



Identity Round Robin Workshop

Permissions Boundaries

Name,
Title

Analogy – giving the keys to your teenager

- Car keys give a lot of power: drive fast, drive anywhere and even drink and drive.
- You can set rules: don't speed, don't go beyond 20 mile range, etc, but that is trust based.
- You can only use detect controls to verify compliance (check odometer, see if they got a speeding ticket or got into an accident.)



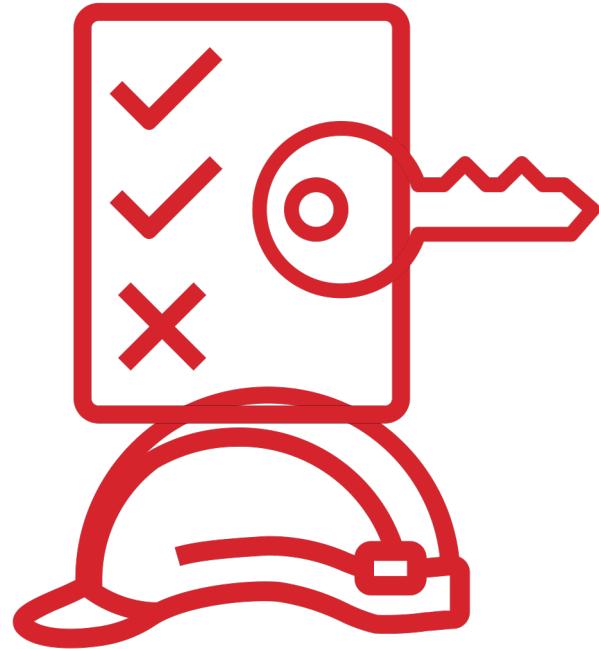
Analogy – giving the keys to your teenager

- Some auto makes have special keys and programming options that allows you to let them drive but they are restricted by your settings.
- Their ability in the car (drive fast, blast the radio or even spin the tires) is the intersection between their desire and your settings.



Analogy – giving the keys to a developer

- In the same way you can give the keys to a developer (ability to create user or roles) and all the power that comes with that.
- The developer can attach an identity-based policy with full admin rights (their desire) to the role but they must also attach a permissions boundary (like the auto restriction settings).
- Effective permission of the role is the intersection of the two.



Agenda

- Intro & basics
- Demo
- Permission categories
- Permissions boundary mechanism
- Resource restrictions

What are permissions boundaries?

Controls the **maximum permissions** that a user and/or role can have, but does not itself provide any permissions (necessary but not sufficient).

Mechanism to **delegate the permission to create users and/or roles** while **preventing privilege escalation** or unnecessarily broad permissions.

Before and After Permissions Boundaries

Before

- Certain IAM policy actions (e.g. PutUserPolicy, AttachRolePolicy) are **essentially full admin-like permissions**.
- Doing any form of self-service permissions management **was non-trivial**.

Now

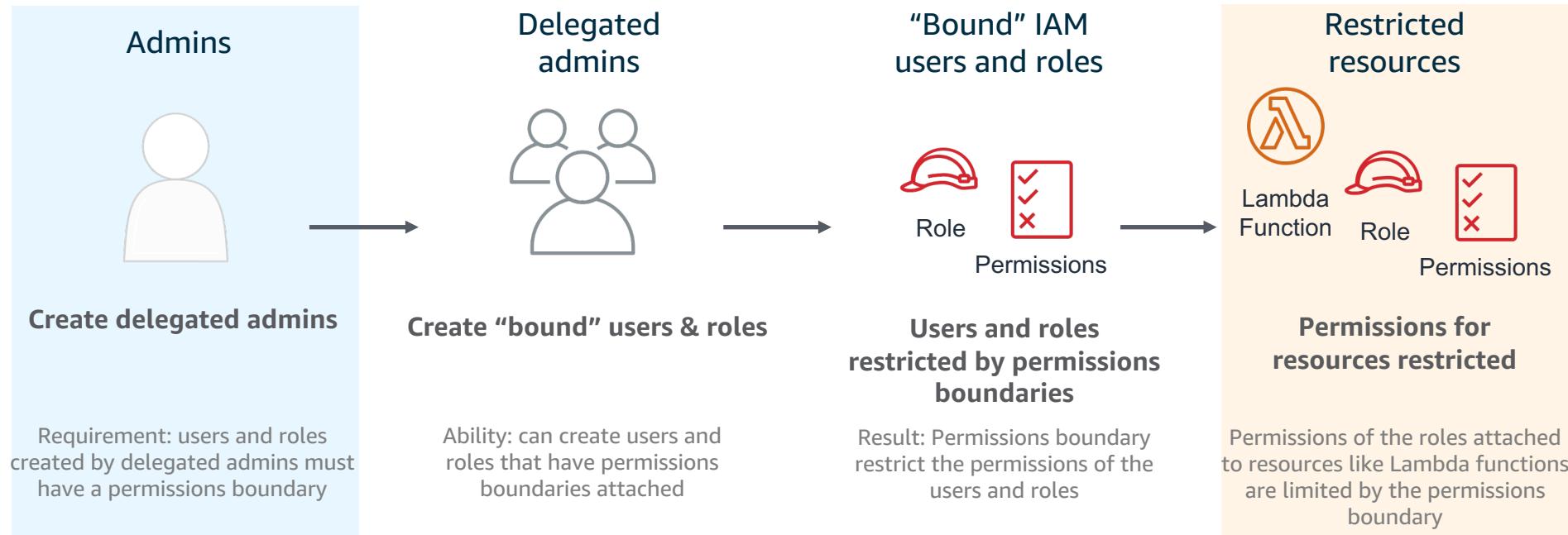
- Administrators can grant these full admin-like permissions, but **specify a “permissions boundary.”**
- Allows developers **to create principals** for their applications and attach policies, but only **within the boundary**.

Use cases

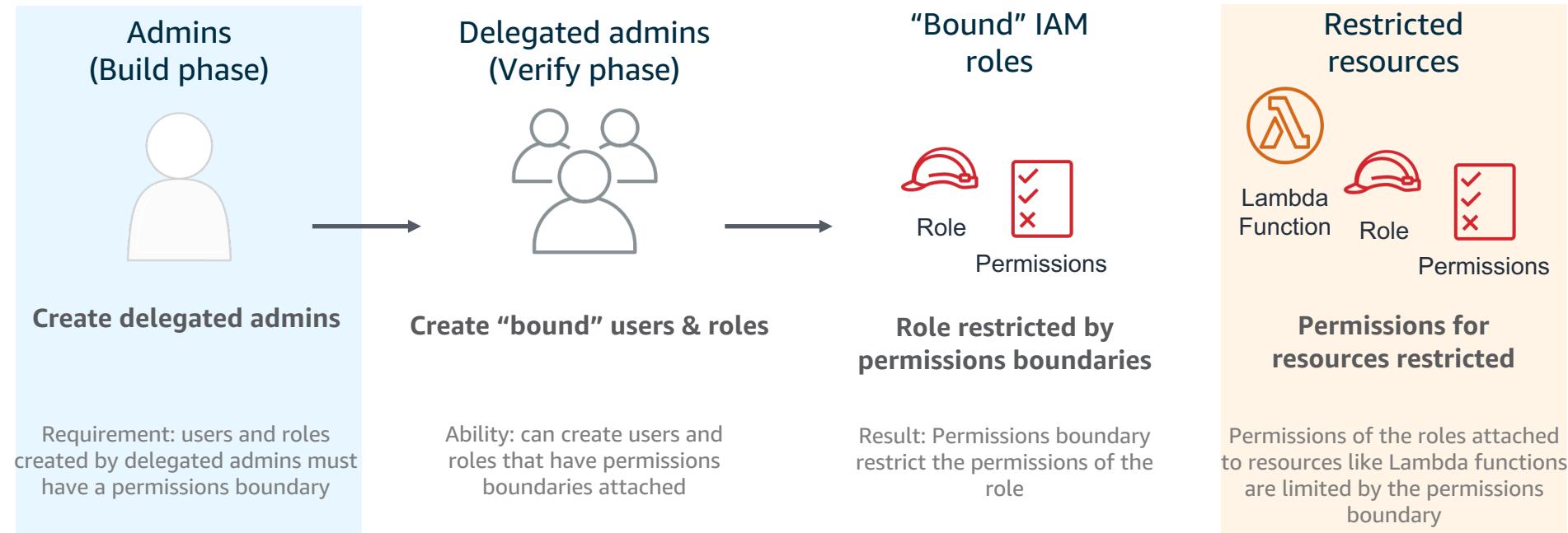
- Developers that need to create roles for Lambda functions
- Application owners that need to create roles for EC2 instances
- Admins that need to be able to create users for particular use cases
- Any others?

Permissions boundary basics

Permissions boundaries – workflow



Permissions boundaries – workflow



An IAM condition context key

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

... applied to principal creation actions (users and roles)

```
"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

Create role for a Lambda function

Step 1: Create role and attach permissions boundary

```
$ aws iam create-role --role-name roleforlambda  
--assume-role-policy-document file://Role_Trust_Policy_Text.json  
--permissions-boundary arn:aws:iam::<ACCOUNT_NUMBER>:policy/department_a/boundary_1
```

Step 2: Create identity-based policy

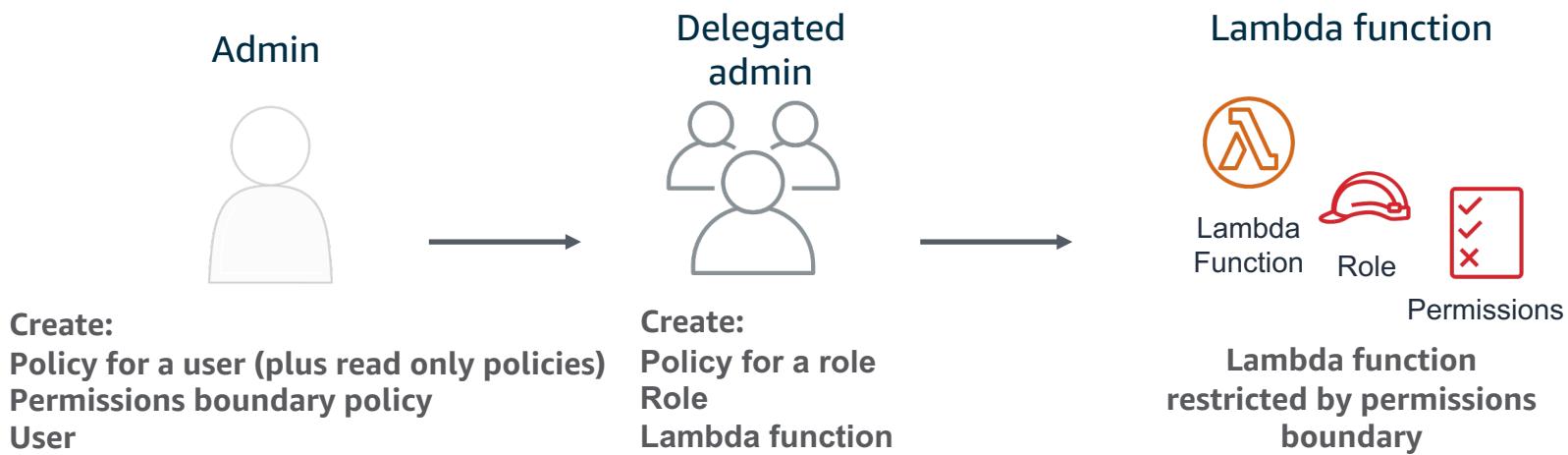
No change

Step 3: Attach identity-based policy

No change

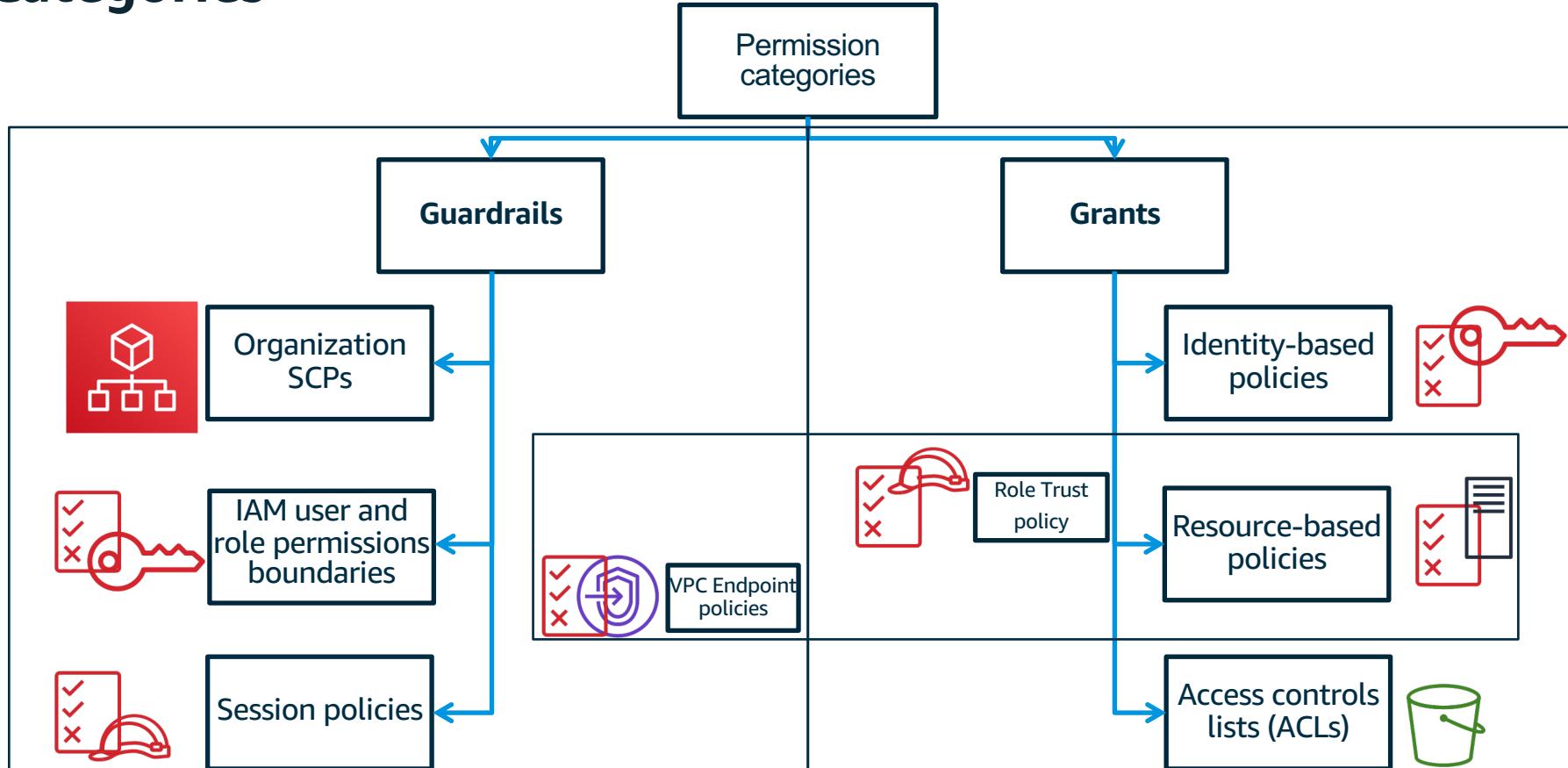
Demo

- **User requirements:**
 - Lambda function that reads from an S3 bucket
 - Lambda function must have an IAM role to access the bucket
 - Role must be created with the correct permissions
- **Company requirements:**
 - Policies attached to the role must not allow privilege escalation or unneeded permissions
 - Don't get in the way of the user

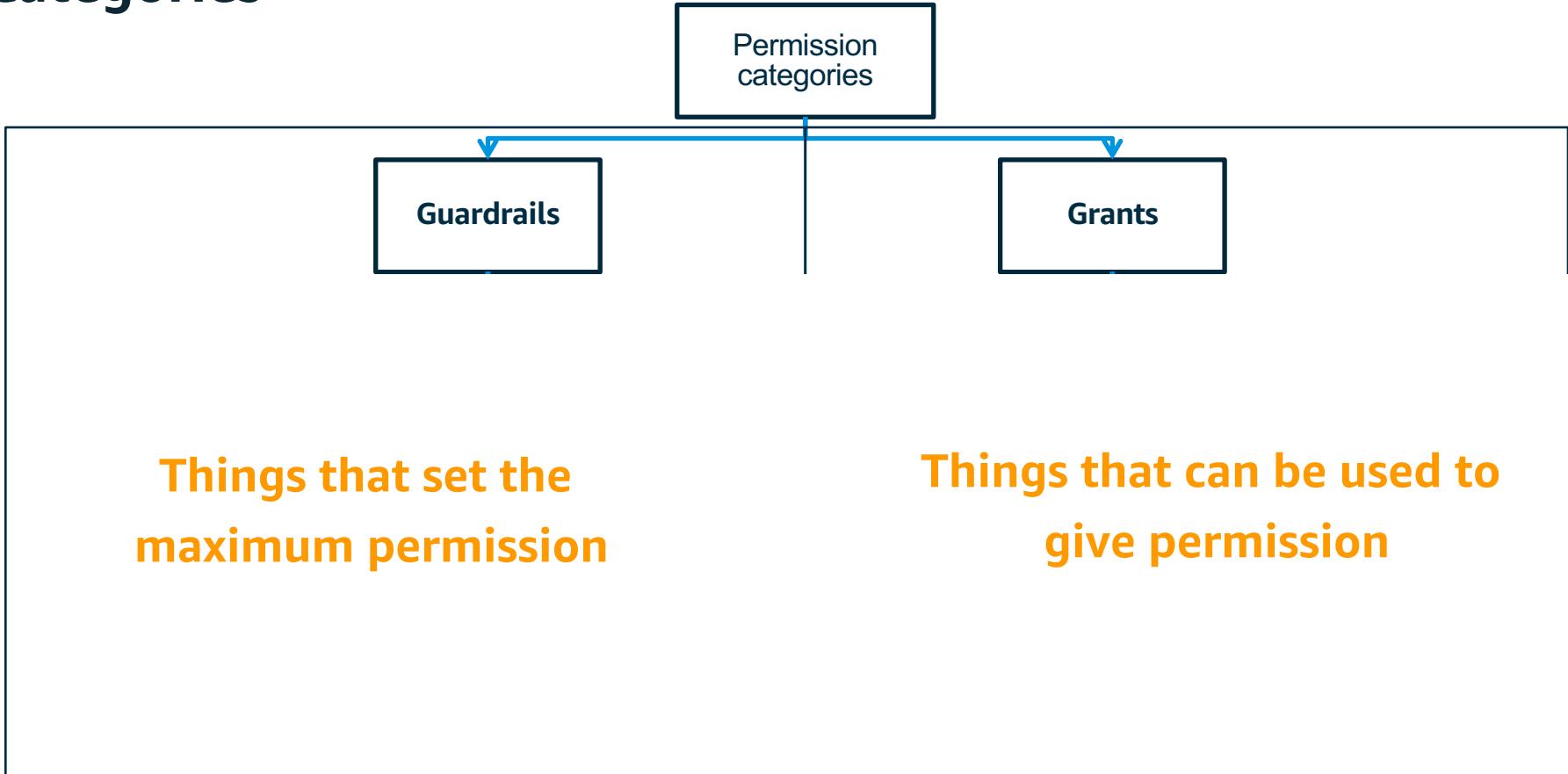


Permission categories

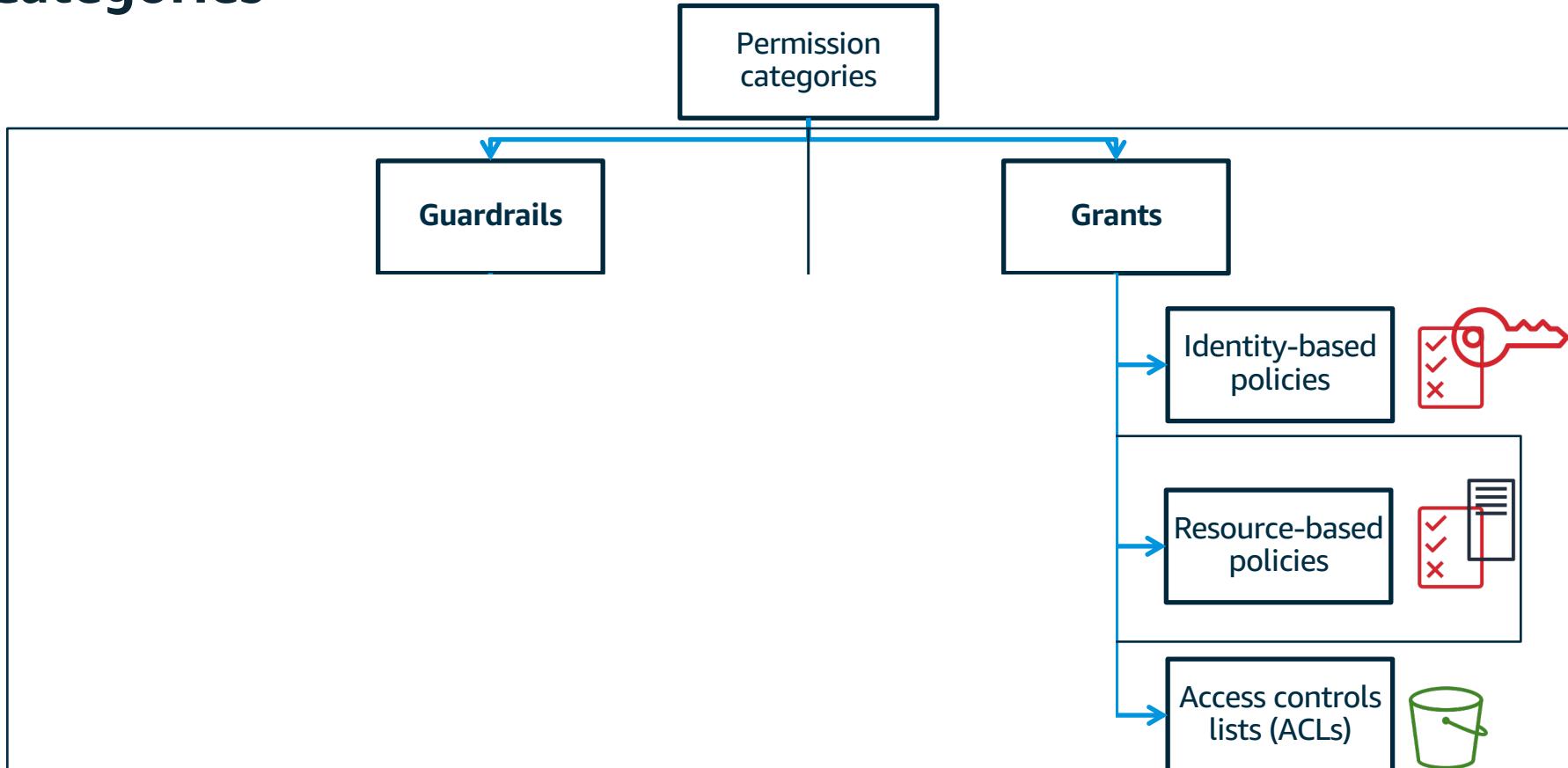
Categories



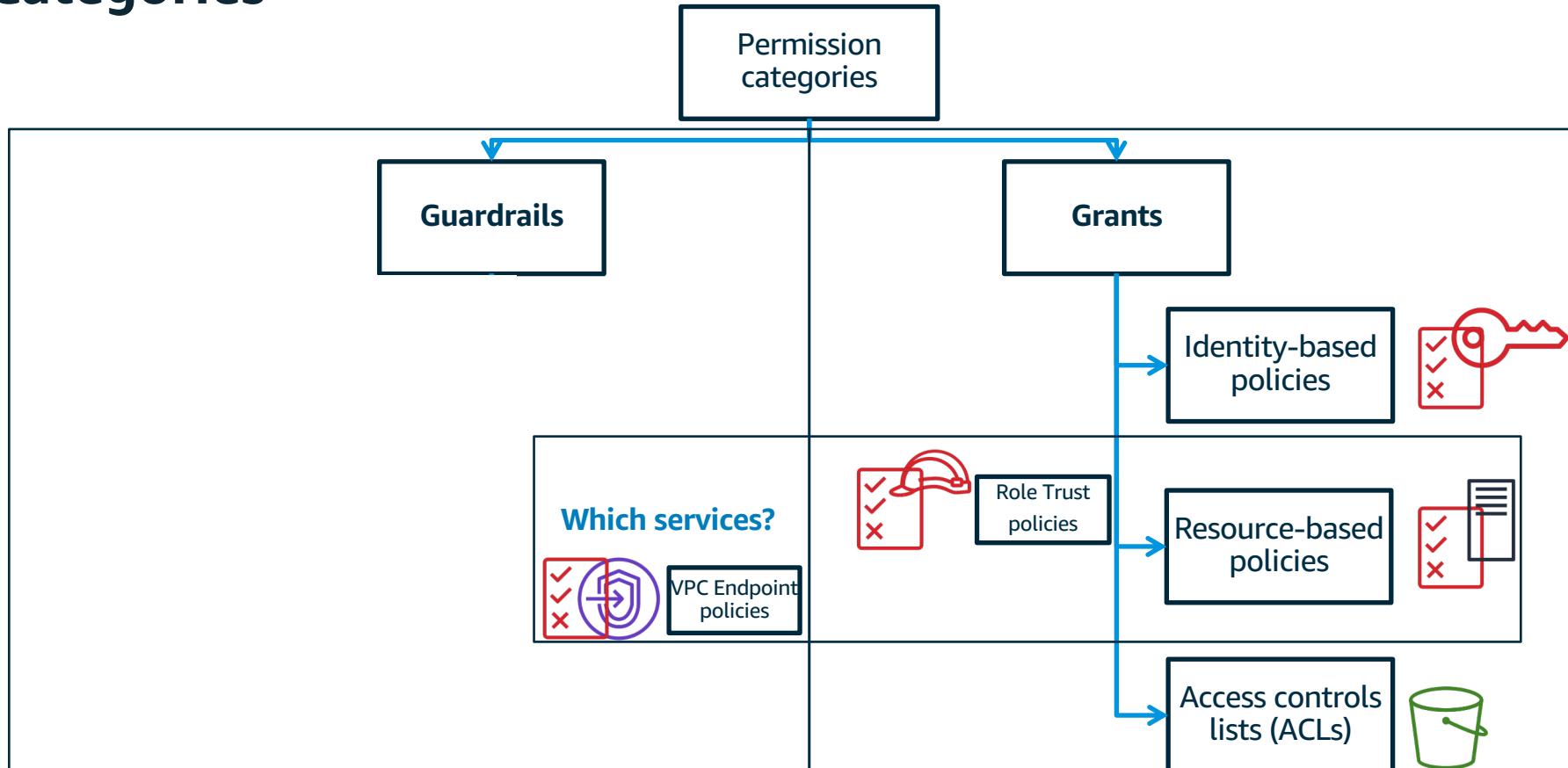
Categories



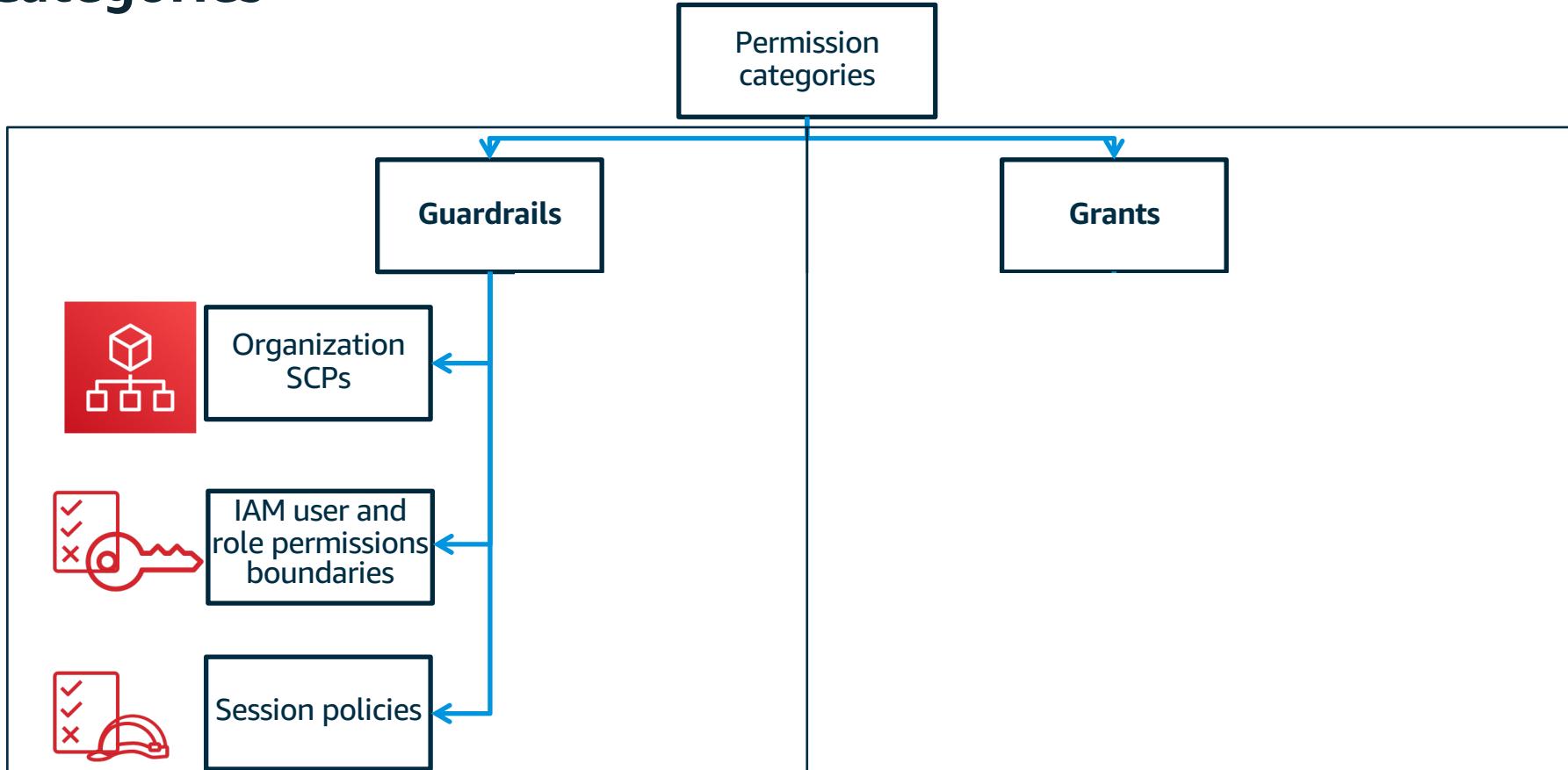
Categories



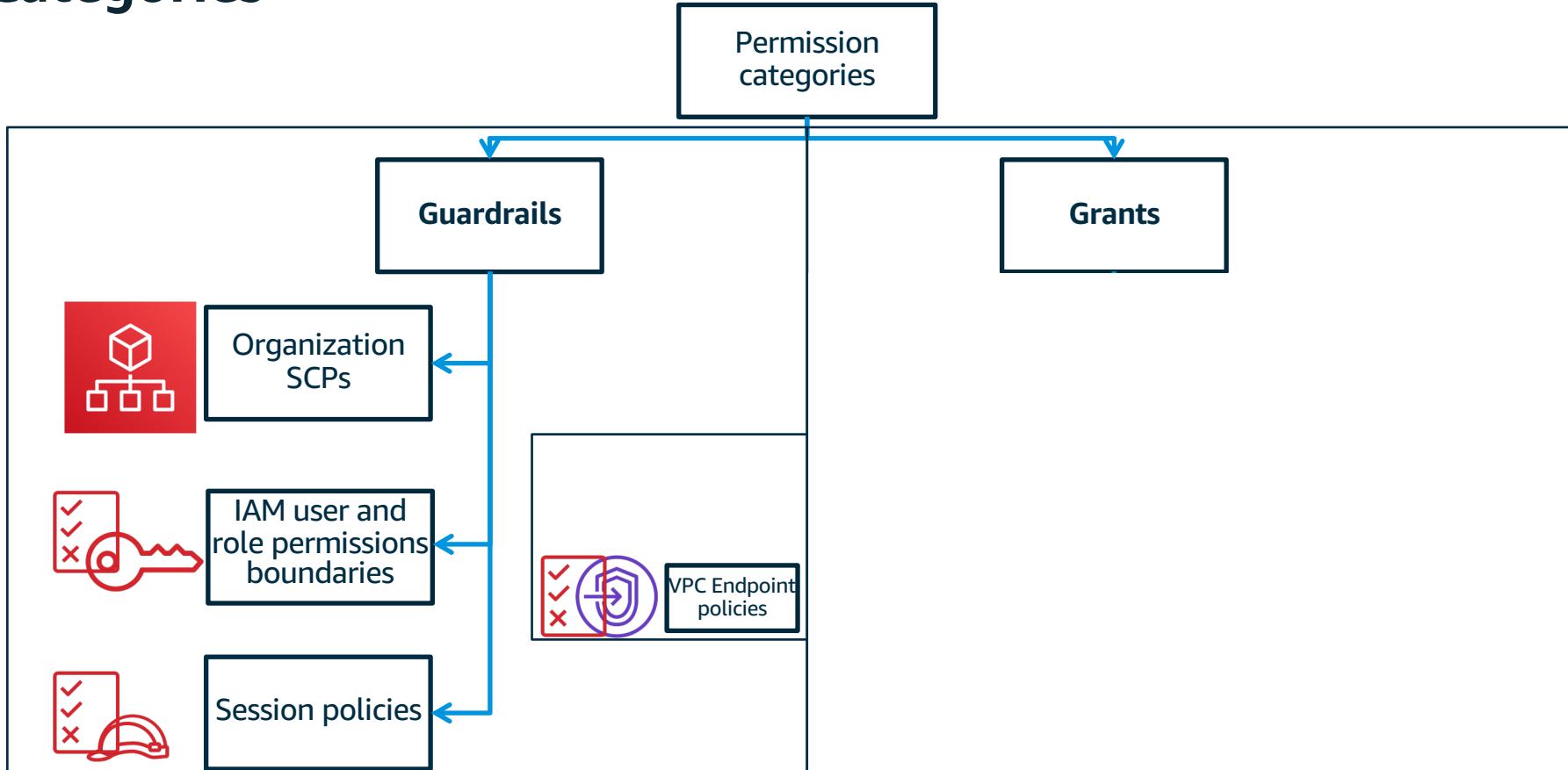
Categories



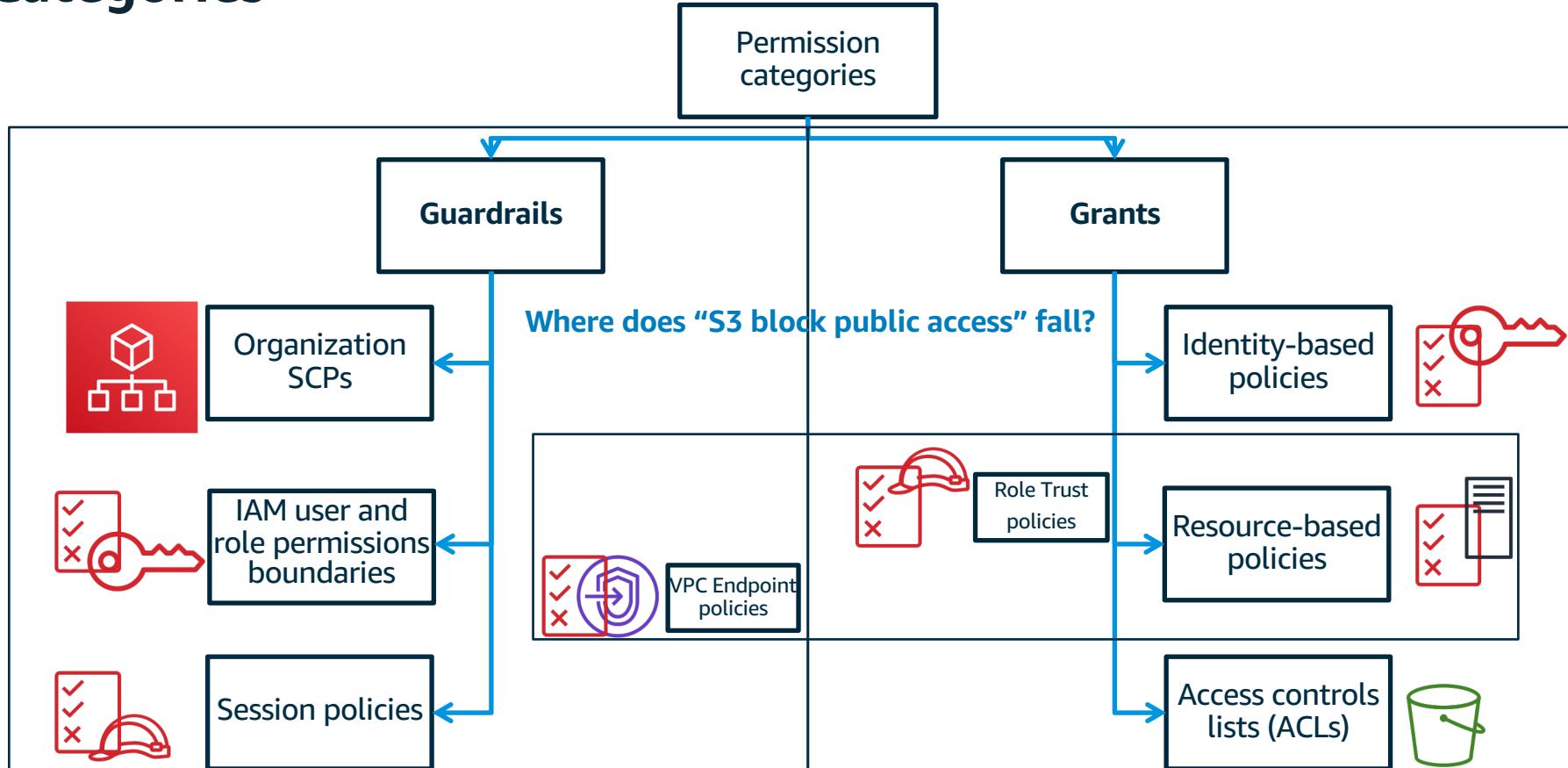
Categories



Categories



Categories



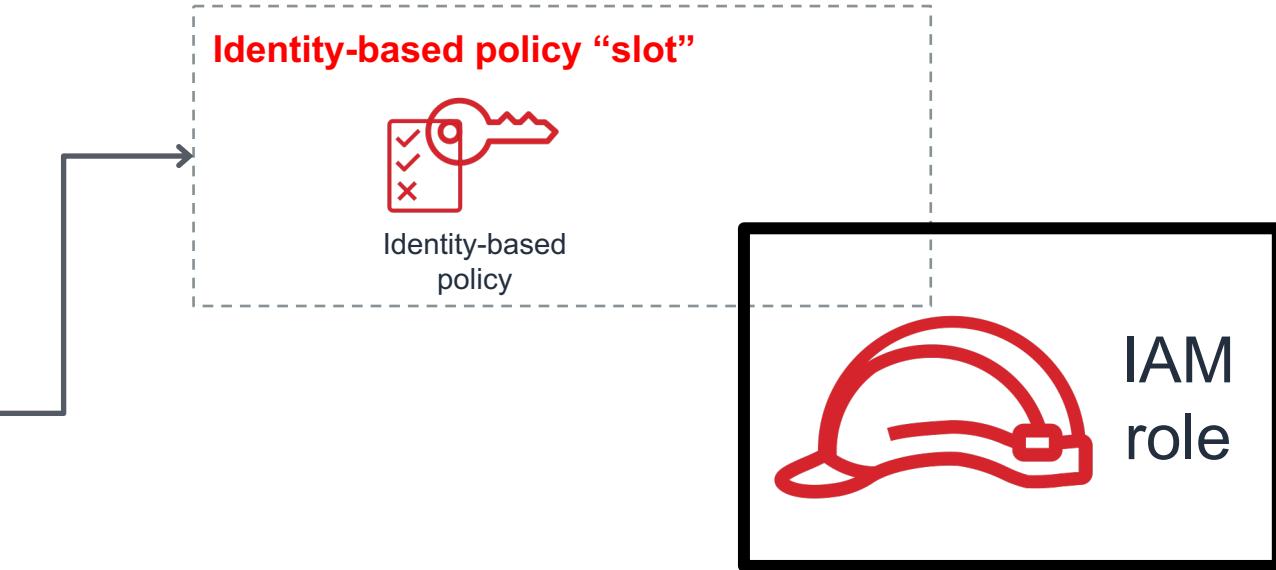
But, it's just a an IAM policy right?

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

Before Permissions
Boundaries were
launched

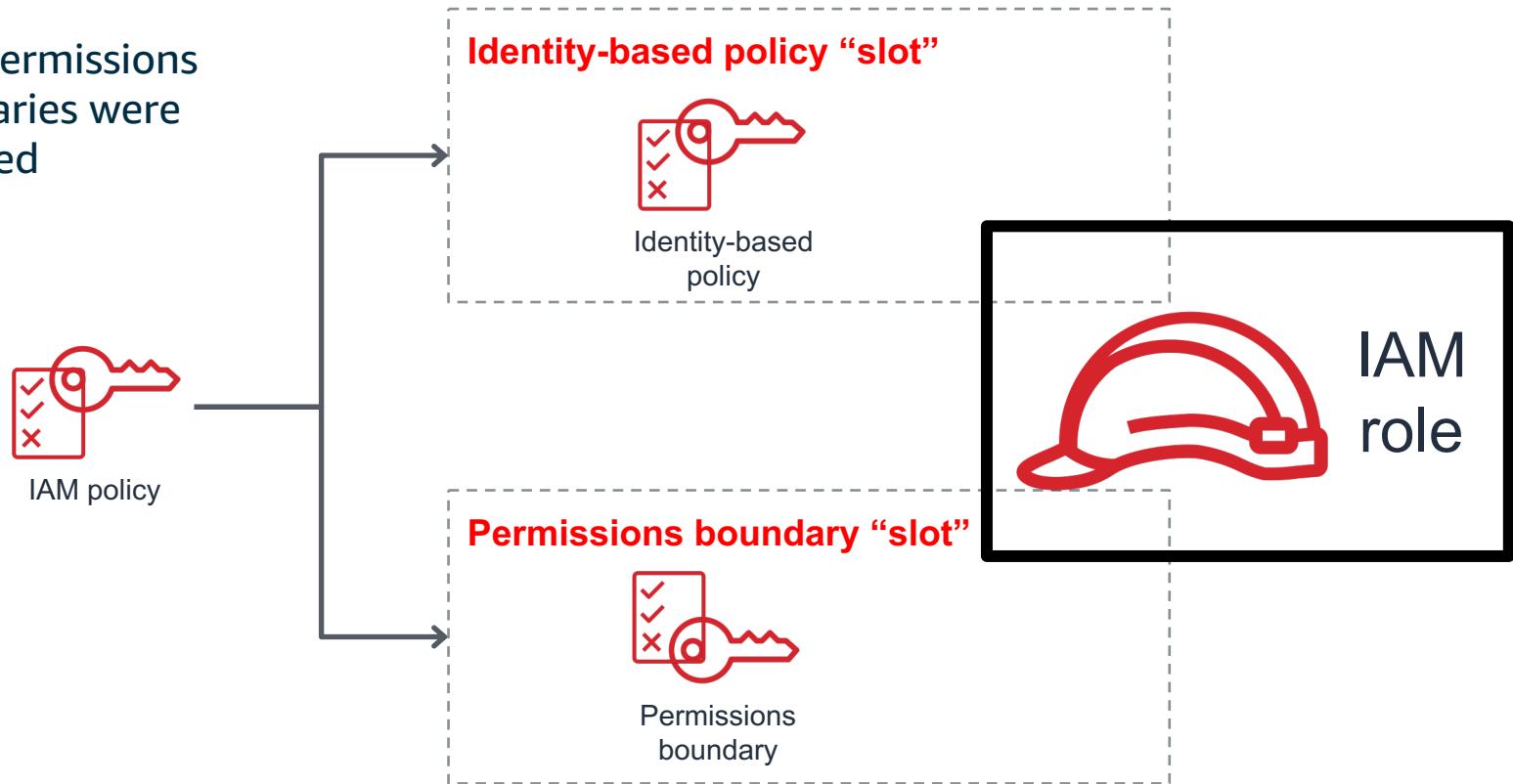


IAM policy



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

After Permissions
Boundaries were
launched



But, it's just an IAM policy right?

▼ Attach permissions policies

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

[Create policy](#)

[Refresh](#)

Filter policies [Search](#) Showing 582 results

	Policy name	Used as	Description
<input type="checkbox"/>	AdministratorAccess	Permissions policy (9)	Provides full access to AWS services an...
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	None	Provide device setup access to AlexaFor...
<input type="checkbox"/>	AlexaForBusinessFullAccess	None	Grants full access to AlexaForBusiness r...
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	None	Provide gateway execution access to AI...
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	None	Provide read only access to AlexaForBu...
<input type="checkbox"/>	AllowAssumeDeleteDDBRole	None	
<input type="checkbox"/>	AllowDeleteofDDBTable	Permissions policy (1)	
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	None	Provides full access to create/edit/delete...

▼ 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

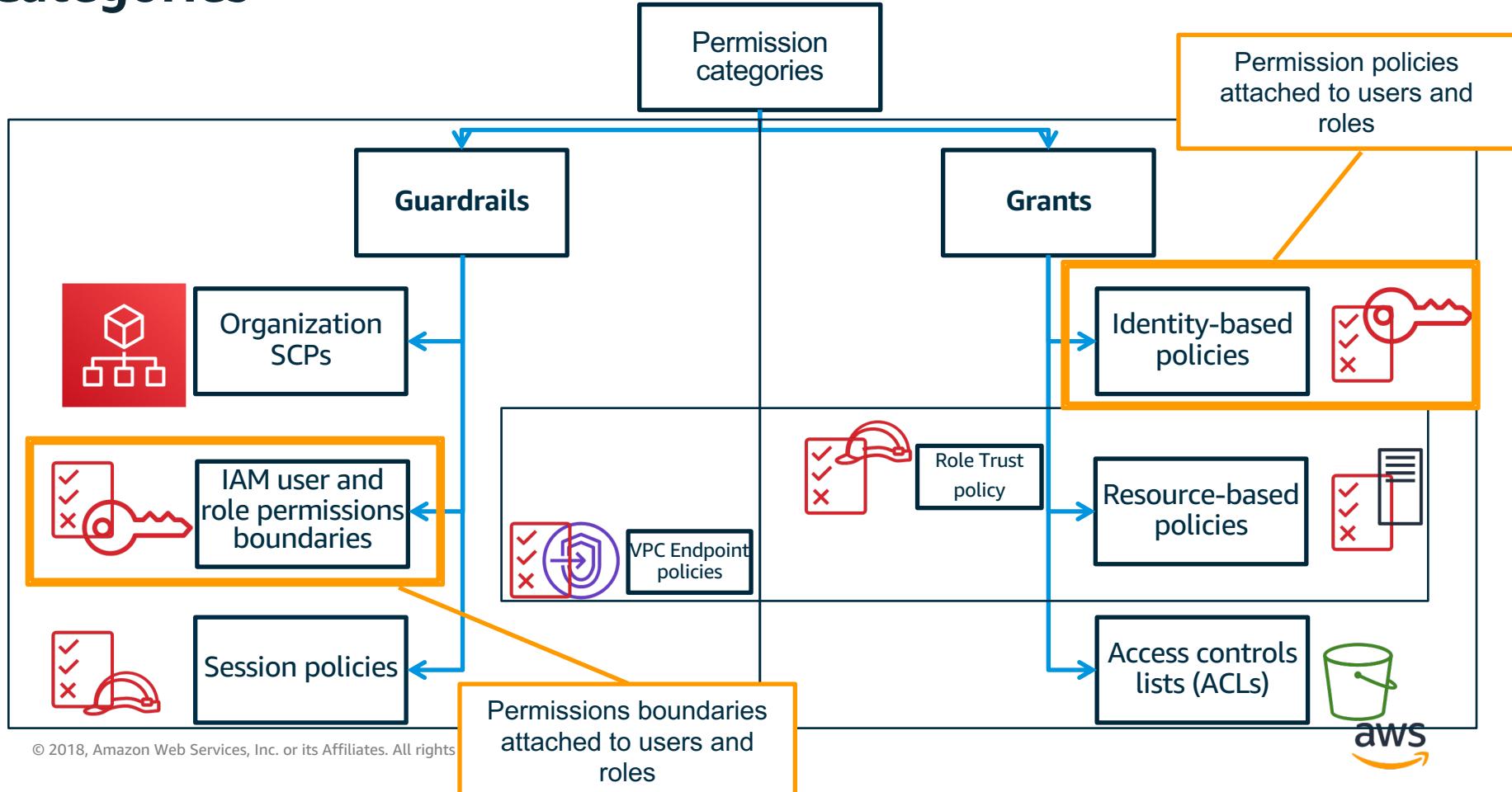
Use a permissions boundary to control the maximum role permissions

Presentation questions

- What is the condition context key used for permission boundaries?
- How does a permission boundary differ from a standard IAM policy?
- What are some permission boundary use cases?

Permissions boundary mechanism

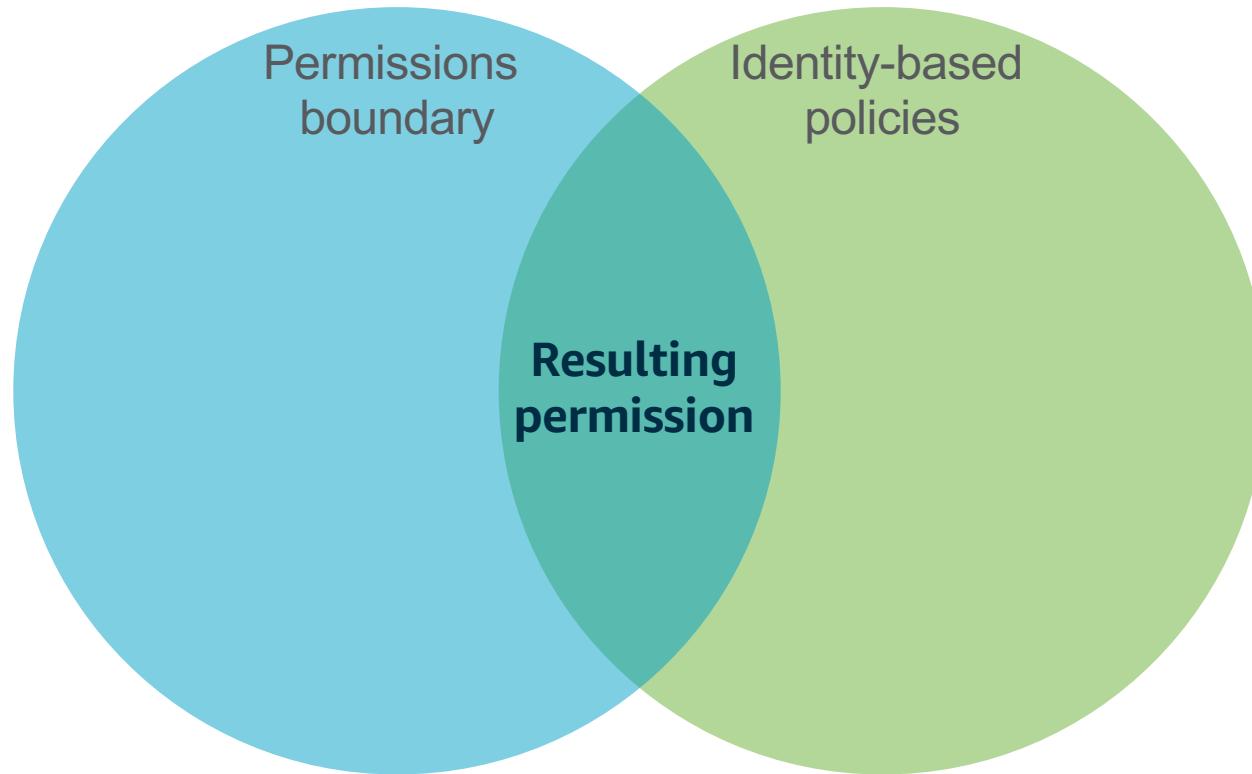
Categories



Everything after authentication

1. **Authenticate** the principal
2. Determine which **policies** apply to the request
3. **Evaluate** the different policy types that apply which affect the order in which they are evaluated.
4. **Allow or Deny** the request

Policy evaluation – Venn diagrams



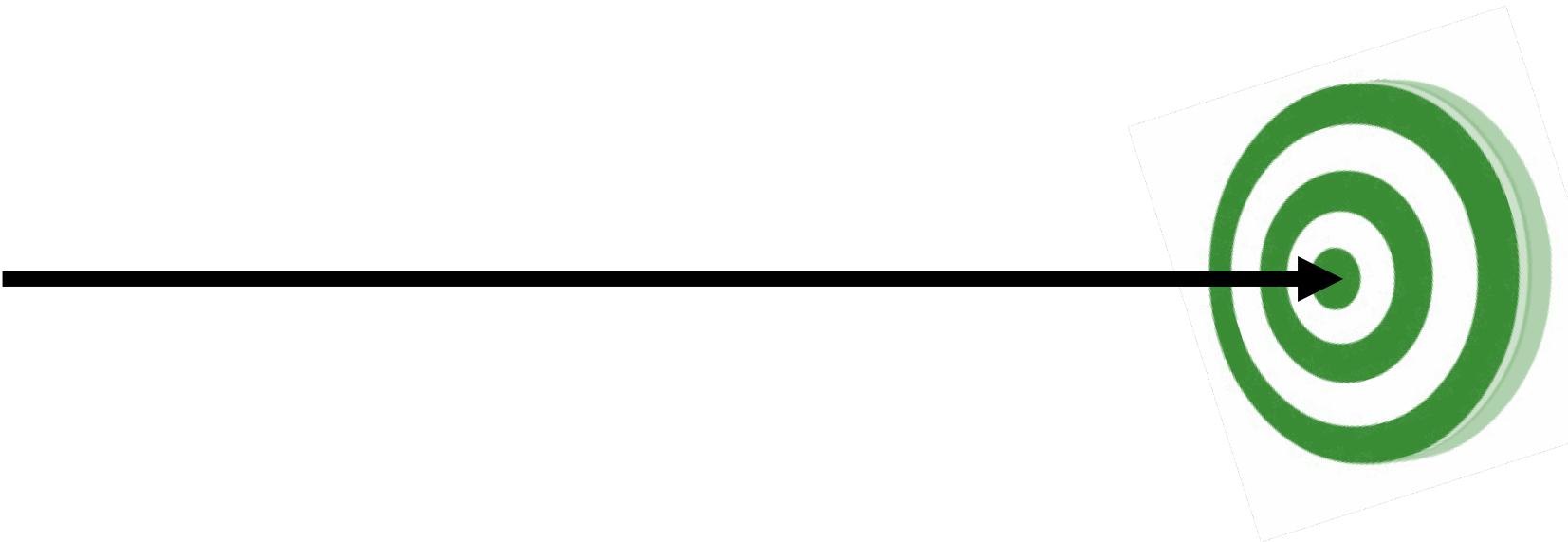
Policy evaluation – the **archery** analogy

API
Request

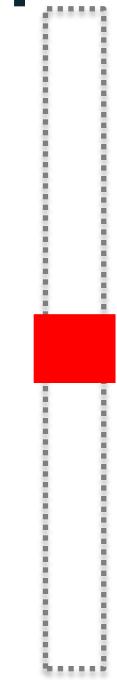


API request trying to hit the target

API
Request



Two types of barriers



Explicit
deny



Allow

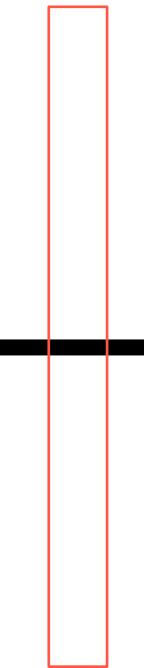
All other
policies



Allow example

Request:
s3:GetObject

API
Request



Explicit
deny



Permissions
boundary

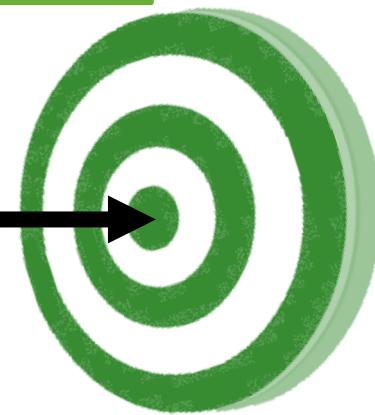


Identity-based
policy

Allow

Allow

Request
allowed



Allow example

Request:
s3:GetObject

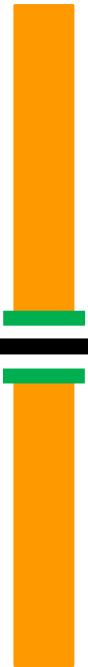
API
Request



Explicit
deny



Permissions
boundary

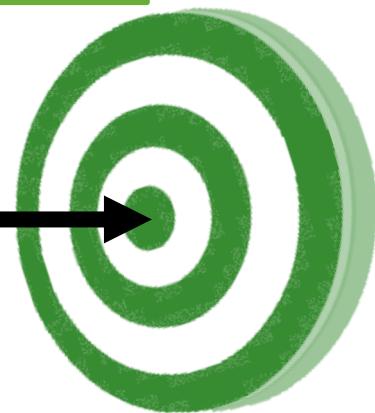


Identity-based
policy

Allow

Allow

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:*:*:*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": ["s3:GetObject"],  
      "Resource": "arn:aws:s3:::example1/*"  
    }  
  ]  
}
```

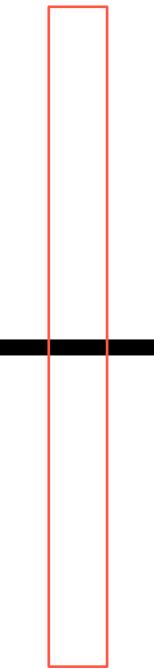
Identity-based policy

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

Allow example

Request:
s3:GetObject

API
Request



Explicit
deny



Permissions
boundary

Request
denied

Allow



Identity-based
policy

Allow



Effective permissions – scenario 2

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:*:*:*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": ["s3:GetObject"],  
      "Resource": "arn:aws:s3:::example1/*"  
    }  
  ]  
}
```

Identity-based policy

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

Allow example

Request:
s3:GetObject

API
Request



Explicit
deny



Permissions
boundary

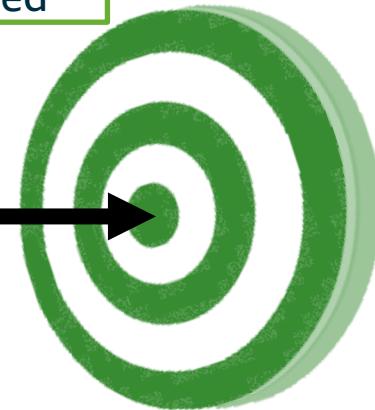


Identity-based
policy

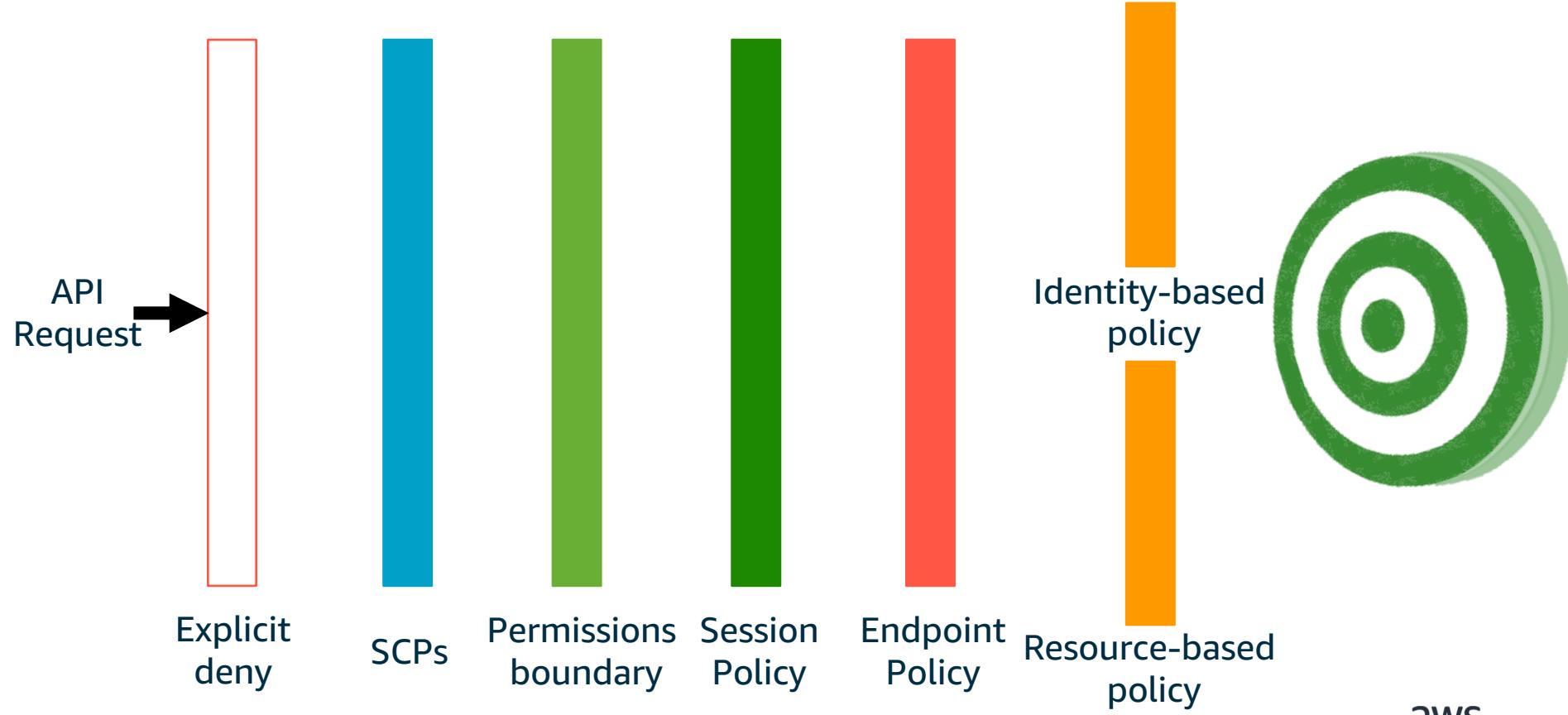
Allow

Allow

Request
allowed



All of the obstacles



Resource Restrictions

Goal: carve out a space for the delegated admins to be able to modify resources without impacting other resources.

Paths are preferred but require the CLI. Naming (e.g. department1*) can also be used.

<https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html#arns-paths>

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns

Resource Restrictions - examples

Resource restriction using **paths**:

"Resource": "arn:aws:iam::123456789012:role/**department1**/*"

Example role:

arn:aws:iam::123456789012:role/department1/role1

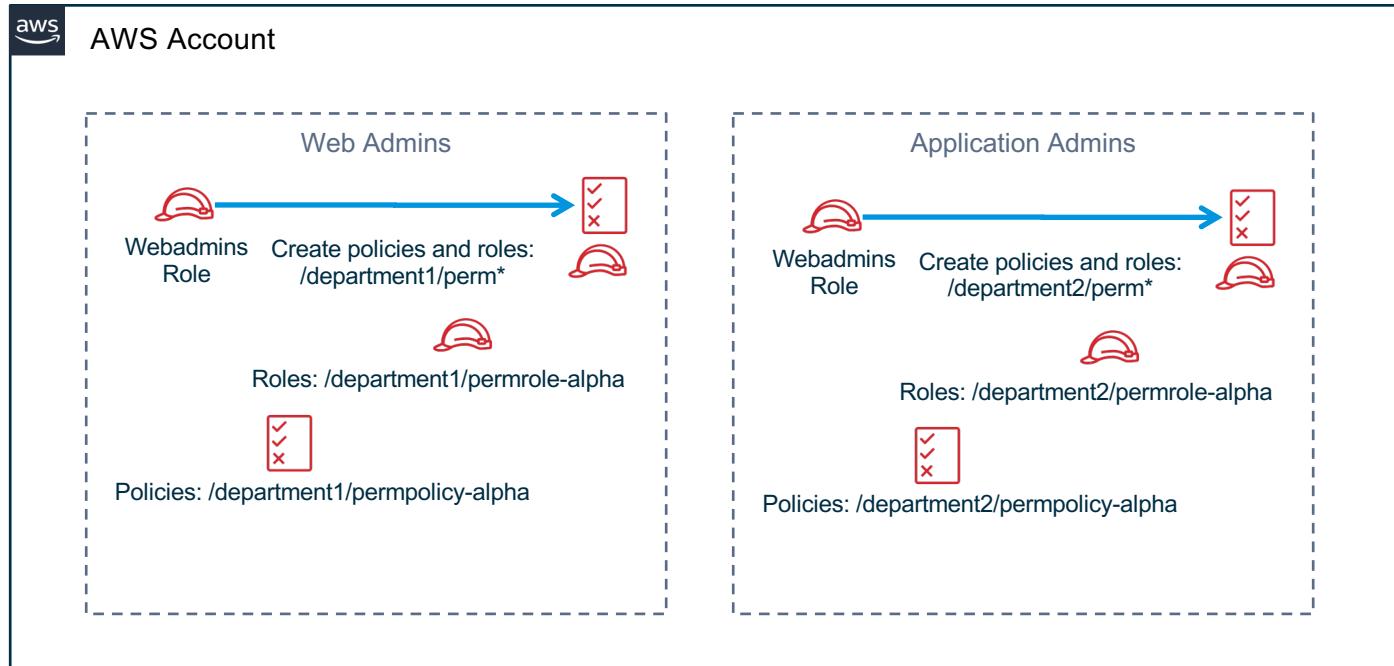
Resource restriction using **names**:

"Resource" : "arn:aws:iam::123456789012:policy/**development-users**/*"

Example policy:

arn:aws:iam::123456789012:policy/development-users-policy1

Pathing Walled Garden

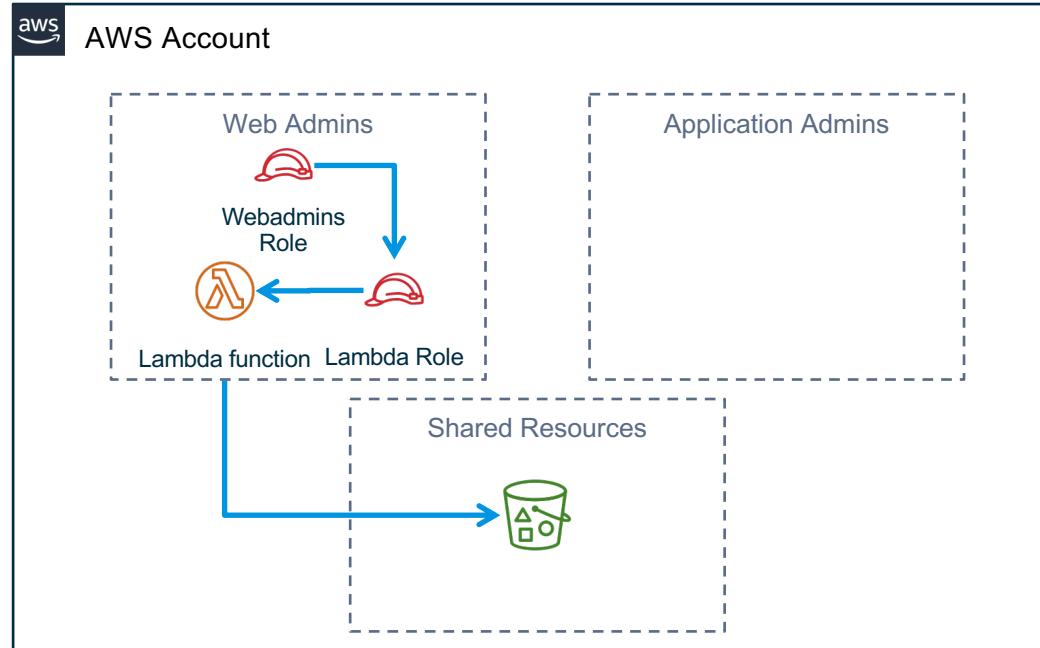




Permissions Boundary Workshop

Overview

Workshop



Workshop

The round is broken down into a BUILD and VERIFY phase.

BUILD (60 min): First each team will carry out the activities involved in the BUILD phase.

VERIFY (30 min): Each team will carry out the VERIFY activities as if they were part of the web admin team.

Permissions boundaries

Build Phase (60 min)

Workshop

Use: US East (Ohio)
us-east-2

Click on **Overview**, read through this, then click on **Build** and run the CloudFormation template:

<https://identity-round-robin.awssecworkshops.com/permission-boundaries/>

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark



Permissions boundaries

Verify Phase (15 min)

Swap credentials with
another team

Workshop

Use: US East (Ohio)
us-east-2

Click on **Verify**:

<https://identity-round-robin.awssecworkshops.com/permission-boundaries/>

Final Q & A

End of workshop questions

- What is the risk of implementing permissions boundaries without resource restrictions?
- Can the same IAM policy be used as both a permissions boundary and a Identity-based policy?
- Is one resource restriction method preferred over the other?

Summary

- The mechanism at the policy level is just a condition context key
- Supported only for user and roles
- Not all IAM actions support the condition context key
- It's just a managed policy
- The user or role can do only the actions allowed by both the attached identity-based policies and the permissions boundary
- Resource restrictions should also be used