KPLABS Course

AWS Certified Security - Specialty 2020

Domain 4 - Identity & Access Management

ISSUED BY
Zeal Vora

REPRESENTATIVE
instructors@kplabs.in

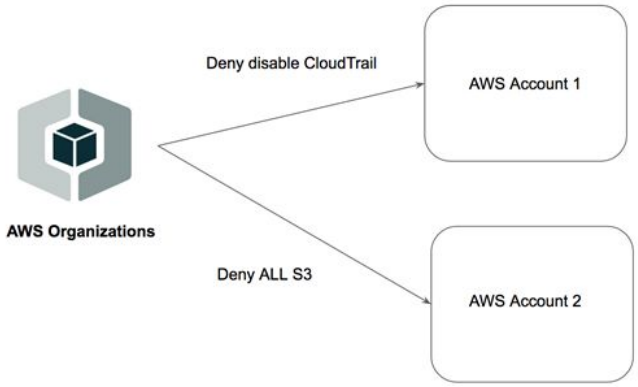
Module 1: Understanding AWS Organizations

AWS offers centralized policy-based management as well as the feature of consolidated billing for multiple AWS accounts through the feature of AWS Organizations.

There are two primary features of AWS Organizations that we can use:

- Consolidated Billing Only
- All Features

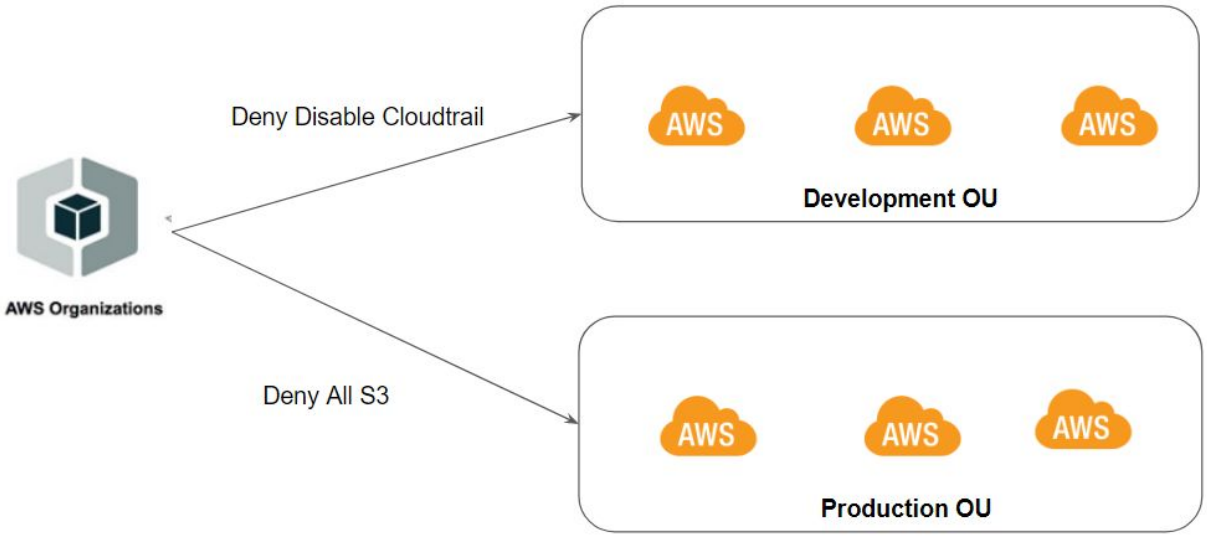
AWS Organization can also be used to set the Service Control Policies (SCP) which can further restrict access.

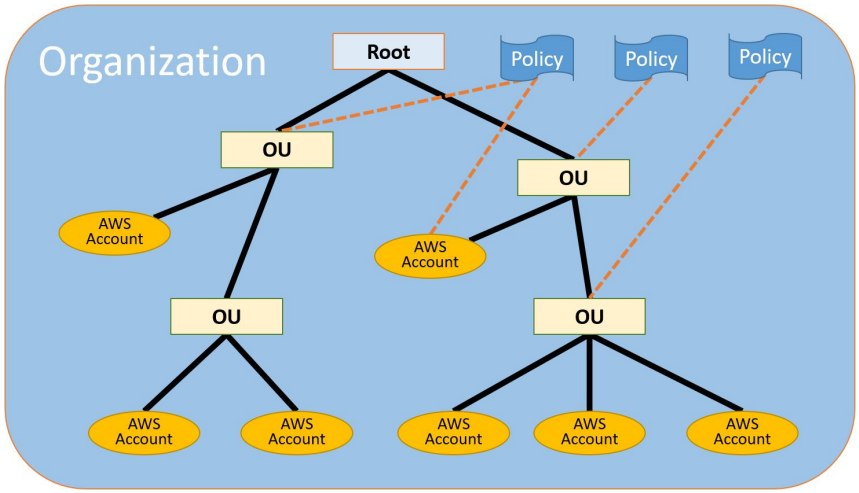


1.1 OU in AWS Organization

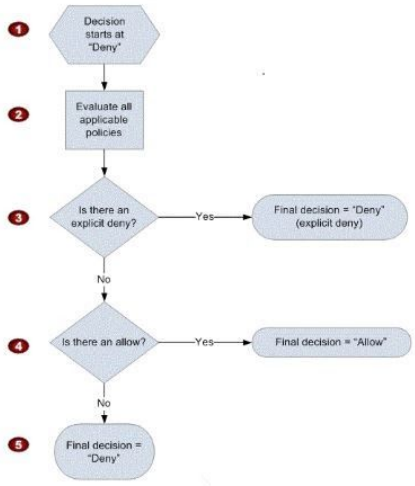
Multiple AWS accounts can be mapped to a specific Organizational Unit based on which policies can be applied.

The following two diagrams can provide a better understanding of the same.


Deny Disable Cloudtrail
AWS Organizations
Deny All S3
AWS
AWS
AWS
Development OU
AWS
AWS
AWS
Production OU


Organization
Root
Policy
Policy
Policy
OU
OU
AWS Account
AWS Account
OU
OU
AWS Account
AWS Account
AWS Account
AWS Account
AWS Account

Module 2: IAM Policy Evaluation Logic



1) Decision starts with assumption that the request will be denied.
2) Then, all the attached policies are evaluated.
3) Code will look if there is any explicit deny in the policy.
4) If explicit deny is not found, code will look for allowed instruction and if yes then decision is allowed.
5) If no allow is found, decision is deny.

2.1 Explicit Deny vs Default Deny

A request will be denied by default if there is no allow policy present for the resource.

Example:

- If a user has EC2 ReadOnlyAccess and tries to open the S3 console, it will be denied.

A request will be denied irrespective of full access if it is explicitly denied in the policy

Example:

- Deny user from accessing bucket 05.

Module 3: Identity vs Resource-Based Policies

3.1 Overview of Identity-Based Policies

Identity-based policies are attached to an IAM user, group, or role.

These policies define what an Identity can do.

Example:

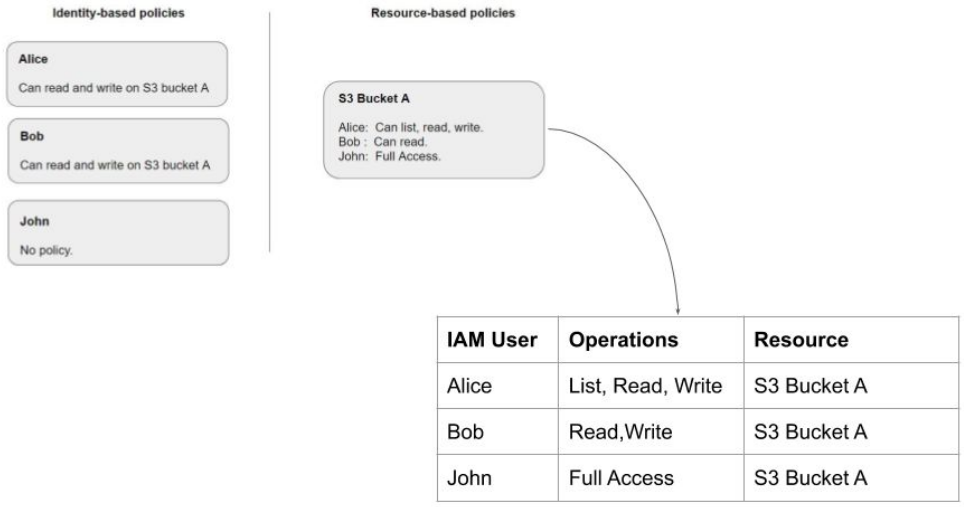
- We attach IAM policy to a user name John and define that John can start and stop EC2 instances that belong to the development environment.

3.2 Overview of Resource-Based Policies

Resource-based policies are attached to a resource.

We attach these policies directly to a resource like S3 bucket, SQS Queue, KMS keys.

With resource-based policies, you can specify who has access to the resource and what actions they can perform on it.



IAM User	Operations	Resource
Alice	List, Read, Write	S3 Bucket A
Bob	Read,Write	S3 Bucket A
John	Full Access	S3 Bucket A

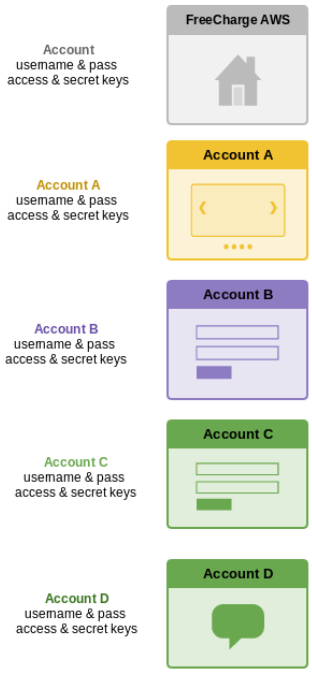
Module 4: Delegation - Cross-Account IAM Role

Initially, there used to be a single AWS account and everything was simple.

Every user would have a single set of:

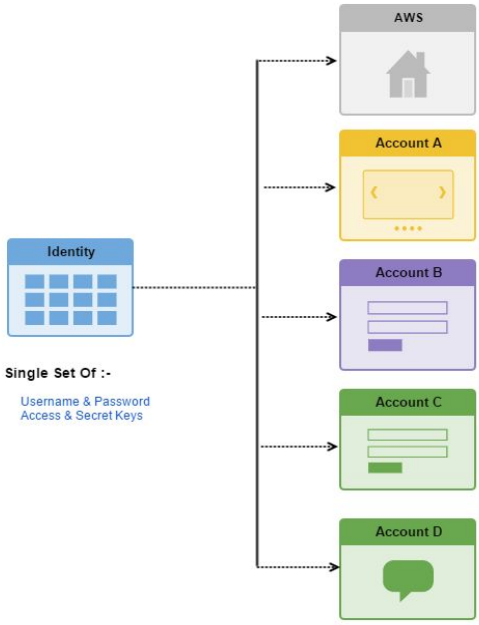
- Username and password
- Access and Secret Keys

In modern architecture, there are multiple AWS accounts. Every account has multiple sets of access / secret keys and usernames/passwords. This is difficult to work with.

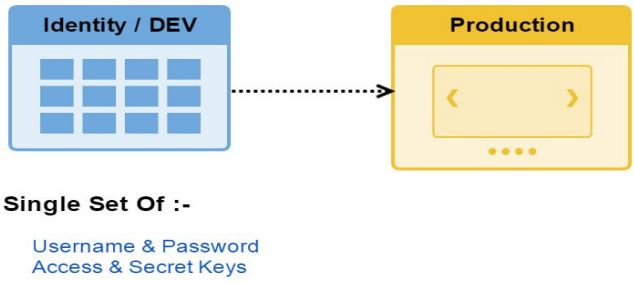


To deal with such a scenario, the architecture of Identity Account plays an important role.

With Identity Account, users can have a single set of credentials and keys.



With the Identity Account architecture, all the Identities are stored in the Identity Account. A user would log in to the Identity Account and perform AssumeRole operation on the Production Account as shown in the below diagram.



There are three major steps that we need to perform as part of Cross-Account IAM Roles:

i) Create a user in Account A.

ii) Create a Cross-Account role in Account B.

iii) Allow User to switch to Account-B Role.

Following is the sample JSON code for Cross-Account IAM Role Policy used in the video:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
   "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"
  }
}
```

Module 4: IAM Role

IAM Role contains a set of policies and any entity assuming that role will be able to have permissions mentioned in the role.

A role can be used by :

- An IAM user
- A web service offered by the AWS such as EC2
- An external user authenticated by external IdP service compatible with SAML etc.

Module 5: IAM Policy - Version Element

A Version element defines the version of the policy language.

The overall syntax rules can differ based on the version of the policy language being used.

Syntax:

"Version": "2012-10-17"

There are two supported Version Element Values:

Version	Description
2012-10-17	• Current version of the policy element. • It is recommended to always define version element to 2012-10-17
2008-10-17	• Earlier version of the policy element. • Avoid defining this version in any new policies. • If you do not include “Version Element”, it defaults to this.

Version Element != Policy Versions.

Version Element specifies the language of the policy.

Policy Version is created when we change the customer managed policy.

Module 6: IAM Policy Variables

Policy Variables are used when you don't know the exact value of a resource or condition key when you write the policy.

Example:

"arn:aws:iam::888913816489:user/${aws:username}"

Module 7: IAM - Principal and NotPrincipal Element

A Principal element is used to specify things like IAM user, federated user, IAM role, AWS account, AWS service, or other principal entity that is allowed or denied access to a resource.

You cannot use the Principal element in an IAM identity-based policy.

"Principal": { "AWS": "arn:aws:iam::123456789012:root" }

Principal Field can include various aspects like:

- IAM User
- IAM Role
- IAM Service Name ("datapipeline.amazonaws.com")
- Federated Users

When you use NotPrincipal in the same policy statement as "Effect": "Deny", the actions specified in the policy statement are explicitly denied to all principals except for the ones specified.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Deny",
        "NotPrincipal": {"AWS": [
            "arn:aws:iam::444455556666:user/Bob",
            "arn:aws:iam::444455556666:Alice"
        ]},
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::BUCKETNAME",
            "arn:aws:s3:::BUCKETNAME/*"
        ]
    }]
}
```

Module 8: IAM - Condition Element

The Condition element lets you specify conditions for when a policy is in effect.

We build it by making use of condition operators like (equal, less than, etc).

Sample policy:

Allow full access to User Alice on EC2 Resource {ONLY WHEN CONDITION IS MET

```
"Condition": {
  "DateGreaterThan" : {
     "aws:CurrentTime" : "2013-12-15T12:00:00Z"
   }
}
```

There are multiple condition operators available which you can make use of.

Some of these include:

- String Condition Operators
- Numeric Condition Operators
- Date Condition Operators
- Boolean Condition Operators
- Binary Condition Operators
- IP Address Condition Operators
- ARN Condition Operators

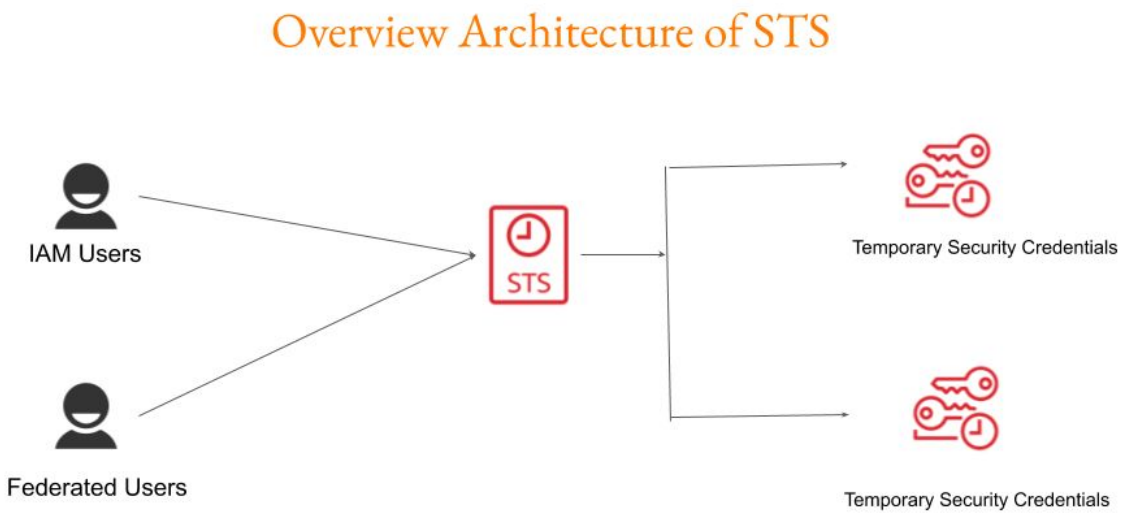
Module 9: AWS STS

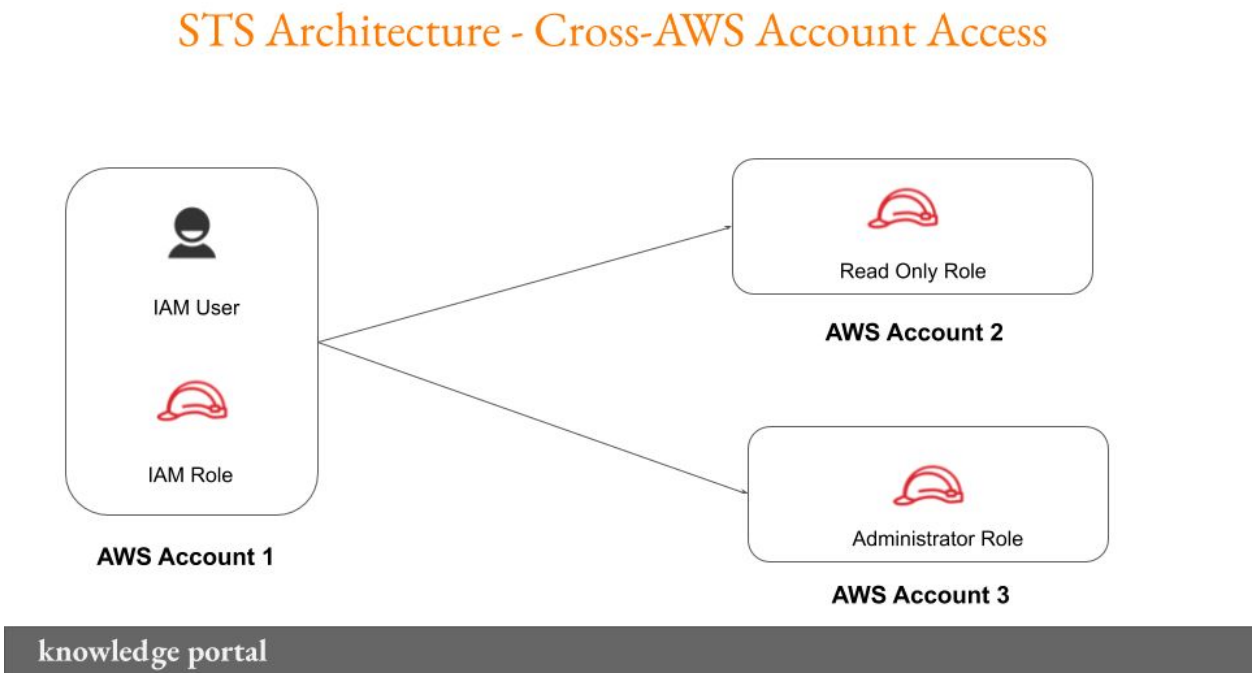
The AWS STS is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

Temporary security credentials are short term and expire after a certain duration.

Since they have a limited lifetime, the key rotation is no longer explicitly needed.

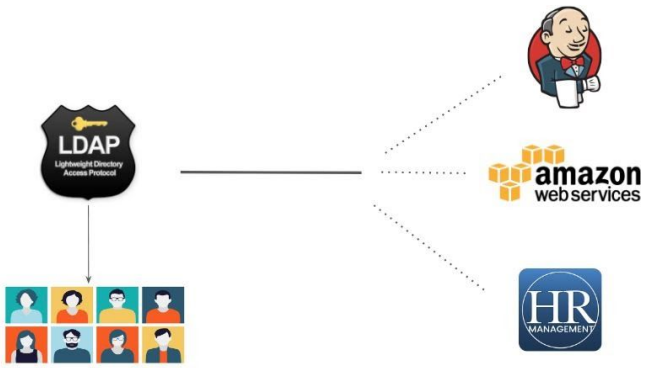

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAJOTOADWSDZD53Z7VS:temp",
        "Arn": "arn:aws:sts::836802967410:assumed-role/CA-EC2RO/temp"
    },
    "Credentials": {
        "SecretAccessKey": "LKtyaWrhxGnBNP3tx7dMK2nv0H1VdwMP1RVP5Sob",
        "SessionToken": "FQoDYXdzEMj//////////wEaDHwScBw1Hmr5eGqKXyLHAdeXEJZ0oSuJxFd/PGtU
Z5F3XhjgIawg7ytJXXWRgpyvaq9eMKNfUqmiDca/NM+FLwqy5iek5VKPGkPut+/pAzOWOH3ddVmcuHsJowHxaDGHa
d6S2IyhyMfAF9bk9FQjMfHNtl/oDl74KvkAV6xAE4Q0cPZ4sDGes130Im4r5TulKT/I2qvg0w/LVRjraJ8UBnopMu
gU=",
        "Expiration": "2017-09-20T23:36:41Z",
        "AccessKeyId": "ASIAJWPD367QI4GHCNAQ"
    }
```



Module 10: Federation

Federation allows external identities (Federated Users) to have secure access in your AWS account without having to create any IAM users.



There are various solutions available which can store users centrally:-

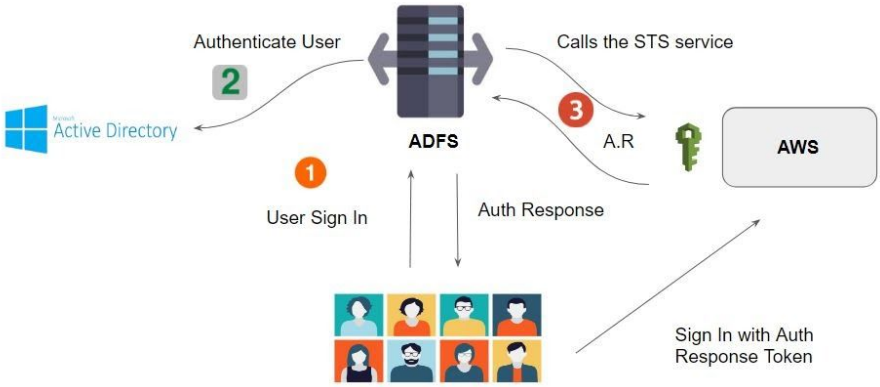
Microsoft Active Directory
RedHat Identity Management / freeIPA

Important Steps to Remember

1. User logs in with username & Password.
2. These credentials are given to the Identity Broker.
3. Identity Broker validates it against the AD.
4. If credentials are valid, the Broker will contact the STS token service.
5. STS will share the following 4 things:-

    Access Key + Secret Key + Token + Duration

Users can now use to login to AWS Console or CLI.

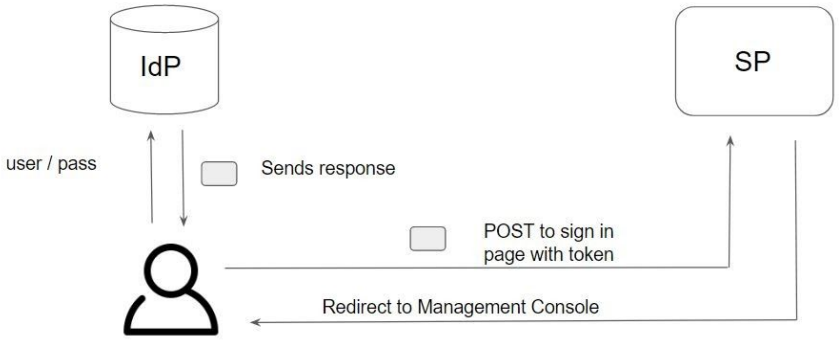


Module 11: SAML

SAML stands for Security Assertion Markup Language.

It is a secure XML based communication mechanism for communicating identities across organizations.

SAML eliminates the need to maintain multiple authentication credentials, such as passwords in multiple locations.



The flow gets initiated when the user opens the IDP URL and enters the username and password and selects the appropriate application.

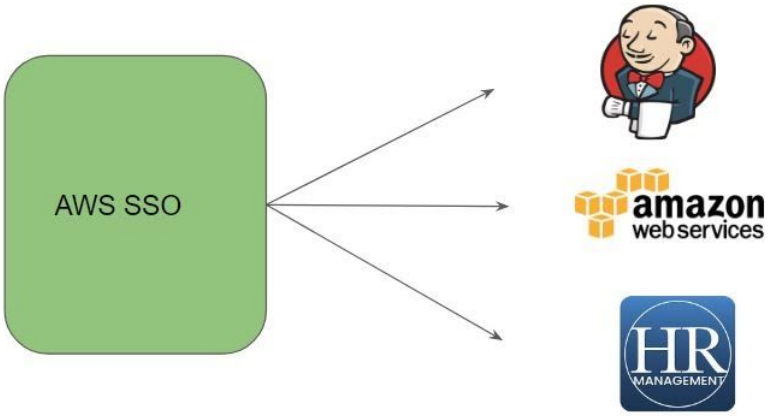
IdP will validate the credentials and associated permissions and then the user receives SAML assertion from the IdP as part of the response.

User does a POST of that SAML assertion to the SAAS sign in page and SP will validate that assertion.

On validation, SP will construct relevant temporary credentials, and constructs sign-in URL for the console and sends to the user.

Module 12: AWS Single Sign-On

AWS Single Sign-On (SSO) makes it easy to centrally manage access to multiple AWS accounts and business applications and provide users with single sign-on access to all their assigned accounts and applications from one place.



12.1 SSO with AWS CLI

AWS CLI integrates with the SSO.

SSO users can authenticate via CLI, and they will be able to perform the CLI operations without having to add keys in their ~/.aws/credentials file.

```
bash-4.2# aws sso login --profile Production-Account
Attempting to automatically open the SSO authorization page in your default browser.
If the browser does not open or you wish to use a different device to authorize this request, open the following URL:

https://device.sso.ap-southeast-1.amazonaws.com/

Then enter the code:
```

Module 13: AWS Cognito

Amazon Cognito provides authentication, authorization, and user management service for your web and mobile apps.

Let’s understand this with a use-case:

Andrew is a mobile developer in a start-up organization. They have begun with a mobile wallet system, and there are specific requirements as follows:

- Users should be able to sign-in with social network platforms like FB, Twitter, G+.
- There should be a post-sign-up process (one-time password) for verification.

- Account recovery feature should be present.
- Guest access must be allowed for users to see the app.

At a high level, there are two major features under AWS Cognito

i) User Pools
ii) Identity Pools

Cognito user pool takes care of the entire authentication, authorization process.

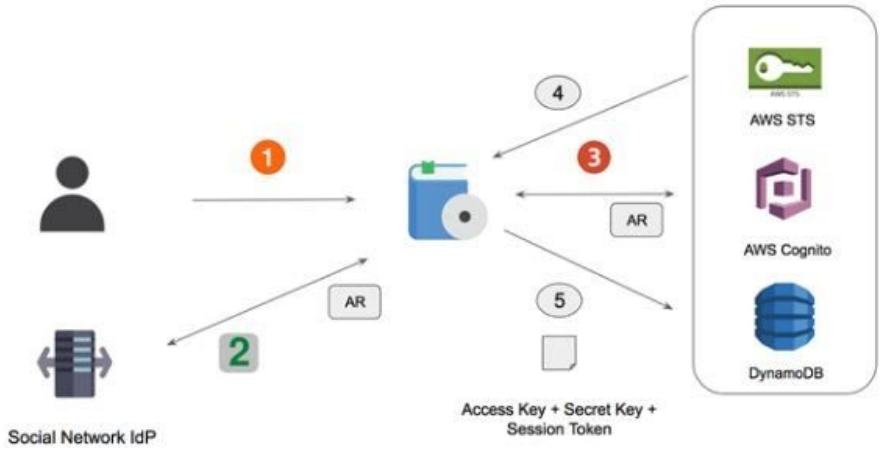
The identity pool provides the functionality of federation for users in user pools.

Cognito Identity pools also referred to as AWS Cognito Federated Identities allow developers to authorize the users of the application to use various AWS services.

Use-Case:

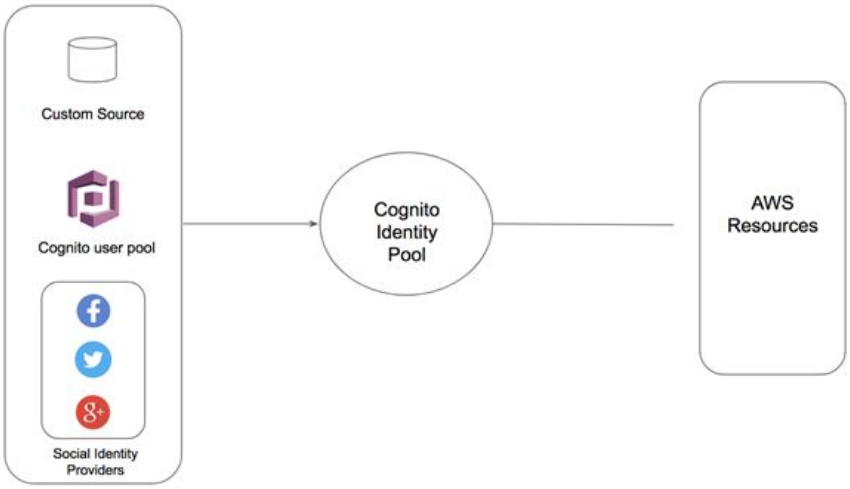
> We have a quiz based mobile application. At the end of the quiz, the user's results should be stored in the DynamoDB table.
>
> If we hard-code the access/secret keys, chances of reverse engineering are high.

The working of Cognito Identity Pool can be described in the following diagram:



13.1 User Pool vs Identity Pool

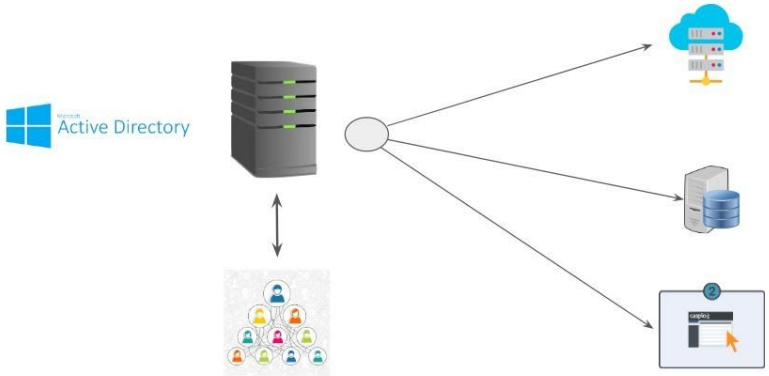
The Cognito Identity pool then takes these identities and federates them and then can give secure access to the AWS services regardless of where the user comes from.



Module 14: Active Directory

Active Directory is one of the most popular directory service developed by Microsoft.

The server running the Active Directory service is called the domain computer and it can authenticate and authorize the users and computers which are associated with it.



Module 15: Active Directory Service

AWS Directory Service is a managed service based on the cloud that allows us to create directories and let AWS experts handle and manage the other parts like high availability, monitoring, backups, recovery, and others.

There are three important components :

- Active Directory Service with Microsoft Active Directory
- Simple AD
- AD Connector

15.1 Active Directory Service with Microsoft Active Directory

AWS Directory Service for Microsoft Active Directory is powered by an actual Microsoft Windows Server Active Directory (AD) in the AWS Cloud.

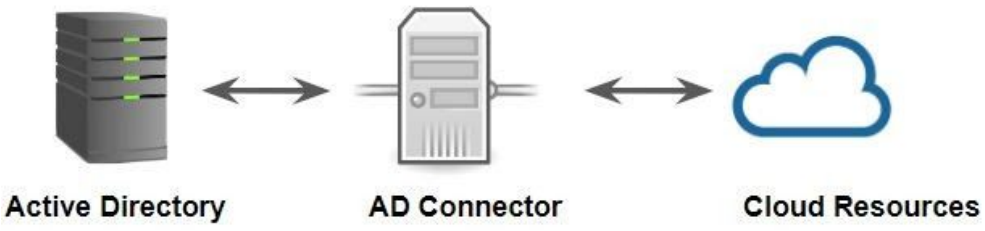
There are two types:

- Standard Edition -- For small and midsize (up to 5000 users)
- Enterprise Edition -- For larger deployments.

15.2 AD Connector

It is a proxy service that provides an easy way to connect applications in the cloud to your existing on-premise Microsoft AD.

When users log in to the applications, AD Connector forwards sign-in requests to your on-premises Active Directory domain controllers for authentication.



15.3 Simple AD

Simple AD is a Microsoft Active Directory–compatible directory from AWS Directory Service that is powered by Samba 4.

Simple AD supports basic Active Directory features such as user accounts, group memberships, joining a Linux domain or Windows-based EC2 instances, Kerberos-based SSO, and group policies. AWS provides monitoring, daily snapshots, and recovery as part of the service.

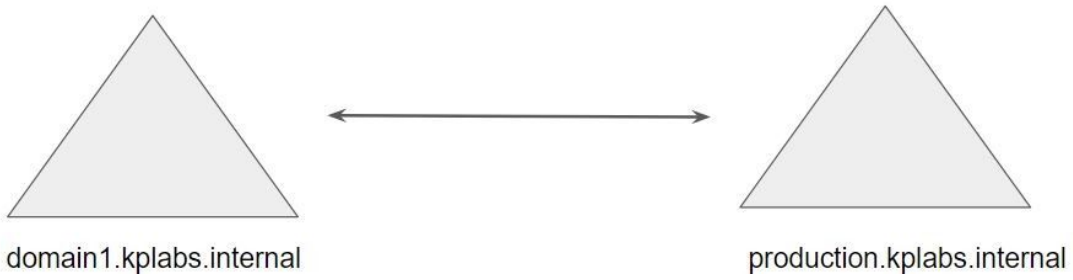
Simple AD does not support trust relationships, DNS dynamic update, schema extensions, multi-factor authentication, communication over LDAPS, PowerShell AD cmdlets, or FSMO role transfer.

Module 16: Active Directory Trusts

In AD, domain to domain communication can occur through Trusts.

An AD DS trust is a secured, authentication communication channel between entities, such as AD DS domains.

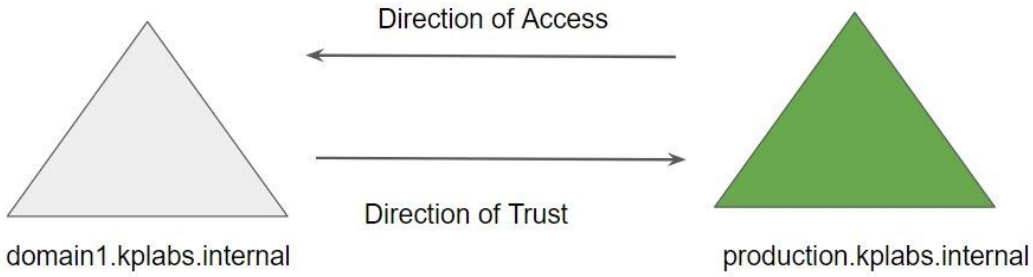
Trusts enable you to grant access to resources to users, groups, and computers across entities



16.1 Direction of Trust

Trust can either be one-way or two-way.
In a two-way trust, the domain from either side can access the other side.

In the following diagram, we have one-way trust.



16.2 Migrating AD Aware Workloads

If you already have an AD infrastructure and want to use it when migrating AD-aware workloads to the AWS Cloud, you can use AD trusts to connect AWS Microsoft AD (Standard Edition) to your existing AD.

This means your users can access AD-aware and AWS applications with their on-premises AD credentials, without needing you to synchronize users, groups, or passwords.

Module 17: S3 Bucket Policies

One of the limitations of IAM is that it is generally restricted to the principles like the user/group/roles within AWS account.

When we speak about the S3 bucket which is used by external entities, we need much more granular permission.

S3 Bucket policies are attached directly to the S3 buckets.

Module 18: Canned ACL

Every bucket and its objects have an ACL associated with them.

When a request is received, AWS S3 will check against the attached ACL to either allow or block access to that specific object.

When we create a bucket or an object, AWS S3 by default will grant the resource owner full control over the resource.

AWS S3 supports a set of pre-defined grants, known as Canned ACL's.

Each canned ACL has a predefined set of permission associated with them.

This canned ACL can be specified in the request using x-amz-acl header.

ACL Name	Description
Private	Owner gets FULL_CONTROL. No one else will have access rights (default)
Public-read	Owner has FULL_CONTROL. All others will have public read permission.
Bucket-owner-read	Owner of the object has FULL_CONTROL. Bucket owner will get read permissions.
Bucket-owner-full-control	Both the object owner and the bucket owner get FULL_CONTROL over the object.

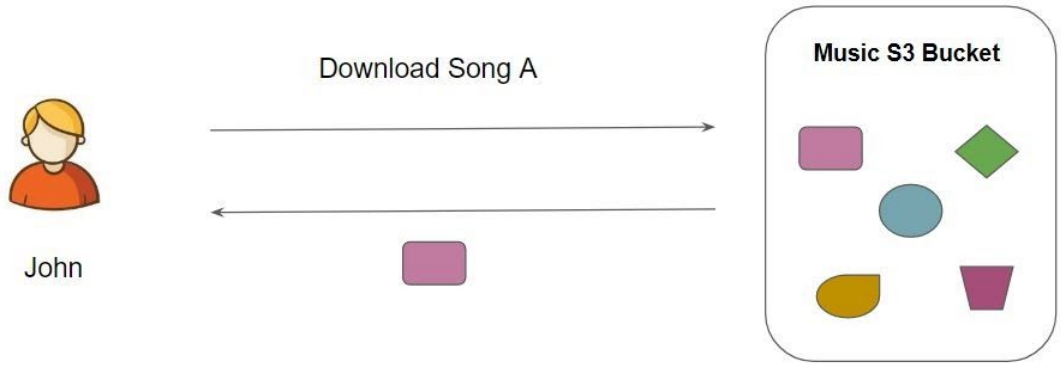
Module 18: Presigned URLs

All objects in S3 are ‘Private’ by default.

However, Object owner can optionally share objects with others by creating a pre-signed URL to grant time-limited permission to download the object.

Achieving the Use Case:-

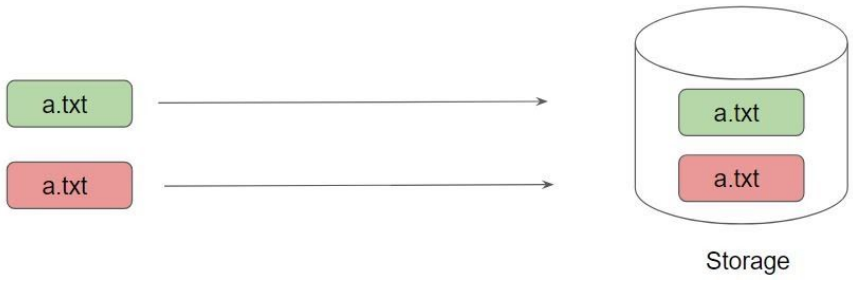
After a user purchases a song and requests to Download, the application should generate a pre-signed URL that will allow the ‘MP3’ file stored in S3 to be downloaded by the user.



Module 19: S3 Versioning

Versioning allows users to keep multiple variants of an object in the same S3 bucket.

You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket.



Once you version enable a bucket, it can never return to an unversioned state. You can, however, suspend versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket.

Once you version enable a bucket, it can never return to an unversioned state. You can, however, suspend versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket.

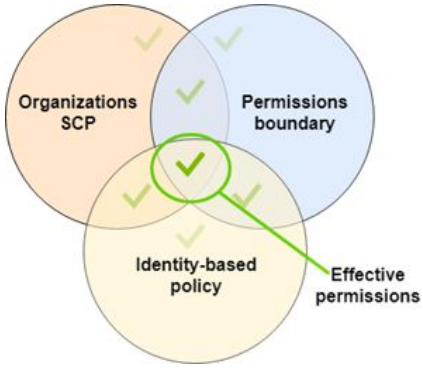
Module 20: IAM Permission Boundary

A permissions boundary is an advanced feature in which you use a managed policy to set the maximum permissions that an identity-based policy can grant to an IAM entity.

When you set a permissions boundary for an entity, the entity can perform only the actions that are allowed by both its identity-based policies and its permissions boundaries.

The effective permissions for an entity are the permissions that are granted by all the policies associated with the user/role/account.

Within an AWS account, the permissions for an entity can be affected by identity-based policies, resource-based policies, permissions boundaries, Organizations SCPs, or session policies.



Note: For Troubleshooting IAM Policy, please refer to the course videos directly.