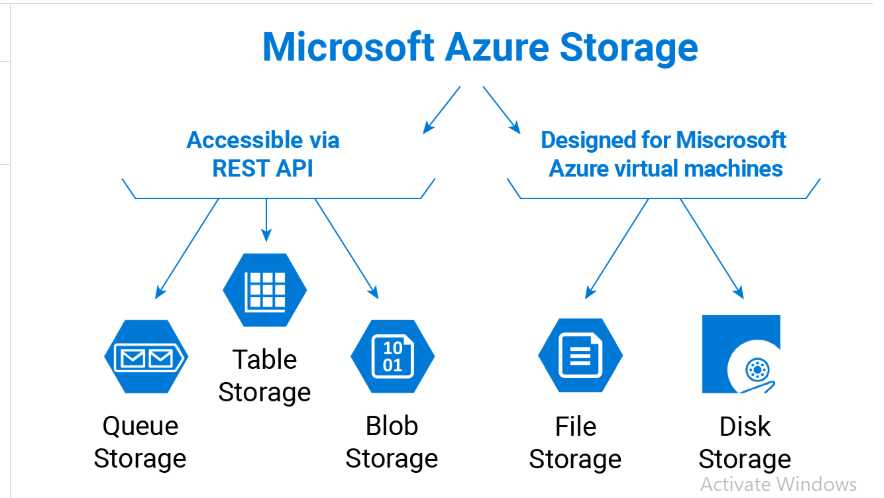
** Azure Storage**

**Azure Blob Storage — object store for files, images, logs, backups**

Blob Storage is Azure’s massively scalable object store designed for unstructured data — any binary or text file: images, videos, backups, logs, parquet/ORC files for analytics, etc.

**Data model & types**

* **Block blobs** — optimized for streaming and storing large files (most common). Blocks are uploaded and committed to form a blob. Large files are built from up to 50,000 blocks.
* **Append blobs** — optimized for append-only scenarios (log files).
* **Page blobs** — optimized for random read/write (used by VHDs for Azure VMs).

**Access patterns & tiers**

* Blobs can be accessed via REST, SDKs, or mounted via Data Lake Gen2 features (hierarchical namespace).
* Access tiers: **Hot, Cool, Archive** (and Premium options) — pick the tier based on frequency of access and cost profile; lifecycle policies can move blobs between tiers automatically. Note: some storage account types (e.g., premium block blob accounts) have tiering limitations. ([Microsoft Learn](https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview?utm_source=chatgpt.com))

**Key scale/limits (important)**

* Maximum size of a **block blob**: up to ~**190.7 TiB** (50,000 blocks × 4,000 MiB per block in current service limits). Page blobs and append-blob limits differ. Blob name length and other limits apply. ([Microsoft Learn](https://learn.microsoft.com/en-us/azure/storage/blobs/scalability-targets?utm_source=chatgpt.com))

**Availability & redundancy**

* Supports LRS, ZRS, GRS, RA-GRS, GZRS — choose based on durability and geographic resiliency requirements. ([Microsoft Learn](https://learn.microsoft.com/en-us/azure/storage/common/storage-redundancy?utm_source=chatgpt.com))

**When to use**

* Store media, backups, app logs, data lake files (analytics), container images, large binary objects, or “claim-check” pattern for messaging (store payload in blob, message with pointer in queue).

**Best practices**

* Use lifecycle management to tier/archive cold data. ([Microsoft Learn](https://learn.microsoft.com/en-us/azure/storage/blobs/lifecycle-management-policy-access-tiers?utm_source=chatgpt.com))
* Use **SAS tokens** for delegated access and storage encryption (server-side encryption enabled by default).
* Use versioning and snapshots for recoverability.
* For analytics, use block blobs with Parquet/ORC and consider hierarchical namespace (Data Lake Gen2) when working with big data.

**Azure File Storage (Azure Files) — managed SMB/NFS file shares**

**What it is (short):** Azure Files provides fully managed file shares in the cloud that use standard SMB (and in some SKUs NFS) protocols so you can **mount** them from Windows, Linux, or macOS — or use them as network file shares for lift-and-shift applications.

**Modes & features**

* **SMB** support (typical Windows file share scenarios). Some tiers support **NFS** (NFS v3) for Linux workloads.
* **Azure File Sync** lets you cache Azure file shares on on-prem Windows Servers so users/apps keep the same access patterns while the cloud becomes central storage.

**Key scale/limits (important & slightly SKU-dependent)**

* Max **file share** size depends on account/sku: modern provisioned v2 shares support very large sizes (documentation shows **up to 256 TiB** in current scale-target tables), while older pay-as-you-go large-file-share limits are commonly referenced around **100 TiB** depending on the configuration/legacy account. Check the specific SKU when provisioning.
* Maximum single file size and IOPS/throughput are SKU-dependent (premium vs standard), so choose based on performance needs.

**When to use**

* Replace on-prem file servers (lift & shift), host shared application data, home directories or user profiles, containerized apps that need an SMB mount, or share files between VMs.

**Best practices**

* Use **Azure File Sync** to keep hot data cached on-prem and tier less-used files to the cloud.
* Choose premium SSD-backed shares for low-latency/high-IOPS workloads.
* Protect shares with snapshots and RBAC; use network rules and private endpoints for security.

**Table Storage — simple, cost-effective NoSQL key/value store**

**What it is (short):** Azure Table Storage is a **schemaless, NoSQL key–attribute store** for structured, non-relational data (telemetry, device metadata, user profiles, small JSON-like entities). It’s cheap and very scalable.

**Data model**

* Each **entity** (row) contains properties (columns). Each entity has a **PartitionKey** and **RowKey** which together form a unique primary key; PartitionKey is used to scale and localize operations.

**Key scale/limits (important)**

* **Maximum size per entity** (sum of all properties) is **1 MiB**.
* Maximum size of a single table can be very large (documentation lists up to **500 TiB** per table in scale targets).

**When to use**

* Large-volume, low-cost storage of structured but non-relational records (e.g., IoT telemetry indexes, user preferences, logs, simple session stores). If you need rich query semantics or multi-region low latency, consider Cosmos DB Table API instead.

**Best practices**

* Design PartitionKey to balance load and avoid hotspots (partitioning is how you scale). Use row key for efficient lookups. If an entity can exceed 1 MiB, store the large blob in Blob Storage and store a pointer in Table Storage.

**Queue Storage — simple, durable message queuing for decoupling**

**What it is (short):** Azure Queue Storage is a lightweight, scalable queue service that stores messages for asynchronous communication between application components.

**Model & behavior**

* Producers enqueue messages; consumers invisibly dequeue and process them; messages have visibility timeouts and TTLs. It’s accessed via REST or SDKs.

**Key scale/limits (important)**

* **Message size**: up to **64 KiB** per message (for service versions 2011-08-18 and later).
* A single queue and the account can scale to very large capacity (scale targets list up to **500 TiB** per queue and high throughput rates depending on message sizes).

**When to use**

* Simple task queues, background job processing, decoupling front-end requests from back-end work, or buffering bursts of work. For advanced messaging features (FIFO, sessions, large messages, advanced routing), consider **Azure Service Bus**.

**Best practices**

* If you need to pass >64KiB payloads, store the payload in Blob Storage and put the pointer (URL + SAS) in the queue (claim-check pattern).
* Monitor queue length and processing latency; design idempotent consumers because messages may be delivered more than once.

**Cross-cutting: security, redundancy, monitoring & costs**

* **Encryption**: Server-side encryption is enabled by default for all Azure Storage; you can manage keys with Microsoft-managed keys, customer-managed keys (Key Vault), or bring-your-own-key patterns.
* **Access control**: Use Azure AD + role assignments where supported, Shared Access Signatures (SAS) for scoped access, and network-level controls (firewall, private endpoints).
* **Replication**: Choose redundancy (LRS/ZRS/GRS/GZRS/RA-GZRS) based on RPO/RTO and compliance needs. Some features (archive tier or certain SSD file shares) have compatibility caveats with specific redundancy types — check docs before picking.
* **Monitoring & lifecycle**: Use Azure Monitor/Diagnostics and lifecycle policies to manage costs and retention (e.g., automatically move blobs to cooler tiers or delete after retention window).