Intelligent AWS EC2 Cost Optimization and Stale Resource Management

Student: Ajad Ramjit Chauhan (BITS ID: 2021wa15838)
Mentor: Shafique Khan
Submission Date: September 2025

Abstract

This project focuses on designing and implementing a cloud-native system to optimize EC2 costs and manage stale resources in AWS. The system leverages AWS-native services such as Lambda, CloudWatch, and Cost Explorer, alongside automation scripts, to monitor, analyze, and clean up unused or idle resources. The primary objectives are to reduce unnecessary cloud expenditure, automate monitoring and resource management, and enhance operational efficiency. The methodology includes configuring CloudWatch agents, generating system metrics through load simulation using stress-ng, and visualizing results via dashboards. Progress so far includes AWS setup, EC2 provisioning, CloudWatch agent installation, stress tests for metric generation, and preliminary dashboard creation. The project has demonstrated effective monitoring and cost tracking mechanisms, laying a strong foundation for automation and optimization in the final stages.

Introduction

Problem Statement:
Cloud users often leave resources such as EC2 instances, volumes, and Elastic IPs running idle, leading to unnecessary costs. Currently, there is no automated system in place to detect and clean up stale resources in real-time.

Objectives and Scope:
- Automate detection and cleanup of stale AWS resources.
- Optimize EC2 usage and reduce idle costs.
- Provide dashboards for real-time monitoring and cost analysis.

Significance of the Project:
The project introduces an intelligent, automated approach to AWS cost optimization, reducing human oversight and enhancing cloud governance.

Literature Review

This work builds upon existing studies in cloud cost optimization and automation. The AWS Well-Architected Framework emphasizes cost optimization and operational excellence. Serverless architectures (Sbarski) highlight automation potential, while Armbrust et al. (2010) discuss challenges in cloud reliability and cost. Li et al. (2020) introduce Lambda-based resilience models, reinforcing the choice of serverless systems. Gartner reports confirm that automation can reduce cloud outages, making this project's automation-oriented methodology highly relevant.

Gap Identification:
Existing methods lack comprehensive stale resource detection and cleanup automation, which this project aims to address with Lambda-based workflows and custom dashboards.

Methodology

The project follows a three-phase methodology:

Design Phase: Define architecture, data flow, and identify AWS services.
Implementation Phase: Configure CloudWatch Agent, simulate EC2 load with stress-ng, develop Lambda automation, and build dashboards.
Evaluation Phase: Compare pre- and post-optimization costs and validate automation.

Tools and Technologies: AWS EC2, CloudWatch, Lambda, Cost Explorer, AWS CLI, Streamlit, Python.

Implementation Plan:
- Setup AWS account and EC2 instance.
- Configure CloudWatch Agent and create dashboards.
- Install stress-ng for generating metrics.
- Develop Lambda automation for stale resource cleanup.
- Build visualization dashboard with Streamlit.

Current Progress

So far, the following tasks have been completed:
- AWS account setup and configuration.
- EC2 instance launched with CloudWatch Agent enabled.
- stress-ng installed and used to simulate workloads.
- CloudWatch dashboards created to monitor CPU, memory, disk, and network usage.

Preliminary Results:
The dashboards successfully capture performance metrics, validating monitoring setup.

Challenges Faced:
- Learning curve in setting up CloudWatch custom metrics.
- Managing IAM roles and permissions for secure automation.
- Initial delay in configuring dashboards for multiple metrics.

Solutions:
- Used AWS documentation and community forums.
- Applied IAM best practices for role-based access.
- Automated dashboard widget creation using scripts.

Timeline for Remaining Work

Week	Task	Deliverable
1-2	Project Setup & Requirement Analysis	AWS Free Tier account, Requirements Document
3-4	EC2 Instance Launch & Monitoring	EC2 running, CloudWatch metrics
5-6	Load Simulation & Metrics Collection	stress-ng tests, data collected
7	Dashboard Creation	CloudWatch dashboards
8-9	Cost Analysis & Automation	Cost Explorer scripts, Lambda setup
10-12	Stale Resource Detection	Boto3 scripts, dashboard integration
13-14	Lambda Automation & Testing	Cleanup automation, notifications
15-16	Integration & Finalization	Full system demo, cost comparison

References

1. Amazon Web Services. (2023). AWS Well-Architected Framework. 2. Sbarski, P. (2017). Serverless Architectures on AWS. Manning Publications. 3. Armbrust, M., et al. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58. 4. Li, Z., et al. (2020). Serverless resilience with AWS Lambda. IEEE Cloud Computing. 5. Gartner. (2022). Cloud cost management and optimization trends. 6. AWS Documentation. (2024). CloudWatch Agent configuration. 7. AWS Documentation. (2024). Lambda for automation. 8. Boto3 Documentation. (2024). AWS SDK for Python. 9. AWS CLI Documentation. (2024). Cost Explorer commands. 10. Streamlit Documentation. (2024). Dashboard development with Streamlit.