Assignment 2: Automated S3 Bucket Cleanup Using AWS Lambda and Boto3

Objective: To gain experience with AWS Lambda and Boto3 by creating a Lambda function that will automatically clean up old files in an S3 bucket.

Task: Automate the deletion of files older than 30 days in a specific S3 bucket.

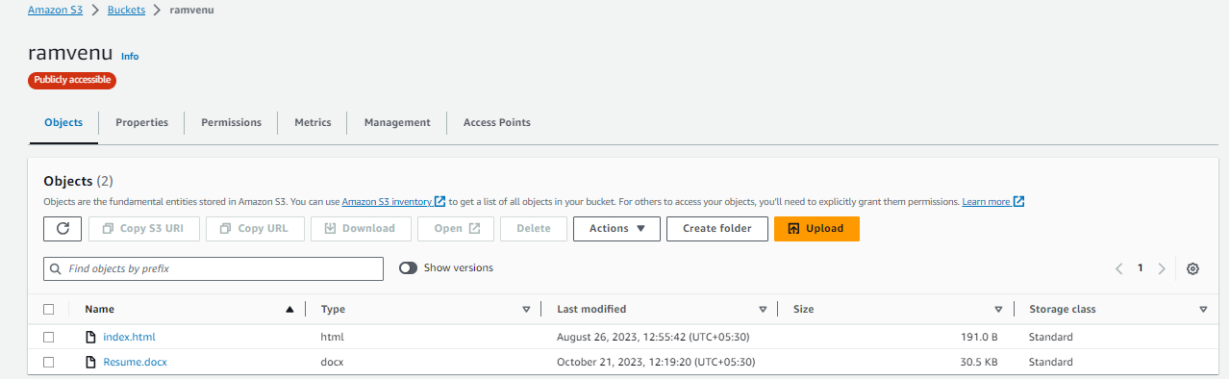
Instructions:

1. S3 Setup:

- Navigate to the S3 dashboard and create a new bucket.

- Upload multiple files to this bucket, ensuring that some files are older than 30 days (you may need to adjust your system's date temporarily for this or use old files).

Screenshot: S3 bucket with files which are older than 30 days

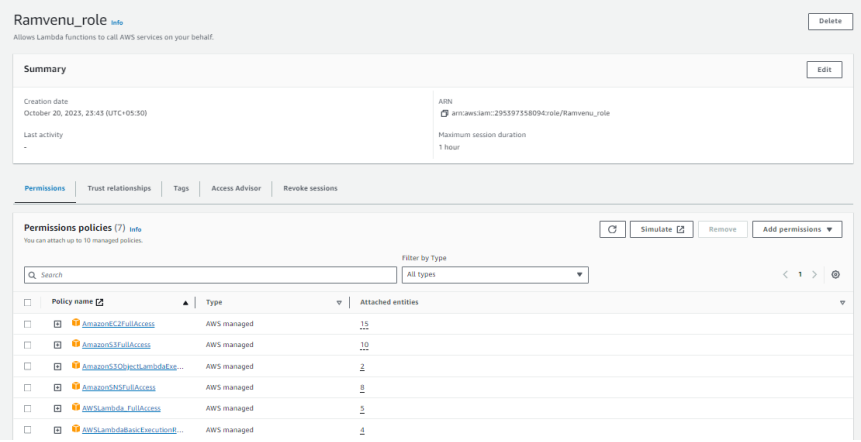


2. Lambda IAM Role:

- In the IAM dashboard, create a new role for Lambda.

- Attach the `AmazonS3FullAccess` policy to this role. (Note: For enhanced security in real-world scenarios, use more restrictive permissions.)

Screenshot: IAM Role created with all the access required to complete the tasks. (one role for all assignment questions



3. Lambda Function:

- Navigate to the Lambda dashboard and create a anew function.

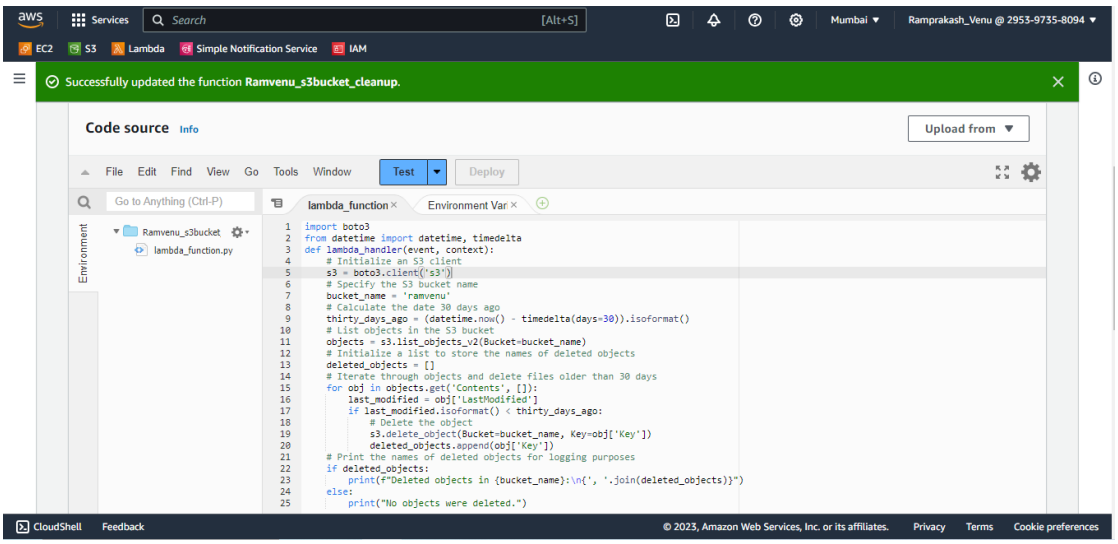
- Choose Python 3.x as the runtime.

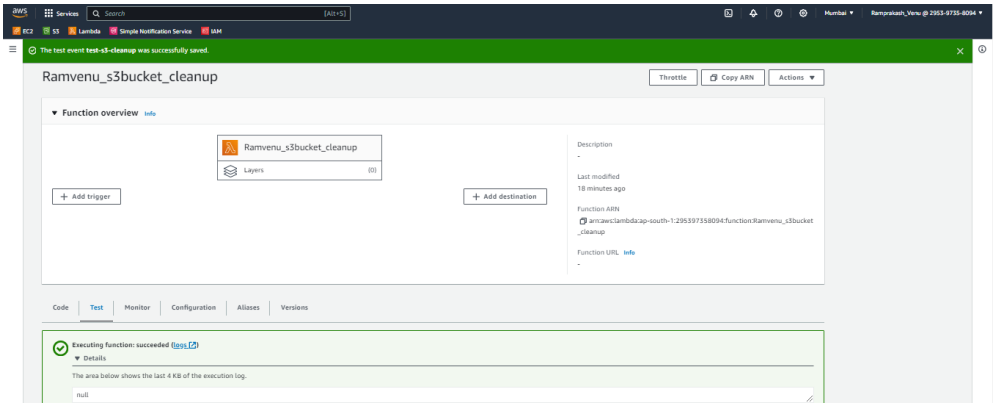
- Assign the IAM role created in the previous step.

- Write the Boto3 Python script to:

1. Initialize a boto3 S3 client.
2. List objects in the specified bucket.
3. Delete objects older than 30 days.
4. Print the names of deleted objects for logging purposes.

Screenshot: deploying Lambda codes and running it.



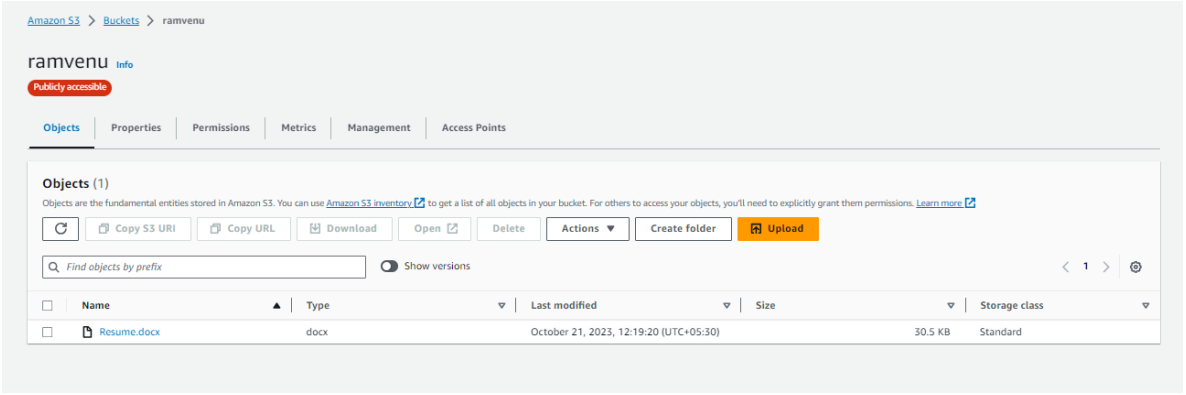


4. Manual Invocation:

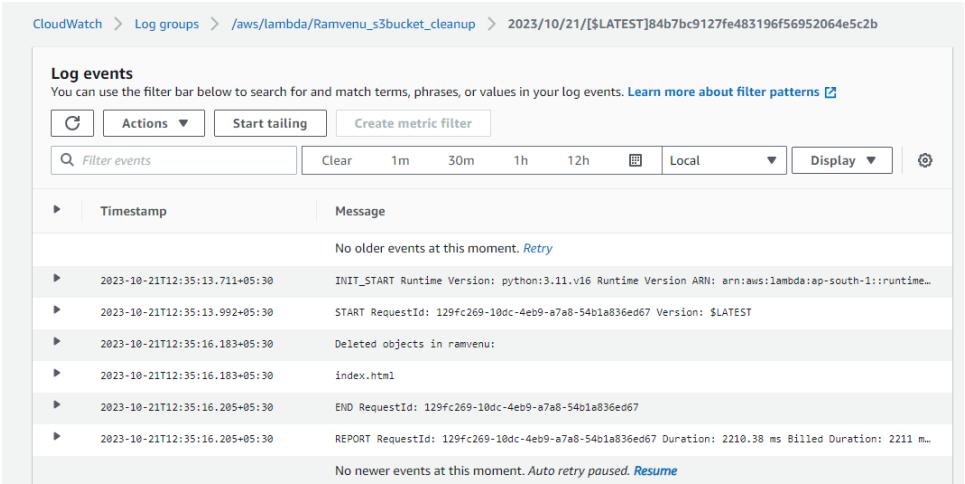
- After saving your function, manually trigger it.

- Go to the S3 dashboard and confirm that only files newer than 30 days remain.

Screenshot: S3 Bucket after clearing the file older than 30 days



Screenshot: Logfile indicating the task executed and the file name which was deleted



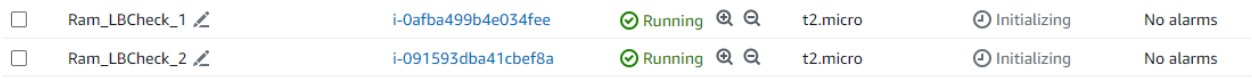
Assignment 20: Load Balancer Health Checker

Objective: Design a Lambda function that checks the health of registered instances behind an Elastic Load Balancer (ELB) and notifies via SNS if any instances are unhealthy.

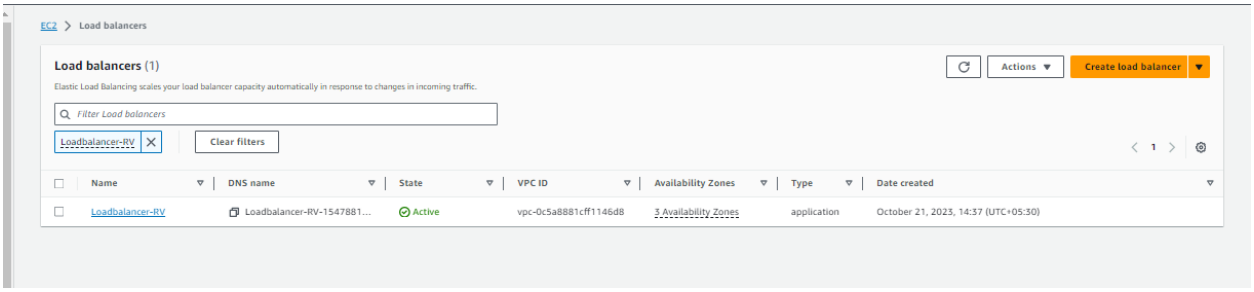
Instructions:

1. Launch Two Ec2 instances , create Loadbalancer and add these Ec2 instance into aTarget Group

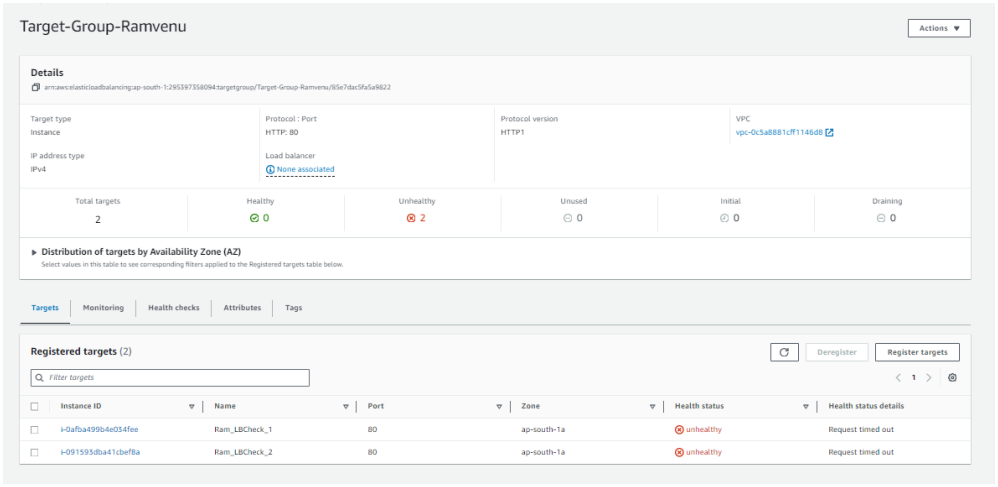
   Screenshot: Ec2 instances Launched for the task



Screenshot: Load balancer Created



Screenshot: Target Group with two instances

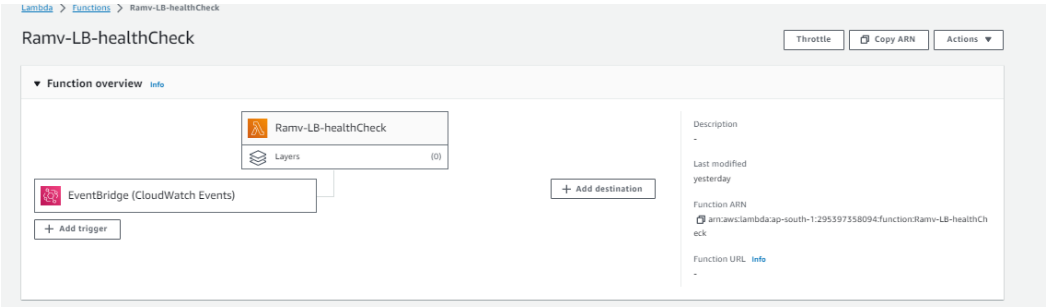


2. Create a Lambda function.

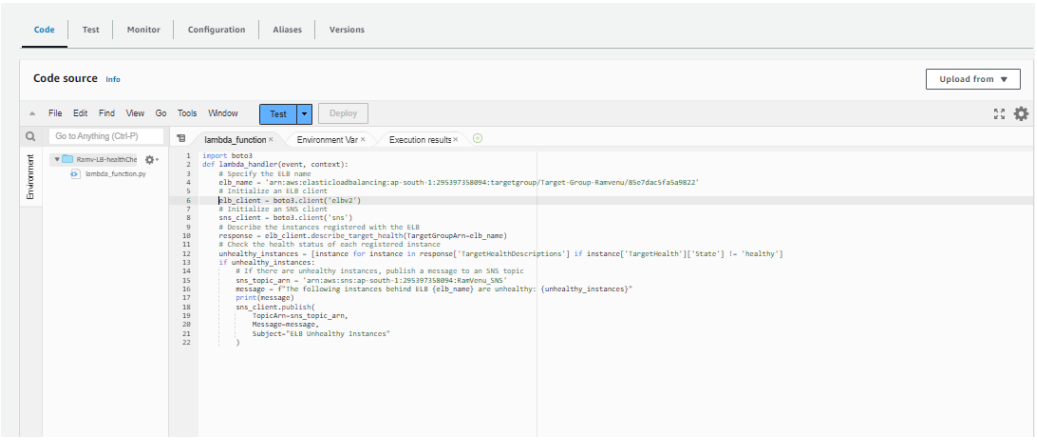
   With Boto3, configure the function to:

Check the health of registered instances behind a given ELB. If any instances are found to be unhealthy, publish a detailed message to an SNS topic.

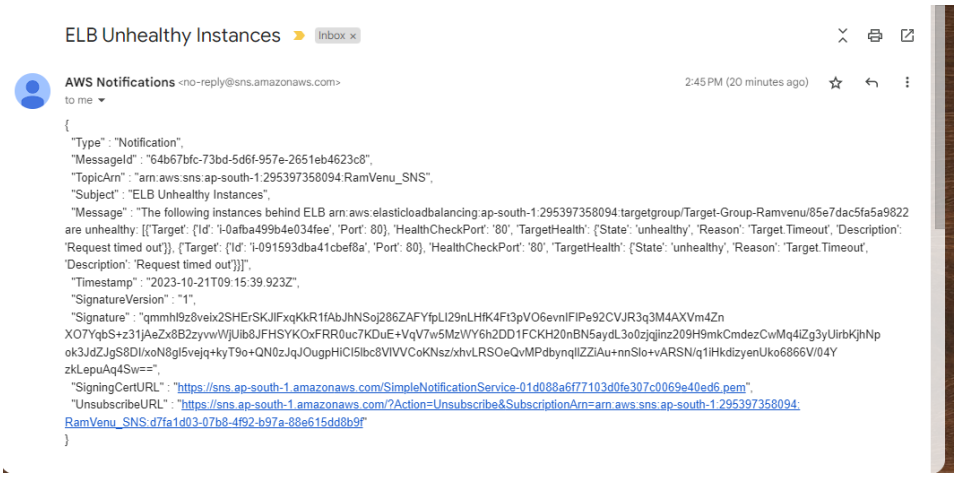
Screenshot: Lambda function created



Screenshot : Boto3 Code

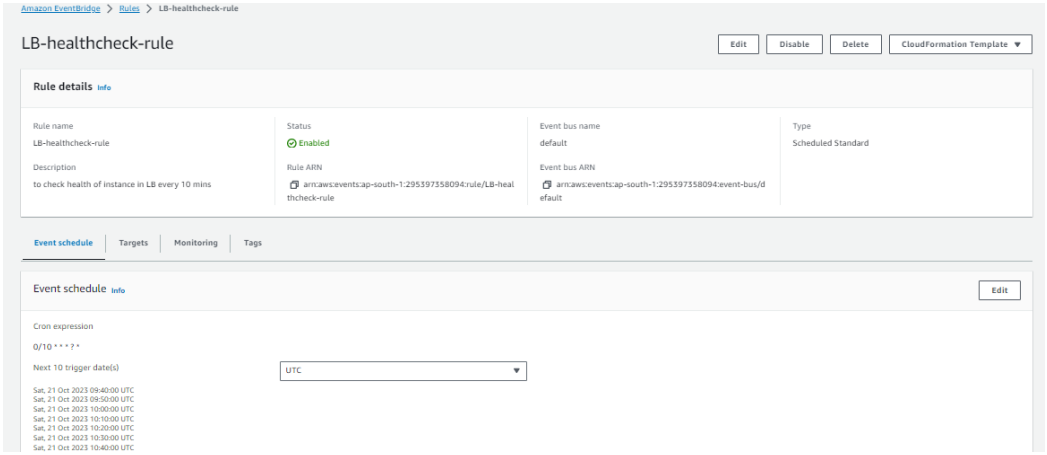


Screenshot : Detailed message with instance id notified on email

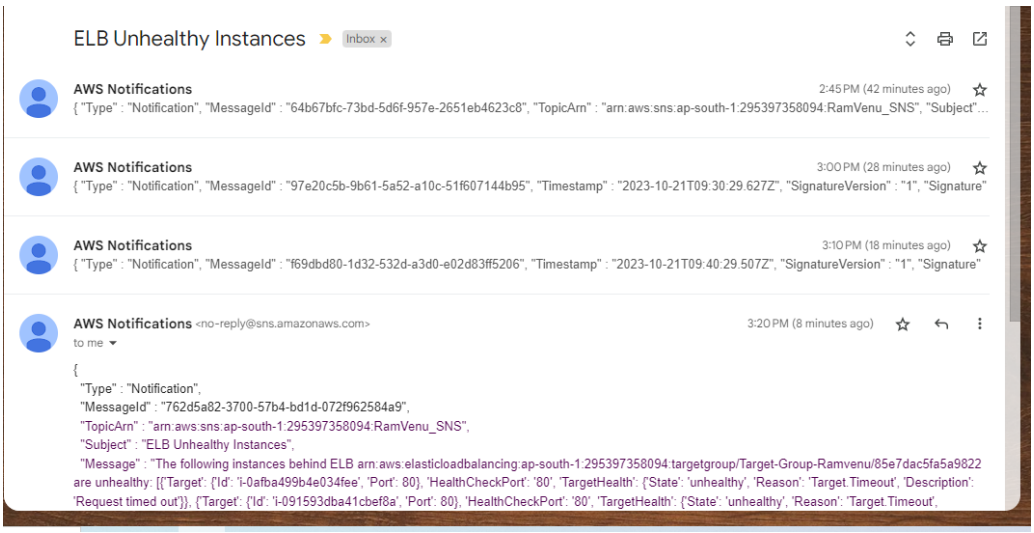


3. Set up a CloudWatch event to trigger this Lambda function every 10 minutes.

Screenshot: event trigger - Cron Job set to check health status every 10 mins and notify if instances is unheathy



Screenshot : SNS message on email for every 10 mins



Assignment 6: Auto-Tagging EC2 Instances on Launch Using AWS Lambda and Boto3

Objective: Learn to automate the tagging of EC2 instances as soon as they are launched, ensuring better resource tracking and management.

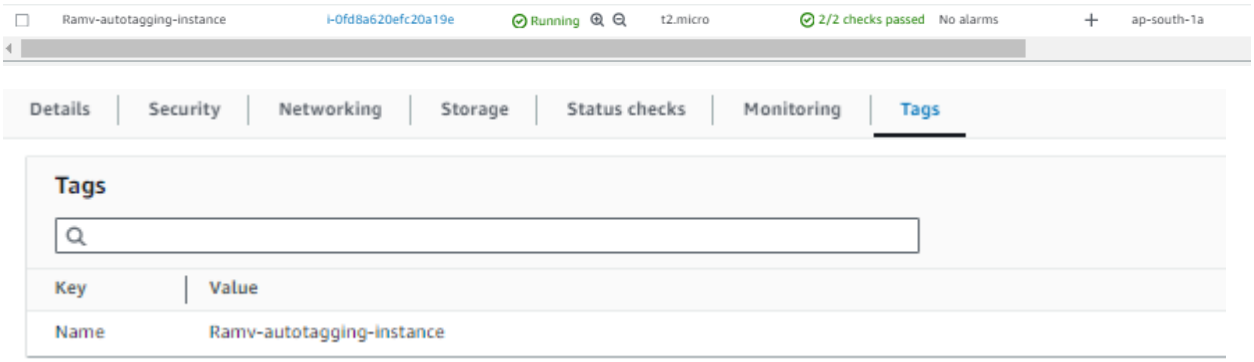
Task: Automatically tag any newly launched EC2 instance with the current date and a custom tag.

Instructions:

1. EC2 Setup:

- Ensure you have the capability to launch EC2 instances.

Screenshot: EC2 instance launched and tags before.

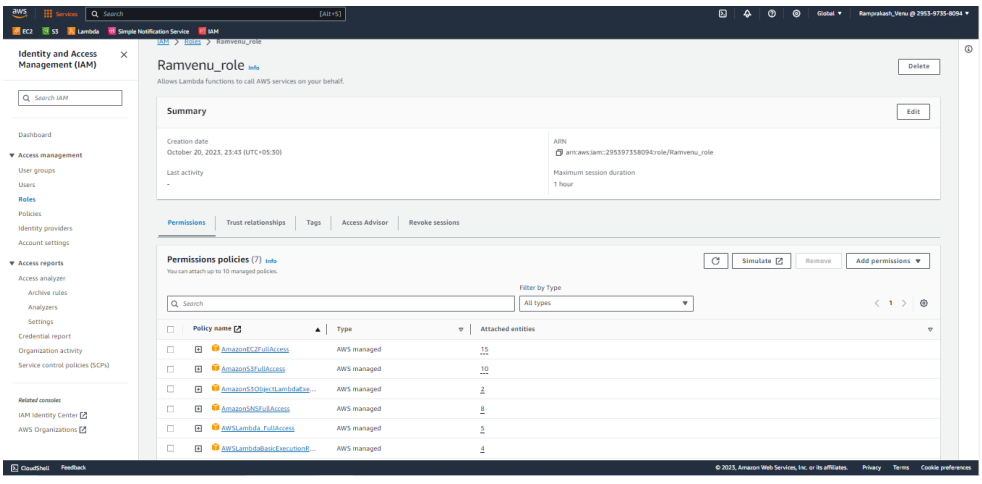


2. Lambda IAM Role:

- In the IAM dashboard, create a new role for Lambda.

- Attach the `AmazonEC2FullAccess` policy to this role.

Screenshot: IAM Role created with all the access required to complete the tasks. (one role for all assignment questions)



3. Lambda Function:

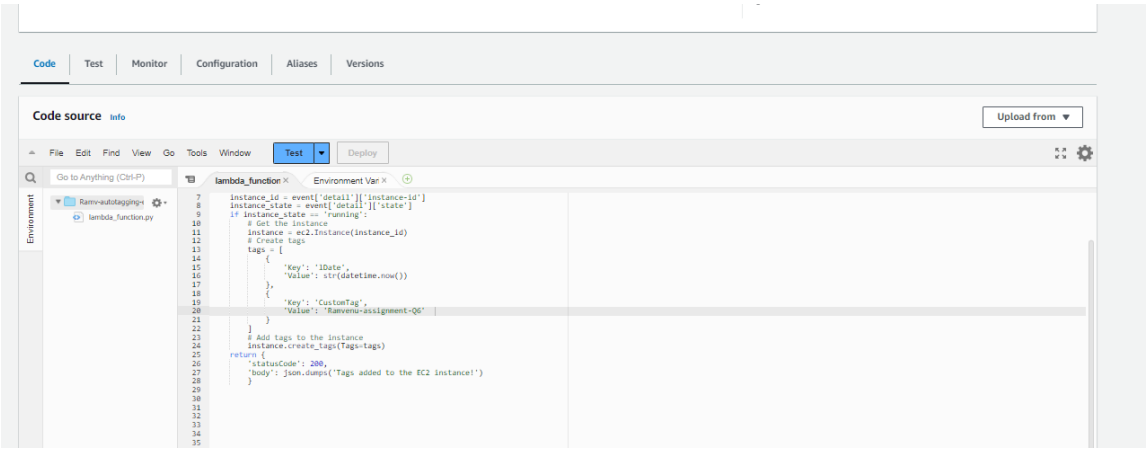
- Navigate to the Lambda dashboard and create a new function.

- Choose Python 3.x as the runtime.

- Assign the IAM role created in the previous step.

- Write the Boto3 Python script to:

1. Initialize a boto3 EC2 client.
2. Retrieve the instance ID from the event.
3. Tag the new instance with the current date and another tag of your choice
4. Print a confirmation message for logging purposes.

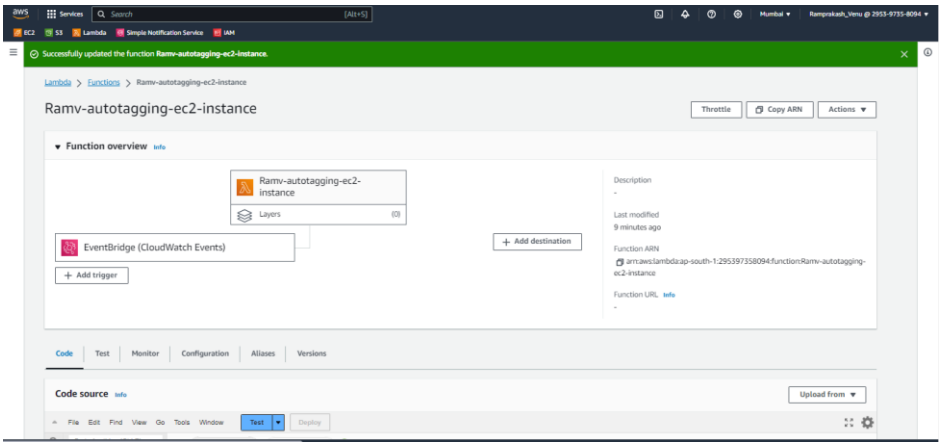


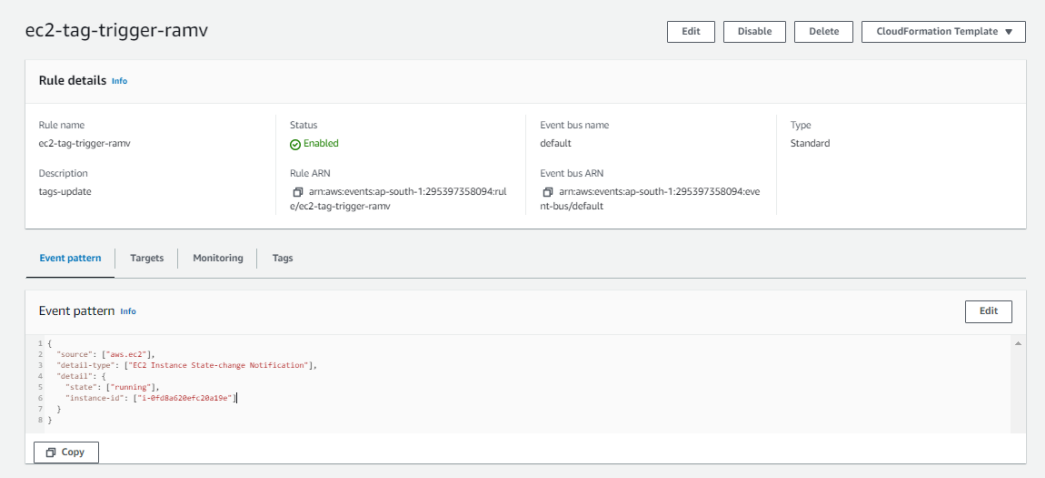
4. CloudWatch Events:

- Set up a CloudWatch Event Rule to trigger on the EC2 instance launch event.

- Attach the Lambda function as the target.

Screenshot: Triggers configured for lambda function



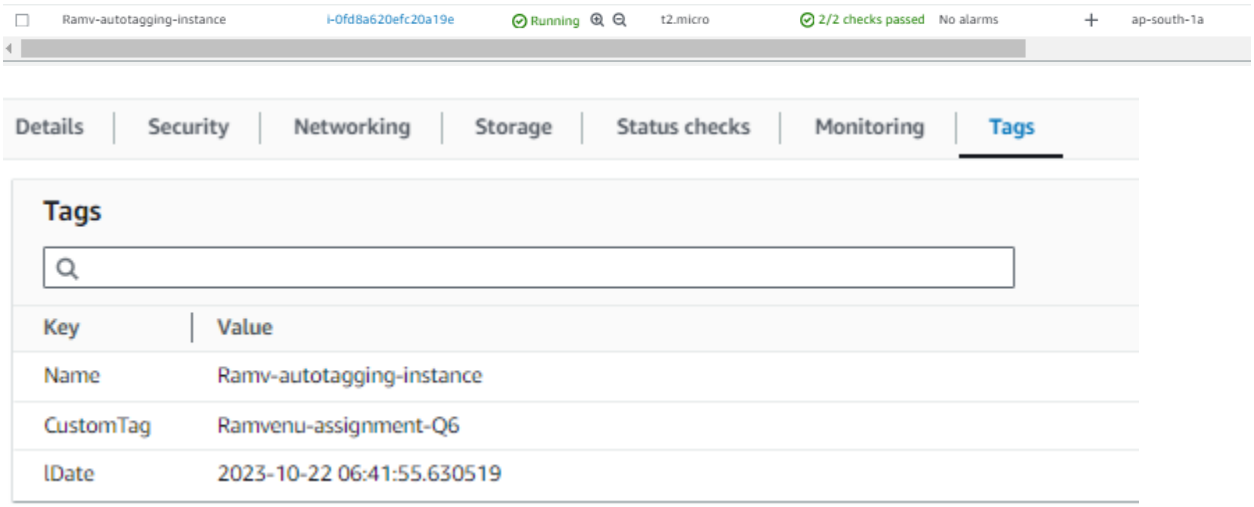


5. Testing:

- Launch a new EC2 instance.

- After a short delay, confirm that the instance is automatically tagged as specified.

Screenshot: Tags successfully updated



Assignment 15: Monitor EC2 Instance State Changes Using AWS Lambda, Boto3, and SNS

Objective: Automatically monitor changes in EC2 instance states and send notifications whenever an instance is started or stopped.

Task: Set up a Lambda function that listens to EC2 state change events and sends SNS notifications detailing the state changes.

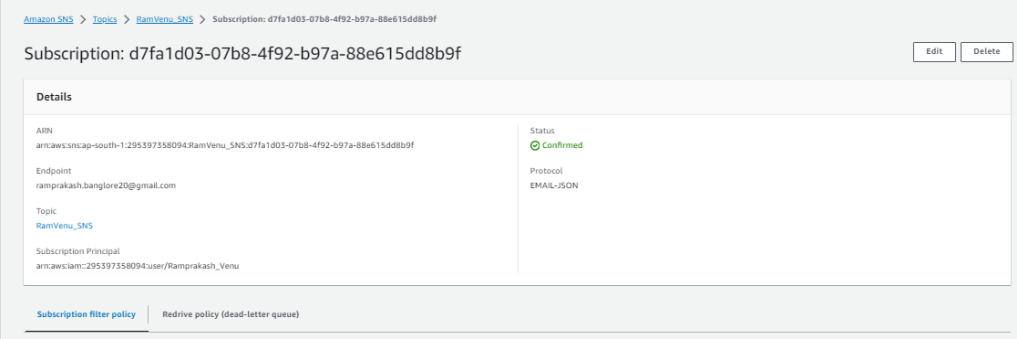
Instructions:

1. SNS Setup:

- Navigate to the SNS dashboard and create a new topic.

- Subscribe to this topic with your email.

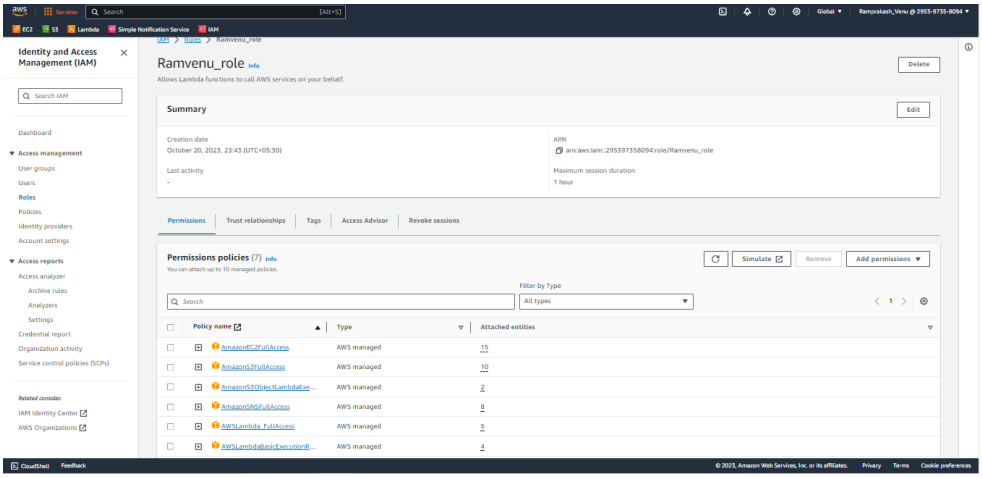
Screenshot : SNS dashboard



2. Lambda IAM Role:

- Create a role with permissions to read EC2 instance states and send SNS notifications.

Screenshot: IAM Role created with all the access required to complete the tasks. (one role for all assignment questions)



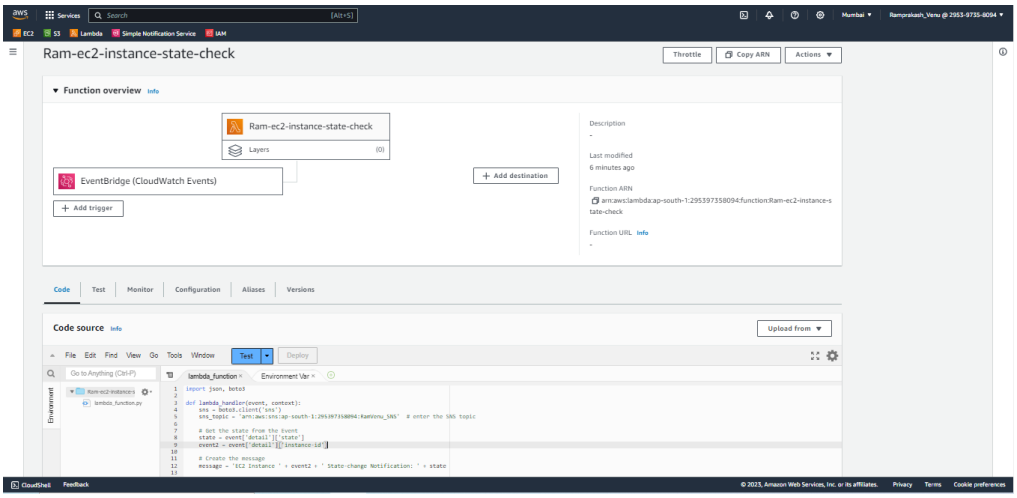
3. Lambda Function:

- Create a function and assign the above IAM role.

- Use Boto3 to:

1. Extract details from the event regarding the EC2 state change.
2. Send an SNS notification with details about which EC2 instance changed state and the new state (e.g., started, stopped).

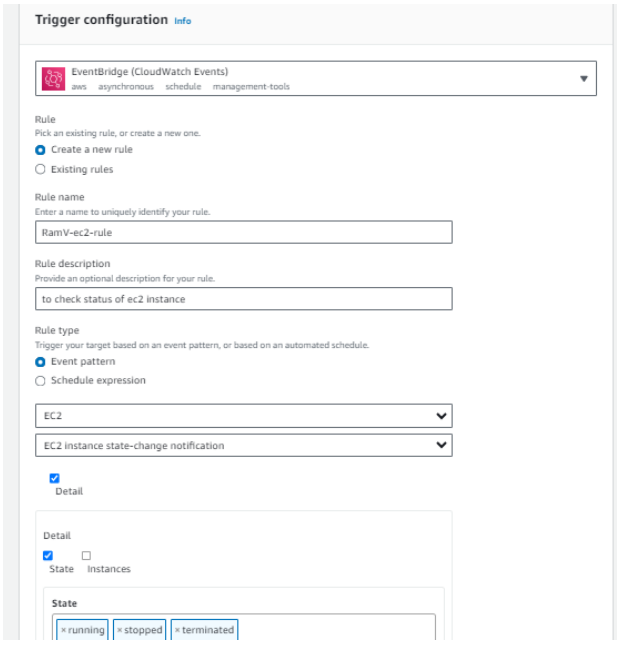
Screenshot: Lambda Function

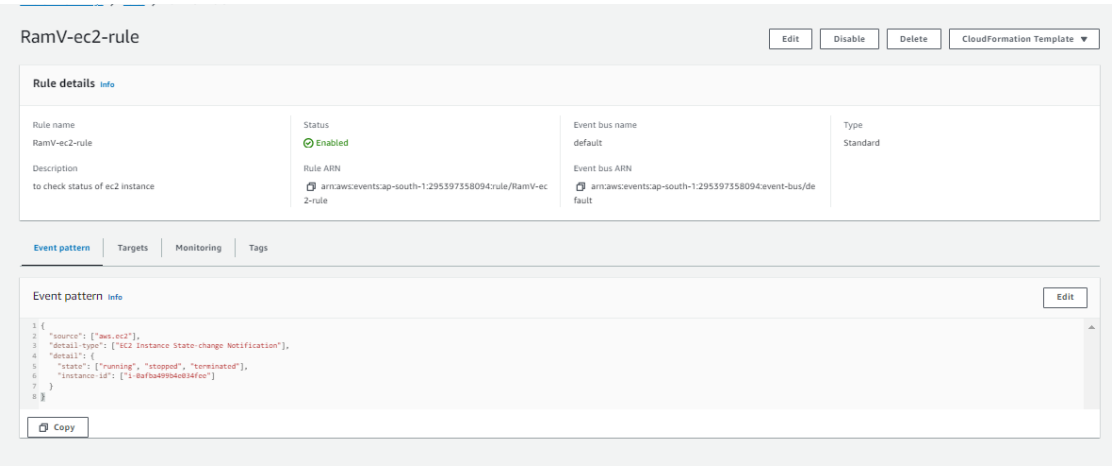


4. EC2 Event Bridge (formerly Cloud Watch Events):

- Set up an Event Bridge rule to trigger your Lambda function whenever an EC2 instance state changes.

Screenshot: EventBridge rule configuration



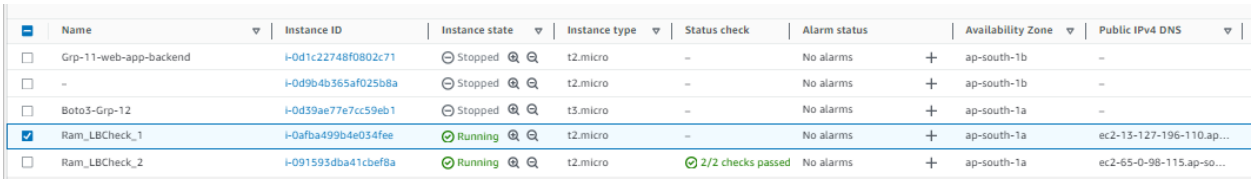


5. Testing:

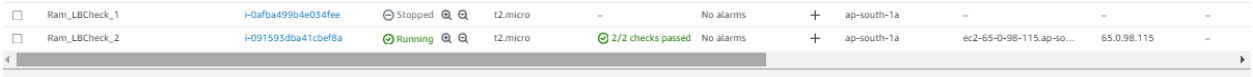
- Start or stop one of your EC2 instances.

- Confirm you receive an SNS notification about the state change.

Screenshot: State of Ec2 instance Ram_LBcheck_1 before testing



Screenshot: State changed from running to stopped.



Screenshot: Notified through email about state change an related information