CLOUD COMPUTING PROJECT

Topic: Gray scaling of image with Lambda.

Presented by:
YASWANTH RAHUL YARLAGADDA
JYOTIR VINAY NARAM
VENKATA TEJESWAR SAVITRI

Contents

Project Aim and Requirements

Project Aim: This project demonstrates efficient grayscale image processing using AWS Lambda, specifically focusing on the conversion of images uploaded to Amazon S3. The goal is to showcase the scalability and cost-effectiveness of serverless computing for image manipulation tasks.

Primary Objective: The primary objective is to create an AWS Lambda function that automatically converts images uploaded to a designated Amazon S3 bucket into grayscale format. The converted grayscale images will then be stored in a separate S3 bucket. We will be processing JPEG and PNG images, up to a maximum size of 5MB.

Key Requirements:

- **AWS Lambda:** Provides the serverless compute environment to execute the image processing code. This allows for automatic scaling based on demand.
- **Amazon S3:** Serves as the storage for both the original images (input bucket) and the processed grayscale images (output bucket). S3 provides high availability and durability for our image data.
- **CloudWatch:** Enables monitoring of Lambda function executions, including error rates, invocations, and duration. This is crucial for identifying and resolving issues in the image processing pipeline.
- **IAM Role:** Grants the Lambda function the necessary permissions to access S3 buckets and CloudWatch. This ensures secure access control and prevents unauthorized actions.

Architecture Overview

1

Source S3 Bucket

Holds the original images uploaded by users

2

AWS Lambda Function

Processes images and converts them to grayscale format

3

Destination S3 Bucket

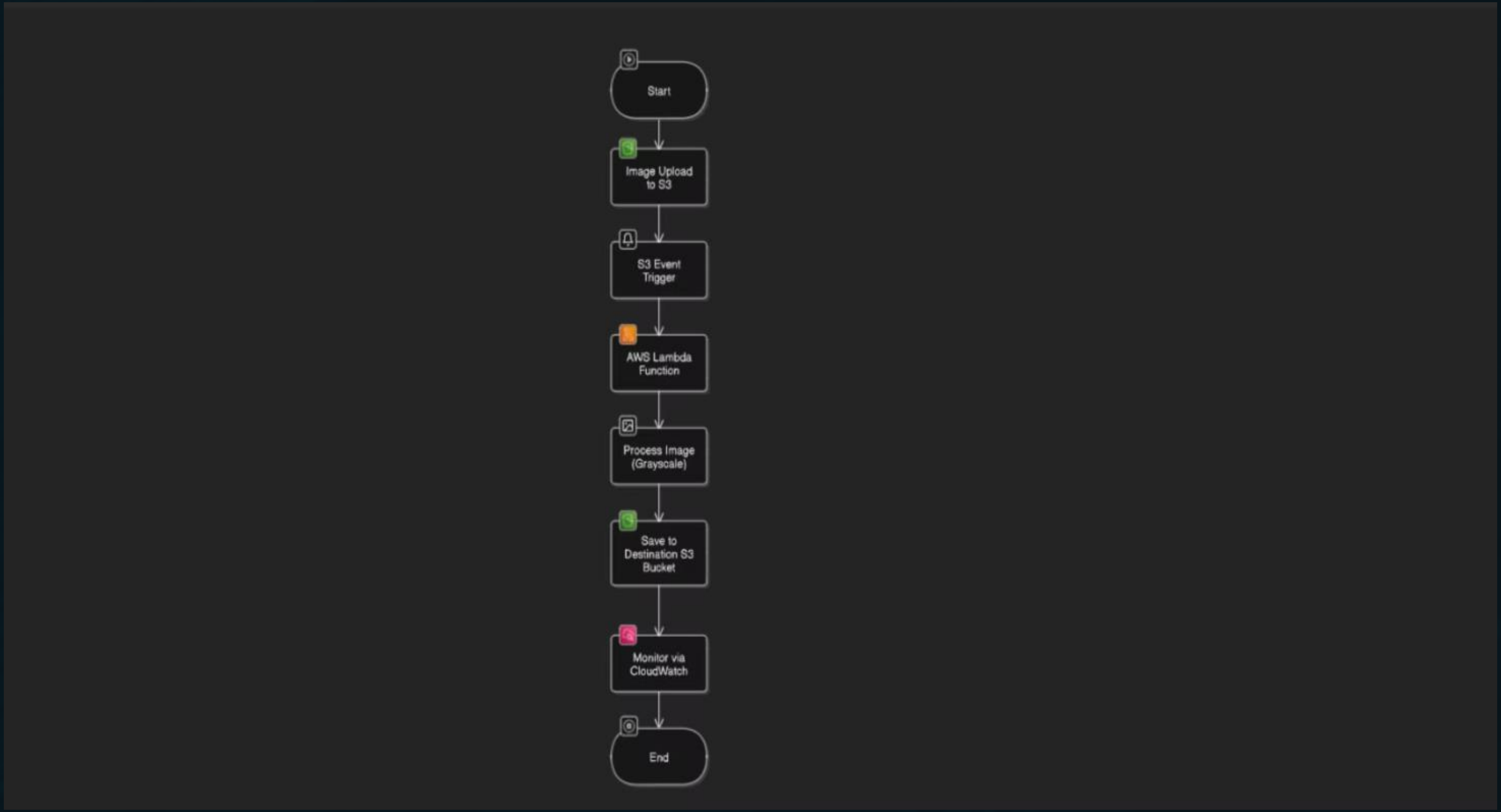
Stores the processed grayscale images

4

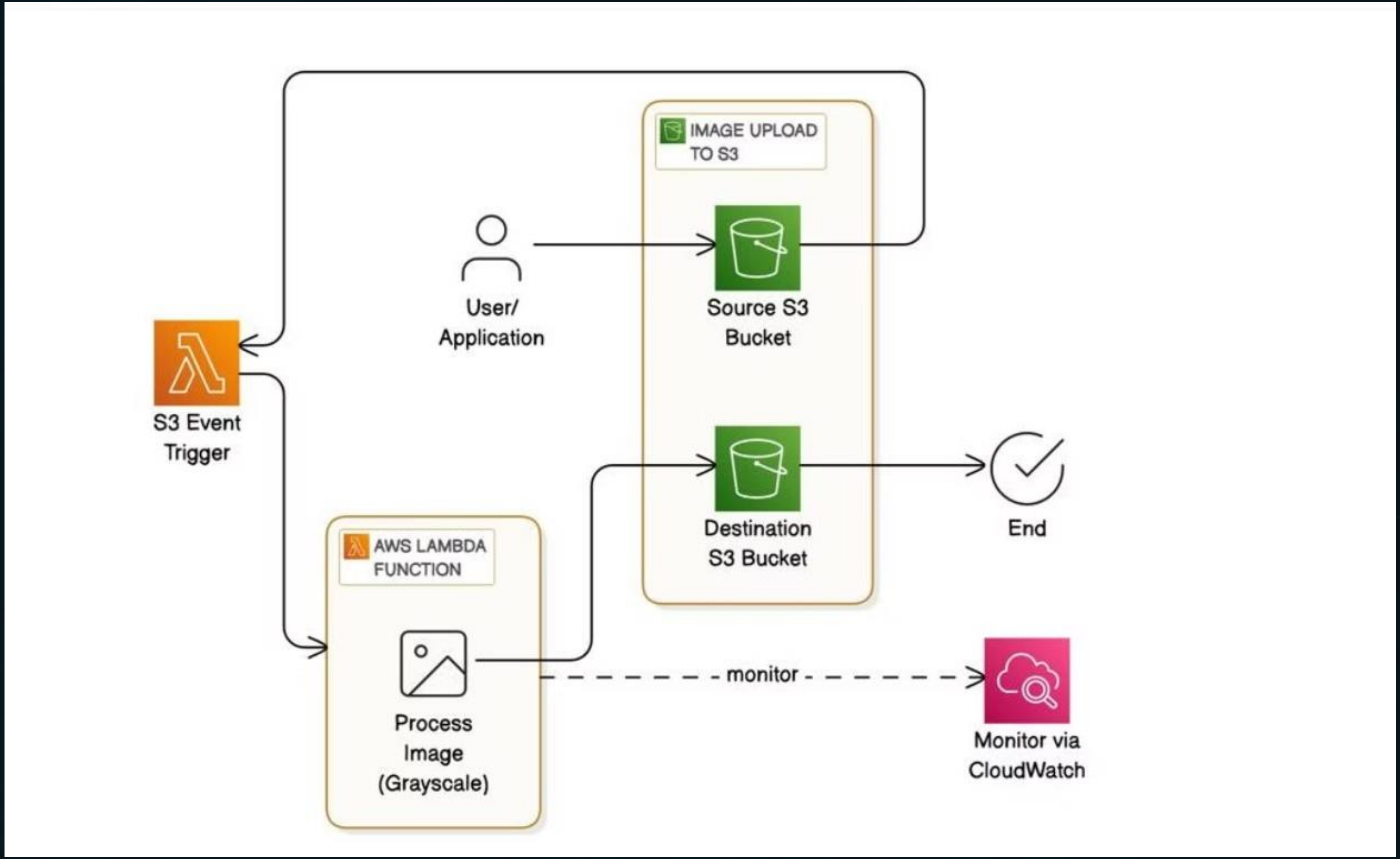
CloudWatch Integration

Tracks metrics like execution time, invocation count, and errors

Flowchart diagram.



Architecture diagram



Architecture Overview

Source S3 Bucket: This bucket serves as the repository for original images uploaded by users. It's configured with appropriate permissions to allow the Lambda function access for image retrieval. The bucket's lifecycle policies are set to manage storage costs and data retention.

AWS Lambda Function: This function is the core of our grayscale image processing. It's written in Python and utilizes the Pillow library for efficient image manipulation. The function retrieves images from the source bucket, converts them to grayscale, and uploads the processed images to the destination bucket. Error handling and logging are implemented to ensure robustness.

Destination S3 Bucket: This bucket stores the grayscale images produced by the Lambda function. It's configured similarly to the source bucket with appropriate permissions and lifecycle policies. The processed images are organized within this bucket to ensure easy retrieval and management.

CloudWatch Integration: We leverage CloudWatch for comprehensive monitoring. This allows us to track key metrics such as invocation counts, duration, errors, and throttles. These metrics are crucial for understanding Lambda function performance, identifying potential issues, and optimizing resource allocation. Custom dashboards and alarms provide real-time visibility into the system's health.

Implementation Steps

1

S3 Bucket Setup

Create source and destination buckets

2

Lambda Function Creation

Write the Python-based Lambda function

3

Trigger Configuration

Configure the S3 event trigger

4

Testing Configuration

Upload test images and verify processed images

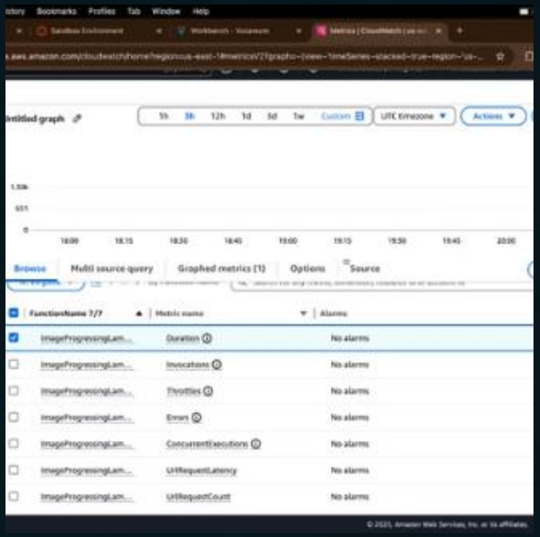
Implementation Steps

1. **S3 Bucket Setup:** Create two S3 buckets, one for the original images and another for the processed grayscale images. Configure access permissions for the Lambda function to read from the source bucket and write to the destination bucket.
2. **Lambda Function Creation:** Write a Python-based Lambda function that takes an image from the source S3 bucket as input, converts it to grayscale using the Pillow library, and uploads the processed image to the destination S3 bucket.
3. **Trigger Configuration:** Configure an S3 event trigger for the Lambda function. When a new image is uploaded to the source S3 bucket, the trigger will invoke the Lambda function to process the image.
4. **Testing Configuration:** Upload test images to the source S3 bucket and verify that the processed images are correctly uploaded to the destination S3 bucket. Use CloudWatch logs to monitor the Lambda function's execution and identify any errors.

Monitoring and Testing

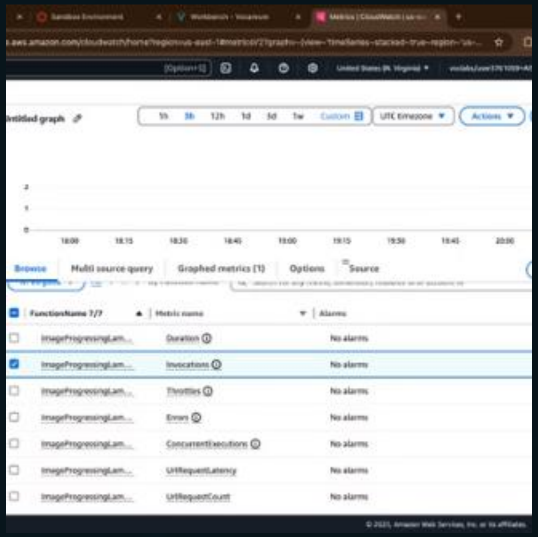
- **Testing Scenarios:** Implement a variety of testing scenarios to evaluate the Lambda function's robustness and scalability under different conditions. This includes single image uploads for baseline performance, multiple concurrent uploads to assess scaling, and stress testing to determine the maximum throughput and breaking point.
- **Scaling Testing Methods:** Conduct rigorous scaling tests by simulating a high volume of concurrent requests. This involves uploading multiple images simultaneously to the S3 source bucket, assessing the Lambda function's ability to handle increased load and maintain responsiveness.
- **Performance Monitoring:** Utilize CloudWatch metrics for comprehensive monitoring of Lambda function performance. Key metrics include invocation count, execution duration, error rates, and throttling events. This allows for proactive identification of performance bottlenecks and potential issues.

Results



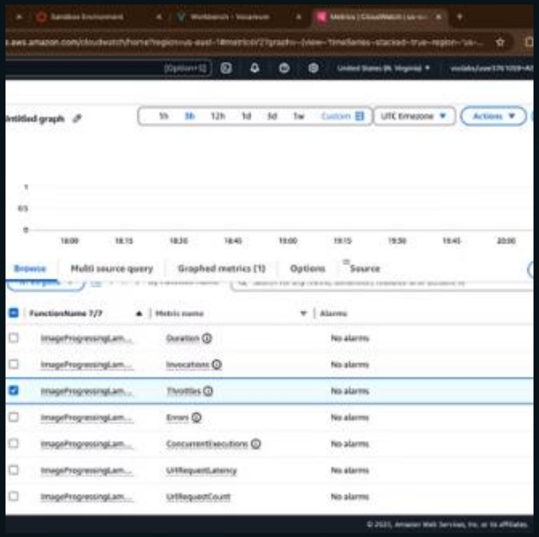
Graphed metrics (1)

	FunctionName 7/7	Metric name	Alarms
✓	ImageProgressingLam...	Duration	No alarms
☐	ImageProgressingLam...	Invocations	No alarms
☐	ImageProgressingLam...	Throttles	No alarms
☐	ImageProgressingLam...	Errors	No alarms
☐	ImageProgressingLam...	ConcurrentExecutions	No alarms
☐	ImageProgressingLam...	UrlRequestLatency	No alarms
☐	ImageProgressingLam...	UrlRequestCount	No alarms



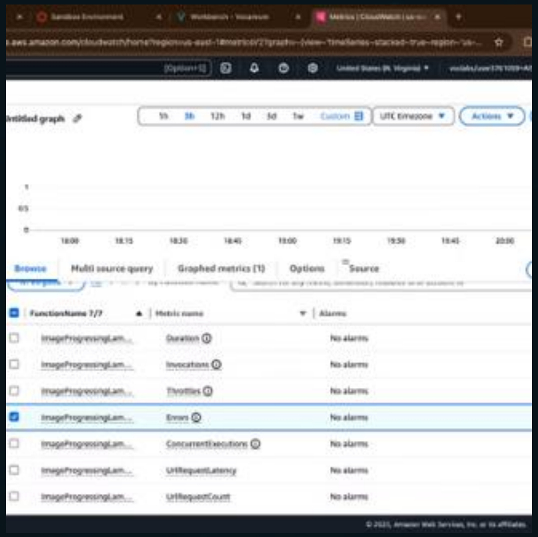
Graphed metrics (1)

	FunctionName 7/7	Metric name	Alarms
☐	ImageProgressingLam...	Duration	No alarms
✓	ImageProgressingLam...	Invocations	No alarms
☐	ImageProgressingLam...	Throttles	No alarms
☐	ImageProgressingLam...	Errors	No alarms
☐	ImageProgressingLam...	ConcurrentExecutions	No alarms
☐	ImageProgressingLam...	UrlRequestLatency	No alarms
☐	ImageProgressingLam...	UrlRequestCount	No alarms



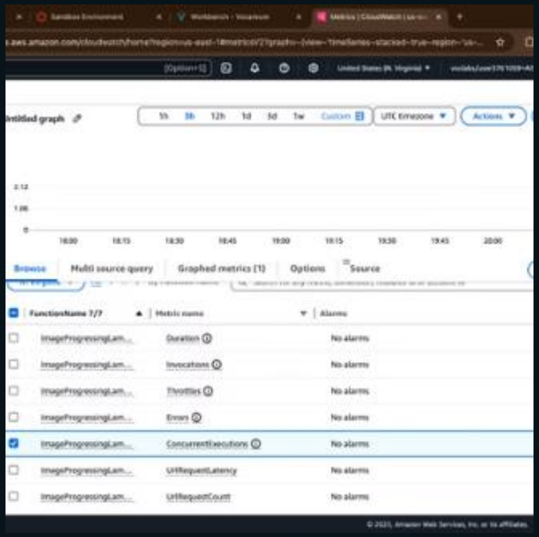
Graphed metrics (1)

	FunctionName 7/7	Metric name	Alarms
☐	ImageProgressingLam...	Duration	No alarms
☐	ImageProgressingLam...	Invocations	No alarms
✓	ImageProgressingLam...	Throttles	No alarms
☐	ImageProgressingLam...	Errors	No alarms
☐	ImageProgressingLam...	ConcurrentExecutions	No alarms
☐	ImageProgressingLam...	UrlRequestLatency	No alarms
☐	ImageProgressingLam...	UrlRequestCount	No alarms



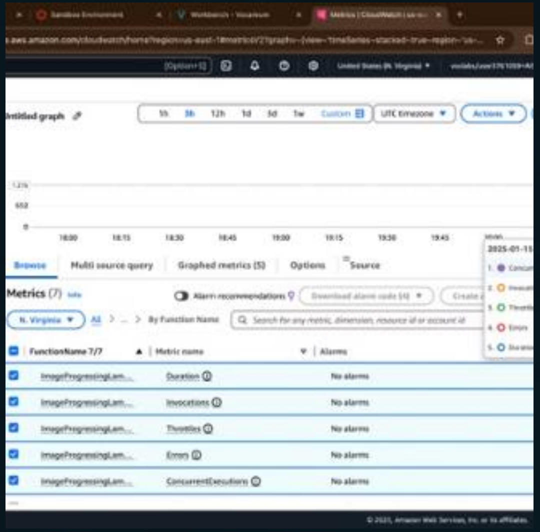
Graphed metrics (1)

	FunctionName 7/7	Metric name	Alarms
☐	ImageProgressingLam...	Duration	No alarms
☐	ImageProgressingLam...	Invocations	No alarms
☐	ImageProgressingLam...	Throttles	No alarms
✓	ImageProgressingLam...	Errors	No alarms
☐	ImageProgressingLam...	ConcurrentExecutions	No alarms
☐	ImageProgressingLam...	UrlRequestLatency	No alarms
☐	ImageProgressingLam...	UrlRequestCount	No alarms



Graphed metrics (1)

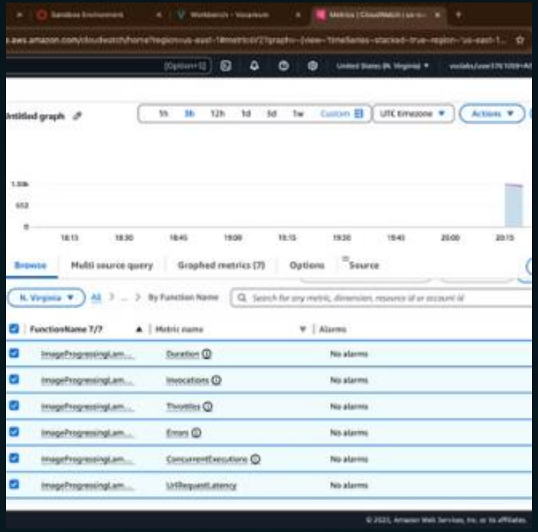
	FunctionName 7/7	Metric name	Alarms
☐	ImageProgressingLam...	Duration	No alarms
☐	ImageProgressingLam...	Invocations	No alarms
☐	ImageProgressingLam...	Throttles	No alarms
☐	ImageProgressingLam...	Errors	No alarms
✓	ImageProgressingLam...	ConcurrentExecutions	No alarms
☐	ImageProgressingLam...	UrlRequestLatency	No alarms
☐	ImageProgressingLam...	UrlRequestCount	No alarms



Graphed metrics (5)

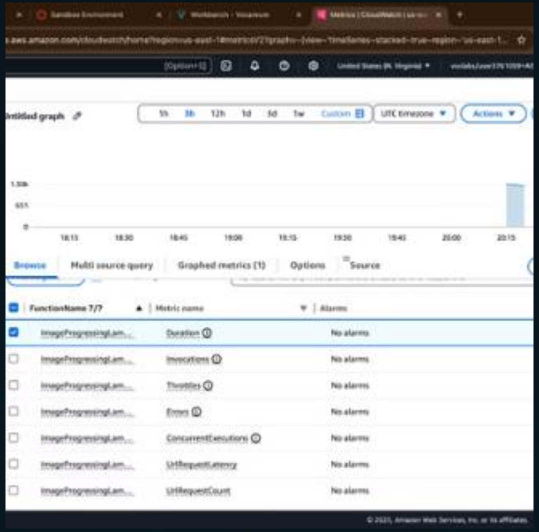
Metrics (7)

	FunctionName 7/7	Metric name	Alarms
✓	ImageProgressingLam...	Duration	No alarms
✓	ImageProgressingLam...	Invocations	No alarms
✓	ImageProgressingLam...	Throttles	No alarms
✓	ImageProgressingLam...	Errors	No alarms
✓	ImageProgressingLam...	ConcurrentExecutions	No alarms



Graphed metrics (7)

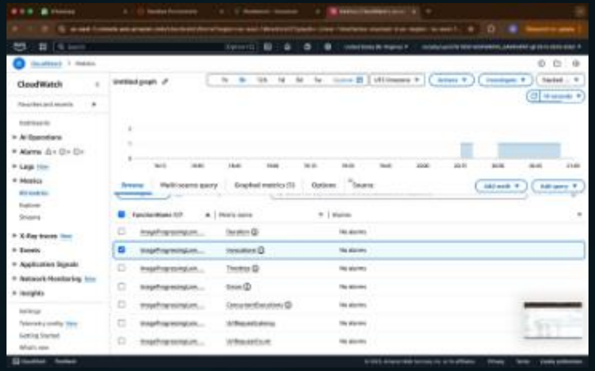
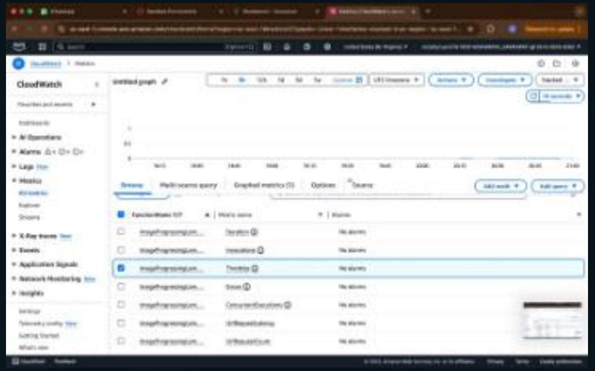
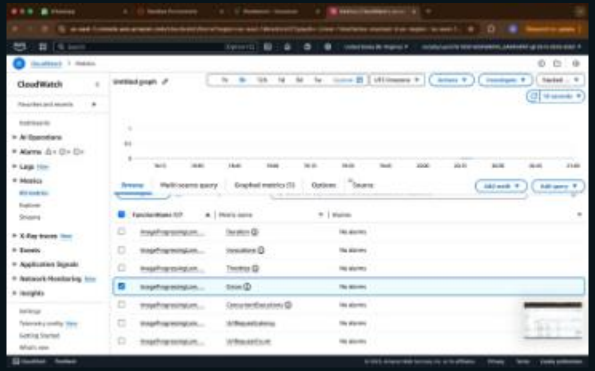
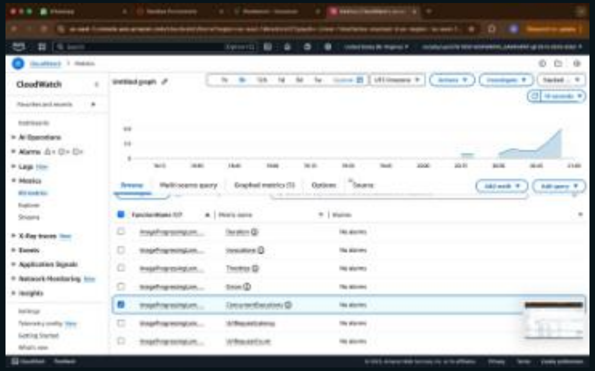
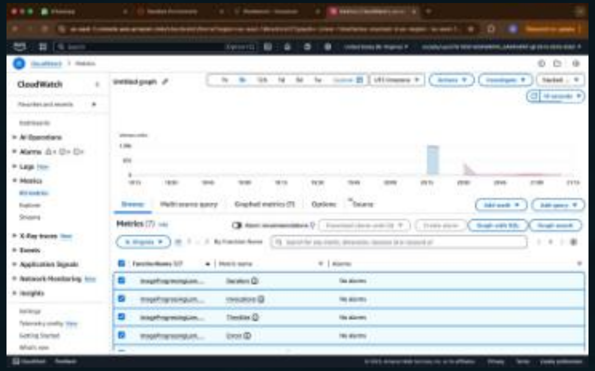
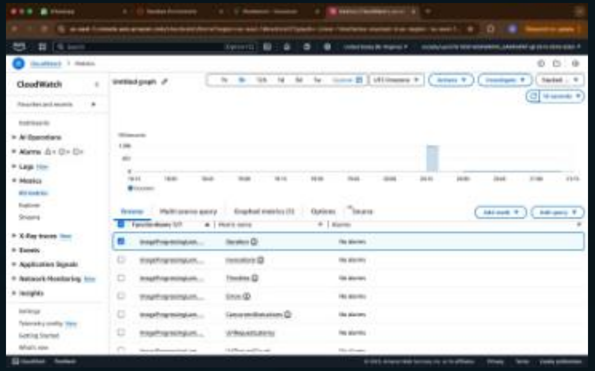
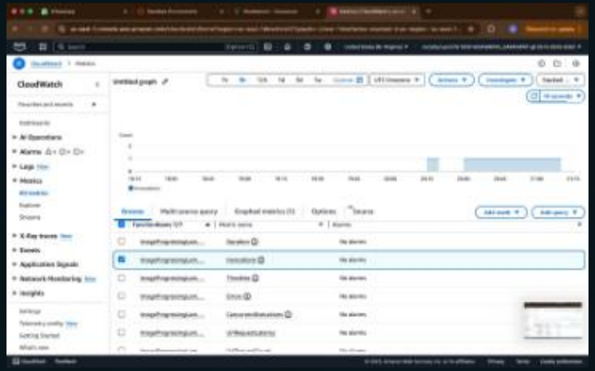
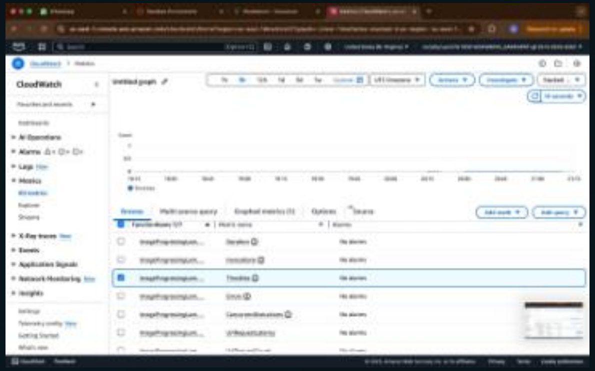
	FunctionName 7/7	Metric name	Alarms
✓	ImageProgressingLam...	Duration	No alarms
✓	ImageProgressingLam...	Invocations	No alarms
✓	ImageProgressingLam...	Throttles	No alarms
✓	ImageProgressingLam...	Errors	No alarms
✓	ImageProgressingLam...	ConcurrentExecutions	No alarms
✓	ImageProgressingLam...	UrlRequestLatency	No alarms



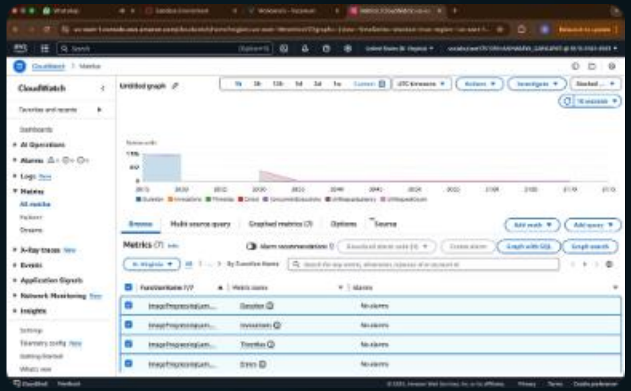
Graphed metrics (1)

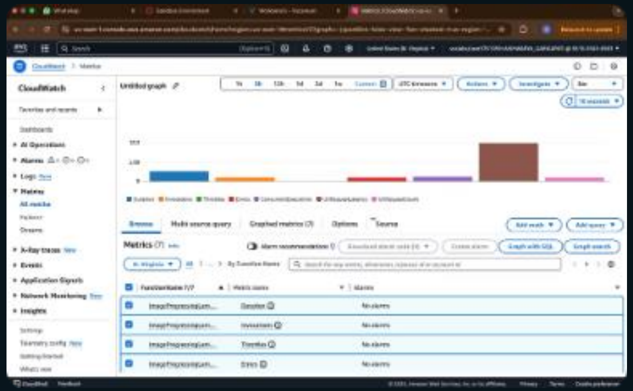
	FunctionName 7/7	Metric name	Alarms
✓	ImageProgressingLam...	Duration	No alarms
☐	ImageProgressingLam...	Invocations	No alarms
☐	ImageProgressingLam...	Throttles	No alarms
☐	ImageProgressingLam...	Errors	No alarms
☐	ImageProgressingLam...	ConcurrentExecutions	No alarms
☐	ImageProgressingLam...	UrlRequestLatency	No alarms
☐	ImageProgressingLam...	UrlRequestCount	No alarms

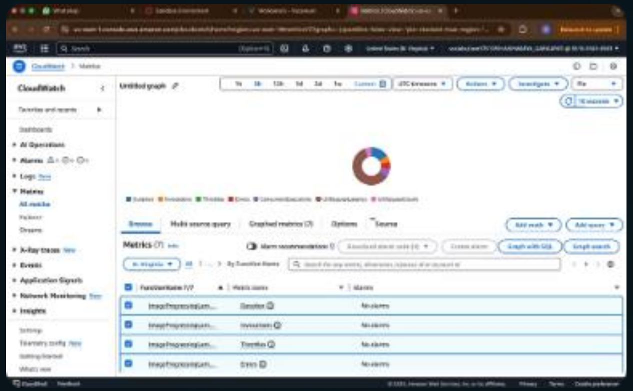
Results

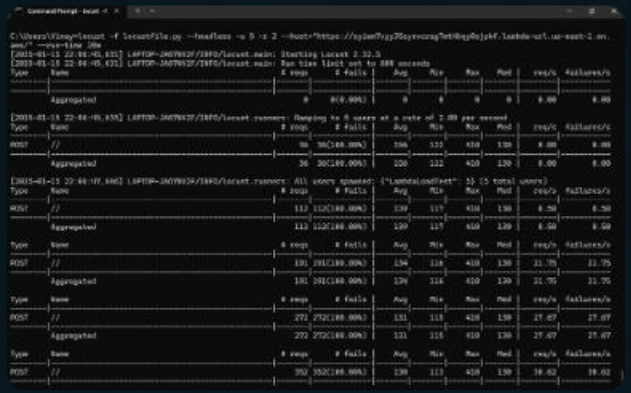









Results

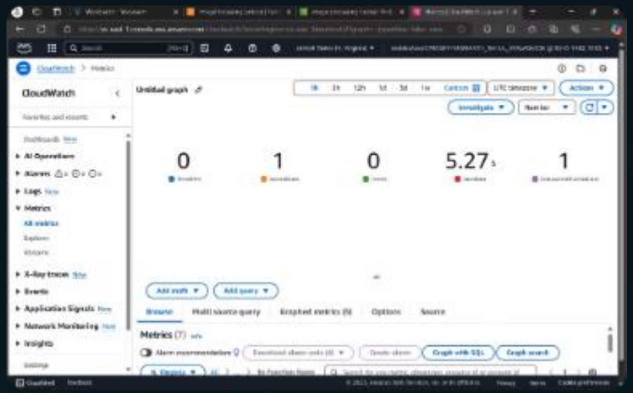


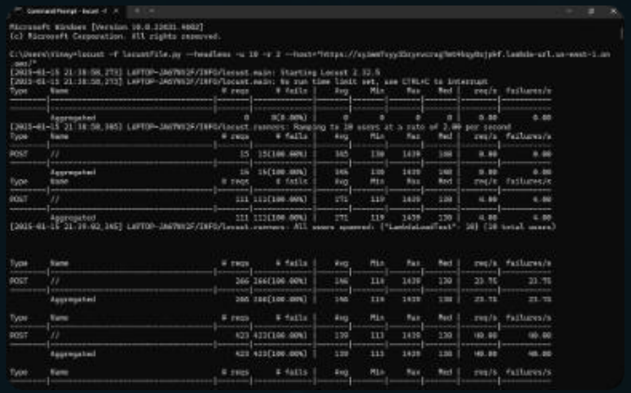






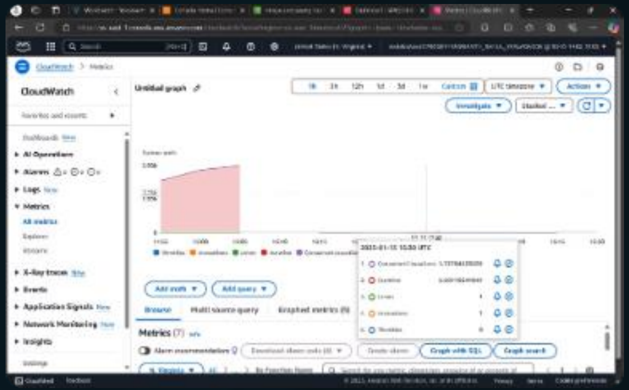
```
C:\Users\Vinay>locust -f locustfile.py --headless -u 5 -r 2 --host="https://[illegible].lambda-url.us-east-1.on.aws/" --run-time 10m
[2025-01-15 22:06:45,631] LAPTOP-JA6TNV2F/INFO/locust.main: Starting Locust 2.32.5
[2025-01-15 22:06:45,631] LAPTOP-JA6TNV2F/INFO/locust.main: Run time limit set to 600 seconds
Type     Name                                          # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                         0     0(0.00%) |      0       0       0      0 |    0.00        0.00

[2025-01-15 22:06:45,638] LAPTOP-JA6TNV2F/INFO/locust.runners: Ramping to 5 users at a rate of 2.00 per second
Type     Name                                          # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                                36  36(100.00%) |    196     122     410    130 |    0.00        0.00
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                        36  36(100.00%) |    196     122     410    130 |    0.00        0.00

[2025-01-15 22:06:47,640] LAPTOP-JA6TNV2F/INFO/locust.runners: All users spawned: {"LambdaLoadTest": 5} (5 total users)
Type     Name                                          # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                               112 112(100.00%) |    139     117     410    130 |    8.50        8.50
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                       112 112(100.00%) |    139     117     410    130 |    8.50        8.50

Type     Name                                          # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                               191 191(100.00%) |    134     116     410    130 |   21.75       21.75
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                       191 191(100.00%) |    134     116     410    130 |   21.75       21.75

Type     Name                                          # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                               272 272(100.00%) |    131     115     410    130 |   27.67       27.67
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                       272 272(100.00%) |    131     115     410    130 |   27.67       27.67

Type     Name                                          # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                               352 352(100.00%) |    130     113     410    130 |   30.62       30.62
--------|--------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
```

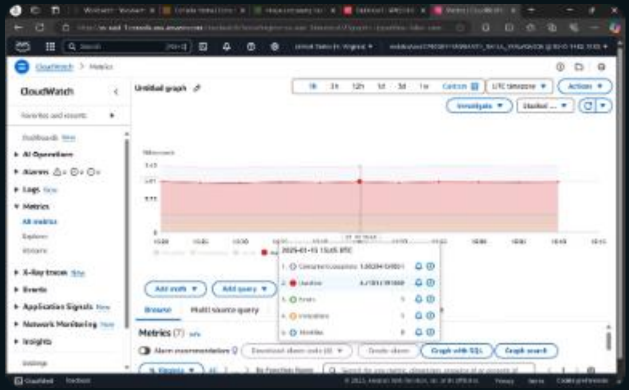


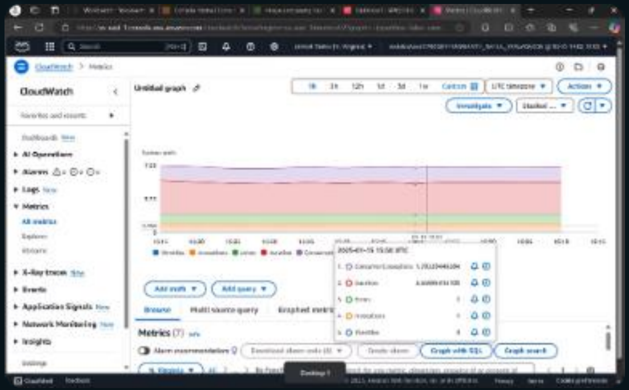


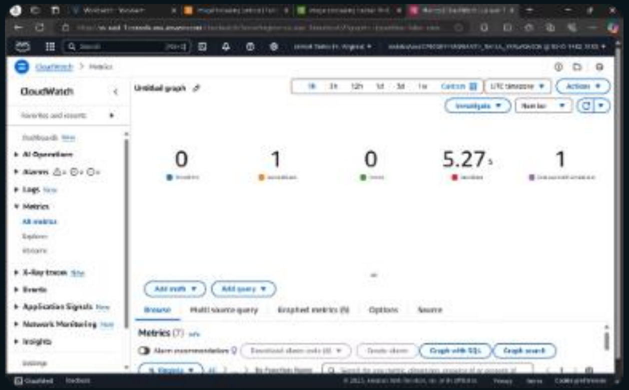
```
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vinay>locust -f locustfile.py --headless -u 10 -r 2 --host="https://[illegible].lambda-url.us-east-1.on.aws/"
[2025-01-15 21:18:58,273] LAPTOP-JA6TNV2F/INFO/locust.main: Starting Locust 2.32.5
[2025-01-15 21:18:58,273] LAPTOP-JA6TNV2F/INFO/locust.main: No run time limit set, use CTRL+C to interrupt
Type     Name                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                 0     0(0.00%) |      0       0       0      0 |    0.00        0.00
[2025-01-15 21:18:58,385] LAPTOP-JA6TNV2F/INFO/locust.runners: Ramping to 10 users at a rate of 2.00 per second
Type     Name                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                        15  15(100.00%) |    165     130    1439    160 |    0.00        0.00
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                15  15(100.00%) |    165     130    1439    160 |    0.00        0.00
Type     Name                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                       111 111(100.00%) |    271     119    1439    130 |    4.00        4.00
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                               111 111(100.00%) |    271     119    1439    130 |    4.00        4.00
[2025-01-15 21:19:02,385] LAPTOP-JA6TNV2F/INFO/locust.runners: All users spawned: {"LambdaLoadTest": 10} (10 total users)


Type     Name                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                       266 266(100.00%) |    146     114    1439    130 |   23.75       23.75
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                               266 266(100.00%) |    146     114    1439    130 |   23.75       23.75

Type     Name                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                       423 423(100.00%) |    139     113    1439    130 |   40.00       40.00
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                               423 423(100.00%) |    139     113    1439    130 |   40.00       40.00

Type     Name                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
```

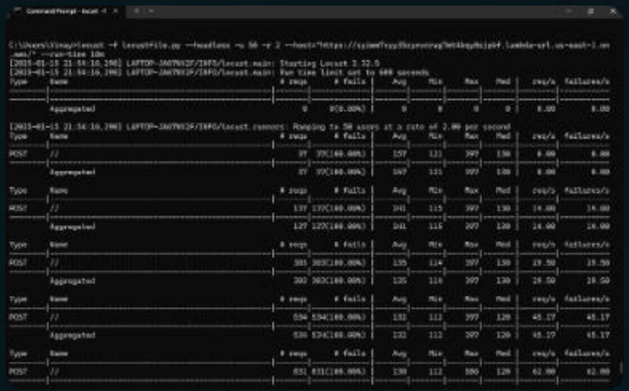
Results



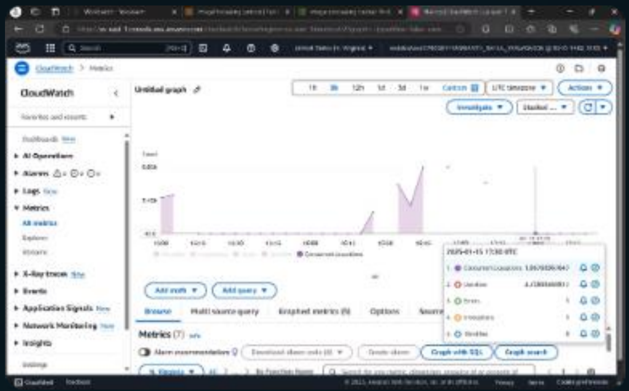




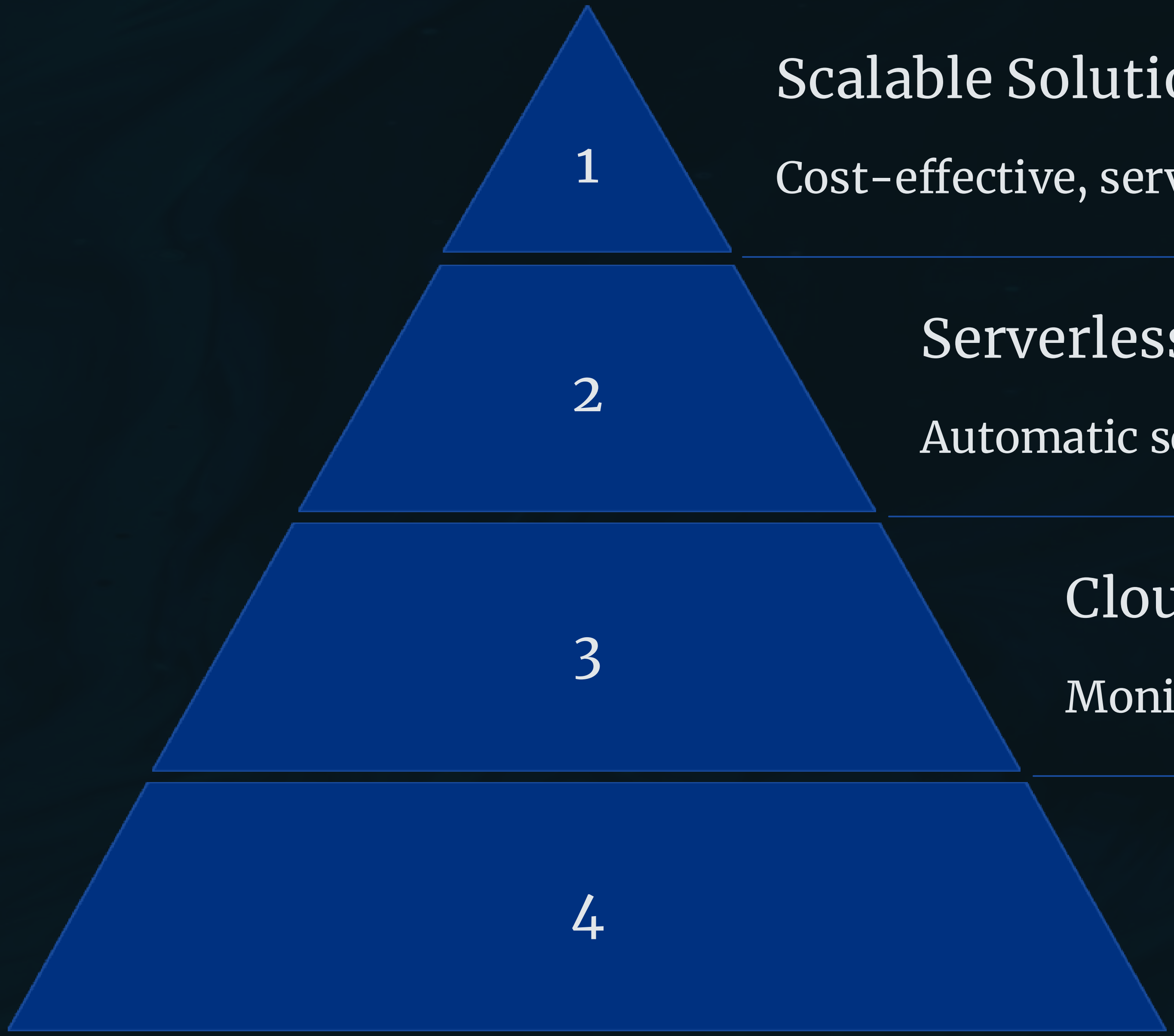




```
C:\Users\Vinay>locust -f locustfile.py --headless -u 50 -r 2 --host="https://[illegible].lambda-url.us-east-1.on.aws/" --run-time 10m
[2025-01-15 21:54:16,298] LAPTOP-JA07N92F/INFO/locust.main: Starting Locust 2.32.5
[2025-01-15 21:54:16,298] LAPTOP-JA07N92F/INFO/locust.main: Run time limit set to 600 seconds
Type     Name                                                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                                                 0     0(0.00%) |      0       0       0      0 |    0.00        0.00

[2025-01-15 21:54:16,298] LAPTOP-JA07N92F/INFO/locust.runners: Ramping to 50 users at a rate of 2.00 per second
Type     Name                                                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                                                        37  37(100.00%) |    157     121     397    130 |    0.00        0.00
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                                                37  37(100.00%) |    157     121     397    130 |    0.00        0.00

Type     Name                                                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                                                       137 137(100.00%) |    141     115     397    130 |   14.00       14.00
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                                               137 137(100.00%) |    141     115     397    130 |   14.00       14.00

Type     Name                                                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                                                       303 303(100.00%) |    135     114     397    130 |   19.50       19.50
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                                               303 303(100.00%) |    135     114     397    130 |   19.50       19.50

Type     Name                                                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                                                       534 534(100.00%) |    132     112     397    120 |   45.17       45.17
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
         Aggregated                                                               534 534(100.00%) |    132     112     397    120 |   45.17       45.17

Type     Name                                                                  # reqs      # fails |    Avg     Min     Max    Med |   req/s  failures/s
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
POST     //                                                                       831 831(100.00%) |    130     112     580    120 |   62.00       62.00
--------|--------------------------------------------------------------------|-------|-------------|-------|-------|-------|-------|--------|-----------
```



Conclusion



Scalable Solution

Cost-effective, serverless architecture

Serverless Capabilities

Automatic scaling to meet workloads

CloudWatch Benefits

Monitoring and optimizing performance

Future Exploration

Serverless technologies for future projects

Conclusion

Scalable Solution

A cost-effective, serverless architecture was implemented, allowing for efficient resource utilization and minimizing infrastructure costs. This scalability ensures that the application can adapt to future growth without significant changes to the core architecture.

Serverless Capabilities

The serverless architecture automatically scales to meet changing workloads, guaranteeing optimal performance during peak demands. This eliminates the need for manual scaling and reduces operational overhead. Resources are automatically adjusted to match the application's requirements in real time.

CloudWatch Benefits

Amazon CloudWatch provided real-time monitoring and performance optimization, allowing for proactive identification and resolution of issues. The comprehensive monitoring capabilities provided valuable insights into the application’s health and efficiency, enabling continuous improvement.

Future Exploration

This project paves the way for future exploration of serverless technologies and their integration in similar projects. The experiences gained will guide the implementation of scalable and efficient cloud solutions for future endeavours.

Thank you!