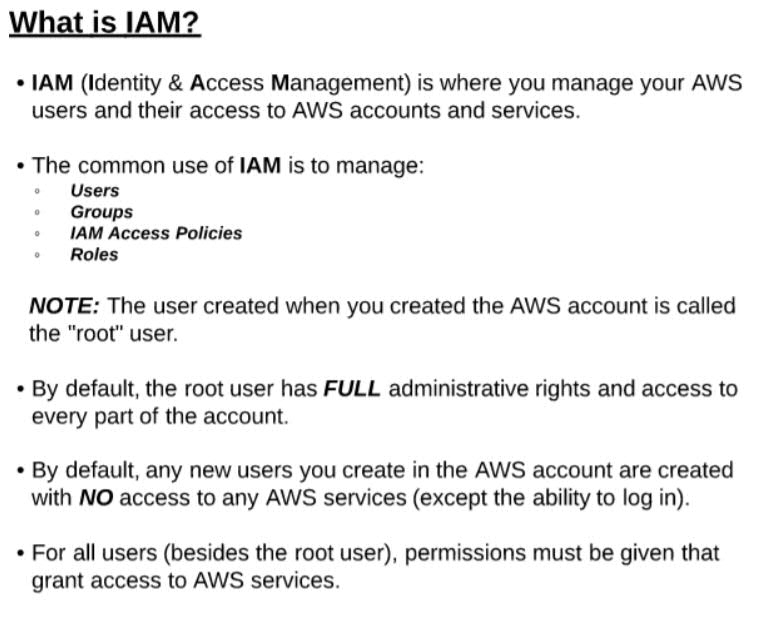
**IAM (Identity and Access management)**

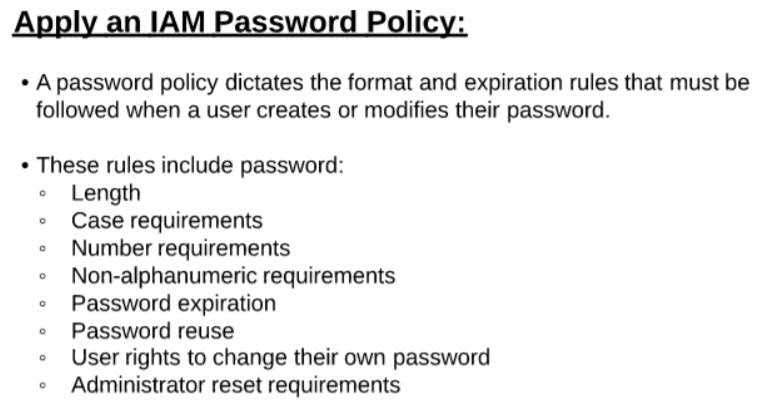
Whatever the user we get when we create AWS account is called the master/root account and we pay all the AWS bills using it.  
  
But, this master account cannot be shared with all the users as it has complete access privileges. We have to give only required access to the users.   
For example, if a user wants to work on S3, then we have to give access only to S3 but not to EC2. Or if some user wants to work only on few specific buckets in S3, then we should give access to only those buckets and not to entire S3. So, for this we have to create users and assign permissions to those users. Giving privileges can be done using IAM (Identity and Access Management) in AWS. In AWS we can give access to a person or a program.

If the privileges are for user interaction, then we do user account creations.  
If it is for c, then we create roles.  
Application in the sense, where we write the scripts to download the stuff from Aws or to keep backup in AWS.

Authentication, Authorization and Audit are the key components in Security.



**Policy:** Policy says for what we are giving the access (Authorization). User and role say whom to give the access.

  
For example, if a person wants to access AWS account, then we create a user.  
If an application has to access AWS account, then create a role.

**Let’s create a user and see in detail**:

Login to AWS account and click on **IAM** under **Security and Identity**.

Then click on Users, then click on **Add User,** then give the user name **Ex:** myuser.  
  
Under, **Access type**, we can see 2 options, 1) **Programmatic access** and 2) **AWS management console access**

Select the **AWS management console access** option.

Programmatic access is for applications/machines. Aws console access it to login from UI console.

Under Console Password choose custom password to give our own password and give the password.

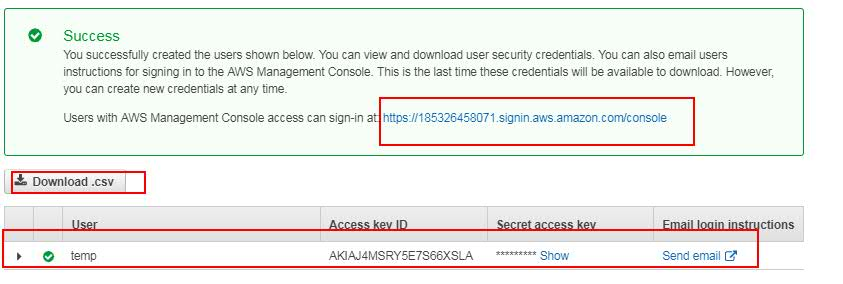
Select the require password reset option, if you want to allow the user to change the password for every login. And then click on **Permissions**.

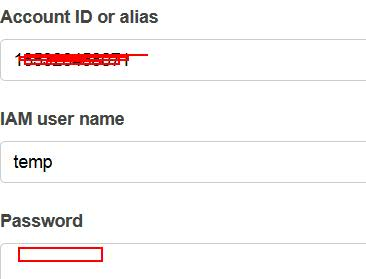
Under Permissions we have to specify the authorization.

We can select the **Add user to a group** option, if we want to add the user to any group.  
We can select the **Copy permissions from existing user** option, if we want to give same permissions of that user.

For now, select the **Attach existing policies directly** option to give specific selected access. For now, search and select **AmazonS3FullAccess** option.

So, the user will get full access only to S3. Then click on **Review**, review the changes and click on **Create user** option. Download the Downlaod.csv file for backup as it is one time downloadable.

  
  
Copy the console url and try to access with new user credentials.



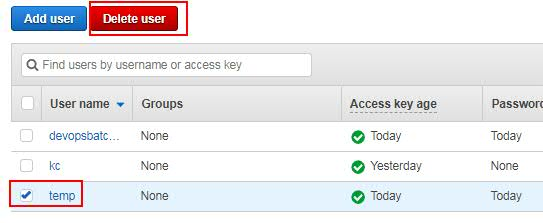
Once login, go to Services and click on **EC2** and you will get not authorized message as the user has privileges only to S3.

Now, go to Services and click on S3 and you can see full details of S3 as the user has full permissions to S3.

Now, let’s do the same for a program.  
**Deleting user:**

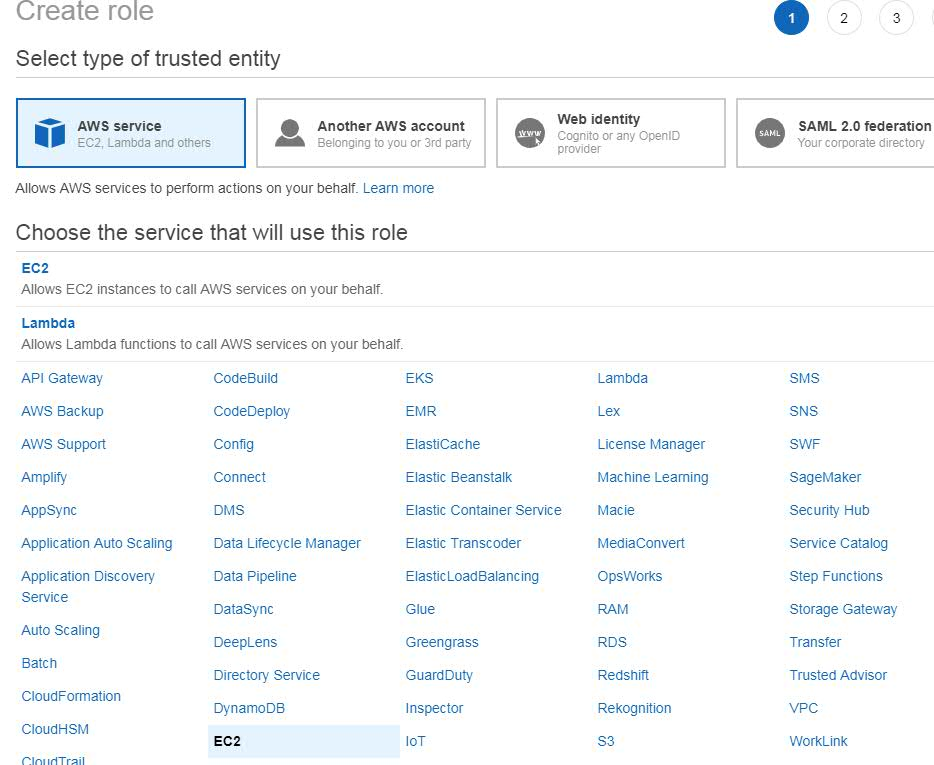
For this, we need to login with master account. For this while signing, select the option “**Signin using root account credentials”** and login.

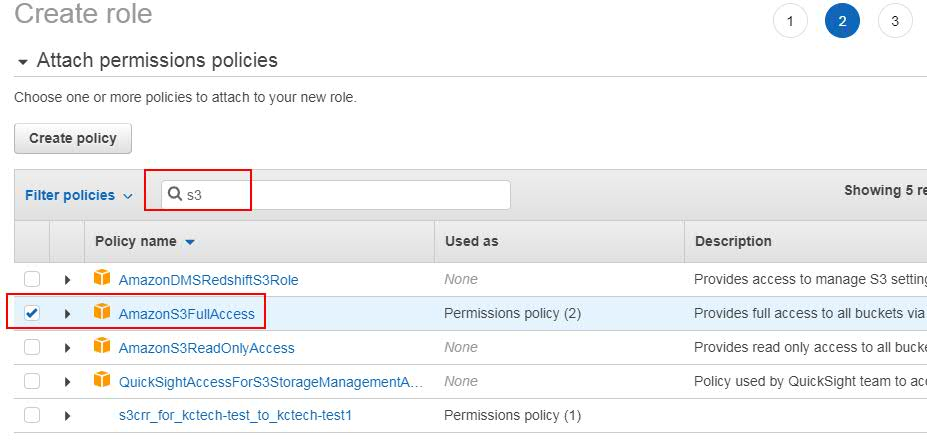
Then go to Services->IAM->Users and select the created user and delete that user.



Now, to give permissions for the applications using Roles, Go to Services->IAM->Roles and click on **Create new role** option. Then select where to apply the Role. Ex: EC2 Directory, Lambda etc.

**Roles:** For example, one Service wants to use another service. Like EC2 wants access to S3 to use. In that cases we use Roles. So, we have to select the Role for EC2 and assign the S3 policy.





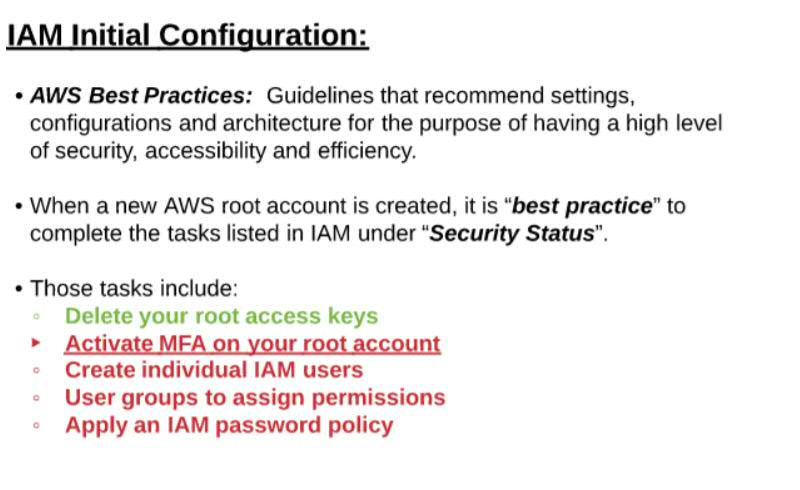
Based on the created role, we can give access to the application.  
For Example, if our application is running on EC2 and needs access to S3.  
then select “**Amazon EC2**” option under Select role type and click on Next.

Then search and Select “AmazonS3FullAccess” and click on next.

Then give the role name, **Ex:** myrole, give the description and click on Create role.

We use a concept called “**ARN**” to give user access to only specific S3 buckets.

**IAM Initial Configuration**



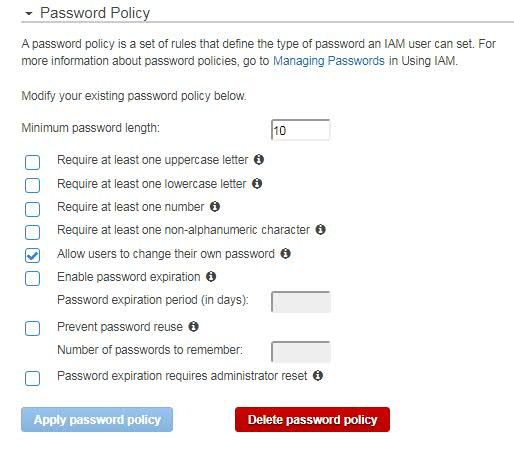
Aws has suggested some best practices when we create a root account.

From AWS console, if we select IAM from services, then we can see few things in red color.



We should not give AWS CLI access to root user as it may get into many issues. This can be restricted using “**Delete your root access keys**”.

User login password rules can be changed using “Apply an IAM password Policy”. For this click on Apply an IAM password Policy and see the configurations.



If you want, you can do modifications to configurations and click on Apply.

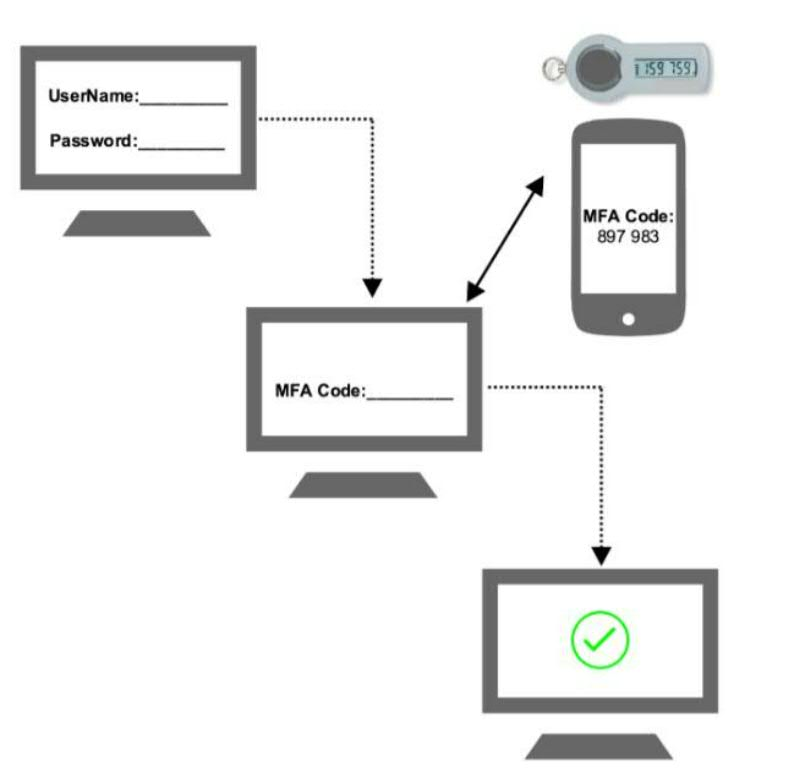
We can create individual users and can give permissions for them using “**Create Individual IAM users**” option.

We can create users, groups and can give permissions for them using “**Use groups to assign permissions**” option.

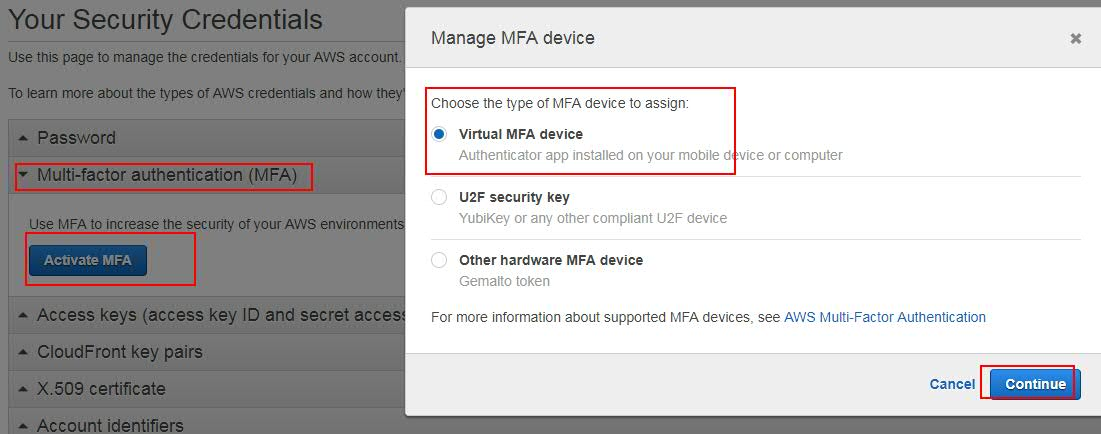
**MFA (Multi factor Authentication):**

This is like an OTP authentication. When signing to AWS account, along with user name and credentials, we use one more level of authentication.

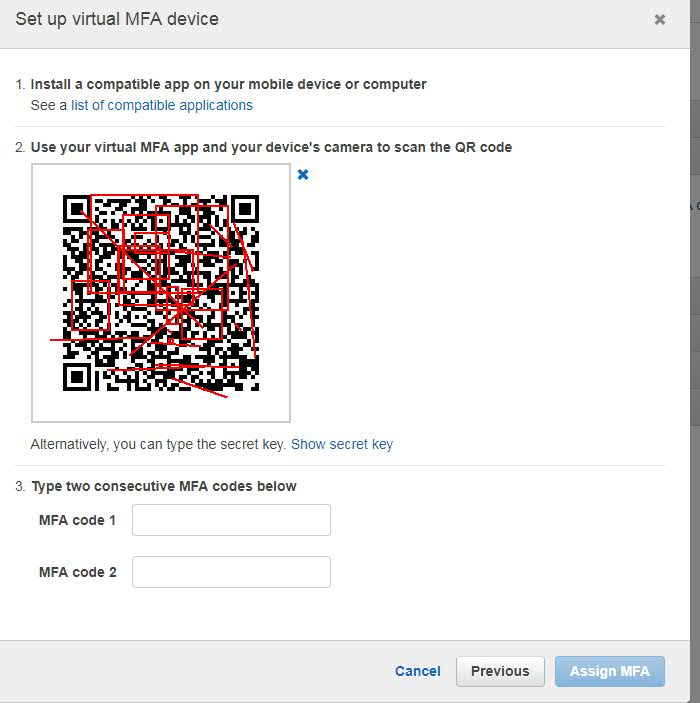
To configure this, first we have to install an authenticator app in our device (mobile or system), then from AWS select the MFA option and it will generate an QR code. Then scan that from your device and then enter the 2 authentications codes generated by authenticator in the devices. So that, for ever user login, along with username and password, Aws will ask for authentication token. This authentication token is generated my authenticator tool. There are so many authenticator tools available.  
MFA can we done with Software or Hardware authentication. For hardware type, we can use different tools like RSA, Symantec AIP etc. Hardware authenticator is chargeable where Software type is free of cost type. But most of the companies using S/W type.

  
  
Let’s see one example with Software type of MFA authentication.

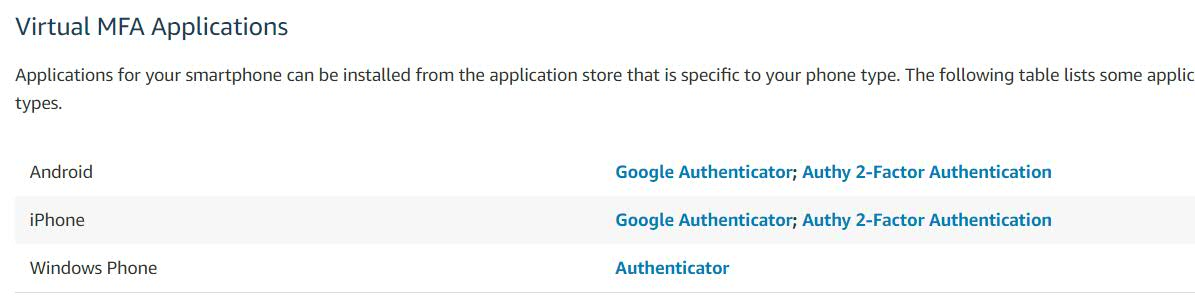
For this, click on “**Activate MFA on your root account**” option, then **Manage MFA**, then select **Multi Factor authentication(MFA),** and then select “**Activate MFA”,** then select **“Virtual MFA device”** option and click on **Continue**.



Then it will show a QR code.



Then we need to install AWS MFA compatible application on our smart phone, PC r other device. To know the compatible application, click on the “here” option. <https://aws.amazon.com/iam/details/mfa/>



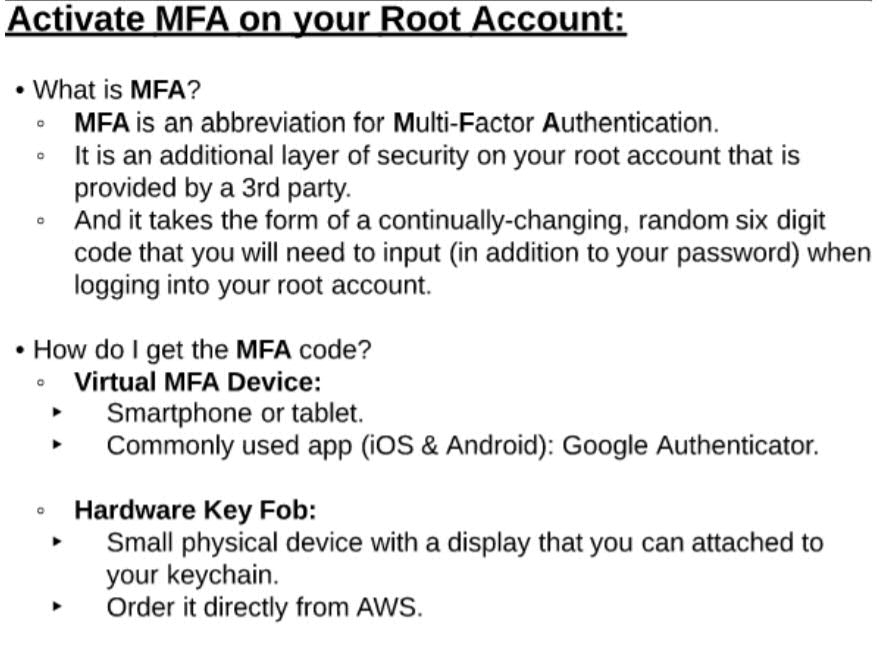
Select the type of authenticator. And also install that authenticator in your phone. For example, if the phone is Android, then Select “Google authenticator” for Android, then in our smartphone we have to install google authenticator from play store.

Then in the Aws account, after selecting “**A virtual MFA device”** option and click on next.  
Then open the authenticator app from your phone, then scan the QR, so it will generate 2 consecutive authentication codes, those need to be entered in the AWS UI and then click on “**Assign MFA**”.

This QR codes are generated by the authenticator apps install on our phone.

To test this, logout from the account and try to login. So after entering the username and password, then it will ask authentication code. Then only we will be able to login to AWS account.   
In corporate lever, every user account has to eb done with MFA.

This code is generated by the authenticator aps install on our phone.



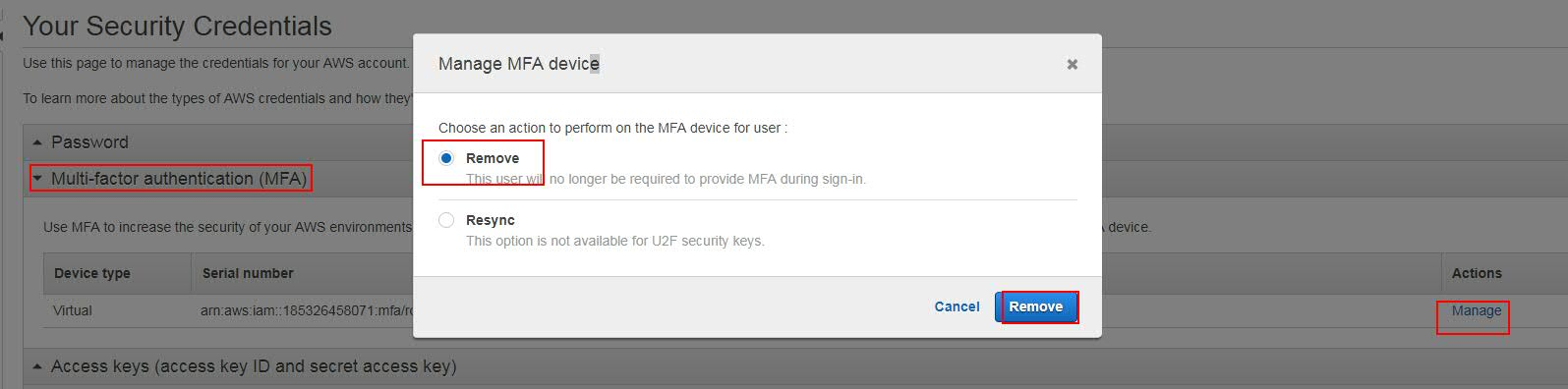
**Disabling Multi Factor Authentication:**

Login to AWS console using **Multi factor Authentication** and go to IAM.

Then select “**Activate MFA on your root account**” and then select “**Manage MFA**”.

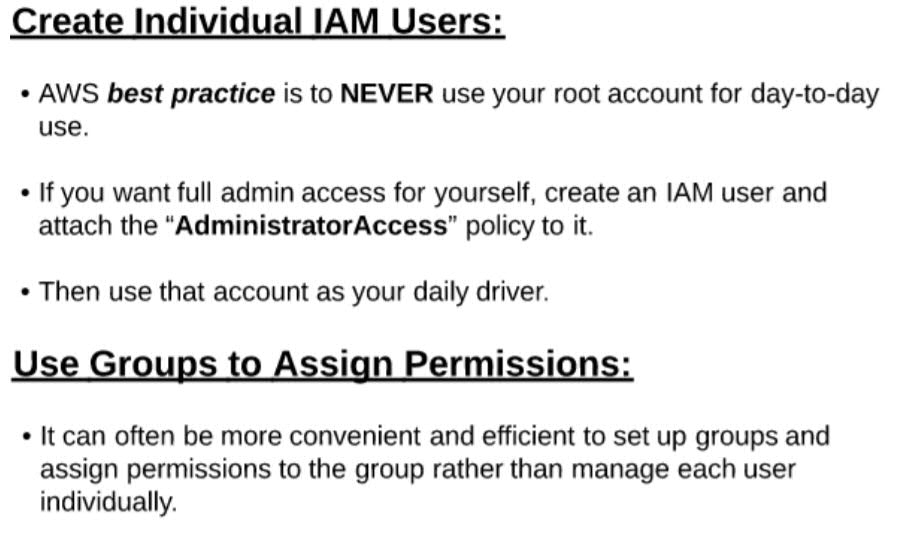


Then select “**Multi-factor authentication (MFA)**” and then click on **Manage**, then select **Remove** radio button and click on **Next**. So that your MFA will be deactivated.



**Let’s create users and add them to groups:**

If same permissions are required for many users, then instead of giving access to individually, we can create groups with required permissions and add the users to the group. Let’s create 3 types of group and do some examples.

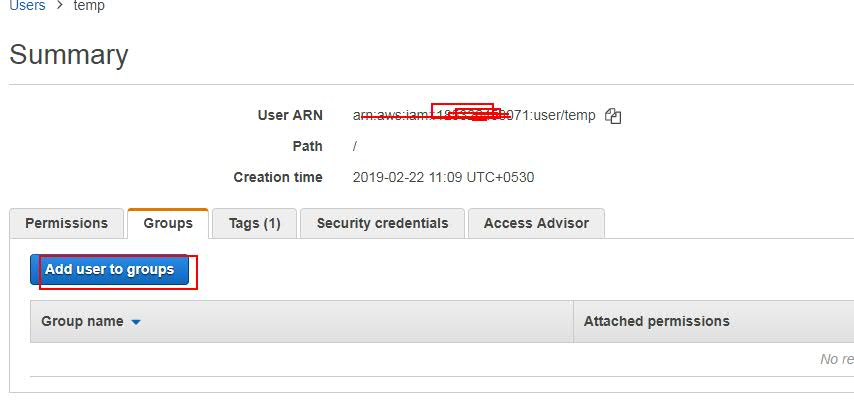
  
To do this, go to **Services** -> **IAM** -> **Groups** -> click on “**Create New Group**”.

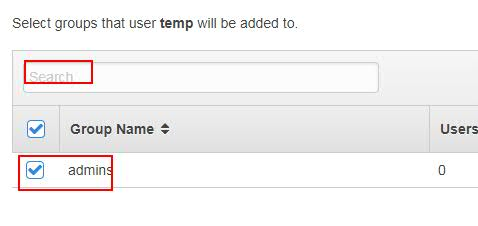
Then give the group name and click on next. **Ex:** **admins**

Then give the required access to the group. **Ex:** **AdminstatorAccess**, and click on **Next**.  
Then again click on “**Create New group**” from right bottom, and click on “**Create New Group**” and give some name. **Ex:** **developers** and click on next  
If the developers needs Read only access to S3, and full access to EC2 and enhanced access to RDS.  
Then search and select these 3 permissions.  
**AmazonS3ReadAccess**, **AmazonEC2FullAccess**, **AmazonRDSEnhanced** and click on **Next**.

Then again click on “**Create group**” from right bottom and click on “**Create New Group**” and give some name. **Ex:** **testers**. As the testers needs least permissions, then give least permissions.   
So give **S3ReadOnly**, **RDSReadOnly** access.

Now, let’s create 3 users.  
For the Go to **Service**s -> **IAM** -> **Users** -> **Add User**. Then give the user name ex:**admin** and give Console and programmatic access and click on **Next**. Then select “**Add user to group**” option and select the admins group.





We created click on **Review** and the click on **Create user**. We can download the csv file to preserve the credentials.  
Repeat the same steps developer and tester users and add them to groups.

Verify the access by logging with 3 types of users from console and trying to access S3, EC2 and RDS.  
Whatever we have done in AWS console UI, those can be done using aws cli. For this, go the aws-shell and try few commands.  
Ex**: aws iam list-users**

Here you can see the arn for each user.

Search in google for **aws cli iam commands** and try few more commands.

<https://docs.aws.amazon.com/cli/latest/reference/iam/index.html>

Practice on aws cli (aws-shell):  
Create an Administrator user from UI console and then do below steps. Don’t use root user.

1. Create user.

Ex: **aws iam create-user --user-name myfirstuser**

1. Create group.

Same how we created user

1. Assign user to a group.
2. Assign some policies to user.

**To get the arn list**: **aws iam list-policies | more**

Find the suitable policy and give the user

**Ex:** **aws iam attach-user-policy --user-name myfirstuser --policy-arn arnfullnumber**

**Ex:** **aws iam attach-user-policy --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --user-name Alice**

1. Delete user and group.

Then verify by trying to login with the user details and secret key.  
Search in google for “**aws iam cli**” for reference doc.

iam, s3, ec2 what we use at the starting are called services.  
After service, list and describe are the starting words we need to use get the complete details under Service. Ex: **iam list users**, **ec2 describe-instances**, **ec2 describe-vpcs**.

<https://aws.amazon.com/getting-started/tutorials/backup-to-s3-cli/>

**ARN (Amazon Resource Names)**ARN is to give unique permissions for anything in AWS. Suppose, if we want to give access to only 2 buckets out of 10, then is can be done using ARN.

Anything we create in AWS will get a unique id. For example, if we create a user or an ec2 instance or a s3 bucket, for everything an unique id will be created. This is very useful when we work with CLI.

**Note:** In AWS, more priority is given to Deny. For example, if a user contains high permissions as individual user and the same person is in a group which has more permissions. Then for that user the less permissions will be applicable (individual access) because AWS give more priority to Deny permissions.

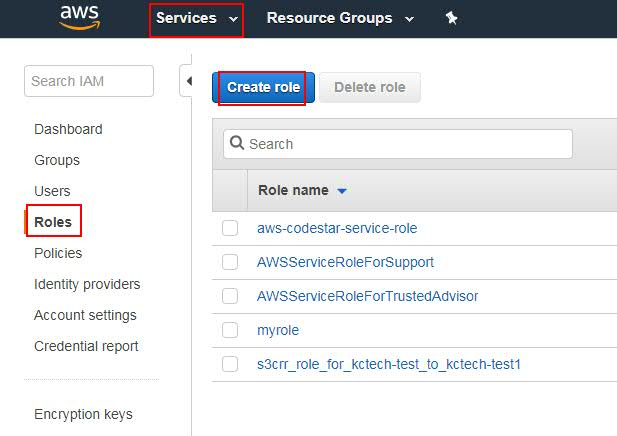
**Roles:**

Roles are generally permission given to services. For example, EC2 instance needs only Read permissions to S3. Then we create a Role for EC2 with S3 Read only permissions.

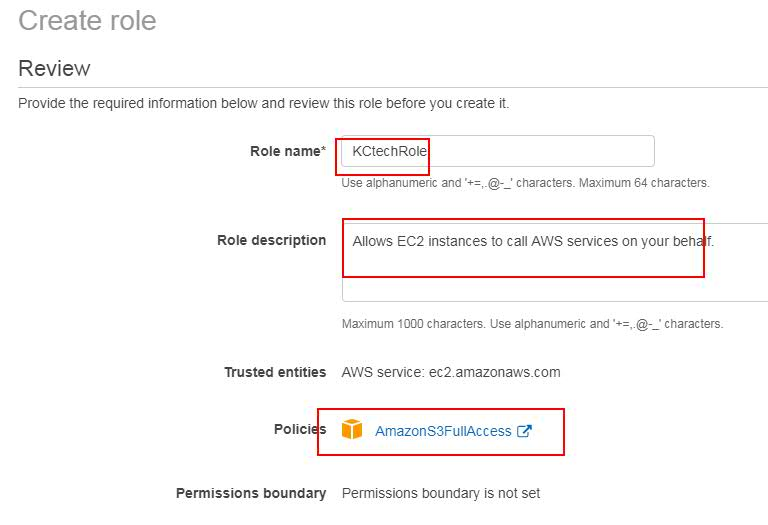
Let’s create a Role:

Login and Navigate to **Services**-> **IAM** and then click on **Roles**. There you can see few predefined Roles. We can also create Custom Roles.

To create custom Role, click on **Create role**



Then select the service to which you want to create Role (**Ex: EC2**) and click on **Permissions**. Then select the type of Policy you want to grant (**Ex: AmazonS3FullAccess**), then click on **Next:tags.** Then give the tag if required. Then give the **Role name**, required **description**, and click on **create role**.



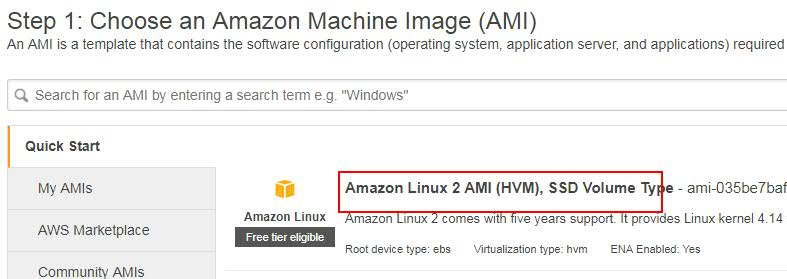
Then you can see the created role.

**Applying Role**:

As we created the role to EC2, let’s go and attach to an EC2 instance.

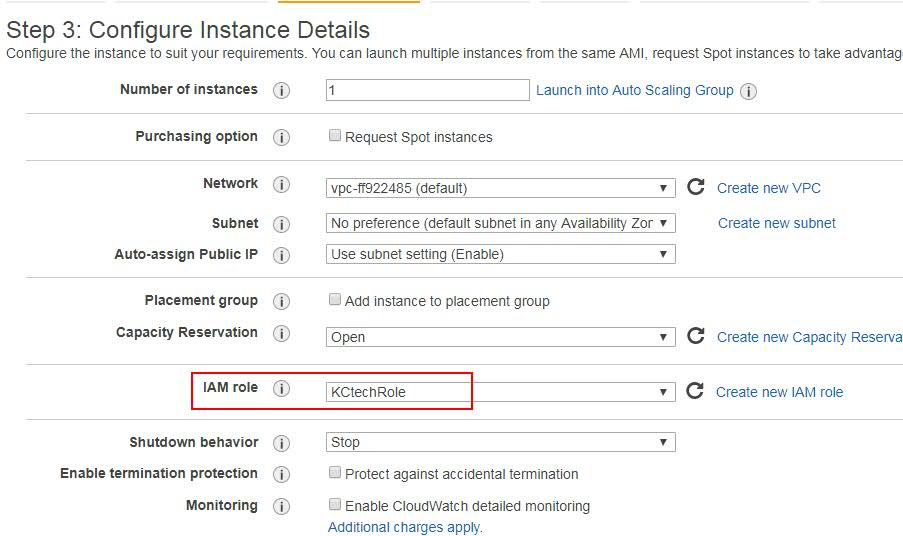
For that, go to **Services**->**EC2**, then select **Launch instance**, select and any instance (Ex: RedHat).

I am going to select Amazon Linux Ami, as AWS CLI is already installed. For other type of instances, we have to install AWS CLI manually.



Then select t2.micro Instance Type and click on next.

Then select the **IAM Role** what we have created.



Then continue by clicking on **Next** and launch the instance and connect from local using Pem file.

Then switch as root user and run this command: **aws s3 ls**

You will be able to see the list of buckets in S3, even though we haven’t done any configurations related to AMI. This is because as we attached Role to the EC2 instance.

**This role can be reused to multiple instances.**

**Note**: Once we assign a Role to a Service, then we cannot change or delete the role to the Service. But in the same role we can change the policies. And only 1 role is applicable to a machine.

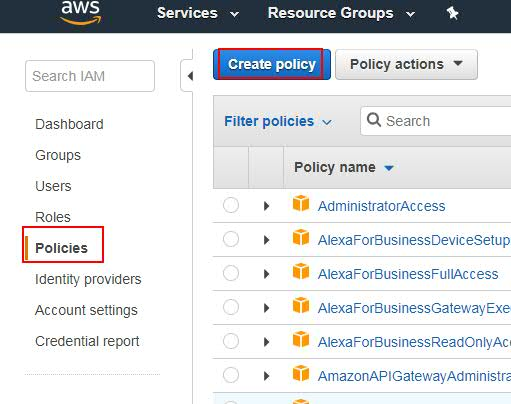
**Note:** We use roles, so that we can avoid human intervention to configure AWS cli. And we can reuse the roles to multiple instances. For example, if an ec2 instances need to move the generated files to S3 bucket, then the developers who has access to EC2 can use S3 to check their files.

**Note:** We can delete the role when it is in use, so that it will be detached from the Instance.

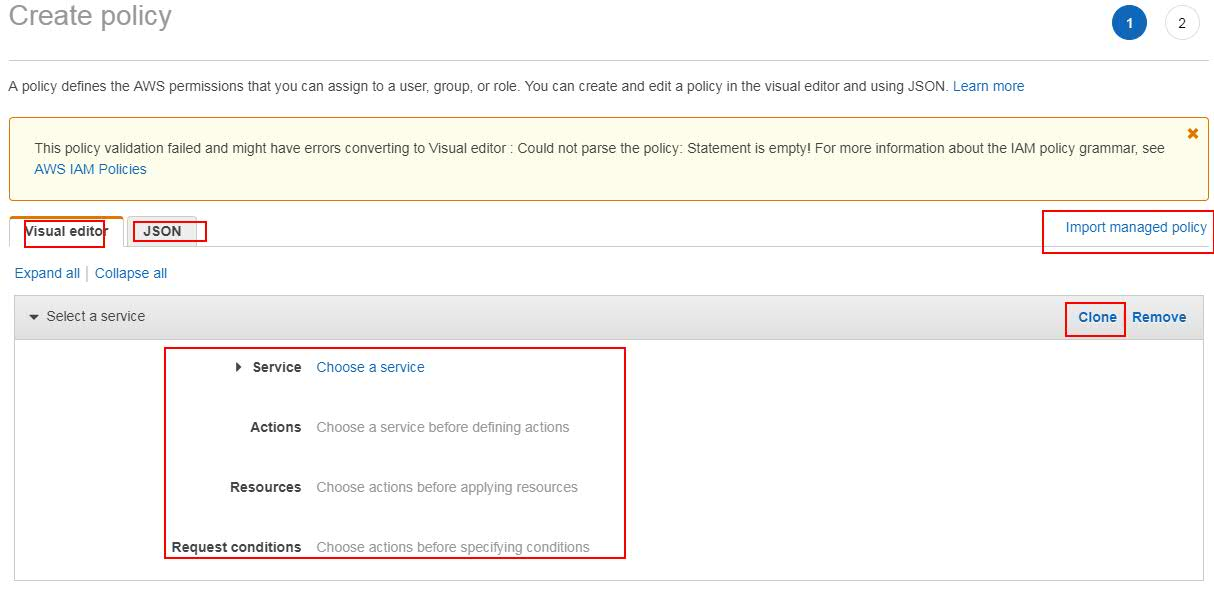
**Policies:**

In a policy, we can give an explicit **Allow** or **Deny** to a particular set ARNs or also to a particular set if users/groups.

**Creating Policy:** Go to **Services** -> **IAM** -> select **Policies**, then select **Create Policy**.



Then you can see the options like below.



There you can see options which means:

**Visual Editor:** Where you can create policy using Ui options.

**Json**: To validate the Policy which is created through json.

**Clone**: Which will create one more set of policy with the same configurations.

**Import managed policy**: To get a policy from the existing one. So, it will get the configurations of the selected policy.

For testing purpose, just select the **Import managed policy** option and import one policy like **AmazonS3FullAccess** and observer the json it generated.

{

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

"Statement": [

{

"Effect": "Allow",

"Action": "s3:\*",

"Resource": "\*"

}

]

}

Here you can observer 3 values: **Effect: Allow, Action: S3: and Resource:\***.

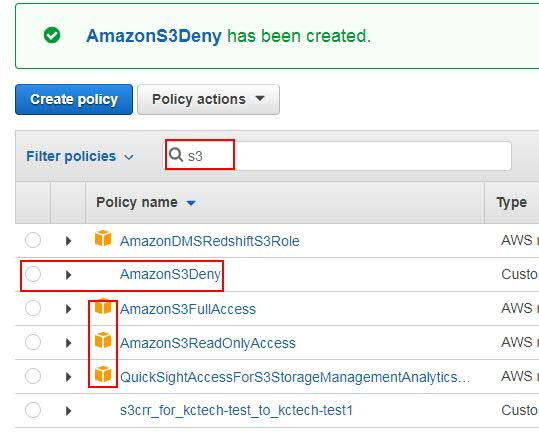
We can create our own policy using the same Json.

Let’s create a policy for S3 Deny access.



Then **Review** the policy. Give a name to it and click on **Create Policy**.

Then you can see the policy created. You can select this policy to any User or group or Roles etc.



We can differentiate Amazon and our custom policies by seeing the Amazon icon.

Now click on the created Policy and observe the json.

Also click on **AmazonS3FullAccess** policy and observer the json. You can see what are allowed and Denied.

* **Version** – Specify the version of the policy language that you want to use. As a best practice, use the latest 2012-10-17 version.
* **Statement** – Use this main policy element as a container for the following elements. You can include more than one statement in a policy.
* **Sid** – Include an optional statement ID to differentiate between your statements.
* **Condition** (Optional) – Specify the circumstances under which the policy grants permission.

**Effect** – Use Allow or Deny to indicate whether the policy allows or denies access.

**Action** – Include a list of actions that the policy allows or denies.

**Resource** – Specify a list of resources to which the actions apply.

Simply to say, Effect is to allow or Deny access.

Action is the type of Service.

Resource is the type of resource under Service.

**Note**: For example, if we want to deny access to only one Bucket under a Service. Then in the Resource, we have to mention the bucket ARN. So that access will be denied to only that Resource. We can separate many resources by comma separated.

\* indicate any resource of the service irrespective of the ARN number.

As we all know ARN is unique.

**Ex:** To get the ARN of a bucket, go to S3, select the bucket, then you can copy the ARN. **Ex:** arn:aws::s3:::kctech-s3-test

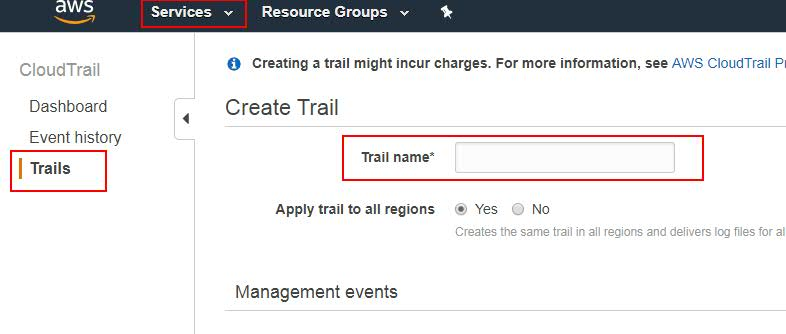
We can create a policy using CLI by creating the json and using it.

To see the ARN numbers of the users: **aws ami list-users**

**AWS CloudTrail:**

We use audit trail to audit, then we use CloudTrial. For example, we want to monitor what done by whom.

Go to **Service** and select **CloudTrial**, then go to **Trails** and click on **Create Trail**.



Then give the Trail name. If you select **Yes** for **Apply trail to all regions**, then it costs more. So, select **No**.

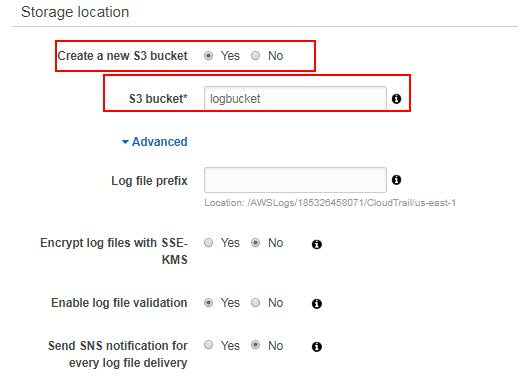
Management Events indicates what type of events it has to Audit.



Under **Data events**, select the buckets which has to be audited.

Under **Storage location**, we have to choose where the audited logs has to be store. We can either create a new bucket or create a new one.

We can give a prefix to the prefix file, so that it will be east for us to track.



Then click on **Create**.

Now you can go to S3 and can find the bucket is create for trial logs. Try to expand and se it.