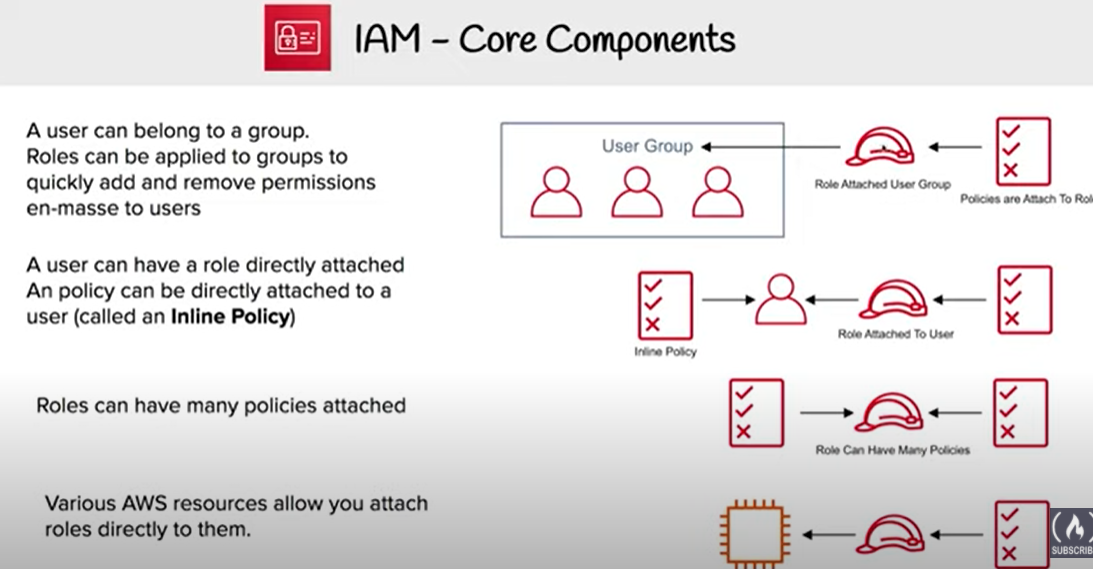
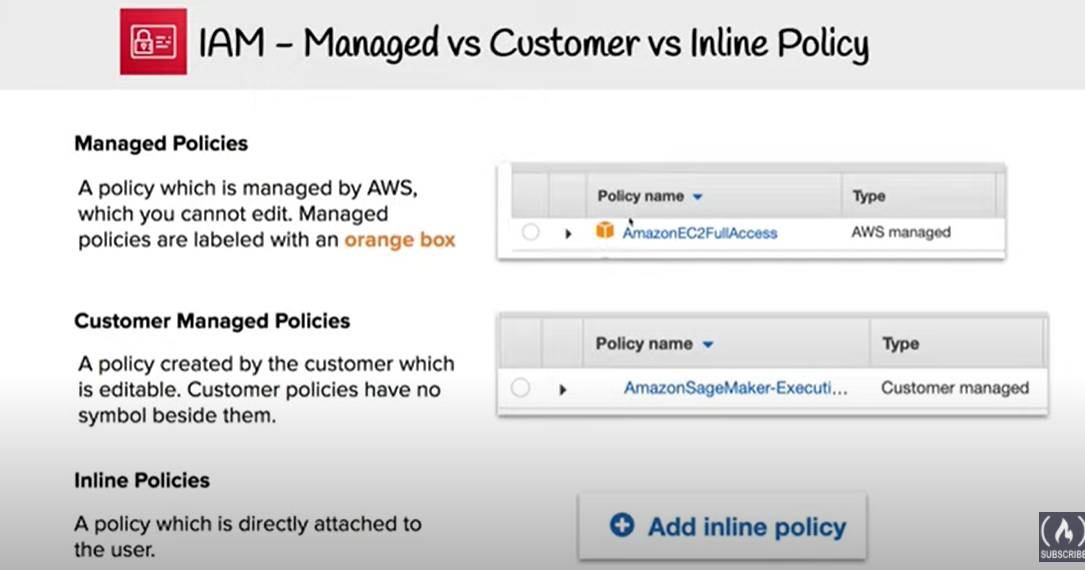
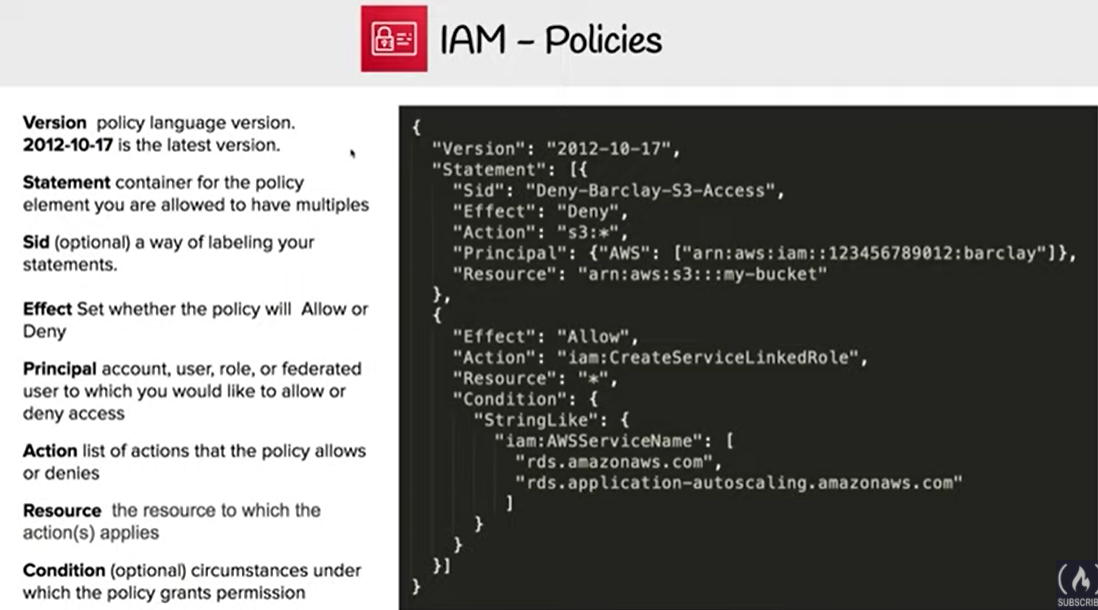
AWS IAM Role

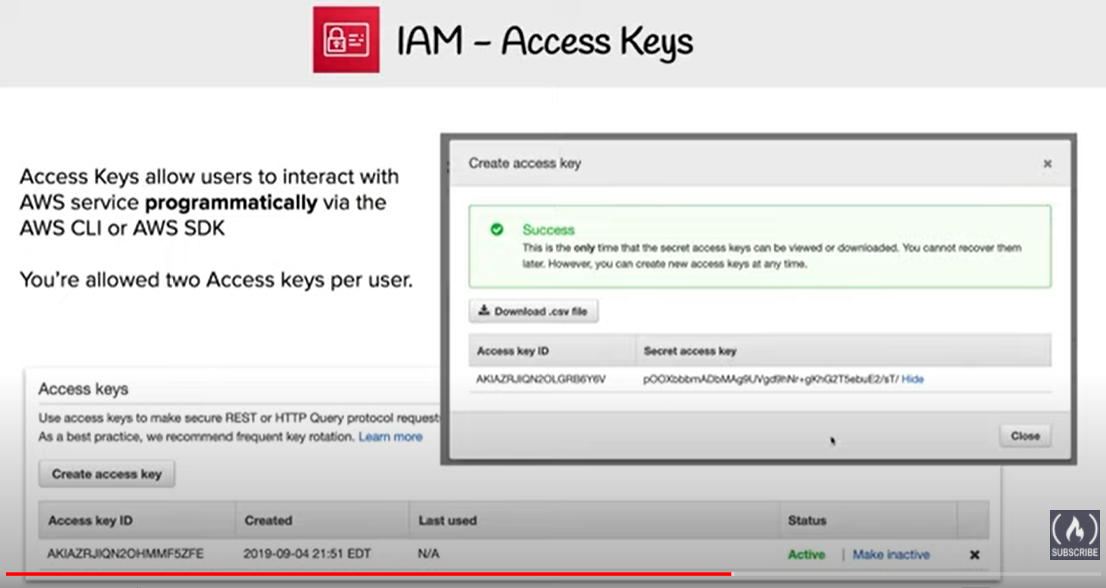
# 

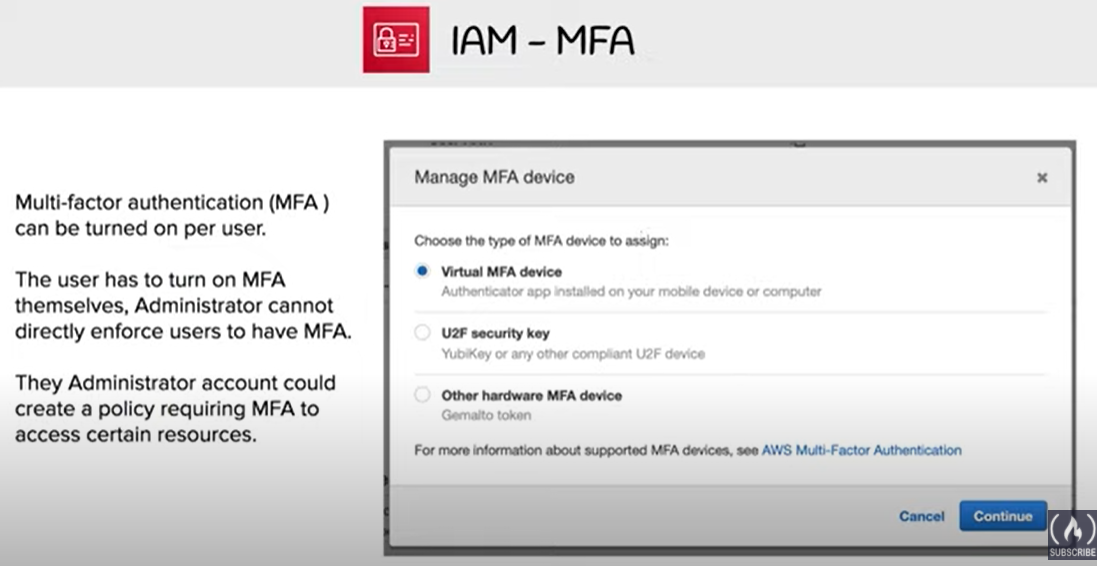












Graphical user interface, application

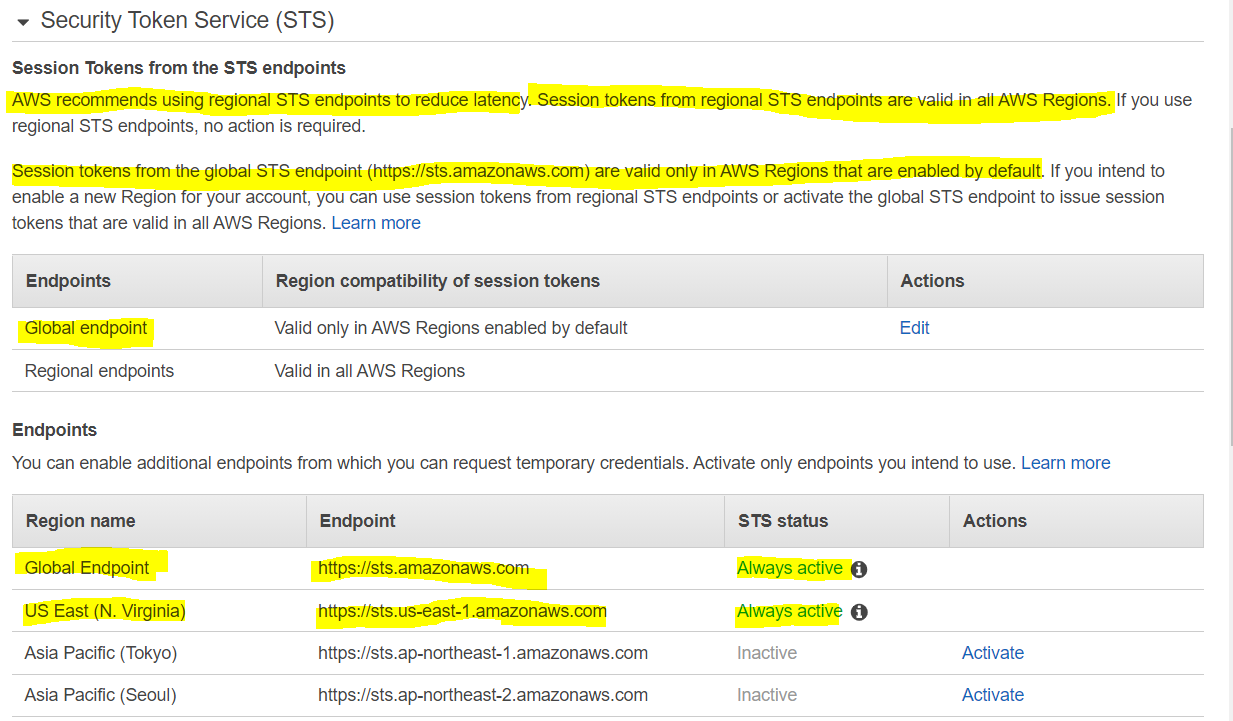
Description automatically generated

# AWS IAM Role

* IAM role is very similar to a user, in that it is an identity with permission policies that determine what the identity can and cannot do in AWS.
* **IAM role is not intended to be uniquely associated with a particular user, group or service and is intended to be assumable by anyone who needs it.**
* **Role does not have any credentials (password or access keys) associated with it and whoever assumes the role is provided with a dynamic temporary credentials**
* **Role helps in access delegation to grant permissions to someone that allows access to resources that you control**
* **Roles can help to prevent accidental access to or modification of sensitive resources**
* **Modification of a Role can be done anytime and the changes are reflected across all the entities associated with the Role immediately**
* IAM Role plays a very important role in the following scenarios
  + **Services like EC2 instance running an application that needs to access other AWS services**
  + **Allowing users from different AWS accounts have access to AWS resources in different account, instead of having to create users**
  + **Company uses a Corporate Authentication mechanism and don’t want the User to authenticate twice or create duplicate users in AWS**
  + **Applications allowing login through external authentication mechanism e.g. Amazon, Facebook, Google etc**
* Role can be assumed by
  + IAM user within the same AWS account
  + IAM user from a different AWS account
  + AWS service such as EC2, EMR to interact with other services
  + **An external user authenticated by an external identity provider (IdP) service that is compatible with SAML 2.0 or OpenID Connect (OIDC), or a custom-built identity broker.**
* Role involves defining two policies
  + **Trust policy**
    - Trust policy defines – who can assume the role
    - Trust policy involves setting up a trust between the account that owns the resource (trusting account) and the account who owns the user that needs access to the resources (trusted account)
  + **Permissions policy**
    - Permissions policy defines – what they can access
    - Permissions policy determines authorization, which grants the user of the role with the needed permissions to carry out the desired tasks on the resource
* **Federation** is creating a trust relationship between an external Identity Provider (IdP) and AWS
  + Users can also sign into an enterprise identity system that is compatible with SAML
  + Users can sign into a web identity provider, such as Login with Amazon, Facebook, Google, or any IdP that is compatible with OpenID connect (OIDC).
  + When using OIDC and SAML 2.0 to configure a trust relationship between these external identity providers and AWS, the user is assigned to an IAM role and receives temporary credentials that enables the user to access AWS resources
* **IAM Best Practice** – **Use roles for applications running on EC2 instances**
* **IAM Best Practice** – **Delegate using roles instead of sharing credentials**

## AWS STS & Temporary Credentials

* AWS Security Token Service (STS) helps create and provide trusted users with temporary security credentials that control access to AWS resources
* **STS is a global service with a single endpoint https://sts.amazonaws.com**
* AWS STS API calls can be made either to a global endpoint or to one of the regional endpoints. Regional endpoint can help reduce latency and improve the performance of the API calls
* Temporary Credentials are similar to Long Term Credentials except for
  + are short term and are regularly rotated
  + can be configured to last from few minutes to several hours
  + do not have to be embedded or distributed
  + are not stored or attached with the User, but are generated dynamically and provided to the user as and when requested



## Role types

### AWS Service Roles

* Some AWS services need to interact with other AWS services for e.g. EC2 interacting with S3, SQS etc
* Best practice is to assign these services with IAM roles instead of embedding or passing IAM user credentials directly into an instance, because distributing and rotating long-term credentials to multiple instances is challenging to manage and a potential security risk.
* AWS automatically provides temporary security credentials for these services e.g. Amazon EC2 instance to use on behalf of its applications
* Deleting a role or instance profile that is associated with a running EC2 instance will break any applications running on the instance

#### **Complete Process Flow**

1. Create a IAM role with services who would use it for e.g. EC2 as trusted entity and define permission policies with the access the service needs
2. Associated a Role (actually an Instance profile) with the EC2 service when the instance is launched
3. Temporary security credentials are available on the instance and are automatically rotated before they expire so that a valid set is always available
4. **Application can retrieve the temporary credentials either using the**[**Instance metadata**](https://jayendrapatil.com/aws-ec2-instance-metadata-userdata/)**directly or through AWS SDK**
5. Applications running on the EC2 instance can now use the permissions defined in the Role to access other AWS resources
6. Application, if caching the credentials, needs to make sure it uses the correct credentials before they expire

Timeline

Description automatically generated

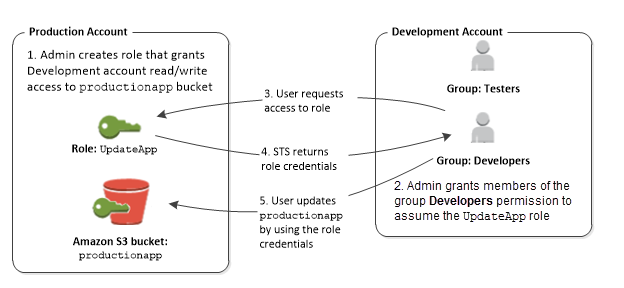
#### **Instance Profile**

* An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.
* **If a Role is created for EC2 instance or any other service that uses EC2 through AWS Management Console, AWS creates a Instance profile automatically with the same name as the Role. However, if the Role is created through CLI the instance profile needs to create as well**
* **An instance profile can contain only one IAM role**. **However, a role can be included in multiple instance profiles.**

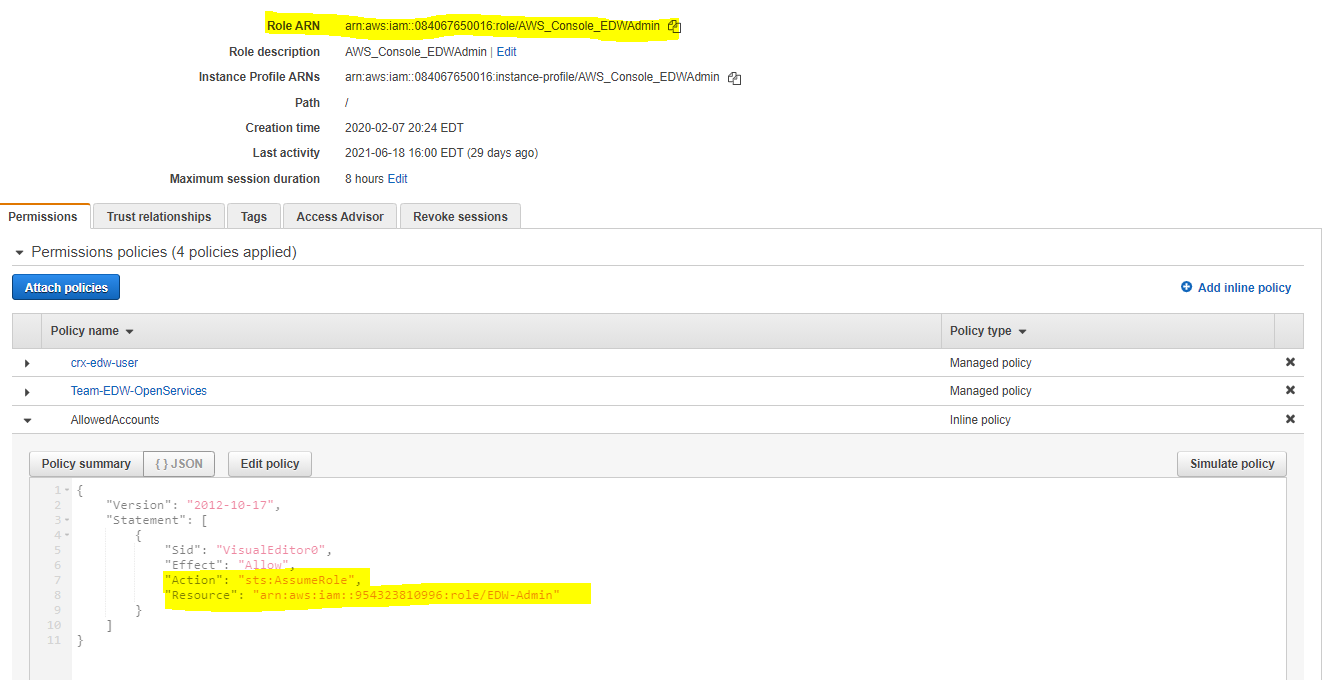
### Cross-Account access Roles

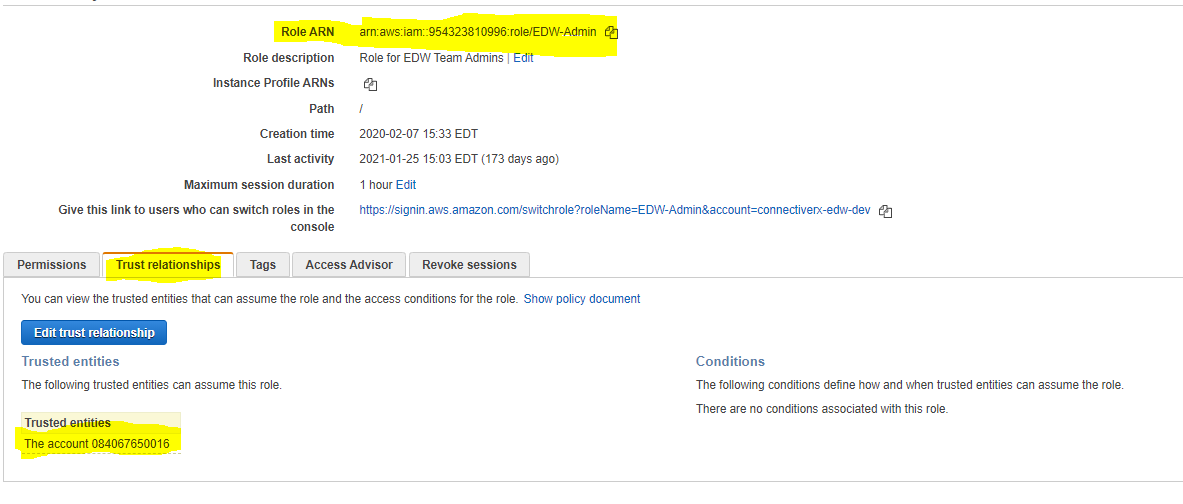
* IAM users can be granted permission to switch roles within the same AWS account or to roles defined in other AWS accounts that you own.
* Roles can also be used to delegate permissions to IAM users from AWS accounts owned by Third parties
  + You must explicitly grant the users permission to assume the role.
  + Users must actively switch to the role using the AWS Management Console.
  + Multi-factor authentication (MFA) protection can be enabled for the role so that only users who sign in with an MFA device can assume the role
* **However, only One set of permissions are applicable at a time. User who assumes a role temporarily gives up his or her own permissions and instead takes on the permissions of the role. When the user exits, or stops using the role, the original user permissions are restored.**

#### **Complete Process Flow**



1. Trusting account creates a IAM Role with a
   * Trust policy which defines the account (trusted account) as a principal who can access the resources and a
   * Permissions policy to define what resources can the user in the trusted account access
2. Trusting account provides the Account ID and the Role name (or the ARN) to the trusted account





1. If the Trusting account is own by Third Party it can optionally provide an External ID (recommended for additional security), required to uniquely identify the trusted account, which can be added to the trust policy as a condition
2. Trusted account creates a IAM user who has permissions (Permission to call the AWS Security Token Service (AWS STS) ***AssumeRole***API for the role) to assume the role/switch to the role.
3. IAM User in the Trusted account switches to the Role/assumes the role and passes the ARN of the role
4. Trusted account belonging to the Third party would also pass the External ID mapped to the Trusting account
5. AWS STS verifies the request for the role ARN, External ID if any and if it is from the trusted resource matching the roles’s trust policy and
6. AWS STS upon successful verification returns temporary credentials
7. Temporary credentials allow the user to access the resources of the Trusting account
8. When the user exits the role, the user’s permissions revert to the original permissions held before switching to the role

#### **External ID and Confused Deputy Problem**

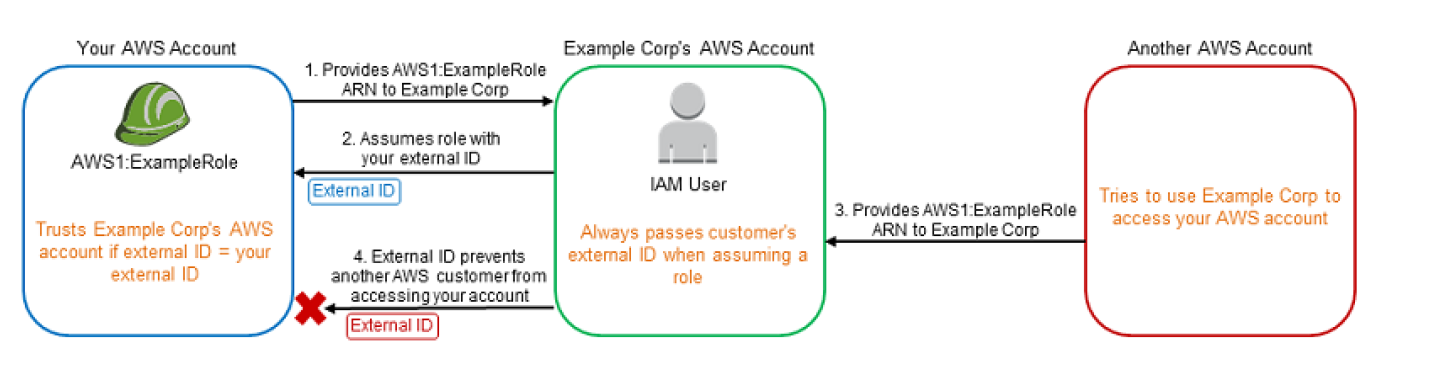
* External ID allows the user that is assuming the role to assert the circumstances in which they are operating.
* External ID provides a way for the account owner to permit the role to be assumed only under specific circumstances and prevents an unauthorized customer from gaining access to your resources
* Primary function of the external ID is to address and prevent the “confused deputy” problem.

##### **Confused Deputy Problem**



1. Example Corp’s AWS Account provides the services (access, analyze and process data and provide back reports) to multiple different AWS accounts
2. Preferred mechanism is to have each AWS account customer define a Role which Example Corp’s AWS Account users can assume and act upon
3. You provide Example Corp’s AWS Account access to your AWS account through Role and providing Role ARN
4. Example Corp when working on your account assumes the IAM role and provides the ARN with the request
5. As Example Corp is already trusted by your account it will received the temporary security credentials and gain access to your resources
6. If an other AWS account is able to know or guess your ARN (Role with Account ID), it can provide the same to Example Corp
7. Example Corp’s would use the ARN (belonging to your AWS account) to process the data but would provide the same data to the other AWS account
8. This form of privilege escalation is known as the confused deputy problem

**Address Confused Deputy Problem using External ID**



1. Using External ID, Example Corp’s generates a unique External ID for each of its Customer which is known only to them and is kept secret
2. Example Corp provides you an External ID which needs to added as a condition while defining the trust policy
3. You provide Example Corp’s AWS Account access to your AWS account through Role and providing Role ARN
4. Example Corp when working on your account uses the IAM role and provides the ARN along with the External ID and as it is already trusted would be able to gain access
5. Other AWS account registered with Example Corp would have a Unique External ID assigned to it
6. If the Other AWS account is able to know or guess your ARN (Role with Account ID), it can provide the same to Example Corp
7. Example Corp’s would request access to your Account using the ARN (belonging to your AWS account) but with the External ID belonging to Other AWS account as the request was made on its behalf
8. As the External ID provided by Example Corp does not match the condition defined in the Role trust policy, the authentication would fail and hence denied access

Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated with medium confidence

Diagram

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Timeline

Description automatically generated with medium confidence

Diagram

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Diagram

Description automatically generated

Text

Description automatically generated

Diagram

Description automatically generated with medium confidence

## Text Description automatically generated

Graphical user interface, text, chat or text message

Description automatically generated

Text

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

## Identity Providers and Federation

Refer to My Blog Post about [IAM Role – Identity Providers and Federation](https://jayendrapatil.com/iam-role-identity-providers-federation/)

## AWS Certification Exam Practice Questions

1. A company is building software on AWS that requires access to various AWS services. Which configuration should be used to ensure that AWS credentials (i.e., Access Key ID/Secret Access Key combination) are not compromised?
   1. Enable Multi-Factor Authentication for your AWS root account.
   2. **Assign an IAM role to the Amazon EC2 instance.**
   3. Store the AWS Access Key ID/Secret Access Key combination in software comments.
   4. Assign an IAM user to the Amazon EC2 Instance.
2. A company is preparing to give AWS Management Console access to developers. Company policy mandate’s identity federation and role-based access control. Roles are currently assigned using groups in the corporate Active Directory. What combination of the following will give developers access to the AWS console? (Select 2) Choose 2 answers
   1. **AWS Directory Service AD Connector**
   2. AWS Directory Service Simple AD
   3. AWS Identity and Access Management groups
   4. **AWS identity and Access Management roles**
   5. AWS identity and Access Management users
3. A customer needs corporate IT governance and cost oversight of all AWS resources consumed by its divisions. The divisions want to maintain administrative control of the discrete AWS resources they consume and keep those resources separate from the resources of other divisions. Which of the following options, when used together will support the autonomy/control of divisions while enabling corporate IT to maintain governance and cost oversight? Choose 2 answers
   1. Use AWS Consolidated Billing and disable AWS root account access for the child accounts.
   2. **Enable IAM cross-account access for all corporate IT administrators in each child account.** (Provides IT governance)
   3. Create separate VPCs for each division within the corporate IT AWS account.
   4. **Use AWS Consolidated Billing to link the divisions’ accounts to a parent corporate account.**(Will provide cost oversight)
   5. Write all child AWS CloudTrail and Amazon CloudWatch logs to each child account’s Amazon S3 ‘Log’ bucket.
4. Which of the following items are required to allow an application deployed on an EC2 instance to write data to a DynamoDB table? Assume that no security keys are allowed to be stored on the EC2 instance. (Choose 2 answers)
   1. **Create an IAM Role that allows write access to the DynamoDB table**
   2. **Add an IAM Role to a running EC2 instance.**(With latest enhancement from AWS, IAM role can be assigned to a running EC2 instance)
   3. Create an IAM User that allows write access to the DynamoDB table.
   4. Add an IAM User to a running EC2 instance.
   5. Launch an EC2 Instance with the IAM Role included in the launch configuration (This was the correct answer before, as AWS did not allow IAM role to be added to an existing instance)
5. You are looking to migrate your Development (Dev) and Test environments to AWS. You have decided to use separate AWS accounts to host each environment. You plan to link each accounts bill to a Master AWS account using Consolidated Billing. To make sure you Keep within budget you would like to implement a way for administrators in the Master account to have access to stop, delete and/or terminate resources in both the Dev and Test accounts. Identify which option will allow you to achieve this goal. **[PROFESSIONAL]**
   1. Create IAM users in the Master account with full Admin permissions. Create cross-account roles in the Dev and Test accounts that grant the Master account access to the resources in the account by inheriting permissions from the Master account.
   2. Create IAM users and a cross-account role in the Master account that grants full Admin permissions to the Dev and Test accounts.
   3. **Create IAM users in the Master account Create cross-account roles in the Dev and Test accounts that have full Admin permissions and grant the Master account access**
   4. Link the accounts using Consolidated Billing. This will give IAM users in the Master account access to resources in the Dev and Test accounts
6. You have an application running on an EC2 Instance which will allow users to download flies from a private S3 bucket using a pre-assigned URL. Before generating the URL the application should verify the existence of the file in S3. How should the application use AWS credentials to access the S3 bucket securely? **[PROFESSIONAL]**
   1. Use the AWS account access Keys the application retrieves the credentials from the source code of the application.
   2. Create a IAM user for the application with permissions that allow list access to the S3 bucket launch the instance as the IAM user and retrieve the IAM user’s credentials from the EC2 instance user data.
   3. **Create an IAM role for EC2 that allows list access to objects in the S3 bucket. Launch the instance with the role, and retrieve the role’s credentials from the EC2 Instance metadata**
   4. Create an IAM user for the application with permissions that allow list access to the S3 bucket. The application retrieves the IAM user credentials from a temporary directory with permissions that allow read access only to the application user.
7. An administrator is using Amazon CloudFormation to deploy a three tier web application that consists of a web tier and application tier that will utilize Amazon DynamoDB for storage when creating the CloudFormation template which of the following would allow the application instance access to the DynamoDB tables without exposing API credentials?  **[PROFESSIONAL]**
   1. Create an Identity and Access Management Role that has the required permissions to read and write from the required DynamoDB table and associate the Role to the application instances by referencing an instance profile.
   2. Use the Parameter section in the Cloud Formation template to nave the user input Access and Secret Keys from an already created IAM user that has me permissions required to read and write from the required DynamoDB table.
   3. **Create an Identity and Access Management Role that has the required permissions to read and write from the required DynamoDB table and reference the Role in the instance profile property of the application instance.**
   4. Create an identity and Access Management user in the CloudFormation template that has permissions to read and write from the required DynamoDB table, use the GetAtt function to retrieve the Access and secret keys and pass them to the application instance through user-data.
8. An enterprise wants to use a third-party SaaS application. The SaaS application needs to have access to issue several API commands to discover Amazon EC2 resources running within the enterprise’s account. The enterprise has internal security policies that require any outside access to their environment must conform to the principles of least privilege and there must be controls in place to ensure that the credentials used by the SaaS vendor cannot be used by any other third party. Which of the following would meet all of these conditions? **[PROFESSIONAL]**
   1. From the AWS Management Console, navigate to the Security Credentials page and retrieve the access and secret key for your account.
   2. Create an IAM user within the enterprise account assign a user policy to the IAM user that allows only the actions required by the SaaS application create a new access and secret key for the user and provide these credentials to the SaaS provider.
   3. **Create an IAM role for cross-account access allows the SaaS provider’s account to assume the role and assign it a policy that allows only the actions required by the SaaS application.**
   4. Create an IAM role for EC2 instances, assign it a policy mat allows only the actions required tor the SaaS application to work, provide the role ARM to the SaaS provider to use when launching their application instances.
9. A user has created an application which will be hosted on EC2. The application makes calls to DynamoDB to fetch certain data. The application is using the DynamoDB SDK to connect with from the EC2 instance. Which of the below mentioned statements is true with respect to the best practice for security in this scenario?
   1. **The user should attach an IAM role with DynamoDB access to the EC2 instance**
   2. The user should create an IAM user with DynamoDB access and use its credentials within the application to connect with DynamoDB
   3. The user should create an IAM role, which has EC2 access so that it will allow deploying the application
   4. The user should create an IAM user with DynamoDB and EC2 access. Attach the user with the application so that it does not use the root account credentials
10. A customer is in the process of deploying multiple applications to AWS that are owned and operated by different development teams. Each development team maintains the authorization of its users independently from other teams. The customer’s information security team would like to be able to delegate user authorization to the individual development teams but independently apply restrictions to the users permissions based on factors such as the users device and location. For example, the information security team would like to grant read-only permissions to a user who is defined by the development team as read/write whenever the user is authenticating from outside the corporate network. What steps can the information security team take to implement this capability? **[PROFESSIONAL]**
    1. Operate an authentication service that generates AWS STS tokens with IAM policies from application-defined IAM roles. (no user separation, will just help generate temporary tokens)
    2. **Add additional IAM policies to the application IAM roles that deny user privileges based on information security policy.**(Different policy with deny rules based on location, device and more restrictive wins)
    3. Configure IAM policies that restrict modification of the application IAM roles only to the information security team. (Authorization should still be in developers control)
    4. Enable federation with the internal LDAP directory and grant the application teams permissions to modify users.
11. You are creating an Auto Scaling group whose Instances need to insert a custom metric into CloudWatch. Which method would be the best way to authenticate your CloudWatch PUT request?
    1. **Create an IAM role with the Put MetricData permission and modify the Auto Scaling launch configuration to launch instances in that role**
    2. Create an IAM user with the PutMetricData permission and modify the Auto Scaling launch configuration to inject the users credentials into the instance User Data
    3. Modify the appropriate Cloud Watch metric policies to allow the Put MetricData permission to instances from the Auto Scaling group
    4. Create an IAM user with the PutMetricData permission and put the credentials in a private repository and have applications on the server pull the credentials as needed