12/19/2021

Project Documentation

Proxima Centauri- SkipQ

Waheed Ahmad

Table of Contents

1. Introduction

1.1. AWS:

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform, offering over 200 fully featured services from data centers globally. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—are using AWS to lower costs, become more agile, and innovate faster.

1.2. Cloud computing

Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

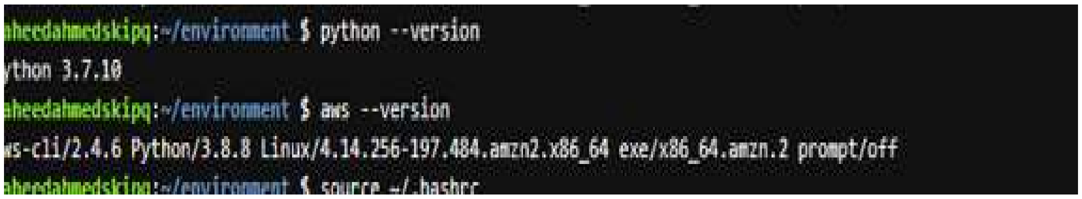
Sprint1

1. Day_1

1.1. Setting up environment:

The environment in AWS console was created in cloud9

The specifications of environment were RAM=1 GB and CPU= 1, all the required packages were installed, and necessary updates were made.

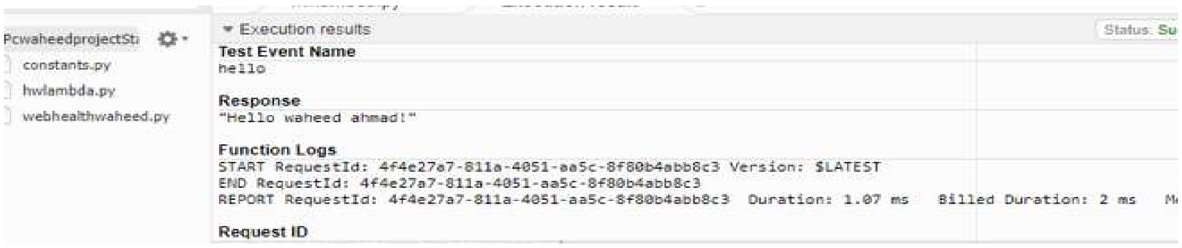


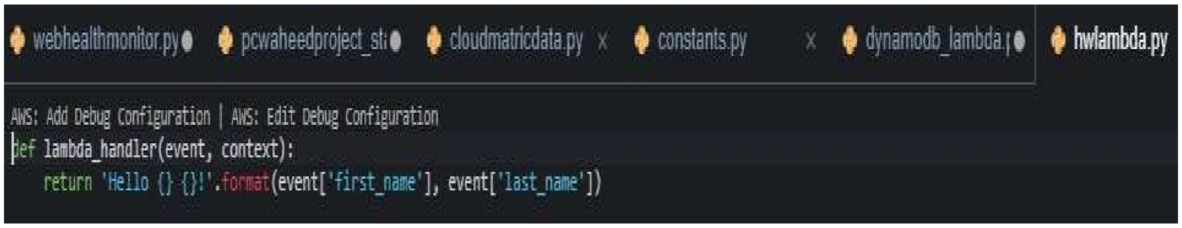
```
aheedahmedskipq:~/environment $ python --version
ython 3.7.10
aheedahmedskipq:~/environment $ aws --version
ws-cli/2.4.6 Python/3.8.8 Linux/4.14.256-197.484.amzn2.x86_64 exe/x86_64.amzn.2 prompt/off
aheedahmedskipq:~/environment $ source ~/.bashrc
```

Issues faced: In updating AWS version and python version, it was solved by changing ***'alias python=python3'*** in the code.

1.2. Hello Lambda!

After setting up the environment the next task was writing a lambda function for hello world , it was simple task the program was tested and proper output

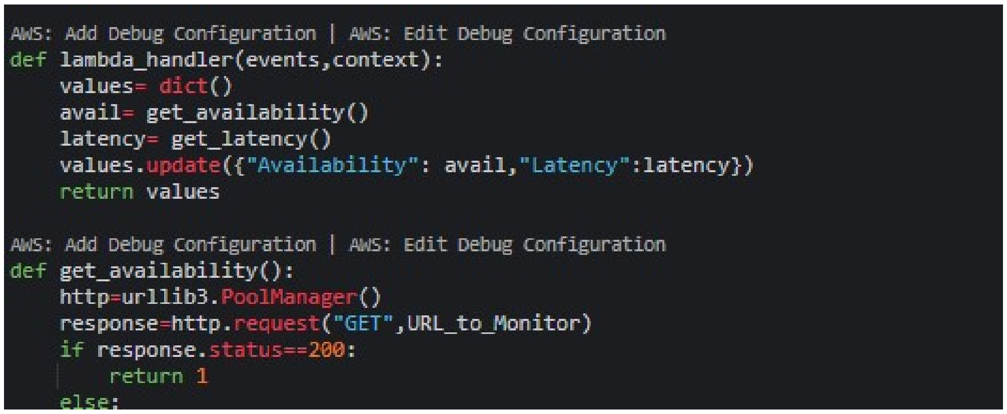

```
Pcwaheedprojectst
  constants.py
  hwlambda.py
  webhealthwaheed.py

▾ Execution results                                              Status: Su
Test Event Name
hello

Response
"Hello waheed ahmad!"

Function Logs
START RequestId: 4f4e27a7-811a-4051-aa5c-8f80b4abb8c3 Version: $LATEST
END RequestId: 4f4e27a7-811a-4051-aa5c-8f80b4abb8c3
REPORT RequestId: 4f4e27a7-811a-4051-aa5c-8f80b4abb8c3  Duration: 1.07 ms   Billed Duration: 2 ms   M

Request ID
```


```
webhealthmonitor.py   pcwaheedproject_st   cloudmatricdata.py   constants.py   dynamodb_lambda.   hwlambda.py

AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def lambda_handler(event, context):
    return 'Hello {} {}!'.format(event['first_name'], event['last_name'])
```

Day_2:

1.3. Webhealth_lambda:

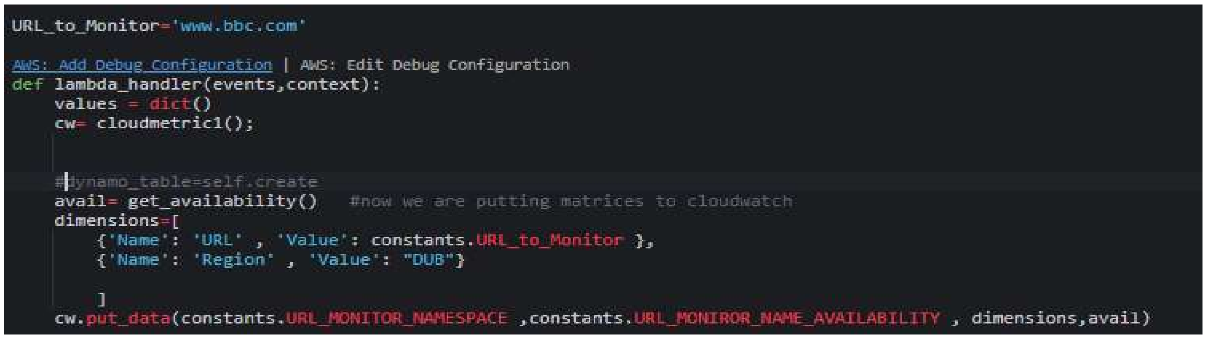
This function is programmed to check whether a website is performing okay or not, in terms of availability and latency ,we defined two functions for availability and latency and deployed it.


```
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def lambda_handler(events,context):
    values= dict()
    avail= get_availability()
    latency= get_latency()
    values.update({"Availability": avail,"Latency":latency})
    return values

AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def get_availability():
    http=urllib3.PoolManager()
    response=http.request("GET",URL_to_Monitor)
    if response.status==200:
        return 1
    else:
```

Issues faced: while writing in stack , there was an issue with lambda handler, which was rectified.

1.4. Webhealth_Monitor/ periodic lambda:

This is an extension of web _health ,and it creates graph of availability and latency metrics for monitoring the status of a website , metrics were created using this lambda function and can be seen in cloudwatch , these latency and availability values can further be used to raise alarms.

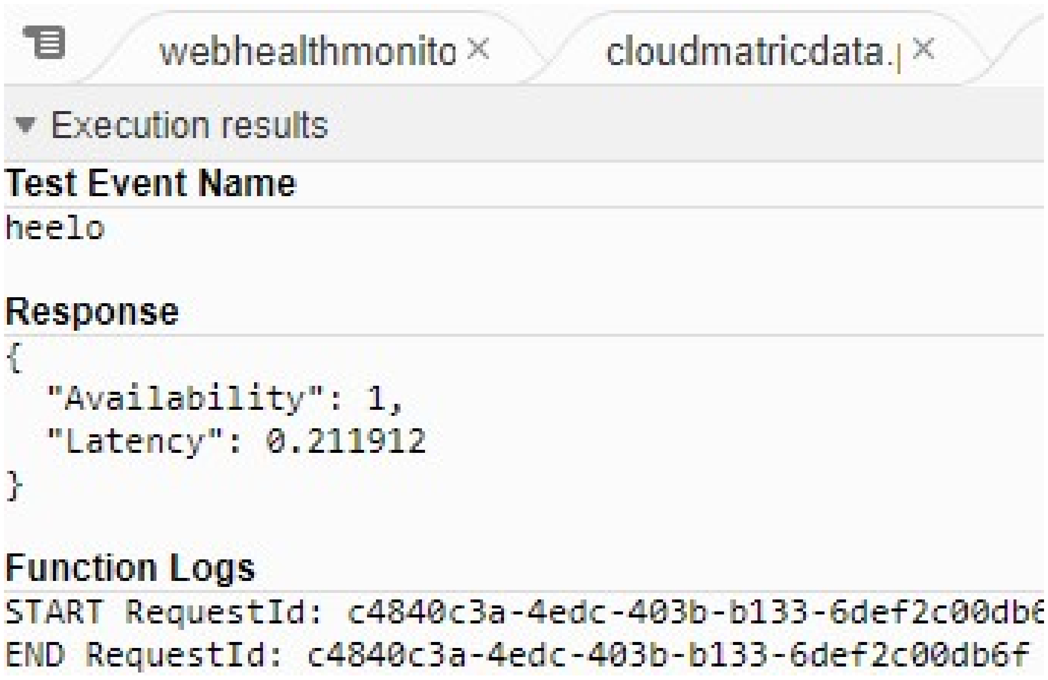


```
URL_to_Monitor='www.bbc.com'

AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def lambda_handler(events,context):
    values = dict()
    cw= cloudmetric1();


    #dynamo_table=self.create
    avail= get_availability()   #now we are putting matrices to cloudwatch
    dimensions=[
        {'Name': 'URL' , 'Value': constants.URL_to_Monitor },
        {'Name': 'Region' , 'Value': "DUB"}

        ]
    cw.put_data(constants.URL_MONITOR_NAMESPACE ,constants.URL_MONIROR_NAME_AVAILABILITY , dimensions,avail)
```

Issues faced: Importing from constants.py file caused an error, which was rectified by changing the names of variables

1.5. Cloudmetric data:

This function is created to load the availability and latency metric graphs .



Test Event Name

```
heelo
```

Response

```
{
  "Availability": 1,
  "Latency": 0.211912
}
```

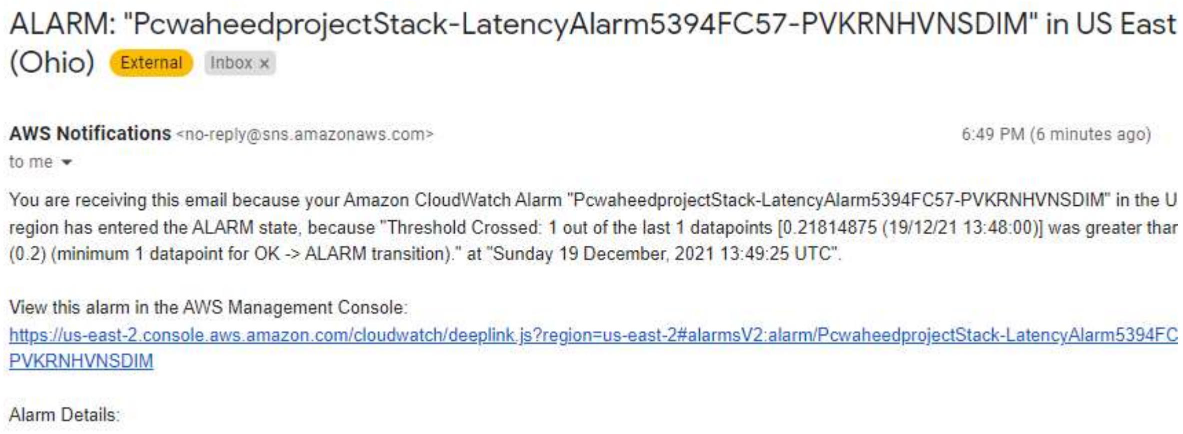
Function Logs

```
START RequestId: c4840c3a-4edc-403b-b133-6def2c00db6
END RequestId: c4840c3a-4edc-403b-b133-6def2c00db6f
```

Day_3:

1.6. Alarms

Now that the metrics for the availability and latency are defined, we can set threshold to them and raise an alarm when a certain threshold is reached



ALARM: "PcwaheedprojectStack-LatencyAlarm5394FC57-PVKRNHVNSDIM" in US East (Ohio) External Inbox ×

AWS Notifications <no-reply@sns.amazonaws.com> 6:49 PM (6 minutes ago)
to me

You are receiving this email because your Amazon CloudWatch Alarm "PcwaheedprojectStack-LatencyAlarm5394FC57-PVKRNHVNSDIM" in the U region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [0.21814875 (19/12/21 13:48:00)] was greater thar (0.2) (minimum 1 datapoint for OK -> ALARM transition)." at "Sunday 19 December, 2021 13:49:25 UTC".

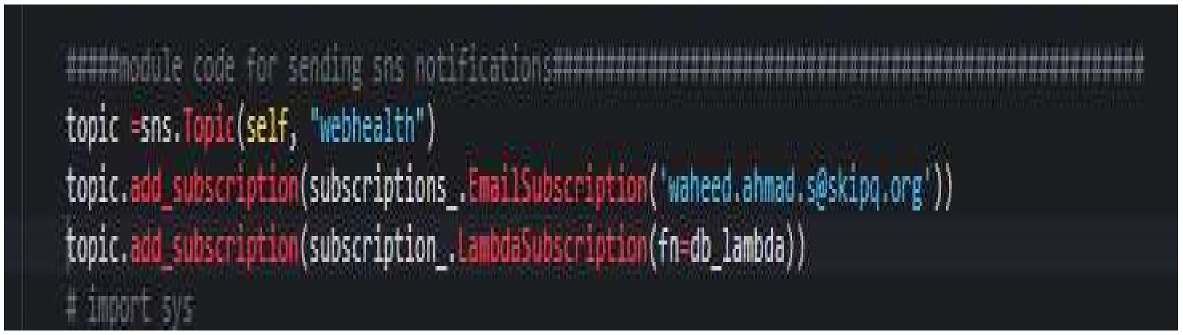
View this alarm in the AWS Management Console:
https://us-east-2.console.aws.amazon.com/cloudwatch/deeplink.js?region=us-east-2#alarmsV2:alarm/PcwaheedprojectStack-LatencyAlarm5394FC PVKRNHVNSDIM

Alarm Details:

1.7. SNS

After setting up alarms to be raised when a certain threshold is reached, we need to add subscriptions to it , to be notified in case of an alarm is breached

Lambdafunction subscription:

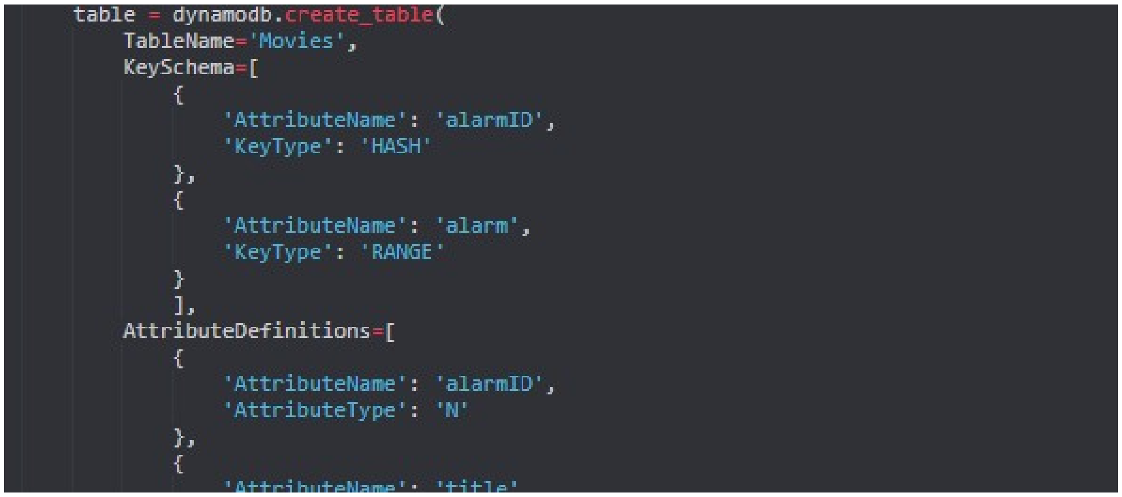


```
#####module code for sending sns notifications########################################################
topic =sns.Topic(self, "webhealth")
topic.add_subscription(subscriptions_.EmailSubscription('waheed.ahmad.s@skipq.org'))
topic.add_subscription(subscription_.LambdaSubscription(fn=db_lambda))
# import sys
```

Day_4:

1.8. Working with DynamoDB

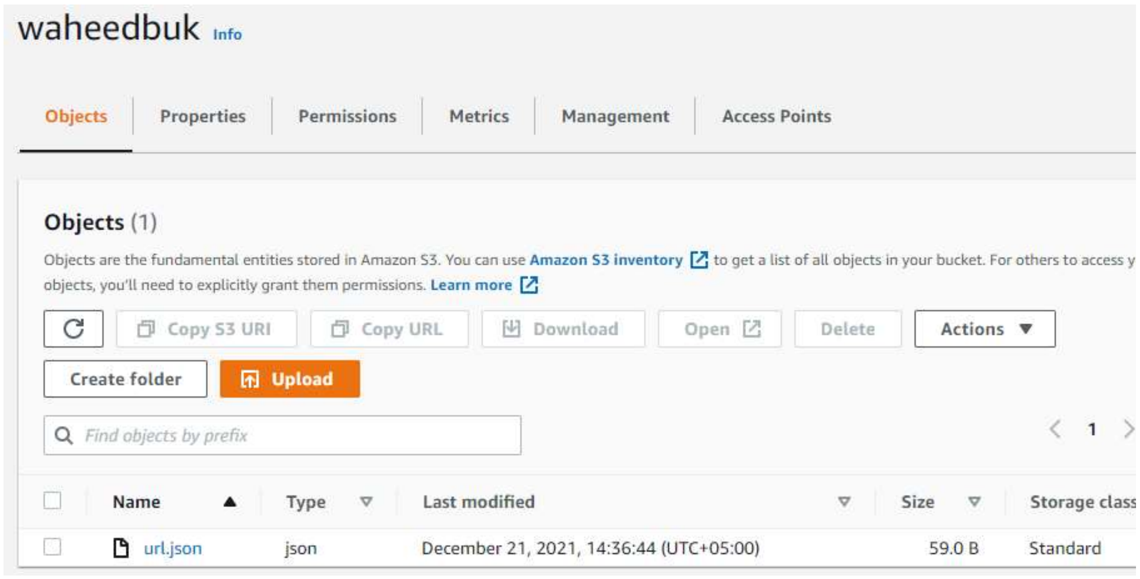
The DynamoDB table is created to store the alarm values ,



```
    table = dynamodb.create_table(
        TableName='Movies',
        KeySchema=[
            {
                'AttributeName': 'alarmID',
                'KeyType': 'HASH'
            },
            {
                'AttributeName': 'alarm',
                'KeyType': 'RANGE'
            }
            ],
        AttributeDefinitions=[
            {
                'AttributeName': 'alarmID',
                'AttributeType': 'N'
            },
            {
                'AttributeName': 'title'
```

1.9. S3 bucket :

S3 bucket was created to read the URLs from the file



Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access y objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
url.json	json	December 21, 2021, 14:36:44 (UTC+05:00)	59.0 B	Standard

Sprint_2

2. Task 1

2.1. Intro to pipelines:

Code Pipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to deliver features and updates rapidly and reliably.

2.2. CI vs CD:

Continuous integration: is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component and a cultural component (e.g., learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

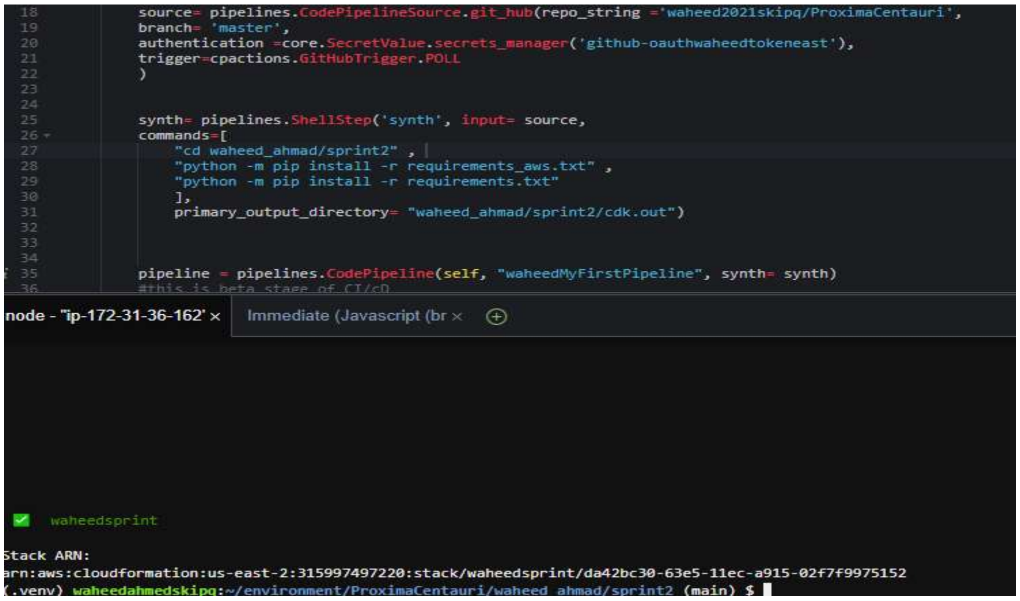
Continuous delivery : is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.
This means that on top of automated testing, you have an automated release process and you can deploy your application any time by clicking a button.

2.3. First pipeline:

Create first pipeline stack for deployment

Create a new stack for pipeline

Define source and authentication for your source (i.e. GitHub)



```
        source= pipelines.CodePipelineSource.git_hub(repo_string ='waheed2021skipq/ProximaCentauri',
        branch= 'master',
        authentication =core.SecretValue.secrets_manager('github-oauthwaheedtokeneast'),
        trigger=cpactions.GitHubTrigger.POLL
        )


        synth= pipelines.ShellStep('synth', input= source,
        commands=[
            "cd waheed_ahmad/sprint2" ,
            "python -m pip install -r requirements_aws.txt" ,
            "python -m pip install -r requirements.txt"
            ],
            primary_output_directory= "waheed_ahmad/sprint2/cdk.out")



        pipeline = pipelines.CodePipeline(self, "waheedMyFirstPipeline", synth= synth)
        #this is beta stage of CI/CD
```

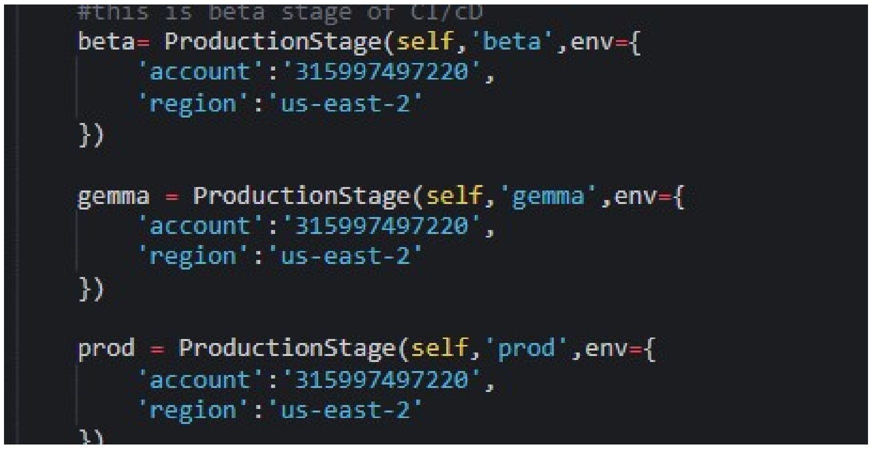
```
 ✅  waheedsprint

Stack ARN:
arn:aws:cloudformation:us-east-2:315997497220:stack/waheedsprint/da42bc30-63e5-11ec-a915-02f7f9975152
(.venv) waheedahmedskipq:~/environment/ProximaCentauri/waheed_ahmad/sprint2 (main) $
```

Create environments (beta etc.)

Add stages for final deployments

3 Task 2

3.1.3 Beta , gamma and Production stages :



```
    #this is beta stage of CI/CD
    beta= ProductionStage(self,'beta',env={
        'account':'315997497220',
        'region':'us-east-2'
    })

    gemma = ProductionStage(self,'gemma',env={
        'account':'315997497220',
        'region':'us-east-2'
    })

    prod = ProductionStage(self,'prod',env={
        'account':'315997497220',
        'region':'us-east-2'
    })
```

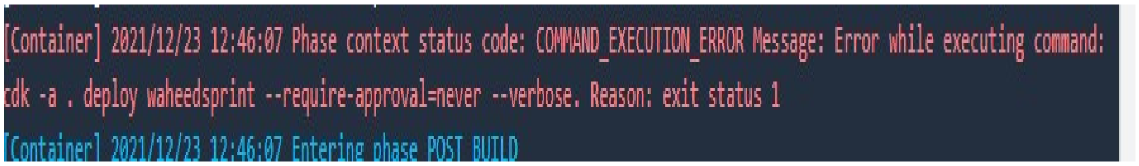
Issues faced: I- aws_cdk module not found

Solution : I had to explicitly install requirements in /bin/pip3.6 module

ii- `cd: can't cd to waheed_ahmad/sprint2`

Solution : changed the structure of directory

iii-



```
[Container] 2021/12/23 12:46:07 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
cdk -a . deploy waheedsprint --require-approval=never --verbose. Reason: exit status 1
[Container] 2021/12/23 12:46:07 Entering phase POST_BUILD
```

This was solved by adding a cdk.out to sprint2 GitHub.

3. Task 3 :

3.1. Add unit test and integration testing:

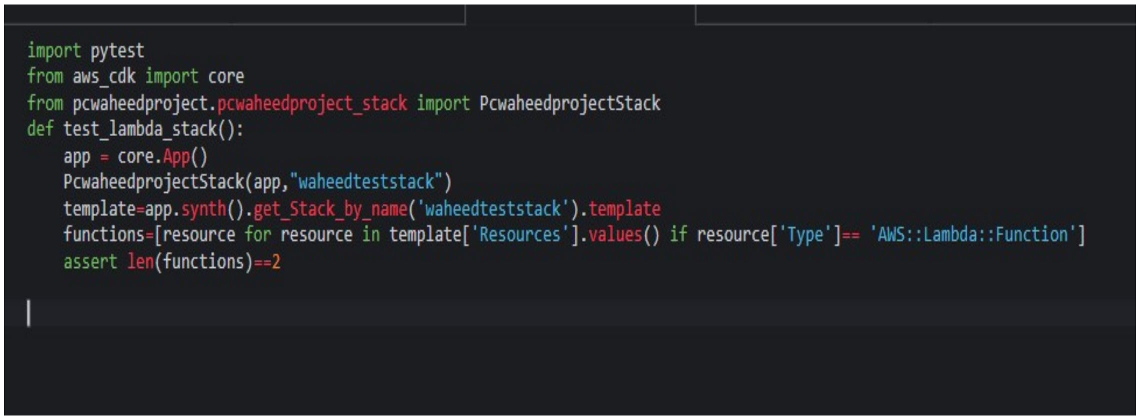
3.1.1. What Is unit testing?

UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

3.1.2. What is integration testing:

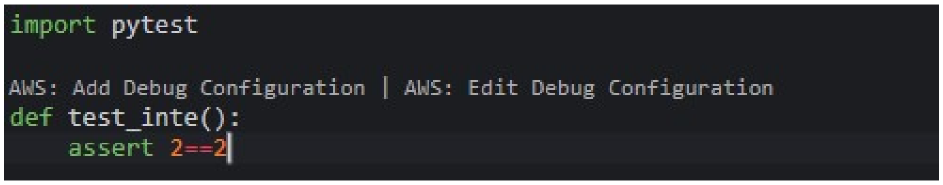
INTEGRATION TESTING is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

A new module named as 'unittest' was added for unit testing :



```
import pytest
from aws_cdk import core
from pcwaheedproject.pcwaheedproject_stack import PcwaheedprojectStack
def test_lambda_stack():
    app = core.App()
    PcwaheedprojectStack(app,"waheedteststack")
    template=app.synth().get_Stack_by_name('waheedteststack').template
    functions=[resource for resource in template['Resources'].values() if resource['Type']== 'AWS::Lambda::Function']
    assert len(functions)==2
```

A new module named as 'integtest' was added for integration testing:

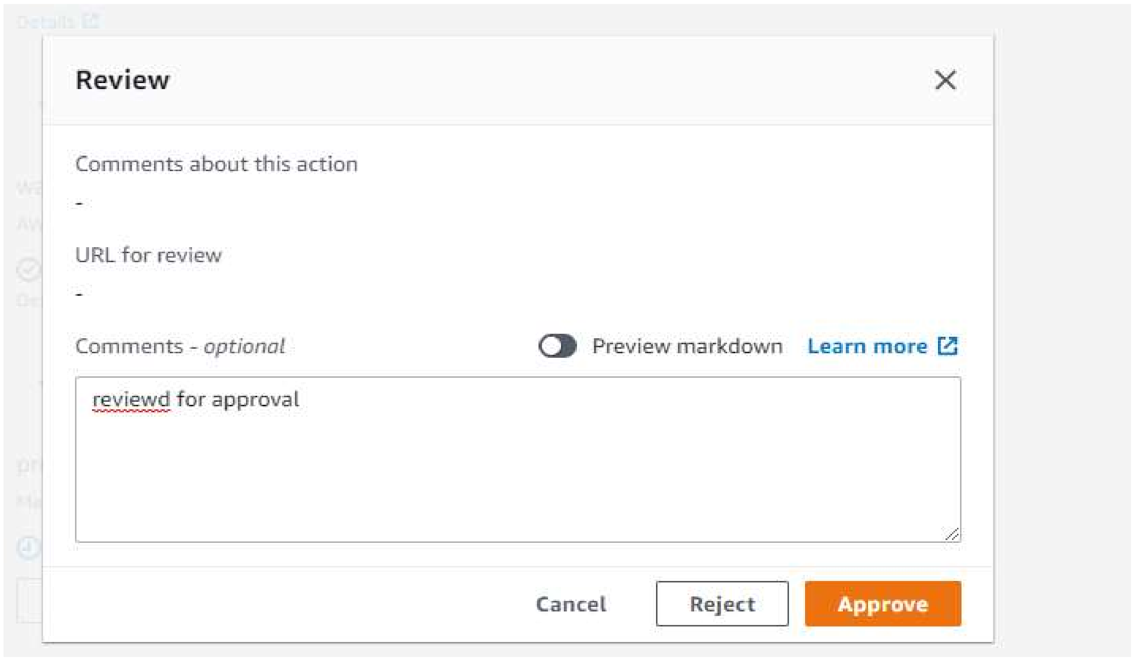


```
import pytest

AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_inte():
    assert 2==2
```

Issues : namespace error, the results were showing 0 tests run, so rectified this error by following a specific nomenclature (test_*.py or *_test.py) for test files and directories.

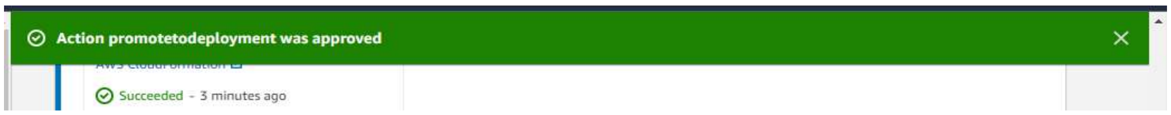
3.1.3 Add manual approval step:

By adding this step a human manual approval is required before proceeding further :



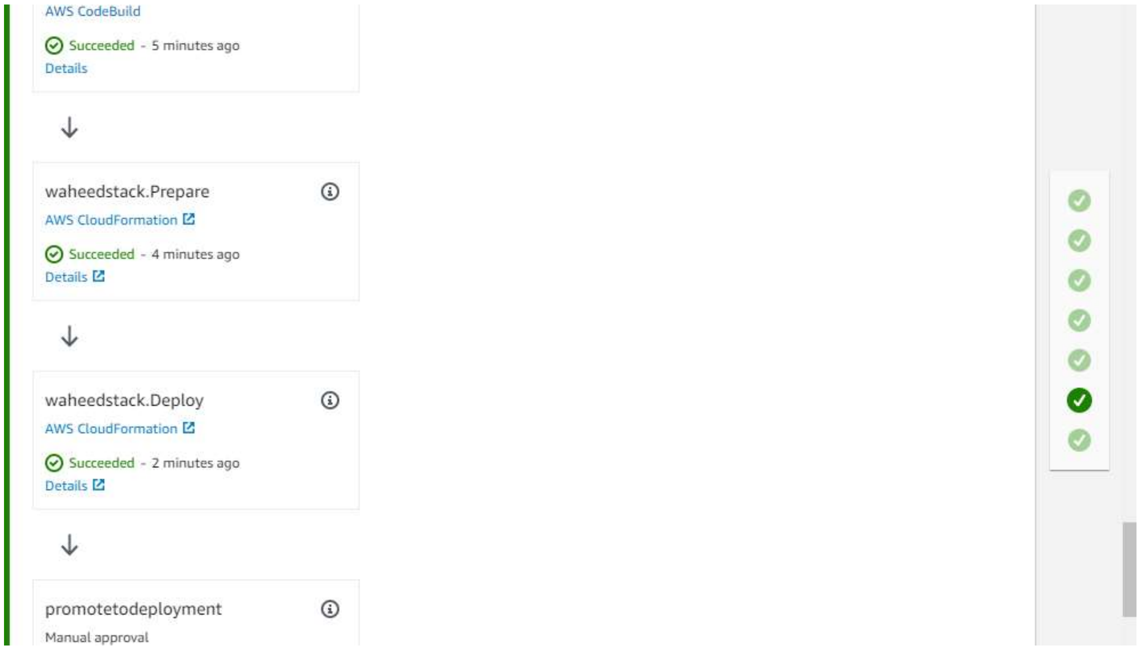
reviewd for approval

Promoted :



Action promotetodeployment was approved

4 Successful implementation:


AWS CodeBuild
Succeeded - 5 minutes ago
Details
waheedstack.Prepare
AWS CloudFormation
Succeeded - 4 minutes ago
Details
waheedstack.Deploy
AWS CloudFormation
Succeeded - 2 minutes ago
Details
promotetodeployment
Manual approval