**IAM (IDENTITY AND ACCESSS MANAGEMENT)**

IAM stands for identity and access management.It is a global service because in IAM,we are going to create our users and assign them to group.

**When we create an account for us we create a root account.Root account should not be used.**

We create user and and each person in a company is an IAM user.

**ROOT ACCOUNT**

A root account is the root user of our accounts.And the only things you should use it for is to set up your account But then you shouldn't use that account anymore, or even share it. What you should be doing instead, is create users.

**IAM USERS AND GROUPS**

We create IAM users from the root account and one user represents one person within your organization.users can added to groups.groups can only contain users, not other groups.

**Example:** we have an organization with six people.

You have Alice, Bob, Charles, David, Edward and Fred so all these people are in your organization. Now Alice, Bob, and Charles they work together.They're all developers.So we're going to create a group called the group developers .

And it turns out that David and Edward also work together. So we're going to create an operations group. Now we have two groups within IAM.

Some users don't have to belong to a group.For example, Fred right here is alone, he does not correspond to any group.That is not best practice. But it is something you can do in AWS.



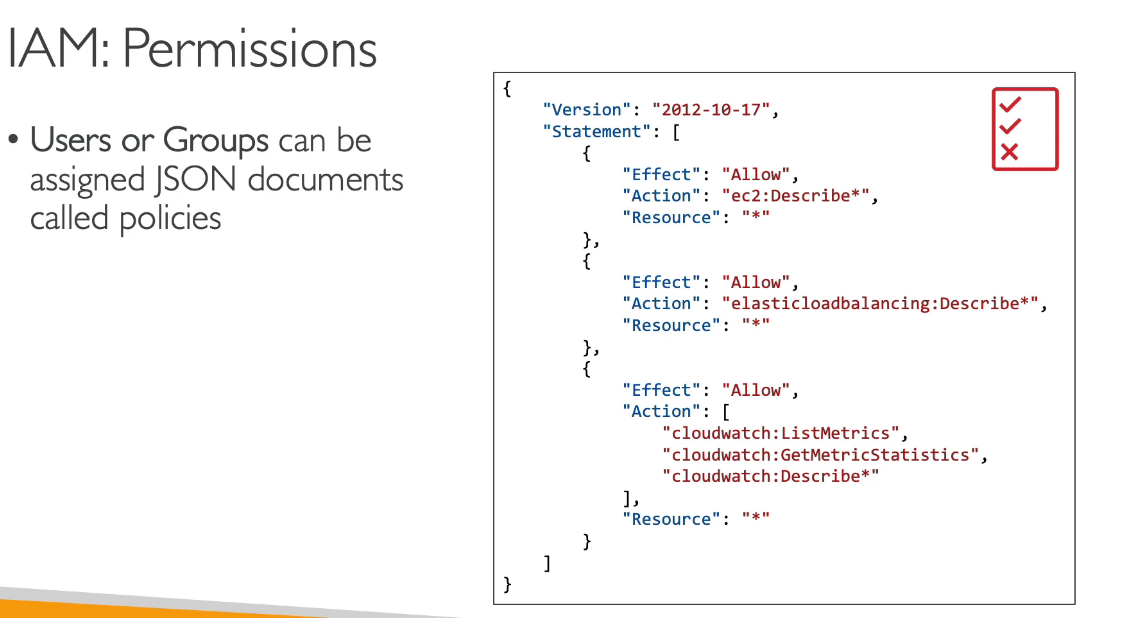
A user can belong to multiple groups. That means that for example, if you know that Charles and David worked together, and they're part of your audit team, you can create a third group with Charles and David. Charles and David are part of two different groups.So this is the possible configurations for IAM.

**why do we create users and why do we create groups?**

Well, because we want to allow them to use our AWS accounts and to allow them to do so, we have to give them permissions. So users or groups can be assigned what's called a JSON document called as Policy,an IAM policy.

what a user is allowed to do or what a group.

A policy simply describes what all users in the group are allowed to do.

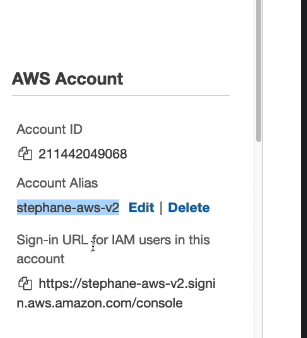
**Why we don’t allow everyone do everything?**

And so in AWS, you don't allow everyone to do everything that would be catastrophic, because a new user could basically launch so many services and they will cost you a lot of money or would be valid for security.

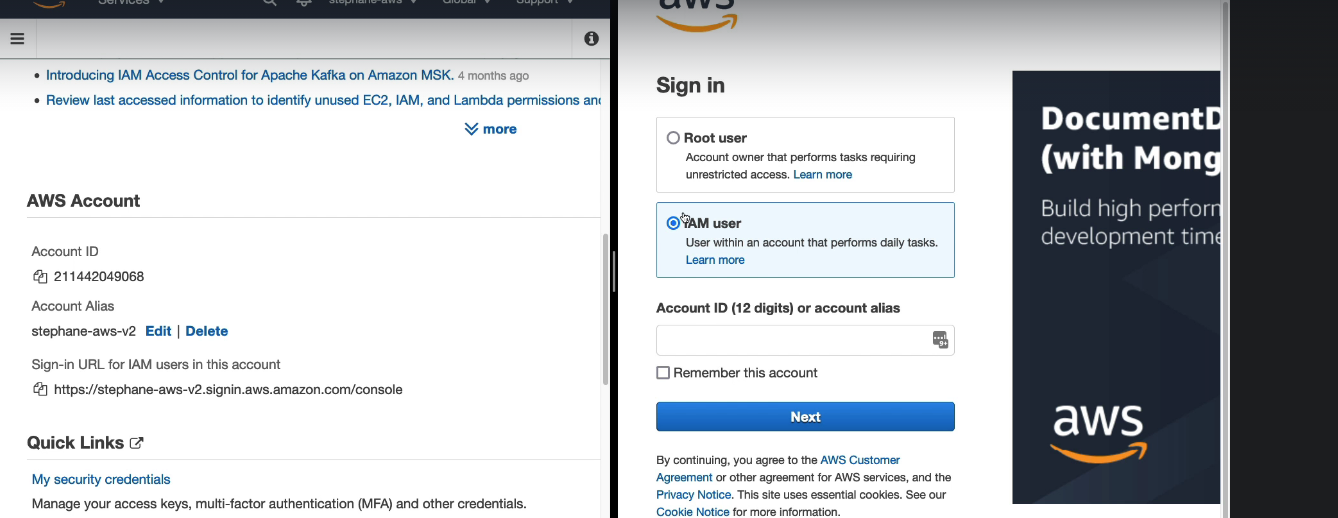
So in AWS, you apply a principle called the **least privilege principle.** So you don't give more permissions than a user needs.Okay, so if a user just needs access to three services,just create a permission for that user.

**Account alias**

Generally we have an account which is our rrot account and by default a number is associated to it but it is difficult to remember that number so we make an account alias which marks an easy to remember name for our account.This account id is used in case to generate arn of that account.



We require this alias in order to login to our iam account



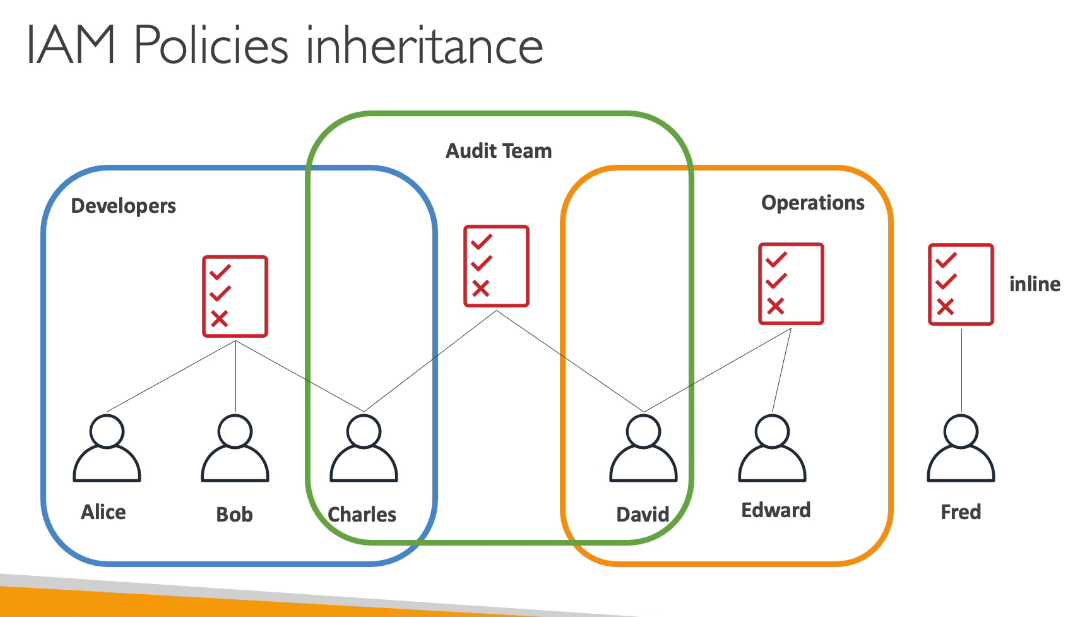
**IAM POLICIES**

Now let's discuss, IAM policies in depth.So let's imagine we have a group of developers,Alice, Bob and Charles, and we,attach a policy at the group level.In that case, the policy will get appliedto every single member of the groupso both Alice, Bob, and Charlesthey will all get access and inherit this policy.

Now, if you have a second group with operationswith a different policy,David and Edward will have a different policythan the group of developers.

If Fred is a user,it has the possibility not to belong to a group.And we have the possibility to create what's called an **inline policy** which has a policy that's only attached to a user.So that user could or could not belong to a group you can have inline policies for whatever user you want.

If Charles and David both belong to the audit team and you attach a policy to the audit team as well, Charles and David will also inherit that policy from the audit team. So in this case, Charles has a policy from developers and a policy from audit team. And David has a policy from audit team and a policy from the operations team.



**the policy structure**

you just need to know at a high level how it works, as well as how it is named. An IAM policy structure consists of :

1.**version number**: so usually it's 2012-10-17, this is the policy language version.

2.**ID:** which is how to identify that policy,this is optional.

3.**statements:S**tatements can be one or multiple ones, a statement has some very important parts.

**3.1 Sid** is a statement ID, which is an identifierfor the statement, which is optional as well

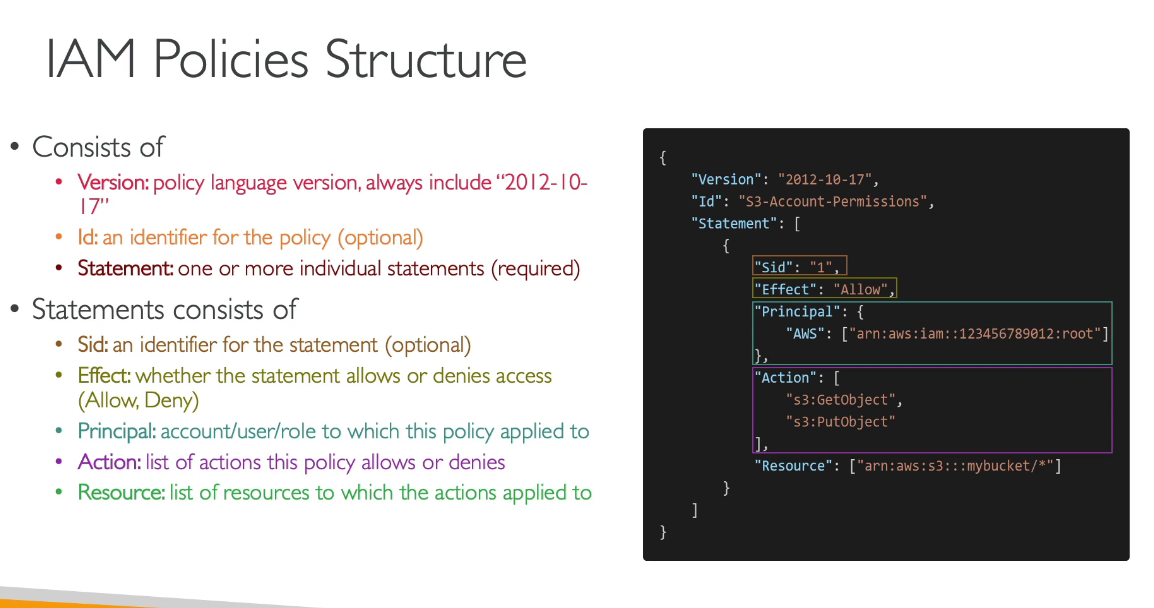
3.2 **effect of the policy** : it is whether or not the statement allows or denies access to certain API.It is either allow or deny

3.3**The principle**: consists of which accounts, user or role to which this policy will be applied to.

3.4**Action** :It is the list of API calls that will be either denied or allowed based on the effect.

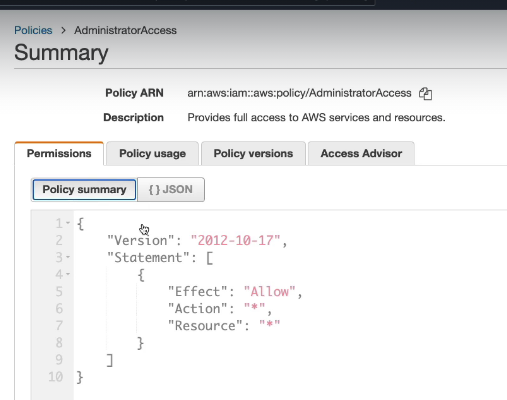
3.5 **the resource**: is a list of resources, to which the actions will be applied to.

4.**condition**: to which when this statement should be applied or not, and this is not representative here because it is optional.



For **example:**

We have an administrator access policy which allws all actions on all the resources as can be seen below:



**MFA(MULTI FACTOR AUTHENTICATION)**

Now that we have created users and groups, it is time for us to protect these users and groups from being compromised.So for this we can have two defense mechanisms.

**1.Password Policy**

The stronger the password you use the more security for your accounts.

So in AWS, you can set up a password policy with different options.

* set a minimum password length,
* require specific character types,
* you may want to have an uppercase letter,lowercase letter, number, non-alphanumeric characters,
* Then you can allow or not, IAM users to change their own passwords or you can require users to change their password, after some time, to make your password expired, for example,to say every 90 days, users have to change their passwords.
* Finally, you can also prevent password reuse so that users when they change their passwords, don't change it to the one they already have or change it to the one they had before.

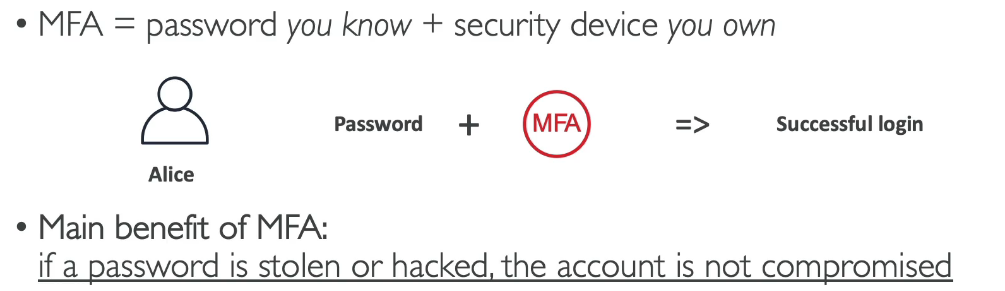
2.**Multi Factor Authentication or MFA.**

It is possible you already to use it, on some websites, but on AWS it's a must and it's very recommended to use it. So, users have access to your account, and they can possibly do a lot of things, especially if they're, administrators, they can change configuration, delete resources and other things. So you absolutely want to protect at least your Root Accounts and hopefully all your IAM users. So how do you protect them on top of the password? Them you use an MFA device.

**So what is MFA?**

MFA is using the combination of a password that you know,and a security device that you own, and these two things together, have a much greater security than just a password.

So for **example**, let us take Alice. Alice knows her password, but she also has an MFA generating token, and by using these things together while logging in, she is going to be able to do a successful login on MFA. So the benefit of MFA is that even if Alice has lost her password, because it's stolen or it's hacked, the account will not be compromised because the hacker, will need to also get a hold of the physical device of Alice that could be a phone for example to do a login. Obviously, that is much less likely.



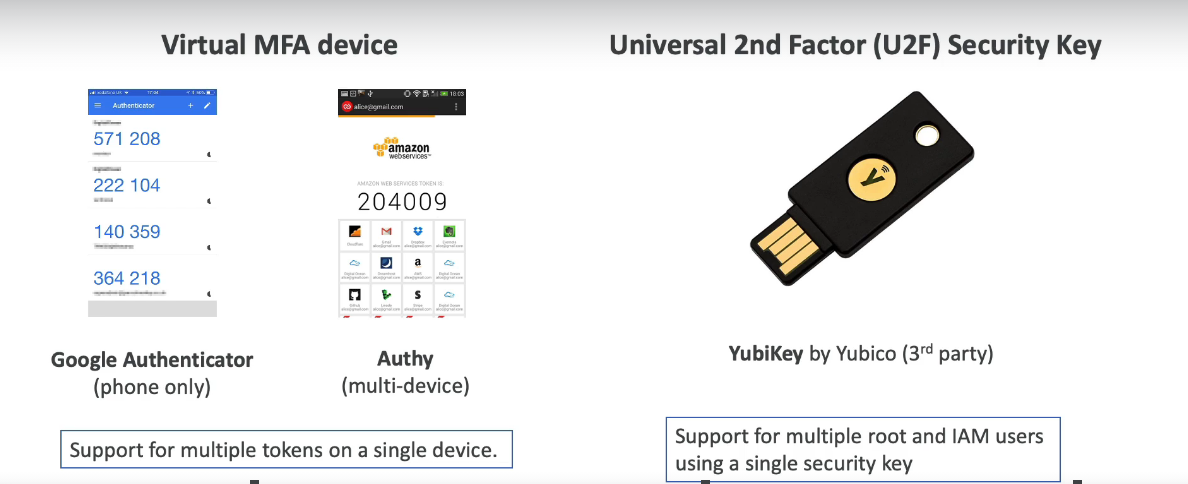
**MFA devices option in AWS**

**1.Virtual MFA device:** this is what we'll be using in the hands on

**Google Authenticator,** which is just working on one phone at a time,

**Authy** which is multi-device they both work the same except one is multi-device.for Authy you have support for multiple tokens on a single device.

So, that means that with a Virtual MFA device, you can have your root account, your IAM user, and another account, and another IAM user, its up to you, you can have as many users and accounts as you want on your Virtual MFA device, which make it a very easy solution to use.



**2.Universal 2nd Factor or U2F Security Key,**

that is a physical device, for example, a YubiKey by Yubico and Yubico is a 3rd party to AWS,

this is most the AWS that provided, this is a 3rd party and we use a physical device, because maybe it's super easy, you put it your Key Fobs and you're good to go. So this YubiKey supports multiple root and IAM users using a single security so you don't need as many keys

as users otherwise that will be a nightmare.

Then your other options,

you have **a Hardware Key Fob MFA device** for example this one provided by Gemalto which is also a third party to AWS and finally, if you are using the cloud of the government in the US, the AWS GovCloud then you have a special Key Fob, that looks like this, that is provided by SurePassID which is also a 3rd party.

Graphical user interface, text, application, email

Description automatically generated

**Note:**if you do mfa and if you loose your mfa token then you will belocked out of your aws account by the aws.

**AWS CLI, ACCESS KEYS AND SDK**

So we have seen how to access AWS using the Management console, which is the Web interface. there are, actually, three different options to access AWS.

1.**Management console: It** is protected by your username, password, maybe multifactor authentication.

2. **CLI, Command Line Interface: T**his is something we will set up on our computer,

and this is protected by access keys, and access keys our credentials we're going to download that will allow us to access AWS from our terminal.

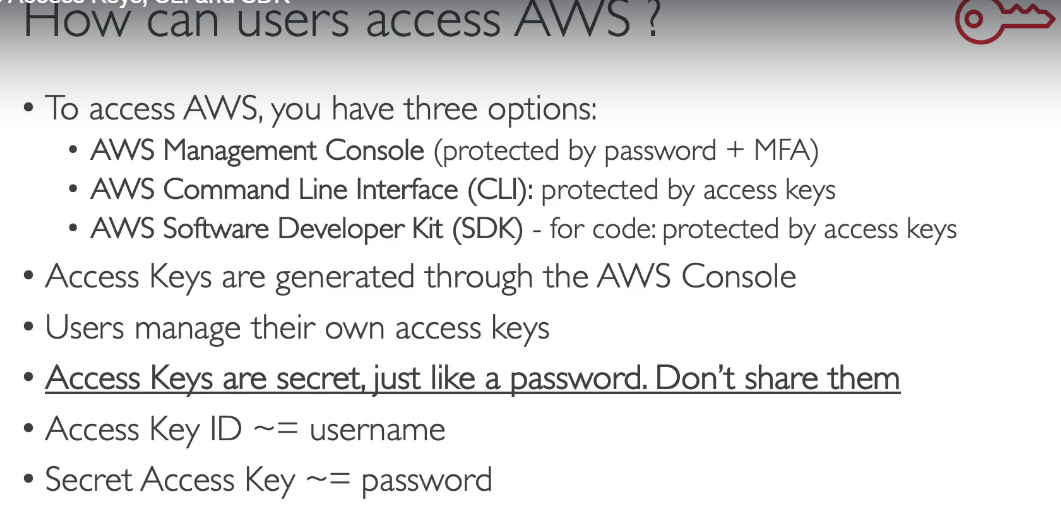
3.**SDK:**the AWS Software Development Kit, which is used whenever you want to call APIs from AWS from within your application code. these will be protected by the exact same access keys.

**How do we generate access keys?**

we will do this through the Management console,and users are responsible for their own access keys,and access keys, from the user perspective,are secret, just like a password,so if you generate your own access keysdo not share them with your colleagues,because they can generate their own access keys as well.So really make sure that you treat:

* your access key IDjust like your username,and
* your secret access key just like your password,you do not share them with other people.

Access keys are generated from the management console and then using those accesss key id an dkey we can access to our aws console from the CLI.

****

**CREATING AND USING AN ACCESS KEY**

We can create an access key and an access key id and we can use it to login to our account from the cli .

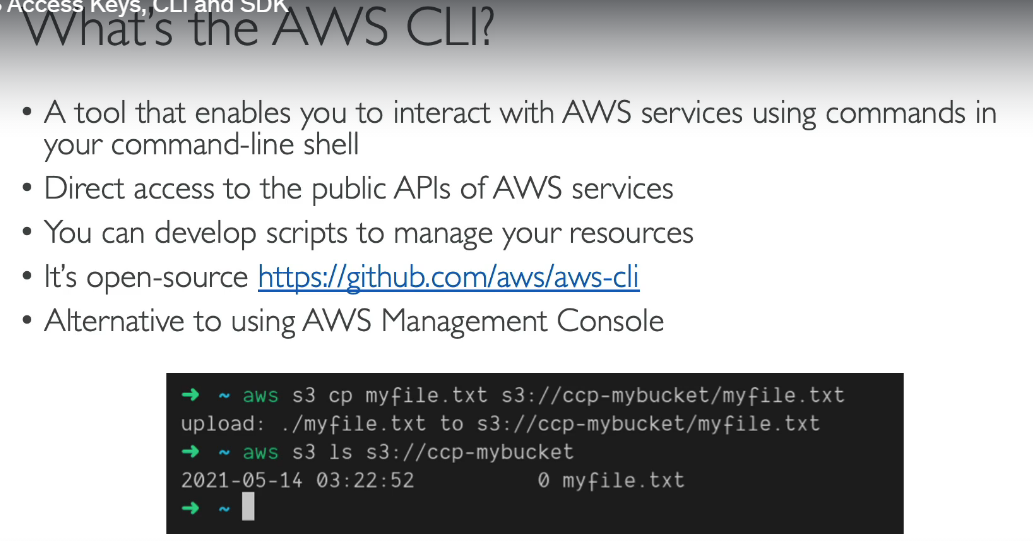
**CLI(COMMAND LINE INTERFACE)**

CLI stands for Command Line Interface, and the AWS CLI is a tool that allows you to interact

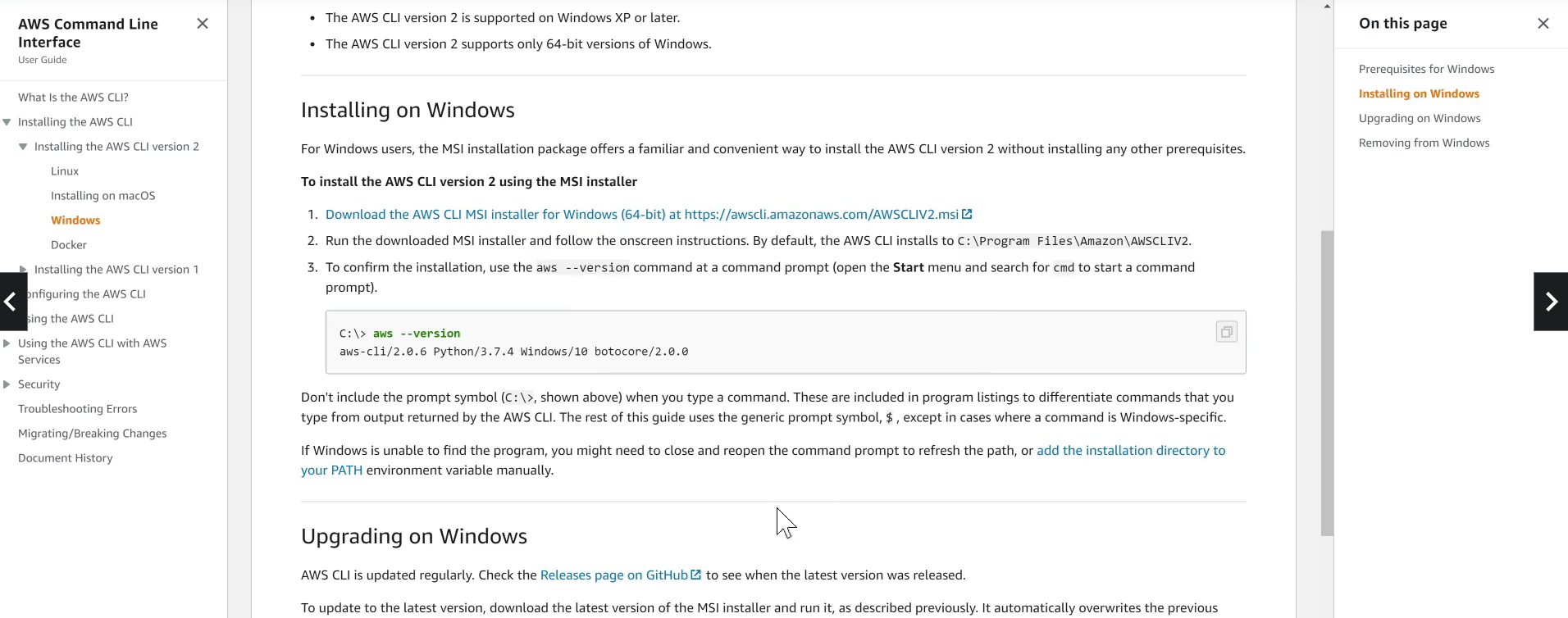
with the AWS services using commands from your command-line shell.

Now with this CLI, you get direct access to the public APIs of your AWS services And, then, using the CLI you can develop scripts to manage your resources and automate some of your tasks.

The CLI is open-source, you can find all the source code on GitHub, and it is an alternative to using the AWS Management console.



**Install aws cli**

****

**Creating security credentials to login to the console from cli**

Note:Always create the security credentials from the iam user account and do not use your root account to create those credentials.

Each user can create access keyed for himself and then we can use that access key id pair to login to our console from the aws cli as you can see below:

Graphical user interface, text, application

Description automatically generated

**what's the SDK now?**

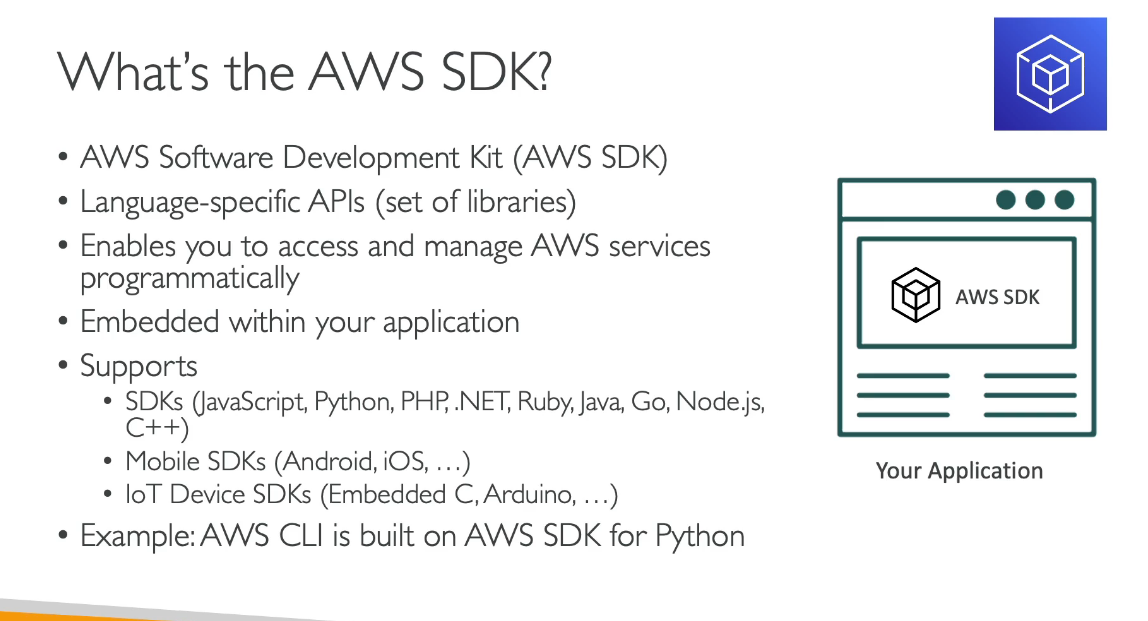
SDK stands for Software Development Kit, and this is a set of library, this is going to be language specific, so you're going to have an SDK for different programming languages, and similarly, it will allow you to access and manage your AWS services and APIs programmatically, but this time the SDK is not something that you use within your terminal, it is something that you embed within your application that you have to code. So your application will have the AWS SDK from within them.

It supports many different programming languages, such as JavaScript, Python, PHP.NET,

Ruby, Java, Go, Node.js, C++,

There's also the mobile SDK, if you're using Android or iOS, and the IoT, so Internet of Things device SDK in case you're using some thermal sensors or backlogs that are connected, all these kinds of things.

So to give you an example of what you can build with the SDK, well the AWS CLI that we're going to be using in this course is actually built on the AWS SDK for Python named Boto.



**AWS CLOUDSHELL**

Aws cloudshell is an alternative to the cli.It is not present in some regions.

* When we login to the cloudshell it by default take the credentials of the account that we are logged in with
* In cloudshell the region is the same we have logged in with.

**IAM ROLES FOR SERVICES**

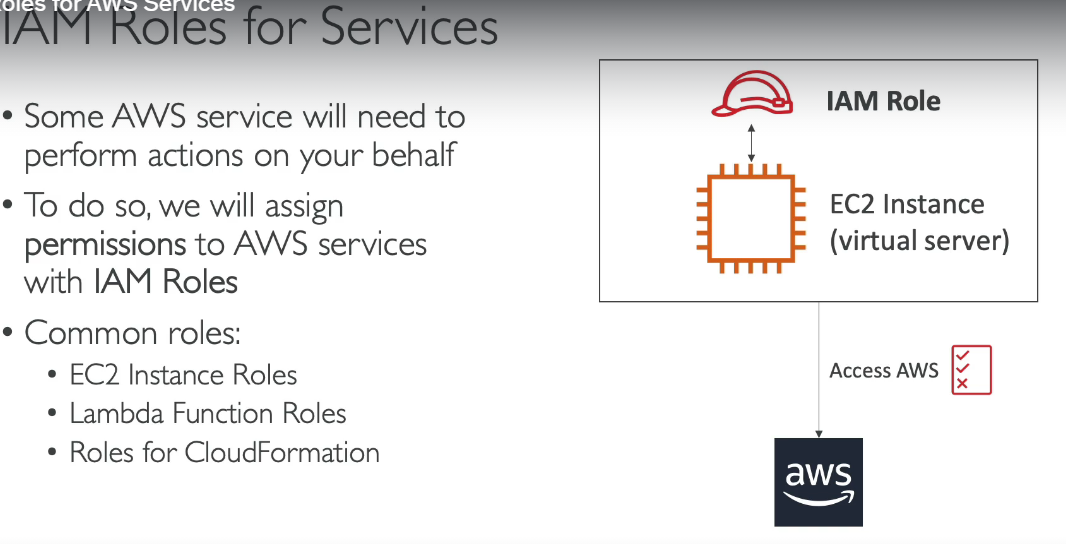
IAM Role is an AWS componenet that will act as a user who need some kind of permissions. iself.Suppose we want some services to perform actions on our behalf.

So these IAM role will be just like a user, but they are intended to be used not by physical people,

but instead they will be used by AWS services.

**example,**

we are going to create throughout this course, an EC2 Instance. An EC2 Instance is just like a virtual server, and we'll see this in the next section. But so this EC2 Instance may want to perform some actions on AWS and to do so, we need to give permissions to our EC2 Instance. To do so, we're going to create an IAM Role and together they're going to make one entity. And together, once the EC2 Instance is trying to access some information from AWS, then it will use the IAM Role. And if the permission assigned to the IAM Role is correct, then we're going to get access to the call we're trying to make. So some common roles include what I just showed you, EC2 Instance roles, but also other things that perform actions against AWS

****

For example if we make a role of doing a readonly from the iam and assign it to an ec2 instance then through that ec2 instance we can list all the users we have into our account.

**IAM SECURITY TOOLS**

Graphical user interface, text, application, email

Description automatically generated

**A** credentials report will give us an excel file consisting of the number of accounts we have associated with our account and also gives report about the password last changed and when was it changed.SO this will be a helpful file in order to keep us updated about the various users in our account.

Using the access advisor you can keep a check on the services that are being used by aparticular user and if there is any service which is of no use to that user then we can invoke access to that service because that user does never use it.

**IAM GUIDELINES AND BEST PRACTICES**

Text

Description automatically generated

Aws security assist you in safeguarding sensitive data and information while also meeting compliance and confidentiality standards.

Aws enables you to automate tedious security.