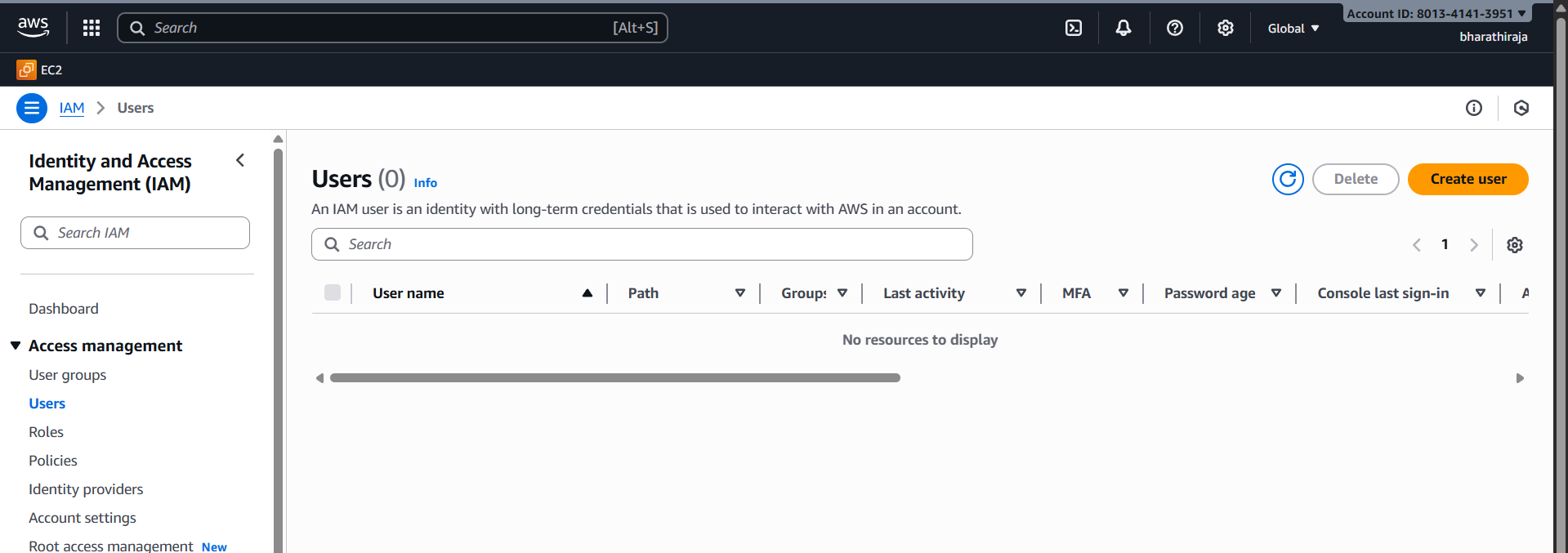
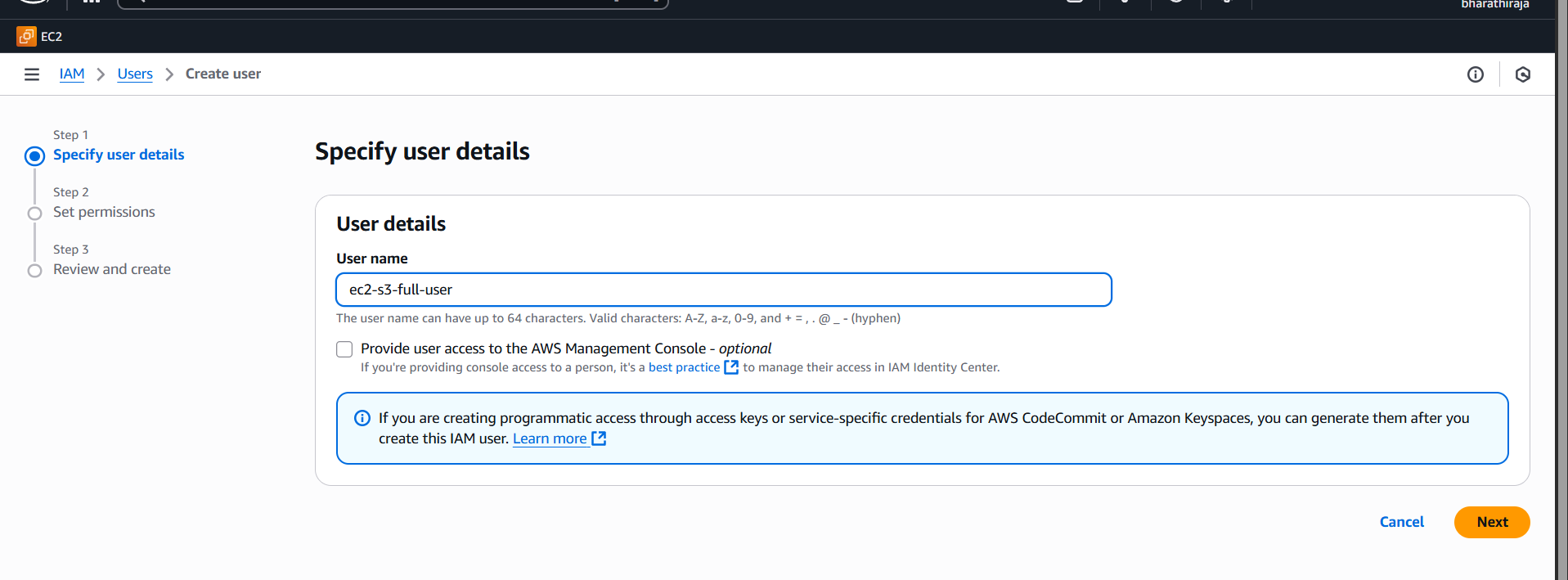
1. Create one IAM user and assign EC2 and S3 full access roles.

Go to the AWS console => go to the IAM part, click on user



Give a name as you wish, or else relate to your project

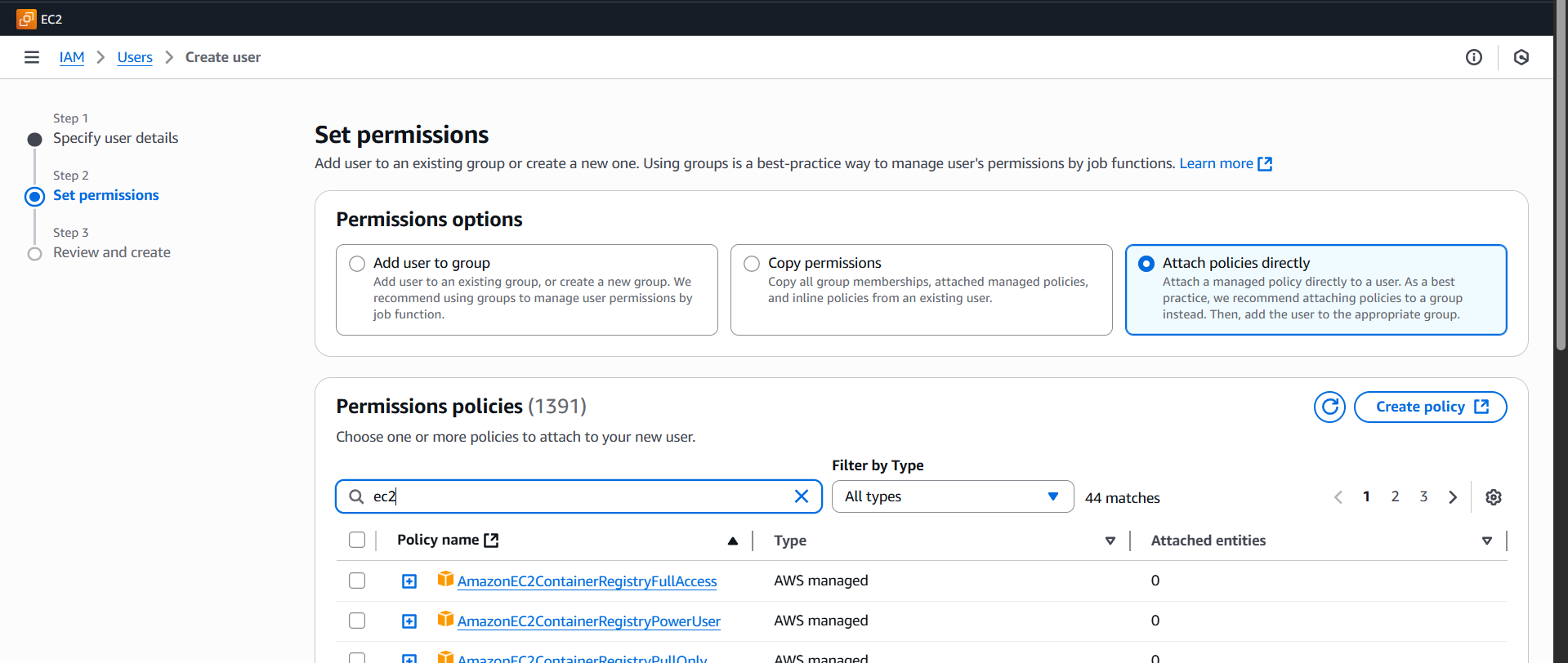


Check **Console access** if the user needs to sign in to the AWS Console (you’ll set a password)

I did not choose the password permission

Click next: permissions.

Choose one of these (recommended: use a group or a quick way to attach managed policy directly



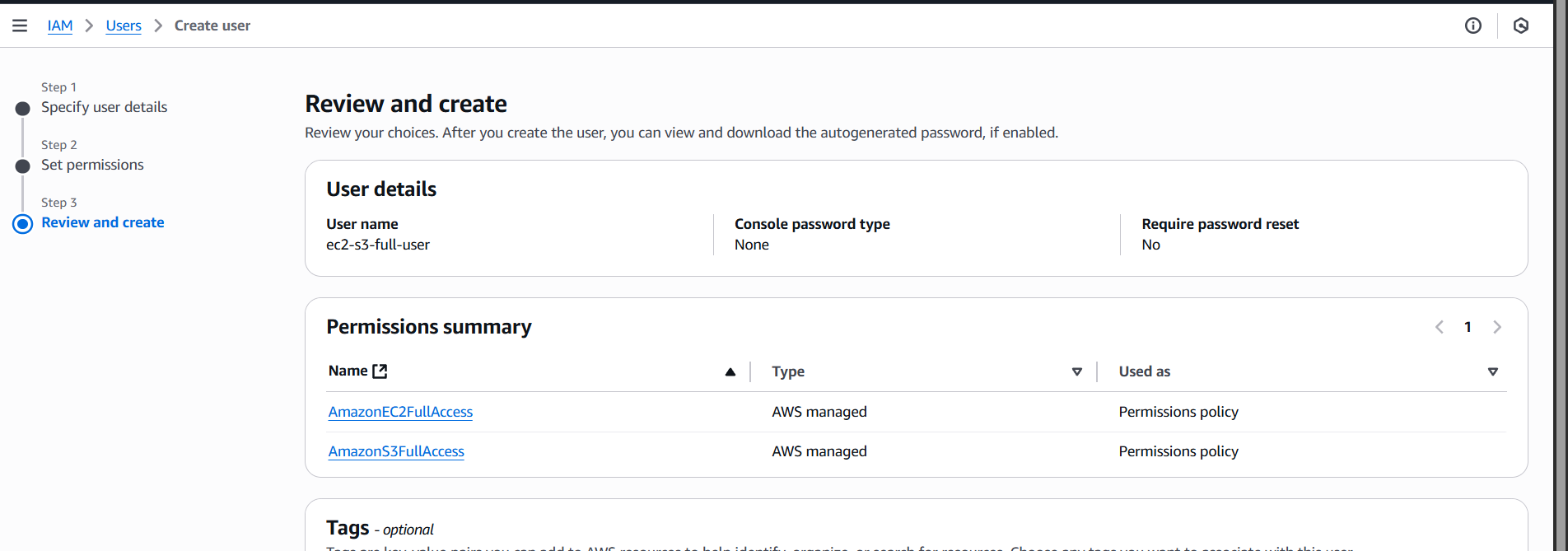
Choice access control of ec2

Ec2 full access



Then search and **attach these AWS managed policies**:

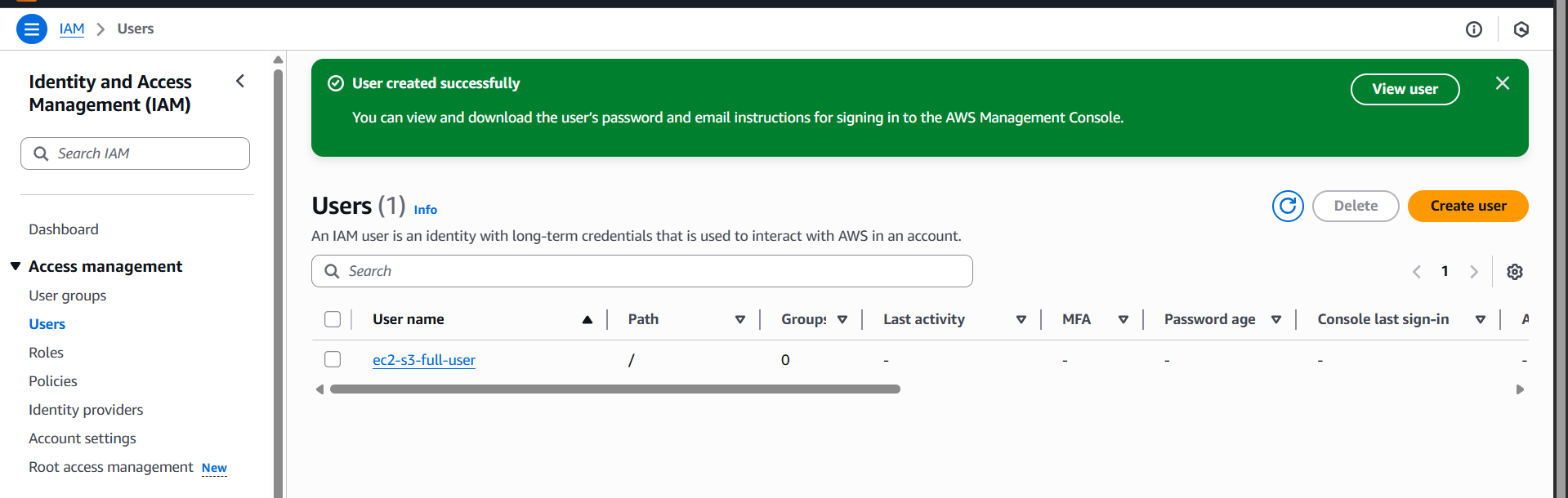
* **AmazonEC2FullAccess**
* **AmazonS3FullAccess**



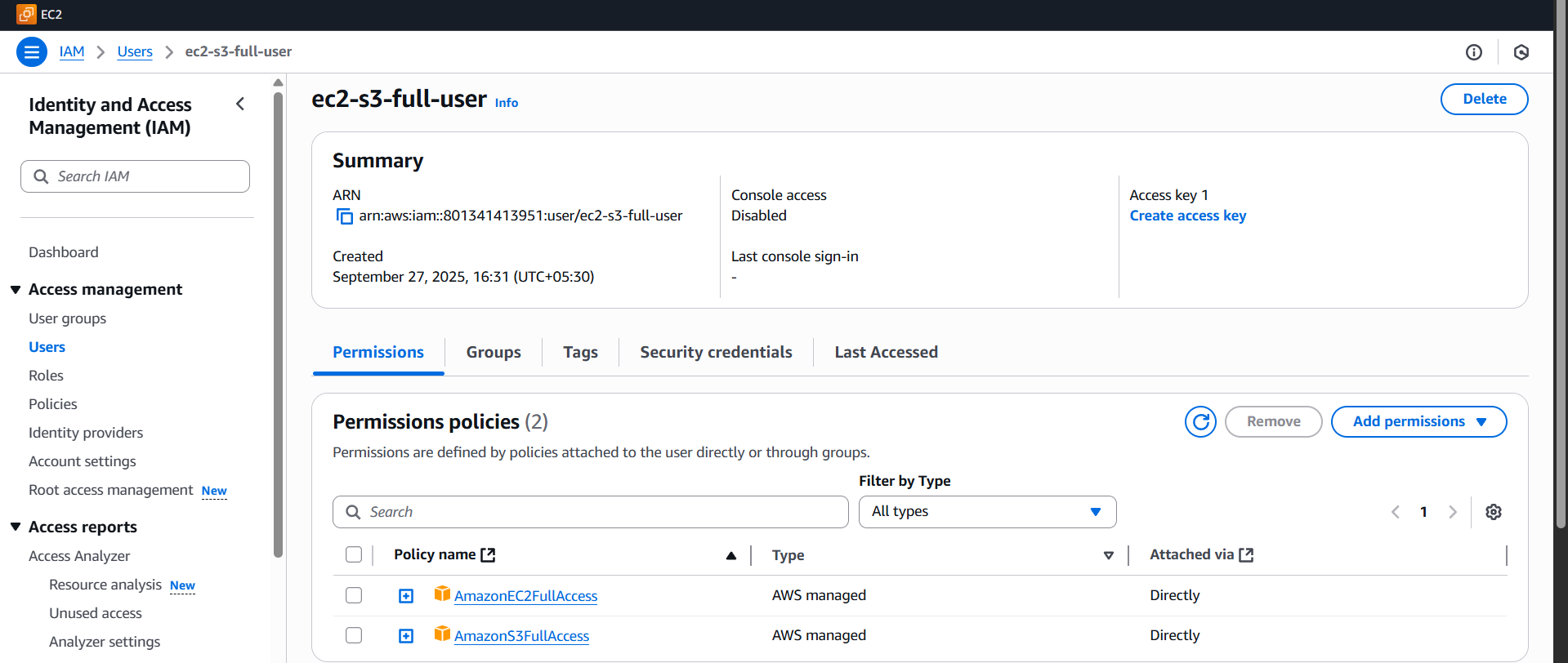
Then, in the bottom right corner, click on the successfully created one user

Click **Next: Tags** (optionally add tags) → **Next: Review** → **Create user**.

On success, you’ll see the user summary. If you enabled programmatic access,



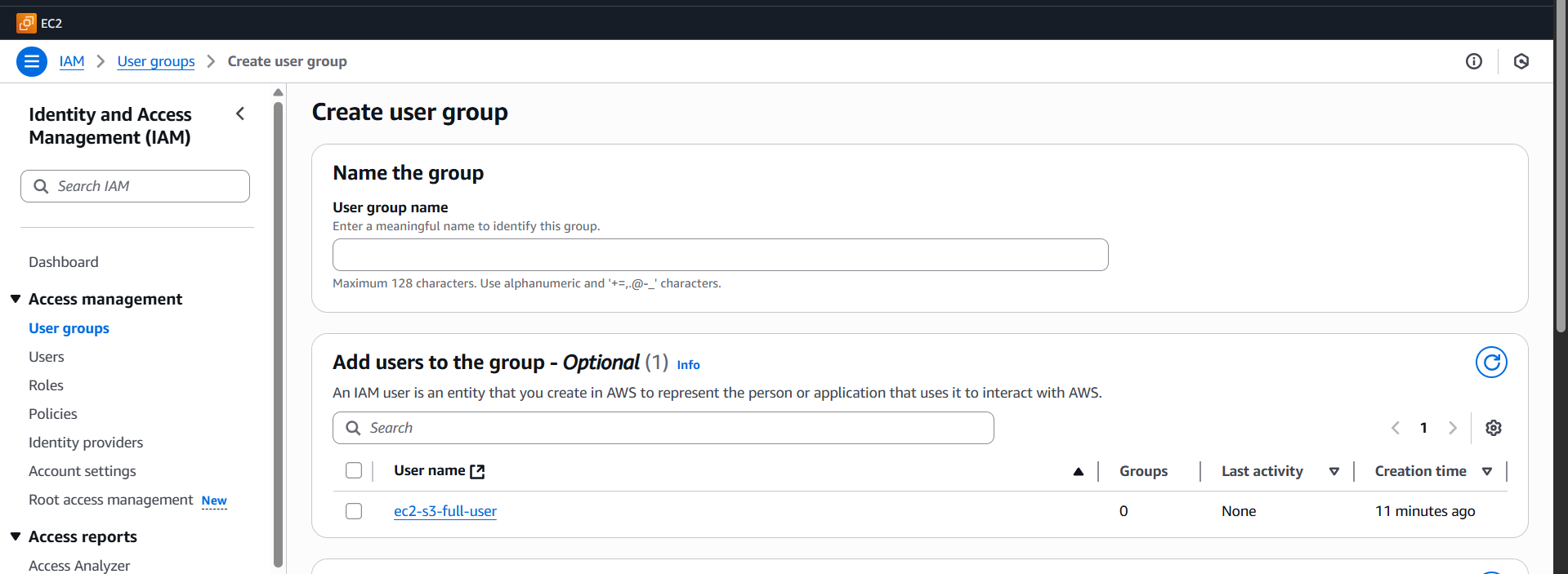
download or copy the **Access key ID** and **Secret access key** now (secret is shown only once). If console access was enabled, you can copy the sign-in link and password or require password reset.



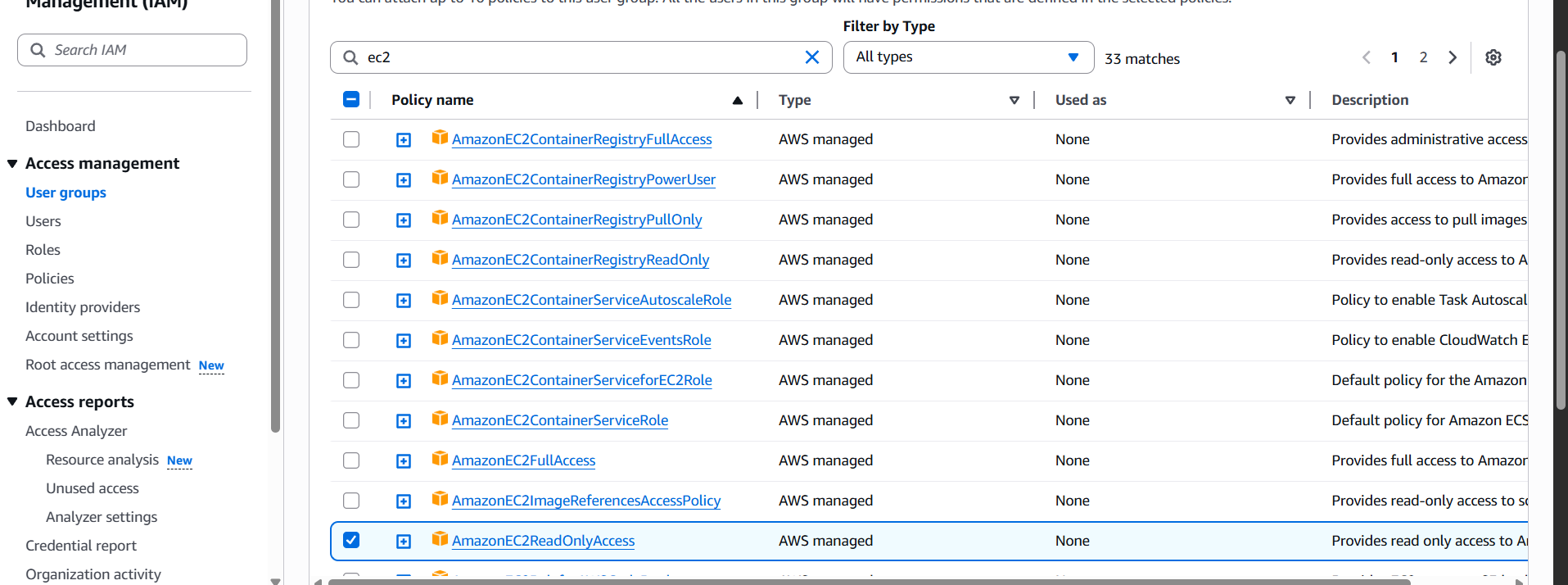
1. Create one group in IAM and assign read access for EC2.

In the left navigation, click **User groups** → **Create group**.

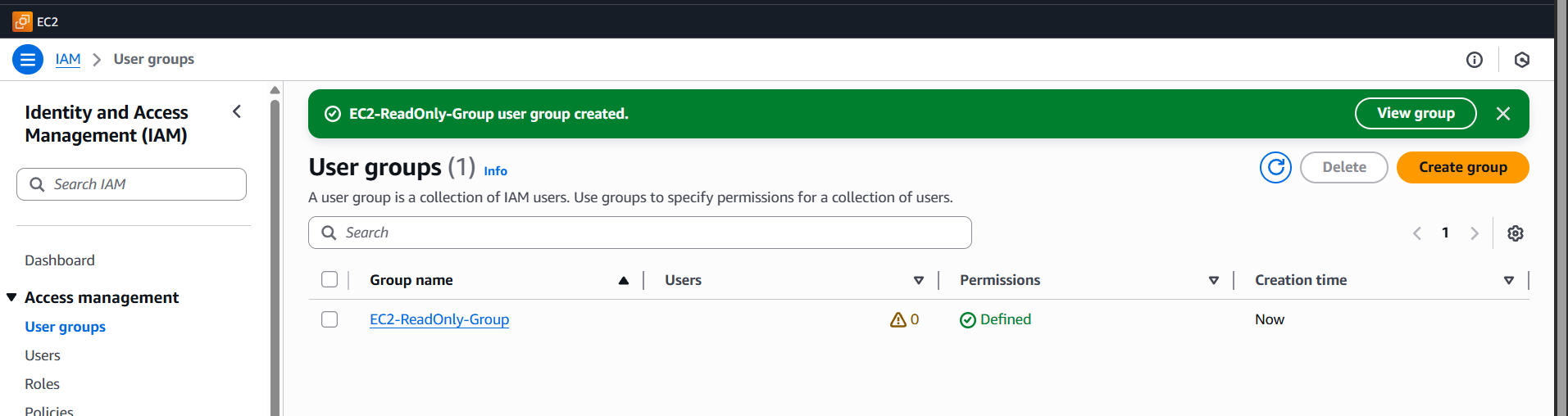
Enter a **Group name** (e.g. EC2-ReadOnly-Group)



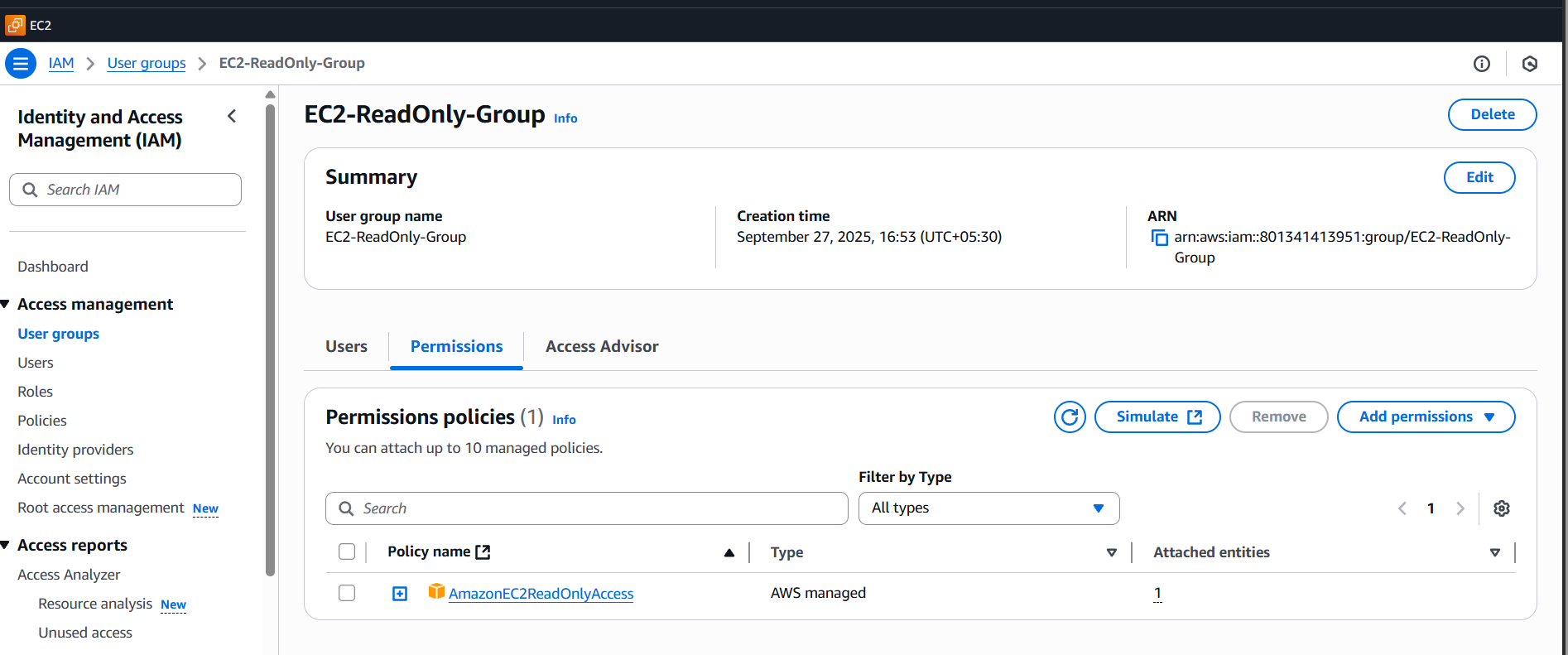
Under **Attach policies**, search for AmazonEC2ReadOnlyAccess.



Click **Create group**.

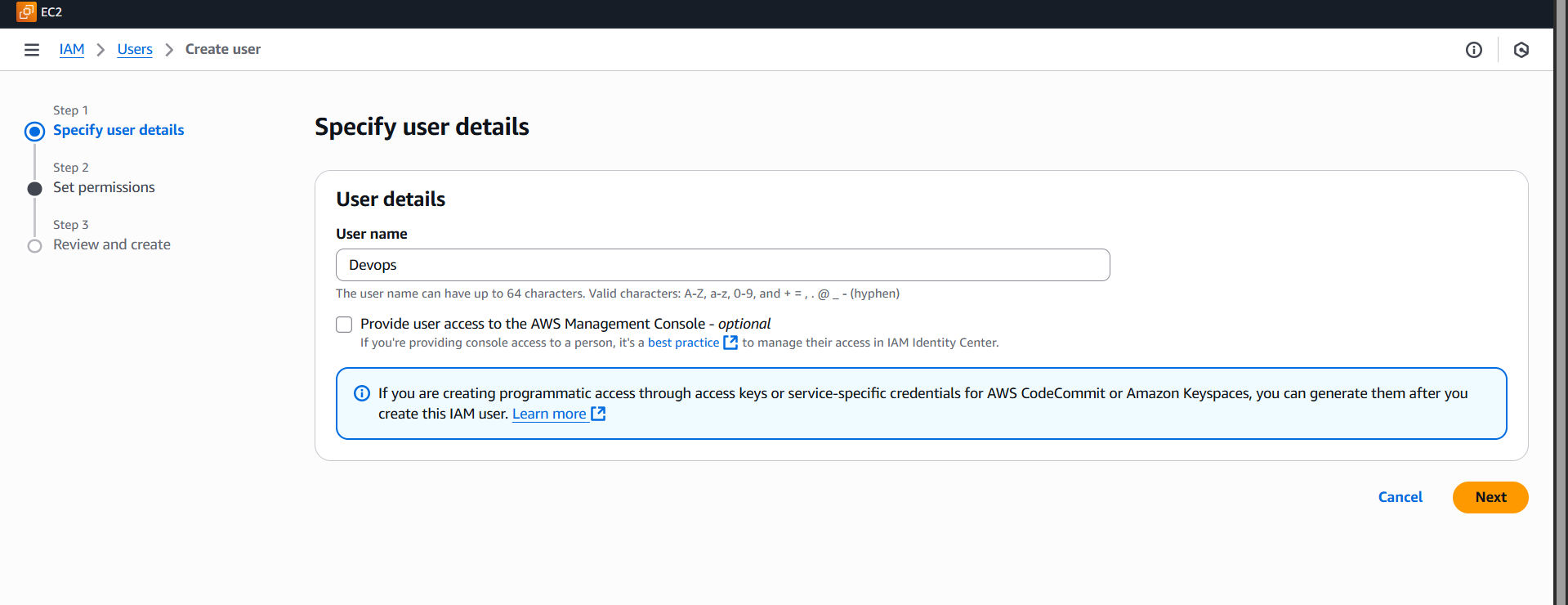


View the group, see what is there

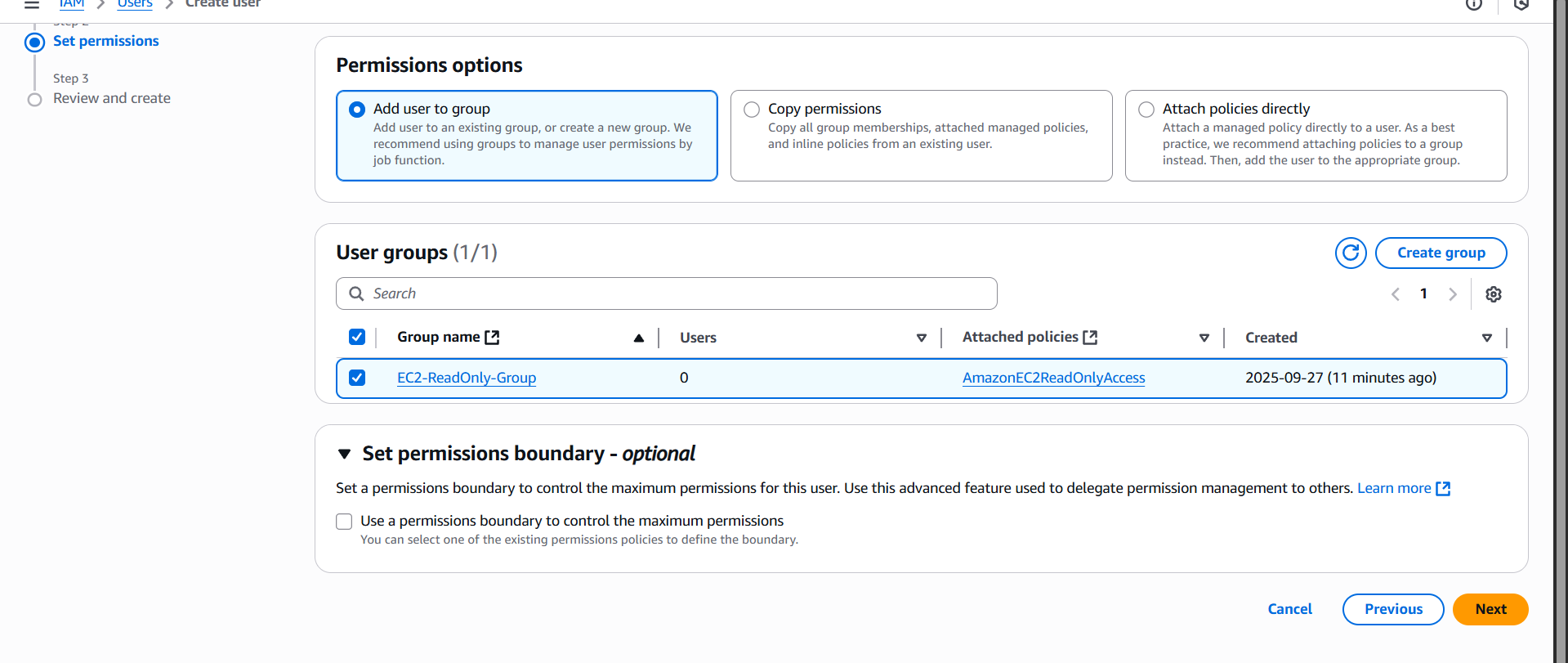


1. Create a new user named "Devops" and add to the group created in task 2.

Go to user click create user, give name devops

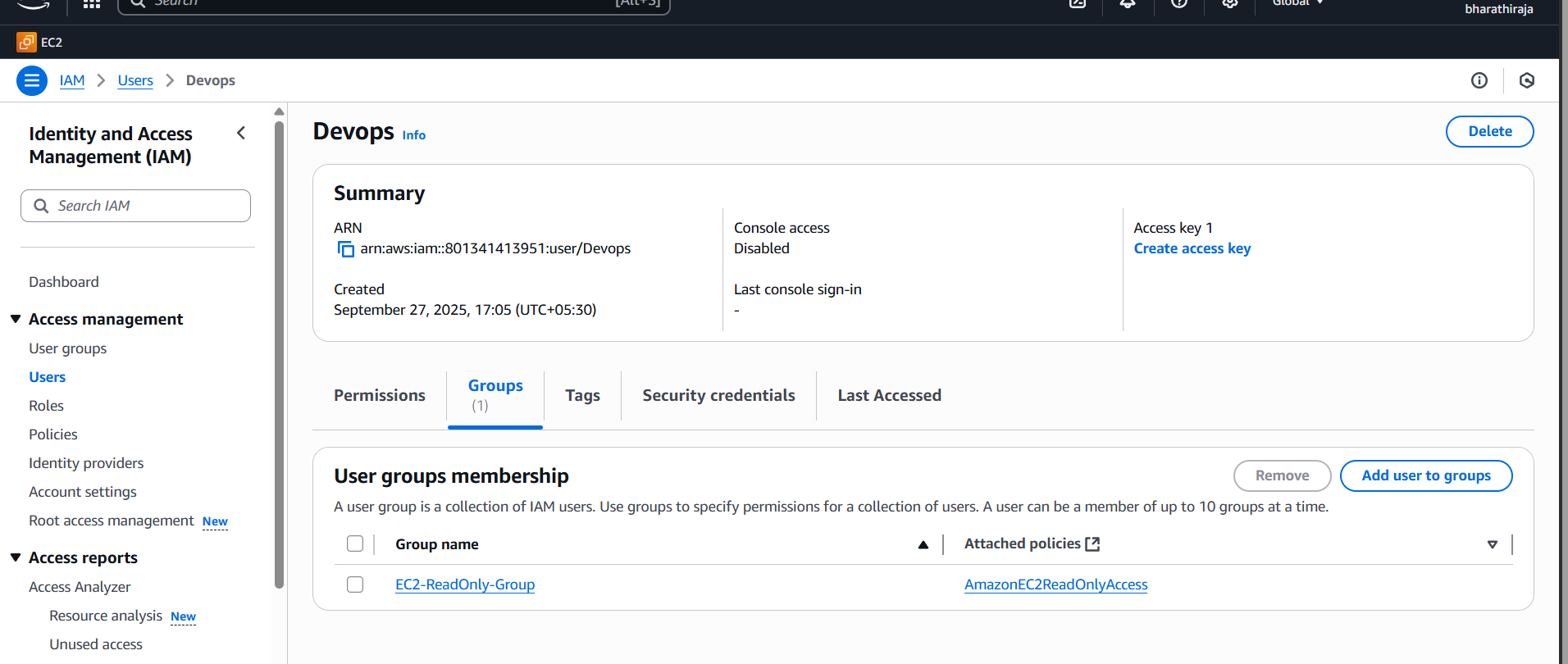


We add the user to the group and add the EC2 read-only group



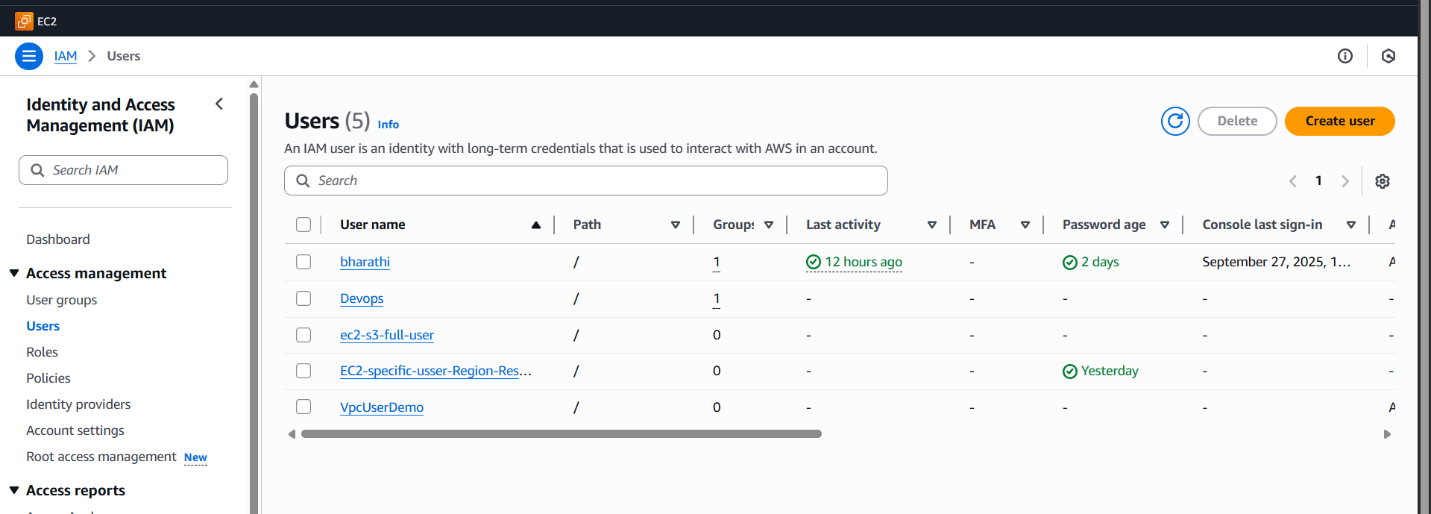
Review settings => click create user

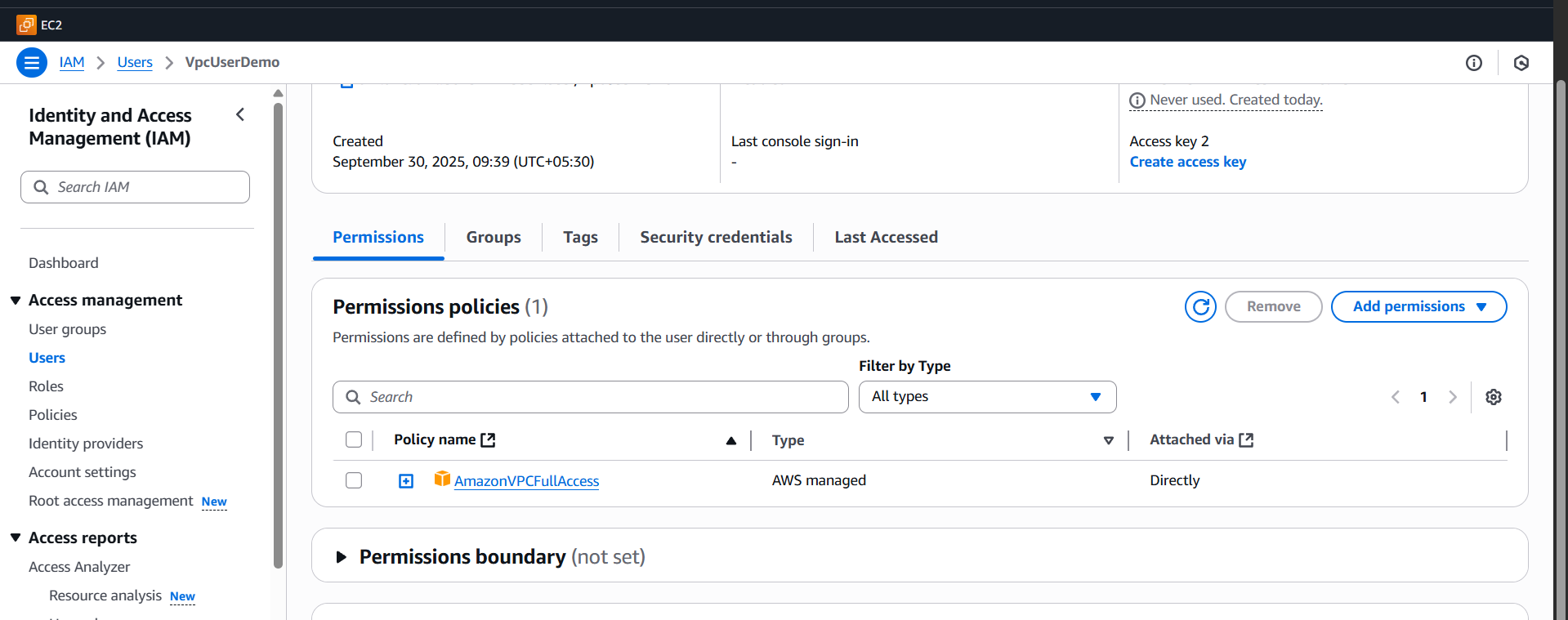
The devops user is now in the EC2-read-only group.



1. Write a bash script to create an IAM user with VPC full access.



**it has created one vpc user demo**

****

**We need follow this steps**

1. **Install AWS CLI**
2. sudo yum install -y awscli # Amazon Linux / RHEL
3. sudo apt-get install -y awscli # Ubuntu/Debian
4. **Configure AWS CLI with admin credentials** (this user must have IAM & VPC policy management rights):
5. aws configure

Provide:

* + AWS Access Key
  + AWS Secret Key
  + Default region (example: us-east-1)
  + Output format (json)

##!/bin/bash

# ============ VARIABLES ============

IAM\_USER="VpcUserDemo"

POLICY\_ARN="arn:aws:iam::aws:policy/AmazonVPCFullAccess"

# ============ CREATE IAM USER ============

echo "Creating IAM user: $IAM\_USER ..."

aws iam create-user --user-name $IAM\_USER

# ============ CREATE ACCESS KEYS ============

echo "Creating access keys for $IAM\_USER ..."

aws iam create-access-key --user-name $IAM\_USER > ${IAM\_USER}\_creds.json

echo "Access keys saved in ${IAM\_USER}\_creds.json"

# ============ ATTACH POLICY ============

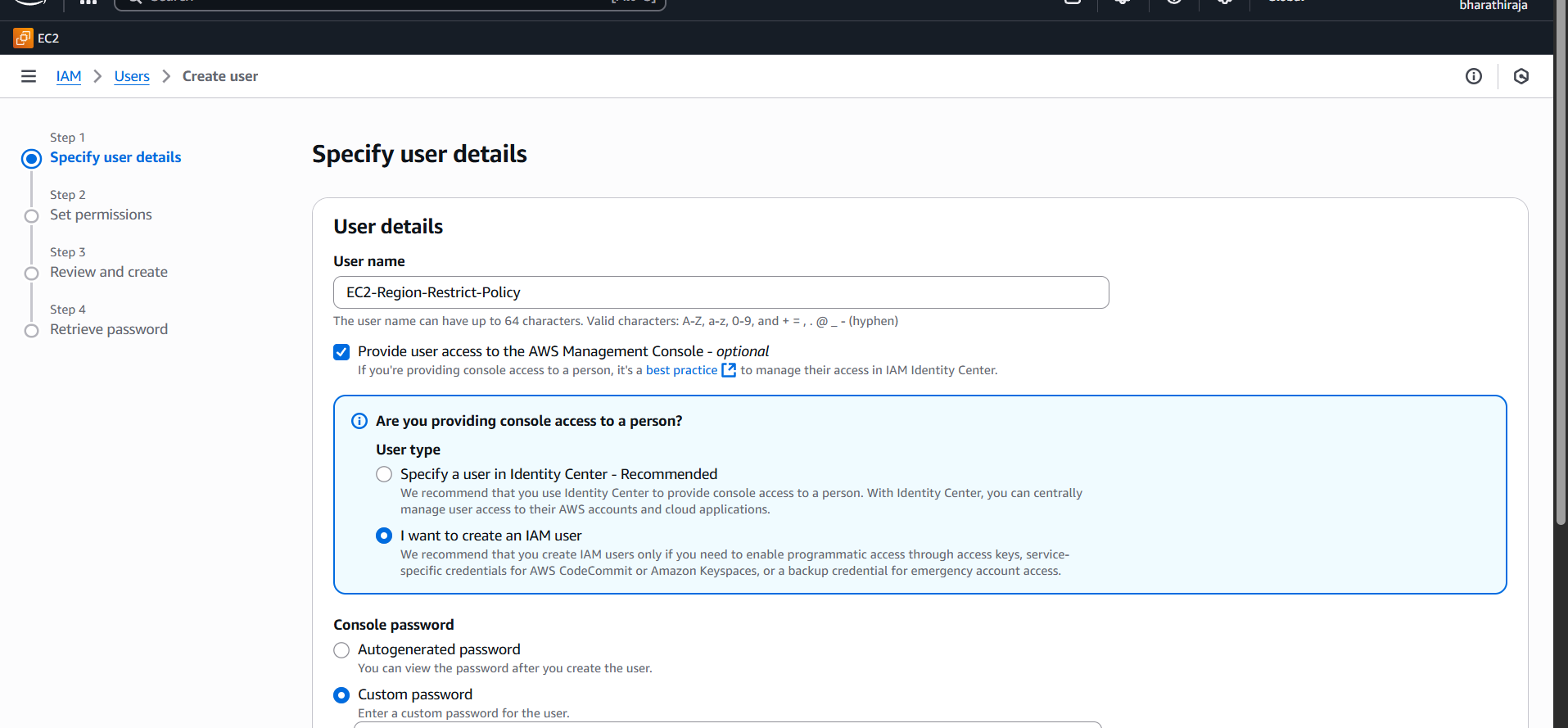
echo "Attaching VPC Full Access policy to $IAM\_USER ..."

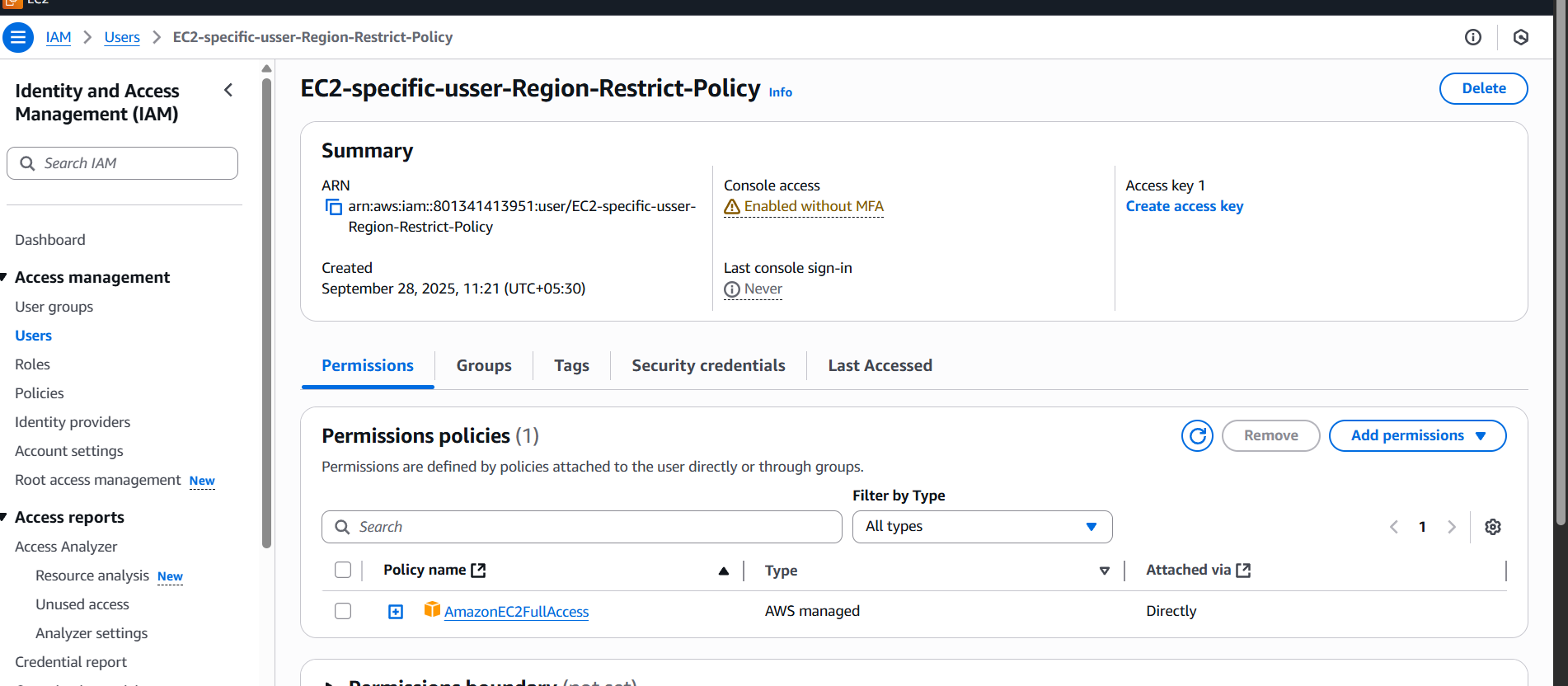
aws iam attach-user-policy --user-name $IAM\_USER --policy-arn $POLICY\_ARN

echo "User $IAM\_USER created successfully with VPC Full Access."

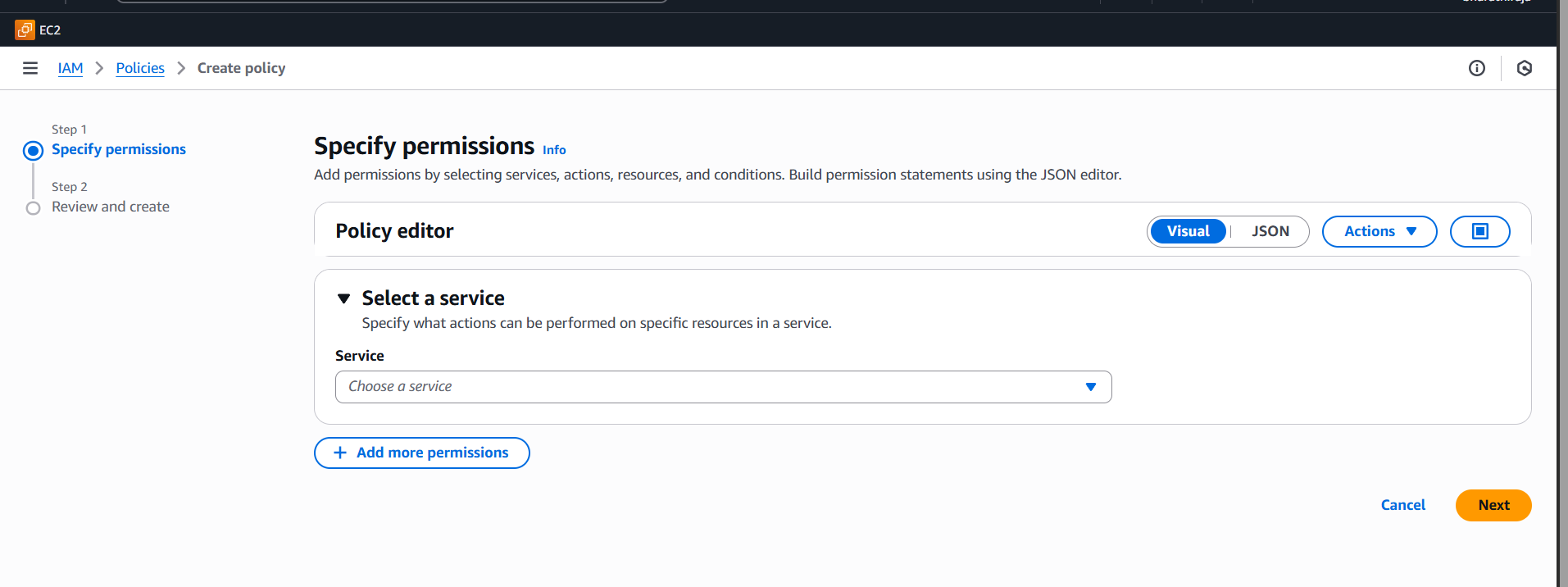
1. Create an IAM policy to allow EC2 access for a specific user in specific regions only.

I created an IAM user name, EC2-Region-Restrict-Policy





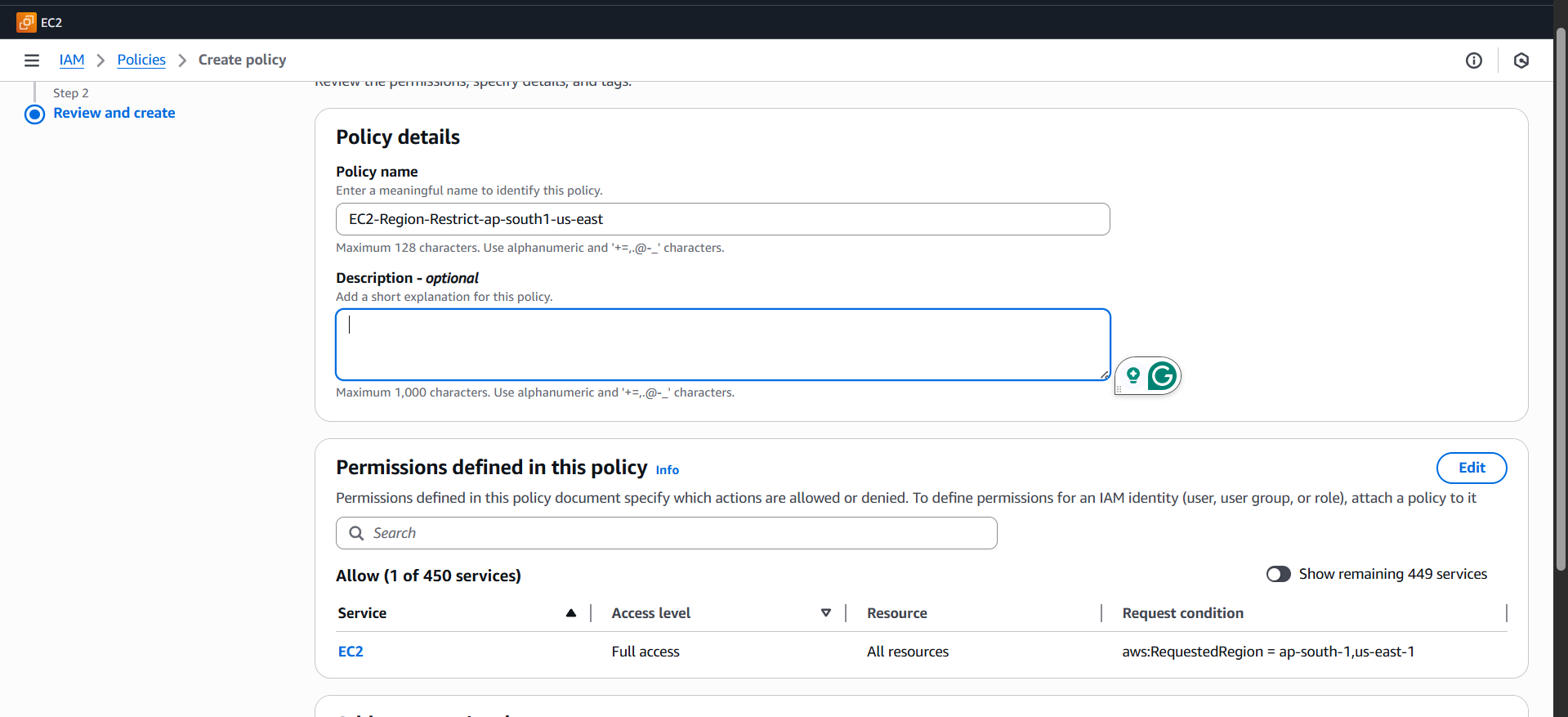
Then I give ec2 full access to a specific user in a specific region only

In aws, there are three ways in which you can create a policy   
using the visual editor   
using json

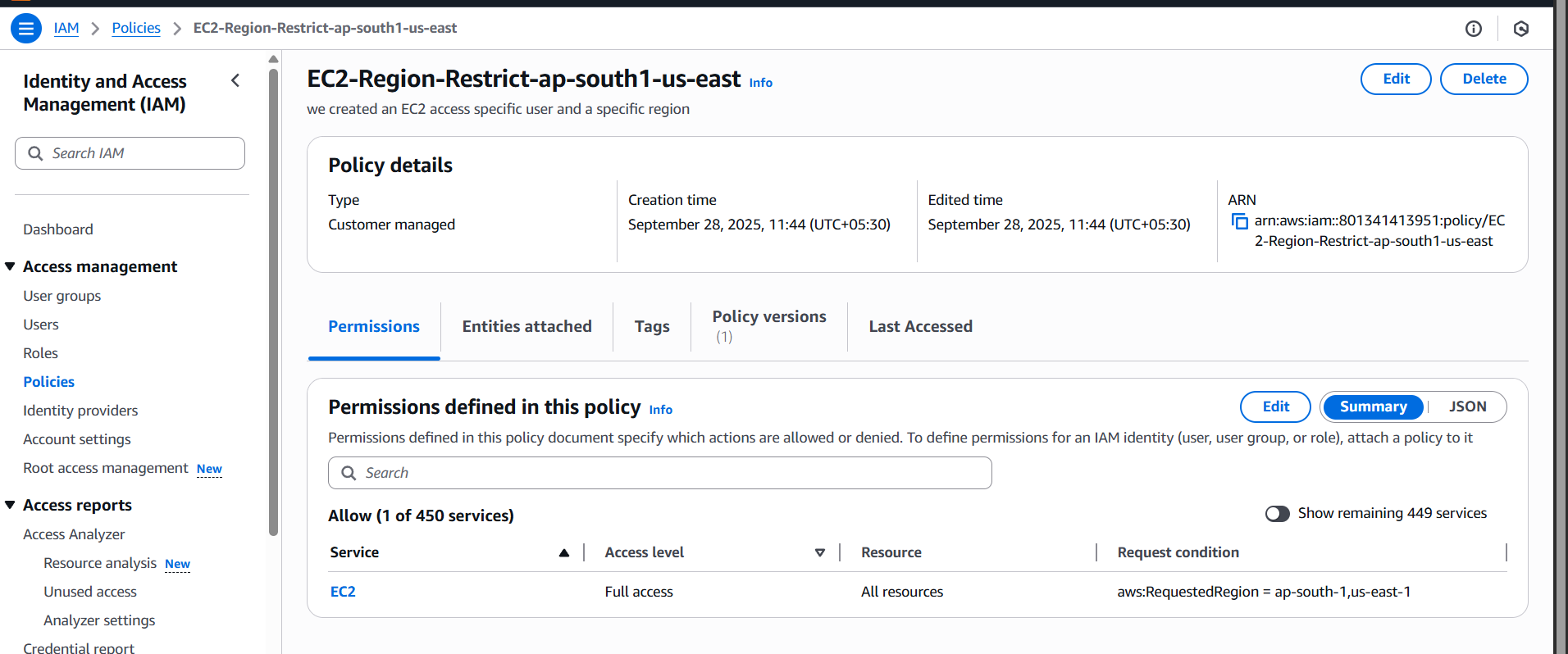
**

As this a custom policy, we will be using JSON to write it from scratch.

1. Select the **JSON** tab to write the policy manually.This policy above will prevent users Launching EC2 instances in any region other than us-east-1.
2. Creating S3 buckets in any region other than us-east-1.
3. Give the Policy a name and click “create policy*”.*

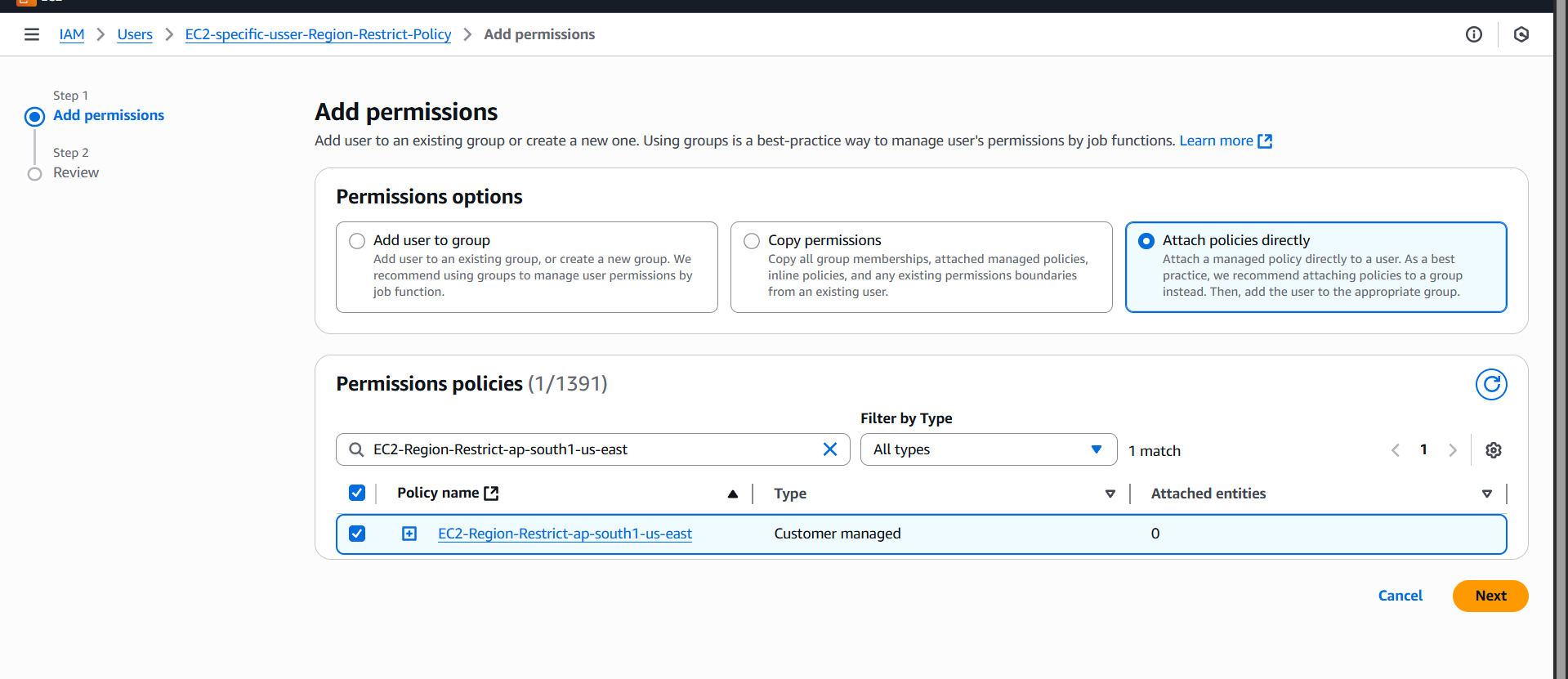
**

*we created*

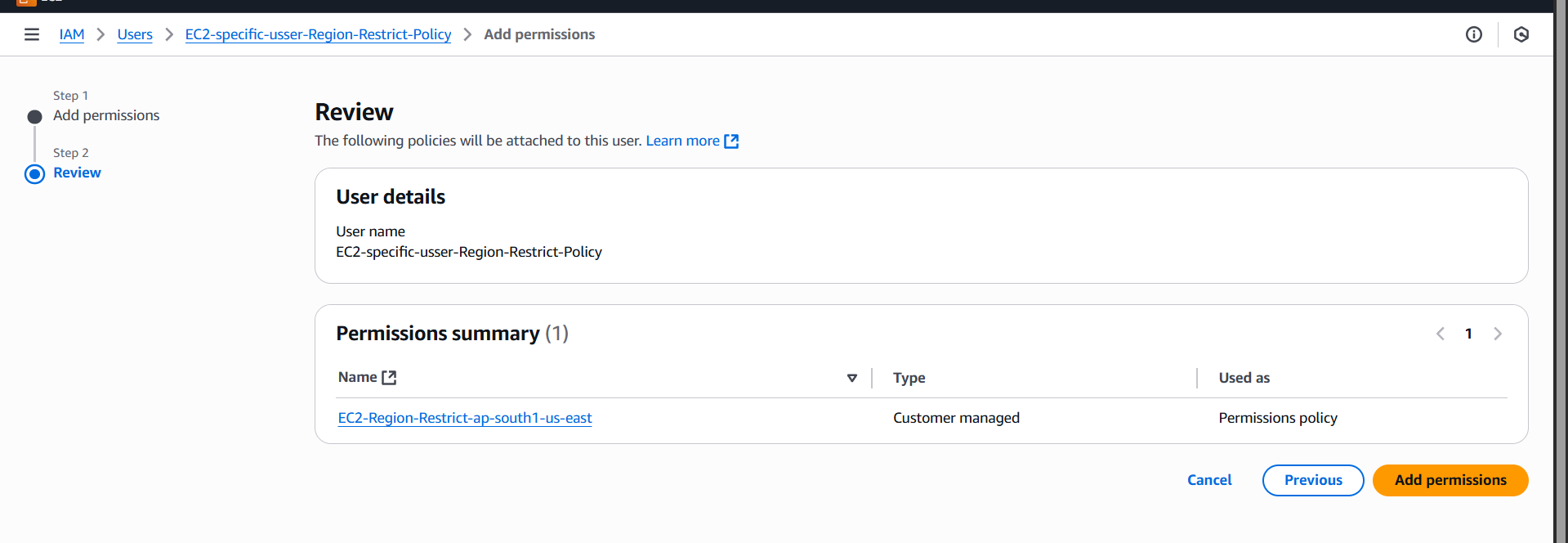


Give a name (e.g. EC2-Region-Restrict-ap-south1-us-east1) and description → **Create policy**.

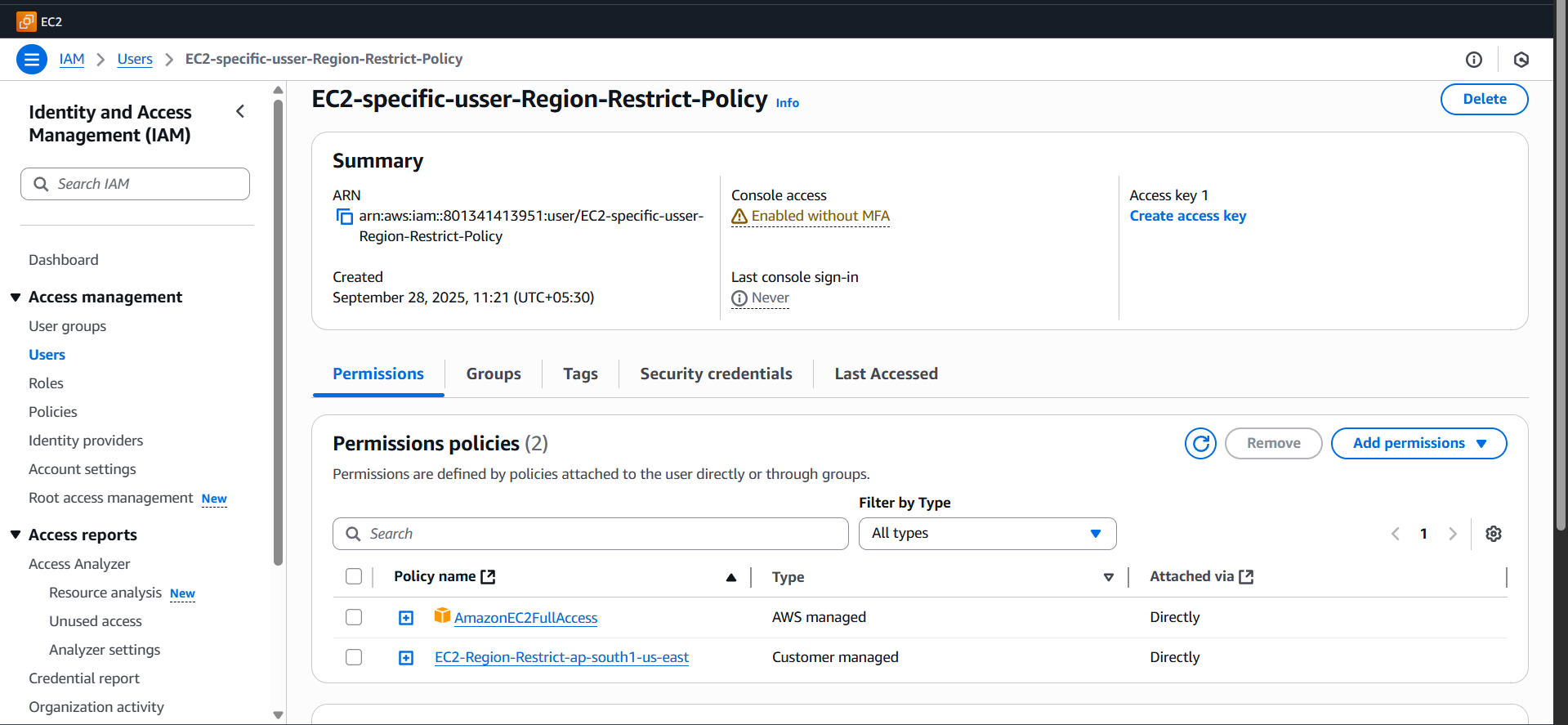
Go to **IAM → Users**, click the target user, **Permissions → Add permissions → Attach existing policies**



Give add

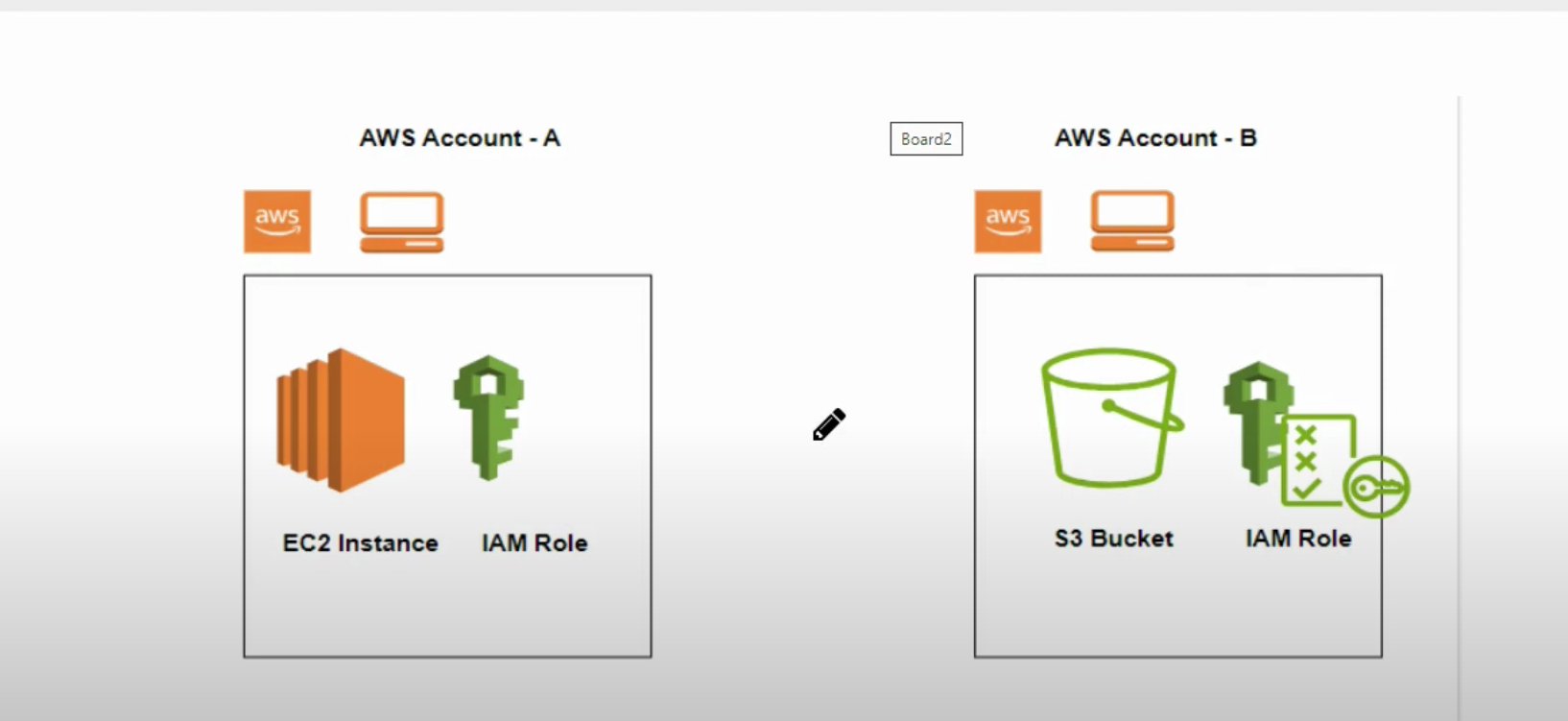


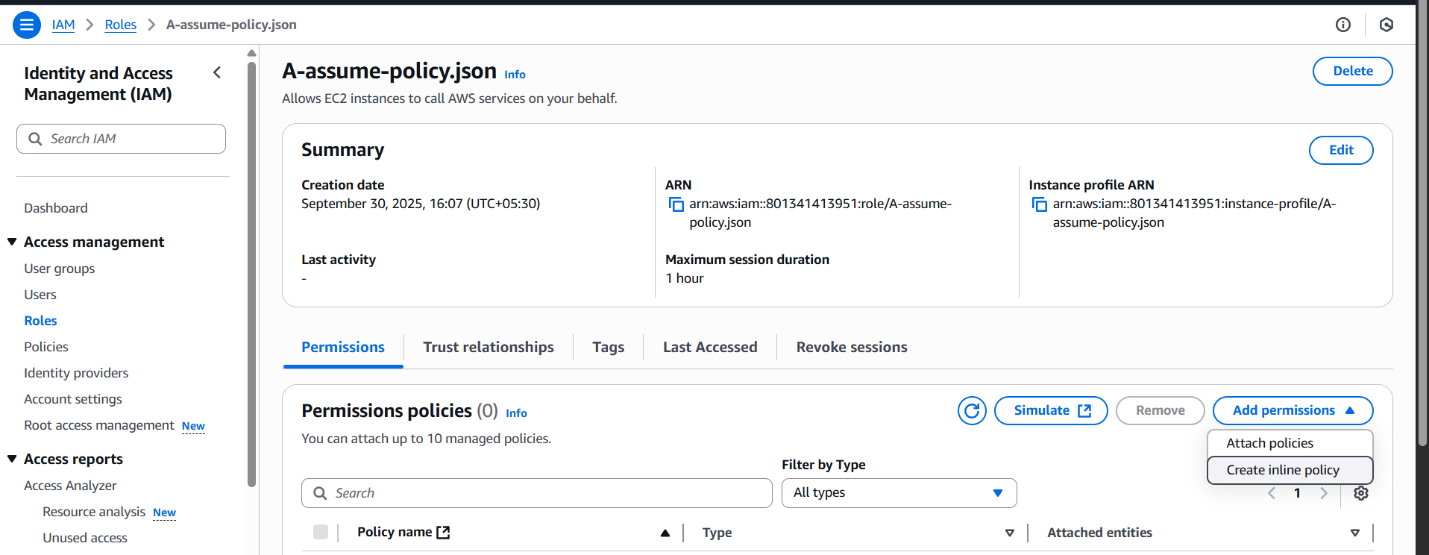
Search for the policy name you created, check it, and **add permissions**. The user now has the policy applied.

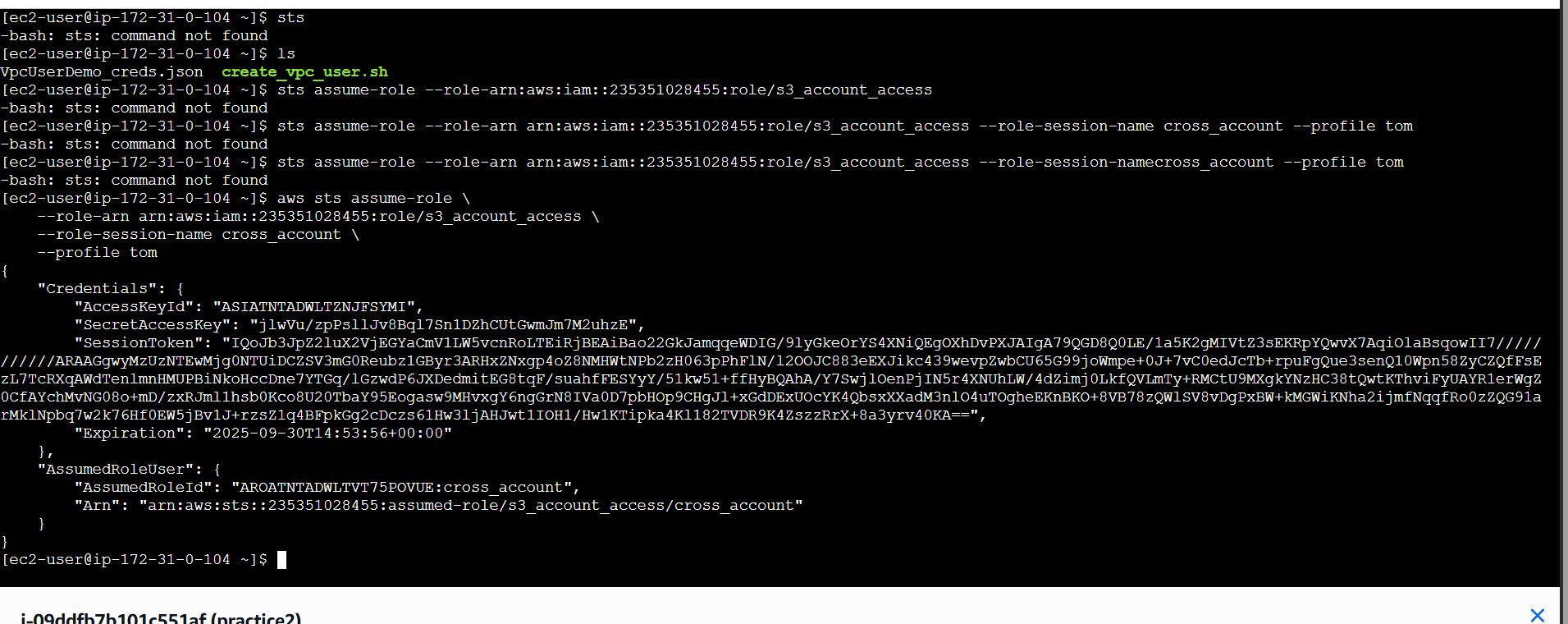
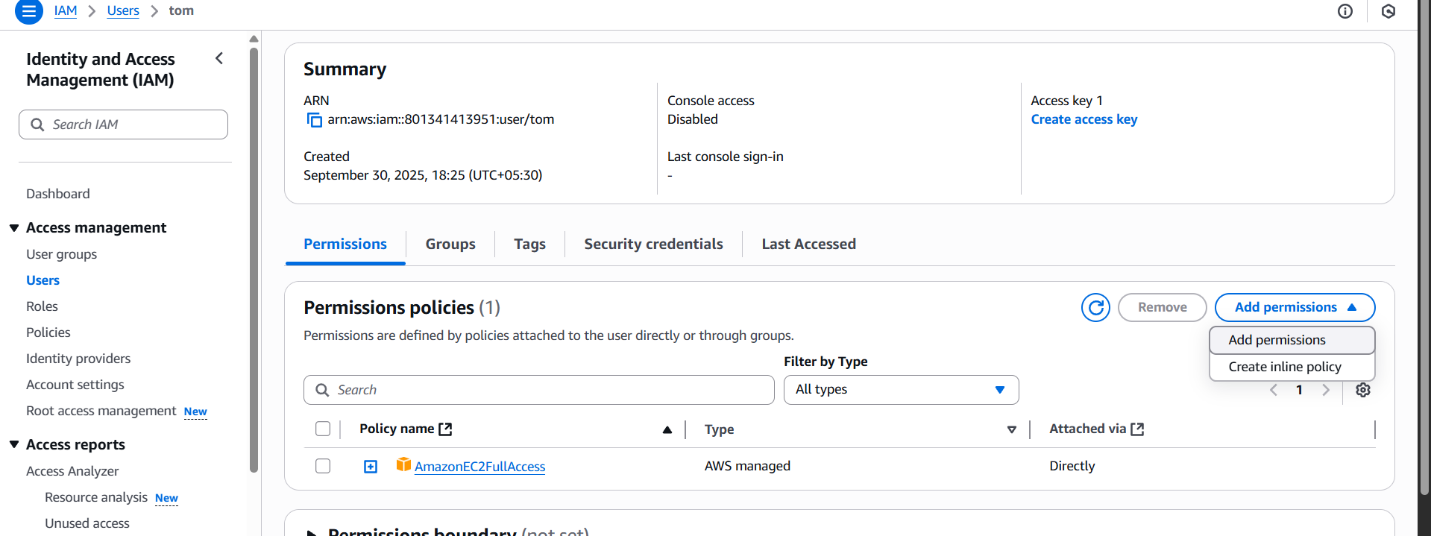
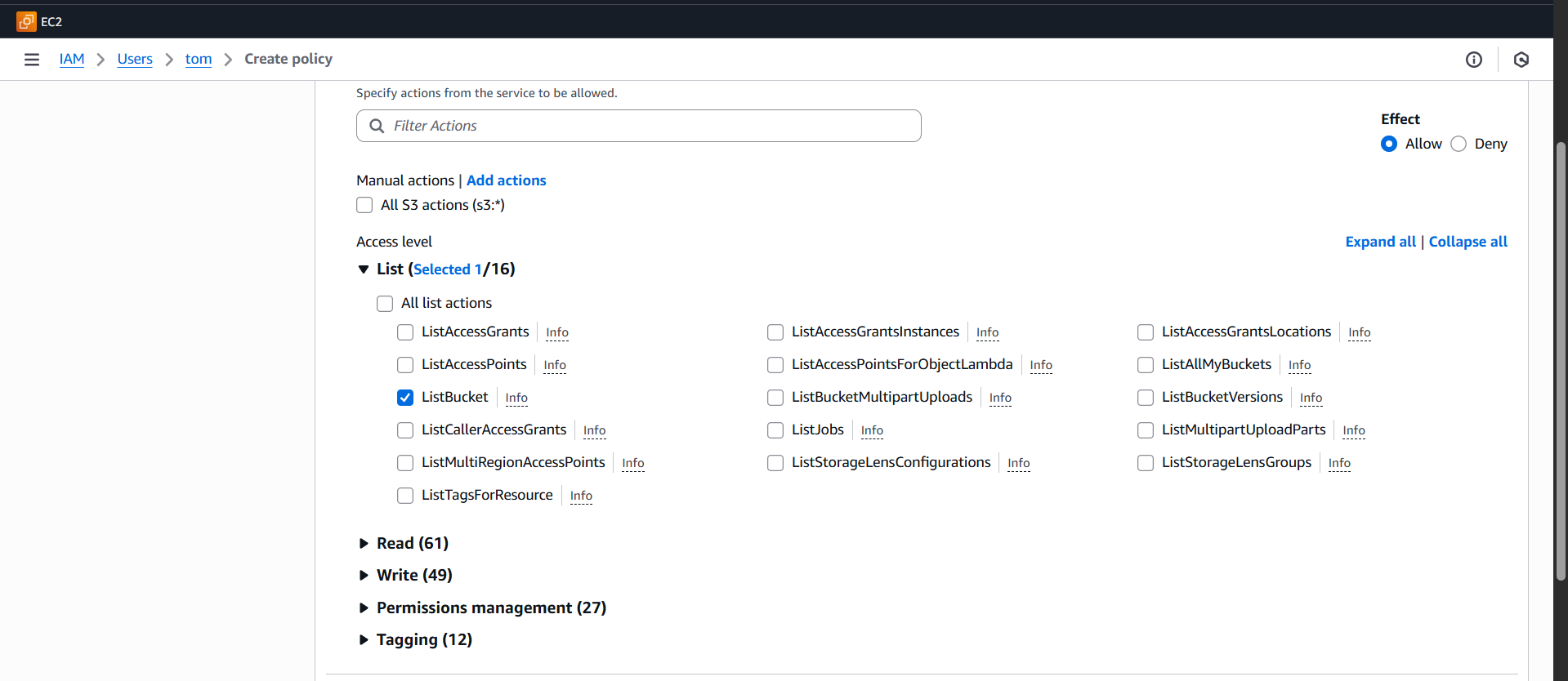
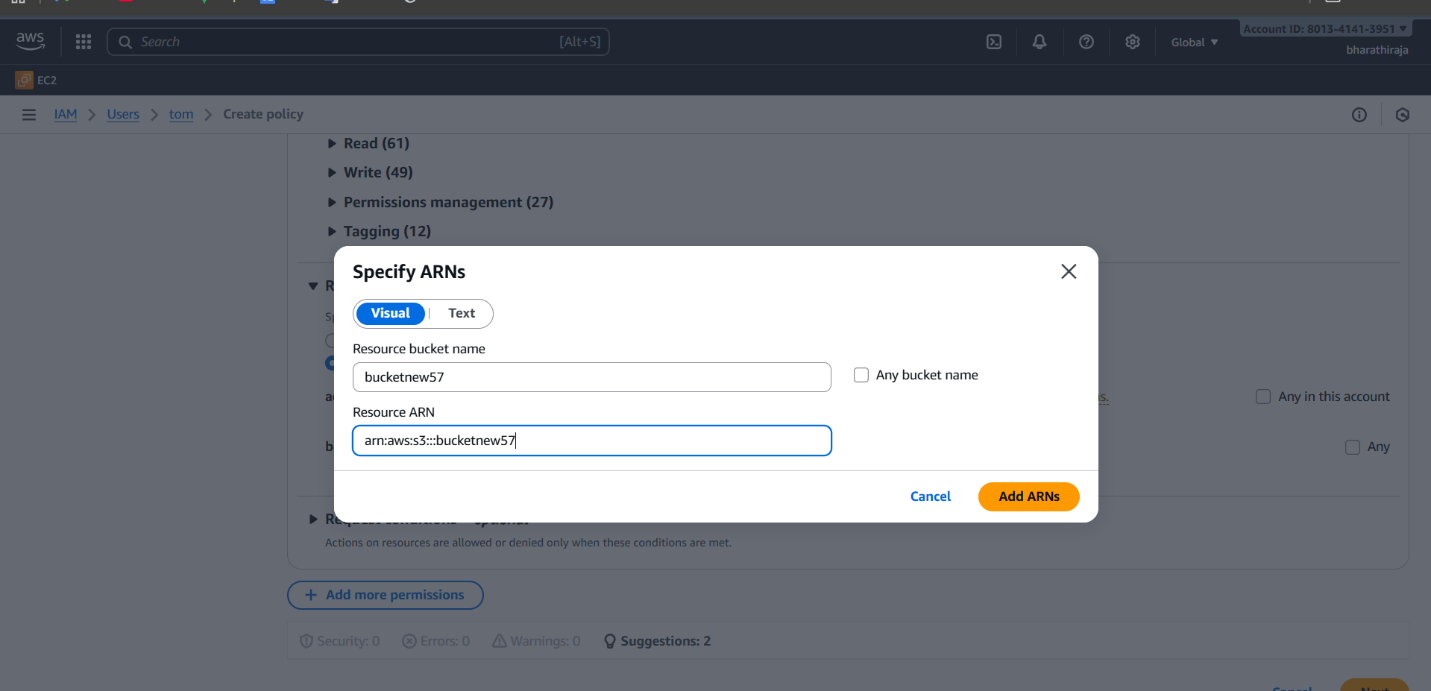
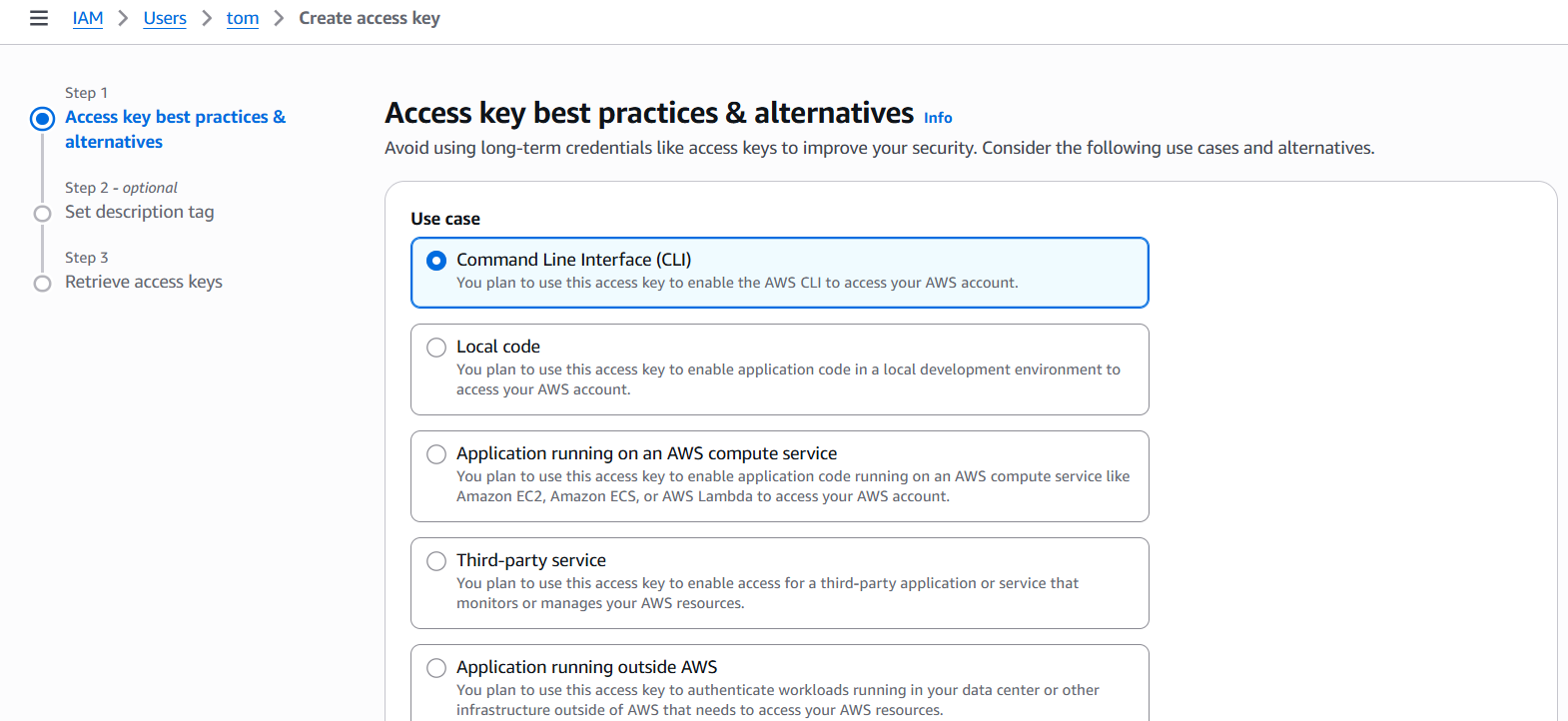
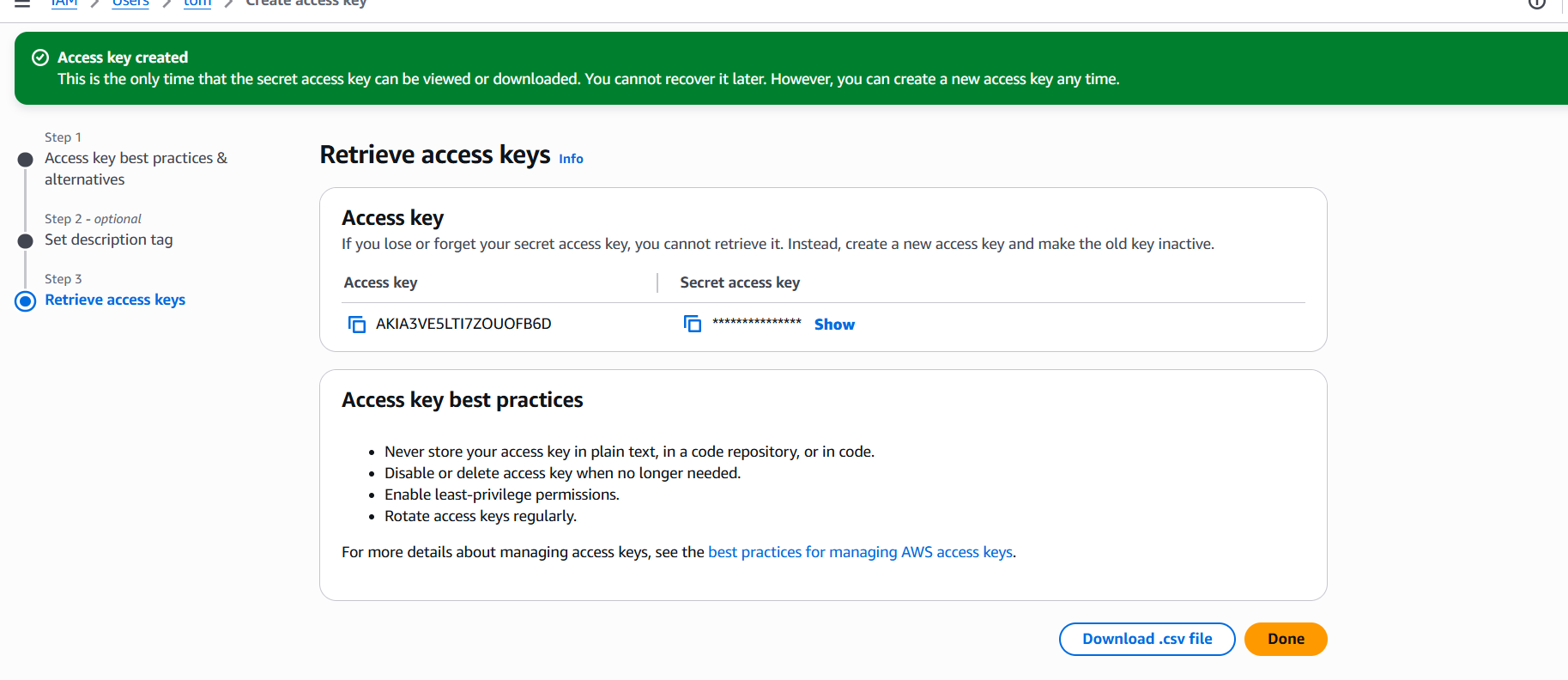
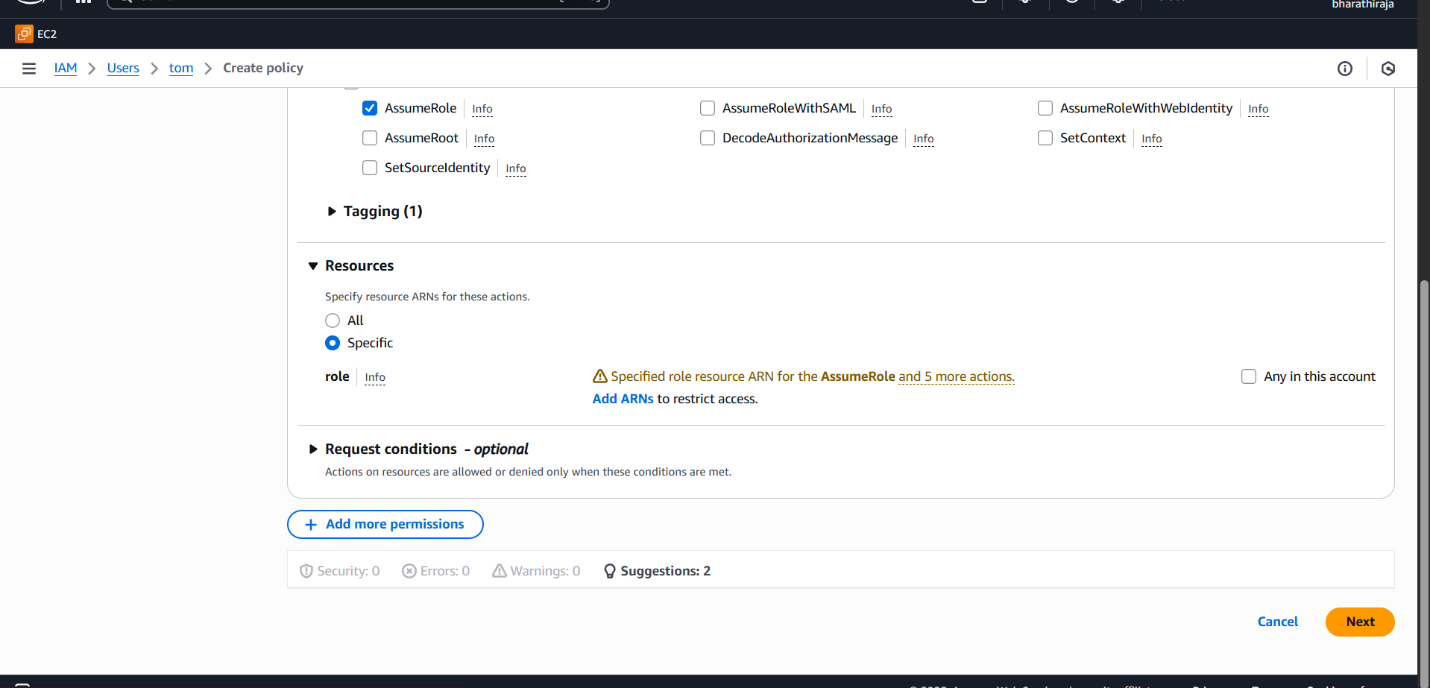
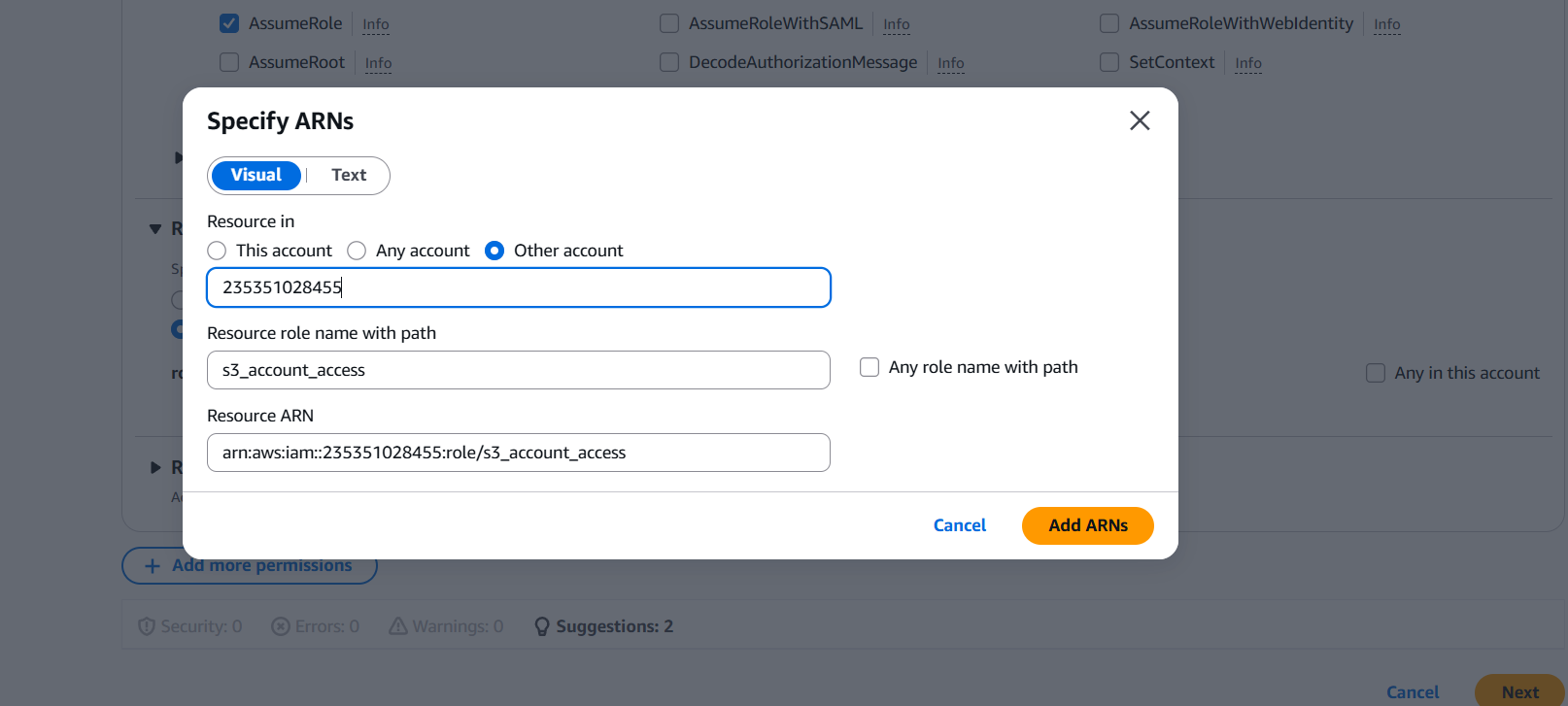


1. We have two accounts: Account A and Account B. The Account A user should access an S3 bucket in Account B.

<https://youtu.be/eUEJQhKax7A?si=secV1gUl8Yu_LmfL> => full details how to do







Great — here’s a **short, clear step-by-step** checklist you can follow right now. **Copy/paste** the policy JSON and CLI commands, replacing the placeholders (ACCOUNT\_A\_ID, ACCOUNT\_B\_ID, ACCOUNT\_A\_USER\_OR\_ROLE\_ARN, ROLE\_NAME, my-bucket-b, etc.) with your real values.

**Quick summary (recommended method — AssumeRole)**

**Goal:** create a role in **Account B** that the user in **Account A** can assume and which has S3 access to the bucket.

**1) Create trust policy in Account B (trust a specific Account A principal — do not use :root)**

Save as trust-policy.json:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": { "AWS": "arn:aws:iam::ACCOUNT\_A\_ID:user/YourUserName" },

"Action": "sts:AssumeRole"

}

]

}

If your Account A principal is a role, use arn:aws:iam::ACCOUNT\_A\_ID:role/RoleName.

**2) Create the role in Account B**

aws iam create-role \

--role-name CrossAccountS3ReadRole \

--assume-role-policy-document file://trust-policy.json

**3) Give that role S3 permissions (attached inline or managed). Example inline policy — save as role-policy.json:**

{

"Version":"2012-10-17",

"Statement":[

{ "Effect":"Allow", "Action":["s3:ListBucket"], "Resource":["arn:aws:s3:::my-bucket-b"] },

{ "Effect":"Allow", "Action":["s3:GetObject"], "Resource":["arn:aws:s3:::my-bucket-b/\*"] }

]

}

Attach it:

aws iam put-role-policy \

--role-name CrossAccountS3ReadRole \

--policy-name S3ReadPolicy \

--policy-document file://role-policy.json

(Or attach your existing managed policy:  
aws iam attach-role-policy --role-name CrossAccountS3ReadRole --policy-arn arn:aws:iam::ACCOUNT\_B\_ID:policy/account-b)

**4) In Account A — allow the user to assume the role**

Save assume-policy.json:

{

"Version":"2012-10-17",

"Statement":[

{

"Effect":"Allow",

"Action":"sts:AssumeRole",

"Resource":"arn:aws:iam::ACCOUNT\_B\_ID:role/CrossAccountS3ReadRole"

}

]

}

Attach to the Account A user (or group):

aws iam put-user-policy --user-name YourUserName --policy-name AllowAssumeCrossRole --policy-document file://assume-policy.json

**5) Configure local AWS CLI (Account A machine)**

Ensure ~/.aws/credentials contains the Account A credentials (permanent access key/secret) under [default]:

[default]

aws\_access\_key\_id = AKIA...

aws\_secret\_access\_key = ...

In ~/.aws/config add a profile to auto-assume:

[profile cross-account-b]

role\_arn = arn:aws:iam::ACCOUNT\_B\_ID:role/CrossAccountS3ReadRole

source\_profile = default

region = us-east-1

**6) Test (from Account A machine)**

Check identity (assume role via profile):

aws sts get-caller-identity --profile cross-account-b

If success you’ll see Account: ACCOUNT\_B\_ID and an assumed-role ARN.

List the bucket:

aws s3 ls s3://my-bucket-b --profile cross-account-b

**Common errors & fixes (from your earlier attempts)**

* **Roles may not be assumed by root accounts.** → your role trust policy used arn:aws:iam::ACCOUNT\_A\_ID:root. **Replace** that with the exact user or role ARN in Account A.
* **Partial credentials found ... missing: source\_profile** → add source\_profile = default to the profile and ensure ~/.aws/credentials has [default] keys.
* **Unable to parse config file** → check /root/.aws/config for bad formatting (use plain INI format; no stray characters).
* **AccessDenied when listing objects** → ensure both (a) the assumed role has S3 permissions (role policy), and (b) the bucket/object ARNs are correct (s3:ListBucket uses arn:aws:s3:::bucket, s3:GetObject uses arn:aws:s3:::bucket/\*).

**Alternate (simpler) option — Bucket policy + Account A identity**

If you prefer not to use AssumeRole, you can:

1. Add a **bucket policy** in Account B allowing a specific Account A principal (arn:aws:iam::ACCOUNT\_A\_ID:user/YourUserName) s3:GetObject/s3:ListBucket.
2. Give the Account A user an identity policy that allows s3:GetObject on arn:aws:s3:::my-bucket-b/\*.

Both resource policy (bucket) and identity policy must permit the action.

If you want, I can now:

* **(A)** Generate trust-policy.json, role-policy.json, assume-policy.json, and the exact ~/.aws/config/credentials snippets **filled with your real Account A ID, Account A user name, Account B ID, role name and bucket name** — paste those values and I’ll produce ready-to-run files.
* **(B)** Or walk you through the exact CLI commands you should run next on whichever account (A or B) you are currently logged into.

1. (Collaborate with a team member and execute this. This is mostly asked in every interview.)

**Q:**  
*"You have two AWS accounts: Account A and Account B. A user in Account A needs access to an S3 bucket located in Account B. How will you set this up step by step?"*

**🔹 Best Possible Answer (Step by Step Explanation)**

**Answer:**  
To enable cross-account access from Account A to Account B’s S3 bucket, I will use **IAM Role + Bucket Policy**. Here’s how:

**Step 1: Create an IAM Role in Account B**

* Go to **Account B (Bucket Owner)**.
* Create a new IAM Role with **Trusted Entity = Another AWS Account**.
* Enter **Account A’s Account ID** as the trusted account.
* Attach a policy that allows **S3 access to the specific bucket** (for example: s3:GetObject, s3:ListBucket).

✅ Example **Trust Policy** (Account B Role):

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::ACCOUNT-A-ID:root"

},

"Action": "sts:AssumeRole"

}

]

}

✅ Example **Role Policy** (Access to S3 bucket in Account B):

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["s3:ListBucket"],

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

},

{

"Effect": "Allow",

"Action": ["s3:GetObject"],

"Resource": "arn:aws:s3:::my-bucket/\*"

}

]

}

**Step 2: Update S3 Bucket Policy in Account B**

* Add a bucket policy so the role can access it.

✅ Example:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "AllowCrossAccountRoleAccess",

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::ACCOUNT-B-ID:role/CrossAccountS3Role"

},

"Action": ["s3:GetObject", "s3:ListBucket"],

"Resource": [

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

"arn:aws:s3:::my-bucket/\*"

]

}

]

}

**Step 3: Allow Account A User to Assume the Role**

* In **Account A**, attach a policy to the IAM user that lets them **assume the role from Account B**.

✅ Example (Account A User Policy):

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::ACCOUNT-B-ID:role/CrossAccountS3Role"

}

]

}

**Step 4: Assume Role from Account A**

From Account A’s AWS CLI:

aws sts assume-role \

--role-arn arn:aws:iam::ACCOUNT-B-ID:role/CrossAccountS3Role \

--role-session-name CrossAccountSession

This returns **temporary security credentials** (AccessKey, SecretKey, SessionToken).

**Step 5: Use Temporary Credentials**

Export the keys:

export AWS\_ACCESS\_KEY\_ID=xxxx

export AWS\_SECRET\_ACCESS\_KEY=xxxx

export AWS\_SESSION\_TOKEN=xxxx

Now list the bucket:

aws s3 ls s3://my-bucket

✅ Done! The Account A user can now access the S3 bucket in Account B.

**🔹 How to Explain in an Interview**

👉 *"I will not directly share credentials between accounts. Instead, I’ll create a role in Account B with a trust relationship to Account A, allow that role to access the S3 bucket, and let the Account A user assume that role using sts:AssumeRole. Then the user gets temporary credentials to access Account B’s S3 bucket. This is secure, auditable, and the recommended AWS best practice."*

**🔹 Follow-up Questions They Might Ask**

**Q1:** What if multiple users in Account A need access?  
**A:** We can allow all those users to assume the role. Instead of attaching policies user by user, we can use IAM Groups or even give access to the root of Account A.

**Q2:** Why not just make the bucket public?  
**A:** That’s not secure. Cross-account IAM roles are best practice.

**Q3:** What if access must be read + write?  
**A:** Modify the S3 bucket policy and role policy to include s3:PutObject.