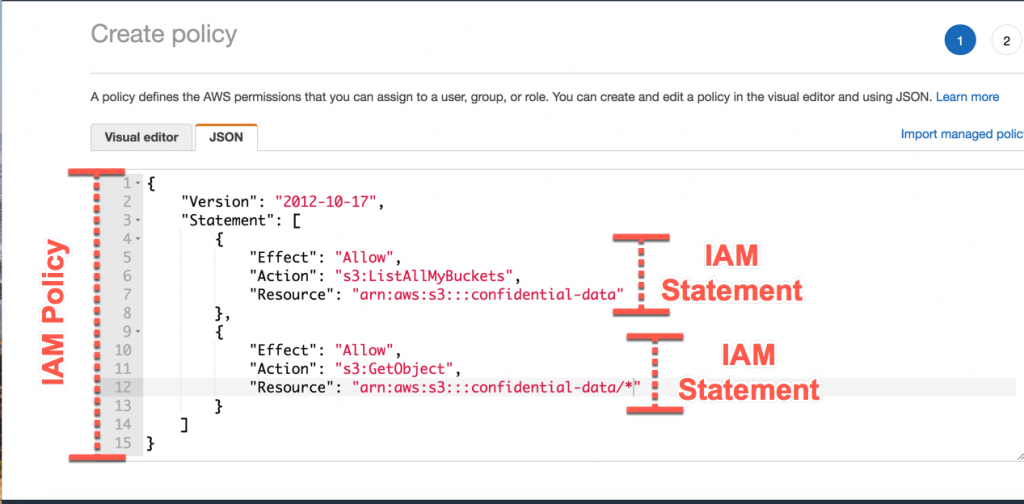
**Identity And Access Management IAM Global Service**

**Pre-Introduction**

* AWS Identity and Access Management (IAM) provides fine-grained access control across all of Amazon Web Services. With IAM, you can specify who can access which services and resources, and under which conditions. With IAM policies, you manage permissions to your workforce and systems to ensure least-privilege permissions.
* First and foremost, to be part of the IAM global service. The root user must add you to the IAM service
* **Note**: The AWS’s main user aka the Root user has access to all the different services. Therefore, The root user can create users that have access to a specific AWS services and specific features of those services. And the root user makes users that have access to all the services and as much privileges as the Root User, expect the ability to delete the root user. If a user has no access to a specific service. Then, they will not be able to go to that service.
* For example, if user43 has no access to IAM then that user will not be able to go to access anything on the IAM (Identity Access Management) service

**Introduction**

* The IAM is in the AWS Management Console:
* The AWS Management Console is a browser-based GUI for Amazon Web Services (AWS). Through the console, a customer can manage their cloud computing, cloud storage, web services, and other resources running on the Amazon Web Services infrastructure.
* In IAM we create the users(people within your organization) and assign them to groups
* Groups may contain individual users and not a combination of groups
* A user may belong to multiple groups and a user may belong to no groups (not best practice)
* Why do we create users and groups ?
* Because we want our organization to use our AWS account and we want to give them certain permissions for specific items
* Users or Groups are assigned documents that are called policies and are written in JSON
* JASON -> JavaScript Object Notation, more commonly known by the acronym JSON, is an open data interchange format that is both human and machine-readable. Despite the name JavaScript Object Notation, JSON is independent of any programming language and is a common API output in a wide variety of applications.
* In AWS we apply the ‘Grant Least Privilege Principal”. Which states that users should not be given more permissions that what they need
* Below is an example of a policy
* The group or users are allowed to do the following API calls/actions listAllMyBuckets & allowed for GetObjects



**Creating Users/Groups**

* 1. Search for IAM and go to the IAM dashboard
* Steps for creating a user + Creating a groups
* 1. Click on Users in the left-hand window of Access management
* Even if you are working alone. Create an admin user. This is a better practice, because you want to avoid using the **root user**, because that user can make a lot of damages and we should not always use the **root user** and this is for security purposes. Therefore, create a secondary user with almost all the root user’s privileges. The user I created I has the name mohamd
* 2. Add the user to a group. Therefore, create a group called admin (You do not need to create a new group for each new user). If a group exists, you can add the users to that group
* Add the AdminstratorAcess policy name to the admin group. This will allow all the users in the admin group to have AdminstratorAcess
* 3. Add the tags. This is optional. IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this use
* To add a tag. Type what ever you wish in the Key and type whatever the Key corresponds to in the Value. For example, Key is Department. Then the Value is Engineering
* 4. Download the csv, which contains all the credentials or email the instructions to another user
* 5. A new username will be created. In this case the username is mohamd. Therefore, when we sign in next to AWS we can sign-in using the **IAM user** instead of the **Root user** option. When click into the **IAM user** you will be required to enter:
* **Account ID**
* The account ID. Is a number that is given for each individual AWS account. It is not a user specific. However, this number is unique and could be hard to remember. Thus, we can add aliases for it to make it easier to remember. The alias I added was 25541818
* We can always sign in using the original account ID or the alias we made to represent it
* Every time we sign in to the root-user we will see the original account ID we made
* **IAM username**
* The username is the username was added. In my case it is mohamd
* **Password**
* This is a password that was set when the user was created. It is not the same password as the root user. However, in my case when I made the user mohamd. I set the password to be the same as the root user

**IAM Policies Inheritance**

* Policies may be created for a group or individuals. When a policy is created for a group all members of that group will be eligible for that policy.
* An IAM policy is an entity that, when attached to an identity or resource, defines their permissions.
* An individual user can have his own privileges by assigning him an inline policy
* The policies are written in a JSON file. The policy file contains
* A Version number (Required)
  + The version number stands for the policy language version. For example, “2012-10-17”
  + Exp -> "Version": "2012-10-17",
* An ID (optional
  + The ID is an identifier for the policy
  + Exp -> "Id": “S3-Account-Premissions”,
* Statement (Required)
  + This includes one or more statements and statements include:
    - Sid (optional)
      * Sid (statement Id) which is an identifier for the statement, and it is (optional)
      * For Exp -> “Sid” : “1”,
    - Effect, on access
      * This statement indicates if it is an allow or deny access to certain APIs
      * Effect will have either ”Allow” or “Deny”
    - Principal
      * Indicates which account, user, role this policy is applied too
      * For exp -> “Principal” : {“AWS”: [“arn:aws:iam::1235684165:root”]}
        + This principal is applying the policy to the root user
    - Action
      * The list of API calls that the users will be allowed or denied based on the Effects
      * For exp -> “Action” : [“s3:GetObject”]
    - Resource
      * A list of resources in which the Action will be applied too
      * For exp -> “Resources” : [“arn:aws:iam:::mybucket/\*”]}
    - Condition (optional)
      * This is a condition for when the policy should be applied
* IAM Policies Notes:
  + When adding policies to a user you can click on “Attach existing policies directly”. Otherwise, if you want the user to have all the permissions of a group you must add the user to a group by clicking on “Add user to a group”
  + **Important**: To view all the users’ privileges of each AWS service. Those privileges must be added by the Root user. Then, we must click on the service (e.g., IAM) and that will display the user’s privileges sheet. The sheet displays the policies that were attached manually to the user and the policies that were added to the user, because the user is part of a group.
  + To view all the available policies of all web services. You must click on Policies while at any web service (e.g., IAM) and under permissions you can view any policy’s JSON form
    - Like Linux, \* means all. Therefore, “Action”: “\*”. This means the action is on everything
  + If we click on Policies, we can also create our own policies. Either using the UI or typing the JSON file. It is much easier to use the UI
  + Always try to grant users the minimum but workable number of privileges to users. Give users what they exactly need. Not what they want

**IAM Security**

* While in the IAM service of AWS
* There are two methods of setting IAM security for users
* Method 1: Password Policy
  + Password policy is helpful against brute force attacks on your account. However, MFA is better than the sole password policy
  + When setting up a password policy you may control exactly what the password must contain and must not contain aka setting passwords with different options. For example:
    - Setting up a minimum password length
    - Requiring specific character types (At least one upper case letter, lower case, numbers, non-alphanumeric characters)
  + Allowing the IAM users to change their own passwords
  + Requiring IAM users to change their passwords after some time, because you made sure that the passwords expire after some time
* Method 2: The MFA (Very recommended to use it)
  + AWS Multi-Factor Authentication (MFA) is a simple best practice that adds an extra layer of protection on top of your username and password.
  + MFA = The password you own + A code from a physical device you own (code sent to your phone)
  + To have the possibility of MFA, you require an MFA generating token.
  + The MFA is used for at least the Root Accounts and all the IAM users. The **benefits** of MFA is that even if users’ passwords are hacked. Their accounts are not compromised. Because whom every stole the password would need to also steal their physical device that generates the token or code)
  + MFA devices options in AWS:
    - Virtual MFA device
      * Google Authenticator (Only works for phones)
        + Basically, Google Authenticator receives a scanned code from the app that is setting up 2FA, and then it produces a 2FA code to access the app or online account. However, since it only works on Android and iOS devices, it cannot work on desktop PCs directly.
      * Authy (Multi-device Authenticator)
        + Multiple tokens on a single device, this allows for multiple users and multiple accounts
      * Universal 2nd Factor (U2F) Security Key, (e.g., YubiKey)
        + A physical device by a third-party company, namely: YubiKey by Yubico. YubiKey supports multiple root and IAM users using a single security key. (You do not need as many keys as users)
      * Hardware Key Fob MFA Device
        + For example, a device is provided by a third-party company known as Gemalto
      * Hardware Key Fob MFA Device for AWS GovCloud(US). This if you are working in the American gov cloud
        + For example, a device is provided by a third-party company known as SurePassID
* Setting up security in AWS for specific services
* The Root user can set up the password’s settings and the root user’s MFA
* Each Individual user can setup their own individual password and MFA
* Setting a Password Policy step:
  + 1. As a Root User in an AWS service (e.g., IAM). Click on ‘Access management’
  + 2. Click on ‘Account settings. Then, click on ’Change password policy’. Then that will change the policy for the IAM service for all the users that do not have admin privileges
    - Any new user will have to follow the password policy and any existing user must comply as well
* Setting up MFA in IAM service
  + Each user can set up MFA on their own
  + While in logged in to the user mohamd -> Click on ‘Security Credentials” under mohamd @25541818
  + Click on “Assign MFA device”

**Accessing AWS**

* There are three options to access AWS
  + AWS Management Console. Through a password or password + MFA
  + AWS Command Line Interface (CLI). This type of access is protected by an access key
    - The AWS CLI
      * This section on the Command Prompt or Terminal just discussing the ability to access the command line and to do that you need to have access to IAM. From here you can access other instances like EC2 instance
      * Is similar to the Linux command line. However, it allows us to communicate with AWS services using the command line shell
      * The CLI allows users to have direct access to the public APIs of the AWS services
      * The CLI enable users to develop scripts to mange their resources. Also, it is open source, and its code is available on GitHub
      * The CLI is similar to the management console. Both provide similar information. However, the CLI is better IMO
  + AWS Software Developer Kit (SDK). This type of access is protected by an access key. Also, SDK is used when you want to call APIs from within your own application code. It allows you to manage your AWS service using a programming language
    - SDK:
      * It is a language specific API. Therefore, different libraries of SDK for different programming languages (e.g., Python)
      * It allows you to access and manage your AWS services programmatically
      * SDK is not used within the terminal. SDK is embedded within the application
      * SDK supports mobile and IOT devices (thermal sensor)
      * SDK supports many programming languages
      * In the course we used AWS CLI which is built on AWS SDK for Python. Named BOTO3
      * SDK is used to implement a programming language for AWS
  + AWS Cloud Shell
    - It is an alternative method to CLI to communicate to AWS. It is not available in all regions
    - To access the cloud shell. Click on the cloud shell icon.
    - You can create files in the cloud shell. Just like you would in your Linux terminal and download them. Also, you can upload files into your cloud shell from your computer, by simply selecting the files you want to upload
    - To download a file from the cloud shell.
      * 1. Make sure you know the files path working directory by typing pwd
      * 2. Click download then add the file name at the end of its pwd
    - Once you create a file in the Cloud Shell it will be on the cloud shell unless you delete it
* Setting up CLI in windows steps:
  + 1. Follow the steps from <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>
  + 2. Confirm the version by typing the following in the command line -> aws –version
* Setting up CLI in Linux steps:
  + 1. Follow the steps from <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>
  + 2. Confirm the version by typing the following in the command line -> aws --version
* Generating an access key:
  + Access keys are generated through the AWS Management Console
  + Access Key ID is treated as a username & Secrete Access Key is treaded as a password
  + Access keys are helpful when working within the CLI or when the SDK is used to implement a programming language for AWS
  + Steps to generating an access key & configure the AWS CLI:
    - 1. Login to the user on AWS management console. Do not use the root user. Use a user that have administrative privileges but not the root user. User a user that has access to a specific service
    - 2. Go to IAM
      * This is to generate the credentials for AWS users of IAM access
    - 3. Click on users and select a user
    - 4. Click on “Security Credentials”
    - 5. Scroll down to “Access keys” and click on “Create access key”
    - 6. Configure the AWS CL by typing the following in the command line -> aws configure
    - 7. You will be asked to enter the access key id and the secret access key. Downloaded both as an excel sheet
    - 8. Enter the region. This should be a region close to you for example -> ca-central-1
    - 9. Enter the default output format -> just press enter
    - Note: The CLI is not only for IAM It is for all the other AWS services
  + Notes for the AWS CLI
    - If you type the following in the CL -> aws iam list-users
      * This will list all the users you have that are within your AWS account and have access to the IAM service
  + This section on the Command Prompt or Terminal just discussing the ability to access the command line and to do that you need to have access to IAM. From here you can access other instances like EC2 instance

**IAM Roles for Services**

* Roles are important, because often services are required to perform actions on behalf of the user. Therefore, permissions must be added to AWS services so such actions can be performed
* The definition of a Role, it is an IAM identity you can create in your account with specific permissions. A role is not unique to a user it is assumed by anyone that needs it. Also, roles do not require passwords. Roles are many used to delegate access to users, applications, or services that do not normally have access to the specific AWS resource
* A Role is an IAM entity that defines a set of permissions for making an AWS service requests that will be used by Amazon web services
* is that an IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust. Some of those roles are for AWS services
* The act of assigning permissions to AWS services with IAM roles is what will allow the services to act on behalf of the users. An IAM role has some similarities to an IAM user. Roles and users are both AWS identities with permissions policies that determine what the identity can and cannot do in AWS
  + This will be like creating a new user but a user we don’t use. This user will allow the AWS service to perform work automatically
* Usually, a virtual server on AWS is assigned to an IAM role. Together they make a single entity. Then, when the virtual server attempts to utilize a specific service that service must be permitted within the IAM role that was assigned to the server. For example, IAM role with EC2 instance (a virtual server)
* Common roles are made for:
  + EC2 Instance Roles
  + Lambda Function Roles
  + Roles for CloudFormation

**IAM Roles for Services – Creating a Role Steps**

* 1. While in the IAM page, click on “Roles”
* 2. Click on “Create Roles”
* 3. Select “Type of trusted entity”
  + In the example we selected “AWS service”
* 4. Under “Use Case” type the service you wish to assign a role too
  + In the example we typed and selected “EC2”
* 5. Click on “Next”
* 6. After click “Next” you will taken to the “Add permissions” page. Where, you can add Permissions policies. This basically allows you to choose what the EC2 instance we selected in step 4 will be allowed to do
  + In the example we selected “IAM Read Only”
* 7. Click “Next” . The add the Role’s name and description and similar to creating a user you will be asked to add “tags” . Then when done click “Create Role”
  + The name was entered as DemoRoleForEC2
  + The description did not change
  + No tags were added

**IAM Security Tools to Audit the system**

* The security tools of IAM include:
  + 1. IAM Credential Report ( At Account-level) (You have one account & multiple users)
    - This is a report that lists all the account’s users and the status of their various credentials
  + 2. IAM Access Advisor (User level)
    - This is a report that shows the service permissions granted to a specific user and when those services were accessed
* IAM Credentials report lists all your account's users and the status of their various credentials. The other IAM Security Tool is IAM Access Advisor. It shows the service permissions granted to a user and when those services were last accessed.

**IAM Security Tools – IAM Credential Report Creation Steps in AWS**

* 1. While in the IAM page, click on “Credential Report” and then click on “Download Report” to download the report

**IAM Security Tools – IAM Access Advisor (User level)**

* 1. While in the IAM page, click on “User” and then click on a specific user
* 2. Click on “Access Advisor”
  + This will show
    - when some services were last used by the specific user you selected
* This helps reducing the number of services provided to a user, because there is not point of providing permission to a service that the user does not use

**IAM Best practices**

* Do not use a root account expect for AWS account setup
* For each AWS user assign him/her one real user
* Users may be assigned to groups and permissions can be assigned to individual users or groups
* Create a strong password policy
* Try to enforce MFA
* Create and use Roles for giving permissions to AWS services
* If you wish to se the command line. Make sure you use access keys and do not share your passwords
* If want to review the permission generate the reports

**IAM Shared Responsibility Model**

* AWS is responsible for: Infrastructure (Global network security), Configuration of analysis, and the validation of their servers
* The IAM user of AWS is responsible for:
* Creating, managing, and monitoring: users, groups, policies.
* Also, the user is responsible for enabling MFA, rotating keys (access keys), providing the appropriate permissions, analyzing access patterns, and reviewing permissions