

## What is IAM?

- IAM ( Identity & Access Management ) is where you manage your AWS users, groups, roles, Access policies etc....
- IAM provides access and access permissions to AWS resources ( such as EC2, S3 & DynamoDB)
- By default, new users in AWS created using IAM will have NO access to any AWS services. This is a **non-explicit deny** rule set on all new IAM users.
- Hence, for all the users (except root), permissions must be given that grant access to AWS services ( this is done through IAM policies )

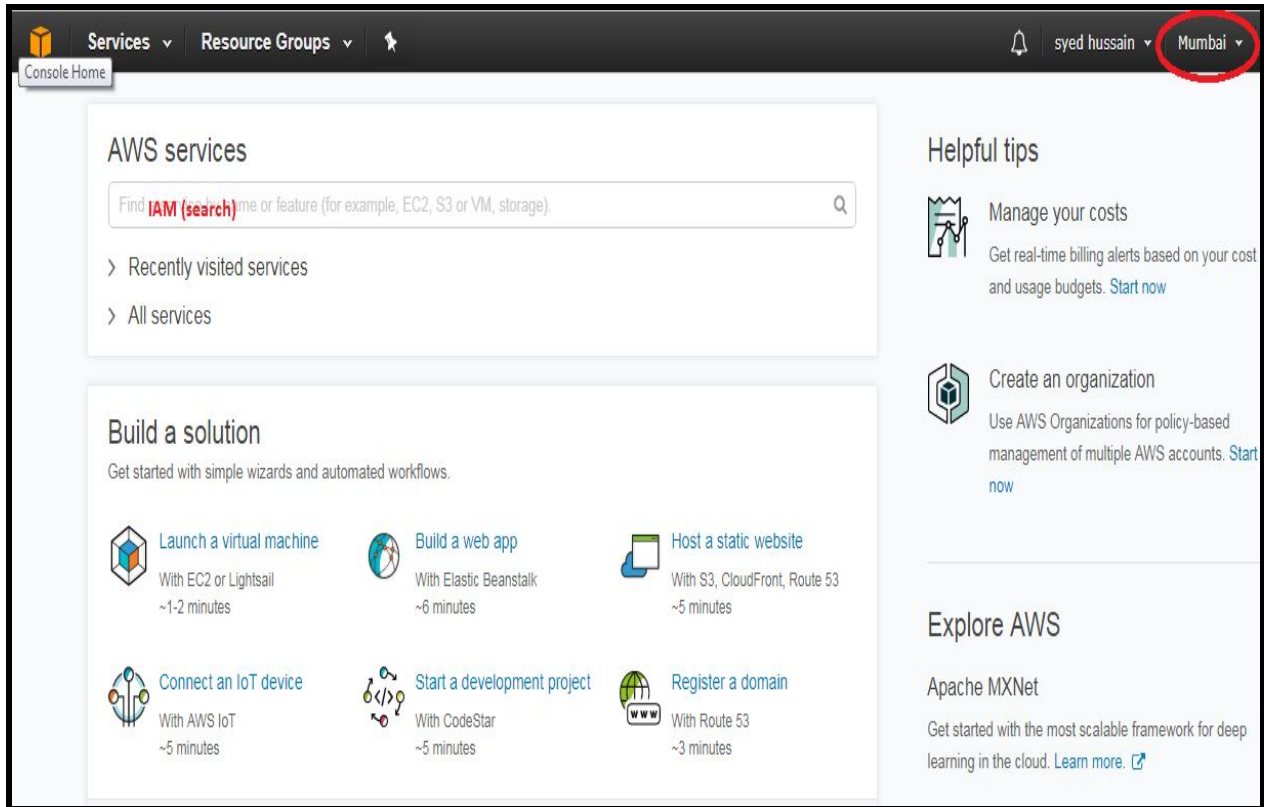
## Best Practices

- When a new AWS root account is created, it is **"best practice"** to complete the below tasks listed in Security Status ( We will do it in LAB)
- Delete your root access keys

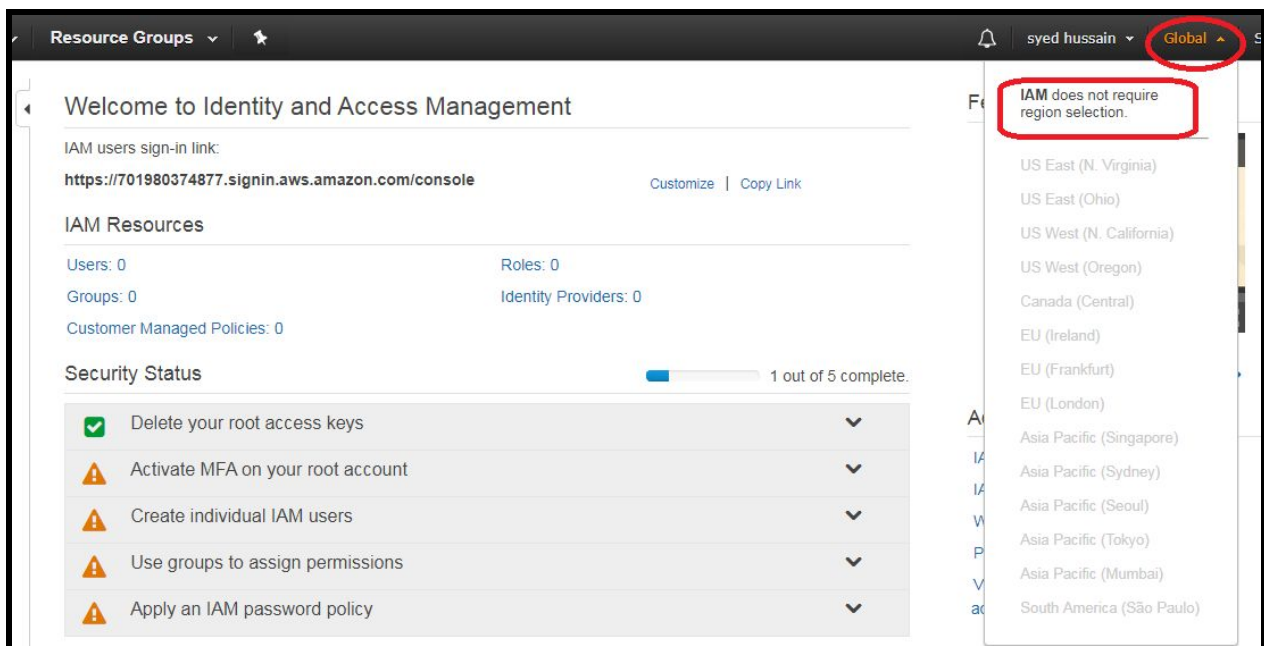
- Activate MFA on your root account
  - Create individual IAM users
  - User groups to assign permissions
  - Apply an IAM password policy
- 
- Always login with normal user for daily work - NOT as root user
  - It is best practice to follow “**Principle of Least privilege**” when administering AWS accounts, users, groups and roles.

## **Login to Management Console**

Make sure you have selected the nearest geographic location where you want to host the AWS services. It is good for reduce the network delay and for low-latency  
Hence I selected mumbai.



Search for IAM and navigate to its page.



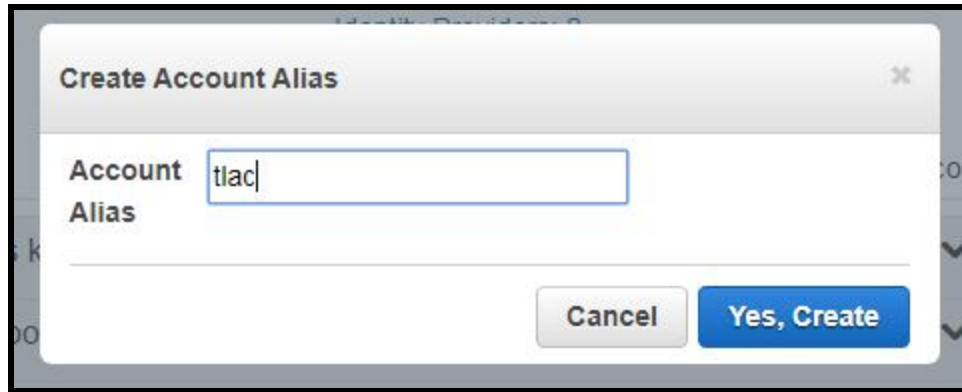
Here the most important thing you notice here is that it changes to Global location.  
It means **IAM is not region specific, it applies globally for your account**

Second thing you will notice here is URL Link for IAM page, You can customize that link as per your company standard instead of that default link

<https://companyname.signin.aws.amazon.com/console>

Click on Customize

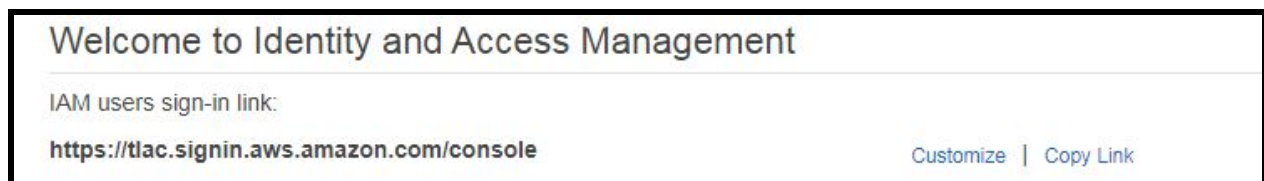
Then



Now you will see as below

IAM users sign-in link:

<https://tlac.signin.aws.amazon.com/console>



## Root Account :

First-Time Access Only:Your Root Account Credentials

When you create an AWS account, you create an account (or "root") identity, which you use to sign in to AWS. You can sign in to the AWS Management Console using this root identity—that is, the email address and password that you provided when creating the account. This combination of your **email address and password** is also called your **root account credentials**.

When you use your root account credentials, you have complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password.

This level of access is necessary when you first setup your account.

However, it's recommended that you don't use root account

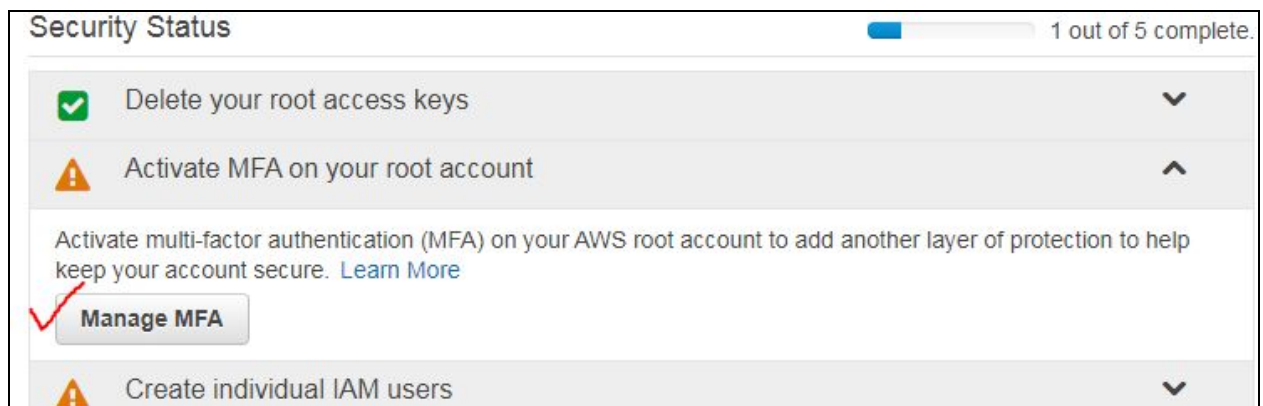
credentials for everyday access. It is not possible to restrict the permissions that are granted to the root account. Never share root Credentials.

Follow the below best practice for root credentials

### 1) By default root account access keys are removed



### 2) Second thing is MFA (Multi-Factor Authentication)



Click on ***Manage MFA***

Manage MFA Device

Select the type of MFA device to activate:

☒ A virtual MFA device

☐ A hardware MFA device

For more information about supported MFA devices, see [AWS Multi-Factor Authentication](#).

Cancel

Next Step

Select **Virtual MFA device** and then Next

Manage MFA Device

To activate a virtual MFA device, you must first install an AWS MFA-compatible application on the user's smartphone, PC, or other device. You can find a list of AWS MFA-compatible applications [here](#). After the application is installed, click Next Step to configure the virtual MFA.

☐ Don't show me this dialog box again.

Cancel

Previous

Next Step


Virtual MFA Applications	
Applications for your smartphone can be installed from the application store that is specific to your phone type. The following table lists some applications for different smartphone types.	
Android	<a href="#">Google Authenticator; Authy 2-Factor Authentication</a>
iPhone	<a href="#">Google Authenticator; Authy 2-Factor Authentication</a>
Windows Phone	<a href="#">Authenticator</a>
Blackberry	<a href="#">Google Authenticator</a>

Click on **Next Step**



Manage MFA Device

If your virtual MFA application supports scanning QR codes, scan the following image with your smartphone's camera.



► [Show secret key for manual configuration](#)

After the application is configured, enter two consecutive authentication codes in the boxes below and click Activate Virtual MFA.

Authentication Code 1

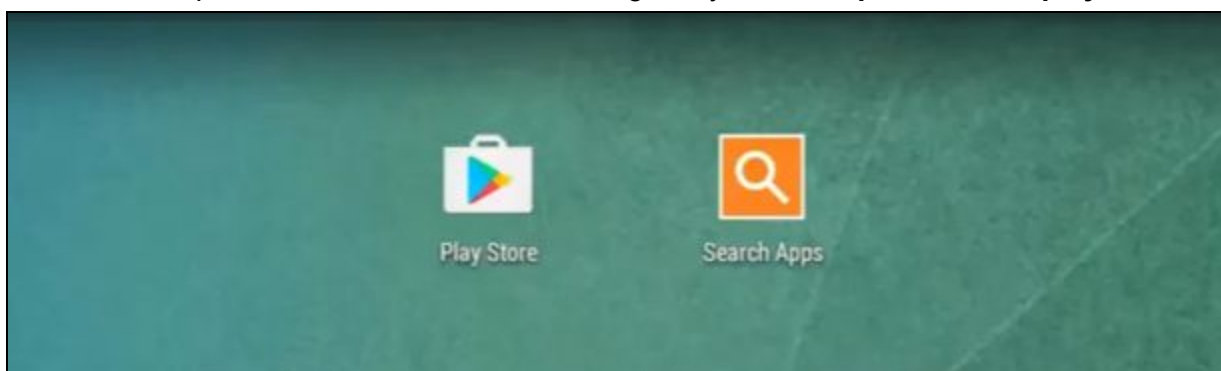
Authentication Code 2

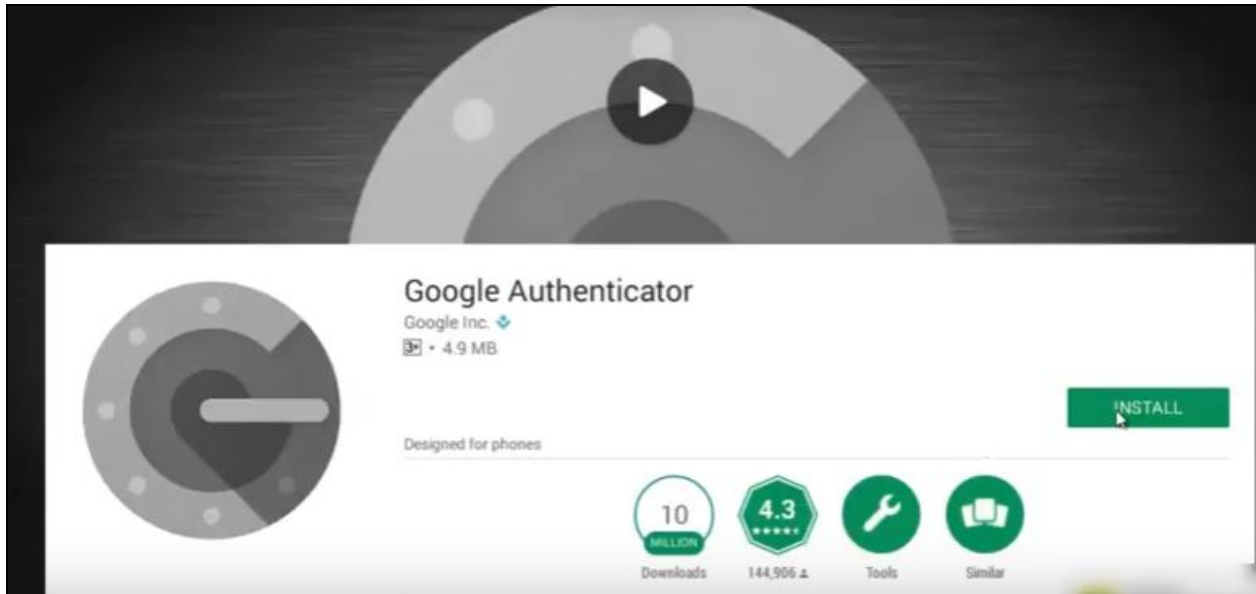
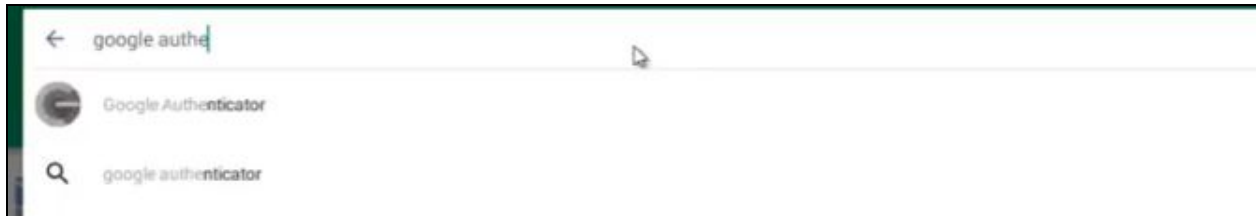
Cancel

Previous

Activate Virtual MFA

In order to fill up the above screen ,You need to go on your **Smartphone**, Go to **play Store**





Install and Open it


Then scan that above code presented in the above screen on AWS site

And enter the numbers in AWS Site to that above page



Manage MFA Device

If your virtual MFA application supports scanning QR codes, scan the following image with your smartphone's camera.



Show secret key for manual configuration

After the application is configured, enter two consecutive authentication codes in the boxes below and click Activate Virtual MFA.

Authentication Code 1

674781

Authentication Code 2

468150

Click on **Activate Virtual MFA button**

Manage MFA Device

The MFA device was successfully associated.

Finish

Now the status of security is looks like below

Security Status

2 out of 5 complete.

	Delete your root access keys	▼
	Activate MFA on your root account	▼
	Create individual IAM users	▼
	Use groups to assign permissions	▼
	Apply an IAM password policy	▼

Due to this MFA setting, whenever you try to login , you have to enter OTP which you get in your smartphone. Hence more security you place to your root credentials.

### 3) Create individual IAM users

Create individual IAM users

^

Create IAM users and give them only the permissions they need. Do not use your AWS root account for day-to-day interaction with AWS, because the root account provides unrestricted access to your AWS resources. [Learn More](#)

Manage Users

Click on Manage Users

Add userDelete user

Find users by username or access key

Showing 0 results

<input type="checkbox"/>	User name ▼	Groups	Access key age	Password age	Last activity	MFA
--------------------------	-------------	--------	----------------	--------------	---------------	-----

There are no IAM users. [Learn more](#)

Click on Add user

## Add user

1
2
3
4

Details
Permissions
Review
Complete

### Set user details

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

User name\*

[+ Add another user](#)

Create 2 users as “admin” and “srini” (don’t give any permissions as of now)  
(Remember the important point that user will have no permissions by default)

### 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.

Console password\* ☒ Autogenerated password ✓  
☐ Custom password

Require password reset ☒ User must create a new password at next sign-in ✓  
Users automatically get the `IAMUserChangePassword` policy to allow them to change their own password.

\* Required

[Cancel](#) [Next: Permissions](#) ✓

Then Click on Next:Permissions

Set permissions for srini

Add user to group

Copy permissions from existing user

Attach existing policies directly

**Get started with groups**  
You haven't created any groups yet. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Get started by creating a group. [Learn more](#)  
[Create group](#)

[Cancel](#) [Previous](#) [Next: Review](#) ✓

Then Click on **Next Review**

Or Create group by name Admin-Group or TempAdmin ( don't attach any policy as of now)

Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. [Learn more](#)

Group name

Create policy Refresh

Filter: Policy type Search Showing 275 results

	Policy name	Type	Attachments	Description
<input type="checkbox"/>	AdministratorAccess	Job function	0	Provides full access to AWS services and resources.
<input type="checkbox"/>	AmazonAPIGatewayAdministr...	AWS managed	0	Provides full access to create/edit/delete APIs in Amazon API Gateway v...
<input type="checkbox"/>	AmazonAPIGatewayInvokeFull...	AWS managed	0	Provides full access to invoke APIs in Amazon API Gateway.
<input type="checkbox"/>	AmazonAPIGatewayPushToCl...	AWS managed	0	Allows API Gateway to push logs to user's account.
<input type="checkbox"/>	AmazonAppStreamFullAccess	AWS managed	0	Provides full access to Amazon AppStream via the AWS Management C...

Cancel Create group

Create a TempAdmins

And add user to this group

## Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

### User details

User name	srini
AWS access type	Programmatic access and AWS Management Console access
Console password type	Autogenerated
Require password reset	Yes

### Permissions summary

The user shown above will be added to the following groups.

Type	Name
✓ Group	TempAdmins
Managed policy	IAMUserChangePassword

Cancel Previous Create user

## Add user



### ✓ 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://tlaac.signin.aws.amazon.com/console>

Download .csv

	User	Access key ID	Secret access key	Password	Email login instructions
▶	✓ srini	AKIAJEF4BMY73KDIT3FA	***** Show	***** Show	Send email <a href="#">↗</a>

Close

## List users

Search IAM

Dashboard

Groups

**Users**

Roles

Policies

Add user

Delete user

Find users by username or access key

<input type="checkbox"/>	User name ▼	Groups	Access key age	Password age	Last activity
<input type="checkbox"/>	srini	TempAdmins	None	Today	None

To manage user properties, Go inside the user

Search IAM

Dashboard

Groups

**Users**

Roles

Policies

Identity providers

Account settings

Credential report

Encryption keys

Users > srini

Summary

User ARN

arn:aws:iam::701980374877:user/srini

Path

/

Creation time

2017-08-29 03:14 EDT

Permissions

Groups (1)

Security credentials

Access Advisor

Add permissions

Attached policies: 1

Policy name ▼	Policy type ▼
Attached directly	
▶ IAMUserChangePassword	AWS managed policy

You can change permission individually to individual users

You can provide or revoke group membership

Security credentials or manage password for console you can do it)

Assign MFA and change it from here

If you want to cut off the programmatic access that also u can do it.

#### 4) User groups to assign permissions

You can create a new group like EC2admin  
Select the policy type like EC2 related policies  
Amazon EC2 full access , Next , create Group

Then Group action and add users or delete user etc.....

#### 5)



You can set password properties here as per your company requirement.

ROLES - we will see in EC2 section why we are using those one

NOTE: Make sure to store the access key and secret key at some good location , if you lost then only way to get back is to create again.

### IAM POLICIES

- A policy is a document that formally states one or more permissions
- IAM is providing some in-build policy templates to assign to users and groups

For example:

**Administrator access:** Full access to ALL AWS resources

**Power user access :** Admin access except it does not allow user/group management

**Read Only access :** Only view AWS resources (i.e. users can only view what is in an S3 bucket)

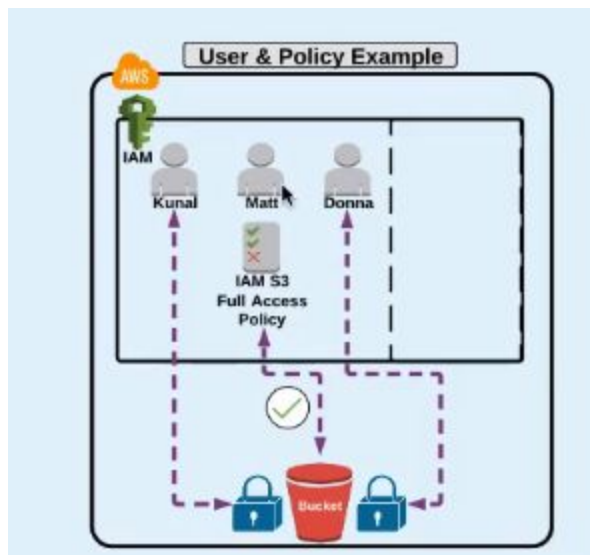
- More than one policy can be attached to a user or group at the same time
- Policies cannot be directly attached to AWS resources (such as an EC2 instance)
- By default, an **explicit deny** rule always overrides and an **explicit allow** rule

It means that the use of a “deny all” policy to quickly restrict ALL access that a user may have through multiple attached policies.

- You can also create custom IAM permission policies using the **policy generator** or written from scratch

## IAM user (normal)

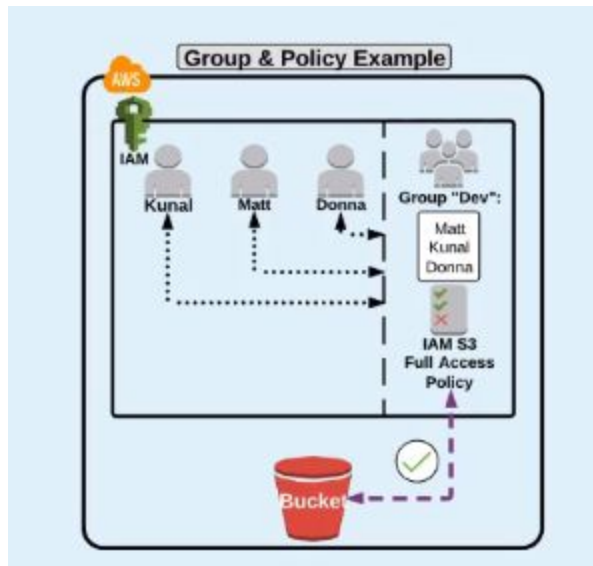
- when first created, by default an IAM user has a non-explicit “deny” for all AWS services - means it does not have access to anything until a policy granting allow access has been applied to the user or to the group the user belongs to.
- Users is applied with **single policy or multiple policies** applied to them at any given time
- Users are added to group also so that he will get all right as per hte policies assigned to that group.
- MFA can be applied on a per user basis for login and resource access/actions.
- IAM users receive unique access credentials so you do not (and should not) share with others
- User credential should NEVER be stored or “passed” to an EC2 instance



## IAM Group

- Allow you to assign IAM permission policies to more than one user at a time
- This ability allows for easier access management to AWS resources.





## IAM ROLES

- A role is something that another entity can “assume”, and in doing so acquires the specific permissions defined in the role
- In the context of this course, “entities” that can assume a role include
  - (a) AWS resources ( such as EC2 instance)
  - (b) non-AWS account holder who may need temporary access to an AWS resource (through a service like Active Directory)
- Roles must be used because policies cannot be directly attached to AWS services



For example:

If you are using an EC2 instance and it need to access an S3 bucket:

- Instance should assume a role from IAM with the proper required permissions (supp S3 read only)
- Instance can then perform actions based on the role it assumes ( read from S3 )
- You “can” but should never pass or store credentials in or to an EC2 instance - so roles are

- Until recently, you could only assign a role to an EC2 instance during the EC2 instance creation process. However, you can now assign/change a role that is assigned to an EC2 instance after the creation process via the CLI or the EC2 management console
- An EC2 instance can only have ONE role attached at a time

**Other example where we use the roles:**

- Other users can assume a “role” for temporary access to AWS accounts and resources through having something like Active Directory or Single Sign On service ( facebook, Google) assume an “Identity Provider Access” role.
- Create “cross account” access where a user from one account can assume a role with permissions in another account