CloudWatch Logging with Python and AWS

Introduction

In this project, I have developed a logging solution that integrates with AWS CloudWatch. This solution sends log data to AWS CloudWatch Logs using Python's watchtower library. The logs are formatted in JSON format, providing detailed information such as timestamp, log level, message, file name, line number, and function name.

Prerequisites

- **AWS Account**: To use AWS services like CloudWatch.
- **Python**: I used Python to write the script.

Install Required Libraries

To install the required libraries, I ran the following commands:

```
pip install boto3 watchtower
```

AWS Setup

1.Create an IAM User for Programmatic Access:

1.First, I created an IAM (Identity and Access Management) user in the AWS console with **Programmatic Access** enabled.

2.I ensured that the user had sufficient permissions to write logs to CloudWatch.

2.Attach Policies to the IAM User:

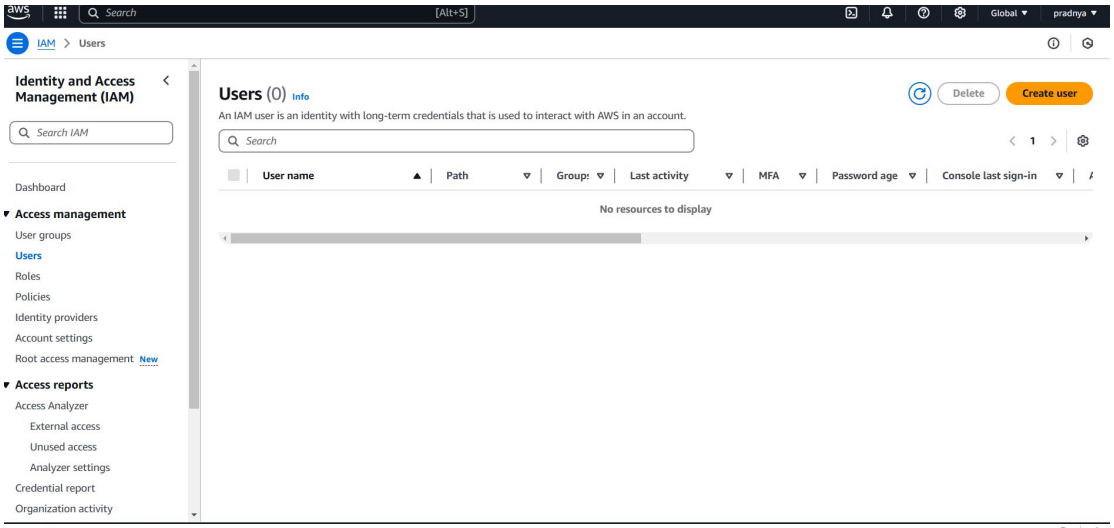
1.I attached the CloudWatchLogsFullAccess policy to the IAM user. This policy grants full access to AWS CloudWatch Logs.

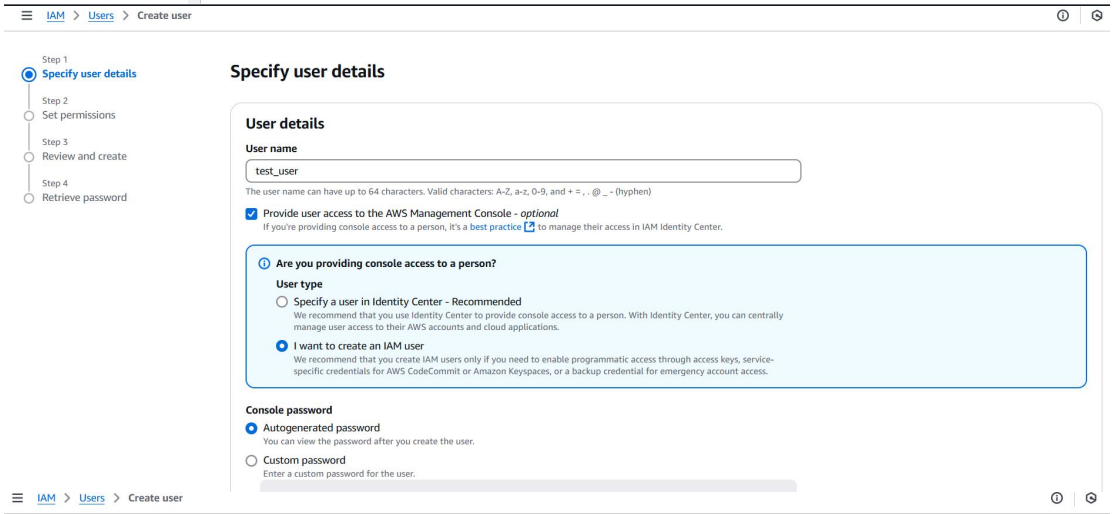
2.To attach the policy, I navigated to **IAM > Users > [username] > Permissions > Add Permissions**, then selected the ***CloudWatchLogsFullAccess*** policy.

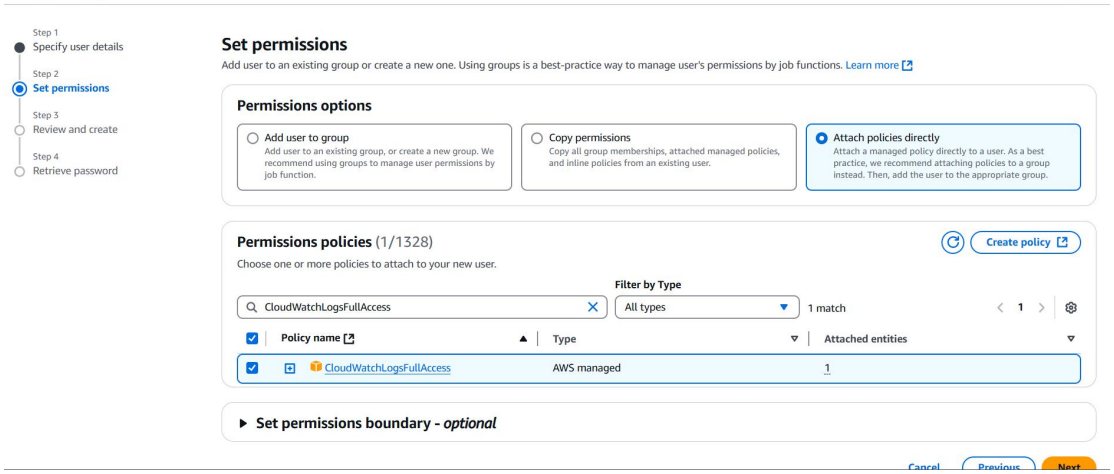
3.Generate AWS Access Keys:

1.After creating the IAM user, I generated **Access Key ID** and **Secret Access Key**.

2.I downloaded the .csv file containing the keys or saved them for later use in the code.







IAM > Users > Create user

Specify user details

Step 2
Set permissions

Step 3
Review and create

Step 4
Retrieve password

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
test_user	Custom password	Yes

Permissions summary

Name	Type	Used as
CloudWatchLogsFullAccess	AWS managed	Permissions policy
IAMUserChangePassword	AWS managed	Permissions policy

Tags - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel | Previous | Create user

Identity and Access Management (IAM)

Search IAM

Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report
Organization activity

test_user Info

Delete

Summary

ARN	Console access	Access key 1
arn:aws:iam::039612863156:user/test_user	Enabled without MFA	Create access key
Created February 14, 2025, 22:30 (UTC+05:30)	Last console sign-in: Never	

Permissions | Groups | Tags | Security credentials | Last Accessed

Console sign-in

Manage console access

Console sign-in link
https://039612863156.signin.aws.amazon.com/console

Console password
Updated Now (2025-02-14 22:30 GMT+5:30)

Last console sign-in
Never

Multi-factor authentication (MFA) (0)

Remove | Resync | Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. Learn more

Type	Identifier	Certifications	Created on

IAM > Users > test_user

Identity and Access Management (IAM)

Search IAM

Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
External access
Unused access
Analyzer settings
Credential report

Multi-factor authentication (MFA) (0)

Remove | Resync | Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. Learn more

Type	Identifier	Certifications	Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Assign MFA device

Access keys (0)

Create access key

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

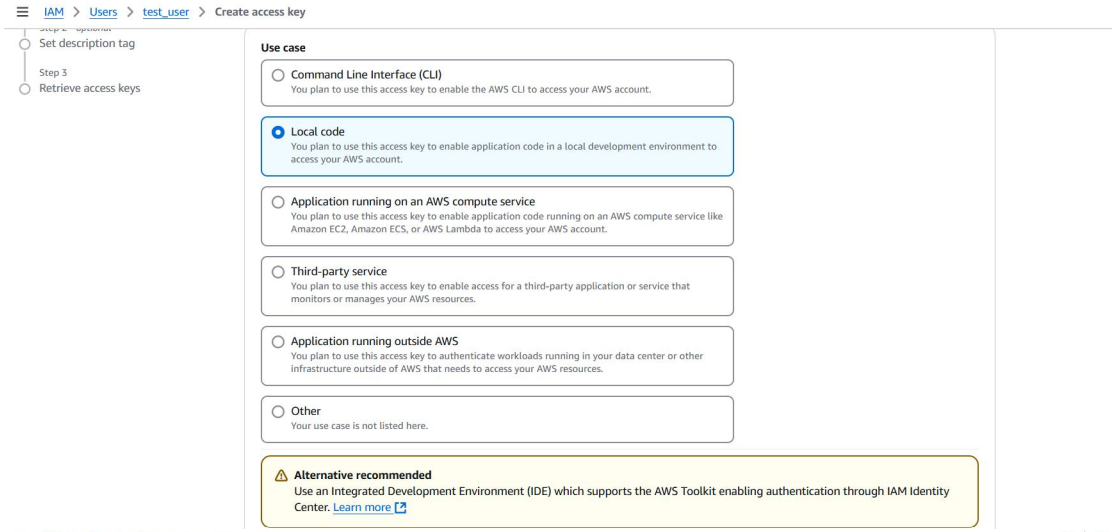
Create access key

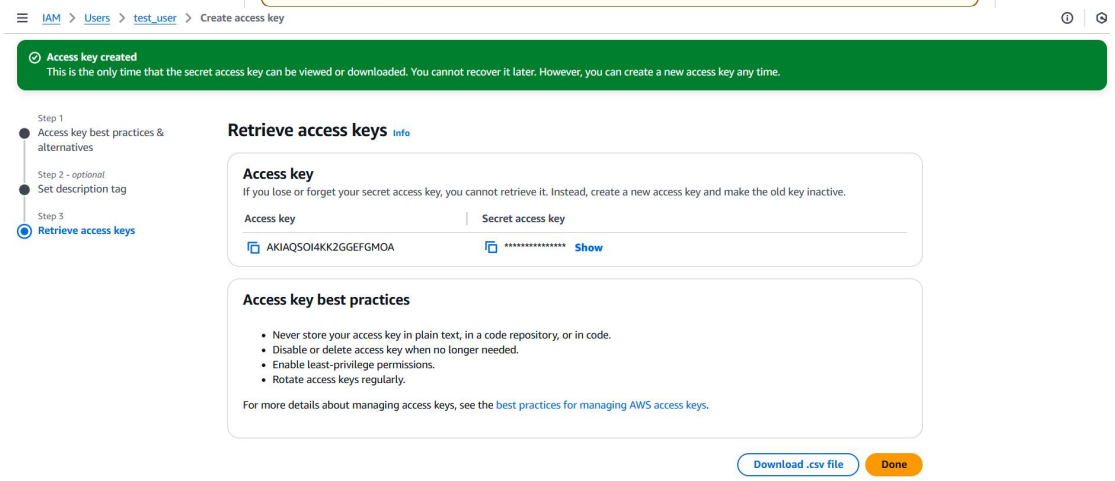
SSH public keys for AWS CodeCommit (0)

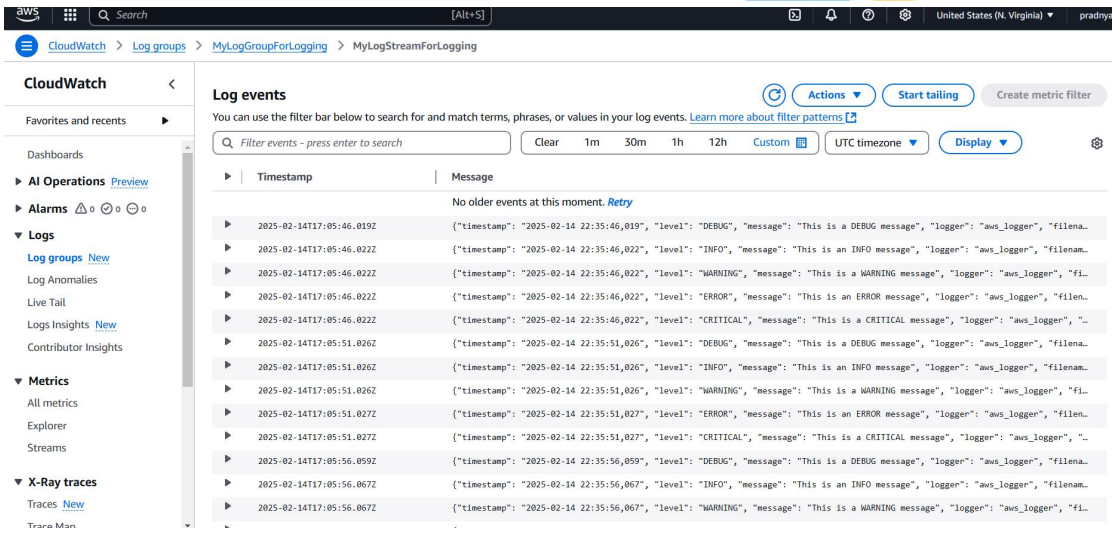
Actions | Upload SSH public key

User SSH public keys to authenticate access to AWS CodeCommit repositories. You can have a maximum of five SSH public keys (active or inactive) at a time. Learn more

SSH Key ID	Uploaded	Status







Python Script: CloudWatch Logging

The Python script is used to send logs to AWS CloudWatch, formatted as JSON.

Code Explanation

1. **boto3 client**: I created a boto3 client to communicate with AWS CloudWatch using the credentials provided.
2. **Logger**: I initialized the logger and set the logging level to DEBUG to capture all log levels (DEBUG, INFO, WARNING, ERROR, CRITICAL).
3. **Custom JSON Formatter**: I created a custom JSONFormatter class that formats the logs as JSON, including timestamp, log level, message, logger name, filename, line number, and function name.
4. **Log Group and Log Stream Creation**: Using boto3, I created the log group and log stream programmatically .
5. **CloudWatch Log Handler**: I configured the watchtower.CloudWatchLogHandler to send logs to the AWS CloudWatch log group and stream.
6. **Logging Loop**: I used an infinite loop (while True) to log messages continuously every 5 seconds.

Conclusion

In this project, I successfully created a logging system using AWS CloudWatch, utilizing a Python script to send JSON-formatted logs. The logs are continuously sent to CloudWatch, providing a detailed record of events. This logging solution ensures that I can easily monitor and troubleshoot the application by centralizing logs in AWS CloudWatch.