

GETTING STARTED WITH SNOWFLAKE

BEST PRACTICES FOR LAUNCHING YOUR SNOWFLAKE PLATFORM

TABLE OF CONTENTS

Introduction.....	3	Managing Database Objects.....	15
Defining a Chargeback Model	4	Repeatable Process	15
Expenses.....	4	Workspaces.....	16
Configuring Your Snowflake Account	5	How Will Your Projects Transform Data?.....	19
Data Retention	5	Views.....	19
Timezone.....	5	Streams and Tasks.....	20
Security.....	6	Spark	20
Connection Performance.....	6	Monitoring Your Account.....	21
Cost Savings.....	7	Monitoring	21
Managing Access	9	Alerting	22
When to use OAuth	10	Auditing.....	22
When to use SAML2.....	10	Get started.....	23
When To Use SCIM vs phData TRAM.....	10	phData is Here To Make Your Life Easier	23
Handling Service Accounts	11		
Loading Data.....	12		
Internal vs. External Stages.....	12		
Using a Storage Integration.....	12		
Snowpipe vs. Inserts vs. Copy Into	12		
Data Pipelines.....	14		

CONGRATULATIONS!

You have a shiny new Snowflake account, and business units at your company are lining up to be onboarded. If all goes well, you'll soon be scaling up and beginning to integrate with existing business processes with built-in expectations.

However, if there's one thing we've learned from years of successful cloud data implementations here at phData, it's the importance of defining and implementing processes, building automation, and performing configuration even before you create the first user account.

This isn't meant to be a technical how-to guide — most of those details are readily available via a quick Google search — but rather an opinionated review of key processes and potential approaches.

Each step of the way, we'll explore the options available to you and how they impact your operating costs, complexity, security, and performance — allowing you to get the most out of the Snowflake platform and deliver results back to the business.

DEFINING A CHARGEBACK MODEL

Unlike traditionally licensed on-premises data solutions, Snowflake operates with a flexible pay-as-you-go model, allowing you to create an account and start using it without delay. However, without the proper planning to ensure governance and visibility around utilization, this model can also make it easy to run up a significant bill as multiple business units ask for access.

To combat this, decide in advance how to pay for Snowflake credits. For example, will each project pay for its usage? How is that funded after the project releases?

Furthermore, Snowflake gives discounts on the volume of purchased credits. That means making consolidated purchases can save you money, but calculating how you distribute those costs to business units or projects is up to you.

EXPENSES

Expenses to account for with Snowflake include:

- Warehouses
- Snowpipe
- Materialized Views
- Cloud Services
- Data Transfers
- Storage

While it is possible to minimize or prevent expenses in some areas, you should still plan for all potential expense sources. And whatever model you choose, you will need to track these expenses within Snowflake to determine how much consumption has occurred and how to charge it to the right budget.

More on this topic on [page 21](#); but for now, keep in mind that the simplest method is to create a naming convention for database objects that allows you to identify the owner and associated budget.

It's possible to have more elaborate methods to store metadata in another table or even (cringe) in a spreadsheet; but whatever you do, you'll want it to remain accurate and easy to maintain over time.

Also note that this metadata could be used to create other reports that help individual business units optimize their cost and performance — a possibility we'll explore in more depth when we talk about monitoring.



TIP You will need metadata to associate charges to the correct budget.

CONFIGURING YOUR SNOWFLAKE ACCOUNT

Snowflake has many parameters to configure default behavior. Most will likely work for your requirements without changing them, but there are a few properties that deserve special attention from enterprise customers.

In particular, be sure to evaluate Snowflake policies that influence:

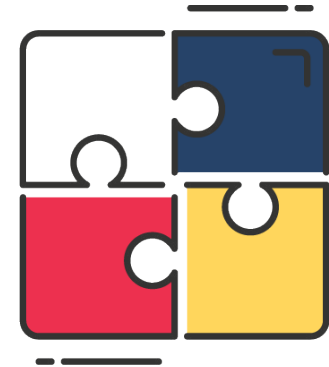
- Data Retention
- Timezone
- Security
- Connection Performance
- Cost

DATA RETENTION

Parameter: `DATA_RETENTION_TIME_IN_DAYS` (90)

Data retention time is how long Snowflake will retain historical views of your data.

The default data retention time is one day, but if you are paying for the Enterprise Edition, you will likely want this to be 90 days. The extended period will allow you to perform Time Travel activities, such as undropping tables or comparing new data against historical values.



TIP For cost savings, you can set this value for each database and choose to have non-production data stored for fewer days. One day is usually adequate for development use.

TIMEZONE

Parameter: `TIMEZONE` (etc/UTC)

Snowflake will present time-related values with the timezone in your configuration.

The default timezone used by Snowflake is for Los Angeles. However, you should evaluate what works best for you. Some companies set the timezone to their corporate headquarters' timezone, while others use UTC.



TIP It is possible to set this both at an account level and at a user level; however, consumers should always request time-related fields with a timezone to avoid being reliant on this default.

SECURITY

Storage Integrations

Parameters:

- `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION` (True)
- `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION` (True)

A storage integration is a secure means of creating connectivity between Snowflake and your cloud storage provider. You will need at least one if you create external stages to load data.

External stages should require storage integrations. By setting `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_CREATION` and `REQUIRE_STORAGE_INTEGRATION_FOR_STAGE_OPERATION` to “True,” you can prevent the exposure of access tokens or secret keys to Snowflake users.



TIP Do not create external stages without storage integrations.

Network Policies

Parameter: `NETWORK_POLICY`

A network policy defines a list of valid network locations for user connections. Preventing access from unwanted networks is vital to protecting your Snowflake account.

You can set an account-wide network policy, as well as configure network policies at a user level.

Network policies are an area of account setup that is difficult and time consuming. You will frequently run into issues where integrations will come from cloud providers with broad ranges of potential network locations (CIDR blocks). You may also have users connect through a VPN and find that split-tunnel connections originate from a user’s home network rather than the corporate network.

Work with your network team to find the right CIDR blocks but expect to make multiple adjustments to this over your first few weeks.

Snowflake will not let you activate a network policy that would lock you out. However, if your admins are connecting over a VPN, you may find that your IP address shifts day-to-day. If your account admins are remote and do not have a guaranteed known IP address, make a fail-safe network policy that will allow the account admins to connect regardless of their network location. Once you are confident that you will not get locked out of your account, remove the fail-safe network policy.



TIP Be cautious when creating your network policies — especially if you are connecting over a VPN.

CONNECTION PERFORMANCE

Parameter: `CLIENT_METADATA_REQUEST_USE_CONNECTION_CTX` (True)

The “`CLIENT_METADATA_REQUEST_USE_CONNECTION_CTX`” parameter is a bit technical, but it boils down to reducing the amount of information that JDBC and ODBC connections pull when they connect. Set this to “True” at an account level and only set it to “False” for those users who require account-wide metadata. This property is strictly a performance optimization, but it’s an important one as it can make short-lived connections much faster, especially as your catalog of databases and schemas grows.

COST SAVINGS

Users

Parameters:

- LOCK_TIMEOUT (60)
- STATEMENT_TIMEOUT_IN_SECONDS (600)
- STATEMENT_QUEUED_TIMEOUT_IN_SECONDS (30)

In your Snowflake account, you will have system users and human users. You might choose different default settings for each, but regardless, it is essential to consider which configuration will work best for your organization.

Lock timeout determines how long a query will wait for a resource that is locked. For most users, if you are waiting for more than 60 seconds, there is likely an issue, and there is no reason to waste further warehouse credits.

A statement timeout prevents poorly optimized queries from running all weekend and racking up substantial charges. (Yes, this happens.) For human users who are waiting for results, ten minutes is a reasonable timeout. A user still has the option of increasing this timeout for a given session if they know their query will take an exceptional amount of time. Ten minutes may not be long enough for system users, but some reasonable limits will prevent multi-day queries from creating unexpected costs.

Warehouses that are overworked have queue times that slow queries down. If you do not have a queue timeout, users may sometimes repeatedly queue new queries until your warehouse is working for many hours to clear the queue. Thirty seconds is a good default for human users; if you find that queries are regularly queueing, consider making your warehouse a multi-cluster that scales on-demand.

Warehouses

Parameters:

- RESOURCE_MONITOR
- AUTO_SUSPEND (60)
- WAREHOUSE_SIZE (XS)
- MIN_CLUSTER_COUNT / MAX_CLUSTER_COUNT (1/1)

Warehouse credit consumption can make up most of your invoice, so optimizing for cost can have a huge benefit.

RESOURCE MONITORS

A resource monitor is a Snowflake object that observes credit usage for warehouses and can alert your account administrators or even suspend the warehouses. You can configure the monitors with a finite number of credits and either not refresh or refresh at a frequency of your choice.

The first and most important thing you can do to manage your costs is to create a resource monitor for each budget and associate it to the appropriate warehouses. Whether your budget is tied to business units or projects — or even your entire company — ensuring that a warehouse does not use excessive credits is essential.



TIP Only Account Administrators who have opted in can see alerts on resource monitors.

AUTO SUSPEND

Ingesting and transforming data will utilize warehouses for fixed periods, and then they won't be needed again for an hour or more. Therefore, there's no reason for these warehouses to continue running for another ten minutes. If a warehouse resumes once per

hour, you could save roughly three and a half hours worth of credits per day by suspending quickly!

In many cases, setting your warehouses to auto suspend after 60 seconds works well. The main exception where you should avoid auto-suspending quickly is when you have frequent traffic over a period and you want to keep a warehouse's local disk cache populated.



TIP Create warehouses for each specific purpose, allowing you to configure them optimally for that purpose.

WAREHOUSE SIZE

The "t-shirt size" of your warehouse (XS, S, M, L, XL) will significantly impact your credit usage. Sizing your warehouses to perform well and remain cost-efficient is always a challenge, and it's a bit more art than science. But while you may need some trial and error, there are some rules of thumb for picking your starting point.

Begin with a size that roughly correlates to how much data the warehouse in question will be processing for a given query:

XS: Multiple megabytes

S: A single gigabyte

M: Tens of gigabytes

L: Hundreds of gigabytes

XL: Terabytes



TIP A warehouse for data ingestion is generally two sizes smaller than a warehouse for transformation or analytics.

CLUSTER COUNT

If your warehouse has to serve many concurrent requests, you may need to increase the cluster count to meet demand.

For most transformation and ingest warehouses, you can leave the cluster default of one minimum and one maximum. However, for analytics warehouses, you may need to scale for usage.

As a rule of thumb, you will need a max cluster count of roughly one for every ten concurrent requests. So, if you expect 20 concurrent queries and do not want them to queue, a multi-cluster count of two should be adequate. In this case, the max cluster count should also be two.

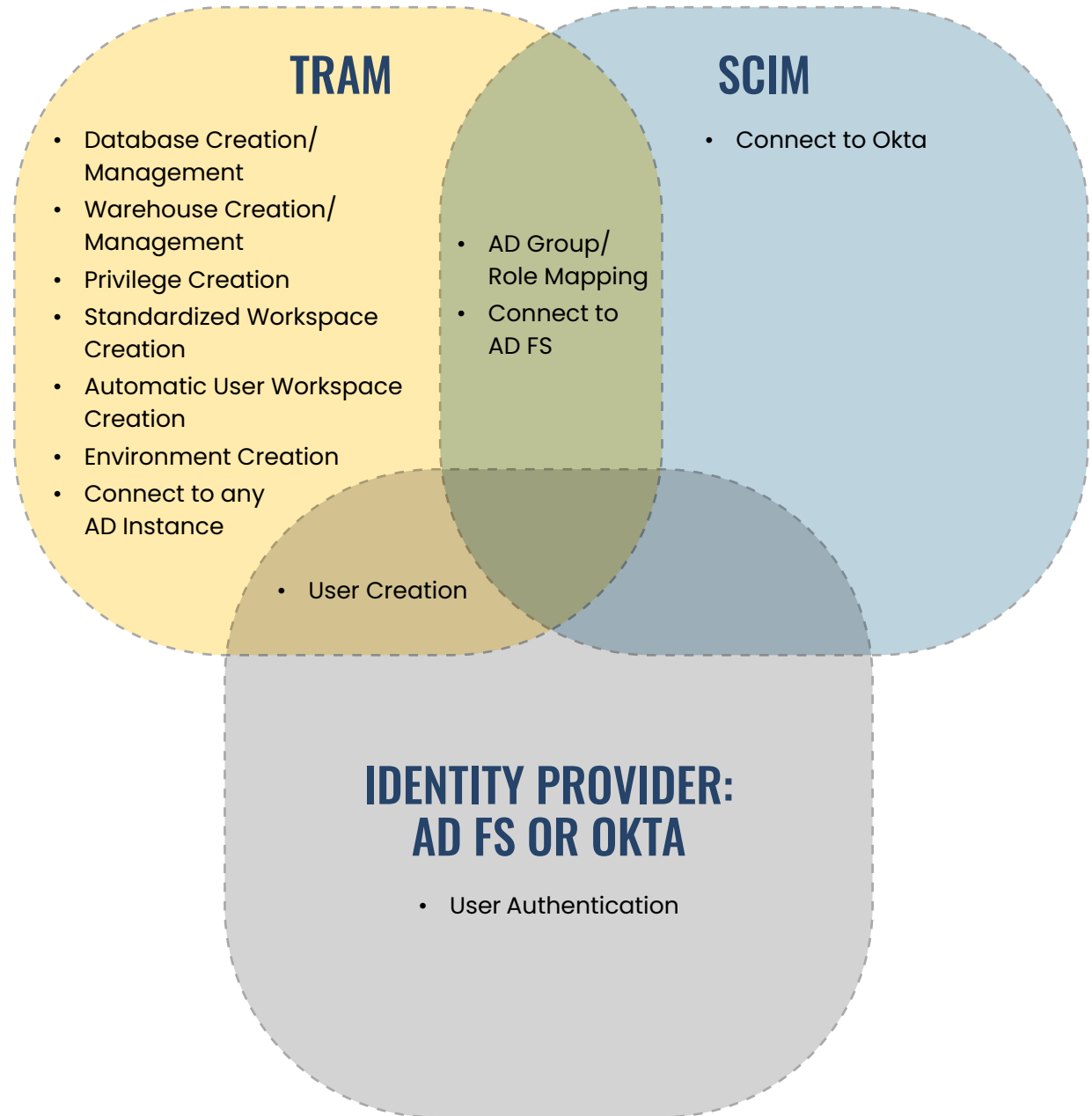
It's always best to start small and scale up as needed, so setting proper configurations for the scenario above needs to be taken into consideration. Starting with a minimum cluster count of one will provide some cost savings but as the number of concurrent queries ramp up this may cause performance issues. If performance issues are encountered, increasing the minimum cluster count higher to two should be considered.

MANAGING ACCESS

Most enterprises have Active Directory or another OAuth or SAML2 provider and want users to log in with their existing accounts to receive access to resources based on their roles. To this end, Snowflake security integrations allow you to specify an external provider to authenticate or authorize users.

OAuth and SAML2 will allow existing users to log into Snowflake – but for access only. For tasks such as creating users or mapping existing groups to roles, you'll need to look to SCIM (System for Cross-domain Identity Management), and to tools like [phData Tram](#).

See the diagram for an idea of which technologies work best for your organization.



WHEN TO USE OAUTH

OAuth relies on cryptographic keys and an external provider for authentication and is used to provide authorization to use Snowflake. For Snowflake, this is a situation where third-party software is involved.

A user's client software initially authenticates with the identity provider. It then uses a token on all calls to Snowflake until that token expires, at which point, the client software either refreshes the token or forces the user to authenticate again.

The software you might use OAuth with includes:

- Tableau
- Power BI
- Looker

If so, you will need an OAuth provider like Okta, Microsoft Azure AD, Ping Identity PingFederate, or a Custom OAuth 2.0 authorization server.

WHEN TO USE SAML2

SAML2 is used to authenticate users logging into the Snowflake UI, for Snowflake connectors, or ODBC/JDBC connections that rely on credentials.

If you use Active Directory Federation Services (ADFS) or Okta, you may use the "basic" option and configure a SAML Identity Provider. This method requires updating an account-level parameter named `SAML_IDENTITY_PROVIDER`. However, the more standard security integration syntax is replacing the identity provider parameter method.



TIP Even if you start with `SAML_IDENTITY_PROVIDER`, you can migrate to a security integration later using the system function `migrate_saml_idp_registration()`.

For greater detail, see the [Snowflake documentation](#).

WHEN TO USE SCIM VS PHDATA TRAM

SCIM manages users and groups with Azure Active Directory or Okta. phData Tram manages users and groups using any Active Directory instance, or through text file configuration.

PHDATA TRAM MAKES MANAGING SNOWFLAKE USERS AND PROJECTS SIMPLE

Tram is a software tool designed by phData to streamline the creation and management of project resources within Snowflake rather than having to handle provisioning manually. This includes users, roles, schemas, databases, and warehouses.

With phData Tram, you can:

- Quickly ramp up new projects through the reuse of information architecture
- Speed user onboarding and simplify access management
- Quickly apply new changes to Snowflake
- Manage hundreds or thousands of project environments, groups, and workspaces automatically
- More easily create, verify, and reuse complex information architectures

Additionally, Tram facilitates your information architecture.



HANDLING SERVICE ACCOUNTS

For most companies, directory services hold human users; but for systems that migrate or consume data, you need service accounts.

In most cases, you should use RSA key pairs and rotate those keys every 90 days. Some systems will not support key pairs (or cannot do it securely), and you will need passwords associated with the system accounts. As with key pairs, you should rotate these passwords frequently.

Creating a process for doing the rotation is the biggest obstacle to success for service accounts. Begin planning early on how the rotations will work. Consider:

- Who will update the new keys or passwords?
- Who generates them?
- How are they transmitted?
- Can more than one service use the same account or password?

(Note however that this merely outlines the basic thought process here. The actual implementation is typically very complex. For further information or assistance, phData recommends having deeper conversations with our technical experts.)

LOADING DATA

Now there is a gameplan in place for handling cost attribution, configuration optimization, and access management. But before teams can use Snowflake, you'll also need a well thought-out strategy for ingesting data. Without one, people will no doubt find ways to get data into the platform, but they will ultimately waste a lot of credits and time doing it poorly.

In practice, you may not be able to use the same ingestion tool for all of your needs. Establishing a core technology that satisfies most requirements, paired with a process to review and approve new technologies as needed, is therefore critical to keeping projects moving forward, managing costs, and keeping your data secure.

INTERNAL VS. EXTERNAL STAGES

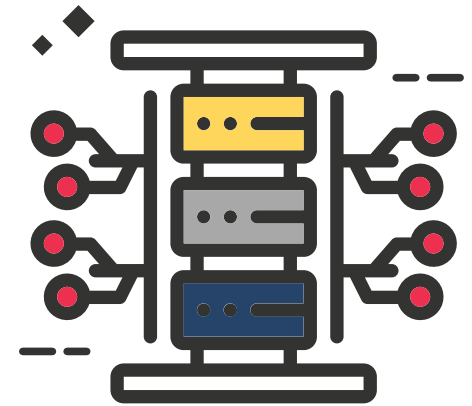
The first choice for you to make when loading data is whether to use internal or external stages.

An internal stage utilizes cloud storage managed by Snowflake, whereas an external stage is one you manage yourself. To determine which is right for you, consider these questions:

- Will I have processes which, for cost or performance reasons, would need the raw data directly from cloud storage?
- Do I have unique encryption requirements around data at rest?
- Do I have tools or technologies that will only integrate with Snowflake through my cloud storage?

If you answer “yes” to any of these questions, you will need cloud storage, such as Amazon AWS’s S3 or Azure Data Lake Storage. This decision may influence whether you choose to have your Snowflake account backed by Azure or AWS, in addition to which region you select (as there are performance and cost considerations when moving data between regions).

You may have a strategy that mixes internal and external stages, which is fine – if the choice is purposeful. Internal Stages tend to be easy to use and manage, but do limit options to consuming data strictly through Snowflake.



USING A STORAGE INTEGRATION

If you choose to use external stages, you should create storage integrations to interact with them. While it is possible to make an external stage without them, there are security risks in doing so, as Snowflake users with enough access may see them.

SNOWPIPE VS. INSERTS VS. COPY INTO

When loading data into Snowflake, the very first and most important rule to follow is: *do not load data with SQL inserts!*

Loading small amounts of data is cumbersome and costly:

- Each insert is slow — and time is credits.
- It creates poorly partitioned tables, which will make queries slow.

Instead, always load data in bulk by using either Snowpipe or the SQL “COPY INTO” statement, which is how all Snowflake-approved connectors work.

File Sizes

So, you should load data in bulk—but what qualifies as bulk, and why does it matter?

Snowflake stores data in micro-partitions of up to 500 MB. Each of these stores metadata needed to optimize query performance. By having small micro-partitions, or micro-partitions that aren't homogeneous, your queries will read additional partitions to find results. These unoptimized partitions will return results slower, frustrate consumers, and increase credit consumption.

Knowing this, you want to have data prepared in a way to optimize your load. It might be tempting to have massive files and let the system sort it out. Unfortunately, having excessively large files will make the loads slower and more expensive.

Aim for 100-megabyte files. Less than ten megabytes or more than a gigabyte, and you will notice suboptimal performance. Snowflake publishes file size guidelines, and phData recommends checking periodically to see if they have changed.

File Format

If you want pure performance, compressed CSVs load fastest and use the least credits. But there are other considerations if applications other than Snowflake pull files from your cloud storage.

The CSV format only supports structured data, which can be a nonstarter in some situations. In cases where CSVs may be

infeasible, Parquet — a semi-structured format used by Spark and other applications that consume data — is a reasonable option.

You may be limited to the formats that your data sources produce. There is no “wrong” choice when it comes to file format, but having a policy may help with performance and help developers with common patterns.

Snowflake Connectors

For accessing data, you'll find a slew of Snowflake connectors on the Snowflake website. You can use whatever works best for your technology (e.g., ODBC, JDBC, Python Snowflake Connector), and generally, things will be okay. Be sure to test your scenarios, though. Some connectors, like the one for Python, appear to use S3 for handling large amounts of data, and that can fail if your network does not allow the connectivity.

And once again, for loading data, do not use SQL Inserts. You will find options for most major data migration tools and technologies like Kafka and Spark.

Optimizing Data for Use

There is no need to optimize your data prematurely. Load it as-is and see how queries perform. If they are slow, check the query profile to see whether queries are reading many micro-partitions. If they are, you have options.

Micro-partitions help queries run faster when sized well, but you can also influence performance by making the frequently used columns homogeneous in partitions. Sending files from your source systems pre-sorted by frequently filtered-upon columns may help optimize partitions.

If you keep your data volume low and your file sizes small, you may not be able to influence the micro-partitions in this way. But don't despair: you still have an option.

After the data is initially loaded, but before end-users query it, you can periodically optimize the table with a task. The task would use a stored procedure that performs a “create table as select” to generate a temp table which is sorted by columns that are filtered on, then uses an “alter table swap” statement to plop in the optimally partitioned table.

Additionally, Snowflake has the concept of clustering keys. This can help performance with huge tables, but isn't meant for small tables and can hurt speed in some cases. Use with caution, and test before committing to using them.

DATA PIPELINES

“Data pipeline” means moving data in a consistent, secure, and reliable way at some frequency that meets your requirements.

Data pipelines can be built with third-party tools alone or in conjunction with Snowflake's tools.

Snowpipe

If you can get data into an external stage, you can get the data into Snowflake using Snowpipe.

For technical details, search for Snowpipe how-to guides online. But know that there are some significant decisions to make before using Snowpipe:

- How will data make it reliably and securely to your cloud storage?
- How will you create the schema for data loads?
- How will you handle changes to the schema of the incoming data?

ESTABLISHING SCHEMAS AND HANDLING SCHEMA DRIFT

If you are using Snowpipe, you might maintain schemas manually. This may work if you expect your schemas to be static, but the safer approach is to have a plan for detecting and adjusting to schema changes.

One way is to load data in a semi-structured format that inherently does not have a schema. As data matures through the transformation life cycle, tasks build curated or conformed schemas by mapping fields from the schemaless raw data layer.

Data will load without failure when schemas change, but you will still need to change the view or task that maps data to account for the source fields being different. The biggest drawback to this approach is that you may not realize that the incoming data schema has changed for some time because nothing will break.

Alternatively, for a structured data approach, there are two tools that phData provides to customers that can assist with establishing schemas: **Streamliner** and **SQLMorph**.

STREAMLINER

Streamliner is an open source tool that crawls a source database and creates a schema within Snowflake.

SQLMORPH

SQLMorph is a free SaaS application that can translate SQL from one dialect to another.

Third-Party Products

Many third-party products can migrate data and manage schema drift. A word of caution: some applications are new to the Snowflake arena, so verify that they will work well for you.

phData has partnerships with Qlik, HVR, Fivetran, and StreamSets. We're able to help customers identify the tooling that best fits their particular needs.

Whatever products you choose, be sure to establish a process to manage change over time, and handle failures in your data pipelines.

MANAGING DATABASE OBJECTS

Anything you make in Snowflake is a database object.

Make sure to have a plan to allocate the sets of objects that comprise a given project, so you can track the expenses and resources around them as a unit, which is a critical part of managing your Snowflake budget.

REPEATABLE PROCESS

It is possible to look up the necessary SQL syntax to create a table or establish a role, and to simply use the Snowflake UI to make objects. This manual approach works well when you want to make a single object. However, this is not a good practice overall.

Take role creation, for example. As a general best practice, you should grant all custom roles to the SYSADMIN role; otherwise, you would end up with roles floating around that cannot be managed by the people who manage the account. Beyond being granted to the SYSADMIN role, you may need to enforce your custom role hierarchies so people in charge of a given project can see the objects created by people working on that project. You could always write a document that specifies these steps and rely on people following them to create Snowflake roles correctly; but in practice, you will eventually have issues. Fortunately, automation can make this process far less manual, time-consuming, and prone to error.

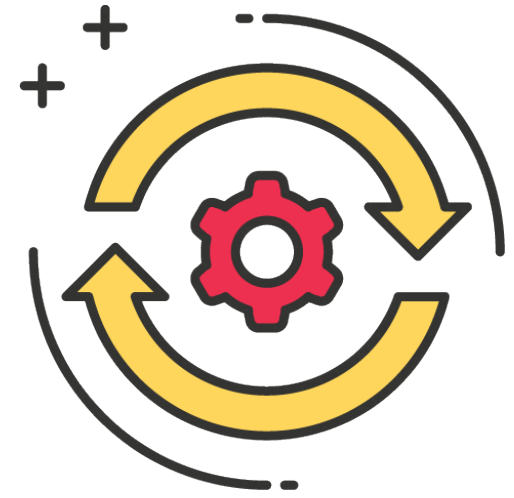
Stored Procedures

One way to help people properly create new roles within the hierarchy is to create a stored procedure that can create roles with the requisite grants and ownership on behalf of the user without actually permitting the user to create the role on their own.

While a little syntactically clumsy, using stored procedures is easier and more cost-effective than trying to fix role hierarchies by hand later.

You might end up with one fancy stored procedure that takes in multiple parameters to allow admins to make roles for more than one project. The stored procedure would verify that they have access to do the creation. Or, a more straightforward but verbose approach, might be one stored procedure per project that only admins of that project can access.

Whatever you design, find a means to make these roles in a repeatable, secure, and correct way within your development process.



Tram

phData offers customers a free tool called [Tram](#), to create Snowflake users, databases, warehouses, schemas, and other objects in a repeatable and secure fashion.

Tram brings the concept of a “workspace template” — a collection of resources which you can define, and then utilize to allocate a new project efficiently.

By establishing a naming convention and association within its configuration, you can manage the workspace resources more quickly and consistently.

WORKSPACES

Workspaces are a collection of resources needed by a project, funded by a single budget and with tightly coupled, interrelated objects.

Having the ability to allocate a workspace as required will significantly reduce the time it takes for new projects to be productive, and will make adhering to standards much easier.

Whether you have Tram or not, having the concept of a workspace and the automation to support them is valuable because they provide you:

- A familiar, repeatable pattern and process that meets standards
- Time-saving automation that prevents errors and omissions
- Enforcement of proper naming and metadata structures that allow you to manage and monitor your projects
- Simplified Role Management
- Facilitation of Continuous Integration and Continuous Delivery (CI/CD)

Metadata

One way or another, you’ll need metadata on objects within Snowflake so that you can associate them with budgets and users. You need this to manage credits effectively, so if something goes wrong, you know who to contact.

The simplest way to introduce metadata is to create a naming convention that tracks the business unit and project associated with an object. Two other useful attributes to include might be the environment the object is used for and the purpose of that object. The purpose attribute distinguishes objects of the same type that serve different purposes.

Depending on your requirements, there are also more elaborate possibilities involving custom tables and stored procedures to track other metadata elements that you may need.

One consideration to track when defining your process is that an object may initially be created and managed by one group, but then later transferred to another.

Standardized Structure and Naming

An example format may include environment label, business unit, project, and purpose joined with underscores. The order of these elements may impact your developer tools’ code-complete feature and drop-down menus within the Snowflake UI, so be sure to choose an order that is convenient and sensible to you.

Continuous Integration and Continuous Delivery

With Snowflake, having multiple environments for a project is easy if you have the automation to generate workspaces.

Having a single production database might seem simpler than implementing CI/CD in the short term. However, there are a number of issues that make this approach infeasible for larger organizations:

- **Security:** Always limit the number of people who can change and view data in production
- **Stability:** Making untested changes in production is going to break something eventually (and probably often)
- **Rollback:** If things go wrong, even though you tested, being able to revert changes quickly can save money and time
- **Accountability:** Having an approval process for the promotion of changes holds people to a higher standard
- **Auditability:** Processes produce artifacts and can be used to track how things went right or wrong
- **Testing:** Someone verified that everything worked before deploying it to production

For each project, create a development workspace for each person doing work. They can test their changes and verify their work before committing the DDL into source control for a build tool to promote code to a shared non-prod environment.

Then, after testing the changes in your non-prod environment(s), promote them to production to release it to your consumers.

There are great articles on CI/CD and tools to help you implement the process successfully. Find what works for you and your organization, and create a plan to promote changes into production.

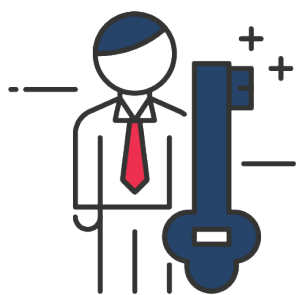
Some tools to check out related to doing CI/CD with Snowflake:

- Flyway
- Sqitch
- Liquibase
- Snowchange

Roles and Security

When defining standards and processes around your workspaces, consider roles and security early in the process. By creating a consistent pattern, users will develop expectations on how a given role will behave based upon naming conventions. You will then have less confusion and a lower likelihood of elevating someone's access unnecessarily.

Unlike some other databases where users who have multiple roles can see all tables that those roles grant them, Snowflake users can only assume one role at a time, and can only see the resources that one role allows.



LOOKING FOR MORE INFORMATION ABOUT ROLES AND SECURITY?

Check out this [blog post](#).

This role design has both positives and negatives. For example, it's great for situations where regulations prohibit the combination of certain data sets. However, it can be tricky when your former model for roles is not compatible with this structure. You might end up with thousands of roles that hold every permutation of role combinations in an attempt to imitate your original role design.

For each workspace, plan out your role hierarchy. Plan for roles that are not granted directly to users, but instead granted to other roles for other projects.

ROLES AND SECURITY EXAMPLE

A simple hierarchy for a non-production environment might look like this:

- Administrator
 - CICD
 - Developer
 - ◊ Tester
 - accounting_view
 - hr_view

In this example, two roles grant read access to two different views. We may grant one or both of those roles to another project that should read data from this workspace's views, but those roles cannot see or interact with our tables or other internal objects.

The Tester role is granted both view roles in this example, so they could assume either as needed. We grant the tester role to the developer role, plus we grant the developer role the ability to write to any table or change other objects within the project database.

We create the CI/CD role to modify anything in the database that your CI/CD process needs.

We grant the developer role to the administrator role, and we also grant the ability to create or drop objects like schemas to the administrator.

Your production environment would only have the Administrator and the CICD roles.

Note: the SYSADMIN role has been omitted from the hierarchy for simplicity.

HOW WILL YOUR PROJECTS TRANSFORM DATA?

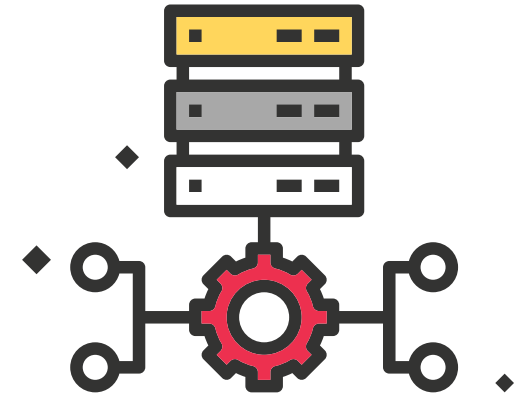
You now have a plan to get data into Snowflake, but there is still more work to do for that data to be effectively consumed and utilized by all the right teams.

VIEWS

The simplest means of transforming data is to put a view over it. There are multiple types of views, each of which has its own benefits and drawbacks.



TIP A view calling another view roughly doubles the compilation time of the query, even when pulling data from the result cache.



Materialized

Materialized views are very restrictive in terms of what you can do with them, but their benefit is performance.

Materialized views use credits and that frequently changing data can run up the meter. So be conscientious of when you use them and of how they impact your budget. It's also worth noting that you can't attach a resource monitor to a materialized view. If you do not have custom monitoring and alerting from your operations team, you might not know how much you are spending until you run out of credits.

Non-materialized

Non-materialized views are your standard, average view with some optimizations to help with performance. The caller's warehouse will pay the bill for any transformation done using this view.

Secure

Secure views are specialized to avoid specific vulnerabilities. You can read more about them on the [Snowflake website](#), but it is important to note that they are slow, as they cannot utilize some optimizations that other views are allowed to perform.

Before using a secure view, consider whether you can use another type of view or do most of the transformation work outside of the secure view.

STREAMS AND TASKS

Streams help with Change Data Capture (CDC) feeds and allow you to handle more than simple append-only data. They can trigger a task to run SQL when they have new data, providing an opportunity to move and transform it.

You do not need to have a stream to use tasks, in which case your root task would trigger on a schedule.

Tasks are hierarchical and can work together. A top-level task may move data into one table. A dependent task might transform the combination of multiple tables into a new temporary table that you finally swap in as a curated table.

While there are a lot of possibilities, keep in mind that tasks use credits.

If you migrate data from an existing database, note that phData offers customers a tool called [SQLMorph](#) to automate the process of translating SQL dialects.

SPARK

Either before moving data into your external stage or after the data is loaded, new datasets can be created by combining multiple files using external systems such as a Databricks Spark application.

MONITORING YOUR PLATFORM AND PIPELINES

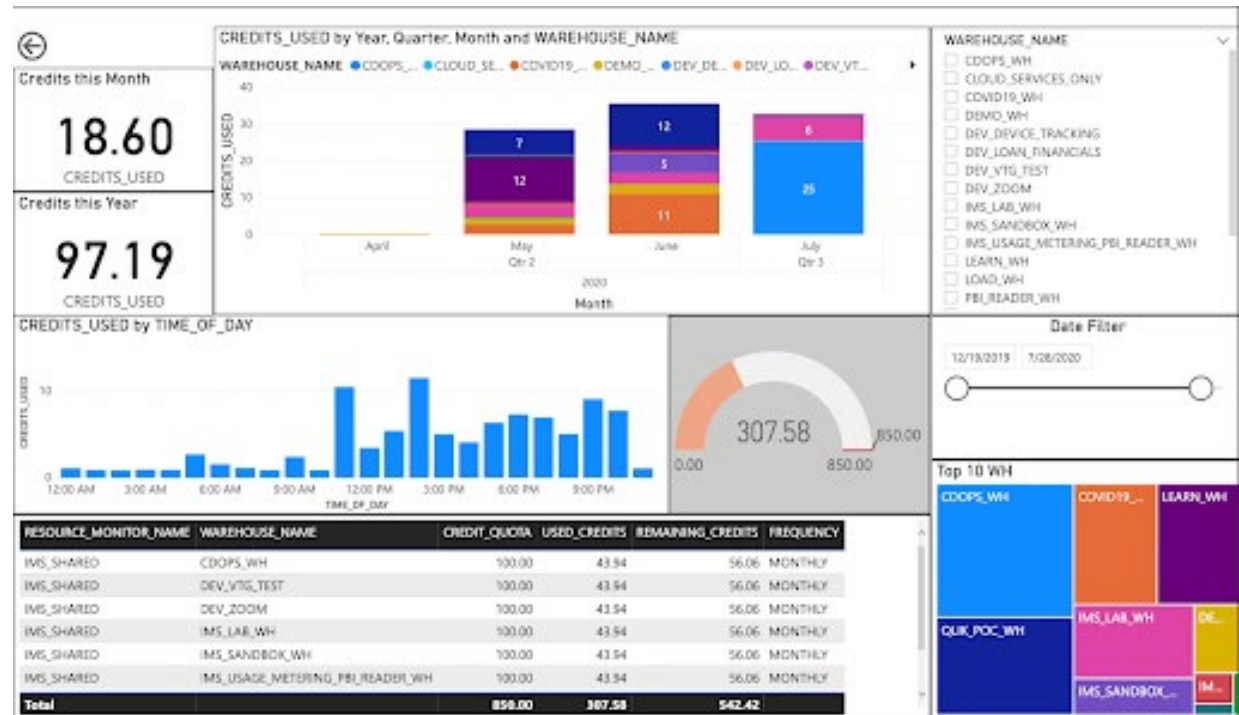
You have data flowing, and everything is great—or so you think! But then you realize a data pipeline stopped working two days ago, and an out-of-control query (which has apparently been running since last weekend) has eaten up the entire budget for a small project.

The point? All systems tend toward entropy; things go wrong. But without monitoring, nobody is even aware until there's a customer complaint or the next budget review happens.

Don't let this happen to you! Here is what phData recommends for monitoring.

MONITORING

Snowflake provides some essential account-level usage information and a dashboard, but that dashboard is only useful if someone is looking at it. For building custom monitoring, several Snowflake views have metadata about your account usage.



You will quickly run into two issues:

- First, the only role that can see everything useful is the ACCOUNTADMIN role, and giving out access to this is like handing the nuclear codes to a toddler.
- Second, Snowflake does not organize information by your company's budget groupings. Project X has a budget, and Project Y has its budget. Although you may have bought the credits in bulk for both projects to save money, you will presumably want to deduct credits from specific budgets.


Custom metadata associated with workspaces enable you to create more holistic reports such as understanding the spending at the regional or business-unit level rather than just at the project level. Or maybe you categorize your projects and want to understand which categories utilize the most storage.

Planning out your monitoring needs can help you manage your budgets in this way. But there's also another significant benefit: good monitoring tools can help you identify areas to improve.

For example, having a list of the top ten warehouses that are consuming credits, and then looking at the top ten queries for each might present improvement opportunities. Cost optimization is also why it is valuable to have warehouses for each purpose: it makes it easier to identify these situations.

But no matter how simple or complex your needs, be sure to make a plan to track your daily and monthly usage.

This data is valuable to the business units using your platform, which otherwise wouldn't have access to aggregate it. With the proper design, you can expose details projects need to identify the warehouses, queries, and processes that can be optimized.



TOO MUCH WORK?
phData has a Cloud DataOps offering that will monitor your platform and data pipelines for you. They operate 24x7 and keep your data moving.
[Find out how.](#)

ALERTING

Monitoring and alerting are closely related, with one key distinction: monitoring requires somebody to see what is happening, whereas alerts will send an email or other notification to let the right people know there is an immediate issue.

Built-in Alerts

Snowflake comes with some built-in alerting; however, it's only available to people with an ACCOUNTADMIN role — and only if those people opt in.

You won't be giving out the ACCOUNTADMIN role to many people, so the project members who need to know about an issue with their data pipelines will not know until you tell them. You may therefore want to devise a custom solution on top of the base Snowflake offering, in order to ensure that the people associated with a workspace resource are notified of issues in a timely fashion.

AUDITING

Snowflake tracks 365 days of most audit-type information. If you need more, you may need to come up with a custom solution to store history beyond that period.

And even if you don't have compliance reasons to store everything, having data aggregated by day may allow you to create usage forecasts if you have access to data science resources. (And if you don't, that's another area where [phData can help.](#))

GET STARTED

Now that you've worked your way through all the critical decisions that need to be made upfront, you are, at long last, ready to hand the keys over to those eager business units.

Or then again, maybe you're not; after all, this is a lot easier said than done.

That's why it's so valuable to have experienced data engineers on your side, like the ones here at phData. As the largest pure-play provider for data engineering and machine learning, and Snowflake's 2020 Emerging Partner of the Year, phData offers everything you need to be successful with Snowflake. From solution design to 24x7 data pipeline monitoring to software and automation tools, we're here to streamline many of the complex processes required to launch Snowflake.

PHDATA IS HERE TO MAKE YOUR LIFE EASIER.

Drawing from years of implementation experience, our Snowflake teams bring the services and expertise you need to get your Snowflake-based data products into production, with a focus on driving down costs and delivering the best user experience possible.

We offer a free, expert-led, Accelerator Program workshop to get you and your team started with Snowflake. This is a high-level, strategic look at the powerful capabilities available to you in Snowflake so you can deploy it in a way that makes the most sense for your needs and business objectives.

Click the button below to schedule your workshop, and start unlocking the full value of your data.

[SCHEDULE YOUR WORKSHOP TODAY](#)