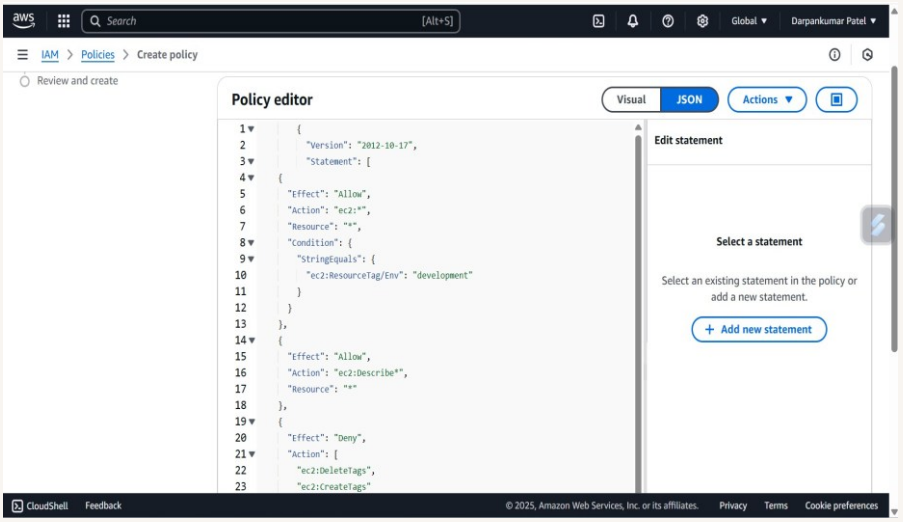
Darpan Patel

Cloud Security with AWS IAM



 Darpan Patel

Introducing Today's Project!

In this project, I will demonstrate how to use AWS Identity and Access Management (IAM) to control access to cloud resources like a pro. I'll launch an EC2 instance (a virtual server), create IAM policies to set permissions, organize access with IAM users and groups, and add an AWS account alias for a smoother sign-in experience. I'm doing this project to learn the ins and outs of AWS security, understand how IAM works in real-world scenarios, and build hands-on skills for managing cloud resources.

Tools and concepts

Services I used were Amazon EC2, AWS Identity and Access Management (IAM), and Account Alias for simplified login. Key concepts I learnt include: Creating and managing IAM policies using JSON to define precise access control. The principle of least privilege, by giving users access only to what they need (e.g., access to development EC2 instance only). User and group management in IAM, which simplifies permissions across teams. Testing permissions using IAM users to verify access controls before deploying to real users. Creating account aliases to improve user login experience and professionalize the AWS environment.

Project reflection

This project took me approximately 1 hour. The most challenging part was designing and testing the IAM policy correctly to ensure the intern had access only to the development EC2 instance and was denied access to production. It required attention to detail with instance IDs and understanding how AWS permissions work. It was most rewarding to see the access controls function exactly as intended during testing— allowing the intern user to interact with the development environment while being blocked from production.

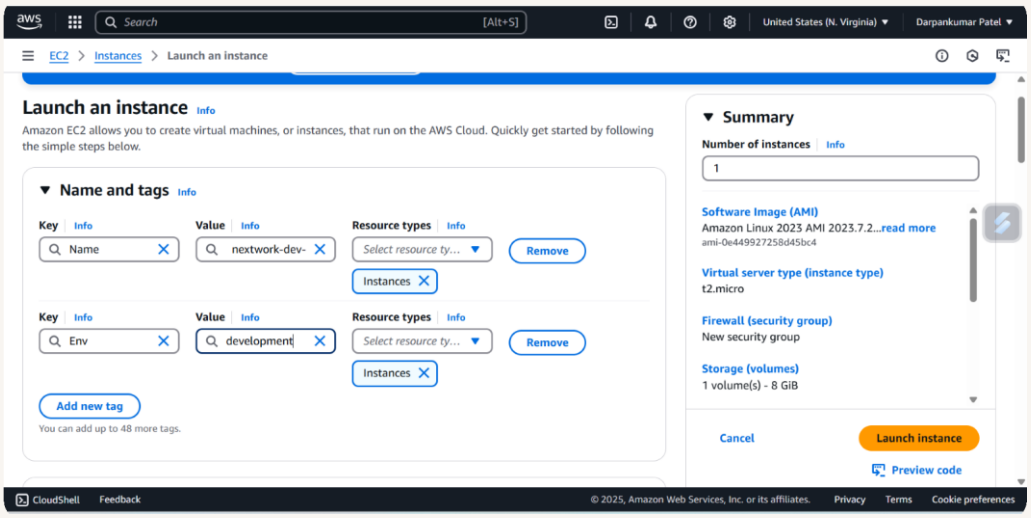
This reinforced the importance and power of well-crafted IAM policies in securing cloud resources.

 Darpan Patel

Tags

In this project, I used AWS tags—think of them as sticky notes I put on my cloud resources (like EC2 instances) to keep everything organized. Tags are key-value pairs, like Env: development that make managing my AWS setup a breeze. Why I love tags: Easy organization: I can filter my resources, like finding all Env: development instances for NextWork's testing environment in a snap. Cost tracking: By tagging with Project: NextWork, I can see exactly what my project's costing. Smarter security: I used tags in my IAM policy to let our intern manage only Env: development instances, keeping the production ones safe. Automation vibes: Tags help me automate tasks, like shutting down dev instances to save money. In this project, I tagged my EC2 instances (Env: production and Env: development) to separate environments and control access with IAM. Tags made my cloud setup clean, secure, and ready for the holiday traffic surge

The tag I've used on my EC2 instances is called Env. The values I've assigned for my instances are production for the nextwork-prod-darpankumar-patel instance and development for the nextwork-dev-darpankumar-pateel instance.



 Darpan Patel

IAM Policies

IAM Policies are JSON documents in AWS that define permissions. They specify what actions are allowed or denied on which AWS resources, and under what conditions. These policies help manage access control for users, groups, and roles by enforcing security boundaries. For example, an IAM policy can allow a user to start and stop EC2 instances in a specific environment but prevent access to production resources.

The policy I set up

For this project, I've set up a policy using the JSON editor. I wrote the policy manually to define the specific actions, resources, and conditions for the intern's access to the development EC2 instance. This approach gave me more control and flexibility over the permissions.

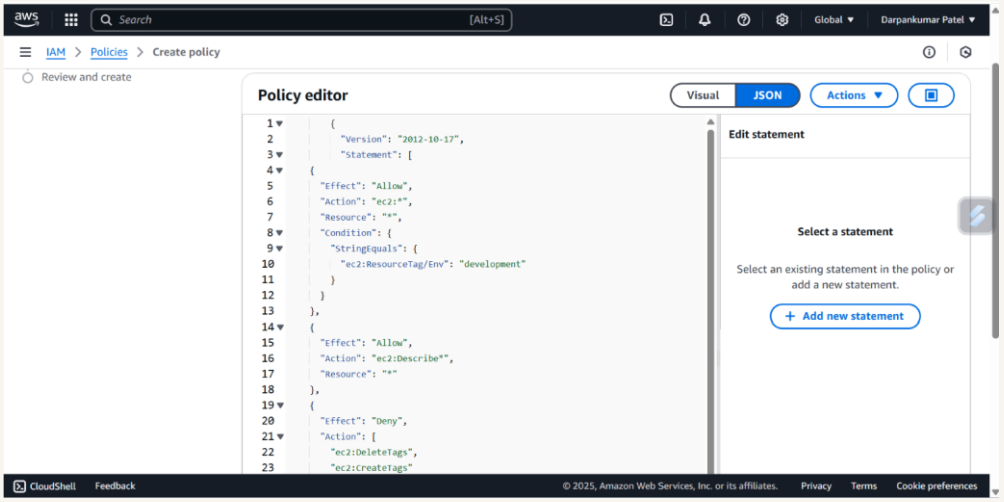
I've created a policy that allows the intern to perform specific actions—such as starting, stopping, and describing—on the development EC2 instance only. This ensures they can interact with the dev environment without having access to the production instance, protecting critical infrastructure from accidental changes.

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: Effect defines whether the policy allows or denies access. It can be "Allow" or "Deny". Action specifies which AWS service actions are permitted or denied (e.g., ec2:StartInstances, s3:PutObject). Resource identifies the specific AWS resource(s) the policy applies to, usually given in the form of an ARN (Amazon Resource Name).

 Darpan Patel

My JSON Policy

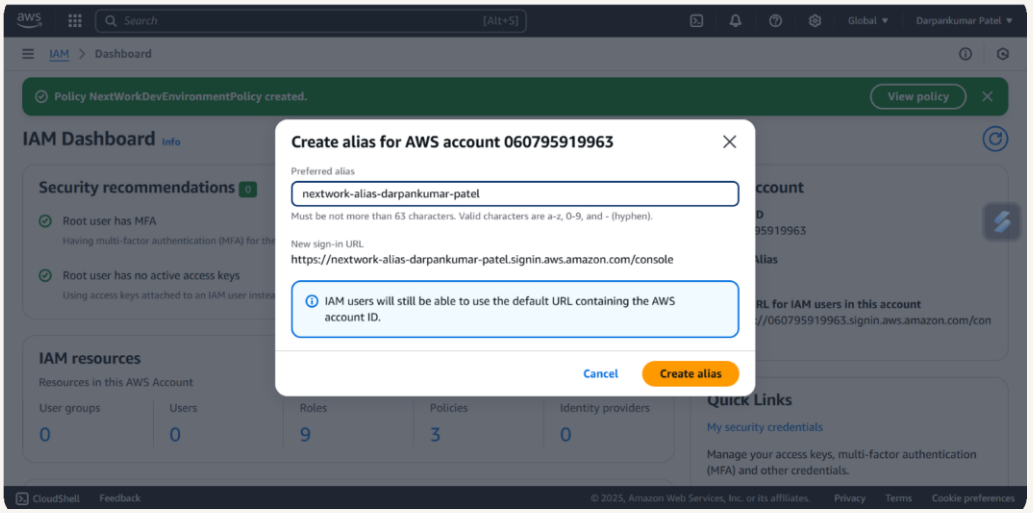


Account Alias

An account alias is a custom, human-readable name that you assign to your AWS account, which replaces the AWS account ID in the URL for the AWS Management Console. This alias helps simplify and personalize the login process, allowing users to log in to your AWS account using a URL that is easier to remember, instead of the long, numeric AWS account ID. For example, instead of logging in via: https://<aws_account_id>.signin.aws.amazon.com/console You can use an alias like: https://myteamname.signin.aws.amazon.com/console This makes the login process more intuitive and can help teams stay organized by using a consistent and identifiable alias.

 Darpan Patel

Creating an account alias took me just a few minutes. Now, my new AWS console sign-in URL is:https://nextwork-alias-darpankumarpatel.signin.aws.amazon.com/console



IAM Users and User Groups

Users

IAM users are individual identities created within your AWS account that represent real people or applications needing access to AWS resources. Each IAM user has unique credentials (such as a username and password, or access keys) and can be assigned specific permissions through policies. This allows you to control and monitor what each user can do within your AWS environment securely.

 Darpan Patel

User Groups

An IAM user group is a collection of IAM users that makes it easier to manage permissions for multiple users. Instead of assigning policies to each individual user, you attach policies to the group — and all users in that group inherit those permissions.

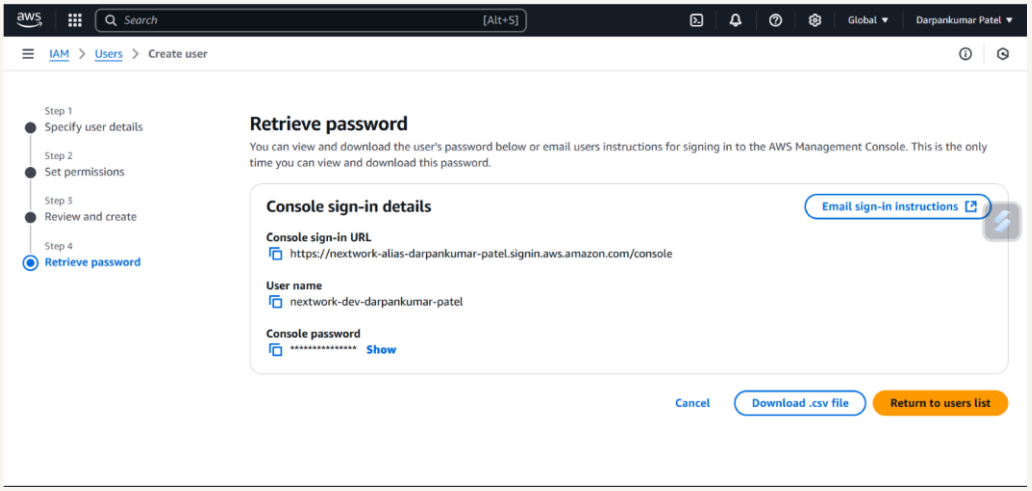
I attached the policy I created to this user group, which means every user added to the group automatically gets the permissions defined in that policy. In this case, all members of the nextwork-dev-group now have controlled access to the development EC2 instance, without needing individual policy assignments. This simplifies permission management and ensures consistency across all interns.

Logging in as an IAM User

The first way is to download the .csv file containing the user's login credentials (username, password, and sign-in URL) immediately after creating the IAM user. The second way is to copy and manually send the sign-in link, username, and temporary password to the user via a secure communication method such as email or a password manager.

Once I logged in as my IAM user, I noticed that access to certain services and resources was restricted, with some dashboard panels showing “Access Denied.” This was because the IAM user only had permissions granted through the attached policy, which was specifically designed to allow access only to the development EC2 instance — not the production instance or any other services. This confirms that the policy is correctly limiting access according to the principle of least privilege.

 Darpan Patel



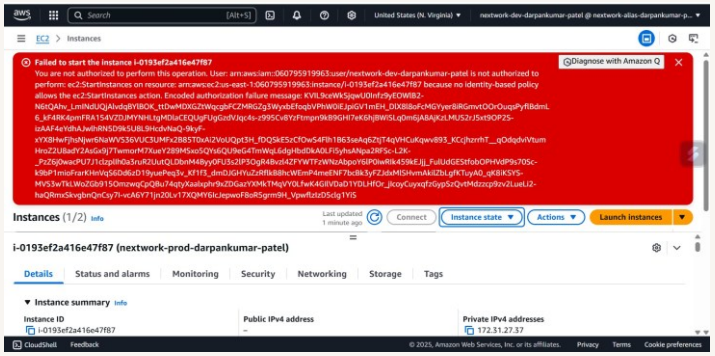
Testing IAM Policies

I tested my JSON IAM policy by attempting to stop both the development and production EC2 instances. The action was successful for the development instance, confirming that the IAM user had the correct permissions. However, access was denied for the production instance, which verified that the policy correctly restricts access to only the development environment.

Stopping the production instance

When I tried to stop the production instance, the action was denied, and I received an “UnauthorizedOperation” or “Access Denied” message. This was because the IAM policy attached to the user only allows actions on the development instance, and explicitly or implicitly denies permissions for the production EC2 instance. This behavior confirms that our access control setup is working correctly and prevents interns from accidentally interfering with production resources.

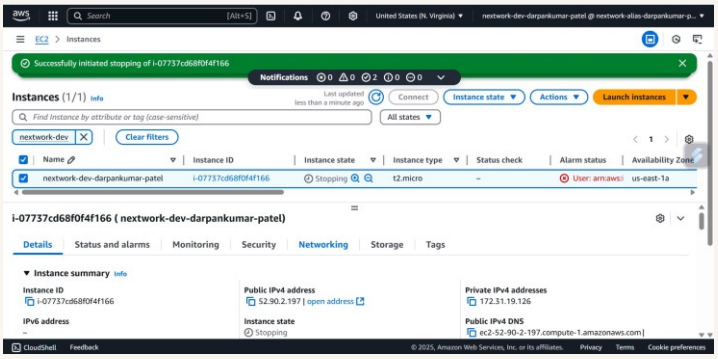
 Darpan Patel



Testing IAM Policies

Stopping the development instance

Next, when I tried to stop the development instance, the action was successful. This was because the IAM policy attached to the user or user group explicitly grants permission to perform actions like StopInstances on the specific development EC2 instance, identified by its resource ARN.



 Darpan Patel

The IAM Policy Simulator

The IAM Policy Simulator is a tool provided by AWS that allows you to test and troubleshoot IAM policies without making real changes to your resources. It's useful for validating whether a user, group, or role has the permissions to perform specific actions on specific AWS resources, all without having to actually execute those actions. This is especially helpful for identifying misconfigurations and ensuring that your security policies are working as intended—without the risk of affecting production environments.

How I used the simulator

I set up a simulation for the "StopInstances" action on the development EC2 instance using the IAM Policy Simulator for my intern IAM user. The results were "Allowed", which confirmed that the user has the necessary permissions to interact with the development instance. I did not need to adjust the policy because it was correctly scoped to only allow actions on the development instance and not the production instance.