Proxima Centauri Sprint1

Web Health Monitor (Latency & Availability)

Documentation

Name	Muhammad Irfan Hassan
Date	19-12-2021

Contents

Project Description

This aim of this project to measure availability and latency of custom list (a json file placed in s3 bucket) using AWS CDK. It will update latency and availability after each 1 minu and will write metrics for latency and availability on cloud watch using cloud watch's API. Also set alarm to notify the subscriber when threshold for latency and availability is preached. Push SNS notification to subscribers using email address and also trigger lambda and store alarm data into dynamo dB when alarm generated.

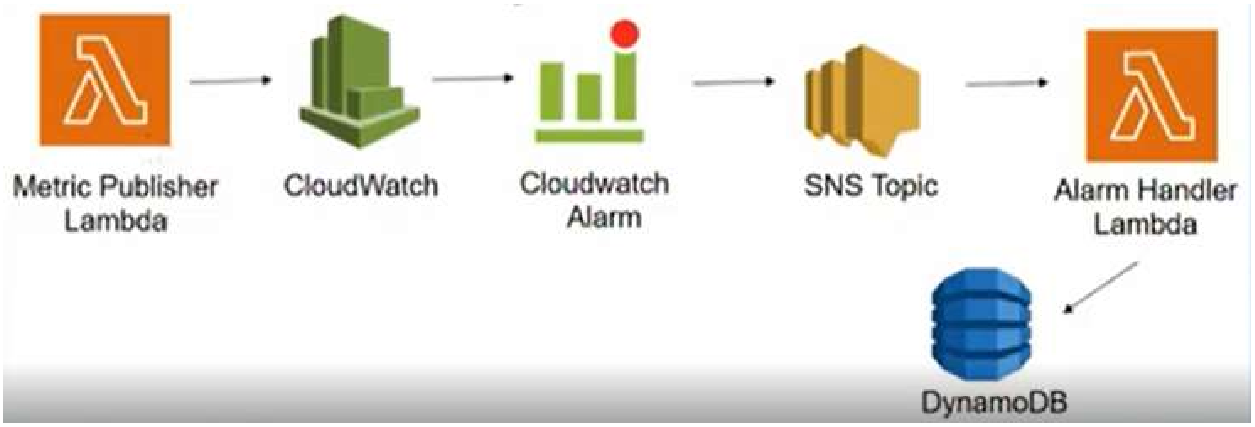


Figure 1: Web Health Monitor CDK Application

Technologies/Services

To build this application we will use the following AWS services

- Cloud9
- Lambda
- CloudWatch
- SNS
- Dynamo DB

Cloud9

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with an essential tool for popular programming languages, including JavaScript, Python, PHP, and more, don't need to install files or configure a development machine to start the project.

Lambda

Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging. With Lambda, you can run code for virtually any type of application or backend service.

CloudWatch

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real-time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached.

SNS

Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication. The A2A pub/sub functionality provides topics for high-throughput, push-based, many-to-many messaging between distributed systems, micro services, and event-driven server less applications. Using Amazon SNS topics, your publisher systems can fanout messages to a large number of subscriber systems, including Amazon SQS queues, AWS Lambda functions, HTTPS endpoints, and Amazon Kinesis Data Firehose, for parallel processing. The A2P functionality enables you to send messages to users at scale via SMS, mobile push, and email.

Dynamo DB

Amazon Dynamo DB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. Dynamo DB offers built-in security, continuous backups, automated multi-region replication, in-memory caching, and data export tools.

Setup

Before starting the project, we have to set up the environment and install requirements for the project. The steps for setup are following.

- First of all, log in to AWS amazon and create a virtual machine.
- check the version of python and if it is an old version check new version is available then make a new version as the default version.

```
python --version
python3 --version
source ~/.bashrc
alis python='/usr/bin/python3' (press ESC on keybaord)
:w! (press Enter on keybaord)
:q! (press Enter on keybaord)
```

- check the version of AWS and if it is an old version then update it to the new version.

```
aws --version
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

- create a directory of your choice and change the directory to new created

```
mkdir IrfanskipQ_Project1
cd IrfanskipQ_Project1
```

- Create CDK project in python language

```
cdk init app --language python
```

- Install all requirements for project.

```
python -m pip install aws-cdk.core==1.135.0
python -m pip install -r requirements.txt
nvm install v16.3.0 && nvm use 16.3.0 && nvm alias default v16.3.0
npm install -g aws-cdk
export PATH=$PATH:$(npm get prefix)/bin
python -m pip install aws-cdk.aws-s3 aws-cdk.aws-lambda
```

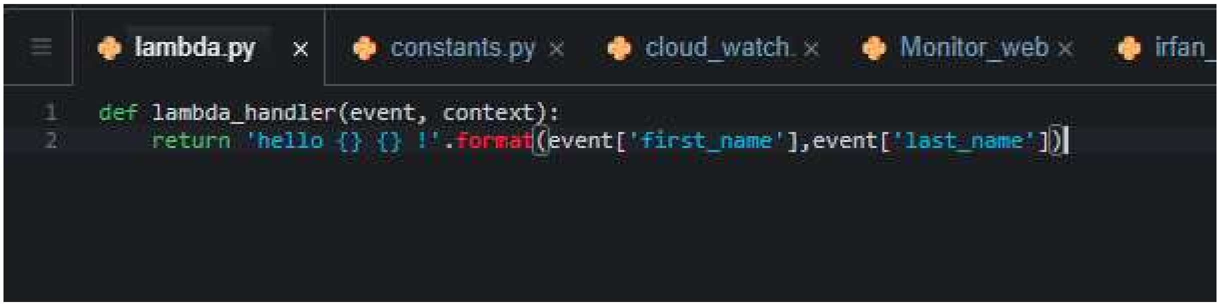
- Create a new folder resource and add a new lambda.py file.

Project milestones

I have divided my project into subtasks and then completed subtasks on daily basis. Let's discuss each subtask in details.

Implementing Hello Lambda Function:

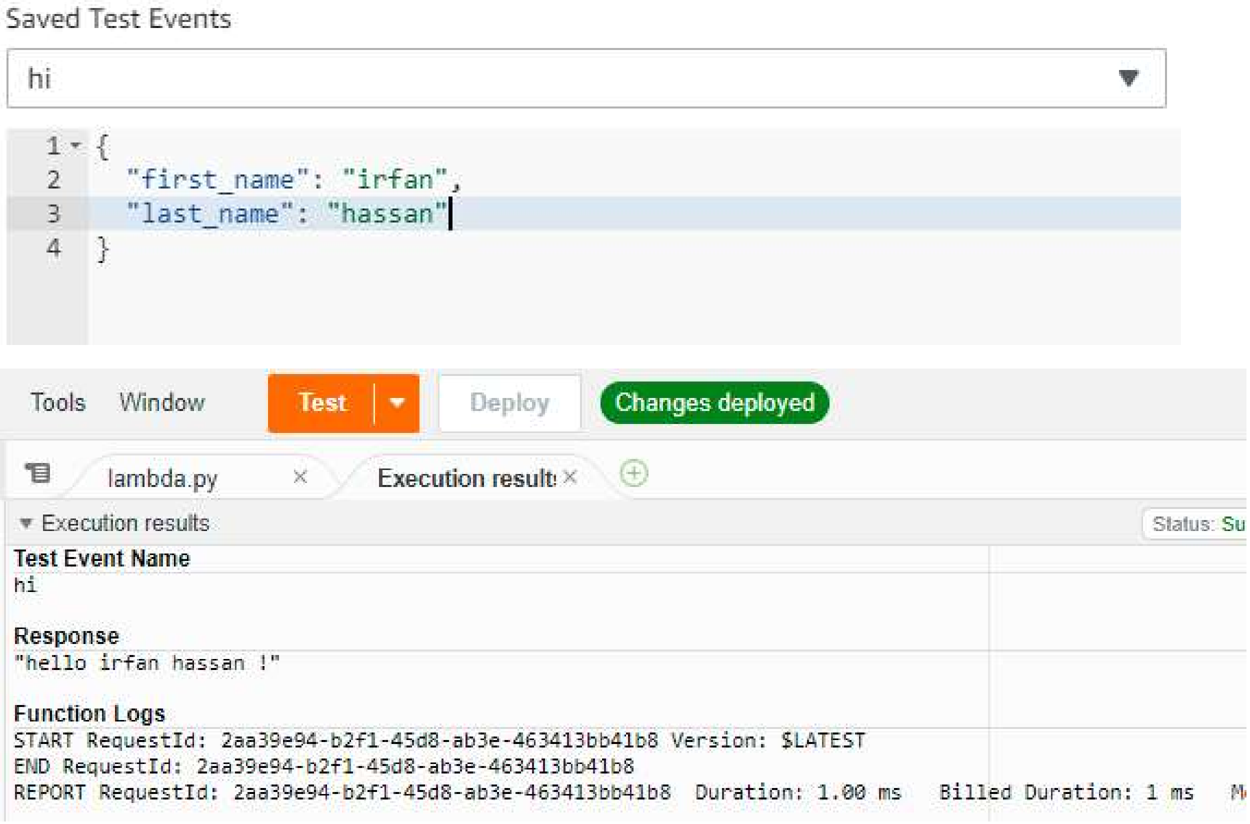
First I started with hellolambda function using lambda handler. In this function it takes two string as input "first_name" and "last_name". code for hellolambda function is given below.



```
def lambda_handler(event, context):
    return 'hello {} {} !'.format(event['first_name'],event['last_name'])
```

Figure 2: Hello Lambda function

To test this function, I put my first name and last name and here is output of this function.



```
{
  "first_name": "irfan",
  "last_name": "hassan"
}
```

Test Event Name
hi

Response
"hello irfan hassan !"

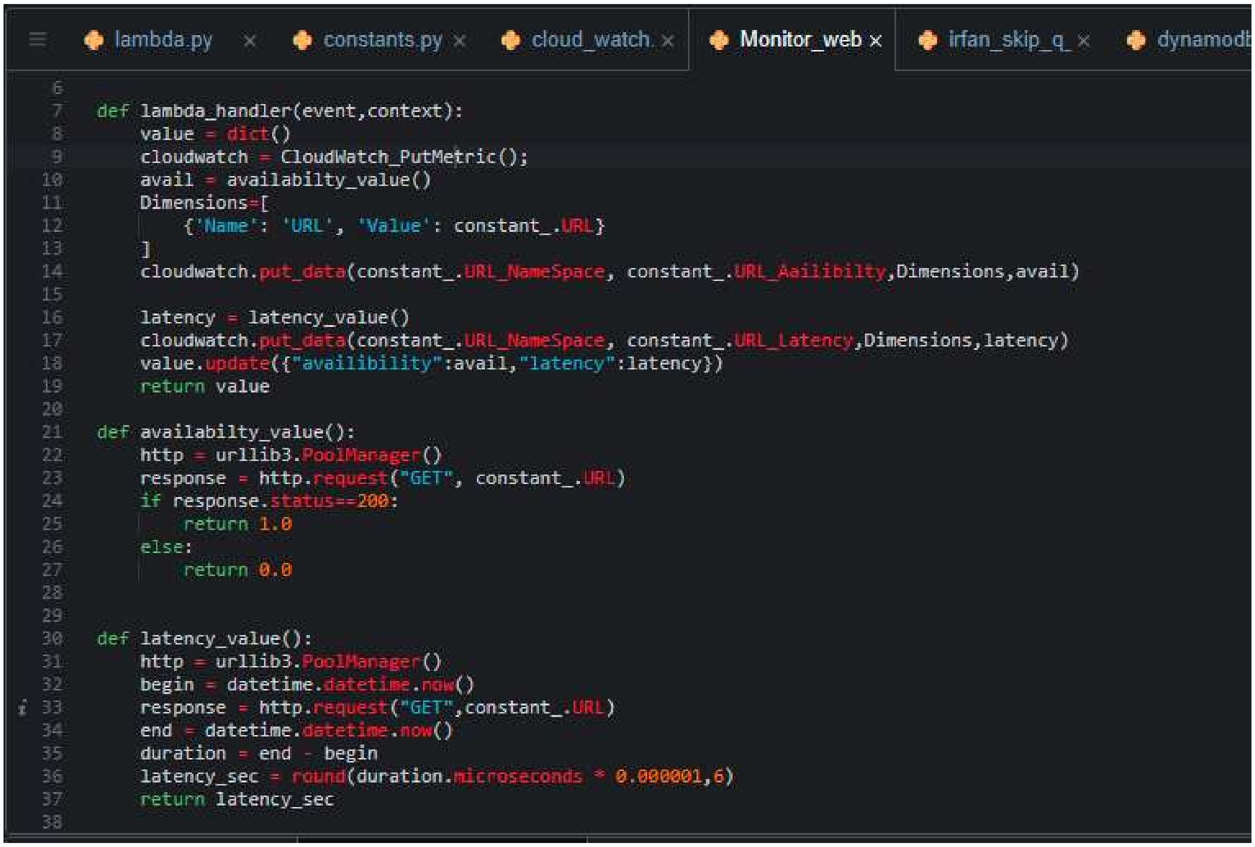
Function Logs

```
START RequestId: 2aa39e94-b2f1-45d8-ab3e-463413bb41b8 Version: $LATEST
END RequestId: 2aa39e94-b2f1-45d8-ab3e-463413bb41b8
REPORT RequestId: 2aa39e94-b2f1-45d8-ab3e-463413bb41b8  Duration: 1.00 ms   Billed Duration: 1 ms   M
```

Figure 3:Testing hellolambda

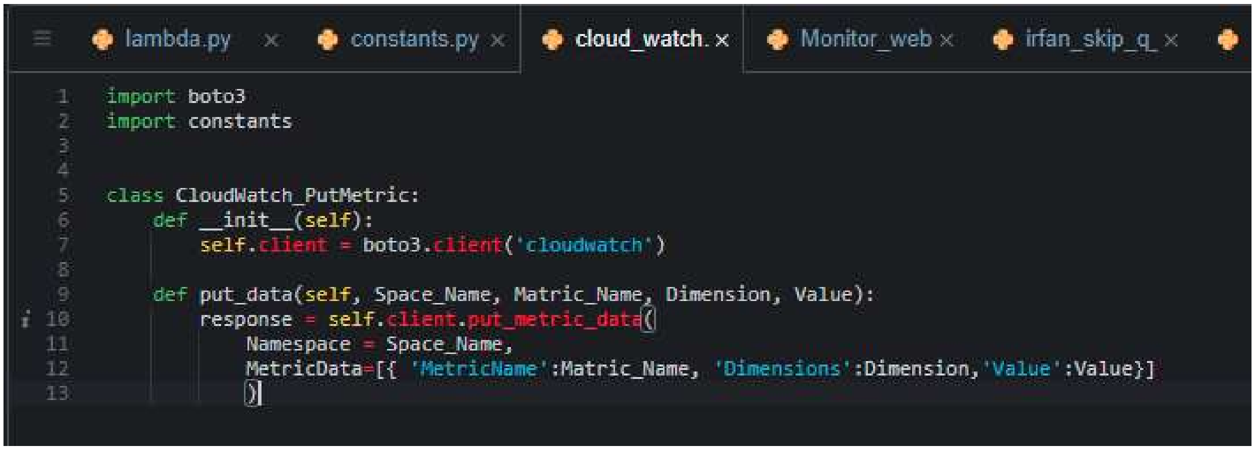
Availability and latency of webpage by using periodic lambda function

Then we create another lambda function, which trigger after 1 minutes and it check availability and latency of webpage (URL will be given) and store metric for latency and availability on cloud watch.



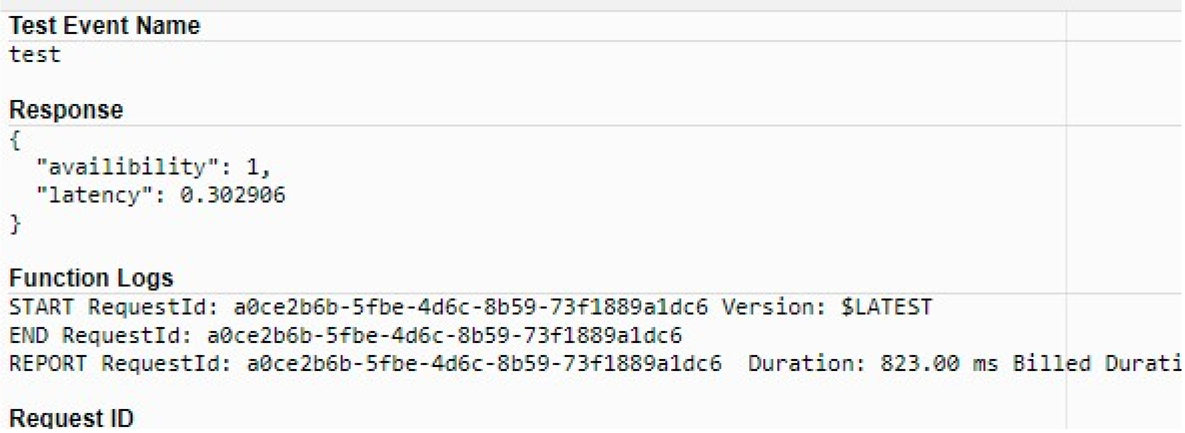
```python

def lambda_handler(event,context):
    value = dict()
    cloudwatch = CloudWatch_PutMetric();
    avail = availabilty_value()
    Dimensions=[
        {'Name': 'URL', 'Value': constant_.URL}
    ]
    cloudwatch.put_data(constant_.URL_NameSpace, constant_.URL_Aailibilty,Dimensions,avail)

    latency = latency_value()
    cloudwatch.put_data(constant_.URL_NameSpace, constant_.URL_Latency,Dimensions,latency)
    value.update({"availibility":avail,"latency":latency})
    return value

def availabilty_value():
    http = urllib3.PoolManager()
    response = http.request("GET", constant_.URL)
    if response.status==200:
        return 1.0
    else:
        return 0.0


def latency_value():
    http = urllib3.PoolManager()
    begin = datetime.datetime.now()
    response = http.request("GET",constant_.URL)
    end = datetime.datetime.now()
    duration = end - begin
    latency_sec = round(duration.microseconds * 0.000001,6)
    return latency_sec

```

Figure 4: Periodic lambda for availability and latency of webpage



```python
import boto3
import constants


class CloudWatch_PutMetric:
    def __init__(self):
        self.client = boto3.client('cloudwatch')

    def put_data(self, Space_Name, Matric_Name, Dimension, Value):
        response = self.client.put_metric_data(
            Namespace = Space_Name,
            MetricData=[{ 'MetricName':Matric_Name, 'Dimensions':Dimension,'Value':Value}]
            )
```

Figure 5:putting metric on Cloud watch

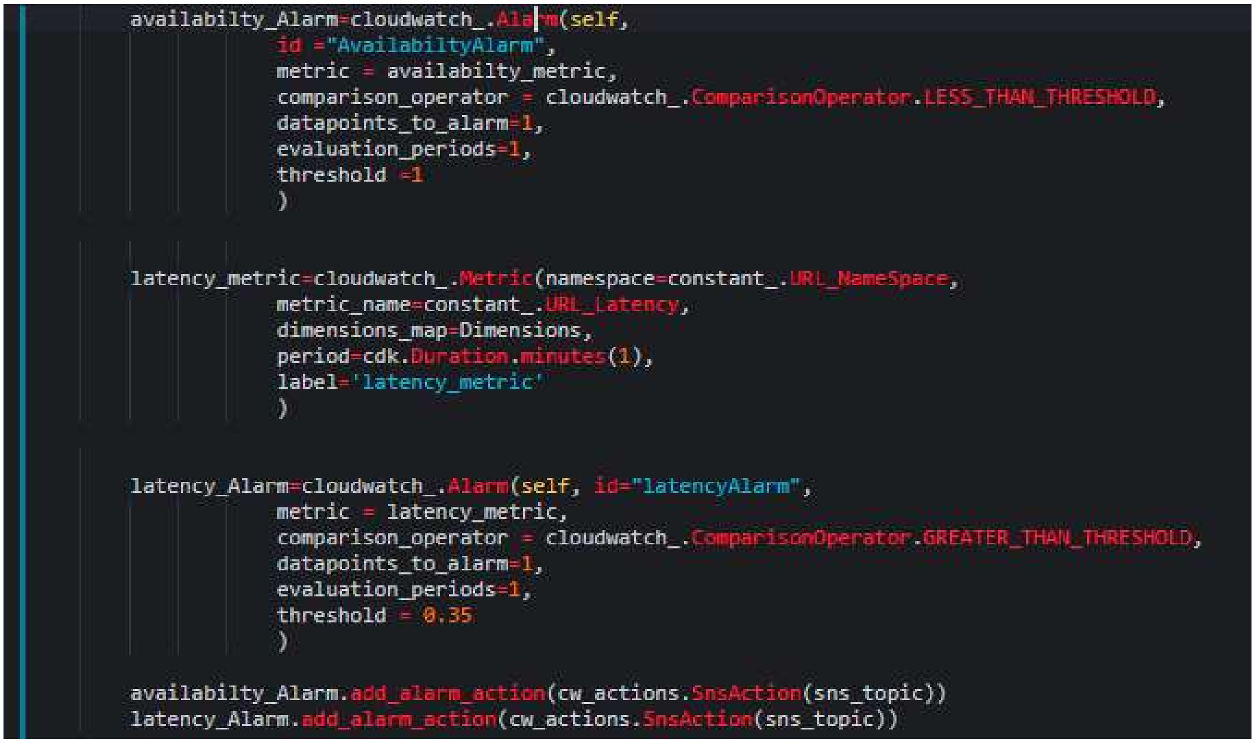


```
Test Event Name
test

Response
{
  "availibility": 1,
  "latency": 0.302906
}

Function Logs
START RequestId: a0ce2b6b-5fbe-4d6c-8b59-73f1889a1dc6 Version: $LATEST
END RequestId: a0ce2b6b-5fbe-4d6c-8b59-73f1889a1dc6
REPORT RequestId: a0ce2b6b-5fbe-4d6c-8b59-73f1889a1dc6  Duration: 823.00 ms Billed Durati

Request ID
```

Figure 6: Latency and availability test result

Alarm generate when threshold breached and send sns to subscribers

Then we set a threshold on metrics and generate an alarm when the threshold is breached. The alarm will notify the subscriber about threshold breached through email notification.



```
        availabilty_Alarm=cloudwatch_.Alarm(self,
                    id ="AvailabiltyAlarm",
                    metric = availabilty_metric,
                    comparison_operator = cloudwatch_.ComparisonOperator.LESS_THAN_THRESHOLD,
                    datapoints_to_alarm=1,
                    evaluation_periods=1,
                    threshold =1
                    )


        latency_metric=cloudwatch_.Metric(namespace=constant_.URL_NameSpace,
                    metric_name=constant_.URL_Latency,
                    dimensions_map=Dimensions,
                    period=cdk.Duration.minutes(1),
                    label='latency_metric'
                    )


        latency_Alarm=cloudwatch_.Alarm(self, id="latencyAlarm",
                    metric = latency_metric,
                    comparison_operator = cloudwatch_.ComparisonOperator.GREATER_THAN_THRESHOLD,
                    datapoints_to_alarm=1,
                    evaluation_periods=1,
                    threshold = 0.35
                    )

        availabilty_Alarm.add_alarm_action(cw_actions.SnsAction(sns_topic))
        latency_Alarm.add_alarm_action(cw_actions.SnsAction(sns_topic))
```

Figure 7: Alarm for latency and availability on cloud watch

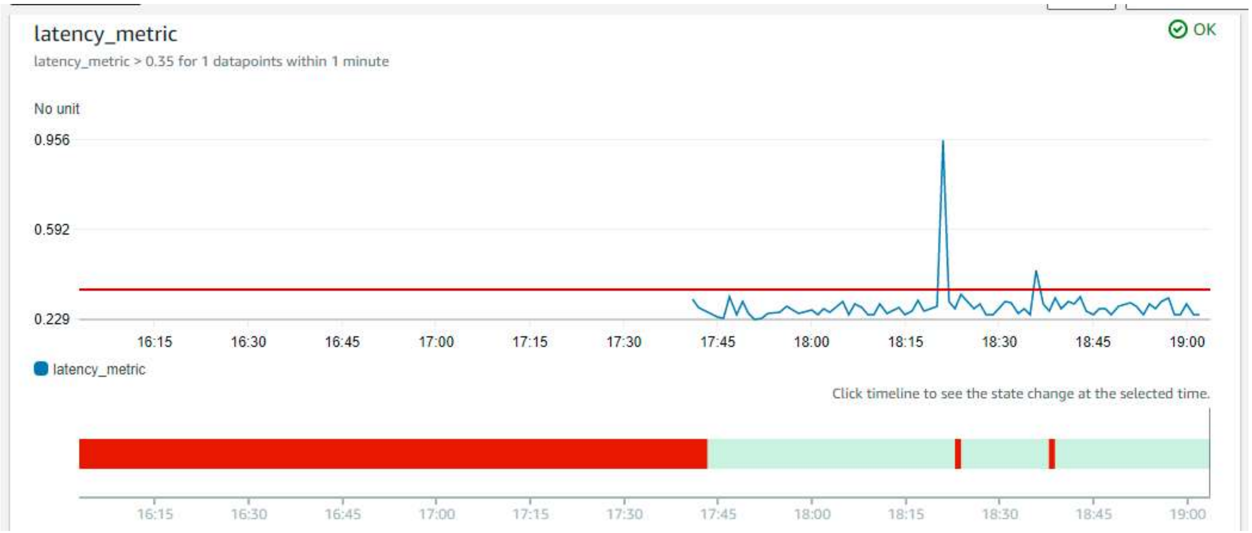
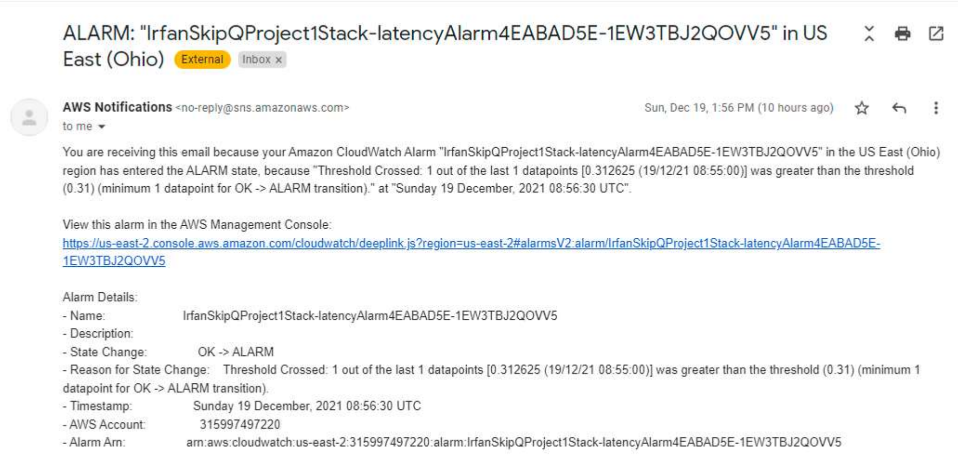


Figure 8: Alarm triggered when threshold breached


ALARM: "IrfanSkipQProject1Stack-latencyAlarm4EABAD5E-1EW3TBJ2QOVV5" in US East (Ohio)

AWS Notifications <no-reply@sns.amazonaws.com> Sun, Dec 19, 1:56 PM (10 hours ago)
to me

You are receiving this email because your Amazon CloudWatch Alarm "IrfanSkipQProject1Stack-latencyAlarm4EABAD5E-1EW3TBJ2QOVV5" in the US East (Ohio) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [0.312625 (19/12/21 08:55:00)] was greater than the threshold (0.31) (minimum 1 datapoint for OK -> ALARM transition)." at "Sunday 19 December, 2021 08:56:30 UTC".

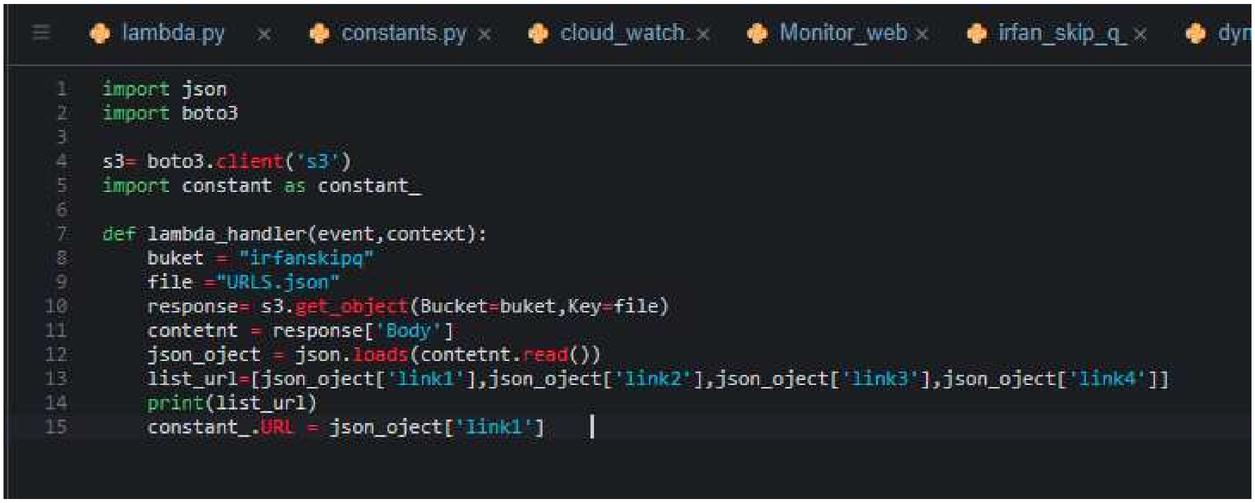
View this alarm in the AWS Management Console:
https://us-east-2.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-2#alarmsV2:alarm/IrfanSkipQProject1Stack-latencyAlarm4EABAD5E-1EW3TBJ2QOVV5

Alarm Details:
- Name: IrfanSkipQProject1Stack-latencyAlarm4EABAD5E-1EW3TBJ2QOVV5
- Description:
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [0.312625 (19/12/21 08:55:00)] was greater than the threshold (0.31) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Sunday 19 December, 2021 08:56:30 UTC
- AWS Account: 315997497220
- Alarm Arn: arn:aws:cloudwatch:us-east-2:315997497220:alarm:IrfanSkipQProject1Stack-latencyAlarm4EABAD5E-1EW3TBJ2QOVV5

Figure 9: Email notification to subscriber

Read json file from S3 bucket

Our project is to monitor web health of custom provided webpages. We have data. Json file having URL of each webpage and this file is stored in AWS S3 bucket. We write another lambda function to read URLs from json file and store these URLs in array.



```python
import json
import boto3

s3= boto3.client('s3')
import constant as constant_

def lambda_handler(event,context):
    buket = "irfanskipq"
    file ="URLS.json"
    response= s3.get_object(Bucket=buket,Key=file)
    contetnt = response['Body']
    json_oject = json.loads(contetnt.read())
    list_url=[json_oject['link1'],json_oject['link2'],json_oject['link3'],json_oject['link4']]
    print(list_url)
    constant_.URL = json_oject['link1']
```

Figure 10: S3 bucket reading

Error and solution:

Here are some common errors I faced and their solution as well.

- Unknown variable
  to solve this issue check spelling and if it is an issue when importing some function then install using the command “pip install –m zyz==1.135.0”
- Syntax Error
  Check the syntax from the API reference for the function.
- Insufficient data:
  Check dimension parameter and duration time. Also check threshold is right or not.

References:

- API Reference — AWS Cloud Development Kit 1.134.0 documentation (amazon.com)
- AWS S3 Tutorial For Beginners | AWS S3 Bucket Tutorial | AWS Training | Edureka - YouTube
- AWS DynamoDB Tutorial | AWS Services | AWS Tutorial For Beginners | AWS Training Video | Simplilearn - YouTube
- Insufficient data: CloudWatch alarm based on custom metric filter | by Marta Tatiana | Medium
- comm command in Linux with examples - GeeksforGeeks