Cloud Security with AWS IAM

Melvin green

Introducing Today's Project!

Project overview

In this project, I will demonstrate how to use AWS IAM to control access and permission settings in our AWS account. I'm doing this project to learn about cloud security from the absolute foundations. - Every company thinks about access permissions, and there are even entire jobs called " IMA Engineers" focused on the skills I am about to learn today.

Tools and concepts

Services I used today were Amazon EC2 and AWS IAM. Key concepts I learnt include IAM users, policies, user groups, and account aliases. I also learned how to use the policy simulator and how JSON policy worked. How to launch an instance, how to tag an instance, and how to log in as another user.

Project reflection

This project took me approximately 1 hour today, including demo time. The most challenging part was understanding the IAM policy since it was written in JSON and it contained multiple statements. It was most rewarding to see permission denied when our intern tried to delete our production instance - The IAM access management system worked!

Tags

What I did in this step

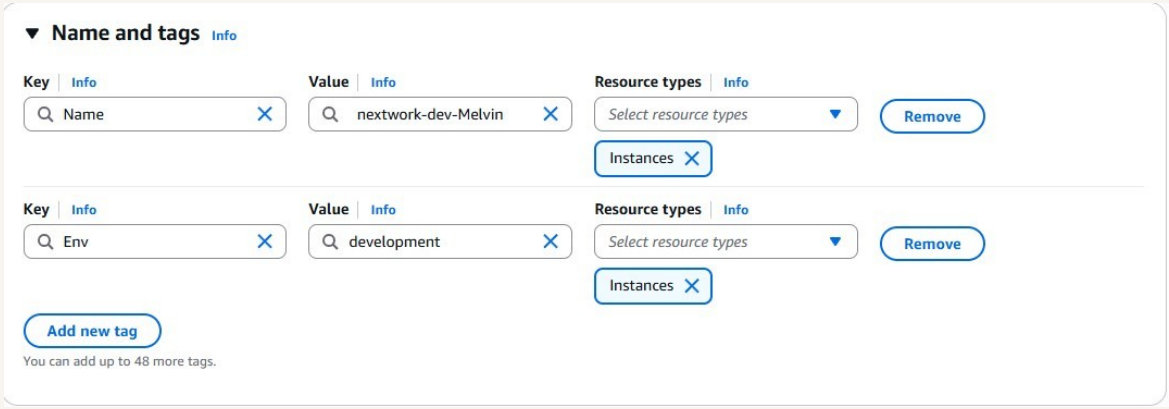
In this step, I will launch two EC2 instances because we need to boost
Nextwork's computing power, as we are expecting more users and traffic on our website.

Understanding tags

Tags are organisational tools that let us label our resources. They are helpful for grouping resources, cost allocation, and applying policies for all resources with the same tag.

My tag configuration

The tag I've used on my EC2 instances is called Env, which stands for environment. The values I've assigned for my instances are production and development



IAM Policies

What I did in this step

In this step, I will use IAM policies to control the access level of a new Intern because they should have access to the development environment (i.e the development instance) but NOT the production environment.

Understanding IAM policies

IAM Policies are rules that determine who can do what in the AWS account. I am using policies today to contorl who has access to our production/environment instance.

The policy I set up

For this project, I've set up a policy using JSON.

Policy effect

I've created a policy that allows the policyholder (the intern) permission to do anything they want in any instance tagged with "development". They can also see information for any instance, but they're denied access to deleting/creating tags for any instance.

Understanding Effect, Action, and Resource

The Effect, Action, and Resource attributes in a JSON policy define permissions in AWS. The Effect determines whether access is allowed or denied. The Action specifies what the policyholder can or cannot do. The Resource identifies the specific AWS resources that the policy applies to.

My JSON Policy

Account Alias

What I did in this step

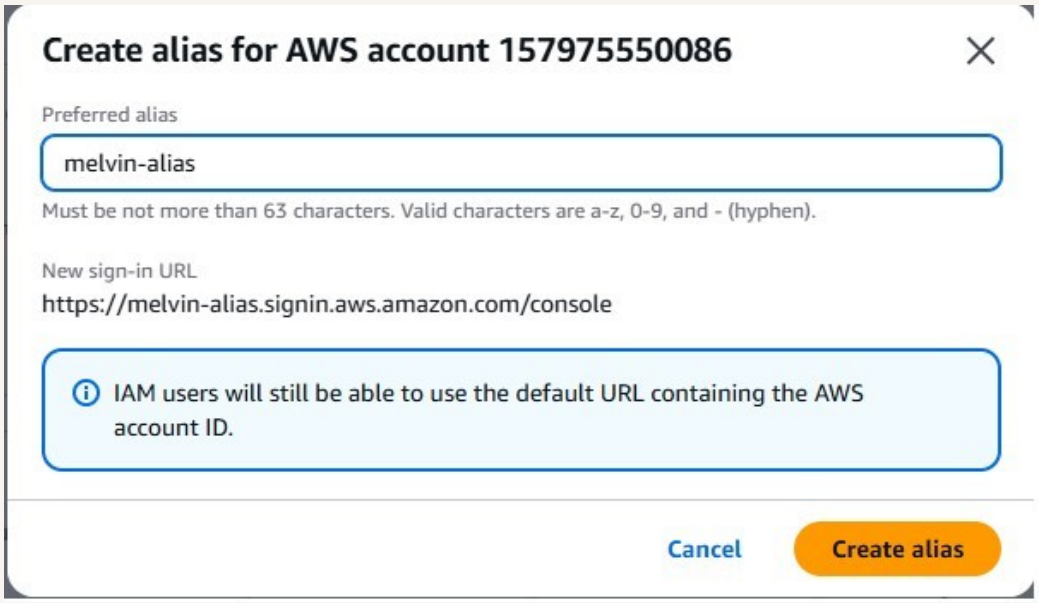
In this step, I will set up an account alias, which is considered a nickname for our AWS account's console login. This is because an account alias makes it simpler for our users to log in.

Understanding account aliases

An account alias is simply a nickname for the AWS account. Instead of a long account ID, we can now reference our account alias.

Setting up my account alias

Creating an account alias took me less than 1 min - it's a simple configuration in the IAM dashboard. Now, my new AWS console sign-in URL uses the alias instead of my account ID.



IAM Users and User Groups

What I did in this step

In this step, I will set up two IAM resources - IAM users and IAM user groups. This is because IAM users are like logins for people who want access to our AWS account, while user groups are like folders to manage users that have the same level of access

Understanding user groups

IAM user groups are like folders that collect IAM users so that you can apply permission settings at the group level.

Attaching policies to user groups

I attached the policy I created to this user group, which means any user created inside this group will automatically get the permissions attached to our NextWorkDevEnvironmentPolicy IA policy.

Understanding IAM users

IAM users are people or entities that have access/can login to your AWS account.

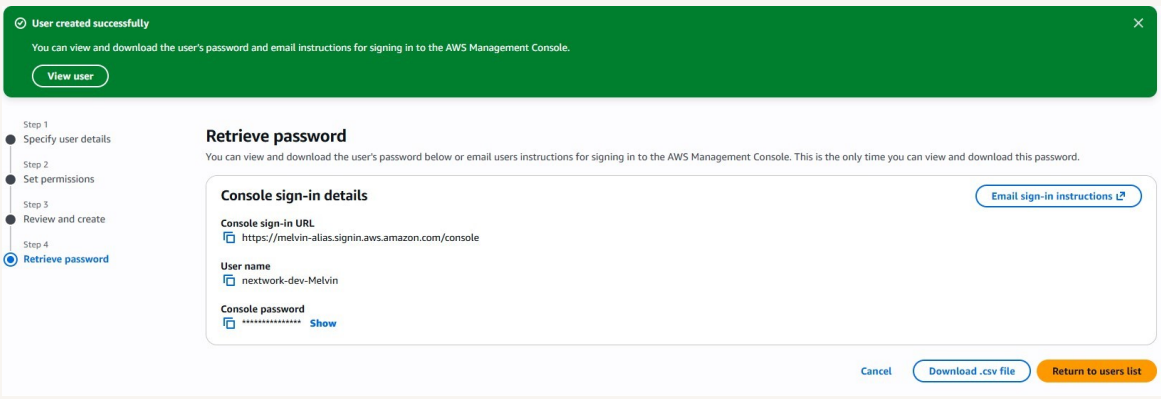
Logging in as an IAM User

Sharing sign-in details

The first way is to email sign-in instructions to the user, while the second way is to download a .csv file with the sign-in details inside.

Observations from the IAM user dashboard

Once I logged in as my IAM user, I noticed that our user already had denied access to panels in the main AWS dashboard. This was because we only set up permission to our development EC2 instances, so the intern wouldn't have access to even see anything else.



Testing IAM Policies

What I did in this step

In this step, I will log in to my own AWS account as the intern and test access to the production and development instances because I want to make sure the intern can't do anything that affects our production environment.

Testing policy actions

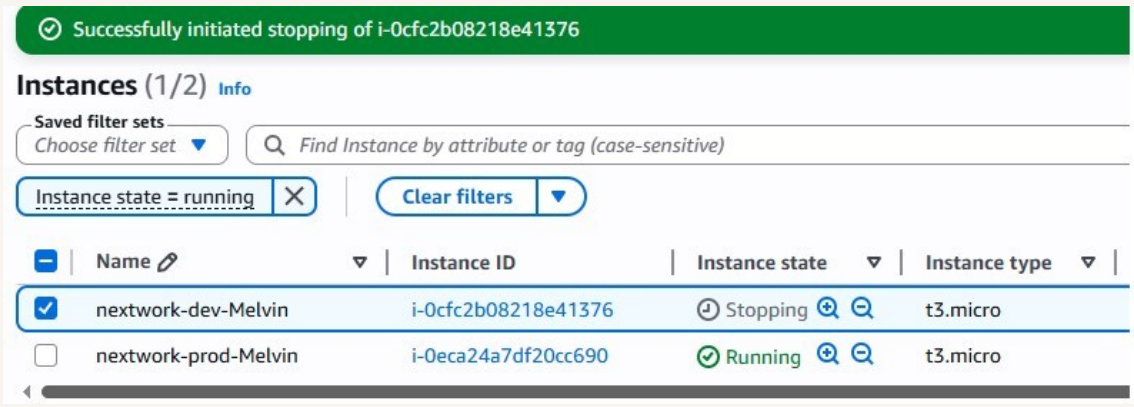
I tested my JSON IAM policy by attempting to stop both the development and production instances.

Stopping the production instance

When I tried to stop the production instance, I was met with an error. This was because the production instance was tagged with the "production" label, which is outside of the scope of the permission policy. Interns are only allowed to do things to the development instance.

Stopping the development instance

Next, when I tried to stop the development instance, I successfully saw the instance state change to stopping and then stopped. This was because the permission policy allowed the intern(i.e users in the nextwork-dev-group) to stop instances.



IAM Policy Simulator

To extend my project, I'm going to test the permission policy in a safer and more controlled way - a tool called the IAM policy simulator. I'm doing this because having to stop instances and log into AWS accounts as other users is a bit disruptive. Let's find a more efficient way!

Understanding the IAM Policy Simulator

The IAM Policy Simulator is a tool that lets me simulate actions and test permission settings by defining a specific user/group/role and the action I want to test for. It's useful for saving time when testing permission settings!

How I used the simulator

I set up a simulation for whether the dev user group has permission to stop instances or delete tags. The results were denied for both - we had to adjust the scope of the EC2 instances to ones that are tagged with "development". Once I applied the tag, permission was allowed.


Policy Simulator
Amazon EC2
2 Action(s) sele...
Select All
Deselect All
Reset Contexts
Clear Results
Run Simulation
Global Settings
Action Settings and Results [2 actions selected. 0 actions not simulated. 1 actions allowed. 1 actions denied.]
Service
Action
Resource Type
Simulation Resource
Permission
Amazon EC2
StopInstances
instance
*
allowed 1 matching statements.
Amazon EC2
DeleteTags
not required
*
denied 1 matching statements.