Practical - 4

Name: Sakshi Deshmukh
Roll no. 158
PRN: 202301040191

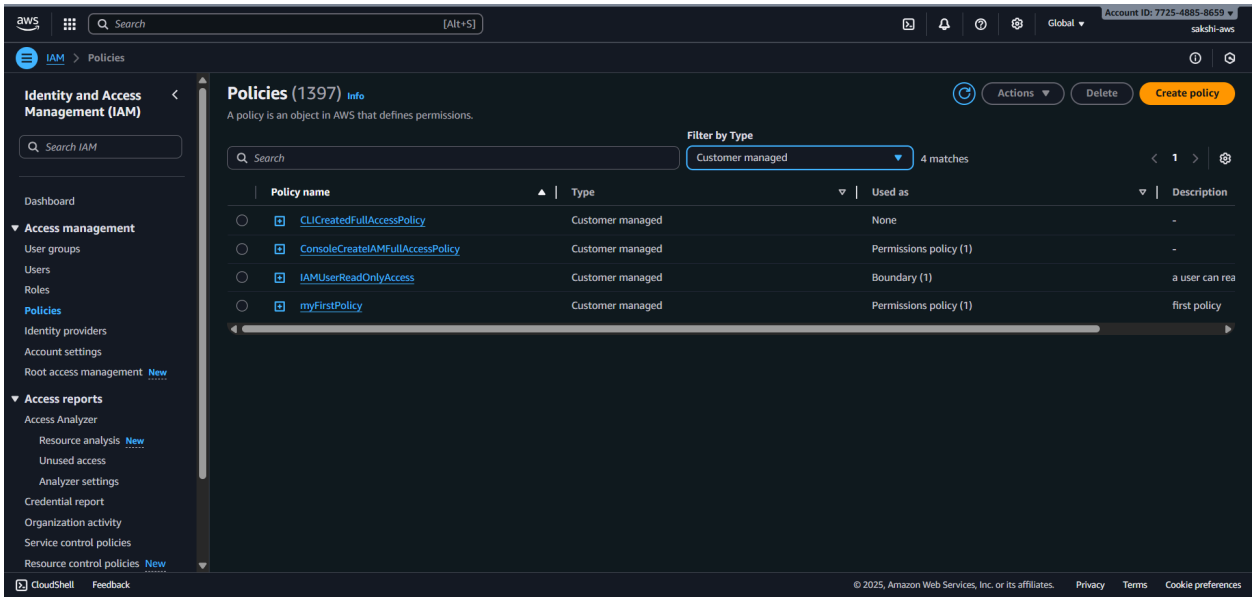
 # Permissions Boundary Configuration

In modern cloud environments, managing permissions at scale is a critical challenge. When development teams need to create resources for their applications, relying on a central security team for every IAM role creates a significant bottleneck. The AWS IAM Permissions Boundary is a feature designed to solve this problem by enabling safe delegation.

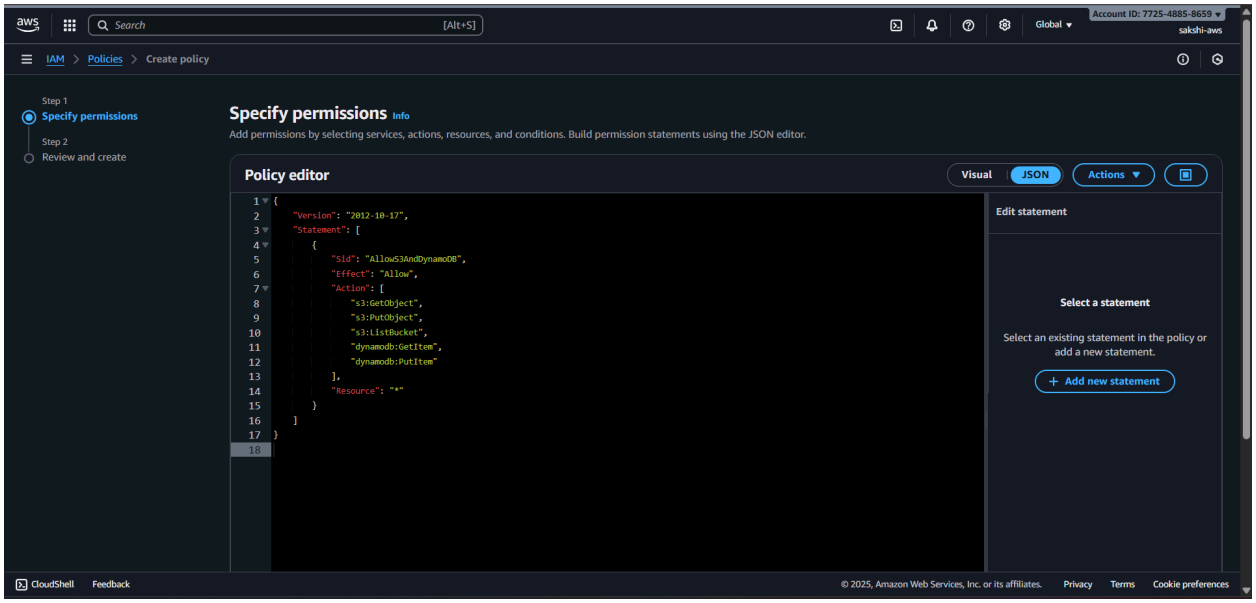
This practical demonstrates the primary use case for IAM Permissions Boundaries. We will simulate a real-world scenario where a central administrator empowers a developer to create IAM roles. The administrator will first create a "guardrail" policy (the Permissions Boundary) that defines the maximum possible permissions for any new role. They will then create a policy for the developer that allows them to create roles, but only if they attach the pre-defined boundary.

The objective is to show how the effective permissions of the new role become the intersection of the broad policy the developer attaches and the restrictive boundary set by the admin. This ensures that even if a developer tries to create a role with administrative privileges, its actual capabilities are safely limited by the boundary, thus maintaining security while enabling developer agility.

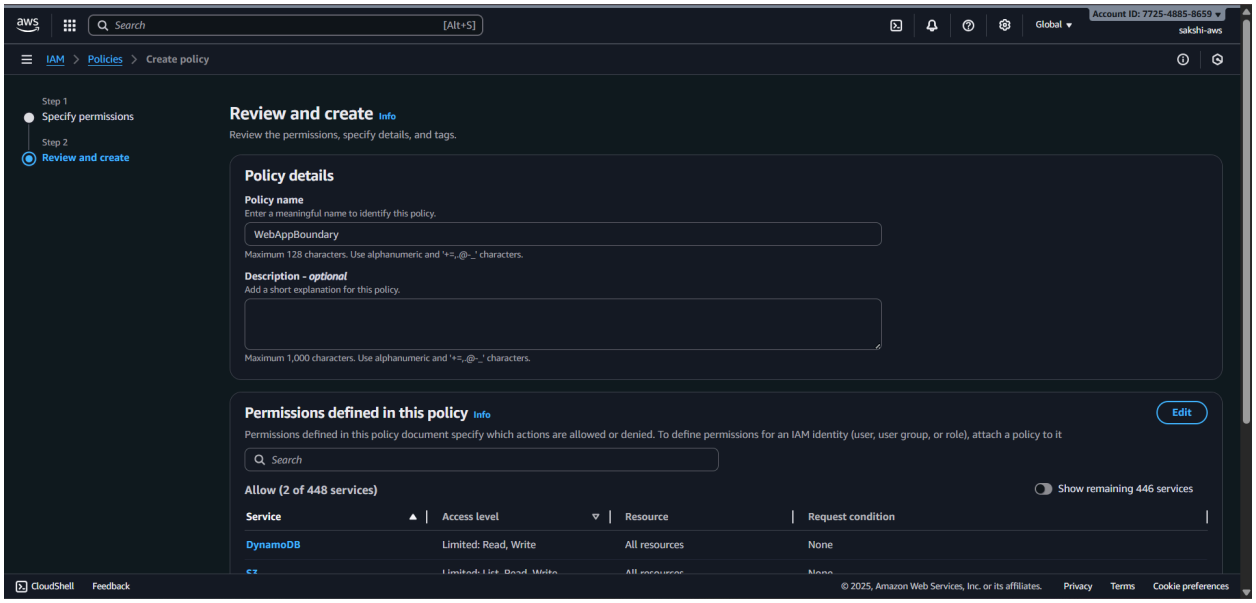
Admin View: Starting in the IAM Policies console to create two policies.



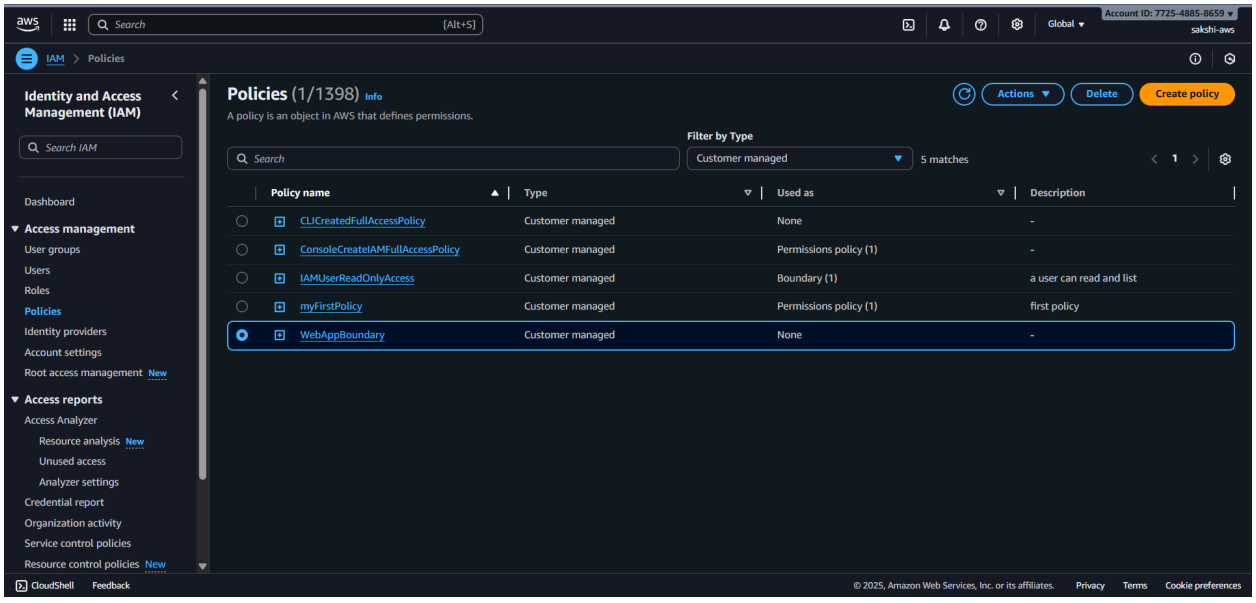
Creating WebAppBoundary, the policy that will act as our permissions guardrail.



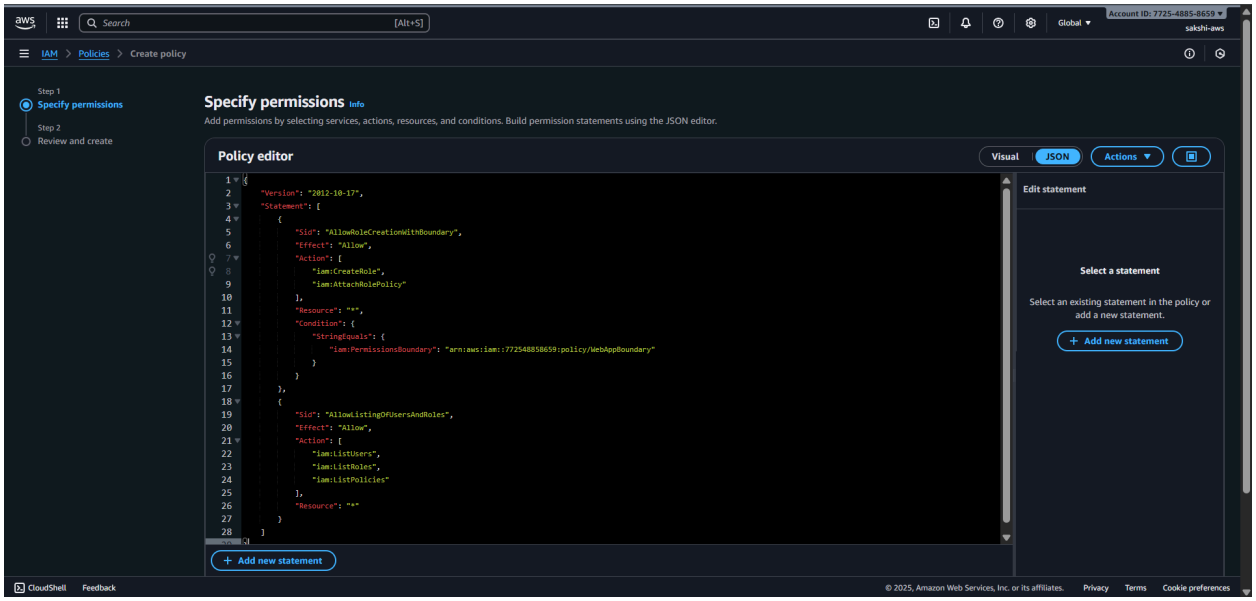
Naming and reviewing the WebAppBoundary policy before creation.



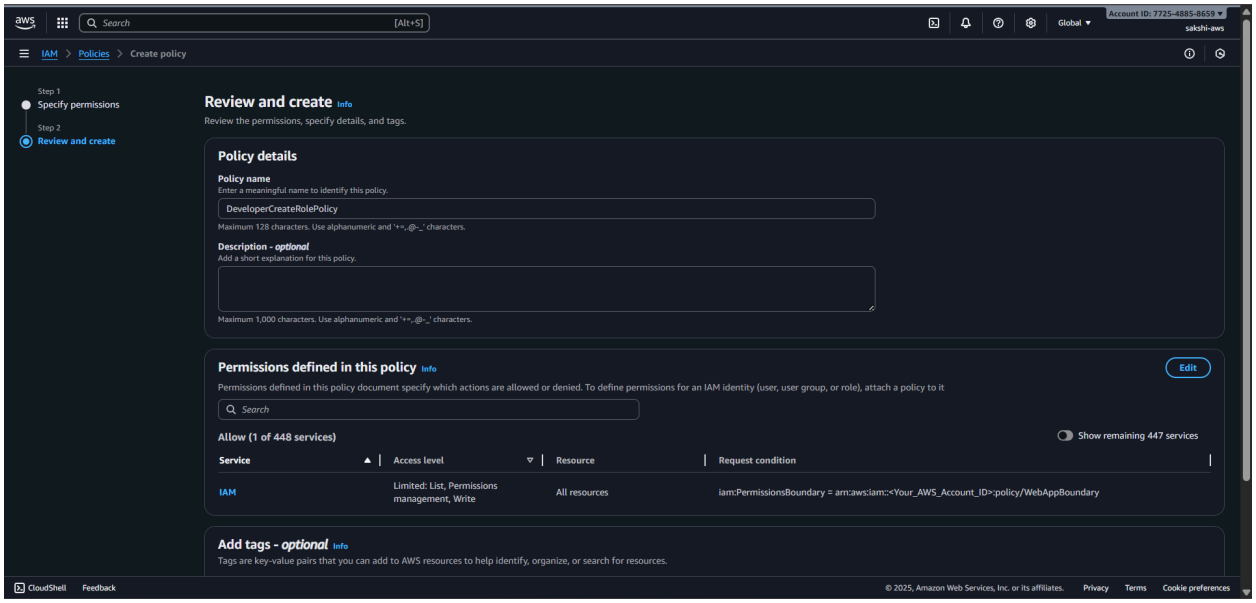
Confirming WebAppBoundary is created and available in the policy list



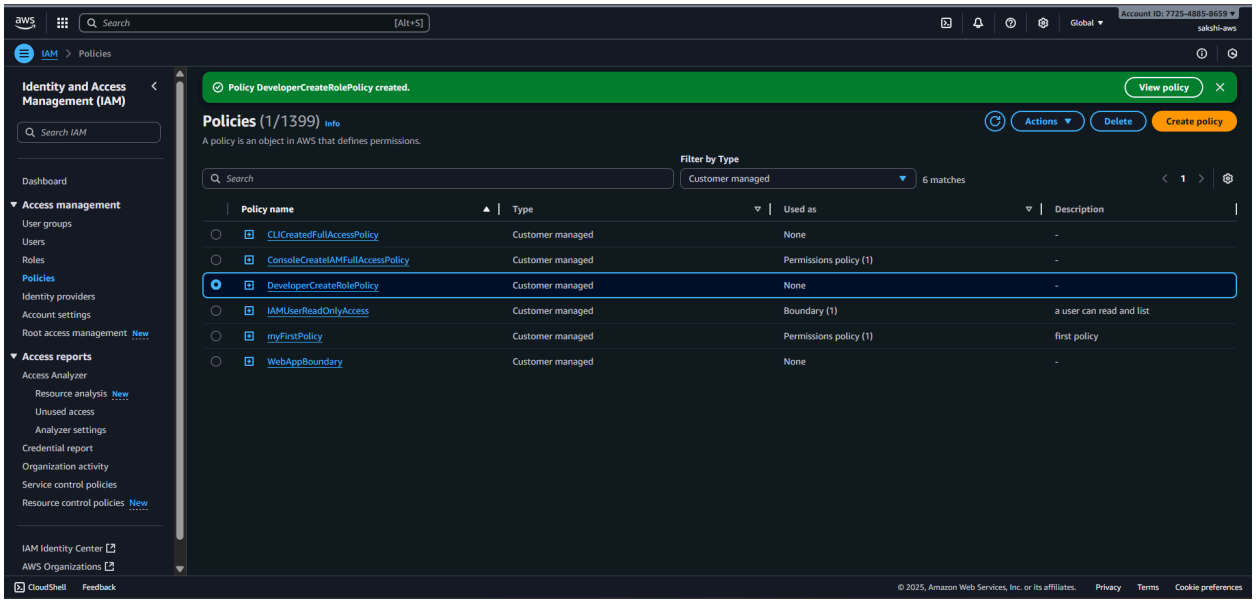
Creating DeveloperCreateRolePolicy to grant, yet control, developer actions.



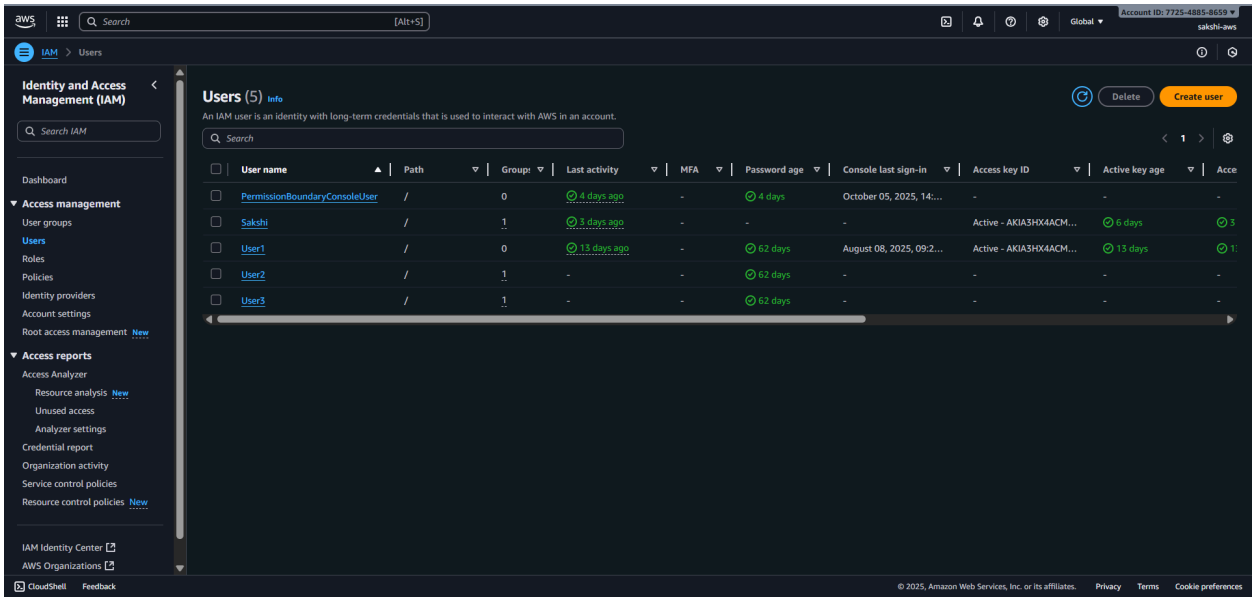
Reviewing the Condition that forces developers to use our boundary.



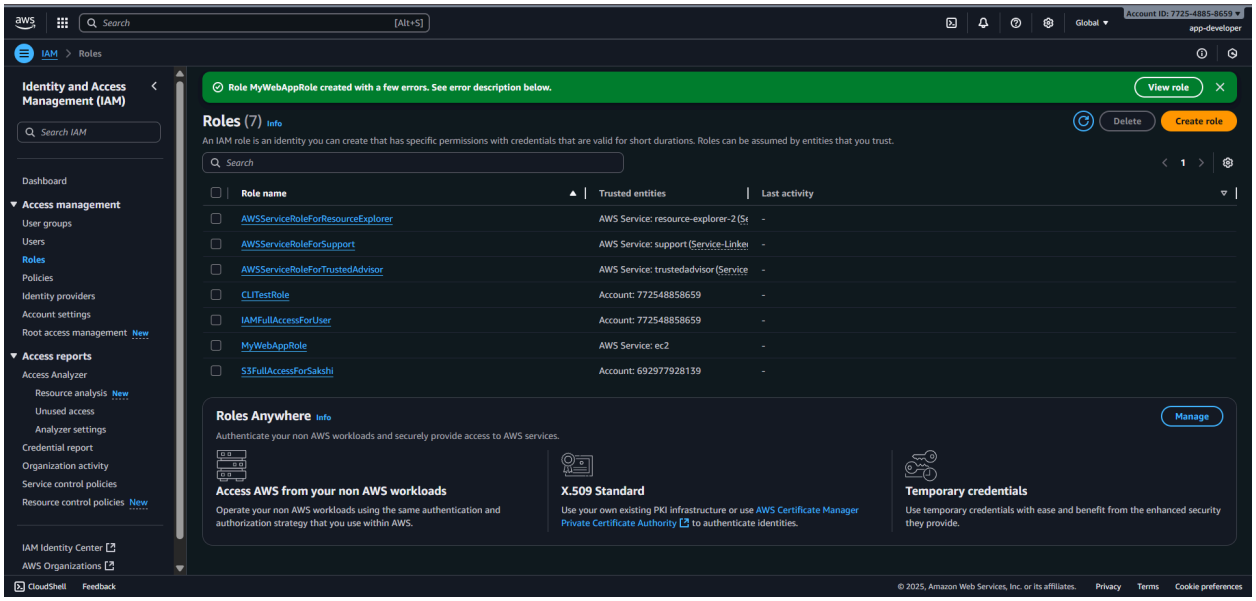
Admin prep complete: both WebAppBoundary and DeveloperCreateRolePolicy exist.



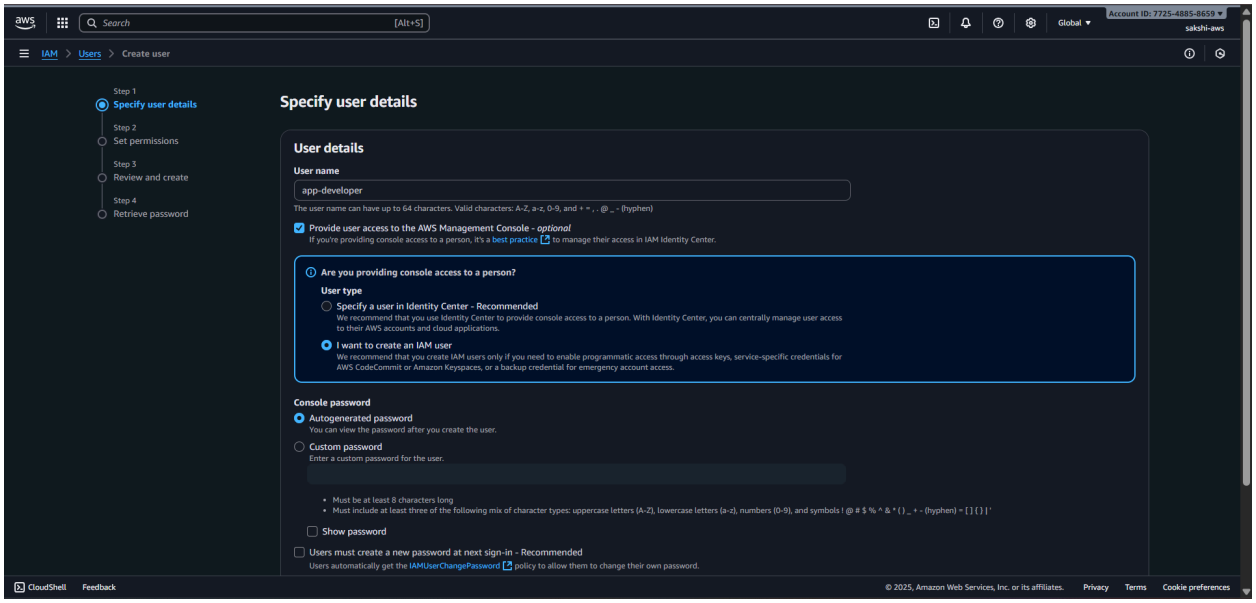
Admin View: Navigating to the IAM Users list before adding the developer.



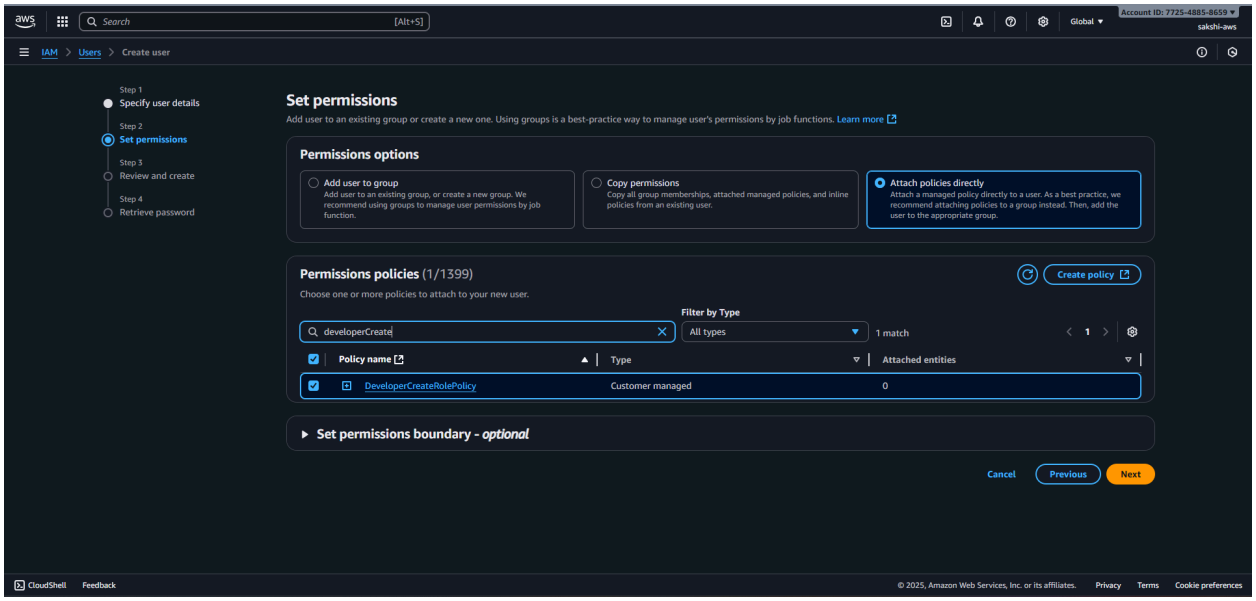
Confirmation: MyWebAppRole is created and its permissions are now limited.



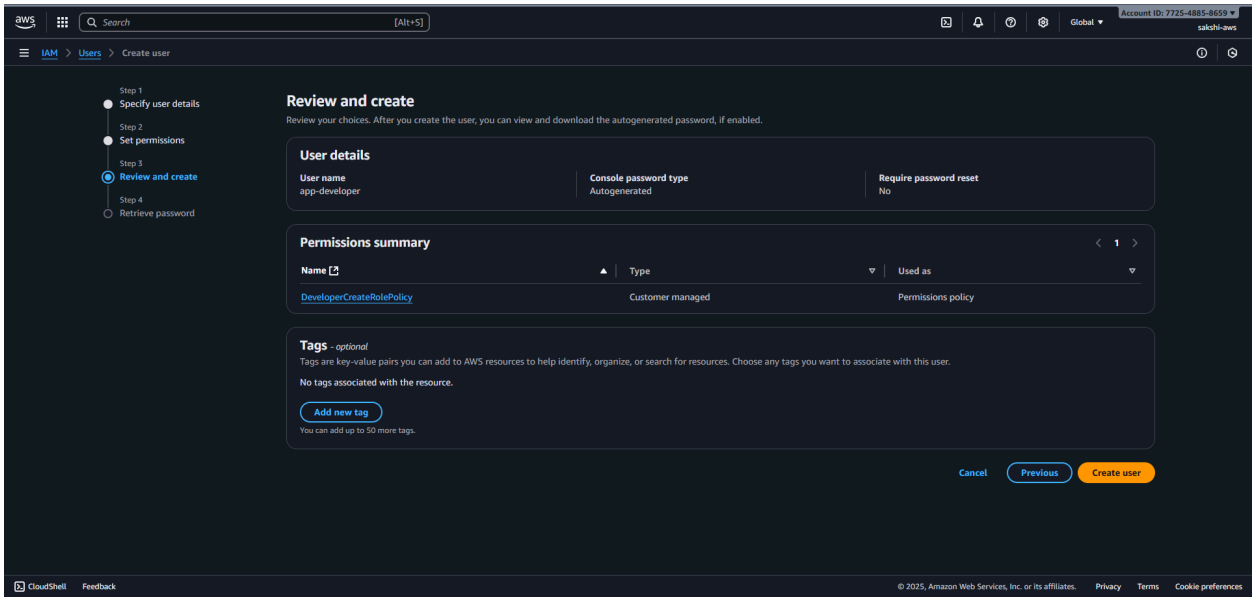
Defining the app-developer user and enabling console access.



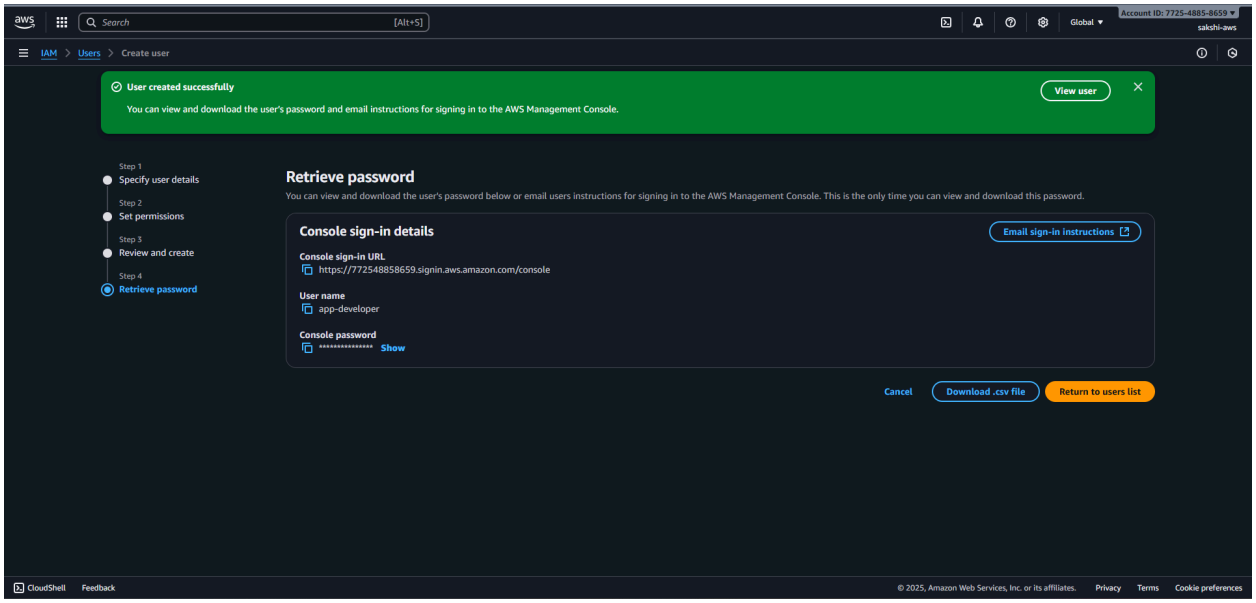
Attaching DeveloperCreateRolePolicy to the app-developer user.



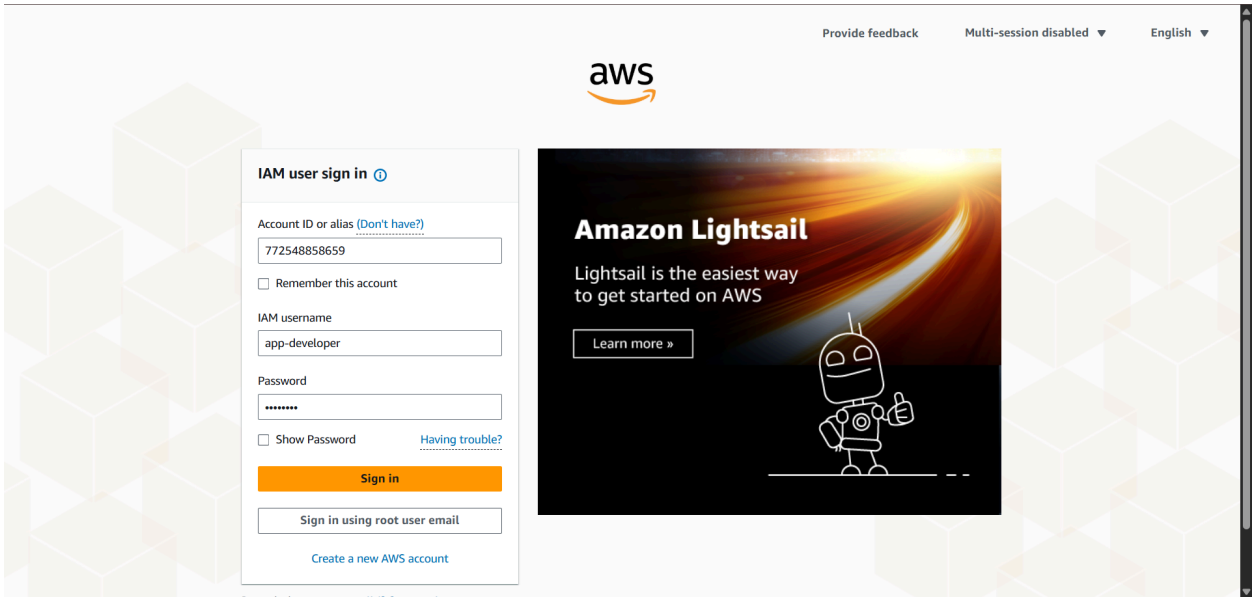
Final review of the app-developer user and its permissions.



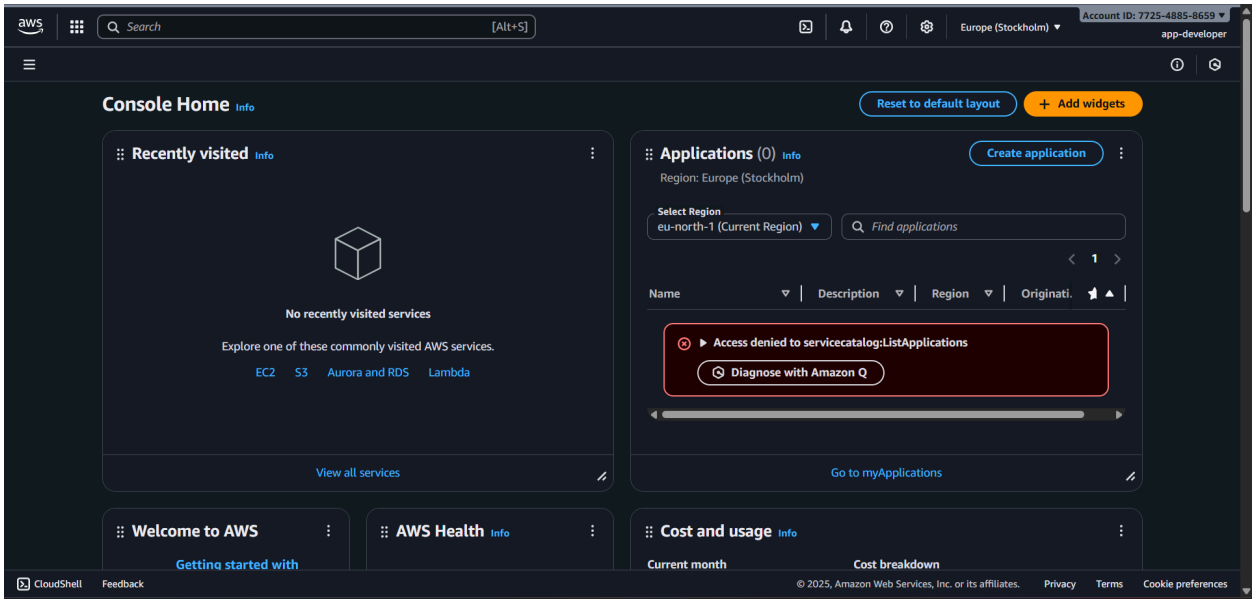
Retrieving the unique Console sign-in URL for the new user.



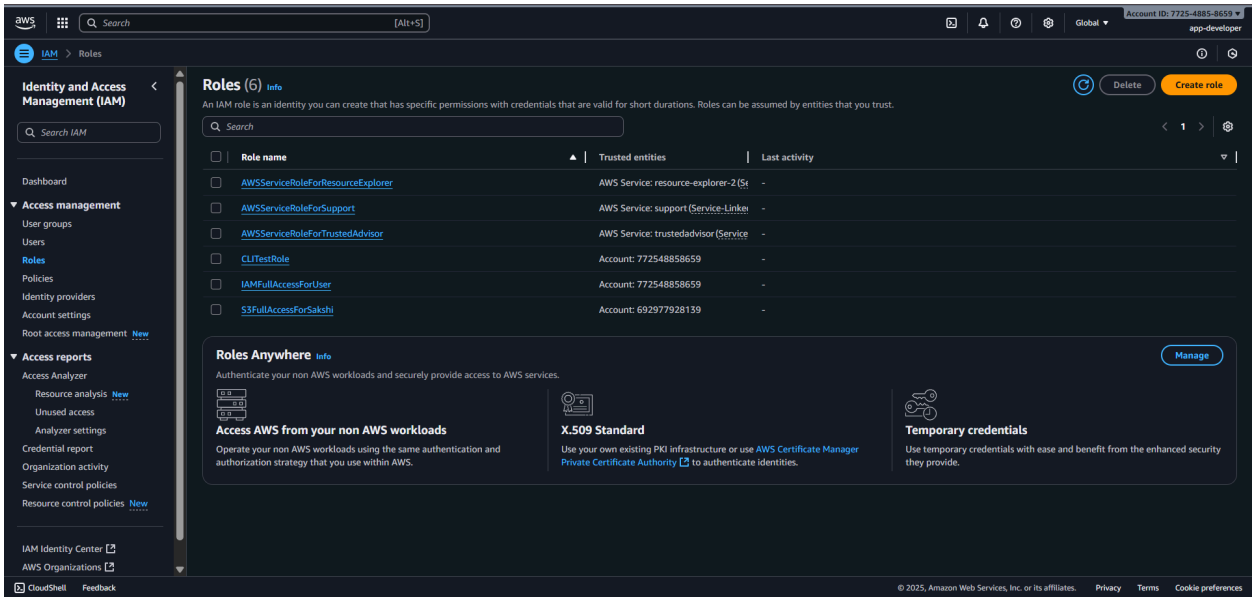
Developer View: Signing in with the app-developer credentials.



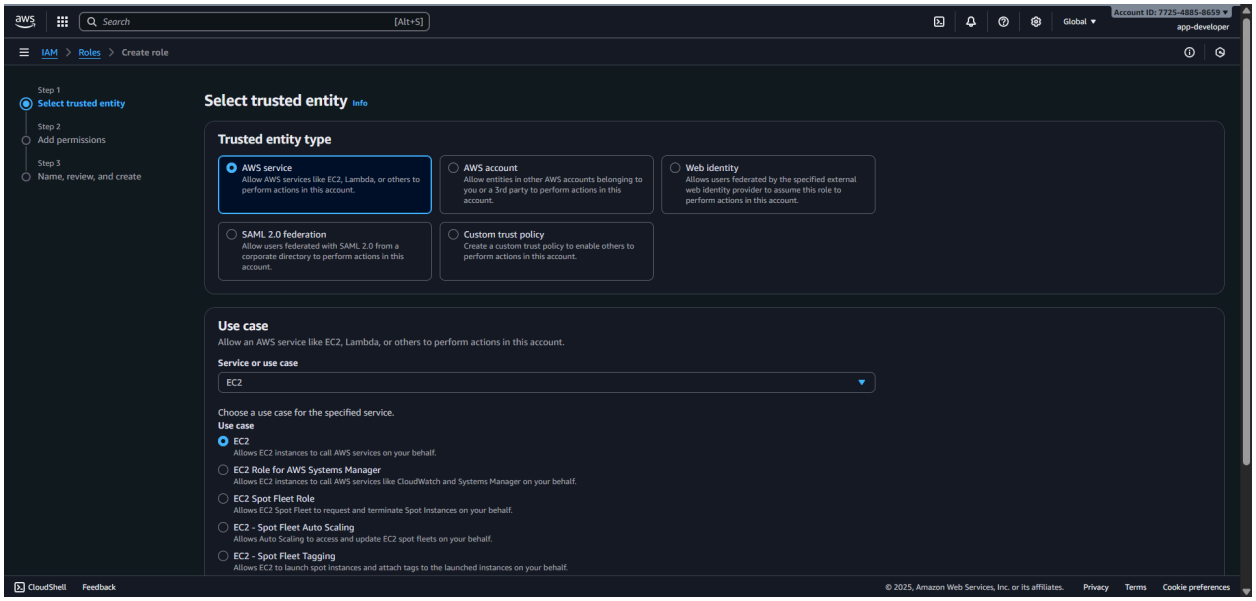
Developer is logged in; limited permissions are visible.



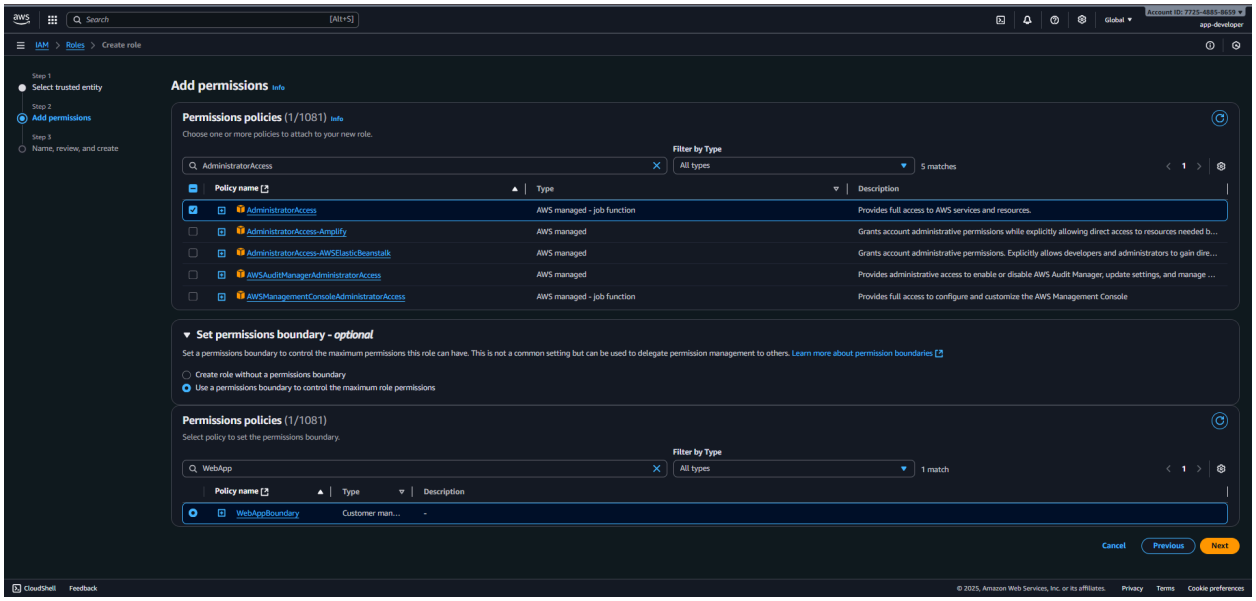
Navigating to the IAM Roles dashboard to perform their task.



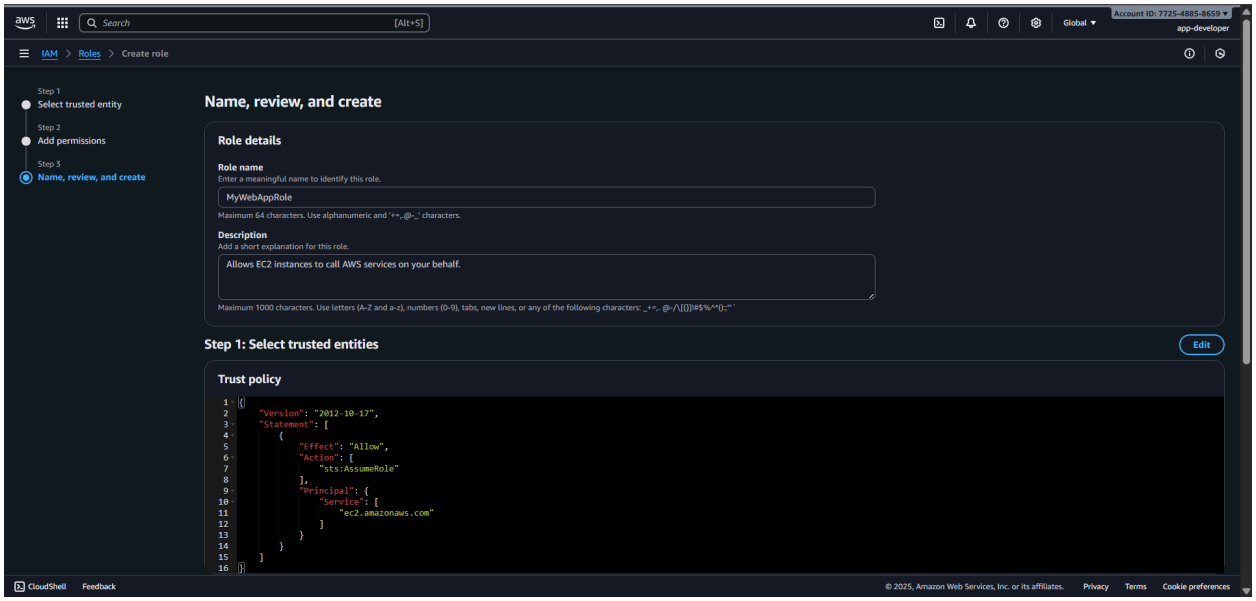
Starting role creation by selecting EC2 as the use case.



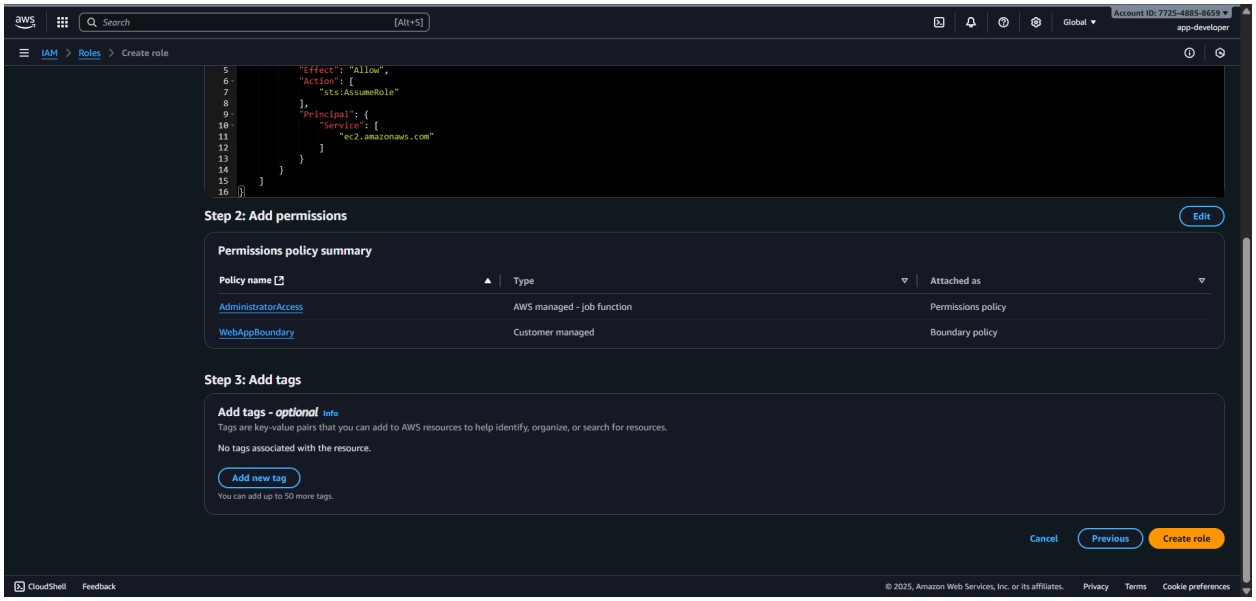
Attaching the AdministratorAccess policy to the role and setting the WebAppBoundary policy as the role's permission boundary.



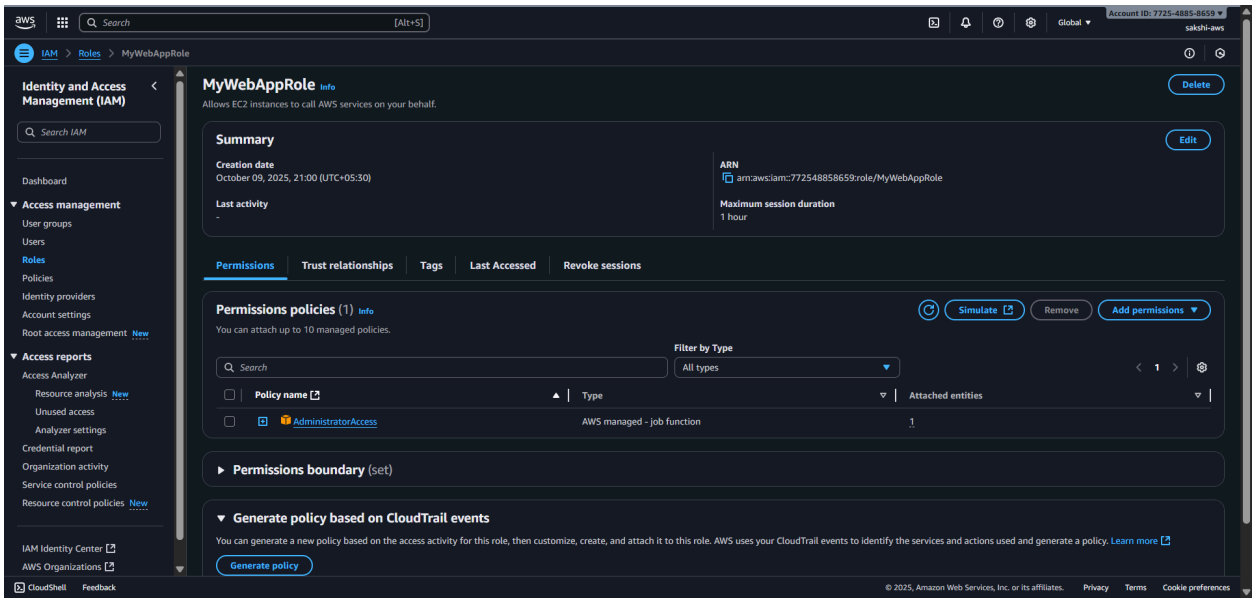
Reviewing MyWebAppRole, which has both the admin policy and the boundary.



Reviewing MyWebAppRole, which has both the admin policy and the boundary.



The summary page for MyWebAppRole visually confirms that both policies are active. The AdministratorAccess policy is listed under "Permissions policies," but the WebAppBoundary is also set, acting as the ultimate permissions guardrail for the role.



This practical has successfully demonstrated the use of AWS IAM Permissions Boundaries as a powerful feature for secure delegation. The objective, as stated in the introduction, was to simulate a scenario where a developer could be empowered to create IAM roles without posing a security risk.
The exercise was conducted in distinct phases: the administrator prepared the environment by creating a restrictive WebAppBoundary policy and a special DeveloperCreateRolePolicy that enforced its use. Subsequently, the app-developer user was created and granted these limited permissions. The final test involved the developer creating the MyWebAppRole, where they attempted to attach the highly permissive AdministratorAccess policy while being forced to apply the boundary.
The result, confirmed by the final verification screenshot, is a new role (MyWebAppRole) that has both policies attached simultaneously. This proves the core concept: the role's effective permissions are the intersection of both policies. Therefore, MyWebAppRole is prevented from performing any administrative actions outside of those explicitly allowed by the WebAppBoundary. This practical confirms that permissions boundaries are an essential tool for enabling developer agility and scaling cloud operations without compromising on security governance.