aws professional services

Identity & Access Management
Authentication & Authorization

Agenda

- Authentication in AWS IAM
- Federated access to AWS
- Authorization in AWS IAM
- Cross Account Access
- AWS Organizations Service Control Policies
- AWS IAM Permission Boundaries

Goals

- Understand AWS IAM policy language
- Understand when and where to use AWS IAM
- Discover identity federation options
- Understand options for roles and responsibilities
- Discover AWS Organizations Service Control Policy options
- Discover appropriate use of Permission Boundaries

Outcomes

- Decision on the AWS access model (IAM users vs federated access)
- Agreed upon basic set of roles and/or policies
- Decision on the use of AWS Organizations and SCP’s
- Decision on the use of Permission Boundaries

Authentication in AWS IAM

AWS Identity & Access Management

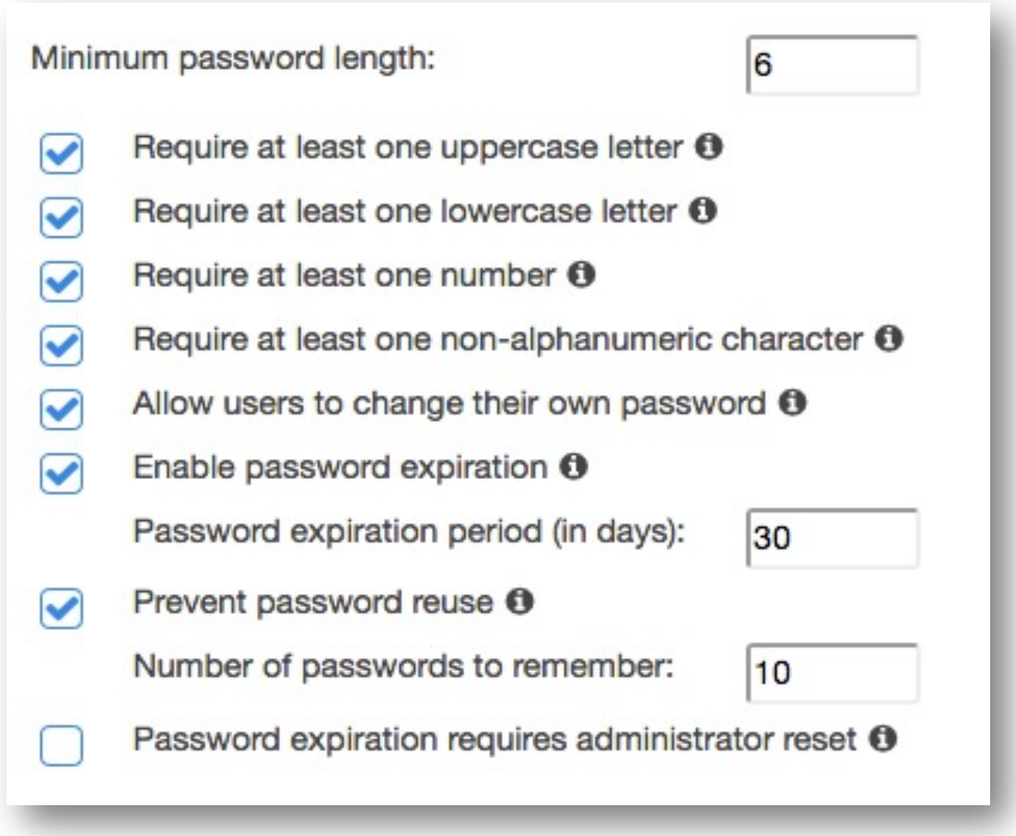
Authentication & Authorization

aws professional services

Authentication

Username/Password

- Console access
- Can set an IAM Password Policy



Authentication

Access Key

- CLI/API access
- Used to sign requests without sending the Secret on the network
- Not retrievable from AWS again – you lose it, generate a new pair

Identifier **ACCESS KEY ID**

`AKIAIOSFODNN7EXAMPLE`

Secret **SECRET KEY**

`UtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

Authentication

Multifactor Authentication (MFA)

- Helps prevent anyone with unauthorized knowledge of your credentials from impersonating you.
- Virtual, Hardware, U2F
- Works with
  - Root credentials
  - IAM Users
  - Application
- Integrated into
  - AWS API
  - AWS Management Console
  - Key pages on the AWS Portal
  - S3 (Secure Delete)

Keys or Password?

Depends on how your users will access AWS

- Console → Password
- API, CLI, SDK → Access keys

In either case, make sure to rotate credentials regularly

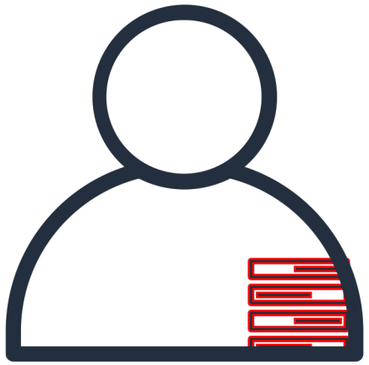
- Use Credential Report to audit credential rotation
- Configure password policy
- Configure policy to allow access key rotation

Federated Access to AWS

AWS Identity & Access Management
Authentication & Authorization

aws professional services

Federation - access patterns



AWS IAM federation

External IdP

AWS SSO federation

AWS SSO

AWS SSO federation

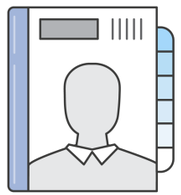
External IdP

Federation – AWS IAM federation

IdP configuration

External IdP

Configure the relying party



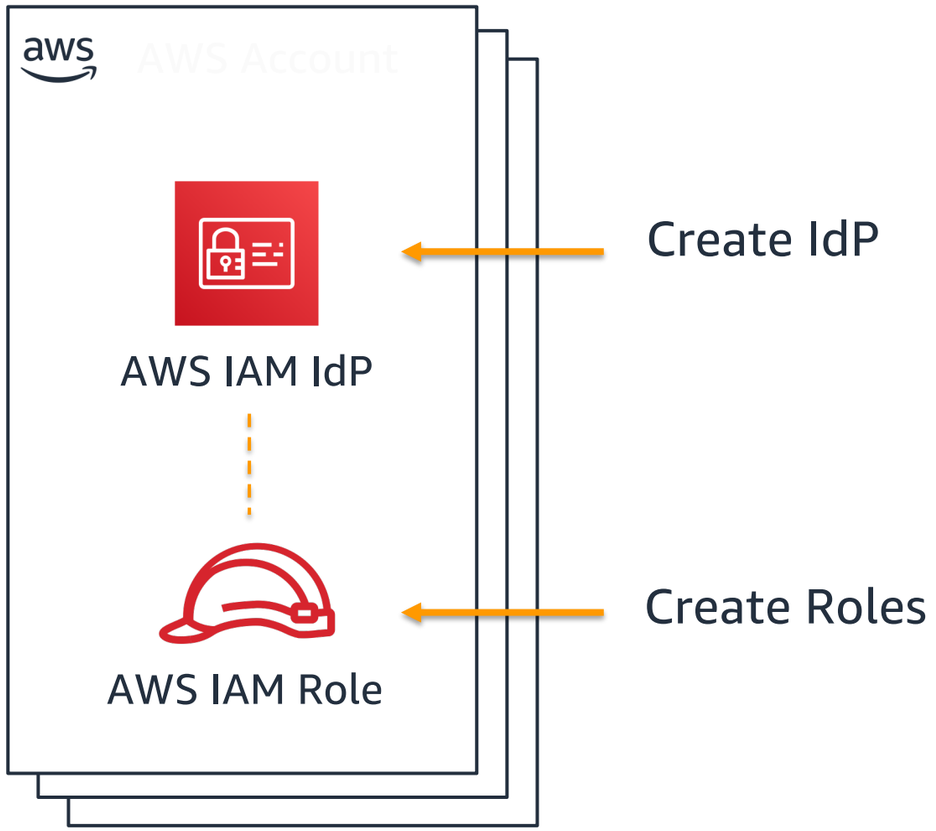
User Directory

Create matching groups

Cloud builder

Add users to groups

AWS configuration

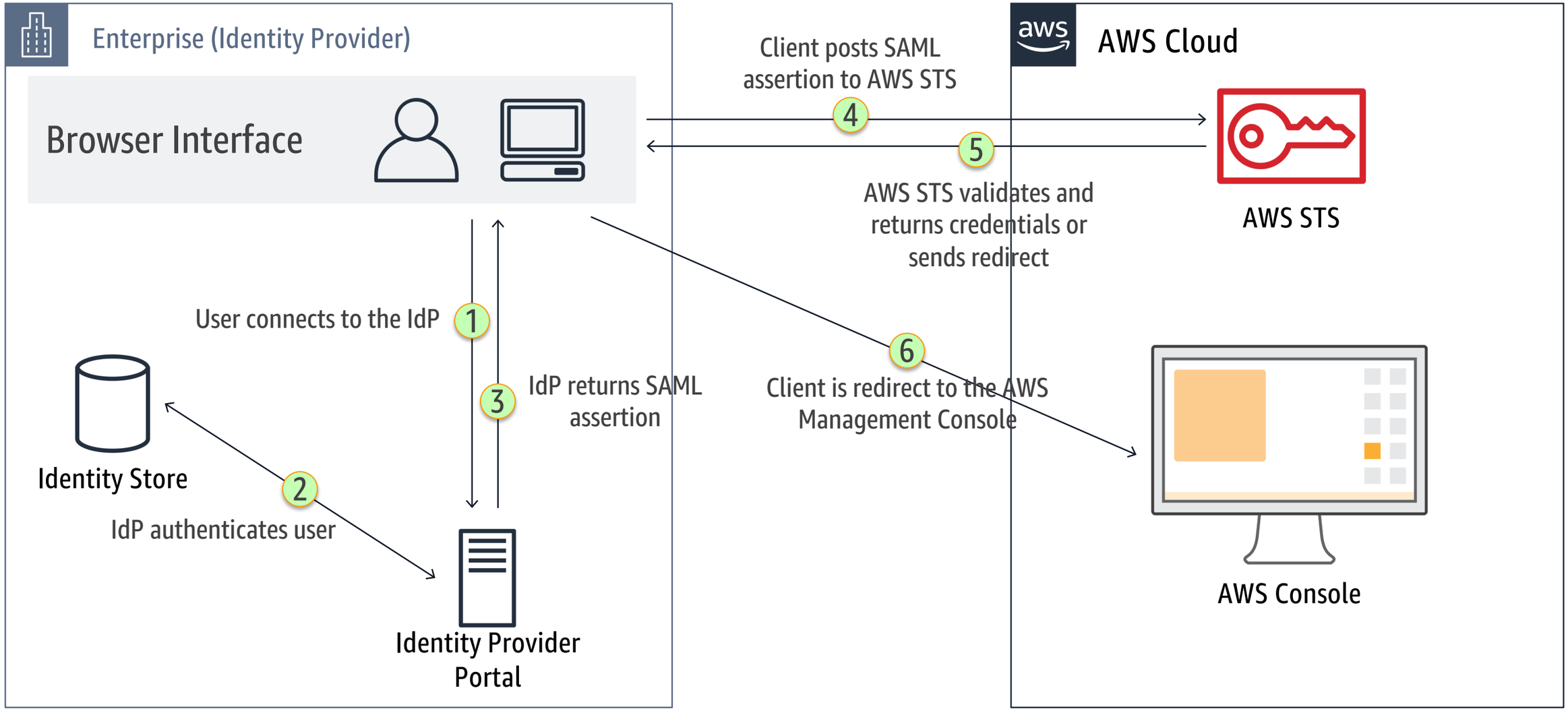


Create IdP

Create Roles

aws professional services

Federation – SAML2.0



aws professional services

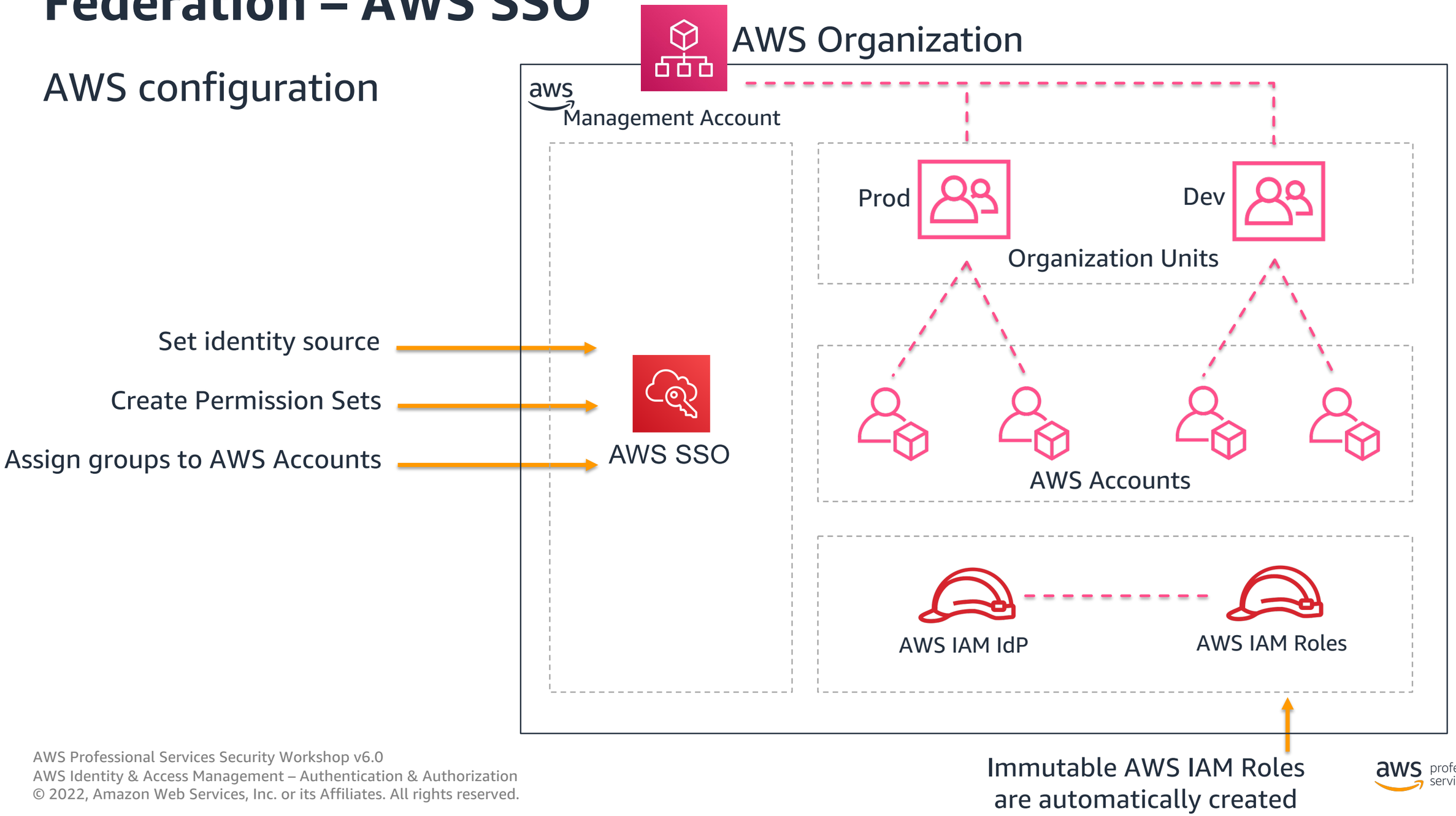
Federation – AWS SSO

Centrally manage Single Sign-On access

- Flexible directory options
- Manage AWS account access at scale
- SSO for AWS integrated applications
- SSO for business applications

Federation – AWS SSO

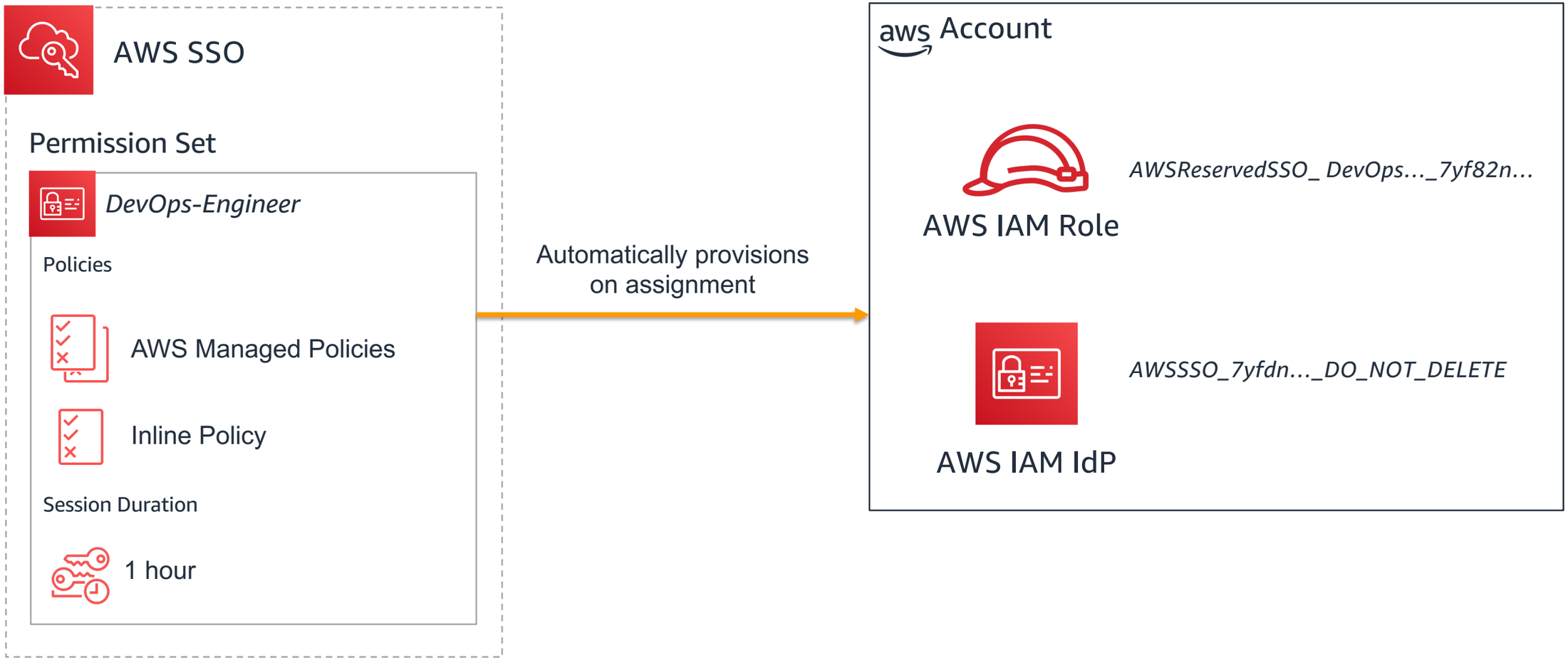
AWS configuration



aws professional services

Federation – AWS SSO

Permission Sets are Role definitions



aws professional services

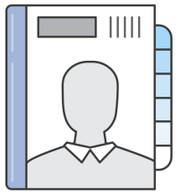
Federation – AWS SSO with external IdP

IdP configuration

External IdP

Configure the relying party

Configure SCIM provisioning



User Directory

Create matching groups

Cloud builder

Add users to groups

AWS configuration

Set identity source as an *external IdP*

Enable automatic provisioning

Create Permission Sets

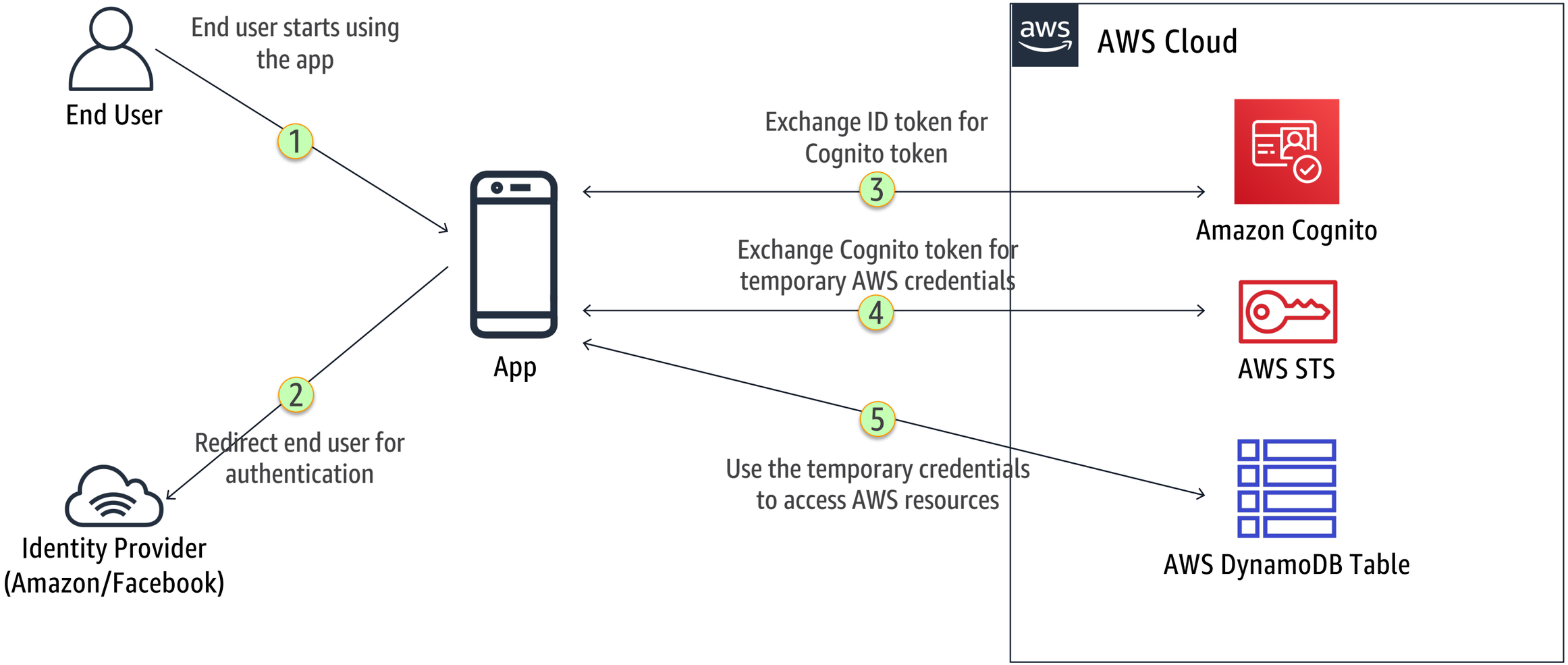
Assign groups to AWS Accounts

aws professional services

Federation - access patterns

Access pattern	Identity provider	User directory source	Credentials	AWS Organizations support	AWS IAM management	Immutable AWS IAM Roles	Group to Role mapping	CLI Support	Pattern maturity
AWS IAM federation	External	External	External	No	Decentralized	With SCPs	1-1	Open source / homegrown	Leader
AWS SSO federation	AWS SSO	AWS SSO / AD	AWS SSO	Yes	Centralized	Inherent	Flexible	AWS CLI integration	Trending
AWS SSO federation	External & AWS SSO	External	External	Yes	Centralized	Inherit	Flexible	AWS CLI integration	Trending

Federation – Cognito (for applications)



aws professional services

Authorization in AWS IAM

AWS Identity & Access Management

Authentication & Authorization

aws professional services

Authorization

Permissions are to specify

Who can access to AWS resources

What action can be performed on those AWS resources

How is it done?

- Organized in **Policies** *(JSON)*

Authorization

Identity-Based Permissions

User: Brian

Can Read, Write, List

On Resource X

Group: Admins

Can Read, Write, List

On Resource XYZ

Group: Developers

Can Read, List

On Resource YZ

Resource-Based Permissions

Resource X

Brian: Read, Write, List
Admins: Read, Write, List
Developers: List

Resource Y

Brian: Read, Write, List
Bob: List
Iris: Read

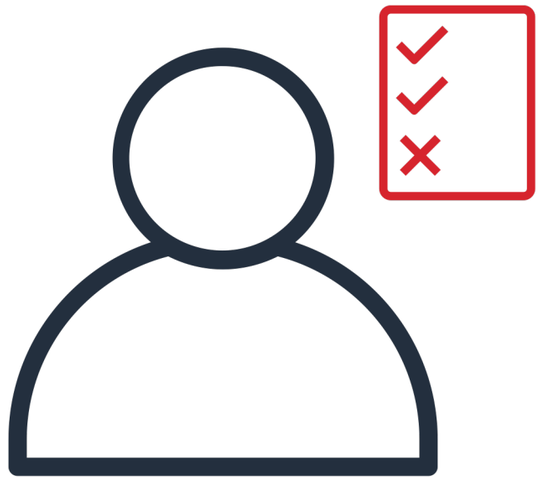
Resource Z

Admins: Read, Write, List
Developers: Read

aws professional services

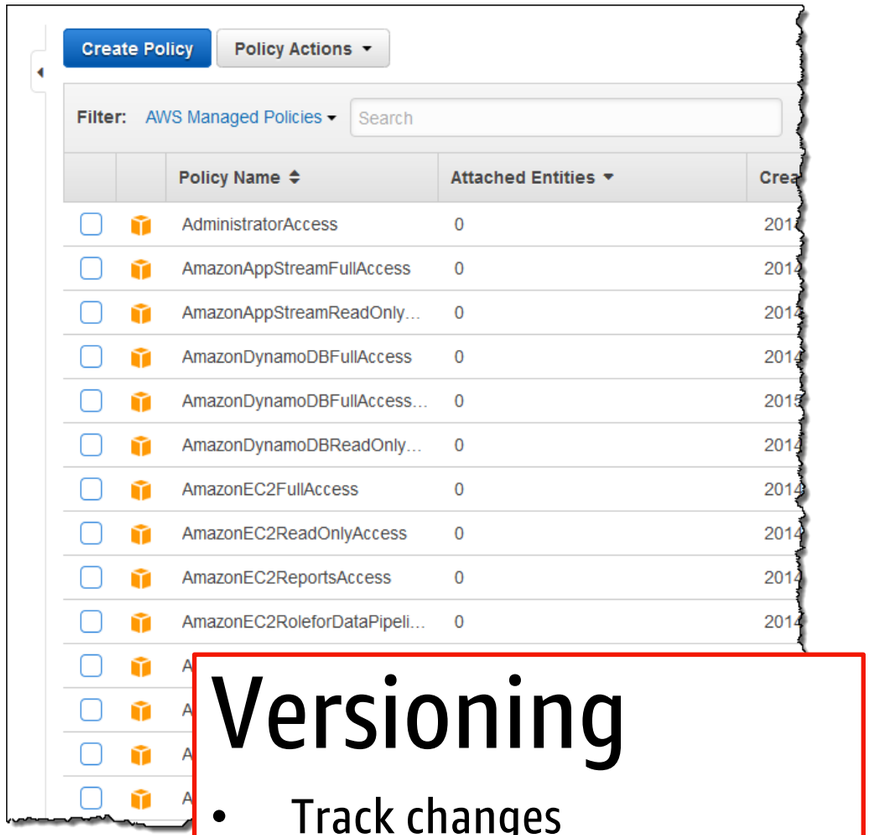
Authorization – Identity-Based Permissions

- Are built in Policies
- Attached to an IAM user, group, or role
- Enable you specify what that user, group, or role can do
- User-based policies: **managed** or **inline**



Authorization – Identity-Based Permissions

- Managed Policies
  - AWS managed policies
  - Customer managed policies
  - Reusable
  - Versioning
- Inline Policies
  - Embedded into a user, group or role
  - Disposable / Temporary



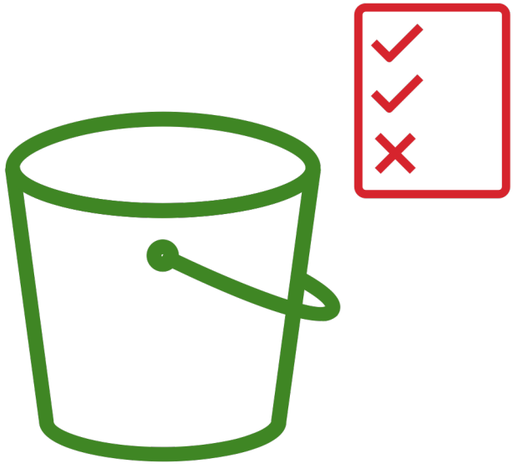
Versioning

- Track changes
- Enables rollback
- Keep up to five versions

aws professional services

Authorization – Resource-Based Permissions

- Are built in Policies
- Attached to a resource
- Resources include:
    - Amazon S3 buckets
    - Amazon Glacier vaults
    - Amazon SNS topics
    - Amazon SQS queues
    - VPC Endpoints
    - AWS Key Management Service encryption keys
- Specify who has access to the resource and what actions they can perform on it
- Resource-based policies : inline only



Authorization – Resource-Based Permissions

- JSON-formatted documents
- Contain a statement (permissions) that specifies:
  - Which actions a principal can perform
  - Which resources can be accessed

```
{
 "Statement":[{
   "Effect":"effect",
   "Principal":"principal",
   "Action":"action",
   "Resource":"arn",
   "Condition":{
     "condition":{
       "key":"value" }
     }
   }
  ]
}
```

Principal

Action

Resource

Condition

You can have multiple statements and each statement is comprised of PARC.

aws professional services

Authorization – Policies

Identity-Based versus Resource-Based

```
{
 "Statement":[{
   "Effect":"effect",
   "Action":"action",
   "Resource":"arn",
   "Condition":{
     "condition":{
       "key":"value" }
     }
    }
   ]
 }
```

Identity-based Policy

```
{
 "Statement":[{
   "Effect":"effect",
   "Principal":"principal",
   "Action":"action",
   "Resource":"arn",
   "Condition":{
     "condition":{
       "key":"value" }
     }
    }
   ]
}
```

Resource-based Policy

Principal

Action

Resource

Condition

aws professional services

Authorization – Policies

```
{
   "Version": "2012-10-17",
   "Statement": {
     "Effect": "Allow",
     "Action": "s3:ListBucket",
     "Resource": "arn:aws:s3:::example_bucket”
  }
}
```

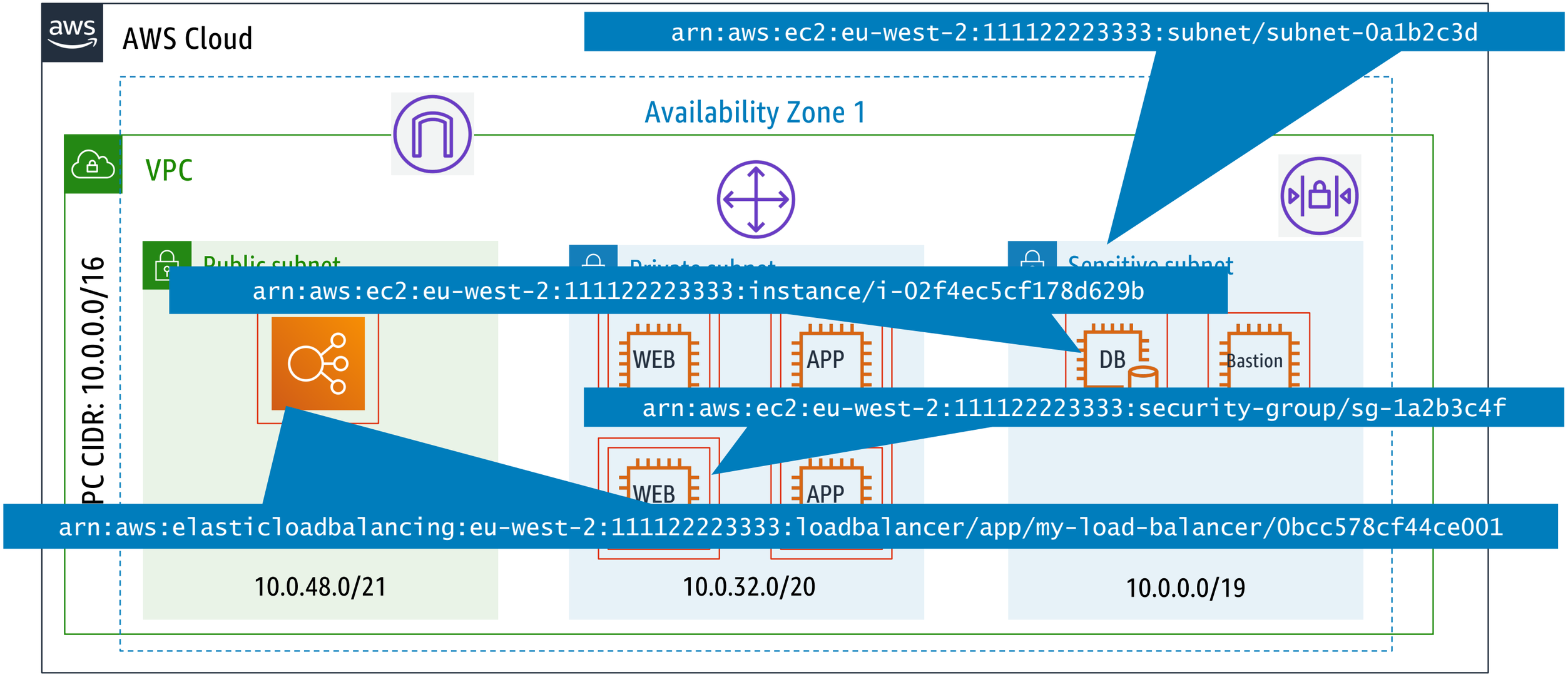
You can attach this policy to an IAM user or group. If that's the only policy for the user or group, the user or group is allowed to perform only this one action (ListBucket) on one Amazon S3 bucket (example_bucket).

Authorization – Amazon Resource Name (ARN)

```
arn:partition:service:region:account-id:resource
arn:partition:service:region:account-id:resourcetype/resource
arn:partition:service:region:account-id:resourcetype:resource
```

- **Partition**: For standard AWS regions, the partition is `aws`.
- **Service**: The service namespace (example: `iam`)
- **Region**: The region the resource resides in (example: `us-west-2`)
- Note that the ARNs for some resources do not require a region, so this component might be omitted.
- **Account-id**: Example: `123456789012`

Authorization – Amazon Resource Name (ARN)



Authorization – Principals

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"

<!-- Federated user (using web identity federation) -->
"Principal":{"Federated":"www.amazon.com"}
"Principal":{"Federated":"graph.facebook.com"}
"Principal":{"Federated":"accounts.google.com"}

<!-- Specific role -->
"Principal":{"AWS":"arn:aws:iam::123456789012:role/rolename"}

<!-- Specific service -->
 "Principal":{"Service":"ec2.amazonaws.com"}
```

Replace with your AWS account number

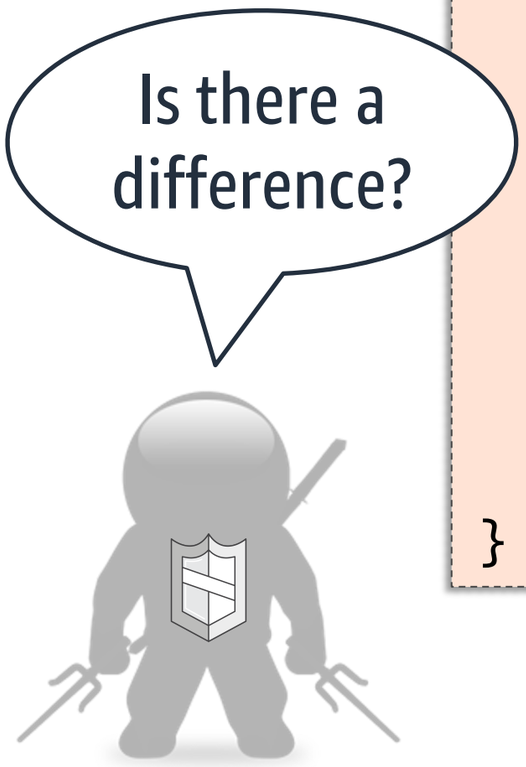
aws professional services

Authorization – Actions

```
<!-- EC2 action -->
"Action":"ec2:StartInstances"

<!-- IAM action -->
"Action":"iam:ChangePassword"

<!-- S3 action -->
"Action":"s3:GetObject"

<!-- Specify multiple values for the Action element -->
"Action":["sqs:SendMessage","sqs:ReceiveMessage"]

<-- Use wildcards (* or ?) as part of the action name. This would cover
Create/Delete/List/Update -->
"Action":"iam:*AccessKey*"
```

Authorization – NotAction

```
{
  "Version": "2012-10-17",
  "Statement": [ {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

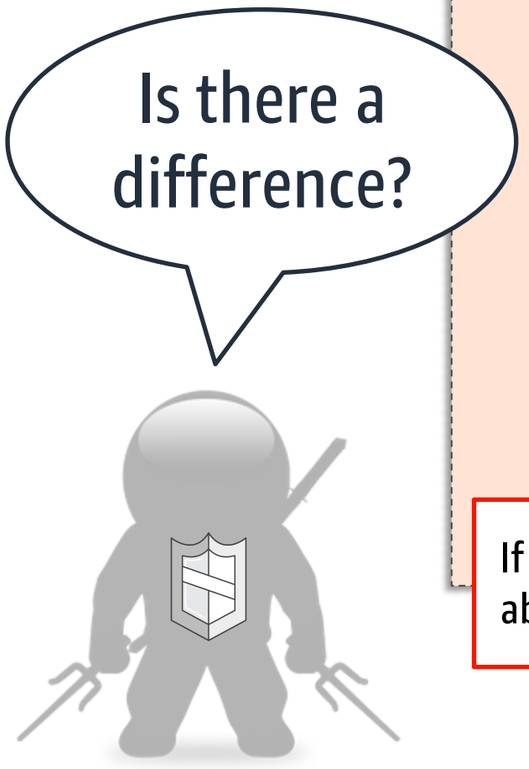



```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
```

Authorization – NotAction

```
{
  "Version": "2012-10-17",
  "Statement": [ {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

This is not a **Deny**. A user could still have a separate policy that grants **IAM:***




```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
```

If you want to prevent the user from ever being able to call IAM APIs, use an **explicit deny**.

Authorization – Conditions

Restricting access to a time frame and IP address

```
"Condition" :  {
  "DateGreaterThan" : {"aws:CurrentTime" : "2015-10-08T12:00:00Z"},
  "DateLessThan": {"aws:CurrentTime" : "2015-10-08T15:00:00Z"},
  "IpAddress" : {"aws:SourceIp" : ["192.0.2.0/24", "203.0.113.0/24"]}
}
```

AND

OR

Allows a user to access a resource under the following conditions:

- The time is after 12:00 P.M. on 10/8/2015 **AND**
- The time is before 3:00 P.M. on 10/8/2015 **AND**
- The request comes from an IP address in the 192.0.2.0 /24 **OR** 203.0.113.0 /24 range

All of these conditions must be met in order for the statement to evaluate to TRUE.

Authorization – Conditions

Examples:

- aws:CurrentTime
- aws:EpochTime
- aws:MultiFactorAuthAge
- aws:MultiFactorAuthPresent
- aws:SecureTransport
- aws:UserAgent
- aws:PrincipalOrgID
- aws:PrincipalType
- aws:Referer
- aws:RequestedRegion
- aws:RequestTag/*tag-key*
- aws:ResourceTag/*tag-key*
- aws:SourceAccount
- aws:SourceArn
- aws:SourceIp
- aws:SourceVpc
- aws:SourceVpce
- aws:TagKeys
- aws:TokenIssueTime
- aws:userid
- aws:username

Authorization – Policy Variables

- Predefined variables based on service request context
  - **Global** keys (`aws:SourceIP`, `aws:MultiFactorAuthPresent`, etc.)
  - **Principal-specific** keys (`aws:username`, `aws:userid`, `aws:PrincipalType`)
  - **Provider-specific** keys (`graph.facebook.com:id`, `www.amazon.com:user_id`)
  - **SAML** keys (`saml:cn`, `saml:edupersonassurance`)
  - See documentation for service-specific variables
- Benefits
  - Simplify policy management
  - Reduce the need for hard-coded, user-specific policies

Authorization – Policy Variables

Applicable to Brian:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["iam:*AccessKey*"],
      "Effect": "Allow",
      "Resource": ["arn:aws:iam::123456789012:user/Brian"]
    }
  ]
}
```

Applicable to all users:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["iam:*AccessKey*"],
      "Effect": "Allow",
      "Resource": ["arn:aws:iam::123456789012:user/${aws:username}"]
    }
  ]
}
```

Authorization – Policy Variables

Grants a user access to a home directory in S3 that can be accessed programmatically

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Action": ["s3:ListBucket"],
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":
                {"StringLike":
                   {"s3:prefix":["home/${aws:username}/*"]}
        }
    },
    {
      "Effect":"Allow",
      "Action":["s3:*"],
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                   "arn:aws:s3:::myBucket/home/${aws:username}/*"]
    }
  ]
}
```

Version is required

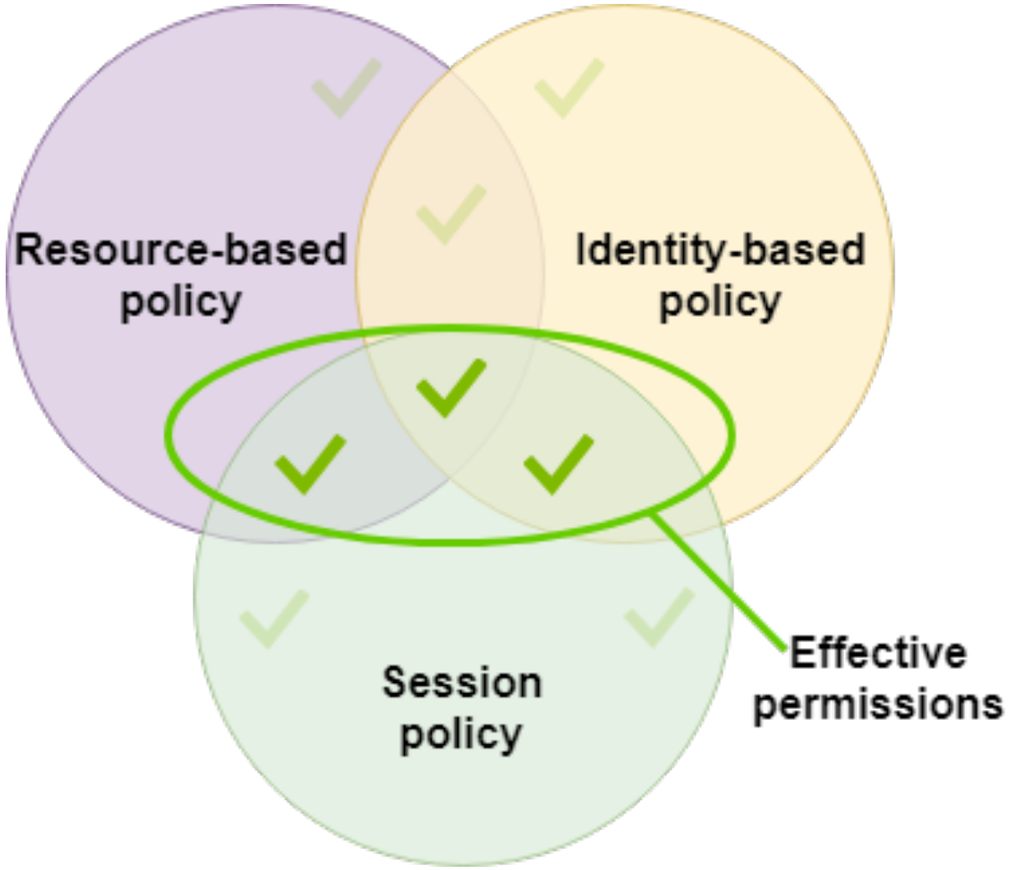
Variable in conditions

Variable in resource ARNs

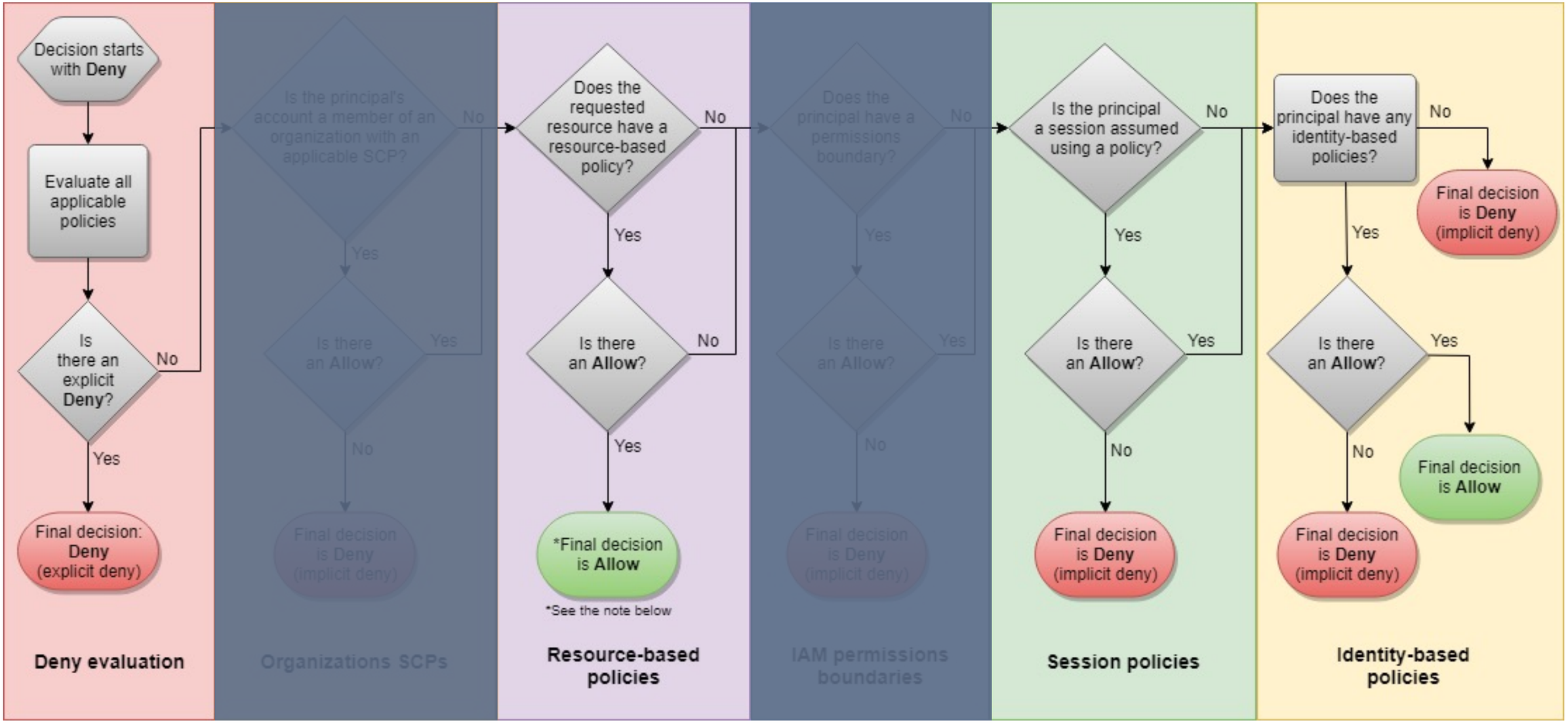
Authorization – Session Policies

Session Policies

- Can be passed as a parameter for programmatically created sessions.
- Effective permissions come from:
    - Identity-based permissions
    - Resource-based permissions
    - Session-based based permissions



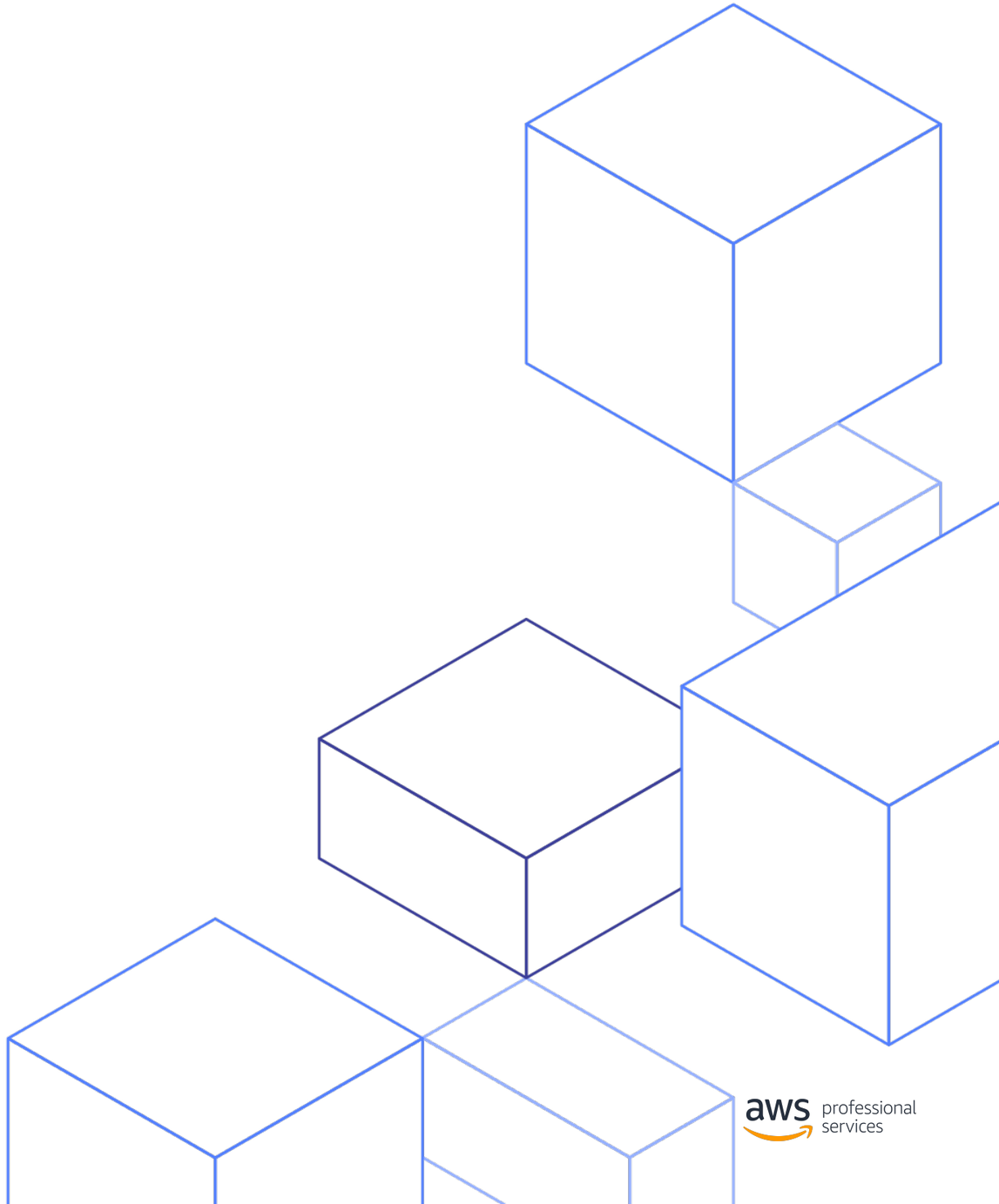
IAM Evaluation Logic



aws professional services

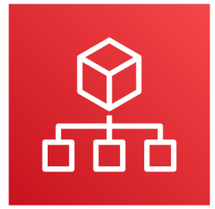
AWS Organizations

AWS Identity & Access Management
Authentication & Authorization



aws professional services

AWS Organizations

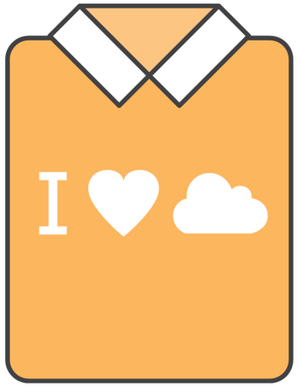


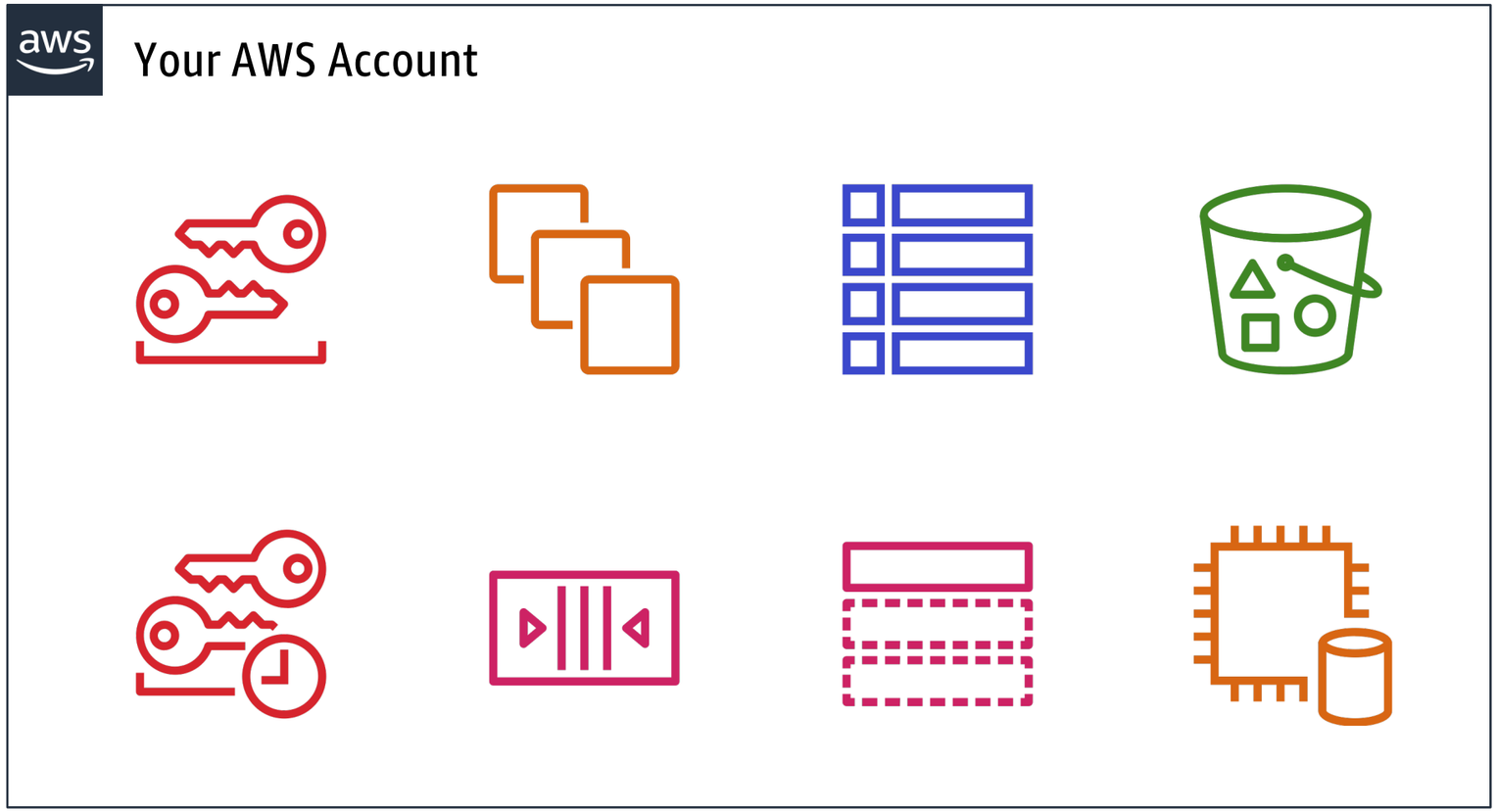
- Manage/control multiple AWS accounts centrally
- Enable multi-account functionality for AWS services

Key features:

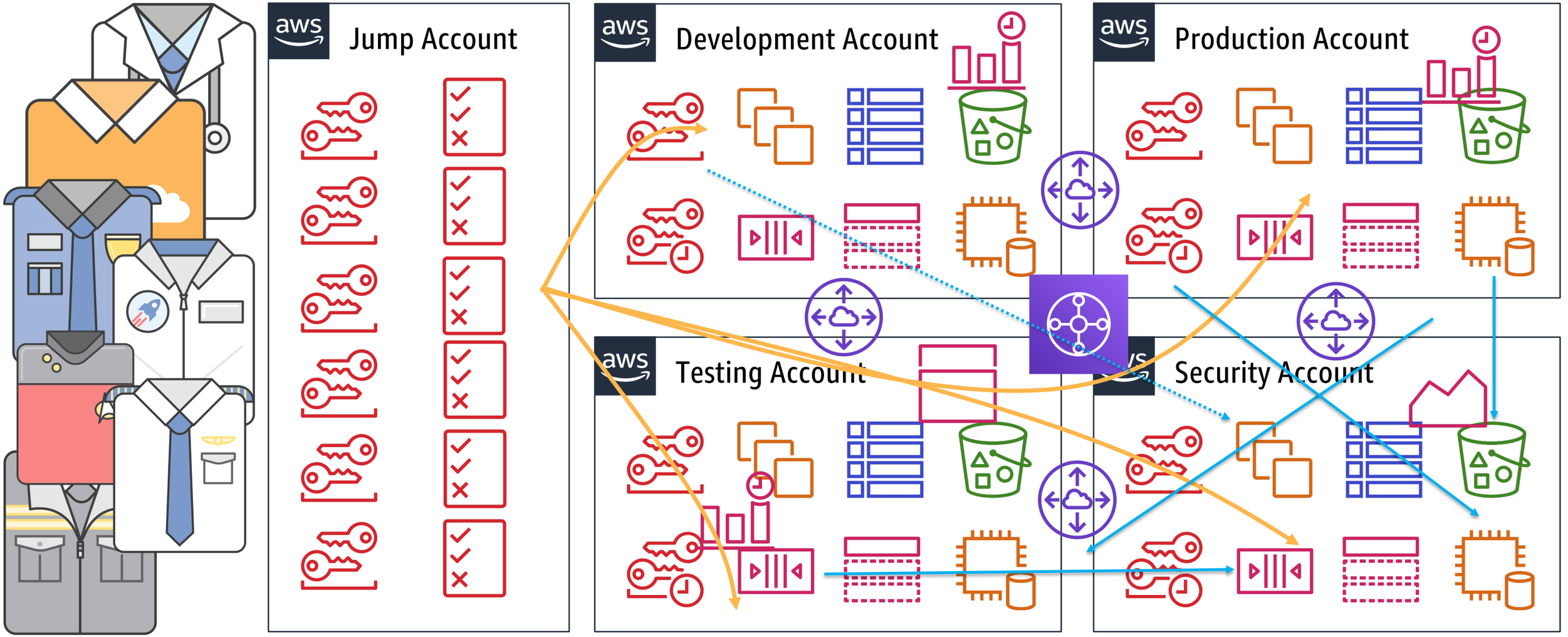
- Simplified creation of new AWS accounts
- Logically group AWS accounts for management convenience
- Apply organizational policies to control AWS services
- Consolidate billing and usage across all accounts into one bill

AWS Organizations – In the beginning…



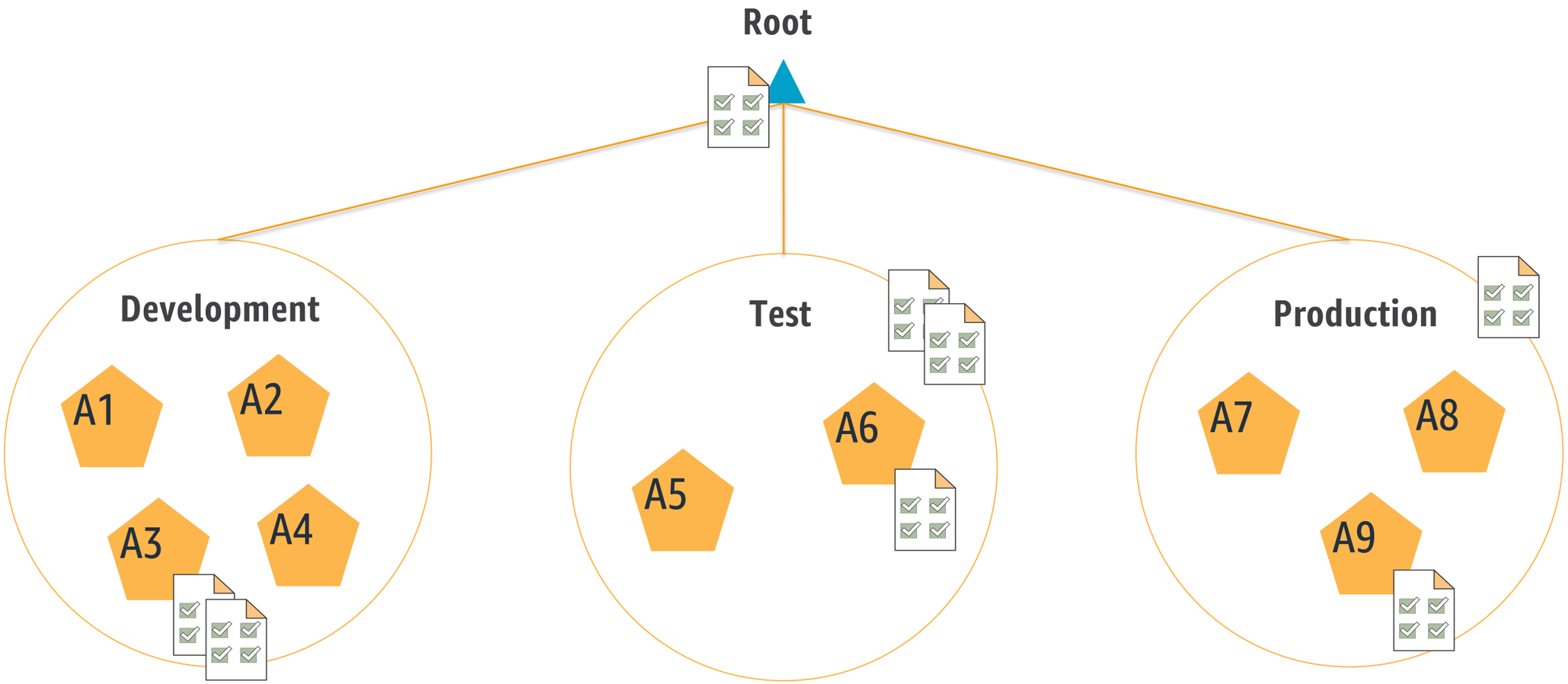


AWS Organizations – Today



aws professional services

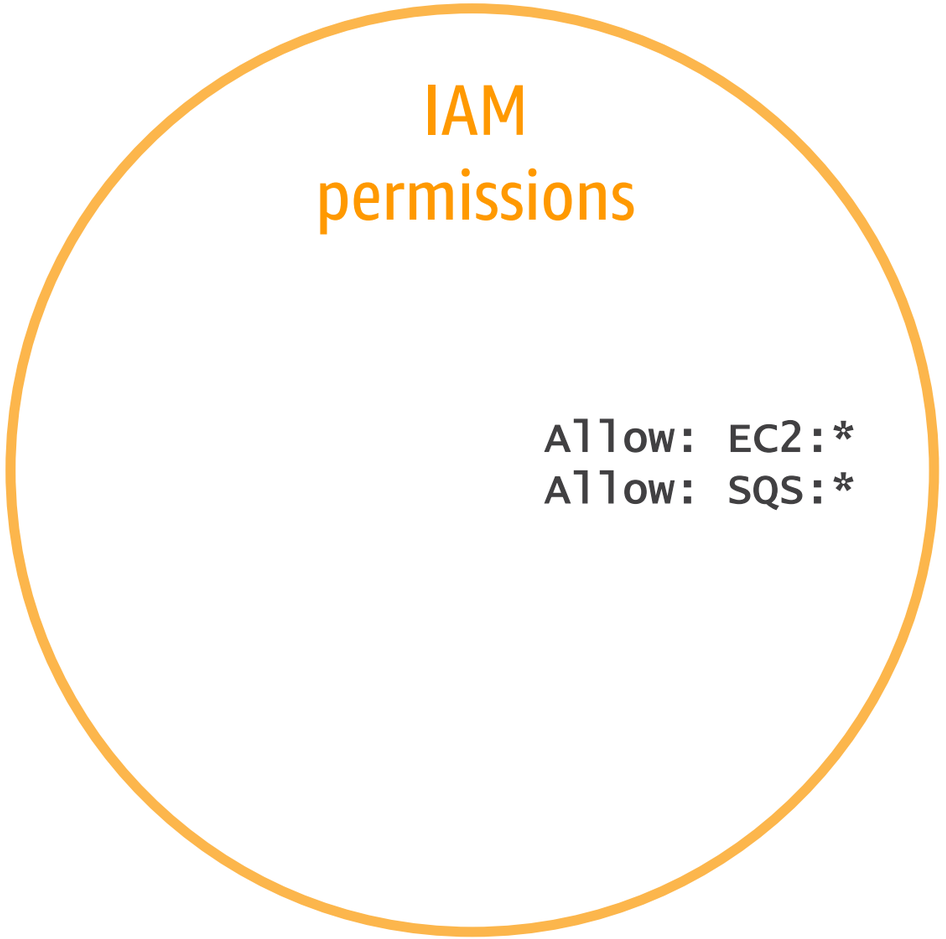
AWS Organizations – Hierarchy and Policies



Service Control Policies use the IAM policy language

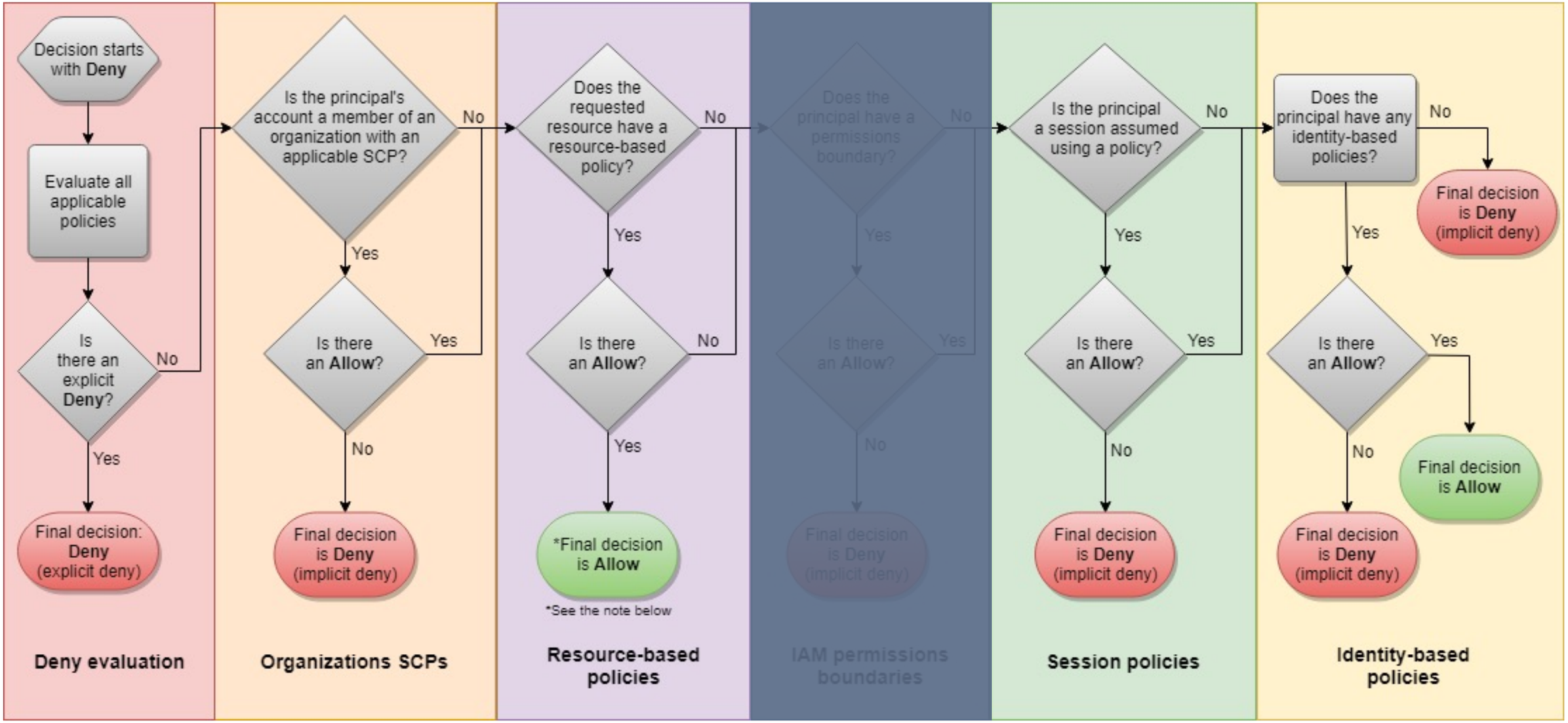
aws professional services

AWS Organizations – Hierarchy and Policies



AWS Organizations – Hierarchy and Policies

IAM Evaluation Logic

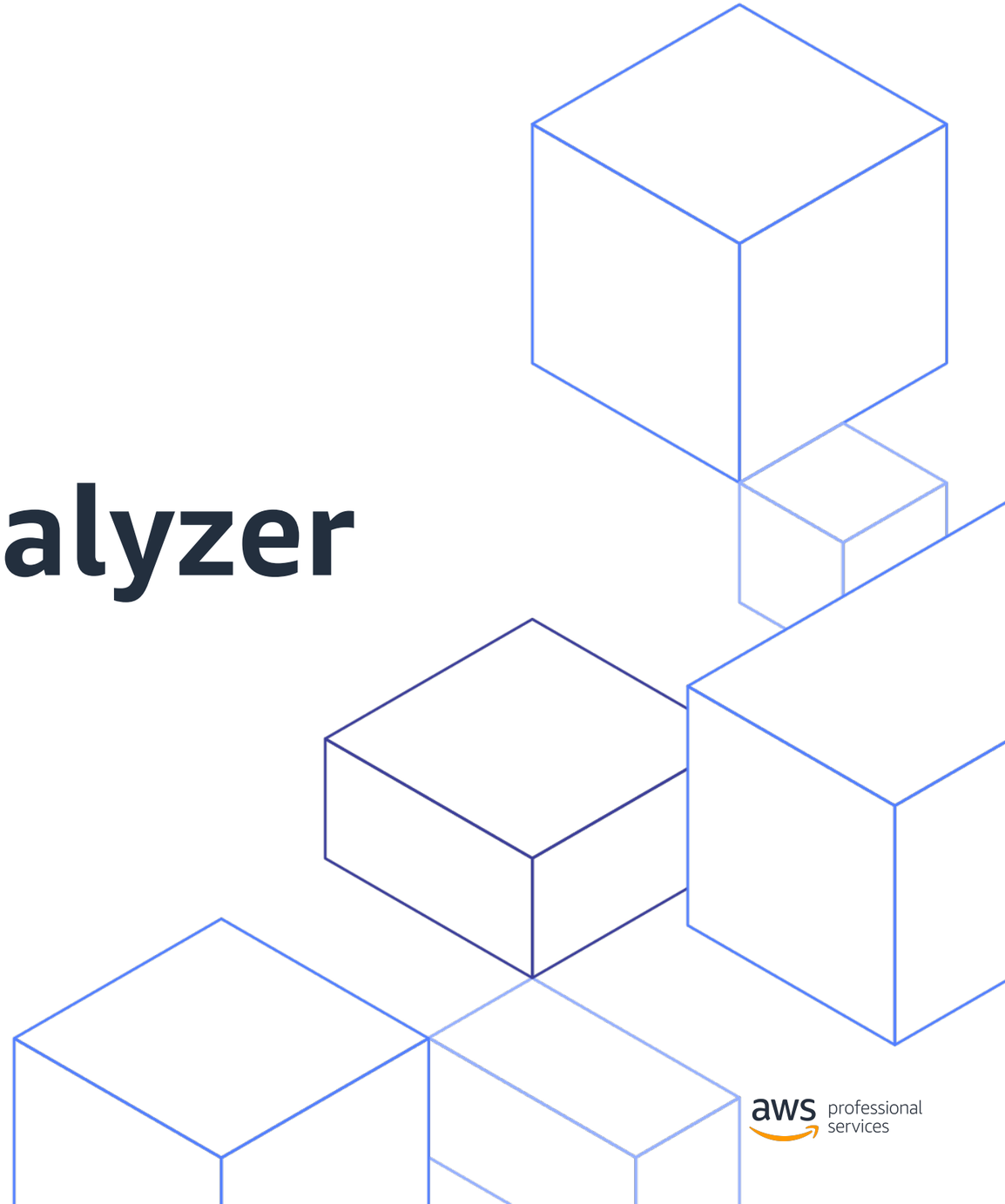


aws professional services

AWS IAM Access Analyzer

AWS Identity & Access Management

Authentication & Authorization



aws professional services

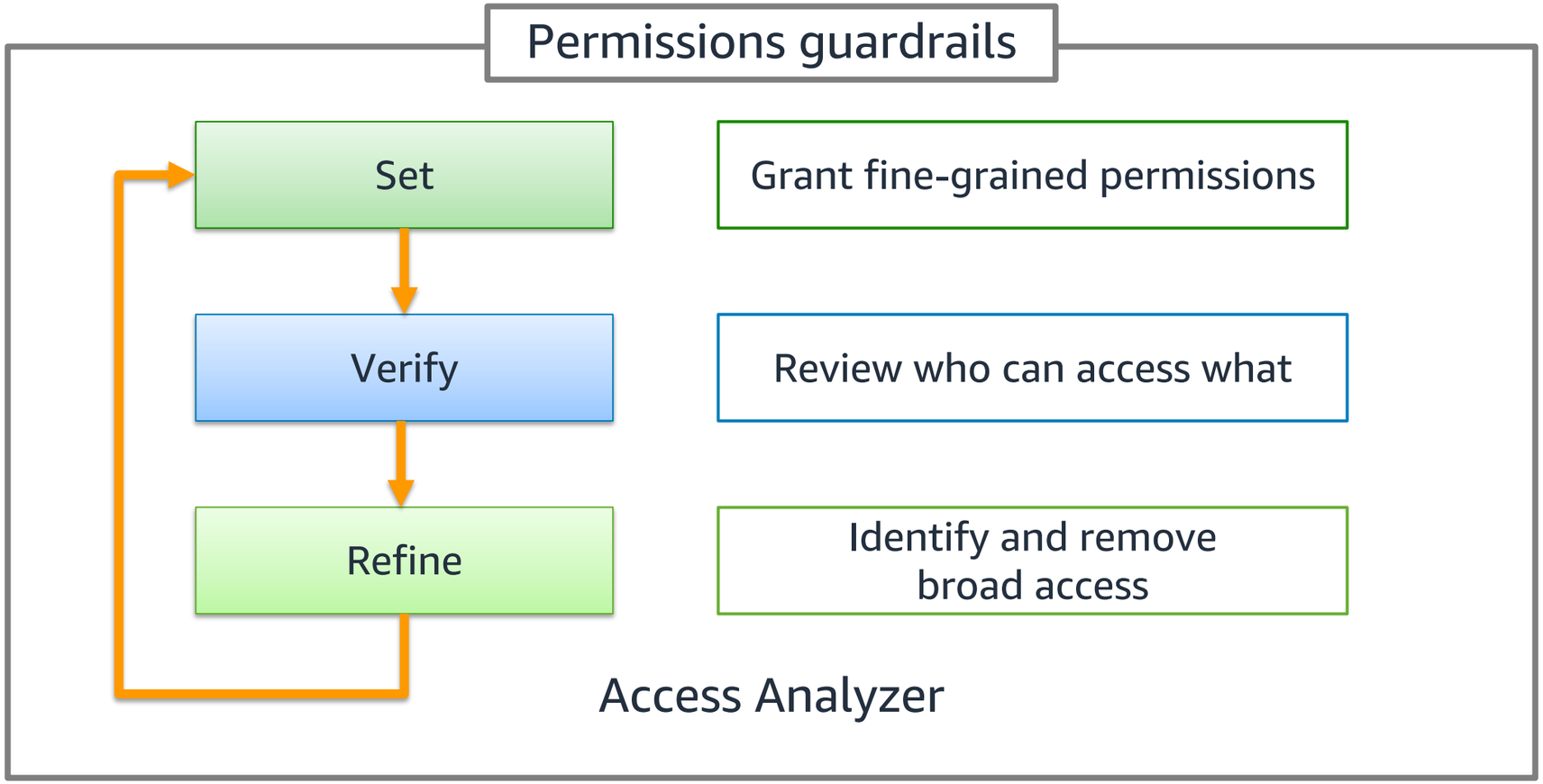
AWS IAM Access Analyzer

Access Analysis

- Prescriptive guidance (more than 100 checks) about authoring error-free policies.
- Generate fine-grained policies based on access activity found in CloudTrail.
- Validate account access before deploying permissions changes.
- Tighten permissions for IAM principals by using access history.
- Identify Secrets Manager secrets that can be accessed publicly or from other accounts or organizations.

AWS IAM Access Analyzer

Getting to the right permissions



AWS IAM Access Analyzer

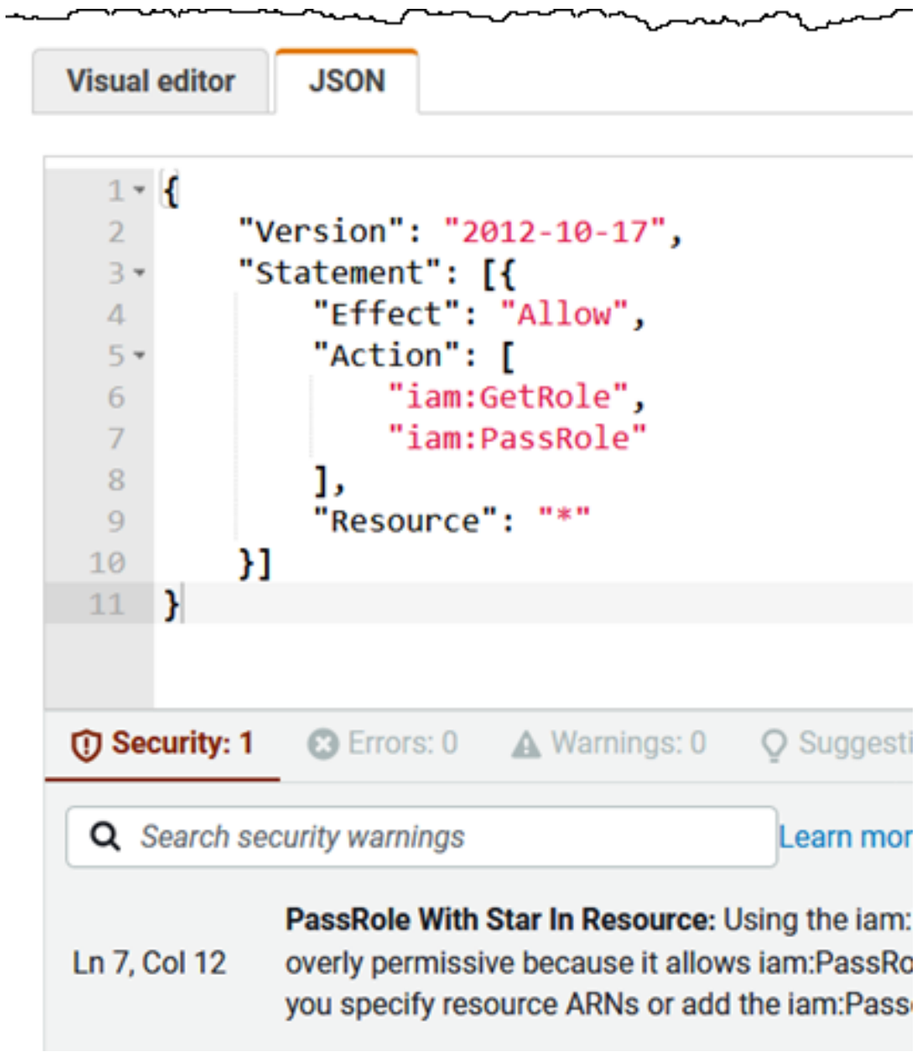
Set permissions: policy generation with Access Analyzer

- Access Analyzer helps you get to the right permissions more quickly by analyzing your access activity in CloudTrail and generating policies.
- Includes action-level policies for more than 50 popular services—with more on the way.
- Guides you to set least privilege because policies include resource-level templates.
- Supports organization trails.

AWS IAM Access Analyzer

Set permissions: policy validation with Access Analyzer

- Makes it easier to author secure and functional policies with more than 100 checks.
- Validation performed during policy authoring in the IAM console or via API.
- Each check includes guidance for resolving issues.



aws professional services

AWS IAM Access Analyzer

Refine permissions with Access Analyzer

- Public and cross-account findings with Access Analyzer guide you to ensure that the existing access meets your intent.
- Access Analyzer uses provable security to analyze all access paths and provide comprehensive analysis of external access to your resources.
- You can create an analyzer at the account or AWS Organization level. Access Analyzer continuously monitors for new or updated resource permissions to help you identify external access.
- Using this same analysis, Access Analyzer makes it easier to review and validate public and cross-account access before deploying permissions changes.

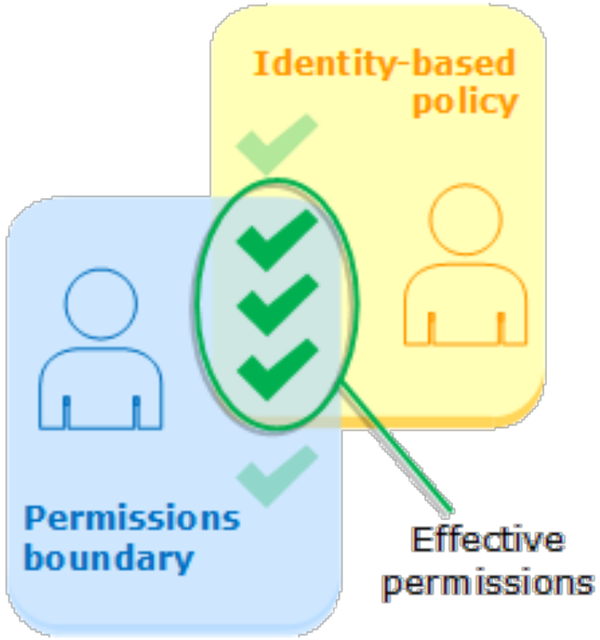
Permission Boundaries

AWS Identity & Access Management
Authentication & Authorization

aws professional services

Permissions Boundaries

- limit the maximum permissions that a principal can have
- Can be used to delegate IAM tasks but limit the permissions granted to the IAM principals they create
- Use the same policy language as regular IAM policies



Permissions Boundaries – Use Case

- **IAM Administrator**: Principal responsible for provisioning user, roles, and policies for the enterprise.
- **Delegated IAM Administrator**: Principal with delegated responsibility to provision users, roles, and policies for their business unit, team, or workload.
- **Business unit, team, or workload member**: Principal interacting with AWS to accomplish their business unit, team, or workload goals.

Permissions Boundaries – Tasks

1. IAM Administrator creates MyAppPermissionsBoundary.
2. IAM Administrator creates Delegated IAM Administrator role.
3. IAM Administrator creates Delegated IAM Administrator Permissions Boundary and Permissions Policy.
4. Attach permissions boundary and policy to role.

Permissions Boundaries – How does it work?

```
"Effect": "Allow",
            "Action": [
                "iam:DetachRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:PutRolePermissionsBoundary",
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam:::role/apps/my_app/*",
            "Condition": {
                "StringEquals": {

"iam:PermissionsBoundary":"arn:aws:iam::<account>:policy/MyAppPermissionsBoundary"
                }
            }
```

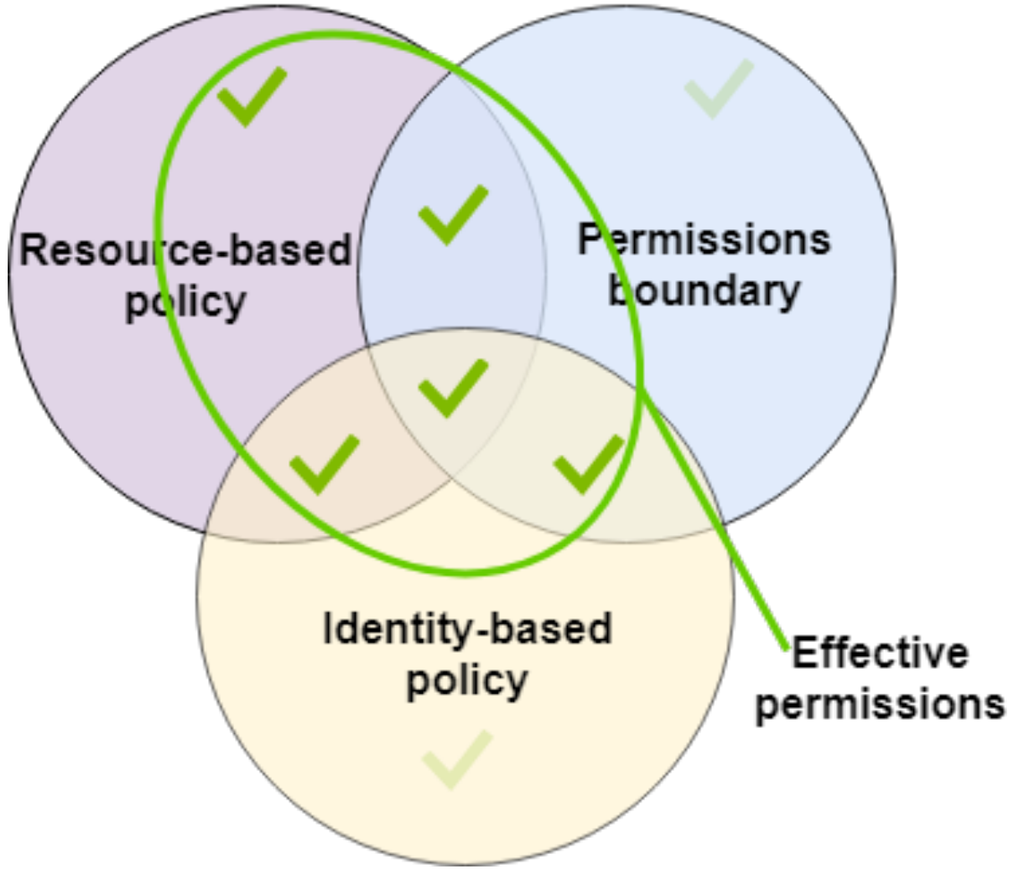
Permissions Boundaries – How do I enforce?

```
{
            "Sid": "DenyDeletePermBoundary",
            "Effect": "Deny",
            "Action": [
                "iam:DeletePolicy",
                "iam:DeleteRolePermissionsBoundary"
            ],
            "Resource": [
                "arn:aws:iam::<account>:policy/MyAppPermissionsBoundary",
                "arn:aws:iam:::policy/apps/my_app/DelegatedPermissionsBoundary",
                "arn:aws:iam::*:role/*"
            ]
}
```

Permissions Boundaries

Resource-based Policies

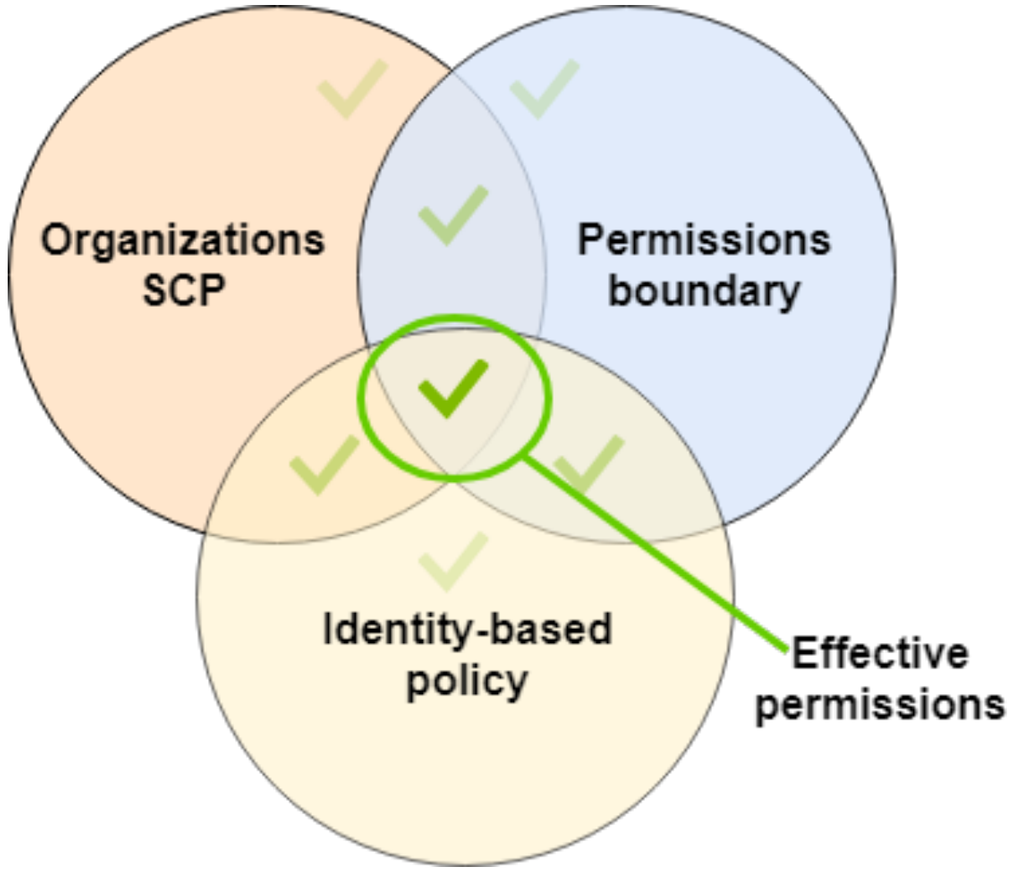
- Boundaries do not effect permissions granted through resource-based policies.
- Effective permissions consist of everything that is allowed by the resource-based
policy and everything that is allowed by both the permissions boundary and the identity-based policy.



Permissions Boundaries

Organizations SCP's

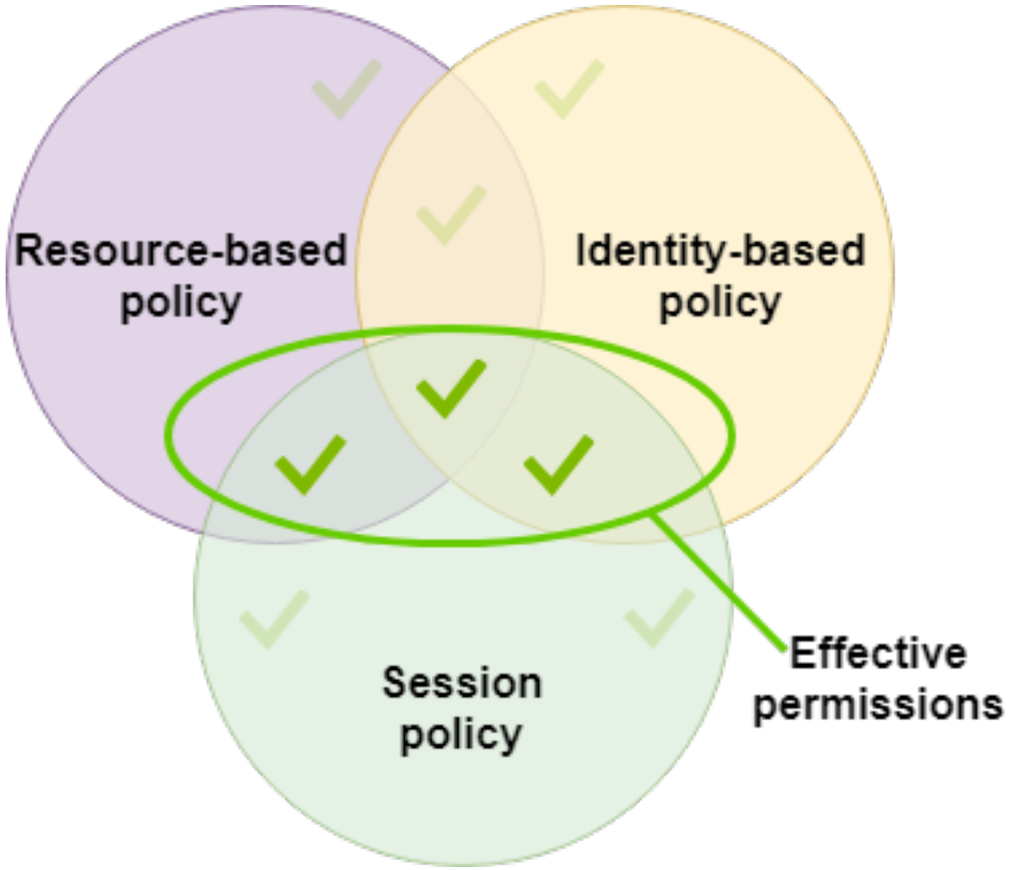
- SCP's do not grant any permissions to a principal.
- SCP's only limit the operations in an account and apply to all principals.
- Effective permissions consist of any operations that is allowed by any of the three policies.



Permissions Boundaries

Session Policy

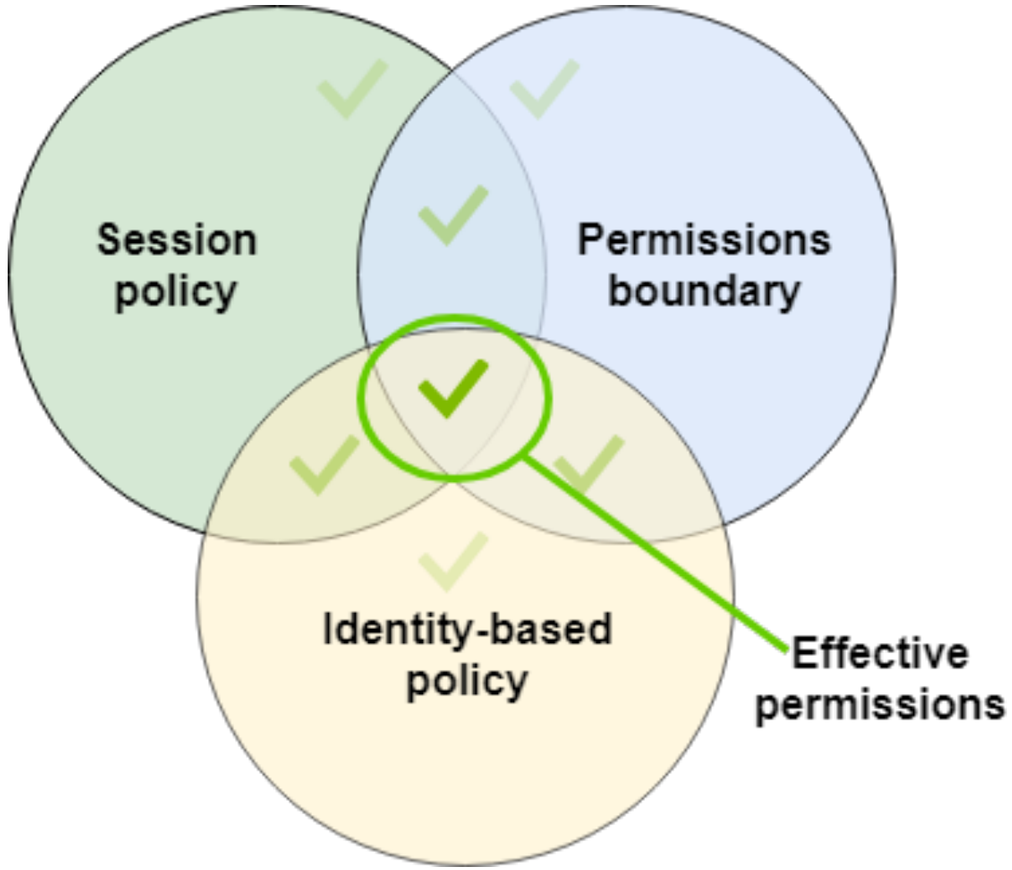
- Session policies are applied per session when a session is created.
- Effective permissions consist of any operation that is allowed by the session policy and either the identity-based policy of the principal that created the session, or an applicable resource-based policy.



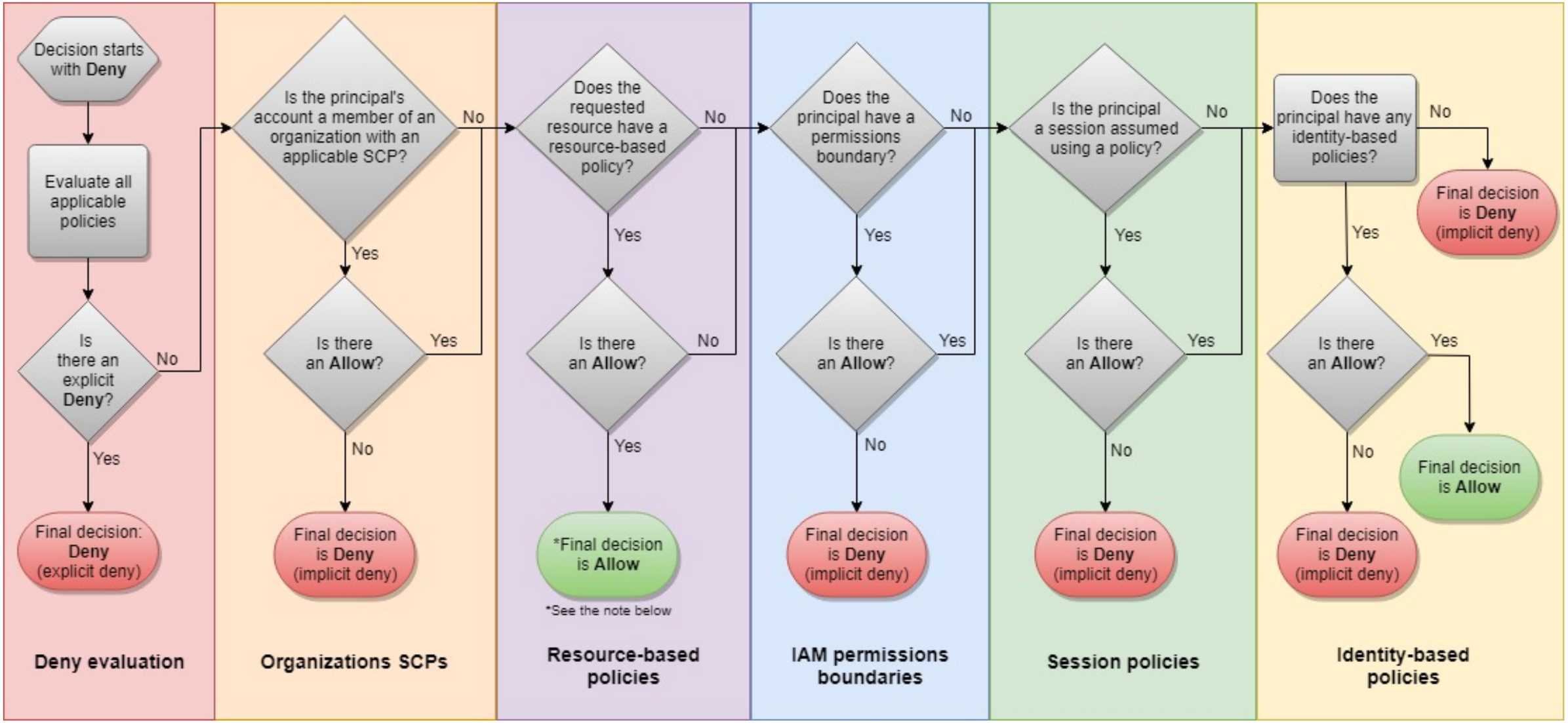
Permissions Boundaries

Session Policy with Permissions Boundary

- A permission boundary limits both the identity-based policy and the session policy.
- Effective permissions consist of any operation that is allowed by all of the three policies (and the SCP applied if applicable).



IAM Evaluation Logic



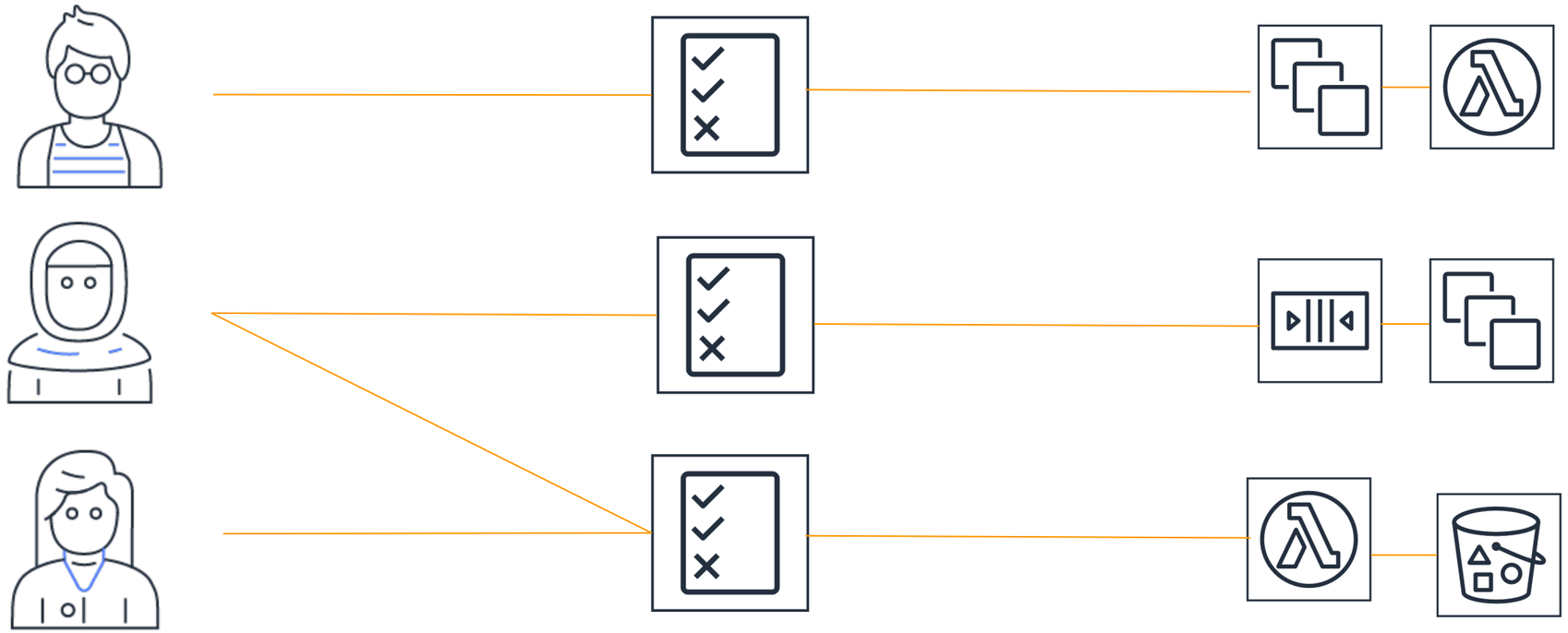
aws professional services

Attribute-based access control (ABAC)

Authentication & Authorization

aws professional services

Recall Role-based Access Control (RBAC)



Developers

Permissions

Resources

ABAC - A little bit about attributes

Attributes are a **key** or a **key and value** pair

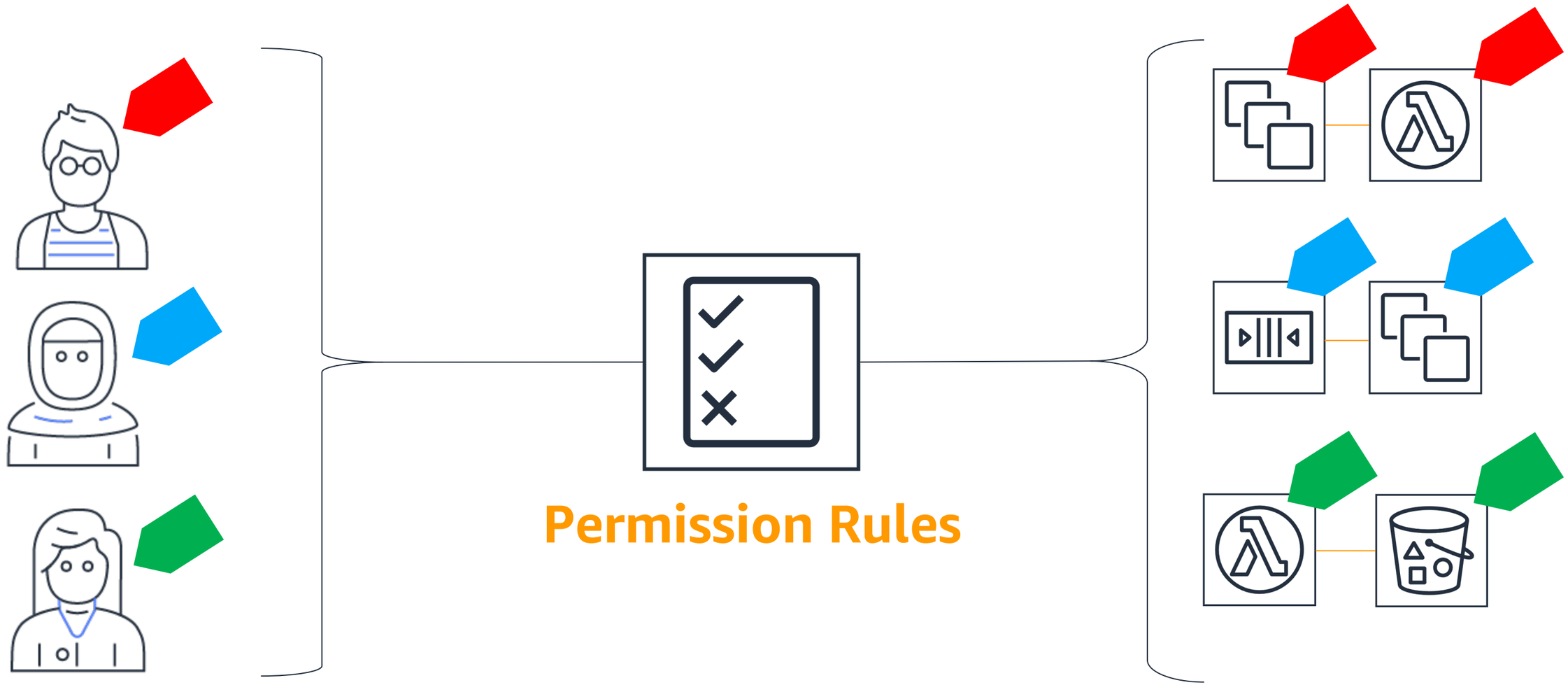
Pre-defined by a provider or custom

Examples

UserID = ziggy
Team = Unicorns
Project = Pickles

Project = Pickles
Env = Development
CreatedBy = ziggy

ABAC - A scalable permissions model based on attributes



aws professional services

Benefits of ABAC

Permissions scale with innovation

Teams move fast as permissions automatically apply

Granular permissions without a permission update for every item

Audit attributes to determine access

Examples of attribute-based permissions

- Grant developers read and write access to their project resources
- Require developers to assign their project to new resources
- Grant developers read access to resources common to their team
- Manage only the resources I own

Steps to implement ABAC

1

Create identities with access control attributes

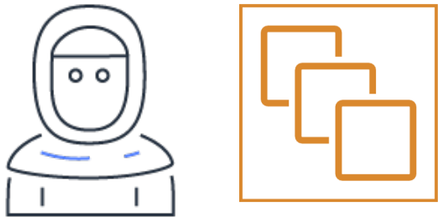
2

Require attributes for new resources

3

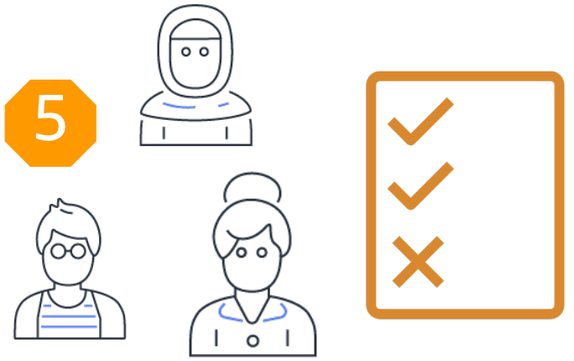
Set permissions based on developer attributes

4



Create new resources

5



Permissions automatically apply to Developers as workloads scale

aws professional services

ABAC best practices

1. Reserve a subset of attributes used for access control

2. Only approved entities can set or modify attributes

3. Tag everything during creation so that permissions apply immediately

4. Rely on attributes to grant permissions to manage resources

5. Periodically audit to ensure that resources are tagged appropriately

Roles & Responsibilities

AWS Identity & Access Management

Authentication & Authorization

aws professional services

Roles & Responsibilities

What roles are needed?

- Human roles are needed for human operators to assume into the AWS account to use services.
- System roles may be needed for your applications running on EC2 instances or for AWS services to utilize your AWS resources.

Roles & Responsibilities

What roles are needed?

- Consider the current roles in your organization, and compare to the end-goal job roles your organization wants to achieve (i.e. DevOps).
- Consider increasing business agility by removing operational blockers
- Consider separation of duties and least privilege

Roles & Responsibilities

What roles are needed?

- Administrator
- IAMAdmin
- NetworkAdmin
- SystemOperator
- Developer
- SecurityOperator
- ReadOnlyAnalyst
- FinanceAdmin
- ComplianceAnalyst
- StarPortalWebTier
- StarPortalAppTier
- StarPortalDataTier
- ApplicationDelivery

Authorization – Identity Recap

Govern your AWS environment	**with**	AWS accounts and AWS Organizations
Centralize identity management	**with**	AWS SSO
Rely on short-term credentials	**with**	federation and application roles
Organize your resources	**with**	tags and resource groups
Establish a data perimeter	**with**	policies and conditions
Journey to least privilege	**with**	AWS Identity and Access Management (IAM) policies and IAM Access Analyzer

Authorization – Permission Quiz

Service control policies (SCPs)

Permissions boundaries

IAM permissions policy

Scoped-down policies

Resource-based policies

Endpoint policies

Restrict powerful actions except for admins

Grant only the actions your role uses

Developers can manage roles safely

Grant direct, cross-account access

aws professional services

Authorization – Use the right permissions tool

Service control policies (SCPs)

Permissions boundaries

IAM permissions policy

Scoped-down policies

Resource-based policies

Endpoint policies

Restrict powerful actions except for admins

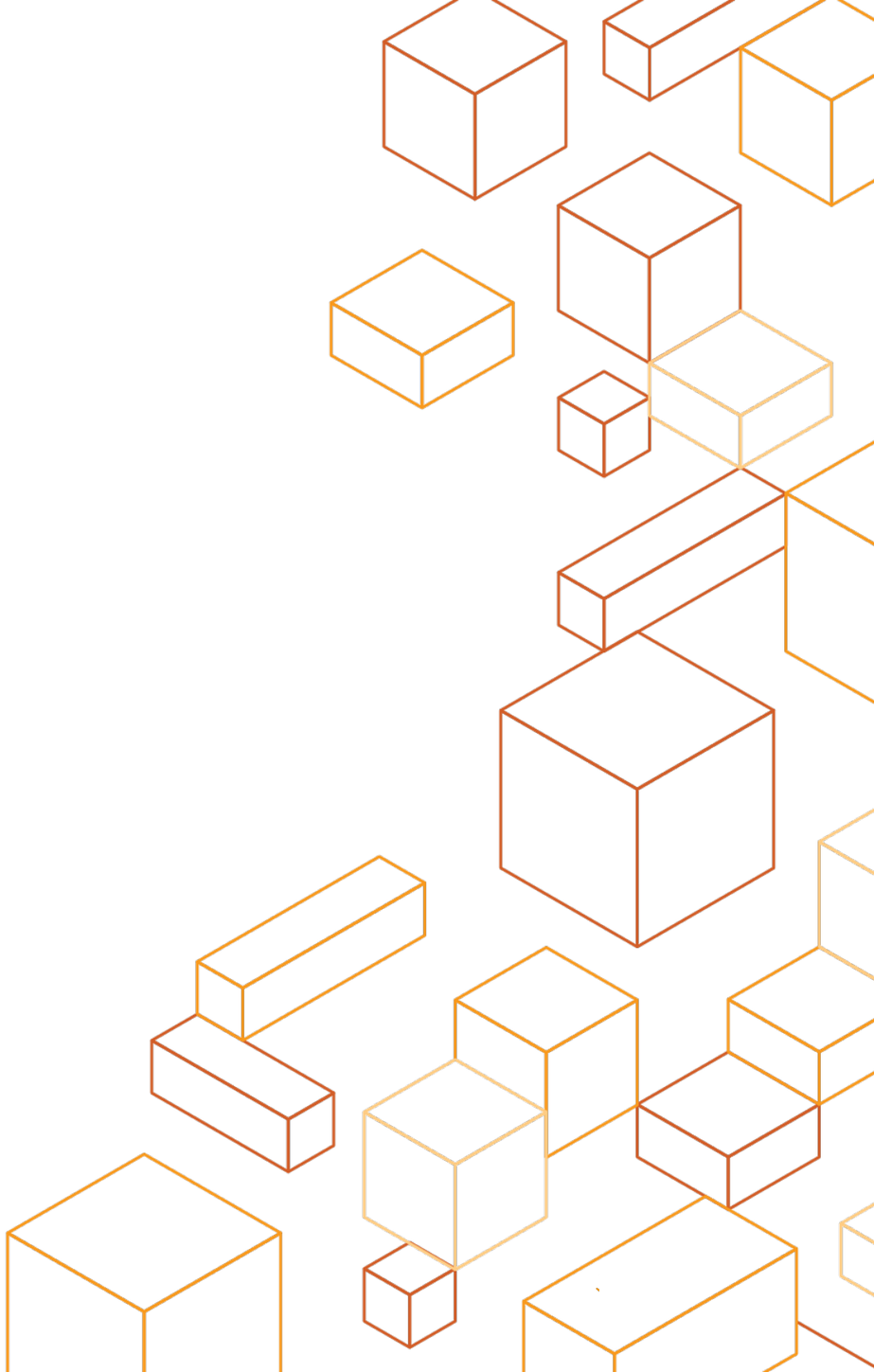
Grant only the actions your role uses

Developers can manage roles safely

Grant direct, cross-account access

aws professional services

Questions



Appendix A – Example IAM Naming Convention

AWS Identity & Access Management

Authentication & Authorization

aws professional services

Example IAM User Naming Convention

IAM User Names

- Syntax
  - [<organization/business unit short name>][<team/project name>]<product/technology>
  - <user alias>
- Examples
  - TeamASecMonitoring
  - TeamBSecMonitoring
  - AppMonitoring
  - JohnDoe

Example IAM Group Naming Convention

IAM Group Names

- Syntax
  - [<organization/business unit short name>][<team/project name>]<function name>
- Examples
  - CompanyTeamEngineering
  - Developers
  - Admins

Example IAM Role Naming Convention

IAM Role Names

- Syntax
  - [<organization/business unit short name>]<team/project name><entity name>
  - [<organization/business unit short name>]<team/project name><app name><app tier>
- Examples
  - CompanyTeamDevelopers
  - Developers
  - TeamAStarPortalWebTier
  - StartPortalAppTier

Appendix B –IAM Best-Practices

AWS Identity & Access Management

Authentication & Authorization

aws professional services

IAM Best-Practices

- Lock away your AWS account (root) access keys
- Create individual IAM users
- Use groups to assign permissions to IAM users
- Grant least privilege
- Configure a strong password policy for your users
- Enable MFA for privileged users

IAM Best-Practices

- Use roles for applications that run on Amazon EC2 instances
- Delegate by using roles instead of by sharing credentials
- Rotate credentials regularly
- Remove unnecessary credentials
- Use policy conditions for extra security
- Monitor activity in your AWS account

More info: http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Appendix C – AWS Services

AWS Identity & Access Management

Authentication & Authorization

aws professional services

AWS Services That Work With IAM

The following webpage describes the IAM permission types each service supports, specifically:

- Action-level permissions
- Resource-level permissions
- Resource-based policies
- Tag-based authorization
- Requests via temporary security credentials
- Service-linked roles

http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html