**IAM Introduction:**

* AWS IAM (Identity and Access Management) is a service provided by Amazon Web Services that allows you to manage user access to your AWS resources.
* In simple terms, it allows you to create and manage users and groups, and control what resources they can access and what actions they can perform on those resources.
* With IAM, you can also set up security policies and permissions that determine who can access your resources and under what conditions. This allows you to grant or restrict access to specific AWS resources such as S3 buckets, EC2 instances, and more.
* IAM is important for securing your AWS environment and ensuring that only authorized users have access to your resources.

**Root User:**

* The AWS root user is the initial user created when you sign up for an AWS account. This user has complete access to all AWS services and resources in the account and has full administrative privileges, which means they can create, modify or delete any resource in the account.
* As the root user, you have access to sensitive information such as billing details, security credentials, and the ability to change account-level settings. It's important to secure your root user account and use it only when necessary.
* AWS recommends that you create an IAM user with administrative permissions and use it instead of the root user for day-to-day tasks. This is a best practice to follow, as it reduces the risk of accidental changes to your AWS environment and improves security by limiting access to the root user account. Additionally, using IAM users allows you to audit and monitor access and actions performed within your AWS account, making it easier to track and troubleshoot issues.
* The root user is similar in concept to the UNIX root or Windows Administrator account — it has full privileges to do anything in the account, including closing the account.
* The root user can be used for both console and programmatic access to AWS resources.

**IAM Users:**

* IAM users are similar to normal users in Linux, this user can interact with the console and use the CLI.
* AWS IAM users are identities that you create within AWS Identity and Access Management (IAM) and that you can use to interact with AWS services and resources.
* In simple terms, IAM users are the accounts you create to grant access to individuals or applications that need to interact with your AWS resources. Each user has a unique set of security credentials (access key ID and secret access key) that are used to authenticate the user and authorize access to the resources.
* With IAM users, you can grant permissions to access specific AWS resources or actions within those resources. You can also set up multi-factor authentication (MFA) and password policies to enforce security and reduce the risk of unauthorized access.
* IAM users can be organized into groups, which makes it easier to manage access permissions for a set of users with similar requirements. You can also assign IAM roles to users, which provide temporary permissions to resources for specific tasks.
* Using IAM users is a recommended best practice in AWS, as it helps you to control and manage access to your AWS resources in a granular and secure way. It also provides an audit trail of actions performed by each user, which helps with compliance and troubleshooting.
* We can create a maximum of 5000 users in an AWS account.

**Groups:**

AWS IAM groups are collections of IAM users that share the same permissions. By creating IAM groups, you can simply access management and make it easier to manage permissions for a set of users with similar requirements.

Here are some key points to know about AWS IAM groups:

* Creating IAM Groups: You can create an IAM group by logging into the AWS Management Console and navigating to the IAM console. You can also use the AWS CLI or AWS SDKs to create IAM groups programmatically.
* IAM Group Permissions: Permissions can be granted to IAM groups using IAM policies, which are a set of permissions that define what actions the group can perform on what resources.
* IAM Group Users: IAM users can be added to IAM groups, and the permissions granted to the group automatically apply to all the users in the group. Users can belong to multiple groups, and the permissions granted to those groups are cumulative.
* IAM Group Management: You can manage IAM groups by adding or removing IAM users, and by modifying the permissions granted to the group. IAM groups can be deleted when they are no longer needed.
* IAM Group Best Practices: Best practices for using IAM groups include creating groups that reflect job functions or business processes, assigning permissions based on least privilege, and regularly reviewing and updating group permissions.
* By using IAM groups, you can simply access management and make it easier to manage permissions for a set of users with similar requirements. IAM groups can be used in combination with IAM roles and policies to provide a granular and secure way to control access to AWS resources.
* We can create a maximum of 300 groups in an AWS account.
* An IAM user can be a member of 10 groups.

**Role:**

Roles are a way to grant temporary permissions to access AWS resources to trusted users or services. IAM roles enable you to define a set of permissions that determine what actions a user or service can perform on specific AWS resources.

Here are some key points to know about AWS IAM roles:

* Creating IAM Roles: You can create an IAM role by logging into the AWS Management Console and navigating to the IAM console. You can also use the AWS CLI or AWS SDKs to create IAM roles programmatically.
* IAM Role Trust Policy: An IAM role must have a trust policy that specifies the trusted entities that are allowed to assume the role. The trust policy determines who can assume the role and under what conditions.
* IAM Role Permissions: Permissions for an IAM role are defined by attaching an IAM policy to the role. The policy specifies what actions are allowed or denied on what resources.
* IAM Role Access: To access AWS resources using an IAM role, a user or service must first assume the role. This is done by requesting temporary security credentials from the AWS Security Token Service (STS). Once the user or service has assumed the role, they can access the resources and perform the actions allowed by the role's permissions.
* IAM Role Best Practices: Best practices for using IAM roles include defining roles that are based on job functions or business processes, using policies that grant the least privilege access, and regularly reviewing and updating role permissions.
* By using IAM roles, you can grant temporary permissions to access AWS resources to trusted users or services. IAM roles provide a granular and secure way to control access to AWS resources, and they can be used in combination with IAM policies and groups to manage access at scale. IAM roles are commonly used with services like EC2 instances, Lambda functions, and S3 buckets, to provide secure access to AWS resources.
* Each role can have up to 10 policies attached.
* Max 1000 roles in an AWS account.

**Policy:**

In the context of AWS Identity and Access Management (IAM), an AWS policy is a document that defines permissions for an IAM identity or an AWS resource. Policies are used to grant or deny access to specific actions and resources in AWS services.

Here are some key points to know about AWS policies:

* Policy Structure: AWS policies are structured JSON documents that define a set of permissions for one or more IAM identities or AWS resources. A policy consists of one or more statements, and each statement defines a single permission.
* Policy Scope: Policies can be attached to IAM users, groups, roles, and to AWS resources. When a policy is attached to an identity or a resource, it defines the actions that the identity or resource is allowed to perform on specified resources.
* Policy Syntax: AWS policies use a specific syntax for defining permissions. This includes specifying the action, resource, effect, and condition for each statement in the policy.
* Policy Versions: AWS policies can have multiple versions, which allows you to update policies over time while maintaining the ability to roll back to a previous version if needed.
* Policy Best Practices: Best practices for using AWS policies include granting the least privilege access, using conditions to further restrict access, and regularly reviewing and updating policies to ensure they are still relevant.
* AWS policies are a powerful tool for controlling access to AWS resources. By using policies, you can grant permissions to specific actions and resources on an as-needed basis, which helps to ensure the security and integrity of your AWS environment. Policies can be used in combination with IAM roles and groups to provide a granular and secure way to control access to AWS resources.

Here's an example of an AWS policy:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"s3:ListBucket",

"s3:GetObject",

"s3:PutObject"

],

"Resource": [

"arn:aws:s3:::example-bucket",

"arn:aws:s3:::example-bucket/\*"

]

}

]

}

This policy allows the user to perform three actions (list, get, and put) on an Amazon S3 bucket named example-bucket. The policy is structured as follows:

* Version: This specifies the policy language version, which is always set to "2012-10-17".
* Statement: This is an array of one or more statements that define the access control rules.
* Sid: This is an optional identifier for the statement.
* Effect: This specifies whether the statement allows or denies access to the resource. In this case, it is set to "Allow".
* Action: This specifies the action or actions that are allowed or denied. In this case, it allows the actions "s3:ListBucket", "s3:GetObject", and "s3:PutObject".
* Resource: This specifies the resource or resources that the actions apply to. In this case, it allows access to the S3 bucket "arn:aws:s3:::example-bucket" and all objects within it ("arn:aws:s3:::example-bucket/\*").

This policy is an example of a simple S3 bucket access policy that grants a user permission to perform certain actions on the bucket. Other policies can be more complex and grant different permissions for different resources or conditions.

**Identity Provider:**

An AWS Identity Provider (IdP) is a service that enables you to use an external identity management system to authenticate users and provide them with access to AWS resources. With an identity provider, you can allow users to sign in with their existing credentials, rather than creating new AWS-specific credentials.

Here are some key points to know about AWS Identity Providers:

* Supported Identity Providers: AWS supports several popular identity providers, including Amazon Cognito, Microsoft Active Directory, Okta, and others.
* Authentication Workflow: When a user attempts to access an AWS resource, they are redirected to the identity provider's login page to enter their credentials. The identity provider then authenticates the user and returns an access token to AWS that is used to grant access to the requested resource.
* Identity Federation: An identity provider can be used to enable identity federation, which allows you to grant temporary access to AWS resources to users who are authenticated by an external identity provider. This allows you to provide access to AWS resources without requiring the user to have an AWS account or credentials.
* Identity Provider Configuration: To configure an AWS Identity Provider, you must first create an identity provider in the AWS Management Console, and then create an IAM role that allows users authenticated by the identity provider to assume the role and access AWS resources.
* Best Practices: Best practices for using AWS Identity Providers include ensuring that the identity provider meets your organization's security and compliance requirements, regularly reviewing and updating identity provider configurations, and using multi-factor authentication to further secure access to AWS resources.
* AWS Identity Providers provide a secure and flexible way to authenticate users and provide them with access to AWS resources. By using an identity provider, you can enable users to sign in with their existing credentials, reduce the number of AWS-specific credentials that need to be managed, and simplify access management for your AWS environment.

**Multi-Factor Authentication (MFA):**

* Multi-Factor Authentication (MFA) can add an extra layer of security to your infrastructure by adding a second method of authentication beyond just a password or access key. With MFA, authentication also requires entering a One-Time Password (OTP) from a small device.
* The MFA device can be either a small hardware device you carry with you (for example SafeNet IDProve 100 (OTP Token)) or a virtual device via an app on your smartphone (for example google authentication)

**Password Policy:**

An AWS Password Policy is a set of rules that specify the requirements for user passwords in an AWS account. Password policies are used to enforce strong password security and help prevent unauthorized access to AWS resources.

Here are some key points to know about AWS Password Policies:

* Password Requirements: An AWS Password Policy specifies the minimum and maximum length of passwords, as well as requirements for password complexity, including the use of uppercase and lowercase letters, numbers, and special characters.
* Password Expiration: A Password Policy can also specify how often users must change their passwords, and how long a password can be used before it must be changed.
* Account Lockout: A Password Policy can specify how many failed login attempts will result in an account being locked out, and for how long.
* Password Policy Enforcement: Password policies are enforced by AWS IAM, which checks the password of each user against the policy requirements at the time the password is set or changed.
* Best Practices: Best practices for using AWS Password Policies include setting strong password requirements, regularly reviewing and updating password policies, and providing user education and training on password security best practices.
* AWS Password Policies are an important security feature that helps prevent unauthorized access to AWS resources. By enforcing strong password security, you can help to ensure the security and integrity of your AWS environment. It's important to regularly review and update your password policies, and to provide user education and training on password security best practices to further enhance your security posture.

**Credential Report:**

An AWS Credential Report is a detailed report that provides information about the status of AWS credentials in an AWS account. The report contains information on IAM users, access keys, passwords, and other security-related information.

Here are some key points to know about AWS Credential Reports:

* Report Content: The report includes information about IAM users, access keys, password status, and password last used time. It also includes information on service-linked roles, MFA status, and other security-related information.
* Access and Format: Credential Reports can be accessed through the AWS Management Console or the AWS CLI, and can be downloaded in either CSV or JSON format.
* Use Cases: Credential Reports can be used to monitor the security status of your AWS environment, identify security vulnerabilities and compliance issues, and track the usage of IAM users and access keys.
* Report Scheduling: You can schedule the automatic generation of Credential Reports on a daily or weekly basis, and specify a preferred delivery method for the report.
* Best Practices: Best practices for using AWS Credential Reports include reviewing the report regularly to identify security issues, disabling unused or unnecessary access keys, and regularly rotating access keys and passwords to ensure security.
* AWS Credential Reports are an important security tool that provides valuable information about the status of AWS credentials in your account. By regularly reviewing and analyzing the report, you can identify and address potential security vulnerabilities and ensure the ongoing security and integrity of your AWS environment.

**Points to Remember:**

* IAM is universal. It does not apply to regions
* The “root account” is simply the account created when you first set up your AWS account. It has complete Admin access.
* New Users have NO permissions when first created.
* New Users are assigned Access Key ID & Secret Access Keys when first created.
* These are not the same as a password, and you cannot use the Access key id & Secret Access key to login into the console. You can use this to access AWS via the APIs and Command Line, however.
* You only get to view these once. If you lose them, you have to regenerate them. So save them in a secure location.
* Always set up Multifactor Authentication on your root account.
* You can create and customize your password rotation policy.

=======================================================

AWS configuration:

>aws configure

Enter access key

Enter secreat key

Default region name

Default file format

Max iam users can create -5000.

Max iam group can create is 300.

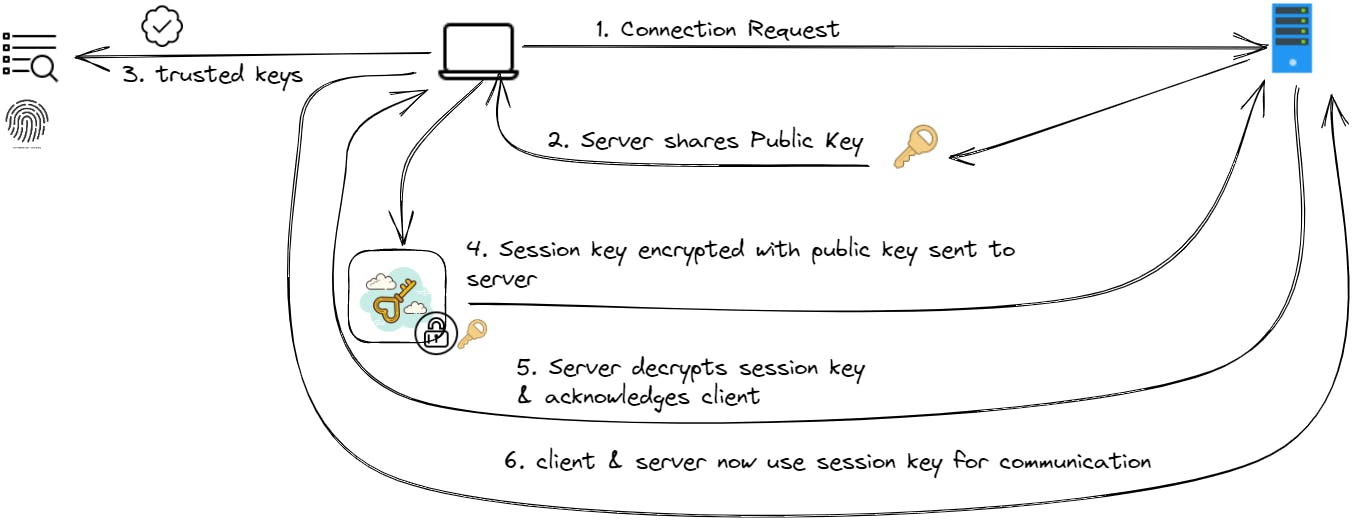
Iam user can be a member of 10 member.

Max 1000 roles in aws account.

Each role can have up to 10 policies attached.

# How SSH (Secure Shell) Works?

SSH (Secure Shell) is a protocol used to securely log into a remote computer and execute commands on that computer over an unsecured network. SSH provides strong authentication and encryption for secure data transmission over an insecure network.



Here's how SSH works:

1. The client initiates a connection to the server and requests to establish an SSH session. The client and server agree on the version of SSH to use.
2. The server sends its public key to the client.
3. The client verifies the server's public key by checking it against a list of trusted keys or by prompting the user to confirm the key's fingerprint.
4. The client generates a session key and encrypts it with the server's public key. This encrypted session key is sent to the server.
5. The server decrypts the session key using its private key and sends an acknowledgment to the client.
6. The client and server now use the session key to encrypt and decrypt all data that is sent over the SSH connection.
7. The client is now authenticated and can execute commands on the server. The client can also request port forwarding, X11 forwarding, or file transfers.
8. Once the session is complete, the client sends a message to the server indicating that it wishes to disconnect. The server acknowledges the message and terminates the session.

Overall, SSH provides secure authentication and encrypted communication between two networked devices.

SSH is widely used to remotely access and manage servers, perform administrative tasks, transfer files, and perform other tasks that require secure access to a remote computer over an unsecured network. SSH provides a secure and encrypted alternative to traditional remote access protocols, such as Telnet, that transmit data in clear text, making them vulnerable to eavesdropping and tampering.

***# Create custom Amazon Machine Images (AMI) from an EC2 instance::::>>>>>***

An Amazon Machine Image (AMI) contains all the information necessary to boot an Amazon EC2 instance with your custom software.

Essentially, an AMI is a virtual machine template that can define custom software, standard system packages, or any files required by the virtual machine.

Creating your own AMI is an essential practice if you plan to build an infrastructure that utilizes EC2 Auto Scaling Groups.

AWS Auto Scaling Groups require self-configurable instances in order to automatically scale up or down to match your application requirements. Your AMI serves as the basic unit of deployment, enabling the Auto Scaling Group to dynamically boot new custom instances as you need them.

AMIs can be backed by Amazon Elastic Block Storage (EBS) or instance store volumes. When an instance is launched using an EBS-backed AMI, the root volume for the instance is created using an EBS snapshot. When an instance is launched using an instance store-backed AMI, the root volume is created using a template stored in Amazon S3.

In this demo, you will create an AMI from a custom EC2 instance that has the Nginx web server preinstalled.

## Topics Covered:

• Storage

• Compute

• High Availability

• Amazon Web Services

• Compute for AWS

• Storage for AWS

• Amazon Elastic Block Storage (EBS)

• Amazon EC2

## Commands used in the video tutorial:

### Login as a root user:

sudo su -

### Update the system:

yum update -y

### Install Nginx:

yum install nginx -y

(If you are using Amazon Linuix 2 please use this command: sudo amazon-linux-extras install nginx1)

### Start Nginx Service:

sudo service nginx start

### Status of Nginx Service:

systemctl status nginx.service