

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.

The screenshot shows the AWS CloudFront console with a success message: "Successfully created distribution: LabCloudFront". The distribution is listed with the following details:

Setting	Value
Distribution ID	d1234567890123456789012345678901
ARN	arn:aws:cloudfront::1234567890123456789012345678901/distribution/d1234567890123456789012345678901
Region	us-east-1
Status	Active
Last modified	1 hour ago

The "Listeners and rules" section shows one rule for port 80:

Protocol	Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
HTTP	80	Forward to target group	1 rule	arn:aws:cloudfront::1234567890123456789012345678901/rule/1	Not applicable	Not applicable	Not applicable	Not applicable

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.

The screenshot shows the AWS EC2 Target Groups interface. On the left, a sidebar menu includes EC2, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main content area is titled "Labgroup". It displays "Details" for a target group named "arn:aws:elasticloadbalancing:us-east-1:165859327443:targetgroup/Labgroup/0563d411a70fce10". The "Protocol" is set to "HTTP: 80" and the "Protocol version" is "HTTP1". The "VPC" is associated with "vpc-0421f12256735ac9". Below this, a summary table shows 0 total targets, 0 healthy, 0 unhealthy, 0 unused, 0 initial, and 0 draining. A "Targets" tab is selected, showing a table for "Registered targets (0)". The table has columns for Instance ID, Name, Port, Zone, Health status, Health status details, Administrative o..., Override details, and Launch time. A note states: "Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets." A "Register targets" button is present. At the bottom right, there are links for CloudShell, Feedback, and a footer with copyright information.

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.

The screenshot shows the AWS EC2 Security Groups page. On the left, there's a navigation sidebar with links like Instances, Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main content area displays a table titled "Security Groups (1/5) Info". The table has columns for Name, Security group ID, Security group name, VPC ID, Description, and Owner. It lists four entries: "Web Security Group" (sg-00b9f5b0c66cfeed7), "default" (sg-0cc66a891eed686ea), "BastionSecurityGroup" (sg-0e10feea72f88a9f), and "DB Security Group" (sg-03178a526aaecf78). Below this, a specific security group named "sg-00b9f5b0c66cfeed7 - Web Security Group" is selected, showing its details: Security group name (Web Security Group), Security group ID (sg-00b9f5b0c66cfeed7), Description (Enable HTTP access), VPC ID (vpc-0421f512256735ac9), Owner (165859327443), Inbound rules count (2 Permission entries), and Outbound rules count (1 Permission entry).

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

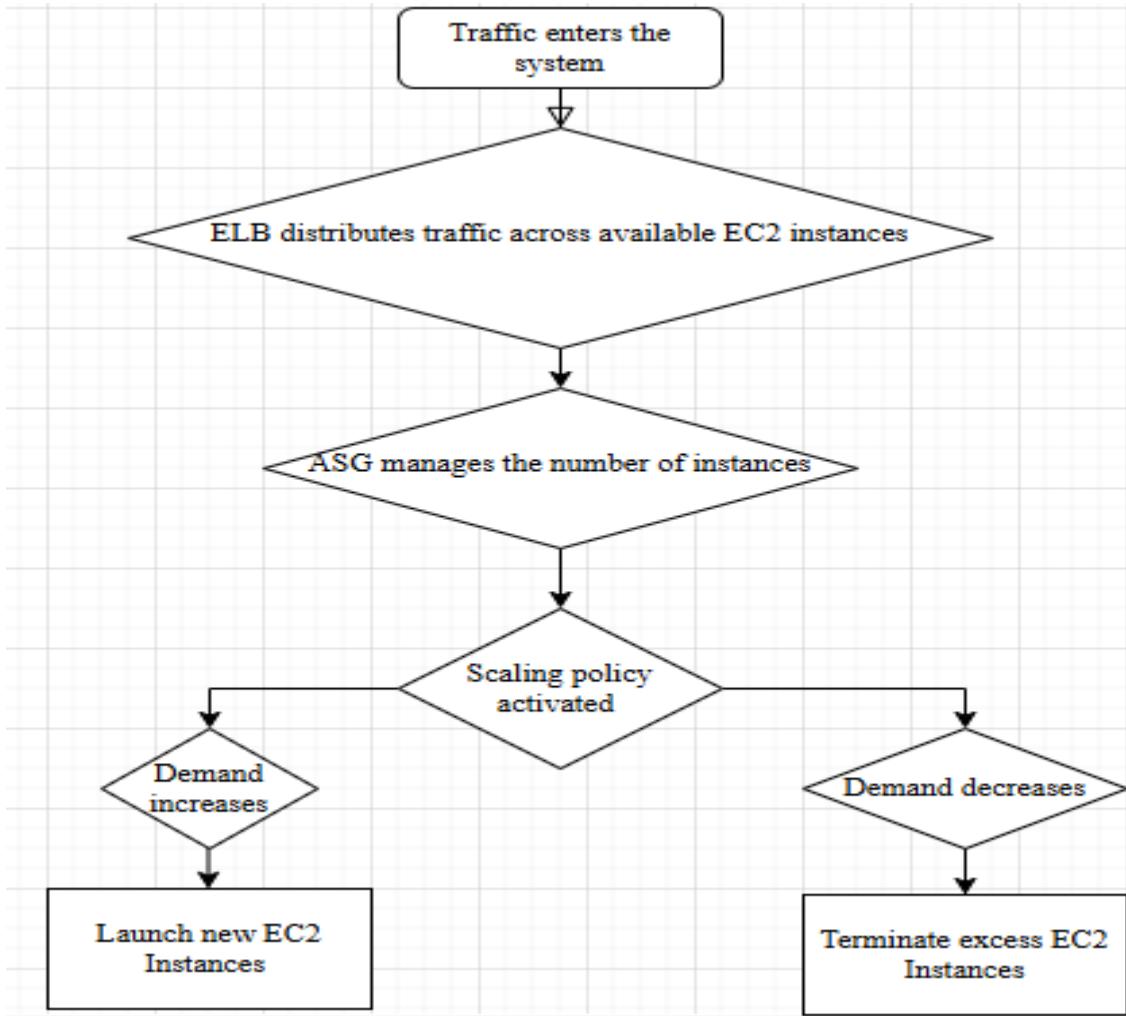
An Autoscaling Group (ASG) is fundamental in dynamically controlling and managing resources (Nguyen et al., 2020). When developing the social media site, the process of creating an autoscaling group will commence with creating a launch template. The launch template will configure the AMI and instance 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

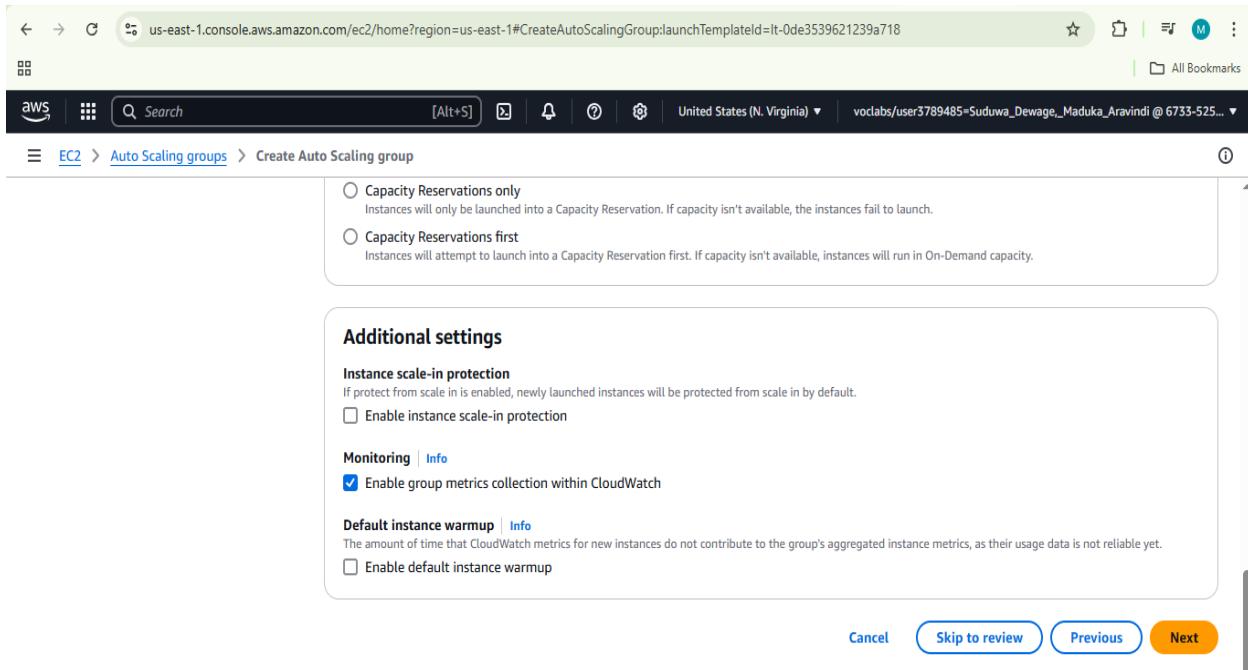
It is important to maintain reliability, security and performance as UCW digital platform is used to keep the real time connections and resources sharing among students and faculty. To accomplish these objectives, Continuous monitoring using Amazon CloudWatch throughout the system is crucial as it provides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

The screenshot shows the AWS EC2 'Create launch template' wizard. On the left, there are several dropdown menus for configuration options: 'Termination protection' (set to 'Don't include in launch template'), 'Stop protection' (set to 'Don't include in launch template'), 'Detailed CloudWatch monitoring' (set to 'Enable'), 'Credit specification' (set to 'Don't include in launch template'), 'Placement group' (set to 'Don't include in launch template'), and 'EBS-optimized instance' (set to 'Don't include in launch template'). On the right, the 'Summary' section displays the selected configurations: Software Image (AMI) is 'Lab AMI for Web Server ami-0e3d896c9bde15349'; Virtual server type (instance type) is 't2.micro'; Firewall (security group) is 'Web Security Group'; and Storage (volumes) is '1 volume(s) - 8 GiB'. A tooltip for the 'Free tier' is visible, stating: 'Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage for free.' At the bottom right are 'Cancel' and 'Create launch template' buttons, with the latter being orange.

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

The screenshot shows a web browser window with the URL `labelb-327477421.us-east-1.elb.amazonaws.com`. The page displays AWS navigation links for Load Test and RDS. Below these, there is a table titled "Meta-Data" with two rows: "InstanceId" (value: `i-071a0d6e895c8844c`) and "Availability Zone" (value: `us-east-1b`). At the bottom of the page, the text "Current CPU Load: 5%" is displayed.

The screenshot shows the AWS CloudWatch Alarms console at the URL `us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#alarmsV2:`. The left sidebar lists various monitoring services: AI Operations, Alarms, Logs, Metrics, X-Ray traces, Events, and Application Signals. The "Alarms" section is selected, showing 2 alarms. The top navigation bar includes search, filter, and action buttons. The main table lists the alarms with the following details:

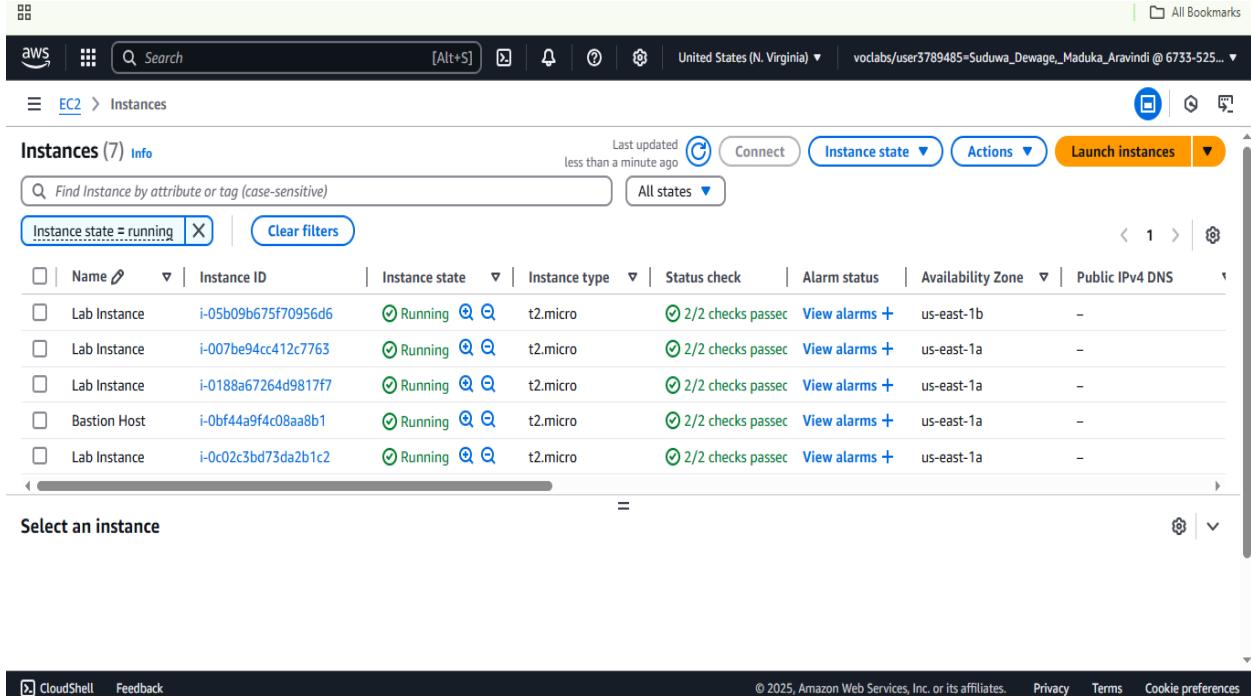
Name	State	Last state update (UTC)	Conditions
TargetTracking-Lab Auto Scaling Group- f21ce14-5538-48d0-8809-f08a6ae31f58	OK	2025-03-20 23:40:20	CPUUtilization > 60 for 3 datapoints within 3 minutes
TargetTracking-Lab Auto Scaling Group- AlarmLow-436a50ac-0217-45fa-84b2-937bebc7f6c6	Insufficient data	2025-03-20 23:38:49	CPUUtilization < 54 for 15 datapoints within 15 minutes

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.

The image contains two screenshots. The top screenshot shows a web browser window titled 'AWS Technical Essentials v4.1' with the URL 'labelb-327477421.us-east-1.elb.amazonaws.com/load.php'. The page displays a message: 'Under High CPU Load! (auto refresh in 5 seconds)' and 'Current CPU Load: 100%'. The bottom screenshot shows the AWS CloudWatch Alarms console in the 'Alarms' section. It lists two alarms: 'AlarmHigh-' and 'AlarmLow-'. The 'AlarmHigh-' alarm is in an 'In alarm' state, triggered on March 20, 2025, at 23:51:20, with the condition 'CPUUtilization > 60 for 3 datapoints within 3 minutes'. The 'AlarmLow-' alarm is in an 'OK' state, triggered on March 20, 2025, at 23:49:27, with the condition 'CPUUtilization < 45 for 15 datapoints within 15 minutes'.

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



The screenshot shows the AWS EC2 Instances page with the following details:

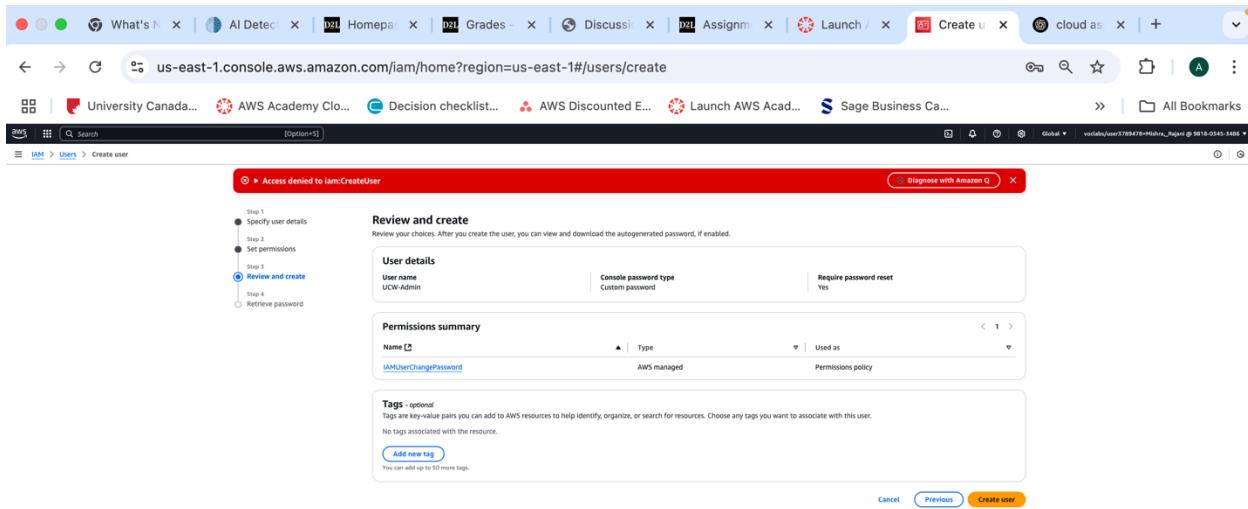
- Instances (7) Info**: The page displays 7 instances.
- Last updated**: less than a minute ago.
- Actions**: Connect, Instance state, Actions, Launch instances.
- Filters**: Instance state = running, Clear filters.
- Table Headers**: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS.
- Table Data** (5 rows shown):

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Lab Instance	i-05b09b675f70956d6	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	-
Lab Instance	i-007be94cc412c7763	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-
Lab Instance	i-0188a67264d9817f7	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-
Bastion Host	i-0bf44a9f4c08aa8b1	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-
- Select an instance**: A dropdown menu for selecting an instance.
- Page Bottom**: CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

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.

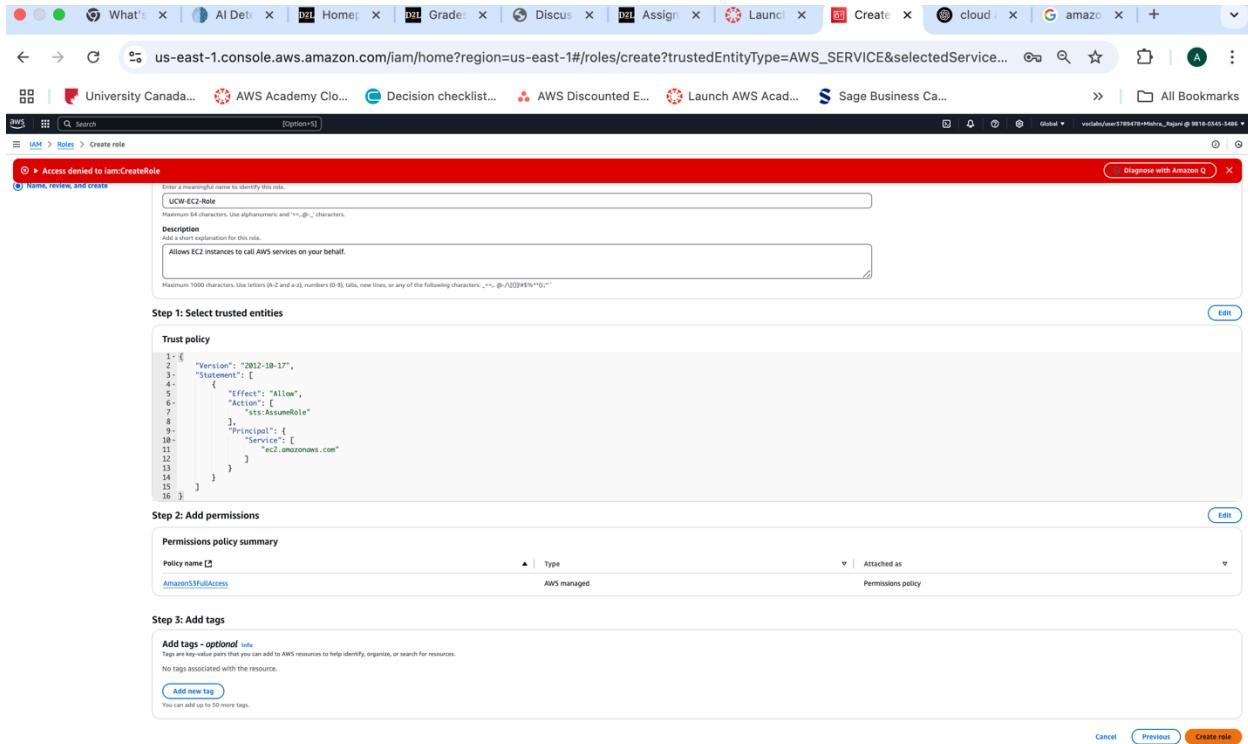
4. Implementing AWS Identity and Access Management (IAM)

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



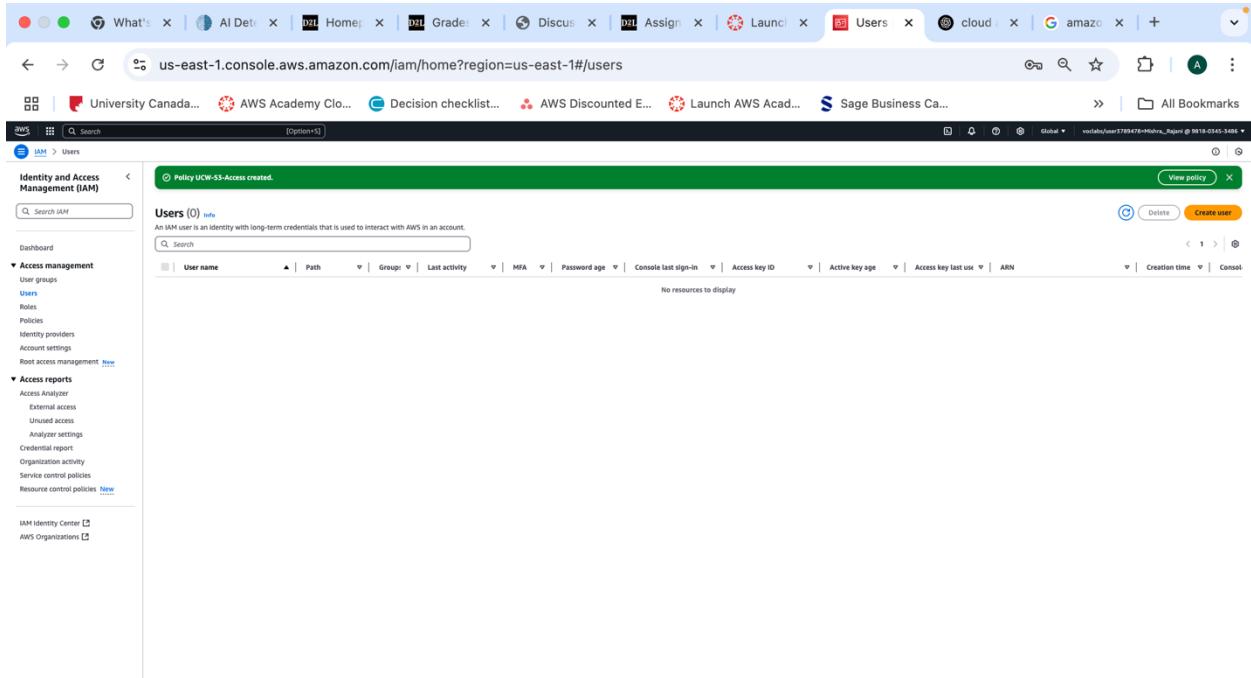
In this screenshot, we can see a visual depiction of an unauthorized attempt to establish an IAM group under the name UCW-Developers. The action failed while creating an IAM group named UCW-Developers because the system denied access to `iam:CreateGroup` permissions. AWS best practice suggests building 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

The screenshot shows the AWS IAM Policies page. The left sidebar includes sections for Identity and Access Management (IAM), Access management, Access reports, and IAM Identity Center. The main content area displays a table of 1342 policies. The columns in the table are Policy name, Type, Used as, and Description. The table lists various AWS managed policies such as AccessAnalyzerServiceRolePolicy, AdministratorAccess, AdminstratorAccess-Amplify, AdminstratorAccess-AWSFleetDeployment, AIOpsAssistantPolicy, AIOpsConsoleAdminPolicy, AIOpsOperatorAccess, AIOpsReadOnlyAccess, AlexaForBusinessDeviceSetup, AlexaForBusinessFullAccess, AlexaForBusinessGatewayExecution, AlexaForBusinessGatewayInvokeAccess, AlexaForBusinessNetworkProfileServicePolicy, AlexaForBusinessPolicyDelegateAccessPolicy, AlexaForBusinessReadOnlyAccess, AmazonAPIGatewayAdministrator, AmazonAPIGatewayInvokeAccess, AmazonAPIGatewayPushToCloudWatchLogs, AmazonAppFlowFullAccess, and AmazonAppFlowReadOnlyAccess. The 'Used as' column shows 'None' for all listed policies.

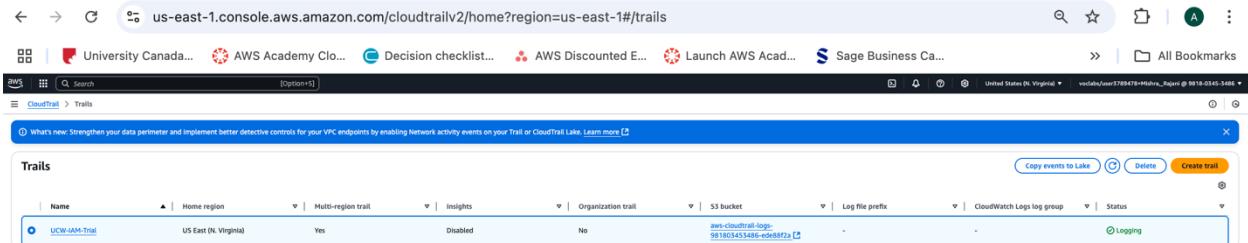
This screenshot captures the failure in creating the IAM role named UCW-EC2-Role. AWS services including EC2 instances need IAM roles to establish secure access to other AWS resources because embedding credentials would be insecure. The "Access denied to iam:CreateRole" error resulted in this failure 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 of role-based access systems in AWS deployments.

5. Creating a Specific Lambda Function

Step 1: Create an S3 Bucket for Snapshots

The screenshot shows the AWS S3 'Create bucket' wizard. At the top, the URL is `us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general`. The navigation bar includes links for EC2, VPC, CloudWatch, Aurora and RDS, Elastic Beanstalk, CloudShell, and S3. The current page is 'Create bucket'. The 'General configuration' tab is active, showing the 'AWS Region' set to 'US East (N. Virginia) us-east-1'. Under 'Bucket type', 'General purpose' is selected (indicated by a blue outline). The 'Bucket name' field contains 'rds-snapshots-backup-ucw'. The 'Object Ownership' tab is also visible at the bottom.

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type Info

General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info
rds-snapshots-backup-ucw

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

Format: s3://bucket/prefix

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.

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Trending videos Ted Lasso lives!...

Search

8:14 PM 2025-03-18

Create S3 bucket | S3 | us-east-1

us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general

AWS Search [Alt+S] United States (N. Virginia) vocabs/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-1

us-east-1.console.aws.amazon.com/s3/bucket/create?region=us-east-1&bucketType=general

AWS Search [Alt+S] United States (N. Virginia) vocabs/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

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

8°C Mostly cloudy Search ENG US 8:15 PM 2025-03-18

Step 2: Create IAM Roles

The screenshot shows the 'Create role' wizard in the AWS IAM console. The current step is 'Select trusted entity'. The navigation bar at the top includes links for EC2, VPC, CloudWatch, Aurora and RDS, Elastic Beanstalk, CloudShell, S3, and IAM.

The main interface displays the 'Trusted entity type' section. A sidebar on the left shows the progress: 'Step 2' (selected), 'Add permissions', and 'Step 3' (disabled). The 'Service or use case' dropdown is set to 'Lambda'.

The 'Trusted entity type' section contains four options:

- AWS service**: Allows AWS services like EC2, Lambda, or others to perform actions in this account. This option is selected.
- AWS account**: Allows entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation**: Allows users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy**: Create a custom trust policy to enable others to perform actions in this account.

The 'Use case' section indicates that Lambda is chosen for the specified service. The 'Service or use case' dropdown also shows 'Lambda'.

At the bottom, there are buttons for 'Next Step' and 'Cancel'.

Screenshot of the AWS IAM 'Create role' wizard, Step 2: Add permissions.

Add permissions

Permissions policies (3/1044) Info

Choose one or more policies to attach to your new role.

Filter by Type: All types | 20 matches

Policy name	Type	Description
AmazonS3ObjectLambdaExecutionRole	AWS managed	Provides AWS Lambda functions perm...
AmazonSageMakerPartnerServiceCatalog	AWS managed	Service role policy used by the AWS La...
AmazonSageMakerServiceCatalogPro	AWS managed	Service role policy used by the AWS La...
AWSCodeDeployRoleForLambda	AWS managed	Provides CodeDeploy service access to ...
AWSCodeDeployRoleForLambdaLimited	AWS managed	Provides CodeDeploy service limited a...
AWSDeepLensLambdaFunctionAccess	AWS managed	This policy specifies permissions requir...
AWSLambda_FullAccess	AWS managed	Grants full access to AWS Lambda serv...
AWSLambda_ReadOnlyAccess	AWS managed	Grants read-only access to AWS Lamb...
AWSLambdaBasicExecutionRole	AWS managed	Provides write permissions to CloudW...
AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to Dynam...

Selected policy: AWSLambdaBasicExecutionRole

https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/roles/details... © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 8:19 PM 2025-03-18

Screenshot of the AWS IAM 'Create role' wizard, Step 3: Name, review, and create.

Name, review, and create

Role details

Role name: Lambda_RDS_Backup_Role

Description: Allows Lambda functions to call AWS services on your behalf.

Step 1: Select trusted entities

Trust policy

```

1  [
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "lambda:InvokeFunction"
8       ]
9     }
10  ]
11 ]
  
```

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 8:20 PM 2025-03-18

Step 3: Create a Lambda Function

The screenshot shows the AWS Lambda 'Create function' wizard. The top navigation bar includes tabs for 'Create role | IAM | Global' and 'Create function | Functions | L'. The main title is 'Create function' with an 'Info' link. Below it, a sub-header says 'Choose one of the following options to create your function.' with three options:

- Author from scratch: Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
rds_backup_lambda

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.13

Architecture Info
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
LabRole

[View the LabRole role](#) on the IAM console.

Additional Configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Cancel **Create function**

Step 4: Add Lambda Code

```

lambda_function.py
1 import boto3
2 import datetime
3 import os
4
5 # AWS Clients
6 rds_client = boto3.client('rds')
7 s3_client = boto3.client('s3')
8
9 # Configurations
10 DB_INSTANCE_ID = "ucw-social-db"
11 S3_BUCKET_NAME = "rds-snapshots-backup-ucw"
12
13 def lambda_handler(event, context):
14     try:
15         # Generate Snapshot Name
16         snapshot_id = f'{DB_INSTANCE_ID}-snapshot-{datetime.datetime.now().strftime("%Y%m%d-%H%M%S")}'
17
18         # Create RDS Snapshot
19         response = rds_client.create_db_snapshot(
20             DBSnapshotIdentifier=snapshot_id,
21             DBInstanceIdentifier=DB_INSTANCE_ID
22         )
23         print(f"Snapshot: {snapshot_id} initiated successfully!")

```

The screenshot shows the AWS Lambda console interface. In the top navigation bar, the URL is `us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/rds_backup_lambda?newFunction=true&tab=code`. The sidebar shows the Lambda function `rds_backup_lambda` under the `RDS_BACKUP_LAMBDA` layer. The main area displays the Python code for the `lambda_handler` function. Below the code editor, there are buttons for `Deploy (Ctrl+Shift+U)` and `Test (Ctrl+Shift+I)`. A message at the top says, "Successfully created the function rds_backup_lambda. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." The bottom part of the screenshot shows the CloudShell interface.

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

Key

Value

[Remove](#)

[Add environment variable](#)

Encryption configuration

[Cancel](#)

[Save](#)

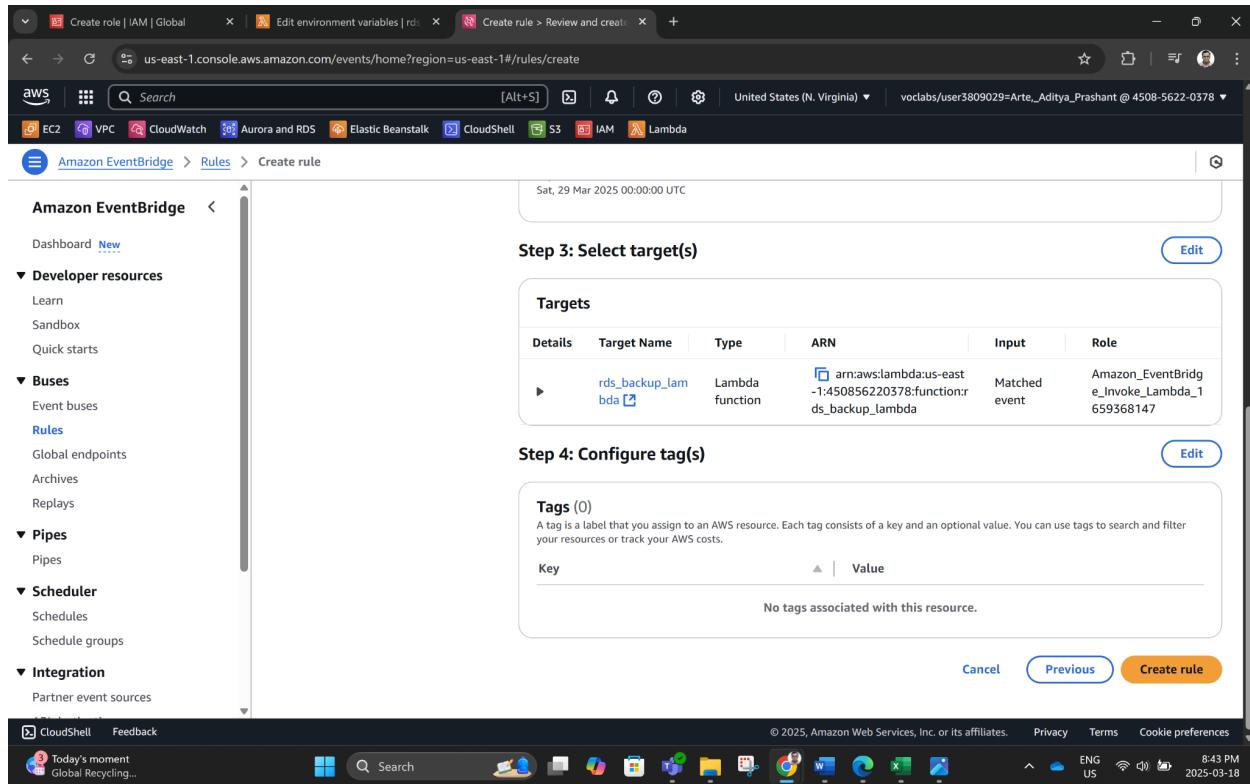
The screenshot shows the AWS Lambda console interface. The URL in the address bar is `us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/rds_backup_lambda/edit/schedule?subtab=schedule&tab=schedule`. The sidebar shows the Lambda function `rds_backup_lambda` under the `RDS_BACKUP_LAMBDA` layer. The main area displays the schedule configuration for the function. A message at the top says, "Successfully scheduled the function rds_backup_lambda. You can now change its schedule or configuration. To invoke your function with a test event, choose 'Test'." The bottom part of the screenshot shows the CloudShell interface.

Step 4: Schedule Automatic Execution

Screenshot of the AWS CloudShell interface showing the creation of an Amazon EventBridge rule.

The top section shows the "Define rule detail" step, where the rule is named "DailyRDSEBackup" and is set to run on the "default" event bus. The "Rule type" is selected as "Schedule".

The bottom section shows the "Define schedule" step, where the "Schedule pattern" is set to a fine-grained schedule running at 8:00 a.m. PST on the first Monday of every month. The cron expression is defined as "cron(0 0 * * 1)".



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>.

