Report of Building an administrative console in Amazon QuickSight to analyze usage metrics

Figure 1: The diagram of the workflow of the solution

Here's a simple breakdown of the workflow:

1. The AWS Lambda function, **Data\_Prepare**, runs every hour to gather information about QuickSight users, groups, and access permissions, saving this data to an S3 bucket.
2. **CloudTrail logs** are also stored in the same S3 bucket.
3. **Amazon Athena** then creates tables and views from this data, including user-group details, QuickSight asset permissions, and CloudTrail login events. These tables can be optionally joined with other tables for deeper insights.
4. **QuickSight datasets** pull the data from Athena tables using SPICE mode, and a **QuickSight dashboard** is generated based on these datasets for reporting and analysis.

Phase I: Setup  
After connecting to the AWS-provided Sandbox Account (thanks to Mme Meriem), I attempted to launch the AWS CloudFormation stack, which is supposed to set up the AWS Lambda, scheduler as an event, and IAM roles for permissions. However, after following the steps, the event logger displayed an error indicating that the configuration uses Python 3.7, which is no longer supported. As a result, the setup rolled-back.

Phase II: Connection   
I decided to use Terraform after setting it up with AWS CLI. I then downloaded the template from :  
<https://admin-console-dataprepare-code.s3.amazonaws.com/admin_console.template>  
and manually modified it to use Python 3.9 instead. Afterward, I created the stack with the local file using Terraform, so I wrote the following main.tf:

provider "aws" {

  region = "us-east-1"

}

resource "aws\_cloudformation\_stack" "quick\_sight\_admin\_console" {

  name         = "adminconsole"

  template\_body = file("admin\_console.template")

  capabilities  = ["CAPABILITY\_IAM", "CAPABILITY\_NAMED\_IAM"]

}

*A simple Terraform code to create the needed stack, which was successful.  
  
Phase III : Querying*

*Now that the stack is successfully created, we’re supposed to begin with the Lab from step 3 in Fig.1.*

*So, I wrote the following code:*

data "aws\_caller\_identity" "current" {}

resource "aws\_s3\_bucket" "hoge" {

  bucket = "admin-console-381491857320"

}

locals {

  create\_tables\_query = templatefile("create\_athena\_tables.sql", {

    account\_id = data.aws\_caller\_identity.current.account\_id

  })

}

resource "aws\_athena\_database" "hoge" {

  name   = "users"

  bucket = aws\_s3\_bucket.hoge.id

}

resource "aws\_athena\_named\_query" "create\_tables" {

  name      = "create\_athena\_tables"

  database  = aws\_athena\_database.hoge.name

  query     = local.create\_tables\_query

}

This connects to the AWS S3 bucket, which is supposed to contain the required CSV files, and runs the SQL file create\_athena\_tables.sql. This file contains the query from the documentation intended to create two external tables from the CSV files in the bucket.

However, after completing this step, running a SELECT on either of the tables shows that they are empty.  
  
Phase IV : Debugging  
I started investigating the source of the issue, first verifying if the query was running correctly, and indeed it was. Then, I checked if my Terraform setup was working properly, and it was. The S3 bucket was created, but unfortunately, it was empty and didn't contain the required CSV files. The S3 bucket is supposed to be populated by the Lambda function, so I examined the Python script. The script consists of a group of functions that handle the data retrieval and processing, with the most important being the lambda\_handler function, which performs the following tasks:

1. Initialization: Sets up AWS clients and variables, such as the account ID, region, and S3 bucket name.

2. Fetching Data: Retrieves data on namespaces, users, and their group memberships from QuickSight through various API calls. It also fetches access permissions for QuickSight resources like dashboards, datasets, data sources, analyses, and themes.

3. Processing Data: Writes the user-group membership information and access permissions to CSV files. These files are then uploaded to the S3 bucket.

4. Output: The result is two CSV files: one for group membership data and another for object access data, both uploaded to the S3 bucket.

In summary, the function collects user and permission data from QuickSight and stores it in S3 for monitoring or auditing purposes, and everything seemed to be working as expected.  
  
While checking QuickSight, I encountered a permissions error message. I'm unsure if it's truly a permissions issue or if it might stem from the change in Python version from 3.7 to 3.9, potentially causing some libraries to function differently than expected. I am still investigating the root cause of the issue. As Mr. Hassan indicated, AWS documentation is considered one of the best available globally, so it's likely an error on my end. I recognize that this current result is far from what was originally requested.

Phase V: Attempting

After multiple unsuccessful attempts to use the template to create the stack, I began manually creating the components that the stack was supposed to generate, one by one, using Terraform. These components included the S3 bucket, the IAM role with associated policies, the Lambda function, and the scheduler.

A screenshot of a computer

Description automatically generated

A black screen with green text

Description automatically generated  
  
While the creation and configuration of these individual components succeeded, the overall issue persists.  
  
Phase VI: Completing ???!!!  
I believe that the remaining tasks are straightforward and follow the same approach as running the queries on the tables. We will run the queries to populate the desired views in the same manner. With the Athena views ready, we can build some QuickSight datasets. We can load the view called admin\_console to create a SPICE dataset named admin\_console and schedule this dataset to be refreshed on an hourly basis.