IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

Cloud Guardian: An AI-Assisted Edge-Based AWS Audit and Optimization System

Mr. E. Lingamurthy[1], Md Majid[2], P. Mallikarjun[3], M. Yashwanth[4]

Computer Science & Engineering – Internet of Things, Malla Reddy University, Hyderabad, India[1-4]

Abstract: Cloud providers like Amazon Web Services (AWS) are easy to deploy and operate applications on, but at the same time, they are difficult to keep track of costs and security. It is found that some users have been inadvertently keeping idle virtual machines, unattached storage volumes and outdated snapshots in their accounts, resulting in wastage. Besides, security misconfigure like ports opened to internet can lead to security risks. It is impossible to identify these issues manually since most cloud infrastructure is made up of complex structure of interrelated resources and huge amount of configuration data. This paper proposes a cloud audit and optimization tool Cloud Guardian to address this problem, which is a command line-based tool that performs cloud audit on user's AWS infrastructure, with the aim of identifying the most expensive and insecure resources. Cloud Guardian performs cloud infrastructure audit by collecting resource data, analyzing resource utilization, and using a rule- based analysis approach to flagged resource inefficiency and insecurity. In addition, Cloud Guardian also calculates a health score of the cloud account to make the analysis results more understandable. To make the analysis results more understandable, an AI advisor module is also built into the tool to explain the problems and suggest the solutions.

Keywords: Cloud Computing, AWS, Cloud Governance, Cost Optimization, Security Analysis, Cloud Audit, Command-Line Interface (CLI), Rule-Based Analysis, AI Advisory Systems, Cloud Resource Monitoring, Health Scoring, Edge Computing, ESP32, Credential Isolation, DevOps Automation, Cloud Security, Infrastructure Optimization.

I. INTRODUCTION

Cloud computing is the favored platform for hosting applications due to the fact that users are capable of creating and maintaining resources without relying on physical resources [1], [2]. Services that can be provided through AWS including virtual machines, storage and network elements can be instantly provisioned and configured to the users' requirements [8]. While this is an advantage to users and students it becomes a problem in terms of maintaining the resources in that users usually "forget" to switch of virtual machines they are not using, to remove redundant storage that is not attached to any virtual machines and also to remove stale snapshots [6]. Although the user is not actively using the resources, they are incurring costs to the user [6].

Configuration of security settings is also a critical part in cloud infrastructure. In the cloud computing there is a set of security groups and access controls offered for resources' security in AWS [12]. Inappropriately configuring security settings might be one of the reasons of exposing resource to the Internet by, for example opening port like SSH or RDP to each IP [11].

Most of the actual tools providing a cloud monitoring service only offer a dashboard with some raw metrics [4]. The tools don't necessarily bring clear explanation about the actual cost impact of unused resources or insecure configuration. The user is shown that a resource is underutilized at 70%, but is left to sort through it himself, find an action, and then enact it. The information is not very helpful, especially to people unfamiliar to cloud computing [8]. Also, storing the cloud credentials locally on the computer from which a monitoring tool is run to give the cloud provider the ability to access the AWS account would cause an AWS security leak [11]. So something needs to be found to be able to analyze the resources and also store them securely.

In order to address these problems, we present Cloud Guardian, an Amazon Web Service auditing and optimization tool based on command line interface, which aims to help user understand and upgrade their cloud environment. We collect various information of resources such as EC2, EBS, AMI, Security Group, and IAM users, analyze resource usage and dependencies, and find unused or deficient resources by rule, based auditing engine, meanwhile, find security risk and cost waste [6]. Finally, we calculate a health score to represent the overall health level of the cloud account so that users could know intuitively whether they have a healthy cloud environment.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

Cloud Guardian also has an advisory feature that gives explanations and recommendations based on the identified results. This is important as it helps the user understand why a resource is identified as inefficient or risky and what can be done. Additionally, the system brings forth an edge credential protection method through the use of an ESP32 microcontroller [17]. This is important as it ensures that AWS credentials are not stored in the host computer but rather in the external device and only retrieved when needed.

The primary goal of this project is to build a tool that is straightforward to use, functionally efficient, and cost, effective in assisting users to monitor their AWS resources and to minimize their expenditure, as well as security risks [6], [11]. Cloud Guardian offers a methodological process through automated examination, lucid suggestions and security conscious management of credentials.

II. LITERATURE SURVEY

Survey 1: Multi-Time Series Forecasting for Cloud Resource Utilization
In cloud resource allocation, understanding the change of usage over time is a crucial aspect. Researchers have recently investigated the availability of machine learning models integrated with multi, time series forecasting methods to predict resource utilization of clouds. These systems analyze historical data of resource as CPU, memory, IP, and disk activity for multiple services on various time scales and for several periods. Through these processes, they be able to understand the tendency of the data, and then using machine learning models to predict future demand, and how the cloud platform should reallocate resources. This approach optimizes system efficiency and avoids both excess and lacking resources. Nevertheless, these systems only focus on the predict usage trend and improve resource allocation efficiency at system level, Users can not gain any external help in finding idle resources and get exact suggestions to meet current requirement.

Survey 2: Cloud Resource Optimization and Cost Management
Cloud resource optimization has become an essential research domain since it is worsened by the high costs implied by inappropriate usage of resources. Researches state that many users still keep cloud resources active when they are not used, for instance inactive virtual machines, unattached volumes of storage or Elastic IP Addresses. These “unnecessary” resources cost money, although we gain no value from them. Therefore, cloud service providers give their customers tools as AWS Cost Explorer, enabling them to visualize the billing and billing trends. The users are able to see billings data in graphical and report format, providing the ability to follow the billings for services over time. Though, only providing visibility, the information users gain does not help in optimizing resources automatically. Users should analyze the data and make the decision of what is inefficient, as not trivial.

Survey 3: Cloud Security and Misconfiguration Challenges
Security misconfiguration is one of the most common problems to occur in cloud environments. It has been shown that all insecure security configurations such as associated ports are open, unused access permissions, security groups being misconfigured etc. Can make cloud resources vulnerable to attacks. Misconfigurations can be a result of lack of monitoring or lack of understanding of cloud security. Many cloud security tools are available that can find the vulnerabilities and configuration errors however they generate their finding in a complicated technical language, which a person without knowledge of cloud security cannot understand. Therefore, there is need of a system that finds the security issues and show it to user in a way that user can understand.

Survey 4: Artificial Intelligence for Cloud Management
The application of artificial intelligence in cloud computing is growing rapidly to enhance the areas of monitoring, detection of abnormalities and resource management. AI enabled solutions can analyze vast amount of cloud information and detect abnormal behaviour or predict future resources requirement. They help in enhancing operation efficiency by eliminating human monitoring effort. Machine learning enabled models can learn the pattern in data and surface inefficiencies or system abnormalities. However, most of the existing AI based solutions in cloud are only focused on prediction and abnormality detection. There are very less that provide comprehensible explanation and actionable recommendations to make use of knowledge for improving the environment. The solutions are also missing explicable advisory support features.

III. SYSTEM ANALYSIS

Cloud environments typically have many interconnected resources, such as compute instances, storage volumes, networking components, and identity and access control configurations [8]. Managing these resources can be difficult, as monitoring and analyzing them on an ongoing basis is necessary to make sure unused resources do not incur costs and that

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154113

bad security configurations are not present [6], [11]. Despite this need, many users do not regularly audit their cloud environments, as manually checking each resource and configuration is too complicated.

The core aim of the proposed system, Cloud Guardian, is to conduct an automated analysis of a set of AWS cloud resources and assist a user in optimizing the cost and security profile of this set [6]. The system is made to use data aggregated from various AWS resources to analyze the state of usage of each of these resources and identify potential conditions of waste or risk (such as inactive compute instances or unattached storage volumes, out of date snapshots, unattached Elastic IP addresses, insecure security groups) [6], [11].

Relationship between resource, another important consideration when analyzing is the relationship between resource. In a cloud setup, resource is not stand alone [8]. For examples, an EC2 instance could have one or more attached volumes, and those volumes could have snapshots. If the instances are not use in any customers activity, the attached volumes, and the snapshots could be fallen into the unnecessary cost as well [6]. System should analyse those resource in the context rather than separately. Cloud Guardian do this by construct an internal classes, a model of resource and relationship between them to analyse.

Uptake about usability and accessibility is also very important. The majority of current cloud analysis tools use graphical dashboards, which is useless in case one want to automate the process or integrate it in a development pipeline. To overcome this problem, Cloud Guardian has been designed as a command, line interface tool, so you can run cloud audits with a simple command and be able to integrate it in scripts, automations and IDEs. Nevertheless, the command, line approach makes Cloud Guardian very lightweight and always available.

Security is another issue with the cloud systems [11]. AWS credentials will allow complete access to the cloud resources. If kept on the local system where the application resides then there is a risk of the AWS credentials being compromised [11], [12]. This system will implement an edge based credential isolation scheme whereby the credentials are stored on a separate external device, (ESP32), where they will be used only when necessary [17].

Furthermore, the system is capable of generating a number output which enables the users to take appropriate decisions. In addition to presenting raw figures, it receives the data, applies various rules of analysis, and evaluates a health score which is the measure of the health of the cloud space. This brings about an immediate understanding whether the infrastructure is healthy or not and further provides advisory information to the users.

All in all, the system analysis demonstrates that Cloud Guardian offers an automated, fully, integrated system for auditing of cloud infrastructure. It convincingly tackles its critical problem areas: identifying cost inefficiencies, locating vulnerabilities, examining resource relationships and safeguarding credentials [6], [11]. The aggregation of all of these in the format of a Linux command, line tool, makes the system an immediately useful and impressive tool.

IV. PROPOSED SYSTEM

The proposed system, Cloud Guardian, is a command-line based cloud audit and optimization tool that intends to analyze AWS environments and help users in identifying cost inefficiencies and security risks. The system is based on a modular pipeline architecture, where each stage of the pipeline is intended to accomplish a specific task such as resource collection, metric calculation, rule analysis, and advisory generation.

The described system Cloud Guardian is a cloud audit and optimization CLI tool to analyse the AWS environments and guide the user to identify cost inefficiency and security threats. It works on a pipeline architecture in which different modules are developed to do different jobs (resource collection, metric check, rule check, advisory generation etc.). This makes the system flexible and easy to add more modules.

The system initiates Cloud Guardian's execution once the user issues the audit command from the CLI interface. AWS credentials are loaded, first from the local system and then from an authenticated ESP32 edge device (if available). With authentication established, the orchestrator part of the system begins by executing an audit, collecting all the necessary inventory of the different AWS resources (EC2, EBS, snapshots, Elastic IP, security groups, IAM users, etc.). It then populates the internal data store (CloudState) which serves as the unique source of truth.

In addition to resource discovery and data collection, resource usage data like CPU consumption are retrieved from compute resources to determine if the resource is idle or utilized. In addition, the relations between resources (like

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

volumes related to instances, or snapshots relating to volumes) are examined. Using this kind of relation the system will discover indirect wastage of resources (like unused instances having storage volumes mounted).

The data is then analysed by the rule engine against a defined set of cost and security rules. Rules are implemented to identify conditions like: compute resources idle, un-associated storage devices, orphaned Elastic IP addresses, unnecessarily exposed network ports, etc. All detected conditions are logged as findings, with each finding being rated as to it's severity and predicted impact. From the number and severity of the findings a health score can be computed indicating the condition of the cloud resources.

In order to enhance the user experience, an advisory module was added to the system to explain why issues are flagged up and what can be done. This can assist users to understand where the inefficiency lies and what measures can be taken to address it. Furthermore, Cloud Guardian also implement edge, based credential isolation mechanism through ESP32 micro, controller. Sensitive credentials will not be permanently stored on the host machine, but they are securely stored on the edge, based peripheral and can only be retrieved when needed.

The resultant output is presented in a user, friendly manner via command, line interface and the report feature generates reports that may be needed for documentation purposes or other forms of analysis.

Advantages

- **Automated Detection of Cost Waste:** The system helps users to save costs on cloud infrastructure by identifying idle resources like unused EC2 instances, un-attached EBS volumes and un-used Elastic IPs.
- **Improved Cloud Security Analysis:** It reports insecure settings (like open ports, open security groups) to users so that security issues can be addressed promptly.
- **Context, Aware Resource Evaluation:** It looks at relationships between resources instead of resources themselves which increases the validity of the results and avoids bad recommendations to optimize something that does not actually need to be optimized.
- **Health Score for Infrastructure Assessment:** Cloud Guardian uses a combined health score that includes both the cost and security findings in an easy, to, understand format for the users.
- **Secure Credential Protection using ESP32:** Security is increased as AWS credentials would be on a separate ESP32 device, reducing the chance that a credential could be exposed on the host.
- **Command, Line Based Automation Support:** This CLI, based design can be readily integrated into automation scripts and the overall DevOps lifecycle for production environments.
- **Provides Clear Recommendations:** The advisor is a functional element that advises the user of identified problems and proposes solutions, thus enabling the user to make reasoned decisions without extensive cloud knowledge.

V. METHODOLOGY

A. System Architecture

The system architecture of Cloud Guardian uses a layered and modular architecture. This modular and layered architecture will help to efficiently analyze cloud resources, scale out, store credentials in a secure manner. The architecture follows a pipeline style architecture where each component plays its part in the collecting, analyzing and reporting of cloud audit information.

In the top layer the system starts from the CLI (Command Line Interface), it is the point where the user interacts with the system. Through the CLI, the user will run commands for auditing initiation and analyzing. The CLI should accept input from user, call the audit process and output the results. The interface allows the system to integrate well in automated process and development tool.

When the audit is started Orchestrator element is responsible for the flow of execution. Orchestrator component plays the role of the main control module. It directs the different stages of analysis; It is responsible to assure that resources inventory is collected, then metrics are collected, then rules are evaluated in proper sequence. Hence this kind of process would guarantee more reliable system since all necessary information required for analysis will be available.

Below is the Resource Collection and Metrics Collection. Resource collection module receives inputs from AWS and pulls resource information from AWS cloud like EC2, EBS volumes, snapshots, Elastic IPs, Security Groups and IAM users. Metrics collection module collects statistics like CPU utilization to understand whether the resource is being used actively or it is idle. This data can be then used for intelligent decision making.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

The data gathered is centrally stored in an internal data model (CloudState). This part is the actual storage unit of the system and holds information on resources, their links and analysis results. The centralized model prevents redundancy in the different processing phases.
The Rule Engine processes the data after it has been gathered and stored. Rule Engine process the gathered data by applying set of cost rules and security rules and identifying issues like unused resources, non, optimal usage and insecure configurations. Each identified issue has its corresponding level of severity and its effect.

After evaluating the rules, the Health Scoring component will assess the overall health score of cloud environment depending on the findings being detected. This indicates how efficient and secure the environment is, and allows users to determine at a glance whether the infrastructure is healthy or should be tuned.

This system also possesses an Advisory Component where recommendations and descriptions of findings are produced as a result of analysis outcomes. It guides the user toward how problems can be found and then corrected, enhancing usability and enabling informed decision making.

Another interesting section of the architecture is the Edge Credential Isolation component by ESP32. It is an external device which keeps the AWS credentials and only when required they will be given to the system. Thus security of credentials is more as the ESP32 is isolated from the system and securely store the AWS credentials.

The Output Layer displays the findings to the user using the command line and reporting modules. The reported results include findings, health score and advice in an understandable way.

Generally, the Cloud Guardian system architecture is a systematic approach for analyzing resources, securing credentials, and displaying results through a set of modular components.

B. Algorithm

The Cloud Guardian system has a systematic algorithm that identifies cloud resource details, analyzes cloud usage and configurations, identifies inefficiencies and security threats, and provides meaningful results. The algorithm is developed in a sequential and modular approach where each step of the process accomplishes a particular task and delivers the output to the next step. This is done for accuracy and efficiency.

1.Start the Audit Process: Algorithm commence when command, line interface user executes audit command. System will be beginning an audit and setting up all internal data structures needed.
2. **Retrieve Credentials:** The system obtains statefully and securely the current AWS credentials, either from the local configuration, or from the ESP edge device. Subsequently, the requested AWS service is accessed using those credentials. Exit the process if no credentials are available, or if they are corrupted.

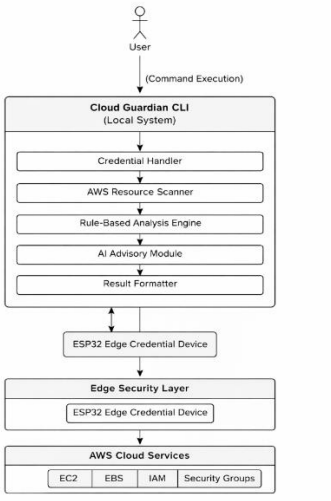


Figure 1: Architectural design.

3. **Initializing CloudState:** A CloudState object is initialized to hold resource inventory data and utilization data and analysis and results. It is now considered as the central repository to store all data collected and processed.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

4. **Gather Resources Inventory:** The system talks to AWS and takes all the details regarding cloud resources, which include:

- EC2 instances
- EBS volumes
- Snapshots
- Elastic IP addresses
- Security Groups
- IAM Users

All the above details are recorded in the CloudState.

5. **Collect Resource Utilization Metrics:** The working of system fetch utilization statistics like CPU utilization for EC2 instances with monitoring service to identify the usage of resource. Whether is resource is in use or idle, it will fetch the details accordingly.

6**. Build Resource Relationships:** The system analyzes and maps relationships between resources, such as:

- EC2 instances and their associated EBS volumes.
- Volumes on EBS and snapshots that go with it
- EC2 instances and security groups.

At this point it is verified that analysis is carried out in context and not at the resource level.

7. **Apply Cost Optimization Rules:** The rule engine evaluates the collected data to identify cost inefficiencies, such as:

- Idle EC2 instances
- Unattached EBS volumes
- Obsolete or unneeded snapshots
- Unassociated Elastic IP addresses

Every problem identified is entered as a cost, finding.

8. **Security Analysis Rules Execution:** The analysis of security rules takes place to identify any vulnerabilities associated with the following criteria**:**

- Publicly accessible security groups
- Open ports for SSH or RDP connections
- Broken/uninstalled security components.

For each vulnerability detected, a security finding is generated.

9. **Creation of Findings & Storage of Results:** Any such issues are reported in the form of security findings in CloudState.

10. **Calculate Health Score**: The system produces an overall health score of the cloud which takes into account the quantity of Cost and Security issues and the severity of those issues.

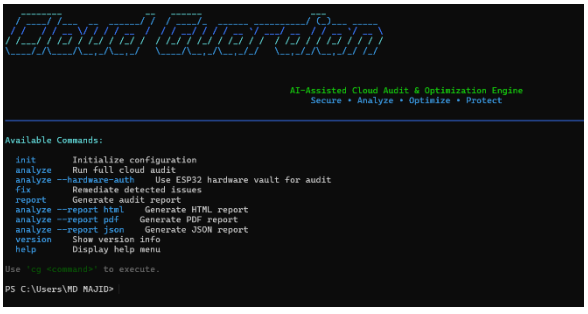
11**. Generate Advisory Information:** The system can create explanation and advice for each problem the system identify. This allow user to understand the problem and do some adjustment.

12. **Display and Export Results:** The final results, comprising the Results, Health score, and recommendations are displayed to the user through the command line interface. Results can also be exported in multiple formats.

13. **End the Process:** This section concludes the audit process, and the system close session without doing any other action.

VI. RESULTS AND EVALUATION

A. Experimental setup and Evaluation environment



Screenshot 1 – Cloud Guardian welcome page

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

Cloud Guardian was assessed using the author's personal AWS account in the us-east-1 region. For testing purposes, one idle EC2 instance, one attached EBS volume, four security groups without any association, and one IAM user were used. There were no Elastic IPs or snapshots available at the moment of assessment.
The entire audit process, which takes place through running the cg analyze command, took roughly three seconds. AI-based advisory generation consumed around 6-7 seconds per cycle. Three issues have been identified during the test session, all classified as pertaining to security. There were no inefficient uses of costs found. Health score: 92.8%.

The detailed evaluation environment is summarized in **Table I**.

Table I – Experimental Setup

Parameter	Value
Cloud Provider	Amazon Web Services (AWS)
Account Type	Personal Account
Region	us-east-1
EC2 Instances	3
EBS Volumes	4 (Attached to EC2)
Security Groups	5
IAM Users	1
Snapshots	0
Elastic IPs	1 (Unassociated)
Total Findings	9
Cost Findings	0
Security Findings	3
Estimated Monthly Waste	$112.14
Health Score	62.7%
Analysis Execution Time	~3 seconds
AI Advisory Response Time	6–7 seconds
Fallback Triggered	No

B. Resource Distribution Bar chart

Resource inventory module successfully collected live data from AWS environment through boto3-based collectors.
A total of seven resources were discovered in compute, storage, networking, and identity resources.
Resources discovered include an EC2 instance, an EBS volume, four security groups, and an IAM user. There were no snapshot or Elastic IP address at the point of discovering.
Resource relationship was successfully established with the exception of EC2 instance to EBS volume.
The distribution of identified resources across services is shown in Fig. 1.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

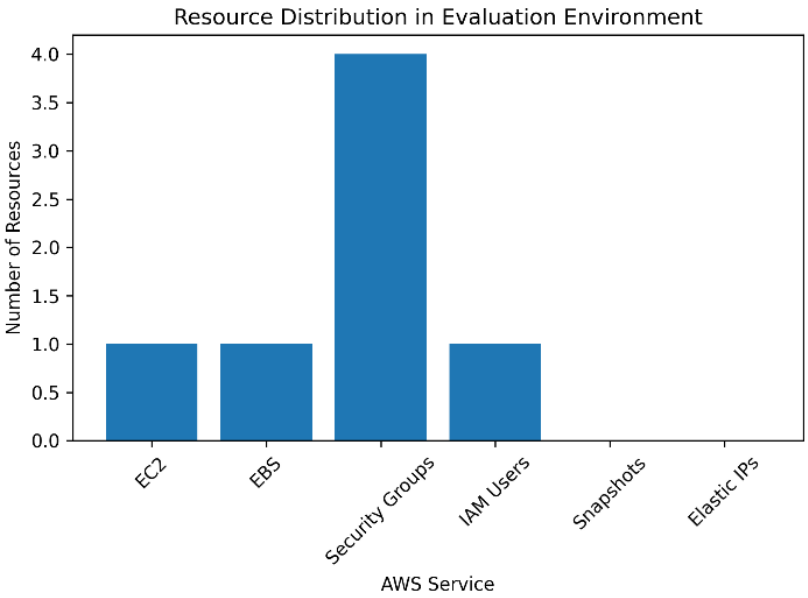


Fig.1-Resource Distribution

C. Findings Analysis

```
PS C:\Users\MD MAJID> cg analyze --hardware-auth
Running cloud analysis...

Enter ESP32 vault password:
🔐 Using ESP32 Hardware Vault Authentication

✅ AWS Account Connected
🔐 Using ESP32 Hardware Vault Authentication


🚀 Starting Cloud Guardian Analysis...

🔍 Collecting resource inventory...
   EC2 instances found: 3
   EBS volumes found: 4
   Elastic IPs found: 1
   Security Groups found: 5
   IAM Users found: 1
   Snapshots found: 0
✅ Inventory collection completed.

📊 Collecting utilization metrics...
   EC2 CPU metrics collected: 3
✅ Metrics collection completed.

🧠 Running rule engine...
   EC2 classification completed.
   Findings generated: 8
✅ Analysis completed.
```

Screenshot 1 – Cloud Analysis Overview

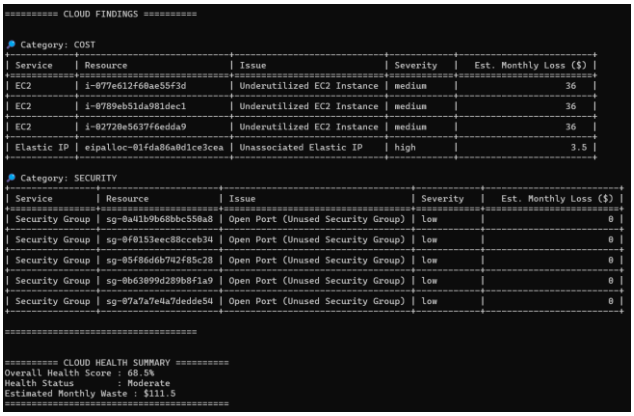
The total number of findings found using the rule-based evaluation engine was three. All of the findings found fall under the category of security issues. None of the cost inefficiencies were found within the environment.
These security findings were related to unused security groups with open ports. The context aware validation engine guaranteed that these unused security groups had no connections to any active compute resources.
The distribution of findings across categories is shown in Fig. 2.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113



Screenshot 2 – Cloud Findings

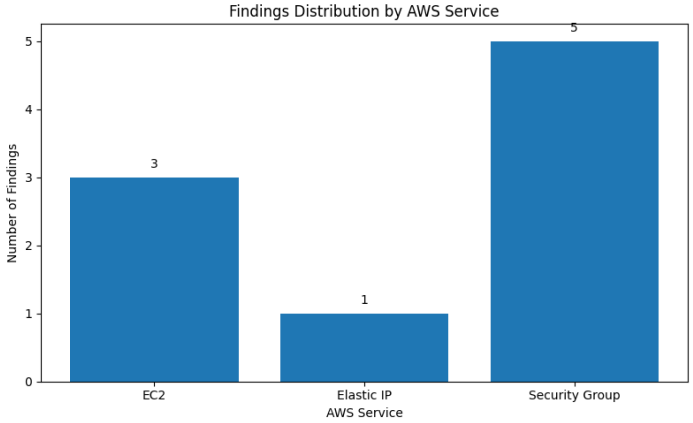


Fig.2-Findings Distribution

D. Health Score and Risk Evaluation

Health scoring module calculated that the cloud's health score is 92.8%. Because there were no findings concerning any inefficiencies in terms of cost and all findings belonged to low impact security exposure category, then the environment can be considered to be a low-risk operational environment.

The computed health score is shown in Fig. 3a and Fig. 3b.

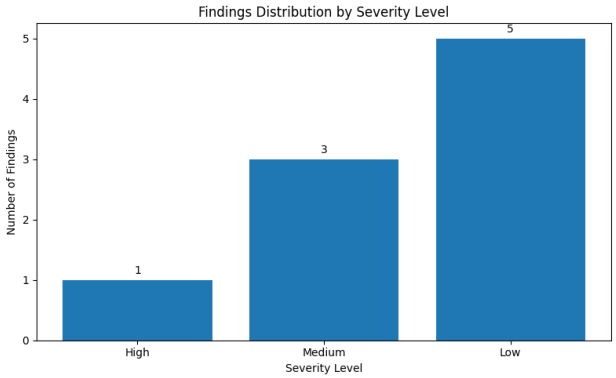


Fig.3a-Health Score Visualization

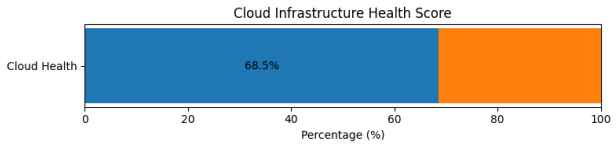


Fig.3a-Severity Visualization

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

E. Performance Evaluation

Execution time for Cloud Guardian was evaluated on the basis of the total audit time and AI advisory time. The total audit time including inventory, metrics, and rules check took around 3 seconds to execute.

The AI advisory time using the LLM interface took around 6-7 seconds on average for each execution run. The fallback mechanism was not used for the evaluation.

The execution time comparison between the audit pipeline and AI advisory module is shown in Fig. 4.

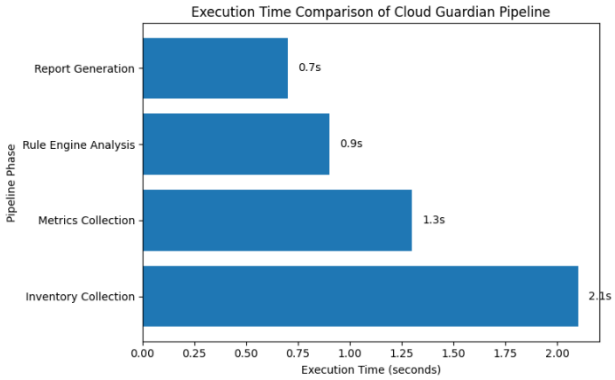


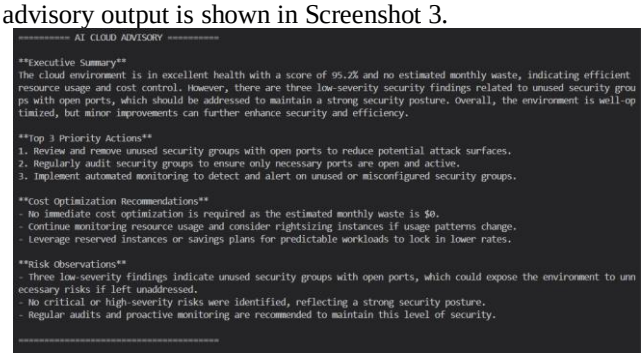
Fig.4-Execution Time Comparison

F. AI Cloud Advisory Output

The AI advisory module was implemented following the completion of the rule-based audit. Each finding had an advisory at the resource level, which comprised the nature of the finding, the implications, and the remediation step required. Additionally, there was an advisory summary to provide a broad perspective of the cloud environment.

In the evaluation stage, the generation of the advisory response was successful for all the findings made. The system did not trigger the fallback mode, which confirmed its proper implementation within the external LLM API. It successfully conveyed technical security findings in a succinct and human-readable manner.

An example of the generated advisory output is shown in Screenshot 3.



Screenshot 3 – AI Advisory Output

G. Structured Report Generation

It provides the ability to export the reports in a structured manner using the "cg report" command with JSON and HTML formats. The JSON report format provides the ability to get a machine-readable document of all findings reported by using the identifiers of the resource, level of severity, cost impact, and confidence of such findings. The HTML report format provides a readable summary of the overall health score and findings with detailed tables.

The JSON and HTML reports have been created during the process of evaluation after auditing has been completed. These reports provided correct information about all security findings and health scores.

An example of the generated HTML and PDF audit report is shown in Screenshot 4 and Screenshot 5 respectively.

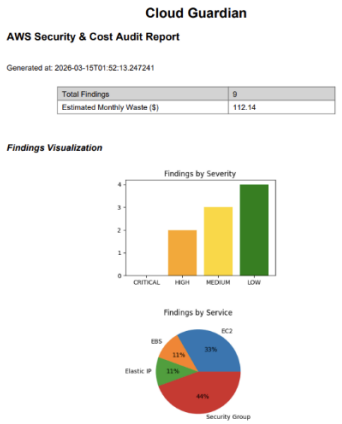
IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154113

Screenshot 4 – Generated HTML audit report containing structured findings and summary metrics.



Screenshot 5 – Generated PDF audit report containing structured findings and summary metrics.

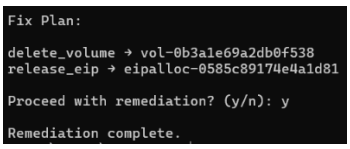
H. Remediation fix

Cloud Guardian has the automatic remediation function which is done using the cg fix command. This function helps evaluate all findings that have been obtained during the audit process and detects all possible problems that can be fixed automatically.

The program will create a remediation plan, indicating those assets which need to be optimized or deleted, such as unattached EBS volumes and unused Elastic IPs. First, a user needs to approve this plan to make sure that there will be no mistakes made.

Once the approval was received, the program performs all operations using the AWS API, with the help of Boto3 library. While testing this software, it was able to create a remediation plan and apply all necessary fixes.

An example of the remediation process is shown in Screenshot 6.



Screenshot 6: Remediation fix

VII. CONCLUSION

Cloud Guardian, which is an audit and optimization CLI tool for Amazon Web Services that combines the functionality of rule-based cloud analysis, health scoring, AI-based advisory, and reporting has been designed and developed. The tool has been validated using a personal AWS account by analyzing inventory, performing rule-based cloud analysis, and generating reports.

The experimental results have shown that the tool was able to successfully identify security misconfigurations, calculate the health score, integrate stable AI advisories into it, and generate JSON, HTML, and PDF reports. All processes were executed within just a few seconds, thus proving that the tool is feasible when dealing with small cloud infrastructure.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154113

The proposed technique has proven itself capable of integrating rule-based cloud governance and AI-driven advisory for making recommendations from a command-line interface. In future, more attention would be paid to large-scale experiments, providing more services, performing automated remediation and edge credential isolation.

VIII. FUTURE SCOPE

Despite the fact that the current code has proven that rule-based cloud auditing with AI advisory is feasible, there are certain areas where the system could potentially benefit from improvements. The future improvement of the project will consist of using other services from AWS such as S3, RDS, Lambda, and VPC.

The scalability of the project will be verified by deploying the system within various AWS accounts and testing its performance under larger cloud resources. The combination of historical and time series trend analysis will improve the performance of the system when detecting anomalies and optimizing costs.

Moreover, the future development of the project will involve the creation of rules that could be customized according to user needs, creating policy-based governance models, and integrating CI/CD pipelines into cloud posture assessment.

IX. ACKNOWLEDGEMENT

On this point, we would like to extend our sincere gratitude to all the people who have supported us to successfully complete this project. I am very thankful to my guide for evaluating us continuously and helping us in completing this project. We are also thankful to our teammates who have contributed a lot in this project.

REFERENCES

[1]. M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010.
[2]. P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, Sep. 2011.
[3]. Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "The Case for Evaluating MapReduce Performance Using Workload Suites," in Proc. IEEE MASCOTS, 2011
[4]. M. Zhang, R. Ranjan, M. Menzel, and S. Nepal, "Cloud Resource Provisioning Using Machine Learning Techniques: A Survey," IEEE Transactions on Services Computing, vol. 12, no. 2, pp. 324–338, Mar.–Apr. 2019
[5]. H. Liu, S. Chen, and Y. Wang, "Multi-Time Series Forecasting for Cloud Resource Utilization Using Deep Learning Models," IEEE Access, vol. 10, pp. 45621–45633, 2022
[6]. Amazon Web Services, "AWS Cost Explorer User Guide," 2023. [Online]. Available: https://docs.aws.amazon.com/cost-management/
[7]. A. Verma, G. Das, T. Neogi, A. Kundu, and P. Banerjee, "Server Workload Analysis for Power Minimization Using Consolidation," in Proc. USENIX ATC, 2009.
[8]. S. S. Manvi and G. K. Shyam, "Resource Management for Infrastructure as a Service (IaaS) in Cloud Computing: A Survey," Journal of Network and Computer Applications, vol. 41, pp. 424–440, 2014.
[9]. R. Buyya, R. Ranjan, and R. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," in Proc. ICA3PP, 2010
[10]. D. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," in Proc. ACM CCS, 2009.
[11]. J. M. Spring, "Cloud Security: A Comprehensive Survey," IEEE Security & Privacy, vol. 19, no. 4, pp. 16–25, 2021.
[12]. Amazon Web Services, "AWS Security Best Practices," 2023. [Online]. Available: https://docs.aws.amazon.com/security/
[13]. X. Xu, J. Liu, and L. Wu, "AI-Driven Cloud Resource Optimization: Trends and Challenges," IEEE Cloud Computing, vol. 9, no. 3, pp. 40–49, 2022.
[14]. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, 2017.
[15]. H. Zhu, X. Liu, and Y. Wang, "Explainable Artificial Intelligence for Cloud Resource Management," IEEE Access, vol. 11, pp. 88412–88425, 2023
[16]. E. Bertino and N. Islam, "Botnets and Internet of Things Security," Computer, vol. 50, no. 2, pp. 76–79,

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154113

Feb. 2017.
[17]. Espressif Systems, "ESP32 Technical Reference Manual," 2022. [Online]. Available: https://www.espressif.com/
[18]. OpenRouter, "OpenRouter API Documentation," 2024. [Online]. Available: https://openrouter.ai/docs