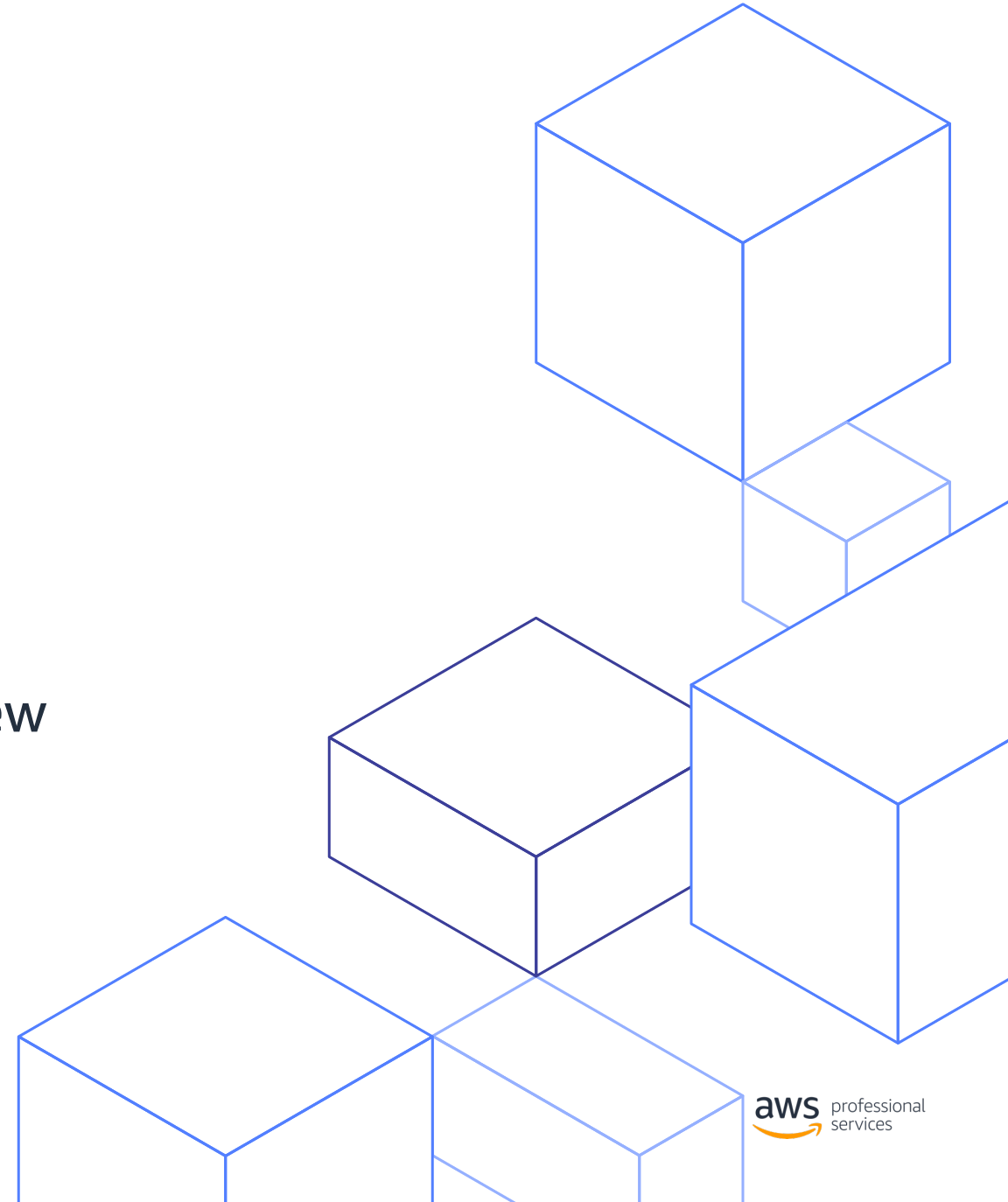
Identity and Access Management

Overview / Auth

Jinsung Heo

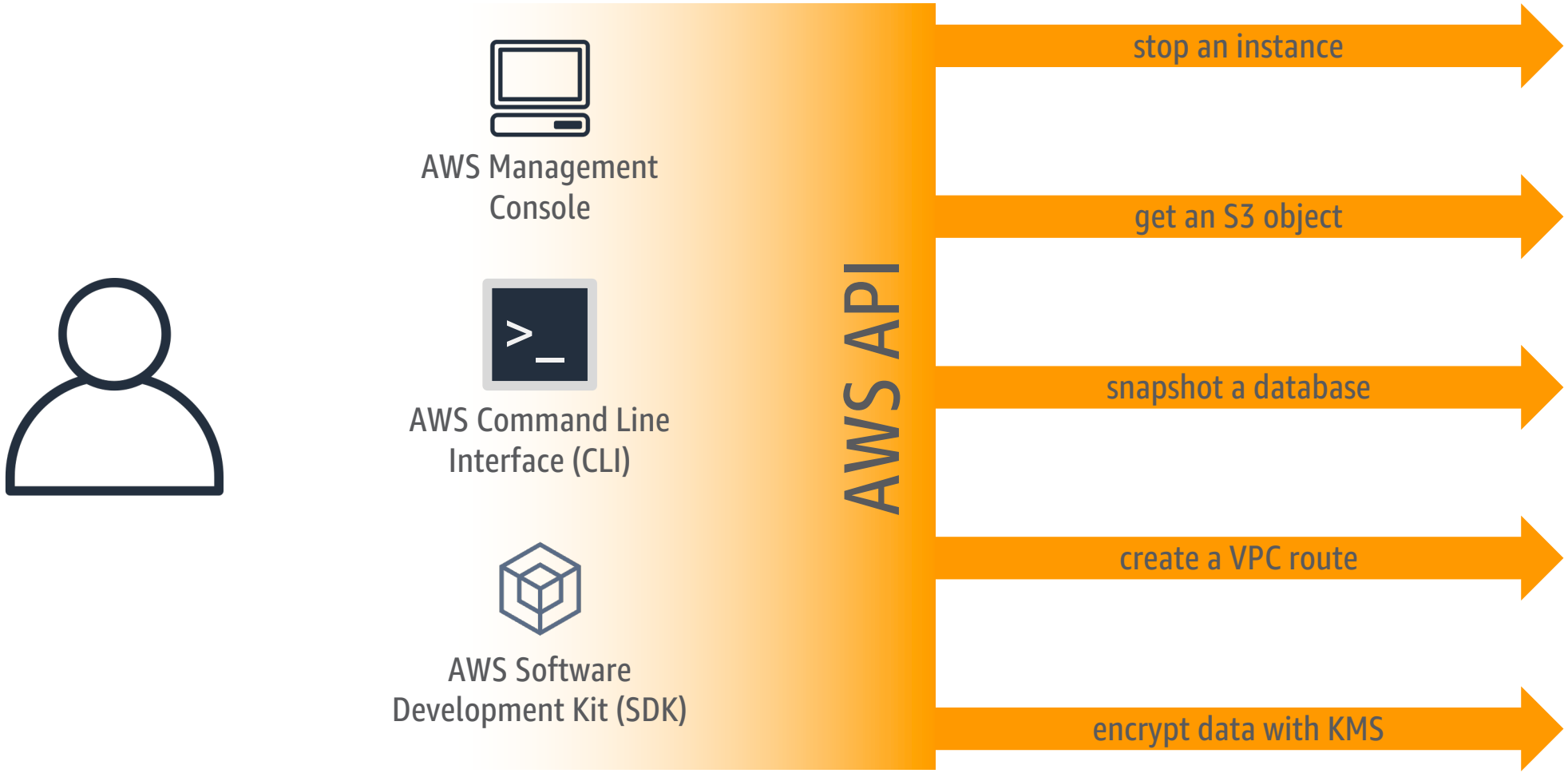
AWS API Calls

AWS Identity & Access Management Overview

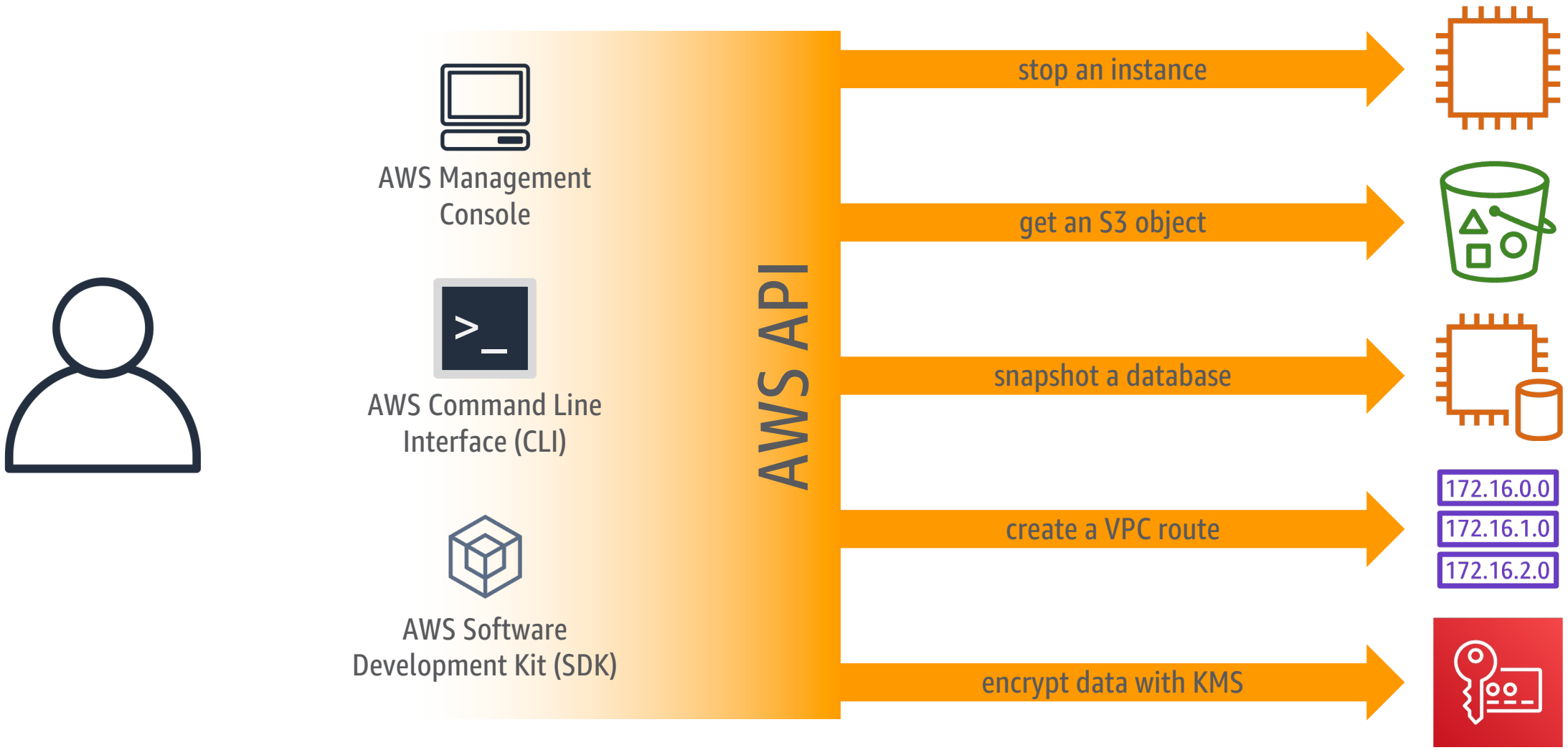


aws professional services

Making API Calls

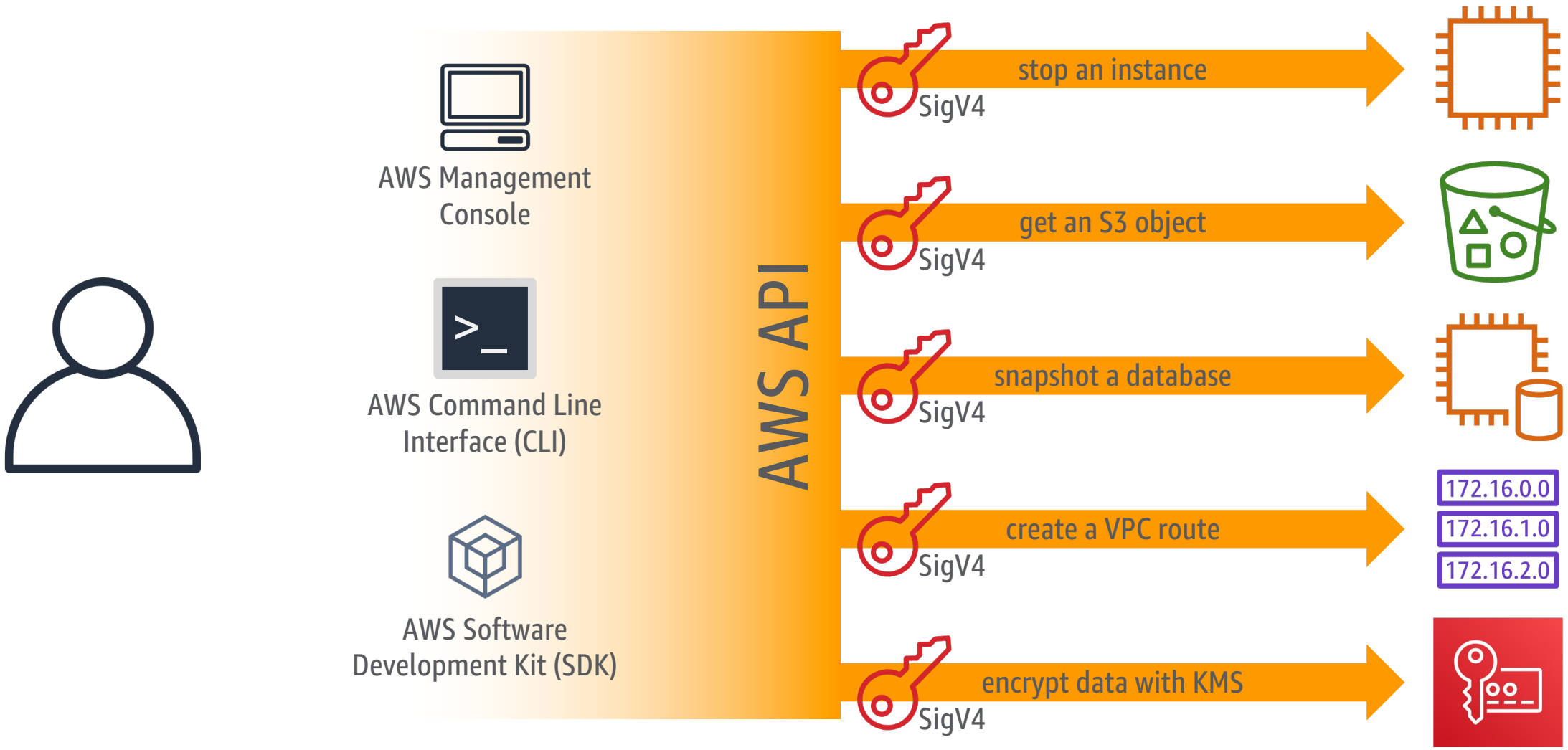


Making API Calls



aws professional services

Making API Calls



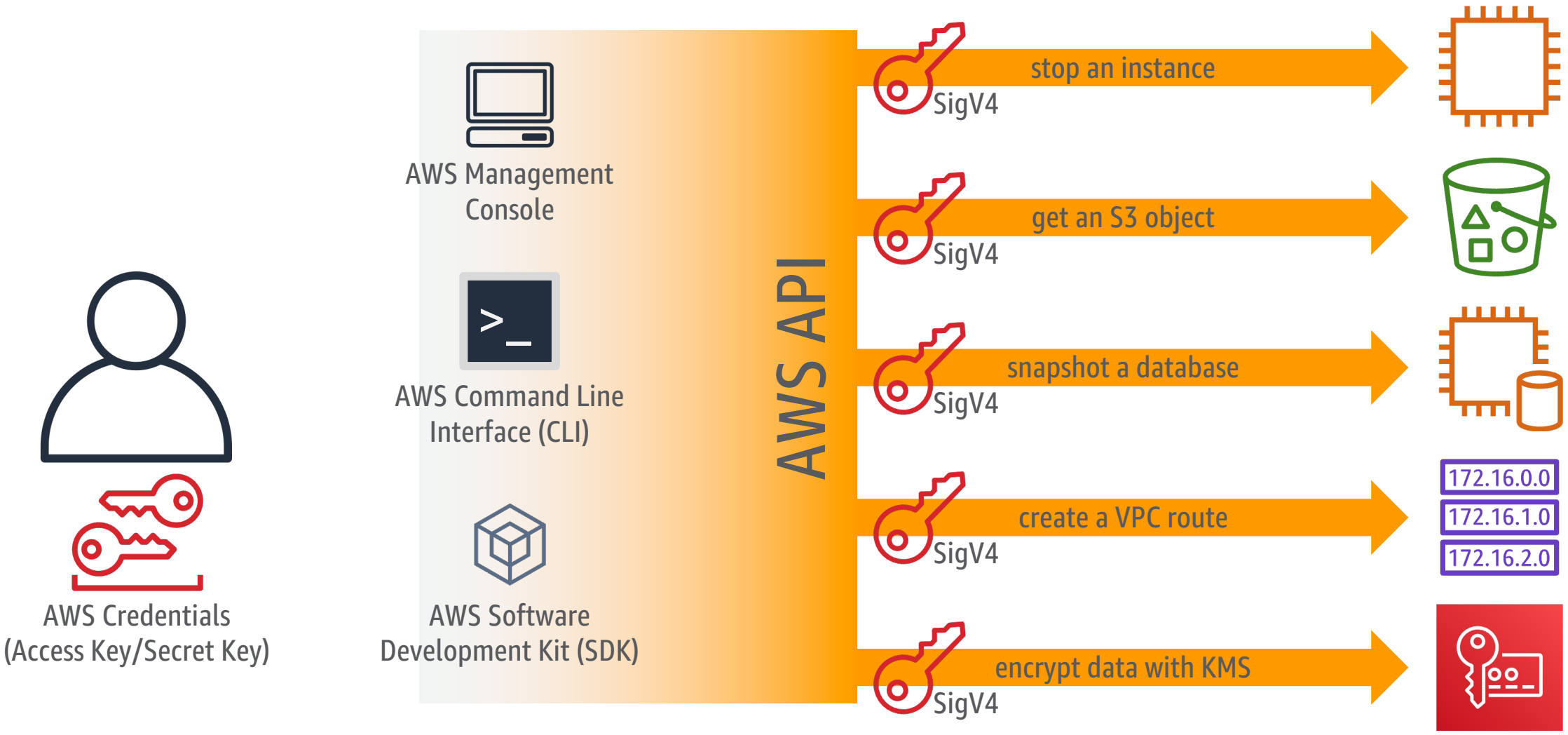
aws professional services

AWS Signature Version 4

- AWS Signature Version 4 is the process to add authentication information to AWS requests.
    - The AWS SDKs or CLI tools will construct, sign, and send requests for you, with the **access keys** you provide.
    - If you are constructing AWS API requests yourself, you will have to include code to sign the requests.

More information can be found here:
http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

Making API Calls



aws professional services

AWS Identity & Access Management

AWS Identity & Access Management Overview

aws professional services

AWS IAM Concepts

AWS Accounts

- Strong separation of duties
- Consolidate billing into a single account
- Plan your account strategy in advance (e.g. per function, per criticality, etc.)

AWS IAM Concepts

AWS Resources

Defined uniquely by an **Amazon Resource Name** (ARN)

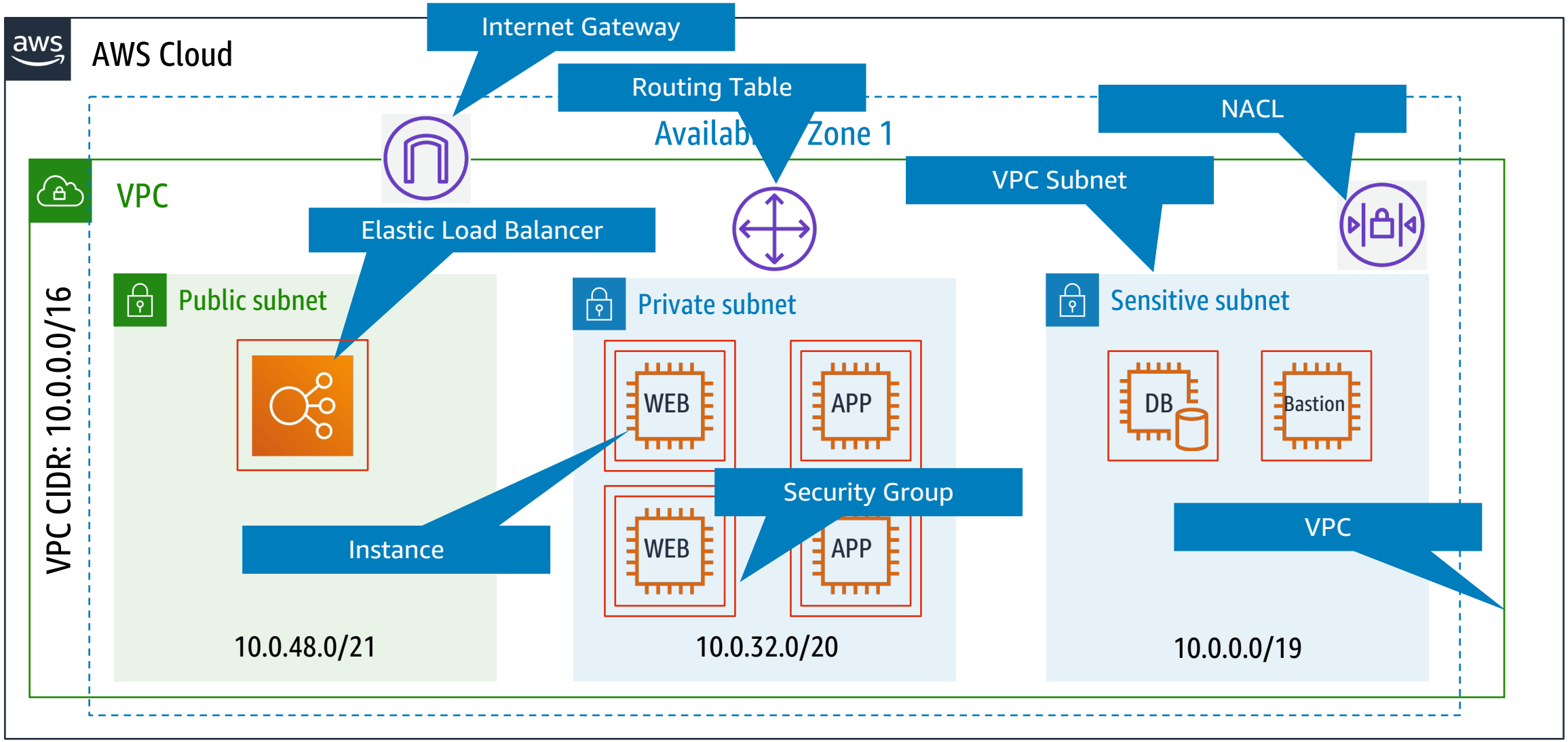
Ex: EC2 instance, DynamoDB table, IAM user, etc.

Not: OS installed on EC2, data inside an EBS volume, etc.

```
arn:aws:service:region:account:resource
```

```
<!- Amazon EC2 instance -->
arn:aws:ec2:us-east-1:123456789012:instance/i-1a2b3c4d

<!-- Amazon RDS tag -->
arn:aws:rds:eu-west-1:123456789012:db:mysql-db

<!-- Amazon S3 all objects in a bucket -->
arn:aws:s3:::my_corporate_bucket/*
```

AWS IAM Concepts



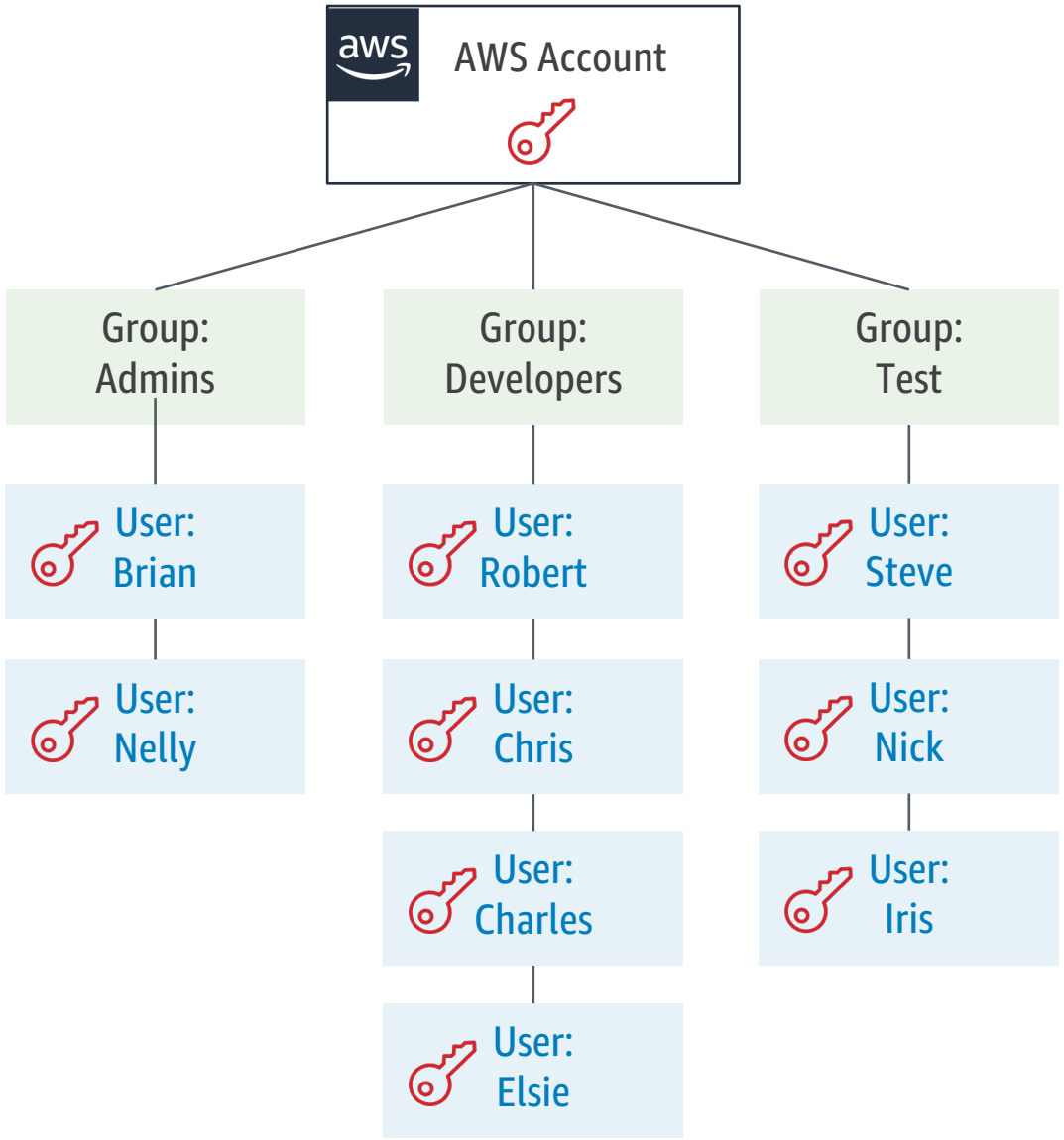
aws professional services

AWS IAM Concepts - Resources

```
<-- S3 Bucket -->
"Resource":"arn:aws:s3:::my_corporate_bucket/*"

<-- SQS queue-->
"Resource":"arn:aws:sqs:us-west-2:123456789012:queue1"

<-- Multiple DynamoDB tables -->
"Resource":["arn:aws:dynamodb:us-west-2:123456789012:table/books_table",
          "arn:aws:dynamodb:us-west-2:123456789012:table/magazines_table"]

<-- All EC2 instances for an account in a region -->
"Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*"
```

AWS IAM Concepts

- A username for each user
- Groups to manage multiple users
- Centralised access control
- Optional provisions:
    - Password for console access
    - Policies to control access
    - Use Access Key to sign API calls
    - Multifactor Authentication



What is AWS Identity and Access Management (IAM)?

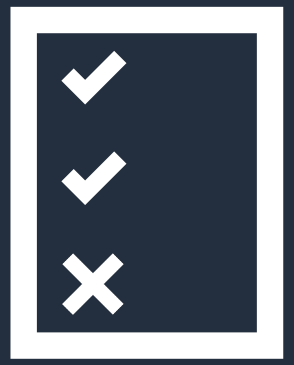
AWS IAM helps you **securely control access to AWS resources**. You use IAM to control who is **authenticated** (signed in) and **authorized** (has permissions) to use resources.

- Free to use
- Shared access to your AWS account
- Granular permissions
- Secure access to AWS resources
- Multi-factor authentication (MFA)
- Identity federation
- Integrated with many AWS services
- Eventually Consistent

https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

Identity, access, and resource management

Who can access what



Identity management

Name
Credentials
Metadata

Access management

Policies
Compliance

Resource management

Isolation
Grouping
Tagging
Sharing

AWS identity management – Who

- **IAM user**
  - Represents a person or application
  - Not a separate account – user within your account
  - Individual passwords and access keys
- **IAM group**
  - Collection of IAM users
- **IAM role**
  - Not tied to one individual
  - Can be assumed by anyone who needs it

AWS access management – Can access

AWS Identity and Access Management (IAM)

Identity-based policies

Resource-based policies

Two AWS policy types shown

Policies define permissions

AWS resource management – What

AT CLOUD SCALE FOR . . .

Hundreds of AWS accounts

Thousands of Amazon EC2 instances

Billions of Amazon S3 objects

Trillions of Amazon DynamoDB items

And thousands of workloads

Identity and access management

Define, enforce, and audit user permissions across AWS services, actions, and resources

AWS Identity and Access Management (IAM)

Securely manage access to AWS services and resources

AWS IAM Identity Center (previously AWS SSO)

Centrally manage workforce access to multiple AWS accounts and business apps

AWS Directory Service

Managed Microsoft Active Directory in AWS

Amazon Cognito

Add user sign-up, sign-in, and access control to your web and mobile apps

AWS Organizations

Policy-based management for multiple AWS accounts

AWS Resource Access Manager

Simple, secure service for sharing AWS resources

IAM Users

- **AWS account root user**
  - When you create an AWS account, you create an AWS account root user identity.
  - Root user credentials: email address and password that you provided when creating the account.
  - It has complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password.
- **AWS IAM user**
  - Person or application requiring access to AWS services.
  - By default, a new IAM user has no permissions to do anything.

Root Account Best Practices

- Create an IAM admin user and user group
- Limit the tasks you perform with the root user
- Lock away your AWS account root user access keys
- Use a strong password; Configure a strong password policy
- Enable Multi-Factor Authentication (MFA)
- Configure account security challenge questions
- Configure account alternate contacts

https://docs.aws.amazon.com/IAM/latest/UserGuide/root-user-best-practices.html

Tasks That Require Root User Capabilities

- Change your account settings:
  - email, root user password, and root user access keys
- Restore IAM user permissions
- Activate IAM access to the Billing and Cost Management Console
- Close your AWS account
- Register as a seller in the Reserved Instance Marketplace
- Sign up for GovCloud

https://docs.aws.amazon.com/IAM/latest/UserGuide/root-user-tasks.html

AWS IAM Concepts - Roles

- Set of permissions granted to a **trusted entity**
- Assumed by IAM users, applications or AWS services like EC2
    - Use case:
    - Cross-services
    - Temporary access
    - Cross-account
    - Federation
- Benefits
    - Security: no sharing of secrets
    - Control: revoke access anytime

AWS IAM Concepts - Roles: Breakdown of IAM role use cases

Grant Access to AWS Services

Federate Identities into AWS

Enable Cross Account Access

AWS IAM Concepts - Roles: Grant Access to AWS Services

Roles for EC2

- Enable applications running on EC2 to make AWS API calls.
- Manage security credentials for you.
- Rotates credentials automatically.
- Easy to attach and detach an IAM role to a new or existing instance.
- Add/update permissions without logging into the instance.

Service Roles

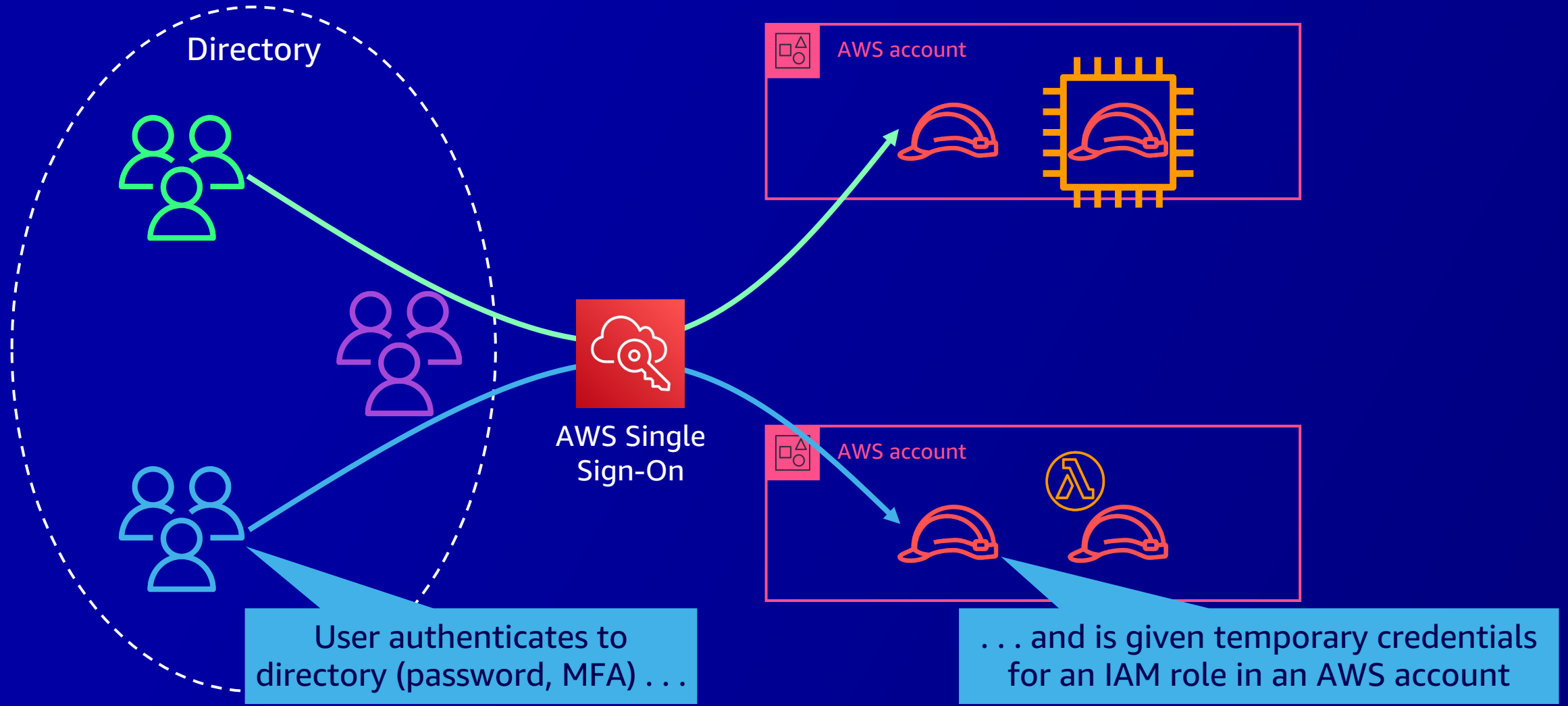
- Grant AWS services access to perform actions on your behalf.
- Control permissions that service can run.
- Track actions AWS services perform on your behalf using CloudTrail.
- Examples: AWS Config, AWS OpsWorks, and AWS Directory Service.

Service-linked Roles

- Grant AWS services access to perform actions on your behalf.
- ***Pre-defined*** permissions that the linked service requires.
- Protect you from inadvertently deleting a role.
- Track actions AWS services perform on your behalf using CloudTrail.
- Examples: Amazon Lex

Identity federation in AWS

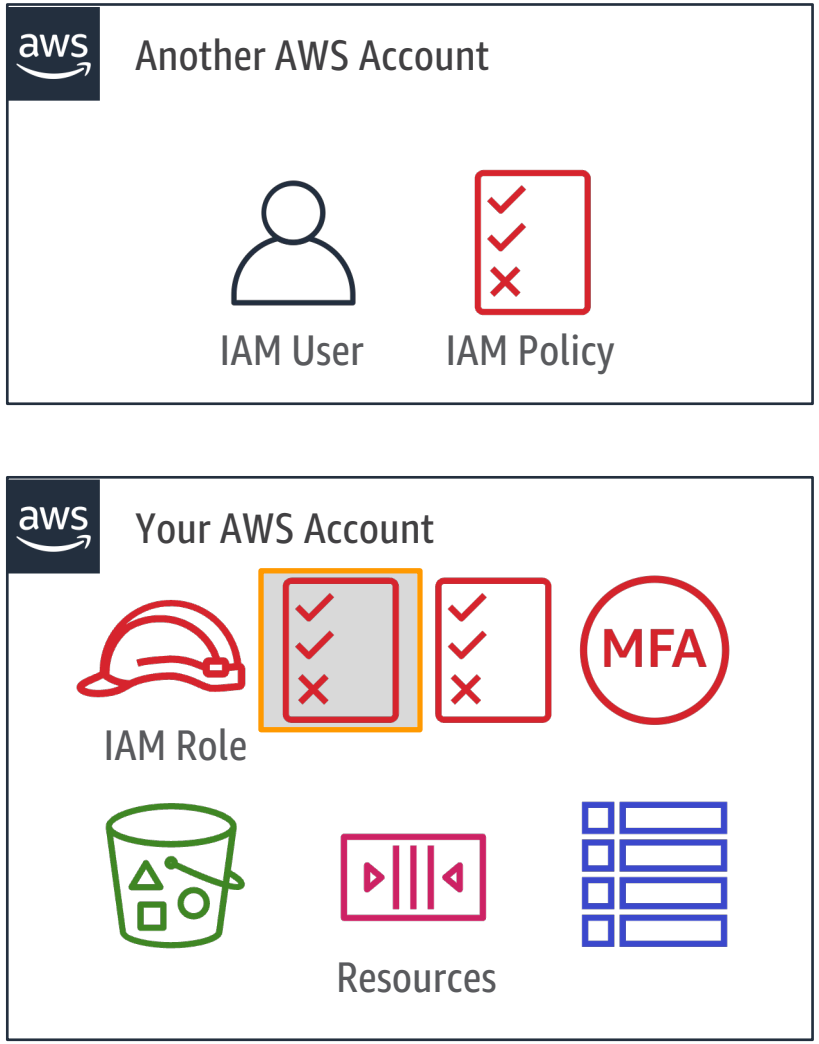
"WHO ARE YOU, REALLY?"



Identity Federation with AWS IAM Identity Center

- Centrally manage federated access with AWS IAM Identity Center (previously AWS SSO)
- Grant access without having to create IAM users
- Identity providers include Okta Universal Directory, Azure Active Directory (AD), Microsoft Active Directory, or a SAML 2.0-compatible identity provider
- Web identity providers such as Amazon, Facebook, Google, or other any OpenID Connect (OIDC)-compatible provider
- No long-term credentials

AWS IAM Concepts - Roles



Create an **IAM Role**

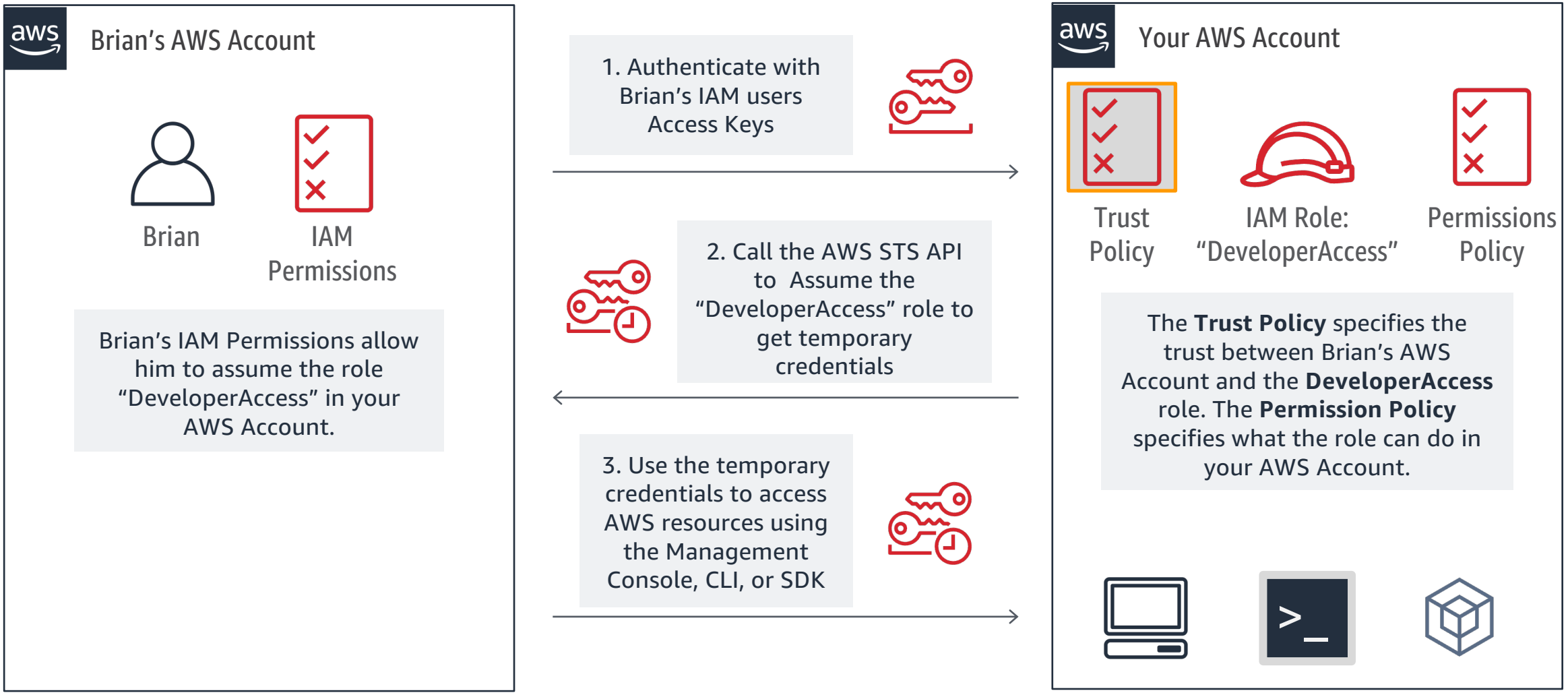
Trust Policy: Trust another AWS Account

Permission Policy: Grant Permissions

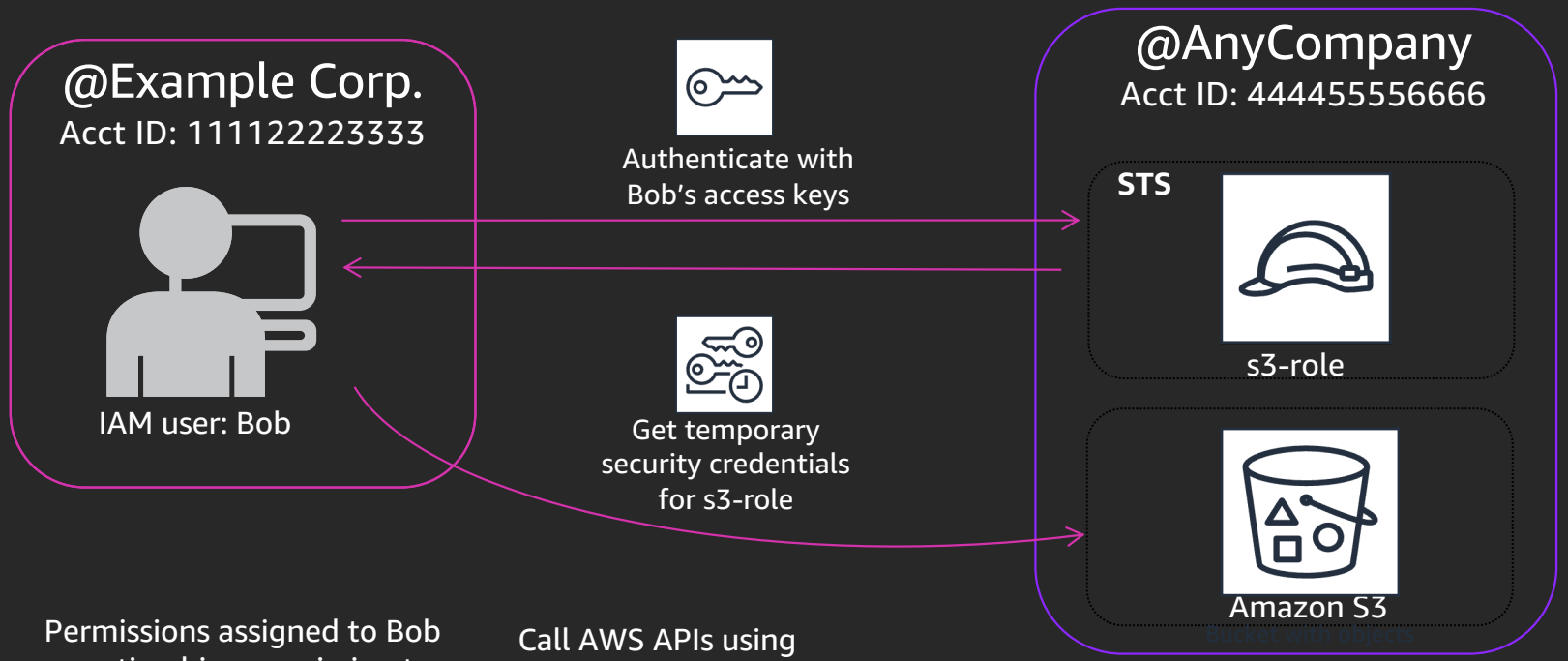
Another account's **IAM user** can assume the role if his **Permission Policy** allows him to

Use **MFA** to protect role assumption for privileged access

AWS IAM Concepts - Roles



Delegate access to contractor



s3-role trusts IAM users from the Example Corp. account (111122223333**)**

```
{ "Statement": [
  {
   "Effect":"Allow",
   "Principal":{"AWS":“1111222233334444"},
   "Action":"sts:AssumeRole"
}]}
```

Permissions assigned to Bob granting him permission to assume s3-role in account AnyCompany

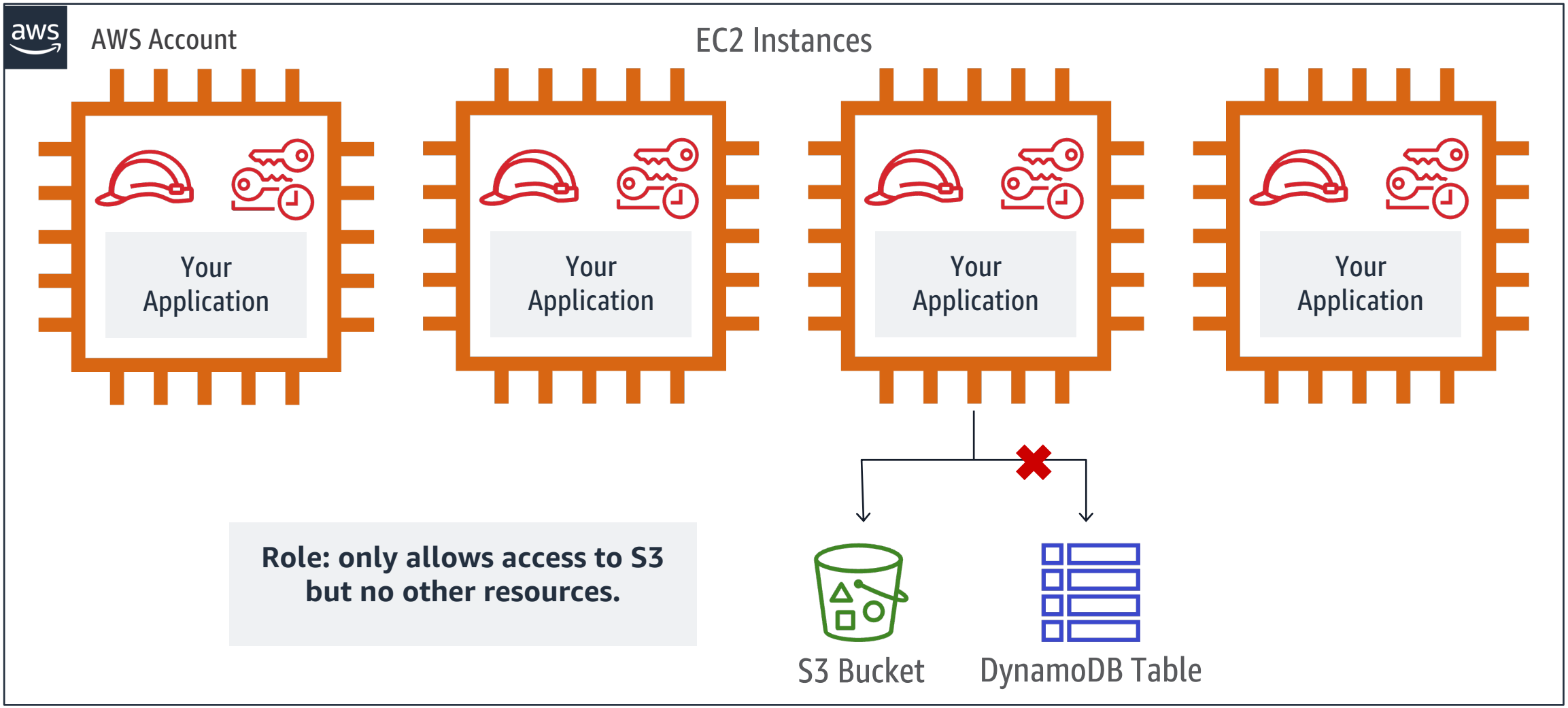
```
{ "Statement": [
  {
   "Effect": "Allow",
   "Action": "sts:AssumeRole",
   "Resource":
       "arn:aws:iam::444455556666:role/s3-role“}
]}
```

Call AWS APIs using temporary security credentials of s3-role

Permissions assigned to s3-role

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::ExampleCorp"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject"      ],
      "Resource": ["arn:aws:s3:::ExampleCorp/*"
}]}
```

AWS IAM Concepts - Roles



aws professional services

AWS IAM Concepts - Roles: Best practices

Trust: Validate your trust policies to ensure only appropriate entities can assume the role.

Permissions: Set granular permissions on your IAM roles an use scope down policies to further control access.

Tracking and Monitoring: Track cross account access using AWS CloudTrail and IAM Access Analyzer

Analogy

Account Owner ID (Root Account)

- Access to all subscribed services.
- Access to billing.
- Credentials can't be disabled.
- Access to console and APIs.

DO NOT USE after initial set-up

Door Key
Keys to the Kingdom

IAM Users

- Access to specific services.
- Access to console and/or APIs.
- Credentials can be revoked and invalidated.

Employee ID Badge

Temporary Security Credentials / IAM Roles

- Access to specific services.
- Access to console and/or APIs.

Hotel Key

aws professional services

AWS IAM Concepts

- Permissions
  - Authorize (or not) to perform an action
  - Use Policies to grant permission
- Policy
  - Set of instructions which define permission
  - Can be simple or very granular

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[ "s3:ListBucket" ],
         "Resource": “*”
      }
   ]
}
```

Authentication in AWS IAM

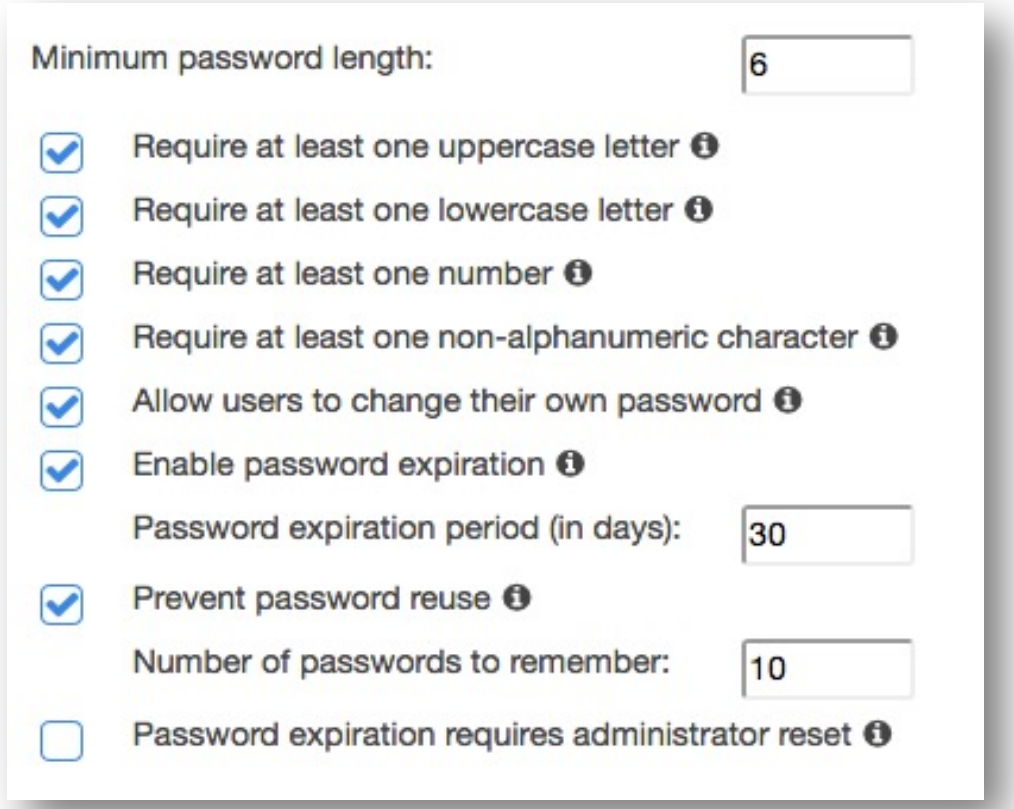
AWS Identity & Access Management
Authentication & Authorization

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)

aws professional services

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

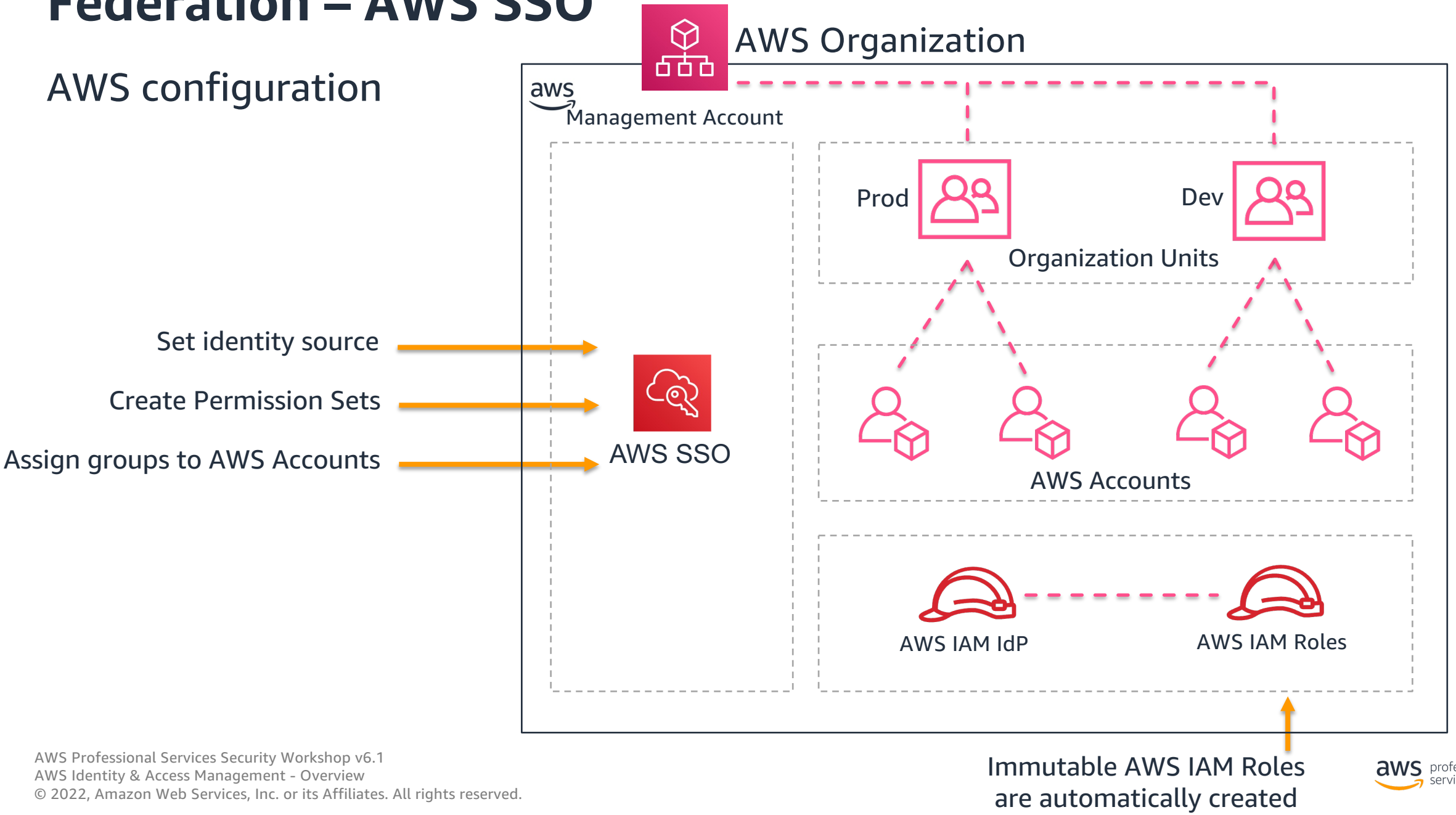
Federation – AWS SSO

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

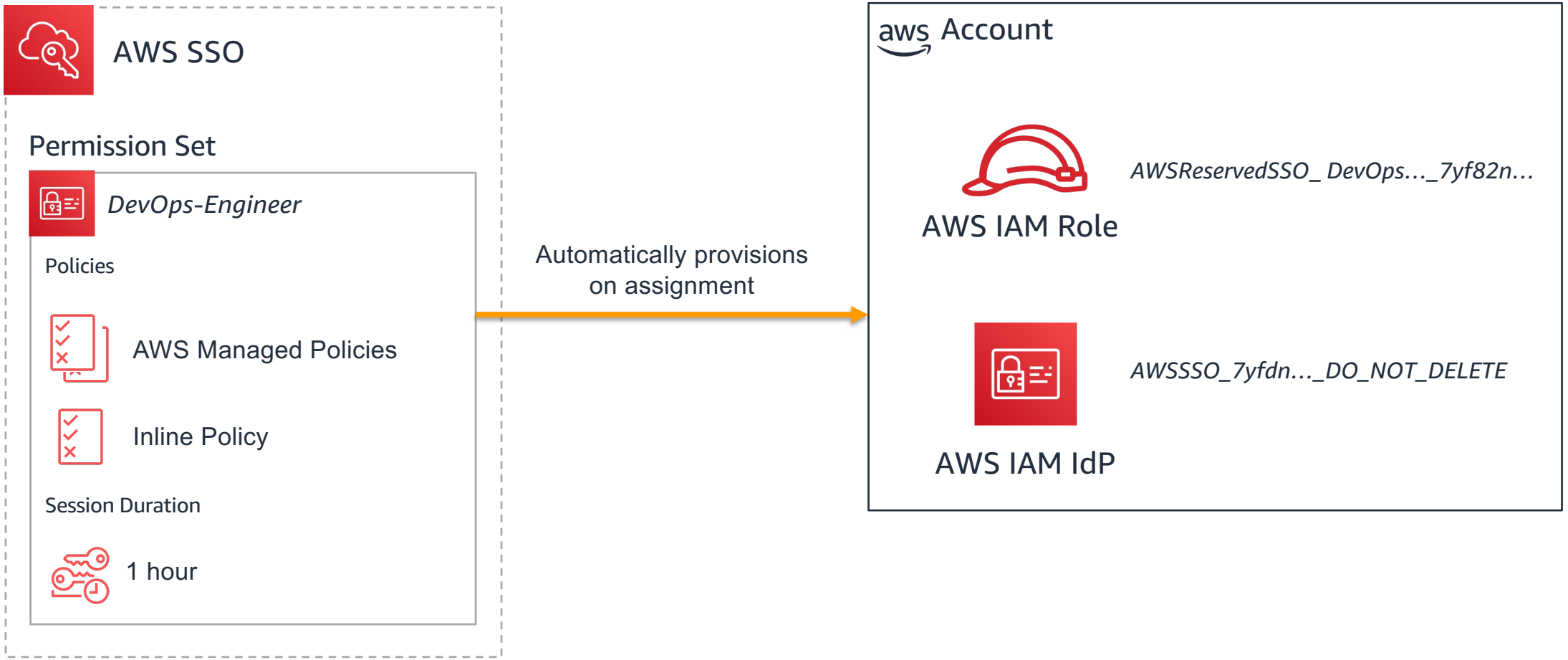
AWS configuration



aws professional services

Federation – AWS SSO

Permission Sets are Role definitions



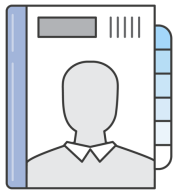
aws professional services

Federation – AWS SSO with external IdP

IdP configuration

External IdP

Configure the relying party *Configure SCIM provisioning*



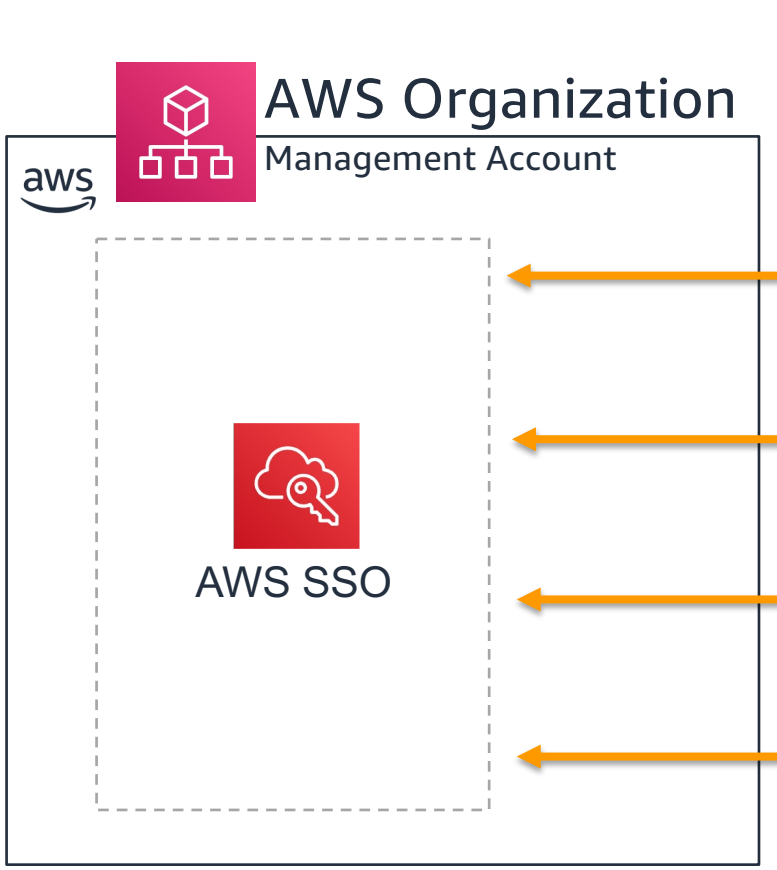
User Directory

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

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

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 – 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

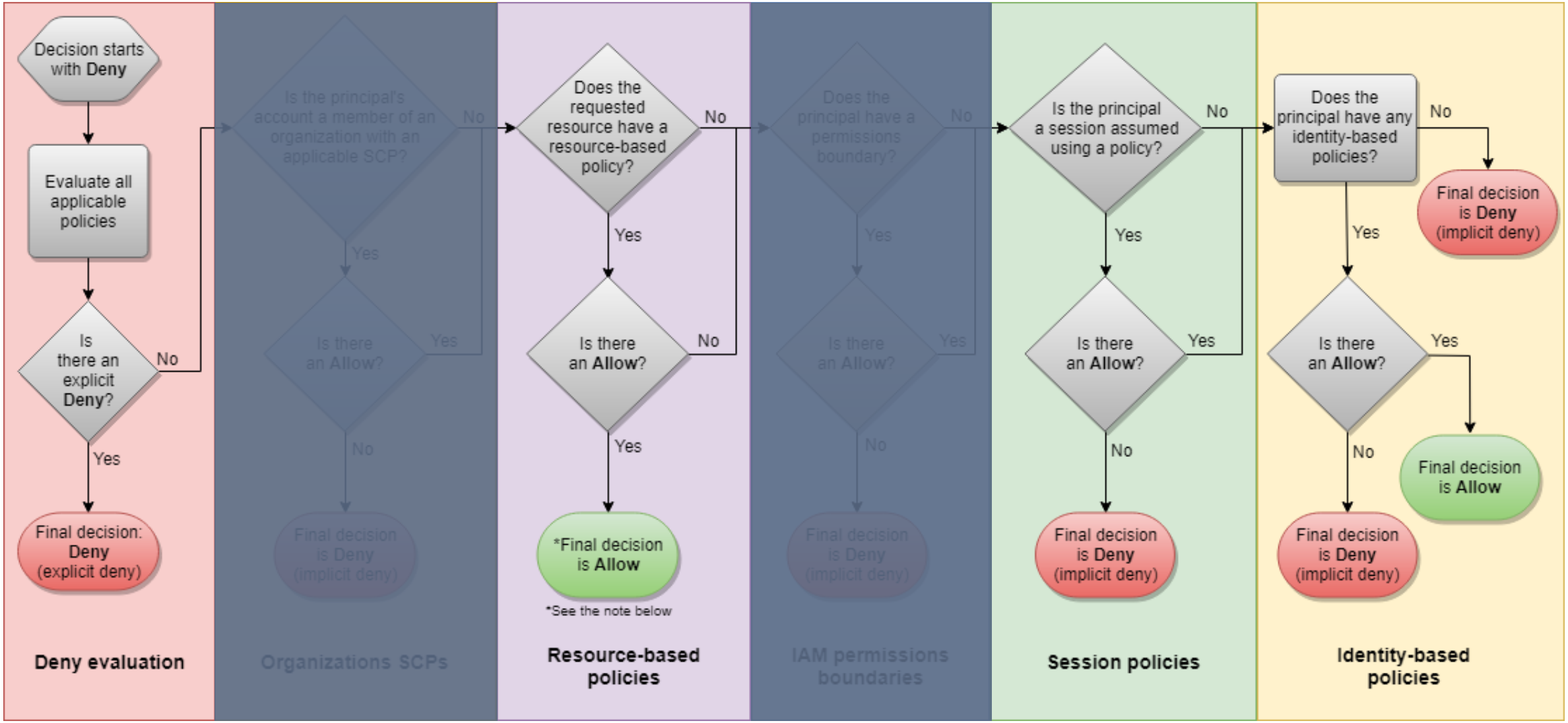
```
{
  "Version": "2012-10-17",
  "Statement": [ {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

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**.

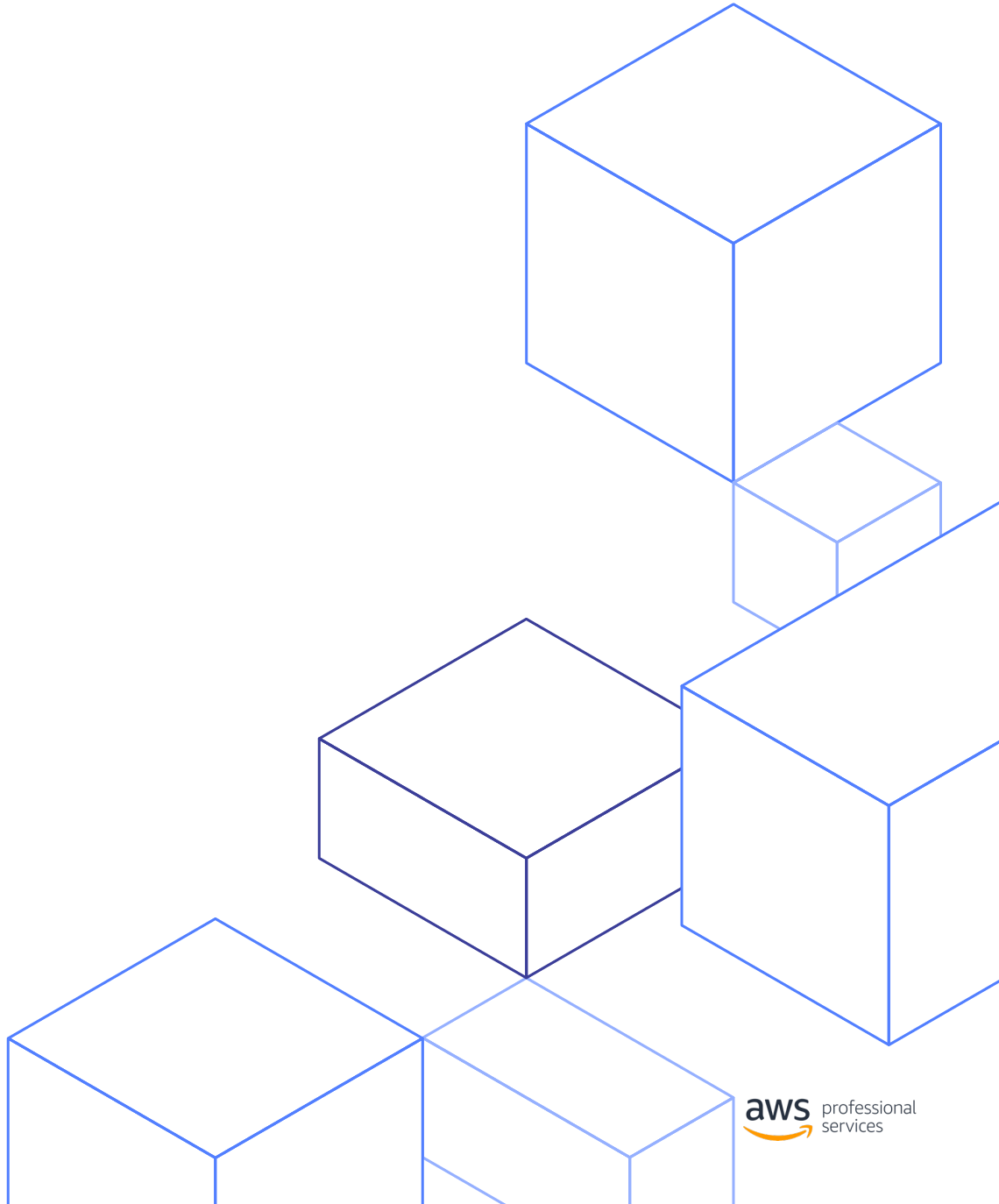
IAM Evaluation Logic



aws professional services

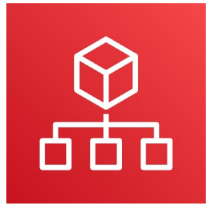
AWS Organizations

AWS Identity & Access Management
Authentication & Authorization



aws professional services

AWS Organizations

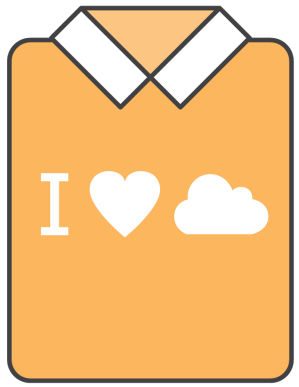


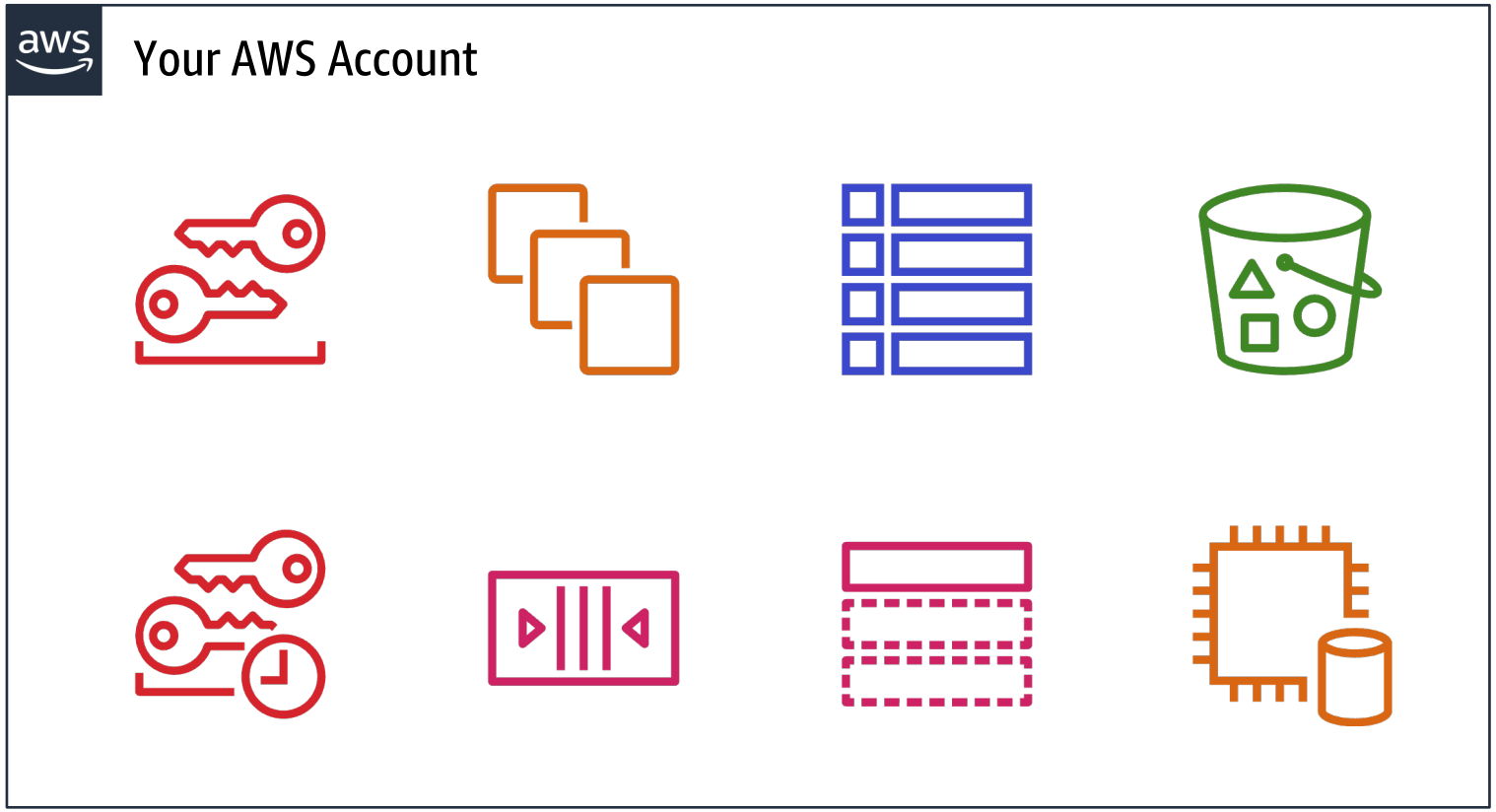
- Manage/control multiple AWS accounts centrally
- Enable multi-account functionality for AWS services

Key features:

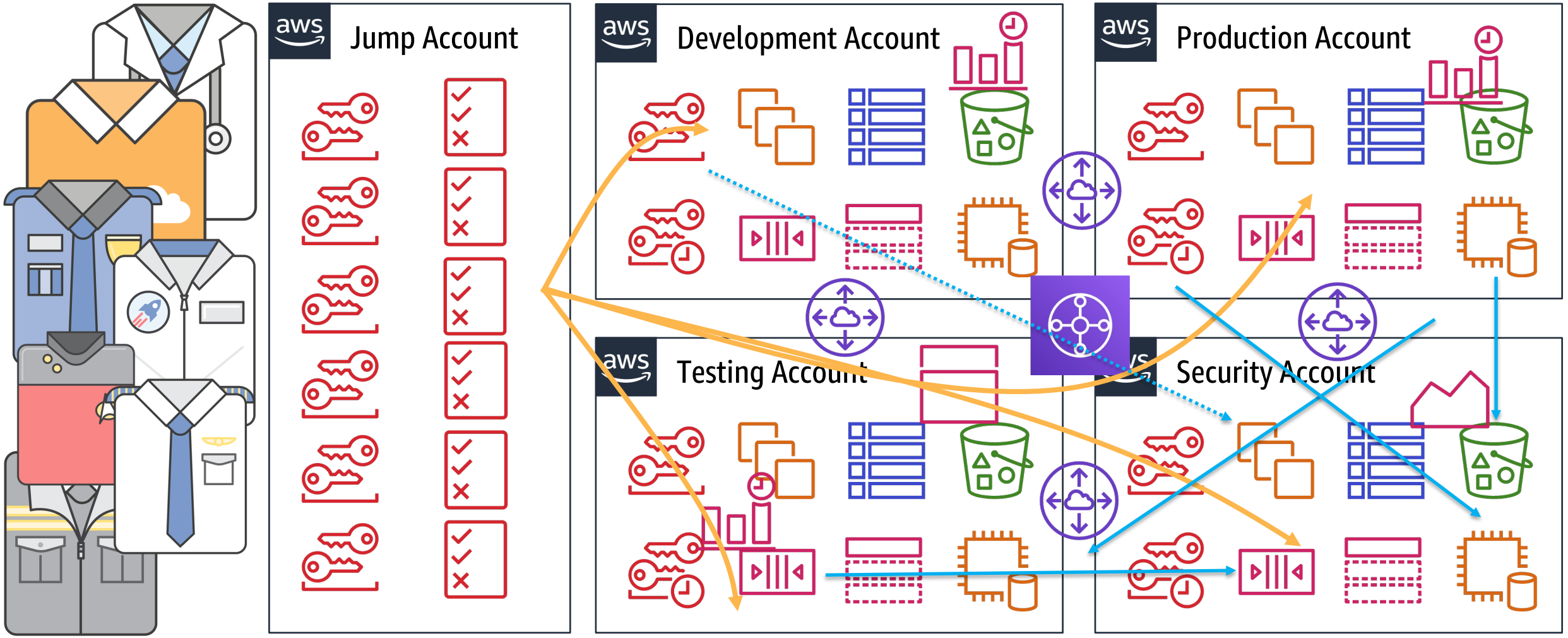
- Simplified creation of new AWS accounts
- Logically group AWS accounts for management convenience
- Apply organizational policies to control AWS services
- Consolidate billing and usage across all accounts into one bill

AWS Organizations – In the beginning...



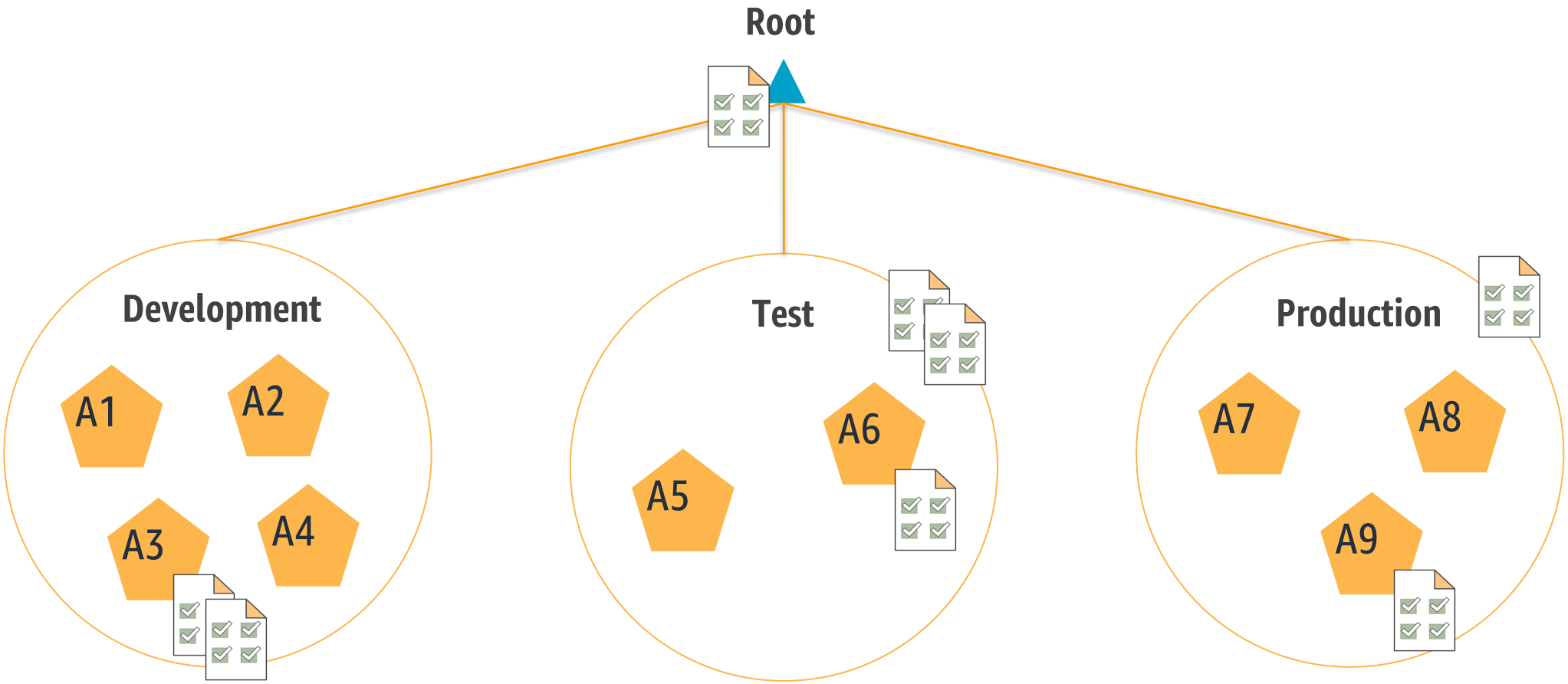


AWS Organizations – Today



aws professional services

AWS Organizations – Hierarchy and Policies



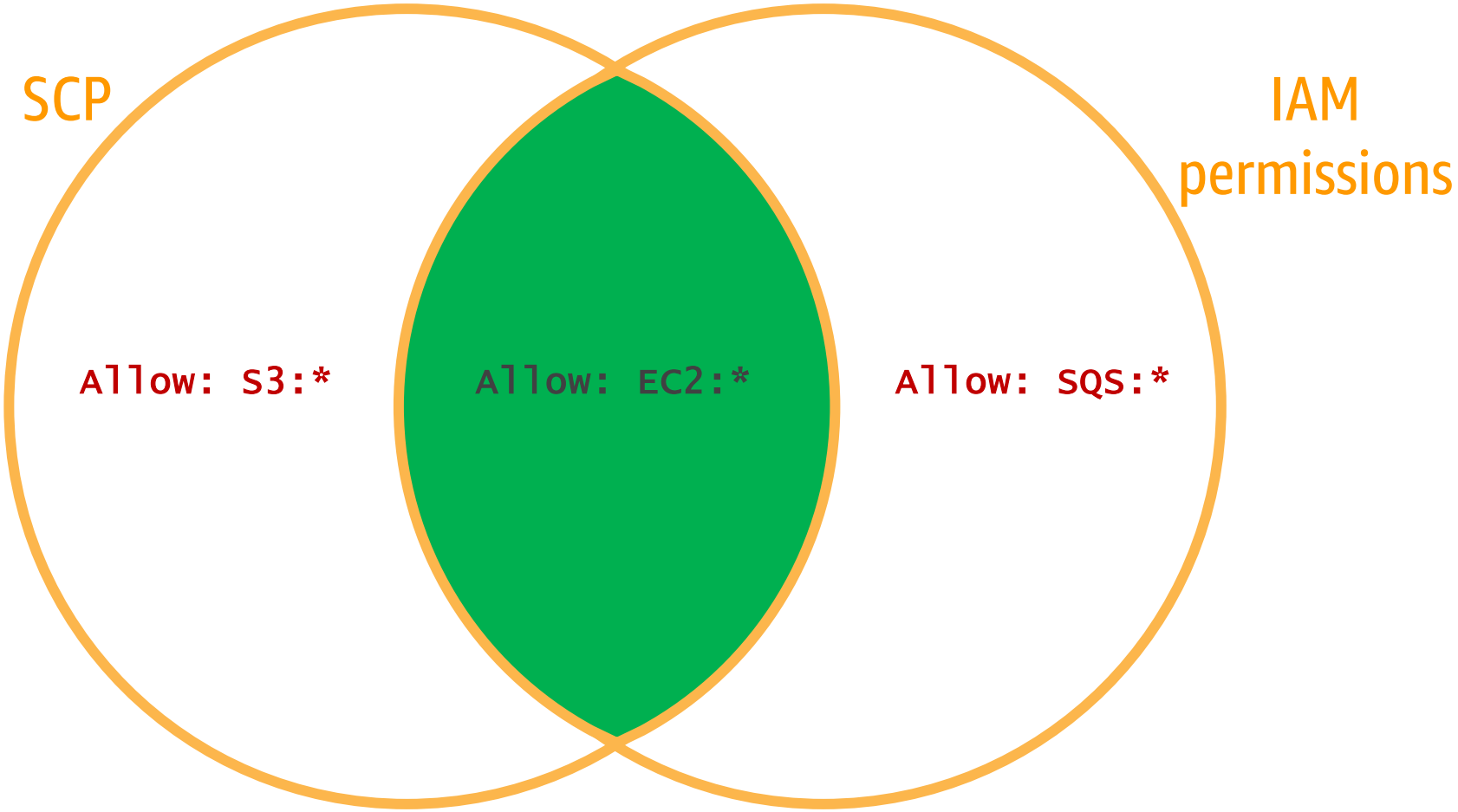
Service Control Policies use the IAM policy language

aws professional services

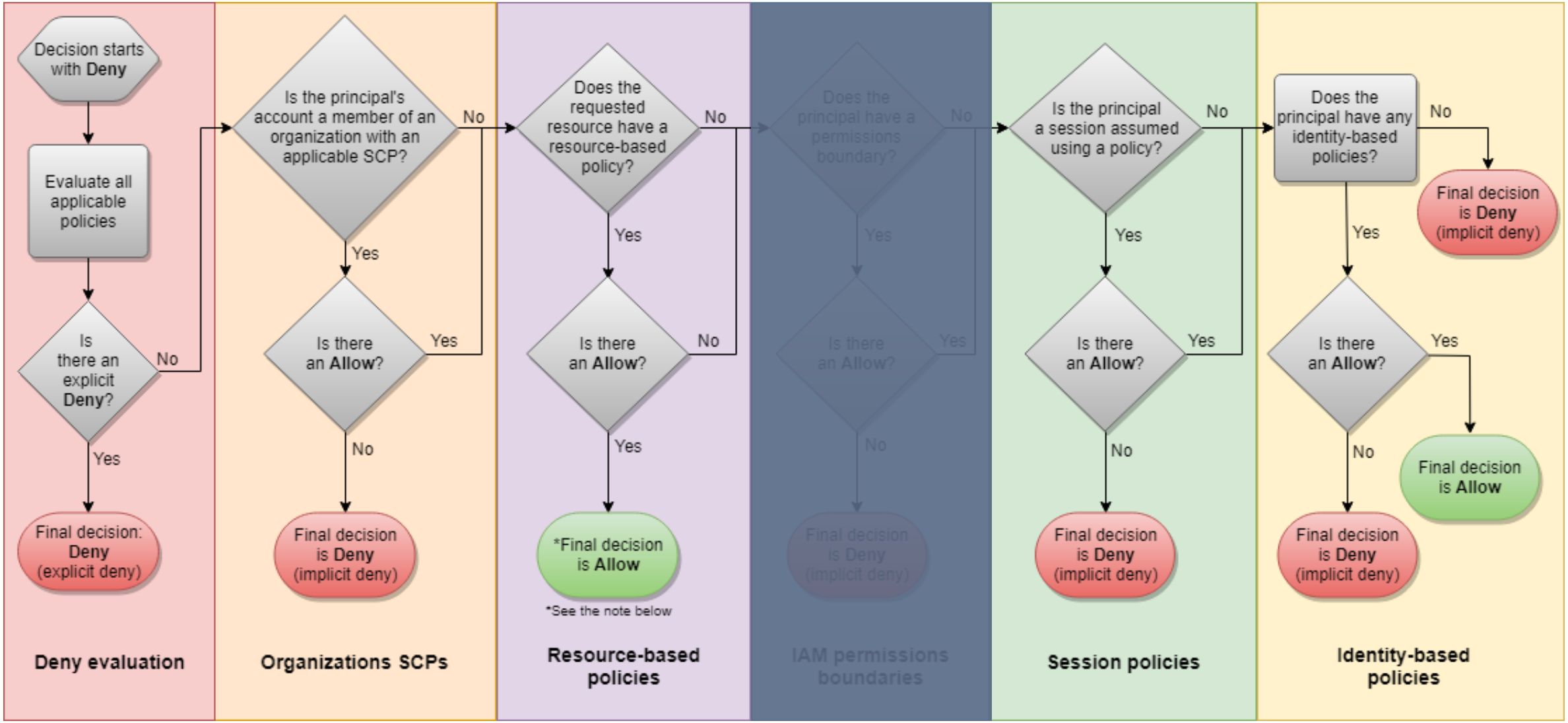
AWS Organizations – Hierarchy and Policies

aws professional services

AWS Organizations – Hierarchy and Policies



IAM Evaluation Logic



aws professional services

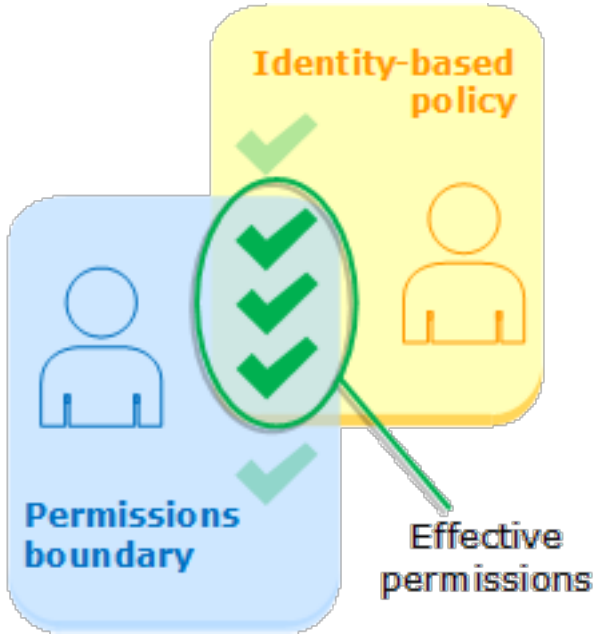
Permission Boundaries

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.

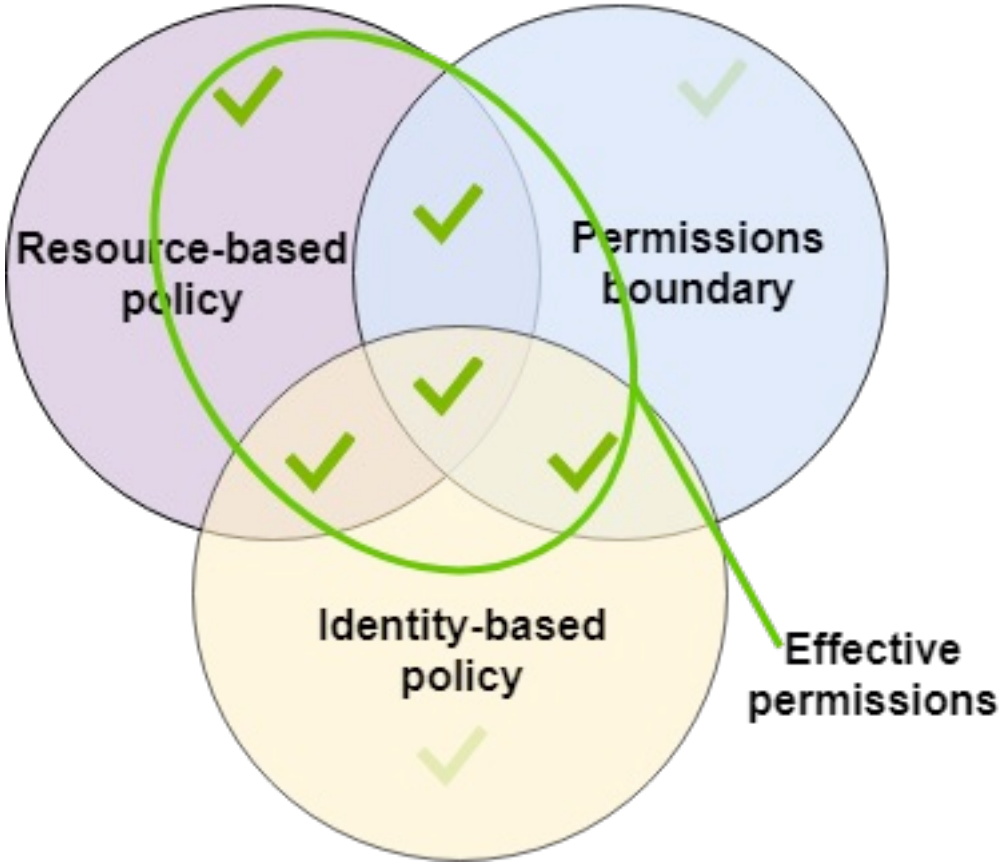
Permissions Boundaries – How do I enforce?

```
{
            "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

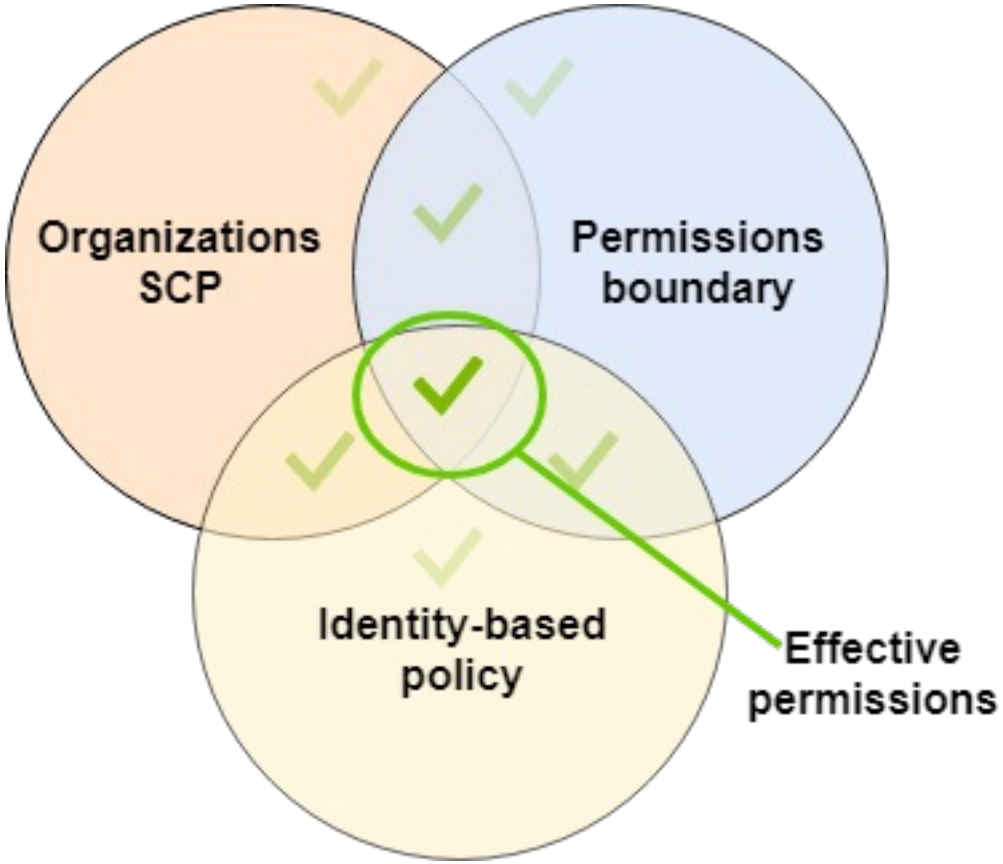
- Boundaries do not effect permissions granted through resource-based policies.
- Effective permissions consist of everything that is allowed by the resource-based
policy and everything that is allowed by both the permissions boundary and the identity-based policy.



Permissions Boundaries

Organizations SCP's

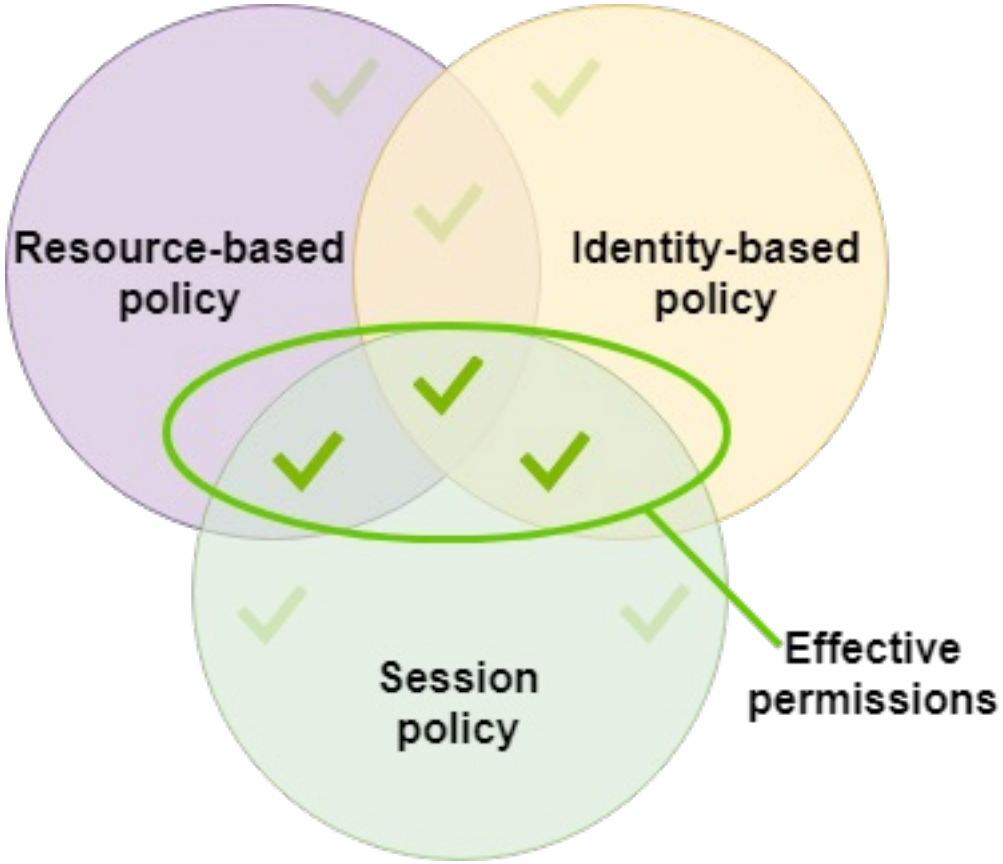
- SCP's do not grant any permissions to a principal.
- SCP's only limit the operations in an account and apply to all principals.
- Effective permissions consist of any operations that is allowed by any of the three policies.



Permissions Boundaries

Session Policy

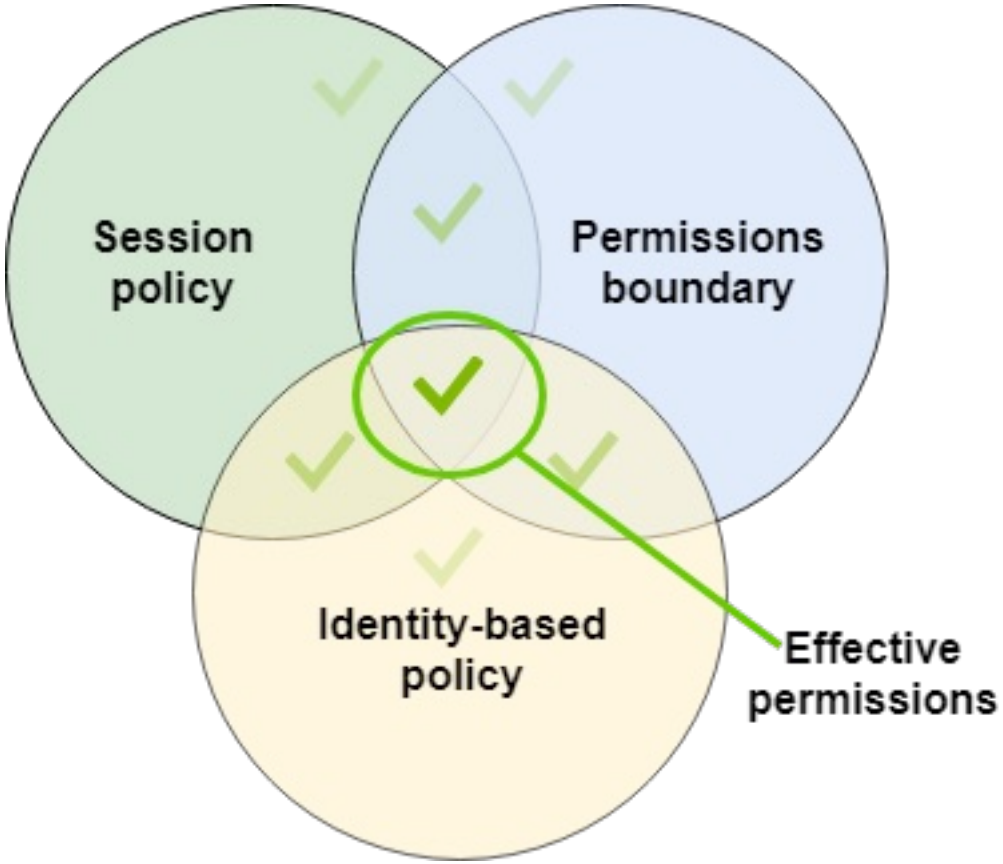
- Session policies are applied per session when a session is created.
- Effective permissions consist of any operation that is allowed by the session policy and either the identity-based policy of the principal that created the session, or an applicable resource-based policy.



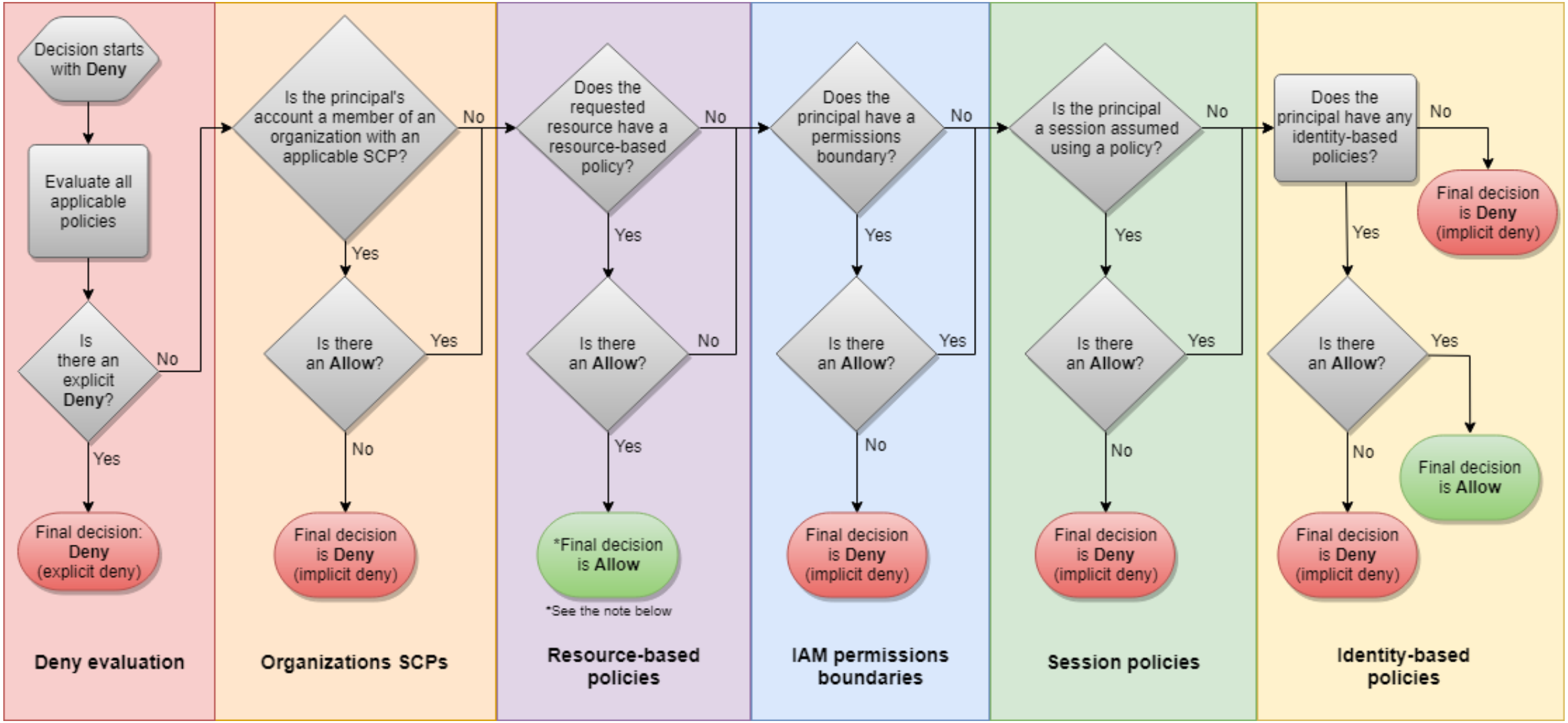
Permissions Boundaries

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

Thank you!

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