AWS E-Commerce Analytics Platform

Gabrielle Glasgow, Jason Fearnell, Justin Quinn, Max Ross

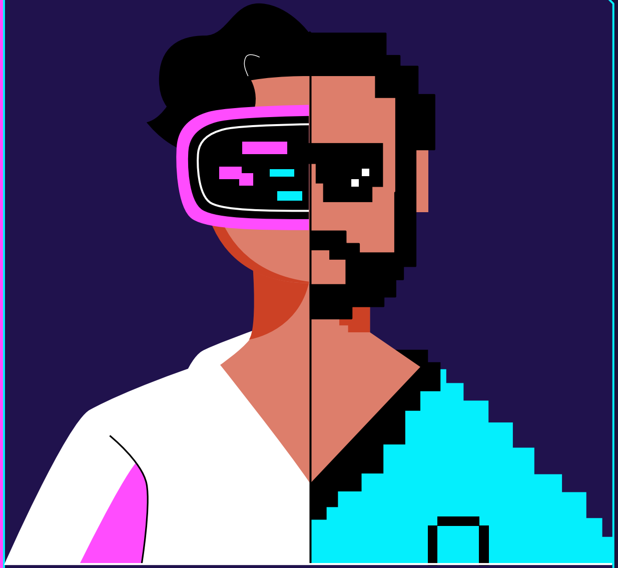
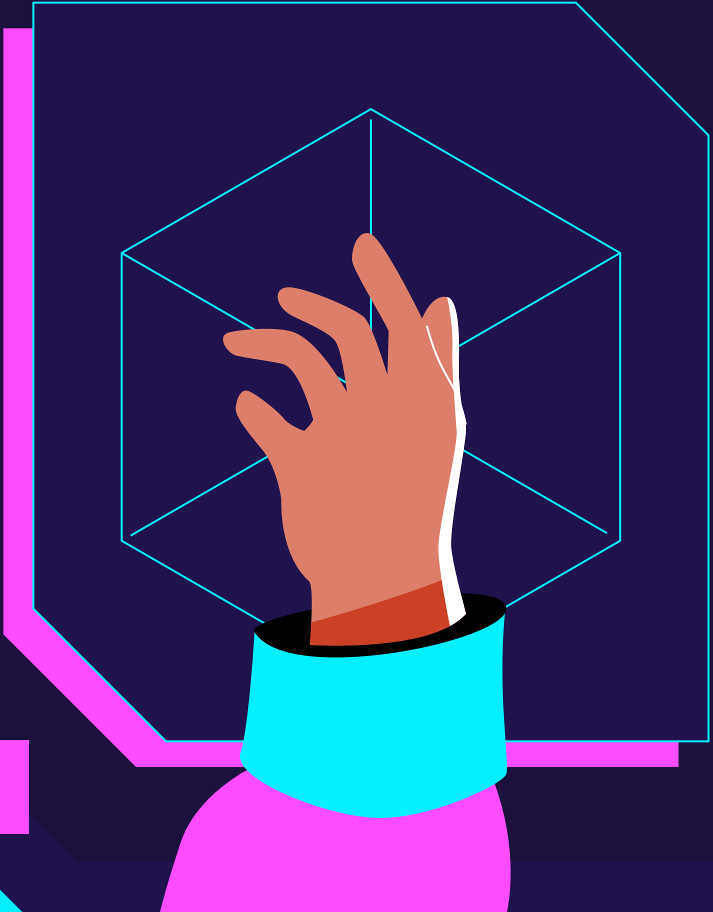


TABLE OF *CONTENTS*



01

INTRODUCTION

INTRODUCTION

01 Our company

The aim is to leverage the power of AWS services to capture, process and visualize our data, providing actionable insights into customer behavior.

02 Key Objectives

- Efficient Data Management
- Streamlined Data Processing
- Insightful Analytics
- Secure Integration

KEY Objective Deep Dive

Data Management

Design and implement a DynamoDB schema to store data about users, transactions, and products ensuring it supports querying

Streamlined Data Processing

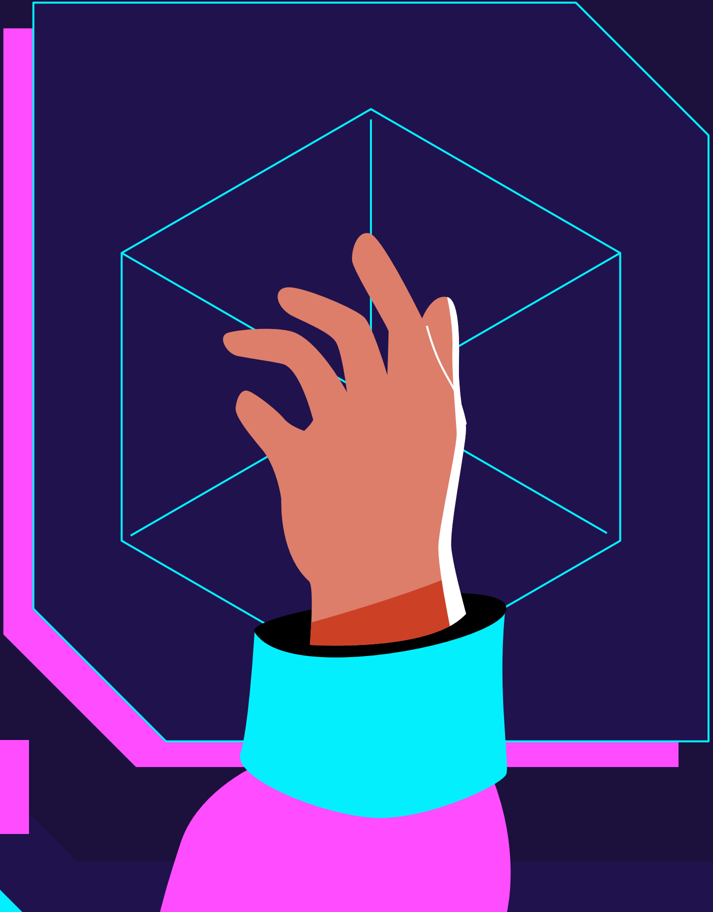
Create Lambda functions for ETL tasks including data validation, transformation, and loading into DynamoDB

Insightful Analytics

Use CloudWatch for Monitoring and setting up a dashboard for real-time insights

Secure Integration

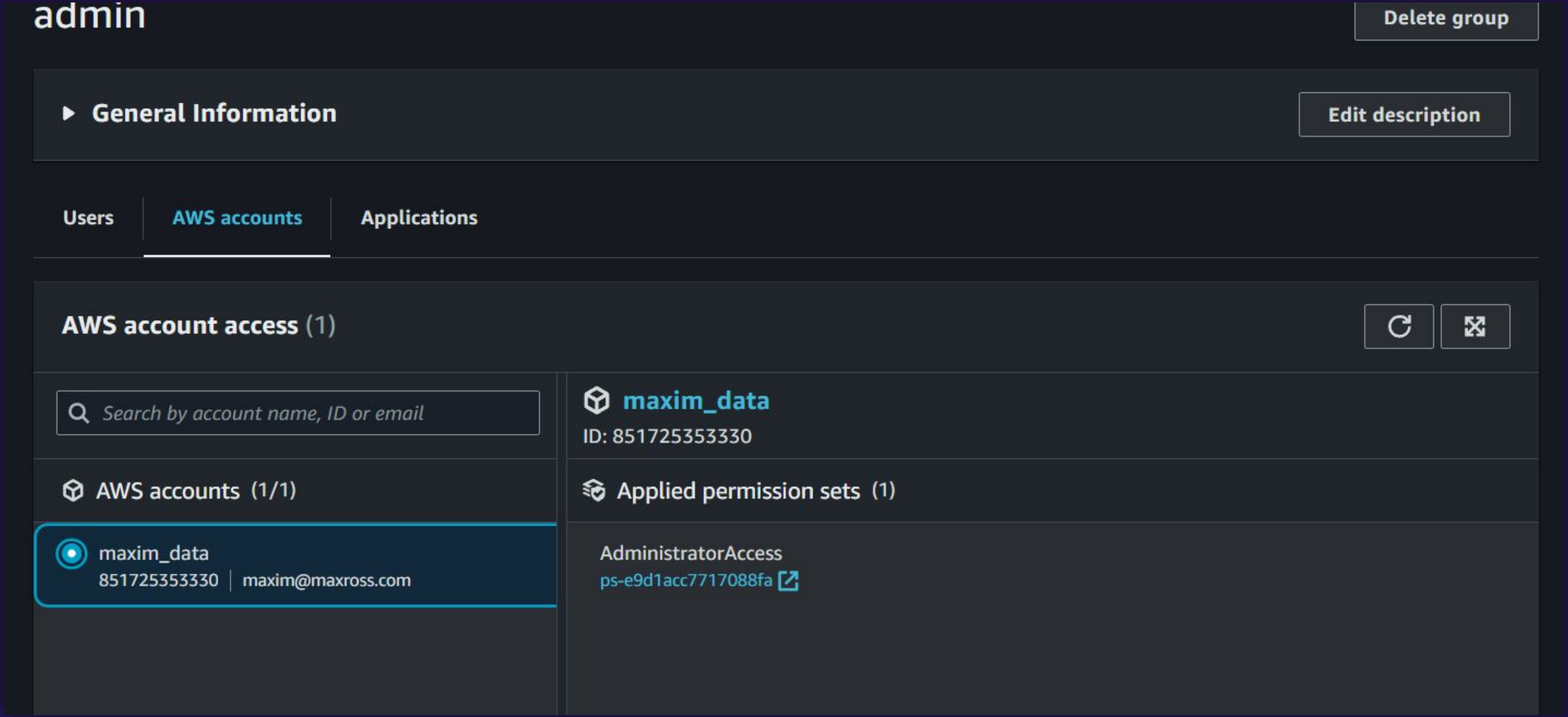
Set up IAM for managing access permissions, enforcing the principle of least privilege



02

Integration & Security

IAM Identity Center



admin

Delete group

General Information

Edit description

Users | AWS accounts | Applications

AWS account access (1)

Search by account name, ID or email

AWS accounts (1/1)

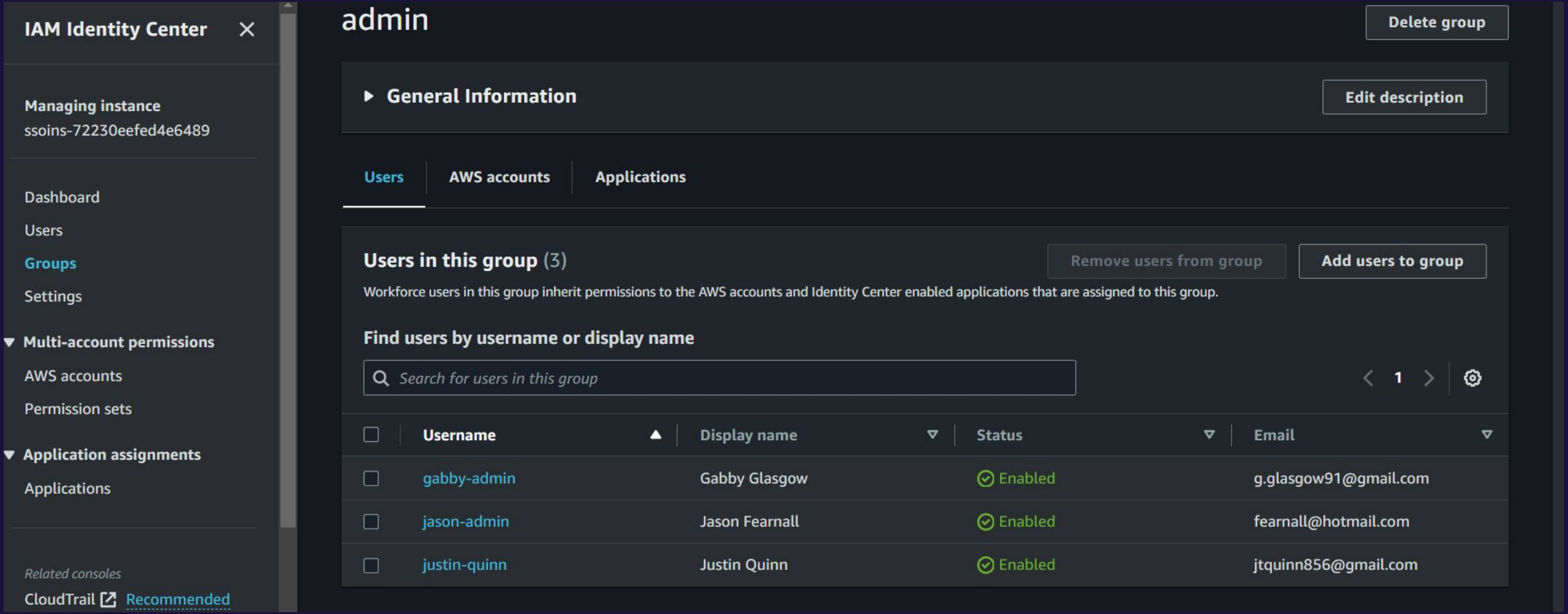
maxim_data
851725353330 | maxim@maxross.com

maxim_data
ID: 851725353330

Applied permission sets (1)

AdministratorAccess
ps-e9d1acc7717088fa

Admin Group



admin

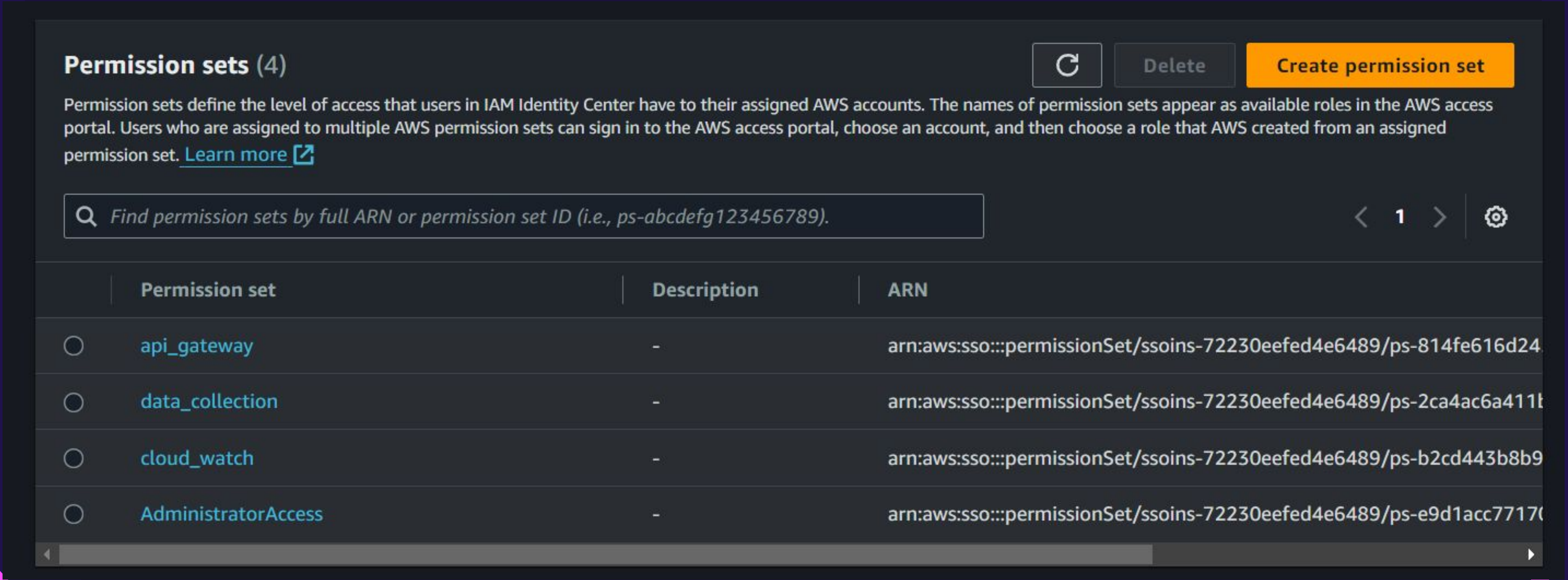
Users in this group (3)

Workforce users in this group inherit permissions to the AWS accounts and Identity Center enabled applications that are assigned to this group.

Find users by username or display name

Username	Display name	Status	Email
gabby-admin	Gabby Glasgow	Enabled	g.glasgow91@gmail.com
jason-admin	Jason Fearnall	Enabled	fearnall@hotmail.com
justin-quinn	Justin Quinn	Enabled	jtquinn856@gmail.com

Admin Permission Sets



Permission sets (4)

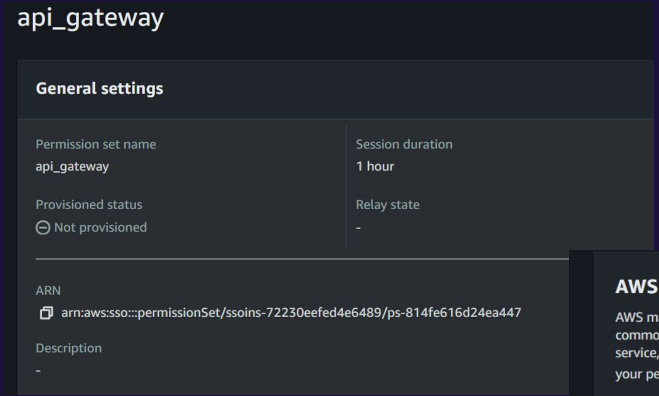
Permission sets define the level of access that users in IAM Identity Center have to their assigned AWS accounts. The names of permission sets appear as available roles in the AWS access portal. Users who are assigned to multiple AWS permission sets can sign in to the AWS access portal, choose an account, and then choose a role that AWS created from an assigned permission set. Learn more

Find permission sets by full ARN or permission set ID (i.e., ps-abcdefg123456789).

1

	Permission set	Description	ARN
○	api_gateway	-	arn:aws:sso:::permissionSet/ssoins-72230eefed4e6489/ps-814fe616d24
○	data_collection	-	arn:aws:sso:::permissionSet/ssoins-72230eefed4e6489/ps-2ca4ac6a411
○	cloud_watch	-	arn:aws:sso:::permissionSet/ssoins-72230eefed4e6489/ps-b2cd443b8b9
○	AdministratorAccess	-	arn:aws:sso:::permissionSet/ssoins-72230eefed4e6489/ps-e9d1acc7717

Permission Roles = Least Privilege



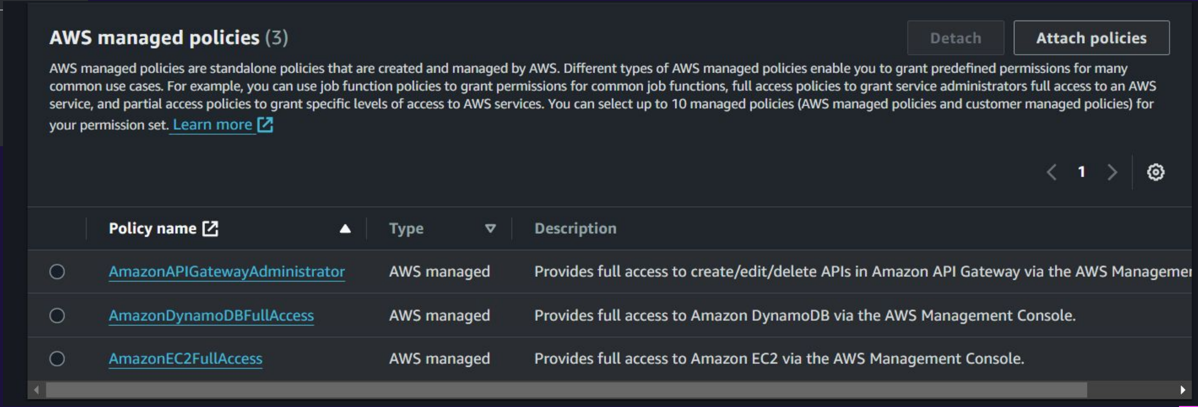
api_gateway

General settings

Permission set name api_gateway	Session duration 1 hour
Provisioned status Not provisioned	Relay state -

ARN
arn:aws:sso:::permissionSet/ssoins-72230eefed4e6489/ps-814fe616d24ea447

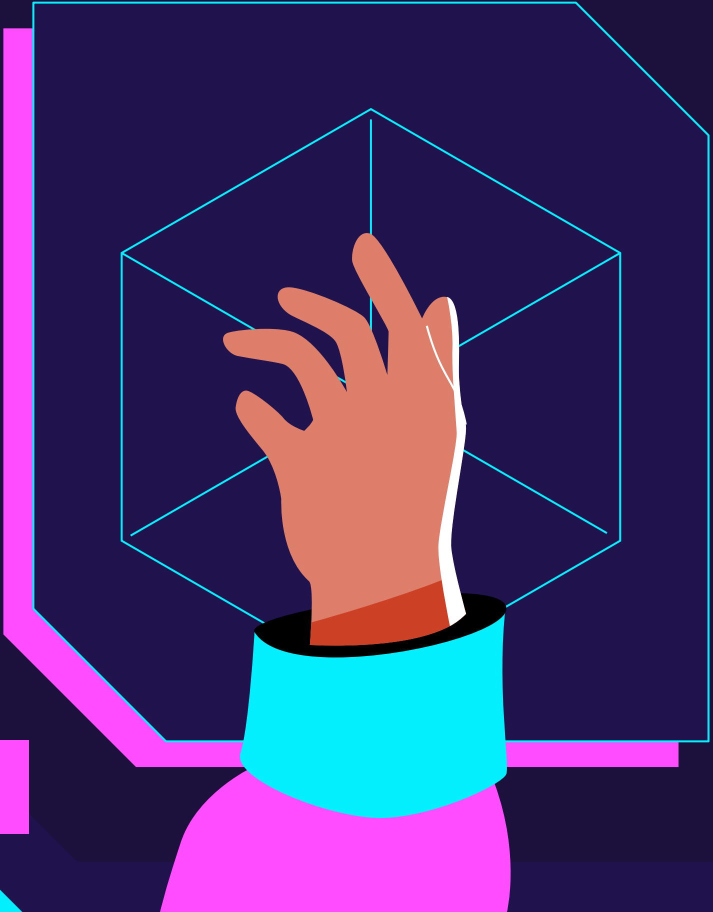
Description
-



AWS managed policies (3)

AWS managed policies are standalone policies that are created and managed by AWS. Different types of AWS managed policies enable you to grant predefined permissions for many common use cases. For example, you can use job function policies to grant permissions for common job functions, full access policies to grant service administrators full access to an AWS service, and partial access policies to grant specific levels of access to AWS services. You can select up to 10 managed policies (AWS managed policies and customer managed policies) for your permission set. Learn more

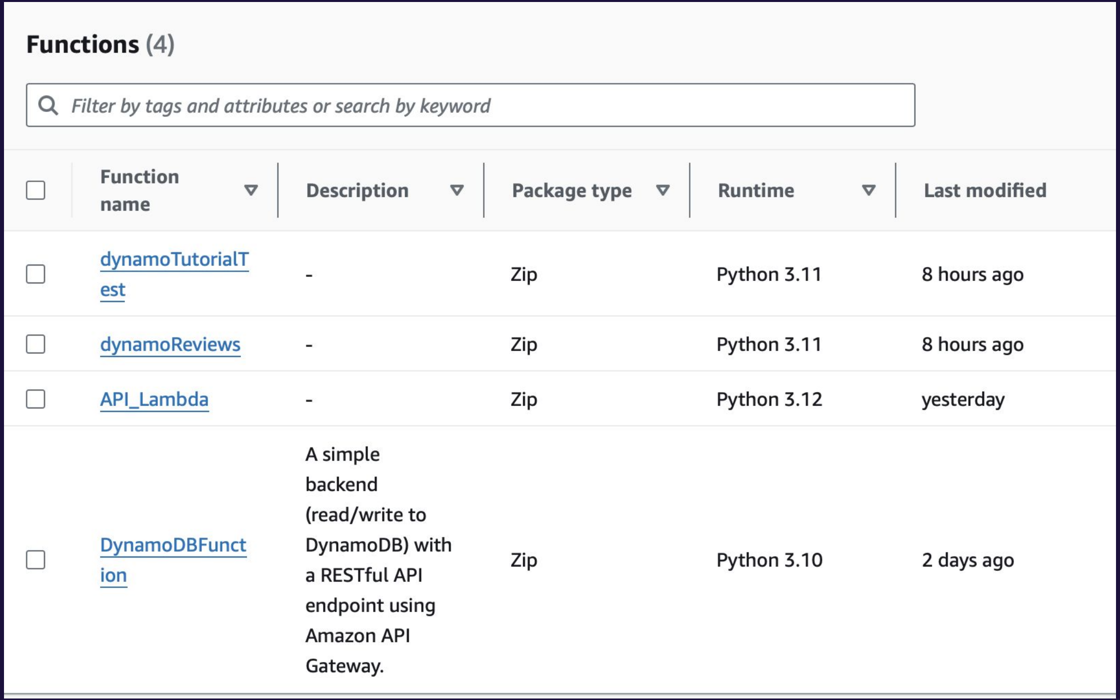
Policy name	Type	Description
AmazonAPIGatewayAdministrator	AWS managed	Provides full access to create/edit/delete APIs in Amazon API Gateway via the AWS Managemer
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB via the AWS Management Console.
AmazonEC2FullAccess	AWS managed	Provides full access to Amazon EC2 via the AWS Management Console.



03

Data Processing w/ Lambda

Lambda Functions



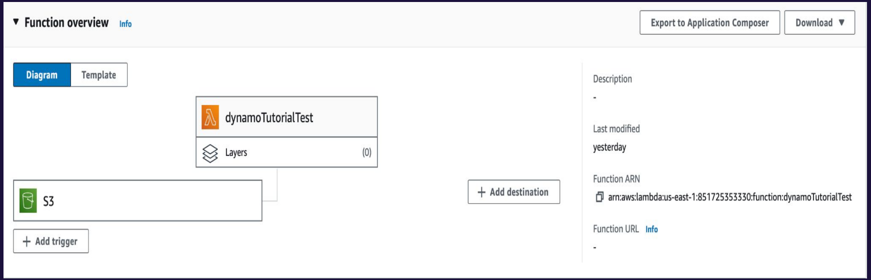
Functions (4)

Filter by tags and attributes or search by keyword

Function name	Description	Package type	Runtime	Last modified
dynamoTutorialTest	-	Zip	Python 3.11	8 hours ago
dynamoReviews	-	Zip	Python 3.11	8 hours ago
API_Lambda	-	Zip	Python 3.12	yesterday
DynamoDBFunction	A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway.	Zip	Python 3.10	2 days ago

Product Table Lambda

This lamba is triggered by a S3 object creation event. It reads the data stored in a s3 bucket, processing the data, and then inserts each row of the CSV data into a DynamoDB table called "product_table".

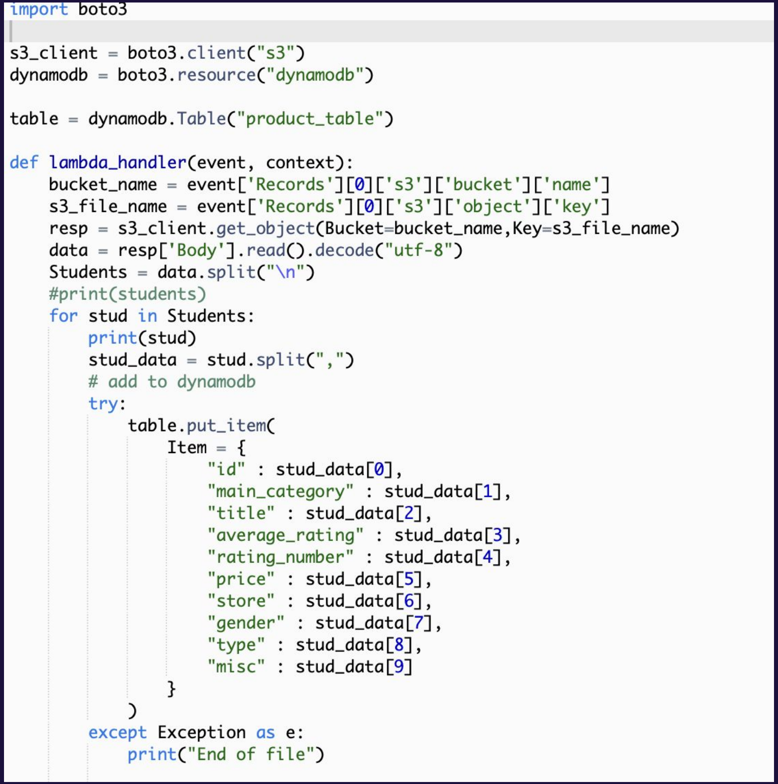


Description
-

Last modified
yesterday

Function ARN
arn:aws:lambda:us-east-1:851725353330:function:dynamoTutorialTest

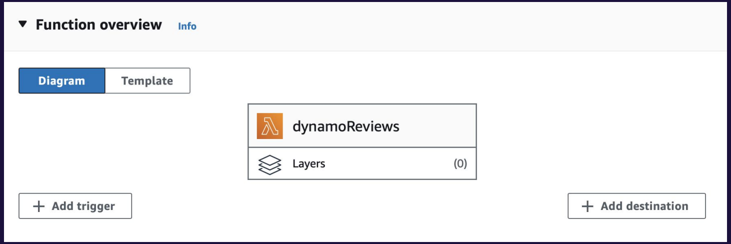
Function URL Info
-

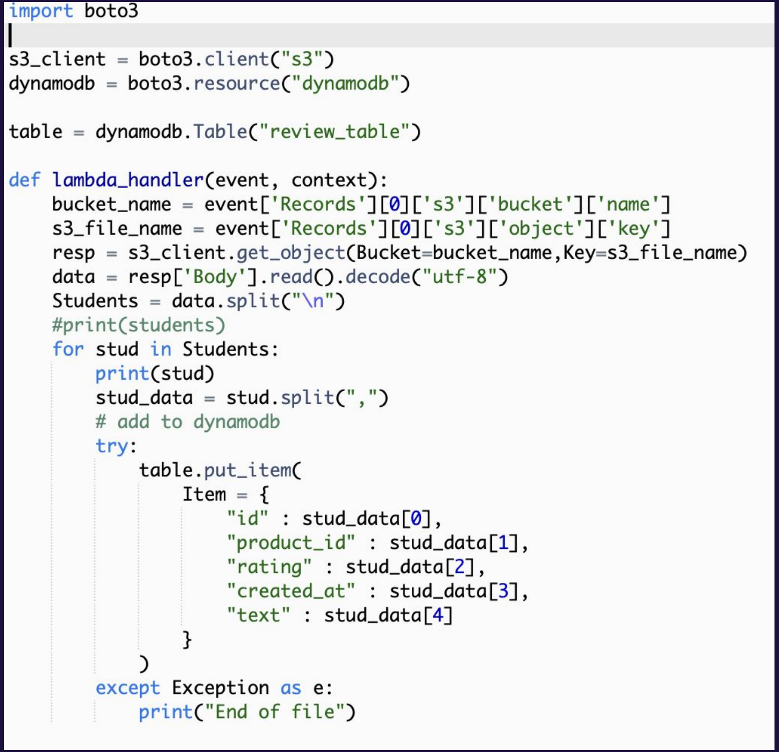


```python
import boto3

s3_client = boto3.client("s3")
dynamodb = boto3.resource("dynamodb")

table = dynamodb.Table("product_table")

def lambda_handler(event, context):
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    s3_file_name = event['Records'][0]['s3']['object']['key']
    resp = s3_client.get_object(Bucket=bucket_name,Key=s3_file_name)
    data = resp['Body'].read().decode("utf-8")
    Students = data.split("\n")
    #print(students)
    for stud in Students:
        print(stud)
        stud_data = stud.split(",")
        # add to dynamodb
        try:
            table.put_item(
                Item = {
                    "id" : stud_data[0],
                    "main_category" : stud_data[1],
                    "title" : stud_data[2],
                    "average_rating" : stud_data[3],
                    "rating_number" : stud_data[4],
                    "price" : stud_data[5],
                    "store" : stud_data[6],
                    "gender" : stud_data[7],
                    "type" : stud_data[8],
                    "misc" : stud_data[9]
                }
            )
        except Exception as e:
            print("End of file")
```

Review Table Lambda

This Lambda function is triggered by an S3 object creation event. It reads data from an object stored in an S3 bucket, processes the data, and then inserts each row of the CSV data into a DynamoDB table named "review_table".

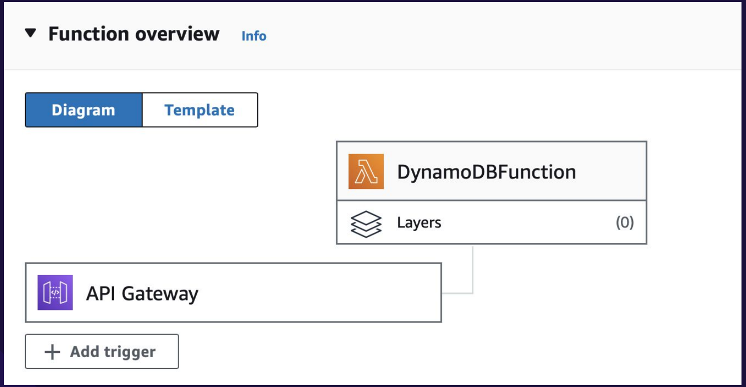


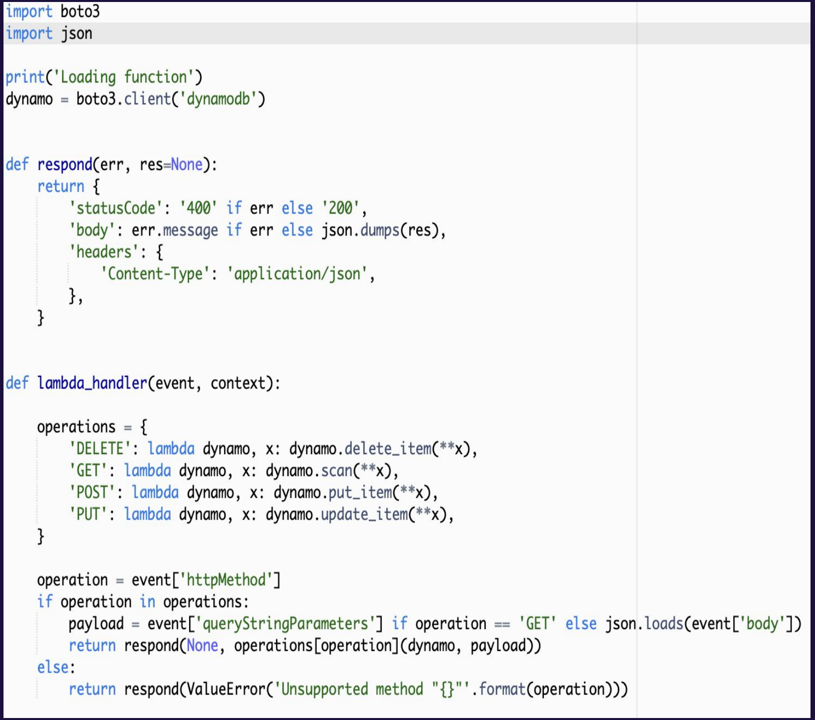


```python
import boto3

s3_client = boto3.client("s3")
dynamodb = boto3.resource("dynamodb")

table = dynamodb.Table("review_table")

def lambda_handler(event, context):
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    s3_file_name = event['Records'][0]['s3']['object']['key']
    resp = s3_client.get_object(Bucket=bucket_name,Key=s3_file_name)
    data = resp['Body'].read().decode("utf-8")
    Students = data.split("\n")
    #print(students)
    for stud in Students:
        print(stud)
        stud_data = stud.split(",")
        # add to dynamodb
        try:
            table.put_item(
                Item = {
                    "id" : stud_data[0],
                    "product_id" : stud_data[1],
                    "rating" : stud_data[2],
                    "created_at" : stud_data[3],
                    "text" : stud_data[4]
                }
            )
        except Exception as e:
            print("End of file")
```

Lambda for API Gateway

This Lambda function acts as an HTTP endpoint using API Gateway to interact with DynamoDB. It supports CRUD (Create, Read, Update, Delete) operations on a DynamoDB table based on the HTTP method of the incoming request.

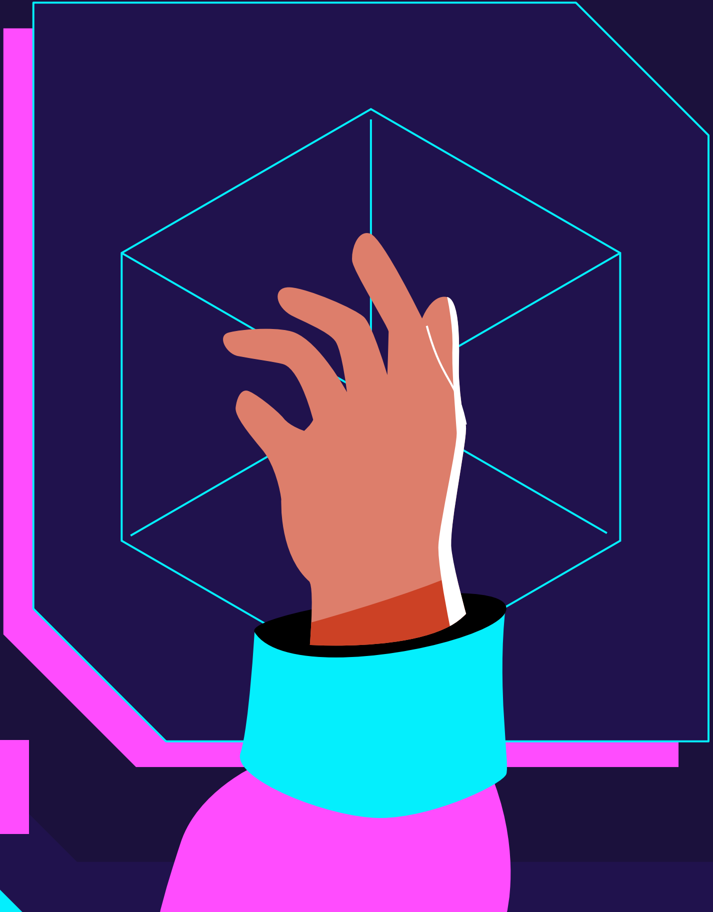




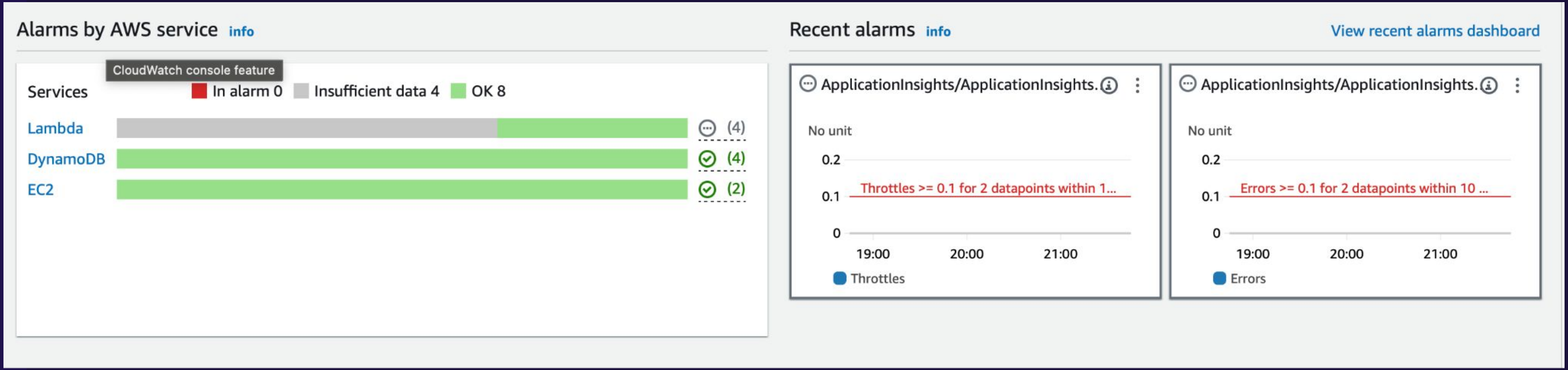
```python
import boto3
import json

print('Loading function')
dynamo = boto3.client('dynamodb')


def respond(err, res=None):
    return {
        'statusCode': '400' if err else '200',
        'body': err.message if err else json.dumps(res),
        'headers': {
            'Content-Type': 'application/json',
        },
    }


def lambda_handler(event, context):

    operations = {
        'DELETE': lambda dynamo, x: dynamo.delete_item(**x),
        'GET': lambda dynamo, x: dynamo.scan(**x),
        'POST': lambda dynamo, x: dynamo.put_item(**x),
        'PUT': lambda dynamo, x: dynamo.update_item(**x),
    }

    operation = event['httpMethod']
    if operation in operations:
        payload = event['queryStringParameters'] if operation == 'GET' else json.loads(event['body'])
        return respond(None, operations[operation](dynamo, payload))
    else:
        return respond(ValueError('Unsupported method "{}"'.format(operation)))
```

04

Data Collection, Storage and Alarms



Cloudwatch



Alarms by AWS service info

CloudWatch console feature

Services 🟥 In alarm 0 ▨ Insufficient data 4 🟩 OK 8

Lambda (4)
DynamoDB (4)
EC2 (2)

Recent alarms info View recent alarms dashboard

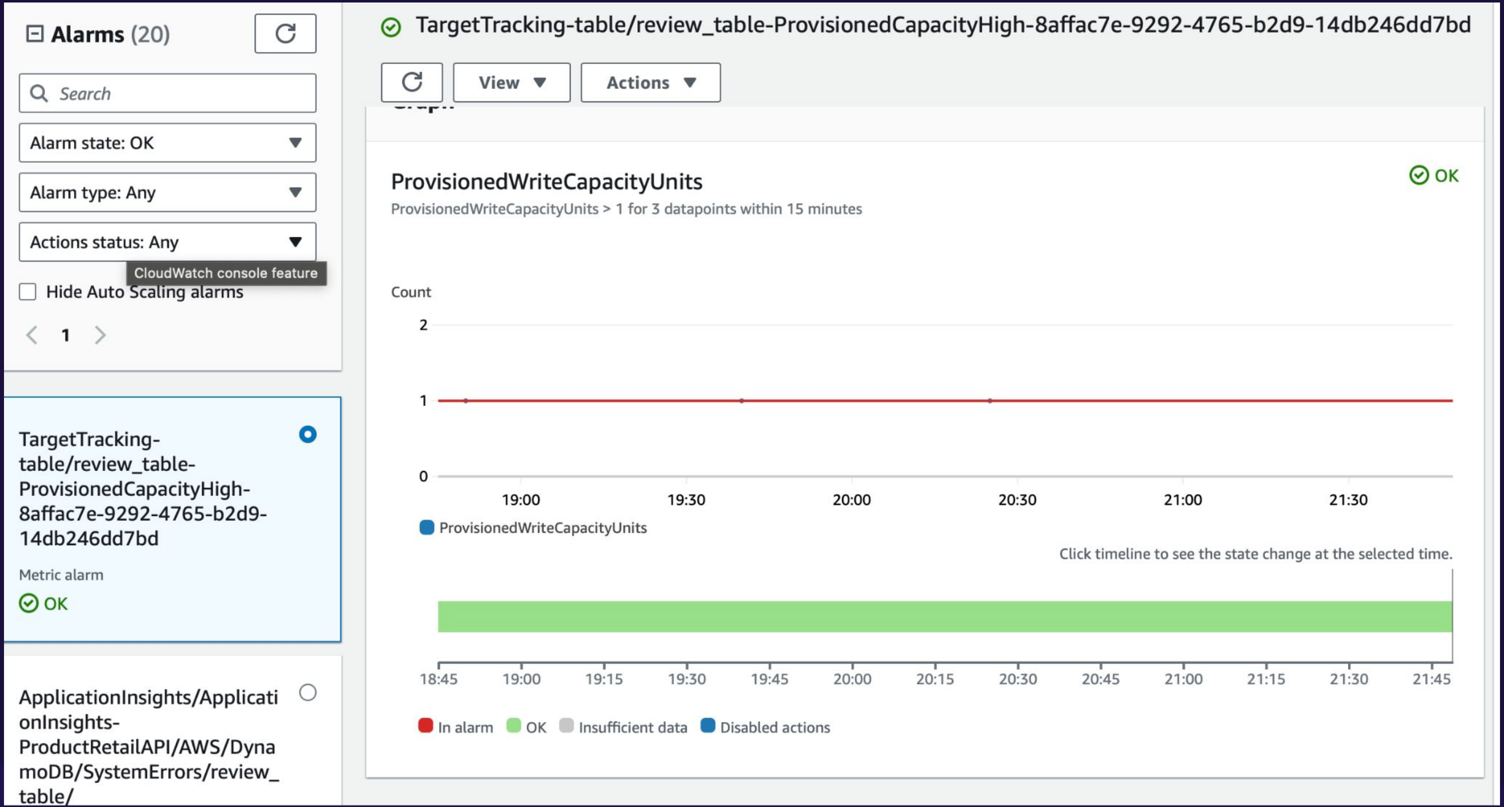
ApplicationInsights/ApplicationInsights.

Throttles >= 0.1 for 2 datapoints within 1...

ApplicationInsights/ApplicationInsights.

Errors >= 0.1 for 2 datapoints within 10 ...

Cloudwatch pt 2



Alarms (20)

Search

Alarm state: OK

Alarm type: Any

Actions status: Any

CloudWatch console feature

☐ Hide Auto Scaling alarms

< 1 >

TargetTracking-table/review_table-ProvisionedCapacityHigh-8affac7e-9292-4765-b2d9-14db246dd7bd

Metric alarm

✓ OK

ApplicationInsights/ApplicationInsights-ProductRetailAPI/AWS/DynamoDB/SystemErrors/review_table/

✓ TargetTracking-table/review_table-ProvisionedCapacityHigh-8affac7e-9292-4765-b2d9-14db246dd7bd

View ▼ Actions ▼

ProvisionedWriteCapacityUnits

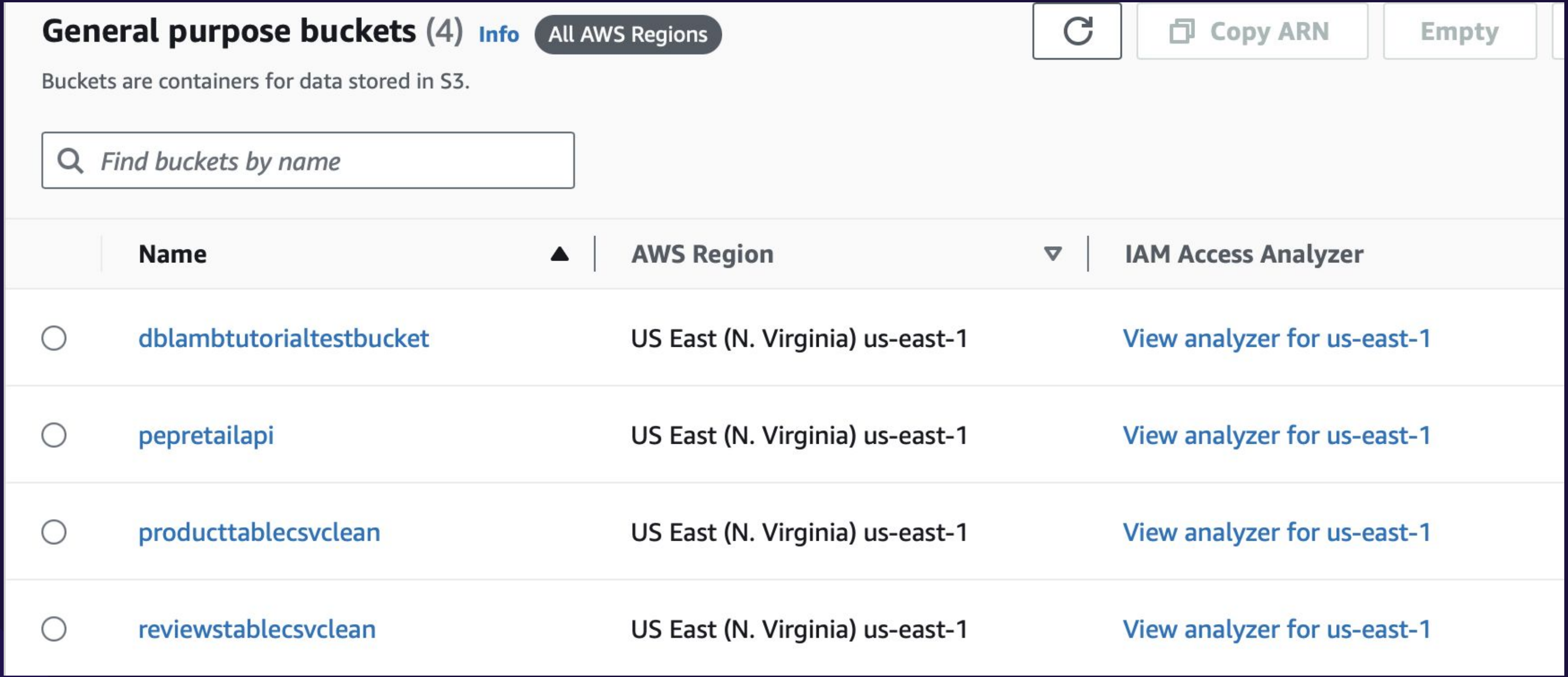
ProvisionedWriteCapacityUnits > 1 for 3 datapoints within 15 minutes

✓ OK

Click timeline to see the state change at the selected time.

🟥 In alarm 🟩 OK ▨ Insufficient data 🟦 Disabled actions

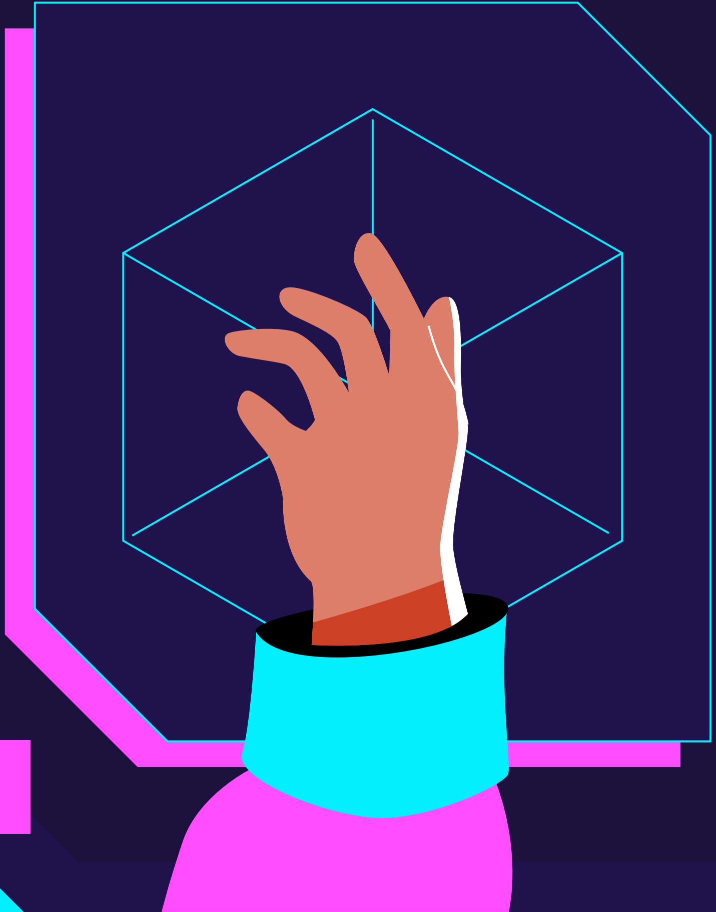
S3



General purpose buckets (4) Info All AWS Regions

Buckets are containers for data stored in S3.

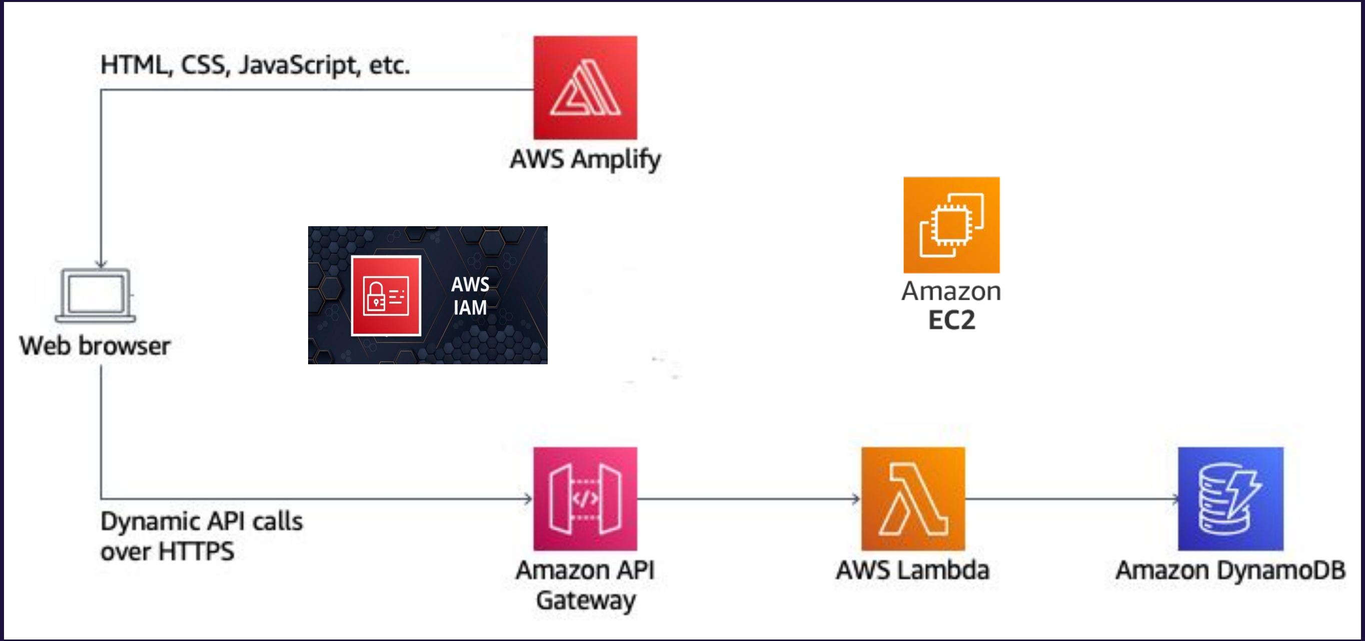
	Name	AWS Region	IAM Access Analyzer
○	dblambtutorialtestbucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1
○	pepretailapi	US East (N. Virginia) us-east-1	View analyzer for us-east-1
○	producttablecsvclean	US East (N. Virginia) us-east-1	View analyzer for us-east-1
○	reviewstablecsvclean	US East (N. Virginia) us-east-1	View analyzer for us-east-1



05

Analytics and Visualization

Serverless Architecture

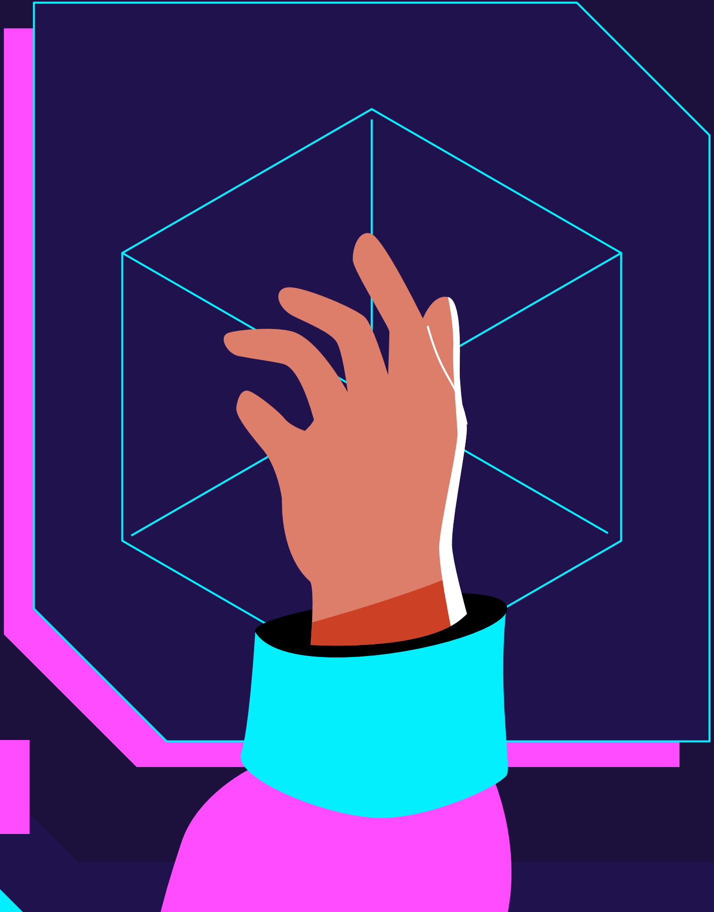


AWS Serverless

LIVE DEMO

06

Challenges and Triumphs



- CloudWatch underlying charges (DynamoDB Table Alerts, etc.)
- Challenging and Unexpected Team Changes
- Redesigned Front End

THANKS!

Do you have any questions?