

Software Engineer for Cloud Project 2

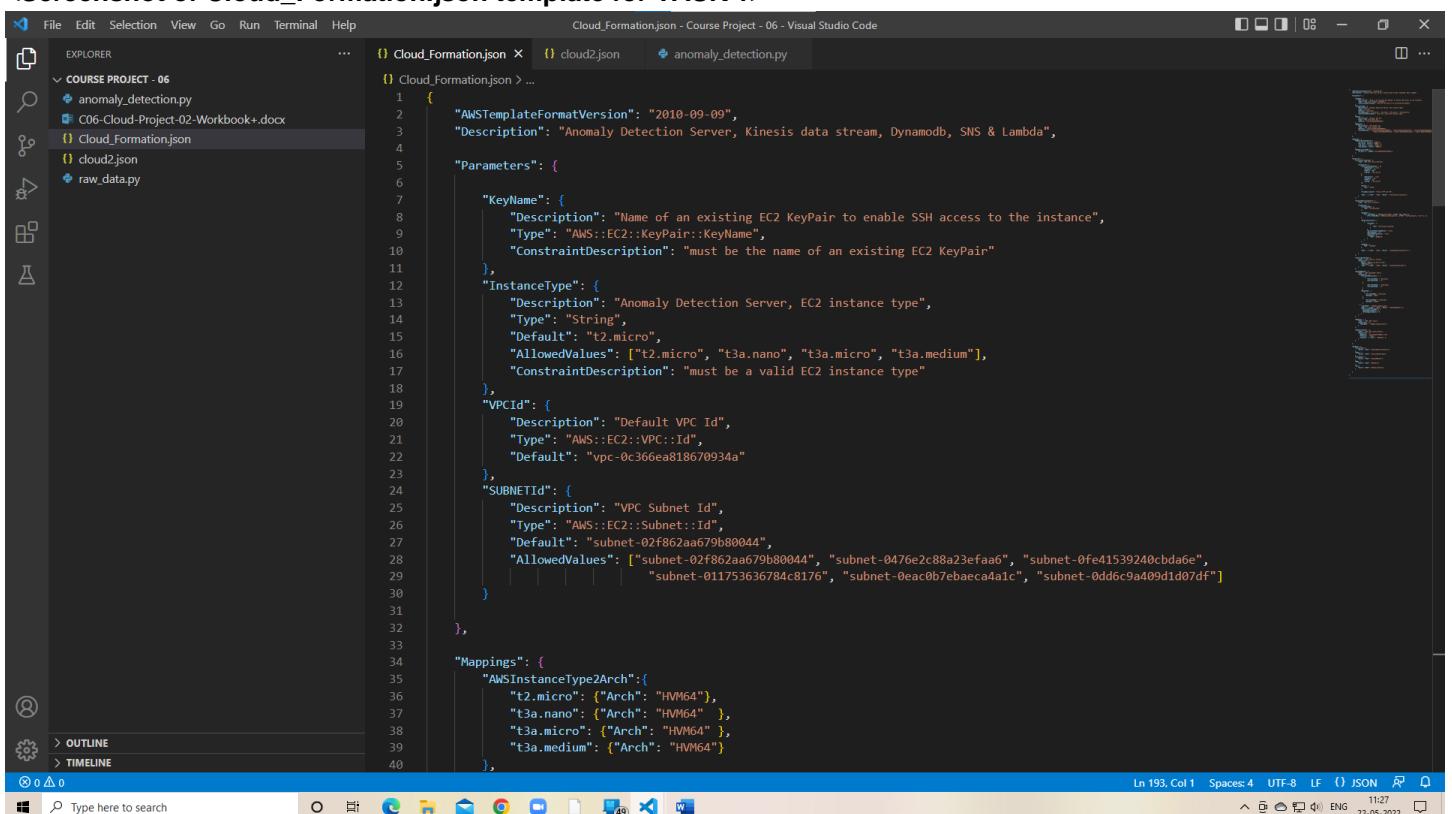
Anomaly Detection using CloudFormation and Codedeploy

Tasks with Screenshots

Task 1 Screenshots:

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

<Screenshot of Cloud_Formation.json template for TASK 1>



The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "COURSE PROJECT - 06" folder, including "anomaly_detection.py", "C06-Cloud-Project-02-Workbook+.docx", "Cloud_Forematation.json", "cloud2.json", and "raw_data.py".
- Code Editor:** Displays the content of the "Cloud_Forematation.json" file.
- Content:** The JSON template defines parameters, mappings, and AWS instance type mappings.

```
1 {  
2     "AWSTemplateFormatVersion": "2010-09-09",  
3     "Description": "Anomaly Detection Server, Kinesis data stream, Dynamodb, SNS & Lambda",  
4     "Parameters": {  
5         "KeyName": {  
6             "Description": "Name of an existing EC2 KeyPair to enable SSH access to the instance",  
7             "Type": "AWS::EC2::KeyPair::KeyName",  
8             "ConstraintDescription": "must be the name of an existing EC2 KeyPair"  
9         },  
10        "InstanceType": {  
11            "Description": "Anomaly Detection Server, EC2 instance type",  
12            "Type": "String",  
13            "Default": "t2.micro",  
14            "AllowedValues": ["t2.micro", "t3a.nano", "t3a.micro", "t3a.medium"],  
15            "ConstraintDescription": "must be a valid EC2 instance type"  
16        },  
17        "VPCId": {  
18            "Description": "Default VPC Id",  
19            "Type": "AWS::EC2::VPC::Id",  
20            "Default": "vpc-0c366ea818670934a"  
21        },  
22        "SUBNETId": {  
23            "Description": "VPC Subnet Id",  
24            "Type": "AWS::EC2::Subnet::Id",  
25            "Default": "subnet-02f862aa679b80044",  
26            "AllowedValues": ["subnet-02f862aa679b80044", "subnet-0476e2c88a23efaa6", "subnet-0fe41539240cbda6e",  
27                "subnet-011753636784c8176", "subnet-0eac0b7ebaeca4a1c", "subnet-0dd6c9a409d1d07df"]  
28        },  
29    },  
30    "Mappings": {  
31        "AWSInstanceType2Arch": {  
32            "t2.micro": {"Arch": "HVM64"},  
33            "t3a.nano": {"Arch": "HVM64" },  
34            "t3a.micro" : { "Arch": "HVM64" },  
35            "t3a.medium": {"Arch": "HVM64"}  
36        }  
37    }  
38}  
39},  
40}
```

- Status Bar:** Shows line 193, column 1, spaces: 4, UTF-8, LF, JSON, and a timestamp of 22-05-2022 11:27.

<Screenshot of CloudFormation.json template for TASK 1>

This screenshot shows the Visual Studio Code interface with the CloudFormation.json file open. The code defines a ServerSecurityGroup resource with two ingress rules (TCP ports 80 and 22) and a VPC ID reference. It also includes a GroupDescription and Tags.

```
34     "Mappings": {
35         "AWSInstanceType2Arch": {
36             "t2.micro": {"Arch": "HVM64"},
37             "t3a.nano": {"Arch": "HVM64" },
38             "t3a.micro": {"Arch": "HVM64" },
39             "t3a.medium": {"Arch": "HVM64" }
40         },
41         "AWSRegionArch2AMI":{
42             "us-east-1": {"HVM64":"ami-005de95e8ff495156"}
43         }
44     },
45
46
47     "Resources": {
48         "ServerSecurityGroup": {
49             "Type": "AWS::EC2::SecurityGroup",
50
51             "Properties": {
52                 "SecurityGroupIngress": [
53                     {
54                         "IpProtocol" : "tcp",
55                         "FromPort" : 80,
56                         "ToPort" : 80,
57                         "CidrIp" : "0.0.0.0/0"
58                     },
59                     {
60                         "IpProtocol" : "tcp",
61                         "FromPort" : 22,
62                         "ToPort" : 22,
63                         "CidrIp" : "0.0.0.0/0"
64                     }
65                 ],
66                 "VpcId": {
67                     "Ref": "VPCId"
68                 },
69                 "GroupDescription": "Allows HTTP and SSH",
70                 "Tags" : [ {"Key": "Name", "Value" : "ServerSecurityGroup"} ]
71             }
72         }
73     }
74 }
```

<Screenshot of CloudFormation.json template for TASK 1>

This screenshot shows the Visual Studio Code interface with the CloudFormation.json file open. The code defines an AnomalyDetectionServer resource. It uses AWS::FindInMap to resolve Region and InstanceType. It also defines NetworkInterfaces with a GroupSet and subnet associations, and includes a KeyName and Tags.

```
70         "Tags" : [ {"Key": "Name", "Value" : "ServerSecurityGroup"} ]
71     }
72 },
73 },
74
75     "AnomalyDetectionServer": {
76         "Type": "AWS::EC2::Instance",
77
78         "Properties": {
79             "InstanceType": {
80                 "Ref": "InstanceType"
81             },
82
83             "ImageId": {
84                 "Fn::FindInMap": [ "AWSRegionArch2AMI", {"Ref": "AWS::Region"}, {"Fn::FindInMap": [ "AWSInstanceType2Arch", {"Ref": "InstanceType"}, "Arch" ] } ]
85             },
86
87             "NetworkInterfaces": [
88                 {
89                     "GroupSet": [
90                         {
91                             "Ref": "ServerSecurityGroup"
92                         }
93                     ],
94                     "AssociatePublicIpAddress": "true",
95                     "DeviceIndex": "0",
96                     "DeleteOnTermination": "true",
97                     "SubnetId": {
98                         "Ref": "SUBNETId"
99                     }
100                }
101            ],
102
103            "KeyName": {
104                "Ref": "KeyName"
105            },
106
107            "Tags" : [ {"Key": "Name", "Value" : "AnomalyDetectionServer"} ]
108        }
109    }
110 }
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the project: COURSE PROJECT - 06, anomaly_detection.py, C06-Cloud-Project-02-Workbook+.docx, CloudFormation.json, cloud2.json, and raw_data.py.
- Code Editor:** Displays the CloudFormation.json file content. The code defines a KinesisDataStream and a DynamoDBTable. The KinesisDataStream is named "m03p02_raw_data_stream" with 2 shards and a tag for "KinesisDataStream". The DynamoDBTable is named "m03p02_anomaly_data" with a HASH key for "deviceid" and a RANGE key for "timestamp". It has a provisioned throughput of 5 units for both read and write.
- Bottom Status Bar:** Shows the current line (Ln 191), column (Col 1), and encoding (UTF-8). It also includes icons for search, file operations, and JSON format.

<Screenshot of CloudFormation.json template for TASK 1>

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the project: COURSE PROJECT - 06, anomaly_detection.py, C06-Cloud-Project-02-Workbook+.docx, CloudFormation.json, cloud2.json, and raw_data.py.
- Code Editor:** Displays the updated CloudFormation.json file content. It now includes an SNS topic named "m03p02_anomaly_alerts" and a subscription to it from the email "marystephie1@gmail.com". The "Outputs" section provides references to the InstanceId, KinesisDataStream, and DynamoDBTable.
- Bottom Status Bar:** Shows the current line (Ln 191), column (Col 1), and encoding (UTF-8). It also includes icons for search, file operations, and JSON format.

<Screenshot of a (1) Cloud Formation summary page >

<Creation of “m03p02stack” successful >

The screenshot shows the AWS CloudFormation console with the 'Events' tab selected for the stack 'm03p02anomalystack'. The left sidebar shows a single stack entry: 'm03p02anomalystack' (2022-05-19 18:03:00 UTC+0530) with status 'CREATE_COMPLETE'. The main content area displays a table of events with columns: Timestamp, Logical ID, Status, and Status reason. The table lists 20 events, all of which are 'CREATE_COMPLETE' except for one 'CREATE_IN_PROGRESS' event for 'MySubscription' which is noted as 'Resource creation initiated'. A search bar at the top of the table allows filtering by event name.

Timestamp	Logical ID	Status	Status reason
2022-05-19 18:03:52 UTC+0530	m03p02anomalystack	CREATE_COMPLETE	-
2022-05-19 18:03:50 UTC+0530	AnomalyDetectionServer	CREATE_COMPLETE	-
2022-05-19 18:03:59 UTC+0530	DynamoDBTable	CREATE_COMPLETE	-
2022-05-19 18:03:23 UTC+0530	MySubscription	CREATE_COMPLETE	-
2022-05-19 18:03:23 UTC+0530	MySubscription	CREATE_IN_PROGRESS	Resource creation initiated
2022-05-19 18:03:22 UTC+0530	MySubscription	CREATE_IN_PROGRESS	-
2022-05-19 18:03:20 UTC+0530	SNSTopic	CREATE_COMPLETE	-
2022-05-19 18:03:18 UTC+0530	AnomalyDetectionServer	CREATE_IN_PROGRESS	Resource creation initiated
2022-05-19 18:03:16 UTC+0530	KinesisDataStream	CREATE_COMPLETE	-
2022-05-19 18:03:15 UTC+0530	AnomalyDetectionServer	CREATE_IN_PROGRESS	-
2022-05-19 18:03:13 UTC+0530	ServerSecurityGroup	CREATE_COMPLETE	-

< “m03p02stack” Resources successful creation>

The screenshot shows the AWS CloudFormation console with the 'Resources' tab selected for the stack 'm03p02anomalystack'. The left sidebar shows the same stack entry as the previous screenshot. The main content area displays a table of resources with columns: Logical ID, Physical ID, Type, Status, Status reason, and Module. The table lists 6 resources, all of which are 'CREATE_COMPLETE': AnomalyDetectionServer (AWS::EC2::Instance), DynamoDBTable (AWS::DynamoDB::Table), KinesisDataStream (AWS::Kinesis::Stream), MySubscription (AWS::SNS::Subscription), SNSTopic (AWS::SNS::Topic), and ServerSecurityGroup (AWS::EC2::SecurityGroup). A search bar at the top of the table allows filtering by resource name.

Logical ID	Physical ID	Type	Status	Status reason	Module
AnomalyDetectionServer	i-09da316398c709a38	AWS::EC2::Instance	CREATE_COMPLETE	-	-
DynamoDBTable	m03p02_anomaly_data	AWS::DynamoDB::Table	CREATE_COMPLETE	-	-
KinesisDataStream	m03p02_raw_data_stream	AWS::Kinesis::Stream	CREATE_COMPLETE	-	-
MySubscription	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:de253d34-e2af-45df-b61d-24ac0956a9fc	AWS::SNS::Subscription	CREATE_COMPLETE	-	-
SNSTopic	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts	AWS::SNS::Topic	CREATE_COMPLETE	-	-
ServerSecurityGroup	sg-0c7e5a51bddadbb3d	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-	-

< “m03p02stack” Outputs summary page>

The screenshot shows the AWS CloudFormation Outputs summary page for the stack "m03p02anomalystack". The stack status is shown as "CREATE_COMPLETE". The Outputs section displays five entries:

Key	Value	Description	Export name
ARN	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts	-	-
ID	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:de253d34-e2af-45df-b61d-24acb956a9fc	-	-
InstanceId	i-09da316598c709a38	-	-
Name	m03p02_raw_data_stream	-	-
TableName	m03p02_anomaly_data	-	-

Feedback Looking for language selection? Find it in the new [Unified Settings](#).

© 2022, Amazon Internet Services Private Ltd. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Step number	b
Step name	Enabled EC2 instance
Instructions	<ol style="list-style-type: none">1) Make sure that cloud formation operation is successful2) Check if the EC2 services are successful, if yes, check for the created instance on EC2 console
Expected screenshots	<ol style="list-style-type: none">1) Created EC2 instance

< Screenshot b (1), EC2 Instance (Ubuntu server)>

The screenshot shows the AWS Management Console with the EC2 service selected. The main pane displays a table of instances. One instance is listed: AnomalyDetectionServer, with Instance ID i-09da316398c709a38, State Running, Type t2.micro, and Status check 2/2 checks passed. The Public IPv4 DNS is ec2-54-167-118-161.co... The sidebar on the left shows navigation links for EC2 Dashboard, Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store.

< EC2 Instance logged in via putty>

```
ubuntu@ip-172-31-26-69: ~
 * Support:      https://ubuntu.com/advantage
 System information as of Thu May 19 12:50:37 UTC 2022
 System load:   0.0          Processes:           94
 Usage of /:    15.5% of 7.69GB  Users logged in:     0
 Memory usage: 18%
 Swap usage:   0%
 0 updates can be applied immediately.

 The programs included with the Ubuntu system are free software;
 the exact distribution terms for each program are described in the
 individual files in /usr/share/doc/*copyright.

 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
 applicable law.

 To run a command as administrator (user "root"), use "sudo <command>".
 See "man sudo_root" for details.

ubuntu@ip-172-31-26-69:~$
```

Step number	c
Step name	Enabled Kinesis stream
Instructions	<ol style="list-style-type: none">1) Make sure that cloud formation operation is successful2) Check if the kinesis services are successful, if yes, check for the created data stream on kinesis console
Expected screenshots	<ol style="list-style-type: none">1) Create kinesis data stream

< c (1) Created Kinesis Data Stream>

The screenshot shows the AWS Amazon Kinesis Data Streams console. On the left, a sidebar menu includes 'Amazon Kinesis' (selected), 'Dashboard', 'Data streams' (selected), 'Delivery streams', 'Analytics applications', and 'Resources' (CloudFormation templates, AWS Glue Schema Registry). The main content area shows a summary message: 'New on-demand mode for Kinesis data streams. On-demand mode eliminates the requirement to manually provision and scale your data streams. With on-demand mode, your data streams automatically scale their write capacity of up to 200 MiB/second. Learn more'.

The 'Data streams' section displays a table with one row:

Name	Status	Capacity mode	Provisioned shards	Data retention period	Encryption	Consumers with enhanced fan-out
m03p02_raw_data_stream	Active	Provisioned	2	1 day	Disabled	0

At the bottom, there are links for 'Feedback', 'Unified Settings', 'Privacy', 'Terms', and 'Cookie preferences'.

< c (1) Created Kinesis Data Stream>

The screenshot shows the AWS Amazon Kinesis Data Streams console for the specific stream 'm03p02_raw_data_stream'. The sidebar is identical to the previous screenshot. The main content area shows the 'Data stream summary' and 'Monitoring' tab selected.

Data stream summary:

Status: Active	Capacity mode: Provisioned	ARN: arn:aws:kinesis:us-east-1:929150012839:stream/m03p02_raw_data_stream	Creation time: May 19, 2022, 18:03 GMT+5:30
Data retention period: 1 day			

Monitoring:

The 'Monitoring' tab displays two line charts: 'GetRecords - sum (MB/s)' and 'GetRecords iterator age - maximum (Milliseconds)'. Both charts show 'No data available.' with a note to 'Try adjusting the dashboard time range.' The x-axis for both charts ranges from 10:00 to 12:30. The y-axis for 'GetRecords - sum (MB/s)' ranges from 0 to 1. The y-axis for 'GetRecords iterator age - maximum (Milliseconds)' ranges from 0 to 1.

At the bottom, there are links for 'Feedback', 'Unified Settings', 'Privacy', 'Terms', and 'Cookie preferences'.

Step number	d
Step name	Enabled SNS arn
Instructions	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 1. Created SNS arn 2. Created subscription and accepted subscription through mail

< Screenshot d (1) created SNS arn and SNS topic m03p02_anomaly_alerts >

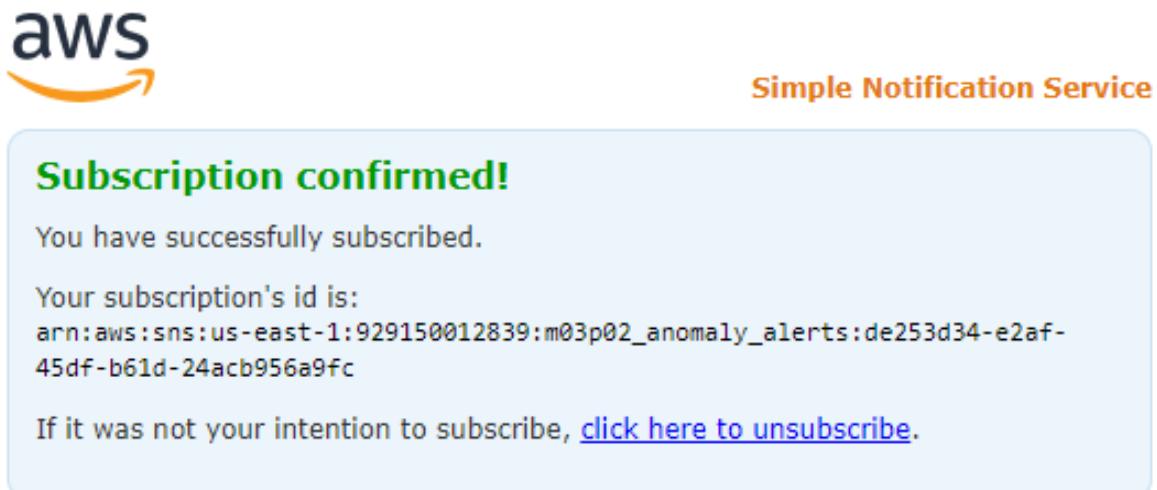
The screenshot shows the AWS Amazon SNS Topics page. The left sidebar has 'Topics' selected. The main area displays a table of topics:

Name	Type	ARN
POLICY_a51122ca-acdf-4a24-8145-264a1ca5cc66_NL_708968e2-e85e-4591-a107-c8e8d895eb21_Suspend	Standard	arn:aws:sns:us-east-1:929150012839:POLICY_a51122ca-acdf-4a24-8145-264a1ca5cc66_NL_708968e2-e85e-4591-a107-c8e8d895eb21_Suspend
m03p02_anomaly_alerts	Standard	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts

< SNS topic m03p02_anomaly_alerts and subscription >

The screenshot shows the AWS SNS console. In the top navigation bar, 'Amazon SNS' is selected. The main page displays the 'm03p02_anomaly_alerts' topic. The 'Details' section shows the topic's name ('m03p02_anomaly_alerts'), ARN ('arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts'), and type ('Standard'). The 'Subscriptions' tab is active, showing one pending confirmation subscription for 'marystephie1@gmail.com' via EMAIL. Other tabs include 'Access policy', 'Delivery retry policy (HTTP/S)', 'Delivery status logging', 'Encryption', and 'Tags'. At the bottom, there are links for 'Feedback', language selection, and copyright information.

<Screenshot d (2) created subscription confirmed through mail >



<Screenshot d (2) created subscription confirmed >

The screenshot shows the AWS SNS Topics page. The left sidebar has 'Topics' selected. The main area shows a topic named 'm03p02_anomaly_alerts'. Below it, the 'Subscriptions' tab is selected, showing one subscription entry:

ID	Endpoint	Status	Protocol
de253d34-e2af-45df-b61d-24acb956a9fc	marystephie1@gmail.com	Confirmed	EMAIL

<Screenshot d (2) created subscription confirmed >

The screenshot shows the AWS SNS Subscription details page for a specific subscription. The top navigation bar includes a warning about sending text messages to US destinations. The main area shows the subscription details:

ARN	Status	Protocol
arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:de253d34-e2af-45df-b61d-24acb956a9fc	Confirmed	EMAIL

Below the details, there are tabs for 'Subscription filter policy' and 'Redrive policy (dead-letter queue)'. The 'Subscription filter policy' section indicates 'No filter policy configured for this subscription.'

<Screenshot of created Dynamodb table---“m03p02_anomaly_data”>

The screenshot shows the AWS DynamoDB console. In the left sidebar, under the 'Tables' section, there is a single table named 'm03p02_anomaly_data'. The table details are as follows:

Name	Status	Partition key	Sort key	Indexes	Read capacity mode	Write capacity mode	Size	Table class
m03p02_anomaly_data	Active	deviceid (\$)	timestamp (\$)	0	Provisioned (5)	Provisioned (5)	0 bytes	DynamoDB Standard

At the bottom of the page, there are links for 'Feedback', 'Language selection', 'Privacy', 'Terms', and 'Cookie preferences'.

<Screenshot of created Dynamodb table---“m03p02_anomaly_data” EMPTY>

The screenshot shows the AWS DynamoDB Items page for the 'm03p02_anomaly_data' table. The table details are identical to the previous screenshot. The 'Scan/Query items' section shows the following configuration:

- Table or index: m03p02_anomaly_data
- Filters: None
- Run button: The 'Run' button is highlighted in orange.

The results section indicates that the query completed successfully with 0.5 read capacity units consumed and 0 items returned. The message 'The query did not return any results.' is displayed.

At the bottom of the page, there are links for 'Feedback', 'Language selection', 'Privacy', 'Terms', and 'Cookie preferences'.

<Lambda created manually for TASK 1 >

The screenshot shows the AWS Lambda console interface. At the top, a green banner indicates: "Successfully created the function anomaly_alert. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the function name "anomaly_alert" is displayed, along with a "Layers" section showing "(0)". On the right, there are buttons for "Throttle", "Copy ARN", and "Actions". The "Function overview" tab is selected. In the main area, there is a placeholder for triggers and destinations. Below the overview, tabs for "Code", "Test", "Monitor", "Configuration", "Aliases", and "Versions" are visible. The "Code" tab is active, showing the code source editor. The code editor has tabs for "File", "Edit", "Find", "View", "Go", "Tools", "Window", "Test", and "Deploy". The "Test" tab is selected. The code itself is a Python script named "lambda_function.py":

```
1 from pprint import pprint
2 import boto3
3 import json
4 import csv
5 import datetime
6 import time
7 import random
8 import base64
9 from decimal import Decimal
10 from botocore.exceptions import ClientError
11
12
13 def lambda_handler(event, context):
14     AWS_REGION = 'us-east-1'
15     #print(event)
16
17     dynamodb_res = boto3.resource('dynamodb', region_name=AWS_REGION)
18     anomaly_table = dynamodb_res.Table('m03p02_anomaly_data')
19
20     sns_client = boto3.client('sns', region_name=AWS_REGION)
21     topic_arn = "arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts"
22
23     for record in event['Records']:
24         data_point = base64.b64decode(record['kinesis']['data'])
25         data_point = str(data_point, 'utf-8')
26         pprint(data_point, sort_dicts=False)
27         data_point = json.loads(data_point)
28
29         anomaly_type = {}
30
31         if data_point["value"] <= (1.1 * float(data_point['lowest_temp'])):
32             anomaly_type = "Cold"
33         elif data_point["value"] >= (0.9 * float(data_point['highest_point'])):
34             anomaly_type = "Hot"
35
36         anomaly_data = {'deviceid': data_point["deviceid"],
```

<Lambda code deployed manually for TASK 1 >

The screenshot shows the AWS Lambda console interface, similar to the previous one but with different content. A green banner at the top says: "Successfully updated the function anomaly_alert." The "Code source" tab is selected, showing the same Python code as before:

```
1 from pprint import pprint
2 import boto3
3 import json
4 import csv
5 import datetime
6 import time
7 import random
8 import base64
9 from decimal import Decimal
10 from botocore.exceptions import ClientError
11
12
13 def lambda_handler(event, context):
14     AWS_REGION = 'us-east-1'
15     #print(event)
16
17     dynamodb_res = boto3.resource('dynamodb', region_name=AWS_REGION)
18     anomaly_table = dynamodb_res.Table('m03p02_anomaly_data')
19
20     sns_client = boto3.client('sns', region_name=AWS_REGION)
21     topic_arn = "arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts"
22
23     for record in event['Records']:
24         data_point = base64.b64decode(record['kinesis']['data'])
25         data_point = str(data_point, 'utf-8')
26         pprint(data_point, sort_dicts=False)
27         data_point = json.loads(data_point)
28
29         anomaly_type = {}
30
31         if data_point["value"] <= (1.1 * float(data_point['lowest_temp'])):
32             anomaly_type = "Cold"
33         elif data_point["value"] >= (0.9 * float(data_point['highest_point'])):
34             anomaly_type = "Hot"
35
36         anomaly_data = {'deviceid': data_point["deviceid"],
```

<Lambda trigger added (Kinesis data stream) manually for TASK 1 >

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Services' (selected), a search bar ('Search for services, features, blogs, docs, and more'), a keyboard shortcut '[Alt+S]', and account information ('N. Virginia' and 'greatlearningco1652963382161 @ 9291-5001-2839'). Below the navigation is a breadcrumb trail: 'Lambda > Functions > anomaly_alert'. The main title is 'anomaly_alert'. On the right, there are three buttons: 'Throttle', 'Copy ARN', and 'Actions'. A green success message box states: 'The trigger m03p02_raw_data_stream was successfully added to function anomaly_alert. The function is now receiving events from the trigger.' Below this, a section titled 'Function overview' has an 'Info' link. The central area shows the function name 'anomaly_alert' with a Kinesis icon, 'Layers (0)', and a button '+ Add destination'. To the left, there's a 'Triggers' section with a 'Kinesis' entry and a '+ Add trigger' button. On the right, detailed function information is listed: 'Description -', 'Last modified 1 minute ago', 'Function ARN arn:aws:lambda:us-east-1:929150012839:function:anomaly_alert', and 'Function URL' with an 'Info' link. At the bottom, tabs for 'Code', 'Test', 'Monitor', 'Configuration' (selected), 'Aliases', and 'Versions' are visible. The 'Configuration' tab has a sidebar with sections: 'General configuration', 'Triggers (1)' (selected), 'Permissions', and 'Destinations'. The 'Triggers' section shows a table with one row: 'Trigger' (with a checkbox). Buttons at the top of this table include 'C' (Create), 'Enable', 'Disable', 'Fix errors', 'Delete', and 'Add trigger'. Navigation arrows and a page number '1' are also present.

<Data push began from EC2 server (Ubuntu server) manually for TASK 1 >

<Data POPULATED in Dynamodb table “m03p02_anomaly_data” for TASK 1 >

The screenshot shows the AWS DynamoDB console. On the left, the navigation pane includes 'Dashboard', 'Tables', 'Update settings', 'Explore items' (which is selected), 'PartiQL editor', 'Backups', 'Exports to S3', and 'Reserved capacity'. Below these are sections for 'DAX' (Clusters, Subnet groups, Parameter groups, Events) and feedback links ('Tell us what you think', 'Return to the previous console experience', 'Density settings').

The main area displays the 'm03p02_anomaly_data' table details. It shows 1 item in the table. The table structure includes columns: deviceid, timestamp, anomaly..., anomaly..., and value. The data returned is as follows:

deviceid	timestamp	anomaly...	anomaly...	value
DHT_001	2022-05-19 13:12:45.901884	2022-05-19	Cold	98.2
DHT_001	2022-05-19 13:12:46.901829	2022-05-19	Cold	95.9
DHT_001	2022-05-19 13:12:47.901812	2022-05-19	Cold	91.6

<Data POPULATED in Dynamodb table “m03p02_anomaly_data” for TASK 1 >

This screenshot is identical to the one above, showing the AWS DynamoDB console with the 'Explore items' section selected. The main area displays the 'm03p02_anomaly_data' table details, showing 226 items returned. The table structure and data are the same as in the first screenshot.

deviceid	timestamp	anomaly...	anomaly...	value
DHT_001	2022-05-19 13:12:50.901820	2022-05-19	Cold	102.5
DHT_001	2022-05-19 13:12:51.901811	2022-05-19	Cold	96.5
DHT_001	2022-05-19 13:12:52.901823	2022-05-19	Cold	98.7
DHT_001	2022-05-19 13:12:53.901836	2022-05-19	Cold	96
DHT_001	2022-05-19 13:12:54.901826	2022-05-19	Cold	95.9
DHT_001	2022-05-19 13:12:55.901842	2022-05-19	Cold	98.2
DHT_001	2022-05-19 13:12:56.901817	2022-05-19	Cold	96
DHT_001	2022-05-19 13:12:57.901846	2022-05-19	Cold	100.2
DHT_001	2022-05-19 13:12:58.901835	2022-05-19	Cold	96.5
DHT_001	2022-05-19 13:12:59.901863	2022-05-19	Cold	94.8
DHT_001	2022-05-19 13:13:00.901828	2022-05-19	Cold	99
DHT_001	2022-05-19 13:13:01.901821	2022-05-19	Cold	98
DHT_001	2022-05-19 13:13:02.901752	2022-05-19	Cold	96.4
DHT_001	2022-05-19 13:13:03.901841	2022-05-19	Cold	97
DHT_001	2022-05-19 13:13:04.901837	2022-05-19	Cold	96
DHT_001	2022-05-19 13:13:05.901817	2022-05-19	Cold	96.5

<Data POPULATED in Dynamodb table “m03p02_anomaly_data” for TASK 1 >

Items returned (226)						
	deviceid	timestamp	anomalyDate	anomalyType	value	
	DHT_001	2022-05-19 13:13:22.901850	2022-05-19	Cold	95.3	
	DHT_001	2022-05-19 13:13:23.901825	2022-05-19	Cold	98	
	DHT_001	2022-05-19 13:13:24.901815	2022-05-19	Cold	101.3	
	DHT_001	2022-05-19 13:13:25.901829	2022-05-19	Cold	97.1	
	DHT_001	2022-05-19 13:13:26.901886	2022-05-19	Cold	97.7	
	DHT_001	2022-05-19 13:13:27.901838	2022-05-19	Cold	95.8	
	DHT_001	2022-05-19 13:13:28.901834	2022-05-19	Cold	95.6	
	DHT_001	2022-05-19 13:13:29.901823	2022-05-19	Cold	95.4	
	DHT_001	2022-05-19 13:13:30.901835	2022-05-19	Cold	96.4	
	DHT_001	2022-05-19 13:13:31.901807	2022-05-19	Cold	95.4	
	DHT_001	2022-05-19 13:13:32.901844	2022-05-19	Cold	98.6	
	DHT_001	2022-05-19 13:13:33.901916	2022-05-19	Cold	95.8	
	DHT_001	2022-05-19 13:13:34.901835	2022-05-19	Cold	97	
	DHT_001	2022-05-19 13:13:35.901824	2022-05-19	Cold	98.8	
	DHT_001	2022-05-19 13:13:36.901817	2022-05-19	Cold	98.1	
	DHT_001	2022-05-19 13:13:37.901896	2022-05-19	Cold	94.9	

<SNS notification via email received for TASK 1 >

Cold temperature is detected.

AWS Notifications <no-reply@sns.amazonaws.com>
to me
Anomaly value = 96.5 is detected. Detected temperature can be categorized as Cold

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_de253d34-e2af-45df-b61d-24acb956a9fc&Endpoint=marystephie1@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

Anomaly value = 102.5 is detected. Detected temperature can be categorized as Cold

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_de253d34-e2af-45df-b61d-24acb956a9fc&Endpoint=marystephie1@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

Anomaly value = 98.7 is detected. Detected temperature can be categorized as Cold

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_de253d34-e2af-45df-b61d-24acb956a9fc&Endpoint=marystephie1@gmail.com

2 New Messages Show Ignore

<SNS notification via email received for TASK 1 >

Compose

Inbox 1

Starred

Snoozed

Sent

Drafts

More

Meet

New meeting

Join a meeting

Hangouts

stephie

No recent chats Start a new one

Type here to search

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 99.0 is detected. Detected temperature can be categorized as Cold

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 96.5 is detected. Detected temperature can be categorized as Cold

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 96.0 is detected. Detected temperature can be categorized as Cold

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 96.5 is detected. Detected temperature can be categorized as Cold

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 92.5 is detected. Detected temperature can be categorized as Cold

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 96.4 is detected. Detected temperature can be categorized as Cold

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 97.3 is detected. Detected temperature can be categorized as Cold

18:46 19-05-2022

<SNS notification via email received for TASK 1 >

Compose

Inbox 1

Starred

Snoozed

Sent

Drafts

More

Meet

New meeting

Join a meeting

Hangouts

stephie

No recent chats Start a new one

Type here to search

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 95.6 is detected. Detected temperature can be categorized as Cold

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_de253d34-a2af-45df-b61d-24acb956a9fc&Endpoint=marystephie1@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 100.2 is detected. Detected temperature can be categorized as Cold

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_de253d34-a2af-45df-b61d-24acb956a9fc&Endpoint=marystephie1@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 97.3 is detected. Detected temperature can be categorized as Cold

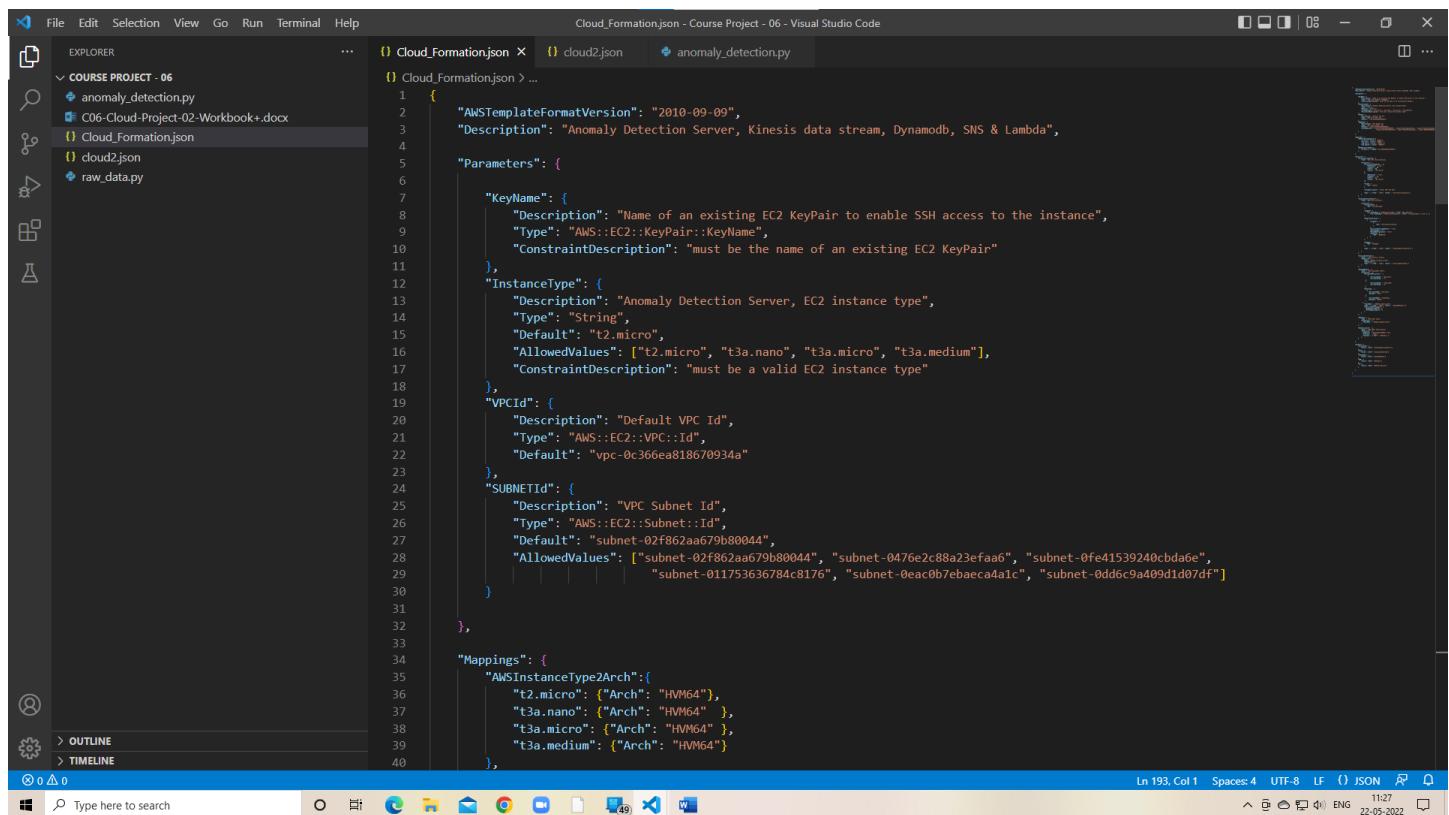
AWS Notifications <no-reply@sns.amazonaws.com> to me 6:43 PM (2 minutes ago)
Anomaly value = 96.1 is detected. Detected temperature can be categorized as Cold

18:47 19-05-2022

Task 2 Screenshots:

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

<Screenshot for CloudFormation.json template for TASK 2>



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "COURSE PROJECT - 06" folder, including "anomaly_detection.py", "C06-Cloud-Project-02-Workbook.docx", "CloudFormation.json", "cloud2.json", and "raw_data.py".
- Code Editor:** Displays the content of the "CloudFormation.json" file.
- Status Bar:** Shows "Ln 193, Col 1" and "Spaces: 4" and "UTF-8".
- Bottom Bar:** Shows the Windows taskbar with various pinned icons.

```
1  {
2      "AWSTemplateFormatVersion": "2010-09-09",
3      "Description": "Anomaly Detection Server, Kinesis data stream, Dynamodb, SNS & Lambda",
4
5      "Parameters": {
6          "KeyName": {
7              "Description": "Name of an existing EC2 KeyPair to enable SSH access to the instance",
8              "Type": "AWS::EC2::KeyPair::KeyName",
9              "ConstraintDescription": "must be the name of an existing EC2 KeyPair"
10         },
11         "InstanceType": {
12             "Description": "Anomaly Detection Server, EC2 instance type",
13             "Type": "String",
14             "Default": "t2.micro",
15             "AllowedValues": ["t2.micro", "t3a.nano", "t3a.micro", "t3a.medium"],
16             "ConstraintDescription": "must be a valid EC2 instance type"
17         },
18         "VPCId": {
19             "Description": "Default VPC Id",
20             "Type": "AWS::EC2::VPC::Id",
21             "Default": "vpc-0c366ea818670934a"
22         },
23         "SUBNETId": {
24             "Description": "VPC Subnet Id",
25             "Type": "AWS::EC2::Subnet::Id",
26             "Default": "subnet-02f862aa679b80044",
27             "AllowedValues": ["subnet-02f862aa679b80044", "subnet-0476e2c88a23efaa6", "subnet-0fe41539240cbda6e",
28                             "subnet-011753636784c8176", "subnet-0eac0b7ebaeca4a1c", "subnet-0dd6c9a409d1d07df"]
29         }
30     },
31
32     "Mappings": {
33         "AWSInstanceType2Arch": {
34             "t2.micro": {"Arch": "HVM64"}, // Line 34
35             "t3a.nano": {"Arch": "HVM64"}, // Line 35
36             "t3a.micro": {"Arch": "HVM64"}, // Line 37
37             "t3a.medium": {"Arch": "HVM64"} // Line 38
38         }
39     }
40 }
```

<Screenshot for CloudFormation.json template for TASK 2>

The screenshot shows the Visual Studio Code interface with the file 'CloudFormation.json' open. The code defines a ServerSecurityGroup with two ingress rules (TCP ports 80 and 22) and a VPC ID reference. It also defines an AnomalyDetectionServer EC2 instance with specific properties like instance type, image ID, and network interfaces.

```
File Edit Selection View Go Run Terminal Help
CloudFormation.json - Course Project - 06 - Visual Studio Code
EXPLORER
COURSE PROJECT - 06
anomaly_detection.py
C06-Cloud-Project-02-Workbook+.docx
CloudFormation.json
cloud2.json
raw_data.py
CloudFormation.json > ...
Mappings": {
    "AWSInstanceType2Arch": {
        "t2.micro": {"Arch": "HVM64"},
        "t3a.nano": {"Arch": "HVM64" },
        "t3a.micro": {"Arch": "HVM64" },
        "t3a.medium": {"Arch": "HVM64" }
    },
    "AWSRegionArch2AMI": {
        "us-east-1": {"HVM64": "ami-005de95e8ff495156" }
    }
},
Resources": {
    "ServerSecurityGroup": {
        "Type": "AWS::EC2::SecurityGroup",
        "Properties": {
            "SecurityGroupIngress": [
                {
                    "IpProtocol": "tcp",
                    "FromPort": 80,
                    "ToPort": 80,
                    "CidrIp": "0.0.0.0/0"
                },
                {
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "CidrIp": "0.0.0.0/0"
                }
            ],
            "VpcId": {
                "Ref": "VPCId"
            },
            "GroupDescription": "Allows HTTP and SSH",
            "Tags": [ {"Key": "Name", "Value": "ServerSecurityGroup"} ]
        }
    }
},
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
CloudFormation.json - Course Project - 06 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
CloudFormation.json - Course Project - 06 - Visual Studio Code
EXPLORER
COURSE PROJECT - 06
anomaly_detection.py
C06-Cloud-Project-02-Workbook+.docx
CloudFormation.json
cloud2.json
raw_data.py
CloudFormation.json > ...
Tags": [ {"Key": "Name", "Value": "ServerSecurityGroup"} ]
},
"AnomalyDetectionServer": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "InstanceType": {
            "Ref": "InstanceType"
        },
        "ImageId": {
            "Fn::FindInMap": [ "AWSRegionArch2AMI", {"Ref": "AWS::Region"}, {
                "Fn::FindInMap": [ "AWSInstanceType2Arch", {"Ref": "InstanceType"}, "Arch" ] }
            ]
        },
        "NetworkInterfaces": [
            {
                "GroupSet": [
                    {
                        "Ref": "ServerSecurityGroup"
                    }
                ],
                "AssociatePublicIpAddress": "true",
                "DeviceIndex": "0",
                "DeleteOnTermination": "true",
                "SubnetId": {
                    "Ref": "SUBNETId"
                }
            }
        ],
        "KeyName": {
            "Ref": "KeyName"
        },
        "Tags": [ {"Key": "Name", "Value": "AnomalyDetectionServer"} ]
    }
}
CloudFormation.json - Course Project - 06 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
CloudFormation.json - Course Project - 06 - Visual Studio Code
EXPLORER
COURSE PROJECT - 06
anomaly_detection.py
C06-Cloud-Project-02-Workbook+.docx
CloudFormation.json
cloud2.json
raw_data.py
CloudFormation.json > ...

```

This screenshot shows the continuation of the CloudFormation.json template. It adds a 'Tags' section to the AnomalyDetectionServer resource, which contains a single tag 'Name' with the value 'AnomalyDetectionServer'. The code editor interface and project structure remain the same as the first screenshot.

```
File Edit Selection View Go Run Terminal Help
CloudFormation.json - Course Project - 06 - Visual Studio Code
EXPLORER
COURSE PROJECT - 06
anomaly_detection.py
C06-Cloud-Project-02-Workbook+.docx
CloudFormation.json
cloud2.json
raw_data.py
CloudFormation.json > ...
Tags": [ {"Key": "Name", "Value": "AnomalyDetectionServer"} ]
},
"AnomalyDetectionServer": {
    "Type": "AWS::EC2::Instance",
    "Properties": {
        "InstanceType": {
            "Ref": "InstanceType"
        },
        "ImageId": {
            "Fn::FindInMap": [ "AWSRegionArch2AMI", {"Ref": "AWS::Region"}, {
                "Fn::FindInMap": [ "AWSInstanceType2Arch", {"Ref": "InstanceType"}, "Arch" ] }
            ]
        },
        "NetworkInterfaces": [
            {
                "GroupSet": [
                    {
                        "Ref": "ServerSecurityGroup"
                    }
                ],
                "AssociatePublicIpAddress": "true",
                "DeviceIndex": "0",
                "DeleteOnTermination": "true",
                "SubnetId": {
                    "Ref": "SUBNETId"
                }
            }
        ],
        "KeyName": {
            "Ref": "KeyName"
        },
        "Tags": [ {"Key": "Name", "Value": "AnomalyDetectionServer"} ]
    }
}
CloudFormation.json - Course Project - 06 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
CloudFormation.json - Course Project - 06 - Visual Studio Code
EXPLORER
COURSE PROJECT - 06
anomaly_detection.py
C06-Cloud-Project-02-Workbook+.docx
CloudFormation.json
cloud2.json
raw_data.py
CloudFormation.json > ...

```

<Screenshot for CloudFormation.json template for TASK 2>

This screenshot shows the Visual Studio Code interface with the CloudFormation.json file open. The code defines a KinesisDataStream and a DynamoDBTable.

```
    },
    "KinesisDataStream": {
        "Type" : "AWS::Kinesis::Stream",
        "Properties": {
            "Name": "m03p02_raw_data_stream",
            "ShardCount": 2,
            "Tags" : [ {"Key" : "Name", "Value" : "KinesisDataStream"} ]
        }
    },
    "DynamoDBTable" : {
        "Type" : "AWS::DynamoDB::Table",
        "Properties" : {
            "AttributeDefinitions" : [
                {
                    "AttributeName" : "deviceid",
                    "AttributeType" : "S"
                },
                {
                    "AttributeName" : "timestamp",
                    "AttributeType" : "S"
                }
            ],
            "KeySchema" : [
                {
                    "AttributeName": "deviceid",
                    "KeyType": "HASH"
                },
                {
                    "AttributeName": "timestamp",
                    "KeyType": "RANGE"
                }
            ],
            "TableName" : "m03p02_anomaly_data",
            "Tags" : [ {"Key" : "Name", "Value" : "DynamoDBTable"} ],
            "ProvisionedThroughput": {
                "ReadCapacityUnits": 5,
                "WriteCapacityUnits": 5
            }
        }
    }
},
```

The status bar at the bottom indicates the code is at line 191, column 1, in JSON mode, with the timestamp 11:29 22-05-2022.

This screenshot continues the Visual Studio Code interface with the CloudFormation.json file open. It adds provisions for an SNS topic, an S3 bucket, and outputs for instance ID and Kinesis Data Stream.

```
        "ProvisionedThroughput" : {
            "ReadCapacityUnits": 5,
            "WriteCapacityUnits": 5
        }
    },
    "SNSTopic" : {
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "m03p02_anomaly_alerts"
        }
    },
    "MySubscription" : {
        "Type" : "AWS::SNS::Subscription",
        "Properties" : {
            "Endpoint" : "marystephie1@gmail.com",
            "Protocol" : "email",
            "TopicArn" : { "Ref" : "SNSTopic" }
        }
    },
    "S3Bucket" : {
        "Type": "AWS::S3::Bucket",
        "DeletionPolicy": "Retain",
        "Properties": {
            "BucketName": "m03p02databucket",
            "Tags" : [ {"Key" : "Name", "Value" : "S3Bucket"} ]
        }
    },
    "Outputs": {
        "InstanceId": {
            "Value": { "Ref": "AnomalyDetectionServer" }
        },
        "Name": {
            "Value": { "Ref": "KinesisDataStream" }
        }
    }
},
```

The status bar at the bottom indicates the code is at line 203, column 2, in JSON mode, with the timestamp 11:37 22-05-2022.

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer:** COURSE PROJECT - 06, anomaly_detection.py, C06-Cloud-Project-02-Workbook+.docx, CloudFormation.json, cloud2.json, raw_data.py.
- Code Editor:** The main editor pane displays the CloudFormation JSON template code. The code defines a stack with resources like a KinesisDataStream, DynamoDBTable, SNSTopic, and S3Bucket, along with their properties and outputs.
- Bottom Status Bar:** Ln 203, Col 2, Spaces: 4, UTF-8, LF, JSON, 11:37, 22-05-2022.

< Screenshot a (1) Cloud Formation stack successfully created>

<Screenshot of Cloud Formation “m03p02stack” with s3 bucket service added for TASK 2>

The screenshot shows the AWS CloudFormation console with the following details:

- Header:** Services, Search for services, features, blogs, docs, and more, [Alt+S].
- Breadcrumbs:** CloudFormation > Stacks > m03p02stack.
- Stack Details:** m03p02stack (1 Stack). The stack status is ACTIVE.
- Events Tab:** Shows 23 events. The table includes columns: Timestamp, Logical ID, Status, and Status reason.
- Table Data:** The events listed are:

Timestamp	Logical ID	Status	Status reason
2022-05-22 11:52:09 UTC+0530	m03p02stack	CREATE_COMPLETE	-
2022-05-22 11:52:07 UTC+0530	AnomalyDetectionServer	CREATE_COMPLETE	-
2022-05-22 11:51:54 UTC+0530	DynamoDBTable	CREATE_COMPLETE	-
2022-05-22 11:51:45 UTC+0530	S3Bucket	CREATE_COMPLETE	-
2022-05-22 11:51:37 UTC+0530	MySubscription	CREATE_COMPLETE	-
2022-05-22 11:51:37 UTC+0530	MySubscription	CREATE_IN_PROGRESS	Resource creation initiated
2022-05-22 11:51:36 UTC+0530	MySubscription	CREATE_IN_PROGRESS	-
2022-05-22 11:51:35 UTC+0530	AnomalyDetectionServer	CREATE_IN_PROGRESS	Resource creation initiated
2022-05-22 11:51:34 UTC+0530	SNSTopic	CREATE_COMPLETE	-
2022-05-22 11:51:32 UTC+0530	AnomalyDetectionServer	CREATE_IN_PROGRESS	-
- Bottom Navigation:** Feedback, Looking for language selection? Find it in the new Unified Settings, © 2022, Amazon Internet Services Private Ltd. or its affiliates., Privacy, Terms, Cookie preferences.

<Screenshot of “m03p02stack” successful “Resources” for TASK 2>

m03p02stack

Resources (7)

Logical ID	Physical ID	Type	Status	Status reason	Module
AnomalyDetectionServer	i-0f0079ed45c779bf2	AWS::EC2::Instance	CREATE_COMPLETE	-	-
DynamoDBTable	m03p02_anomaly_data	AWS::DynamoDB::Table	CREATE_COMPLETE	-	-
KinesisDataStream	m03p02_raw_data_stream	AWS::Kinesis::Stream	CREATE_COMPLETE	-	-
MySubscription	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:c13ae94a-bacd-4277-a6ff-b12b45ead5ca	AWS::SNS::Subscription	CREATE_COMPLETE	-	-
S3Bucket	m03p02databucket	AWS::S3::Bucket	CREATE_COMPLETE	-	-
SNSTopic	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts	AWS::SNS::Topic	CREATE_COMPLETE	-	-
ServerSecurityGroup	sg-002011550dee8e9e0	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-	-

<Screenshot “m03p02stack” successful “Outputs” for TASK 2>

m03p02stack

Outputs (6)

Key	Value	Description	Export name
ARN	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts	-	-
BucketName	m03p02databucket	-	-
ID	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:c13ae94a-bacd-4277-a6ff-b12b45ead5ca	-	-
InstanceId	i-0f0079ed45c779bf2	-	-
Name	m03p02_raw_data_stream	-	-
TableName	m03p02_anomaly_data	-	-

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

<Screenshots for b (1) >

<Screenshot of Enabled S3 Service, “m03p02databucket” successfully created for TASK 2>

The screenshot shows the AWS S3 Buckets page. The left sidebar includes links for Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. Under Storage Lens, there are links for Dashboards and AWS Organizations settings. A Feature spotlight section is also present. The main content area displays an Account snapshot with total storage of 21.0 B, object count of 1, and avg. object size of 21.0 B. Below this, a table lists two buckets:

Name	AWS Region	Access	Creation date
cf-templates-18xx9nkgptmtr-us-east-1			May 22, 2022, 11:50:38 (UTC+05:30)
m03p02databucket			May 22, 2022, 11:51:24 (UTC+05:30)

At the bottom of the page, there are links for Feedback, Unified Settings, and a footer with copyright information.

<Screenshot “m03p02bucket” empty for TASK 2>

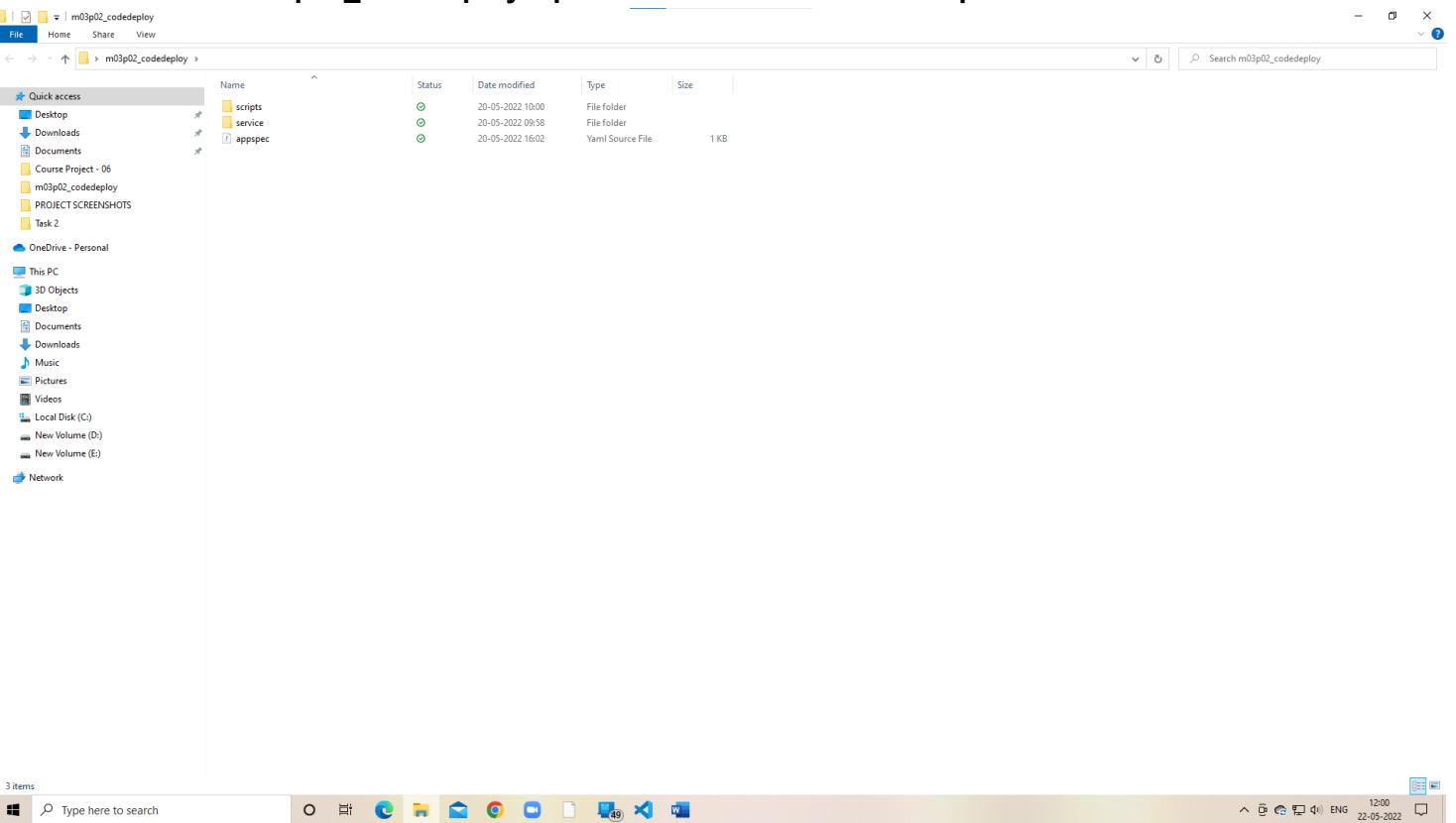
The screenshot shows the AWS S3 console interface. The left sidebar is titled "Amazon S3" and includes sections for Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. It also features a "Storage Lens" section with links to Dashboards and AWS Organizations settings, and a "Feature spotlight" section. The main content area shows the "m03p02databucket" bucket. The "Info" tab is selected. The "Objects" tab is active, showing a table with one row: "No objects". A message below the table states, "You don't have any objects in this bucket." At the bottom of the table is a large orange "Upload" button. Above the table are several actions: Copy S3 URI, Copy URL, Download, Open, Delete, Actions (with a dropdown arrow), Create folder, and Upload. A search bar labeled "Find objects by prefix" is positioned above the table. The top navigation bar includes the AWS logo, a "Services" menu, a search bar, and a global status bar indicating "greatlearningco1653200011197 @ 9291-5001-2839".

< Screenshot of first empty folder where.zip file will be uploaded for TASK 2 >

The screenshot shows the AWS S3 console interface. The left sidebar is identical to the previous screenshot, showing the "Amazon S3" section with various access and management tools. The main content area shows the "rawdatapackage/" folder within the "m03p02databucket". The "Info" tab is selected. The "Objects" tab is active, showing a table with one row: "No objects". A message below the table states, "You don't have any objects in this folder." At the bottom of the table is a large orange "Upload" button. Above the table are several actions: Copy S3 URI, Copy URL, Download, Open, Delete, Actions (with a dropdown arrow), Create folder, and Upload. A search bar labeled "Find objects by prefix" is positioned above the table. The top navigation bar includes the AWS logo, a "Services" menu, a search bar, and a global status bar indicating "greatlearningco1653200011197 @ 9291-5001-2839".

<Screenshots for b (2) and b (3) >

< Screenshot of “m03p02_codedeploy.zip” folder which needs to be uploaded in S3 bucket for TASK 2>



<Screenshot of “m03p02_codedeploy.zip” which contains both application and appspec.yml uploaded for TASK 2>

Files and folders (1 Total, 2.9 KB)					
<input type="text"/> Find by name					
Name	Folder	Type	Size	Status	Error
m03p02_codedeploy.zip	-	application/x-zip-compressed	2.9 KB	Succeeded	-

<Screenshot of “m03p02_codedeploy.zip” which contains both application and appspec.yml uploaded for TASK 2>

The screenshot shows the AWS S3 console interface. The left sidebar has a tree view with 'Amazon S3' selected. Under 'Buckets', 'm03p02databucket' is selected. The main area shows the contents of the 'rawdatapackage/' folder, which contains a single object named 'm03p02_codedeploy.zip'. The object details show it is a zip file, 2.9 KB in size, and was last modified on May 22, 2022, at 12:02:04 (UTC+05:30). The top navigation bar includes the AWS logo, services menu, search bar, and global settings.

Step number	c
Step name	CodeDeploy
Instructions	<ol style="list-style-type: none">1) Perform the CodeDeploy operation through AWS console2) If CodeDeploy is successful look if the Anomaly detection related data is going into the relevant DB and SNS services
Expected screenshots	<ol style="list-style-type: none">1) Successful CodeDeploy operation summary page2) Anomaly data pushed in the DynamoDB3) SNS notification

<Screenshot c (1)>

<Screenshot of CodeDeploy application creation for TASK 2 >

The screenshot shows the AWS CodeDeploy application creation interface. The left sidebar navigation includes: Developer Tools, CodeDeploy (selected), Source (CodeCommit), Artifacts (CodeArtifact), Build (CodeBuild), Deploy (CodeDeploy, selected), Getting started, Deployments, Applications (Application selected), Application settings, Deployment configurations, On-premises instances, Pipeline (CodePipeline), and Settings. The main content area displays the following information:

- Application created**: In order to create a new deployment, you must first create a deployment group.
- m03p02_anomaly_detection**: The application name.
- Application details**: Name (m03p02_anomaly_detection) and Compute platform (EC2/On-premises).
- Deployment groups**: A table showing one entry: "No deployment groups". A note below states: "Before you can deploy your application using CodeDeploy, you must create a deployment group." A "Create deployment group" button is present.
- Navigation and Footer**: Includes links for "Feedback", "Language selection", "Privacy", "Terms", and "Cookie preferences".

<Screenshot of CodeDeploy application Group creation for TASK 2 >

The screenshot shows the AWS CodeDeploy application group creation interface. The left sidebar navigation is identical to the previous screenshot. The main content area displays the following information:

- Success**: Deployment group created.
- m03p02_anomaly_detection_group**: The deployment group name.
- Deployment group details**:
 - Deployment group name: m03p02_anomaly_detection_group
 - Application name: m03p02_anomaly_detection
 - Compute platform: EC2/On-premises
 - Deployment type: In-place
 - Service role ARN: arn:aws:iam::929150012839:role/ec2-multi-role
 - Deployment configuration: CodeDeployDefault.AllAtOnce
 - Rollback enabled: False
 - Agent update scheduler: Learn to schedule update in AWS Systems Manager
- Environment configuration: Amazon EC2 instances**:
 - Key: Name, Value: AnomalyDetectionServer
- Triggers**: A table showing one entry: Name (Events) and Type (AWS Lambda).
- Navigation and Footer**: Includes links for "Feedback", "Language selection", "Privacy", "Terms", and "Cookie preferences".

<Screenshot of CodeDeploy Successful c (1) for TASK 2 >

The screenshot shows the AWS CodeDeploy console for a successful deployment named 'd-USMRJ2XPH'. The left sidebar navigation includes 'CodeCommit', 'CodeArtifact', 'CodeBuild', 'CodeDeploy', 'CodePipeline', and 'CodePipeline' settings. The main content area displays the deployment status, deployment details, and revision details. The deployment status shows 'Installing application on your instances' with a progress bar at 100% and 1 instance updated, all succeeded. Deployment details show the application 'm03p02_anomaly_detection', deployment ID 'd-USMRJ2XPH', deployment configuration 'CodeDeployDefault.AllAtOnce', deployment group 'm03p02_anomaly_detection_group', and a deployment description: 'Deploying the raw_data.py script on AnomalyDetectionServer to detect anomalies in temperature values.' Revision details show the revision location 's3://m03p02databucket/rawdatapackage/m03p02_codedeploy.zip' and the revision created '1 minute ago'. The revision description is 'Application revision registered by Deployment ID: d-USMRJ2XPH'. The deployment status is marked as 'Succeeded'.

<Screenshot of CodeDeploy Successful c (1) for TASK 2 >

The screenshot shows the AWS CodeDeploy console for a successful deployment named 'd-USMRJ2XPH'. The left sidebar navigation includes 'CodeCommit', 'CodeArtifact', 'CodeBuild', 'CodeDeploy', 'CodePipeline', and 'CodePipeline' settings. The main content area displays the deployment details and revision details. Deployment details show the application 'm03p02_anomaly_detection', deployment ID 'd-USMRJ2XPH', deployment configuration 'CodeDeployDefault.AllAtOnce', deployment group 'm03p02_anomaly_detection_group', and a deployment description: 'Deploying the raw_data.py script on AnomalyDetectionServer to detect anomalies in temperature values.' Revision details show the revision location 's3://m03p02databucket/rawdatapackage/m03p02_codedeploy.zip' and the revision created '1 minute ago'. The revision description is 'Application revision registered by Deployment ID: d-USMRJ2XPH'. Below this, a table lists deployment events: ApplicationStop, DownloadBundle, BeforeInstall, Install, AfterInstall, ApplicationStart, and ValidateService, all with a status of 'Succeeded' and a duration of less than one second. The deployment status is marked as 'Succeeded'.

<Screenshot c (2) anomaly data in Dynamodb table for TASK 2 >

DynamoDB > Items > m03p02_anomaly_data

m03p02_anomaly_data

Scan/Query items

Scan Query

Table or index: m03p02_anomaly_data

Completed Read capacity units consumed: 3.5

Items returned (279)

	deviceid	timestamp	anomalyDate	anomalyType	value
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:34.823237	2022-05-22	Cold	95.6
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:35.823216	2022-05-22	Cold	100.8
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:36.823240	2022-05-22	Cold	97.4

<Screenshot c (2) anomaly data in Dynamodb table for TASK 2 >

DynamoDB > Items > m03p02_anomaly_data

Items returned (279)

	deviceid	timestamp	anomalyDate	anomalyType	value
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:40.823241	2022-05-22	Cold	99.4
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:41.823248	2022-05-22	Cold	96.8
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:42.823248	2022-05-22	Cold	96.9
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:43.823235	2022-05-22	Cold	98.1
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:44.823222	2022-05-22	Cold	101.1
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:45.823220	2022-05-22	Cold	95.1
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:46.823231	2022-05-22	Cold	99.6
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:47.823218	2022-05-22	Cold	96.5
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:48.823228	2022-05-22	Cold	92.7
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:49.823259	2022-05-22	Cold	95.4
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:50.823253	2022-05-22	Cold	95.3
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:51.823250	2022-05-22	Cold	94.6
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:52.823210	2022-05-22	Cold	97.3
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:53.823235	2022-05-22	Cold	97.4
<input type="checkbox"/>	DHT_001	2022-05-22 06:54:54.823228	2022-05-22	Cold	97.8

<Screenshot c (2) anomaly data in Dynamodb table for TASK 2 >

The screenshot shows the AWS DynamoDB console. On the left, there's a sidebar with options like Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, Reserved capacity, DAX, Clusters, Subnet groups, Parameter groups, and Events. The main area shows a table titled 'Items returned (339)'. The table has columns: deviceid, timestamp, anomalyDate, anomalyType, and value. The data shows multiple entries for deviceid 'DHT_001' with various timestamp values and cold anomalies.

deviceid	timestamp	anomalyDate	anomalyType	value
DHT_001	2022-05-22 06:59:34.823229	2022-05-22	Cold	97.6
DHT_001	2022-05-22 06:59:35.823219	2022-05-22	Cold	95.9
DHT_001	2022-05-22 06:59:36.823227	2022-05-22	Cold	96.2
DHT_001	2022-05-22 06:59:37.823214	2022-05-22	Cold	97.8
DHT_001	2022-05-22 06:59:38.823286	2022-05-22	Cold	95.6
DHT_001	2022-05-22 06:59:39.823227	2022-05-22	Cold	95.1
DHT_001	2022-05-22 06:59:40.823216	2022-05-22	Cold	99.6
DHT_001	2022-05-22 06:59:41.823223	2022-05-22	Cold	95.5
DHT_001	2022-05-22 06:59:42.823209	2022-05-22	Cold	95.9
DHT_001	2022-05-22 06:59:43.823236	2022-05-22	Cold	95.8
DHT_001	2022-05-22 06:59:44.823229	2022-05-22	Cold	96.5
DHT_001	2022-05-22 06:59:45.823226	2022-05-22	Cold	95

<Screenshot c (3) SNS Notification for TASK 2 >

The screenshot shows the Gmail inbox. There are three new messages from 'AWS Notifications'. The first message says 'Cold temperature is detected.' The second message says 'Anomaly value = 96.9 is detected. Detected temperature can be categorized as Cold'. The third message says 'Anomaly value = 95.1 is detected. Detected temperature can be categorized as Cold'. Each message includes an unsubscribe link and a support contact link.

M Gmail

Compose

Inbox 1

Starred

Snoozed

Sent

Drafts

More

Meet

New meeting

Join a meeting

Hangouts

No recent chats

Start a new one

Search mail

1 of 2

12:24 PM (1 minute ago)

AWS Notifications <no-reply@sns.amazonaws.com> to me

Anomaly value = 95.6 is detected. Detected temperature can be categorized as Cold

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts.c13ae94a-bacd-4277-a6ff-b12b45ead5ca&Endpoint=marystephie1@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

12:24 PM (1 minute ago)

AWS Notifications <no-reply@sns.amazonaws.com> to me

Anomaly value = 96.9 is detected. Detected temperature can be categorized as Cold

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts.c13ae94a-bacd-4277-a6ff-b12b45ead5ca&Endpoint=marystephie1@gmail.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

12:24 PM (1 minute ago)

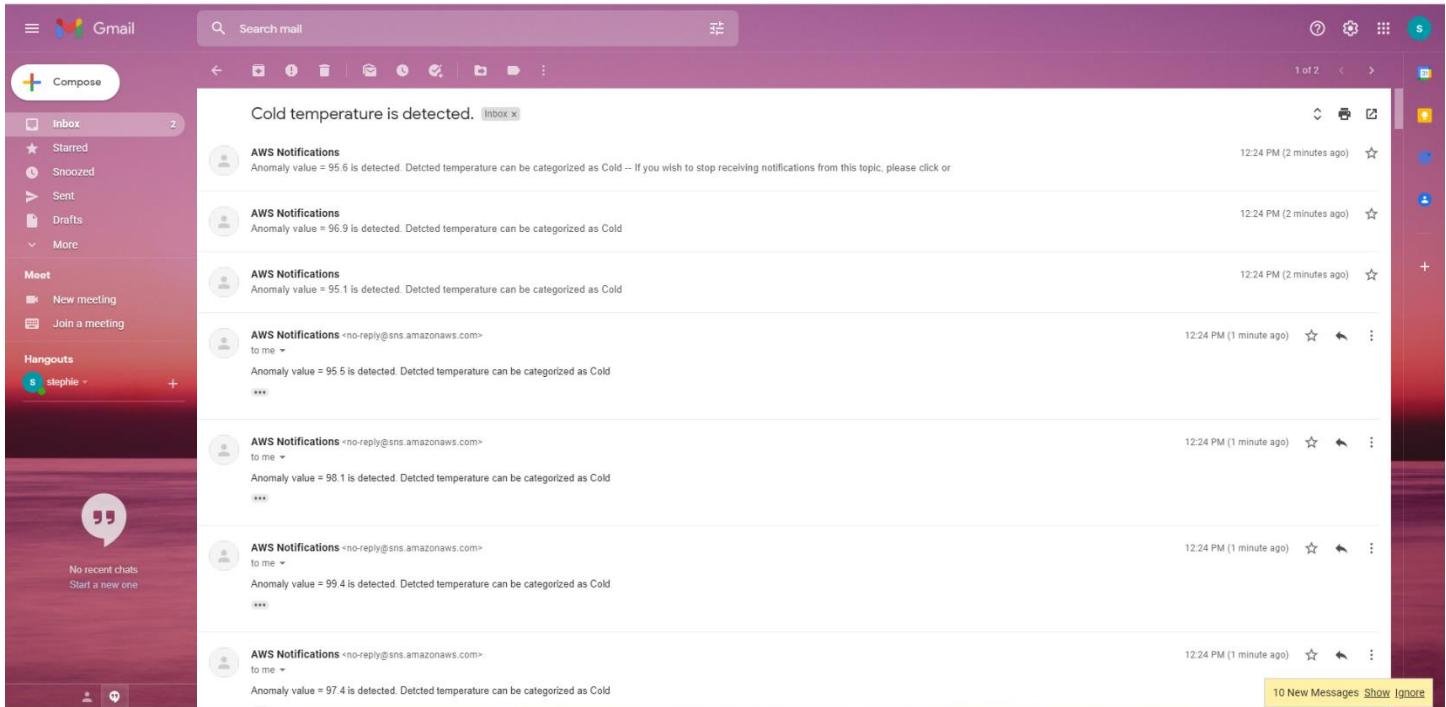
AWS Notifications <no-reply@sns.amazonaws.com> to me

Anomaly value = 95.1 is detected. Detected temperature can be categorized as Cold

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts.c13ae94a-bacd-4277-a6ff-b12b45ead5ca&Endpoint=marystephie1@gmail.com

6 New Messages Show Ignore

<Screenshot c (3) SNS Notification for TASK 2 >



Task 3 Screenshots:

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

<Screenshot of CloudFormation.json template creation for TASK 3>

The screenshot shows the Visual Studio Code interface with the file 'CloudFormation.json' open. The code defines parameters for an EC2 instance, including KeyName, InstanceType, VPCId, and SubnetId. It also includes Mappings for AWSInstanceType2Arch and AWSRegionArch2AMI, and a Resources section for a ServerSecurityGroup.

```
1 "AWSTemplateFormatVersion": "2010-09-09",
2 "Description": "Anomaly Detection Server, Kinesis data stream, Dynamodb, SNS & Lambda",
3
4 "Parameters": [
5
6     "KeyName": {
7         "Description": "Name of an existing EC2 KeyPair to enable SSH access to the instance",
8         "Type": "AWS::EC2::KeyPair::KeyName",
9         "ConstraintDescription": "must be the name of an existing EC2 KeyPair"
10    },
11    "InstanceType": {
12        "Description": "Anomaly Detection Server, EC2 instance type",
13        "Type": "String",
14        "Default": "t2.micro",
15        "AllowedValues": ["t2.micro", "t3a.nano", "t3a.micro", "t3a.medium"],
16        "ConstraintDescription": "must be a valid EC2 instance type"
17    },
18    "VPCId": {
19        "Description": "Default VPC Id",
20        "Type": "AWS::EC2::VPC::Id",
21        "Default": "vpc-0c366ea818670934a"
22    },
23 ],
24 "Mappings": {
25     "AWSInstanceType2Arch": {
26         "t2.micro": {"Arch": "HVM64"},
27         "t3a.nano": {"Arch": "HVM64" },
28         "t3a.micro": {"Arch": "HVM64" },
29         "t3a.medium": {"Arch": "HVM64" }
30     },
31 },
32
33
34 "Resources": {
35     "ServerSecurityGroup": {
36         "Type": "AWS::EC2::SecurityGroup",
37
38         "Properties": {
39             "SecurityGroupIngress": [
40                 {
41                     "IpProtocol": "tcp",
42                     "FromPort": 80,
43                     "ToPort": 80,
44                     "CidrIp": "0.0.0.0/0"
45                 },
46                 {
47                     "IpProtocol": "tcp",
48                     "FromPort": 22,
49                     "ToPort": 22,
50                     "CidrIp": "0.0.0.0/0"
51                 }
52             ],
53
54             "VpcId": {
55                 "Ref": "VPCId"
56             },
57
58             "GroupDescription": "Allows HTTP and SSH",
59
60             "Tags": [ {"Key": "Name", "Value": "ServerSecurityGroup"} ]
61         }
62     }
63 }
```

This screenshot shows the continuation of the CloudFormation.json template. It adds a new Resources section for a NetworkInterface, which is associated with the ServerSecurityGroup and has specific subnet and interface details defined.

```
34     "AWSRegionArch2AMI": {
35         "us-east-1": {"HVM64": "ami-005de95e8ff495156"}
36     }
37 },
38
39 },
40
41 },
42
43 },
44
45 },
46
47 },
48
49 },
50
51 },
52
53 },
54
55 },
56
57 },
58
59 },
60
61 },
62
63 },
64
65 },
66
67 },
68
69
70 },
71
72 },
73 }
```

<Screenshot of CloudFormation.json template creation for TASK 3>

The screenshot shows the Visual Studio Code interface with the file 'CloudFormation.json' open. The code defines an AWS Lambda function named 'AnomalyDetectionServer' with specific properties like instance type and network interfaces. It also creates a Kinesis Data Stream and a DynamoDB table.

```
CloudFormation.json - C06-Cloud-Project-02 - Visual Studio Code
```

```
CloudFormation.json > {} Parameters
```

```
70     "Tags" : [ {"Key" : "Name", "Value" : "ServerSecurityGroup"} ]  
71 },  
72 },  
73 },  
74  
75 "AnomalyDetectionServer": {  
76     "Type": "AWS::EC2::Instance",  
77  
78     "Properties": {  
79         "InstanceType": {  
80             "Ref": "InstanceType"  
81         },  
82  
83         "ImageId": {  
84             "Fn::FindInMap": [ "AWSRegionArch2AMI", {"Ref": "AWS::Region"},  
85                 {"Fn::FindInMap": [ "AWSInstanceType2Arch", {"Ref": "InstanceType"}, "Arch" ] } ]  
86         },  
87  
88         "NetworkInterfaces": [  
89             {  
90                 "GroupSet": [  
91                     {  
92                         "Ref": "ServerSecurityGroup"  
93                     }  
94                 ],  
95                 "AssociatePublicIpAddress": "true",  
96                 "DeviceIndex": "0",  
97                 "DeleteOnTermination": "true",  
98                 "SubnetId": {  
99                     "Ref": "SUBNETId"  
100                }  
101            },  
102        ],  
103  
104         "KeyName": {  
105             "Ref": "KeyName"  
106         },  
107  
108         "Tags" : [ {"Key" : "Name", "Value" : "AnomalyDetectionServer"} ]  
109     }
```

```
Ln 23, Col 11 Spaces: 4 UTF-8 LF () JSON ⌂ 2019 26-05-2022
```

```
Type here to search
```

The screenshot shows the continuation of the CloudFormation.json template. It adds more properties to the Lambda function, specifies the Kinesis Data Stream, and defines the DynamoDB table with its attribute definitions and key schema.

```
CloudFormation.json - C06-Cloud-Project-02 - Visual Studio Code
```

```
CloudFormation.json > {} Parameters
```

```
107     "Tags" : [ {"Key" : "Name", "Value" : "AnomalyDetectionServer"} ]  
108 },  
109 },  
110  
111 "KinesisDataStream":{  
112     "Type" : "AWS::Kinesis::Stream",  
113     "Properties": {  
114         "Name": "m03p02_raw_data_stream",  
115         "ShardCount": 2,  
116         "Tags" : [ {"Key" : "Name", "Value" : "KinesisDataStream"} ]  
117     }  
118 },  
119  
120 "DynamoDBTable" : {  
121     "Type" : "AWS::DynamoDB::Table",  
122     "Properties" : {  
123         "AttributeDefinitions" : [  
124             {  
125                 "AttributeName" : "deviceid",  
126                 "AttributeType" : "S"  
127             },  
128             {  
129                 "AttributeName" : "timestamp",  
130                 "AttributeType" : "S"  
131             }  
132         ],  
133         "KeySchema" : [  
134             {  
135                 "AttributeName" : "deviceid",  
136                 "KeyType": "HASH"  
137             },  
138             {  
139                 "AttributeName" : "timestamp",  
140                 "KeyType": "RANGE"  
141             }  
142         ],  
143         "TableName" : "m03p02_anomaly_data",  
144         "Tags" : [ {"Key" : "Name", "Value" : "DynamoDBTable"} ]  
145     },  
146 }
```

```
Ln 23, Col 11 Spaces: 4 UTF-8 LF () JSON ⌂ 2020 26-05-2022
```

```
Type here to search
```

<Screenshot of CloudFormation.json template creation for TASK 3>

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `CloudFormation.json`, `lambda_deployment_package.zip`, and `m03p02_codedeploy.zip`.
- Code Editor:** Displays the `CloudFormation.json` file content, which defines a Lambda function with a specific configuration.
- Bottom Status Bar:** Shows the current file is `CloudFormation.json`, line 23, column 11, with 4 spaces, in UTF-8 format, and the file was last saved on 26/05/2022 at 20:20.

```
CloudFormation.json - C06-Cloud-Project-02 - Visual Studio Code
```

```
CloudFormation.json
Parameters
  "TableName" : "m03p02_anomaly_data",
  "Tags" : [ {"Key" : "Name", "Value" : "DynamoDBTable"} ],
  "ProvisionedThroughput": {
    "ReadCapacityUnits": 5,
    "WriteCapacityUnits": 5
  }
},
"SNSTopic" : {
  "Type" : "AWS::SNS::Topic",
  "Properties" : {
    "TopicName" : "m03p02_anomaly_alerts"
  }
},
"MySubscription" : {
  "Type" : "AWS::SNS::Subscription",
  "Properties" : {
    "Endpoint" : "marystephiel@gmail.com",
    "Protocol" : "email",
    "TopicArn" : { "Ref" : "SNSTopic" }
  }
},
"S3Bucket": {
  "Type": "AWS::S3::Bucket",
  "DeletionPolicy": "Retain",
  "Properties": {
    "BucketName": "m03p02databucket",
    "Tags" : [ {"Key" : "Name", "Value" : "S3Bucket"} ]
  }
},
"KinesisEventSourceMap": {
  "Type" : "AWS::Lambda::EventSourceMapping",
  "Properties" : {
    "BatchSize" : 50,
    "MaximumBatchingWindowInSeconds" : 20
  }
}
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under "C06-CLOUD-PROJECT-02".
- Cloud Formation.json:** The active file is a CloudFormation JSON template. It defines a Lambda function named "AnomalyLambdaFunction" with a specific configuration. The function is triggered by a Kinesis event source map.
- Code Editor:** The main pane displays the JSON code for the Lambda function's properties and triggers.
- Status Bar:** Shows the current line (Ln 23), column (Col 11), and other settings like spaces, UTF-8, LF, JSON, and a search icon.

```
CloudFormation.json - C06-Cloud-Project-02 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

CloudFormation.json x

C06-CLOUD-PROJECT-02
CloudFormation.json
lambda_deployment_package.zip
m03p02_codedeploy.zip
README.txt

CloudFormation.json > {} Parameters
1/8
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217

    "KinesisEventSourceMap": {
        "Type" : "AWS::Lambda::EventSourceMapping",
        "Properties" : {
            "BatchSize" : 50,
            "MaximumBatchingWindowInSeconds" : 30,
            "EventSourceArn" : {
                "Fn::GetAtt": [
                    "KinesisDataStream",
                    "Arn"
                ]
            },
            "FunctionName": {
                "Fn::GetAtt": [
                    "AnomalyLambdaFunction",
                    "Arn"
                ]
            },
            "StartingPosition": "LATEST"
        }
    },
    "AnomalyLambdaFunction": {
        "Type": "AWS::Lambda::Function",
        "Properties": {
            "Description": "Detects anomalies. Stores them in dynamodb table & sends sns notification to subscriber",
            "FunctionName": "anomalydetection",
            "PackageType": "Zip",
            "Handler": "anomaly_detection.lambda_handler",
            "Role": "arn:aws:iam::929150012839:role/LambdaExecutionRole",
            "Code": {
                "S3Bucket": "m03p02-codedeploy",
                "S3Key": "lambda_function.zip"
            }
        }
    }
},
```

<Screenshot of CloudFormation.json template creation for TASK 3>

The screenshot shows the Visual Studio Code interface with the file 'CloudFormation.json' open. The code defines a Lambda function named 'AnomalyLambdaFunction' with specific properties like runtime and tracing mode. It also includes a 'Tags' array and an 'Outputs' section mapping various AWS resources by name.

```
196     "Arn": "arn:aws:lambda:us-east-1:123456789012:function:AnomalyLambdaFunction"
197   },
198   "StartingPosition": "LATEST"
199 }
200 },
201 },
202 },
203 },
204 },
205 },
206 },
207 },
208 },
209 },
210 },
211 },
212 },
213 },
214 },
215 },
216 },
217 },
218 },
219 },
220 },
221 },
222 },
223 },
224 },
225 },
226 },
227 },
228 },
229 },
230 },
231 },
232 },
233 },
234 },
235 },
236 },
237 },
238 },
239 },
240 },
241 },
242 },
243 },
244 },
245 },
246 },
247 },
248 },
249 },
250 },
251 },
252 },
253 },
254 },
255 },
256 },
257 },
258 },
259 },
260 },
261 }
```

The screenshot shows the Visual Studio Code interface with the file 'CloudFormation.json' open. The code has been updated to include a 'TracingConfig' section and a 'Tags' array. The 'Outputs' section remains largely the same, mapping various AWS resources by name.

```
226   "TracingConfig": {
227     "Mode": "Active"
228   },
229   "Tags": [
230     {
231       "Key": "Name",
232       "Value": "AnomalyLambdaFunction"
233     }
234   },
235 },
236 },
237 },
238 },
239 },
240 },
241 },
242 },
243 },
244 },
245 },
246 },
247 },
248 },
249 },
250 },
251 },
252 },
253 },
254 },
255 },
256 },
257 },
258 },
259 },
260 },
261 }
```

<Screenshot a (1) Cloud “m03p02stack” successful creation for TASK 3>

The screenshot shows the AWS CloudFormation console with the stack named "m03p02stack". The "Events" tab is selected, displaying 29 events. All events are in the "CREATE_COMPLETE" status, indicating successful creation. The table includes columns for Timestamp, Logical ID, Status, and Status reason.

Timestamp	Logical ID	Status	Status reason
2022-05-26 10:04:41 UTC+0530	m03p02stack	CREATE_COMPLETE	-
2022-05-26 10:04:39 UTC+0530	AnomalyDetectionServer	CREATE_COMPLETE	-
2022-05-26 10:04:31 UTC+0530	KinesisEventSourceMap	CREATE_COMPLETE	-
2022-05-26 10:04:30 UTC+0530	KinesisEventSourceMap	CREATE_IN_PROGRESS	Resource creation Initiated
2022-05-26 10:04:26 UTC+0530	KinesisEventSourceMap	CREATE_IN_PROGRESS	-
2022-05-26 10:04:26 UTC+0530	DynamoDBTable	CREATE_COMPLETE	-
2022-05-26 10:04:23 UTC+0530	AnomalyLambdaFunction	CREATE_COMPLETE	-
2022-05-26 10:04:16 UTC+0530	S3Bucket	CREATE_COMPLETE	-
2022-05-26 10:04:10 UTC+0530	MySubscription	CREATE_COMPLETE	-
2022-05-26 10:04:09 UTC+0530	MySubscription	CREATE_IN_PROGRESS	Resource creation Initiated
2022-05-26 10:04:09 UTC+0530	KinesisDataStream	CREATE_COMPLETE	-

This screenshot is identical to the one above, showing the AWS CloudFormation console with the stack named "m03p02stack". The "Events" tab is selected, displaying 29 events. All events are in the "CREATE_COMPLETE" status, indicating successful creation. The table includes columns for Timestamp, Logical ID, Status, and Status reason.

Timestamp	Logical ID	Status	Status reason
2022-05-26 10:04:31 UTC+0530	KinesisEventSourceMap	CREATE_COMPLETE	-
2022-05-26 10:04:30 UTC+0530	KinesisEventSourceMap	CREATE_IN_PROGRESS	Resource creation Initiated
2022-05-26 10:04:26 UTC+0530	KinesisEventSourceMap	CREATE_IN_PROGRESS	-
2022-05-26 10:04:26 UTC+0530	DynamoDBTable	CREATE_COMPLETE	-
2022-05-26 10:04:23 UTC+0530	AnomalyLambdaFunction	CREATE_COMPLETE	-
2022-05-26 10:04:16 UTC+0530	S3Bucket	CREATE_COMPLETE	-
2022-05-26 10:04:10 UTC+0530	MySubscription	CREATE_COMPLETE	-
2022-05-26 10:04:09 UTC+0530	MySubscription	CREATE_IN_PROGRESS	Resource creation Initiated
2022-05-26 10:04:09 UTC+0530	KinesisDataStream	CREATE_COMPLETE	-
2022-05-26 10:04:09 UTC+0530	MySubscription	CREATE_IN_PROGRESS	-
2022-05-26 10:04:06 UTC+0530	AnomalyDetectionServer	CREATE_IN_PROGRESS	Resource creation Initiated
2022-05-26 10:04:06 UTC+0530	SNSTopic	CREATE_COMPLETE	-
2022-05-26 10:04:04 UTC+0530	AnomalyDetectionServer	CREATE_IN_PROGRESS	-
2022-05-26 10:04:01 UTC+0530	ServerSecurityGroup	CREATE_COMPLETE	-

<Screenshot a (1) Cloud “m03p02stack” successful creation of “Resources” for TASK 3>

The screenshot shows the AWS CloudFormation console with the stack named "m03p02stack". The "Resources" tab is selected, displaying 9 resources:

Logical ID	Physical ID	Type	Status	Status reason	Mod
AnomalyDetectionServer	i-0503f87676ae83ba7	AWS::EC2::Instance	CREATE_COMPLETE	-	-
AnomalyLambdaFunction	anomalydetection	AWS::Lambda::Function	CREATE_COMPLETE	-	-
DynamoDBTable	m03p02_anomaly_data	AWS::DynamoDB::Table	CREATE_COMPLETE	-	-
KinesisDataStream	m03p02_raw_data_stream	AWS::Kinesis::Stream	CREATE_COMPLETE	-	-
KinesisEventSourceMap	ccbe424e-c4eb-456c-bfcf-cc742f101672	AWS::Lambda::EventSourceMapping	CREATE_COMPLETE	-	-
MySubscription	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:560b4598-a5a7-4943-b25e-e0253792d94b	AWS::SNS::Subscription	CREATE_COMPLETE	-	-
S3Bucket	m03p02databucket	AWS::S3::Bucket	CREATE_COMPLETE	-	-
SNSTopic	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts	AWS::SNS::Topic	CREATE_COMPLETE	-	-
ServerSecurityGroup	sg-0b1fe62c2540ef9f4	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-	-

<Screenshot a (1) Cloud “m03p02stack” successful “Outputs” for TASK 3>

The screenshot shows the AWS CloudFormation console with the stack named "m03p02stack". The "Outputs" tab is selected, displaying 7 outputs:

Key	Value	Description	Export name
ARN	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts	-	-
BucketName	m03p02databucket	-	-
FunctionName	anomalydetection	-	-
ID	arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts:560b4598-a5a7-4943-b25e-e0253792d94b	-	-
InstanceId	i-0503f87676ae83ba7	-	-
Name	m03p02_raw_data_stream	-	-
TableName	m03p02_anomaly_data	-	-

<Screenshot of separate S3 “lambda-deploymentpackage-bucket” created to upload lambda zip file for TASK 3>

The screenshot shows the AWS S3 console interface. In the top left, the 'Amazon S3' service is selected. The main navigation bar includes 'Search for services, features, blogs, docs, and more' and a global search bar. The breadcrumb path 'Amazon S3 > Buckets > lambda-deploymentpackage-bucket' is visible. The bucket name 'lambda-deploymentpackage-bucket' is displayed above the object list. Below the header, there are tabs for 'Objects' (which is selected), 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. A sub-header 'Objects (0)' indicates no objects are currently present. A prominent orange 'Upload' button is located at the bottom right of the object list area. The left sidebar contains sections for 'Buckets', 'Access Points', 'Storage Lens', 'AWS Marketplace', and 'Feedback'.

<Screenshot of uploading “lambda_deployment_package.zip” to lambda-deploymentpackage-bucket for TASK 3 >

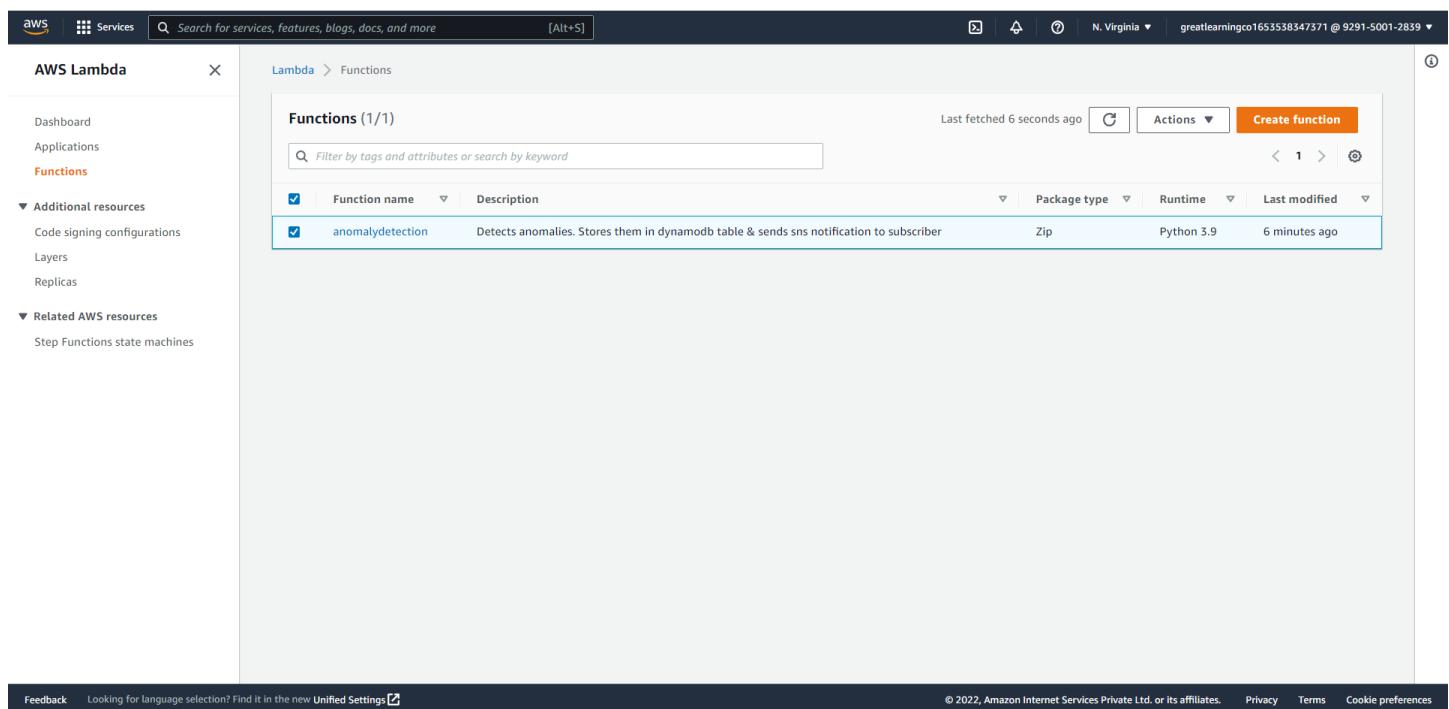
The screenshot shows the AWS S3 console after a file has been uploaded. A green success message at the top states 'Upload succeeded' with a link to 'View details below.' The breadcrumb path 'Amazon S3 > Buckets > lambda-deploymentpackage-bucket' is shown. The bucket name 'lambda-deploymentpackage-bucket' is displayed above the object list. The 'Objects' tab is selected. The sub-header 'Objects (1)' indicates one object is present. The object list table shows the following details:

Name	Type	Last modified	Size	Storage class
lambda_deployment_package.zip	zip	May 26, 2022, 09:58:59 (UTC+05:30)	958.0 B	Standard

The left sidebar is identical to the previous screenshot, showing 'Buckets', 'Access Points', 'Storage Lens', 'AWS Marketplace', and 'Feedback'.

Step number	b
Step name	Enabled lambda handler service
Instructions	<ol style="list-style-type: none"> 1) Make sure that cloud formation operation is successful 2) Check if the lambda handler service is successfully enabled
Expected screenshots	<ol style="list-style-type: none"> 1) Lambda handler

<Screenshot b (1) Lambda Handler successfully created through Cloudformation for TASK 3>



The screenshot shows the AWS Lambda Functions page. The left sidebar has 'AWS Lambda' selected under 'Services'. The main area shows a table of functions:

Function name	Description	Package type	Runtime	Last modified
anomalydetection	Detects anomalies. Stores them in dynamodb table & sends sns notification to subscriber	Zip	Python 3.9	6 minutes ago

At the bottom, there are links for 'Feedback', 'Unified Settings', '© 2022, Amazon Internet Services Private Ltd. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

<Screenshot b (1) Lambda Handler successfully created through Cloudformation for TASK 3>

The screenshot shows the AWS Lambda console with the 'anomalydetection' function selected. The top navigation bar includes 'Services', a search bar, and account information ('N. Virginia' and 'greatlearningco165538347371 @ 9291-5001-2839'). The main area displays the function details:

- Name:** anomalydetection
- Description:** Detects anomalies. Stores them in dynamodb table & sends sns notification to subscriber
- Last modified:** 7 minutes ago
- Function ARN:** arn:aws:lambda:us-east-1:929150012839:function:anomalydetection
- Application:** m03p02stack
- Function URL:** Info

Below the details, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is active, showing the code editor with the file 'anomaly_detection.py' open. The code implements a lambda handler to process Kinesis data and store anomalies in DynamoDB while sending SNS notifications.

<Screenshot b (1) Lambda Handler successfully created through Cloudformation for TASK 3>

The screenshot shows the AWS Lambda code editor for the 'anomalydetection' function. The top navigation bar includes 'Services', a search bar, and account information ('N. Virginia' and 'greatlearningco165538347371 @ 9291-5001-2839'). The main area displays the code source:

```
from pprint import pprint
import boto3
import json
import base64
from decimal import Decimal
from botocore.exceptions import ClientError

def lambda_handler(event, context):
    AWS_REGION = 'us-east-1'
    #print(event)

    dynamodb_res = boto3.resource('dynamodb', region_name=AWS_REGION)
    anomaly_table = dynamodb_res.Table('m03p02_anomaly_data')

    sns_client = boto3.client('sns', region_name=AWS_REGION)
    topic_arn = "arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts"

    for record in event['Records']:
        data_point = base64.b64decode(record['kinesis']['data'])
        data_point = str(data_point, 'utf-8')
        pprint(data_point, sort_dicts=False)
        data_point = json.loads(data_point)

        anomaly_type = {}
        if data_point['value'] <= (1.1 * float(data_point['lowest_temp'])):
            anomaly_type = "Cold"
        elif data_point['value'] >= (0.9 * float(data_point['highest_point'])):
            anomaly_type = "Hot"

        anomaly_data = {'deviceid': data_point['deviceid'],
                        'anomalyDate': data_point['date'],
                        'timestamp': data_point['timestamp'],
                        'value': data_point['value'],
```

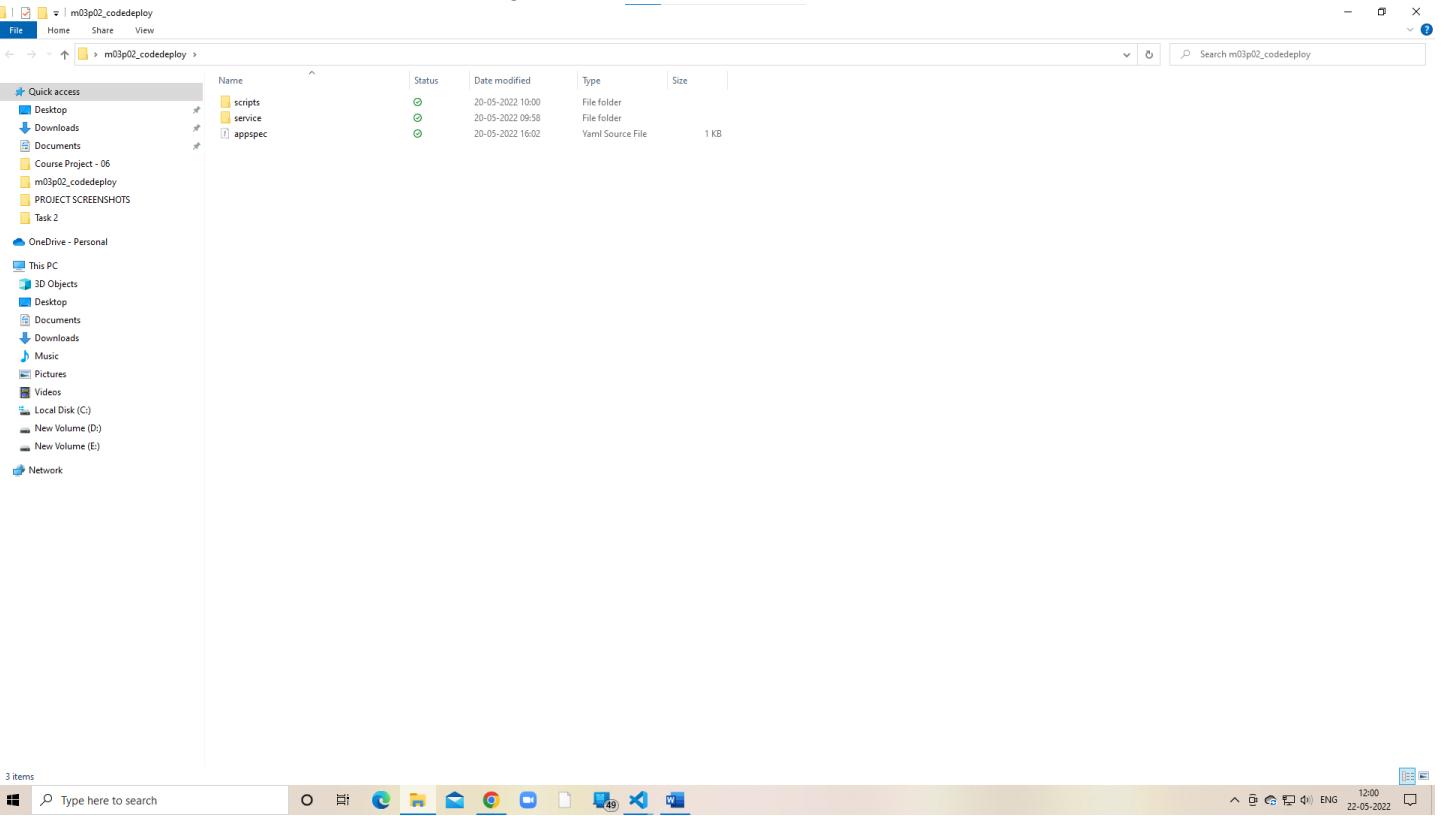
The code defines a lambda handler that processes Kinesis data records. It extracts the data point, decodes it from base64, and prints it. Then, it checks if the value is below the lowest temperature threshold (1.1 times the lowest temp) or above the highest temperature threshold (0.9 times the highest temp), classifying it as 'Cold' or 'Hot' respectively. Finally, it constructs a DynamoDB item and a corresponding SNS message payload for each record.

<Screenshot b (1) Lambda Handler successfully created through Cloudformation for TASK 3>

The screenshot shows the AWS Lambda function configuration page. At the top, there's a search bar and navigation links for services and regions (N. Virginia). Below the header, there are three tabs: 'Code properties', 'Runtime settings', and 'Layers'. The 'Code properties' tab is active, displaying details like Package size (958.0 byte), SHA256 hash (Y0dWv2EDFE8Uv6oDaHUXiF1uwGu1ds6YFaZ12PE9RE=), and Last modified (May 26, 2022, 10:04 AM GMT+5:30). The 'Runtime settings' tab shows Runtime (Python 3.9), Handler (anomaly_detection.lambda_handler), and Architecture (x86_64). The 'Layers' tab shows a message: 'There is no data to display.' At the bottom, there are footer links for Feedback, Unified Settings, and various AWS services.

Step number	c
Step name	CodeDeploy
Instructions	<ol style="list-style-type: none">3) Perform the CodeDeploy operation through AWS console4) If CodeDeploy is successful look if the Anomaly detection related data is going into the relevant DB and SNS services
Expected screenshots	<ol style="list-style-type: none">4) Successful CodeDeploy operation summary page5) Anomaly data pushed in the DynamoDB6) SNS notification

<Screenshot of “m03p02_codedeploy.zip” folder which needs to be uploaded in S3 bucket for TASK 3>



<Screenshot of “m03p02_codedeploy.zip” which contains both application and appspec.yml uploaded to m03p02bucket for TASK 3>

Destination	Succeeded	Failed
s3://m03p02databucket/rawdatapackage/	1 file, 2.9 KB (100.00%)	0 files, 0 B (0%)

Files and folders (1 Total, 2.9 KB)	
Name	Type
m03p02_codedeploy.zip	application/x-zip-compressed

<Screenshot of “m03p02_codedeploy.zip” which contains both application and appspec.yml uploaded for TASK 3>

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various navigation options like Buckets, Storage Lens, and Feature spotlight. The main area shows a bucket named 'm03p02databasebucket'. Inside it, there's a folder named 'rawdatapackage/'. Underneath, there's a table titled 'Objects (1)' showing one item: 'm03p02_codedeploy.zip'. The object details show it's a zip file, 2.9 KB in size, and was last modified on May 22, 2022, at 12:02:04 (UTC+05:30). There are buttons for Actions (Copy S3 URI, Copy URL, Download, Open, Delete, Create folder, Upload) and a search bar.

<Screenshot c (4) of Successful CodeDeploy Summary page for TASK 3 >

The screenshot shows the AWS CodeDeploy Summary page for a deployment named 'd-4AM71WISH'. The top bar indicates 'Success' and 'Deployment created'. The main content area shows the deployment status: 'Installing application on your instances' with a progress bar at 100% completion, showing 1 of 1 instances updated and a status of 'Succeeded'. Below this, the 'Deployment details' section provides information about the application ('anomaly-detection-service'), deployment ID ('d-4AM71WISH'), deployment configuration ('CodeDeployDefault.AllAtOnce'), deployment group ('m03p02_anomaly_detection_group'), and deployment description ('-'). The 'Revision details' section shows the revision location ('s3://m03p02databasebucket/rawdatapackage/m03p02_codedeploy.'), revision created ('Just now'), and revision description ('Application revision registered by Deployment ID: d-4AM71WISH').

<Screenshot c(4) of Successful CodeDeploy for TASK 3>

The screenshot shows the AWS CodeDeploy console. On the left, a sidebar navigation includes 'Developer Tools' and 'CodeDeploy' under 'Deploy'. Other options like 'Source', 'Artifacts', 'Build', 'Deploy', 'Getting started', 'Deployments', 'Deployment', 'Applications', 'Deployment configurations', 'On-premises instances', 'Pipeline', and 'Settings' are listed. Below these are 'Feedback' and 'Go to resource' links.

The main content area displays deployment details for the 'anomaly-detection-service'. It shows the 'Deployment ID' as 'd-4AM71WISH', 'Status' as 'Succeeded', and 'Initiated by' as 'User action'. Deployment configuration is 'CodeDeployDefault.AllAtOnce' and the deployment group is 'm03p02_anomaly_detection_group'. The deployment description is '-'.

Revision details:

- Revision location: s3://m03p02databucket/rawdatapackage/m03p02_codedeploy.zip
- Revision created: 2 minutes ago
- Revision description: Application revision registered by Deployment ID: d-4AM71WISH

Event Log:

Event	Duration	Status	Error code	Start time	End time
ApplicationStop	less than one second	Succeeded	-	May 26, 2022 10:34 AM (UTC+5:30)	May 26, 2022 10:34 AM (UTC+5:30)
DownloadBundle	less than one second	Succeeded	-	May 26, 2022 10:34 AM (UTC+5:30)	May 26, 2022 10:34 AM (UTC+5:30)
BeforeInstall	46 seconds	Succeeded	-	May 26, 2022 10:34 AM (UTC+5:30)	May 26, 2022 10:35 AM (UTC+5:30)
Install	less than one second	Succeeded	-	May 26, 2022 10:35 AM (UTC+5:30)	May 26, 2022 10:35 AM (UTC+5:30)
AfterInstall	less than one second	Succeeded	-	May 26, 2022 10:35 AM (UTC+5:30)	May 26, 2022 10:35 AM (UTC+5:30)
ApplicationStart	less than one second	Succeeded	-	May 26, 2022 10:35 AM (UTC+5:30)	May 26, 2022 10:35 AM (UTC+5:30)
ValidateService	less than one second	Succeeded	-	May 26, 2022 10:35 AM (UTC+5:30)	May 26, 2022 10:35 AM (UTC+5:30)

At the bottom, there are 'Feedback', 'Language selection', and copyright information: © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences.

<Screenshot c(5) of Anomaly Data pushed in Dynamodb table for TASK 3>

The screenshot shows the AWS DynamoDB console. The left sidebar includes 'Dashboard', 'Tables', 'Update settings', 'Explore items' (which is selected), 'PartiQL editor', 'Backups', 'Exports to S3', 'Reserved capacity', 'DAX' (with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'), and 'Tell us what you think' with links to 'Return to the previous console experience' and 'Density settings'.

The main area shows the 'Tables (1)' section for the 'm03p02_anomaly_data' table. It has an 'Autopreview' button, 'Actions' dropdown, 'Create item' button, and 'Update table settings' button. A search bar and a 'Find tables by table name' input field are also present.

Scan/Query items:

- Table or index: m03p02_anomaly_data
- Filters
- Run button
- Completed message: Read capacity units consumed: 3.5
- Note: This table has more items to retrieve. To retrieve the next page of items, choose Retrieve next page.

Items returned (300):

deviceid	timestamp	anomalyDate	anomalyType	value
DHT_001	2022-05-26 05:05:24.957143	2022-05-26	Cold	95.1
DHT_001	2022-05-26 05:05:25.958226	2022-05-26	Cold	95
DHT_001	2022-05-26 05:05:26.958226	2022-05-26	Cold	95.2

At the bottom, there are 'Feedback', 'Language selection', and copyright information: © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences.

< Screenshot c (5) of Anomaly Data pushed in Dynamodb table for TASK 3>

Scan Query

Table or index: m03p02_anomaly_data

Completed Read capacity units consumed: 3.5

Items returned (595)

	deviceid	timestamp	anomalyDate	anomalyType	value
□	DHT_001	2022-05-26 05:10:24.958263	2022-05-26	Cold	98.3
□	DHT_001	2022-05-26 05:10:25.958271	2022-05-26	Cold	94.3
□	DHT_001	2022-05-26 05:10:26.958241	2022-05-26	Cold	93.4
□	DHT_001	2022-05-26 05:10:27.958268	2022-05-26	Cold	96.4
□	DHT_001	2022-05-26 05:10:28.958275	2022-05-26	Cold	97
□	DHT_001	2022-05-26 05:10:29.958270	2022-05-26	Cold	98.8
□	DHT_001	2022-05-26 05:10:30.958255	2022-05-26	Cold	97.1
□	DHT_001	2022-05-26 05:10:31.958266	2022-05-26	Cold	96.2

< Screenshot c (5) of Anomaly Data pushed in Dynamodb table for TASK 3>

Scan Query

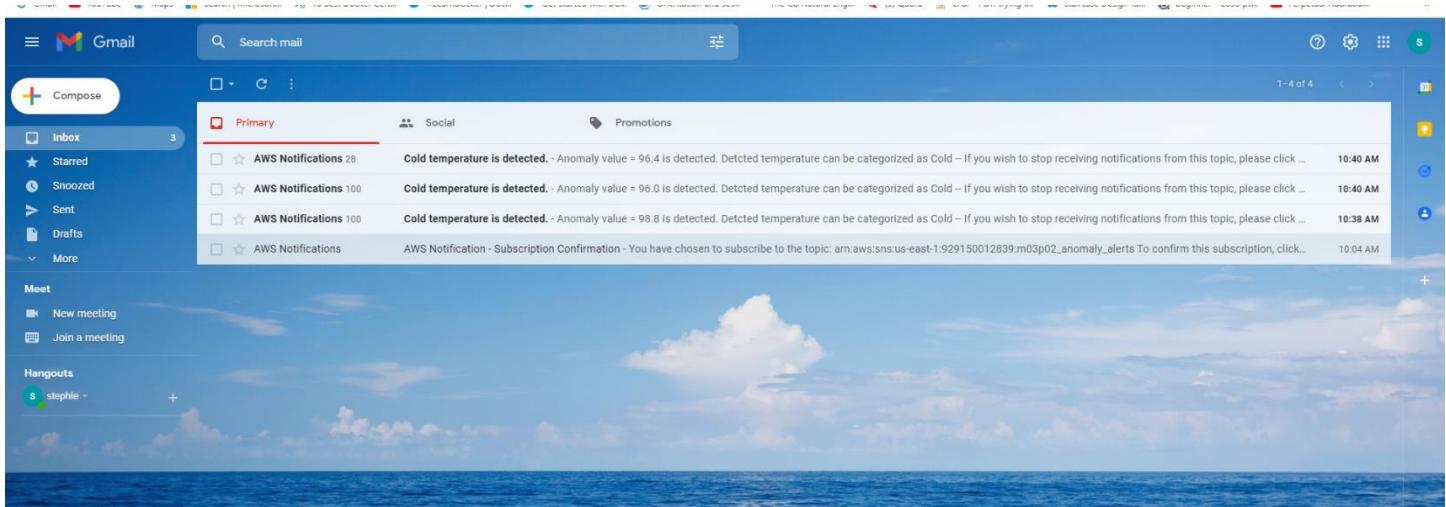
Table or index: m03p02_anomaly_data

Completed Read capacity units consumed: 3.5

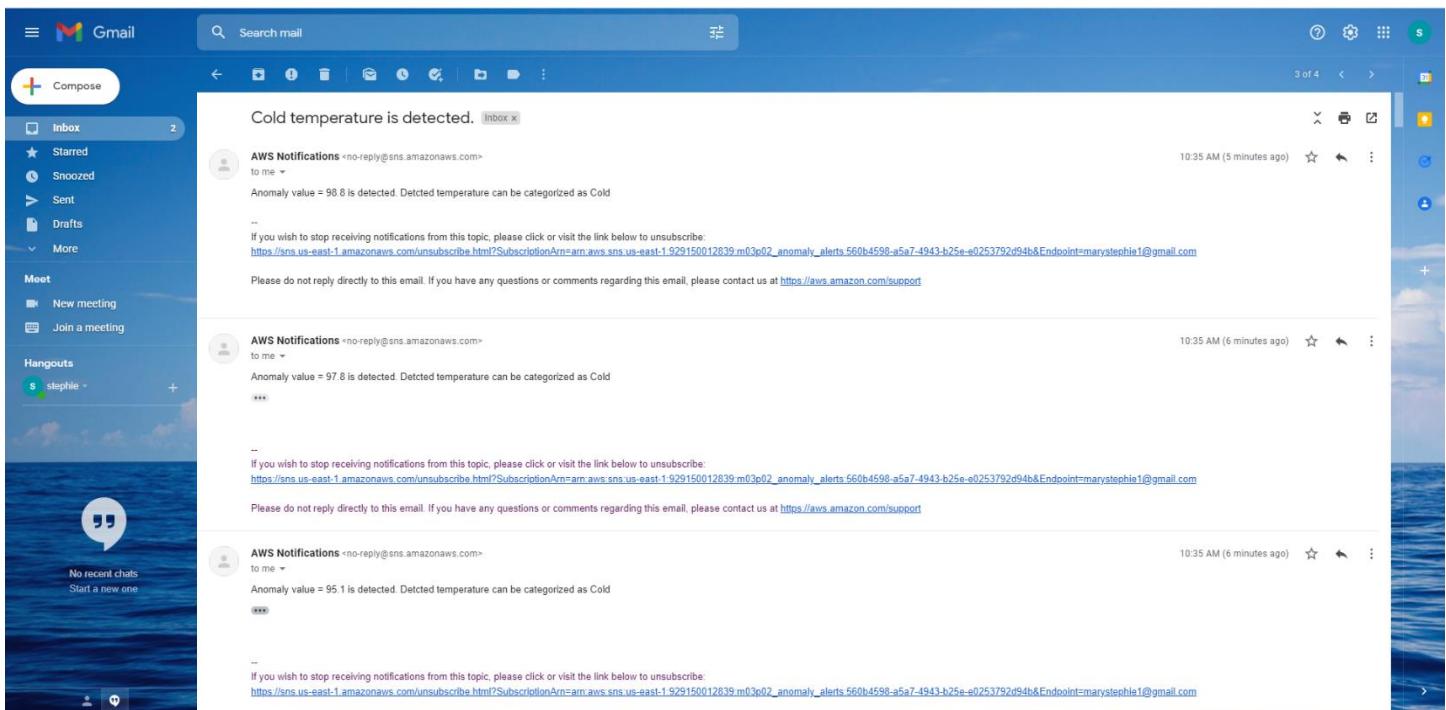
Items returned (300)

	deviceid	timestamp	anomalyDate	anomalyType	value
□	DHT_001	2022-05-26 05:06:00.958251	2022-05-26	Cold	98.9
□	DHT_001	2022-05-26 05:06:01.962186	2022-05-26	Cold	97.6
□	DHT_001	2022-05-26 05:06:02.958253	2022-05-26	Cold	96.7
□	DHT_001	2022-05-26 05:06:03.958235	2022-05-26	Cold	96.1
□	DHT_001	2022-05-26 05:06:04.958266	2022-05-26	Cold	95.2
□	DHT_001	2022-05-26 05:06:05.958284	2022-05-26	Cold	96.4
□	DHT_001	2022-05-26 05:06:06.958244	2022-05-26	Cold	96
□	DHT_001	2022-05-26 05:06:07.958265	2022-05-26	Cold	98
□	DHT_001	2022-05-26 05:06:08.958268	2022-05-26	Cold	98.1
□	DHT_001	2022-05-26 05:06:09.958277	2022-05-26	Cold	93.2
□	DHT_001	2022-05-26 05:06:10.958263	2022-05-26	Cold	95.9
□	DHT_001	2022-05-26 05:06:11.958253	2022-05-26	Cold	96.1
□	DHT_001	2022-05-26 05:06:12.958223	2022-05-26	Cold	92.4
□	DHT_001	2022-05-26 05:06:13.958257	2022-05-26	Cold	96.4
□	DHT_001	2022-05-26 05:06:14.958270	2022-05-26	Cold	91.6

<Screenshot c (6) of SNS notification for TASK 3>



<Screenshot c (6) of SNS notification for TASK 3>



<Screenshot c (6) of SNS notification for TASK 3>

The screenshot shows a Gmail inbox with five recent messages from 'AWS Notifications <no-reply@sns.amazonaws.com>'. Each message is directed to 'me' and contains the subject 'Cold temperature is detected.'. The body of each message includes the following text:
Anomaly value = [value] is detected. Detected temperature can be categorized as Cold
...
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_560b4598-a5a7-4943-b25e-e0253792d94b&Endpoint=marystephie1@gmail.com
Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

10:35 AM (5 minutes ago)
10:35 AM (6 minutes ago)

<Screenshot c (6) of SNS notification for TASK 3>

The screenshot shows a Gmail inbox with six recent messages from 'AWS Notifications <no-reply@sns.amazonaws.com>'. Each message is directed to 'me' and contains the subject 'Cold temperature is detected.'. The body of each message includes the following text:
Anomaly value = [value] is detected. Detected temperature can be categorized as Cold
...
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:929150012839:m03p02_anomaly_alerts_560b4598-a5a7-4943-b25e-e0253792d94b&Endpoint=marystephie1@gmail.com
Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

10:35 AM (5 minutes ago)
10:35 AM (5 minutes ago)