# AWS IAM

## IAM – Identity and Access Management

IAM (Identity and Access Management) is a service in AWS that helps you securely manage access to AWS resources. It allows you to control who is authenticated (signed in) and authorized (has permissions) to use specific services and resources.

Why use IAM?

1. **Security**
   * Prevent unauthorized access to your resources
   * Apply **least privilege principle** (only give necessary permissions)
2. **Granular Access Control**
   * You can allow specific actions (like S3:PutObject) on specific resources
3. **Centralized Management**
   * Manage all users, roles, and policies in one place
4. **Temporary Access**
   * Use IAM roles for **temporary access**, especially for EC2 or Lambda
5. **Multi-Factor Authentication (MFA)**
   * Add an extra layer of security for sign-ins
6. **Audit and Compliance**
   * Monitor access and changes using AWS logs and IAM policies

## Components of IAM:

1. Users
2. User Groups
3. Roles
4. Policies
5. Permissions

### Users:

* A user is an identity representing a person or application.
* Each user has credentials (username, password, access keys).
* Example: You create a user named developer-ajinkya to allow login and access to AWS services.
* Use case: You want to give your team members individual access to AWS with specific permissions.

### **2. IAM Groups**

* A **group** is a collection of IAM **users**.
* You can **attach policies** to the group, and all users in the group inherit those permissions.
* Example: A group named “Developers” with permissions to EC2 and S3.
* **Use case:** Makes managing permissions easier—just add users to the group instead of assigning policies one by one.

### **3. IAM Policies**

* A **policy** is a JSON document that defines **permissions** (what actions are allowed or denied).
* You can attach policies to **users, groups, or roles**.
* Example: A policy that allows listing objects in an S3 bucket:

{

"Effect": "Allow",

"Action": "s3:ListBucket",

"Resource": "arn:aws:s3:::my-bucket"

}

* Use case: Define and control who can do what in your AWS environment.

### **4. IAM Roles**

* A **role** is an identity **you assume temporarily** to get permissions.
* Roles do **not require username/password**—they are used by AWS services or users who "assume" the role.
* Example: An EC2 instance needs to access an S3 bucket, so you assign it a role with S3 permissions.
* **Use case:** Use for cross-account access, AWS service access, or temporary access without credentials.

### **5. Permissions**

* **Permissions** are the actual **rules inside policies** that define what actions are allowed or denied.
* They are written using actions like s3:PutObject, ec2:StartInstances, etc.
* Example: Giving a user permission to only start and stop EC2 instances.
* **Use case:** Fine-grained access control over AWS services.

Practical use:  
Creation of User and user group is based on each other so we will create groups first and then users.

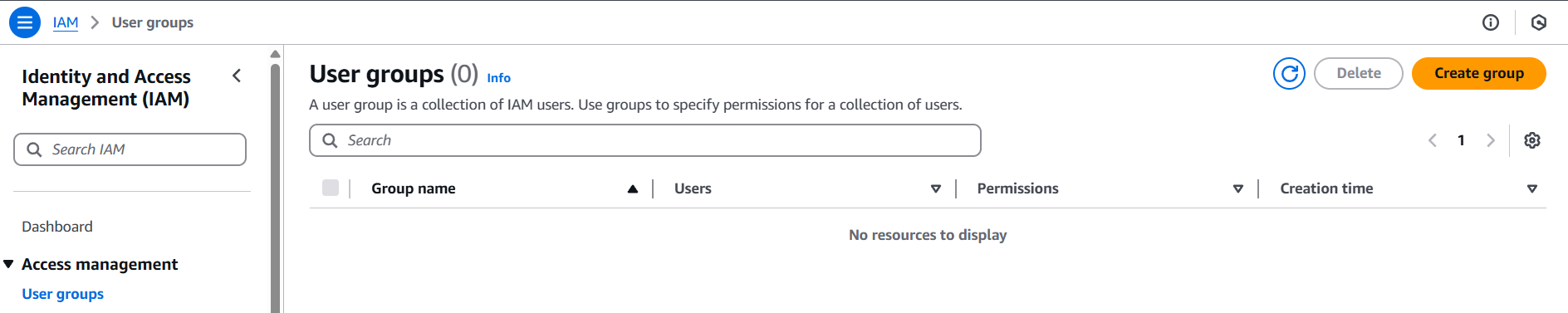
We will create 3 groups:  
**Admin:** Allow permissions to users in the group to access account almost like root user (Administrator access).

**EC2:** Allow permissions to users in the group to access services related to EC2 service.

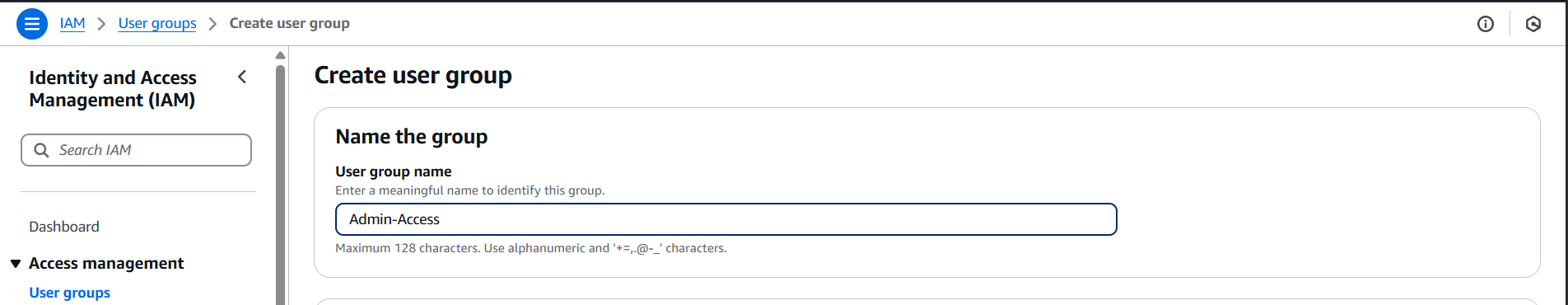
**S3:** Allow permissions to the users in the group to access services related to S3 service.

### Creating groups:

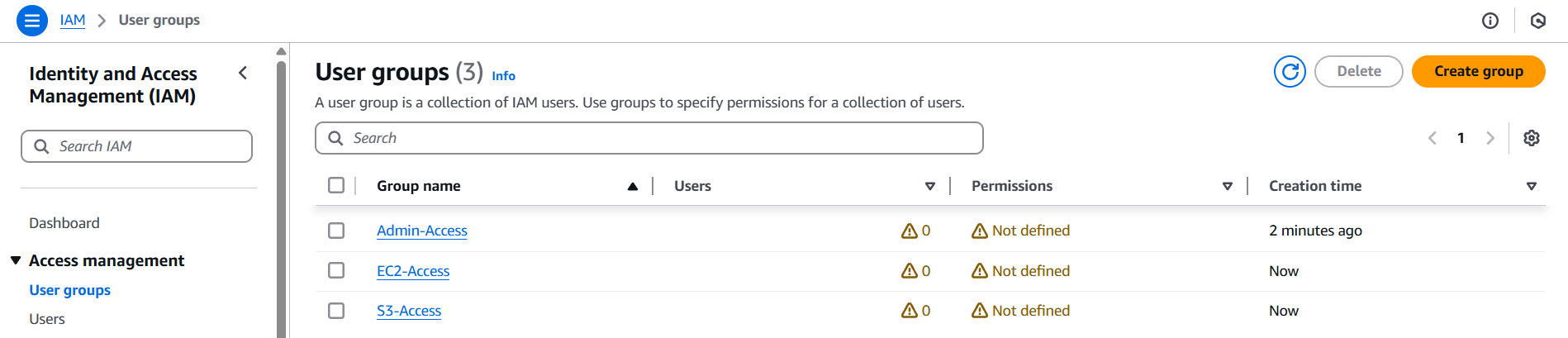
* Open user groups interface from IAM dashboard and menu.



* Click on “Create group”.
* Enter the name of the group (Admin-Access).
* Keep “add user …” and “permissions” section as they are and click on “create user group”.
* We will do those things later.



* A user group named “Admin-Access” will be visible on the “user groups” dashboard.
* Create two more groups named “EC2-Access” and “S3-Access” in the same way.
* So, now we have 3 groups with no users and no permissions.



We will create 3 users:

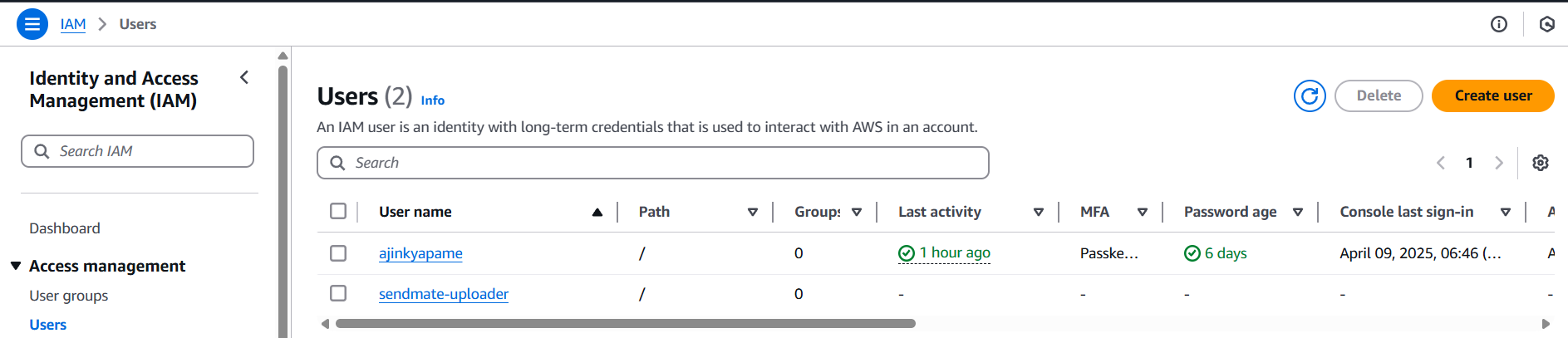
**Admin-user:** Part of Admin group.

**Ec2-user:** Part of EC2 group.

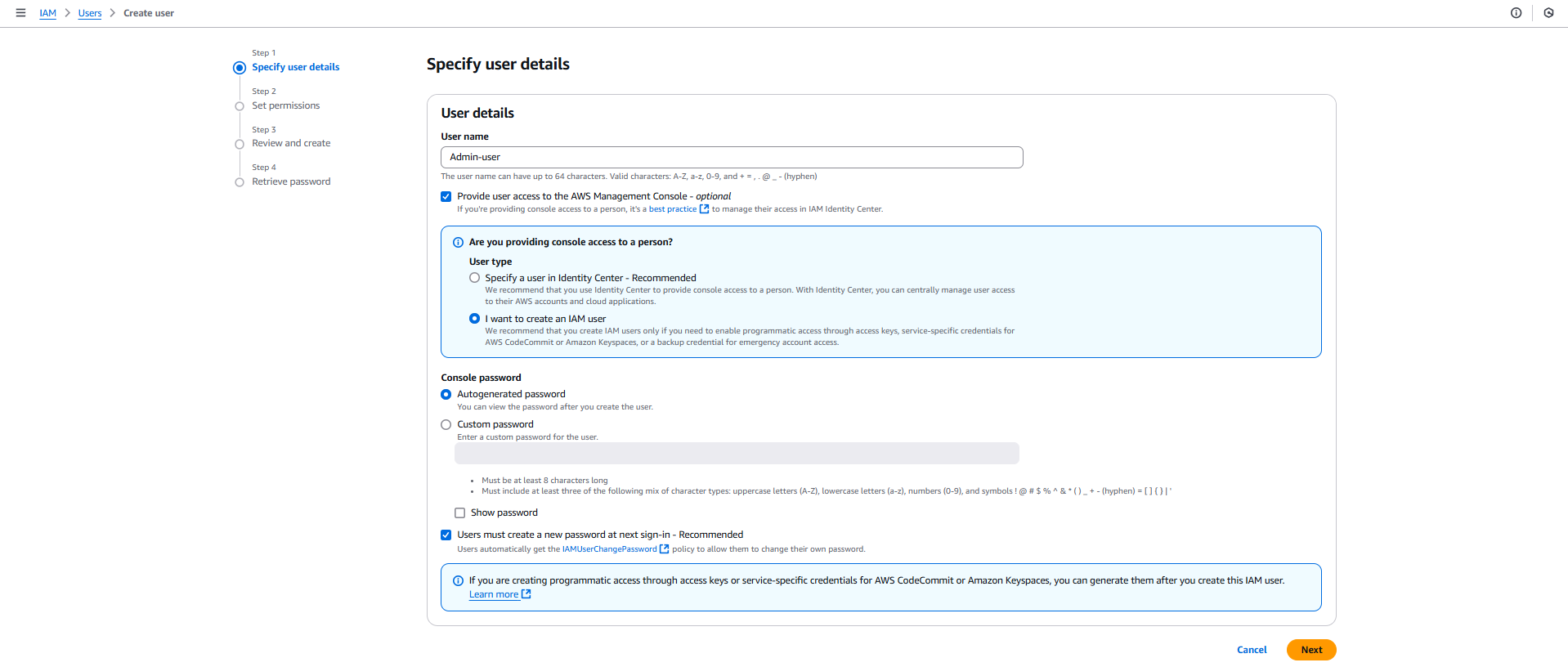
**S3-user:** Part of S3 group.

### Creating users:

* Open users interface from IAM dashboard and menu.



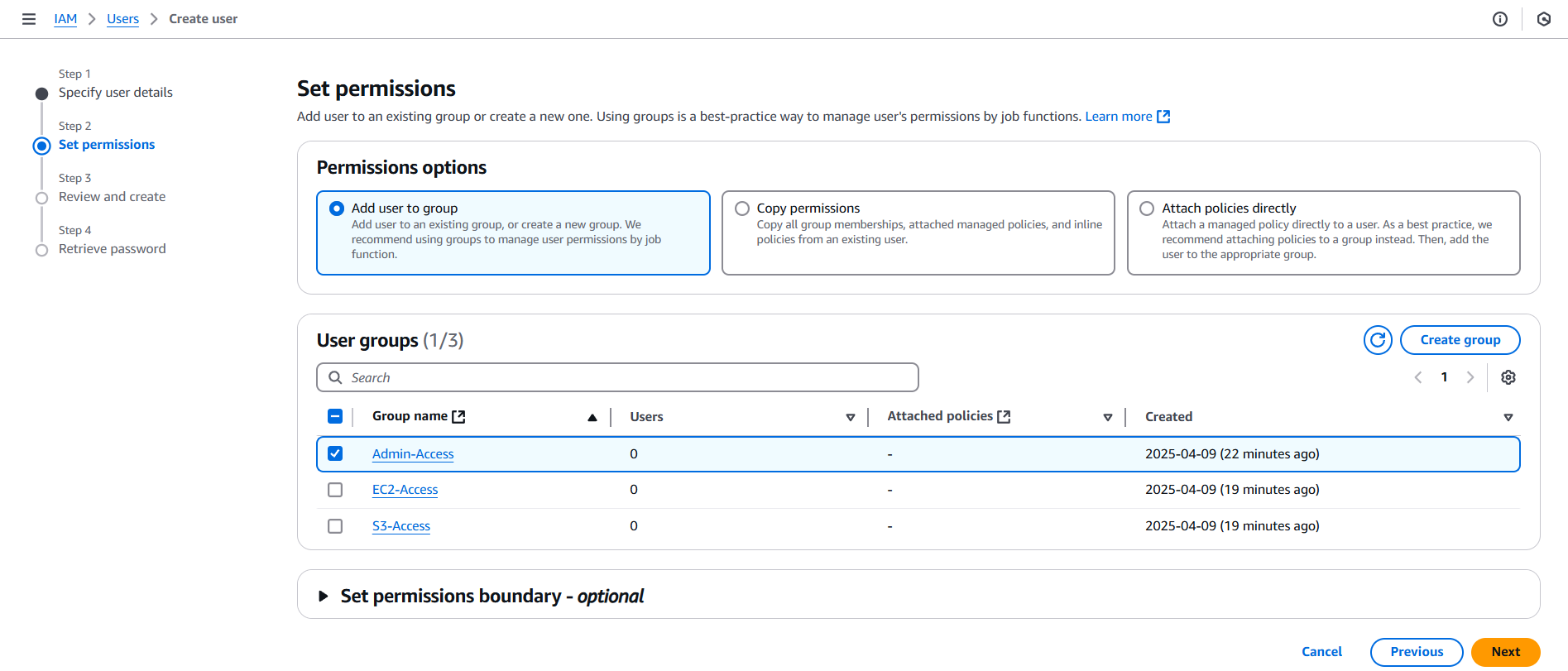
* Click on “Create user”.
* Enter name of user (Admin-user).
* Tick “Provide user access to …”.
* It will expand and ask “Are you providing console access to a person?”.
* Select “I want to create an IAM user”.
* Keep option as “Autogenerate password”.
* Don’t forget to tick “Users must create a new password at next sign-in – Recommended”. Click on next.
* Why we tick that?  
  When working on real project, IAM role assigning person creates the user and password for it but later on only the user means the person who is the user should only have that password. This helps to avoid any issues related to the usage by that user (account).



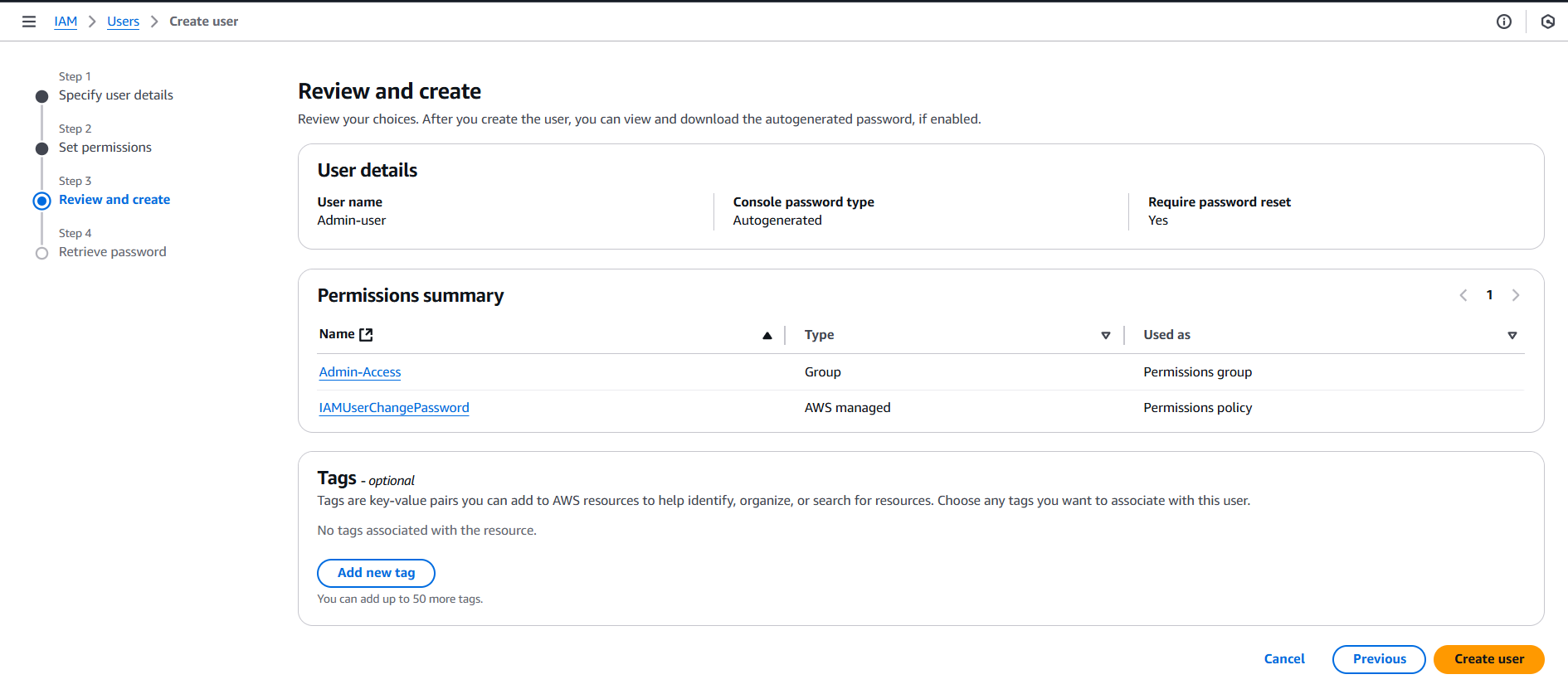
* The next step is to set permissions.
* There are three options:

1. Add user to groups: means add the user to the existing group/s that is/are already present.
2. Copy permissions: Copy the permissions of the user that is already present.
3. Attach policies directly: Attach the policy that is custom or AWS managed directly to the user. Use this in case you don’t need/want to put that user to any group.

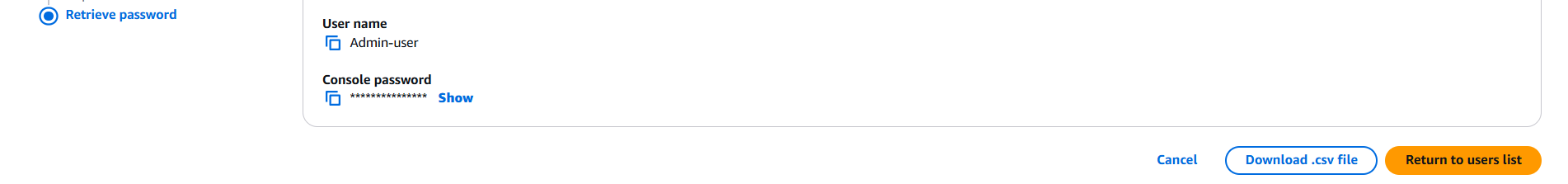
* We will go with “Add user to groups”, as we have created groups.
* So, choose that option and select “Admin-Access” group from present user groups. Click on next.



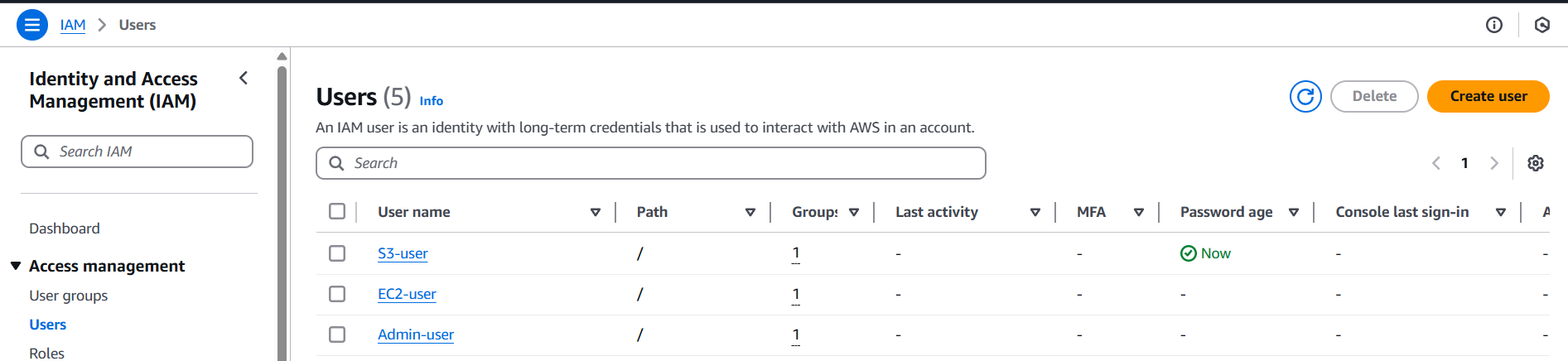
* Now we are at “Review and create” step where we will review all details of user group and make sure “Require password reset” is set to ‘yes’ and permission summary has the permissions that we gave (Admin-Access group and change password).



* Click on create user and the user will be created.
* Without fail download the .csv file using download option that is available on the immediate next page (Remember this option is available only once).



* That csv file contains username, console password and console sign-in URL.
* All these things we need to sign-in to the “Admin-user” console.
* Create two more users in the same way using respective groups and download their .csv files (EC2-user, S3-user).
* So now we have 3 users each associated with respective groups.

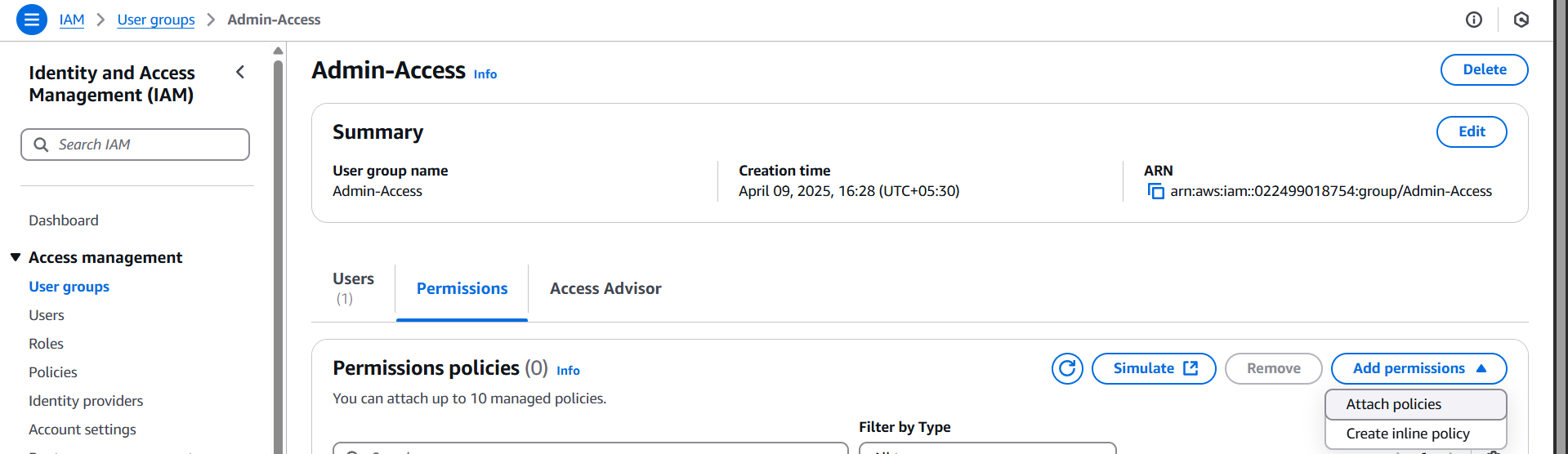


By now we have created 3 groups as well as 3 users and they are connected but we have not attached any kind of permissions to anyone of them. Without permission association there is no use of these groups and users so let’s attach them permissions.

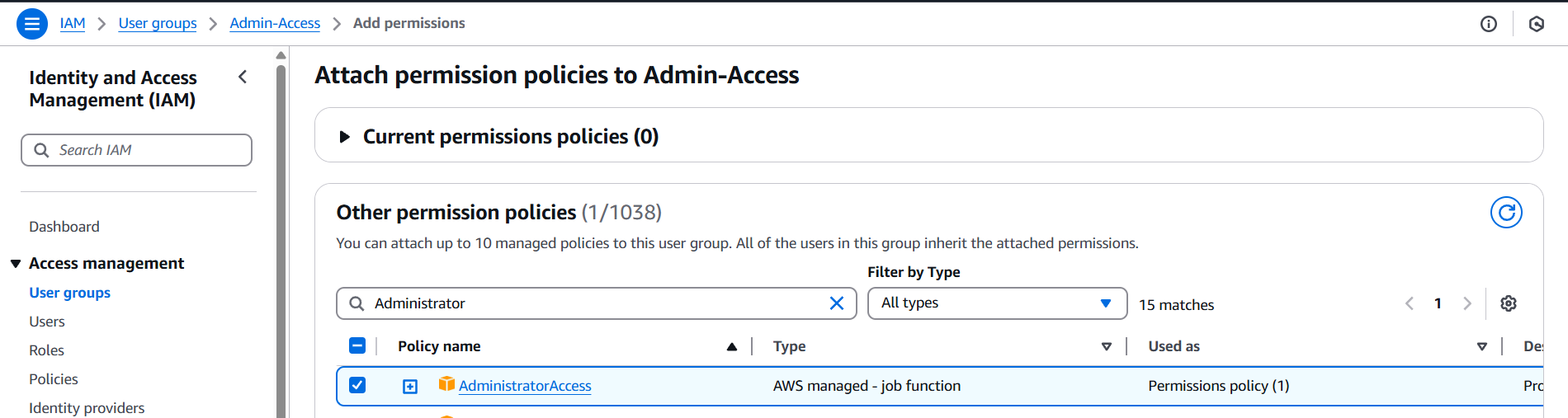
Permissions are mentioned in the policies. Attaching policies means attaching permissions. A policy is made up of one or more permissions so using single policy we can give multiple permissions to the user/group.

### Attaching permission/policy:

* We will attach the “permission policies” to the groups and they will be attached to the users present in them.
* Go to “User groups” dashboard and select “Admin-Access” group.
* Select “permissions” tab there and from dropdown of “Add permissions” select ‘Attach policies”.



* Search for “AdministratorAccess” and select it.
* Click on “Attach policies”.
* AdministratorAccess: this policy allows user to access account almost similar to root user with some restrictions.
* It is strongly suggested to use AWS services via this user and avoid use of account as a root user.
* If you click on that policy name you can see that 439 out of 439 permissions of AWS are defined in it.



* Do the same for EC2-Access and S3-Access groups and attach their full access policies to them.
* EC2-Access: AmazonEC2FullAccess
* S3-Access: AmazonS3FullAccess
* Now the users present in the groups have access to use only the services of which permissions are defined.
* The setup of users, their groups and permissions is complete. Now we can login to respective console and check that only limited access is given.

Before that we will see about IAM alias.

### IAM alias:

* An IAM alias is a custom URL name you can create for your AWS account’s sign-in page, so it’s easier to remember and more branded than the default.
* By default, the AWS sign-in URL looks like this:

https://123456789012.signin.aws.amazon.com/console

* After creating an alias, it can look like:

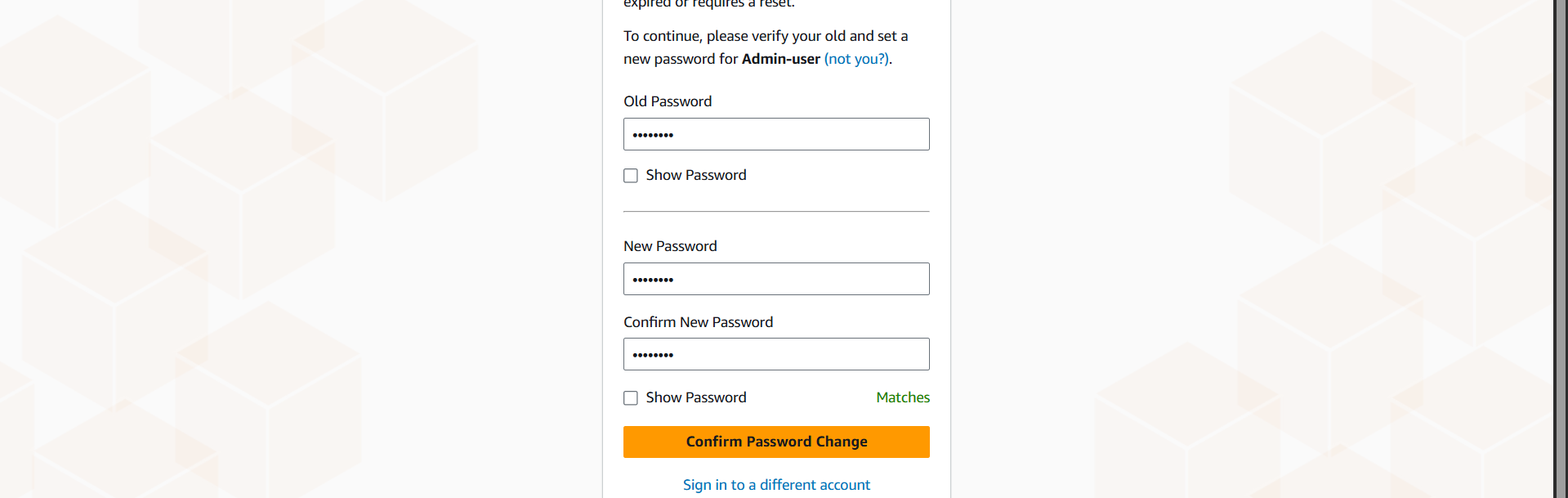
https://my-company.signin.aws.amazon.com/console

**Why Use an IAM Alias?**

* Makes login easier to remember.
* Looks more professional, especially for teams or organizations.
* Useful if you're managing multiple AWS accounts.
* To create an alias, go to dashboard of IAM and on the right side look for “AWS Account” section which has “Account alias” and create option below it.
* Click on that and enter the alias of your own.
* Create the alias and we will use it ahead in the practical to login to the console of those three users.

### Logging to Admin-user:

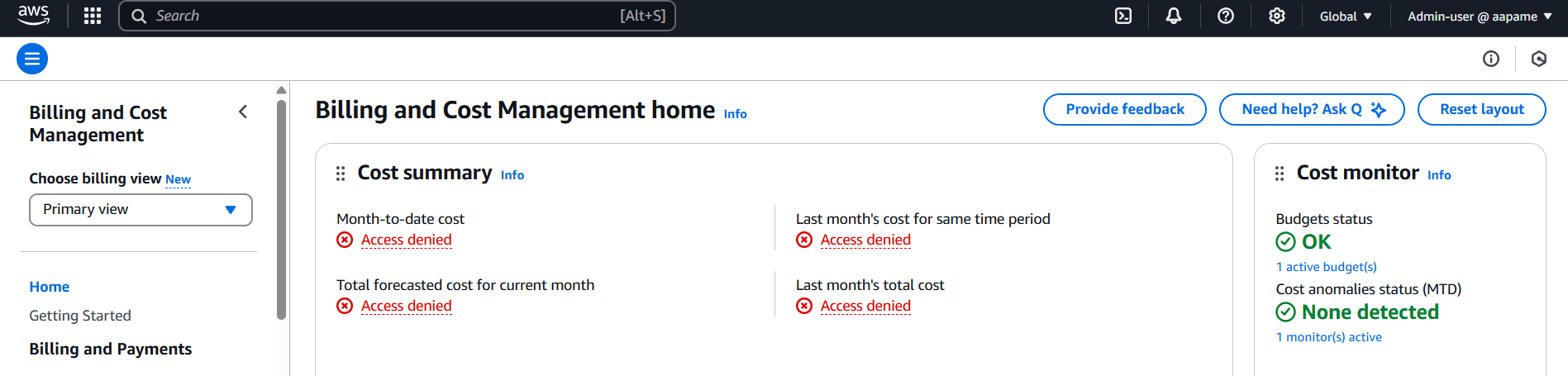
* Open the .csv file that was downloaded for Admin-user.
* Copy the console URL from there and browse it in the browser.
* Enter alias, username (Admin-user) and the password that will be available in the .csv file.
* Click on sign-in and it will take to the password change page because we opted for password change while creating the user.
* Enter old password (password from .csv file), new password of your own and again in confirm password.



* Click on “Confirm password change”.
* Click on Continue to sign-in on next page and now we are logged in as “Admin-user” to our own account. Check in the right-top corner there will be username@alias and not the name that appears for root user.



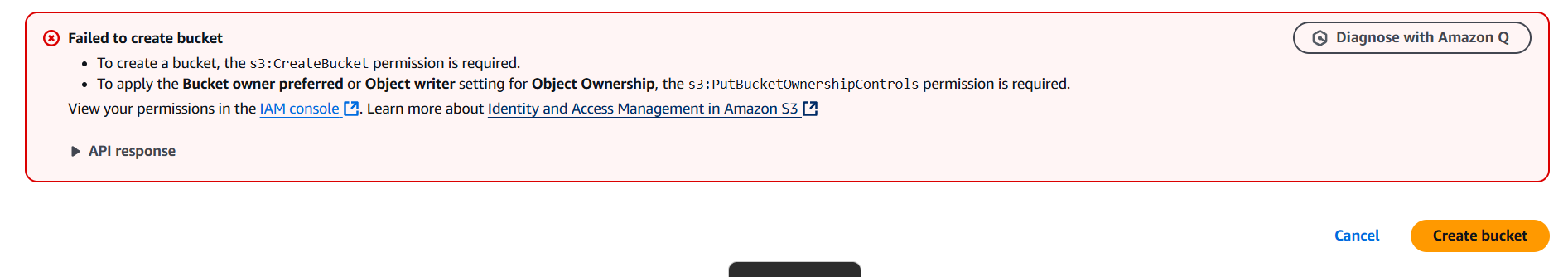
* This user has permission to almost every service but has restrictions on some services that are made available only to root user.
* If you try to launch an instance, create a S3 bucket or VPC creation, you can do that but you cannot access the “bills and payment” section as it is only root user specific.



* Let’s try same for other users.

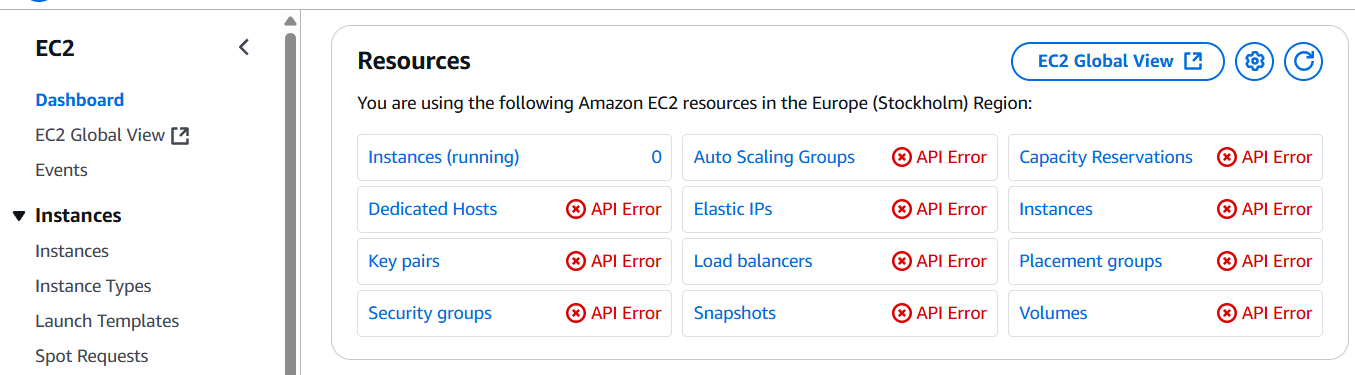
### Logging to EC2-user:

* Use the same steps as Admin-user to login to the console (open .csv file, enter URL, enter username & password and change the password).
* On the console page itself we can see that access to list the applications is denied.
* Using this user, we can only use the EC2 related services.
* To check for denied permission, try to create a bucket and there will be access denied error.



### Logging to S3-user:

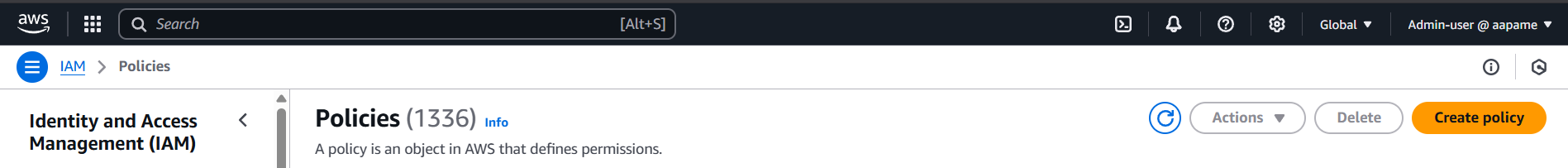
* Use the same steps as Admin-user to login to the console (open .csv file, enter URL, enter username & password and change the password).
* Same like Ec2-user, on the console page itself we can see that access to list the applications is denied.
* Using this user, we can only use the S3 related services.
* To check for denied permission, open EC2 dashboard and there will be access denial to every service.



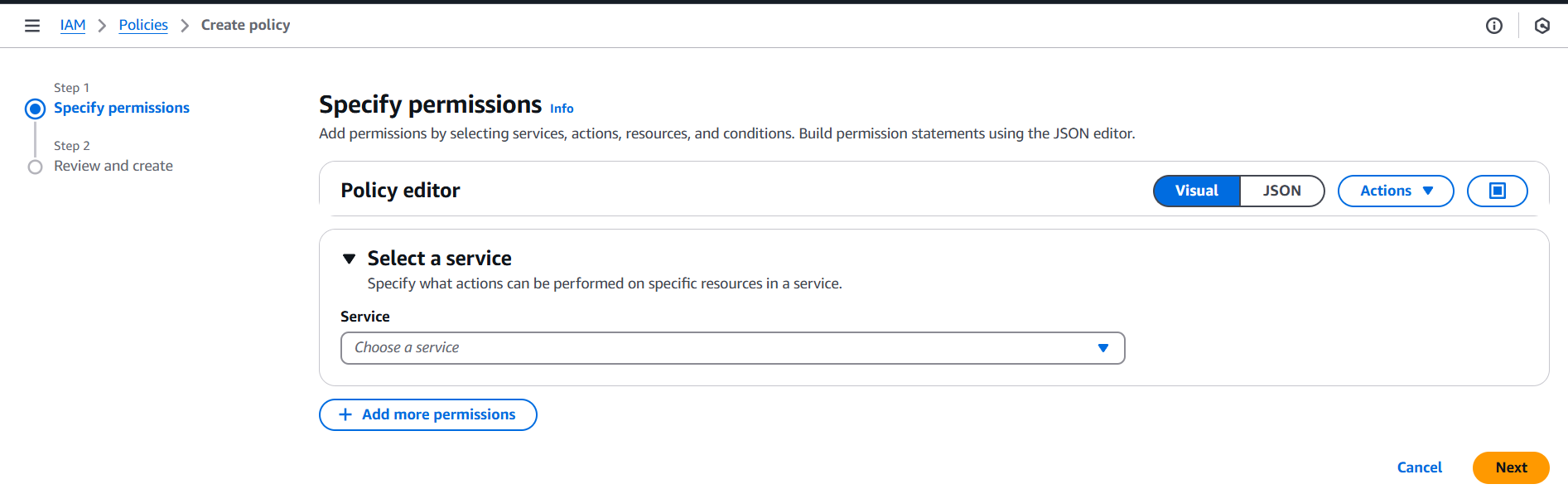
* If we add S3-user to EC2-Access group to allow access of EC2 services and vice versa then both of them can access each other’s services.

### Creating a Policy:

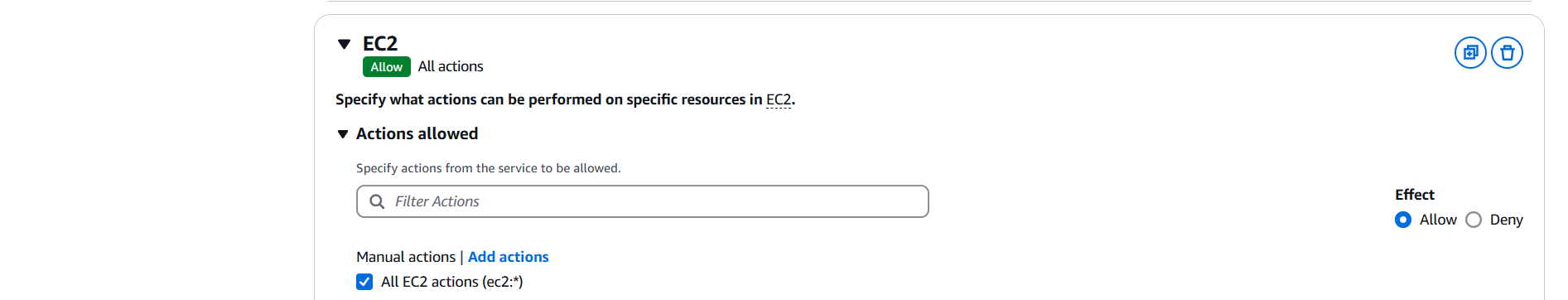
* We will create a policy that has permission of both S3 and EC2 services and attach it to the EC2-Access group.
* This will allow EC2-user in that group to use S# services as well and we can remove the previous EC2FullAccess permission as it will be already present in our custom policy.
* Go to policies from sidebar and there we can see all the policies that are AWS managed.
* Click on “Create policy” button to create our own custom policy.

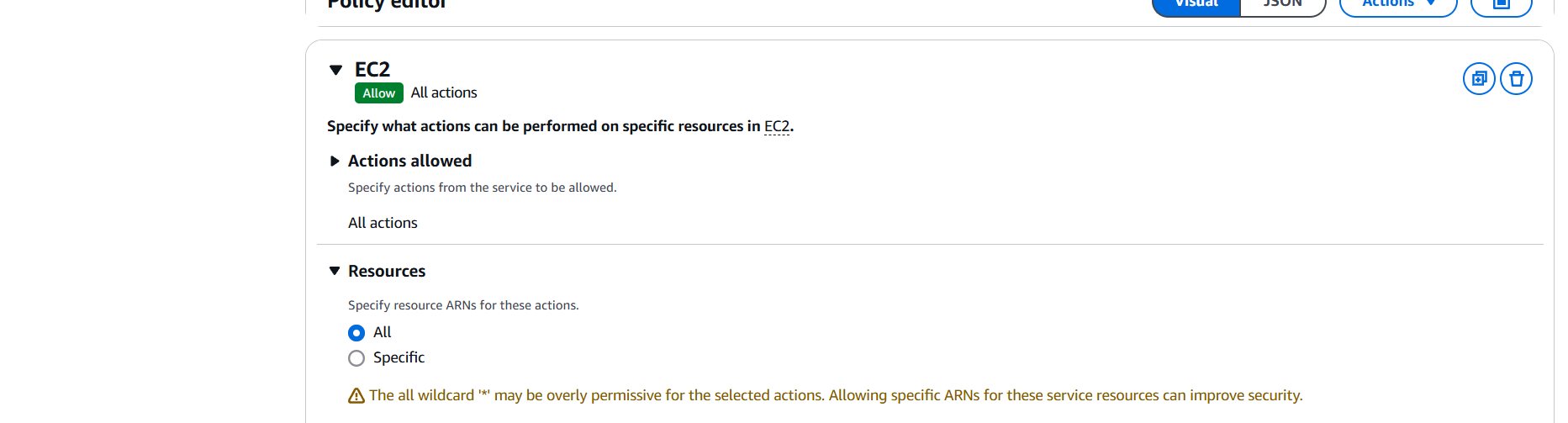


* Remember one thing I am doing all these things as Admin-user and I do not need to login as root user.
* From drop down choose EC2 service.

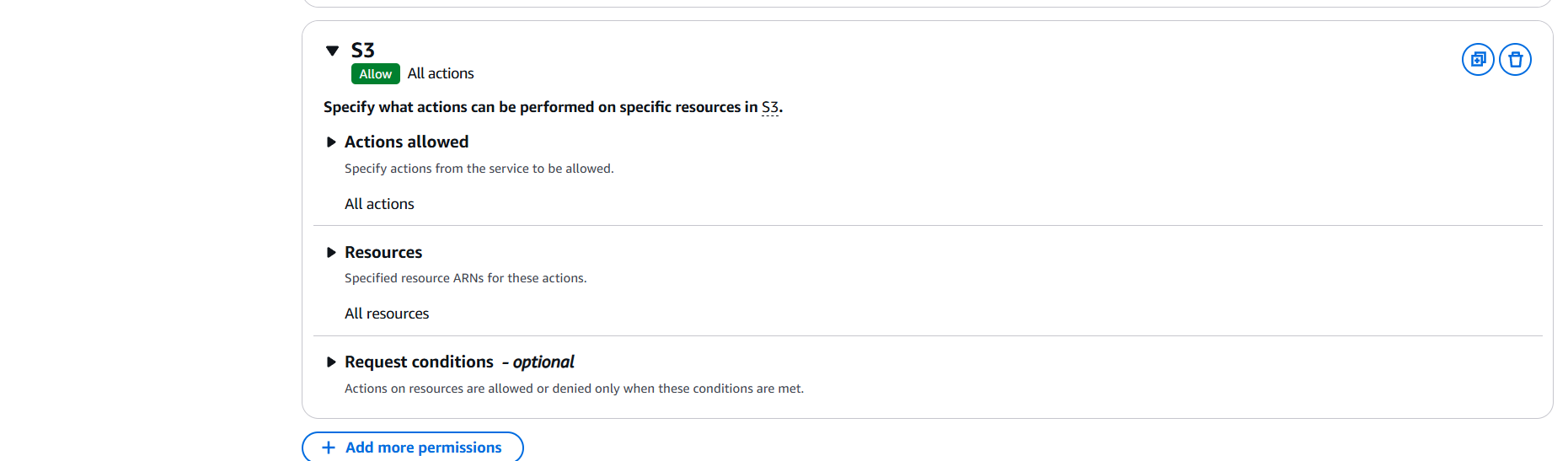


* Tick the box of all actions to allow usage of all actions related to EC2 service to whoever is accessing the policy.
* Tick the box of all resources to allow usage of all resources related to EC2 service to whoever is accessing the policy.
* In short, we are giving the EC2FullAccess permission but we can give only read instances, list instances, etc. type of access only through this dashboard.

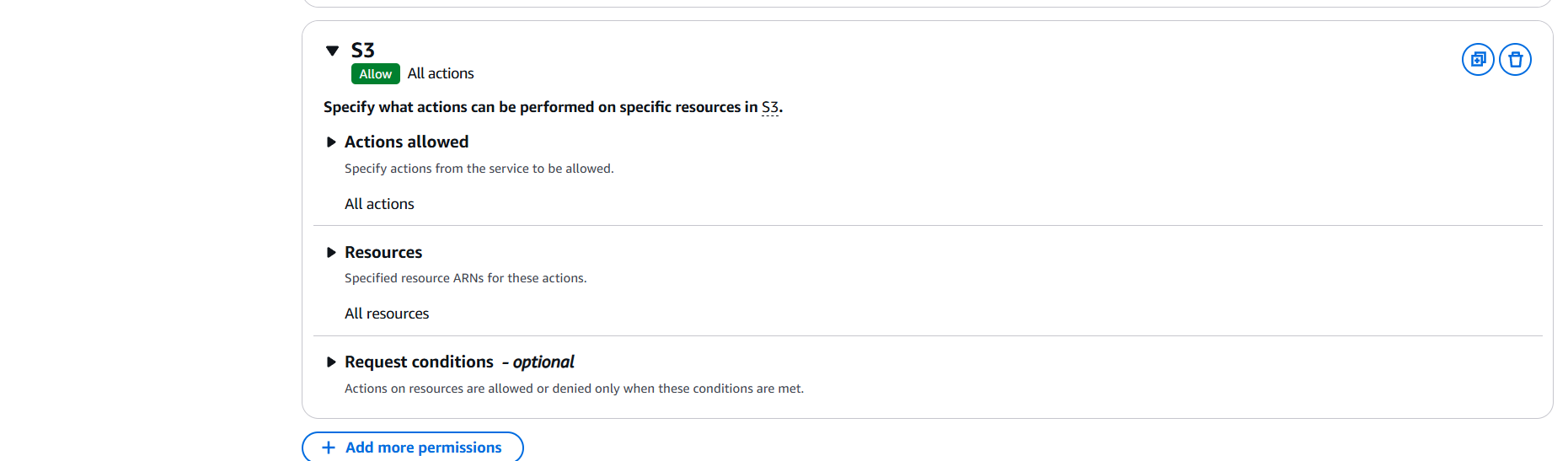




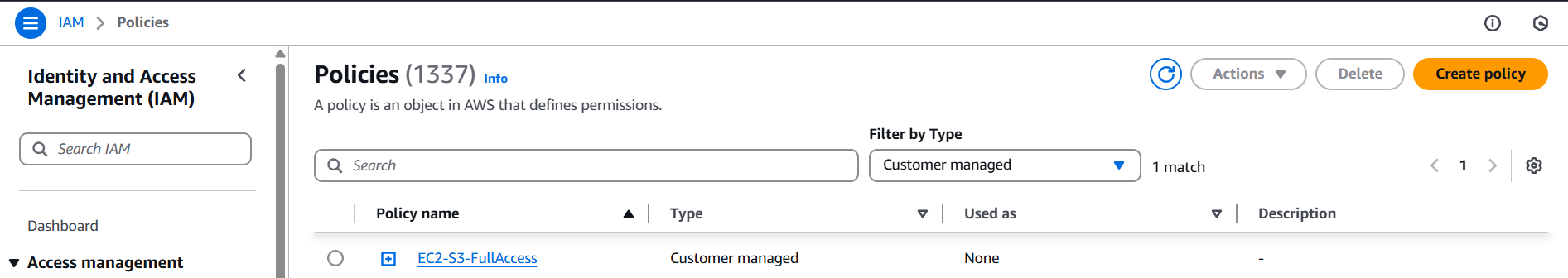
* Click on “Add more permissions” and do the same by selecting S3 service.



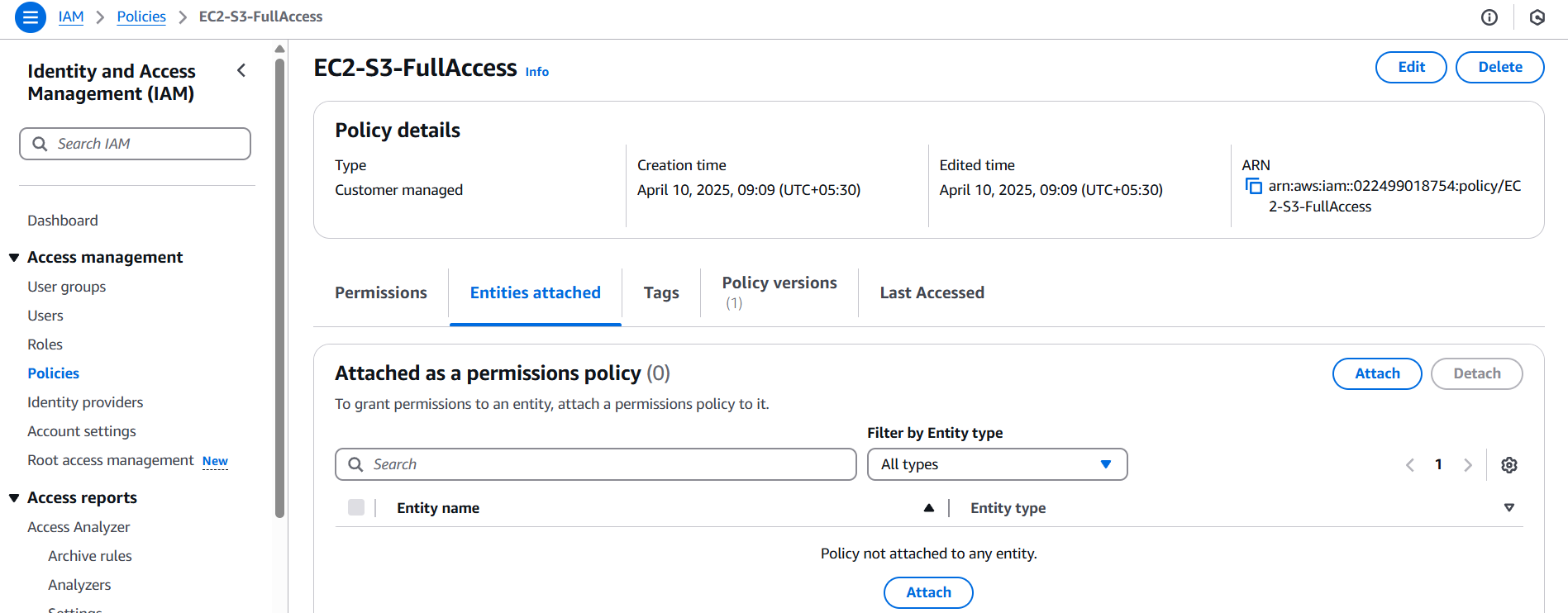
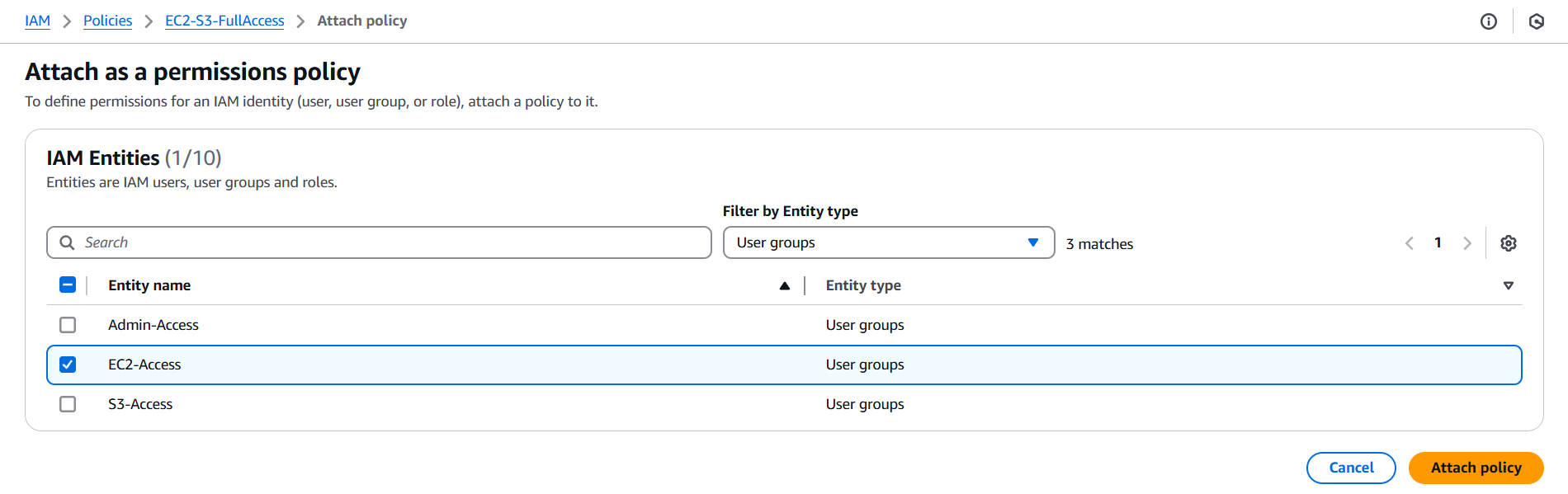
* Click on next and enter name of policy and create the policy.



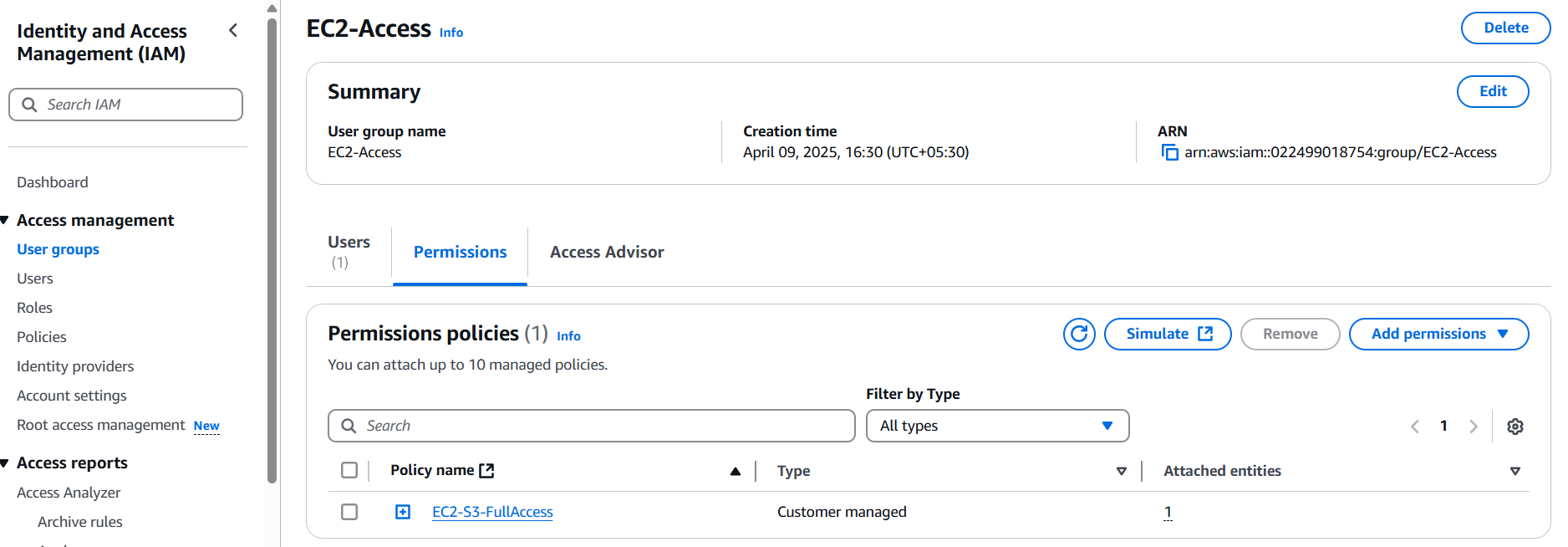
* Go to policies dashboard again and choose “Customer managed” from “Filter by Type” dropdown. We can see our policy.



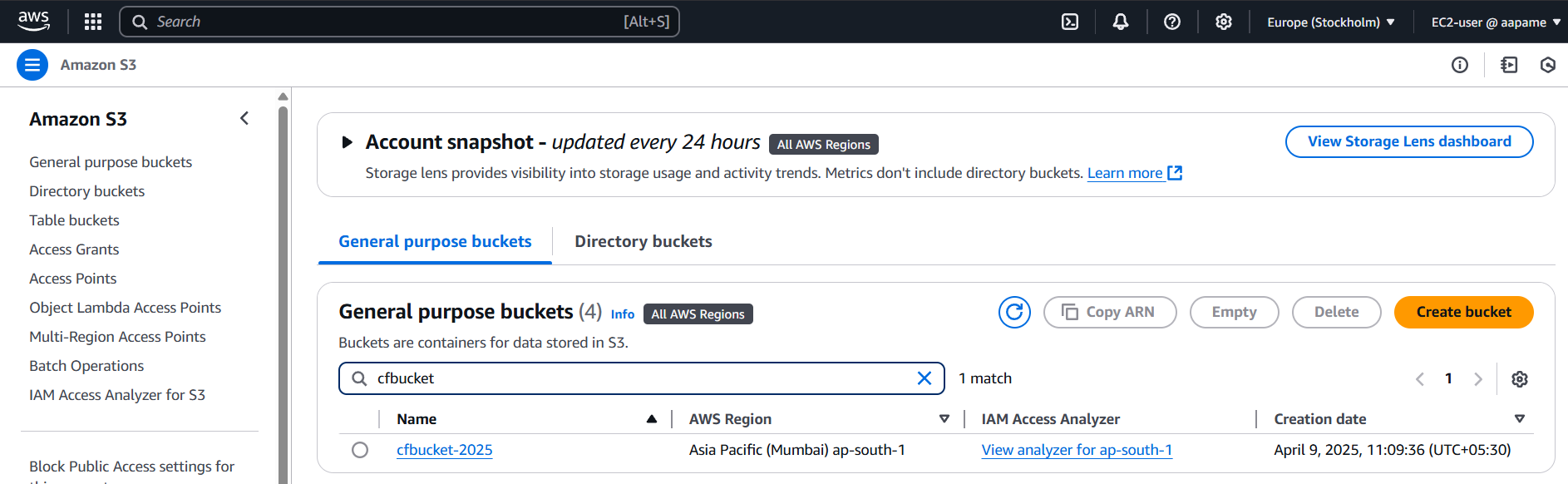
* Click on that policy name and open “Entities-Attached” section and click on attach to attach it to the EC2-Access group as discussed.
* Select the EC2-Access group from there and click on “Attach policy”.

* Now if you try to login as EC2-user and access S3 service, you will be able to do that and also remove EC2FullAccess permission of EC2-Access group as it is of no need.



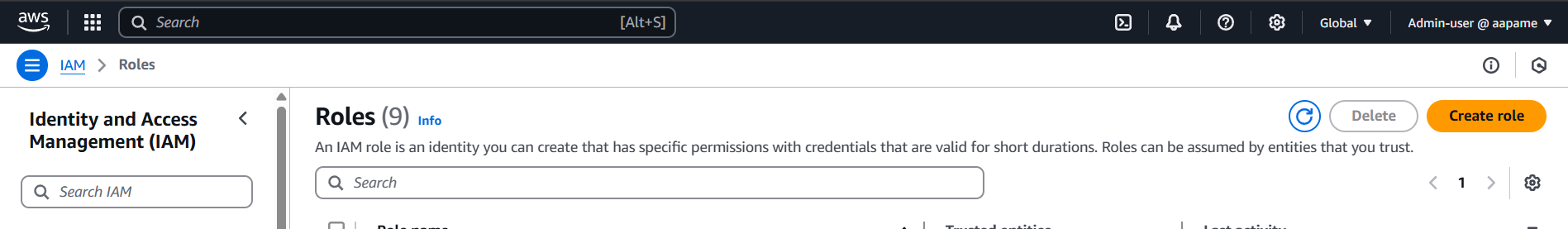
* I have removed the EC2FullAccess permission and let’s check can I access the S3 service?



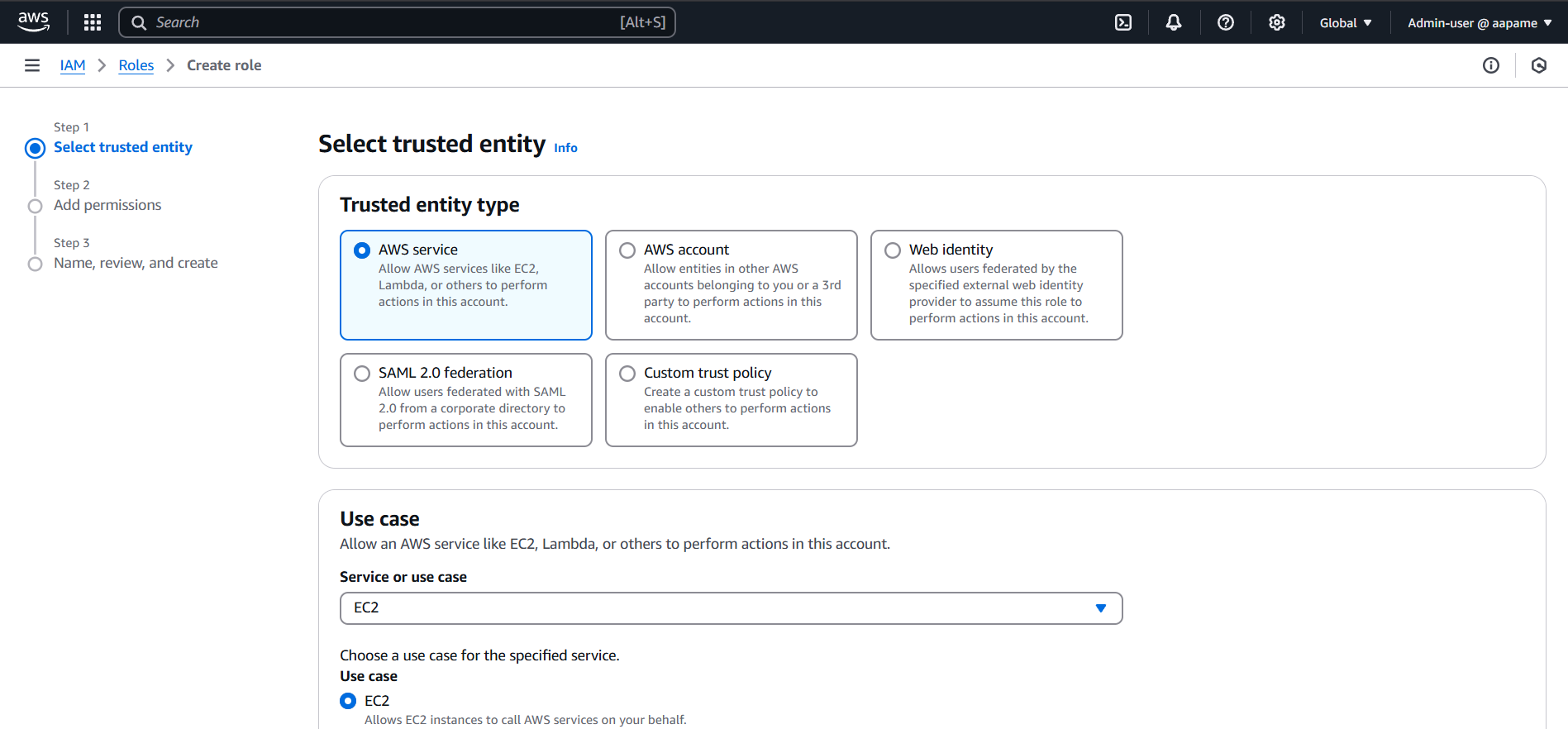
* I could list the S3 buckets without any error.
* This is how we can create custom policies and attach to specific user or group and access the services.

### Creating an IAM Role:

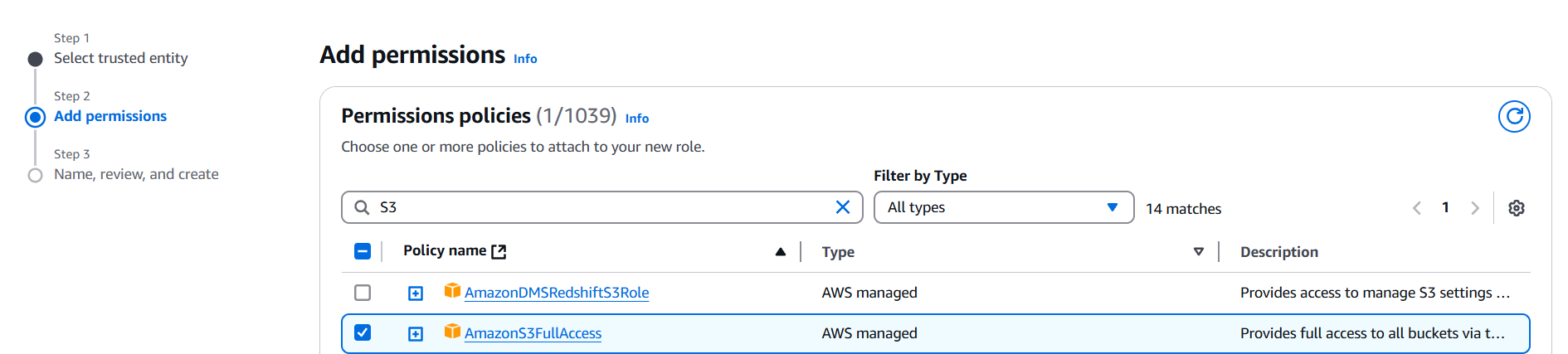
* Go to IAM role dashboard using Roles option from sidebar.
* There can be already some roles present that may have been created by AWS during our any usage.
* We will create a role of type EC2 which allows EC2 instance to access S3 service means that instance can use S3 service and create, delete, etc. do anything in that because we are assigning a role to it.
* Click on “Create role” button.



* Select “Trusted entity type” as AWS service as we are going to assign the role to EC2 instance which is an AWS service.
* Choose EC2 in use case and click on next.



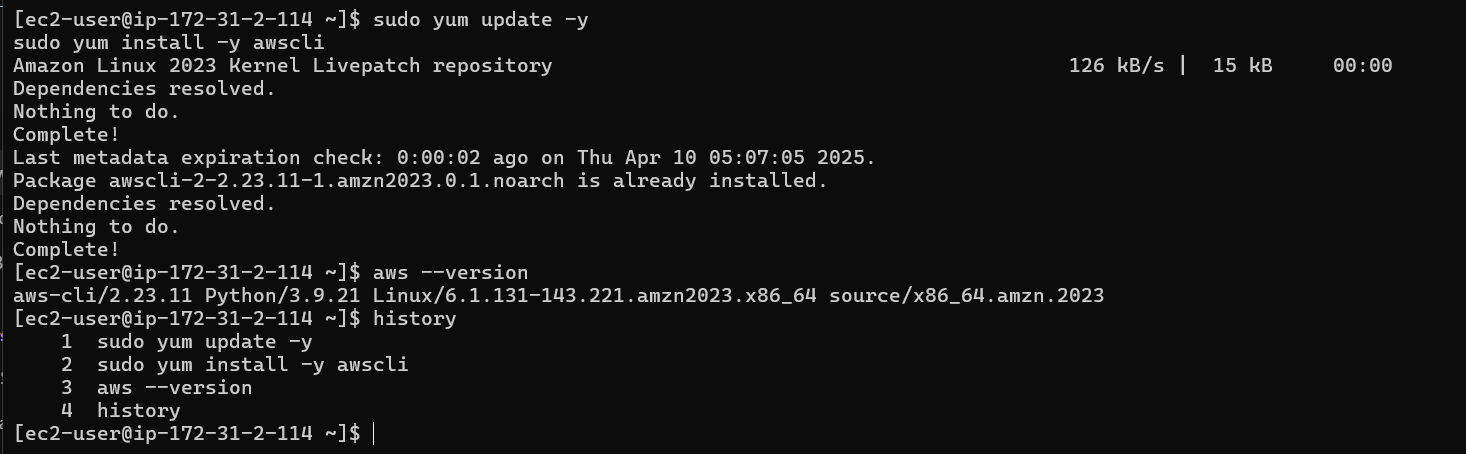
* Search for S3 and choose “AmazonS3FullAccess” so that the instance can access S3 service and click on next.



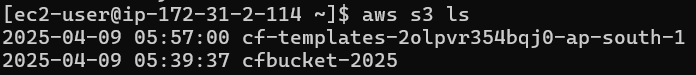
* On the next step, enter the name for role (ec2-s3-access-role) and create the role.
* Now go to EC2 dashboard and start launching the instance.
* Enter instance name, choose AMI & OS (Use Amazon Linux image) , key-pair, instance type and SG.
* Now before launching instance click on “Advanced details” below storage option and click on “IAM instance profile”.
* Select our role from there and launch the instance.



* Connect to the instance via ssh and run below commands to download AWS CLI and SDK. Check version to confirm installation is complete.



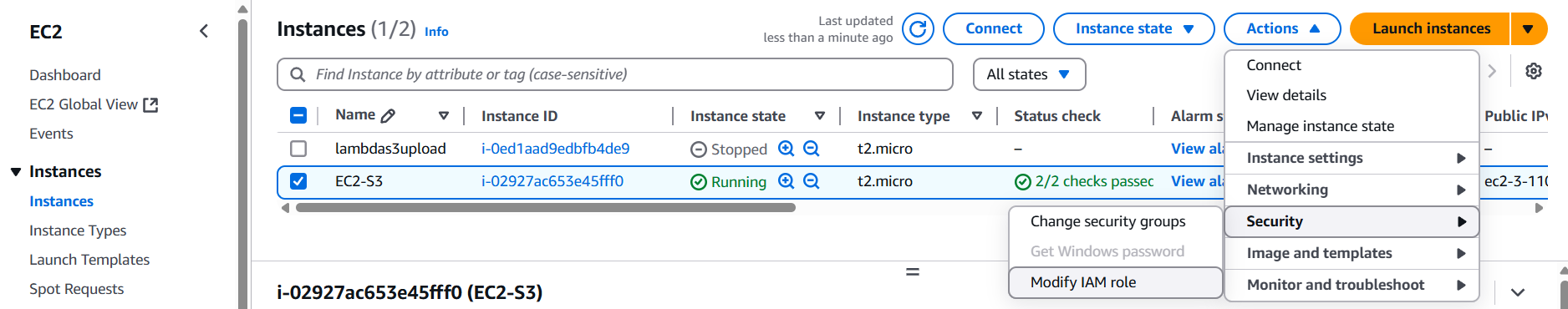
* Now if I run command to list buckets in my account, I can list those.



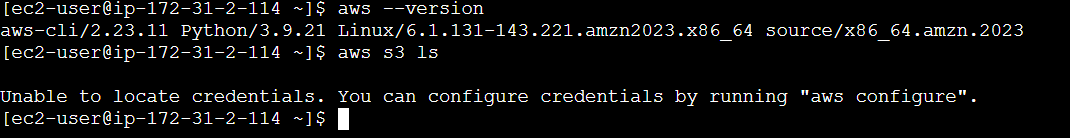
* What would I need to do to access S3 service if I don’t do this?

If we do not add the role to the instance then we need to configure CLI and give AWS access key, security key and region name explicitly. Hardcoding these things is not a good practice and complex compared to role assigning way.

* We can assign/remove the role even after an instance is launched.
* To do so, Select the instance, click on actions and choose “modify IAM role” from security option.



* From dropdown choose “No IAM role” to remove the role from instance.
* Click on “Update IAM role” and enter “Detach” in pop-up and detach the role from instance.
* Now, if we again go to powershell and try to list S3 buckets from there then we will not be able to do so.



* In this way, we can use IAM roles seamlessly to allow any AWS service to use other services easily without hardcoding the configurations.

### Multi-Factor Authentication (MFA):

Multi-Factor Authentication (MFA) adds an extra layer of security to your AWS account by requiring not just a password, but also a second factor—usually a time-based one-time password (OTP) from an app like Google Authenticator or Authy.

* This helps protect against unauthorized access, even if a user's password is compromised.
* It is strongly recommended to enable MFA for the root user and privileged IAM users, especially those with administrative permissions.
* To enable MFA, go to the **IAM dashboard**, choose the **user**, and under the **Security credentials** tab, follow the steps to **assign an MFA device**.

In this document, we explored how AWS IAM (Identity and Access Management) helps secure and manage access to AWS resources. We covered the creation of IAM users, groups, roles, and policies, and how to assign appropriate permissions to ensure that users have access only to the services they require.

We also discussed the use of IAM alias for a cleaner login experience and demonstrated how to test user access by logging in to the AWS Management Console with different user roles. The document further explains the creation of custom policies and how IAM roles allow EC2 instances to access services like S3 securely without hardcoding credentials.

Additionally, we introduced Multi-Factor Authentication (MFA), which significantly improves account security by requiring an extra layer of verification during login.

Overall, IAM not only enforces the principle of least privilege but also streamlines user and resource management, making it an essential part of any secure AWS setup.

♣♣♣