Cloud Security with AWS IAM

Introduction to the Project

In this project, I demonstrate my practical knowledge of cloud security by implementing Identity and Access Management (IAM) on Amazon Web Services (AWS). The goal is to control access and define permissions for different users interacting with AWS resources, specifically within development and production environments.

The services used for this project include:

- Amazon EC2 (Elastic Compute Cloud)
- AWS IAM (Identity and Access Management)

The key concepts and tools I explored are:

- IAM users and user groups
- IAM policies (including JSON policy structure)
- Account alias configuration
- AWS tagging for organisation and policy control
- IAM policy simulator
- Launching, tagging, and managing EC2 instances
- Logging in as IAM users and testing policies

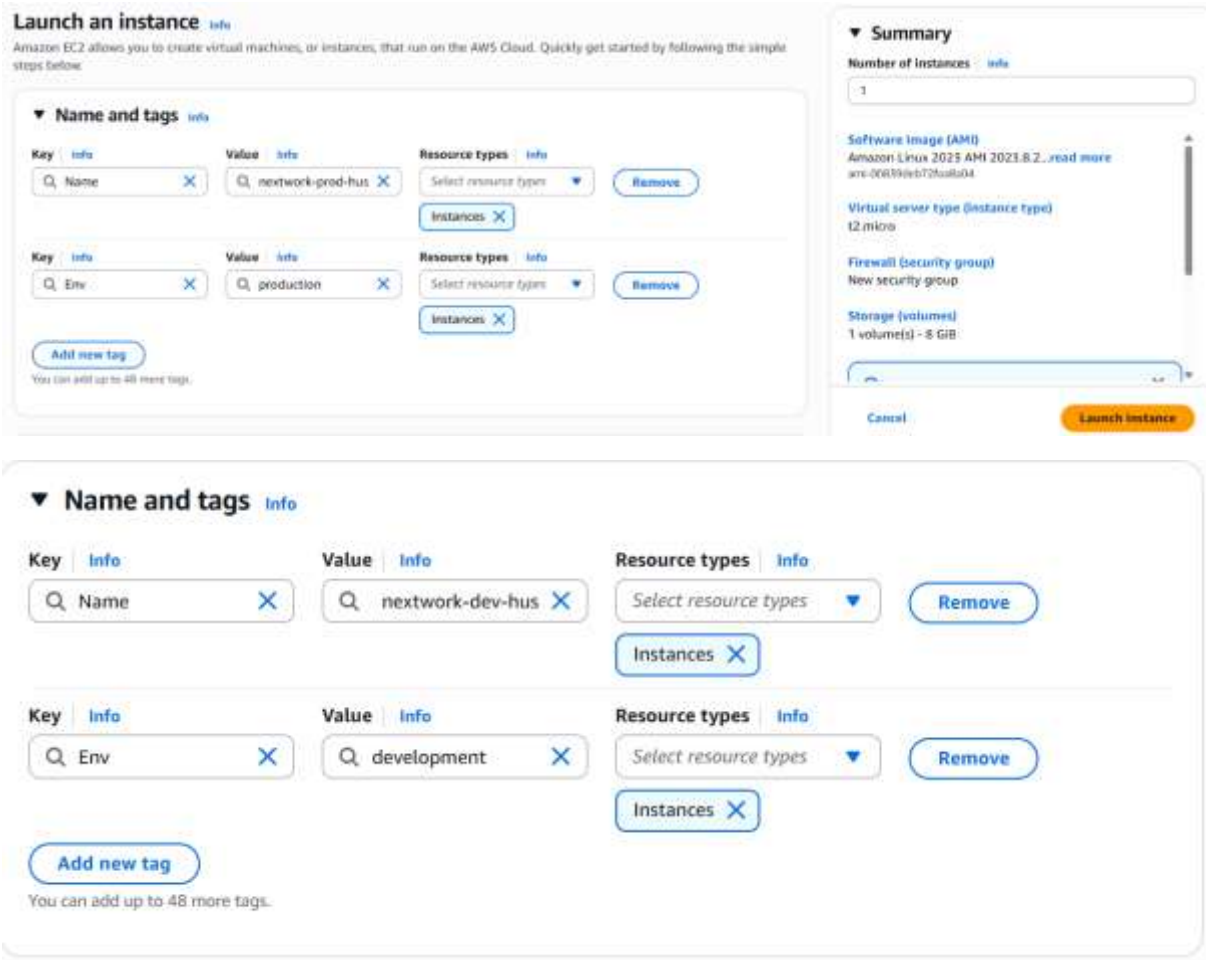
This hands-on experience helped me understand real-world practices in cloud security, especially how to follow the principle of least privilege and separate production from development workflows using industry standards.

Tags: Organising Resources with Purpose

Tags in AWS are key-value pairs attached to resources that improve organisation and management. They are often used for:

- Grouping and categorising resources
- Enabling cost tracking by project or environment
- Applying IAM policies to resource groups dynamically

In this project, I used the tag Env (short for "Environment") with values development and production to distinguish instances. This made it easier to apply access controls to the correct environments and is in line with industry best practices for managing infrastructure.



IAM Policies: Defining Access Control

IAM Policies are the backbone of AWS access control. They are written in JSON and define who can do what with which resources, using three core components:

- **Effect** – Allow or Deny
- **Action** – The operations the policy permits or restricts (e.g., ec2:StartInstances)
- **Resource** – The specific AWS resource(s) the policy applies to

My Custom IAM Policy

I created a custom JSON IAM policy with the following rules:

- The policy allows users in the development group to perform any action on instances tagged with Env=development.
- They are allowed to view all instances**,** regardless of their tags.
- However, they cannot modify tags on any instance.

This structure ensures a secure separation between environments, while still giving developers the flexibility they need in the development environment.

Sample JSON Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Env": "development"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "ec2:DeleteTags",
        "ec2:CreateTags"
      ],
      "Resource": "*"
    }
  ]
}
```

IAM Users and User Groups

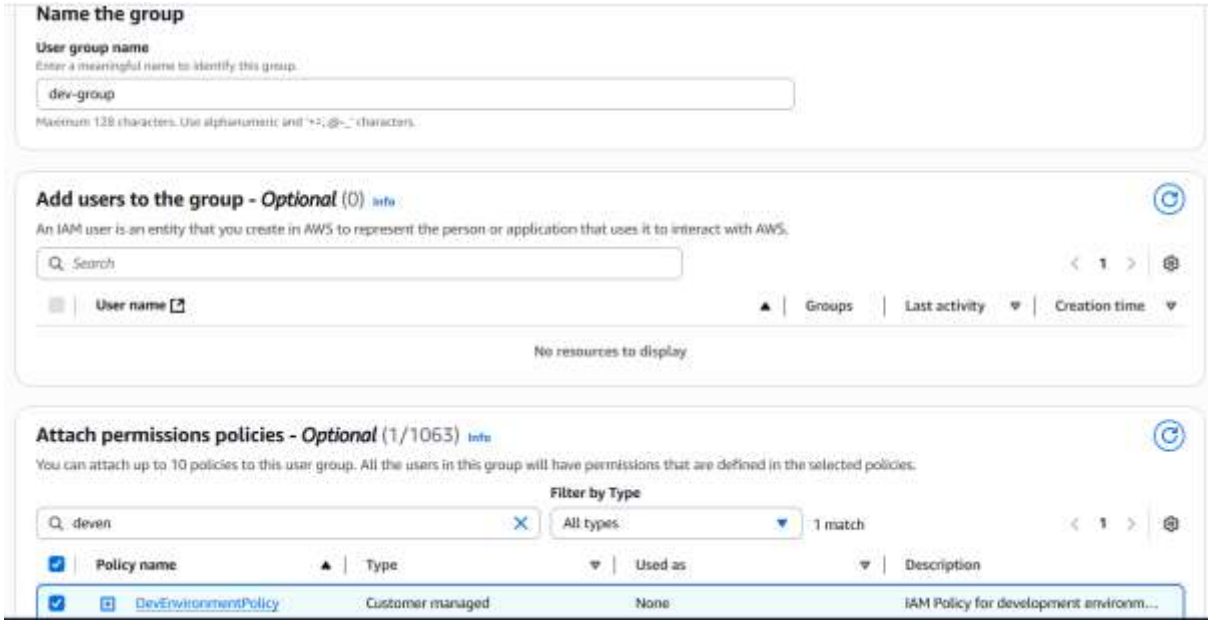
IAM Users

IAM users are individual entities who can sign in to the AWS Management Console or interact with AWS via the CLI or SDKs. Each user has their own credentials.

IAM User Groups

IAM user groups allow administrators to assign permissions at scale. Instead of applying policies to individual users, I attached my custom IAM policy to a user group called dev-group. Any user added to this group automatically inherits the associated permissions.

This is efficient, scalable, and mirrors real-world organisational IAM structures.

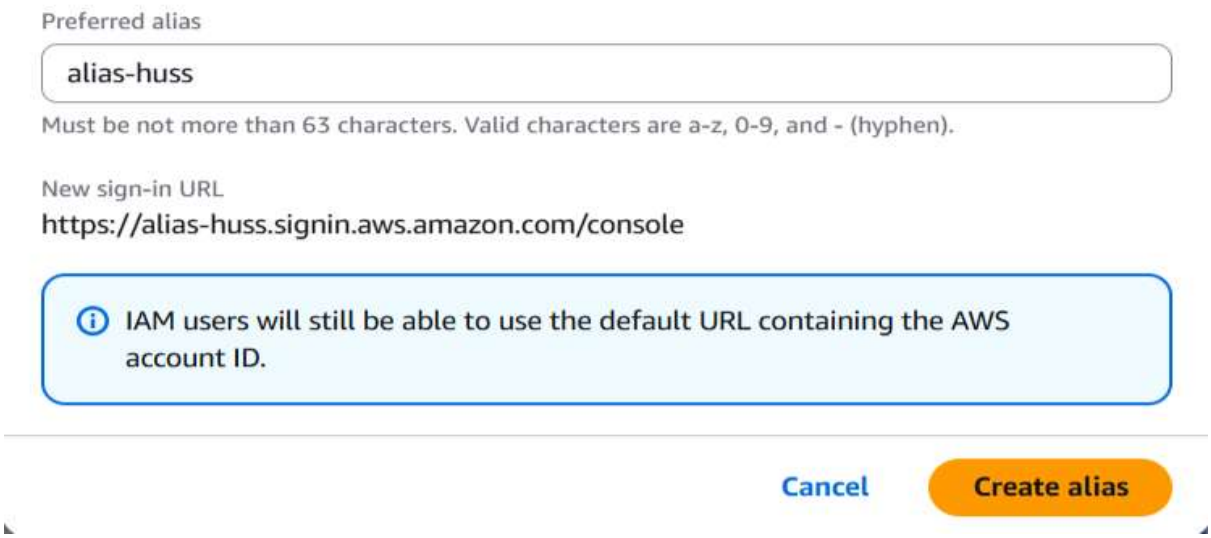


Creating an Account Alias

An account alias is a custom, human-readable name for your AWS account that replaces the default numeric account ID in sign-in URLs.

I created a temporary alias which is now deleted, and it took less than a minute and immediately made logging in easier. For example, the sign-in URL became:

This improves usability, particularly for IAM users who may not be familiar with AWS account IDs.

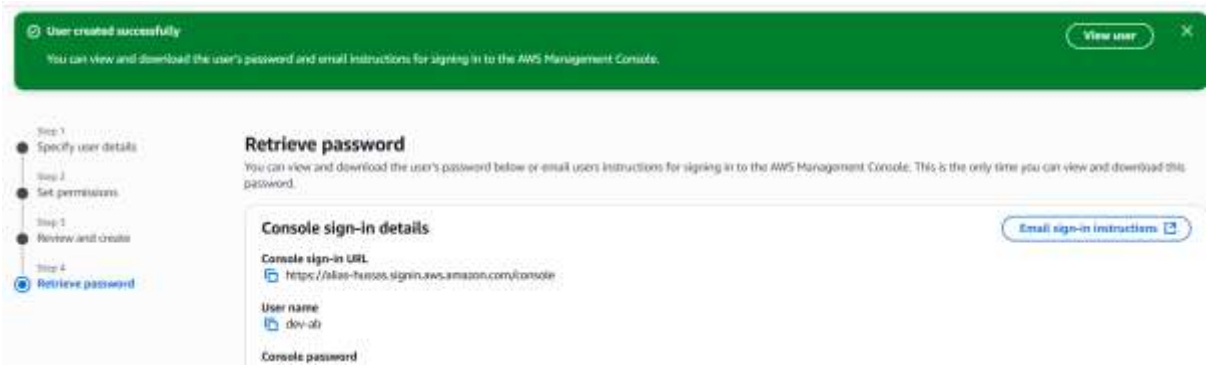


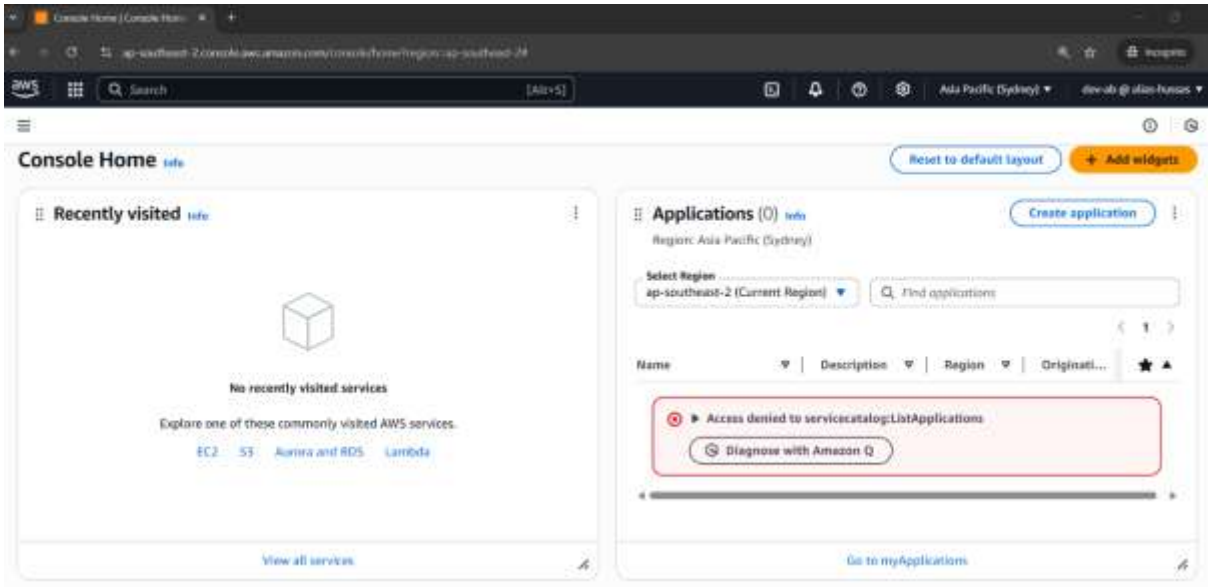
Logging In as an IAM User

I explored two methods of IAM user onboarding:

1. Sending sign-in instructions via email
2. Downloading a .csv file containing login credentials

Upon logging in as an IAM user, I observed restricted access across the AWS Console. This was expected, the user's permissions were explicitly limited to interacting with EC2 instances tagged with Env=development. This reflects principle of least privilege, a key concept in security.

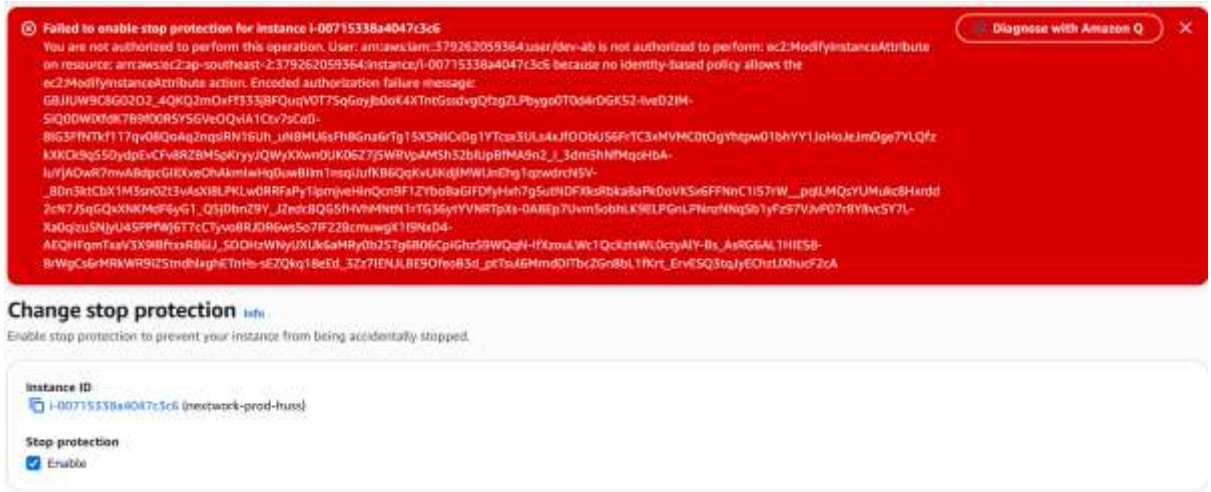




Testing IAM Policies

Stopping the Production Instance

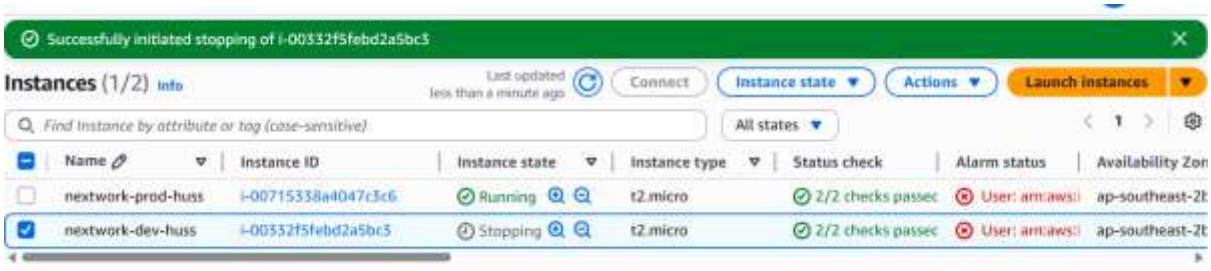
As a test, I attempted to stop an EC2 instance tagged as production. As expected, I received a permission denied error. This confirmed that the policy correctly blocked access to production resources.



Stopping the Development Instance

I then stopped an instance tagged with Env=development, and the operation succeeded. This verified that the IAM policy allowed full access to development resources.

This level of policy testing is critical to ensure access controls work as intended before real deployment.

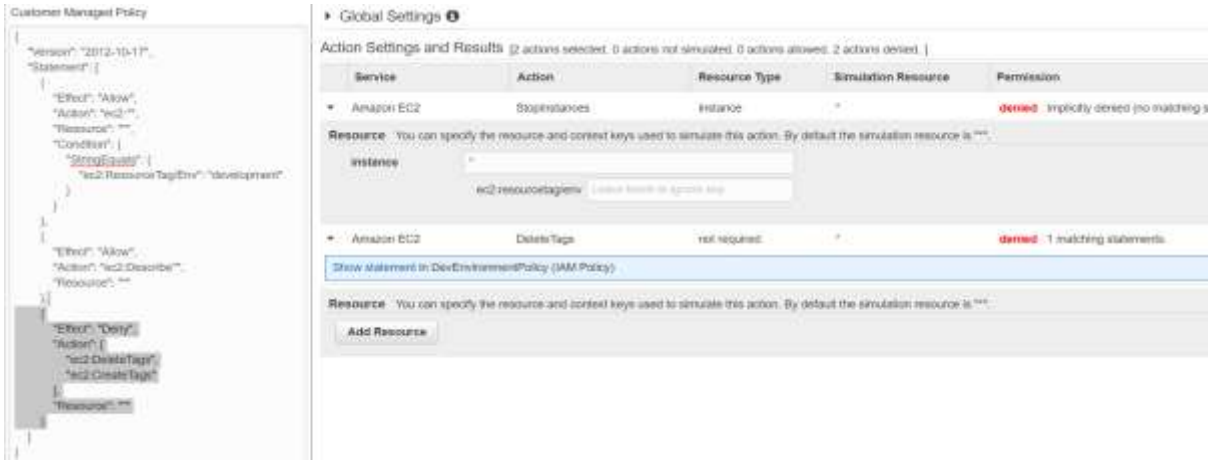


Policy Simulator: Safer Testing of IAM Permissions

After testing the policies directly by stopping instances, I reflected on how disruptive this could be in a real-world production environment. Stopping live instances can interrupt services or workflows.

To avoid this, I used the IAM Policy Simulator, a tool provided by AWS to test policies without executing real actions. It lets you simulate how a policy affects specific API calls and ensures access control logic is correct, without any impact on infrastructure.

This aligns with real industry practice, where policy simulation and audit are essential steps before applying changes to live systems.



Final Thoughts

This project gave me strong practical exposure to AWS IAM and foundational cloud security concepts, including:

- Implementing least-privilege access
- Separating environments using resource tags
- Writing and testing JSON-based IAM policies
- Managing users and permissions at scale
- Safely simulating access policies using AWS tools

By leveraging IAM and EC2 with security best practices, I now have hands-on experience with the kind of access control and policy management used by professionals in cloud operations and DevSecOps roles.