AWS Cloud Application Design Challenge

Project

Submitted to:

Mr. Zunnurain Hussain

Group Members:

Muhammad Usman Azeem 20L-2064

Zaid Khan 20L-2072

National University Of Computer and Emerging Sciences
Department of Computer Science
Lahore, Pakistan

1. Introduction

In this assignment, our team was tasked with designing a three-tier web application to be hosted on the AWS cloud. The primary objective was to ensure that our design adheres to the five pillars of a well-architected architecture, focusing on reliability, security, performance efficiency, cost optimization, and operational excellence. By the help of previous labs performed we managed to design and deploy web application successfully on AWS. All steps involved in designing and creating the web application were recorded and screenshots were taken on the regular interval. To ensure fault tolerance we applied appropriate techniques to ensure maximum availability of our web application and minimal downtime. Our design aims to leverage AWS services to achieve scalability, high availability, and fault tolerance.

2. Planning Phase

In the planning phase we initially assigned tasks among team members and the team members contributed successfully in the achievement of milestones. the contributions are as follows:

2.1 Team Member Contributions

The team member contributions are described below:

2.1.1 Team Member Zaid's Contribution:

- Zaid took charge of setting up the Virtual Private Cloud (VPC) on AWS, ensuring that it's properly configured to meet our project requirements. This involved defining the subnets, routing tables, and internet gateway to establish network connectivity within the AWS environment.
- Zaid was responsible for visually representing our system architecture using Draw.io. This included depicting the various components of our three-tier web application, their interactions, and how they are deployed on AWS.
- He ensured the web app is able to configure itself properly and all the requirement were met.

2.1.2 Team Member Usman's Contribution:

- Usman worked on setting up a Multi-AZ Relational Database Service (RDS) instance on AWS, ensuring data reliability and fault tolerance by replicating the database across multiple Availability Zones (AZs).
- He handled the configuration of Elastic Load Balancers (ELB) and Auto Scaling groups for both the web and application tiers. This ensured that our application can handle varying levels of traffic efficiently and maintain high availability by automatically adjusting resources based on demand.
- He executed the necessary scripts to deploy the application on the web tier, ensuring that it was done accurately and efficiently.

2.1.3 Reflecting on Teamwork and Individual Contributions:

Our team collaborated effectively to design and deploy the three-tier web application on AWS. Each team member contributed their expertise and skills to different aspects of the project, resulting in a well-architected and robust solution. While one member demonstrated strong leadership in network configuration and architectural visualization. His expertise in VPC setup and diagram designing provided a solid foundation for our project. Other member’s contributions were instrumental in ensuring the reliability, scalability, and high availability of our application. His work on implementing the RDS database, setting up load balancers, and managing auto scaling groups significantly enhanced the performance and fault tolerance of our system.

Overall, our teamwork enabled us to leverage AWS services effectively to achieve the objectives of reliability, security, performance efficiency, cost optimization, and operational excellence. By dividing responsibilities according to each member's strengths and expertise, we were able to design and deploy a successful web application on the AWS cloud.

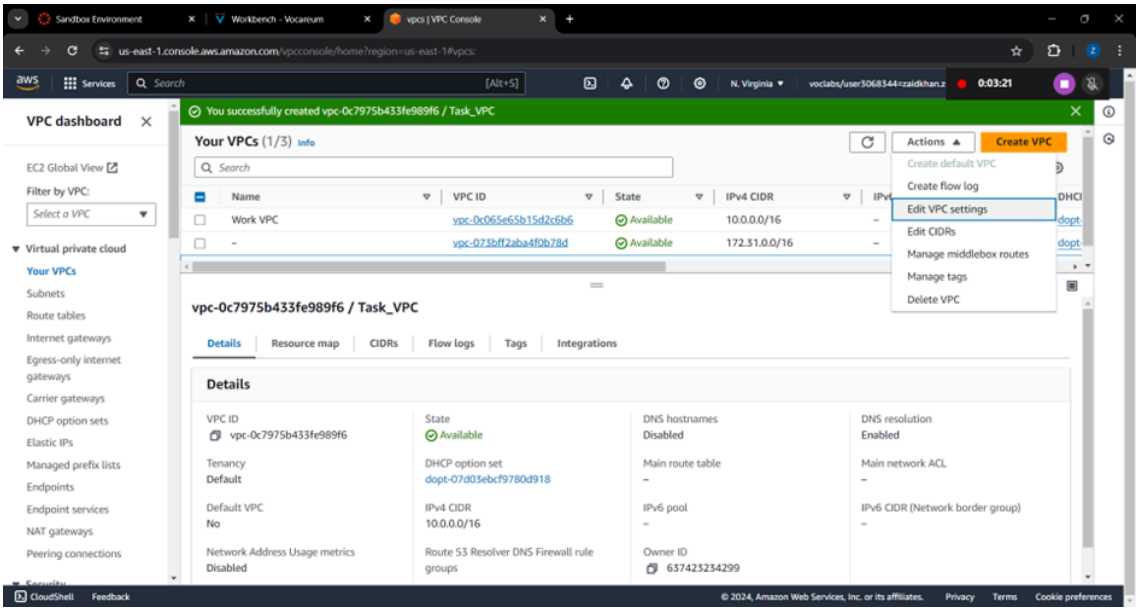
Our team collaborated effectively, leveraging each member's expertise to fulfill the requirements of the task. Regular communication and coordination were key to ensuring smooth progress and resolving any challenges encountered during the implementation phase.

2.2 Working Phase

The working on deployment of web application is described below:

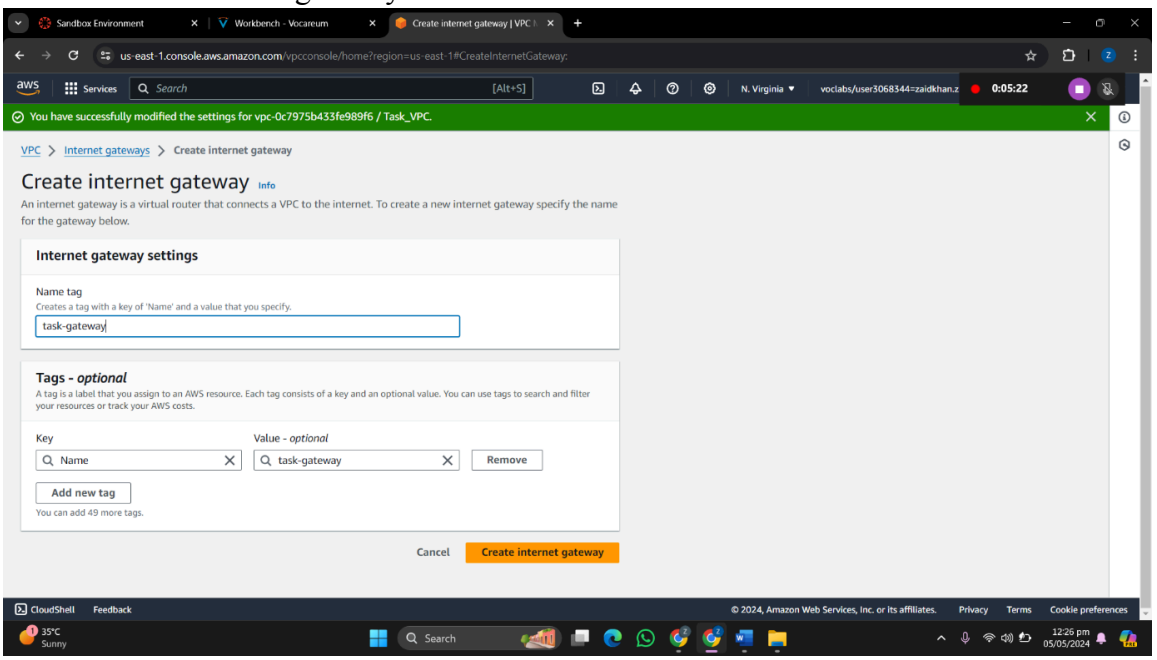
2.2.1 Creation of VPC

We started our working by firstly creating a VPC (Virtual Private Cloud) that would enable to connect resources and manage them securely in the environement. It would also provide isolation and control over the infrastructure. We chosed the option of Create VPC Only.



2.2.2 Internet Gateway Creation

We created an Internet gateway and attached it to the VPC we created earlier.

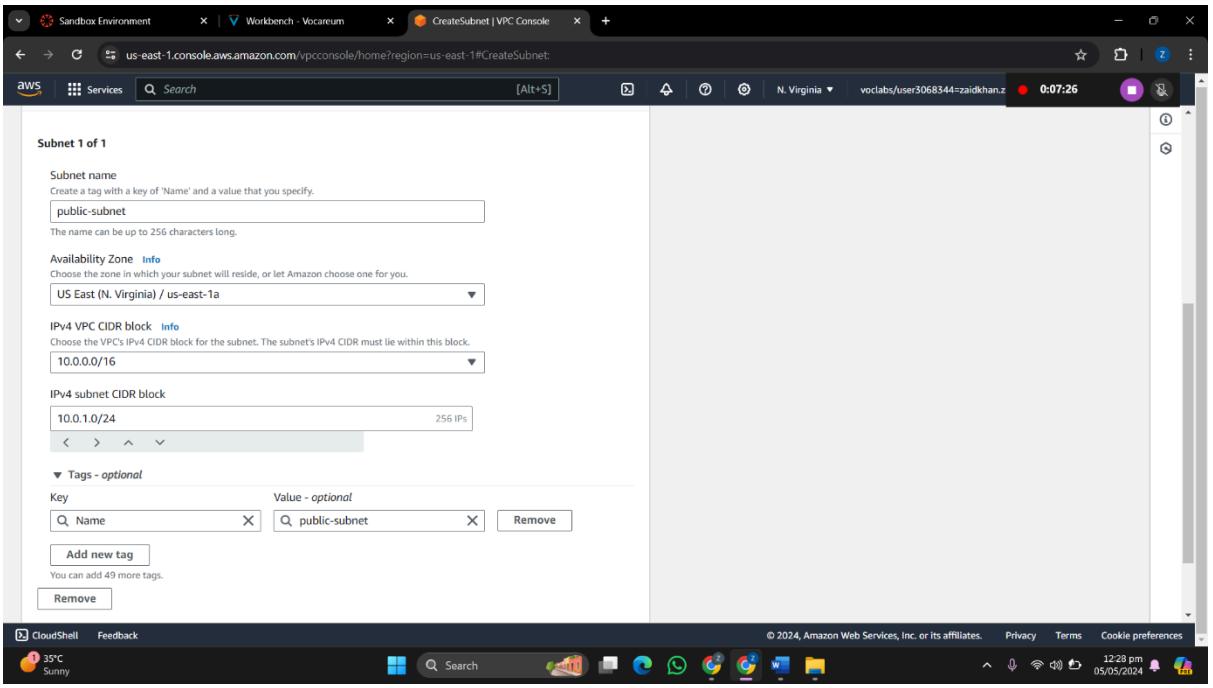


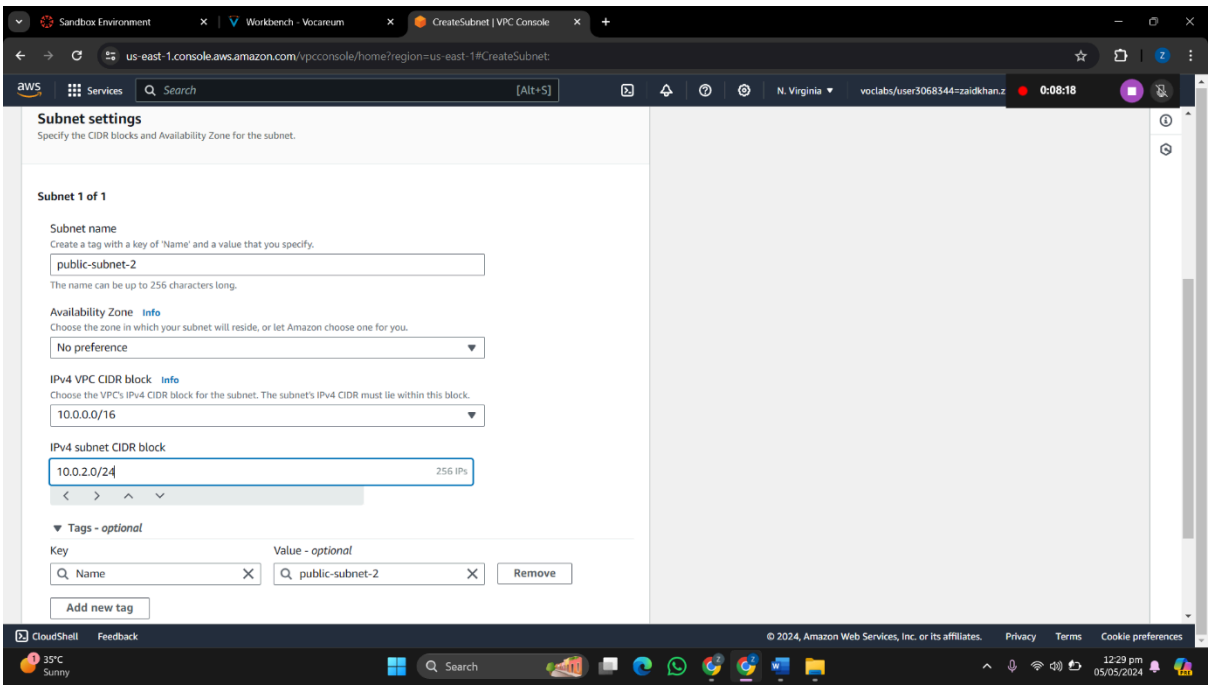
2.2.2 Subnets

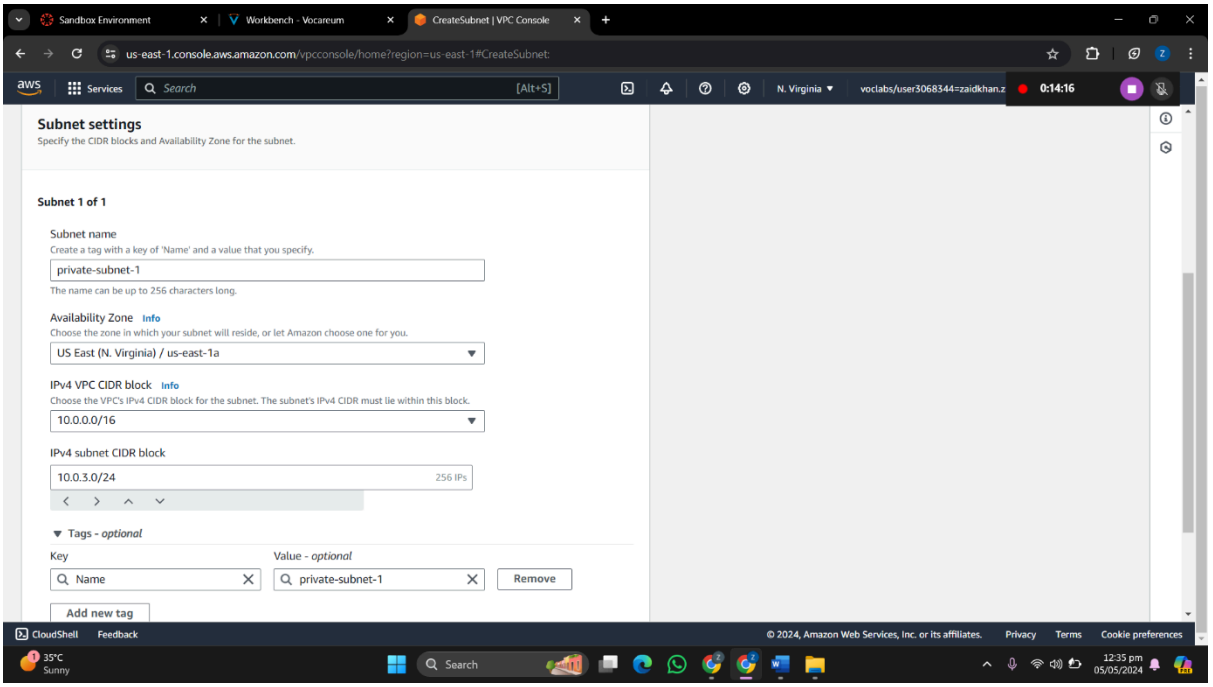
We created 2 public and 2 private subnets in Mutilple Availability Zones i.e US East 1a and US East 1b.

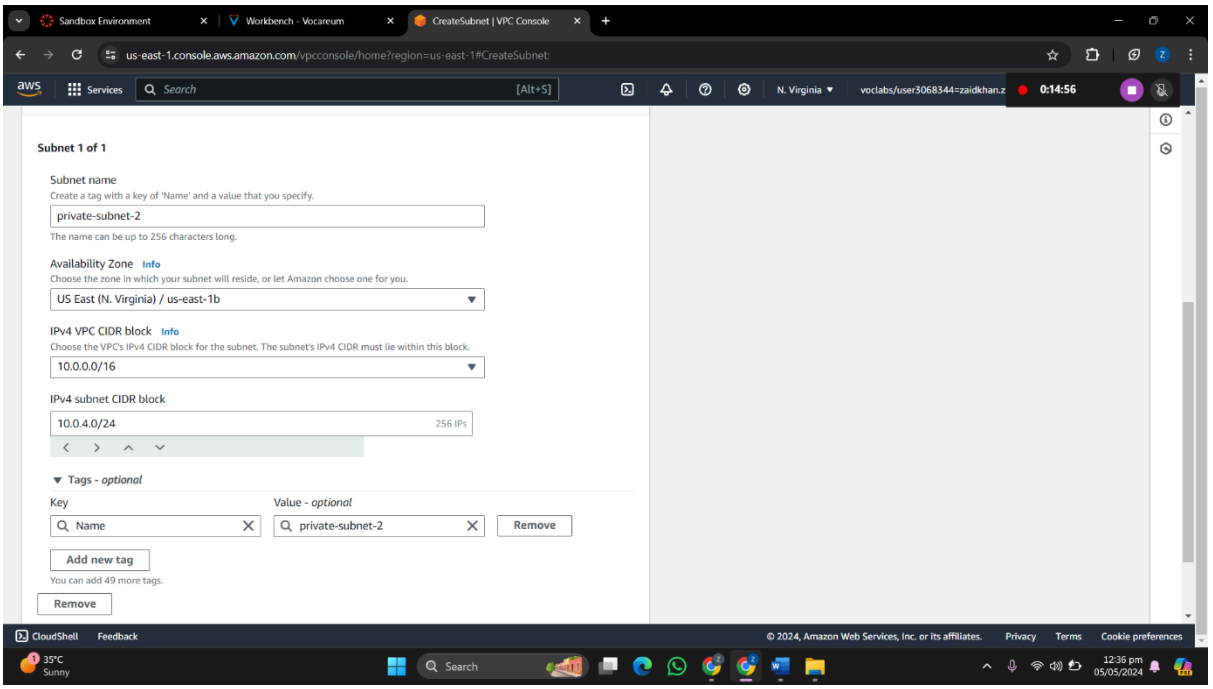
- CIDR Block Address : 10.0.0.0/16
- Pubic Subnet 1 CIDR Block Address : 10.0.1.0/24
- Pubic Subnet 2 CIDR Block Address : 10.0.2.0/24
- Private Subnet 1 CIDR Block Address : 10.0.3.0/24
- Private Subnet 2 CIDR Block Address : 10.0.4.0/24

Public Subnets were attached to an Internet Gateway and default IP's were assigned but this was not done in Private Subnets case.



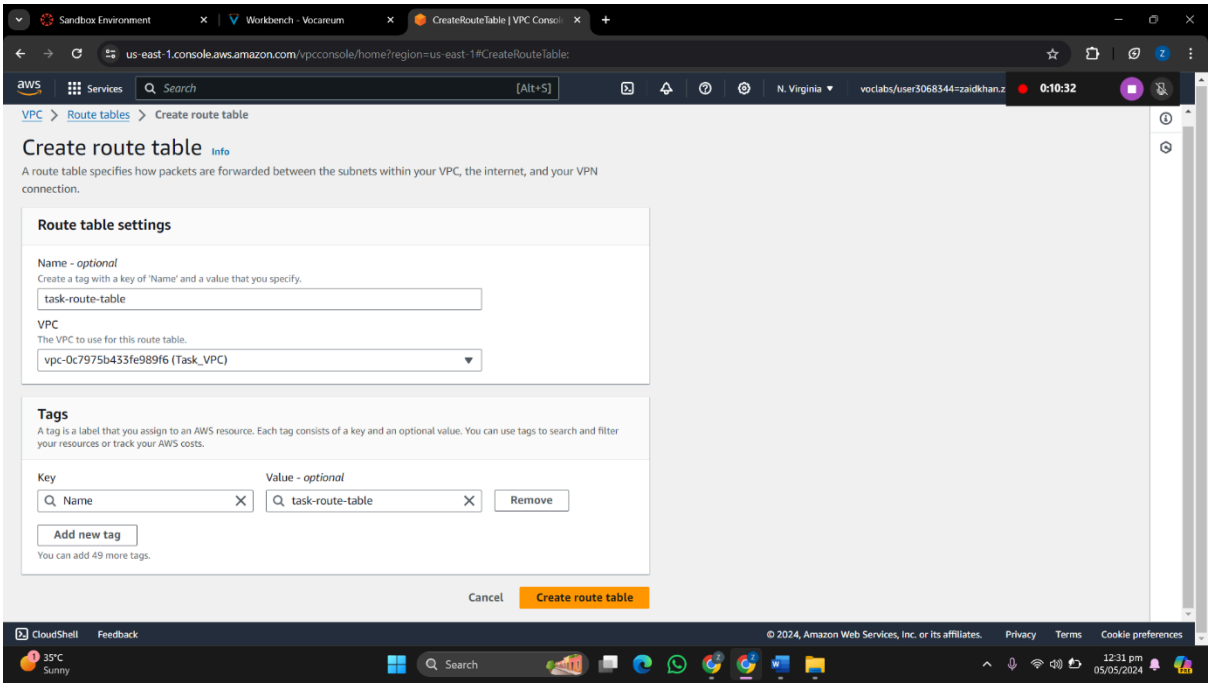


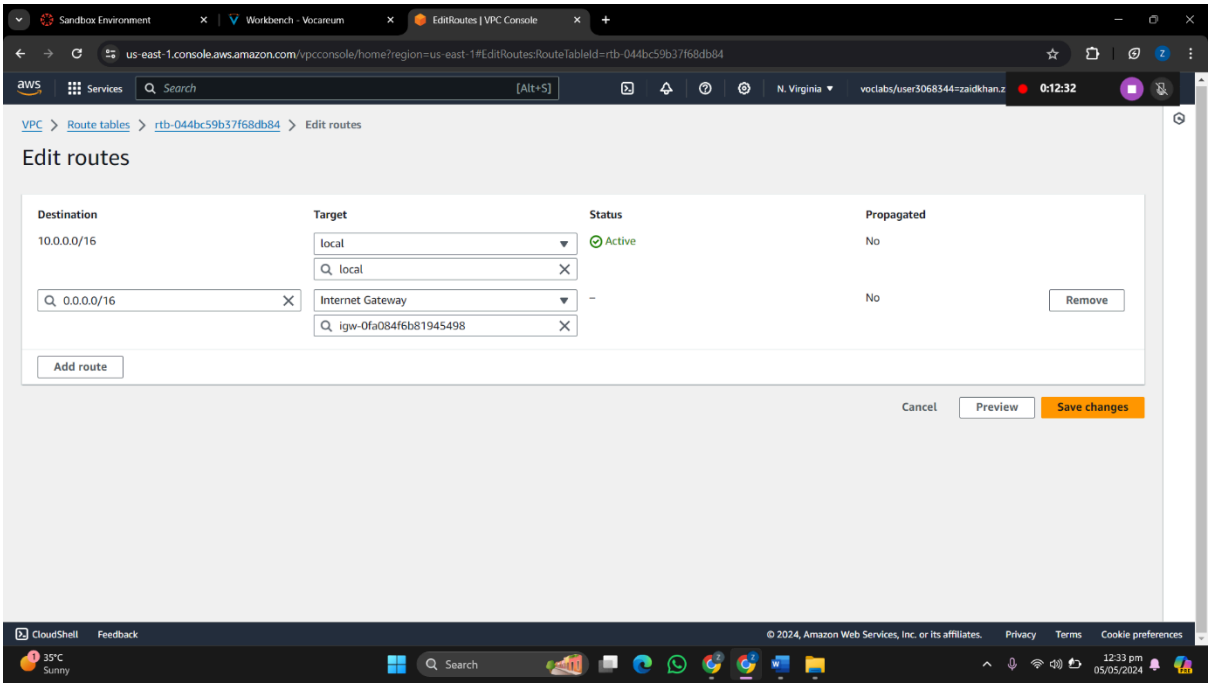


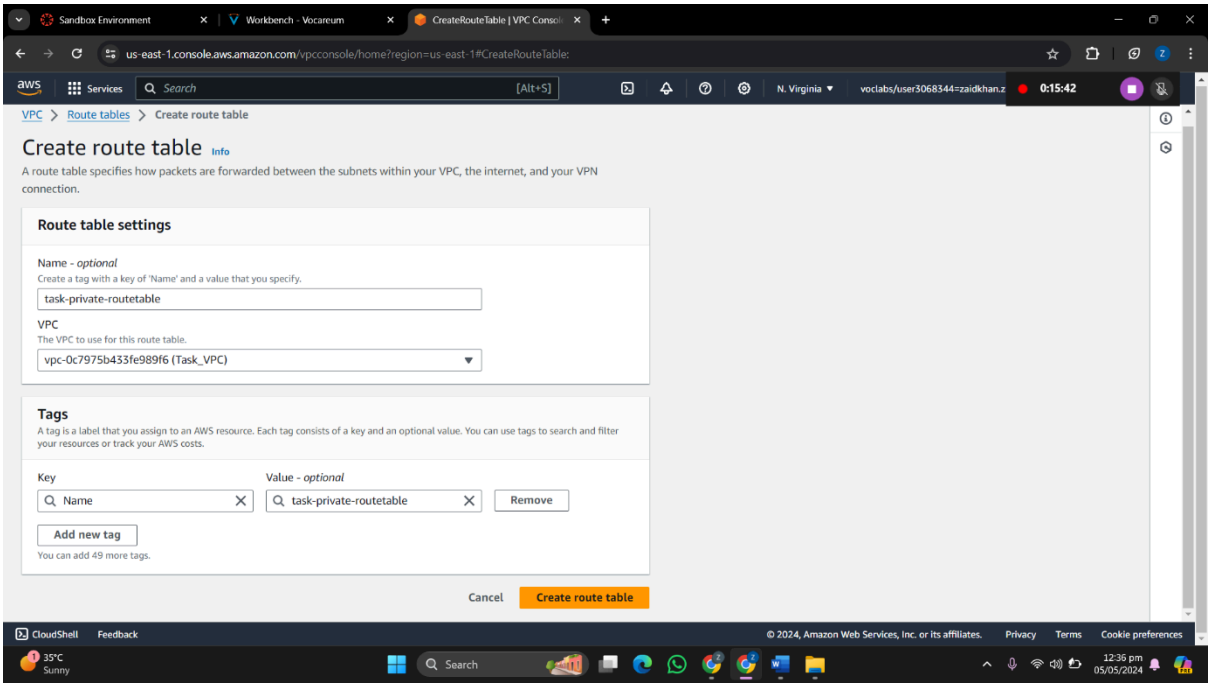


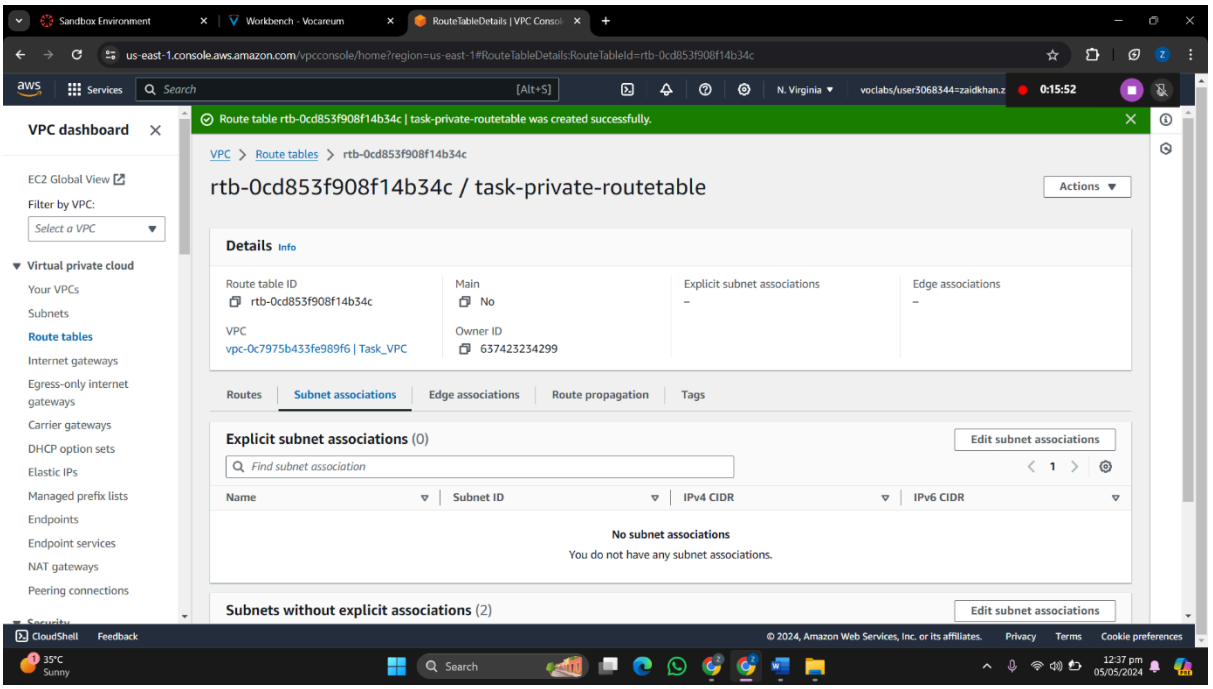
2.2.3 Route Tables

We created two separate route tables one for the public subnets and other for the private subnets. We edited the route table settings and for the Public one we attached alsoa designation as Internet Gateway with 0.0.0.0/16. but it was not done for the private route tables.

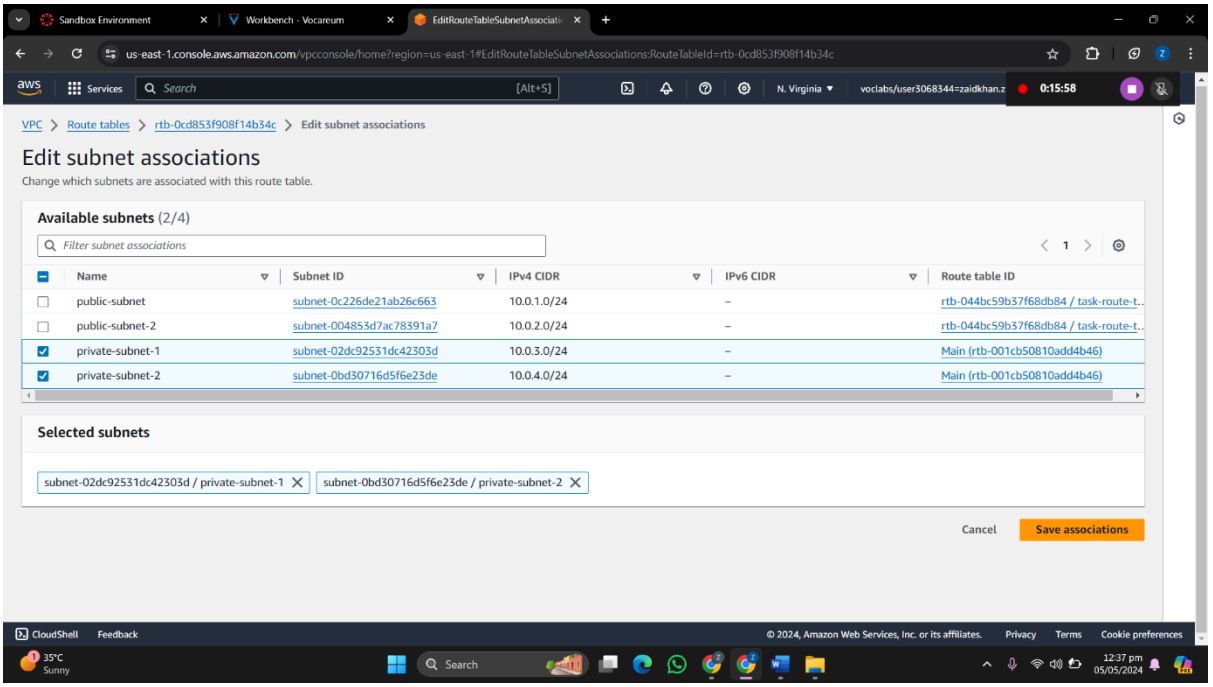

Sandbox Environment
Workbench - Vocareum
CreateRouteTable | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateRouteTable:
Services
Search
[Alt+S]
N. Virginia
voclabs/user3068344=zaidkhan.z
0:10:32
VPC > Route tables > Create route table
Create route table Info
A route table specifies how packets are forwarded between the subnets within your VPC, the internet, and your VPN connection.
Route table settings
Name - optional
Create a tag with a key of 'Name' and a value that you specify.
task-route-table
VPC
The VPC to use for this route table.
vpc-0c7975b433fe989f6 (Task_VPC)
Tags
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Key
Value - optional
Name
task-route-table
Remove
Add new tag
You can add 49 more tags.
Cancel
Create route table
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
35°C
Sunny
Search
12:31 pm
05/05/2024


Sandbox Environment
Workbench - Vocareum
EditRoutes | VPC Console
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#EditRoutes:RouteTableId=rtb-044bc59b37f68db84
Services
Search
[Alt+S]
N. Virginia
voclabs/user3068344=zaidkhan.z
0:12:32
VPC > Route tables > rtb-044bc59b37f68db84 > Edit routes
Edit routes
Destination
Target
Status
Propagated
10.0.0.0/16
local
Active
No
local
0.0.0.0/16
Internet Gateway
–
No
Remove
igw-0fa084f6b81945498
Add route
Cancel
Preview
Save changes
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
35°C
Sunny
Search
12:33 pm
05/05/2024



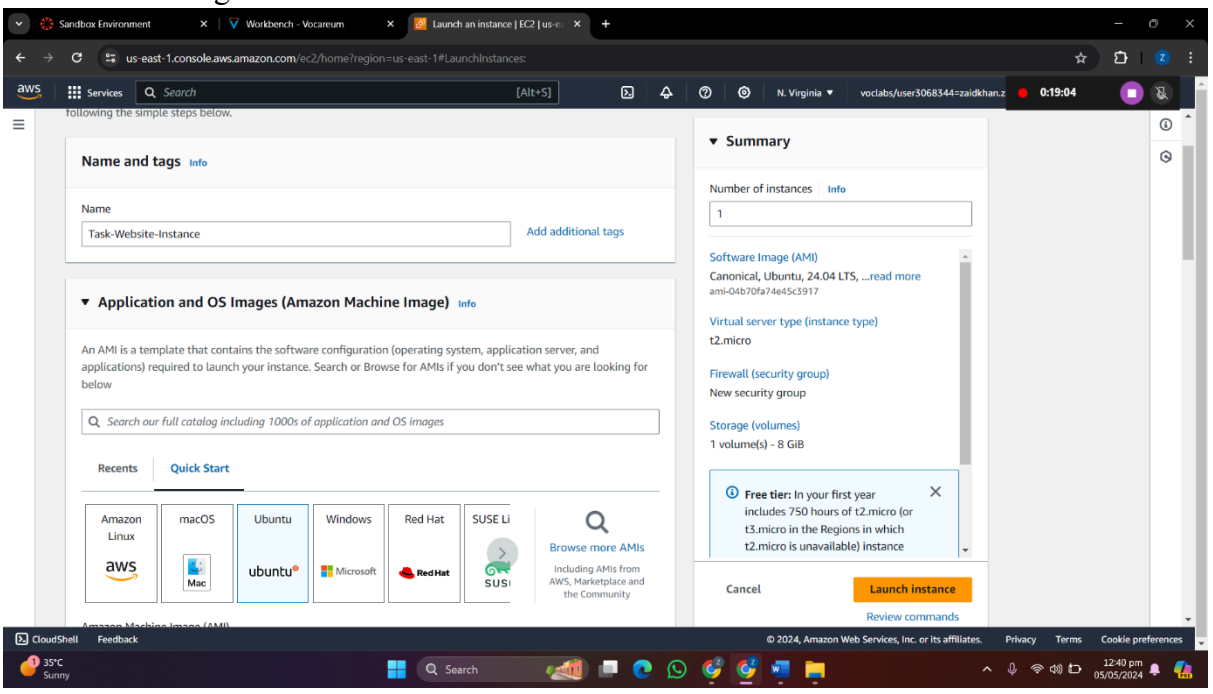


We decided to edit the subnet associations for the route table and we assigned the subnets to the respective route table which is depicted in the below image.



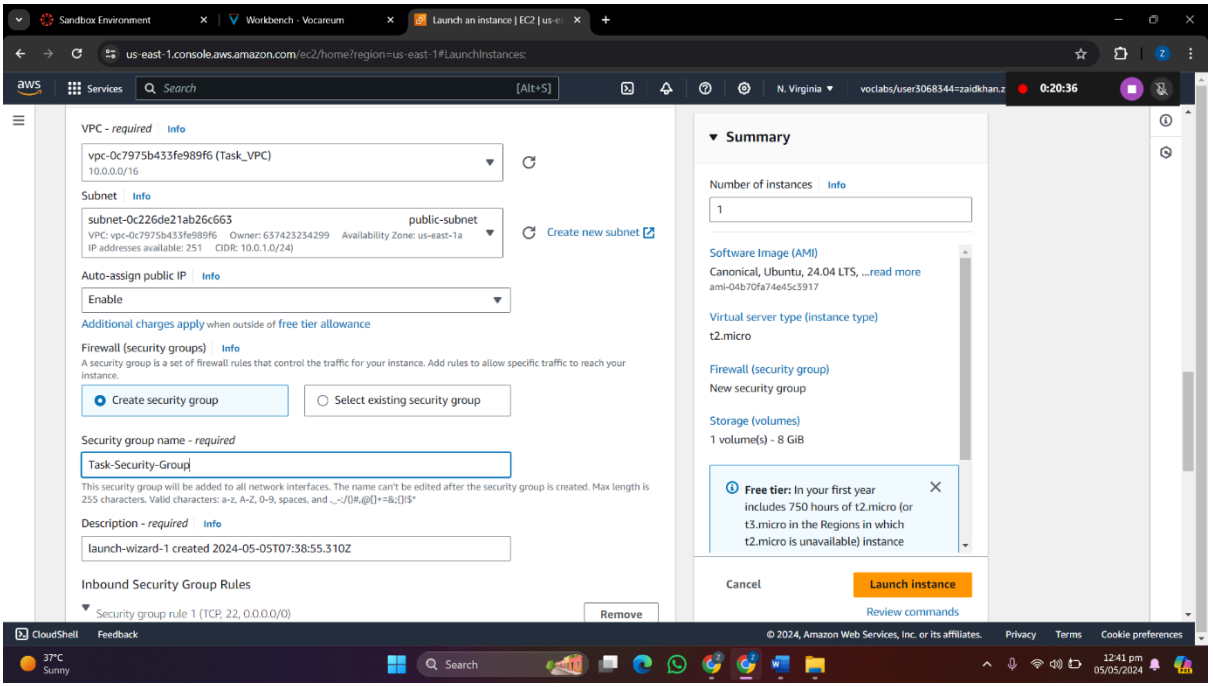
2.2.4 EC2 Instances

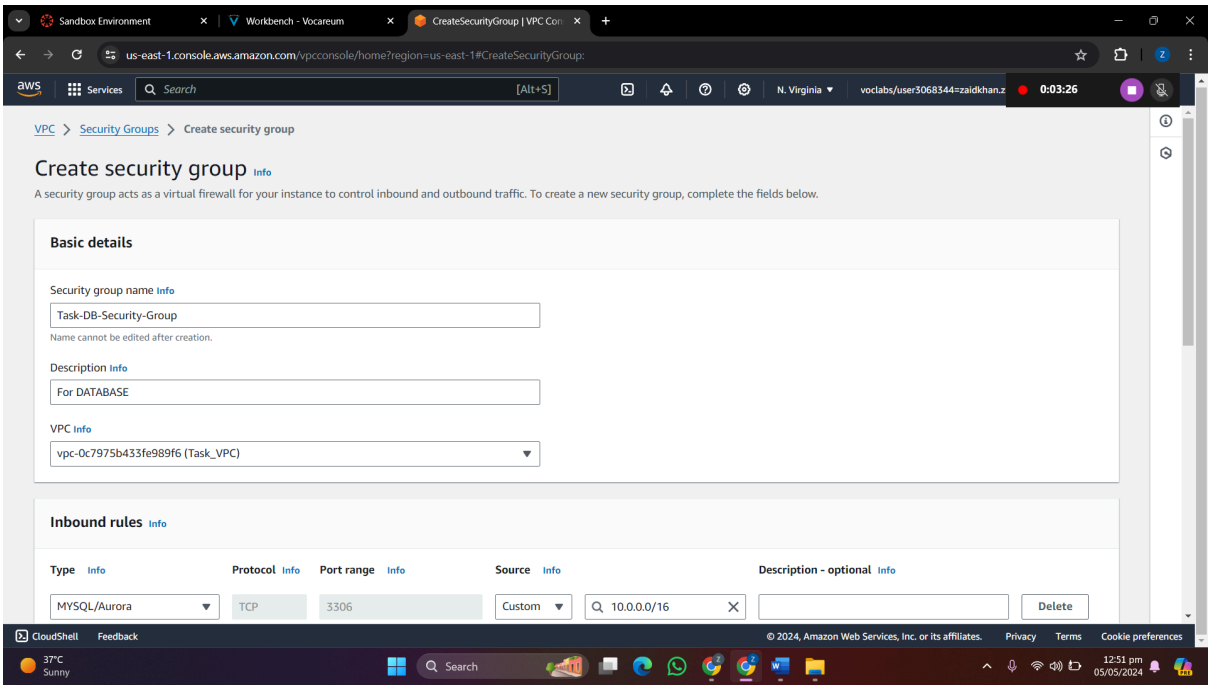
We navigated to EC2 and decided to create and EC2 instance for our project with Linux based Image. We selected **t2.micro** as Instance Type and also created security groups for the instance and assigned to it.

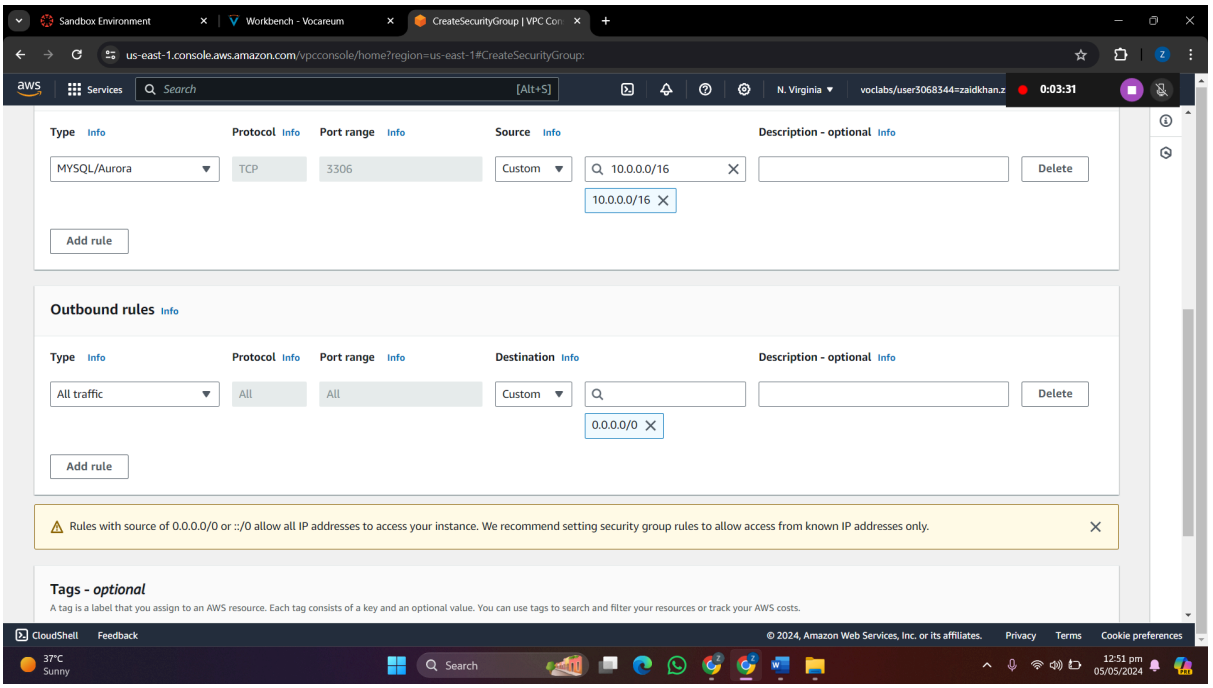


2.2.5 Security Groups

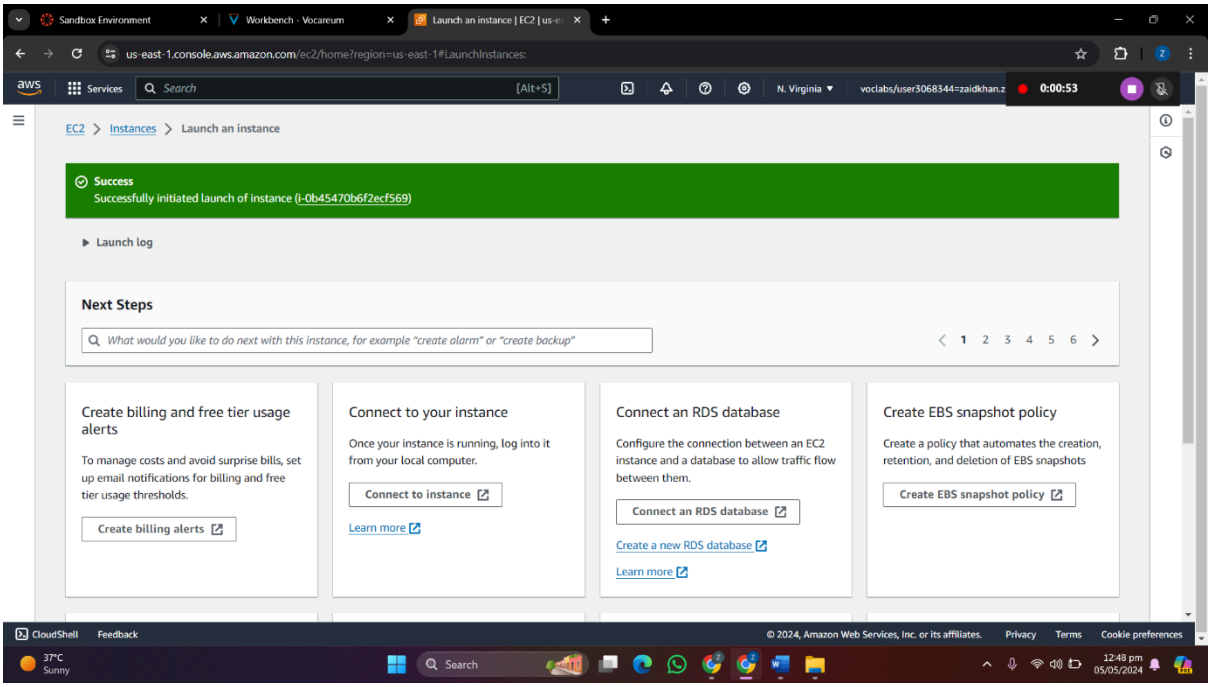
We created the security groups to limit the inbound and outbound traffic to the VPC. The rules were defined where we allowed allowed **HTTP from Anywhere** and for the other type we chose MySQL/Aurora with source as **10.0.0.0/16.**


Sandbox Environment
Workbench - Vocareum
Launch an instance | EC2 | us-e
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
Services
Search
[Alt+S]
N. Virginia
voclabs/user3068344=zaidkhan.z
0:20:36
VPC - required Info
vpc-0c7975b433fe989f6 (Task_VPC)
10.0.0.0/16
Subnet Info
subnet-0c226de21ab26c663 public-subnet
VPC: vpc-0c7975b433fe989f6 Owner: 637423234299 Availability Zone: us-east-1a
IP addresses available: 251 CIDR: 10.0.1.0/24)
Create new subnet
Auto-assign public IP Info
Enable
Additional charges apply when outside of free tier allowance
Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.
Create security group
Select existing security group
Security group name - required
Task-Security-Group
This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*
Description - required Info
launch-wizard-1 created 2024-05-05T07:38:55.310Z
Inbound Security Group Rules
Security group rule 1 (TCP, 22, 0.0.0.0/0)
Remove
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04 LTS, ...read more
ami-04b70fa74e45c3917
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance
Cancel
Launch instance
Review commands
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
37°C
Sunny
Search
12:41 pm
05/05/2024


Sandbox Environment
Workbench - Vocareum
CreateSecurityGroup | VPC Con
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#CreateSecurityGroup:
Services
Search
[Alt+S]
N. Virginia
voclabs/user3068344=zaidkhan.z
0:03:26
VPC > Security Groups > Create security group
Create security group Info
A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.
Basic details
Security group name Info
Task-DB-Security-Group
Name cannot be edited after creation.
Description Info
For DATABASE
VPC Info
vpc-0c7975b433fe989f6 (Task_VPC)
Inbound rules Info
Type Info
Protocol Info
Port range Info
Source Info
Description - optional Info
MYSQL/Aurora
TCP
3306
Custom
10.0.0.0/16
Delete
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
37°C
Sunny
Search
12:51 pm
05/05/2024

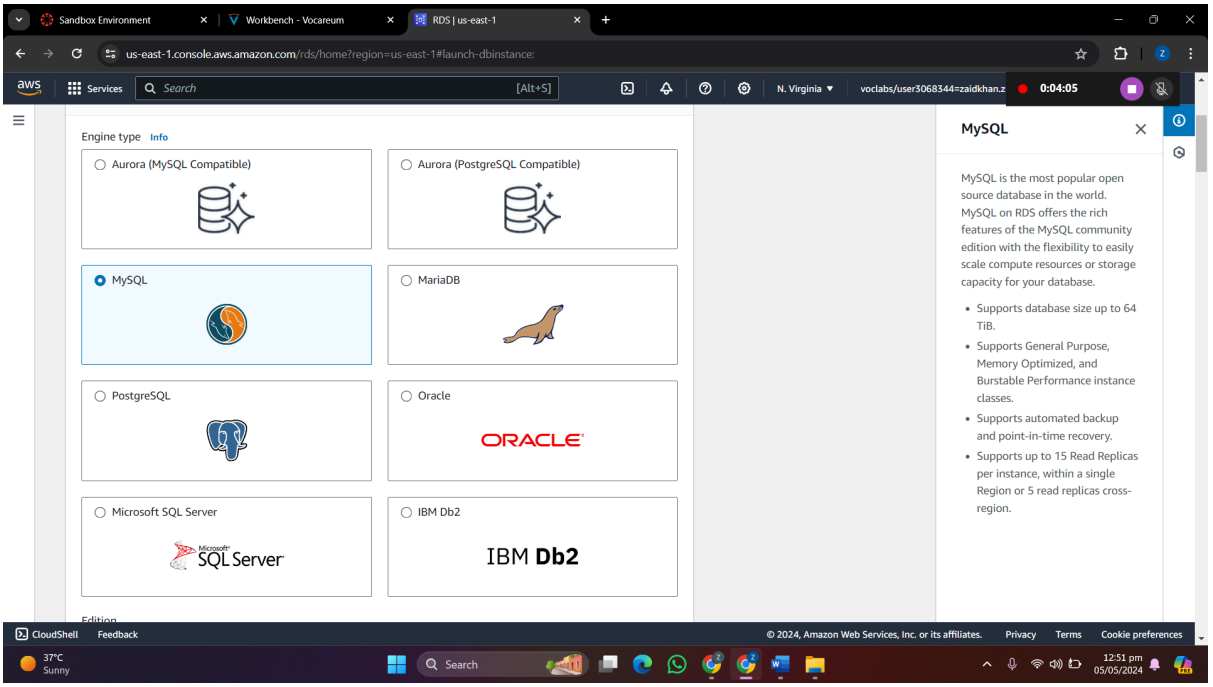


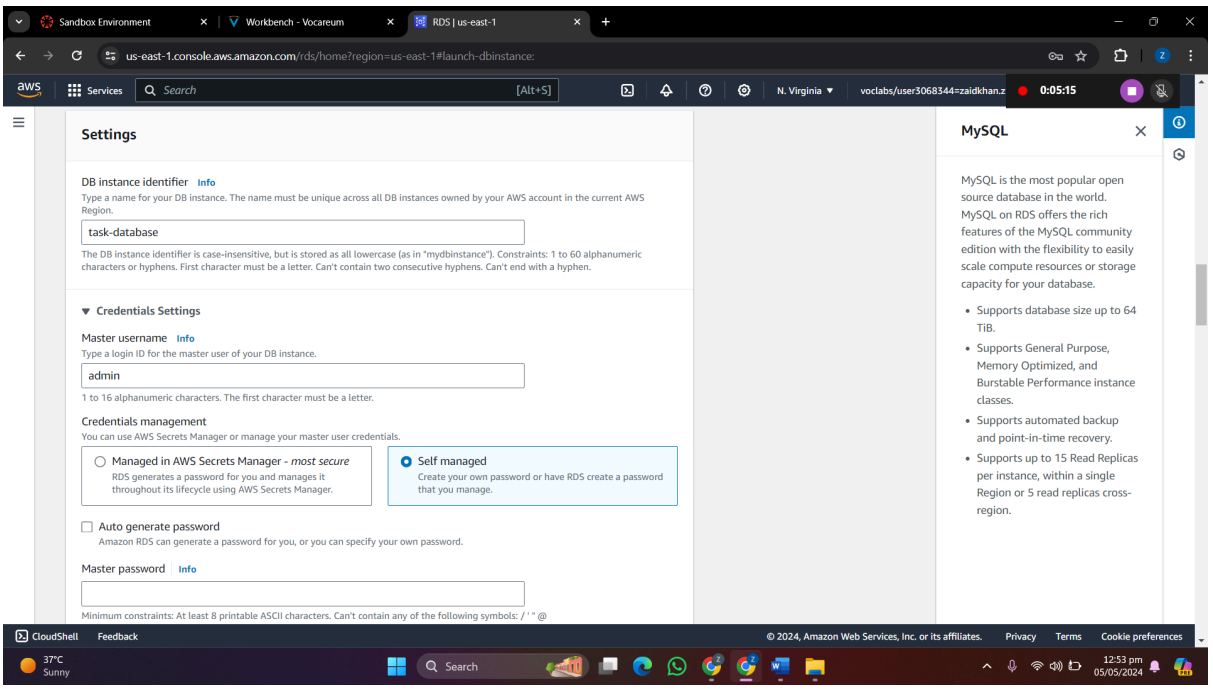
We were able to successfully create the Instances.

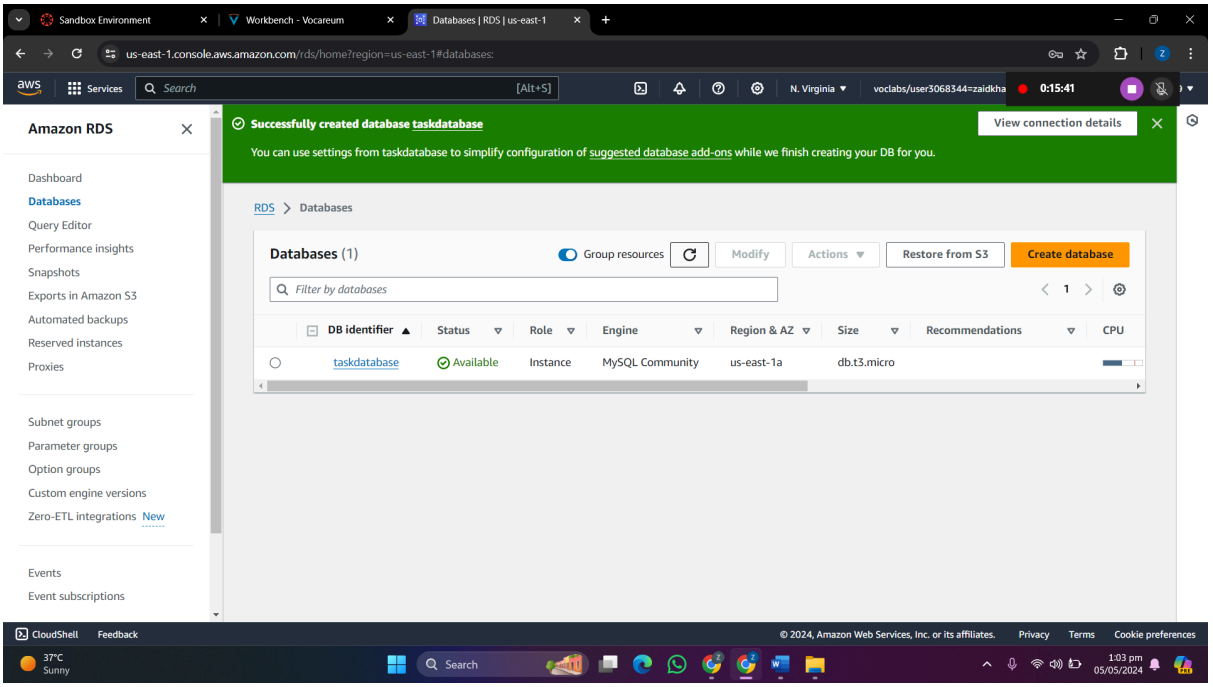


2.2.6 Creating RDS

The database used was My SQL with Free tier. Self Managed password and username were used. The database was linked to the Website EC2. The security groups created earlier were also linked with the database.


Sandbox Environment
Workbench - Vocareum
RDS | us-east-1
us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:
Services
Search
[Alt+S]
N. Virginia
voclabs/user3068344=zaidkhan.z
0:04:05
Engine type Info
Aurora (MySQL Compatible)
Aurora (PostgreSQL Compatible)
MySQL
MariaDB
PostgreSQL
Oracle
Microsoft SQL Server
IBM Db2
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
37°C
Sunny
Search
12:51 pm
05/05/2024

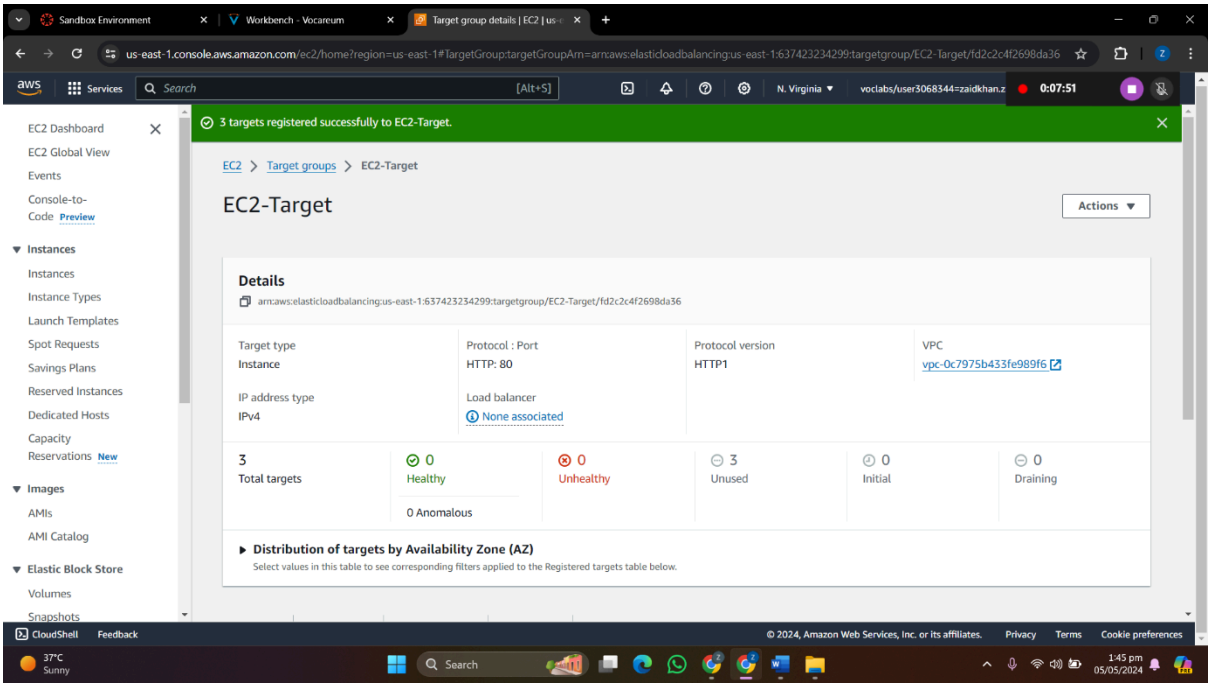

Sandbox Environment
Workbench - Vocareum
RDS | us-east-1
us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:
Services
Search
[Alt+S]
N. Virginia
voclabs/user3068344=zaidkhan.z
0:05:15
Settings
DB instance identifier Info
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.
task-database
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.
Credentials Settings
Master username Info
Type a login ID for the master user of your DB instance.
admin
1 to 16 alphanumeric characters. The first character must be a letter.
Credentials management
You can use AWS Secrets Manager or manage your master user credentials.
Managed in AWS Secrets Manager - most secure
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.
Self managed
Create your own password or have RDS create a password that you manage.
Auto generate password
Amazon RDS can generate a password for you, or you can specify your own password.
Master password Info
Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
37°C
Sunny
Search
12:53 pm
05/05/2024



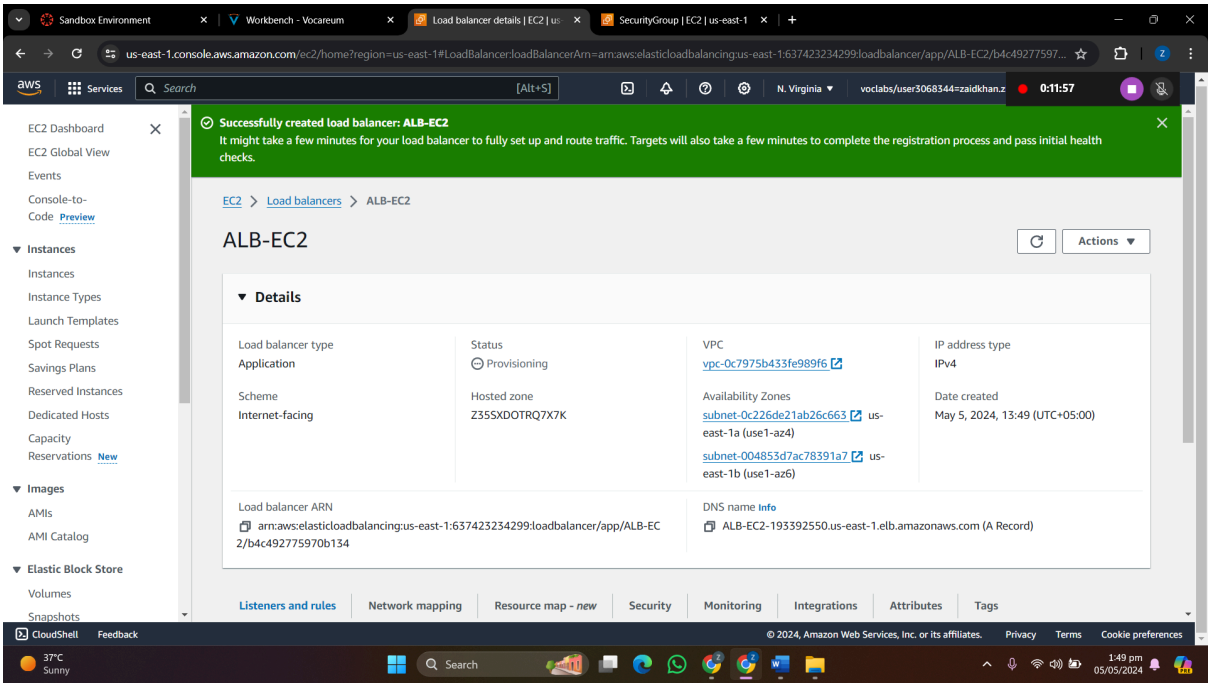
2.2.7 Load Balancing

The application load balancer was used which works at OSI layer 7 allowing manipulation and distribution of traffic at the application layer. We also created our own specific security group for the load balancer.

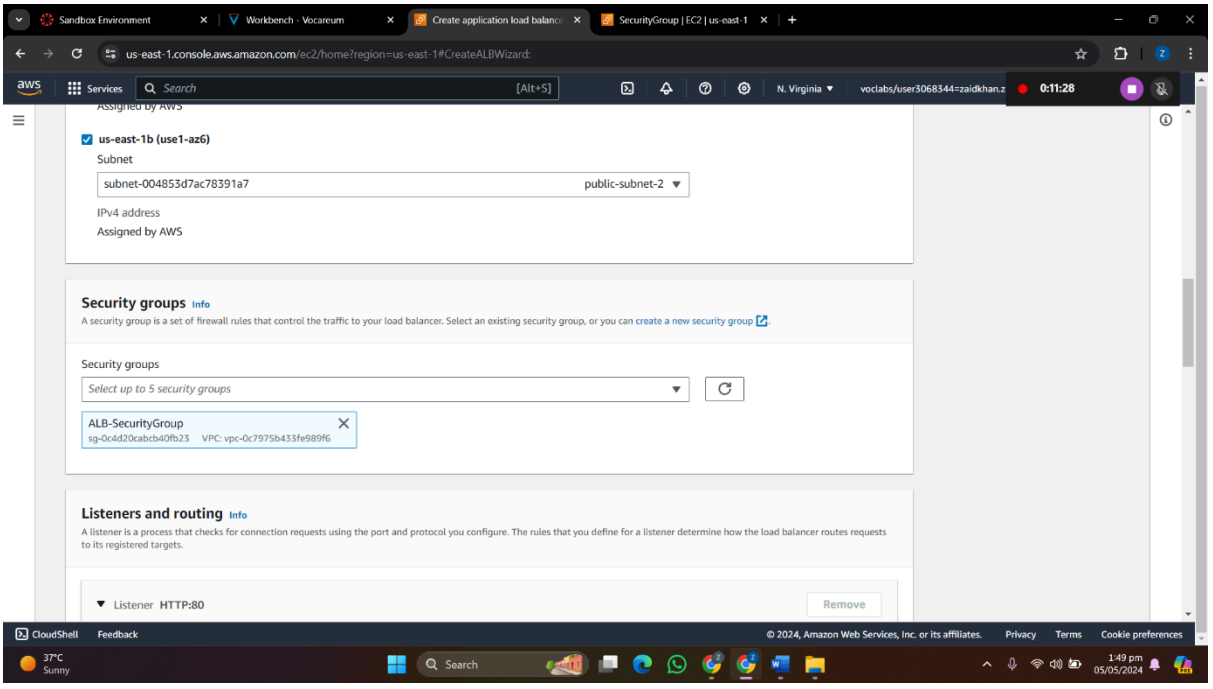
We created **Target Groups** and attached EC2 Instance .



We then attached Application Load Balancer and created a security group for it.

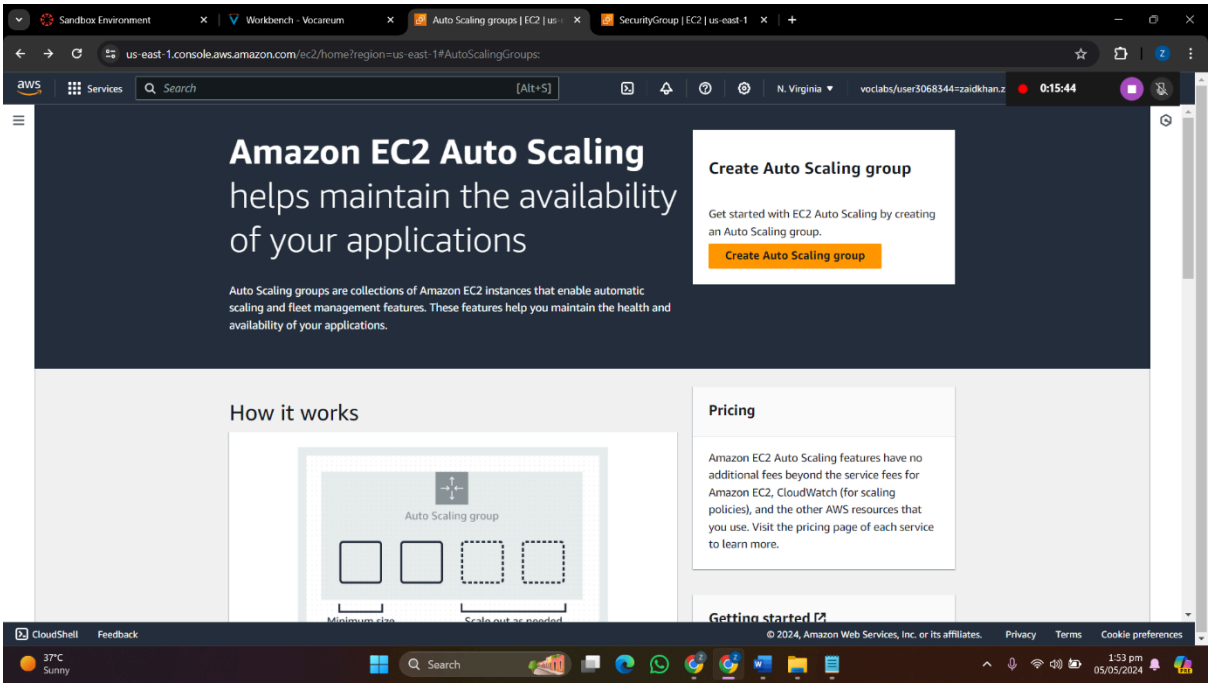


The Security group was attached to it.

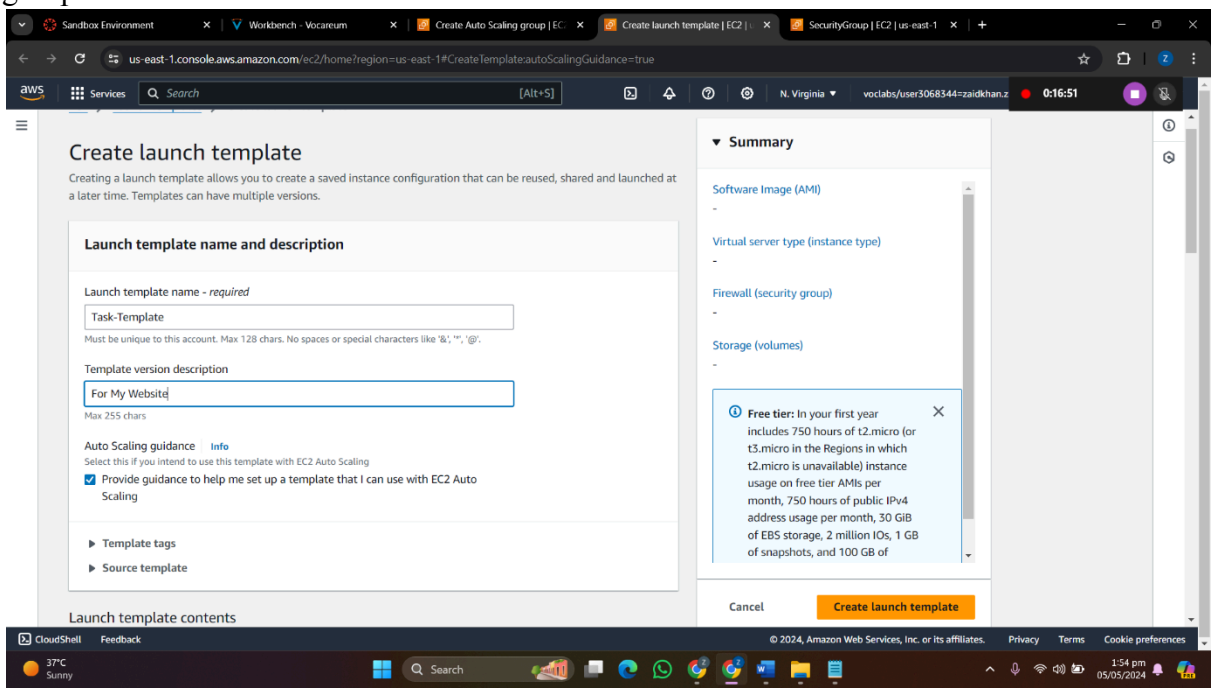


2.2.8 Amazon EC2 Auto Scaling

The auto scaling template was configured with t2.micro and vockey as a key pair. It was linked with the ALB Security group. Auto scaling automatically adjusts the resources based on usage along with minimizing the cost of operation.



Launch template was also created for it with the instance type of t2.micro and ALB Secuity group was attached to it.



We then configured the Instance Launch options and attached exisitng Load Balancer. and Created auto scaling group.

2.2.9 User Authentication

To enhance user authentication, AWS Identity and Access Management (IAM) is utilized extensively. IAM allows the creation and management of users, groups, roles, and policies to securely control access to AWS resources. In this scenario, IAM roles are assigned to EC2 instances to grant them specific permissions for accessing other AWS services or resources. By correctly configuring IAM policies and roles, users and groups can securely authenticate and access AWS resources based on their defined permissions, helping to enforce the principle of least privilege and strengthen overall security posture within the AWS environment.
Following are the roles and policies.

- **EC2 Instance Role :**
  This role is assigned to EC2 instances and grants them specific permissions to interact with other AWS services.
  Policy:
  Some user were assigned only EC2 Ready Only Access while the admin has the both read and write permissions.

The User groups are as below:

- **Admin Group:** This group has EC2 Full Access permissions and can also create instances.
  Policy: Provides full access to Amazon EC2 resources, allowing users to create, manage, and delete instances, volumes, and snapshot ans also grants full access to Amazon VPC resources, enabling users to create and manage virtual private clouds, subnets, and route tables**.**

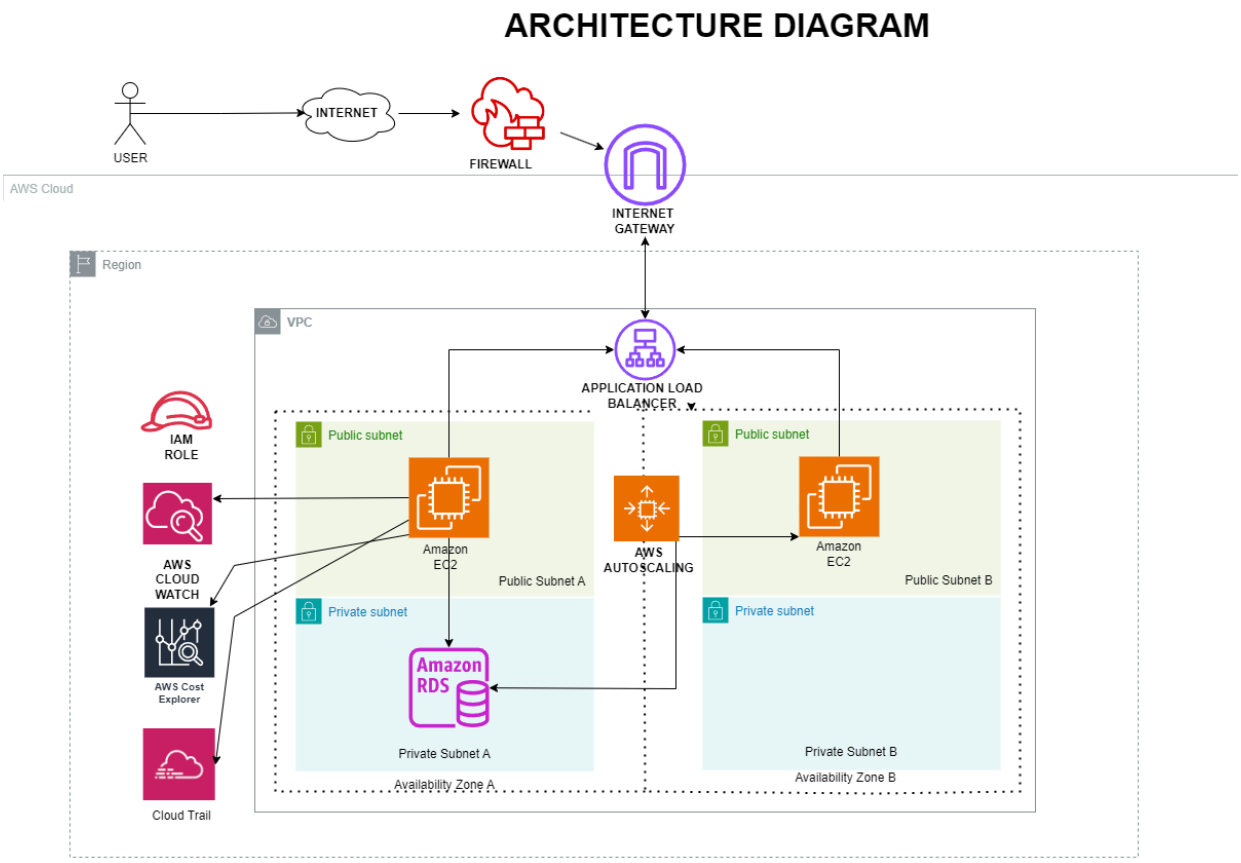
- **User Group:** This group has limited access and for EC2 has read only access this is mainly for less privileged users.
  Policy: Mainly Read Similar to EC2Admins, but with a broader access level to EC2 resources.

2.2.10 Auditing Services

We implemented **Cloud Watch, Cloud Trail** and **Cost Explorer** for monitoring to allow the administrator to get a close insight about the resource usage along with the pricing. This enables the administrator to take appropriate actions in case of overuse and allows them to set alerts also.

3. Task Architecture

After the successful deployment of our web application, we designed the architecture diagram. The diagram was designed on draw.io to enable the users to understand and get an overview of the design. AWS architecture diagrams are not mere eye candy; they are essential tools for architects, developers, and stakeholders. They provide clarity, validate designs1., document decisions, and guide the implementation of reliable, secure, and scalable systems.



4. Conclusion and Future Development

In conclusion, our team successfully designed and deployed a three-tier web application on the AWS cloud, adhering to the principles of a well-architected framework. By leveraging AWS services and following best practices, we ensured that our design achieved reliability, security, performance efficiency, cost optimization, and operational excellence.

Each team member played a crucial role in the planning and execution phases. Zaid's expertise in VPC setup and architectural visualization provided a solid foundation for our project, while Usman's contributions in implementing the RDS database, load balancers, and auto scaling groups significantly enhanced the performance and fault tolerance of our system.

Moving forward, we plan to continue monitoring and optimizing our application for further scalability, performance improvement, and cost optimization. We will be testing our application for load to ensure that it supports the community needs. Additionally, we will explore additional AWS services and features to enhance security, automate tasks, and streamline operations. Overall, this project has provided valuable hands-on experience in designing and deploying cloud-based applications on AWS, and we look forward to applying these learnings to future projects and endeavors.

References

[1] Ali, M. A., Ghugal, C., Bawane, N., Parate, S., & Wankhede, P. DEPLOYING 4 TIER MAILING WEB APPLICATION ON AWS.

[2] Kingsley, M. S. (2023). Amazon Web Services (AWS) Lab. In *Cloud Technologies and Services: Theoretical Concepts and Practical Applications* (pp. 159-247). Cham: Springer International Publishing.

[3] Anwar, N. (2018). Architecting Scalable Web Application with Scalable Cloud Platform.

[4] Wadia, Y., Udell, R., Chan, L., & Gupta, U. (2019). *Implementing AWS: Design, Build, and Manage your Infrastructure: Leverage AWS features to build highly secure, fault-tolerant, and scalable cloud environments*. Packt Publishing Ltd.