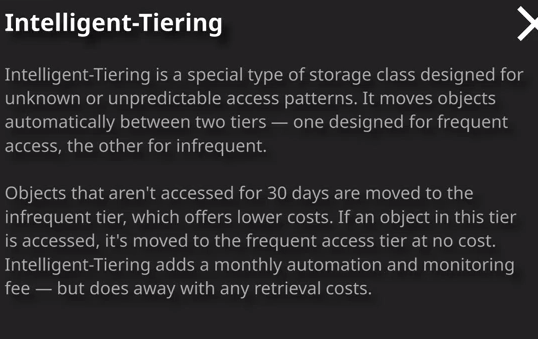
The first is another storage class known as **intelligent tiering**, and second I want to talk about **lifecycle rules, which allow us to automate the process of moving objects between all of the available storage classes.** So let's jump in and get started.



Now the first thing I want to talk about is intelligent tiering and intelligent tiering the way I want you to think about this is essentially it's another storage class available within S3 and the attempts to combine the two different main types of storage classes. It's designed for data where you've got unknown or unpredictable access patterns but unlike the other storage classes, you don't get penalized for being unable to pick between these two. Essentially, what intelligent tiering does is you can select an object. So, for example, Winky.jpg go to actions, change storage class, and then change it to intelligent tiering. Now the process of changing this object storage class to intelligent tiering is exactly the same as any of the other static storage classes. The difference being intelligent tiering is actually an automated process that moves an object between two different types of storage based on its access patterns. So I'm going to go ahead and change this object to use intelligent tiering. What that essentially does is that if this object doesn't get access for 30 days, then it moves the object into an infrequent tier which offers lower storage costs for that object. If an object in this tier is accessed, then it gets moved from this infrequent tier to the frequent access tier, and it's moved at no cost. So unlike the other static storage classes, instead of paying a retrieval fee or paying for any movement between these classes with intelligent tiering, you don't. All that you do with intelligent tiering is that you essentially pay a monthly automation and monitoring costs, but it essentially does away with any retrieval costs. So if you've got any data where the access patterns can be random or they're unknown or you're just not sure, then you can put it into intelligent tiering and have S3 automatically moved the data between these different tiers, as required. Now for an exam perspective, **generally only pick intelligent tiering for those use cases where you really don't want the admin overhead or when you don't know the access pattern, because this monthly automation of monitoring fee especially for objects that don't need to move between these different storage classes this can add up. If you've got data where you know the access patterns where they're consistent, on where you know exactly where to place them then if you pick intelligent tiering then you're essentially spending money for no benefit but if your object access patterns are unknown or inconsistent than intelligent tiering does offer a really efficient place to put those objects, so keep that in mind for the exam**. Now, from a configuration perspective, intelligent tiering is configured in exactly the same way as any of the static storage classes. You can move objects into this storage class after you've uploaded them to S3 or you can pick intelligent tiering when you're uploading the objects initially. For everything else, it's exactly the same as the existing storage classes. It's just that once objects are in this storage class, then they benefit from the automation and management that S3 provides. Okay, so that's intelligent tiering.

The next thing I want to talk about are lifecycle rules. **Storage classes inside S3 can be controlled either manually or using lifecycle rules and lifecycle rules allow for the automated transition of objects between storage classes or in certain cases, you can actually configure automated expiration of objects that are no longer required. Now, lifecycle rules are actually defined at a bucket level.** So if I click on the management tab and then go to lifecycle and then add a lifecycle rule it's here that you can define that inside the console U. So I'm going to create a rule. I'm going to call it test, **and you can actually have a rule applying to the entire bucket or you can filter based on prefixes or tags.** That's important to remember the exam but for this demonstration I was just going to leave this blank so click on next. Now you're able to select for the storage class transition whether you want it to occur for the current version of objects or the current version and the previous versions. So this offers a useful way that you can act on either the latest or previous versions of an object. In this case, I'm going to select both. At this point, **you're able to create a rule, a transition rule, so I might want something to happen after a certain amount of time to the current version or the previous version of objects**. Let's say that for any previous version of objects I want to add a transition, and I want to transition any previous object versions to Glacier after the objects become noncurrent. Now noncurrent is the word AWS uses to specify anything but the most current version of an object. So what this will do if I set this to 30 it means that for any versions of objects inside this bucket after they become the noncurrent versions so 30 days after that, they'll transition to Glacier. I can also define a similar rule for the current version of objects, so I could define that for the current version of objects. I want to transition it to intelligent tiering or one zone IA 30 days after creation. So this allows can automate a process where people can upload objects to this bucket and the potentially say, 30 or 60 or 90 days after this upload you could change the storage class to something which offers cheaper storage costs and then you can say for any older versions of those objects after they become noncurrent, you could transition them to Glacier and so offers you a rule based feature to change the storage class of objects in your S3 bucket. Now as well as these transitions between storage classes. **You're also able to expire objects. So you could, for example, say that for previous versions of an object, you could permanently delete it 395 days after it becomes a previous version. So recall from the previous lesson when I talked about versioning inside an S3 bucket, I mentioned that you're actually build for both for current and all of the previous versions of objects inside an S3 bucket. This offers you the ability to permanently delete any previous versions after they become a previous version. Now you can also configure expiration on a current version. So you could say that after a certain amount of time that a current version is created, it also gets deleted and again, this reduces the admin overhead of managing objects in an S3 bucket. So if you, for example, run a stock images website or you're running a social media image sharing website then you might want to manage all of the current and previous versions of objects in the S3 bucket and by using lifecycle rules, you get an automated way of doing that.**

This diagram on the right of my screen shows the path that these lifecycle rules can follow so you can go from S3 standard—note how it's standard is the default storage class available in S3. You can go from that to any of the other storage classes so standard IA, one zone IA, intelligent tiering, and even Glacier but as you move to these other storage classes, you can't go back to standard using these automated lifecycle rules. So that's important to understand for the exam. So think about this is moving downwards so you go from standard to stand that IA and then to one zone IA and then to Glacier but you can't follow that same process in reverse, So lifecycle rules are especially important for the exam. They could be used to transition between storage classes, or they could be used for automatically clearing up i.e. expiring objects inside the S3 bucket.

