**IAM users** one of the identity types provided by IAM.

IAM users are very much like other accounts that you may use. You got a Linux Academy account, which you've logged into and are using to watch this video. You've probably got an email account. Maybe you've got something like a Netflix account. IAM users adjust accounts like this. For the exam, there is going to be an expectation that you know not only how to use an IAM user, but when and where they make sense to use versus groups and versus roles. Keep in mind, IAM users generally have a 1:1 ratio with a human being or an application. **Something actually logs in to an IAM user**. If you see any exam questions, where there is login mentioned then it has to be an IAM user. Since groups and roles can't log in or be logged in to. It's important to understand this architecture because there are equally many situations where an IAM role, which I'll talk about later in this topic is much more suitable than IAM user. What I need to do in this topic is to help you understand the difference.

So let's step through the flow of an IAM user. Let's say that you want to log in to an IAM user may be from the console or maybe using the command line. In this scenario, you're what's known as a security principal, and you're wanting to prove to IAM that you are an IAM user an identity and to do that you authenticate. Now authentication with IAM is done either using username and password if you're human being looking to log in to the AWS console and optionally, you can also use MFA or multi factor authentication, which you'll have used if you've set up your AWS account as per the lesson earlier in the course, MFA just provides another factor. It's another thing that you have or another thing that you know but in either case, you use both of these things to authenticate that you are who you say you are. Now you can also use what are known as access keys. We haven't talked about those yet, but if you utilize in the command line or any APIs directly, then you don't use username and passwords. You use access keys, but in either of those cases, what you're essentially doing is the same process. You're authenticating to IAM. You're attempting to prove to IAM that you are an IAM identity. If you're successful in that process, then you become what's known as an **authenticated identity.** You authenticate to the IAM user that you're attempting to use. In the previous lesson I talked to you about IAM policies well, the identity that you are now authenticated as will probably have policies attached. So IAM policies attached to either the user directly or a group that the user is in, In the unlikely event that you, as an authenticated IAM user, don't have any applicable IAM policies then as we talked about in the last lesson, **there's this default implicit deny. So with no policies you have access to nothing.** If you do have any applicable IAM policies then they're used in the next step of the process, which is authorization, you are authorized to access a resource or not.

I think the best way to show this is to actually demonstrate the process and to do that, I'm going to move over to the AWS console I'm in the IAM console and I'm gong to click on users and create a user. So I'm going to click on Add User, I'm going to call this user test user. I need to select the box that says AWS Management Console Access, because I'm going to be utilizing this user to log in to the AWS console. Now I'll untick require password reset. I don't want to have to go through the hassle of resetting the password when I first log in. If this is a production situation you might want to leave this and to allow the user to select their own password. In this case, I'm going to go ahead and click on Custom Password and then enter a strong password of my own. Once I've done that, I'm going to go to next permissions, and at this point I won't be attaching any policies. I'm just going to skip all the way to the end of the process and create user. Once I've done that I'll click on close. I'll just select the test user just to explain a number of the important elements of an IAM user. Now, an IAM user is essentially a container an identity container inside AWS. Every IAM user has an associated ARN. **It's a real identity inside AWS and an IAM user can be referred to from elsewhere**. So when I say real identity, just remember that term for the exam. **Anything that has an ARN and can be referred to or referenced from elsewhere in AWS for example, in a resource policy, that is what I term a real identity.** Now AWS do use that term. It is in some documentation, but it's not all that common that you'll hear it but I have seen it on an exam question. So if I do mention a **real identity then I'm talking about users and roles**. **Groups are not real identities**. IAM users do have an ARN. You can refer to them. Now this is where you'll specify inline permissions. If you want to add an inline policy, you can do it for within the permissions section of the IAM user. The format is exactly the same. So if we did want to apply this same IAM policy as an inline policy. We could do so but I won't be doing so in this demo because I've already gone ahead and created the customer managed policy in the previous lesson. Now we're not going to cover **permissions boundaries** till later in the course but what they essentially do is allow you to **define boundaries beyond which a user's permissions can never go**. So I could, for example, set a boundary to only allow this IAM user access to S3. Even if this user was then given permissions to all the AWS services they wouldn't have access to them because they would breach the boundary.

Now **IAM users can be assigned to groups, and you can do that either from the group perspective**. So go to groups and then add users, or **you can do it from a user perspective** so I could click add users or groups, and then I could go ahead and add this user to a particular group but obviously I don't have any groups defined yet, so I won't do that. **Most AWS services are capable of adding tags, which are key value pairs,** and you can use these either for business reasons, maybe tagging which business unit somebody belongs or which cost center or you can use tags for other business specific reasons. So maybe which team they remember or which projects they're currently working or what their level of authorization is. Maybe some automated processes in the background use tags to automatically add or remove access.

Now security credentials are where most of the relevant settings are for an IAM user. It's in here where you can set the console password, so the username is always the same as the IAM user. The password could be changed after the fact, so you can change the password that's used to authenticate the console here. You can also assign an MFA. We haven't done this for the test user, but I did demonstrate this when we were creating your initial AWS account together in the lesson earlier in the course. So that's something to keep in mind. You can always assign an MFA device to an IAM user, the same as you can the account root user, and that makes it much more secure. Access keys are the alternative type of authentication that can be used for the CLI and the APIs we're not going to configure those in this lesson because we do have a lesson coming up specifically where I'm talking about access keys, but it's here where they're defined, and then if you're using any other AWS services such as AWS CodeCommit, then you've got the option of defining additional authentication elements inside the IAM user. We're not going to be covering those at this stage because we are going to be talking about these specifically later in the course, and I will be returning to them. For now, I just want you to focus on these core IAM user elements. So the access keys, which we'll talk about in a lesson that's coming up very soon, the MFA device, the password, and then permissions that could be assigned in line or via a managed policy. Now, that's what I'm going to do now.

I'm going to associate the customer managed policy that I created in the previous lesson. Whether you're adding an inline policy or using managed policies, you can add permissions from the same location. So I'm going to go ahead and click, Add Permissions, going to attach an existing policy directly, and I'm going to search for cats and dogs. There we go. I can see the policy of created in the previous lessons, so I'll select that go to next review and then add permissions and that will associate that managed policy with this IAM user. Now, at this point, I can test this process out. If I go back to the dashboard and I copy the IAM URL for this account. I'll copy that into my clipboard and I'll paste it in, and I can use that to log in to this account. So this will log me out of the current user. Check the account ID or Alias is correct, which it is and then I'm going to use test user, which is the user that I created in the previous step and then I'm going to use my strong password. Once I've done that, I'll hit Sign In. So now I'm logged into this account but I have a limited set of permissions on this IAM user and an example is, if I go back to the IAM console, remember, this time I'm logged in using test user. Then I'll get lots of permissions errors because, unlike the account root user, which has unrestricted permissions with this user, I have a very limited set. That limited set only allows me to access this set of actions on these set of resources so remember I'm able to list all the buckets in the account, list all the objects in the buckets, and then interact with the Dog pics buckets so I'll go ahead and do that. I'll go back to the console. I'll move to S3. Now because I have those permissions because I have the list all buckets action and that's allowed for all S3 resources I'm able to list all of the buckets in this account. If I go in to the cat pics bucket I'm able to list all of the objects in this bucket. If I select one of the objects in this bucket so penny. jpeg and click open, I'll get an access denied and why is that? If I go back to the policy in my code editor, this is the relevant statement. This is the statement that allows s3:wild card object. The wild card will match get and get object is the action that I have just attempted to perform. The reason why I can't do it is because it matches only objects in the dog pics bucket not the cat pics bucket. While I'm able to list all the buckets I'm able to list all of the objects in both of the buckets I'm only able to interact with the objects inside the dog pics bucket. Remember what I'm trying to do at the moment is my identity is trying to run get object on an object in cat pics because there's no matching statement. This statement doesn't apply. This statement doesn't apply, and neither does this one because none of them apply that implicit deny takes effect. So the implicit deny denies my request. If I go back to the console and this time I moved back and go in the dog pics bucket. In this case, I'll be able to interact with both of these objects if I click on sparky.jpeg and click Open this time I'll be able to open this particular image. So I'll get to see Sparky in all his relaxed majesty because this wild card action so s3:\*object that also matches actions which delete objects. So if I want to go back to the AWS console and maybe I don't want Mike in a picture with Sparky, so if I open this image on one hand, it's got Sparky in, so the image is amazing, but on the other hand, it's got Mike maybe I don't want that. So if I go back to S3 console, I can actually go ahead and delete this particular object. So select Sparky Mike, go to actions, go to delete, accept that by clicking delete, at this point, I can delete that object and the reason I can do that is this is the statement that is matching that delete operation. So I hope this is making sense that if you don't have an applicable statement then by default, there's an implicit deny. If there is no allow, you are denied, and that is why I can interact with anything inside the dog pics bucket but I can't interact with anything inside the cat pics bucket. I can list the objects because there's a different statement that allows my user to do that. I can list all of the objects inside either of the buckets, but I can only interact with objects inside the dog pics bucket because that's the only applicable statement.

I want to cover a number of key, architecturally related facts. This is important to understand, because for the exam a number of key facts relating to IAM users are really important to remember.

* **There is actually a hard limit of 5,000 IAM users per account**. Now this is important because if you face an exam question where you're asked maybe to merge two different companies and one of the companies has 6,000 users then you know you can't do that with AWS because you're not able to create more than 5,000 IAM users, and so you need to approach it in a different way, maybe using roles, and we'll talk about that later in this topic. Always remember the 5,000 IAM user limit. That's a per account limit. It's not per region. It's per account because IAM is a global service.
* **There's also a limit of 10 group memberships per IAM user.** The number of groups you can have in the account is much bigger, and that's adjustable on a per account basis but a single IAM user can only be in 10 groups
* There's a **default maximum of 10 managed policies per user**. Again, you can adjust that, but it's useful to be aware of.
* There's **no limit on the number of inline policies but you cannot exceed 2048 characters for all inline policies on an IAM user.** So this is fairly weird to understand at first but all of the inline policies that you have on IAM user the characters that they consist of—ao not the white space, just the characters—there's a maximum of 2,048 characters for all of the inline policies on an IAM user, so keep that in mind.
* **Every IAM user can have one MFA device attached.**
* **There is a limit of two access keys per user**.