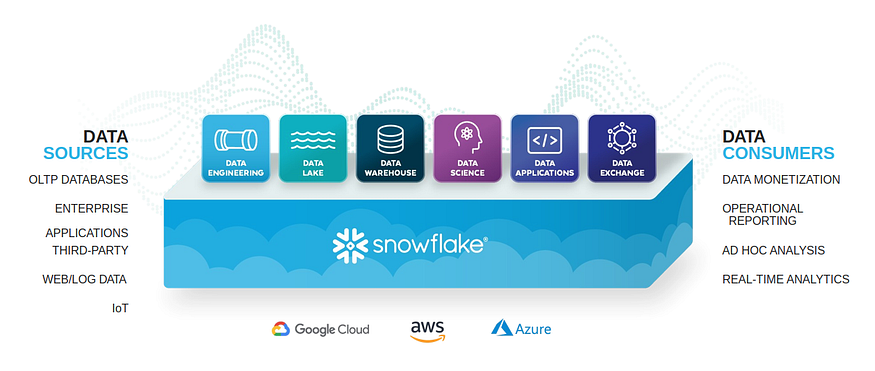
**What is Snowflake?**

Developed in 2012, **Snowflake** is a fully managed SaaS (software as a service) that provides a single platform for data warehousing, data lakes, data engineering, data science, data application development, and secure sharing and consumption of real-time / shared data.



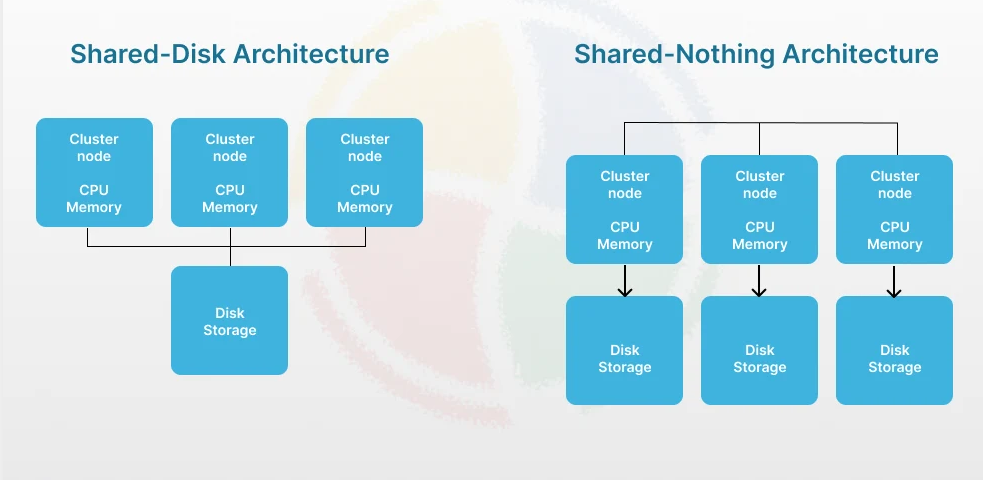
**Snowflake is a true SaaS offering**

* There is no hardware (virtual or physical) to select, install, configure, or manage.
* There is virtually no software to install, configure, or manage.
* Ongoing maintenance, management, upgrades, and tuning are handled by Snowflake.

**History:**

* Snowflake was developed in 2012 in San Mateo, California.
* It was built by three data warehouse experts:
  + Benoit Dageville
  + Thierry Cruanes
  + Marcin Zukowski
* Benoit Dageville and Thierry Cruanes previously worked as data architects at Oracle Corporation.
* The company’s name was chosen as tribute to the founders love of snow sports.

**Architechture:**



**Shared Disk Architecture:**

* Each node has its own CPU and memory, but **all nodes are connected to a common storage system.**
* The database (or application) ensures **synchronization and concurrency control** so multiple nodes don’t corrupt data.

**How it works:**

* Every node can read/write from the same disk.
* The storage layer is centralized, while processing is distributed.
* A lock manager is required to prevent conflicts (since many nodes may try to update the same data).

**Advantages:**

* **High availability:** If one node fails, others can still access the same data.
* **Easy data management:** No need to replicate or partition data.
* **Flexible scaling for reads:** Any node can serve read requests.

**Disadvantages:**

* **Scalability limits:** As the number of nodes grows, contention for the shared disk increases.
* **Performance bottleneck:** Storage I/O can become a choke point.
* **Expensive storage systems** required (like SAN/NAS).
* **Single point of failure**

**Examples:**

* **Oracle RAC (Real Application Clusters)**
* **IBM DB2 pureScale**
* **VMware vSphere clusters** (for shared storage VMs)

**Shared Nothing Architecture**

In a **shared nothing** system, **each node has its own CPU, memory, and disk/storage**.

* There is **no single point of contention** because nodes do not share memory or disk.
* Data is partitioned (sharded) across nodes, and queries are distributed to the appropriate nodes.

**🔹 How it works:**

* Each node manages its own subset of the data.
* Queries are executed by sending requests to relevant nodes, and results are combined.
* Fault tolerance comes from **replication** (data is copied across multiple nodes for redundancy).

**🔹 Advantages:**

* **Highly scalable:** Add more nodes to increase both compute and storage capacity.
* **No central bottleneck:** Each node works independently.
* **Commodity hardware friendly:** Runs well on cheap distributed servers.

**🔹 Disadvantages:**

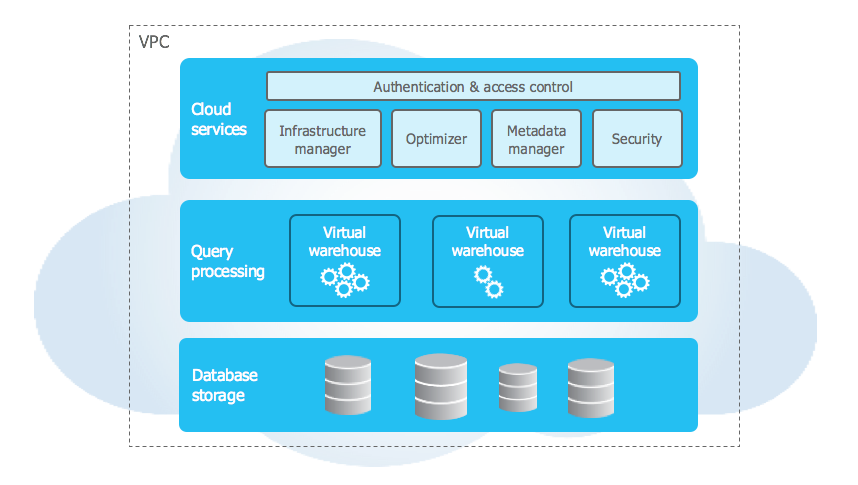
* **Complex data management:** Partitioning/sharding must be carefully designed.
* **Query complexity:** Cross-node joins can be slow.
* **Harder fault recovery:** If a node fails, replication/sync mechanisms must kick in.

**🔹 Examples:**

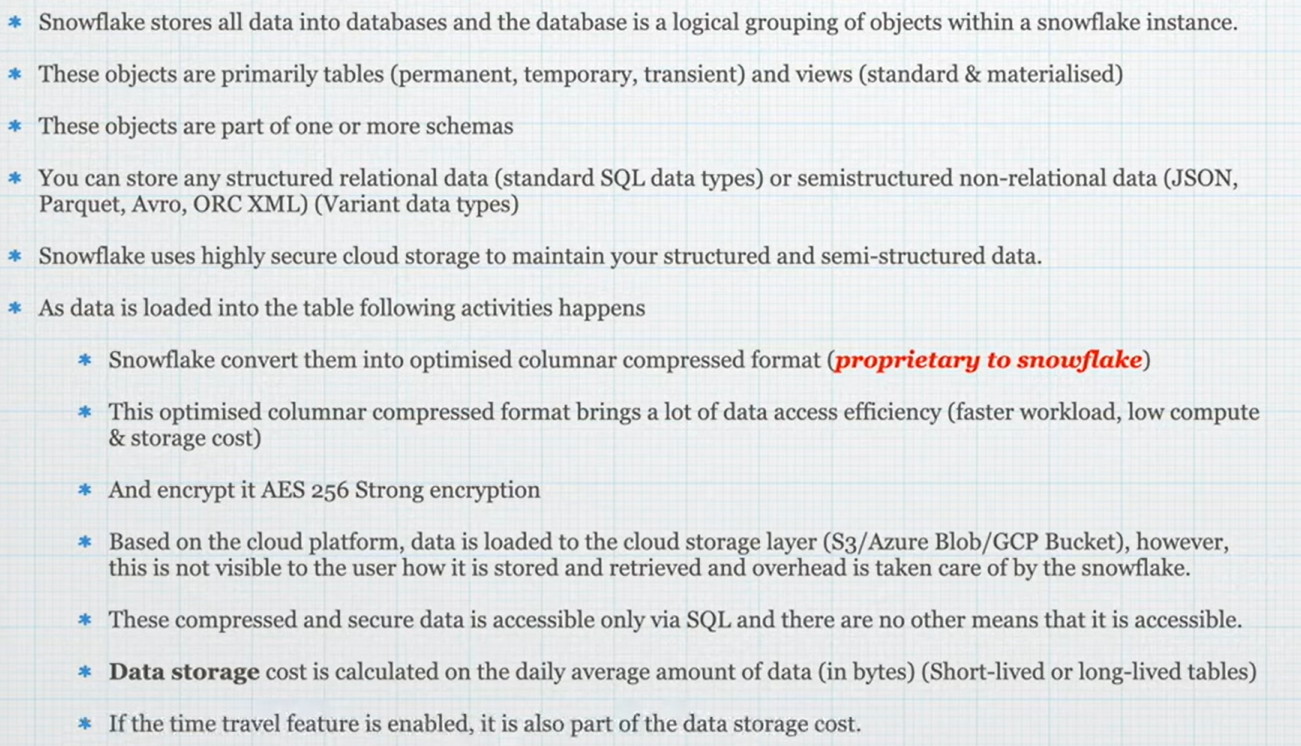
* **Google Bigtable**
* **Amazon DynamoDB**
* **MongoDB, Cassandra, HBase**

**Snowflake’s Architechture:**

* Decouple compute and storage – scale them independently
* Multi cluster shared data architechture.

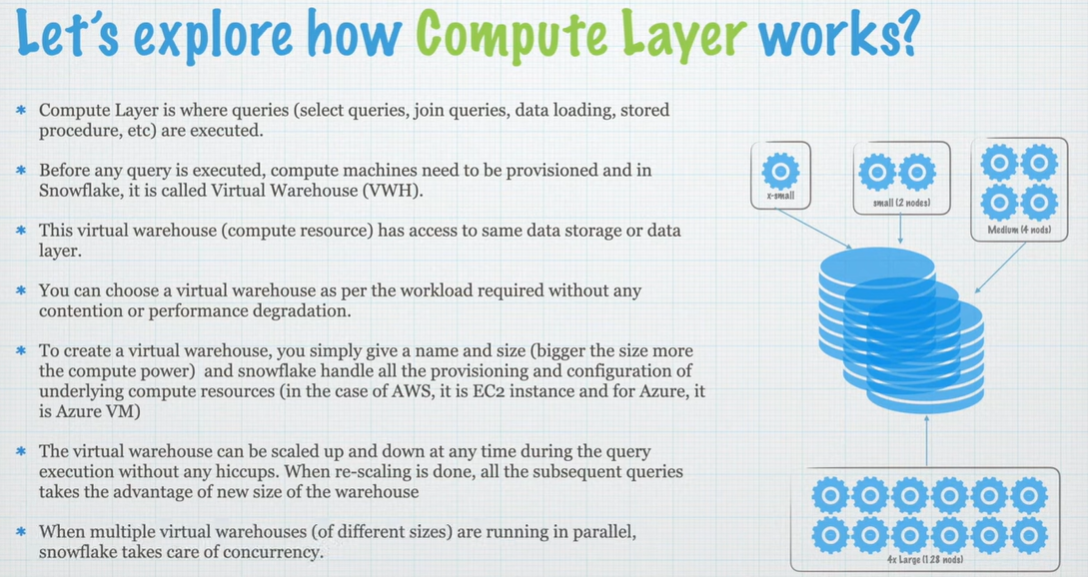


**Storage Layer:**

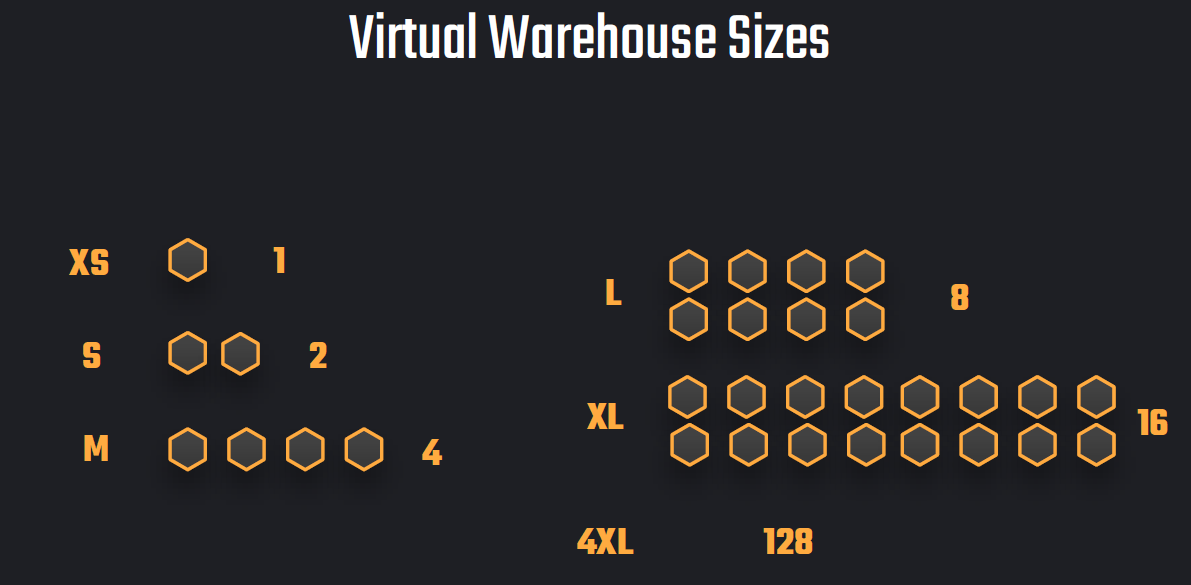


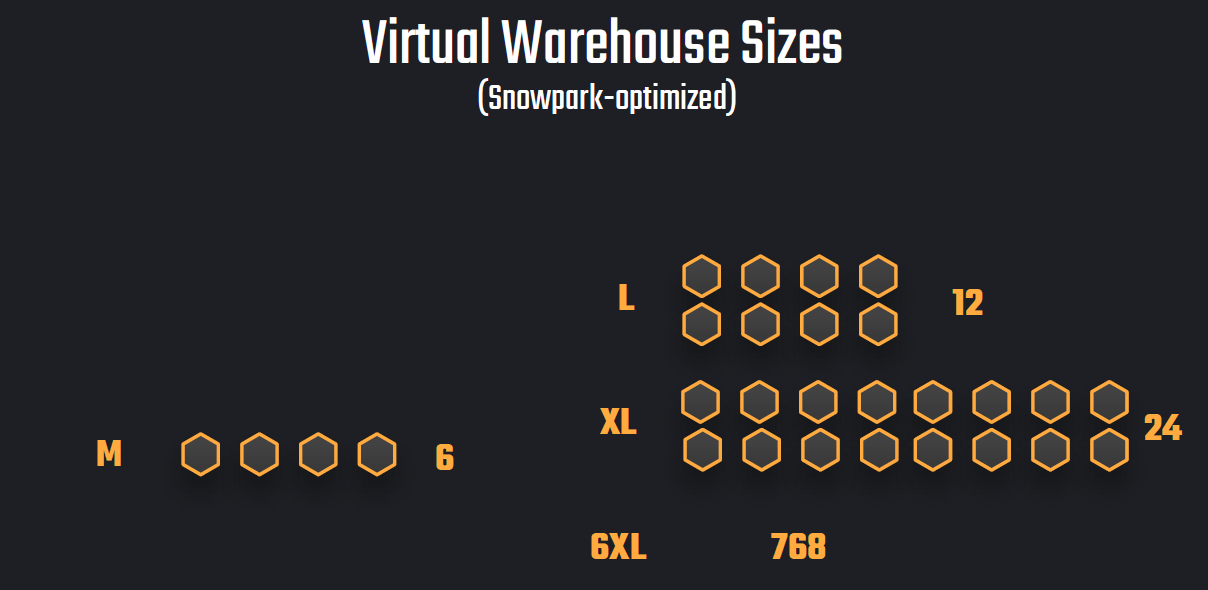
**Compute Layer:**

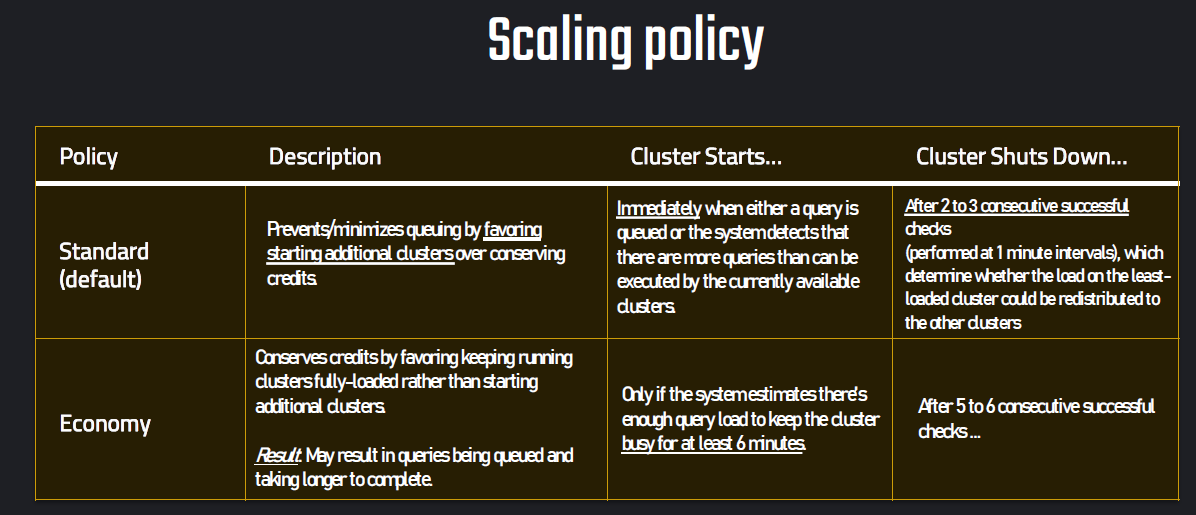
* It is also called “Muscle of the system”.
* Provides resources: CPU, Memory and temporary storage
* Warehouse sizes: XS(1), S(2),M(4), L(8), XL(16), 2XL(32), 3XL(64), 4XL(128)

****

****

****

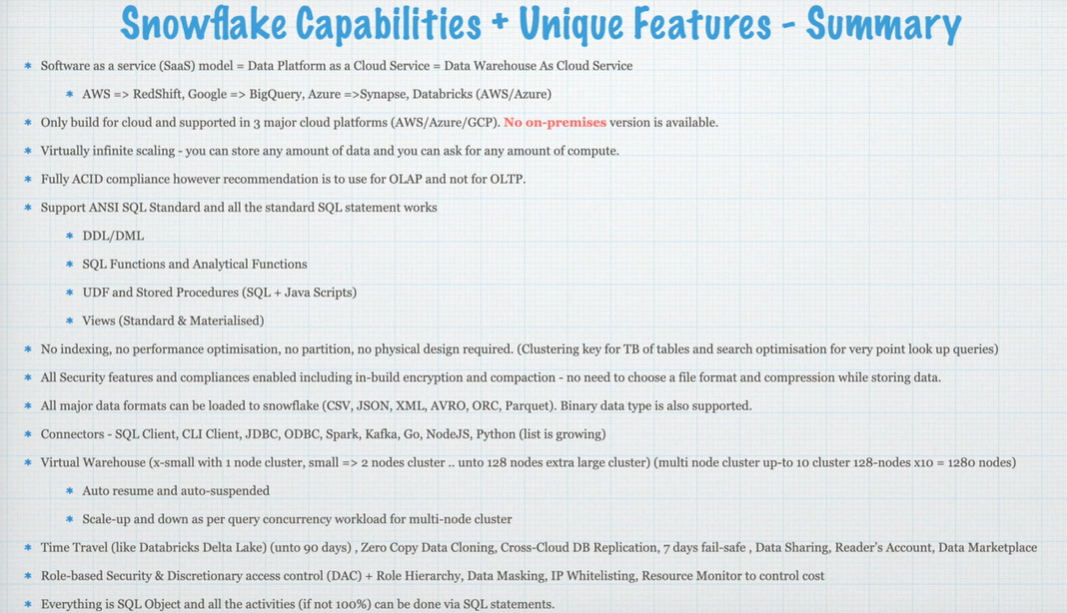
****

****

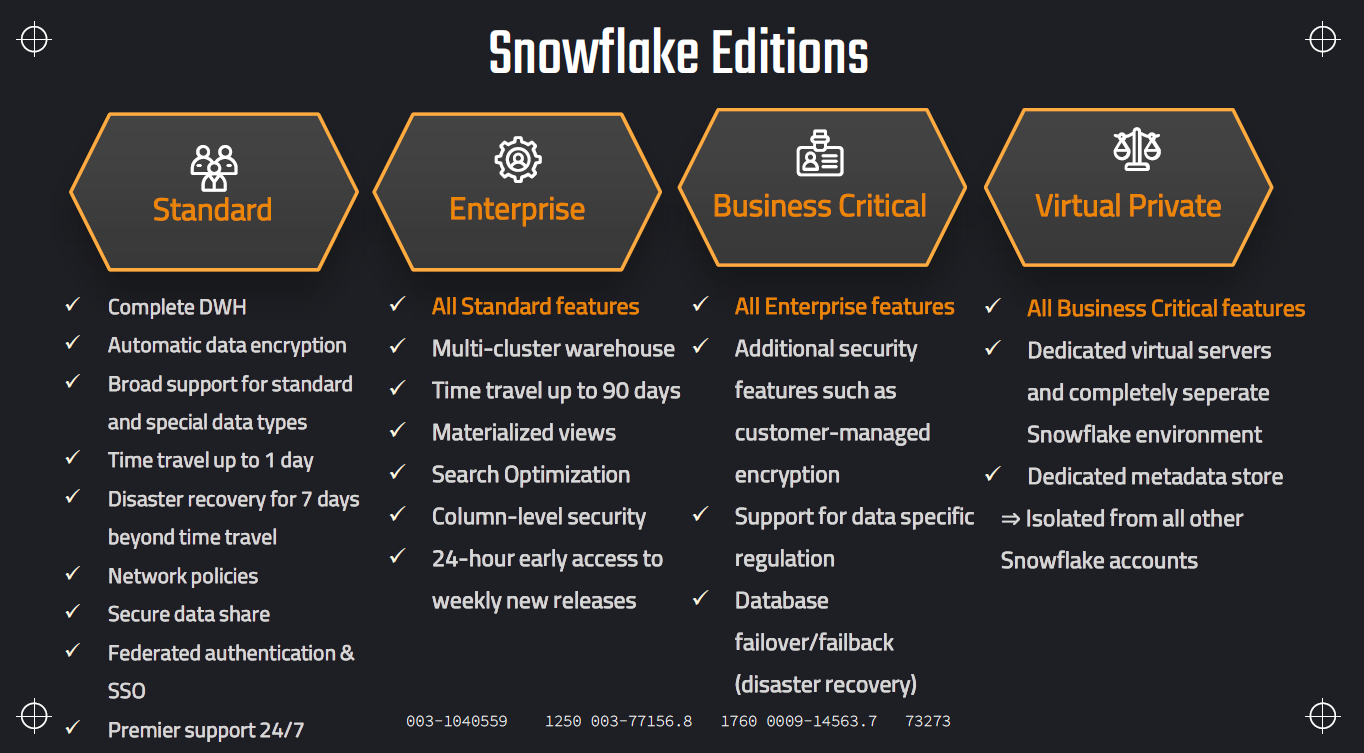
| Scenario | Recommended Policy | Reason |
| --- | --- | --- |
| BI dashboards, data analysts | STANDARD | Fast query response, less queuing |
| ETL/ELT batch jobs | ECONOMY | Cost efficiency over speed |
| Development / Testing | ECONOMY | Lower cost |
| Mission-critical reporting | STANDARD | Minimal performance degradation |

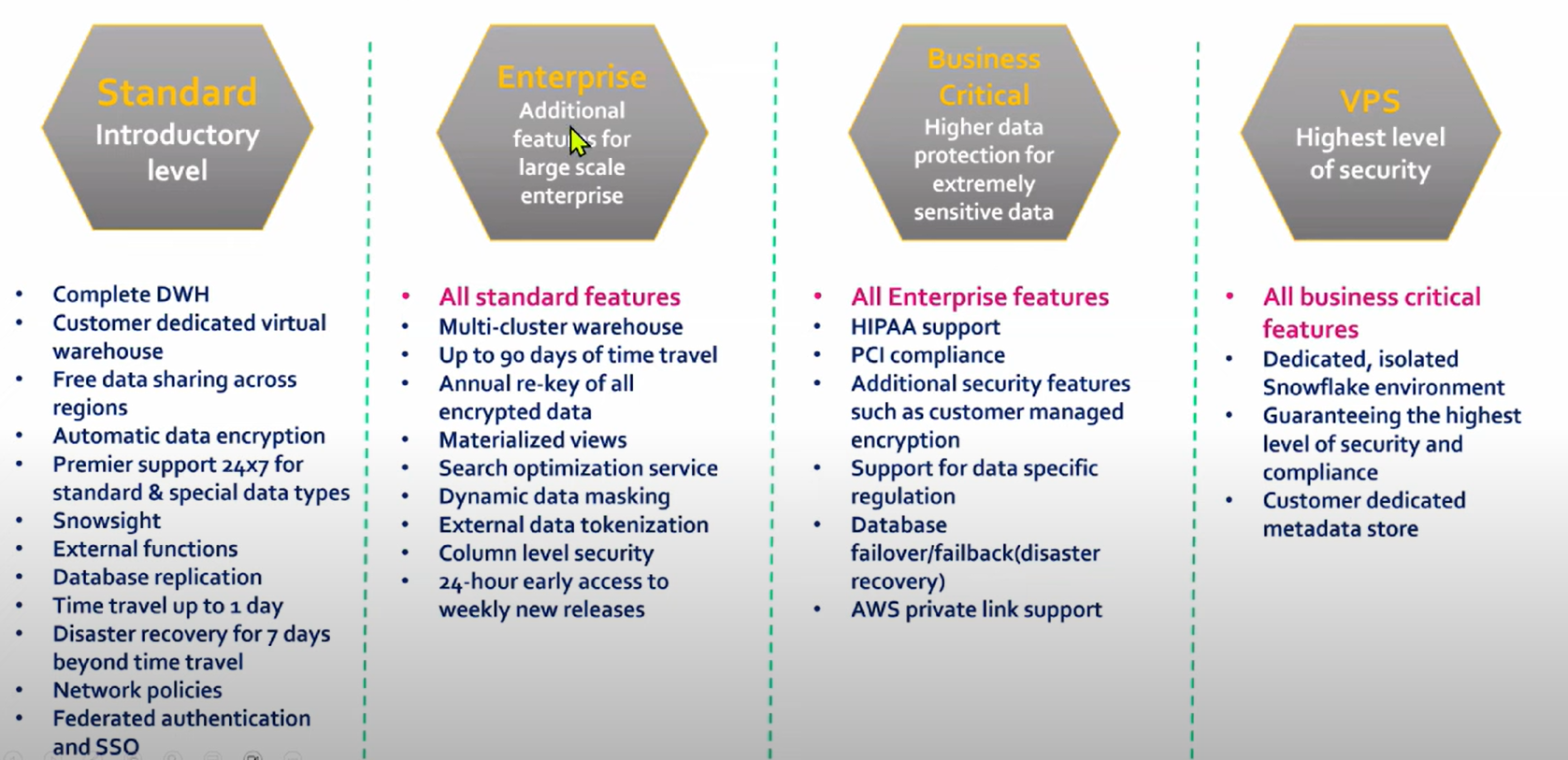
**Cloud Services Layer:**

****

****

**Snowflake Editions:**

****

****

**Compute, Storage and Data transfer costs:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Cost Type** |  |  | | --- | |  | | **Description** | | **Measured In** | | --- |  |  | | --- | |  | |
| |  | | --- | | Compute |  |  | | --- | |  | | |  | | --- | | Cost of running warehouses |  |  | | --- | |  | | |  | | --- | | Snowflake credits |  |  | | --- | |  | |
| Storage | |  | | --- | | Cost of data stored in tables/ stages |  |  | | --- | |  | | |  | | --- | | $ per TB per month |  |  | | --- | |  | |
| |  | | --- | | Data Transfer |  |  | | --- | |  | | |  | | --- | | Cost for egress or replication between regions/clouds |  |  | | --- | |  | | $ per GB |

**Compute Cost:**

Compute cost refers to the charges for the virtual warehouses (compute resources) that Snowflake uses to process your queries, load data, perform transformations, and more.

Compute cost depend upon:

* Virtual warehouse size
* Time active
* Region and cloud provider
* Edition
* Type of workload(single vs multi cluster WH)
* Auto suspend/ resume

Compute Cost = (Credits per hour × Run time in hours) × Credit price

Eg. If we run medium warehouse(4 credits/hour) for 15 minutes and cost per credit is $3 then

Cost = 4 \* (15/60) \* $3 = $3

Compute cost in different workloads:

Data loading: Copying data from stage to tables

Query execution: BI dashboards, reports

Data transformation: ELT/ DBT pipelines

Serverless: snowpipe(continuous data loading), Search optimization

**Factors Affecting Compute Cost**

|  |  |
| --- | --- |
| **Warehouse size** | Larger warehouses cost more credits per hour. |
| **Concurrency scaling** | If enabled, additional clusters can be auto-spun up, increasing cost. |
| **Multi-cluster warehouses** | Multiple clusters consume more credits. |
| **Auto-suspend/Auto-resume settings** | Helps reduce cost by pausing unused compute. |
| **Query performance** | Inefficient queries increase compute time and cost. |
| **Materialized views / Tasks / Streams** | Scheduled or background compute can add cost. |

**Tracking Compute Cost**

**Snowflake UI:**

Admin → Usage → Compute History

**SQL Table:**

SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_METERING\_HISTORY

|  |  |
| --- | --- |
| **WAREHOUSE\_METERING\_HISTORY** | Primary view to track compute credit consumption |

|  |  |
| --- | --- |
| **QUERY\_HISTORY** | Useful for linking query activity to compute usage |

|  |  |
| --- | --- |
| **LOGIN\_HISTORY** | See who’s triggering workloads |

|  |  |
| --- | --- |
| **TASK\_HISTORY** | Cost from background tasks |

|  |  |
| --- | --- |
| **PIPE\_USAGE\_HISTORY** | Cost from continuous data ingestion |

**Storage Cost:**

Storage cost is the charge for:

* Data stored in Snowflake tables (structured, semi-structured)
* Staged data (internal stages, not external)
* Time Travel and Fail-safe copies
* Cloning metadata (physical storage for changes)

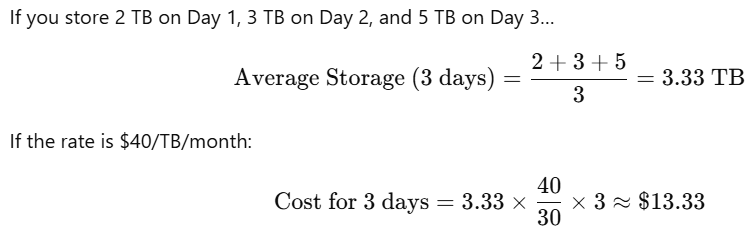
Storage costs are **billed monthly** and measured in **compressed TB per month**.

Tracked in SQL using SNOWFLAKE.ACCOUNT\_USAGE.STORAGE\_USAGE.

**On-Demand vs Capacity Storage**

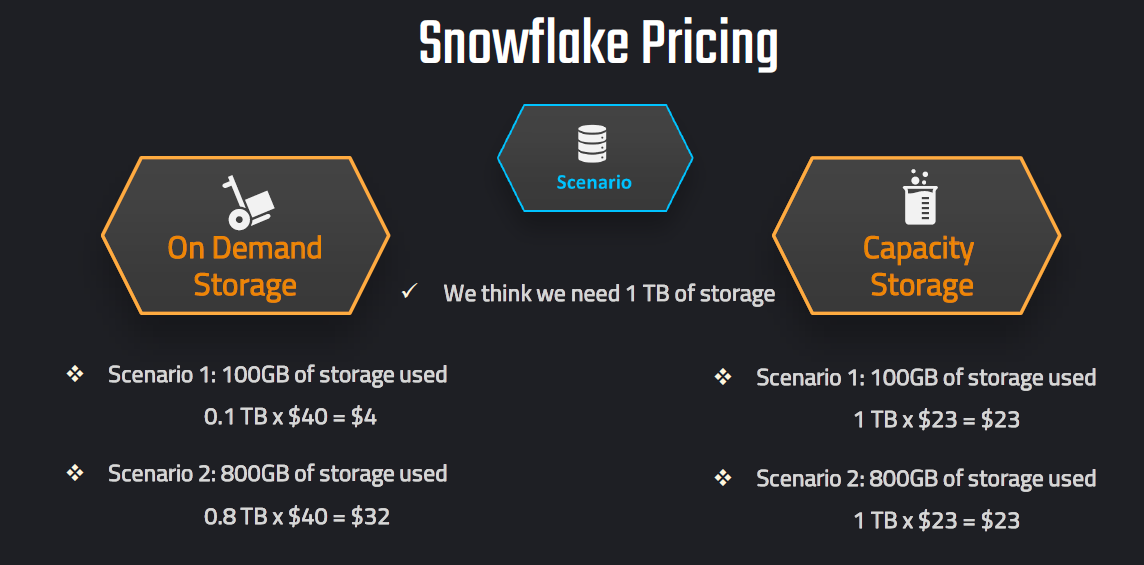
| Model | Description | Best for |
| --- | --- | --- |
| On-Demand Storage | Pay only for the actual storage you use each month. No commitment required. | Small to medium workloads, unpredictable growth |
| Capacity Storage | You commit to a predefined amount of storage upfront (e.g., 100 TB for 1 year) at a discounted rate. | Large enterprises with predictable data growth |

****

****

**Key Comparision:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **On-Demand** | **Capacity** |
| **Payment Model** | **Pay-as-you-go** | **Prepaid/Commited** |
| **Commitment Term** | **None** | **1-3 years typically** |
| **Cost per TB** | **Higher** | **Discounted (10-20%)** |
| **Flexibility** | |  | | --- | | **High — scale up/down easily** |  |  | | --- | |  | | |  | | --- | | **Lower — you pre-purchase capacity** |  |  | | --- | |  | |
| **Billing Frequency** | **Monthly** | **Upfront or scheduled** |
| **Unused Capacity** | **No issue** | **You still pay for it** |

****

**Data Transfer Cost**

| Scenario | Cost Applies? | Explanation |
| --- | --- | --- |
| Querying data in the same account and region | ❌ No | No egress |
| Secure data sharing in the same region | ❌ No | Free |
| Secure data sharing across regions | ✅ Yes | Egress from source region |
| Database replication to another region | ✅ Yes | Data transferred out |
| Data unloaded to external storage (e.g., S3 in another region) | ✅ Yes | Egress from Snowflake |
| Data moved between cloud providers | ✅ Yes | Cross-cloud charge |
| External table query (if Snowflake must pull from another region) | ✅ Possibly | Depends on data location |

**Egress** = data transferred **out of your Snowflake region** or **to another cloud provider**.

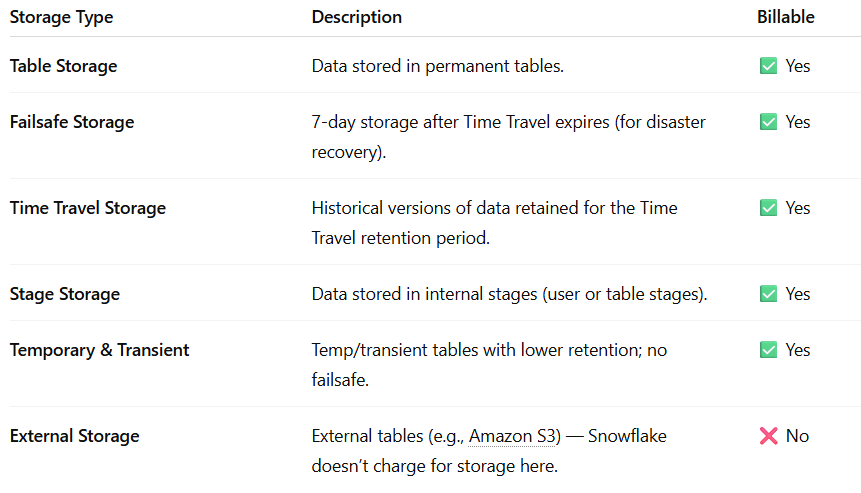
If data **stays inside the same region and cloud**, there’s **no egress cost**.

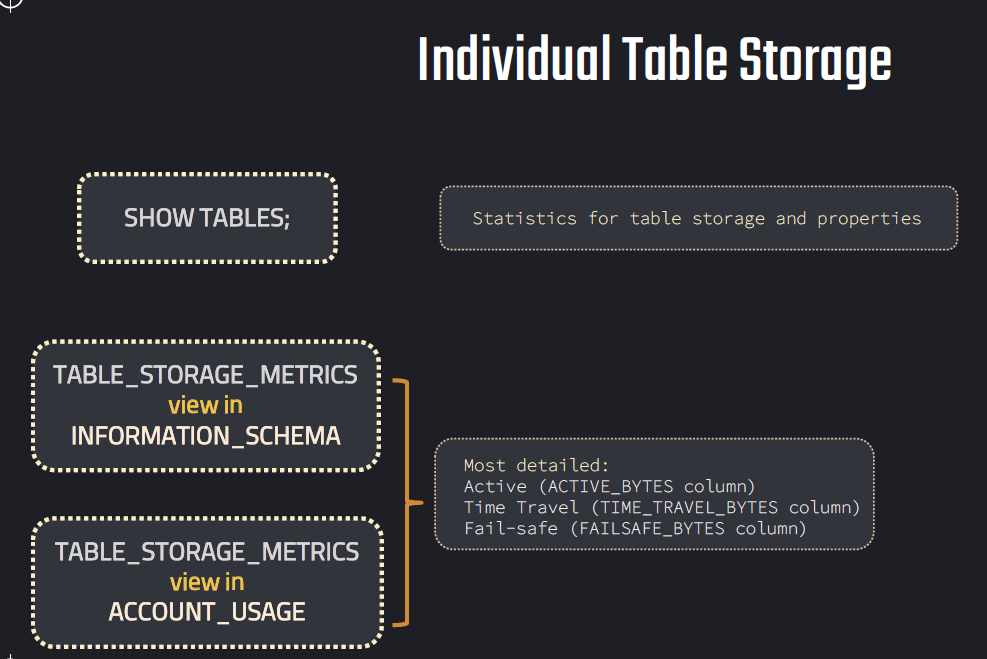
Tracked in SNOWFLAKE.ACCOUNT\_USAGE.DATA\_TRANSFER\_HISTORY.

Data Transfer Cost = (GB Transferred) × (Per GB Rate)

**Storage monitoring:**

Storage monitoring help you track how much storage is used, where it’s used, and by whom.





Navigate to Admin -> Cost Management -> storage

You can view

* Storage by database
* Storage by stage
* Time travel and fail-safe data
* Trend over time



| **Method** | **Purpose** | **Refresh Lag** |
| --- | --- | --- |
| ACCOUNT\_USAGE.STORAGE\_USAGE | Account-level daily storage | ~45 min |
| ACCOUNT\_USAGE.TABLE\_STORAGE\_METRICS | Table-level storage details | ~45 min |
| INFORMATION\_SCHEMA.TABLES | Real-time table size info | Real-time |
| Snowsight UI | Visual trends and breakdown | Near real-time |

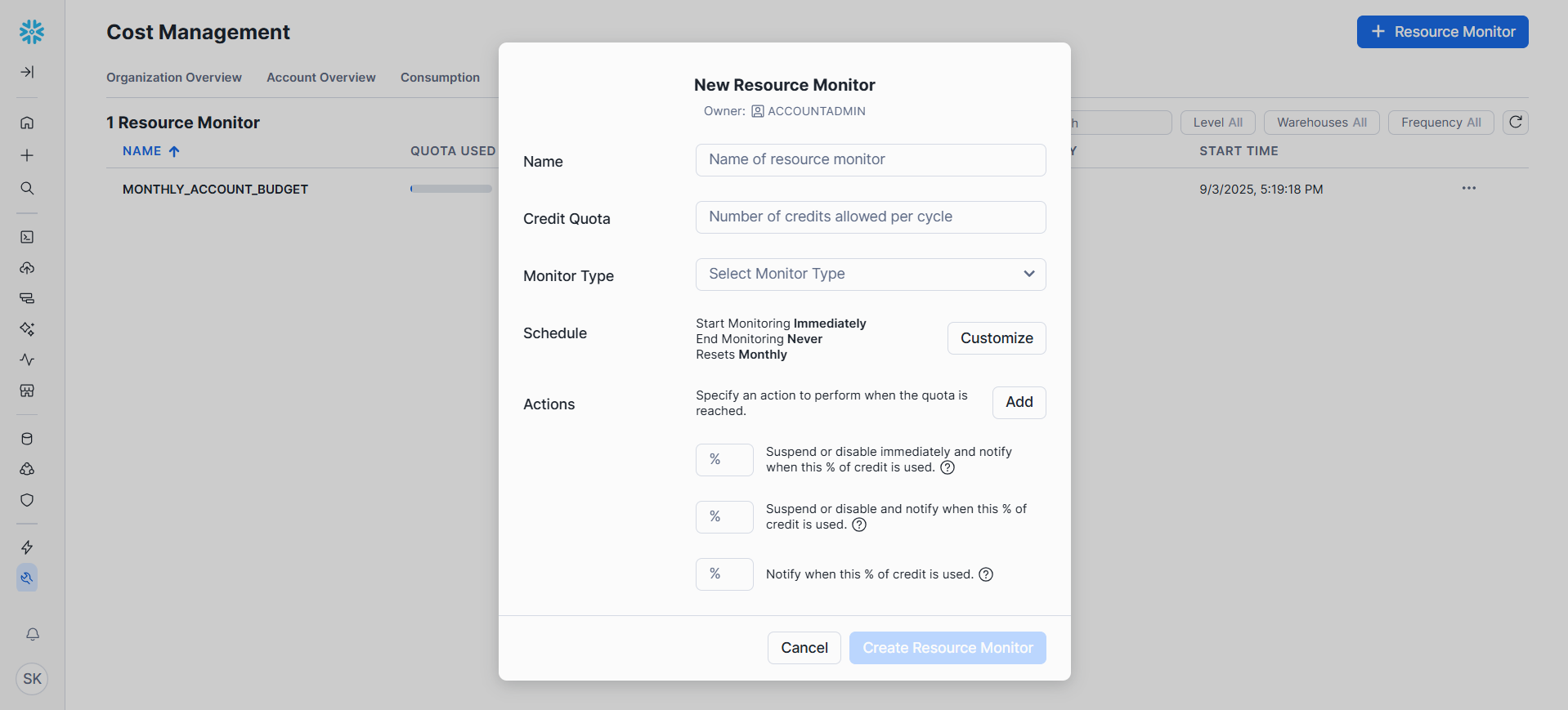
**Resource Monitors:**

Resource Monitors are a powerful feature that allows you to track and control compute credit usage to help manage costs effectively.

* Resource monitors **only track compute credit usage**, not storage or data transfer.
* They can **only be created by account administrators** or roles with MONITOR privileges.
* You can assign one resource monitor to multiple warehouses, but **a warehouse can only have one resource monitor**.
* Triggers are based on **percentage of credit quota**, not on absolute credits consumed.

Query:

SELECT \* FROM SNOWFLAKE.ACCOUNT\_USAGE.RESOURCE\_MONITORS;



Name: Unique name of the monitor

Credit Quota: Maximum number of credits allowed within the specified period

Monitor type: Account | Warehouse

Schedule: resets Daily, Weekly, Monthly, Yearly, Never

Actions:

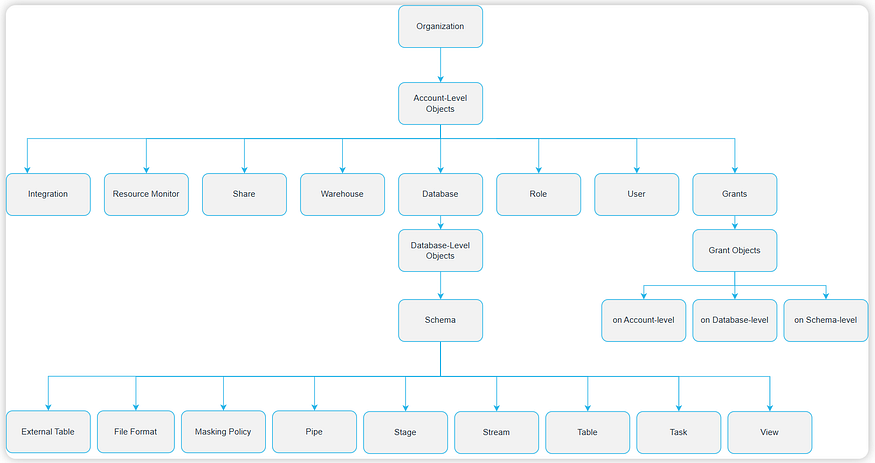
NOTIFY: Sends an alert when the threshold is reached

SUSPEND: Suspends the assigned warehouses after completing running queries

SUSPEND\_IMMEDIATE: Suspends the warehouse immediately, canceling running queries.

**Best Practice:** Use NOTIFY at 75–80% and SUSPEND or SUSPEND\_IMMEDIATE at higher percentages to control cost gracefully.

**Object hierarchy:**



**SnowSQL:**

* SnowSQL is Snowflake’s official command-line client.
* Lets you connect, run SQL, manage objects, and automate tasks.
* Supports both interactive mode and script execution.
* Useful for DevOps, CI/CD, data loading/unloading, and admin operations.