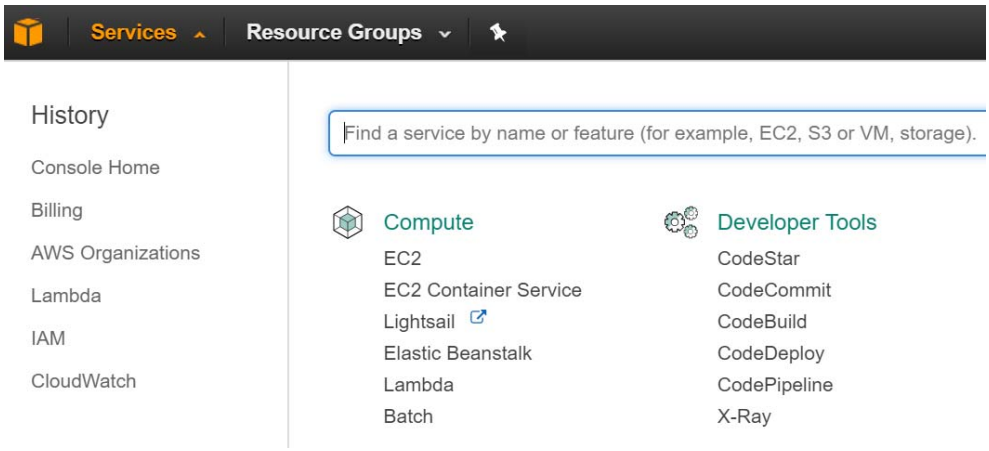
Run codes with AWS Lambda

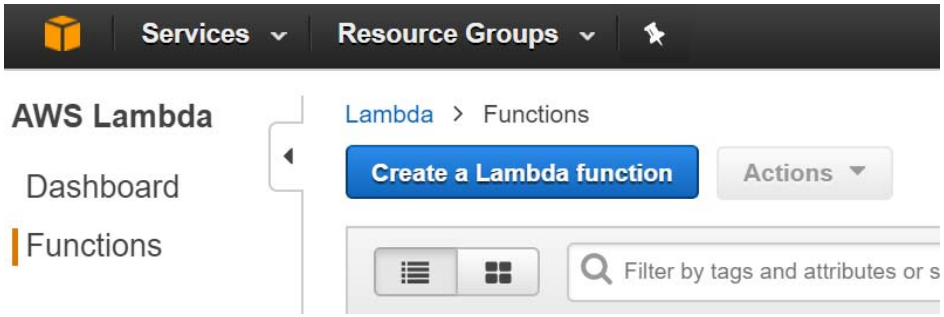
1. Python

- LambdaWithSNS

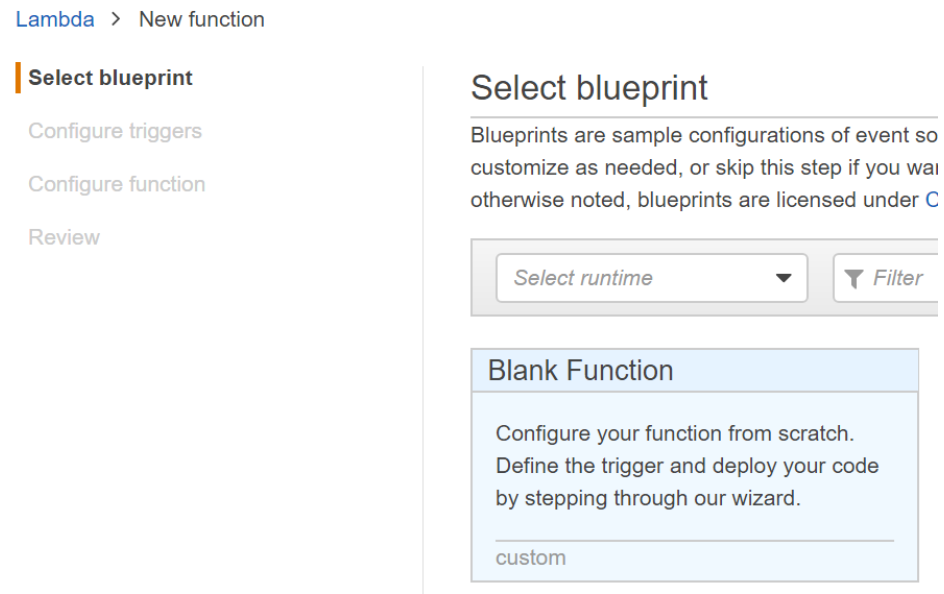
1) Log in AWS Console, Select Services -> Lambda



2) Create a Lambda function



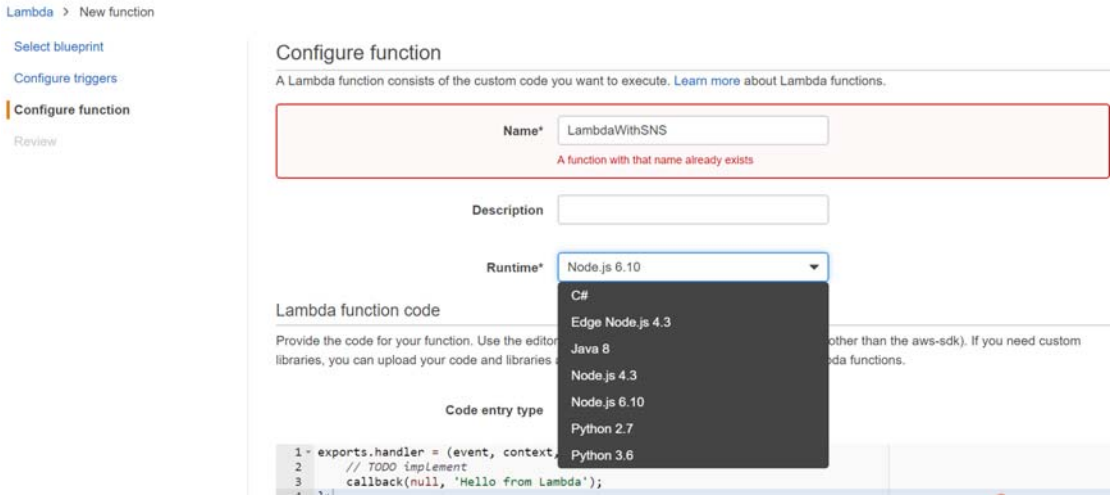
3) Select blueprint -> blank function



4) No need a trigger. Go to next step to configure the function

Name the function and choose the Runtime (Python 2.7)

AWS supports C#, Edge Node.js 4.3, Java 8, Node.js 4.3, Node.js 6.10, Python 2.7, and Python 3.6



5) Code input

AWS supports:

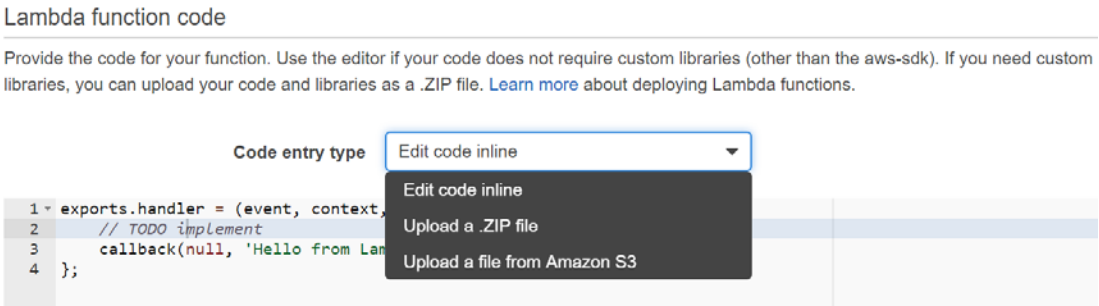
Edit code inline – Edit the code under the AWS console. Errors in the code will be shown.

Upload a .ZIP file – If there are multi files, we can zip the files and upload them. After creating the function, the codes will be shown in AWS console.

Upload a file from Amazon S3

For LambdaWithSNS, we have 1 file. Choose Edit code inline and input the code.

```
from __future__ import print_function
import json
print('Loading function')

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))
    message = event['Records'][0]['Sns']['Message']
    print("From SNS: " + message)
    return message
```



6) Lambda function handler and role

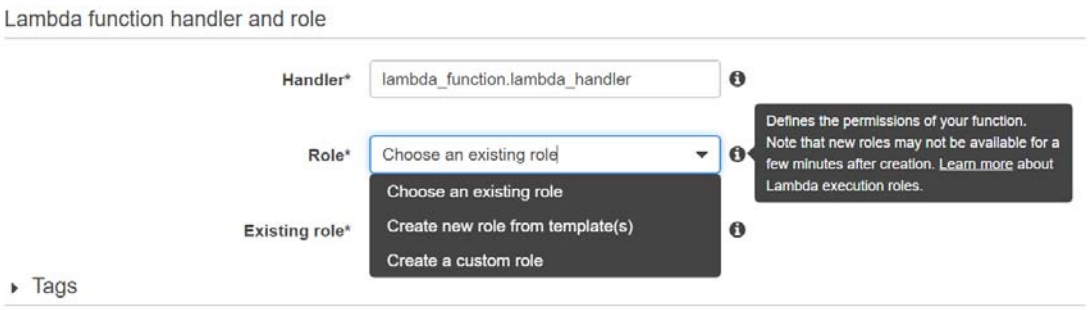
Hanlder: The filename.handler-method value in your function. For example, "main.handler" would call the handler method defined in main.py.

If the method name in the code is not lambda_hanlder shown in below figure, need change either one to keep constant.

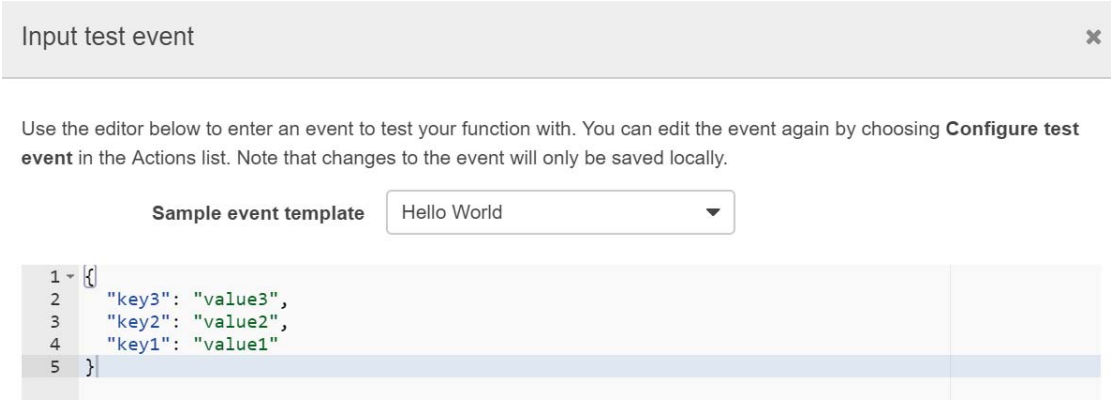
Role:

Choose an existing role: Defines the permissions of your function. Note that new roles may

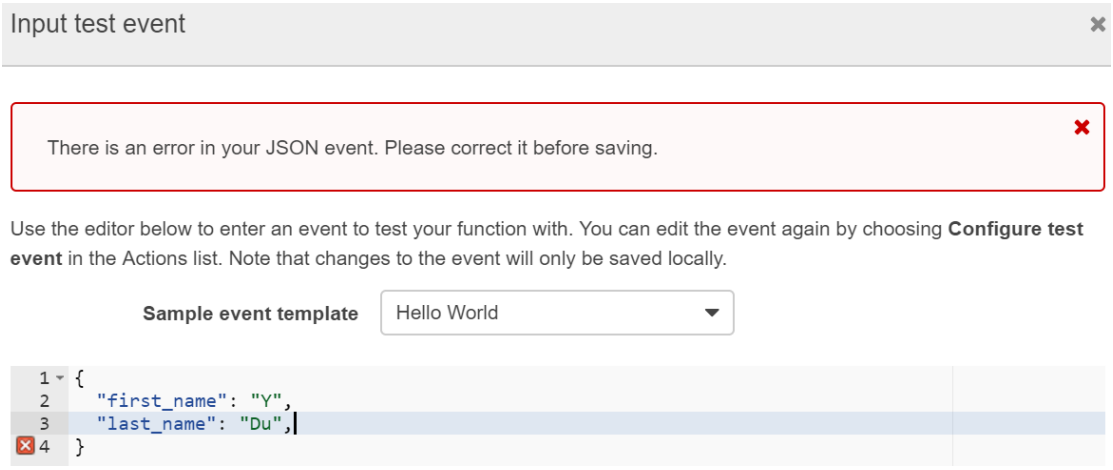
not be available for a few minutes after creation. Learn more about Lambda execution roles.
Existing role: need create a role before using 'Choose an existing role'. Here use AWSPowerUser.



7) Then create the function and test it.
The default test event is Hello World. Need choose different one to edit according to your needs.



If there is any error, AWS console will report it.



For LambdaWithSNS, select SNS as the Sample event template. We can modify the event as needed.

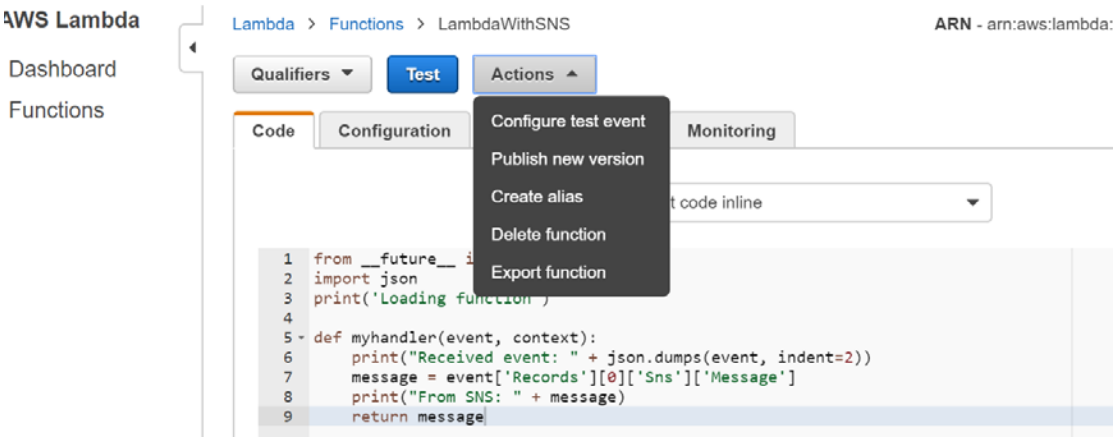
```
{
  "Records": [
    {
      "EventVersion": "1.0",
      "EventSubscriptionArn": "arn:aws:sns:EXAMPLE",
      "EventSource": "aws:sns",
```

```
        "Sns": {
          "SignatureVersion": "1",
          "Timestamp": "1970-01-01T00:00:00.000Z",
          "Signature": "EXAMPLE",
          "SigningCertUrl": "EXAMPLE",
          "MessageId": "95df01b4-ee98-5cb9-9903-4c221d41eb5e",
          "Message": "Hello from SNS!",          // message = event['Records'][0]['Sns']['Message']
          "MessageAttributes": {
            "Test": {
              "Type": "String",
              "Value": "TestString"
            },
            "TestBinary": {
              "Type": "Binary",
              "Value": "TestBinary"
            }
          },
          "Type": "Notification",
          "UnsubscribeUrl": "EXAMPLE",
          "TopicArn": "arn:aws:sns:EXAMPLE",
          "Subject": "TestInvoke"
        }
      }
    ]
}
```

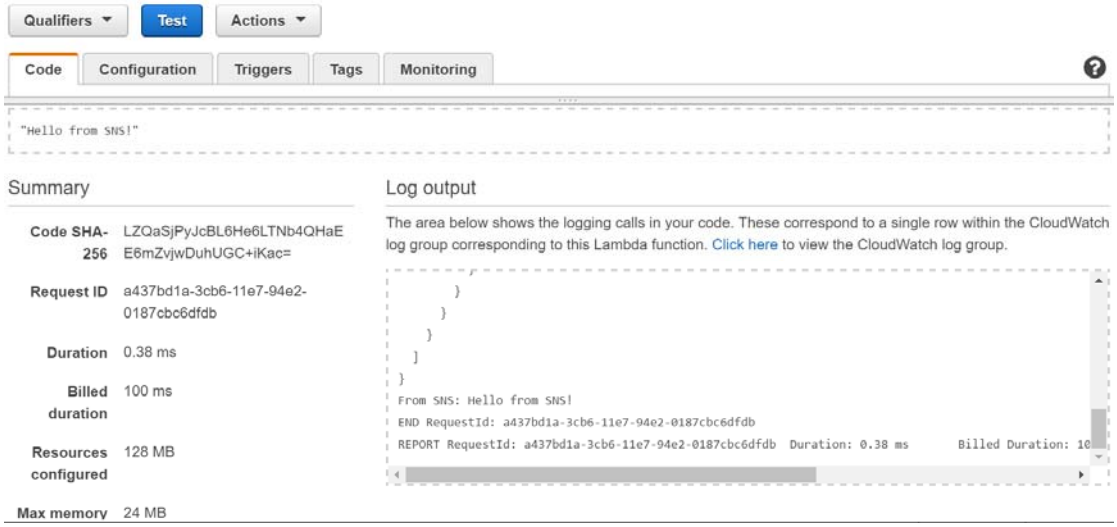
Sample event template SNS


```
{
  "Records": [
    {
      "EventVersion": "1.0",
      "EventSubscriptionArn": "arn:aws:sns:EXAMPLE",
      "EventSource": "aws:sns",
      "Sns": {
        "SignatureVersion": "1",
        "Timestamp": "1970-01-01T00:00:00.000Z",
        "Signature": "EXAMPLE",
        "SigningCertUrl": "EXAMPLE",
        "MessageId": "95df01b4-ee98-5cb9-9903-4c221d41eb5e",
        "Message": "Hello from SNS!",
        "MessageAttributes": {
          "Test": {
            "Type": "String",
            "Value": "TestString"
          },
          "TestBinary": {
            "Type": "Binary",
            "Value": "TestBinary"
          }
        },
        "Type": "Notification",
        "UnsubscribeUrl": "EXAMPLE",
        "TopicArn": "arn:aws:sns:EXAMPLE",
        "Subject": "TestInvoke"
      }
    }
  ]
}
```


8) Before running the test, we can re-configure the test event.



9) After running the test, the test results will be shown to list the duration, memory usage and the log output.



- ProcessDynamoDBStream

1) Code input

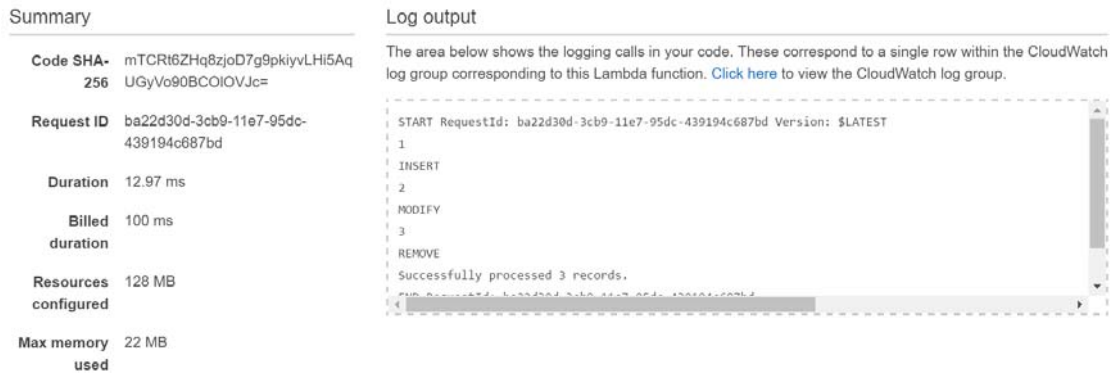
```
from __future__ import print_function

def lambda_handler(event, context):
    for record in event['Records']:
        print(record['eventID'])
        print(record['eventName'])
    print('Successfully processed %s records.' % str(len(event['Records'])))
```

2) DynamoDB event

```
{
  "Records": [
    {
      "eventID": "1",
      "eventVersion": "1.0",
      "dynamodb": {
        "Keys": {
          "Id": {
```

```
            "N": "101"
          }
        },
        "NewImage": {
          "Message": {
            "S": "New item!"
          },
          "Id": {
            "N": "101"
          }
        },
        "StreamViewType": "NEW_AND_OLD_IMAGES",
        "SequenceNumber": "111",
        "SizeBytes": 26
      },
      "awsRegion": "us-west-2",
      "eventName": "INSERT",
      "eventSourceARN":
"arn:aws:dynamodb:us-west-2:account-id:table/ExampleTableWithStream/stream/2015-06-27T0
0:48:05.899",
      "eventSource": "aws:dynamodb"
    },
  …
 ]
}
```


3) Result



2. Node.js

1) Code input

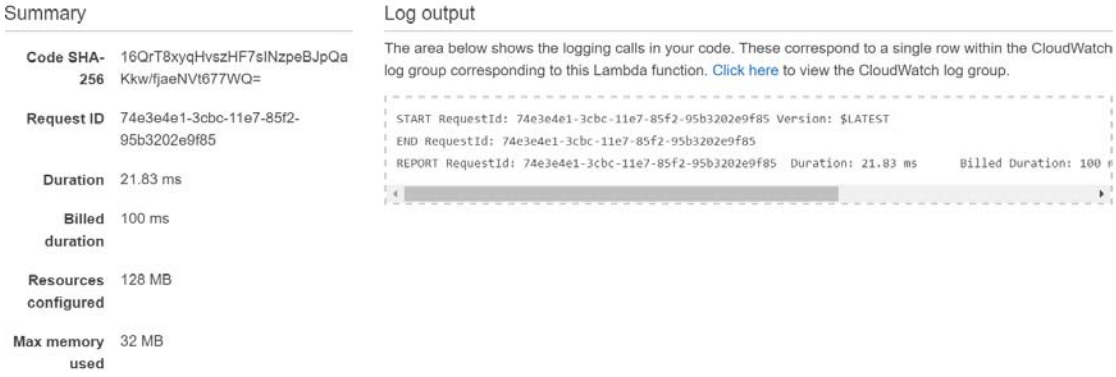
```
var AWS = require('aws-sdk');

    exports.handler = function(event, context, callback) {

        var bucketName = process.env.S3_BUCKET;
        callback(null, bucketName);
```

}

2) S3 put event

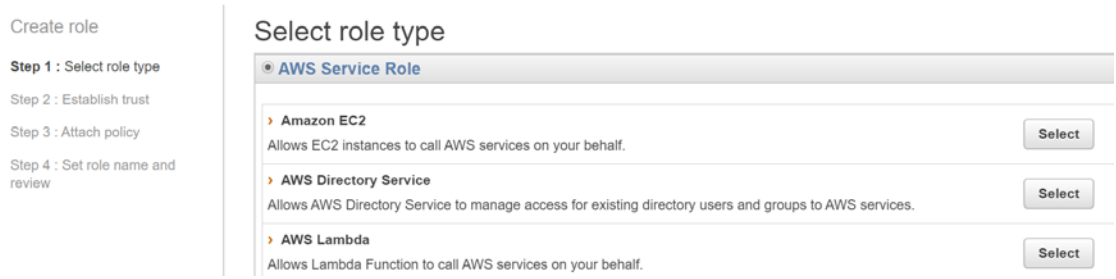

```
{
   "Records": [
      {
         "eventVersion": "2.0",
         "eventTime": "1970-01-01T00:00:00.000Z",
         "requestParameters": {
            "sourceIPAddress": "127.0.0.1"
         },
         "s3": {
            "configurationId": "testConfigRule",
            "object": {
               "eTag": "0123456789abcdef0123456789abcdef",
               "sequencer": "0A1B2C3D4E5F678901",
               "key": "index.js",
               "size": 1024
            },
            "bucket": {
               "arn": "arn:aws:s3:::mybucket",
               "name": "Nodejs",
               "ownerIdentity": {
                  "principalId": "EXAMPLE"
               }
            },
            "s3SchemaVersion": "1.0"
         },
         "responseElements": {
            "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH",
            "x-amz-request-id": "EXAMPLE123456789"
         },
         "awsRegion": "us-east-1",
         "eventName": "ObjectCreated:Put",
         "userIdentity": {
            "principalId": "EXAMPLE"
         },
         "eventSource": "aws:s3"
      }
   ]
}
```


3) Result



3. Create an Identity and Access Management (IAM) role

1) Select AWS Lambda



2) Search and select PowerUserAccess to create a new role