**Software Engineer for Cloud Project 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Instructions** |  |  |  |  |  |
| Every screenshot requested in this workbook is compulsory and carries 5 marks. Each of (EC2, Dynamodb, Kinesis, and SNS) setup in Task 1 will carry 20 marks each. In Task 2, S3 setup via Cloudformation will carry 20 marks. Also, complete codedeploy setup will also carry 20 marks. In Task 3, complete Lambda setup via Cloudformation will carry 20 marks. | | | | | |
| Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed. | | | | | |
| All screenshots must be in the order mentioned under "Expected Screenshots" for every step | | | | | |
| DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances. | | | | | |
| The file should be renamed in the format BATCH\_FIRSTNAME\_LASTNAME\_PROJECT1.  For example: ACSEOCT20\_VIJAY\_DWIVEDI\_PROJECT2.docx | | | | | |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Resource Clean Up** | |  |  |  |  |
| Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you’ve completed your lab or project is always necessary. This is true whether you’re doing a lab or implementing a project at your workplace. | | | | | |
| After completing the lab, make sure to delete each resource created in reverse chronological order. | | | | | |

# **Anomaly Detection using CloudFormation and CodeDeploy**

## **Introduction**

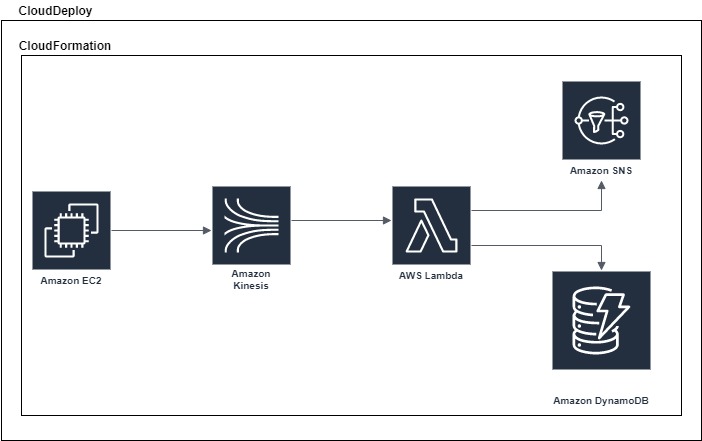
The anomaly detection M03P01 where AWS services used were all provisioned manually by using the AWS console. In the mentioned Project, we have configured services such as Kinesis, SNS, EC2 instance creation, setting up lambda handler, setting up the kinesis trigger, pushing data in DynamoDB etc. Once the services are up and running and data is pushed, the deployed lambda handler will perform anomaly detection, and it will trigger the SNS notification and detected data will be pushed on DynamoDB.

As we are moving more and more towards task automation. It makes sense if we can automate the service setup used in Anomaly detection Project using CloudFormation and CloudDeploy. The idea here is to slowly and gradually automate all the tasks involved in the M03P01. That is why the basic dummy python script which pushes data over cloud will be provided. The lambda handler code that performs anomaly detection will also be provided as a file. The focus is on automated setup, no changes are expected to be made in the actual python code which is a simple functioning dummy example, except for SNS topic ARN.  
  
Expected Kinesis data stream name - m03p02\_raw\_data\_stream

Expected Dynamodb anomaly table name - m03p02\_anomaly\_data

Expected SNS topic name - m03p02\_anomaly\_alerts

1. **Raw data (Python script):**
   1. This file is named as dummy\_temp\_data.py will generate the random environment temperature data using the random library.
   2. This python file will publish the data over Kinesis stream, from there data will be pulled by lambda service.
2. **Anomaly detection (Lambda Handler):**
   1. This anomaly\_detection.py python file is configured to perform the anomaly detection on the raw data.
   2. After anomaly detection, data (detected) will be pushed in DB and a notification will be generated using SNS arn.



## **Problem Statement**

In this project we will be doing the step by step automation of Anomaly detection given in the M03P01. This project will essentially be using Kinesis, SNS, EC2, S3, DynamoDB and lambda handler. In each step some of the services will be provisioned through Cloud formation, which in turn will automate the mentioned tasks. In the next step, a few more services such as CodeDeploy will be provisioned to increase the automation part. In the last Task, the lambda handler part will be provisioned by using cloud formation, this will in turn make the whole process automated starting from trivial service enabling to CodeDeploy and then doing anomaly detection.

## **Task Organization**

This project is divided into three tasks: **Easy, Moderate and Hard**. Please note that if you plan to complete all three tasks, then everything will be automated. If you finish only till Task 1 or Task 2 in your final solution, then you should implement the remaining items manually so that you have a running system. Please find more details related to each task below:

**Task - 1 (Easy)**

1. Provision the services (EC2, kinesis, dynamodb, SNS) using cloud formation.
2. Setup the ingestion python script in the EC2 instance manually.
3. Create the lambda handler function from lambda handler service manually through AWS console. Copy the lambda handler code and make changes in SNS arn and Table name.
4. The entire setup for anomaly detection should work i.e notification should be generated and detected data should be pushed in DynamoDB

**Task - 2 (Moderate)**

1. Provision S3 service using Cloud formation, store the original application of Task-1 and the ***appspec.yml*** here.
2. Deploy the contents of S3 using CodeDeploy into the EC2 instance at the same location where it was running earlier.
3. Lambda handler will still be there and deployed manually from AWS console
4. The entire setup for anomaly detection should work i.e notification should be generated and detected data should be pushed in DynamoDB

**Task - 3 (Hard)**

1. Provision the lambda handler service using cloud formation
2. Remove the older lambda handler function from Task-2, edit the python code in newly created lambda handler service.
3. The entire setup for anomaly detection should work i.e notification should be generated and detected data should be pushed in DynamoDB

**Submission Files**

**Task - 1**

1. Cloud template file that has JSON script for
   1. Created Linux EC2 machine configuration
   2. Created Kinesis stream configuration
   3. Created SNS configuration along with Topic & subscription name
   4. Created DynamoDB table configuration

**Task - 2**

1. Cloud template file that has JSON script:
   1. Created S3 service configuration
2. CodeDeploy file:
   1. After successfully uploading appec.yml file

**Task - 3**

1. Cloud template file that has JSON script:
   1. Lambda handler configuration that enables lambda services for the existing application
2. CodeDeploy file:
   1. After successfully uploading appspec.yml file

## **Reference Links**

1. AWS EC2 CloudFormation developer guide: [EC2 Instance creation CF user guide](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html)
2. AWS Kinesis CloudFormation developer guide: [Kinesis Stream CF user guide](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-kinesis-stream.html)
3. AWS Lambda CloudFormation developer guide: [Lambda Handler CF user guide](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-function.html)
4. AWS SNS CloudFormation developer guide:
   1. Topic creation: [Topic CF user guide](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-sns-topic.html)
   2. Subscription: [Subscription CF user guide](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-sns-subscription.html)
5. AWS DynamoDB CloudFormation developer guide: [DynamoDB table CF user guide](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-dynamodb-table.html)

# **Evaluation Rubric**

1. All images at the image place-holders: 85 Marks
2. Creation of EC2, Kinesis, DynamoDB, SNS: 80 Marks
3. Creation of S3 and CodeDeploy: 40 Marks
4. Creation of Lambda handler and Python script: 20 Marks

**Total**: **225 Marks**

# 

# **Task List**

|  |  |
| --- | --- |
| **Architecture Implementation (Task 1)** | |
| 1 | Cloud formation engine page, Take screenshot of the summary page after successful operation |
| 2 | Enabled EC2 instance |
| 3 | Enabled Kinesis stream |
| 4 | Enabled SNS arn |
| 5 | Enabled DynamoDB table |

|  |  |
| --- | --- |
| **Architecture Implementation (Task 2)** | |
| 1 | Cloud formation engine page, screenshot of the summary page after successful operation |
| 2 | Enabled S3 service |
| 3 | Uploaded files over S3 directory |
| 4 | Successful CodeDeploy operation summary age |

|  |  |
| --- | --- |
| **Architecture Implementation (Task 3)** | |
| 1 | Cloud formation engine page, screenshot of the summary page after successful operation |
| 2 | Enabled lambda handler service |
| 3 | CodeDeploy summary page after successful operation |

# **Tasks with image place-holders**

## Task 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Step number | a |  |  |
| Step name | Cloud formation summary page |  |  |
| Instructions | 1. Template file (JSON script) to enable mentioned services in the Task -1 |  |  |
| Expected screenshots | 1. Cloud formation summary page |  |  |

**<A screenshot of a computer

Description automatically generated>**

|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Enabled EC2 instance |  |
| Instructions | 1. Make sure that cloud formation operation is successful 2. Check if the EC2 services are successful, if yes, check for the created instance on EC2 console |  |
| Expected screenshots | 1. Created EC2 instance |  |

**<**A screenshot of a computer

Description automatically generated**>**

|  |  |  |
| --- | --- | --- |
| Step number | c |  |
| Step name | Enabled Kinesis stream |  |
| Instructions | 1. Make sure that cloud formation operation is successful 2. Check if the kinesis services are successful, if yes, check for the created data stream on kinesis console |  |
| Expected screenshots | 1. Create kinesis data stream |  |

**<A screenshot of a computer

Description automatically generated>**

|  |  |
| --- | --- |
| Step number | d |
| Step name | Enabled SNS arn |
| Instructions | 1. Make sure that cloud formation operation is successful 2. Check if the SNS service are successful, if yes, goto SNS console and create a subscription for the created topic/arn |
| Expected screenshots | 1. Created SNS arn 2. Created subscription and accepted subscription through mail |

**<A screenshot of a computer

Description automatically generated>**

**<A screenshot of a computer

Description automatically generated>**

## 

## 

## Task - 2:

|  |  |  |
| --- | --- | --- |
| Step number | a |  |
| Step name | Cloud formation summary page |  |
| Instructions | 1. Template file (JSON script) to enable mentioned services in Task -1 and Task-2 |  |
| Expected screenshots | 1. Cloud formation summary page |  |

**<A screenshot of a computer

Description automatically generated>**

|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Enabled S3 service |  |
| Instructions | 1. Make sure that cloud formation operation is successful 2. Check if the S3 service are successful, if yes, goto S3 console and assure the created directory is available 3. Upload the relevant scripts and appsec.yml |  |
| Expected screenshots | 1. Enabled empty S3 service 2. S3 directory after uploading the application 3. S3 directory after uploading appsec.yml |  |

**< Bucket Created after Cloud formation template**

**A screenshot of a computer

Description automatically generated>**

**<**

**A screenshot of a computer

Description automatically generated**

**Screen 1: Showing DeploymentPackage.zip**

**NOTE: DeploymentPackage.zip consists Scripts Folder, Appspec.yml and RawDataService.zip. I made single zip file**

**>**

**<A screenshot of a computer

Description automatically generated>**

**Screen 2: Showing DeploymentPackage.zip contents**

|  |  |  |
| --- | --- | --- |
| Step number | c |  |
| Step name | CodeDeploy |  |
| Instructions | 1. Perform the CodeDeploy operation through AWS console 2. If CodeDeploy is successful look if the Anomaly detection related data is going into the relevant DB and SNS services |  |
| Expected screenshots | 1. Successful CodeDeploy operation summary page 2. Anomaly data pushed in the DynamoDB 3. SNS notification |  |

**<A screenshot of a computer

Description automatically generated>**

**<A screenshot of a computer

Description automatically generated>**

**<Insert Screenshot c(3) here>**

## Task - 3:

|  |  |  |
| --- | --- | --- |
| Step number | a |  |
| Step name | Cloud formation summary page |  |
| Instructions | 1. Template file (JSON script) to enable mentioned services in all three tasks |  |
| Expected screenshots | 1. Cloud formation summary page |  |

**<A screenshot of a computer

Description automatically generated>**

|  |  |  |
| --- | --- | --- |
| Step number | b |  |
| Step name | Enabled lambda handler service |  |
| Instructions | 1. Make sure that cloud formation operation is successful 2. Check if the lambda handler service is successfully enabled |  |
| Expected screenshots | 1. Lambda handler |  |

**<**

**A screenshot of a computer

Description automatically generated>**

**Lambda Handler Screen 1 after it automatically captured from S3[new] , I removed manual operation**

**A screenshot of a computer

Description automatically generated**

**Lambda Handler Screen 2**

**A screenshot of a computer

Description automatically generated**

**Lambda Handler Screen 3 ‘CloudWatch Logs‘ showing all operations successful**

|  |  |  |
| --- | --- | --- |
| Step number | c |  |
| Step name | CodeDeploy |  |
| Instructions | 1. Perform the CodeDeploy operation through AWS console 2. If CodeDeploy is successful look if the Anomaly detection related data is going into the relevant DB and SNS services |  |
| Expected screenshots | 1. Successful CodeDeploy operation summary page 2. Anomaly data pushed in the DynamoDB 3. SNS notification |  |

**<A screenshot of a computer

Description automatically generated>**

**CodeDeploy Secreen 1 – Summary Page**

**A screenshot of a computer

Description automatically generated**

**CodeDeploy Secreen 2 – DeploymentPackage.zip**

**<A screenshot of a computer

Description automatically generated>**

**<A screenshot of a computer

Description automatically generated>**

Additional Screens and notes:

* I also automated at step 3 getting Lambda Code from another S3 bucket at CloudFormation template , no manual operation exits

A screenshot of a computer

Description automatically generated

* I used single Zip file having deployment package.