

Comprehensive Guide for EC2 Backup and Notification System Using AWS Services

Contents

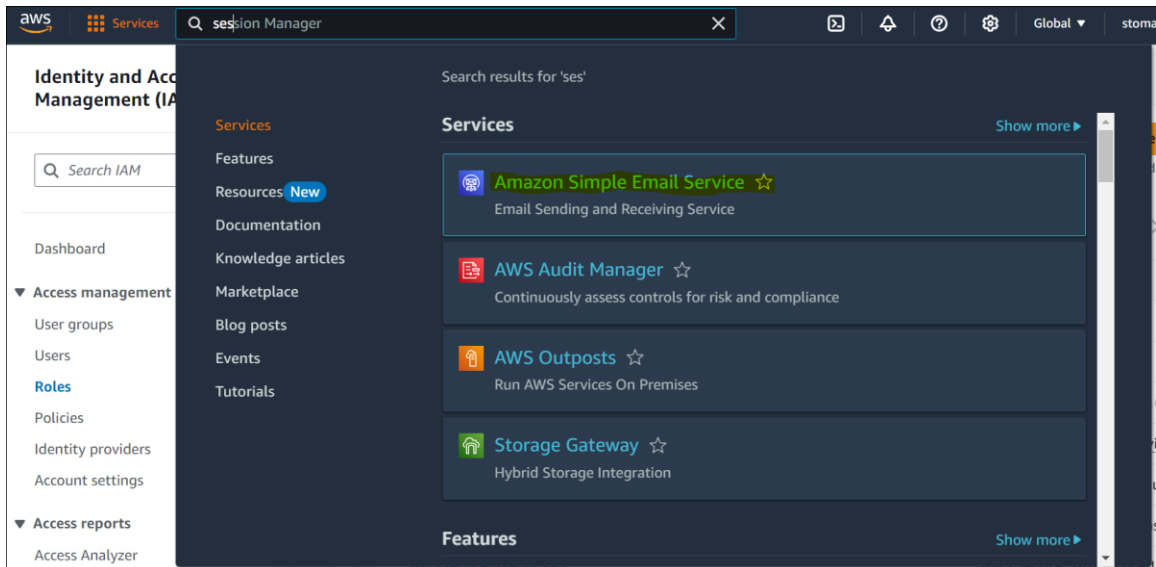
1. Setting up AWS SES for Sending Emails
2. Configuring AWS Backup for EC2 (Optional if no backup policy exists)
3. IAM Roles and Permissions
4. Create an Event Bridge Rule for Lambda Trigger
5. Creating the Lambda Function
6. Lambda Code for EC2 Backup Status and Email Notification

1. Setting up AWS SES for Sending Emails

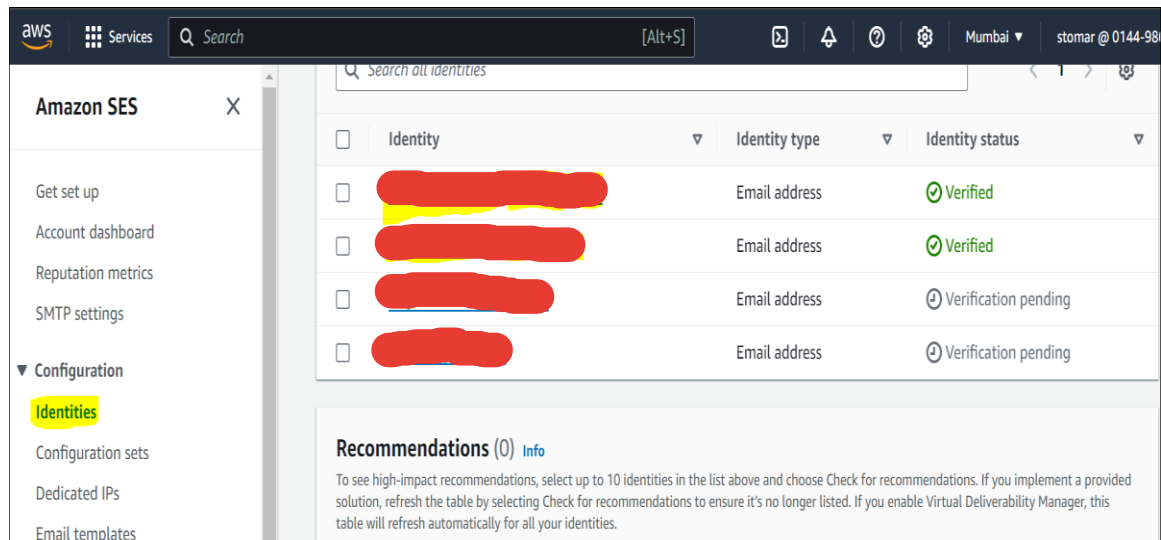
Before sending email notifications, AWS Simple Email Service (SES) needs to be configured.

Steps to Set up AWS SES:

1. Navigate to SES: Go to the SES console by searching for "SES".



2. Verify Email Address: Click Email Addresses under Identity Management and Verify a New Email Address. Enter and verify the sender and recipient email addresses.

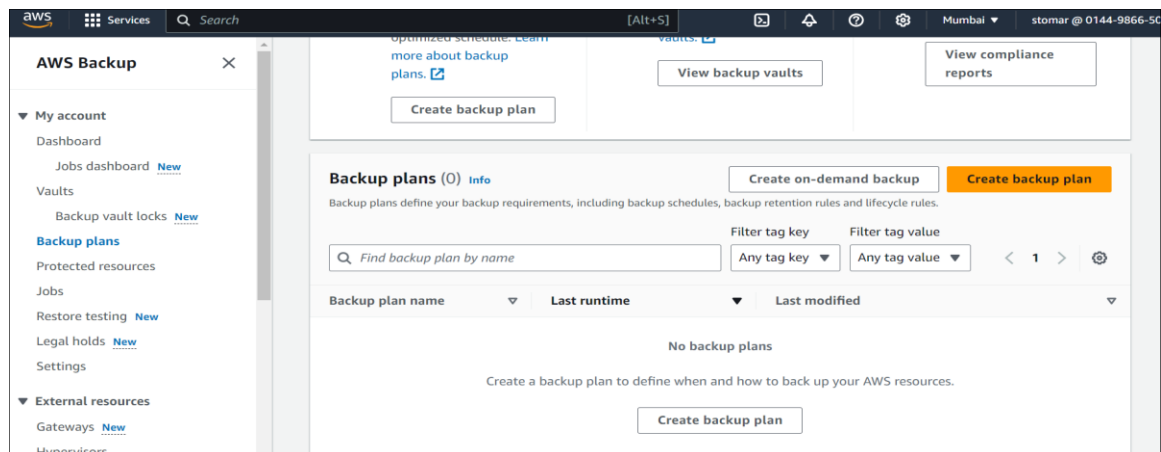


2. Request Production Access: If needed, submit a request to move SES out of the sandbox to send emails to unverified addresses.

2. Configuring AWS Backup for EC2 (Optional if no backup policy exists)

Steps to Set Up AWS Backup for EC2:

1. Go to AWS Backup Console.



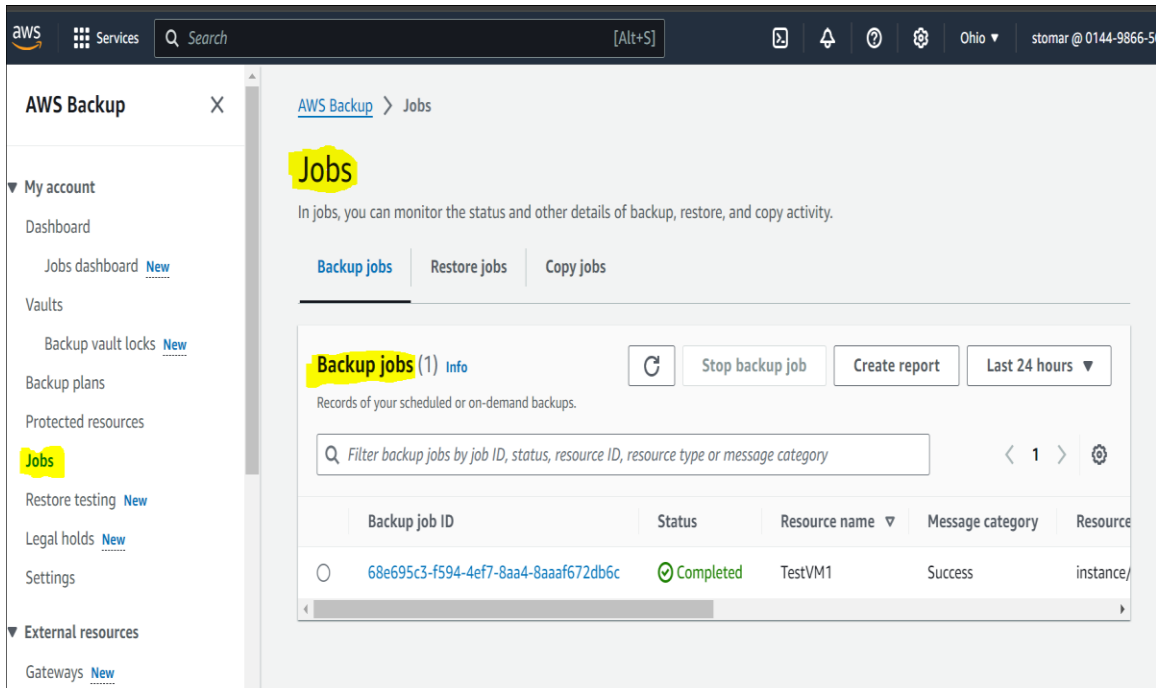
2. Create a Backup Plan: Name it, set the backup frequency (e.g., daily), and retention period (e.g., 7 days). Choose EC2 as the resource type.

The screenshot shows the 'Create Backup Plan' page in the AWS Backup console. The 'Backup plan options' section has three radio buttons: 'Start with a template', 'Build a new plan' (which is selected and highlighted with a yellow box), and 'Define a plan using JSON'. Below this is a text input field for 'Backup plan name' with a note: 'Backup plan name is case sensitive. Must contain from 1 to 50 alphanumeric or '-' characters.' There is also a section for 'Tags added to backup plan - optional'. The 'Backup rule configuration' section is partially visible, showing a 'Schedule' section with a 'Backup rule name' input field.

3. Assign Resources: Choose your EC2 instance by selecting its ID.

The screenshot shows the 'Assign Resources' page in the AWS Backup console. The left sidebar shows the 'AWS Backup' menu with 'Backup plans' highlighted. The main content area has a heading 'Assign resources to this Backup plan using tags and resource IDs.' and three steps: 1. 'Define resource selection' (with a note 'Protect all resources or specify resources by type or ID.'), 2. 'Select specific resource types' (with a note 'Choose specific resource types that you want to protect with this backup plan. You can also exclude specific resource IDs from the selection.'), and 3. 'Exclude specific resource IDs from the selected resource types - optional'. In step 2, the 'Include specific resource types' radio button is selected and highlighted. Below it, a dropdown menu shows 'EC2' as the selected resource type, and a text input field contains 'Instance IDs'. A 'Remove' button is also visible.

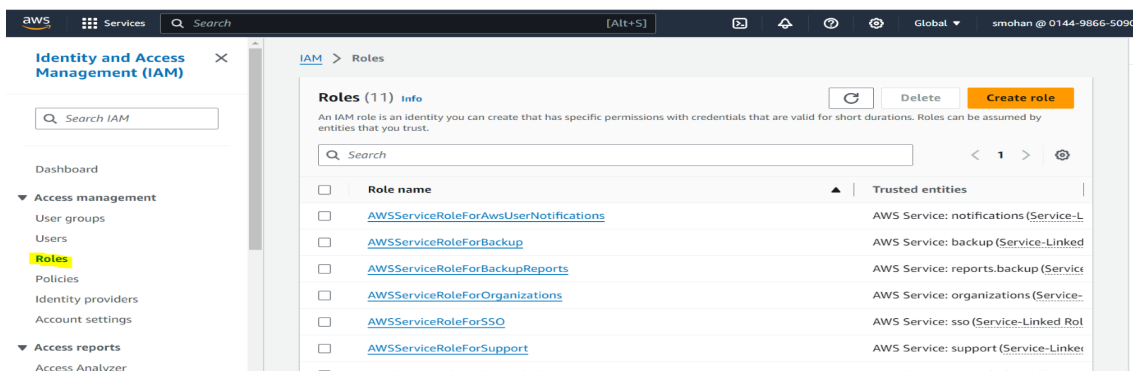
4. Monitor Backups: The backups will automatically start based on the schedule defined in the plan. You can monitor the backups under Backup Jobs in the Backup console.



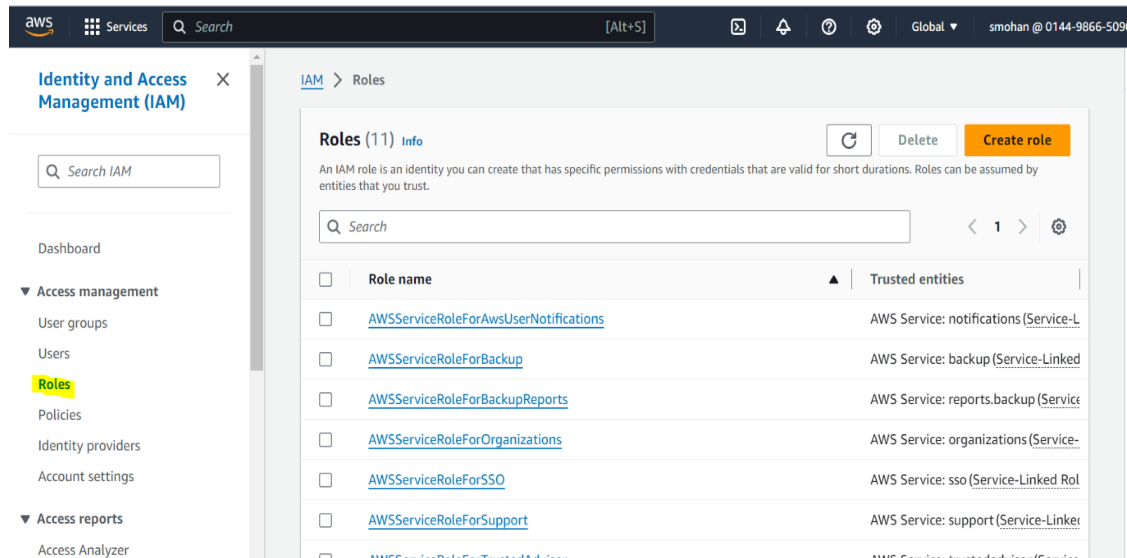
3. IAM Roles and Permissions

Steps to Create an IAM Role for Lambda:

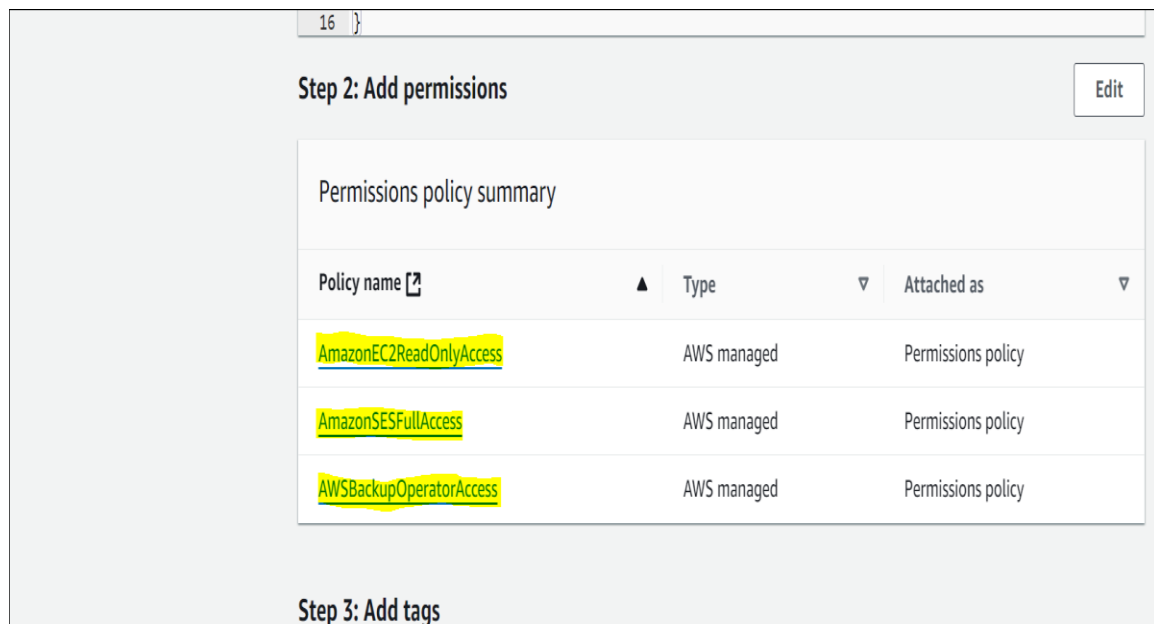
1. Go to IAM Console.



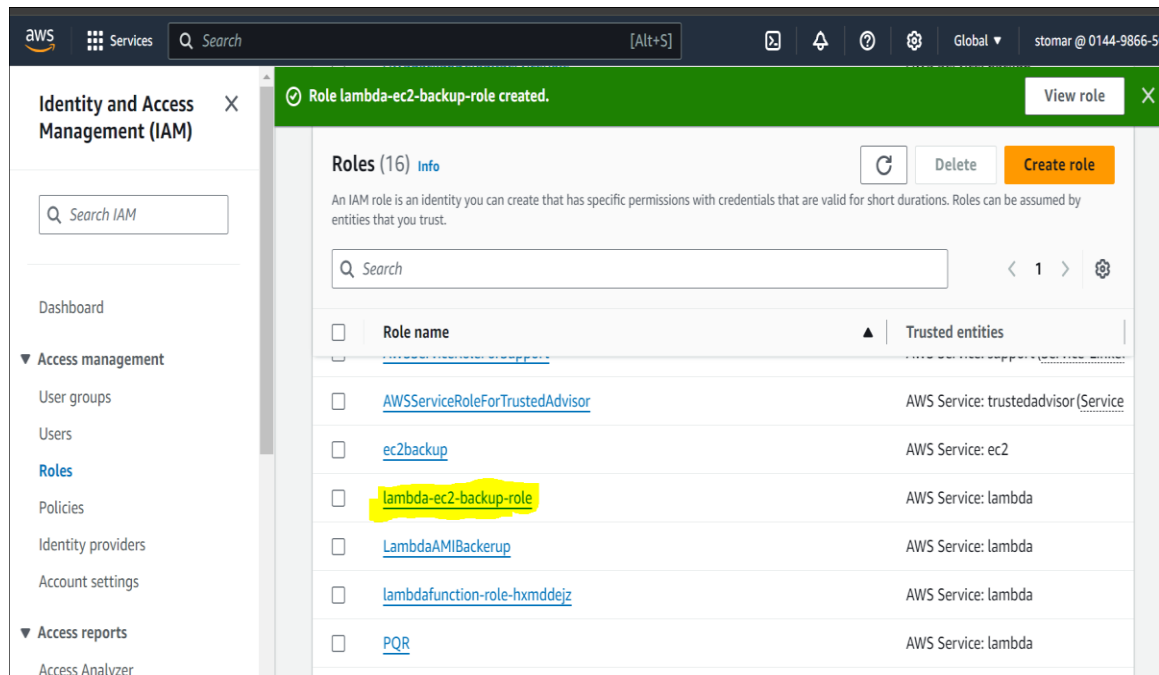
2. Create a New Role: Choose AWS Service as the trusted entity and select Lambda.



3. Attach Policies: Attach AmazonEC2ReadOnlyAccess, AWSBackupOperatorAccess, and AmazonSESFULLAccess policies.



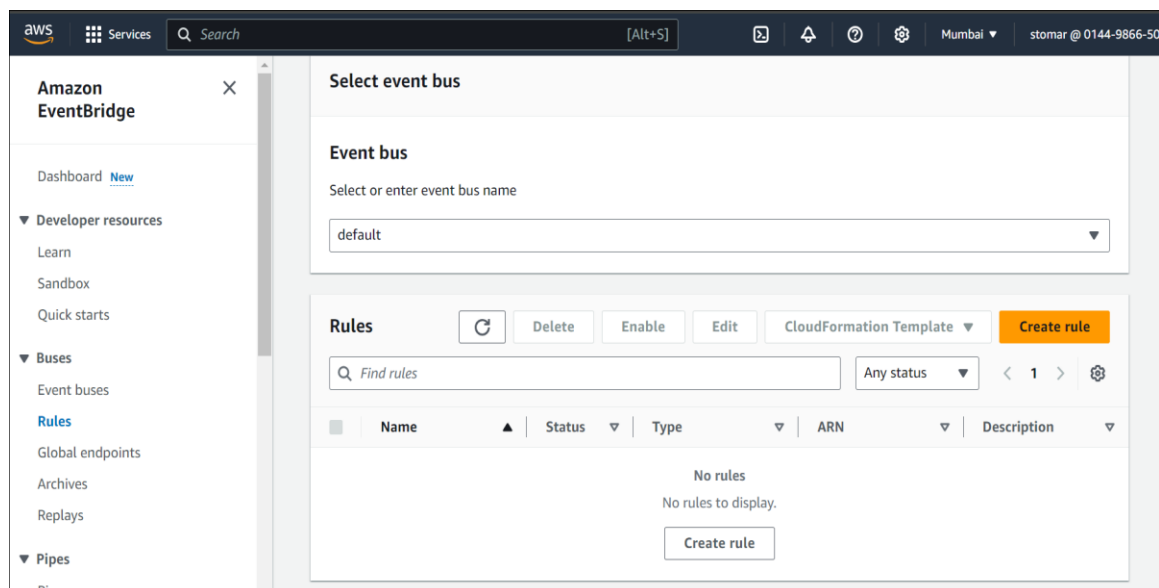
4. Name the Role: Name the role (e.g., lambda-ec2-backup-role).



4. Create EventBridge Rule for Lambda Trigger

Steps to Set Up EventBridge Rule:

1. Go to EventBridge Console.



2. Create a New Rule: Name it, choose Event Source and AWS Services with Backup as the service name, or schedule a daily cron trigger (e.g., cron(0 8 * * ? *)).

The screenshot shows the 'Define rule detail' page in the Amazon EventBridge console. The breadcrumb navigation at the top reads 'Amazon EventBridge > Rules > Create rule'. On the left, a sidebar lists five steps: Step 1 (Define rule detail), Step 2 (Build event pattern), Step 3 (Select target(s)), Step 4 - optional (Configure tags), and Step 5 (Review and create). The main content area is titled 'Define rule detail' with an 'Info' link. Under the 'Rule detail' section, there is a 'Name' field with the placeholder 'rule-name' and a note: 'Maximum of 64 characters consisting of numbers, lower/upper case letters, -, _, .'. Below this is a 'Description - optional' field with the placeholder 'Enter description'. The 'Event bus' section has an 'Info' link and a dropdown menu currently set to 'default', with a note: 'Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.' Below the dropdown is a checked radio button labeled 'Enable the rule on the selected event bus'. The 'Rule type' section has an 'Info' link and two options: 'Rule with an event pattern' (selected with a radio button) and 'Schedule' (unselected with a radio button).

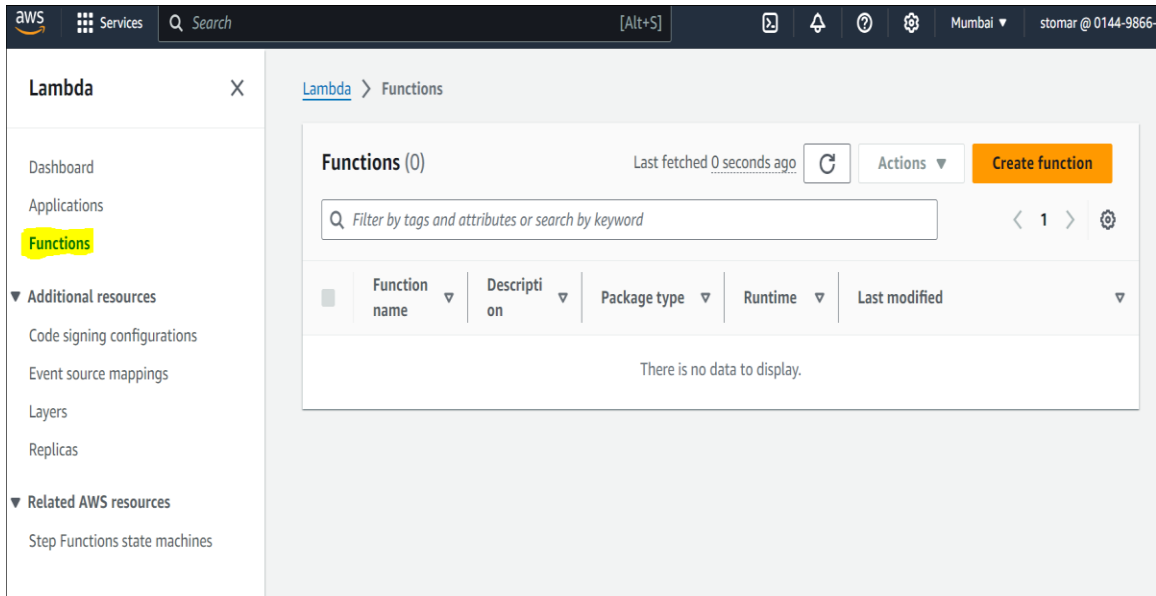
3. Set Lambda as the Target: Select the Lambda function as the target.

The screenshot shows the 'Select target(s)' page in the Amazon EventBridge console. The breadcrumb navigation at the top reads 'aws > Services > Search > [Alt+S]'. The left sidebar shows the same five steps as the previous screenshot, with Step 3 (Select target(s)) being the active step. The main content area is titled 'Target 1'. A 'Permissions' box at the top contains a note: 'Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure the proper permissions.' Below this, the 'Target types' section has a note: 'Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.' There are three radio button options: 'EventBridge event bus', 'EventBridge API destination', and 'AWS service' (which is selected). Below the radio buttons is a 'Select a target' section with an 'Info' link and a note: 'Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule)'. A dropdown menu is open, showing 'Lambda function' as the selected option. Below the dropdown is a 'Function' section with a text input field labeled 'Enter the Lambda function ARN' and a 'G' icon. At the bottom, there is a label 'Function ARN'.

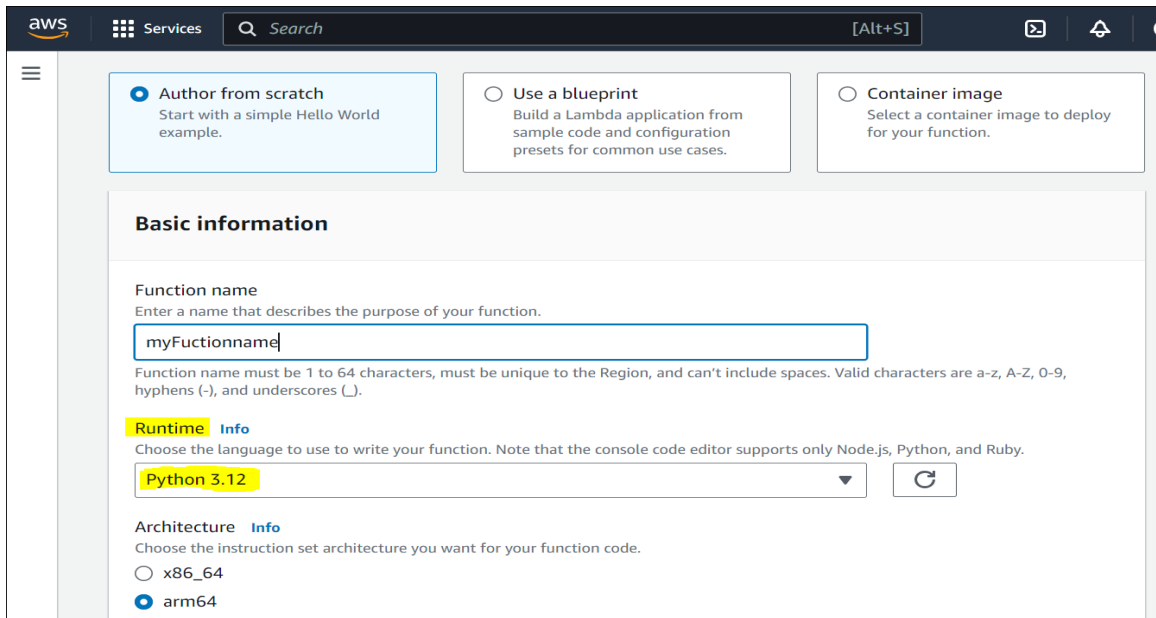
5. Creating the Lambda Function

Steps to Create Lambda Function:

1. Go to Lambda Console.



2. Create Lambda Function: Name it, set Python 3.9 (or higher) as the runtime, and choose the IAM role.



3. Set Timeout and Memory: Adjust timeout and memory as needed.

Lambda > Functions > myFunctionname > Edit basic settings

Edit basic settings

Basic settings Info

Description - optional

Memory Info
Your function is allocated CPU proportional to the memory configured.

128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#).

None

Supported runtimes: Java 11, Java 17, Java 21.

Timeout

0 min 3 sec

4. Deploy Code: Copy the Lambda code (in the next section) into the Lambda editor.

BWS Services Search [Alt+S] Mumbai stammar @ 0144-9886

Code source Info Upload from

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Go to Anything (Ctrl-P)

Environment

myfunctionname

lambda_function.py

```
1 import boto3
2 import json
3 from datetime import datetime, timezone
4 from botocore.exceptions import ClientError
5
6 # Initialize AWS clients
7 backup_client = boto3.client('backup', region_name='ap-south-1')
8 ec2_client = boto3.client('ec2', region_name='ap-south-1')
9 ses_client = boto3.client('ses', region_name='ap-south-1')
10
11 SENDER = "Your mail"
12 RECIPIENT = "Your mail"
13
14 def send_email(subject, body):
15     try:
16         response = ses_client.send_email(
17             Source=SENDER,
18             Destination={"ToAddresses": [RECIPIENT]},
19             Message={"Subject": {"Data": subject, "Charset": "UTF-8"}, "Body": {"Html": {"Data": body, "Charset": "UTF-8"}}})
20         print(f"Email sent! Message ID: {response['MessageId']}")
21     except ClientError as e:
22         print(f"Failed to send email: {e.response['Error']['Message']}")
23         raise
24
25
```

Output :-

Hello Team,

This is an autogenerated report that shows AWS VMs Backup status.

VM Name	Backup Status	Resource ID	Resource Type	Message Category
TestVM2	COMPLETED	i-0bde6586122055223	EC2	N/A
TestVM1	COMPLETED	i-0cf9b9aea56cd2824	EC2	N/A

Thanks and Regards

Team Azure