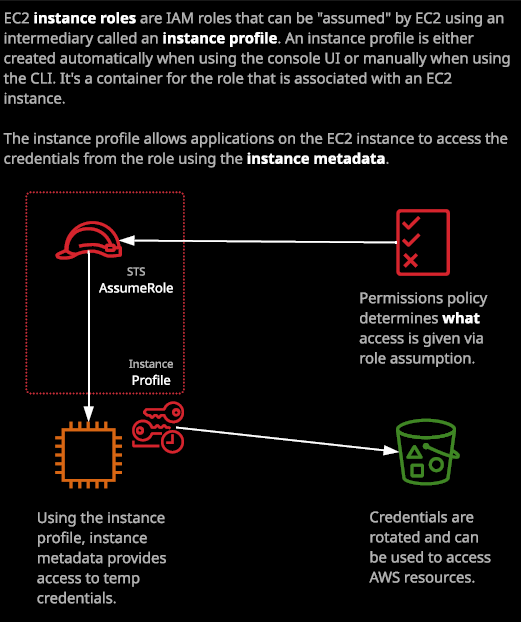
**EC2 instance roles**

  
  
An instance role is essentially a type of IAM role, which could be used by EC2 and EC2 can use it by using an intermediary known as an instance profile.

IAM roles and they're an identity, which gets assume don't log in to a role, they are always something that you assume in order to get additional permissions. So an EC2 instance role is just a role that could be assumed by EC2.

So an instance role is nothing special. It's just an IAM role, but it's one that's been configured in a certain way to allow it to be used by EC2.

Now what type of scenarios you might want to use an instance role do vary but essentially it allows you to give an EC2 instance or any applications running on that instance permission to interact with AWS.

Now, an example of this is that EC2 instances do come with the AWS command line tools installed. So, for example, if I wanted to run a list of every S3 buckets inside this region, I could go aws s3 ls and that would do an S3 listing. So show every bucket available in this account. Now, if attempt to do that, it's going to tell me that I don't have any credentials and that's logical because I haven't run AWS configure. Running AWS configure would result in access keys being stored permanently on this EC2 instance and I might not want that this instance might be something that runs several applications or it might be an instance that's running a third party application or something that could even be exposed as this instance is to the public internet. **So if I configured any long term access credentials on this instance and it was available to the public internet, it could, in theory, be a security risk and instance roles are one way to avoid that.** **Instance roles allow the EC2 instance to assume that role and to gain access to temporary security credentials.**

So this is the architecture. What actually happens is you take a normal IAM role and an IAM role is something that can be assumed by an identity in AWS. Now, the EC2 service is an identity, but applications running on the instance are not valid AWS identities, they're just applications and so what you need to do is create something that's known as an **instance profile and an instance profile is essentially just a container for an IAM role that you can use to pass that role information to EC2.** It's essentially just a **translating container**. **It allows applications on EC2 to use that role on your behalf, so you're not just letting EC2 assume the role you're allowing the applications to utilize that role as well.** Now, creating the instance role and creating the instance profile are actually two different steps, and if you do this from the command line, you'll need to create them both separately. If you use CloudFormation, you'll need to create them both separately but if you do it from the console, it's one step. Creating an instance role in the console so that's just a normal IAM role that's designed for EC2 to also creates an instance profile of the same name.

STS, also known as the security token service now, this is the service that's responsible for generating short term or temporary credentials whenever a role is assumed. What happens at this point is that the EC2 instance begins assuming the role. The STS service every time that occurs, which is constantly while this role is associated with the instance, the STS service begins delivering short term temporary access credentials, and they're available from within the instance metadata. So the STS service delivers temporary security credentials into that instance. Now that means that anything that's running inside the instance, has access to those temporary security credentials. So now if I do an aws s3 ls and press enter I no longer get the requirement to run AWS configure and that's because the AWS command line tools are aware of the metadata of an EC2 instance, and if it's present, they'll use that information to access AWS. So they're using temporary access credentials that are inside the instance metadata in order to interact with AWS. Now, the AWS command line tools will use a number of different sources for authentication. The first that you're familiar with so far is the statically configured information when you run AWS configure. So the access credentials that you enter if you run an AWS configure. You're also able to set AWS credentials. So the access key ID and the secret access key set, though, is inside environment variables and when you do that, when you name the environment variables a certain name then the command line tools can interact using those and then obviously this way is that they can access the metadata inside the instance and interact that way.

Now this presents a number of significant advantages.

I'm able to create a normal IAM role and an associated instance profile that I can associate that with 1, 10, or hundreds or even thousands of EC2 instances. Those instances work in conjunction with STS to generate temporary security credentials that are available inside that instance to applications running on that instance. Now the key thing is that these are temporary security credentials, and they're rotated every time EC2 assumes this IAM role and so it means if this instance is exploited and the temporary credentials are leaked, it means that the impact of that is limited because they're not permanent credentials they expire periodically and so, even though you might have the credentials short term, once they do expire, you'd need to reassume the role to get access to new credentials and so, **by using this temporary, rotating credential architecture, you offer a much improved security situation.**   
It also means that you don't need to worry about how you deliver credentials to these instances. Imagine if we didn't have this feature, you'd need to find a way of delivering access keys into an instance. Now maybe you'd run a user data script to deliver these in but keep in mind they're not encrypted when you do that, and they're accessible to applications on the instance, and these would be long term credentials, so they're much more dangerous. You could store the credentials in a file on S3, but then you need to think, how would I give access to S3? How could you initially give that instance the ability to access the object with the credentials in? **Using IAM roles in this way offers a really great way to deliver security credentials to AWS services. You don't have to worry about supplying the actual credentials. You can allow the service to assume the role on your behalf and be given temporary security credentials** and you'll see that that's the way that a lot of AWS services work. Whether it's EC2 or Lambda or any other service that interacts on your behalf, it's always preferable to use IAM roles. Remember that for the exam. Whenever you have a preference, use IAM roles.

The only scenario where IAM roles really don't work is when a log in is required. So when a user needs to log in because, of course, IAM roles don't have any long term credentials. Now as well as attaching an IAM role to this instance I could go to instance settings and attach and replace another. I could change it back to no role. I could even change the role and then apply it and then when that role gets assumed again, it would get a new set of temporary access credentials that would have the new role's permissions policy. **So it's not a one time only thing you can associate roles now, with a pre running instance, you can change it and then disassociate the role afterwards. It's a lot more flexible than supplying the credentials of a particular IAM user into the instance in a static way.**