Intelligent Cloud Infrastructure Review with GenAI Assistance
(AWS Cloud Security & Optimization Project)

Name: SUMITHA A

Reg. No.: 727823TUCY048

1. Executive Summary

Modern cloud environments generate a massive volume of infrastructure telemetry, configuration data, and operational logs. Manually analyzing this data to identify security risks, inefficiencies, and misconfigurations is time-consuming and error-prone. This project demonstrates an Intelligent Cloud Infrastructure Review using Amazon Web Services (AWS) combined with Generative AI (GenAI) in an advisory, human-in-the-loop role.

The objective of this project is to design, deploy, monitor, and review a containerized cloud environment while leveraging GenAI to:

- Summarize infrastructure behavior
- Identify security risks and misconfigurations
- Suggest optimization and hardening strategies

The environment consists of an Amazon EC2 instance running Docker, hosting an Nginx web application, deployed inside a custom Amazon VPC. Amazon CloudWatch Logs is used for centralized logging of system and container logs, which are later reviewed using GenAI for insights.

Importantly, GenAI is used only for analysis and decision support. It has no direct access to the AWS environment, ensuring security and compliance. This approach aligns with enterprise best practices and the AWS Well-Architected Framework.

2. Problem Statement

2.1 Background

Traditional cloud monitoring and observability tools primarily focus on collecting metrics, events, and raw log data from cloud resources. While these tools are effective at providing visibility, they often lack the contextual intelligence required to interpret the data and derive meaningful insights. As cloud environments grow in complexity, manually correlating logs and configurations across multiple services becomes increasingly difficult.

IT leadership and cloud administrators require smarter, high-level insights into key areas such as:

- Infrastructure usage and resource efficiency
- Security exposure and access control weaknesses
- Operational inefficiencies affecting performance and cost
- Misconfigurations across compute, networking, storage, and containerized workloads

Although Artificial Intelligence can assist in analyzing this data, fully automated AI systems with direct write access to production infrastructure introduce significant security, compliance, and governance risks. Therefore, a controlled, advisory approach is necessary, where AI supports decision-making while critical actions remain under human oversight.

2.2 Problem Definition

The core challenge addressed in this project is the design of a GenAI-assisted cloud infrastructure review system that enhances visibility and decision-making without compromising security or control. The system must be capable of analyzing infrastructure logs and configuration data collected from cloud resources to extract meaningful operational and security insights.

Specifically, the solution should be able to:

- Analyze system and application logs along with configuration metadata
- Identify risky patterns, security gaps, and misconfigurations
- Suggest actionable improvements and optimization measures
- Maintain full human control by ensuring GenAI operates strictly in an advisory role

This approach balances the analytical strengths of Generative AI with the need for human oversight, making it suitable for real-world enterprise cloud environments.

2.3 Scope of the Project

The scope of this project is limited to the analysis and review of a controlled AWS-based cloud environment using GenAI as a decision-support tool. The project focuses on evaluating infrastructure behavior, security posture, and operational efficiency based on logs and configuration data.

The key areas covered within the scope include:

- Reviewing AWS infrastructure logs and configuration settings related to compute, networking, and containers
- Identifying potential security risks, misconfigurations, and operational inefficiencies
- Utilizing GenAI strictly for advisory analysis, insight generation, and recommendation purposes
- Emphasizing documentation, analysis, and reporting rather than automated remediation
- Ensuring that no real production workloads, sensitive information, or confidential data are used

By clearly defining these boundaries, the project maintains security, compliance, and ethical use of AI while effectively demonstrating GenAI-assisted cloud infrastructure review capabilities.

3. Objectives of the Project

The primary objective of this project is to demonstrate how Generative AI can be effectively integrated into cloud infrastructure reviews while maintaining security and human oversight. To achieve this, the project is guided by the following specific objectives:

1. To design and implement a secure AWS cloud architecture that follows industry best practices
2. To deploy a containerized web application using Docker on an Amazon EC2 instance
3. To enable centralized logging and monitoring using Amazon CloudWatch Logs
4. To review infrastructure behavior and log data using GenAI for analytical insights
5. To identify security misconfigurations and operational risks within the environment
6. To propose security hardening and optimization recommendations based on the review
7. To document the complete implementation, analysis, and findings with supporting evidence

These objectives ensure a structured approach to building, analyzing, and documenting a GenAI-assisted cloud infrastructure review.

4. High-Level Architecture

4.1 Architecture Overview

The cloud environment is designed using a modular and isolated architecture, ensuring clear separation between compute, networking, storage, and monitoring components. This approach improves security, scalability, and manageability by allowing each layer of the infrastructure to be independently monitored and controlled. The architecture follows AWS best practices and supports effective logging and analysis for GenAI-assisted reviews.

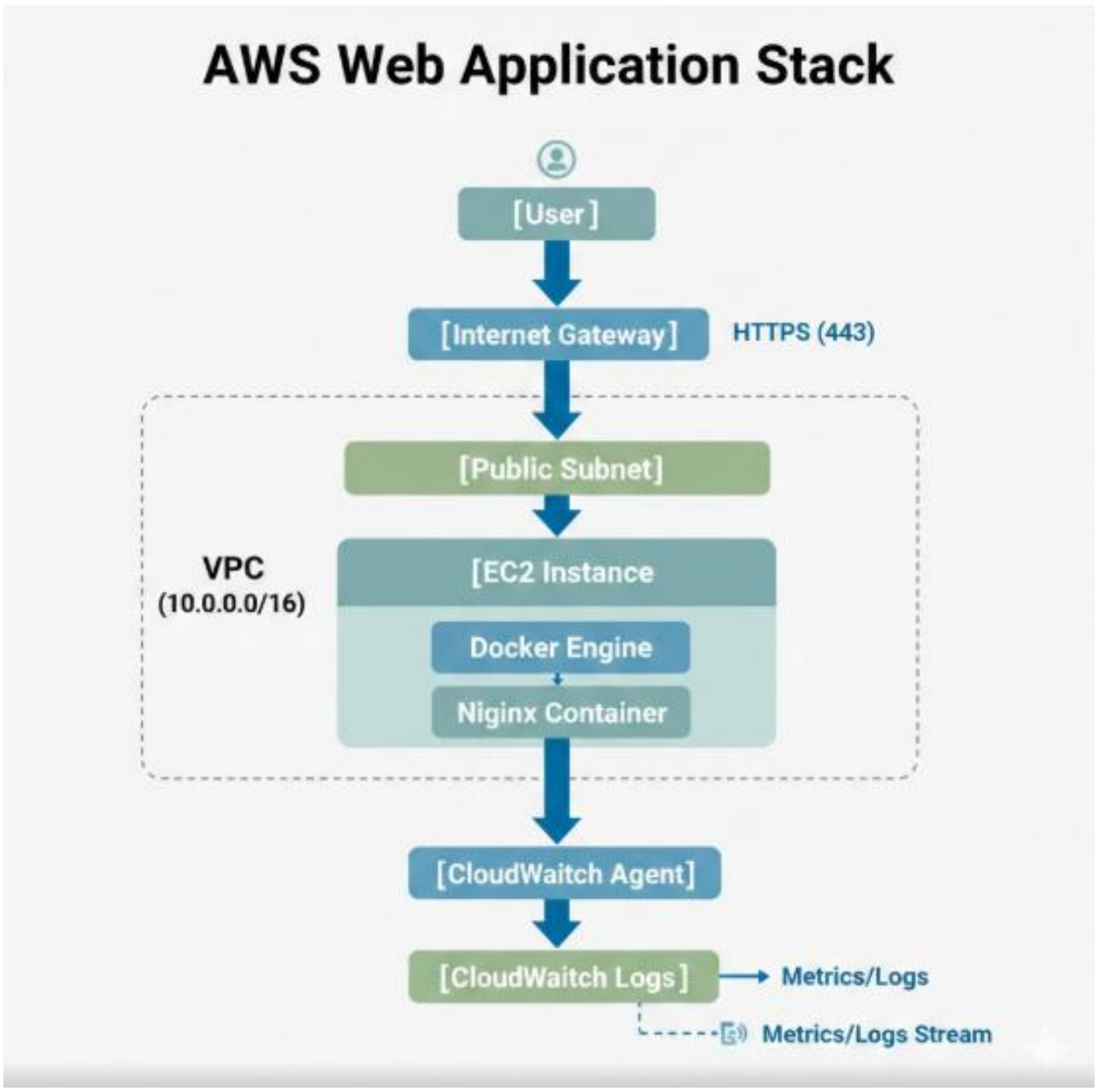
4.2 Architecture Components

The major components of the architecture are described below:

- **Amazon Virtual Private Cloud (VPC)**
  - CIDR Block: 10.0.0.0/16
  - Provides an isolated virtual network for hosting cloud resources
- **Public Subnet**
  - Hosts the EC2 instance running the containerized application
  - Enables controlled access to the application through defined security rules
- **Amazon EC2 (t2.micro)**
  - Acts as the primary compute resource
  - Runs the Docker Engine and hosts the Nginx container
- **Docker (Container Runtime)**
  - Provides a lightweight and portable environment for running the Nginx web server
  - Simplifies application deployment and management
- **Amazon CloudWatch Logs**
  - Collects and centralizes system and container logs

    - Enables monitoring, troubleshooting, and analysis
- **Amazon S3 (Optional)**
    - Used for long-term archival of logs and audit data
    - Supports cost-effective storage and compliance requirements

4.3 Architecture Diagram

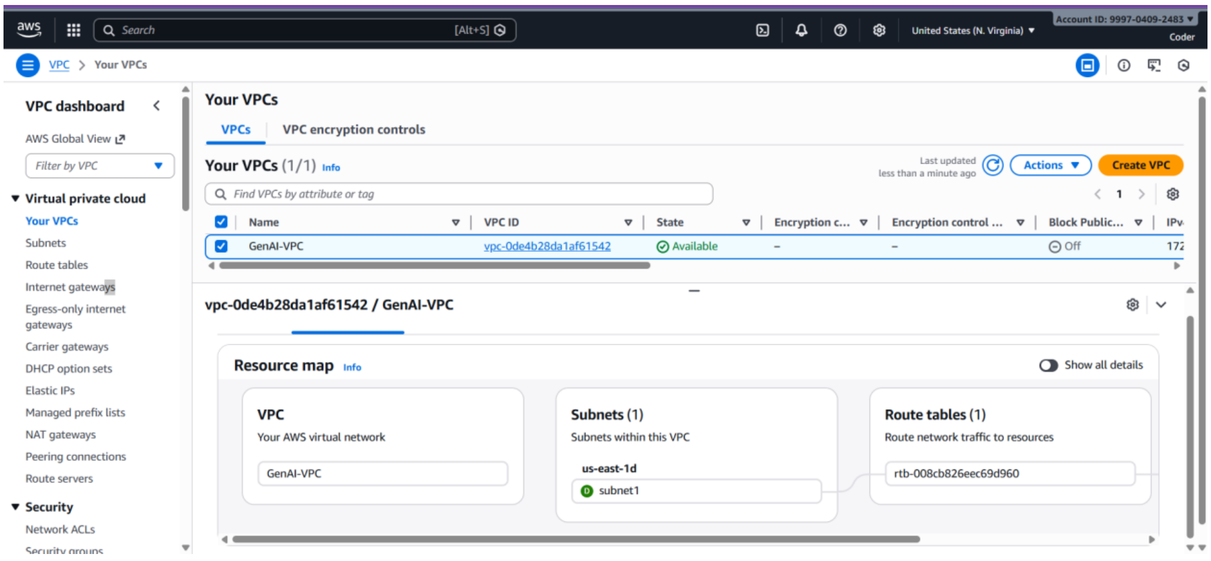


5. Infrastructure Implementation

5.1 VPC and Network Configuration

A custom Amazon VPC was created to provide network isolation for the application environment. Public subnets, route tables, and an Internet Gateway were configured to enable controlled external access. Security Groups acted as virtual firewalls, allowing only required inbound and outbound traffic.

VPC Resource Map:

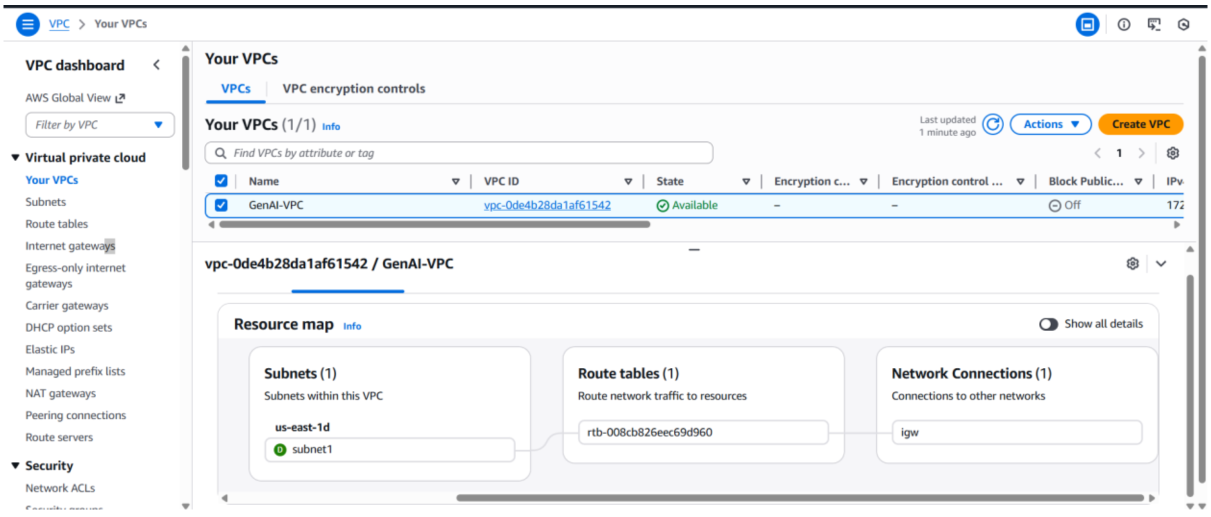


	Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv
✓	GenAI-VPC	vpc-0de4b28da1af61542	Available	–	–	Off	172

vpc-0de4b28da1af61542 / GenAI-VPC

Resource map

VPC	Subnets (1)	Route tables (1)
Your AWS virtual network	Subnets within this VPC	Route network traffic to resources
GenAI-VPC	us-east-1d: subnet1	rtb-008cb826eec69d960



	Name	VPC ID	State	Encryption c...	Encryption control ...	Block Public...	IPv
✓	GenAI-VPC	vpc-0de4b28da1af61542	Available	–	–	Off	172

vpc-0de4b28da1af61542 / GenAI-VPC

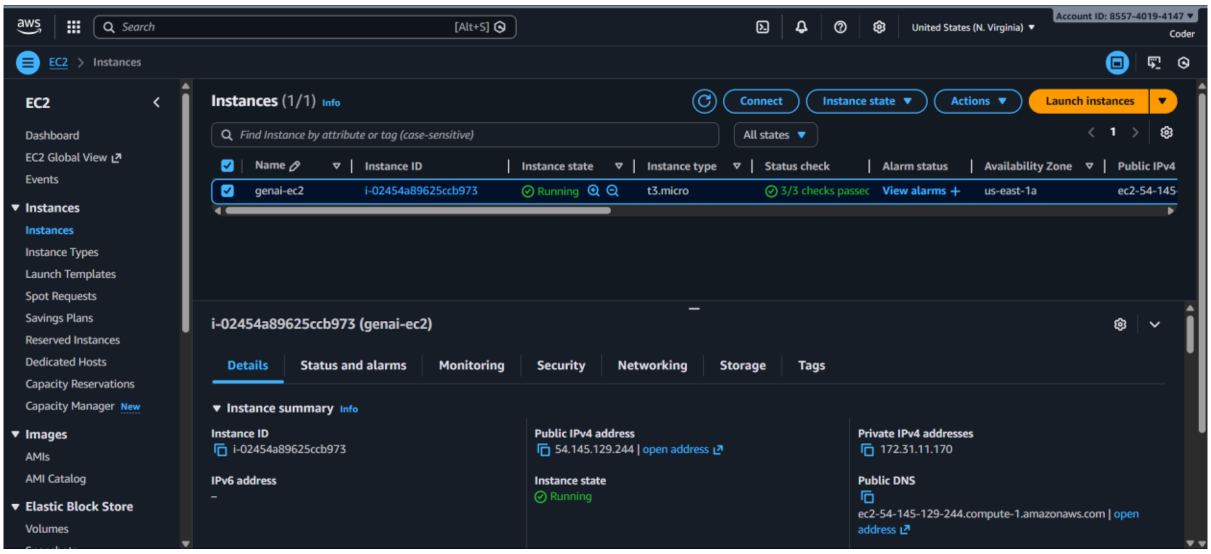
Resource map

Subnets (1)	Route tables (1)	Network Connections (1)
Subnets within this VPC	Route network traffic to resources	Connections to other networks
us-east-1d: subnet1	rtb-008cb826eec69d960	igw

5.2 EC2 Instance Setup

An Amazon EC2 instance running Amazon Linux 2023 was launched within the public subnet. Docker was installed using user-data scripts to ensure consistency and repeatability across deployments.

EC2 Instance Summary:


Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
genai-ec2	i-02454a89625ccb973	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-54-145-

i-02454a89625ccb973 (genai-ec2)

Instance summary

Instance ID: i-02454a89625ccb973

Public IPv4 address: 54.145.129.244 | open address

Private IPv4 addresses: 172.31.11.170

IPv6 address: –

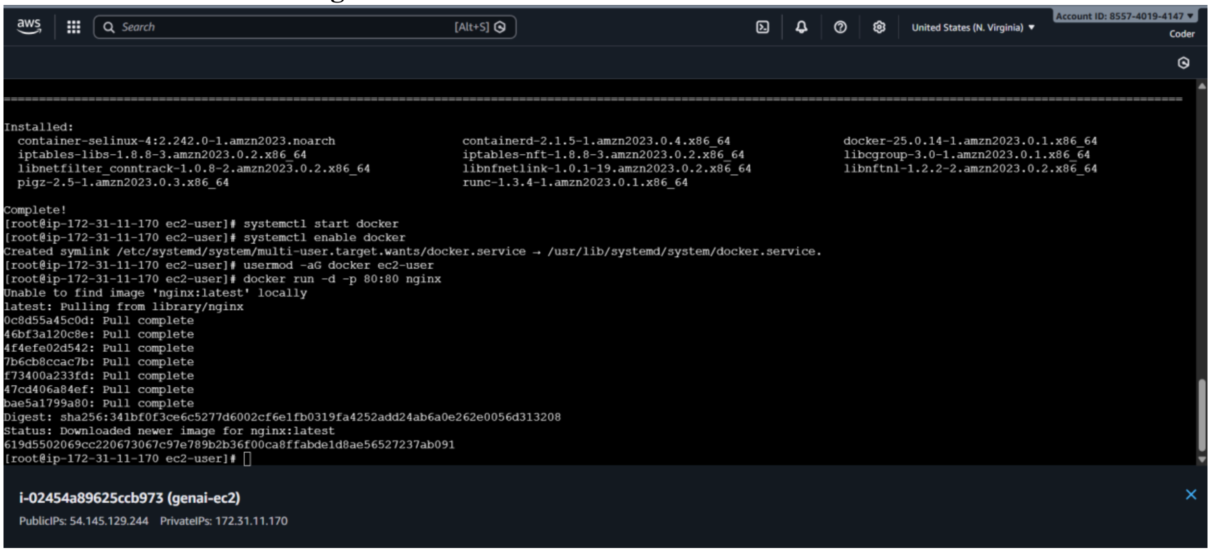
Instance state: Running

Public DNS: ec2-54-145-129-244.compute-1.amazonaws.com | open address

5.3 Docker and Application Deployment

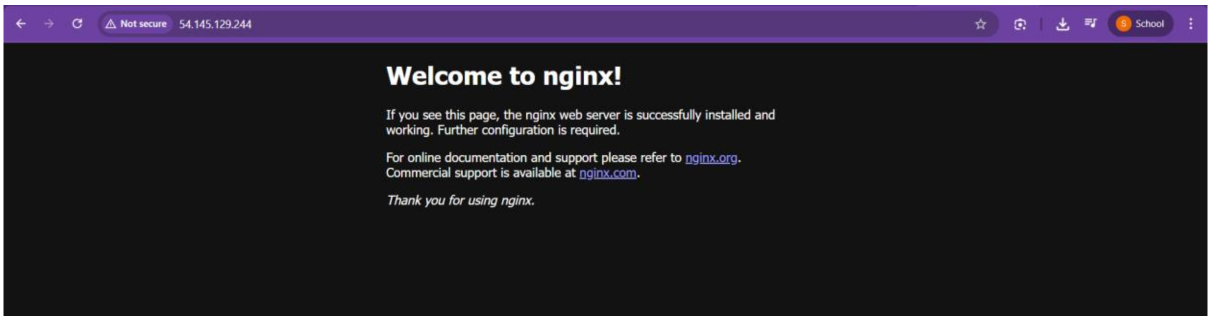
Docker was configured on the EC2 instance to run an Nginx container. The container exposes port 80, enabling HTTP access to the web application and validating successful container deployment.

Docker Container Running:


```
================================================================================================================================

Installed:
  container-selinux-4:2.242.0-1.amzn2023.noarch               containerd-2.1.5-1.amzn2023.0.4.x86_64           docker-25.0.14-1.amzn2023.0.1.x86_64
  iptables-libs-1.8.8-3.amzn2023.0.2.x86_64                   iptables-nft-1.8.8-3.amzn2023.0.2.x86_64         libcgroup-3.0-1.amzn2023.0.1.x86_64
  libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64          libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64        libnftnl-1.2.2-2.amzn2023.0.2.x86_64
  pigz-2.5-1.amzn2023.0.3.x86_64                              runc-1.3.4-1.amzn2023.0.1.x86_64

Complete!
[root@ip-172-31-11-170 ec2-user]# systemctl start docker
[root@ip-172-31-11-170 ec2-user]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-172-31-11-170 ec2-user]# usermod -aG docker ec2-user
[root@ip-172-31-11-170 ec2-user]# docker run -d -p 80:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
0c8d55a45c0d: Pull complete
46bf3a120c8e: Pull complete
4f4efe02d542: Pull complete
7b6cb8ccac7b: Pull complete
f73400a233fd: Pull complete
47cd406a84ef: Pull complete
bae5a1799a80: Pull complete
Digest: sha256:341bf0f3ce6c5277d6002cf6e1fb0319fa4252add24ab6a0e262e0056d313208
Status: Downloaded newer image for nginx:latest
619d5502069cc220673067c97e789b2b36f00ca8ffabde1d8ae56527237ab091
[root@ip-172-31-11-170 ec2-user]# 
```

i-02454a89625ccb973 (genai-ec2)

PublicIPs: 54.145.129.244   PrivateIPs: 172.31.11.170


Not secure 54.145.129.244

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

6. Logging and Monitoring Configuration

6.1 Need for Centralized Logging

In cloud-native environments, logs are generated across multiple layers such as operating systems, applications, containers, and network components. Centralized logging consolidates these logs into a single platform, making analysis and monitoring more efficient.

Centralized logging enables:

- Faster troubleshooting by correlating events across services
- Improved security auditing and forensic analysis
- Easier compliance reporting and log retention
- Effective AI-assisted analysis by providing structured and complete data

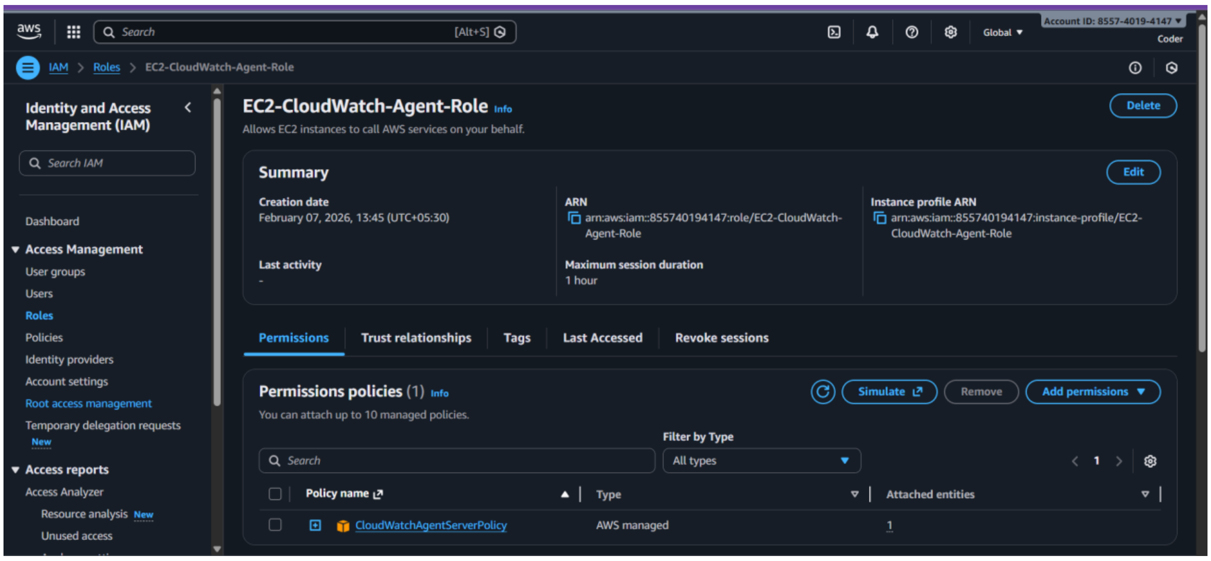
Amazon CloudWatch Logs was selected as it integrates seamlessly with AWS services and supports scalable log ingestion and analysis.

6.2 CloudWatch Agent Configuration

The Amazon CloudWatch Agent was installed and configured on the EC2 instance to collect system-level and application logs. Since Amazon Linux 2023 uses journald by default, rsyslog was enabled to forward logs into /var/log/messages, ensuring compatibility with the CloudWatch Agent configuration.

The agent configuration defined log file paths, log group names, and stream naming conventions based on the EC2 instance ID. This ensured logs were reliably collected and easily traceable to the source instance.

CloudWatch Agent Running:



EC2-CloudWatch-Agent-Role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date: February 07, 2026, 13:45 (UTC+05:30)

ARN: arn:aws:iam::855740194147:role/EC2-CloudWatch-Agent-Role

Instance profile ARN: arn:aws:iam::855740194147:instance-profile/EC2-CloudWatch-Agent-Role

Last activity: -

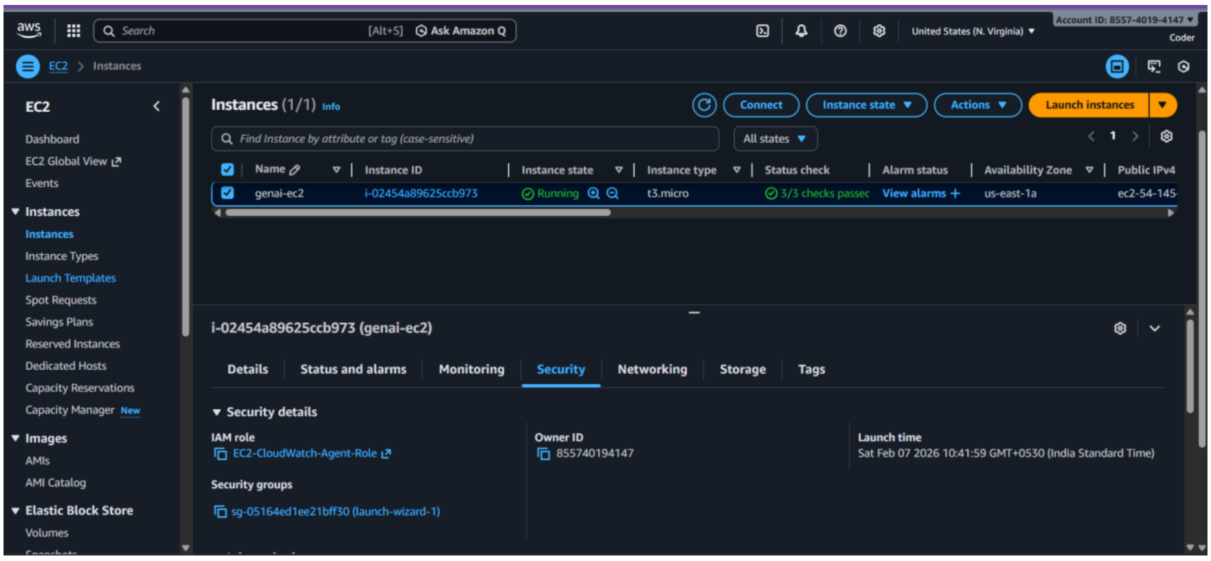
Maximum session duration: 1 hour

Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
CloudWatchAgentServerPolicy	AWS managed	1



Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
genai-ec2	i-02454a89625ccb973	Running	t3.micro	3/3 checks passec	View alarms +	us-east-1a	ec2-54-145-

i-02454a89625ccb973 (genai-ec2)

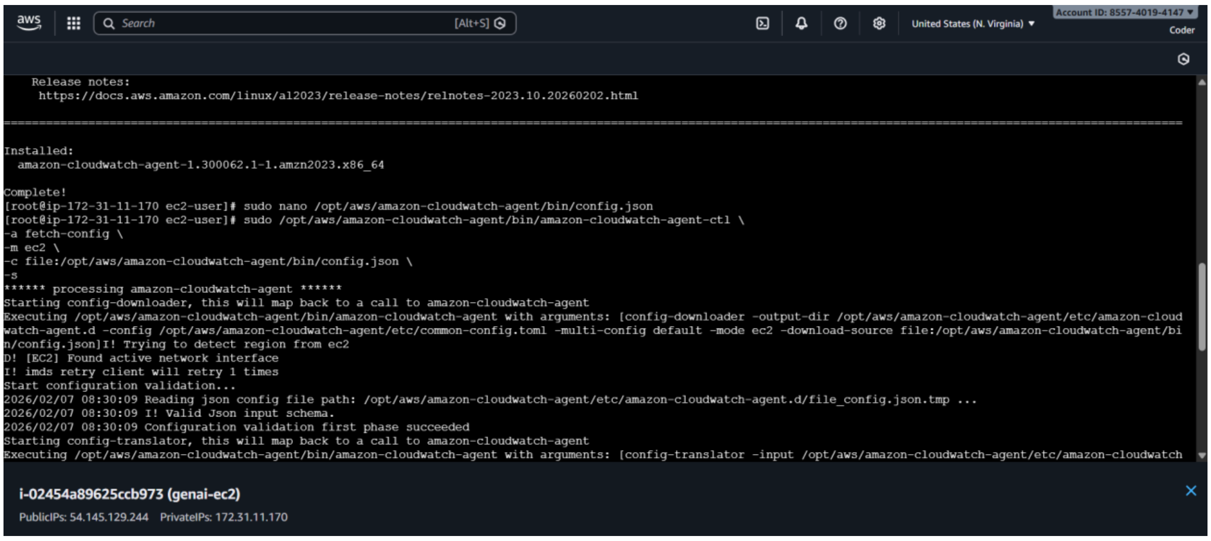
Security details

IAM role: EC2-CloudWatch-Agent-Role

Owner ID: 855740194147

Launch time: Sat Feb 07 2026 10:41:59 GMT+0530 (India Standard Time)

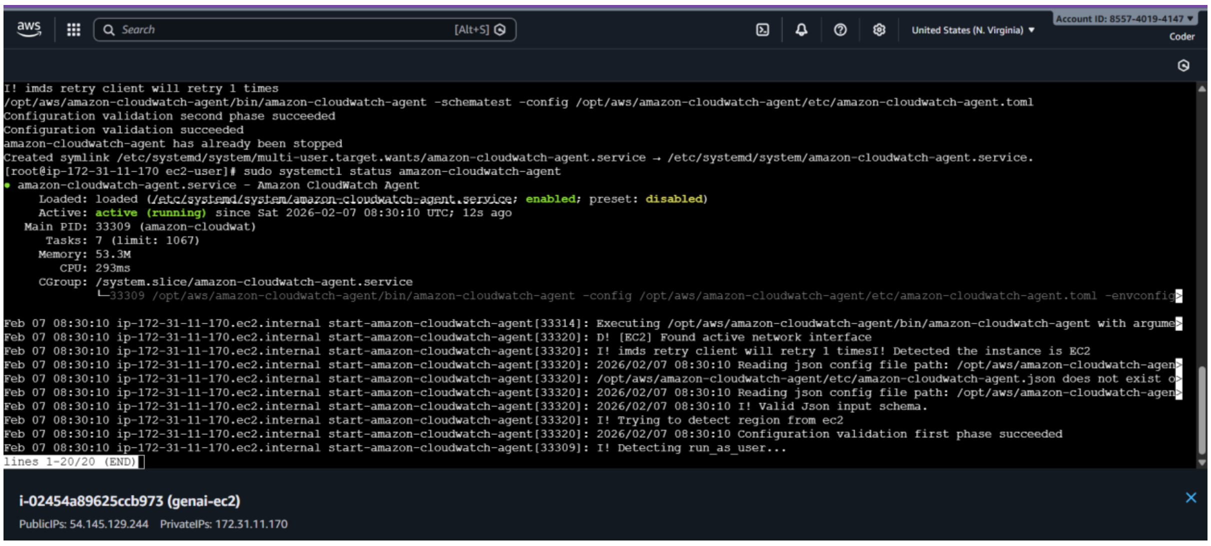
Security groups: sg-05164ed1ee21bff30 (launch-wizard-1)



```
    Release notes:
     https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.10.20260202.html

================================================================================================================================================================

Installed:
  amazon-cloudwatch-agent-1.300062.1-1.amzn2023.x86_64

Complete!
[root@ip-172-31-11-170 ec2-user]# sudo nano /opt/aws/amazon-cloudwatch-agent/bin/config.json
[root@ip-172-31-11-170 ec2-user]# sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \
-a fetch-config \
-m ec2 \
-c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json \
-s
****** processing amazon-cloudwatch-agent ******
Starting config-downloader, this will map back to a call to amazon-cloudwatch-agent
Executing /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent with arguments: [config-downloader -output-dir /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloud
watch-agent.d -config /opt/aws/amazon-cloudwatch-agent/etc/common-config.toml -multi-config default -mode ec2 -download-source file:/opt/aws/amazon-cloudwatch-agent/bi
n/config.json]I! Trying to detect region from ec2
D! [EC2] Found active network interface
I! imds retry client will retry 1 times
Start configuration validation...
2026/02/07 08:30:09 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp ...
2026/02/07 08:30:09 I! Valid Json input schema.
2026/02/07 08:30:09 Configuration validation first phase succeeded
Starting config-translator, this will map back to a call to amazon-cloudwatch-agent
Executing /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent with arguments: [config-translator -input /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch
```

i-02454a89625ccb973 (genai-ec2)

PublicIPs: 54.145.129.244 PrivateIPs: 172.31.11.170



```
I! imds retry client will retry 1 times
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -schematest -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml
Configuration validation second phase succeeded
Configuration validation succeeded
amazon-cloudwatch-agent has already been stopped
Created symlink /etc/systemd/system/multi-user.target.wants/amazon-cloudwatch-agent.service → /etc/systemd/system/amazon-cloudwatch-agent.service.
[root@ip-172-31-11-170 ec2-user]# sudo systemctl status amazon-cloudwatch-agent
● amazon-cloudwatch-agent.service - Amazon CloudWatch Agent
     Loaded: loaded (/etc/systemd/system/amazon-cloudwatch-agent.service; enabled; preset: disabled)
     Active: active (running) since Sat 2026-02-07 08:30:10 UTC; 12s ago
   Main PID: 33309 (amazon-cloudwat)
      Tasks: 7 (limit: 1067)
     Memory: 53.3M
        CPU: 293ms
     CGroup: /system.slice/amazon-cloudwatch-agent.service
             └─33309 /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml -envconfig>

Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33314]: Executing /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent with argume>
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: D! [EC2] Found active network interface
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: I! imds retry client will retry 1 timesI! Detected the instance is EC2
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: 2026/02/07 08:30:10 Reading json config file path: /opt/aws/amazon-cloudwatch-agen>
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json does not exist o>
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: 2026/02/07 08:30:10 Reading json config file path: /opt/aws/amazon-cloudwatch-agen>
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: 2026/02/07 08:30:10 I! Valid Json input schema.
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: I! Trying to detect region from ec2
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33320]: 2026/02/07 08:30:10 Configuration validation first phase succeeded
Feb 07 08:30:10 ip-172-31-11-170.ec2.internal start-amazon-cloudwatch-agent[33309]: I! Detecting run_as_user...
lines 1-20/20 (END)
```

i-02454a89625ccb973 (genai-ec2)

PublicIPs: 54.145.129.244 PrivateIPs: 172.31.11.170

6.4 Amazon S3 Log Archival Configuration

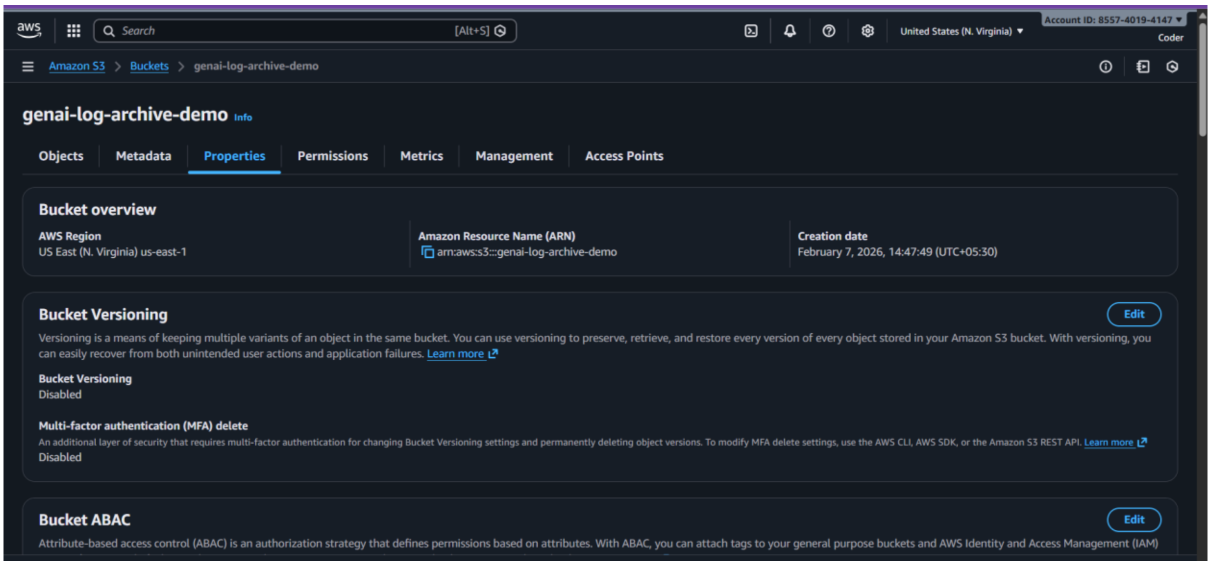
To support long-term log retention and cost-effective storage, an Amazon S3 bucket was created for archiving CloudWatch Logs. While CloudWatch is suitable for real-time monitoring and short-term analysis, S3 provides durable, scalable, and low-cost storage for historical data.

The S3 bucket was configured with:

- Private access enabled by default
- Server-side encryption (SSE-S3) for data at rest
- Block Public Access enabled to prevent accidental exposure

CloudWatch Logs can be exported periodically to the S3 bucket for compliance audits, forensic investigations, and offline analysis using analytics tools such as Amazon Athena.

Amazon S3 Bucket Configuration:



genai-log-archive-demo

Bucket overview

AWS Region
US East (N. Virginia) us-east-1

Amazon Resource Name (ARN)
arn:aws:s3:::genai-log-archive-demo

Creation date
February 7, 2026, 14:47:49 (UTC+05:30)

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Disabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled

Bucket ABAC

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. With ABAC, you can attach tags to your general purpose buckets and AWS Identity and Access Management (IAM)

6.5 Benefits of Using Amazon S3 for Log Storage

Using Amazon S3 for log archival offers several advantages:

- Long-term retention of logs beyond CloudWatch limits
- Reduced operational cost for older log data
- High durability and availability
- Easy integration with analytics and AI-based review tools

This approach ensures that critical infrastructure and security logs remain accessible for future audits and GenAI-assisted analysis without impacting real-time system performance.

6.3 Log Groups Created

Two primary CloudWatch Log Groups were created for structured monitoring:

- **/ec2/system/logs** → Captures operating system and service logs
- **/ec2/docker/logs** → Captures Docker container logs

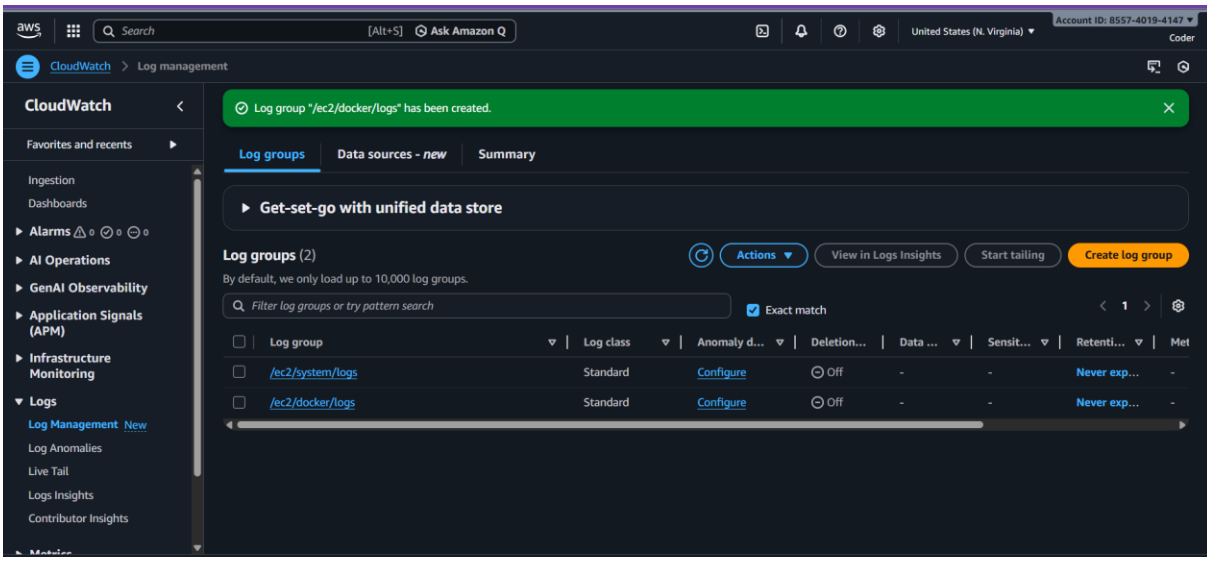
Log streams were automatically generated using the EC2 instance ID, enabling clear identification of log sources and simplifying multi-instance scalability.

7. Log Verification and Evidence

7.1 System Logs

System logs from /var/log/messages were successfully ingested into Amazon CloudWatch Logs. These logs include system boot events, service status updates, and security-related messages. The availability of these logs confirms correct agent configuration and network connectivity.

System Log Stream:

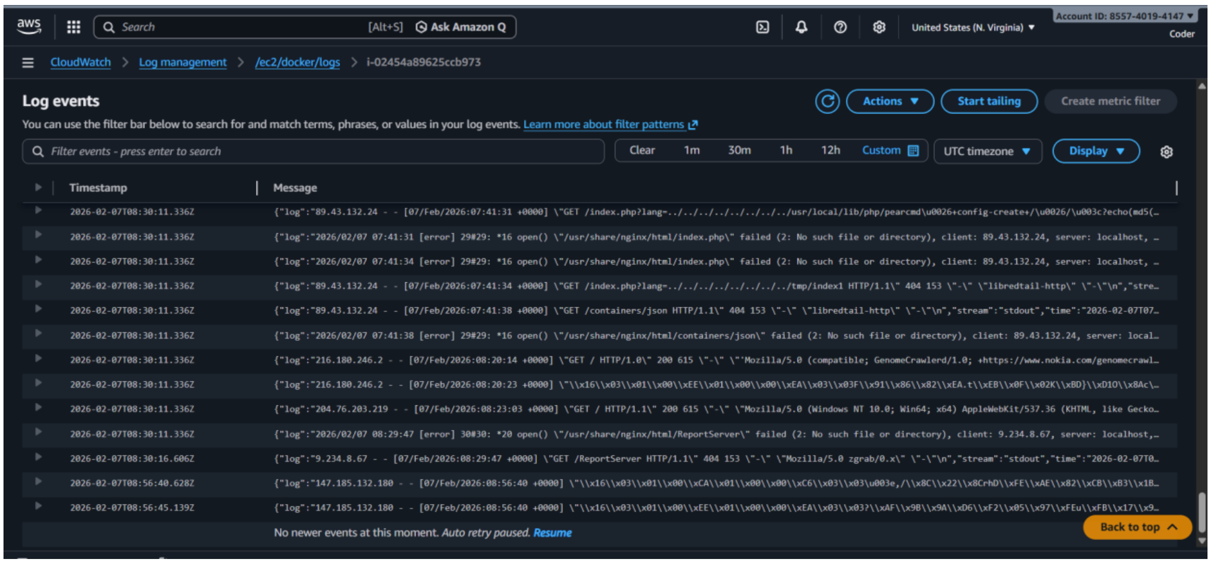


Log group "/ec2/docker/logs" has been created.

Log groups (2)

By default, we only load up to 10,000 log groups.

Log group	Log class	Anomaly d...	Deletion...	Data ...	Sensit...	Retenti...	Met
/ec2/system/logs	Standard	Configure	Off	-	-	Never exp...	-
/ec2/docker/logs	Standard	Configure	Off	-	-	Never exp...	-



aws
Search [Alt+S] Ask Amazon Q
United States (N. Virginia)
Account ID: 8557-4019-4147
Coder
CloudWatch > Log management > /ec2/docker/logs > i-02454a89625ccb973
Actions
Start tailing
Create metric filter
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Back to top

Log events

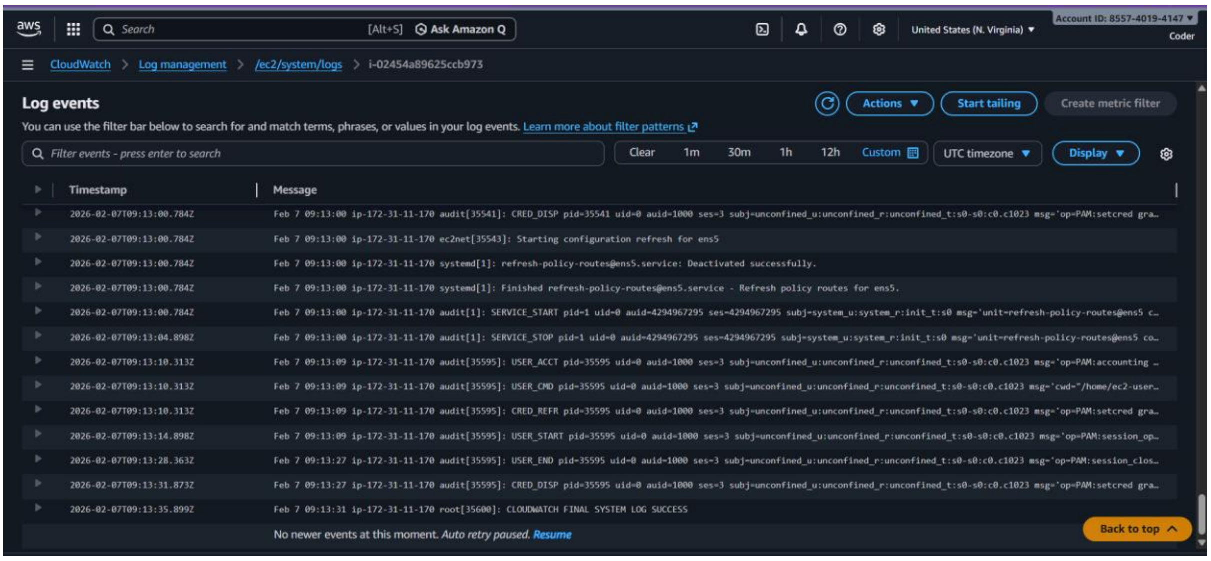
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

```
Timestamp                  Message
2026-02-07T08:30:11.336Z   {"log":"89.43.132.24 - - [07/Feb/2026:07:41:31 +0000] \"GET /index.php?lang=../../../../../../../../usr/local/lib/php/pearcmd\u0026+config-create+/\u0026/\u003c?echo(md5(…
2026-02-07T08:30:11.336Z   {"log":"2026/02/07 07:41:31 [error] 29#29: *16 open() \"/usr/share/nginx/html/index.php\" failed (2: No such file or directory), client: 89.43.132.24, server: localhost, …
2026-02-07T08:30:11.336Z   {"log":"2026/02/07 07:41:34 [error] 29#29: *16 open() \"/usr/share/nginx/html/index.php\" failed (2: No such file or directory), client: 89.43.132.24, server: localhost, …
2026-02-07T08:30:11.336Z   {"log":"89.43.132.24 - - [07/Feb/2026:07:41:34 +0000] \"GET /index.php?lang=../../../../../../../../tmp/index1 HTTP/1.1\" 404 153 \"-\" \"libredtail-http\" \"-\"\n","stre…
2026-02-07T08:30:11.336Z   {"log":"89.43.132.24 - - [07/Feb/2026:07:41:38 +0000] \"GET /containers/json HTTP/1.1\" 404 153 \"-\" \"libredtail-http\" \"-\"\n","stream":"stdout","time":"2026-02-07T07…
2026-02-07T08:30:11.336Z   {"log":"2026/02/07 07:41:38 [error] 29#29: *16 open() \"/usr/share/nginx/html/containers/json\" failed (2: No such file or directory), client: 89.43.132.24, server: local…
2026-02-07T08:30:11.336Z   {"log":"216.180.246.2 - - [07/Feb/2026:08:20:14 +0000] \"GET / HTTP/1.0\" 200 615 \"-\" \"'Mozilla/5.0 (compatible; GenomeCrawlerd/1.0; +https://www.nokia.com/genomecrawl…
2026-02-07T08:30:11.336Z   {"log":"216.180.246.2 - - [07/Feb/2026:08:20:23 +0000] \"\\x16\\x03\\x01\\x00\\xEE\\x01\\x00\\x00\\xEA\\x03\\x03F\\x91\\x86\\x82\\xEA.t\\xEB\\x0F\\x02K\\xBD}\\xD1O\\x8Ac\…
2026-02-07T08:30:11.336Z   {"log":"204.76.203.219 - - [07/Feb/2026:08:23:03 +0000] \"GET / HTTP/1.1\" 200 615 \"-\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko…
2026-02-07T08:30:11.336Z   {"log":"2026/02/07 08:29:47 [error] 30#30: *20 open() \"/usr/share/nginx/html/ReportServer\" failed (2: No such file or directory), client: 9.234.8.67, server: localhost,…
2026-02-07T08:30:16.606Z   {"log":"9.234.8.67 - - [07/Feb/2026:08:29:47 +0000] \"GET /ReportServer HTTP/1.1\" 404 153 \"-\" \"Mozilla/5.0 zgrab/0.x\" \"-\"\n","stream":"stdout","time":"2026-02-07T0…
2026-02-07T08:56:40.628Z   {"log":"147.185.132.180 - - [07/Feb/2026:08:56:40 +0000] \"\\x16\\x03\\x01\\x00\\xCA\\x01\\x00\\x00\\xC6\\x03\\x03\u003e,/\\x8C\\x22\\x8CrhD\\xFE\\xAE\\x82\\xCB\\xB3\\x1B…
2026-02-07T08:56:45.139Z   {"log":"147.185.132.180 - - [07/Feb/2026:08:56:40 +0000] \"\\x16\\x03\\x01\\x00\\xEE\\x01\\x00\\x00\\xEA\\x03\\x03?\\xAF\\x9B\\x9A\\xD6\\xF2\\x05\\x97\\xFEu\\xFB\\x17\\x9…
                           No newer events at this moment. Auto retry paused. Resume
```

7.2 Docker Logs

Docker container logs were collected from /var/lib/docker/containers/*/*.log. These logs capture application-level events generated by the Nginx container, including startup messages and HTTP request handling. This ensures visibility into container behavior and application health.

Docker Log Stream:


aws
Search [Alt+S]
Ask Amazon Q
United States (N. Virginia)
Account ID: 8557-4019-4147
Coder
CloudWatch > Log management > /ec2/system/logs > i-02454a89625ccb973
Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
Clear
1m
30m
1h
12h
Custom
UTC timezone
Display
Back to top

```
Timestamp                  Message
2026-02-07T09:13:00.784Z   Feb 7 09:13:00 ip-172-31-11-170 audit[35541]: CRED_DISP pid=35541 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:setcred gra…
2026-02-07T09:13:00.784Z   Feb 7 09:13:00 ip-172-31-11-170 ec2net[35543]: Starting configuration refresh for ens5
2026-02-07T09:13:00.784Z   Feb 7 09:13:00 ip-172-31-11-170 systemd[1]: refresh-policy-routes@ens5.service: Deactivated successfully.
2026-02-07T09:13:00.784Z   Feb 7 09:13:00 ip-172-31-11-170 systemd[1]: Finished refresh-policy-routes@ens5.service - Refresh policy routes for ens5.
2026-02-07T09:13:00.784Z   Feb 7 09:13:00 ip-172-31-11-170 audit[1]: SERVICE_START pid=1 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0 msg='unit=refresh-policy-routes@ens5 c…
2026-02-07T09:13:04.898Z   Feb 7 09:13:00 ip-172-31-11-170 audit[1]: SERVICE_STOP pid=1 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0 msg='unit=refresh-policy-routes@ens5 co…
2026-02-07T09:13:10.313Z   Feb 7 09:13:09 ip-172-31-11-170 audit[35595]: USER_ACCT pid=35595 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:accounting …
2026-02-07T09:13:10.313Z   Feb 7 09:13:09 ip-172-31-11-170 audit[35595]: USER_CMD pid=35595 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='cwd="/home/ec2-user…
2026-02-07T09:13:10.313Z   Feb 7 09:13:09 ip-172-31-11-170 audit[35595]: CRED_REFR pid=35595 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:setcred gra…
2026-02-07T09:13:14.898Z   Feb 7 09:13:09 ip-172-31-11-170 audit[35595]: USER_START pid=35595 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:session_op…
2026-02-07T09:13:28.363Z   Feb 7 09:13:27 ip-172-31-11-170 audit[35595]: USER_END pid=35595 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:session_clos…
2026-02-07T09:13:31.873Z   Feb 7 09:13:27 ip-172-31-11-170 audit[35595]: CRED_DISP pid=35595 uid=0 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAM:setcred gra…
2026-02-07T09:13:35.899Z   Feb 7 09:13:31 ip-172-31-11-170 root[35600]: CLOUDWATCH FINAL SYSTEM LOG SUCCESS
                           No newer events at this moment. Auto retry paused. Resume
```

8. GenAI-Assisted Security Review

8.1 Methodology

Infrastructure metadata, architectural details, and summarized log outputs were manually provided to GenAI (ChatGPT) for analysis. GenAI was used strictly in an advisory, human-in-the-loop role to:

- Identify risky configurations
- Highlight deviations from AWS best practices
- Suggest security and optimization improvements

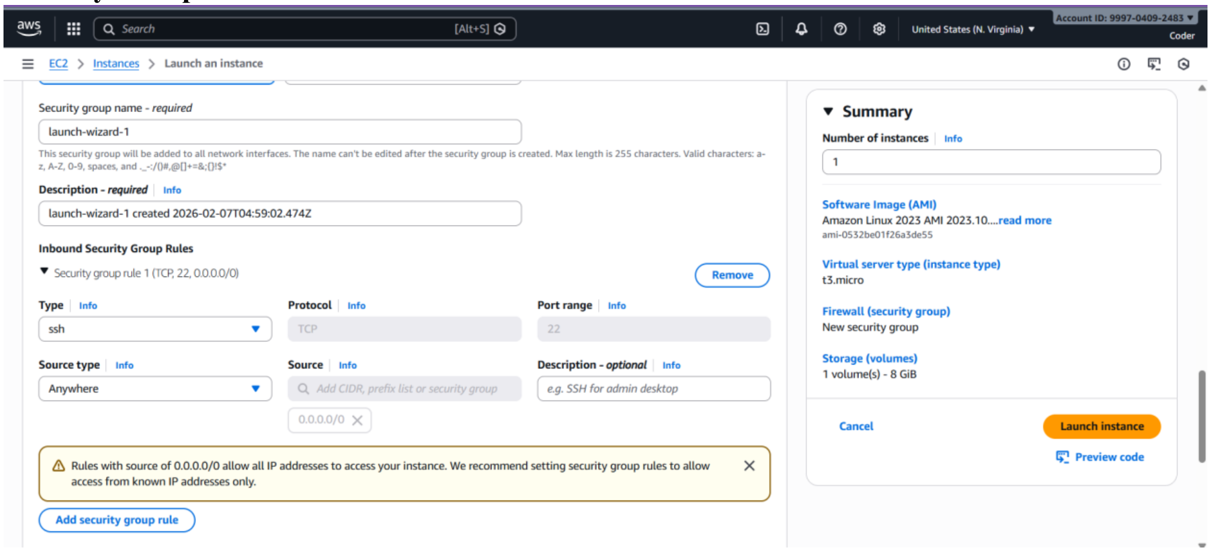
GenAI had no direct access to the AWS environment, ensuring full human control and eliminating the risk of unintended automated changes.

8.2 Findings and Risk Analysis

Based on the review, the following key findings were identified:

Category	**Finding**	**Risk Level**	**GenAI Recommendation**
Network	SSH (port 22) open to 0.0.0.0/0	High	Restrict IP range or use SSM
IAM	Broad permissions assigned	Medium	Apply least-privilege policies
Logging	Logs manually reviewed	Low	Enable alerts and automation

Security Group Inbound Rules:


aws
Search
[Alt+S]
United States (N. Virginia)
Account ID: 9997-0409-2483
Coder
EC2 > Instances > Launch an instance

Security group name - *required*

launch-wizard-1

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* | Info

launch-wizard-1 created 2026-02-07T04:59:02.474Z

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type: ssh
Protocol: TCP
Port range: 22
Source type: Anywhere
Source: 0.0.0.0/0
Description - *optional*: e.g. SSH for admin desktop

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Summary

Number of instances: 1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.10....read more
ami-0532be01f26a3de55

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

9. Security Hardening Recommendations

9.1 Identity and Access Management

To strengthen access control, the following IAM best practices are recommended:

- Enable Multi-Factor Authentication (MFA) for all IAM users
- Use IAM roles instead of long-term access keys
- Apply least-privilege permissions to reduce attack surface

9.2 Network Security

Network-level protections significantly reduce exposure to attacks:

- Restrict SSH access to specific IP addresses
- Replace SSH access with AWS Systems Manager Session Manager
- Enable VPC Flow Logs to monitor network traffic patterns

9.3 Container Security

Securing containers is essential in modern cloud deployments:

- Store images in Amazon ECR with vulnerability scanning enabled
- Avoid running containers with root privileges
- Regularly update base images and dependencies

9.4 Logging and Monitoring Enhancements

Further improvements to observability include:

- Creating CloudWatch Alarms for error patterns and anomalies
- Archiving logs to Amazon S3 with lifecycle policies
- Enabling Amazon GuardDuty for continuous threat detection

10. Benefits of GenAI Integration

Integrating GenAI into cloud reviews provides multiple benefits:

- Faster analysis of large and complex log datasets
- Reduced human error during manual reviews
- Improved decision-making through contextual insights
- Scalable advisory support for growing environments

GenAI acts as a security and operations sidekick, enhancing human expertise rather than replacing it.

11. Limitations of the Approach

Despite its benefits, the approach has certain limitations:

- GenAI analysis depends heavily on the quality of provided inputs
- No real-time or automated remediation is performed
- Requires skilled engineers to interpret and act on recommendations

12. Conclusion

This project successfully demonstrates an Intelligent Cloud Infrastructure Review using AWS services combined with GenAI assistance. By implementing centralized logging, secure architecture, and AI-driven analysis, the system delivers actionable security and operational insights while maintaining full human control.

The project aligns closely with enterprise-grade practices and the AWS Well-Architected Framework, proving that GenAI can significantly enhance cloud governance when used responsibly.

13. Future Enhancements

Future improvements to this project may include:

- Integrating AWS Config for continuous compliance monitoring
- Using Amazon Athena for advanced log analytics
- Adding automated alerting and incident workflows
- Extending the solution to multi-account AWS environments