NextWork.org

Cloud Security with AWS IAM

DI dineshrajdhanapathy@gmail.com

Introducing today's project!

What is AWS IAM?

AWS IAM (Identity and Access Management) is a service for securely managing user access and permissions in AWS. It ensures resources are protected by allowing fine-grained control over who can access them.

How I'm using AWS IAM in this project

In today's project, I used AWS IAM to create custom policies and assign two EC2 instances, Production and Development. This ensured secure user access control for development and did not access to the production environment.

One thing I didn't expect...

One thing I didn’t expect in this project was a misconfigured IAM policy that unintentionally gave access to a critical production and development stop. Debugging and refining the policy resolved the issue efficiently to not access the production.

This project took me...

This project took approximately 2 hours to complete, including setting up EC2 instances, configuring AWS IAM roles and policies, troubleshooting access issues, and testing the overall functionality.

Tags

Tags are labels assigned to resources for better organization. In production, they track costs and monitor resources. In development, they separate environments, aid testing, and automate cleanup.

The tag I've used on my EC2 instances is called "Environment." The values I've assigned for my instances are "Production" for the live server and "Development" for the testing environment.

IAM Policies

IAM Policies are documents that define permissions for AWS users, groups, or roles. They control access to AWS resources, specifying allowed or denied actions, ensuring secure and managed resource access.

The policy I set up

For this project, I've set up a policy using the visual editor. It provided an intuitive interface to configure permissions, ensuring accuracy, but JSON remains available for advanced customizations.

I've created a policy that allows starting, stopping, and describing EC2 instances tagged with "Env=development" while denying the ability to create or delete tags for all instances, ensuring control.

When creating a JSON policy, you have to define its Effect, Action and Resource.

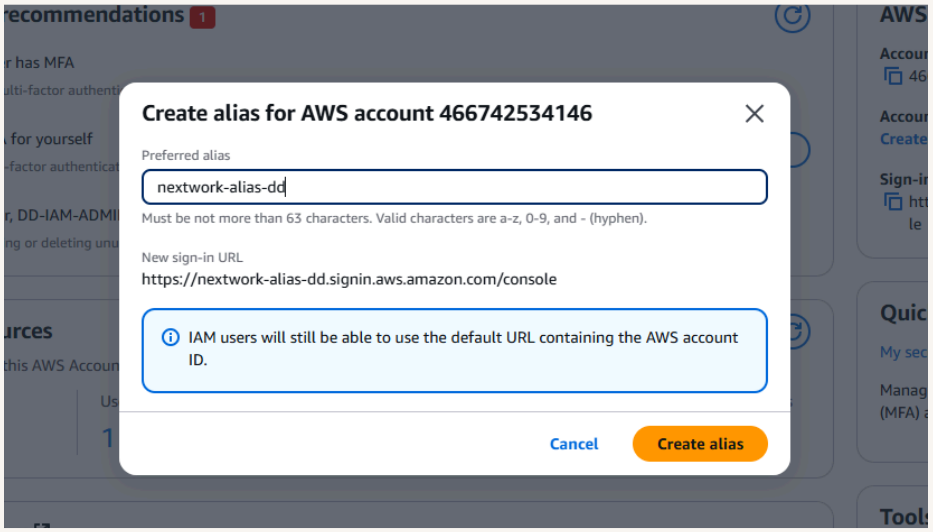
The Effect, Action, and Resource attributes of a JSON policy mean the following: Effect determines Allow/Deny actions, Action specifies permitted/denied operations, and Resource targets impacted entities.

My JSON Policy

Account Alias

An account alias is a user-friendly name for your AWS account, replacing the default numerical account ID. It simplifies account identification and is used in the AWS Management Console sign-in URL.

Creating an account alias took me just a few minutes. Now, my new AWS console sign-in URL is easier to use and looks like https://<alias>.signin.aws.amazon.com/console.



IAM Users and User Groups

Users

IAM users are entities within AWS that represent individual people or applications. They have specific permissions assigned to them, allowing them to interact with AWS resources based on their defined roles.

User Groups

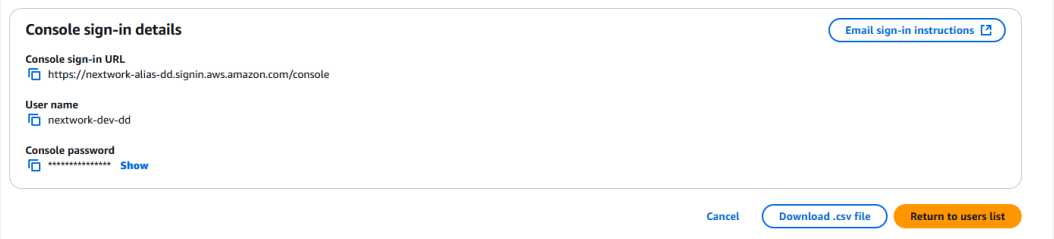
IAM user groups are collections of IAM users that allow you to manage permissions collectively. Instead of assigning permissions to individual users, you can assign them to a group, simplifying management.

I attached the policy I created to this user group, which means all members of the group inherit the defined permissions, allowing them to perform the specified actions on the designated resources.

Logging in as an IAM User

The first way is to email the new user's sign-in details, including their username and password. The second way is to provide a secure URL link for them to set up their password via the AWS console.

Once I logged in as my IAM user, I noticed an "Access Denied" message on the AWS dashboard. This was because the user didn't have the necessary permissions to view or manage any resources initially.

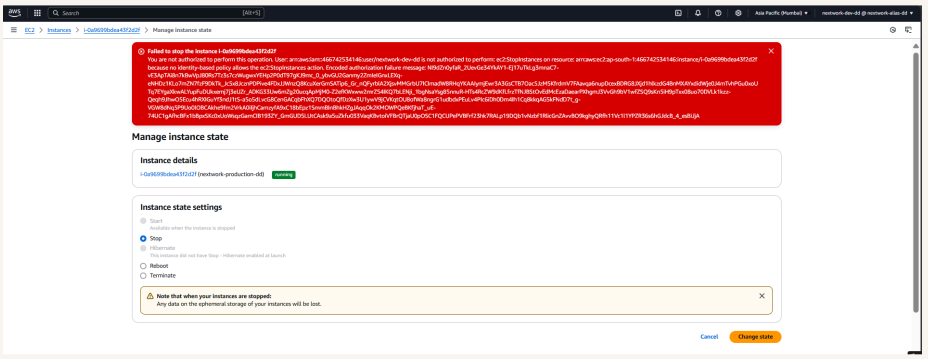


Testing IAM Policies

I tested my JSON IAM policy by applying it to two EC2 instances and verifying permissions. One instance allowed the intended actions, while the other blocked unauthorized access as per the policy setup.

Stopping the production instance

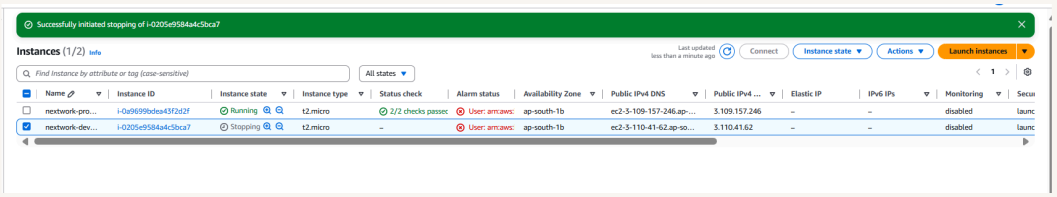
When I tried to stop the production instance, it failed due to an active dependency preventing shutdown. This was because a critical service relied on the instance, blocking the termination process.



Testing IAM Policies

Stopping the development instance

Next, when I tried to stop the development instance, it stopped successfully without issues. This was because no dependencies or active services were linked to the instance at the time.



NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for more projects