Permissions Boundary workshop

Advanced edition

Question

Who would be comfortable giving developers permission to create IAM roles (e.g. for Lambda functions) in production accounts?

Problem: safe delegation of permission management

- Should use caution when granting permission to create users and roles
- But there are many situations where user and role creation is required
- So, we need a way to safely delegate permission management

Solution: Permissions boundaries

Safely delegate permission management.

Free up developers (get out of their way) and do so securely.

Also allow multiple teams in the same account to do permission management.

Agenda

Basics

Demo

Mechanism

Resource Restrictions

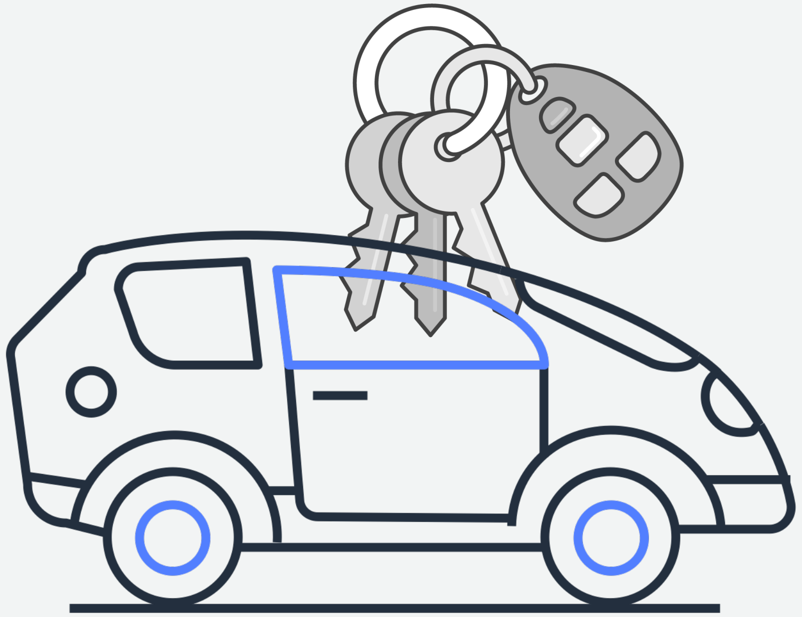
Hands on

Basics

Bob gives the car keys to his teenager

- Car keys give a lot of power: drive fast, drive anywhere, etc.
- You can set rules: don’t speed, don’t go beyond 20 mile range, etc.
- …but, you can only verify that they followed your rules (check odometer, see if they got a speeding ticket or got into an accident.)

Once you have the car keys you can drive however you want.

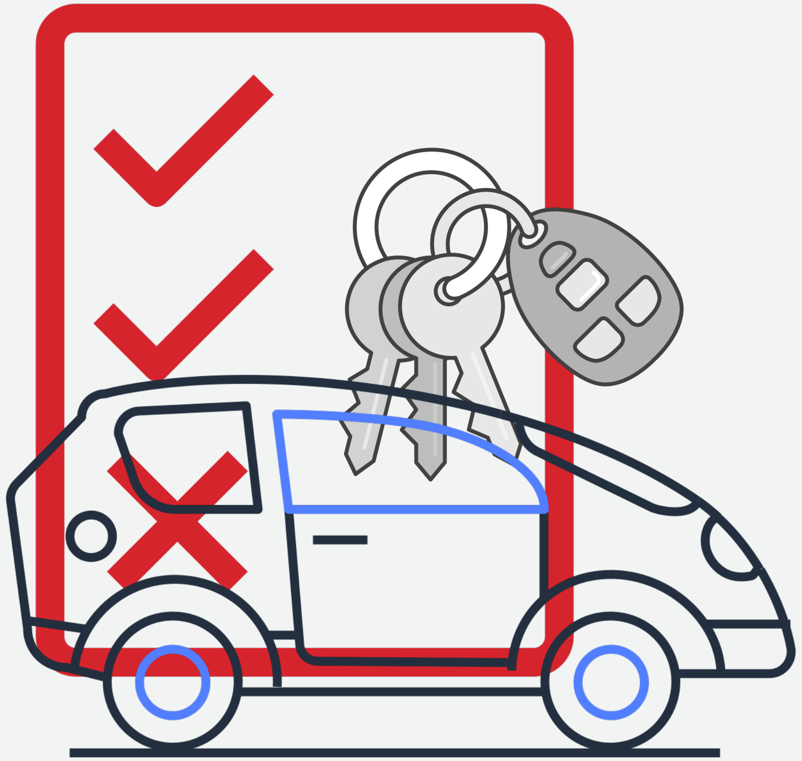


Bob gives the car keys to his teenager

- Some cars have programmable keys so you can restrict certain parameters.

- Ability (permission) of the car key is the intersection between the desire of the driver and the settings you program. **Bound keys.**

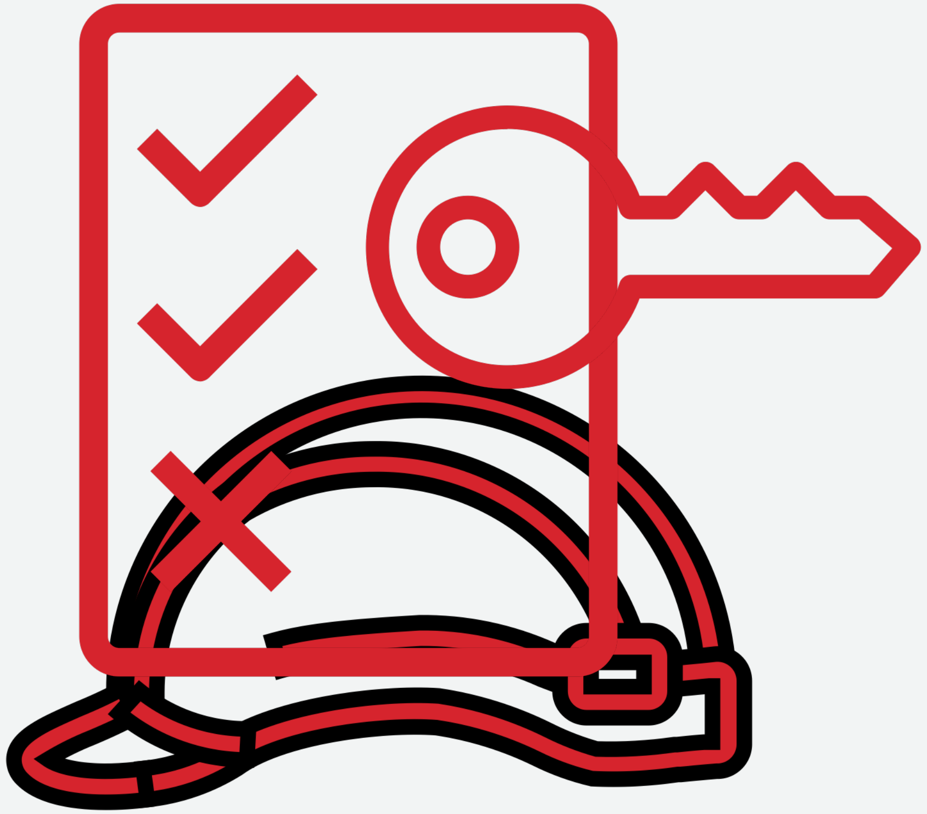
Key programming sets maximum ability of the key.



Bob gives permissions management to developers

- Permission to create users or roles provides a lot of power.
- Developer attaches policies (what they want a role to be able to do) but you also require a permissions boundary (like the programming on the car key).
- Effective permission of the role is the intersection of the two. **Bound roles.**

Permissions boundary sets maximum permissions of the role they create.



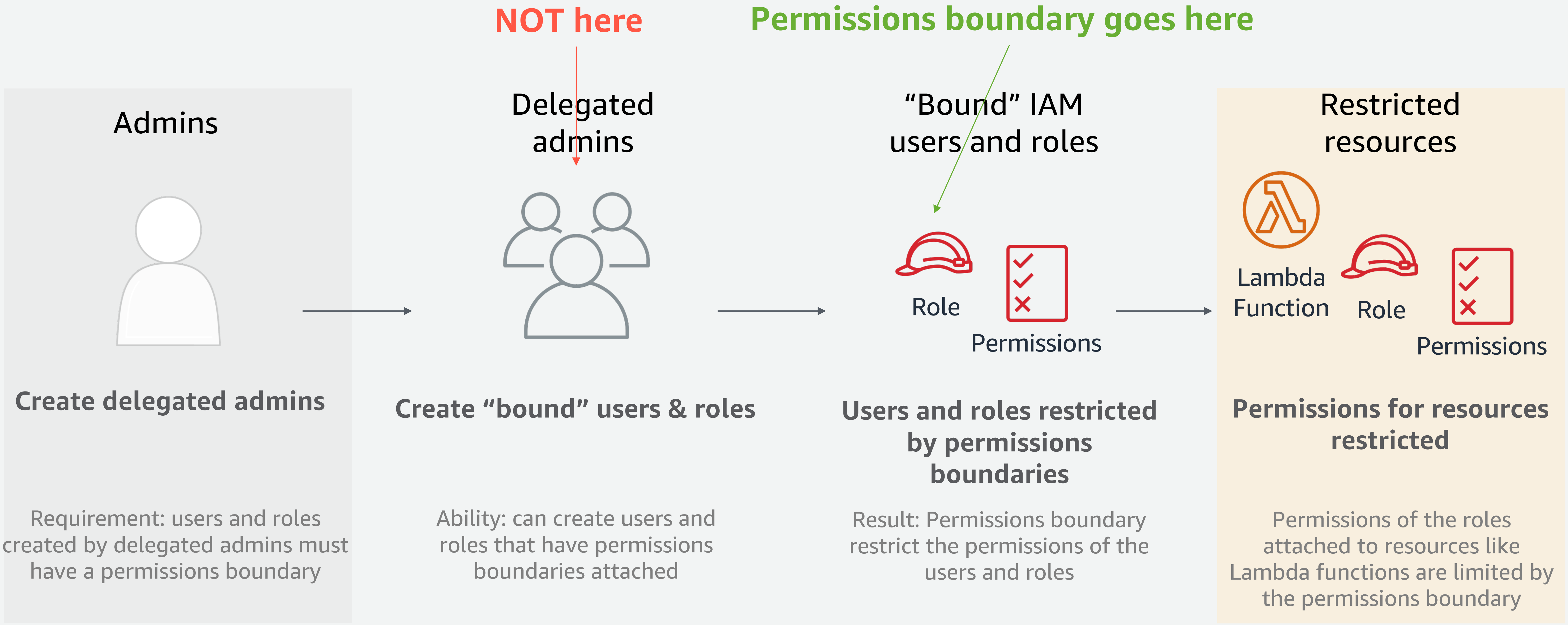
What are permissions boundaries?

Allow you to delegate permission to create users and roles while preventing privilege escalation or unnecessarily broad permissions.

How do Permissions Boundaries work?

Control the maximum permissions of a user or role created by a delegated admin.

Permissions boundaries – workflow



NOT here

Permissions boundary goes here

Admins

Create delegated admins

Requirement: users and roles created by delegated admins must have a permissions boundary

Delegated admins

Create “bound” users & roles

Ability: can create users and roles that have permissions boundaries attached

“Bound” IAM users and roles

Role

Permissions

Users and roles restricted by permissions boundaries

Result: Permissions boundary restrict the permissions of the users and roles

Restricted resources

Lambda Function

Role

Permissions

Permissions for resources restricted

Permissions of the roles attached to resources like Lambda functions are limited by the permissions boundary

It’s just a condition …

```
"Condition": {"StringEquals":
    {"iam:PermissionsBoundary":
    "arn:aws:iam::ACCOUNT_ID:policy/permissionboundary"
    }
}
```

… applied to principal actions

```
"Effect": "Allow",
"Action": ["iam:CreateRole"],
"Resource": ["arn:aws:iam::ACCOUNT_ID:role/path/*"],
"Condition": {"StringEquals":
    {"iam:PermissionsBoundary":
    "arn:aws:iam::ACCOUNT_ID:policy/permissionboundary"
    }
}
```

End user experience changes little

Create role for a Lambda function

# Step 1: Create role and attach permissions boundary

```
$ aws iam create-role -role-name Some_Role -path /Some_Path/
-assume-role-policy-document file://Some_Trust_Policy.json
-permissions-boundary arn:aws:iam::<ACCOUNT_NUMBER>:policy/Permissions_Boundary
```

# Step 2: Create identity-based policy

```
No change
```

# Step 3: Attach identity-based policy

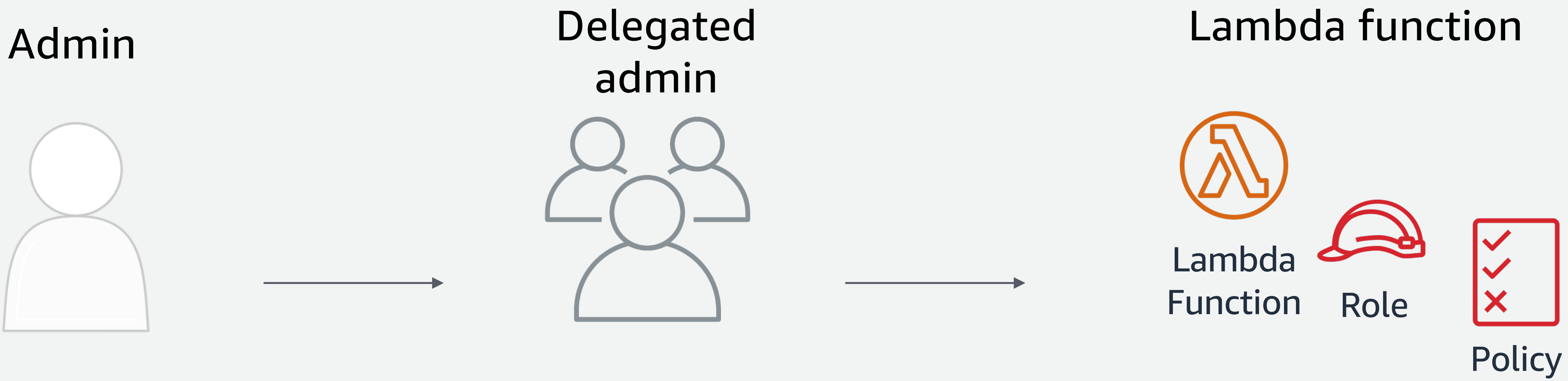
```
No change
```

Use cases

- Builders (e.g. creating roles for Lambda functions)
- Application owners creating roles for EC2 instances
- Admins creating users for particular situations
- Any others?

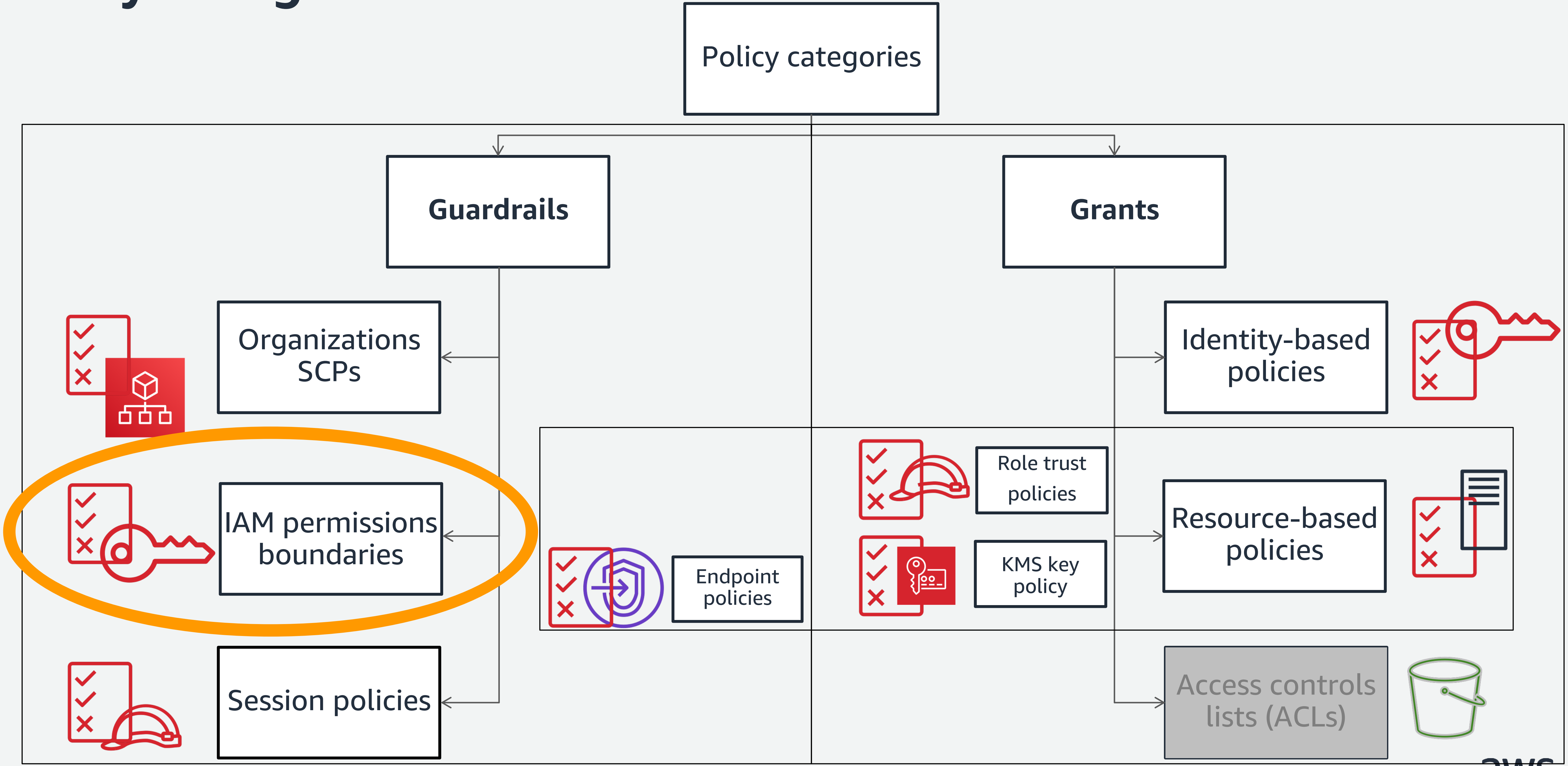
Demo

Demo


Admin
Delegated admin
Lambda function
Lambda Function
Role
Policy

Categories

Policy categories



Policy categories

Guardrails

Grants

Organizations SCPs

IAM permissions boundaries

Session policies

Endpoint policies

Role trust policies

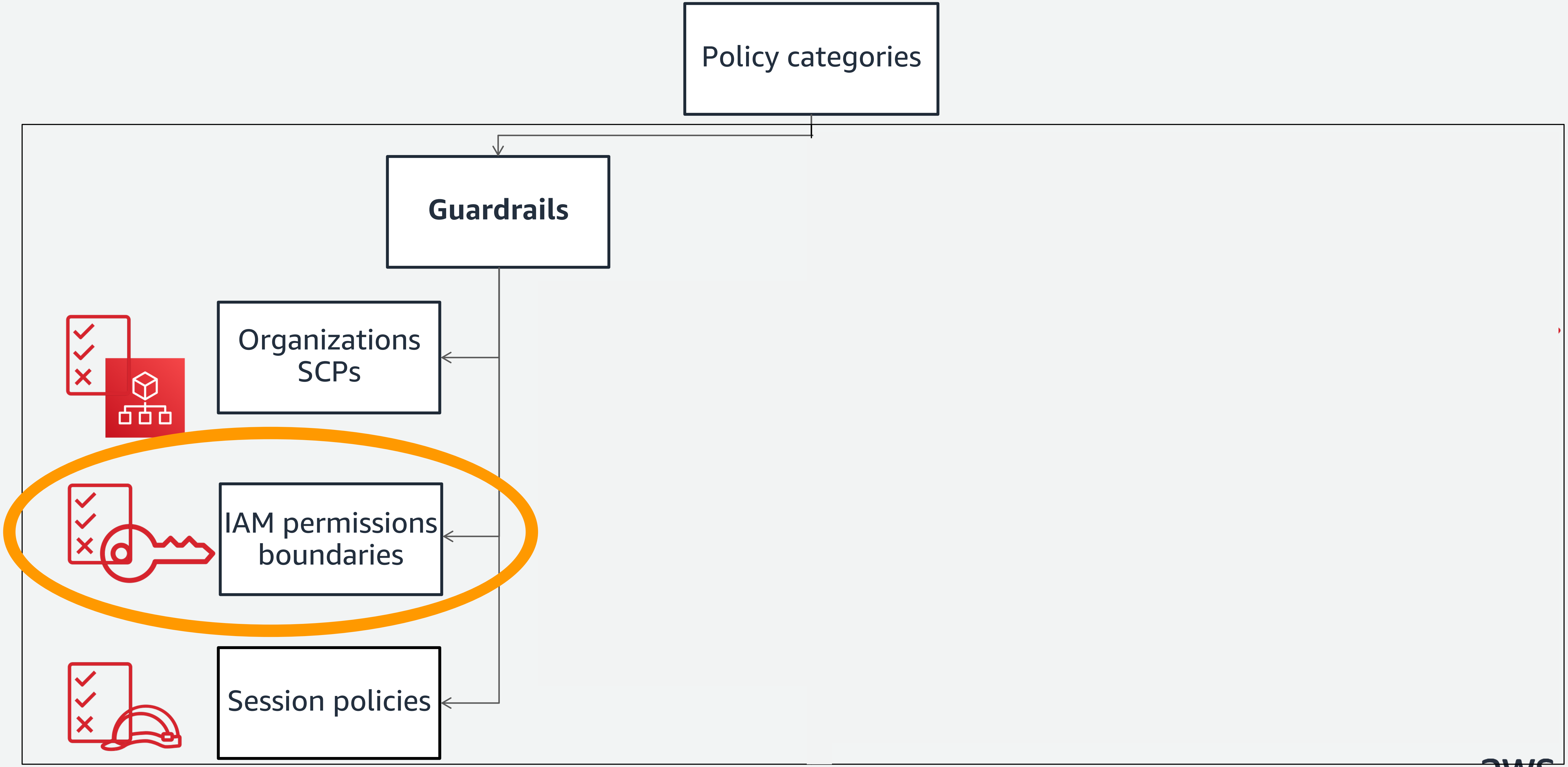
KMS key policy

Identity-based policies

Resource-based policies

Access controls lists (ACLs)

Guardrails



Policy categories

Guardrails

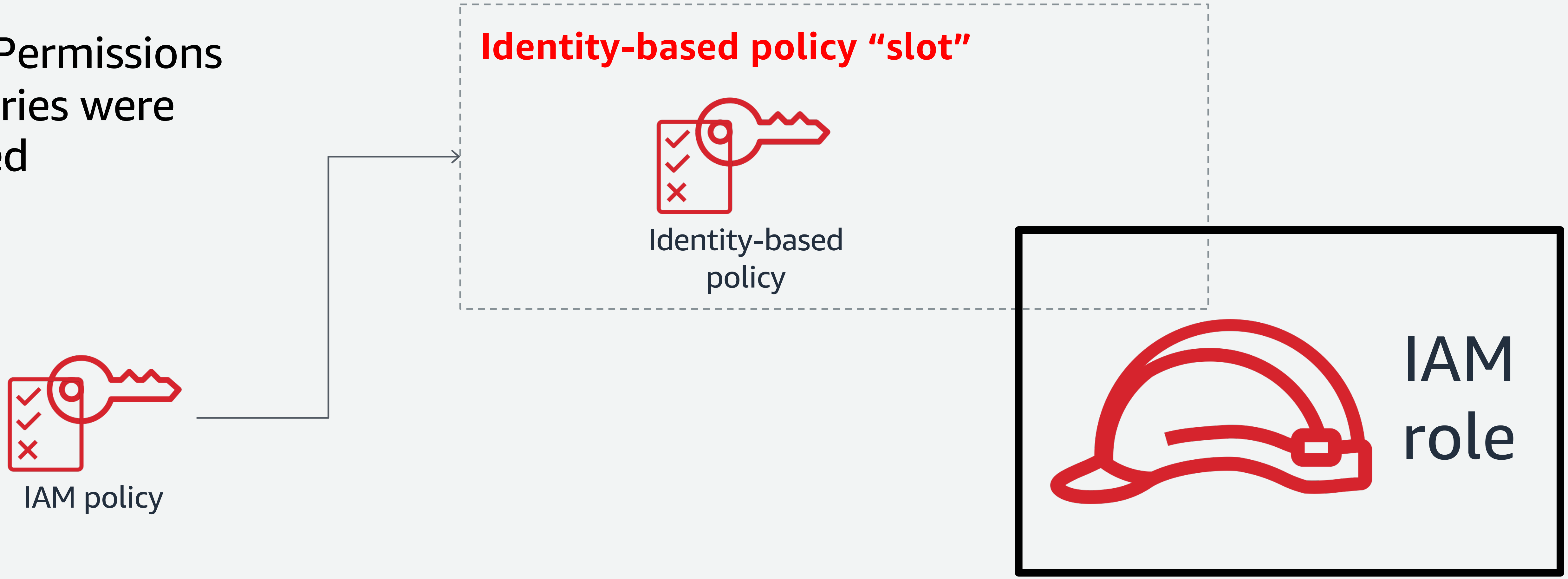
Organizations SCPs

IAM permissions boundaries

Session policies

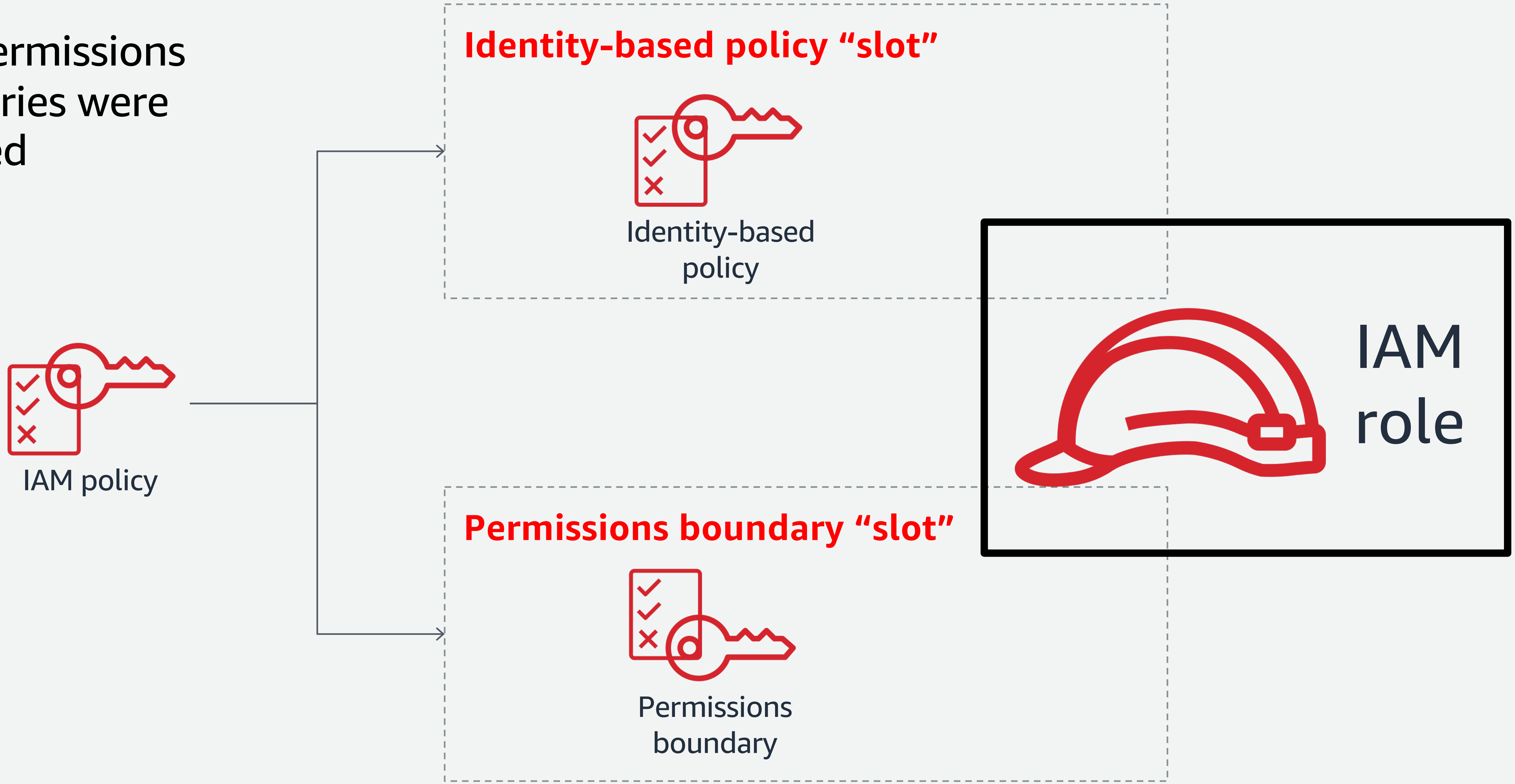
But, it’s just a managed IAM policy right?

Before Permissions Boundaries were launched

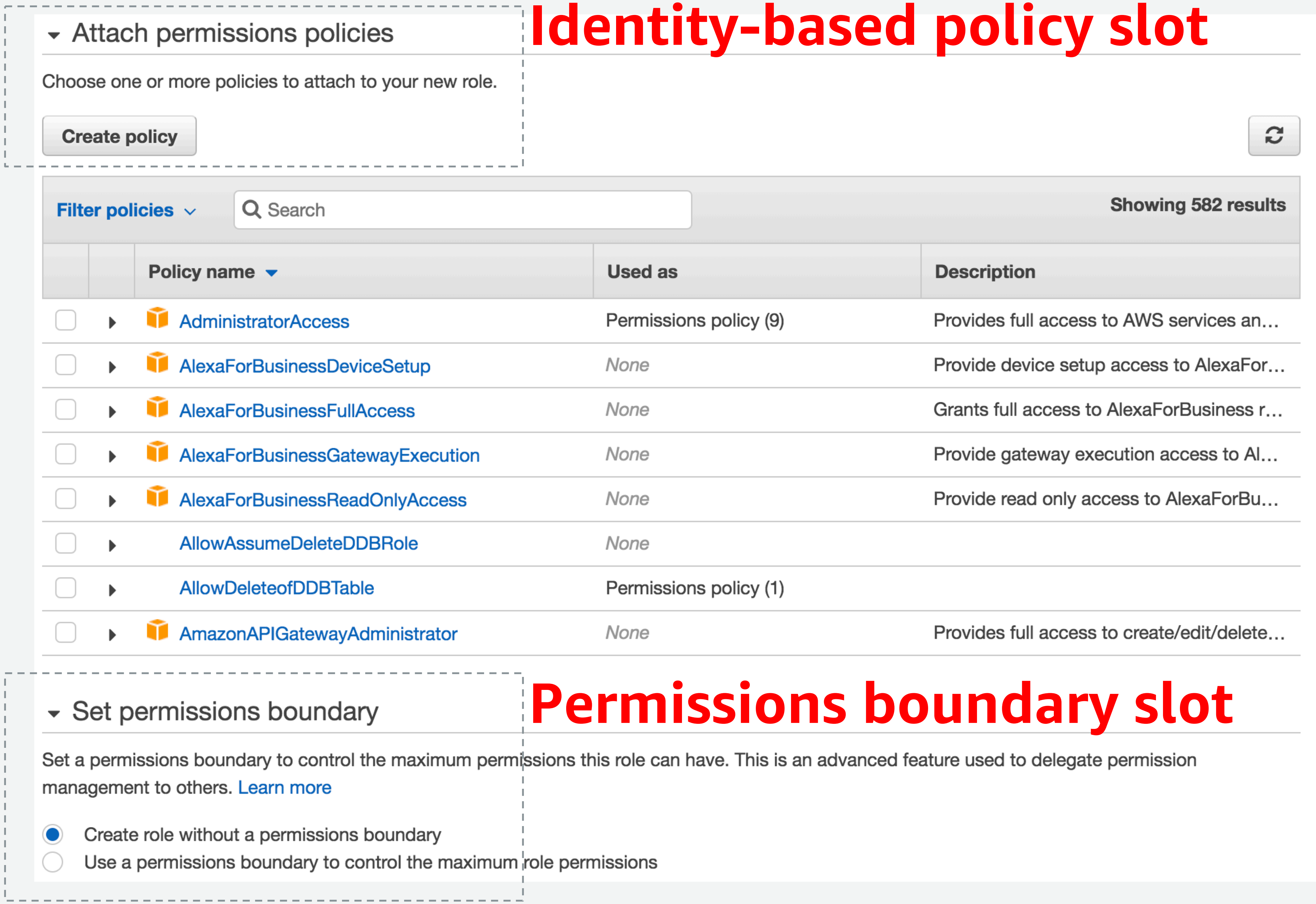


But, it’s just a managed IAM policy right?

After Permissions Boundaries were launched



But, it’s just an IAM policy right?



Identity-based policy slot

▾ Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy

Filter policies ⌄ Search Showing 582 results

		Policy name ▾	Used as	Description
□	▸	AdministratorAccess	Permissions policy (9)	Provides full access to AWS services an...
□	▸	AlexaForBusinessDeviceSetup	*None*	Provide device setup access to AlexaFor...
□	▸	AlexaForBusinessFullAccess	*None*	Grants full access to AlexaForBusiness r...
□	▸	AlexaForBusinessGatewayExecution	*None*	Provide gateway execution access to Al...
□	▸	AlexaForBusinessReadOnlyAccess	*None*	Provide read only access to AlexaForBu...
□	▸	AllowAssumeDeleteDDBRole	*None*	
□	▸	AllowDeleteofDDBTable	Permissions policy (1)	
□	▸	AmazonAPIGatewayAdministrator	*None*	Provides full access to create/edit/delete...

Permissions boundary slot

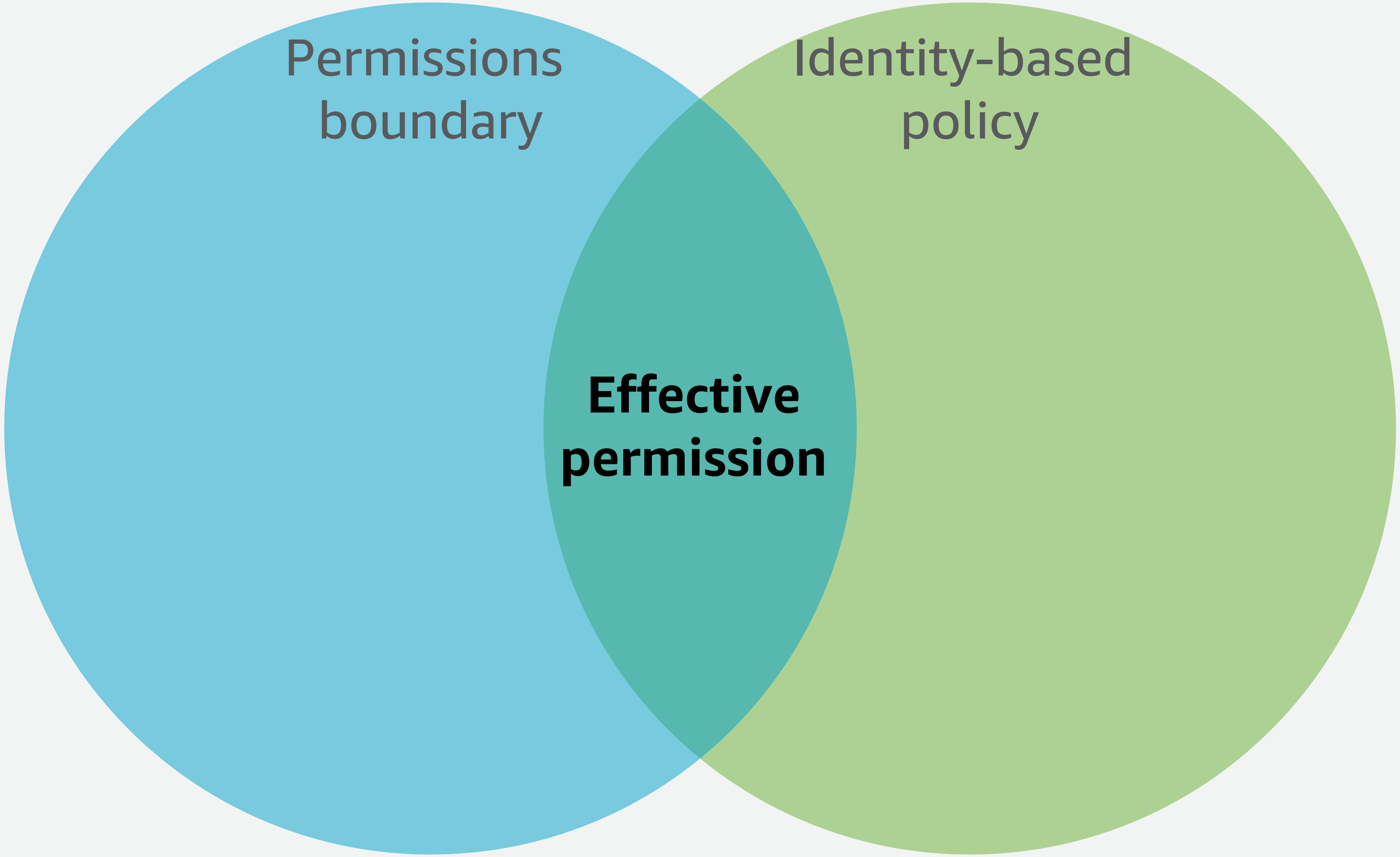
▾ Set permissions boundary

Set a permissions boundary to control the maximum permissions this role can have. This is an advanced feature used to delegate permission management to others. Learn more

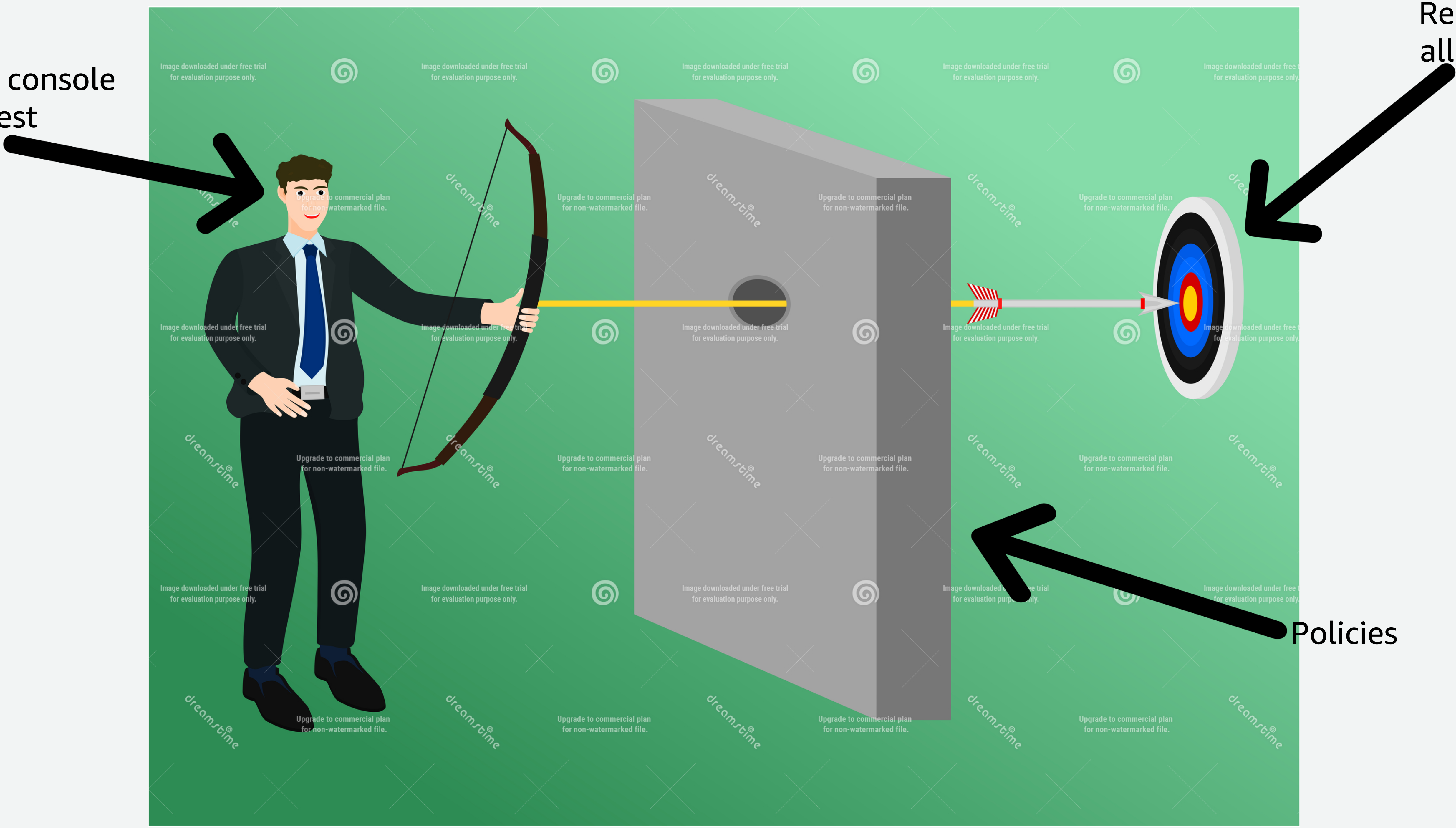
● Create role without a permissions boundary
○ Use a permissions boundary to control the maximum role permissions

Mechanism

Policy evaluation – Venn diagrams



Policy evaluation – the archery analogy

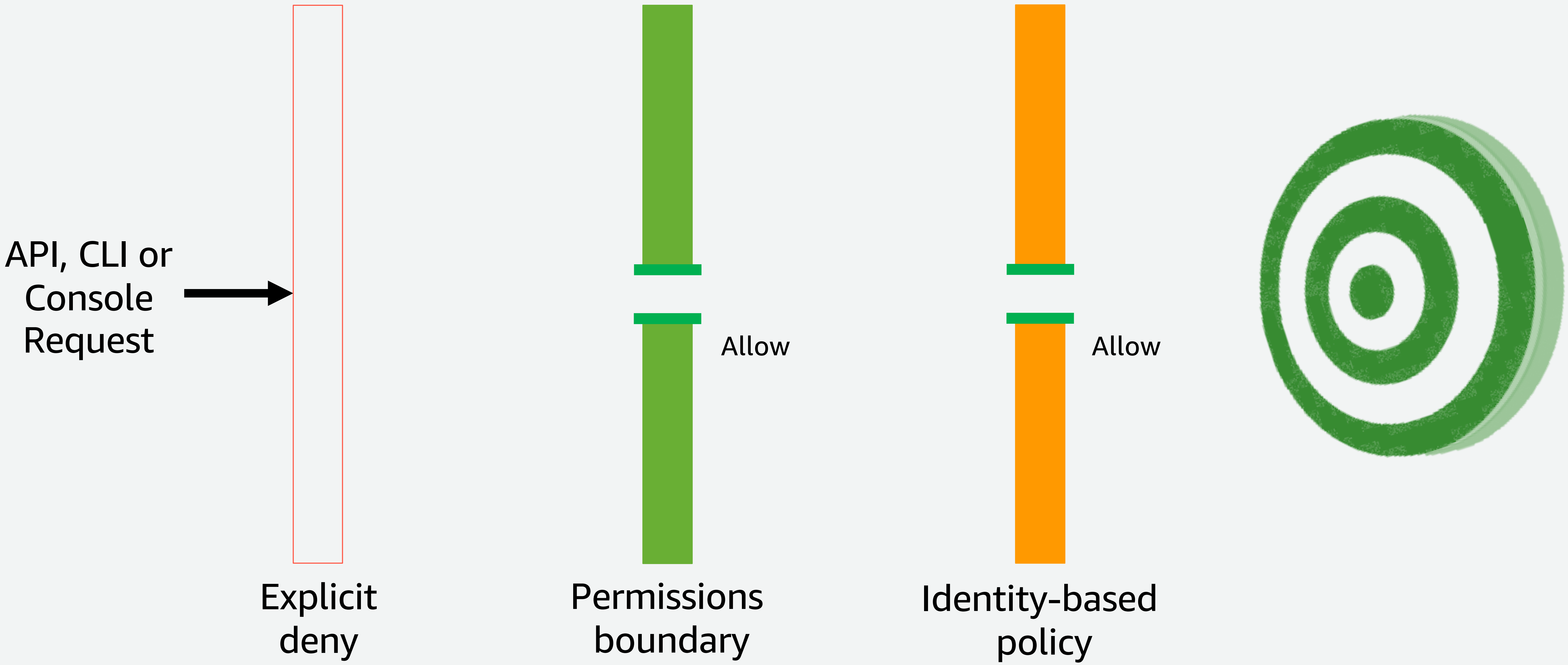


API, CLI or console request

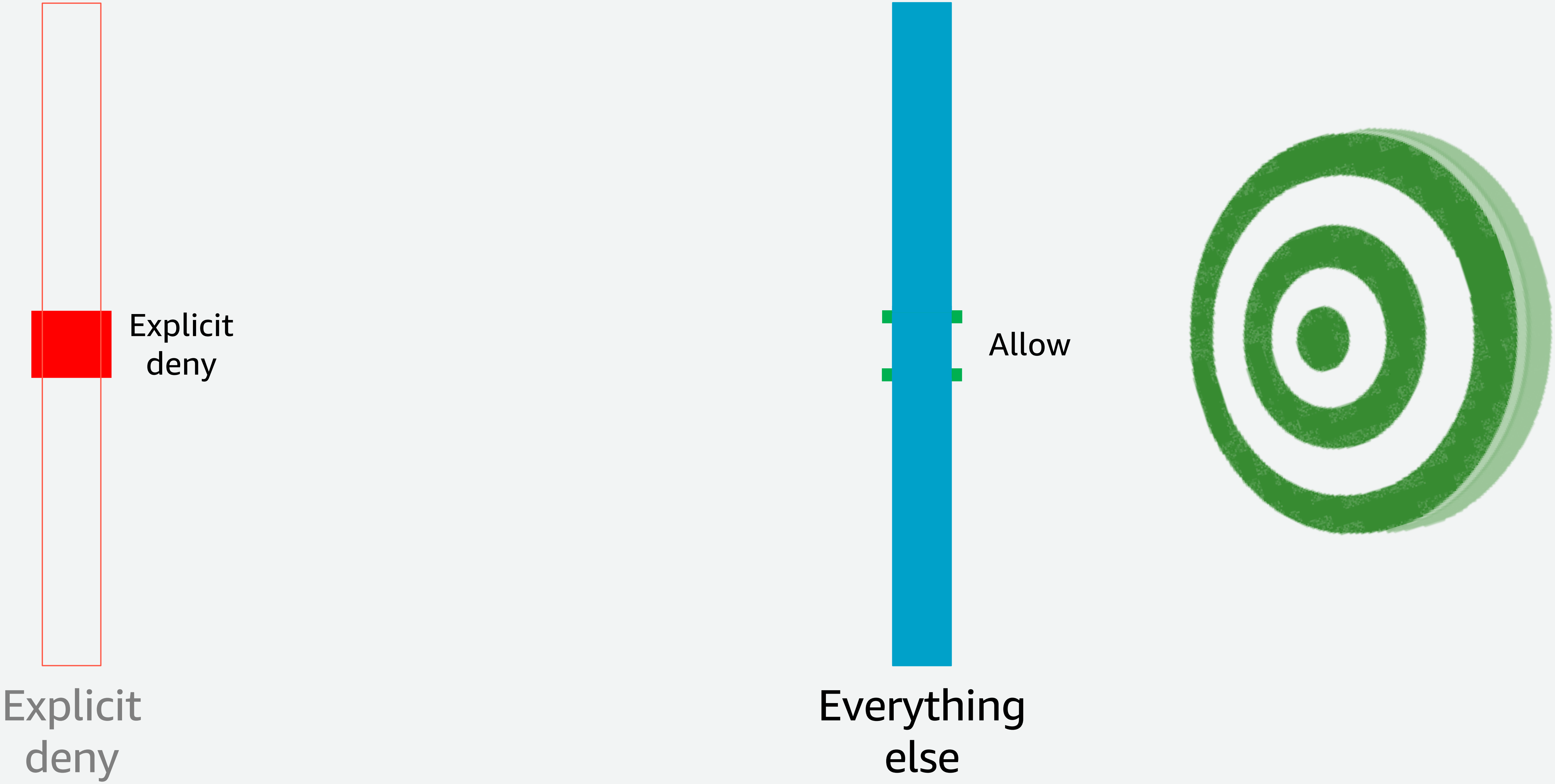
Request allowed

Policies

Trying to hit the target – must go through obstacles



Two types of obstacles



Explicit deny

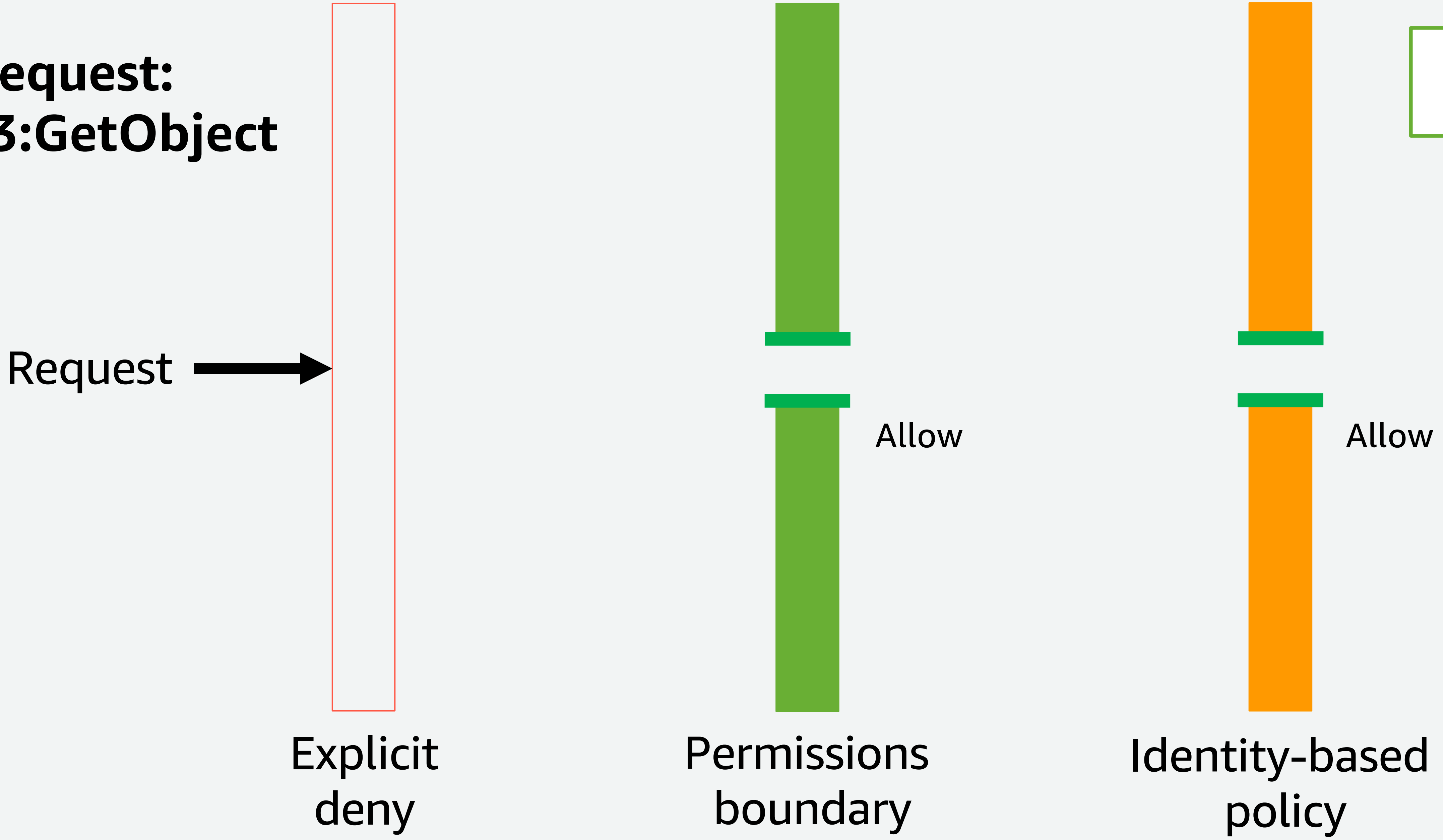
Allow

Explicit deny

Everything else

Allow example

Request: s3:GetObject



Request allowed

Effective permissions – scenario 1

Request: s3:GetObject / bucket name: example1

Permissions boundary

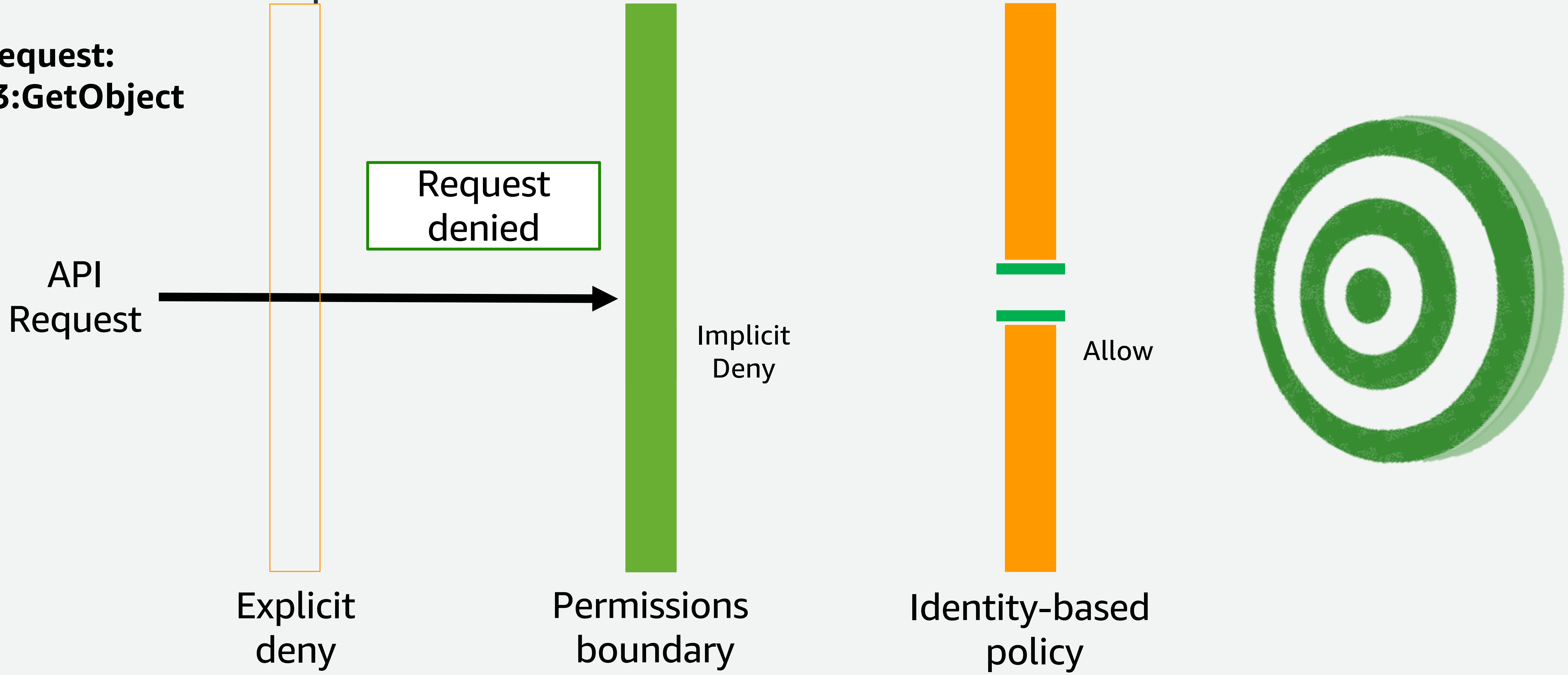
```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream”,
            "logs:PutLogEvents”
       ],
        "Resource": "arn:aws:logs:*:*:*"
     },
}
```

Identity-based policy

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Effect": "Allow",
       "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
        ],
      "Resource": "*"
     },
     {
        "Effect": "Allow",
        ”Action": ["s3:GetObject"],
        "Resource”:"arn:aws:s3:::example1/*"
     }
    ]
}
```

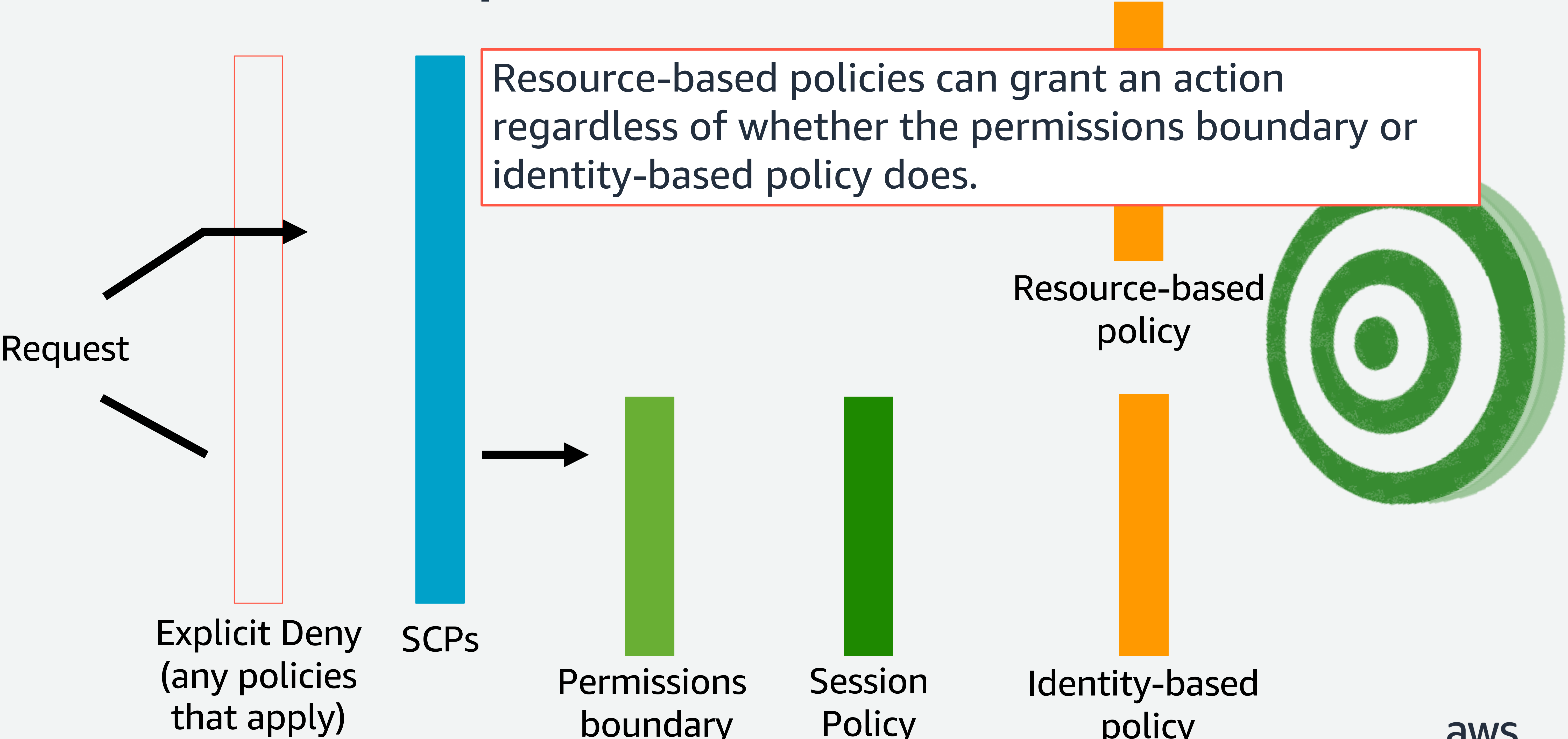
Effective permissions – scenario 1

Request:
s3:GetObject

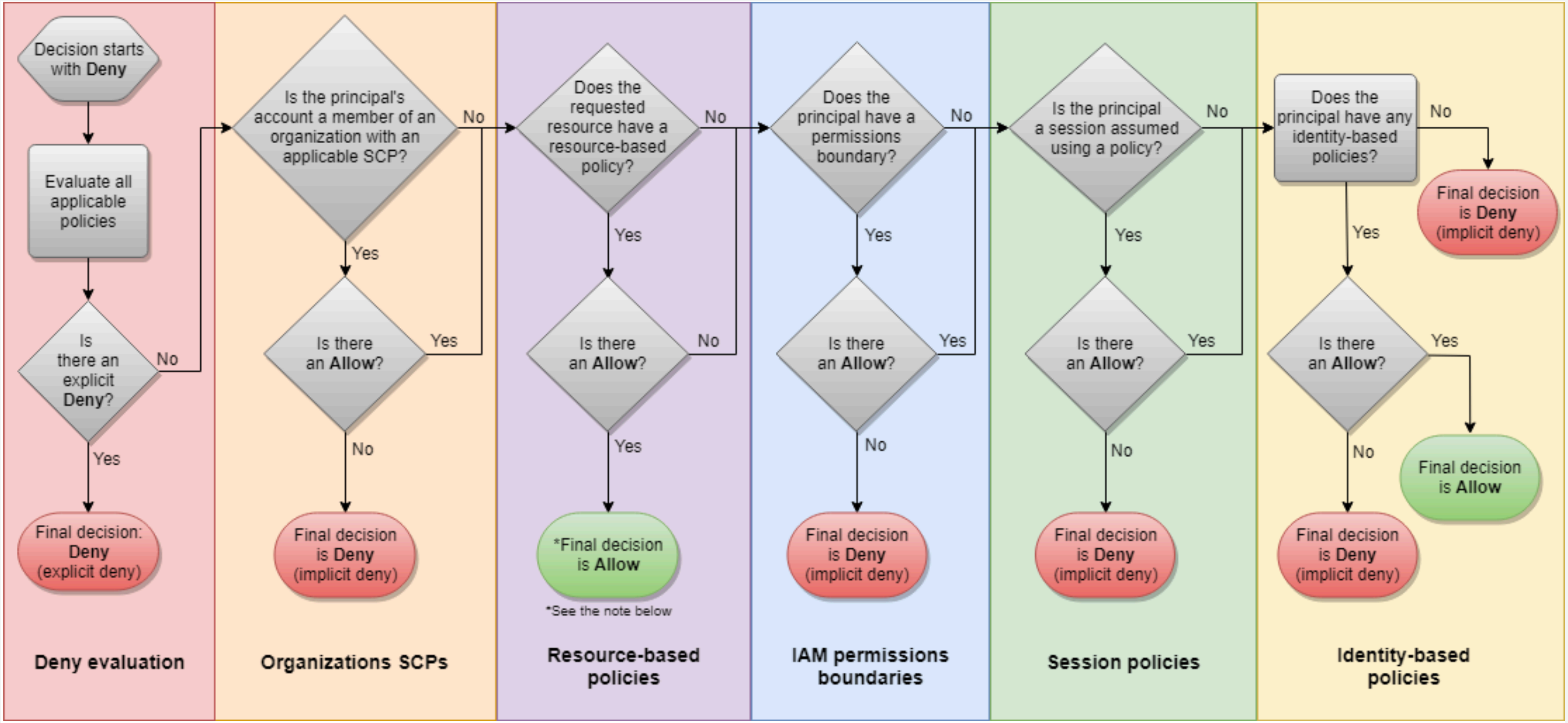


Resource based policies – intra account

Resource-based policies can grant an action regardless of whether the permissions boundary or identity-based policy does.



Policy Evaluation Logic



Decision starts with **Deny**

Evaluate all applicable policies

Is there an explicit **Deny**?

No

Yes

Final decision: **Deny** (explicit deny)

Deny evaluation

Is the principal's account a member of an organization with an applicable SCP?

No

Yes

Is there an **Allow**?

Yes

No

Final decision is **Deny** (implicit deny)

Organizations SCPs

Does the requested resource have a resource-based policy?

No

Yes

Is there an **Allow**?

No

Yes

*Final decision is **Allow**

*See the note below

Resource-based policies

Does the principal have a permissions boundary?

No

Yes

Is there an **Allow**?

Yes

No

Final decision is **Deny** (implicit deny)

IAM permissions boundaries

Is the principal a session assumed using a policy?

No

Yes

Is there an **Allow**?

Yes

No

Final decision is **Deny** (implicit deny)

Session policies

Does the principal have any identity-based policies?

No

Final decision is **Deny** (implicit deny)

Yes

Is there an **Allow**?

Yes

Final decision is **Allow**

No

Final decision is **Deny** (implicit deny)

Identity-based policies

Resource restrictions

Resource Restrictions

- Goal: create a “walled garden” for the delegated admins
- Important since not all actions support the permissions boundary condition
- Also allows different teams to safely do delegated permissions management in the same account
- Pathing preferred (requires CLI). “Naming restrictions” can also be used. **Tags are also an option.**

Resource Restrictions - paths

Basic path example:

arn:aws:iam::123456789012:role/**webadmins/????**

Naming example:

arn:aws:iam::123456789012:role/**webadmins***

Resource Restrictions - paths

Role: arn:aws:iam::123456789012:role/**webadmins**

Role with a path: arn:aws:iam::123456789012:role/**namer/webadmins**

Role with paths: arn:aws:iam::123456789012:role/**namer/dept1/webadmins**

Permission:

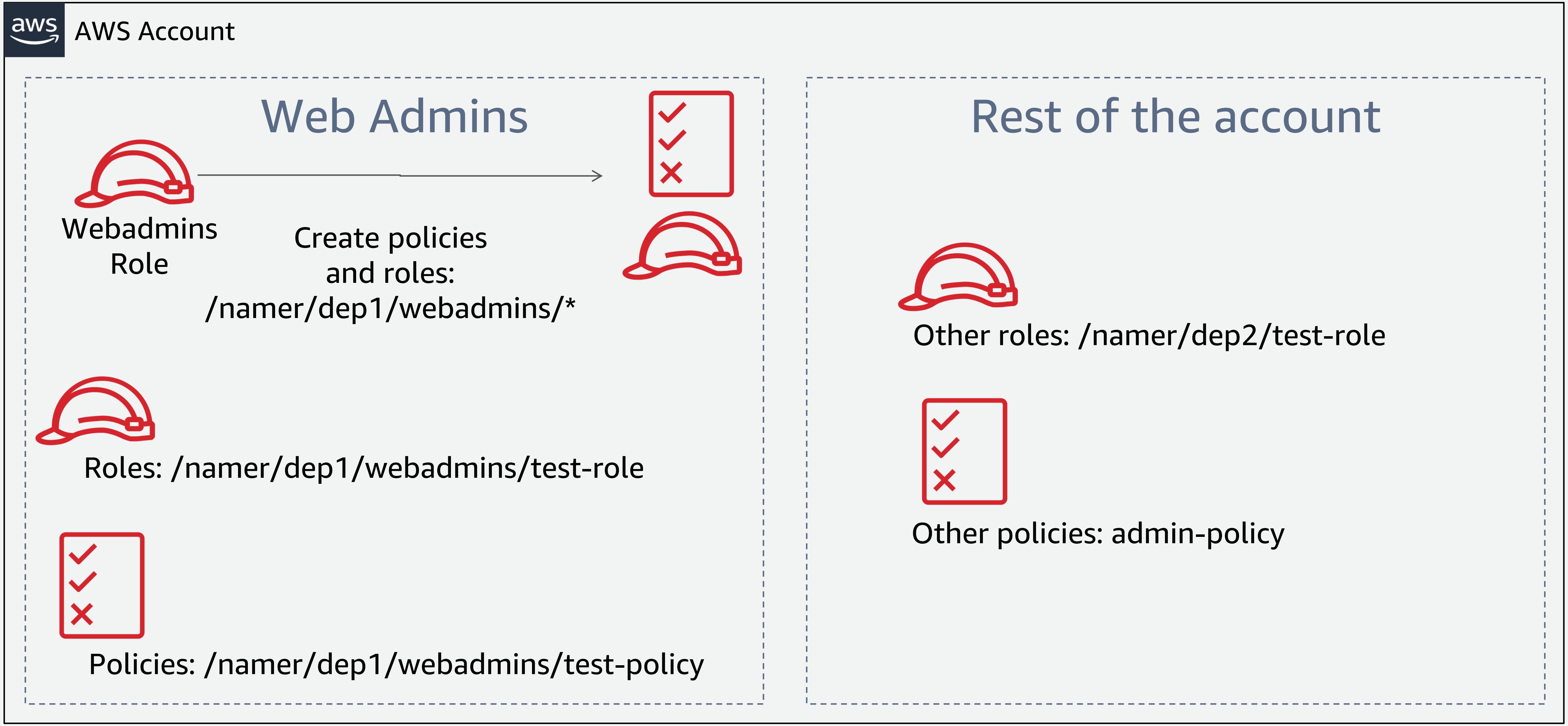
    "Effect": "Allow",
    "Action": "iam:DeleteRole”,
    "Resource": "arn:aws:iam::123456789012:/namer/dept1/*”

    or "Resource": "arn:aws:iam::123456789012:/namer/*”

Command:

```
    aws iam create-role --role-name webadmin --path /namer/dept1/ --assume-
role-policy-document file://policydoc
```

Pathing Walled Garden



AWS Account

Web Admins

Webadmins Role

Create policies and roles: /namer/dep1/webadmins/*

Roles: /namer/dep1/webadmins/test-role

Policies: /namer/dep1/webadmins/test-policy

Rest of the account

Other roles: /namer/dep2/test-role

Other policies: admin-policy

Condition context key suport

- AttachRolePolicy
- AttachUserPolicy
- CreateRole
- CreateUser
- DeleteRolePermissionsBoundary
- DeleteUserPermissionsBoundary
- DeleteRolePolicy
- DeleteUserPolicy
- DetachRolePolicy
- DetachUserPolicy
- PutRolePermissionsBoundary
- PutUserPermissionsBoundary
- PutRolePolicy
- PutUserPolicy

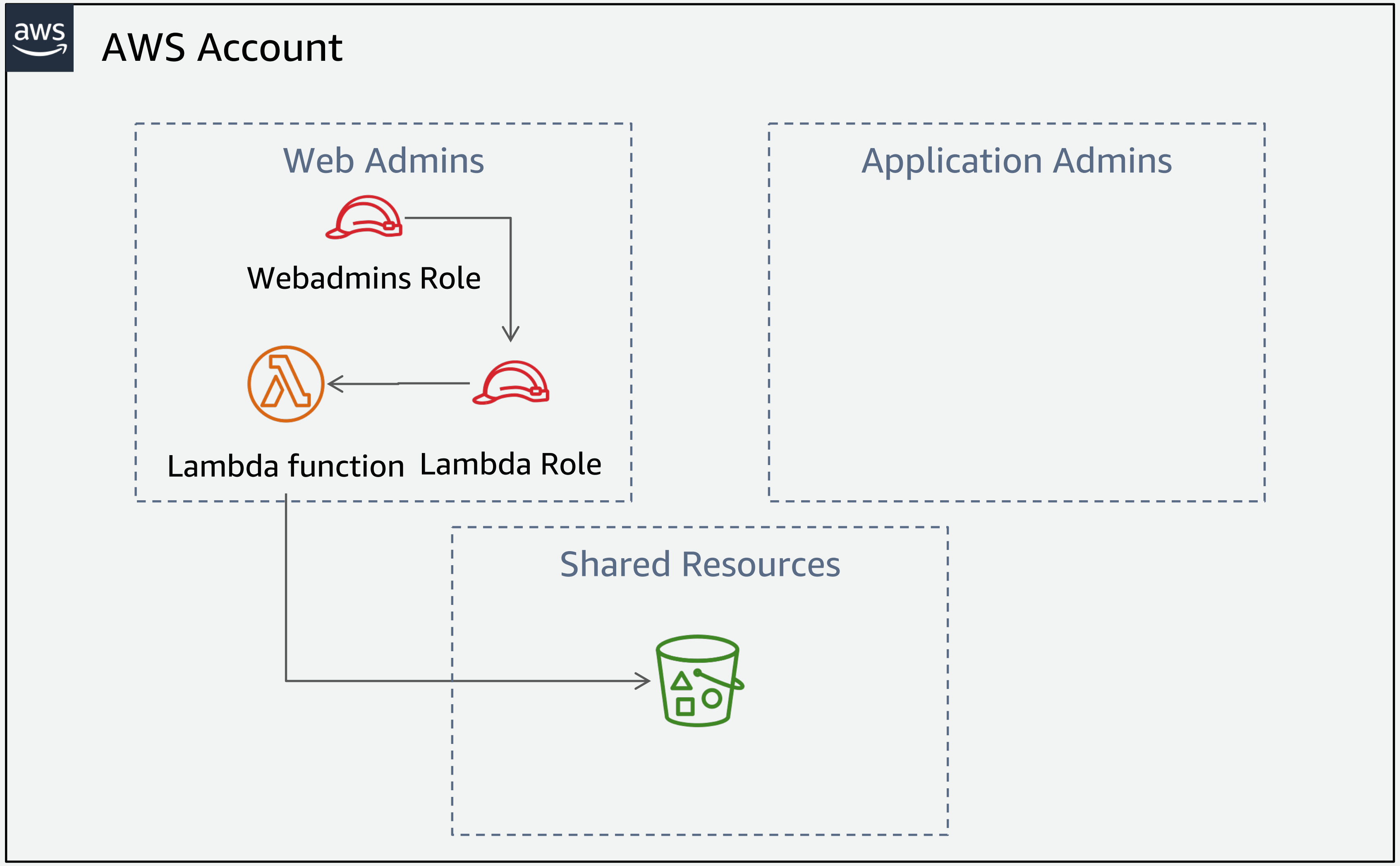
Q & A

End of presentation questions

- What is the condition context key used for permissions boundaries?
- What are some of the advantages of using pathing for policies, users and roles?
- What are some permissions boundaries use cases?

Hands on

Workshop setup



AWS Account

Web Admins

Webadmins Role

Lambda function

Lambda Role

Application Admins

Shared Resources

Permissions Boundaries Workshop

Build Phase (60 min)

https://bit.ly/2CMjqmh

Read through the **Overview**, then click on **Build Phase**:
Follow instructions under "**Click here if you are *using your own AWS account …***"
https://identity-round-robin.awssecworkshops.com/permission-boundaries-advanced/

Permissions Boundaries Workshop

Verify Phase (15 min)

https://bit.ly/2CMjqmh

Click on **Verify Phase:**
https://identity-round-robin.awssecworkshops.com/permission-boundaries-advanced/

Final Q & A

End of workshop questions

- What is the risk of implementing permissions boundaries without resource restrictions?

- What do you attach the permissions boundary to?

- How does a permissions boundary differ from an IAM policy?

Summary

Safely delegate permission management.

Free up developers (get out of their way) and do so securely.

Also allow multiple teams in the same account to do permission management.