



Automate EC2 AMI Backup Using Lambda Function

By DevopsAdmin | September 27, 2019

0 Comment



If you are Working on AWS you must be working with EC2. Creating and managing Your Application On EC2 Would definitely require to have a backup strategy. Its quite annoying to take the backup of Selected Servers on Regular Basis or in some interval of time. Taking backup is Important but managing the retention time of the backup is also important from the cost prospective

In this blog, We will see how we can take the AMI Backup of an EC2 Instance with the user defined Retention period. We are going to use lambda function with **python2.7** as a runtime to completely automate this process. We Will be covering the following,

- **AMI Backup of an EC2 Instance**
- **Older Ami backup Deletion**
- Scheduling the Backup Using Cloudwatch Events
- Pushing Execution Logs to Cloudwatch Logs
- Monitoring the lambda Function Using Cloudwatch metrics

The Logic Which we are going to use is very simple and clear,

Office 365 Era - Managed Services

Ad Expand your managed email service offering

custom.crn.com

Learn more

- We Will Write 2 lambda Function Once for Taking backup and the other one if for deleting the Backup based on the defined retention period.
- Both lambda functions are scheduled to trigger at some time interval using the cloudwatch events.
- The AMI backup Script will Search for all the Instances With Specific tags and initiate a backup for it.

Privacy & Cookies Policy

- Once the backup is done, It Will create a DeleteOn date to the AMI and snapshots depending upon the retention period you define.
- Lambda Script for Ami deletion will check for the current date on Every AMI taken and If finds the same day date then it will delete those AMI's or else ignore.
- This Process can take the backup of multiple EC2 Instances all together and will keep on deleting the Older AMI which no more need to be retained.
- Both the Lambda Function will be triggered using Cloudwatch Events at perticular time.
- We Can also use Cloudwatch for monitoring the lambda Function failure metrics and can be notified in case backup Script fails.

Let's get Started,

Create a lambda IAM Role Which Allow lambda Fuction to Communicate with the Required resources. Please find the Sample [IAM Role In DevOps Github Repository](#).

Note: You can modify the IAM policy to make it more specific in terms of permissions. For this Example I have assigned Full permission for EC2 and Cloudwatch Logs.

Now, Go to lambda and **Click On Create Function**, Next Go ahead and fill the *Basic Information* as Shown in the Screenshot below and **Click Create Function**.

Is Jesus Really God?

Ad Discover the Evidence From Sc
Jesus' Claims to be God

Y-Jesus

Open

Make Sure to choose the Runtime as **python 2.7** and Also Select the lambda Role which you have created above.

Basic information

Function name
Enter a name that describes the purpose of your function.
AmiBackup
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function.
Python 2.7

Permissions [Info](#)
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an 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

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.
AMIBackupDeletionLambdaRole
[View the AMIBackupDeletionLambdaRole role on the IAM console.](#)

Cancel Create function

Once lambda Function Created, grab the [AMI backup Script](#) and paste in there.

Note: We are taking the Amazon Account Number and Retention period from the Environment variable, So make Sure you pass in the Correct Environment Variable.

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#)

AWS_ACCOUNT_ID	XXXXXXXXXXXXXXXX	Remove
RETENTION_DAYS	2	Remove
Key	Value	Remove

► Encryption configuration

Now Verify Whether You have Selected the Correct Role or Not. Give Tag to the Function, Add Descriptions etc. Let memory be **128MB** (default) and **Increase the Timeout Period to say 2 min.** (Depending Upton the number of Instances you have)

Now, **Click Save**

Important:

Most Importantly, Go to the EC2 and Create a Tag On All the Instances Which You wanted to Backup.



Description Status Checks Monitoring **Tags**

Add/Edit Tags

Key	Value	
Backup	True	Show Column
Name	swarm-worker	Hide Column

You can define any tag but you have to change the tag section in the code as well.

Now Create a test event and **Click on Test to Run** the lambda function

Also Check the **Cloudwatch Logs** for the Detailed Execution logs or in case of any errors. The Log will help you to understand what actually is happening.

AmiBackup

Throttle Qualifiers Actions test **Test** Save

✓ Execution result: succeeded ([logs](#))

► Details

Configuration Monitoring

Go to the AMI Section and Check Whether the AMI of all those instances which was tagged for backup happened or not. Also verify the tags on the Created AMI, It will have **DeleteOn Tag** based on the retention period You defined. *Just for an Example*, If the backup taken on 27th and the retention period is 2 days, then the DeleteOn Tag will be 29th. So this AMI Will be deleted by the [AMIDeletion Script](#) on the Same date Which Will keep on maintaining you 2 days of latest backup.

Tags		
Add/Edit Tags		
Key	Value	
Backup	True	Show Column
DeleteOn	09-29-2019	Show Column
Name	swarm-master-Lambda-i-0304d008afd77e4c5-09-27-2019-16-27-28	Hide Column

Similarly, create another lambda function with same runtime and configure the same way and name it as say **AMIDeletion**. The Script for the **AMIDeletion** you can find it from the Official Github Repository of DevOpsAGE.

Note: Make sure to change the region name at line number 15 of the script.

On-Premises Performance -
Cloud Cost Savings

Ad Modernize application infrastru
hybrid cloud services and improve

custom.crn.com

Learn more

Triggering lambda Function using Cloudwatch Events

Now, as we have created a lambda function, It need to be triggered for the execution. We can use the Cloudwatch Events to schedule a trigger.

Go to the **Cloudwatch** and select **Events** and **Click on Get Started**. Select the **Cron Expression** and Schedule to run on Specific time. Target Will be the lambda Functions which we have created.

Click Configure Details.

Give name to the Event and you are all Set. This Event will keeps on triggering lambda functions on time and Instance backup will be automatically taken and rotated.

Step 1: Create rule

Create rules to invoke Targets based on Events happening in your AWS environment.

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

☐ Event Pattern ☒ **Schedule**

☐ Fixed rate of 5 Minutes

☒ **Cron expression** 30 8 * * *

Next 10 Trigger Date(s)

1. Sat, 28 Sep 2019 08:30:00 GMT
2. Sun, 29 Sep 2019 08:30:00 GMT
3. Mon, 30 Sep 2019 08:30:00 GMT
4. Tue, 01 Oct 2019 08:30:00 GMT
5. Wed, 02 Oct 2019 08:30:00 GMT
6. Thu, 03 Oct 2019 08:30:00 GMT
7. Fri, 04 Oct 2019 08:30:00 GMT
8. Sat, 05 Oct 2019 08:30:00 GMT
9. Sun, 06 Oct 2019 08:30:00 GMT
10. Mon, 07 Oct 2019 08:30:00 GMT

[Learn more about CloudWatch Events schedules.](#)

[Show sample event\(s\)](#)

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Lambda function

Function* AmiBackup

Configure version/alias

Configure input

Lambda function

Function* AMIDeletion

Configure version/alias

Configure input

[Add target*](#)

Cancel [Configure details](#)

Note: Cron Expression uses the GMT Time Zone. i.e 8:30 GMT = 4:30 EST

Monitoring Lambda Functions using Cloudwatch Metrics.

Go to the Cloudwatch, **Click On Create Alarm** and Select **Lambda Metrics**

The screenshot shows the AWS CloudWatch Metrics console. The 'All metrics' tab is selected. A search bar at the top allows searching by metric, dimension, or resource ID. Below the search bar, a grid of metric tiles is displayed. The 'Lambda' tile is highlighted with a red box, indicating it has 22 metrics. Other visible tiles include EBS (18 Metrics), EC2 (32 Metrics), Firehose (2 Metrics), Logs (6 Metrics), RDS (80 Metrics), and States (4 Metrics).

Make Sure to select the metrics by function name, then Select the function with **metrics name as Errors**.

The screenshot shows the AWS CloudWatch Metrics console with the 'Graphed metrics (1)' tab selected. The breadcrumb navigation shows 'All > Lambda > By Function Name'. A search bar is present. Below, a table lists metrics for the 'AmiBackup' function. The 'Errors' metric is highlighted with a red box.

FunctionName (8)	Metric Name
<input type="checkbox"/> AmiBackup	Invocations
<input type="checkbox"/> AmiBackup	Throttles
<input type="checkbox"/> AmiBackup	Duration
<input checked="" type="checkbox"/> AmiBackup	Errors
<input type="checkbox"/> AMIDeletion	Invocations

Now Set the Condition like,

if Error is ≥ 1

Under Additional Configuration,

Treat missing data as good (Not breaching the threshold)

Click next, Select the **SNS topic** and Give Proper name to the Alarm, Verify the Setup and **Click Create Alarm**.

That's it, Now if your lambda function fails then you will be notified based on the SNS Topic and you can See the Cloudwatch logs for troubleshooting.

[AMI Backup Script](#)

[AMI Deletion Script](#)

Note: Please make sure you test this setup in the Dev Environment before implementing directly in the production Environment.

If you Like Our Content here at Devopsage, then please support us by sharing this post.

Please Like and follow us at, [LinkedIn](#), [Facebook](#), [Twitter](#), and [GitHub](#)



Also, Please comment on the post with your views and let us know if any changes need to be done.

Thanks!



[Privacy & Cookies Policy](#)

Category: [AWS Cloud](#) [DevOps](#) [Monitoring](#) [Python](#) Tags: [automate ami backup](#), [aws](#), [AWS Lambda](#), [boto3](#), [cloudwatch events](#), [EC2 AMI Backup](#), [lambda function to create ec2 ami backup](#), [python](#), [Python program to take ami backup](#)