differentiator
Alias
Value/Route traffic to
miniproject.com
NS
Simple
-
No
ns-1536.awsdns-00.co.uk.
ns-0.awsdns-00.com.
ns-1024.awsdns-00.org.
ns-512.awsdns-00.net.
miniproject.com
SOA
Simple
-
No
ns-1536.awsdns-00.co.uk. a...
db01.miniproject.com
A
Simple
-
No
172.31.17.221
mc01.miniproject.com
A
Simple
-
No
172.31.34.113
rmq01.miniproject.com
A
Simple
-
No
172.31.18.92
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Conclusion

In conclusion, the successful deployment of the multi-service application on Amazon Web Services (AWS) EC2 instances marks a significant achievement in addressing the challenges posed by complex web application architectures. Through meticulous planning, deployment, and configuration, the project has demonstrated the feasibility of hosting and managing a diverse array of services including Tomcat, Memcached, MariaDB, and RabbitMQ within the AWS environment.

By leveraging AWS services such as Auto Scaling, Elastic Load Balancer (ELB), and Route 53, the project has effectively addressed key requirements including scalability, reliability, and cost-effectiveness. The implementation of Auto Scaling for the application instance (app01) ensures dynamic resource allocation based on traffic patterns, enabling the application to seamlessly handle fluctuations in demand while optimizing cost-efficiency.

Furthermore, the project has emphasized the importance of performance optimization through the utilization of caching mechanisms, efficient database management, and asynchronous communication, thereby enhancing the responsiveness and reliability of the application.

Looking ahead, the insights gained from this project lay a solid foundation for further enhancements and optimizations. Continuous monitoring, evaluation, and refinement of the infrastructure and application architecture will be essential to adapt to evolving business requirements and technological advancements.

Overall, the successful execution of this project underscores the value of leveraging cloud-native solutions and best practices to build scalable, resilient, and cost-effective web applications that meet the demands of modern businesses in the digital era.