Working with Amazon S3 Lifecycle Configurations

NOTE: This content will be available as a video course soon! We are making it available as a Learning Path resource in the meantime to help you prepare for the newly updated AWS Certified Solutions Architect - Associate (SAA-C03) exam.

Introduction

Hello and welcome to this course, which will explain the ins and outs of S3 lifecycle configurations.

Before we begin, I'd like to introduce myself - my name is Alana Layton and I'm an AWS content creator here at Cloud Academy. Feel free to connect with me to ask any questions using the details shown on the screen. Alternatively you can always get in touch with us here at Cloud Academy by sending an e-mail to support@cloudacademy.com, where one of our Cloud experts will reply to your question.

This course has been created for those that are looking to better manage their storage lifecycle and S3 storage costs, or those already planning on implementing or using lifecycle configurations that may need a bit more information on how they work.

By the end of this course, you will have a greater understanding of:

- what a lifecycle configuration is
- why they are beneficial,
- how to create a lifecycle configuration, and
- the limitations of lifecycle configurations

To get the most out of this course, you should have a strong understanding of S3, including knowledge on storage classes, multipart uploads, object versioning, object tags, and prefixes. Familiarity with XML and JSON will help as well to better understand some of the practical parts of this course. For more information on the S3 service, check out existing content here:

Deep Dive: Amazon S3
https://cloudacademy.com/learning-paths/deep-dive-amazon-s3-955/

Feedback on our courses here at Cloud Academy is valuable to both us as trainers and any students looking to take the same course in the future. If you have any feedback, positive or negative, it would be greatly appreciated if you could contact support@cloudacademy.com.

The case for Lifecycle Configurations

Consider the following scenario: you have a data lake workload in a versioned S3 bucket that grows at a very fast and consistent pace. Tons of overwrites are occurring daily and large objects are being uploaded via multipart upload.

And because bad things often happen to good people - that means, that from the data management perspective, you may have incomplete multipart uploads every so often, and tons of out-of-date non-current versions of your data.

I'm sure you can probably imagine how costly it is to run this data swamp - I mean, lake.

So it's important that you consider every cost tool at your disposal. And one of these tools is to manage the storage lifecycle of your data, by moving your data to lower cost storage classes, or deleting data you no longer need. You can, of course, move and delete your data manually, but that can be difficult to manage at scale. So, there are two ways to automate this process:

The first way is to use the S3 Intelligent-Tiering storage class. You'll pay a monthly object monitoring and automation charge, and in return S3 Intelligent-Tiering will monitor your object access patterns, and automatically move your objects between three tiers: frequent, infrequent and archival. S3 Intelligent-Tiering is recommended for data access patterns that are unknown or unpredictable, and is meant to give a more "hands-off" approach to managing your data lifecycle.

The second approach is by using Lifecycle configurations. You can use lifecycle configurations to transition data to a lower cost S3 storage class, or to delete data. Lifecycle configurations additionally provide options to clean up incomplete multipart uploads and manage noncurrent versions of your data - which ultimately, helps reduce storage spend.

Using lifecycle configurations is the most cost-effective strategy when your objects and workloads have a defined lifecycle and follow predictable patterns of usage.

For example, a defined access pattern may be that you use S3 for logging and only access your logs frequently for at most, a month. After that month, you may not need real-time access, but due to company data retaining policies, you cannot delete them for a year.

With this information, you can create a solid lifecycle configuration based on this access pattern. You could create an S3 Lifecycle configuration that transitions objects from the S3 standard storage class to the S3 Glacier Flexible Retrieval storage class after 30 days. By simply changing the storage class of your objects, you will begin to see significant cost savings in your overall storage spend. And after 365 days, you can then delete the objects and continue to save on costs.

You may find that a lot of your data follows a similar access pattern: you slowly stop needing real time access to your data, and can eventually delete the data after a certain period of time passes. Or you may have data that you need to save to meet some compliance or governance regulation that can be moved from S3 Standard to archival storage and left alone for long periods of time. Or perhaps, you have a ton of objects in S3 Standard storage and you want to transition all of those objects into the S3- Intelligent Tiering storage class.

If these patterns sound similar to your use case, then using lifecycle configurations makes sense for your workload.

In summary, Lifecycle configurations are an important cost tool that can enable you to delete or transition old unused versions of your objects, clean up incomplete multipart uploads, transition objects to lower cost storage tiers and delete objects that are no longer needed.

Components of a Lifecycle Configuration

An S3 Lifecycle configuration is technically an XML file. While you won't need to know XML to create lifecycle configurations in the AWS console, the XML format is helpful in understanding the various components to see how it all works under the hood. For that reason, I'll be describing the anatomy of a lifecycle configuration using XML throughout this video.

Each lifecycle configuration contains a set of rules.

```
<LifecycleConfiguration>
    <Rule>
         ...
    </Rule>
    <Rule>
         ...
    </Rule>
</LifecycleConfiguration>
```

Each rule is broken up into four components: ID, Filters, Status, and Actions.

```
<LifecycleConfiguration>
    <Rule>
        <ID></ID>
        <Filter></Filter>
        <Status></Status>
         Actions
    </Rule>
</LifecycleConfiguration>
```

The ID uniquely identifies the lifecycle rule - you can consider this the name of the rule. This is important, as one lifecycle configuration can have up to 1000 rules, and the ID can help you keep track of what rule does what.

```
   <Rule>
        <ID>ProjectBlueRule</ID>
   </Rule>
```

The filters section defines WHICH objects in your bucket you'd like to take action on. You can choose to apply actions to all objects or a subset of objects in a bucket. If you choose a subset of objects, you can filter based on prefix, object tag, or object size. Or to be very granular, you could filter based on a *combination* of these attributes. For example, you can create a filter that transitions all objects with the prefix ProjectBlue/, if they also are tagged with the Classification tag value "Secret". Notice that when you combine multiple filters, you have to use the keyword "And" in XML.

```
   <Rule>
        <ID>ProjectBlueRule</ID>
        <Filter>
          <And>
             <Prefix>ProjectBlue/</Prefix>
              <Tag>
                <Key>Classification</Key>
                <Value>Secret</Value>
              </Tag>
          </And>
        </Filter>
    </Rule>
```

For object size, I can transition or expire objects if they're greater than a specific size, less than a specific size, or if they're in a range between two size bounds. For example, I can create a filter, where I'm transitioning my objects if they're larger than 500 Bytes, but smaller than 10,000 Bytes. Keep in mind that the maximum filter size is 5TB.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>10000</ObjectSizeLessThan>
            </And>
        </Filter>
    </Rule>
</LifecycleConfiguration>
```

The next component is status. With the status field, you can enable and disable each lifecycle rule. This can be helpful when you're testing lifecycle configurations out, as you figure out what the best rules for your workload are. Once you change the status to "disabled", S3 won't run the actions defined in that rule - essentially stopping the lifecycle action. And when you're ready to run those actions on your objects again, you can always change the status back to enabled.

```
   <Rule>
        <ID>ProjectBlueRule</ID>
        <Filter>
           <Prefix>ProjectBlue/</Prefix>
        </Filter>
        <Status>Enabled</Status>
    </Rule>
```

The last, and arguably the most important component is the actions section. This is where you define WHERE you want your objects to move to - are you transitioning them to another storage class or are you deleting them?

There are six main actions that you can use: Transition, expiration, NoncurrentVersionTransition, NoncurrentVersionExpiration, ExpiredObjectDeleteMarker, and the AbortIncompleteMultipartUpload action.

```
<Rule>
     <Transition></Transition>
     <Expiration></Expiration>
     <NoncurrentVersionTransition></NoncurrentVersionTransition>
     <NoncurrentVersionExpiration></NoncurrentVersionExpiration>
     <ExpiredObjectDeleteMarker></ExpiredObjectDeleteMarker>
     <AbortIncompleteMultipartUpload></AbortIncompleteMultipartUploa
     d>
```

```
</Rule>
```

The first two actions are transition actions and expiration actions. Transition actions enable you to move data automatically between S3 storage classes.

Expiration actions enable you to automate the deletion of your objects in S3.

For both transition and expiration actions, you can define when to move or delete these objects based on object age. And age is based on when the object was last created or modified.

Here's an example of how both of these actions are structured in XML. In this example, the first action transitions all objects that are prefixed with ProjectBlue/ to the S3 Glacier Flexible Retrieval storage class 365 days after they were created. The second action says after 2,550 days - which is 7 years, delete the objects prefixed with ProjectBlue/ because they aren't needed any longer.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ProjectBlueRule</ID>
        <Filter>
           <Prefix>ProjectBlue/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>2555</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

If you have versioning turned on for your bucket, transition actions and expiration actions only work for the current version of your object.

If you want to transition *noncurrent* versions of your object, you must use the NoncurrentVersionTransition action.

Similarly, if you want to delete *noncurrent* versions of your object, you must use the NoncurrentVersionExpiration action.

For both the NoncurrentVersionTransition and NoncurrentVersionExpiration actions, you can define when to move or delete objects based on two things.

- First is the number of days since the object has been noncurrent, which Amazon calculates as the number of days since the object was overwritten or deleted.
- And second, is the maximum number of versions to retain. This is helpful when you want to save a few versions to rollback to for data protection, while removing old versions of your object to save on storage spend.

Here's an example of a lifecycle configuration that uses both a noncurrent version transition and a noncurrent version expiration action. In this rule, all noncurrent versions are moved to S3 Standard - Infrequent Access 30 days after they become noncurrent. In addition, it deletes all noncurrent versions 730 days, or two years, after they become noncurrent, while retaining the 3 latest versions. Keep in mind, you can choose to retain any number of versions between 1 and 100.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ProjectBlueRule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>STANDARD_IA</StorageClass>
        </NoncurrentVersionTransition>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>3</NewerNoncurrentVersions>
            <NoncurrentDays>730</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

There are additional special actions that you can take as well.

One of them is the Expired object delete marker action. This is helpful if you have objects that have zero versions, that only have a delete marker left. This is referred to as the expired object delete marker, and you can use this action to remove them.

And last, there is the AbortIncompleteMultipartUpload action. If you have incomplete multipart uploads that you need to clean up, you should use this action. With this action, you can specify the maximum time, in days, that your multipart uploads can remain in progress.

For example, you can specify that your multipart uploads can remain in progress for 14 days. If the upload does not complete in 14 days, S3 will delete it.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ProjectBlueRule</ID>
        <Filter>
           <Prefix>ProjectBlue/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
           <DaysAfterInitiation>14</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

In summary, S3 lifecycle configurations are made up of an ID, filters, status, and actions. That's all for this one!

Other Considerations and Limitations

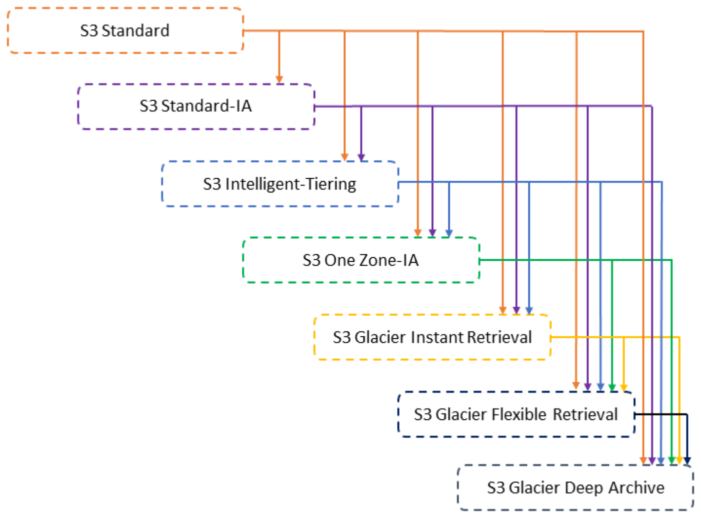

Image from https://docs.aws.amazon.com/AmazonS3/latest/userguide/images/lifecycle-transitions-v3.png.

I want you to think about the S3 storage classes as a staircase. S3 Standard storage is at the top of the staircase, while S3 Glacier Deep Archive is at the bottom of the staircase. And all of the other storage classes are in between.

With lifecycle configurations, this staircase only goes one way: down. Once you transition data down the staircase to a lower cost storage class, you can't move objects back up. For example, let's say I move my data to S3 Standard-Infrequent Access. Once my data transitions to that storage class, I can't use a lifecycle configuration to move my data back to S3 Standard. This is also true if I move my data to S3 One Zone - IA, I can’t move it back to S3 Intelligent-Tiering, S3 Standard-IA, or S3 Standard.

This becomes important if I'm using archival storage. If I reach the bottom of the staircase, by moving my data to the lowest cost storage class, S3 Glacier Deep Archive, I can't transition my data back to any other storage tier.

Lifecycle Configuration costs follow a similar staircase model. I categorize these costs in two ways: minimum storage duration fees, and storage transition costs. Both of which increase as you move down the staircase.

For example, let's take storage transition costs. You get charged when you move data to other storage classes and this fee increases as you move down the staircase.

For example, at the top of the staircase, you’re charged $0.01 for every 1,000 lifecycle transition requests when objects are moved from S3 Standard to the S3 Standard-IA storage class.

As you go down the staircase, all the way to S3 Glacier Deep Archive, this cost increases, and can be up to $0.05 for every 1000 transition requests.

While it might not seem like a huge cost, it can stack up over time, especially if you're consistently moving data to archival storage. For example, if you need to transition millions of small objects to archival storage, that transition cost can be very high. To minimize this cost, you should consider transitioning mostly large objects that need to be retained over long periods of time. You can also consider aggregating several small objects into one large object to save on this fee as well.

The second cost factor related to lifecycle configurations is minimum storage duration fees. Most storage classes have a minimum storage duration that requires you to keep data in a storage class for a certain period of time before you delete, overwrite, or transition those objects.

These minimum storage duration periods increase as you go down the staircase as well. For example, S3 Standard and S3 Intelligent-Tiering have no minimum storage duration. Infrequent access tiers like S3 Standard-IA and S3 One Zone - IA have a minimum storage duration of 30 days. Archival storage tiers like S3 Glacier Instant Retrieval and S3 Glacier Flexible Retrieval have a minimum storage duration of 90 days, while S3 Glacier Deep Archive has a minimum storage duration of 180 days.

So what happens if you delete or overwrite these objects before the minimum storage duration is reached? You get charged. For example, say you transition an object into S3 Glacier Deep Archive for 30 days, and then delete it. In this case, you will still be charged for the full 180 days of storage.

So when you're setting up your lifecycle configurations, ensure that you're keeping the limitations and costs in mind. That's all for this one! See you next time!

Summary

In this course, I talked a lot about lifecycle configurations. At this point, you should have a better understanding of lifecycle configurations, including what they are, how to use them, and how they can help minimize storage spend.

Hopefully you're familiar with the structure of a lifecycle configuration, and excited to get started with the service and create lifecycle configurations to better manage your storage costs.

If you would like to get some hands-on experience with Lifecycle configurations, then take a look at the following lab:

Handling S3 Objects Events With Lifecycle Policies and Server Access Logging
https://cloudacademy.com/lab/handling-s3-objects-with-life-cycle-policies/

Once again, my name is Alana Layton and I hope you've enjoyed our time together. If you have any feedback, positive or negative, please contact us at support@cloudacademy.com. Your feedback is greatly appreciated. Thank you and till next time!

Alana Layton, Sr. AWS Content Creator
alana.layton@cloudacademy.com