KodeKloud

Visit www.kodekloud.com to learn more.

Exploring Azure Blob Storage

Introduction

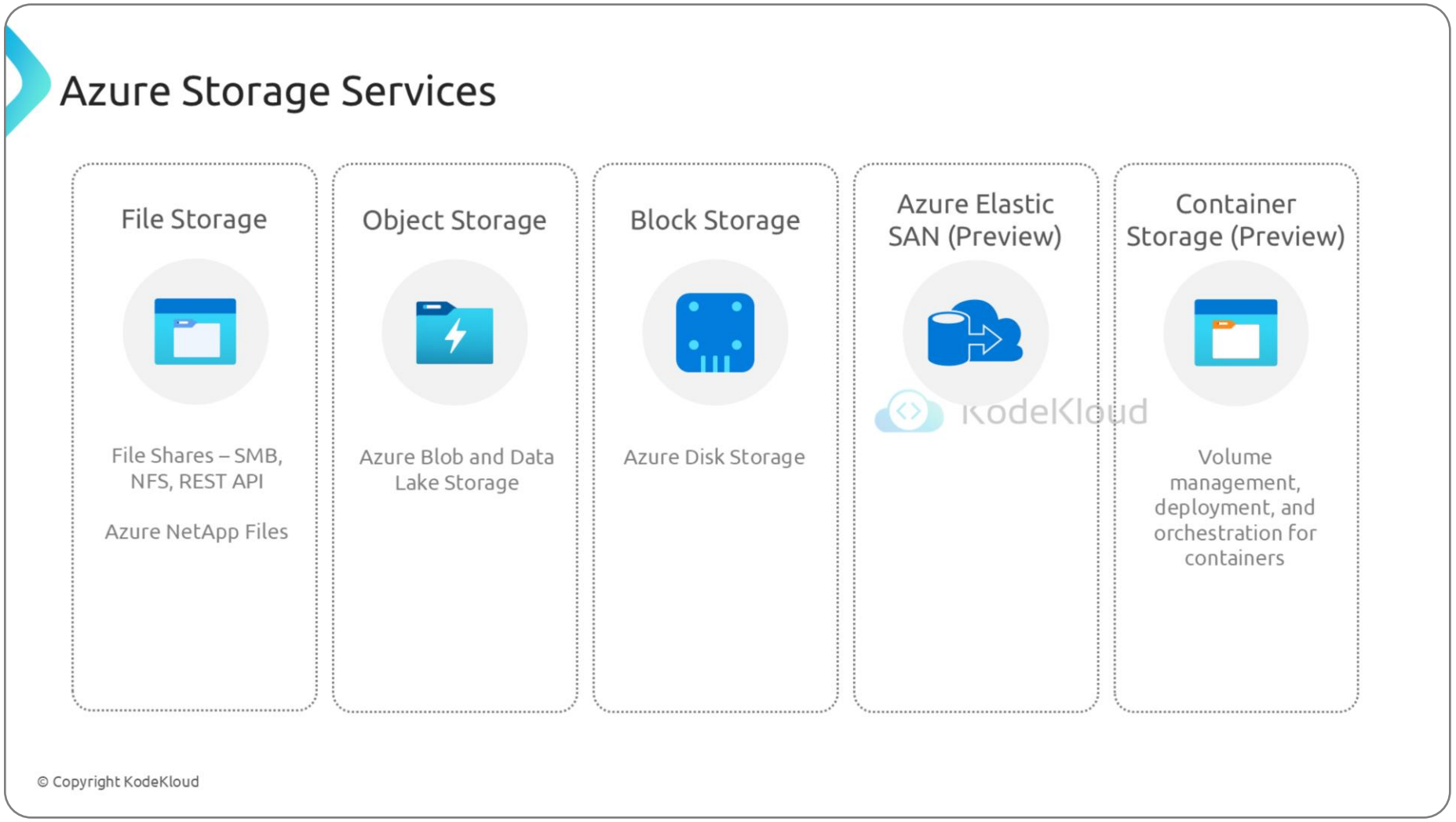
01 Understand the core features and functionality of Azure Storage and Azure Blob Storage.

02 Understand the availability and performance tiers of Azure Storage.

03 Learn blob hierarchy, naming convention, and blob types.

04 Understand Azure Storage security features.

Azure Storage Services



While the AZ204 certification focuses entirely on Object storage, it is worth noting that Azure also provides other storage services such as:

File storage: Which is designed to store files in a shared folder structure. If you need extremely high performance, you can use NetApp files as a native service on Azure.

Object Storage: As I mentioned we will focus almost entirely on object storage in this module, but just to differentiate with

other options - Object storage is designed to store small files, such as images, videos, and documents. Object storage is highly scalable and cost-effective, and it can be used to store large amounts of data.

Block storage: Block storage is crafted for the storage of sizable files, including virtual machine disks and database files. It boasts high availability and durability, offering various tiers for selection based on your specific performance and cost preferences.

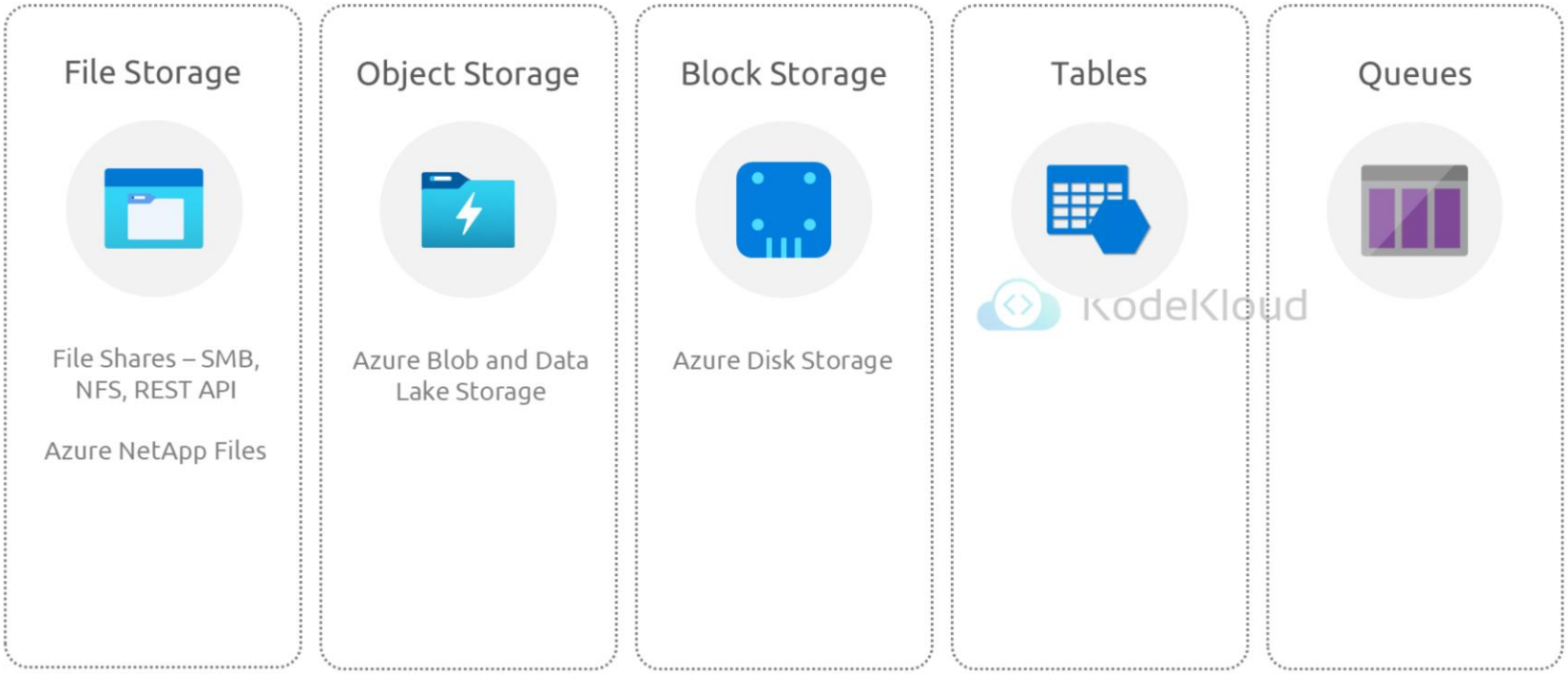
The Container Storage, which is still in Preview as of this recording, seamlessly integrates with Azure Kubernetes Service. This service enables automatic and dynamic provisioning of persistent volumes for storing data used by stateful applications running on AKS clusters.

Typically, Microsoft certification exams do not include assessments for services that are in Preview, unless the exam itself is in a pre-release stage.

Apart from these key services, Storage also supports, Queues which can maintain a list of messages that can be added by a sender component and processed by a receiver component.

If you had used Azure storage before, you might have heard table storage too. That service is now part of Cosmos DB.

Azure Storage Services



While the AZ204 certification focuses entirely on Object storage, it is worth noting that Azure also provides other storage services such as:

File storage: Which is designed to store files in a shared folder structure. If you need extremely high performance, you can use NetApp files as a native service on Azure.

Object Storage: As I mentioned we will focus almost entirely on object storage in this module, but just to differentiate with other options - Object storage is designed to store small files, such as images, videos, and documents. Object storage is highly scalable and cost-effective, and it can be used to store large amounts of data.

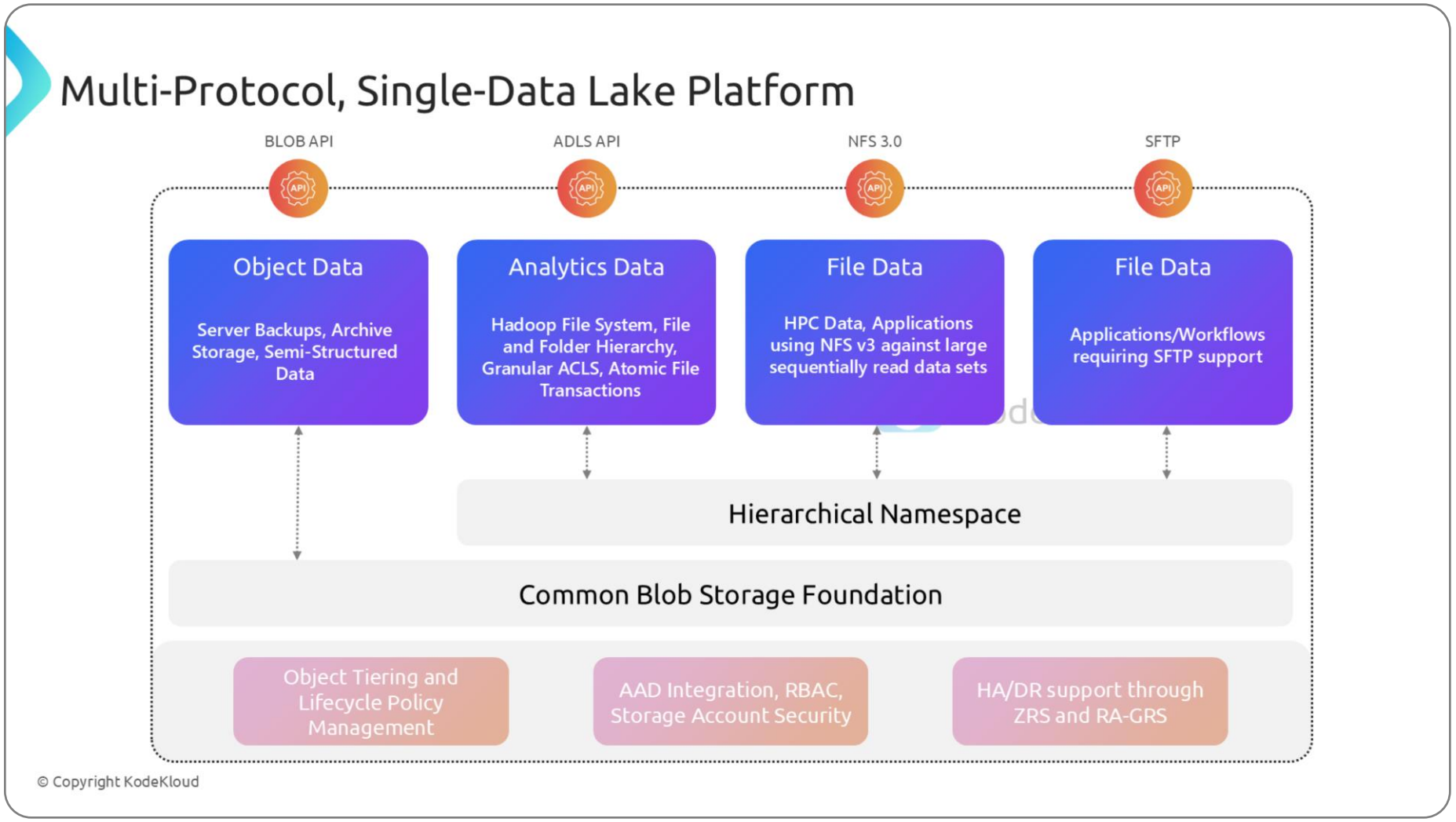
Block storage: Block storage is crafted for the storage of sizable files, including virtual machine disks and database files. It boasts high availability and durability, offering various tiers for selection based on your specific performance and cost preferences.

The Container Storage, which is still in Preview as of this recording, seamlessly integrates with Azure Kubernetes Service. This service enables automatic and dynamic provisioning of persistent volumes for storing data used by stateful applications running on AKS clusters.

Typically, Microsoft certification exams do not include assessments for services that are in Preview, unless the exam itself is in a pre-release stage.

Apart from these key services, Storage also supports, Queues which can maintain a list of messages that can be added by a sender component and processed by a receiver component.

If you had used Azure storage before, you might have heard table storage too. That service is now part of Cosmos DB.

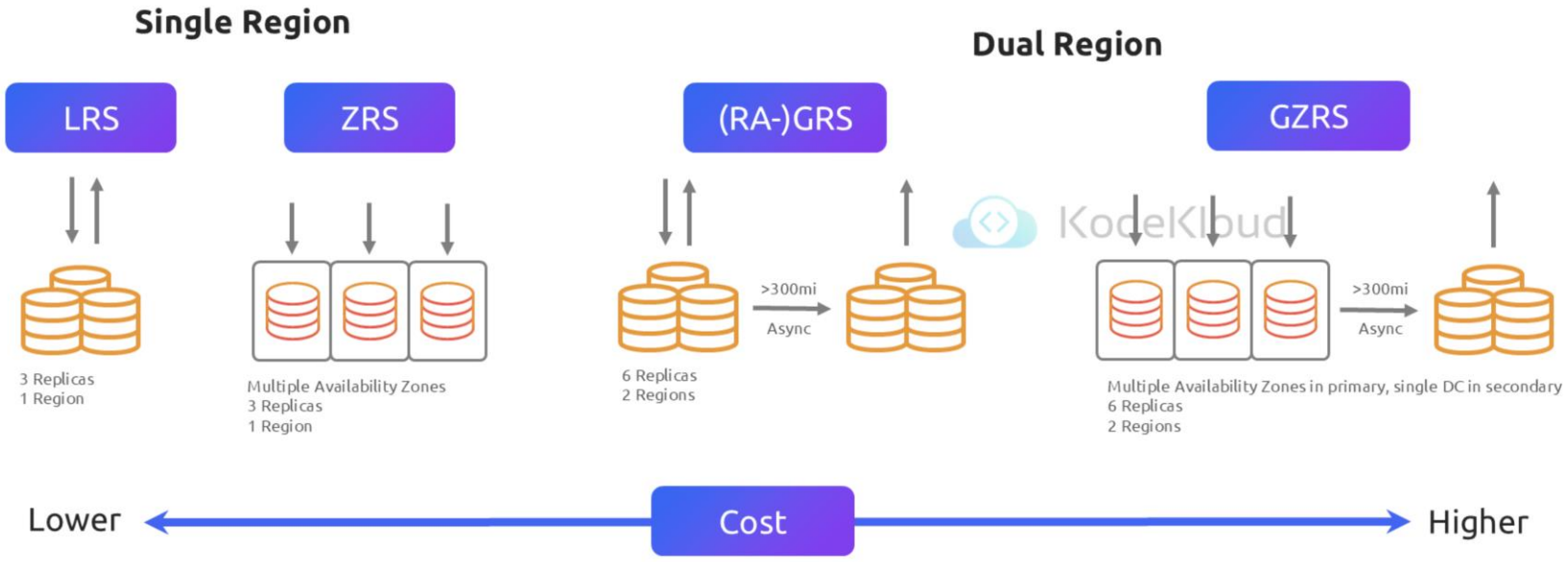


So lets jump on to Blob storage concepts

Blob storage is an object storage service, provides a REST API i.e the Blob API build on top of the Azure Blob storage layer related to reliability and security features. As you can notice, there is another API endpoint that you can access called ADLS Gen2 API that allows analytics engines to plug in and treat blob storage like a native HDFS interface. We have also other endpoints like NFS and sFTP that can only be enabled if you are deploying a Hierarchical namespace based blob storage.

Azure Storage – Availability and Performance

Azure Storage – Durability and Availability



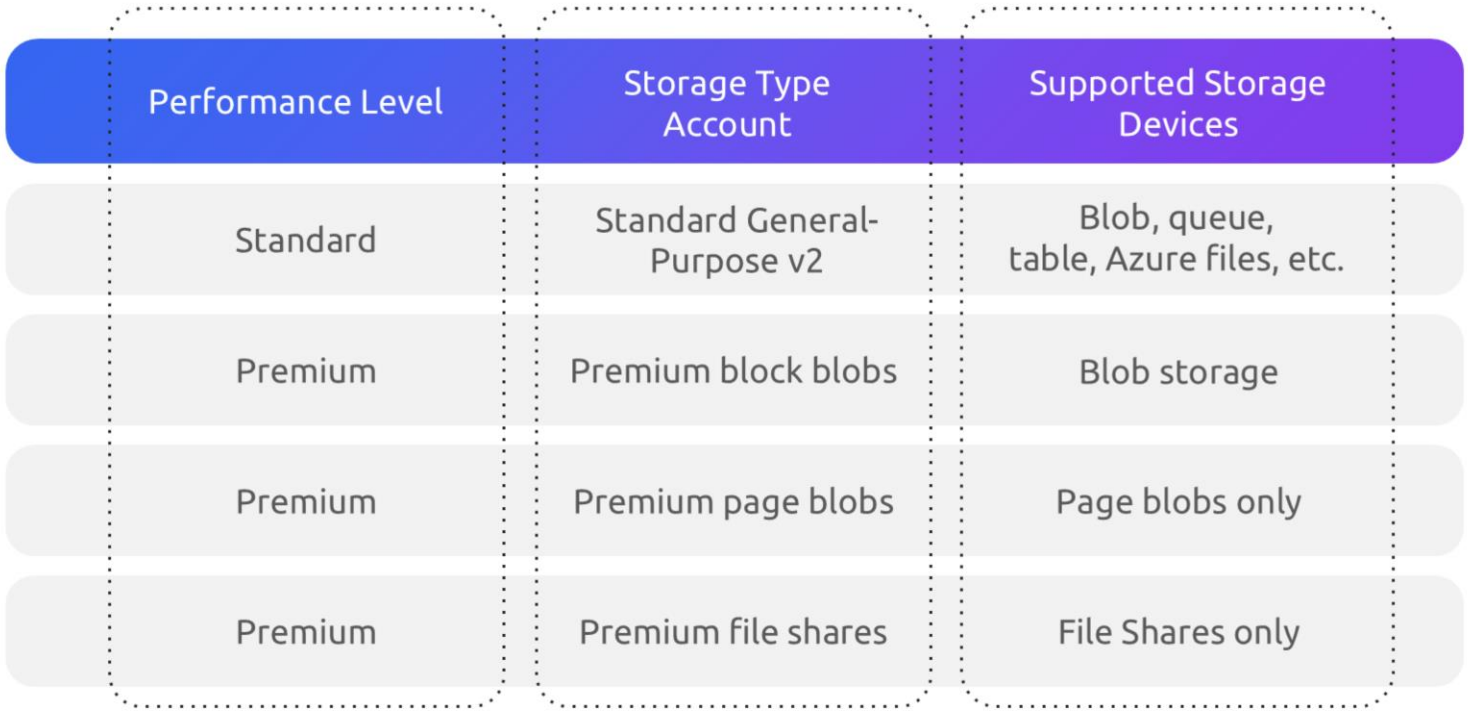
it's critical to understand how Azure ensures your data remains safe and accessible. Azure offers several storage redundancy options, designed to provide different levels of durability and availability based on your needs and budget.

Let's start with the Single Region options:
Locally-Redundant Storage (LRS): With LRS, your data is copied three times within a single datacenter in a single region. This means if one copy goes kaput, you've still got two backups ready to take over without breaking a sweat. Think of it as having triplicate keys to your house. And remember, opting for LRS isn't a choice; it's the minimum commitment when you

store data in Azure, ensuring you have triple redundancy baked into the cost.
Zone-Redundant Storage (ZRS): Next up is Zone Redundant Storage, ZRS for short, which is like LRS's more travel-savvy sibling. ZRS steps it up by spreading your data across three separate Availability Zones within the same region. Each zone is essentially a separate datacenter, meaning if one goes down (maybe due to a natural disaster), the other two have got your back. It’s an excellent choice for critical data that you can't afford to lose due to localized issues.
Geo-Redundant Storage (GRS) and Read-Access Geo-Redundant Storage (RA-GRS): With GRS, you get the benefits of LRS, but your data is also asynchronously replicated to a second region more than 500 KM or 300 miles away. For added read access in the secondary location, RA-GRS has got you covered. These are your go-to options when you need robust disaster recovery solutions.
Geo-Zone-Redundant Storage (GZRS): This combines the best of both worlds - the high availability of ZRS in the primary region with asynchronous replication to a secondary region for disaster recovery. It's like having an extra safety net. It’s suitable for workloads that require strong persistence and the ability to withstand regional outages.
As we move from LRS to GZRS, the cost increases – that's the trade-off for higher levels of durability and availability. The right choice depends on your application's needs and how critical it is to maintain data integrity in the face of outages and disasters.

When you're preparing for the AZ-204, it's not just about knowing what each of these acronyms stands for. It's about understanding the real-world scenarios where you'd apply each of these options and how they would affect both your wallet and your data's well-being. So, whether it's keeping a mobile app's data snappy with LRS or ensuring a global enterprise's data survives a disaster with GZRS, you'll need to know the ins and outs of Azure’s storage redundancy options.

Azure Storage – Performance Tiers



Performance Level	Storage Type Account	Supported Storage Devices
Standard	Standard General-Purpose v2	Blob, queue, table, Azure files, etc.
Premium	Premium block blobs	Blob storage
Premium	Premium page blobs	Page blobs only
Premium	Premium file shares	File Shares only

Azure Blob Storage offers different performance levels to cater to varying needs. The Standard performance level supports diverse storage options like blobs, queues, tables, Azure Files etc in the same storage account for a broad range of use cases. Premium performance levels are available for more specialized needs: Premium Block Blobs for high-throughput block storage, and Premium Page Blobs, tailored exclusively for page blobs which are often used for VHD files and as VM disks. These premium options are designed for scenarios requiring higher throughput and lower latency.

Also note that hierarchical namespace feature i.e ADLS Gen 2, which enables you to organize and manage files into a

directory structure, is a capability specific to the Standard performance tier only.

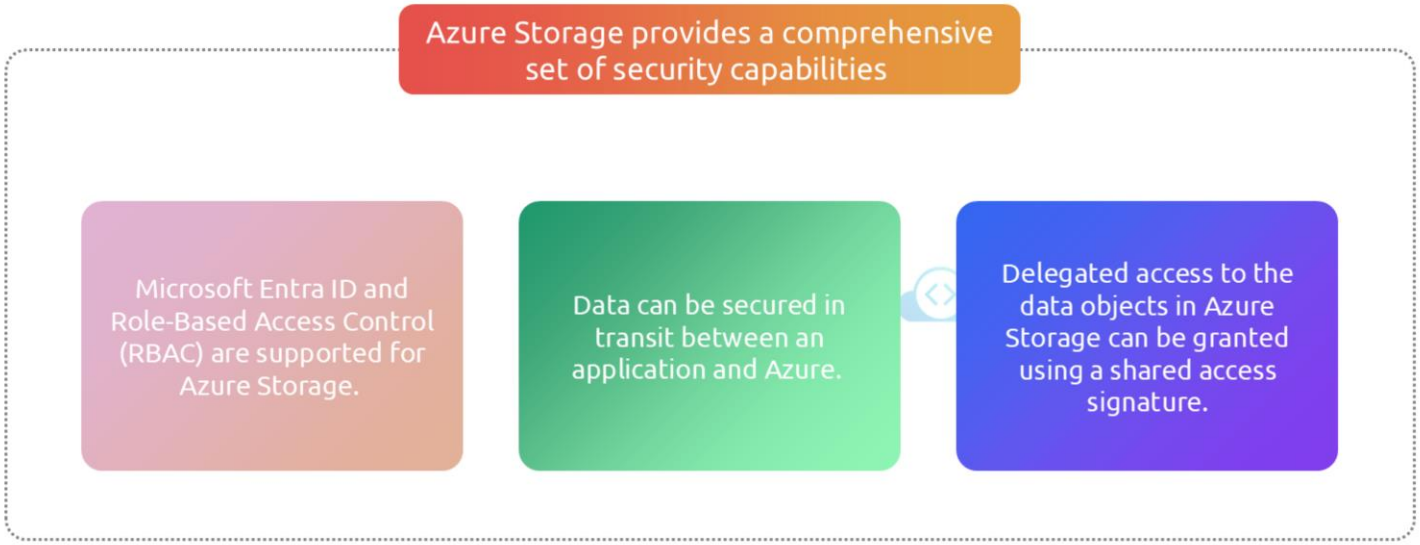
The segregation of supported storage types within Azure's Premium performance tier accounts is a technical decision to optimize performance. A Premium Fileshare account is specifically optimized for file share workloads, ensuring high IOPS and throughput, and it does not support hosting other types of blobs such as block or page blobs.

When you opt for Premium performance level Azure Blob Storage for block blobs, the high availability options are limited to Locally Redundant Storage (LRS) and Zone Redundant Storage (ZRS).

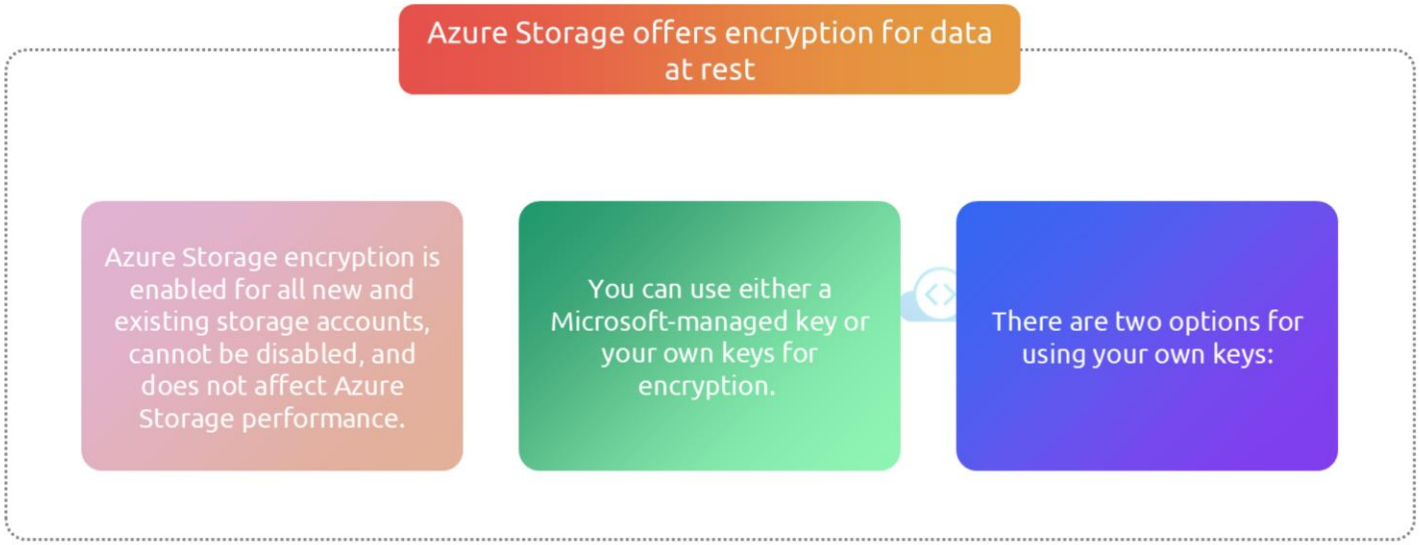
Lastly, Once you choose a performance tier for your Azure Storage account—be it Standard or Premium—you cannot switch between these performance tiers post-deployment. This is due to the underlying architectural differences in the way that data is stored and accessed within these tiers.

Azure Storage – Security Features

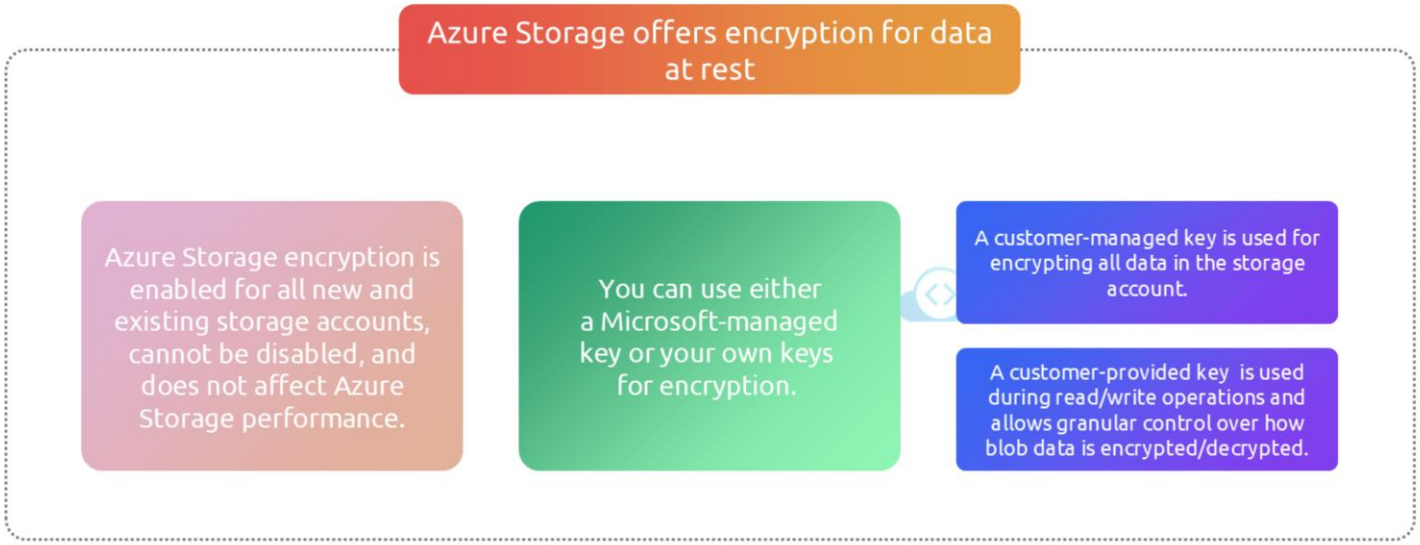
Azure Storage – Security Features



Azure Storage – Security Features

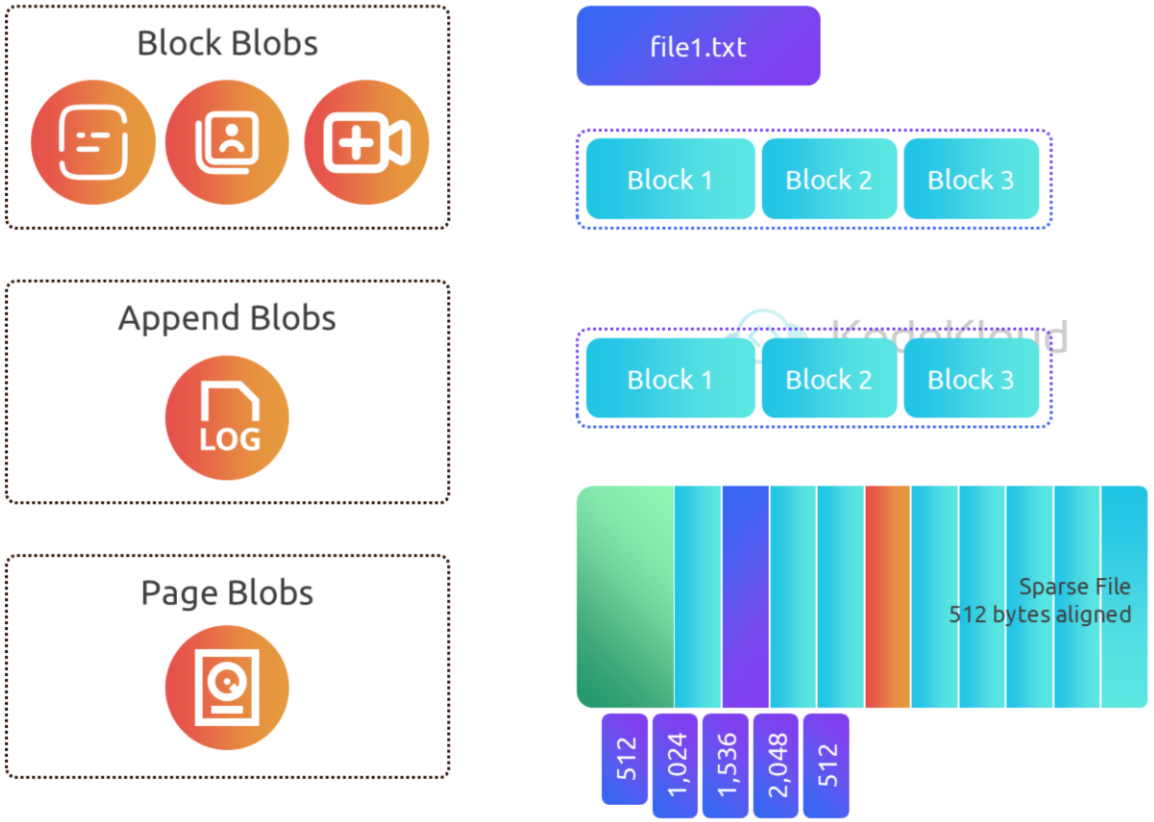


Azure Storage – Security Features



Azure Blob Storage

Blob Types



The other interesting thing about blobs is that there are 3 different kind of blobs.

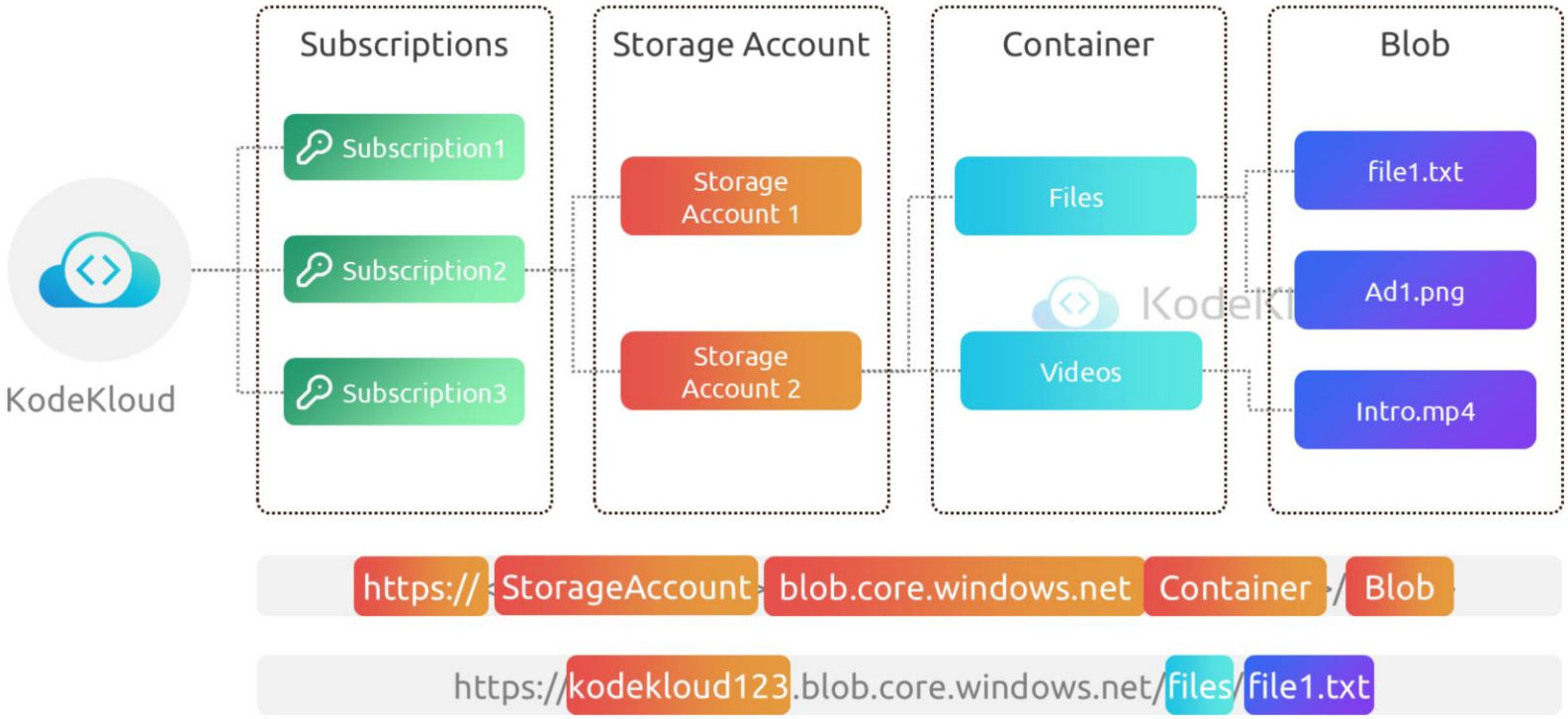
Block blobs – this is what you will be using most of the time. Ideal for storing text or binary data like files, images, and videos. Here there is an object called File1.txt, which has data chunks that you put using rest call
A block blob can include up to 50,000 blocks, allowing for a total size of approximately 200 TiB

Append blobs are a special kind of block blobs that allow you to append to the end of the blob and works well for any kind

of logging scenarios. Each append operation adds blocks to the end but does not overwrite existing data. i.e Updating or deleting of existing blocks is not supported.

Page Blobs are designed for frequent read/write operations and scenarios requiring random read/write access, like virtual hard disk files (VHDs) for Azure virtual machines. It is made up to 512-byte pages optimized for random read and write operations. There are not many use cases where you will use REST API to manage virtual hard disk files (VHDs), but azure internally used this type of blob to manage your disks asscoated with Virtual machines. The maximum size for a page blob is 8 TiB.

Blob Storage Hierarchy



When we think about Blob storage, think of a company, let's say, KodeKloud which can have many many subscriptions on Azure. Each subscription in turn can have many storage accounts. There are some soft and hard limits on the number of storage accounts per subscription per Azure region. You can go up to 5000 such storage account per subscription today and these numbers keep changing. This number used to be 250 about a year back.

Each storage account can have any many Containers as you like. Azure Blob Storage uses the concept of "containers" to organize and group blobs, similar to a folder in a file system. If you have used Amazon S3 and Google Cloud Storage, the

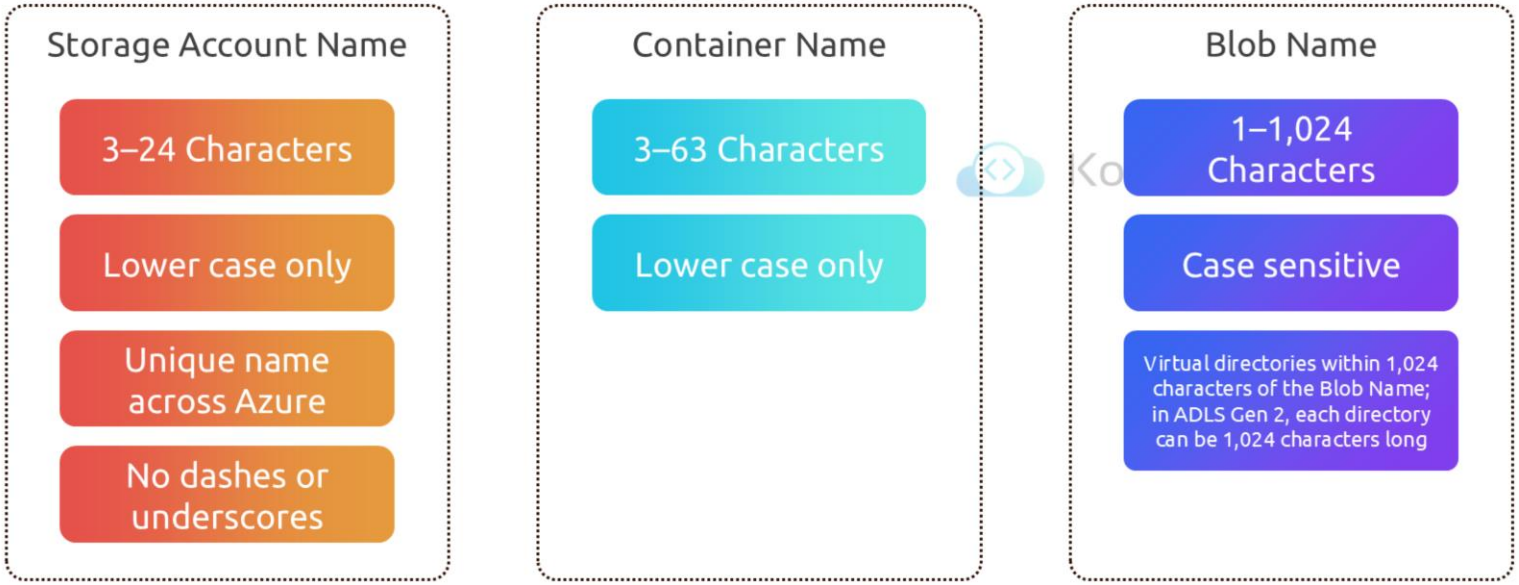
equivalent concept is called a "bucket," which serves as the top-level container for storing objects. Azure Blob Storage supports the concept of "virtual directories" inside containers. Although the underlying storage architecture is flat and doesn't support true nested containers or directories, virtual directories can be simulated using blob naming conventions. For example, naming a blob directory1/subdirectory1/file.txt gives the impression that file.txt is stored inside nested directories, but in reality, this is just a flat structure with a blob named directory1/subdirectory1/file.txt.

So how does the all of of this end up looking – its storageaccount name.blob.core.windows.net slash the container name slash the blob name. as I mentioned before the blob itself can have virtual directory names.

Blob Naming

https://<StorageAccount>.blob.core.windows.net/<Container>/<Blob>

https://kodekloud123.blob.core.windows.net/files/virtualdirectory1/file1.txt



Lets talk a little bit more about blob naming scheme.

Like I explained in the previous slide, you have your account name, followed by the container name, followed by the blob name which can include virtual directory name like this.

There is an important caveat here that I did not mention in the previous slide, When you turn on Hierarchical namespace in the storage account, this virtual directory converts into real folders and you can do operations like renaming them, moving

them around etc.
In ADLS Gen2, you can create a hierarchical file system with directories and subdirectories within containers, similar to a traditional file system. So it is important to know that difference between Blob API and ADLS Gen2 API.

There are some naming rules that need to be followed when creating a storage account and blobs inside it. I have listed some of the keys ones.

Managing Azure Blob Storage Lifecycle

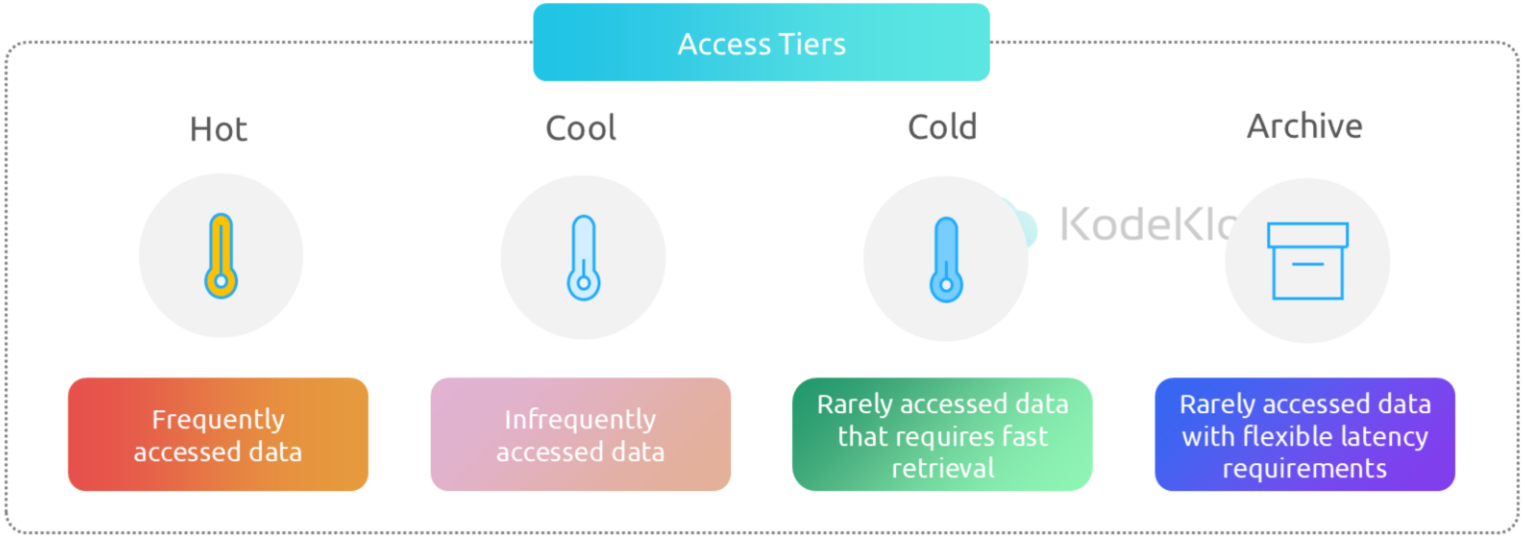
Introduction

01 Understand access tiers in Azure Storage

02 Learn Lifecycle Management policies in Azure Blob Storage to optimize storage cost

Storage Access Tiers

Storage Access Tiers



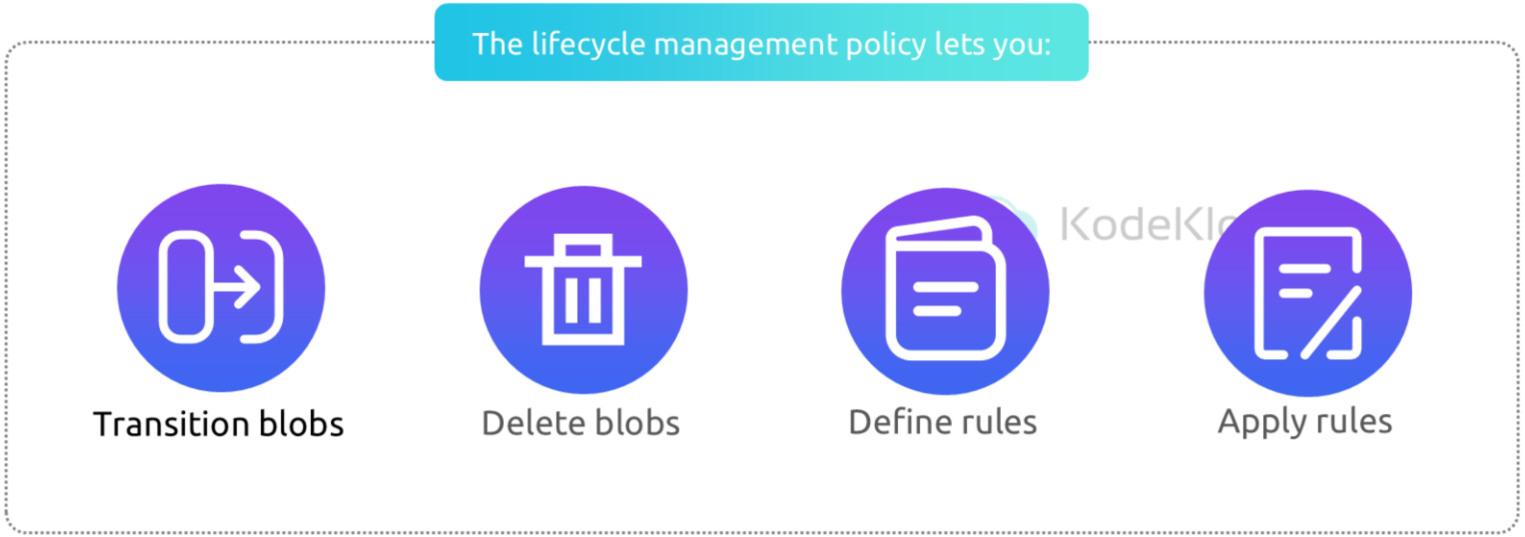
The Hot access tier has the lowest access cost, but the highest storage cost.
Moving through the tiers from Hot to Archive the storage costs decrease, but the access costs increase.

The following considerations apply to the different access tiers:
The access tier can be set on a blob during or after upload.
The archive access tier can only be set at the blob level.
Data in the archive access tier is stored offline. The archive tier offers the lowest storage costs but also the highest access

costs and latency.
The archive tier supports only LRS, GRS, and RA-GRS.
Data storage limits are set at the account level and not per access tier.

Azure Blob Lifecycle Management Policies

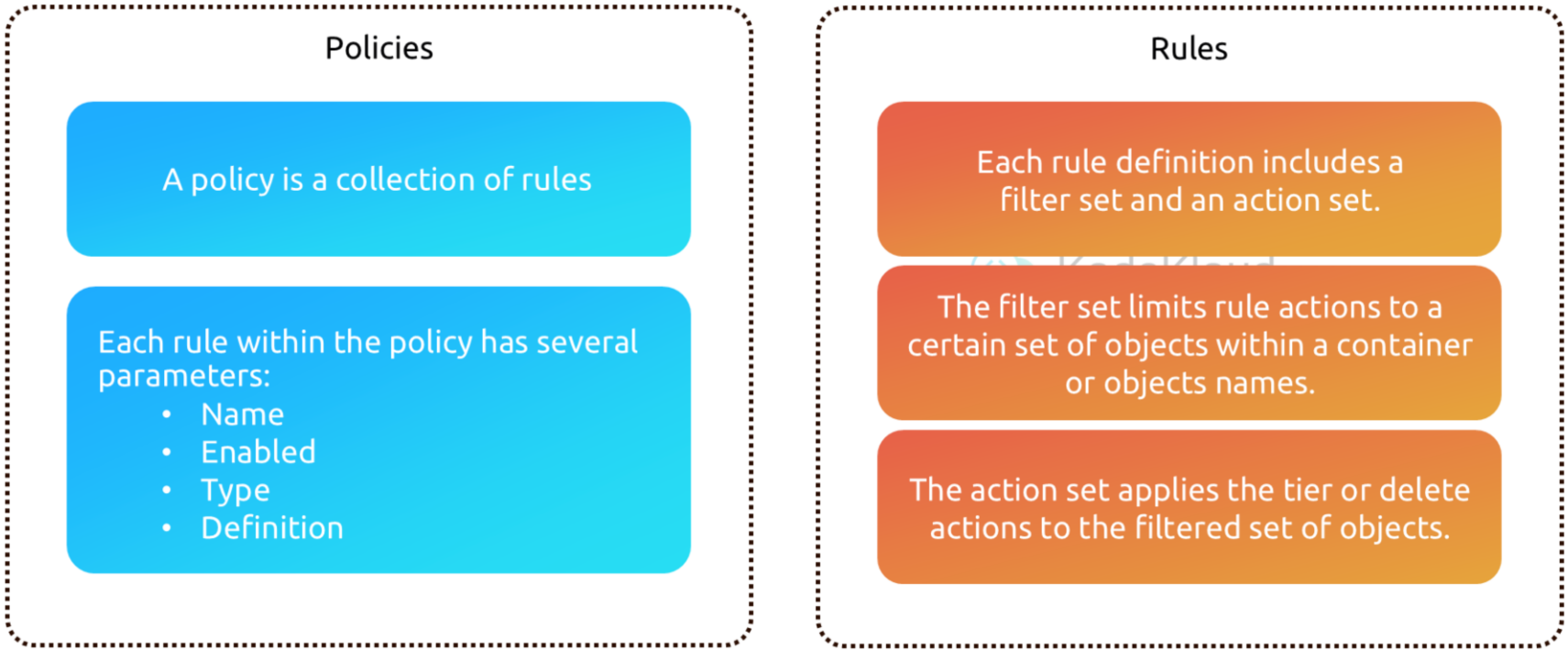
Azure Blob Storage Lifecycle



Azure Blob storage lifecycle management offers a rich, rule-based policy for General Purpose v2 and Blob storage accounts. Use the policy to transition your data to the appropriate access tiers or expire at the end of the data's lifecycle.
The lifecycle management policy lets you:

- Transition blobs to a cooler storage tier to optimize for performance and cost
- Delete blobs at the end of their lifecycles
- Define rules to be run once per day at the storage account level
- Apply rules to containers or a subset of blobs (using prefixes as filters)

Blob Storage Lifecycle Policies



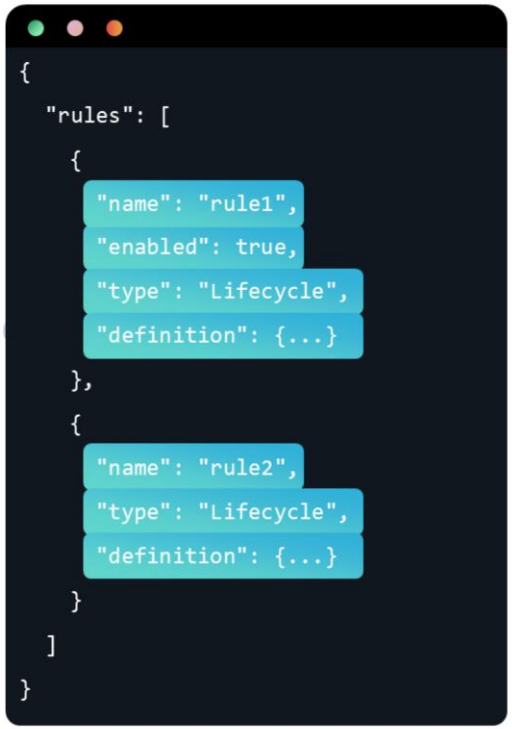
Rule filters
blobTypes
prefixMatch
blobIndexMatch

Rule actions
tierToCool

enableAutoTierToHotFromCool
tierToArchive
delete

Discovering Blob Storage Lifecycle Policies

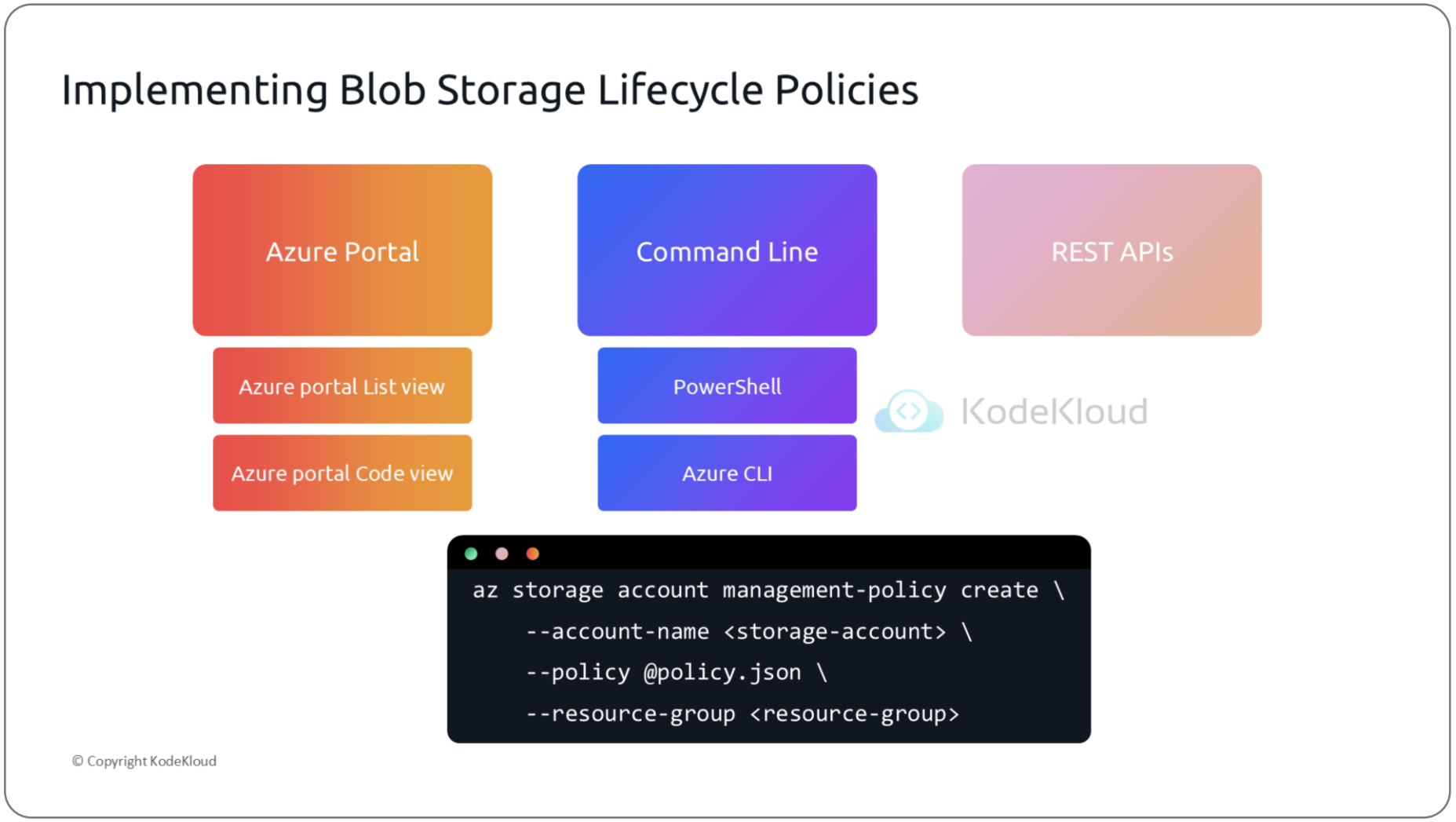
Parameter Name	Parameter Type	Required
name	String	True
enabled	Boolean	False
type	An enum value	True
definition	An object that defines the lifecycle rule	True



A policy must include at least one rule. You can define up to 100 rules in a policy.

A rule name can include up to 256 alphanumeric characters, and the name is case-sensitive. It must be unique within a policy.

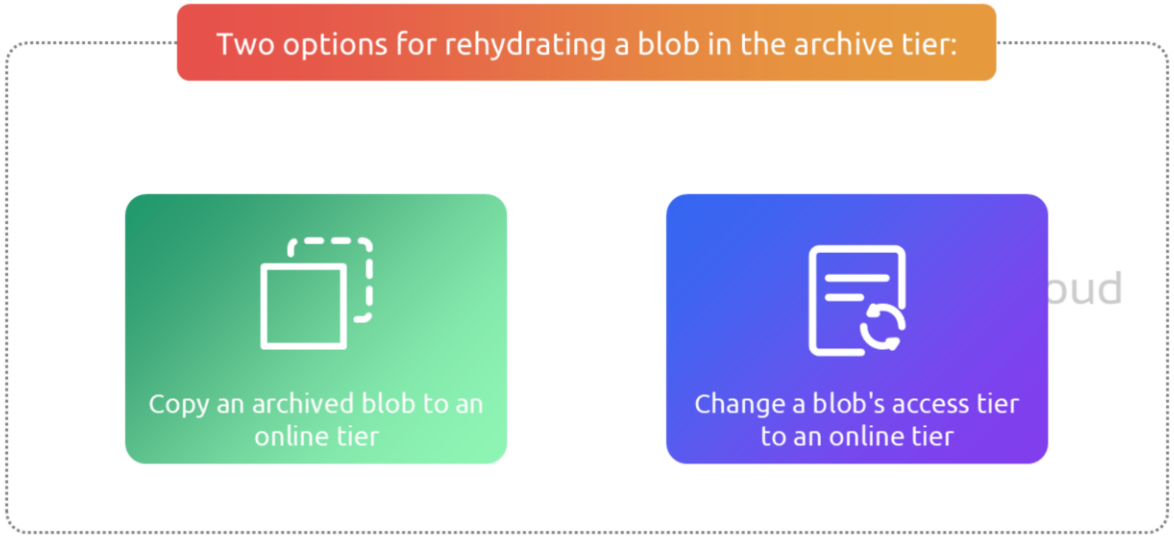
Each rule definition is made up of a filter set and an action set.



You can add, edit, or remove a policy by using any of the following methods:
Azure portal
Azure PowerShell
Azure CLI
REST APIs
Azure portal
There are two ways to add a policy through the Azure portal: Azure portal List view, and Azure portal Code view.

Rehydrating Data From Archive Tier

Rehydrating Blob Data From Archive Tier



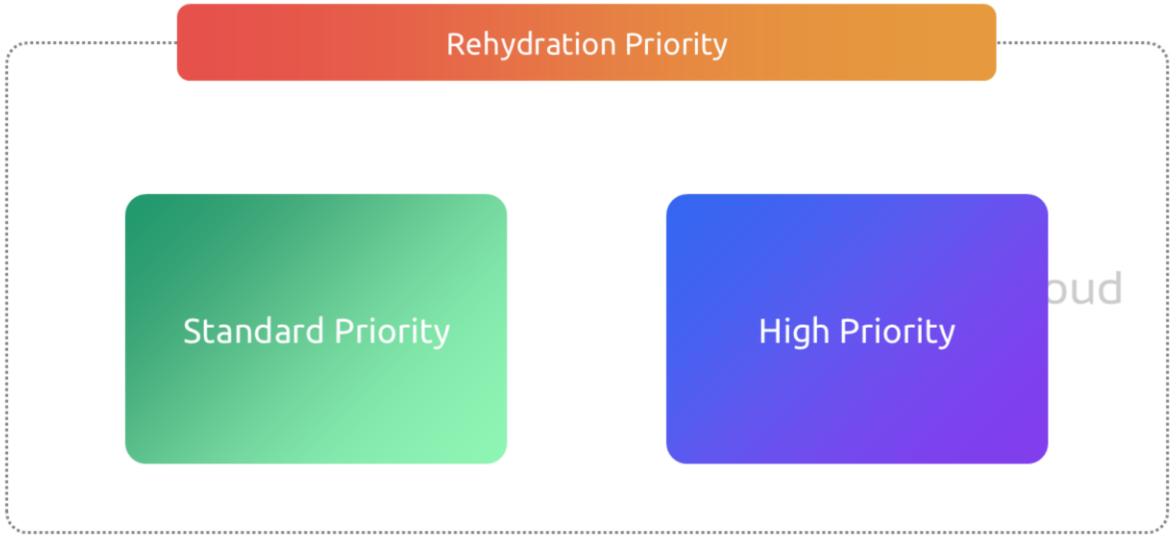
While a blob is in the archive access tier, it's considered to be offline and can't be read or modified. In order to read or modify data in an archived blob, you must first rehydrate the blob to an online tier, either the hot, cool, or cold tier. There are two options for rehydrating a blob that is stored in the archive tier:

Copy an archived blob to an online tier: You can rehydrate an archived blob by copying it to a new blob in the hot, cool, or cold tier with the Copy Blob or Copy Blob from URL operation. Microsoft recommends this option for most scenarios. Change a blob's access tier to an online tier: You can rehydrate an archived blob to hot or cool by changing its tier using

the Set Blob Tier operation.

Rehydrating a blob from the archive tier can take several hours to complete. Microsoft recommends rehydrating larger blobs for optimal performance. Rehydrating several small blobs concurrently may require additional time.

Rehydrating Blob Data From Archive Tier



Rehydration priority
When you rehydrate a blob, you can set the priority for the rehydration operation via the optional x-ms-rehydrate-priority header on a Set Blob Tier or Copy Blob/Copy Blob From URL operation. Rehydration priority options include:
Standard priority: The rehydration request will be processed in the order it was received and may take up to 15 hours.
High priority: The rehydration request will be prioritized over standard priority requests and may complete in under one hour for objects under 10 GB in size.

To check the rehydration priority while the rehydration operation is underway, call Get Blob Properties to return the value of the x-ms-rehydrate-priority header. The rehydration priority property returns either Standard or High.

Working With Azure Blob Storage

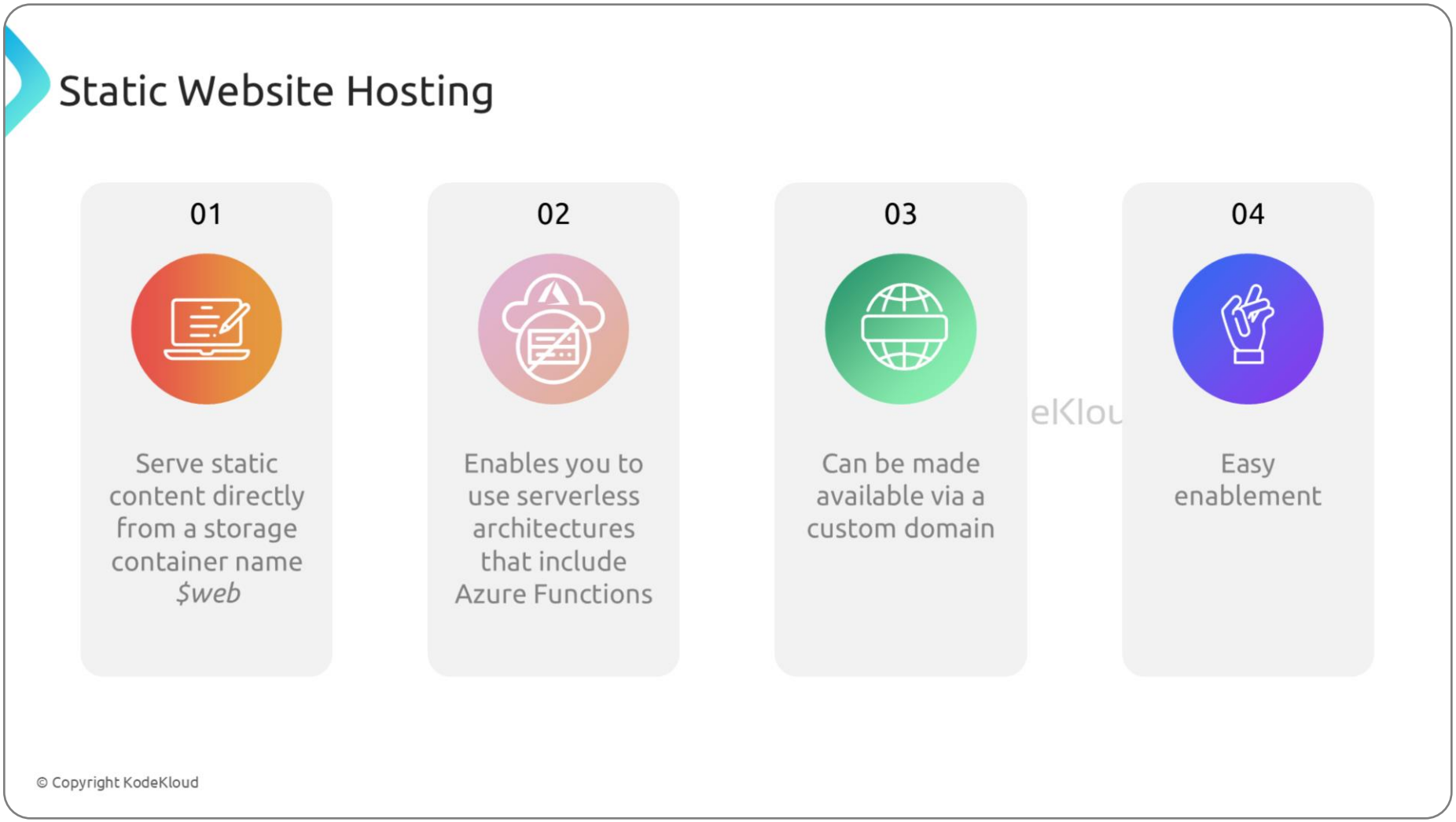
Introduction

01 Develop static websites using Azure Blob Storage

02 Use .NET Azure Storage client to work with Azure Blob storage

Developing Static Websites

Static Website Hosting



Lets talk about Static Website Hosting on Azure. This feature is a game-changer for hosting static content — think HTML, CSS, JavaScript, and images — that doesn't require server-side processing. You should be aware of some of the features and limitation of this service:

Serve Static Content: Azure Blob Storage allows you to serve static content right from a storage container named $web. This is a special container that's used exclusively for hosting static websites.
Serverless Architectures: It perfectly integrates with serverless architectures, including Azure Functions. This means you can

build dynamic applications without managing infrastructure, as Azure Functions can respond to events in your Blob Storage and execute code to render dynamic content.
Custom Domain: You can map a custom domain to your Blob Storage endpoint, giving your static website a professional look and feel.
Easy Enablement: To enable static website hosting, simply go to the 'Static website' section in the Azure Storage account navigation bar and select 'Enabled'. Enter the name of your index and error document, and you're good to go!

Static Website Hosting – Limitations

01

Inability to configure headers as part of the static website feature

02

AuthN and AuthZ not supported

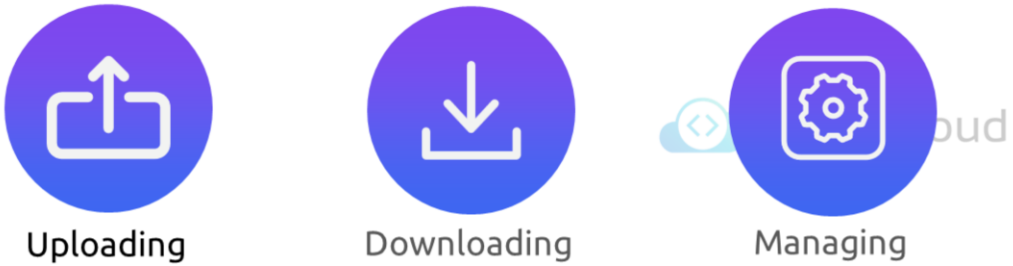
Limitations:
No Header Configuration: Currently, there is no built-in feature to configure HTTP headers for your static website. This means you can't set security headers or control caching directly through the static website feature.
Authentication and Authorization: The standard authentication (AuthN) and authorization (AuthZ) mechanisms are not supported directly in static website hosting. If your application requires user authentication, you'll need to implement it at the application level or use other Azure services like Azure Active Directory B2C.

Developing Using Azure Blob Storage Client Library

Azure Blob Storage Client Library – Overview

Class	Purpose
BlobClient	Facilitates the manipulation of Azure Storage blobs.
BlobClientOptions	Defines configuration settings for connecting to Azure Blob Storage.
BlobContainerClient	Manages Azure Storage containers and the blobs within them.
BlobServiceClient	Handles Azure Storage service resources and blob containers, providing the top-level namespace for the service.
BlobUriBuilder	Enables modification of a URI to direct it to various Azure Storage resources, such as an account, container, or blob.

Creating a Client Object



Passing the Endpoint URI: When creating a client object in an application, an endpoint URI must be provided.
Constructing the Endpoint URI: The endpoint string can either be manually constructed, as illustrated in the example.
Retrieving the Endpoint at Runtime: The endpoint can also be queried at runtime using the Azure Storage management library.
Authentication with DefaultAzureCredential: The code example demonstrates using DefaultAzureCredential to handle authentication.

Creating a Client Object

- Passing the Endpoint URI
- Constructing the Endpoint URI
- Retrieving the Endpoint at Runtime
- Authentication with DefaultAzureCredential

```
using Azure.Identity;
using Azure.Storage.Blobs;

public BlobServiceClient GetBlobServiceClient(string
accountName)
{
BlobServiceClient client = new(
new
Uri($"https://{accountName}.blob.core.windows.net"),
new DefaultAzureCredential());

return client;
}
```


Passing the Endpoint URI: When creating a client object in an application, an endpoint URI must be provided.
Constructing the Endpoint URI: The endpoint string can either be manually constructed, as illustrated in the example.
Retrieving the Endpoint at Runtime: The endpoint can also be queried at runtime using the Azure Storage management library.
Authentication with DefaultAzureCredential: The code example demonstrates using DefaultAzureCredential to handle authentication.

Managing Container Properties and Metadata

Managing Container Properties and Metadata Using .NET

Support for system properties and metadata

Retrieve container properties

Set metadata

```
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Models;
using System;
using System.Threading.Tasks;

public class BlobContainerExample
{
      public static async Task
      SetMetadataAndGetPropertiesAsync(BlobContainerClient
      containerClient)
      {
      // Set metadata
      await containerClient.SetMetadataAsync(new Dictionary<string,
      string> { { "owner", "admin" }, { "environment", "production" } });

      // Get properties
      BlobContainerProperties properties = await
      containerClient.GetPropertiesAsync();
      Console.WriteLine($"Container: {containerClient.Name}, Last
      Modified: {properties.LastModified}, Access:
      {properties.PublicAccess}");
      }
}
```


Support for System Properties and Metadata: Blob containers support both system properties and user-defined metadata, in addition to the data they contain.
Retrieve Container Properties: Use the GetProperties or GetPropertiesAsync methods to retrieve container properties.
Set Metadata: Apply the SetMetadata or SetMetadataAsync methods to set user-defined metadata.

Managing Container Properties and Metadata Using .NET

Metadata Header format

x-ms-meta-name:string-value

Retrieve properties and metadata

GET/HEAD https://myaccount.blob.core.windows.net/mycontainer?restype=container

GET/HEAD https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=metadata

Set properties and metadata

PUT https://myaccount.blob.core.windows.net/mycontainer?restype=container

PUT https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=metadata

```
$accountName = "yourstorageaccount"
$containerName = "yourcontainer"
$accountKey = "youraccountkey"
$baseUri = "https://$accountName.blob.core.windows.net/$containerName"

# Set Metadata
$headers = @{
"x-ms-version" = "2021-04-10"
"x-ms-date" = (Get-Date).ToString("R")
"x-ms-meta-owner" = "${$admin}"
"x-ms-meta-environment" = "${$production}"
}
Invoke-RestMethod -Uri "$baseUri?restype=container" -Method Put -Headers $headers
-Header ("Authorization", "SharedKey
${accountName}:$([Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes($accoun
tKey)))")

# Get Properties
$response = Invoke-RestMethod -Uri "$baseUri?restype=container&comp=metadata" -
Method Get -Headers $headers
Write-Host "Owner: $($response.Headers.'x-ms-meta-owner'), Env:
$($response.Headers.'x-ms-meta-environment'), Last Modified:
$($response.Headers.'Last-Modified')"
```


Metadata Header Format: x-ms-meta-name:string-value
Retrieve Properties and Metadata:
URI syntax to retrieve properties and metadata from containers and blobs:

GET/HEAD https://myaccount.blob.core.windows.net/mycontainer?restype=container

GET/HEAD https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=metadata

Set Properties and Metadata:
URI syntax to set properties and metadata for containers and blobs:

PUT https://myaccount.blob.core.windows.net/mycontainer?restype=container
PUT https://myaccount.blob.core.windows.net/mycontainer/myblob?comp=metadata

KodeKloud

Visit www.kodekloud.com to learn more.