

INTERNSHIP REPORT ON  
**MONITORING AND LOGGING FOR EC2 INSTANCES  
USING CLOUDWATCH**

Submitted in partial fulfilment of the requirements for the  
award of the degree of

**BACHELOR OF TECHNOLOGY  
ELECTRICAL AND ELECTRONICS**

**ENGINEERING**

Submitted by

**THONTA LALITHA DEVI**

**21B91A 02B3**

UNDER THE  
SUPERVISION OF

**YATISH KUMAR APPU P RP**

NILET VIRTUAL ACADAMEY ,CHENNAI

Duration: 05 -07 -2024 to 05 -09-2024



DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

**S.R.K.R ENGINEERING COLLEGE (A)**

(Affiliated to JNTUKAKINADA)(Approved by A.I.C.T.E, New Delhi )

(Accredited by N.B.A., NAAC with 'A+' grade, New Delhi)

SRKR MARG, CHINNAAMIRAM, BHIMAVARAM -534204.

(2024)

# S.R.K.R ENGINEERING COLLEGE(A)

(Affiliated to JNTUKAKINADA) (Recognized by A.I.C.T.E, New Delhi)  
(Accredited by N.B.A., NAAC with 'A+' grade, New Delhi)

CHINNAAMIRAM, BHIMAVARAM - 534204

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING



## Certificate

This is to certify that the summer Internship Report titled " **MONITORING AND LOGGING FOR EC2 INSTANCES USING CLOUDWATCH** " is the bonafied work done by **Ms. THONTA LALITHA DEVI (22B91A02B3)** at the end of Second year B. Tech., at **NIELIT VIRTUAL ACADEMY** from **05-06-2023 to 30-07-2023** in partial fulfilment of the requirements for the award of the "Degree of Bachelor of Technology" with specialization of Electrical and Electronics Engineering in S.R.K.R. Engineering College (A), Bhimavaram.

---

Dean, T&P

---

Mentor

---

Internship Coordinator

---

Head of the Department

## **TITLE:**

# **MONITORING AND LOGGING FOR EC2 INSTANCES USING CLOUDWATCH**

## **ABSTRACT :**

Cloud-based infrastructure requires robust monitoring and logging to ensure system reliability, performance, and security. This project focuses on implementing monitoring and logging solutions for Amazon EC2 instances using AWS CloudWatch. The objective is to provide real-time insights into the performance and health of EC2 instances, enabling proactive issue resolution and optimization.

The project involves configuring the CloudWatch agent, setting up detailed monitoring, collecting custom metrics, and configuring log streams to centralize logs for analysis. Additionally, it explores setting up alarms and notifications to alert administrators about critical events. The result is an automated and efficient system for maintaining EC2 instance performance and ensuring business continuity.

This solution leverages the scalability and efficiency of AWS CloudWatch, highlighting its role as a vital tool in cloud infrastructure management.

## TABLE OF CONTENT :

### 1. **\*\*Abstract\*\***

- Overview of the project and its importance ..... 1

### 2. **\*\*Introduction\*\***

- Background on EC2 instances and CloudWatch
- Importance of monitoring and logging in cloud environments .....2

### 3. **\*\*Literature Survey\*\***

- Existing tools and methods for cloud monitoring
- Comparative analysis of AWS CloudWatch .....3

### 4. **\*\*Objective\*\***

- Goals of the project
- Key deliverables .....4

### 5. **\*\*Block Diagram\*\***

- High-level architecture of the setup .....5 - 17

### 6. **\*\*Proposed Setup\*\***

- Step-by-step setup for monitoring and logging
- Explanation of workflow ..... 17 - 20

### 7. **\*\*Hardware & Software Tools\*\***

- AWS services involved
- Other tools used

### 8. **\*\*Results\*\***

- Insights from the monitoring setup
- Improvements and optimizations observed

### 9. **\*\*Conclusion\*\***

- Summary of findings
- Future scope

### 10. **\*\*References\*\***

- List of resources and citations

# INTRODUCTION

## BACKGROUND ON EC2 INSTANCES AND CLOUDWATCH :

Amazon Elastic Compute Cloud (EC2) is a core component of AWS, providing scalable virtual servers in the cloud. EC2 allows users to configure instances with desired computing power, storage, and operating systems, making it ideal for hosting applications, databases, and services. The flexibility of EC2 enables dynamic scaling to meet varying workload demands, ensuring cost-effectiveness and performance.

AWS CloudWatch is a monitoring and observability service designed to provide actionable insights into AWS resources, including EC2 instances. It collects and tracks metrics, monitors log files, and sets alarms to react to changes in the environment. CloudWatch plays a critical role in maintaining the health, performance, and availability of cloud infrastructure by offering real-time monitoring and detailed logs.

## IMPORTANCE OF MONITORING AND LOGGING IN CLOUD ENVIRONMENTS :

Monitoring and logging are vital in cloud environments to ensure the stability, security, and performance of applications and systems. Key reasons include:

1. Proactive Issue Detection
  - Monitoring enables early detection of anomalies, such as resource exhaustion, high latency, or unexpected failures.
  - Logs provide detailed event information to troubleshoot issues effectively.
2. Performance Optimization
  - Insights from metrics help optimize resource usage, reduce latency, and improve application performance.
  - Analysis of usage patterns aids in making cost-effective decisions.
3. Security and Compliance
  - Logging tracks user activities and system changes, ensuring compliance with security policies and regulations.
  - Alarms notify administrators of potential security breaches or misconfigurations.
4. Automation and Scalability
  - CloudWatch facilitates automation by triggering actions like scaling or notifications based on monitored metrics.
  - It supports managing infrastructure at scale without manual intervention.

# LITERATURE SURVEY

## EXISTING TOOLS AND METHODS FOR CLOUD MONITORING:

- Nagio: Offers basic infrastructure monitoring but requires significant manual configuration.
- Datadog: Provides advanced analytics and dashboards but comes at a higher cost.
- Prometheus: Ideal for metrics collection but lacks seamless integration with cloud services.

## COMPARATIVE ANALYSIS OF AWS CLOUDWATCH:

- Native AWS Integration: Unlike external tools, CloudWatch directly integrates with all AWS services.
- Scalability: Automatically adapts to AWS resources without additional setup.
- Real-Time Monitoring: Offers immediate metrics, logs, and alarms, ensuring prompt action.
- Cost Efficiency: Tailored for AWS users with pay-as-you-go pricing, reducing overhead..
- CloudWatch is optimized for AWS environments, making it a powerful and efficient choice for cloud monitoring.

# OBJECTIVE

## GOALS OF THE PROJECT:

- Implement an efficient monitoring and logging setup for EC2 instances using AWS CloudWatch.
- Enable real-time performance tracking and proactive issue detection.
- Configure alerts and notifications for critical events to ensure system reliability.

## KEY DELIVERABLES:

- Installation and configuration of the CloudWatch agent on EC2 instances.
- Centralized logging and custom metric setup.
- Alarm configuration to monitor thresholds and trigger notifications.
- A streamlined system to enhance infrastructure management.

# OBJECTIVE

## WHAT IS EC2?

**\*\*Amazon EC2 (Elastic Compute Cloud)\*\*** is a web service provided by Amazon Web Services (AWS) that allows you to run virtual machines (known as **\*\*instances\*\***) in the cloud. EC2 provides scalable computing capacity, making it easier to deploy and manage applications without having to invest in physical hardware.



## KEY FEATURES OF EC2:

### 1. **\*\*Scalability\*\***:

EC2 instances can be easily scaled up or down based on the demand. You can start with one instance and scale to hundreds or thousands of instances as your application's needs grow.

### 2. **\*\*Flexible Instance Types\*\***:

EC2 offers different types of instances optimized for various workloads, including compute, memory, storage, or network-intensive applications.

### 3. **\*\*Pay-as-You-Go Pricing\*\***:

With EC2, you only pay for the compute capacity you actually use. You can select from different pricing models such as On-Demand, Reserved, or Spot instances based on your cost and usage needs.

### 4. **\*\*Customizable\*\***:

You can choose the operating system (Linux, Windows, etc.), the instance type, the size, and the network configurations of your EC2 instance.



. **\*\*Security\*\***:

EC2 instances are secured within Virtual Private Clouds (VPC), and you can control access using security groups and network access control lists (ACLs).

6. **\*\*High Availability\*\***:

You can launch EC2 instances across multiple availability zones within AWS to improve the availability and fault tolerance of your application.

7. **\*\*Integration with AWS Services\*\***:

EC2 integrates with other AWS services like S3 (for storage), RDS (for databases), CloudWatch (for monitoring), and many others.

## **USE CASES OF EC2:**

- **\*\*Web Hosting\*\***: Run websites or web applications on EC2 instances.
- **\*\*Application Hosting\*\***: Host enterprise applications, databases, and services in the cloud.
- **\*\*Data Processing\*\***: Perform high-performance computing (HPC) tasks, machine learning, and data analytics.
- **\*\*Development and Testing\*\***: Create environments for testing and development without having to set up physical infrastructure.

## **HOW EC2 WORKS:**

1. **\*\*Launch an Instance\*\***:

You can launch an EC2 instance by selecting an AMI (Amazon Machine Image) that defines the operating system and other configurations you need. You also select an instance type based on your performance requirements.

## 2. **\*\*Configure Networking\*\***:

Each EC2 instance is launched within a **\*\*Virtual Private Cloud (VPC)\*\***, where you can configure security settings, IP addresses, and routing.

## 3. **\*\*Access and Management\*\***:

Once your instance is running, you can SSH (for Linux) or RDP (for Windows) to the instance to install software, manage files, and run applications.

## 4. **\*\*Monitor and Scale\*\***:

You can monitor the performance of your EC2 instances using **\*\*CloudWatch\*\*** and adjust the capacity using features like **\*\*Auto Scaling\*\*** to automatically add or remove instances based on demand.

EC2 is the foundation for many applications in the cloud, providing the computing power needed to run everything from small websites to large-scale enterprise applications

## **AMAZON CLOUDWATCH:**

Amazon CloudWatch is a monitoring and observability service provided by AWS that gives you insight into the performance and health of your AWS resources and applications. It collects and tracks metrics, logs, and events, allowing you to monitor resource usage, set alarms, and automate actions based on those metrics.



## **KEY FEATURES:**

1. Metrics Monitoring:
2. CloudWatch collects metrics from AWS services like EC2, RDS, and Lambda, allowing you to monitor CPU usage, disk activity, network traffic, and more.
3. Logs Management:
4. It helps you collect, monitor, and store log data from various AWS resources and applications. This includes server logs, application logs, and custom logs.
5. Alarms:
6. CloudWatch allows you to set alarms based on specific thresholds (like CPU utilization, memory usage, etc.). If the metrics breach the threshold, CloudWatch triggers actions such as sending notifications or invoking AWS Lambda functions.
7. Dashboards:
8. You can create visual dashboards in CloudWatch to get an overview of key metrics, making it easier to monitor multiple resources at once.
9. Events and Automation:
10. CloudWatch can be set up to trigger automated responses to specific events or conditions, such as scaling up EC2 instances when high traffic is detected.

## **USE CASES FOR CLOUDWATCH:**

1. **Infrastructure Monitoring:** Monitor EC2 instances, EBS volumes, and RDS databases to track their health and performance.
2. **Log Aggregation:** Aggregate and analyze logs from various AWS services and applications to troubleshoot and optimize performance.
3. **Alerting and Auto Scaling:** Set up alarms to be notified when thresholds are breached and use auto-scaling to handle changes in demand.

## **AMAZON SNS (SIMPLE NOTIFICATION SERVICE)**

Amazon SNS is a fully managed messaging service that allows you to send notifications from the cloud to distributed endpoints such as email addresses, SMS, or other services. It's a simple, cost-effective way to send messages and alerts.

## **USE CASES FOR SNS:**

1. **Alerting:** Send notifications about critical system events or alarms from CloudWatch (e.g., CPU utilization alerts).
2. **Application Integration:** Notify multiple services or systems about state changes (e.g., file uploads or order status changes).
3. **Messaging:** Send push notifications or messages to users on mobile devices or web applications.

## **USE CASES FOR CLOUDWATCH:**

1. Infrastructure Monitoring: Monitor EC2 instances, EBS volumes, and RDS databases to track their health and performance.
2. Log Aggregation: Aggregate and analyze logs from various AWS services and applications to troubleshoot and optimize performance.
3. Alerting and Auto Scaling: Set up alarms to be notified when thresholds are breached and use auto-scaling to handle changes in demand.

## **AMAZON SNS (SIMPLE NOTIFICATION SERVICE)**

Amazon SNS is a fully managed messaging service that allows you to send notifications from the cloud to distributed endpoints such as email addresses, SMS, or other services. It's a simple, cost-effective way to send messages and alerts.

## **USE CASES FOR SNS:**

1. Alerting: Send notifications about critical system events or alarms from CloudWatch (e.g., CPU utilization alerts).
2. Application Integration: Notify multiple services or systems about state changes (e.g., file uploads or order status changes).
3. Messaging: Send push notifications or messages to users on mobile devices or web applications.

## **CLOUDWATCH AGENT**

The CloudWatch Agent is a lightweight software that you install on your EC2 instances (or other resources) to collect additional metrics and logs that are not automatically available through CloudWatch. The agent sends these metrics and logs to CloudWatch for monitoring and analysis.

### **USE CASES:**

1. Enhanced EC2 Monitoring: Collect detailed resource metrics, such as memory usage and swap usage, which are not available by default.
2. Log Aggregation: Capture application logs, error logs, or other custom logs and store them in CloudWatch for analysis.

## **AM ROLE FOR EC2 INSTANCE**

IAM (Identity and Access Management) roles are used to grant permissions to AWS services like EC2. In your project, an IAM role was used to grant the EC2 instance permissions to publish logs to CloudWatch and send messages to SNS topics.

### **USE CASES:**

- Granting Permissions: In your project, an IAM role was created and attached to the EC2 instance to give it permission to send logs to CloudWatch and notifications to SNS

# CLOUDWATCH ALARMS

CloudWatch Alarms are used to monitor CloudWatch metrics and trigger actions (like sending notifications) when specific thresholds are breached. In your project, you set up alarms to track CPU utilization on the EC2 instance, and when the CPU usage exceeded a predefined threshold (e.g., 80%), an SNS notification was triggered.

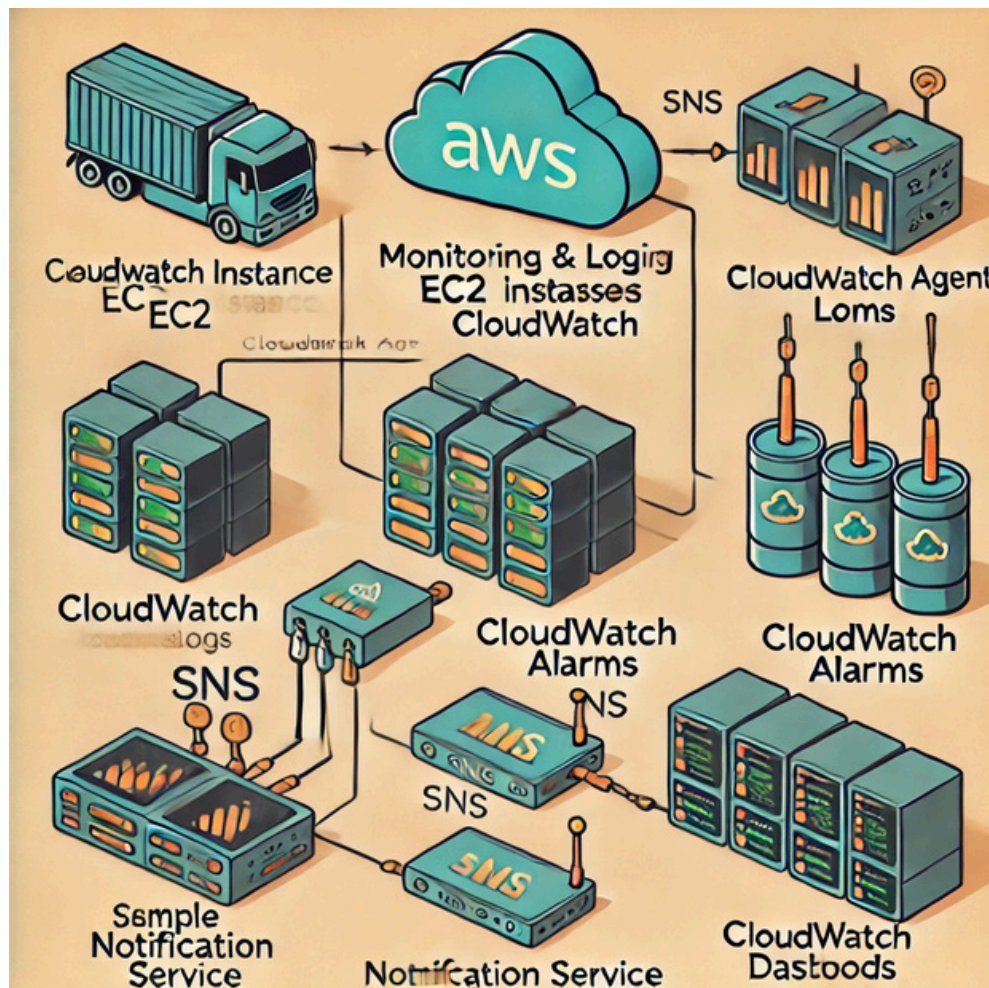
## KEY FEATURES:

- Threshold-based Alerts:
- Alarms are based on specific thresholds (e.g., when CPU usage exceeds 80%) and can be customized based on your requirements.
- Automatic Actions:
- Alarms can trigger automated actions like scaling EC2 instances or sending notifications.

## USE CASES:

- Monitor EC2 Instance Health: You used CloudWatch Alarms to monitor CPU usage and alert you when utilization was high, helping to prevent performance degradation or system failures.

# BLOCK DIAGRAM



- For a high-level architecture of an EC2 monitoring setup using CloudWatch, the block diagram could include the following components:
- 1. **EC2 Instances**: These are the virtual machines running applications or services.
- 2. **CloudWatch Agent**: Installed on EC2 instances, it collects system-level metrics and logs (CPU, memory, disk, etc.).
- 3. **CloudWatch Metrics**: These are the collected data points sent to CloudWatch for real-time monitoring.
- 4. **CloudWatch Alarms**: Set based on specific thresholds (e.g., CPU usage), they trigger notifications or actions when exceeded.



- 5. **\*\*CloudWatch Logs\*\***: Captures and stores log data for troubleshooting and monitoring.
- 6. **\*\*SNS (Simple Notification Service)\*\***: Sends notifications to administrators or triggers automated actions based on alarm conditions.

This architecture ensures real-time monitoring and management of EC2 instances, enabling efficient resource allocation and issue resolution

## VISUALIZING METRICS WITH DASHBOARDS



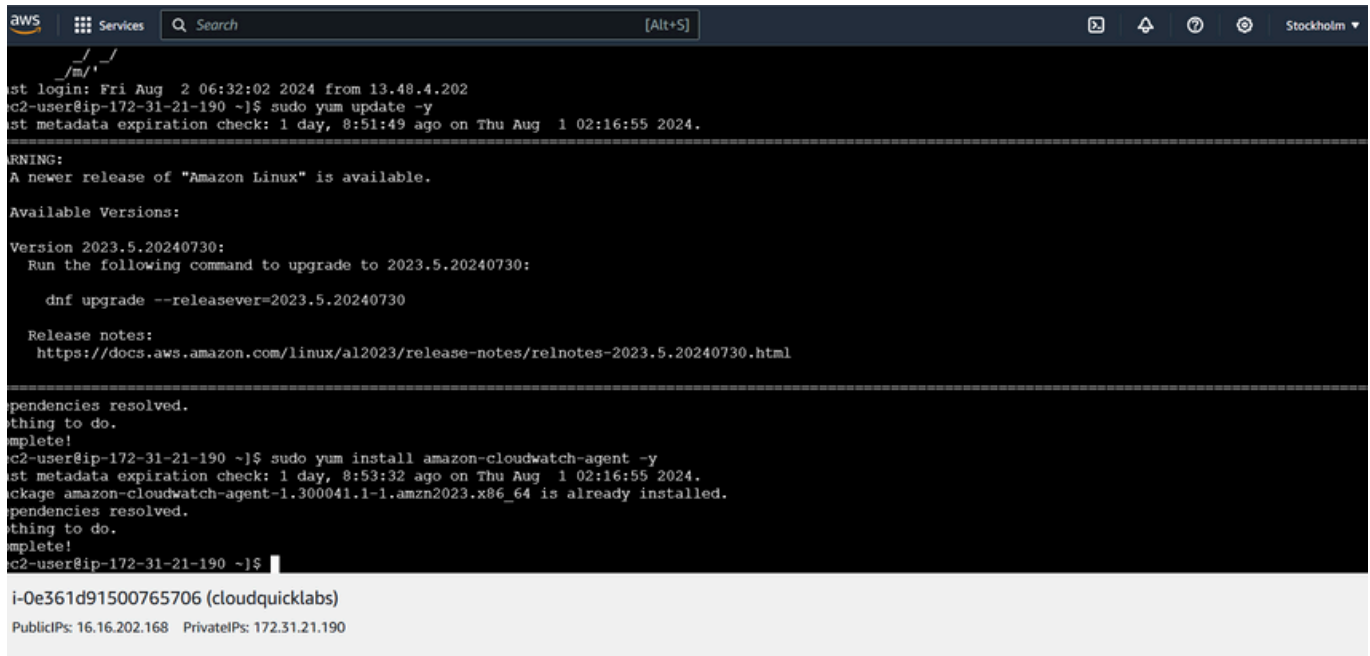
Creating CloudWatch dashboards to visualize metric and logs



Customizing dashboards for effective monitoring

# PROPOSED SETUP: STEP-BY-STEP SETUP FOR MONITORING AND LOGGING

- STEP 1 : INSTALL THE CLOUDWATCH AGENT



```
aws
Services Search [Alt+S] Stockholm
st login: Fri Aug 2 06:32:02 2024 from 13.48.4.202
c2-user@ip-172-31-21-190 ~]$ sudo yum update -y
st metadata expiration check: 1 day, 8:51:49 ago on Thu Aug 1 02:16:55 2024.

WARNING:
A newer release of "Amazon Linux" is available.

Available Versions:

Version 2023.5.20240730:
Run the following command to upgrade to 2023.5.20240730:

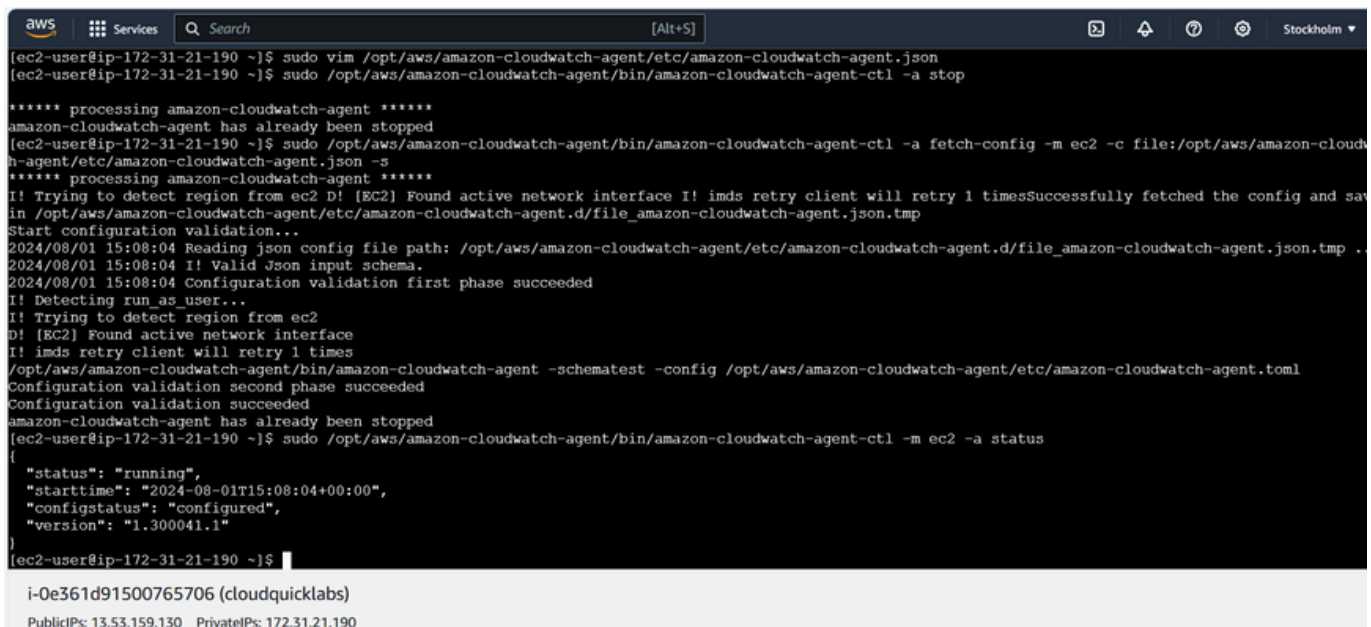
dnf upgrade --releasever=2023.5.20240730

Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.5.20240730.html

Dependencies resolved.
Nothing to do.
Complete!
c2-user@ip-172-31-21-190 ~]$ sudo yum install amazon-cloudwatch-agent -y
st metadata expiration check: 1 day, 8:53:32 ago on Thu Aug 1 02:16:55 2024.
Package amazon-cloudwatch-agent-1.300041.1-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
c2-user@ip-172-31-21-190 ~]$
```

i-0e361d91500765706 (cloudquicklabs)  
PublicIPs: 16.16.202.168 PrivateIPs: 172.31.21.190

- Update EC2 Instance
- Install The CloudWatch Agent
- Create a CloudWatch Agent Configuration File
- Start the Cloudwatch Agent

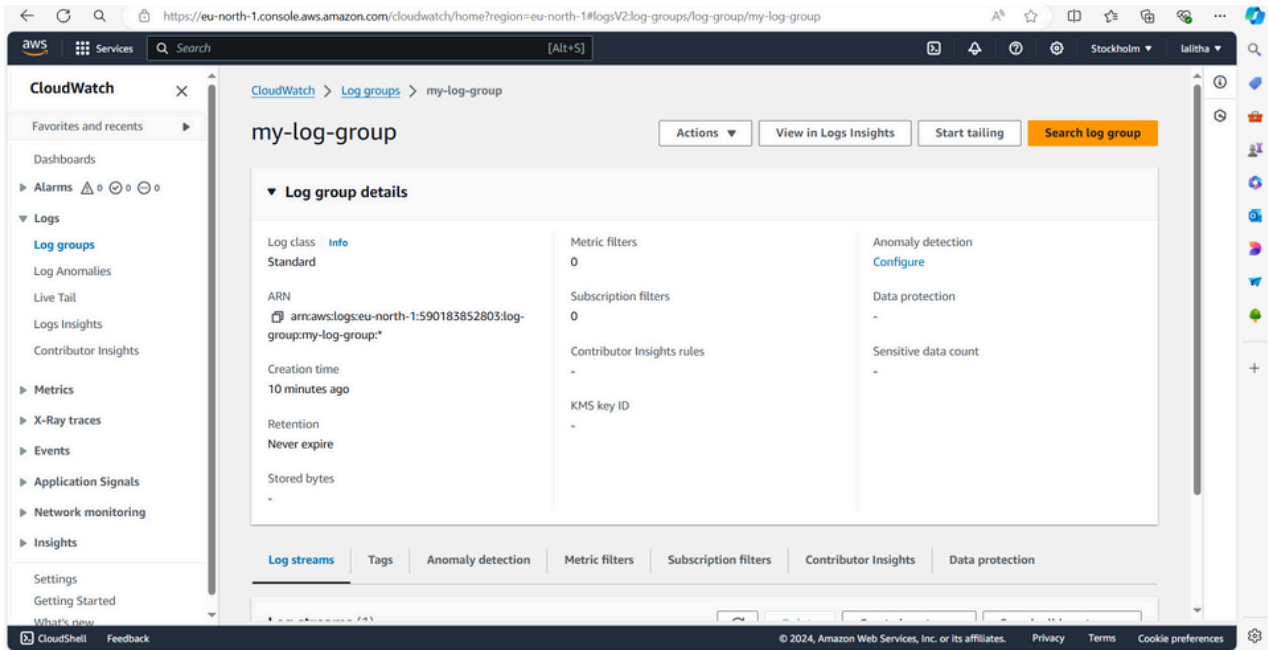


```
aws
Services Search [Alt+S] Stockholm
[ec2-user@ip-172-31-21-190 ~]$ sudo vim /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json
[ec2-user@ip-172-31-21-190 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a stop

***** processing amazon-cloudwatch-agent *****
amazon-cloudwatch-agent has already been stopped
[ec2-user@ip-172-31-21-190 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json -s
***** processing amazon-cloudwatch-agent *****
I! Trying to detect region from ec2 D! [EC2] Found active network interface I! imds retry client will retry 1 timesSuccessfully fetched the config and saved it in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_amazon-cloudwatch-agent.json.tmp
Start configuration validation...
2024/08/01 15:08:04 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_amazon-cloudwatch-agent.json.tmp
2024/08/01 15:08:04 I! Valid Json input schema.
2024/08/01 15:08:04 Configuration validation first phase succeeded
I! Detecting run_as user...
I! Trying to detect region from ec2
D! [EC2] Found active network interface
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
[ec2-user@ip-172-31-21-190 ~]$ sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status
{
  "status": "running",
  "starttime": "2024-08-01T15:08:04+00:00",
  "configstatus": "configured",
  "version": "1.300041.1"
}
[ec2-user@ip-172-31-21-190 ~]$
```

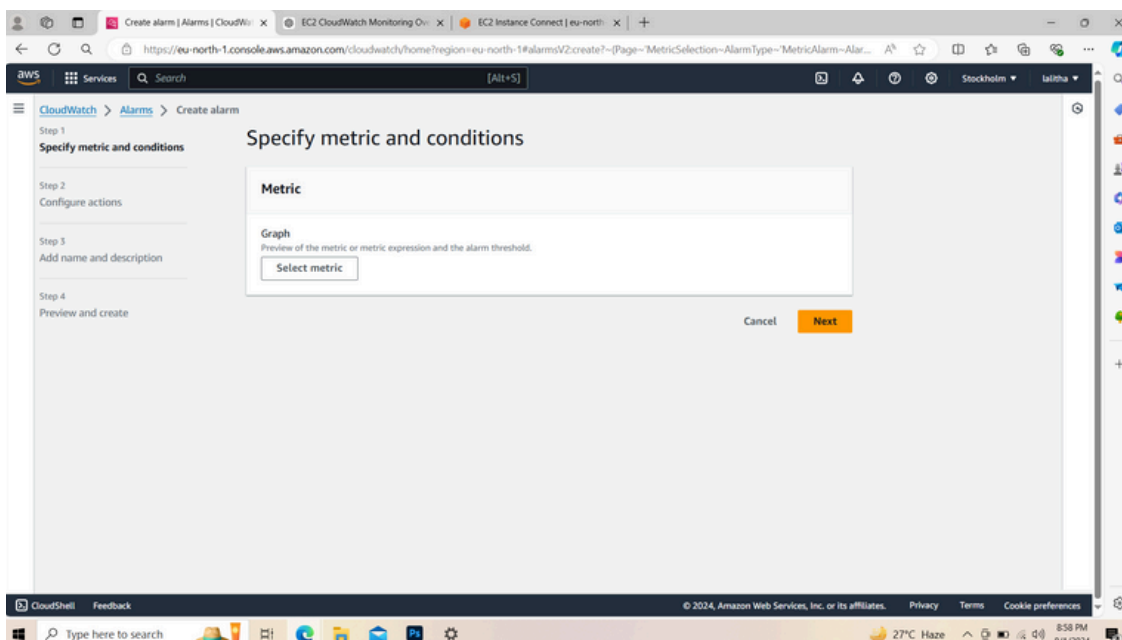
i-0e361d91500765706 (cloudquicklabs)  
PublicIPs: 13.53.159.130 PrivateIPs: 172.31.21.190

# STEP 2 : CONFIGURE CLOUDWATCH LOGS



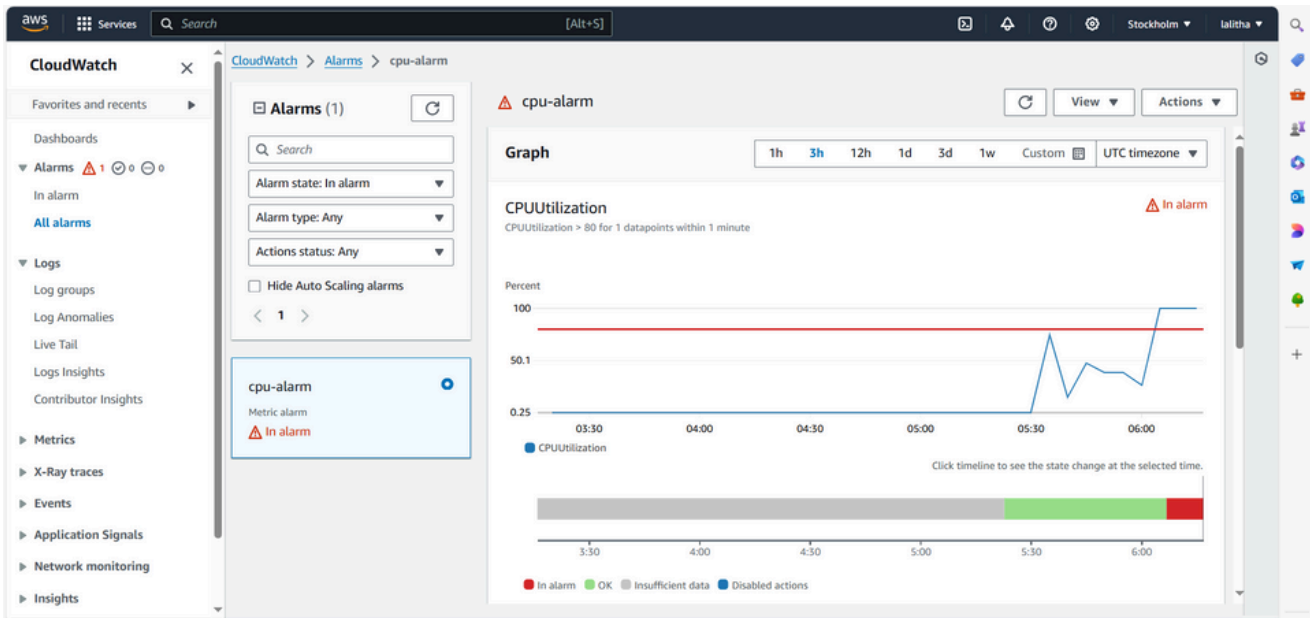
- Create a Log Group
- Configure Log Stream
- Verify Log Data

# STEP 3 : SET UP CLOUDWATCH ALARMS

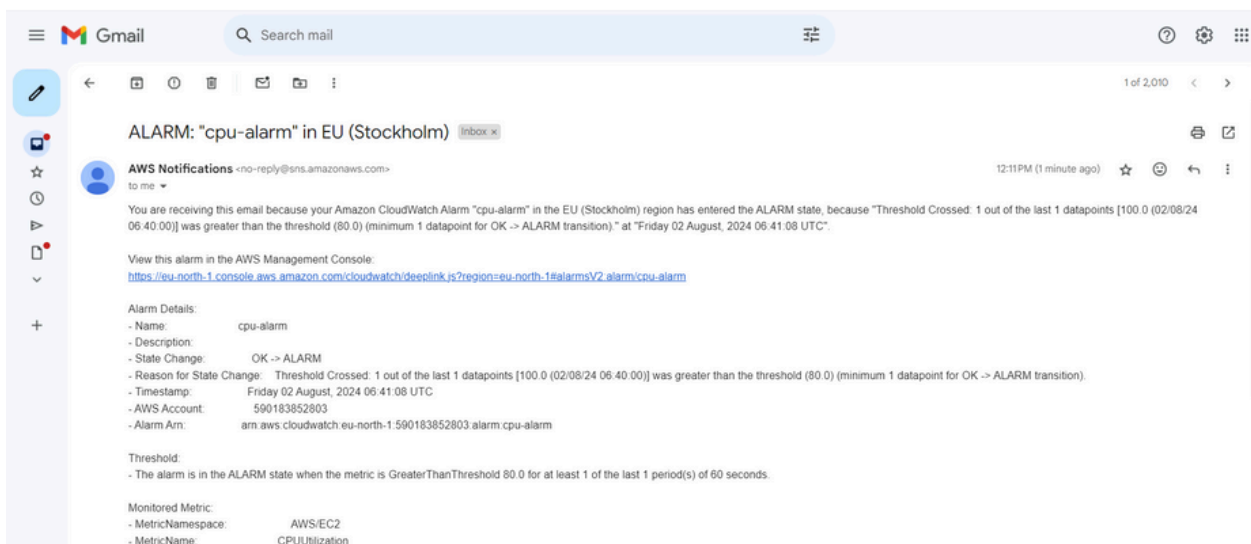


- Create an Alarm
- Setup SNS for Notifications

## STEP 5 : TESTING AND VALIDATION



- Verify Monitoring and Alarms
- Generate Load



## LINUX COMMANDS

- `sudo yum install -y amazon-cloudwatch-agent`
- `sudo vim /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json`
- `sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a action=configure -c file:/opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json -s`
- `aws logs describe-log-groups`
- `aws logs describe-log-streams --log-group-name "MyLogGroup"`
- `aws sns create-topic --name MyCPUTopic`
- `aws sns list-topics`
- `aws cloudwatch put-metric-alarm --alarm-name "HighCPUUtilizationAlarm" --metric-name "CPUUtilization" --namespace "AWS/EC2" --statistic "Average" --period 60 --threshold 80 --comparison-operator "GreaterThanOrEqualToThreshold" --evaluation-periods 1 --alarm-actions "arn:aws:sns:region:account-id:MyCPUTopic"`
- `stress --cpu 2 --timeout 60`
- `aws cloudwatch describe-alarms`
- `aws ec2 terminate-instances --instance-ids i-xxxxxxx`
- `aws sns delete-topic --topic-arn arn:aws:sns:region:account-id:MyCPUTopic`

## Hardware & Software Tools:

### \*AWS Services Involved\*

#### 1. \*\*Amazon EC2 (Elastic Compute Cloud)\*\*:

- **Purpose**: EC2 provides resizable compute capacity in the cloud. We used EC2 instances to deploy our application and run the necessary services.

- **Role in Project**: The EC2 instance was the main host where we installed the necessary software and executed workloads that were monitored.

#### 2. \*\*Amazon CloudWatch\*\*:

- **Purpose**: CloudWatch is a monitoring and observability service used to collect and track metrics, collect and monitor log files, and set alarms.

- **Role in Project**: CloudWatch was used to track the performance metrics of the EC2 instance, including CPU utilization, disk activity, and network traffic. CloudWatch alarms were set up to trigger notifications if any thresholds (like CPU utilization exceeding 80%) were breached.

#### 3. \*\*Amazon CloudWatch Logs\*\*:

- **Purpose**: CloudWatch Logs helps collect and store logs from various sources for monitoring and troubleshooting.

- **Role in Project**: We used CloudWatch Logs to capture logs from the EC2 instance. This provided detailed insights into system and application behavior, allowing us to identify errors, troubleshoot issues, and improve performance.

#### 4. \*\*Amazon SNS (Simple Notification Service)\*\*:

- **Purpose**: SNS is a fully managed messaging service for sending notifications.

- **Role in Project**: SNS was used to send email notifications whenever CloudWatch alarms were triggered (e.g., CPU utilization exceeded 80%). This helped alert the team about potential issues with the EC2 instance in real-time.

resources.

5. **AWS IAM (Identity and Access Management)**:

- **Purpose**: IAM is used to securely control access to AWS resources.

- **Role in Project**: IAM was used to create roles and permissions for users and resources, ensuring that the necessary access rights were in place to configure CloudWatch, EC2, and SNS.

6. **Amazon CloudWatch Agent**:

- **Purpose**: The CloudWatch agent is used to collect and send metrics and logs from EC2 instances to CloudWatch.

- **Role in Project**: The CloudWatch agent was installed on the EC2 instance to collect system-level metrics and send them to CloudWatch for monitoring and logging.

**Other Tools Used**

1. **SSH (Secure Shell)**:

- **Purpose**: SSH is a protocol used for secure remote login to servers.

- **Role in Project**: SSH was used to access the EC2 instance for configuration and to install software, including the CloudWatch agent.

2. **AWS CLI (Command Line Interface)**:

- **Purpose**: AWS CLI is a command-line tool for interacting with AWS services.

- **Role in Project**: The AWS CLI was used to configure CloudWatch alarms, manage EC2 instances, and interact with SNS notifications.

3. **Vim or Nano (Text Editors)**:

- **Purpose**: These are text editors used to modify configuration files on Linux-based EC2 instances.

- **Role in Project**: We used Vim or Nano to edit configuration files, such as the CloudWatch agent's configuration files, to specify the metrics and logs to collect.

. **Linux Tools (Top, ps, etc.)**:

- **Purpose**: These are native Linux utilities to monitor system resources.

- **Role in Project**: Tools like `top`, `ps`, and `htop` were used to monitor the system's resource utilization (e.g., CPU, memory) and validate the CloudWatch monitoring setup.

5. **Stress Testing Tools (e.g., stress, stress-ng)**:

- **Purpose**: Stress testing tools are used to put a load on the system and simulate high CPU utilization.

- **Role in Project**: We used these tools to simulate high CPU usage and trigger CloudWatch alarms, ensuring the monitoring and alerting systems were functioning correctly.

## RESULTS

### - INSIGHTS FROM THE MONITORING

- **Real-Time Monitoring:**

By setting up CloudWatch monitoring, we were able to monitor key metrics such as CPU utilization, disk activity, and network traffic in real-time. This allowed us to identify performance trends, spot potential issues, and keep track of how the EC2 instance was performing over time.

- **Threshold Breaches and Alerts:**

The CloudWatch alarms effectively detected when CPU utilization exceeded the predefined threshold (e.g., 80%). Once the threshold was breached, CloudWatch triggered the SNS notification, alerting us of potential system performance issues. This demonstrated the effectiveness of the alarm setup for proactive monitoring.

- **Logging and Troubleshooting:**

With the CloudWatch Logs integration, we were able to collect logs from the EC2 instance, providing detailed insights into application behavior, system performance, and any errors or issues that might have occurred. This data was crucial for troubleshooting and improving the application's stability.

- **Real-time Notifications:**

The SNS notifications were successfully triggered whenever the defined threshold for CPU utilization was breached. This allowed for immediate action to be taken if needed, either by investigating the cause of the high CPU usage or by scaling the EC2 instance resources.



## **-IMPROVEMENTS AND OPTIMIZATIONS OBSERVED**

- **Proactive Resource Management:**

By leveraging CloudWatch alarms, we were able to proactively manage EC2 resources. When CPU utilization exceeded thresholds, the alert prompted immediate investigation, helping to identify when scaling or additional resources were required, thus avoiding performance degradation.

- **Better Visibility into EC2 Instance Health:**

The CloudWatch dashboard provided a centralized view of the EC2 instance's performance, giving a quick overview of system health. This helped identify patterns in resource consumption, enabling us to adjust the instance's configuration or load balancing strategies to optimize performance.

- **Log Aggregation and Error Handling:**

CloudWatch Logs allowed us to aggregate all logs in a single place. By analyzing these logs, we identified potential bottlenecks or inefficient processes, enabling us to optimize the EC2

## **CONCLUSION**

The monitoring and logging setup using CloudWatch in this project provided valuable insights into the EC2 instance's performance and resource utilization. By leveraging CloudWatch's real-time monitoring, alarms, and logs, we were able to optimize the EC2 instance for better performance, troubleshoot more effectively, and ensure that any system anomalies were promptly detected and addressed. This setup laid the foundation for a robust, scalable, and cost-efficient AWS infrastructure.

## DECLARATION

The Summer Internship Project report entitled “Monitoring and Logging for EC2 Instances Using CloudWatch ” has been carried out by me in the partial fulfilment of therequirements for the award of the degree of Bachelor of Technology in the Department ofElectrical and Electronics Engineering ,Sagi Rama Krishnam Raju Engineering College (A),Bhimavaram. I hereby declare that this work has not been submitted to any otheruniversity/institutefortheawardofanyotherdegree/diploma.

ROLL. NO.

\_\_\_\_\_

22B91A02B3

NAME

\_\_\_\_\_

T.LALITHA DEVI

\_\_\_\_\_

Signature