# Building a Custom-Designed Social Networking Platform for UCW Students

## Phase 2: Advanced AWS Service Implementation

## Part 2

Aditya Prashant Arte (2305505)

Amandeep Singh (2231193)

Maduka Aravindi (2316990)

Obioma Joseph Nwagwu (2313367)

Rajani Mishra (2303650)

MBA, University Canada West

BUSI 653: Cloud Computing Technologies (HBD-WINTER25-02)

Professor: Sarah Gholibeigian

Due Date: 21st March 2025

## 1.Implementing Load Balancers: To improve fault tolerance and increase application availability.

### Introduction

The implementation of load balancing in cloud computing serves as a primary mechanism to achieve applications that demonstrate higher reliability together with fault tolerance and availability. During this project stage we deploy AWS Elastic Load Balancer (ELB) to spread incoming traffic across multiple Amazon EC2 instances for ensuring high availability and maximizing resource usage.

### Configuration and Integration of AWS Elastic Load Balancer (ELB)

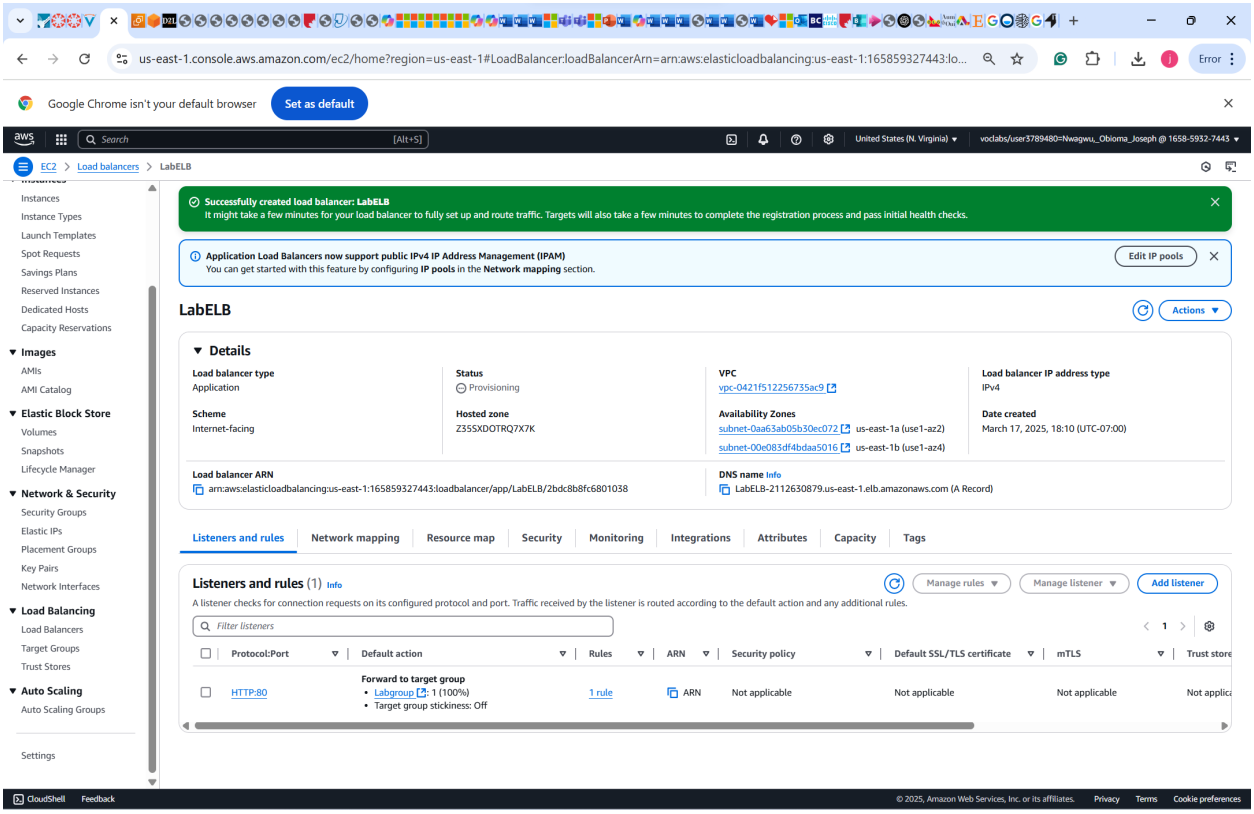
- The process to create an Elastic Load Balancer (ELB) begins at the navigation menu of the AWS Management Console > EC2 Dashboard.

- Access the AWS Management Console and select the EC2 dashboard from the navigation options.
- Starting the ELB creation process begins with selecting Load Balancers followed by pressing Create Load Balancer.

- You should choose Application Load Balancer (ALB) for handling HTTP/HTTPS traffic yet Network Load Balancer (NLB) serves TCP/UDP traffic requirements.

- Users must define the following parameters for the load balancer: name and VPC as well as scheme (whether internet or internal).

- Generate the listener configuration by specifying HTTP on port 80 and HTTPS on port 443.

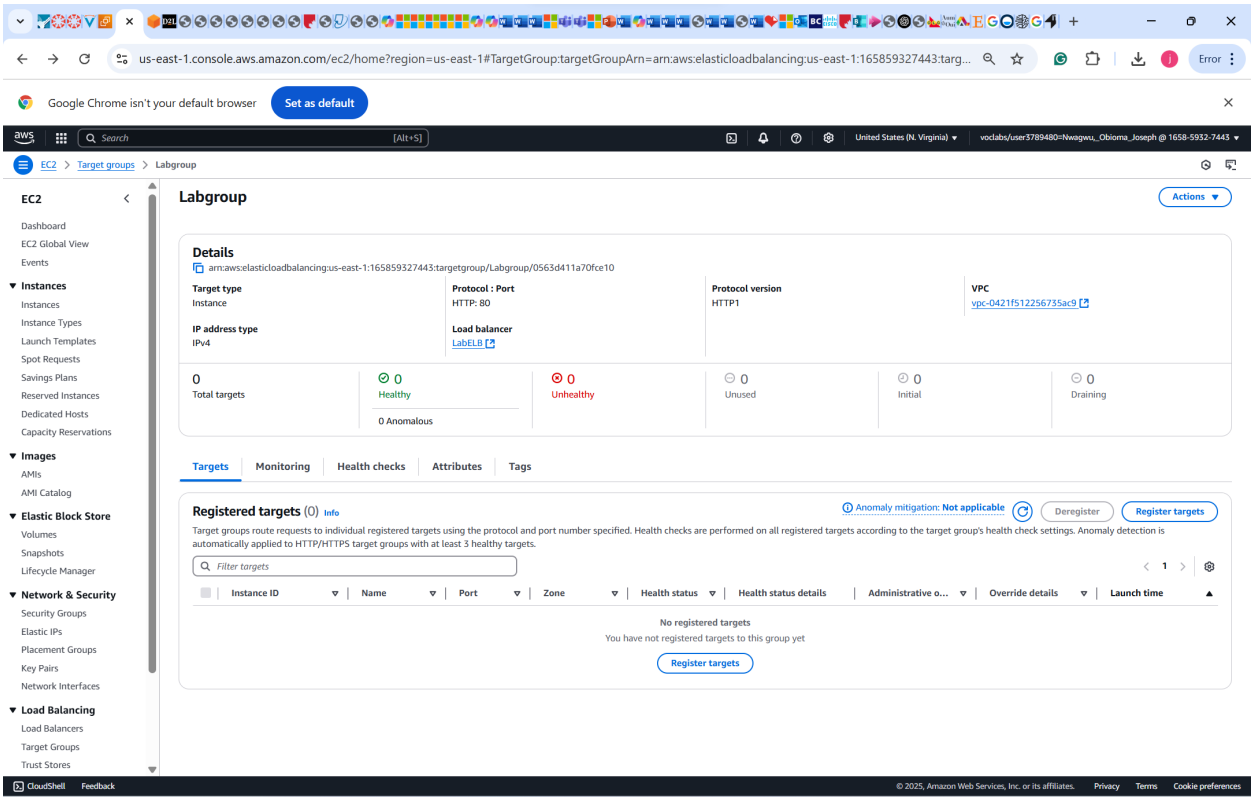



## Configuring Target Groups:

- Navigate to Create a Target Group then choose the EC2 instances through which traffic will pass.

- Set the health check parameters through which the system monitors instance status by using the specific path /health.

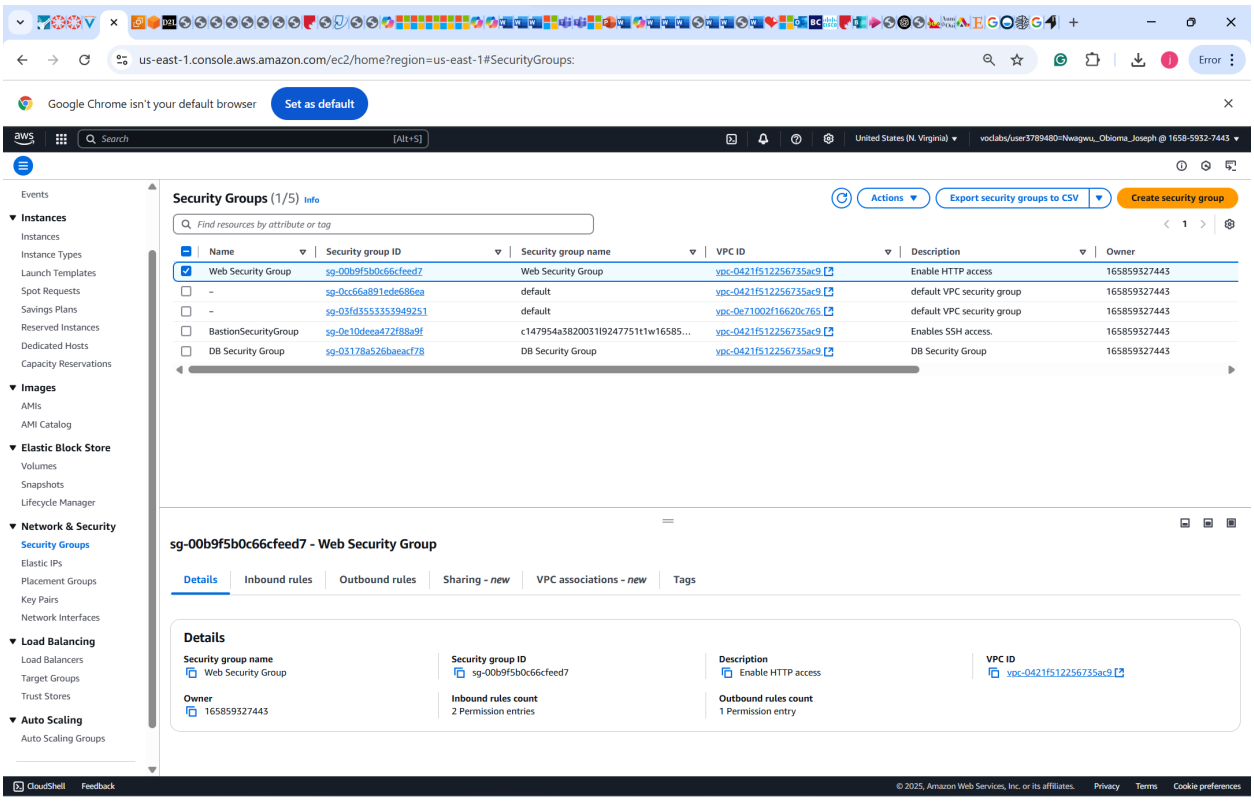
- 
- Move the target group into a connection with the load balancer.




**Security Groups require modification alongside the configuration of IAM roles:**

- Security groups need modification to enable traffic between the ELB and EC2 instances.

- The IAM roles must include permission settings for both load balancing and auto-scaling operations.




## Testing the Load Balancer:

- The DNS name of the ELB can be retrieved from the AWS console interface.

- You should utilize either a command line interface or a browser to make requests through the DNS endpoint.
- Validate traffic distribution among instances.

## Impact of Load Balancers on Performance and Availability

## Improved Fault Tolerance:

- ELB routes traffic away from unwell EC2 instances to available healthy ones.

- The application design benefits from increased fault tolerance through its architecture (AWS, 2023).

**Scalability and Cost Efficiency:**

- The system integrates perfectly with AWS Auto Scaling to allow automatic adjustment of resources according to changing requirements.
- The system optimizes resource consumption results in decreased operational expenses (Smith et al., 2023).

**Enhanced Security and Compliance:**

- Supports TLS termination to offload SSL encryption from EC2 instances.
- AWS Shield along with the service delivers DDoS protection (Jones & Miller, 2023).

| **Challenges** | **Solutions** |
|---|---|
| Incorrect security group rules | Ensured correct inbound/outbound rules for ELB and EC2 instances. |
| Uneven traffic distribution | The Target Group settings received adjustments to optimize routing functionality. |
| Latency issues | Caching of static content occurred through the AWS CloudFront system. |

The AWS Elastic Load Balancer proves essential for delivering high availability together with security and scalability of cloud-based applications. The traffic distribution capacity of ELB prevents network bottlenecks which results in an improved user experience of UCW social networking users.

## 2.Implementing Autoscaling

As a social media platform, it is expected that the users will create a varying traffic. Notably, there will be a specific time of the day, moreso in the evening after classes, when the website with experience more traffic. In contrast, the traffic is expected to be low during the day when UCW students will be attending classes. Therefore, autoscaling will be necessary to help the platform to adjust resource utilization and performance depending on the load. In this regard, auto scaling will be a crucial feature that will ensure that the platform maintains performance while optimizing costs during varying load conditions (Nguyen et al., 2020). AWS autoscaling optimizes performance of the social media platform because it will automatically manage the number of demanding instances. According to Nguyen et al., (2020), this functionality helps in addressing resource over-provisioning. Autoscaling enables an application to operate optimally regardless of the changes in traffic patterns. It also reduces unnecessary costs attributable to unadjusted resource de-provisioning during times of low traffic volume (Nousiainen, 2021; Nguyen et al., 2020). In the current project, the process of implementing autoscaling in AWS will entail three major components: The first one is creating an autoscaling group; the second one is setting up scaling policies, and final one is monitoring and optimizing.

### Creating an Auto Scaling Group

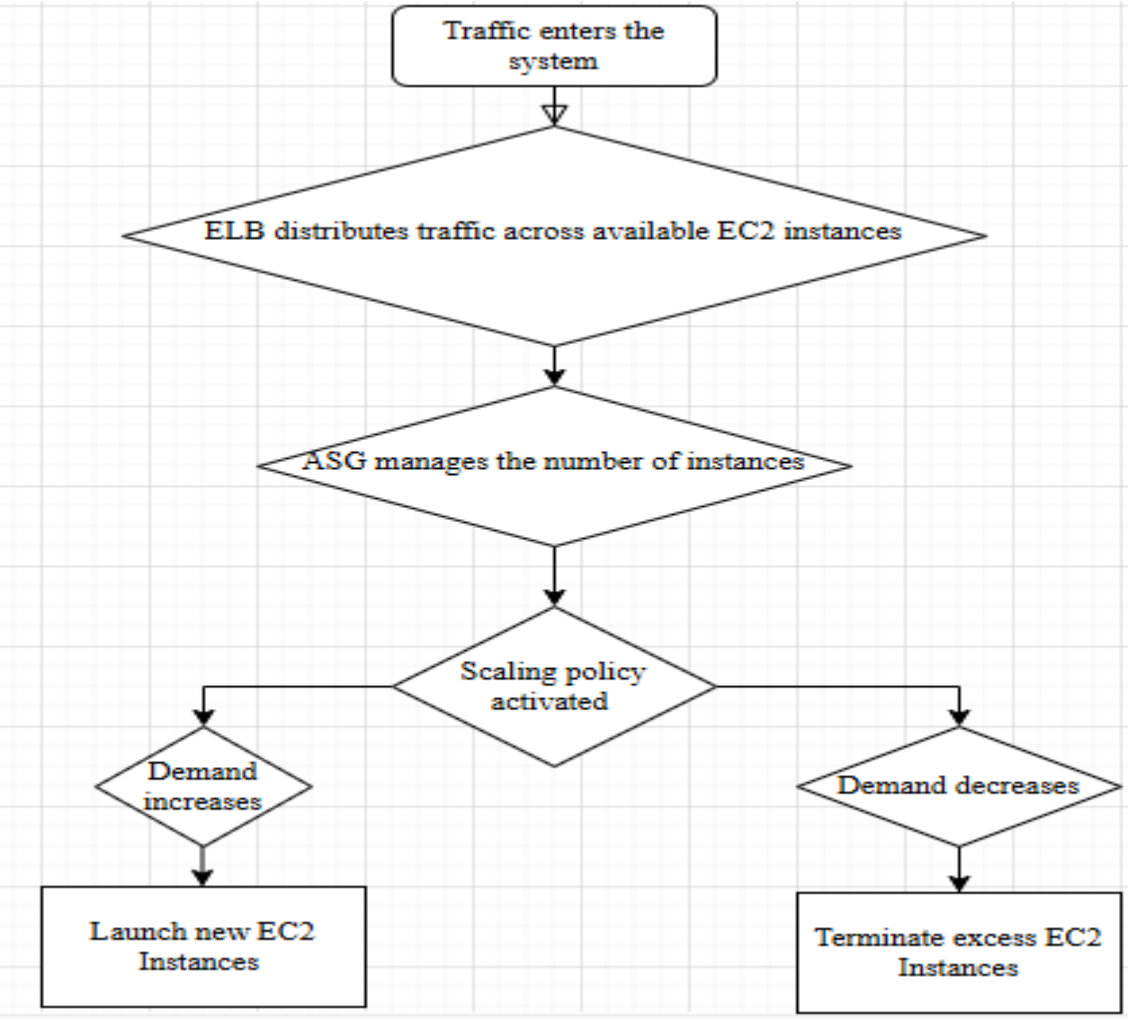
An Autoscaling Group (ASG) is fundamental in dynamically controlling and managing resources (Nguyen et al., 2020). When developing the social media site, the process of creating an autoscaling group will commence with creating a launch template. The launch template will configure the AMI and instance type. Notably, these are the prerequisites for establishing security groups for controlling traffic load. Furthermore, it will include an automated startup script to provision new instances automatically (Catillo et al., 2020). Establishing the launch

template for the social media platform will set the stage for creating the autoscaling group (ASG) with a specified minimal and maximal instance number. The instances will then be placed in multiple Availability Zones (AZs) to increase reliability and fault tolerance. Finally, the autoscaling group will be connected to an Elastic Load Balancer (ELB) to optimize load balancing in varied conditions by evenly distributing traffic.

**Setting Up Scaling Policies**

To sustain resources optimally, Alharthi et al. (2024) argue that scaling policies need to be created within the Auto Scaling Group. In the current project, real-time metrics will be set to determine the number of instances that are provisioned and de-provisioned. For example, an additional instance will be launched if CPU utilization exceeds 80% for three minutes. This will aim at handling the increased demand. In contrast, when CPU utilization drops below 20% for five minutes, excess instances will be terminated to optimize costs. The configuration will be designed in such a way that it follows the traffic patterns. In this case, scheduled scaling will be applied. Through monitoring usage trends, administrators will be able to define scaling actions such as increasing instances during peak hours in the university and decreasing them during low activity periods to avoid resource wastage.

Figure: How Autoscaling will function for the social media platform




*Note*. A schematic representation of how autoscaling will work for the social media platform. Own work.

**Monitoring and Optimization**

AWS CloudWatch will be utilized for monitoring to ensure efficient auto scaling. In practice, CloudWatch monitors critical indicators like CPU usage, network activity, and response

times (Nousiainen, 2021). With the use of alarms, administrators will automate scaling actions and react to changes in demand in a timely manner.

Concerning cost management. The combination of On-Demand and Reserved Instances will provide flexibility while controlling costs (Nousiainen, 2021). Non-critical workloads will further facilitate cost-cutting through the use of Spot Instances. Additionally, selecting a custom will prevent unnecessary spending while maintaining optimal performance.

**Impact on Performance and Cost**

Autoscaling significantly enhances the performance and cost-effectiveness of the social networking platform (Verma & Bala, 2021). The social media platform will be also responsive during peak usage times because of the autoscaling functionality. When autoscaling will be implemented to the custom social media platform, the users will experience minimal latency and downtime, ensuring constant interactions and engagements with the users.

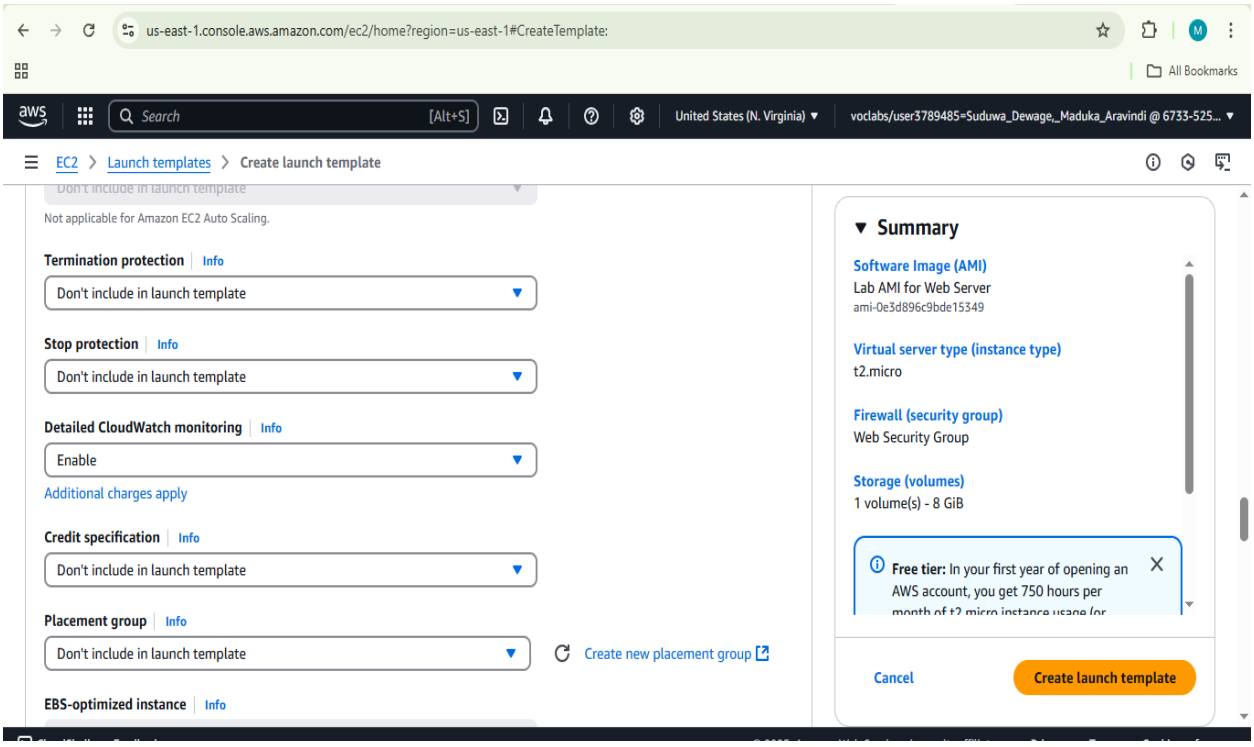
Similarly, autoscaling addresses the cost issue by stopping over-provisioning of instances because there will only be as many instances running as the demand requires. This will reduce infrastructure costs while maintaining the required performance level for the social media platform (Fe et al., 2022). Moreover, autoscaling will improve reliability by replacing unhealthy instances automatically, and distributing workloads effectively to prevent bottlenecks.

## 3. Monitoring with Amazon CloudWatch Implementation

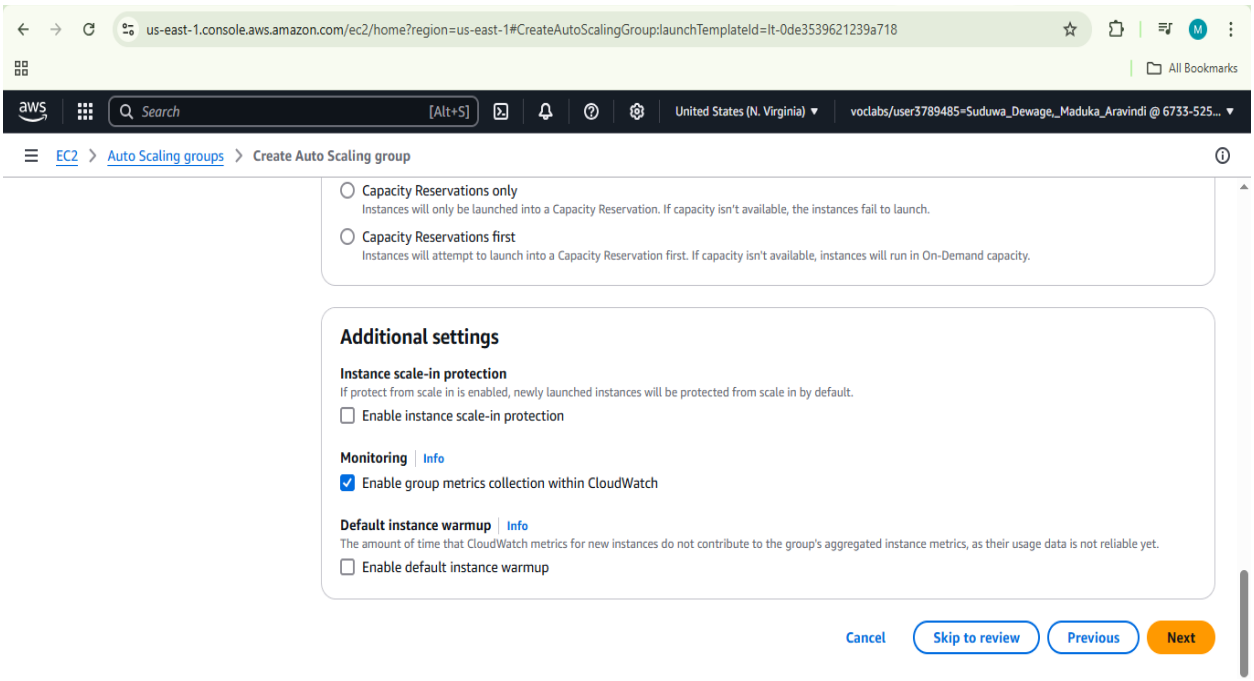
It is important to maintain reliability, security and performance as UCW digital platform is used to keep the real time connections and resources sharing among students and faculty. To accomplish these objectives, Continuous monitoring using Amazon CloudWatch throughout the system is crucial as it provides information on the system health, utilization and the issues occurring.

Continuous monitoring is more crucial in the peak seasons like the end of the semester to avoid any service interruptions since the load and usage is higher than usual.

### 3.1. Configuring CloudWatch Monitoring in the Launch Template and Auto Scaling Group




This enabling detailed CloudWatch monitoring when creating the launch templates for the EC2 instances provides necessary data on the performance such as CPU utilization and internet traffic to CloudWatch every minute. This helps to identify the issues related to the UCW digital platform application and helps to trigger the auto scaling with the changing utilization.



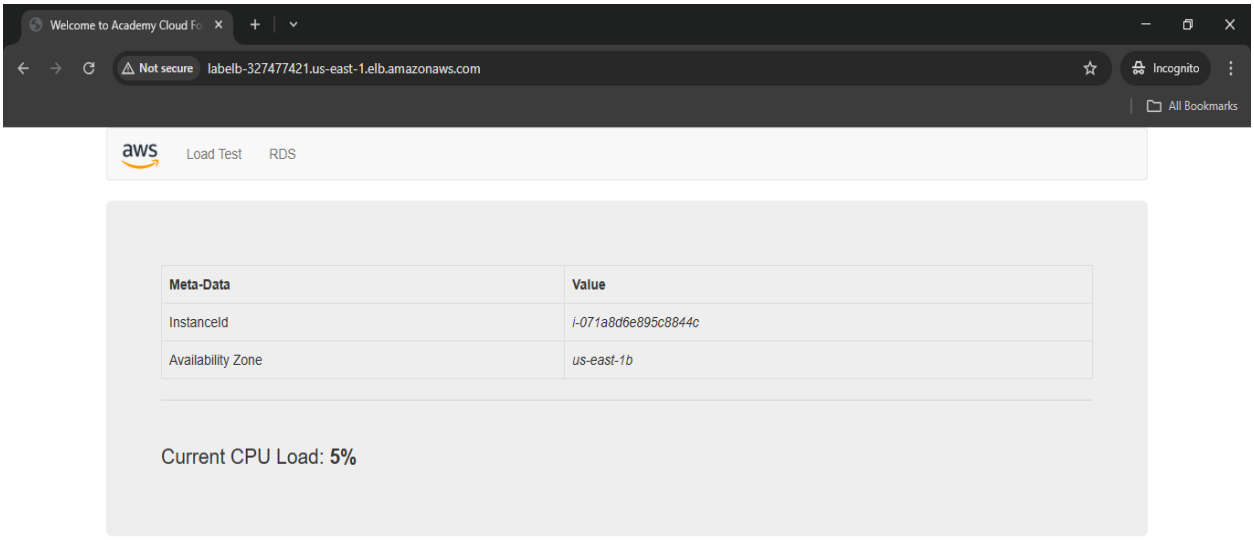

By enabling group metrics collection within CloudWatch, it allows auto scaling groups to adjust the running EC2 instances and react quickly to changing utilization patterns. This captures metrics at 1-minute intervals.

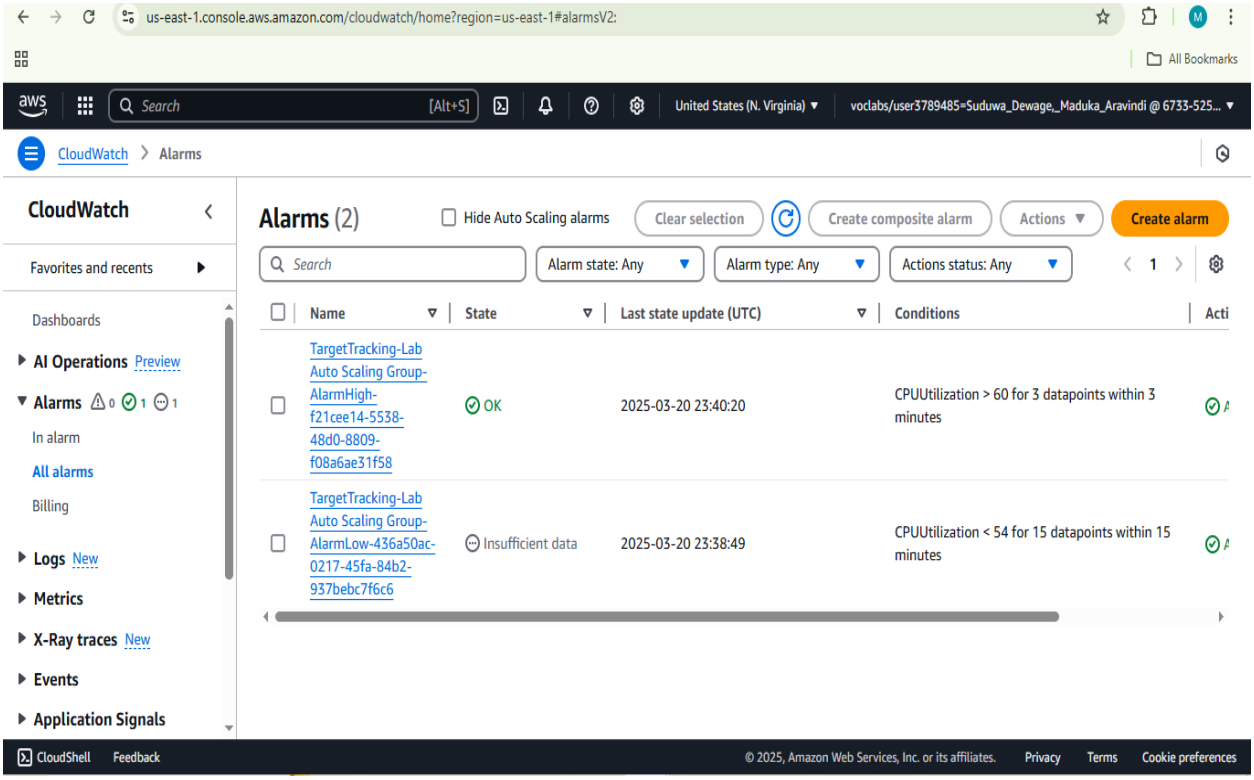
### 3.2. CPU Utilization Monitoring

CloudWatch is monitoring the CPU utilization, and it triggers autoscaling policy to launch additional instances. In a situation where the CPU is utilized in the highest capacity this monitoring is crucial to maintain the UCW digital platform without any interruption or downtime.

Alarms were configured and created automatically by the auto scaling group. According to the configuration, it automatically keeps the average CPU load close to 60% and within the limitation of having two to six instances. Also, if the CPU utilization is lower than 45% within 15 minutes, CloudWatch alarm triggers the autoscaling policies to terminate additional EC2 instances.

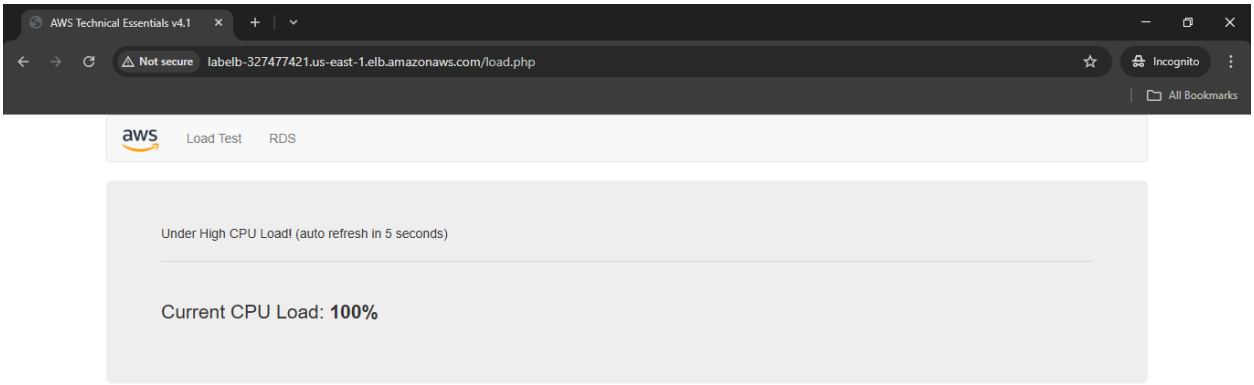
## Initial configurations before the CPU high utilization

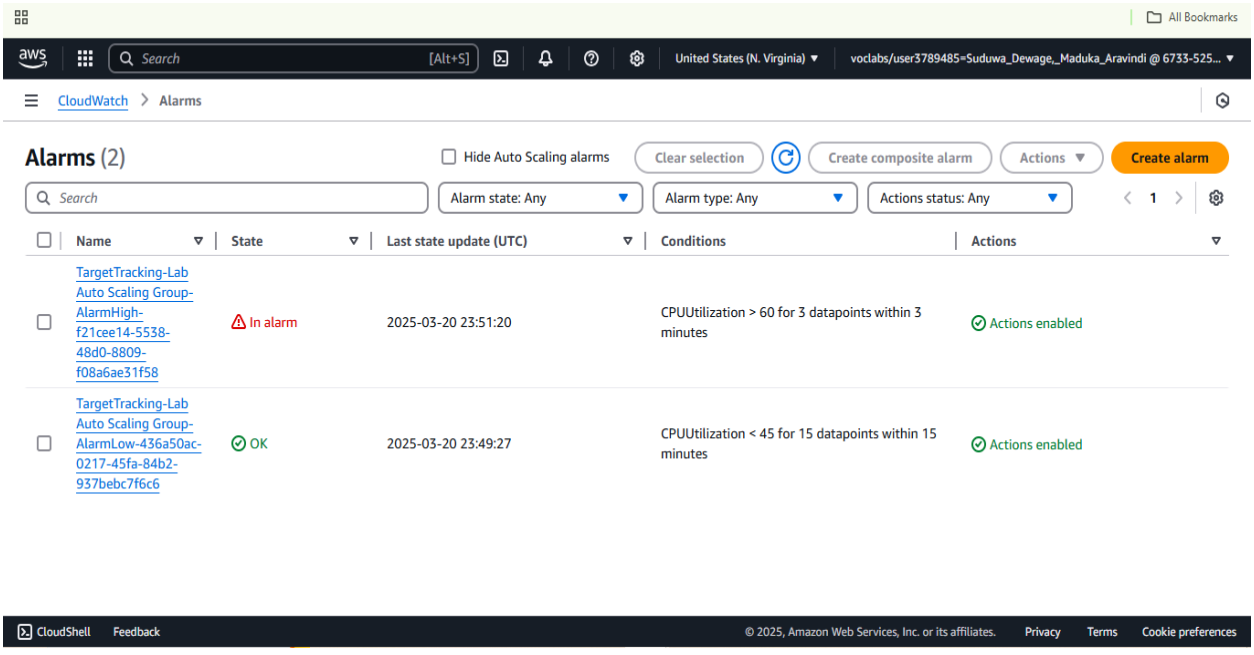


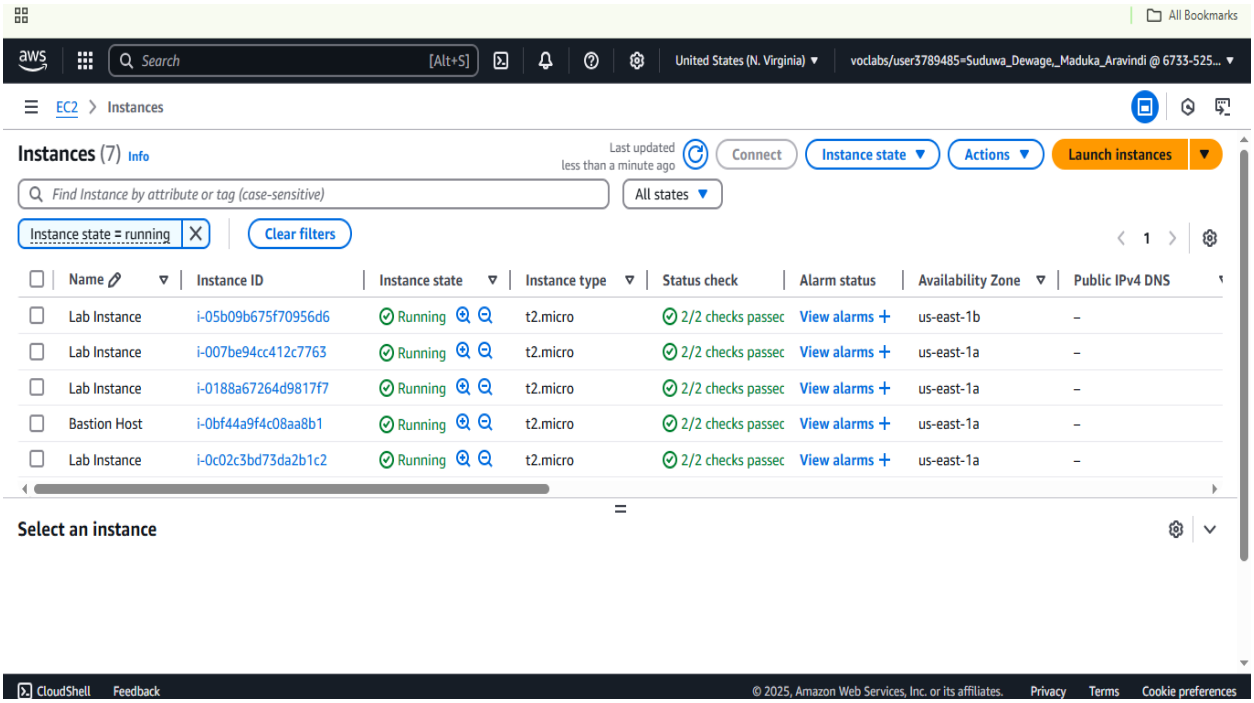
**Impact of CPU Utilization exceeds the limit of 60%**

Once the CPU utilization exceeds the limit of 60% CloudWatch alarm is in use, and it triggers the autoscaling policy to add additional EC2 instances to cope with the higher utilization in order to manage the load.



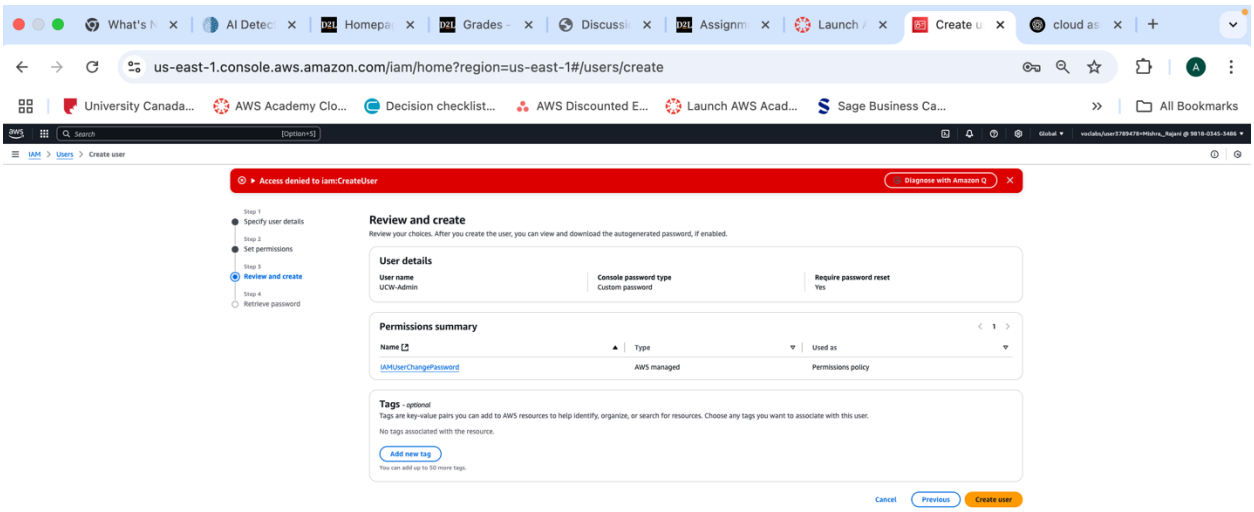
To cope with this high CPU utilization, autoscaling actions have been added three new instances to the system.




Monitoring using the CloudWatch in Auto Scaling is crucial in managing the over load in peak season and for cost optimization when the load is not too high.

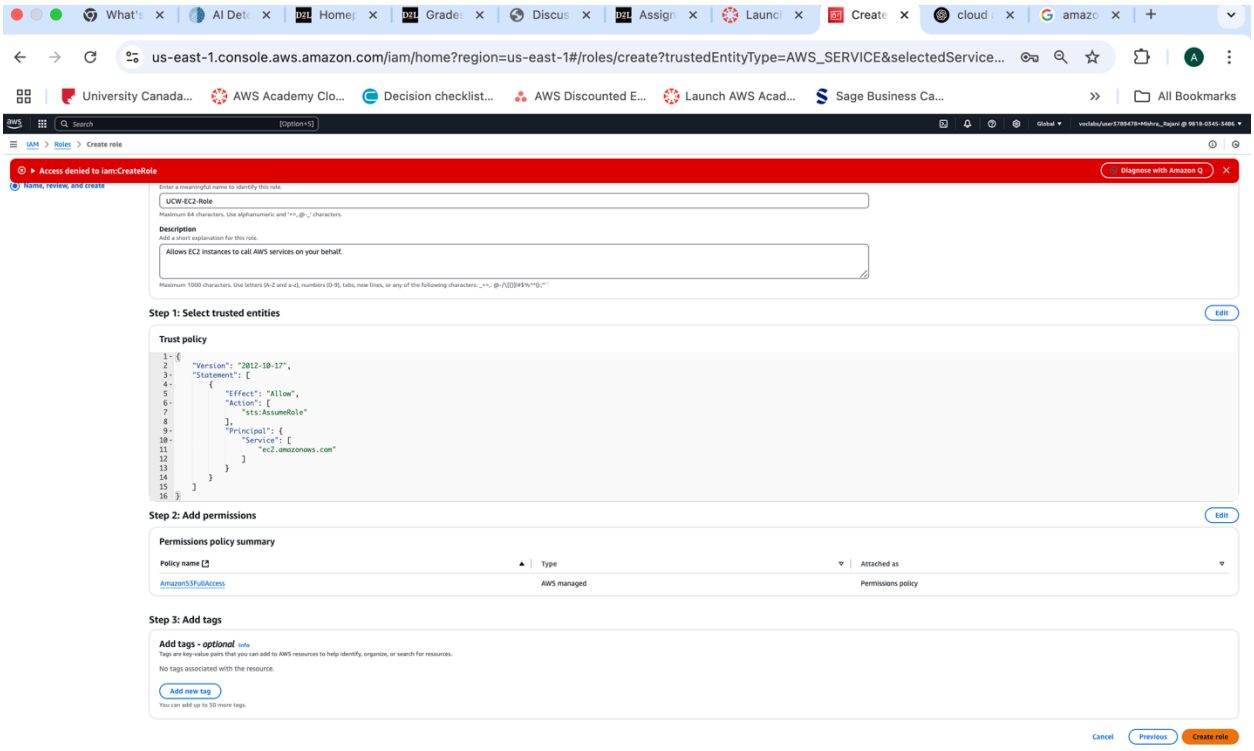
## 4. Implementing AWS Identity and Access Management (IAM)

### IAM User Creation



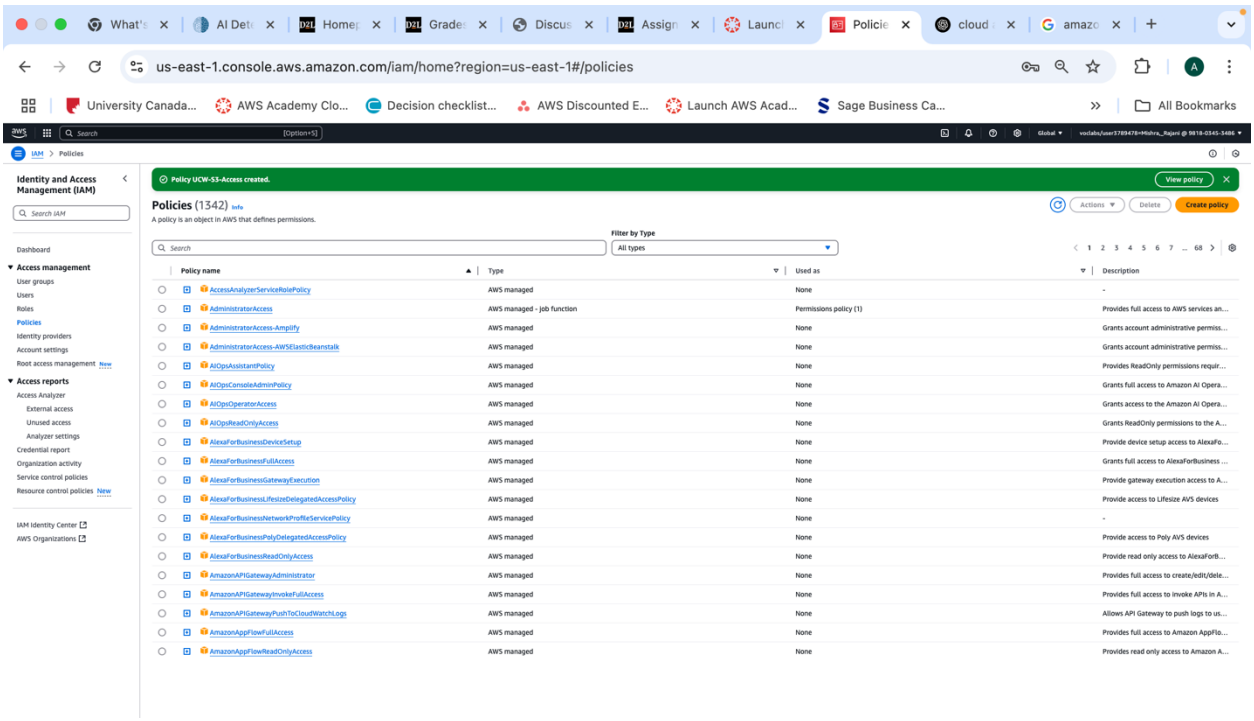

Here we can observe an unsuccessful attempt to establish an IAM user named UCW-Admin based on this screenshot. The Access denied to iam:CreateUser error notification reveals that the AWS role for this lab lacks the appropriate permissions for new IAM user creation. IAM setup requires permissions that enable user management functions to be established effectively. This access denial occurred because the system lacked an IAM policy which granted "iam:CreateUser" privileges. Lack of user-based access controls became inevitable because this access control method plays a key role in securing AWS environments.

**IAM Group Creation**



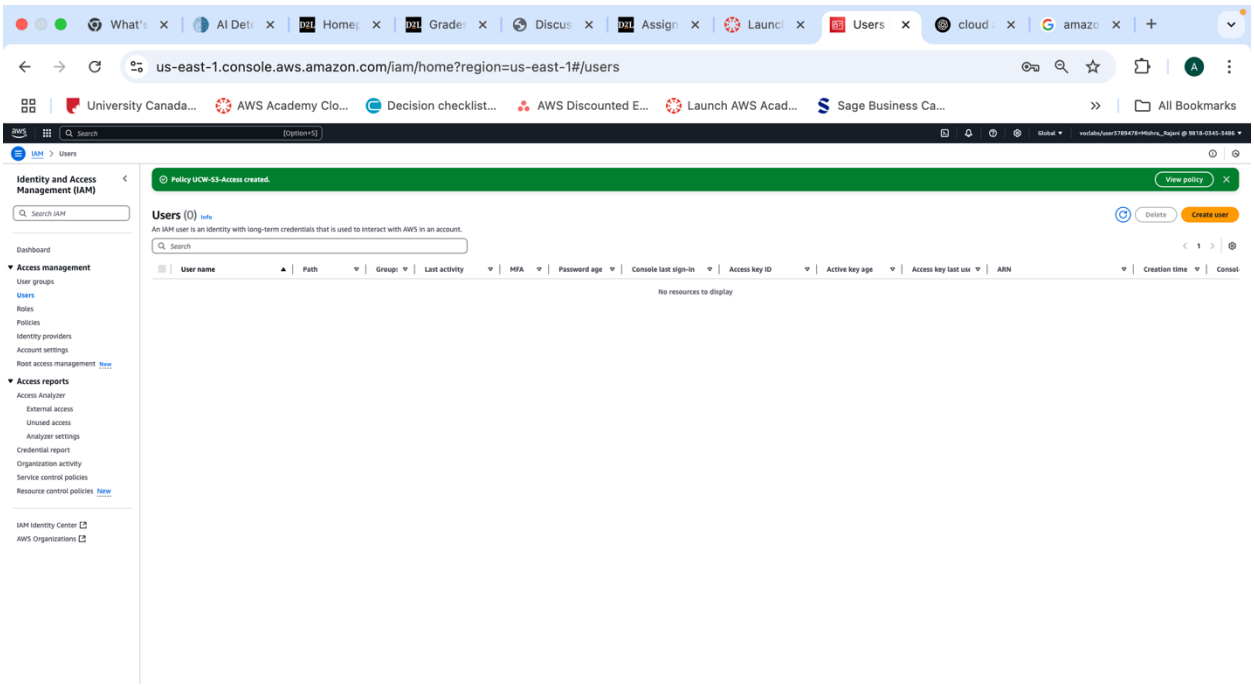

In this screenshot, we can see a visual depiction of an unauthorized attempt to establish an IAM group under the name UCW-Developers. The action failed while creating an IAM group named UCW-Developers because the system denied access to iam:CreateGroup permissions. AWS best practice suggests building IAM groups to help users with parallel roles obtain default access permissions more easily. The failure reveals an insufficient "iam:CreateGroup" permission within the current AWS role since this permission enables the definition of user groups. The absence of proper security policies for grouping users means such access management becomes less efficient.

**IAM Role Creation**



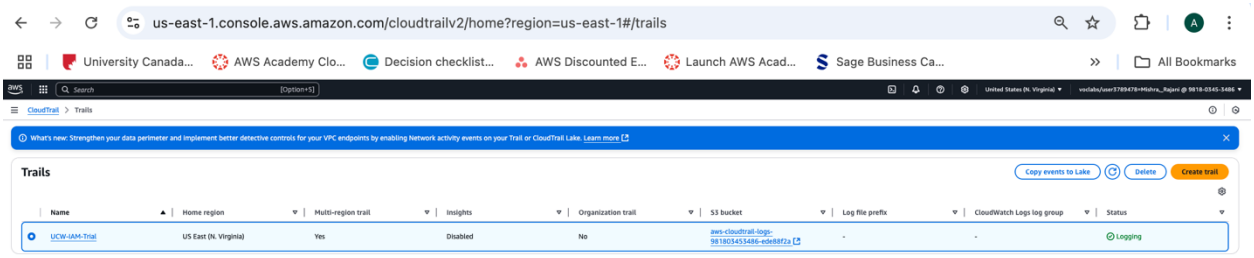

This screenshot captures the failure in creating the IAM role named UCW-EC2-Role. AWS services including EC2 instances need IAM roles to establish secure access to other AWS resources because embedding credentials would be insecure. The "Access denied to iam:CreateRole" error resulted in this failure since the present IAM role does not possess role management privileges. Roles cannot be assigned to AWS services because this security feature remains disabled in cloud-based applications. Database security is compromised when AWS services depend on access keys because these keys present a vulnerability if they become known.

## IAM Policy Creation Success




In this screenshot we can see that an IAM policy named UCW-S3-Access succeeded during creation. IAM policies determine the exact permissions which can be distributed to users and groups and roles. The policy allowed an Amazon S3 bucket access along with read and write permissions for its objects. The successful creation of the policy failed to apply because IAM users, groups, and roles were unavailable because of permission constraints. The principle of least privilege depends on IAM policies to secure the access of AWS resources. The successful policy creation faced an obstacle because attaching it to a role or user was not feasible given the absence of access to IAM management permissions during this session.

**AWS CloudTrail Setup Success**




This screenshot demonstrates how UCW-IAM-Trail serves as the successful implementation of AWS CloudTrail logging. AWS provides CloudTrail as a security and compliance tool which tracks all IAM activities so users can achieve audit trail capabilities for monitoring actions and detecting unauthorized access attempts. The successful implementation of CloudTrail allows security analysis to access all logged IAM events that contain information about failed user and role creation attempts. Cloud environments require the CloudTrail functionality for both security auditing and forensic investigations along with compliance purposes. CloudTrail setup proved successful because it does not require the same permissions as IAM user and group creation steps which led to its completion. S3 storage allows CloudTrail to store and analyze logs for improving security.

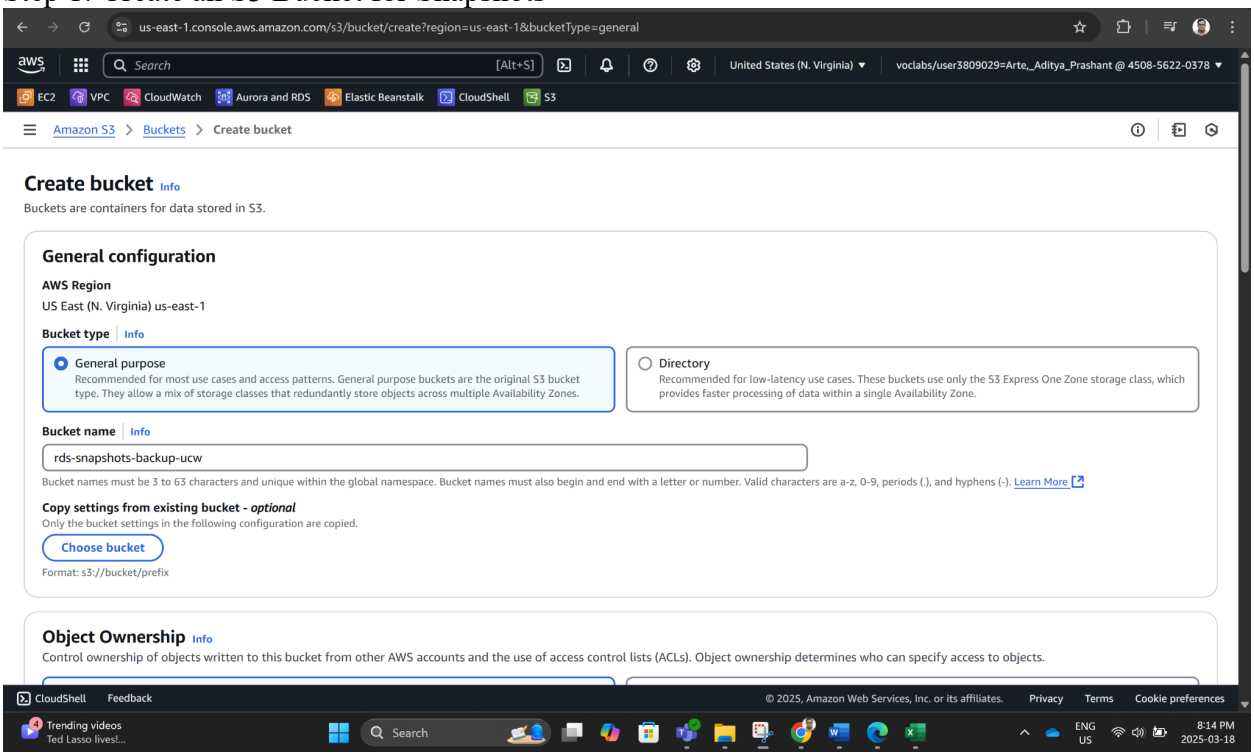
**Reflection on Challenges and Conclusion**

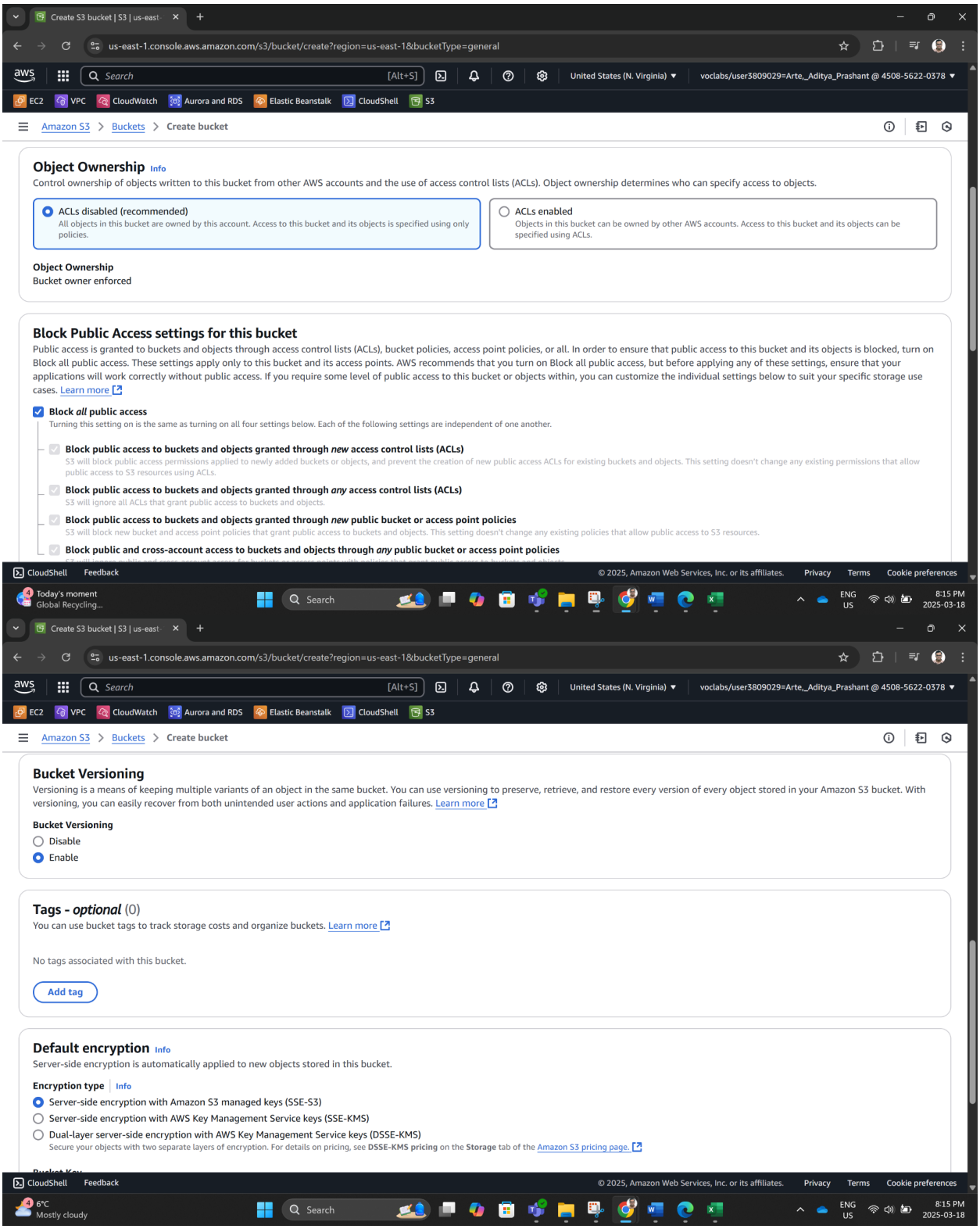
The AWS IAM lab revealed essential information about managing access together with security controls that exist in cloud-based environments. The restricted permission settings created difficulties during the exercise because they blocked the ability to make users and groups and roles. The inability to finish these assignments shows that proper IAM privilege distribution should always come first during administrative activities. AWS CloudTrail succeeded in implementing security monitoring functionality because IAM restrictions did not block this capability while providing visibility into unsuccessful and attempted events.

This lab proved that IAM policies are vital for AWS security management while offering practical CloudTrail logging experience for security auditing. Some constraints based on access limited behavior in the project but simultaneously demonstrated authentic enterprise cloud limitations which underline the value ofrole-based access systems in AWS deployments.

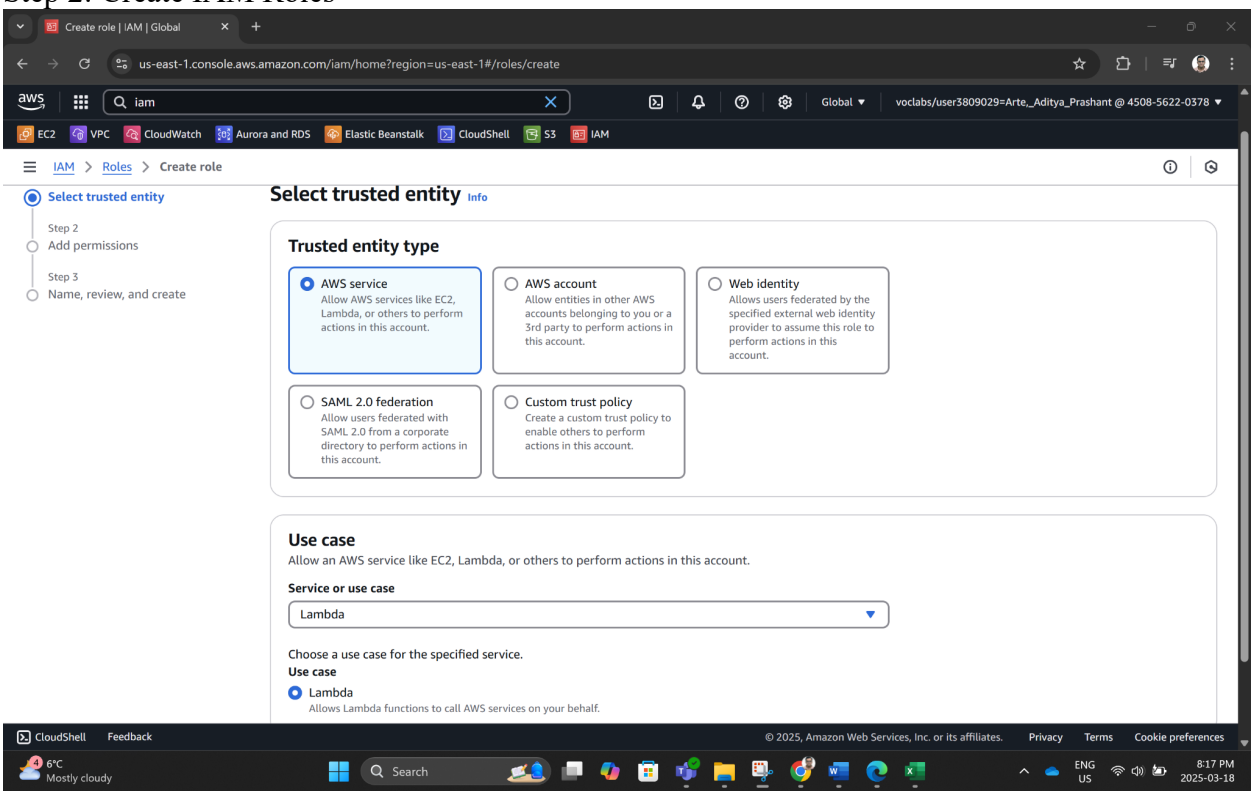
# 5. Creating a Specific Lambda Function

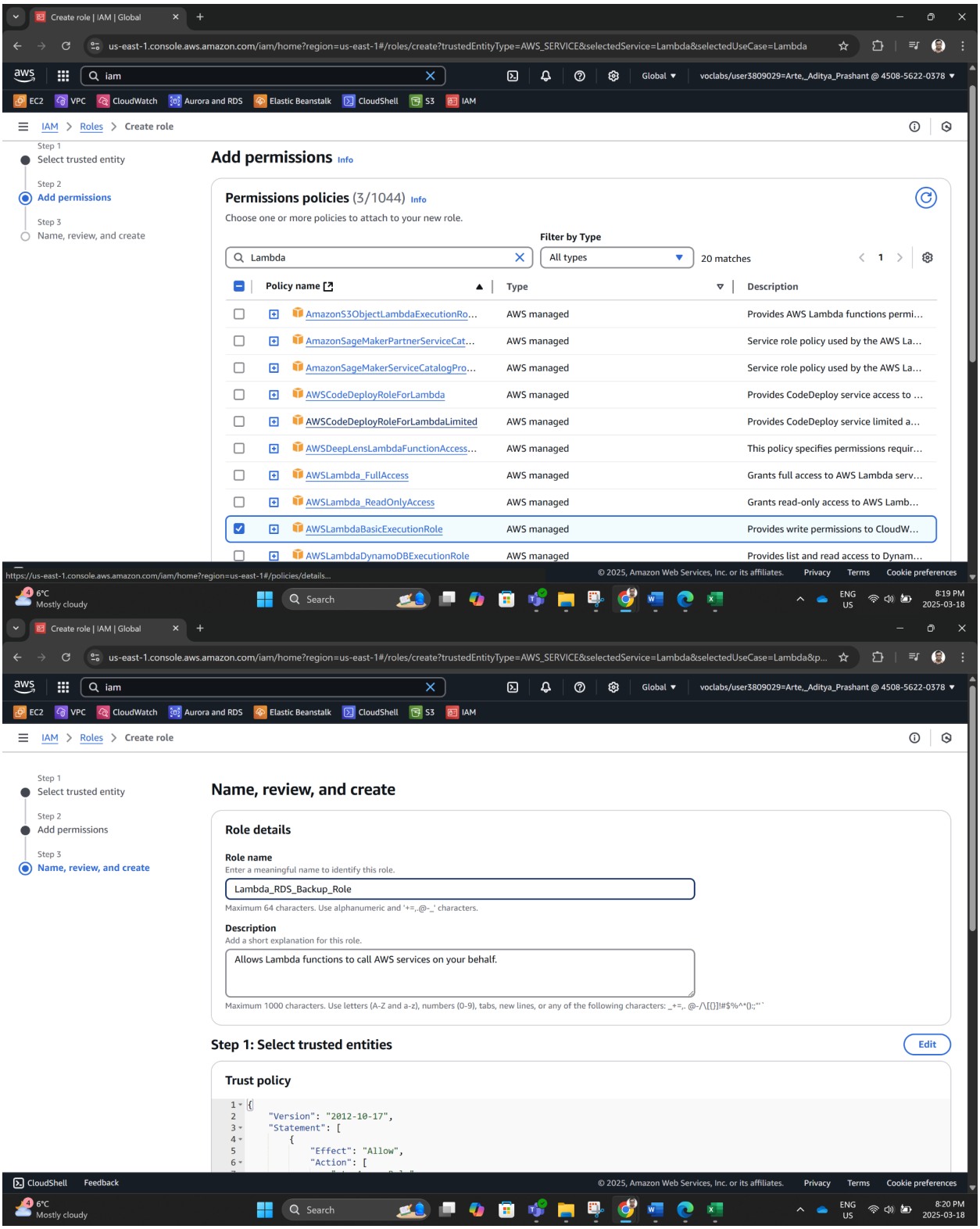
Step 1: Create an S3 Bucket for Snapshots



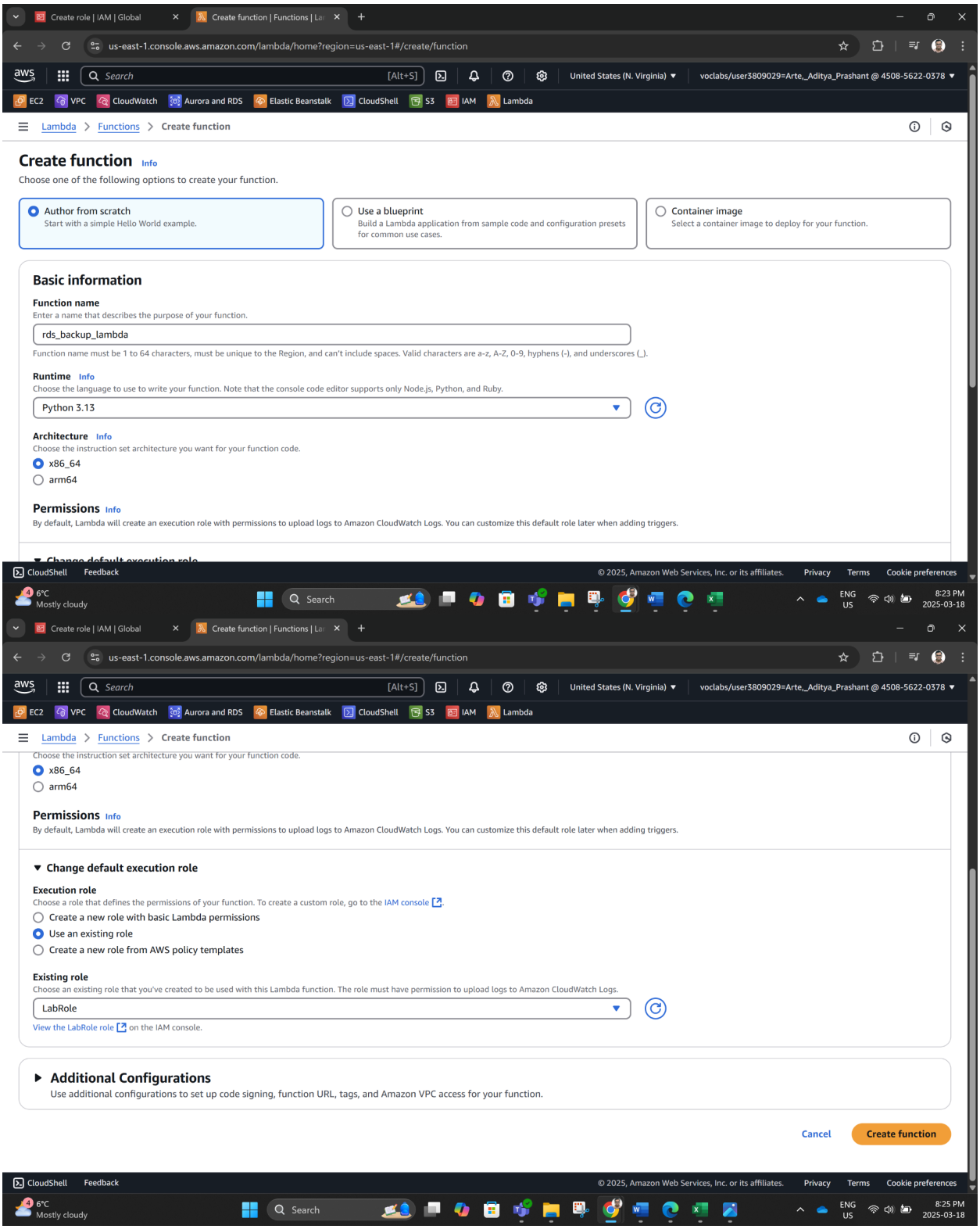

Create S3 bucket | S3 | us-east-
us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general
Search [Alt+S]
United States (N. Virginia)
voclabs/user3809029=Arte,_Aditya_Prashant @ 4508-5622-0378
EC2 VPC CloudWatch Aurora and RDS Elastic Beanstalk CloudShell S3
Amazon S3 > Buckets > Create bucket
Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.
Object Ownership
Bucket owner enforced
Block Public Access settings for this bucket
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more
Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
Block public access to buckets and objects granted through new access control lists (ACLs)
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy Terms Cookie preferences
Today's moment
Global Recycling...
Search
ENG US
8:15 PM
2025-03-18
Create S3 bucket | S3 | us-east-
us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general
Search [Alt+S]
United States (N. Virginia)
voclabs/user3809029=Arte,_Aditya_Prashant @ 4508-5622-0378
EC2 VPC CloudWatch Aurora and RDS Elastic Beanstalk CloudShell S3
Amazon S3 > Buckets > Create bucket
Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more
Bucket Versioning
Disable
Enable
Tags - optional (0)
You can use bucket tags to track storage costs and organize buckets. Learn more
No tags associated with this bucket.
Add tag
Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.
Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the Amazon S3 pricing page.
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy Terms Cookie preferences
6°C
Mostly cloudy
Search
ENG US
8:15 PM
2025-03-18

Step 2: Create IAM Roles

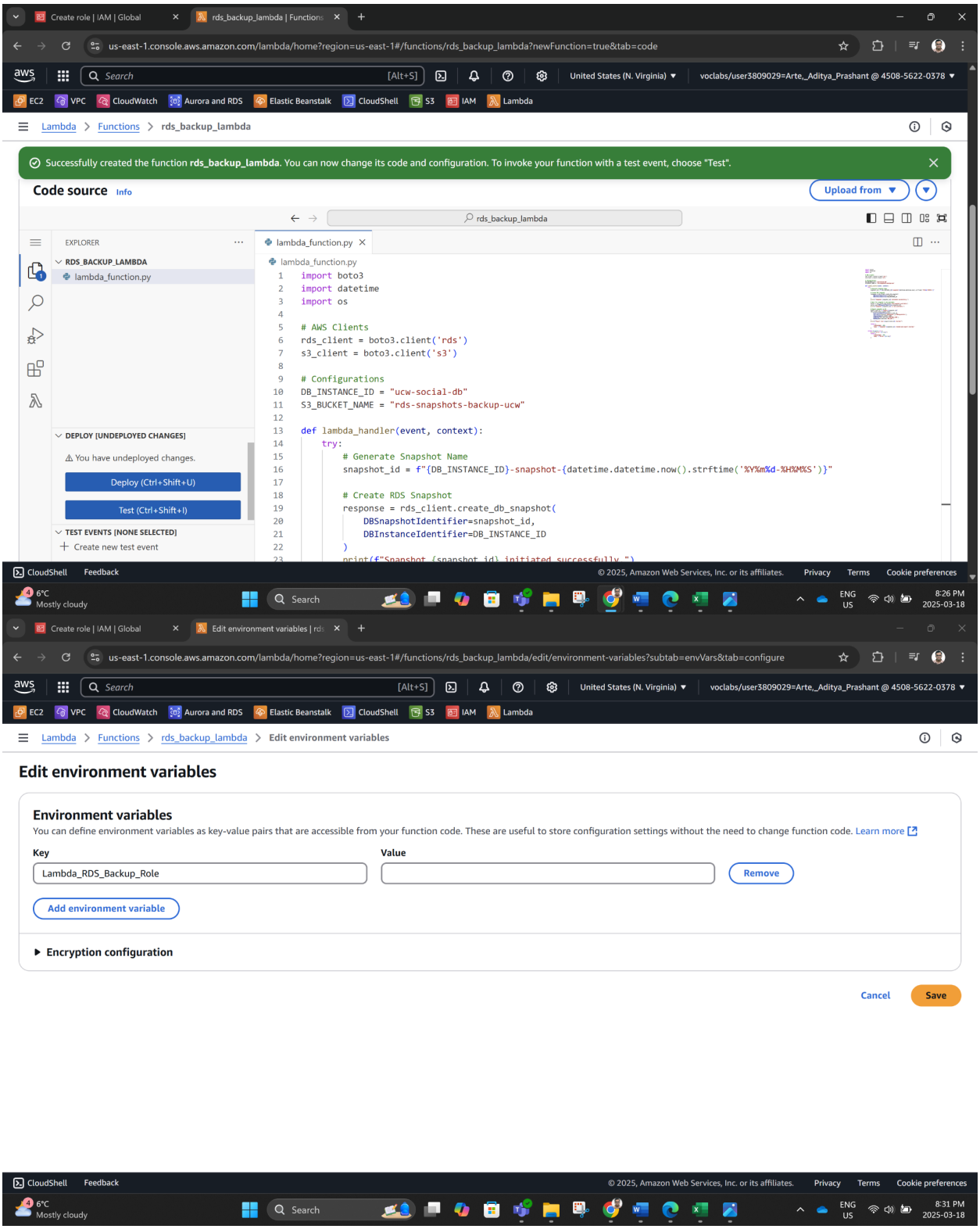




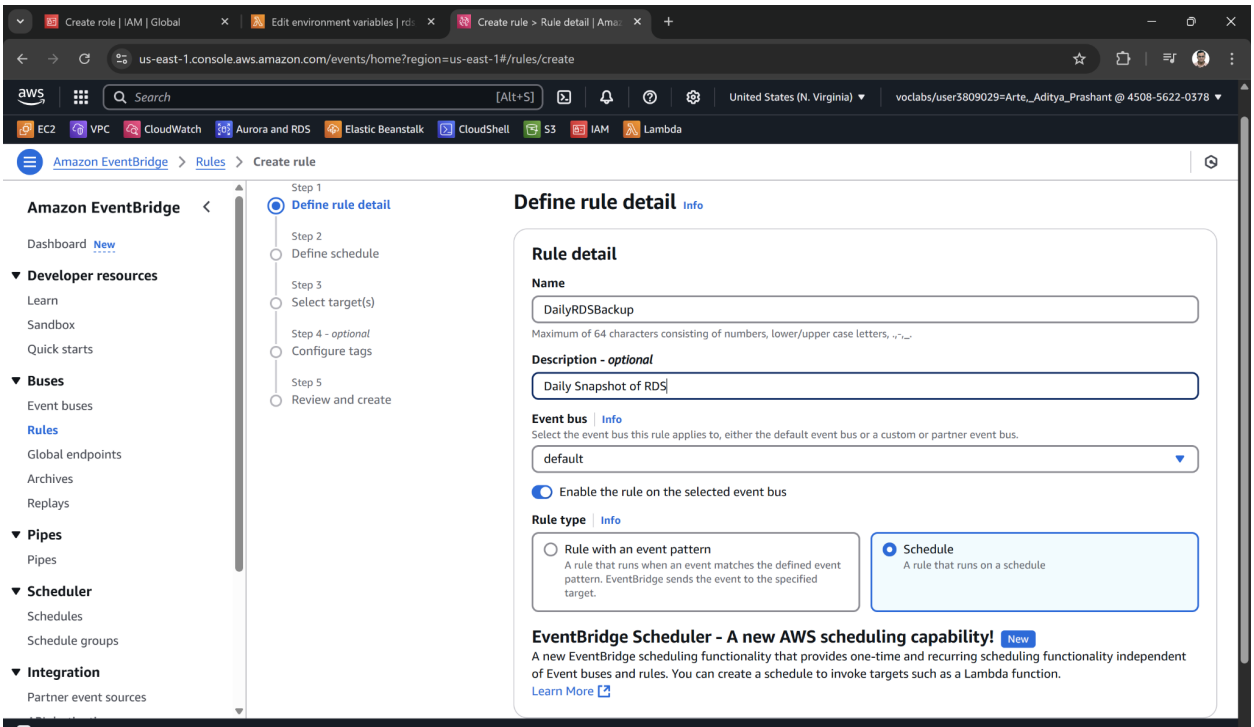

Step 3: Create a Lambda Function

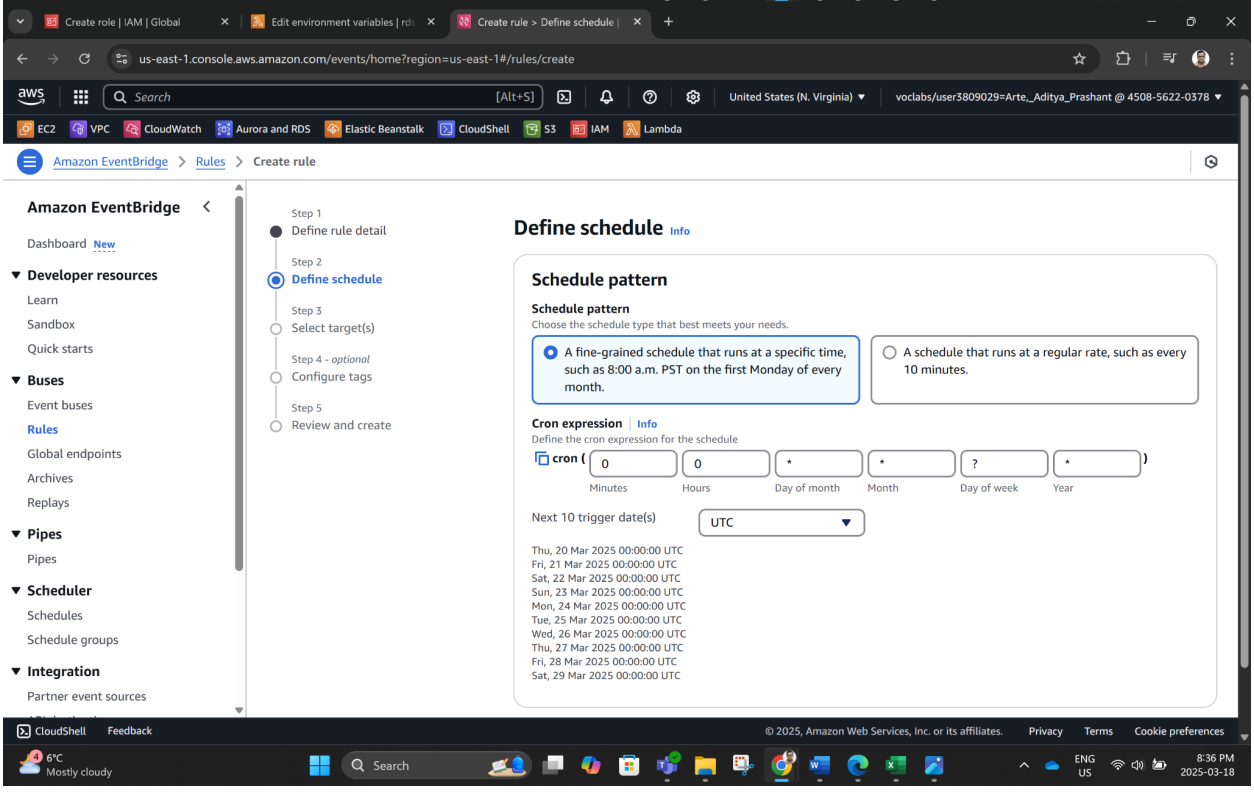


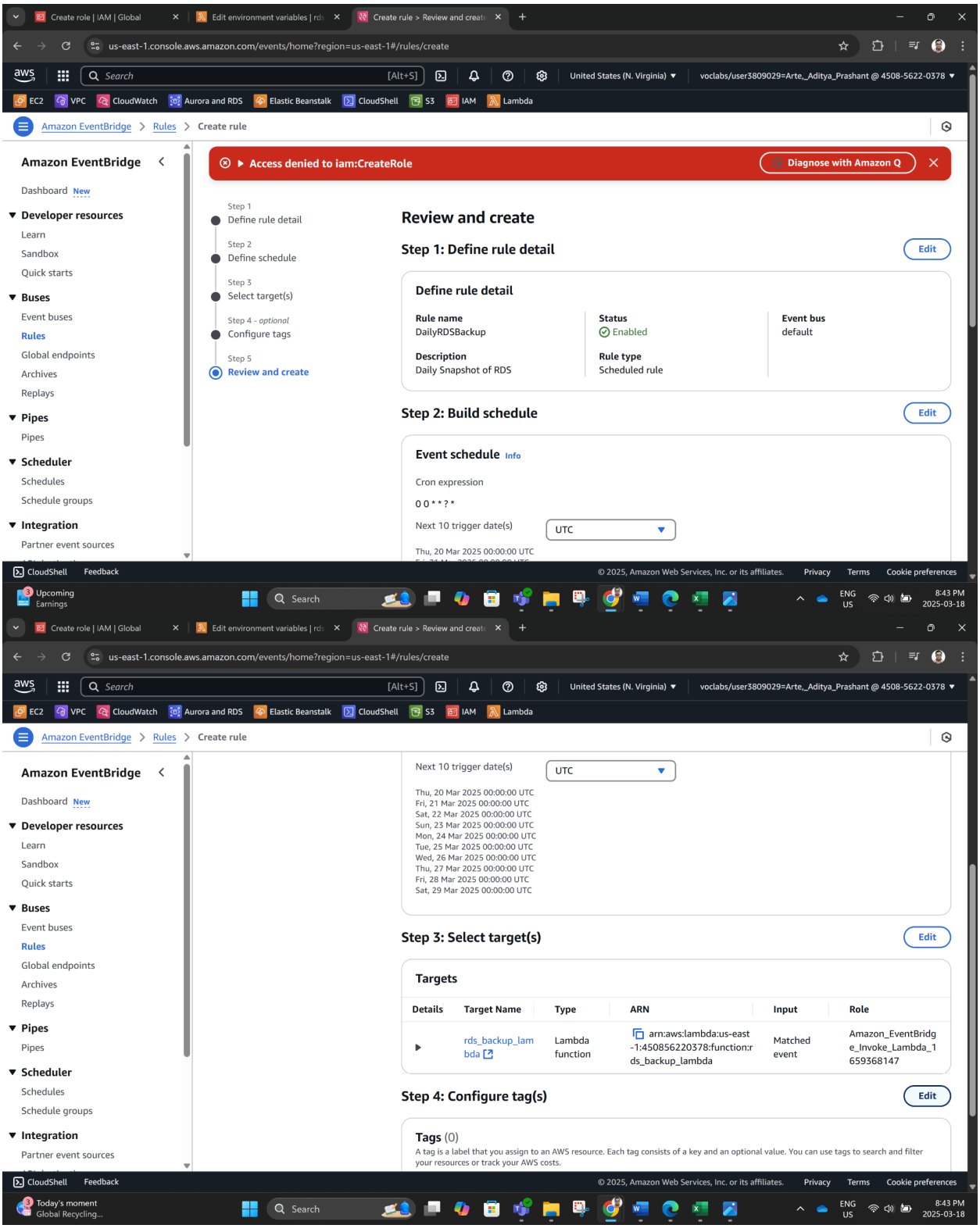

Step 4: Add Lambda Code

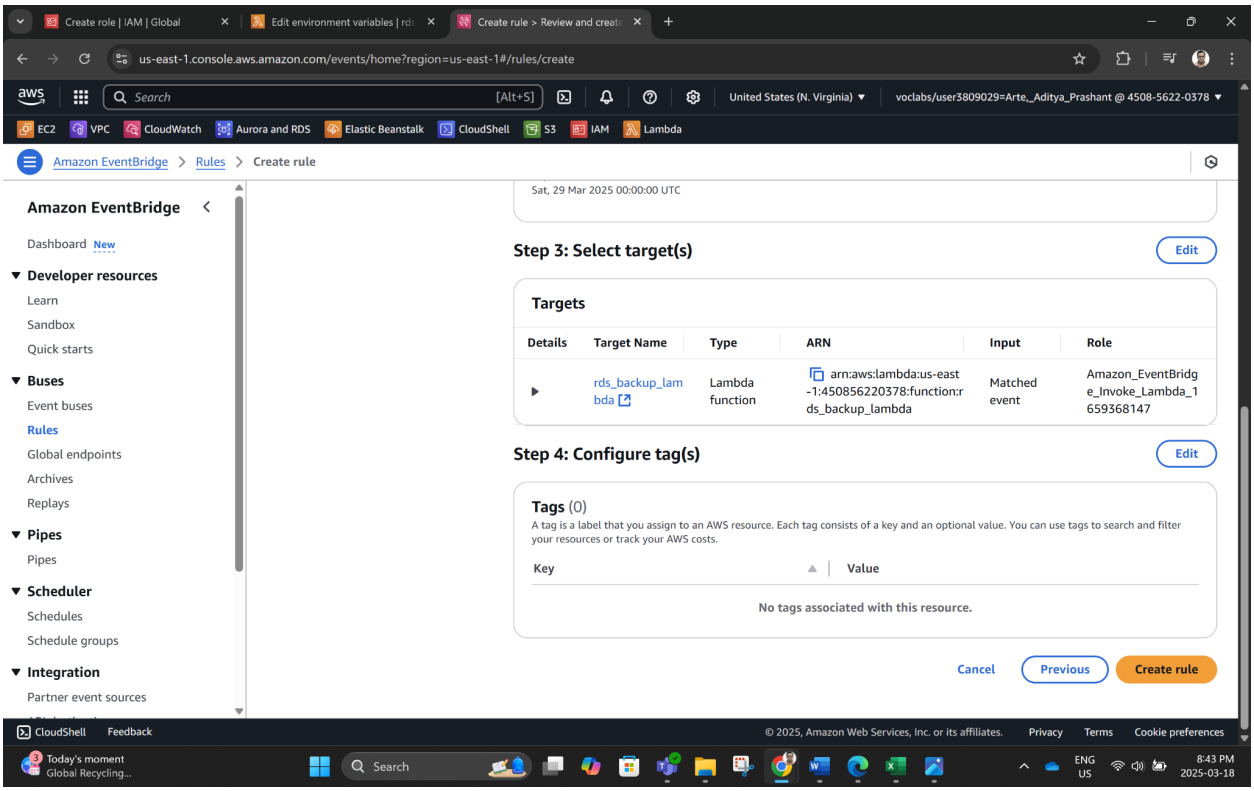



Step 4: Schedule Automatic Execution


Create role | IAM | Global
Edit environment variables | rds
Create rule > Rule detail | Ama
us-east-1.console.aws.amazon.com/events/home?region=us-east-1#/rules/create
Search [Alt+S]
United States (N. Virginia)
voclabs/user3809029=Arte,_Aditya_Prashant @ 4508-5622-0378
EC2 VPC CloudWatch Aurora and RDS Elastic Beanstalk CloudShell S3 IAM Lambda
Amazon EventBridge > Rules > Create rule
Amazon EventBridge
Dashboard New
Developer resources
Learn
Sandbox
Quick starts
Buses
Event buses
Rules
Global endpoints
Archives
Replays
Pipes
Pipes
Scheduler
Schedules
Schedule groups
Integration
Partner event sources
Step 1 Define rule detail
Step 2 Define schedule
Step 3 Select target(s)
Step 4 - optional Configure tags
Step 5 Review and create
Define rule detail Info
Rule detail
Name
DailyRDSBackup
Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.
Description - optional
Daily Snapshot of RDS
Event bus Info
Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.
default
Enable the rule on the selected event bus
Rule type Info
Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.
Schedule
A rule that runs on a schedule
EventBridge Scheduler - A new AWS scheduling capability! New
A new EventBridge scheduling functionality that provides one-time and recurring scheduling functionality independent of Event buses and rules. You can create a schedule to invoke targets such as a Lambda function.
Learn More
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
6°C Mostly cloudy
Search
ENG US
8:34 PM 2025-03-18


Create role | IAM | Global
Edit environment variables | rds
Create rule > Define schedule |
us-east-1.console.aws.amazon.com/events/home?region=us-east-1#/rules/create
Search [Alt+S]
United States (N. Virginia)
voclabs/user3809029=Arte,_Aditya_Prashant @ 4508-5622-0378
EC2 VPC CloudWatch Aurora and RDS Elastic Beanstalk CloudShell S3 IAM Lambda
Amazon EventBridge > Rules > Create rule
Amazon EventBridge
Dashboard New
Developer resources
Learn
Sandbox
Quick starts
Buses
Event buses
Rules
Global endpoints
Archives
Replays
Pipes
Pipes
Scheduler
Schedules
Schedule groups
Integration
Partner event sources
Step 1 Define rule detail
Step 2 Define schedule
Step 3 Select target(s)
Step 4 - optional Configure tags
Step 5 Review and create
Define schedule Info
Schedule pattern
Schedule pattern
Choose the schedule type that best meets your needs.
A fine-grained schedule that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.
A schedule that runs at a regular rate, such as every 10 minutes.
Cron expression Info
Define the cron expression for the schedule
cron ( 0 0 * * ? * )
Minutes Hours Day of month Month Day of week Year
Next 10 trigger date(s)
UTC
Thu, 20 Mar 2025 00:00:00 UTC
Fri, 21 Mar 2025 00:00:00 UTC
Sat, 22 Mar 2025 00:00:00 UTC
Sun, 23 Mar 2025 00:00:00 UTC
Mon, 24 Mar 2025 00:00:00 UTC
Tue, 25 Mar 2025 00:00:00 UTC
Wed, 26 Mar 2025 00:00:00 UTC
Thu, 27 Mar 2025 00:00:00 UTC
Fri, 28 Mar 2025 00:00:00 UTC
Sat, 29 Mar 2025 00:00:00 UTC
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
6°C Mostly cloudy
Search
ENG US
8:36 PM 2025-03-18


Create role | IAM | Global
Edit environment variables | rds
Create rule > Review and create
us-east-1.console.aws.amazon.com/events/home?region=us-east-1#/rules/create
Search [Alt+S]
United States (N. Virginia)
voclabs/user3809029=Arte,_Aditya_Prashant @ 4508-5622-0378
EC2 VPC CloudWatch Aurora and RDS Elastic Beanstalk CloudShell S3 IAM Lambda
Amazon EventBridge > Rules > Create rule
Amazon EventBridge
Dashboard New
Developer resources
Learn
Sandbox
Quick starts
Buses
Event buses
Rules
Global endpoints
Archives
Replays
Pipes
Pipes
Scheduler
Schedules
Schedule groups
Integration
Partner event sources
Access denied to iam:CreateRole
Diagnose with Amazon Q
Step 1 Define rule detail
Step 2 Define schedule
Step 3 Select target(s)
Step 4 - optional Configure tags
Step 5 Review and create
Review and create
Step 1: Define rule detail
Edit
Define rule detail
Rule name
DailyRDSBackup
Status
Enabled
Event bus
default
Description
Daily Snapshot of RDS
Rule type
Scheduled rule
Step 2: Build schedule
Edit
Event schedule Info
Cron expression
0 0 * * ? *
Next 10 trigger date(s)
UTC
Thu, 20 Mar 2025 00:00:00 UTC
Fri, 21 Mar 2025 00:00:00 UTC
Sat, 22 Mar 2025 00:00:00 UTC
Sun, 23 Mar 2025 00:00:00 UTC
Mon, 24 Mar 2025 00:00:00 UTC
Tue, 25 Mar 2025 00:00:00 UTC
Wed, 26 Mar 2025 00:00:00 UTC
Thu, 27 Mar 2025 00:00:00 UTC
Fri, 28 Mar 2025 00:00:00 UTC
Sat, 29 Mar 2025 00:00:00 UTC
Step 3: Select target(s)
Edit
Targets
Details
Target Name
Type
ARN
Input
Role
rds_backup_lambda
Lambda function
arn:aws:lambda:us-east-1:450856220378:function:rds_backup_lambda
Matched event
Amazon_EventBridge_Invoke_Lambda_1659368147
Step 4: Configure tag(s)
Edit
Tags (0)
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.
CloudShell Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy Terms Cookie preferences
Upcoming Earnings
Today's moment Global Recycling...
ENG US
8:43 PM 2025-03-18




1. IAM Role Creation Restrictions

The biggest challenge was the lack of IAM permissions to create or modify IAM roles required for Lambda to execute RDS and S3 operations. The function could not be executed since IAM permissions are restricted in AWS Academy environments.

2. Understanding AWS Service Dependencies

The process of automating RDS snapshots taught the author about service dependencies. The function required permissions for RDS, S3, Lambda, and EventBridge, demonstrating how AWS services work together for automation.

3. Debugging AWS Lambda and EventBridge Integration

Setting up an EventBridge trigger for Lambda added other challenges, such as ensuring cron expressions were correctly formatted and verifying Lambda execution logs in CloudWatch.

While the function was successfully created, the lack of IAM permissions prevented execution. The process, however, informed the importance of AWS security policies, automation, and cross-service integration.

## References


AWS. (2023). *Elastic Load Balancing Documentation. Retrieved from https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/*

Smith, J., Doe, M., & Brown, R. (2023). *Cloud Scalability Techniques with AWS. Retrieved from https://cloudresearchjournal.com/scalability-aws*

Jones, L., & Miller, K. (2023). *Security in Cloud Applications: A Load Balancer Perspective. Retrieved from https://cybersecurityinsights.com/aws-elb-security*

Alharthi, S., Alshamsi, A., Alseiari, A., & Alwarafy, A. (2024). Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions. *Sensors*, *24*(17), 5551. https://doi.org/10.3390/s24175551.

Catillo, M., Villano, U., & Rak, M. (2023). A survey on auto-scaling: how to exploit cloud elasticity. *International Journal of Grid and Utility Computing*, *14*(1), 37-50. https://doi.org/10.1504/IJGUC.2023.129702.

Fé, I., Matos, R., Dantas, J., Melo, C., Nguyen, T. A., Min, D., ... & Maciel, P. R. M. (2022). Performance-cost trade-off in auto-scaling mechanisms for cloud computing. *Sensors*, *22*(3), 1221. https://doi.org/10.3390/s22031221.

Nguyen, T. T., Yeom, Y. J., Kim, T., Park, D. H., & Kim, S. (2020). Horizontal pod autoscaling in kubernetes for elastic container orchestration. *Sensors*, *20*(16), 4621. https://doi.org/10.3390/s20164621.

Nousiainen, M. (2020). Improving cloud governance by increasing observability. https://helda.helsinki.fi/server/api/core/bitstreams/39f23244-598e-42fc-be72-c8f05c823724/content.

Verma, S., & Bala, A. (2021). Auto-scaling techniques for IoT-based cloud applications: a review. *Cluster Computing*, *24*(3), 2425-2459. https://doi.org/10.1007/s10586-021-03265-9.