SIR Epidemic Spread Simulator (SIR-Sim)

Michael Vilsoet
Jonathan Reider
Benjamin Huang

1. Introduction

Our project implements the SIR mathematical framework for epidemic simulation. The SIR model is used in epidemiology to describe the spread of infectious diseases within a population. It divides the population into three compartments: **Susceptible (S)**, representing individuals who are vulnerable to infection; **Infected (I)**, those who are currently carrying and transmitting the disease; and **Recovered (R)**, individuals who have either recovered from the disease and gained immunity or are no longer infectious. The model uses differential equations to track transitions between these compartments over time, based on parameters such as the infection rate and recovery rate. Since we run on a timestep (discrete, not continuous) basis, we simplify these differential equations into numerical increases or decreases. By capturing the dynamics of disease transmission, the SIR model provides insights into epidemic patterns, peak infection periods, and the effectiveness of interventions.

SIR Epidemic Spread Simulator (**SIR-Sim**) is a cloud-based implementation of the aforementioned SIR model for simulating epidemic spread. The system is designed to process both regular user requests and handle sudden spikes in traffic efficiently. Features are:

- Simulation based on user-defined epidemic parameters.
- Frontend to send and view simulation results.
- Distributed architecture using AWS services and handling of sudden traffic spikes.
- Results are saved and retrievable at any time for analysis.

1.1 User Interaction and Web App

Users interact with the simulator by filling out a form with the following inputs: **Population Size** (an integer between 100 and 1000), **Initial Infection Rate** (a decimal between 0 and 1, representing the proportion of the population initially infected), **Initial Number of Infected Individuals** (an integer value), and **Recovery Rate** (a decimal between 0 and 1), representing the rate of recovery from the disease). After completing the form, users click the **"START"** button to initiate the simulation. After submitting the form, our system kicks off and results are shown within a few seconds.

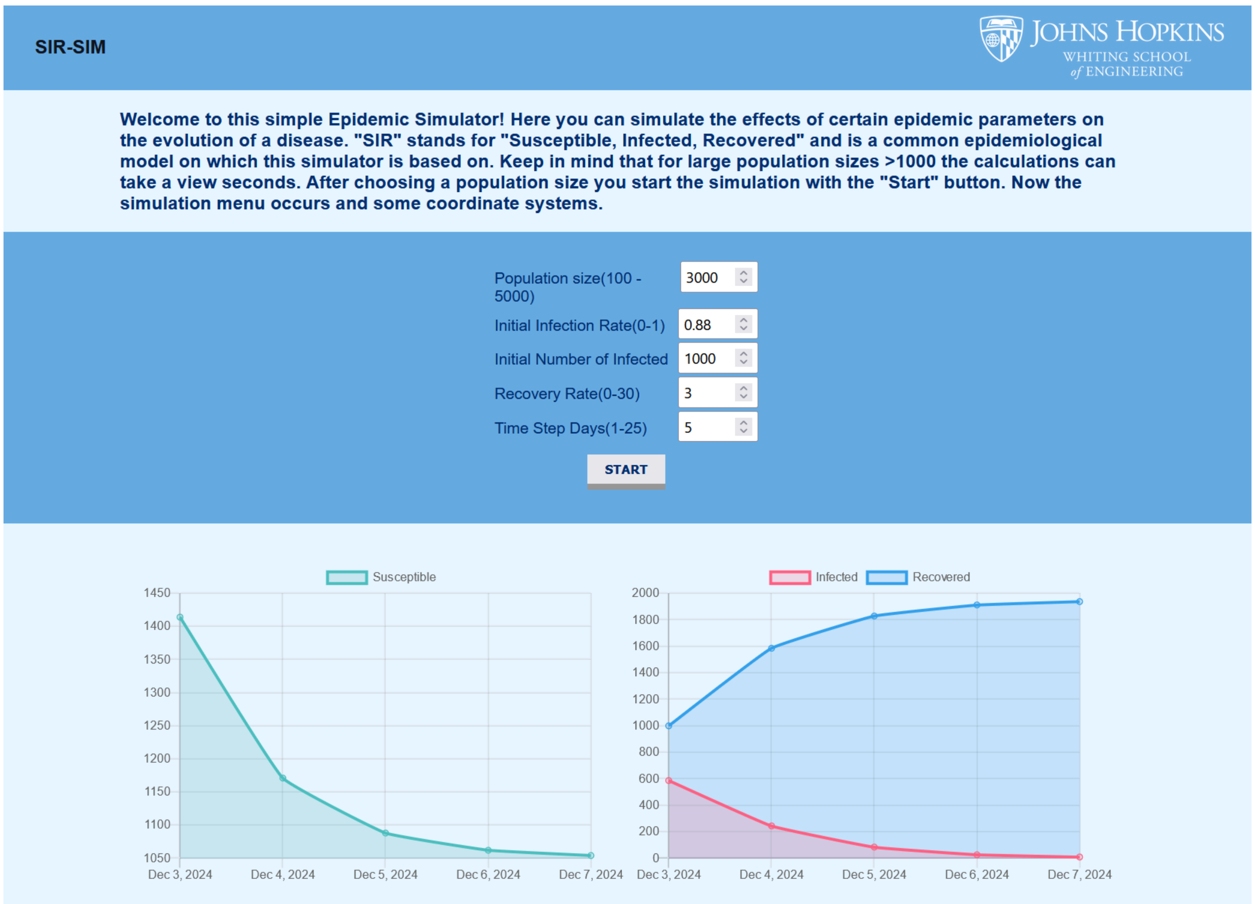

Figure 1-1: Screenshot of SIR-SIM Frontend

2. Architecture Overview

2.1 Cloud Diagram

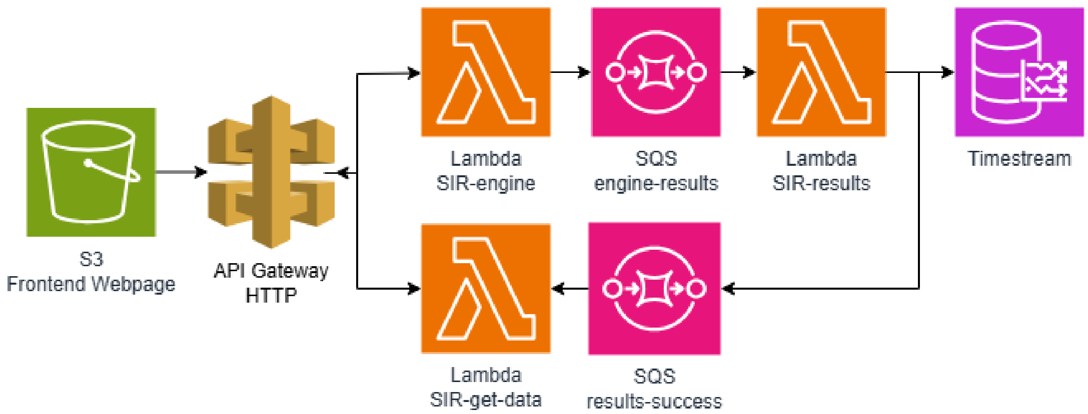

*Figure 2-1: One or many items in the diagram are contained within our four components. S3 HTML/CSS/Javascript Webpage makes the **Frontend**. HTTP Gateway, Lambda SIR-get-data makes up the **API Gateway & Request Handler.** Lambda SIR-results is the **Simulation Engine Service**. Finally, Lambda SIR-results, Timestream DB are the **Results Management Service (DB).***

2.2 Cloud Security

In this system, each component is granted AWS IAM permissions exhibiting the Principle of Least Privilege (**PoLP**). The Frontend (Javascript/S3) is given read-only access to S3 and API Gateway, preventing direct manipulation of backend resources. The API Gateway & Request Handler is assigned limited SQS queue access to send messages to the Simulation Engine Service, with strict input validation to prevent unauthorized or malicious requests. The Simulation Engine Service (AWS Lambda) receives read-only permissions for SQS queue consumption and limited write access to Timestream for result storage, ensuring it cannot modify existing records or access unrelated database tables. The Results Management Service is granted precise Timestream read and write permissions scoped only to the specific database and table for the simulation results, with no ability to create or delete database resources. Each component's IAM role is tightly constrained, using AWS's granular policy controls to restrict actions to only those absolutely necessary for its specific function, thereby minimizing potential attack surfaces and reducing the risk of unauthorized access or system compromise.

2.3 System Components

The system allows end-users to simulate epidemics based on the SIR model by inputting custom parameters. We describe the following four modules used to complete our business process:

1. **Frontend**

- **Technology:** Javascript, AWS S3 Bucket
- **Purpose:** Create a basic entrypoint for end users to interact with the application.
- **Inputs:** User inputs on predefined and input-validated web forms.
- **Outputs:** Requests are sent to the **API Gateway & Request Handler** component.

2. **API Gateway & Request Handler**
    - **Technology:** AWS API HTTP Gateway, AWS SQS
    - **Purpose:** Acts as the interface between the user and the simulation system, handling both requests to start simulations and retrieval of stored results.
    - **Inputs:** User-defined epidemic parameters:
        - *Population Size*: Total number of individuals in the simulation.
        - *Infection Rate*: Proportion of susceptible individuals infected per time step.
        - *Recovery Rate:* Rate of recovery for infected individuals.
        - *Time Steps:* Number of discrete time intervals for the simulation.
        - *Initial Infected Count:* Number of individuals infected at the start of the simulation.
    - **Outputs:**
        - Forwards validated simulation requests to the **Simulation Engine Service** via SQS.
        - Processes database queries to retrieve previously stored results.
3. **Simulation Engine Service**
    - **Technology:** Python with Cython (optimized for performance) on AWS Lambda
    - **Purpose:** Performs computationally intensive SIR model calculations for user-defined epidemic parameters.
    - **Inputs:** Formatted requests received from the API Gateway, including:
        - Population size, infection rate, recovery rate, time steps, and initial infected count.
    - **Outputs:** A list of time-series results for the SIR model:
        - Timestamp
        - S (Susceptible)
        - I (Infected)
        - R (Recovered)
4. **Results Management Service (DB)**
    - **Technology:** Python with **AWS Timestream**.
    - **Purpose:** Store, retrieve, and manage simulation results using AWS Timestream database
    - **Input 1:** Simulation engine outputs (see appendix for example).
    - **Input 2:** Retrieval requests from API Gateway
    - **Outputs:**
        - Successful write acknowledgement
        - Queried simulation results

3. Design Decisions

AWS Lambda (FaaS) and serverless services were utilized to eliminate cost of resources when nothing was running. Using this we can use event-driven architecture (EDA) to only run when a request is made. Each component was given permissions only for its intended purpose, following the principle of least privilege with IAM roles.

We went with containerized environments to control the environment of our applications. Doing this we can use the optimal language for any micro-service and it would not affect others. For instance, we wanted to use Java Spring Boot for the results manager and it would have been possible, however, learning from the previous project it was discovered that Spring Boot is not a cost efficient framework for Serverless applications. Therefore, early on this was migrated to using Python because it has the best compatibility with Lambda.

For the user interface, it is expensive to utilize PaaS such as AWS Beanstalk or even to use EC2 instances. Therefore as a cost efficient solution, we utilized S3's static webpage hosting ability to run a simple HTML webpage.

For a database solution we decided to go with AWS Timestream. This provides a serverless database solution that is optimized for storing timed events. From early testing we found that the cost of small requests that we defined were minimal. After being burned by timestamp formatting, we decided that using epoch time (int) in UTC was easiest. One of the first tests that was run to determine the cost of a burst of a thousand data writes to the database over an hour and the results of the test was that it cost about one cent over this time period that made it an obvious choice for the database.

4. Performance Analysis

4.1 Performance Metrics

Module	Performance (*ms*, average)
sir-engine	96
results-management	2,690
get-data	1,311

Sub-5-Second Response time achieved through stateless architecture utilizing SQS ensuring modularity and parallel processing. AWS Lambda provides rapid, scalable simulation execution. Finally, simplified constant-time algorithms and compiled (Cython) *for* loops are used to speed up calculation. The Lambdas take an average of 4,097 ms in total to return results to the user.

4.2 Traffic Management

- **Normal Load:** 24 requests/day (1 per hour).
- **Spike Handling:** Below the Cloudwatch metrics show the application handling 1000 requests in an hour (This was simulated using a script that sends a request via the frontend every 1.5 seconds

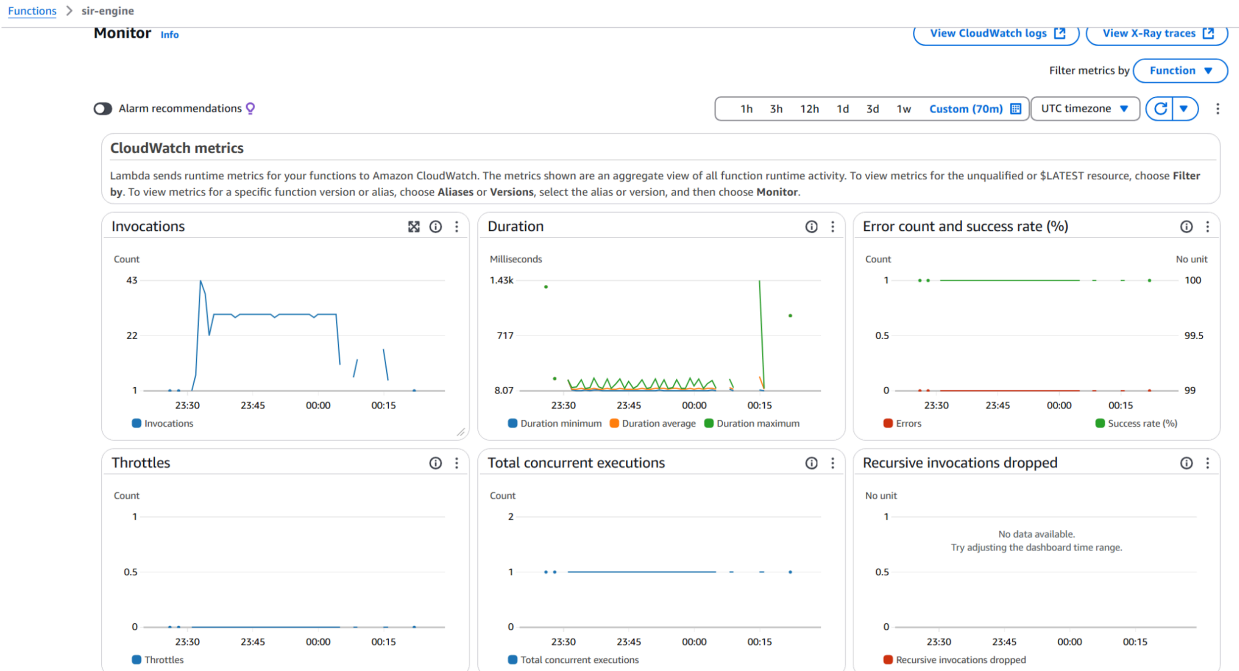

Figure 4-1: sir-engine AWS Lambda CloudWatch Metrics during the 1000 requests in an hour test

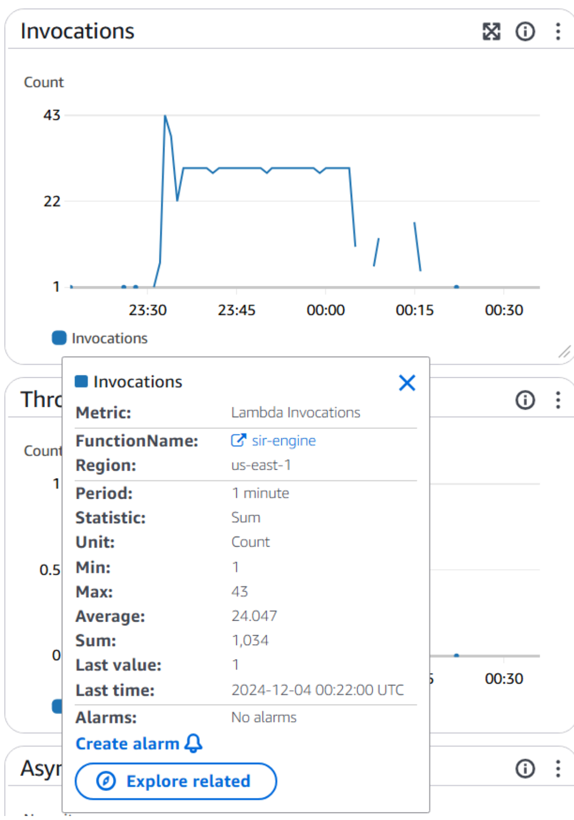

Figure 4-2: sir-engine AWS Lambda CloudWatch Metrics showing 1,034 invocations within 1 hr

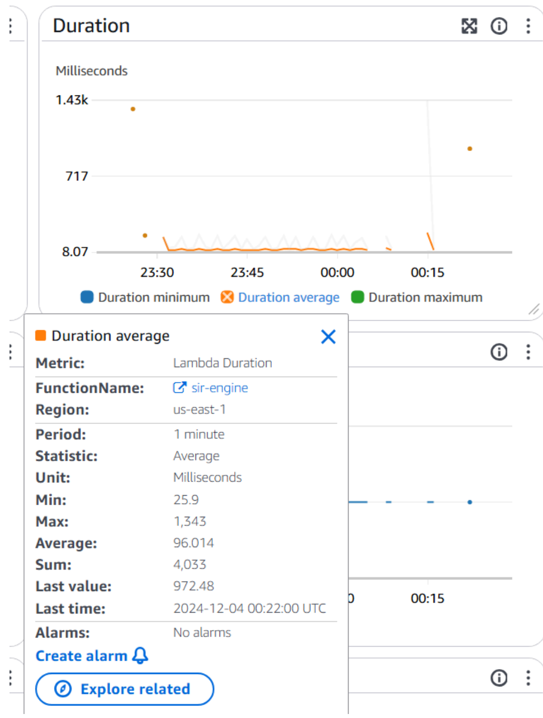

Figure 4-3: sir-engine AWS Lambda CloudWatch Metrics showing avg 96 ms runtime

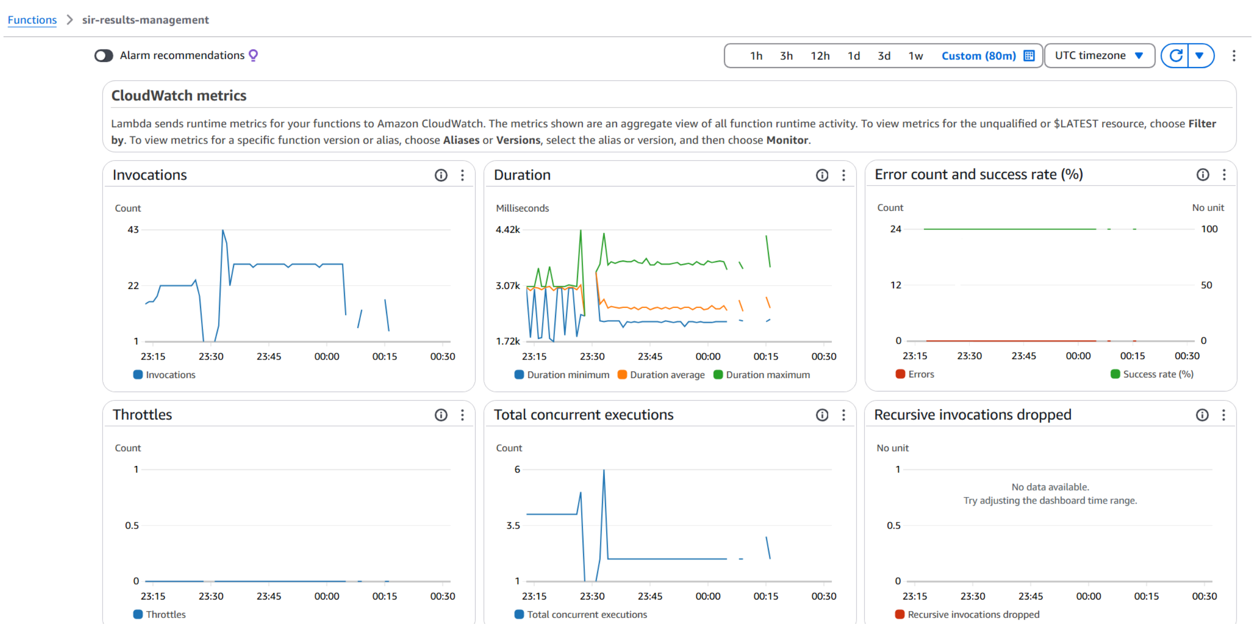

Figure 4-4: sir-results-management AWS Lambda CloudWatch Metrics during the 1000 requests in an hour test

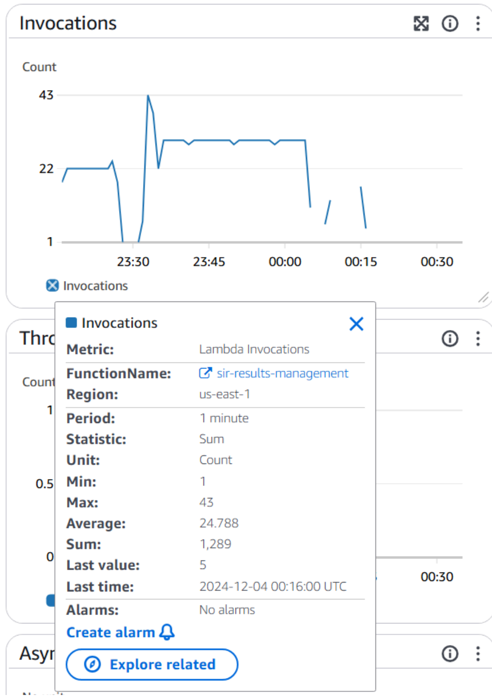

Figure 4-5: sir-results-management AWS Lambda CloudWatch Metrics showing 1,289 invocations within 1 hr

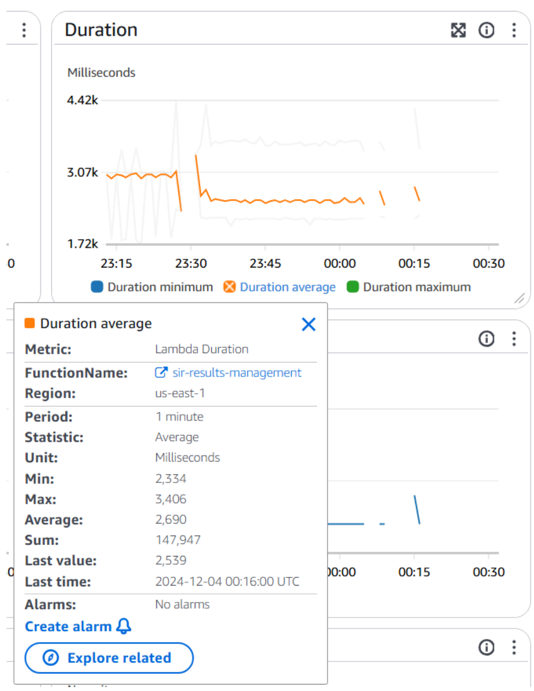

Figure 4-6: sir-results-management AWS Lambda CloudWatch Metrics showing avg 2,690 ms runtime

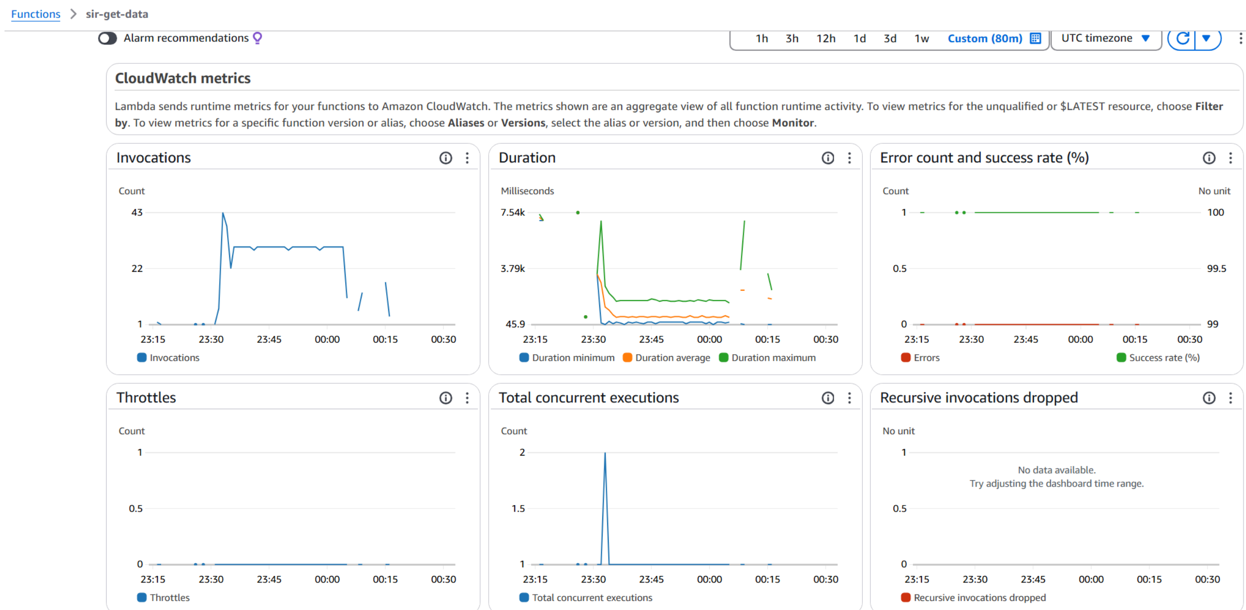

Figure 4-7: sir-get-data AWS Lambda CloudWatch Metrics during the 1000 requests in an hour test

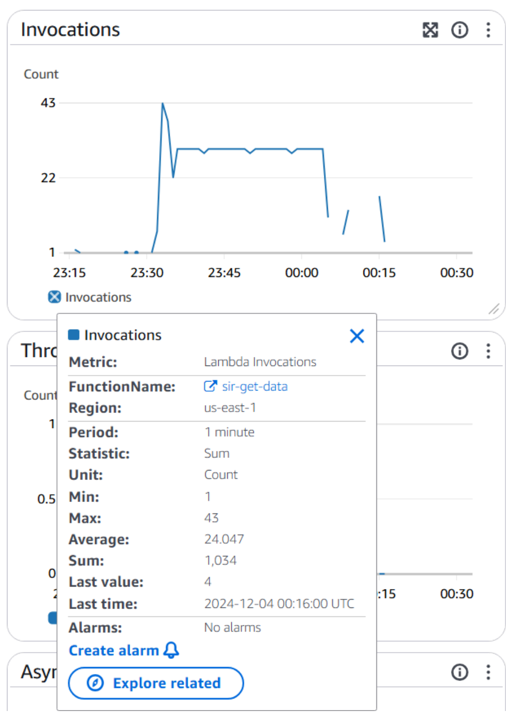

Figure 4-8: sir-get-data AWS Lambda CloudWatch Metrics showing 1,034 invocations within 1 hr

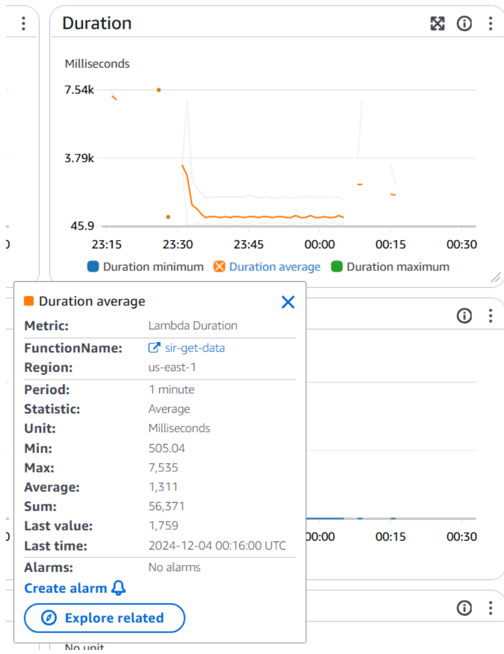

Figure 4-9: sir-get-data AWS Lambda CloudWatch Metrics showing avg 1,311 ms runtime

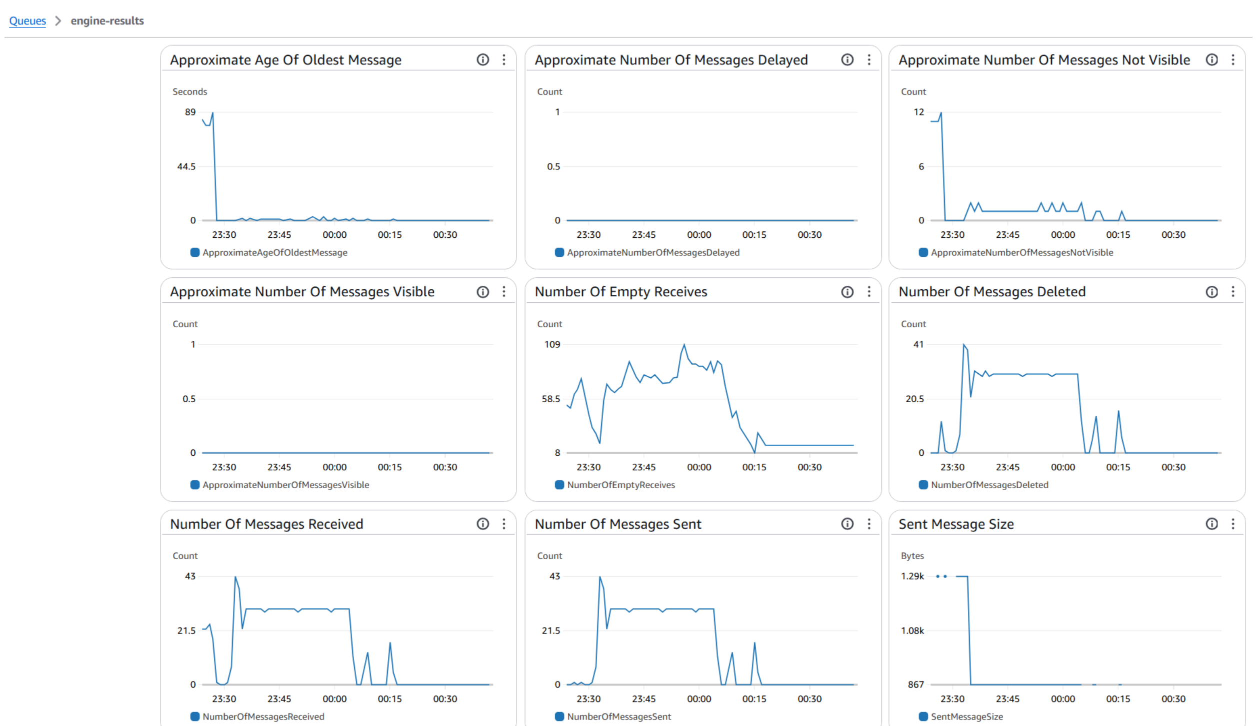

Figure 4-10: engine-results AWS SQS CloudWatch Metrics during the 1000 requests in an hour test

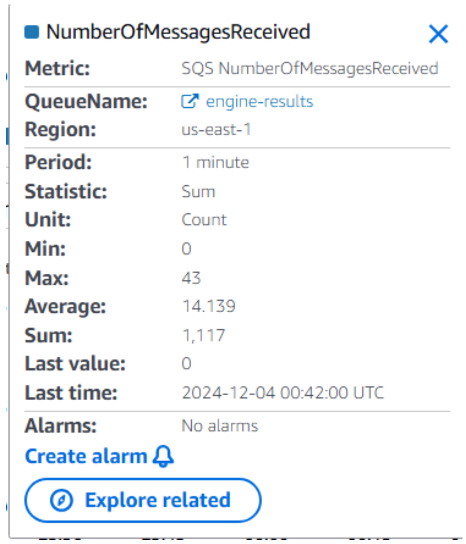

Figure 4-11: engine-results AWS SQS CloudWatch Metrics showing 1,117 total messages received

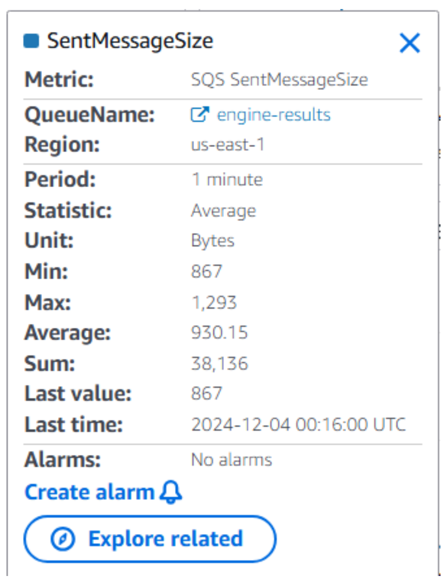

Figure 4-12: engine-results AWS SQS CloudWatch Metrics showing avg size 930 bytes/msg

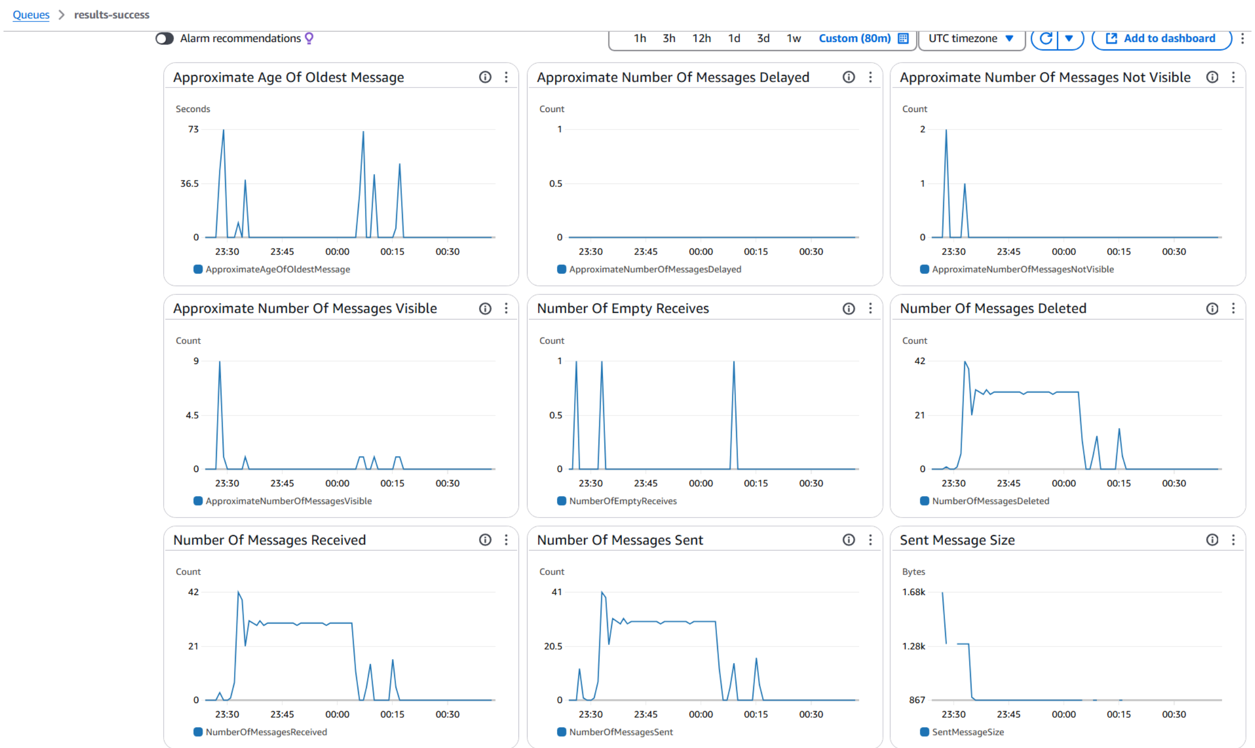

Figure 4-13: results-success AWS SQS CloudWatch Metrics during the 1000 requests in an hour test

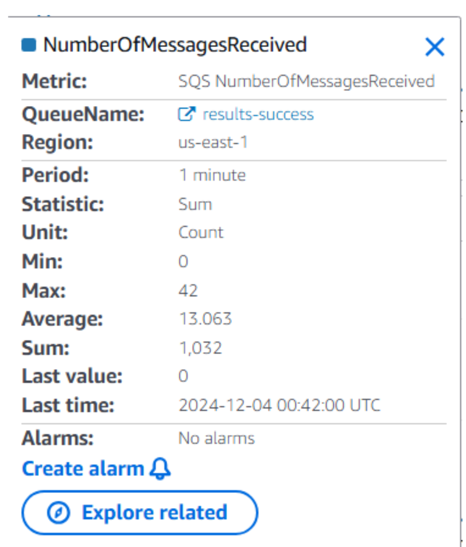

Figure 4-14: results-success AWS SQS CloudWatch Metrics showing 1,032 total messages received

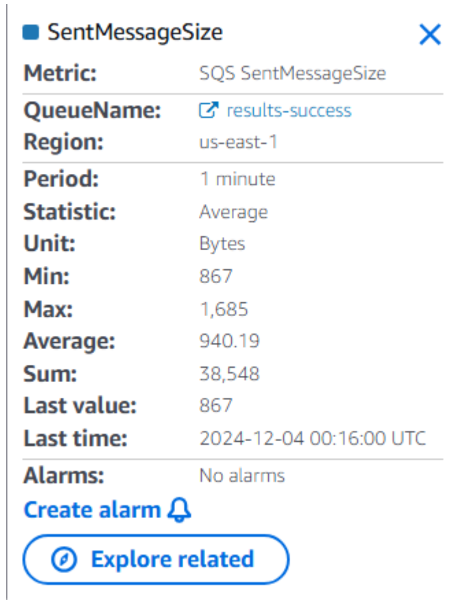

Figure 4-15: results-success AWS SQS CloudWatch Metrics showing avg size 940 bytes/msg

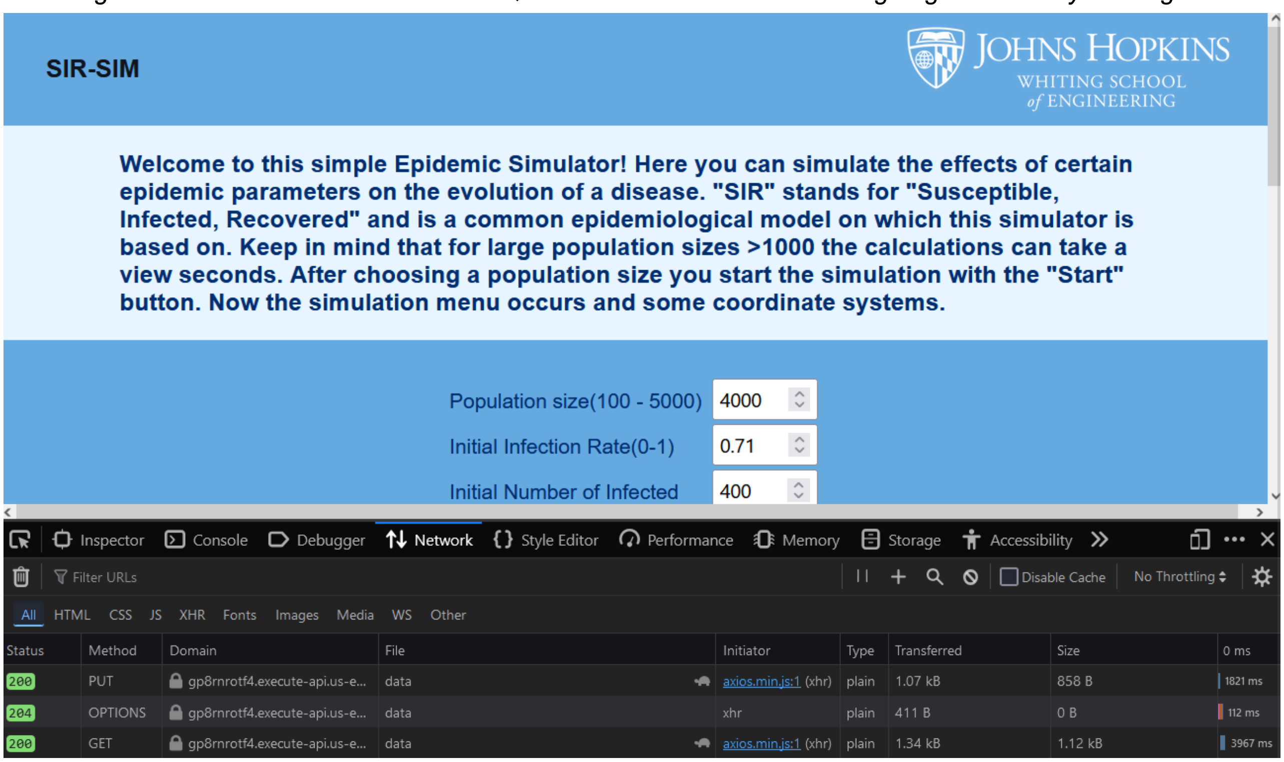

Figure 4-16: Browser Network tool showing 1.07 KB sent for PUT request and 1.34 KB received for GET request

5. Cost Analysis

5.1 Cost Estimation (1024 simulation requests per day)

Cost Estimation using the AWS Calculator: $18.60 per year or $0.62 per month
https://calculator.aws/#/estimate?id=0c3ea387af585647468a9c3f37633b52a3b90bdb

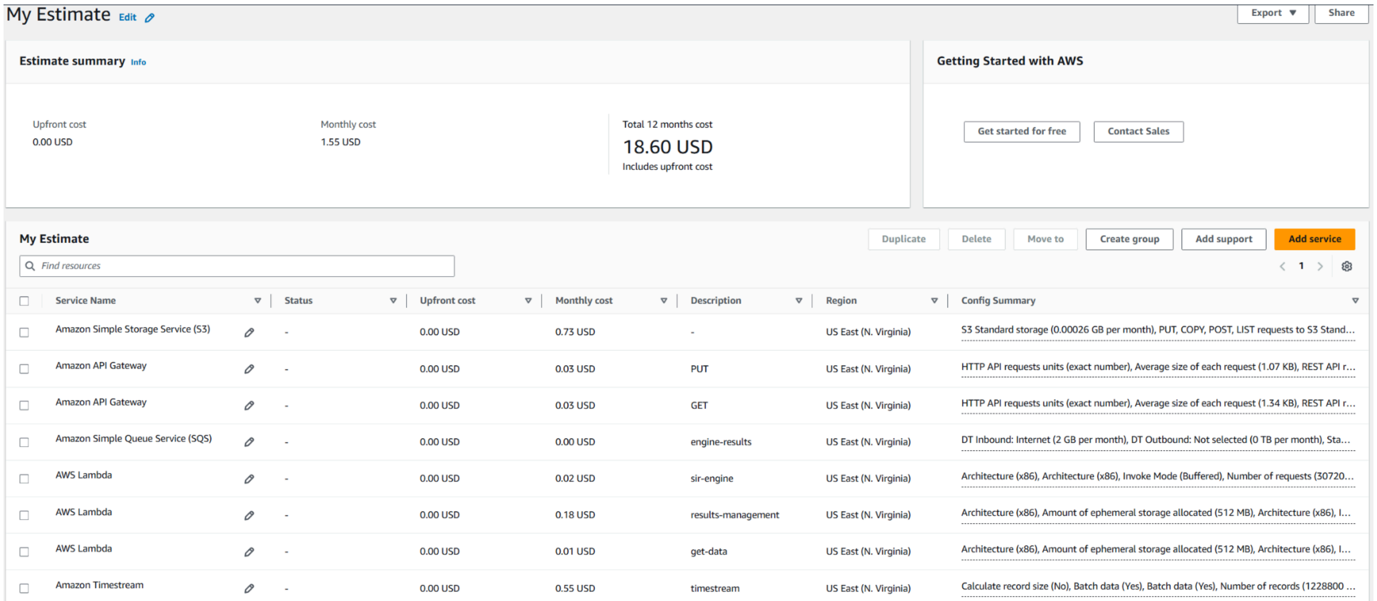

Figure 5-1: Screenshot of the estimate summary

Below is a table with the estimated detailed breakdown:

Services	Usage	Total cost per month
S3	0.00026 Gb total storage to store frontend	Tiered price for: 0.00026 GB 0.00026 GB x 0.023 USD = 0.00 USD Total tier cost = 0.00 USD (S3 Standard storage cost) 30,720 GET requests in a month x 0.0000004 USD per request = 0.0123 USD (S3 Standard GET requests cost) **S3 Standard cost (monthly): 0.01 USD**
S3 data transfer	0.00026Gb*1024 request a day*30 days = ~8Gb a month accessed	Inbound: All other regions: 0 GB x 0 USD per GB = 0.00 USD Outbound: Internet: 1 GB x 0.09 USD per GB = 0.09 USD **Data Transfer cost (monthly): 0.09 USD**

2 SQS	- 30,720 request/mo each SQS - 930 bytes/msg (<1Gb transferred)	1 requests per month x 1000000 multiplier for million = 1,000,000.00 total standard queue requests Tiered price for: 1,000,000.00 requests 1,000,000 requests x 0.00 USD = 0.00 USD **Total tier cost = 0.00 USD (Standard queue requests cost)** **Total SQS cost (monthly): 0.00 USD**
Lambda (sir-engine)	- 30,720/mo - Avg 96ms runtime	30,720 requests x 86 ms x 0.001 ms to sec conversion factor = 2,641.92 total compute (seconds) 0.125 GB x 2,641.92 seconds = 330.24 total compute (GB-s) Tiered price for: 330.24 GB-s 330.24 GB-s x 0.0000166667 USD = 0.01 USD Total tier cost = 0.0055 USD (monthly compute charges) **Monthly compute charges: 0.01 USD** 30,720 requests x 0.0000002 USD = 0.01 USD (monthly request charges) **Monthly request charges: 0.01 USD** 0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function **Monthly ephemeral storage charges: 0 USD** 0.01 USD + 0.01 USD = 0.02 USD **Lambda cost (monthly): 0.02 USD**
Lambda (results-management)	- 30,720/mo - Avg 2,690ms	30,720 requests x 2,690 ms x 0.001 ms to sec conversion factor = 82,636.80 total compute (seconds) 0.125 GB x 82,636.80 seconds = 10,329.60 total compute (GB-s) Tiered price for: 10,329.60 GB-s 10,329.60 GB-s x 0.0000166667 USD = 0.17 USD Total tier cost = 0.1722 USD (monthly compute charges) **Monthly compute charges: 0.17 USD** 30,720 requests x 0.0000002 USD = 0.01 USD (monthly request charges) **Monthly request charges: 0.01 USD** 0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function **Monthly ephemeral storage charges: 0 USD** 0.17 USD + 0.01 USD = 0.18 USD **Lambda cost (monthly): 0.18 USD**
Lambda (get-data)	- 30,720/mo - Avg 1,311ms	30,720 requests x 1,311 ms x 0.001 ms to sec conversion factor = 40,273.92 total compute (seconds) **Monthly compute charges: 0.00 USD** 30,720 requests x 0.0000002 USD = 0.01 USD (monthly request charges) **Monthly request charges: 0.01 USD** 0.50 GB - 0.5 GB (no additional charge) = 0.00 GB billable ephemeral storage per function **Monthly ephemeral storage charges: 0 USD** **Lambda cost (monthly): 0.01 USD**
Timestream DB	- 30,720 writes/mo - 900 bytes per record	0.87890625 KB / 1048576 KB per GB = 0.00000083819031715393 GB 1,228,800 records per months / 730 hours per month = 1,683.2876712328766 records per hours RoundUp (0.87890625) = 1 KB 1 - 0 = 1.00 not repeated data per batch 100 records - 1 = 99.00 records

		99.00 records x 1.00 not repeated data per batch x 0.87890625 KB = 87.01 KB 87.01 KB + 0.87890625 KB = 87.89 KB per batch RoundUp (87.89) = 88 KB per batch 1,228,800 / 100 = 12,288.00 writes per month RoundUp (12288.00) = 12288 writes per month 12,288 writes per month x 88 KB per batch x 0.000001 KB in GB = 1.08 GB per month 1.08 GB x 0.50 USD = 0.54 USD **Writes price (monthly): 0.54 USD**
HTTP API Gateway PUT	30,720/month	1.07 KB per request / 512 KB request increment = 0.00208984375 request(s) RoundUp (0.00208984375) = 1 billable request(s) 30,720 requests per month x 1 unit multiplier x 1 billable request(s) = 30,720 total billable request(s) Tiered price for: 30,720 requests 30,720 requests x 0.000001 USD = 0.03 USD Total tier cost = 0.0307 USD (HTTP API requests) **HTTP API request cost (monthly): 0.03 USD**
HTTP API Gateway GET	30,720/month	1.34 KB per request / 512 KB request increment = 0.0026171875 request(s) RoundUp (0.0026171875) = 1 billable request(s) 30,720 requests per month x 1 unit multiplier x 1 billable request(s) = 30,720 total billable request(s) Tiered price for: 30,720 requests 30,720 requests x 0.000001 USD = 0.03 USD Total tier cost = 0.0307 USD (HTTP API requests) **HTTP API request cost (monthly): 0.03 USD**

5.2 Actual Cost (1024 simulation requests per day test)

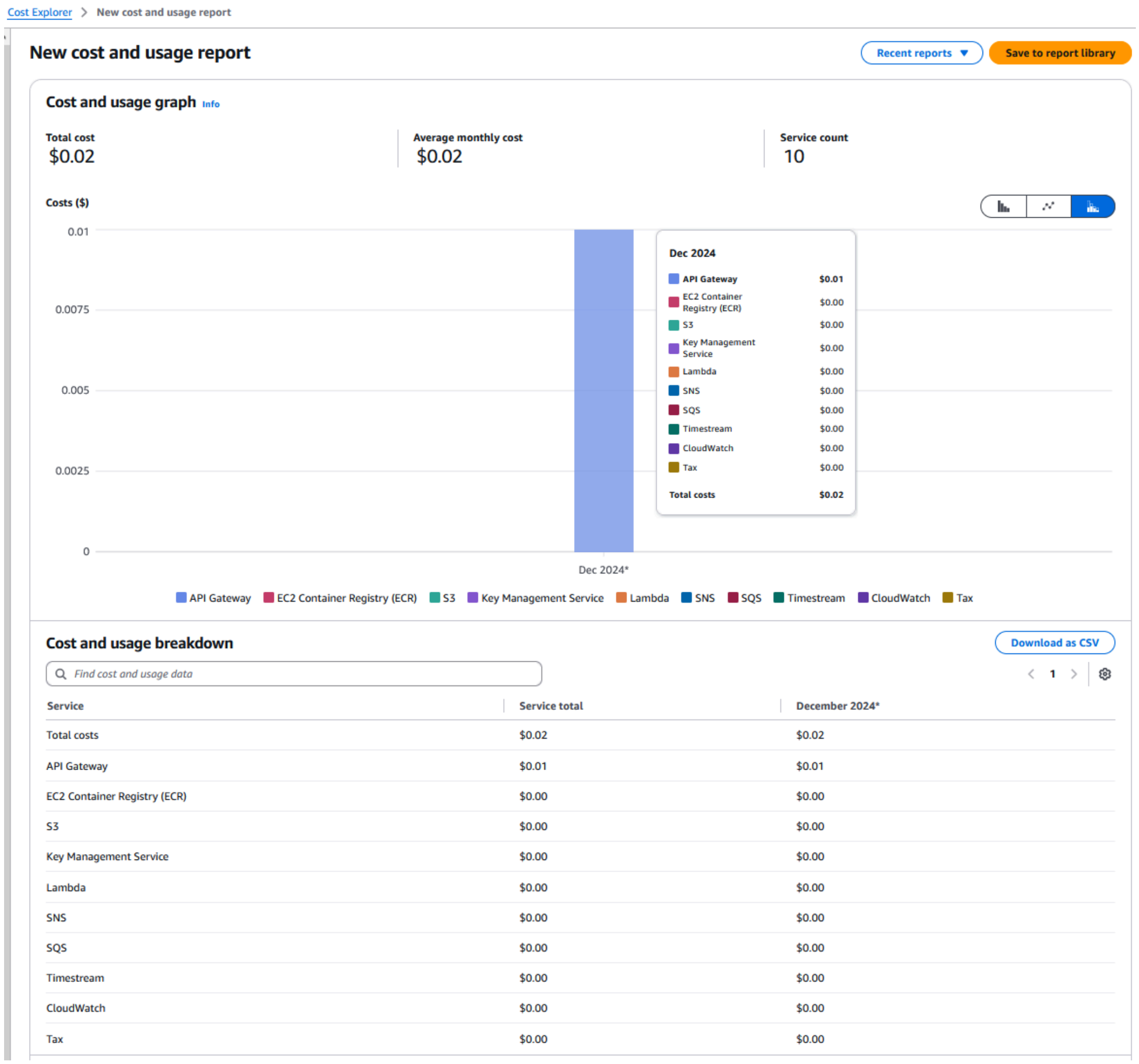

Figure 5-2: Screenshot of the Cost Explorer graph showing $0.02 costs

Free Tier offers in use (11)

Find service name

1 2

Service	AWS Free Tier usage limit	Current usage	Forecasted usage	MTD actual usage %	MTD forecasted usage %
Amazon Simple Queue Service	1000000.0 Requests are always free per month as part of AWS Free Usage Tier (Global-Requests)	70,218 Requests	435,352 Requests	7.02%	43.54%
AWS Lambda	1000000.0 Request are always free per month as part of AWS Free Usage Tier (Global-Request)	15,324 Request	95,009 Request	1.53%	9.50%
AmazonCloudWatch	5.0 GB are always free per month as part of AWS Free Usage Tier (Global-DataProcessing-Bytes)	0 GB	0 GB	0.79%	4.88%
AWS Lambda	400000.0 seconds are always free per month as part of AWS Free Usage Tier (Global-Lambda-GB-Second)	2,088 seconds	12,946 seconds	0.52%	3.24%
Amazon Timestream	50.0 GB for free per month during a short-term trial as part of AWS Free Usage Tier (Global-DataIngestion-Bytes)	0 GB	0 GB	0.01%	0.07%
AWS Key Management Service	20000.0 Requests are always free per month as part of AWS Free Usage Tier (Global-KMS-Requests)	2 Requests	12 Requests	0.01%	0.06%
Amazon Timestream	750.0 GB-Hours for free per month during a short-term trial as part of AWS Free Usage Tier (Global-MemoryStore-ByteHrs)	0 GB-Hours	0 GB-Hours	0.00%	0.01%
Amazon Simple Notification Service	1000000.0 Requests are always free per month as part of AWS Free Usage Tier (Requests-Tier1)	18 Requests	112 Requests	0.00%	0.01%
AmazonCloudWatch	5.0 GB-Mo are always free per month as part of AWS Free Usage Tier (Global-TimedStorage-ByteHrs)	0 GB-Mo	0 GB-Mo	0.00%	0.01%
AmazonCloudWatch	1000000.0 Requests are always free per month as part of AWS Free Usage Tier (Global-CW:Requests)	2 Requests	12 Requests	0.00%	0.00%
Amazon Timestream	100.0 GB-Mo for free per month during a short-term trial as part of AWS Free Usage Tier (Global-MagneticStore-ByteHrs)	0 GB-Mo	0 GB-Mo	0.00%	0.00%

Figure 5-3: Screenshot of the service usage and always free tier usage in December

Description	Usage Quantity	Amount...
API Gateway		**USD 0.01**
US East (N. Virginia)		USD 0.01
Amazon API Gateway ApiGatewayHttpApi		USD 0.01
$1/million requests - API Gateway HTTP API (first 300 million)	14,959 Requests	USD 0.01
CloudWatch		**USD 0.00**
US East (N. Virginia)		USD 0.00
Amazon CloudWatch		USD 0.00
$0.00 per request - first 1,000,000 requests	2 Requests	USD 0.00
AmazonCloudWatch PutLogEvents		USD 0.00
First 5GB per month of log data ingested is free.	0.039 GB	USD 0.00
AmazonCloudWatch USE1-TimedStorage-ByteHrs		USD 0.00
First 5GB-mo per month of logs storage is free.	0 GB-Mo	USD 0.00
Data Transfer		**USD 0.00**
US East (N. Virginia)		USD 0.00
Bandwidth		USD 0.00
$0.000 per GB - data transfer in per month	0 GB	USD 0.00
$0.000 per GB - data transfer out under the monthly global free tier	0.019 GB	USD 0.00
EC2 Container Registry (ECR)		**USD 0.00**
US East (N. Virginia)		USD 0.00
Amazon EC2 Container Registry (ECR) TimedStorage-ByteHrs		USD 0.00
$0.10 per GB-month of data storage	0.026 GB-Mo	USD 0.00
Key Management Service		**USD 0.00**
US East (N. Virginia)		USD 0.00
AWS Key Management Service us-east-1-KMS-Requests		USD 0.00
$0.00 per request - Monthly Global Free Tier for KMS requests	2 Requests	USD 0.00
Lambda		**USD 0.00**
US East (N. Virginia)		USD 0.00
AWS Lambda Lambda-GB-Second		USD 0.00
AWS Lambda - Compute Free Tier - 400,000 GB-Seconds - US East (N	2,088.029 seconds	USD 0.00
AWS Lambda Request		USD 0.00
AWS Lambda - Requests Free Tier - 1,000,000 Requests - US East (No	15,324 Requests	USD 0.00

Simple Notification Service		**USD 0.00**
US East (N. Virginia)		USD 0.00
Amazon Simple Notification Service Requests-Tier1		USD 0.00
First 1,000,000 Amazon SNS API Requests per month are free	18 Requests	USD 0.00
Simple Queue Service		**USD 0.00**
US East (N. Virginia)		USD 0.00
Amazon Simple Queue Service Requests-RBP		USD 0.00
First 1,000,000 Amazon SQS Requests per month are free	70,218 Requests	USD 0.00
Simple Storage Service		**USD 0.00**
Any		USD 0.00
Amazon Simple Storage Service GeneralPurposeBuckets		USD 0.00
$0.00 per bucket / month for the first 2000 general purpose buckets	0.003 Bucket-Mo	USD 0.00
US East (N. Virginia)		USD 0.00
Amazon Simple Storage Service Requests-Tier1		USD 0.00
$0.005 per 1,000 PUT, COPY, POST, or LIST requests	45 Requests	USD 0.00
Amazon Simple Storage Service Requests-Tier2		USD 0.00
$0.004 per 10,000 GET and all other requests	1,240 Requests	USD 0.00
Amazon Simple Storage Service TimedStorage-ByteHrs		USD 0.00
$0.023 per GB - first 50 TB / month of storage used	0 GB-Mo	USD 0.00
Timestream		**USD 0.00**
US East (N. Virginia)		USD 0.00
Amazon Timestream USE1-DataIngestion-Bytes		USD 0.00
USD0 price per GB ingested in Timestream - US East (N. Virginia)	0.006 GB	USD 0.00
Amazon Timestream USE1-MagneticStore-ByteHrs		USD 0.00
USD0 price per GB-Month of magnetic storage in Timestream - US Ea	0 GB-Mo	USD 0.00
Amazon Timestream USE1-MemoryStore-ByteHrs		USD 0.00
USD0 price per GB-Hour of memory storage in Timestream - US East	0.015 GB-Hours	USD 0.00

Figure 5-4: Screenshot of the Billing Breakdown and usage

Appendix

A.1 Website URL

- http://sir-sim.com.s3-website-us-east-1.amazonaws.com/

A.2 Source Code Repository

- https://github.com/benjaminhuang13/SIR-Sim/tree/main

A.3 Example: Sir-Engine Output to SQS


```
{
 "Records": [
   {
     "messageId": "19dd0b57-b21e-4ac1-bd88-01bbb068cb78",
     "receiptHandle": "MessageReceiptHandle",
     "body": "{\"userInputs\": {\"populationSize\": 10000, \"infectionRate\": 0.3, \"numInfected\": 100,
\"recoveryRate\": 0.1, \"timeStepsDays\": 20}}",
     "attributes": {
       "ApproximateReceiveCount": "1",
       "SentTimestamp": "1523232000000",
       "SenderId": "123456789012",
       "ApproximateFirstReceiveTimestamp": "1523232000001"
     },
     "messageAttributes": {},
     "md5OfBody": "7b270e59b47ff90a553787216d55d91d",
     "eventSource": "aws:sqs",
     "eventSourceARN": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
     "awsRegion": "us-east-2"
   }
 ]
}
```


A.4 Example: Simulation Request from User (as json)


```
{
   "userInputs": {
       "populationSize": 1000,
       "infectionRate": 0.05,
       "numInfected": 10,
       "recoveryRate": 0.01,
       "timeStepsDays": 20
   }
}
```

A.5 Example: TimescaleDB data