COMP5349– Cloud Computing Week 6: Cloud Security

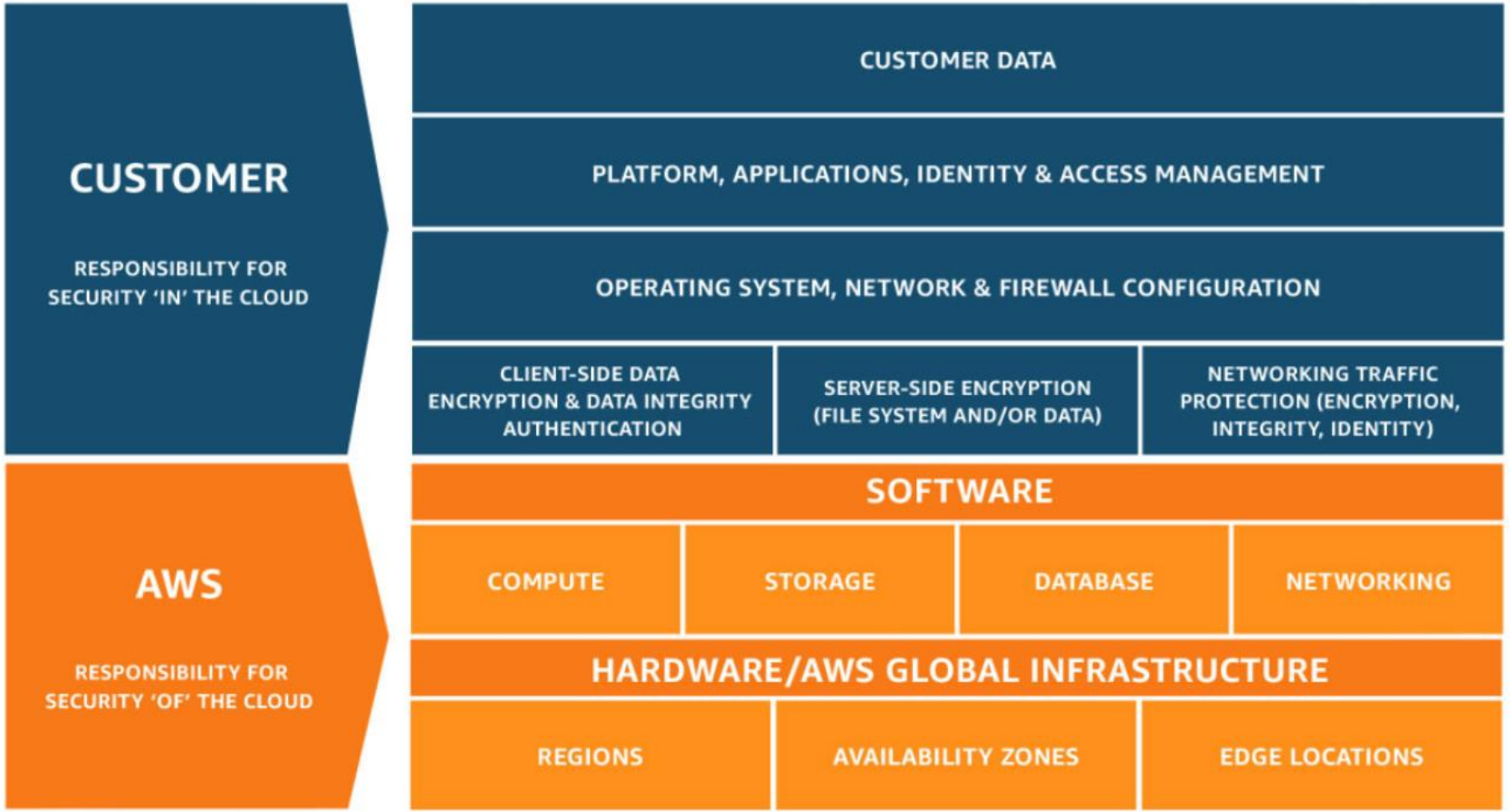
Dr. Ying Zhou

The University of Sydney

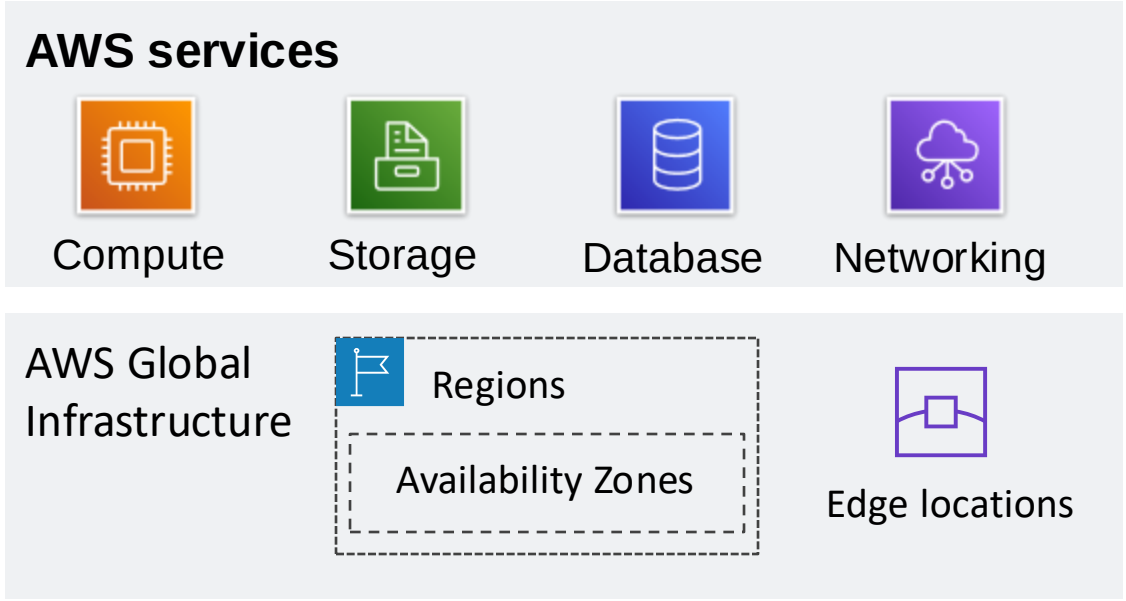
Table of Contents

AWS shared responsibility model

AWS shared responsibility model



AWS responsibility: Security *of* the cloud

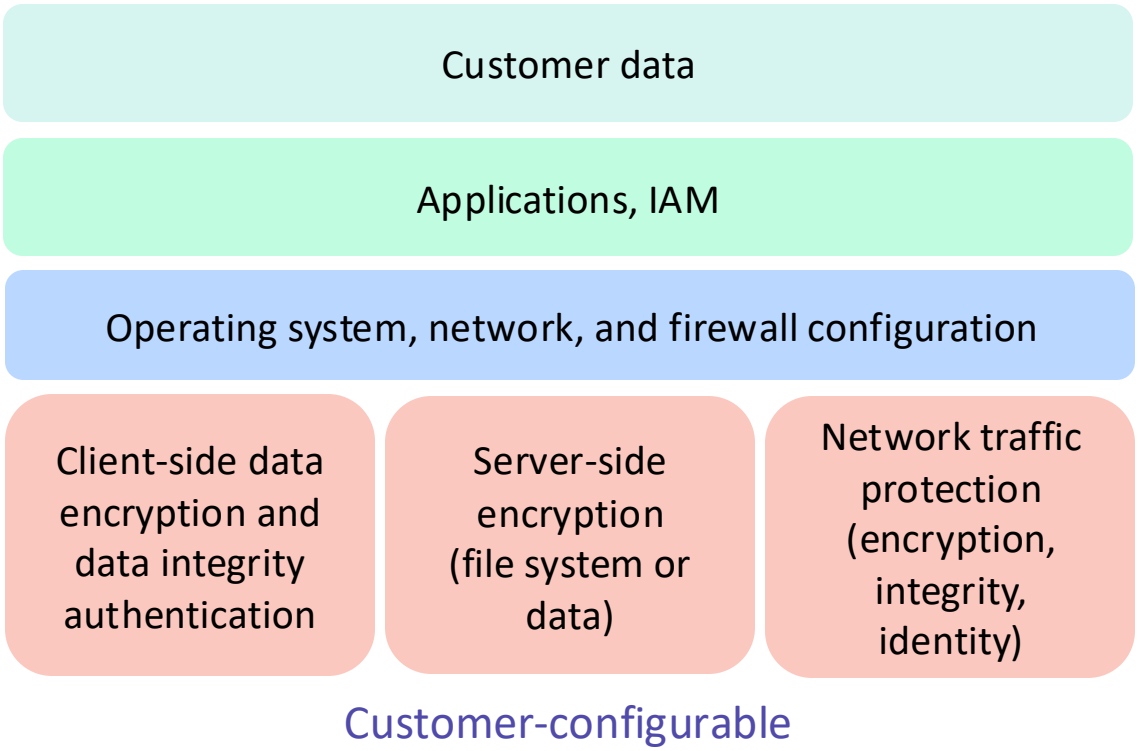


AWS responsibilities:

- Physical security of data centers
  - Controlled, need-based access
- Hardware and software infrastructure
  - Storage decommissioning, host operating system (OS) access logging, and auditing
- Network infrastructure
  - Intrusion detection

- Virtualization infrastructure
  - Instance isolation

Customer responsibility: Security *in* the cloud



Customer responsibilities:

- Amazon Elastic Compute Cloud (Amazon EC2) instance operating system
  - Including patching, maintenance
- Applications
  - Passwords, role-based access, etc.
- Security group configuration
- OS or host-based firewalls
  - Including intrusion detection or prevention systems
- Network configurations
- Account management
  - Login and permission settings for each user

Service characteristics and security responsibility (1 of 2)

Example services managed by the customer

Amazon EC2

Amazon Elastic Block Store (Amazon EBS)

Amazon Virtual Private Cloud (Amazon VPC)

Example services managed by AWS

AWS Lambda

Amazon Relational Database Service (Amazon RDS)

AWS Elastic Beanstalk

Infrastructure as a service (IaaS)

- Customer has more flexibility over configuring networking and storage settings
- Customer is responsible for managing more aspects of the security
- Customer configures the access controls

Platform as a service (PaaS)

- Customer does not need to manage the underlying infrastructure
- AWS handles the operating system, database patching, firewall configuration, and disaster recovery
- Customer can focus on managing code or data

Service characteristics and security responsibility (2 of 2)

SaaS examples

AWS Trusted Advisor

AWS Shield

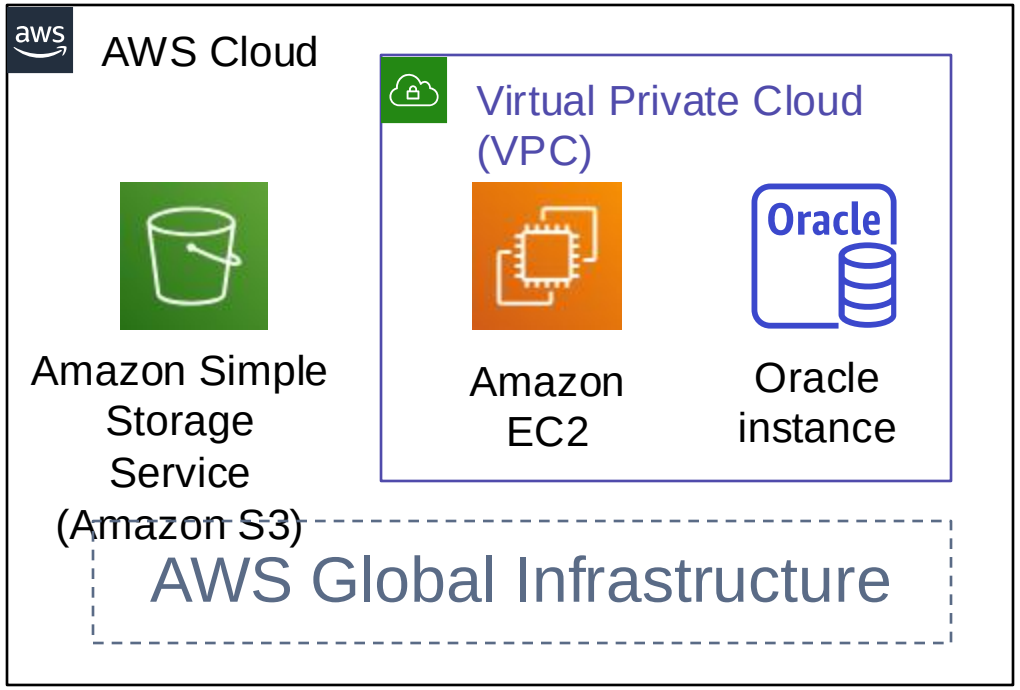
Amazon Chime

Software as a service (SaaS)

- Software is centrally hosted
- Licensed on a subscription model or pay-as-you-go basis.
- Services are typically accessed via web browser, mobile app, or application programming interface (API)
- Customers do not need to manage the infrastructure that supports the service

Activity: Scenario 1 of 2

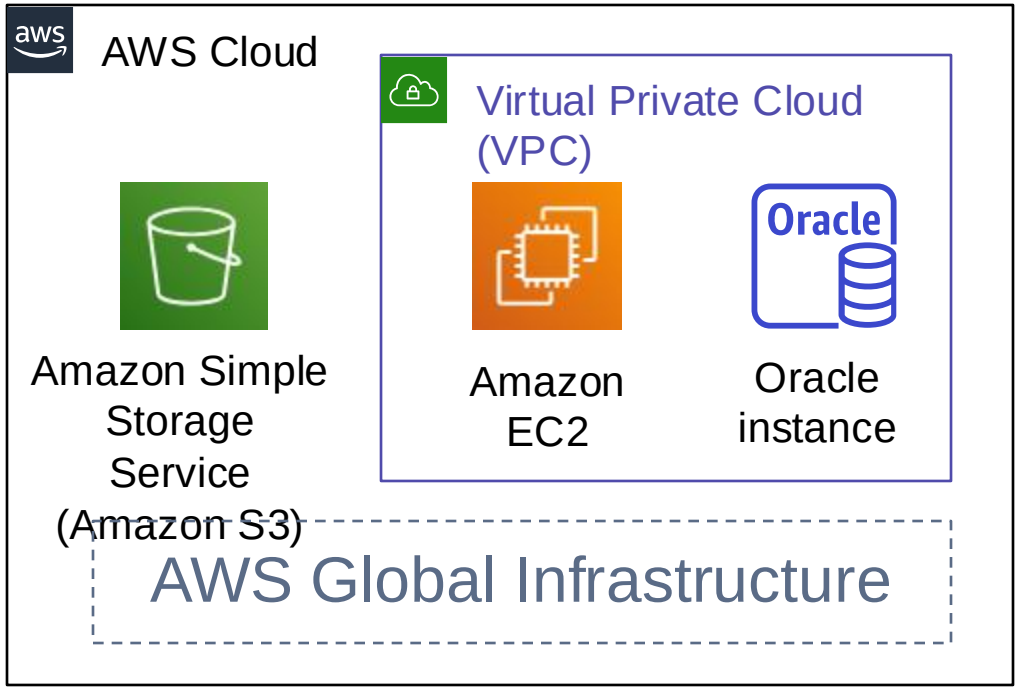
Consider this deployment. Who is responsible – AWS or the customer?



1. Upgrades and patches to the operating system on the EC2 instance?
2. Physical security of the data center?
3. Virtualization infrastructure?
4. EC2 security group settings?
5. Configuration of applications that run on the EC2 instance?
6. Oracle upgrades or patches If the Oracle instance runs as an Amazon RDS instance?
7. Oracle upgrades or patches If Oracle runs on an EC2 instance?
8. S3 bucket access configuration?

Activity: Scenario 1 of 2 Answers

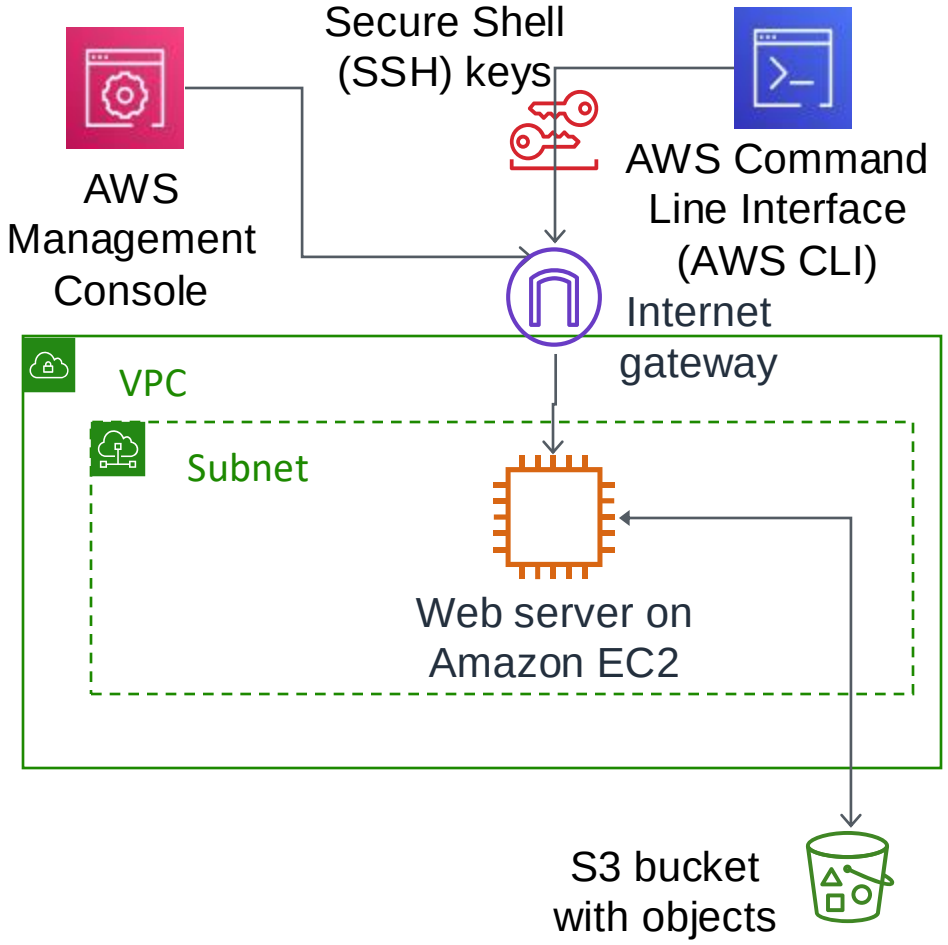
Consider this deployment. Who is responsible – AWS or the customer?



1. Upgrades and patches to the operating system on the EC2 instance?
   - **ANSWER:** The customer
2. Physical security of the data center?
   - **ANSWER:** AWS
3. Virtualization infrastructure?
   - **ANSWER:** AWS
4. EC2 security group settings?
   - **ANSWER:** The customer
5. Configuration of applications that run on the EC2 instance?
   - **ANSWER:** The customer
6. Oracle upgrades or patches If the Oracle instance runs as an Amazon RDS instance?
   - **ANSWER:** AWS
7. Oracle upgrades or patches If Oracle runs on an EC2 instance?
   - **ANSWER:** The customer
8. S3 bucket access configuration?
   - **ANSWER:** The customer

Activity: Scenario 2 of 2

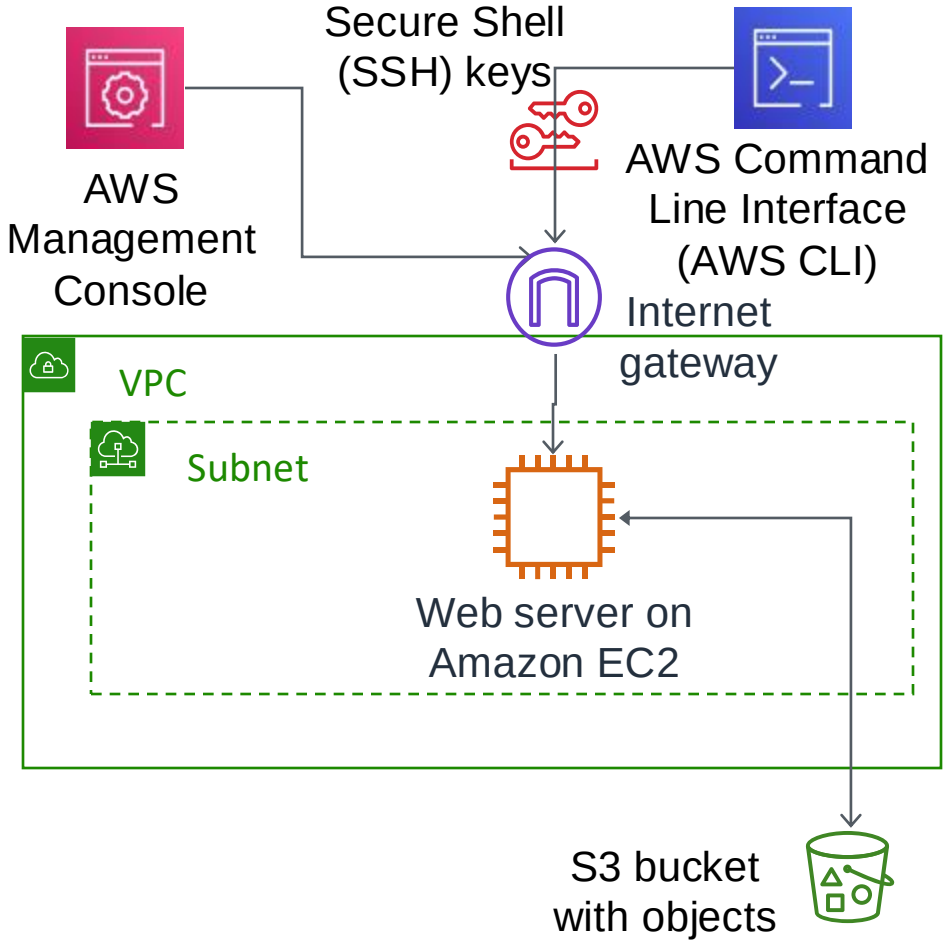
Consider this deployment. Who is responsible – AWS or the customer?



1. Ensuring that the AWS Management Console is not hacked?
2. Configuring the subnet?
3. Configuring the VPC?
4. Protecting against network outages in AWS Regions?
5. Securing the SSH keys
6. Ensuring network isolation between AWS customers' data?
7. Ensuring low-latency network connection between the web server and the S3 bucket?
8. Enforcing multi-factor authentication for all user logins?

Activity: Scenario 2 of 2 Answers

Consider this deployment. Who is responsible – AWS or the customer?



1. Ensuring that the AWS Management Console is not hacked?
   - **ANSWER:** AWS
2. Configuring the subnet?
   - **ANSWER:** The customer
3. Configuring the VPC?
   - **ANSWER:** The customer
4. Protecting against network outages in AWS Regions?
   - **ANSWER:** AWS
5. Securing the SSH keys
   - **ANSWER:** The customer
6. Ensuring network isolation between AWS customers' data?
   - **ANSWER:** AWS
7. Ensuring low-latency network connection between the web server and the S3 bucket?
   - **ANSWER:** AWS
8. Enforcing multi-factor authentication for all user logins?
   - **ANSWER:** The customer

Identity and Access Management

Identity and Access

Identity

- Each entity (such as a user, administrator, or system) needs an *identity*. The process of **verifying that identity is called *authentication***

Access

- *Access* management is about ensuring that entities can perform only the tasks they need to perform. The process **of checking what access an entity should have is called *authorization***

Typical identity management in Linux OS

- Users
  - Root user – with a user id 0
    - Created when the OS is installed
  - System/service users - with a user id 1-999
    - Created when certain packages, e.g. When a database server is installed
  - Regular/normal/local users – with a user id > 1000
    - Usually one regular user will be created during installation;
    - The others can be created by the root user or a user with administrative privileges
- Groups
  - Most Linux OS has a **sudo** group to represent users with admin privilege
  - Other groups can be created

Typical security practice

- The root user is disabled by default in many Linux systems including AWS EC2 instances
  - An initial user is added to the **sudo** group
  - Sudo allows an authorized user to temporarily elevate their privileges using their own password instead of having to know the password belonging to the root account.
  - e.g. "ec2-user" or "ubuntu"

Typical identity management in application

- E.g. In MySQL
- Account Users
- Privileges
  - Administrative, database, table, etc,
- Roles
  - Reprents collections of privileges
- Privileges can be granted to users
- Roles can be set for users.

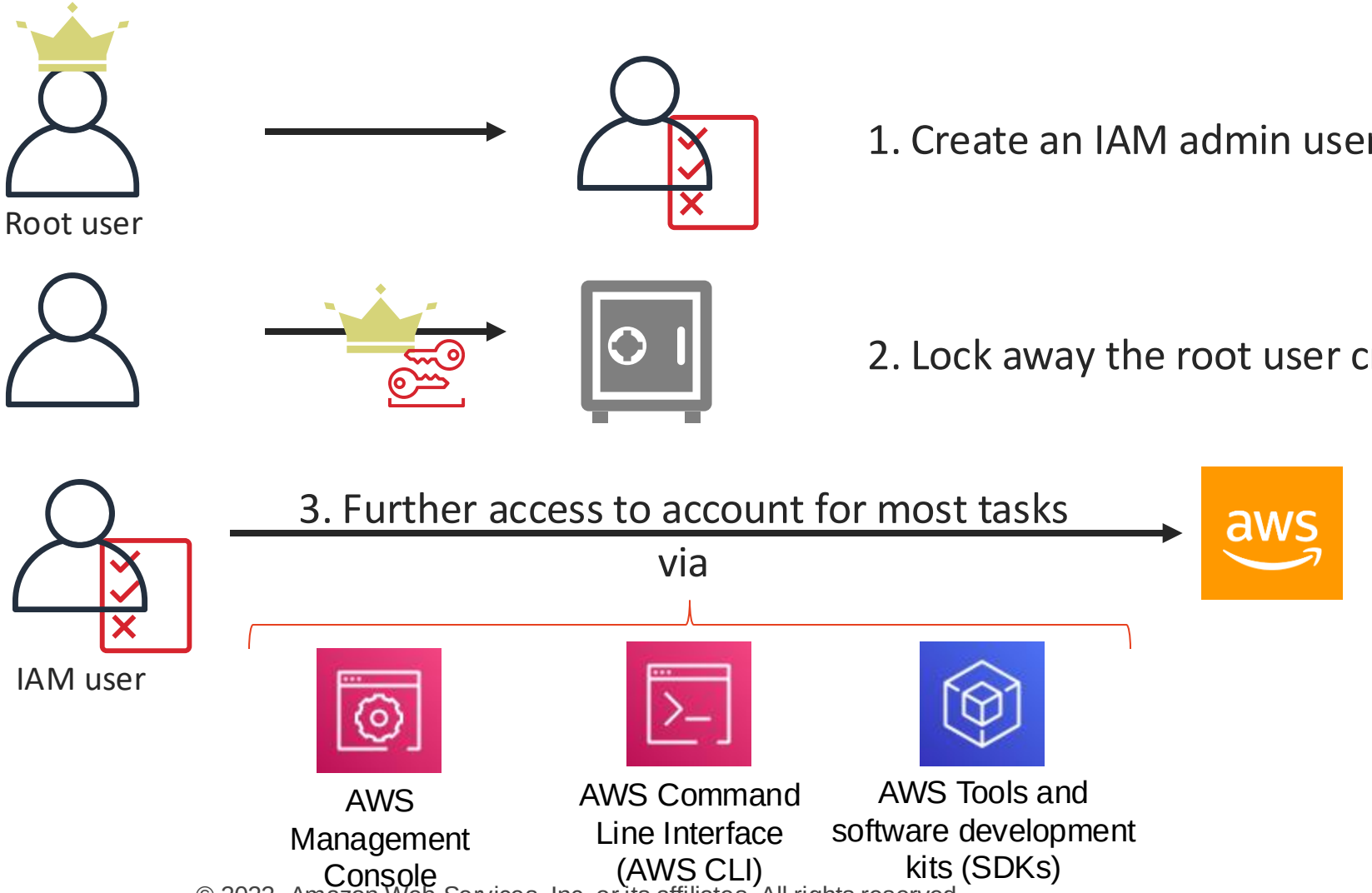
Terminologies and Concepts in AWS

- Users
  - *Account users* and IAM users
- Groups
  - IAM groups, representing a collection of IAM users
- Roles
  - Can be assumed by user or services
- Policies
  - Define the actions permitted

AWS Identity and Access Management (IAM)

Secure the root account

The account root user has a large amount of power. Recommended security steps:



IAM components: Review

IAM user

Defined in your AWS account. Use credentials to authenticate programmatically or via the AWS Management Console.

IAM group

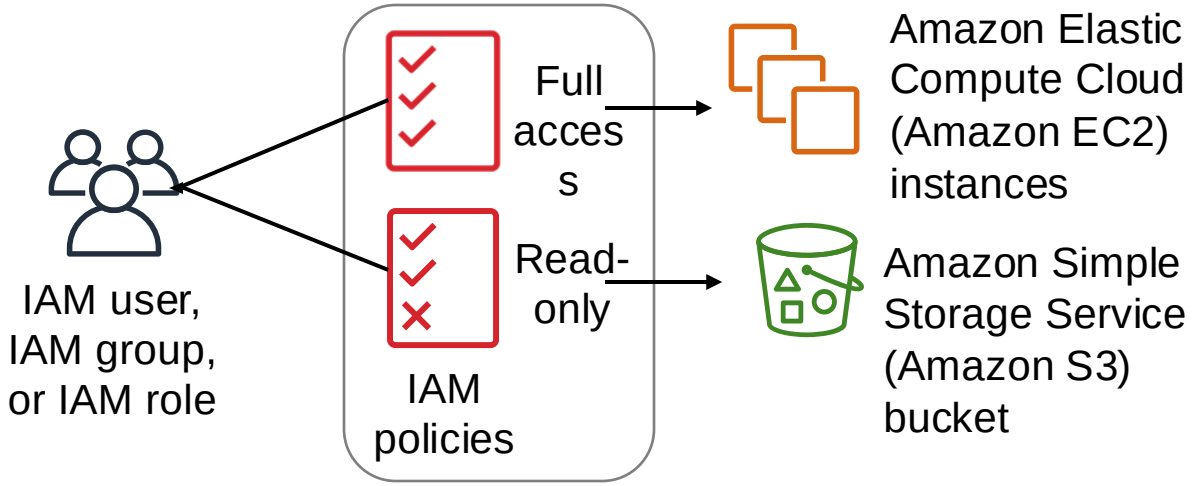
A collection of IAM users that are granted identical authorization.

IAM policy

Defines which resources can be accessed and the level of access to each resource.

IAM role

Mechanism to grant temporary access for making AWS service requests. *Assumable* by a person, application, or service.



IAM user authentication: Review

When you define an **IAM user**, you select what ***types of access*** the user is permitted to use.

Programmatic access

- Authenticate using:
  - Access key ID
  - Secret access key
- Provides AWS CLI and AWS SDK access

AWS CLI

AWS Tools and SDKs

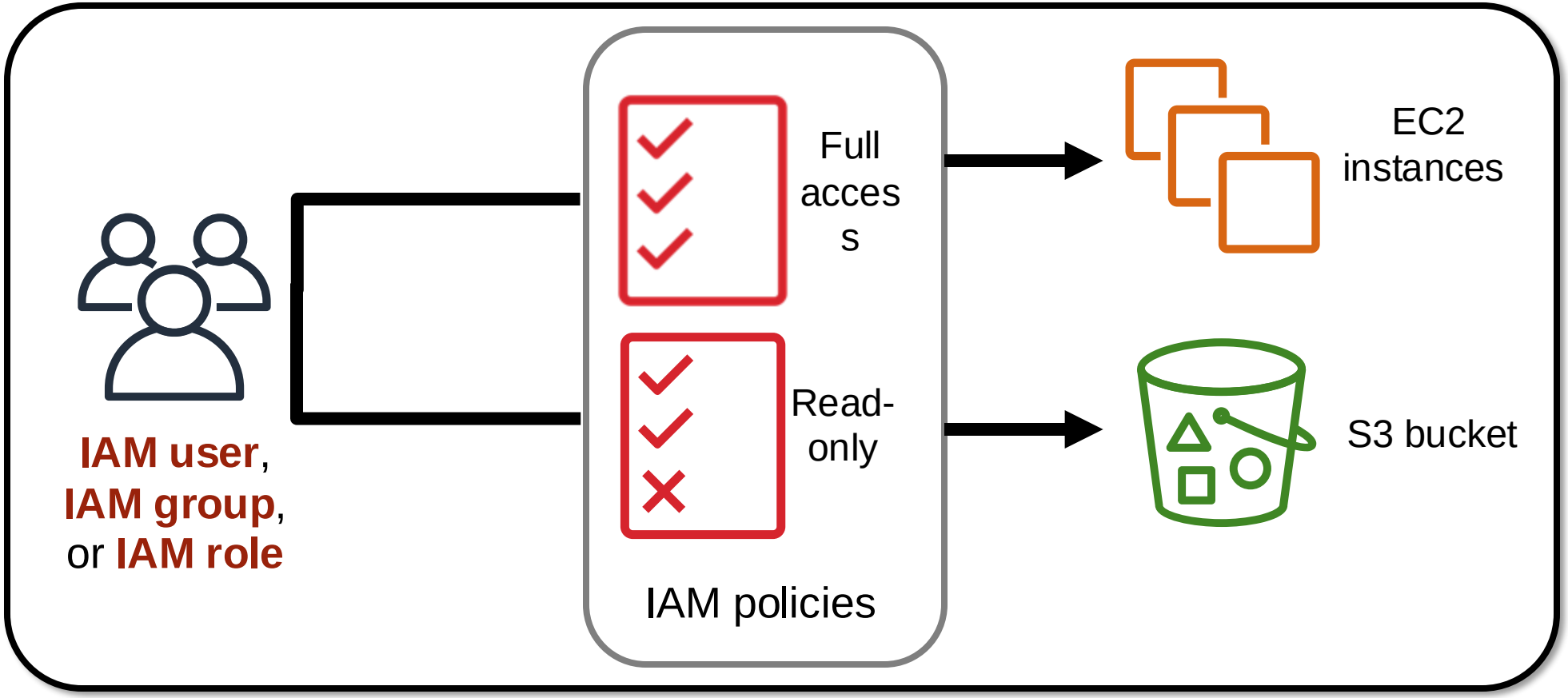
AWS Management Console access

- Authenticate using:
  - 12-digit Account ID *or* alias
  - IAM user name
  - IAM password
- If enabled, **multi-factor authentication (MFA)** prompts for an authentication code.

AWS Management Console

IAM Authorization: Review

After the user or application is connected to the AWS account, what are they allowed to do?



Groups and Roles in non-cloud setting

- The "group" concept is mainly used in OS identity management
- The "role" concept is mainly used in application identity management
- They refer to similar mechanism
  - Representing a collection of permissions that can be granted to different users
  - A user can be assigned to multiple groups or roles

Group and roles in AWS

- They all represent a collection of permissions
- IAM group is similar to the group concept in OS and the role concept in application
- IAM roles are used to handle ***cloud specific*** authorization scenarios
  - Allow AWS resources to use AWS services
    - E.g. Allow EC2 instance to access S3 or Parameter Store in system manager
    - OS uses service or system accounts for similar purpose
  - Provide access to externally authenticated users
  - Provide access to third parties
  - Provide cross-account access

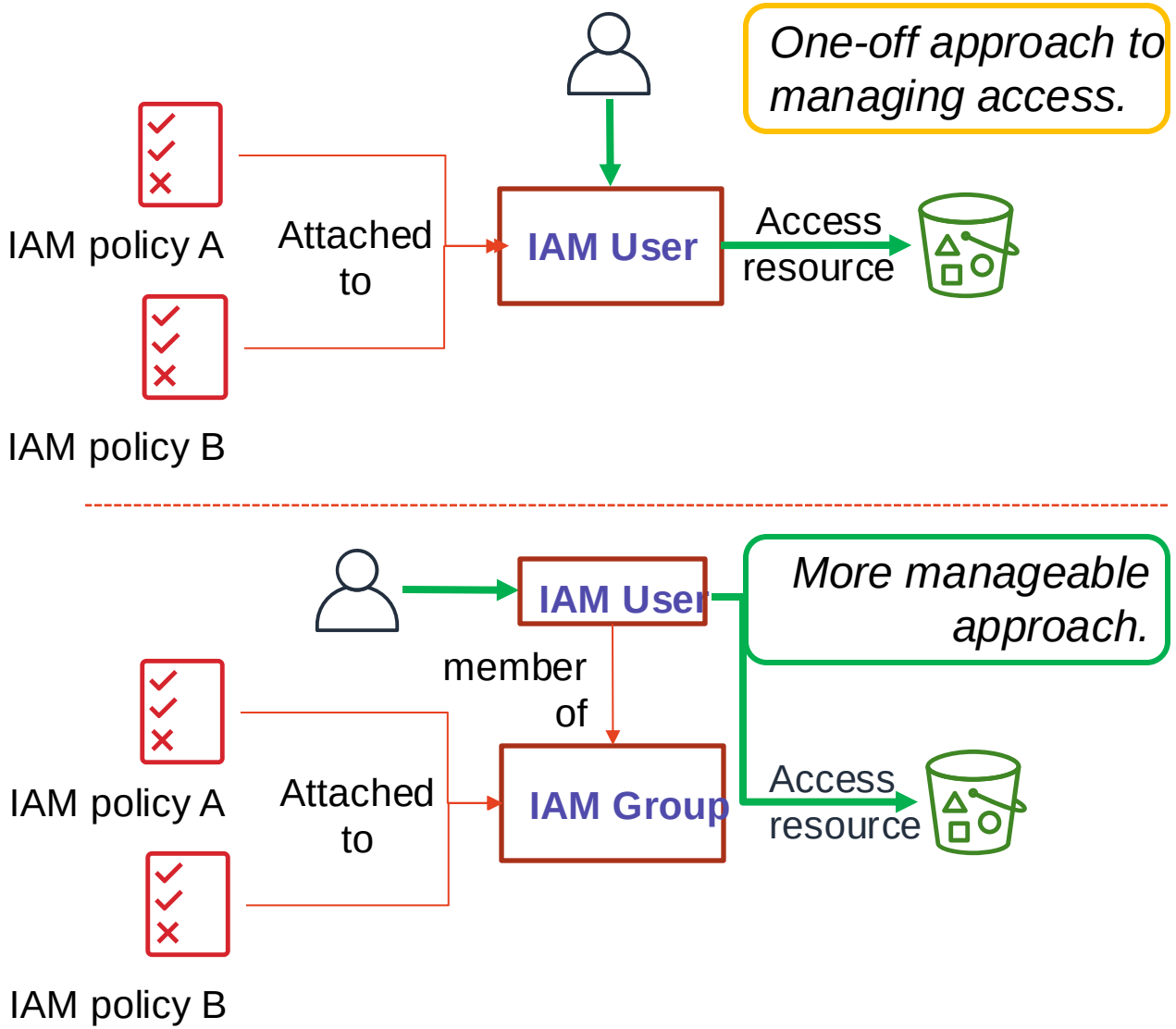
IAM groups

Use IAM groups to grant the same access rights to multiple users.

- All users in the group inherit the permissions assigned to the group
  - Makes it easier to manage access across multiple users

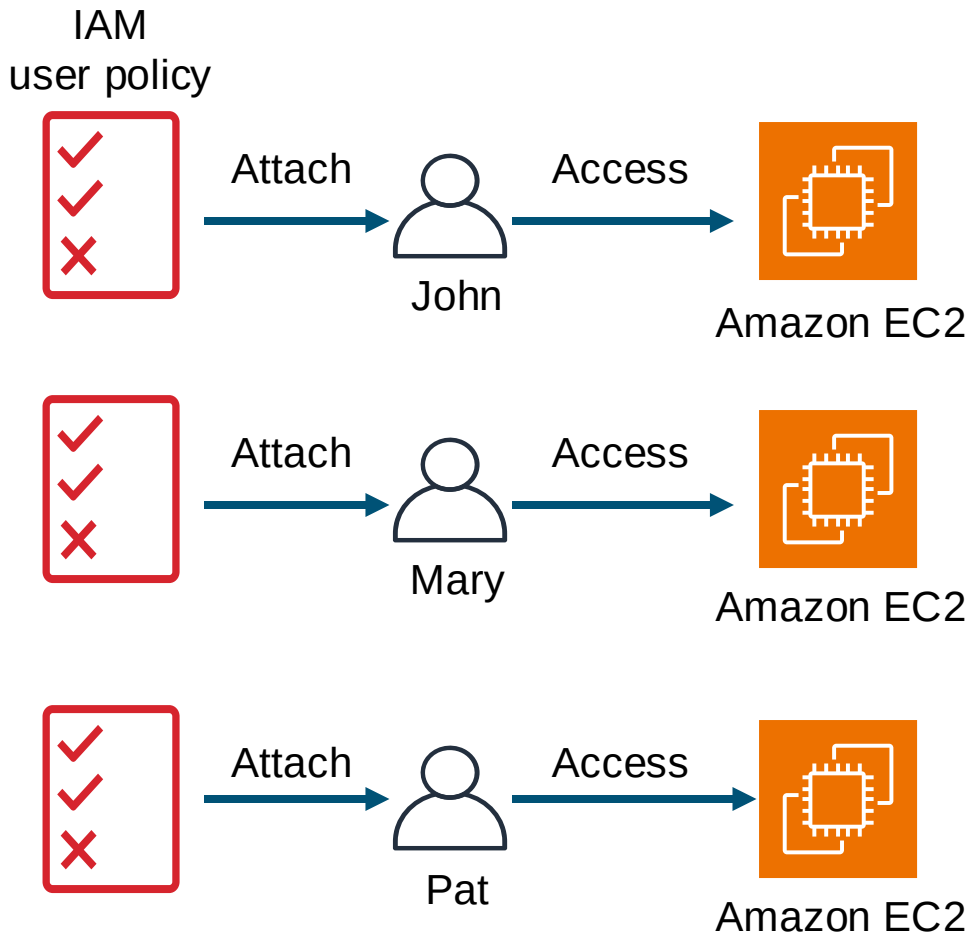
Tip: Combine approaches for fine-grained individual access

- Add the user to a group to apply standard access based on job function
- Optionally attach an additional policy to the user for needed exceptions



Challenge of managing permissions

Assigning permissions directly to users is difficult to manage



Initial configuration

Each developer is given full access to Amazon Elastic Compute Cloud (Amazon EC2) through policies that are attached to individual users.

Additional access is needed

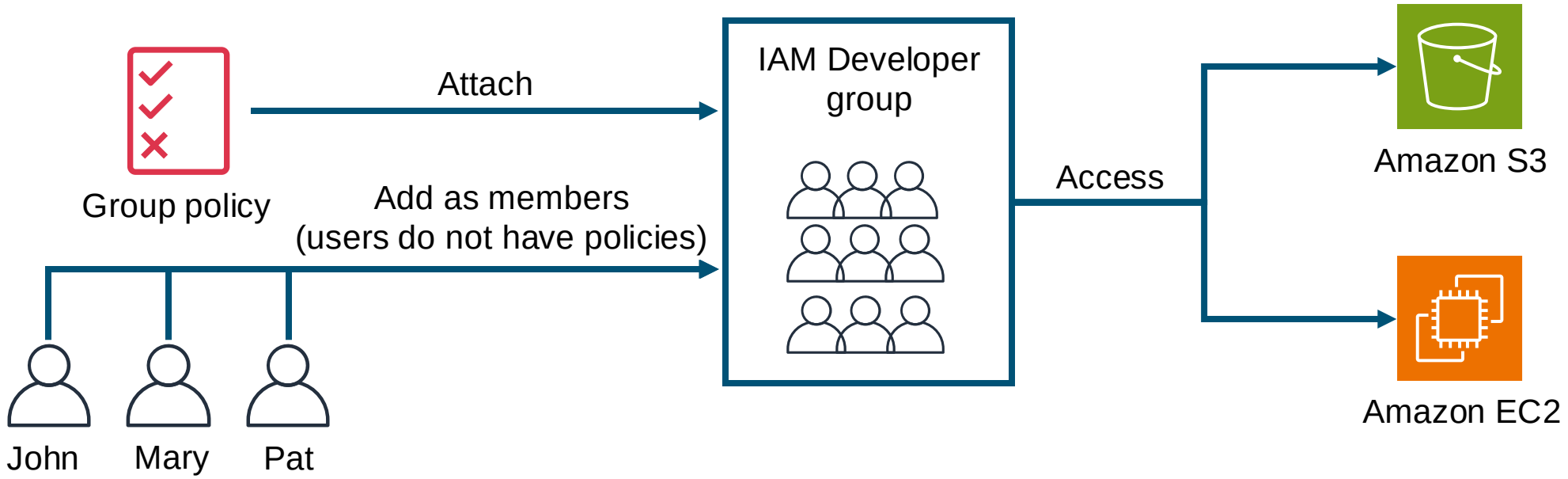
Each developer needs access to Amazon Simple Storage Service (Amazon S3). To implement this change, an administrator needs to make three modifications, one for each AWS Identity and Access Management (IAM) user's policy.

The number of developers grows

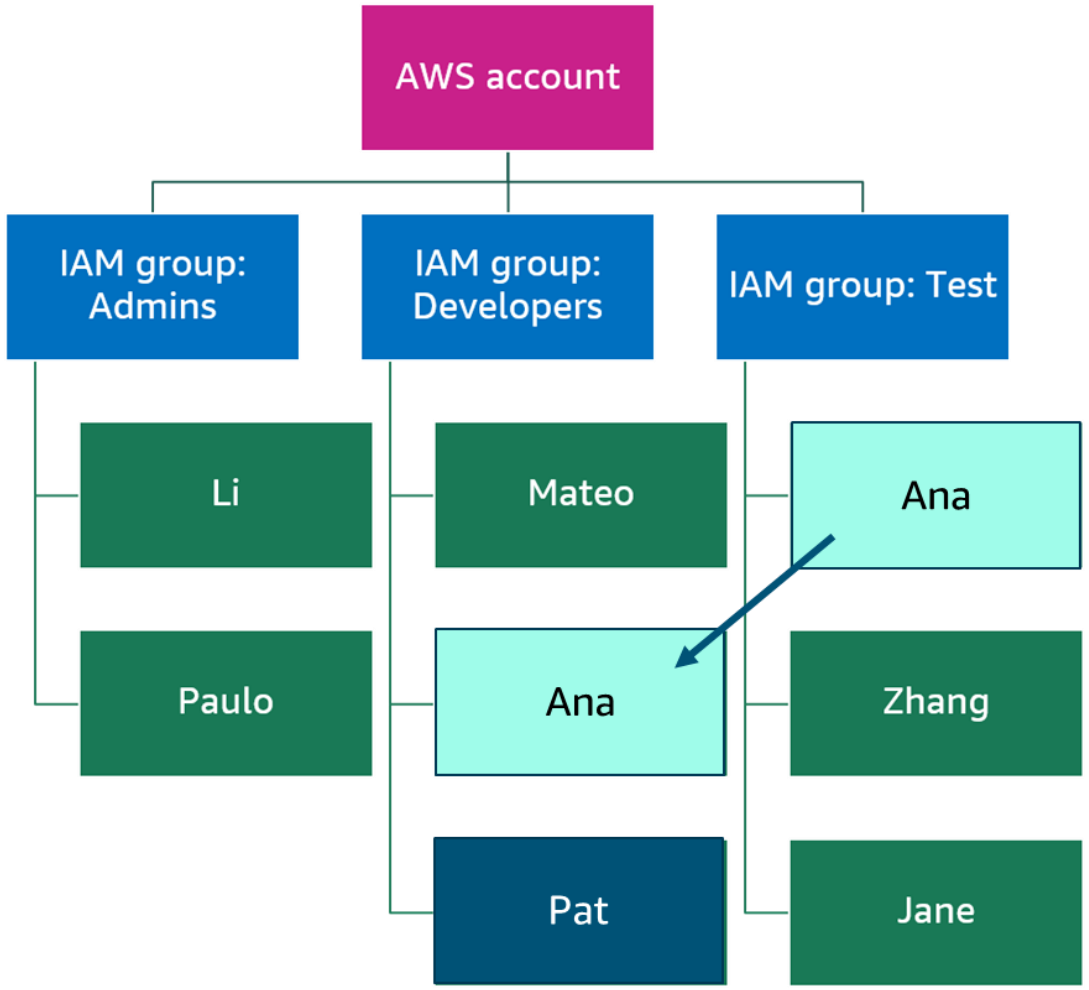
This approach becomes hard to manage. Is there an easier way to manage multiple permissions and users?

Use IAM groups to attach permissions to multiple users

- Permissions in a group can be based on a job function.
- All users in the group inherit the permissions assigned to the group.



Example: Using IAM groups to reflect job role



If a new developer (Pat) is hired, add them to the developers group. Pat will immediately inherit the same access that's granted to other developers.

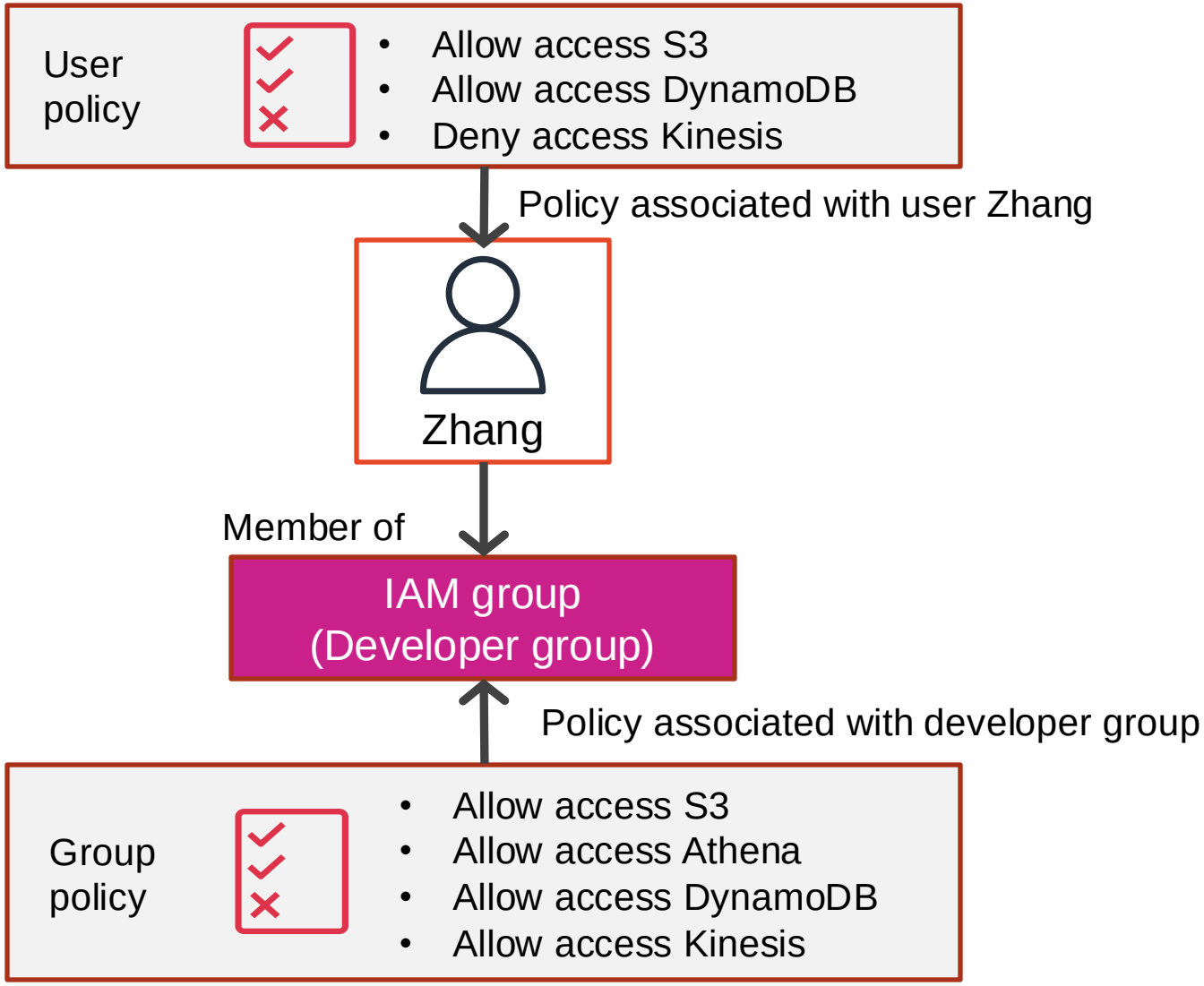
If Ana takes on the new role of developer, do the following:

- Remove her from the test group.
- Add her to the developers group.

Users can belong to more than one group, but groups cannot be nested.

Permissions in policies directly attached to a user (user policies) override permissions in group policies if they are more restrictive.

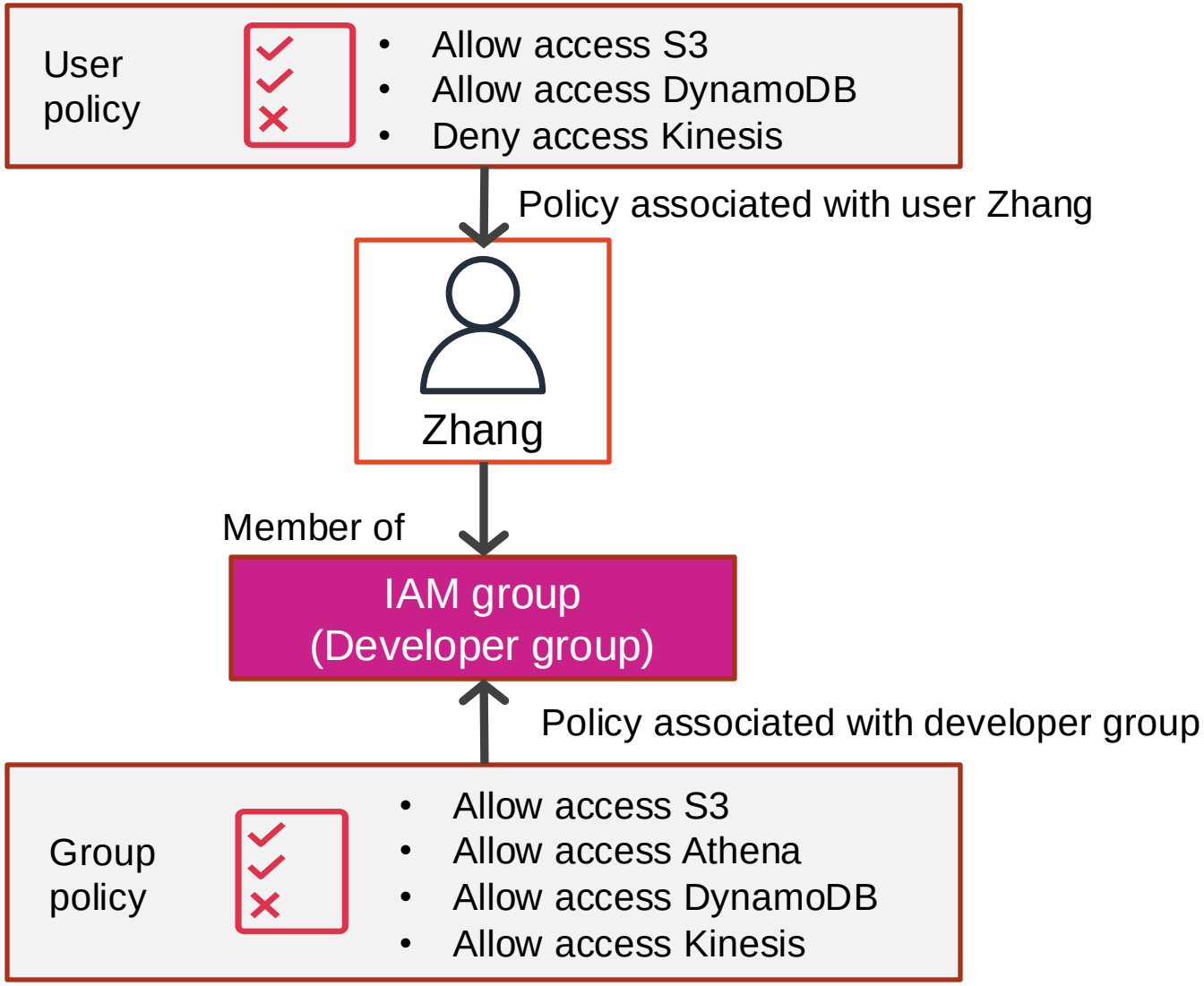
Example: Using a user policy and group policy together



Can Zhang access Amazon Athena?

Can Zhang access Amazon Kinesis?

Example: Resulting permissions



Zhang can access Athena

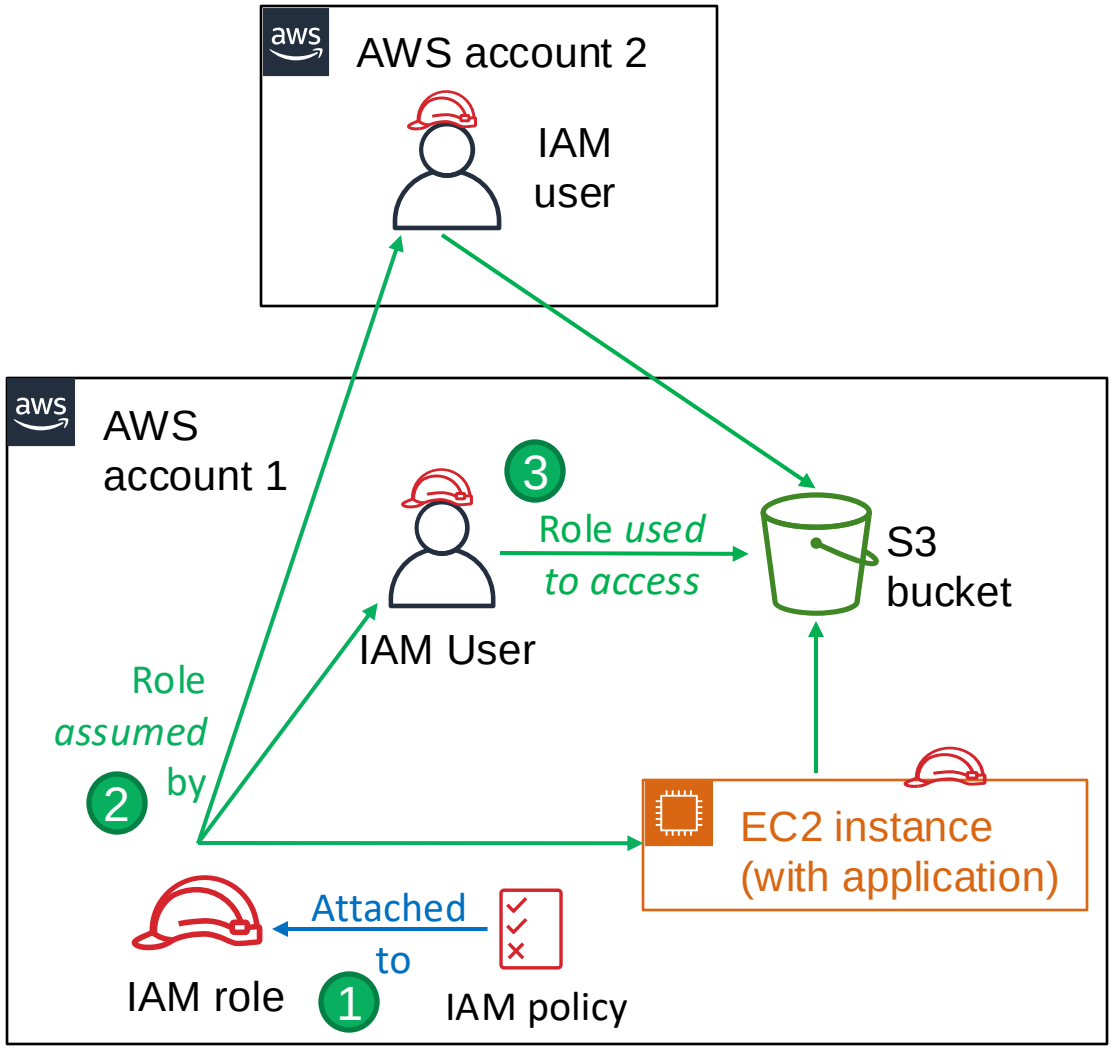
Zhang is a member of a developer group. The developer group allows access to Athena.

Zhang cannot access Kinesis

Zhang is a member of a developer group that allows Kinesis access. But the user policy explicitly denies access to Kinesis.

IAM roles

- IAM role characteristics
  - Provides *temporary* security credentials
  - Is not uniquely associated with one person
  - Is *assumable* by a person, application, or service
  - Is often used to delegate access
- Use cases
  - Provide AWS resources with access to AWS services
  - Provide access to externally authenticated users
  - Provide access to third parties
  - Switch roles to access resources in –
    - Your AWS account
    - Any other AWS account (cross-account access)



IAM Policies

IAM policies and permissions

Use policies to fine-tune the permissions that are granted to principals.

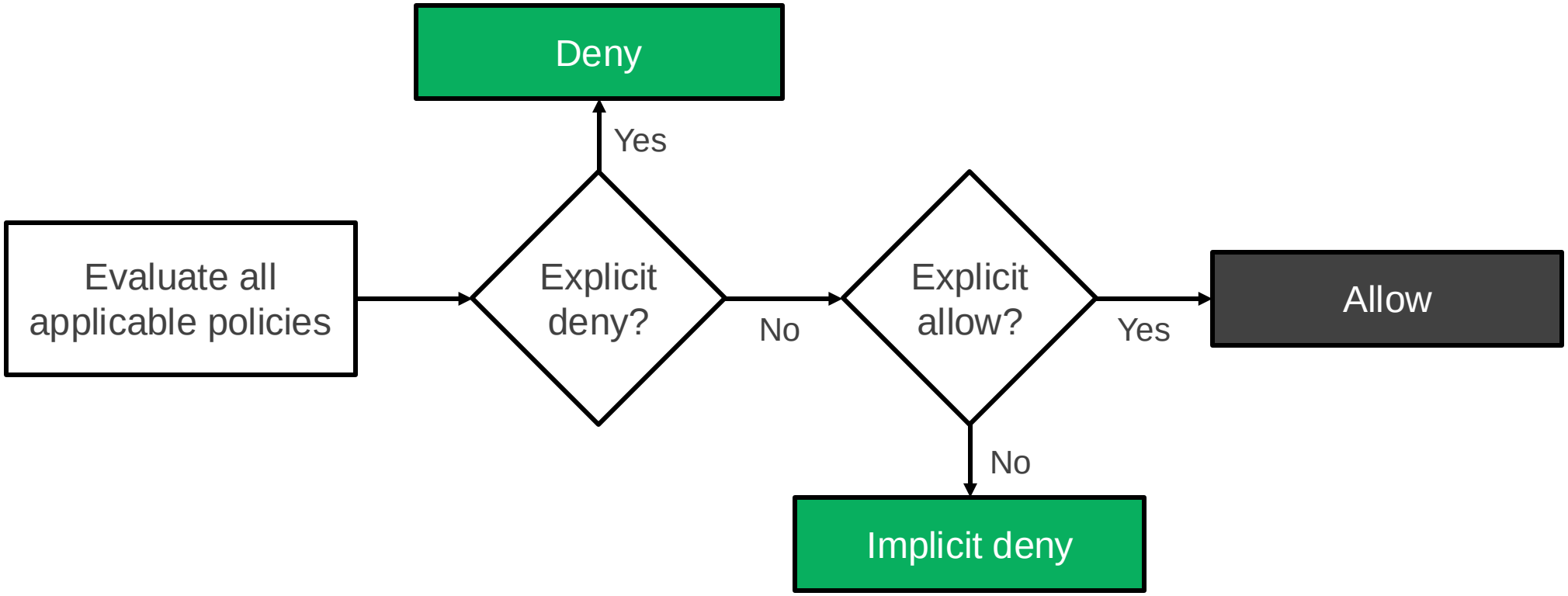
Two types of policies:

- **Identity-based:** Attach to an IAM user, group, or role.
- **Resource-based:** Attach to an AWS resource.

Define permissions in IAM policy documents.

- The document is formatted in JSON.
- The policy defines which resources and operations are allowed or denied.
- Follow the principle of least privilege.

Determining permissions at the time of request



Identity-based and resource-based policies

Identity-based
(Attached to an *IAM user, group, or role*)

What does a particular identity have access to?

Carlos

Resource	Read	Write	List
Resource X	Allow	Allow	Allow

Richard

Resource	Read	Write	List
Resource Y	Allow	N/A	N/A
Resource Z	Allow	N/A	N/A

Managers

Resource	Read	Write	List
Resource X	N/A	N/A	Allow
Resource Y	N/A	N/A	Allow
Resource Z	N/A	N/A	Allow

Resource-based
(Attached to an *AWS resource*)

Who has access to a particular resource?

Resource X

User	Read	Write	List
Ana	Allow	Allow	Allow
Akua	Allow	Allow	Allow

Resource Y

User	Read	Write	List
Paulo	Allow	Allow	Allow
Nikki	Allow	N/A	N/A
Mateo	N/A	Allow	Allow

Policies: Example 1

Identity-based
(Attached to an *user, group, or role*)

Bob

Resource	Get	Put	List
Bucket X	Allow	Allow	Allow
Bucket Y	N/A	N/A	Allow

Resource-based
(Attached to an *AWS resource*)

Bucket X

User	Get	Put	List
Bob	Allow	Deny	Allow

Bucket Y

User	Get	Put	List
Bob	Allow	N/A	Allow

Can Bob GET, PUT, or LIST for bucket X?

The identity-based policy allows him to use the GET, PUT, and LIST APIs for bucket X. However, the resource-based policy for bucket X allows him to use GET and LIST, but denies the ability to use PUT. This means that Bob cannot PUT objects into bucket X, even though his identity-based policy allows it.

Policies: Example 2

Identity-based
(Attached to an *user, group, or role*)

Bob

Resource	Get	Put	List
Bucket X	Allow	Allow	Allow
Bucket Y	N/A	N/A	Allow

Resource-based
(Attached to an *AWS resource*)

Bucket X

User	Get	Put	List
Bob	Allow	Deny	Allow

Bucket Y

User	Get	Put	List
Bob	Allow	N/A	Allow

Can Bob GET or LIST for bucket Y?

The identity-based policy allows the LIST action but doesn't explicitly allow or deny the GET and PUT actions on bucket Y. The resource-based policy for bucket Y allows Bob to use GET and LIST, but doesn't specify the PUT action. Therefore, Bob can read objects from the bucket, even though his identity-based policy doesn't explicitly allow it.

IAM policy document structure

Element	Information
Version	Version of the policy language that you want to use
Statement	Defines what is allowed or denied based on conditions
Effect	Allow or deny
Principal	For a resource-based policy, the account, user, role, or federated user to allow or deny access to. For an identity-based policy, the principal is implied as the user or role that the policy is attached to.
Action	Action that is allowed or denied Example: "Action": "s3:GetObject"
Resource	Resource or resources that the action applies to Example: "Resource": "arn:aws:sqs:us-west-2:123456789012:queue1" (ARN = AWS resource name)
Condition	Conditions that must be met for the rule to apply

IAM policy document structure example

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Effect": "effect",
      "Action": "action",
      "Resource": "arn",
      "Condition":{
         "condition":{
            "key": "value"
         }
      }
   }]
}
```

- Effect: Effect can be either *Allow* or *Deny*
- Action: Type of access that is allowed or denied

```
"Action": "s3:GetObject"
```

- Resource: Resources that the action will act on

```
"Resource": "arn:aws:sqs:us-west-2:123456789012:queue1"
```

- **Condition**: Conditions that must be met for the rule to apply

```
"Condition" : {
   "StringEquals" : {
         "aws:username" : "johndoe"
   }
}
```

Example: resource-based policy

```
{
  "Version":"2012-10-17",
  "Statement":[{
     "Effect":"Allow",
     "Action":["dynamoDB:*","s3:*"
     ],
     "Resource":[
      "arn:aws:dynamodb:region:account-number-without-hyphens:table/course-notes",
      "arn:aws:s3:::course-notes-web",
      "arn:aws:s3:::course-notes-mp3/*"]
   },
   {
     "Effect":"Deny",
     "Action":["dynamodb:*","s3:*"
     ],
     "NotResource":[
      "arn:aws:dynamodb:region:account-number-without-hyphens:table/course-notes",
      "arn:aws:s3:::course-notes-web",
      "arn:aws:s3:::course-notes-mp3/*"]
   }
  ]
}
```

Explicitly allow any (*) DynamoDB or S3 action on the DynamoDB table course-notes, the S3 bucket course-notes-web and any object in the S3 bucket course-notes-mp3.

Deny any (*) DynamoDB or S3 action on tables or S3 buckets except for those listed under NotResource.

Example: Identity-based policy

```
{
 "Version":"2012-10-17",
 "Statement":[{
   "Effect":"Allow",
   "Action":[
     "iam:*LoginProfile",
        "iam:*AccessKey*",
        "iam:*SSHPublicKey*"
    ],
    "Resource":[
     "arn:aws:iam::account-id-without-hyphens:user/${aws:username}"
 }
}
```

The Action element lists all the actions that are allowed by the Effect: Allow.

The Resource element lists the AWS resources that the allowed actions can be performed on.

Example: Cross-account, resource-based policy

Policy created by account A

```
{
    "Version": "2012-10-17",
    "Statement": {
      "Sid": "AccountBAccess1",
        "Principal": {"AWS": "111122223333"},
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET,
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*
        ]
    }
}
```

Account number of Account B

Allow account B to take any S3 action on the DOC-EXAMPLE-BUCKET.

ARNs and wildcards

- Resources are identified by using Amazon Resource Name (ARN) format
  - Syntax – `arn:`*`partition`*`:`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
  - Example – `"Resource": "arn:aws:iam::123456789012:user/mmajor"`

- You can use a wildcard (*) to give access to all actions for a specific AWS service
  - Examples –
    - `"Action": "s3:*"`
    - `"Action": "iam:*AccessKey*"`

Activity: IAM policy analysis (1 of 3)

Consider this IAM policy, and then answer the questions as they are presented.

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

1. Which AWS service does this policy grant access to?

   Answer: The IAM service.

2. Does the policy allow the user to create an IAM user, group, policy, or role?

   Answer: No. Access is limited to *get* and *list* requests. The policy effectively grants read-only permissions.

3. What are at least three specific actions that the iam:Get* action allows?

   Answer: iam:Get* allows actions such as GetGroup, GetPolicy, and GetRole.

Activity: IAM policy analysis (2 of 3)

Consider this IAM policy, and then answer the questions as they are presented.

```
{
  "Version":"2012-10-17",
  "Statement":[{
      "Effect":"Allow",
"Action":"ec2:TerminateInstances",
      "Resource":"*"
    },
    {
      "Effect":"Deny",
    "Action":"ec2:TerminateInstances",
      "Condition":{
        "NotIpAddress":{
          "aws:SourceIp":[
            "192.0.2.0/24",
            "203.0.113.0/24"
          ]
        }
      },
      "Resource":"*"
...
```

1. Does the policy allow the user to terminate any EC2 instance at any time without conditions?

   Answer: No. The policy allows the action but applies a condition.

2. Does the policy allow the user to terminate an EC2 instance from anywhere?

   Answer: No. The call must come from one of the two IP address ranges that are specified in *aws:SourceIp*.

3. If the user's IP address is 192.0.2.243, could they terminate an EC2 instance according to this policy?

   Answer: Yes, because the 192.0.2.0/24 IP address range includes addresses 192.0.2.0 through 192.0.2.255.

Activity: IAM policy analysis (3 of 3)

Consider this IAM policy, and then answer the questions as they are presented.

```
{
  "Version":"2012-10-17",
  "Statement":[{
      "Condition":{
        "StringNotEquals":{
          "ec2:InstanceType":[
            "t2.micro",
            "t2.small"]
        }
      },
  "Resource":"arn:aws:ec2:*:*:instance
/*",
      "Action":[
        "ec2:RunInstances",
        "ec2:StartInstances"
      ],
      "Effect":"Deny"
    }
  ]
}
```

1. What actions does the policy allow?

   Answer: It doesn't allow any actions. The effect is to *deny*.

2. If the policy also included the following statement, what would be the impact?

   ```
   {
      "Effect": "Allow",
      "Action": "ec2:*"
   }
   ```

   Answer: The policy would allow full access to Amazon EC2. However, the policy would only allow a user to launch or start EC2 instances of type t2.micro or t2.small. The allow would be independent of the deny statement with its conditions.

3. If the policy included the statement from question 2, would the user be able to terminate an m3.xlarge instance in the account?

   Answer: Yes. The policy would only deny running or starting a t2.micro or t2.small instance.

Attribute Based Access Control

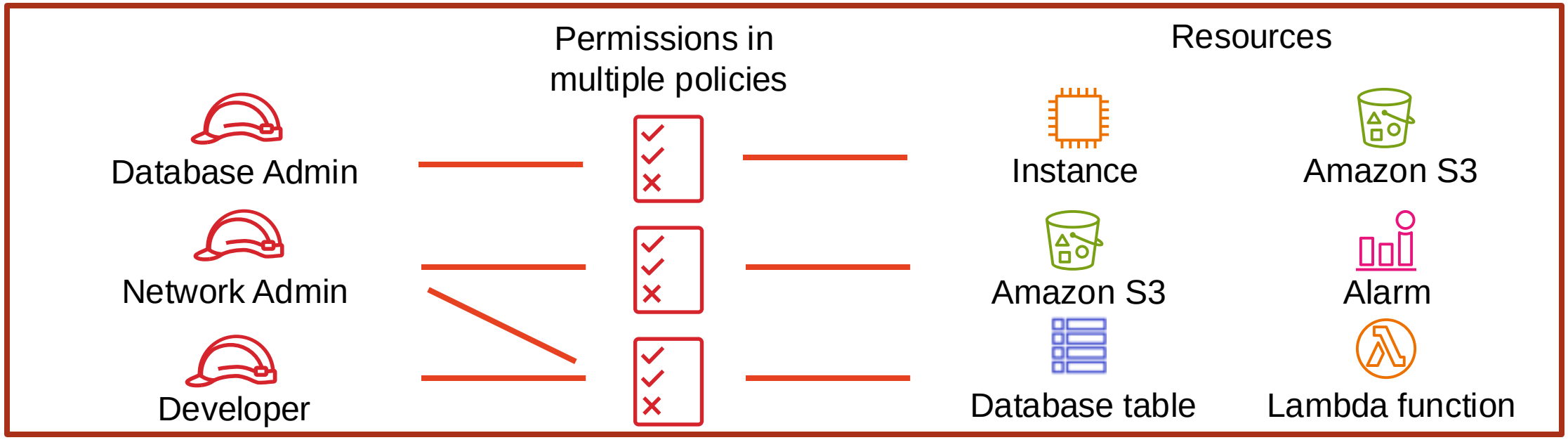
Challenges of scaling with role-based access control (RBAC)

To setup RBAC with IAM, do the following:

- Create an IAM policy with the permissions for the job role. The policy lists the individual resources to be accessed.
- Attach the policy to an IAM entity (user, group, or role).

To update policies when access to a new resource is needed, do the following:

- Update the policy.
- Modify multiple policies if the new resource is used by multiple roles or to add access to multiple resources.



Using attribute-based access control (ABAC)

ABAC

- This authorization strategy defines permissions based on attributes.
- Attributes are a key or a key-value pair.
- In AWS, these attributes are called tags.
- Tags can apply to IAM resources (users or roles) and AWS resources.

Benefits

- It’s more flexible than policies that require you to list each individual resource.
- Granular permissions are possible *without* a permissions update for every new user or resource.
- It’s a highly scalable approach to access control.
- It’s fully auditable.

Tagging in AWS

- Tags are resource metadata consisting of a key/value pair.
- Tags can apply to resources across AWS accounts and IAM users or roles.
- Customers can create user-defined tags.
- Many different AWS API operations return tag keys and values.
- Tags have multiple practical uses like billing, filtered views, and access control.

Example tags applied to an EC2 instance

Key	Value
Name	Web server
Project	Unicorn
Env	Dev

Resource tag and access control example

Allows the policy holder to stop any EC2 instance with a tag
“project = DataAnalytics”

```
{ "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "ec2:StopInstances" ,
   "Resource": "arn:aws:ec2:region:account-id:instance/*",
   "Condition": {
      "StringEquals": {
         "aws:ResourceTag/Project": "DataAnalytics”
             }
    }
   }
  ]
}
```

Specifies a tag belonging to an AWS resource

Specifies the tag key

Specifies the tag value

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_ec2-start-stop-tags.html

Tag and access control example: two conditions

```
{ "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StartStopIfTags",
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances" ],
      "Resource": "arn:aws:ec2:region:account-id:instance/*",
      "Condition": {
          "StringEquals": {
              "aws:ResourceTag/Project": "DataAnalytics",
              "aws:PrincipalTag/Department": "Data"
          }
      }
    }
  ]
}
```

Allows the policy holder with a tag "department = Data" to start or stop any EC2 instance with a tag "project = DataAnalytics"

Specifies a tag belonging to an IAM identity

Specifies the tag key

Specifies the tag value

Tag and access control example: matching value

```
{
  "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
       "ec2:startInstances",
       "ec2:stopInstances" ],
     "Resource": "*",
     "Condition": {
        "StringEquals": {
           "aws:ResourceTag/CostCenter":"${aws:PrincipalTag/CostCenter}”
        }
     }
  }
}
```

Allows a principal to start or stop an Amazon EC2 instance when the instance’s “CostCenter” tag and the principal’s “CostCenter” tag have the same value.

Federating users

Identity federation

A system of trust between two parties to authenticate users and convey information that's

- Identity provider (IdP) is responsible for user authentication.
- Examples:
  - OpenID connect (OIDC) IdPs like Login with Amazon, Facebook, and Google
  - Security Assertion Markup Language (SAML) IdPs like Shibboleth or Active Directory Federation Services
- Service provider (SP) is responsible for controlling access to its resources.
- Examples
  - AWS services
  - Social media platforms
  - Online bank

AWS services that support identity federation

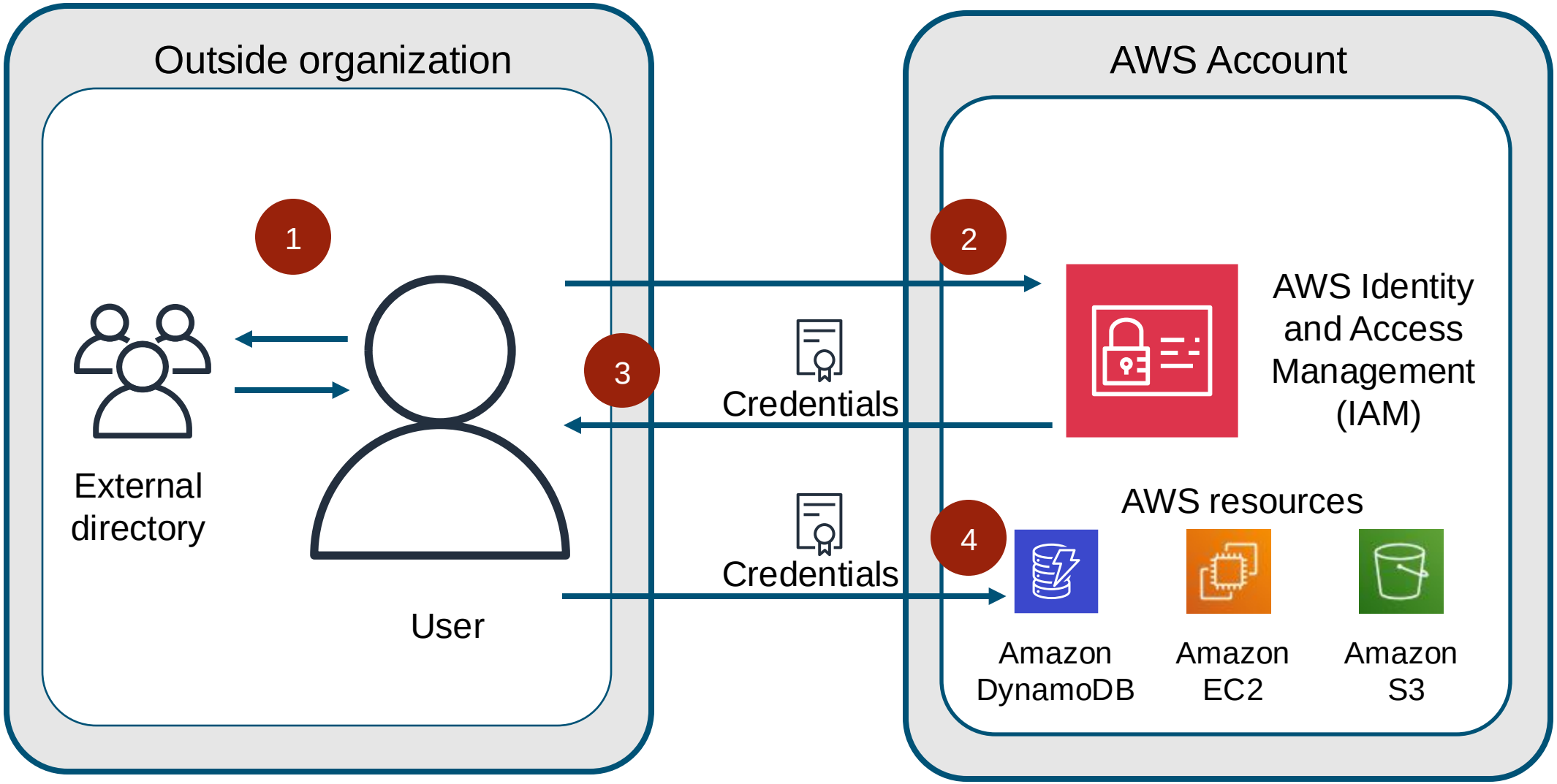
- AWS Identity and Access Management (IAM)

- AWS IAM Identity Center (successor to AWS Single Sign-On)

- AWS Security Token Service (AWS STS)

- Amazon Cognito

Workforce identity federation



AWS IAM Identity Center

IAM Identity Center

- Successor to AWS Single Sign-On
- Can create or connect identities once and manage access centrally across your AWS accounts
- Provides a unified administration experience to define, customize, and assign fine-grained access
- Provides a user portal to access all assigned AWS accounts or cloud applications
- Used optionally in conjunction with IAM

AWS Security Token Service (AWS STS)

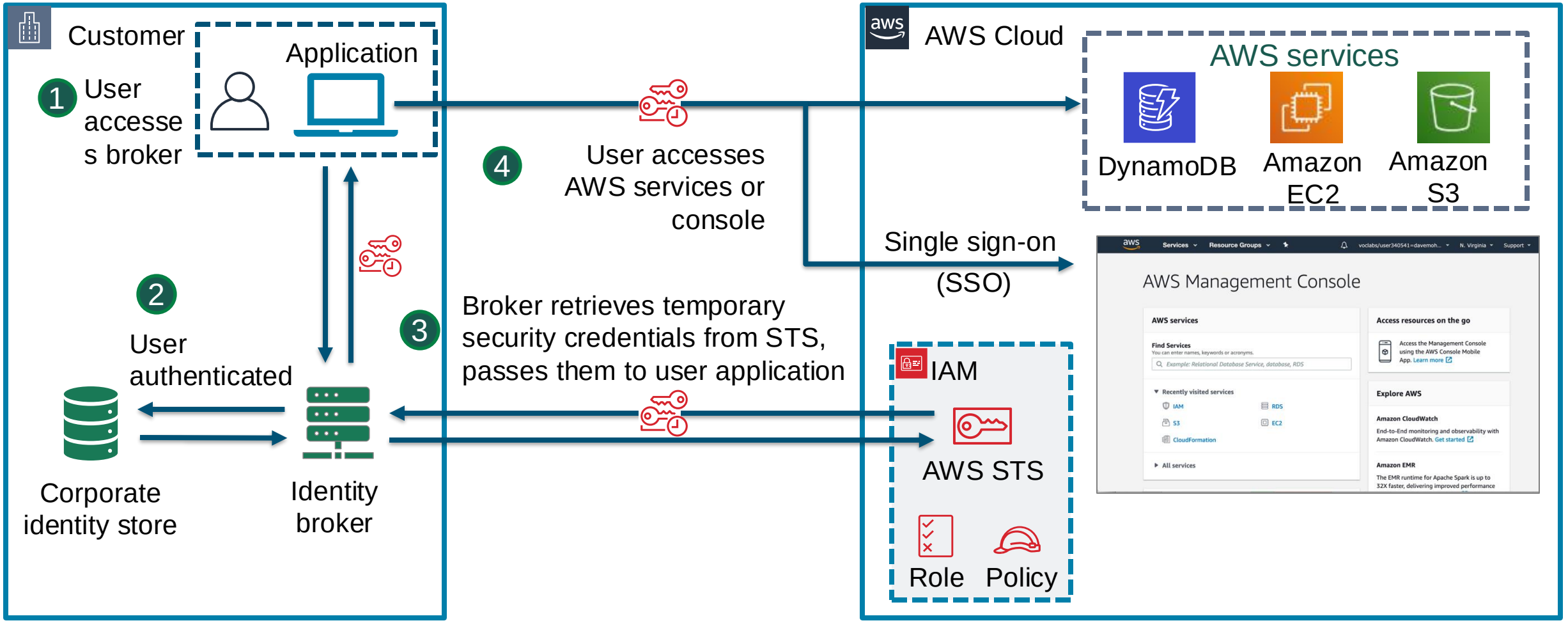
AWS STS

- AWS STS is a web service (API) that enables you to request temporary, limited-privilege credentials.
- The credentials can be used by IAM users, federated users, or applications.

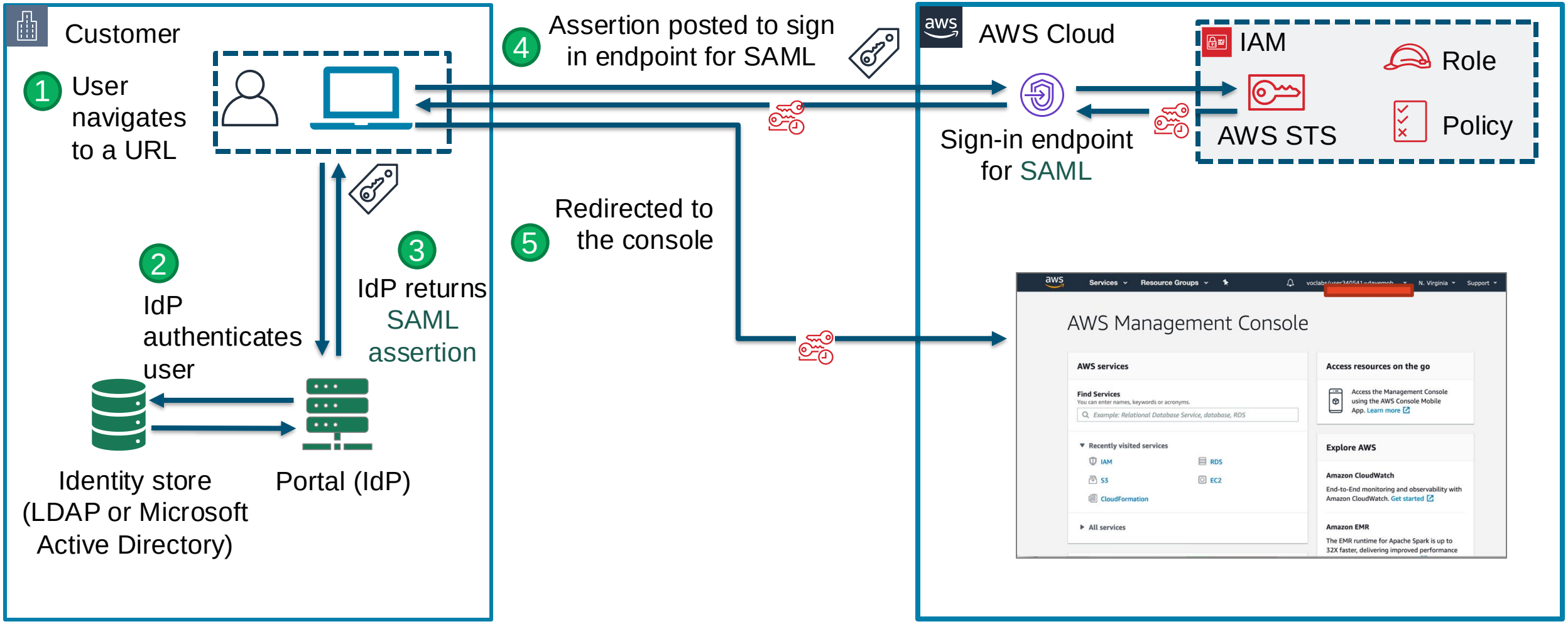
Identity federation to AWS with an identity broker

User signs in with existing credentials for their IdP	Identity broker acts as an intermediary between IdP and SP	AWS STS generates temporary credentials dynamically	Identity broker passes temporary credentials to application
• Users sign in using an identity that is already known by an IdP (for example their Amazon.com ID or a corporate login).	• The identity broker requests temporary credentials from AWS STS.	• The credentials last from a few minutes to several hours and are not recognized after the credentials expire.	• AWS STS returns the temporary credentials to the identity broker. • The identity broker passes them to the application for the user.

Example: Identity federation for AWS Management Console access



Example: Identity federation for AWS Management Console using SAML



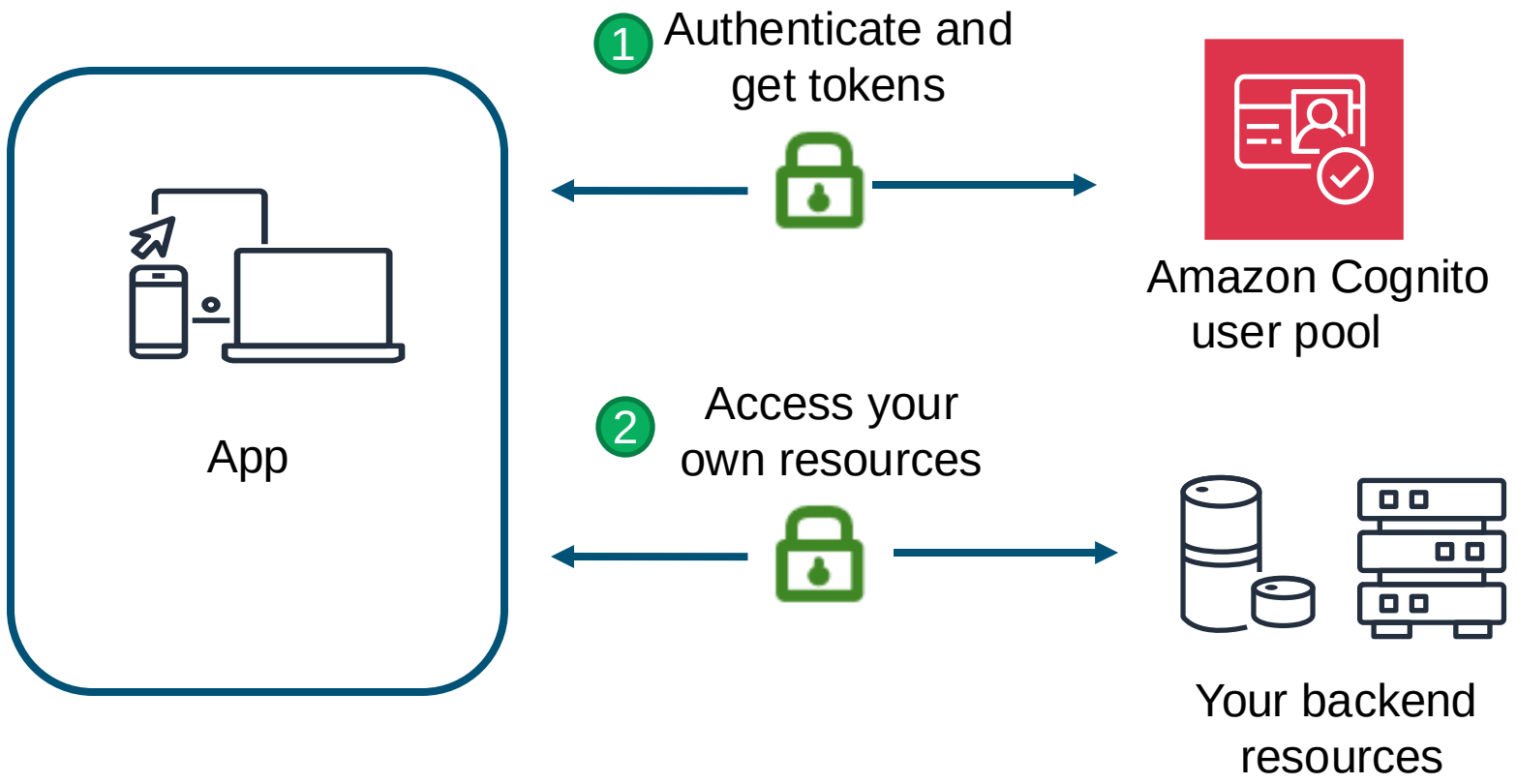
Amazon Cognito

Amazon Cognito

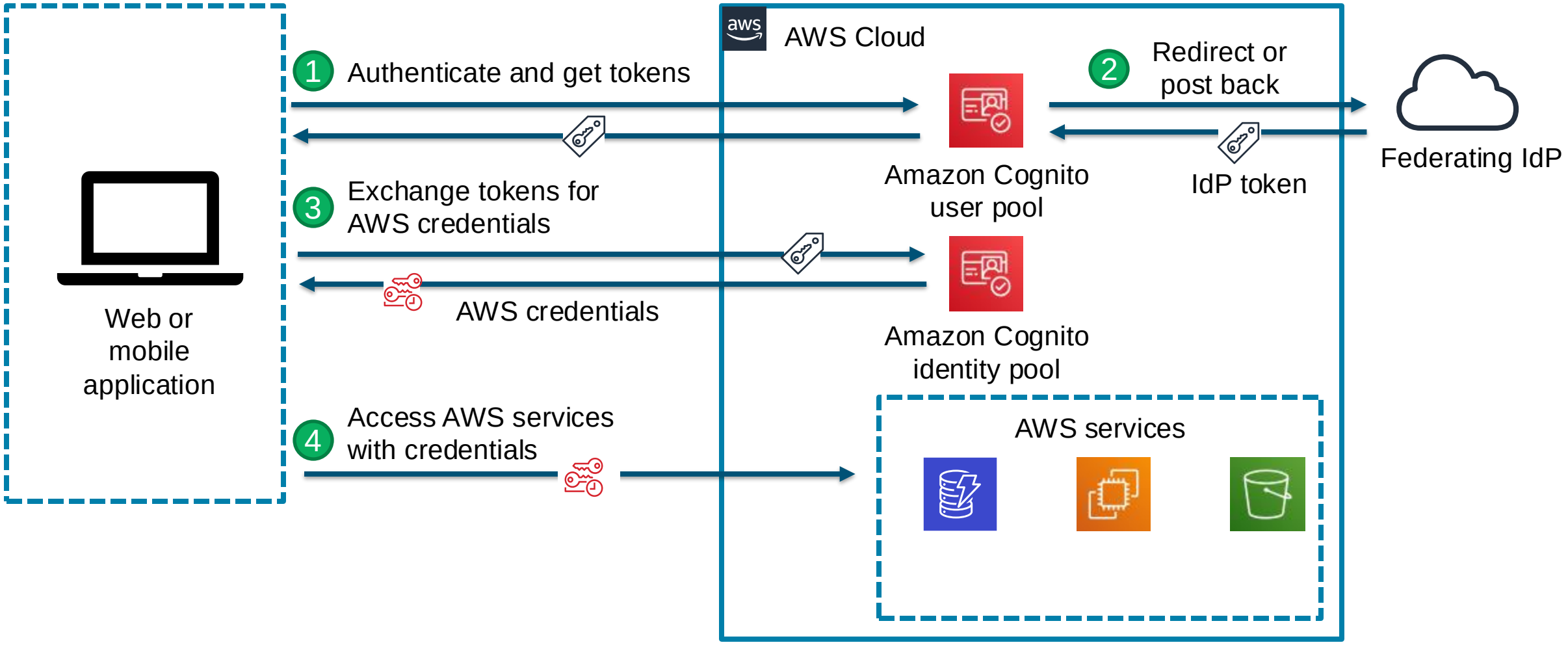
A fully managed service that provides the following services and features:

- Authentication, authorization, and user management for web and mobile applications
- Federated identities for sign in with social identity providers (Amazon, Facebook, Google) or with SAML
- User pools that maintain a directory with user profiles authentication tokens
- Identity pools that enable the creation of unique identities and permissions assignment for users

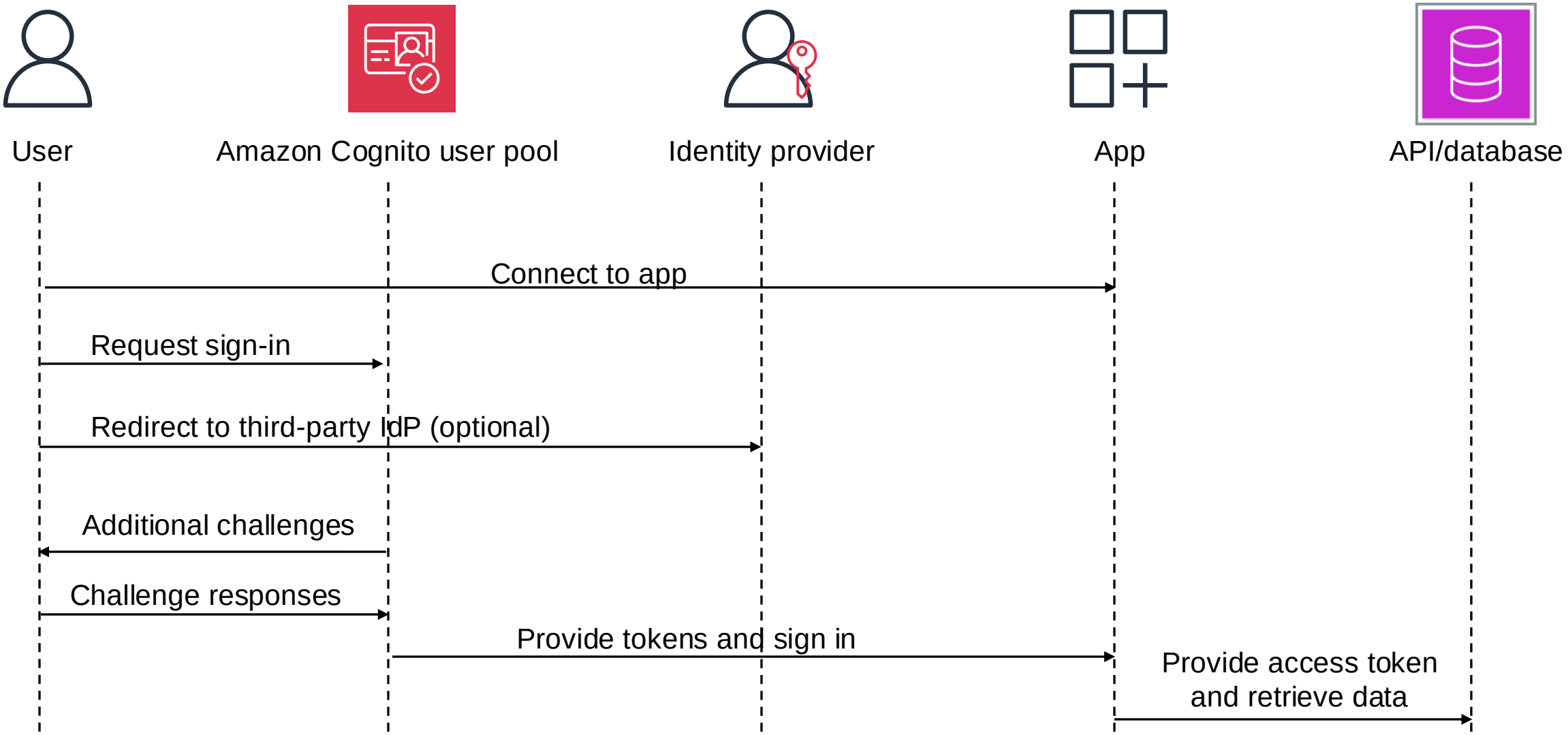
Application access identity federation



Amazon Cognito example

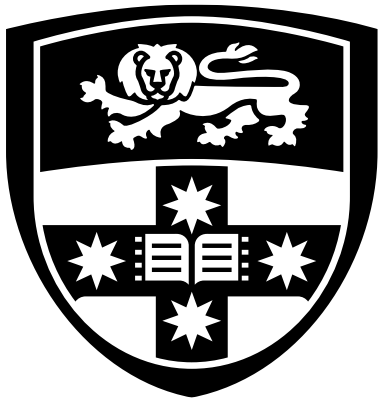


Amazon Cognito user pools



Resources

- ACA Module 3: Securing Access
- ACA Module 9: Securing User, Application and Data



THE UNIVERSITY OF
SYDNEY