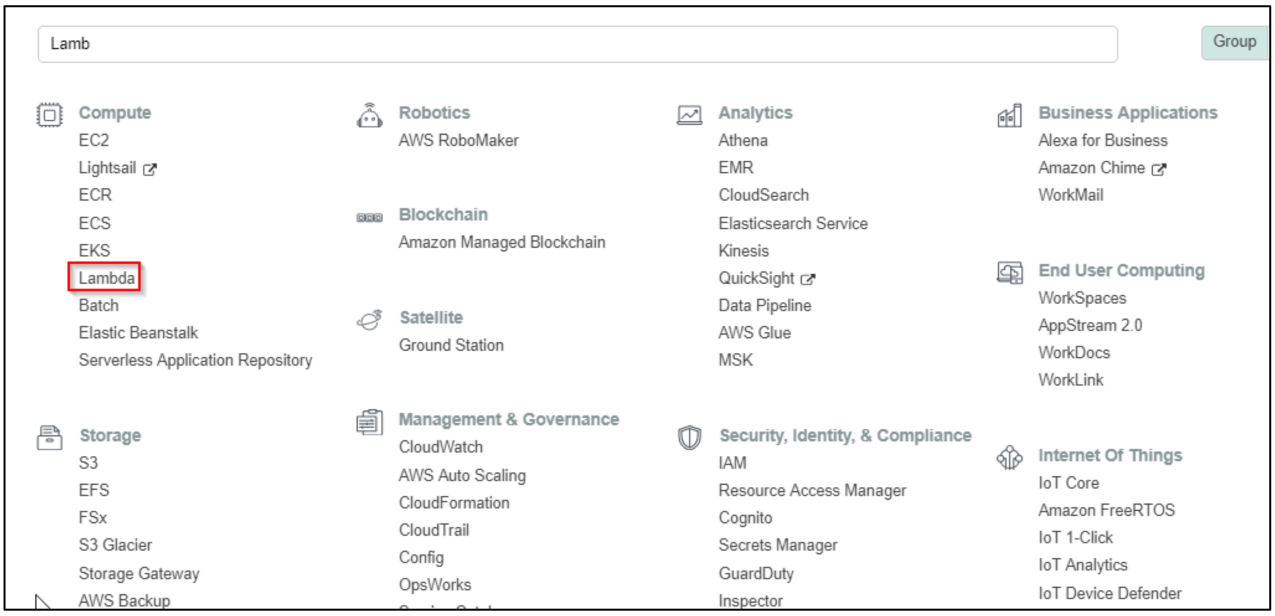
Module 3: Automating Infrastructure Using CloudFormation

Demo Document 4

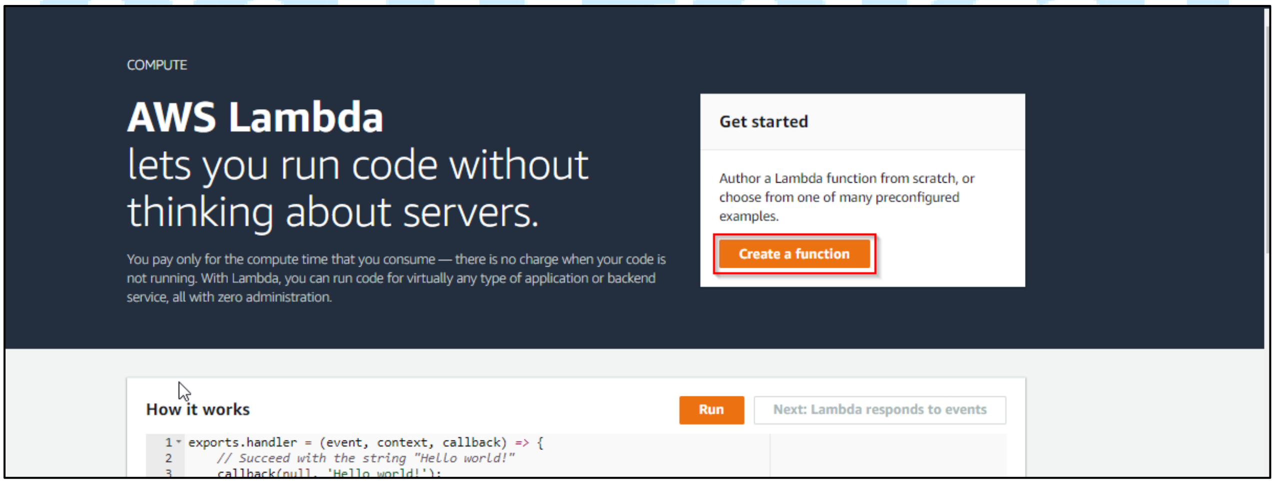
edureka!

Creating Lambda Backed Custom Resource

Step 1: Go to ***AWS Management Console*** and Select "Lambda"



Step 2: In Lambda Console click on " Create a function"

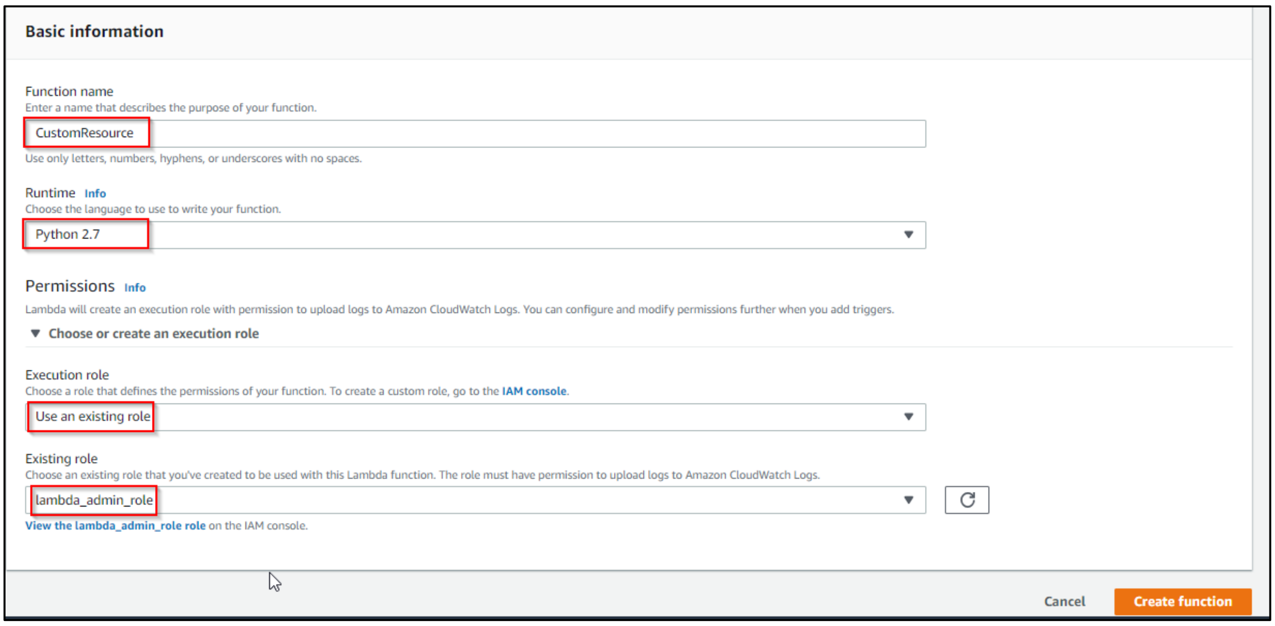

COMPUTE
AWS Lambda
lets you run code without thinking about servers.
Get started
Create a function
How it works
Run
Next: Lambda responds to events

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

```
exports.handler = (event, context, callback) => {
    // Succeed with the string "Hello world!"
    callback(null, 'Hello world!');
```

Step 3: Entered the required information and click on "create function"



Basic information

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

Runtime Info
Choose the language to use to write your function.
Python 2.7

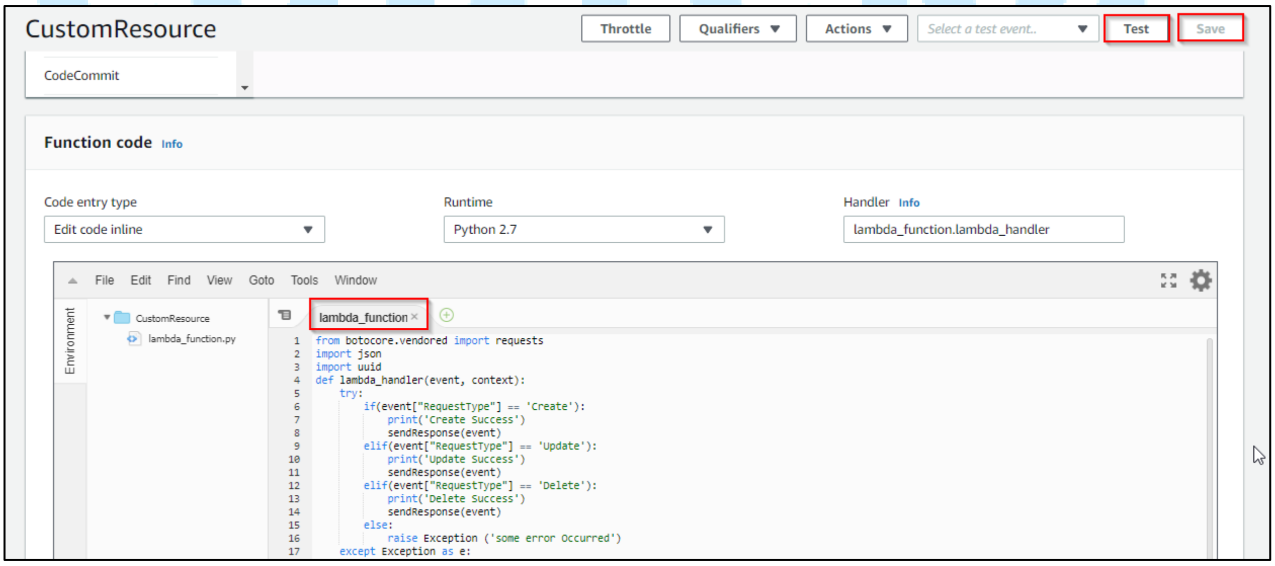
Permissions Info
Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ 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.
Use an existing role

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.
lambda_admin_role
View the lambda_admin_role role on the IAM console.

Step 4: Paste the python script-lambda function mentioned below, save it and later click on "Test"



Function code Info

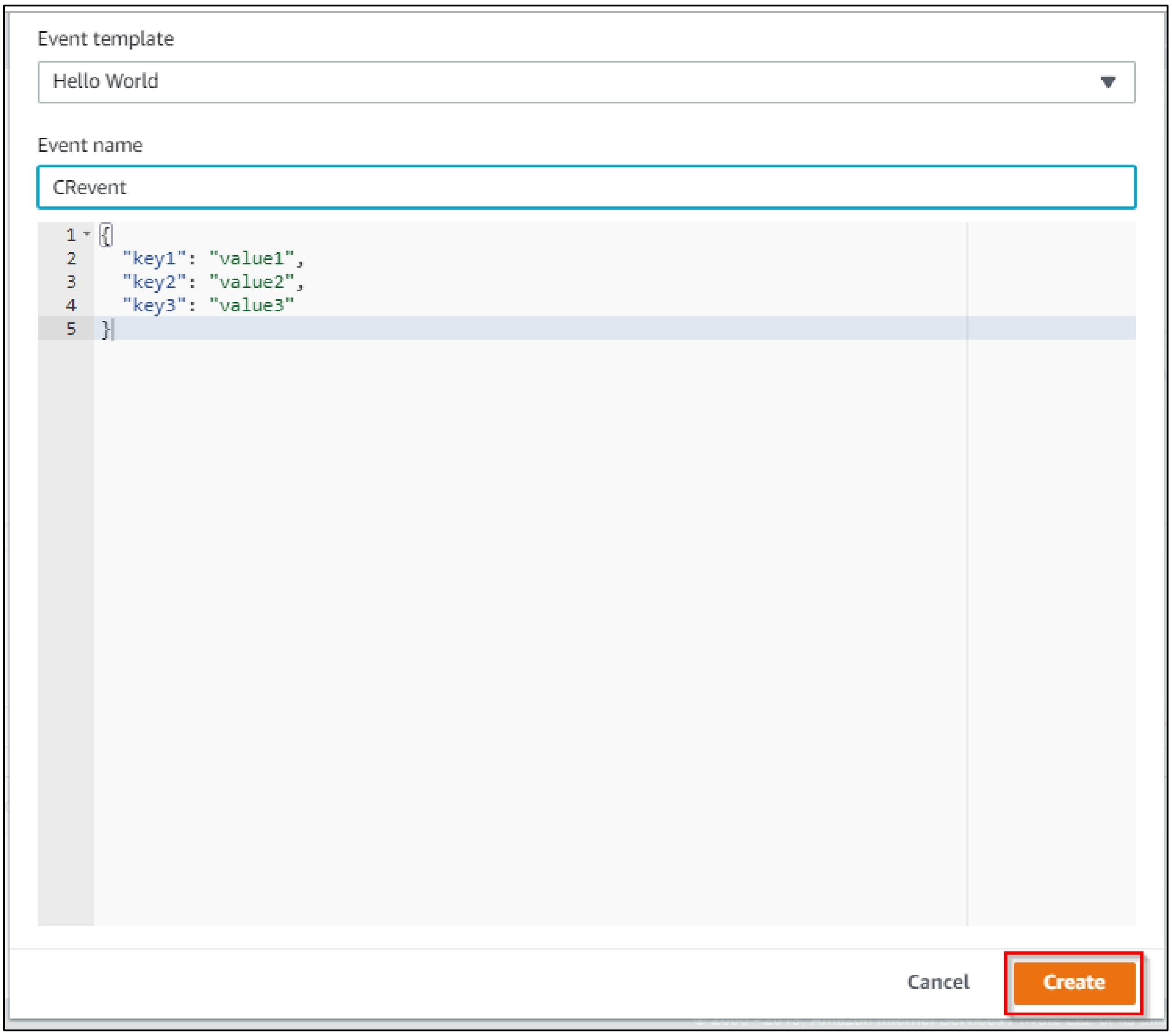
Code entry type: Edit code inline
Runtime: Python 2.7
Handler Info: lambda_function.lambda_handler

```
from botocore.vendored import requests
import json
import uuid
def lambda_handler(event, context):
    try:
        if(event["RequestType"] == 'Create'):
            print('Create Success')
            sendResponse(event)
        elif(event["RequestType"] == 'Update'):
            print('Update Success')
            sendResponse(event)
        elif(event["RequestType"] == 'Delete'):
            print('Delete Success')
            sendResponse(event)
        else:
            raise Exception ('some error Occurred')
    except Exception as e:
```

Lambda function:-

```
from botocore.vendored import requests
import json
import uuid
def lambda_handler(event, context):
    try:
        if(event["RequestType"] == 'Create'):
            print('Create Success')
            sendResponse(event)
        elif(event["RequestType"] == 'Update'):
            print('Update Success')
            sendResponse(event)
        elif(event["RequestType"] == 'Delete'):
            print('Delete Success')
            sendResponse(event)
        else:
            raise Exception ('some error Occurred')
    except Exception as e:
        print e
        respObj = {
                   "Status" : "FAILED",
                   "PhysicalResourceId" : "none",
                   "StackId" : event["StackId"],
                   "RequestId" : event["RequestId"],
                   "LogicalResourceId" : event['LogicalResourceId'],
                   "Data" : {
                        "key" : "none"
                   }
                }
        respJson=json.dumps(respObj)
        requests.put(event["ResponseURL"], data = respJson)
def sendResponse(event):
    respObj = {
               "Status" : "SUCCESS",
               "PhysicalResourceId" : "none",
               "StackId" : event["StackId"],
               "RequestId" : event["RequestId"],
               "LogicalResourceId" : event['LogicalResourceId'],
               "Data" : {
                    "key" : "none"
               }
            }
    respJson=json.dumps(respObj)
    requests.put(event["ResponseURL"], data = respJson)
```

Step 5: Enter event name and click on "Create"

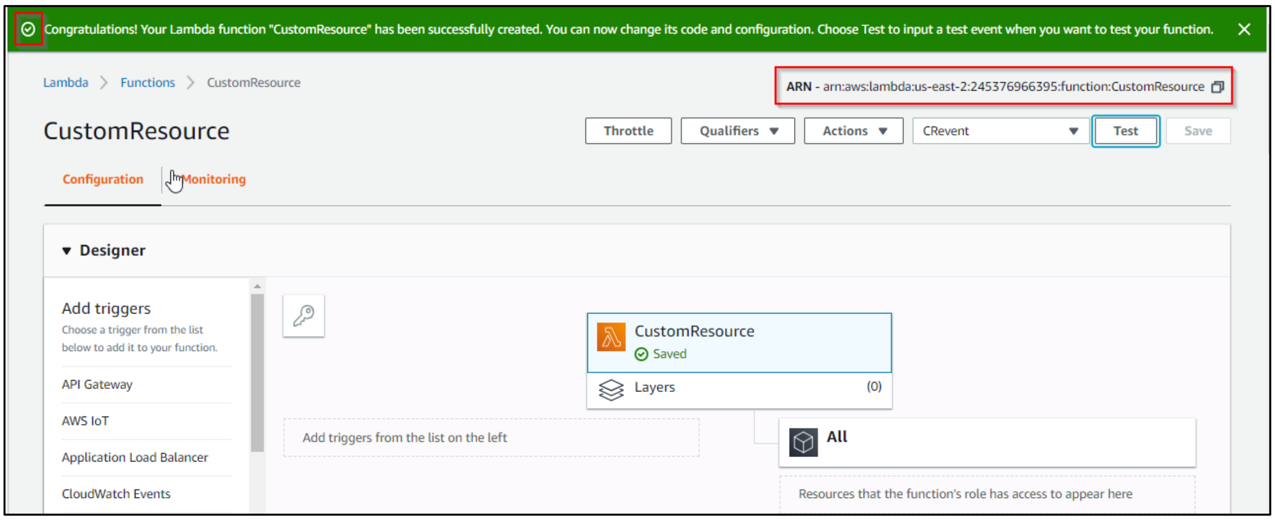


Event template: Hello World

Event name: CRevent

```
{
  "key1": "value1",
  "key2": "value2",
  "key3": "value3"
}
```

Step 6: If lambda function is created successfully then copy the ARN



Congratulations! Your Lambda function "CustomResource" has been successfully created. You can now change its code and configuration. Choose Test to input a test event when you want to test your function.

ARN - arn:aws:lambda:us-east-2:245376966395:function:CustomResource

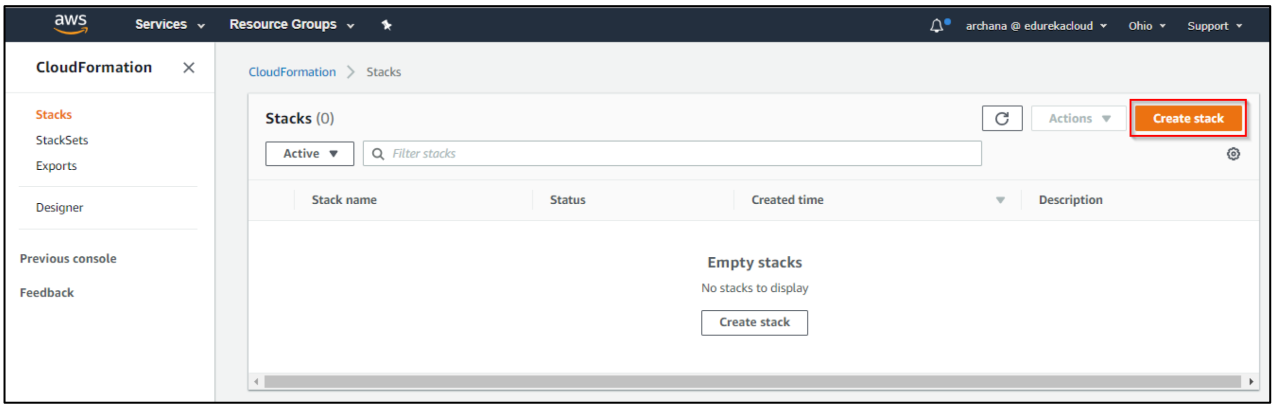
Lambda > Functions > CustomResource
CustomResource
Throttle
Qualifiers
Actions
CRevent
Test
Save
Configuration
Monitoring
Designer
Add triggers
Choose a trigger from the list below to add it to your function.
API Gateway
AWS IoT
Application Load Balancer
CloudWatch Events
CustomResource
Saved
Layers (0)
Add triggers from the list on the left
All
Resources that the function's role has access to appear here

Step 7: Write a ***CloudFormation Template*** to create a custom resource and enter the ***ARN*** to connect to the lambda function

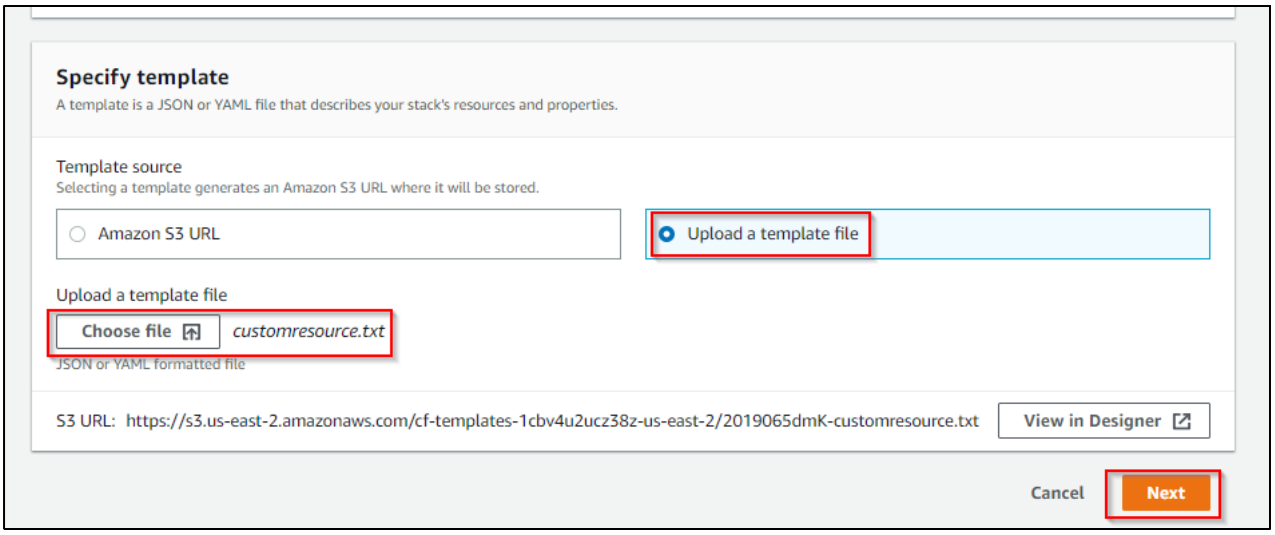
CloudFormation Template:-

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Description": "Test Custom resource",
    "Parameters": {
      "GroupDescription": {
        "Type": "String",
        "Default": "Enter data to be passed"
      }
    },
    "Resources": {
      "createSg": {
         "Type": "AWS::CloudFormation::CustomResource",
         "Version": "1.0",
         "Properties": {
            "ServiceToken": " arn:aws:lambda:us-east-
2:245376966395:function:CustomResource",
            "GroupDescription": { "Ref": "GroupDescription" }
         }
      }
    }
}
```

Step 8: Switch to ***CloudFormation console*** and click on "create stack"



Step 9: ***Upload*** the created CloudFormation Template and click on "Next"



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]

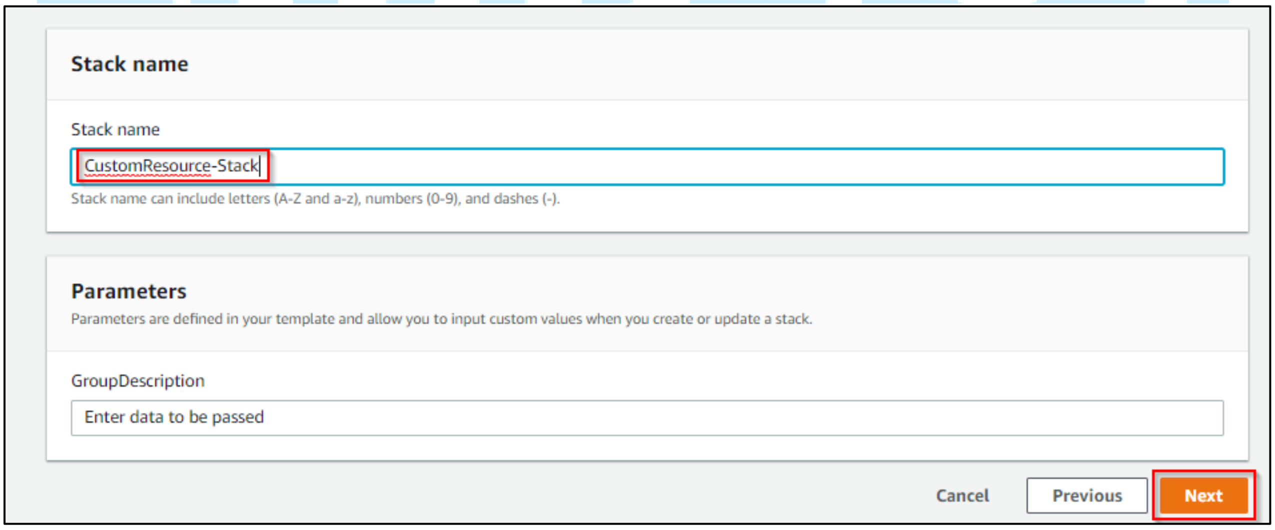
Upload a template file

Choose file *customresource.txt*

JSON or YAML formatted file

S3 URL: https://s3.us-east-2.amazonaws.com/cf-templates-1cbv4u2ucz38z-us-east-2/2019065dmK-customresource.txt

Step 8: Enter the ***name of Stack*** and click on "Next"



Stack name

Stack name: CustomResource-Stack

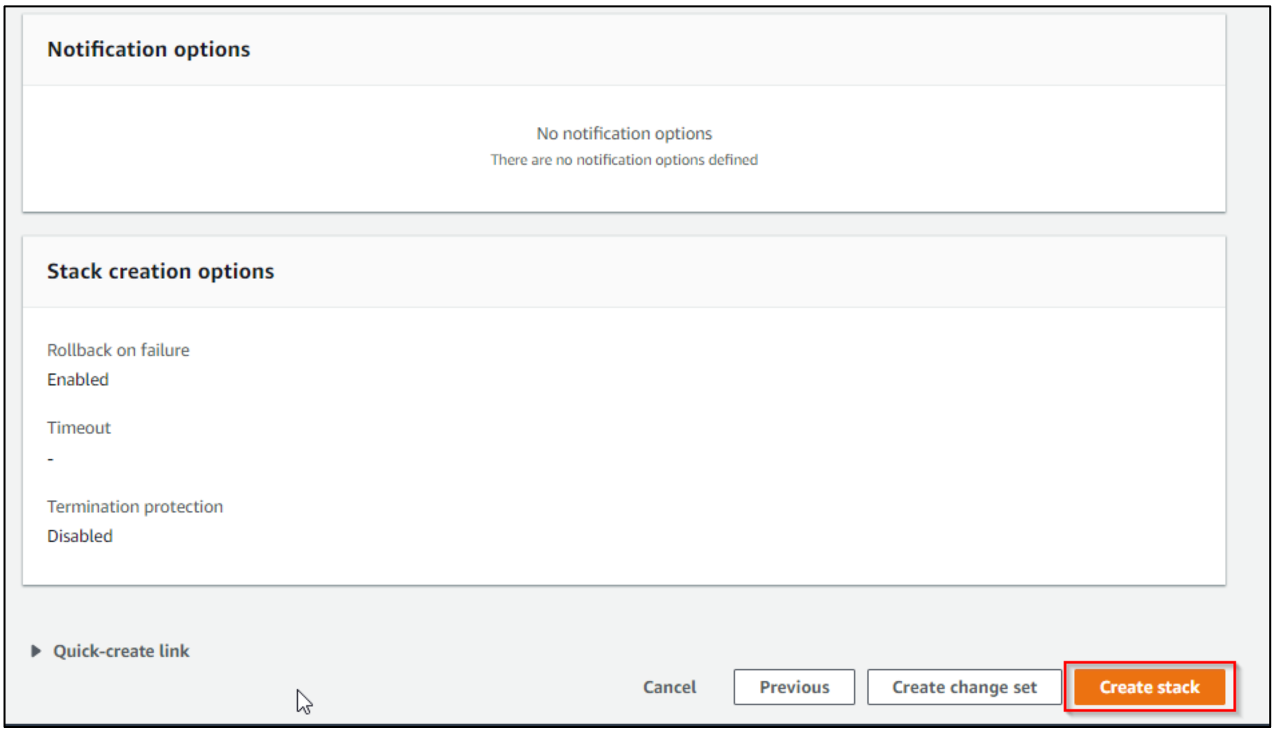
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

GroupDescription: Enter data to be passed

Step 9: ***Review*** the entered details and click on "Create Stack"



Notification options

No notification options
There are no notification options defined

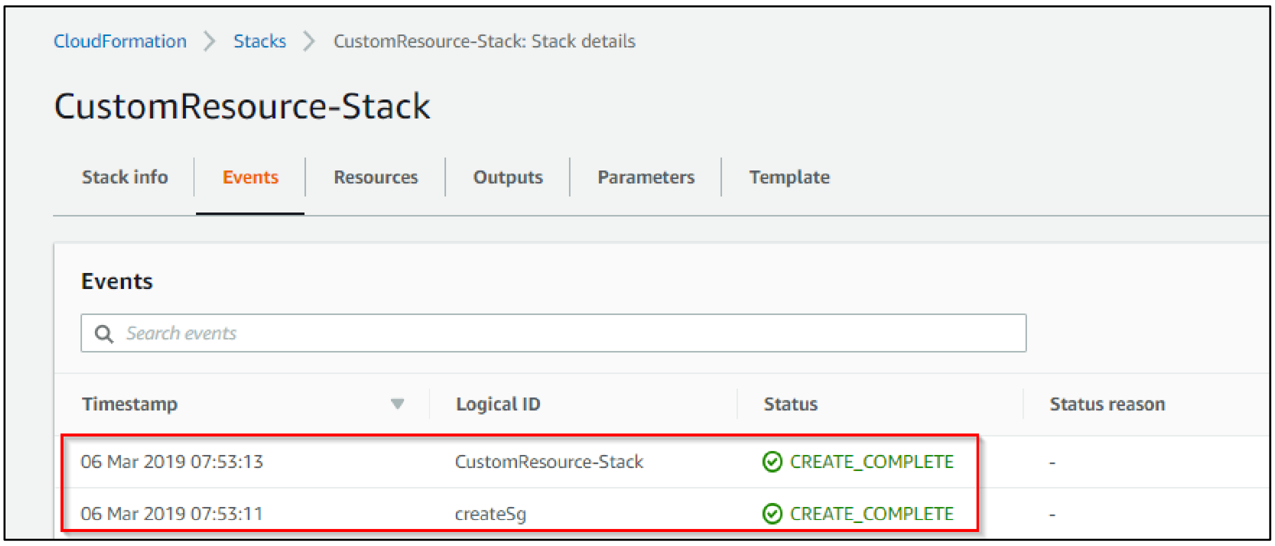
Stack creation options

Rollback on failure
Enabled

Timeout
-

Termination protection
Disabled

Step 10: If the stack is created ***successfully*** then its ***status*** changes from ***Create in-progress to Create complete*** (Refresh the console to check the status update)

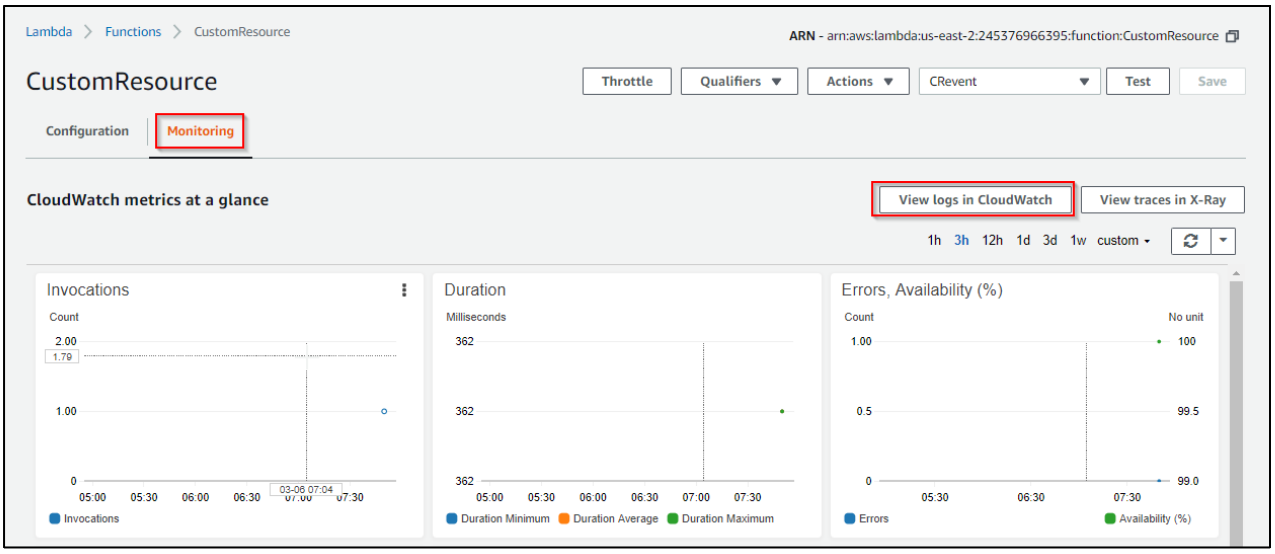


CustomResource-Stack

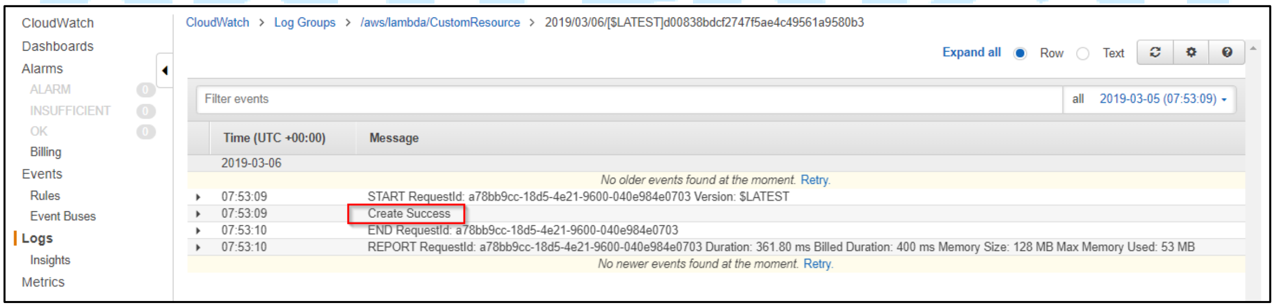
Events

Timestamp	Logical ID	Status	Status reason
06 Mar 2019 07:53:13	CustomResource-Stack	✓ CREATE_COMPLETE	-
06 Mar 2019 07:53:11	createSg	✓ CREATE_COMPLETE	-

Step 11: Switch back to ***Lambda console*** and click on Monitoring and later click on "View logs in CloudWatch"

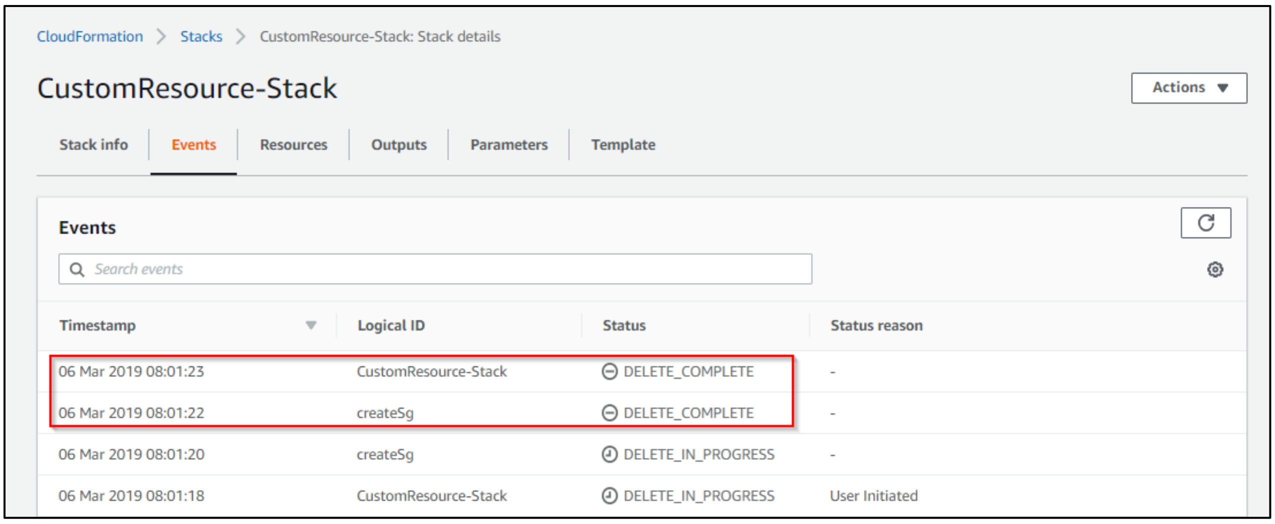


Step 11: You will get the ***printed statement*** of Status of CloudFormation Template


CloudWatch
Dashboards
Alarms
ALARM
INSUFFICIENT
OK
Billing
Events
Rules
Event Buses
Logs
Insights
Metrics
CloudWatch > Log Groups > /aws/lambda/CustomResource > 2019/03/06/[$LATEST]d00838bdcf2747f5ae4c49561a9580b3
Expand all
Row
Text
Filter events
all 2019-03-05 (07:53:09)

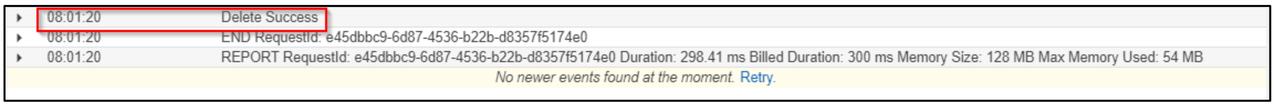
```
Time (UTC +00:00)   Message
2019-03-06
                    No older events found at the moment. Retry.
07:53:09            START RequestId: a78bb9cc-18d5-4e21-9600-040e984e0703 Version: $LATEST
07:53:09            Create Success
07:53:10            END RequestId: a78bb9cc-18d5-4e21-9600-040e984e0703
07:53:10            REPORT RequestId: a78bb9cc-18d5-4e21-9600-040e984e0703 Duration: 361.80 ms Billed Duration: 400 ms Memory Size: 128 MB Max Memory Used: 53 MB
                    No newer events found at the moment. Retry.
```

Step 12: Go back to CloudFormation console and ***Delete*** the stack



Timestamp	Logical ID	Status	Status reason
06 Mar 2019 08:01:23	CustomResource-Stack	DELETE_COMPLETE	-
06 Mar 2019 08:01:22	createSg	DELETE_COMPLETE	-
06 Mar 2019 08:01:20	createSg	DELETE_IN_PROGRESS	-
06 Mar 2019 08:01:18	CustomResource-Stack	DELETE_IN_PROGRESS	User Initiated

Step 12: Again you can check the ***printed statement*** of Status of CloudFormation Template



```
08:01:20    Delete Success
08:01:20    END RequestId: e45dbbc9-6d87-4536-b22b-d8357f5174e0
08:01:20    REPORT RequestId: e45dbbc9-6d87-4536-b22b-d8357f5174e0 Duration: 298.41 ms Billed Duration: 300 ms Memory Size: 128 MB Max Memory Used: 54 MB
            No newer events found at the moment. Retry.
```

Conclusion

We have successfully created Lambda backed Custom resource