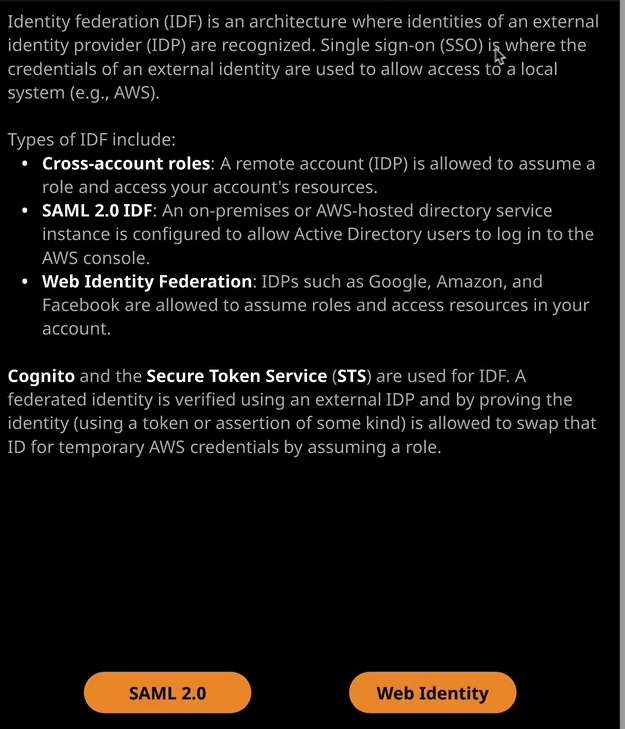
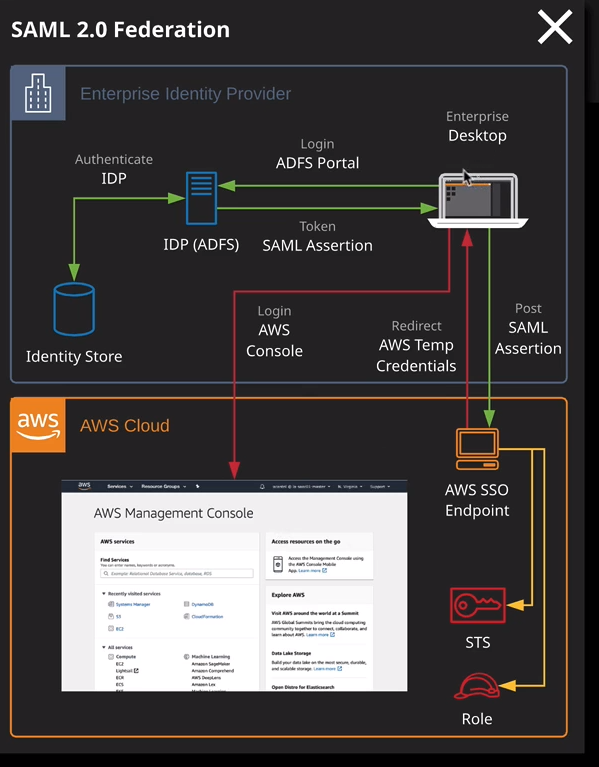
Welcome back and in this topic of the course, I'm going to be discussing ID Federation and single sign on. This topic is going to be two lessons and a hands-on lab. I'm not going to have much in the way of practical demonstration in either of these lessons, because you're going to get a chance to experience how ID Federation works in a practical setting in the hands-on lab at the end of this topic. So for now, I'm going to cover the architecture, the key features, and then talk about when and where you might use ID Federation. So let's get started.



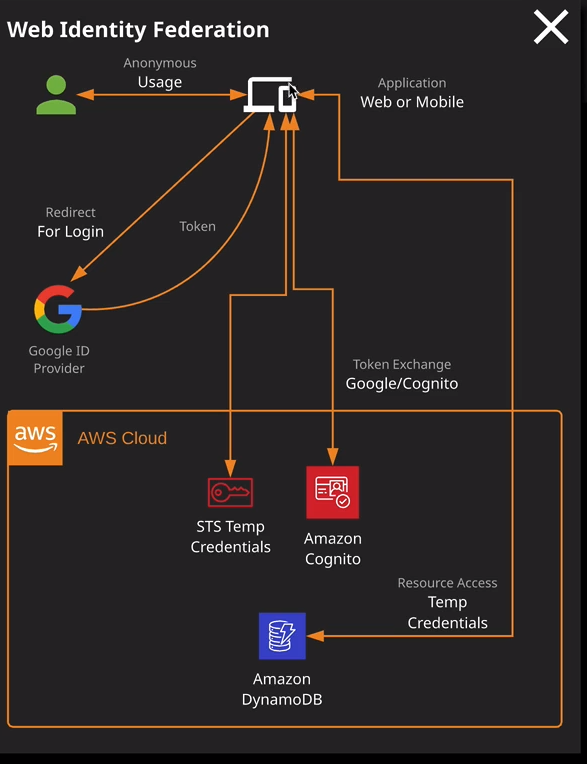
Now **Identity Federation, known as ID Federation, or IDF, is an architecture where identities of an external identity provider— so an IDP—is recognized single sign on is where you as a user can use the credentials of an external identity and use those to access local systems such as AWS**. So, an example of this if you've ever used a mobile application that asks you to sign in with your Facebook, your Twitter, or your Google credentials, that's an example of ID Federation and single sign on. You're taking an external identity so external to the application itself, and you're using it to log in to that application, and the process that the application has to go through is known as Identity Federation. So before we move on, I just want to touch upon what an identity provider is. The name gives it away. **It's essentially a provider of identities, so within your business, you might have an identity provider such as Microsoft Active Directory. When you log onto Google Mail, Google Mail is functioning as an identity provider. When you use Facebook or Twitter, they use identities. They have an identity internally for you, and you use that identity to log on to their platform** but it's possible to write and design an application to make use of external identities an external identity provider.

Now there are many different types of identity federation at its simplest cross account roles that I've talked about earlier in the course are an example of this. **So with a cross account role you will create a role in your AWS account. You will trust an external AWS account so a different set of identities and you'll trust them to be able to assume that role and then perform actions in your AWS account. You're essentially federating an identity.** You're allowing an external set of identities to get access to your account, to swap their credentials for something that's valid in your account. Now, with cross account roles we're using the same AWS provider of identities, it may be that a different account has their own IAM instance, and you have your own IAM instance but at the backend, it's AWS providing those identities. So that's why we don't always refer to it as Identity Federation. We only just say cross account roles, but it is the same process.

We also got **SAML 2.0 Identity Federation** and SAML 2.0 is a standard which is often used for certain on premises systems such as Microsoft Active Directory or another AWS hosted directory service. So SAML is a way that you can integrate different identity providers with applications and allow users or identities to be reused. We've also got **Web Identity Federation, which is where we use identity providers such as Google, Amazon, and Facebook and we allow them to assume roles inside our AWS accounts and access resources.** Now the way that SAML 2.0 or Web Identity Federation is supported in AWS is using the **Cognito and secure token service, or STS products. Now remember, STS is actually the product that allows you to get short term temporary credentials.** When you assume a role inside your account or another AWS account, it's a secure token service, or STS, that's responsible for generating the temporary or short term security credentials that you will use to access that account. **Cognito is a service that can essentially broker this single sign on or ID Federation, and it allows you to create what are known as ID pools**. These could be discreet pools of identities, but also it can support the architecture where you might have a Twitter log in and a Facebook login and a Google login and there are three different identities but they all represent you. So **Cognito can allow you to merge identities and treat them as one individual identity**.



So now that I have talked about the theory I want to just show you the architecture of these two main types of identity, and then I'm going to go across to a product that AWS provides, which is the Web Identity Federation Playground. Just to show you how this looks from a technical perspective. Now, the important thing to understand about AWS is that you cannot access AWS products without a set of AWS credentials. You can't access for example, S3 directly using a Google login. There has to be some form of identity exchange, and that is the foundation for any form of identity federation. So let's look at that and we'll start by looking at a SAML 2.0 identity flow. So I want you to imagine this scenario where you're operating in a large global enterprise and you utilize Microsoft Active Directory system for your identities. What you want to do is make it so that you can log in to the AWS console using your internal identity. Now to do that you might browse to your local ADFS portal on ADFS is the active directory federation server or federation service. It's a way that you can use your internal identities and make them available to other applications. So you browse this AWS portal and you have to authenticate using your internal Microsoft Active Directory identity. So your username and password and if you've already logged in to a local workstation and you're accessing this from a local workstation, you might not have to log in again. It might use single sign on. So let's say at this point you're logged in to this ADFS server. Well, what happens next is you get returned what's known as a SAML assertion. You can think of this as a token that proves that you are you. What happens next is this SAML assertion is delivered to the AWS single sign on endpoint. Now, this is an endpoint that you configure inside AWS if you want to use SAML Federated Identity, this essentially checks this SAML assertion potentially it might communicate with its ADFS server to verify everything's okay but there is a trust between this ADFS server and AWS at this point. So it will know that this SAML assertion is from this ADFS server and that it's valid. Now at this point, the SSL endpoint will return to your local workstation a URL redirect. Your desktop will use that URL and browse directly to the AWS console. Now what's happening in the background is the **single sign on endpoint communicates with STS on your behalf and assumes a predefined role and it assumes this role using the SAML assertion essentially, the token. Token that identifies you because we preconfigured it, STS allows this role assumption.** So essentially, your SAML assertion is swapped for some temporary security credentials and those credentials are the things that are used as part of this URL to access the AWS console and all that happens behind the scenes without you actually realizing it. So you're logged on to the console using a custom URL that includes these credentials. So you don't see any of the backend process, but you are able to log on to the AWS console without having to remember a separate username or password.



Now a similar process happens for Web Identity Federation with Web Identity Federation, I want you to imagine that you've downloaded and you're using either a mobile application or a web application. So you're browsing to this application on a local device, and you might be able to do certain elements without logging in so anonymous usage but at some point, maybe it's a forum system or maybe you need to log a high score, but at some point you'll be redirected to the ID provider to log in. So you're redirected for example, let's say, using a Google ID you'll log on using your Google ID and if successful, you get back a token. Now, once you've got this token, the mobile or web application then communicates either directly with STS or using Amazon Cognito, and it passes in that token and asks to assume a role. The role assumption happens, and you get back some temporary security credentials, and you can use those temporary security credentials to directly access AWS resources. Again really important thing to understand you **cannot access AWS resources directly with these external identities. You need to do an exchange first, and that's what will happen with Web Identity Federation.** You **exchange this token for temporary AWS credentials**. Now in the next lesson I'm going to be talking about when you might use Identity Federation. For the remainder of this lesson I want to give you a quick, brief demonstration. Now remember, there will be a hands-on lab at the end of this topic, where you'll get to experience ID Federation in an environment that we provide. For now, though, I just want to demonstrate this quick and simply.

SO this is a product that AWS make available called the Web Identity Federation Playground and this mirrors this type of architecture. I want you to pretend that the Web Identity Federation playground is this mobile or web application. So imagine that I've been using it for a while anonymously but now comes the time where I need to log in and get some additional access. I'll need to select an identity provider to use. We've got some different options made available. So we got Amazon, Google, and Facebook. There are others, but these ones that these tool supports. So I'll select Google and then click on Google sign in. Now I'm already logged in to my Google account in this web browser. I actually logged in twice once in my personal account and one in my Linux Academy account because I'm already logged in I'll be asked to select which identity to use. So this is this part the diagram. This is me authenticating with Google. This is not the application itself. The application has no involvement with my authentication, using username or passwords. So I'll click on my identity. My identity is going to be verified by Google and in response to that, because I'm successful, I'm going to get a token back in response and this is this token. It's essentially a unique representation that I've logged into Google. It successfully verified that I am that identity and in response, it's given me this token. Now, this token is something that AWS will know to trust because in the backend there is a trust configuration set up between AWS and this particular Google identity provider. So what I can do next is proceed to step two, and it's at this step where I will use that token that I have just got from Google. Remember, that token is a Google identity. It's not an AWS identity, so it can't be used to directly access any AWS services. What I need to do now is run an assume role with Web Identity that will assume a role inside AWS. It'll pass this token in as a parameter of that, and if the token is valid, it will allow me to assume that role and give me some temporary AWS credentials. So let's try that. So I'll click on call assume role with Web Identity. If I scroll down and look at the response the response is actually a set of temporary security credentials. So now I have some AWS credentials, and I can use those credentials by proceeding to step three to perform a list bucket operation. So I'll do that. I'll click on go and that will actually perform a list bucket operation. It'll be using my temporary security credentials and although this is text, it won't make a lot of sense. I can see that this bucket does exist and it's got a file or an object inside this bucket that's inside a folder called user\_fun and it's called user\_doc. What I might do is go down and go to get object. Click on go and when I run get object, it'll run on that object that I've just seen and that is actually the contents of the object on S3. So what I've done is I've logged into an external provider, got a token, tokens been returned to me, I have exchanged the token using an assumed role command, I have got back temporary security credentials, and I have used those to access an AWS resource.

Now you've got a number of different type of assume role API calls and each of these grant security credentials that are valid for different time frames. Now it's not essential that you know this for the exam, but I will include a link in the lesson description that provides additional details. We've got different time frames, depending on whether it's a normal assume role call, whether it's an assume role with Web Identity or whether it's the assume role call that's used with SAML ID Federation. They're all different time frames with different validities on these temporary credentials, but the architecture is always the same. You log in to an external identity provider, you are given something as a proof that you are logged in successfully, whether it's a SAML assertion or whether it's a token, you exchange that the AWS credentials and use those credentials to log in. That's always the same process that's followed with any form of Identity Federation. Now that's what I wanted to cover in this lesson about the technical details of ID Federation, remember, at the end of this topic, there will be a hands-on lab where you can experience this from a practical perspective. So don't worry about the practical exposure and in any case, you probably won't need to understand the implementation details for the exam but I always find that if you do it in practice, you get a little bit more confidence. Now this is the end of this lesson. In the next lesson I want to focus specifically on why and when you would use ID Federation because for the Solutions Architect Associate exam, it's often more important to understand when you need to use ID Federation. It's not about the implementation details, it's about the architecture. So go ahead and mark this video as complete, and when you're ready, you can join me in the next.