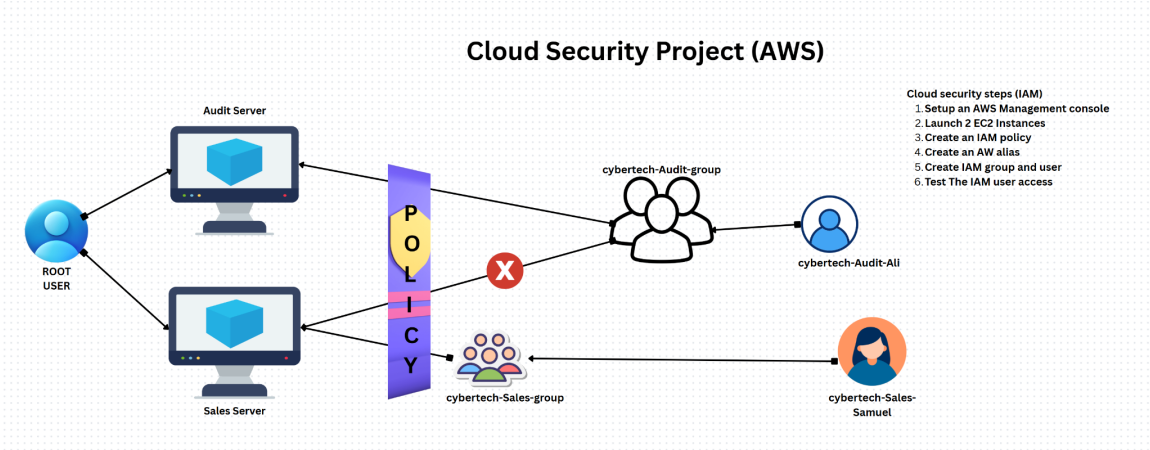
AWS IAM Cloud Security Project

1. Project Overview

Recently, completed a project focused on cloud security controls using Amazon Web Services (AWS), with an emphasis on Identity and Access Management (IAM). The objective was to design and implement a least-privilege policy, attach it to a specific user group, and ensure it effectively limited access to only the necessary actions. To test the policy, I applied it to two Amazon EC2 instances—**audit** and **sales**—and confirmed that the restrictions worked as intended.



2. Tools & Concepts

Throughout the project, I worked with several key AWS services and concepts, including:

AWS IAM: Managing users, groups, and policies, and setting a custom account alias

Amazon EC2: Tagging instances and managing their lifecycle actions

JSON policy syntax: Defining permissions using Effect, Action, and Resource

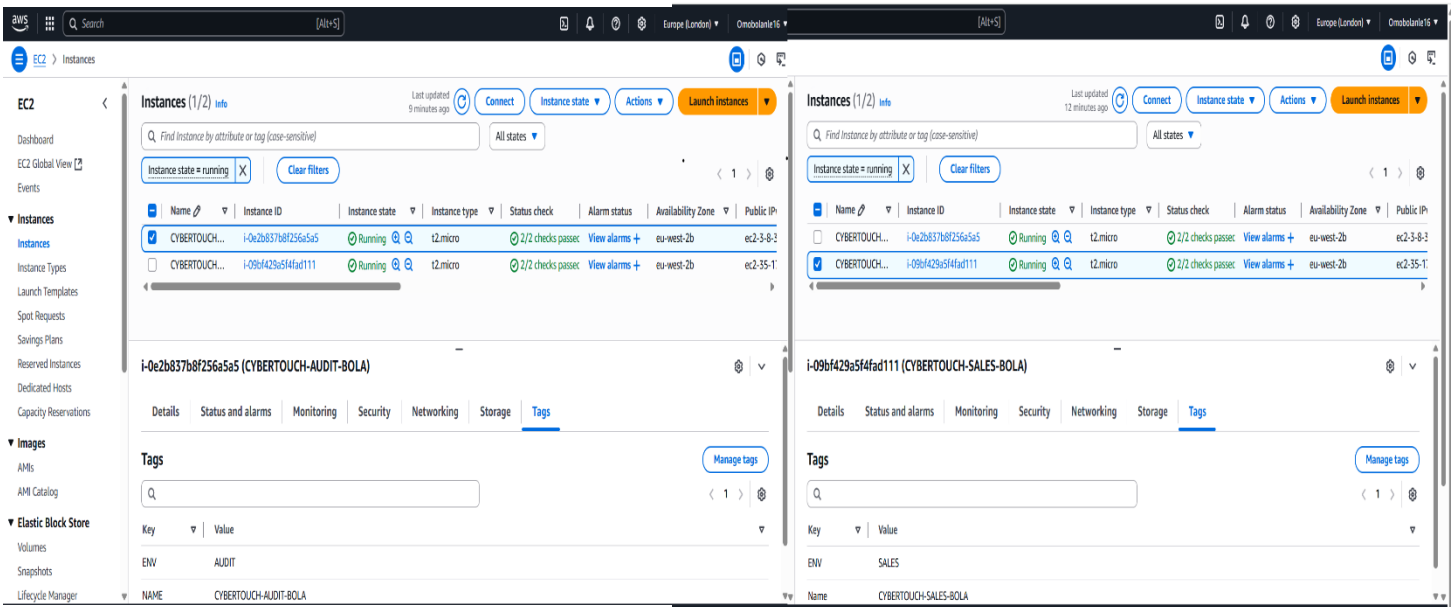
Policy testing: Verifying that the least privileged permissions behaved as expected

This hands-on experience deepened my understanding of secure access control in cloud environments and reinforced the importance of precise permission management in AWS.

3. **Tagging Strategy**

The objective was to create a custom IAM policy that restricts user actions based on roles and attach it to a specific user group. I then tested this policy on two **Amazon EC2** instances—tagged as **Audit** and **Sales**—to verify that permissions were enforced appropriately.

To clearly distinguish the instances and apply context-based permissions, I used descriptive tags:



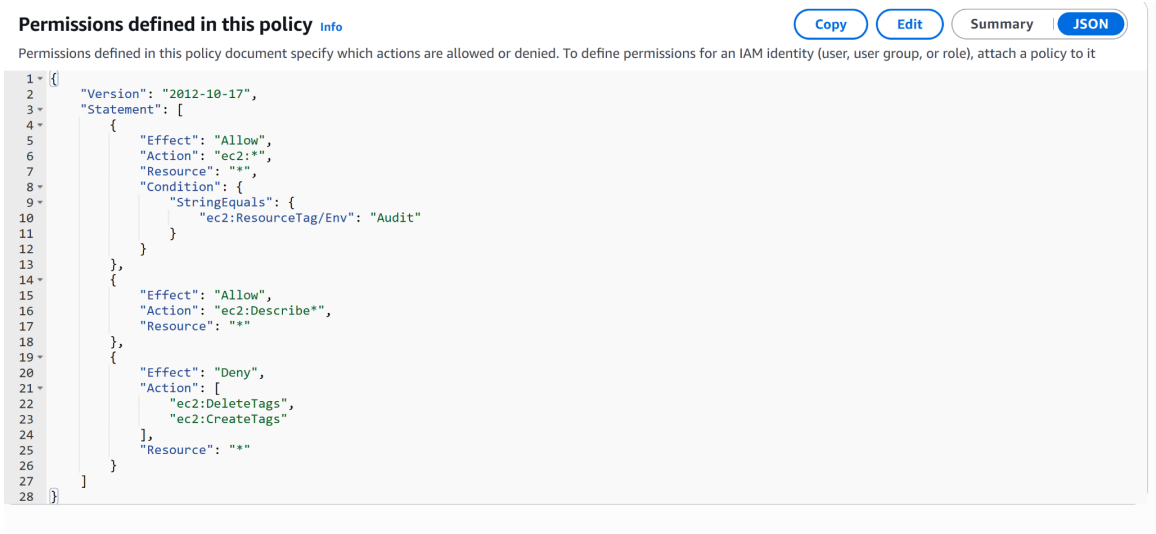
4. Creating the IAM Policy

I authored a **custom JSON IAM policy** to enforce role-based access controls on EC2 instances, applying the **principle of least privilege**. Specifically, the policy was designed to:

Block Stop Instances and Start instances actions on the **Audit** server

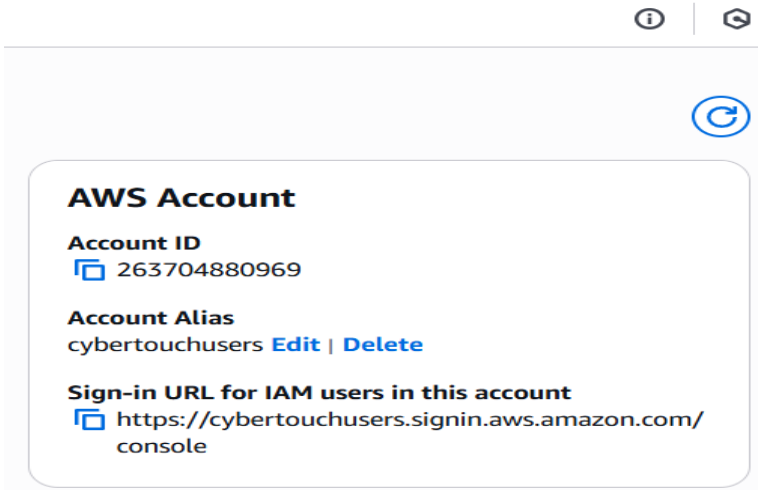
Allow the same actions on the **Sales** server

This was achieved by using condition-based logic that references EC2 instance **tags** (Environment: Audit vs. Environment: Sales) to fine-tune permissions. The approach ensured that users in the assigned IAM group could only manage EC2 instances intended for their role, improving operational security while maintaining usability.



5. Account Alias

Additionally, I configured a custom AWS **account alias** to replace the default numeric URL, streamlining access to the AWS Console and improving the overall user experience for the team.



6. IAM Users & Groups

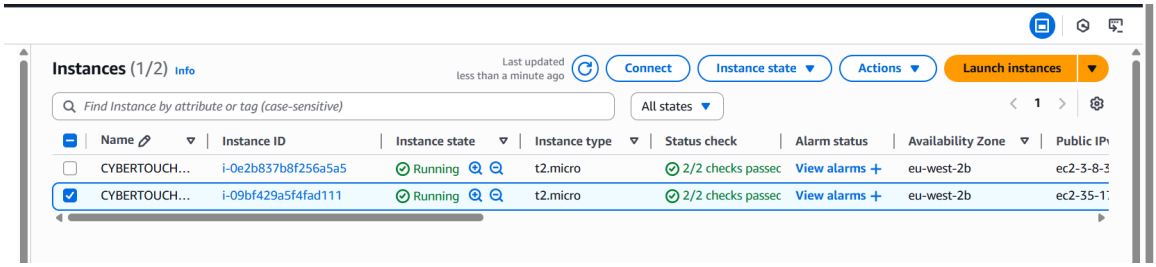
To implement secure access controls, I carried out the following IAM configurations:

Created an IAM user group named **Developers** to manage permissions collectively.

Attached a custom policy (CybertouchAuditEnvPolicy) to the group, enforcing least-privilege access based on EC2 instance tags.

Added individual IAM users to the group granting them controlled access to EC2 resources according to their operational needs.

This setup streamlined permission management while ensuring that users could only perform approved actions aligned with their responsibilities.

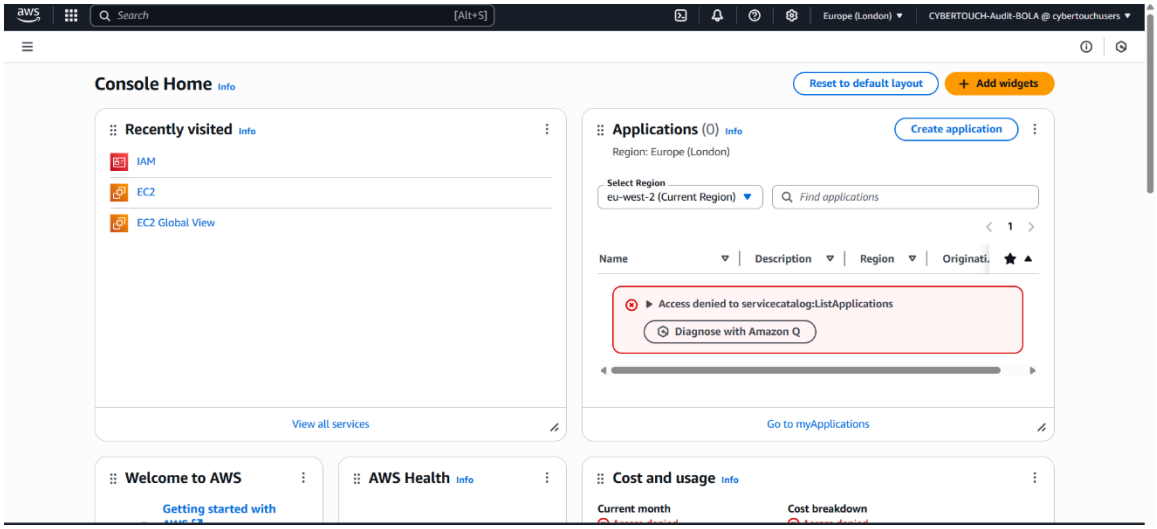


7. Logging in as an IAM User

IAM users can access AWS in two main ways:

Via the AWS Management Console, using the custom account alias for a more user-friendly login experience

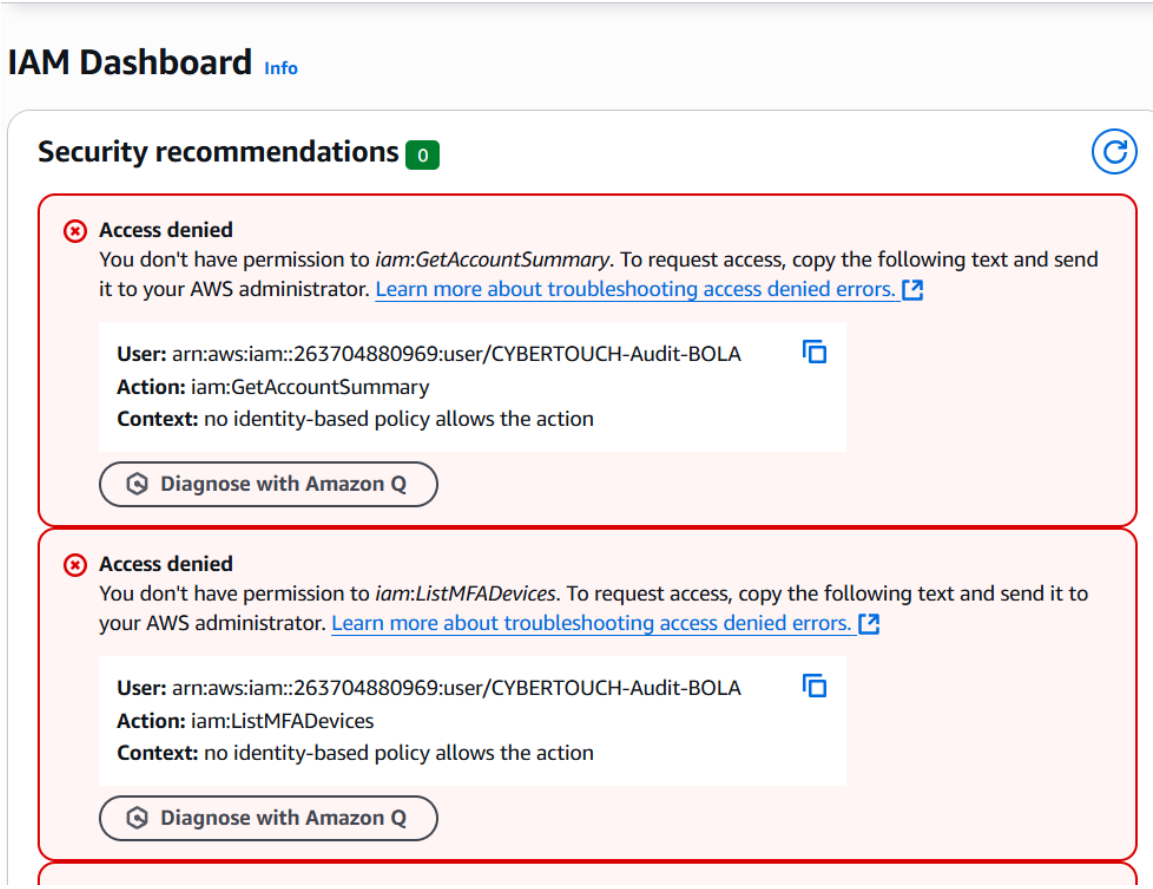
Via the AWS Command Line Interface (CLI), using securely generated **programmatic access keys**

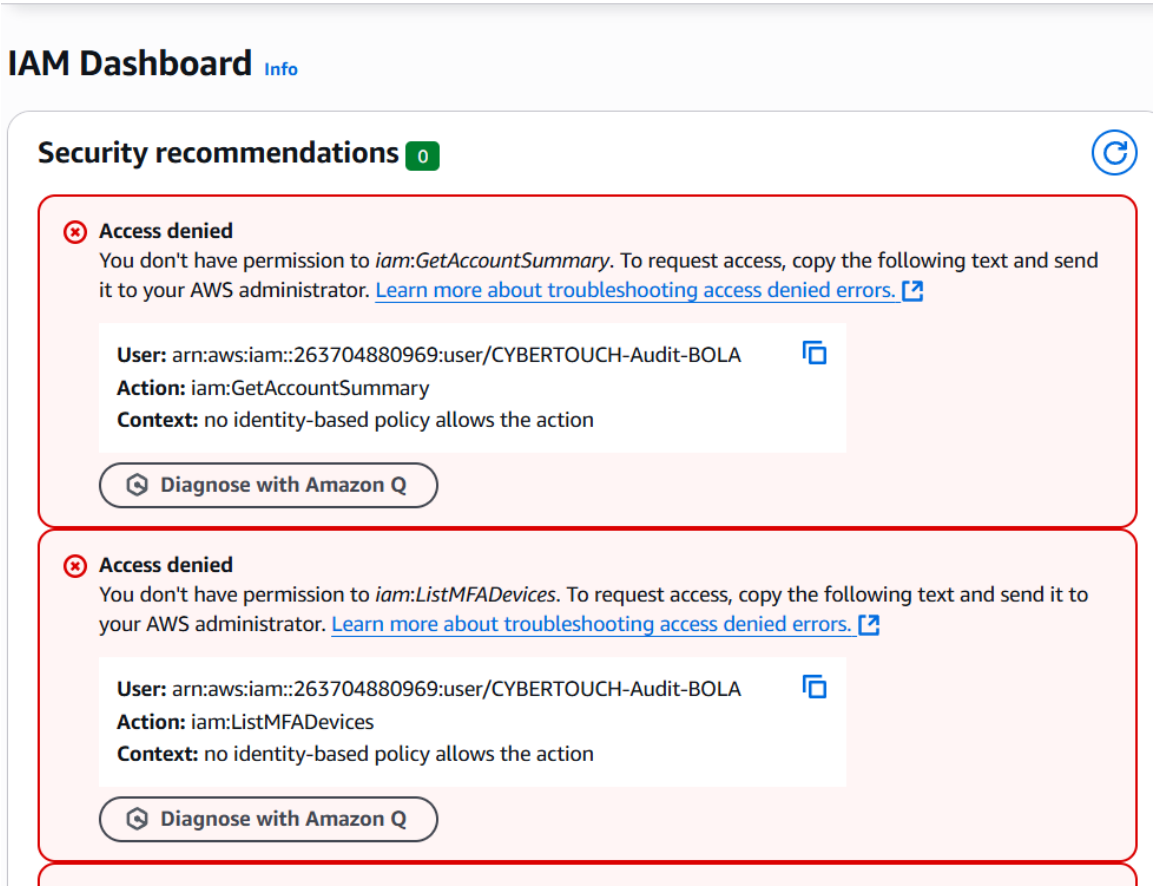


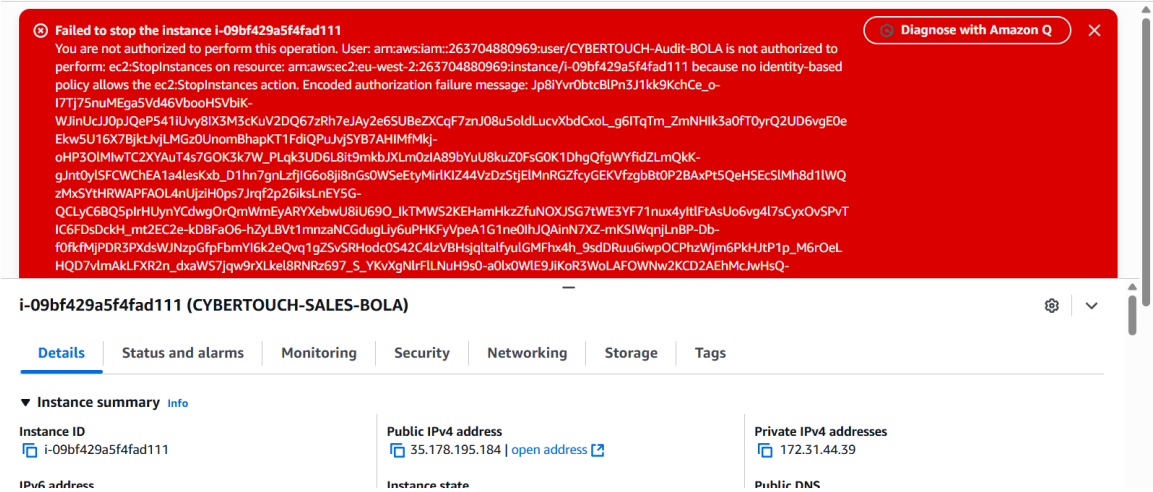
8. Testing the Policy

Test Action	Expected Result	Actual Result
Stop Audit Instance	Denied	Access denied error displayed
Stop Sales Instance	Allowed	Instance stopped successfully

Test Action	Expected Result	Actual Result
Start Audit Instance	Denied	Access denied error displayed
Start Sales Instance	Allowed	Instance started successfully




IAM Dashboard Info
Security recommendations 0
Access denied
You don't have permission to iam:GetAccountSummary. To request access, copy the following text and send it to your AWS administrator. Learn more about troubleshooting access denied errors.
User: arn:aws:iam::263704880969:user/CYBERTOUCH-Audit-BOLA
Action: iam:GetAccountSummary
Context: no identity-based policy allows the action
Diagnose with Amazon Q
Access denied
You don't have permission to iam:ListMFADevices. To request access, copy the following text and send it to your AWS administrator. Learn more about troubleshooting access denied errors.
User: arn:aws:iam::263704880969:user/CYBERTOUCH-Audit-BOLA
Action: iam:ListMFADevices
Context: no identity-based policy allows the action
Diagnose with Amazon Q


Failed to stop the instance i-09bf429a5f4fad111
You are not authorized to perform this operation. User: arn:aws:iam::263704880969:user/CYBERTOUCH-Audit-BOLA is not authorized to perform: ec2:StopInstances on resource: arn:aws:ec2:eu-west-2:263704880969:instance/i-09bf429a5f4fad111 because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure message: Jp8iYvr0btcBlPn3J1kk9KchCe_o-
Diagnose with Amazon Q
i-09bf429a5f4fad111 (CYBERTOUCH-SALES-BOLA)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-09bf429a5f4fad111
Public IPv4 address
35.178.195.184 | open address
Private IPv4 addresses
172.31.44.39

Project Summary – IAM Policy Verification (Least Privilege Access Test)

This project was conducted to validate the implementation of a least privilege IAM policy for an IAM user within AWS. The test focused on ensuring that the user could only perform EC2 instance operations on authorized resources.

Test Scenarios & Outcomes:

The IAM user attempted to **start and stop** EC2 instances tagged as Environment: Audit and Environment: Sales.

All actions targeting the **Audit instance were correctly denied**, confirming restricted access.

All actions targeting the **Sales instance were successfully executed**, confirming appropriate permission was granted.

Conclusion:

The IAM policy performed as expected, enforcing the principle of least privilege by restricting access to sensitive resources while allowing necessary operational access to permitted instances.