AWS LAB Introduction

Sunday, 19 October 2025 10:01 am

The purpose of this project is to have hands on practice with AWS for better understanding and exposure to Cloud and AWS infrastructure and its inner workings.

In this LAB, the use case will be adopting AWS cloud services for a company's ecommerce website to allow for scalability as the business grows. Low latency and accessibility are also considered here in the choosing of AWS to host our webapp.

Below is a diagram of the cloud infrastructure

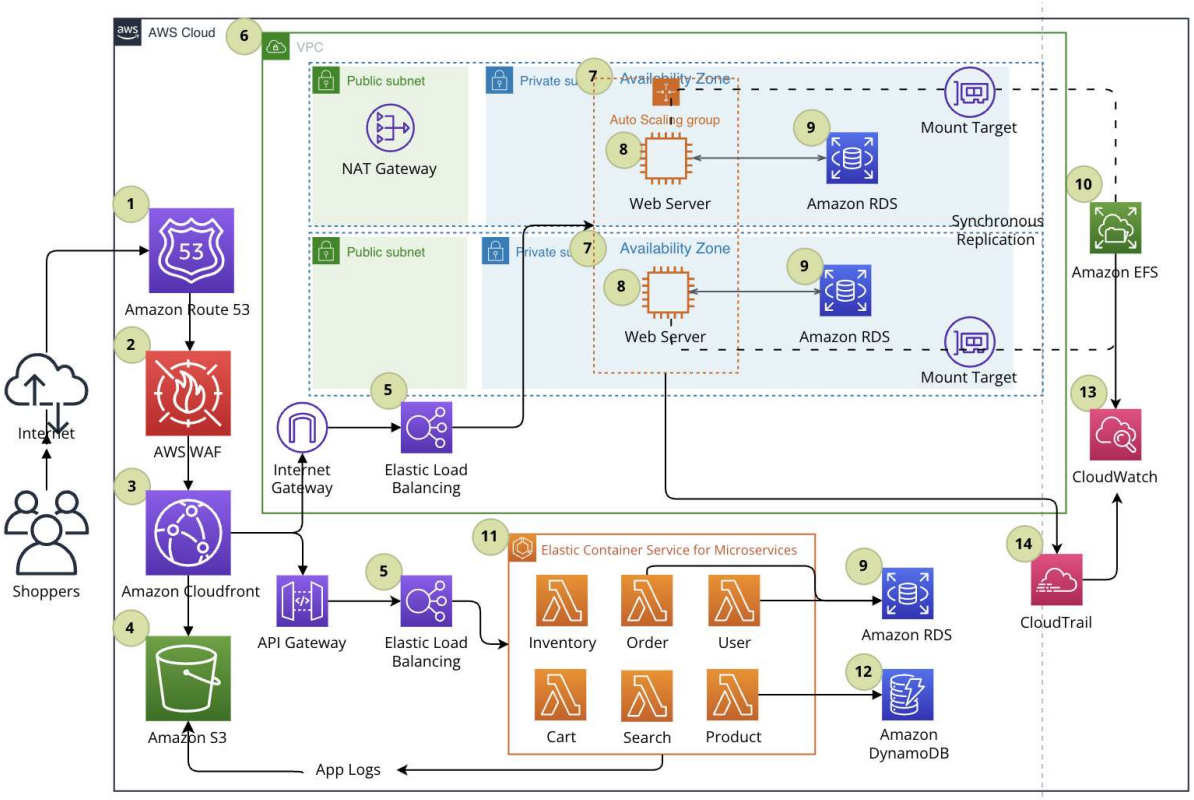


Figure 1: Cloud Infrastructure Diagram

1. Amazon Route 53: Routes shoppers DNS requests resolving domain name service (DNS) and provides global traffic management capabilities
2. AWS WAF: Web application firewall that protects the e-commerce website against common web exploits.
3. Amazon CloudFront: Provides a highly secure and programmable content delivery network (CDN) with edge locations around the globe. It can deliver dynamic content with low latency and high transfer speeds from locations close to the customer.
4. Amazon S3: Stores all static catalogue content, such as product images and videos, log files of the Microservices (#6) and clickstream information from Amazon CloudFront (# 3).
5. Elastic Load Balancing (ELB): Distributes network traffic to improve the scalability and availability of web servers across multiple AZs.
6. Single AWS Region and single Amazon Virtual Private Cloud (Amazon VPC): Provide

similar architecture as an on-premise data centre setup.

7. Multiple Availability Zones (AZs): Provide resilience and high availability for the production workload.
8. Frontend application Web Server: The frontend application web server for the e-commerce website is deployed by AWS EC2 Auto Scaling, which automatically handles the details of capacity provisioning, load balancing, auto scaling, and application health monitoring. Elastic IP will be sitting on top of the EC2 instance which allows for IP failover. If the EC2 instance needs to be reconfigured, the Elastic IP and Route 53 will work together to ensure that the service is still available when the web server comes back up.
9. Amazon RDS: To provide high availability, the user and orders databases are hosted redundantly on a multi-AZ (multi Availability Zone) deployment of Amazon Relational Database Service (Amazon RDS) within private subnets that are isolated from the public Internet.
10. Amazon Elastic File System: Shared file storage mounted to instances in the Auto Scaling group.
11. Microservices: Microservices hosted in Amazon Elastic Container Service that represent domain constructs such as products, carts, orders, inventory and users as well as search services.
12. Amazon DynamoDB: Amazon DynamoDB is a fully-managed, high performance, NoSQL database service that is easy to set up, operate, and scale. It is used both as a session store for persistent session data, such as the shopping cart, and as the product database. Because DynamoDB does not have a schema, we have a great deal of flexibility in adding new product categories and attributes to the catalogue.
13. Amazon CloudWatch: Used for application logging, monitoring, and alarms for dynamic scaling. When a performance threshold is breached, a CloudWatch alarm triggers an automatic scaling event that either scales out or scales in EC2 instances in the Auto Scaling group.
14. Amazon CloudTrail: Used to track user activity and API usage i.e. logging the date, time, and identity of users accessing directory data. Logs from CloudTrail to be sent to CloudWatch for dynamic monitoring.

For demonstration purposes, I will be showing the setup and security hardening for cloud services 4, 6, 7, 8, 9, 13 and 14 labelled above.

VPC Set-Up

Sunday, 19 October 2025 10:06 am

The first step to cloud migration is setting up the Virtual Private Cloud (VPC) for the company. For demonstration, I will be showcasing the migration of our web server onto the cloud, with screenshots, as well as a link to the screen recording videos in 4.13.

Region : N. Virginia (us-east-1)

VPC Name: MRSH

IPv4 CIDR block: 192.168.254.0/24

The VPC settings are as follows with the creation of 2 public and 2 private subnets:

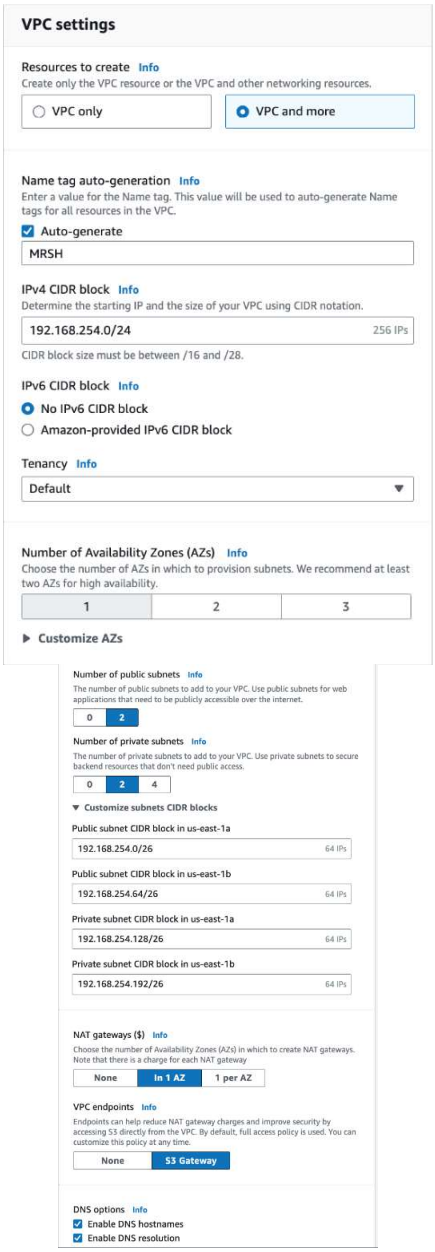


Figure 2 & 3: VPC Creation Settings

Following the creation of the above subnets, we can check to see the route tables have been configured accordingly with Figures 2 & 3 showing the public route table MRSH-rtb-public containing a route to the target starting with “igw”, which stands for Internet Gateway; thereby granting our public subnets direct

access to the internet.

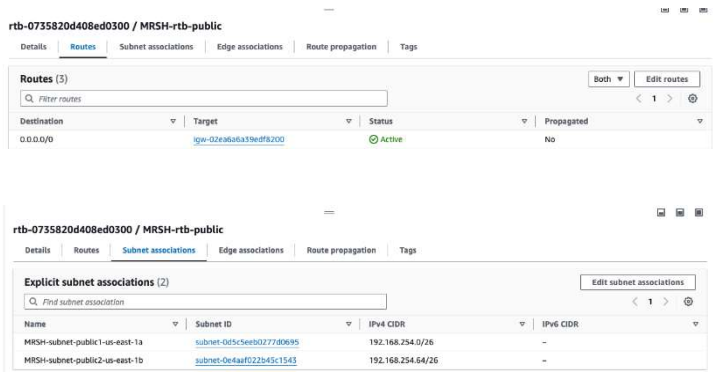
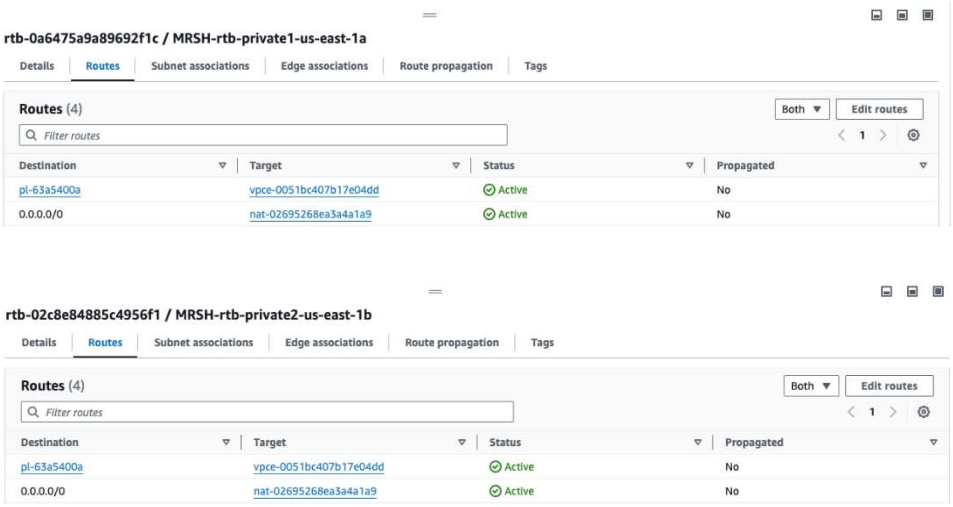


Figure 4 & 5: Public route table

Figures 6 & 7 show that both route tables MRSH-rtb-private1-us-east-1a and MRSH-rtb-private1-us-east-1b contain a route 0.0.0.0/0 connected to the NAT gateway, which grants our private subnets internet connection whilst keeping resources private.



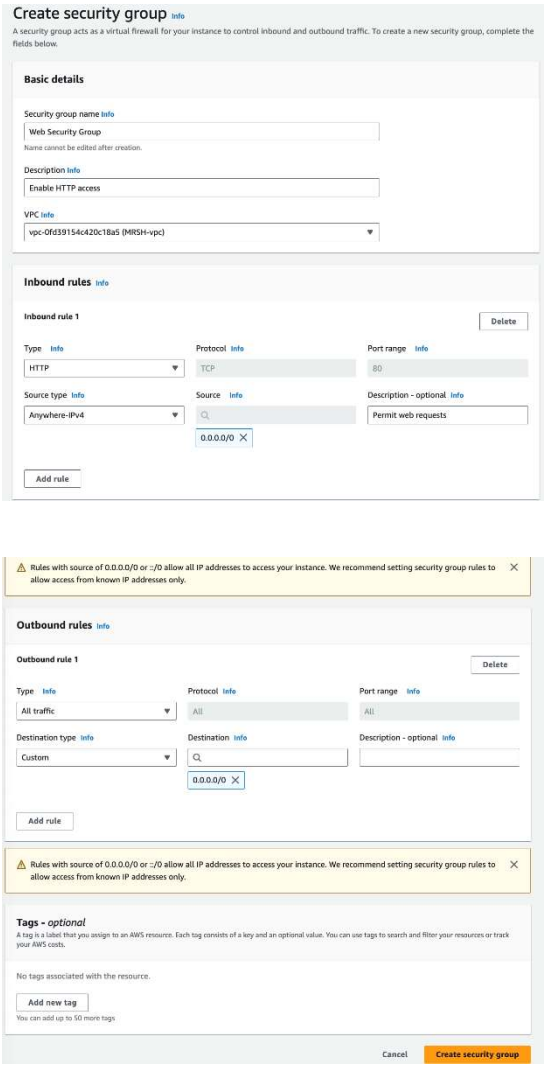
Figures 6 & 7: Public route table

EC2 Instance - Web Server

Sunday, 19 October 2025 10:12 am

Web Server Security Group

Before launching the web server, I prepare a security group which acts as a virtual firewall for our EC2 Web Server instance, consisting of rules to monitor and control inbound and outbound traffic. In this case, as the VPC will contain the web server, it will need a security group to allow inbound and outbound HTTP and HTTPS access:



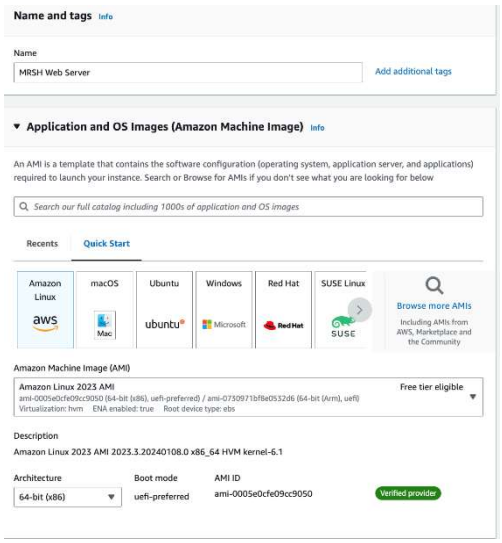
Figures 8 & 9: Web Server Security Group

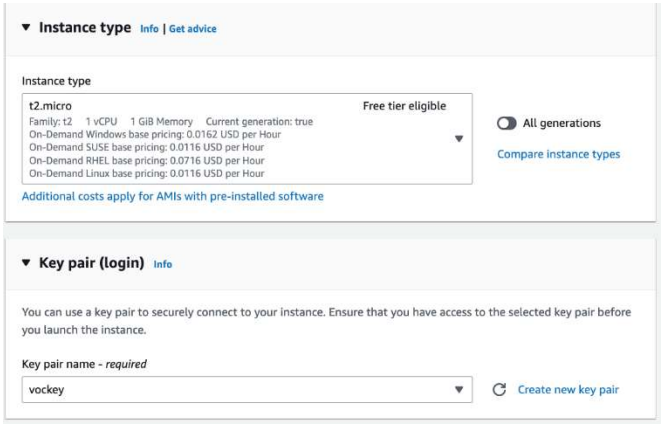
Web Server Set-Up

In the next step, I launch an EC2 instance configured as a web server.

Name: MRSH Web Server

AMI: Amazon Linux 2023 AMI



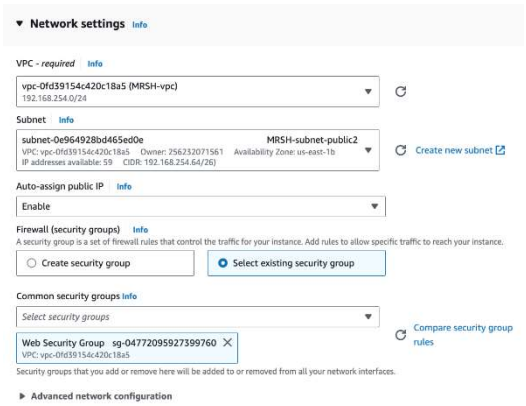


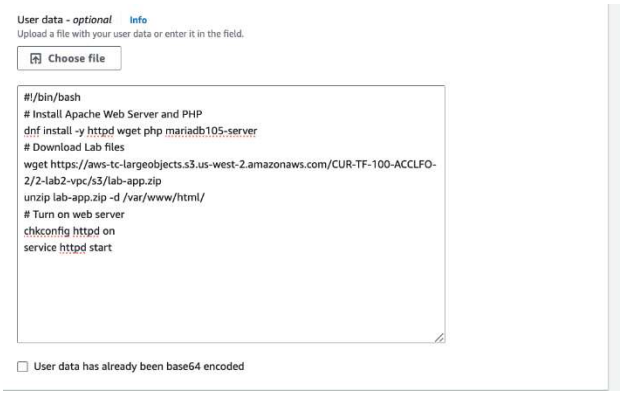
Figures 10 & 11: Web Server Settings 1

In Figures 10 & 11, we connect our web server to our Web Security Group created in the VPC Set-Up section. And as shown in Figure 13, we insert the following script under the "User data" section:

```
#!/bin/bash
# Install Apache Web Server and PHP
dnf install -y httpd wget php mariadb105-server
# Download Lab files
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-100-ACCLFO-2/2-lab2-vpc/s3/lab-app.zip
unzip lab-app.zip -d /var/www/html/
# Turn on web server
chkconfig httpd on
service httpd start
```

The script above will run root user permissions on the instance's guest OS automatically upon first launch. It will also install a web server (Apache), PHP libraries and PHP web application, and a database.





Figures 12 & 13: Web Server Settings 2

RDS Set-Up (DB)

Sunday, 19 October 2025 10:15 am

Amazon Relational Database Service (RDS)

DB Instance Set-Up

After launching our web server, we go onto the Amazon RDS console to create a database (DB) instance with the following settings shown above. The following table shows the comparison of Engine Types in Amazon RDS:

Engine Type	Use Cases	Strengths	Weaknesses	Typical Costs
Amazon Aurora MySQL/PostgreSQL	High-performance, scalable OLTP workloads	• High throughput and low latency • Automatic scaling • Read replicas for high availability • Cost-effective	• Not suitable for complex queries • Limited feature set compared to standard MySQL/PostgreSQL	Generally lower than standard MySQL/PostgreSQL on RDS due to automatic scaling capabilities
Standard MySQL/PostgreSQL	General-purpose workloads, web applications	• Familiar engine for many developers • Wide range of features and tools • Flexible configuration options	• May not be as scalable or performant as Aurora • Can be expensive for high-traffic applications	More expensive than Aurora, pricing varies depending on instance type and configuration
MariaDB	Open-source alternative to MySQL	• High compatibility with MySQL • Lower cost than Amazon MySQL	• Limited feature set compared to Amazon MySQL • Fewer tools and resources available	Generally slightly cheaper than Amazon MySQL on RDS
Oracle Database Enterprise Edition	Mission-critical enterprise applications	• Robust security and compliance features • High availability and disaster recovery options • Powerful features for complex workloads	• Higher cost than other options • Can be complex to manage	Most expensive option on RDS, pricing varies depending on edition and instance type
Microsoft SQL Server	Windows-based applications,	• Familiar for Windows	• Can be expensive for high-traffic	Typically more expensive than Amazon

	enterprise workloads	developers • High compatibility with other Microsoft products • Strong performance for certain workloads	applications • Licensing can be complex	MySQL/PostgreSQL, but cheaper than Oracle
Amazon DynamoDB	NoSQL database for high-throughput, low-latency applications	• Extremely scalable and performant • Pay-per-use billing model • Flexible data model	• Not suitable for relational data • More complex to query than a relational database	Generally lower cost than RDS options for high-throughput workloads

Table 1: Engine Types in Amazon RDS

I selected MariaDB as our Engine type as it is a modified version of MySQL and has more features, higher scalability and query speed compared to MySQL, while at the same time still supporting/retaining many features of MySQL (e.g. structure, naming conventions, connectors, connections, ports).

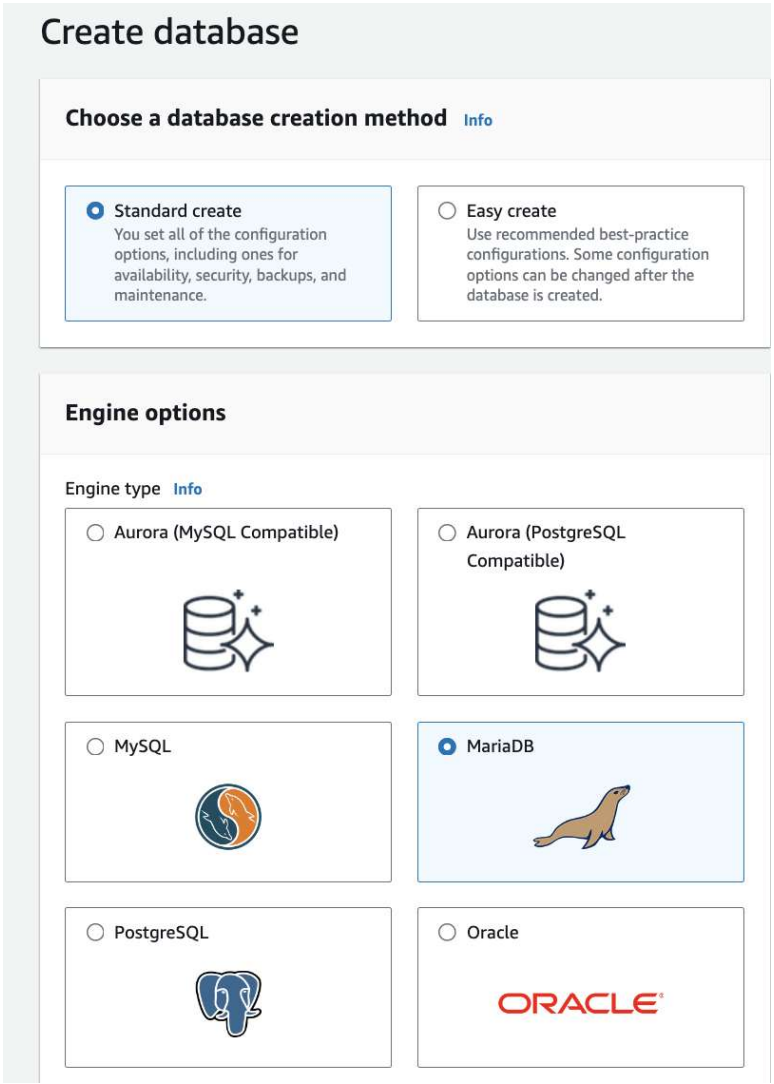


Figure 14: DB Instance Setting 1-1

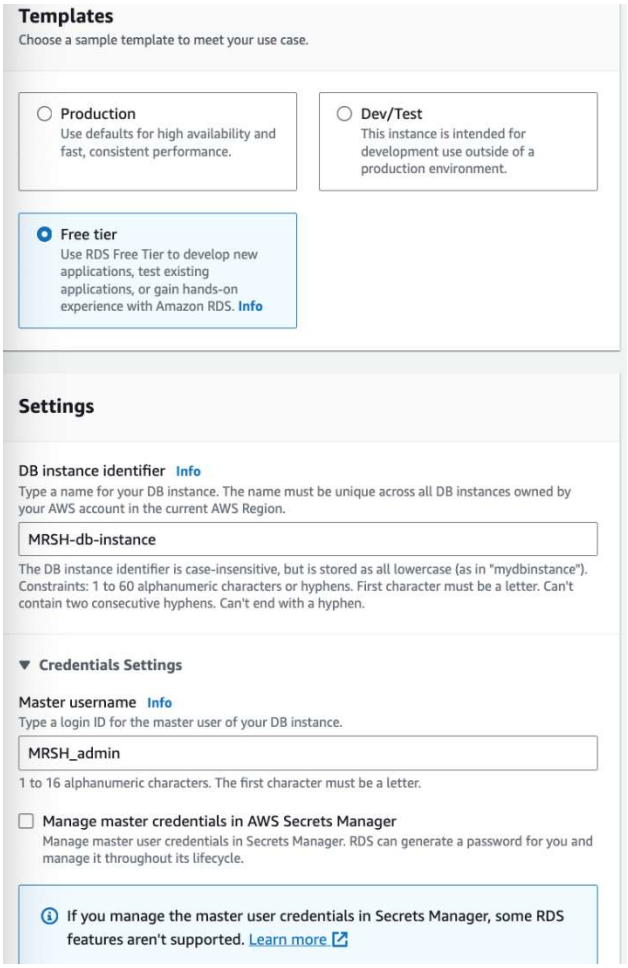


Figure 15: DB Instance Setting 1-1

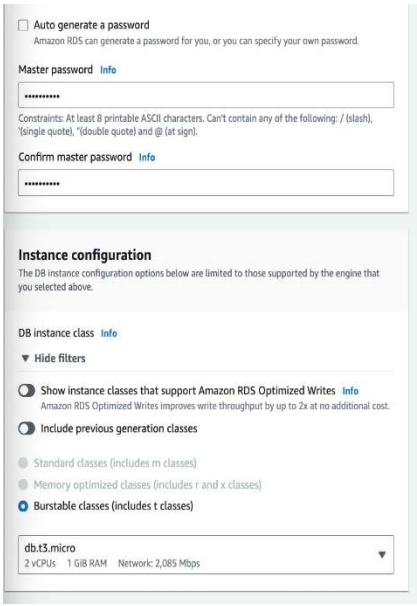
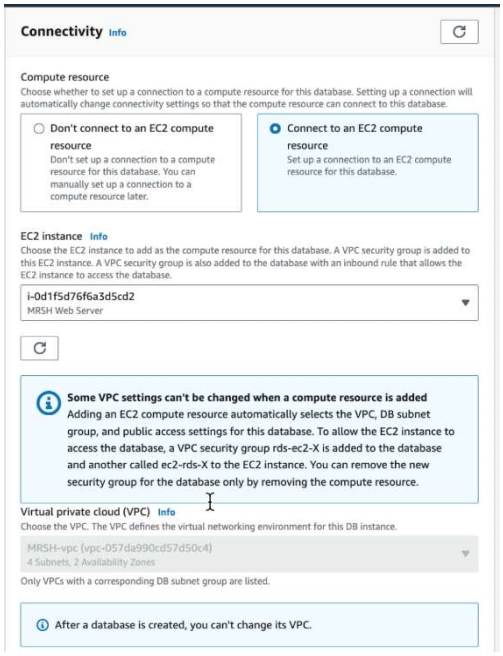
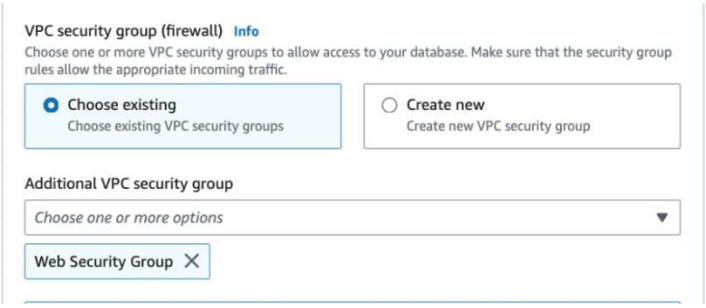


Figure 16: DB Instance Settings 1

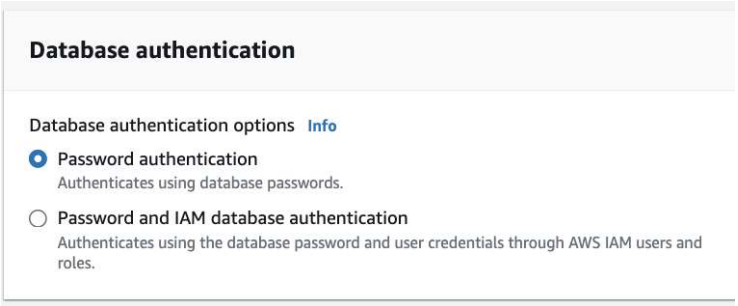
Under Connectivity, I select the option to connect to our MRSH Web Server compute resource.





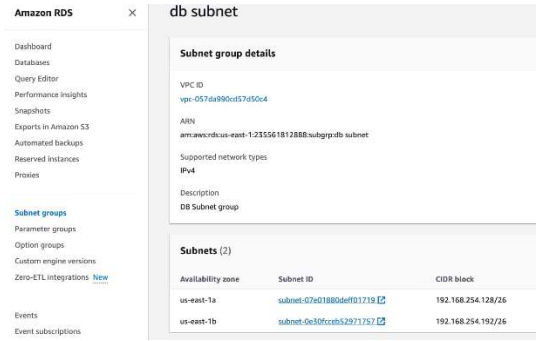
Figures 17 & 18: DB Instance Settings 2

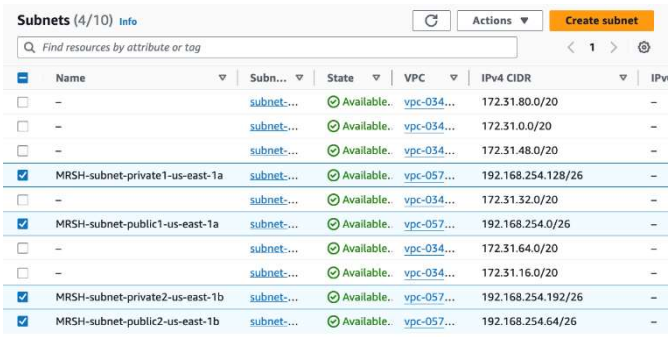
In the figure below, I select Password authentication as it is suitable for my scenario of an SME. This enables easier administration especially in our case of a relatively smaller user community. Additionally, due to the AWS account restrictions, I am unable to utilise IAM capabilities although IAM is of a higher security standard.



Figures 19: DB Instance Settings 3

With the creation of the RDS, we must also ensure that we have 2 private subnets for the allocation of our RDS. For reference, we can see that the 2 subnets (192.168.254.128 & 192.168.254.192) assigned as db subnets are private subnets in our VPC.





Figures 20: DB Instance private subnets allocation

Upon successful connection of the DB and EC2 web server instances, the following will pop up:

Figure 21: EC2-RDS successful connection

RDS-EC2 Web Server Security Group

Sunday, 19 October 2025 10:19 am

RDS-EC2 Web Server Security Group

The VPC, subnets, and security groups are required in order for the DB instance and Web Server to communicate. The VPC and subnets have already been configured in previous sections. Next, a security group to ensure connection is required for private RDS access where I create a new inbound rule allowing for MySQL traffic via port 3306 and the EC2 Web Server's private IP address as the Destination address. From this, we can connect our web server to the RDS instance for the purpose of retrieving or storing data between the web server and database.

Firstly, we create a new security group called WebServerConnect for the abovementioned purpose:

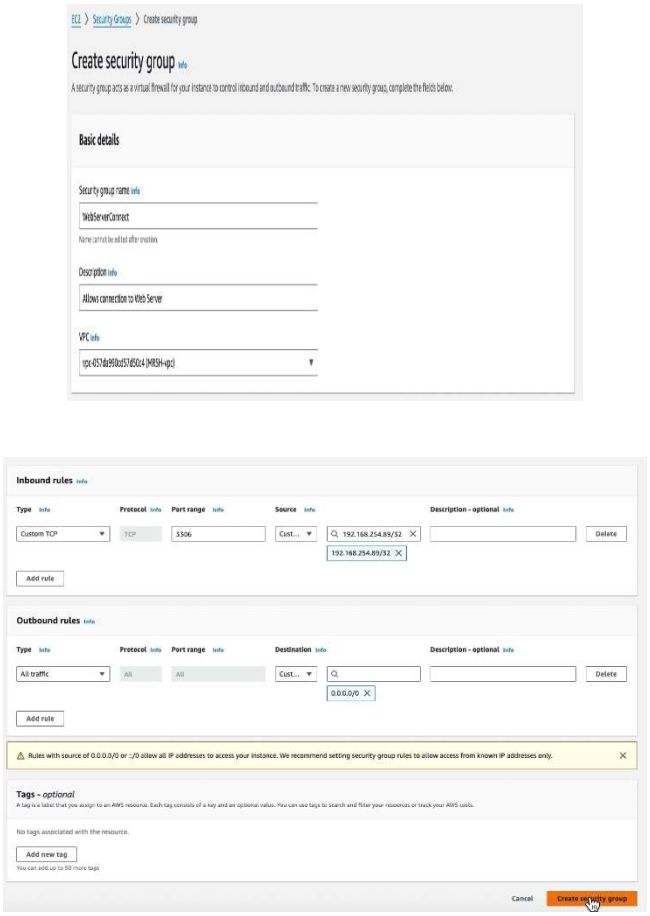


Figure 22: WebServerConnect Security Group

Thereafter, we return to the RDS console and add the WebServerConnect Security Group to our DB instance under "Modify".

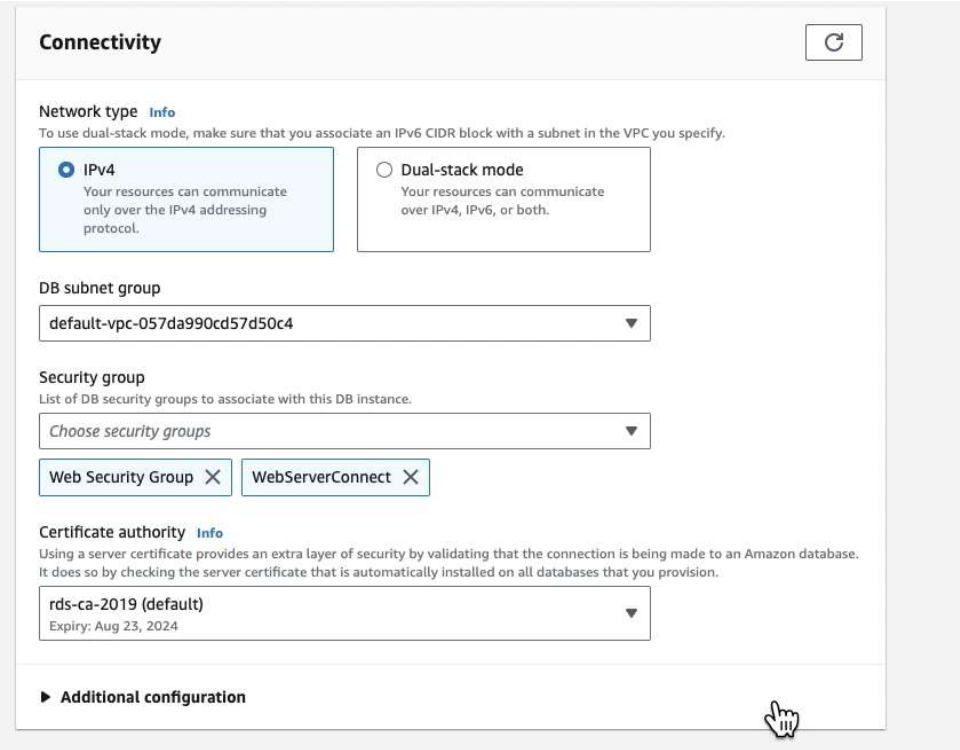


Figure 23: Adding WebServerConnect to DB Instance

After modifying, the WebServerConnect security group should appear under VPC security groups as seen below with the "Active" symbol:

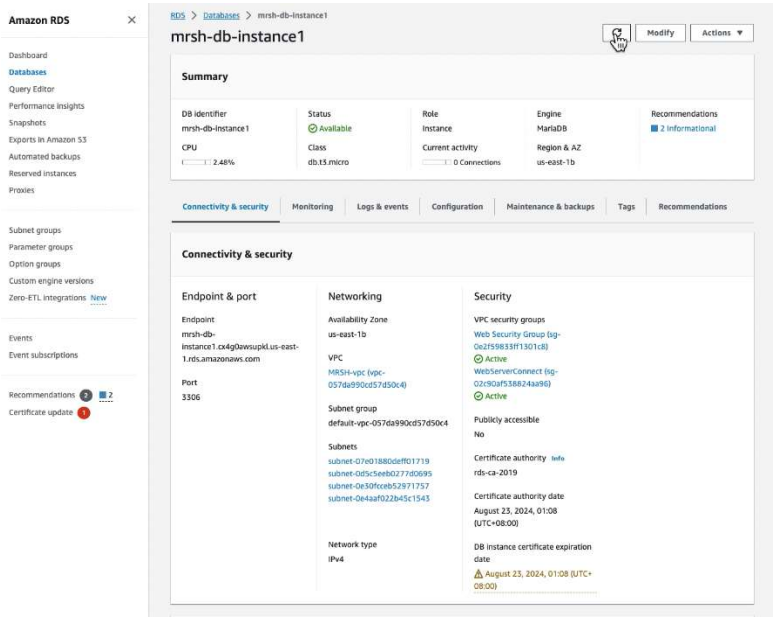


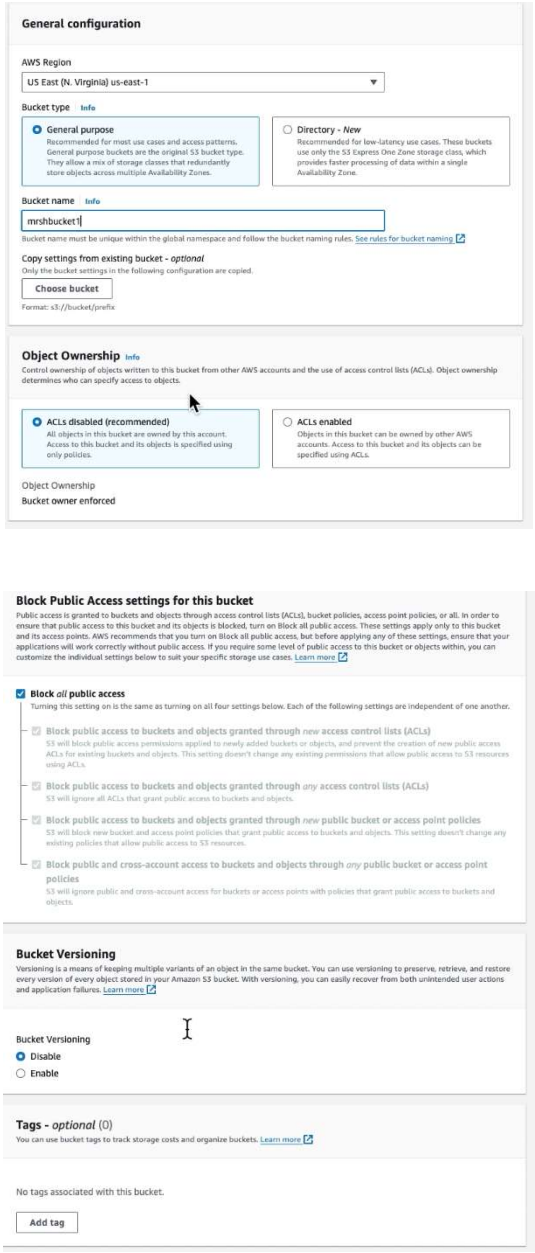
Figure 24: WebServerConnect Security Group successfully added

S3 Bucket

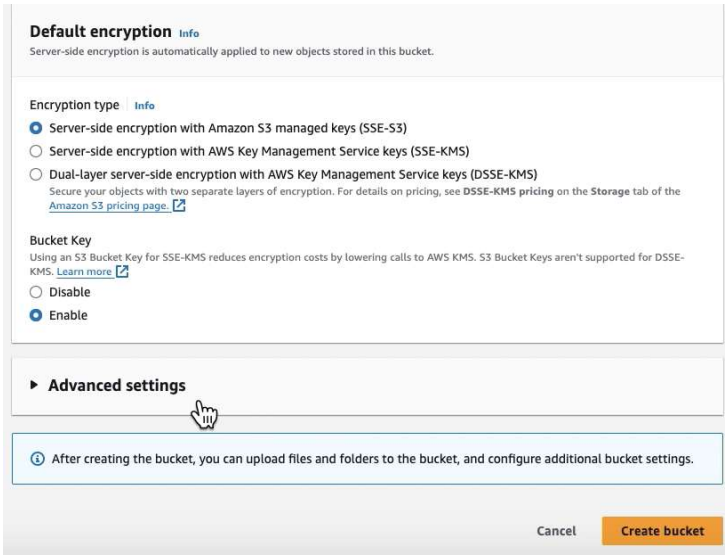
Sunday, 19 October 2025 10:20 am

Bucket

S3 buckets can be created by moving to the S3 console and creating a bucket under “Buckets”. The basic settings for a bucket are as follows:



Figures 25 & 26*: S3 Bucket Set-Up 1*



Figures 27: S3 Bucket Set-Up 2

Due to account restrictions, I am unable to connect the S3 bucket to the EC2 Web Server instance due to the unavailability of IAM functions. Below is a snapshot from the AWS website detailing the need of IAM to enable me to do so.

Short description

To connect to your S3 buckets from your EC2 instances, you must do the following:

1. Create an AWS Identity and Access Management (IAM) profile role that grants access to Amazon S3.
2. Attach the IAM instance profile to the instance.
3. Validate permissions on your S3 bucket.
4. Validate network connectivity from the EC2 instance to Amazon S3.
5. Validate access to S3 buckets.

Figure 28: AWS guide S3-EC2 connect

Access Control List (ACL)

Sunday, 19 October 2025 10:21 am

Network Access Control List (ACL)

We use Network ACLs to control inbound and outbound traffic at the subnet level for added security. To add ACLs, we go on the VPC console and click on Network ACLs. Then, we select our MRSH VPC:

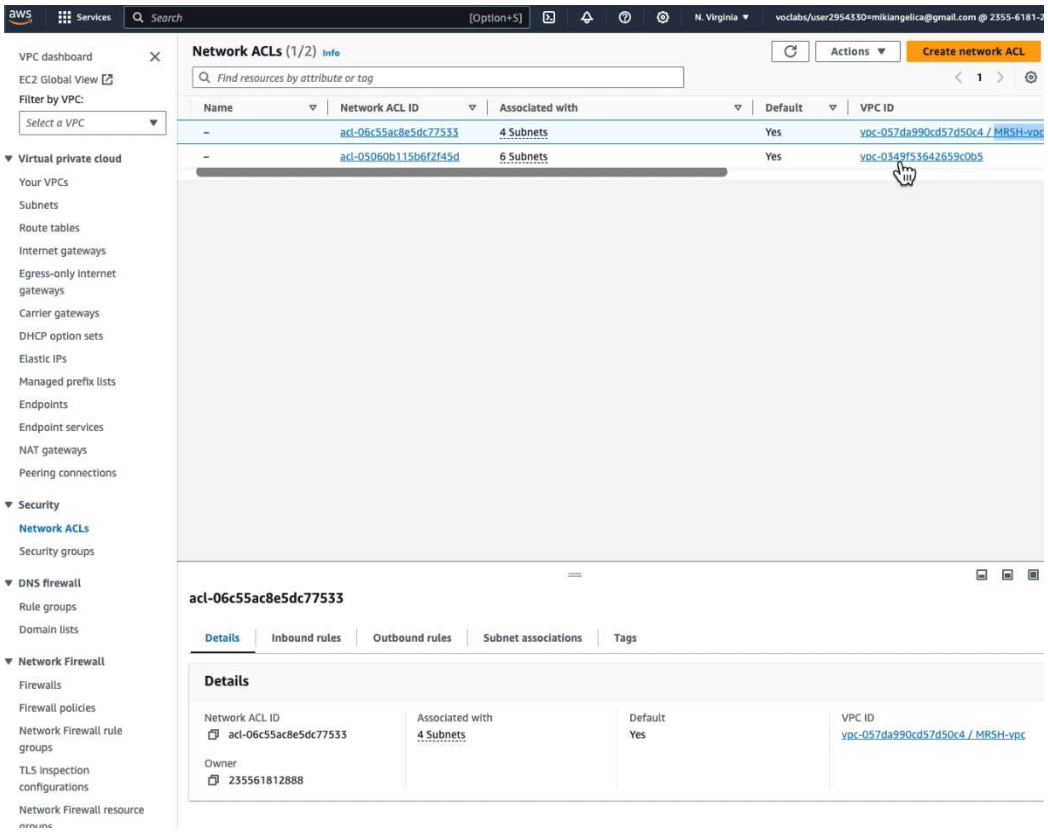


Figure 29: Network ACLs page.

From this page, we can edit inbound and outbound rules under “Details” and add in the following rules. The rules below are a Custom network ACL for a IPV4 supported VPC. It allows for two-way traffic for HTTP, HTTPS (for internet connection), as well as rule 140 which covers ephemeral ports 32768-65535 (example). SSH and RDP is also enabled for remote access from a specific IP address. The use of Network ACLs adds an additional layer of security for our subnets as all traffic that are not specifically Allowed access from our ACLs will be directed to rule * which Denies all ineligible traffic.

acl-06c55ac8e5dc77533

Details | Inbound rules | Outbound rules | Subnet associations | Tags

Inbound rules (6)

Edit inbound rules

Filter inbound rules

1

Rule number	Type	Protocol	Port range	Source	Allow/Deny
100	HTTP (80)	TCP (6)	80	0.0.0.0/0	Allow
110	HTTPS (443)	TCP (6)	443	0.0.0.0/0	Allow
120	SSH (22)	TCP (6)	22	42.0.0.0/8	Allow
130	RDP (3389)	TCP (6)	3389	42.0.0.0/8	Allow
140	Custom TCP	TCP (6)	32768 - 65535	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Figure 30: Network ACL Inbound rules

acl-06c55ac8e5dc77533

Details | Inbound rules | Outbound rules | Subnet associations | Tags

Outbound rules (5)

Edit outbound rules

Filter outbound rules

1

Rule number	Type	Protocol	Port range	Destination	Allow/Deny
100	HTTP (80)	TCP (6)	80	0.0.0.0/0	Allow
110	HTTPS (443)	TCP (6)	443	0.0.0.0/0	Allow
120	SSH (22)	TCP (6)	22	42.0.0.0/8	Allow
140	Custom TCP	TCP (6)	32768 - 65535	0.0.0.0/0	Allow
*	All traffic	All	All	0.0.0.0/0	Deny

Figure 31: Network ACL Outbound rules

Security Groups

Sunday, 19 October 2025 10:22 am

Security Groups

Security Groups are yet another security feature that protects our instances from unauthorised traffic. For our EC2 Web Server instance, we can see in Figure 32 that it has been assigned 3 Security Groups;

SSH – This security group allows for traffic via port 22 (SSH) connected to a specific IP address; which means that only that IP address can connect to the EC2 instance via SSH. All other SSH attempts from other IP addresses are denied. This is illustrated in Figure 33.
Web Security Group – This group allows for HTTP and HTTPS traffic from all sources, as shown in Figure 34.
ec2-rds-1 – This security group was automatically created upon setting up the DB instance (RDS) and selecting the option to connect to this web server. Figure 35 shows the outbound rule for MYSQL via port 3306, enabling connectivity between the web server and the DB instance.

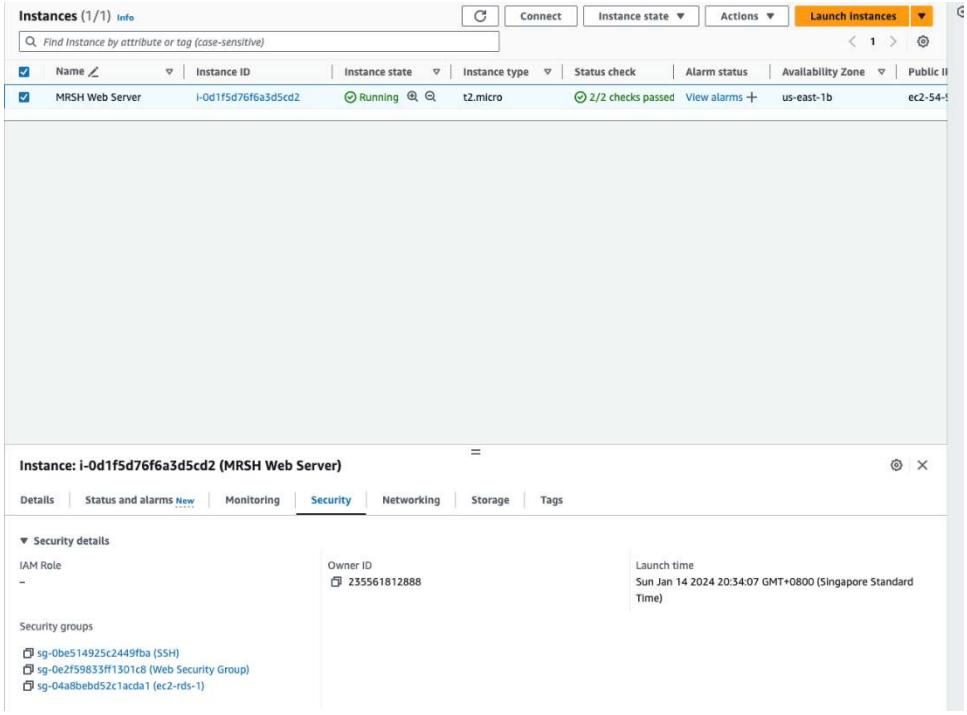


Figure 32: EC2 Security Groups

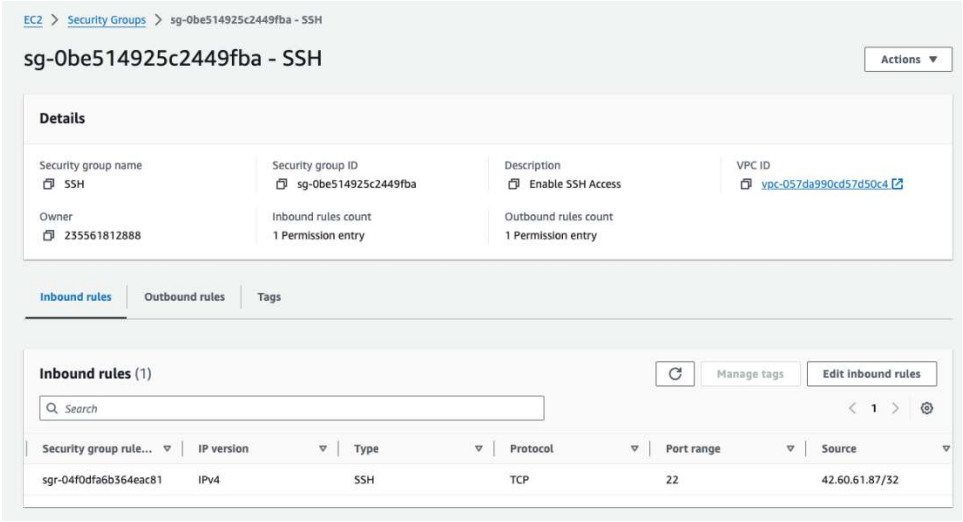


Figure 33: EC2 SSH Security Group

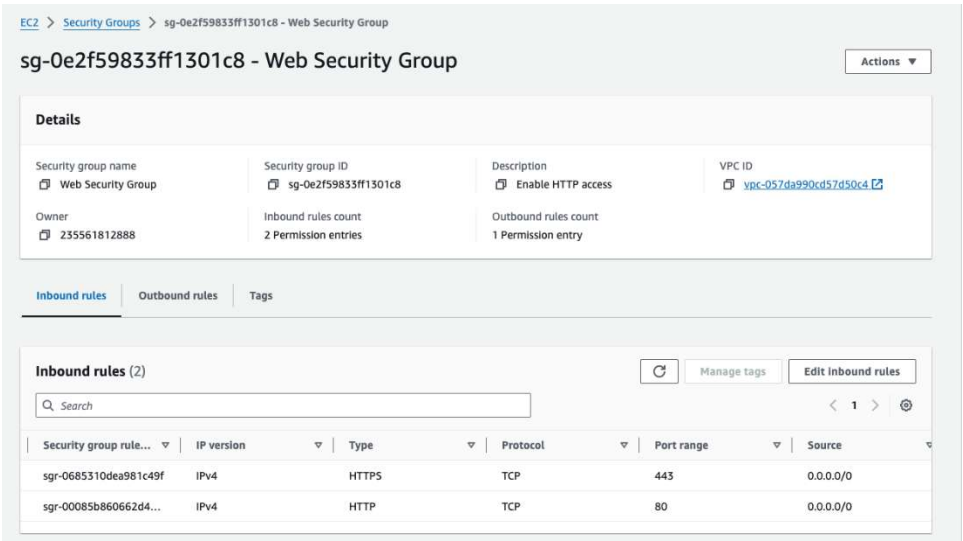


Figure 34: EC2 Web Security Group

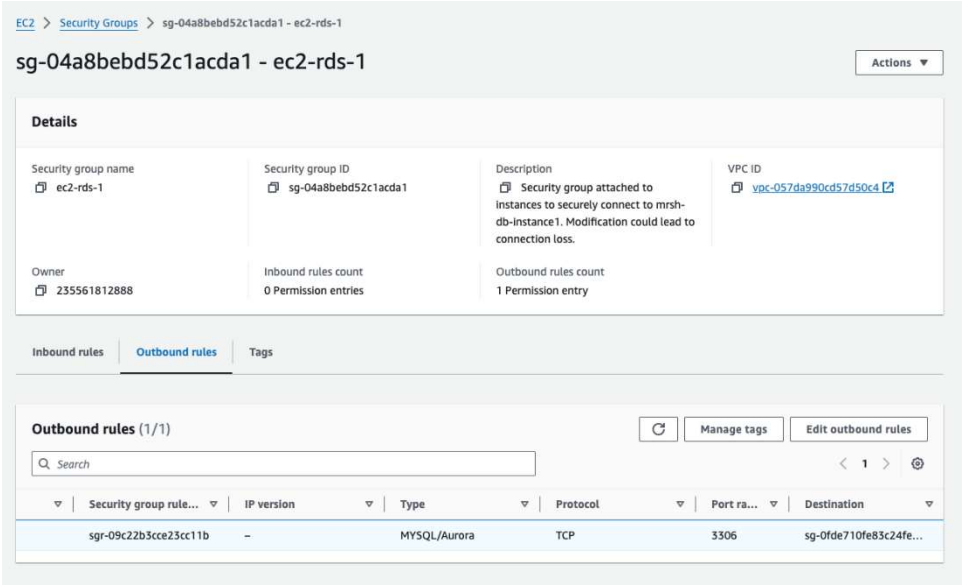


Figure 35: EC2 ec2-rds-1 Security Group

As for the DB Instance, the final security group configuration is depicted in Figure 36under “VPC Security groups”, containing only the WebServerConnect Security Group for the DB instance to communicate with the web server. All other traffic is denied access for added security on top of our DB instance being placed in a private subnet.

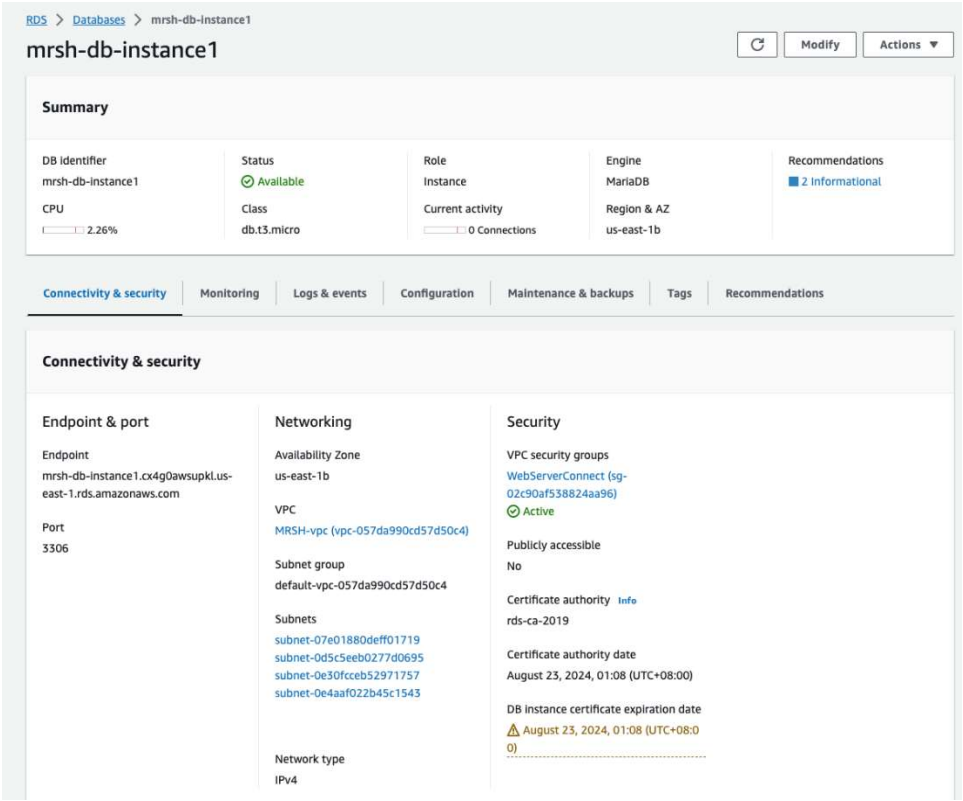


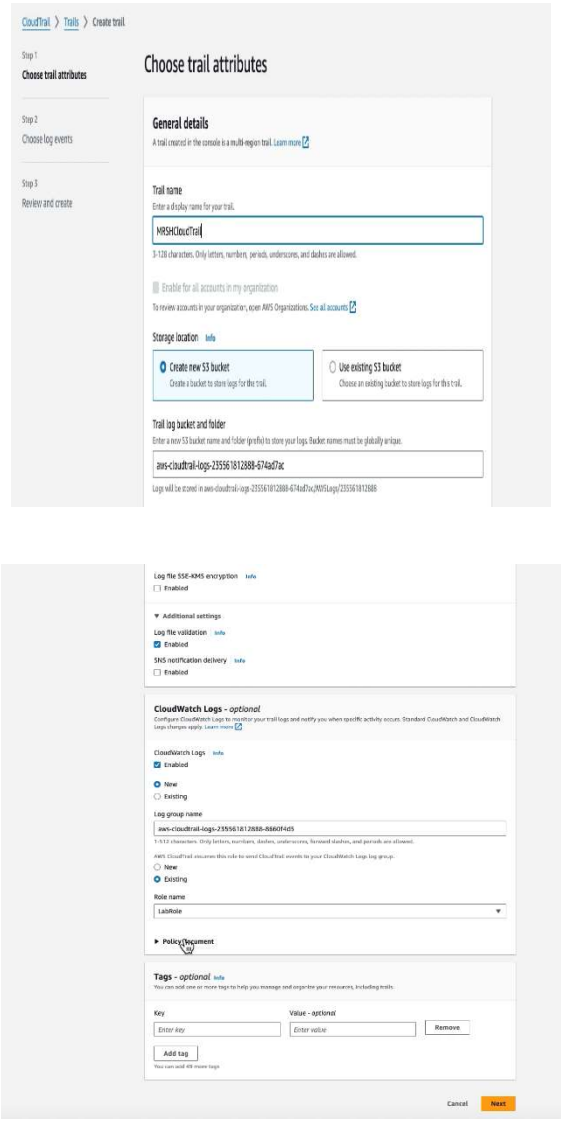
Figure 36: RDS DB instance Security Group

CloudTrail Trail

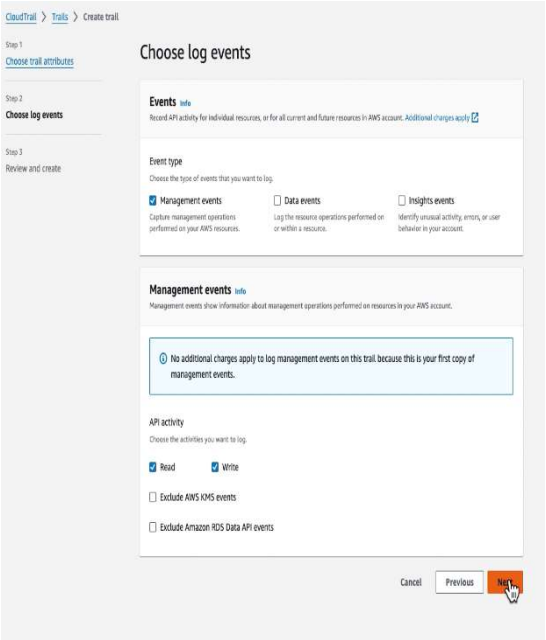
Sunday, 19 October 2025 10:25 am

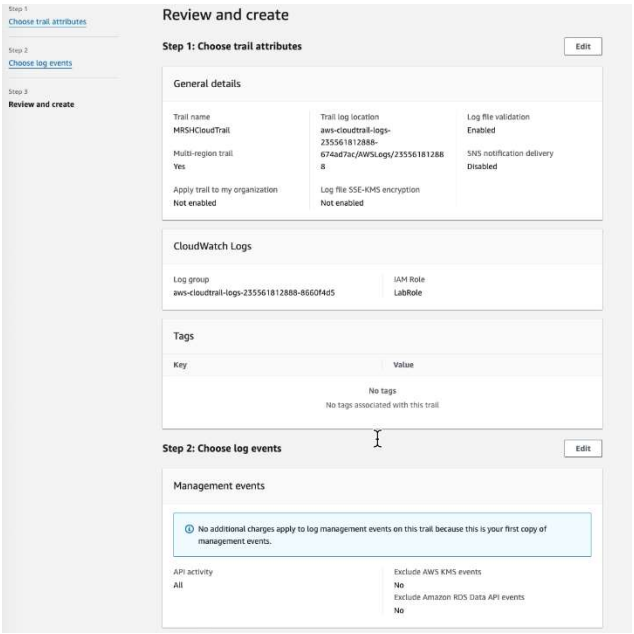
CloudTrail Trail

CloudTrail is an AWS function that lets us track user/role/service activities down to each action (AKA events). This helps to monitor our cloud environment and conduct audits and troubleshooting. I created a CloudTrail trail by entering the CloudTrail console and clicking on “Create trail” under Trails. I also enabled CloudWatch Logs in Figures 37 & 38 to centralise and monitor my trail logs.



Figures 37 & 38: CloudTrail trail Set-up 1





Figures 39 & 40: CloudTrail trail Set-up 2

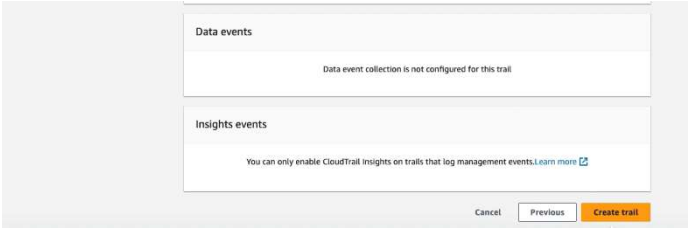


Figure 41: CloudTrail trail Set-up 3

After completing the trail set-up, the following pop-up emerges, which may be due to the security restrictions placed on AWS accounts that are used for labs:

Figure 42: Trail Creation Error

The new trail will appear under Trails:

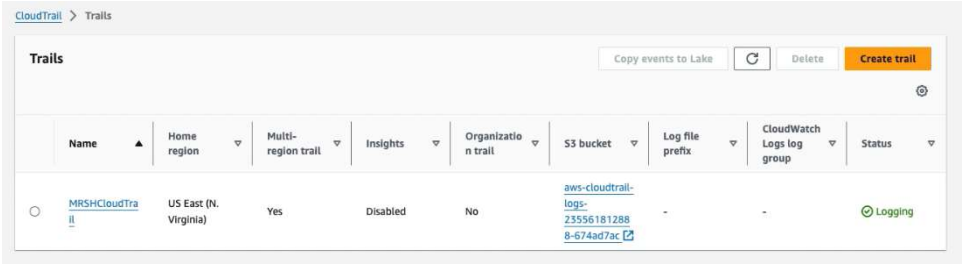


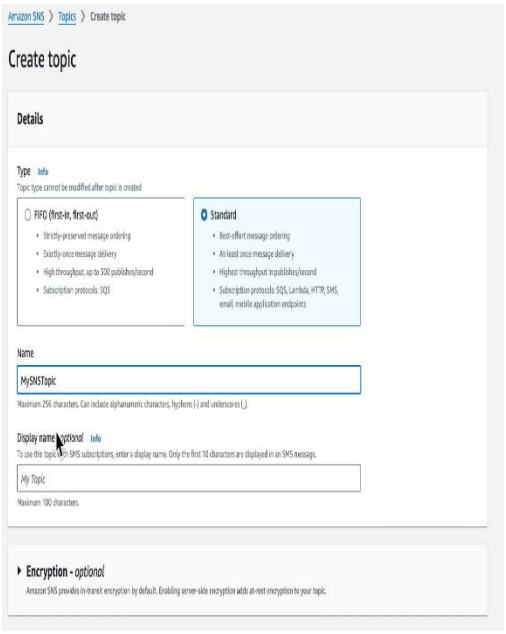
Figure 43: Trails

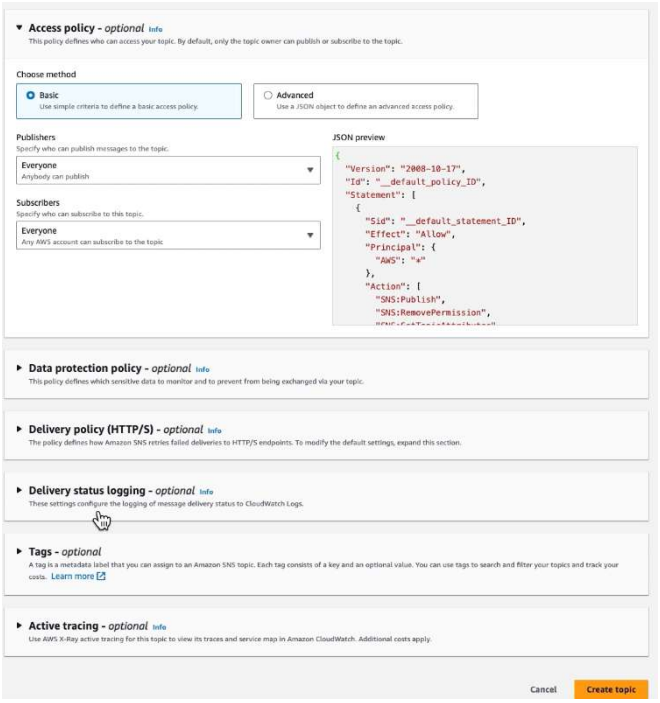
Simple Notification Service (SNS)

Sunday, 19 October 2025 10:26 am

Simple Notification Service

Amazon Simple Notification Service (SNS) is an AWS service that provides notifications from publishers to subscribers. We can utilise this service to send us notification messages whenever a certain event occurs. We create an SNS by going onto the SNS and clicking on “Create Topic” under Topics. The SNS Set-Up is as follows:





Figures 44 & 45: SNS Set-Up

After setting up the topic, we can create subscriptions to the topic for subscribers to receive notifications.

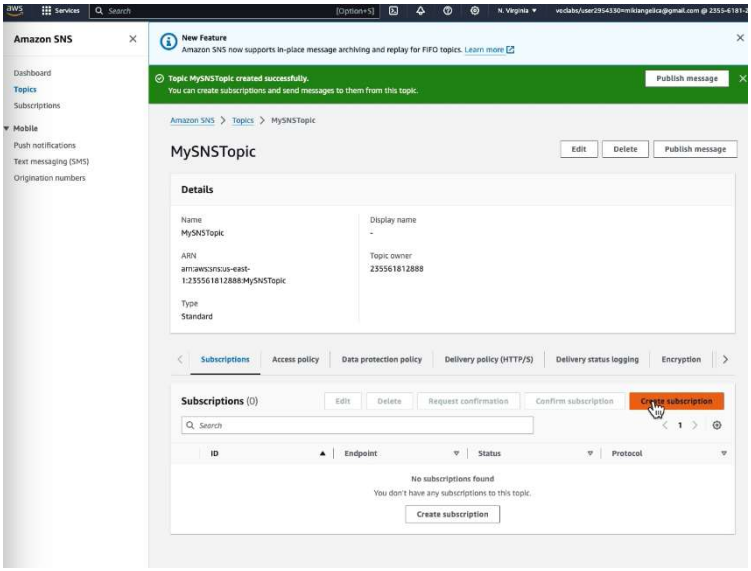


Figure 46: SNS successful creation

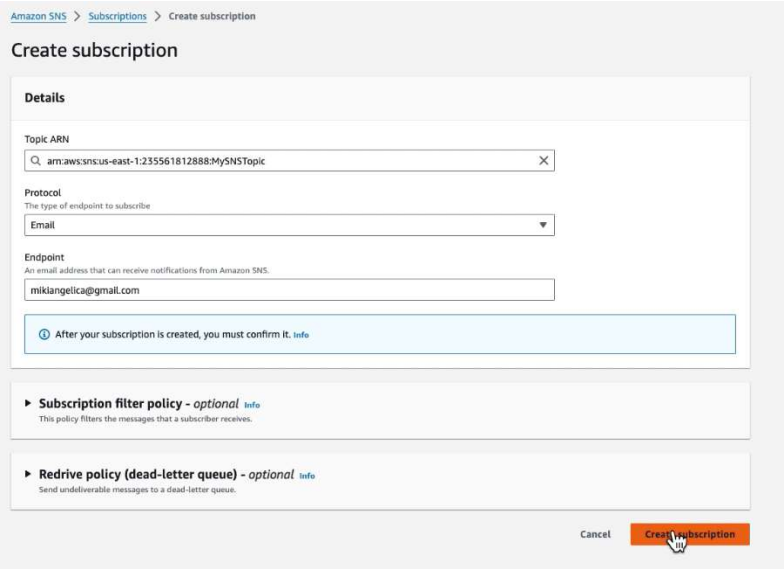


Figure 47: Create Subscription

Upon subscription, a message will be sent to the subscriber to confirm subscription:

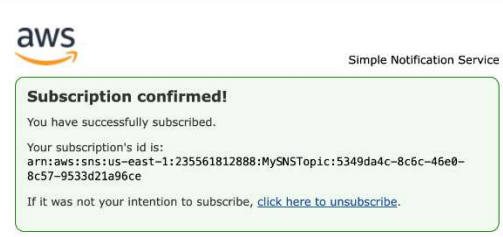


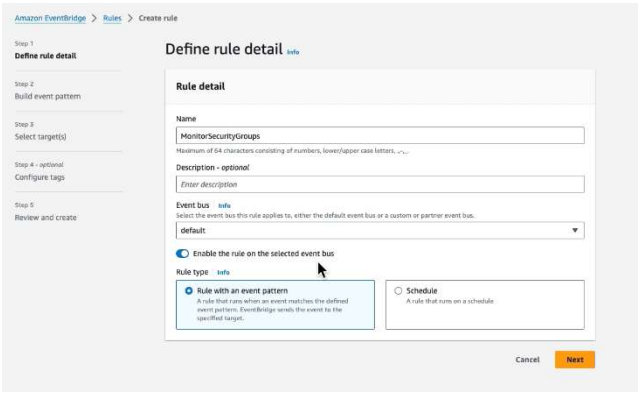
Figure 48: Subscription confirmed

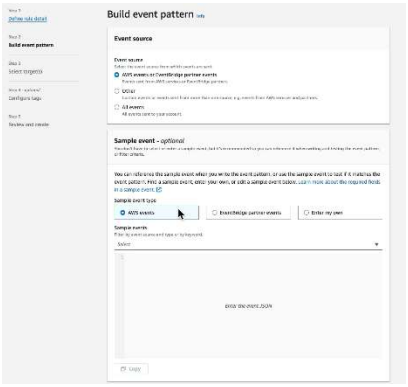
EventBridge

Sunday, 19 October 2025 10:27 am

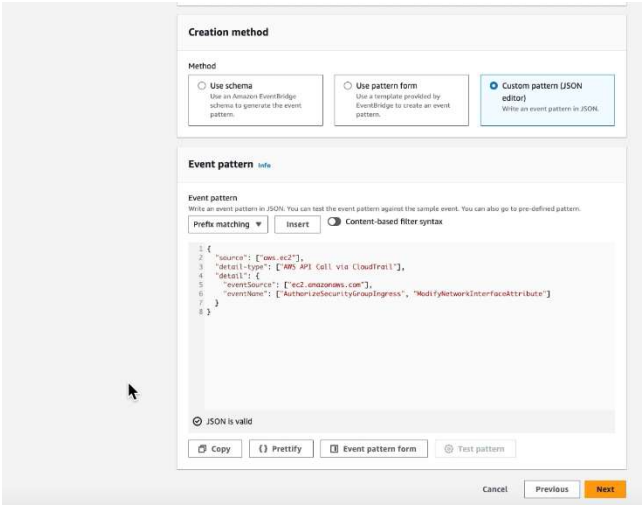
EventBridge Rule

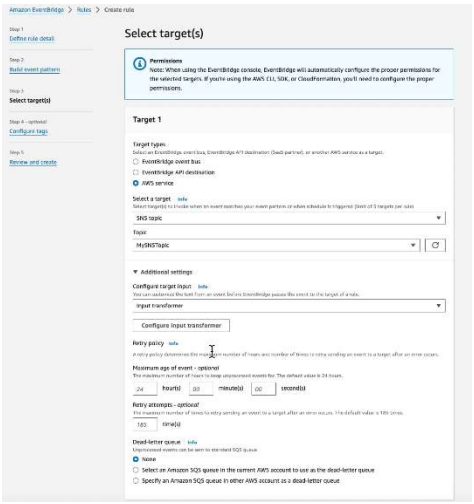
In this section, I create an EventBridge Rule using AWS EventBridge which will detect the occurrence of a specified event and send it to our target (SNS). The set-up is as follows:





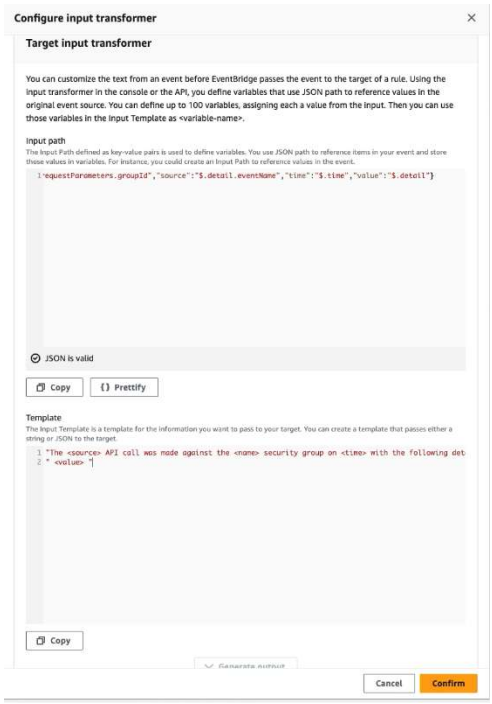
Figures 49 & 50: EventBridge Rule Set-Up 1





Figures 51 & 52: EventBridge Rule Set-Up 2

Click on Configure input transformer to paste the following code that configures the message:



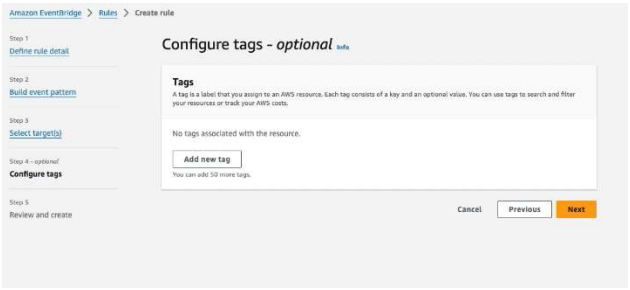


Figure 53 & 54: EventBridge Rule Set-Up 3

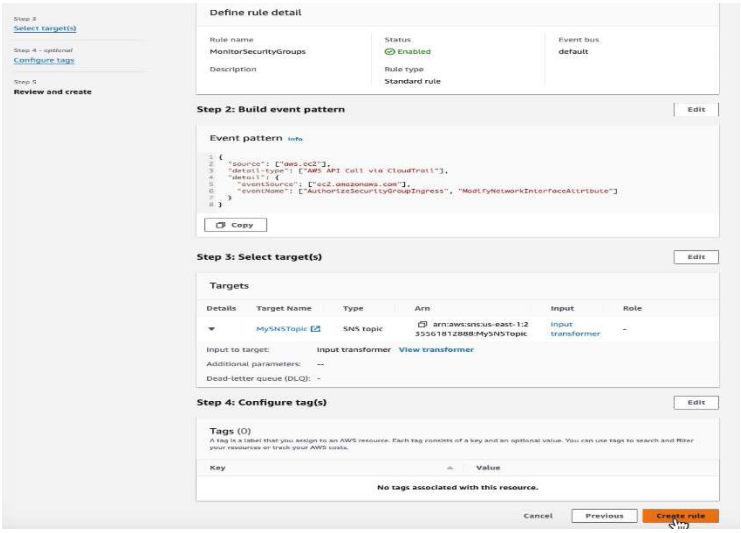


Figure 55: EventBridge Rule Set-Up 4

To test the EventBridge rule that we have set up above, we go back onto the EC2 console and select our MRSH Web Server instance > Security > click on Web Security Group link > Edit inbound rules > add SSH port:

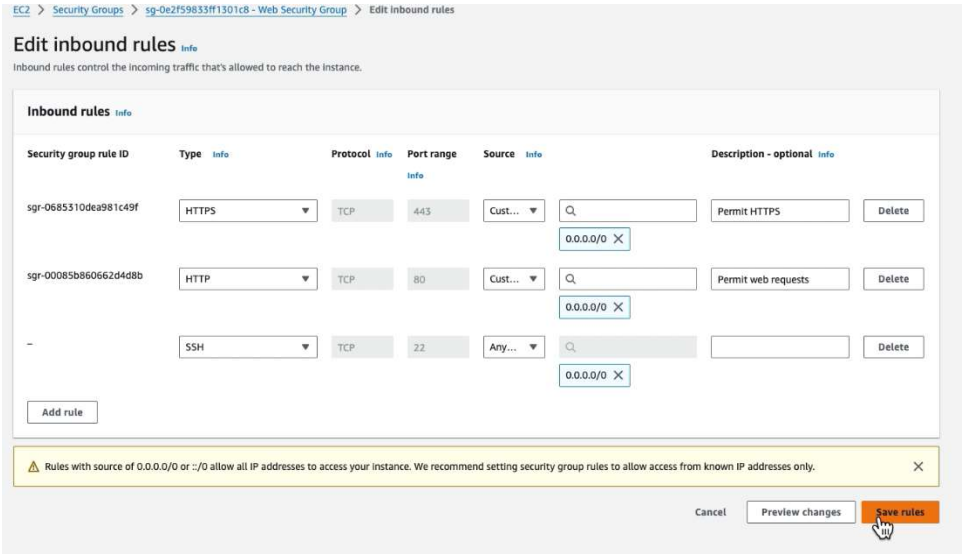


Figure 56: EventBridge Test: Add SSH port to EC2 Security Group

Then we return to CloudTrail > Event history. After a minute or two, the event "AuthorizeSecurityGroupIngress" should appear:

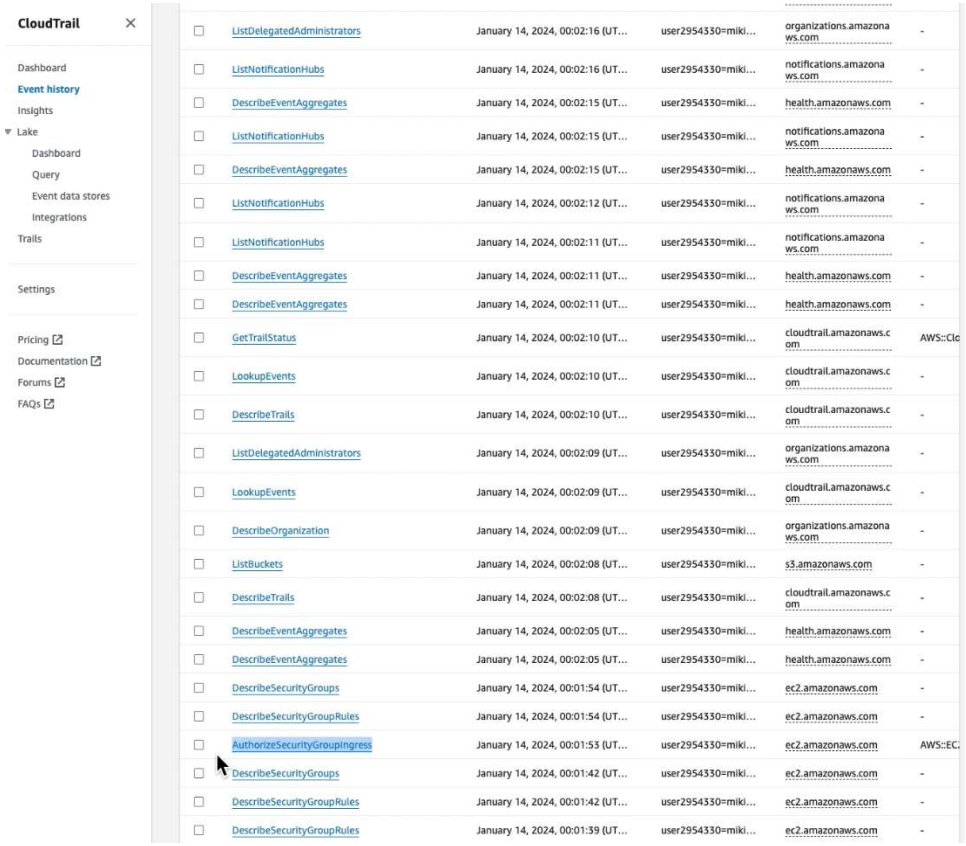


Figure 57: EventBridge Test: Check CloudTrail Event History

Upon clicking onto the event “AuthorizeSecurityGroupIngress”, we will be able to view its event record containing important details such as userIdentity, eventTime, and awsRegion:

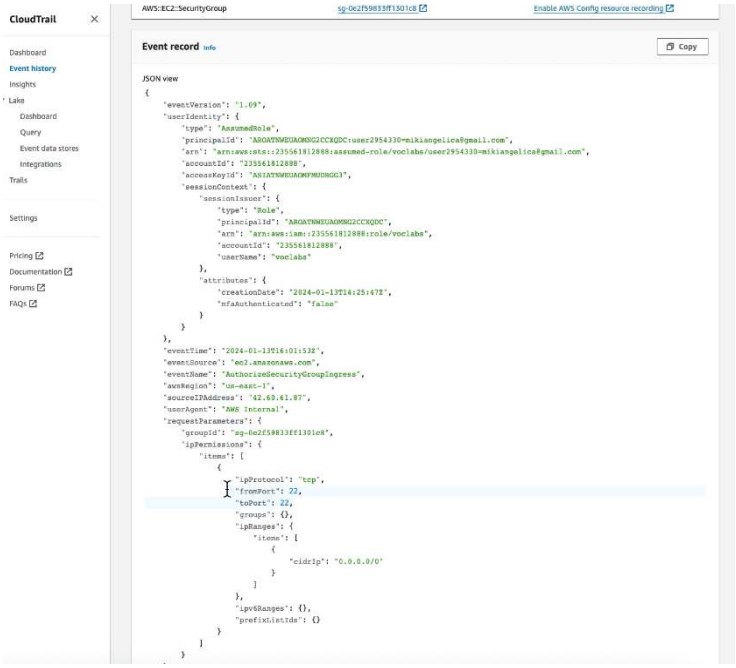


Figure 58: EventBridge Test: ‘AuthorizeSecurityGroupIngress’ Event Record

Subscribers to the relevant SNS topic for this EventBridge rule will receive the following email signalling test success:

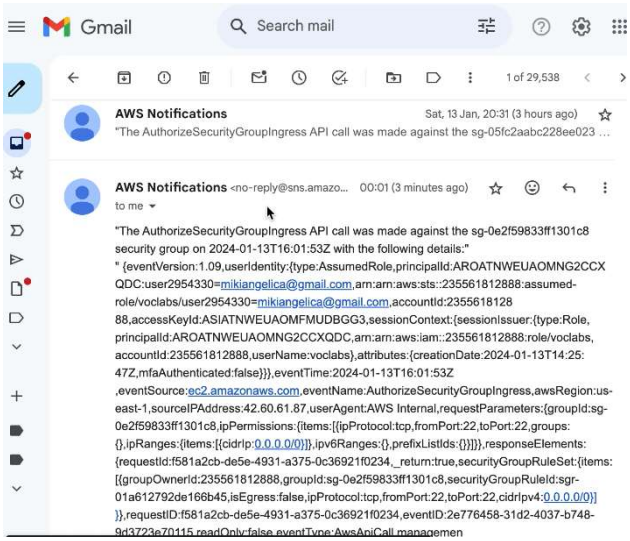


Figure 59: EventBridge Test: Email Notification

Cloudwatch Alarm

Sunday, 19 October 2025 10:30 am

Cloudwatch Alarm

Using CloudWatch, we can set alarms to monitor and detect CloudWatch metrics that we define. Notifications will also be sent if defined metric thresholds are met. CloudWatch Alarms can also be used to automatically stop, terminate, reboot, or recover our EC2 instances.

To create an alarm, we go to CloudWatch console > Navigation bar > Log groups > Select your Trail log group > Actions > Create metric filter:

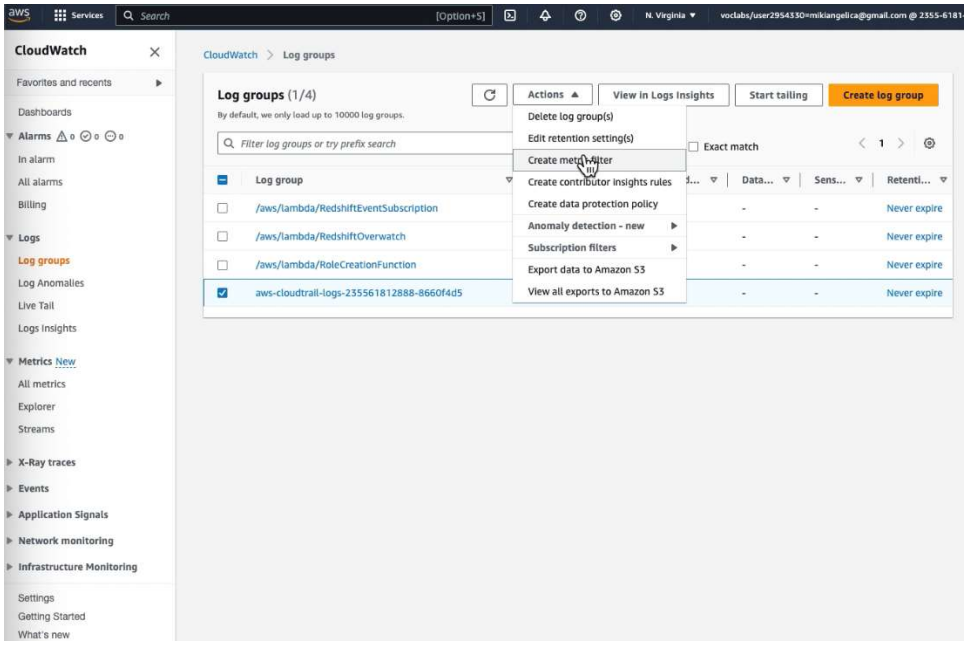
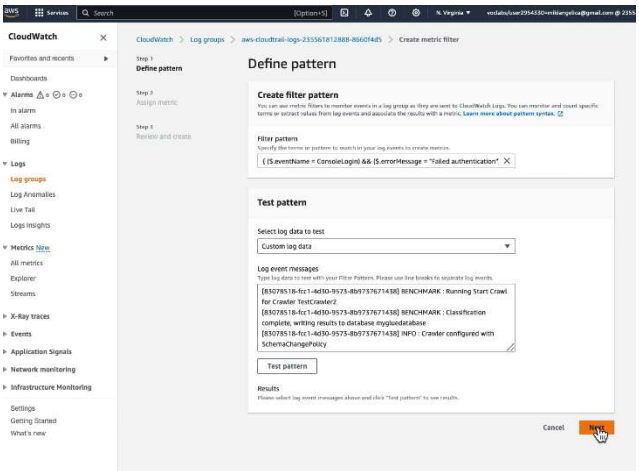


Figure 60: CloudWatch Log groups > Create metric filter



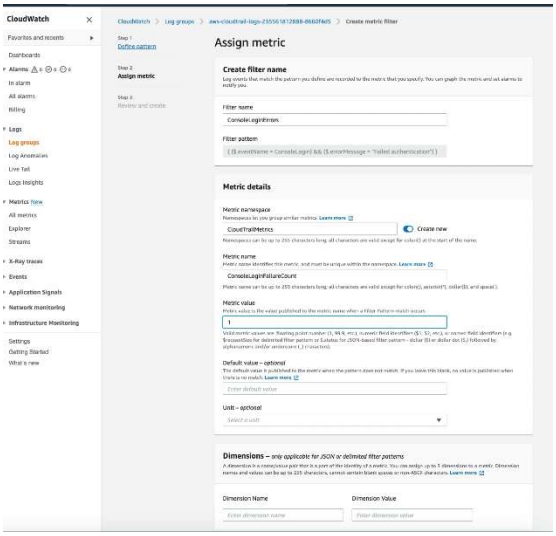


Figure 61 & 62: Metric Filter Set-up 1

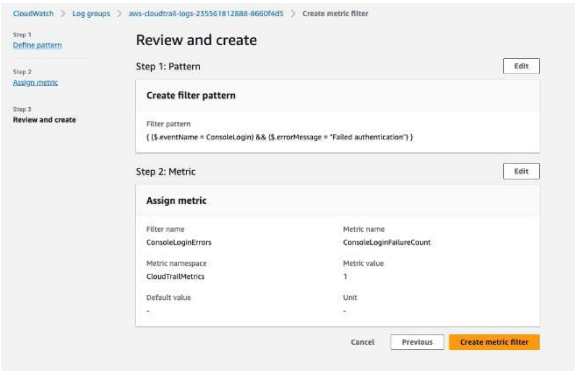


Figure 63: Metric Filter Set-up 2

After setting up the metric filter that detects and counts login failures, we create an alarm that detects when the Login Failure count reaches our defined threshold (3 or more), which will then send a notification to our topic subscribers.

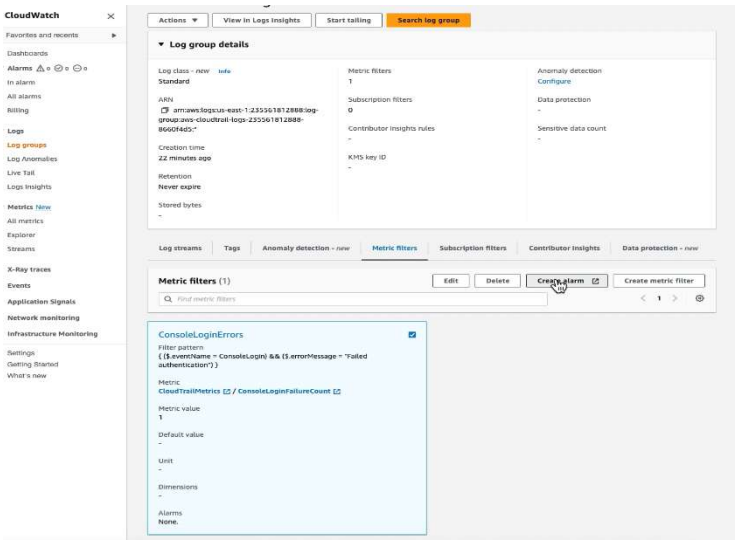
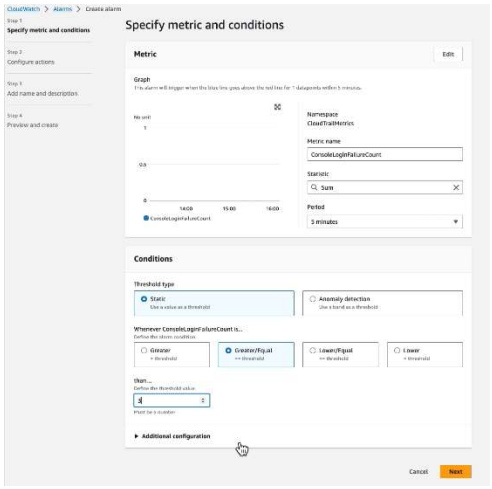
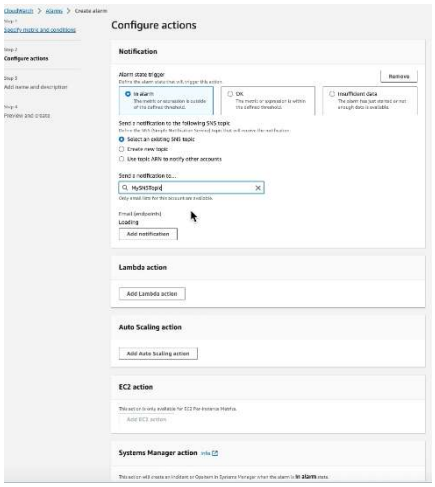
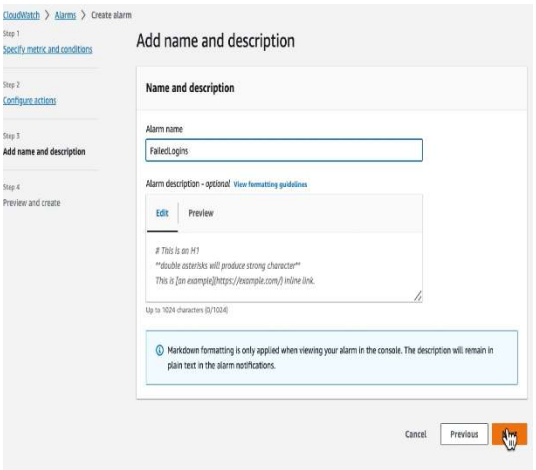


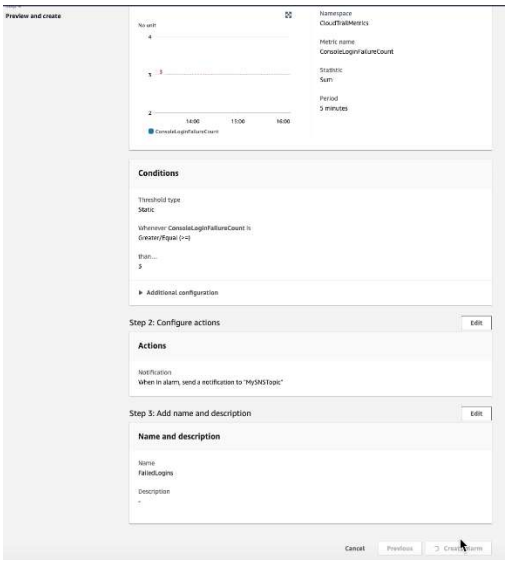
Figure 64: Metric Filters





Figures 65 & 66: Alarm Set-up 1





Figures 67 & 68: Alarm Set-up 2

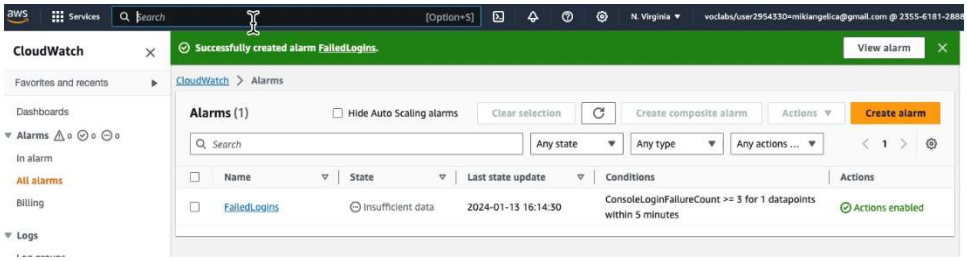


Figure 69: Alarms

Due to the inability to access IAM features, I am unable to test the alarms above from Failed Logins.

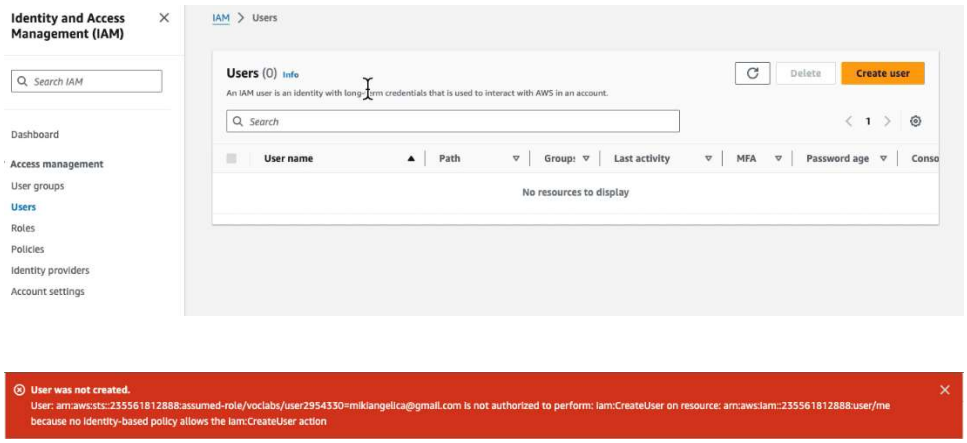


Figure 69: IAM console

Querying CloudTrail Logs

Sunday, 19 October 2025 10:33 am

Querying CloudTrail logs

To query CloudTrail logs, we go back to CloudWatch > Navigation Bar > Logs Insights > Select your log group > Paste code > Run query > A Histogram should appear below for our FailedLogins log but because we are unable to test, no data is shown.

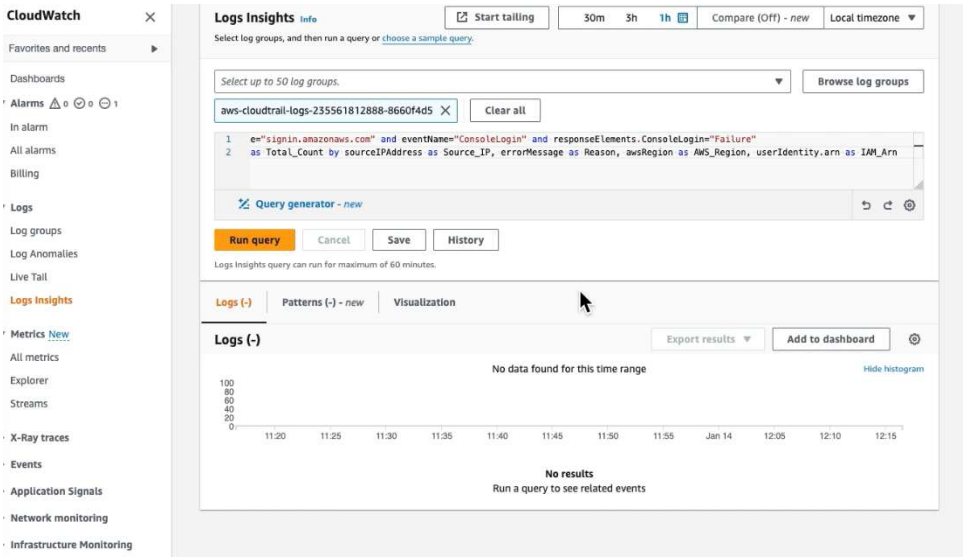


Figure 70: Logs Insights

Auto-scaling and Load Balancing

Sunday, 19 October 2025 10:33 am

Auto-Scaling and Load Balancing

Load-Balancing

We create an instance-type Target Group named “MRSHGroup1. Target groups define the destination to send incoming traffic from load balancers to.

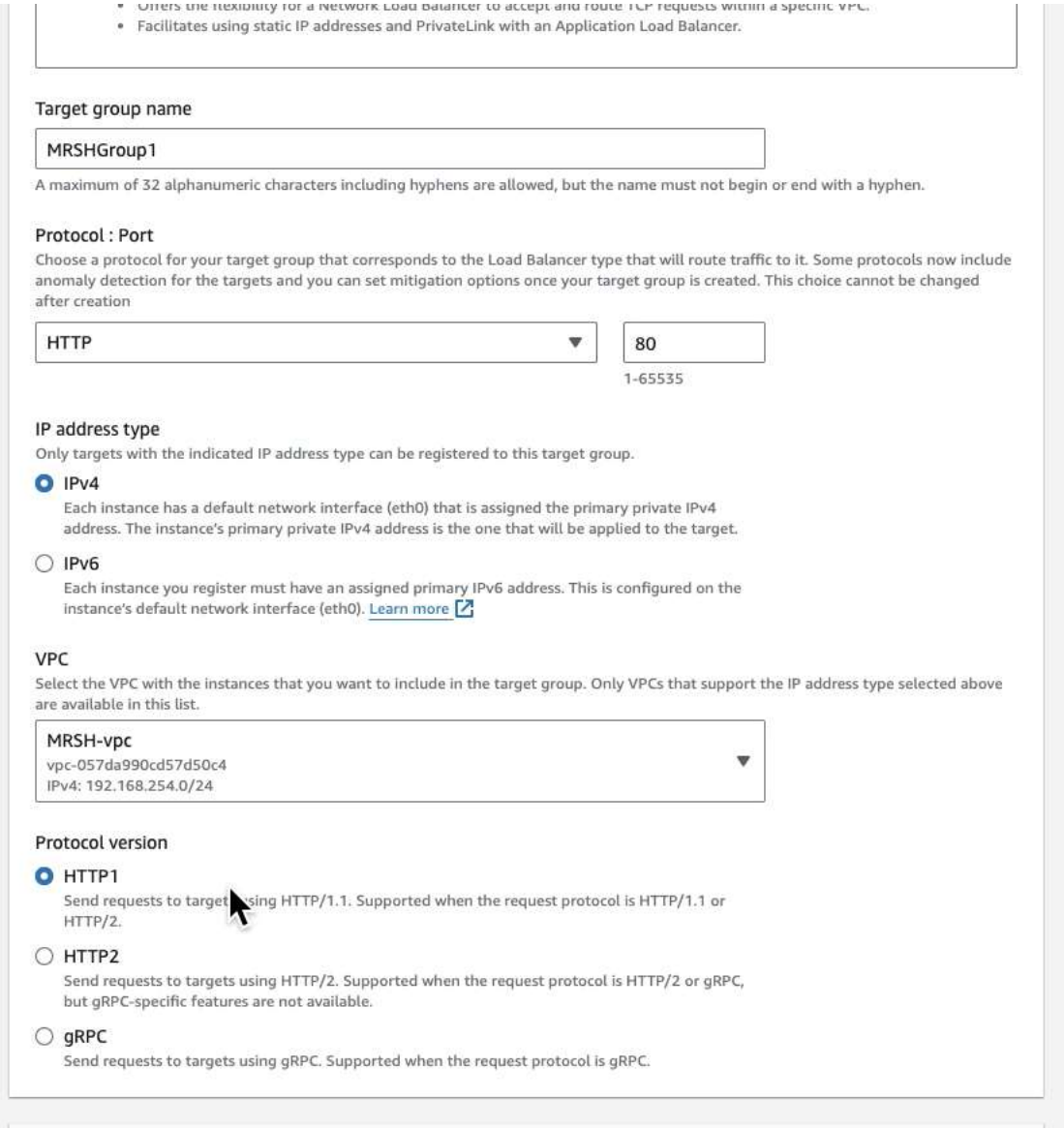


Figure 71: Target Group MRSHGroup1

Then, we create a Load Balancer. Load balancing allows MRSH to spread traffic across multiple instances and availability zones to accommodate high or low traffic periods. In this case, we will be making an Application Load Balancer so that we can send traffic to our instance-type target group. We create a Load Balancer mapped to our 2 public subnets for our MRSH-vpc. This Load Balancer will be assigned the Web Security Group.

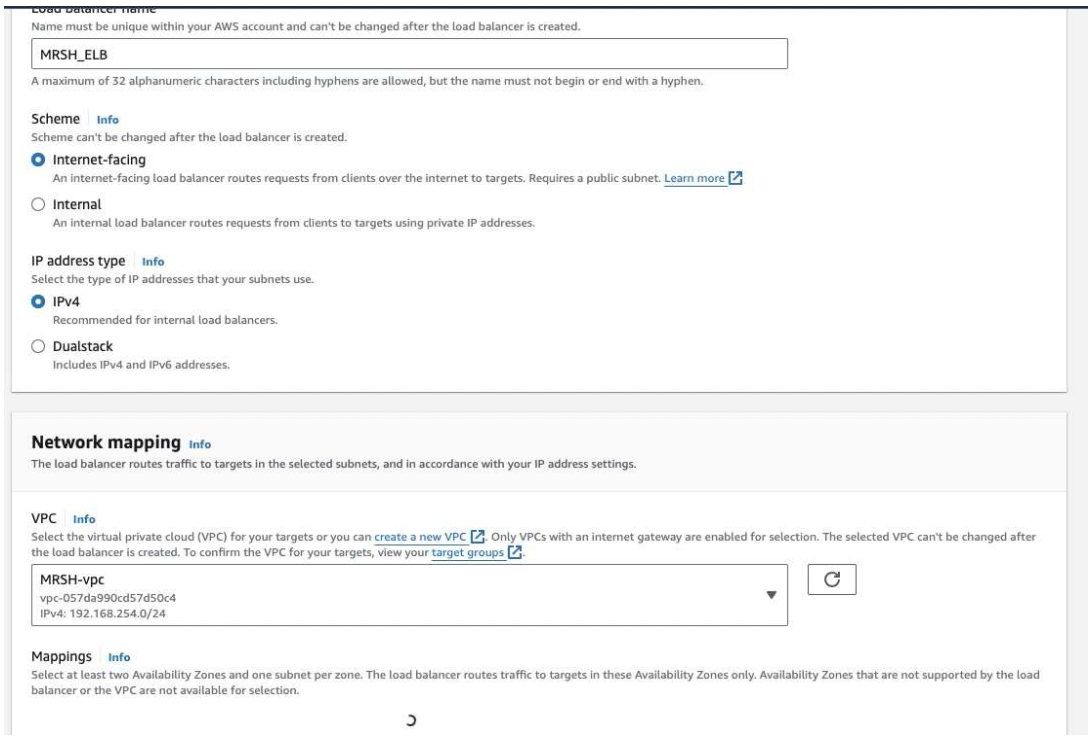


Figure 72: Load Balancer MRSH_ELB

We connect the Load Balancer to our MRSHGroup1 Target Group created prior:

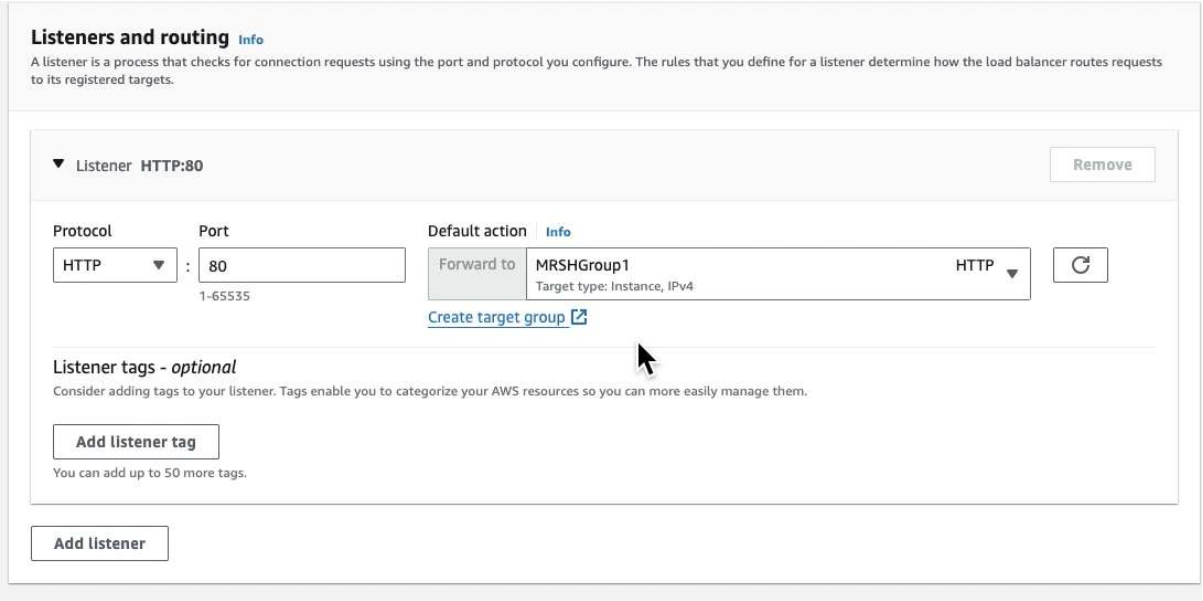


Figure 73: Connection to MRSHGroup1 Target Group

Auto-scaling

On the EC2 console, we create an image for our MRSH Web Server as in the figure below for autoscaling. This allows us to save boot disk contents to launch instances with identical initial contents.

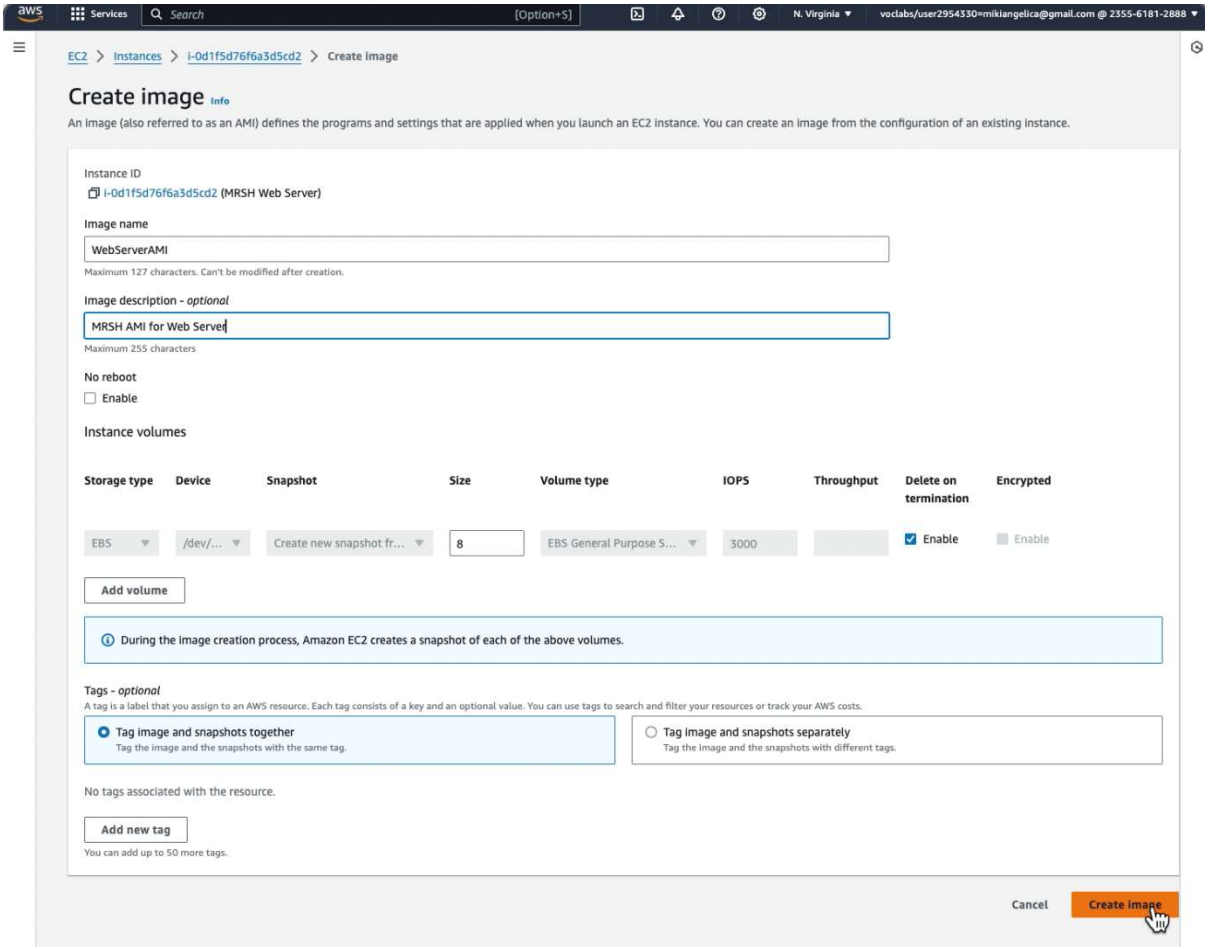


Figure 74: WebServerAMI Image

Next, we create the MRSHConfig Launch Template for our Auto Scaling group to use when launching EC2 instances. This launch template will contain a default template instance containing information such as AMI (using the image created prior), instance type, key pair, and security group. We attach this launch template to our auto scaling group “MRSH Auto Scaling Group” and load balancer “MRSH_ELB”. In our Auto Scaling, we input a desired capacity of 2 instances, a minimum of 2, and a maximum of 6. The Auto Scaling feature will align with the Load Balancer and add or remove instances as needed, keeping between 2 and 6 instances at any one time.

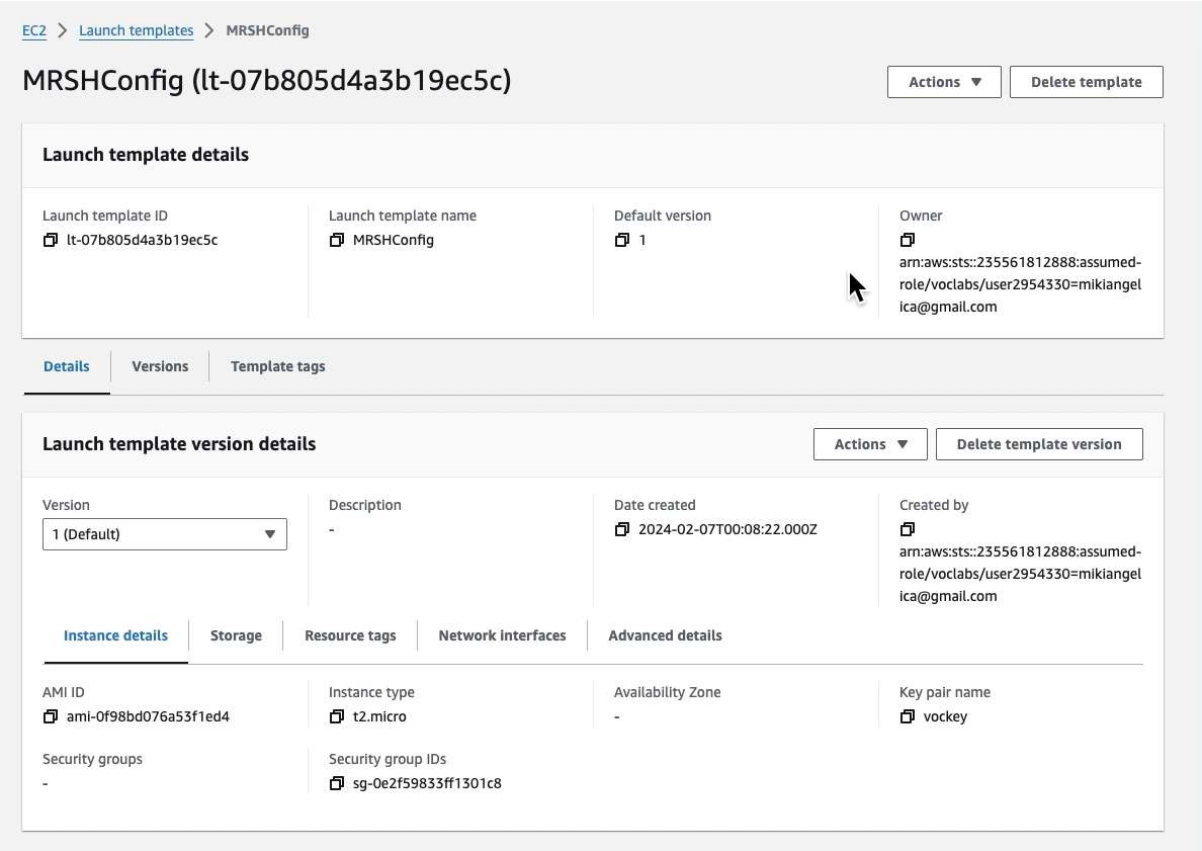


Figure 75: Launch Template MRSHConfig

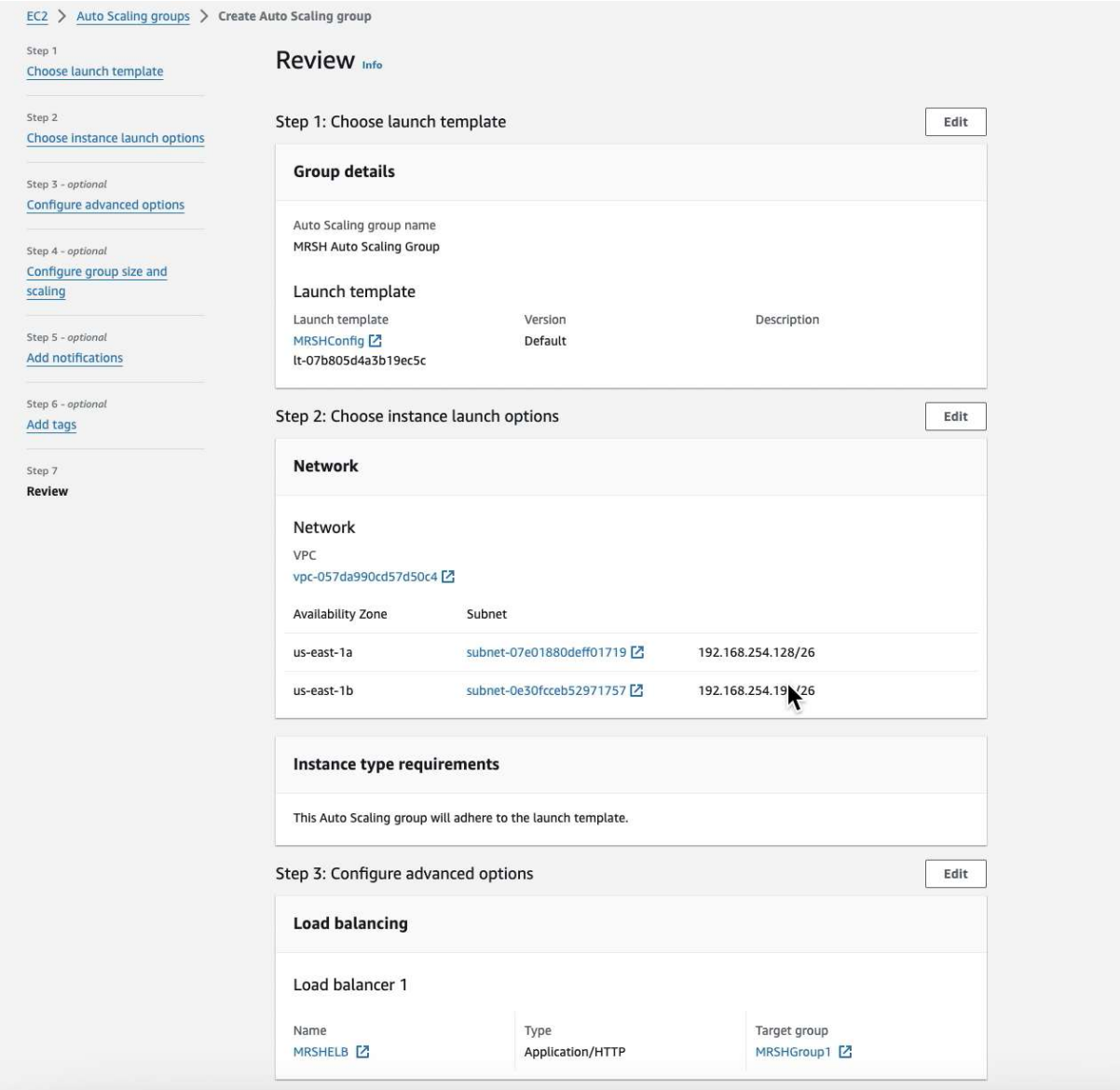


Figure 76: MRSH Auto Scaling Group

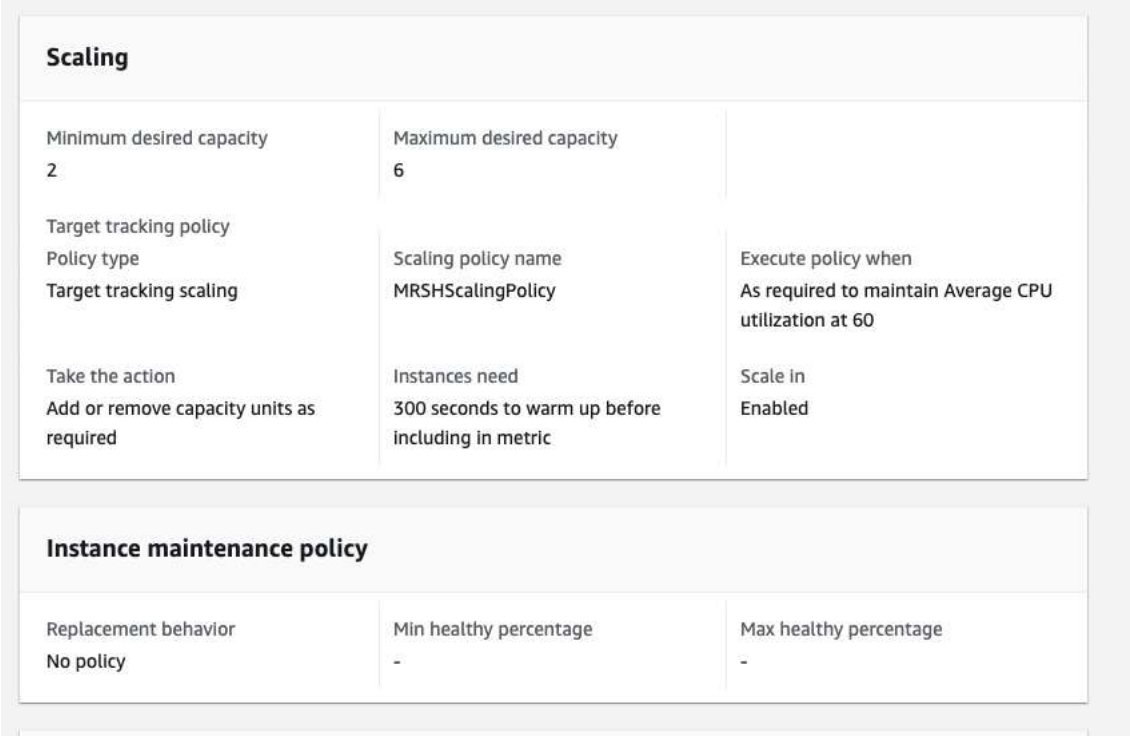


Figure 77: Target Group MRSHGroup1 Scaling

VPC Considerations

Sunday, 19 October 2025 10:35 am

VPC Security Considerations

In a cloud environment, multiple VPC security features should be used to protect our instances and databases from attacks and unauthorised access. Figure 78 illustrates some common VPC security features.

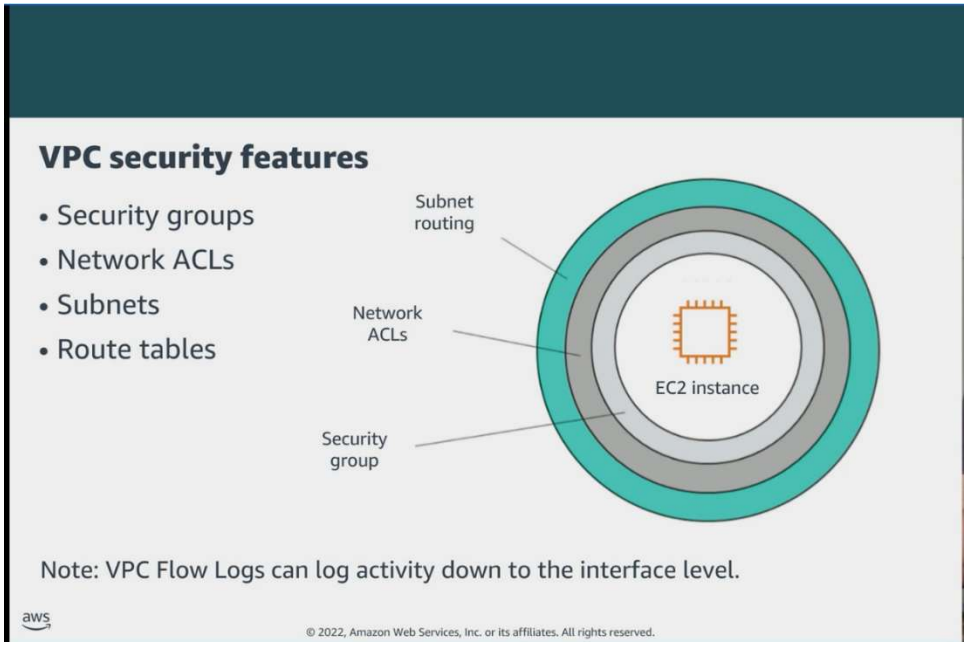


Figure 78: VPC security features

In the case of this AWS cloud environment, the following features were implemented:

Multiple Availability Zones (AZ): Subnets are spread across multiple AZs for high availability & durability, increased fault tolerance (in the event of an AZ/database instance failure) and scalability.

Use of private and public subnets: Private subnets enhance security as they are not directly exposed to the internet, thereby hindering potential unauthorised access attempts and/or cyber threats. Instances that do not need to be connected to the internet directly are placed in our private subnets. In this case, the DB Instance is located in the private subnets and connected to the EC2 web instance via security groups. All other access attempts and traffic are denied.

Security Groups: Security groups protect the instances by monitoring and controlling traffic in and out of it. The EC2 web server and RDS DB instances are assigned Security Groups that only allow the necessary traffic needed for their respective purpose.

Network ACLs: Network ACLs protect our subnets by controlling inbound and outbound traffic. The assigned ACL for our subnets allow certain inbound and outbound traffic, whilst denying all else.

CloudTrail and CloudWatch: These functions allow for monitoring and auditing of events and defined metrics within our cloudspace, sending out notification messages and activating alarms if need be, so that network administrators have high visibility of what is happening in the environment to take the necessary actions.

Other VPC Security Considerations

Due to either the account restrictions in place, or their unsuitability to this use case, the following are a few VPC security features that may be implemented in other cloud systems:

IAM: Due to account restrictions, I am unable to use IAM to create users, roles, and policies for restricted AWS service access within our VPCs.

Bastion Hosts: A bastion host withstands cybersecurity attacks and protects VPCs from security vulnerabilities and cybersecurity threats from the internet by filtering and directing user and network activity. Purposing an additional EC2 instance as a Bastion Host will incur costs and administrative duties and efforts.

Proxy Servers: Proxy servers are another layer of security as they act as a web filter for network users and the internal network. In our case, as we only have our web server on the cloud and they are usually placed on public subnets for direct access to the internet and lower costs than if they were to be placed in a private access (traffic through NAT gateway is charged).