**AWS has Global Services:**

* IAM
* Route53 (DNS Service)
* CloudFront (Content Delivery Network)
* WAF (Web Application Firewall)

**Most AWS services are Region-scoped:**

* Amazon EC2 (Infrastructure as a Service)
* Elastic Beanstalk (Platform as a Service)
* Lambda (Function as a Service)
* Rekognition (Software as a Service)

**IAM Introduction: Users, Groups, Policies**

* IAM – Identity and Access Management, Global service.
* Root account created by default, shouldn’t be used or shared.
* Users are people within your organization, and can be grouped.
* Groups only contain users, not other groups.
* Users don’t have to belong to a group, and user can belong to multiple groups.

**IAM: Permissions**

* Users or Groups can be assigned JSON documents called policies.
* These policies define the permissions of the users.
* In AWS you apply the least privilege principle: don’t give more permissions than a user needs.

Login to IAM Service 🡪 Create users and group and attach policy administrator access.

**IAM Policies Structure**

Consists of

* **Version:** policy language version.
* **Id:** an identifier for the policy (optional)
* **Statement:** one or more individual statements (required)

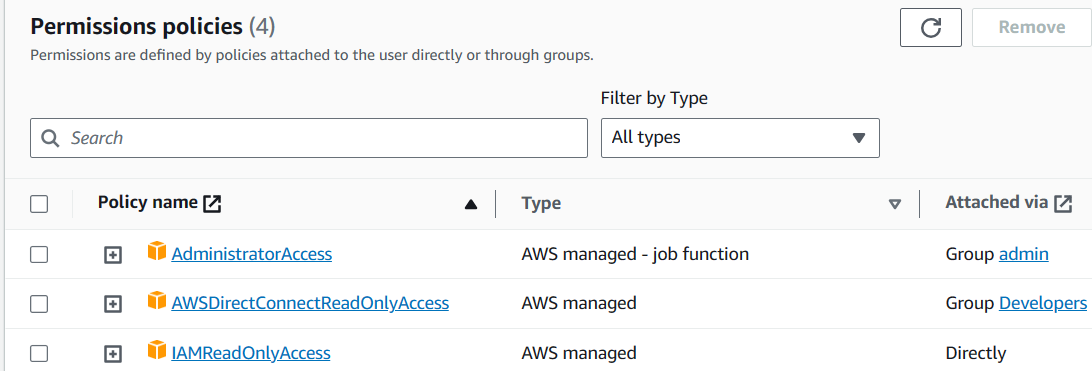
Statements consists of

* **Sid:** an identifier for the statement (optional)
* **Effect:** whether the statement allows or denies access (Allow, Deny)
* **Principal:** account/ user/ role to which this policy applied to
* **Action:** list of actions this policy allows or denies.
* **Resource:** list of resources to which the actions applied to
* **Condition:** conditions for when this policy is in effect (optional)

Admin Group 🡪 existing user 🡪 attach IAMReadOnly access policy.

Create group developers and add the existing user and add policy AWSDirectConnectReadOnlyAccess

Go to users 🡪 the user have 3 policies one from admin one from developers and one for the user.



386 services are there up to date.

**IAM – Password Policy**

Strong passwords = higher security for your account

In AWS, you can setup a password policy:

Set a minimum password length

Require specific character types:

* Including uppercase letters
* Lowercase letters
* Numbers
* Non-alphanumeric characters

Allow all IAM users to change their own passwords

Require users to change their password after some time (password expiration)

Prevent password re-use

**Multi Factor Authentication – MFA**

Users have access to your account and can possibly change configurations or delete resources in your AWS account

You want to protect your Root Accounts and IAM users

MFA = password you know + security device you own

User = password + MFA => Successful login

**Main benefit of MFA:**

If a password is stolen or hacked, the account is not compromised.

MFA devices options in AWS

Virtual MFA device

Google Authenticator (phone only)

Authy (multi-device)

Support for multiple tokens on a single device.

Universal 2nd Factor (U2F) Security Key – YubiKey by Yubico (3rd party)

Support for multiple root and IAM users using a single security key

**AWS Access Keys, CLI and SDK**

* To access AWS, you have three options:

AWS Management Console (protected by password + MFA)

AWS Command Line Interface (CLI): Protected by access keys

AWS Software Developer Kit (SDK) – for code: protected by access keys

* Access keys are generated through the AWS Console
* Users manage their own access keys
* Access keys are secret, just like a password. Don’t share them
* Access key ID ~= username
* Secret Access Key ~= password

**What’s the AWS CLI?**

A tool that enables you to interact with AWS services using commands in your command-line shell

Direct access to the public APIs of AWS services

You can develop scripts to manage your resources

It’s open-source <https://github.com/aws/aws-cli>

Alternative to using AWS Management Console

**What’s the AWS SDK?**

AWS Software Development Kit (AWS SDK)

Language-specific APIs (set of libraries)

Enables you to access and manage AWS services programmatically

Embedded within your application

Supports SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)

Mobile SDKs (Android,iOS,..)

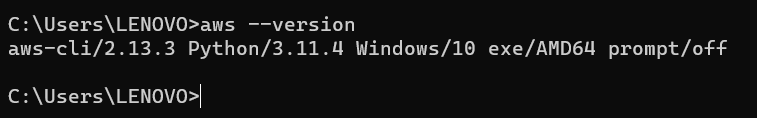
IoT Device SDKs (Embedded C, Arduino,…)

Example: AWS CLI is built on AWS SDK for Python

**AWS CLI Setup on Windows**

Browse aws cli install windows on google

Install the aws-cli version 2 on windows which is the latest version.



Aws cli on linux install

$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"

$ unzip awscli-bundle.zip

$ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws

$ aws –version

**AWS CLI Hands On**

IAM > Users > username > create access key

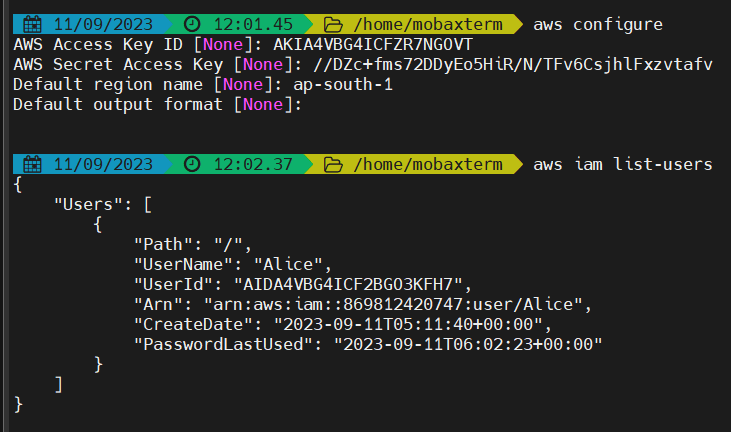
Check cli and I understand box 🡪 next 🡪 create access key.

$ aws configure

Enter your access key id and secret access key

Enter Default region name and default output format

$ aws iam list-users



**IAM Roles for Services**

* Some AWS service will need to perform actions on your behalf
* To do so, we will assign permissions to AWS services with IAM Roles.

IAM Role attached to EC2 Instance (virtual server) will have access to do some action from aws.

* Common roles:
* EC2 Instance Roles
* Lambda Function Roles
* Roles for CloudFormation

IAM 🡪 Roles 🡪 AWS Service from trusted entity 🡪 EC2 – as we are going to create iam role for ec2 instances 🡪 Next 🡪 Assign policies – iamreadonlyaccess 🡪 next rolename: DemoRoleforEC2 🡪 create role

**IAM Security Tools**

* **IAM Credentials Report (account-level)**
* A report that lists all your account’s users and the status of their various credentials
* **IAM Access Advisor (user-level)**
* Access advisor shows the service permissions granted to a user and when those services were last accessed.
* You can use this information to revise your policies.

**IAM Guidelines & Best Practices**

* Don’t use the root account except for AWS account setup
* One physical user = One AWS user
* Assign users to groups and assign permissions to groups
* Create a strong password policy
* Use and enforce the use of Multi Factor Authentication (MFA)
* Create and use Roles for giving permissions to AWS services
* Use Access Keys for Programmatic Access (CLI/ SDK)
* Audit permissions of your account using IAM credentials Report & IAM Access Advisor
* Never share IAM users & Access keys

**Shared Responsibility Model for IAM**

|  |  |
| --- | --- |
| AWS | You |
| Infrastructure (global network security) | Users, Groups, Roles, Policies management and monitoring |
| Configuration and vulnerability analysis | Enable MFA on all accounts |
| Compliance validation | Use IAM tools to apply appropriate permissions |
|  | Analyze access patterns & review permissions |

**IAM Section – Summary**

**Users:** mapped to a physical user, has a password for AWS Console

**Groups:** contains users only

**Policies:** JSON document that outlines permissions for users or groups

**Roles:** for EC2 instances or AWS services

**Security:** MFA + Password Policy

**AWS CLI:** manage your AWS services using the command-line

**AWS SDK:** manage your AWS services using a programming language

**Access Keys:** access AWS using the CLI or SDK

**Audit:** IAM Credential Reports & IAM Access Advisor

IAM User Groups can not be part of other User Groups. It can contain only IAM users.