Student Name- Siddhi Bhise

Student ID- 100946768

Course Code- COSC 1104-01

Course Name- Scripting

A green logo on a black background

Description automatically generated

Part 1: Identify the Problem (MS Word; Week 11/12)

Automating Cloud Resource Optimization to Reduce Costs

Introduction:

The challenges in the management of cloud costs have always faced most organizations that rely on AWS, Azure, or Google Cloud. These usually consist of unused virtual machines, unattached volumes, and over-provisioned instances. This project automates the identification and optimization required for these resources to reduce waste.

Relevance to Cloud Computing:

Cloud Infrastructure: Efficient use of cloud resources directly relates to managing cloud costs and sustainability.  
Hosting and Development: Optimized infrastructure provides hosting and development of an application with reduced latency and eliminates unnecessary overhead.

Proposed Solution:  
  
Description: The script will be an automation in Python that provides a resource utilization analysis of the cloud environment, finds idle or underutilized resources, and provides actionable recommendations on their deletion or downsizing.

Approach:  
Use the SDKs in the cloud: boto3 for AWS, google-cloud for GCP, or azure-mgmt-resource for Azure.  
Gather resource metrics and deliver automated analysis.  
Generate reports with optimization recommendations.

Why This Project is Valuable?  
  
Practical: Reduces operational costs and increases efficiency in the use of resources.  
Educational: Develop skills in Python programming, SDK use, and cloud cost controlling.  
Practical Application: Prepares students for the challenges associated with cloud-based enterprise.

Libraries and Tools:  
  
SDKs: boto3, google-cloud, azure-mgmt-resource  
Additional Libraries: pandas for data processing, matplotlib for visualization.

Complexities and Challenges:  
  
Challenges:  
Handling API rate limits.  
Ensuring interoperability between different cloud providers.  
Estimated Complexity: Moderate to complete high, depending on multi-cloud support.

Part 2: Solve the Problem Using Python (Python code files; Week 12/13) Steps to Implement the Solution  
  
Code Structure

Module 1: Authenticate and setup SDK for cloud providers.

Module 2 - Read resource data in and calculate some utilization metrics.

Module 3: Create a summary report with optimization recommendations.

Script:  
import boto3

from botocore.exceptions import NoCredentialsError, PartialCredentialsError

def fetch\_ec2\_instances():

"""

Fetch all EC2 instances in the AWS account and region.

:return: List of EC2 instances with ID, state, and launch time.

"""

try:

ec2 = boto3.client('ec2') # Initialize EC2 client

response = ec2.describe\_instances() # Fetch instance details

instances = []

for reservation in response.get('Reservations', []):

for instance in reservation.get('Instances', []):

instances.append({

'InstanceId': instance['InstanceId'],

'State': instance['State']['Name'],

'LaunchTime': instance['LaunchTime']

})

return instances

except NoCredentialsError:

print("Error: AWS credentials not found. Please configure your credentials.")

return []

except PartialCredentialsError:

print("Error: Incomplete AWS credentials. Ensure both access key and secret key are set.")

return []

except Exception as e:

print(f"An unexpected error occurred: {e}")

return []

def identify\_idle\_instances(instances):

"""

Identify idle EC2 instances (stopped state).

:param instances: List of EC2 instance data.

:return: List of idle instances.

"""

idle\_instances = [inst for inst in instances if inst['State'].lower() == 'stopped']

return idle\_instances

if \_\_name\_\_ == "\_\_main\_\_":

print("Fetching EC2 instances...")

ec2\_instances = fetch\_ec2\_instances()

if not ec2\_instances:

print("No EC2 instances found or an error occurred.")

else:

print("Identifying idle instances...")

idle\_instances = identify\_idle\_instances(ec2\_instances)

if idle\_instances:

print(f"Found {len(idle\_instances)} idle instance(s):")

for instance in idle\_instances:

print(f"- Instance ID: {instance['InstanceId']}, Launch Time: {instance['LaunchTime']}")

else:

print("No idle instances found.")

Conclusion:

One such success in the automation of the process to optimize cloud resources for cost reduction underlines one important problem that organizations face when trying to manage their unused or underutilized resources for cost-cutting: a well-structured solution by using Python and cloud SDKs that automate the resource analysis and give recommendations about optimization.

Below is a Python script proposing a very practical functionality of your solution, fetching data for EC2 instances and deducing which of those AWS resources are idle. Again, this would be relevant for expanding the solution to encompass other cloud vendors such as GCP and Azure: Overcoming some of them, like API rate limits to interoperability between APIs with multi-cloud environments, illustrates the complexity involved in what one is trying to accomplish in this respect.

This project enhances skills in practical cloud management and provides one of the most efficient scalable manners toward cost optimization. Such insights and tools developed can be very valuable in many educational and enterprise applications due to their ability to contribute to better preparation for everyday cloud computing challenges.  
  
References:

<https://aws.amazon.com/aws-cost-management/cost-optimization/>

<https://azure.microsoft.com/en-us/products/cost-management/>

<https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

<https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/>