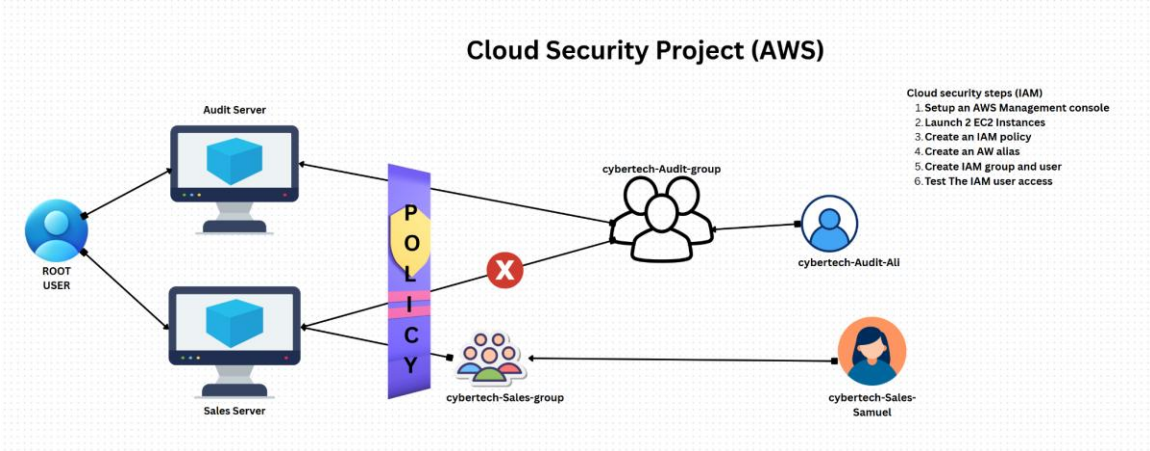
AWS IAM Cloud Security Project

1. Project Overview

The goal of this project is to apply cloud security best practices on Amazon Web Services AWS. Focusing on Identity and Access Management (IAM). This involves setting up a secure environment, launching instances, creating user roles and policies, and testing access control, all while ensuring compliance with the principle of least privilege, attach it to a user group, and verify that the policy correctly restricts actions on two Amazon EC2 instances (audit and sales).



2. Tools & Concepts

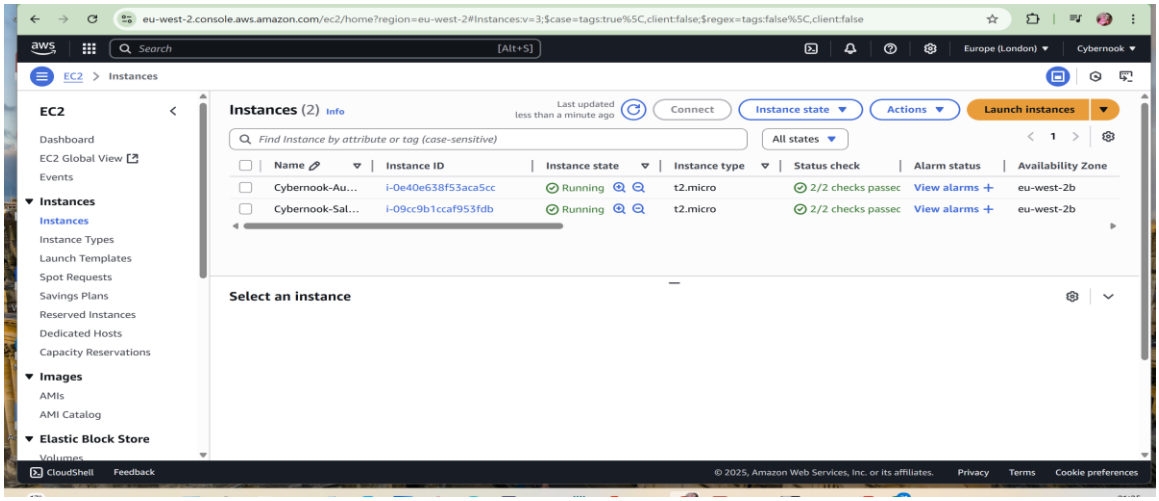
- AWS IAM – users, groups, policies, account alias
- Amazon EC2 – instance tagging and lifecycle actions
- JSON policy syntax – Effect, Action, Resource
- Principle of least privilege and policy testing

3. Tagging Strategy

To enhance resource identification and management within AWS, I applied descriptive tags to each EC2 instance based on their functional roles. This helps improve visibility, streamline cost tracking, and enforce policy controls.

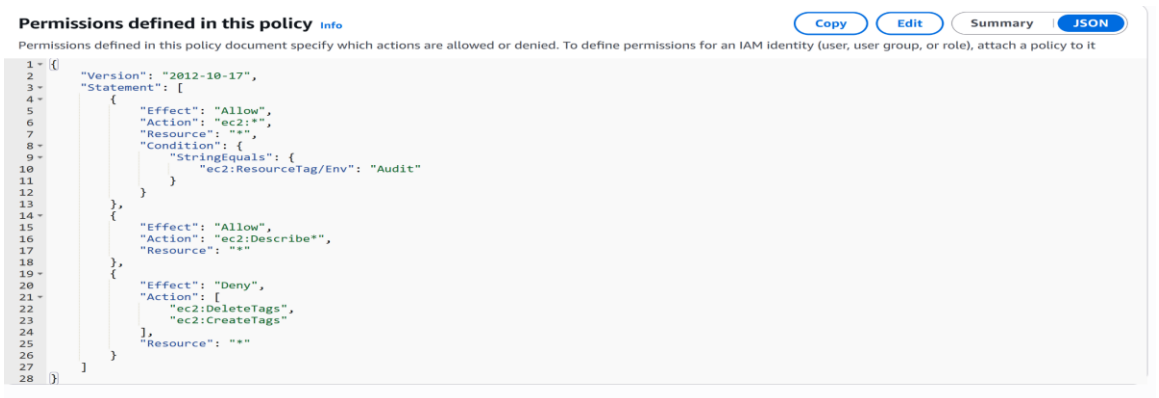
INSTANCES	TAG KEY	TAG VALUE
Audit	Environment	Audit
Sales	Environment	Sales

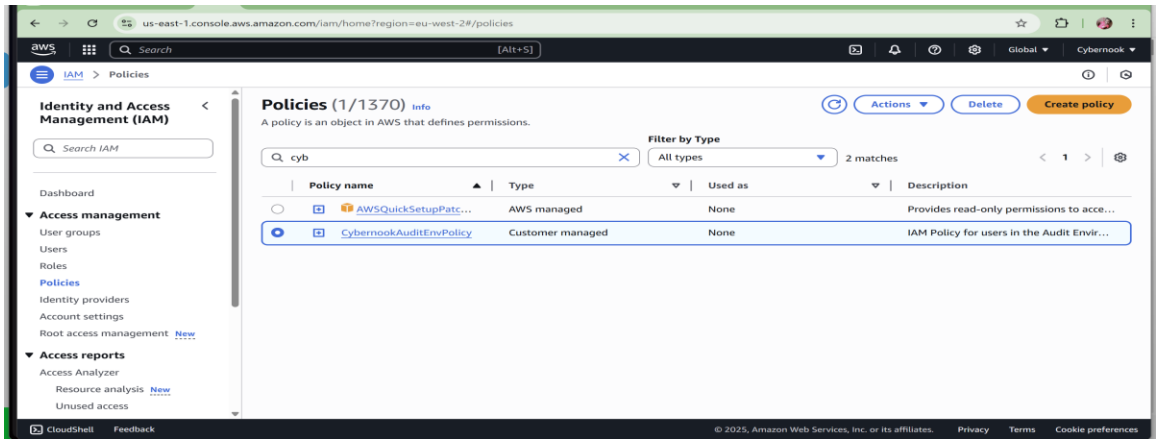
Each tag was applied using the **AWS Management Console**, ensuring that team members can quickly recognize the purpose of each instance at a glance.



4. Creating the IAM Policy

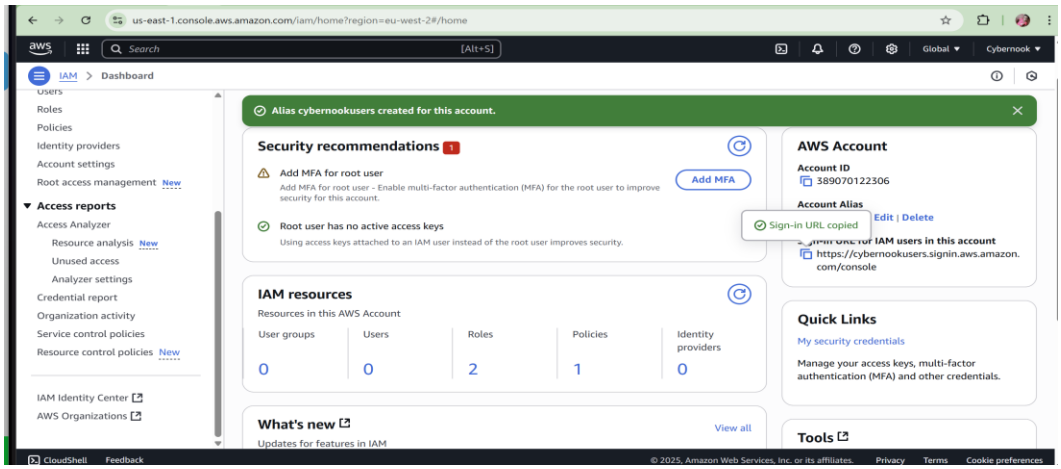
Created a **custom IAM policy** with limited access permissions: following JSON policy to block instance stop/start actions on the audit server but allow those actions on the sales server:





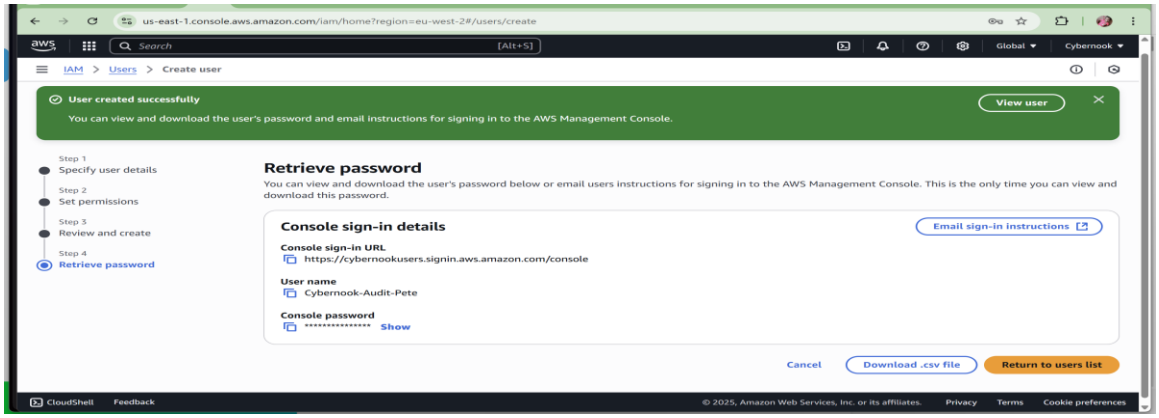
5. Account Alias

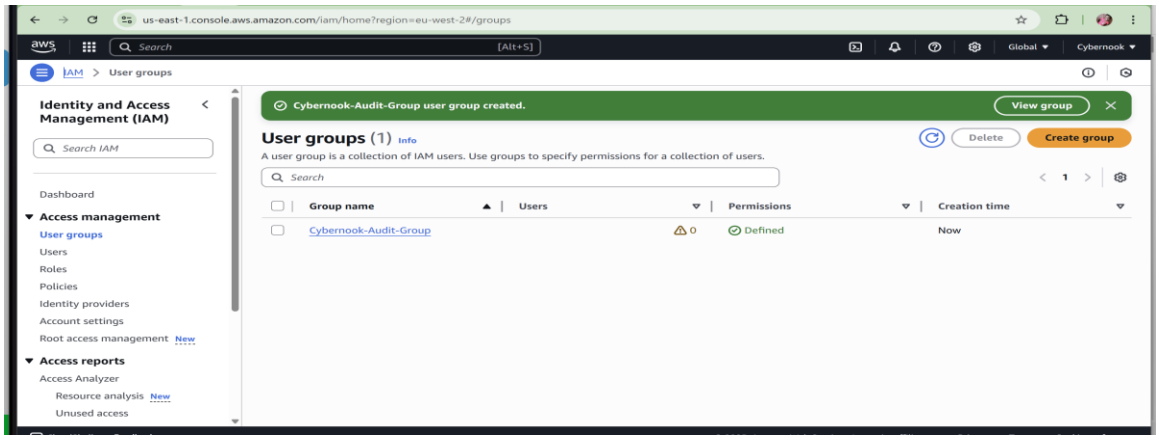
Set up a custom **AWS account alias** (e.g., *secure-team-console*) for easy login and branding and URL format changed from the default



6. IAM Users & Groups

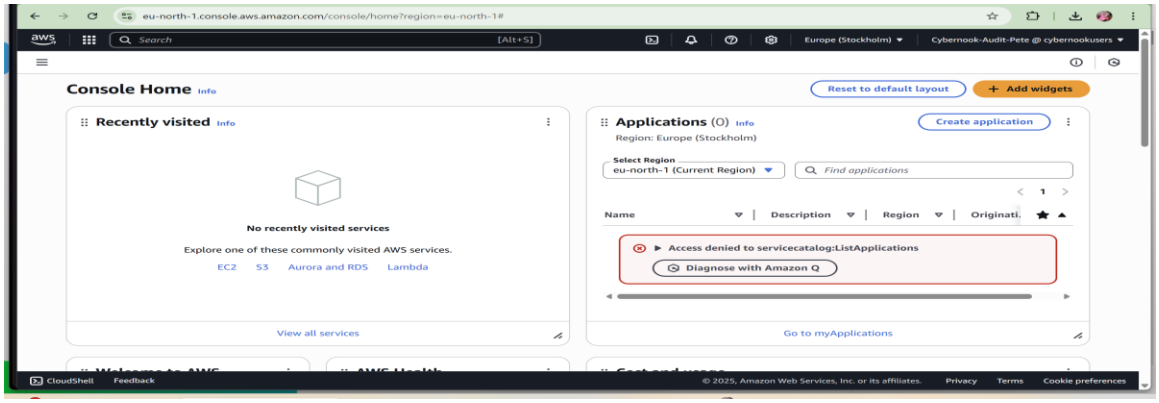
1. Created an IAM user group called Developers.
2. Attached the **CybertnookAuditEnvPolicy** policy to the group.
3. Added individual IAM users who require controlled EC2 access.





7. Test the IAM User Access

- Logged on to AWS Management Console using the custom alias URL.
- Verified the user could:
    - Access EC2 read-only (e.g., view instances, but not start/stop them).
- Attempted unauthorized actions (e.g., EC2 termination) to confirm policy enforcement access was denied as expected.



8. Test the Policy

Test Action	Expected Result	Actual Result
Stop audit instance	Denied	Access denied error displayed
Stop sales instance	Allowed	Instance stopped successfully
Start audit instance	Denied	Access denied error displayed
Start sales instance	Allowed	Instance started successfully