Linux Academy

Hands-On Training

Creating EBS Snapshots with Lambda

Contents

Introduction

In this lab, we learn how to create a Lambda function that generates snapshots of an EBS volume. We have two EBS volumes, and the function takes snapshots of active instances each time it runs. With this working, we then learn how to schedule events, allowing us to set a rate or cron expression to take a snapshot of our servers at defined intervals.

Log in to your AWS Dashboard with the given credentials.

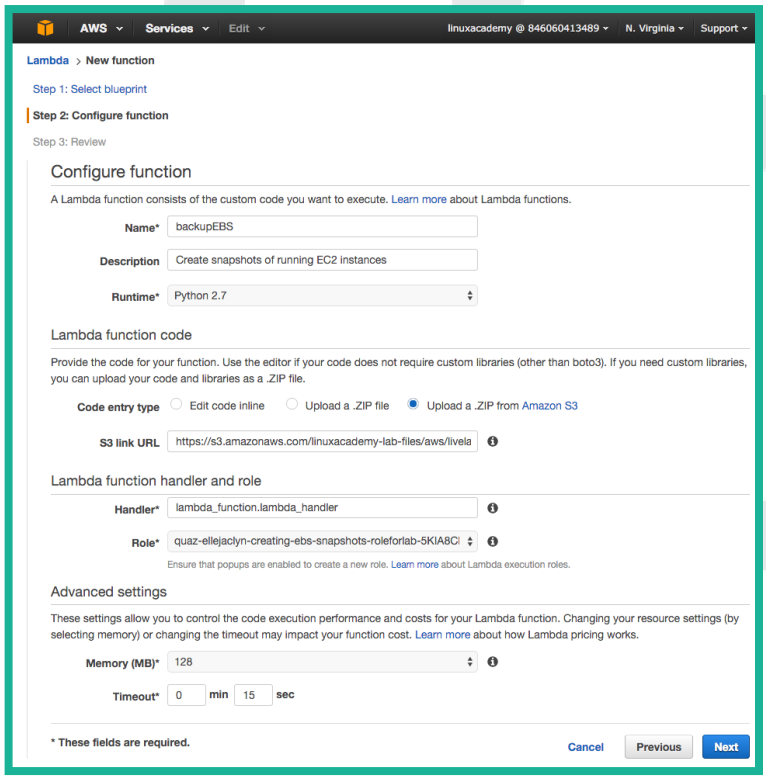
Getting Started

Navigate to the **EC2 Dashboard**. Here, you see two instances, *Stop* and *Keep Running*. You can disregard these titles for now.

Next, view your **Snapshots**. There should be none available, but if there is simply ignore it. From here, navigate to the **Lambda Dashboard**.

Creating a Lambda Function

From the **Lambda Dashboard** (located under **Compute**), select **Get Started Now** to go to the **Select blueprint** page. **Skip** this, and begin instead at the **Configuration function** page.



We gave our function the **Name** *backupEBS*, with the **Description** of *Create snapshots of running EC2 instances*. Because we are using Python, select *Python 2.7* for the **Runtime**.

For this lab, the Lambda function has already been written and uploaded to S3. As such, select *Upload a .ZIP from Amazon S3* as the **Code entry** type. The **S3 URL** is as follows: `https://s3.amazonaws.com/linuxacademy-lab-files/aws/livelabs/backupEBS.zip`

The **Handler** does not need to change but set the **Role** to the *option in the drop-down menu under* ***Use existing role***. This is different for every person; you should see your username.

Set the **Timeout** to *15 sec*, and then press **Next**. Review your options, and select **Create function**.

From here, you can go to **Actions**, then select *Download function code* so that we can review the

function. Unzip the files, take note of the `pytz` dependency, then open the `.py` file using your preferred text editor. The function code is also provided below:

```
1  import boto3
2  import datetime
3  import pytz
4
5  ec2 = boto3.resource('ec2')
6
7  def lambda_handler(event, context):
8     print("\n\nAWS snapshot backups starting at %s" % datetime.datetime.now())
9     instances = ec2.instances.filter(
10       Filters=[{'Name': 'instance-state-name', 'Values': ['running']}])
11
12    for instance in instances:
13       instance_name = filter(lambda tag: tag['Key'] == 'Name', instance.tags)[0]['Value']
14
15       for volume in ec2.volumes.filter(Filters=[{'Name': 'attachment.instance-id', 'Values':
[instance.id]}]):
16          description = 'scheduled-%s.%s-%s' % (instance_name, volume.volume_id,
17             datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
18
19       if volume.create_snapshot(VolumeId=volume.volume_id, Description=description):
20          print("Snapshot created with description [%s]" % description)
21
22       for snapshot in volume.snapshots.all():
23          retention_days = 15
24          if snapshot.description.startswith('scheduled-') and ( datetime.datetime.
now().replace(tzinfo=None) - snapshot.start_ti   me.replace(tzinfo=None) ) > datetime.
timedelta(days=retention_days):
25             print("\t\tDeleting snapshot [%s - %s]" % ( s.snapshot_id, snapshot.description ))
26             snapshot.delete()
27
28    print("\n\nAWS snapshot backups completed at %s" % datetime.datetime.now())
29    return True
```

The function begins by importing `boto3`, `datetime`, and `pytz`. Line **5** defines an ec2 resource, used to filter through our instances.

The `lambda_handler` begins on line **7**; line **8** simply prints that the AWS snapshot has begun, and provides the time in which it started. Line **9** goes through our EC2 instances and filters them by whether or not the instance is running. This then begins a loop (**12**) to pull each instance name, filtered by using a Python lambda (not AWS Lambda) function. The function then loops through the instances' volumes (line

15).

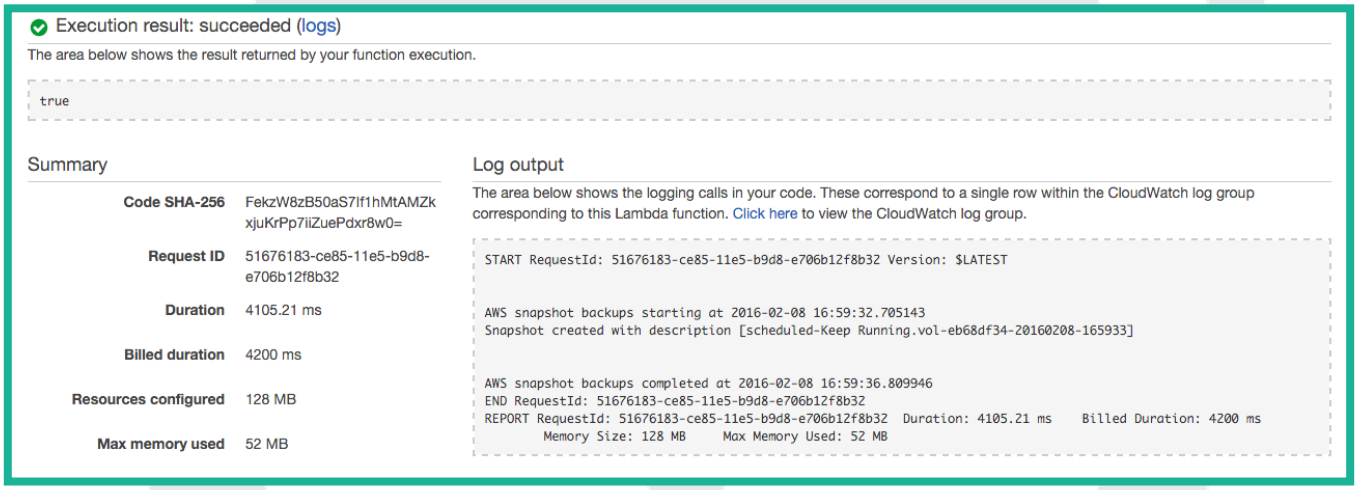
Line **19** begins the process of trying to create a snapshot by taking the volume and performing the `create_snapshot` operation. It prints that the snapshot has been created.

Line **22** begins an optional function that removes snapshots older than fifteen days. This is where `pytz` is used to help with timezones.

Finally, on line **28**, the function prints that the snapshot has been created and the time it was completed.

Testing the Function

From the AWS Lambda Console in which we left off, press the **Test** button. We do not have any events configured yet, so press **Save and test**, leaving the default values. Here we can see that the function returned *true*, as expected. The **Log output** also shows the printed items included in the code. Two snapshots should be created, one for each volume. Should you navigate to the **EC2 Dashboard** and view **Snapshots**, you should see that these two snapshots are there.



Now, to demonstrate that this code runs only on running instances, go to your instances and stop the server named *Stop*. Return to your Lambda function, and press **Test** again. You can see that only one snapshot has been created. This confirms that the function is filtered only by running servers.

Creating an Event Function

From the **Lambda Console** for your function, select **Event sources**, then **Add event source**. The **Event source type** is *CloudWatch Events - Schedule*. The give the **Rule name** a designation of *backupEBS*, and set the **description** to *Backs up EBS volumes on a set schedule*. Leave the **Schedule expression set** to *rate(5 minutes)*. Press **Submit**.

Now is the time to take a small break and wait five to ten minutes, before returning to your **EC2 Dashboard** and viewing your snapshots — there should be more than the three we had left it with.

Setting Up a Cron Expression

Remove your previous **Event source**, and add a new *CloudWatch Events - Schefule* event. We gave our rule the **name** of *ebs-set-schedule*, with a **Rule description** of *Snapshots every week*. The **Schedule expression** should be set to a cron expression. We set it to run every Sunday at 23:59 with the cron expression of *cron(59 23 ? * SUN *)*. Press **Submit**.

You may now complete your lab!