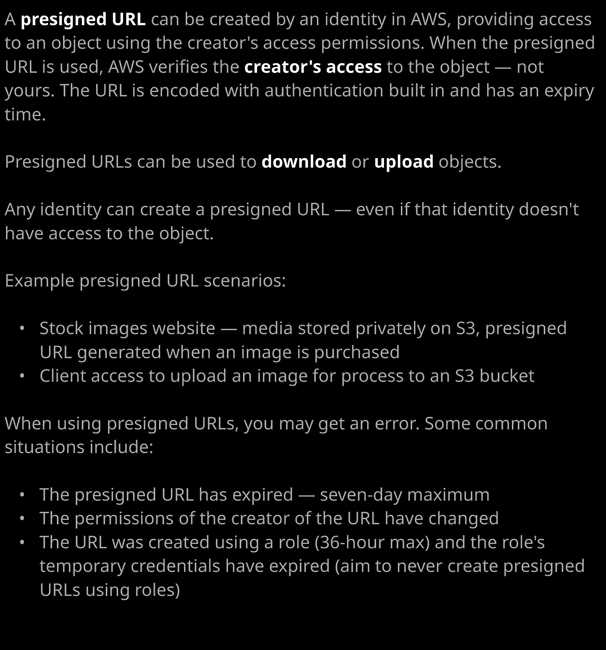
Presigned URLs.



Now a presigned URL is one that can be created by an identity inside AWS providing access to an object using the creator's access permissions. So what do I mean by that? Well, let's take this example of where we've got oops.jpg, which is an object in this S3 bucket. Now, at the moment that says S3 bucket is publicly accessible. So I'm going to fix that. I'm going to go to the bucket policy, and I'm going to delete the bucket policy that I created in an earlier lesson. So I'll select delete, delete that bucket policy. Now, at this point, that object is no longer publicly accessible because the bucket policy no longer applies. So if I select that particular object and I get the object URL. So I'll copy that into my clipboard, and then I move over to an incognito browser window and I attempt to load that I'll get an access denied error because I don't have any permissions to access the object. Now, by default, whenever you access an object in S3 the permissions check, the authorization check is performed at the point of accessing the object and essentially, what it's doing is determining what level of permissions my identity has and in this case, I'm using an unauthenticated or an anonymous identity so I have no permissions but that check is done at the point of when I access that object and it's done using my identity, the identity that I'm using at the point of access but presigned URLs change the way that that happens.

**Presigned URLs allow me as an AWS identity to generate a specific URL that has my access rights encoded into it.** So let me demonstrate this process. I'm going to generate a presigned URL. To do that, I'm going to move across my terminal and using the IAM user that has permissions on this S3 object, I'm going to run an AWS S3 presign command. What this is going to do is generate a presigned URL and then as a parameter, it needs the object that I'll be generating the URL for so s3:// and then the bucket name ac-catpics1337 and then the name of the object itself so oops.jpg. If I run that command, it's going to generate a presigned URL which includes the necessary security credentials to use my identity to access that object. So I'll copy that into my clipboard, I'll move across my web browser, I'll paste in this presigned URL, and this time I'll be allowed access to the object. The important thing to understand architecturally is when I created this presigned URL I essentially delegated myself. I made this URL and this URL accesses the object as me. So even though I'm accessing the object in an unauthenticated fashion because I'm using the URL, I'm essentially passing to S3 something that demonstrates that it's actually my identity.

Now the architecture of how this might be useful is that you could have an S3 bucket that is not public access. It's completely private. Let's say you've got some sensitive images on there or you're running a stock images website, and an application could generate a presigned URL that gives the bearer of that URL access to the object using the permissions of the entity that generated it. So essentially an application could be running on an EC2 instance, it could have some AWS credentials to use. It would generate a presigned URL, deliver that URL to a customer of your application in their web browser, and they could use that presigned URL to access restricted objects.

**Now a presigned URL has an expiry time.** Essentially, it will operate until it expires, and it's a configurable value. You don't need to know the range of values for the associate level exam, but do be aware that it can expire. Now another important thing to understand is that when you're using the presigned URL you're essentially acting on behalf of the person who created it. So you are using their credentials**. Now, if that user credentials—so let's say that I generated a presigned URL and gave that URL to you if my credentials were changed, if I no longer have permission to that object, you would not have permission to the object either**. So the URL is tied to the person who created it. The other important thing to understand is that you can actually create a presigned URL to an object that you don't have permissions to and if you do that then of course, whenever anyone uses that presigned URL to access an object, they will also get an error.

So there are some common things that you might face in your exam. **If you can't use a presigned URL to get access to an object then either the presigned URL might have expired, the permissions that the creator of the URL might have changed, and another really interesting one is that the presigned URL is linked to the credentials that generate it.** So, for example, **if you generate a presigned URL using an EC2 instance that has an attached instance role and the credentials that role expire, which they do on a periodic basis then any presigned URLs that you generate using that role will expire as soon as that role does.**

So it's really important to understand this. **AWS recommends that you don't generate presigned URLs using roles because you will have expiry issues. You should always use identities with long term credentials to generate these presigned URLs because by doing that you'll make sure that the only way that they can expire is when their own expiry time arrives and the URL itself expires.** So, essentially, **presigned URLs give the holder of the URL access to an object as though they were the person who created that URL** and as I said at the start of this lesson **it's generally used if you're running things like stock image websites or you want to provide people access to a certain object in a time limited fashion. Now you can use presigned URLs to upload objects to S3 as long as the creator of that URL does have access rights to be able to put objects into S3 and then holder of the URL can use it to upload those objects to S3. So that's another important thing to understand.** They could be used to download or get as well as upload or put objects into S3.