



**Written by**  
Adeel Shahzad  
SkipQ ProximaCentauri Cohort  
Trainee Dec/2021 to Jan/2022

## Catalog

<b>1. Orientation.....</b>	<b>3</b>
1.2 Cloud Computing.....	3
1.1.1. What is cloud computing.....	3
1.1.2. Benefits.....	3
1.2. Amazon Web Services(AWS).....	3
1.2.1. What is AWS.....	3
1.2.2. Benefits.....	3
1.3. DevOps.....	4
1.3.1. What is DevOps.....	4
1.3.2. Benefits.....	4
<b>2. Sprint 1.....</b>	<b>4</b>
2.1. Technologies Used.....	4
2.1.1. Cloud9.....	4
2.1.2. Lambda.....	4
2.1.3. Cloud Watch.....	5
2.1.4. Dynamo DB.....	5
2.2. Task 1: Hello Lambda.....	5
2.2.1. Implementation.....	5
2.2.2. Results.....	5
2.3. Task 2: Web Health Lambda.....	6
2.3.1. Implementation.....	6
2.3.2. Results.....	6
2.4. Task 3: Web Matrix Lambda.....	6
2.4.1. Implementation.....	6
2.4.2. Results.....	7
2.5. Task 4: Web Alarm Lambda.....	7
2.5.1. Implementation.....	7
2.5.2. Results.....	7
2.6. Task 5: Dynamo DB & Web Lambda.....	8
2.6.1. Implementation.....	8
2.6.2. Results.....	8
2.7. Errors & Solution.....	9
<b>3. Sprint 2.....</b>	<b>11</b>
3.1. Technologies Used.....	11
3.1.1. Pipelines.....	11
3.1.2. CloudFormation.....	12
3.2. Task 1: Pipeline Source & build.....	12
3.2.1. Implementation.....	12
3.2.2. Results.....	12
3.3. Task 2: Beta Stage.....	13
3.3.1. Implementation.....	13
3.3.2. Results.....	13
3.4. Task 3: Unit and integration tests.....	14
3.4.1. Implementation.....	14
3.4.2. Results.....	14
3.5. Task 4: Production stage.....	14
3.5.1. Implementation.....	14
3.5.2. Results.....	14
3.6. Task 5: Roll Back.....	15
3.6.1. Implementation.....	15
3.6.2. Results.....	15
3.7. Errors & Solution.....	15
<b>4. Sprint 3.....</b>	<b>18</b>
4.1. Technologies Used.....	18

4.1.1. API Gateway.....	18
4.1.2. S3.....	18
4.2. Task 1: Moving data from S3.....	19
4.2.1. Implementation.....	19
4.2.2. Results.....	19
4.3. Task 2: Setting Alarms on dynamo data.....	20
4.3.1. Implementation.....	20
4.3.2. Results.....	20
4.4. Task 3: API gateway and its methods.....	20
4.4.1. Implementation.....	20
4.4.2. Results.....	21
4.5. Task 4: Extra tests.....	22
4.5.1. Implementation.....	22
4.5.2. Result.....	22
4.6. Errors & Solution.....	22
<b>5. Sprint 4.....</b>	<b>24</b>
5.1. Technologies Used.....	24
5.1.1. AWS Amplify.....	24
5.1.2. AWS Cognito.....	24
5.2. Task 1: React App.....	25
5.2.1. Implementation.....	25
5.2.2. Results.....	25
5.3. Task 2: API connection.....	25
5.3.1. Implementation.....	25
5.3.2. Results.....	25
5.4. Task 3: Deploy to S3.....	26
5.4.1. Implementation.....	26
5.4.2. Results.....	26
5.5. Task 4: OAuth.....	27
5.5.1. Implementation.....	27
5.5.2. Result.....	27
5.6. Errors & Solution.....	27
<b>6. Sprint 5.....</b>	<b>28</b>
6.1. Technologies Used.....	28
6.1.1. AWS ECR.....	28
6.1.2. AWS ECS.....	28
6.1.3. Docker Image.....	29
6.1.4. PyRestTest.....	29
6.2. Task 1: Syntribos & PyRestTest Docker Image.....	29
6.2.1. Implementation.....	29
6.2.2. Results.....	29
6.3. Task 2: Publish to ECR.....	30
6.3.1. Implementation.....	30
6.3.2. Results.....	30
6.4. Task 3: Deploy in ECS.....	30
6.4.1. Implementation.....	30
6.4.2. Results.....	30
6.5. Errors & Solution.....	31

# **1. Orientation**

## **1.2 Cloud Computing**

### **1.1.1. What is cloud computing**

The delivery of various services over the Internet is known as cloud computing. These resources include data storage, servers, databases, networking, and software, among other tools and applications. As long as an electronic gadget has internet access, it has access to data and the software programmers needed to execute it.

### **1.1.2. Benefits**

#### **1.1.2.1. Cost saving**

If you are worried about the price tag that would come with making the switch to cloud computing, you aren't alone 20% of organisations are concerned about the initial cost of implementing a cloud-based server. But those who are attempting to weigh the advantages and disadvantages of using the cloud need to consider more factors than just initial price they need to consider ROI.

#### **1.1.2.2. Security**

Many organization have security concerns when it comes to adopting a cloud-computing solution. After all, when files, programs, and other data aren't kept securely onsite, how can you know that they are being protected? If you can remotely access your data, then what's stopping a cyber criminal from doing the same thing? Well, quite a bit, actually.

#### **1.1.2.3. Flexibility**

Your business has only a finite amount of focus to divide between all of its responsibilities. If your current IT solutions are forcing you to commit too much of your attention to computer and data-storage issues, then you aren't going to be able to concentrate on reaching business goals and satisfying customers. On the other hand, by relying on an outside organization to take care of all IT hosting and infrastructure, you'll have more time to devote toward the aspects of your business that directly affect your bottom line.

## **1.2. Amazon Web Services(AWS)**

### **1.2.1. What is AWS**

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

### **1.2.2. Benefits**

#### **1.2.2.1. Easy to Use**

AWS is designed to allow application providers, ISVs, and vendors to quickly and securely host your applications – whether an existing application or a new SaaS-based application. You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.

#### **1.2.2.2. Reliable**

With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon.com's multi-billion dollar online business that has been honed for over a decade.

### 1. 2. 2. 3. **Scale able**

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

## 1. 3. **DevOps**

### 1. 3. 1. **What is DevOps**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.

### 1. 3. 2. **Benefits**

#### 1. 3. 2. 1. **Ensure Fast Deployment**

Faster and more frequent delivery of updates and features will not only satisfy the customers but will also help your company take a firm stand in a competitive market.

#### 1. 3. 2. 2. **Stabilize Work Environment**

Do you know that the tension involved in the release of new features and fixes or updates can topple the stability of your workspace and decreases the overall productivity? Improve your work environment with a steady and well-balanced approach of operation with DevOps practice.

#### 1. 3. 2. 3. **Improvement In Product Quality**

Collaboration between development and operation teams and frequent capturing of user feedback leads to a significant improvement in the quality of the product.

## 2. **Sprint 1**

### 2. 1. **Technologies Used**

#### 2. 1. 1. **Cloud9**

AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects. Since your Cloud9 IDE is cloud-based, you can work on your projects from your office, home, or anywhere using an internet-connected machine. Cloud9 also provides a seamless experience for developing serverless applications enabling you to easily define resources, debug, and switch between local and remote execution of serverless applications. With Cloud9, you can quickly share your development environment with your team, enabling you to pair program and track each other's inputs in real time.

#### 2. 1. 2. **Lambda**

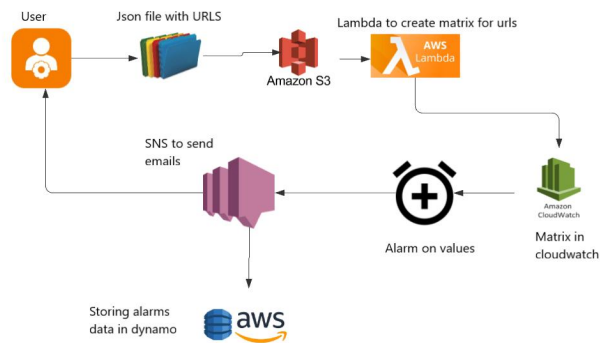
Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and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

### 2. 1. 3. Cloud Watch

Amazon CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), IT managers, and product owners. CloudWatch provides you with data and actionable insights to monitor your applications, respond to system-wide performance changes, and optimize resource utilization. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events. You get a unified view of operational health and gain complete visibility of your AWS resources, applications, and services running on AWS and on-premises. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.

### 2. 1. 4. Dynamo DB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data. For more information, see DynamoDB Encryption at Rest.



Flow diagram for sprint 1

## 2. 2. Task 1: Hello Lambda

### 2. 2. 1. Implementation

For our first task on AWS we implemented a Hello world lambda function, as beginners do in any new programming field. Created the directory for project imported the important libraries to our stack file and created a Lambda file in resources folder to start the process. First of all Defined a handler function for our Lambda file in stack. And in that file defined all the parameters required for a lambda function invoke in our stack file. In lambda file defined function for event and context and then given two strings to it for printing in console. User name concatenated with the hello word to print my first cloud computing message.

### 2. 2. 2. Results

Results of first hello lambda function are given in figure 1.

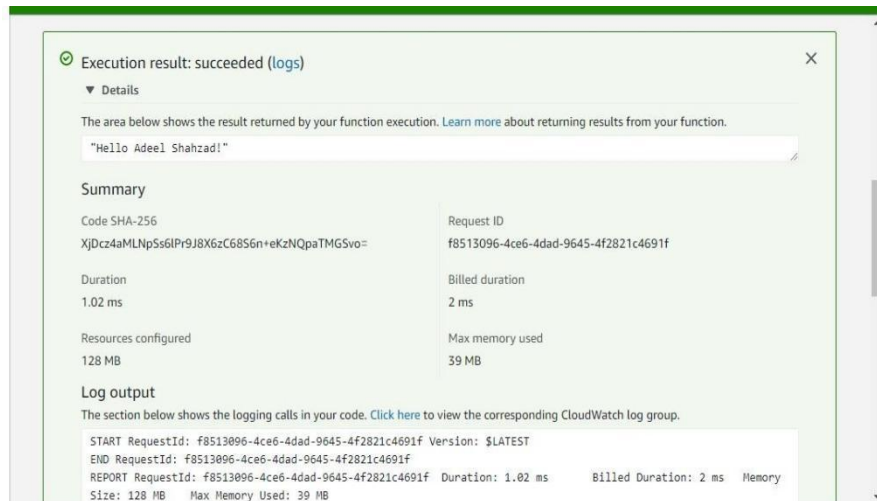


Figure 1 : Hello lambda results

## 2. 3. Task 2: Web Health Lambda

### 2. 3. 1. Implementation

In this task we tried to implement a web health lambda function. This function returns the latency and availability of a given URL in Numeric values. As the same way started the lambda function and stack file for this task. Our stack files remains same because we don't need any extra libraries for this task. We made changes in lambda file and created two functions. First one is for calculating the availability of given URL in terms of ones and zeros. Used built in function for that which returns boolean output, so using if conditions converted them in ones and zeros. If a Web is not down it returns one else zero. And then created a function for latency of that web. It returns float values between zero and one. For latency requested a response from the given URL and saved the time before and after the response. Taken their difference and converted it to seconds as latency value.

### 2. 3. 2. Results

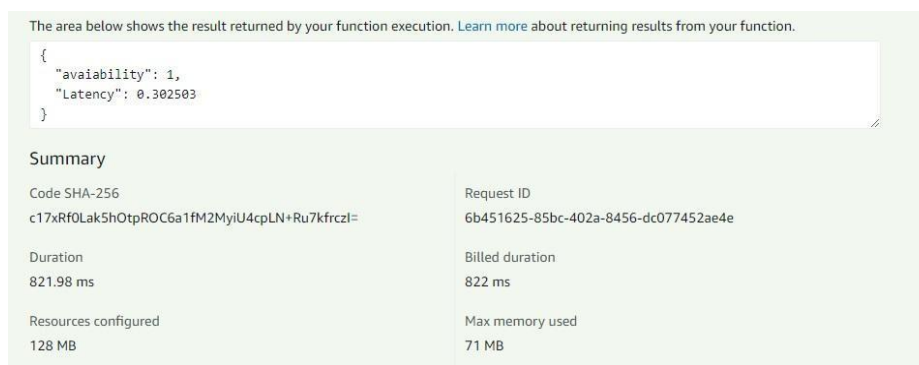


Figure 2: web heath parameter

## 2. 4. Task 3: Web Matrix Lambda

### 2. 4. 1. Implementation

In this task created a periodic lambda function that invokes after a certain duration to show the availability and latency of URL in a matrix graph to observe the trends related to that two parameters. For this task again used the functions from previous file that returns the values of latency and availability in numeric form. Created a matrix and given the values to that matrix. To be plotted in a graph and shown in cloud watch. In stack file created schedule for the lambda function to invoke after certain amount of time intervals. Imported cloud watch and related libraries in the lambda file. Initialized lambda rule function in the stack file to get access to cloud watch services.

## 2. 4. 2. Results

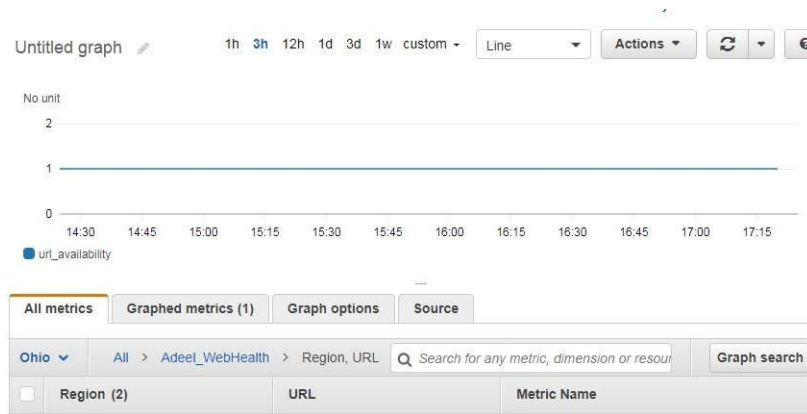


Figure 3:Availability Matrix

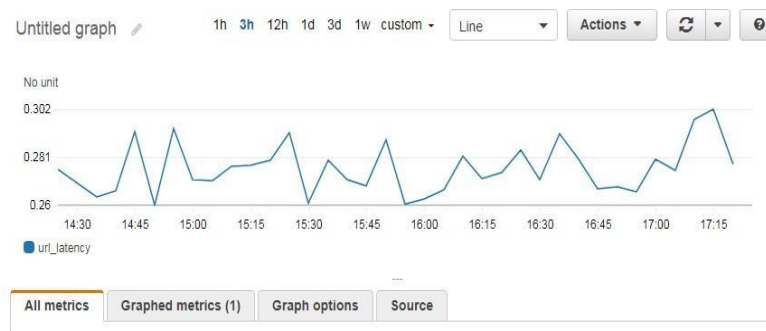


Figure 4:Latency Matrix

## 2. 5. Task 4: Web Alarm Lambda

### 2. 5. 1. Implementation

This function not only shows the values of latency and availability of URL but creates an alarm If the values go beyond a certain threshold. For this function we used the functions from previous task. To implement it created another matrix in stack file and given it the same parameters as the matrix in lambda function. So that both of them get merged. Generated a threshold for each parameter and defined a alarm function from AWS libraries to create an alarm, Given this function all the parameters required to generate an alarm including the matrix and threshold. After that generated a action for each alarm by sending an email to the related person . For this imported the actions and subscriptions libraries. Subscribed to the email of given user and created a action for that email. So it sends data of the alarm to all subscribed emails.

### 2. 5. 2. Results



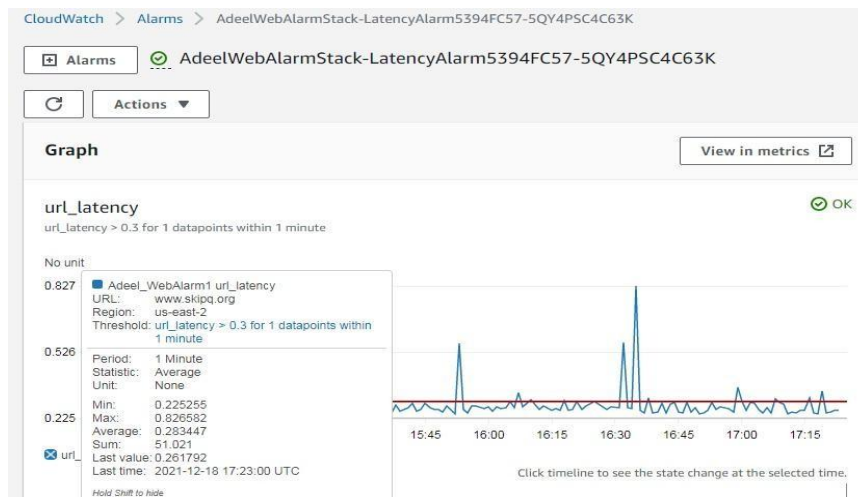


Figure 5:Latency Alarm Matrix

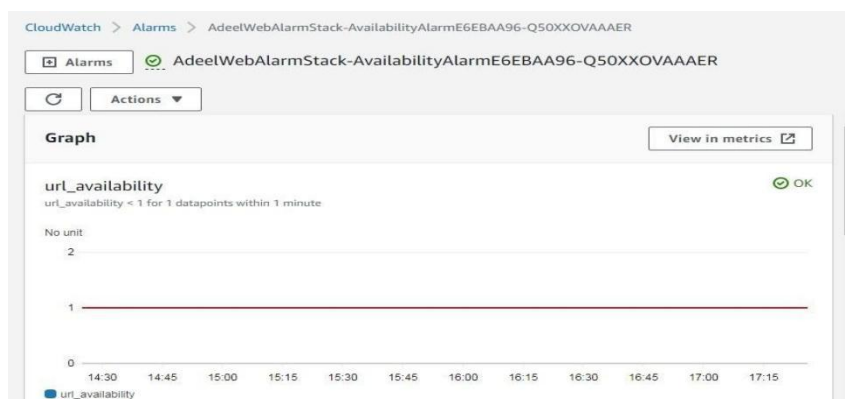


Figure 6:Availability Alarm Matrix

## 2. 6. Task 5: Dynamo DB & Web Lambda

### 2. 6. 1. Implementation

During this task initialized a new bucket of multiple urls for alarm generation and that created a table for dynamoDB database to update the alarm values in it.First of all created the table for dynamoDB and then given lambda function all the access to read and write in that function. Created a separate lambda file for adding sns alarm values to it created events for message of sns alarm and provided the message values to our table to be stored in dynamoDB database. Created a bucket to read more than one urls

### 2. 6. 2. Results

► AdeelAlarmdynamo View table details

Expand to query or scan items.

Items returned (300) Refresh Actions Create item

<input type="checkbox"/>	Timestamp	Reason
<input type="checkbox"/>	2021-12-2...	Threshold Crossed: 1 out of the last 1 datapoints [0.318658 (22/12/21 03:43:00)] was grea...
<input type="checkbox"/>	2021-12-2...	Threshold Crossed: 1 out of the last 1 datapoints [0.347319 (22/12/21 17:21:00)] was grea...

Figure 7:Dynamo table

## 2. 7. Errors & Solution

```
(.venv) adeelshahzadskipq:~/environment/MyDemoRepo (master) $ cdk synth
Traceback (most recent call last):
  File "app.py", line 12, in <module>
    from my_demo_repo.my_demo_repo_stack import MyDemoRepoStack
  File "/home/ec2-user/environment/MyDemoRepo/my_demo_repo/my_demo_repo_stack.py", line 1, in <module>
    from aws_cdk import (
ImportError: cannot import name 'aws_es2' from 'aws_cdk' (unknown location)
Subprocess exited with error 1
```

Figure 8:CDK library missing

### Solution:

This was the first error I encountered. My Ide was not able to find CDK module from the Libraries installed so I first used PIP commands to install the requirements to get started with the coding.

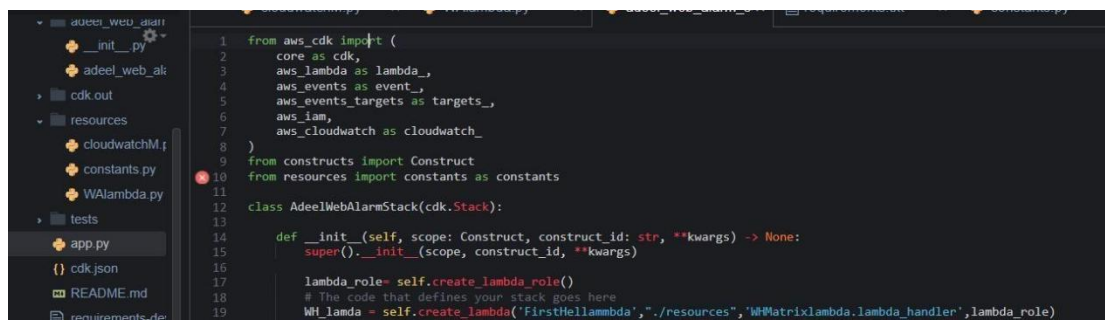


Figure 9:Not picking up the constants files

### Solution:

My module wasn't able to find any constants file from the resources folder, but it was right there. This error occurred due to multiple files with same names. So I changed the names and got rid of it.

```
Immediate (Javascript (br x) bash - "ip-172-31-38-191.x" +
File app.py, line 0, in <module>
  from adeel_who_matrix.adeel_who_matrix_stack import AdeelWhoMatrixStack
File "/home/ec2-user/environment/Project1/AdeelWhoMatrix/adeel_who_matrix/adeel_who_matrix_stack.py", line 1, in <module>
  from aws_cdk import (
ImportError: cannot import name 'Stack' from 'aws_cdk' (unknown location)
Subprocess exited with error 1
adeelshahzadskipq:~/environment/Project1/AdeelWhoMatrix (master) $ cdk synth
Traceback (most recent call last):
  File "app.py", line 9, in <module>
    app = cdk.App()
AttributeError: module 'aws_cdk' has no attribute 'App'
Subprocess exited with error 1
adeelshahzadskipq:~/environment/Project1/AdeelWhoMatrix (master) $ cdk synth
Traceback (most recent call last):
  File "app.py", line 9, in <module>
    app = cdk.App()
AttributeError: module 'aws_cdk' has no attribute 'App'
Subprocess exited with error 1
adeelshahzadskipq:~/environment/Project1/AdeelWhoMatrix (master) $
```

Figure 10:App.py file was not working fine

### Solution:

Same as first error but occurred due to file name was given in a wrong way inside the stack file. Changed the way it was given and got rid of it

```
(.venv) adeelshahzadskipq:~/environment/MyDemoRepo (master) $ cdk deploy
MyDemoRepoStack: deploying...

✔ MyDemoRepoStack (no changes)

Stack ARN:
arn:aws:cloudformation:us-east-2:315997497220:stack/MyDemoRepoStack/34f30740-5d72-11ec-ba6e-02bc742e1cea
(.venv) adeelshahzadskipq:~/environment/MyDemoRepo (master) $
```

Figure 11:No changes gets detected

### Solution:

Different functions started to get deployed without any change. And all the changes made to them were not detected so I figured that in my main machine another file with same name is present so changed the name and got it working fine.

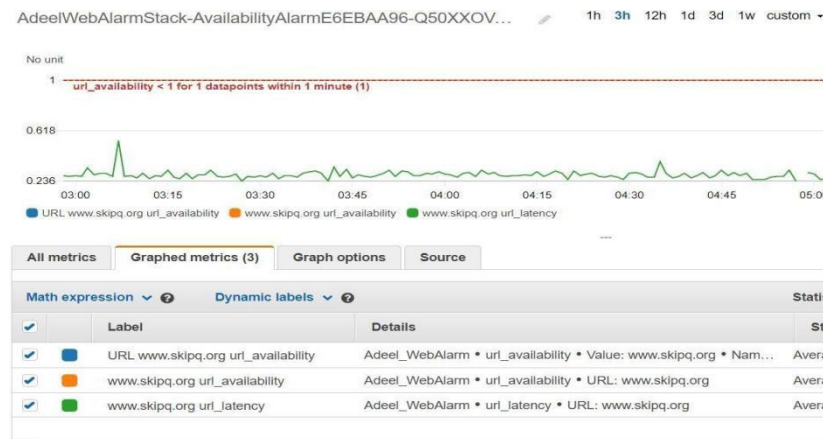


Figure 12:Alarm matrix not merging

### Solution:

Both matrices failed to merge with each other all the latency and availability values from lambda file shown on one matrix and alarm and threshold from stack file shown in an other matrix. So to solve this error looked deeply into the code and found out that parameters for both matrices from lambda file and stack file are different given them same parameters and and run the code smoothly.

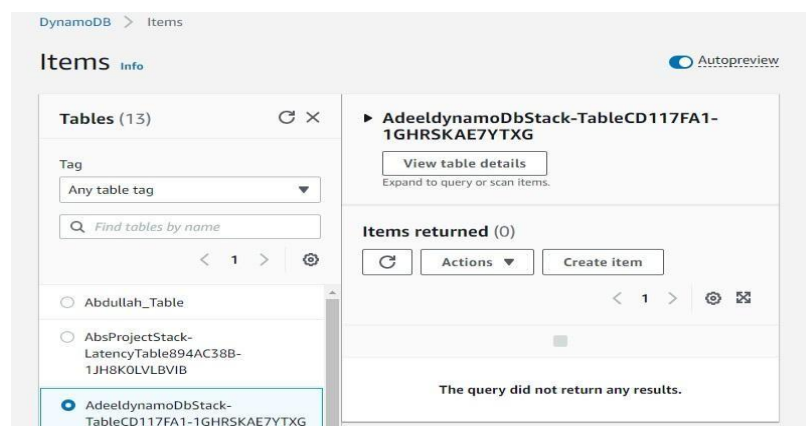


Figure 13:Table is not showing the values

### Solution:

Table is created successfully but failed to get values. Get to now the index of message that was going to be printed in the table. Given it the right way and solved it.

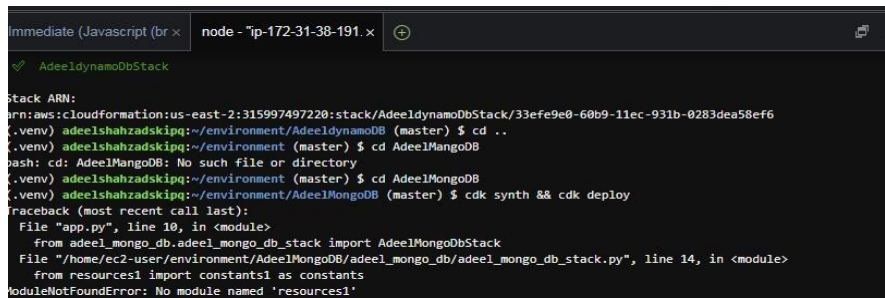
A terminal window with a dark background. The title bar shows 'Immediate (Javascript (br x...))' and 'node - "ip-172-31-38-191.x...'. The prompt is 'adeelshahzadskipq@adeelshahzadskipq:~/environment/AdeelMongoDB (master)'. The user has run 'cd AdeelMongoDB' and 'cdk synth', which resulted in a 'ModuleNotFoundError: No module named 'resources1''.

Figure 14:folder not found in directory

**Solution:**

Failed to find the resources file So changed the names to solve the issue. App file was not given any CDK library. Defined a proper library and done.

## 3. Sprint 2

### 3. 1. Technologies Used

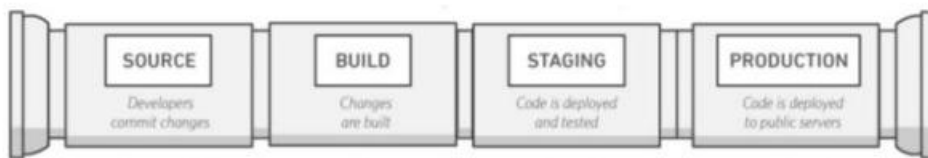
#### 3. 1. 1. Pipelines

This walkthrough builds a pipeline for a sample WordPress site in a stack. The pipeline is separated into three stages. Each stage must contain at least one action, which is a task the pipeline performs on your artifacts (your input). A stage organizes actions in a pipeline. CodePipeline must complete all actions in a stage before the stage processes new artifacts, for example, if you submitted new input to rerun the pipeline.

- By the end of this walkthrough, you'll have a pipeline that performs the following workflow:
- The first stage of the pipeline retrieves a source artifact (an AWS CloudFormation template and its configuration files) from a repository.
- You'll prepare an artifact that includes a sample WordPress template and upload it to an S3 bucket.
- In the second stage, the pipeline creates a test stack and then waits for your approval.
- After you review the test stack, you can choose to continue with the original pipeline or create and submit another artifact to make changes. If you approve, this stage deletes the test stack, and then the pipeline continues to the next stage.
- In the third stage, the pipeline creates a change set against a production stack, and then waits for your approval.
- In your initial run, you won't have a production stack. The change set shows you all of the resources that AWS CloudFormation will create. If you approve, this stage executes the change set and builds your production stack.

### 3. 1. 2. CloudFormation

Developers can deploy and update compute, database, and many other resources in a simple, declarative style that abstracts away the complexity of specific resource APIs. AWS CloudFormation is designed to allow resource lifecycles to be managed repeatably, predictable, and safely, while allowing for automatic rollbacks, automated state management, and management of resources across accounts and regions. Recent enhancements and options allow for multiple ways to create resources, including using AWS CDK for coding in higher-level languages, importing existing resources, detecting configuration drift, and a new Registry that makes it easier to create custom types that inherit many core CloudFormation benefits.



Flow Diagram for Sprint 2

## 3. 2. Task 1: Pipeline Source & build

### 3. 2. 1. Implementation

First of all started with the basic implementation of pipeline. In this task tried to implement the first three stages of pipelines. First of all wrote the code for source stage in this stage all the important libraries get installed in the pipeline environment and behaviors are defined. Then comes the build stage in which code is picked up from the github repository and synth the code for any errors to build a exe file for the code so that code can be available at any time in forms of executable file. If there are problems or errors in the build, the process stops and issues are reported back to the developers for remediation. Typical problems include functional errors, such as a divide-by-zero math error, or missing components -- for example, a required library or module is not present in the build manifest.

We must first construct a pipeline source. GitHub will be a third-party source that we will integrate into our workflow. Build the source after you've added it. The source code is presented below. Add a repository, a branch, and the name of the secret where you've saved your personal access token to the repo string. Code Pipeline examines the source for changes on a regular basis using GitHub's "POLL" feature.

### 3. 2. 2. Results



Figure 15:Source state



Figure 16:Build State

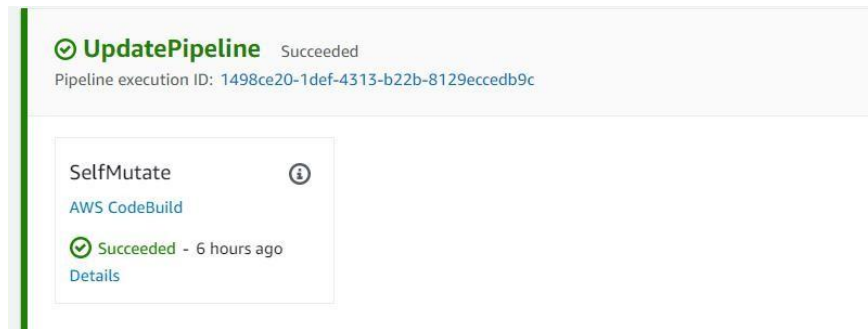


Figure 17:Update pipeline step

## 3. 3. Task 2: Beta Stage

### 3. 3. 1. Implementation

We must now develop the Beta Stage. In addition, a unit test is built. The test is set to pre when adding the Beta stage. It signifies that if the test succeeds, the code will be deployed. Commit the code once again and upload it to GitHub. On Code Pipeline, you'll find these results.

### 3. 3. 2. Results

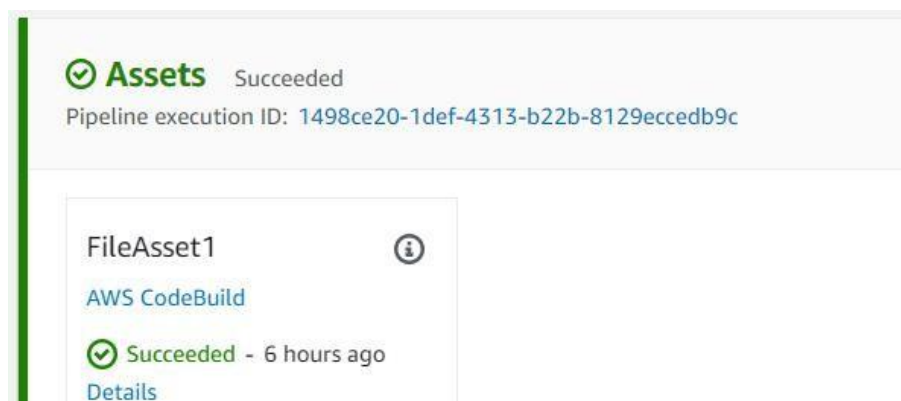


Figure 18:Assets update step



Figure 19:Beta stage

### 3. 4. Task 3: Unit and integration tests

#### 3. 4. 1. Implementation

We have deployed the code in two stages: beta and production. For both the Beta and Production stages, we collect Availability and Latency Metrics. The availability graphs for Beta and Prod for one URL are provided below. Metrics and alarms are built in the same way for both the Beta and Prod phases. Separate Dynamo DB tables are also generated. As a result, we update the dynamo Lambda code to reflect this. Alarm generates in the Pro or Beta level, according to Alarm. It saves the details of the alert in the relevant database. To reach these results we added different test in our stages so that a desired output acceptable by the client can be obtained on the screen.

#### 3. 4. 2. Results



Figure 20:Unit test

### 3. 5. Task 4: Production stage

#### 3. 5. 1. Implementation

The final step in the Pipeline is to add production. We established a production stage and added it to the pipeline as a pre with manual permissions. It implies it will ask the user for approval before deploying the code to the server. The code for adding a production stage may be found here. Commit and send the code to the GitHub repository once more. These are the outcomes you'll receive if you use Code Pipeline.

#### 3. 5. 2. Results

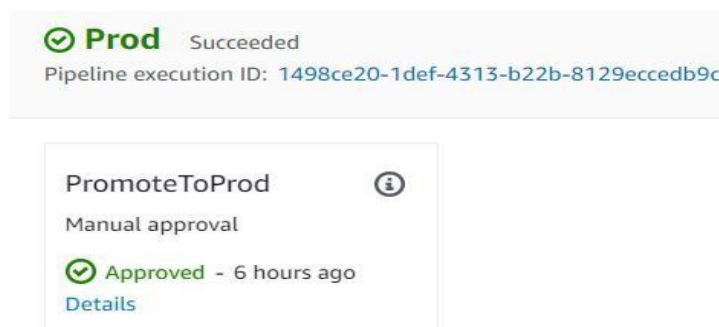


Figure 21:Production Step



## 3. 6. Task 5: Roll Back

### 3. 6. 1. Implementation

Added a roll back method to the pipeline so that whenever an error appears it should roll back to previous state and avoid the new state. This thing is done for the sake of successful pipeline setup. In this task we added a simple roll back for time error. If our lambda function takes more time than the added threshold to update than we roll it back to previous version and send the traffic to new lambda function according to our desired limit.

### 3. 6. 2. Results



Figure 22:Alias

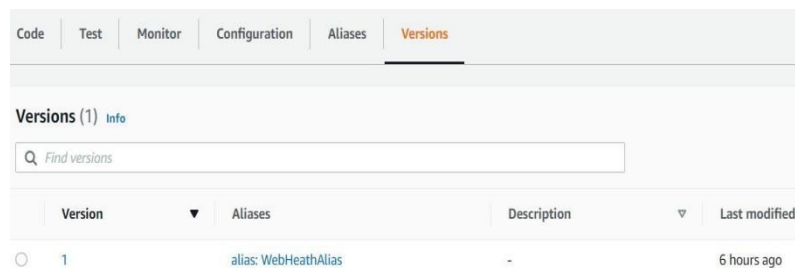


Figure 23:Versions

## 3. 7. Errors & Solution

```
error: failed to push some refs to 'https://github.com/muhammadskipq2021/ProximaCentauri.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Figure 24:Token stopped working

#### Solution:

Git hub started to reject my supplied token key . I checked and found out that it was expired, I renewed and got the process working.



```
raise JSIIError(resp.error) from JavaScriptError(resp.stack)
jsii.errors.JSIIError: Object of type @aws-cdk/core.Stack is not convertible to @aws-cdk/core.Stage
```

Figure 25:Not controvertible to stage

**Solution:**

Given wrong name to my stage file. So changed the name form stack to stage and that error got resolved.

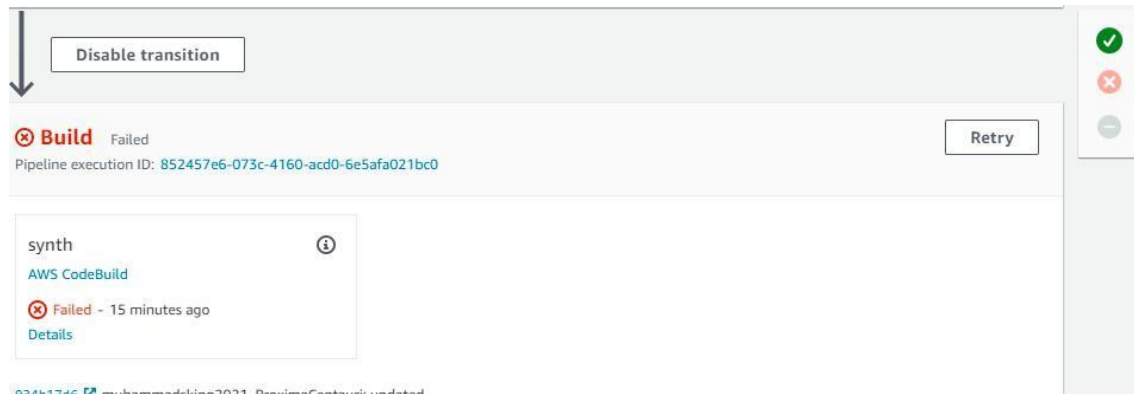


Figure 26:Build stage failed

**Solution:**

My build state started to get failed. I looked into the event logs and found out that it was an error related to wrong names specification in the code. So changed the names and got this error resolved.

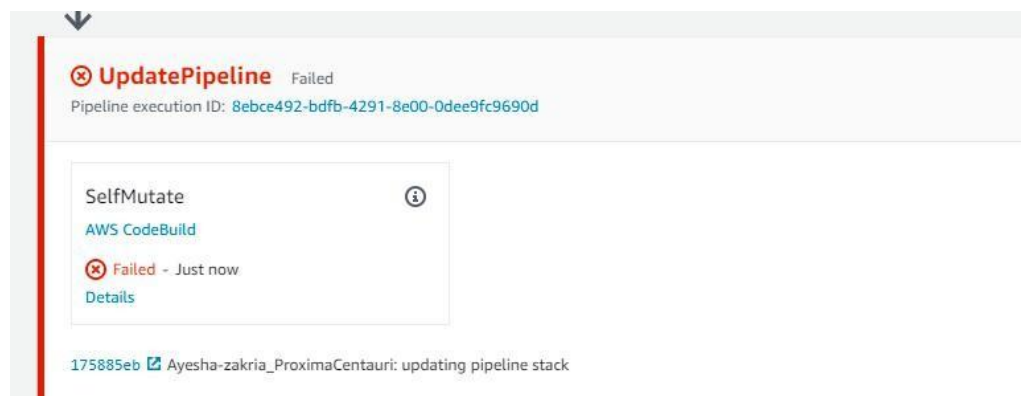


Figure 27:Update step failed

**Solution:**

Update pipeline started to give the same error. Resolved it same way.

```

inst = super().__call__(*args, **kwargs)
File "/codebuild/output/src388266275/src/Adeel/Sprint2/AdeeldynamoDB/adeeldynamo_db/adeeldynamo_db_stack.py", line
__init__
Url_Monitor = bo().bucket_as_list()
File "/codebuild/output/src388266275/src/Adeel/Sprint2/AdeeldynamoDB/resources1/bucket.py", line 7, in __init__
self.Object = boto3.client('s3').get_object(Bucket='adeelskipq',Key='urls.json')
File "/root/.pyenv/versions/3.9.5/lib/python3.9/site-packages/botocore/client.py", line 388, in _api_call
return self._make_api_call(operation_name, kwargs)
File "/root/.pyenv/versions/3.9.5/lib/python3.9/site-packages/botocore/client.py", line 708, in _make_api_call
raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (AccessDenied) when calling the GetObject operation: Access Denied
Subprocess exited with error 1
[Container] 2021/12/23 05:52:40 Command did not exit successfully cdk -a . deploy AdeelPipelineStack2 --require-approval=never --verbose exit status 1

```

Figure 28: Acces denied to Some functions

#### Solution:

It was the error related to access of different objects in pipeline environment. Added some policies in it to get access and run the code smoothly.

```

}
--app points to a cloud assembly, so we bypass synth
No stacks match the name(s) AdeelPipelineStack2
Error: No stacks match the name(s) AdeelPipelineStack2
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 05:52:24 Command did not exit successfully cdk -a . deploy AdeelPipelineStack2 --require-approval=never --verbose exit status 1
[Container] 2021/12/23 05:52:24 Phase complete: BUILD State: FAILED
[Container] 2021/12/23 05:52:24 Phase context status code: COMMAND_EXECUTION_ERROR Message: Error while executing cdk -a . deploy AdeelPipelineStack2 --require-approval=never --verbose. Reason: exit status 1

```

Figure 29:No Module name found

#### Solution:

Pipeline failed to detect my stack file. It was the error related to git hub push. I have pushed the latest changes to github and got it working.

```

AdeeldynamoDBStack(app, AdeeldynamoDBStack ,
File "/home/ec2-user/environment/AdeeldynamoDB/.venv/lib64/python3.7/site-packages/jsii/_runtime.py", line 86, in __call__
inst = super().__call__(*args, **kwargs)
File "/home/ec2-user/environment/AdeeldynamoDB/adeeldynamo_db/adeeldynamo_db_stack.py", line 46, in __init__
Url_Monitor = bo('Adeelskipq').bucket_as_list('urls.json')
File "/home/ec2-user/environment/AdeeldynamoDB/resources1/bucket.py", line 10, in bucket_as_list
data = Object['body']
TypeError: 's3.Object' object is not subscriptable
Subprocess exited with error 1
(.venv) adeelshahzadskipq:~/environment/AdeeldynamoDB (master) $

```

Figure 30:Not able to subscribe to S3

#### Solution:

S3 bucket in which I have stored my links started to give errors. It was due to permission setting. So changed the permission setting and code started working.

```

Traceback (most recent call last):
  File "app.py", line 31, in <module>
    PipelineStack(app,'AdeelPipelineStack',env = core.environment(account='315997497220',r
AttributeError: module 'aws_cdk.core' has no attribute 'environment'
Subprocess exited with error 1
adeelshahzadskipq:~/environment/ProximaCentauri/Adeel/Sprint2/AdeeldynamoDB (main) $

```

Figure 31:environment error

**Solution:**

Environment started to give errors because different stages had different env to run. Deployed them in same env and solved the error.

## 4. Sprint 3

### 4.1. Technologies Used

- Api gateway
- DynamoDB
- Lambda
- S3

#### 4.1.1. API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management. API Gateway has no minimum fees or startup costs. You pay for the API calls you receive and the amount of data transferred out and, with the API Gateway tiered pricing model, you can reduce your cost as your API usage scales.

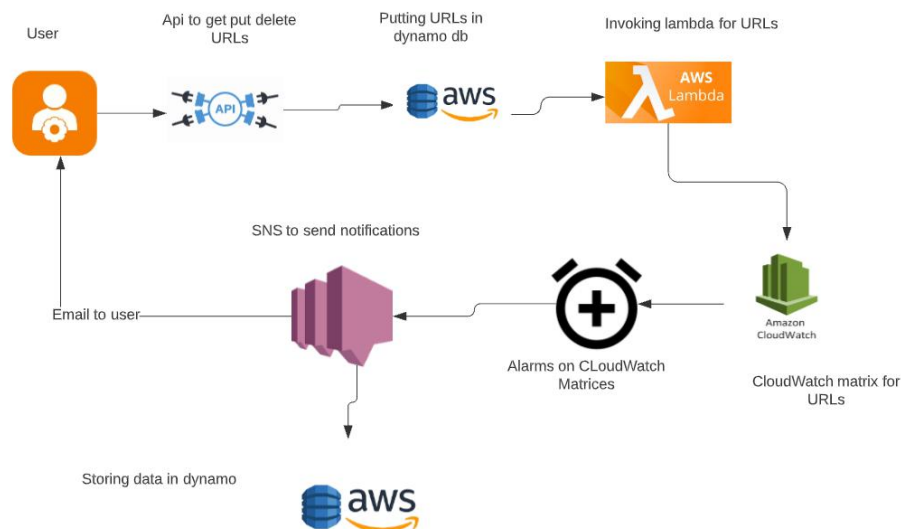
#### 4.1.2. S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.

- [S3 Lifecycle](#) – Configure a lifecycle policy to manage your objects and store them cost effectively throughout their lifecycle. You can transition objects to other S3 storage classes or expire objects that reach the end of their lifetimes.
- [S3 Object Lock](#) – Prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use Object Lock to help meet regulatory requirements that require write-once-read-many (WORM) storage or to simply add another layer of protection against object changes and deletions.
- [S3 Replication](#) – Replicate objects and their respective metadata and object tags to one or more destination buckets in the same or different AWS Regions for reduced latency, compliance, security, and other use cases.

- **S3 Batch Operations** – Manage billions of objects at scale with a single S3 API request or a few clicks in the Amazon S3 console. You can use Batch Operations to perform operations such as Copy, Invoke AWS Lambda function, and Restore on millions or billions of objects.



## 4. 2. Task 1: Moving data from S3

### 4. 2. 1. Implementation

First of all created a dynamo table to store the urls from the s3. Moved the contents of s3 from bucket to a dynamo table. Created a lambda function in my main stack. And passed it the S3 json file change as a invocation source. So whenever a S3 bucket is uploaded to s3 it will invoke the lambda file which has the tables name and read write access. It gathers all the data from bucket and puts it into the table for user to modify as he wants.

### 4. 2. 2. Results

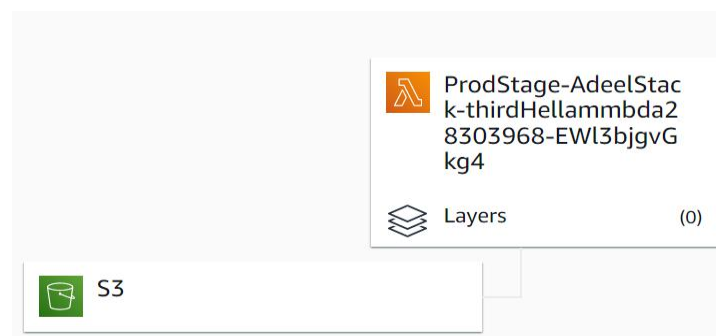


Figure 32: Lambda function with s3 trigger

<input type="checkbox"/>	Links
<input type="checkbox"/>	<a href="http://www.netflix.com">www.netflix.com</a>
<input type="checkbox"/>	<a href="http://www.amazon.com">www.amazon.com</a>
<input type="checkbox"/>	<a href="http://www.skipq.org">www.skipq.org</a>
<input type="checkbox"/>	<a href="http://www.hulu.com">www.hulu.com</a>

Figure 33: S3 values in table

```

2021-12-30T18:41:49.069+05:00    30 Dec 2021 13:41:49,069 [INFO] (/var/runtime/bootstrap.py) init complete at
2021-12-30T18:41:49.096+05:00    {"Records": [{"eventVersion": "2.1", "eventSource": "aws:s3", "awsRegion": "us-east-2", "eventTime": "2021-12-30T13:41:47.683Z", "eventName": "ObjectCreated:Put", "userIdentity": {"principalId": "AWS:AIDAUTEXLE6CC6XP2UTRG"}, "requestParameters": {"sourceIPAddress": "119.160.98.246"}, "responseElements": {"x-amz-request-id": "5W1ATBMTFPXWZH", "x-amz-id-2": "zT3Ud9L2AdI/5oHdgJCBc2syuhXAawj1J6nHC6vA3FcyfsoB5e0GPjM4Pxp+4RLevF0sbyBLNeldRMLhwVHej1N8T2W0v/Zg91AxJ4Xi"}, "s3SchemaVersion": "1.0", "configurationId": "287c7438-345c-4d1b-9c27-b44884bf85dc", "bucket": {"name": "adeelskipq", "principalId": "A22WMZRJMLRIZ8"}, "arn": "arn:aws:s3:::adeelskipq"}, {"object": {"key": "urls.json", "size": 120, "e": "a1f74f9ada42348468d6894b4cbb657", "sequencer": "0061CDB71B9FD27B44"}}]}
2021-12-30T18:41:49.096+05:00    urls.json
2021-12-30T18:41:49.096+05:00    adeelskipq

```

Figure 34:S3 log for bucket trigger

## 4. 3. Task 2: Setting Alarms on dynamo data

### 4. 3. 1. Implementation

Then in the next task I tried to read the the data moved from s3 and at now present in dynamo table to generate alarms on this data. First of all created handler file to wrote function having to perform different operation on my table. In this task used the scan table method that scans the whole table and returns the desired column as output. The data contains my other things in addition with the URLs. So used different methods offered by python open source libraries to get a list of URLs. Passed this list of URLs to my Web health lambda and stack file to generate alarms on the data. This list is goes into the loop that covers the functions to generate Latency and Availability of given urls to plot a graph for them and if the values crosses the threshold than send an email to the user for danger.

### 4. 3. 2. Results

2022-01-01T22:23:48.657+05:00	01 Jan 2022 17:23:48,657 [INFO] (/var/runtime/bootstrap.py) init complete at epoch 1641057828657
2022-01-01T22:23:50.607+05:00	['www.netflix.com', 'www.amazon.com', 'www.skipq.org', 'www.hulu.com']
2022-01-01T22:23:50.994+05:00	['www.netflix.com', 'www.amazon.com', 'www.skipq.org', 'www.hulu.com']
2022-01-01T22:23:51.014+05:00	['www.netflix.com', 'www.amazon.com', 'www.skipq.org', 'www.hulu.com']
2022-01-01T22:23:56.553+05:00	END RequestId: f118cd1a-245e-4218-896e-5a668176b3ea

Figure 35:Table values as list

## 4. 4. Task 3: API gateway and its methods

### 4. 4. 1. Implementation

In the this task tried to create a API gateway for the easiness of user. Defined the asked methods in the API. Created a API lambda file which will invoke on every request from The API. Given the lambda access to dynamo table. So it will do operations on the dynamo table as user asks. First method is to get URLs from the table. In this case user can look at all the URLs he is using for alarms. Next method is to put data, in that method if user wants he

can put another URL in the table for alarm generation. Next is delete method that gives the user to delete any unnecessary URLs from the table So that he can save his time and money be looking at what is required. Last method is update, that updates any URLs users requests to be get updated. As in at first he was looking at the all site but now he is concerned with a certain section of that site. So he can update his previous URL and give more focus to that section.

#### 4. 4. 2. Results

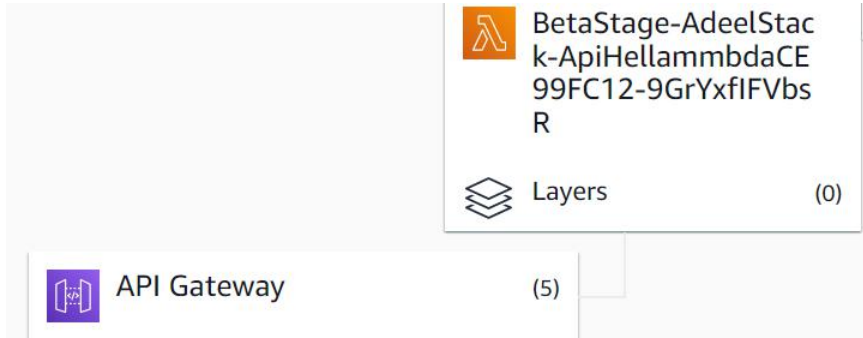


Figure 35: Lambda with API gateway trigger

Method

GET

Path

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

Request: /

Status: 200

Latency: 939 ms

Response Body

```
data from table is = ['www.netflix.com', 'www.amazon.com', 'www.skipq.org', 'www.hulu.com']
```

Figure 36: GET method results

Method

PUT

Path

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

Request: /

Status: 200

Latency: 235 ms

Response Body

```
Url = www.youtube.com is successfully added into the table
```

Response Headers

Figure 37:PUT method results



Method	Request: /
DELETE ▾	Status: 200
	Latency: 246 ms
Path	Response Body
No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.	Url= www.youtube.com is successfully deleted from the table
	Response Headers

Figure 38:DELETE method results

## 4. 5. Task 4: Extra tests

### 4. 5. 1. Implementation

In the last method added some tests in the code. So that it can be tested before it can be deployed to any environment. First of all added a test to get the read and write time it takes for dynamo table on each request of user and if time crosses a certain threshold it gives an error also added test for the methods that API is operating in.

### 4. 5. 2. Result

```

(.venv) adeelshahzadskipq:~/environment/ProximaCentauri/Adeel/Sprint3/AdeelProject3 (main) $ pytest integtests
===== test session starts =====
platform linux -- Python 3.7.10, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: /home/ec2-user/environment/ProximaCentauri/Adeel/Sprint3/AdeelProject3
collected 2 items

integtests/lambda_test.py ..                                         [100%]

===== 2 passed in 1.08s =====

```

Figure 39:Unit tests get passed

```

(.venv) adeelshahzadskipq:~/environment/ProximaCentauri/Adeel/Sprint3/AdeelProject3 (main) $ pytest unittests
===== test session starts =====
platform linux -- Python 3.7.10, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: /home/ec2-user/environment/ProximaCentauri/Adeel/Sprint3/AdeelProject3
collected 5 items

unittests/lambda_test.py .....                                     [100%]

===== 5 passed in 4.33s =====

```

Figure 40: Integration Tests

## 4. 6. Errors & Solution

```

"aws-apigateway@1.135.0.jsii.tgz",
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_runtime.py", line 43, in
_kernel.load(assembly.name, assembly.version, os.fspath(assembly_path))
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_kernel/_init_.py", line
self.provider.load(LoadRequest(name=name, version=version, tarball=tarball))
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_kernel/providers/process
return self._process.send(request, LoadResponse)
File "/home/ubuntu/.local/lib/python3.6/site-packages/jsii/_kernel/providers/process
raise JSIIError(resp.error) from JavaScriptError(resp.stack)
jsii.errors.JSIIError: ENOSPC: no space left on device, write

```

Figure 41: low space error

### Solution:

Environment started to give error about low space. Cleaned all the resources deleted all the stacks but error remained. So looked at local files and deleted the cdk.out folder and resolved the error.

```

Parameter validation failed:
Missing required parameter in input: "Key"
Unknown parameter in input: "Item", must be one of: TableName, Key, Expected, ConditionalOperator,
ReturnConsumedCapacity, ReturnItemCollectionMetrics, ConditionExpression, ExpressionAttributeName
ParamValidationError
Traceback (most recent call last):
  File "/var/task/api_lambda.py", line 32, in lambda_handler
    'Links': {'S' : new_url}
  File "/var/runtime/botocore/client.py", line 386, in _api_call
    return self._make_api_call(operation_name, kwargs)
  File "/var/runtime/botocore/client.py", line 678, in _make_api_call
    api_params, operation_model, context=request_context)
  File "/var/runtime/botocore/client.py", line 726, in _convert_to_request_dict
    api_params, operation_model)
  File "/var/runtime/botocore/validate.py", line 337, in serialize_to_request
    raise ParamValidationError(report=report.generate_report())
botocore.exceptions.ParamValidationError: Parameter validation failed:
Missing required parameter in input: "Key"

```

Figure 42: Delete key not found

#### Solution:

For my API tests delete function started to give a internal error as output message. Checked the cloud watch logs it says key is not found but I have looked in table and key was there. Looked for the syntax of delete item from table and found that I was a syntax error. As the variable item should be defined as key in this function

```

► 2021-12-30T00:46:02.215+05:00 START RequestId: 6d993ccc
► 2021-12-30T00:46:02.497+05:00 29 Dec 2021 19:46:02,215
► 2021-12-30T00:46:02.497+05:00 29 Dec 2021 19:46:02,497
▼ 2021-12-30T00:46:02.549+05:00 'key': KeyError Traceback
'key': KeyError
Traceback (most recent call last):
  File "/var/task/db_s3_lambda.py", line 13, in lambda_handler
    key = event['Records'][0]['s3']['bucket']['key']
KeyError: 'key'

```

Figure 43:Key defined wrong way

#### Solution:

S3 bucked lambda started to give the same key error and found that it also has a syntax error. To get name of object in S3 you call the event on object not the bucket.

```

START RequestId: 2dfbe508-ab48-4c8f-9ae3-38aa0034c001 Version: $LATEST
29 Dec 2021 20:30:23,231 [INFO] (/var/runtime/bootstrap.py) main started at epoch 1640809823232
29 Dec 2021 20:30:23,517 [INFO] (/var/runtime/bootstrap.py) init complete at epoch 1640809823518
END RequestId: 2dfbe508-ab48-4c8f-9ae3-38aa0034c001
REPORT RequestId: 2dfbe508-ab48-4c8f-9ae3-38aa0034c001 Duration: 372.93 ms Billed Duration: 373 ms Me
No server events at this moment. Auto-extended. Reserve

```

Figure 44: Lambda gets finished without any results

#### Solution:

As I started to working on API it was unable to show any results for any kind of request. Oped the cloudwatch logs and found out that lambda function is working fine without any error. So when i looked at lambda function it ha a return statement before the real execution.

```

29 Dec 2021 22:14:58,420 [INFO] (/var/runtime/bootstrap.py) init complete at epoch 1640809823518
Syntax error in module 'db_s3_lambda': invalid syntax (db_s3_lambda.py, line 18)
END RequestId: 28213ae6-e246-4efd-91b1-074b8d5b5435
REPORT RequestId: 28213ae6-e246-4efd-91b1-074b8d5b5435 Duration: 1.72 ms Billed Duration: 1.72 ms
START RequestId: 28213ae6-e246-4efd-91b1-074b8d5b5435 Version: $LATEST
Syntax error in module 'db_s3_lambda': invalid syntax (db_s3_lambda.py, line 18)
: invalid syntax (db_s3_lambda.py, line 18)

```

Figure 45:Invalid syntax

#### Solution:

As can be seen from cloud watch logs it was an syntax error in the lambda function. Forget to but a comma between two objects of a dictionary.



```

END RequestId: 3863482d-8dea-4454-a05c-cf504a602472
REPORT RequestId: 3863482d-8dea-4454-a05c-cf504a602472 Duration: 369.74 ms Billed Duration: 370 ms Mem...
START RequestId: 8f5a00df-4865-42de-9430-d1c633c1c1e5 Version: $LATEST
name 'event' is not defined: NameError Traceback (most recent call last): File "/var/task/api_lambda.p...
END RequestId: 8f5a00df-4865-42de-9430-d1c633c1c1e5
REPORT RequestId: 8f5a00df-4865-42de-9430-d1c633c1c1e5 Duration: 62.77 ms Billed Duration: 63 ms Memor...

```

Figure 46: Event not found

**Solution:**

Event was not defined properly in the lambda function.

## 5. Sprint 4

### 5.1. Technologies Used

- AWS amplify
- ReactJS
- AWS cognito
- API gateway

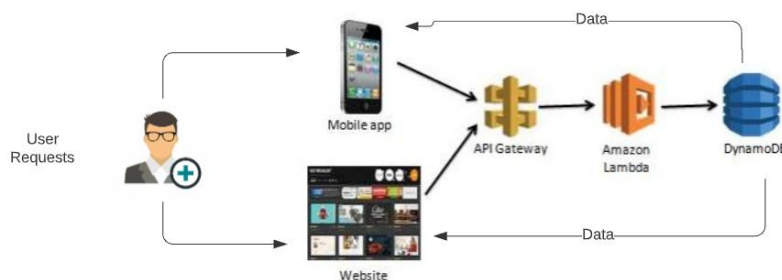
#### 5.1.1. AWS Amplify

AWS Amplify is a set of purpose-built tools and features that lets front end 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.

Amplify facilitates getting started with AWS for web and mobile app development because it is easy to use and flexible. The Amplify libraries accelerate implementation of functionality like user authentication, data storage, analytics, and predictions, using AWS services for the back-end functionality.

#### 5.1.2. AWS Cognito

Amazon Cognito is a simple user identity and data synchronization service that helps you securely manage and synchronize app data for your users across their mobile devices. You can create unique identities for your users through a number of public login providers (Amazon, Facebook, and Google) and also support unauthenticated guests. You can save app data locally on users' devices allowing your applications to work even when the devices are offline. With Amazon Cognito, you can save any kind of data in the AWS Cloud, such as app preferences or game state, without writing any backend code or managing any infrastructure. This means you can focus on creating great app experiences instead of having to worry about building and managing a backend solution to handle identity management, network state, storage, and sync.



## 5. 2. Task 1: React App

### 5. 2. 1. Implementation

This task is done by using npm libraries. Installed npm libraries using commands than added react app libraries to the local machine. After that created a react app using these libraries. If react app is created using commands it fetches all the required files with it and at local host 3000 it generates a dummy app using some styles and react logo. As of project requirements added Chakra UI libraries to local machine. Made changes to index.js and app.js file according to our project requirements. After that using npm start command observed the changes and made modifications according to debug the errors. This app has a title box after that two buttons one for search the database and other one is to show data from the database. And a search bar where user can input the url to search.

### 5. 2. 2. Results

#### API WEB INTERFACE

Here you can search URL

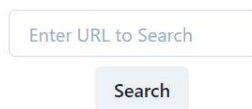


Figure 47:Web app for search

#### Show Data

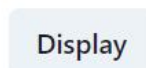


Figure 48:app for show data

## 5. 3. Task 2: API connection

### 5. 3. 1. Implementation

In this task connected the API created in sprint 3 to my react app so that user can easily see what kind of URLs he is using for alarm generation and if it is a large database he can search for his URL using each button. In my app file imported the axios library used for fetching data from different type of APIs. Used the get function to get values from my API. Provided it with invoke URL of my API and a header for what type of files to read. After that modified my lambda for API to return response with certain headers that are required for data fetching. And in my API enabled the CORs on get method also added a extension of my web browser to unblock all the CORS. After that tested my app and worked fine.

### 5. 3. 2. Results

```
{ "Access-Control-Allow-Origin": "*", "Access-Control-Allow-Methods": "GET", "Access-Control-Allow-Headers": "*", "X-Amzn-Trace-Id": "Root=1-61dc6af0-af07517e190dff071f78cc6;Sampled=0" }
```

Figure 49: API headers

Display

www.netflix.com  
www.perusall.com

- <Back
- 1
- 2
- 3
- 4
- 5
- Next>

Figure 49:URLs data in pagination

## API WEB INTERFACE

Here you can search URL

www.youtube.com

Search

Hurry!URL is found

Figure 50: URL found

## API WEB INTERFACE

Here you can search URL

www.youtube.co

Search

Sorry! The URL is not found

Figure 51: URL not found

### 5. 4. Task 3: Deploy to S3

#### 5. 4. 1. Implementation

After completing all the functions for my app, moved toward the deployment of that app so that users can access it with certain domain. To complete that task I used the AWS amplify and S3 bucket. First of all in my stack code defined a amplify app and added a branch in it with my app file in build.zip. This branch function takes data from S3 assets. To create a S3 asset used the CDK library named assets and provided that asset zip file of all build components of my react app. So amplify deploy all the files present in the S3 bucket and creates a URL for for your app. You can also provide your own domain but it should be registered and have amazon certificates which is a long process and out of the scope of this project. So used the custom app domain. I can share this URL with my users and they can see the database files and can search from them.

#### 5. 4. 2. Results

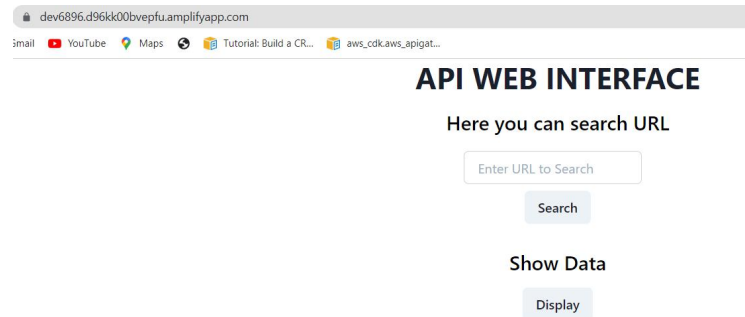


Figure 52: App with a deployed domain

## 5.5. Task 4: OAuth

### 5.5.1. Implementation

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.

### 5.5.2. Result

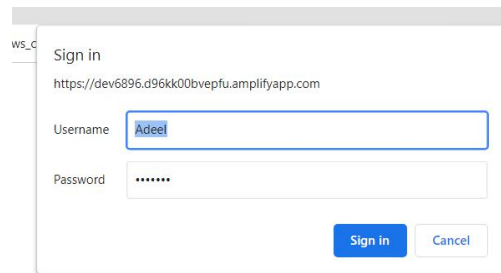


Figure 53: App OAuth

## 5.6. Errors & Solution

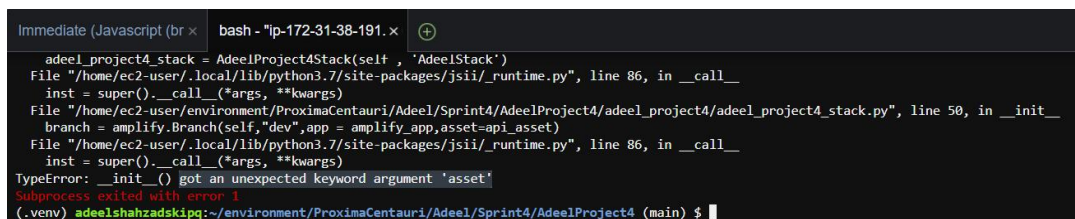


Figure 54: Asset as unexpected argument

### Solution:

I ave used amplify library to deploy the urls into S3 bucket. But it started to give error about the asset argument is not present. To deal with this error first of all created a app in code but deployed the app in that app manually.

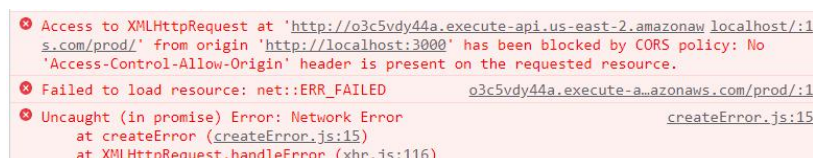


Figure 55: CORS error

**Solution:**

My fetch request get no results because CORs were not enabled in the browser. So added a CORS unblock extension into the browser and got it working.

- ✗ Add Access-Control-Allow-Origin Method Response Header to DELETE method ⚠
- ✗ Add Access-Control-Allow-Origin Integration Response Header Mapping to DELETE method ⚠

Figure 56: Failed to update cors

**Solution:**

COPS were not getting enabled in my API. So added header files to my lambda function and got it working.

```
Mon Jan 10 13:07:46 UTC 2022 : Endpoint response body before transformations: {"errorMessage": "Handler 'lambda_handler' missing on module 'api_lambda'"}
Mon Jan 10 13:07:46 UTC 2022 : Lambda execution failed with status 200 due to customer function error: Handler 'lambda_handler' missing on module 'api_lambda'. Lambda request id: 6ee40918-7c1c-44ba-ad7c-c2db1cbe013f
```

Figure 57: Lambda is missing

**Solution:**

API started to give error about lambda file not found or is missing. In lambda functions checked my lambda function and it was empty. I have mistakenly deployed the project with empty lambda. So added the lambda files and get it working..

## 6. Sprint 5

### 6.1. Technologies Used

- AWS ECS
- AWS ECR
- Docker Image
- PyRestTest

#### 6.1.1. AWS ECR

Amazon ECR private registries host your container images in a highly available and scalable architecture. You can use your private registry to manage private image repositories consisting of Docker and Open Container Initiative (OCI) images and artifacts. Each AWS account is provided with a default private Amazon ECR registry.

#### 6.1.2. AWS ECS

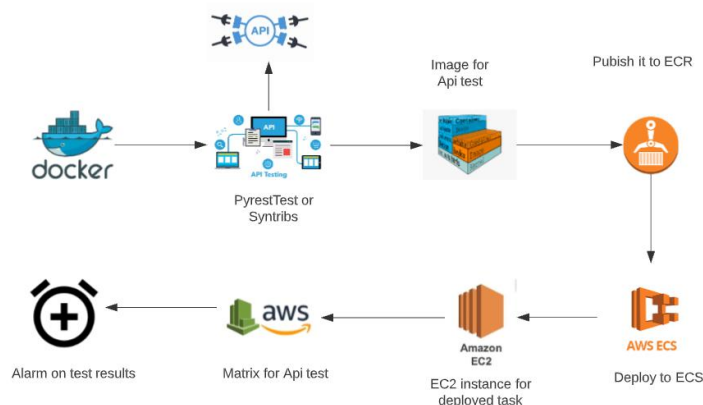
Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service. In this context, a service is a configuration that enables you to run and maintain a specified number of tasks simultaneously in a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon EC2 instances that you manage.

### 6.1.3. Docker Image

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

### 6.1.4. PyRestTest

A REST testing and API microbenchmarking tool Tests are defined in basic YAML or JSON config files, no code needed Minimal dependencies (pycurl, pyyaml, optionally future), making it easy to deploy on-server for smoketests/healthchecks



Flow Chart of Sprint 5

## 6.2. Task 1: Syntribos & PyRestTest Docker Image

### 6.2.1. Implementation

In this task implemented the docker Images for API test libraries that are OS like pyresttest and syntribos etc. First of learned how to create an image using docker. In docker images all the necessary libraries are installed inside a container and then this container can be used to run our apps in any system. To create a image for pyresttest, first copied a exiting python image from docker hub and then run the commands to install all the necessary libraries in that image. Pyresttest uses libCurl library to invoke the API. After installing all the required libraries given entry point of pyresttest. When you run this image using API url and test file it will give you results for API tests passed and how much time it take to connect to API and fetch the data. Test file is yaml type in which API method and other things are defined.

### 6.2.2. Results

docker/getting-started	IN USE	latest	26d80cd96d69	about 2 months ago	28.55 MB
pyrestful	IN USE	latest	87bc7e014c66	less than a minute ago	92.79 MB

Figure 58: Docker Image for Pyresttest

```
C:\Users\92345\pyresttest>docker run --rm pyrestful https://oob9333t07.execute-api.us-east-2.amazonaws.com/prod/ github_
api_smoketest.yaml
{"failures": 0, "aggregates": [{"total_time", "mean", 0.7841986}], "group": "Default", "results": {"total_time": [0.7544
32, 0.71459, 0.795112, 0.867168, 0.789691], "size_download": [56.0, 56.0, 56.0, 56.0, 56.0]}, "name": "Basic put"}
{"failures": 0, "aggregates": [{"total_time", "mean", 1.2084034}], "group": "Default", "results": {"total_time": [1.3814
88, 0.965864, 0.928534, 1.124367, 1.641764], "size_download": [102.0, 102.0, 102.0, 102.0, 102.0]}, "name": "Basic get"}
```

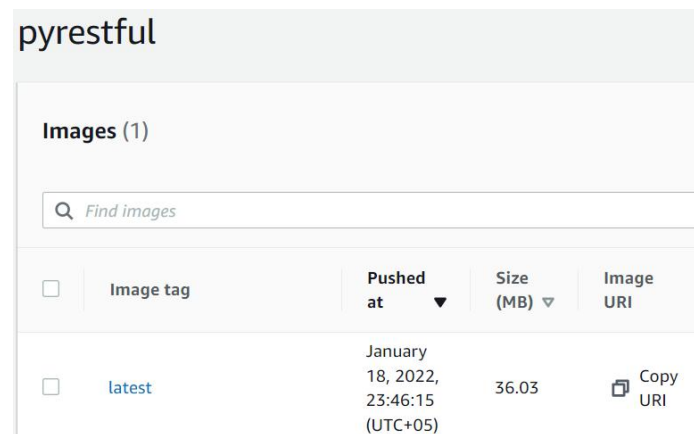
Figure 59: Results for tests on API

## 6. 3. Task 2: Publish to ECR

### 6. 3. 1. Implementation

After that published that image to our AWS registry. As image was built in local PC, to upload it first configured the AWS CLI in CMD. After configuration created a private repo on ECR and and oped the actions for pulling image on ECR. Given the repo same name as our docker image and Using pull commands uploaded the image to ECR. As image is uploaded to ECR configured it to scan for latest push and use that in our main work that will be done in later stages.

### 6. 3. 2. Results



The screenshot shows the AWS ECR console interface for a repository named 'pyrestful'. Under the 'Images (1)' section, there is a search bar labeled 'Find images'. Below it is a table with columns: 'Image tag', 'Pushed at', 'Size (MB)', and 'Image URI'. A single image is listed with the tag 'latest', pushed on January 18, 2022, at 23:46:15 (UTC+05), with a size of 36.03 MB. A 'Copy URI' button is visible next to the image entry.


<input type="checkbox"/>	Image tag	Pushed at ▼	Size (MB) ▼	Image URI
<input type="checkbox"/>	latest	January 18, 2022, 23:46:15 (UTC+05)	36.03	 Copy URI

Figure 60: Latest image on ECR

## 6. 4. Task 3: Deploy in ECS

### 6. 4. 1. Implementation

After that deployed the image pushed to ECR on ECS. To deploy any docker image first you have to create a cluster for that service. And to create a cluster I have created a Virtual Private Cloud (VPC) with certain sub-net masks. Passed this VPC to Cluster on ECS. ECS cluster can have two types of services to create instance for your docker image. First service is amazon Fargate and second one is AWS EC2. Fargate is less complex then its predecessor but it has security issues, so we are going with second one. Defined the instance type for EC2 to micro for a small image to run effectively and not to use to much space for simple task like that. Defined a task for the image and passed it to previously created service.

### 6. 4. 2. Results



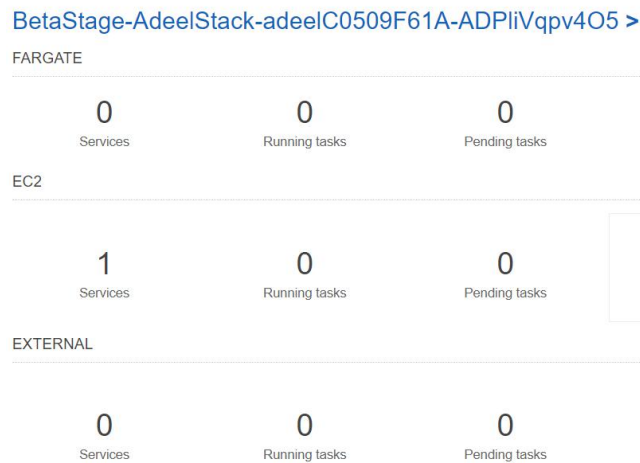


Figure 61: Running cluster

## Task definition name : AdeelPipelineStack3BetaStage

Select a revision for more details

Create new revision Actions

Status: **Active** Inactive

Filter in this page

Task Definition Name : Revision	Status
AdeelPipelineStack3BetaStageAdeelStackTaskDefB688695B:3	Active

Figure 62: Active task for deployed image

## 6. 5. Errors & Solution

```
C:\Users\92245\pyresttest>docker run --rm pyrestful https://oob9333t07.execute-api.us-east-2.amazonaws.com/prod/ github_
api_smoketest.yaml
Traceback (most recent call last):
  File "/usr/local/bin/pyresttest", line 4, in <module>
    resttest.command_line_run(sys.argv[1:])
  File "/usr/local/lib/python2.7/site-packages/pyresttest/resttest.py", line 911, in command_line_run
    main(args)
  File "/usr/local/lib/python2.7/site-packages/pyresttest/resttest.py", line 815, in main
    test_structure = read_test_file(test_file)
  File "/usr/local/lib/python2.7/site-packages/pyresttest/resttest.py", line 174, in read_test_file
    teststruct = yaml.safe_load(read_file(path))
  File "/usr/local/lib/python2.7/site-packages/pyresttest/resttest.py", line 300, in read_file
    with open(path, "r") as f:
IOError: [Errno 2] No such file or directory: 'github_api_smoketest.yaml'
```

Figure 63: Test file not found

### Solution:

When I tried to run the image after building it. It shows the error for no test file present. This error occur due to no test file in image directory copied the test file and got that error solved.

```
>cd hello-world
hello-world>(Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin 315997497220.dkr.ecr.us-east-2.amazonaws.com
his time.
```

Figure 64: Login failed



**Solution:**

Failed to log into AWS for the upload of docker image. Searched through documentation and found that it was happening due to no AWS configuration. So configured the AWS in CMD and resolved it.

```
__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:
    cdk synth. Reason: exit status 1
282 [Container] 2022/01/18 15:30:16 Entering phase POST_BUILD
```

Figure 65: Wrong Argument

**Solution:**

Due to wrong argument name this error appeared. Through documentation given it right name and solved the error.

```
... (link ecsputimagecontainer) (link ecsputimagecontainer) and (link ecsputimagecontainer)
This API will be removed in the next major release.
[WARNING] @aws-cdk/aws-ecs.Cluster#addAutoScalingGroup is deprecated.
Use (@link Cluster.addAsgCapacityProvider) instead.
This API will be removed in the next major release.
['www.netflix.com', 'www.prt.com', 'www.amazon.com', 'www.skipq.org', 'www.myprt.com', 'www.hulu.com']
[Error at /AdeelPipelineStack3/BetaStage/AdeelStack] You are not authorized to perform this operation.
Found errors
```

Figure 66: Not authorized

**Solution:**

In pipeline build stage failed to run due to permission errors found out the siutable permissions and passed then to stack file for smooth run.