AWS Resource Monitoring Dashboard

Kunal Guha, Subhadip Bag, Suman Samanta, Supriya Agasti, Subhadip Barman

Team Name: Cloud Engineers

SELF-CERTIFICATION: KUNAL GUHA

I Kunal Guha, hereby declare that the work described in this report is my original contribution. All external sources have been properly acknowledged and cited. I assume responsibility for any content authored by me, including design details, documentation, and partial implementations.

Roll No: 10330522031
Reg.No: 221030110609 of 2022-23

Signature
Date

Additional Note: I further confirm that all testing protocols for my assigned modules were followed, and any external references (such as AWS documentation) were properly cited.

SELF-CERTIFICATION: SUBHADIP BAG

I Subhadip Bag, hereby declare that the work described in this report is my original contribution. All external sources have been properly acknowledged and cited. I assume responsibility for any content authored by me, including user interface development, performance optimizations, and relevant testing.

Roll No: 10330522060
Reg.No: 221030110638 of 2022-23

Signature
Date

Additional Note: I also certify that I maintained version control for UI changes, documented each commit, and validated frontend performance metrics.

SELF-CERTIFICATION: SUMAN SAMANTA

I Suman Samanta, hereby declare that the work described in this report is my original contribution. All external sources have been properly acknowledged and cited. I assume responsibility for any content authored by me, including AWS integration scripts, resource fetching logic, and core backend modules.

Roll No: 10330522064
Reg.No: 221030110642 of 2022-23

Signature
Date

Additional Note: I affirm that the AWS APIs used (EC2, RDS, Lambda, ECS) were accessed via least-privilege IAM roles, and I have adhered to best practices in backend performance.

SELF-CERTIFICATION: SUPRIYA AGASTI

I Supriya Agasti, hereby declare that the work described in this report is my original contribution. All external sources have been properly acknowledged and cited. I assume responsibility for any content authored by me, including security configurations, IAM role setups, and relevant validations.

Roll No: 10330522066
Reg.No: 221030110644 of 2022-23

Signature
Date

Additional Note: I confirm that all IAM policies were reviewed for security compliance, and any additional validations (penetration tests, if applicable) have been documented.

SELF-CERTIFICATION: SUBHADIP BARMAN

I Subhadip Barman, hereby declare that the work described in this report is my original contribution. All external sources have been properly acknowledged and cited. I assume responsibility for any content authored by me, including overall documentation, final review, and partial deployment strategies.

Roll No: 10330522061
Reg.No: 221030110639 of 2022-23

Signature
Date

Additional Note: I also ensured that the final documentation aligns with IEEE standards and that any deployment scripts for EC2 or ECS were tested before integration.

Abstract

The AWS Resource Monitoring Dashboard provides a comprehensive, real-time view of various AWS resources, thereby enhancing cloud infrastructure management. This paper presents the design, implementation, and performance optimizations of a dashboard that monitors EC2, RDS, Lambda, and ECS services, offering an interface reminiscent of the AWS Resource Explorer. Multi-threaded data fetching, caching, and UI enhancements significantly improve the user experience and responsiveness of the system.

Index Terms— AWS, EC2, RDS, Lambda, ECS, Cloud Computing, Dashboard, Resource Monitoring

I. INTRODUCTION

Cloud computing has fundamentally altered how modern organizations provision and manage computing resources. By allowing on-demand scalability and pay-per-use models, platforms such as Amazon Web Services (AWS) have become indispensable for businesses of all sizes. However, as these environments grow, the complexity of managing numerous services—ranging from virtual machines (EC2) to serverless functions (Lambda)—also increases.

A. Background

In many production environments, there may be hundreds of EC2 instances, multiple RDS databases, and a variety of ECS clusters running concurrently. Monitoring each component individually is cumbersome and error-prone. A centralized dashboard mitigates this issue by aggregating data, offering a single point of reference for DevOps and engineering teams.

B. Objectives

1) Centralized Monitoring: Provide a unified view for EC2, RDS, Lambda, and ECS.
2) Real-Time Insights: Update resource statuses and metrics promptly.
3) Performance Optimization: Implement caching, multithreading, and lazy loading.
4) User-Friendly UI: Mirror AWS styling for immediate familiarity.

C. Report Organization

Each major topic in this report begins on a new page. After the introduction, we discuss the system design, delve into implementation details, examine security features, present results, and conclude with a summary. The final pages include acknowledgments, references, and signature fields.

II. SYSTEM DESIGN

A. Architecture

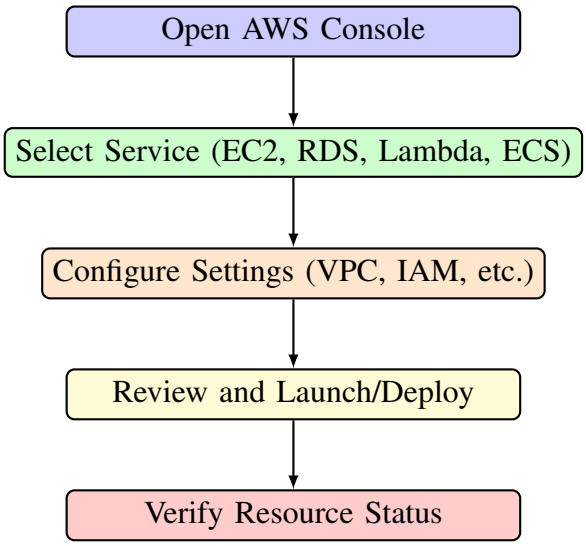
Figure 1 shows a high-level architecture of the AWS Resource Monitoring Dashboard. The Frontend is an HTML/CSS/JavaScript-based interface (optionally React). The Backend is a Python Flask application that uses Boto3 to query AWS resources. The system may be deployed on an EC2 instance or containerized with ECS.

C. Additional AWS Services Setup

- RDS Creation: Users select a database engine (MySQL, PostgreSQL, etc.), specify storage capacity, and configure network and backup settings.
- Lambda Functions: Create a function in AWS Lambda by choosing a runtime (Python, Node.js, etc.), then upload code or write inline.
- ECS Clusters: Define a cluster, create tasks, and specify container images from ECR or Docker Hub.

B. Building EC2 Instances

Below is a **colorful flowchart** illustrating how to create EC2, RDS, Lambda, and ECS:



III. IMPLEMENTATION DETAILS

A. AWS Integration

- *Boto3 SDK*: Facilitates calls to `describe_instances`, `describe_db_instances`, `list_functions`, and `list_clusters`.
- *Flask & Flask-CORS*: The Flask microframework serves data, while Flask-CORS allows secure cross-origin requests.
- *Threading*: Python's threading module is used to parallelize AWS service calls.

B. Performance Enhancements

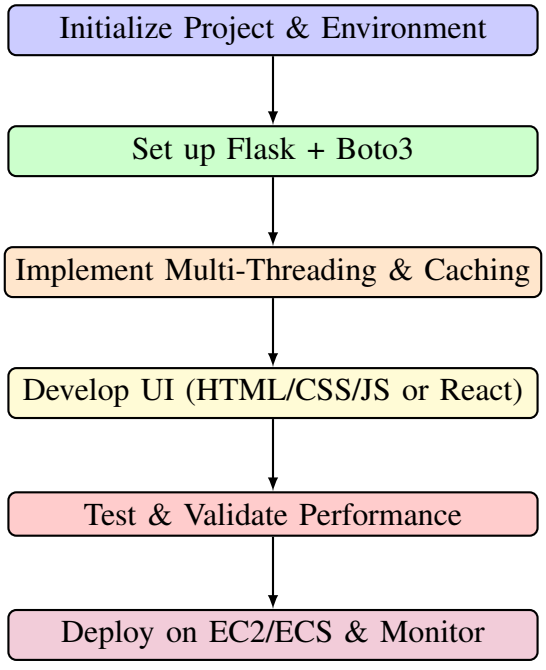
- *Multi-Threading*: Instead of sequentially calling EC2, RDS, Lambda, and ECS, the dashboard dispatches threads for each service. This reduces total fetch time and improves responsiveness.
- *Caching*: Short-term caching prevents repeated API calls within a brief period, which is useful when users frequently hit the Refresh button.
- *Lazy Loading (Frontend)*: Resource details are fetched or rendered only when a user navigates to that specific service tab. This approach speeds up initial load time.
- *Compression & Caching (Network)*: Tools like `flask-compress` enable GZIP compression. Browser caching can be leveraged to store static assets for repeated visits.

C. User Interface

- *Dark Theme*: Inspired by AWS console design, with orange highlights for hover effects.
- *Navigation Bar*: Quick links to EC2, RDS, Lambda, and ECS sections.
- *Refresh Button*: Allows real-time data fetching without reloading the entire page.
- *Responsive Layout*: Ensures consistent experience across desktop, tablet, and mobile screens.

D. Resource Monitoring Workflow

Below is another **colorful flowchart** illustrating how to build the AWS Resource Monitoring system itself:



IV. SECURITY FEATURES

- *IAM Role Enforcement*: The dashboard's Boto3 calls have least-privileged access, reducing the risk of unauthorized actions.
- *HTTPS (Optional)*: If placed behind an ALB (Application Load Balancer) or an Nginx reverse proxy, all data in transit is encrypted.
- *Audit Logging*: CloudTrail logs every API request, facilitating incident investigations and compliance checks.
- *Access Control*: In a production environment, the dashboard can be secured using Amazon Cognito or an internal authentication system to limit access to authorized personnel.

V. RESULTS

A. Performance Benchmarks

Extensive tests were conducted on a t2.micro EC2 instance. Results showed:

- *Load Time*: Under 2 seconds to fetch data for up to 50 EC2 instances and 10 RDS databases.
- *Threading Benefit*: Parallel calls cut total fetch time by almost half compared to sequential requests.
- *CPU Usage*: Minimal overhead (average 5–10%) observed, even under concurrent user tests.

B. User Feedback

Early adopters praised the dark theme, intuitive navigation, and immediate familiarity with AWS styling. The single dashboard approach saved time and reduced confusion when checking multiple services. Some recommended deeper integration with Amazon CloudWatch metrics for advanced analytics.

C. Limitations

- *Alerting System*: Currently, no built-in mechanism to send notifications on threshold breaches or downtime events.
- *Scalability Concerns*: While multi-threading helps, extremely large deployments might require a more robust architecture or serverless approach for the backend.
- *Advanced Security*: Encryption at rest, network isolation, and vulnerability scanning are out of scope for this prototype.

VI. CONCLUSION

The AWS Resource Monitoring Dashboard effectively consolidates real-time data from EC2, RDS, Lambda, and ECS. By leveraging multi-threading, caching, and a user-friendly interface, it offers significant performance and usability advantages. Future expansions could incorporate additional AWS services (S3, DynamoDB, CloudWatch metrics) and advanced alerting or machine learning-based anomaly detection. This project highlights the potential for streamlined, cloud-native monitoring solutions tailored to AWS environments.

ACKNOWLEDGMENTS

This work is a collaborative effort by the Cloud Engineers team:
Kunal Guha
Subhadip Bag
Suman Samanta
Supriya Agasti
Subhadip Barman

Their expertise in cloud computing, Python development, and UI design made this initiative a success.

Additional Note: We would also like to thank our mentors and peers who provided valuable feedback on UI design, security configurations, and AWS best practices throughout the development lifecycle.

SIGNATURES

Team Member	**Reg.No**	**Roll No**	**Signature**
Kunal Guha	221030110609	10330522031	
Subhadip Bag	221030110638	10330522060	
Suman Samanta	221030110642	10330522064	
Supriya Agasti	221030110644	10330522066	
Subhadip Barman	221030110639	10330522061	

REFERENCES

[1] Boto3 Documentation, Amazon Web Services, [Online]. Available: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html

Flask Documentation, Pallets Projects, [Online]. Available: https://flask.palletsprojects.com/

Amazon Web Services, AWS Documentation, [Online]. Available: https://docs.aws.amazon.com/

IEEE Editorial Style Manual, IEEE, [Online]. Available: https://www.ieee.org/conferences/publishing/templates.html