

Practical - 4: IAM Permissions Boundary Configuration

Objective: To demonstrate how an **IAM Permissions Boundary** acts as a "guardrail" to safely delegate permission to developers to create new IAM roles, limiting the effective permissions of the created roles, regardless of the policies they attach¹¹¹¹.

Scenario: A central administrator empowers a developer ([app-developer](#)) to create new application roles ([MyWebAppRole](#)). The administrator ensures that any role created will not exceed a set of predefined, restricted permissions by enforcing the use of a Permissions Boundary.

Phase 1: Administrator Setup - Create the Permissions Boundary Policy

This policy ([WebAppBoundary](#)) defines the **maximum permissions** that any role created by the developer is allowed to possess.

1. **Navigate to IAM Policies:** Log in as the administrator and go to **Identity and Access Management (IAM) -> Policies**
2. **Create Policy:** Click **Create policy**.
3. **Define Permissions:** Use the **JSON** editor to define the maximum allowed permissions (e.g., restricted access to DynamoDB and SQS for a web app).
 - o The screenshot shows permissions for **DynamoDB** (Limited Read, Write).
4. **Review and Name:** Name the policy **WebAppBoundary**.
5. **Create Policy:** Click **Create policy** and verify that [WebAppBoundary](#) is listed.

Phase 2: Administrator Setup - Create the Delegation Policy

This policy ([DeveloperCreateRolePolicy](#)) is attached to the developer user. It grants the necessary **IAM** permissions to create a role but includes a **Condition** that forces the developer to attach the [WebAppBoundary](#)⁹.

1. **Create Policy:** Click **Create policy** again.
2. **Define Permissions (IAM Actions):** Use the JSON editor to grant permissions for role creation. The policy must allow actions like [iam:CreateRole](#), [iam:PutRolePolicy](#), and [iam:AttachRolePolicy](#).

3. **Enforce Boundary (Condition): CRITICAL STEP.** The policy must include a condition using `iam:PermissionsBoundary` to enforce that the `WebAppBoundary` policy is used on any created role.
 4. **Review and Name:** Name the policy `DeveloperCreateRolePolicy`.
 5. **Create Policy:** Click **Create policy** and verify that `DeveloperCreateRolePolicy` is listed.
-

Phase 3: Administrator Setup - Create the Developer User

The administrator creates the user that will be delegated the permission to create roles.

1. **Create IAM User:** Navigate to **IAM -> Users**.
 2. **User Details:** Create a new user named `app-developer`
 3. **Console Access:** Enable **Console access** and set a password.
 4. **Attach Policy:** On the **Set permissions** page, select **Attach policies directly**.
 5. **Attach Delegation Policy:** Search for and select the `DeveloperCreateRolePolicy` created in Phase 2.
 6. **Review and Create:** Review the user configuration and click **Create user**.
-

Phase 4: Developer Action - Create a New Role (The Test)

The developer logs in and attempts to create a powerful role, which should be limited by the boundary.

1. **Developer Login:** Log in to the AWS Console using the `app-developer` user credentials.
2. **Start Role Creation:** Navigate to **IAM -> Roles** and click **Create role**.
3. **Select Trusted Entity:** Select **AWS service** and the **EC2** use case
4. **Add Permissions:**
 - Attach the highly permissive **AdministratorAccess** policy (the developer is *trying* to create an admin role).
 - Under **Set permissions boundary - optional**, the developer must choose a permissions boundary due to the `DeveloperCreateRolePolicy` condition. They select the `WebAppBoundary`
5. **Name and Create:** Name the role `MyWebAppRole` and click **Create role**. The role creation succeeds.

Phase 5: Validation - The Intersection of Permissions

The final result demonstrates the core concept of the Permissions Boundary.

1. **View Role Summary (Admin View):** Log back in as the administrator and navigate to the summary page for **MyWebAppRole**.
2. **Verify Policies:**
 - The role has the **AdministratorAccess** policy listed under "Permissions policies".
 - The role also has the **WebAppBoundary** policy listed under "Permissions boundary".
3. **Conclusion:** The role's **effective permissions** are the **intersection** of the highly permissive **AdministratorAccess** and the restricted **WebAppBoundary**. Therefore, even though the developer attached the **AdministratorAccess** policy, **MyWebAppRole** can only perform actions explicitly allowed by the **WebAppBoundary**, maintaining security governance³³.