

Automation and Scaling

It is important to backup monthly, weekly or daily to protect the data inside EC2 instances. For this, our goal is to ensure that snapshots are taken every day within the project.

AWS Lambda is a service that allows you to run code without dealing with server management or infrastructure. Lambda can interact with event triggers (e.g. file uploads, data updates, scheduled tasks) and automatically run the specified code when these events occur.

After navigating to the lambda service in the AWS Management console, select the "Create Function" option to create a new function. Select the "Author from Scratch" option in the service. This refers to the creation of an AWS Lambda function starting from scratch. This approach means starting a Lambda function with a blank state and adding its code step by step. Then name the function, select the desired programming language in the "Runtime" option (Python 3.7 will be used in the project). In the "Change Default Execution Role" tab, select "Create a new role with basic Lambda permissions" option (This option will be changed in the future. Then create function by selecting create function option.

Create function [Info](#)

Choose one of the following options to create your function.

☒ 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.

☐ Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

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

Use only letters, numbers, hyphens, or underscores with no spaces.

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 3.7

⌵

🔄

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

🕒 Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named Takes-Snapshot-Nginx-Server-role-1o0guhgy, with permission to upload logs to Amazon CloudWatch Logs.

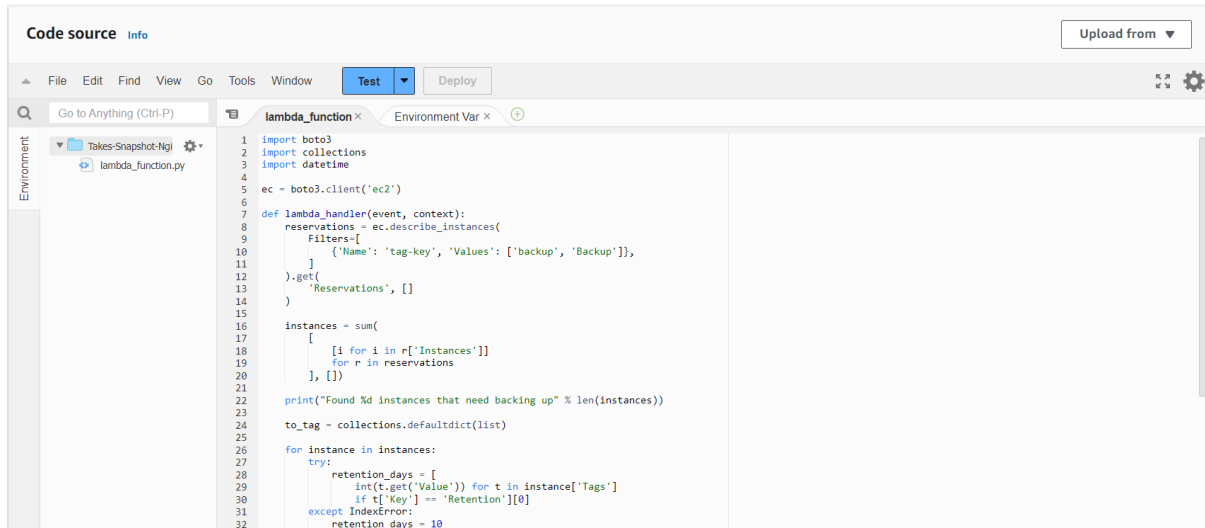
► Advanced settings

Cancel

Create function

A code snippet is written in Lambda. Below you can find the github link to the code. This code allows the ec2 instance to take a snapshot. By selecting the Deploy option, it deploys the AWS Lambda function created in the local development environment of the code to the AWS Lambda service and propagates the updates realized.

<https://github.com/mdurmus99/Lambda-Function>



The screenshot shows the AWS Lambda console's 'Code source' tab. The interface includes a menu bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window'. Below the menu is a toolbar with 'Test' and 'Deploy' buttons. The main area displays the code for 'lambda_function.py'. The code imports boto3, collections, and datetime, then defines a lambda_handler function that interacts with the EC2 service to describe instances and manage snapshots. The code is as follows:

```
1 import boto3
2 import collections
3 import datetime
4
5 ec = boto3.client('ec2')
6
7 def lambda_handler(event, context):
8     reservations = ec.describe_instances(
9         Filters=[
10             {'Name': 'tag-key', 'Values': ['backup', 'Backup']},
11         ],
12     ).get(
13         'Reservations', []
14     )
15
16     instances = sum(
17         [
18             [i for i in r['Instances']]
19             for r in reservations
20         ], []
21     )
22
23     print("Found %d instances that need backing up" % len(instances))
24
25     to_tag = collections.defaultdict(list)
26
27     for instance in instances:
28         try:
29             retention_days = [
30                 int(t.get('Value')) for t in instance['Tags']
31                 if t['Key'] == 'Retention'
32             ][0]
33         except IndexError:
34             retention_days = 10
```

Select the Test option and proceed to the code testing phase. After specifying the name, the test is saved by selecting the "Save" option. If you want to try this test it will not work. Because lambda function does not have access to ec2 instances.

Creating AWS IAM Role

AWS Identity and Access Management (IAM) is a service that provides identity-based access and authorization management for users, groups and roles in Amazon Web Services (AWS) services.

To create a new role via the IAM console, select the "Create Role" option in the "Roles" tab. On the "Select trusted entity" page, select "AWS Services" option in the "Trusted entity type" tab and "Lambda" option in the "Use case" tab.

After selecting the "Create Policy" option on the right side of the "Add permissions" page, the following code snippet is written as JSON and the new Policy is created.

After selecting the Policy created on the Roles page, a new role is created.

- Step 1
Select trusted entity
- Step 2
Add permissions
- Step 3
Name, review, and create

Select trusted entity [info](#)

Trusted entity type

- ☒ **AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- ☐ **AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- ☐ **Web identity**
Allow users federated by the specified external web identity provider to assume this role to perform actions in this account.
- ☐ **SAML 2.0 federation**
Allow 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.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- ☐ **EC2**
Allows EC2 instances to call AWS services on your behalf.
- ☒ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Choose a service to view use case

Cancel

Next

[IAM](#) > [Policies](#) > Create policy

Step 1
Specify permissions

Step 2
Review and create

Specify permissions [info](#)

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

Visual

JSON

Actions

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "ec2:*",
7       "Resource": "*"
8     }
9   ]
10 }
11 }
```

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

+ Add new statement

Step 1
Specify permissions

Step 2
Review and create

Review and create

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

EC2-Full-Access-Policy

Maximum 128 characters. Use alphanumeric and '+', '@', '-', characters.

Description - optional

Add a short explanation for this policy.

Maximum 1,000 characters. Use alphanumeric and '+', '@', '-', characters.

[IAM](#) > [Roles](#) > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Add permissions [info](#)

Permissions policies (Selected 1/880) [info](#)

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

Filter policies by property or policy name and press enter.

<

1

2

3

4

5

6

7

...

44

>

⊗

	Policy name ?	Type	Description
<input type="checkbox"/>	Amazon_EventBridge_Invoke_Action_On_EBS_Volume_1...	Custom...	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-1a065579-063c-49b3-8...	Custom...	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-485ed90c-4c4b-408c-b...	Custom...	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-75639c6c-1640-42b2-9...	Custom...	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-a38b76de-91b1-404b-a...	Custom...	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-c08be9fe-7529-4106-94...	Custom...	
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-d27f24cc-ba15-44f8-a1...	Custom...	
<input type="checkbox"/>	CloudWatch-Log-Project	Custom...	
<input checked="" type="checkbox"/>	EC2-Full-Access-Policy	Custom...	

IAM > Roles > Create role

Step 1
Select trusted entity

Step 2
Add permissions

Step 3
Name, review, and create

Name, review, and create

Role details

Role name

Enter a meaningful name to identify this role.

Lambda-Roles-Full-Access-Ec2-Instances

Role name is required

Maximum 64 characters. Use alphanumeric and "+,=, @, _" characters.

Description

Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and "+,=, @, _" characters.

Let's go back to the Lambda page. On the page, in the "Configuration" tab, select the "Permissions" option and select the "Edit" option to edit. Select the role we created in the "Existing Role" option at the bottom.

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.

Lambda-Roles-Full-Access-Ec2-Instances

View the Lambda-Roles-Full-Access-Ec2-Instances role

on the IAM console.

The code block is tested and it is confirmed that there is a snapshot named "LIVE-BACKUP" on the snapshot page.

Successfully updated the function Takes-Snapshot-Nginx-Server.

Code source

File Edit Find View Go Tools Window Test Deploy

lambda_function x

Environment Var x

Execution result x

Takes-Snapshot-Ngi

lambda_function.py

Test Event Name

Test-Takes-Snapshot

Response

null

Function Logs

START RequestId: 262348f5-e59c-4e0b-9f78-59e51f3193e5 Version: \$LATEST
Found 1 instances that need backing up
Found EBS volume vol-02389a2944240cc4d on instance i-0d56e82e895ae8826
Retaining snapshot: snap-084b4a9f2de7fe9dc of volume vol-02389a2944240cc4d from instance i-0d56e82e895ae8826 for 10 days
Will delete 1 snapshots on 2023-08-26
END RequestId: 262348f5-e59c-4e0b-9f78-59e51f3193e5
REPORT RequestId: 262348f5-e59c-4e0b-9f78-59e51f3193e5 Duration: 724.44 ms Billed Duration: 725 ms Memory Size: 128 MB Max Memory Used: 86 MB Init Duration: 509.33 ms

Request ID

262348f5-e59c-4e0b-9f78-59e51f3193e5

New EC2 Experience

EC2 Dashboard

EC2 Global View

Snapshots (3)

Owned by me

Search

Recycle Bin

Actions

Create snapshot

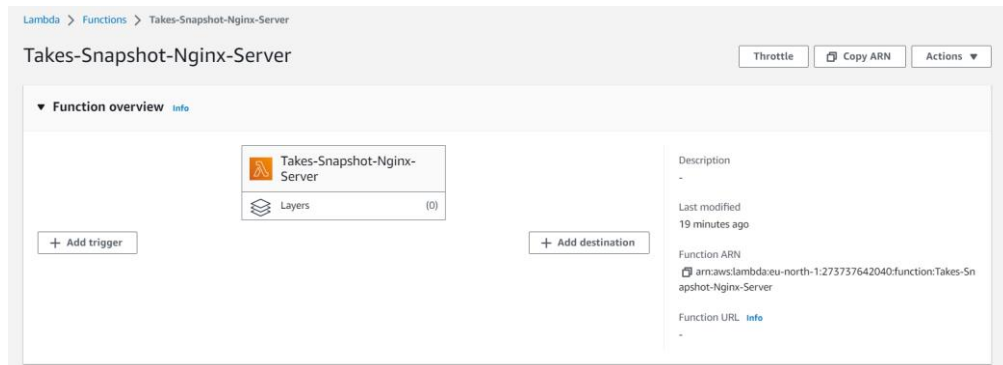
1

<input type="checkbox"/>	Name	Snapshot ID	Volume size	Description	Storage tier	Snapshot status	Started
<input type="checkbox"/>	LIVE-BACKUP	snap-084b4a9f2de7fe9dc	8 GiB	-	Standard	Pending	2023/08/17 00:00 GMT

Automating Work in Cron


AWS EventBridge enables applications and microservices to integrate with event-based architectures, create scheduled transactions and manage events in real time.

An EventBridge rule is created by selecting the "Add Trigger" option on the Lambda service page.



In the "Trigger configuration" tab, select the "EventBridge" option. After adding a rule name and description, select "Rule Type" option as "Schedule expression". In the "Schedule expression" field, type "cron(0 23 * * ? *)". This expression specifies the minute field 0, the hour field 23, any day of the day (every day of every month) and any month. The asterisk symbol (*) represents all values, while the question mark (?) is used for a specific 5ont hor day.

Trigger configuration [Info](#)

 **EventBridge (CloudWatch Events)**
aws asynchronous schedule management-tools

Rule
Pick an existing rule, or create a new one.

☒ Create a new rule

☐ Existing rules

Rule name
Enter a name to uniquely identify your rule.

EventBridge-Takes-Snapshot-Rule

Rule description
Provide an optional description for your rule.

This rule will run the lambda function to take a snapshot every evening at 23:00.

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.

☐ Event pattern

☒ Schedule expression

Schedule expression
Self-trigger your target on an automated schedule using [Cron or rate expressions](#). Cron expressions are in UTC.

cron(0 23 * * ? *)

e.g. rate(1 day), cron(0 17 ? * MON-FRI *)