Project Document: Cloud and Edge Infrastructures

1. Proposed Solution Architecture

Architecture Overview :

The architecture processes IoT traffic data in the following stages:

1. **Upload Client**:
   - IoT branch offices upload their traffic data (CSV files) to an Amazon S3 bucket using the `UploadClient` Java application.
   - Each file is tagged with a unique branch ID and stored under structured folders (e.g., `branchID/date/filename.csv`).

   For example, Amazon S3 > Buckets > newbucket37920>

2. **Summarize Worker**: Lambda Function
   - Triggered automatically when a file is uploaded to S3 using an **AWS Lambda function**.
   - The function reads the uploaded CSV file, processes the traffic data (grouping by source and destination IP and date), and generates summarized results (total flow duration and forward packets).
   - Results are stored back in a separate S3 folder (e.g., `processed/branchID/date/filename-summary.csv`).

   Example : Amazon S3 > Buckets > newbucket37920 > processed > branch001> 2025-01-10> summary.csv

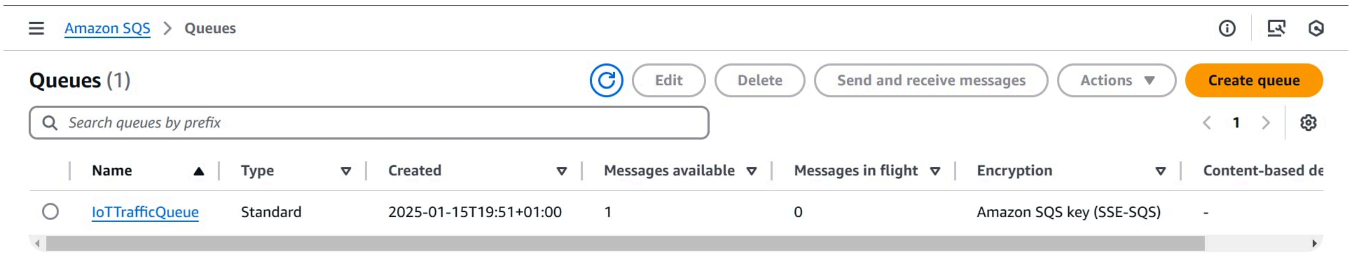
3. **Consolidator Worker**:
   - Triggered by new summarized files in S3 using another **AWS Lambda function**.
   - The worker calculates statistics (e.g., averages, standard deviations) across all available summarized data.
   - Consolidated results are stored in S3 (e.g., `processed/processed/branchID/date/filename_summary_consolidated.csv`).
4. **Export Client**:
   - A Java application fetches processed data (summarized and consolidated) from S3 and generates a final CSV file containing complete traffic information for a device pair.

Mehdi Nemri / Ouadie Boussetta

AWS Services Used

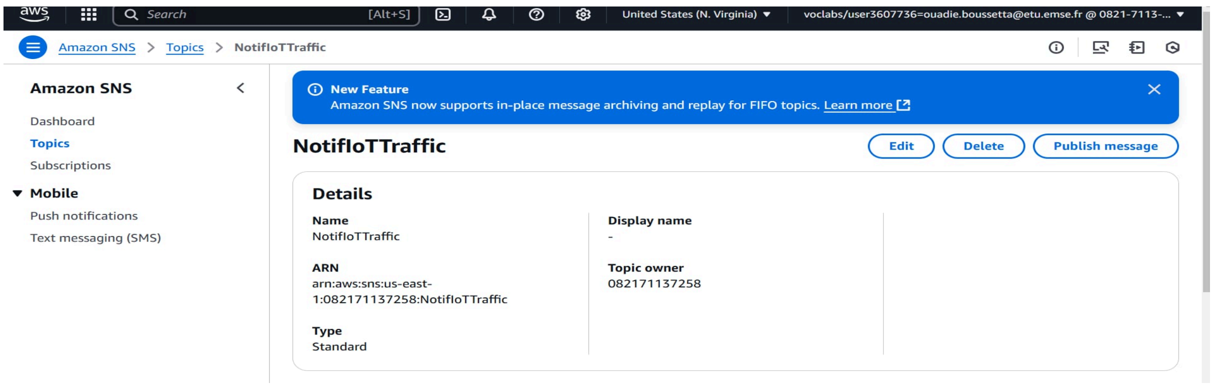
1. **Amazon S3**:
    - Acts as the central storage for uploading, summarizing, and consolidating IoT traffic data.
    - Cost-efficient for large-scale storage and integrates well with Lambda.
2. **AWS Lambda**:
    - Processes files in real-time without needing dedicated servers (serverless)
    - Summarize Worker and Consolidator Worker are implemented as Lambda functions.
    - Provides scalability and reduces operational overhead.
3. **Amazon CloudWatch**:
    - Monitors the Lambda functions and logs their activity.
    - Helps track errors, performance metrics, and resource usage.
4. **IAM Roles**:
    - Ensures secure access for Lambda functions to read/write data in S3.
5. **EC2** :
    - Amazon EC2 instances run workers that fetch messages from SQS, process IoT traffic files, and handle high workloads with scalable compute power**.**

6. **SQS & SNS:**
    - Amazon SNS broadcasts notifications from S3 to subscribed endpoints, ensuring new file events are communicated instantly.
    - Amazon SQS queues these notifications, providing reliable, scalable message delivery for worker processing even during high load or worker downtime.


Amazon SQS > Queues

Queues (1)

Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based de
IoTTrafficQueue	Standard	2025-01-15T19:51+01:00	1	0	Amazon SQS key (SSE-SQS)	-


Search [Alt+S]
United States (N. Virginia)
voclabs/user3607736=ouadie.boussetta@etu.emse.fr @ 0821-7113-...
Amazon SNS > Topics > NotifIoTTraffic
Amazon SNS
Dashboard
Topics
Subscriptions
Mobile
Push notifications
Text messaging (SMS)

New Feature
Amazon SNS now supports in-place message archiving and replay for FIFO topics. Learn more

NotifIoTTraffic

Edit
Delete
Publish message

Details

Name: NotifIoTTraffic
Display name: -
ARN: arn:aws:sns:us-east-1:082171137258:NotifIoTTraffic
Topic owner: 082171137258
Type: Standard

Mehdi Nemri / Ouadie Boussetta

2. Justifying the Proposed Solution Architecture

Why have we used AWS Lambda?

- **Scalability**: Automatically scales based on the number of incoming files.
- **Cost Efficiency**: Charges only for the execution time, avoiding idle server costs.
- **Real-Time Processing**: Processes files as soon as they're uploaded, minimizing delays.

Why have we used Amazon S3?

- **High Availability**: Ensures data is accessible across all AWS regions.
- **Integration**: Directly integrates with Lambda for event-driven workflows.
- **Cost Optimization**: Provides cost-efficient storage options for large datasets.

Why have we used Both Lambda and Java Applications?

- **Lambda**: Handles real-time processing tasks (Summarize and Consolidate Workers) effectively without requiring long-running servers.
- **Java Applications**: Suitable for tasks requiring complex user interactions or controlled offline processing (e.g., Export Client).

Why have we Used Amazon CloudWatch?

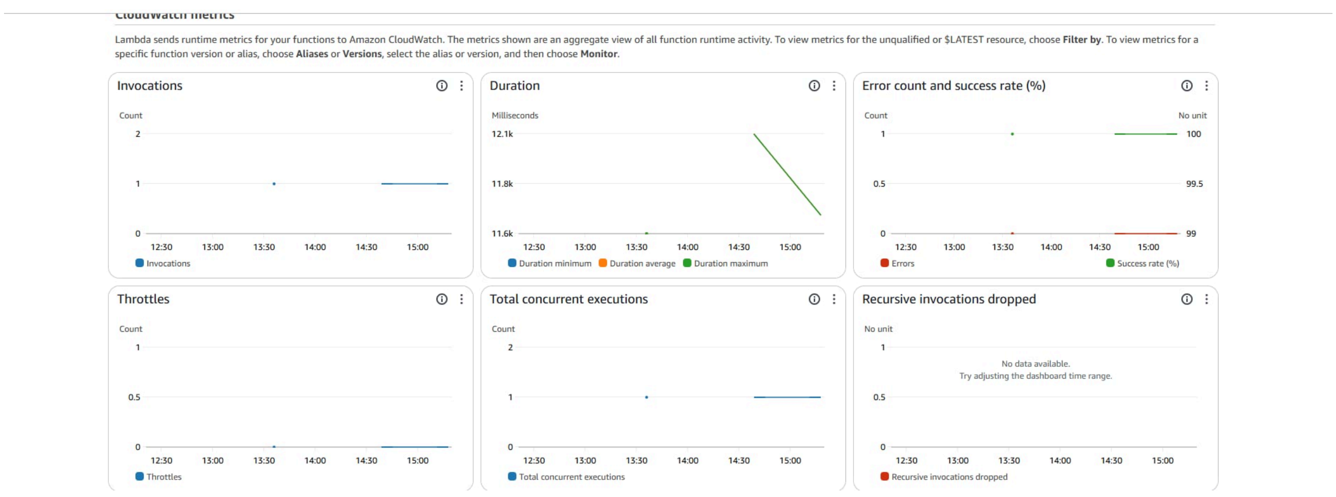
1. **Real-Time Monitoring**:
   - CloudWatch provides detailed metrics and logs for AWS Lambda, Amazon S3, and EC2.
   - Enables real-time monitoring of application performance, execution times, and error rates.
2. **Troubleshooting and Debugging**:
   - Logs from Lambda functions (e.g., SummarizeWorker and ConsolidatorWorker) are automatically sent to CloudWatch Logs.
   - Helps identify issues like input file formatting errors, S3 access problems, or processing bottlenecks.
3. **Performance Insights**:
   - CloudWatch Metrics track key performance indicators (KPIs) such as:
     - Lambda duration and memory usage.
     - S3 read/write operations.
     - EC2 CPU, memory, and network utilization.

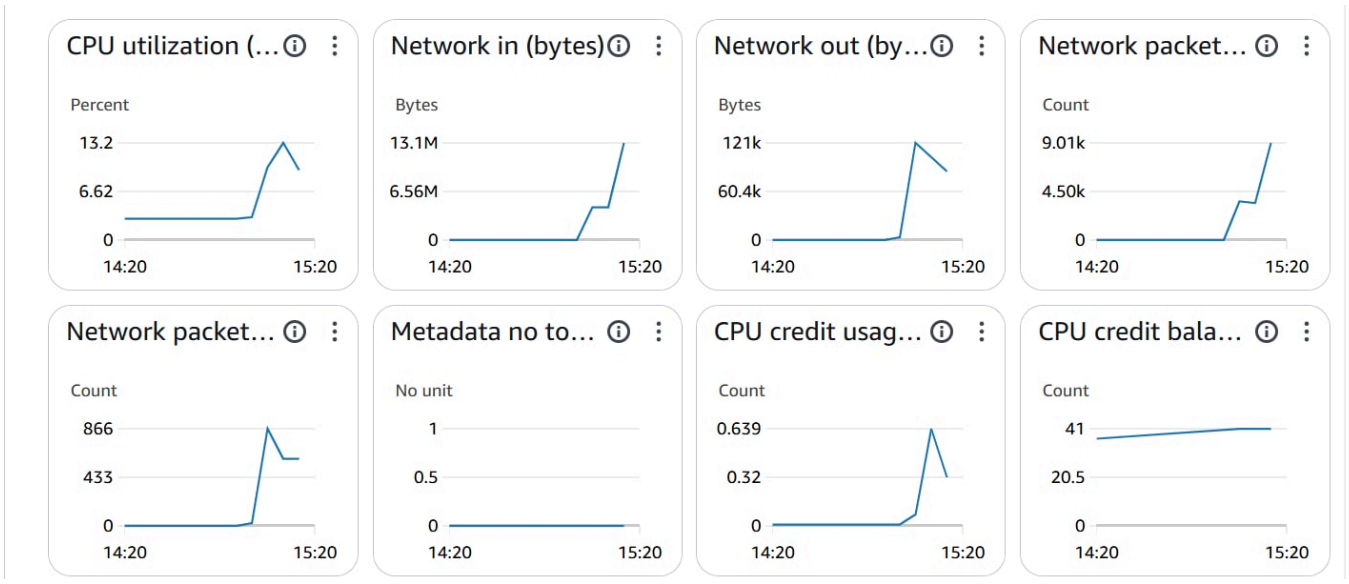
---

3. Quantitative and Qualitative Comparison :

Quantitative Comparison :

Lambda Function :


Lambda sends runtime metrics for your functions to Amazon CloudWatch. The metrics shown are an aggregate view of all function runtime activity. To view metrics for the unqualified or $LATEST resource, choose **Filter by**. To view metrics for a specific function version or alias, choose **Aliases** or **Versions**, select the alias or version, and then choose **Monitor**.
Invocations
Count
Invocations
Duration
Milliseconds
Duration minimum
Duration average
Duration maximum
Error count and success rate (%)
Count
No unit
Errors
Success rate (%)
Throttles
Count
Throttles
Total concurrent executions
Count
Total concurrent executions
Recursive invocations dropped
No unit
No data available.
Try adjusting the dashboard time range.
Recursive invocations dropped



Java Application:

Qualitative Comparison (Lambda vs. Java Application) :

Aspect	AWS Lambda	Java Application
Ease of Deployment	Simple and serverless	Requires setup on virtual or physical machines
Error Handling	Integrated with CloudWatch Logs	Requires manual integration with monitoring tools
Flexibility	Limited by runtime constraints	Suitable for long-running tasks
Maintenance	Minimal (no servers to manage)	Requires regular updates and patching

Mehdi Nemri / Ouadie Boussetta