


1. Create a Lambda function which get triggered from EC2 Action and Notify about changes via SNS Topic.

Trigger configuration

 CloudWatch Events/EventBridge
aws events management-tools

▼

Rule

Pick an existing rule, or create a new one.

Create a new rule ▼

Select or create a new rule

Rule name*

Enter a name to uniquely identify your rule.

ec2-action

Rule description

Provide an optional description for your rule.

action on ec2

Rule type

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

☒ Event pattern

☐ Schedule expression

EC2 ▼

All events ▼

ques-1

Throttle

Qualifiers ▼

Actions ▼

Select a test event ▼

Test


Save


Configuration | Permissions | Monitoring

▼ Designer

 ques-1

 Layers (0)

 CloudWatch Events/EventBridge

 Amazon SNS

+ Add trigger

+ Add destination

2. Create a Lambda function which gets invoked whenever a image is added to a s3 bucket and push the key to SQS.

Trigger configuration



S3
aws storage



Bucket

Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

ques-2-s3



Event type

Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

PUT



Prefix - optional

Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

e.g. images/

Suffix - optional

Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

e.g. .jpg

Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

☒ Enable trigger

Enable the trigger now, or create it in a disabled state for testing (recommended).

Create New Queue

Queue Actions

Filter by Prefix:

Enter Text...

X

K

<input type="checkbox"/>	Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
<input checked="" type="checkbox"/>	s3-image-ques-2.fifo	FIFO	Disabled	0	0	2020-0

1 SQS Queue selected

Details

Permissions

Redrive Policy

Monitoring

Tags

Encryption

Lambda Triggers

Name:

s3-image-ques-2.fifo

URL:

https://sqs.us-east-2.amazonaws.com/835417514122/s3-image-ques-2.fifo

ARN:

arn:aws:sqs:us-east-2:835417514122:s3-image-ques-2.fifo

Created:

2020-04-22 17:35:17 GMT+05:30

Last Updated:

2020-04-22 17:35:17 GMT+05:30

Delivery Delay:

0 seconds

Queue Type:

FIFO

Content-Based Deduplication:

Disabled

Default Visibility Timeout:

30 seconds

Message Retention Period:

4 days

Maximum Message Size:

256 KB

Receive Message Wait Time:

0 seconds

Messages Available (Visible):

0

Messages in Flight (Not Visible):

0

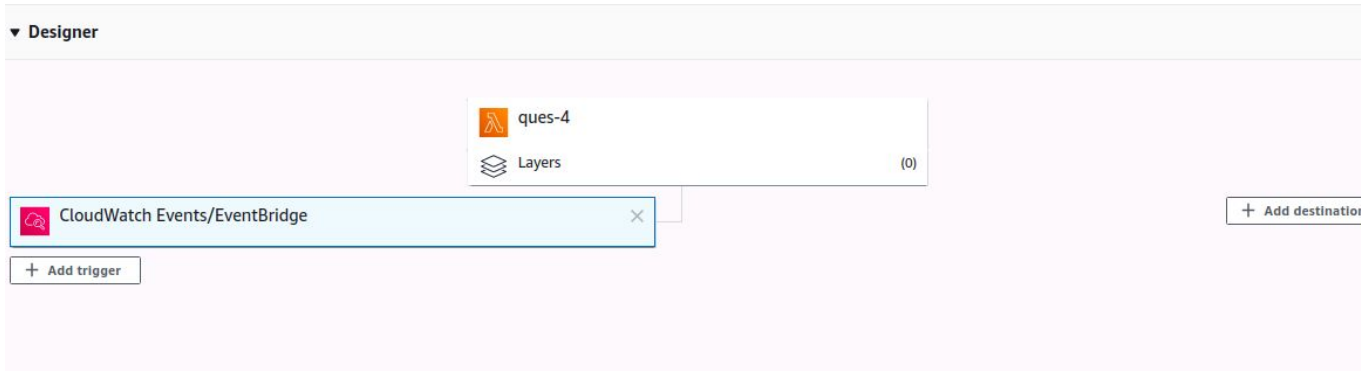
Messages Delayed:

0

3. The SQS should be FIFO and write a Lambda Function which will listen to SQS and compress the image and upload to some other S3 Bucket

Image in s3->ques2 lambda-> sqs -> ques3 lambda->new s3 compressed image

4. Create a Lambda functions which gets triggered daily and takes the AMI of a particular EC2 instance(Filter on the basis of Tag).



```
File Edit Find View Go Tools Window Save Test
ques-4 /
lambda_function.py
lambda_function x
1 import boto3
2 def lambda_handler(event, context):
3     client = boto3.client('ec2')
4     response = client.describe_tags(Filters =[
5         {
6             'Name': 'tag:Name',
7             'Values': [
8                 'ayush-ec2',
9             ]
10        },
11    ],
12    DryRun = False
13    )
14    for instanceId in response['Tags']:
15        res = client.create_image(
16            Description=' lambda AMI',
17            DryRun=False,
18            InstanceId=f'{{instanceId["ResourceId"]}}',
19            Name='Lambda AMI',
20            NoReboot=True,
21        )
22    return res
```

search : i-00201159f3f6dd79a Add filter

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
ayush-ec2	i-00201159f3f6dd79a	t2.micro	us-east-2b	running	Initializing	None

Instance: i-00201159f3f6dd79a (ayush-ec2) Public DNS: ec2-13-58-40-51.us-east-2.compute.amazonaws.com

DescriptionStatus ChecksMonitoringTags

Add/Edit Tags

Key	Value
Name	ayush-ec2

Launch

Actions ▾

Owned by me ▾



Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name ▾	AMI Name ▲	AMI ID ▾	Source ▾	Owner ▾	Visibility
<input type="checkbox"/>		Lambda AMI	ami-0bbddb45dd775d1e9	835417514122/...	835417514122	Private

5. Create a Lambda function which will login to a EC2 instance and prints all the running services. (Use python's paramiko module to do SSH. Also, launch lambda in a VPC).(Keep Keys in S3 and S3 should be encrypted)

ques-5

Overview

Properties

Permissions

Management

Access points

Type a prefix and press Enter to search. Press ESC to clear.

Upload

Create folder

Download

Actions ▾

<input type="checkbox"/>	Name ▾	Last modified ▾
<input type="checkbox"/>	<div>ayush.pem</div>	Apr 22, 2020 9:17:54 PM GMT+0530

Default encryption

Automatically encrypt objects when stored in Amazon S3

[Learn more](#)

 AES-256

```

(ques-5) ayush@ayush:~/ques-5$ ls
bcrypt
bcrypt-3.1.7.dist-info
cffi
cffi-1.14.0.dist-info
_cffi_backend.cpython-36m-x86_64-linux-gnu.so
cryptography
cryptography-2.9.1.dist-info
main.py
nacl
paramiko
paramiko-2.7.1.dist-info
__pycache__
pycparser
pycparser-2.20.dist-info
PyNaCl-1.3.0.dist-info
ques-5-s3.zip
six-1.14.0.dist-info
six.py

(ques-5) ayush@ayush:~/ques-5$ cat main.py
import paramiko
import boto3
def ssh(event,context):
    s3 = boto3.client('s3')
    s3.download_file('ques-5','ayush.pem','/tmp/ayush.pem')
    ayush = paramiko.RSAKey.from_private_key_file('/tmp/ayush.pem')
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    client.connect(hostname='13.58.40.51',username="ubuntu",pkey=ayush)
    stdin,stdout,stderr = client.exec_command('ps -eaf')
    print(stdout.read())
    client.close()

(ques-5) ayush@ayush:~/ques-5$

```

ques-5-s3
Throttle
Qualifiers
Actions
test
Test
Save

Execution result: succeeded (logs)
Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

null

Summary

Code SHA-256
XrUHJypgQtLxJ6dv+uH2GuFxUDPozg7u9koki16BQpQ=

Request ID
c8a5428f-5634-4d0f-81c3-7bfc9197ccd0

Duration
2953.49 ms

Billed duration
3000 ms

Resources configured
128 MB

Max memory used
84 MB Init Duration: 360.21 ms

Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```

d0]\nroot      37      2  0 14:02 ?      00:00:00 [kworker/u31:0]\nroot      38      2  0 14:02 ?      00:00:00 [ecryptfs-kthrea]\nroot      80      2  0 14:02
?      00:00:00 [kthrotld]\nroot      81      2  0 14:02 ?      00:00:00 [nvme-wq]\nroot      82      2  0 14:02 ?      00:00:00 [scsi_ah_0]\nroot      83
2  0 14:02 ?      00:00:00 [scsi_tm_f_0]\nroot      84      2  0 14:02 ?      00:00:00 [scsi_ah_1]\nroot      85      2  0 14:02 ?      00:00:00
[scsi_tm_f_1]\nroot      86      2  0 14:02 ?      00:00:00 [kworker/u30:2]\nroot      90      2  0 14:02 ?      00:00:00 [ipve_addrconf]\nroot      100      2
0 14:02 ?      00:00:00 [kstrp]\nroot      160      2  0 14:02 ?      00:00:00 [kworker/0:1H]\nroot      274      2  0 14:03 ?      00:00:00 [raid5wq]\nroot
324      2  0 14:03 ?      00:00:00 [jbd2/xvda1-8]\nroot      325      2  0 14:03 ?      00:00:00 [ext4-rsv-conver]\nroot      403      2  0 14:03 ?
00:00:00 [kworker/0:2]\nroot      404      1  0 14:03 ?      00:00:00 /lib/systemd/systemd-journald\nroot      406      2  0 14:03 ?      00:00:00
[iscsi_ah]\nroot      408      1  0 14:03 ?      00:00:00 /sbin/lvmetad -f\nroot      412      1  0 14:03 ?      00:00:00 /lib/systemd/systemd-udevd\nroot
413      2  0 14:03 ?      00:00:00 [ib-comp-wq]\nroot      414      2  0 14:03 ?      00:00:00 [ib-comp-unb-wq]\nroot      415      2  0 14:03 ?      00:00:00
[ib_mcast]\nroot      416      2  0 14:03 ?      00:00:00 [ib_nl_sa_wq]\nroot      426      2  0 14:03 ?      00:00:00 [rdma_cm]\nnsystemd+      450      1  0 14:03 ?

```