

AWS Lambda

Introduction

- It is a compute service where you can upload your code to AWS Lambda and the service can run the code on your behalf using AWS infrastructure
- ***It is Serverless Compute***: Run code without managing servers, code only runs when it needs to run (event driven)
- Everything is dynamically auto-scaled so if you have 1 user or 1 billion you pay for usage.
- Lambda handles: Capacity, Scaling, Deployment, Monitoring, Logging, Web Service frontend, Security Patching.

Some Use Cases:

- Send Customized messages to different users (SNS + Lambda)
- Send an offer when a user lives run out of lives in my game (Cognito + Lambda + SNS)
- Transform the records in a click stream or an IoT data stream (Kinesis + Lambda)
- Take an action on a github pull (Github + SNS + Lambda)
- Scan files to added to an s3 bucket and index them (S3 + Lambda + DynamoDB)
- Audit all AWS API usage (CloudTrail + S3 + Lambda)

Terminology

<u>Term</u>	<u>Description</u>
Lambda	lets you run code without provisioning or managing servers.
blueprint	Sample configurations of event sources and Lambda functions.
Trigger	adding triggers for executing functions w.r.t schedule / condition
Alias	Points to the specified Lambda function version
Version	allows you to better manage your in-production Lambda function code by enabling you to publish one or more versions of your Lambda function.
ARN	Amazon Resource Name. Unique and immutable (it can't be changed) Qualified ARN -The function ARN with the version suffix. Unqualified ARN – The function ARN without the version suffix.

Objectives

- Creating Hello world function (**Inline / Zip**)
- Managing Versioning Using the AWS Management Console
- Adding Triggers to Lambda function

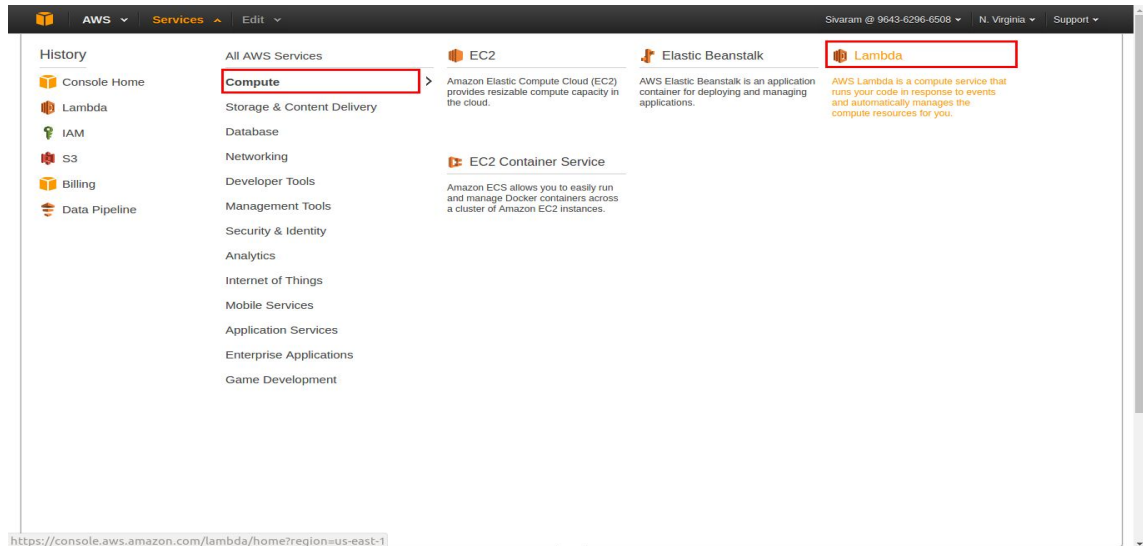
Pre-Requisites

- AWS Account
- Basic understand on AWS Services and Practical knowledge on AWS CLI / AWS Console

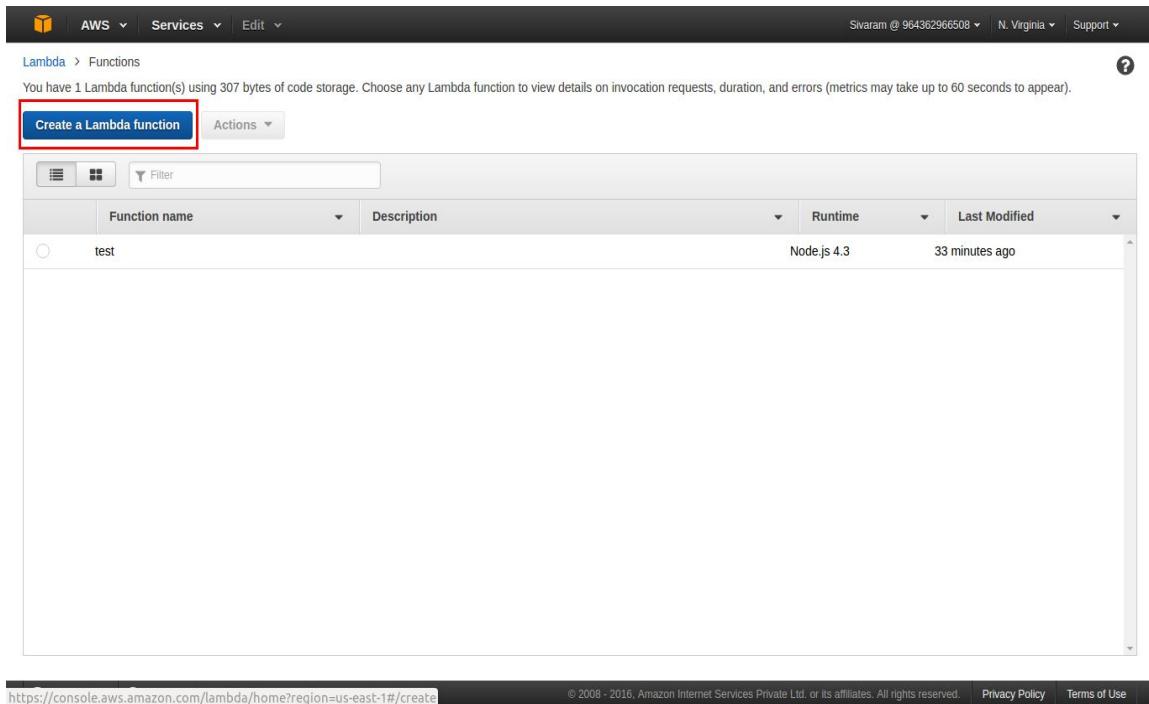
Instructions

Creating Hello world function

1. Go to Services -> Select Compute -> Click on Lambda



2. Click on “Create a Lambda function”



3. Click on **Configure function**

- Configure function: Give the name of function, description and Runtime variable(Node.js / Java / Python)
- Lambda function Code: Select Code entry type as either inline / **Upload a .ZIP file** / Upload a file from s3
- Select **“Upload a .zip file”** from dropdown and upload the hello-world.zip from pc
- Lambda function handler and role: give the handler name select the role for the Lambda function
- Advance Settings: Memory: 128 MB, Timeout: 3 Sec

Configure function
A Lambda function consists of the custom code you want to execute. [Learn more about Lambda functions.](#)

Name* helloworld

Description this is my hello world function

Runtime* Node.js 4.3
 Java 8
 Node.js
 Node.js 4.3
 Python 2.7

Lambda function code
Provide the code for your function. Use the editor to write code, or upload a ZIP file (other than the aws-sdk). If you need custom libraries, you can upload your code and libraries.

Code entry type Upload a .ZIP file

Function package [Upload](#)
For files larger than 10 MB, consider uploading via S3.

Lambda function handler and role

Handler* index.handler

Role* Choose an existing role ⓘ

Existing role lambda_basic_execution ⓘ

Advanced settings
These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)* 128 ⓘ

Timeout* 0 min 3 sec

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. [Learn more](#) about accessing VPCs within Lambda. **Please ensure your role has appropriate permissions to configure VPC.**

4. Click on Next

Lambda function handler and role

Handler* index.handler

Role* Choose an existing role ⓘ

Existing role lambda_basic_execution ⓘ

Advanced settings
These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)* 128 ⓘ

Timeout* 0 min 3 sec

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. [Learn more](#) about accessing VPCs within Lambda. **Please ensure your role has appropriate permissions to configure VPC.**

VPC No VPC ⓘ

* These fields are required.

[Cancel](#) [Previous](#) [Next](#)

5. Review the function details and click on “Create function”

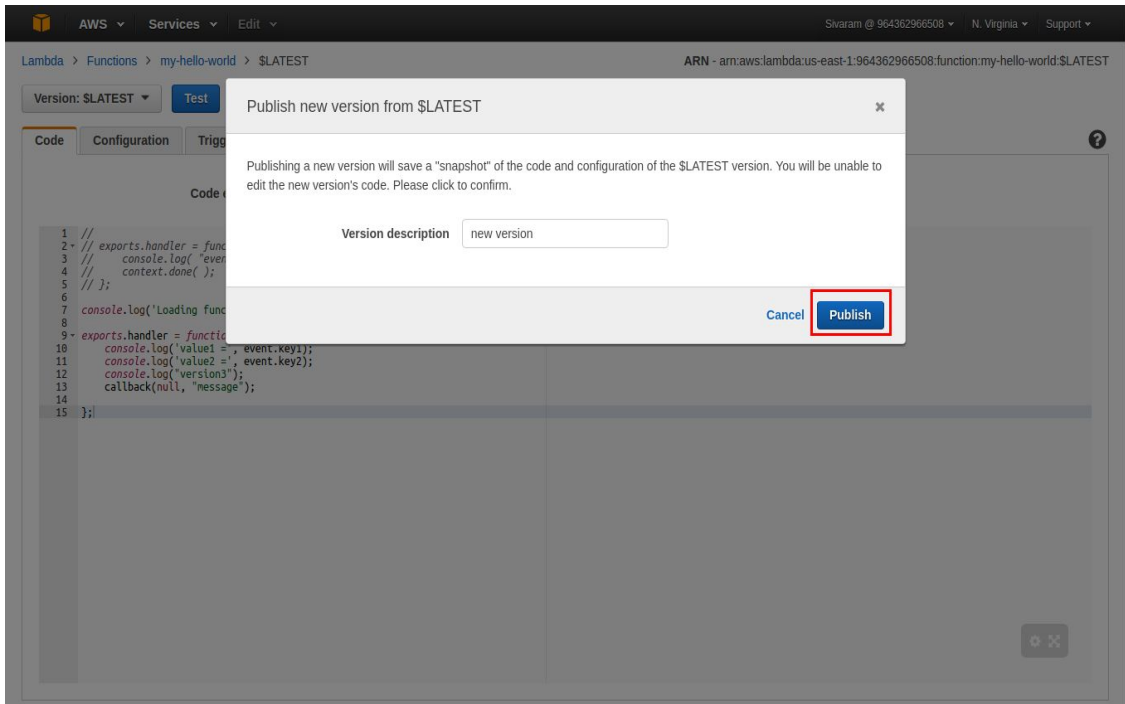
The screenshot shows the AWS Lambda console's 'Review' page for a new function. The left sidebar contains links: 'Select blueprint', 'Configure triggers', 'Configure function', and 'Review' (which is highlighted). The main content area is titled 'Review' and contains a message: 'Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.' Below this is a 'Lambda function' section with an 'Edit' button. The details listed are: Name: hello-world, Description: this is my first hello world function, Runtime: Node.js 4.3, Handler: index.handler, Existing role: lambda_basic_execution, Memory (MB): 128, Timeout: 3, and VPC: No VPC. At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Create function' (which is highlighted with a red box).

6. Click on “Test” to run the function and Check the execution results.

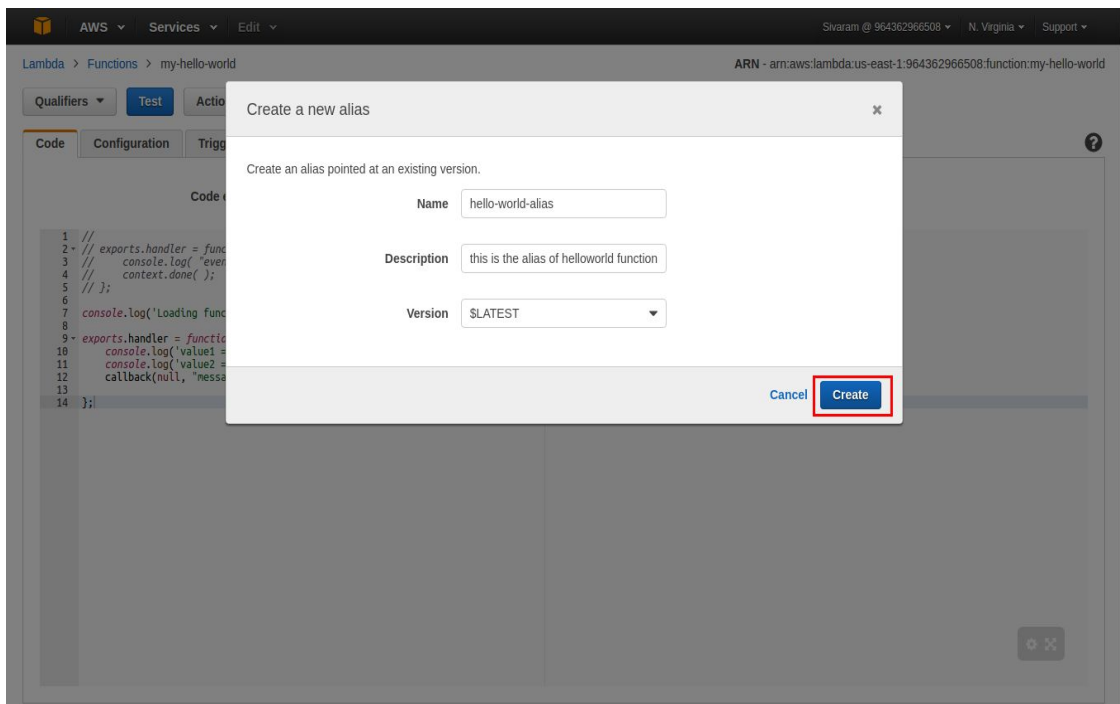
The screenshot shows the AWS Lambda console's 'Test' results page. At the top, there is a large empty box for the function's output. Below it, a green checkmark indicates 'Execution result: succeeded (logs)'. A message states: 'The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.' Below this is a dashed box containing the JSON output: `{ "message" }`. The page is divided into two sections: 'Summary' and 'Log output'. The 'Summary' section shows details for the Code SHA-256, Request ID (e8865d41-4dc3-11e6-8a8c-cd5235f7cba6), Duration (0.58 ms), Billed duration (100 ms), Resources configured (128 MB), and Max memory used (15 MB). The 'Log output' section shows a message: 'The area 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.' Below this is a dashed box containing the log output: `START RequestId: e8865d41-4dc3-11e6-8a8c-cd5235f7cba6 Version: $LATEST`, `2016-07-19T15:17:29.642Z e8865d41-4dc3-11e6-8a8c-cd5235f7cba6 value1 = value1`, `2016-07-19T15:17:29.642Z e8865d41-4dc3-11e6-8a8c-cd5235f7cba6 value2 = value2`, `2016-07-19T15:17:29.642Z e8865d41-4dc3-11e6-8a8c-cd5235f7cba6 value2 = value2`, `END RequestId: e8865d41-4dc3-11e6-8a8c-cd5235f7cba6`, and `REPORT RequestId: e8865d41-4dc3-11e6-8a8c-cd5235f7cba6 Duration: 0.58 ms Billed Duration: 100 ms Memory`.

Managing Versioning Using the AWS Management Console

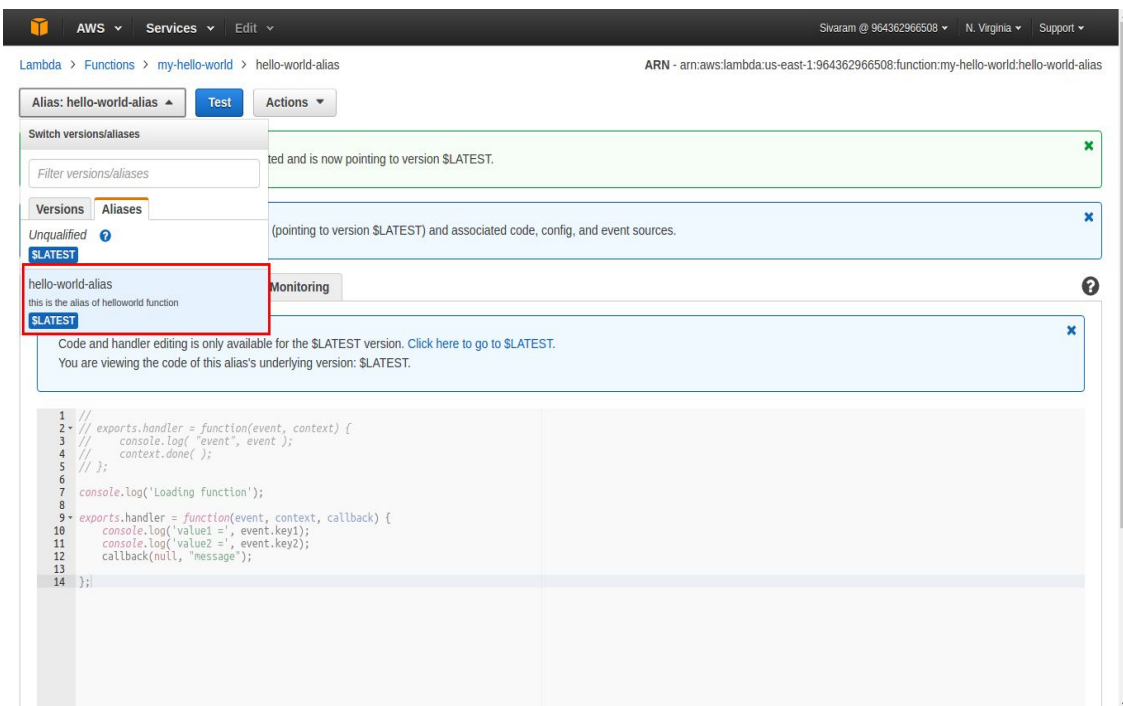
1. Select the existing lambda function and choose “Qualifiers”
2. It has two options as Versions: **\$LATEST** and Aliases
3. Modify the code with any changes and Choose Actions and “Publish new Version”



4. It creates the Version(s)1 (immutable)
5. And also Choose Actions and Click on “**Create Alias**” (and map to specific version)

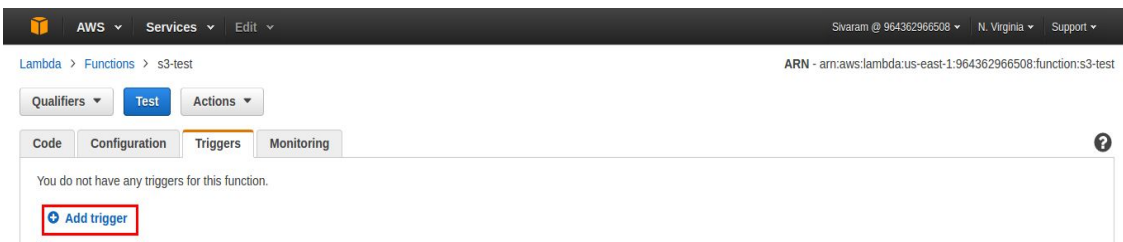


6. The below screen displays the existing versions and alias

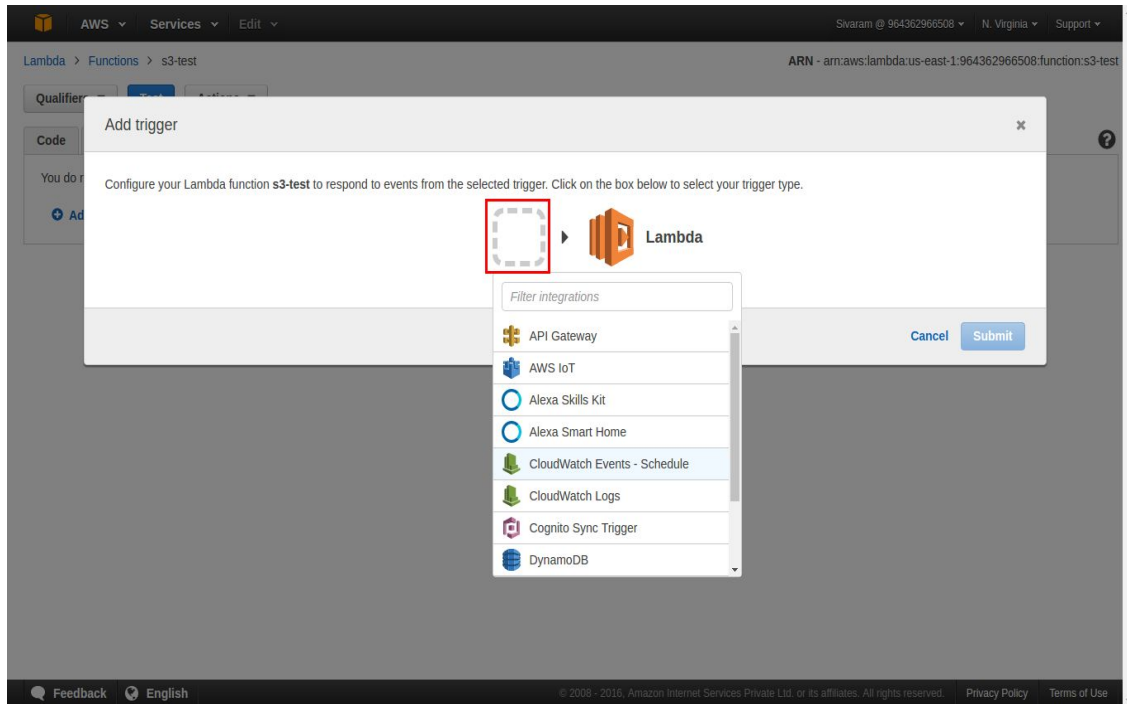


Adding Triggers to Lambda function

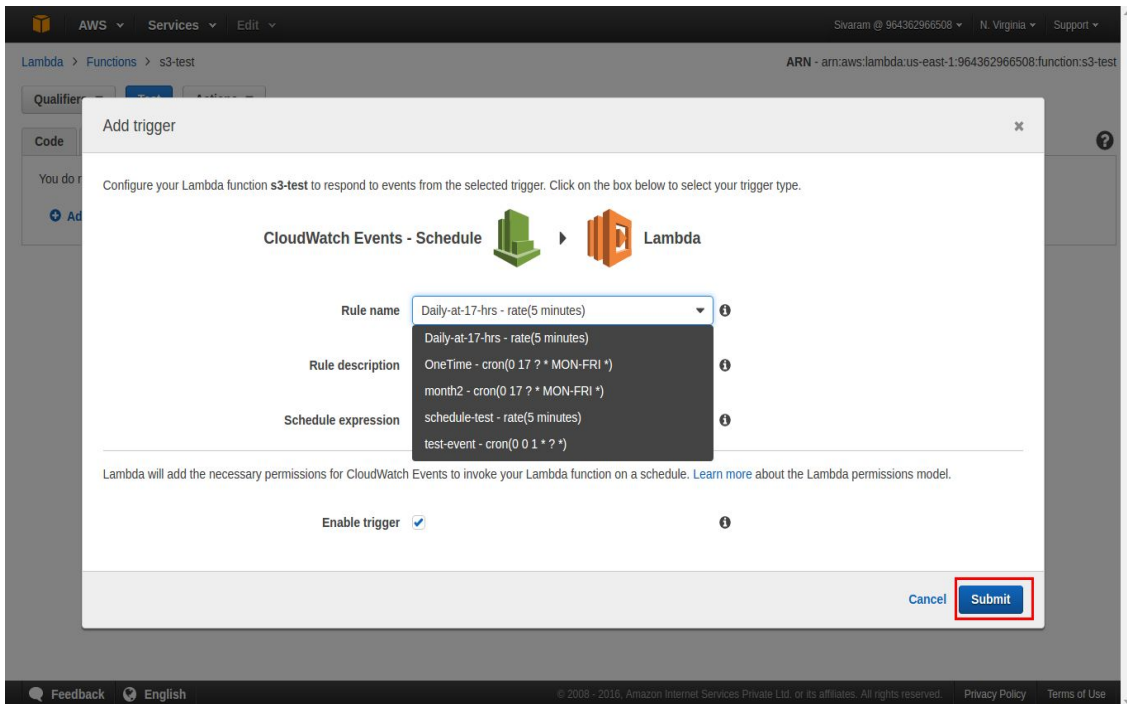
1. Double Click on any Lambda function
2. Select the 'Triggers' tab and Click on "Add trigger"



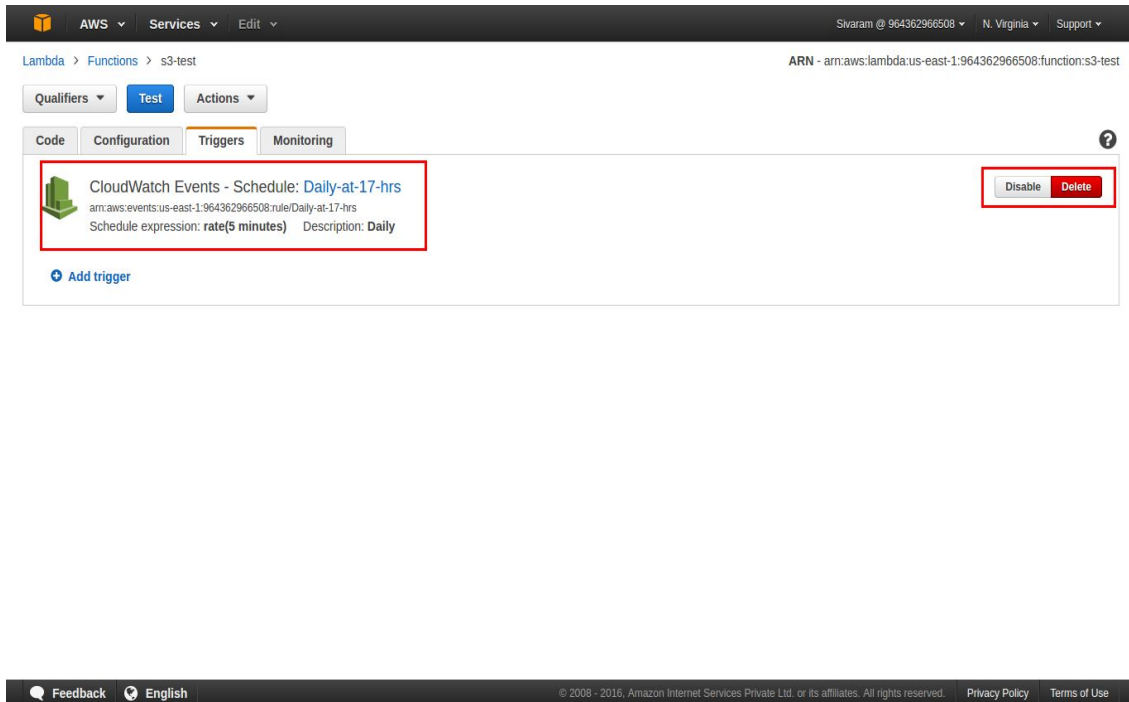
3. Configure the event in below screen



4. Select any integration (event), give the Rule name, schedule expression and schedule and click on Submit



5. Check the created Trigger



Note: We can run the Lambda function by adding triggers, we need to disable, when we have multiple versions, because triggers run automatically in backend. We can monitor the lambda function through cloud-watch.

Sample function Lambda function in Node.js

<p><u>Helloworld.js</u></p> <pre>exports.handler = function(event, context, callback) { console.log('value1 =', event.key1); console.log('value2 =', event.key2); console.log('value3 =', event.key3); callback(null, "message"); };</pre> <p><u>Handler: helloworld.handler</u></p>	<p><u>node-lambda</u> https://www.npmjs.com/package/node-lambda npm install -g node-lambda // Install node-lambda setup // to setup lambda app in local env node-lambda run // to run lambda app in local env node-lambda package // to zip the package node-lambda deploy // to deploy the zip to remote (AWS)</p>
--	--

References:

- <http://docs.aws.amazon.com/lambda/latest/dg/use-cases.html>
- <https://github.com/awslabs/aws-lambda-zombie-workshop>
- <https://github.com/awslabs/aws-lambda-rdbms-integration>

***** **Thank You** *****