Lab 4: Cloud Security Monitoring Dashboard using AWS CloudWatch

1. Objective

The objective of this experiment is to set up a monitoring and alerting system for a cloud application using AWS CloudWatch. The setup includes monitoring EC2 instance performance metrics (CPU utilization), configuring alerts through SNS, visualizing metrics in a dashboard, and validating the alerts by simulating high CPU usage.

2. Prerequisites

- AWS Free Tier account
- AWS CLI installed and configured locally (AWS CLI v2 MSI Installer (Windows 64-bit))
- Python 3.x installed
- EC2 Key Pair (.pem file)
- Basic understanding of AWS services (EC2, CloudWatch, SNS)

3. Step-by-Step Procedure

Step 1: Configure AWS CLI

On your local machine, configure the AWS CLI with your Access Key, Secret Key, region, and output format:

```
aws configure
AWS Access Key ID: <your-access-key>
AWS Secret Access Key: <your-secret-key>
Default region name: us-east-1
Default output format: json
```

Step 2: Launch EC2 Instance

1. In the AWS Console, search for **EC2**.
2. Click **Launch Instance** → Enter a name like Security-Monitoring-Instance.
3. **AMI (Amazon Machine Image):** Select **Amazon Linux 2 AMI (Free Tier)**.
4. **Instance type:** Choose **t3.micro (Free Tier eligible)**.
5. **Key pair (login):**
    - Select "Create new key pair."
    - Download the .pem file and save it securely (you'll use it for SSH).
6. **Network settings:**
    - Enable **Auto-assign Public IP**.
    - Allow **SSH (22)**, **HTTP (80)**, and optionally **HTTPS (443)**.
7. Leave storage at default (8GB).
8. Click **Launch Instance**.
9. Once running → copy **Public IPv4 address**.

Step 3: Connect to the EC2 Instance

Use SSH from your local machine:

```
ssh -i Pair1.pem ec2-user@<EC2-Public-IP>
```

Step 4: Install Stress Tool

Inside the EC2 instance:

```
sudo dnf install stress -y
```

Step 5: Run Stress Test

Simulate CPU load for 5 minutes:

```
stress --cpu 2 --timeout 300
```

Step 6: Create SNS Topic & Subscription

Run the provided Python script (create_sns_topic.py):

python create_sns_topic.py

Confirm the subscription from your email inbox.

```
import boto3

# Initialize SNS client
sns_client = boto3.client("sns", region_name="us-east-1")

def create_sns_topic():
    topic_name = "SecurityAlertsTopic"

    # Create SNS topic
    response = sns_client.create_topic(Name=topic_name)
    topic_arn = response["TopicArn"]
    print(f"SNS topic created: {topic_arn}")

    # Subscribe your email (update with your email ID)
    email = "xyz@gmail.com"
    sns_client.subscribe(
        TopicArn=topic_arn,
        Protocol="email",
        Endpoint=email
    )
    print(f"Subscription request sent to {email}. Please
confirm from your inbox.")

    return topic_arn

if __name__ == "__main__":
    create_sns_topic()
```

Step 7: Create CloudWatch Alarm

Run the provided Python script (create_cloudwatch_alarm.py):
python create_cloudwatch_alarm.py

This creates an alarm that triggers when CPU utilization exceeds 70%.

```
import boto3

# Initialize CloudWatch client
cloudwatch_client = boto3.client("cloudwatch",
region_name="us-east-1")

def create_alarm(instance_id, sns_topic_arn):
    alarm_name = f"HighCPU-{instance_id}"

    response = cloudwatch_client.put_metric_alarm(
        AlarmName=alarm_name,
        ComparisonOperator="GreaterThanThreshold",
        EvaluationPeriods=1,
        MetricName="CPUUtilization",
        Namespace="AWS/EC2",
        Period=300,
        Statistic="Average",
        Threshold=70.0,
        ActionsEnabled=True,
        AlarmActions=[sns_topic_arn],
        AlarmDescription="Alarm when CPU exceeds 70%",
        Dimensions=[
            {"Name": "InstanceId", "Value": instance_id}
        ]
    )

    print(f"CloudWatch Alarm created: {alarm_name}")

if __name__ == "__main__":
    # Replace with your EC2 instance ID and the topic ARN
returned from create_sns_topic.py
    instance_id = "i-12345"
    sns_topic_arn = "arn:abcdef"
    create_alarm(instance_id, sns_topic_arn)
```

You can use the following steps to directly configure in the console:

Create CloudWatch Alarm

1. In **CloudWatch → Alarms → Create Alarm**.
2. Select metric → **SecurityMetrics / FailedLoginCount**.
3. Set condition:
    - **Threshold:** Greater than 5 within 5 minutes.
    - Statistic: **SampleCount**.

4. Notification:
    - Create new **SNS topic** (Simple Notification Service).
    - Add your email to receive alerts.
    - Confirm subscription from email.
5. Finish → Now if >5 failed logins happen, you'll get an email alert

Step 8: Create CloudWatch Dashboard

• Go to AWS Console → CloudWatch → Dashboards.
• Create a new dashboard.
• Add widgets such as CPU Utilization, Memory, Disk (if CloudWatch Agent is installed).

Step 9: Verify Alerts & Results

• Re-run the stress command to generate high CPU load.
• Observe CPUUtilization graph in the dashboard.
• Verify that an email alert is received when the threshold is crossed.

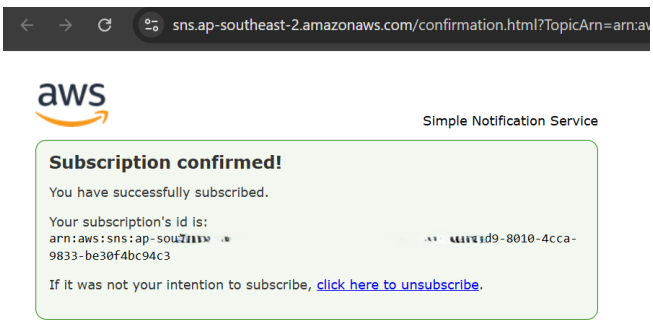
4. Results

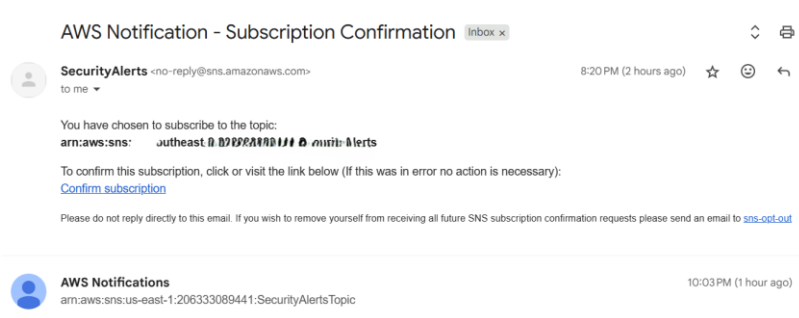
• CPUUtilization metrics displayed in the CloudWatch dashboard.
• CloudWatch Alarm triggered when CPU usage exceeded threshold.
• Email notification received from SNS.
• Dashboard successfully visualized system activity.

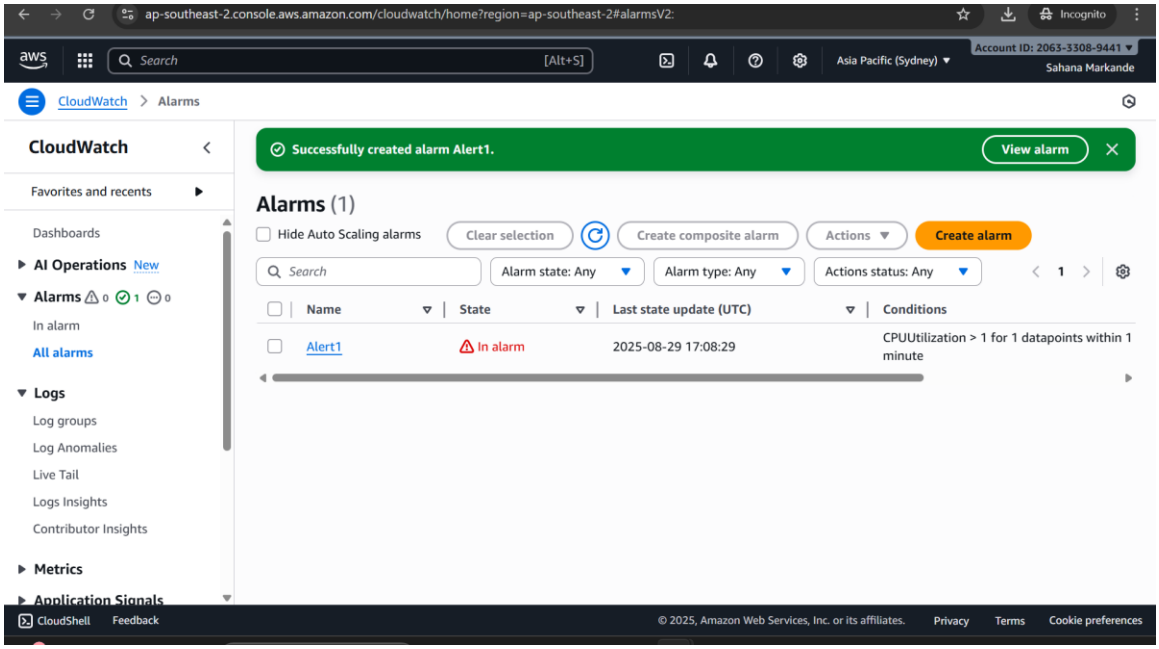
5. Conclusion

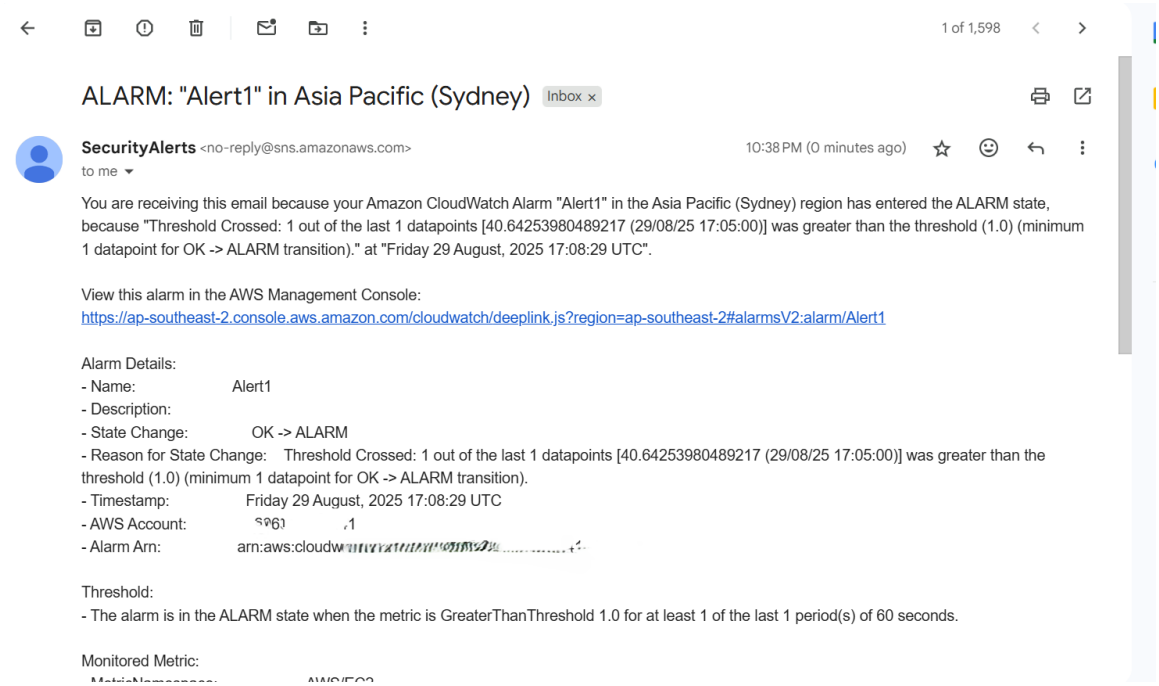
The experiment successfully demonstrated setting up a cloud monitoring and alerting system using AWS CloudWatch. We monitored EC2 instance CPU usage, configured alerts via SNS, visualized metrics using a dashboard, and validated the system by simulating high CPU usage with the stress tool.

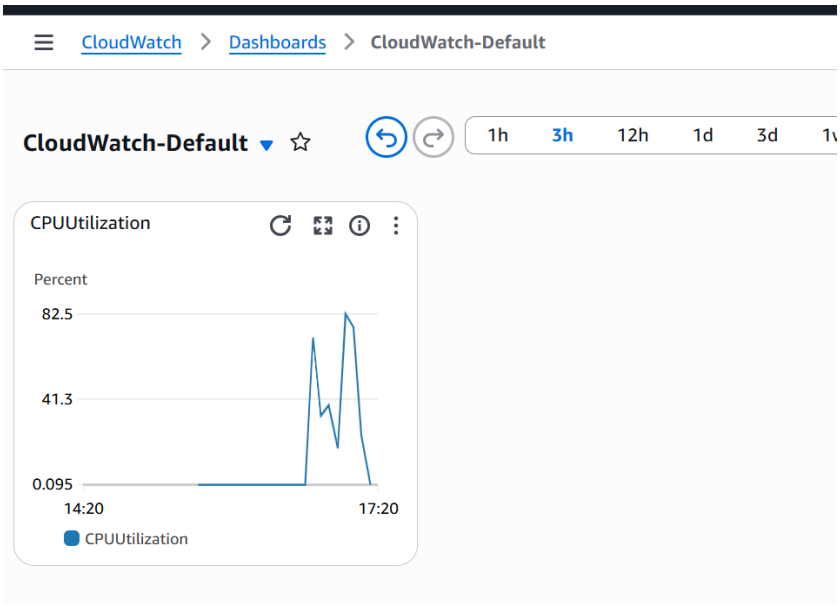
Sample Images/Output:






ap-southeast-2.console.aws.amazon.com/cloudwatch/home?region=ap-southeast-2#alarmsV2:
Incognito
Search
[Alt+S]
Asia Pacific (Sydney)
Account ID: 2063-3308-9441
Sahana Markande
CloudWatch
Alarms
CloudWatch
Successfully created alarm Alert1.
View alarm
Favorites and recents
Dashboards
AI Operations New
Alarms 0 1 0
In alarm
All alarms
Logs
Log groups
Log Anomalies
Live Tail
Logs Insights
Contributor Insights
Metrics
Application Signals
Alarms (1)
Hide Auto Scaling alarms
Clear selection
Create composite alarm
Actions
Create alarm
Search
Alarm state: Any
Alarm type: Any
Actions status: Any
Name
State
Last state update (UTC)
Conditions
Alert1
In alarm
2025-08-29 17:08:29
CPUUtilization > 1 for 1 datapoints within 1 minute
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


1 of 1,598
ALARM: "Alert1" in Asia Pacific (Sydney)
Inbox
SecurityAlerts <no-reply@sns.amazonaws.com>
10:38 PM (0 minutes ago)
to me
You are receiving this email because your Amazon CloudWatch Alarm "Alert1" in the Asia Pacific (Sydney) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [40.64253980489217 (29/08/25 17:05:00)] was greater than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Friday 29 August, 2025 17:08:29 UTC".
View this alarm in the AWS Management Console:
https://ap-southeast-2.console.aws.amazon.com/cloudwatch/deeplink.js?region=ap-southeast-2#alarmsV2:alarm/Alert1
Alarm Details:
- Name: Alert1
- Description:
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [40.64253980489217 (29/08/25 17:05:00)] was greater than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Friday 29 August, 2025 17:08:29 UTC
- AWS Account:
- Alarm Arn: arn:aws:cloudw
Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 1.0 for at least 1 of the last 1 period(s) of 60 seconds.
Monitored Metric:



Appendix:

```
# Check Python version
python --version

# Change directory to project
cd C:\Users\Path\VSCode_AWS_Monitoring

# Activate virtual environment
.\venv\Scripts\Activate

# Configure AWS CLI (first attempt - wrong keys)
aws configure
AWS Access Key ID [None]: 123445
AWS Secret Access Key [None]: abcd
Default region name [None]: Asia Pacific (Sydney)
Default output format [None]: json

# Run SNS topic script
python create_sns_topic.py

# Create CloudWatch alarm
python create_cloudwatch_alarm.py

# Connect to EC2
ssh -i Pair1.pem ec2-user@<EC2-Public-IP>

sudo dnf install stress -y

stress --cpu 2 --timeout 300
```