

# Identity and Access Management

IAM

Chandra Lingam

Cloud Wave LLC

# AWS Cloud Security

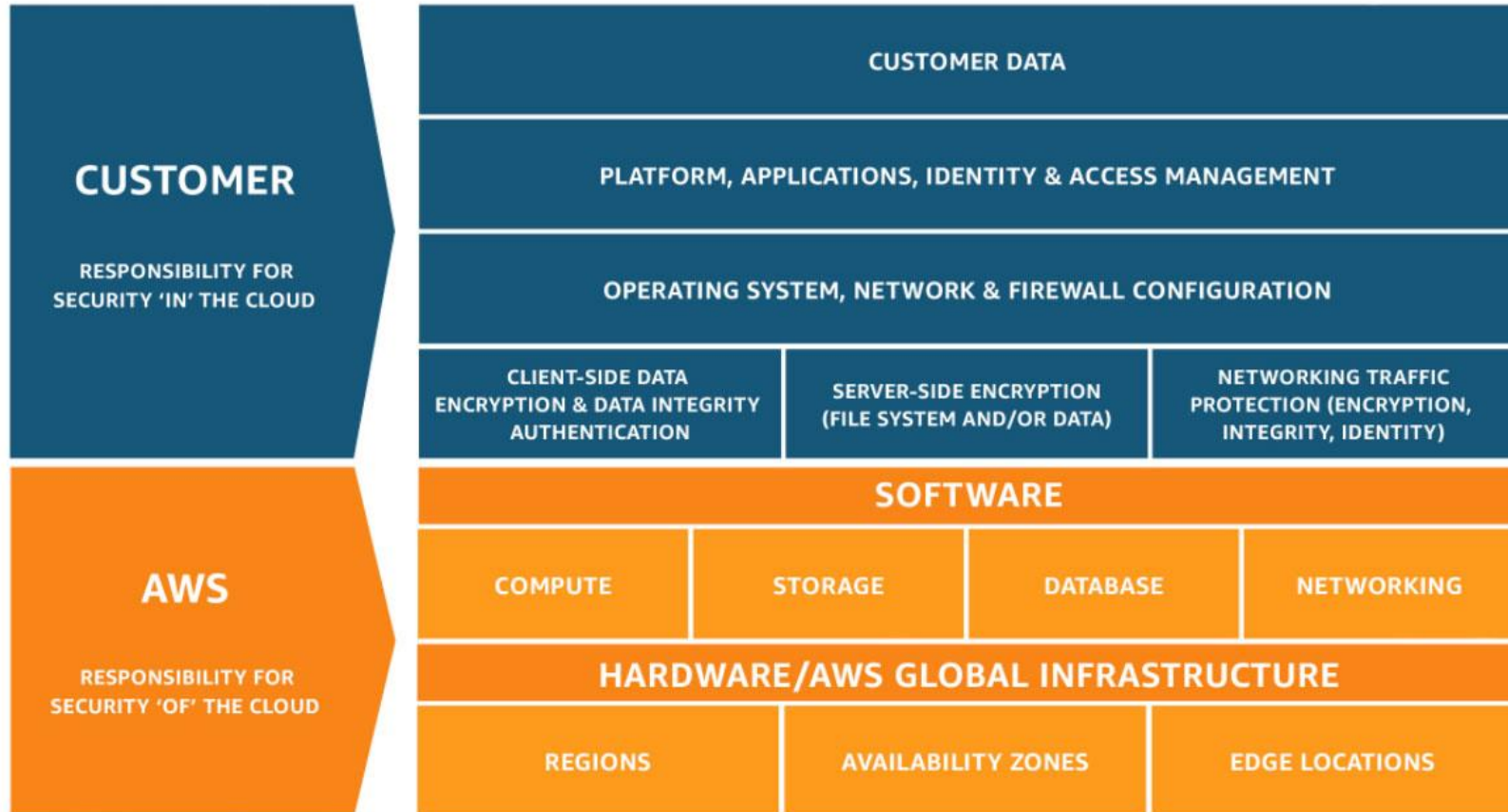


Image Source: [Shared Responsibility Model](#)

# EC2



EC2

The diagram consists of two rounded rectangular boxes stacked vertically. The top box is blue and contains the text 'EC2'. The bottom box is orange and contains the text 'Physical Host'.

Physical Host

## Customer Responsibilities

- Guest OS, Patching
- Firewalls (Security Group, Network ACL)
- Availability, Scalability, Monitoring

## AWS Responsibilities

- Physical Host
- Virtualization

# S3



Bucket

S3

## Customer Responsibilities

- Storage Class
- Access Controls
- Data Encryption

## AWS Responsibilities

- Hardware, Software
- Scalability

# AWS Compliance Programs

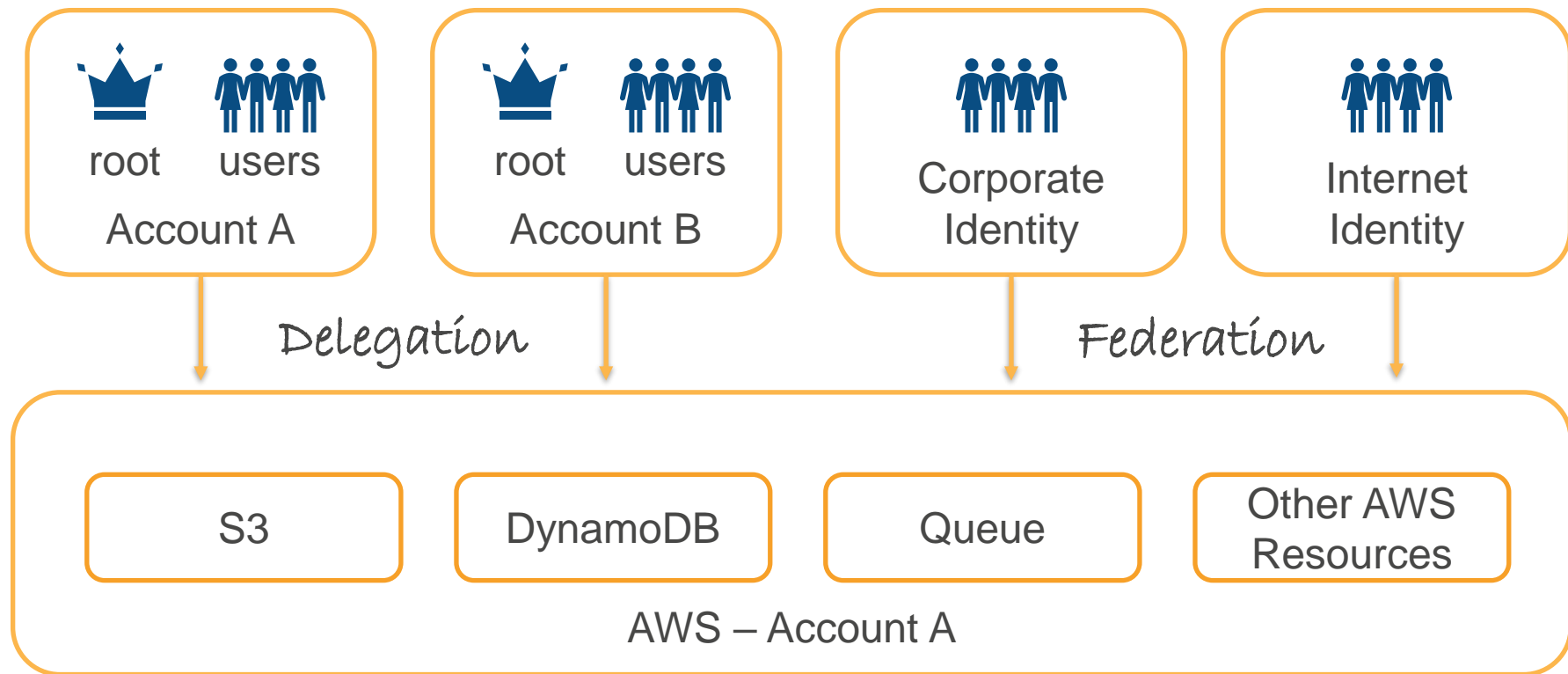
# Useful Resources

[Security is Job Zero | AWS Public Sector Summit 2016](#)

Steve Schmidt

CISO, AWS

# Types of Identities



# Zero Trust

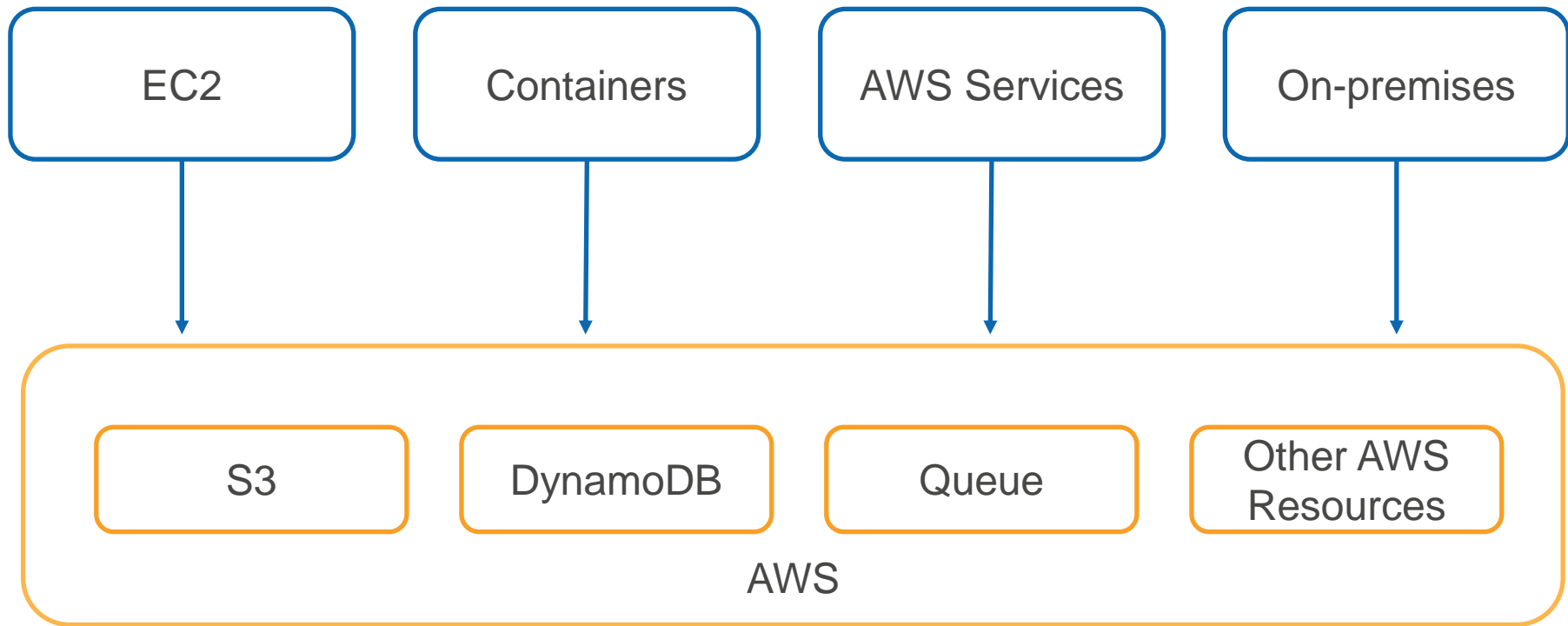
AWS operates on principle of zero trust

- Authentication – caller needs to prove identity
- Authorization – caller needs permission
  
- Some services allow anonymous access
  - S3 bucket with Public Access

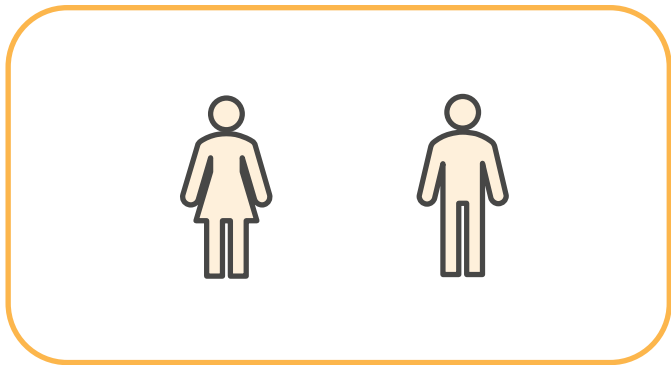
Reference: [AWS re:Invent 2019: Getting started with AWS identity \(SEC209-R1\) by Becky Weiss](#)



# Types of Applications

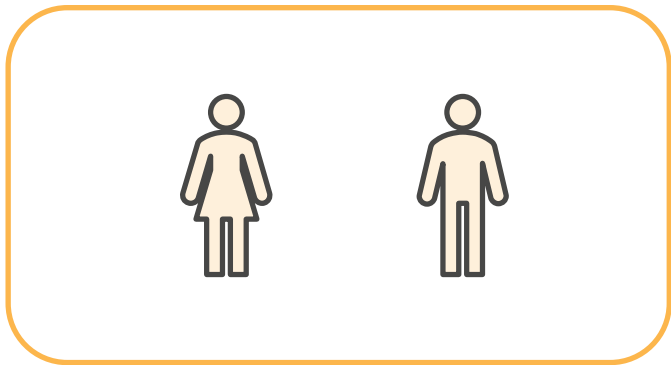


# IAM User Sign-in Credentials



- No sharing of credentials
- User ID and Password for Management Console Access
- Access Key and Secret Access Key for CLI, Programmatic Access
- Optional Multi-Factor Authentication (MFA)

# IAM User Access Management



- No resource access by default
- Identity-based policy - Attach Policy to the User
- Resource-based policy - Attach Policy to the resource (only for supported AWS resources like S3, SQS, Lambda and so forth)
- IAM Roles - Grant temporary access to resources (and user gains privileges assigned to that Role)

# Sample Identity-based Policy

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

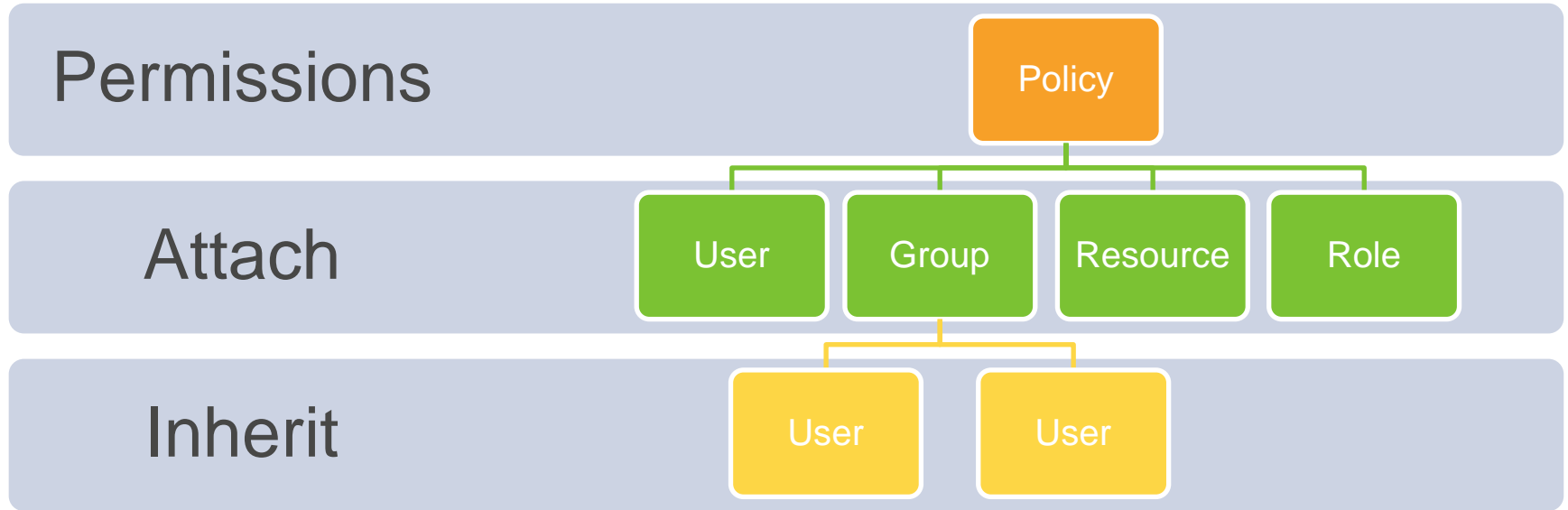
Reference: `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`

# Sample Resource-based Policy

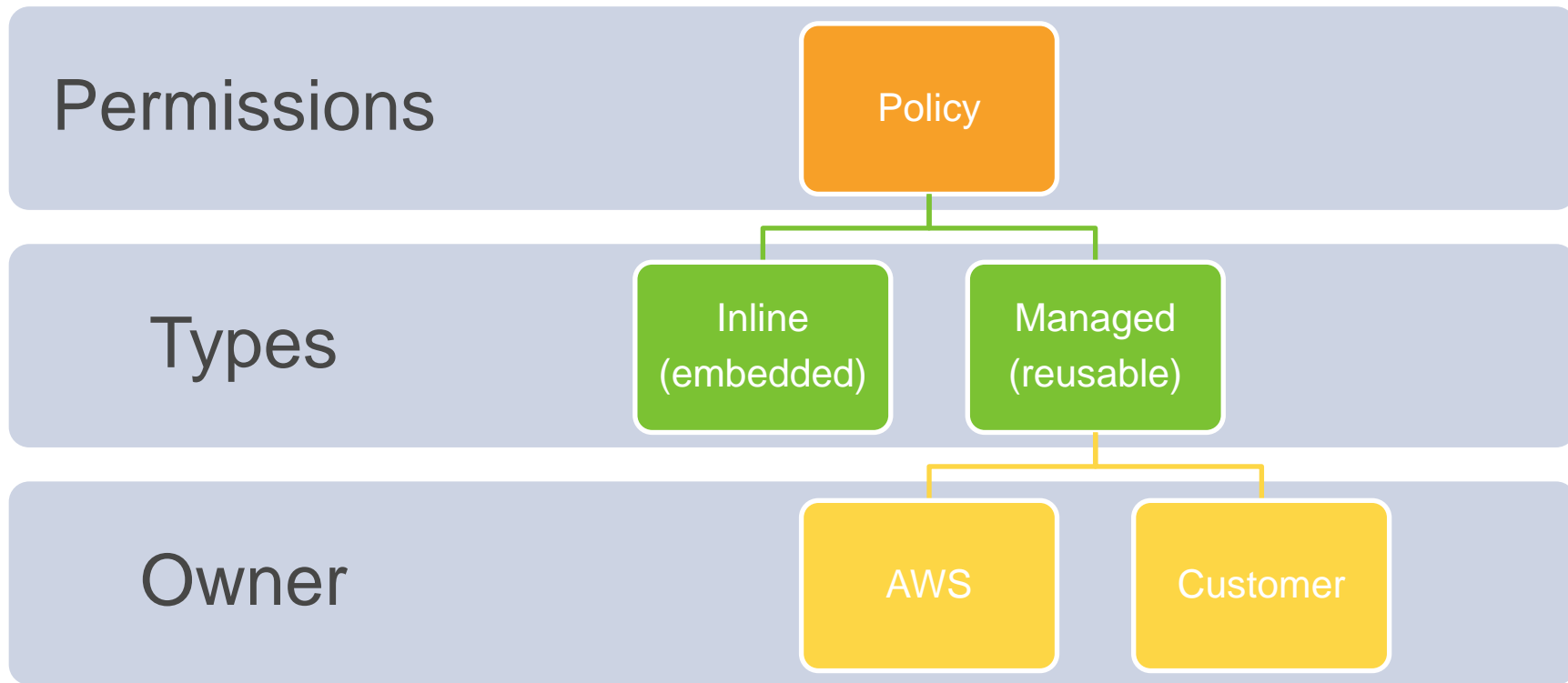
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:Get*", "s3:List*"],
      "Resource": [
        "arn:aws:s3:::bucket_name",
        "arn:aws:s3:::bucket_name/*"],
      "Principal": {
        "AWS": [
          "arn:aws:iam::AWS-account-ID:user/alice",
          "arn:aws:iam::AWS-account-ID:user/bob"]
      }
    ]
  }
}
```

NOTE: [Groups](#) are not supported as principals in any policy

# Access Management Concepts



# Policy Types



# Policy Development and Testing

- Policy Visual Editor

- Policy Examples

[https://docs.aws.amazon.com/IAM/latest/UserGuide/access\\_policies\\_examples.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_examples.html)

- Policy Simulator to Test Policy

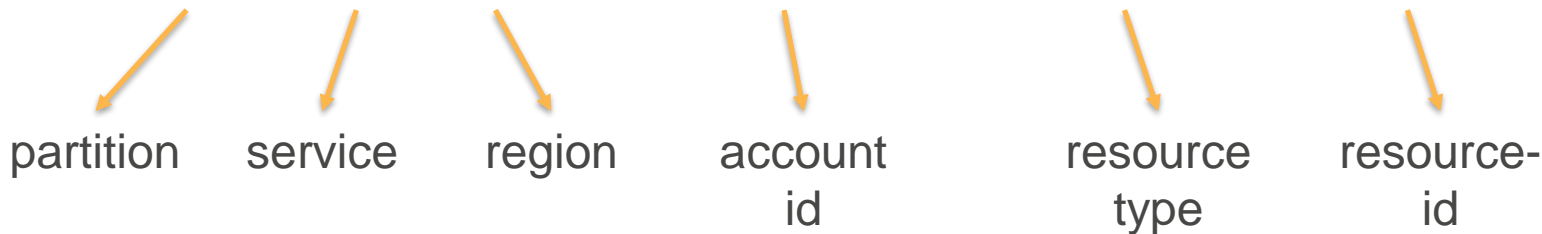
[https://docs.aws.amazon.com/IAM/latest/UserGuide/access\\_policies\\_testing-policies.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html)



# Amazon Resource Name (ARN)

Uniquely identify resource and principal in AWS

`arn:aws:iam::123456789012:user/alice`



# Amazon Resource Name (ARN)

Uniquely identify resource and principal in AWS

`arn:aws:iam::123456789012:user/alice`

`arn:aws:iam::123456789012:policy/db_admin`

`arn:aws:s3:::my_bucket`

`arn:aws:sqs:us-east-2:123456789012:order`

# ARN – Structure and Examples

## Structure

arn:partition:service:region:account-id:resource-id

arn:partition:service:region:account-id:resource-type/resource-id

arn:partition:service:region:account-id:resource-type:resource-id

## Examples

# Policy Document Structure (1/3)

Element	Description
Version	Current version of the policy language. You should always set the version element. Current version is “2012-10-17”.
Statement	Permissions are allowed or denied using Statements. A policy document can have one or more statements
Effect	Specifies if a statement allows or denies permission Valid Values: Allow, Deny
Principal	Identity for which the statement applies. Implied when attached to the identity-based policy (for example, a user) Needs to be specified in resource-based policy and IAM Role trust policy <a href="#">Groups</a> are not supported as principals in any policy

# Example of Principal

Principal	Example
AWS account, root user	<code>"Principal":{"AWS":"arn:aws:iam::123456789012:root"}</code> <code>"Principal":{"AWS":"123456789012"}</code>
IAM Users	<code>"Principal":{"AWS":"arn:aws:iam::123456789012:user/alice"}</code>
IAM Roles	<code>"Principal":{"AWS":"arn:aws:iam::123456789012:role/cross-acct"}</code>
AWS Services	<code>"Principal":{"Service":"elasticmapreduce.amazonaws.com"}</code>
Anonymous users	<code>"Principal": "*"}</code>
Federated Users	<code>"Principal":{"Federated":"www.amazon.com"}</code>
<u>Assumed-role sessions</u>	Use the role session name to uniquely identify a session when the same role is assumed by different principals or for different reasons <code>"Principal":{"AWS":"arn:aws:sts::123456789012:assumed-role/role-name/role-session-name"}</code>

# Policy Document Structure (2/3)

Element	Description
Action	<p>Service API actions for which statement applies</p> <p>Example:</p> <p>“Action”:”s3:Get*” - All S3 API Calls that have a Get Prefix</p> <p>“Action”:”s3:*” - All S3 API Calls</p>
Resource	<p>Resource for which the statement applies</p> <p>Example:</p> <p>“Resource”:”arn:aws:s3:::bucket_name“ – Actions apply to specified bucket</p> <p>“Resource”:”arn:aws:s3:::bucket_name/*“ – Actions apply to objects stored in the specified bucket</p>
Conditions	<p>Additional conditions for fine grained access</p> <p>Example: IP Address, Authentication Mechanism</p>

# Sample Identity-based Policy

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

Reference: `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`

# Sample Resource-based Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:Get*", "s3:List*"],
      "Resource": [
        "arn:aws:s3:::bucket_name",
        "arn:aws:s3:::bucket_name/*"],
      "Principal": {
        "AWS": [
          "arn:aws:iam::AWS-account-ID:user/alice",
          "arn:aws:iam::AWS-account-ID:user/bob"]
      }
    ]
  }
}
```

NOTE: [Groups](#) are not supported as principals in any policy



# Policy Document Structure (3/3)

Element	Description
NotAction	Matches everything except specified API action Example: <code>"NotAction": "s3:DeleteBucket"</code> – All S3 actions except for deleting a bucket
NotResource	Match everything except the specified resource Example: <code>"NotResource": [ "arn:aws:s3:::bucket_name", "arn:aws:s3:::bucket_name/*" ]</code>
NotPrincipal	Match all principals except for specified principal Example: <code>"Effect": "Deny", "NotPrincipal": [ "AWS": "123456789012" ]</code>

# Sample – Limit access to S3 by Source IP

```
"Statement": [  
  {  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Resource": [  
      "arn:aws:s3:::bucket_name",  
      "arn:aws:s3:::bucket_name/*"],  
    "Principal": "*",  
    "Condition": {  
      "NotIpAddress": {  
        "aws:SourceIp": "151.29.0.0/16"  
      }  
    }  
  }  
]
```

Reference: [https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_examples\\_aws\\_deny-ip.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_aws_deny-ip.html)

# Sample Conditional Variables – VPC Endpoint Only Access

```
"Statement": [  
  {  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Resource": [  
      "arn:aws:s3:::bucket_name",  
      "arn:aws:s3:::bucket_name/*"],  
    "Principal": "*",  
    "Condition": {  
      "StringNotEquals": {  
        "aws:sourceVpce": "vpce-0f143973exyzabcdef"  
      }  
    }  
  }  
]
```

Reference: <https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies-vpc-endpoint.html>

# Improved VPC Endpoint Only Access

```
"Statement": [  
  {  
    "Effect": "Deny",  
    "Action": "s3:*",  
    "Resource": [  
      "arn:aws:s3:::bucket_name",  
      "arn:aws:s3:::bucket_name/*"],  
    "NotPrincipal": {  
      "AWS": [  
        "arn:aws:iam::AWS-account-ID:root",  
        "arn:aws:iam::AWS-account-ID:user/myadmin"]},  
    "Condition": {  
      "StringNotEquals": {  
        "aws:sourceVpce": "vpce-0f143973exyzabcdef"  
      }  
    }  
  }  
]
```

Reference: <https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies-vpc-endpoint.html>

# Global Environment Data (request context)

Key	Description
aws:CurrentTime	Data and time of the request – use it for timebased restrictions
aws:RequestedRegion	AWS Region used in the request – use it to limit access only to specific region(s)
aws:PrincipalTag	Compare the tag attached to the principal making the request
aws:SecureTransport	Check if request was sent using SSL. values: True or false
aws:SourceIP, aws:SourceVpc, aws:SourceVpce	Check if request comes from whitelisted IP, VPC or VPC Endpoint

# Attribute Based Access Control (ABAC)

Allow actions only when cost center match

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

Reference: [https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_examples\\_ec2-start-stop-match-tags.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_ec2-start-stop-match-tags.html)

# Role Based Access Control (RBAC)

- “Traditional authorization model used in IAM is called role-based access control. This is based on a person's job-role.” [in AWS context, role refers to IAM-role]
- “In RBAC model, you implement different policies for different job functions.”
- “As best policy, you grant the minimum permissions necessary for the job function. this is known as "granting least privilege".”
- “The disadvantage of RBAC model is that when employees add new resources, you must update policies to allow access to those resources.”

# Attribute Based Access Control (ABAC)

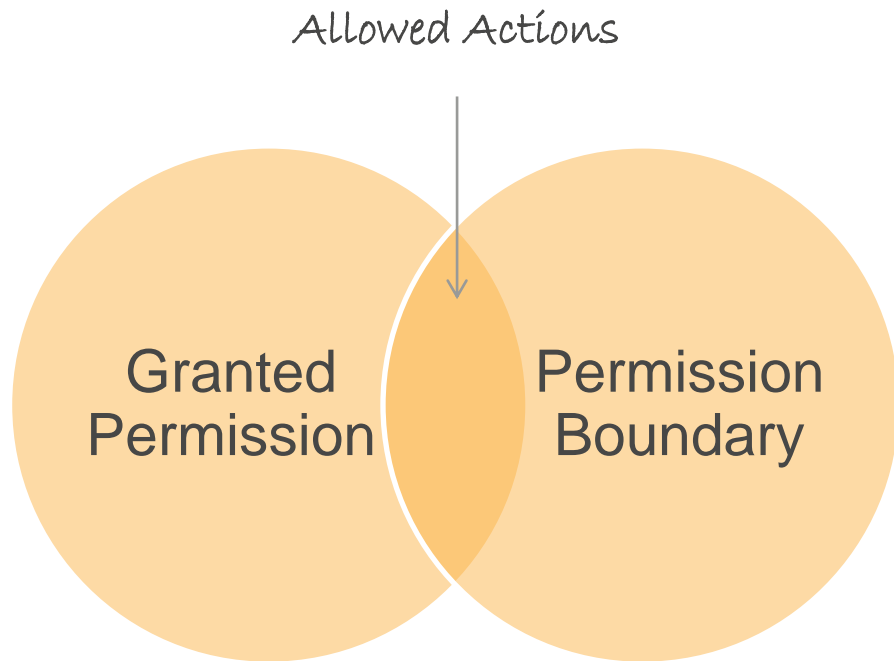
- “ABAC is an authorization strategy that defines permissions based on attributes (also known as Tags)”
- “Tags can be attached to IAM Principals (users or roles) and to AWS resources”
- “With ABAC, policies can be designed to allow operations when the principal's tag matches the resource tag.”
- “ABAC is useful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome”
- “With ABAC, permissions scale - it is no longer necessary for administrator to update existing policies.”
- “ABAC requires fewer policies”



# Policy Evaluation

- Implicit Deny – By default, all requests are implicitly denied
- Explicit Allow – Overrides this default
- Explicit Deny – Overrides explicit allow
- Permissions Boundary, Organization Service Control Policy, Session Policy – it might override the allow with an implicit deny

# Lab – IAM User Boundary Permissions



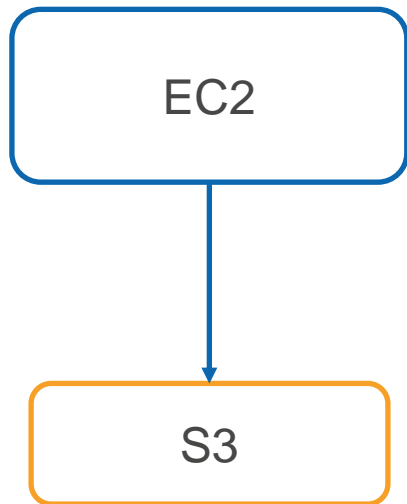
- Demonstrate the purpose of boundary permissions
- How you can implement guardrails to prevent excessive permissions

# Maximum Permissions

- IAM Permission Boundaries – Sets maximum permissions that an identity-based policy can grant to an entity (user or role)
  - When Set, an entity can perform only the actions that are allowed by identity based policies and permission boundaries
- Session Policy – When you assume role for temporary credentials, you can include a policy as a parameter – this is helpful when you want to limit what the temporary credential can do

# IAM Roles

# Application Access to AWS Resources



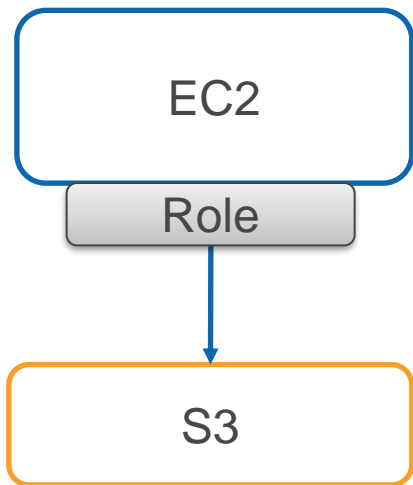
Access Key Credentials (Treat the server as a user):

- Generate Access Key, Secret Access Key
- Store the key in the EC2 instance
- EC2 uses the credentials to talk to services

Issues:

- Access Key Credentials are long-term (several days to months)
- Security risks due to long term credentials (accidental leakage, malicious users)
- Credential rotation is a problem at scale

# Application Access to AWS Resources



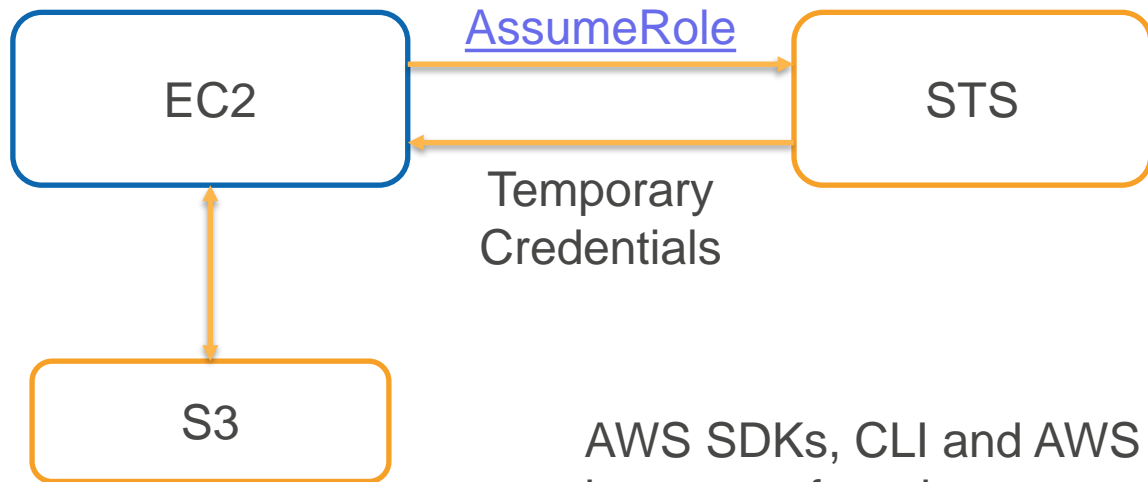
## IAM Roles:

- Attach an IAM Role to the instance
- EC2 instance talks to metadata service to get temporary credentials for the role
- EC2 uses the temporary credentials to talk to services

## Benefits:

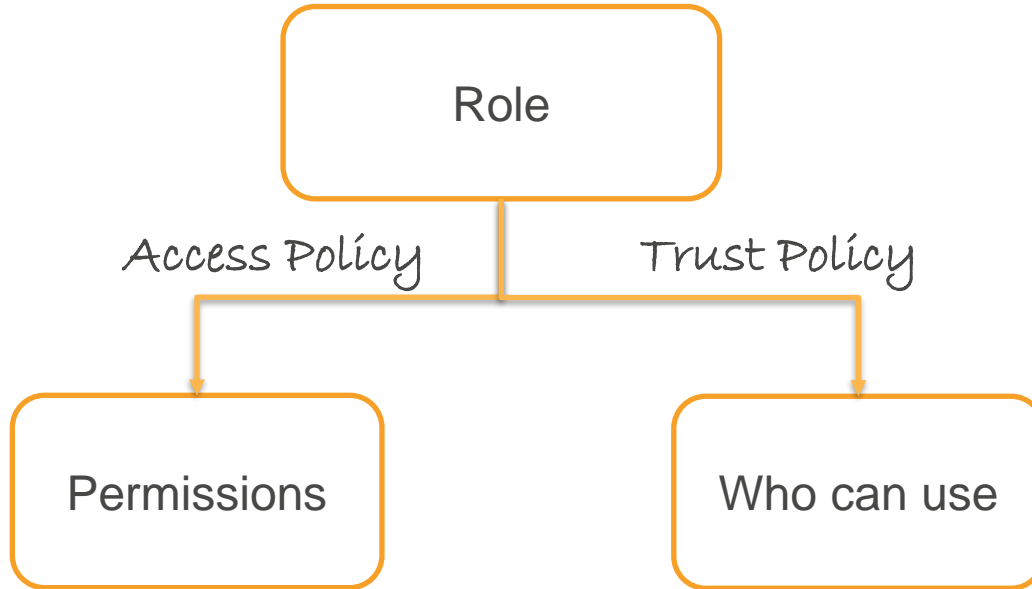
- No need to maintain credentials in the server
- Automatic Credential rotation – Credentials are valid only for a few hours (configurable)
- Reduced impact due to accidental leakage (credential is replaced every few hours)

# Security Token Service (STS)



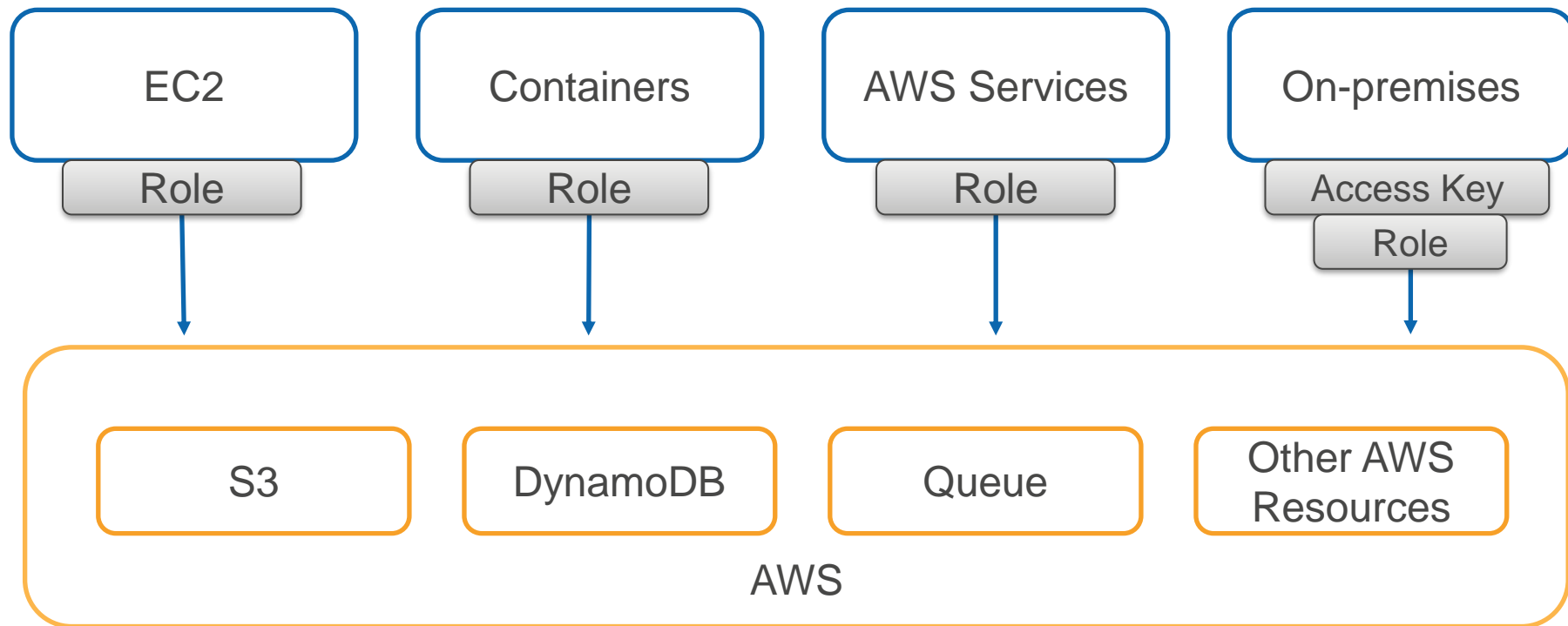
AWS SDKs, CLI and AWS Services have built-in support for roles

# Role Concepts

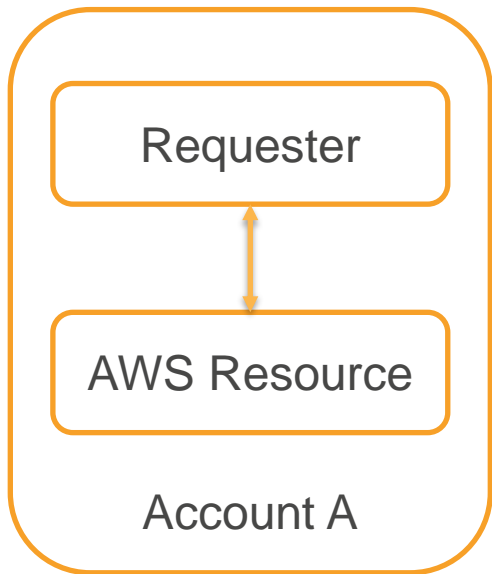




# Application Access To Resources

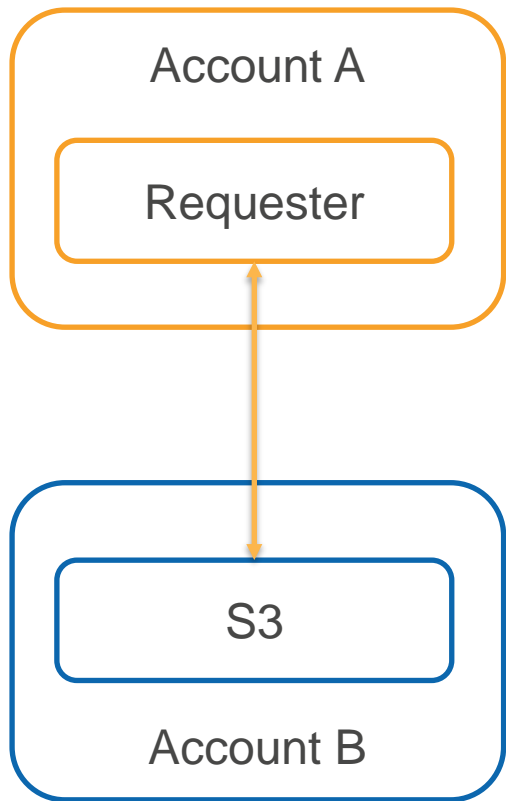


# Same Account Access



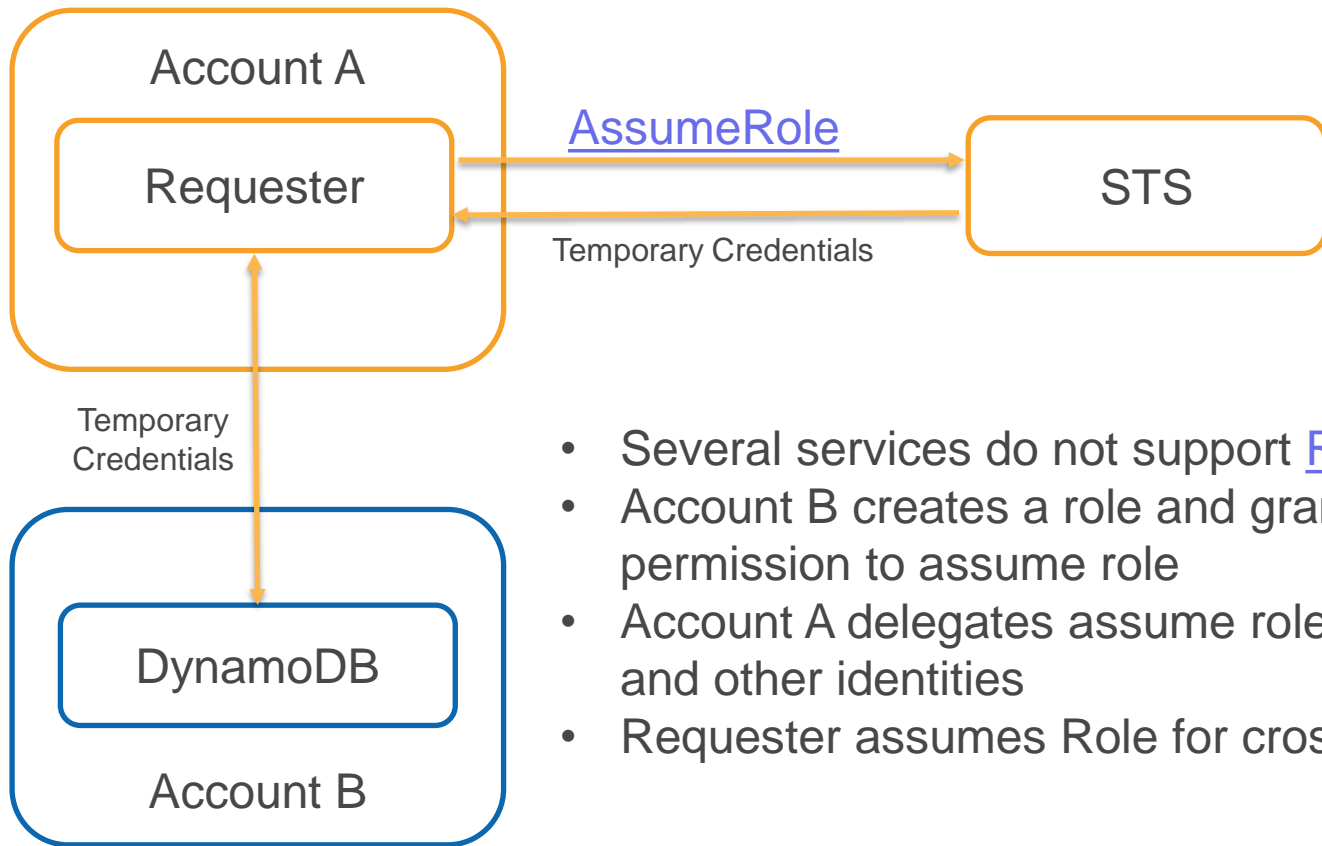
- Identity-based policies – for access to any AWS resource
- Resource-based policies – for access to supported resources
- Roles - for EC2 and Service integration

# Cross Account Access using Resource Based Policy



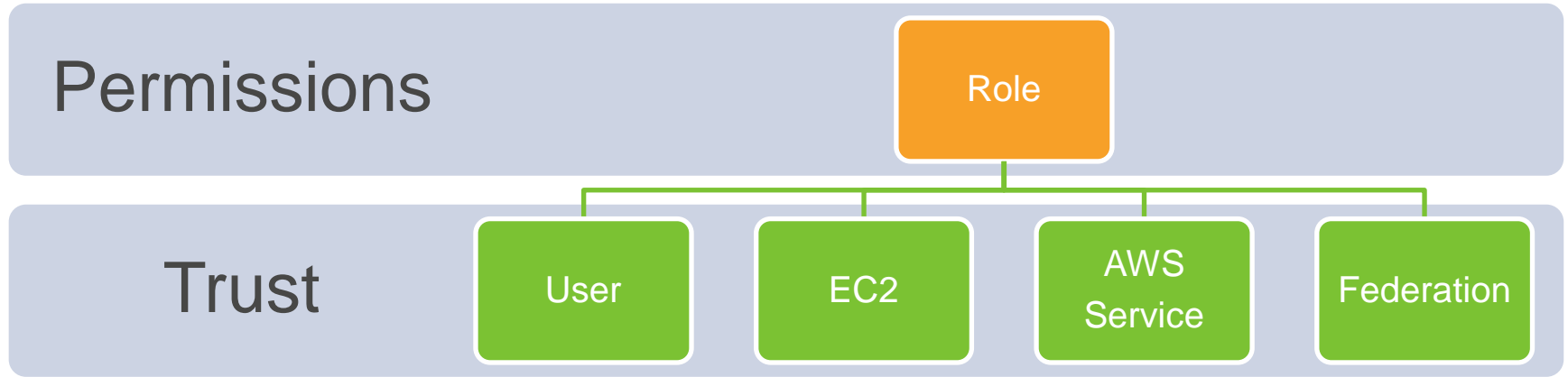
- [Resource Based Policies](#) (for supported resources like S3, SQS, SNS, Lambda)
- Identity in Account A needs access to resource in Account B
  - Resource owner (Account B) grants permission to Account A
  - Account A delegate's permission to the other identities in the account (user, role)

# Cross Account Access Using Roles



- Several services do not support [Resource Based Policy](#)
- Account B creates a role and grants Account A permission to assume role
- Account A delegates assume role permission to users, and other identities
- Requester assumes Role for cross account access

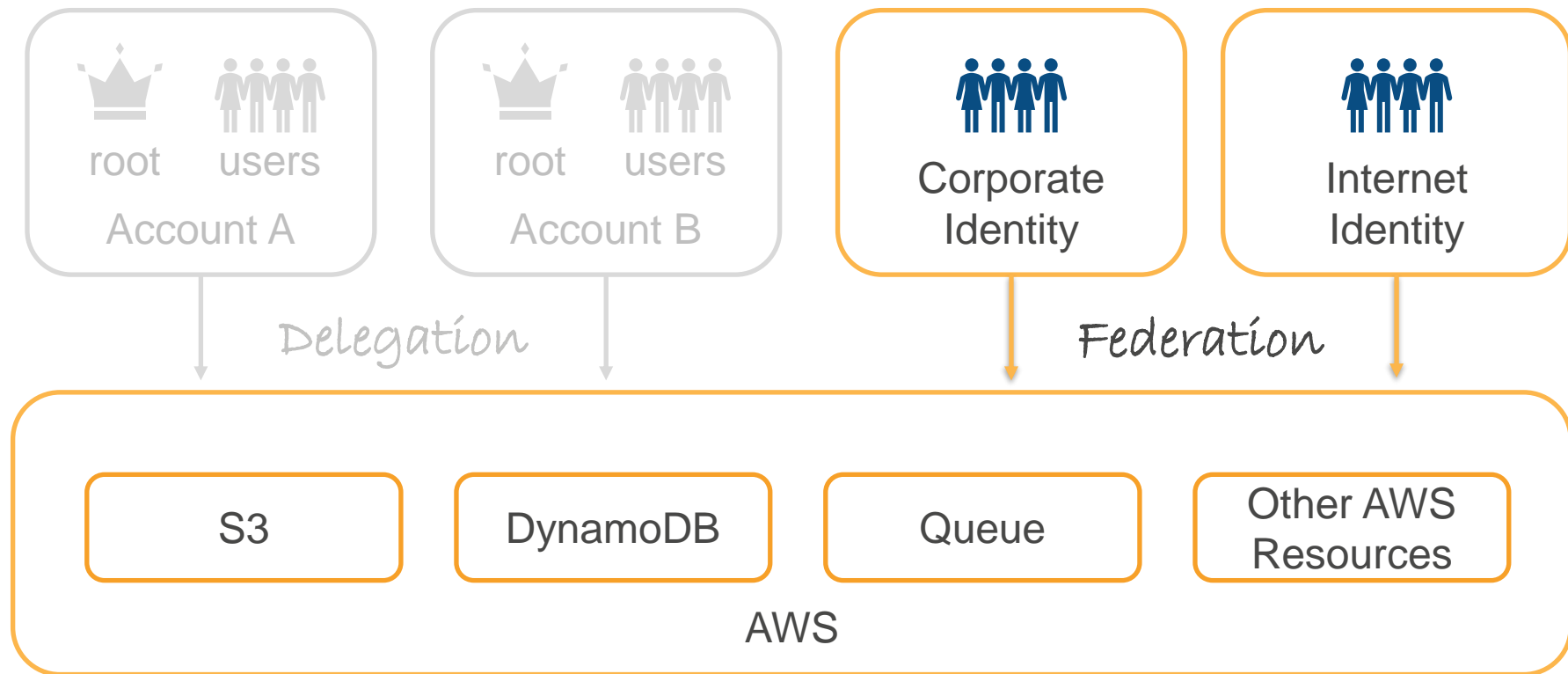
# Role



Role has two parts:

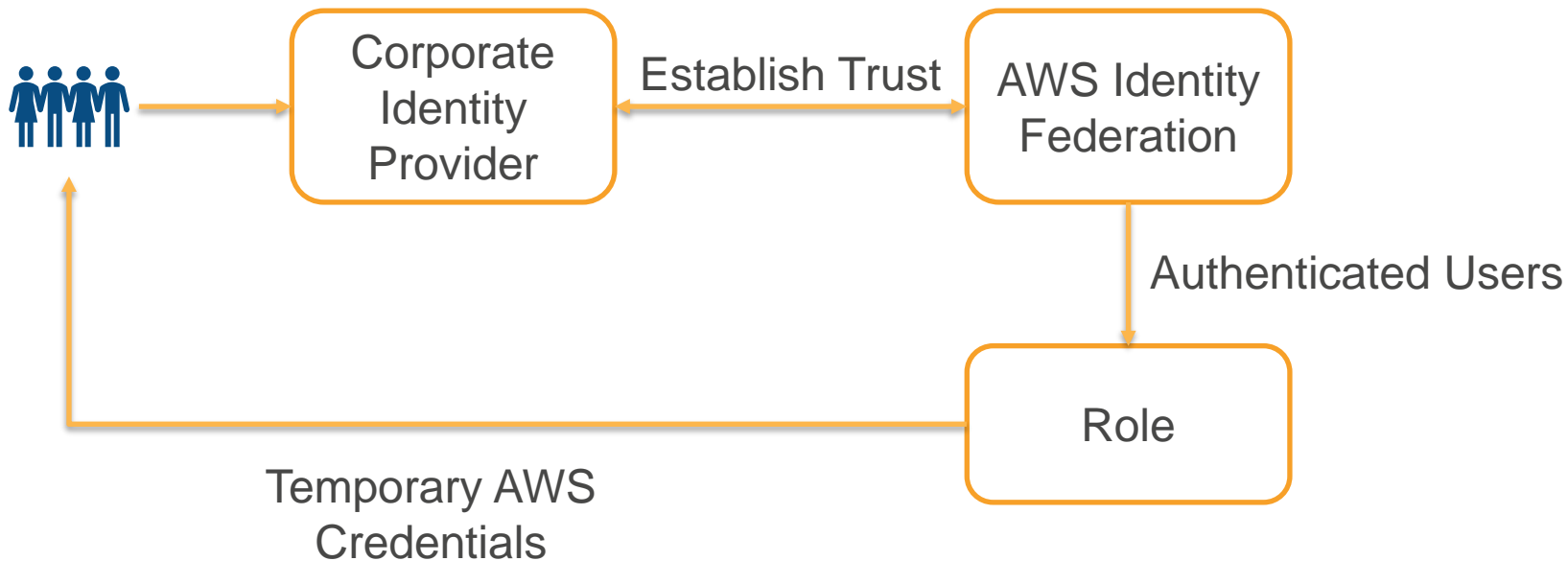
1. Who can assume the role (trust relationship) and
2. What access is allowed (permissions)

# Types of Identities



# Corporate Identity Federation

*SAML 2.0, Microsoft Active Directory*

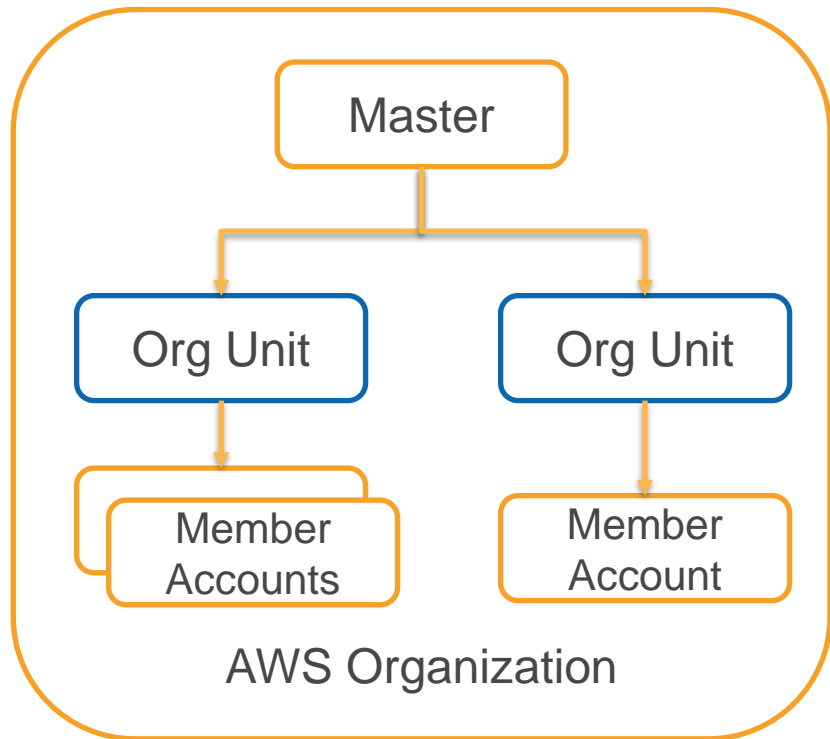


# Corporate Identity Federation

- SAML 2.0 (Security Assertion Mark Up Language) to exchange identity and security information between identity provider and application
- AWS IAM Federation – enables users sign-in to their AWS account with existing corporate credentials
- Non-SAML options – AWS Directory Service for Microsoft Active Directory
- AWS Organizations – Use AWS Single Sign On (SSO) to scale to multiple AWS Accounts (centrally manage access)

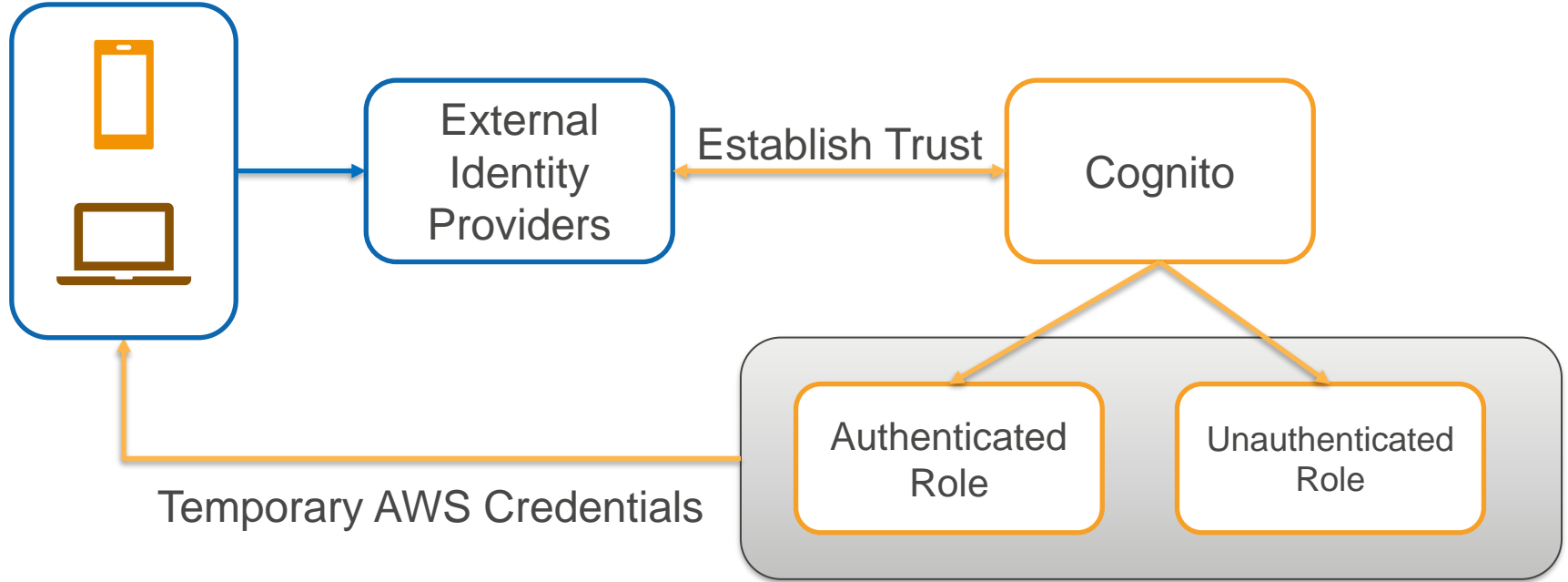


# AWS Organizations



- Centrally manage costs and billing
- Service Control Policy – control services, resources, regions used by member account
- Share resources across accounts
- AWS Single Sign-on – manage access to employees and accounts
- Centralize identity management and federation

# Internet Identity Federation



*SAML 2.0, OAuth 2.0, OpenID Connect*

# Cognito Identity Federation

- Users can sign-in to mobile and web apps using social identity providers like Facebook, google, amazon
- Support for corporate identity federation using SAML 2.0
- Open Standards Support - Oauth 2.0, SAML 2.0, OpenID Connect
- Map users to roles and limit access to resources

# Useful Resources

[AWS re:Invent 2019: Getting started with AWS identity \(SEC209-R1\) by Becky Weiss](#)

[AWS re:Invent 2015: How to Become an IAM Policy Ninja in 60 Minutes or Less \(SEC305\) by Jeff Wierer](#)

# Lab – Identity-based Policy

- IAM Users, Groups
- Programmatic Access and Management Console Access
- Implicit Deny
- Explicit Allow
- Managed Policy

# Lab – Resource-based Policy

Explore resource-based policy

Configure principals, resources and actions

# Lab – IP Based Restriction

Policy with conditional variable

Limit access to S3 Bucket based on requester IP

For this lab, we use the `IPRestrictionPolicy` file available under resources

Reference

[https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_examples\\_aws\\_deny-ip.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_aws_deny-ip.html)

# Lab – VPC Endpoints

Allow access to the bucket only from VPC

Route traffic internally on AWS network using VPC Endpoint

For this lab, we use the policy file available under resources: `VPCEndPoIntRestrictionPolicy`

Reference: <https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-s3.html#vpc-endpoints-policies-s3>



# Lab – Cross Account Access using Resource Policy

- Bucket and the user are in separate accounts
- You need two accounts for this lab
- Manage permission using resource-based policy

For this lab, we use the policy file:

`ResourceBasedCrossAccountAccessPolicy`

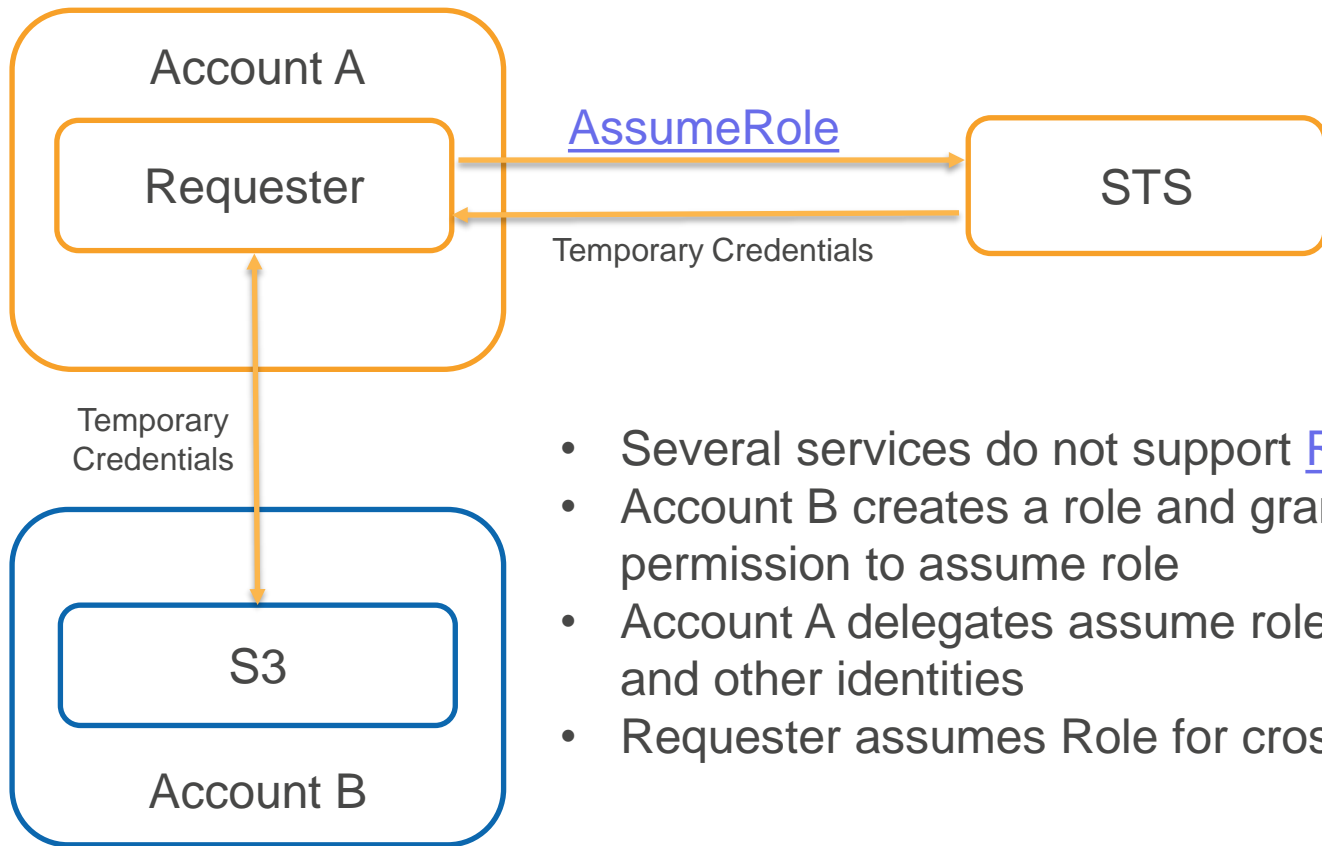
# Lab – Cross Account Access using IAM Roles

- Explore cross-account resource sharing using IAM Roles
- Temporary Credentials using Assume Role
- Demonstrate access from browser, laptop and from EC2 instance

For this lab, we use the policy file:

`RoleBasedCrossAccountAccessPolicy`

# Lab - Cross Account Access Using Roles



- Several services do not support Resource Based Policy
- Account B creates a role and grants Account A permission to assume role
- Account A delegates assume role permission to users, and other identities
- Requester assumes Role for cross account access



Chandra Lingam

57,000+ Students



For AWS self-paced video courses, visit:

<https://www.cloudwavetraining.com/>

