Creation of an AWS Lambda Function using CloudFormation

AWS Lambda is a serverless compute service: run code without worrying about servers. You pay per execution, and get up to a million free executions a month.

Lambda functions are usually built to run in response to computing events.

Step 1: Write a Template

```
Resources:
  HelloLambdaRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: HelloLambdaRole
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
  HelloLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      FunctionName: HelloLambdaFunction
      Role: !GetAtt HelloLambdaRole.Arn
      Runtime: python3.7
      Handler: index.my_handler
      Code:
```

```
ZipFile: |
  def my_handler(event, context):
    message = 'Hello Lambda World!'
    return message
```

Here are a few more details about the Lambda function properties:

The Resource is named “HelloLambdaFunction”. We can use any alphanumeric (A-Za-z0-9) we like as long as it’s unique to the template.

FunctionName does not have to be the same as the Resource‘s name, but it’s cleaner to keep them the same. The FunctionName is what’s stored within your AWS account. If you do not specify a name here, AWS will generate one for you.

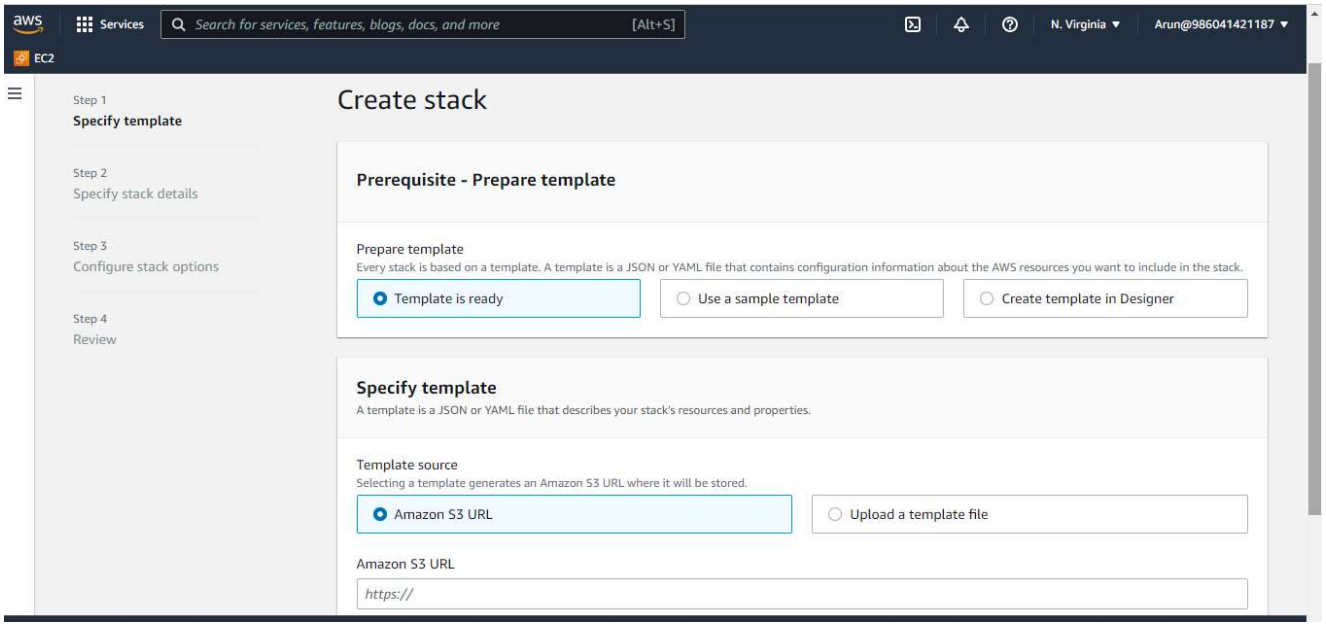
Role is where the function gets its privileges, and it is required. You created the HelloLambdaRole role so you could supply it here. CloudFormation wants the role’s ARN (a long string AWS assigns each resource), not the friendly name you gave it. Since we don’t know the role’s ARN because it hasn’t been created yet, we use a handy intrinsic function: !GetAtt(‘Get Attribute’), supplying it your friendly name and the attribute keyword Arn. The intrinsic function will return our new role’s ARN for us. There are various ways to call the intrinsic function. In the above example, the YAML used:! GetAtt HelloLambdaRole.Arn and JSON used: "Fn::GetAtt": ["HelloLambdaRole","Arn"].

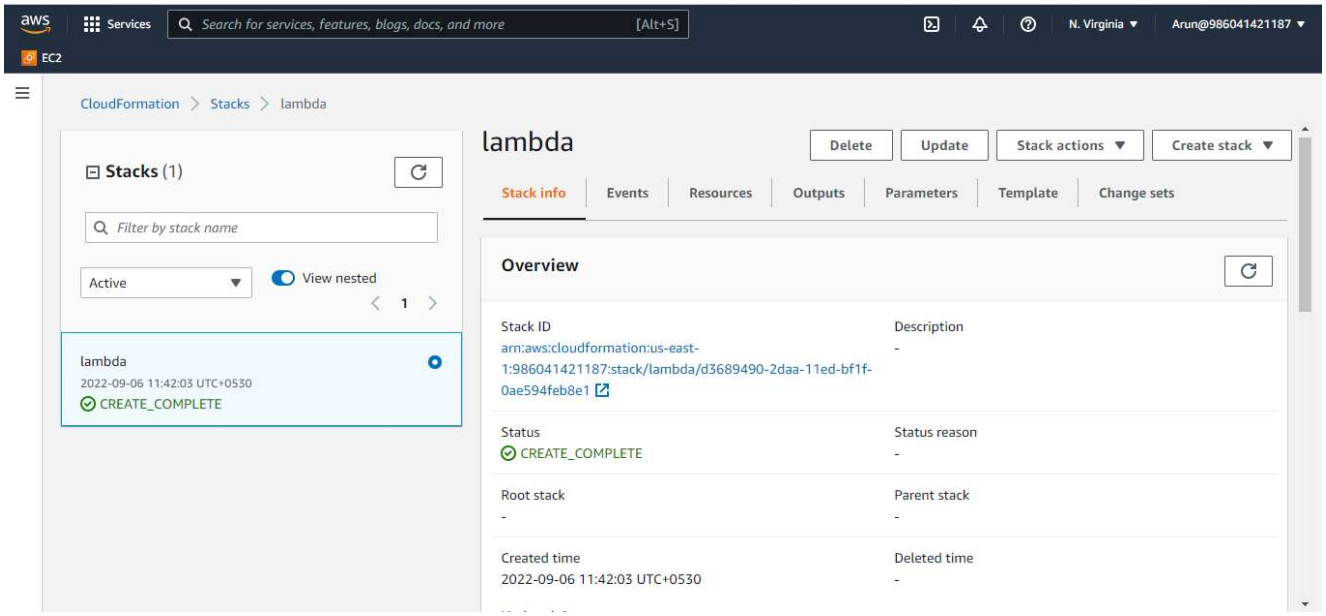
Runtime is the function’s programming language, and you have a set to pick from. Click here for valid Runtime values.

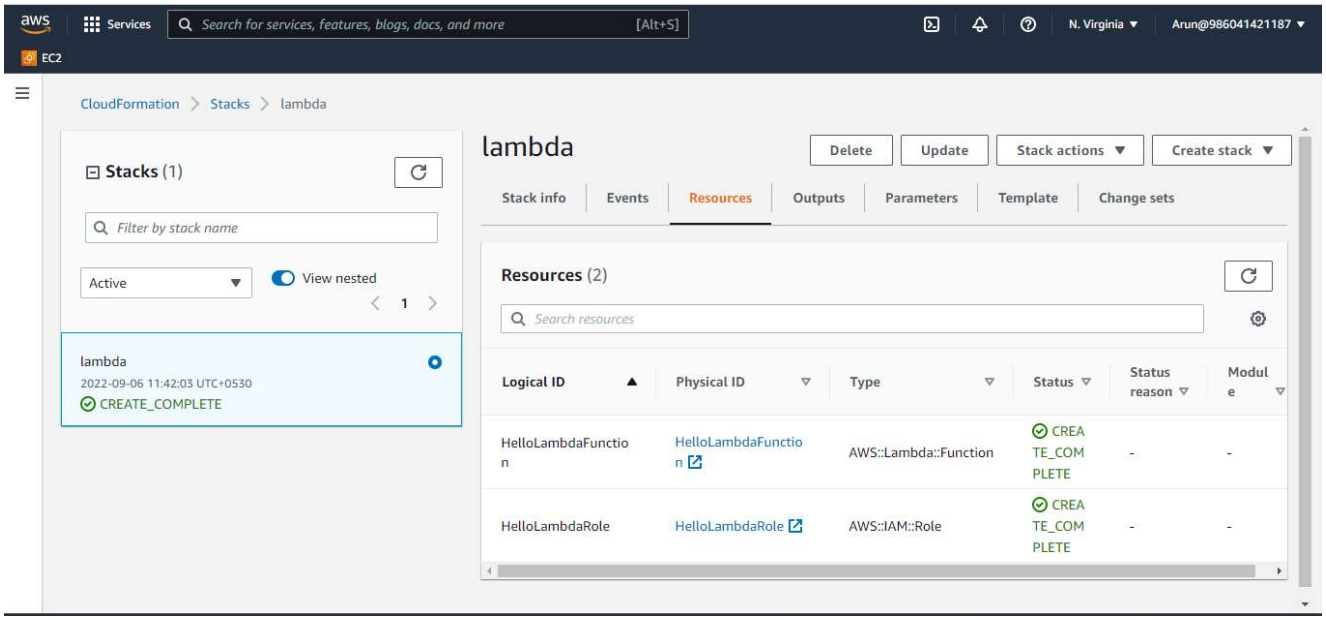
The Handler specifies what function, within the Lambda code, should be invoked. Here, index.my_handler means invoke the my_handler function in the file named index (see next bullet point).

Zipfile: means you are supplying the Runtime code ‘inline’ (within this template, not an external file.) Since you’re using ‘inline’ code, CloudFormation will zip your code for you, storing it in a file named index. This is why the Handler value you supplied above was index.myhandler (<filename>.<runtime_function_name>).

Step 2. Create the Stack

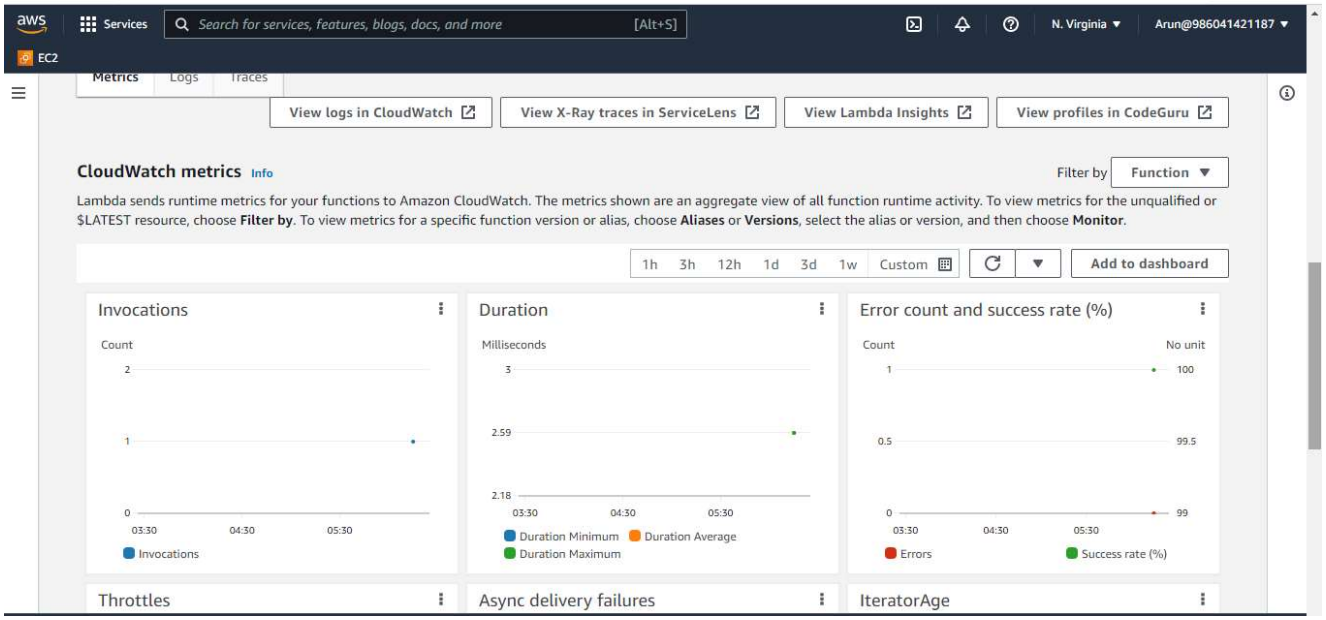

Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
Arun@986041421187
EC2
Step 1
Specify template
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review
Create stack
Prerequisite - Prepare template
Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.
Template is ready
Use a sample template
Create template in Designer
Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.
Template source
Selecting a template generates an Amazon S3 URL where it will be stored.
Amazon S3 URL
Upload a template file
Amazon S3 URL
https://


Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
Arun@986041421187
EC2
CloudFormation
Stacks
lambda
Stacks (1)
Filter by stack name
Active
View nested
1
lambda
2022-09-06 11:42:03 UTC+0530
CREATE_COMPLETE
lambda
Delete
Update
Stack actions
Create stack
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Overview
Stack ID
arn:aws:cloudformation:us-east-1:986041421187:stack/lambda/d3689490-2daa-11ed-bf1f-0ae594feb8e1
Description
-
Status
CREATE_COMPLETE
Status reason
-
Root stack
-
Parent stack
-
Created time
2022-09-06 11:42:03 UTC+0530
Deleted time
-

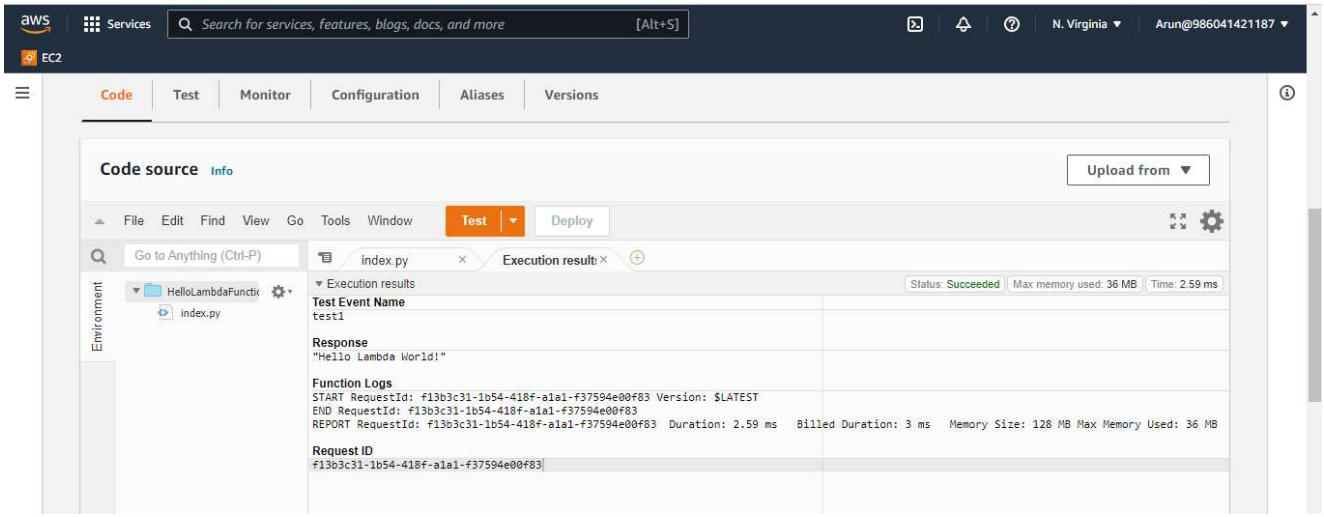


Step 3. Invoke the Lambda Function

Test the new Lambda function by manually invoking it, to simulate an event:



Successfully created an AWS Lambda Function and simulated an event to trigger it!


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
Arun@986041421187
EC2
Code
Test
Monitor
Configuration
Aliases
Versions
Code source
Info
Upload from
File
Edit
Find
View
Go
Tools
Window
Test
Deploy
Go to Anything (Ctrl-P)
index.py
Execution result
Environment
HelloLambdaFunctic
index.py
Execution results
Status: Succeeded
Max memory used: 36 MB
Time: 2.59 ms
Test Event Name
test1
Response
"Hello Lambda World!"
Function Logs
START RequestId: f13b3c31-1b54-418f-a1a1-f37594e00f83 Version: $LATEST
END RequestId: f13b3c31-1b54-418f-a1a1-f37594e00f83
REPORT RequestId: f13b3c31-1b54-418f-a1a1-f37594e00f83 Duration: 2.59 ms Billed Duration: 3 ms Memory Size: 128 MB Max Memory Used: 36 MB
Request ID
f13b3c31-1b54-418f-a1a1-f37594e00f83