Scrum Documentation

Production Grade Programming

Proxima Centauri

By: Talha Naeem
DevOps trainee, skipQ cohort-II
Dec-Jan 2021

In sprint1, training started from very scratch including learning the concepts of cloud computing, DevOps, and AWS. We created a web health monitoring system using cloudwatch services to publish web health metrics and to raise an alarm beyond a specific threshold. The alarms were sent to the Dynamodb lambda function using an SNS subscription. Finally, the alarms were written into the 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.

Table of Contents

1 Sprint1

1.1 Sprint1 Task1

1.1.1 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, that calls its handler with the input parameters; events, and context from the hello_world_lambda file.

1.1.1.1 Code for Hello World Lambda

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

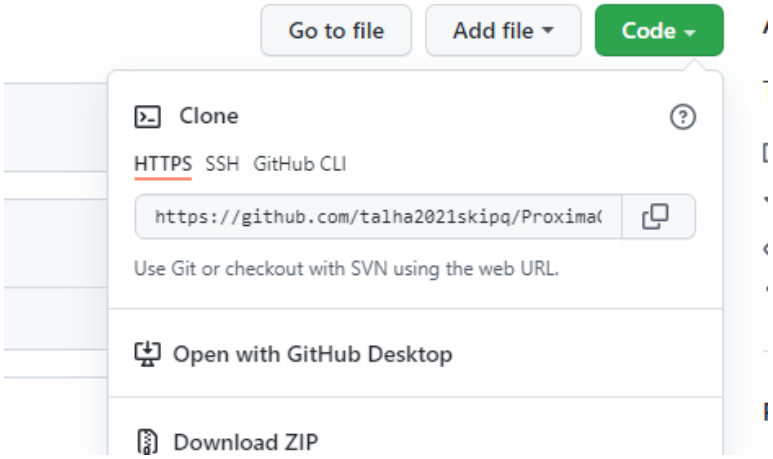
1.1.2 Related Issues

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. And I got success in synthesizing my project by CDK synth and then deploying it using CDK deploy.

1.1.3 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.

1.1.4 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?

1.2 Sprint1 Task2

1.2.1 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. And then those metrics will be sent to a boto3 client for cloud watch to publish them.

1.2.2 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

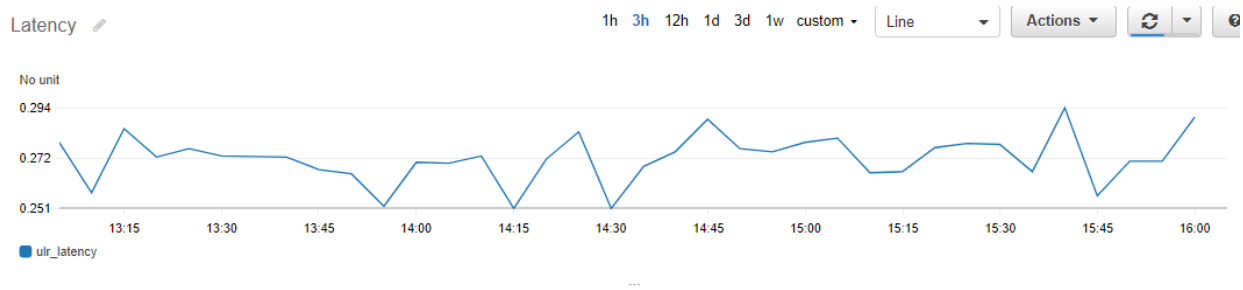


Figure 1 Latency graph on real-time data

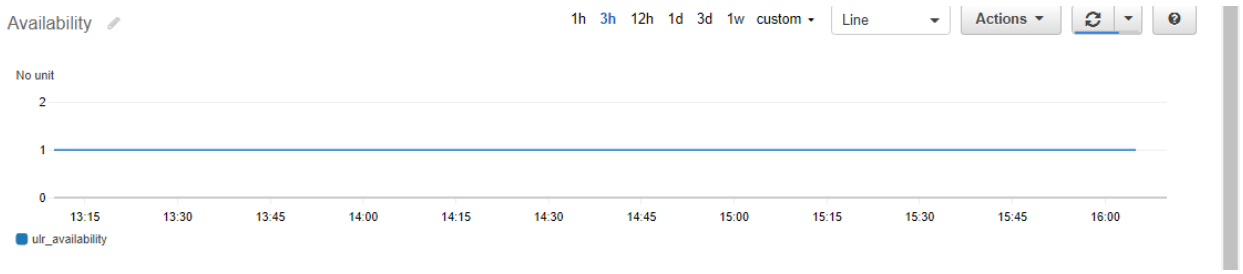


Figure 2 Availability graph on real-time data

1.3 Sprint1 Task3

1.3.1 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:

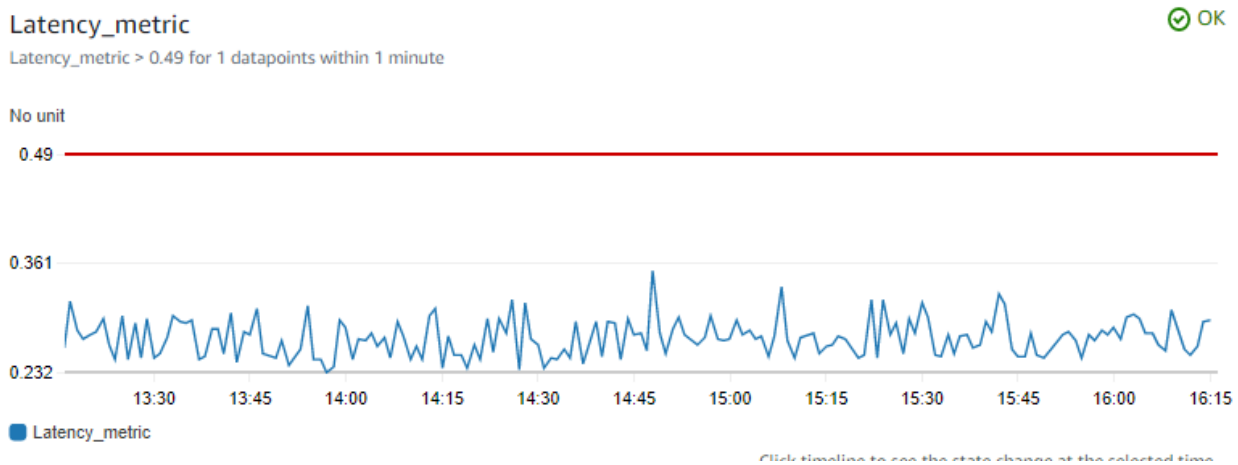


Figure 3 Latency alarm at 0.49

1.3.2 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))
```

1.3.3 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

1.4 Sprint1 Task4

1.4.1 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.

1.4.2 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)
```

1.4.3 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))
```

1.4.4 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.

1.4.4.1 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 path to avoid here in the following:

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

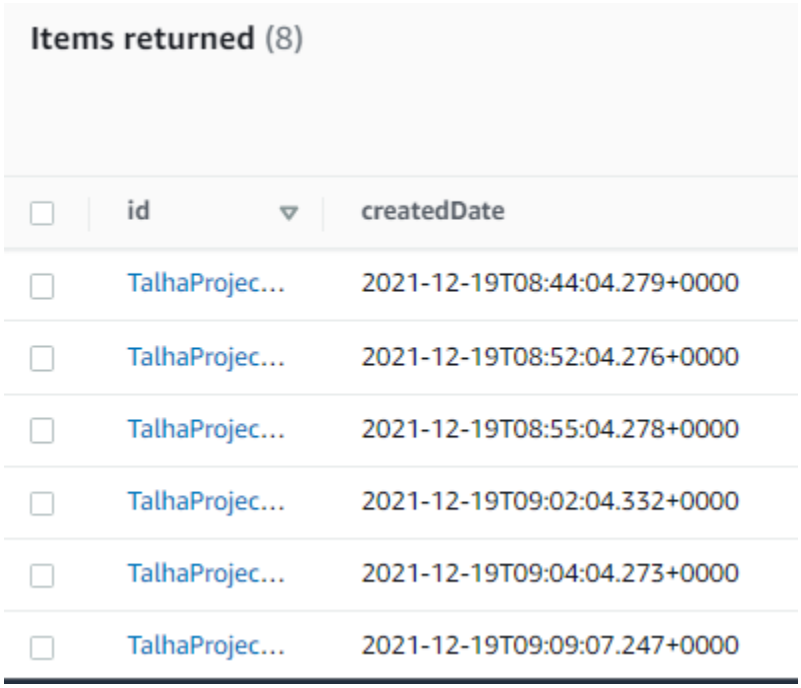
X

1.4.5 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)

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

Figure 4 Alarms table in dynamodb

1.5 Sprint1 Task5

1.5.1 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:

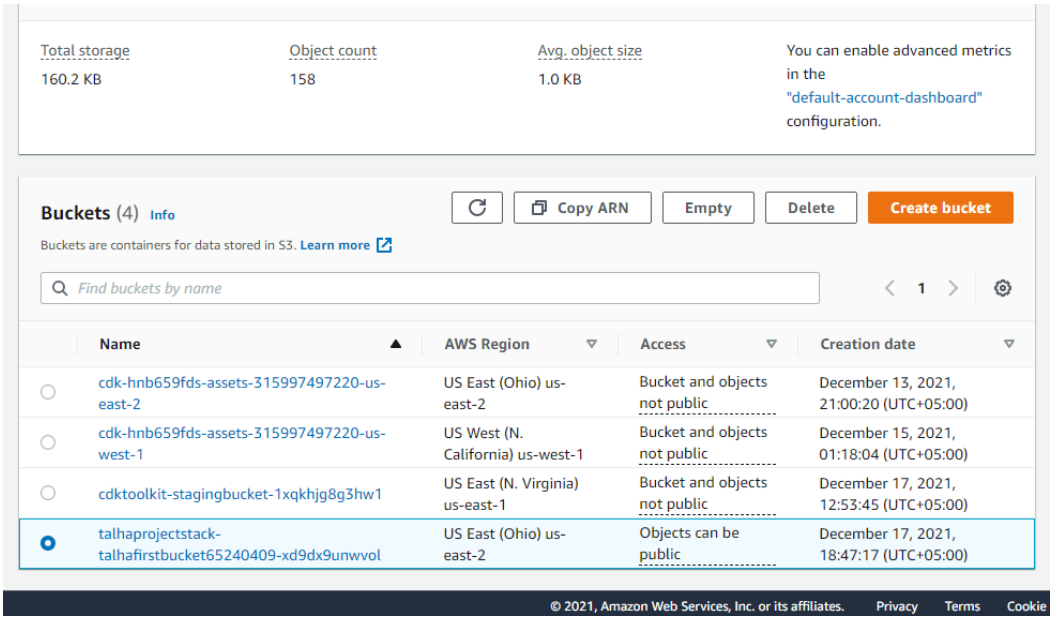
- 

Figure 5 S3 bucket

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

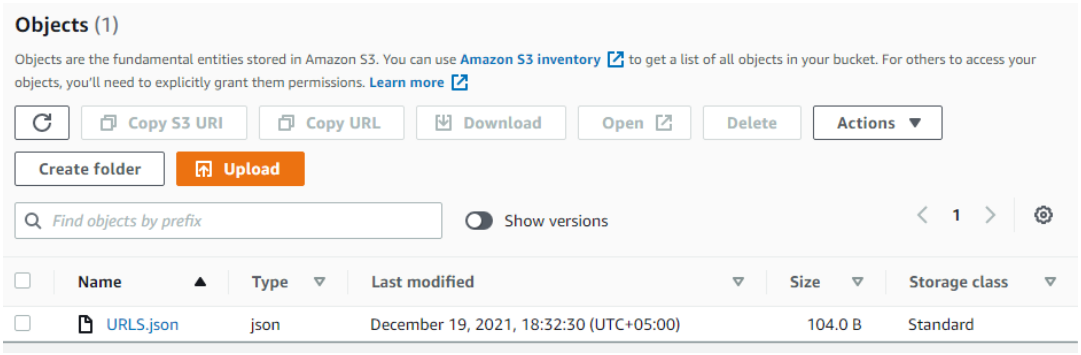


Figure 6 File uploaded to the bucket

1.5.2 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:

1.5.2.1 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.

1.5.2.2 Boto3 (*)

I found another way to read from my bucket using boto3, and yes the method exits 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.

1.5.2.3 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_oject = json.loads(cont.read())
    list_url=[json_oject['w1'],json_oject['w2'],json_oject['w3'],json_oject['w4']]
    return list_url
```

1.6 Sprint1 Task 6

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.

Figure 7 Content in URLs.txt file

1.6.1 Creating Metris 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+'Avaiability_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))
```

113:42 Python Spaces: 4

1.6.2 Results on List of URLs

So now the alarms are being created for the metrics which are found for each URL.

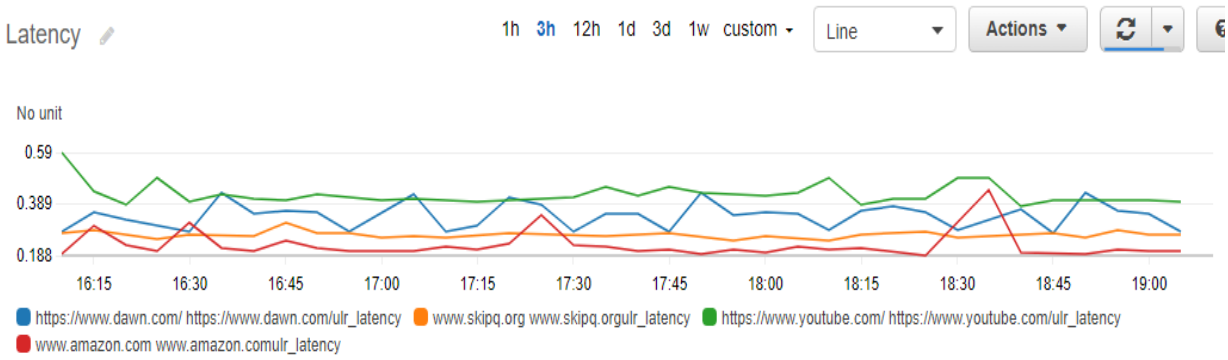


Figure 8 Graphed metrics of 4 URLs

1.7 Cloud Formation Diagram

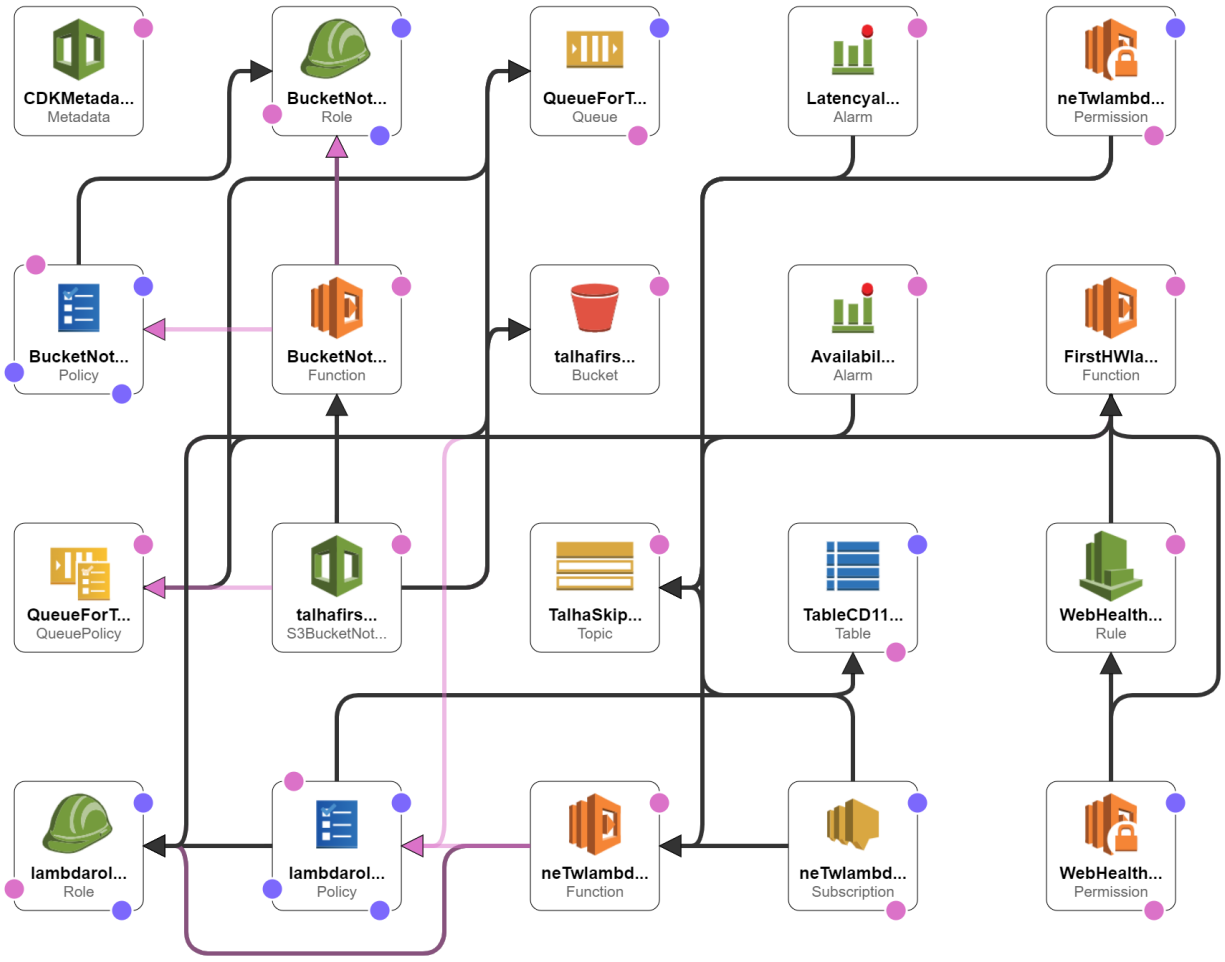


Figure 9 Cloud formation diagram

2 Sprint2

2.1 Introduction

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 rapidly and reliably deliver features or updates. Our code can be built, tested, and deployed in a test environment as well as directly in a production environment. Whenever an error occurs after a commit, the pipeline rolls back and does not accept that change.

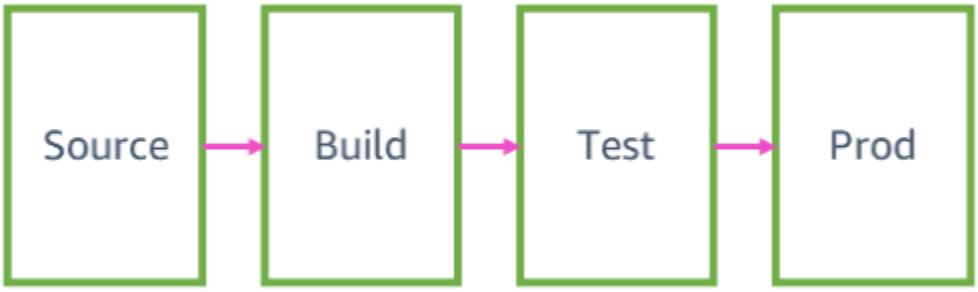


Figure 10 Phases of CD pipeline

2.2 Sprint2 Task1

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 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)
```

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 will be 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.

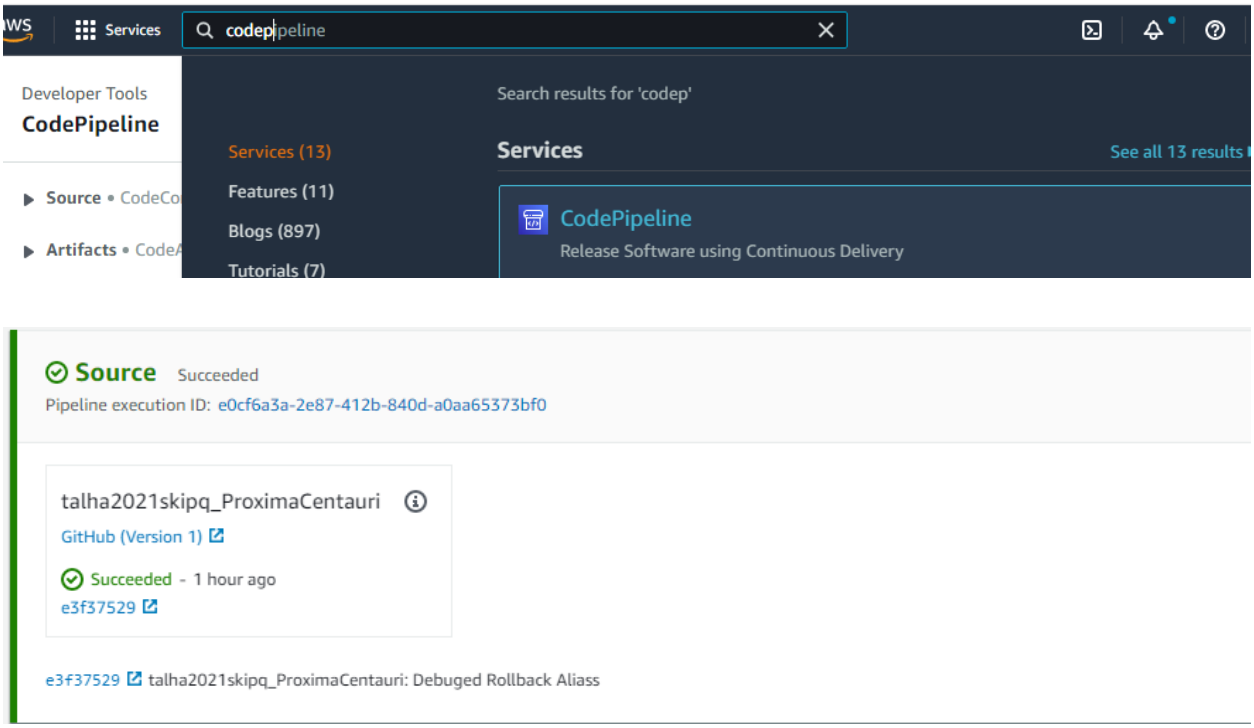
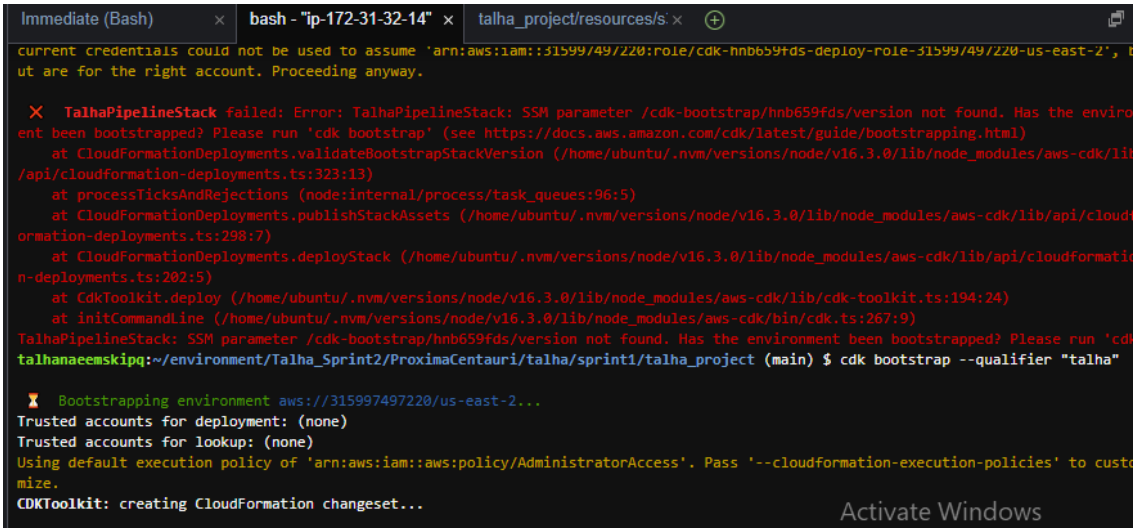


Figure 12 Source in the code pipeline

2.2.1.2 Related Issues

2.2.1.2.1 Internal failure

- Current credentials could not be used



2.2.1.2.1.1 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:

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.

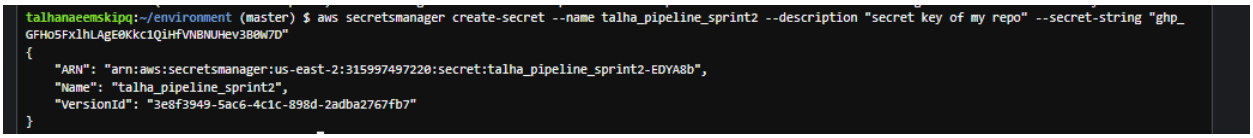


Figure 13 Saving secret key

2.3 Sprint2 Task2

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.

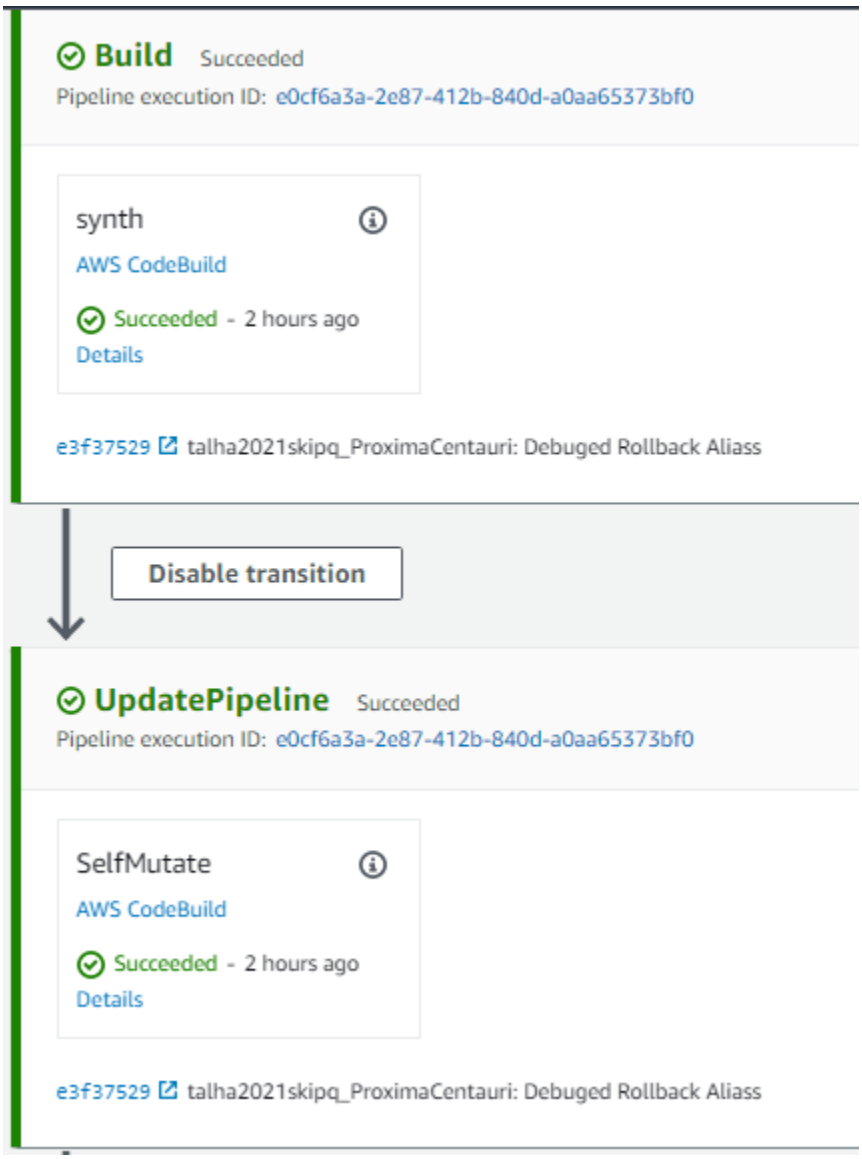
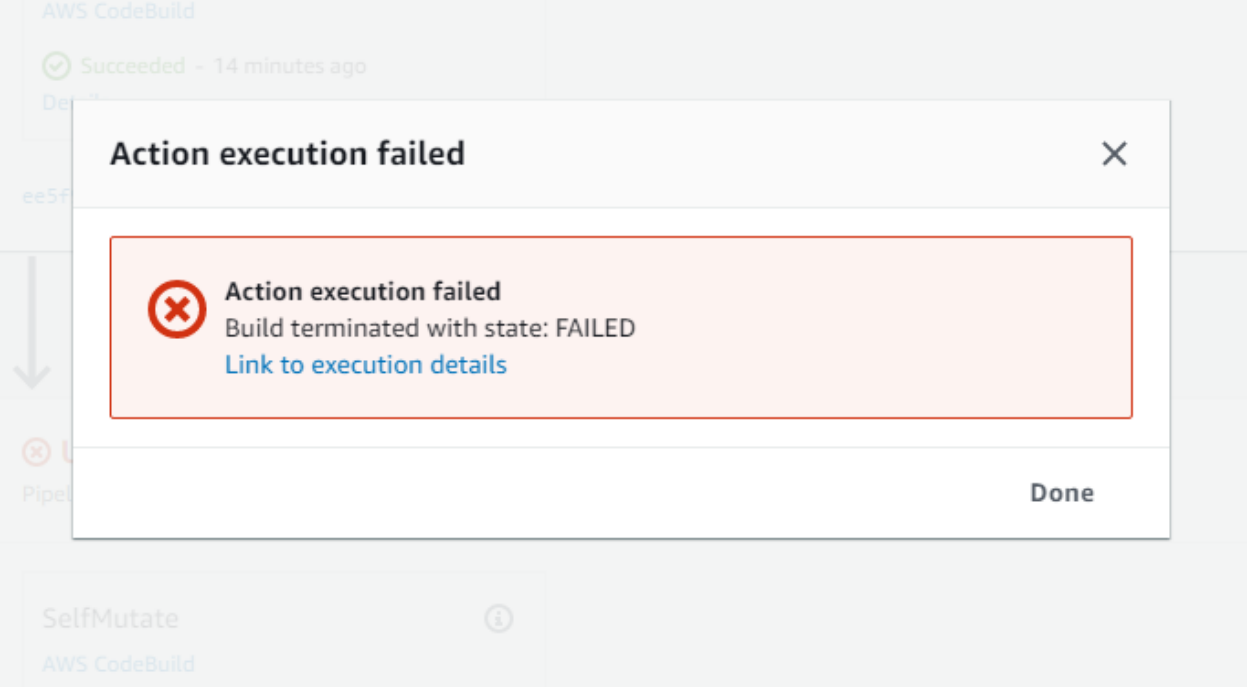


Figure 14 Build phase

2.3.1.1 Related Issues

2.3.1.1.1 Execution failure issue in pipeline update



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

```
Toolkit stack: CDKToolkit
Setting "CDK_DEFAULT_REGION" environment variable to us-east-2
Resolving default credentials
Looking up default account ID from STS
Default account ID: 315997497220
Setting "CDK_DEFAULT_ACCOUNT" environment variable to 315997497220
context: {
  'aws:cdk:enable-path-metadata': true,
  'aws:cdk:enable-asset-metadata': true,
  'aws:cdk:version-reporting': true,
  'aws:cdk:bundling-stacks': [ '*' ]
}
--app points to a cloud assembly, so we bypass synth
No stacks match the name(s) TalhaPipelineStack
Error: No stacks match the name(s) TalhaPipelineStack
    at CdkToolkit.validateStacksSelected (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:545:13)
    at CdkToolkit.selectStacksForDeploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:492:10)
    at CdkToolkit.deploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:120:20)
    at initCommandLine (/usr/local/lib/node_modules/aws-cdk/bin/cdk.ts:267:9)

[Container] 2021/12/23 14:43:24 Command did not exit successfully cdk -a . deploy TalhaPipelineStack --require-
approval=never --verbose exit status 1
[Container] 2021/12/23 14:43:24 Phase complete: BUILD State: FAILED
[Container] 2021/12/23 14:43:24 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
cdk -a . deploy TalhaPipelineStack --require-approval=never --verbose. Reason: exit status 1
```

2.3.1.1.1.1 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.

2.3.1.1.2 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.

```
Toolkit stack: CDKToolkit
Setting "CDK_DEFAULT_REGION" environment variable to us-east-2
Resolving default credentials
Looking up default account ID from STS
Default account ID: 315997497220
Setting "CDK_DEFAULT_ACCOUNT" environment variable to 315997497220
context: {
  'aws:cdk:enable-path-metadata': true,
  'aws:cdk:enable-asset-metadata': true,
  'aws:cdk:version-reporting': true,
  'aws:cdk:bundling-stacks': [ '*' ]
}
--app points to a cloud assembly, so we bypass synth
No stacks match the name(s) TalhaPipelineStack
Error: No stacks match the name(s) TalhaPipelineStack
    at CdkToolkit.validateStacksSelected (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:545:13)
    at CdkToolkit.selectStacksForDeploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:492:10)
    at CdkToolkit.deploy (/usr/local/lib/node_modules/aws-cdk/lib/cdk-toolkit.ts:120:20)
    at initCommandLine (/usr/local/lib/node_modules/aws-cdk/bin/cdk.ts:267:9)

[Container] 2021/12/23 16:16:00 Command did not exit successfully cdk -a . deploy TalhaPipelineStack --require-
approval=never --verbose exit status 1
[Container] 2021/12/23 16:16:00 Phase complete: BUILD State: FAILED
[Container] 2021/12/23 16:16:00 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing command:
cdk -a . deploy TalhaPipelineStack --require-approval=never --verbose. Reason: exit status 1
[Container] 2021/12/23 16:16:00 Entering phase POST_BUILD
[Container] 2021/12/23 16:16:00 Phase complete: POST_BUILD State: SUCCEEDED
[Container] 2021/12/23 16:16:00 Phase context status code:  Message:
```

2.3.1.1.2.1 Solution

1- Delete all buckets from S3(except the ones you defined manually)

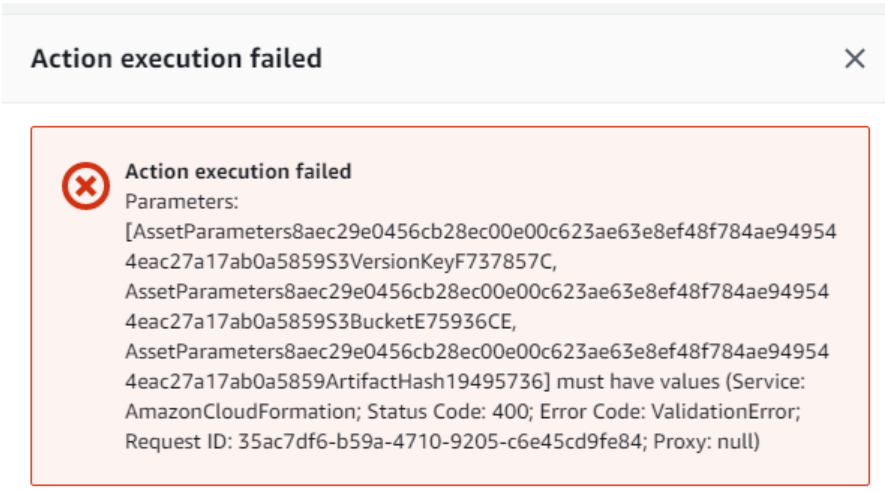
2- Delete pipeline stack(s) from cloud formation

3- Delete pipeline from codepipelines

4- Run the command: cdk bootstrap aws://12-digit-id/region –qualifier talha –toolkit-stack tkname

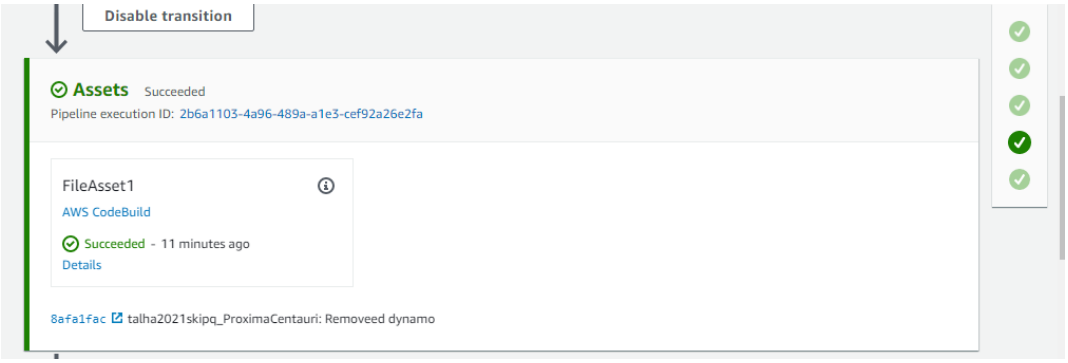
2.3.1.1.3 Execution Failure in Beta Stage

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:



2.3.1.1.3.1 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.



2.3.1.1.4 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.

2.4 Sprint2 Task3

2.4.1 Adding Beta Stage to Pipeline

2.4.1.1 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. So assets phase is a pre-requisite step for the beta stage.

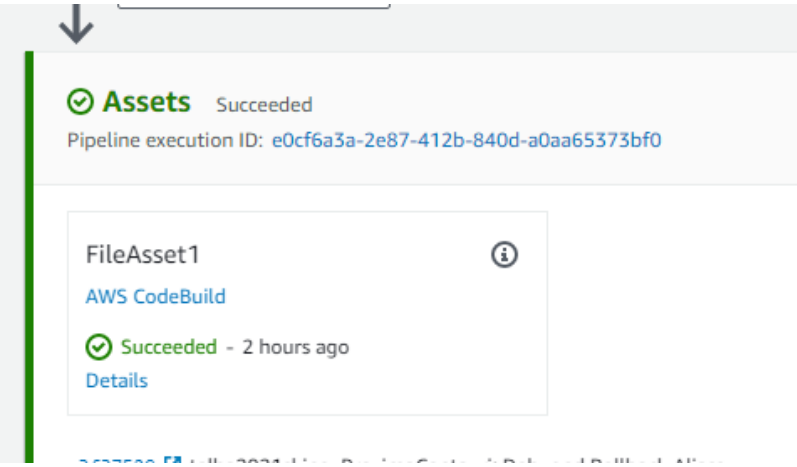


Figure 15 Assets in code pipeline

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:

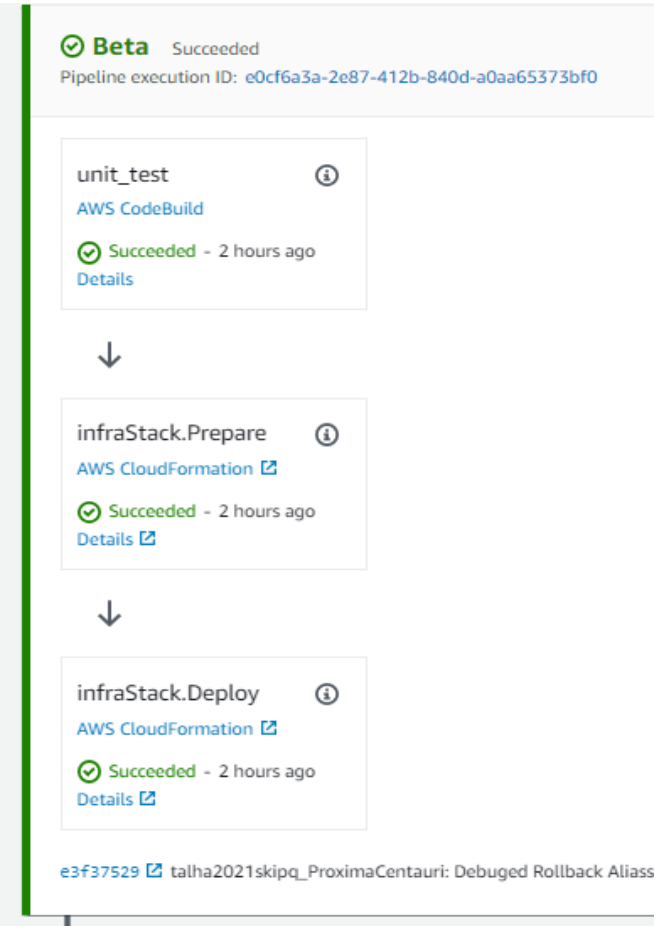


Figure 16 Beta stage in code pipeline

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

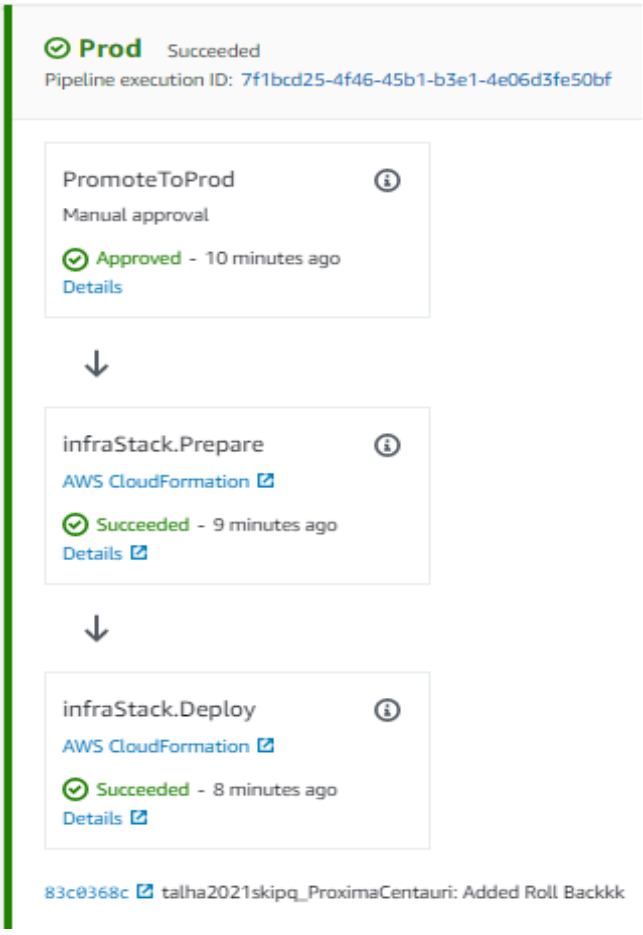
```
from aws_cdk import core
#import aws_cdk.assertions as assertions
from talha_project.talha_project_stack import TalhaProjectStack
def test_lambda():
    app=core.App()
    stack=TalhaProjectStack(app, 'infStack')
    #template = assertions.Template.from_stack(stack)
    template=app.synth().get_stack_by_name('infStack').template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

2.5 Sprint2 Task5

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 Sprint2 Task6

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])
ault: LambdaDeploymentConfig.CANARY_10PERCENT_5MINUTES
```

2.6.2 Failure Alarm Creation

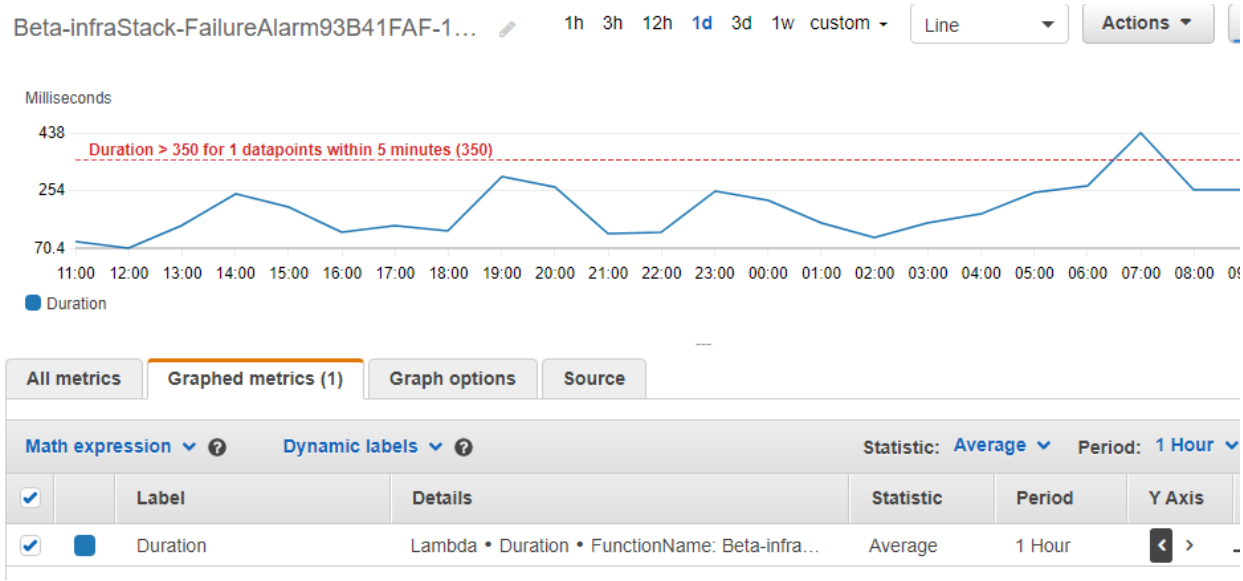
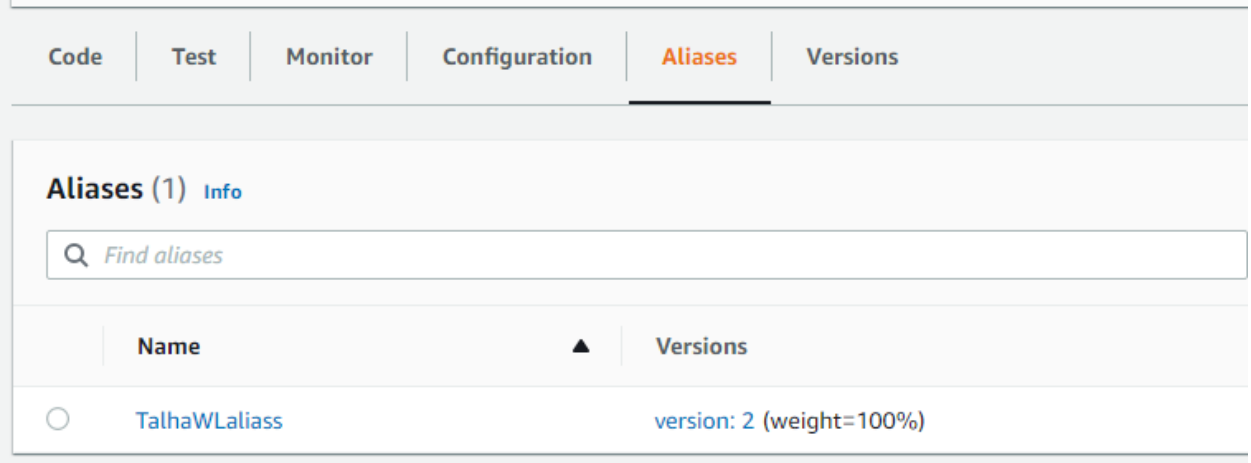


Figure 17 The alarm on lambda duration metric

2.6.3 Alias Generation

An alias is generated 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 new version and then after some wait it will act accordingly on the basis of failure alarms, the supporting figure are shown below.



Here you can see your lambda's auto roll back in 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-TalhaidApplicationE4ACB22B-94Q26J0ZAGF2	Beta-infraStack-TalhaidEF86370C-RTDB56FXC9XO	323299ac…	CodeDeploy rollback	Dec 27, 2021 11:43 PM (UTC+5:00)	Dec 27, 2021 11:43 PM (UTC+5:00)

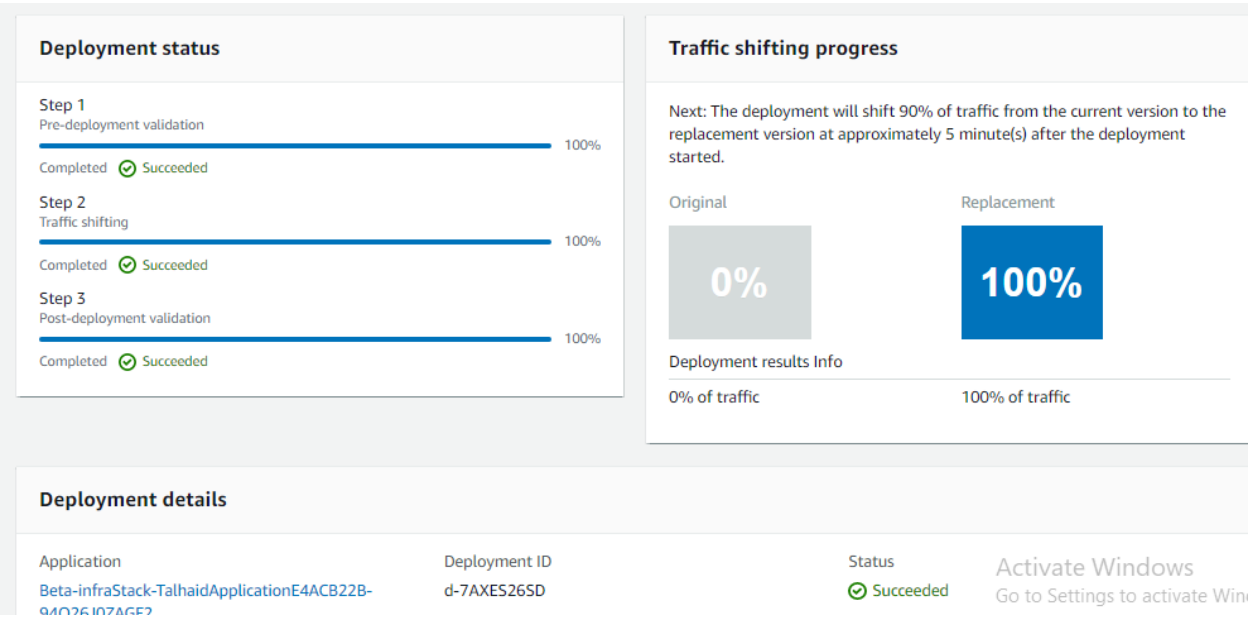


Figure 18 Traffic shifting progress on lambda

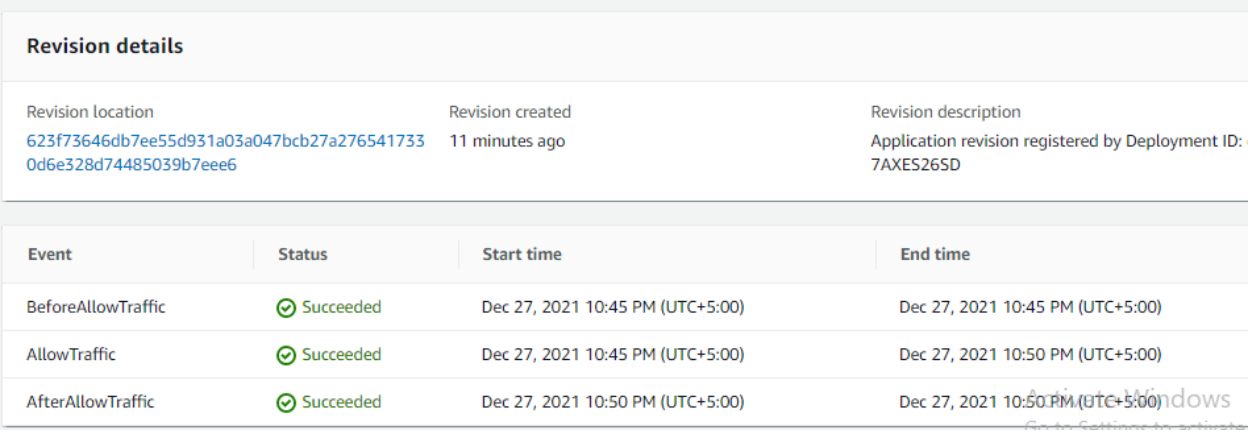


Figure 19 Traffic shift details

2.6.4 Related Issues

2.6.4.1 The issue in Lambda Alias:

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

2.6.4.1.1 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.

3 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