**ARCHITECTURE**

**A diagram of a program

AI-generated content may be incorrect.**

**DETAILED WORKING**

1. AWS CloudWatch event rule triggers the event rule (daily/weekly/monthly) invoking lambda function
2. Lambda function runs the python script.
3. Python script sets up AWS clients to connect with cost explorer, s3 bucket. Looks for the environment variable CUR\_BUCKET i.e. s3 bucket name and CUR\_RANGE whether you want to pull cost and usage for a day, week or a month. Based on CUR\_RANGE (daily/weekly/monthly), it pulls data from cost explorer and writes the data in csv and save it in S3 bucket.
4. For example:
   1. **Daily**: if you run the lambda on 21st March 2025, the csv will contain cost data of 19th March 2025
   2. **Weekly**: if you run the lambda on 21st March 2025, the csv will contain cost data b/w 13th March 2025 and 19th March 2025
   3. **Monthly**: if you run the lambda on 2st March 2025, the csv will contain cost data b/w 1th Feb 2025 and 28th feb 2025
5. Once lambda is triggered, IAM role attached to lambda provide access to read the Cost explorer api and upload to s3 bucket.
6. Logs of lambda are stored in Cloud Watch logs for visibility.
7. Budget alerts is independent resource which will trigger alerts whenever the cost reaches the threshold of 80% of limit cost.

**REASON FOR CHOOSING ABOVE ARCHITECTURE**

1. While creating the above architecture the thinking was to create a system which is flexible and cost effective. At the same time, report should give complete cost for a day or week or a month. Pulling real time data every minute won’t work as cost explorer doesn’t update cost in real time and it takes upto 24 hours to reflect the total cost of a day, so running real time numbers won’t be giving us correct numbers and will cost more (multiple s3 file upload, multiple lambdas runs, multiple lambda invocation).
2. Selecting services native to AWS. So that the integration is smooth.
3. Reason for choosing lambda is that it is best suitable for lightweight tasks/operations which are executed on schedule or on events.
4. As lambda is serverless, it follows pay for what you usemodel thus it is cost efficient.
5. Regarding CloudWatch logs, they are needed for visibility for lambda runs but to make it cost less, we have added a retention of 14 days.
6. Regarding S3 bucket, it has a limited bucket policy, making it restricted within the account and has an INTELLIGENT\_TIERING lifecycle configuration which will save cost automatically and will have same low latency.

**COST ESTIMATION OF ABOVE SETUP**

Below cost estimation is based on AWS Pricing Calculator. <https://calculator.aws/#/> and assuming that **ONLY** above setup runs in the AWS account.

**Lambda**: Assuming it runs daily on average of 40 secs with 128mb. Calculating monthly cost.

A screenshot of a calculator

AI-generated content may be incorrect.

Lambda cost us $0 as it offers free 1 million request per month and 400000 GB-s computes.

**S3**: The size of file can vary based on number of rows in CSV. It could be in KB, MB or in GB. Assuming it is in MB. S3 charges $0.023 per GB for first 50TB/ month. Also, S3 offer 2000 free PUT requests, so no cost for those if we are just sending it once a day i.e. total 30 PUT request.

A screenshot of a computer

AI-generated content may be incorrect.

**Cost Explorer:** Cost Explorer API is free for monthly and daily granularity.

**EventBridge**: You can make 14,000,000 invocations per month for free. Then $1.00/million scheduled invocations per month. (For above scenario, free tier will suffice)