Table of contents

IAM :

- IAM stands for Identity and access management and it is a crucial part of AWS .
- IAM basically allows us to control who can do what in the AWS environment
- Always stick to "Minimum required access policy " .

Policies

- What is IAM policy : A set of rules that, under the correct conditions, define what actions the policy principal or holder can take to specified AWS resources. To put simply Who can do what to which resources and when
  - "Who" : "Who" is trying to do stuff? This can be a User, Groups of users and "roles."
  - "What" : "What" actions can the "Who" take? Run EC2 instances? Put objects to S3? Put logs to cloud watch?
  - "Which": "What" actions can the "Who" take on "Which" resources? So the "Who" can put and get objects to S3? But to which S3 buckets? All of them? Only ones in us-east-1?
  - "When " : If the IP matches a certain range of IPs? If the date-time is before a particular day? If the AWS user's username includes the string "cheese"?
- Policies are core of IAM. These are basically json documents which dictates the what level of access any entity has (user,group,role)
- AWS keeps some basic polices already created in the account which we can use to get started , but we should avoid using the policies as the access granted in the same are very elevated
- Custom policies need to be created which restrict access to certain resources with specific actions
- Sample policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        }
    ]
}
```

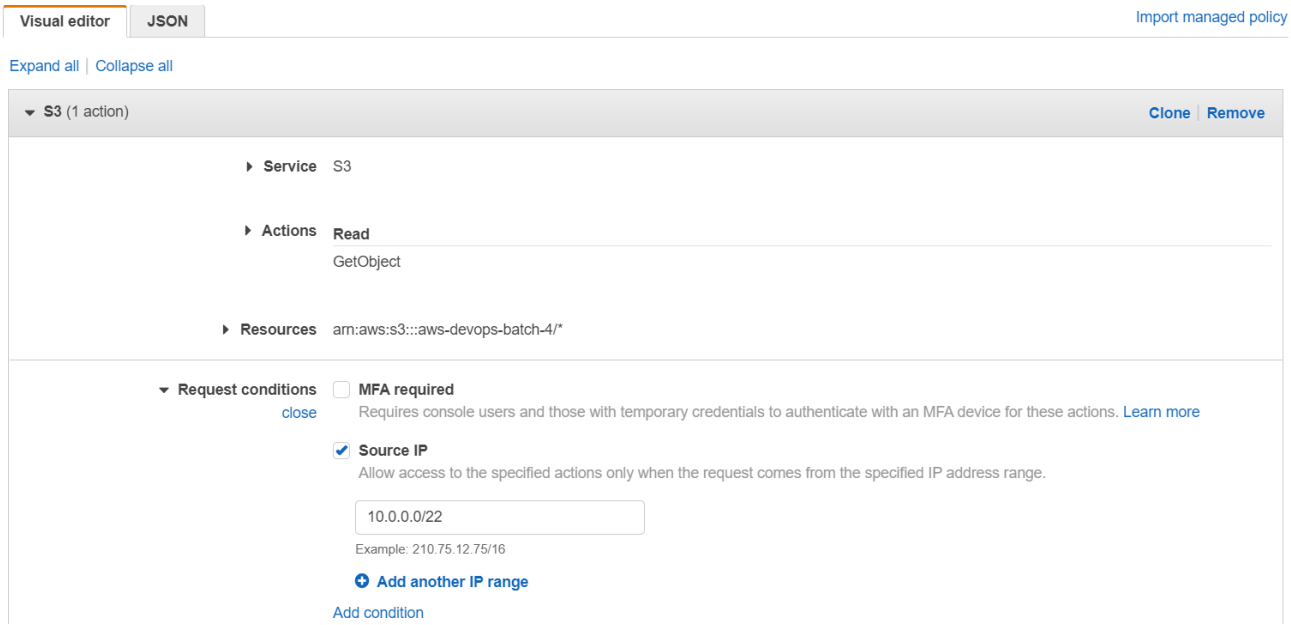
- "Statement": Remember: who can do what to which resources... and when. This is the meat of Policies. This can be one of those statements or an array of many.
- "Effect " : Either Allow or Deny. If we used Deny in the above example, it would just flip the policy. We would deny the user to do specific action on the specified resource
- "Action" : The "What." In above example , any action related to S3 could be performed since we are using wildcard "*"
- "Resource" :The "Which." Which resource they can do "what" to. In above example , any action on any S3 bucket
- Let us now try and create a more customized policy which will be much restricted

Creating custom policies

- We can create policies by going to IAM console and navigating to policy
- Click on create policy
- We can create a policy by either using policy editor or writing a JSON. For starters we can stick to policy editor
- Select the service for which we want to give access to. In this case , we will use S3
- Once the service is selected, we need to select the action that needs to be performed. For this instance we will select getObject.
- Once the action is selected , we can select on what resource that action will be applied to .We will create
- Reqquest conditions are a way to add additional confitions which could be added on top of the policy . for ex source ip . Let us add a request condition for source ip with ip as "10.0.0.0/22"
- Once defined , click next and add a name to the policy and create it
- After creation , these polices can be attached to an IAM user , group or Role
- Policy json will look something like this

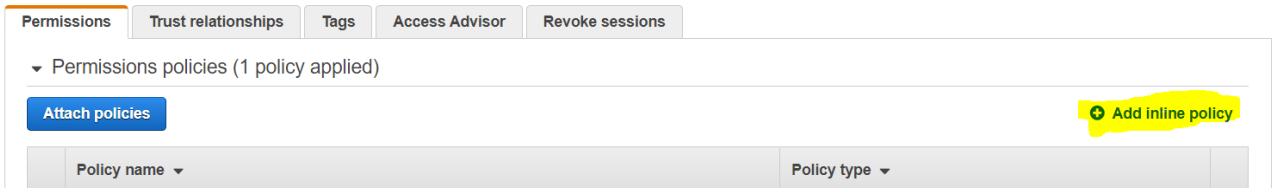
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::aws-devops-batch-4/*",
            "Condition": {
```

```
                "IpAddress": {
                    "aws:SourceIp": "10.0.0.0/22"
                }
            }
        }
    ]
}
```

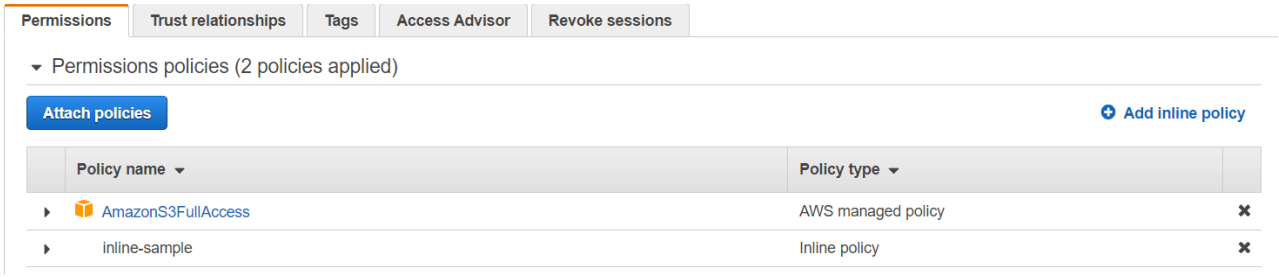


managed policy Vs inline policy

- Managed policies are reusable i.e. they are a seperate entity and has its own ARN
- Inline policy is attached to a role/group/user and cannot be shared or resued . It does not have its own ARN



- Once clicked on create inline policy , we can follow our regular approach to create the policy. Once created it would look something like



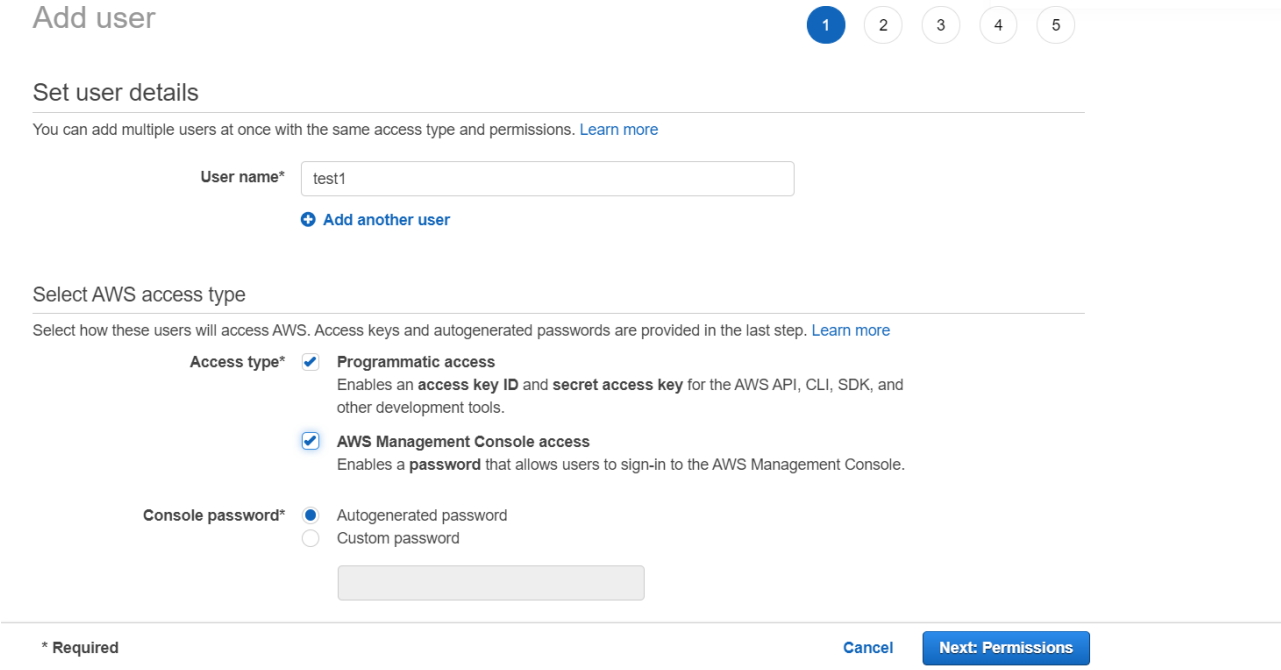
- When to use Inline policy
    - When we want to assign a user/role/group some permissions which we do not want to be reused for any other entity , we should go with inline policy

IAM user

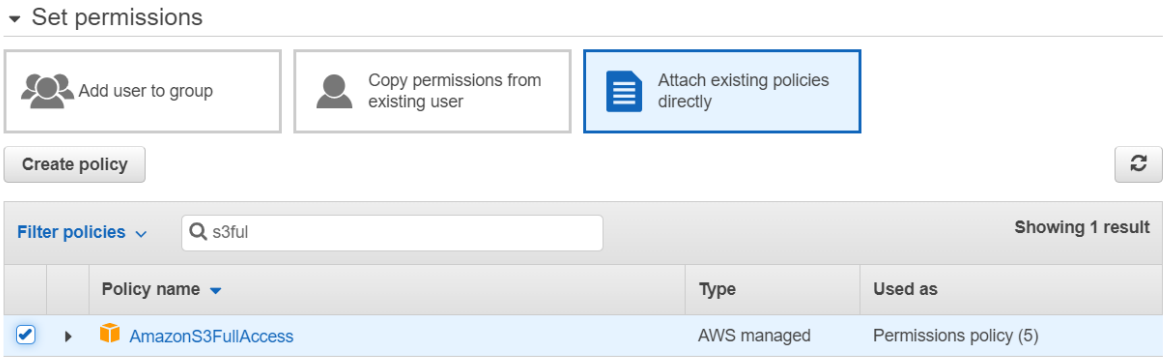
- IAM users are a great way to add multiple users in an AWS account . It can be either a programmtic access , console access or both
- Programmatic access grants us access key ID and secret access key , which can be configured anywhere ,, hence these need to be avoided whhenever we can . We'll be looking additional ways later
- Console access is where users can login using aws console . Users get a seperate url to login where they can enter the username and password which they can use to login

Creating an IAM user

- Navigate to user in IAM
- Click on add user , and select the access type (programmatic or console or both). Let us select both and name it test1



- In the next tabwe can add permission . IF we have created a policy already we can attach that from "attach existing polices directly ". For now let us add "s3fullaccess"



- We can create the user and add the permission later on as well
- Add relevant tags and click next
- Click on create user , if you have selected programmatic access make sure you download the credentials file as the secret access key is only showed once .
- When to create an IAM user
    - In a scenario where you want to share access to aws account to other entites , that may be programmatic or console (We should avoid giving programmatic access whenever possible). For

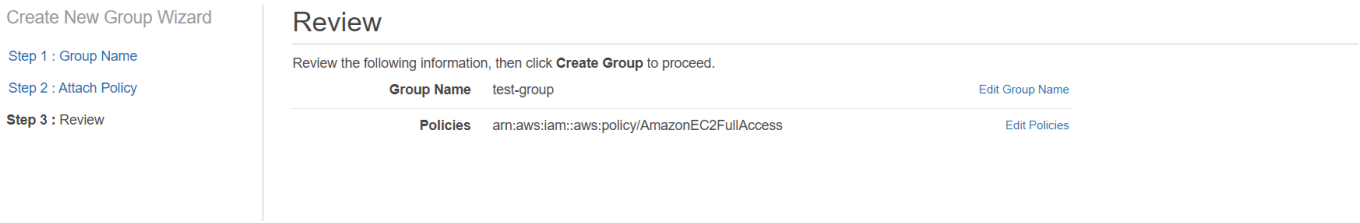
ex: New user added to company , external client/vendor

Group

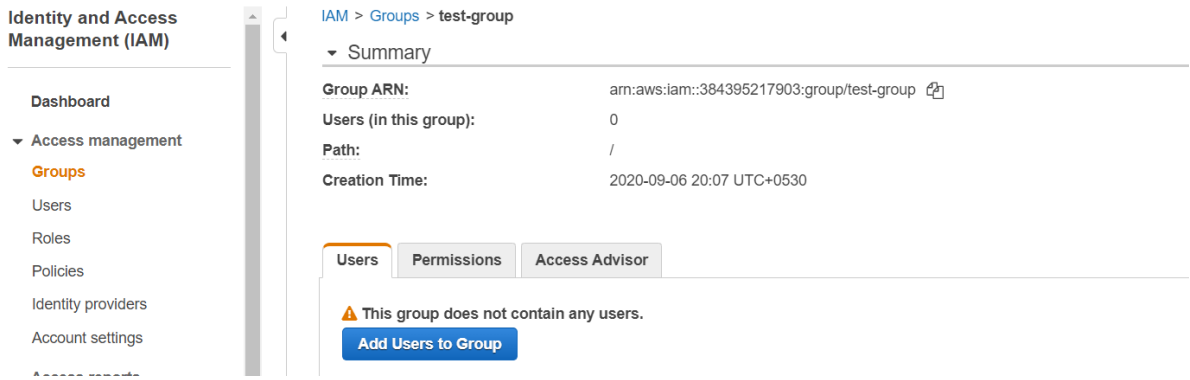
- Group is a conventient way to group iam users together.
- Groups can have IAM policy attached , if we add user to group , the policy will be directly applied to all users
- If user already has another access, group access will be added on top of it . So user will have both permissions

Creating a Group

- Click on add new group in IAM
- Give a group name and click next . Let us name it test-group
- Attach a policy if it already has been created. For now we can add "ec2fullaccess"



-
  - Review and create . We can add users once the group is created



Roles

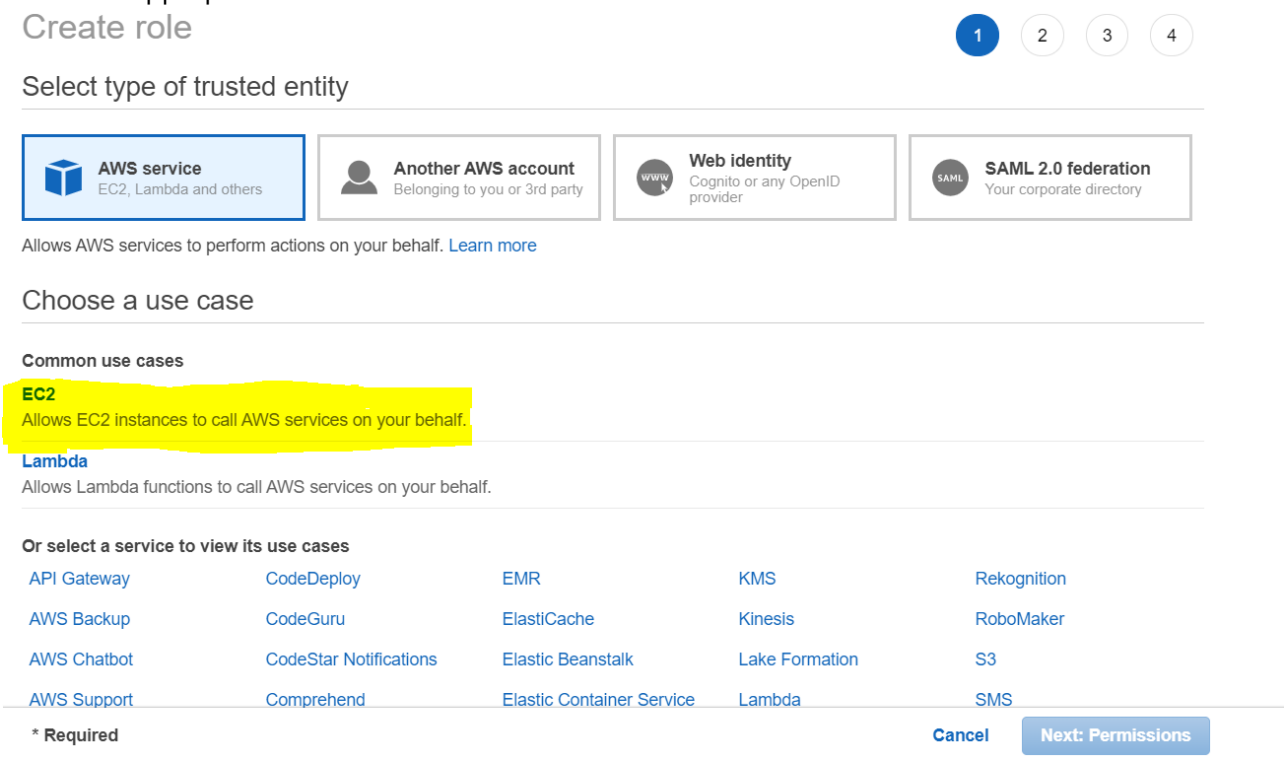
- From an admin perspective , roles are something that we come across very often
- There multiple type of roles that we can create
- We'll be focusing on service role primarily
- Service role are used for aws services to interact with other aws services for ex. EC2 to interact with S3
- Using service role we can usually bypass the need of using access key and secret access key . Without a role , we would have had to configure aws-cli on the instance to access S3 or any other service . With having IAM we do not need to use any access key ID or secret access key at all .

Creating service roles

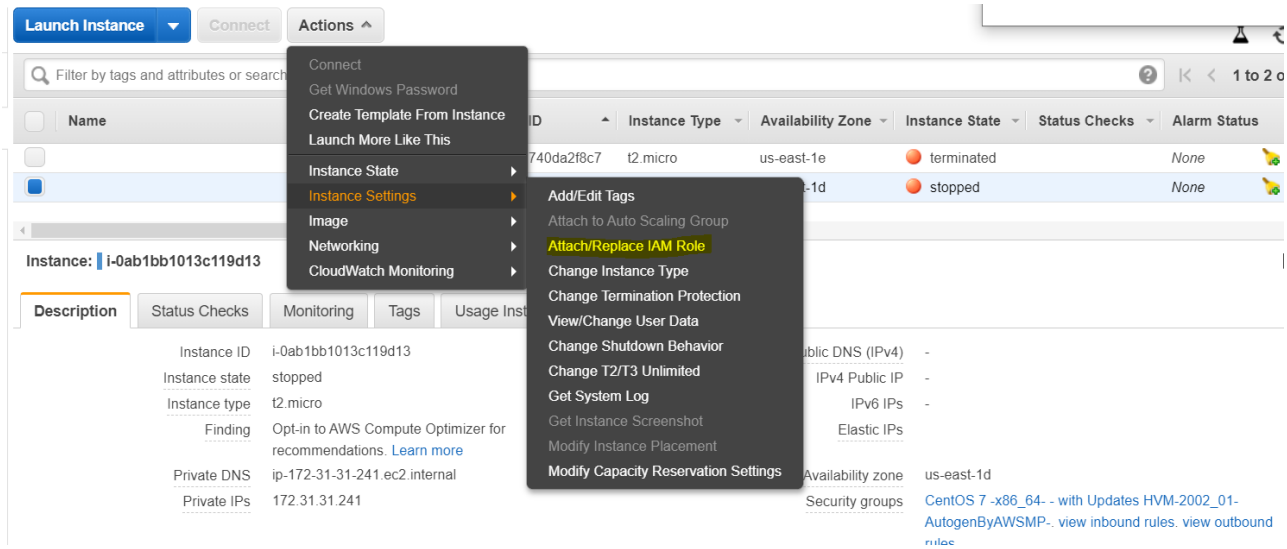
- Navigate to roles and click on create role

- By default "AWS service " section is selected

- Select the appropriate service . for ex EC2



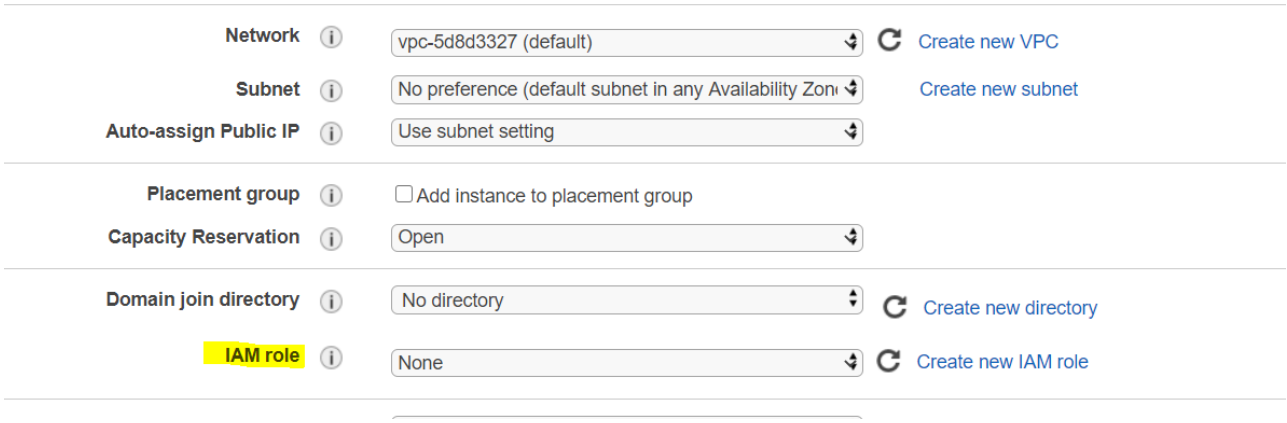
- This dictate that the role that is going to get created can be attached to an EC2 instance .
- Click on permissions
- Here we can assign a policy which states what all permission the ec2 will have . as of now let us select s3readonlyaccess
- Follow the tag and review process and create the role
- Once created , we need to attach the role to the ec2 instance .
- Navigate to ec2 , select an instance and click on actions and select instance settings
- Under instance setting we will find "Attach/replace IAM role" , click on it



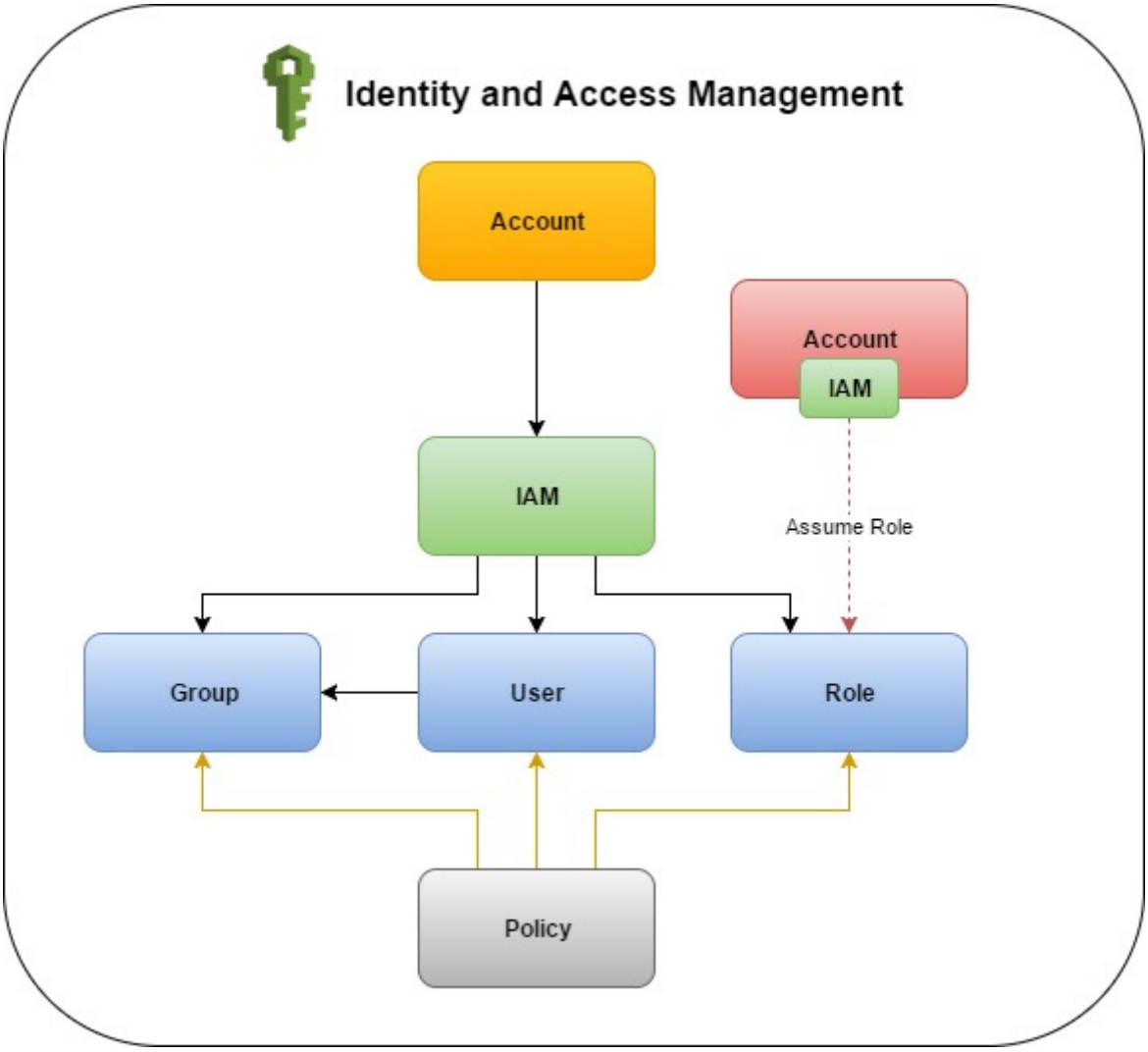
- Select the role we have recently created and click apply
- Test the access by logging on the EC2 and using aws cli for S3 . For ex

```
aws s3 ls
aws s3 abc.txt s3://bucketname
```

- These service roles take away the need to use aws access key and secret key .
- We do not need to configure the aws cli, we can directly start using it
- IAM role can also be attached during the launch of an EC2

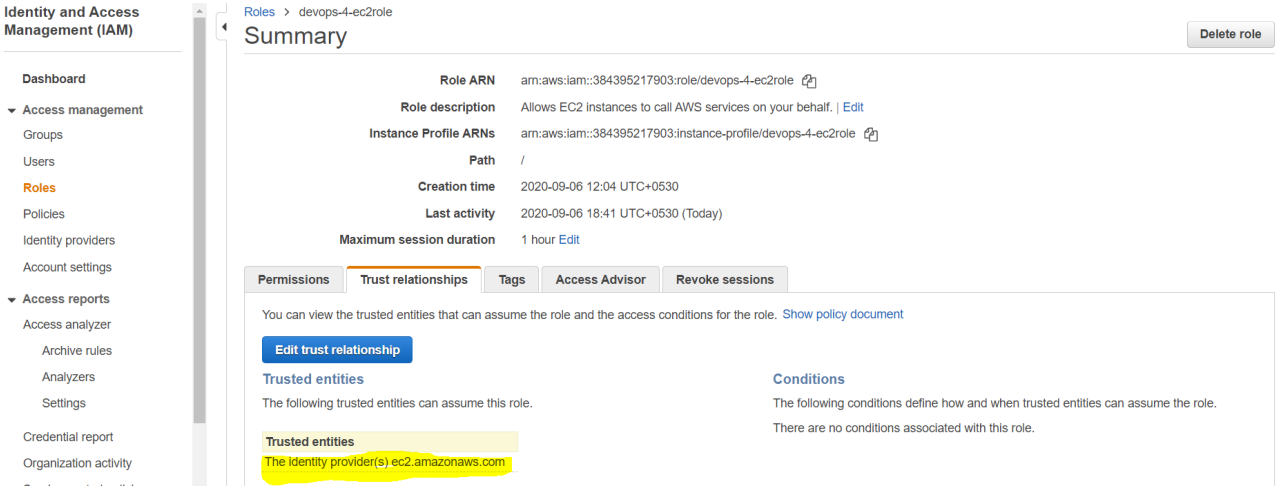


IAM policy: user/group/role flow



Trust relationship

- Trust reationship dictates which entity can assume a certain role. Or simply use that role and the permissions assocaited with it
- Trust relationship can be a service or role or user from same/other aws account as well
- As of now let us look at the trust relationship of the role which we had created earlier for ec2



- As seen above , the trusted entity is "ec2.amazonaws.com"
- generic trust relation usually looks like this

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- We can also use one policy to define 2 trust entities

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service":[
            "ec2.amazonaws.com",
            "lambda.amazonaws.com"

        ]
      },
      "Action": "sts:AssumeRole"
    }
```

```
    ]
}
```

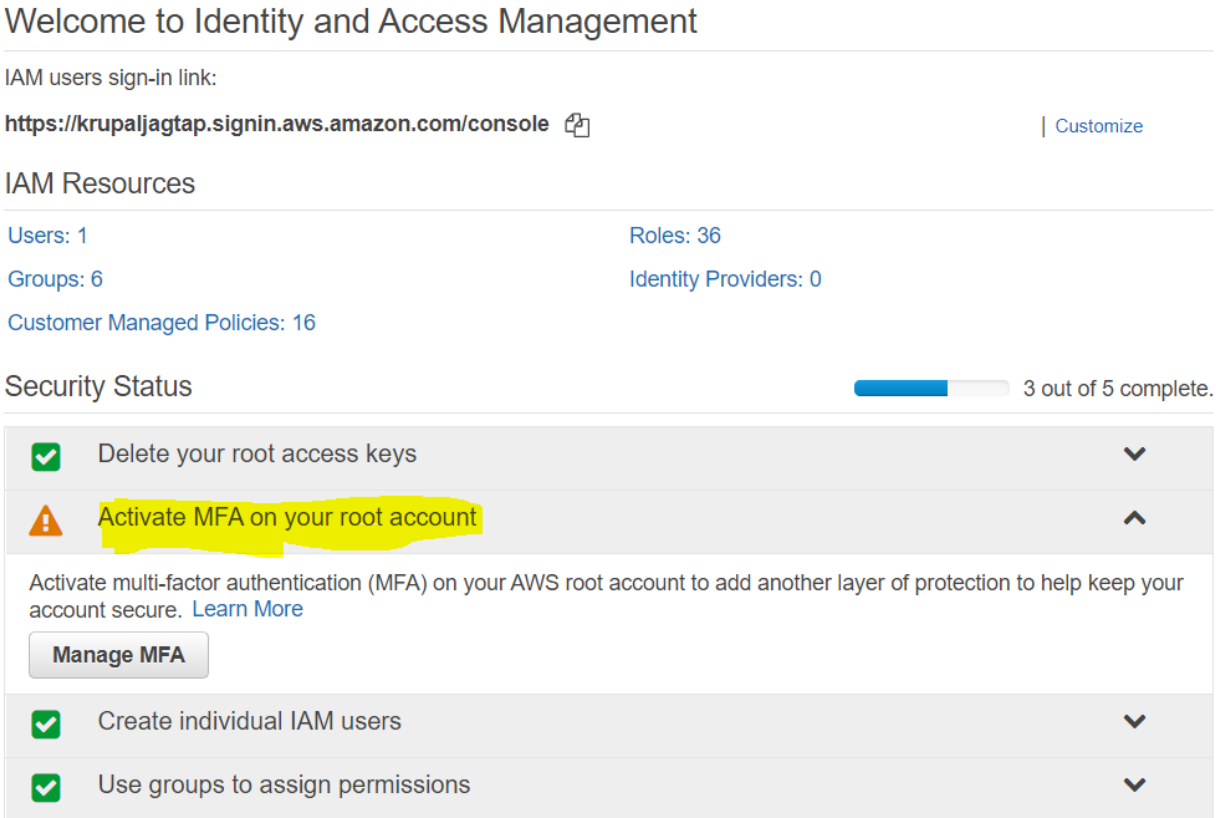
- Above policy defines that ec2 as well as lambda can assume the role

MFA

- Multi Factor Authentication is a common way to add additional layer of safety to you login to aws console
- It works on similar principals as our regular gmail login.
- MFA can be assigned to root user as well as IAM user

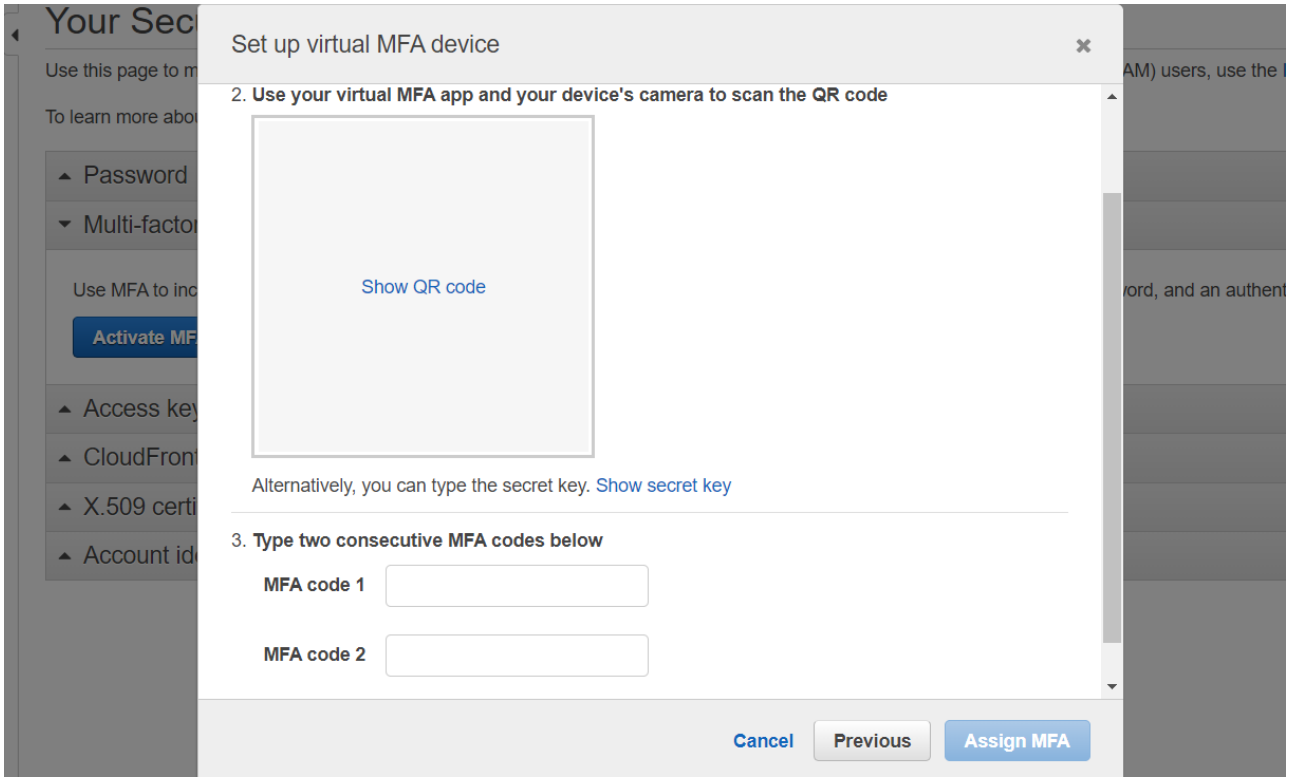
Assign MFA to root

- Navigate to IAM dashbaord
- Select activate MFA for root account. (MFA can be enabled for root user as well as IAM user)



- click on manage MFA and select multi factor authentication
- Select Activate MFA and continue with virtual MFA device
- Download google authenticaor on your phone (available both on IOS and android)

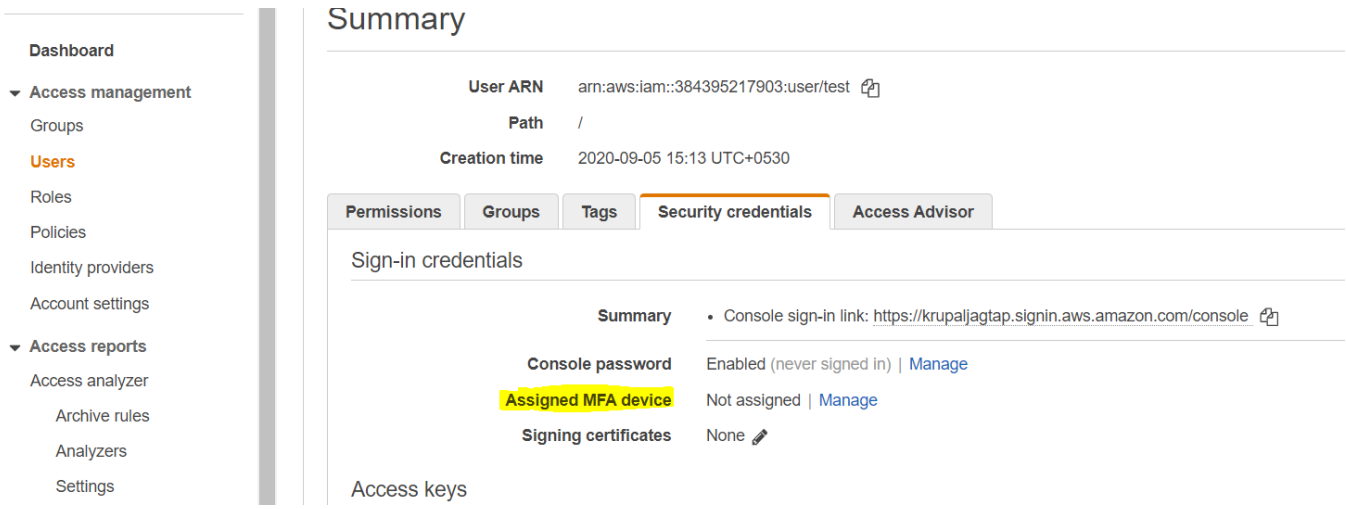
- On aws console click on show qr code , scan it from google authenticator .



- Once scanned , enter the consecutive 2 numbers popping up in google authenticator on the console. and click on assign MFA

Assign MFA to IAM user

- Select an IAM user , and navigate to security credentials
- Select manage in front of "assigned MFA device "



- Follow the same procedure as we followed for root user going forward

IAM cli

- Consider practicing below cli commands which could be useful for extracting information

```
aws iam list-users
aws iam list-roles
aws iam attach-user-policy --user-name <value > --policy-arn <value>
aws iam   delete-user --user-name <value>
aws iam   add-user-to-group --group-name <value> --user-name <value>
aws ec2  associate-iam-instance-profile --iam-instance-profile <value> --instance-
id <value>
```

Points to consider

- IAM has to be one of the most cruial services to be used in an aws account , make sure we always follow the "minumum required access policy"
- Try and avoid using access key and secret key wherever possible. consider using IAM role as an option
- Avoid using aws managed policies whenever possible. Create custom IAM roles to suit the use case
- Always restrict policies based on either ARN or source IP
- Write privileges should be given with care . With respect to S3 however , even read access should be given very carefully