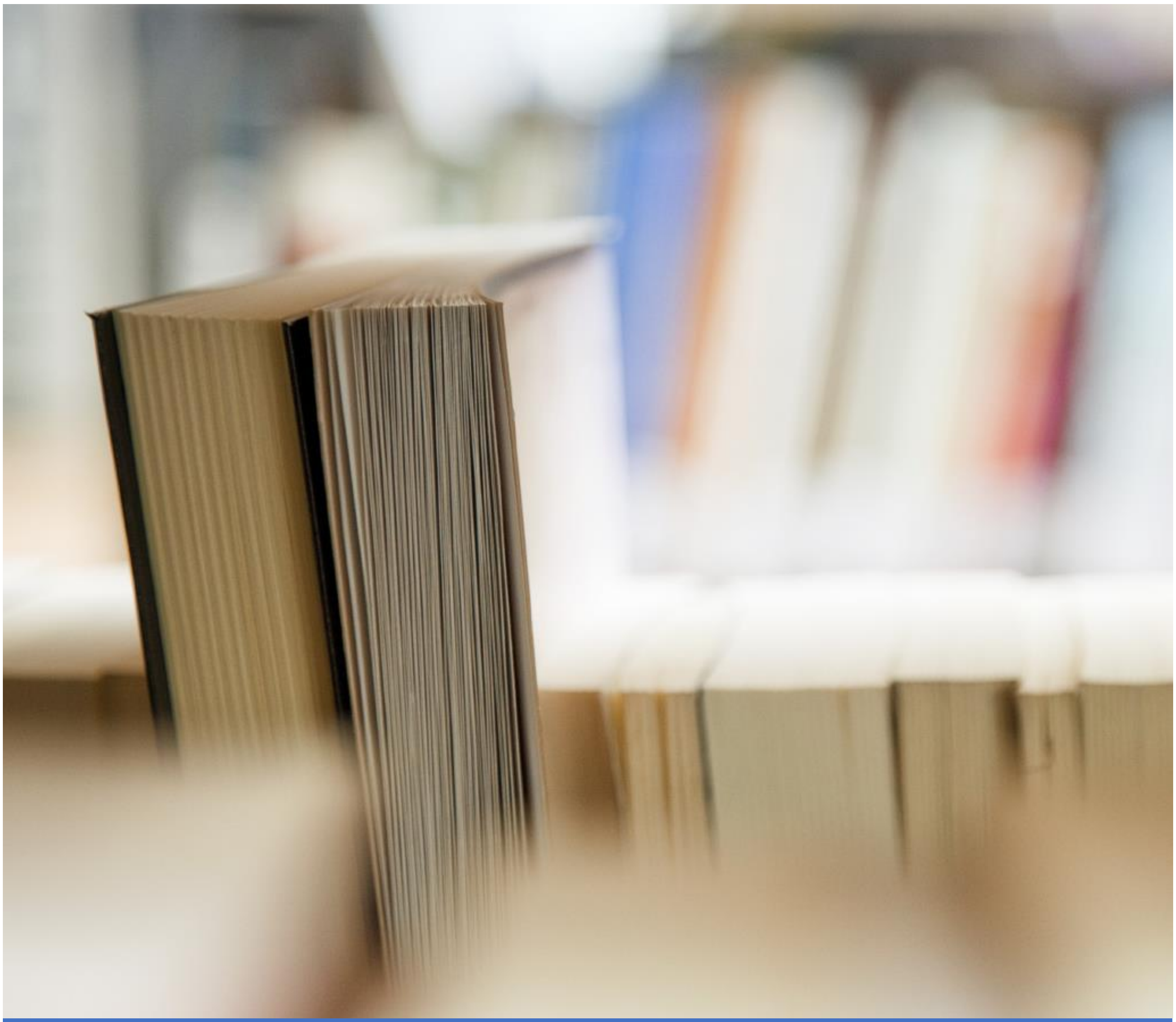
Author: Saurav Raghuvanshi



Demo: Understanding Lambda Function

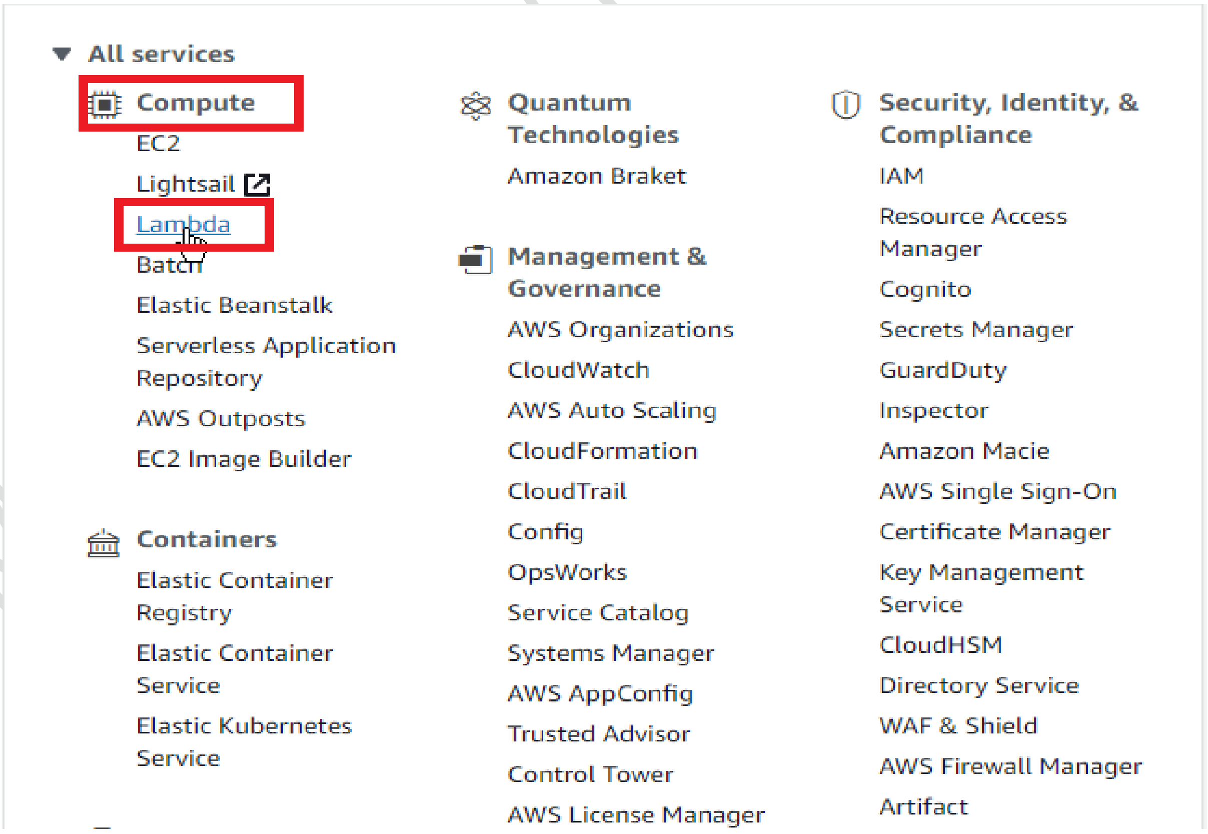
Author: Saurav Raghuvanshi

Aim: In this lab we will walk through all the component of Lambda function and we will create our first lambda function.

Points to remember:

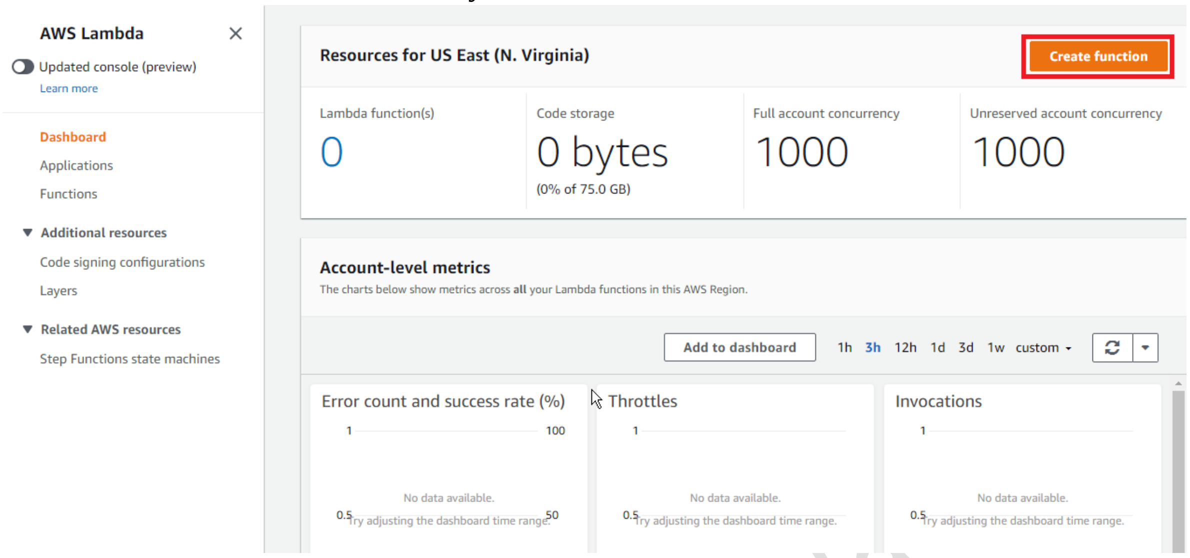
- The lambda function is compiled of your own code that you want to invoke as per the defined trigger
- Prior to creating your function, you need to have some code to add to it
- The deployment package will either be .zip or a .jar file and will contain your code and any dependencies
- How you create these deployment packages depends on the programming language that you decide to use within your function
- Once your deployment package has been created, you will need to modify the permission against your .zip file
- If you write from within Lambda itself, then it would create deployment packages for you.
- You can able to upload your code via SDK, Management Console or AWS CLI

1. Open AWS Management Console and Click on **Lambda** from **Compute Section**



Author: Saurav Raghuvanshi

2. Click on **Create Function** to create your first function



AWS Lambda

Updated console (preview)

Learn more

Dashboard

Applications

Functions

Additional resources

Code signing configurations

Layers

Related AWS resources

Step Functions state machines

Resources for US East (N. Virginia)

Create function

Lambda function(s)	Code storage	Full account concurrency	Unreserved account concurrency
0	0 bytes (0% of 75.0 GB)	1000	1000

Account-level metrics

The charts below show metrics across **all** your Lambda functions in this AWS Region.

Add to dashboard 1h 3h 12h 1d 3d 1w custom

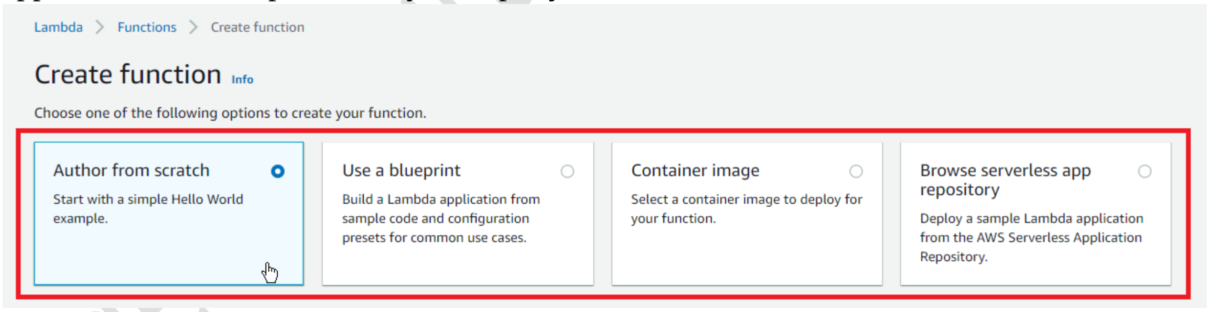
Error count and success rate (%) — No data available. Try adjusting the dashboard time range.

Throttles — No data available. Try adjusting the dashboard time range.

Invocations — No data available. Try adjusting the dashboard time range.

3. Now we have 4 options to choose from for creating the function from this choose **Author from scratch**:

   a. **Author from scratch:** Here we have to write our own code from scratch
   b. **Use a blueprint:** From this option we can able to choose from preconfigured templates which we can able to change the code based on our requirements.
   c. **Container image:** This option provide us with more control over hardware and here we deploy our application on own container image.
   d. **Browse Serverless Application Repository:** We can able to use and deploy application which is provided by third party.



Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

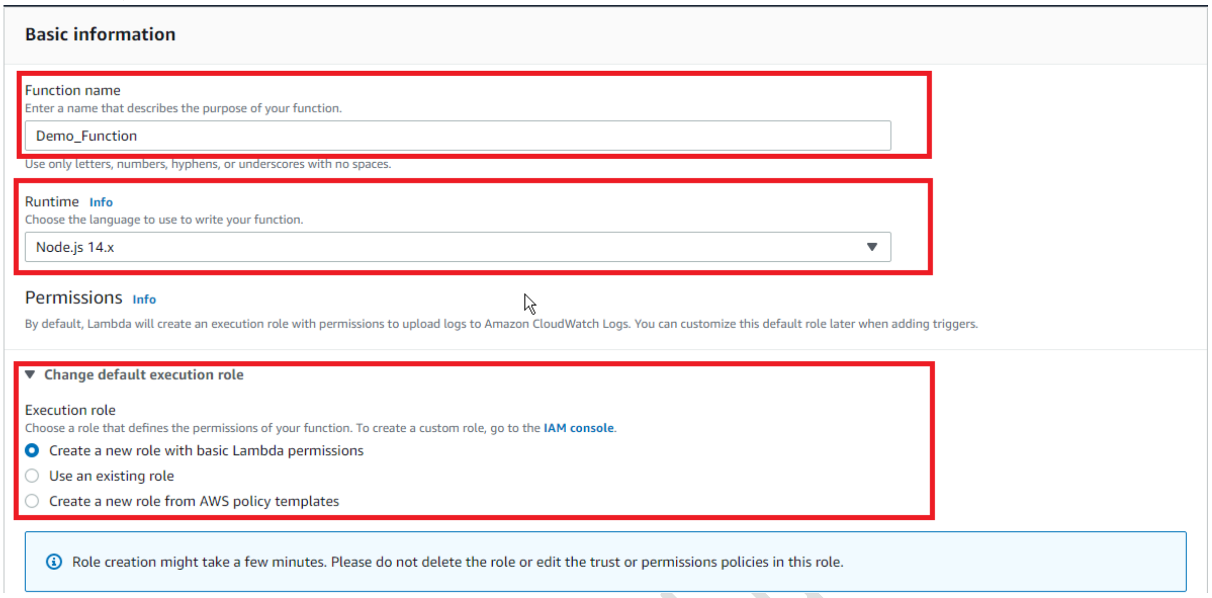
[✓ Author from scratch] — Start with a simple Hello World example.

Use a blueprint — Build a Lambda application from sample code and configuration presets for common use cases.

Container image — Select a container image to deploy for your function.

Browse serverless app repository — Deploy a sample Lambda application from the AWS Serverless Application Repository.

4. Scroll down little bit here we must give **Basic Information** like **Function name, Runtime, and Execution Role** and then click on **Create function**: -



Basic information

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

Runtime Info
Choose the language to use to write your function.
Node.js 14.x

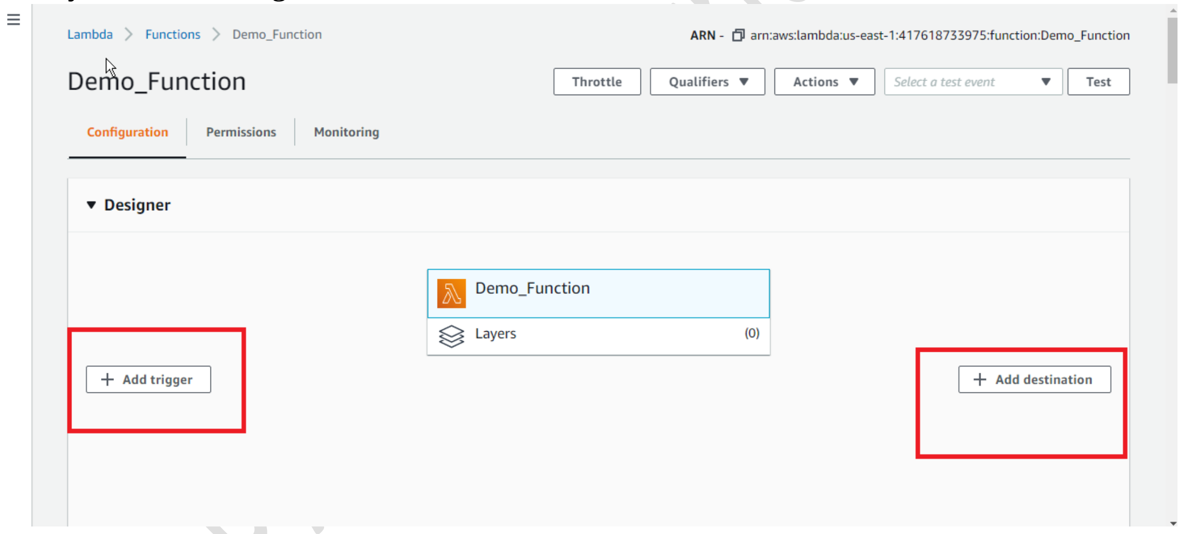
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default 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

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

5. Now you see the designer window of lambda function.



Lambda > Functions > Demo_Function

ARN - arn:aws:lambda:us-east-1:417618733975:function:Demo_Function

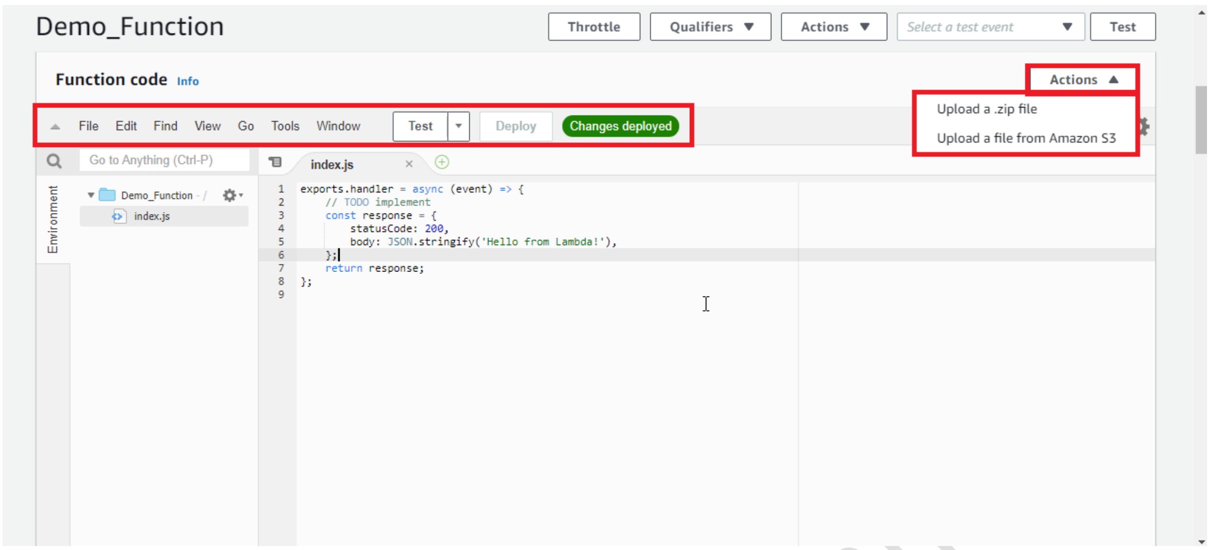
Demo_Function

Add trigger: This option is used to configure event source. An event source is an AWS service that produces the event that your Lambda responds to by invoking it.

Add destination: This option is used to configure downstream source. Downstream are resources that are required during the execution of your Lambda function (that maybe SNS or SQS)

6. Scroll down little bit you will see your lambda function view and all the option to deploy your code.

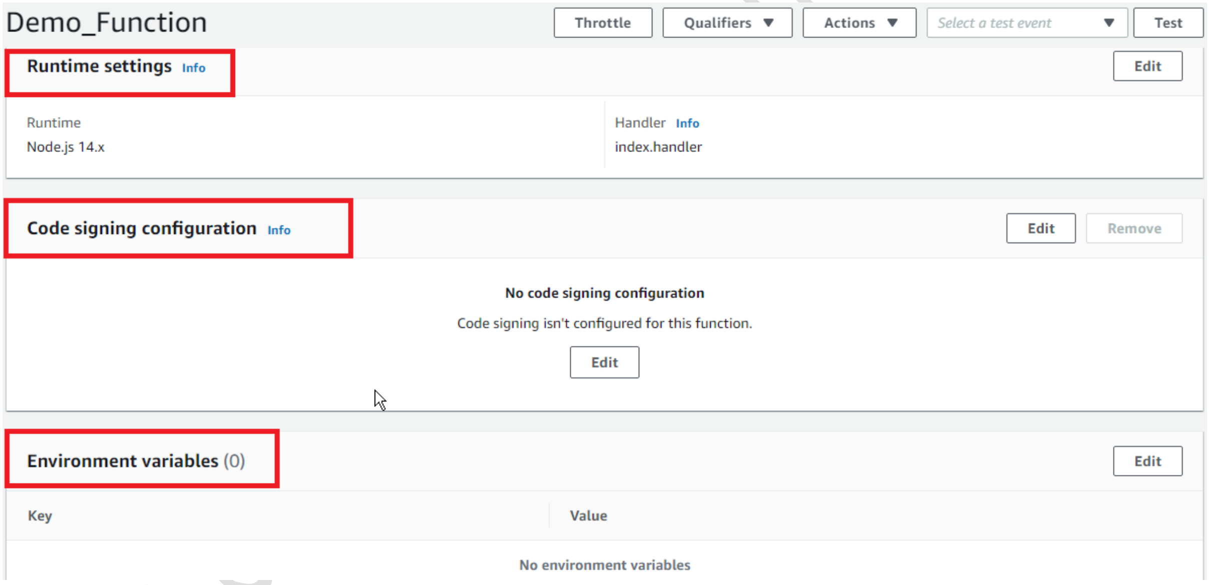
Author: Saurav Raghuvanshi



```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

Action: From this option we can able to upload our code from with complete deployment package from local machine or from S3 bucket.

7. Scroll down little bit more to see below configuration.



Runtime: Node.js 14.x
Handler Info: index.handler

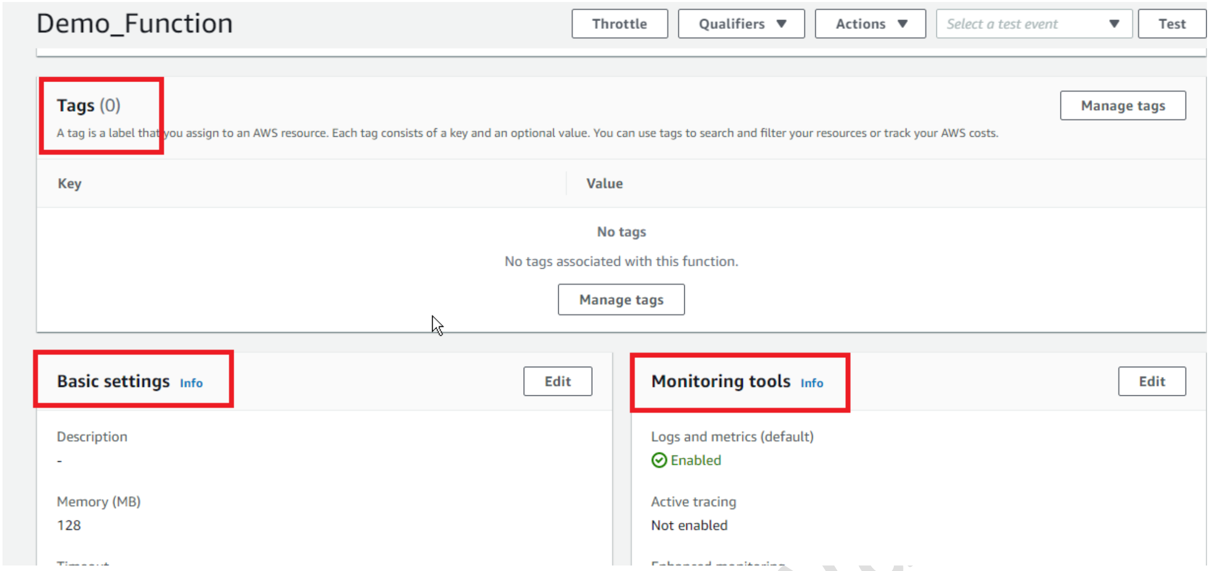
No code signing configuration
Code signing isn't configured for this function.

Key	Value

No environment variables

Runtime settings: Here we can able to set the runtime environment and the handler of the lambda function.
Code signing configuration: We use this option to restrict the deployment of unvalidated code.
Environment variable: It is key value pairs that allow us to incorporate variable into our function without embedding them directly into our code.
By default AWS lambda encrypt your environment variable after the function has been deployed using the default aws/lambda master key within the region via KMS.

8. Scroll down little bit more to see more configuration option that is available in lambda



Demo_Function

Throttle | Qualifiers ▼ | Actions ▼ | Select a test event ▼ | Test

Tags (0)

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Manage tags

Key	Value

No tags

No tags associated with this function.

Manage tags

Basic settings Info — Edit

Description

-

Memory (MB)

128

Monitoring tools Info — Edit

Logs and metrics (default)

✓ Enabled

Active tracing

Not enabled

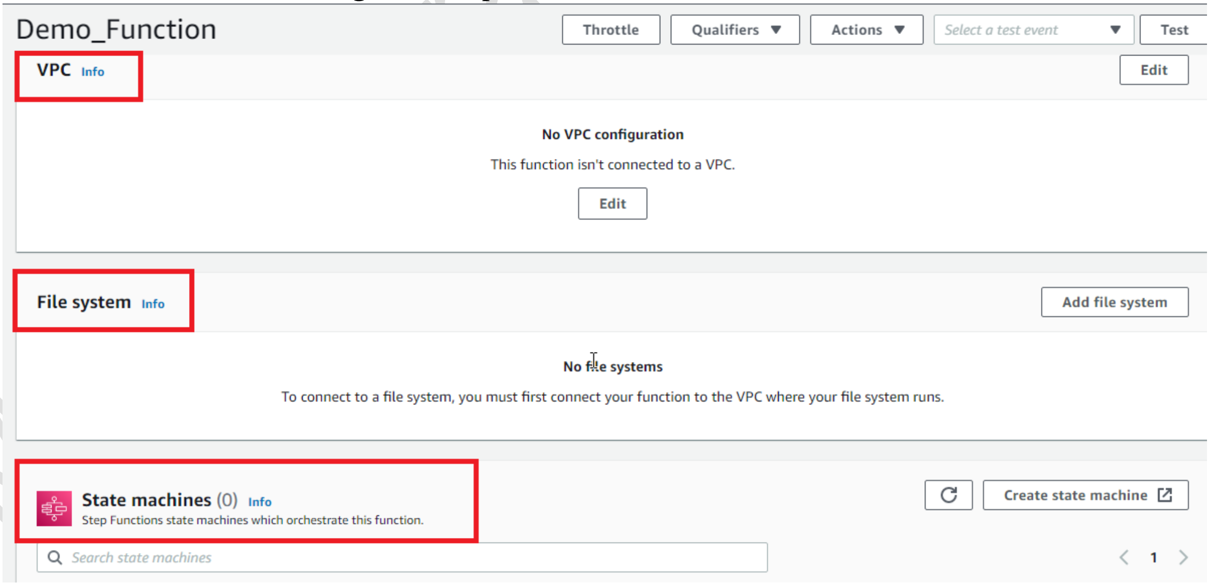
Tags: It is a label that we can use. It is use to search and filter our resources or track our AWS costs

Basic settings: From this section we can set the memory and Time out setting for the lambda function.

Monitoring tools: From this section we can able to manage monitoring settings for our lambda function including Active tracing.

Enable active tracing: By activating this option it will integrate AWS X-Ray to trace the event source that invoked our lambda function.

9. Scroll down for more configuration options



Demo_Function

Throttle | Qualifiers ▼ | Actions ▼ | Select a test event ▼ | Test

VPC Info — Edit

No VPC configuration

This function isn't connected to a VPC.

Edit

File system Info — Add file system

No file systems

To connect to a file system, you must first connect your function to the VPC where your file system runs.

State machines (0) Info

Step Functions state machines which orchestrate this function.

Create state machine

Search state machines

< 1 >

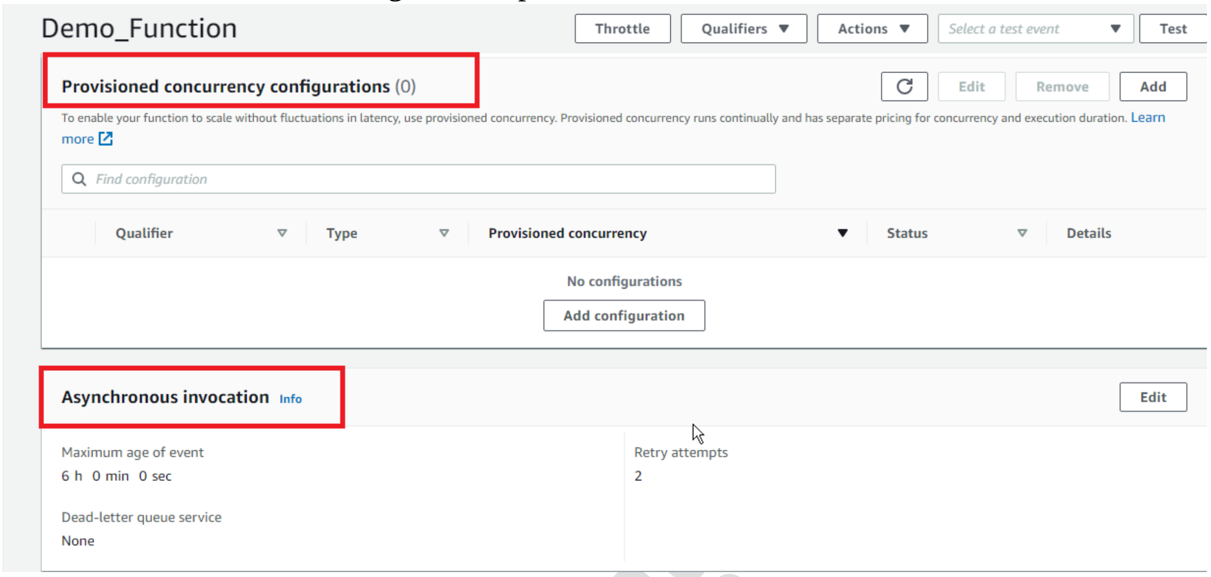
Network Section: The network section provides us with the capability of allowing our function to access resources via VPC. AWS lambda assigns ENIs (Elastic Network Interface) to our resources using a private IP address.

Note: It is important to note that the previous default ability to accessing publicly accessible resources over the internet is removed. To overcome this, we must attach the function to a private subnet which has access to a NAT instance or Gateway. Do not attach it to a public subnet.

Author: Saurav Raghuvanshi

File system: from this we can associate an existing Amazon Elastic File System (Amazon EFS) file system with our function.
State machines: From this feature we can able to orchestrate this function on Step Functions.

10. Scroll down to see more configuration options.



Demo_Function

Provisioned concurrency configurations (0)

To enable your function to scale without fluctuations in latency, use provisioned concurrency. Provisioned concurrency runs continually and has separate pricing for concurrency and execution duration. Learn more

Qualifier	Type	Provisioned concurrency	Status	Details

No configurations

Asynchronous invocation Info

Maximum age of event
6 h 0 min 0 sec

Retry attempts
2

Dead-letter queue service
None

Concurrency configuration: By default, we can have 1000 occurrences of lambda function running at once.
There are 2 Limits:
1. Account level limit (default 1000)
2. Function level limit (reserved)
Reserved concurrency ensures that your function will continue to run even if another function has huge no of requests.
Asynchronous invocation: From this setting we can able to set **Maximum age of event**(The maximum amount of time to keep unprocessed events in the queue), **Retry attempts**(The maximum number of times to retry when the function returns an error) and **Dead-letter queue**(we can send unprocessed events from an asynchronous invocations to an Amazon SQS queue or an Amazon SNS topic.