Preliminary Documentation for Sprint1

Web-Health Monitoring

SkipQ Cohot-2 Sprint1

By: Talha Naeem

In sprint1, training started from very scratch including learning concepts of cloud computing, DevOps, and AWS. In this sprint, we started with creation and deployment of Hello World lambda and then moving towards wen health lambda function. Cloudwatch services were utilized to publish web health metrics and to raise an alarm beyond a specific threshold. The alarms were sent to Dynamodb lambda function using SNS subscription. Finally, the alarms were written into dynamodb table.

Table of Contents

Sprint1 Task1

Creating Hello World lambda

After successful setup of the AWS cloud environment, we employed AWS Cloud Development Kit (CDK). In CDK, we created our very first Helloworld Lambda Function, using AWS Lambda. We defined a lambda function in stack file, that calls its handler with the input parameters; events and context from hello_world_lambda file.

Code for Hello World Lambda

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

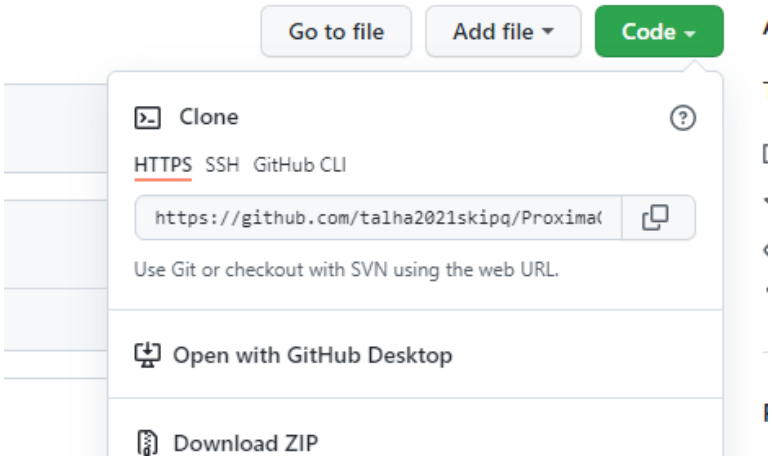
Related Issues

I had once installed the required modules altogether by `python -m pip install –r requirements.txt` but even then I had got the issue that aws_cdk is not found. Then I installed this package using the command `python –m pip install aws-cdk.aws-s3 aws-cdk.aws-lambda` and this worked for me. And I got success in synthesizing my project by `cdk synth` and then deploying it using `cdk deploy`.

Setting up Github

I started from scratch with githhub, learnt from creating repository of a project to cloning to github. I followed following steps:

- Create a new project repository at github
- Copy the https link from clone tab as shown below:



- Put the copied link and clone it with your local virtual machine using the clone command.

  git clone https://github.com/talha2021skipq/ProximaCentauri.git

- Now we can add or update any file into the github repository using commit command.

User Stories in Projects at Github

We employ agile framework for project accomplishment. For that we have to create simple Kanban project at github and then add user stories of our sprints. Each sprint can be divided into tasks for getting clarity of project sections. Moreover, it is helpful to keep a proper record of To-do tasks, if we have multiple project running at the same time. So having said all this, I have created two user stories for my project1 as: Sprint1 Task1 (marked as done), and Sprint1 Task2 (marked as in progress). Each user story must have following parts:

1- Who is responsible?
2- What do they want?
3- What is the expected outcome?

Sprint1 Task2

Measure Web-Health Metrics

We created lambda unction for measuring the web health metrics. The web health lambda will get the availability and latency using urllib3 package. And then those metrics will be sent to a boto3 client for cloudwatch to publish them.

Periodic Invocation of Web-Health lambda

In the next milestone, we have to invoke our web health lambda periodically. So this can be done using Metric method from cloudwatch. Using aws_cloudwatch from CDK, we employed Metric function to put the data points to the cloud after a specified duration. An example on one metric is shown below. The same can be done for all the involved metrics. And then these metrics are sent to cloudwatch putMetric class from boto3 using a client method.

```
latency_metric=cloudwatch_.Metric(namespace= constants.URL_MONITOR_NAMESPACE,
        metric_name=constants.URL_MONITOR_NAME1L,
        dimensions_map=dimension, period=cdk.Duration.minutes(1),label='Latency_metric')
```

So the resultant periodic data graphs for latency and availability are shown here

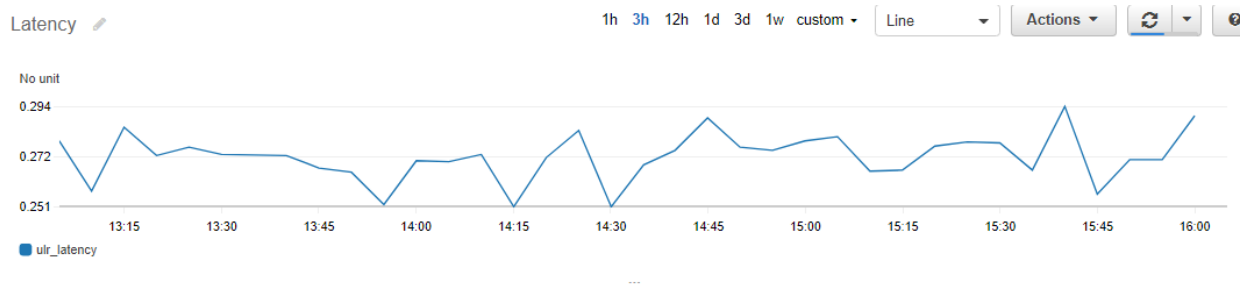


Figure 1 Latency graph on real time data

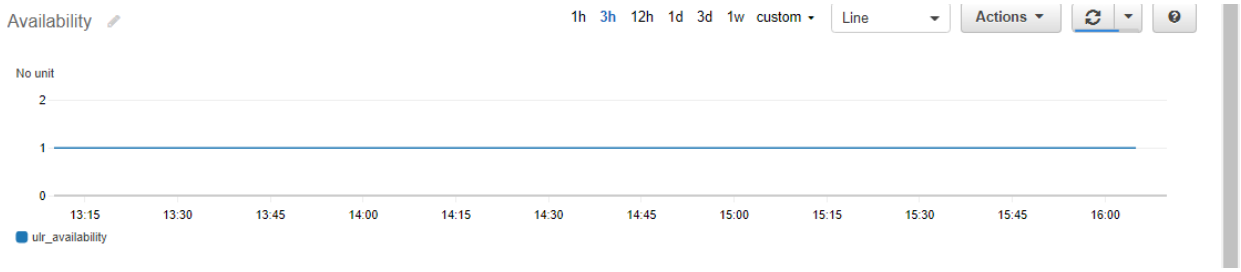


Figure 2 Availability graph on real time data

Sprint1 Task3

Put Alarms on Metrics

Using Alarm metric from aws-cloudwatch, we can set up alarms on the metrics by telling thresholds along with some other parameter.

```
availability_alarm= cloudwatch_.Alarm(self,
        id='Availability_alarm', metric=availability_metric,
        comparison_operator= cloudwatch_.ComparisonOperator.LESS_THAN_THRESHOLD,
        datapoints_to_alarm=1,
        evaluation_periods=1,
        threshold= 1#constants.THRESHOLD_AVAIL
        )
```

Here in cloudwatch, we can see the alarms, as shown below:

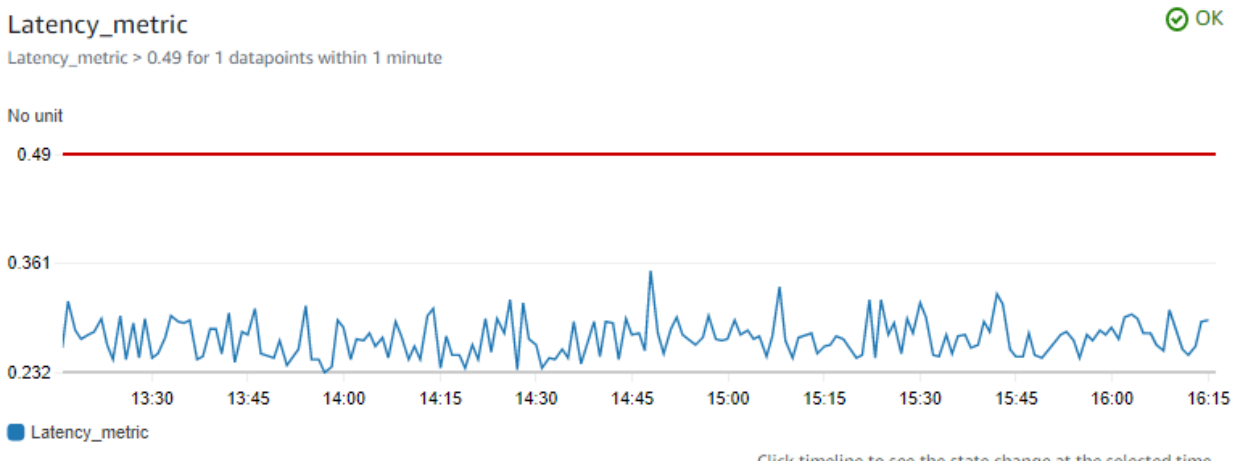


Figure 3 Latency alarm at 0.49

Configure SNS and SNS Subscribers Service(Email)

Using aws-sns we can add a topic of our subscription and then call add_subscription method to set email as subscriber.

```
    topic = sns.Topic(self, "TalhaSkipQWebHealthTopic")
    topic.add_subscription( subscriptions_.EmailSubscription('talha.naeem.s@skipq.org'))
```

Once the subscription topics are created, we have to perform actions on that. We can inform the subscriber (email) or our lambda. We perform this task by aws-cloudwatch-actions using snsaction method.

```
availability_alarm.add_alarm_action(actions_.SnsAction(topic))
latency_alarm.add_alarm_action(actions_.SnsAction(topic))
```

Creating user Stories on github Project

I followed agile framework to take on the project by defining some user stories, tasks, and spike for the project that is going on.

Video1: https://youtu.be/m8ZxTHSKSKE

Video2: https://youtu.be/-MBEnpAgmug

Sprint1 Task4

Configure SNS and SNS Subscribers Service (Dynamodb Lambda)

Using aws-sns we can add a topic of our subscription and then call add_subscription method to set lambda as subscriber.

```
topic = sns.Topic(self, "TalhaSkipQWebHealthTopic")
topic.add_subscription(subscriptions_.LambdaSubscription(fn=Talha_db_lambda))
```

Once the subscription topics are created, we have to perform actions on that. We can inform the subscriber i.e. our lambda. We perform this task by aws-cloudwatch-actions using snsaction method.

Create dynamoDb Table

Using aws_dynamodb, I created a table and privileged read and write right to my dynamodb lambda function. When a table is created in the dynamodb, it gives an error while we re-create the table (with same name). So, I used exception handling while creating my dynamodb table.

```
try:
    dynamo_table= self.create_table()
except: pass
#give read write permissions to our lambda
dynamo_table.grant_read_write_data(Talha_db_lambda)
```

DynamoDb Lambda Invocation:

I have to create a lambda function to write the event (alarm) on dynamodb table. The dynamodb lambda function gets invoked when an alarm is raised. Hence, I have added the sns topic as event to my dynamodb lambda. Other than this, a simpler approach is just to subscribe the lambda function subscription to that topic and it gets all done.

```
topic.add_subscription(subscriptions_.LambdaSubscription(fn=Talha_db_lambda))
```

DynamoDb lambda function

I have created a lambda function for dynamoDB in my stack. And this lambda invokes on Web health SNS topic. And this lambda gets alarm payload in events. Where we can process the payload and put it in the dynamoDB table.

Issued related to dynamoDb

1- My alarms were not being written into my table. And the reason was that, I created an item in my table manually. As solution, I deleted all items from my table and then deployed my code and it started writing to table.

2- I was heading over to creating an SNS event for my lambda function, and I spent much time on that. I must add the wrong path to avoid here in the following:

```
# #  Talha_db_lambda.add_event_source(lambda_events_.SnsEventSource(topic))
#filter_policy={},
#dead_letter_queue=dead_letter_queue))
#dblambda_target= targets_.LambdaFunction(handler=Talha_db_lambda)
#defining rule for lambda function invokation event
#rule=events_.Rule(self, "db_Invokation",
#   description="Db writerLambda",enabled=True,
#  schedule= lambda_schedule,
# targets=[dblambda_target])
```

X

Write Alarms to Dynamodb Table:

From the obtained alarm payload, I extracted StatchangeTime and Alarm ID. And then I wrote these attributes in a dictionary, and I passed that dictionary to put_item() method of table.

```
values = {}
        values['id'] = message
        values['createdDate'] = createdDate
        #values['Reason'] = reason
        table.put_item(Item = values)
```

The alarms will be written into the dynamodb table as follows:

Items returned (8)

id	createdDate
TalhaProjec…	2021-12-19T08:44:04.279+0000
TalhaProjec…	2021-12-19T08:52:04.276+0000
TalhaProjec…	2021-12-19T08:55:04.278+0000
TalhaProjec…	2021-12-19T09:02:04.332+0000
TalhaProjec…	2021-12-19T09:04:04.273+0000
TalhaProjec…	2021-12-19T09:09:07.247+0000

Figure 4 Alarms table in dynamodb

Sprint1 Task5

Creating S3 bucket:

- Created a bucket by using aws-s3
- Have used Bucket method. Name of bucket is talha_first_bucket

```
bucket_talha= s3_.Bucket(self, "talha_first_bucket")
```

- See from S3 console, the bucket will be there as shown below:

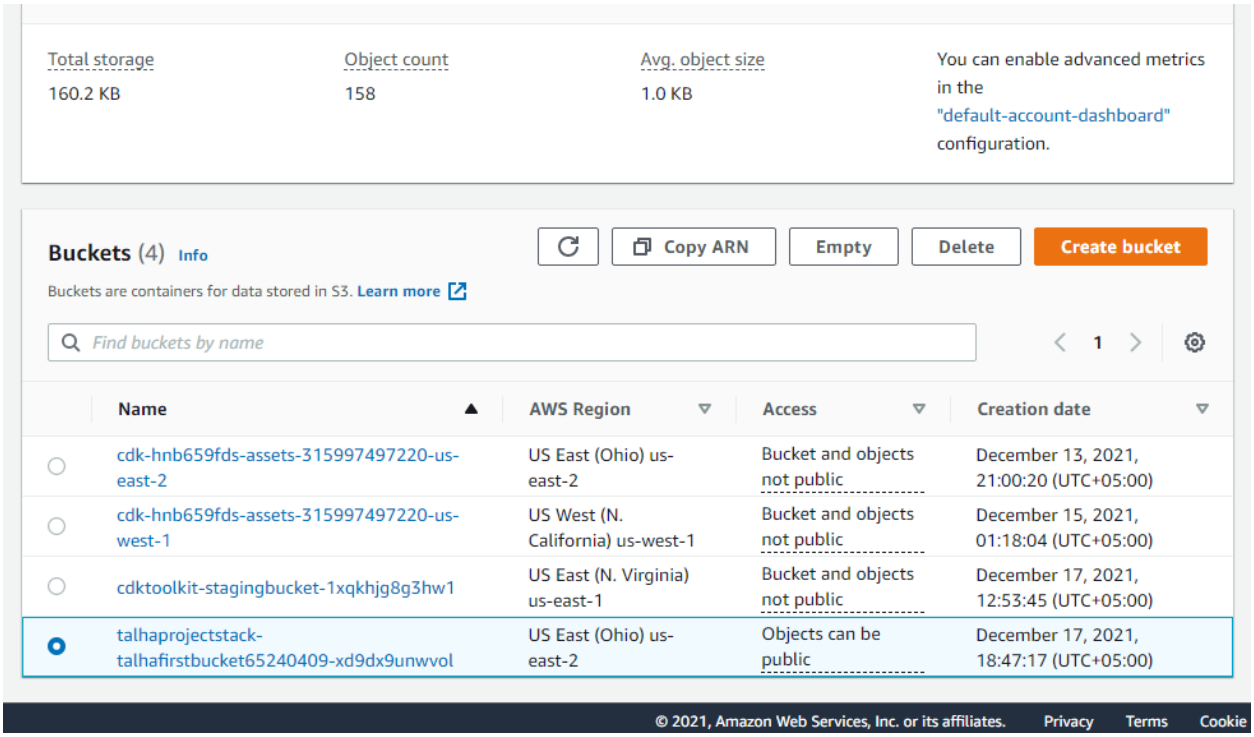


Figure 5 S3 bucket

- Now open the bucket and add file into it. I uploaded a json file having list of URLs.

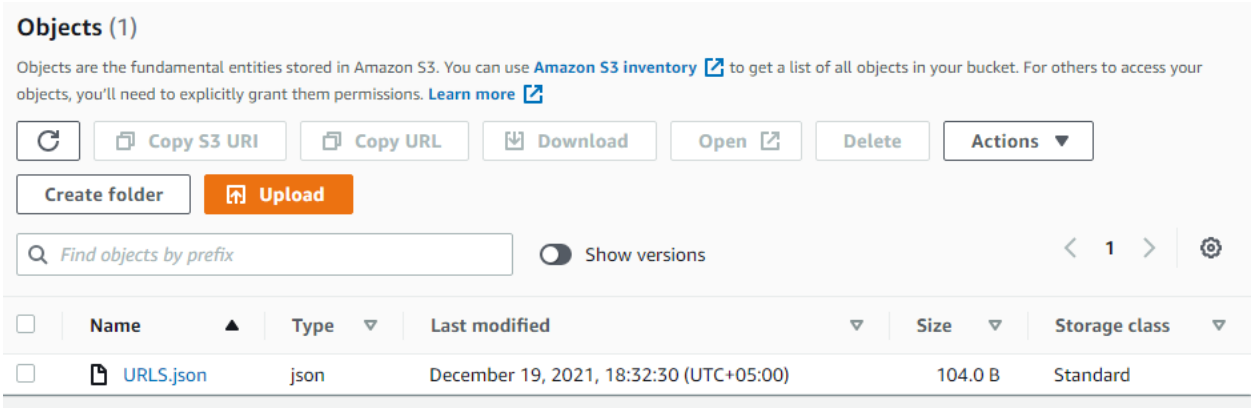


Figure 6 File uploaded to the bucket

Read Contents from Bucket

Now I had to find a way to read contents from S3 bucket. So I started exploring S3 related modules and a brief story is as following:

Amazon S3 inventory (*miss-interpreted)

I jumped to s3 inventory from my s3 bucket at my s3 console. But after exploring its work I came to know that S3 inventory is used to manage the storage, and for auditing and updating on the changes or replications.

Boto3 (*)

I found another way to read from my bucket using boto3, and yes the method exits to get data from s3 bucket. But as I created my bucket in infra-stack and I do not want to use boto3 from SDK in my stack.

Final Decision and Possible Solution

I can create a boto3 client for S3 and with the help of that I can read the contents from S3 bucket. But as I mentioned earlier, that I don't want to use boto3 in my stack. So I implemented this logic in a separate file just for the time being. And I plan to create another lambda function for this task. Yet, I am not sure how to link these two things: URLs list and one-by-one invocation on each URL. This is a question mark so far. However, we can read the contents of the residing file in S3 bucket using the following line of code:

```
import json
import boto3
def read_url_list():
    s3= boto3.client('s3')
    bucket_talha= "talhabucketnew"
    file_name ="URLS.json"
    response= s3.get_object(Bucket=bucket_talha ,Key=file_name)
    cont = response['Body']
    json_oject = json.loads(cont.read())
    list_url=[json_oject['w1'],json_oject['w2'],json_oject['w3'],json_oject['w4']]
    return list_url
```

Cloud Formation Diagram

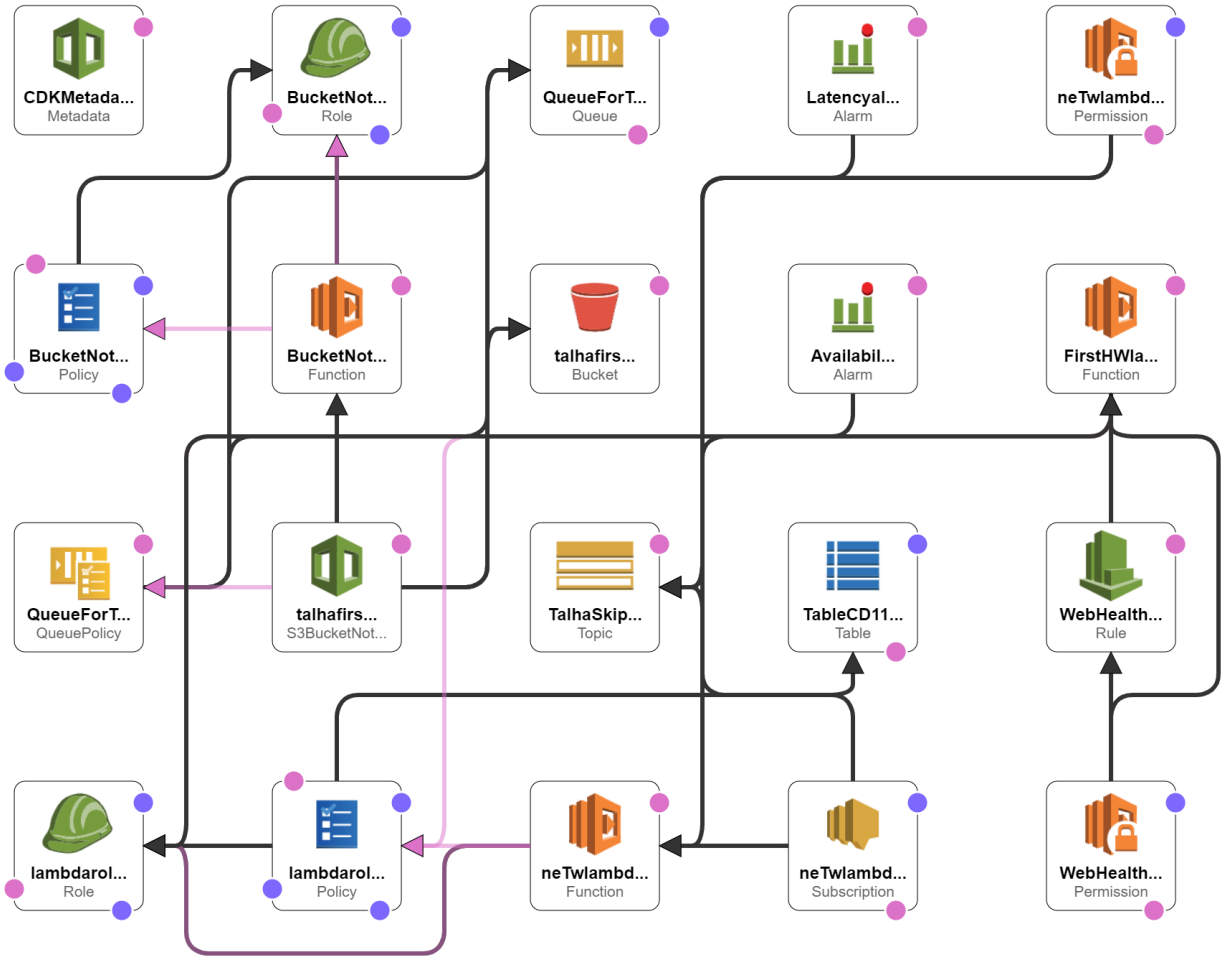


Figure 7 Cloud formation diagram

In Progress Task

I have to run my code on 4 URLs based on the contents read from S3 bucket.