Securing Cloud Infrastructure: AWS IAM-Based Access Control for Scalable and Safe Intern Onboarding

AWS Services Utilized:

- Amazon EC2 – For launching isolated production and development environments.
- AWS Identity and Access Management (IAM) – For creating users, groups, and permissions.
- IAM Policy Management – To define fine-grained access controls.
- AWS Account Alias – For customized identity branding.
- IAM Roles and Groups – To manage permissions across user categories.

By
Nishchal Sreevathsa

Abstract:

This project focuses on implementing cloud security best practices using AWS Identity and Access Management (IAM) to safeguard critical infrastructure while supporting seasonal business expansion. Two Amazon EC2 instances were deployed to represent distinct production and development environments. A scalable access control policy was established whereby interns were granted limited access exclusively to the development environment—ensuring protection of production workloads. IAM users, groups, and roles were configured to enforce least privilege principles, and an AWS Account Alias was assigned to improve login clarity and user management.

With the onset of the holiday season, the project also prepared the company's infrastructure to handle increased traffic from new users, while efficiently onboarding an intern with secure, role-appropriate permissions—balancing accessibility with operational resilience.

Description

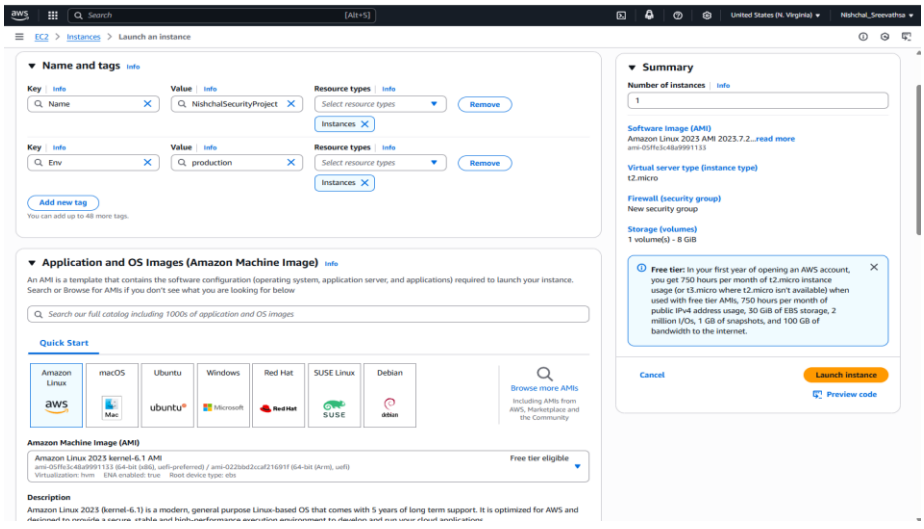
Step1: Launch EC2 Instance

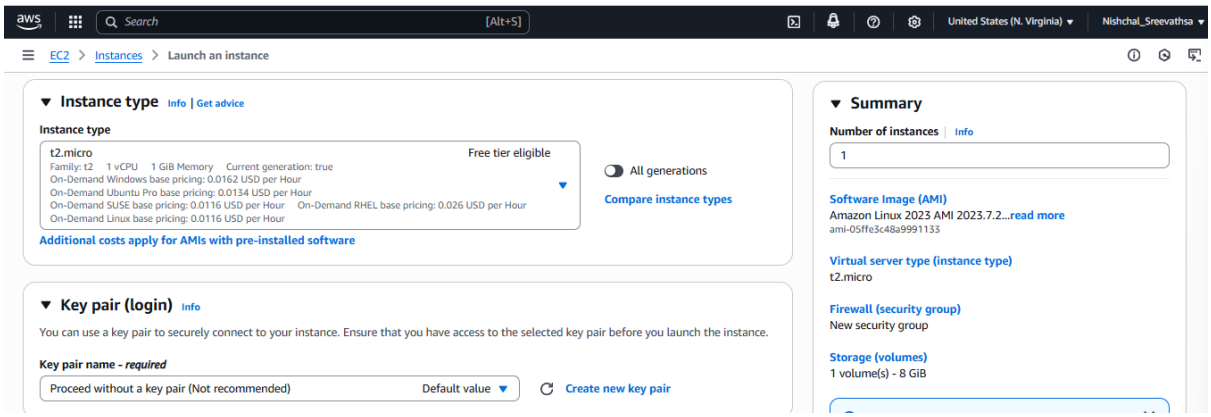
I have set up two EC2 instances to test the effectiveness of the permission settings I'll set up in AWS IAM. I have used tags to label them.

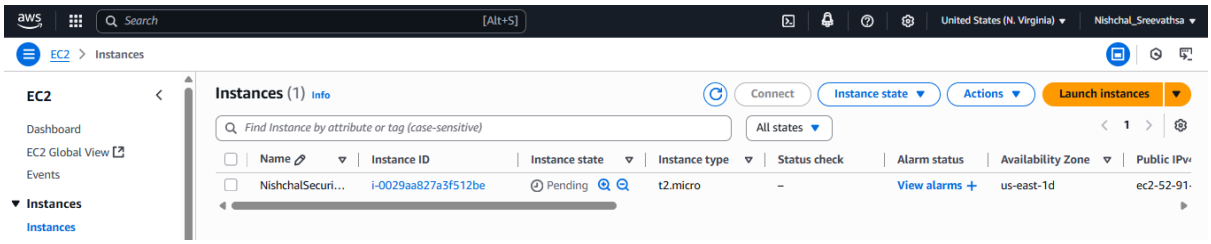
Tags are labels to help AWS account users identify and manage their resources. Tags are useful for grouping mass management and applying security policies.

The Tags I've use on my EC2 instances is called Env. The value I've assigned for my instance are production and development. This represents the two different environments that we are using to build and release my project app.

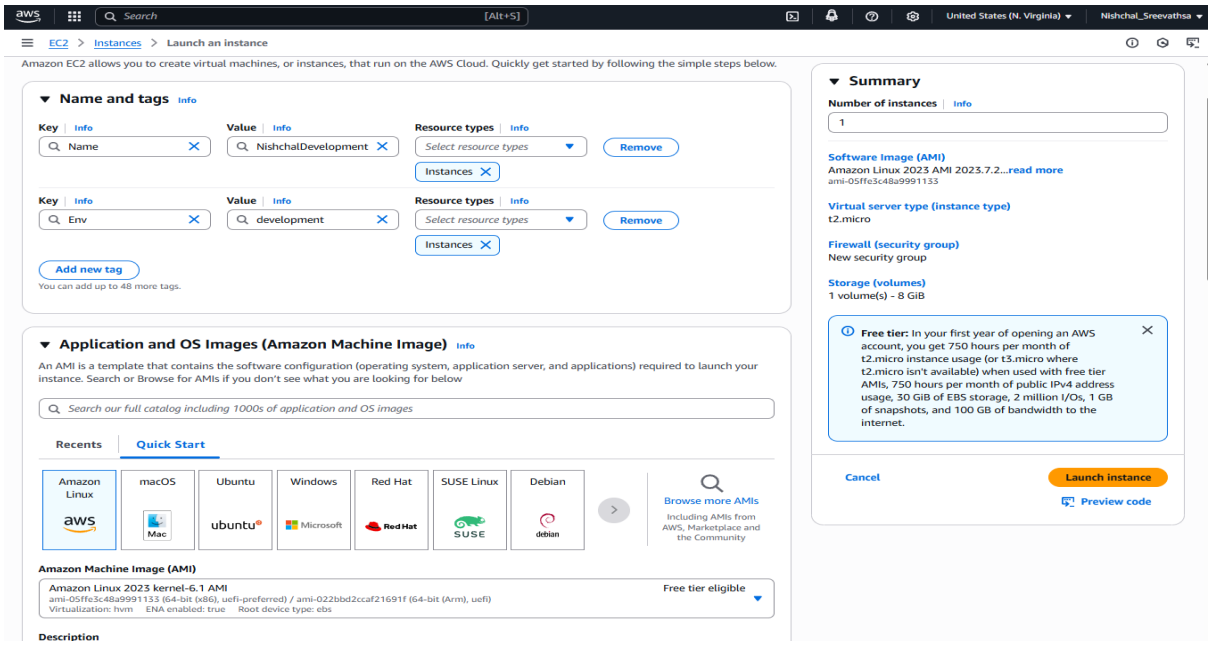
For instance1: I have named the instance name as Nishchal Security Project. The key is 'env' and value is 'production'. Selected Amazon Linux OS image which has free tier eligible and instance type is t2.micro. For key pair, I have selected proceeded without key pair since I am not using any command line interface to login.

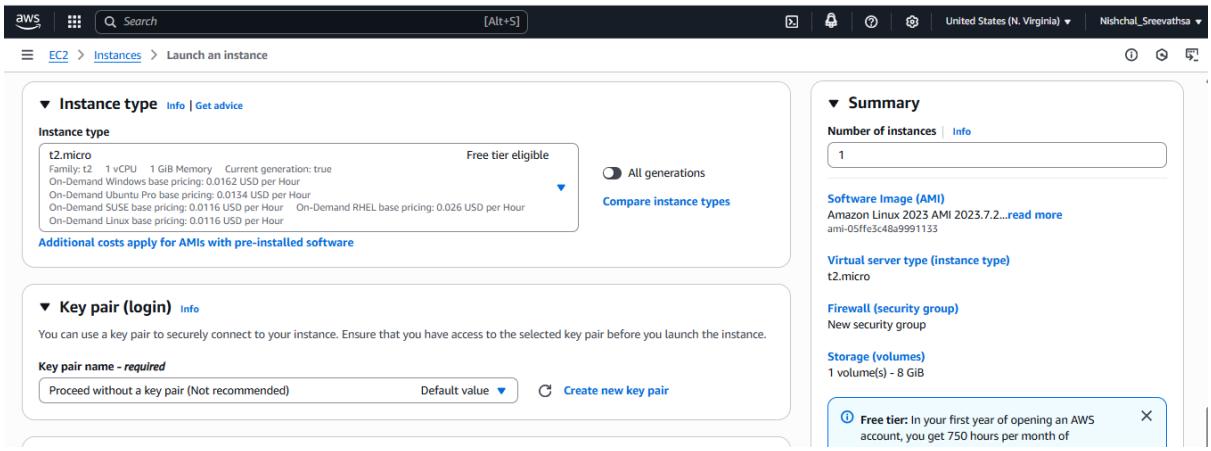


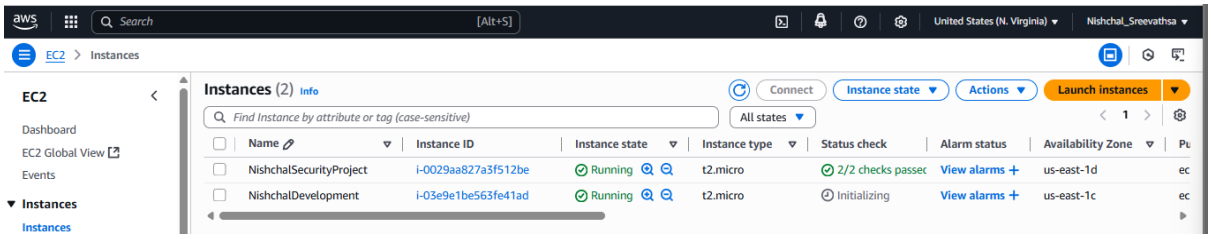




For instance2: I have named the instance name as Nishchal Development. The key is 'env' and value is 'development'. Selected Amazon Linux OS image which has free tier eligible and instance type is t2.micro. For key pair, I have selected proceeded without key pair since I am not using any command line interface to login.





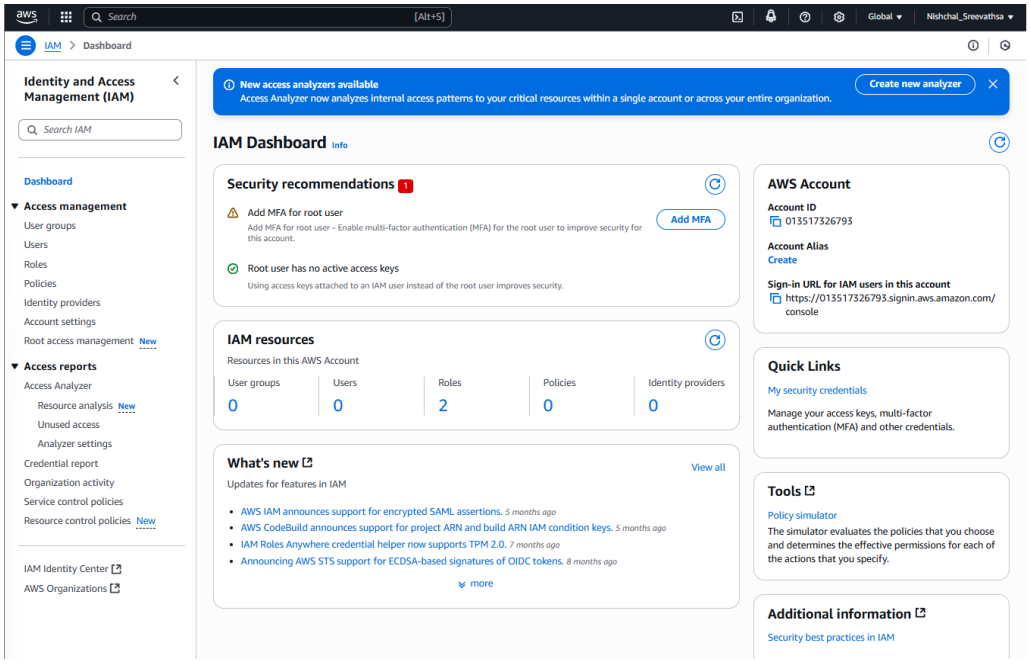


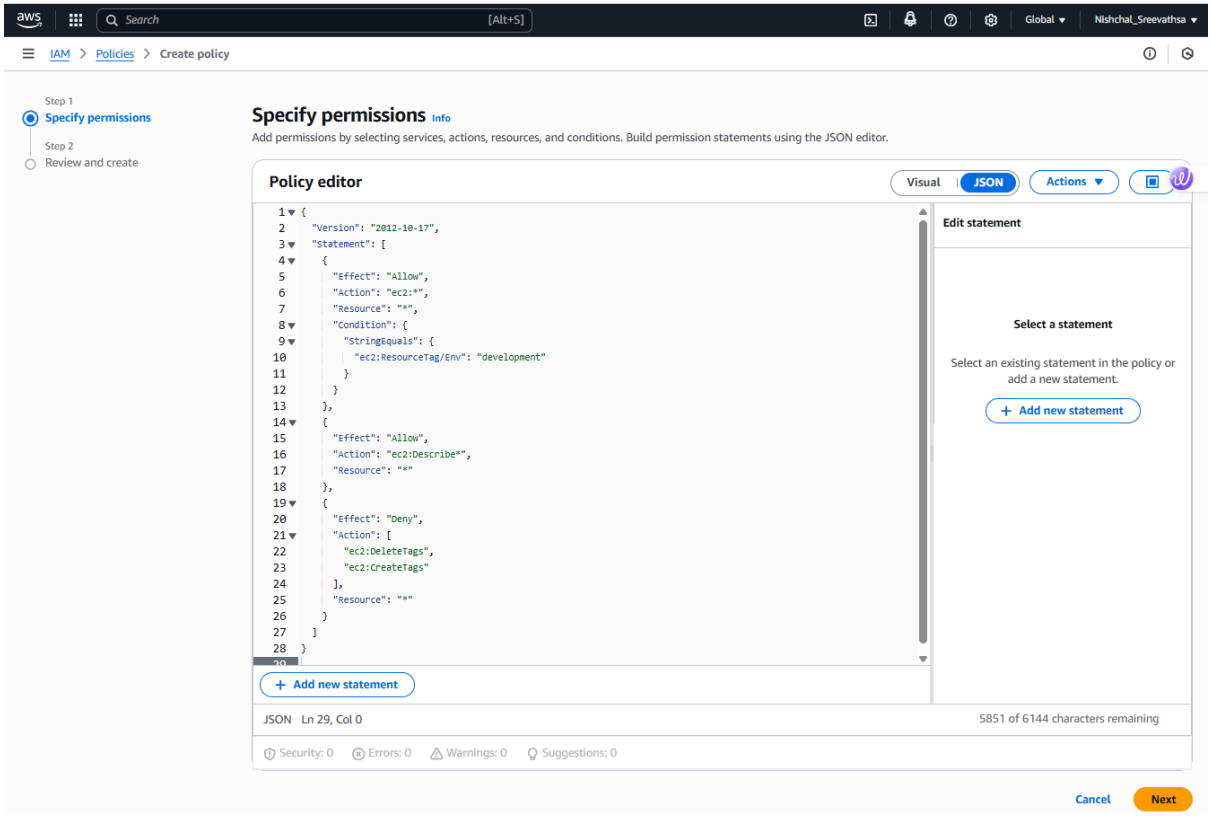
Step 2: Creating an IAM Policy - it's time to onboard the team's new intern and set up permission policies.

Our intern should have permission to the development EC2 instance but **not** the production instance. We don't want them to accidentally shut down the platform or push their changes to the production environment while they're just testing things!

IAM Policies are rules that help to allow/deny users/resources permissions to perform certain actions to my AWS Account's resources.
In this project, I've set up a policy using the JSON editor.
I've created a policy that allows all EC2- related actions to all the EC2 instances that have the Environment ("ENV") tag "Development". But it also denies creating and deleting tags for all EC2 instances.
When writing JSON Policy statements I have specifies the following:

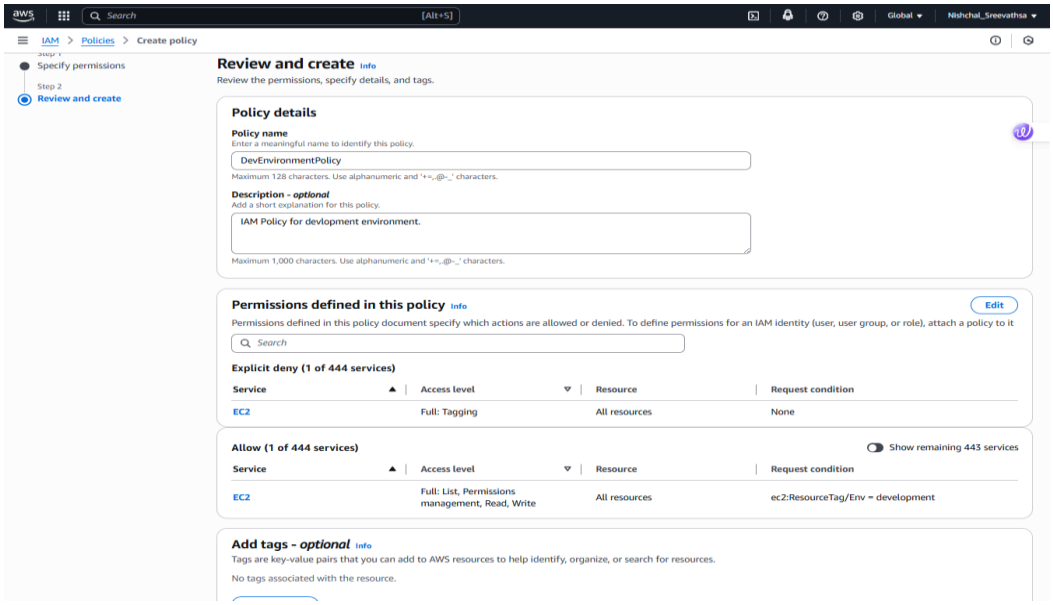
- Effect: i.e. Allow or Deny.
- Action: i.e the specific action that we are wanting to allow or deny.
- Resource: the specific resource/group of resources in my AWS Account that this policy will effect on.





I have entered policy's details as:

- Name: DevEnvironmentPolicy
- Description: IAM Policy for development environment.



Step3: Creating AWS Account Alias

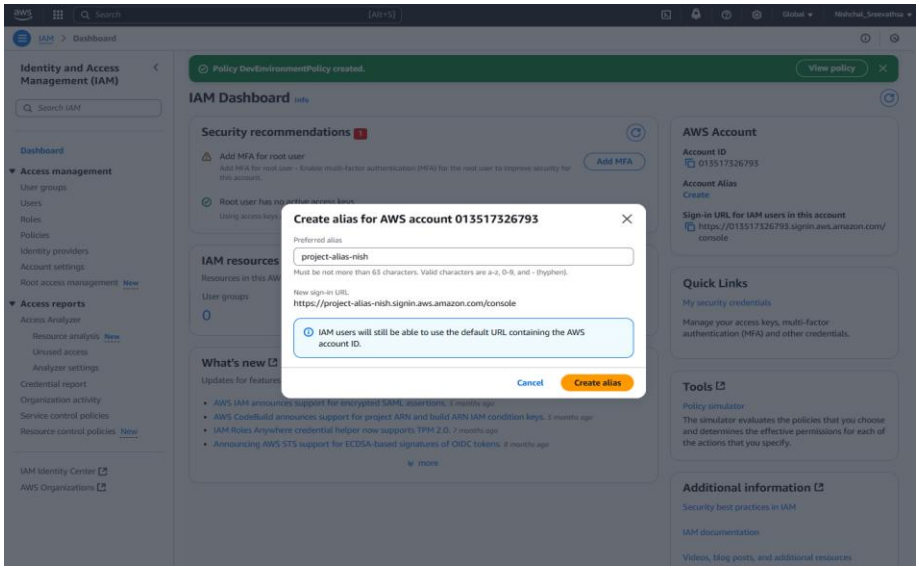
Now that we can give our intern access to the development instance, the intern can't wait to start. They'd love to jump into the team's AWS account right away!

When new users get onboarded onto my AWS account, they get access by signing into a unique URL created for my account's Account ID.

An account alias is a custom name that I can assign to me AWS Account. This custom name would replace my AccountID in my Account's log-in URL.
Creating an account alias took me less than a minute-super fast!

Now, my new AWS console sign-in URL is https://project-alias-nish.signin.aws.amazon.com/console

I have named my account alias name as **project-alias-nish**.



Step 4: Create IAM Users and User Groups

Your account alias is looking slick! It should be a lot faster for users to find your account and log in now.

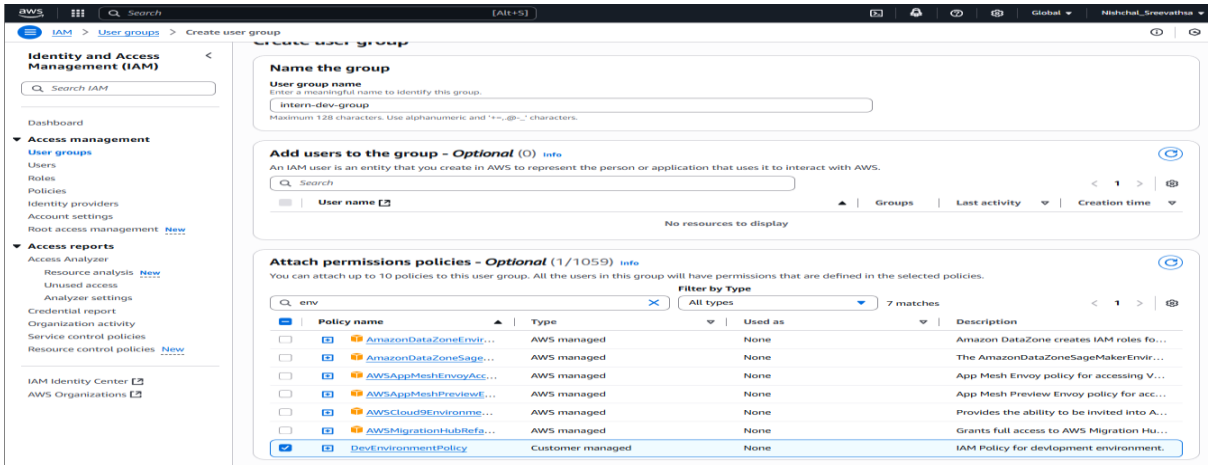
Our new intern doesn't actually have a way to log in to the team's AWS account yet... what should their username and password be?

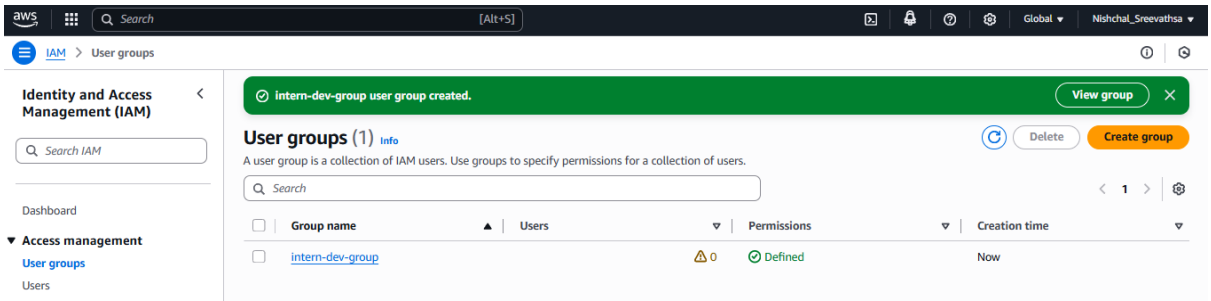
I wouldn't want to just share your account details with them. After all, you have access to the production instance, which they shouldn't get access to.

IAM Users are other log-ins/people who have access to my AWS account. These were users are created by myself using AWS IAM service! I can designate my IAM user's access to my AWS account's resources/services.

I also created a User Group. User groups are useful for grouping and managing user's permissions at a group level. They act similarly to folders when it comes to mass assigning permissions/policies.

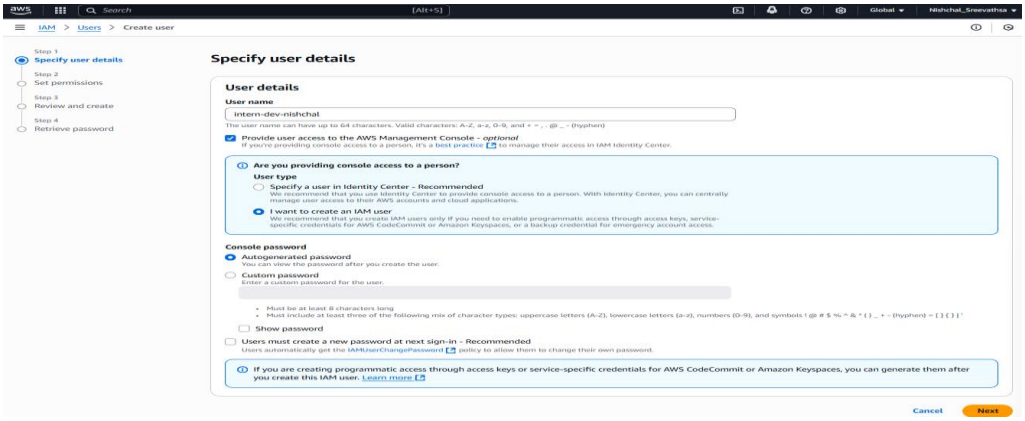
My users Group is called intern-dev-group. I attached the policy I created to this User group which means all the users that are added to that user's group will automatically inherit the user groups access permissions.

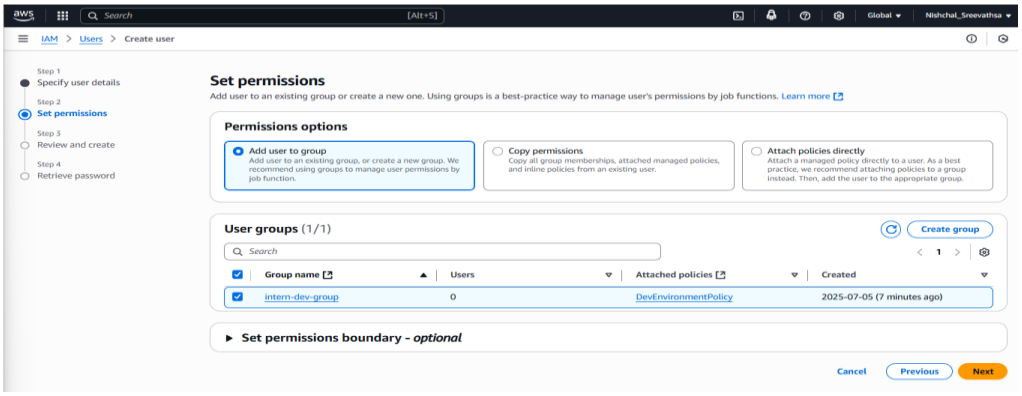


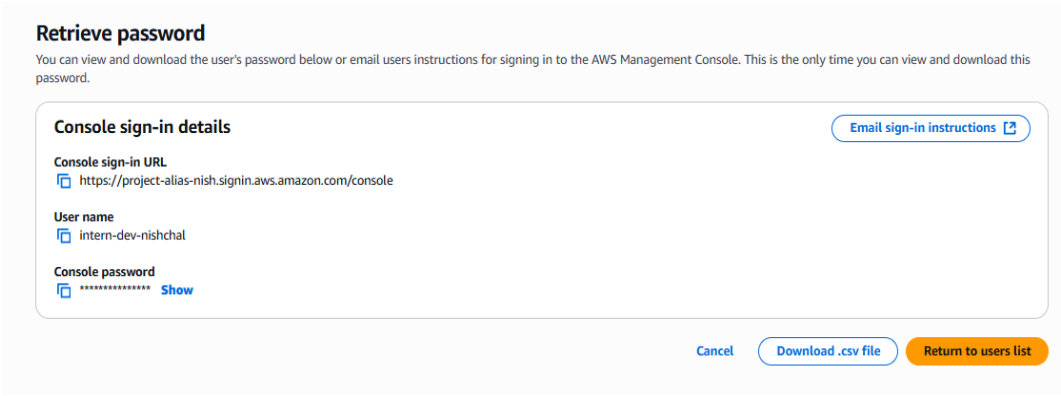


When I created a new User. I had to tick a checkbox that allowed the user access to the AWS Management Console.
Once my new user was set up, there were two ways I could share its sign-in details: firstly: emailing sign-in instructions; secondly: do downloading .csv file.
My new user had unique sign-in URL- this is my Account work!

https://project-alias-nish.signin.aws.amazon.com/console





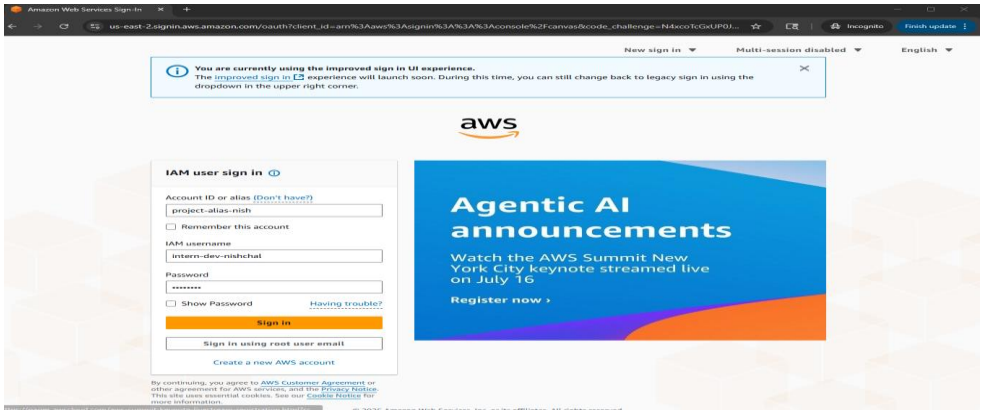


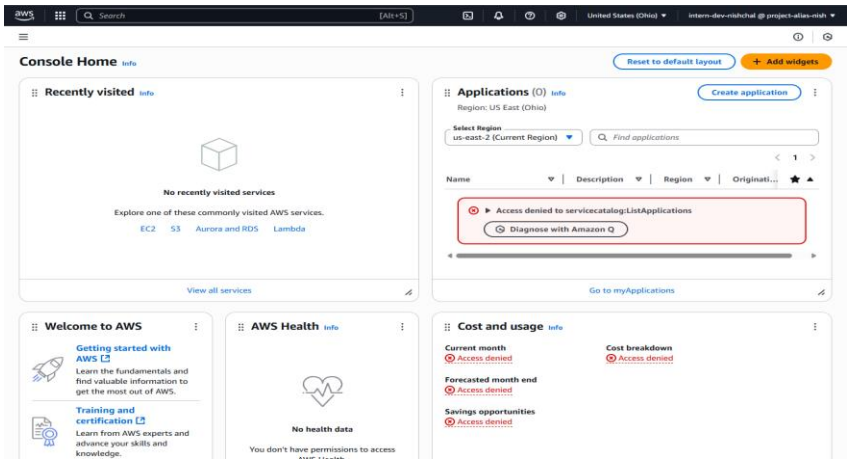
Step5: **Test your intern's access**

The new intern is going to be stoked to receive their keys to the AWS account - well done

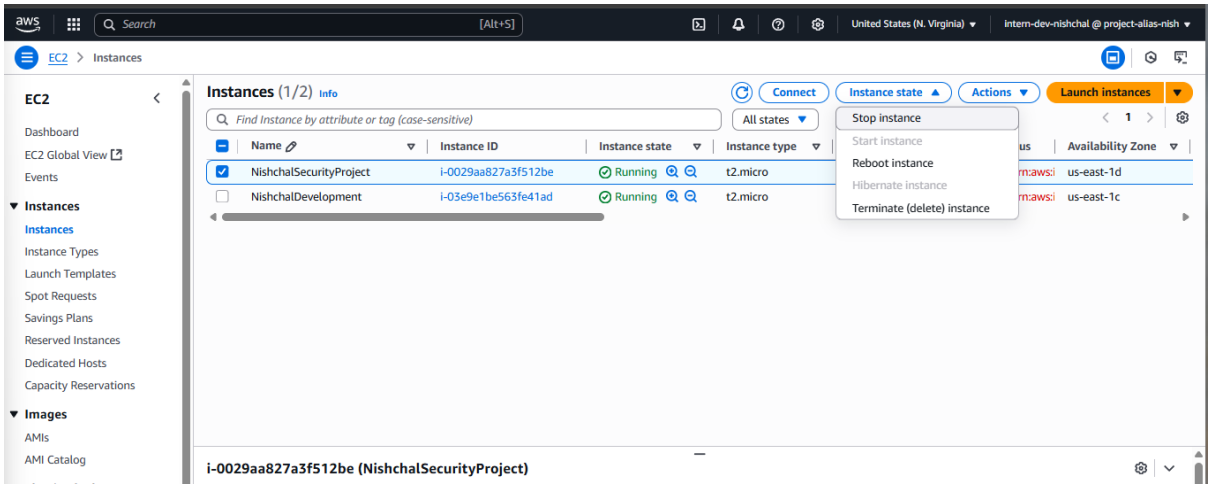
Before we pass them their login details, let's test the interns' IAM User's access first. That way we can make sure that they have the right access to our development instance (and not the production instance).

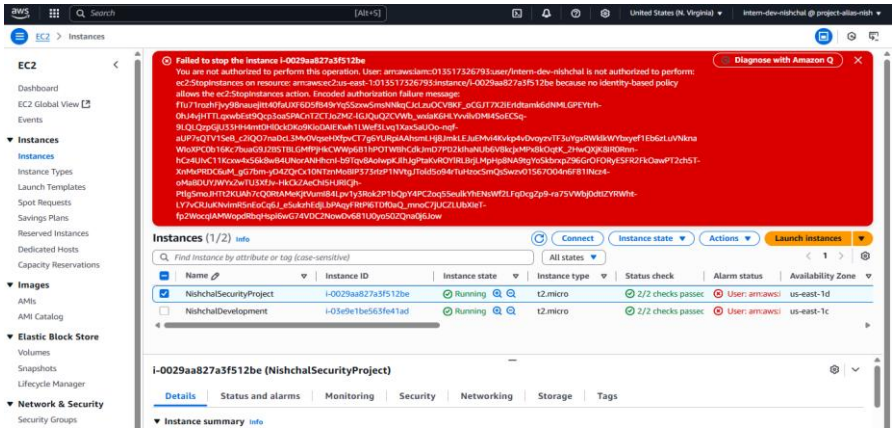
Now with my IAM Policy, IAM User Group and IAM User all set up, let’s put it all together! To do this, I logged into my AWS account as the new user.
To log in as my IAM User, I had to access the console log-in URL that was given to me as I set up the new user.
Once I logged in, I noticed that a lot of panels displayed “Access Denied”. This was clear difference to the dashboard I usually see in my AWS account (where I had unrestricted access to resources and wasn’t denied access to anything.)



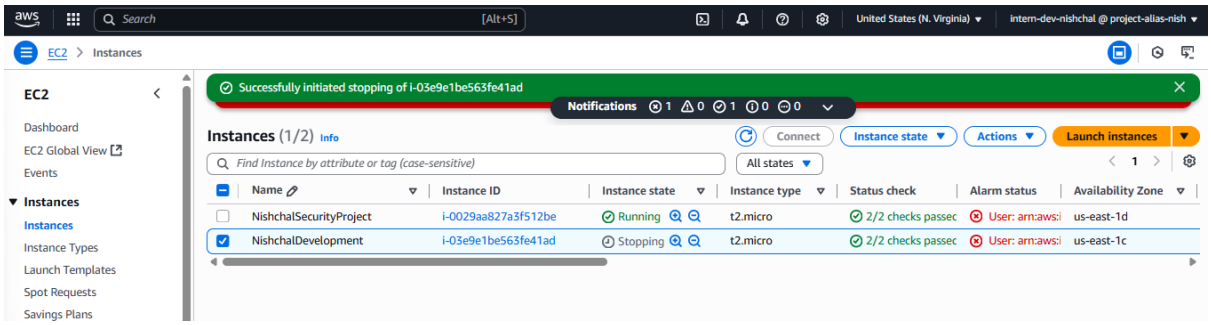


I tested the JSON IAM policy I set up by trying to stop the development and production instances i.e triggering the Stop Instance action.
When I tried to stop the production instance, an error message stopped me and explained that I am not authorized to stop production instance.





Next, When I tried to stop the development instance, the development instance could be stopped. This was because the Policy I created (and attached to the User Group that my User is part of) allowed all EC2 related actions to all EC2 instances/resources with the Env tag development.



Conclusion

I created:

An IAM User Group called **intern-dev-group** with defined permissions using IAM policy.

An IAM User called **intern-dev-nishchal** that is added to the user group.

An EC2 instance with the Env tag **development** and Name **NishchalDevelopment**.

An EC2 instance with Env tag **production** and Name **NishchalSecurityProject**.

My user could stop the development instance but could not stop the production instance. This is by design using custom policy.