Contents

[Objective 2](#_Toc522871561)

[Step-1 Launch EC2 instances 2](#_Toc522871562)

[Step-2 Create an IAM role 2](#_Toc522871563)

[Step-3 Build the Lambda function for app-a snapshots 2](#_Toc522871564)

[Step-4 Observe EBS snapshots 10](#_Toc522871565)

[Step-5 EBS snapshots for app-b 10](#_Toc522871566)

[Step-5 Disable or Delete 11](#_Toc522871567)

Serverless Lambda Use-Case: EBS Snapshots

# Objective

Automate EBS snapshots for EC2 instances tagged with the specified key/value pair

The snapshot execution will be done by a Lambda function that can be run as a CRON job with a CloudWatch scheduled expression as a trigger

# Step-1 Launch EC2 instances

Create two EC2 instances in public subnet within North Virginia

Once the instances are launched, select each instance and in the bottom-window locate the tags tab and Add/Edit to add the below tags

* Add a tag to the **first instance** with a key=**application** & value=**app-a**
* Add a tag to the **second instance** with a key=**application** & value=**app-b**

Basically, you will now have two EC2 instances

1. First EC2 will have the tag with key=application & value=app-a
2. Second EC2 will have the tag with key=application & value=app-b

**Important note:** Ensure that there is no space after you enter the terms application or app-a or app-b. The tags are case sensitive in all manner. The Python code will simply not be able to recognize the tags if there is any mismatch even if it means a space character after the tag key or value

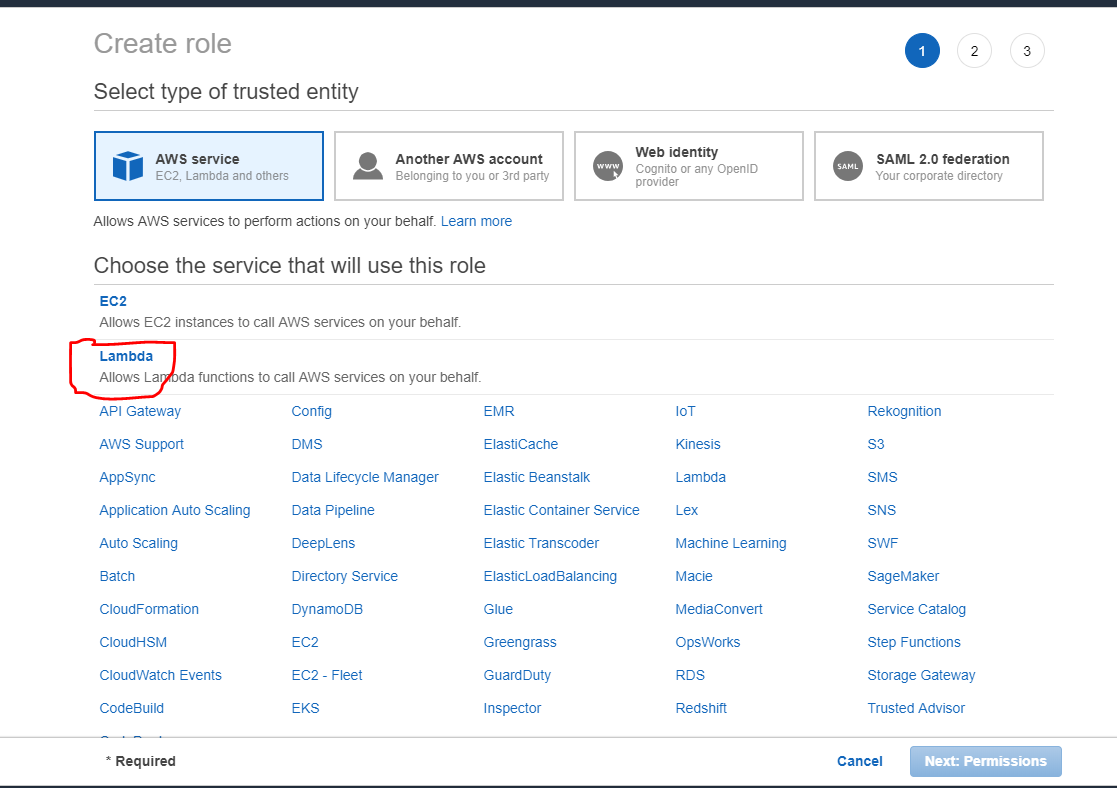
# Step-2 Create an IAM role

Create an IAM role that will be used by the Lambda function

Navigate to services > IAM > roles > create role & select Lambda as the service that will use this role

Click on next permissions > attach a policy directly and select AdministratorAccess as the policy

Save the IAM role as **ebs-lambda-ddmmyyyy** (replace ddmmyyyy with the appropriate values)



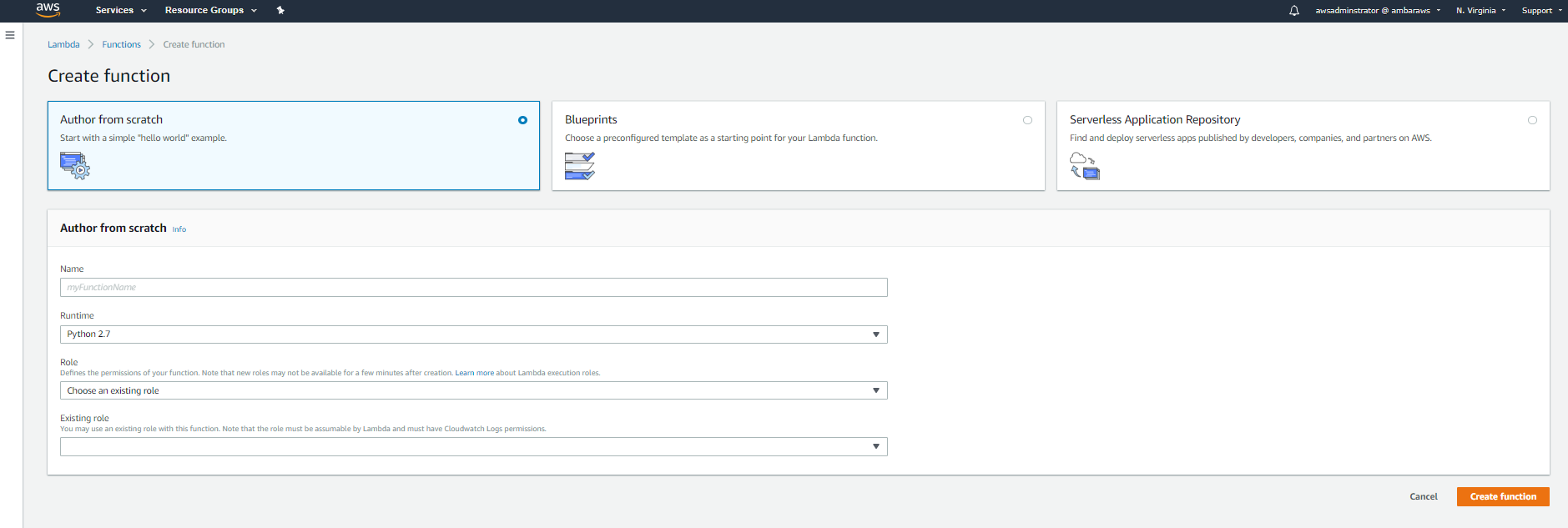
# Step-3 Build the Lambda function for app-a snapshots

You should be in North Virginia region

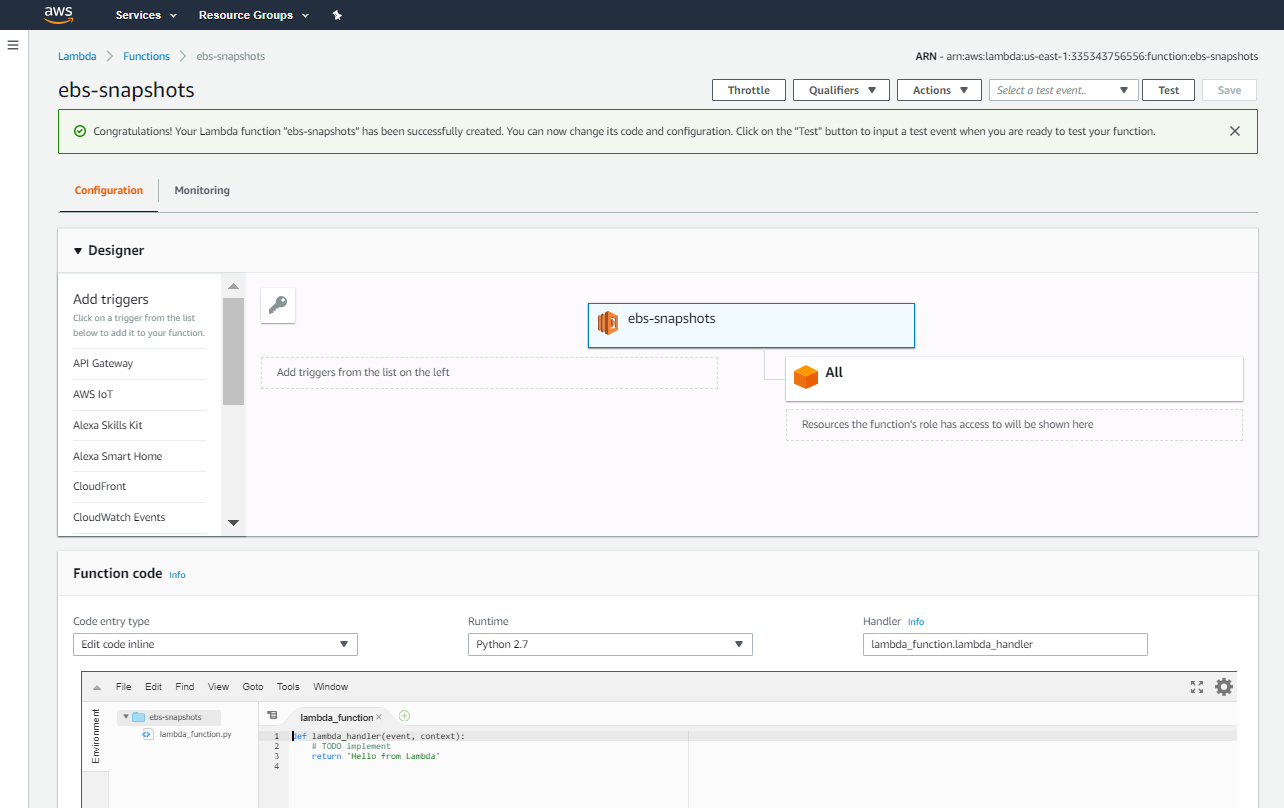
Navigate to services > lambda and chase the orange button that says create function

Keep the default option to create the function, i.e. Author from scratch

* Name: ebs-snapshots
* Runtime: Python 2.7 (note: our function is coded in Python with boto3 package)
* Role: Choose an existing role & select ebs-lambda-ddmmyyyy role. Next create function



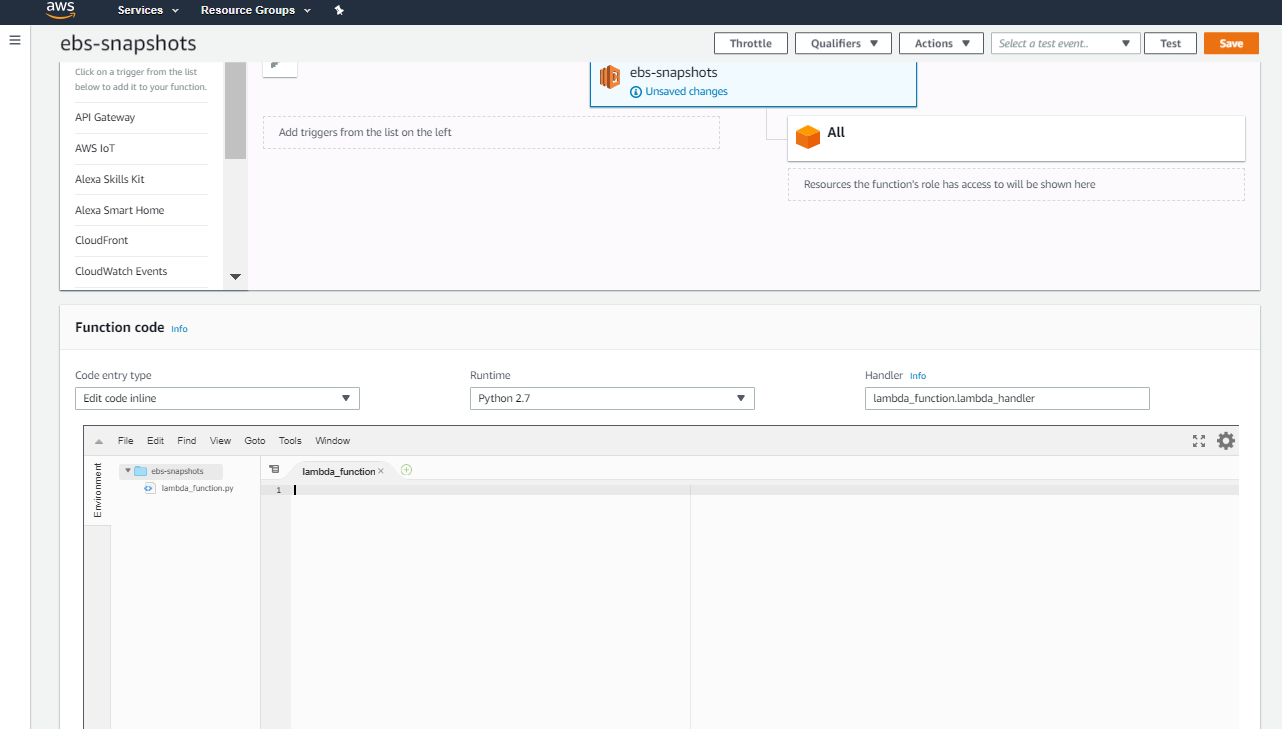
This will create a blank function with a default code.



At this stage, the lambda function has no triggers defined and is populated with a default code.

To start with we will replace the default code with our own code

Delete the default code and your code editor should be a blank space



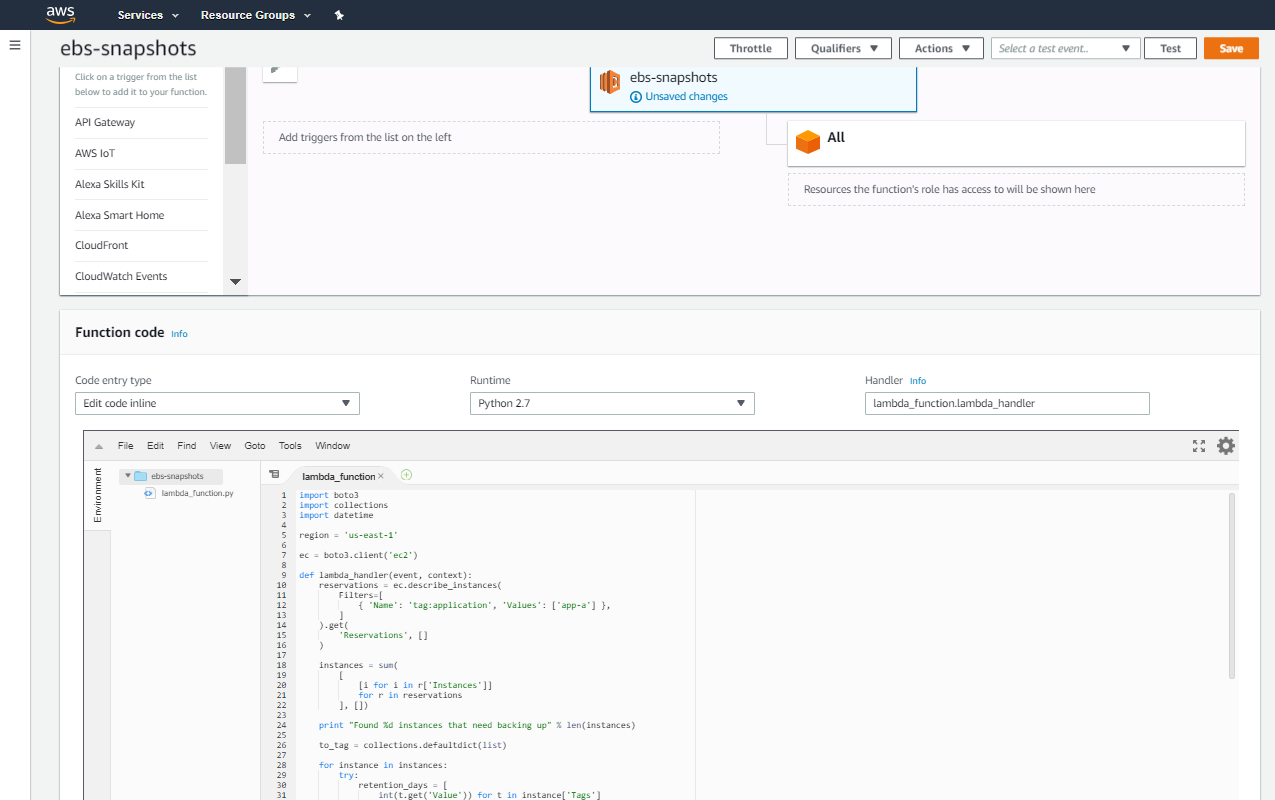
Download the python code from the below URL

<https://s3.amazonaws.com/labs.thecloudgarage.com/lambda/lambda-ebs-snapshots-basic/lambda-ebs-snapshot-basic.py>

Open the code in notepad++

Copy the whole code

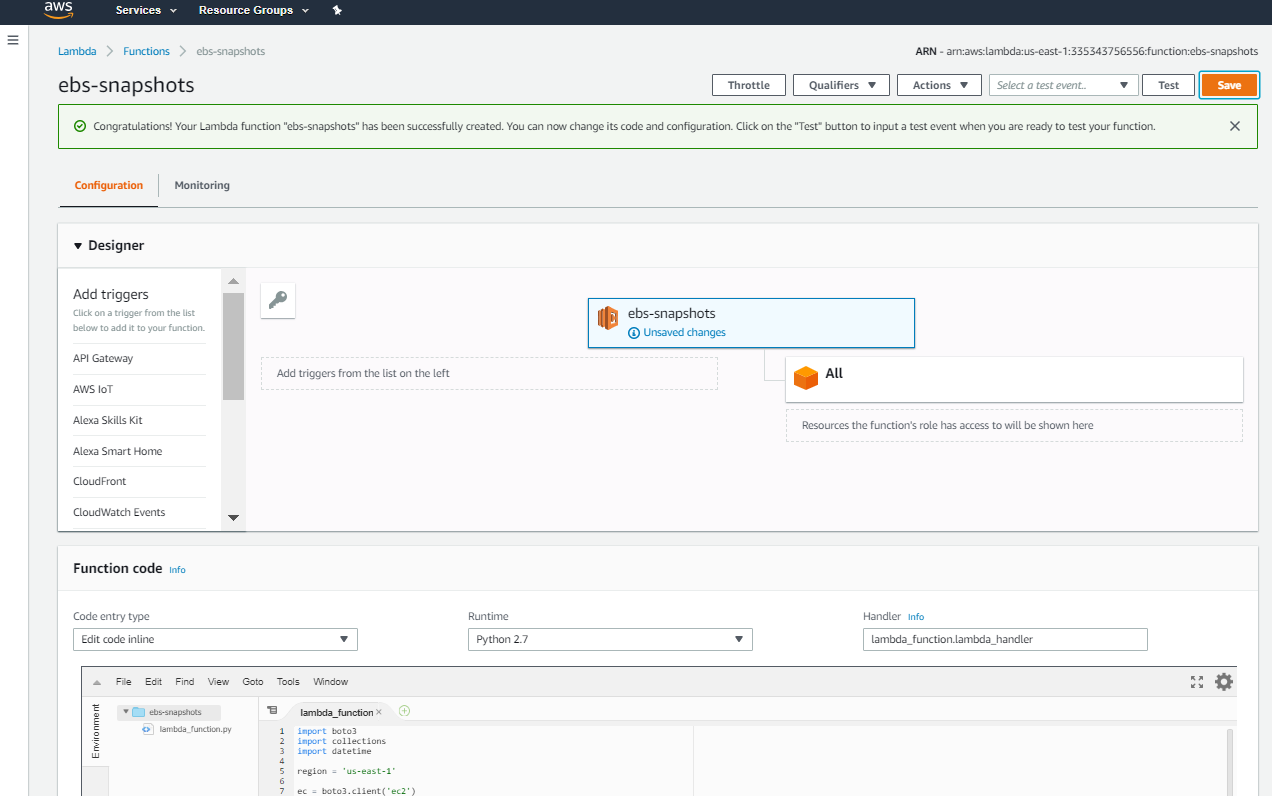
Paste the code in the code editor



Scroll down & ensure that the section labelled Execution role has your lambda-ebs-snapshot role selected

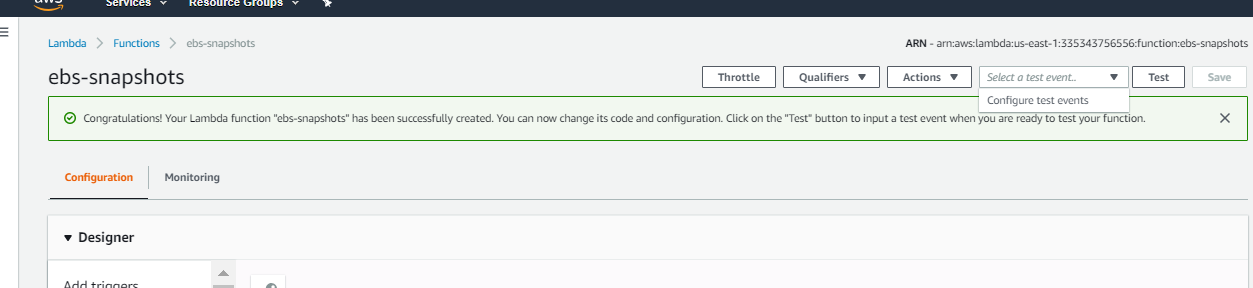
On the right-hand side of Lambda execution role section, you will see another section labelled basic settings. Change the Timeout value in the basic settings section. Enter 1 in the minute field and make the seconds to 0

Scroll back to the top and click the orange button towards top-right hand corner labelled “save”



Once the function is saved, we will test the function

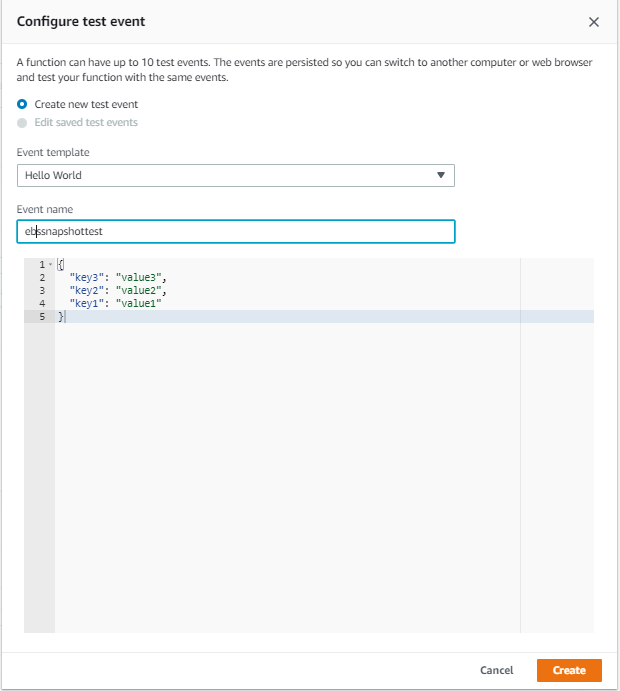
Click on the drop-down menu next to the “test” button (top-right hand corner) and select configure test events



A template will pop-up that allows to you generate mock event that can trigger and test your lambda function.

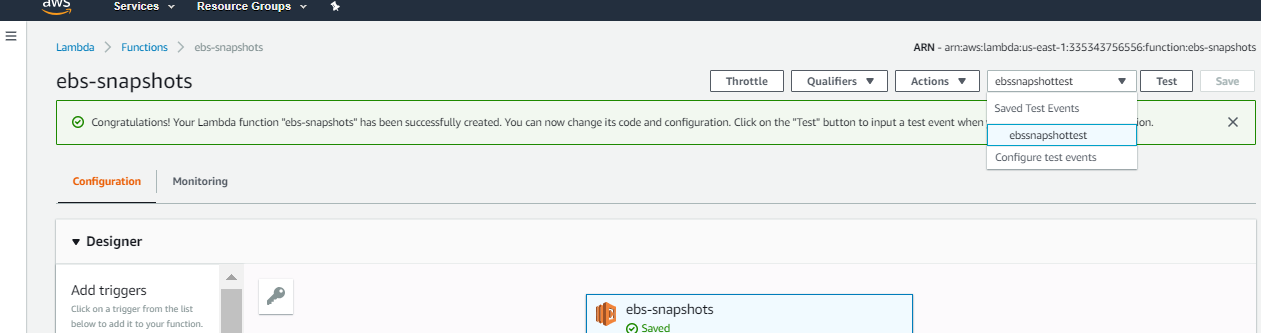
For comprehensive functions, this template can be a wonder-pill. However, for our function we will use the basic default hello-world template without any changes

Type in the Event name as ebssnapshottest (note: special characters and spaces are not allowed)



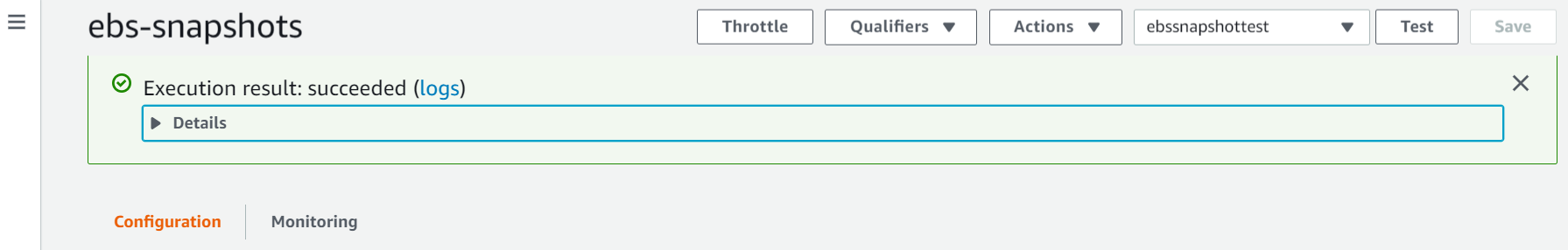
Click create

On the function page, head back to the drop-down menu next to Test button and select the test event created from the previous step. Then click Test button



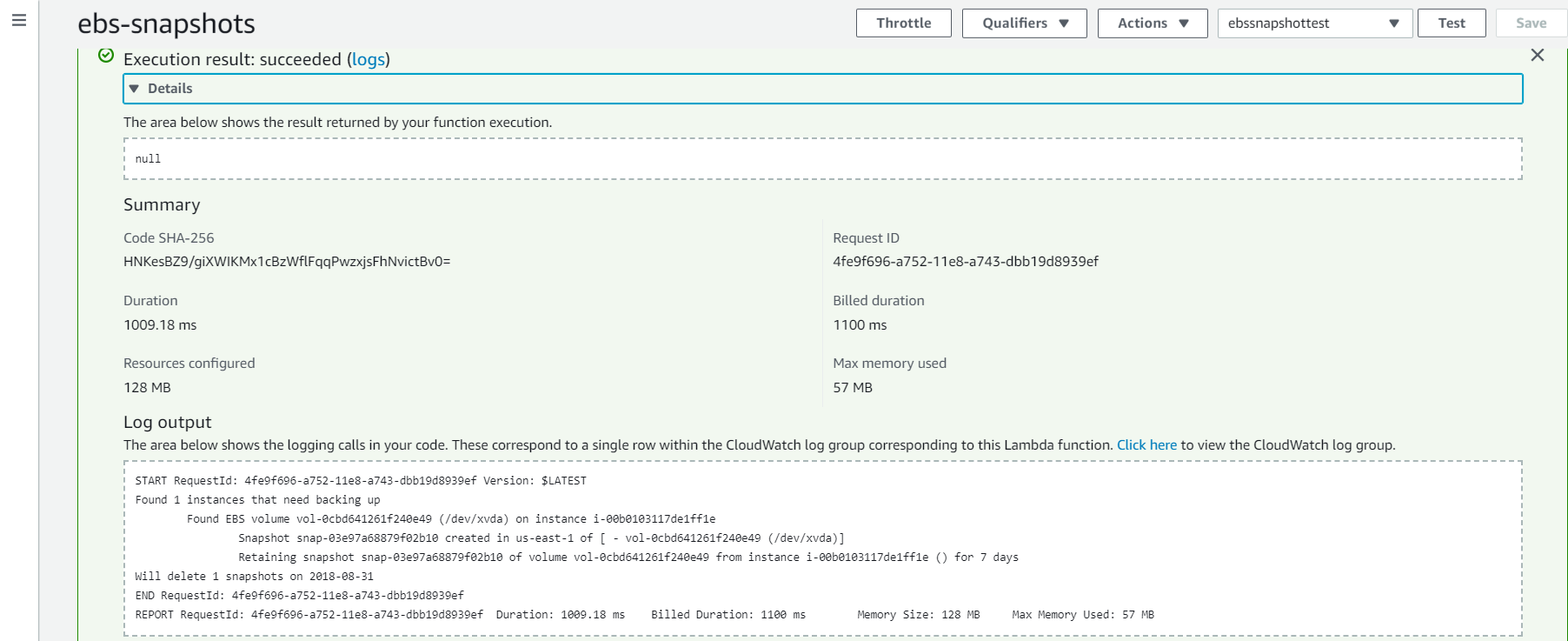
This will trigger the lambda function and we would like to see if it succeeds or fails

You can observe that the function executes and if there are no issues, you should see a success message



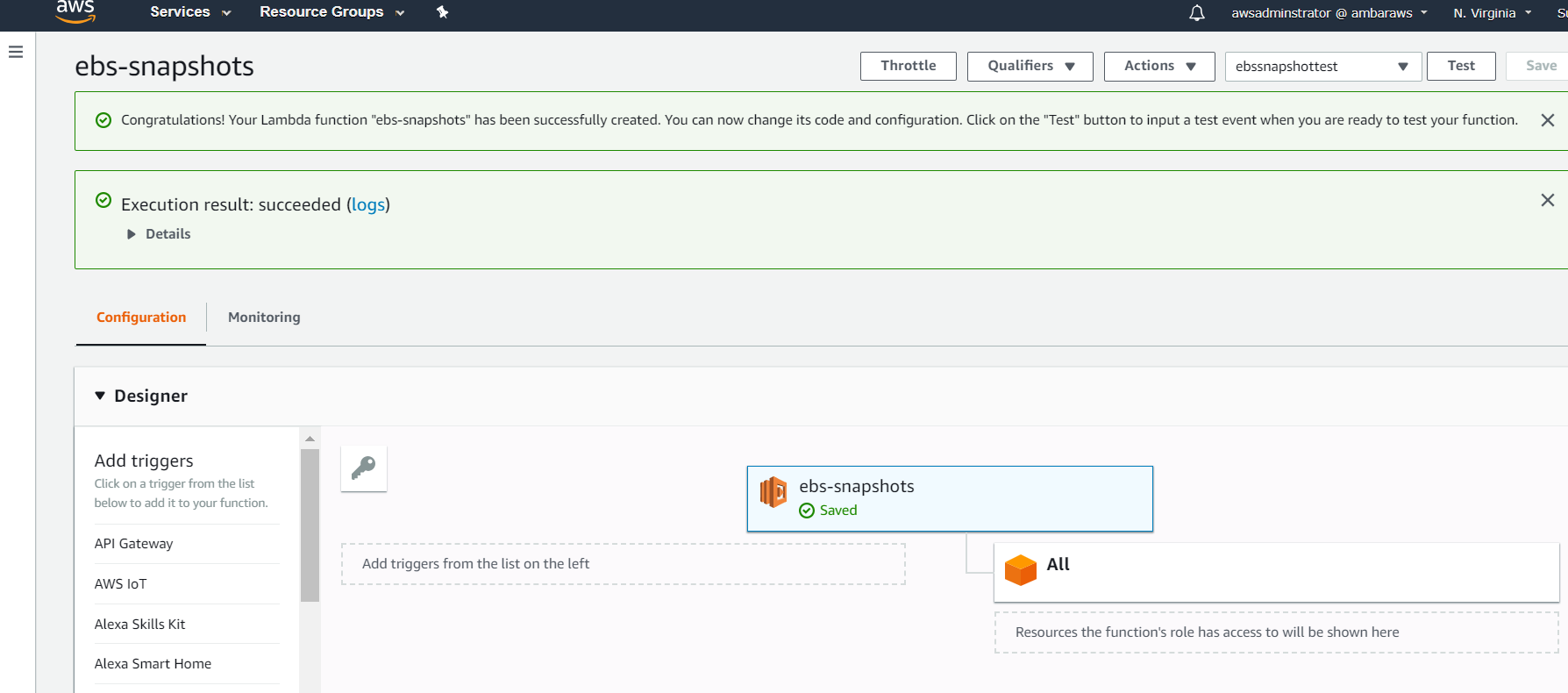
Click on the Details arrow below the success message to expand the log

You can observe that the Python code executed by the function was able to find the Instance with the tag key=application & value=app-a. After that it initiated a snapshot of the EBS volume attached and also labelled it with a delete-marker that will automatically delete the snapshot after 7 days

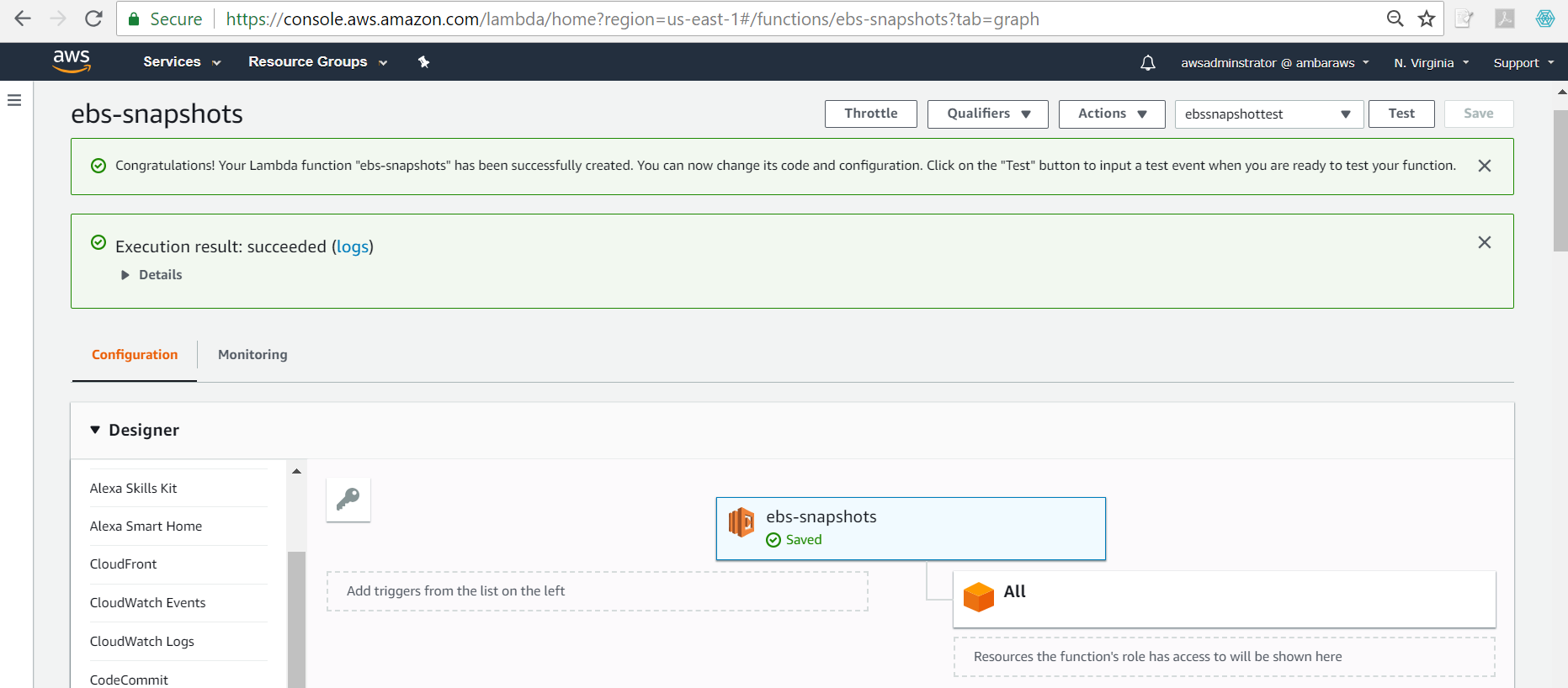


Now that our code is tested and fully functional, we can create a trigger that can automate a CRON schedule to run this function with a desired recurrence

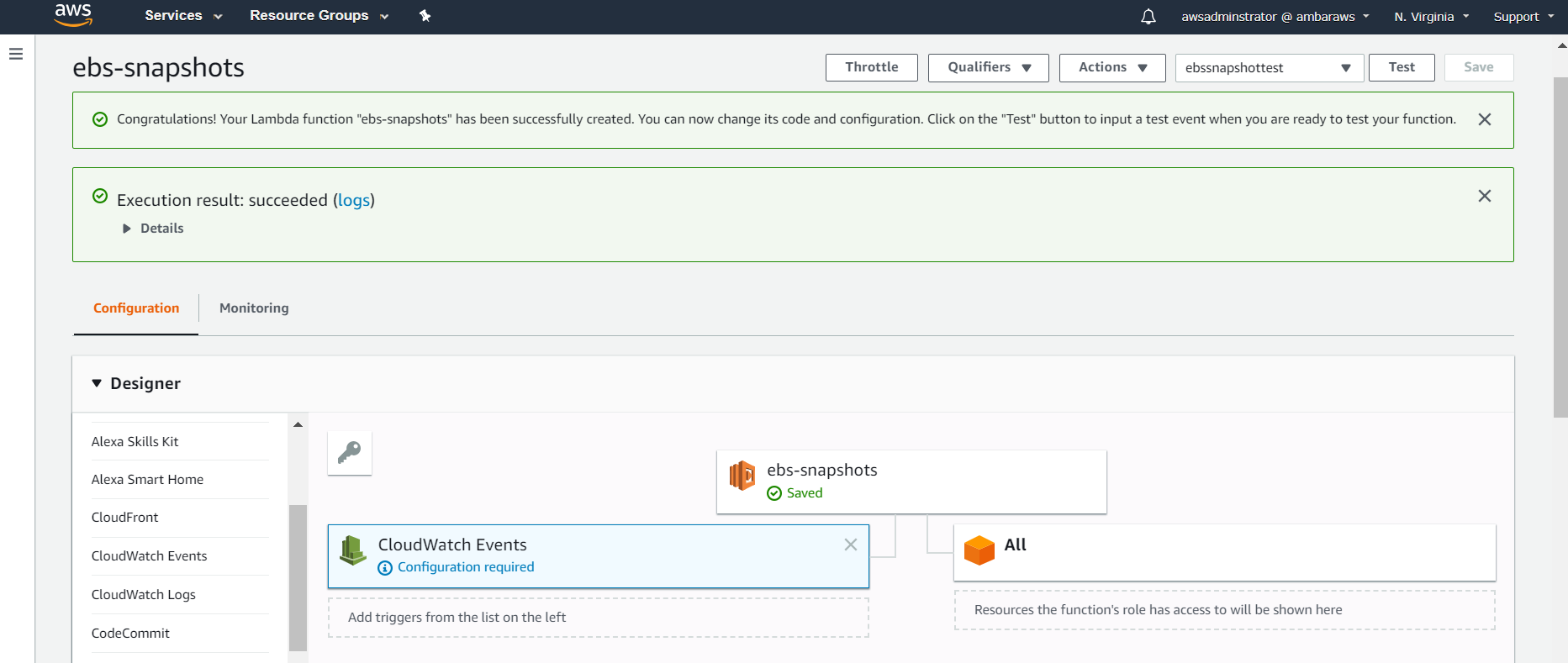
Locate the Triggers section as a left-hand sub-menu where it says Add Triggers



Select CloudWatch Events from the list of triggers

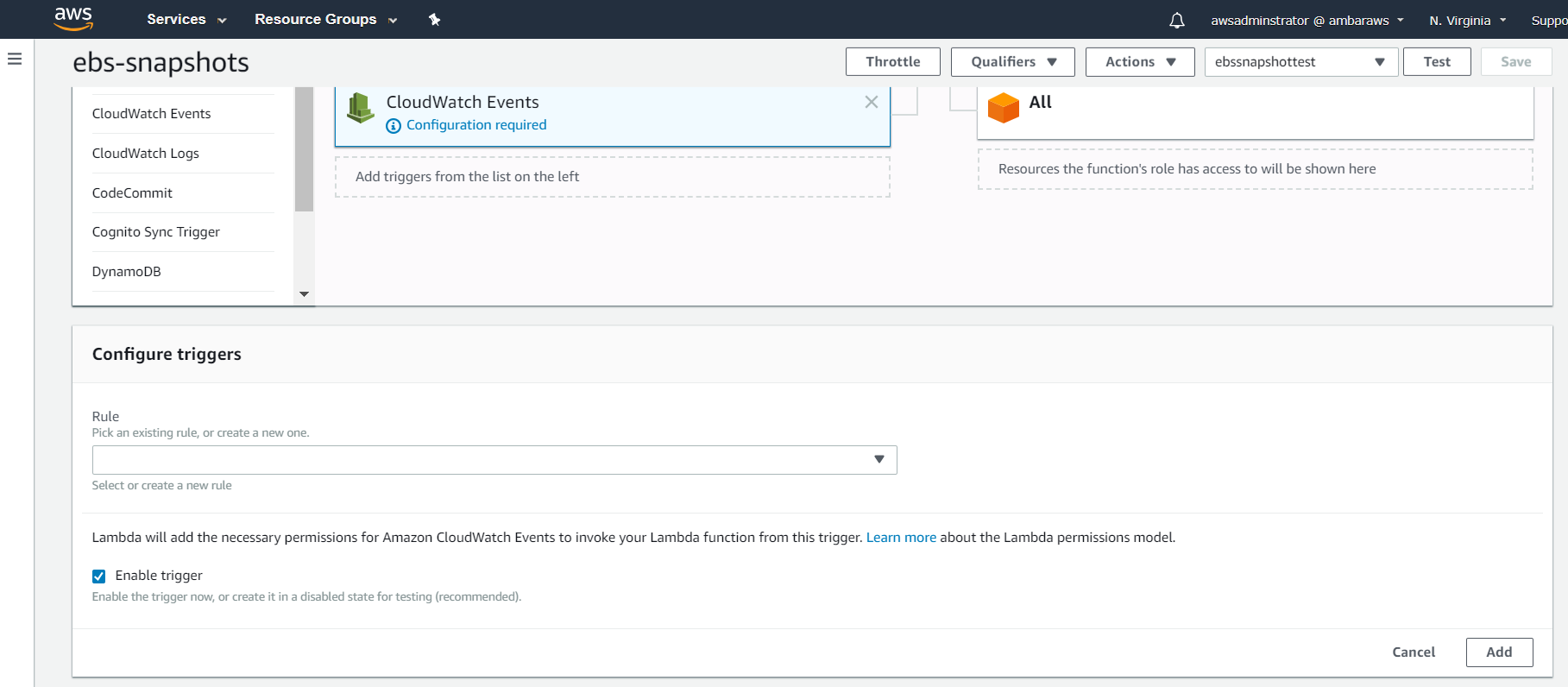


Once selected the CloudWatch Events will appear in the panel

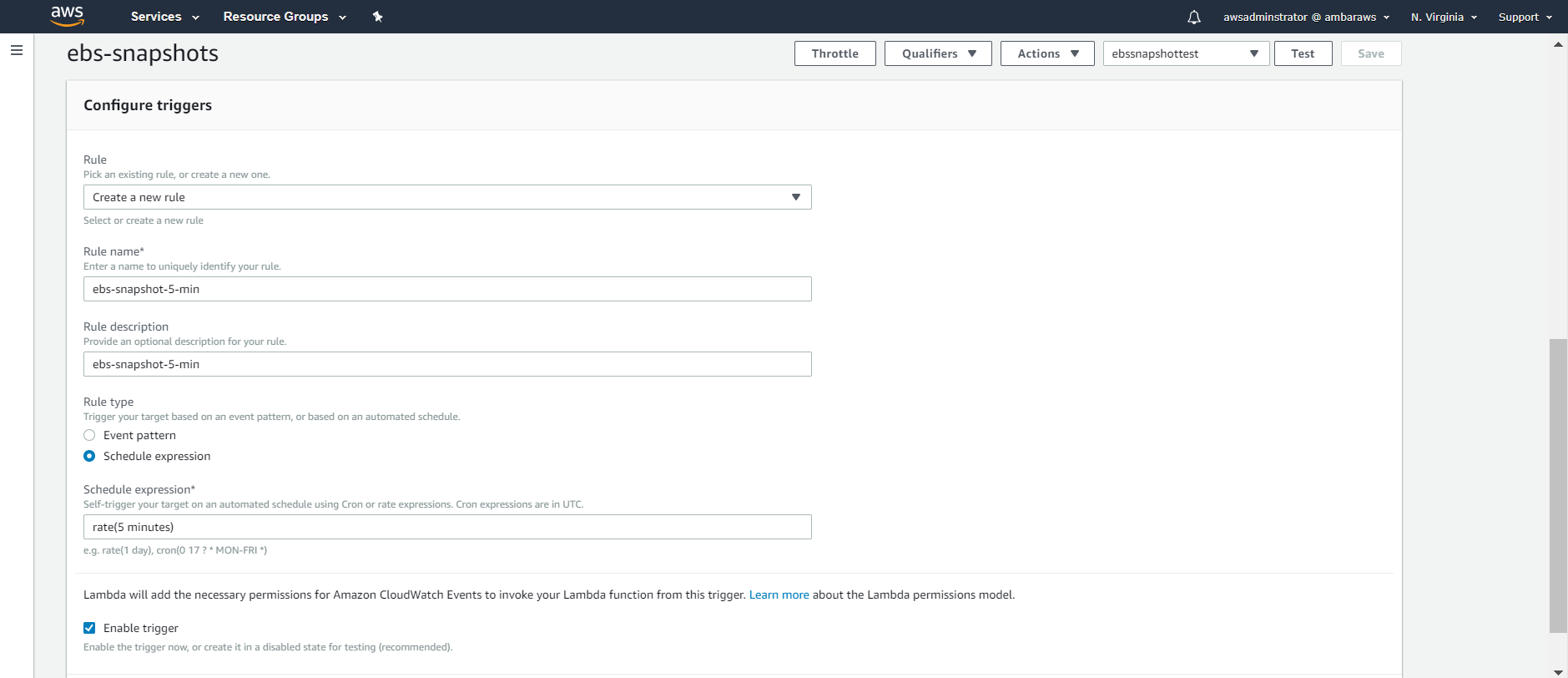


Next, we need to configure our trigger for the CRON scheduled expression

Scroll down a little and you will locate the Configure triggers section



Click on the drop-down for the trigger rule & click on create a new rule



Give the rule name and description as ebs-snapshot-5-min

In the schedule expression field type **rate(5 minutes)**

Note that the expression syntax is important so copy the above into the field if you wish

Ensure that the Enable Trigger is selected

And then click on Add

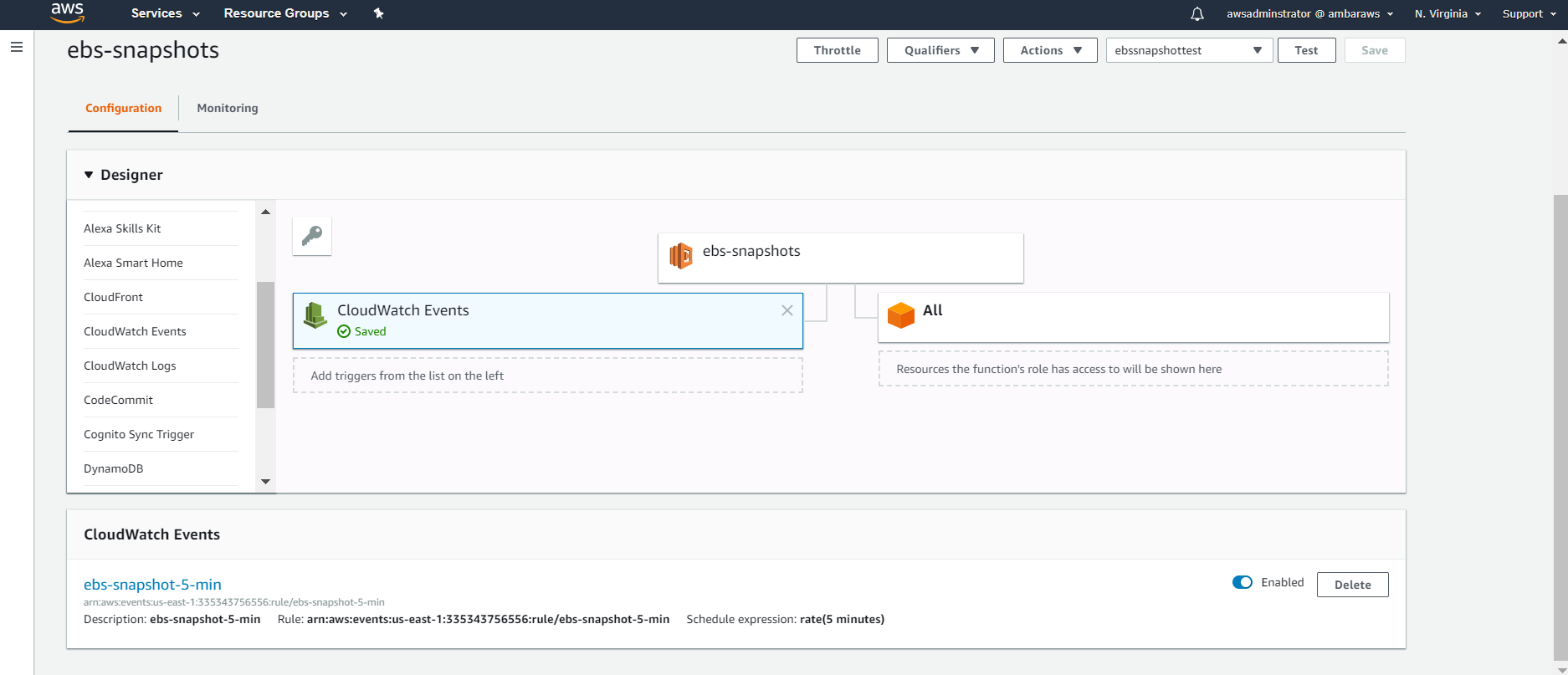
Once you add the rule, the trigger is set

The trigger is basically configured to invoke a CloudWatch mock event that fires the lambda function every 5 minutes

You will need to hit the **save button** and your trigger will be live and start firing events every 5 minutes to invoke the lambda

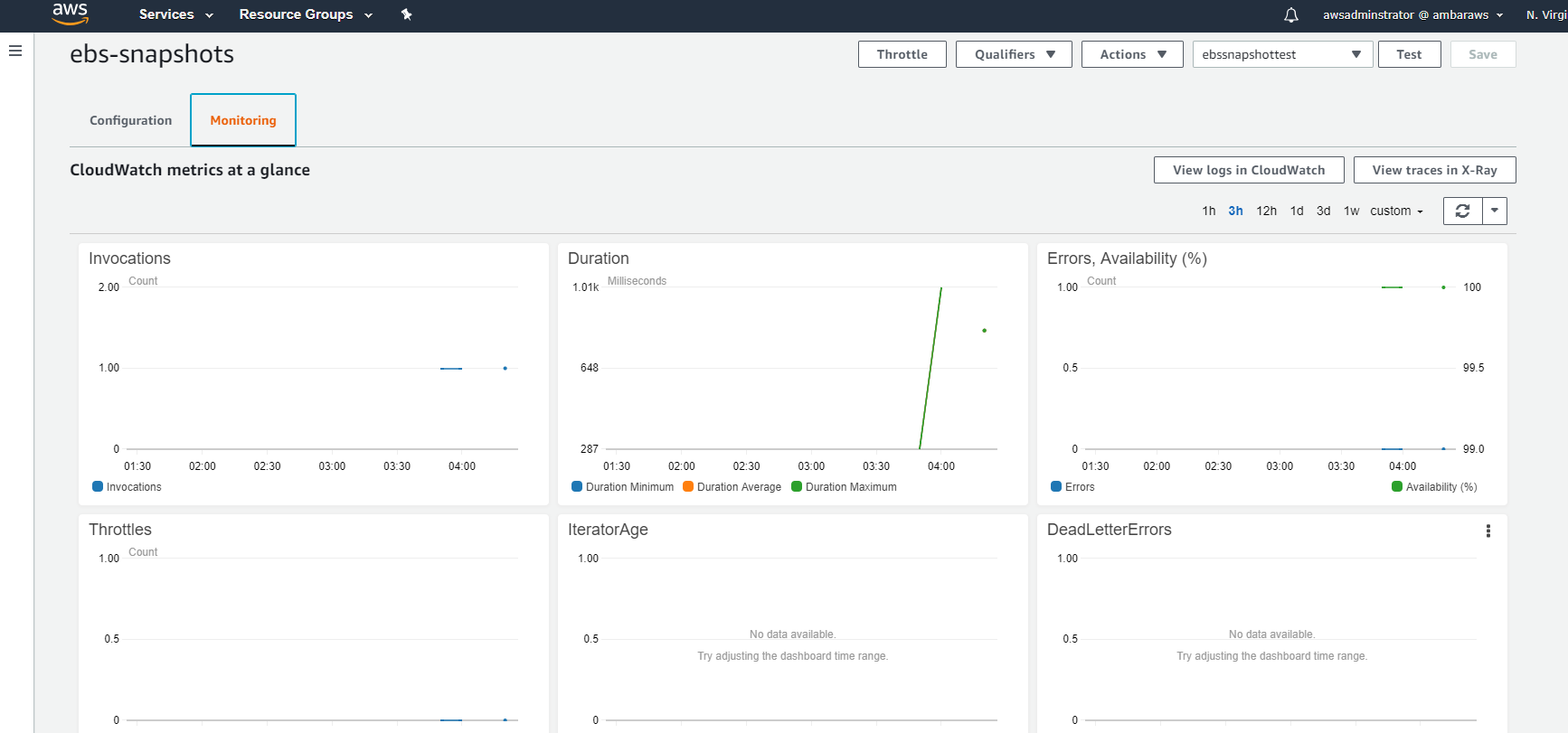
In a nutshell, your lambda function will be invoked for every 5 minutes and take an EBS snapshot of all volumes attached to the EC2 instance with the tag key=application & value=app-a

Once the function is saved, you can see a similar screen



Now click on the monitoring tab next to configuration tab

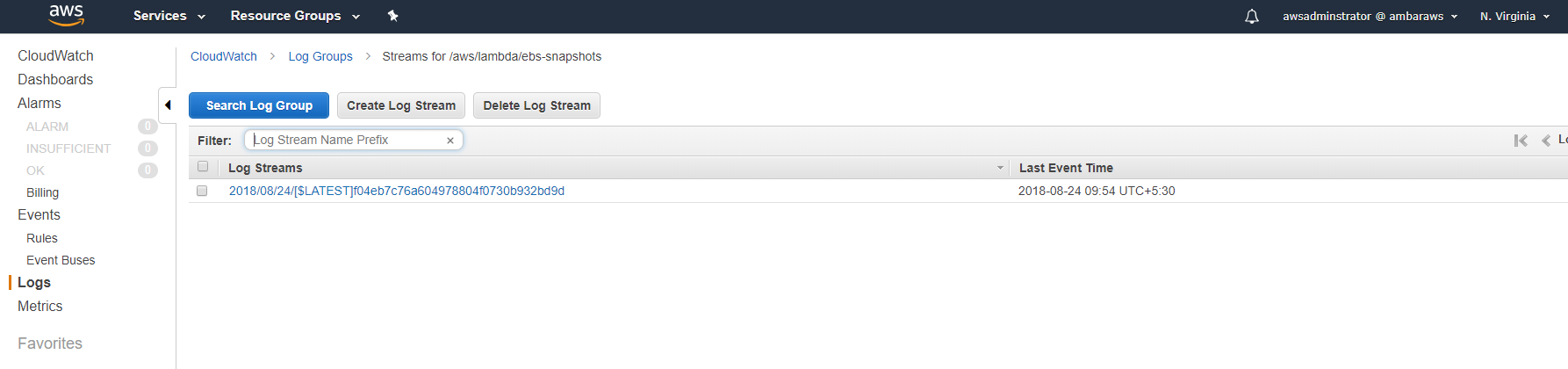
This will take you to the CloudWatch dashboard that details your Lambda metrics



Scroll down on the metrics and observe various graphs most notably number of invocations & if there is any error count

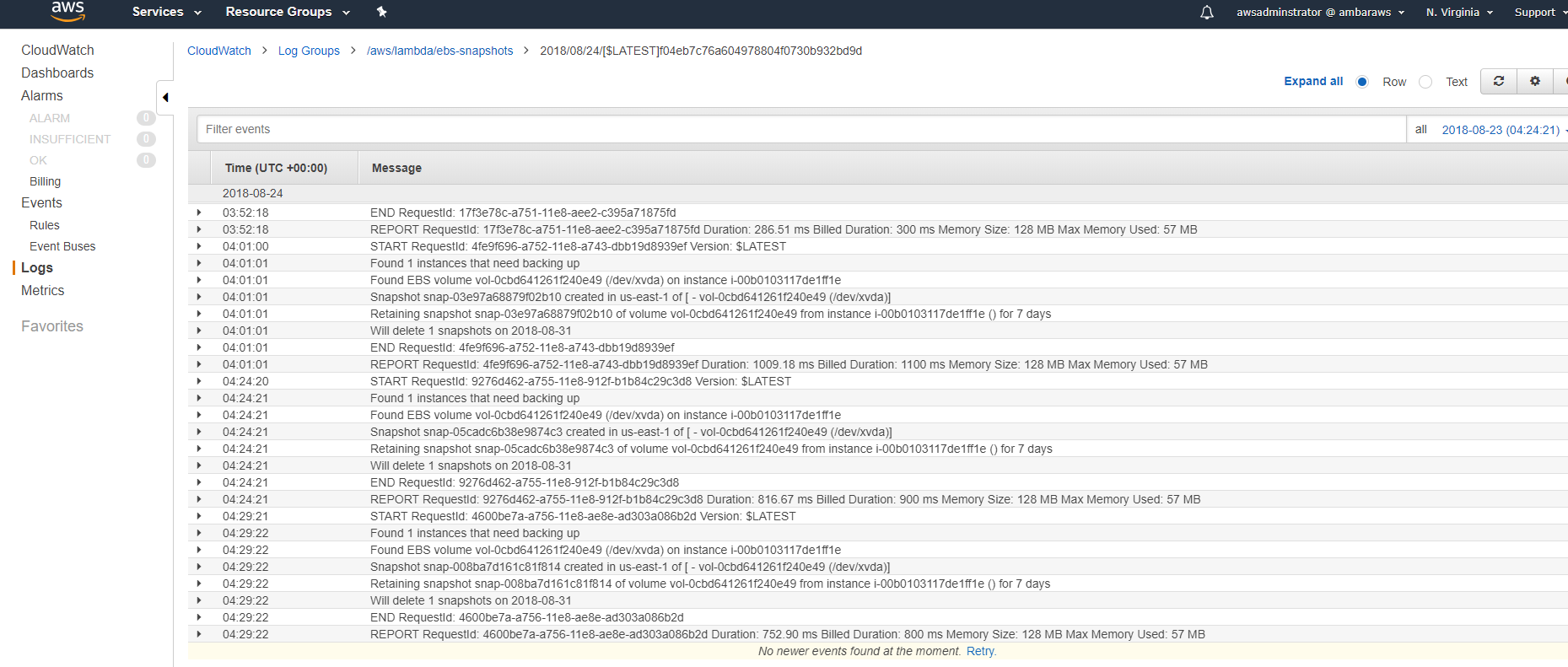
Now click on the button on the Monitoring page labelled “View logs in CloudWatch”

This will open a new page for CloudWatch Logs, where you can see a log stream automatically generated for your Lambda function



This log stream logs all events related to the execution of your Lambda function

Click the log stream and click on any event to see the details



Also observe that there will be events that will specifically highlight your billed duration. This is the amount of lambda function execution time that you will pay for

# Step-4 Observe EBS snapshots

You can have a cup of tea 😊 or wait for 15-20 minutes and then navigate to the EC2 dashboard to observe the EBS snapshots created at a recurrence of 5 minutes

If you ever wanted to change the rate at which EBS snapshots are taken, then you need to go to the trigger and change the schedule expression accordingly

# Step-5 EBS snapshots for app-b

If you want to repeat this process for the second EC2 instance, then you can go back to the configuration page., change the Python code line 12 with the tag value of app-b

Alternatively, you can build a separate individual function for app-b alone

# Step-5 Disable or Delete

You are not billed for any saved lambda functions unless it is executed

In that case, either you can delete the function or disable the trigger or alternatively remove the trigger itself from the Lambda function