

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.

The screenshot shows the AWS IAM Policies console. The left sidebar includes sections for Identity and Access Management (IAM), Access management, Access reports, and CloudShell. The main area displays a table titled 'Policies (1397)'. The table has columns for Policy name, Type, Used as, and Description. A filter bar at the top right shows 'Customer managed' and '4 matches'. The table lists four policies: 'CLICreatedFullAccessPolicy', 'ConsoleCreateIAMFullAccessPolicy', 'IAMUserReadOnlyAccess', and 'myFirstPolicy'. The 'myFirstPolicy' row shows a tooltip for 'first policy'.

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

The screenshot shows the 'Specify permissions' step in the 'Create policy' wizard. The left sidebar shows 'Step 1: Specify permissions' is selected. The main area has tabs for 'Visual' (selected), 'JSON', and 'Actions'. The 'Policy editor' section contains a JSON code block:

```
1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "AllowS3AndDynamoDB",
6              "Effect": "Allow",
7              "Action": [
8                  "s3:GetObject",
9                  "s3:PutObject",
10                 "s3>ListBucket",
11                 "dynamodb:GetItem",
12                 "dynamodb:PutItem"
13             ],
14             "Resource": "*"
15         }
16     ]
17 }
```

The right side of the screen shows a 'Select a statement' panel with a 'Select an existing statement in the policy or add a new statement' message and a '+ Add new statement' button.

Naming and reviewing the WebAppBoundary policy before creation.

The screenshot shows the 'Create policy' wizard in the AWS IAM console. The current step is 'Review and create'. The policy name is set to 'WebAppBoundary'. The 'Description - optional' field is empty. Under 'Permissions defined in this policy', it shows an 'Allow' rule for 'DynamoDB' with 'Read, Write' access level and 'All resources' resource. There are 446 more services listed. The bottom of the screen shows standard AWS navigation links like CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Confirming WebAppBoundary is created and available in the policy list

The screenshot shows the 'Policies' list in the AWS IAM console. The 'WebAppBoundary' policy is visible in the list, highlighted with a blue border. The table columns include Policy name, Type, Used as, and Description. The 'Used as' column for 'WebAppBoundary' shows 'None'. The 'Description' column shows '-'. The left sidebar shows the navigation path: IAM > Policies. The bottom of the screen shows standard AWS navigation links like CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Creating DeveloperCreateRolePolicy to grant, yet control, developer actions.

The screenshot shows the 'Specify permissions' step of creating a new IAM policy. The JSON editor contains the following policy document:

```
1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Sid": "AllowRoleCreationWithBoundary",
5         "Effect": "Allow",
6         "Action": [
7             "iam:CreateRole",
8             "iam:AttachRolePolicy"
9         ],
10        "Resource": "*",
11        "Condition": {
12            "StringEquals": {
13                "iam:PermissionsBoundary": "arn:aws:iam::772548858659:policy/WebAppBoundary"
14            }
15        }
16    },
17    {
18        "Sid": "AllowListingOfUsersAndRoles",
19        "Effect": "Allow",
20        "Action": [
21            "iam:ListUsers",
22            "iam:ListRoles",
23            "iam:ListPolicies"
24        ],
25        "Resource": "*"
26    }
27]
28
```

A modal window titled 'Edit statement' is open on the right, with the sub-titile 'Select a statement'. It says 'Select an existing statement in the policy or add a new statement.' and has a button '+ Add new statement'.

Reviewing the Condition that forces developers to use our boundary.

The screenshot shows the 'Review and create' step of creating a new IAM policy. The 'Policy details' section includes:

- Policy name:** DeveloperCreateRolePolicy
- Description - optional:** A placeholder for a short explanation of the policy.

The 'Permissions defined in this policy' section shows:

- Allow (1 of 448 services):** A table with columns: Service, Access level, Resource, and Request condition. One row is shown: **IAM**, Limited: List, Permissions management, Write, All resources, iam:PermissionsBoundary = arn:aws:iam:<Your_AWS_Account_ID>:policy/WebAppBoundary

The 'Add tags - optional' section is present at the bottom.

Admin prep complete: both WebAppBoundary and DeveloperCreateRolePolicy exist.

The screenshot shows the AWS IAM Policies page. At the top, a green banner displays the message "Policy DeveloperCreateRolePolicy created." Below this, the title "Policies (1/1399) Info" is shown, with a note that a policy is an object in AWS that defines permissions. A search bar and a filter button ("Filter by Type") are present. The main table lists six policies, with "DeveloperCreateRolePolicy" highlighted. The columns in the table are: Policy name, Type, Used as, and Description. The "DeveloperCreateRolePolicy" row shows "Customer managed" under Type, "None" under Used as, and "- " under Description. Other policies listed include "CLICreatedFullAccessPolicy", "ConsoleCreateIAMFullAccessPolicy", "IAMUserReadOnlyAccess", "myFirstPolicy", and "WebAppBoundary".

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

The screenshot shows the AWS IAM Users page. At the top, a green banner displays the message "Users (5) Info". Below this, the title "Users (5) Info" is shown, with a note that an IAM user is an identity with long-term credentials used to interact with AWS. A search bar is present. The main table lists five users: "PermissionBoundaryConsoleUser", "Sakshi", "User1", "User2", and "User3". The columns in the table are: User name, Path, Group, Last activity, MFA, Password age, Console last sign-in, Access key ID, Active key age, and Active. "PermissionBoundaryConsoleUser" is listed under Path "/". "Sakshi" is listed under Path "/" and Group "1". "User1", "User2", and "User3" are listed under Path "/" and Group "0". All users have "4 days ago" as their last activity. "User1" has "62 days" as their password age, while others have "4 days". "User1" has "October 05, 2025, 14:..." as their console last sign-in, while others have "-". "User1" and "User3" have active access key IDs, while others do not. "User1" has an active key age of "13 days", while others have "6 days".

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

The screenshot shows the AWS IAM Roles page. A success message at the top states: "Role MyWebAppRole created with a few errors. See error description below." Below this, a table lists seven roles, including "MyWebAppRole". The table columns are "Role name", "Trusted entities", and "Last activity". The "MyWebAppRole" row shows it was created by "AWS Service: ec2" and has no last activity. At the bottom of the page, there are sections for "Roles Anywhere" and "Access AWS from your non AWS workloads", both of which mention "X.509 Standard".

Defining the app-developer user and enabling console access.

The screenshot shows the "Create user" wizard, Step 1: Specify user details. The user name is set to "app-developer". The "Provide access to the AWS Management Console - optional" checkbox is checked. The "Are you providing console access to a person?" section is expanded, showing the "Specify a user in Identity Center - Recommended" option selected. The "I want to create an IAM user" option is also visible. The "Console password" section shows "Autogenerated password" selected. The "Users must create a new password at next sign-in - Recommended" checkbox is checked. The bottom of the page includes standard AWS footer links: CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Attaching DeveloperCreateRolePolicy to the app-developer user.

The screenshot shows the 'Set permissions' step of creating a new IAM user. The left sidebar shows steps 1 through 4: Step 1 (Specify user details), Step 2 (Set permissions, which is selected), Step 3 (Review and create), and Step 4 (Retrieve password). The main area is titled 'Set permissions' with the sub-section 'Permissions options'. It includes three options: 'Add user to group' (radio button), 'Copy permissions' (radio button), and 'Attach policies directly' (radio button, which is selected). Below this is a section titled 'Permissions policies (1/1399)' with a search bar containing 'developerCreate'. A table lists one policy: 'DeveloperCreateRolePolicy' (Customer managed, Type: Policy name). At the bottom right are 'Cancel', 'Previous', and 'Next' buttons, with 'Next' being highlighted in orange.

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

The screenshot shows the 'Review and create' step of creating a new IAM user. The left sidebar shows steps 1 through 4: Step 1 (Specify user details), Step 2 (Set permissions), Step 3 (Review and create, which is selected), and Step 4 (Retrieve password). The main area is titled 'Review and create' with the sub-section 'User details'. It shows 'User name: app-developer', 'Console password type: Autogenerated', and 'Require password reset: No'. Below this is a 'Permissions summary' table with one row: 'DeveloperCreateRolePolicy' (Customer managed, Type: Policy name, Used as: Permissions policy). There is also a 'Tags - optional' section with a note about key-value pairs and a 'Add new tag' button. At the bottom right are 'Cancel', 'Previous', and 'Create user' buttons, with 'Create user' being highlighted in orange.

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

The screenshot shows the 'Create user' process in the AWS IAM console. Step 4, 'Retrieve password', is selected. A green success message at the top states 'User created successfully'. Below it, instructions say 'You can view and download the user's password and email instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.' A 'View user' button is available. On the left, a sidebar lists steps: Step 1 (Specify user details), Step 2 (Set permissions), Step 3 (Review and create), and Step 4 (Retrieve password). The 'Console sign-in details' section displays the 'Console sign-in URL' (https://772548858659.signin.aws.amazon.com/console), 'User name' (app-developer), and 'Console password' (*****). An 'Email sign-in instructions' button is present. At the bottom are 'Cancel', 'Download.csv file', and 'Return to users list' buttons. The footer includes links for CloudShell, Feedback, and various AWS policies.

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

The screenshot shows the AWS sign-in page. The 'IAM user sign in' form is displayed, requiring an 'Account ID or alias' (772548858659), an 'IAM username' (app-developer), and a 'Password'. A 'Sign in' button is at the bottom. To the right, a promotional banner for 'Amazon Lightsail' is shown, stating 'Lightsail is the easiest way to get started on AWS' with a 'Learn more' link and a cartoon robot icon. The top right of the page shows 'Provide feedback', 'Multi-session disabled', and language settings ('English').

Developer is logged in; limited permissions are visible.

The screenshot shows the AWS Console Home page. In the top right corner, it displays "Account ID: 7725-4885-8659" and "Europe (Stockholm)". The main area features several widgets: "Recently visited" (empty), "Applications (0)" (empty), "Cost and usage" (Current month and Cost breakdown), and "Welcome to AWS" (Getting started with). A red box highlights an error message in the "Applications" section: "Access denied to servicecatalog>ListApplications".

Navigating to the IAM Roles dashboard to perform their task.

The screenshot shows the IAM Roles dashboard. The left sidebar includes sections for Identity and Access Management (IAM) like Dashboard, Access management (User groups, Users, Roles, Policies, Identity providers, Account settings, Root access management), Access reports (Access Analyzer, Resource analysis, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies, Resource control policies), and IAM Identity Center/AWS Organizations. The main content area shows a table of "Roles (6)":

Role name	Trusted entities	Last activity
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Service)	-
AWSServiceRoleForSupport	AWS Service: support (Service-Linker)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
CLITestRole	Account: 772548858659	-
IAMFullAccessForUser	Account: 772548858659	-
S3FullAccessForSakshi	Account: 692977928139	-

Below the table are sections for "Roles Anywhere" (Authenticate your non AWS workloads and securely provide access to AWS services), "Access AWS from your non AWS workloads" (Operate your non AWS workloads using the same authentication and authorization strategy that you use within AWS), "X.509 Standard" (Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to authenticate identities), and "Temporary credentials" (Use temporary credentials with ease and benefit from the enhanced security they provide).

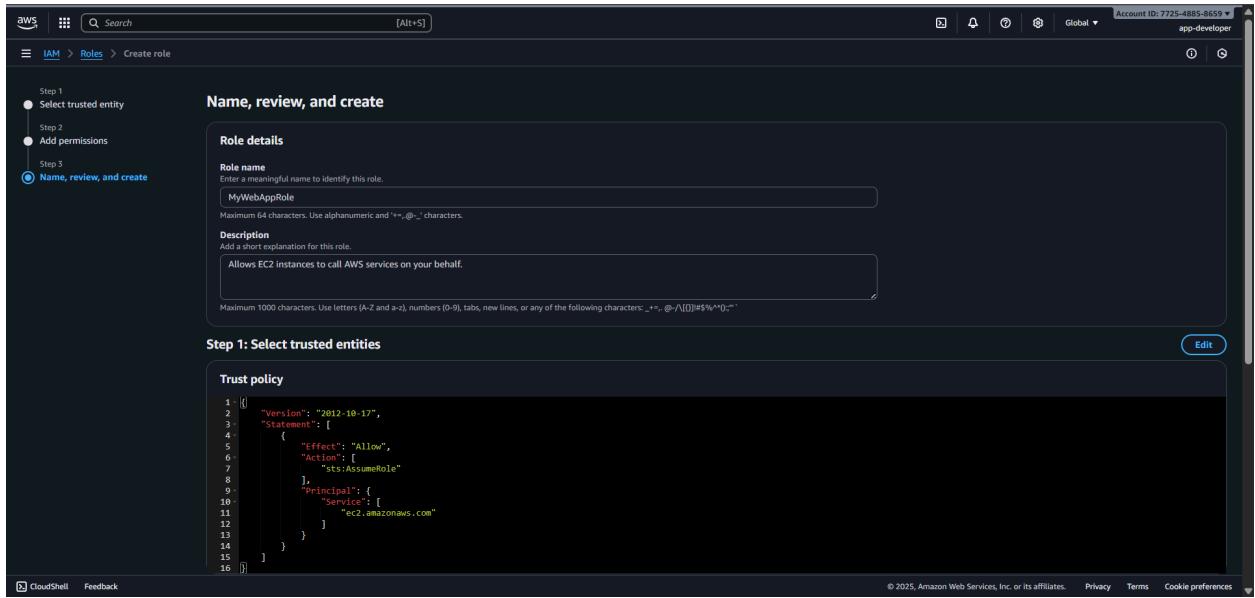
Starting role creation by selecting EC2 as the use case.

The screenshot shows the 'Create role' wizard in the AWS IAM console. The current step is 'Step 1: Select trusted entity'. In the 'Trusted entity type' section, 'AWS service' is selected, which is highlighted with a blue border. Other options like 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy' are shown but not selected. In the 'Use case' section, 'EC2' is selected from a dropdown menu. Below the dropdown, a list of EC2-related use cases is provided, with 'EC2' also being the selected option. The bottom of the screen shows standard AWS navigation and footer links.

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

The screenshot shows the 'Create role' wizard in the AWS IAM console, moving to 'Step 2: Add permissions'. In the 'Permissions policies' section, the 'AdministratorAccess' policy is selected and highlighted with a blue border. Below this, there is a note about setting a permissions boundary. In the 'Set permissions boundary' section, the 'Use a permissions boundary to control the maximum role permissions' option is selected. In the 'Permissions policies' section at the bottom, the 'WebAppBoundary' policy is listed and selected. The bottom of the screen shows standard AWS navigation and footer links.

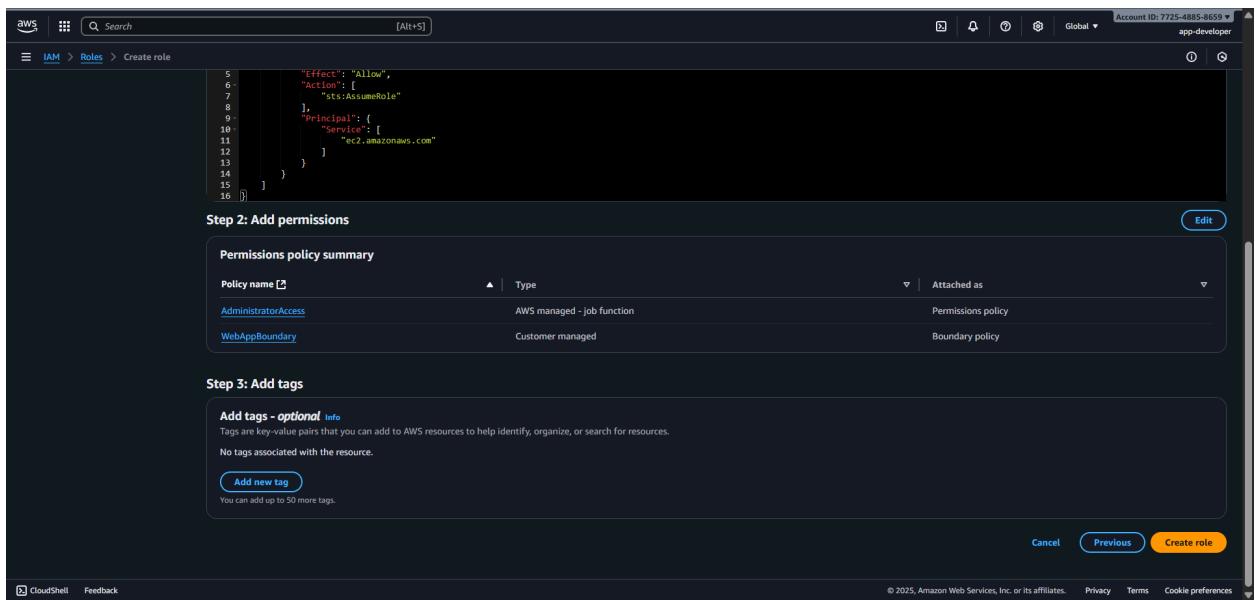
Reviewing MyWebAppRole, which has both the admin policy and the boundary.



The screenshot shows the AWS IAM 'Create role' wizard at Step 1: Select trusted entities. The 'Role name' field is set to 'MyWebAppRole'. The 'Description' field contains the text: 'Allows EC2 instances to call AWS services on your behalf.' Below the form, the 'Trust policy' section displays the following JSON code:

```
1 [{
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "sts:AssumeRole"
8       ],
9       "Principal": [
10      {
11        "Service": [
12          "ec2.amazonaws.com"
13        ]
14      }
15    ]
16  }]
}
```

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



The screenshot shows the AWS IAM 'Create role' wizard at Step 2: Add permissions. The 'Permissions policy summary' table lists two policies:

Policy name	Type	Attached as
AdministratorAccess	AWS managed - job function	Permissions policy
WebAppBoundary	Customer managed	Boundary policy

Below the table, the 'Step 3: Add tags' section is shown. It includes a note: 'Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.' A button labeled 'Add new tag' is present, along with a note: 'You can add up to 50 more tags.'

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.

The screenshot shows the AWS IAM Roles page with 'MyWebAppRole' selected. The left sidebar shows navigation options like Dashboard, Access management, Access reports, and IAM Identity Center. The main panel displays the role's summary, including its ARN (arn:aws:iam::772548858659:role/MyWebAppRole) and creation date (October 09, 2025). It lists one attached policy, 'AdministratorAccess'. The 'Permissions' tab is selected, showing the attached policy and the 'Permissions boundary' section. The 'Permissions boundary' section indicates that the role has a boundary named 'WebAppBoundary'. At the bottom, there are links for 'Generate policy based on CloudTrail events' and 'Generate policy'.

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.