**IAM Essentials**  
  
**Identity and access management**, also known as IAM, is an essential part of AWS because it controls access to all of the AWS API endpoints.

Now as a reminder, an API or an application programming interface is the way in which applications talk to each other. It's also the way that the console UI, the AWS command line tools, and any other applications designed to work with AWS talk to AWS. Whenever you interact with the AWS console in any way, what it's doing is behind the scenes, it's using those APIs to interact with the AWS infrastructure.

So **IAM is the thing that authenticates you to AWS**. It authorizes you, and so together that means it decides if a request that you make is allowed or denied. If you attempt to store an object in an S3 bucket IAM allows or denies that request. Any interaction with AWS is decided on by IAM.

I mentioned earlier in the course that when you create an AWS account, you automatically create an account root user. Initially, the account root user is the only identity in the account, and it's the one which starts with full control of the account, and it also always keeps this full and unrestricted access. There's no real way in an isolated AWS account to restrict what the root user can and can't do. Now there is one small exception, and this involves AWS organizations. **The account root user is responsible for the initial setup of IAM inside an AWS account.**

So, if I move across to the IAM console, I'm logged into this account using the account root user, I want to draw your attention to this IAM user's sign in link. This is the link that any IAM users for this account will use to sign into this account. Now when you first create an account, this sign in link is a standard format. It starts with the account number of the account and then it finishes with .signin.aws.amazon.com/console. You're able to customize this and apply an alias and I've done that. I've added the alias la-saac01-master. so you're able to customize it to make it a little bit easier to identify and a little bit easier to use by your IAM users.

**The account root user is also initially responsible for account wide settings and generally the major component of this is the password policy.** The password policy governs minimum password length and other limitations and requirements of the passwords. So you can set that it's got at least one uppercase and one lowercase character, maybe one number and one nonalphanumeric character and then you're also able to set various options, such as allowing users to change their own password, enabling password expiration, and preventing any reuse of historically used passwords.

There are other options here if I scroll down, such as options relating to the security token service and endpoint options, but we'll be covering those later in the course.

**IAM allows us as architects to design AWS deployments with more granular permissions because unlike the account root user, which always has full and unrestricted access to AWS resources in the account. With IAM, we can assign more granular permissions. That's the benefit that IAM provides.** It's the only way that we can implement a best practice security infrastructure within AWS.

Base level architectural components that IAM provides: We're going to start with **IAM users**. IAM lets us create a few types of identity or things which help us manage those identities. We've got IAM users, IAM groups, IAM roles, and IAM policies.

**IAM users** are generally used to represent actual users. So human beings that log in to an AWS account or access services. IAM users don't have to be users though. They could be service accounts. So maybe accounts that are used by a piece of auditing software or maybe backup software. When you think about IAM users, you should be able to picture the thing or things which use that user—if you can imagine a single human logging into an IAM user or a group of humans or a service account used by one or more applications then you can be fairly confident that an IAM user is the right identity to use.

If you can't picture who or what uses that identity, then you might want to look at another identity type in IAM which is an **IAM role**, and we'll talk about that later in this topic. For now and for the exam IAM users are the type of identity that you log into. They have usernames and passwords and you log in to the console UI using IAM users and later in this section of the course we'll be using IAM users to secure your own personal AWS account that you created at the start of this course.   
  
**ARN or Amazon resource name**. If I opened up this user that I've already configured inside this AWS account, you'll note that it's got a user ARN. **Now an ARN is a standard notation inside AWS**. It's a thing that can uniquely identify any resource in any region in any account.

* ARNs always **begin with:**

ARN:partition:service:region:account-id:  
  
partition = AWS or AWS-CN (for China)  
service = the AWS service: S3, EC2, RDS, and DynamoDB etc.

region = region code: U.S. East 1, A.P. Southeast 2, or any of the other AWS regions

account ID and then after that it follows up with any of this structure of naming. So either the resource, the resource type, and then the resource or any of these other options.

* And depending on service **finish with**:

resource

resourcetype/resource

resourcetype/resource/qualifier

resourcetype:resource

resourcetype:resource:qualifier

Example ARNs:  
  
1. arn : aws : iam : : 123456789012 : user/roffle

2. arn : aws : s3 : : myamazingcatpics/truffs.jpeg

3. arn: aws : dynamodb : us-east-1 : 123456789012 : table/ratemycats

In some cases wild cards are supported:   
  
arn : aws : ec2 : us-east-1 : 123456789012 : instance/\*

Fields with :: omit the value and \* is a wildcard match

The idea is that for any given ARN, you can uniquely identify a resource.

Now, some examples that I have included :  
1 .in this case we've got an IAM user in account 123456789012 and it's an IAM user called Ruffle, and we can see that this is a unique identifier. So if you've got this ARN, you can point to the exact service, the exact account, and the exact type of resource that this is. So this uniquely identifies this Ruffle user globally across every AWS account. Note how **IAM doesn't need the region because this is a global service. So all IAM users are valid inside all regions in a given account**.

2. We've also got another example, which is an object inside an S3 bucket called My Amazing Cat Pics and the object is truffs.jpg. Note that with this example, **in addition to not needing the region, we also don't need the account number and that's because S3 is globally unique across all accounts**

3. in the last example, we've got a DynamoDB service in the U.S. East one region in this account, and it's a table called Rate My Cats.

One last point wild cards are supported and so with this example, you can use an ARN which can represent all EC2 instances or AWS's virtual machines in a region in an account. So by using the star, we're matching all EC2 instances in this account in this region.

Now another feature that **IAM provides is groups and groups allow you to group users together**. If you have a specific team that you want to share a similar set of permissions, maybe a dev team, a finance team, an admin, or support team. A user can be placed in multiple groups, and a group can have multiple users. **Groups don't have any authentication information of their own, so they don't have passwords, but they can be assigned IAM policies.**   
An **IAM policy is the thing that defines permissions, and they could be assigned to users and to groups**. If they're assigned to groups, they affect every user in that group.   
IAM lets you create and manage what are known as **IAM policies. They're essentially JSON documents, and they define under which circumstances a principal in identity is authorized or not to access one or more resources or features of resources in AWS**. So, policies define what can and what cannot be done. They don't do anything by themselves. So policies just define these things. They only take effect when they're attached to users, groups, or roles and when that happens, they do something.   
We've got IAM roles and roles are really interesting identity inside AWS. **Roles can actually have permission so you can attach policies to roles as well as users and groups, but roles can't be logged into they're not used on a permanent basis**. (roles : side jobs inside your job)  
Roles inside AWS serve a number of really important functions. Roles could be used by humans, by applications, or even by AWS services doing things on your behalf.

The features that IAM provides are **credentials**. Credentials are what principles use to prove that they are an identity inside IAM. So imagine that you're logging in to AWS. When you're logging into AWS you have to provide your username and password to prove that you are you and that's an example of IAM credentials. You're supplying your username and password to IAM in order to prove that you are you.

**IAM offers three main types of credentials**. You've got **the username and passwords**, which are used by IAM users they're long term and they're used to access the console UI. We also have **access keys**, which are long term credentials, kind of like username and passwords, but they're used by the command line. If you call AWS's APIs directly, then you're going to be using these access keys and then we've got a different type of access keys, which are known as **short term credentials** and these are used by roles. They're kind of like access keys and kind of like usernames and passwords, but they're time limited. The role gives some additional authority or access rights to an identity for a certain duration and so, when the identity assumes a role that identity gets temporary security credentials that give the identity those additional access rights for a certain duration.

**By default, an IAM identity has no permissions. It has none.** If you create any form of identity inside AWS until you give it some permissions, it has none. This is the default in AWS. You can't access anything you have an implicit deny until you are given the permissions to do something. It's known as an **implicit deny**, I want you to remember that term, because if nothing else applies, if an individual identity, a user, a group, or a role is not given the permissions to do something, they are denied the ability to do that thing. That's one of the most important things that you need to remember for the exam.