HOW TO OPERATE SNOWFLAKE AT ENTERPRISE SCALE

snowflake®

WHITEPAPER

TABLE OF CONTENTS

INTRODUCTION

Snowflake provides a cloud-built data platform that delivers instant and infinite scalability through a multi-cluster, shared data architecture. By separating compute and storage, Snowflake eliminates contention between simultaneous workloads, queries, and users, which all access a single copy of all your structured and semi-structured data. As part of its core engine – governance, data protection, security, performance, and cost optimization – are all built into the architecture.

This paper provides best practices and automation techniques to assist you in operating Snowflake at enterprise scale. Snowflake customers can execute these strategies at any scale. But these recommendations may be most helpful to those with a large number of Snowflake users working across different teams that run a large number of queries or other workloads against their cloud-built data warehouse or data lake.

SECURITY & COMPLIANCE

The paramount concern for every IT organization is the development of robust data security to minimize the risk of malicious or accidental loss of data.

Snowflake provides a secure data environment with near-zero administration, but today's cybersecurity threats often occur "at the seams." The recommendations below help ensure the strongest security possible for potential areas of weakness. The three key security areas to focus on are authentication, authorization and auditing (AAA).

All data stored in Snowflake is protected using AES-256 encryption. For organizations that require additional security measures, this section describes how to set up encryption key procedures and create auditing controls on top of Snowflake.

From a compliance perspective, every organization is beholden to different industry- and region-specific regulations. That's why it's crucial to satisfy all levels of required security and compliance, which are detailed below.

Authentication

Provisioning and managing authentication for any system at large scale (>1000 users) can become a significant overhead for any IT team in a large organization. Snowflake provides enterprise-grade user authentication features that minimize the risk of unauthorized access and reduce management overhead.

Here are some recommendations for user authentication in Snowflake:

1. Most large enterprises already use Single Sign-On (SSO) for cloud applications. All editions of Snowflake support SSO with most SAML2.0 compliant identity providers (IdPs). We recommend using SSO for user authentication. For more information, click here.
2. If using SSO, we highly recommend you create (or alter) users so they have don't have passwords in Snowflake. This effectively disables Snowflake authentication for these users and requires them to log in using federated authentication.
3. While SSO makes it easy to use your existing identity provider (IdP) for authentication, it does not create and maintain users/roles within Snowflake. SCIM is an open specification to help facilitate the automated management of user identities and groups (i.e., roles) in cloud applications using RESTful APIs. Currently, Snowflake supports SCIM 2.0[1] to integrate Snowflake with Okta, which functions as an identity provider. Snowflake also supports identity providers in addition to Okta and Microsoft Azure. Their users and groups can also be provisioned in Snowflake, which functions as the service provider. For more information, click here.
4. It's imperative to use multi-factor authentication (MFA) for all your users given the growing sophistication of password-related hacking via phishing, social engineering and other methods. Snowflake supports MFA as an integrated Snowflake feature. For more information, click here.

1 As of the time of writing of this whitepaper, SCIM 2.0 support in Snowflake is still in private preview.

5. Use OAuth which is an open-standard protocol that allows supported clients authorized access to Snowflake without sharing or storing user login credentials. For more information, click here.

Authorization: Snowflake Object Security

Snowflake's approach to database object access control combines aspects from both of the following models:

- Discretionary access control (DAC): Each object has an owner, who can grant access to that object.
- Role-based Access Control (RBAC): Access privileges are assigned to roles, which are then assigned to users.

Roles can also be granted to other roles, creating a hierarchy of roles. The privileges associated with a role are inherited by any roles above that role in the hierarchy. There are four system generated roles that are pre-provisioned in every Snowflake account: **ACCOUNTADMIN, SECURITYADMIN, SYSADMIN,** and **PUBLIC.**

Detailed information on access control in Snowflake can be found here.

The following are some recommendations and best practices around access control in Snowflake:

1. The account administrator (ACCOUNTADMIN) role is the most powerful role in the system. Users with the ACCOUNTADMIN role can view and operate on all objects in the account, can view and manage Snowflake billing and credit data, and can stop any running SQL statements. Recommendations on the ACCOUNTADMIN role:
    i. Assign the ACCOUNTADMIN role only to a select/limited number of people in your organization.
    ii. All users assigned the ACCOUNTADMIN role should also be required to use multi-factor authentication (MFA) for login (for details, see Configuring Access Control).
    iii. Assign this role to at least two users. We follow strict security procedures for resetting a forgotten or lost password for users with the ACCOUNTADMIN role. These procedures can take up to two business days. Assigning the ACCOUNTADMIN role to more than one user avoids having to go through these procedures because the users can reset each other's passwords.
2. By default, when your account is provisioned, the first user is assigned the ACCOUNTADMIN role. This user should then create one or more additional users who are assigned the SECURITYADMIN role. All remaining users should be created by the user(s) with the SECURITYADMIN role.
3. Avoid using the ACCOUNTADMIN role to create objects. The ACCOUNTADMIN role is intended for performing initial setup tasks in the system and managing account-level objects and tasks on a daily basis. As such, it should not be used to create objects in your account unless you absolutely need these objects to have the highest level of secure access.
4. In Snowflake, privileges are granted to roles, not users. While creating role hierarchies, keep in mind privileges granted to lower-level roles (child roles) are inherited by higher-level roles (parent roles). We recommend creating a hierarchy of roles aligned with business functions in your organization and ultimately assigning these roles to the SYSADMIN role. For more information, click here

Audit Controls

The ability to audit the usage of any IT tool is a key component of any security policy. Snowflake provides the following features to help enable organizations build effective auditing processes and tools on top of Snowflake:

1. The LOGIN_HISTORY and LOGIN_HISTORY_BY_USER returns login activity within Snowflake for the past seven days.
2. The QUERY_HISTORY family of table functions returns the query history within Snowflake along various dimensions for the past seven days.
3. ACCOUNT_USAGE schema includes long term historical data with results held for up to one year but with up to three hours of latency.
4. These system tables provide historical information on Snowflake access and queries run by various users. But we still need to understand how data

was manipulated by those users/queries to satisfy auditing standards. To achieve this, Snowflake provides its Time Travel feature which keeps track of all data changes at a transaction level. In addition to providing instant data recovery, Time Travel can also analyze data usage and manipulation for the previous 90 days.

Snowflake encourages customers to develop automation on top of these Snowflake system tables and features to satisfy their audit control needs.

Network/Site Access Control

Snowflake is reachable on the public Internet. Anyone with valid Snowflake account credentials can access the service from anywhere. However, Snowflake provides additional features for organizations with strict network access control requirements:

1. Network Policies enable you to create IP, address-based blacklists and whitelists to deny or allow network access to Snowflake.
2. Customers using Snowflake on Amazon Web Services (AWS) can avoid having their traffic traverse the public Internet by requesting private VPC-to-VPC connectivity using AWS PrivateLink[2].
3. If your organization needs to comply with stringent security standards that require complete network-level isolation, Snowflake provides Virtual Private Snowflake (VPS). VPS provides a Snowflake deployment, within a separate, dedicated AWS Virtual Private Cloud (VPC).

Data Security

All data stored in Snowflake is encrypted using AES-256 strong encryption. Customers with enhanced data security needs should consider leveraging the following features within Snowflake:

Periodic Rekeying is an enhanced feature available in the Enterprise edition of Snowflake. It automatically replaces active keys with new keys on a periodic basis and retires the old keys. You should consider enabling this for your Snowflake account if your security postures require periodic rekeying of encrypted data.

By default, data in Snowflake is encrypted using a master key, which Snowflake manages. Customers whose security postures requires them to manage their own master key should consider purchasing the Business Critical (BC) edition of Snowflake. The BC edition comes with Tri Secret Secure in which a customer-provided master key is combined with a Snowflake-managed key to produce a composite master key, which is then used to encrypt data.

Security Standards Compliance

We encourage you to refer to the following table and decide on the appropriate Snowflake edition based on your security compliance needs:

	Snowflake Editions			
Compliance Standard	Standard	Premier	Enterprise	Business Critical
Soc 1 Type II	✓	✓	✓	✓
Soc 2 Type II	✓	✓	✓	✓
Support for HIPAA				✓
Support for PCI DSS				✓

See our documentation on Snowflake Editions for the most up-to-date information on features/standards supported in every edition. For an in-depth look at different security related features within Snowflake, please refer to this whitepaper.

2 PrivateLink is an AWS feature. Similar private connectivity for Azure/GCP is planned for early 2020.

BUDGETING & COST OPTIMIZATION

Snowflake's operational expense (OpEx) cost structure represents a huge improvement over the capital expense (CapEx) structure inherent to legacy data warehouses.

While some finance teams may experience initial trepidation over changing the way budgets are created, most quickly recognize the savings associated with the overall cost of a cloud-built data warehouse, data lake or any solution you enable with Snowflake. Features such as per-second pricing, auto-scaling, and zero-copy cloning help to minimize spending. When warehouses are not in use, Snowflake uses auto-suspension and auto-resumption by default to preserve credits. Organizations only pay for what they use, down to the second.

The best practices below are designed to reduce the risk of overspending by illustrating how to achieve oversight and insight into data warehouse usage. By practicing efficient warehouse scaling, creating resource monitors, paying attention to options such as table storage, and using cost chargebacks, organizations can take complete control over their expenditures.

Auto-Suspend & Auto-Resume

Auto-suspension & Auto-resumption can suspend or resume a Snowflake virtual warehouse (compute cluster). Both of these features are on by default and we recommend you do not disable these features. Otherwise, you run the risk of using Snowflake credits when a warehouse is idle. You can use the 'show warehouses' command to see if any warehouses have Auto-Suspend and Auto-Resume turned off.

Scaling Warehouses

Snowflake compute costs depend on three factors: the size of the warehouse, the number of clusters in a multi-cluster warehouse, and the number of seconds the warehouse is in the active state. Because of this pricing model, going with a larger warehouse might cost the same since. In theory, a warehouse double the size should finish processing a query in half the time. However, not every query will benefit 2x in performance by scaling up a size. Moreover, every time a warehouse becomes active, there is a minimum charge of 60 secs. Keeping this in mind, it's always best to test the most frequently used queries against different warehouse sizes. Start with a smaller size and keep scaling the warehouse up until you meet your expected performance (SLA). Most Snowflake customers will size up till either one of the following two conditions are met:

1. Going up a size does not result in double the performance (half the query execution time).
2. Or, scaling up one size makes the query finish in less than 60 seconds (for cases where a SLA is above 60 seconds).

Cost Chargebacks

Large organizations that have multiple business units (BUs) often use an internal "chargeback" cost model for data warehouse usage. Snowflake makes it easy to track usage based on warehouses or users. Basic cost reports can be found in the Snowflake UI billing dashboard. For detailed chargeback reports, we recommend you create a separate Snowflake role for each cost center in your organization. In addition, have that role be the 'owner' of any warehouses created for use by that team or cost center. By doing so, you can easily create monthly or quarterly chargeback reports for each of your cost centers (see Appendix A for example queries).

Efficient Warehouse Usage

Each time you increase a virtual warehouse by a T-shirt size, you double the cost of credits. This can result in inefficient use of resources if a very large (eg. X2-Large) warehouse is kept running in an idle state. Use these large sized warehouses only when needed and suspend them using the Alter Warehouse Suspend command right after the query concludes. If a workload requires a warehouse to run constantly, such as a BI dashboard with a large number of users, we recommend you use smaller sized warehouses with Snowflake's Multi-cluster Warehouse feature enabled for handling concurrency.

Table Storage

While storage costs usually constitute a small portion of your Snowflake costs, it can add up, especially when a large number of tables use the Time Travel and Fail-safe features. We recommend you leverage Temporary and Transient tables for storing temporary data or for data easily reproduced. Temporary/Transient tables can have zero or one day of Time Travel and do not have any fail-safe storage costs.

Zero-Copy Cloning

Use Snowflake's Zero-Copy Cloning to create copies of tables or entire databases. By copying only the metadata, this feature lets you make full working copies of data without having to pay for duplicate storage. Zero Copy Clones are used extensively by Snowflake customers that have mature DevOps processes. For detailed information on using Zero Copy cloning for DevOps, please refer to this blogpost.

Resource Monitors

Snowflake provides resource monitors to help control costs and avoid unexpected credit usage caused by running idle warehouses. You can use resource monitors to impose limits on the number of credits warehouses consume within a specified interval or date range. When a warehouse approaches or reaches these limits, the resource monitor can trigger various actions, such as sending alert notifications and/or suspending the warehouse.

We recommend you create at least two types of resources monitors. First, set one account level resource monitor that alerts you if credit usage has exceeded the whole account. Secondly, create a separate resource monitor for each set of Snowflake warehouses belonging to various cost centers with an appropriate action (Notify or Notify and Suspend) to take when a warehouse exceeds the threshold.

PERFORMANCE OPTIMIZATION

Snowflake provides all the tools necessary to deliver a performant cloud data platform. Understanding how the unique Snowflake features work helps to enable smarter and more cost-effective query execution, and better end-user experiences.

Below are explanations of five Snowflake performance features, as well as advice on how to optimize them to the needs of your organization. First, establish criteria for creating and provisioning new warehouses (workload isolation). Then, determine how to optimize performance through scaling compute, clustering, and caching. Selecting the right method for data ingestion will complement common techniques for performance optimization.

Workload Isolation

First and foremost, it's important to come up with a criteria for when to create a new Snowflake virtual warehouse. Separate warehouses provide workload isolation and thus result in a consistent user experience without impacting performance. In Snowflake, you are not penalized or charged incrementally for additional warehouses. You are only charged when those additional warehouses are provisioned. With this in mind, the most optimized way of using Snowflake is to define the right sized warehouse and uptime for each user group or workload profile. This means you could have hundreds or thousands of these warehouses for an enterprise. Individual user groups or workload profiles are defined by the following:

1. Group of users (or an automated process) running a homogeneous set of queries
2. A standardized business SLA

For example, for a given data set, there might be two user groups accessing that data: executives and analysts. The executive dashboards might scan more data, the query SLA might be tighter, and there will be less usage (fewer people and fewer hours of access). In this scenario, the executive dashboards will most likely need a larger warehouse than the analyst warehouse. The most cost-optimized way to configure the warehouses in this scenario is to assign a larger warehouse for the executives and a smaller, multi-cluster warehouse for the analysts.

Scaling Compute

In Snowflake, there are two ways to scale performance: you can vertically scale an individual cluster or automatically add/remove additional clusters using the Multi-cluster Warehouse capability.

Use vertical scaling if you wish to alter query performance. How can you tell if a certain query would benefit from a larger-sized warehouse? A leading indicator is the spill amount for local and remote storage (shown below) viewable via the Query Profile.

Total Statistics

IO	
Scan progress	100.00 %
Bytes scanned	182.42 GB
Percentage scanned from cache	47.96 %
Bytes written	71.81 GB
Network	
Bytes sent over the network	2.01 TB
Pruning	
Partitions scanned	35,954
Partitions total	87,382
Spilling	
Bytes spilled to local storage	10.40 TB
Bytes spilled to remote storage	1.85 MB

For handling concurrency, we recommend you use a Multi-cluster Warehouse. To help decide whether you have a concurrency issue, you can monitor how much time queries spend waiting in the execution queue of a warehouse. You can do this by looking at the QUEUED_OVERLOAD_TIME attribute available in the QUERY_HISTORY view.

Clustering

Automatic Clustering is a service in Snowflake that orders data according to your query patterns. It results in better query performance for very large tables where the ordering was not ideal at the time the data was inserted or loaded. Or, extensive DML caused the table's natural clustering to degrade. You can turn on automatic clustering by picking one or more columns in a table as clustering keys. Clustering is not needed for most tables. We recommend clustering for large tables (> 1TB in size), where at least one of the following conditions holds true:

The order in which the data is loaded does not match the dimension by which it is most commonly queried (e.g. the data is loaded by date, but reports filter the data by ID). If your existing scripts or reports query the data by both date and ID (and potentially a third or fourth column), you may see some performance improvement by creating a multi-column clustering key.

Query Profile indicates a significant percentage of the total duration time for typical queries against the table is spent scanning. This applies to queries that filter on one or more specific columns.

For large tables that have data loaded in bulk, and do not have new data added constantly, you can sort the data based on the dimension by which it is most commonly queried. Doing this might remove the need to turn on clustering for a table. However, any addition of new data will require you to re-sort the table to maintain query performance.

For detailed information on how clustering works in Snowflake, see here.

Caching

Effectively leveraging the two types of caches (*data cache and results cache*) available in Snowflake can improve performance and reduce costs. Here is what you can do to get the most out of caching in Snowflake:

1. Data cache is populated by moving Snowflake data partitions from remote cloud object storage into the warehouse node SSD and memory. Data cache is reset every time a warehouse exits the active state. Thus, if a particular warehouse runs regular queries on a large data set, then it might be advisable to keep the warehouse running or, at least, increasing the auto-suspend timer value. The extra cost of a query that doesn't leverage the data cache might outweigh the cost of keeping an idle warehouse running.

2. Segment workloads on different warehouses such that users querying the same data end up using the same warehouse. This will enhance the probability of different users hitting the same partitions in the data cache.
3. The results of every query in Snowflake are cached for 24 hours as the results cache. Snowflake will automatically use the results cache if the query is the same and the underlying data has not changed. However, you can also leverage the results cache directly by using the RESULT_SCAN function (for example, to paginate through the results of a query).

Data Ingestion

Most Snowflake customers ingest data using either a dedicated warehouse or by leveraging the Snowpipe service. Here are some things to keep in mind regarding data ingestion:

1. When using a dedicated warehouse to ingest data, make sure there are enough files to take advantage of the 8 files/node parallelization. For example, if using a large-sized warehouse for ingestion, make sure you have at least 64 (8 nodes *8 files/node) files available in the stage to fully parallelize the ingestion and minimize ingestion time.
2. We recommend having individual staged files between 10-100MB in size.
3. For Snowpipe auto-ingest, keep in mind that the SQS integration cost is 0.06 credits for each 1000 notifications. To keep this cost minimal, it's better to have a smaller number of large files than a large number of small files.
4. Lastly, as mentioned under the clustering section, loading data in the right order (e.g. in the same order as the clustering key) will result in optimized querying and will minimize auto-clustering cost.

CONCLUSION

While adoption of a cloud-built data warehouse enables every organization to run operations at enterprise scale with near-zero management, companies are encouraged to customize their set-up to match security and compliance requirements and to optimize performance for their own needs.

With the best practices provided here, organizations can grow their data warehouse usage with confidence and control, they can rest assured with the knowledge that Snowflake's modern infrastructure provides the underlying infinite scalability and unlimited concurrency needed for data-driven decisions at all times.

APPENDIX A: MONITORING QUERIES

This appendix provides some example queries that can be used to monitor Snowflake cost and usage.

1. Queries run in the past 2 hours:

```
select *
from table(snowflake.information_schema.query_history());
```

2. Warehouse load for the last 14 days:

```
select *
from table(information_schema.warehouse_load_history());
```

3. Warehouse Load in 5-minute intervals for the past 14 days:

```
select *
from table(snowflake.information_schema.warehouse_load_history(
                  date_range_start => dateadd('day',-14,current_date()),
                  date_range_end   => current_date(),
                  warehouse_name   => 'JRYAN_VWH'))
where (avg_running > 0 or
      avg_queued_load > 0 or
      avg_queued_provisioning > 0 or
      avg_blocked > 0)
order by start_time desc;
```

4. Snowflake credit usage by month:

```
select wmh.warehouse_name
,     to_char(start_time,'YYYY-MM') as day
,     sum(credits_used)
from snowflake.account_usage.warehouse_metering_history wmh
where wmh.start_time > dateadd(month, -2, current_date())
group by wmh.warehouse_name
,     to_char(start_time,'MON-YYYY')
order by 1,2;
```

5. Warehouse credits (from Information Schema):

```
select date_trunc(month, start_time) as usage_month
  , sum(coalesce(credits_used, 0.00)) as total_credits
  , sum($compute_price * coalesce(credits_used, 0.00)) as billable_warehouse_usage
from table (
    information_schema.warehouse_metering_history (
        date_range_start => date_trunc(month, dateadd(month, -3,
current_timestamp)),
        date_range_end   => dateadd(second, -1, date_trunc(month,
current_timestamp))
      )
  )
group by 1
order by 1;
```

6. Storage Usage:

```
select date_trunc(month, usage_date) as usage_month
  , round(avg(storage_bytes)/power(1024, 4), 3) as billable_database_tb
  , round(avg(failsafe_bytes)/power(1024, 4), 3) as billable_failsafe_tb
  , round(avg(stage_bytes)/power(1024, 4), 3) as billable_stage_tb
  , $storage_price * (
    round(avg(storage_bytes)/power(1024, 4), 3)
    + round(avg(failsafe_bytes)/power(1024, 4), 3)
    + round(avg(stage_bytes)/power(1024, 4), 3)
    ) as total_billable_storage_usd
from snowflake.account_usage.storage_usage
where usage_date >= date_trunc(month, dateadd(month, -3, current_timestamp))
and usage_date    < date_trunc(month, current_timestamp)
group by 1;
```

7. Clustering credit usage:

```
select *
  from table(information_schema.automatic_clustering_history(
   date_range_start=>dateadd(d, -30, current_date),
   date_range_end=>current_date));
```

8. Snowpipe credit usage:

```
select *
from snowflake.account_usage.pipe_usage_history
order by pipe_name, start_time;
```

9. This query identifies queries which process large volumes of data. You can use it to detect large or complex queries running against a small virtual warehouse. The queries identified here could be either candidates for tuning (to reduce the data volumes processed), or moving to a larger virtual warehouse (if run on a regular basis).

```
select  query_type
,     warehouse_name
,       warehouse_size
,       round(total_elapsed_time/1000,0) as total_elapsed_seconds
,     CASE
      WHEN BYTES_SCANNED >= POWER(2, 40) THEN TO_CHAR(ROUND(BYTES_SCANNED /
POWER(2, 40), 1)) || ' Tb'
      WHEN BYTES_SCANNED >= POWER(2, 30) THEN TO_CHAR(ROUND(BYTES_SCANNED /
POWER(2, 30), 1)) || ' Gb'
      WHEN BYTES_SCANNED >= POWER(2, 20) THEN TO_CHAR(ROUND(BYTES_SCANNED /
POWER(2, 20), 1)) || ' Mb'
      WHEN BYTES_SCANNED >= POWER(2, 10) THEN TO_CHAR(ROUND(BYTES_SCANNED /
POWER(2, 10), 1)) || ' K'
      ELSE TO_CHAR(BYTES_SCANNED)
      END AS BYTES
, q.*
from snowflake.account_usage.query_history q
where execution_status = 'SUCCESS'
and bytes_scanned >= 1024 * 1024 * 1024
-- and start_time > dateadd(month, -1, current_date())
limit 500;
```

ABOUT SNOWFLAKE

Snowflake is the only data warehouse built for the cloud, enabling the data-driven enterprise with instant elasticity, secure data sharing, and per-second pricing across multiple clouds. Snowflake combines the power of data warehousing, the flexibility of big data platforms, and the elasticity of the cloud at a fraction of the cost of traditional solutions. Snowflake: Your data, no limits. Find out more at **snowflake.com**