Security-Related Best Practices

Secure Entire Infrastructure

Build security into every layer of the infrastructure

Consider

Amazon
CloudTrail

- Use **managed services**
- **Log** access of resources
- **Automate** deployments to keep security consistent
- **Isolate** parts of your infrastructure
  - VPCs, SGs, NACLs

Secure Entire Infrastructure (cont)

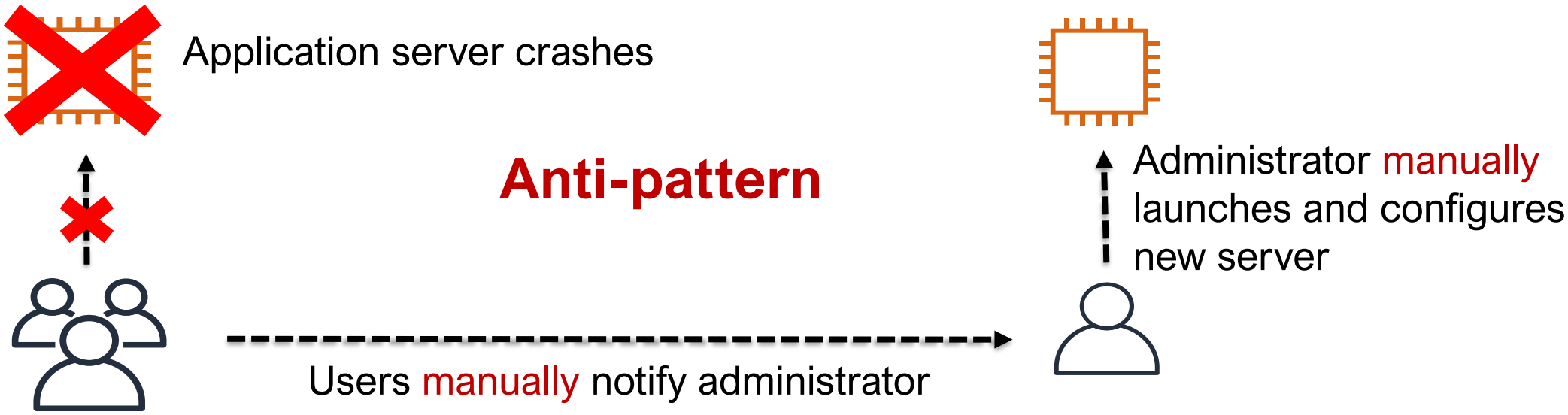
…Consider

- **Encrypt** data in transit and at rest
- Enforce access control **granularly**
  - Using the principle of **least privilege**
  - **IAM policies**
- Use **MFA**

Automate Environment

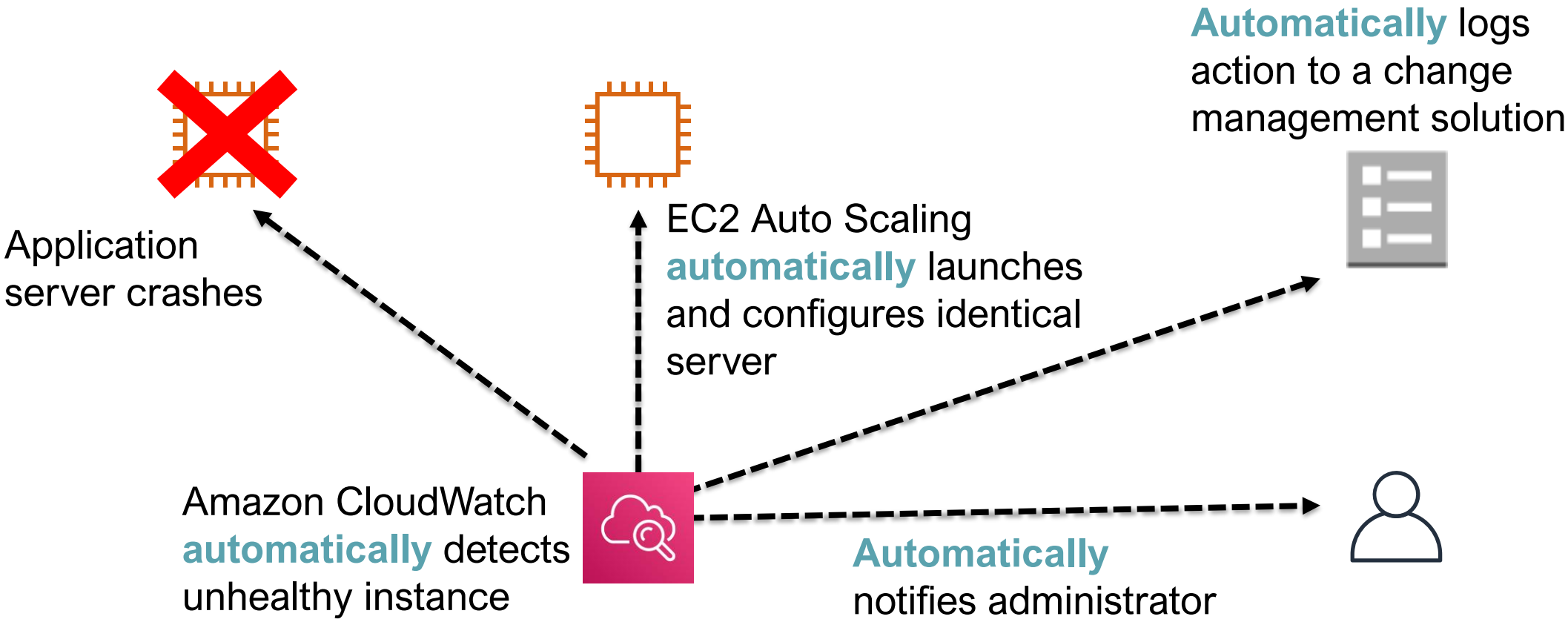
Where possible, automate the provisioning, termination, and configuration of resources

- Scalability, version control, auditing, consistency, security scanning, enforcing security standards, reduce human error and breaches



Automate Environment (cont)

Best Practice



Other Automation Services

AWS CloudFormation

AWS Systems Manager

AWS Config

AWS Elastic Beanstalk

Use Loosely Coupled Components

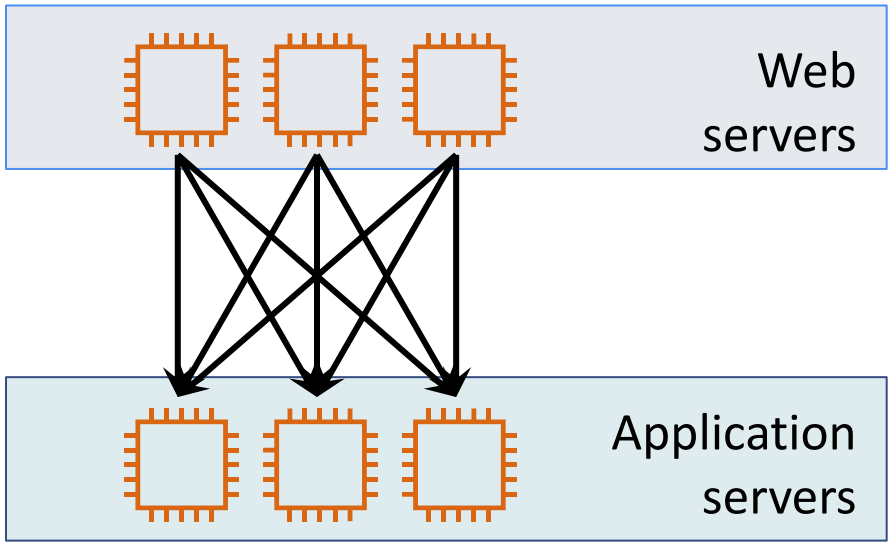
Design architectures with **independent** components

Helps handle **failures** and **scaling**

Amazon Simple Queue Service (Amazon SQS)

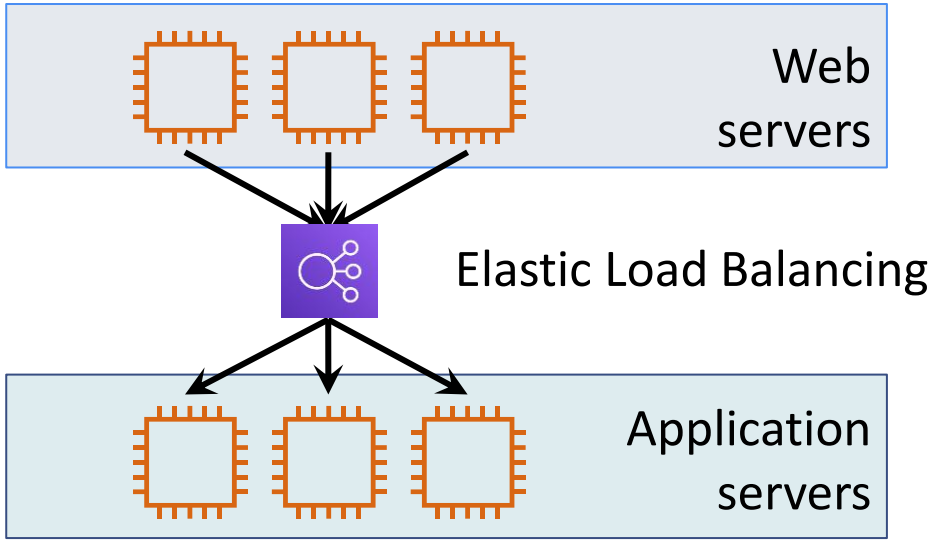
Elastic Load Balancing

Anti-pattern



Web servers tightly coupled to application servers

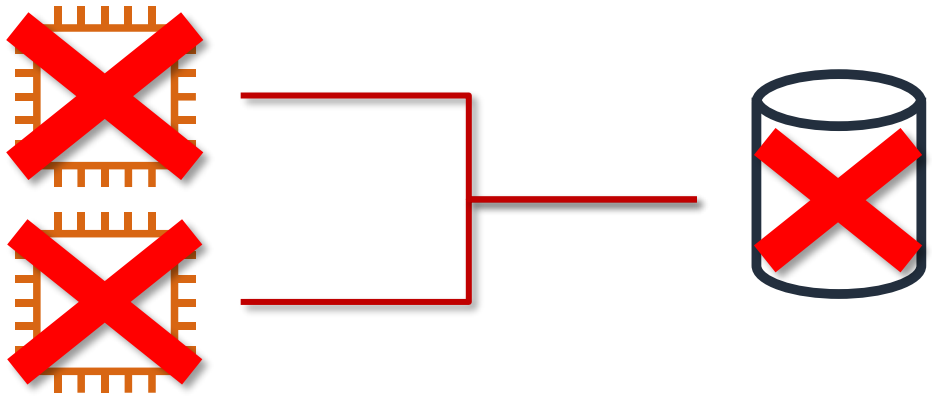
Best Practice



Decoupled with a load balancer

Avoid Single Points of Failure

- *Assume everything fails. Then, design backward.*
- Where possible, use **redundancy** to prevent single points from bringing down an entire system

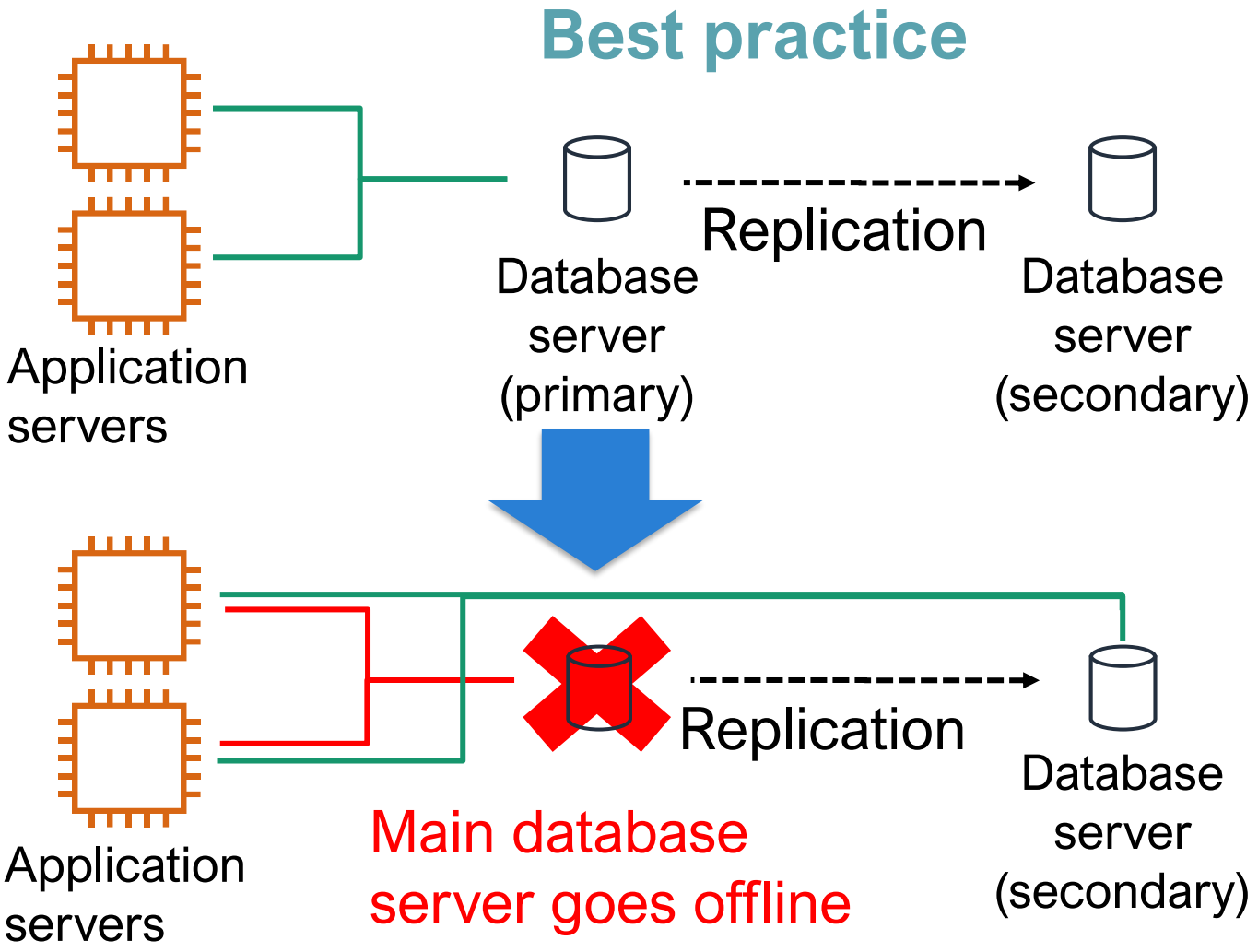


Application servers

Database server

Anti-pattern

Avoid Single Points of Failure (cont)



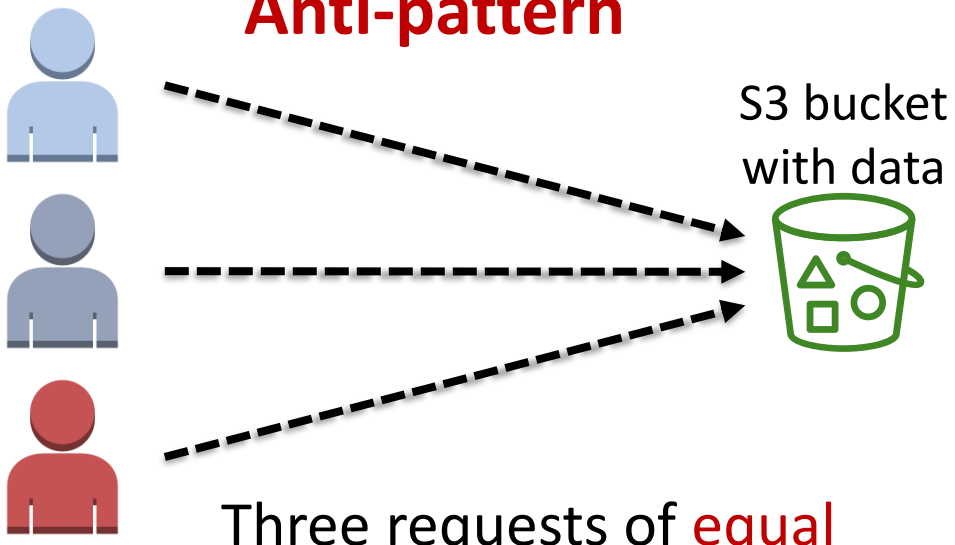
Create a secondary (standby) database server and replicate the data

Secondary server picks up the load

Use Caching

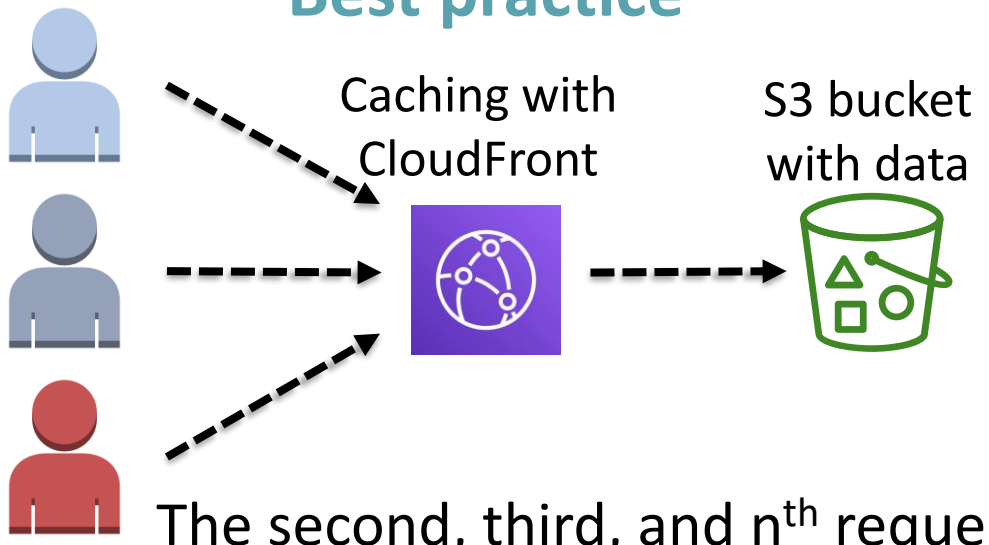
- Caching **minimizes redundant data retrieval** operations, improving performance and cost
  - **Restrict users** from directly accessing files in S3
  - Use **security** features of CloudFront e.g. TLS, DDoS protection (AWS Shield), WAF

Anti-pattern



Three requests of equal latency and cost

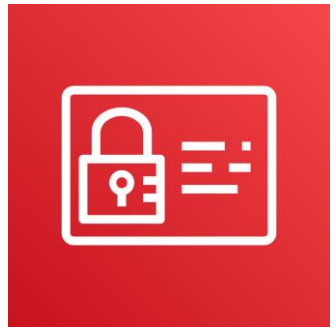
Best practice



The second, third, and n^{th} requests are at a lower latency and cost

Identity and Access Management

AWS IAM



- Use IAM to manage access to **AWS resources**
  - e.g. EC2 instance, S3 bucket
- *Example* –who can terminate EC2 instances
- Define **fine-grained** access rights
  - **Who** can access the resource
  - **Which** resources can be accessed and **what** can the user do to the resource
  - **How** resources can be accessed
  - Which **API calls**
- No-cost AWS account feature

IAM Components

IAM user

Person or **application** that can authenticate with an AWS account

IAM group

Collection of IAM users that are granted identical authorization

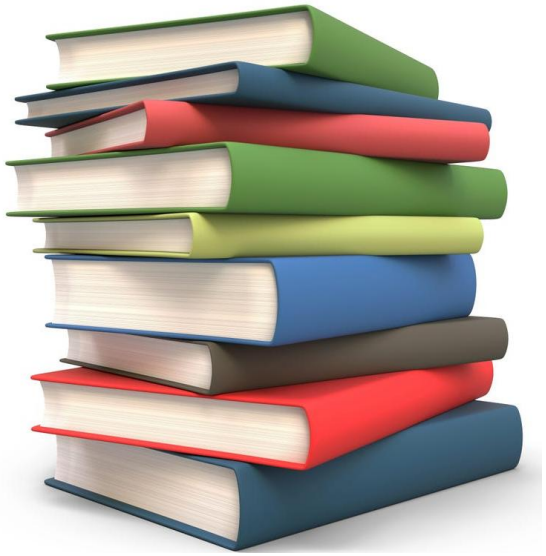
IAM policy

Document defines **which resources can be accessed** and **level of access** to each resource

IAM role

Tool to grant a temporary **permissions** for making AWS service requests

IAM Terminology



- **IAM entity:** Used by AWS for **authentication**
  - Users and roles
- **IAM identity***:* Used to identify and group
  - You can attach a policy to an IAM identity
  - User, group, or role
- **IAM resource***:* User, group, role, policy, and identity provider objects stored in IAM
  - You can add, edit, and remove resources from IAM
- **Principal:** Person/application that uses the AWS account root user, IAM user, or IAM role to sign in and make requests to AWS

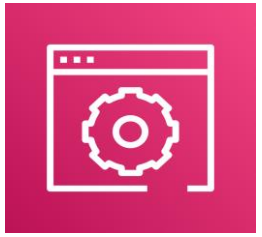
Authenticate as an IAM User

When defining IAM user, select what **types of access** are permitted

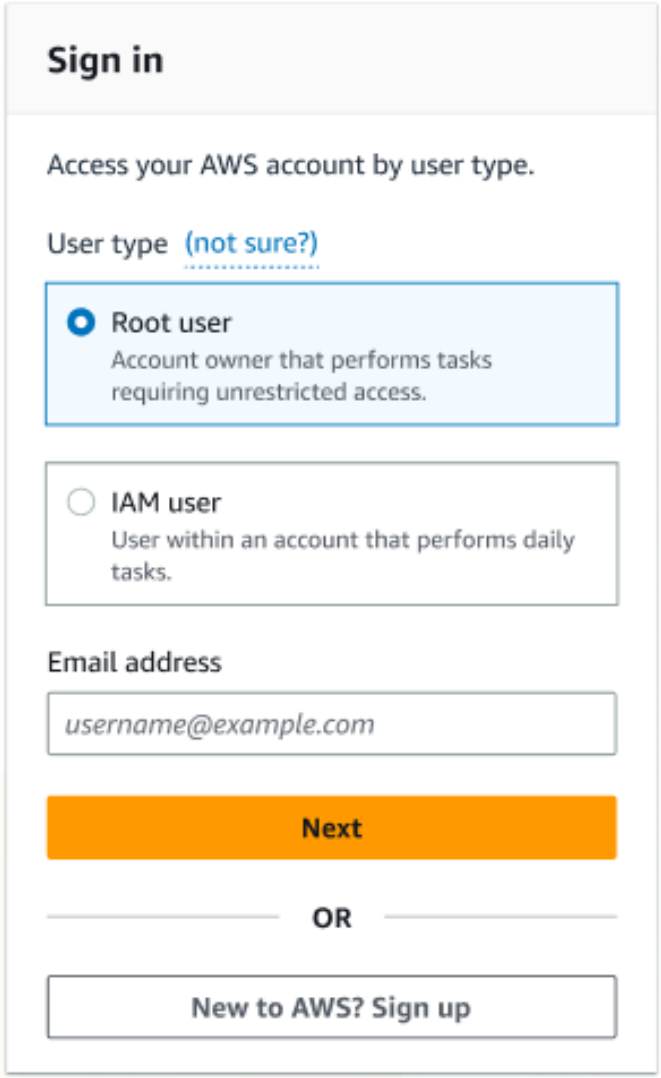
- **Programmatic access**
  - For access to AWS CLI and SDK
  - **Access key ID**
  - **Secret access key**
- **Management Console access**
  - 12-digit **Account ID** or **alias**
  - IAM **username**
  - IAM **password**
  - If enabled, **MFA** prompts for an authentication code

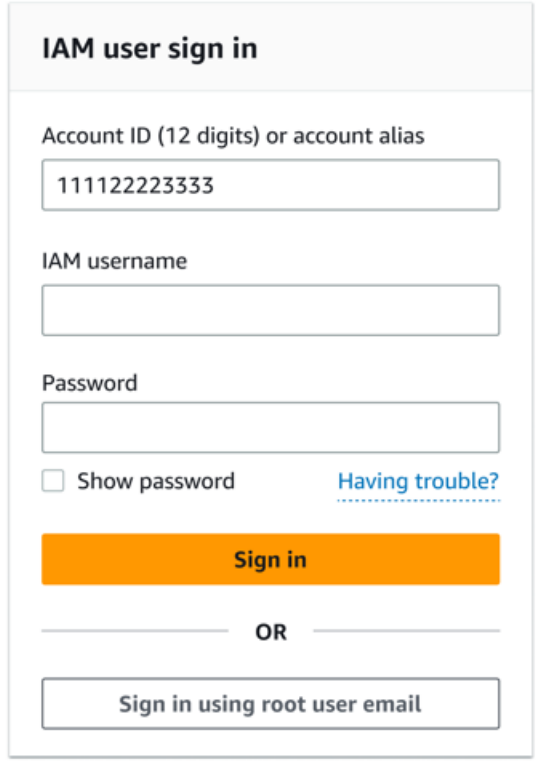
AWS CLI

AWS Tools and SDKs

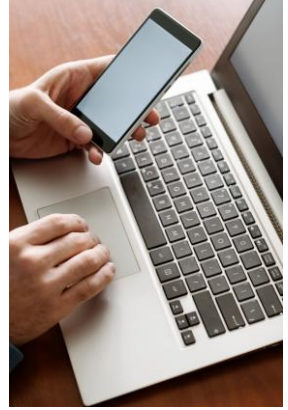


AWS Management Console


Sign in
Access your AWS account by user type.
User type (not sure?)
Root user
Account owner that performs tasks requiring unrestricted access.
IAM user
User within an account that performs daily tasks.
Email address
username@example.com
Next
OR
New to AWS? Sign up
By continuing, you agree to the AWS Customer Agreement or other agreement for AWS services, and the Privacy Notice. This site uses essential cookies. See our Cookie Notice for more information.


IAM user sign in
Account ID (12 digits) or account alias
111122223333
IAM username
Password
Show password
Having trouble?
Sign in
OR
Sign in using root user email

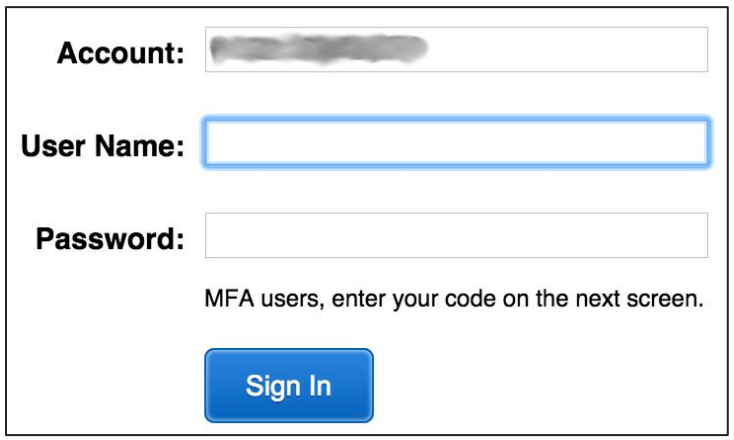
MFA

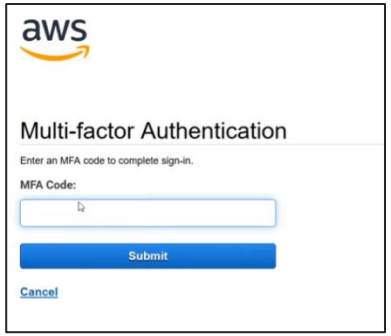


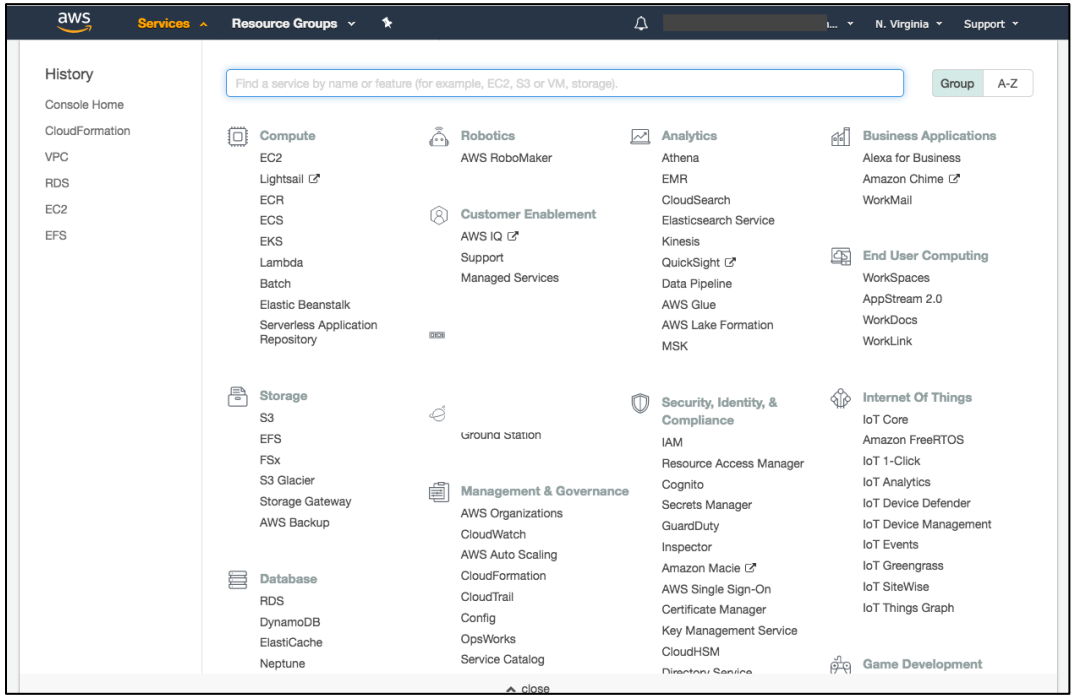
- In addition to username and password
- Requires unique **authentication code** to access AWS services

Username and password

MFA token



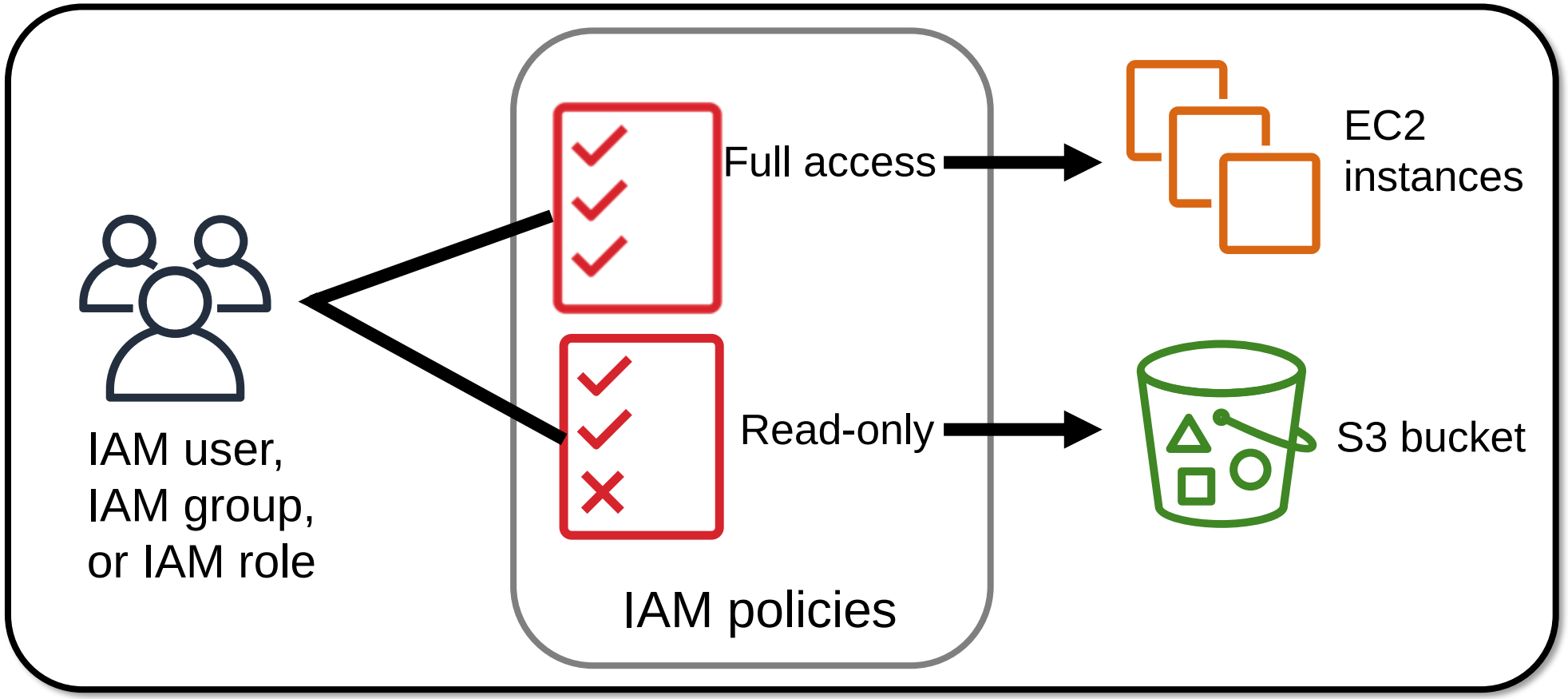




AWS Management Console

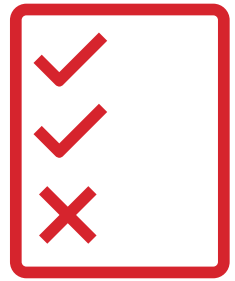
Authorization

- After connecting to the AWS account, what are they allowed to do?
- *By default, IAM users do not have permissions to access any resources*



IAM Permissions

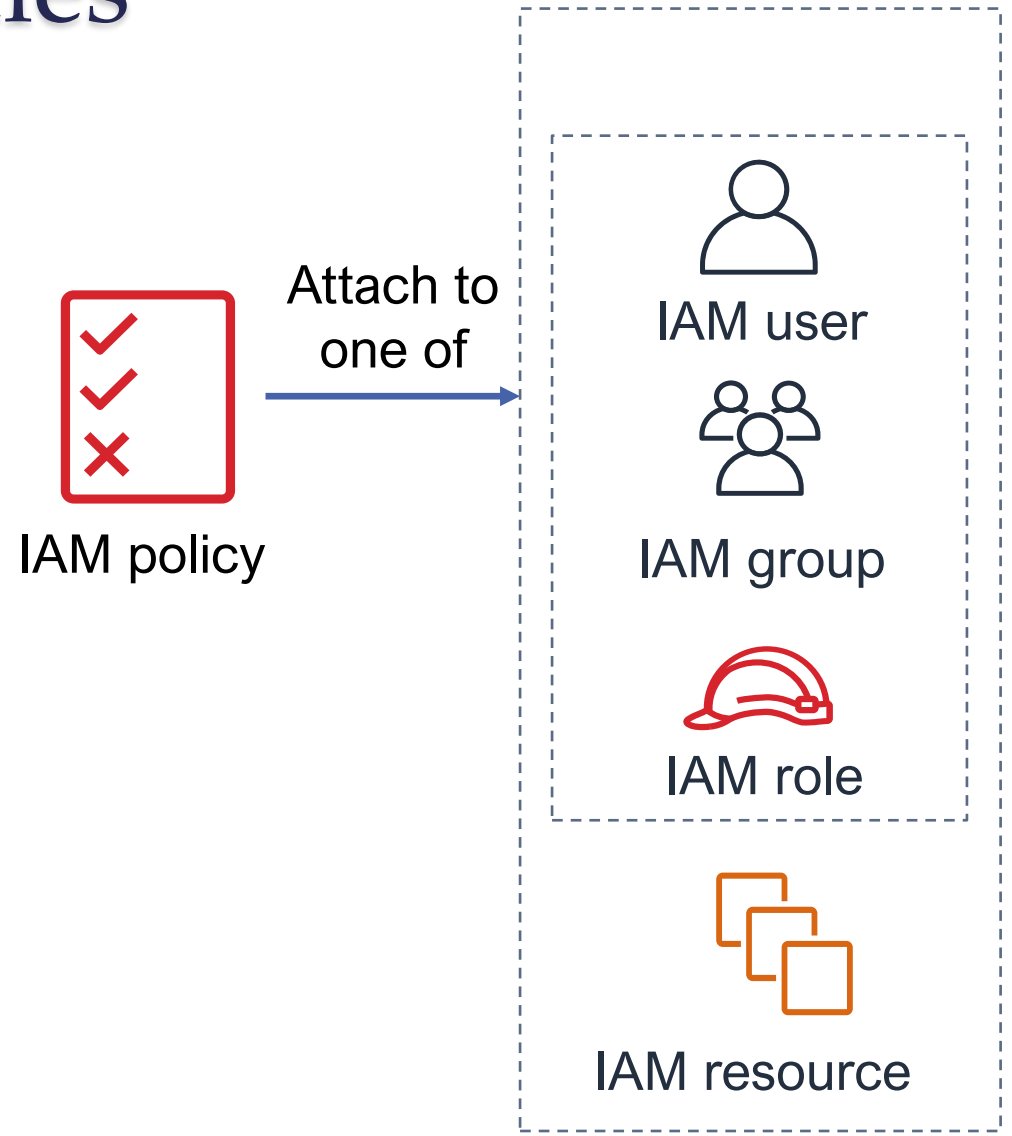
- Assign permissions by creating an **IAM policy**
- Determine **which resources and operations** are allowed
  - All permissions are **implicitly denied** by default
  - If something is **explicitly denied**, it is never allowed
  - Best practice: Follow the principle of **least privilege**
  - Start with **minimum permissions** and add as necessary
- Scope of IAM service configurations is **global**
  - Settings apply across **all** AWS Regions



IAM Policies

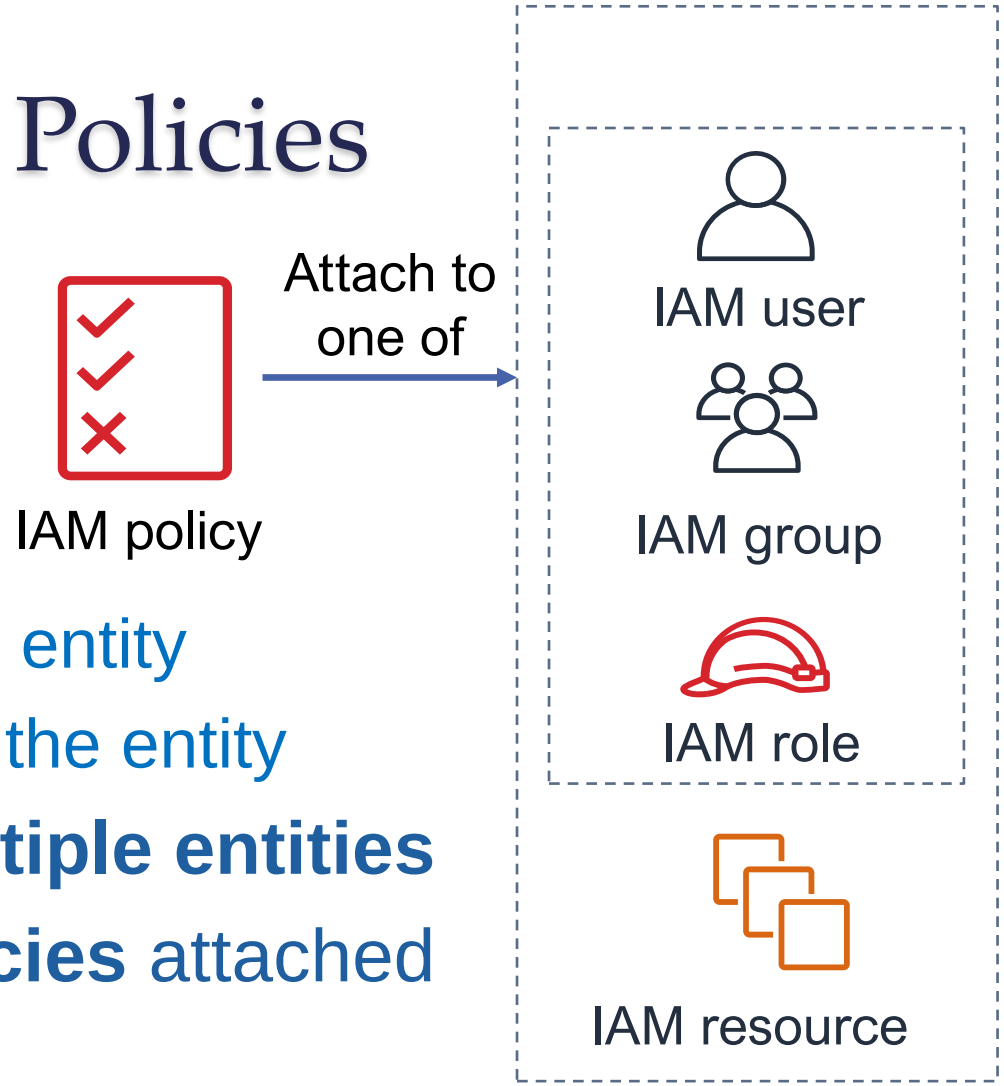
- **Document** that defines permissions
  - Enables **fine-grained** access control
- Two **types** of policies

1. **Resource-based policies**
   - Attached to a resource e.g. S3 bucket
2. **Identity-based policies**
   - **Managed** or **inline**



Identity-based Policies

- Attach to any **IAM identity/entity**
- Policies specify
  - **Actions** that may be performed by the entity
  - Actions that may **not** be performed by the entity
- A single policy can be attached to **multiple entities**
- A single entity can have **multiple policies** attached
- **Managed** or **Inline**



Resource-based Policies

- Attached to a resource
  - **not** to user, group or role
- Specify
  - Who has **access** to the resource and
  - What **actions** they can perform on it
- **Inline** only, not managed
- Supported only by **some** AWS services
- *Question – user Alison granted S3 access through identity-based policy but S3 policy denies access to Alison's group?*

Resource-based Policies

AWS
Account

IAM user
MaryMajor

attached

Identity-based policy

Policy grants list, read objects to the photos bucket

S3 bucket
photos

Defined *inline* on the bucket

Resource-based policy

Policy grants user MaryMajor list, read objects


{
"Version":"2012-10-17",
"Statement":[
{
"Sid":"",
"Effect":"Allow",
"Principal": {
"AWS": ["arn:aws:iam::111122223333:user/John"]
"Action":["s3:*"]
}
]
}
Resource-based policy
GetObject
John
Identity Policy
{
"Version":"2012-10-17",
"Statement":[
{
"Sid":"",
"Effect":"Allow",
"Action":["dynamodb:*"],
"Resource": "*"
}
]
}

IAM Policy Elements

- *Version*
- *Id*
- *Statement*
- *Sid*
- *Effect*
- *Principal*
- *Action*
- *Resource*
- *Condition*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:instance/*"
            ],
            "Condition": {
                "ArnEquals": {"ec2:SourceInstanceARN": "arn:aws:ec2:*:*:instance/instance-id"}
            }
        }
    ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Action":["DynamoDB:*","s3:*"],
    "Resource":[
     "arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"]
    },
    {
    "Effect":"Deny",
    "Action":["dynamodb:*","s3:*"],
    "NotResource":["arn:aws:dynamodb:region:account-number-without-
hyphens:table/table-name”,
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"]
    }
  ]
}
```

Explicit allow gives users access to a specific DynamoDB table and…

…Amazon S3 buckets

Explicit deny ensures that the users cannot use any other AWS actions or resources other than that table and those buckets

An explicit deny statement **takes precedence** over an allow statement

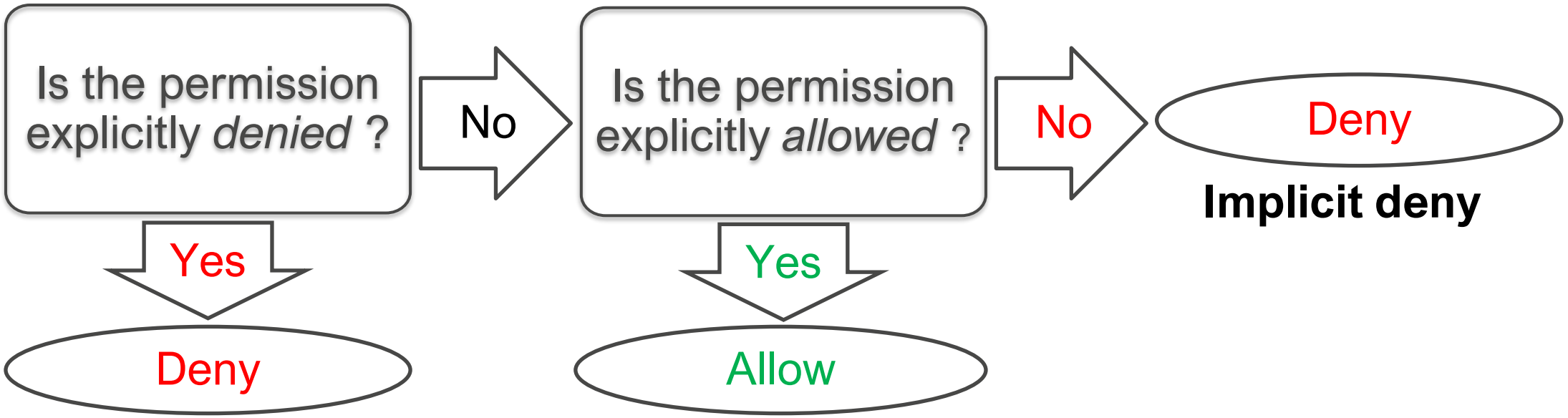
IAM Policy Example

Amazon Resource Names (ARN)

- arn:aws:*service*:*region*:*account-id*:*resource-id*
- arn:aws:*service*:*region*:*account-id*:*resource-type*/*resource-id*
- arn:aws:*service*:*region*:*account-id*:*resource-type*:*resource-id*

e.g. arn:aws:ec2:us-east-1:123456789012:vpc/vpc-0e9801d129EXAMPLE

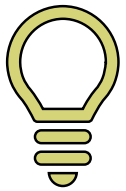
How IAM Determines Permissions

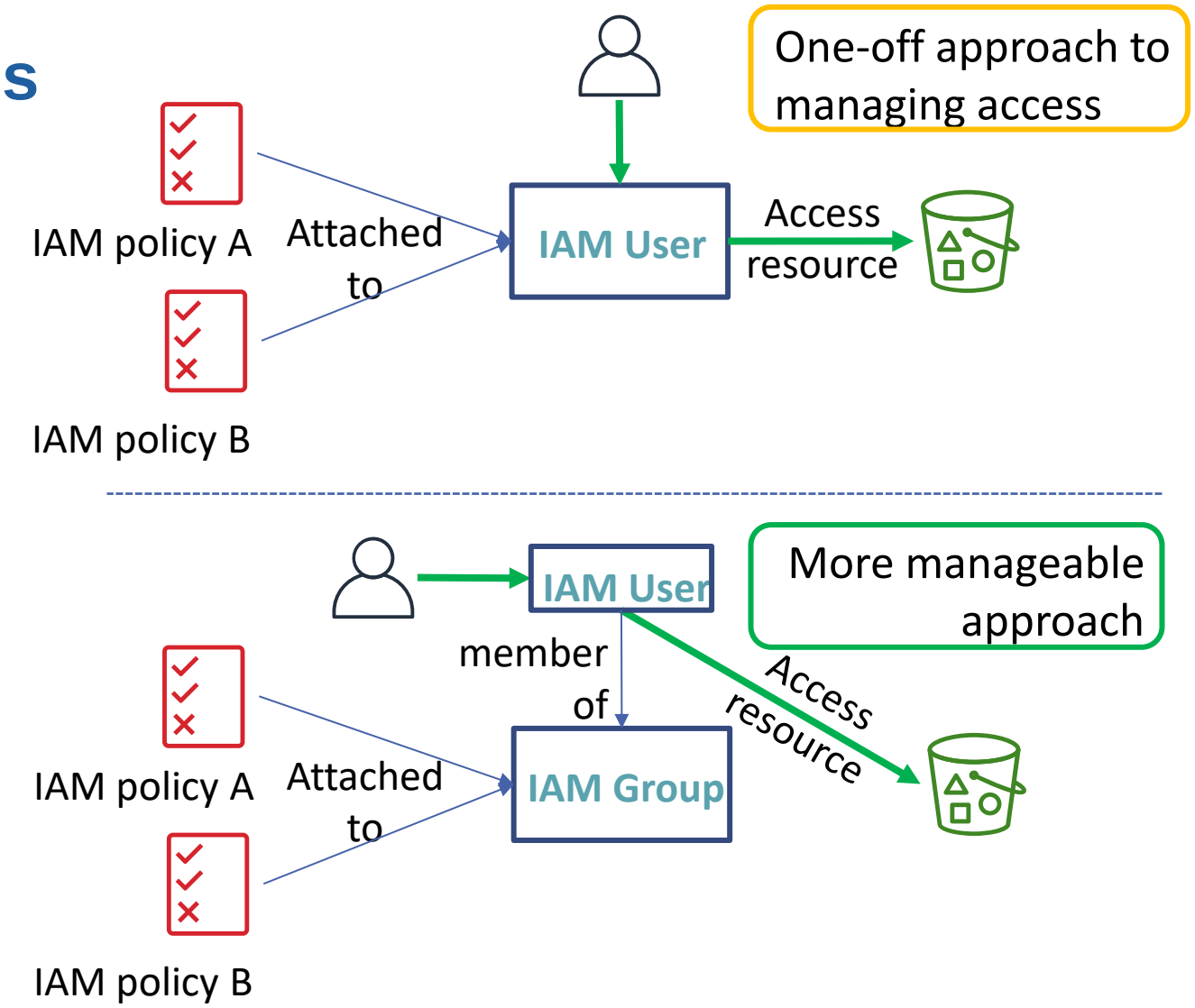


Not explicitly denied and is explicitly allowed

IAM Groups

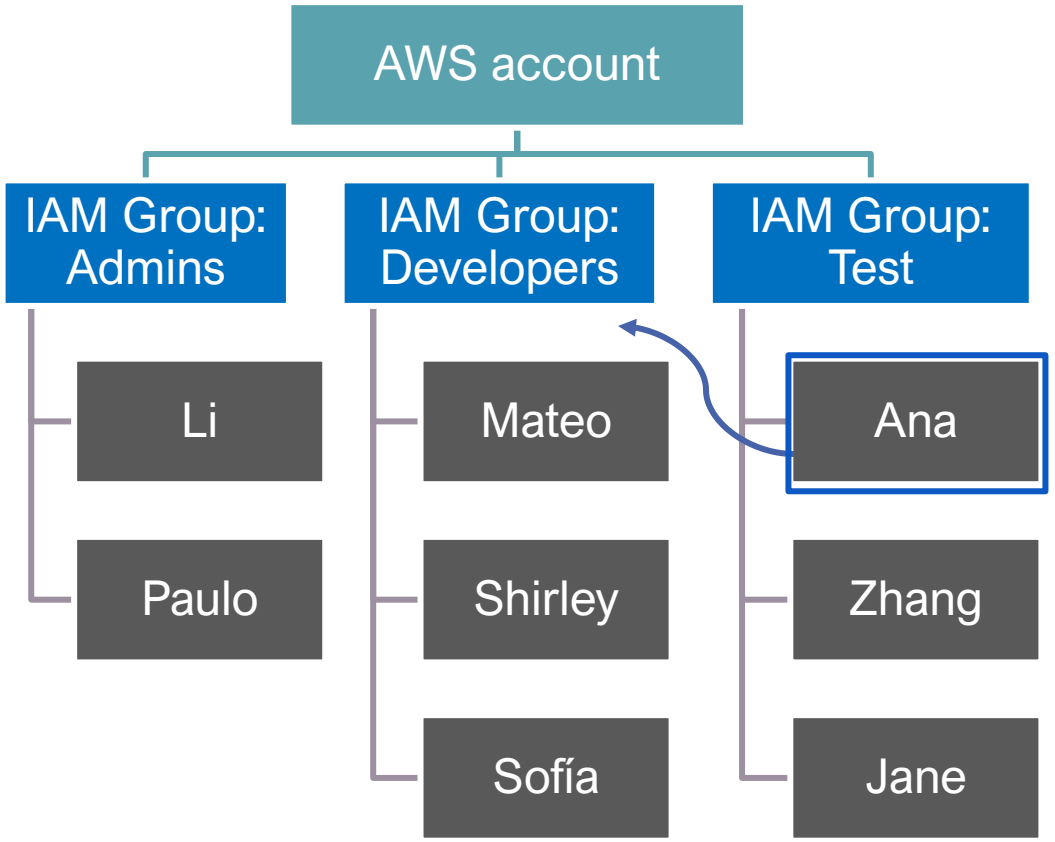
- Users inherit group **permissions**
  - Easier to **manage access** for multiple users
- Combine approaches for **fine-grained** access
  - Add user to group to apply standard access **based on job**
  - Attach additional policy to user for needed **exceptions**



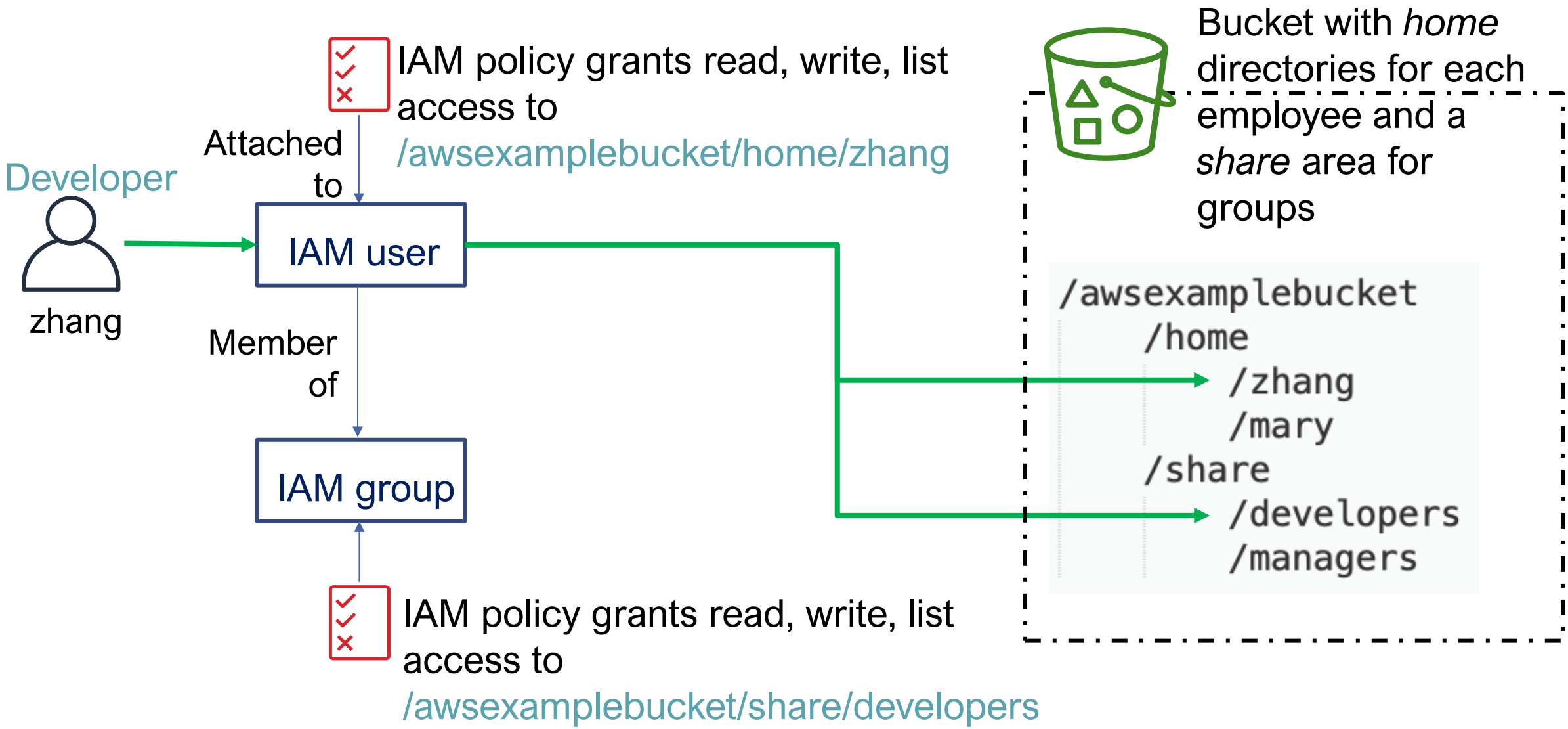


Example IAM Groups

- If a new developer is hired, add them to Developer group
  - Immediately **get the same access** granted to other developers
- If Ana takes on the role of developer
  - **Remove** her from Test group
  - **Add** her to Developer group
- Users can belong to **multiple** groups
  - The **most restrictive** policy applies



Use Case for IAM with Amazon S3

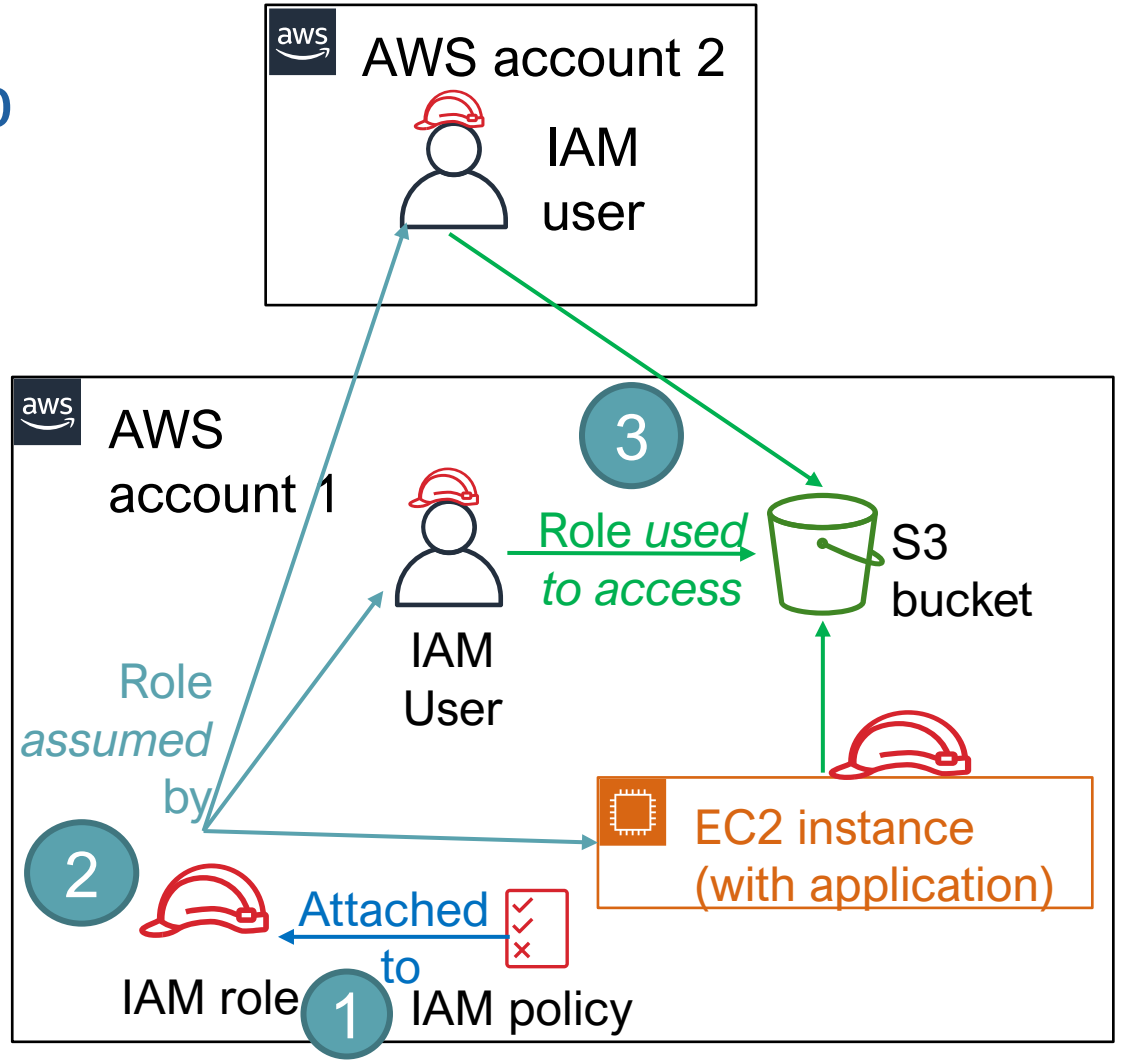


IAM Roles

- IAM identity with specific permissions
- Similar to IAM user
  - **Attach permissions** policies to it
- Different from IAM user
  - **Not** uniquely associated with one person
  - **Assumable** by a person, application or service
- Role provides **temporary** security credentials

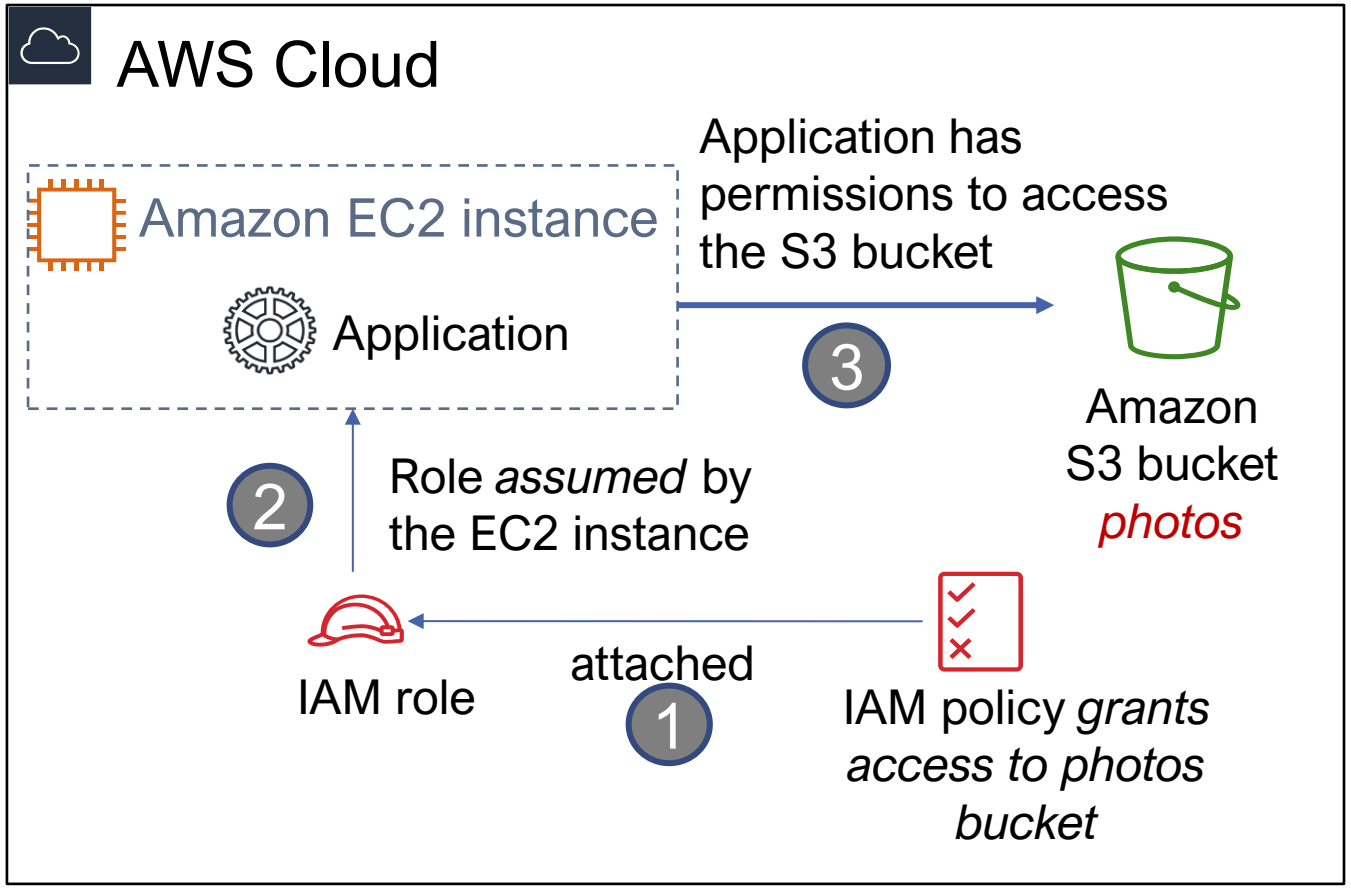
IAM Role Use Cases

- Provide **AWS services** with access to other 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)
  - *User prior permission temporarily* ***forgotten***

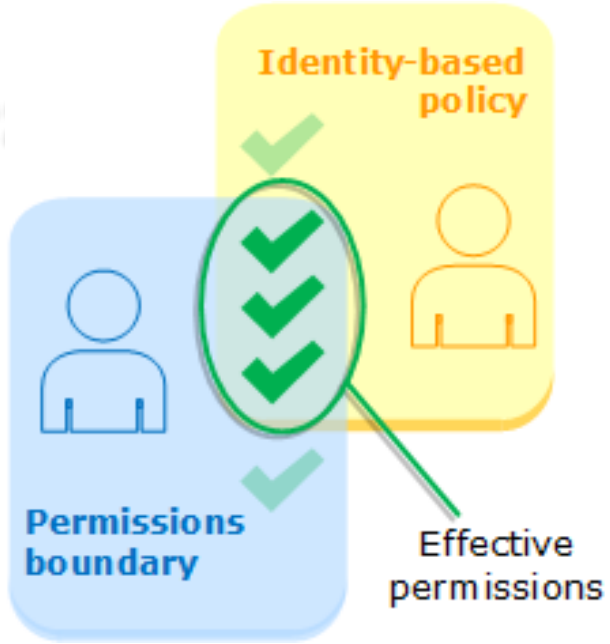


Example Use of IAM Role

- An application that runs on an EC2 instance **needs access to an S3 bucket**
- Define an **IAM policy** that grants access to the S3 bucket
- Attach the policy to a **role**
- Allow the **EC2 instance** to **assume** the role



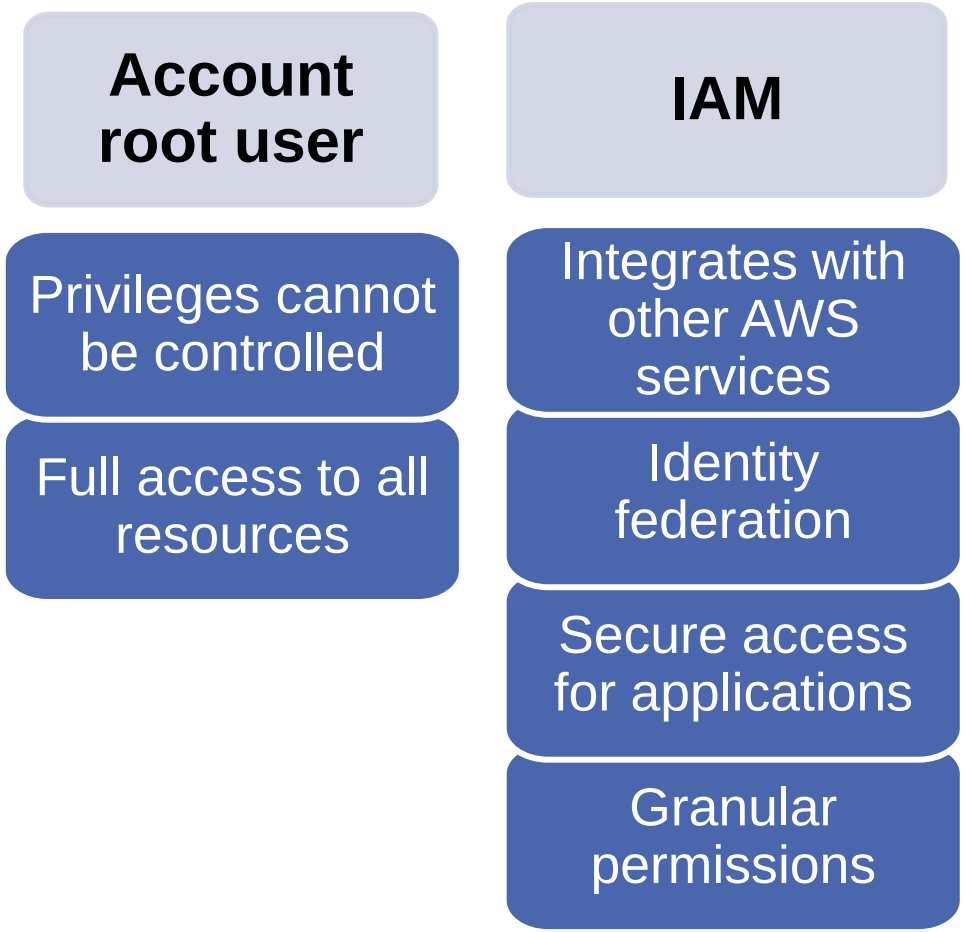
Types of IAM Role Policie



- **Permission policies**
  - Inline and managed
  - Role permissions
- **Permissions boundary**
  - **Max permissions** an identity-based policy can grant to an IAM entity
  - Permissions boundaries can be applied to IAM users and roles
- **Trust policy**
  - Which principals can assume the role and under which **conditions**
  - **Resource-based policy** for the IAM role

AWS Account Root vs IAM

- Best practice: **Do not** use the AWS account **root** user except when necessary
- Actions that can **only** be done with the root user e.g.
  - Update its **password**
  - Change the **AWS Support plan**
  - Change **account settings** e.g. contact information, allowed Regions



Exercise: IAM Policy Analysis (1 of 3)

Consider this IAM policy, 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 you access to?
   - ANSWER:
2. Does it allow you to create an IAM user, group, policy, or role?
   - ANSWER:
3. Go to https://docs.aws.amazon.com/IAM/latest/UserGuide/ and in the left navigation expand *Reference > Policy Reference > Actions, Resources, and Condition Keys.* Choose *Identity And Access Management*. Scroll to the **Actions Defined by Identity And Access Management** list. Name three specific actions that the iam:Get* action allows.
   - ANSWER:

Actions defined by AWS Identity and Access Management

You can specify the following actions in the `Action` element of an IAM policy statement. Use policies to grant permissions to perform an operation in AWS. When you use an action in a policy, you usually allow or deny access to the API operation or CLI command with the same name. However, in some cases, a single action controls access to more than one operation. Alternatively, some operations require several different actions.

The **Resource types** column of the Actions table indicates whether each action supports resource-level permissions. If there is no value for this column, you must specify all resources ("*") to which the policy applies in the `Resource` element of your policy statement. If the column includes a resource type, then you can specify an ARN of that type in a statement with that action. If the action has one or more required resources, the caller must have permission to use the action with those resources. Required resources are indicated in the table with an asterisk (*). If you limit resource access with the `Resource` element in an IAM policy, you must include an ARN or pattern for each required resource type. Some actions support multiple resource types. If the resource type is optional (not indicated as required), then you can choose to use one of the optional resource types.

The **Condition keys** column of the Actions table includes keys that you can specify in a policy statement's `Condition` element. For more information on the condition keys that are associated with resources for the service, see the **Condition keys** column of the Resource types table.

For details about the columns in the following table, see Actions table.

Actions	Description	Access level	Resource types (*required)	Condition keys	Dependent act
AddClientIDToOpenIDConnectProvider	Grants permission to add a new client ID (audience) to the list of registered IDs for the specified IAM OpenID	Write	oidc-provider*		

Exercise: IAM Policy Analysis (2 of 3)

```
{
    "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 you to terminate any EC2 instance at any time without conditions?
   - ANSWER:

2. Are you allowed to make the terminate instance call from anywhere?
   - ANSWER:

3. Can you terminate instances if you make the call from a server that has an assigned IP address of *192.0.2.243*?
   - ANSWER:

Exercise: IAM Policy Analysis (3 of 3)

```
{
    "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:

2. Say that the policy included an additional statement object, like this:

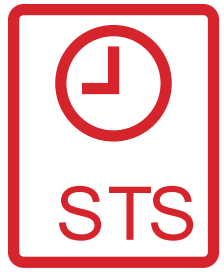
   { "Effect": "Allow",
   "Action": "ec2:*" }

   How would the policy restrict the access by this additional statement?
   - ANSWER:

3. If the policy included both the statement on the left and the statement in question 2, could you terminate an **m3.xlarge** instance that existed in the account?
   - ANSWER:

Grant Permissions to Assume a Role

- For an IAM user, application, or service to assume a role, you must **grant permissions to switch to the role**
- **AWS STS**
  - Enables to request **temporary limited-privilege** credentials
  - Credentials can be used by **IAM users, applications** or **federated users**
  - **STS API** include **AssumeRole** operation returns temporary credentials



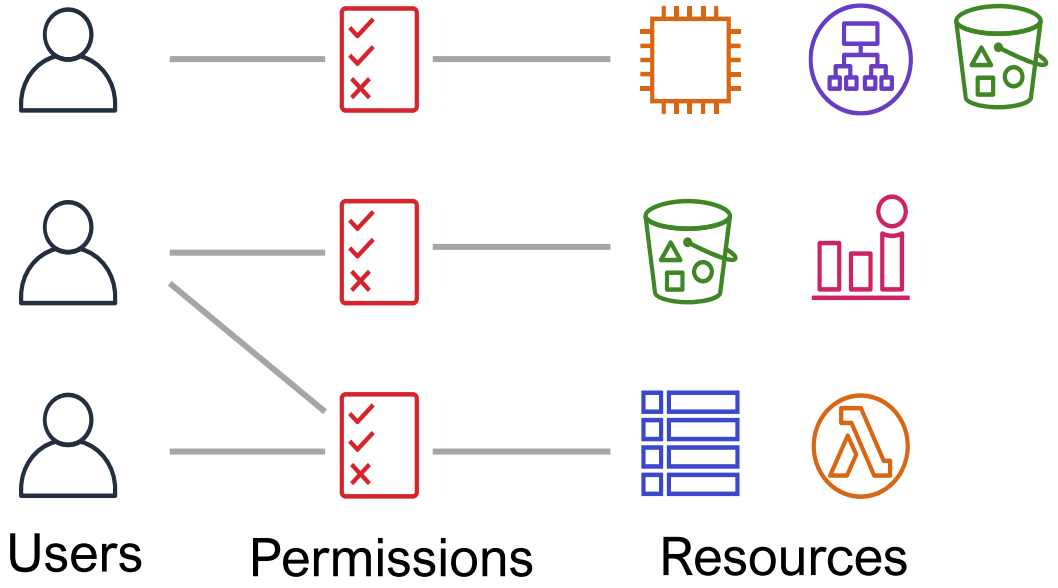
AWS Security Token Service (STS)

```
{
    "Version": "2012-10-17",
    "Statement": {
     "Effect": "Allow",
     "Action": "sts:AssumeRole",
     "Resource": "arn:aws:iam::123456789012:role/Test*"
    }
}
```

RBAC

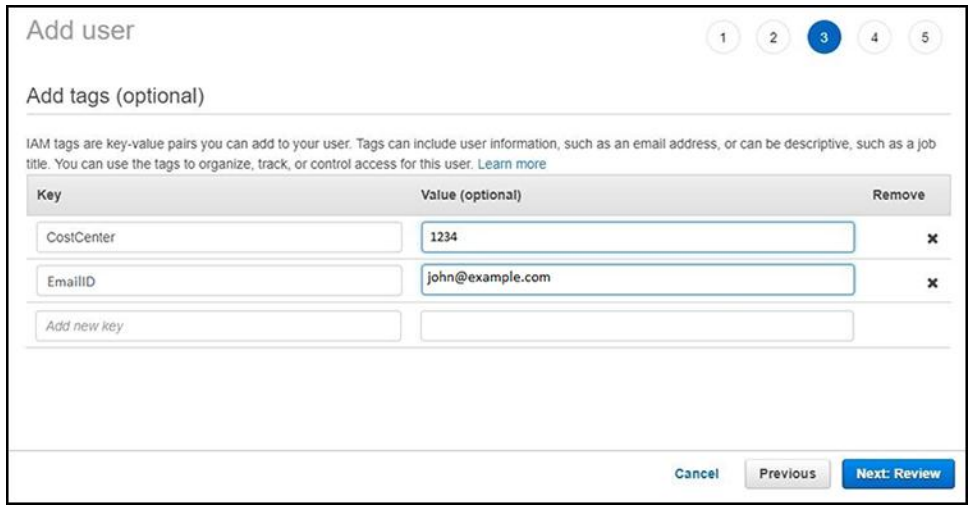
- Grant users specific permissions based on **job function**

  e.g. database admin
- Create a distinct **IAM role** for each permission combination
- Update permissions by adding **access for each new resource**
  - It can become **time-consuming** to keep updating policies



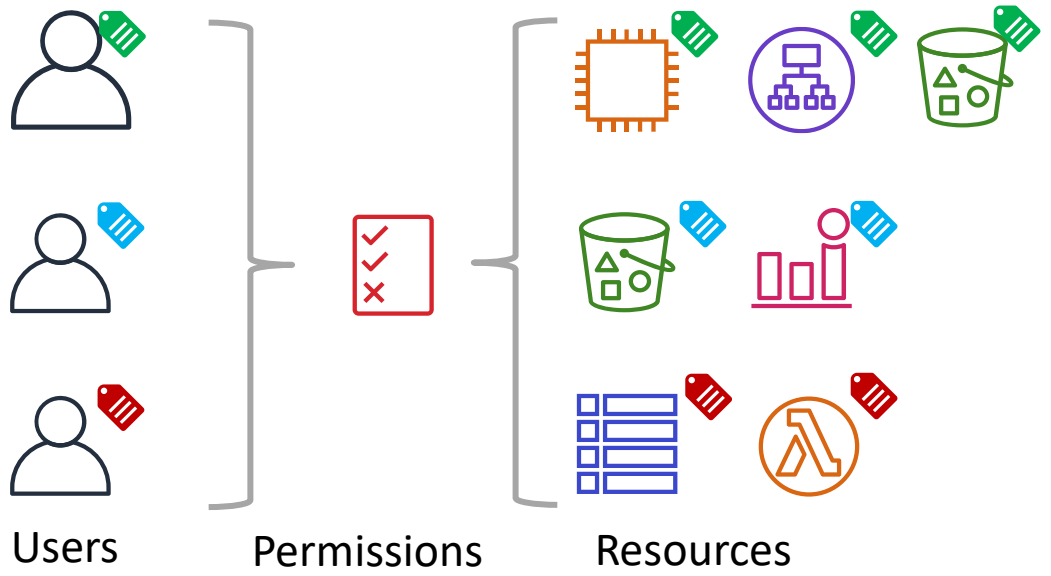
Best Practice: Tagging

- Define custom **metadata**
- Consists of a **name** and **optional value**
  - Can be applied to **resources**
  - Tag keys and values are returned by **API operations**
- Uses
  - Billing, filtered views, **access control**, **confidentiality level**, ...
- Tags can also be applied to **IAM users** and **roles**
- **Example tags** applied to an EC2 instance
  - Name = web server, Project = phoenix, Stack = development



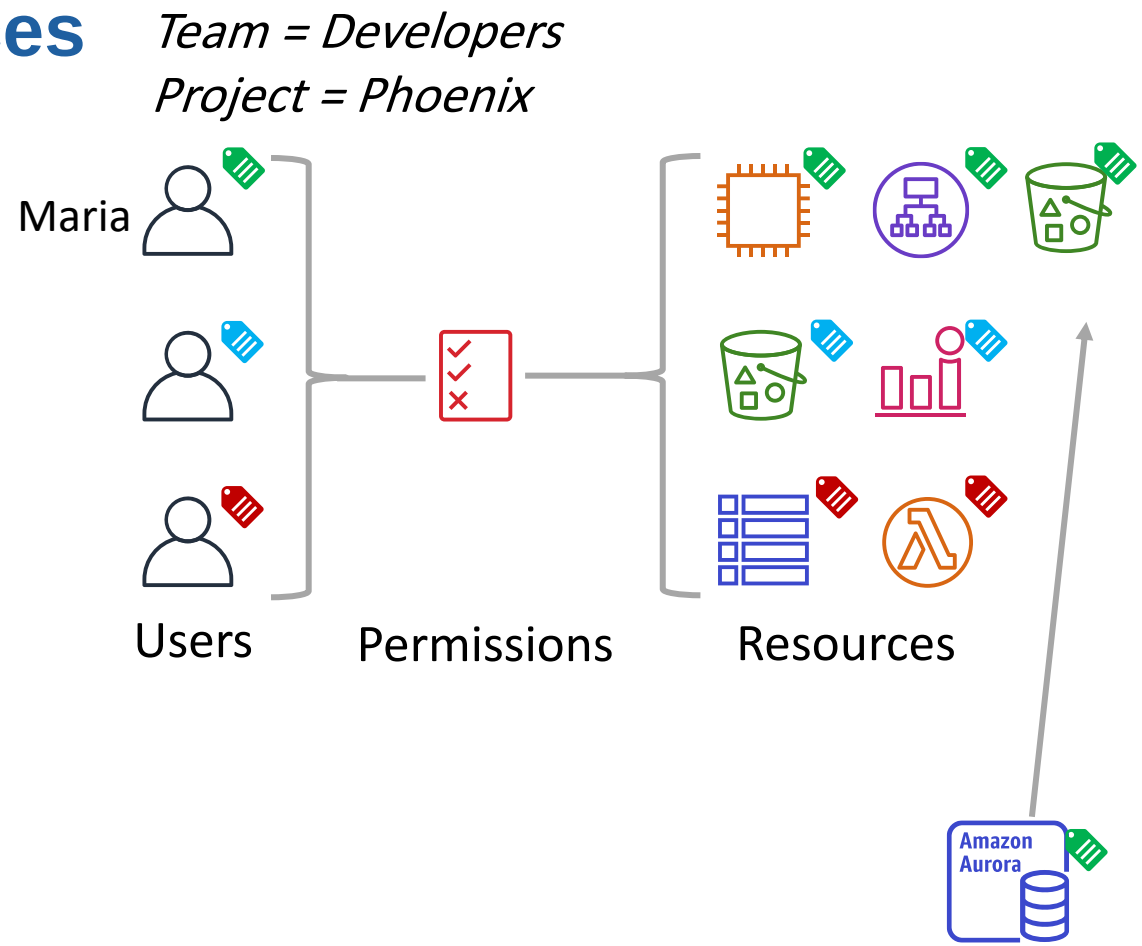
ABAC

- **Scalable** approach to access control
  - Attributes are keys or kv pairs e.g. **tags**
- Permissions rules are **easier** to maintain with ABAC than RBAC
- **Benefits**
  - **Granular permissions** automatically apply based on attributes
    - **Matching tags** between IAM user and resource
  - Possible **without** a permissions update for every new user or resource
  - **Auditable**



Applying ABAC to Organization

1. Set access control attributes on **IAM identities**
2. **Require** attributes for new **resources**
   - Enforce **policy rule**
3. Configure **permissions** based on attributes
4. Test
   a) Create new resources
   b) Verify permissions automatically apply



Externally Authenticated Users

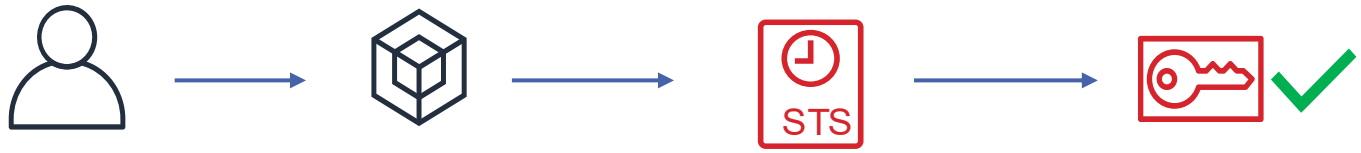
Identity federation

- User authentication completed by a system **external** to AWS account

  e.g. corporate directory
- Allows access through existing identities **without** creating IAM users

Options

1. **AWS STS**
   - Corporate identity providers (**IdP**)
   - Custom identity broker application
2. **SAML** (Security Assertion Markup Language)
3. **Amazon Cognito**



User accesses identity broker via application

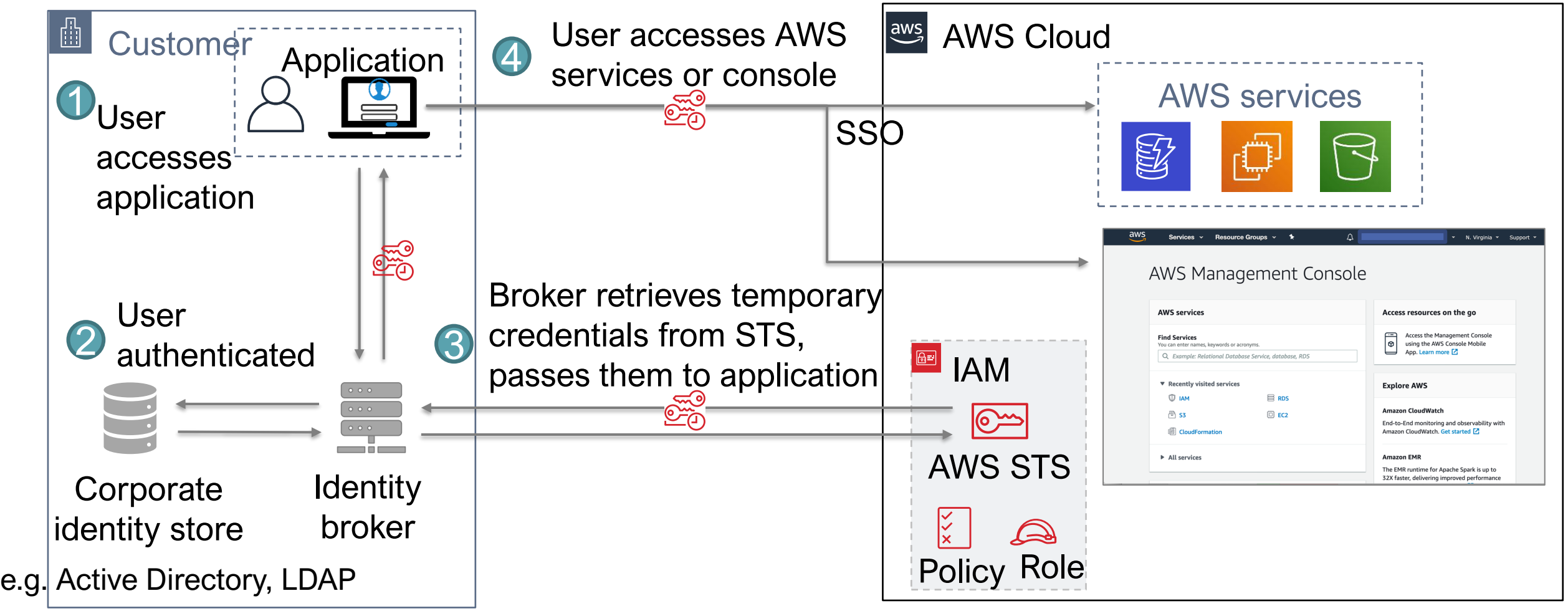
Identity broker authenticates user

Requests temporary credentials from AWS STS

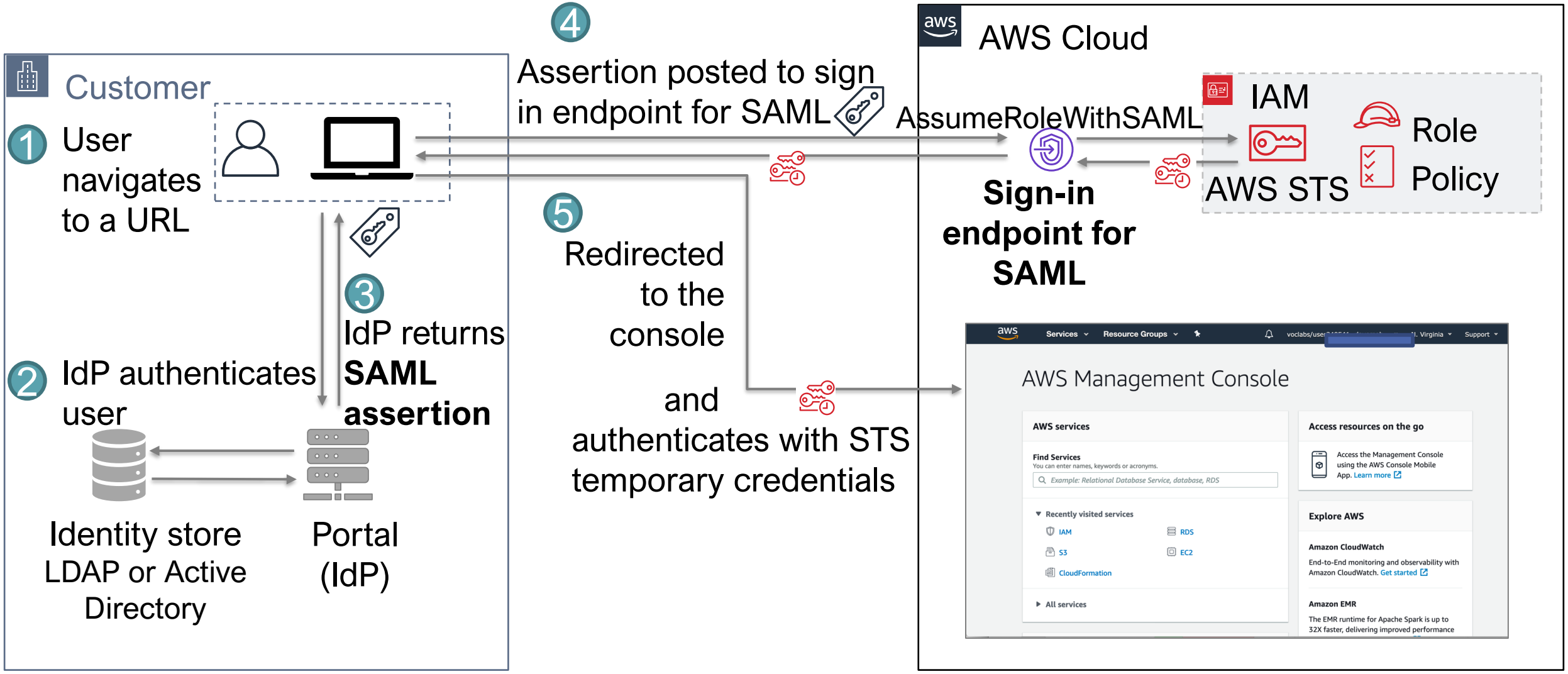
Temporary credentials returned to application

IdP authentication

Identity Federation with Broker



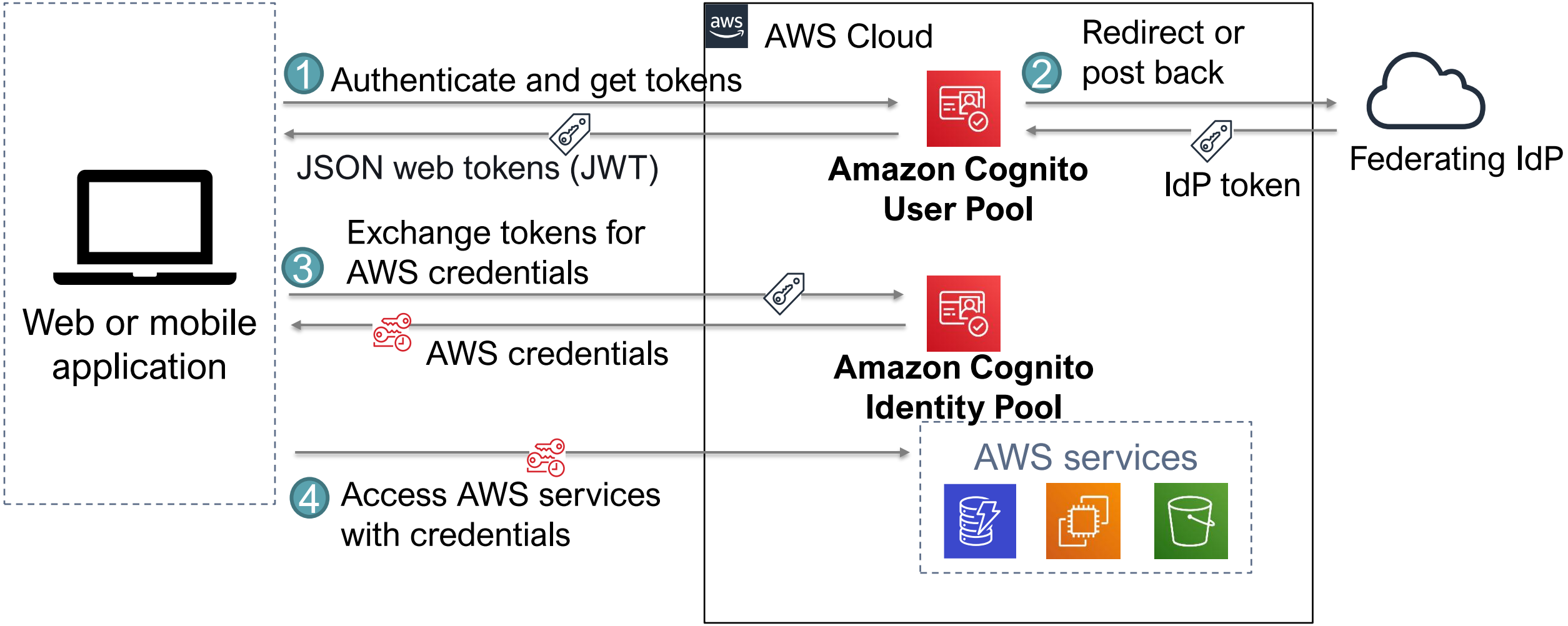
Identity Federation Using SAML



Amazon Cognito

- **Fully managed** service
- **Web** identity federation
- Provides **authentication**, **authorization**, and **user management** for web and mobile applications
  - Used as **identity broker** to support IdPs compatible with **OpenID Connect (OIDC)**
  - Users sign in with **social identity providers** (e.g. Meta, Google), **SAML**, or user pool
- **User pools**
  - You can maintain a directory with **user profiles** authentication tokens
  - Supports **MFA**

Amazon Cognito Example



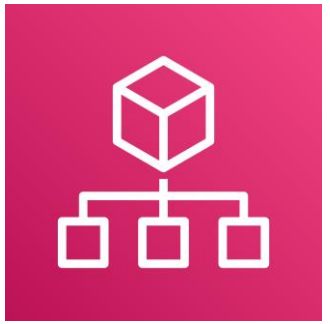
AWS Shield

- Managed distributed **DDoS** protection
  - UDP floods, TCP SYN floods, HTTP GET and POST floods
- Safeguards **applications** running on AWS
  - e.g. EC2 applications, ELB, CloudFront, Route 53
- Always-on **detection** and automatic **mitigations**
- Shield **Standard** enabled for at no additional cost
- Shield **Advanced** is an optional paid service
  - More sophisticated attacks
- Use to minimize application **downtime** and **latency**

Securing AWS Accounts

AWS Organizations

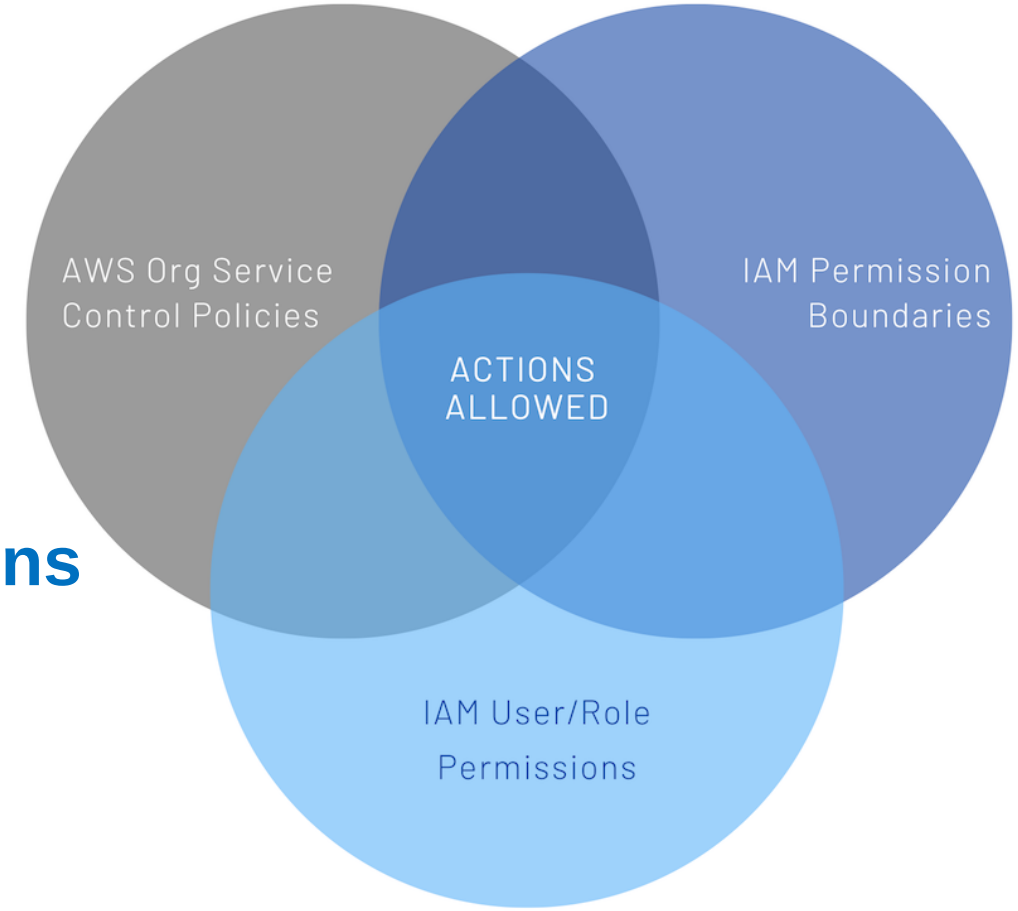
- **Consolidate** multiple AWS accounts to centrally manage them
- **Security features**
  - Group AWS accounts into **organizational units (OUs)** and attach different **access policies** to each OU
  - **Integration** and support for **IAM**
    - Permissions to a user are the **intersection** of what is allowed by AWS Organizations and what is allowed by IAM
  - Use **service control policies (SCPs)** to control the AWS services and API actions each AWS account can access



AWS Organizations

SCPs

- **Centralized** control over accounts
  - **Limit** permissions available in an account that in organization
- Ensures accounts **comply** with access control guidelines
- **Similar to IAM** policy syntax
  - SCP **never grants** permissions
  - SCPs specify the **maximum permissions** for an organization
  - **Not substitute** to IAM

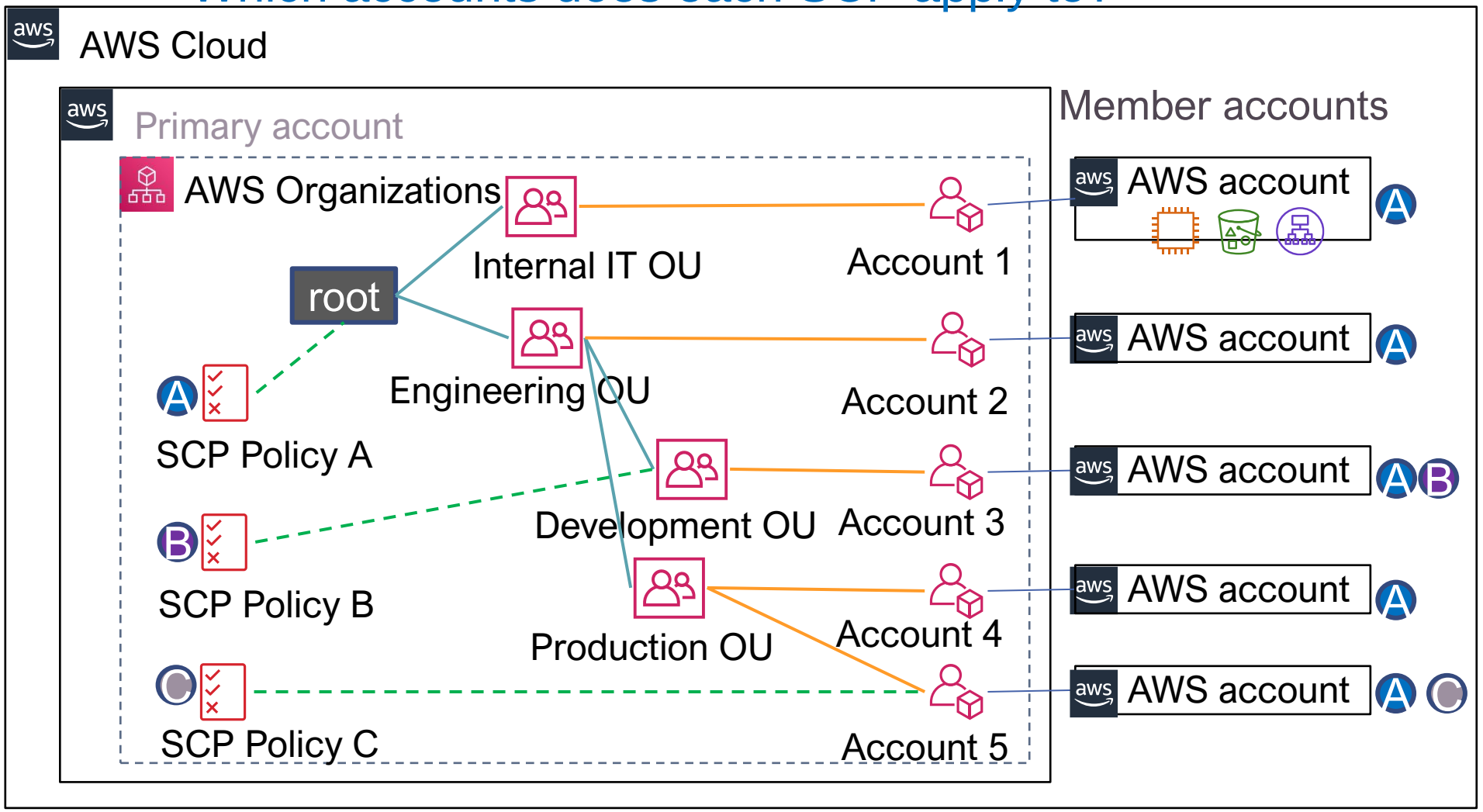


AWS Organizations Illustrated

Which accounts does each SCP apply to?

In AWS Organizations **primary account**:

1. Create a **hierarchy** of OUs
2. **Assign** accounts to OUs as member accounts
3. Define **SCPs** to apply permissions restrictions
4. Attach the SPCs to root, OUs or accounts



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "config:DeleteConfigRule",
        "config:DeleteConfigurationRecorder",
        "config:DeleteDeliveryChannel",
        "config:StopConfigurationRecorder"
      ],
      "Resource": "*"
    }
  ]
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictEC2ForRoot",
      "Effect": "Deny",
      "Action": [
        "ec2:*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::*:root"
          ]
        }
      }
    }
  ]
}
```

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

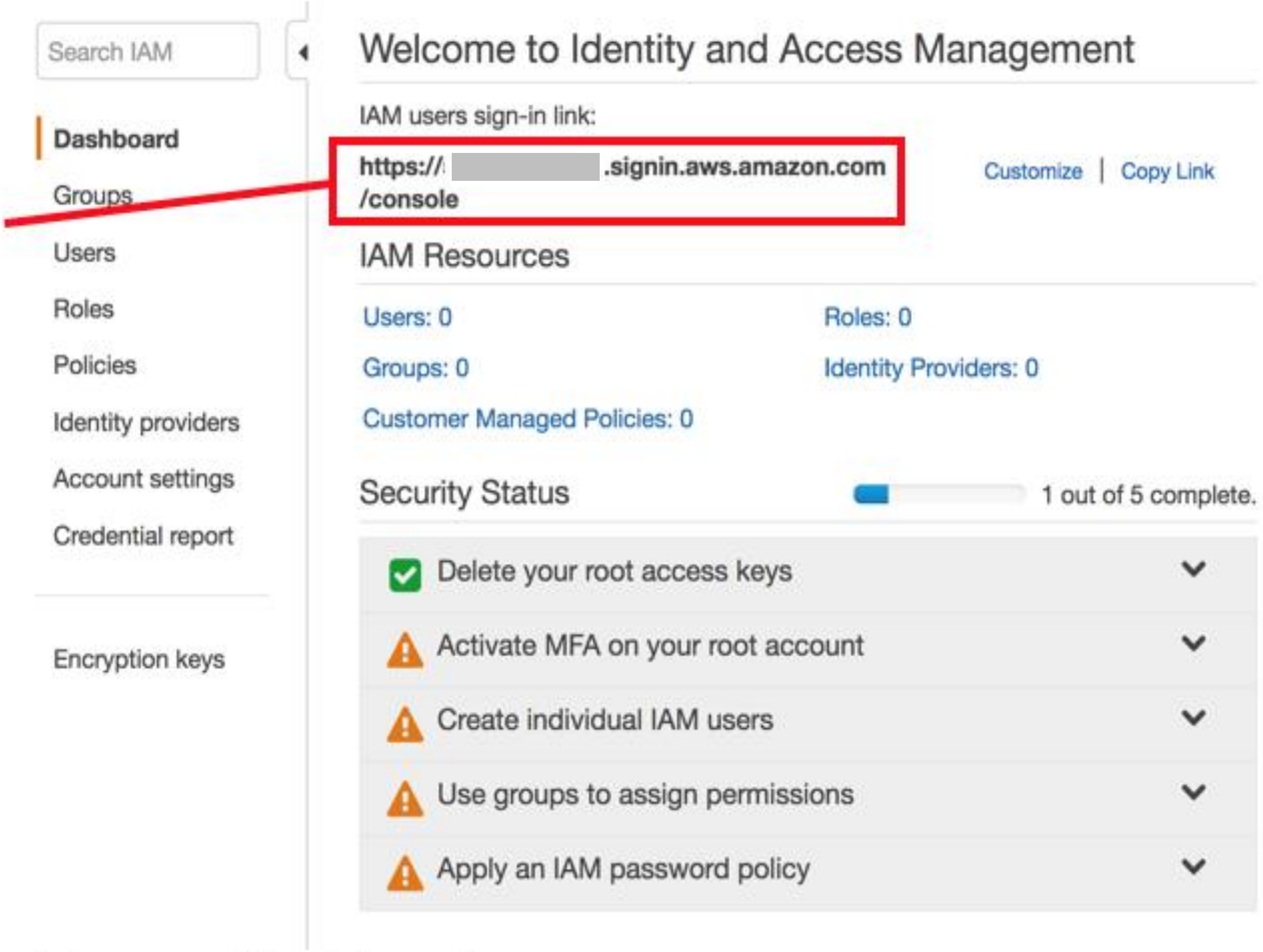
AWS Key Management Service (AWS KMS)

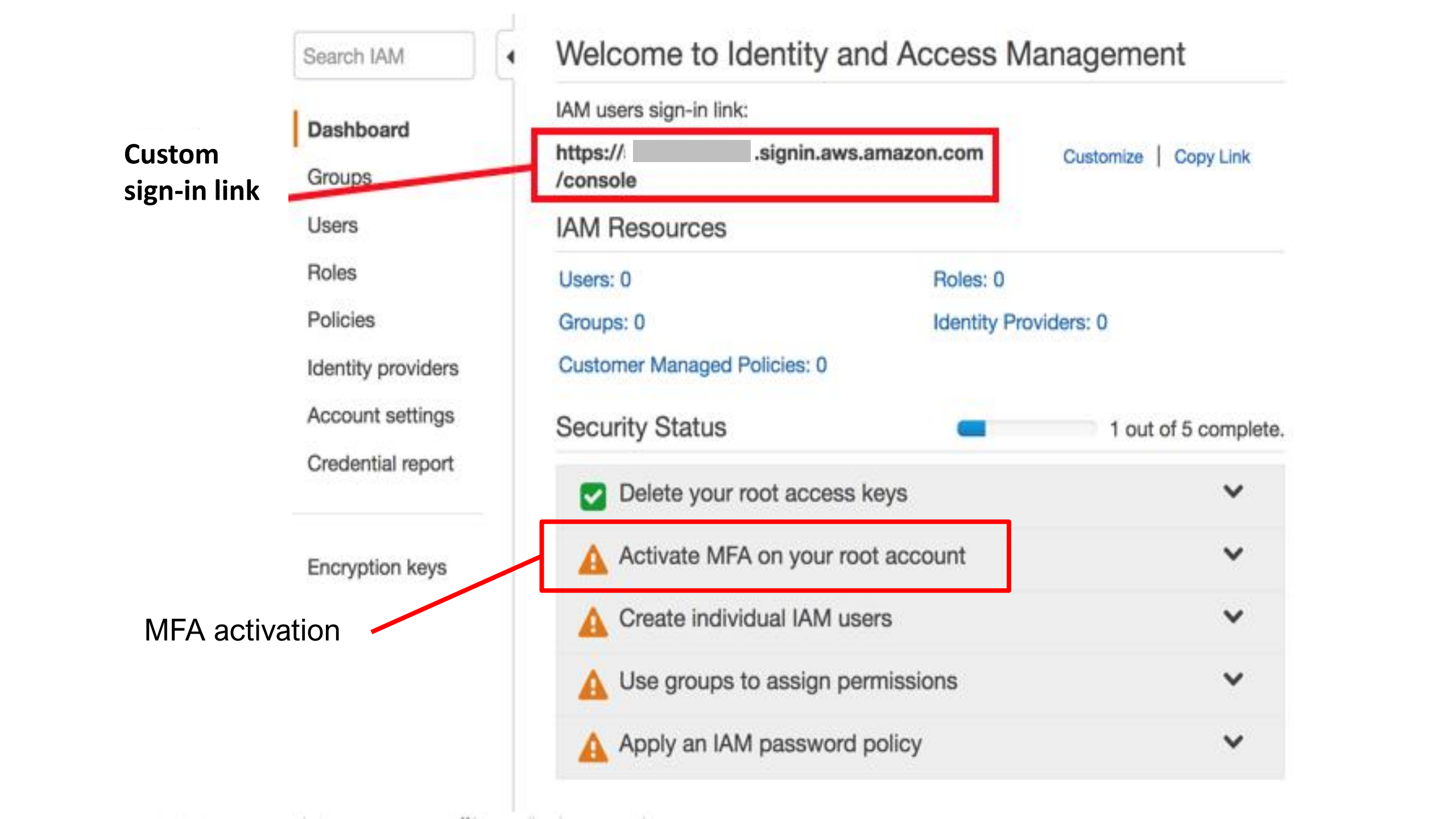
- Enables customers to **create and manage encryption keys**
  - Symmetric and asymmetric
- Allows customers to control the use of **encryption across AWS services** and in **applications**
- Integrates with AWS **CloudTrail** to log all key usage
- Uses **hardware security modules (HSMs)** validated by FIPS (Federal Information Processing Standards) 140-2 to **protect keys**
- **Customer master keys (CMKs)** control access to data encryption keys
  - **Customer-managed** (more control) or AWS-managed CMK

Securing New AWS Accounts

Custom Sign In Link

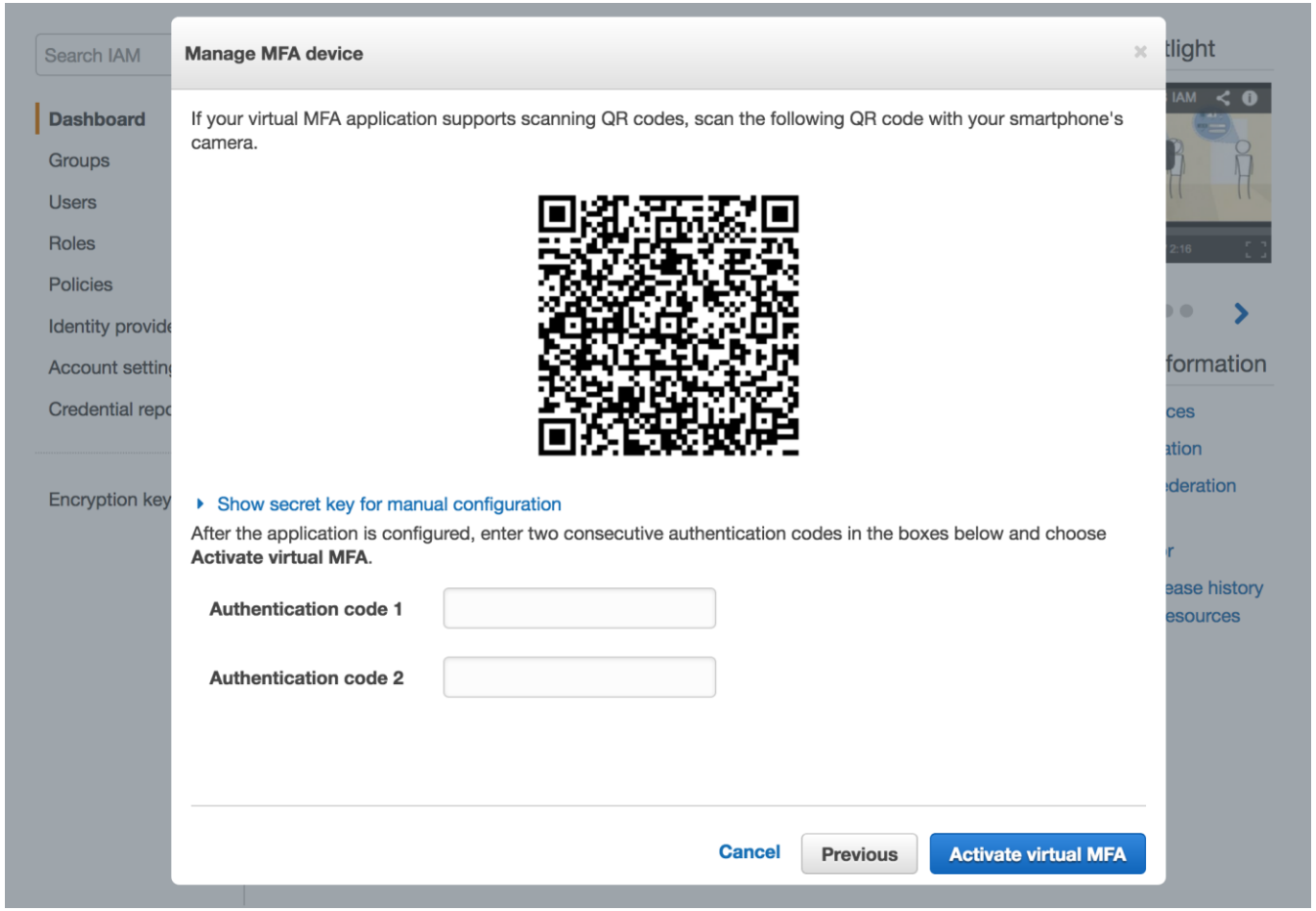
Send to users after their accounts created

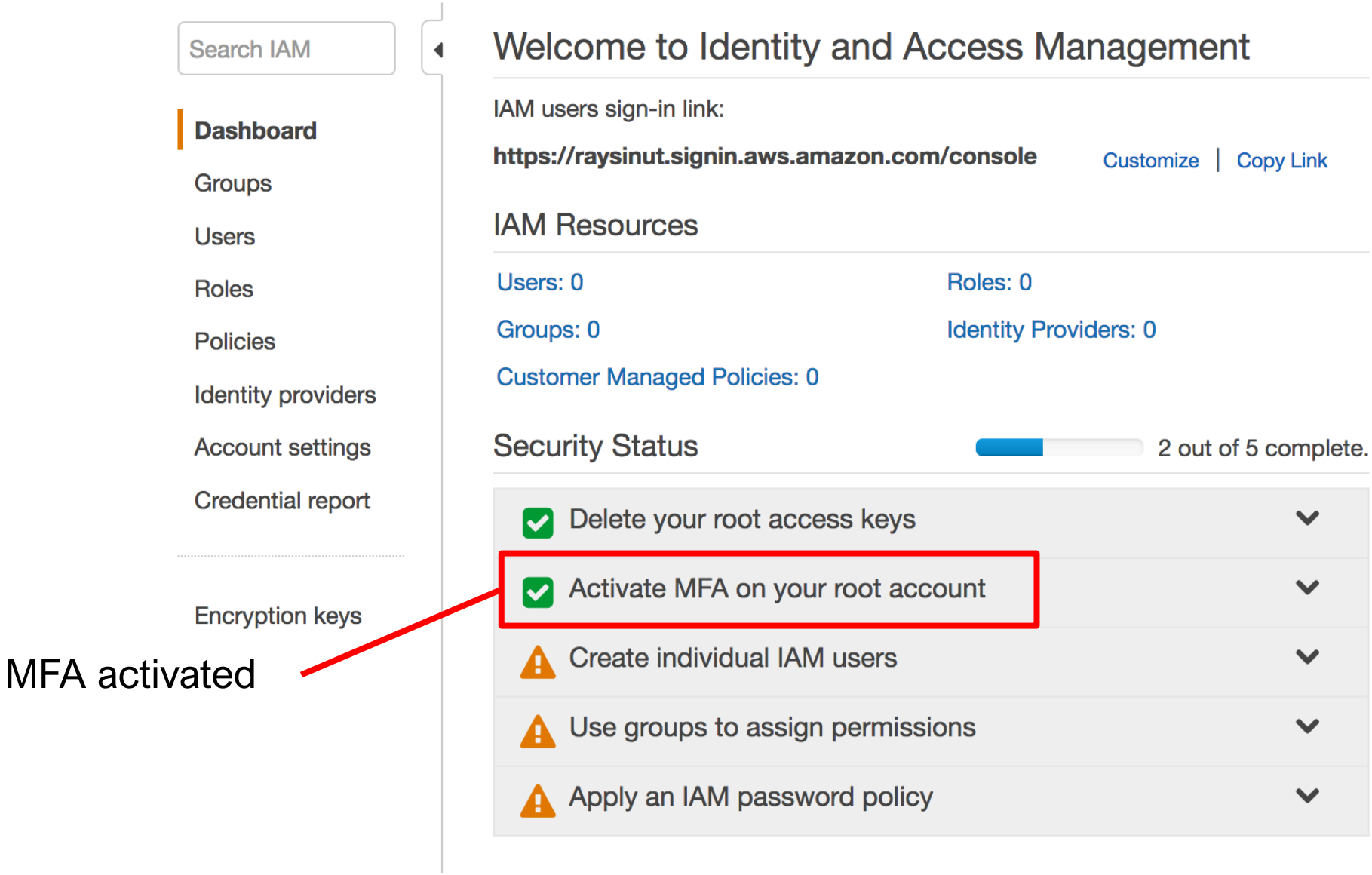



Custom sign-in link
Search IAM
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Encryption keys
Welcome to Identity and Access Management
IAM users sign-in link:
https:// .signin.aws.amazon.com /console
Customize | Copy Link
IAM Resources
Users: 0
Roles: 0
Groups: 0
Identity Providers: 0
Customer Managed Policies: 0
Security Status
1 out of 5 complete.
Delete your root access keys
Activate MFA on your root account
Create individual IAM users
Use groups to assign permissions
Apply an IAM password policy
MFA activation

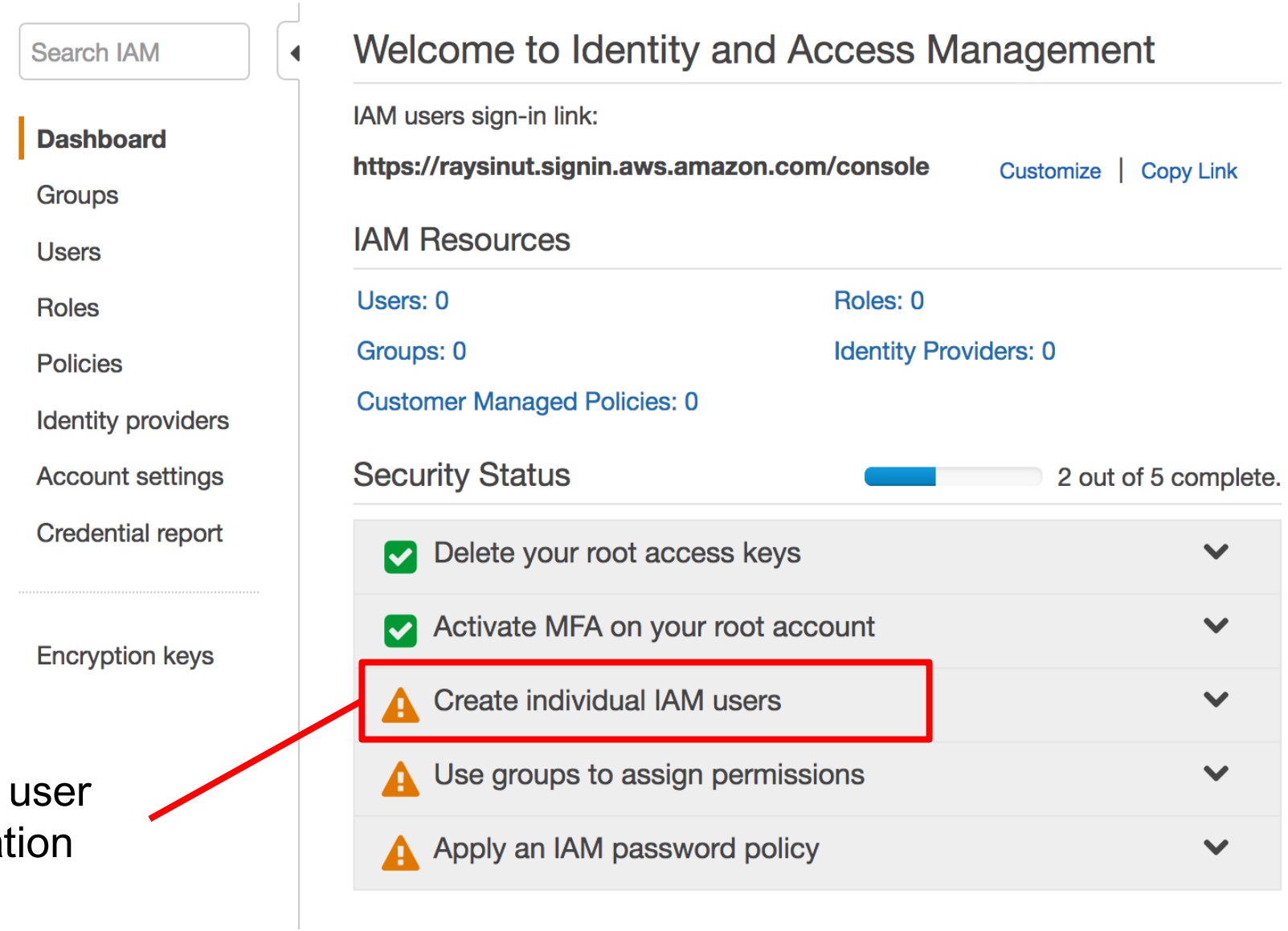
Activate MFA on Account Root User

1. **Activate MFA on your root account** link
2. **Manage MFA**
3. **Assign MFA device**
   Three options: **Virtual MFA device**, **U2F security key**, **Other hardware MFA device**
4. **Virtual MFA device** and **Continue**
5. Dialog box to configure a virtual MFA device
   An **app (e.g. Google Authenticator)** must be downloaded
6. **Show QR code**
7. Scan QR and enter authentication code 1, 2
8. **Assign MFA**




Search IAM
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Encryption keys
Welcome to Identity and Access Management
IAM users sign-in link:
https://raysinut.signin.aws.amazon.com/console
Customize | Copy Link
IAM Resources
Users: 0
Roles: 0
Groups: 0
Identity Providers: 0
Customer Managed Policies: 0
Security Status
2 out of 5 complete.
Delete your root access keys
Activate MFA on your root account
Create individual IAM users
Use groups to assign permissions
Apply an IAM password policy
MFA activated

Create IAM Users



IAM user creation

Add user

1 Details — 2 Permissions — 3 Review — 4 Complete

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* Mic

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* ☑ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
☑ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ◉ Autogenerated password
○ Custom password

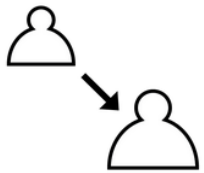
Require password reset ☑ User must create a new password at next sign-in
Users automatically get the IAMUserChangePassword policy to allow them to change their own password.

Add user

1 Details — 2 **Permissions** — 3 Review — 4 Complete

Set permissions for M

Add user to group



Copy permissions from existing user

Attach existing policies directly

Get started with groups

You haven't created any groups yet. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Get started by creating a group. Learn more

Create group

Cancel | Previous | Next: Review

Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Learn more

Group name Administrators

Create policy | Refresh

Filter: Policy type | Search | Showing 313 results

	Policy name	Type	Attachments	Description
☑	AdministratorAccess	Job function	0	Provides full access to AWS services and resources.
☐	AlexaForBusinessDeviceSetup	AWS managed	0	Provide device setup access to AlexaForBusiness services
☐	AlexaForBusinessFullAccess	AWS managed	0	Grants full access to AlexaForBusiness resources and access to relat…
☐	AlexaForBusinessGatewayEx…	AWS managed	0	Provide gateway execution access to AlexaForBusiness services
☐	AlexaForBusinessReadOnlyA…	AWS managed	0	Provide read only access to AlexaForBusiness services
☐	AmazonAPIGatewayAdminist…	AWS managed	0	Provides full access to create/edit/delete APIs in Amazon API Gatew…
☐	AmazonAPIGatewayInvokeFu…	AWS managed	0	Provides full access to invoke APIs in Amazon API Gateway.
☐	AmazonAPIGatewayPushToC…	AWS managed	0	Allows API Gateway to push logs to user's account.
☐	AmazonAppStreamFullAccess	AWS managed	0	Provides full access to Amazon AppStream via the AWS Managemen…
☐	AmazonAppStreamReadOnly…	AWS managed	0	Provides read only access to Amazon AppStream via the AWS Mana…

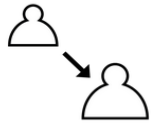
Cancel | Create group

Add user

1 Details — 2 Permissions — 3 Review — 4 Complete

Set permissions for M

Add user to group



Copy permissions from existing user

Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Create group | Refresh

Search — Showing 1 result

Group	Attached policies
Administrators	AdministratorAccess

Cancel | Previous | Next: Review

Add user

1 Details — 2 Permissions — 3 Review — 4 **Complete**

Success

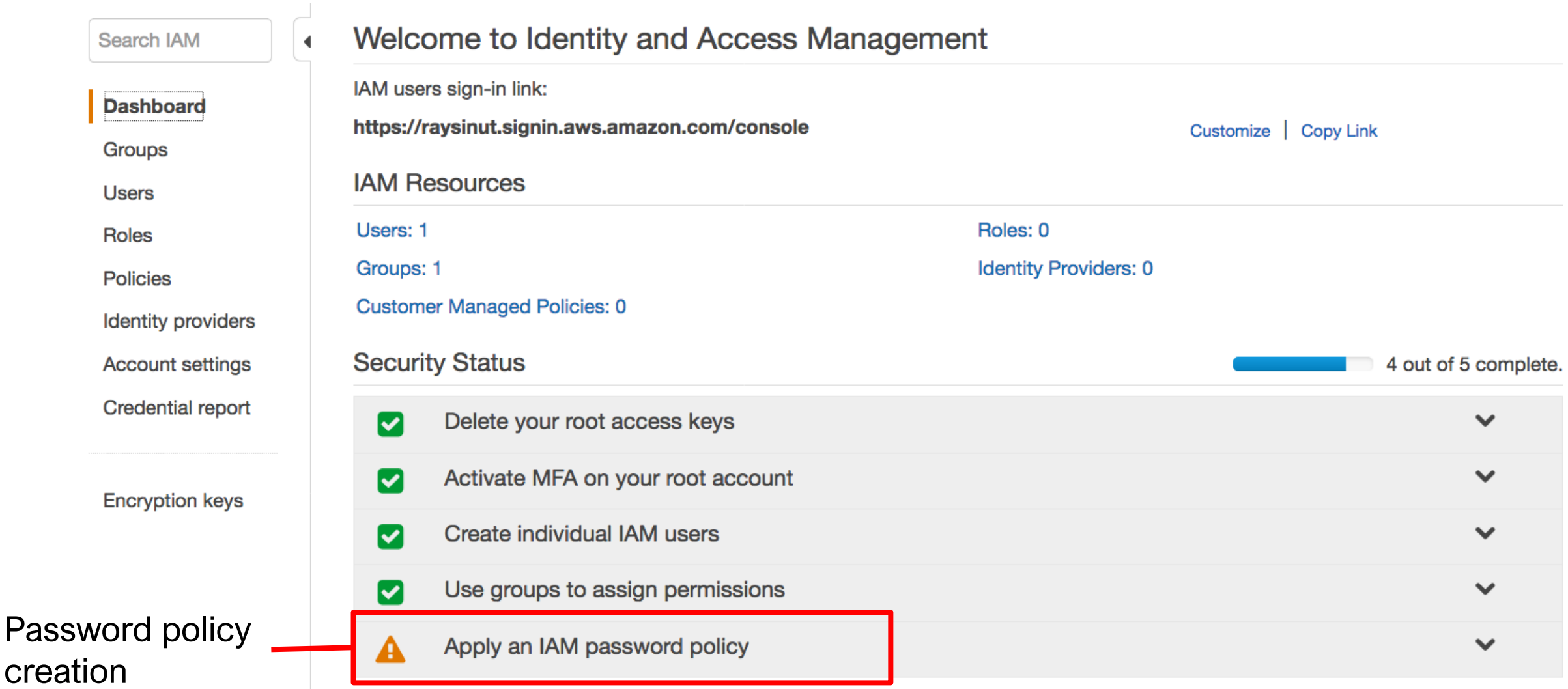
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://raysinut.signin.aws.amazon.com/console

Download .csv

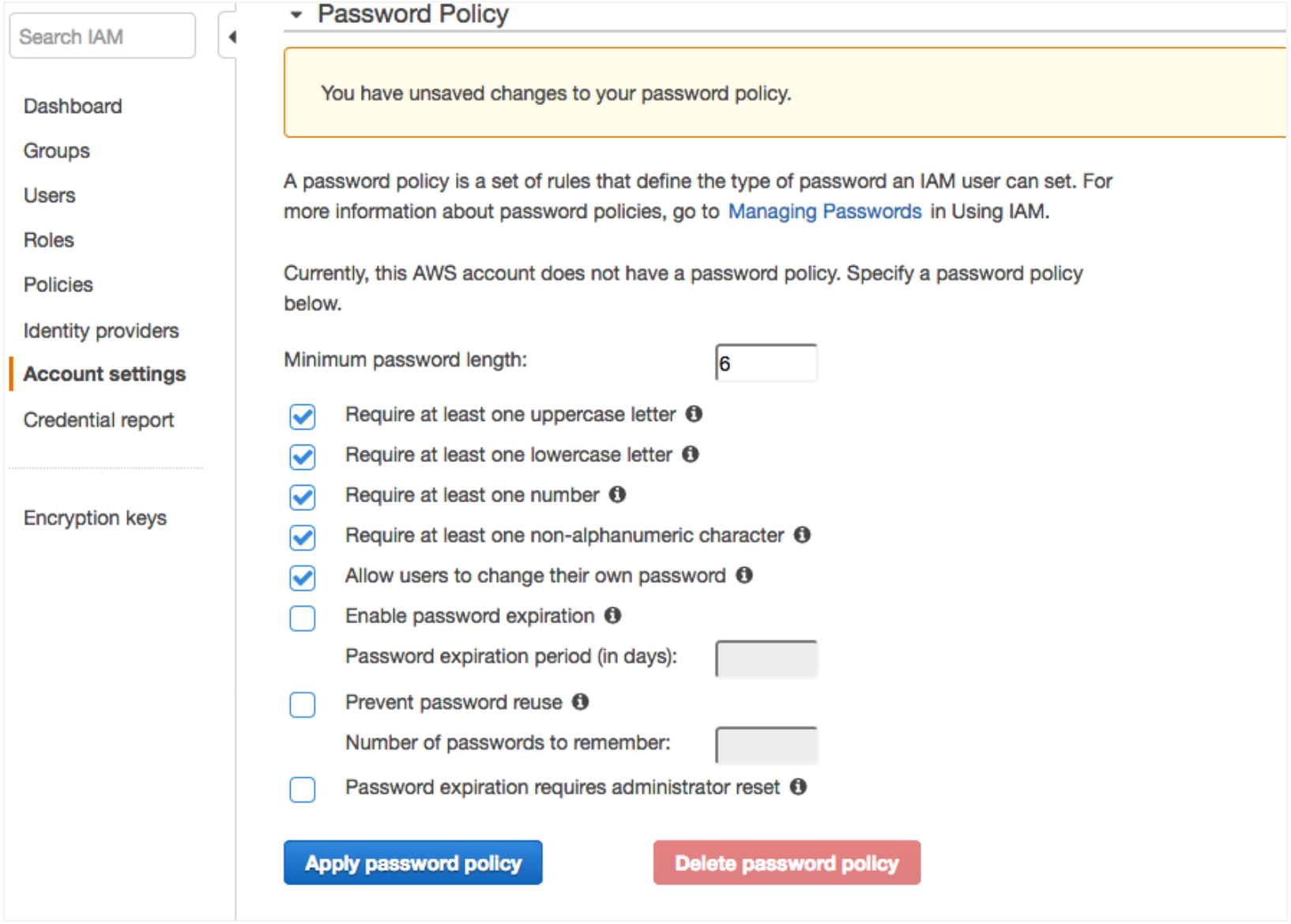
		User	Access key ID	Secret access key	Password	Email login instructions
▸	✓	Mi	AKI	********* Show	********* Show	Send email

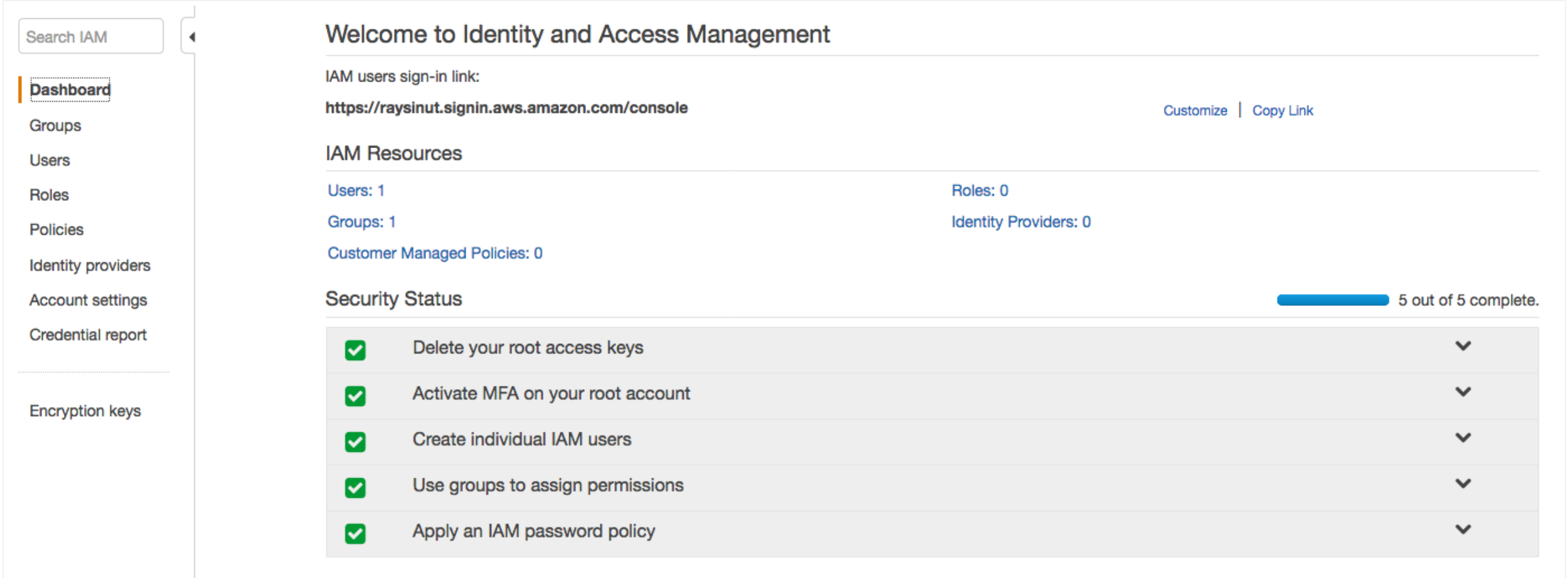
Close



Password policy creation

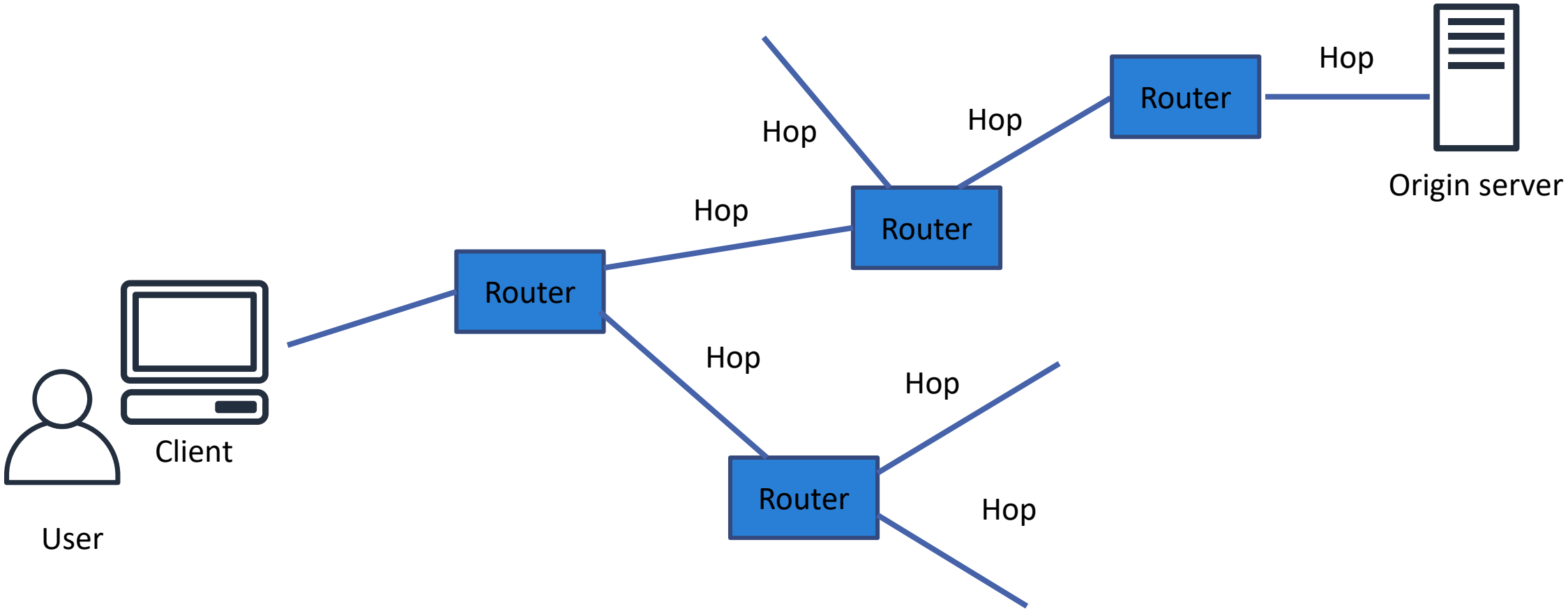
IAM Password Policy




Search IAM
Dashboard
Groups
Users
Roles
Policies
Identity providers
Account settings
Credential report
Encryption keys
Welcome to Identity and Access Management
IAM users sign-in link:
https://raysinut.signin.aws.amazon.com/console
Customize | Copy Link
IAM Resources
Users: 1
Roles: 0
Groups: 1
Identity Providers: 0
Customer Managed Policies: 0
Security Status
5 out of 5 complete.
Delete your root access keys
Activate MFA on your root account
Create individual IAM users
Use groups to assign permissions
Apply an IAM password policy

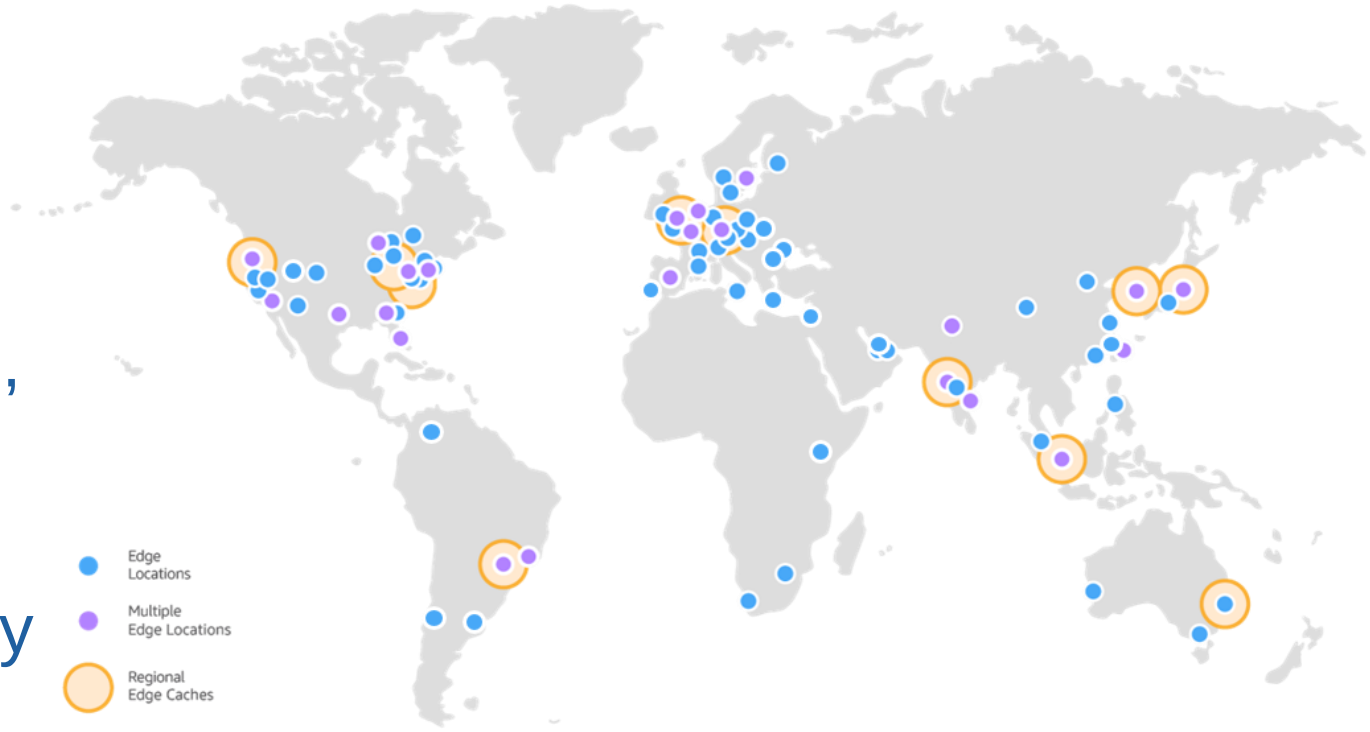
Edge Caching

Network Latency



Content Delivery Network (CDN)

- Globally distributed system of **caching servers**
- Caches copies of **commonly requested files**
  - Static content e.g. html, CSS, JavaScript, video, …
- Delivers a local copy of the requested content from a nearby **edge location**
- Improves application **performance**



Amazon CloudFront

- Amazon’s global CDN
- Optimized for various **delivery use cases**
- **Multi-tier** cache
- Helps **secure** the content and to protect the **underlying source**
  - Move traffic **off the Internet**
  - DDoS and other attack mitigation with **AWS Shield** and **AWS Web Application Firewall (WAF)**
  - Integration with **AWS Certificate Manager**
- Supports WebSockets, HTTP and HTTPS methods

AWS WAF

AWS Shield

AWS ACM

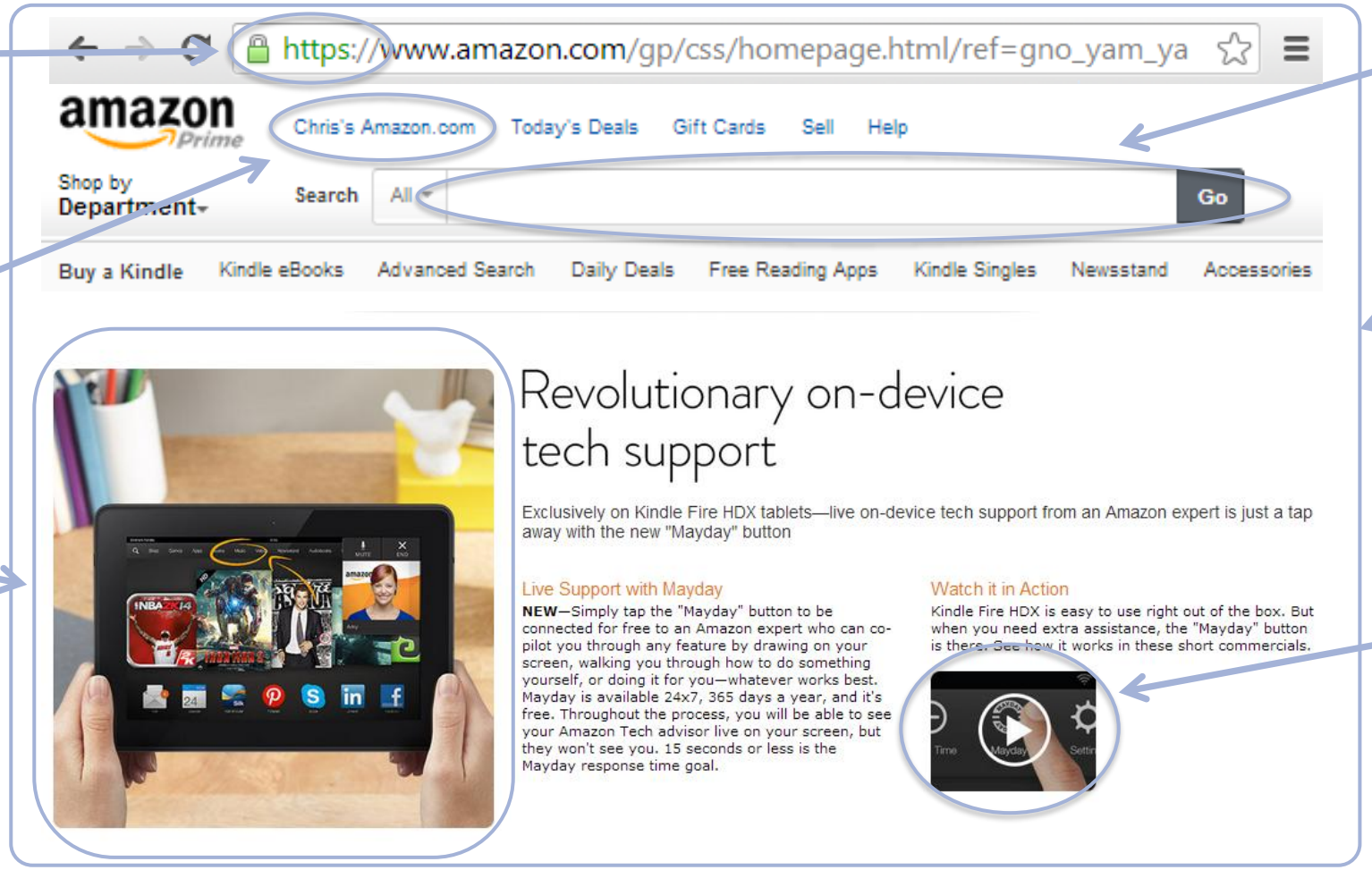
What type of content can be cached in an edge cache?

Secure

Dynamic

Image

Can be cached!



User input

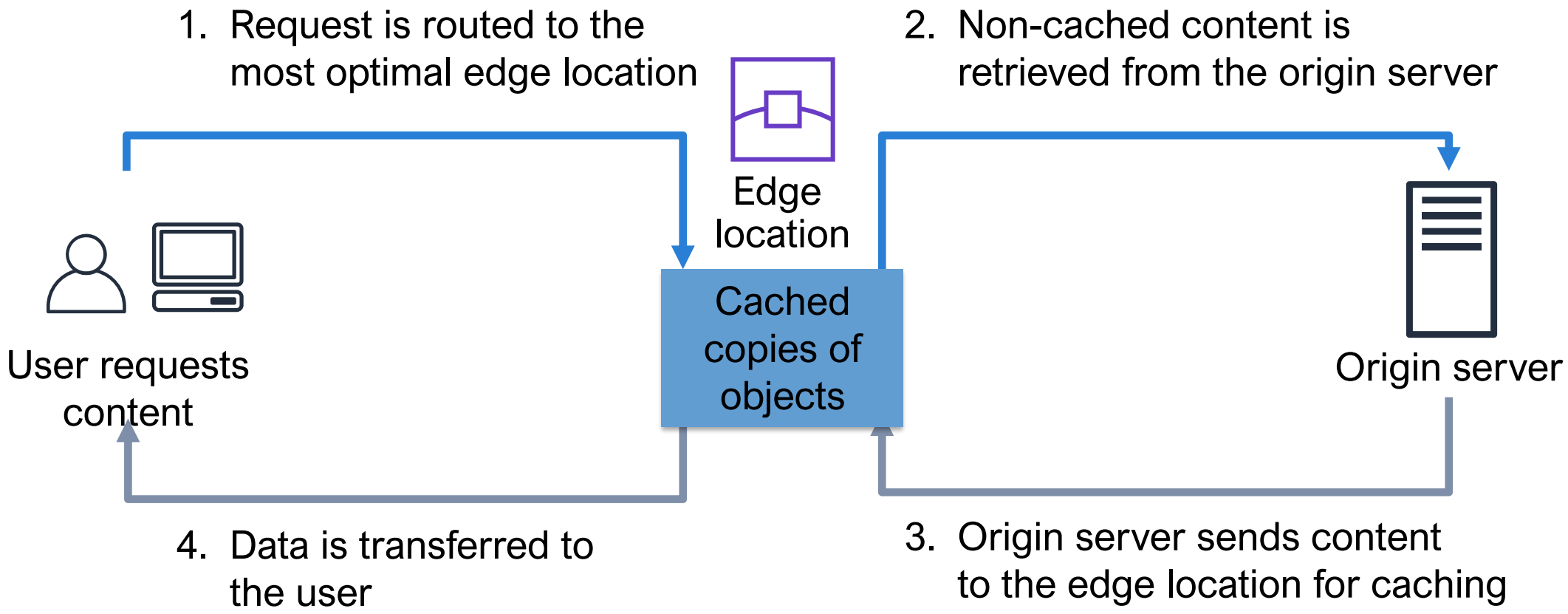
Web objects

Can be cached!

Video

Can be cached!

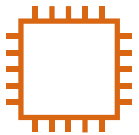
How Caching Works in CloudFront



How to Configure a CloudFront Distribution

1. You specify an origin server

S3 bucket



EC2 instance

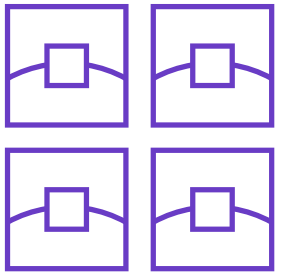
Web server or other custom origin

2. You configure the distribution

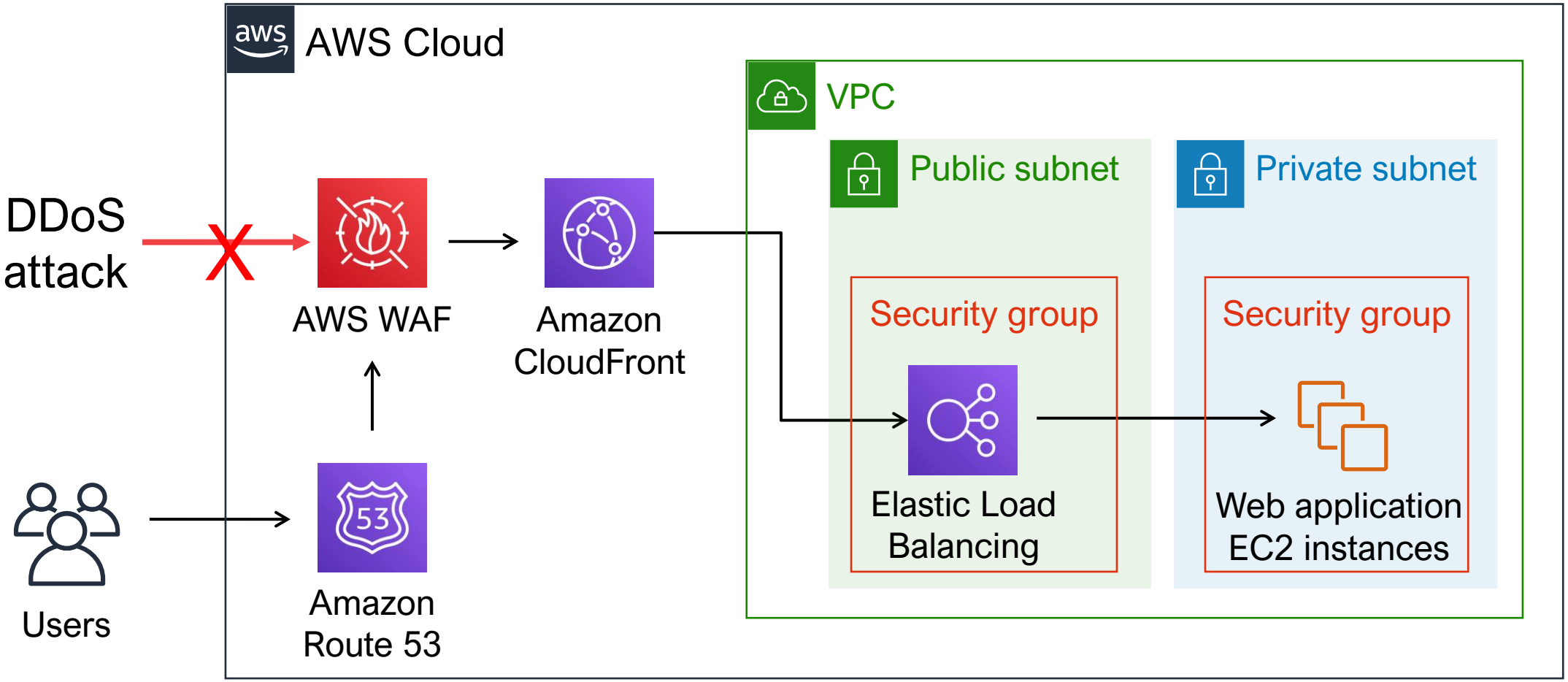
- Access
- Security
- Cookie or query-string forwarding
- Geo-restrictions
- Access logs

3. CloudFront assigns a **domain name**

4. CloudFront sends distribution's **configuration** to edge locations



Example DDoS Mitigation

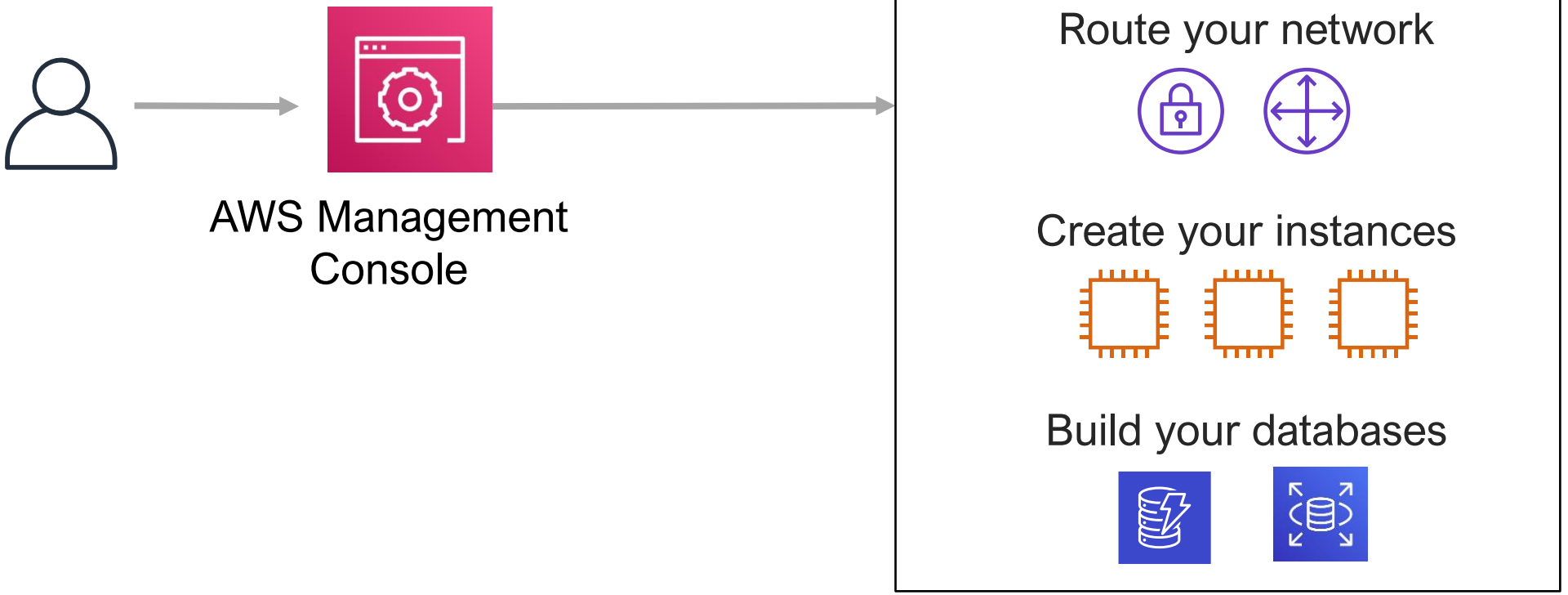


Always-on monitoring, anomaly detection and mitigation against common infrastructure DDoS attacks are built into Route 53 and CloudFront

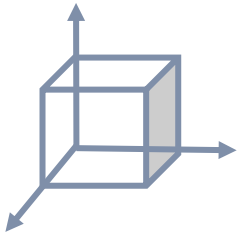
Automating Architecture

Without Automation

Long manual process to build an architecture



Risks from Manual Processes



Does not support **repeatability** at scale

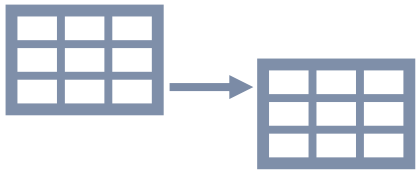
- How will you replicate deployments to multiple Regions?

No **version control**

- How will you roll back the production environment to a prior version?

Lack of **audit** trails

- How will you ensure compliance? How will you track configurations changes?



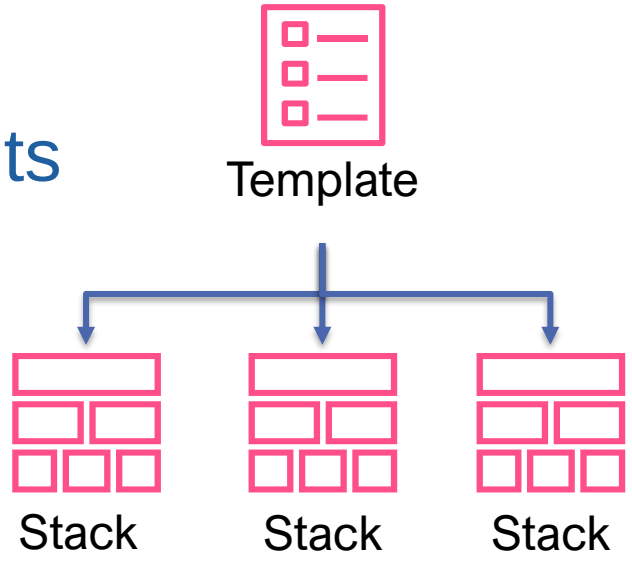
Inconsistent data management

- e.g. how will you ensure matching configurations across EC2 instances?

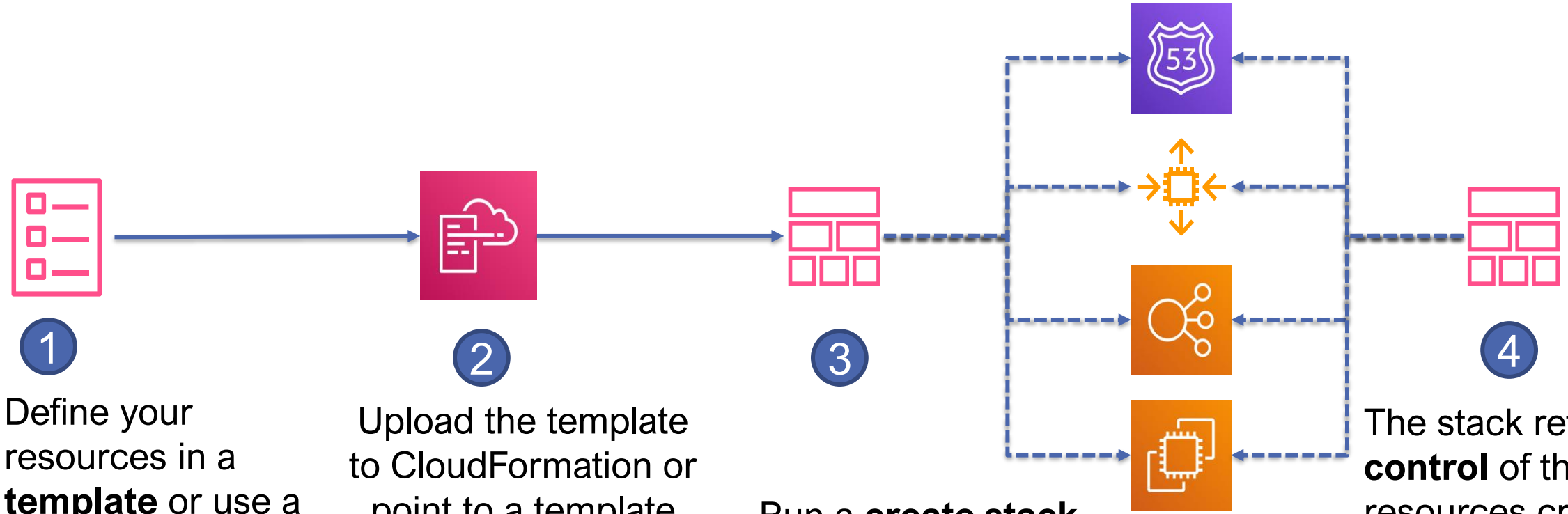
Automating Infrastructure

- CloudFormation provides a simplified way to **model**, **create** and **manage** a collection of AWS resources
  - **CloudFormation stack**
- Create, update and delete stacks
- Enables **orderly** and **predictable** provisioning and updating of resources
- Enables **version control** of resource deployments
  - **IaC**
  - **Rollback** capability

AWS CloudFormation



AWS CloudFormation



1

Define your resources in a **template** or use a pre-built template

2

Upload the template to CloudFormation or point to a template stored in S3

3

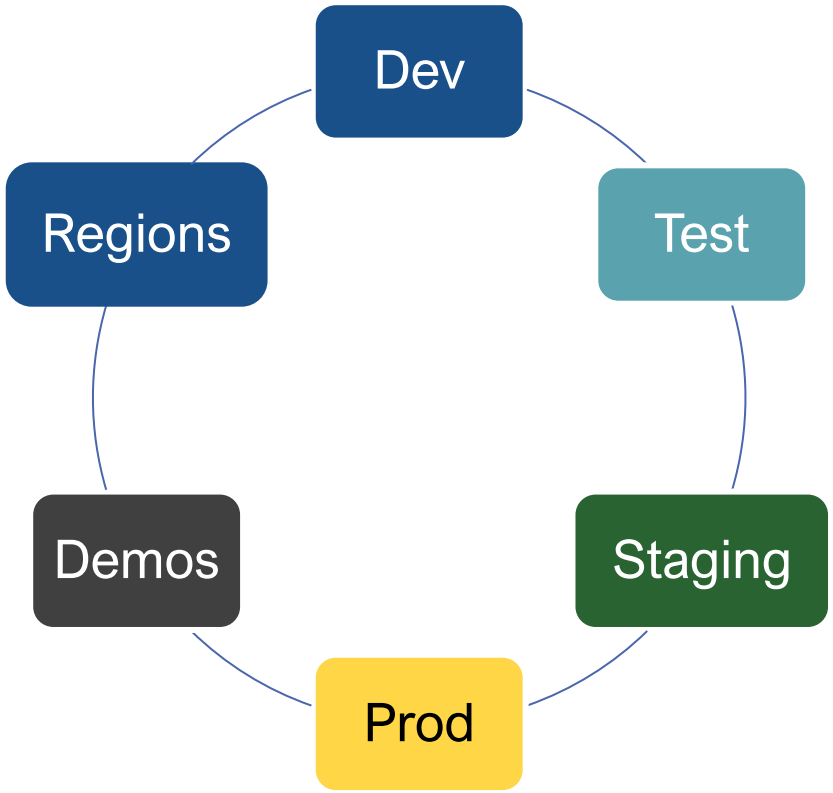
Run a **create stack**
Resources created across multiple services in the AWS account as a running environment

4

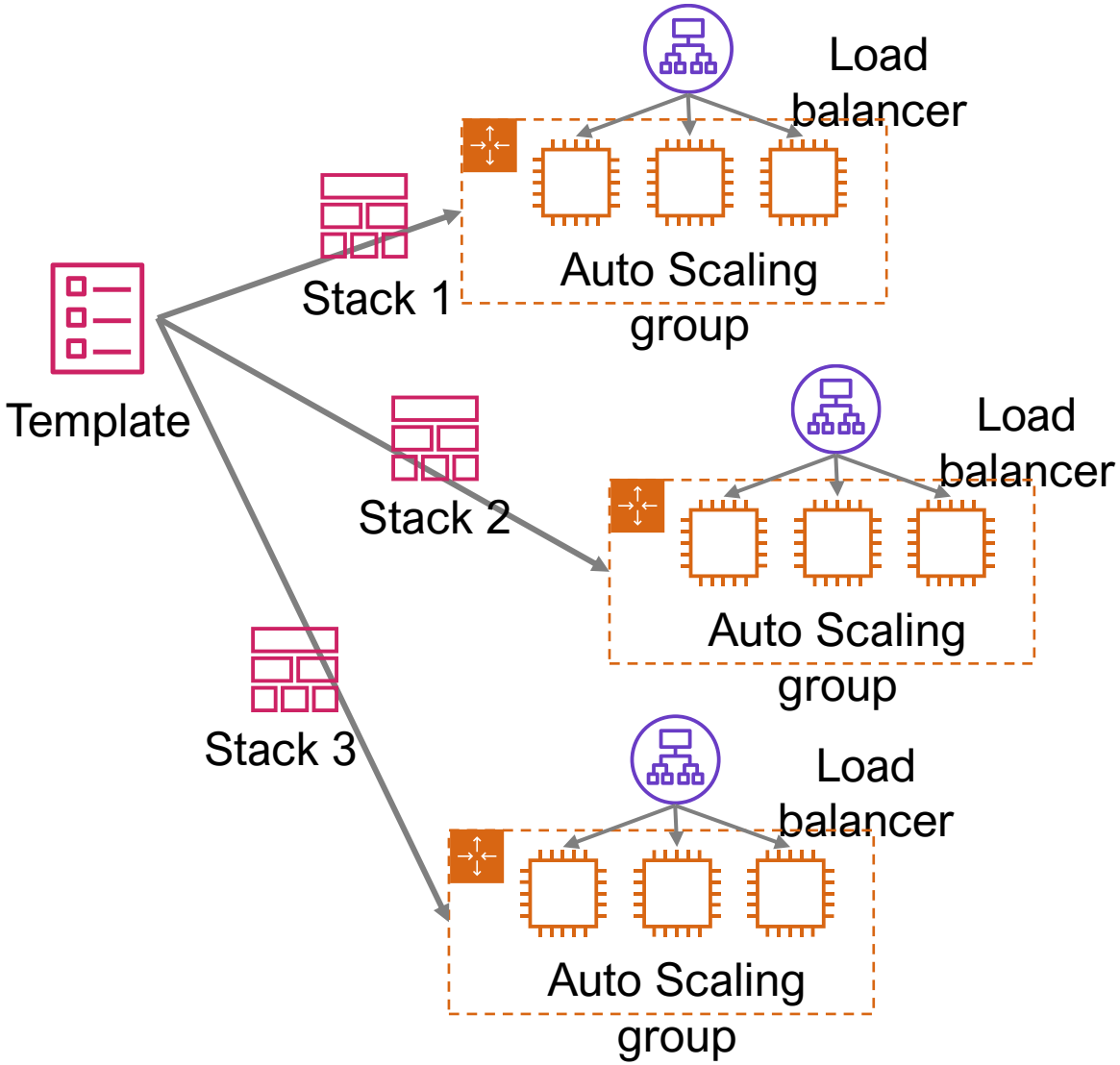
The stack retains **control** of the resources created
You can later **update stack**, **detect drift** or **delete stack**

IaC

- Provisioning and managing cloud resources by writing a **template file**
  - Human readable
  - Machine consumable
- Replicate, re-deploy, **re-purpose** infrastructure
  - **Easily, reliably and consistently**
- You can roll back to the **last good state** on failures
  - **Transactional**



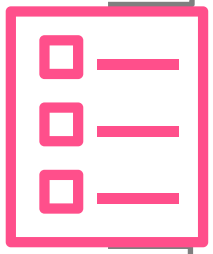
Infrastructure as Code Benefits



- Rapid deployment of multiple **complex** environments
- **Configuration** consistency
- **Propagate** a change to all stacks
  - Modify the template, run update stack on all stacks
  - Repeatability and maintainability
- **Maintain security** standards and best practices and facilitates **security testing**

AWS CloudFormation Template Syntax

- **JSON** or **YAML**
- YAML advantages
  - Less verbose (no {},"")
  - Embedded comments
  - Easier to debug
- JSON advantages
  - More widely used
- Treat templates as source code
  - Store in a code repository



```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Resources" : {
 “awsexamplebucket1" : {
 "Type" : "AWS::S3::Bucket"
 }
 }
}
```

JSON example

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
 awsexamplebucket1:
 Type: AWS::S3::Bucket
```

YAML example

Templates can also be authored in CloudFormation Designer GUI in Management Console

Simple Template: Create an EC2 Instance

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Description": "Create EC2 instance",
 "Parameters": {
 "KeyPair": {
 "Description": "SSH Key Pair",
 "Type": "String"}},
 "Resources": {
 "Ec2Instance": {
 "Type": "AWS::EC2::Instance",
 "Properties": {
 "ImageId": "ami-9d23aeea",
 "InstanceType": "m3.medium",
 "KeyName": {"Ref": "KeyPair}}},
 "Outputs": {
 "InstanceId": {
 "Description": "InstanceId",
 "Value": {"Ref": "Ec2Instance"}
 }
 }
}
```

Parameters – what values can be **set at runtime** when creating the stack e.g. region-specific settings, production versus test environment settings

Resources – what needs to be created e.g. create all VPC components in a region and then create EC2 instances in VPC

Can **reference** parameters

Outputs – values returned after the stack is created e.g. return instanceId, public IP address of EC2 instance

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. Learn more.

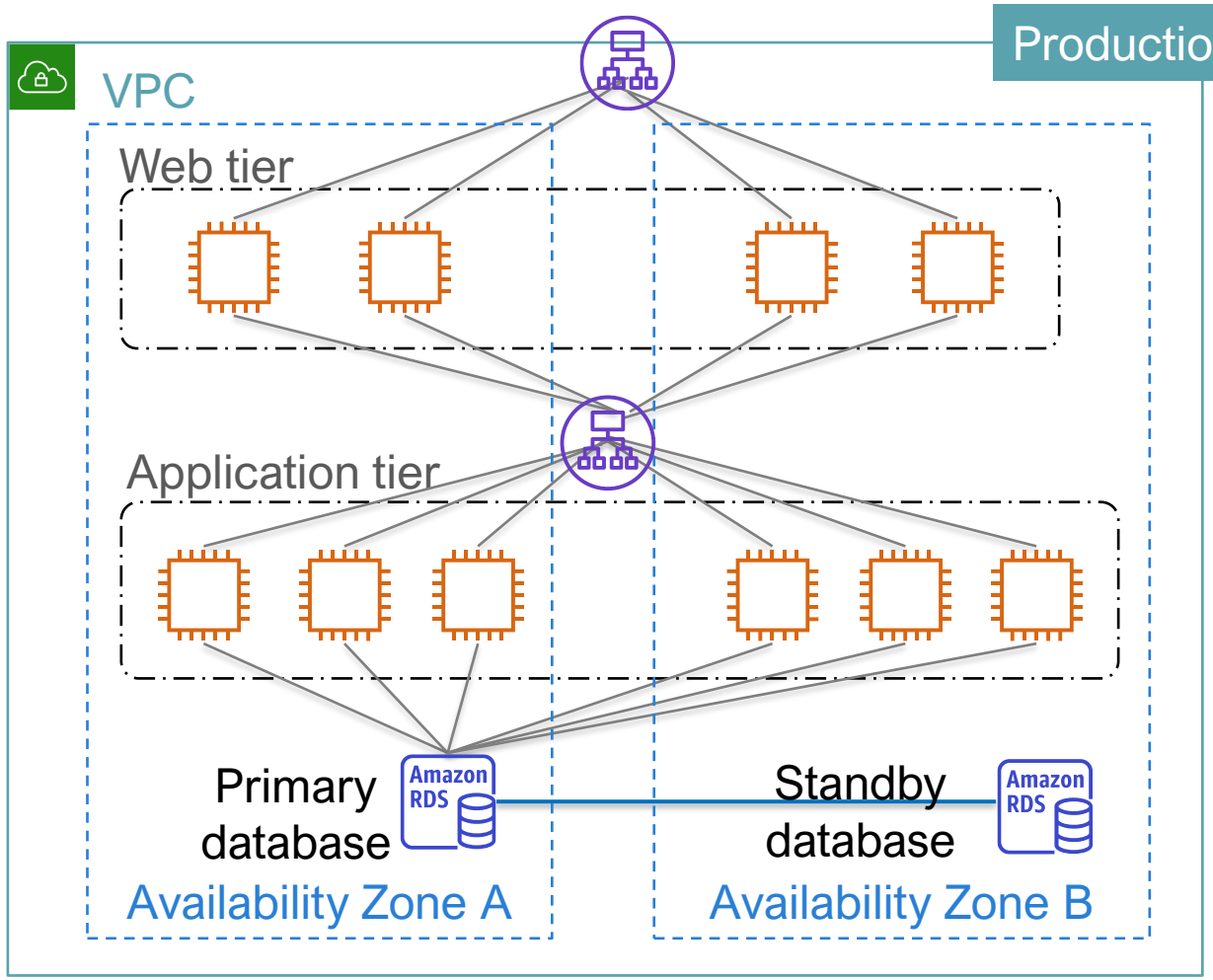
Stack name

Parameters

DBName	MyDatabase	MySQL database name
DBPassword		Password for MySQL database access
DBRootPassword		Root password for MySQL
DBUser		Username for MySQL database access
InstanceType	t2.small	WebServer EC2 instance type
KeyName	Search	Name of an existing EC2 KeyPair to enable SSH access to the instance
SSHLocation	0.0.0.0/0	The IP address range that can be used to SSH to the EC2 instances

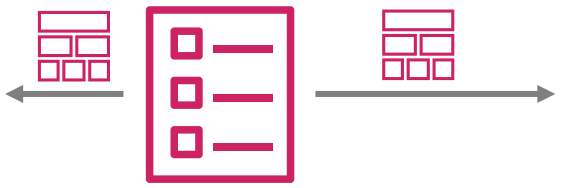
Cancel Previous Next

CloudFormation Conditions

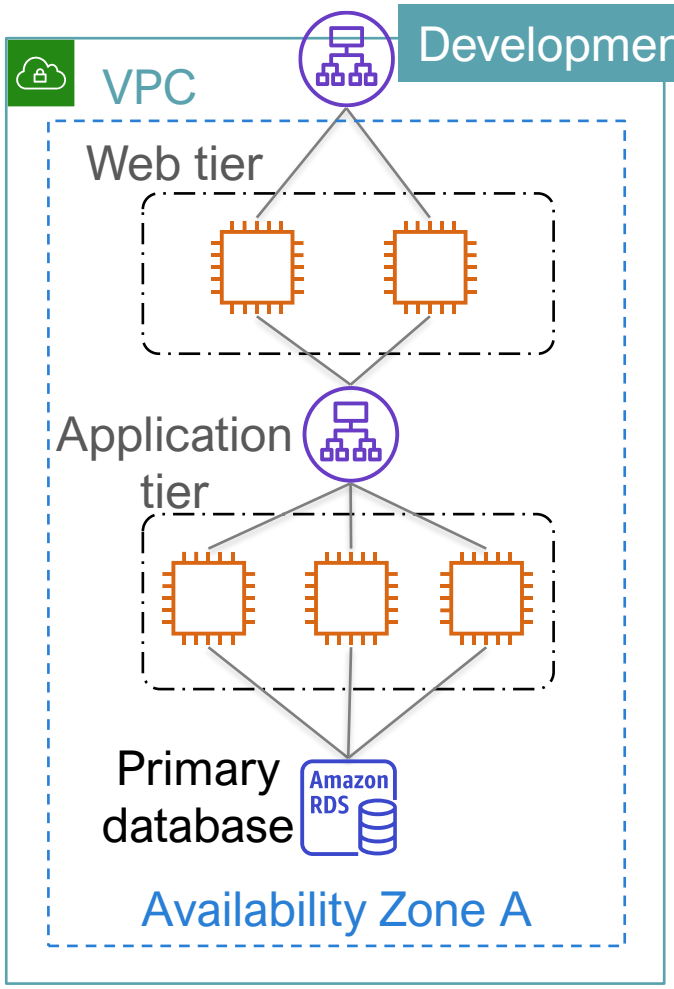


Need to update a configuration?

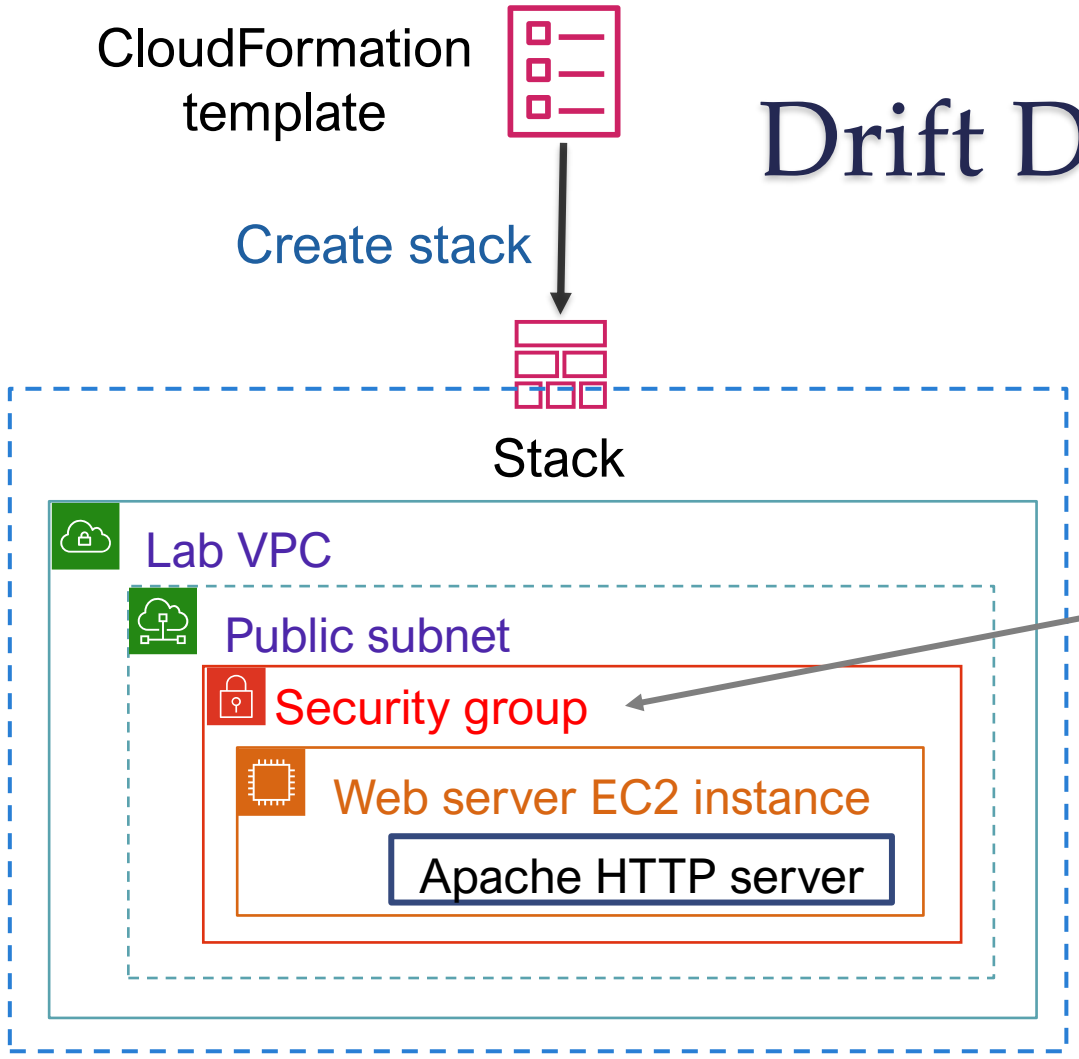
Edit the template details, then update the stacks



A single CloudFormation template with defined **conditions** can deploy a different application infrastructure to different environments



Drift Detection

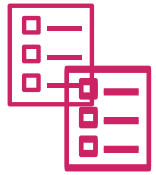
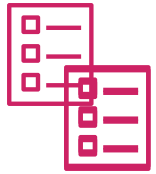
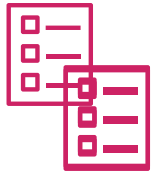
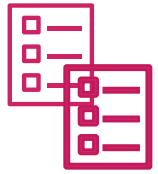
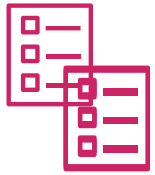


Scenario

1. An application environment created by a CloudFormation **stack**
2. Someone manually modifies the **security group** and opens a new inbound TCP port
3. **Drift detection** is run on the stack
   All resources show **IN_SYNC** except the security group shows status **MODIFIED** with details

Question: In this case, what would be a better approach if the team wants to modify the security group?
Answer: Modify the CloudFormation template security group settings. Then, run **Update Stack**. CloudFormation will **update the security group**. Keeps the model **synchronized** with the deployment.

Scoping and Organizing Templates

Frontend services		Web interfaces, mobile access, analytics dashboard
Backend services		Search, payments, reviews, recommendations
Shared services		CRM databases, monitoring, alarms, subnets, security groups
Network		VPCs, internet gateways, VPNs, NAT devices
Security		IAM policies, users, groups, roles


END