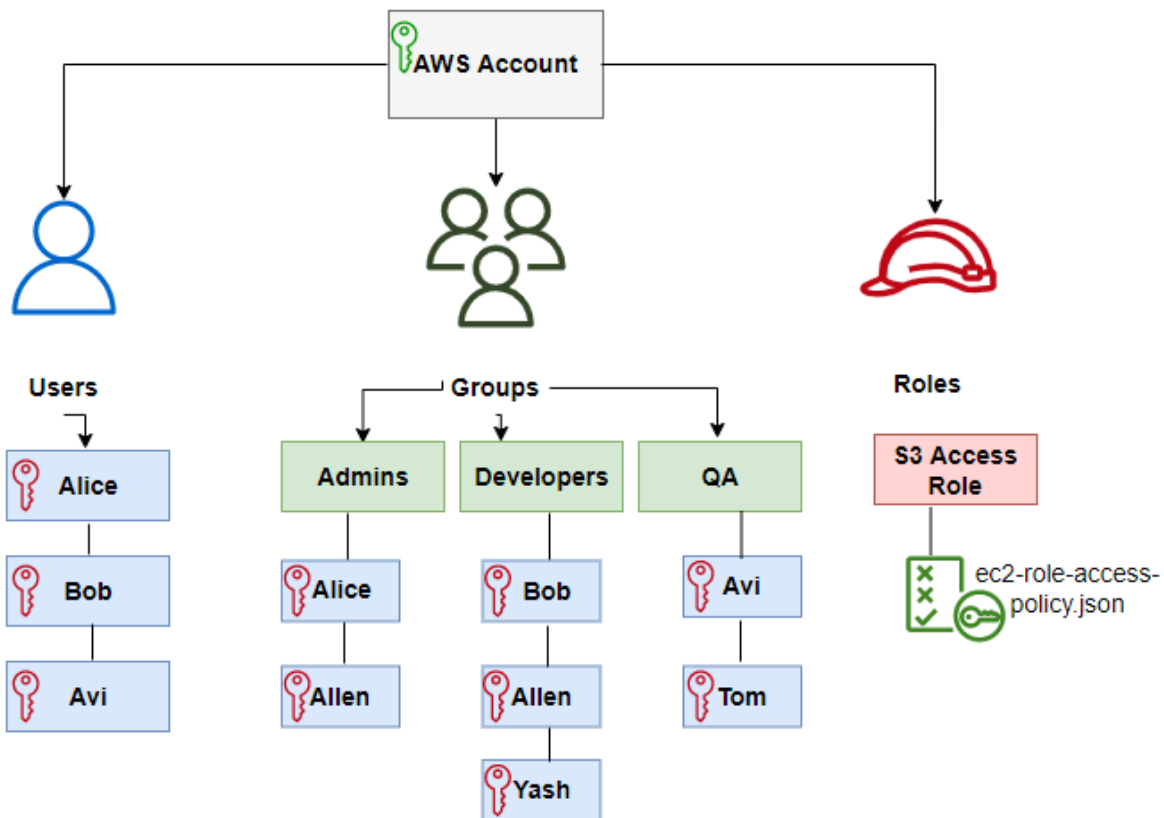


## #3 Task: AWS IAM Roles, Policies, User & Group management along with AWS CLI Operations.



**Identity and Access Management (IAM)** provides fine-grained access control across AWS accounts. With IAM, you can specify who can access which services and resources, and under which conditions. IAM is a pillar of security and provides you with easy ways to secure AWS accounts and resources.

### IAM Key Points

- IAM controls access to AWS services and resources.
- IAM enables access between AWS services (e.g. EC2 to RDS).
- The main feature of IAM is that it allows you to create separate usernames and passwords for individual users or resources and delegate access.
- IAM supports identity federation. If the user is already authenticated, such as through a Facebook or Google account, IAM can be made to trust that authentication method and then allow access based on it.

- IAM is a free service. There is no additional charge for IAM security. There is no additional charge for creating additional users, groups, or policies.
- IAM is PCI DSS compliance.
- IAM supports MFA.

It provides two essential functions that work together:

#### ***Authentication:***

- Validates the identity of a user or a service.
- Create and manage IAM identities.
- Federate corporate identities.

#### ***Authorization:***

- Defines the permissions and limits access to only specific resources for the permitted user.
- Assign permissions to IAM identities.
- Assign permissions to corporate identities.

## **IAM Key Terms**

IAM provides three primary types of identity: users, groups, roles. They serve different purposes, and each has an associated set of permissions using policies.

#### ***IAM User***

- Users represent people (For example, members of the Development and DevOps team).
- Users are used to giving individuals the ability to manage AWS resources via AWS Console or programmatically (e.g. CLI).
- User is an identity with an associated credential and permissions attached to it. i.e. IAM User is an account for a single individual to access AWS resources.
- Users are granted permissions, either by attaching a permissions policy or by adding them to a group. Access appropriate resources through assigned permissions.
- User can be a Person or a Service.
- IAM User can belong to a max of 10 Groups.

- When an AWS account is first created, it has a single user. This is the root user, which has complete access to all resources.

## ***IAM Account Root User Overview***

### ***IAM User Group***

- Group is a collection of users and makes it easy to grant permissions to a class of users.
- You can logically group users in the AWS account and manage the privilege level for all users in that group.
- You set permissions for the group, and those permissions are automatically applied to all the users in the group.
- For example, Group for developers, Group for System administrators. Both of them will have different levels of access to AWS services.
- Group is not a true identity. You can leverage groups for easier administration.
- IAM users can be added to multiple groups.
- IAM policies can be attached to a group. Each group can have up to 10 policies attached.
- IAM users within an IAM Group inherit all policies attached to a group.
- Using groups is not only easier but also more secure and manageable.
- User group cannot be identified as a Principal in a resource-based policy. User group is a way to attach policies to multiple users at one time.

### ***IAM Role***

- Roles have many of the same properties as users. However, roles are not linked to a single individual.
- You can create roles to give permission to use AWS service by another AWS service.
- Role enables a user or AWS service to assume permissions for a task. Roles are assumed by IAM Users (People or Applications) and external (federated) users.
- Roles do not have log-in credentials. Instead, roles are temporarily assumed by users who need a specific set of permissions to complete a task. It delegates access to a trusted entity.

- For example, you can create S3Admin role and assign it to your EC2 instance. This will enable that EC2 instance to manage S3 resources.
- You can leverage roles for easier administration.
- Policies can be attached to Roles.
- Roles rely on the AWS STS service.
- You can assume an IAM role by using AWS Security Token Service (STS) operations or switch to a role in the AWS Management Console to receive a temporary role session.

You should never put keys into code or instances, use Roles instead.

### ***IAM Policy***

- Policies are JSON document that defines permission and controls access AWS resources. Permissions specify who has access to the resources and what actions they can perform.
- Policies are an authorization mechanism. You can create IAM policies to define granular access to resources.
- For example, a policy could allow an IAM user to access one of the buckets in Amazon S3.
- Policy can be attached to IAM identities (Users, Roles, Groups).
- Single policy can include multiple permissions (or statements).
- Policies can be either customer managed or managed by AWS.
- There are multiple policies provided by AWS like Administrator, S3FullAccess, etc.
- You can also create your own policy and use it to manage privilege levels to your AWS resources.

### ***Policy contains the following information:***

- Who can access it.
- What actions that user can take.
- Which AWS resources that user can access.
- When they can be accessed.

### ***Types of policies:***

- **Managed policies:** It is a default policy that you attach to multiple entities (users, groups, and roles) in the AWS account. Managed policies, whether

they are AWS-managed or customer-managed, are stand-alone identity-based policies attached to multiple users and/or groups.

- **Inline policies:** It is a policy that you create that is embedded directly into a single entity (user, group, or role).

## Summary

IAM is one of the major building blocks of AWS. IAM lets you control who can use your resources (authentication) and in which ways (authorization). AWS IAM follows an incredibly granular approach in providing permissions and access control within your AWS environments.

## Installing and Configuring the AWS CLI

You'll need to use the AWS CLI if you want to create or interact with an EMR Cluster using commands. Take these important steps to install and configure it.

*The AWS Command Line Interface (AWS CLI) is a command-line tool that allows you to interact with AWS services using commands in your terminal/command prompt.*

AWS CLI enables you to run commands to provision, configure, list, delete resources in the AWS cloud. Before you run any of the [aws commands](#), you need to follow three steps:

1. Install AWS CLI
2. Create an IAM user with Administrator permissions
3. Configure the AWS CLI

### Step 1. Install AWS CLI v2

Refer to the official [AWS instructions to install/update AWS CLI](#) (version 2) based on your underlying OS. You can verify the installation using the following command in your terminal (macOS)/cmd (Windows).

```
# Display the folder that contains the symlink to the aws cli tool
which aws
# See the current version
aws --version
```

See the sample output below. Note that the exact version of AWS CLI and Python may vary in your system.

A terminal window with a dark background and light text. The window title is 'xyz - bash - 66x8'. The output shows the following commands and results:

```
(base) xyz$
(base) xyz$ aws --version
aws-cli/2.1.11 Python/3.7.4 Darwin/19.6.0 exe/x86_64 prompt/off
(base) xyz$ which aws
/usr/local/bin/aws
(base) xyz$
(base) xyz$
```

Mac/Linux/Windows: Verify the successful installation of AWS CLI 2

## Step 2. Create an IAM user (if you are using your own AWS account)

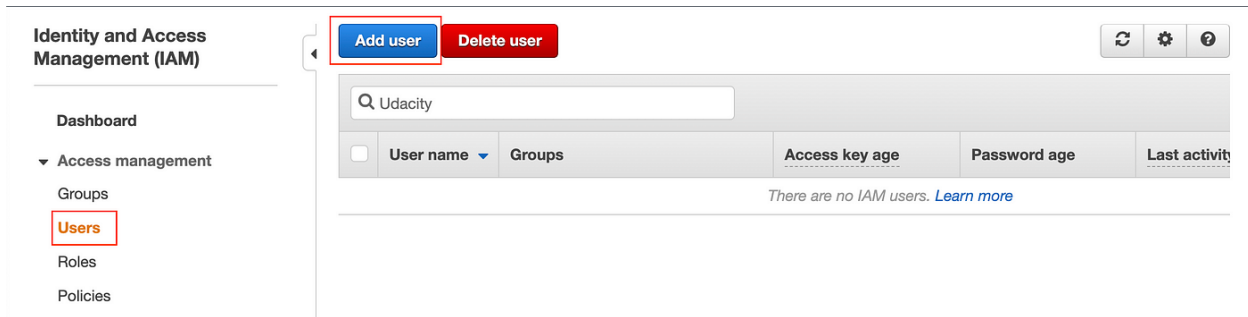
In this step, you will create an IAM user with Administrator permissions who is allowed to perform any action in your AWS account, only through CLI. After creating such an IAM user, we will use its Access key (long-term credentials) to configure the AWS CLI locally.

Let's create an [AWS IAM](#) user, and copy its Access key.

AWS Identity and Access Management (IAM) service allows you to authorize users / applications (such as AWS CLI) to access AWS resources.

The Access key is a combination of an Access Key ID and a Secret Access Key. Let's see the steps to create an IAM user, and generate its Access key.

- Navigate to the [IAM Dashboard](#), and create an IAM user.



## Add a new IAM user

- Set the user details, such as the name, and access type as Programmatic access only.

### Add user



### Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name\*

Admin

[+ Add another user](#)

### Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

Access type\*



**Programmatic access**

Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.



**AWS Management Console access**

Enables a **password** that allows users to sign-in to the AWS Management Console.


## Set the user name, and type (mode) of access


- Set the permissions to the new user by attaching the AWS Managed AdministratorAccess policy from the list of existing policies.


## Add user

1 2 3 4 5

### Set permissions

 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Create policy

Filter policies  Showing 30 results


	Policy name	Type	Used as
<input checked="" type="checkbox"/>	AdministratorAccess	Job function	None
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	None
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	AWS managed	None
<input type="checkbox"/>	AmazonSageMakerAdmin-ServiceCatalogProductsServiceRolePolicy	AWS managed	None
<input type="checkbox"/>	AmazonWorkSpacesAdmin	AWS managed	None
<input type="checkbox"/>	AmazonWorkSpacesApplicationManagerAdminAccess	AWS managed	None
<input type="checkbox"/>	AWSAppSyncAdministrator	AWS managed	None
<input type="checkbox"/>	AWSAuditManagerAdministratorAccess	AWS managed	None

Attach the AdministratorAccess policy from the list of pre-created policies


- Provide tags [optional], review the details of the new user, and finally create the new user.
- After a user is created successfully, download the access key file (.csv) containing the Access Key ID and a Secret Access Key. You can even copy the keys and stay on the same page. Don't skip this step as this will be your only opportunity to download the secret access key file.

## Add user

1 2 3 4 5

 **Success**  
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.  
  
Users with AWS Management Console access can sign-in at: <https://644752792305.signin.aws.amazon.com/console>

Download .csv

User	Access key ID	Secret access key
 Admin	AKIAZMHSC3LY6Q54MTVU	***** Show



### Step 3. Configure the AWS CLI

1. **Access key** — It is a combination of an Access Key ID and a Secret Access Key. Together, they are referred to as Access key. You can generate an Access key from the AWS IAM service, and specify the level of permissions (authorization) with the help of IAM Roles. If you are using the Udacity-provided AWS Gateway and account, these credentials are provided in the popup that appears when you click “Launch AWS Gateway” in the course menu to the left.
2. **Default AWS Region** — It specifies the AWS Region where you want to send your requests by default.
3. **Default output format** — It specifies how the results are formatted. It can either be a json, yaml, text, or a table.
4. **Profile** — A collection of settings is called a profile. The default profile name is default, however, you can create a new profile using the aws configure --profile new\_name command. A sample command is given below.
5. **Session Token** — If you are using the Udacity-provided AWS Gateway and Account, you will also need to add the session token from the AWS Gateway credential popup, as well as the Access Key and SecretKey from the popup as described here.



- Set the default profile credentials

```
# Navigate to the home directory
cd ~
# If you do not use the profile-name, a default profile will be created for you
aws configure --profile <profile-name>
# View the current configuration
aws configure list --profile <profile-name>
# View all existing profile names
aws configure list-profiles
# In case, you want to change the region in a given profile
# aws configure set <parameter> <value> --profile <profile-name>
aws configure set region us-east-1 --profile <profile-name>
```

Moving forward, you can use `--profile <profile-name>` option with any AWS command. This will resolve the conflict if you have multiple profiles set up locally.

The command above will store the access key in a default file `~/.aws/credentials` and store the profile in the `~/.aws/config` file.

Let the system know that your sensitive information is residing in the `.aws` folder

```
export AWS_CONFIG_FILE=~/.aws/config
export AWS_SHARED_CREDENTIALS_FILE=~/.aws/credentials
```

```
(base) xyz$ aws configure list
```

Name	Value	Type	Location
profile	<not set>	None	None
access_key	*****C74A	shared-credentials-file	
secret_key	*****Jn5N	shared-credentials-file	
region	us-east-2	config-file	/Users/xyz/.aws/config

```
(base) xyz$
```

Mac/Linux: A successful configuration

### Setting the Session Token

After a successful credential set-up, your “credentials” file will look like this one below. If you are using the Udacity-provided AWS account, you should have a session token name/value pair in your configuration as well.

Mac/Linux: View the credentials file using `cat ~/.aws/credentials` command

- Check the successful configuration of the AWS CLI, by running either of the following AWS command:

The output will display the details of the recently created user:

## Troubleshoot



1. Configuration basics
2. Configuration and credential file settings
3. Environment variables to configure the AWS CLI
4. Using the Session Token

Updating the specific variable in the configuration

In the future, you can set a single value, by using the command, such as:

# Syntax

# aws configure set <varname> <value> [--profile profile-name]

aws configure set default.region us-east-2

It will update only the region variable in the existing default profile.