AWS Identity and Access Management

Author: Oluwaseun Osunsola
Environment: AWS
Project Link: https://github.com/Oluwaseunoa/DevOps-Projects/tree/main/Cloud%20Computing/AWS%20Identity%20and%20Access%20Management

Overview

We're going to learn about AWS Identity and Access Management (IAM) which helps control who can access what in Amazon Web Services. We'll cover things like users, roles, policies, and groups, and we'll also show you how to actually set them up to keep your AWS resources safe.

But before we get into all that, let's make sure you understand the basics of cloud computing. If terms like "Cloud" sound new to you, it's a good idea to go back and review some earlier materials to get a solid grasp of what it's all about.

Project Goals

- Understand AWS Identity and Access Management (IAM) principles and components.
- Learn to create and manage IAM policies for regulating access to AWS resources securely.
- Apply IAM concepts practically to control access within AWS environments.
- Explore best practices for IAM implementation and security in AWS.

Learning Outcomes

- Recognize IAM components like users, roles, policies, and groups.
- Create and manage IAM policies to define permissions for users and roles.
- Set up IAM users, groups, and roles to control access to AWS services.
- Understand IAM best practices for maintaining security and managing access to AWS resources.

Key Concepts

What is IAM?

IAM, or Identity and Access Management, is the gatekeeper for your AWS resources, deciding who gets in and what they're allowed to do once inside.

Imagine you have a big digital "house" full of all your AWS stuff—your data, your applications, the whole shebang. You don't want just anyone wandering in and messing around with your things, right? That's where IAM steps in.

It's like having your own VIP list for your digital world. IAM helps keep your AWS resources safe, ensuring only the right people get in and can only do what you allow. It's all about keeping your digital house in order and protecting your precious stuff from unwanted guests.

Note: AWS resources are the various services and tools provided by Amazon Web Services (AWS) that users can utilize to build and manage their applications and infrastructure in the cloud.

What is IAM User?

IAM users are individual accounts for different people or entities within your AWS environment.

For example, if you have a team working on a project, you can create separate IAM users for each team member. Each IAM user has a unique username and password, allowing them to access the AWS resources needed for their work.

IAM users help manage and control access to your AWS resources securely, ensuring each user only has access to the resources required for their tasks.

What is IAM Role?

An IAM role defines what someone or something (like an application or service) can do within your AWS account. Each role has a set of permissions that determine which actions it can perform and which AWS resources it can access.

For example, you might have an "admin" role with full access to all resources or a "developer" role that only allows access to certain services for building applications.

Imagine a visitor needing temporary access to your house to fix something. Instead of giving them a permanent key (IAM user), you give them a temporary key (IAM role) that works for a limited time and grants access to specific rooms (AWS resources).

IAM roles are flexible and can be assumed by users, services, or applications as needed. They are used for tasks like granting permissions to AWS services, allowing cross-account access, or providing temporary access to external users. IAM roles enhance security and efficiency by providing controlled access without permanent credentials.

What is IAM Policy?

An IAM policy is a set of rules defining what actions a role or user can take. Think of it as a rulebook outlining which actions are allowed and which are not, ensuring secure and controlled access to your AWS resources.

For example, the rulebook might say the "admin" key (IAM role or user) can open any door and perform any action within the house (AWS resources), while the "viewer" key can only open certain doors and look around without making changes.

IAM policies define permissions for IAM roles and users, specifying which AWS resources they can access and what actions they can take. They are essential for maintaining security and ensuring only authorized actions are performed.

What is IAM Group?

IAM Groups are collections of IAM users. Instead of managing permissions for each user individually, you can organize users into groups based on roles or responsibilities.

Think of IAM Groups as teams based on tasks. You might have a group for developers, another for administrators, and so on. Instead of setting permissions for each person one by one, you set them for the whole group at once.

For example, for a development team, you can create an IAM Group called "Developers" and add all developers to it. Assign permissions to the group, and all developers get the same resource access, simplifying management.

Best Practices

- **Give only the permissions needed**: Avoid granting more access than necessary.
- **Use roles instead of users**: Roles are safer and can be used when needed.
- **Review roles regularly**: Remove unused roles to keep things tidy and secure.
- **Add extra security with MFA**: Use Multi-Factor Authentication for extra protection.
- **Use ready-made policies**: They're safer and easier to use.
- **Keep policies simple**: Create separate policies for different tasks.
- **Keep track of changes**: Record who changes what.
- **Test policies before using them**: Ensure they work as intended before applying.
- **Use descriptive names**: Choose clear names for IAM groups to aid understanding and management.
- **Enforce strong password policies**: Encourage strong passwords with expiration and complexity requirements.

Note (difference between users and roles): Users are like individuals with permanent keys to access resources, tied to specific people. Roles are like special keys providing temporary access, usable by different users or programs as needed. Users are fixed to individuals, while roles are flexible for specific tasks.

For MFA, see Multi-Factor Authentication (MFA) for IAM.

Note on AWS policies:

- **Managed Policies**: Created by AWS, widely used.
- **Customer Managed Policies**: You create and manage them.
- **Inline Policies**: Made for one specific thing.

For further details, refer to Policies and permissions in IAM in the IAM documentation.

Practical Implementation

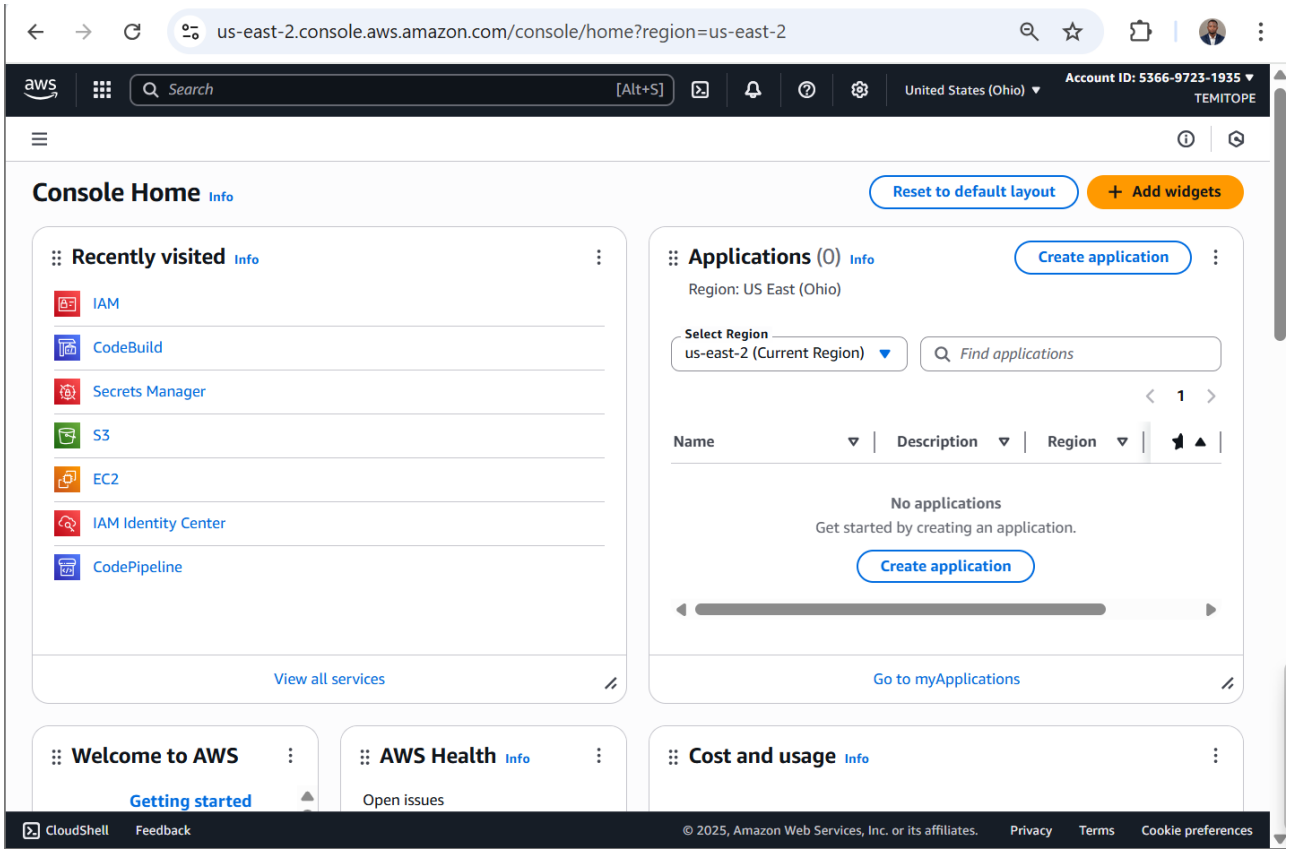
A growth marketing consultancy, GatoGrowFast.com, wants to grant access to AWS resources for employees Seun, Jack, and Ade. The project is divided into three parts:

1. Create a policy granting full EC2 access and assign it to user Seun, along with MFA setup.
2. Create a group, add users Jack and Ade, and create a policy granting full EC2 and S3 access for the group.
3. Enable MFA for user Seun to enhance security.

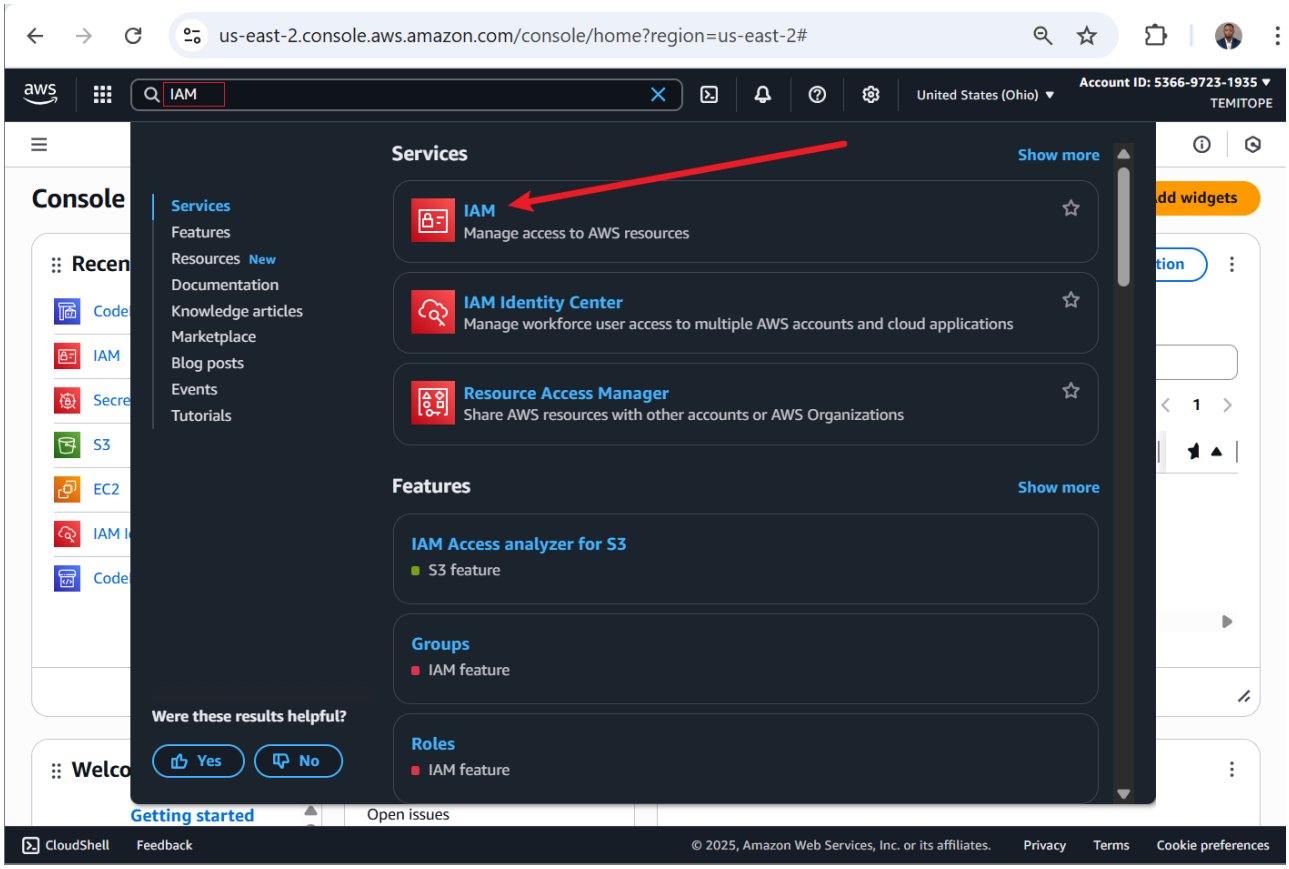
The following steps are ordered numerically based on the provided screenshot names, with each step linked to its corresponding screenshot. Note: Screenshots use "Seun" as the user name, consistent with the provided naming.

Part 1: Create EC2 Policy and IAM User (Seun)

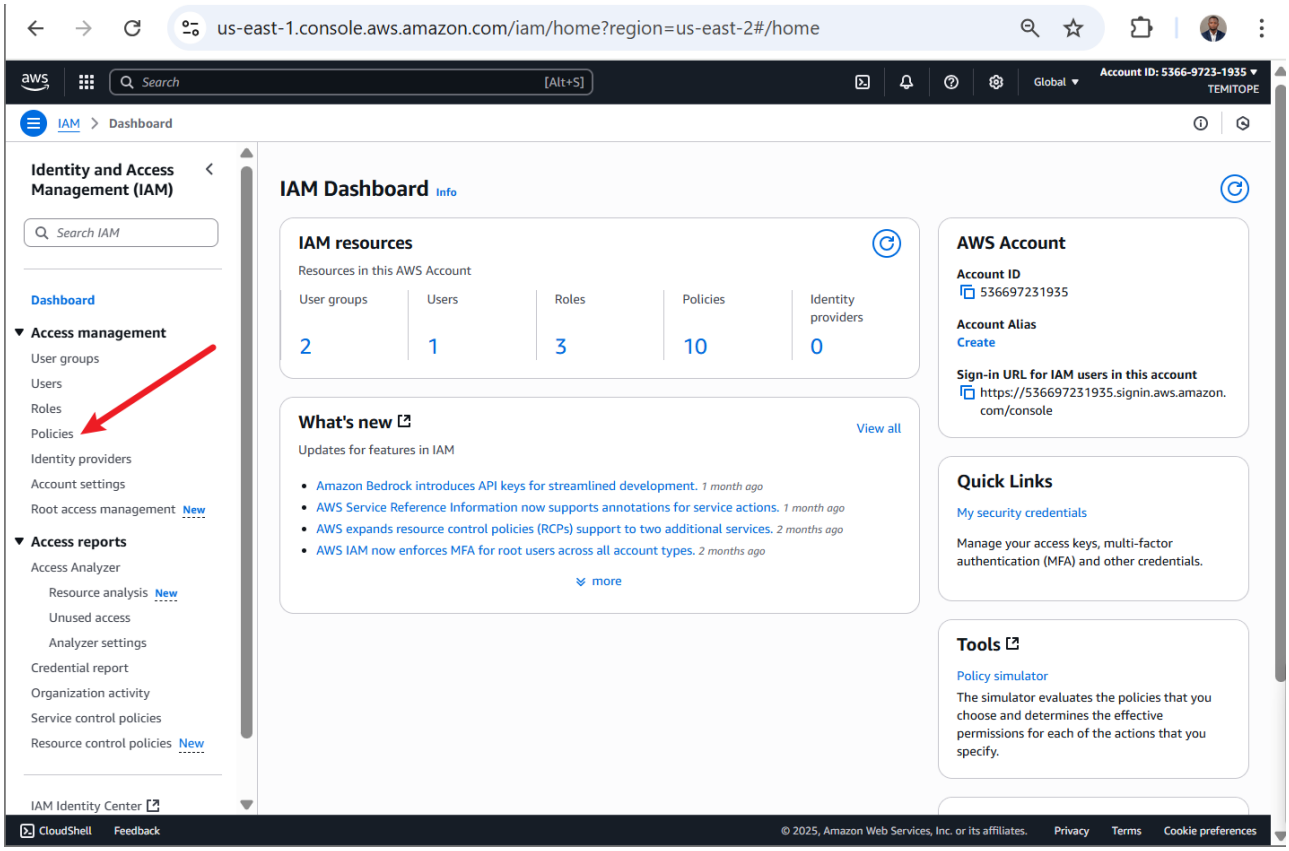
1. **Log In and Navigate to Your AWS Console**



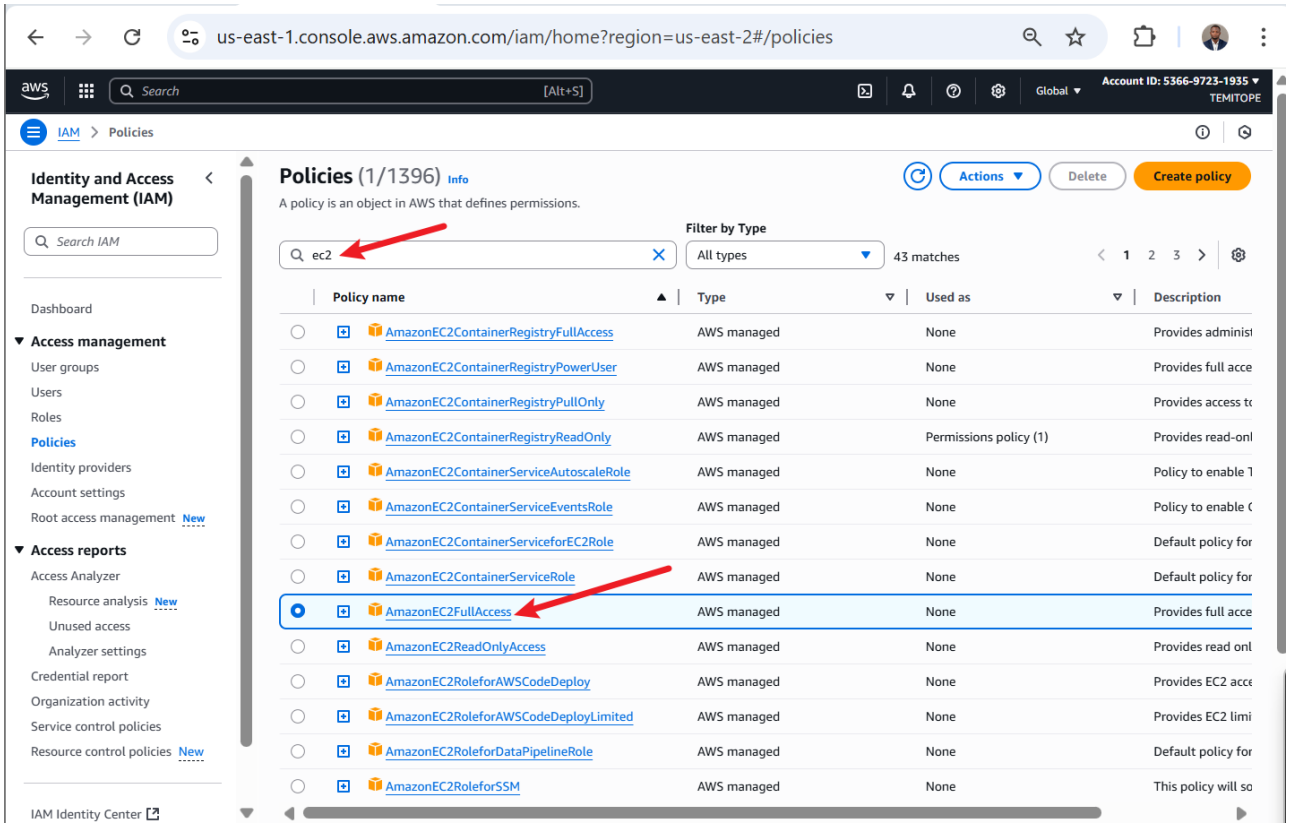
2. **Use the Services Search to Search for IAM and Click to Visit the Page**



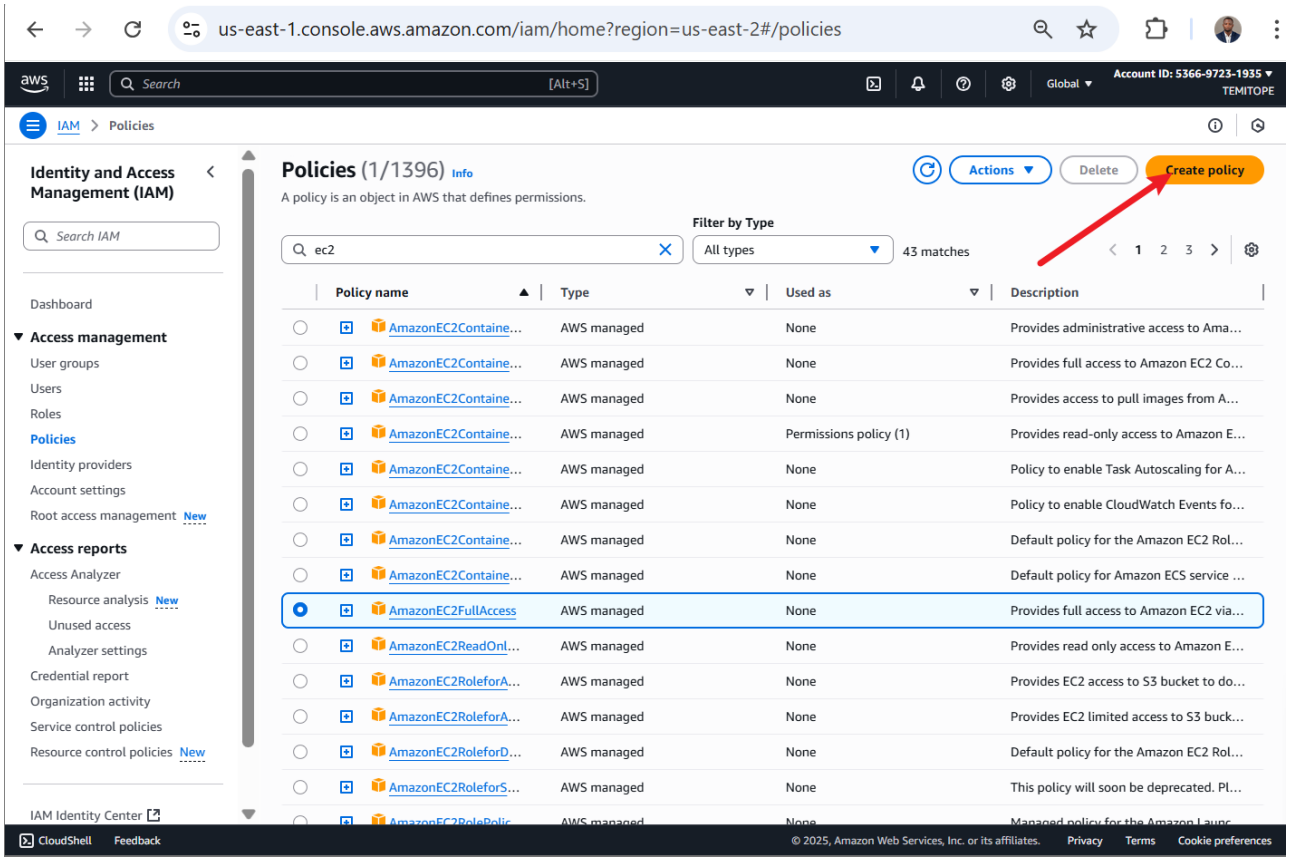
3. **Click on Policies on the Menu Option**



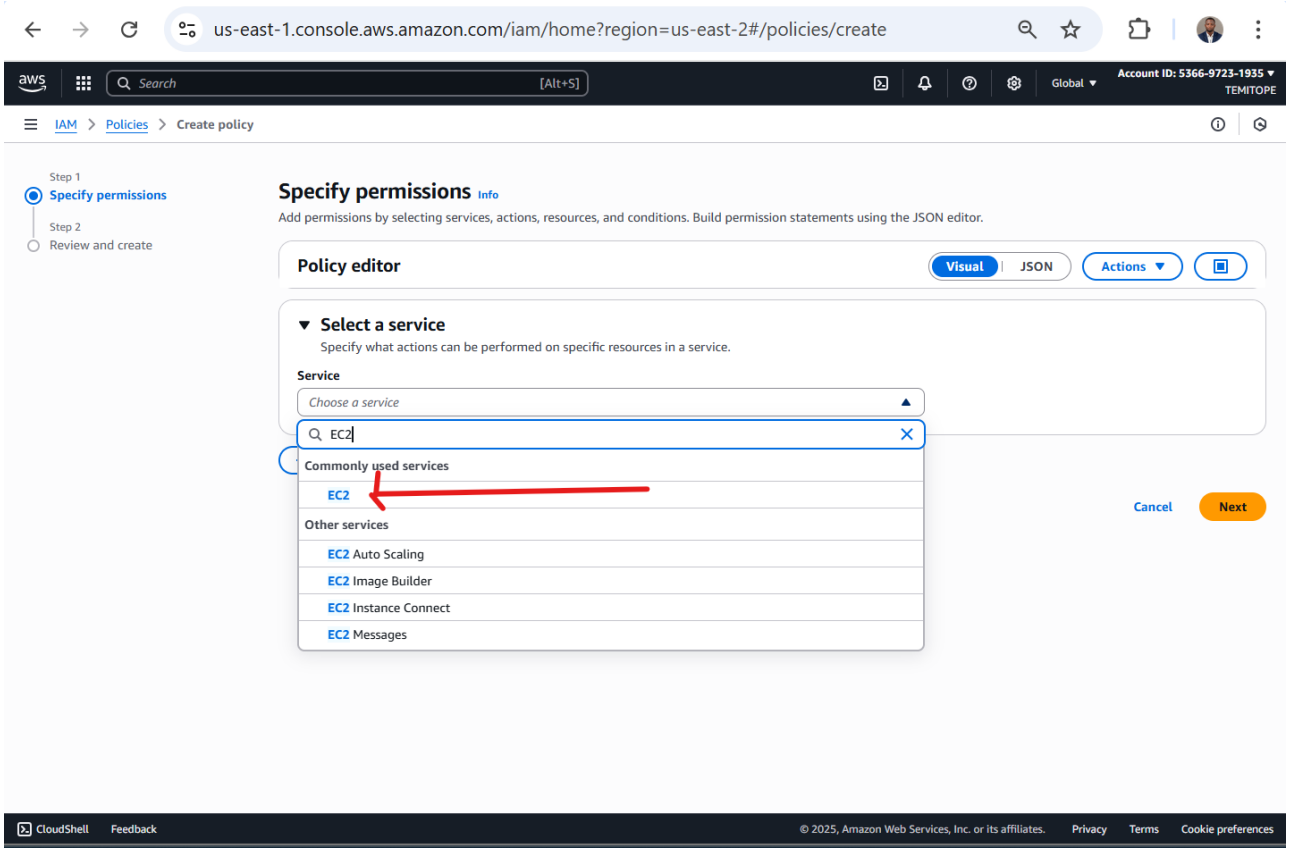
4. **Search for EC2 and Grant by Selecting AmazonEC2FullAccess**



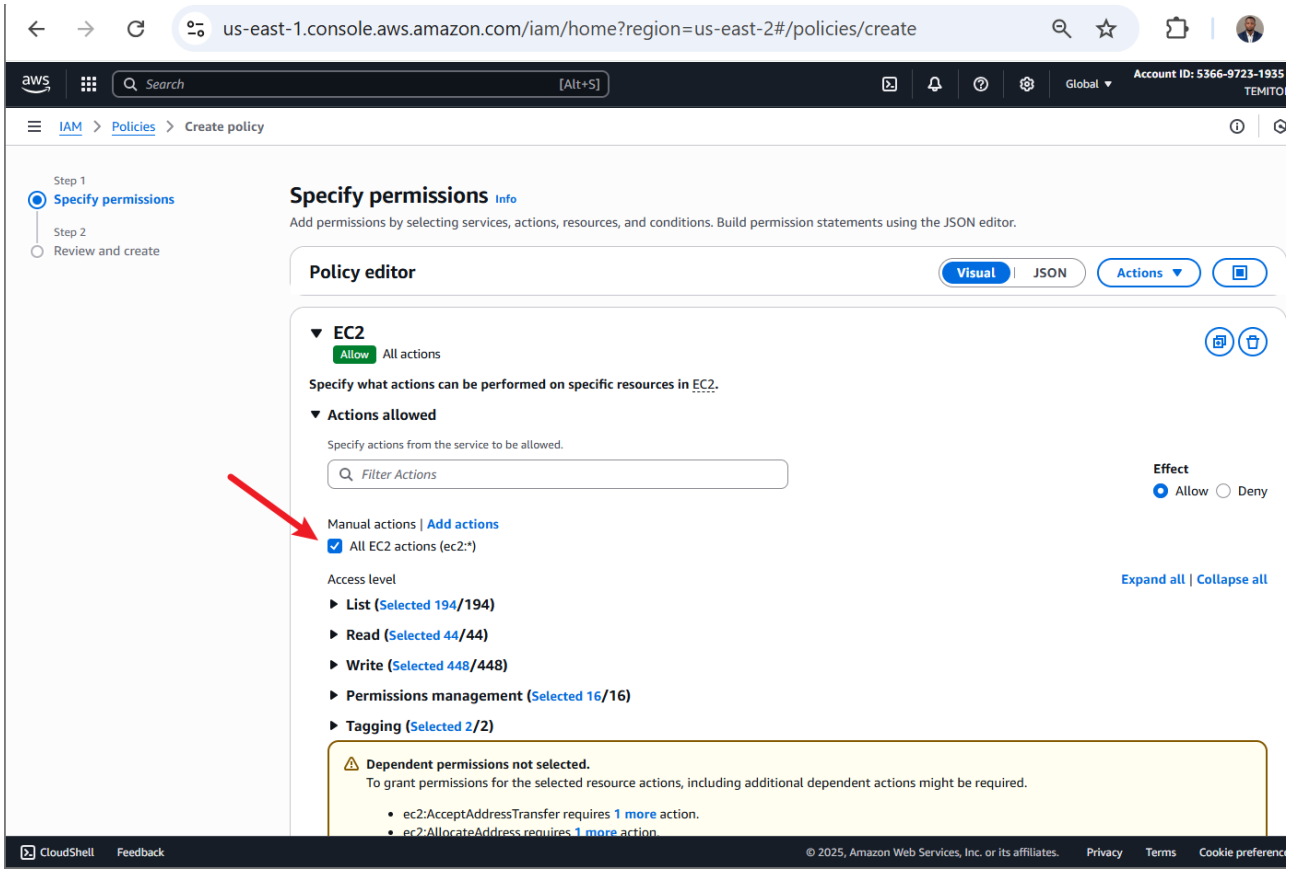
5. **Click Create Policy**



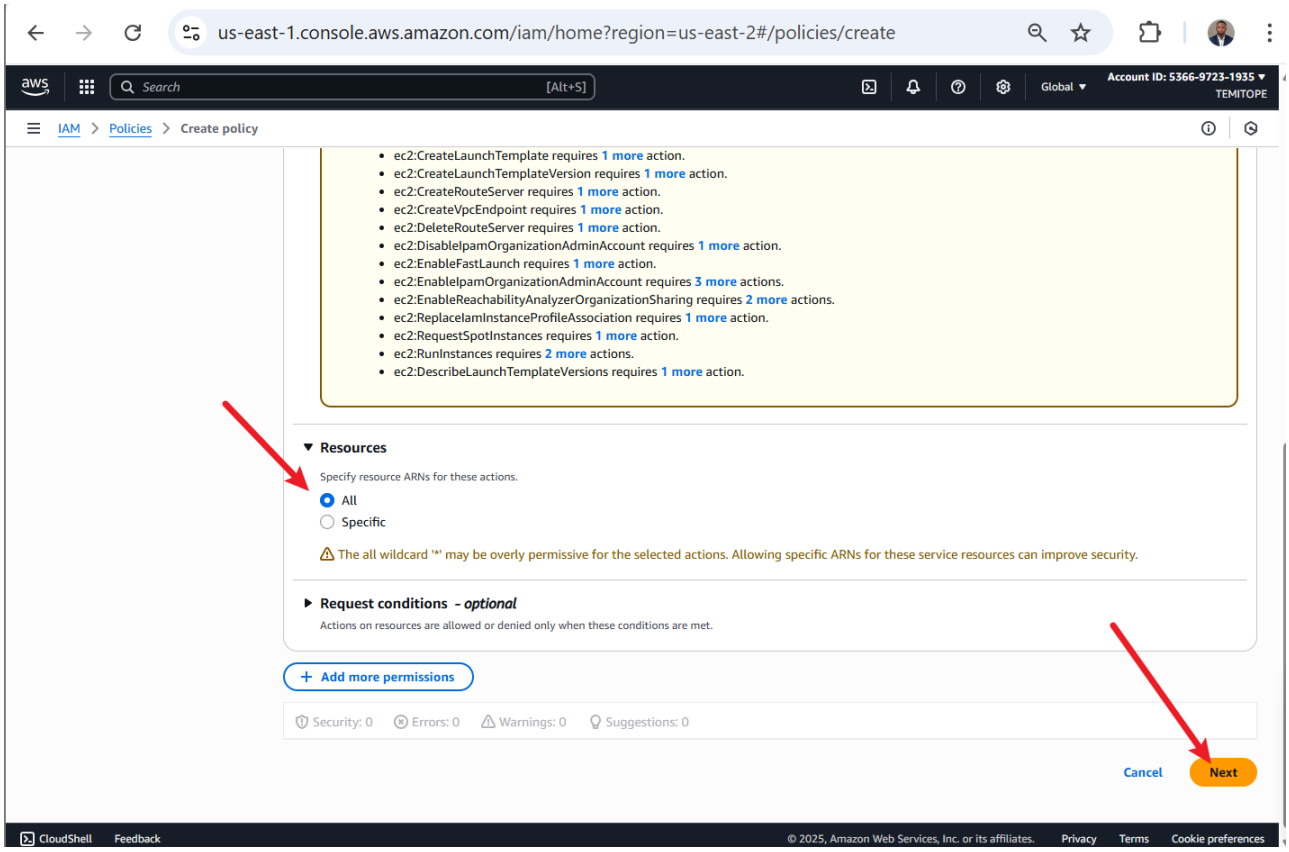
6. **Select EC2 Service**



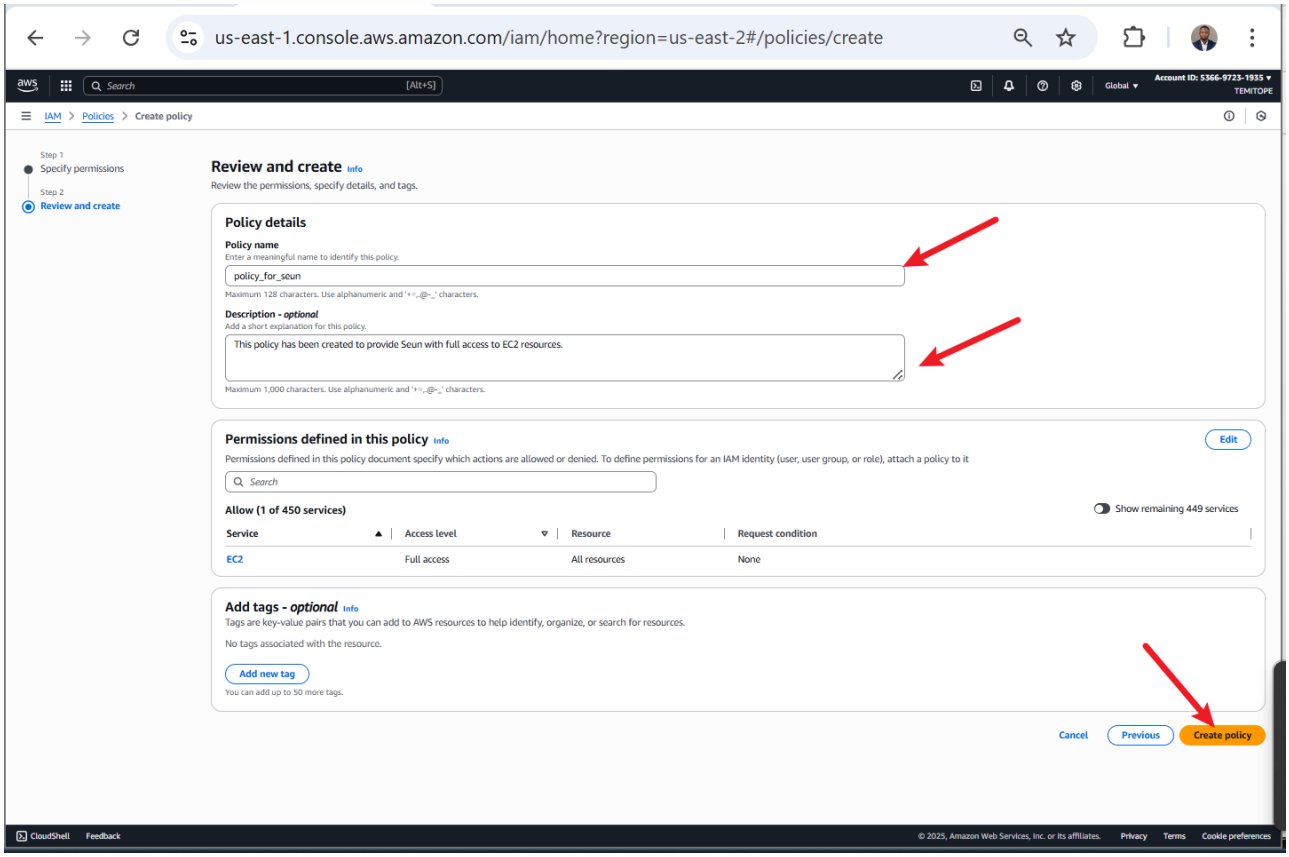
7. **Select All EC2 Actions Under Action and Scroll Down**



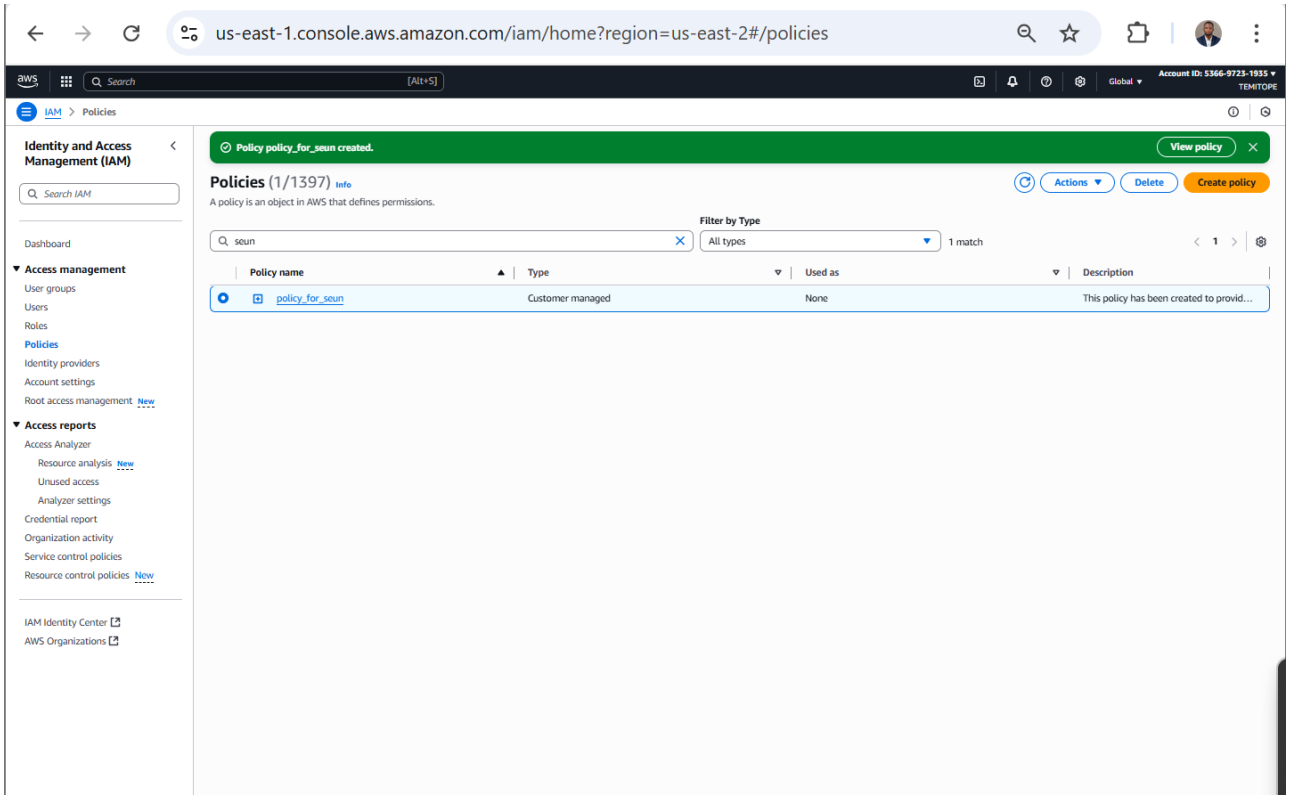
8. **Under Resources, Tick All and Click Next Button**



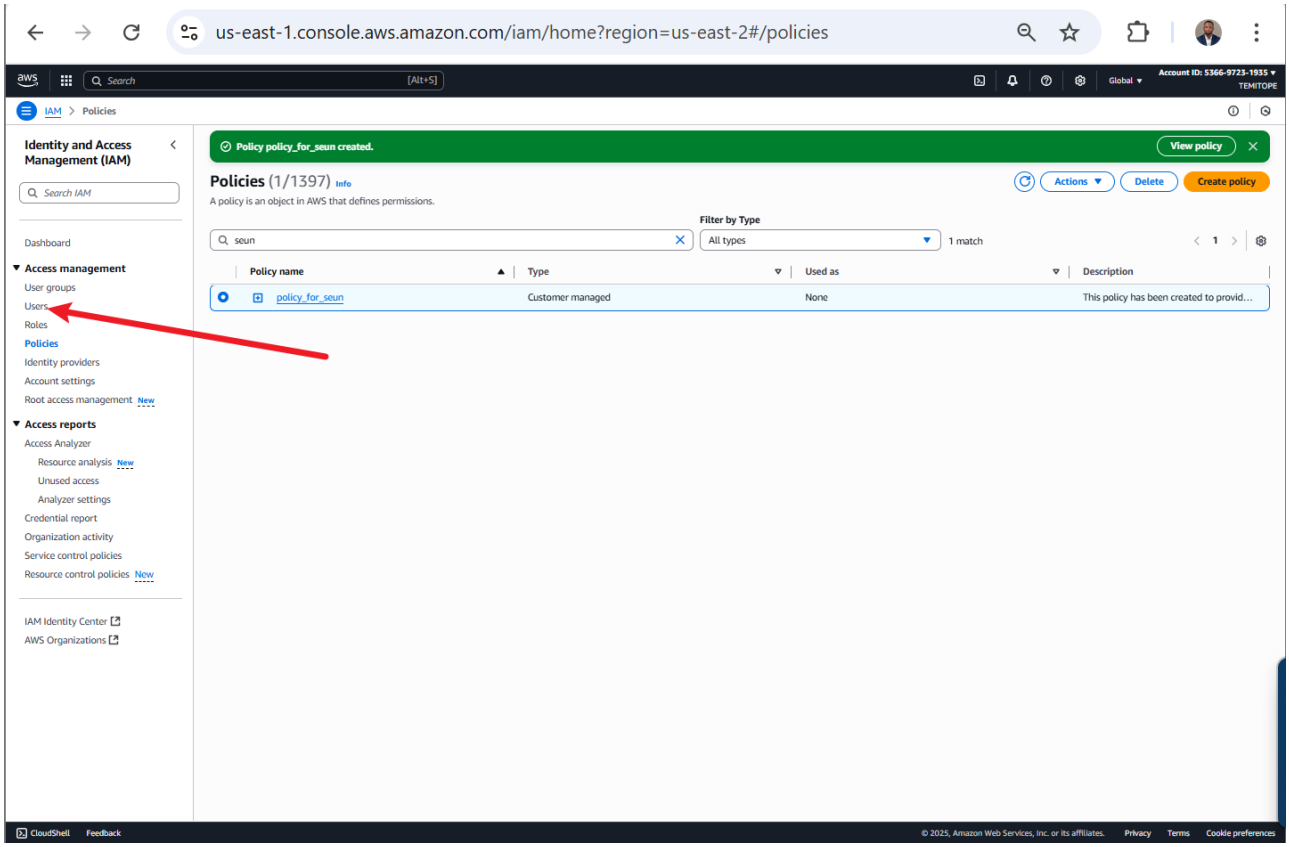
9. **Add Policy Names and Create Policy**



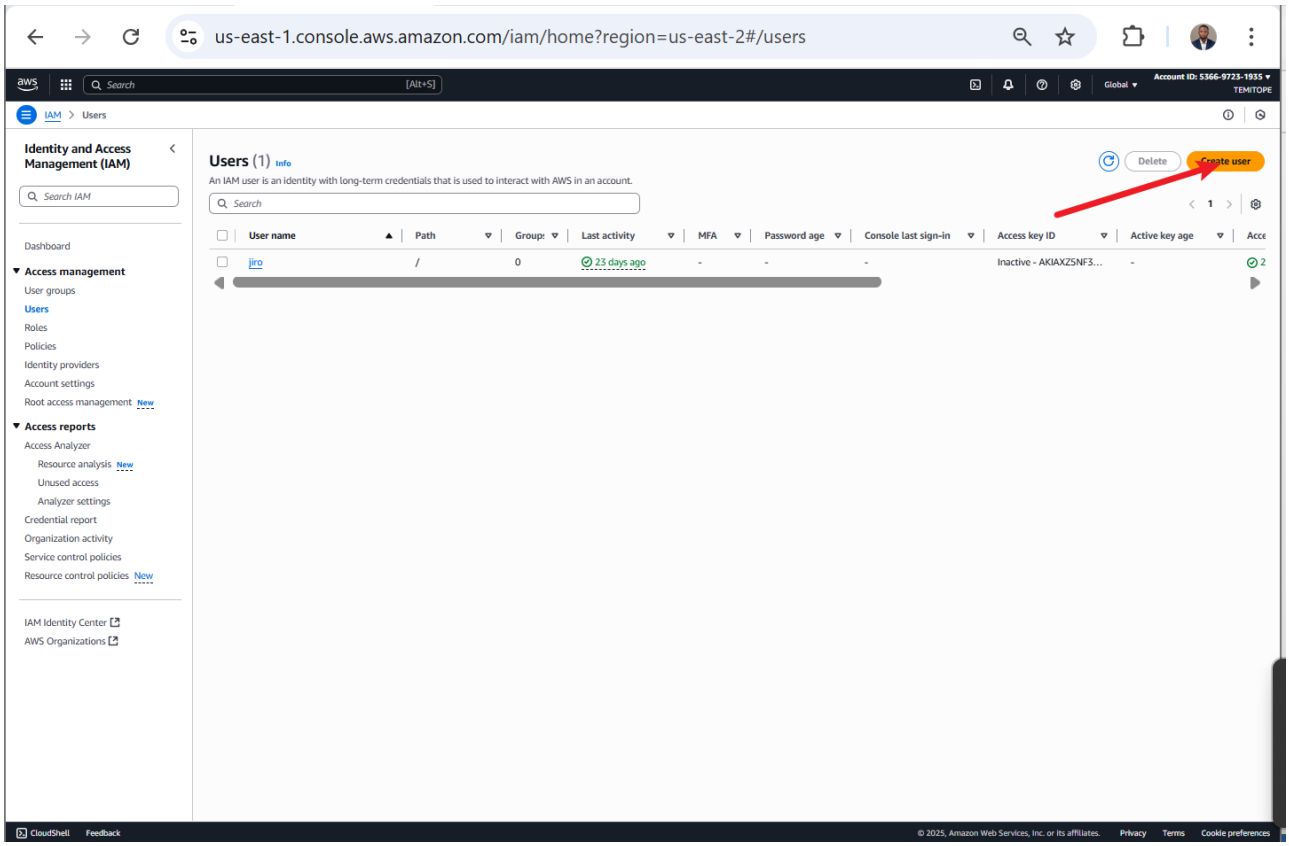
10. **Policy Successfully Created**



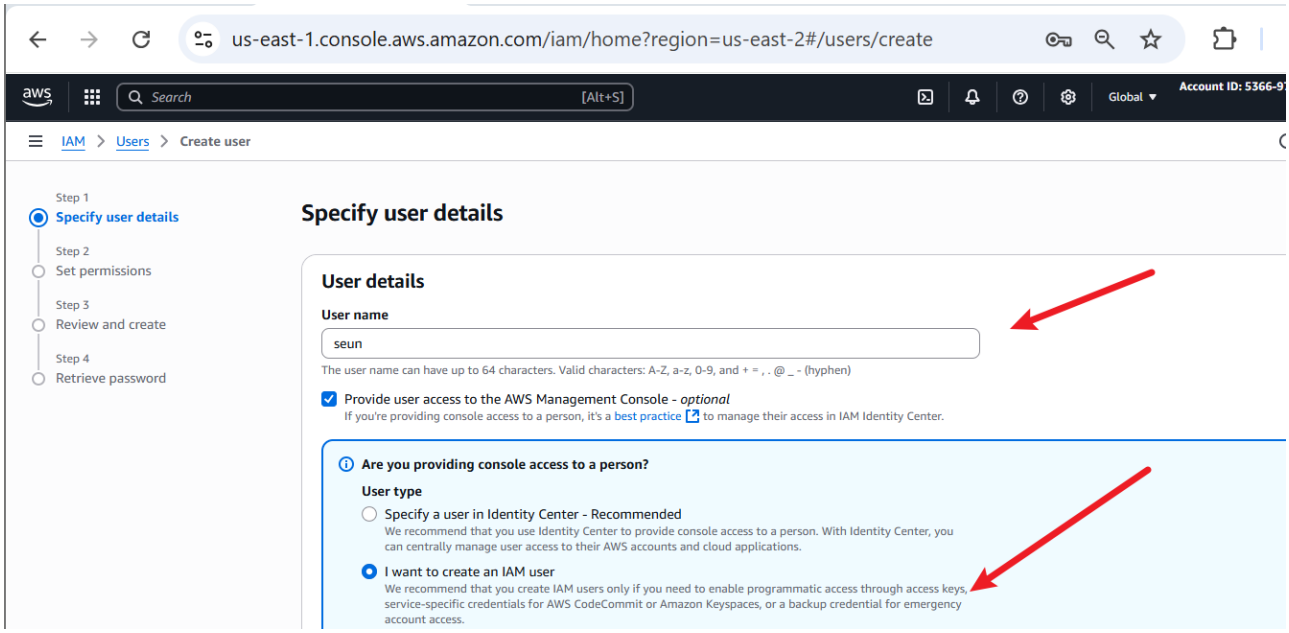
11. **Click on Users, Proceed to Create User**



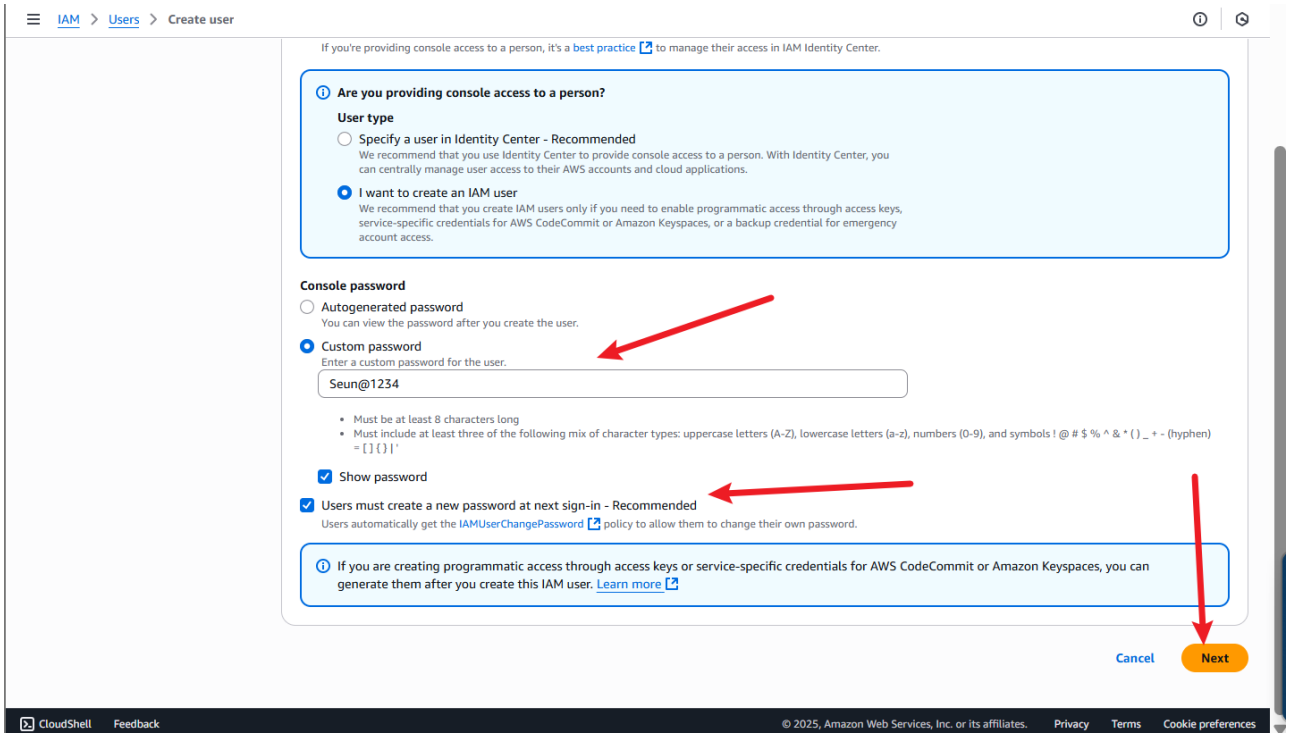
12. **On Users Page, Click Create User**



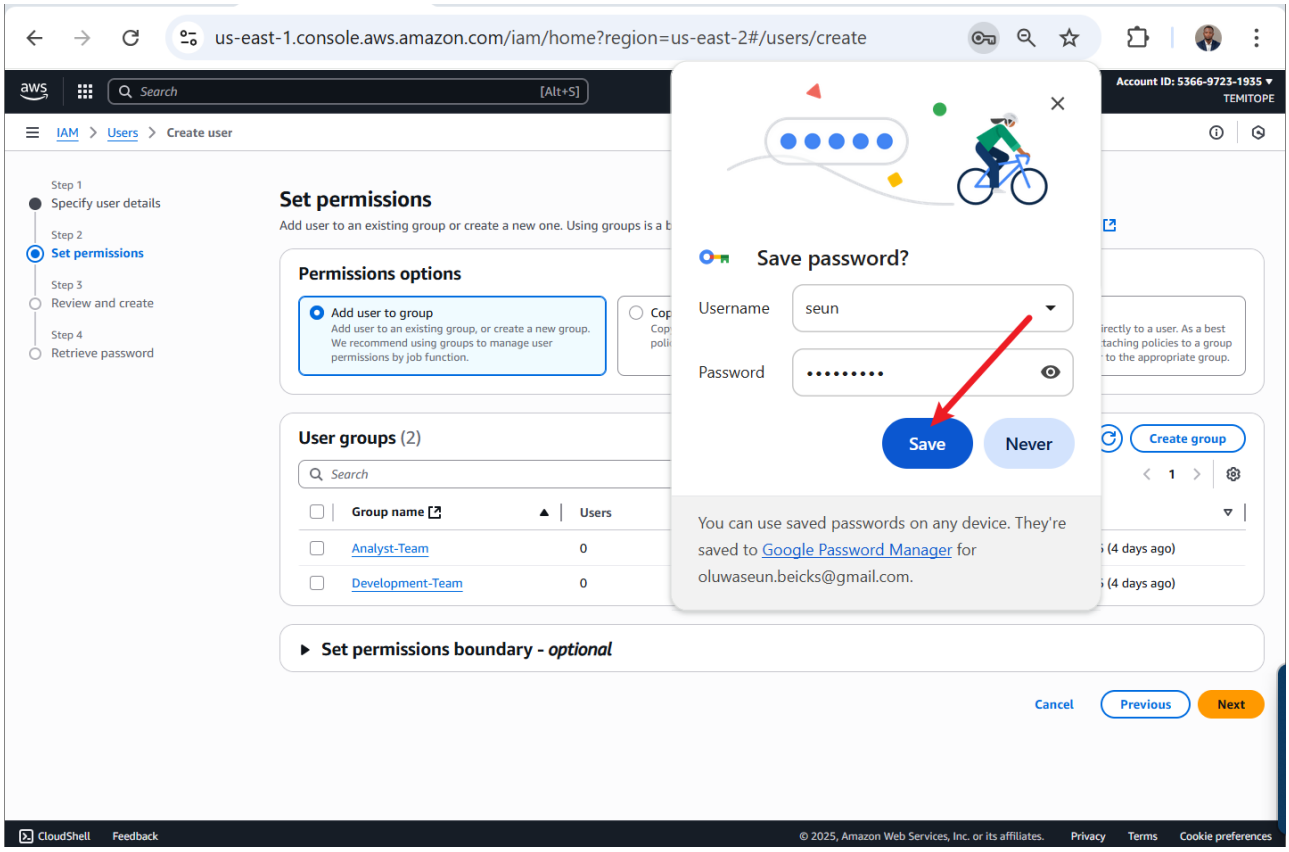
13. **Enter User Name, Give User Access to the Console, and Create User as an IAM User**



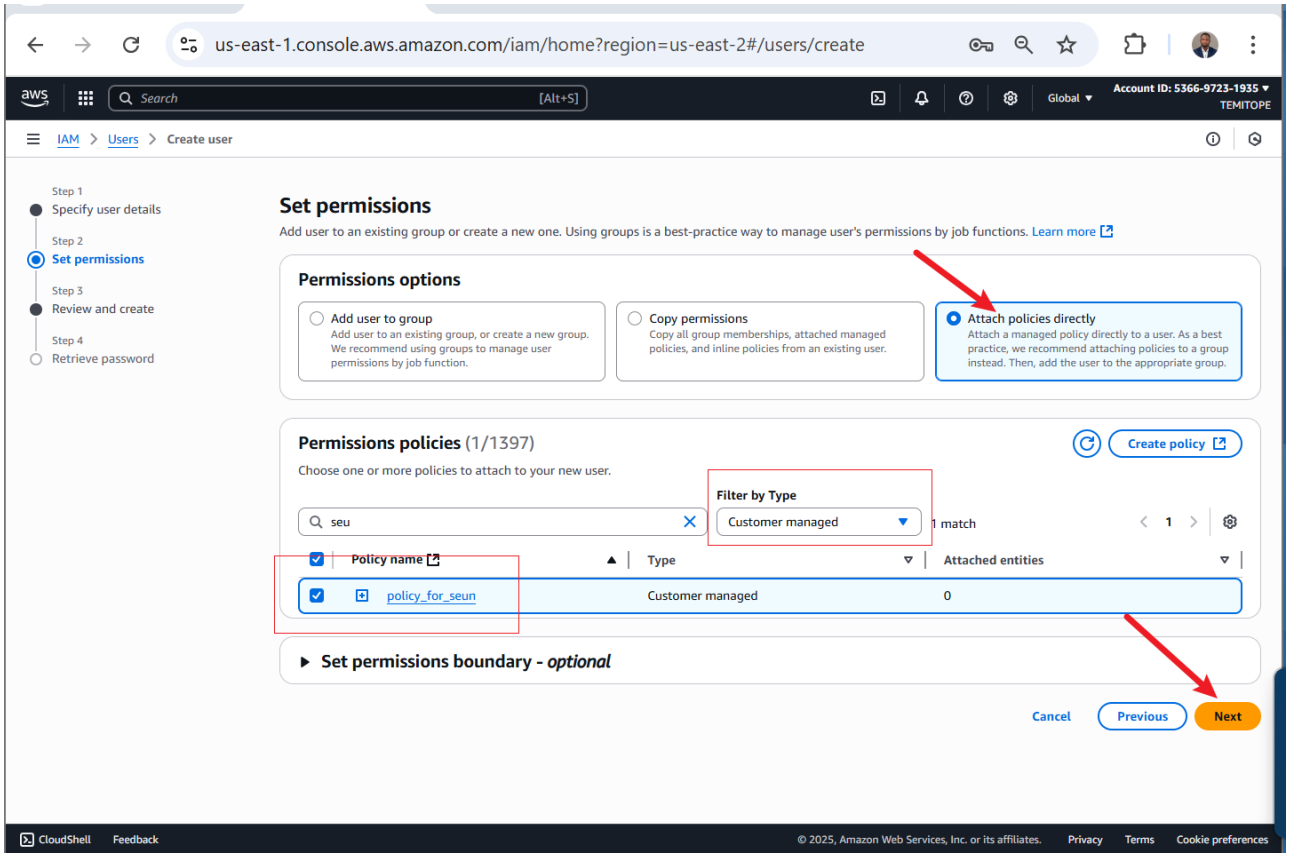
14. **Set Custom Password and Force User to Create New Password at Next Login and Click Next**



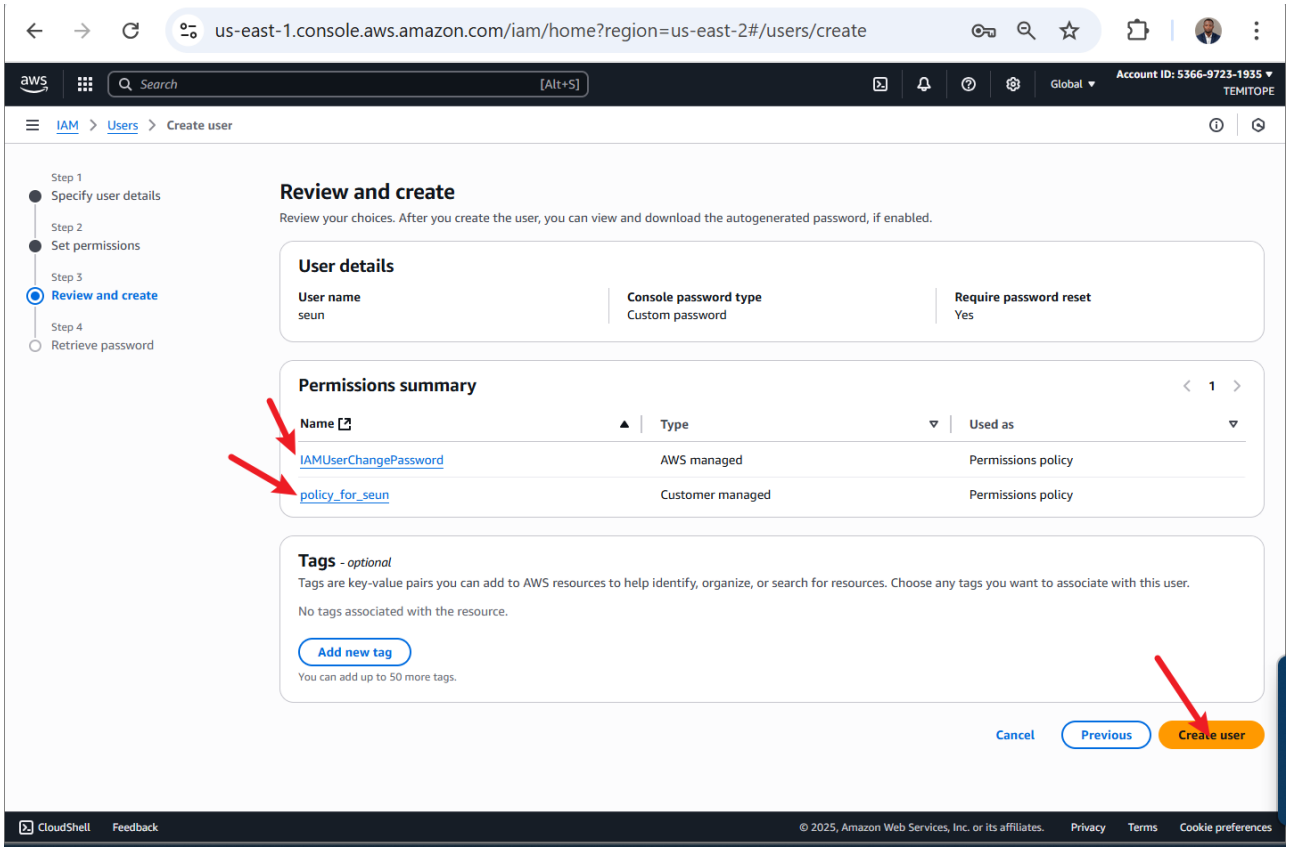
15. **You Can Click Save Password on Google Password Manager If You Prefer for Next Log In**



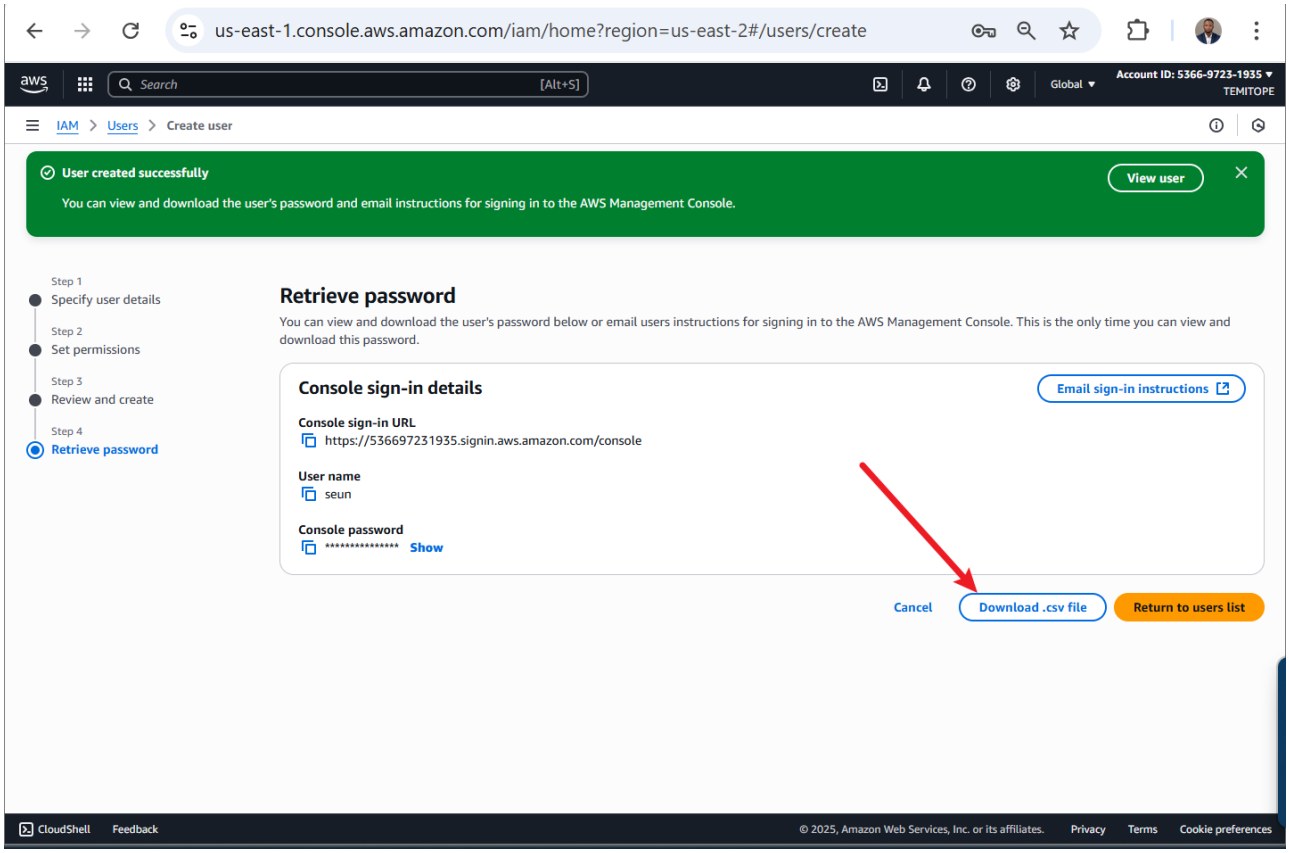
16. **On Permission Page, Click Attach Policy Directly, Filter Policy by Customer Managed Policy, Select Created User Policy, and Click Next**



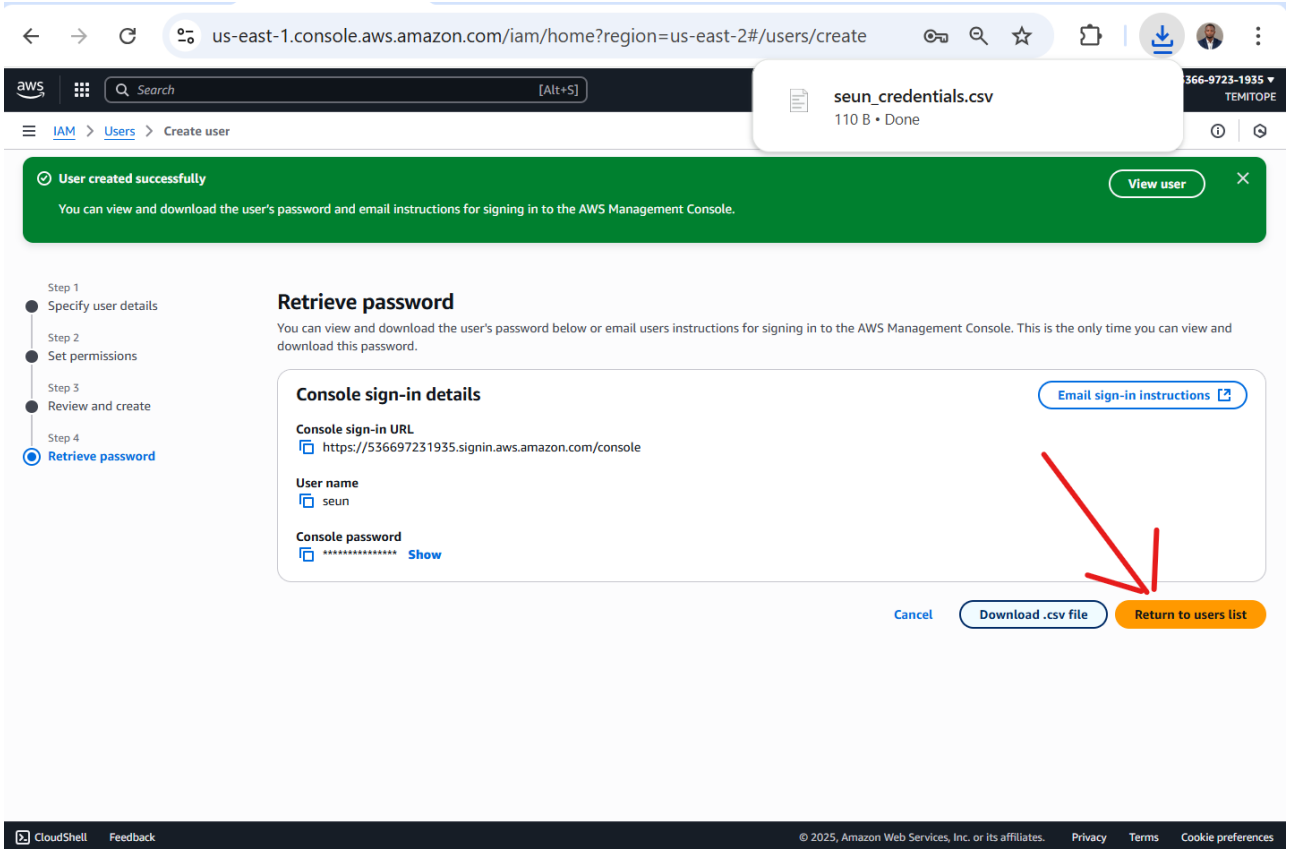
17. **Review and Create**



18. **User Seun Successfully Created, Now Download CSV File to Retrieve Chosen Password**

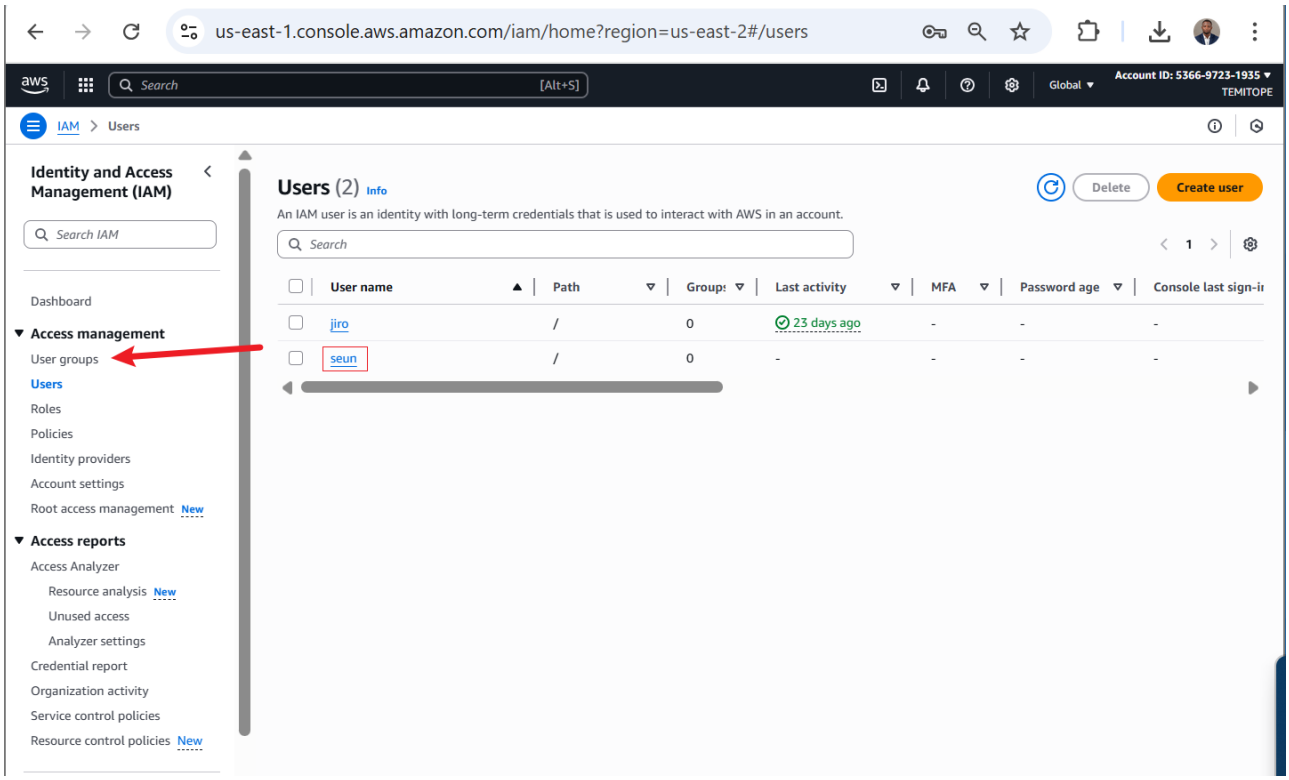


19. **Credential Successfully Downloaded, Proceed to User List Page**

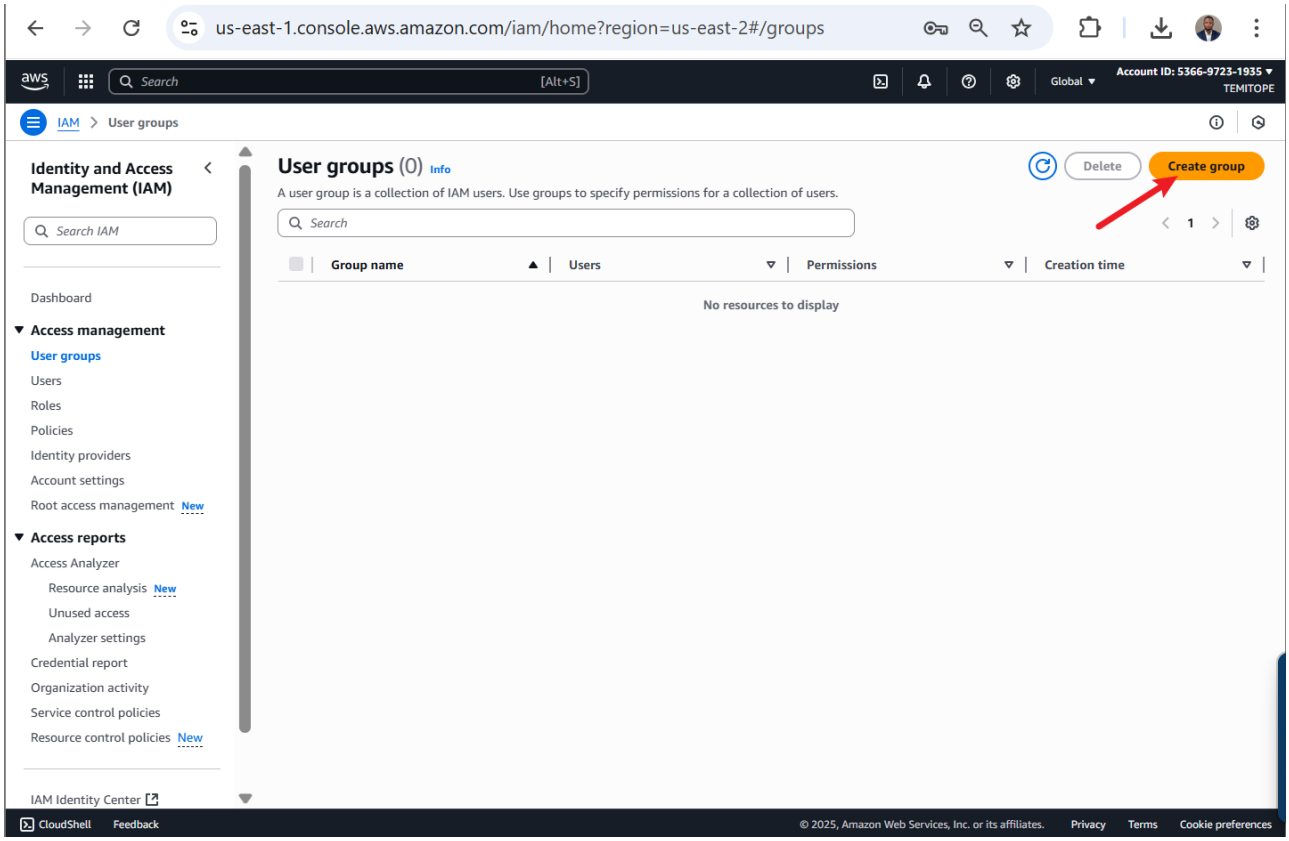


Part 2: Create Group, Add Users (Jack and Ade), Create EC2/S3 Policy, and Attach to Group

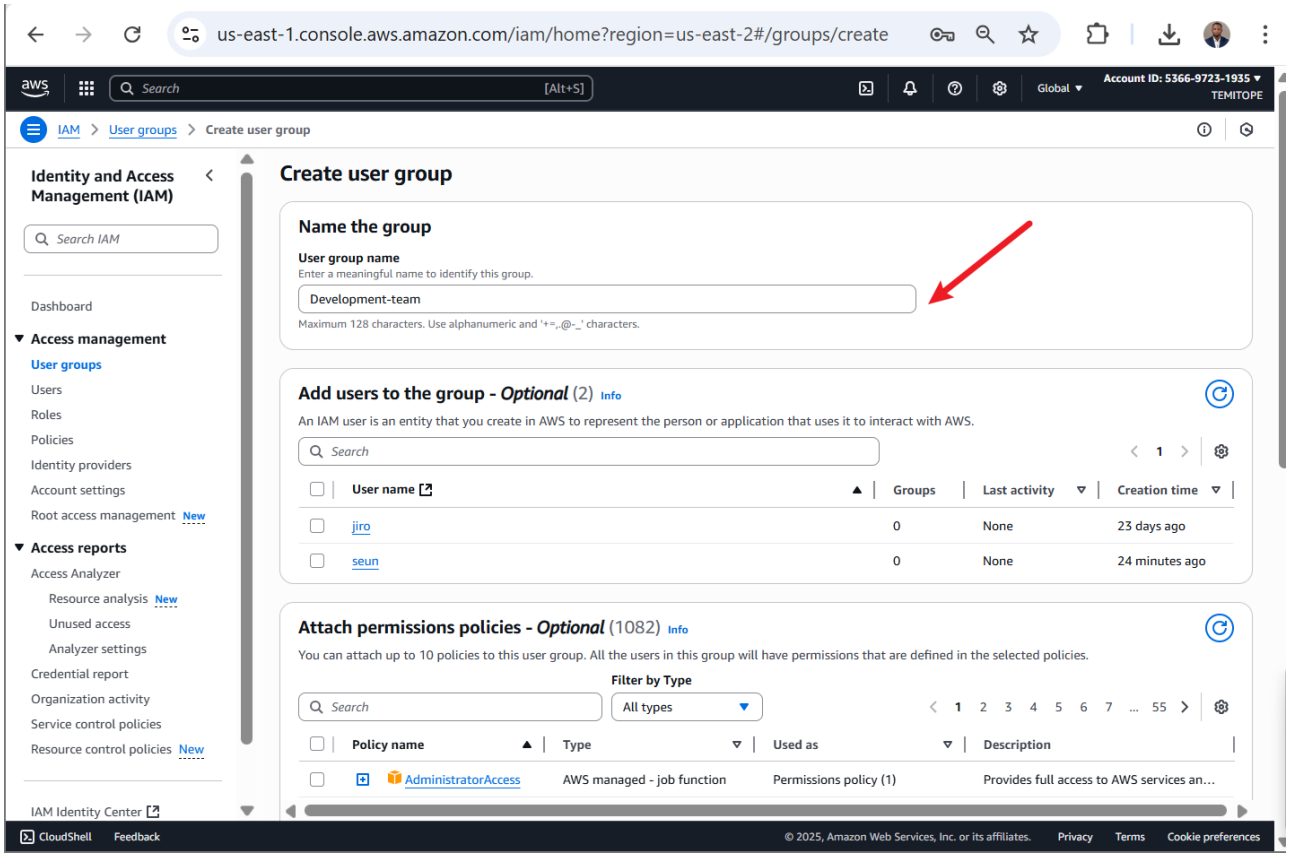
20. **Created User Listed, Now Click on User Group**



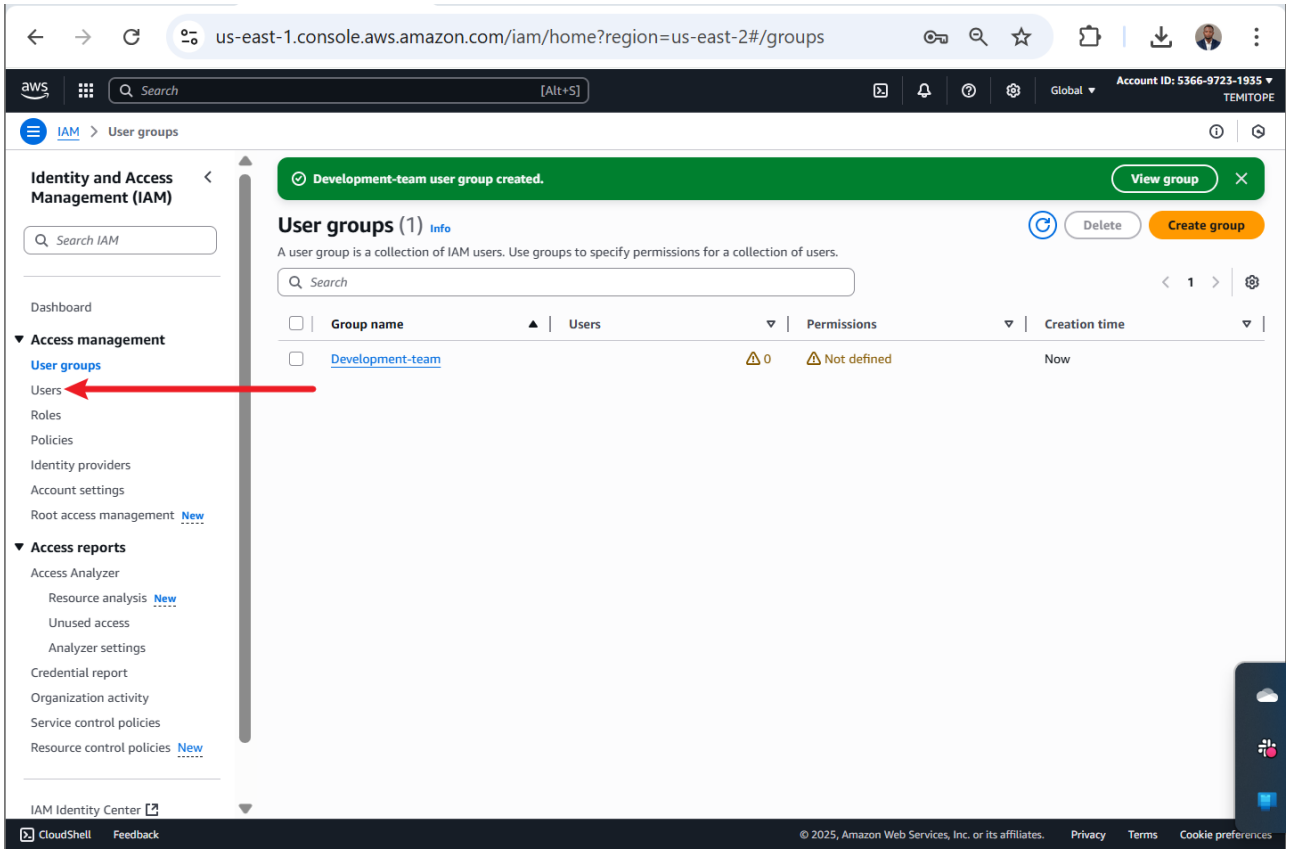
21. **On User Group Page, Click on Create Group**



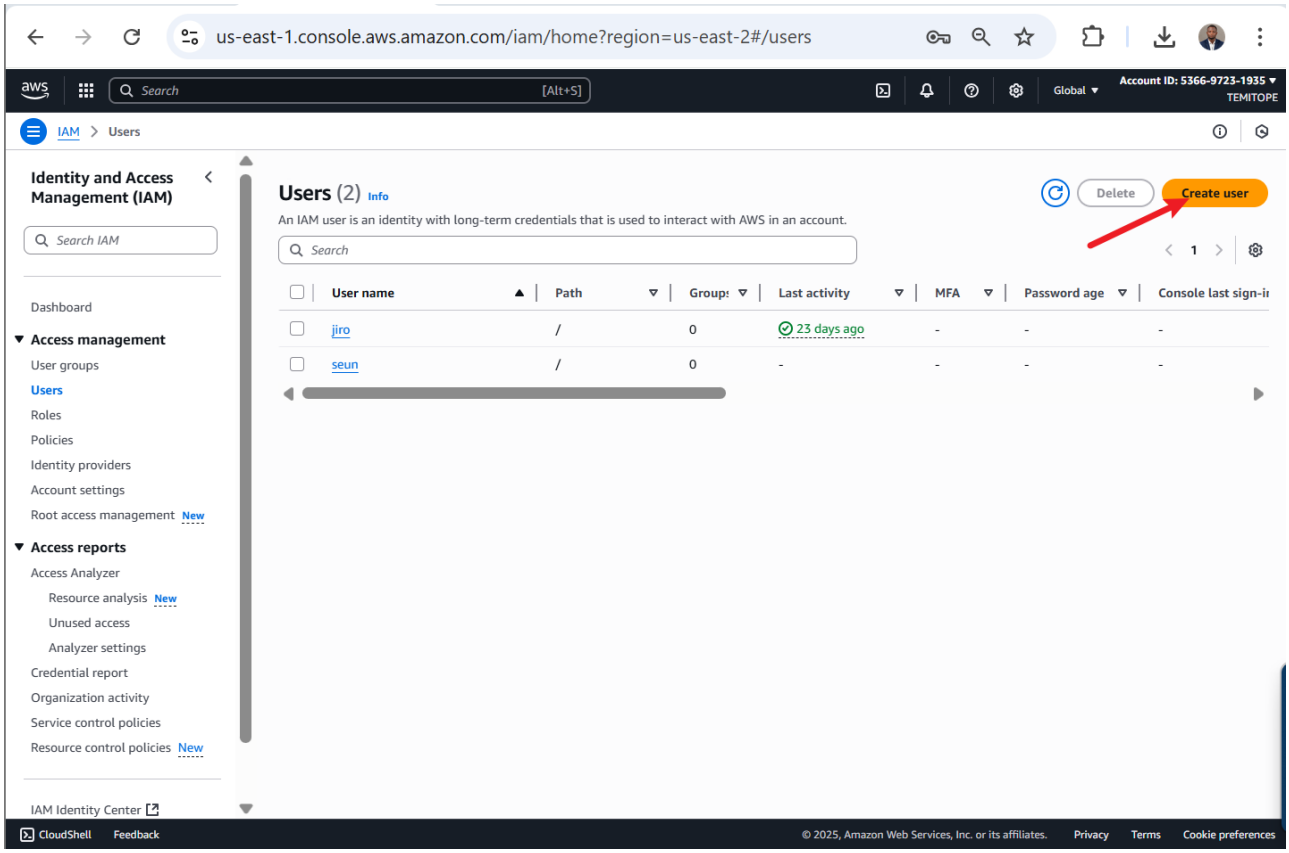
22. **Set Group Name and Scroll Down and Click Create Group**



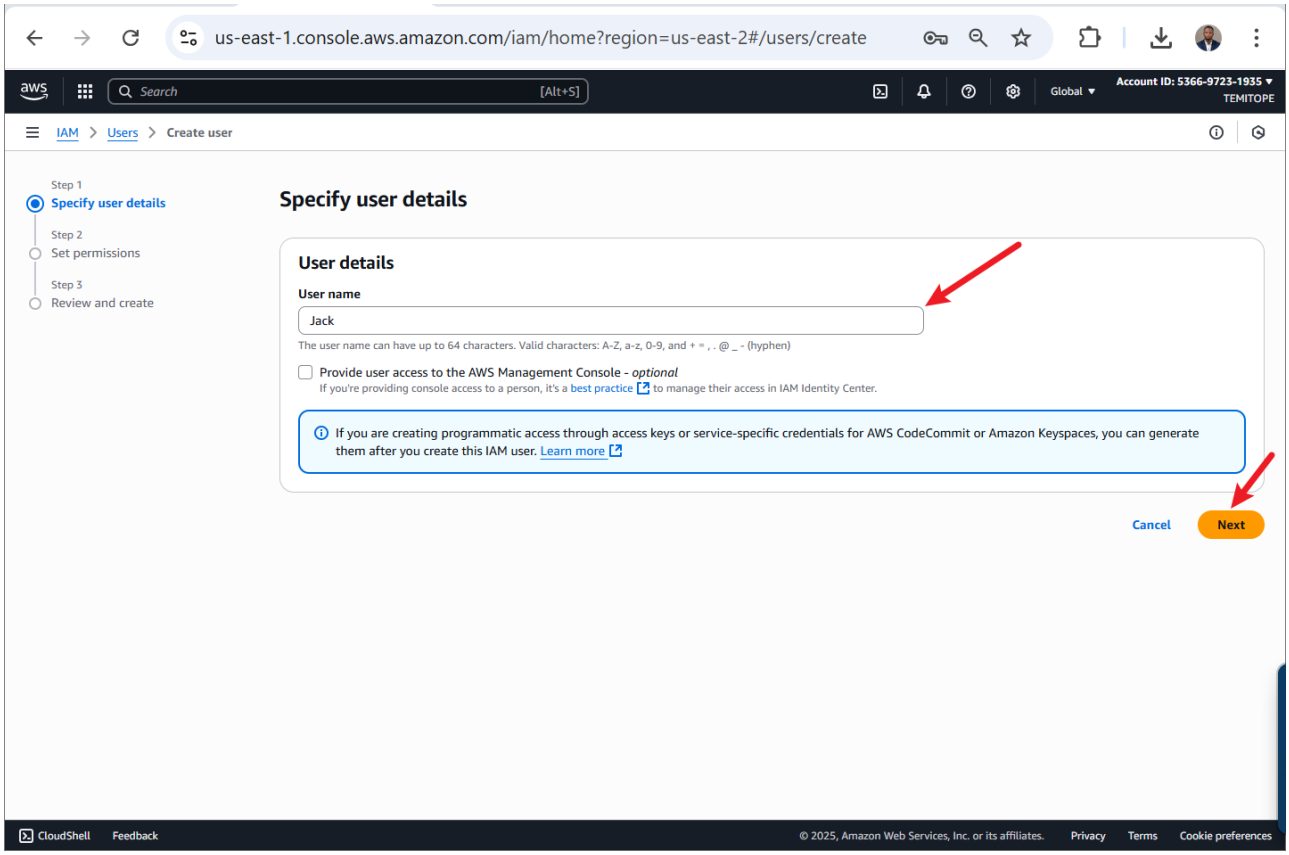
23. **Group Successfully Created, Click Users to Create New User**



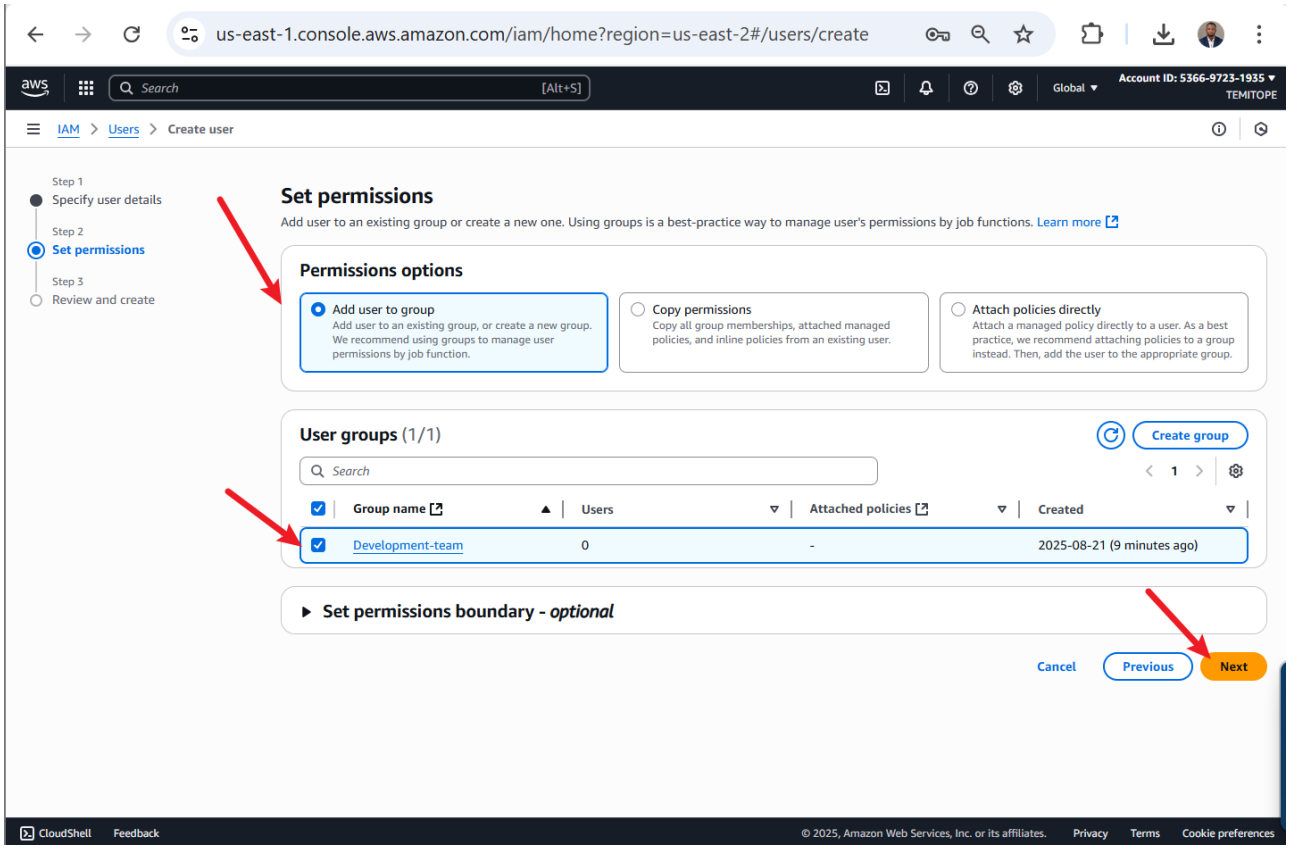
24. **User List Page Appears, Click Create User**



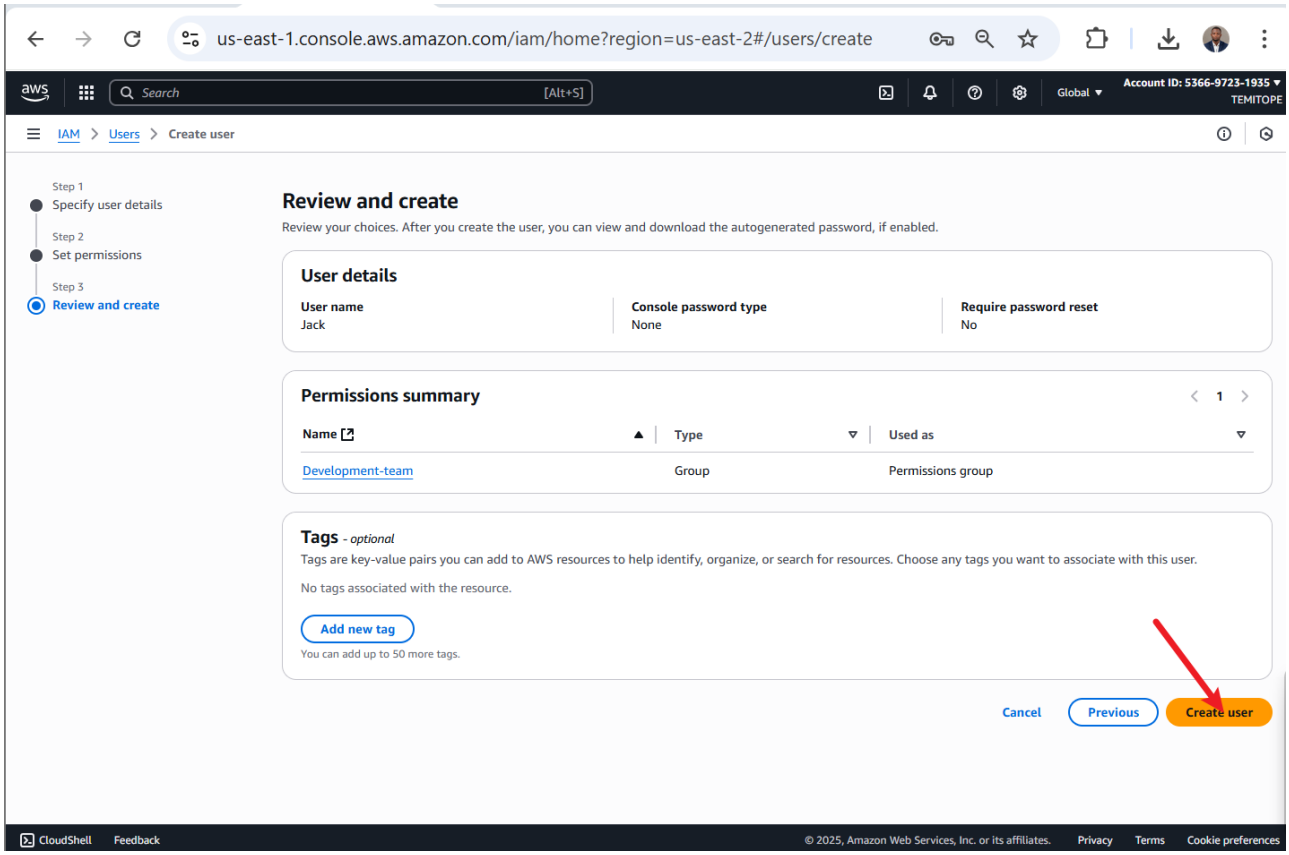
25. **Choose User Name and Click Next**



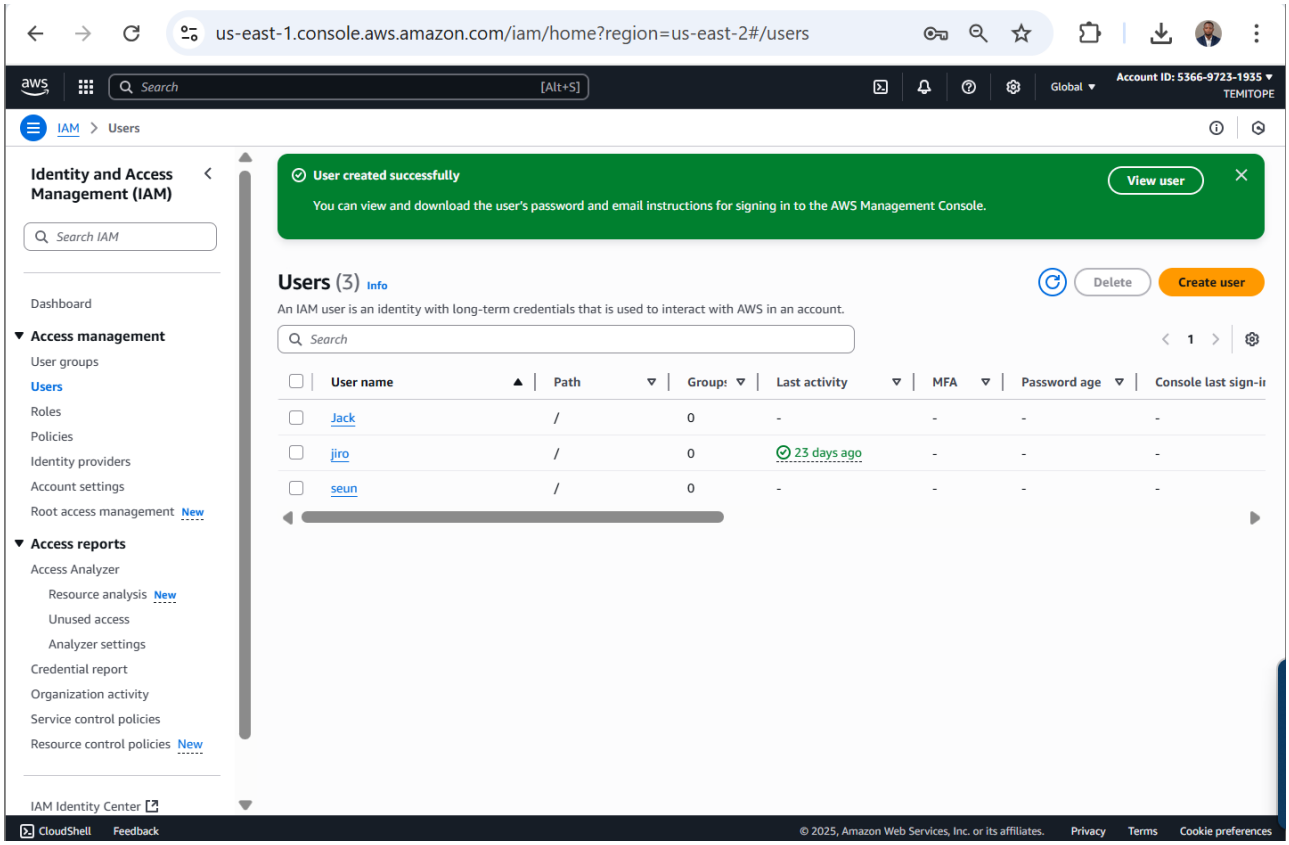
26. **In the Permission Option, Select Add User to Group and Select Development-Team and Click Next**



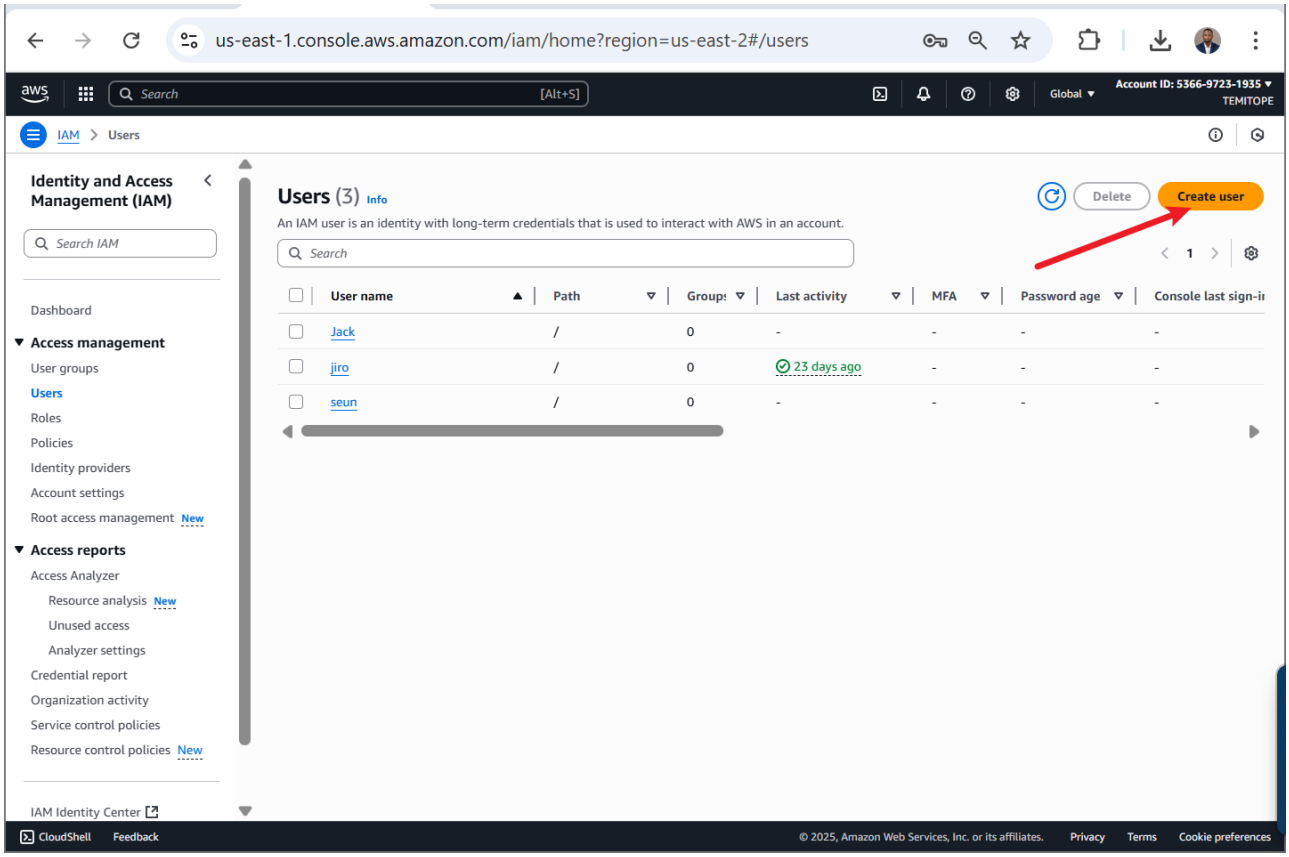
27. **Review and Click Create User**



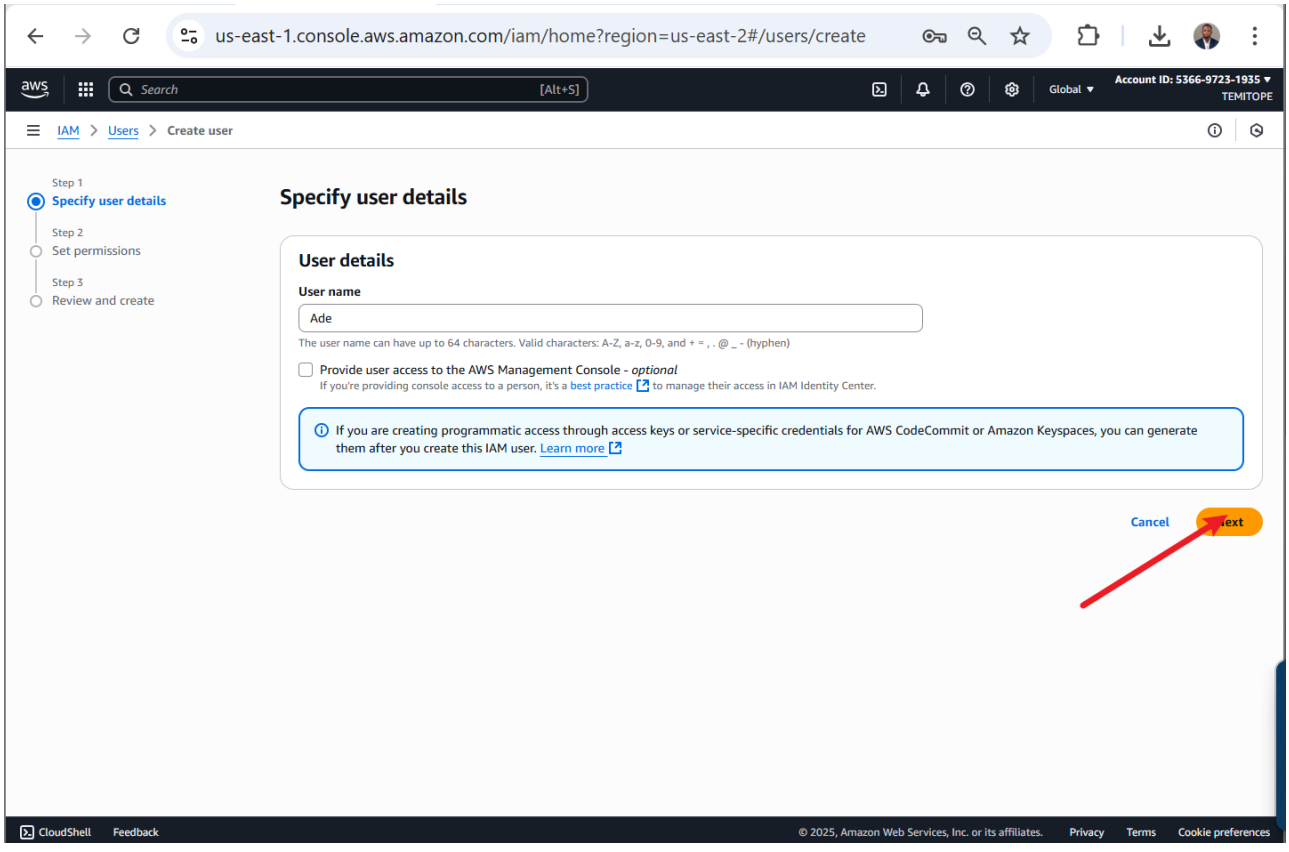
28. **User Jack Successfully Created**



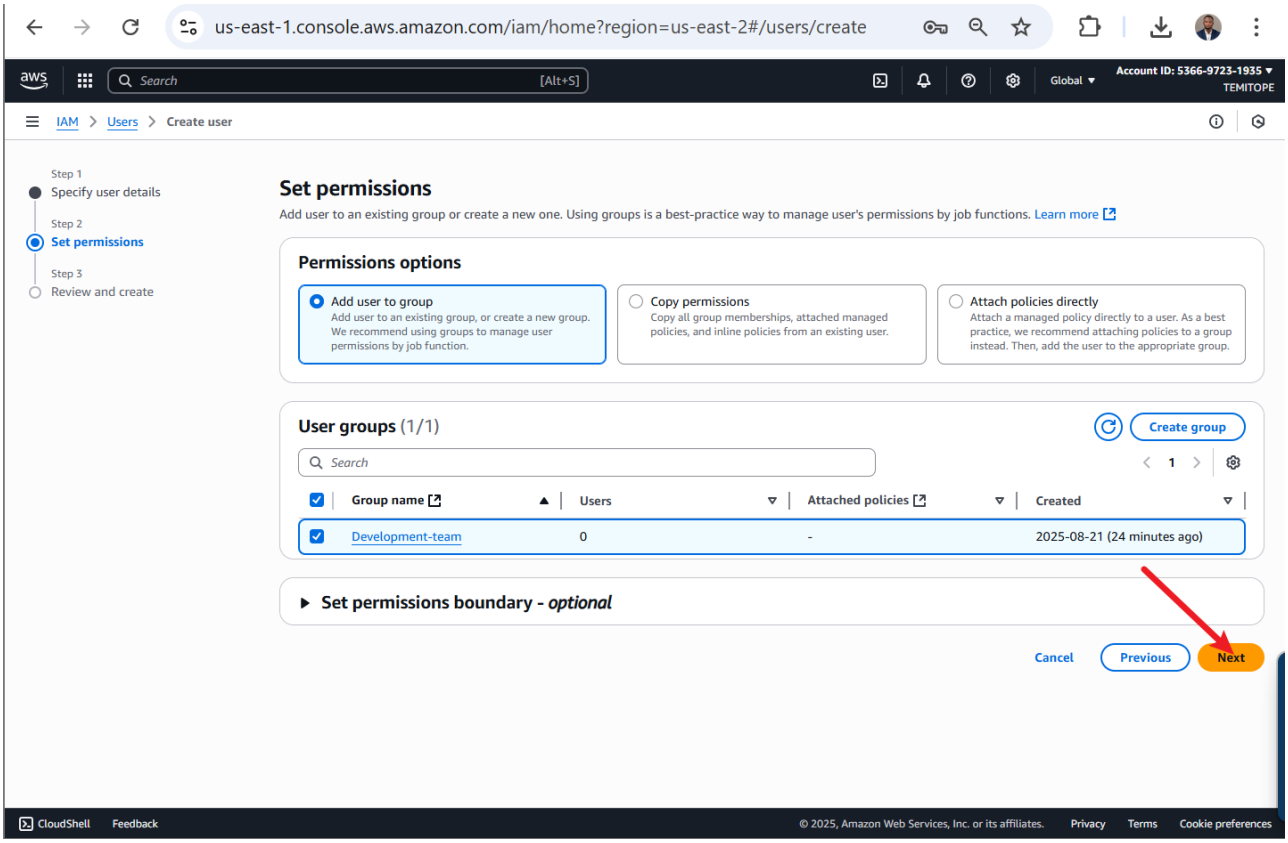
29. **Create Another User by Clicking Create User**



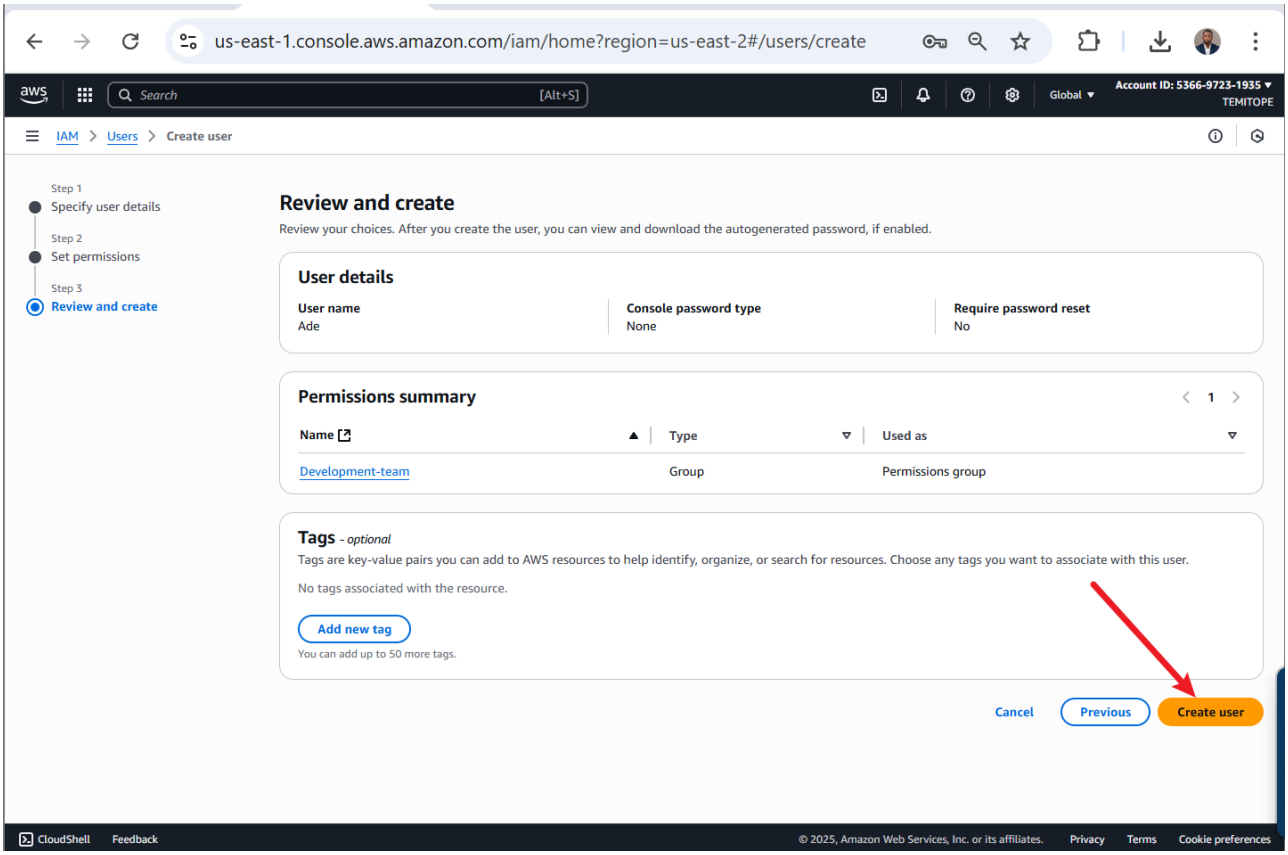
30. **Choose User Name and Click Next**



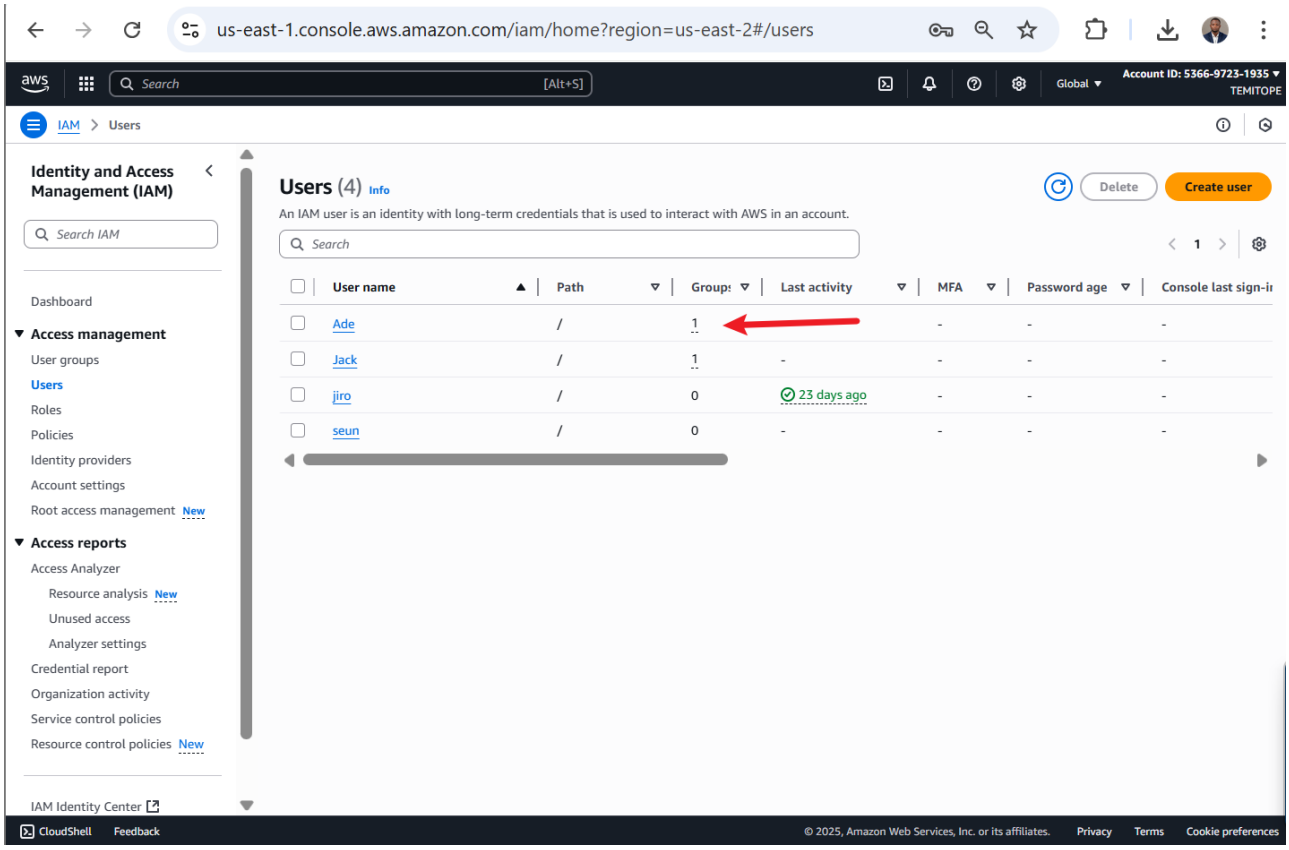
31. **In the Permission Option, Select Add User to Group and Select Development-Team and Click Next**



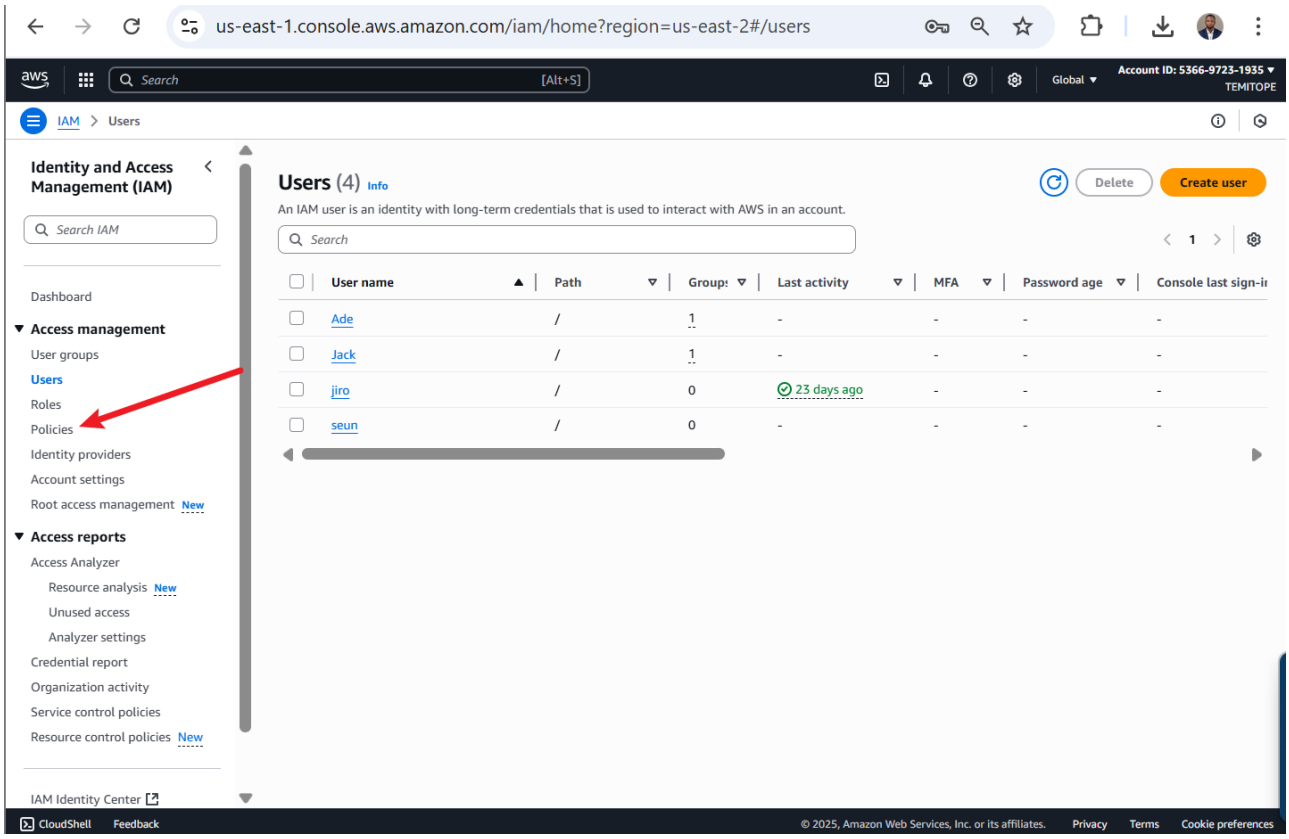
32. **Review and Click Create User**



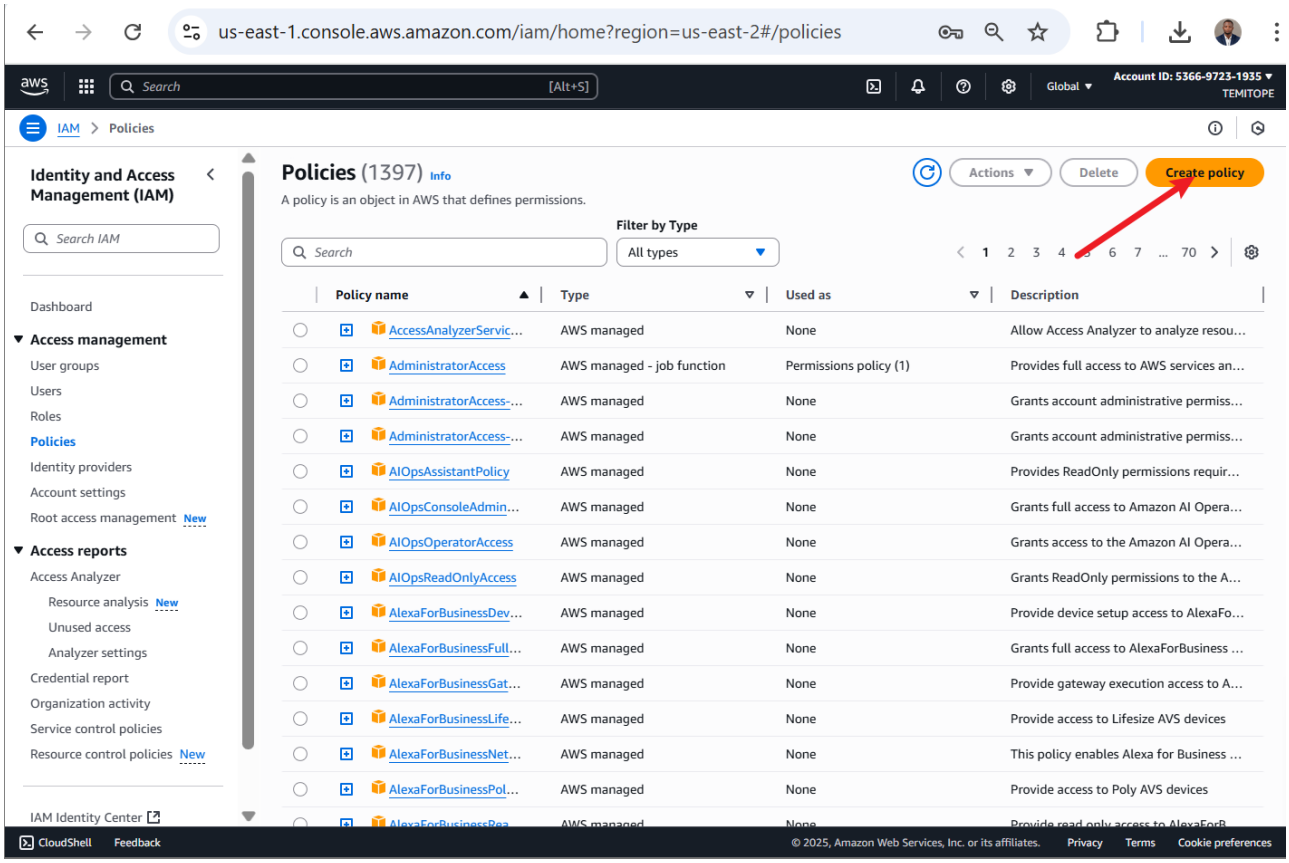
33. **User Ade Successfully Created**



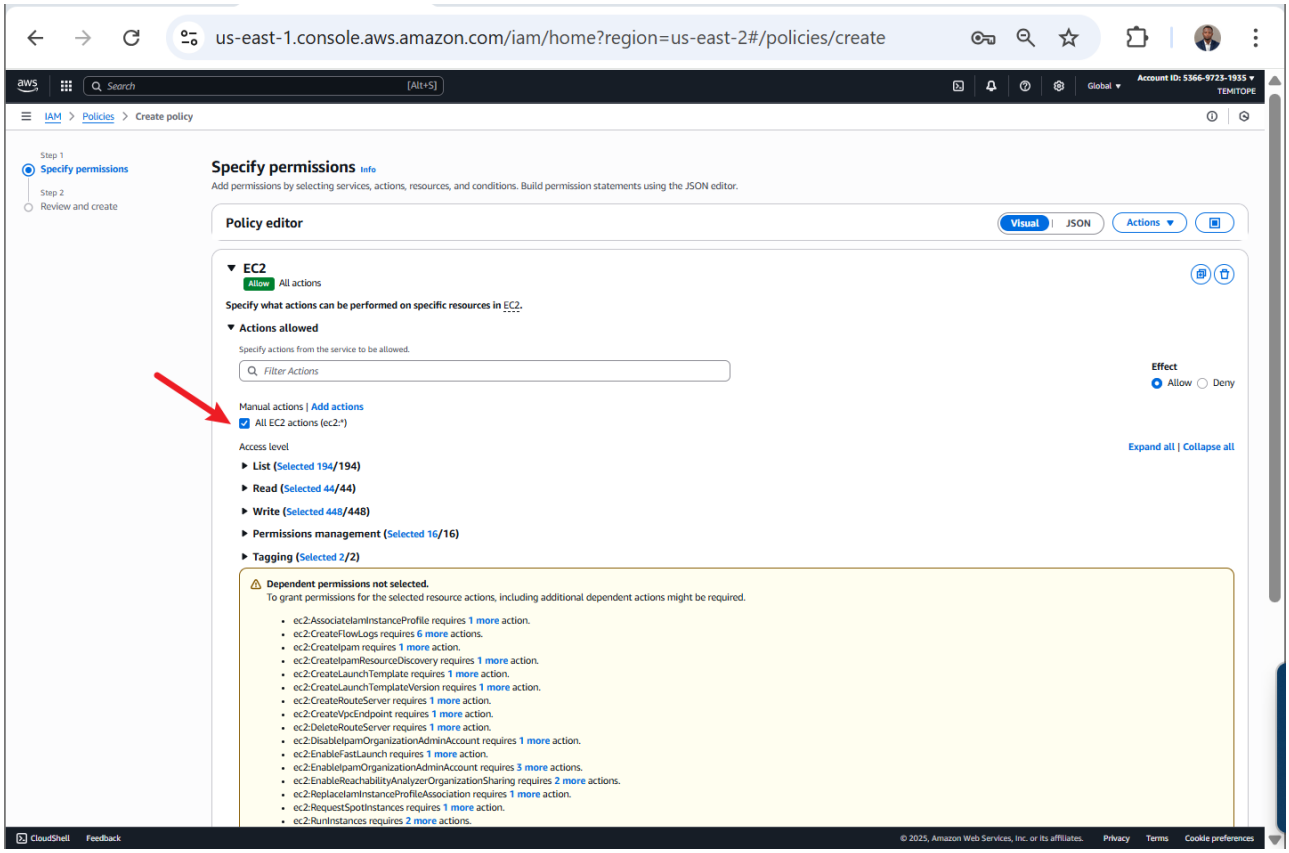
34. **Click Policies to Create New Policy for the Group**



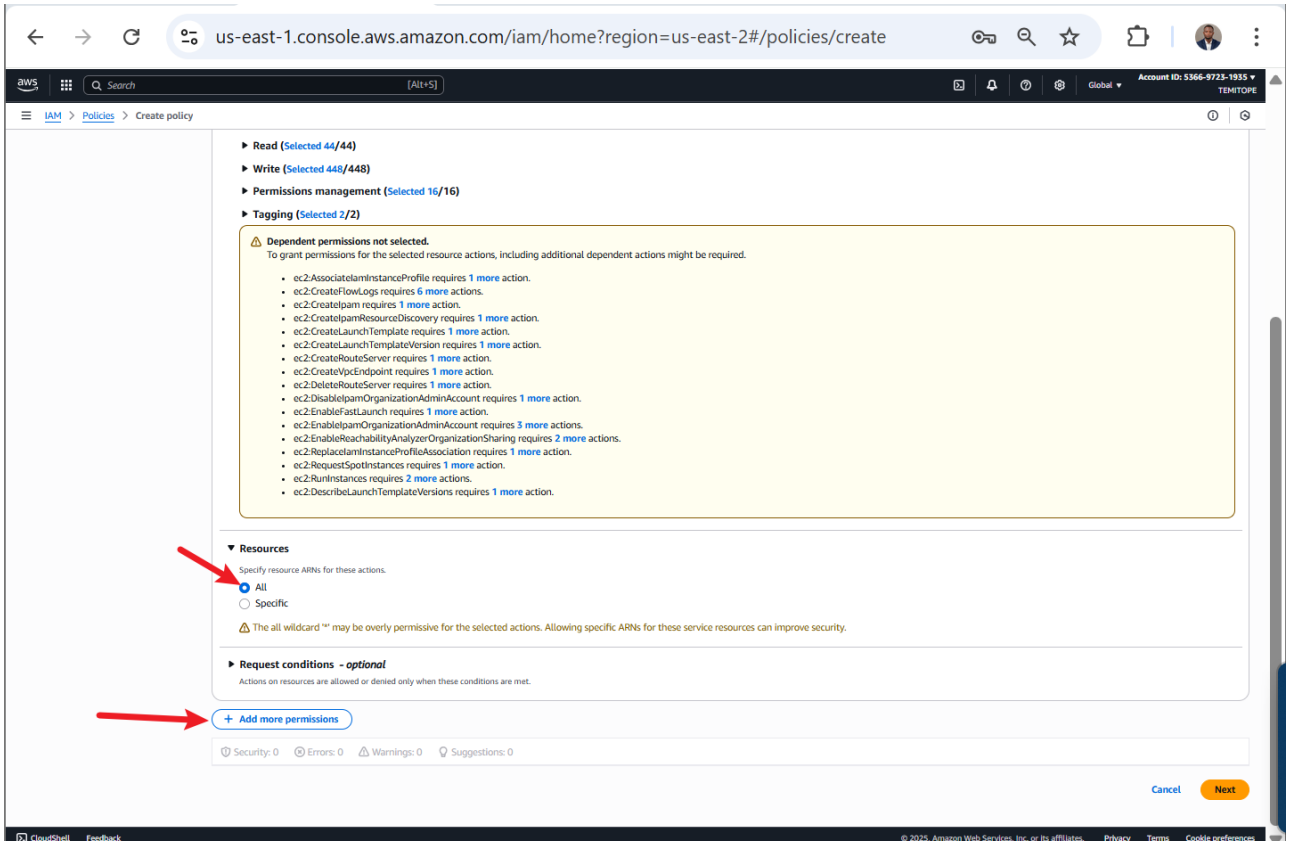
35. **Click Create Policy**



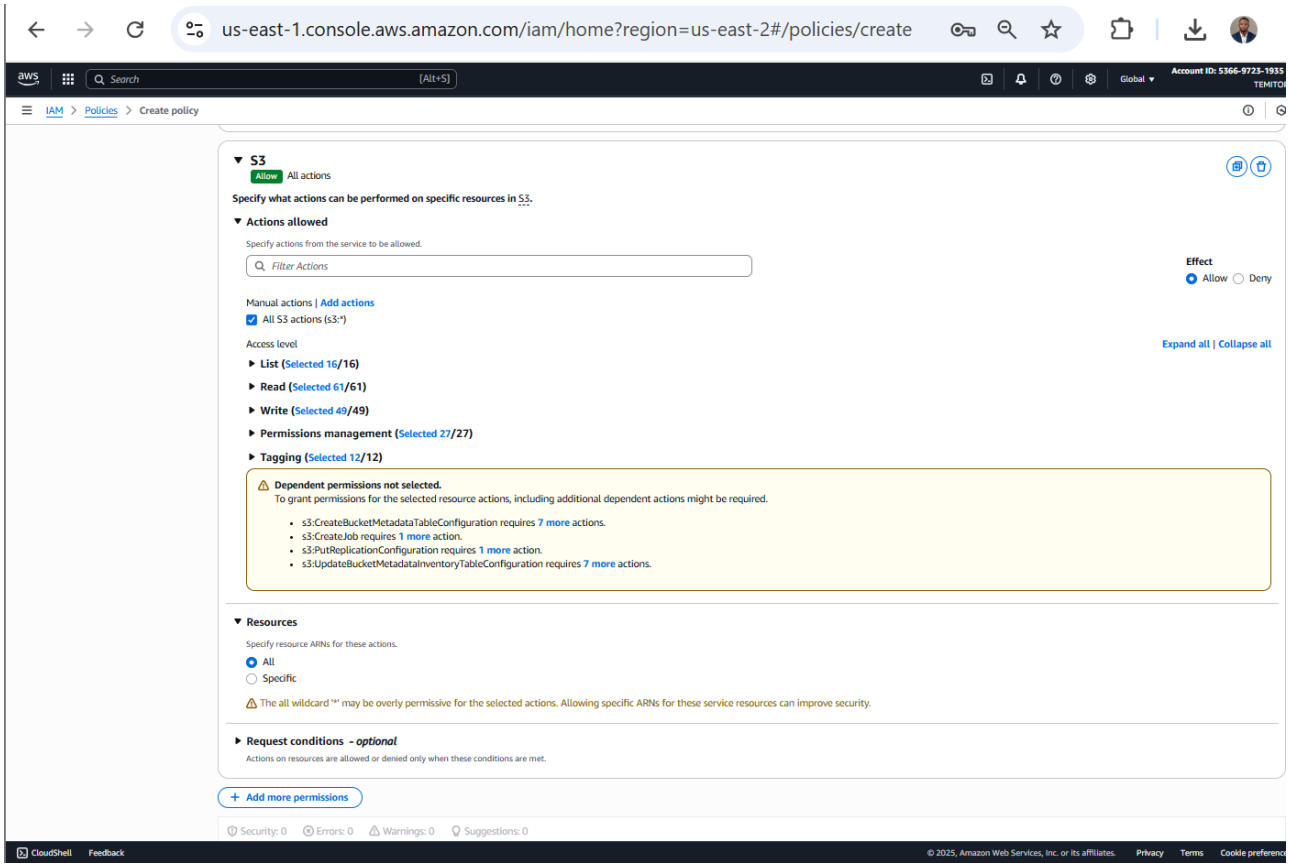
36. **Choose EC2 and Select All Actions and Resources**



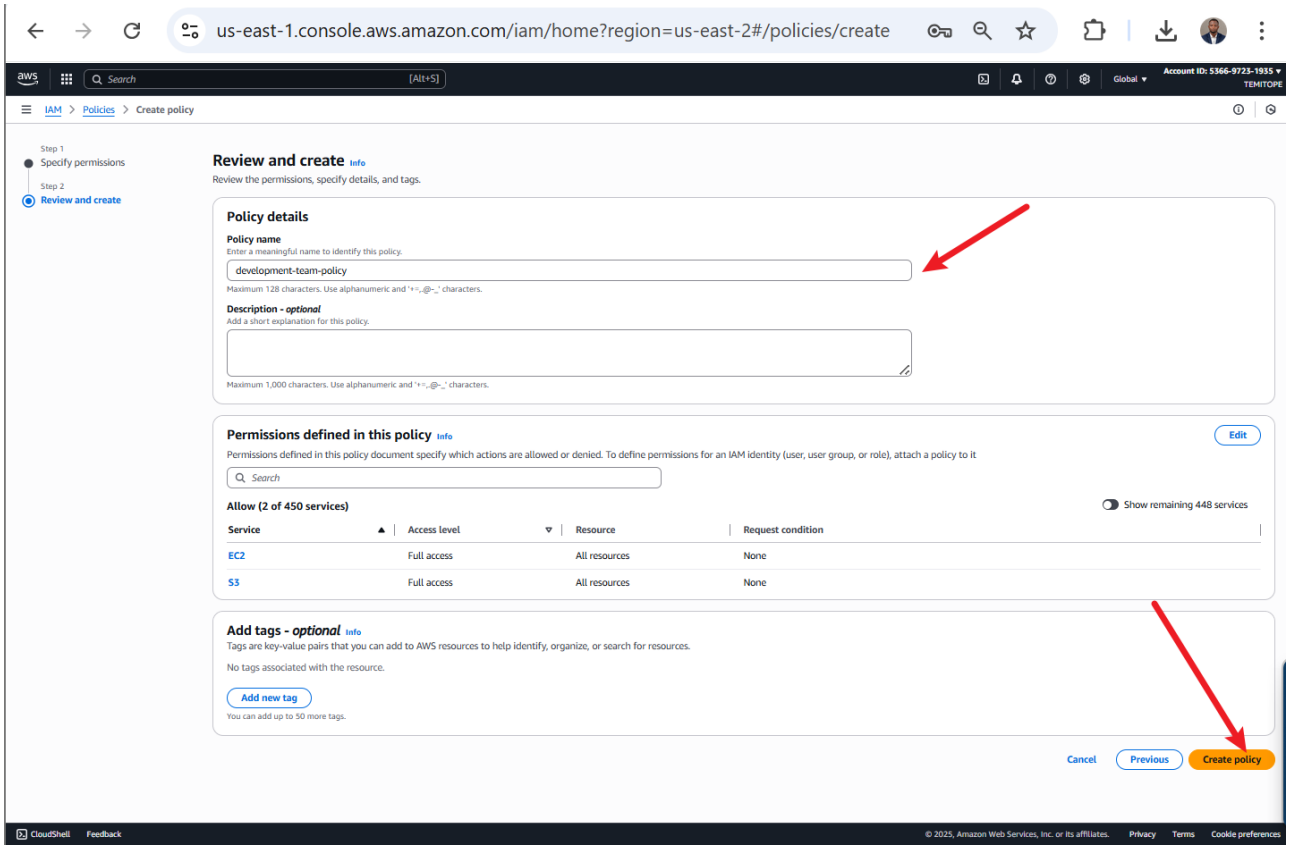
37. **Select Add More Permission and Select S3 Service**



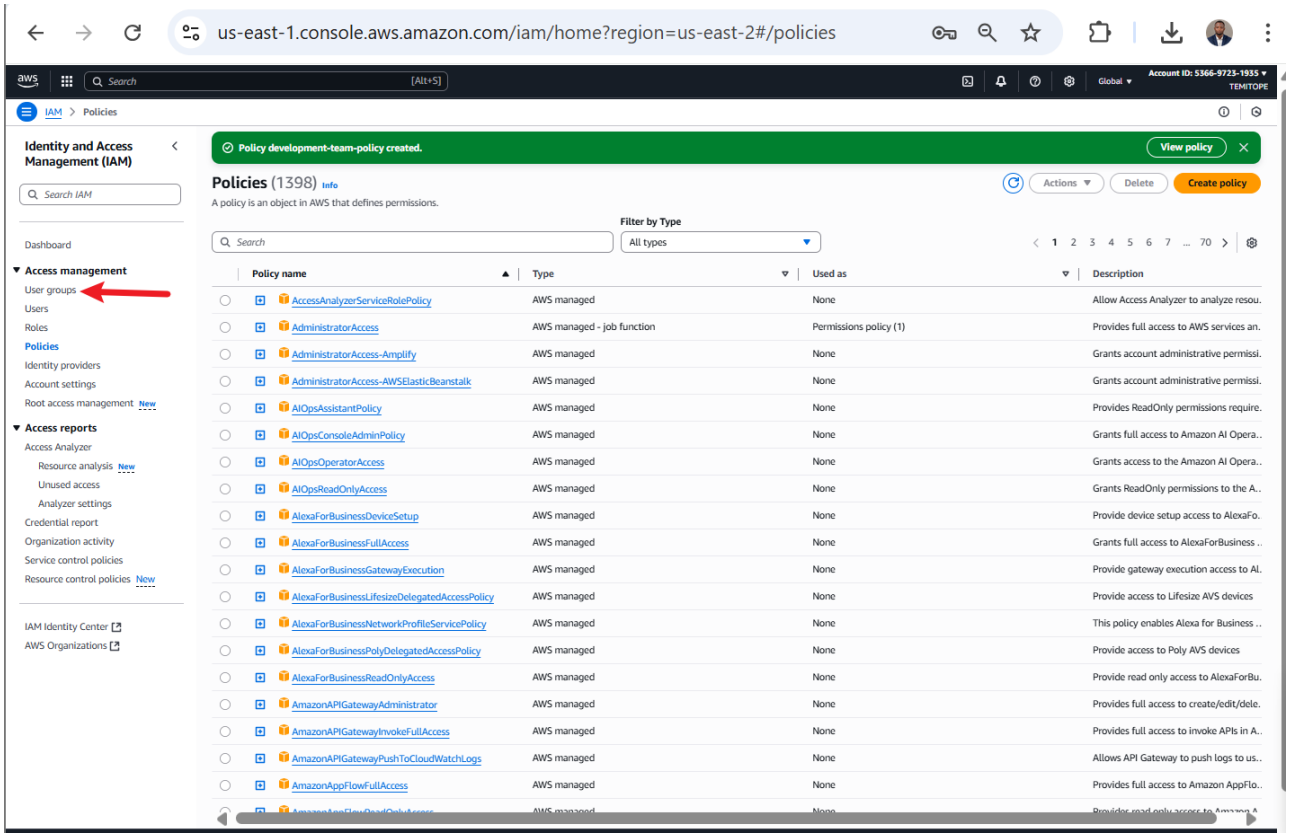
38. **Choose S3 and Select All Actions and All Resources and Click Next**



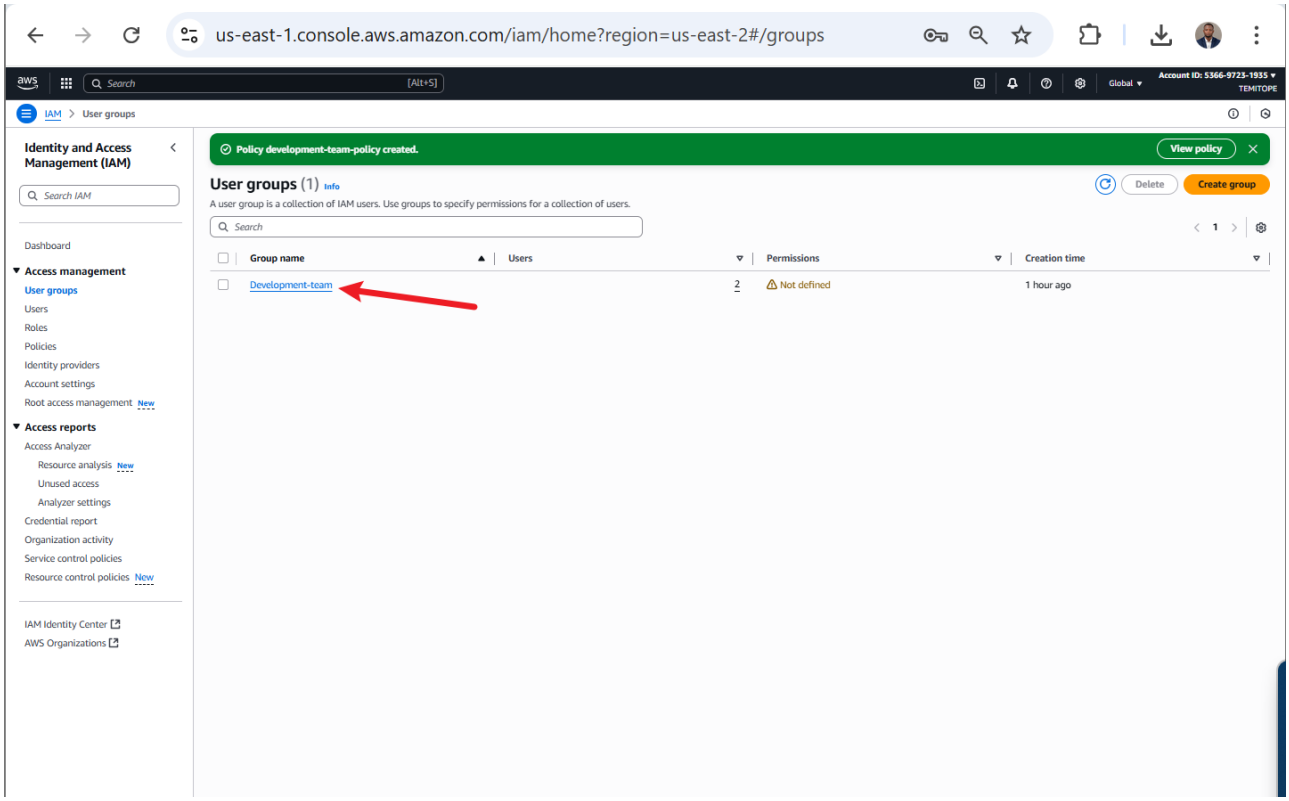
39. **Name Policy and Click Create Policy**



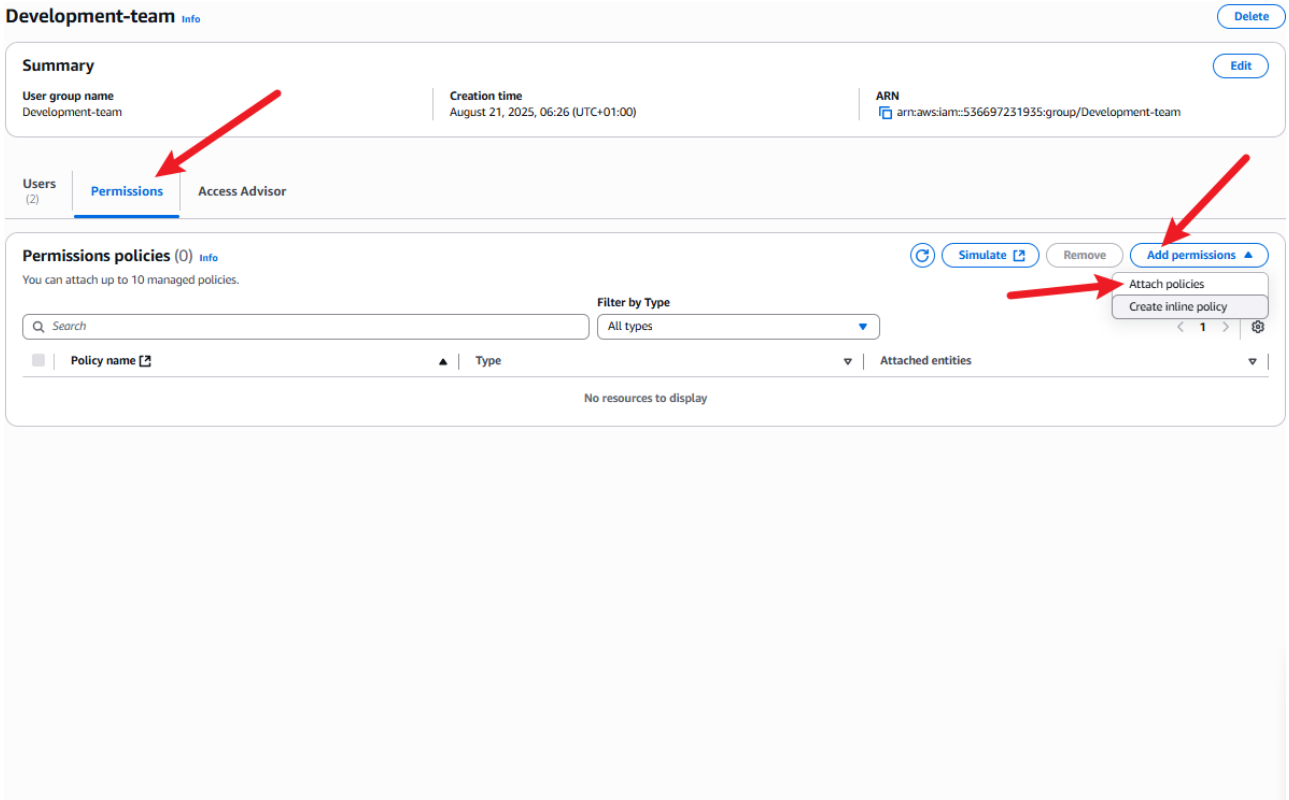
40. **Development-Team-Policy Successfully Created, Navigate to User Group Page**



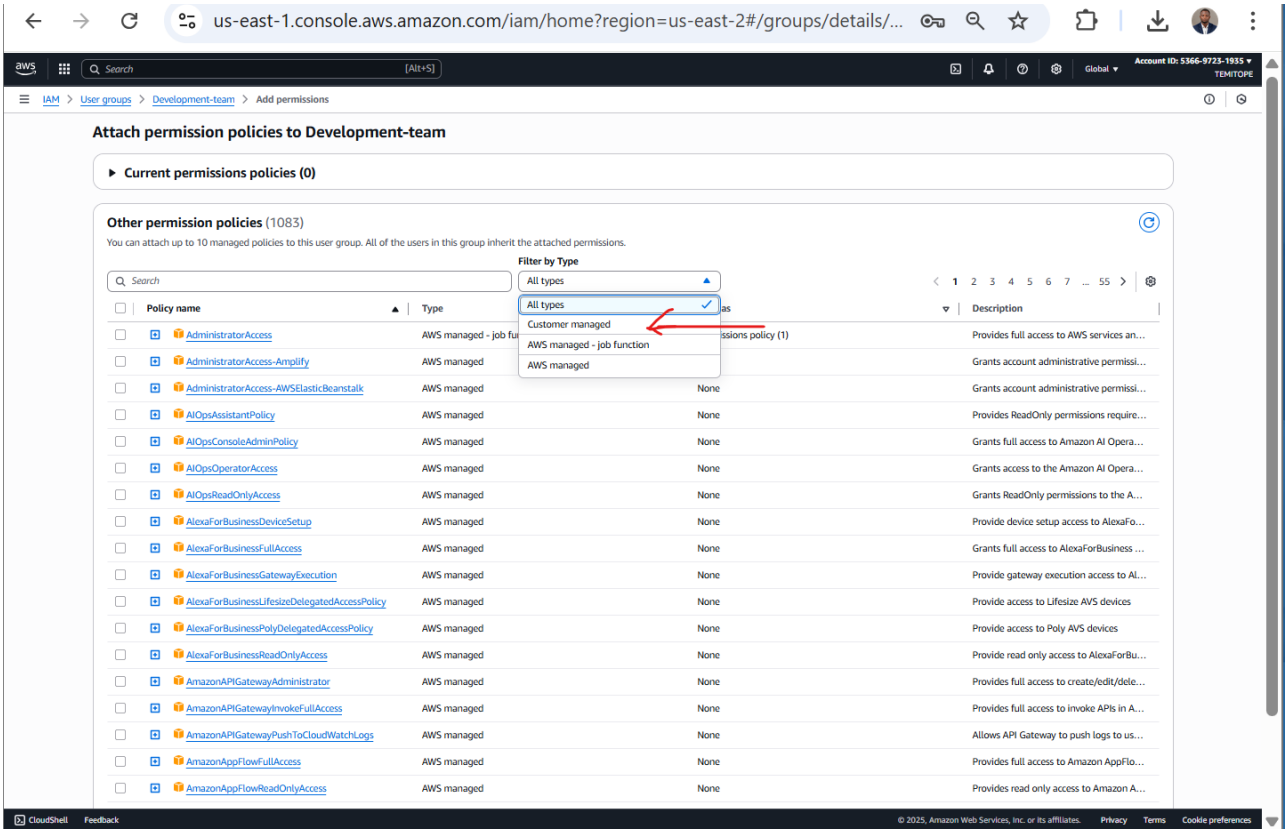
41. **Click Development-Team**



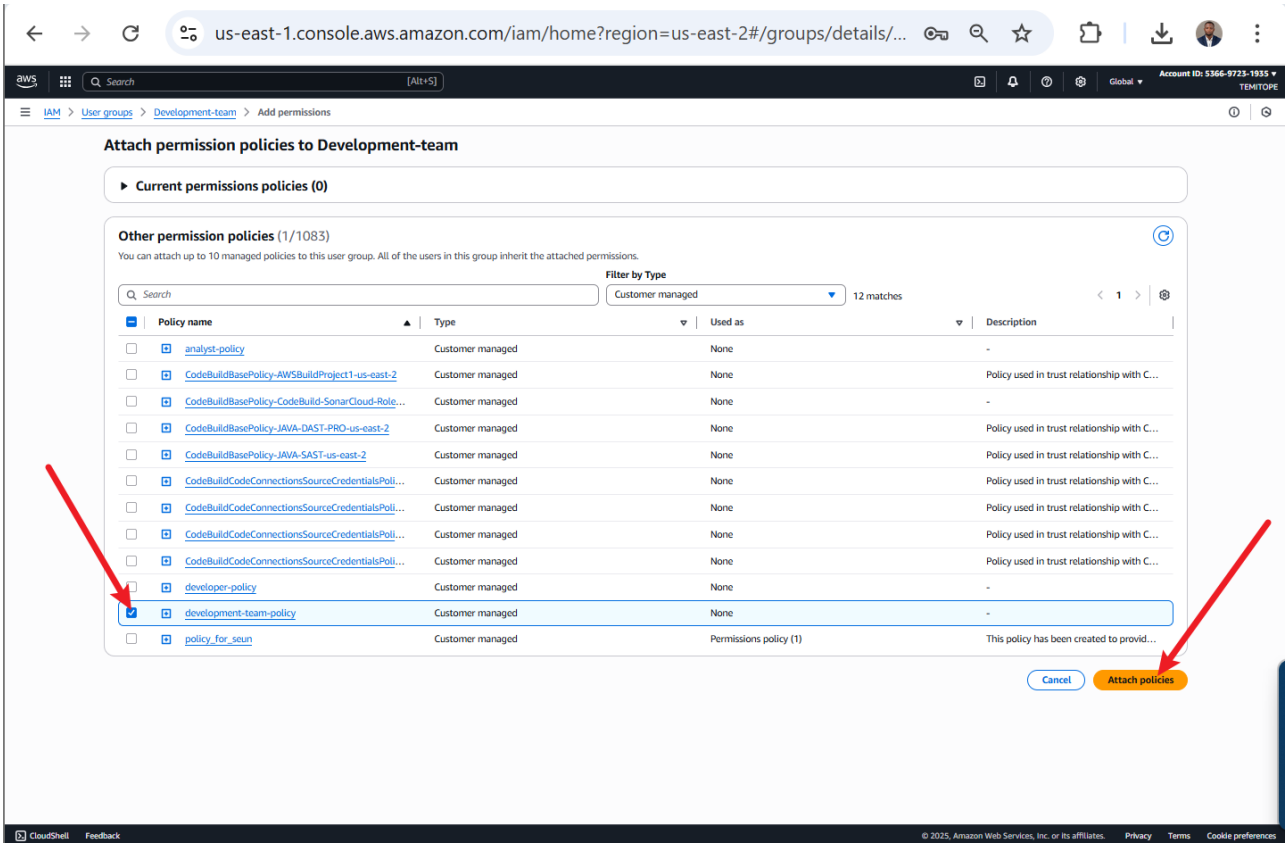
42. **Click on Permission Tab, Then Add Permission Button and Attach Policy**



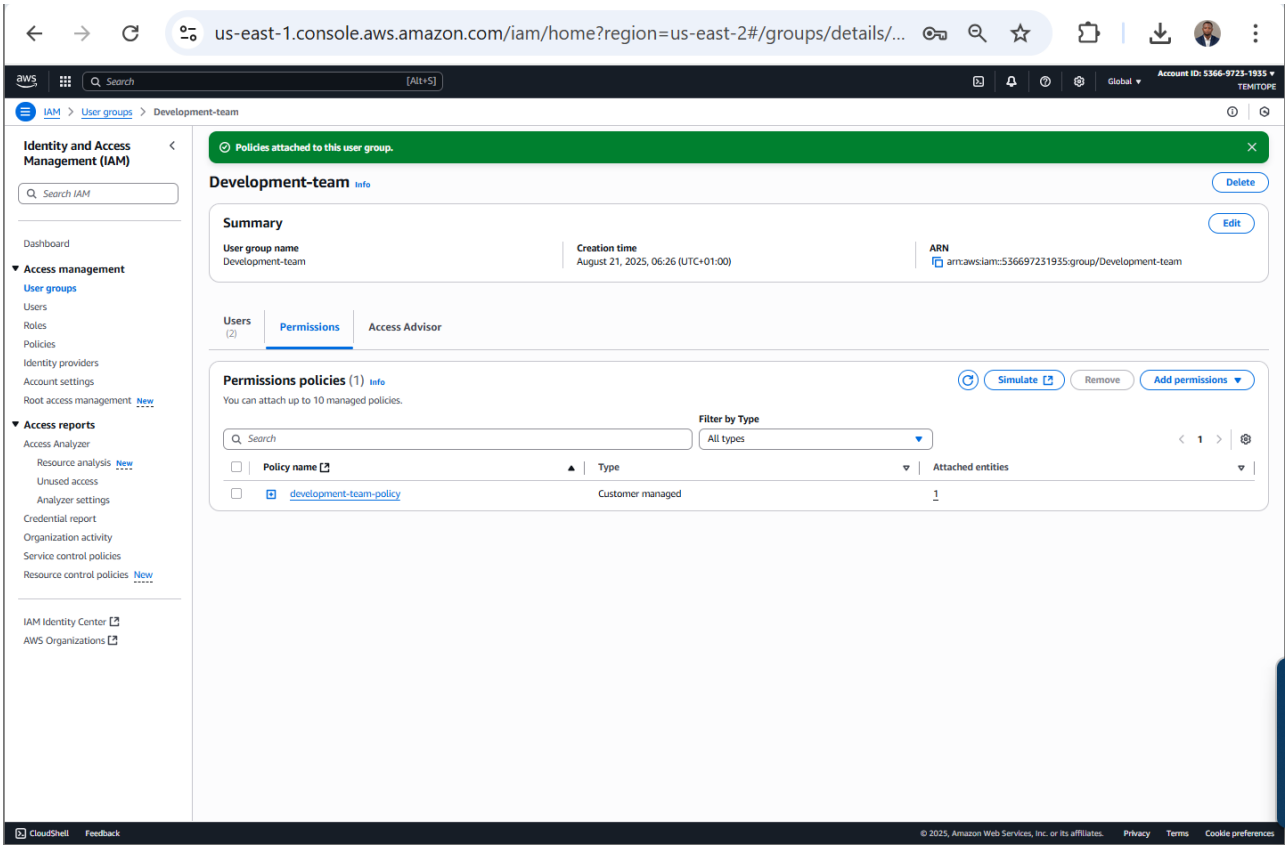
43. **On Policy Name List, Filter by Customer Managed**



44. **Select Development-Team-Policy and Click Attach Policy**

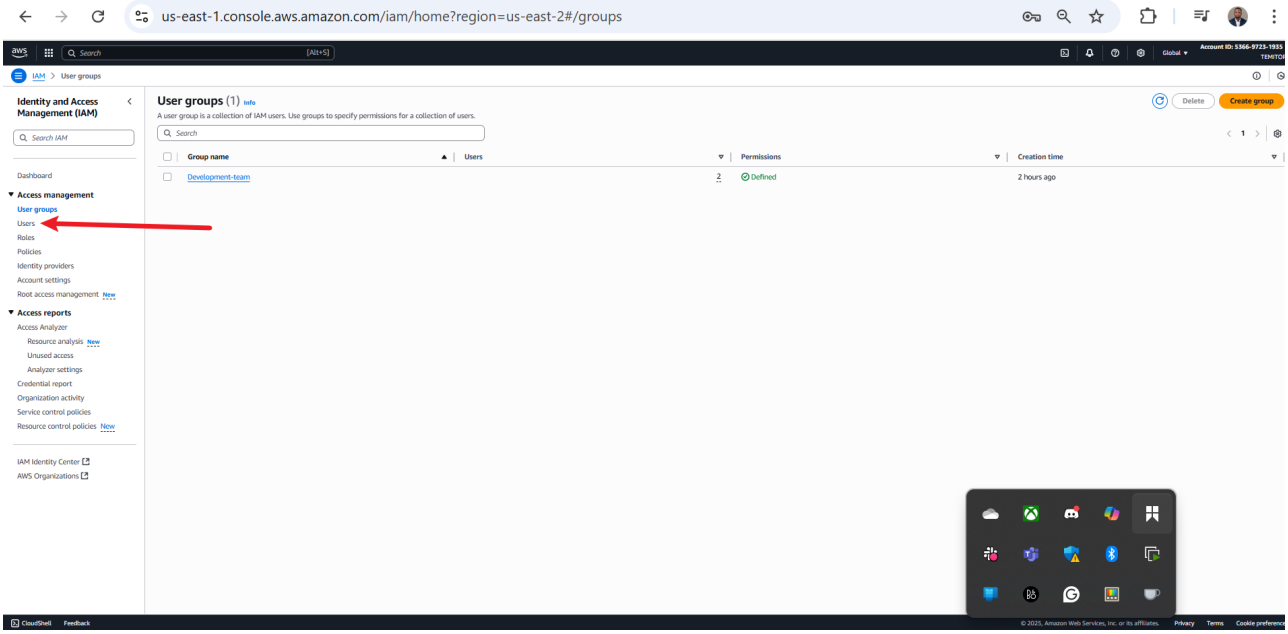


45. **Policy Successfully Attached**

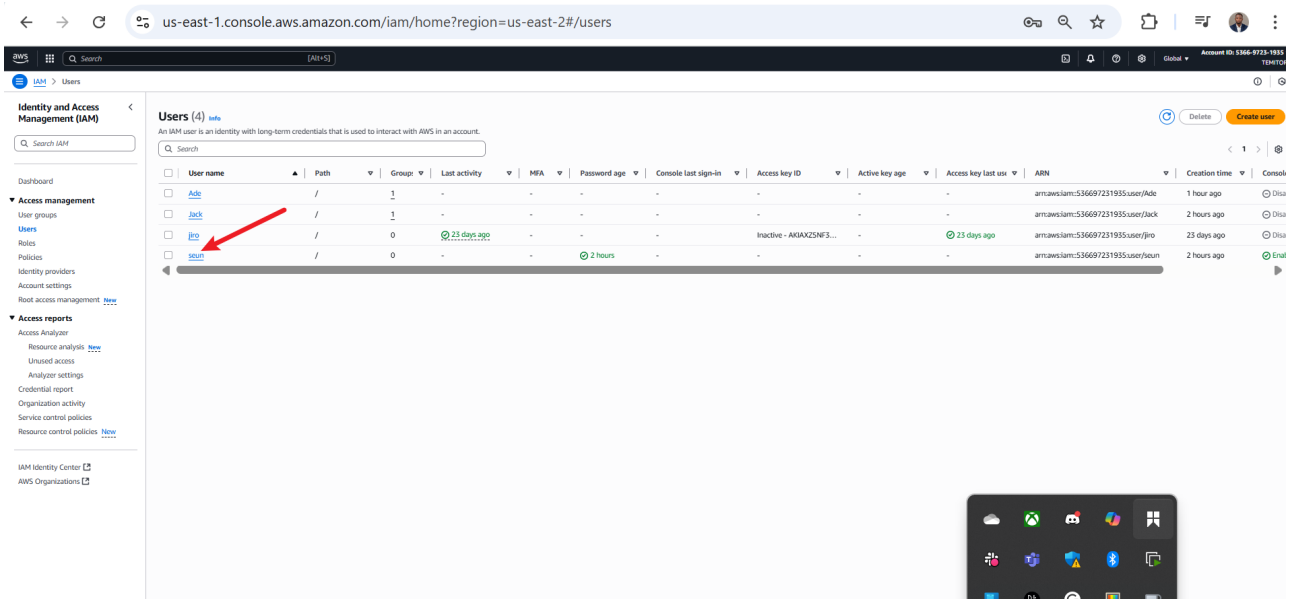


Part 3: Enable MFA for User Seun

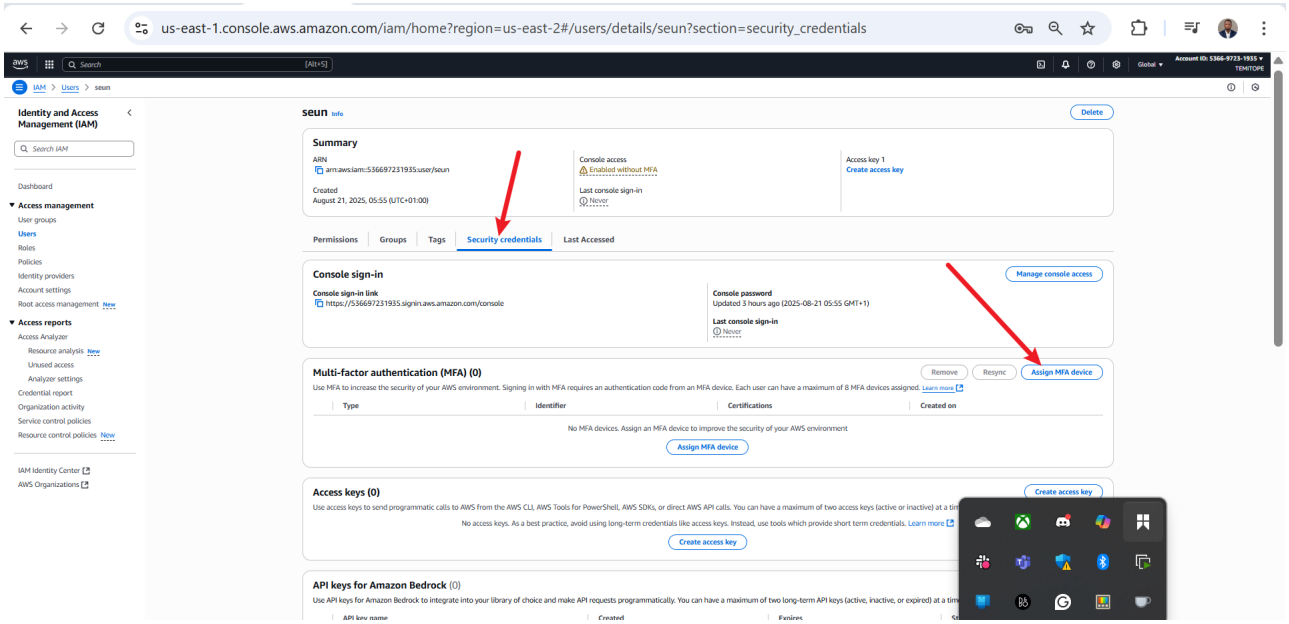
46. **Click Users to Manage User**



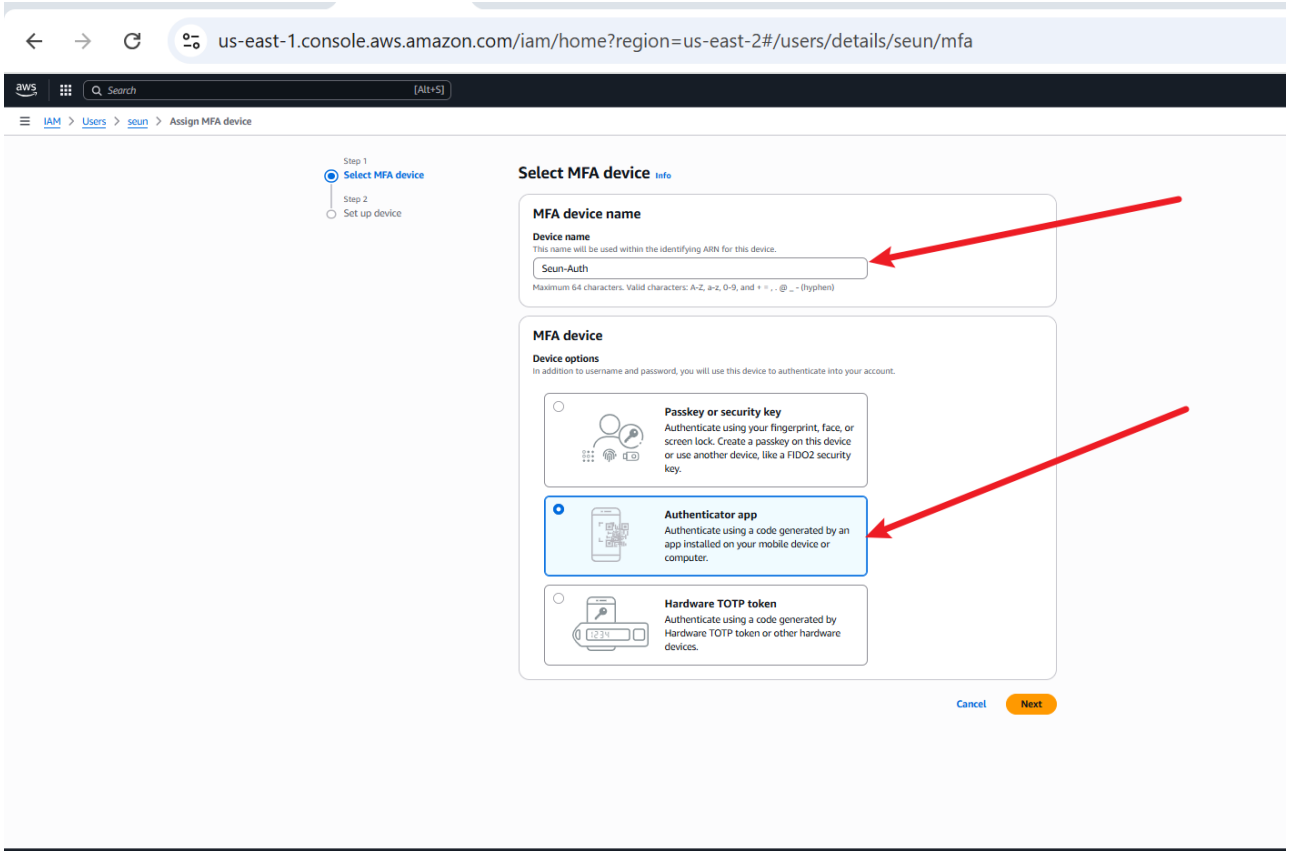
47. **Click on Seun**



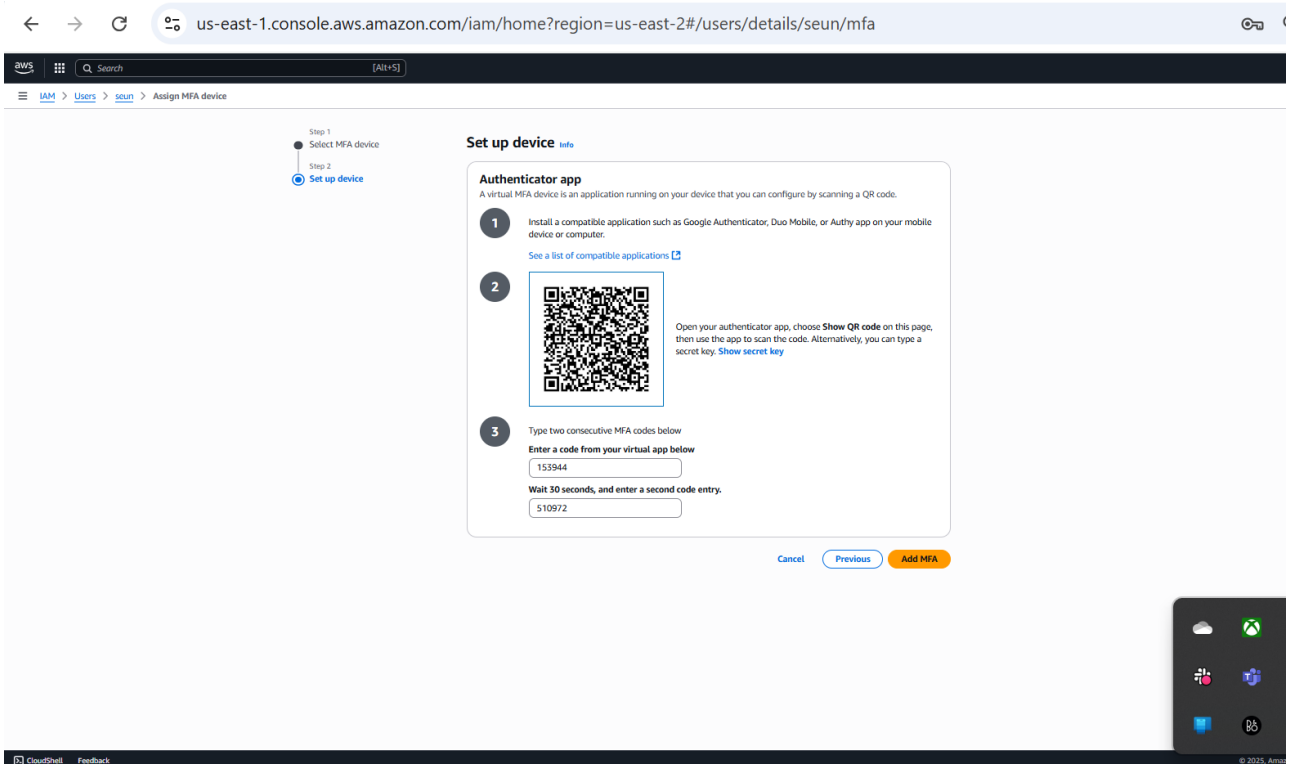
48. **Click on Security Credentials and Assign MFA**



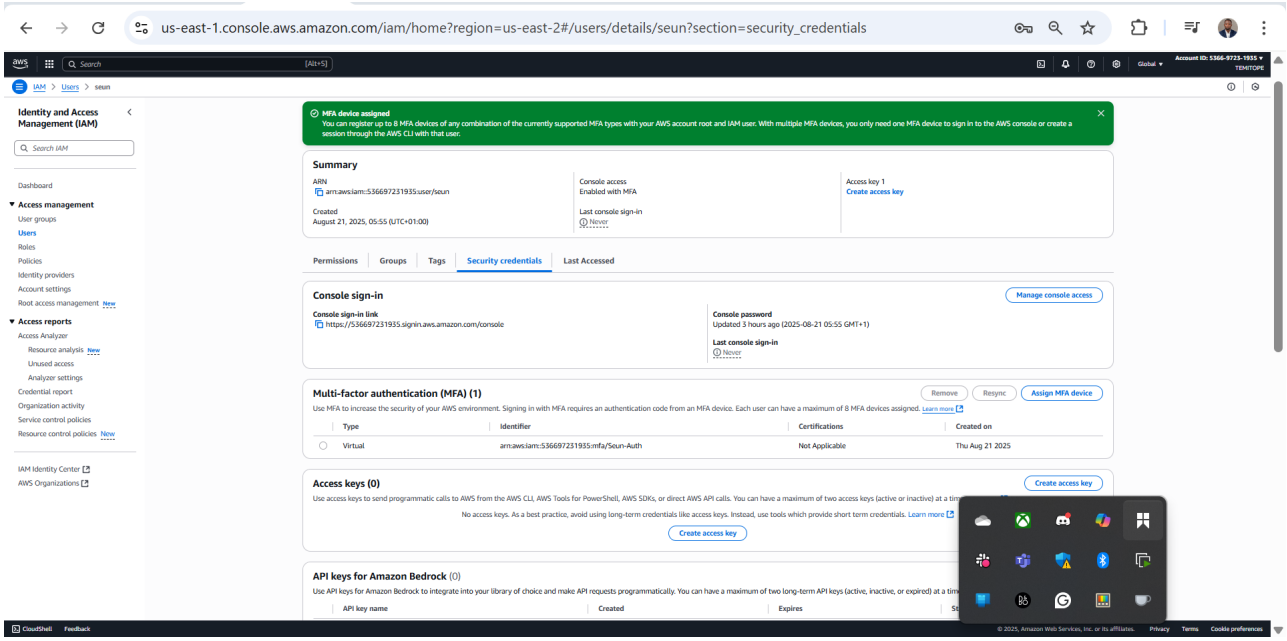
49. **Enter Device Name and Choose Authentication Option**



50. **Show QR Code, Scan QR Code, Provide Two Subsequent Codes on the Authenticator App, and Add MFA**



51. **Successfully Assigned MFA**



Project Reflection

Project Outcomes

The project achieved its goals:

- Understood IAM components (users, groups, policies).
- Created an EC2 policy for user Seun and an EC2/S3 policy for the Development-team group with users Jack and Ade.
- Enabled MFA for Seun, enhancing security.
- Applied best practices like least privilege, descriptive names, and MFA.

Challenges Faced

- Navigating the AWS Console was initially complex; the search bar and sidebar were key.
- Configuring policies required balancing granularity and access.
- Differentiating user vs. group permissions was initially unclear.
- Setting up MFA involved learning authenticator apps and QR code scanning.

Lessons Learned

- Granular policies enhance security.
- Groups streamline permission management for teams.
- MFA significantly boosts account security.
- Clear documentation prevents errors.
- Best practices like least privilege and MFA are practical and effective.

Potential Improvements

- Enable MFA for Jack and Ade for comprehensive security.
- Include IAM role creation (e.g., for AWS services).
- Test policies using the IAM Policy Simulator.

- Ensure consistent naming in documentation.

Conclusion

The project built practical IAM skills, including policy creation, user/group management, and MFA implementation for secure AWS resource management. Challenges like console navigation, policy setup, and MFA configuration were overcome, reinforcing best practices. Adding MFA for all users, role-based exercises, and policy testing would enhance future iterations. I'm now confident in applying IAM and MFA to real-world AWS scenarios.