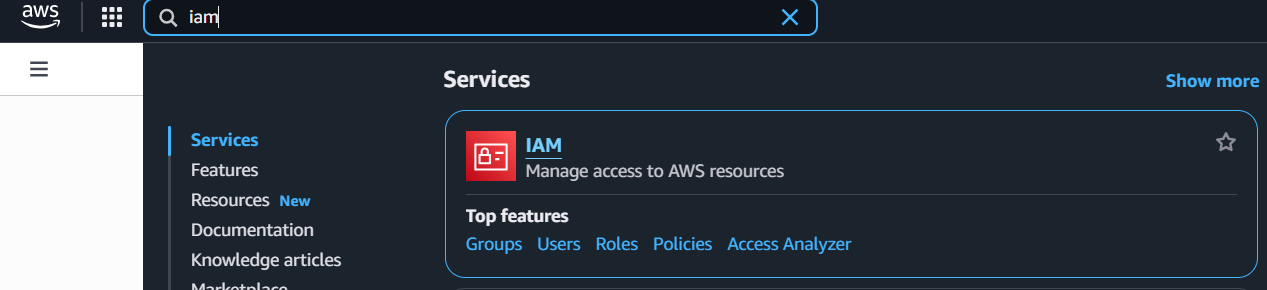
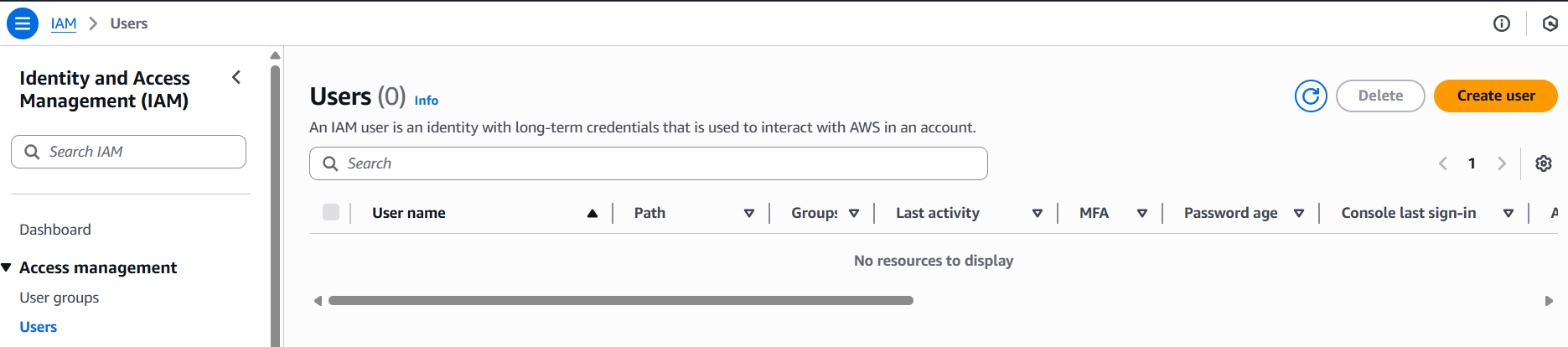
**Tasks on IAM**

1) Create one IAM user and assign ec2,s3 full access role.

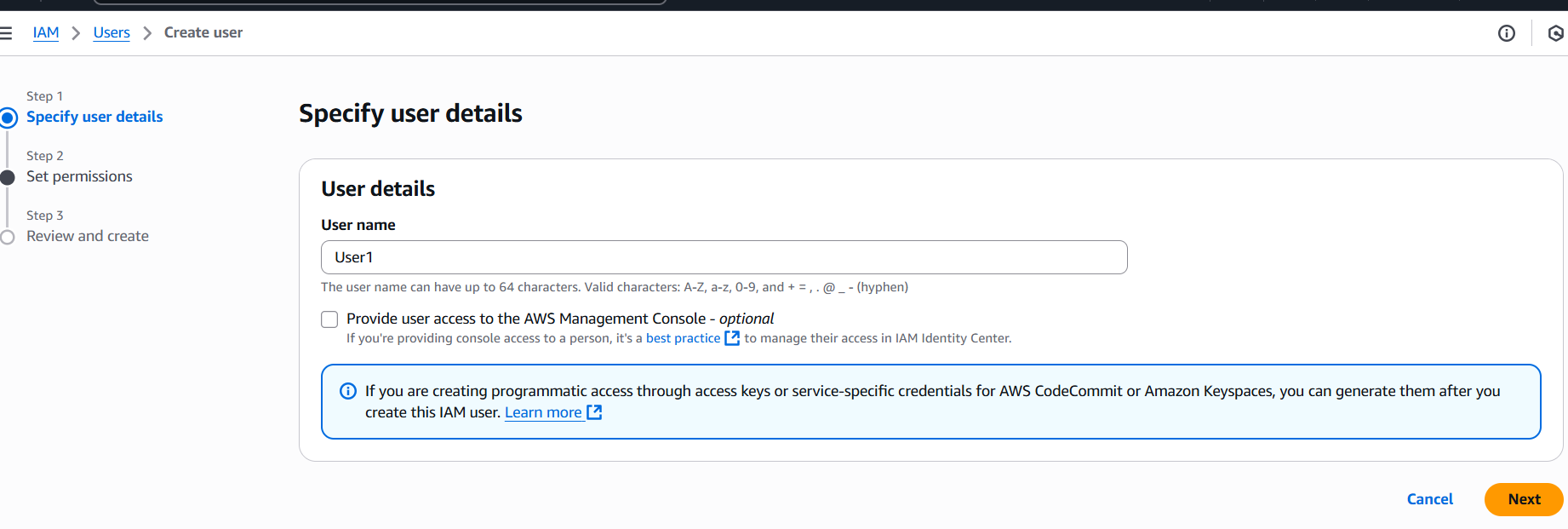
* From the AWS Management Console , search for IAM in the search bar



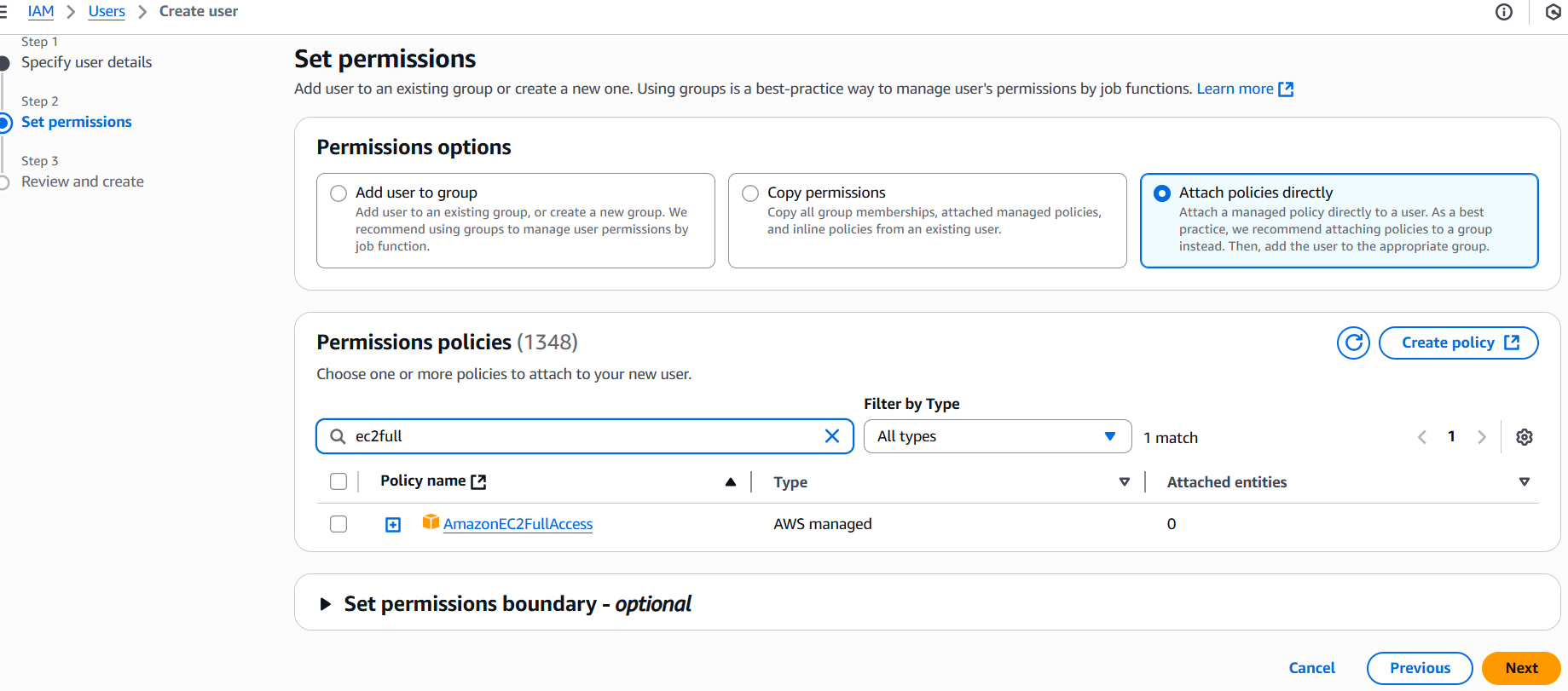
* In the left navigation pane, click Users. Click the Create user.



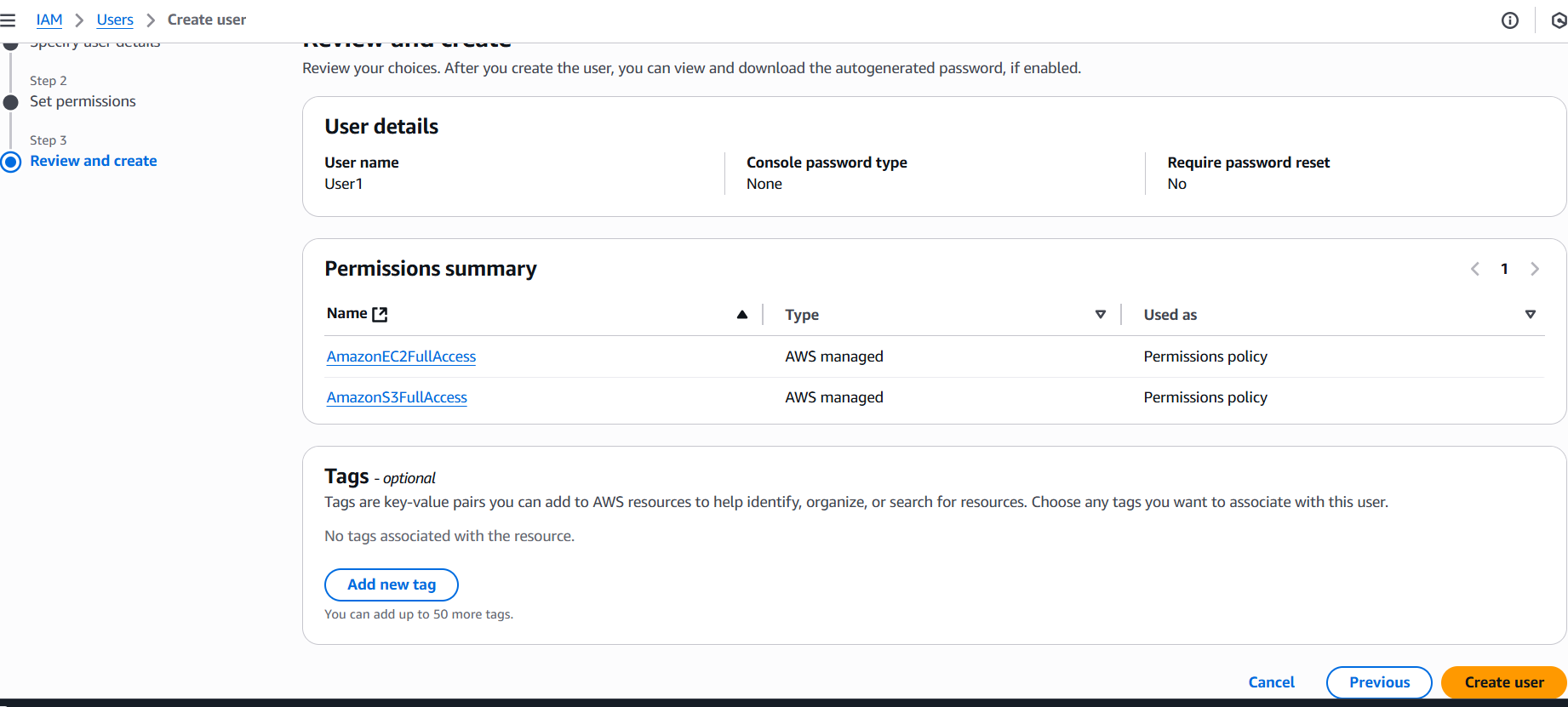
* Give the user a name.



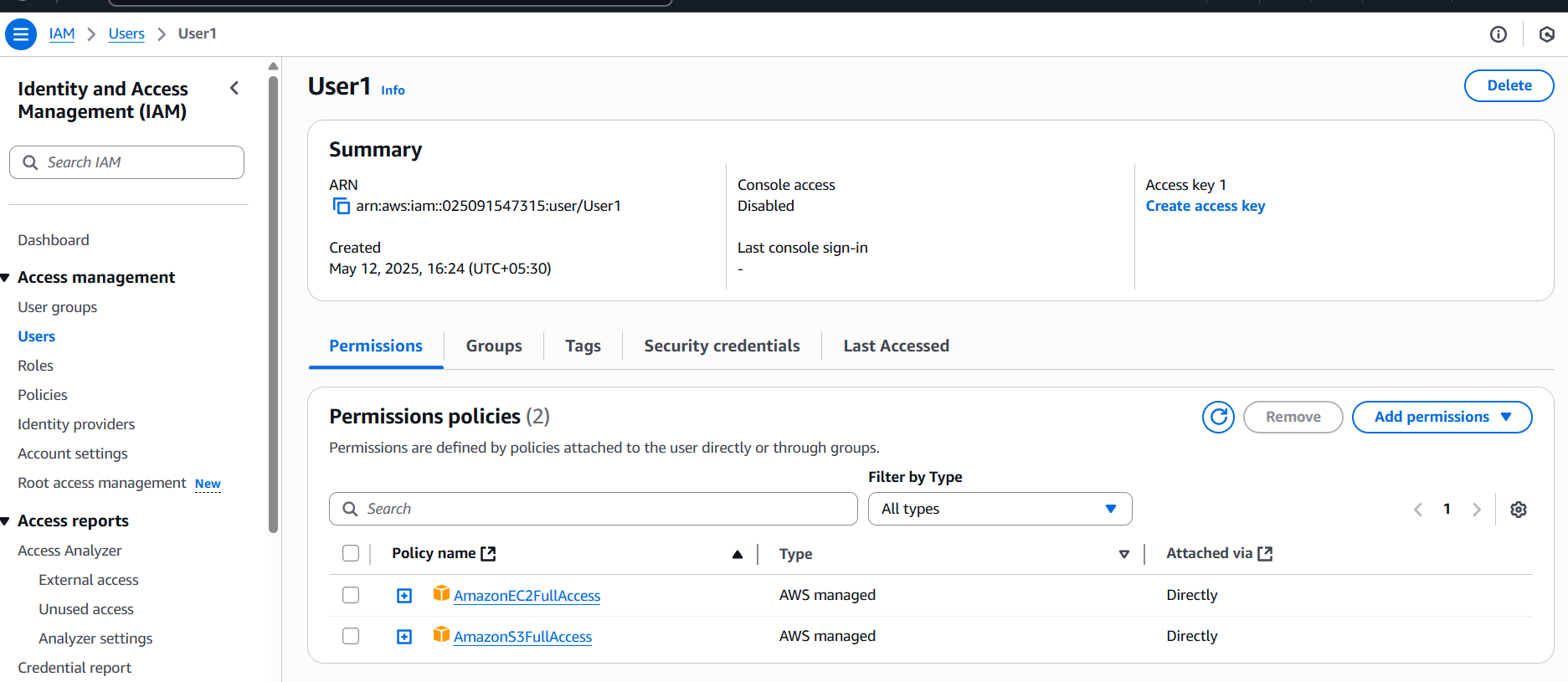
* On the "Set permissions" page, select Attach policies directly.



* On the search box, type AmazonEC2FullAccess and AmazonS3FullAccess. Select the checkbox next to the policy and click Next.

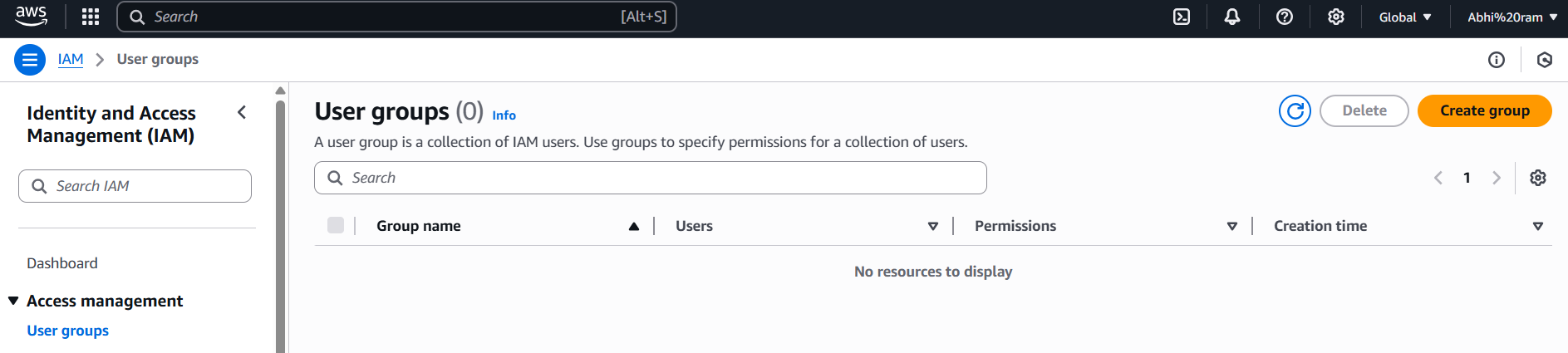


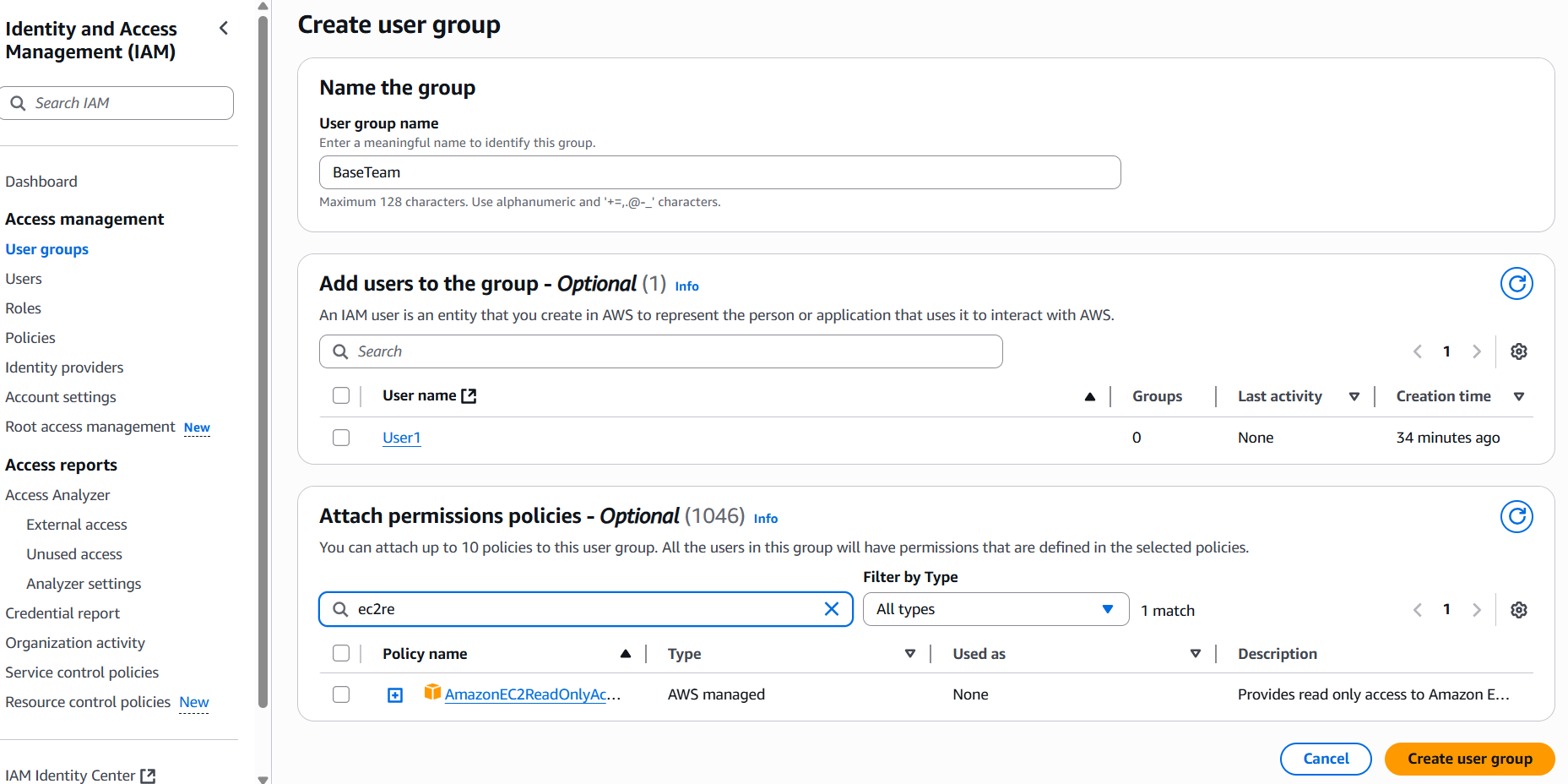
* Review the details and click Create user.



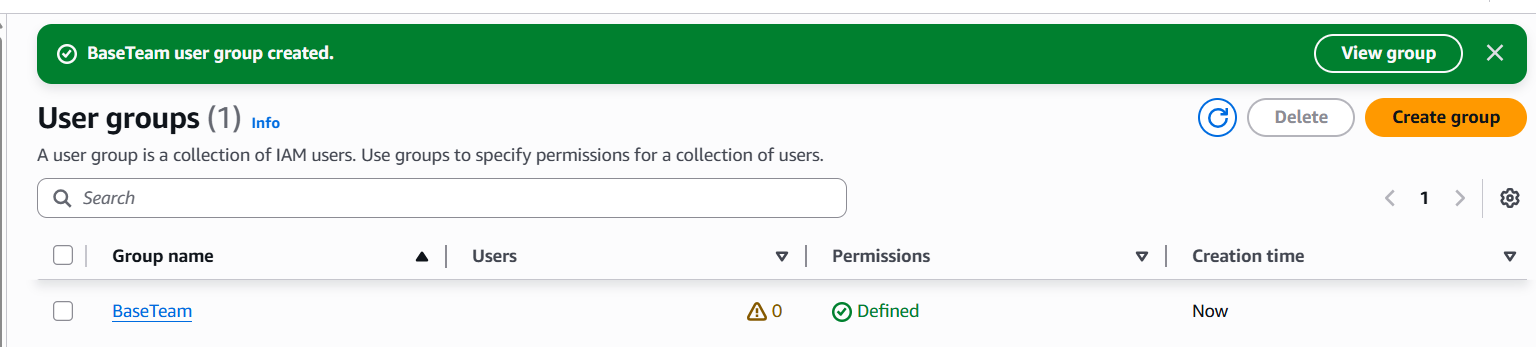
2) Create one Group in IAM and Assign Read access for ec2.

* In the IAM dashboard, click User Groups. Click the Create group button.
* Give the group a name.
* In the "Attach permissions policies" section, search for AmazonEC2ReadOnlyAccess.
* Select the checkbox next to the policy with this name.



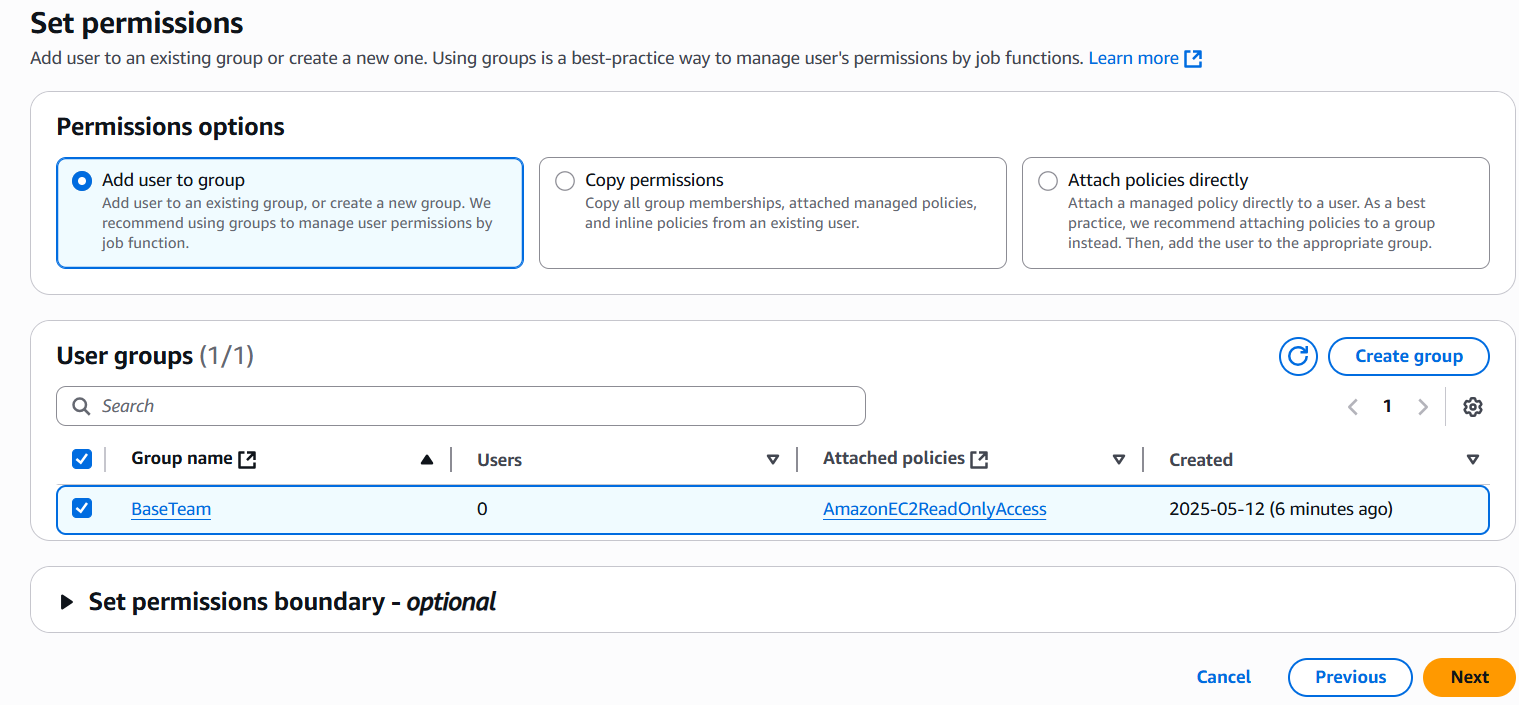


* Click Create group.

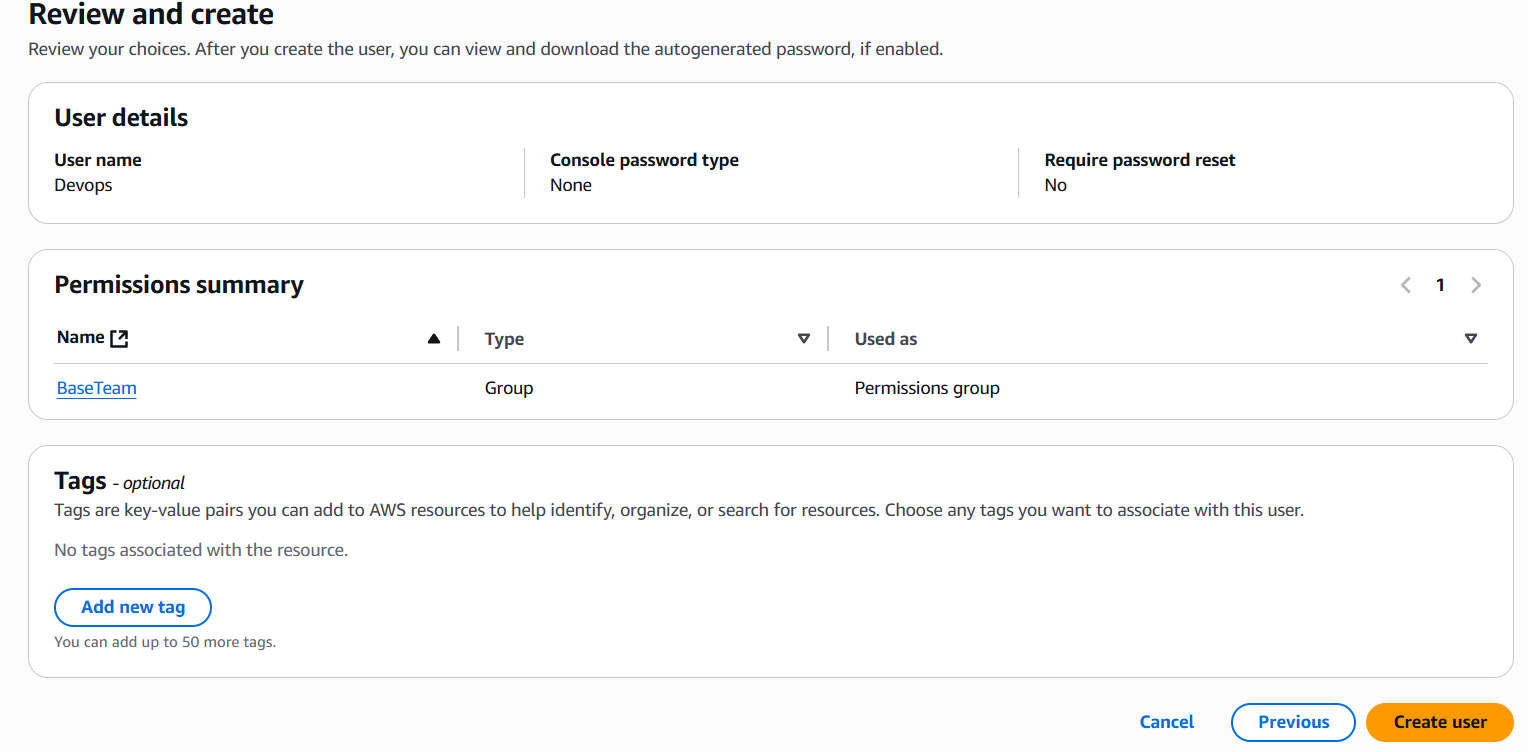


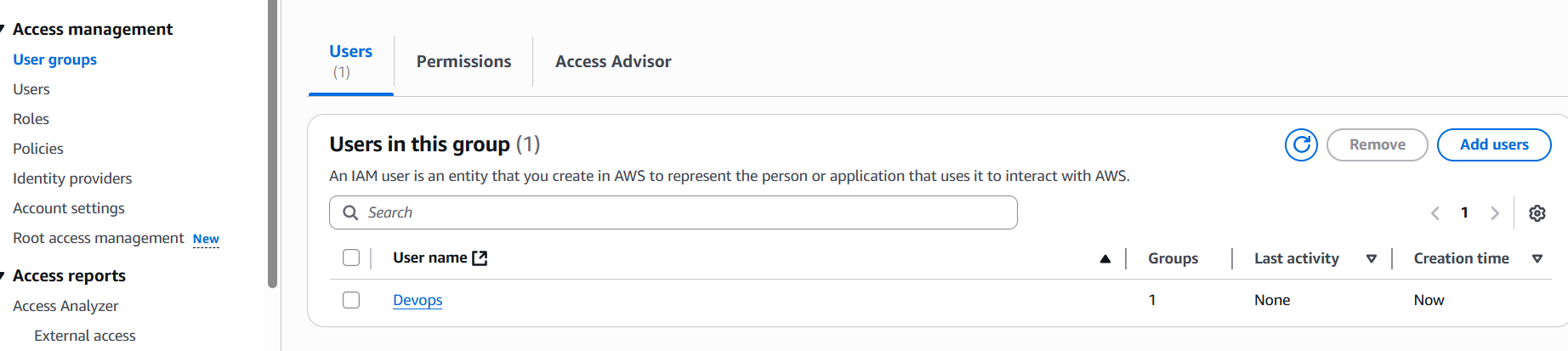
3) Create a new user with name Devops and add to the group created in task2.

* Create a user like we did in task 1, name that user Devops
* On the "Set permissions" page, select Add user to group.
* Find and select the checkbox next to the EC2ReadOnly group we created in Task 2.
* Click Next.



* Review the details and click Create user.





4) Write a bash script to create a IAM user with VPC full access.

* Install and configure AWS CLI .
* If not already done then you need to configure the AWS CLI with credentials (Access Key ID and Secret Access Key) that have permission to create IAM users and attach policies.
* Create a script to create a IAM user with VPC full access, save and quit.

Script Details-

* **NEW\_USER\_NAME** & **POLICY\_ARN** are the variables which store our given user name and the ARN (Amazon Resource Name) for the AWS managed vpcfullaccess policy.
* **aws iam create-user --user-name** **"$NEW\_USER\_NAME"**-

'aws' : This is the command to invoke the AWS CLI tool.

'iam' : This specifies that we want to interact with the IAM service.

'create-user': This is the specific action we want to perform within IAM.

'--user-name': This is an option (or parameter) for the 'create-user' action. It tells AWS what name to give the new user.

"$NEW\_USER\_NAME": This is the value for the '--user-name' option.

* **aws iam attach-user-policy --user-name "$NEW\_USER\_NAME" --policy-arn "$POLICY\_ARN"**

'attach-user-policy': The specific action to attach a policy to a user.

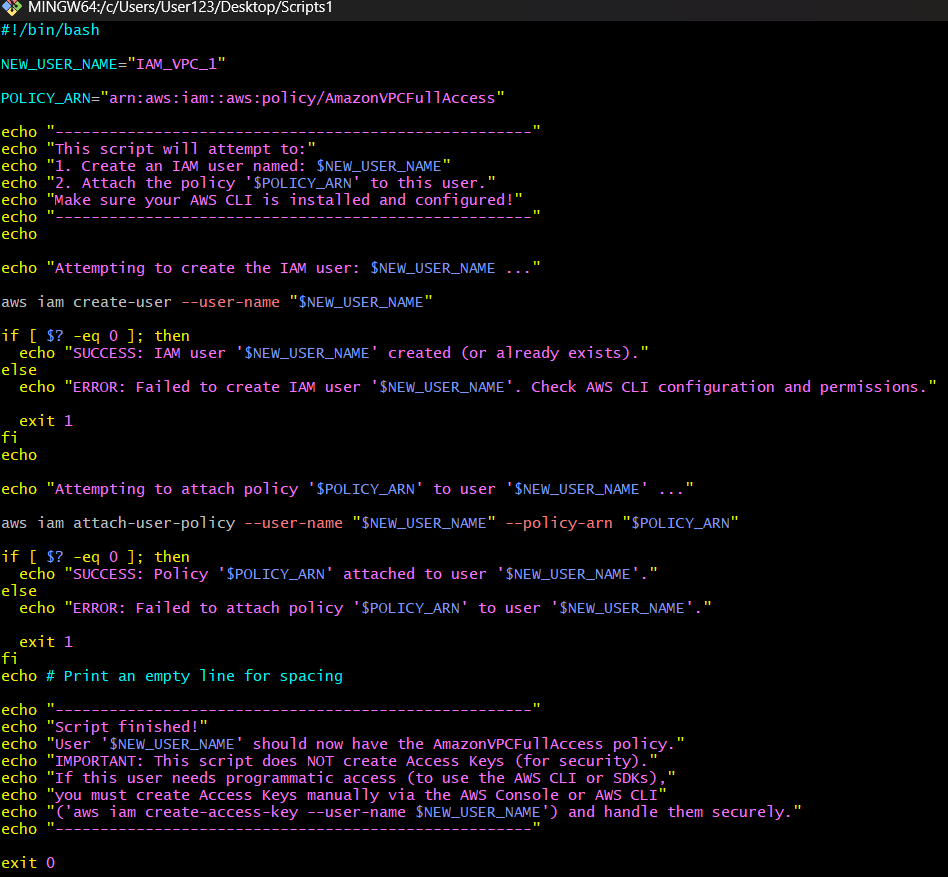
'--user-name' : Specifies which user to attach the policy to.

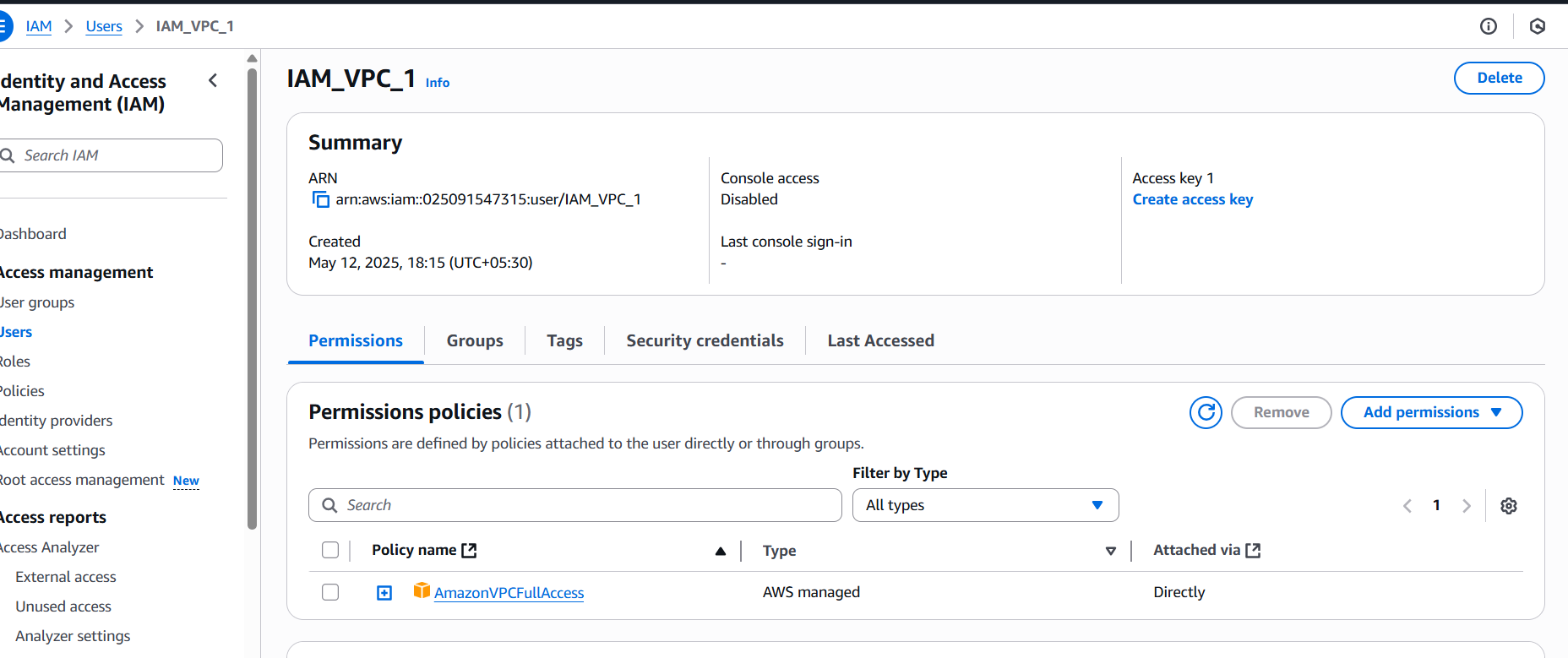
"$NEW\_USER\_NAME" : The name of the user we just created.

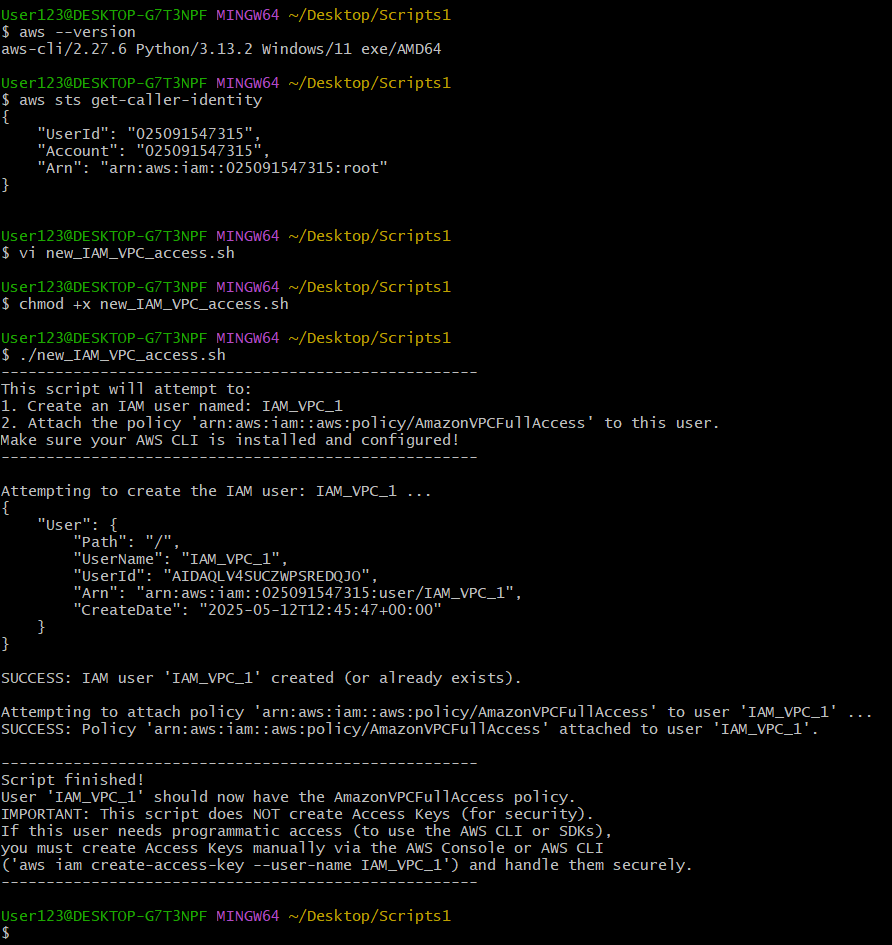
'--policy-arn' : Specifies which policy to attach. Policies are identified by their ARN.

"$POLICY\_ARN" : The ARN of the AmazonVPCFullAccess policy we defined earlier.

* **if [ $? -eq 0 ]; then ... else ... fi** : This is a basic way to check if the previous command was successful. $? holds the exit code (0 for success, non-zero for failure). -eq means "equal to". This block lets the script react differently based on whether a command worked or not.







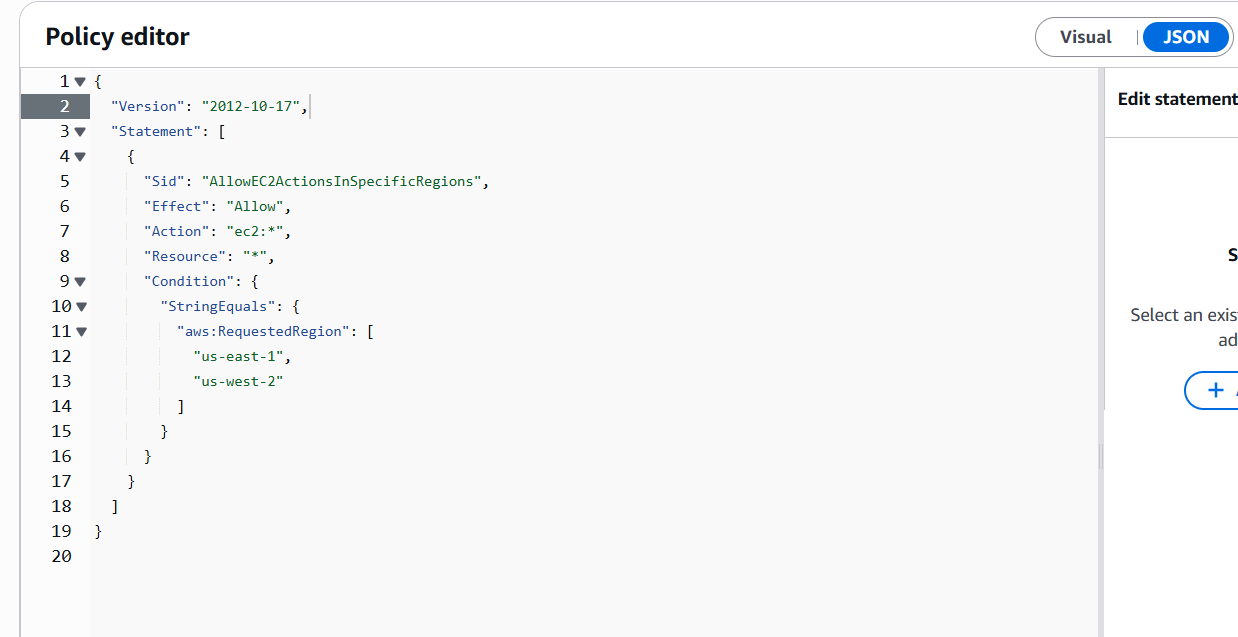
* Make the script executable: chmod +x <ScriptName>.sh
* Run the script: ./<ScriptName>.sh

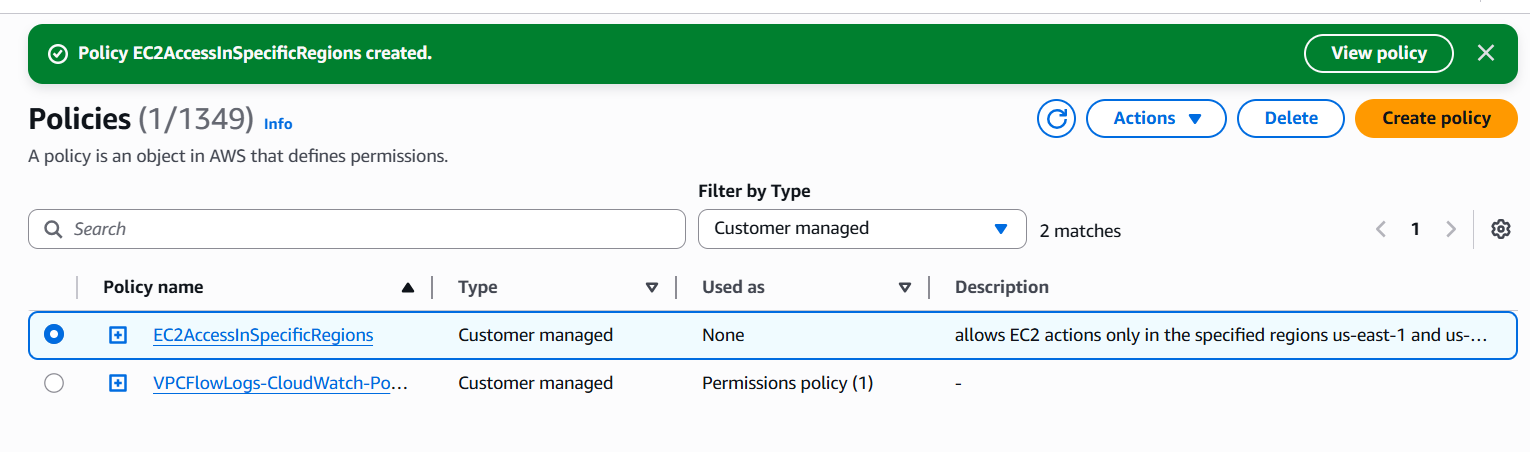
5) Create a IAM policy to access ec2 for a specific user in specific regions only.

IAM policies are JSON documents that define permissions within AWS.

The basic elements of an IAM policy include:

* **Version** : which specifies the version of the policy language (the most current is 2012-10-17)
* **Statement** : This label holds the main part of the policy – a list of individual permission rules (called statements)
* **Sid** : Stands for "Statement ID". This is like a nickname or comment for the statement, making it easier for humans to understand what this rule does. It's optional but highly recommended.
* **Effect** : says whether this rule allows or denies the permissions listed.
* **Action** : which lists the specific AWS API(rules) actions that the policy allows or denies
* **Resource** : This defines which AWS resources the Action applies to. Resources are the things the user can act upon (like specific EC2 instances, S3 buckets, etc.).
* **\*** : The asterisk \* here means "all resources"
* **Condition** : This block adds specific criteria that must be met for the Allow effect in this statement to apply.
* **StringEquals** : This is a type of condition operator. It checks if two text strings are exactly the same.
* **aws:RequestedRegion:** is a global condition key used in IAM policies to control which AWS regions an IAM principal (user or role) can access.
* We can use these elements to create the necessary policy for our need
* Go to policies from the IAM dashboard, click on Create policy.
* Select JSON tab and enter the policy there, click Next
* Give it a Name and description (optional)





* This policy allows EC2 actions only in the specified regions (us-east-1 and us-west-2).

6) We have two accounts Account A and Account B, Account A user should access s3 bucket in Account B.

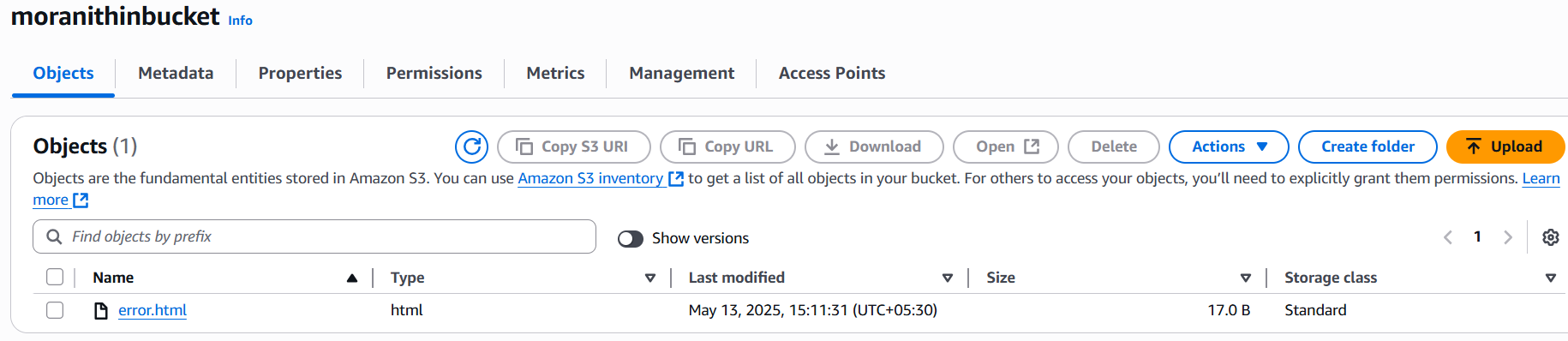
(Collaborate with team member and execute this.Mostly asked in every interview)

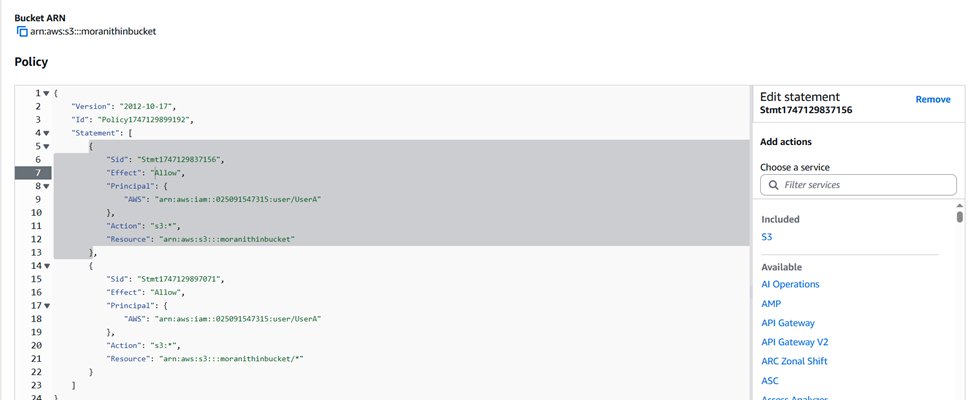
*Cross-account access scenario (S3 Bucket access)*

**Bucket Policy Method:**

ACCOUNT B-

* Log in to the Account B AWS Management Console.
* Go to the S3 service.
* Click on the name of the target bucket (moranithinbucket).
* Go to the Permissions tab.
* Scroll down to the Bucket policy section and click Edit.
* Create a Policy depending on what permission or to what extent Account B wants to grant Account A’s user access to their s3 Bucket.
* Click Save changes.





ACCOUNT A-

* Ensure your AWS CLI is configured in Account A with credentials for the specific user.
* Open your terminal. You can now access the bucket in Account B using AWS CLI commands, specifying the bucket name and your Account A profile
* **aws s3 ls s3://moranithinbucket --profile s3bucket** - List Objects in Account B’s bucket.

