**Object Replication in Azure (Azure Blob Storage)**

**Object Replication** in Azure allows **automatic, asynchronous copying of blobs** from one **storage account** (source) to another **(destination)** within the **same region** or across **different regions**.

**✅ Why Use Object Replication?**

| **Benefit** | **Description** |
| --- | --- |
| 🔄 **Disaster Recovery** | Maintain copies of data in another region |
| 📊 **Data Aggregation** | Collect data from different accounts to one |
| 🔒 **Compliance** | Keep immutable copies with Object Lock |
| ⚙️ **Automation** | No manual copying or scripting needed |

**⚙️ How It Works**

* **Source**: Storage account where blobs are written
* **Destination**: Storage account where blobs are automatically copied
* **Supported Only for**: **Block blobs** in **general-purpose v2** (GPv2) accounts

**🪜 Steps to Enable Object Replication**

**1. Create Two Storage Accounts**

* Both must be **GPv2** and in the same Azure AD tenant
* Can be in **same or different regions**

**2. Enable Versioning on Both**

* Go to each storage account → **Data Protection** → **Enable versioning**

**3. Create Object Replication Policy**

* Go to **Source Storage Account**
* Select **Object Replication** → **Add Policy**
* Choose:
  + **Destination Storage Account**
  + Source and destination **container pair**
  + (Optional) Filter by **prefix**

**4. Policy is Created**

* Azure assigns a **policy ID**
* Replication begins **asynchronously**
* Only **new blobs or changes** are replicated

**🔒 Important Notes**

| **Rule** | **Detail** |
| --- | --- |
| 📌 Only **block blobs** are supported |  |
| 🆕 Only **new changes** are replicated |  |
| 🕒 Replication is **asynchronous** (not real-time) |  |
| 🔁 Replication is **one-way** (source → destination) |  |
| ❌ Doesn't copy **snapshots, metadata**, or **existing blobs** |  |
| ✅ Supports **Object Lock** at destination (for WORM compliance) |  |

**📊 Monitoring**

* Use **Azure Monitor** and **Blob Inventory** to track replication
* **Replication status** can be checked on each blob’s properties

**💡 Use Cases**

* 🔄 **Cross-region replication for DR**
* 📥 **Aggregating logs from multiple locations**
* 🧾 **Creating audit/compliance archives**
* 🌍 **Syncing edge-generated data to central location**

**🧠 Summary**

| **Feature** | **Azure Object Replication** |
| --- | --- |
| Type | Asynchronous, automatic block blob replication |
| Source/Destination | Must be GPv2 accounts |
| Triggers | New blob creation, overwrite |
| Monitoring | Azure Monitor, blob properties |
| Typical Use Cases | DR, backup, analytics, compliance |

**Static Website Hosting using Azure Blob Storage**

**🔹 What it is:**

Azure Blob Storage allows you to **host static websites directly** from a storage account by enabling the **static website feature**.

**🎯 Use Case:**

* Portfolio websites
* Documentation sites
* SPAs (Single Page Applications) like React, Angular

**🚀 Steps to Host Static Website in Blob Storage:**

1. **Create a Storage Account**
   * In the Azure Portal → Create → Storage Account (Standard, GPv2)
2. **Enable Static Website Hosting**
   * Go to your Storage Account → **Data Management** → **Static website**
   * Click “**Enable**”
   * Enter:
     + **Index document name**: e.g., index.html
     + **Error document path** (optional): e.g., 404.html
3. **Upload Website Files**
   * Go to **Containers** → A container called $web will be created
   * Upload your HTML/CSS/JS files to this container
4. **Access the Website**
   * A **public URL** will be provided like:

cpp

https://<yourstorageaccount>.z13.web.core.windows.net

**💡 Example:**

You upload:

* index.html
* style.css
* app.js

Your website is now live at:

arduino

https://mystorageaccount.z13.web.core.windows.net

**🔐 Access Control**

* The $web container is **public by default** when hosting a static website.
* Use **Azure CDN or Front Door** to add caching, custom domains, or HTTPS.

**What is IOPS?**

**IOPS (Input/Output Operations Per Second)**  
It measures how many **read/write operations** a storage disk can handle **per second**.

**🔹 Key Points:**

* Focuses on the **number** of I/O operations.
* Smaller files = higher IOPS usage.
* Ideal for workloads like **databases**, **transaction-heavy apps**, etc.

**💡 Example:**

If a disk supports **6,000 IOPS**, it can perform 6,000 reads/writes every second.

**🚀 What is Throughput?**

**Throughput** measures the **amount of data transferred** per second, typically in **MB/s** or **GB/s**.

**🔹 Key Points:**

* Focuses on **volume of data** moved.
* Larger files = more throughput.
* Important for **big data**, **video streaming**, **backups**, etc.

**💡 Example:**

If a disk has a **throughput of 250 MB/s**, it can transfer 250 megabytes every second.

**🔄 IOPS vs Throughput – Key Differences**

| **Metric** | **IOPS** | **Throughput** |
| --- | --- | --- |
| Measures | Number of read/write operations | Amount of data transferred |
| Unit | Operations per second | MB/s or GB/s |
| Best For | Databases, OLTP systems | Big data, media, backups |
| Affected By | Disk type, block size, latency | File size, IOPS, block size |

**🎯 In Azure (Example):**

| **Disk Type** | **Max IOPS** | **Max Throughput** |
| --- | --- | --- |
| Standard HDD | ~500 | ~60 MB/s |
| Standard SSD | ~6,000 | ~750 MB/s |
| Premium SSD | ~20,000+ | ~900 MB/s |
| Ultra Disk | ~160,000 | ~2,000 MB/s |

**🔍 Quick Analogy:**

Imagine you're filling buckets with water:

* **IOPS** = How many buckets you fill per second
* **Throughput** = How much **total water** (data) you transfer per second

So, 1,000 small buckets vs. 5 large buckets can give different IOPS and throughput values!

**🧠 Summary Table:**

| **Feature** | **IOPS** | **Throughput** |
| --- | --- | --- |
| Measures | Number of operations per second | Amount of data per second |
| Used In | Random small I/O, databases | Large sequential files |
| Unit | ops/sec | MB/s, GB/s |
| Related To | Disk latency, operation size | Data size, IOPS |

**☁️ Azure Availability Set – Full Explanation**

An **Azure Availability Set** is a **high-availability** feature that helps keep your virtual machines (VMs) **online during hardware failures or maintenance** by **distributing them across isolated hardware**.

**🛡️ Why Availability Sets?**

If you run **multiple VMs** for a single application (e.g., web servers or database replicas), placing them in an **Availability Set** ensures that:

* Not all VMs go down at once during **planned maintenance**.
* Not all VMs crash due to the **same hardware failure**.

**🧱 How It Works: 2 Key Concepts**

**1. Fault Domain (FD) – Hardware Isolation**

Represents a **physical rack** in a datacenter (with its own power, cooling, and networking).

* Azure distributes your VMs across **multiple fault domains**.
* So if a **power or hardware failure** hits one rack, others remain unaffected.

**Example**: If FD = 2, your VMs are placed on two different physical racks.

**2. Update Domain (UD) – Maintenance Isolation**

Represents a **group of VMs** that are **updated/rebooted together** during Azure maintenance.

* VMs in different UDs will **not be rebooted at the same time**.
* Azure applies **OS patches or updates** to one UD at a time.

**Example**: If UD = 5, the VMs are divided into 5 groups; updates are done one group at a time.

**🖼️ Diagram:**

pgsql

Availability Set

├── Fault Domain 1

│ ├── Update Domain 1

│ └── Update Domain 2

└── Fault Domain 2

├── Update Domain 3

└── Update Domain 4

**🧠 Key Properties**

| **Property** | **Value** |
| --- | --- |
| Max Fault Domains | 2 or 3 |
| Max Update Domains | Up to 20 |
| Scope | **Single region** and **single datacenter** |
| Cost | Availability Set itself is **free**, only VM cost applies |

**🔄 Use Case Example**

You have a **load-balanced app** with 2 VMs:

✅ If placed in an **Availability Set**, each VM goes into:

* A different **Fault Domain** (e.g., different rack)
* A different **Update Domain** (e.g., rebooted at different times)

❌ If not placed in an Availability Set:

* Both VMs could be **in the same rack**, and a failure or update would take **both down**.

**✅ Benefits of Using Availability Sets**

| **Feature** | **Benefit** |
| --- | --- |
| Hardware Isolation | Survives rack failures |
| Maintenance Isolation | Minimizes downtime during updates |
| SLA Guarantee | **99.95% uptime** when used properly |
| Cost-effective Redundancy | No extra cost for the set itself |

**🚫 Limitations**

* Works **only within a single region**
* Doesn’t protect against **region-wide outages**
* Must assign VMs at **creation time**, cannot move later