

Scrum Documentation

Production Grade Programming

Cloud Computing



By: Talha Naeem
DevOps Trainee, SkipQ Cohort-II, ProximaCentauri

Dec 2021 - Jan 2022

In sprint1, training started from scratch including learning the concepts of cloud computing, DevOps, and IaC in AWS. We created a web health monitoring system using Cloudwatch services to publish web health metrics and raise an alarm beyond a specific threshold, and recorded the alarms in Dynamodb table. In sprint2, we created CI/CD pipeline having a self-mutate update configured for source, beta stage with pre-unit test, and prod stage added along with pre-manual approval functionality. Moreover, we employed the AWS auto traffic shift feature for the lambda function, by creating an alias and then setting a threshold on duration metrics of the lambda function. In sprint3, we create an API gateway with CRUD operations, and the data from events at API were recorded in the Dynamodb table. We added some unit tests and in integration tests, we added some real-time tests for our API operations. In sprint4, a frontend was created with reactJS using Chakra UI and deployed this app build in AWS Amplify with Oauth authorization. In sprint5, we created API test client using docker image and pushed the image to AWS ECR, which was deployed in the ECS EC2 service.

Table of Contents

1	Sprint1: Building a Web Crawler using AWS CDK	4
1.1	Overview	4
1.2	Getting Started with AWS Lambda	4
1.3	Periodical Readings of Web Health Metrics	5
1.4	Metrics Alarms on AWS CloudWatch.....	6
1.5	AWS SNS Subscription on AWS Lambda	8
1.6	Reading URLs Data from AWS S3 Bucket	10
1.7	Modification to Crawl for URLs List.....	12
1.8	Cloud Formation Diagram	14
1.9	Related Issues	14
2	Sprint2: Multi-Stage CI/CD Pipeline using AWS CDK	16
2.1	Overview	16
2.2	Setting up Source Stage.....	16
2.3	Build and Update Stages	18
2.4	Beta Stage for Code Deployment.....	19
2.5	Production Stage with Manual Approval	21
2.6	Auto Roll Back on Alarm.....	22
2.7	Merging the Pull Request.....	25
2.8	Related Issues	27
3	Sprint3: API for Web Crawler.....	32
3.1	Overview	32

3.2	Dynamodb Table for URLs.....	32
3.3	Lambda Restful API Creation.....	33
3.4	Handler Functions for API Requests.....	37
3.5	Auto- Creation of Metrics on New URLs	38
3.6	Unit and Integration Testing	39
3.7	Related Issues	40
4	Sprint 4: Frontend Development in ReactJS and Deployment in AWS Amplify	42
4.1	Overview	42
4.2	Getting Started with ReactJS	42
4.3	Design Parameters of Required Frontend	46
4.4	Frontend Design with Chakra UI in ReactJS	46
4.5	React App Deployment Using AWS-Amplify.....	49
4.6	The Hosted Web.....	50
4.7	Related Issues	52
5	Sprint5: API Functional and Security Testing in Docker-Compose	55
5.1	Overview	55
5.2	Docker-Compose in Docker with Ports	55
5.3	Create and Build Docker Image for pyresttest	57
5.4	Push Image to AWS ECR	59
5.5	Pull Image from ECR.....	61
5.6	Deploy the Image on ECS.....	61
5.7	Related Issues	64
6	References	65

1 Sprint1: Building a Web Crawler using AWS CDK

1.1 Overview

In the sprint, we have to build a canary in lambda function that will measure the availability and latency of a custom list of websites. The crawler will run periodically on a 5-minute cadence and write the web health metrics on Cloudwatch. In the cloudwatch dashboard, we will then set some alarms on the metrics and employ some SNS notification services to receive alarm alerts via email.

1.2 Getting Started with AWS Lambda

1.2.1 Creating Hello World lambda

After a 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 the stack file, which calls its handler with the input parameters; events, and context from the hello_world_lambda file.

1.2.1.1 Code for Hello World Lambda

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

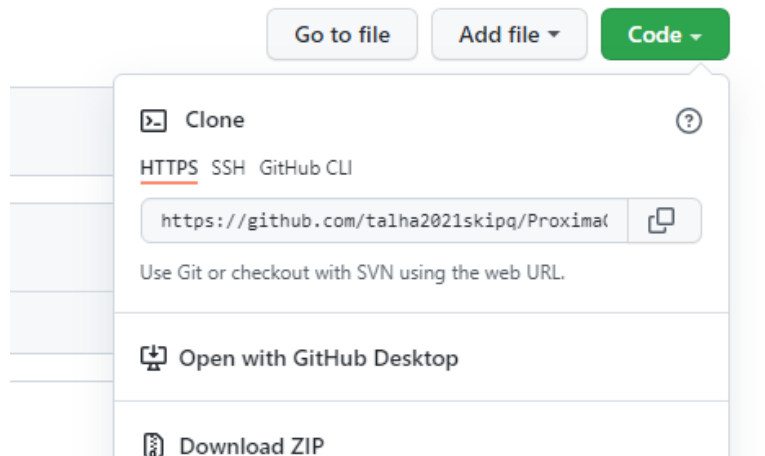
1.2.2 Related Issues

I had once installed the required modules altogether by `python -m pip install -r requirements.txt`, but even then, I 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. In addition, I got success in synthesizing my project by `CDK synth` and then deploying it using `CDK deploy`.

1.2.3 Setting up Github

I started from scratch with Github, learned from creating a repository of a project to cloning to Github. I followed the following steps:

- Create a new project repository at GitHub
- Copy the HTTPS link from the 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 the commit command.

1.2.4 User Stories in Projects at Github

We employ an agile framework for project accomplishment. For that, we have to create a 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 projects running at the same time. So having said all this, I have created two user stories for my project1: Sprint1 Task1 (marked as done), and Sprint1 Task2 (marked as in progress). Each user story must have the following parts:

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

1.3 Periodical Readings of Web Health Metrics

1.3.1 Measure Web-Health Metrics

We created a lambda function for measuring the web health metrics. The web health lambda will get the availability and latency using the urllib3 package. Then those metrics will be sent through a boto3 client for cloud watch to publish them.

1.3.2 Periodic Invocation of Web-Health lambda

We have to invoke our web health lambda periodically in the next milestone. So this can be done using the Metric method from cloud watch. Using `aws_cloudwatch` from CDK, we employed the Metric function to put the data points to the cloud after a specified duration. An example of 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

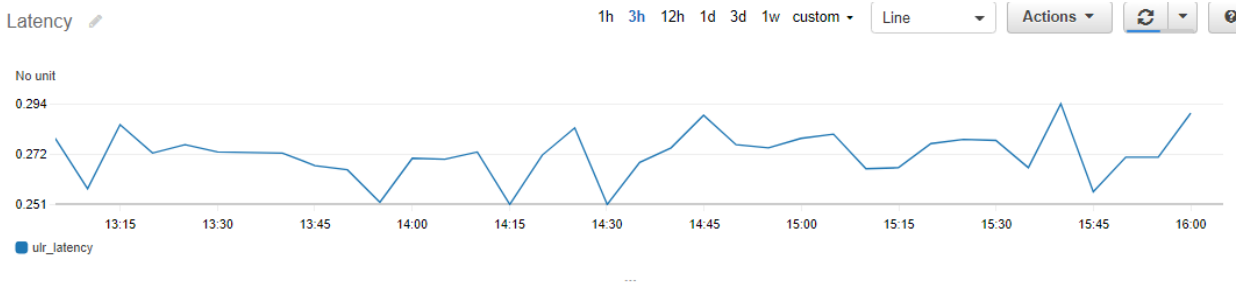


Figure 1 Latency graph on real-time data

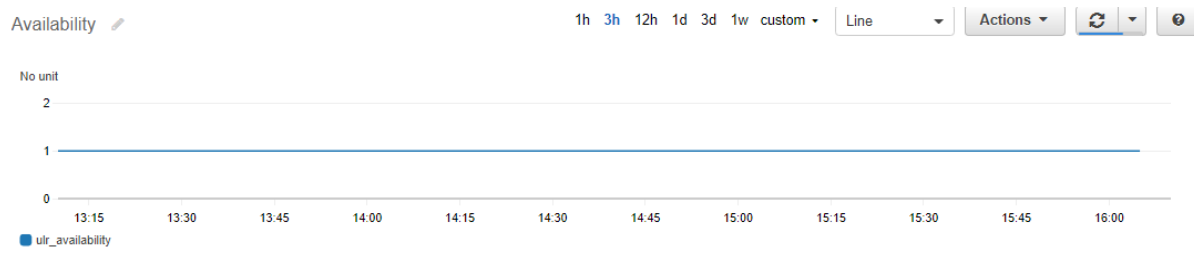


Figure 2 Availability graph on real-time data

1.4 Metrics Alarms on AWS CloudWatch

1.4.1 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 cloud watch, we can see the alarms, as shown below:

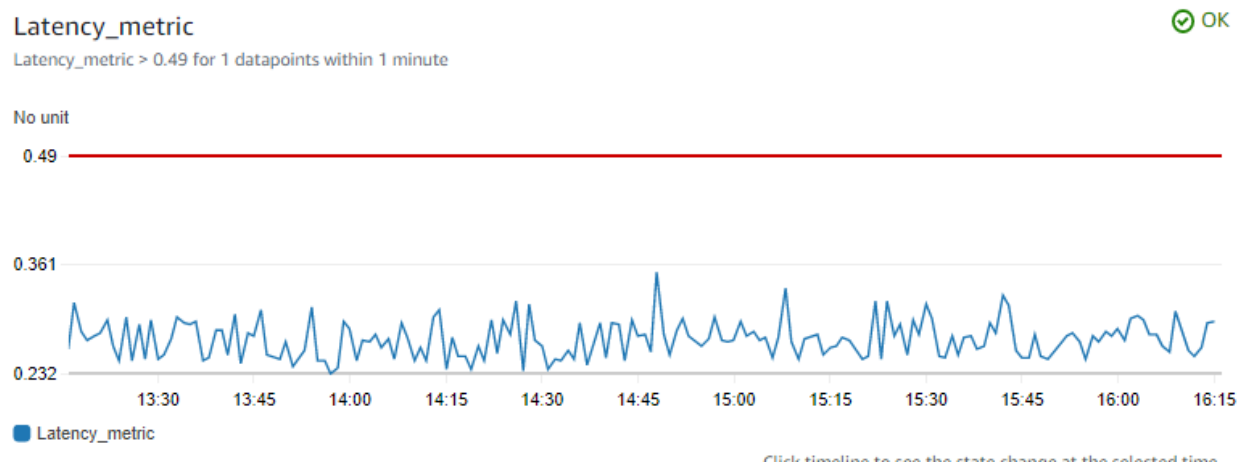


Figure 3 Latency alarm at 0.49

1.4.2 Configure SNS and SNS Subscribers Service (Email)

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

1.4.3 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>

1.5 AWS SNS Subscription on AWS Lambda

1.5.1 Configure SNS and SNS Subscribers Service (Dynamodb Lambda)

Using aws-sns we can add a topic of our subscription and then call the add_subscription method to set lambda as a 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 them. We can inform the subscriber i.e. our lambda. We perform this task by AWS-cloudwatch-actions using the snsaction method.

1.5.2 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 the 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)
```


1.5.3 DynamoDb Lambda Invocation:

I have to create a lambda function to write the event (alarm) on the dynamodb table. The dynamodb lambda function gets invoked when an alarm is raised. Hence, I have added the SNS topic as an 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))
```

1.5.4 DynamoDb lambda function

I have created a lambda function for dynamoDB in my stack. And this lambda invokes 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.

1.5.5 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)		
<input type="checkbox"/>	id ▼	createdDate
<input type="checkbox"/>	TalhaProjec...	2021-12-19T08:44:04.279+0000
<input type="checkbox"/>	TalhaProjec...	2021-12-19T08:52:04.276+0000
<input type="checkbox"/>	TalhaProjec...	2021-12-19T08:55:04.278+0000
<input type="checkbox"/>	TalhaProjec...	2021-12-19T09:02:04.332+0000
<input type="checkbox"/>	TalhaProjec...	2021-12-19T09:04:04.273+0000
<input type="checkbox"/>	TalhaProjec...	2021-12-19T09:09:07.247+0000

Figure 4 Alarms table in dynamodb

1.6 Reading URLs Data from AWS S3 Bucket

1.6.1 Creating S3 bucket:

- Created a bucket by using aws-s3
- Have used the Bucket method. The name of the 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:

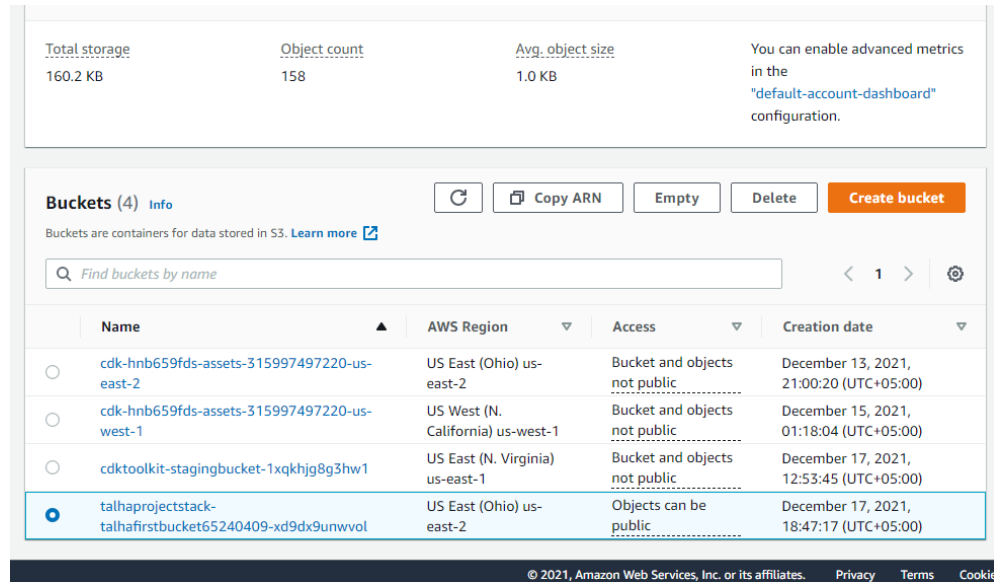


Figure 5 S3 bucket

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

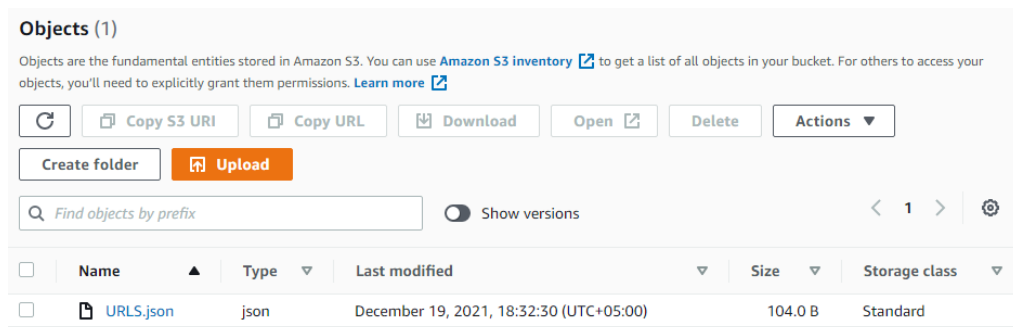


Figure 6 File uploaded to the bucket

1.6.2 Read Contents from Bucket

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

1.6.2.1 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.

1.6.2.2 Boto3 (*)

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

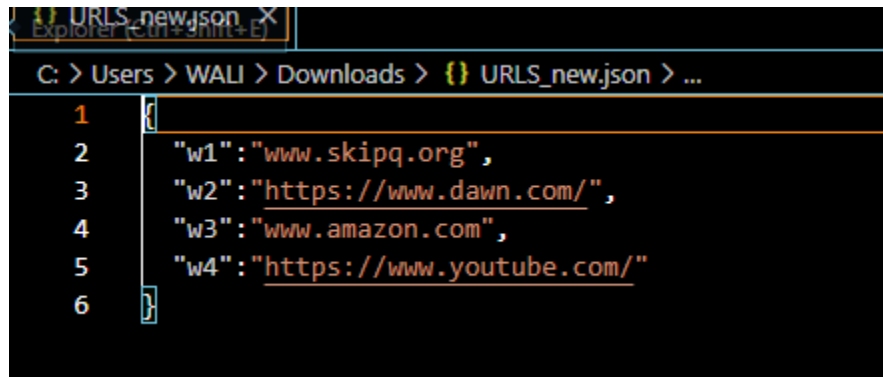
1.6.2.3 Final Decision and Possible Solution

I can create a boto3 client for S3 and with the help of that, I can read the contents from the 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 the 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_object = json.loads(cont.read())
    list_url=[json_object['w1'],json_object['w2'],json_object['w3'],json_object['w4']]
    return list_url
```

1.7 Modification to Crawl for URLs List

I have to run my code on 4 URLs based on the contents read from the S3 bucket. So I read the files located in my S3 bucket. The file had a dictionary of 4 URLs in it.



```

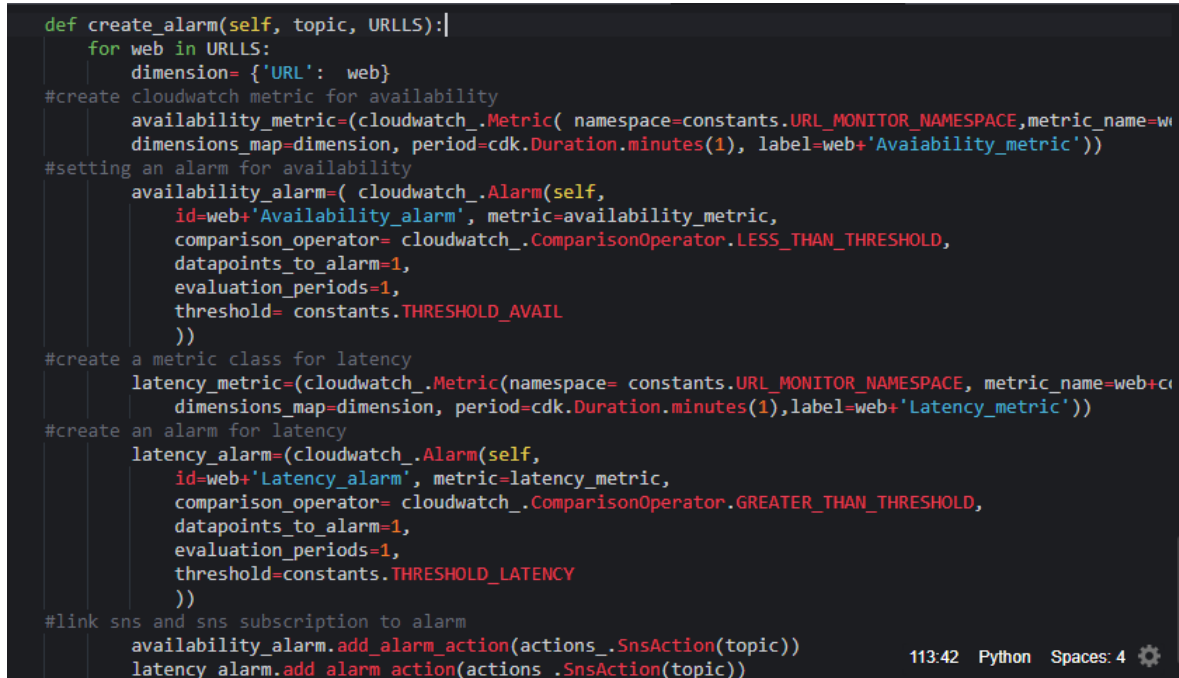
1  [
2    "w1": "www.skipq.org",
3    "w2": "https://www.dawn.com/",
4    "w3": "www.amazon.com",
5    "w4": "https://www.youtube.com/"
6  ]

```

Figure 7 Content in URLs.txt file

1.7.1 Creating Metrics and Alarms on List of URLs

As of now, we have read the contents from the URLs file and we have got a list of 4 URLs. The next step is to create web health metrics for all of these URLs and then to create alarms on them. For that, we just implemented a FOR loop in which I am defining the unique metrics' name by appending the URL name itself just to differentiate. So I have modified my project stack file as shown below:



```

def create_alarm(self, topic, URLLS):
    for web in URLLS:
        dimension= {'URL': web}
        #create cloudwatch metric for availability
        availability_metric=(cloudwatch_.Metric( namespace=constants.URL_MONITOR_NAMESPACE,metric_name=w
        dimensions_map=dimension, period=cdk.Duration.minutes(1), label=web+'Availability_metric'))
        #setting an alarm for availability
        availability_alarm=( cloudwatch_.Alarm(self,
            id=web+'Availability_alarm', metric=availability_metric,
            comparison_operator= cloudwatch_.ComparisonOperator.LESS_THAN_THRESHOLD,
            datapoints_to_alarm=1,
            evaluation_periods=1,
            threshold= constants.THRESHOLD_AVAIL
        ))
        #create a metric class for latency
        latency_metric=(cloudwatch_.Metric(namespace= constants.URL_MONITOR_NAMESPACE, metric_name=web+c
        dimensions_map=dimension, period=cdk.Duration.minutes(1),label=web+'Latency_metric'))
        #create an alarm for latency
        latency_alarm=(cloudwatch_.Alarm(self,
            id=web+'Latency_alarm', metric=latency_metric,
            comparison_operator= cloudwatch_.ComparisonOperator.GREATER_THAN_THRESHOLD,
            datapoints_to_alarm=1,
            evaluation_periods=1,
            threshold=constants.THRESHOLD_LATENCY
        ))
        #link sns and sns subscription to alarm
        availability_alarm.add_alarm_action(actions_.SnsAction(topic))
        latency_alarm.add_alarm_action(actions_.SnsAction(topic))

```

1.7.2 Results on List of URLs

Now the alarms are being created for the metrics, which are found for each URL.

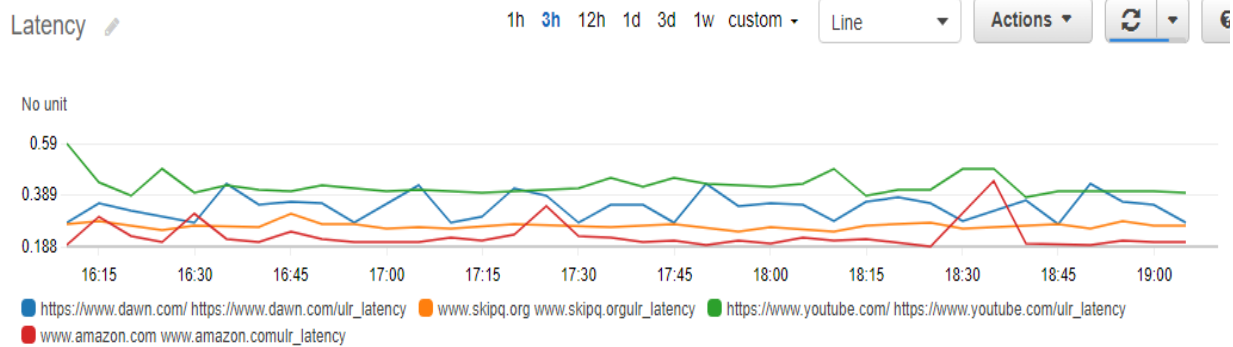


Figure 8 Graphed metrics of 4 URLs

1.8 Cloud Formation Diagram

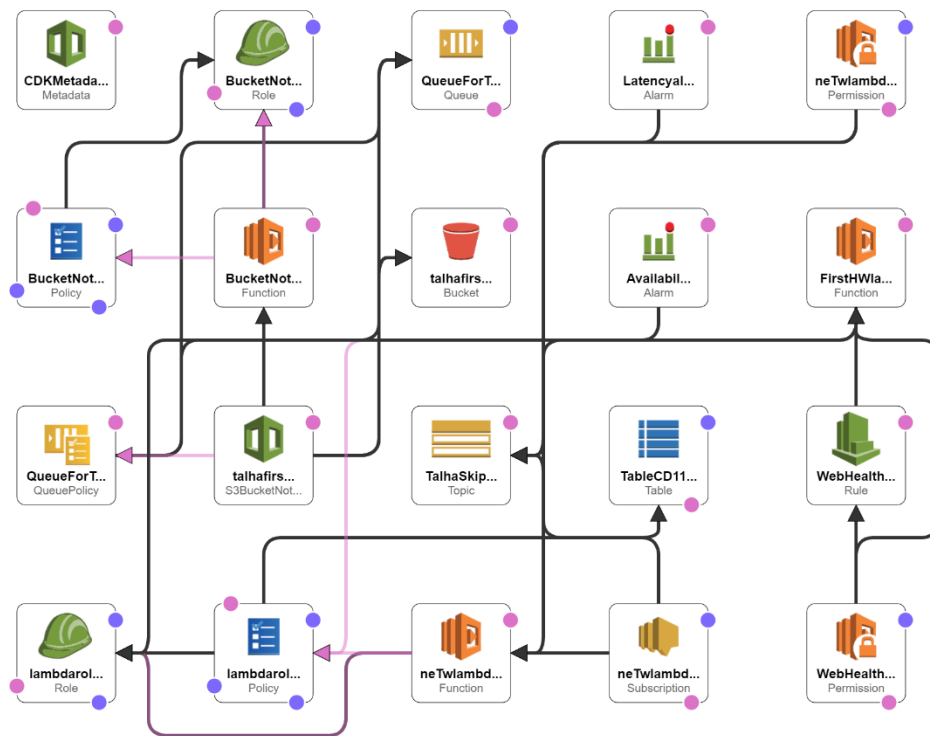


Figure 9 Cloud formation diagram

1.9 Related Issues

1.9.1 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 a solution, I deleted all items from my table and then deployed my code and it started writing to the 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 way to avoid, here in the following:

```
# # Talha_db_lambda.add_event_source(lambda_events_.SnsEventSource(topic))
#filter_policy={},
#dead_letter_queue=dead_letter_queue))
#dblamba_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=[dblamba_target])
```

X

2 Sprint2: Multi-Stage CI/CD Pipeline using AWS CDK

2.1 Overview

In sprint 2, we will create a CI/CD pipeline while will consist of four phases. To automate the build, test, and deploy processes, we use CodePipelines, which enable us to deliver features or updates. The code can be built, tested, and deployed in a test environment as well as directly in production environment. Whenever an error occurs after a commit, the pipeline rolls back and does not accept that change.

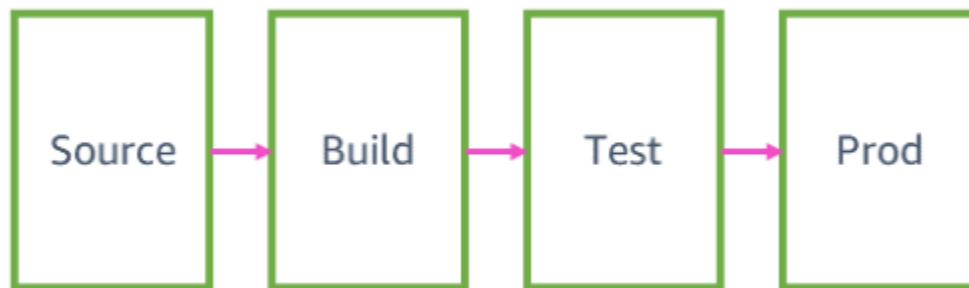


Figure 10 Phases of CD pipeline

2.2 Setting up Source Stage

2.2.1 Source from Repository

The very first step in the process is to get the source repository as the source. We can get the source by specifying the name of the repository either from GitHub or from codeCommit. I have done the same with GitHub as shown below in the code. I am using my access token that was saved in the secrets manager in the same environment (us-east-2). And the trigger is being configured on any changes/commits to the code. And I set up the repository source in our pipeline stack file.

```
source=pipelines.CodePipelineSource.git_hub(repo_string="talha2021skipq/ProximaCentauri",
branch='main',
authentication=cdk.SecretValue.secrets_manager('talha_pipeline_sprint2'),
trigger=captions.GitHubTrigger.POLL)
```

The next step is to prepare the environment i.e. to install the required libraries that will be used within the project (source code). While doing this manually we need to use commands in the terminal, but to automate it we employ *ShellStep()* method of *aws-cdk.pipelines*.

```
synth = pipelines.ShellStep("synth",
```



```
input=source,
commands=["cd talha/sprint2/talha_project",
          "pip install -r requirements.txt",
          "npm install -g aws-cdk", "cdk synth"
          #,"npm ci", "npm run build", "npx cdk synth"
          ], primary_output_directory="talha/sprint2/talha_project/cdk.out")
#Creating a pipeline for Codes, mainly to deploy CDK apps
pipeline=pipelines.CodePipeline(self, "Pipeline", synth=synth)
```

2.2.1.1 Files tree

To deploy my code pipeline, I have to make some changes in my app.py file. And I have to call my pipeline stack from it. Moreover, I have to include one infra-stage.py file as well, which has a stage class and it is called from the pipeline stack.

```
from aws_cdk import core

#from talha_project.talha_project_stack import TalhaProjectStack
from talha_project.talha_pipeline_stack import TalhaPipelineStack

app = core.App()
TalhaPipelineStack(app, 'TalhaPipelineStack', env=core.Environment(account='315997497220',
region='us-east-2'))
app.synth()
```

The file tree is shown below:

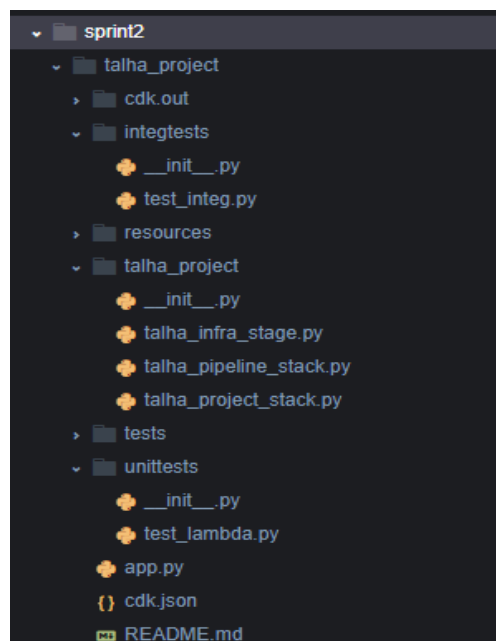


Figure 11 File tree for sprint2

The code repository source is now set up successfully by running command `cdk deploy TalhaPipelineStack`. Now I can go to Code Pipelines from my console to see my pipeline.

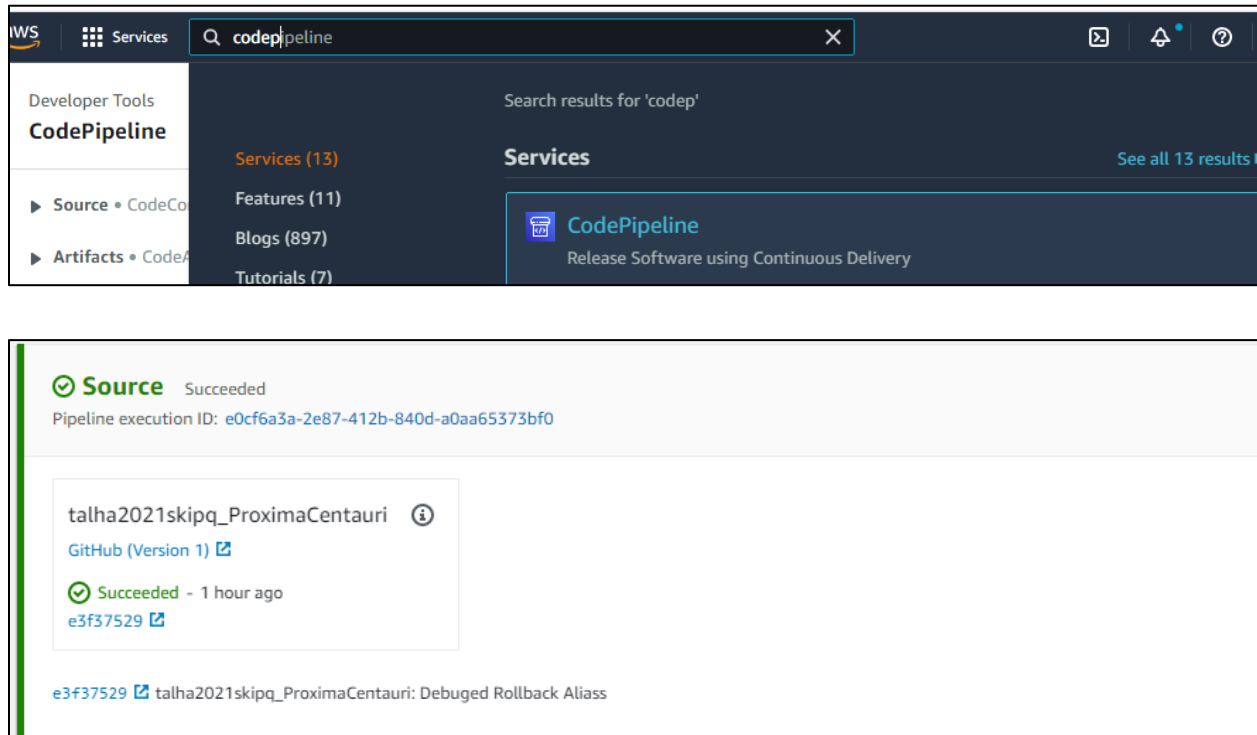


Figure 12 Source in the code pipeline

2.3 Build and Update Stages

2.3.1 Build and Auto Update in Pipeline

As our source is a repository located at GitHub so we have to take the source and then build the environment to deploy it. We can first collect all the required resources from AWS, and then we can build the project in the specified environment by using the `cdk synth` command in ShellStep. We can employ auto-update by commit feature in our code pipeline by using the method of `GitHubTrigger.POLL` from `CodePipelines_actions`.

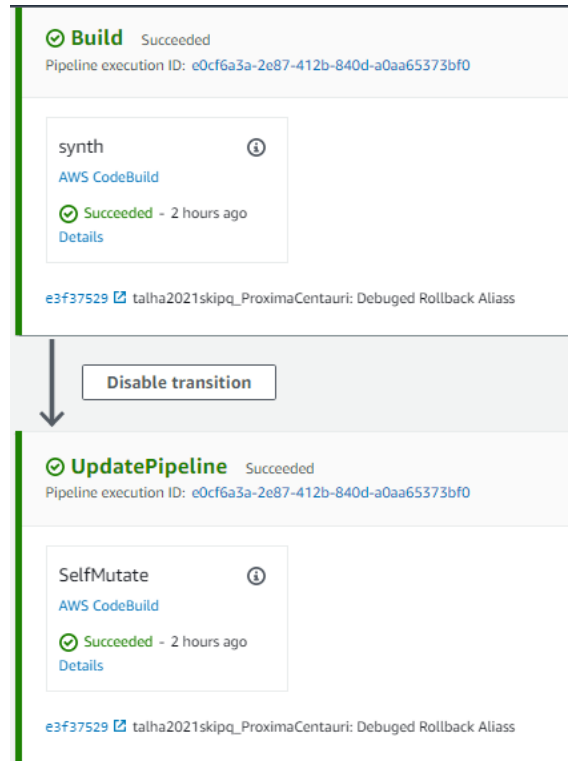


Figure 13 Build phase

2.4 Beta Stage for Code Deployment

2.4.1 Pre-requisite: Assets

Before it goes to the beta stage, the pipeline needs to get the assets from the toolkit generated by the source phase. Therefore, assets phase is a pre-requisite step for the beta stage.

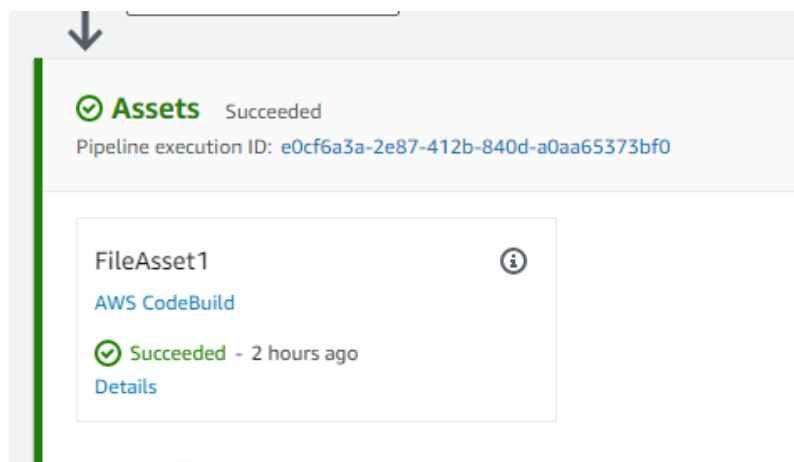


Figure 14 Assets in code pipeline

2.4.2 Adding Beta Stage with Pre-Unit Test

We have created a beta stage for our pipeline, which deploys the project code in the specified region. Before the deployment, it has to go through some unit tests, which will be discussed in the later sections.

```
#Creating a pipeline for Codes, mainly to deploy CDK apps
pipeline=pipelines.CodePipeline(self, "Pipeline", synth=synth)
##### Defining beta stage to my code pipeline #####
beta= TalhaInfraStage(self, "Beta",
env={
    'account': '315997497220',
    'region': 'us-east-2'
})

unit_test=pipelines.ShellStep('unit_test',
    commands=[ "cd talha/sprint2/talha_project",
               "pip install -r requirements.txt",
               "pytest unittests", "pytest integtests"] )
##### Adding beta stage to pipeline with pre test #####
pipeline.add_stage(beta,pre=[unit_test])
```

Now the pipeline from code pipelines looks like this:

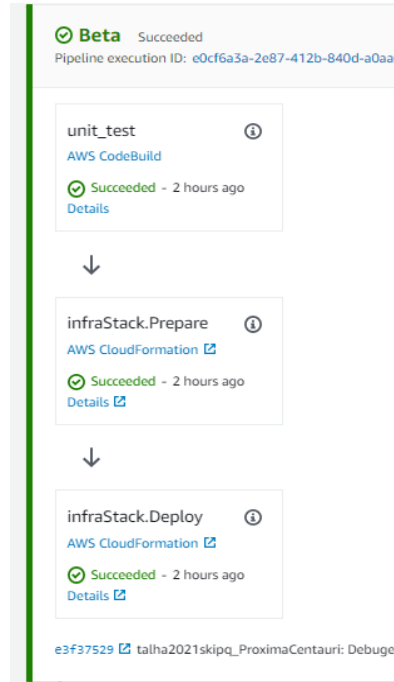


Figure 15 Beta stage in code pipeline

2.4.3 Unit Tests and Integration Tests

Unit testing is actually to test individual modules separately without interaction with dependencies. In the beginning, I have added a unit test for counting and validating the number of lambdas in our project. As we have only two lambda functions in our stack, so I am validating this in my unit test. While Integration testing means checking if different modules are working fine when combined as a group. So far, I have just put a true condition in my integration test, but later on, we can add integration tests to it.

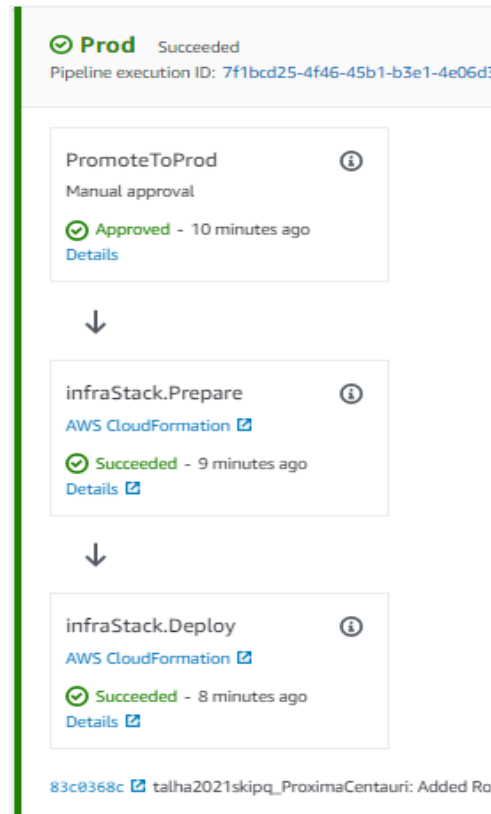
```
print2/talha_project/talha_project/talha_infra_stage.py
from aws_cdk import core
#import aws_cdk.assertions as assertions
from talha_project.talha_project_stack import TalhaProjectStack
def test_lambda():
    app=core.App()
    stack=TalhaProjectStack(app, 'infStack')
    #template = assertions.Template.from_stack(stack)
    template=app.synth().get_stack_by_name('infStack').template
    functions= [resource for resource in template['Resources'].values() if resource['Type']=='AWS::Lambda::Function']
    assert len(functions)==2
```

2.5 Production Stage with Manual Approval

2.5.1 Adding Production Stage with Manual Approval

After beta testing, almost everything must be fine, as our code had gone through unit testing as well integration testing. So we can now move to the production stage, but for the sake of fit, we add a manual approval step before the production stage so that we may finally review the changes and then either approve or reject accordingly. We can deploy our code in some different regions as well so we always specify the stage while creating the stage. Here I have deployed it in the same region.

```
##### Defining and adding production stage in my pipeline
prod= TalhaInfraStage(self, "Prod",
    env={'account': '315997497220',
        'region': 'us-east-2'} )
pipeline.add_stage(prod,
    pre=[ pipelines.ManualApprovalStep("PromoteToProd") ])
```



2.6 Auto Roll Back on Alarm

2.6.1 Rollback on AWS Lambda Function Alarms

AWS has provided some default metrics in cloud watch. we can read those metrics and add them to our cloudwatch. In this task, we have to find the duration metric for our lambda function and then transfer the traffic to an alias lambda function, if the specified threshold is exceeded. So we have to define an alias of the lambda function. And using LambdaDeploymentGroup, we can implement the above-mentioned behavior of automatic rollback.

```
#####Creating Alarm on aws metrics for lambda duration #####
metricduration= cloudwatch_.Metric(namespace='AWS/Lambda', metric_name='Duration',
    dimensions_map={'FunctionName': Talha_db_lambda.function_name} )
failure_alarm=cloudwatch_.Alarm(self, 'FailureAlarm', metric=metricduration,
    threshold=350,
    comparison_operator= cloudwatch_.ComparisonOperator.GREATER_THAN_THRESHOLD,
    evaluation_periods=1)

##Defining alias for my web health lambda
try:
    db_alias=_lambda.Alias(self, "WLaambdaAlias",
        alias_name="TalhaWLaliass",
        version=HWlambda.current_version)
#### Defining code deployment group to auto roll back, on the basis of
#### aws lambda function's Alarm on metrics(Duration). #####
    codedeploy.LambdaDeploymentGroup(self, "id",alias=db_alias,
        alarms=[failure_alarm])
#ult: LambdaDeploymentConfig.CANARY 10PERCENT 5MINUTES
```

2.6.2 Failure Alarm Creation

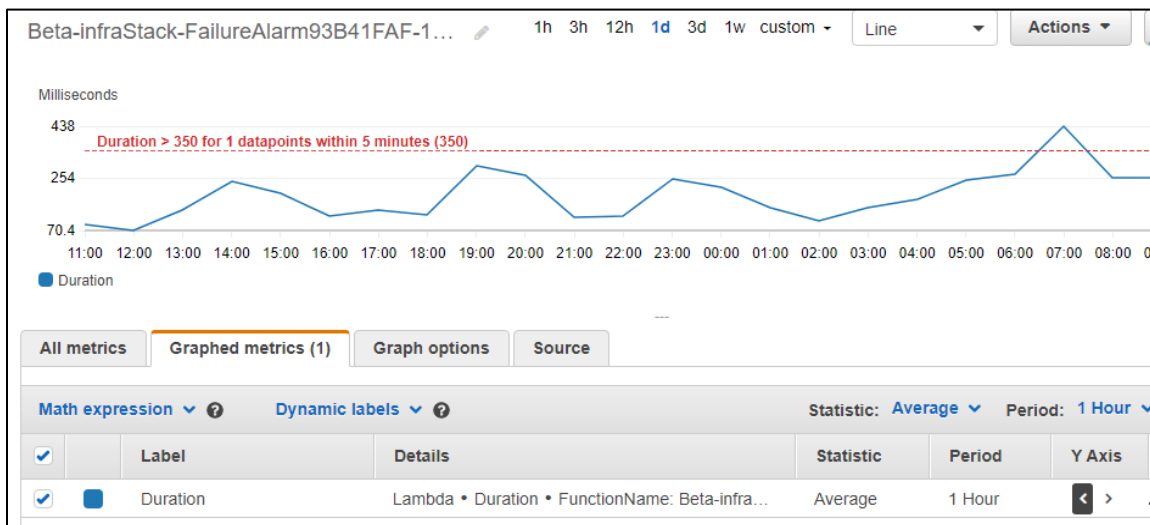
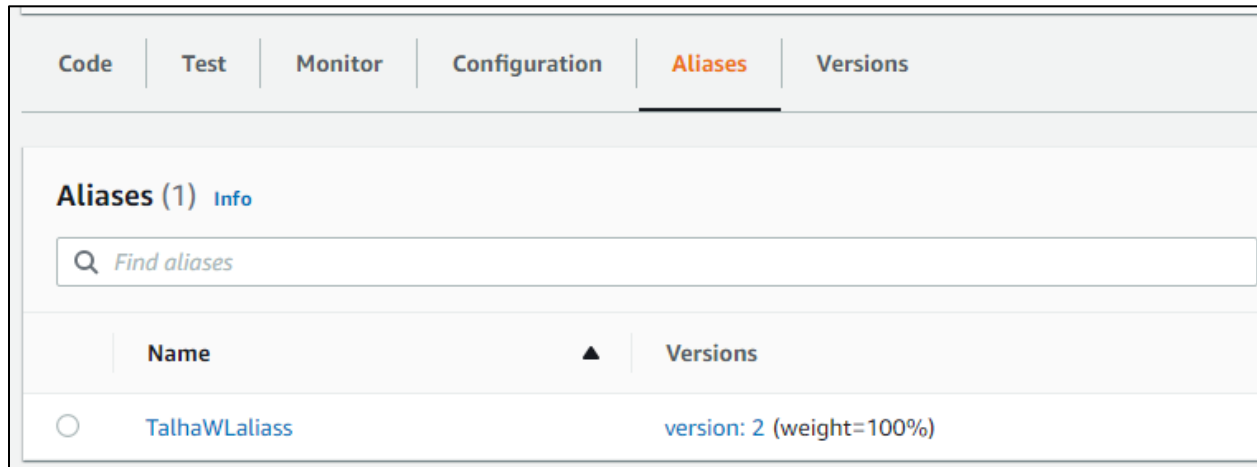


Figure 16 The alarm on lambda duration metric

2.6.3 Alias Generation

I generated an alias for the specified Lambda function. We can see this in the Aliases tab as shown below. Moreover, when the lambda is deployed, all the traffic does not shift to it once, but it will share 10% of the traffic to the new version and then after waiting for some time, it will act accordingly based on failure alarms, the supporting figure is shown below.



Here you can see your lambda's auto roll back in the Code Deploy → Deployment section from your pipeline.

Status	Deployment type	Compute platform	Application	Deployment group	Revision location	Initiating event	Start time	End time
✓ Succeeded	Blue/green	AWS Lambda	Beta-infraStack-TalhadApplicationE4ACB22B-94Q26J0ZAGF2	Beta-infraStack-TalhadEF86370C-RTDB56FXC9XO	323299ac...	CodeDeploy rollback	Dec 27, 2021 11:43 PM (UTC+5:00)	Dec 27, 2021 11:43 PM (UTC+5:00)

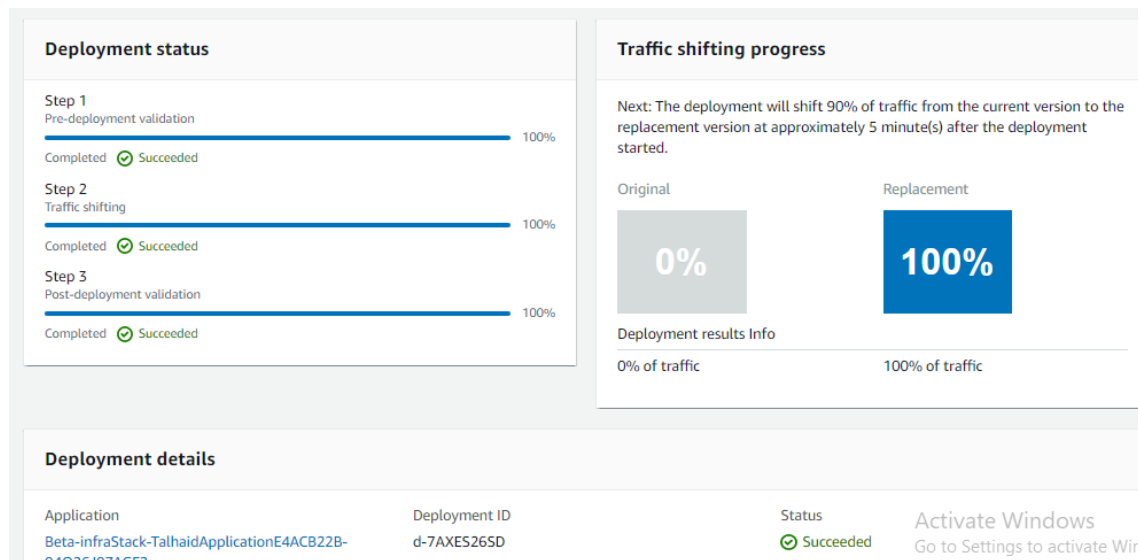


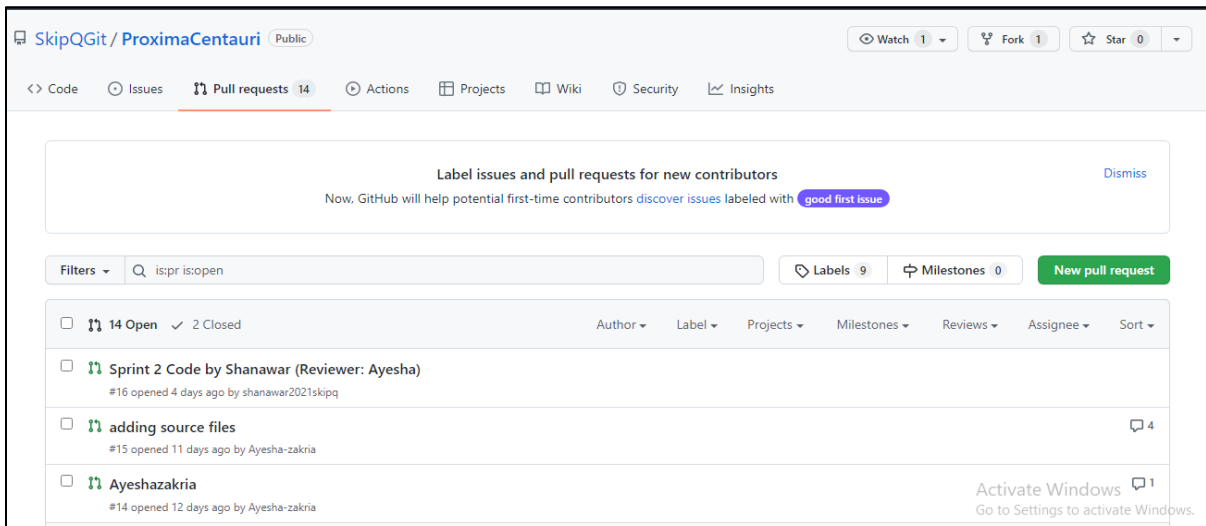
Figure 17 Traffic shifting progress on lambda

Revision details			
Revision location 623f73646db7ee55d931a03a047bcb27a276541733 0d6e328d74485039b7eee6	Revision created 11 minutes ago	Revision description Application revision registered by Deployment ID: 7AXES26SD	
Event	Status	Start time	End time
BeforeAllowTraffic	✔ Succeeded	Dec 27, 2021 10:45 PM (UTC+5:00)	Dec 27, 2021 10:45 PM (UTC+5:00)
AllowTraffic	✔ Succeeded	Dec 27, 2021 10:45 PM (UTC+5:00)	Dec 27, 2021 10:50 PM (UTC+5:00)
AfterAllowTraffic	✔ Succeeded	Dec 27, 2021 10:50 PM (UTC+5:00)	Dec 27, 2021 10:50 PM (UTC+5:00)

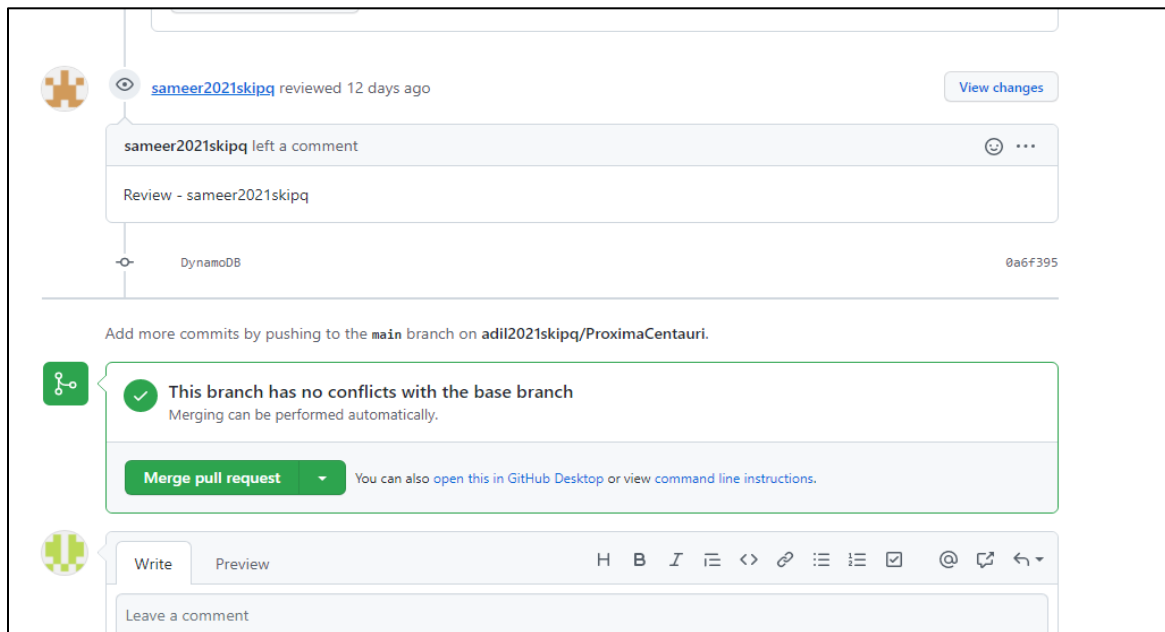
Figure 18 Traffic shift details

2.7 Merging the Pull Request

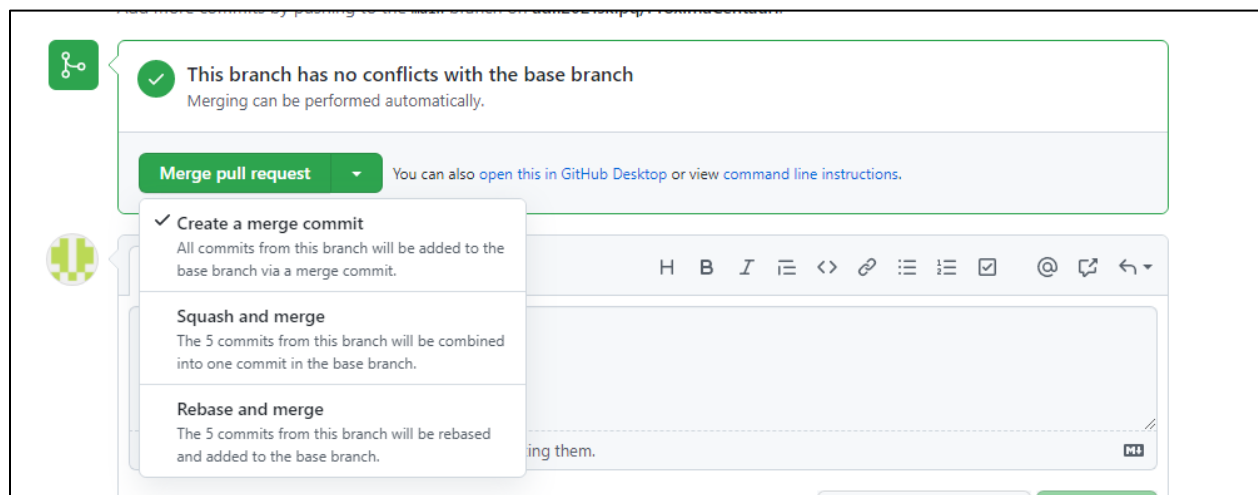
To merge the pull request that you have created recently, you have to follow these easy steps to merge the commits to the base branch.



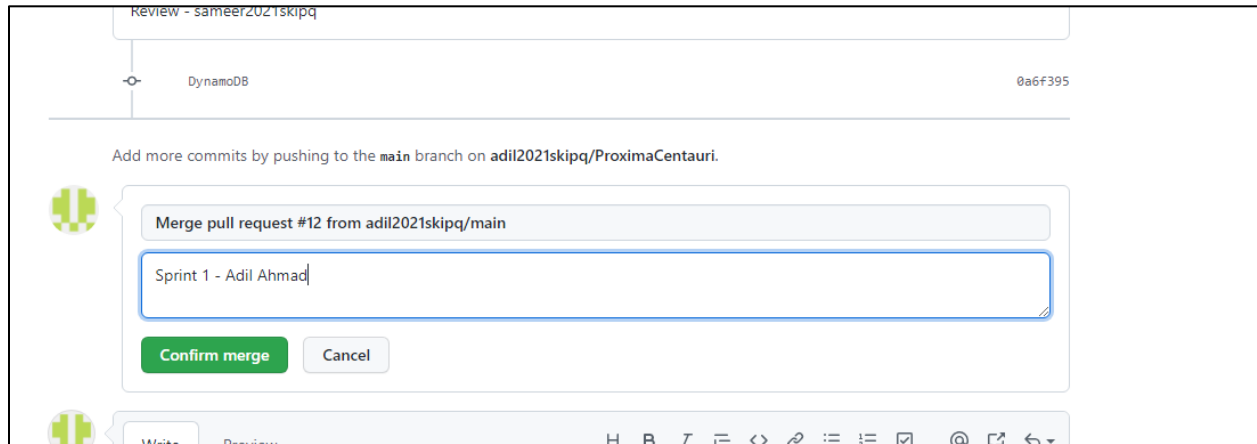
- 1- Open the Pull Request (PR) section where you can see all the open PRs. Like this:
- 2- Click on the PR that you want to merge and then scroll down to see merge option. There should be no conflict with the base branch to do it successfully. So it will look like:



3- Drop down the Merge pull request option to select 'Create a merge commit':



4- Add some optional description and click on 'Confirm Merge' button.



2.8 Related Issues

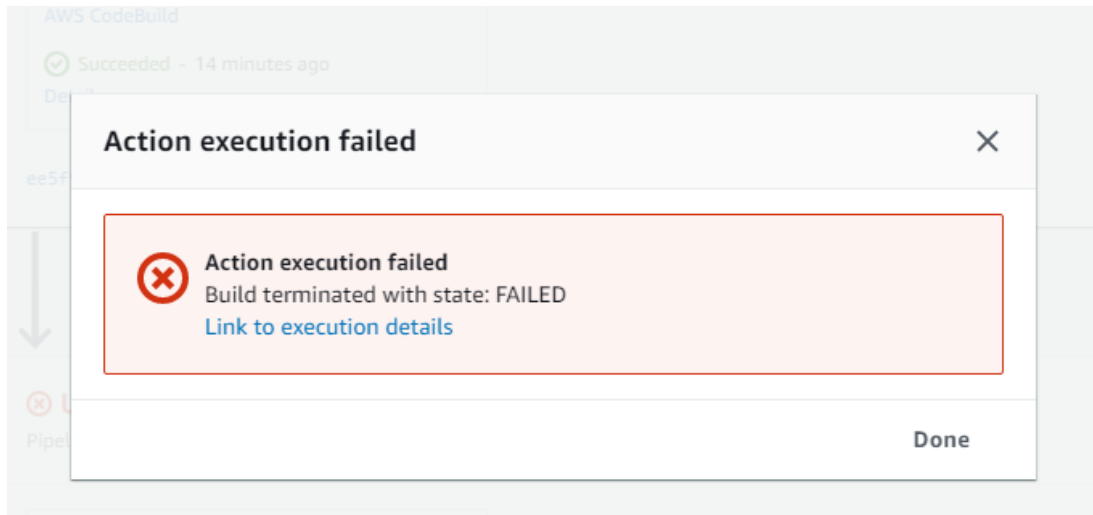
2.8.1 The issue in Lambda Alias:

Timestamp	Logical ID	Status	Status reason
2021-12-25 22:12:14 UTC+0500	neTwlambdaB0 2FCC4B7D	UPDATE_IN_P ROGRESS	-
2021-12-25 22:12:14 UTC+0500	neTwlambdaB0 03825D	UPDATE_IN_P ROGRESS	-
2021-12-25 22:12:06 UTC+0500	Beta-infraStack	UPDATE_ROL LBACK_IN_PR OGRESS	The following resource(s) failed to update: [LambdaAlias9C15A666].
2021-12-25 22:12:06 UTC+0500	LambdaAlias9C 15A666	UPDATE_FAIL ED	Rollback successful
2021-12-25 22:12:05 UTC+0500	LambdaAlias9C 15A666	UPDATE_IN_P ROGRESS	CodeDeploy rollback deployment started: d- 63RTICUQD

2.8.1.1 Reason and Solution

The updates in the same lambda alias are not supported. So I changed the name of my lambda's alias and it worked for me.

2.8.2 Execution failure issue in pipeline update



The error in the build logs tells that it is an error of

```
8 Toolkit stack: CDKToolkit
9 Setting "CDK_DEFAULT_REGION" environment variable to us-east-2
10 Resolving default credentials
11 Looking up default account ID from STS
12 Default account ID: 315997497220
13 Setting "CDK_DEFAULT_ACCOUNT" environment variable to 315997497220
14 context: {
15   'aws:cdk:enable-path-metadata': true,
16   'aws:cdk:enable-asset-metadata': true,
17   'aws:cdk:version-reporting': true,
18   'aws:cdk:bundling-stacks': [ '*' ]
19 }
20 --app points to a cloud assembly, so we bypass synth
21 No stacks match the name(s) TalhaPipelineStack
22 Error: No stacks match the name(s) TalhaPipelineStack
23   at CdkToolkit.validateStacksSelected (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:545:13)
24   at CdkToolkit.selectStacksForDeploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:492:10)
25   at CdkToolkit.deploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:120:20)
26   at initCommandLine (/usr/local/lib/node_modules/aws-cdk/bin/cdk.ts:267:9)
27
28 [Container] 2021/12/23 14:43:24 Command did not exit successfully cdk -a . deploy TalhaPipelineStack --require-approval=never --verbose exit status 1
29 [Container] 2021/12/23 14:43:24 Phase complete: BUILD State: FAILED
30 [Container] 2021/12/23 14:43:24 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
31 cdk -a . deploy TalhaPipelineStack --require-approval=never --verbose. Reason: exit status 1
32 [Container] 2021/12/23 14:43:24 Entering phase POST_BUILD
```

2.8.2.1 Reason and Solution

I was defining the wrong path in which the requirements are to be installed. So I updated my correct path and then deleted the previous stack before re-deployment.

2.8.3 CDKToolkit Not Found

As we are in a group and everyone is using the same environment, so the resources allocation for each IDE gets messed up. Hence, we need to specify a unique identifier to our CDK toolkit.

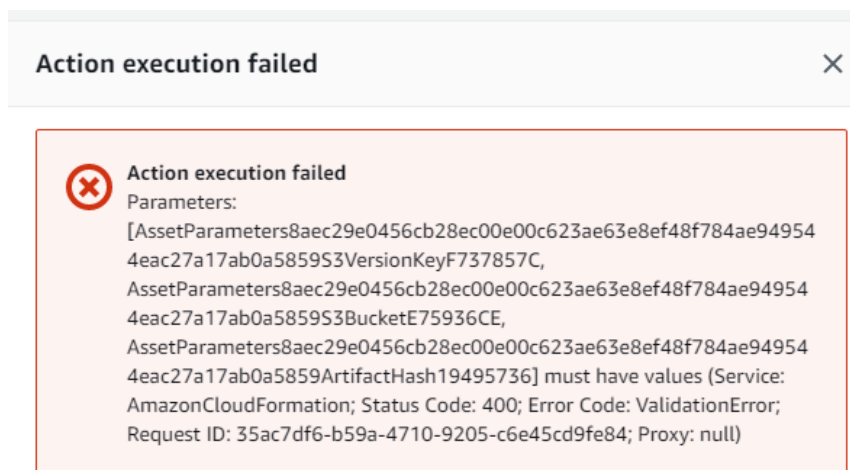
```
8 Toolkit stack: CDKToolkit
9 Setting "CDK_DEFAULT_REGION" environment variable to us-east-2
10 Resolving default credentials
11 Looking up default account ID from STS
12 Default account ID: 315997497220
13 Setting "CDK_DEFAULT_ACCOUNT" environment variable to 315997497220
14 context: {
15   'aws:cdk:enable-path-metadata': true,
16   'aws:cdk:enable-asset-metadata': true,
17   'aws:cdk:version-reporting': true,
18   'aws:cdk:bundling-stacks': [ '*' ]
19 }
20 --app points to a cloud assembly, so we bypass synth
21 No stacks match the name(s) TalhaPipelineStack
22 Error: No stacks match the name(s) TalhaPipelineStack
23   at CdkToolkit.validateStacksSelected (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:545:13)
24   at CdkToolkit.selectStacksForDeploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:492:10)
25   at CdkToolkit.deploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:120:20)
26   at initCommandLine (/usr/local/lib/node_modules/aws-cdk/bin/cdk.ts:267:9)
27
28 [Container] 2021/12/23 16:16:00 Command did not exit successfully cdk -a . deploy TalhaPipelineStack --require-
29 approval=never --verbose exit status 1
30 [Container] 2021/12/23 16:16:00 Phase complete: BUILD State: FAILED
31 [Container] 2021/12/23 16:16:00 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
32 cdk -a . deploy TalhaPipelineStack --require-approval=never --verbose. Reason: exit status 1
33 [Container] 2021/12/23 16:16:00 Entering phase POST_BUILD
34 [Container] 2021/12/23 16:16:00 Phase complete: POST_BUILD State: SUCCEEDED
35 [Container] 2021/12/23 16:16:00 Phase context status code: Message:
```

2.8.3.1 Solution

- 1- Delete all buckets from S3(except the ones you defined manually)
- 2- Delete pipeline stack(s) from cloud formation
- 3- Delete pipeline from codepipelines
- 4- Run the command: `cdk bootstrap aws://12-digit-id/region --qualifier talha --toolkit-stack tkname`

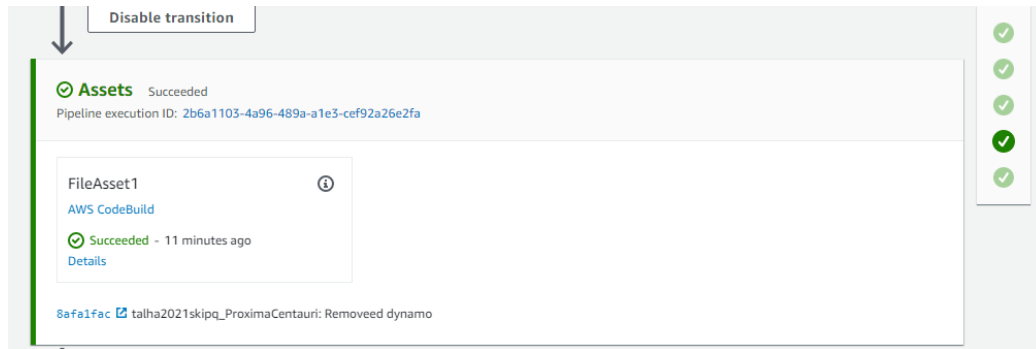
2.8.4 Execution Failure in Beta Stage

the following error occurred in the preparation step of the beta stage. This error occurred due to the non-existence of the Assets stack for the beta. The error is shown below:



2.8.4.1 Solution

As the error occurred because of not getting the assets stack for the beta. So there might be some internal error while bootstrapping. I resolved this issue by deleting my beta infra stack and then doing the bootstrap again. By this, we get our assets succeeded before going to the beta step.



2.8.5 Issue in Accessing Dynamodb Table

There was an issue in pipeline deployment in the beta stage. As I was running my beta stage in the same region in which I was doing the previous sprint, so the dynamodb table was already created in that stack so it was not accessible in my beta stack. To make it run I had to create a table with a new name and then grant it read-write access. So this worked for me when I changed the dynamodb table name.

2.8.6 Internal failure

- Current credentials could not be used

```
Immediate (Bash) x bash - "ip-172-31-32-14" x talha_project/resources/s x +
current credentials could not be used to assume 'arn:aws:iam::315997497220:role/cdk-hnb659fds-deploy-role-315997497220-us-east-2', but are for the right account. Proceeding anyway.

X TalhaPipelineStack failed: Error: TalhaPipelineStack: SSM parameter /cdk-bootstrap/hnb659fds/version not found. Has the environment been bootstrapped? Please run 'cdk bootstrap' (see https://docs.aws.amazon.com/cdk/latest/guide/bootstrapping.html)
  at CloudFormationDeployments.validateBootstrapStackVersion (/home/ubuntu/.nvm/versions/node/v16.3.0/lib/node_modules/aws-cdk/lib/api/cloudformation-deployments.ts:323:13)
  at processTicksAndRejections (node:internal/process/task_queues:96:5)
  at CloudFormationDeployments.publishStackAssets (/home/ubuntu/.nvm/versions/node/v16.3.0/lib/node_modules/aws-cdk/lib/api/cloudformation-deployments.ts:298:7)
  at CloudFormationDeployments.deployStack (/home/ubuntu/.nvm/versions/node/v16.3.0/lib/node_modules/aws-cdk/lib/api/cloudformation-deployments.ts:202:5)
  at CdkToolkit.deploy (/home/ubuntu/.nvm/versions/node/v16.3.0/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:194:24)
  at initCommandLine (/home/ubuntu/.nvm/versions/node/v16.3.0/lib/node_modules/aws-cdk/bin/cdk.ts:267:9)
TalhaPipelineStack: SSM parameter /cdk-bootstrap/hnb659fds/version not found. Has the environment been bootstrapped? Please run 'cdk bootstrap'
talhanaeemskipq:~/environment/Talha_Sprint2/ProximaCentauri/talha/sprint1/talha_project (main) $ cdk bootstrap --qualifier "talha"

X Bootstrapping environment aws:///315997497220/us-east-2...
Trusted accounts for deployment: (none)
Trusted accounts for lookup: (none)
Using default execution policy of 'arn:aws:iam::aws:policy/AdministratorAccess'. Pass '--cloudformation-execution-policies' to customize.
CDKToolkit: creating CloudFormation changeset...
```

2.8.6.1 Solution

These errors might occur due to an existing provisioned bootstrap stack with the same qualifier (default name), which happens in the case when you are using two bootstrap stacks in the same environment.

So run this command to specify a unique qualifier:

```
cdk bootstrap --qualifier "your_qualifiers_name" --toolkit "TK name"
```

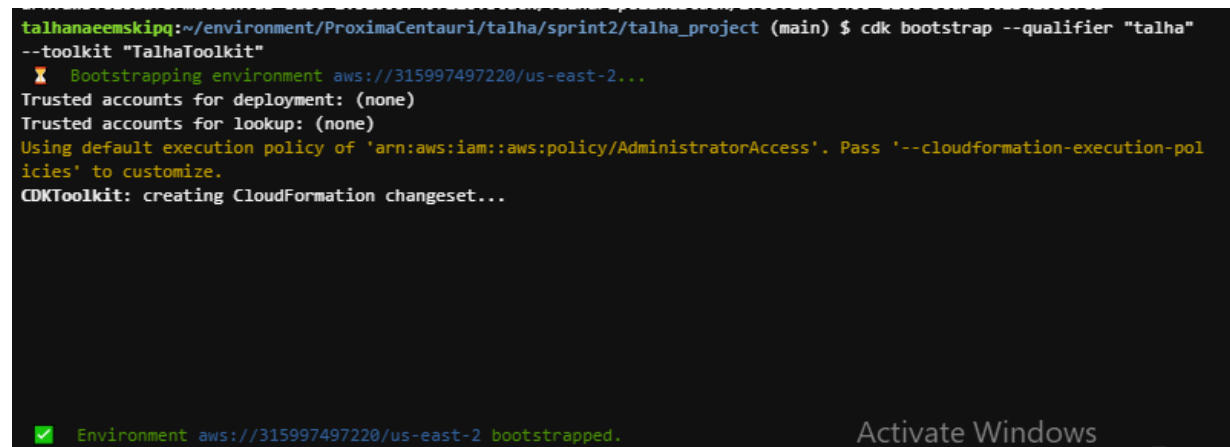
After this, you have to specify this qualifier in your cdk.json file as well, append it at the end of the file,

```
"context": { "@aws-cdk/core:bootstrapQualifier": " your_qualifiers_name " }
```

Now do the bootstrap using the following command and it should work.

```
cdk bootstrap aws://12-digit-id/region --qualifier name --toolkit-stack cdkname
```

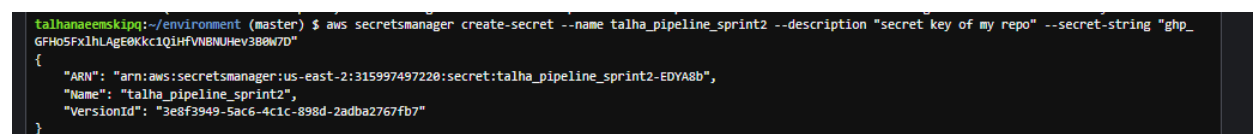
In this way, we get the environment bootstrapped:



```
talhanaemskipq:~/environment/ProximaCentauri/talha/sprint2/talha_project (main) $ cdk bootstrap --qualifier "talha" --toolkit "TalhaToolkit"
🔥 Bootstrapping environment aws://315997497220/us-east-2...
Trusted accounts for deployment: (none)
Trusted accounts for lookup: (none)
Using default execution policy of 'arn:aws:iam::aws:policy/AdministratorAccess'. Pass '--cloudformation-execution-policies' to customize.
CDKToolkit: creating CloudFormation changeset...

✅ Environment aws://315997497220/us-east-2 bootstrapped.
```

Note: Another technicality is that you must have checked carefully in which region you have specified your secret key and in which region you are now deploying your pipeline. These should match.



```
talhanaemskipq:~/environment (master) $ aws secretsmanager create-secret --name talha_pipeline_sprint2 --description "secret key of my repo" --secret-string "ghp_GFH05Fx1hLAgE0Kkc1QihFVNBNuhev380W7D"
{
  "ARN": "arn:aws:secretsmanager:us-east-2:315997497220:secret:talha_pipeline_sprint2-EDYA8b",
  "Name": "talha_pipeline_sprint2",
  "VersionId": "3e8f3949-5ac6-4c1c-898d-2adba2767fb7"
}
```

Figure 19 Saving secret key

3 Sprint3: API for Web Crawler

3.1 Overview

In this sprint, we have to make a CRUD API gateway for the web crawler, so the hat user; a third party can interact with our web crawler. In the API, we have added create, read, update and delete (CRUD) operations.

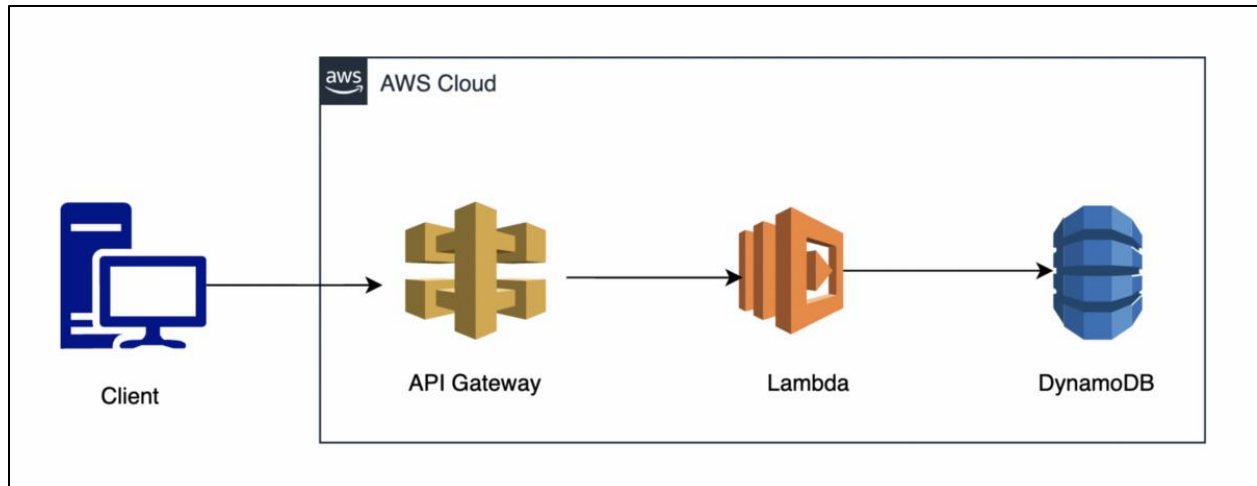


Figure 20 AWS API gateway

3.2 Dynamodb Table for URLs

3.2.1 Data Shift from S3 to Dynamo Table

The first step is to shift the URLs data from the S3 bucket to our dynamodb table. A separate table was created for URLs with only one key i.e. partition key.

```
##### Creating URL Table #####\ndef create_url_table(self):\n    return db_.Table(self,id="URLTable",partition_key=db_.Attribute(name="URL", type=db_.AttributeType.STRING))
```

After table creation, we have to shift the URLs from the S3 bucket to the dynamodb table. Whenever there is a new file uploaded in the S3 bucket, the specific lambda will invoke to read contents from the new file and put the data into dynamodb table.


```
##### Creating lambda to Add URLs to dynamodb Table from S3 bucket #####
db_lambda_role = self.create_db_lambda_role()
lambdaforurl = self.create_lambda('OneTimeLambda', './resources', 's3lambda.lambda_handler', db_lambda_role,
environment={'table_name':urltablename})
URLtable.grant_full_access(lambdaforurl)
##### Event : Whenever a file is uploaded to S3 bucekt #####
bucket = s3.Bucket(self, "TalhaS3Bucket")
lambdaforurl.add_event_source(sources.S3EventSource(bucket,events=[s3.EventType.OBJECT_CREATED],
filters=[s3.NotificationKeyFilter(suffix=".json")]))
print(urltablename)
```

The handler file for the 'lambdaforurl' will read file from S3 bucket and write the content into Dynamo dB table.

```
1 import boto3
2 import os
3
4 import s3bucket_url #import s3bucket_read as bucket
5
6 def lambda_handler(event,context):
7     value = dict()
8     bucketname = event['Records'][0]['s3']['bucket']['name']
9     filename = event['Records'][0]['s3']['object']['key']
10
11     client = boto3.client('dynamodb')
12     list_url= s3bucket_url.read_url_list(bucketname,filename)
13     tablename = os.getenv('table_name')#getting table name
14     for url in list_url:
15         client.put_item(Tablename= tablename,Item={'URL':{'S' : url}})
```

3.3 Lambda Restful API Creation

3.3.1 Creating API Gateway

I have created an API gateway using aws-apigateway module from AWS CDK. In the infra stack I am creating this API and passing it the backed lambda function for backend processing. In addition, I have given some permissions to my backend lambda like, read_write in dynamodb table and invocation by API gateway,

```
# Creating backend lambda for api gateway
apibackendlambda=self.create_dblambda('ApiLambda', './resources','backend_lambda.lambda_handler',db_lambda_role,
environment={'tablename':urltablename})##, "mytopic":topic.topic_arn})
apibackendlambda.grant_invoke(aws_iam.ServicePrincipal("apigateway.amazonaws.com"))
URLtable.grant_read_write_data(apibackendlambda)
URLtable.grant_read_write_data(HWlambda)
#Create API gateway
api=apigateway.LambdaRestApi(self, "TalhaAPI",handler=apibackendlambda)
items = api.root.add_resource("items")
items.add_method("GET") # GET /items
items.add_method("PUT") # Allowed methods: ANY,OPTIONS,GET,PUT,POST,DELETE,PATCH,HEAD POST /items
items.add_method("DELETE")
```

The API has been created with the specified methods and we can view it by going to API gateway from AWS console.

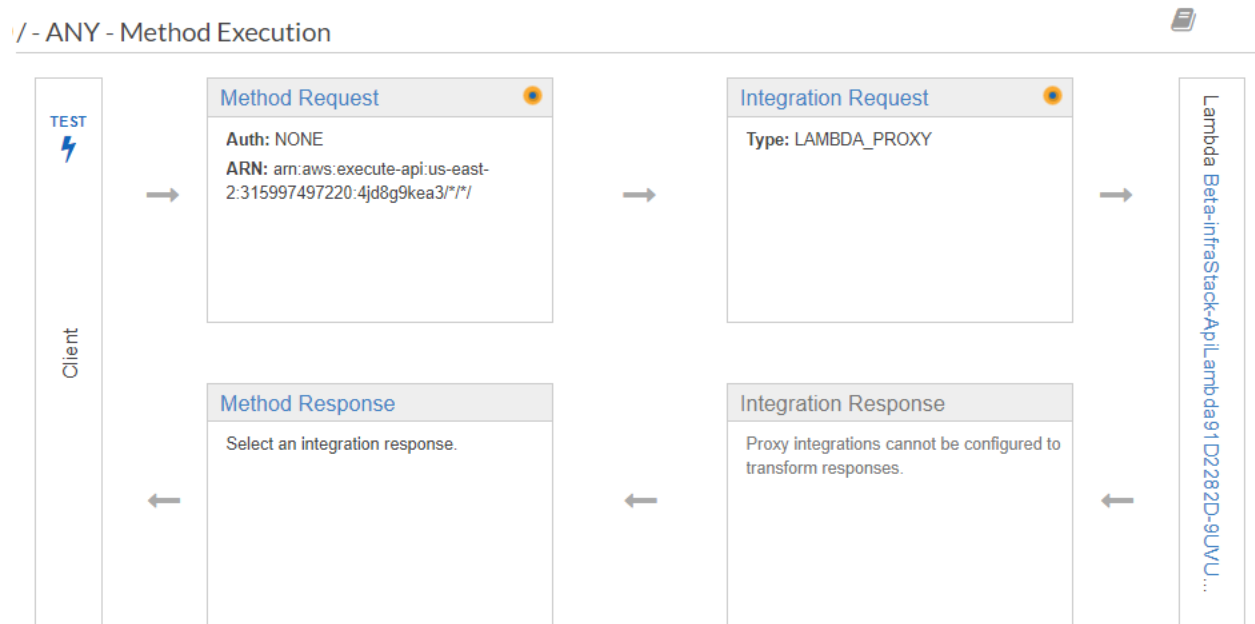


Figure 21 API gateway

Now the API is ready to be tested. After clicking on Test, the following window will appear and we can select the method from drop down menu of Method:

← Method Execution / - ANY - Method Test



Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method

Method

GET

Request: /

Status: 200

Latency: 2338 ms

Path

No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No [stage variables](#) exist for this method.

Response Body

```
{
  "Response": "Your request is acknowledged",
  "httpMethod": "GET",
  "body": [
    {
      "URL": "https://www.dawn.com/"
    },
    {
      "URL": "www.python.org"
    },
    {
      "URL": "www.amazon.com"
    },
    {
      "URL": "www.skipq.org"
    },
    {
      "URL": "www.ieee.org"
    }
  ]
}
```

Activate Windows

Go to Settings to activate Windows.

Figure 22 API GET method

To run PUT or DELETE methods, you have to give the URL that is to be deleted.

DELETE ▾

Path

No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No [stage variables](#) exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

1

www.skipq.org

Status: 200

Latency: 2117 ms

Response Body

```
{
  "Response": "Your request is acknowledged",
  "httpMethod": "GET",
  "body": [
    {
      "URL": "https://www.dawn.com/"
    },
    {
      "URL": "www.python.org"
    },
    {
      "URL": "www.amazon.com"
    },
    {
      "URL": "www.skipq.org"
    },
    {
      "URL": "www.ieee.org"
    },
    {
      "URL": "https://www.youtube.com/"
    }
  ]
}
```

Response Headers

Figure 23 API DELETE method

3.3.2 Backend Lambda for API Gateway

I created a lambda function to work as backend for my API gateway. The backend lambda is invoked on every event request from API.

```
def lambda_handler(events, context):
    print(events)
    db=putdb.dynamoTablePutURLData()
    opt=events['httpMethod']
    path=events['path']
    table_name= os.environ['tablename']
    msg=""
    if opt=='PUT':      #####////////PUT////////
        urls=events['body']#.split()
        db.wdynamo_data(table_name,urls)
        msg="The item has been successfully written."
    elif opt=='GET':    #####////////GET////////
        urllist=db.rdynamo_data(table_name)
        msg="Your request is acknowledged"
        events['body']=urllist
```

```
elif opt=='DELETE': #####////////DELETE////////
    urltodel=events['body']
    response=db.ddynamo_data(table_name,urltodel)
    msg="The Url has been deleted. Use GET method to check!"
else:
    print("Please select an appropriate option. Appropriate Options=[PUT, GET, DELETE]")
datares={"Response" : msg, "httpMethod": events['httpMethod'], "body": events['body'] }
return {'statusCode':200 , 'body':json.dumps(datares)}
```

3.4 Handler Functions for API Requests

3.4.1 Writing Data from API to Dynamodb Table

Whenever a PUT request is generated at the API, we have to take the URL and write it in the dynamodb table. We can do this by using a Boto3 resource of dynamodb.

3.4.1.1 PUT Request Handler

To put data in the dynamodb table ,we use the put_item function.

```
def wdynamo_data(self, tableName, message):
    #db=boto3.client('dynamodb')
    table = self.resource.Table(tableName)
    values = {}
    values['URL'] = message
    #values['Reason'] = reason
    table.put_item(Item = values)
```

3.4.1.2 GET Request Handler

For scanning data from the dynamodb table, we have to read all the entities from the table and return them to API gateway response.

```
def rdynamo_data(self,tableName):
    dynamodb = boto3.resource('dynamodb')
    table = self.resource.Table(tableName)
    table = dynamodb.Table(tableName)
    response = table.scan()
    data = response['Items']
    while 'LastEvaluatedKey' in response:
        response = table.scan(ExclusiveStartKey=response['LastEvaluatedKey'])
        data.extend(response['Items'])
    return data
#Function for dynamodb table to delete an element
```

3.4.1.3 DELETE Request Handler

To delete an entity we have to get the entity from the API gateway and use the delete_item table.

```
#Function for dynamodb table to delete an element
def ddynamo_data(self,tableName,message):
    dynamodb = boto3.resource('dynamodb')
    table = self.resource.Table(tableName)
    #table = self.client.Table(tableName)
    response = table.delete_item(Key={'URL': message})
    return response
```

3.5 Auto- Creation of Metrics on New URLs

3.5.1 Problem Statement

Whenever a method is called at API, the backend Lambda function gets automatically invoked to respond to the query. The URLs table is also updated accordingly. But the cloudwatch metrics are created once in the infra-stack. We want to automate it.

3.5.2 Proposed Solution-I

We shift the code for Cloudwatch Metrics and alarms creation to the backend Lambda function. While the SNS topic was created in an infra-stack file so, we have passed topic ARN to its environment. So that using a boto3 resource, we may take the topic and perform the necessary actions on it. For that, we need some cdk modules to be imported into the lambda handler file. I have an issue while importing aws_cdk, and which is logical, as Lambda is a server-less application.

No older events at this moment. Retry	
▶ 2021-12-31T15:51:39.316+05:00	START RequestId: 1d3dc387-0773-455f-9bc8-d123fd97caa1 Version: \$LATEST
▶ 2021-12-31T15:51:39.319+05:00	31 Dec 2021 10:51:39,316 [INFO] (/var/runtime/bootstrap.py) main started at epoch 1640947899316
▶ 2021-12-31T15:51:39.319+05:00	31 Dec 2021 10:51:39,319 [INFO] (/var/runtime/bootstrap.py) init complete at epoch 1640947899319
▼ 2021-12-31T15:51:39.321+05:00	Unable to import module 'backend_lambda': No module named 'aws_cdk' Unable to import module 'backend_lambda': No module named 'aws_cdk'
Copy	
▶ 2021-12-31T15:51:39.322+05:00	END RequestId: 1d3dc387-0773-455f-9bc8-d123fd97caa1
▶ 2021-12-31T15:51:39.322+05:00	REPORT RequestId: 1d3dc387-0773-455f-9bc8-d123fd97caa1 Duration: 1.57 ms Billed Duration: 2 ms Memory Size: 128 MB Max Mem...
▶ 2021-12-31T15:52:24.694+05:00	START RequestId: 9b5bd2fd-988a-4058-85ad-f4da8cd2612c Version: \$LATEST
▼ 2021-12-31T15:52:24.696+05:00	Unable to import module 'backend_lambda': No module named 'aws_cdk' Unable to import module 'backend_lambda': No module named 'aws_cdk'
Copy	
▶ 2021-12-31T15:52:24.697+05:00	END RequestId: 9b5bd2fd-988a-4058-85ad-f4da8cd2612c

3.5.3 Proposed Solution-II

Another possible solution could be to invoke another lambda function whenever an activity occurs at API gateway i.e. whenever the backend lambda invokes. I tried some methods of lambda function like `add_invoke()` but could not succeed. This context can be streamed another way as well, which is to use `lambda_event_source` from aws. However, in this service we have option of creating an event source of dynamodb.

3.6 Unit and Integration Testing

3.6.1 Unit Testing

3.6.1.1 Test-I: Lambda Test

```
##### TEST 1: Lambda functions #####
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_lambda():
    functions= [resource for resource in template['Resources'].values() if resource['Type']=='AWS::Lambda::Function']
    assert len(functions)>=4
```

3.6.1.2 Test-II: Alarms Test

```
##### TEST 2: Alarms on metrics #####
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_alarms():
    functions= [resource for resource in template['Resources'].values() if resource['Type']=='AWS::CloudWatch::Alarm']
    assert len(functions)>3
    #for metrics and for failure alarm 0total
```

3.6.1.3 Test-III: Bucket Test

```
##### TEST 3: S3 bucket Test #####
#Make sure that we have a bucket(necessary condition)
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_bucket():
    buckets= [resource for resource in template['Resources'].values() if resource['Type']=='AWS::S3::Bucket']
    assert len(buckets)>=1
```

3.6.1.4 Test-IV: Dynamodb Table Test

```
##### TEST 4: Dynamodb Tables #####
#Make sure that we have a Table in which we will store the URLs
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_table():
    tables= [resource for resource in template['Resources'].values() if resource['Type']=='AWS::DynamoDB::Table']
    assert len(tables)>=1
```

3.6.2 Integration Testing

3.6.2.1 Validation of Items in URL Table

```
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_integ():

    urllist=db.rdymano_data(BetaURLtable)
    ##Getting the URLs in real time using API invoke link
    api_res=requests.get('https://4jd8g9kea3.execute-api.us-east-2.amazonaws.com/prod/')
    reply=json.loads(api_res.text)
    print("res from api", api_res)
    print("The reply=", reply)

    assert len(urllist) == len(reply['body'])
```

3.6.2.2 Test for Latency of API DELETE Method

```
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_realtimedel():
    start =datetime.datetime.now()
    api_put_res=requests.delete('https://4jd8g9kea3.execute-api.us-east-2.amazonaws.com/prod/', data="dummy.com")
    end=datetime.datetime.now()
    dif=end-start
    latency=round(dif.microseconds * 0.000001,6)
    assert latency<0.5 #should be less than 500ms
```

3.6.2.3 Test for Latency of API PUT Method

```
AWS: Add Debug Configuration | AWS: Edit Debug Configuration
def test_realtimeput():
    start =datetime.datetime.now()
    api_put_res=requests.put('https://4jd8g9kea3.execute-api.us-east-2.amazonaws.com/prod/', data="dummy.com")
    end=datetime.datetime.now()
    dif=end-start
    latency=round(dif.microseconds * 0.000001,6)
    assert latency<0.5 #should be less than 500ms
```

3.7 Related Issues

3.7.1 Invoke Permission for Lambda

The backend lambda function needs invocation granted by the API gateway. If you do not specify it then your lambda will not invoke on API event request.

```
apibackendlambda=self.create_dblambda('ApiLambda', './resources','backend_lambda.lambda_handler' ,db_lambda_role,
    environment={'tablename':urlltablename})##, "mytopic":topic.topic_arn})
apibackendlambda.grant_invoke([ aws iam.ServicePrincipal("apigateway.amazonaws.com")])
URLtable.grant_read_write_data(apibackendlambda)
```

3.7.2 Table Not Accessible

We have to grant read-write access for the dynamodb table to the backend Lambda function. Otherwise, it will give an error of permission/access denied.

```
apibackendlambda.grant_invoke([ aws iam.ServicePrincipal("apigateway.amazonaws.com")])
URLtable.grant_read_write_data(apibackendlambda)
URLtable.grant_read_write_data(HWlambda)
```


4 Sprint 4: Frontend Development in ReactJS and Deployment in AWS Amplify

4.1 Overview

In this sprint, I had to create a frontend using ReactJS. The frontend should have two features, show the URLs and to search a URL in the database. In reactJS, we have to use Chakra UI library. And we have to add a pagination for URLs display. In the end, we need OAuth authorization and then the react app is to be deployed at AWS Amplify through S3 bucket. The block diagram of this sprint is shown below:

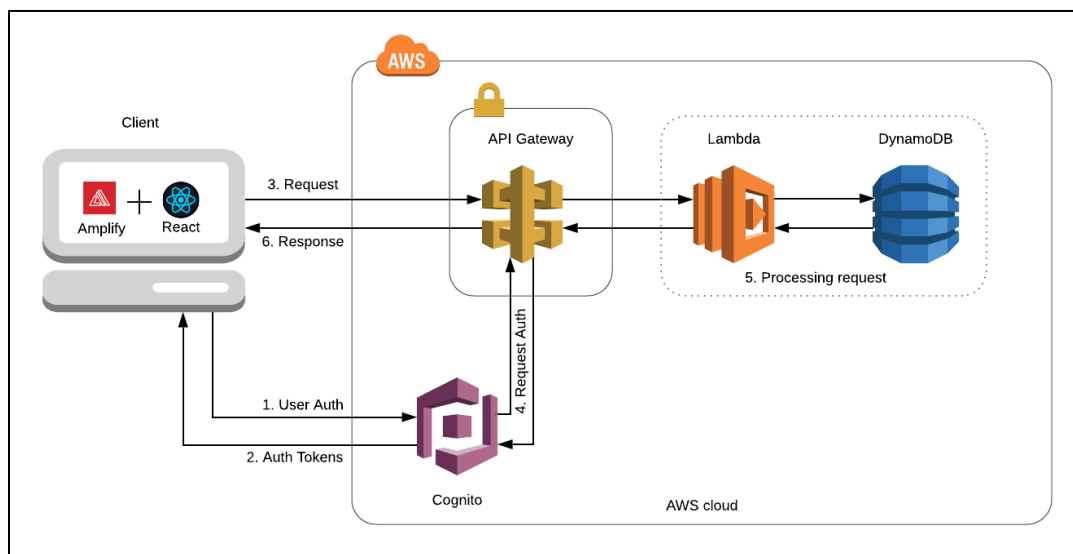


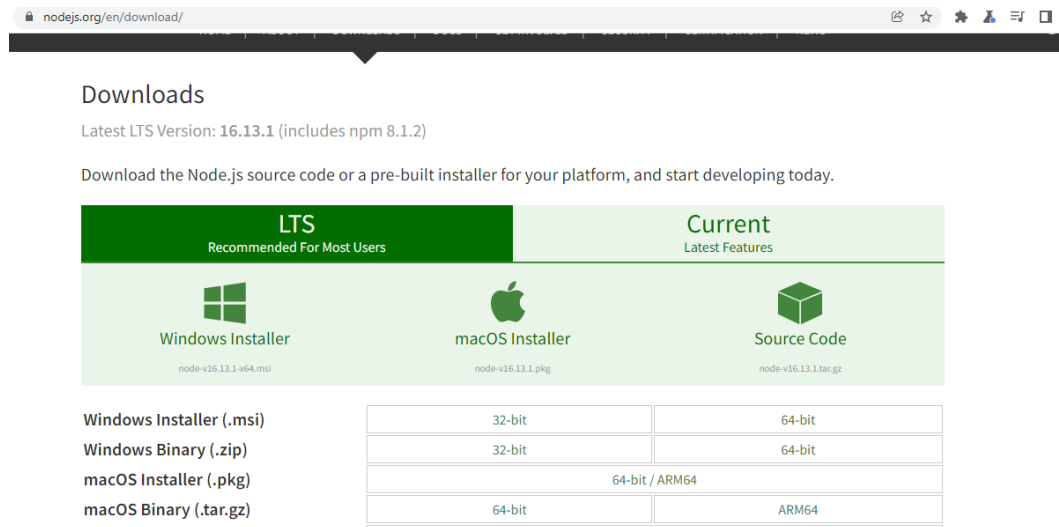
Figure 24 Server less Web Application using ReactJS and AWS Amplify

4.2 Getting Started with ReactJS

To start creating user interface (UI) in reactJS we need to create a react project first and NodeJS is a pre-requisite for this.

4.2.1 Install Node JS

Download the setup that is compliant with your operating system from the website (nodejs.org). A setup file will be downloaded, you have to simply run that set-up file and the setup will be installed within a few minutes.



4.2.2 Installing ReactJS

Now we have to open command prompt and install ReactJS from command line by following commands. Run the commands:

- 1- To Create a raw app of ReactJs

```
npm install -g create-react-app
```

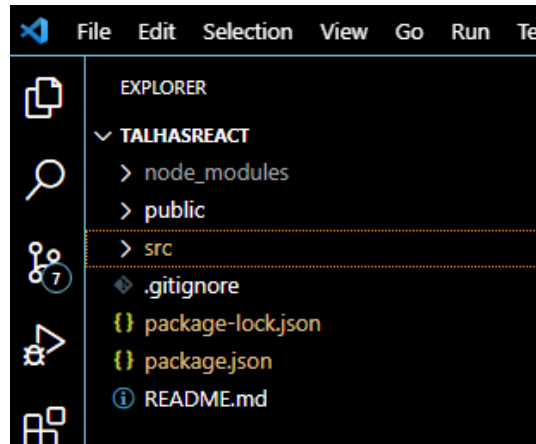
- 2- Run the command to create a ReactJS basic project with complete JS skeleton:

```
npm create-react-app talhas_rjs_app
```

- 3- Go to the directory:

```
cd talhas_rjs_app
```

- 4- Now a ReactJS project folder is created and we can view it in VS code:



- 5- We can view this basic project UI by running the following command in command prompt or VS terminal:

```
npm start
```

The following screen will appear:

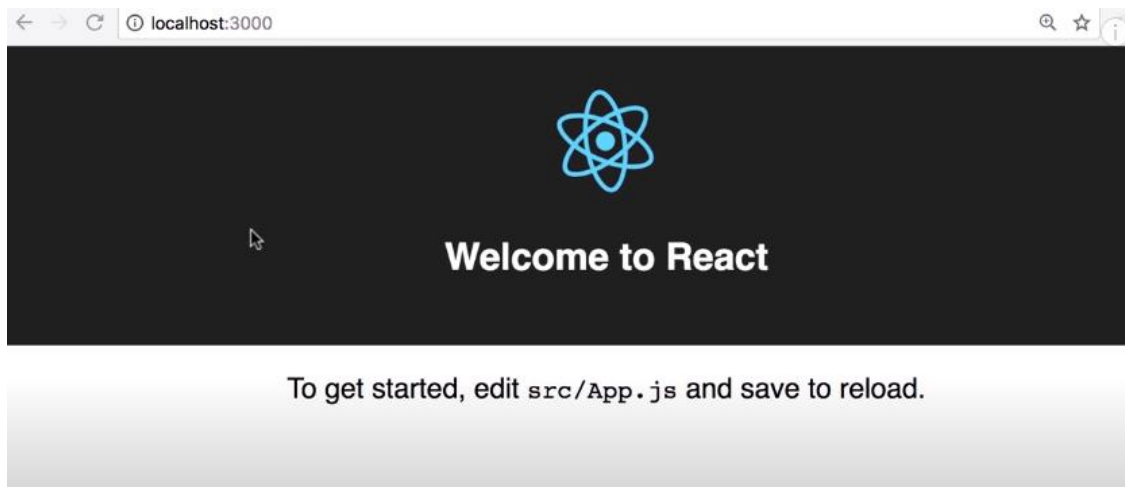


Figure 25 Default UI

4.2.3 Customizing the UI

I have made some changes in index.js, App.js files and src folder. Firstly, I have made some changes in App.js file and I have added some customized components to it as:

```
src > JS App.js > ...
1  import React, { Component } from 'react';
2
3  import './App.css';
4  import Navbar from './components/navbar';
5  import Counter from './components/counter';
6  function App() {
7    return (
8      <React.Fragment>
9        <Navbar />
10       <main className="container" >
11         <Counter />
12       </main>
13     </React.Fragment>
14   )
15 }
```

I have made two more JSX files within src folder, one is navbar file and other is counter file, which contain code for navigation bar and UI buttons respectively.

```
src > components > counter.jsx > ...
1  import React, { Component } from 'react';
2  const element = <h1>Hello! Talhas UI</h1>
3  class Counter extends React.Component
4  {
5    state=
6    {
7      count:1
8    };
9    handleShowurls=
10    {
11
12    };
13
14    styles=
15    {
16      fontSize:30 ,
17      fontWeight: "bold"
18    };
19    render() {
20      return (
21        <div>
22          <span style={this.styles} className="badge badge-primary m-1"> Select the option: </span>
23          <button onClick={this.handleShowurls}> Show Current URLs</button>
24          <button>Enter URL</button>
25          <button > Close </button>
26        </div>
27      );
28    }
29  }
```

Figure 26 Counter.jsx file code

```
src > components > navbar.jsx > default
1  import React, { Component } from 'react';
2  class Navbar extends React.Component {
3    render() {
4      return (
5        <nav class="navbar navbar-light bg-light">
6          <span class="navbar-brand mb-30 h1">User Interface for Web Crawler </span>
7        </nav>);
8      }
9    }
10
```

Figure 27 Navbar.jsx code

The customized UI will look like:

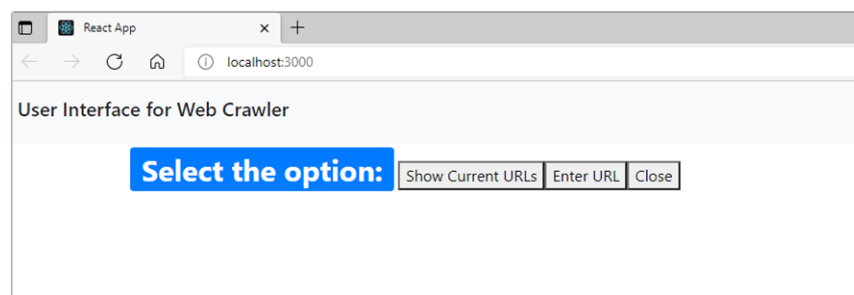


Figure 28 Customized UI

4.3 Design Parameters of Required Frontend

After being done with the dummy UI, now let us customize the UI to meet our requirements. For that, I am supposed to use the chakra UI library. Moreover, I have to make a GET request from my API gateway, so I need some HTTP request library as well.

4.3.1 User Requirements

The list of requirements is as follows:

- A search option to search any URL in the dynamodb database.
- A show button, which should show all the URLs, when clicked.
- The URLs are to be shown in pagination
- An OAuth authentication

4.4 Frontend Design with Chakra UI in ReactJS

4.4.1 Installation of Dependencies

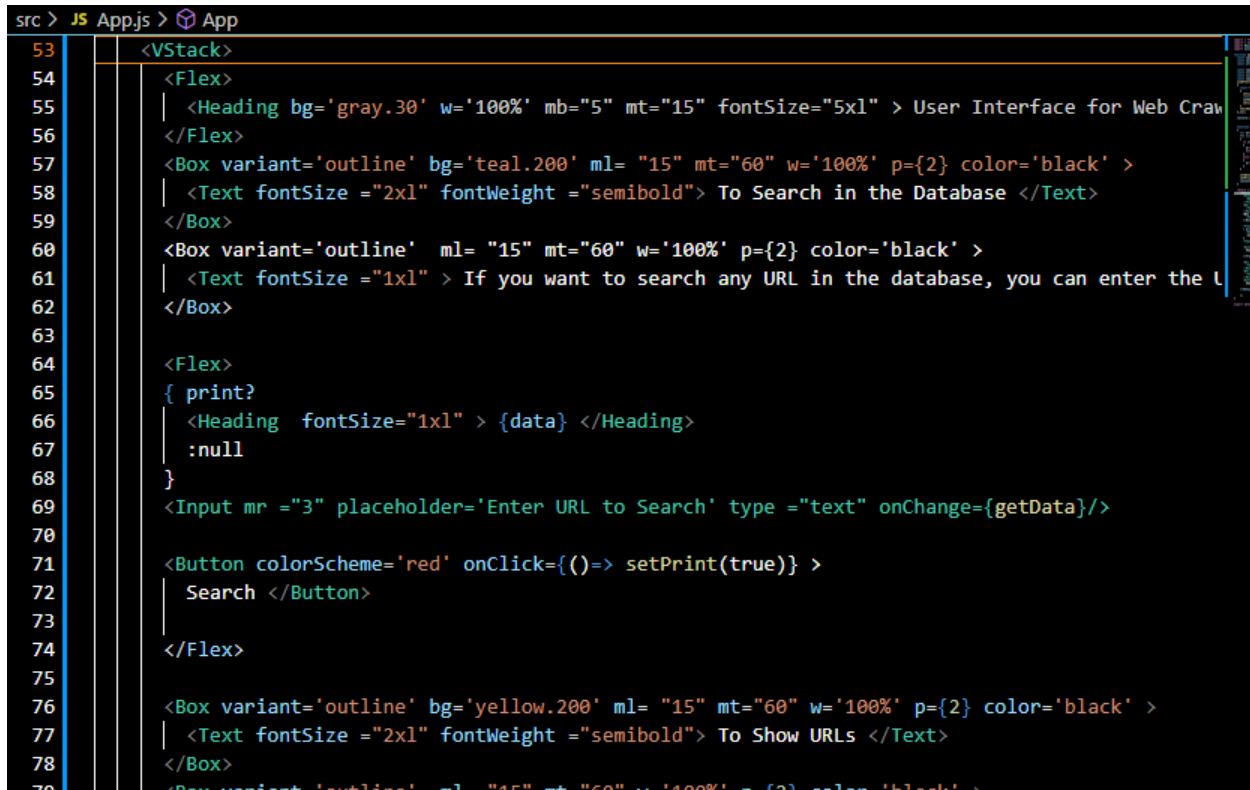
First, install these libraries, which are required:

```
npm install -g chakra
```

```
npm install -g axios
```

4.4.2 UI Design

Modify the file App.js to import the modules and update the boxes and headers to chakra UI, and use axios library to get data from URL.



```
src > JS App.js > App
53 <VStack>
54   <Flex>
55     <Heading bg='gray.300' w='100%' mb='5' mt='15' fontSize='5xl' > User Interface for Web Craw
56   </Flex>
57   <Box variant='outline' bg='teal.200' ml='15' mt='60' w='100%' p={2} color='black' >
58     <Text fontSize='2xl' fontWeight='semibold'> To Search in the Database </Text>
59   </Box>
60   <Box variant='outline' ml='15' mt='60' w='100%' p={2} color='black' >
61     <Text fontSize='1xl' > If you want to search any URL in the database, you can enter the U
62   </Box>
63
64   <Flex>
65     { print?
66     <Heading fontSize='1xl' > {data} </Heading>
67     :null
68     }
69     <Input mr='3' placeholder='Enter URL to Search' type='text' onChange={getData}/>
70
71     <Button colorScheme='red' onClick={()=> setPrint(true)} >
72       Search </Button>
73
74   </Flex>
75
76   <Box variant='outline' bg='yellow.200' ml='15' mt='60' w='100%' p={2} color='black' >
77     <Text fontSize='2xl' fontWeight='semibold'> To Show URLs </Text>
78   </Box>
79   <Box variant='outline' ml='15' mt='60' w='100%' p={2} color='black' >
```

4.4.3 API Integration

To get data from a URL and to check on the search buttock we have used useState from reactJS. Using Axios library, we can forward a request to our API and the response can be stored in some local or global variable. The complete code is as below:

```
function App() {
  const [data, setData]=useState(null);
  const [print, setPrint]= useState(false);
  const urls=["talha", "ali", "umar"];

  const [posts, setPosts]= useState([]);
  const [loading, setLoading]= useState(false);
  const [currentPage, setCurrentPage]= useState(1);
  const [postsPerPage, setPostsPerPage]= useState(10);
  useEffect(() =>
  {
    const fetchPosts= async() =>
    {
      setLoading(true);
      const res=await axios({method: 'get',url:"https://jsonplaceholder.typicode.com/posts"});
      setPosts(res.data);
      setLoading(false);
    }
  },
  [postsPerPage]);
}
```

4.4.4 Promptness upon Query from Frontend

I needed to write a function `getData` so that I may check if the query URL exists in our database or not.

```
src > JS App.js > App > getData
23 |     setLoading(false);
24 |   }
25 |   fetchPosts();
26 | }, []);
27 |
28 | console.log(posts);
29 |
30 |
31 | function getData(val)
32 | {
33 |   var d="";
34 |   for (const el of urls)
35 |   {
36 |     if (val.target.value ==el)
37 |     {
38 |       d="yes";
39 |       break;
40 |     }
41 |     else
42 |     {
43 |       d="no";
44 |     }
45 |   }

```


4.5 React App Deployment Using AWS-Amplify

4.5.1 Introduction

AWS Amplify is a set of purpose-built tools and features that lets frontend web and mobile developers quickly and easily build full-stack applications on AWS, with the flexibility to leverage the breadth of AWS services as your use cases evolve. With Amplify, you can configure a web or mobile app backend, connect your app in minutes, visually build a web frontend UI, and easily manage app content outside the AWS console.

I have created an aws amplify app in my project stack and then adding a stage or my reactjs app to be deployed and hosted.

```
amplify_app = amplify.App(self, 'TalhasNewAmpApp',role=db_lambda_role)
branch = amplify_app.add_branch('dev')
```

The amplify app will look like this:

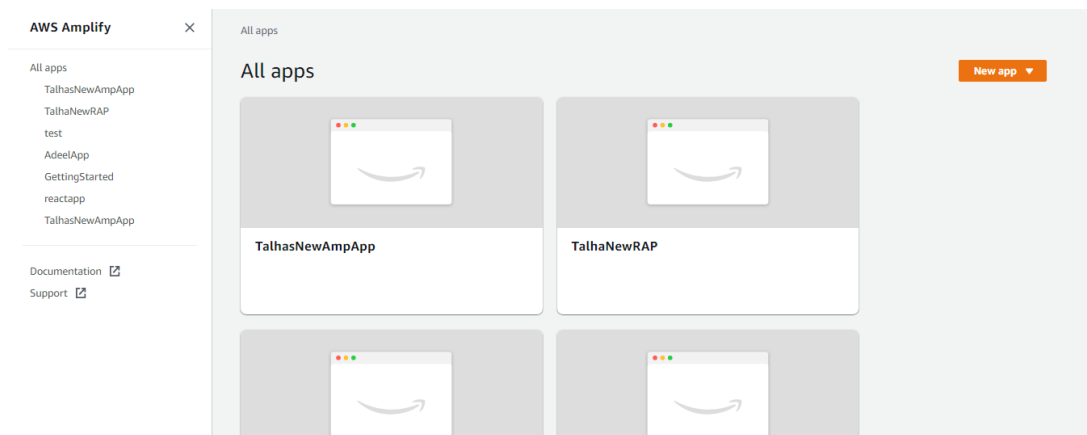


Figure 29 AWS Amplify application

Now we have to zip the build folder that was created from 'npm run build' command in the VS code terminal. Add that zip file to the S3 bucket and then go to AWS amplify and configure source from amazon S3 bucket.

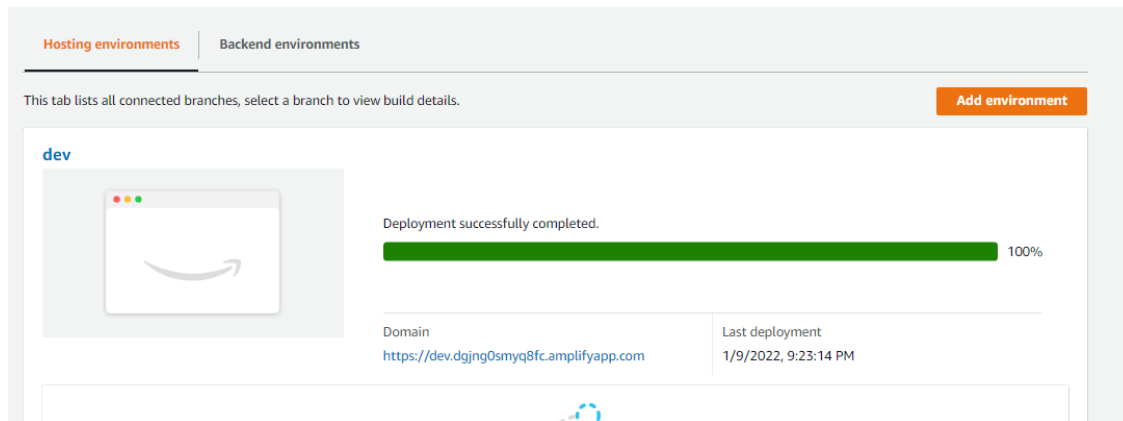


Figure 30 Deployment of Web in dev branch

4.5.2 OAuth Configuration

After completely deploying my app, I moved towards the Privacy part of online domain as this app can access all the files in my database I want a certain verified number of people to access it and see the venture through the database. I implemented a sign in and sign up method on my app. After completely deploying my app, I moved towards the Privacy part of online domain as this app can access all the files in my database I want a certain verified number of people to access it and see the venture through the database. I implemented a sign in and sign up method on my app.

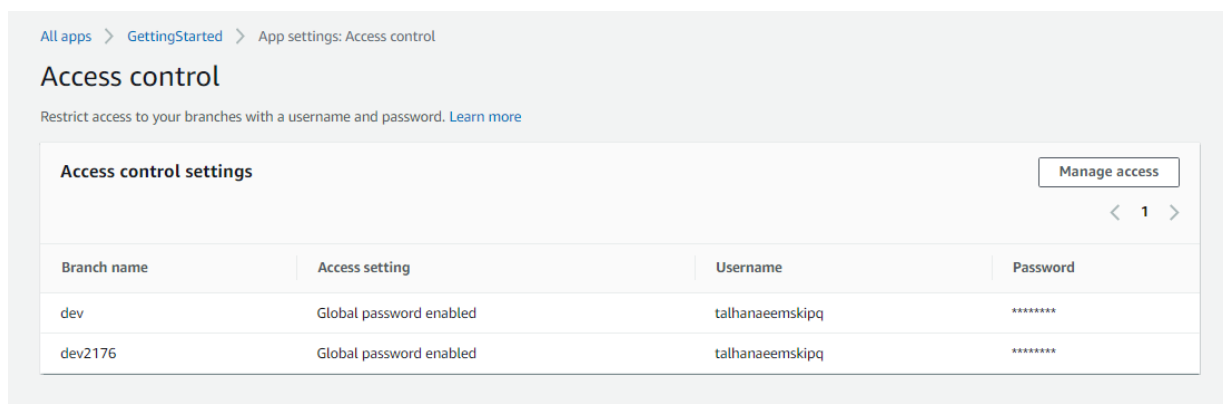
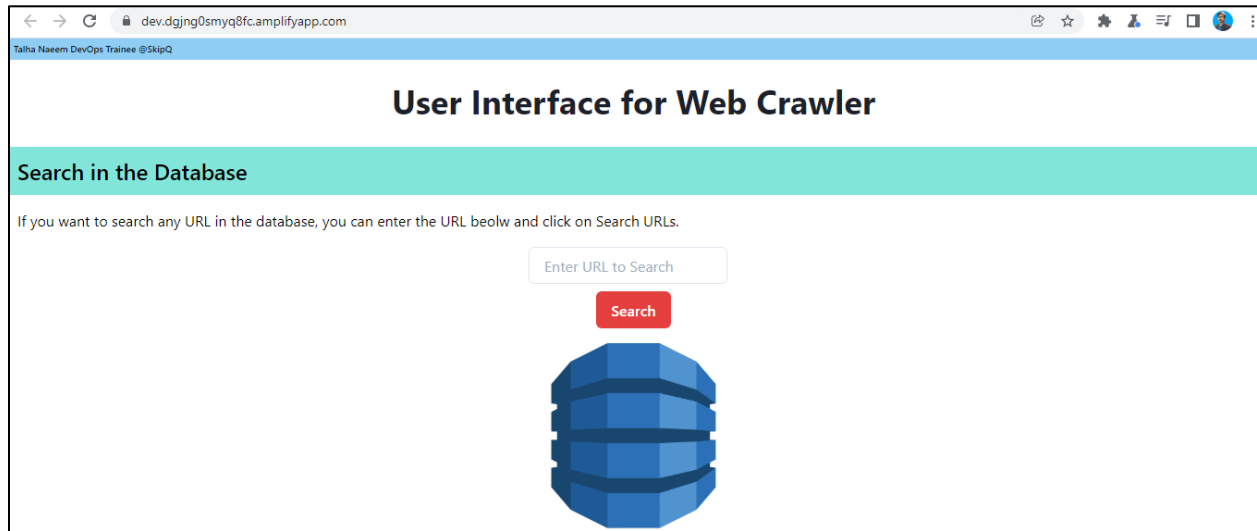


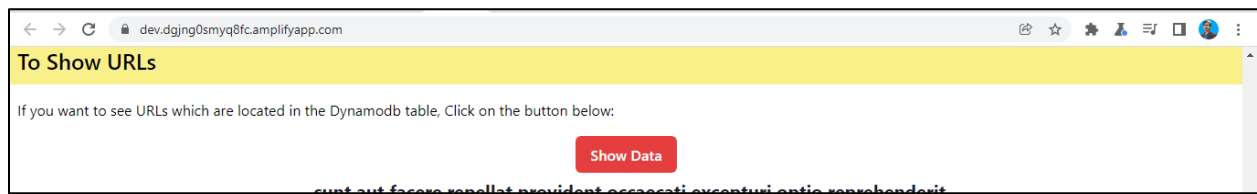
Figure 31 OAuth Configuration

4.6 The Hosted Web

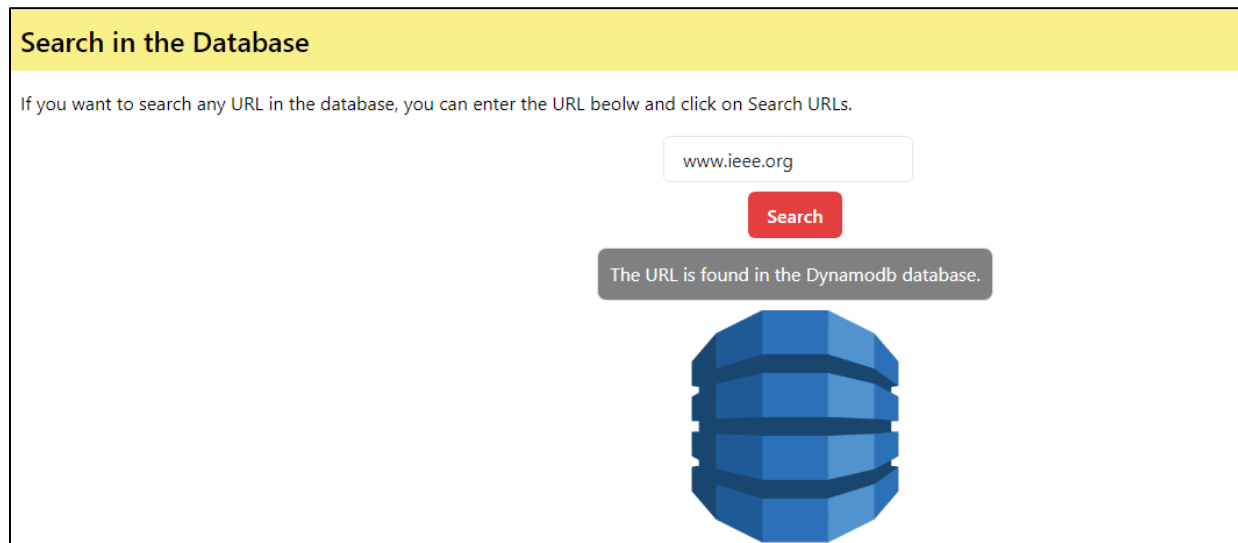
The final UI will look like this:



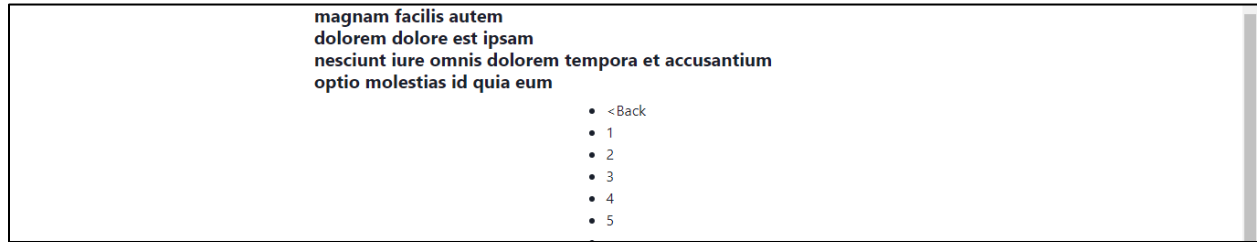
4.6.1 Continued



4.6.2 Data Search



4.6.3 Pagination.



4.6.4 URLs Pagination

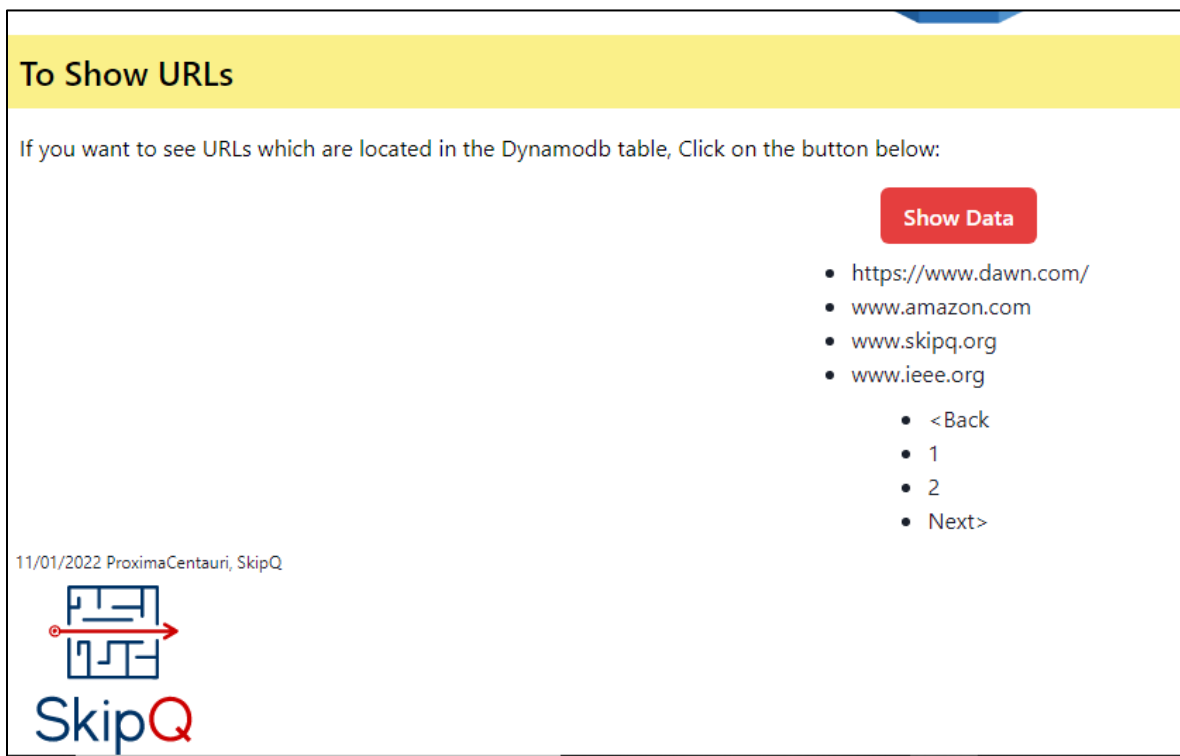


Figure 32 Final web frontend

4.7 Related Issues

4.7.1 No Space Left in Environment

My EC2 instance started giving me the error of low space. I was trying to bootstrap for sprint4, while there was no space left. Therefore, I decided to create new environment with 2GB.

```
from .jsii import *
File "/home/ubuntu/.local/lib/python3.6/site-packages/aws_cdk/aws_events_targets/_jsii/__init__.py", line 13, in <module>
import aws_cdk.aws_codebuild._jsii
File "/home/ubuntu/.local/lib/python3.6/site-packages/aws_cdk/aws_codebuild/__init__.py", line 784, in <module>
from .jsii import *
File "/home/ubuntu/.local/lib/python3.6/site-packages/aws_cdk/aws_codebuild/_jsii/__init__.py", line 33, in <module>
"aws-codebuild@1.135.0.jsii.tgz",
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/runtime.py", line 43, in load
_kernel.load(assembly.name, assembly.version, os.fspath(assembly_path))
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_kernel/__init__.py", line 269, in load
self.provider.load(LoadRequest(name=name, version=version, tarball=tarball))
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_kernel/providers/process.py", line 338, in load
return self._process.send(request, LoadResponse)
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_kernel/providers/process.py", line 326, in send
raise JSIIError(resp.error) from JavaScriptError(resp.stack)
jsii.errors.JSIIError: ENOSPC: no space left on device, write
Subprocess exited with error 1
talhanaemskipq:~/environment/ProximaCentauri/talha/sprint4 (main) $
```

Figure 33 Error on bootstrap: No space left

4.7.1.1 Environment Setup

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

- 1- First, log in to AWS amazon and create a virtual machine.
- 2- 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

alias python='/usr/bin/python3' (press ESC on keyboard)

:w! (press Enter on keyboard)

:q! (press Enter on keyboard)
```

- 3- 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
```

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

```
mkdir talha_sprint4
```

```
cd talha_sprint4
```

5- Create CDK project in python language

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

6- Install all requirements for the 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
```

4.7.2 Access Denied While API Integration

When I tried to get data from my API using AXIO library, it gave me this error:

```

✖ Access to XMLHttpRequest at 'https://4jd8g9kea3.execute-api.us-east-2.amazonaws.com/prod' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
✖ ▶ GET https://4jd8g9kea3.execute-api.us-east-2.amazonaws.com/prod net::ERR_FAILED 200 xhr.js:199
✖ ▶ Uncaught (in promise) Error: Network Error at createError (createError.js:7)
    at XMLHttpRequest.handleError (xhr.js:105)

```

Adding CORS Extension to browser can solve this error, so go to extensions of your browser and add CORS EXTENSION. This can be done by following easy steps mentioned in the references as CORS extension.

5 Sprint5: API Functional and Security Testing in Docker-Compose

5.1 Overview

In this sprint, we have to build API test clients using Docker compose. We have to create an image to our built test clients and store in ECR, from where it is deployed in AWS ECS or Fargate using CDK. We have to build and push API test dockers through CodePipeline. We will push API test results in CloudWatch and some alarms or notifications will be set up on the test results metrics.

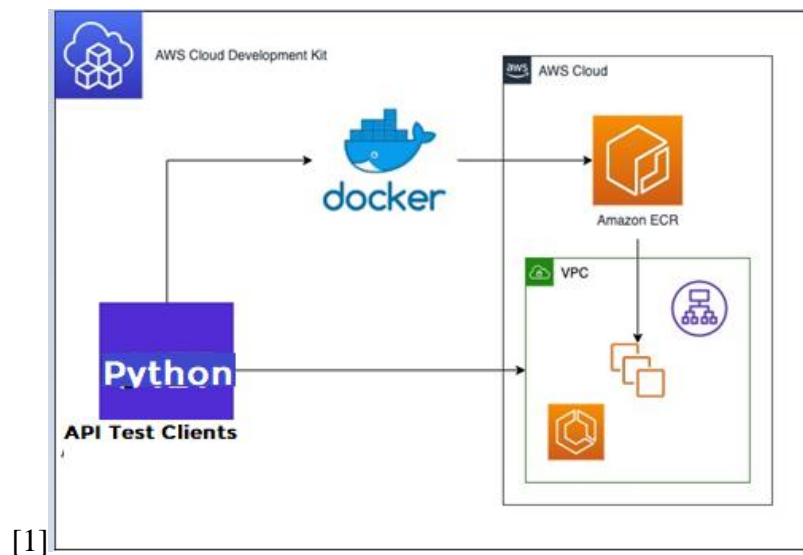


Figure 34 Block diagram for the sprint

5.2 Docker-Compose in Docker with Ports

5.2.1 Docker Installation

You need to have Docker platform installed at your computer, so I installed docker for windows from the [website](#). Just to validate the version, run ‘docker --version’ in the command prompt as,

```
C:\Users\WALI>docker --version
Docker version 20.10.12, build e91ed57

C:\Users\WALI>
```

5.2.2 Getting Started with Docker Compose

Within Docker's platform, docker-compose is a tool that is used to define and share multi-container applications. With Compose, we create a YAML file to define the services and with a single command, can set the environment required for the involved applications. So, we will clone and repo to start with Docker compose for a python application and a PHP application.

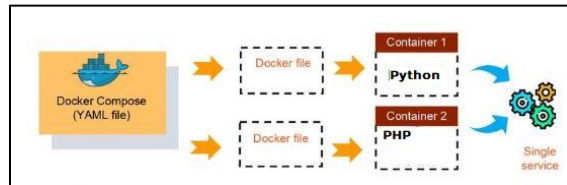


Figure 35 Docker-compose structure

- 1- Run the command in command prompt to get the sources files for docker-compose,

```
git clone https://github.com/talha2021skipq/tutorials.git
```

- 2- Go to the directory,

```
cd ./docker/02-docker-compose
```

- 3- Make sure to have docker-compose YAML file in this directory. And run the docker-compose,

```
docker-compose up
```

- 4- Now the respective containers have been built automatically and we can see the output at ports on localhost as specified in the docker compose file.

```
version: '3'
services:
  product-service:
    build: ./product
    volumes:
      - ./product:/usr/src/app
    ports:
      - 5001:80
    website:
    image: php:apache
    volumes:
      - ./website:/var/www/html
    ports:
      - 5000:80
    depends_on:
      - product-service
```


5.3 Create and Build Docker Image for pyresttest

5.3.1 Dockerfile

While creating a Docker image, we need to set up the environment first and then collect the required resources in it. We will use python library, so we need a python image, for that I have chosen a python image from alpine available at DockerHub. Within that basic alpine image, I have installed the required libraries like pycurl and pyresttest. In the end, I have created a work directory and copied the content in that directory. Our DockerFile code will look like:

```
1 # Importing python image from the default available images
2 FROM python:2-alpine
3 # Setting up the requirments or libraries like pycurl
4 ENV PYCURL_SSL_LIBRARY=openssl
5 RUN apk add --no-cache --update openssl curl \
6     && apk add --no-cache --update --virtual .build-deps build-base python-dev curl-dev \
7     && pip install jmespath jsonschema pyresttest \
8     && apk del .build-deps
9 #Setting the workdirectory and copying the content
10 WORKDIR /tests
11 COPY . /tests
12 ENTRYPOINT ["pyresttest"]
```

Now we can build this image by using CMD. Run the following command to build the image, while making sure to have the Docker desktop open.

```
docker build -t talhanew .
```

This docker-image could be seen from Docker application as shown below:

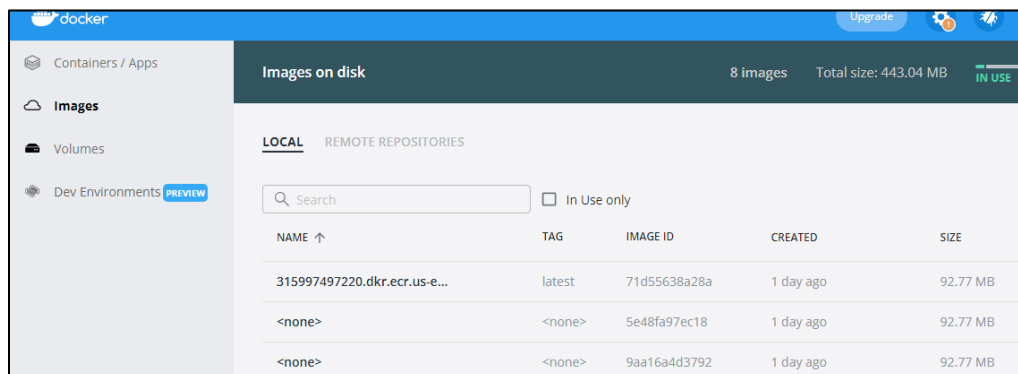


Figure 36 Docker Image


```
- output_format: csv
- metrics:
  - total_time
  - size_download
  - total_time: mean
```

5.3.2.2.1 Result of PUT Test

We can perform the test using the built image with docker-run command,

```
docker run --rm talhanew https://n0q8c9iur5.execute-api.us-east-2.amazonaws.com/prod/ talhaputest.yaml
```

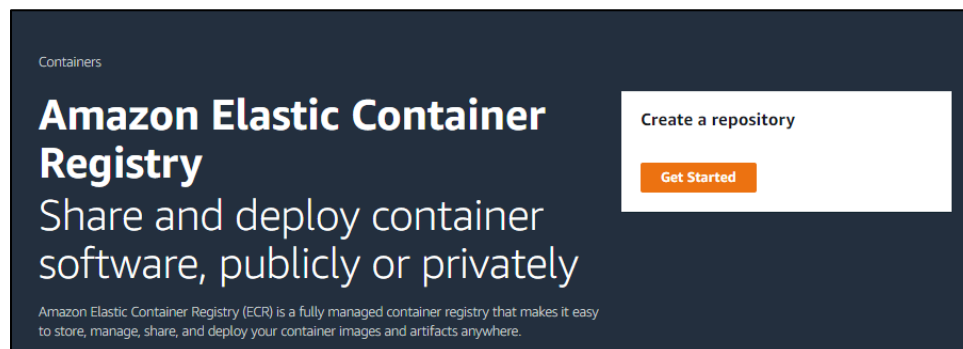
```
{"failures": 0, "aggregates": [{"total_time": "mean", 0.9898874499999998}], "group": "Default", "results": {"total_time": [0.979398, 0.964091, 0.918222, 0.972607, 1.141772, 1.328539, 0.899887, 1.08287, 0.98907, 0.868306, 1.037257, 0.895978, 1.17802, 0.97024, 0.827974, 0.957836, 0.962061, 0.90019, 1.003511, 0.91992], "size_download": [15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0]}, "name": "Basic put"}
```

5.4 Push Image to AWS ECR

Following are the easy steps to push the image to AWS ECR:

1- Create Docker image repo at AWS Elastic Container Registry (ECR).

- Open ECR from the console and click on Get Started:



- Name the repository as the same name of your docker image:

Amazon ECR > Repositories > Create repository

Create repository

General settings

Visibility settings [Info](#)
Choose the visibility setting for the repository.

☒ **Private**
Access is managed by IAM and repository policy permissions.

☐ **Public**
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.

315997497220.dkr.ecr.us-east-2.amazonaws.com/

- c. Click on Create Repository. The repo has been created as shown below:

Amazon ECR > Repositories

Private | Public

Private repositories (1)

	Repository name ▲	URI	Created at ▼	Tag immutability	Scan frequency	Encryption type	Pull through cache
<input type="radio"/>	talhanew	315997497220.dkr.ecr.us-east-2.amazonaws.com/talhanew	January 18, 2022, 20:24:45 (UTC+05)	Disabled	Manual	AES-256	Inactive

2- Installation of AWS CLI in local computer.

- a. Install the AWS CLI by running the following command in command prompt.

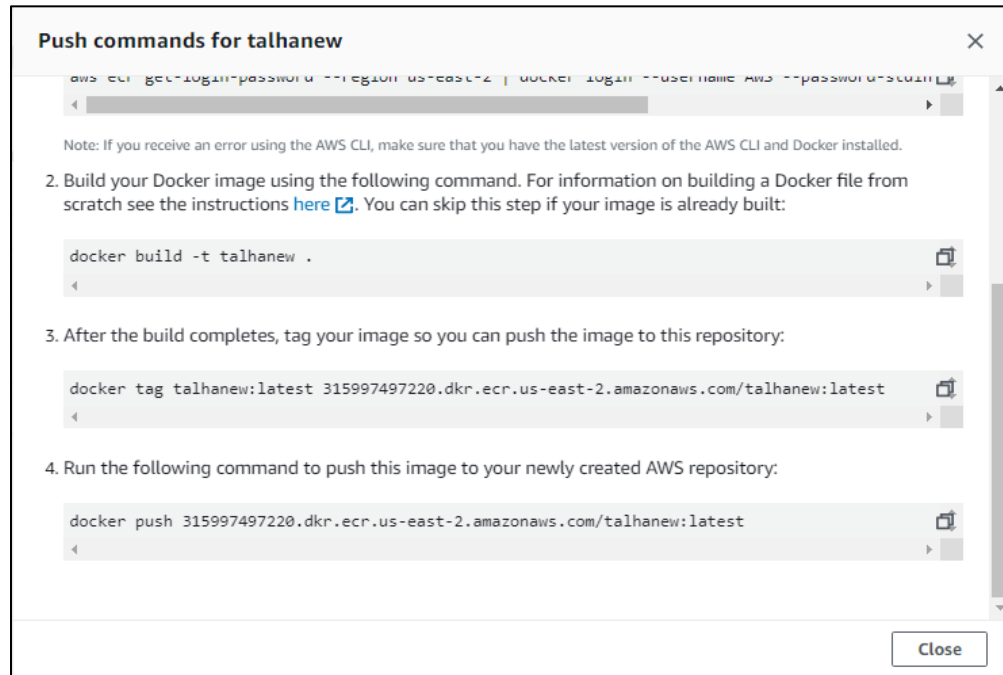
```
msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

3- Configuration of AWS Credentials locally.

- a. Run 'aws configure' in command prompt, it will ask for ID and access key, after authentication the login will succeed.

4- Pushing the Docker Image to AWS ECR.

- a. View push image commands for your repository and run those commands one by one:



5.5 Pull Image from ECR

We can pull a docker image from AWS ECR repository using some methods of aws-ecr module,

```
# Pulling the docker image from AWS ECR Repo
repo = ecr.Repository.from_repository_name(self, "TalhasECR", "talhanew")
image=ecs.EcrImage(repo, "latest")
```

5.6 Deploy the Image on ECS

As we have built the image and now, we have to create container for it and then deploy that container. Therefore, here are the intermediary steps to deploy the image on AWS.

5.6.1 Creating Cluster on ECS

To deploy the image, we first need to have some EC2 instance running, so we define a cluster on ECS with Virtual Private Cloud.

```
# Creating an ECS cluster with EC2 VPC
vpc = ec2.Vpc.from_lookup(self, "Talhasvpc", is_default=True)
cluster = ecs.Cluster(self, "TalhasCluster", vpc=vpc)

# Adding capacity to the Cluster (t2 micro specs)
cluster.add_capacity("TalhasClustercapacity",
    instance_type=ec2.InstanceType("t2.micro"))
```

Here we can see the cluster in AWS ECS clusters section through our console, it will look like:

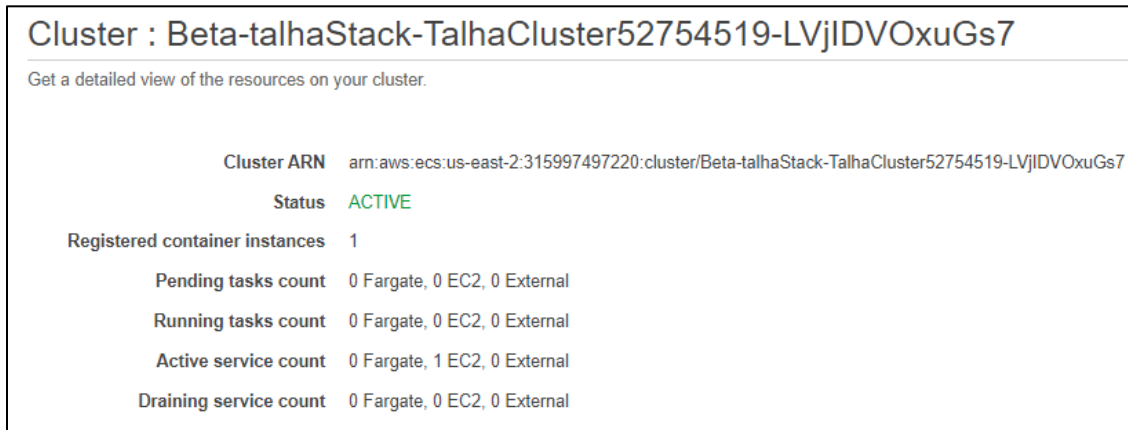


Figure 37 Cluster on ecs

5.6.2 Define Task on ECS

Now we have to create EC2 task-definition for our ECS service.

```
# Creating EC2 Task Definition for ECS ,and giving the command to run pyresttest
task_definition = ecs.Ec2TaskDefinition(self, "TalhasTaskDef")
task_definition.add_container("DefaultContainer",
    image=image,
    command=["docker run --rm 315997497220.dkr.ecr.us-east-2.amazonaws.com/talhanew https://r
    memory_limit_mib=512
    )
```

Here in the below figure, we can see the created task-definition and the status is RUNNING.

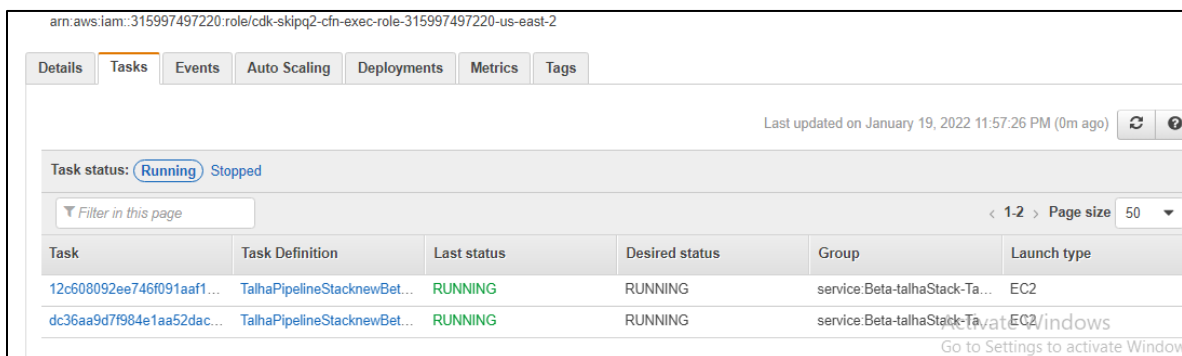


Figure 38 Running task on the container

Container Name	Image	CPU Units	GPU
DefaultContainer	31599749722...	0	

Details

Command

```
[ "docker run --rm 315997497220.dkr.ecr.us-east-2.amazonaws.com/talhanew
https://n0q8c9iur5.execute-api.us-east-2.amazonaws.com/prod/
talhagettest.yaml" ]
```

Figure 39 Task along with commands on the container

5.6.3 Instantiating ECS with Task Definition

Now we have to finally create an EC2 instance, and pass the defined task along with the defined cluster to run in.

```
# Instantiating an Amazon ECS Service with the task definition provided
ecs_service = ecs.Ec2Service(self, "TalhaService",
    cluster=cluster,desired_count=5,enable_execute_command=True,
    task_definition=task_definition )
```

We can view the instance details from ECS on console,

Instance: i-062807033059c5d44 (TalhaPipelineStacknew/Beta/talhaStack/TalhasCluster/TalhasClusterCapacity)		
▼ instance summary info		
Instance ID i-062807033059c5d44 (TalhaPipelineStacknew/Beta/talhaStack/TalhasCluster/TalhasClusterCapacity)	Public IPv4 address 3.16.66.230 open address	Private IPv4 addresses 172.31.1.21
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-16-66-230.us-east-2.compute.amazonaws.com open address

Figure 40 Running ECS EC2 instance

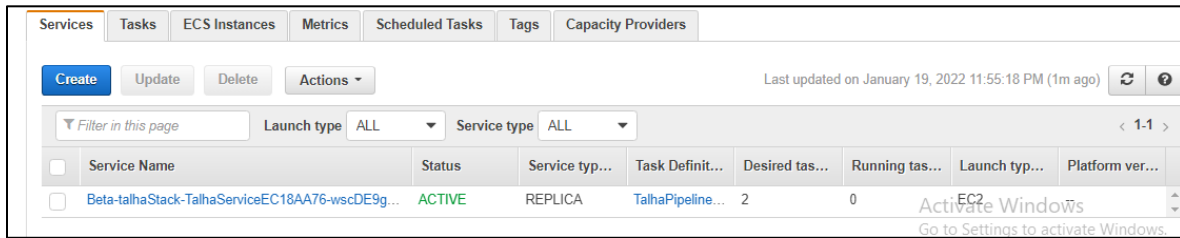


Figure 41 EC2 Service Active

5.7 Related Issues

5.7.1 Unexpected Keyword Argument

```
__init__
275     task_definition.add_container("DefaultContainer",
276 TypeError: add_container() got an unexpected keyword argument 'memory_limit_mi_b'
277 Subprocess exited with error 1
278
279 [Container] 2022/01/18 15:30:16 Command did not exit successfully cdk synth exit status 1
280 [Container] 2022/01/18 15:30:16 Phase complete: BUILD State: FAILED
281 [Container] 2022/01/18 15:30:16 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
cdk synth. Reason: exit status 1
282 [Container] 2022/01/18 15:30:16 Entering phase POST_BUILD
```

5.7.1.1 Reason and Solution

I was using the keyword argument as 'memory_limit_mi_b', as it was in an example in the documentation. Then I checked from the method's parameters and corrected it as 'memory_limit_mib'.

5.7.2 Resource Creation Timeout

While creating the ECS service and the cluster within the pipeline framework, I got a lot of errors, some are listed below:

- 1- Resource Creation Failed: in the deploy step of Beta stage.
- 2- VPC Limit Exceeded: in the beta stage. However, its solution was to avoid using VPC class method.
- 3- Resource Deletion Failed: When I tried to rename the cluster, instance, and task-definition, the pipeline took 3 hours to run and gave an error of resourced deletion failure due to timeout.

5.7.2.1 Reason

The reason is overloading on the AWS account, as the new cohort was also started and they started creating their stacks, so there was a hell of a load on the account. That is why there was a breakdown in resource creation or update.

6 References

Pytest References:

1. <https://docs.pytest.org/en/latest/explanation/goodpractices.html#test-package-name>
2. <https://stackoverflow.com/questions/41748464/pytest-cannot-import-module-while-python-can>
3. <https://docs.pytest.org/en/6.2.x/getting-started.html#create-your-first-test>

AWS Documentation:

- 1- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_cloudwatch/Metric.html
- 2- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_codedeploy/LambdaDeploymentGroup.html
- 3- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_codedeploy/LambdaDeploymentConfig.html

Shift Traffic for AWS Lambda:

- 1- <https://docs.aws.amazon.com/codestar/latest/userguide/how-to-modify-serverless-project.html>
- 2- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_lambda/Alias.html

Lambda DeploymentGroup and Config:

- 1- <https://docs.aws.amazon.com/codestar/latest/userguide/how-to-modify-serverless-project.html>
- 2- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_codedeploy/LambdaDeploymentConfig.html

Dynamodb Boto3 Resource:

- 1- <https://dynobase.dev/dynamodb-python-with-boto3/#:~:text=To%20get%20all%20items%20from,the%20results%20in%20a%20loop.>
- 2- <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.03.html#GettingStarted.Python.03.06>

API Gateway:

- 1- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_apigateway/README.html#metrics
- 2- <https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html>
- 3- <https://www.geeksforgeeks.org/put-method-python-requests/>
- 4- <https://pythonexamples.org/python-requests-http-put/#4>

React JS:

- 1- <https://youtu.be/Ke90Tje7VS0>
- 2- <https://medium.com/front-end-weekly/a-beginners-guide-building-front-end-applications-with-react-1f0d3e75c0a7>
- 3- <https://getbootstrap.com/docs/4.1/components/navbar/>

Node JS Setup:

- 1- <https://nodejs.org/en/download/>

AWS Amplify:

- 1- <https://docs.aws.amazon.com/amplify/latest/userguide/manual-deploys.html>
- 2- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_amplify.html
- 3- <https://medium.com/zenofai/serverless-web-application-architecture-using-react-with-amplify-part1-5b4d89f384f7>

CORS Extension:

- 1- <https://expressjs.com/en/resources/middleware/cors.html>

Push/Pull Image from ECR:

- 1- <https://medium.com/codex/push-docker-image-to-aws-ecr-e9718df8a729>

- 2- <https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-pull-ecr-image.html>

AWS Resources EC2 and ECS:

- 1- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_ecs/Ec2Service.html
- 2- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_ecs/Ec2TaskDefinition.html
- 3- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_ecs/README.html
- 4- https://docs.aws.amazon.com/cdk/api/v1/python/aws_cdk.aws_ec2.html
- 5- <https://www.youtube.com/watch?v=zs3tyVgiBQQ&t=786s>

Deploy Image on ECS:

- 1- <https://blog.clairvoyantsoft.com/deploy-and-run-docker-images-on-aws-ecs-85a17a073281>
- 2- <https://www.youtube.com/watch?v=zs3tyVgiBQQ&t=786s>

Pyresttest and Syntribos:

- 1- <https://github.com/thoom/pyresttest-docker>
- 2- <https://github.com/svanoort/pyresttest>
- 3- <https://github.com/openstack-archive/syntribos>
- 4- <https://gist.github.com/rahulunair/1f2f1fc9962ef4530ade69f63fc82398>