A01184153

akao

# Overview

As a senior solutions architect hired by Magical Unicorns LTD, it was asked to reduce operational overhead with a serverless solution. A new division called the Magical Unicorns for Big Monies was created and works with personal data. Hence, this data needs to be secure. The current prototype uses an Express NodeJS application with data stored locally. The application is used to track and analyze purchases of consumers to encourage better spending. Data is provided by payment card issuers over SFTP. In the future, teams will probably need to be created to cross analyze this data with other datasets.

A solution in the attached PowerPoint is presented. This solution was carefully constructed in order to ensure that it was completely serverless (eg. No EC2 instances).

# Diagram Explanation

Payment card issuers continue to transfer personal data using SFTP, but this time, we use an AWS service called SFTP transfer. This complements the Security pillar in the Well Architected Framework – “*Protect data in transit and at rest*”. This data is sent to an S3 bucket to securely store the data. We ensure that the S3 bucket is private and is only accessible via SFTP. When the data is sent, its contents and information are transferred over messages via SNS to 2 separate SQS queues asynchronously. One is used to send the information to data analysts to cross analyze the data in the future, and the other is polled by a lambda function continuously until it gets stored in a DynamoDB database. This polling happens at intervals between the lambda function and SQS for *Cost Optimization* (pillar) and is a built-in feature in SQS. This is done so that there is a timeout, and the lambda function doesn’t continuously run. On the client-side, the Express NodeJS site content is stored in an S3 bucket and delivered directly to the client. Amazon Cognito provides a simple and secure sign-up procedure to the website to access data via SAML 2.0. This allows us to authenticate users via custom-generated tokens instead of stored passwords by allowing them to login using OAuth. Cognito then is used as input into a lambda function that peaks at the data in the DynamoDB database.

# Well-architected framework

This section discusses how the solution follows the well-architected principles of AWS.

## Operational Excellence

Because this is a fully serverless solution, we can isolate each resource used, and make minor changes. Thus, if there are any issues, we can isolate the resource, swap it out, or fix it.

## Security

As mentioned above, the personal data is transferred using SFTP into a private S3 bucket ensuring security. Cognito is also used to make it so we don’t have to manage user passwords, but auto-generated tokens for authentication.

## Reliability

Its easy to scale this solution up to accommodate reliability because it is serverless. Each part can be swapped out, so it is easy to manage change in automation, and recover from failure.

## Performance Efficiency

Serverless architectures boost performance efficiency and is a design principle listed under ‘Performance Efficiency’ on the AWS website. By creating a CloudFormation, the template should easily be deployed and used within minutes.

## Cost Optimization

It’s cheap to store data in S3, and the Lambda functions are accessed at only intervals. Lambda functions are only pay as you run, so the costs are fairly optimized in this solution.

# Cloudformation:

## Used in YAML:

1. S3 bucket: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-s3-bucket.html>
2. SNS Topic: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-sns-topic.html>
3. SNS Policy: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-sns-policy.html>
4. SNS Sub: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-sns-subscription.html>
5. IAM Role: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-role.html>
6. Lambda Func: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-function.html>

## Also needed in full architecture:

1. SFTP Transfer Server: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-transfer-server.html>
2. SFTP Transfer User: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-transfer-user.html>
3. DynamoDB Table: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-dynamodb-table.html>
4. Cognito: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cognito-identitypool.html>