**Cloud Computing Bootcamp**

**Project Title**

Real-Time Analytics Dashboard for Small Businesses

**Team name**

Code Legends

**Instructors**

Mr. Fredrick, Ms. Vivian

**Group Members**

1. Okwudili J Monye
2. Teslim Oyedotun
3. Lawson Ifeoma
4. Oniyelu Fausat Oluwabunmi
5. Fred Kanwai
6. Banjoko Oluwadamife
7. Alabi, Ashraf A.
8. Anaesiuba Amaka
9. Babatunde Obafemi

**Real-Time Analytics Dashboard Documentation**

**Project overview**.

The Real-Time Analytics Dashboard is designed to provide small business owners with up-to-date insights into their operations. Built on AWS, this system leverages services like Amazon Kinesis, Lambda, DynamoDB, S3, QuickSight, and SNS to collect, process, and visualize data in real-time.

**Project Title Brief**

A **Real-Time Analytics Dashboard for Small Businesses** is a project designed to provide small business owners with instant, actionable insights from their data, helping them make data-driven decisions. Here’s a breakdown of what this type of project entails:

**1. Purpose and Objectives**

* **Immediate Insights**: The dashboard provides real-time data insights, allowing businesses to monitor important metrics (e.g., sales, customer behavior, inventory levels) as they happen.
* **Improved Decision-Making**: With up-to-the-minute data, business owners can respond quickly to trends, demand shifts, or operational issues.
* **Resource Optimization**: Small businesses can maximize efficiency by understanding patterns and adjusting immediately, optimizing resources like inventory, staffing, and marketing.

**2. Key Features of the Dashboard**

* **Real-Time Data Visualization**: Displaying metrics such as sales volume, revenue, customer visits, and website traffic in an easily digestible format with charts, graphs, and tables.
* **Customizable Metrics**: Allowing users to select key performance indicators (KPIs) relevant to their business, like sales by category, customer demographics, or top-selling products.
* **Alerts and Notifications**: Automated notifications for critical thresholds (e.g., low inventory or sudden spikes in traffic), enabling quick response to changing business conditions.
* **Historical and Comparative Views**: Combining real-time data with historical trends for context, helping businesses compare performance over time.

**3. Target Users**

* **Small Business Owners and Managers**: Especially useful for those in retail, e-commerce, and other customer-facing businesses.
* **Sales and Marketing Teams**: Can leverage insights to adjust campaigns or promotions based on real-time customer engagement.
* **Inventory Managers**: Monitor stock levels to prevent shortages and overstock, especially during high-demand periods.

**4. Typical Data Sources**

* **Point of Sale (POS) Systems**: For sales transactions and revenue data.
* **Website and App Analytics**: Data on user behavior, page views, and online interactions.
* **Customer Relationship Management (CRM) Systems**: For insights into customer demographics and purchasing patterns.
* **Inventory Management Systems**: To track stock levels and monitor product turnover.

**5. Benefits for Small Businesses**

* **Enhanced Responsiveness**: Owners can address operational issues (e.g., low stock, high traffic) right away, reducing downtime and lost sales.
* **Informed Resource Allocation**: Real-time insights allow better allocation of resources, such as adjusting staff levels based on customer foot traffic.
* **Competitive Edge**: Small businesses can quickly act on trends or customer feedback, which is especially advantageous in competitive markets.

**6. Example Use Cases**

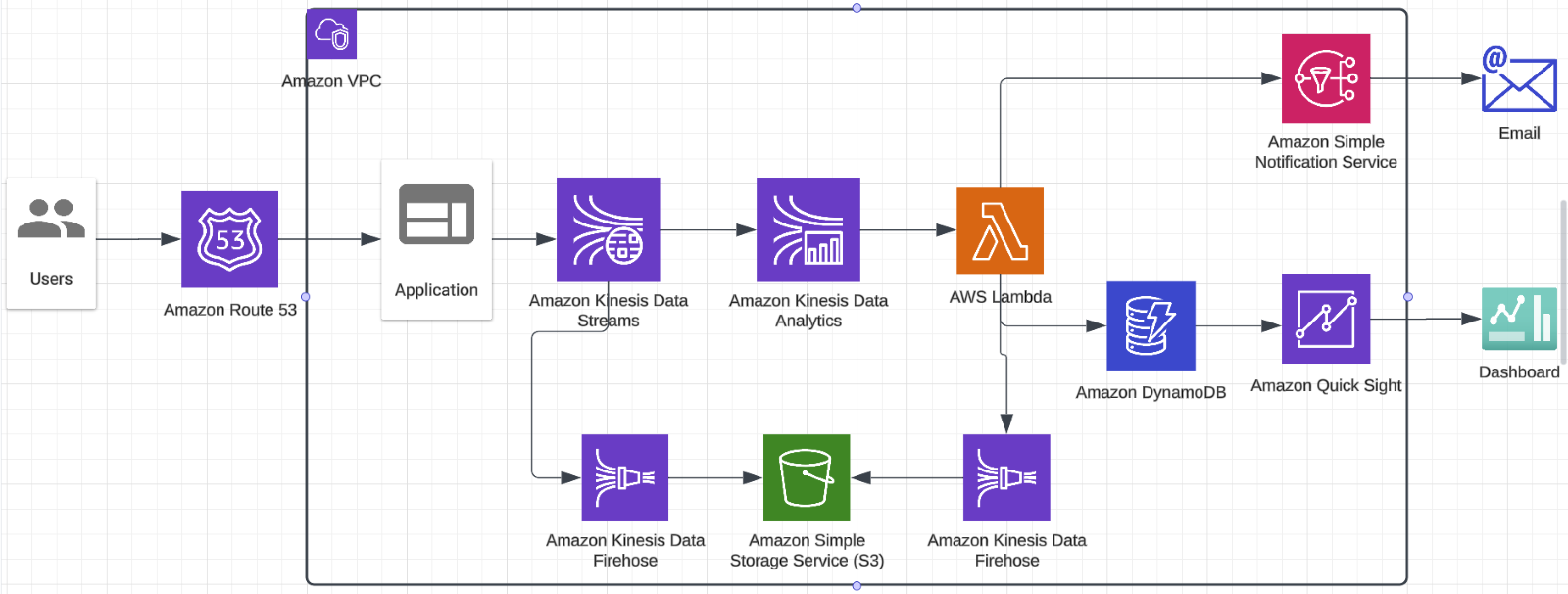
* **Retail Store Dashboard**: Displaying metrics like daily sales, inventory levels, and customer flow, helping owners manage store operations.
* **E-commerce Platform Monitoring**: Tracking real-time website traffic, cart abandonment rates, and popular products.
* **Customer Behavior Analytics**: Monitoring product popularity, customer demographics, and purchase frequency to help with targeted marketing.

In summary, a Real-Time Analytics Dashboard for Small Businesses empowers owners with timely, relevant data insights to boost operational efficiency, improve customer satisfaction, and enhance strategic planning.

**Technical Overview**.

**Architecture and Workflow**

**Architecture Diagram**

****

The architecture consists of the following main components:

1. **Amazon Kinesis** - Streams incoming data for real-time processing.
2. **AWS Lambda** - Processes streaming data and performs ETL tasks.
3. **Amazon DynamoDB** - Stores processed data for low-latency retrieval.
4. **Amazon S3** - Stores historical data and backup for analytics.
5. **Amazon QuickSight** - Provides data visualization and analytics.
6. **Amazon SNS** - Sends alerts and notifications based on business-defined events.

**Workflow Overview**

1. **Route 53**: Acts as the domain name system (DNS) and routes all incoming traffic from users to

the dashboard, hosted on AWS.

1. **Amazon Kinesis**: Captures real-time data streams from different sources (e.g., website activities,

point-of-sale transactions, social media metrics).

1. **AWS Lambda**: Processes incoming data streams, filters, and transforms data in real time before

passing it to other services. It handles real-time computation and enables seamless integration

with downstream services.

1. **Amazon S3**: Stores processed and raw data for analytics and reporting purposes. Data is

organized and managed for efficient retrieval by other services like QuickSight.

1. **Amazon DynamoDB**: Stores metadata, user settings, and real-time statistics. It provides fast,

low-latency storage to support live dashboard updates.

1. **Amazon QuickSight**: Visualizes data from S3 and DynamoDB, creating dynamic dashboards

with charts and insights that update in real time for the end user.

1. **Amazon SNS**: Sends notifications and alerts to users based on specific data events or thresholds,

improving user engagement and awareness.



**Component Documentation**

Each component in the architecture plays a crucial role in data processing, storage, and user interaction.

* 1. **Route 53**
* **Purpose**: Routes traffic to the dashboard.
* **Configuration**: DNS records point to the application load balancer or relevant endpoints.
* **Setup Notes**: Ensure that appropriate records (e.g., A or CNAME) are configured to match your

AWS resources.

* 1. **Amazon Kinesis**
* **Purpose**: Collects and streams data from various sources in real time.
* **Configuration**: Stream setup includes data sources, partition keys, and shard count adjustments.
* **Data Flow**: Ingested data flows from Kinesis to Lambda for real-time processing.
  1. **AWS Lambda**
* **Purpose**: Processes data from Kinesis streams, transforming it before sending it to other

services.

* **Configuration**: Set up Lambda functions to handle data processing tasks, with proper IAM roles

for access to Kinesis, S3, DynamoDB, and SNS.

* **Notes**: Lambda is set to run in real time, triggered by Kinesis events.
  1. **Amazon S3**
* **Purpose**: Stores both raw and processed data.
* **Configuration**: Organize data into logical folders/buckets based on usage (e.g., raw data,

transformed data).

* **Access**: S3 is accessed by QuickSight for data visualization and by Lambda for data processing.
  1. **Amazon DynamoDB**
* **Purpose**: Provides fast, scalable storage for metadata and user-specific settings.
* **Configuration**: Tables include metadata for user sessions, settings, and real-time analytics

summaries.

* **Notes**: DynamoDB is configured to handle high read and write throughput for low-latency

requirements.

* 1. **Amazon QuickSight**
* **Purpose**: Visualizes and analyzes data, presenting it in dashboards tailored to small business

needs.

* **Configuration**: Data sources are connected to S3 and DynamoDB for real-time visualization.
* **Features**: Includes interactive charts, data filters, and user-defined parameters.
  1. **Amazon SNS**
* **Purpose**: Sends automated notifications and alerts to users based on set conditions.
* **Configuration**: Topics and subscriptions are created to trigger notifications, with email.

**Resource Provisioning**

**VPC and Networking**

For the real-time analytics project on AWS, a Virtual Private Cloud (VPC) was created, divided into public and private subnets across multiple Availability Zones for high availability. An Internet Gateway was attached to allow public subnet resources internet access, while a NAT Gateway enabled private subnet resources to access the internet securely. Route tables were configured to manage traffic flow between subnets and the internet. Security groups and Network ACLs provided granular control over inbound and outbound traffic for resources. Additionally, VPC Flow Logs were enabled to monitor and troubleshoot network traffic, ensuring the infrastructure is secure, scalable, and efficient.

**CloudFormation Template**

**Objective**

Automate the setup of all resources using AWS CloudFormation for consistency, ease of deployment, and repeatability.

**Template Structure:**

* **VPC and Networking:**

Define subnets, route tables, security groups, and any NAT configurations if required.

* **EC2:**

Deploy EC2, setup the SG with the appropriate ingress and egress permisions

* **Data Stores:**

Define DynamoDB tables with necessary read and write capacities, indexes, and permissions.

* **S3 Configuration:**

Set up S3 buckets for the data lake with appropriate access policies and lifecycle rules for efficient storage management.

* **Kinesis:**

Setting up the Kinessis Data stream, Data Analytics and Firehose for proper data pipeline.

* **IAM Roles and Permissions:**

Ensure minimal permissions using IAM roles and policies for each component (Lambda, S3, DynamoDB, and QuickSight) for security and efficiency.

* **QuickSight Setup:**

Where feasible, automate the permissions QuickSight needs to access S3 and DynamoDB using the CloudFormation stack output.

**Template Architecture**

**Format: json**

{

    "AWSTemplateFormatVersion": "2010-09-09",

    "Resources": {

      "CLDashboard1": {

        "Type": "AWS::EC2::Subnet",

        "Properties": {

          "VpcId": "vpc-0d300066e07e8394a",

          "CidrBlock": "10.0.8.0/24",

          "AvailabilityZone": "us-east-1a",

          "MapPublicIpOnLaunch": true

        }

      },

      "CLDashboardSG": {

        "Type": "AWS::EC2::SecurityGroup",

        "Properties": {

          "GroupDescription": "Enable HTTP HTTPS SSH access",

          "VpcId": "vpc-0d300066e07e8394a",

          "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"

            },

            {

              "IpProtocol": "tcp",

              "FromPort": 443,

              "ToPort": 443,

              "CidrIp": "0.0.0.0/0"

            }

          ]

        }

      },

      "CLDashboardHZ": {

        "Type": "AWS::Route53::HostedZone",

        "Properties": {

          "Name": "thecldashboard.click"

        }

      },

      "CLDashboardEC2": {

        "Type": "AWS::EC2::Instance",

        "Properties": {

          "ImageId": "ami-06b21ccaeff8cd686",

          "InstanceType": "t2.micro",

          "SecurityGroupIds": [

            {

              "Ref": "CLDashboardSG"

            }

          ],

          "SubnetId": {

            "Ref": "CLDashboard1"

          },

          "UserData": {

            "Fn::Base64": {

              "Fn::Join": [

                "",

                [

                  "#!/bin/bash\n",

                  "yum update -y\n",

                  "yum install -y python3\n",

                  "aws s3 cp s3://my-code-bucket/my-application.zip /home/ec2-user/\n",

                  "unzip /home/ec2-user/my-application.zip -d /home/ec2-user/my-app\n",

                  "cd /home/ec2-user/my-app\n",

                  "python3 app.py &\n"

                ]

              ]

            }

          }

        }

      },

      "CLDashboardKS": {

        "Type": "AWS::Kinesis::Stream",

        "Properties": {

          "Name": "CL-Dashboard-Data-Stream",

          "ShardCount": 1

        }

      },

      "CLDashboardKA": {

        "Type": "AWS::KinesisAnalyticsV2::Application",

        "Properties": {

          "ApplicationName": "cl-dashboard-analytics-app",

          "RuntimeEnvironment": "SQL-1\_0",

          "ServiceExecutionRole": {

            "Fn::GetAtt": [

              "AnalyticsExecutionRole",

              "Arn"

            ]

          },

          "ApplicationConfiguration": {

            "SqlApplicationConfiguration": {

              "Inputs": [

                {

                  "NamePrefix": "input",

                  "InputSchema": {

                    "RecordFormat": {

                      "RecordFormatType": "JSON",

                      "MappingParameters": {

                        "JSONMappingParameters": {

                          "RecordRowPath": "$"

                        }

                      }

                    },

                    "RecordColumns": [

                      {

                        "Name": "EmployeeID",

                        "SqlType": "VARCHAR(64)",

                        "Mapping": "$.EmployeeID"

                      },

                      {

                        "Name": "Name",

                        "SqlType": "VARCHAR(64)",

                        "Mapping": "$.Name"

                      },

                      {

                        "Name": "JobTitle",

                        "SqlType": "VARCHAR(64)",

                        "Mapping": "$.JobTitle"

                      },

                      {

                        "Name": "HireDate",

                        "SqlType": "VARCHAR(64)",

                        "Mapping": "$.HireDate"

                      },

                      {

                        "Name": "Salary",

                        "SqlType": "INTEGER",

                        "Mapping": "$.Salary"

                      },

                      {

                        "Name": "Shift",

                        "SqlType": "VARCHAR(64)",

                        "Mapping": "$.Shift"

                      }

                    ]

                  },

                  "KinesisStreamsInput": {

                    "ResourceARN": {

                      "Fn::GetAtt": [

                        "CLDashboardKS",

                        "Arn"

                      ]

                    }

                  }

                }

              ]

            }

          }

        }

      },

      "AnalyticsExecutionRole": {

        "Type": "AWS::IAM::Role",

        "Properties": {

          "RoleName": "KinesisAnalyticsExecutionRole",

          "AssumeRolePolicyDocument": {

            "Version": "2012-10-17",

            "Statement": [

              {

                "Effect": "Allow",

                "Principal": {

                  "Service": "kinesisanalytics.amazonaws.com"

                },

                "Action": "sts:AssumeRole"

              }

            ]

          },

          "Policies": [

            {

              "PolicyName": "KinesisAnalyticsPolicy",

              "PolicyDocument": {

                "Version": "2012-10-17",

                "Statement": [

                  {

                    "Effect": "Allow",

                    "Action": [

                      "kinesis:DescribeStream",

                      "kinesis:GetRecords",

                      "kinesis:GetShardIterator"

                    ],

                    "Resource": "\*"

                  }

                ]

              }

            }

          ]

        }

      },

      "CLDashboardDBTable": {

        "Type": "AWS::DynamoDB::Table",

        "Properties": {

          "TableName": "CLDashboardAnalyticsTable",

          "AttributeDefinitions": [

            {

              "AttributeName": "ID",

              "AttributeType": "S"

            }

          ],

          "KeySchema": [

            {

              "AttributeName": "ID",

              "KeyType": "HASH"

            }

          ],

          "BillingMode": "PAY\_PER\_REQUEST"

        }

      },

      "MiniMartProducts": {

        "Type": "AWS::DynamoDB::Table",

        "Properties": {

          "TableName": "MiniMartProducts",

          "AttributeDefinitions": [

            {

              "AttributeName": "ItemID",

              "AttributeType": "S"

            }

          ],

          "KeySchema": [

            {

              "AttributeName": "ItemID",

              "KeyType": "HASH"

            }

          ],

          "BillingMode": "PAY\_PER\_REQUEST"

        }

      },

      "MiniMartEmployees": {

        "Type": "AWS::DynamoDB::Table",

        "Properties": {

          "TableName": "MiniMartEmployees",

          "AttributeDefinitions": [

            {

              "AttributeName": "EmployeeID",

              "AttributeType": "S"

            }

          ],

          "KeySchema": [

            {

              "AttributeName": "EmployeeID",

              "KeyType": "HASH"

            }

          ],

          "BillingMode": "PAY\_PER\_REQUEST"

        }

      },

      "DataForProducts": {

        "Type": "AWS::Lambda::Function",

        "Properties": {

          "FunctionName": "InsertMiniMartProductData",

          "Runtime": "python3.8",

          "Role": "arn:aws:iam::565393029077:role/Lambda\_DB\_Access",

          "Handler": "lambda\_function.lambda\_handler",

          "Code": {

            "ZipFile": {

              "Fn::Join": [

                "",

                [

                  "import json\n",

                  "import boto3\n",

                  "def lambda\_handler(event, context):\n",

                  "    dynamodb = boto3.resource('dynamodb')\n",

                  "    table = dynamodb.Table('MiniMartProducts')\n",

                  "    products = [\n",

                  "        {\"ItemID\": \"P001\", \"Name\": \"Apple\", \"Category\": \"Fruit\", \"Price\": 0.5, \"StockQuantity\": 150, \"Supplier\": \"Fresh Farms\"},\n",

                  "        {\"ItemID\": \"P002\", \"Name\": \"Milk\", \"Category\": \"Dairy\", \"Price\": 1.2, \"StockQuantity\": 50, \"Supplier\": \"DairyBest\"},\n",

                  "        {\"ItemID\": \"P003\", \"Name\": \"Bread\", \"Category\": \"Bakery\", \"Price\": 2.0, \"StockQuantity\": 30, \"Supplier\": \"BakeryCo\"},\n",

                  "        {\"ItemID\": \"P004\", \"Name\": \"Orange Juice\", \"Category\": \"Beverage\", \"Price\": 2.5, \"StockQuantity\": 40, \"Supplier\": \"JuiceHub\"}\n",

                  "    ]\n",

                  "    for product in products:\n",

                  "        table.put\_item(Item=product)\n",

                  "    return {\n",

                  "        'statusCode': 200,\n",

                  "        'body': json.dumps('Inserted products')\n",

                  "    }\n"

                ]

              ]

            }

          },

          "MemorySize": 128,

          "Timeout": 10

        }

      },

      "DataForEmployees": {

        "Type": "AWS::Lambda::Function",

        "Properties": {

          "FunctionName": "InsertMiniMartEmployeeData",

          "Runtime": "python3.8",

          "Role": "arn:aws:iam::565393029077:role/Lambda\_DB\_Access",

          "Handler": "lambda\_function.lambda\_handler",

          "Code": {

            "ZipFile": {

              "Fn::Join": [

                "",

                [

                  "import json\n",

                  "import boto3\n",

                  "def lambda\_handler(event, context):\n",

                  "    dynamodb = boto3.resource('dynamodb')\n",

                  "    table = dynamodb.Table('MiniMartEmployees')\n",

                  "    employees = [\n",

                  "        {\"EmployeeID\": \"E001\", \"Name\": \"Oluwadamife\", \"JobTitle\": \"Manager\", \"HireDate\": \"2021-06-15\", \"Salary\": 60000, \"Shift\": \"Morning\"},\n",

                  "        {\"EmployeeID\": \"E002\", \"Name\": \"Oniyelu\", \"JobTitle\": \"Cashier\", \"HireDate\": \"2022-01-10\", \"Salary\": 30000, \"Shift\": \"Afternoon\"},\n",

                  "        {\"EmployeeID\": \"E003\", \"Name\": \"Ashraf\", \"JobTitle\": \"Stock Clerk\", \"HireDate\": \"2020-04-22\", \"Salary\": 28000, \"Shift\": \"Night\"}\n",

                  "        {\"EmployeeID\": \"E003\", \"Name\": \"Joan\", \"JobTitle\": \"Accountant\", \"HireDate\": \"2022-04-22\", \"Salary\": 60000, \"Shift\": \"All Day\"}\n",

                  "        {\"EmployeeID\": \"E003\", \"Name\": \"Daniela\", \"JobTitle\": \"Marketer\", \"HireDate\": \"2022-12-22\", \"Salary\": 45000, \"Shift\": \"Morning\"}\n",

                  "        {\"EmployeeID\": \"E003\", \"Name\": \"Ifeanyi\", \"JobTitle\": \"Auditor\", \"HireDate\": \"2021-06-22\", \"Salary\": 49000, \"Shift\": \"Morning\"}\n",

                  "    ]\n",

                  "    for employee in employees:\n",

                  "        table.put\_item(Item=employee)\n",

                  "    return {\n",

                  "        'statusCode': 200,\n",

                  "        'body': json.dumps('Inserted employees')\n",

                  "    }\n"

                ]

              ]

            }

          },

          "MemorySize": 128,

          "Timeout": 10

        }

      },

      "CLDashboardSNSTopic": {

        "Type": "AWS::SNS::Topic",

        "Properties": {

          "TopicName": "MiniMartNotifications-1"

        }

      },

      "CLDashboardSNSsub": {

        "Type": "AWS::SNS::Subscription",

        "Properties": {

          "TopicArn": {

            "Ref": "CLDashboardSNSTopic"

          },

          "Protocol": "email",

          "Endpoint": "[lawsonifeoma123@gmail.com](mailto:lawsonifeoma123@gmail.com)"

        }

      },

      "EventRuleForProducts": {

        "Type": "AWS::Events::Rule",

        "Properties": {

          "ScheduleExpression": "rate(1 hour)",

          "Targets": [

            {

              "Arn": {

                "Fn::GetAtt": [

                  "DataForProducts",

                  "Arn"

                ]

              },

              "Id": "TargetFunctionV1"

            }

          ]

        }

      },

      "EventRuleForEmployees": {

        "Type": "AWS::Events::Rule",

        "Properties": {

          "ScheduleExpression": "rate(1 hour)",

          "Targets": [

            {

              "Arn": {

                "Fn::GetAtt": [

                  "DataForEmployees",

                  "Arn"

                ]

              },

              "Id": "TargetFunctionV1"

            }

          ]

        }

      },

      "PermissionForProducts": {

        "Type": "AWS::Lambda::Permission",

        "Properties": {

          "Action": "lambda:InvokeFunction",

          "FunctionName": {

            "Ref": "DataForProducts"

          },

          "Principal": "events.amazonaws.com",

          "SourceArn": {

            "Fn::GetAtt": [

              "EventRuleForProducts",

              "Arn"

            ]

          }

        }

      },

      "PermissionForEmployees": {

        "Type": "AWS::Lambda::Permission",

        "Properties": {

          "Action": "lambda:InvokeFunction",

          "FunctionName": {

            "Ref": "DataForEmployees"

          },

          "Principal": "events.amazonaws.com",

          "SourceArn": {

            "Fn::GetAtt": [

              "EventRuleForEmployees",

              "Arn"

            ]

          }

        }

      }

    },

    "Outputs": {

      "MiniMartProductsTableName": {

        "Description": "Name of the MiniMart Products table",

        "Value": {

          "Ref": "MiniMartProducts"

        }

      },

      "MiniMartEmployeesTableName": {

        "Description": "Name of the MiniMart Employees table",

        "Value": {

          "Ref": "MiniMartEmployees"

        }

      },

      "AnalyticsAppArn": {

        "Description": "ARN of the Kinesis Analytics Application",

        "Value": {

          "Ref": "CLDashboardKA"

        }

      },

      "DynamoDBTableArn": {

        "Description": "ARN of the DynamoDB Table",

        "Value": {

          "Fn::GetAtt": [

            "CLDashboardDBTable",

            "Arn"

          ]

        }

      },

      "SNSTopicArn": {

        "Description": "ARN of the SNS Topic",

        "Value": {

          "Ref": "CLDashboardSNSTopic"

        }

      }

    }

**Data Management and Visualization**

**DynamoDB Data Export**

**Objective**

Enable seamless export of real-time data from DynamoDB to Amazon S3 at intervals for archival and extended analysis.

**Implementation Details:**

* Use DynamoDB Streams for continuous data streaming to an intermediary (e.g., AWS Lambda or Kinesis Data Firehose) to trigger export.
* Set up the export pipeline to S3 with batching and transformation, such as JSON-to-Parquet conversion for efficient querying and storage.
* Consider lifecycle policies on the S3 bucket to move older data to lower-cost storage tiers (e.g., S3 Glacier) as it ages.

**Dynamo DB Table update via AWS CLI**

To update a DynamoDB table, particularly one storing product and employee information, Python’s boto3 library is highly effective. boto3 allows users to interact with DynamoDB using Python code by calling functions that can retrieve, modify, and update table items. For this method, Python Package was installed on the EC2 so new data can easily be updated via the CLI.

**S3 Data Lake Configuration**

**Objective**

Configure an S3 data lake to act as the historical repository for analytics.

**Components:**

* **Bucket Structure:** Organize buckets by data type and timestamp for easy retrieval and data management.
* **Partitioning Strategy:** Partition data based on business-relevant attributes, such as date or category, to optimize query performance.
* **Access Policies:** Implement bucket policies that allow QuickSight, and potentially other analytics tools, secure access to the data lake.
* **Data Transformation:** Convert raw data into columnar formats, like Parquet, to reduce storage costs and accelerate query performance when accessed from QuickSight or other analytics tools.

**QuickSight Dashboard Configuration**

**Objective**

Create a QuickSight dashboard to deliver real-time and historical insights to end-users.

**Dashboard Setup:**

* **Data Sources:** Connect QuickSight to both DynamoDB (for real-time views) and S3 (for historical analysis).
* **Data Preparation:** Use SPICE (Super-fast, Parallel, In-memory Calculation Engine) for optimized queries, ensuring that large data sets from S3 are pre-aggregated for faster rendering.
* **Dashboard Layout:** Define key metrics and visualizations that are relevant to small business needs, such as sales trends, customer activity, and operational efficiency metrics.
* **Access Control:** Configure user and group access policies in QuickSight to secure data visibility based on roles or departments, if applicable.
* **Monitoring and Alerts:** Set up metrics and alerts to inform users of key changes, such as spikes in usage, errors, or unexpected trends.

**Project Issues Encountered.**

**AWS Lambda**

During the integration of AWS Lambda with other AWS services for our real-time analytics dashboard, we encountered a critical issue with the AWS SDK. Specifically, the AWS SDK could not be added as a layer, as AWS has phased out the SDK as a managed resource on Lambda. This change required us to adjust our approach by using AWS CloudFormation for managing dependencies and integrating the SDK in our Lambda functions.

**Solution:**   
To resolve this issue, we opted to take the CloudFormation route. In our setup, we have configured CloudFormation templates that:

1. **Download and package the AWS SDK**: We created a custom Lambda layer within the

CloudFormation template, which includes the AWS SDK packaged manually.

1. **Attach the Custom Layer to Lambda Functions**: This layer is then attached to our Lambda

functions, providing access to the SDK without relying on the previously managed layer

resource.

1. **Define SDK Version**: By packaging the SDK ourselves, we control the version used, which

offers flexibility for updating the SDK as new versions become available without requiring

managed updates.

This approach maintains the functionality required for our analytics dashboard while ensuring compliance with AWS’s current policies on SDK usage within Lambda functions.

**Amazon Quicksight**

When connecting Amazon QuickSight to S3 to visualize data, an issue arose due to the DynamoDB `manifest.js` file not being properly formatted as JSON. This formatting discrepancy caused QuickSight to encounter errors in processing the data, leading to visualization issues. Proper JSON formatting is required for successful data ingestion and rendering in QuickSight.

**Solution**

To resolve this issue, we opted to downloading the file and formatting it properly with the aid of VS Code and Chat GPT Ai Asst. The we uploaded the fille via the quicksight dashboard abd we successfully visualized the data from our DynamoDB.

**Publish Visuals from Amazon QuickSight to a Domain**

The connection between Amazon QuickSight and the S3 data source was successfully established, enabling streamlined data visualization. Additionally, the URL of the QuickSight dashboard was hosted under a designated domain name, ensuring convenient and secure access to the analytics dashboard for customers. This setup allows for direct, user-friendly access to real-time insights through a centralized platform.

**Conclusion**

An AWS Cloud template was successfully created to automate the process of reading, writing, and storing data in DynamoDB. This template leverages AWS Lambda functions and the AWS CLI, offering a flexible and scalable solution for interacting with DynamoDB without manual updates. Lambda functions facilitate seamless, serverless execution of tasks such as processing incoming data or updating records in real time, while the AWS CLI provides an alternative method for direct command-line interactions with the database.

Additionally, successful data visualization was achieved by connecting the S3 data lake to Amazon QuickSight. This setup allows for quick, intuitive analysis of the data stored in S3, providing real-time insights into various metrics. The integration supports auditing and monitoring of the stored data, with customizable dashboards and reports to highlight key performance indicators and trends, ensuring that the data is not only accessible but also actionable for decision-making and continuous analysis.

**Results of Project.**

During the development of the Real-Time Analytics Dashboard for small businesses, the team was exposed to a variety of both familiar and new AWS resources. These included services such as Kinesis (Data Streams & Analytics), Lambda, CloudFormation, AWS Firehose, Amazon QuickSight, DynamoDB, SNS, EC2, S3, VPC, IAM, AWS CLI, among others. Each of these resources played a crucial role in building the infrastructure, data processing pipelines, and visualization components required for the dashboard.

The team members actively explored the functionality of these services, learning how they contribute to the success of the project. This was achieved through in-depth research of AWS documentation, collaborative problem-solving, and the use of AI tools like ChatGPT and Amazon Q to assist in understanding key concepts and best practices. As a result, the team gained foundational knowledge of AWS Cloud services and their applications, gaining practical experience in leveraging these resources for real-time data analytics. The project has not only enhanced individual skills but also sparked a desire to continue learning and expanding expertise in other AWS resources and their capabilities, with the aim of tackling more advanced use cases in the future.

**Questions.**

* 1. What approach would have done this project faster?
  2. Can the dataset we publish on our Quicksight Dashboard be shown to the public internet?
     + If yes, how?
  3. What could we have done better?
  4. In a real-life scenario what is the role of a Cloud Engineer in this type of project in a larger scale. i.e If we were hired by Shoprite to set up a similar dashboard for their branch in a new area.

**Tools Used for this Project**

1. **Jira**

A project management tool that helps in tracking issues, managing tasks, and organizing documentation workflows collaboratively. It allows teams to create, assign, and monitor documentation-related tasks efficiently.

1. **AI Tool (ChatGPT)**

An AI-powered language model used for generating content, rewriting text, and providing assistance in structuring documentation. It enhances productivity by quickly producing high-quality written materials.

1. **Lucidchart**

A diagramming tool that enables users to create flowcharts, diagrams, and visual representations of processes or architectures. It is useful for illustrating concepts and structures in the documentation effectively.

1. Amazon Q

An integrated AI in the AWS Console was used to figure out some minor issues we faced setting up some resources needed for our project.

1. VS Code

We used this IDE to format our cloud formation template properly to avoid error in deployment phase.

1. Microsoft Teams.

We used this as a medium of keeping up with everyone’s duties at the end of each day to make sure everyone is doing their duties and brought up to speed.