Session 1 : Identity and Access Management Introduction

Identity & Access Management :

• IAM refers to a framework or policies and technologies for ensuring that the proper people in an organisation have the appropriate access to technology resources

(OR)

AWS IAM is a web service that helps us securely control access to AWS resources.We use IAM to conrtol who is authenticated(signed-in) and authorized(has permission) to use resources

• When we first create an AWS account,we begin with a single signin identity that has complete access to all AWS services and resources in the account
• This identity is called,the AWS account 'Root User' and is accessed by signing in with the email address and password that we used to create the account
• AWS strongly recommends that we do not use the root user for everyday tasks,even the administrative ones
• Use other IAM user accounts to manage the administrative task of our account for securely lock away the root user credentials and use them to perform only a few account and service management tasks
• IAM user limit is 5000 for AWS account.We can add upto 10 users at one time
• We are also limited to 300 groups per AWS account
• We are limited to 1000 IAM roles under AWS account
• Default limits of managed policies attached to an IAM role and IAM user is 10
• IAM user can be a member of 10 groups
• We can assign two access keys(MAX) to an IAM user

Features of IAM :

1.Shared access to our AWS Account :
• We can grant other people permission to administer and use resources in our aws account without having to share our access credentials(password or access key)

2.Granular Permissions:
• We can grant diff permission to diff people for diff resources
• For instance,we can allow same users complete access to EC2,S3,Dynamodb,Redshift while for others,we can allow read only access to just some S3 buckets,or permission to administer just some ec2 instances or to access our billing information but nothing else

3.Secure access to AWS resources for applications that run on Amazon EC2 :
• We can use IAM features to securely give applications that run on ec2 instances the credentials that they need in order to access other aws resources.
• EX : Includes S3 buckets and RDS or DynamoDb databases

4.Multifactor Authentication(MFA) :
• We can add two factor authentication to our account and to individual users to extra security.We can use physical hardware or virtual MFA(Eg : Goog;le Authenticator)

5.Identity Federation :
• We can allow users who already have passwords elsewhere
• EX : In our corporate network or with an internet identity provider to get temporary access to our AWS account

6.Identity information for Assurance :
• If we use AWS cloudtrail,we receive log records that include information about those who made request for resources in our account.That information is based on IAM identities

7.PCI-DSS Compliance :
• IAM support the processing,storage and transmission of credit card by a merchant or service provider,and has been validated as being compliant with payment card Industries(PCI) and Data Security Standard(DSS)

8.Eventually Consistent:
• If a request to change some data is sucessful, the change is comitted and safely stored.However the change must be replicated across IAM,which can take sometime
• IAM achieves high availabilty by replicating data across multiple servers within AWS data centre around the world

• Free to use : AWS IAM is a feature of our AWS account offered at no additional cost
• We will be charged only for the use of other AWS products by our IAM users

Session 2 : IAM Terms-Principal,Request,Authentication,Authorization

Principal :

• A principal is a person or application that can make a request for an action or operation on an AWS resources
• Our administrative IAM user is our first principal
• We can allow users and services to assume a role
• IAM users,roles,federated users and applications are all AWS principals
• We can support federated users or programmetic access to allow an application to access our AWS account

Request :

• When a principal tries to use the AWS resources management console,that aws api,or aws cli,that principal sends a request to aws.
• The request include the following information : Actions,Resources,Principal,Environment data,Resources Data

1.Actions : That the principal want to perform
2. Resources : Upon which the actions are performed
3.Principal Information : including the environment from which the request was made
4.Request Context : Before AWS can evaluate and authorize a request ,aws gathers the request information and principal(the requester),which is determined based on the authorization data.This includes the aggregate permissions that is associated with that principal
5.Environment Data : Such as IP address,user agent,SSL enabled status or the time of day
6.Resource Data : Data related to the resource that is being requested

Authentication :

• A principal sending a request must be authenticated(signed in to aws) to send a request to AWS
• Some aws services like AWS S3 allow request from anonymous users,they are exception to the role
• To authenticate from the console as a root user,we must signin with our user name and password
• To authenticate from the API to CLI,we must provide our access key and secret key
• We might also be required to provide additional security information like MFA

Authorization :

• To authorize request,IAM allows values from the request context to check for matching policies and determie whether to allow or deny the request
• IAM policies are stored in IAM as JSON documents and specify the permissions that are allowed or denied
• *User(identity) based policies :* Specify permission allowed/denied for principal
• *Resource based Policies :* Specify permission allowed/denied for resources .Popular for granting cross account permissions
• IAM checks each policy that matches the context of our request

• If a single policy includes a denied action,IAM denies the entire request and stop evaluating.This is called Explicit Deny
• The evaluation logic follows these rules :
□ By default,all requests are denied(implicit deny)
□ An explict allow overrides this default
□ An explicit deny overrides any allows

Session 3 : IAM Terms-Actions,Resources

• We can create a new IAM policy in the AWS management console using one of the following ways :
1.JSON : We can create our own JSON syntax
2.Visual Editor : We can constuct a new policy from scratch in the visual editor.If we use the Visuual editor,we do not have to understood JSON syntax
3.Import : We can import a managed policy within our aws account and then edit the policy to customize it to our specific requirements

Actions/Operations :

• Actions are defined by a service and are the things that we can do to a resource.Such as viewing,creating,editing and deleting that resources
• IAM supports approx 40 actions for a user resource including create user,del user etc
• Any actions or resources that are not explicitly allowed are denied by default
• After our request has been authenticated and authorized ,AWS approves the actions in our request

Resources :

• A resource is an entity that exist within a service
• Examples are ec2 instances,S3 bucket,IAM user,DynamoDB table
• Each AWS service defines a set of actions that can be performed on each resource
• After AWS approves the actions on our request,these actions can be performed on the related resources within our account
• If we create a request to perform an unrelated action on a resource,that request is denied
• When we provide permissions using an identity based policy in IAM,then we provide permissions to access resources only within the same account

Session 4 : Users,Roles & Groups

Identity Federation :

• If our account users already has a way to be authenticated such as authentication through our corporate network
• We can federate those user identities into AWS
• A user who has already logged to the corporate using their corporate identity
• The corporate can replace their existing identity with a temporary identity in our AWS account
• This user can work in the AWS management console
• Similarly,an application that the user is working with can make a programmatic request using permissions that we define

Federation is particularly useful in these cases :

1.Our users already have identities in a corporate directory

• If our corporate directory is compatible with security assertion markup language(2.0)
□ We can configure our corporate directory to provide single sign on(SSO) access to the AWS mgt. console for our user
• If our corporate identity is not compatible with SAML 2.o
□ We can create Identity broker app to provide single sign on access to the AWS mgt. console for our users
• If our Corporate directory is Microsoft Active directory,we can use AWS directory service to establish trust between our corporate directory and our AWS account

2.Our users already have internet identities :
• If we are creating a mobile app or web based app that can let users identity themselves through an Internet Identity Provider like login with facebook,google,amazon or any open ID connect(OIDC) compatible identity provider,the app can use WEB federation to access AWS
• AWS recommends to use AWS console for Identity Federation

IAM Users and SSO :

• IAM users in our account have access only to the AWS resources that we specify in the policy that is attached to the user or to an IAM group that the user belongs to
• To work in the console,user must have permissions to perform the actions that the console performs,such as listing and creating AWS resources

IAM Identities :

• IAM identities is what we create under our AWS account to provide authentication for people,application and processes in our AWS account
• IAM Identities represent the user and can be authenticated and then authorized to perform actions in AWS
• Each of these can be associated with one or more policies to determine what actions a user,role or member of a group can do with which resources and under what conditions
• IAM group is a collection of IAM users
• IAM role is very similar to IAM user

IAM Users :

• An IAM user is an entity that we create in AWS.It represent the person or service which uses the IAM user to interact with AWS
• We can create 10 users at a time
• An IAM user can represent an actual person or an application that requires AWS access to perform actions on AWS resources
• A primary use for IAM users is to gives people the ability to signin to the AWS mgt. console for interactive task and to make programmatic request to AWS services using the API or CLI

For any User,we can assign them :

• A username and password to access the AWS console
• An access key id and secret key that they can use for programmatic access
• The newly created IAM users have no password and no access key.we need to create the User request
• Each IAM user is associated with one and only one AWS account
• Users are defied within our account,so users do not have to do payment.Bill would be pay by the parent account

IAM Groups :

• An IAM group is a collection of IAM users
• It is a way to assign permissions/policies to multiple users at once
• Use groups to specify permissions for a collection of users,which can make those permissions easier to manage for those users
• Ex : We could have a group called HR and give that group the types of permissions that HR department typically needs

□ Any user in that group automatically has the permission that are assigned to the group
□ If a new user joins our organisation and should have HR previliges,we can assign the appropriate permissions by adding the user to that group
□ If a person changes job in our organization instead of editing that users permissions,we can remove him or her from the old groups and add him or her to the appropriate new groups

IAM Group Limitations :

• A group is not an identity in IAM because it cannot be identified as a principal in a permission policy
• Groups cant be nested
• We have a limit of 300 groups in an AWS account
• A user can be a member of upto 10 IAM groups

IAM Roles :

• An 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
• An IAM role does not have any credentials associated with it
• Instead of being uniquely associated with one person , a role is intended to be assumable by anyone who needs it
• An IAM user can assume a role to temporary take an diff permissions for a specific task
• An IAM role can be assigned to a federated user who signs in by using an external identity provider instead of IAM

IAM temporary Credentials :

• Temporary credentials are primarily used with IAM roles,but there are also other uses
• We can request temporary credentials that have a more restricted set of permissions than our standard IAM users
• This pervent us from accedentially performing task that are not permitted by the more restricted credentials
• A benefit of temporary credentials is that they expire automatically after a set period of time

Session 5 : Permissions & Policies,Federation users,Resource based policies,Root and IAM user

Permissions

• The access mgt portion of AWS identity and access mgt helps us to define what a user or other entity is allowed to do in an account,often referred to as authorization
• Permissions are granted through policies that are created and then attached to users,groups or roles

Policies and Users :

• By default,IAM users cant access anything in our account
• We grant permissions to a user by creating a policy which is a document that defines the effect,actions resource and optional conditions
• Any actions or rsources that are not explicitly allowed are denied by default

IAM Multiple Policies :

• Users or groups can have multiple policies attached to them that grant diff permissions
• in the case of multiple policies attached to a user(or a group),The users's permissions are calculated based on the combination of policies

Federation users and roles :

• Federation user dont have permanent identities in our AWS account the way that IAM users do
• To assign permissions to federated users we can create an entity referred to as a role and define permission for the role
• When a federated user sign in to AWS, the user is associated with the role and is granted the permission that are defined in the role

Resources Based Policies :

• In some cases(Like S3 Bucket),we can attach a policy to a resources in addition to attaching it to a group or user.This is called a resource based policy
• A resource based policy contains slightly different information than a user based policy
□ In a resource based policy we specify what actions are permitted and what resources is affected
□ We can explicitly list who is allowed access to the resource(a principal)
□ Resource based policies include a principal element that specifies who is granted the permissions

IAM user-The root user

• When we first create an AWS account,we create an account(or root user) identity which we use to signin to AWS
• The account root user credentials are the email address used to create the account and a password,which can be used to signin to the AWS mgt console as the root user
• When we signin as the root user,we have complete,unrestricted access to all resources in our AWS account,including access to our billing information and the ability to change our password
• This level of access is necessary when we initially set up the account
• It is not possible to restrict the permissions that are granted to the AWS account

AWS recommends that :

• AWS recommends that we dont use root user credentials for everyday access
• Also AWS recommends that we do not share our root user credentials with anyone,because doing so gives them unrestricted access to our account
• Create an IAM user for ourself and then assign ourself administrative permission for our account
• We can then signin as that user to add more users as needed
• An IAM user with administrator permissions is not the same things as the AWS account root user

IAM Users :

• An IAM user is an entity that we create in AWS.It represents the person or service who uses the IAM user to interest with AWS
• An IAM user can represent an actual person or an application that requires AWS access to perform action on AWS resources
• IAM users are global entities,like an AWS account is today.No region is required to be specified when we define user permissions.Users can use AWS services in any geographic region

For any user we can assign them :

• A username and password to access the AWS console
• An access key(access key id and secret key) that they can use for programmatic access(issuing request) to our AWS services using API and CLI
□ We assign either or both based on the user activities and needs
• We can view and download our secret access key only when we create the access key
• We cannot view or recover a secret access key later
• If we lose our secret access key.We can create a new access key
• Each IAM user is associated with one AWS account

By default a new IAM user :

• A new IAM user has no permission to do anything
• Has no password and no access key(neither an access key ID nor a secret access key).It means no credentials of any kind
• We must create the type of credentials for an IAM users based on what the user will be doing
• We can grant user permissions by attaching IAM policies to them directly or making them members of IAM group where they inherit the group policies/permissions
• We can have upto 5000 users per AWS account

Session 6 : IAM Role, Ways to use a role,IAM role vs Resource based policy,IAM role delegation,Cross Account Permission

IAM Role :

• An IAM role,is a set of permissions that grant access to actions and resources in AWS
• These permissions are attached to the role,not to an IAM user or Group.Instead of being uniquely associated with one person a role is intended to be assumable by anyone who needs it
• A role does not have standard long-term credentials associated with it
• If a user assumes role temporary security credentials are created dynamically and provided to the user

Following entities can use Role :

1. An IAM user in the same AWS account
2.An IAM user in a different AWS account
3.A Webservice offered by AWS such as Amazon Elastic Compute Cloud

There are two ways to use a role :

1.Interactively in the IAM console

• IAM users in our account,using the IAM console can switch to a role to temporarily use the permissions of the role in the console
• The user give up their original permission and take on the permission assigned to the role
• When the user exists the role,their original permissions are restored

2.Programmatically with the AWS CLI,tools for Windows powershell or API

• An application or a service offered by AWS(like amazon ec2) can assume a role by requesting temporary security credentials for a role with which to make programmatic request to AWS
• We use a role this way so that we dont have to share or maintain long-term security credentials for each entity that requires access to a resource

Difference between IAM role and Resource based Policy

• Unlike a user-based policy a resource based policy specifies who can access that resources
• Cross account access with a resource based policy has an advantage over a role with a resource that is accessed through a resource based policy,the user still works in the trusted account and does not have to give up his or her user permissions in place of the role permissions
• In other words,the user continues to have access to resources in the trusted account at the same time as he or she has access to the resource in the trusting account
• This is useful for task such as copying information to or from the shared resource in the other account
• Note that not all services support resource based policy
• The following services support resource-based policy :
1.Amazon S3
2.Amazon SNS

3.Amazon SQS
4.Amazon Glacier Vault

IAM role delegation :

• Delegation is the granting of permissions to someone to allow access to resources that we control
• Delegation involves setting up a trust between the account that owns the resource(the trusting account) and the account that contains the users that need to access the resource(the trusted account)
• The trusted and trusting accounts can be any of the following :
1.The same account
2.Two accounts that are both under our organizations control
3.Two accounts owned by diff organization

• To delegate permission to access a resource,we create an IAM role that has two policies attached :
1.The trust policy
2.The permission policy
• The trusted entity is included in the policy as the principal element in the documennt
• When we create a trust policy,we cannot specify a wildcard(*) as a principal

Cross Account Permission :

• We might need to allow users from another aws account to access resources in our aws account.If so,dont share security credentials,such as access keys between accounts.Instead use IAM roles
• We can define a role in the trusting account that specifies what permissions the IAM user in the other account are allowed
• We can also designate which aws account have the IAM users that are allowed to assume the role.We do not define users here,rather AWS accounts

Role for Cross-Account Access :

• Granting access to resources in one account to a trusted principal in a diff account
• Roles are the primary way to grant cross-account access
• However with some of the webservices offered by AWS,we can attach a policy directly to a resource,these are called resource based policies.We can use them to grant principals in another AWS account access to the resource

LABS :

Session 7 : Creating IAM Users

Session 8 : Creating Groups & Inline Policy

Session 9: Cross Account Access using IAM role

• Login into 1st AWS account
• Create one group and then create two iam users in it
• Attach policy to group(EC2 Readonly access)
• Login to 2nd AWS account
• Create one S3 Bucket
• Create a role-Select Another AWS account-Insert Account ID of 1st AWS account-Attach policy -S3Readonlyaccess-Role Name-S3 read
• Now login back to 1st aws account click on Group-Permission-Inline Policy-Select-AWS security Token service-Assume Role
• Click on ‘ADD Statement’-Apply policy

• Now login as IAM user1
• Switch role-Now check,whether we are able to see the bucket of another account or not
• Repeat above 2 steps but with user 2
• Now login into 2nd AWS account-Role-Trusted Relationship-edit-paste arn of user1-update policy
• Now login again in user1-switch role-test s3 bucket
• Now login as user2-switch role-we will get error

Session 10 : Connecting Windows AD Server to AWS

• Create One Win server 2019 Base
• Change its password and then Install AD DS in ADD role & Features
• Now,create One forest i.e “tg.com”
• Now, go to server manager-Tools-DNS-Reverse Lookup Zone
• Now-DNS-Forward Lookup Zone-Enable Update Associated Pointer-Apply
• Now in Ethernet setting-Enter private IP of Server in Preferred DNS Server
• Now click on Server manager-Tools-AD USers and Computers-Create two users ‘A & B’ -Give Password-India@123
• Now go to AWS Mgt console-search Directory Services-AD Connector
• Directory DNS name-Darkrose-in-DNS IP address-Private Ip of AD server-usernmae-Administrator-Password-Same as AS server Password
• Go to IAM-Role-Directory Services-EC2Full access-Create Role
• Create One more role for Billing
• Add user A & B to above role
• Copy the URL and paste in Incognito Tab