🧾 Azure Blob Storage Policy & Data Lifecycle Management – Overview

📘 Table of Contents

🔹 1. Introduction to Blob Storage Management

Azure Blob Storage is designed for storing large amounts of unstructured data. Managing storage costs and data availability over time is essential for cloud-scale systems. Azure provides **access policies** and **automated lifecycle management rules** to:

- Enforce data retention policies
- Automate blob tiering
- Automatically delete stale data
- Control access securely

🔐 2. Blob Access Policies

✅ Access Tiers

Azure Blob Storage supports multiple **tiers** to optimize cost based on how often you access data:

Tier	Use Case	Minimum Retention
Hot	Frequently accessed data	None
Cool	Infrequently accessed (≥30 days)	30 days
Archive	Rarely accessed (≥180 days)	180 days

You can move blobs between these tiers **manually** or **automatically** using **lifecycle management**.

✅ Shared Access Signatures (SAS)

A **SAS** allows restricted access to blobs without exposing account keys. You can:

- Grant temporary access to a client (e.g., for upload/download)

- Define expiration time, IP range, and permissions

✅ Stored Access Policies

- SAS tokens can be linked to a **stored access policy** on a container.
- Stored policies enable centralized control over shared access tokens.

Example use: Revoke multiple SAS tokens by deleting/modifying the associated policy.

✅ Immutable Blob Policies

Used for regulatory compliance and legal scenarios.

1. **Time-Based Retention Policy:** Prevents deletion/modification for a fixed period.
2. **Legal Hold:** Locks data until manually released.

Features:

- Cannot be overridden or deleted during retention period
- Common in financial and healthcare industries

🔁 3. Data Lifecycle Management

✅ What Is Lifecycle Management?

Lifecycle Management in Azure Blob Storage enables you to define **rules** to automate the transition of data between access tiers or to delete it based on age or conditions.

It helps reduce costs and enforce data governance by:

- Moving stale data to cheaper tiers
- Cleaning up temporary or expired data

- Enforcing retention limits

✅ Rule Structure

Each rule consists of:

- **Scope**: Applies to a container or the whole storage account
- **Filters**: Prefix match or blob type (block, append, etc.)
- **Actions**: Transition to another tier, delete, etc.
- **Conditions**: Age of blob, last modified date, etc.

✅ Common Lifecycle Management Actions

Action	Description
Move to Cool Tier	After X days of last modification
Move to Archive Tier	After Y days of last modification
Delete Blob	After Z days of creation/modification
Delete blob snapshots	After snapshot is older than N days
Tier transition for versions	Manage previous blob versions independently

---

✅ Example Scenarios

Scenario	Lifecycle Rule
Move logs to archive after 180 days	`If blob is modified >180 days ago → Move to Archive`
Delete temp files after 30 days	`If blob name starts with /temp/ and >30 days old → Delete`

Retain compliance data for 5 years	Use **Immutable Blob Policy**, not lifecycle rule

---

🛠️ 4. How to Configure Lifecycle Management

🔹 Via Azure Portal

1. Go to your **Storage Account**
2. Click **“Data Management” > “Lifecycle Management”**
3. Click **+ Add rule**
4. Define:
    - **Rule name**
    - **Scope** (entire account or specific containers)
    - **Filter** (e.g., prefix, blob type)
    - **Action** (move to cool/archive/delete)
    - **Conditions** (days since last modified, creation date)
5. Click **Save**

🔹 Via ARM Template or Bicep

You can define lifecycle rules as Infrastructure-as-Code using ARM/Bicep templates.

```
{
  "rules": [
    {
      "enabled": true,
      "name": "archive-old-logs",
      "type": "Lifecycle",
      "definition": {
        "filters": {
```

```
        "blobTypes": ["blockBlob"],
        "prefixMatch": ["logs/"]
      },
      "actions": {
        "baseBlob": {
          "tierToArchive": {
            "daysAfterModificationGreaterThan": 180
          }
        }
      }
    }
  }
]
}
```

🧠 5. Best Practices

- **Name rules clearly** (e.g., `delete-temp-30days`)
- **Use filters to avoid broad actions**
- **Test rules in a dev storage account first**
- Avoid applying **delete rules** without careful review
- Combine **Immutable Policies** with **Lifecycle Rules** for compliance & cost savings
- Monitor with **Azure Monitor / Storage Logs**

📌 6. Summary

Feature	Description
Access Policies	SAS, stored access policies, immutable policies for secure and compliant access
Lifecycle Management	Automates cost optimization and data retention

Tiers	Hot, Cool, Archive – each for different usage frequency
Portal & Code Support	Easily configurable through UI or templates

Lifecycle management is **crucial** for any scalable cloud system — it helps manage cost, ensure data hygiene, and maintain compliance.

🔗 7. References & Resources

- Azure Blob Storage Lifecycle Management Docs
- Immutable Blob Storage Overview
- Azure Storage Pricing
- Azure CLI – Storage Rules