**Snowflake** – LLD

Version: 8.0

Contents

[1. Document Demographics 4](#_Toc149327190)

[2. Introduction 5](#_Toc149327191)

[2.1 Purpose 5](#_Toc149327192)

[2.2 Scope of this document 5](#_Toc149327193)

[3. Snowflake Account Creation 6](#_Toc149327194)

[4. Snowflake Account Private Link Creation 8](#_Toc149327195)

[5. Snowflake Account Active Directory Setup 10](#_Toc149327196)

[6. Snowflake Account, Environments and DB Configuration 11](#_Toc149327197)

[6.1 Database and Schema Naming conventions 12](#_Toc149327198)

[6.2 Time Travel, Reporting Database and Database Backups 12](#_Toc149327199)

[7. Snowflake Environment Management 15](#_Toc149327200)

[7.1 Environment Setup 15](#_Toc149327201)

[7.2 Virtual Warehouse Management 16](#_Toc149327202)

[7.3 Virtual Warehouse Size Based On Environment 17](#_Toc149327203)

[7.4 Enable Auto-Suspend 17](#_Toc149327204)

[7.5 Enable Auto-Resume 17](#_Toc149327205)

[7.6 Enable Auto-Scaling 17](#_Toc149327206)

[7.7 Should leave Auto-Suspension for All Warehouses 18](#_Toc149327207)

[7.8 Set Timeouts Appropriately for workloads 18](#_Toc149327208)

[7.9 Set Account Statement Timeouts 18](#_Toc149327209)

[7.10 Monitor Cloud Service Billing Threshold 18](#_Toc149327210)

[7.11 Use Resource Monitors 19](#_Toc149327211)

[8. Snowflake Authorizations 20](#_Toc149327212)

[8.1 RBAC Setup 20](#_Toc149327213)

[8.2 Role Based Column Level Security 21](#_Toc149327214)

[8.3 Role Based Row Level Security 21](#_Toc149327215)

[8.4 Role Based Hierarchy 21](#_Toc149327216)

[8.5 Objects Ownership and Grant Management 21](#_Toc149327217)

[8.6 Virtual Warehouses Access Control 22](#_Toc149327218)

[9. Snowflake DR 23](#_Toc149327219)

[9.1 Architecture Diagram of DR Setup 23](#_Toc149327220)

[9.2 Create a DR Account setup 23](#_Toc149327221)

[9.3 Firewall changes 24](#_Toc149327222)

[9.4 Private link setup for DR account 24](#_Toc149327223)

[9.5 AAD link to DR account 26](#_Toc149327224)

[9.6 Enabling replication at account level for both Primary and Secondary (DR) accounts 28](#_Toc149327225)

[9.7 Disable Replication for a Replication Enabled Database 28](#_Toc149327226)

[9.8 Create a Primary Failover Group in a Source Account 28](#_Toc149327227)

[9.9 Create a Secondary Failover Group in the Target Account 28](#_Toc149327228)

[9.10 Connection object creation 29](#_Toc149327229)

[9.11 Private Link set up for Connection 29](#_Toc149327230)

[9.12 AzureAD Link to Connection 29](#_Toc149327231)

[9.13 In the event of DR (Account Fail Over) 29](#_Toc149327232)

[10. Technical Scope of Password rotation 31](#_Toc149327233)

[10.1 Password rotation scripts 31](#_Toc149327234)

[10.2 External OAuth for Snowflake Approach 31](#_Toc149327235)

[11. Snowflake DevOps 33](#_Toc149327236)

[11.1 Scope 33](#_Toc149327237)

[11.1.1 Tools/Services Used 33](#_Toc149327238)

[11.1.1.1 Azure SaaS 33](#_Toc149327239)

[11.1.1.2 Azure PaaS 33](#_Toc149327240)

[11.1.1.3 Other Software 33](#_Toc149327241)

[11.2 Intended Audience 33](#_Toc149327242)

[11.3 Reference 33](#_Toc149327243)

[12. DevOps Solution Architecture for Snowflake 34](#_Toc149327244)

[12.1 Solution Architecture 34](#_Toc149327257)

[12.1.1 Architecture Components: 34](#_Toc149327258)

[12.1.1.1 Azure DevOps 34](#_Toc149327259)

[12.1.1.2 CI – Continuous Integration 34](#_Toc149327260)

[12.1.1.3 Continuous Delivery 37](#_Toc149327261)

[13. Appendix 43](#_Toc149327262)

[14. Snowflake CMDB Discovery: 44](#_Toc149327263)

# Document Demographics

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Print Name** | **E-mail** | **Date** |
| **Author** |  |  |  |
| **Reviewed by** |  |  |  |
| **Approved by** |  |  |  |

# Introduction

Client X onboarded Snowflake to create Data Hub and a modern data platform. This document covers the topics that includes modern application architecture framework as applicable to a data platform.

## Purpose

The document provides complete low-level design and configuration details of Snowflake modern data hub platform.

## Scope of this document

The scope of this document is to provide the complete configuration details of Snowflake DB:

1. Snowflake Account Creation
2. Snowflake Private link setup
3. Snowflake to Azure Active Directory Setup
4. Snowflake DB Configuration
5. Snowflake Environments
6. Snowflake Authorization
7. Disaster Recovery
8. Password Rotation
9. DevOps

# Snowflake Account Creation

Through the CLIENT X corporate account provided by Snowflake, create the primary account. The steps to create a Snowflake account is below:

1. Login to Snowflake through a user who has as Org Admin Role:
2. Click on Organization option.
3. Select the available options as shown below:

A screenshot of a computer

Description automatically generated

1. Enter the Account name and other details as shown in the tab below

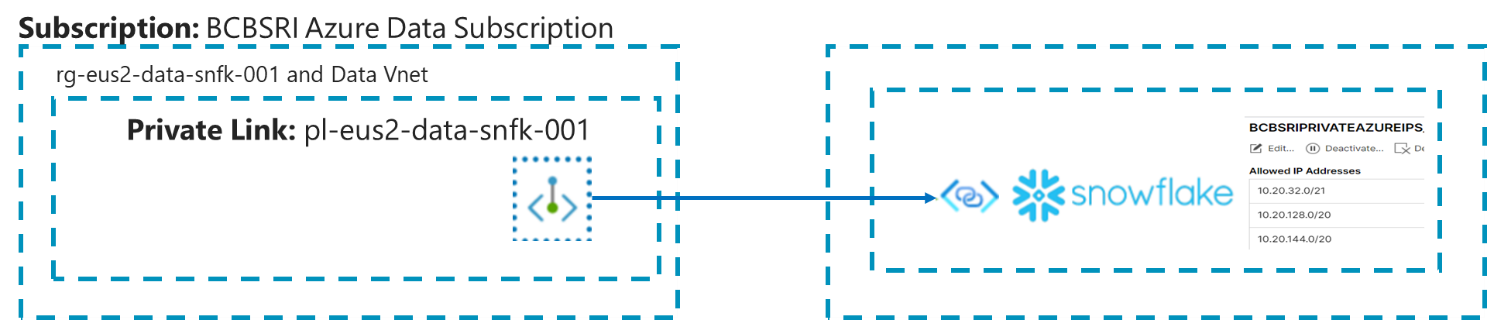
Graphical user interface, table

Description automatically generated with medium confidence

There are following admins when it comes to Snowflake management:

1. Organization Admin – Create Accounts and linking different Accounts of an organization. This role will be managed by the Enterprise Data team.
2. Account Admin – This is an elevated role which one of the Enterprise Data Management team members must have. This is an umbrella role which encompasses Sysadmin, Security Admin and User Admin roles along with other Resource management related privileges (Viz., Credit utilization etc.). This role will be managed by the Enterprise Data team.
3. Next three roles are specific to DBA team members to manage the Snowflake objects within an account.
   1. System Admin – Database, Schema and Object management
   2. Security Admin – This is the role that is used for access management (Privileges and Grants)
   3. User Admin – This is the role to create and manage Users and Roles (Local and AD)

# Snowflake Account Private Link Creation



1. Navigate to Azure portal to create the private link

Graphical user interface, text, application, email

Description automatically generated

1. Click on private endpoints and then click add

Graphical user interface, text, application, email

Description automatically generated

1. In the basics section complete subscription, resource group and region fields

Graphical user interface, text, application, email

Description automatically generated

1. In resource section complete connection method, resource ID and request message

Graphical user interface, text, application, email

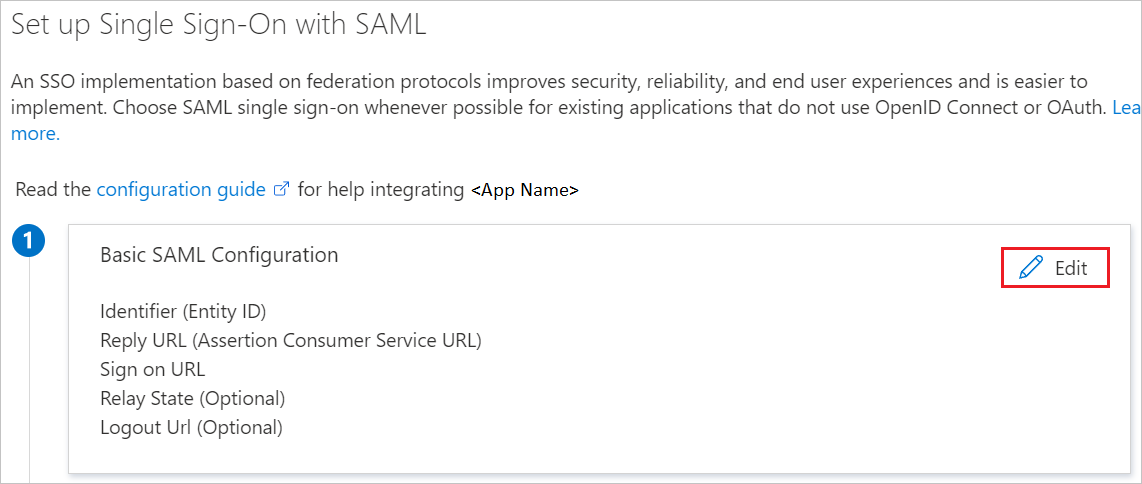
Description automatically generated

* For **connection** method, select connect by resource ID or alias
* In Snowflake, run [SYSTEM$GET\_PRIVATELINK\_CONFIG](https://docs.snowflake.com/pt/sql-reference/functions/system_get_privatelink_config.html) and enter the value for in the Resource ID or Alias field. Note that the screenshot in this step uses the alias value for the region as a representative example, and that Azure confirms a valid alias value with a green check mark. privatelink-pls-ideast-us-2
* **Return to the**Private endpoints**section and wait a few minutes. When approved, the private endpoint displays a**CONNECTION STATE**value of**Pending
* In Snowflake, call the [SYSTEM$AUTHORIZE\_PRIVATELINK](https://docs.snowflake.com/pt/sql-reference/functions/system_authorize_privatelink.html) function, using the value and value as arguments using private endpoint resource id and federated token.

1. Create DNS entry to route all entries through private endpoint.

# Snowflake Account Active Directory Setup

1. Snowflake user authentication is established through CLIENT X Azure Active Directory
2. The users will have single sign on capability
3. Snowflake provides an Azure AD SSO application, which must be granted by the GAM team to the user who wish to access Snowflake
4. Enable Snowflake AD app in Azure.
5. In the Azure portal, on the **Snowflake** application integration page, find the **Manage** section and select **single sign-on**.
6. On the **Select a single sign-on method** page, select **SAML**.
7. On the **Set up single sign-on with SAML** page, click the pencil icon for **Basic SAML Configuration** to edit the settings.



1. In the **Basic SAML Configuration** section, perform the following steps, if you wish to configure the application in **IDP** initiated mode
2. Following URLs are registered with the AD application

**Identifier** **-** [https://bu34048.east-us-2.privatelink.snowflakecomputing.com](https://bu34048.east-us-2.privatelink.snowflakecomputing.com/)

**Reply URL -** <https://bu34048.east-us-2.privatelink.snowflakecomputing.com/fed/login>

**Sign-on URL -** [https://bu34048.east-us-2.privatelink.snowflakecomputing.com](https://bu34048.east-us-2.privatelink.snowflakecomputing.com/)

**Logout URL -** <https://bu34048.east-us-2.privatelink.snowflakecomputing.com/fed/logout>

1. In the Snowflake AD integration is setup using the certificate obtained after configuring the Azure AD SSO

CREATE SECURITY INTEGRATION AZUREADINTEGRATION

TYPE = SAML2

ENABLED = TRUE

SAML2\_ISSUER = 'https://sts.windows.net/2c93d3cb-f285-49c4-8265-fba1bd1fe468/'

SAML2\_SSO\_URL = 'https://login.microsoftonline.com/2c93d3cb-f285-49c4-8265-fba1bd1fe468/saml2'

SAML2\_PROVIDER = 'CUSTOM'

SAML2\_X509\_CERT = ‘Token’

SAML2\_SP\_INITIATED\_LOGIN\_PAGE\_LABEL = 'AzureADSSO'

SAML2\_ENABLE\_SP\_INITIATED = TRUE

saml2\_snowflake\_issuer\_url = 'https://bu34048.east-us-2.privatelink.snowflakecomputing.com'

saml2\_snowflake\_acs\_url = 'https://bu34048.east-us-2.privatelink.snowflakecomputing.com/fed/login';

1. The Snowflake Account is set to use the SSO login

# Snowflake Account, Environments and DB Configuration

Snowflake Data Hub is designed to have two main accounts:

1. Primary Account
2. DR Account

Databases for all environments are created in Primary account with DR account containing replica of Production databases.

We chose to create single account rather than creating multiple accounts for each environment as a single account provides maximum flexibility as well as the opportunity to leverage “unique to Snowflake” features, some of them are listed below:

1. Zero copy cloning of DBs will help the database refresh cycles to be faster.
2. Time travel will enable to precisely clone database with point in time data.
3. Managing users, roles, and privileges would be centralized.
4. Reporting copies can be provisioned to the business users with no delay.

The single account hosts all databases for the following environments:

1. Development
2. QA
3. Test MinorEDR
4. Test Major
5. Production

The only disadvantage of using single account is that the users with elevated access will have a greater number of DBs listed for them to manage. For regular users, developers, support members and analyst’s single vs multiple account will not have any impact as they will be exposed to only the DBs that they are granted to access.

Each environment will have the following type of DBs:

1. Application DB – The data that is exposed to downstream applications, reporting extracts, and ad hoc user queries.
2. Landing DB – This DB hosts transient data that is extracted from source systems
3. Audit DB – This DB hosts the batch audit information and other meta data that is required for the job configuration

Graphical user interface

Description automatically generated

## Database and Schema Naming conventions

The names database names end with the corresponding environment name viz., \*DEV, \*MINOR, \*MAJOR and \*PROD

All related application will be hosted in the same DB as separate schemas, for e.g., EDR DB hosts EDR, INCOME and ACG applications as separate schemas. FDA DB hosts FDA and FDW applications as separate schemas. Schemas are created for each application to logically separate the objects.

Landing DB will host the landing data for all applications, and they are separated by creating separate schema. The naming convention used for the schemas within the Landing DB is as follows <DBName>\_<ApplicationName>. For e.g., the landing schema name for EDR application would be EDR\_EDR, and for Income it would be EDR\_Income.

## Time Travel, Reporting Database and Database Backups

Snowflake Time Travel enables accessing historical data (i.e. data that has been changed or deleted) at any point within a defined period. It serves as a tool for performing the following tasks:

* Restoring data-related objects (tables, schemas, and databases) that might have been accidentally or intentionally deleted.
* Duplicating and backing up data from key points in the past.
* Analyzing data usage/manipulation over specified periods of time.

Using Time Travel, following actions can be performed within a defined period of time:

Diagram

Description automatically generated with medium confidence

* Query data in the past that has since been updated or deleted.
* Create clones of entire tables, schemas, and databases at or before specific points in the past.
* Restore tables, schemas, and databases that have been dropped.

Once the defined period of time has elapsed, the data is moved into Snowflake Fail-safe and these actions can no longer be performed.

To support Time Travel, the following SQL extensions have been implemented in Snowflake:

AT | BEFORE clause which can be specified in SELECT statements and CREATE … CLONE commands (immediately after the object name). The clause uses one of the following parameters to pinpoint the exact historical data you wish to access:

* TIMESTAMP
* OFFSET (time difference in seconds from the present time)
* STATEMENT (identifier for statement, e.g. query ID)
* UNDROP command for tables, schemas, and databases.

Timeline

Description automatically generated

* For transient databases, schemas, and tables, the retention period can be set to 0 (or unset back to the default of 1 day). The same is also true for temporary tables.
* For permanent databases, schemas, and tables, the retention period can be set to any value from 0 up to 90 days.

A user with the ACCOUNTADMIN role can set the [MIN\_DATA\_RETENTION\_TIME\_IN\_DAYS](https://docs.snowflake.com/en/sql-reference/parameters.html#label-min-data-retention-time-in-days) at the account level. This parameter setting enforces a minimum data retention period for databases, schemas, and tables. Setting MIN\_DATA\_RETENTION\_TIME\_IN\_DAYS does not alter or replace the DATA\_RETENTION\_TIME\_IN\_DAYS (set at the DB, Schema or Table) parameter value. It may, however, change the effective data retention period for objects. When MIN\_DATA\_RETENTION\_TIME\_IN\_DAYS is set at the account level, the data retention period for an object is determined by MAX(DATA\_RETENTION\_TIME\_IN\_DAYS, MIN\_DATA\_RETENTION\_TIME\_IN\_DAYS).

**Backup rules followed for the Snowflake DBs**

\*PROD DBs are setup with DATA\_RENTENTION\_TIME as 30 days – We can go as back as 30 days to retrieve data for Production DBs

\*MAJOR and \*MINOR are setup with DATA\_RENTENTION\_TIME as 14 days - We can go as back as 14 days to retrieve data for Production DBs

\*DEV DBs we will have default 1 day retention.

The Time travel command reference can be found in below link:

<https://docs.snowflake.com/en/user-guide/data-time-travel>

# Snowflake Environment Management

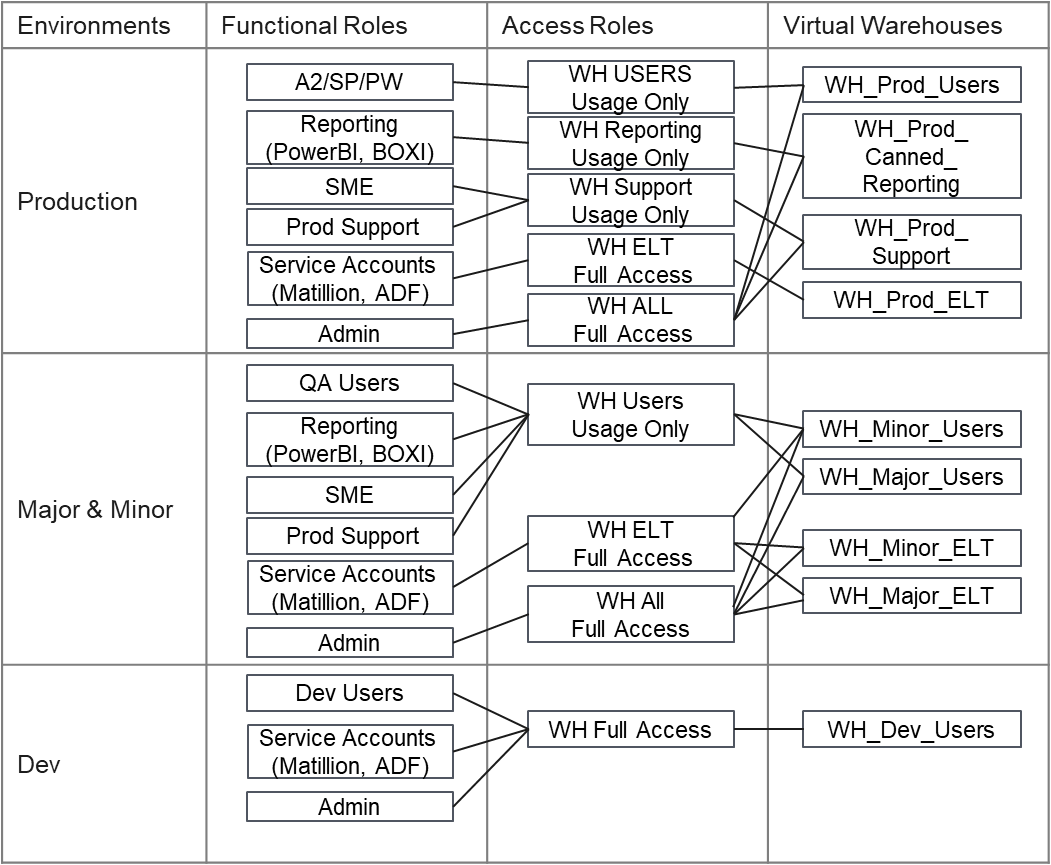
## Environment Setup

Diagram

Description automatically generated

The above diagram depicts the environments that are made available within Snowflake subscription.

## Virtual Warehouse Management



Multiple virtual warehouses need to be designed within snowflake to satisfy different user group access. The Virtual Warehouse design also helps in monitoring & controlling the costs. Separate warehouse to be created for each environment.

Warehouse usage access control should have the same structure with RBAC for databases. Warehouses should be granted to Access Roles and Access Roles should be granted to Functional Roles. See section 8.6.

Below table represents the warehouse size and min/max cluster for each type of warehouse and for each environment. The max cluster & warehouse size will be re-visited based on the stats from the IBM Netezza w.r.t to No. of users & Queries being processed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Environment** | **Warehouse\_Name** | **Warehouse Warehouse\_Size** | **Min Cluster** | **Max Cluster** |
| Production | WH\_Prod\_Users | S | 1 | 5 |
| Production | WH\_Prod\_Canned\_Reporting | S | 1 | 3 |
| Production | WH\_Prod\_Support | S | 1 | 1 |
| Production | WH\_Prod\_ELT | S | 1 | 3 |
| Test Major | WH\_Major\_Users | S | 1 | 1 |
| Test Major | WH\_Major\_ELT | S | 1 | 1 |
| Test Minor | WH\_Minor\_Users | S | 1 | 1 |
| Test Minor | WH\_Minor\_ELT | S | 1 | 1 |

**Note:** The above table represents the VM’s & sizes based on estimated loads and capacity. The VM designs will further be updated based on implementation and learnings. When creating a virtual warehouse in Snowflake, the two most critical factors to consider, from a cost and performance perspective, are:

* Warehouse size (i.e. available compute resources)
* Manual vs automated management (for starting/resuming and suspending warehouses).

Below are few aspects and best practices that need to be considered while designing and maintaining virtual warehouse:

## Virtual Warehouse Size Based On Environment

* + Create separate warehouse for different environment such as development, testing and production.
  + For queries in development and testing environments, smaller warehouses sizes (X-Small, Small, Medium) may be sufficient.
  + For queries in production environments, larger warehouse sizes (Large, X-Large, 2X-Large, etc.) may be more cost effective.
  + Small/simple queries typically do not need an X-Large (or larger) warehouse because they do not necessarily benefit from the additional resources, regardless of the number of queries being processed concurrently.
  + For data loading, the warehouse size will match the number of files being loaded and the amount of data in each file. Make sure the file size will not be greater than snowflake recommended size to achieve best performance and cost optimize.
  + It is always better to create a separate warehouse for your data loading and query execution. Start with smaller size and based on the performance, you can manually resize the warehouse.

## Enable Auto-Suspend

Make sure all virtual warehouses are set to auto-suspend. This way, when they are done processing queries, auto-suspend will turn off your virtual warehouses when they are done processing queries, and thus stop credit consumption.  Day 1 setting for all warehouses is to enable auto suspend as on.

## Enable Auto-Resume

Make sure all virtual warehouses are set to auto-resume. If you are going to implement auto-suspend and set appropriate timeout limits, enabling auto-resume is a must; otherwise, users will not be able to query the system.

## Enable Auto-Scaling

Multi-cluster warehouses will be configured to run in an Auto-scale mode, which enables Snowflake to automatically start and stop clusters as needed.

Day 1 warehouse size would be as below:

1. Warehouse for uses – Extra Small with Standard policy and no clusters with auto suspend after 1 minutes of inactivity
2. Warehouse for Matillion – Extra Small with Standard policy and no clusters with auto suspend after 10 minutes of inactivity
3. Warehouse for SRDM – Medium with Standard policy and no clusters with auto suspend after 5 minutes of inactivity

## Should leave Auto-Suspension for All Warehouses

Should leave auto suspend on, with a timeout greater then the executing range (ie. 10 mins for a 5 min range), since it covers failure scenarios.

Day 1 settings for warehouses are mentioned above section. However, this will be continuously monitored and updated in the steady state

## Set Timeouts Appropriately for workloads

All virtual warehouses will have an appropriate timeout for their particular workload:

* For task, data loading, and ETL/ELT warehouses, set the timeout for suspension immediately upon completion.
* For BI and SELECT query warehouses, set the suspension timeout to 10 minutes in most situations to keep data caches warm for frequent access by end users.
* For DevOps, DataOps, and data science warehouses, set the suspension timeout to 5 minutes because having a warm cache is not as important for ad hoc and highly unique queries.

## Set Account Statement Timeouts

Use the below mentioned  parameters to automatically stop queries that are taking too long to execute, either due to a user error or a frozen cluster. Customize warehouse, account, session, and user timeout-level statements according to your data strategy for long-running queries.

* [STATEMENT\_QUEUED\_TIMEOUT\_IN\_SECONDS](https://docs.snowflake.com/en/sql-reference/parameters.html#statement-queued-timeout-in-seconds)
* [STATEMENT\_TIMEOUT\_IN\_SECONDS](https://docs.snowflake.com/en/sql-reference/parameters.html#statement-timeout-in-seconds)

Ex: ALTER WAREHOUSE LOAD\_WH SET STATEMENT\_TIMEOUT\_IN\_SECONDS= 3600;

Day 1 timeout configuration for Matillion warehouse is 4 hours and user warehouse it is 2 hours

## Monitor Cloud Service Billing Threshold

Snowflake will charge for cloud services only if they exceed 10% of the daily virtual warehouse credit consumption. Cloud services tasks are useful for meta-data operations such as BI tool discovery queries, heartbeat queries, SHOW commands, cache usage, and several other service optimizing features. So if you use 100 compute credits in a day, but you use 15 additional credits for cloud services (unlikely), you will be charged an additional 5 credits for that day for the 5 cloud service credits that were over the 10% allowance.

## Use Resource Monitors

Resource monitors are a great way to proactively control workload budgets and prevent unexpected resource spikes. Resource monitors can help monitor both user usage and service account usage in Snowflake. First, dedicated virtual warehouses are created for each of loading, ELT, BI, reporting, and data science workloads as well as for other workloads. Accounts and warehouses can have total, yearly, monthly, weekly, and daily credit quotas.

Considerations for Resource Monitoring:

* Notify when a certain threshold of consumption is reached.
* When consumption approaches the maximum budgeted level, set the resource monitor to auto-suspend the warehouse or the entire account, allowing queries to complete but preventing future requests.
* It can also be used to terminate all currently running queries and immediately suspend the resource or account. This setting is usually reserved for situations where a hard quota is exceeded.

For customers that do not want to set hard limits, it’s still always a good idea to have notification monitors set on all warehouses in case usage unexpectedly spikes. That way, all admins within the account will get an email or on-screen notification when thresholds are reached.

# Snowflake Authorizations

1. When the user successfully sign on using Azure AD SSO, they will see a blank Snowflake page and no databases are listed for them to be used, unless the DBA grants them the appropriate roles
2. The DBA will grant the user appropriate role after approval from EIM team.
3. The roles are defined by the EIM team for the following category of users.
   1. Analytics/Business users
   2. Development/Support users
   3. Service Accounts
   4. Administrators
4. Based on the role that is granted to the user, they are authorized to perform the appropriate function in Snowflake

Graphical user interface, text, application, email

Description automatically generated

## RBAC Setup

The example diagram above illustrates the overall multi-layer RBAC architecture which consist of the following layers.

1. Create schemas per applications by SysAdmin.
2. Create Access Roles with some access levels for schemas according to security requirements.
3. Create Functional Roles according to business requirements.
4. Assign the Functional Roles to the Snowflake users.

## Role Based Column Level Security

Column Level Security in Snowflake allows to mask the columns which contain sensitive/confidential data and it can be achieved by creating a Data Masking policy at the column level.

Dynamic Data Masking uses the masking policies to mask the data in the selective columns in a table or view. This means the actual data in the table is not masked (no static masking) but while querying the table, based on user role/user group snowflake apply masking policies to show either the masked/unmasked actual data for authorized users.

Below is the process to create column-level masking policies:

1. Create the Dynamic masking policies to restrict the PII or PHI columns based on the user role grants
2. Apply the masking policies to the Stage & Cert views
3. Create the roles (MASK\_READ) and grant to the functional roles (B USERS & C USERS)
4. Assign the functional roles to the Snowflake users

## Role Based Row Level Security

Snowflake achieves Row Level Security using [Row Access Policies](https://docs.snowflake.com/en/user-guide/security-row-intro.html).  A row access policy is a small, centrally defined procedure that returns a Boolean value (TRUE or FALSE) depending whether the user is allowed to view the specific row.

The steps to defining row level security include:

1. Decide the view needs to be secured.
2. Create a mapping table which includes the role & column to be filtered
3. Create a Row Access Policy to implement the rule
4. Deploy the Row Access Policy against the Stage & Cert views

## Role Based Hierarchy

Roles such as A Users, B Users, Service Accounts, etc., will serve as a Functional roles which will be inherited from access roles i.e., App 1 Prod Read, App 1 Prod Read Write, App 2 Prod Read, etc. At any point, only access roles will be amended/updated when a new view or table or columns or rows needs to be restricted which will be automatically applied to the functional roles which in turn will be applied to Snowflake users.

## Objects Ownership and Grant Management

Databases, schemas, and warehouses will be owned by SYSADMIN role, roles will be owned by SECURITYADMIN role. This is for avoiding using ACCOUNTADMIN except account-level administration activity.

All schemas will be created as managed access schemas, which is the feature for centralizing grant management to schema owner. In managed access schemas, owner of tables and views lose the ability to make grant decisions.

When granting basic schema level access, add “on future” key word and grant the same privileges on all new tables created in the schema. It avoids grant management by individual table or view.

## Virtual Warehouses Access Control

Virtual warehouses will be a part of multi-layer RBAC architecture as well as databases and schemas. Each warehouse will be granted appropriate privileges like admin, monitoring, usage to each role. Dev users should be granted monitoring privileges on warehouses for ELT jobs because Dev users need to see system accounts activity by query history.

# Snowflake DR

Overall DR approach for Snowflake and all applications that connects Snowflake must have a separate DR setup.

# Architecture Diagram of DR Setup

Graphical user interface, diagram

Description automatically generated

# Create a DR Account setup

Log in to Snowflake account as an organization administrator and create a Snowflake DR account in the central US region.

Graphical user interface, application

Description automatically generated

# Firewall changes

Open the port 443 connection to the newly created Snowflake account URL from CLIENT X Azure tenets. This helps in establishing the connection from CLIENT X networks.

# Private link setup for DR account

* + 1. Navigate to Azure portal to create the private link

Graphical user interface, text, application, email

Description automatically generated

* + 1. Click on private endpoints and then click add

Graphical user interface, text, application, email

Description automatically generated

* + 1. In the basics section complete subscription, resource group and region fields

Graphical user interface, text, application, email

Description automatically generated

* + 1. In resource section complete connection method, resource ID and request message

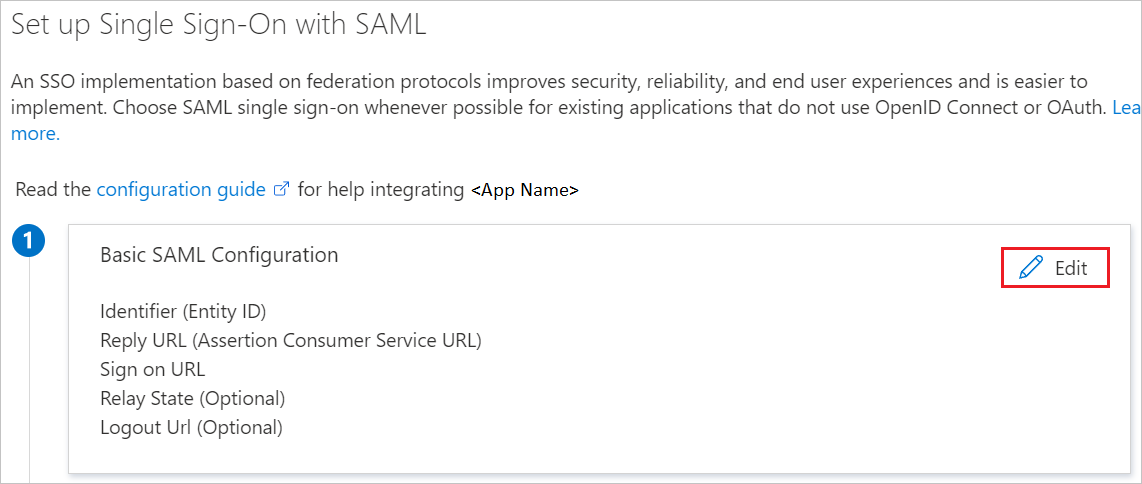
Graphical user interface, text, application, email

Description automatically generated

* For **connection** method, select connect by resource ID or alias.
* In Snowflake, run [SYSTEM$GET\_PRIVATELINK\_CONFIG](https://docs.snowflake.com/pt/sql-reference/functions/system_get_privatelink_config.html) and enter the value for in the Resource ID or Alias field. Note that the screenshot in this step uses the alias value for the region as a representative example, and that Azure confirms a valid alias value with a green check mark. privatelink-pls-ideast-us-2
* **Return to the**Private endpoints**section and wait a few minutes. When approved, the private endpoint displays a**CONNECTION STATE**value of**Pending
* In Snowflake, call the [SYSTEM$AUTHORIZE\_PRIVATELINK](https://docs.snowflake.com/pt/sql-reference/functions/system_authorize_privatelink.html) function, using the value and value as arguments using private endpoint resource id and federated token.
  + 1. Create DNS entry to route all entries through private endpoint.

# AAD link to DR account

1. Enable Snowflake AD app in Azure.
2. In the Azure portal, on the Snowflake application integration page, find the Manage section and select single sign-on.
3. On the **Select a single sign-on method** page, select **SAML**
4. On the **Set up single sign-on with SAML** page, click the pencil icon for **Basic SAML Configuration** to edit the settings.



1. In the **Basic SAML Configuration** section, perform the following steps, if you wish to configure the application in **IDP** initiated mode
2. Click **Set additional URLs** and perform the following step if you wish to configure the application in **SP** initiated mode:
   1. **Identifier** - [https://Client X\_secondary.central-us.privatelink.snowflakecomputing.com](https://bcbsri_secondary.central-us.privatelink.snowflakecomputing.com)
   2. **Reply URL -** [https:// Client X\_secondary.central-us.privatelink.snowflakecomputing.com/fed/login](https://bu34048.east-us-2.privatelink.snowflakecomputing.com/fed/login)
   3. **Sign-on URL -** [https:// Client X\_secondary.central-us.privatelink.snowflakecomputing.com](https://bu34048.east-us-2.privatelink.snowflakecomputing.com)
   4. **Logout URL -** [https:// Client X\_secondary.central-us.privatelink.snowflakecomputing.com/fed/logout](https://bu34048.east-us-2.privatelink.snowflakecomputing.com/fed/logout)
3. In snowflake create a security integration as below

CREATE SECURITY INTEGRATION AZUREADINTEGRATION

TYPE = SAML2

ENABLED = TRUE

SAML2\_ISSUER = 'https://sts.windows.net/2c93d3cb-f285-49c4-8265-fba1bd1fe468/'

SAML2\_SSO\_URL = 'https://login.microsoftonline.com/2c93d3cb-f285-49c4-8265-fba1bd1fe468/saml2'

SAML2\_PROVIDER = 'CUSTOM'

SAML2\_X509\_CERT = ‘Token’

SAML2\_SP\_INITIATED\_LOGIN\_PAGE\_LABEL = 'AzureADSSO'

SAML2\_ENABLE\_SP\_INITIATED = TRUE

saml2\_snowflake\_issuer\_url = 'https://bu34048.east-us-2.privatelink.snowflakecomputing.com'

saml2\_snowflake\_acs\_url = 'https://bu34048.east-us-2.privatelink.snowflakecomputing.com/fed/login';

1. Create network policy in Snowflake to allow only CLIENT X Azure Tenants and Azure AD IPs

# Enabling replication at account level for both Primary and Secondary (DR) accounts

Connect to Snowflake as Organization admin and enable the replication of primary and secondary DR accounts

select system$global\_account\_set\_parameter('LZFCOGN.CLIENT X\_SECONDARY', 'ENABLE\_ACCOUNT\_DATABASE\_REPLICATION', 'true');

select system$global\_account\_set\_parameter('LZFCOGN.CLIENT X\_PRIMARY', 'ENABLE\_ACCOUNT\_DATABASE\_REPLICATION', 'true');

## Disable Replication for a Replication Enabled Database

Account replication can be activated to replicate account level objects like roles and warehouses as well as databases. It removes the need to handle replication yourself with stored procs.

For enabling account replication, if a database was previously enabled for replication, database replication must be disabled before the database can be added to a replication or failover group.

Disable by the following command in the CLIENT X.PRIMARY account as ACCOUNTADMIN role.

SYSTEM$DISABLE\_DATABASE\_REPLICATION('<DB\_NAME>');

## Create a Primary Failover Group in a Source Account

Create a Failover Group, which is object for enabling account replication, in the CLIENT X.PRIMARY account as ACCOUNTADMIN role.

CREATE FAILOVER GROUP DR\_GROUP

OBJECT\_TYPES = USERS, ROLES, WAREHOUSES, RESOURCE MONITORS, DATABASES, INTEGRATIONS, NETWORK POLICIES

ALLOWED\_DATABASES = <DB\_NAME>, <DB\_NAME>

ALLOWED\_ACCOUNTS = CLIENT X.SECONDARY

REPLICATION\_SCHEDULE = '10 MINUTE';

## Create a Secondary Failover Group in the Target Account

Create a Failover Group in the CLIENT X.SECONDARY account as ACCOUNTADMIN as well.

CREATE FAILOVER GROUP DR\_GROUP

AS REPLICA OF CLIENT X.PRIMARY. DR\_GROUP;

## Connection object creation

A common connection object is created which can be used in ODBC, JDBC, Python and Snowflake URL connectors.

In The primary account execute the following:

CREATE CONNECTION DataHub;

ALTER CONNECTION DataHub ENABLE FAILOVER TO ACCOUNTS CLIENT X.SECONDARY

In the secondary account execute the following:

CREATE CONNECTION DataHub as Replica of CLIENT X.PRIMARY.DATAHUB;

## Private Link set up for Connection

* 1. Execute SHOW CONNECTIONS.

SHOW CONNECTIONS;

1. Record the CONNECTION\_URL column value, and create two URLs to support private connectivity and OCSP like follow example.  
     
   myorg-myconnection.privatelink.snowflakecomputing.com.

ocsp. myorg-myconnection.privatelink.snowflakecomputing.com.

1. Set up DNS CNAME for the connection URL and the OCSP URL. Configure the record to have the connection URL resolve to the primary Snowflake account URL.

## AzureAD Link to Connection

* 1. Update setting up single sign-on with SAML for using the connection URL in Azure portal.
  2. Update the SAML2 security integration in the primary account for using the connection URL.

## In the event of DR (Account Fail Over)

1. Connect to Secondary Account and execute the connection switch

ALTER CONNECTION DataHub Primary;

1. Update a DNS CNAME record for the connection URL.
2. Promote Secondary Production Account to Primary.

ALTER FAILOVER GROUP DR\_GROUP PRIMARY;

# Technical Scope of Password rotation

Explain what the scope for password rotation is, (Only service accounts). Call out about the password refresh step, it will be implemented in Prod Snowflake server..

This document covers the password rotation process. Below are the aspects covered in the scope:

1. We must pass the Snowflake username, Key vault name, Secret name, path and path2 as arguments to the python script like below.

Python3 snow\_key\_rotation.py TEST\_PUB\_KEY azkv-eus2-dev-pythn-01 DEMO-SECRET-KEY /apps/srdm/int/key/ /apps/srdm/int/key2/

1. When you execute above script:

a. first it will generate an encrypted private and public key pair in /apps/srdm/int/key2/ path (rsa\_key.p8 and rsa\_key.pub).

b. Second step is, update the corresponding Snowflake user with generated public key through by triggering stored procedure( Attached the procedure).

c. Third step is, update the Azure key vault secret with generated private key.

d. Finally, Remove the private key file and move the public key file from /apps/srdm/int/key2/ to /apps/srdm/int/key/. This moved public key will be used while updating next time.

**Note:** For first time, we must generate the private and public key pair and update the public key in Snowflake and Private key in Azure key vault manually.

ALTER USER <SNOWFLAKE USERNAME> SET RSA\_PUBLIC\_KEY=’<Generated Public key>’;

## Password rotation scripts







## External OAuth for Snowflake Approach

Through External OAuth, the users can programmatically authenticate to Snowflake using the tokens generated by Azure AD application. The configuration of Azure AD for External OAuth is done using the below document.

<https://docs.snowflake.com/en/user-guide/oauth-azure>

Two AD applications are created to generate OAuth Token

1. ClientX-Snowflake OAuth Resource
2. ClientX -Snowflake Client App

An integration object is created in Snowflake to authenticate using the tokens generated by AD application:

create security integration external\_oauth\_azure\_1

type = external\_oauth

enabled = true

external\_oauth\_type = azure

external\_oauth\_any\_role\_mode = 'ENABLE'

external\_oauth\_issuer = 'https://sts.windows.net/2c93d3cb-f285-49c4-8265-fba1bd1fe468/'

external\_oauth\_jws\_keys\_url = 'https://login.microsoftonline.com/2c93d3cb-f285-49c4-8265-fba1bd1fe468/discovery/v2.0/keys'

external\_oauth\_audience\_list = ('api://9f341e20-12ed-4e94-98e4-4674ea8caa56')

external\_oauth\_token\_user\_mapping\_claim = 'upn'

external\_oauth\_snowflake\_user\_mapping\_attribute = 'login\_name';

A sample Python code to generate JWT (Json Web token)





# Snowflake DevOps

# Scope

This document captures the Solution Design for DevOps process designed for SnowFlake, GIT strategy and automated build/deployments. It also covers the overall Azure DevOps components used for designing a CI/CD solution along with the branching strategy designed based on project requirements.

# Tools/Services Used

# Azure SaaS

* 1. Azure DevOps
     + Azure Repos
     + Azure Pipelines

# Azure PaaS

* 1. Azure Key Vault

# Other Software

* 1. Schemachange
  2. Python

# Intended Audience

* NTT DevOps developers and support team
* BIDW D&A Supply and Demand Developer
* Snowflake and Matillion admins

# Reference

* <https://docs.snowflake.com/en/user-guide/intro-key-concepts>
* <https://www.kipi.bi/post/snowflake-ci-cd-using-schemachange>
* <https://github.com/Snowflake-Labs/schemachange#overview>

# DevOps Solution Architecture for Snowflake

## Solution Architecture

DevOps solution is built using Azure DevOps services. This enables source code management (storage, track changes, multi user development), build and deploy code into different environments with a set of policies defined.

## Architecture Components:

### Azure DevOps

Azure DevOps supports a collaborative culture and set of processes that bring together developers, project managers, and contributors to develop software. It allows organizations to create and improve products at a faster pace than they can with traditional software development approaches.

DevOps is a software development practice that promotes collaboration between development and operations, resulting in faster and more reliable software delivery. Commonly referred to as a culture, DevOps connects people, process, and technology to deliver continuous value.

#### Azure Repos

Azure Repos is a set of version control tools that you can use to manage your code. Whether your software project is large or small, using version control as soon as possible is a good idea. Version control systems are software that help you track changes you make in your code over time.

Even if you're just a single developer, version control helps you stay organized as you fix bugs and develop new features. Version control keeps a history of your development so that you can review and even roll back to any version of your code with ease.

#### Azure Artifacts

Azure Artifacts enables developers to share their code efficiently and manage all their packages from one place. Azure Artifacts enables single build and multi deploy (single build is deployed to all environments).

#### Azure Pipeline

Azure Pipelines automatically builds and deploy code projects. Azure pipeline usually consists of CI/CD – Continuous Integration/Continuous Delivery

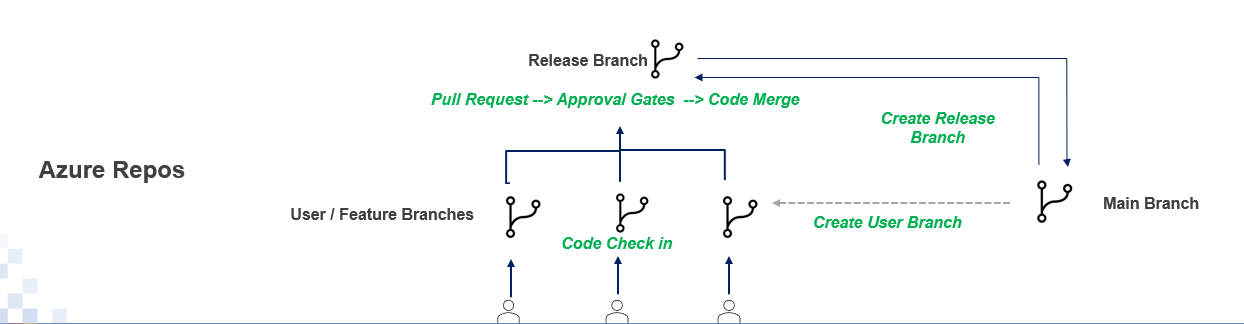
### CI – Continuous Integration

**Continuous integration** (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It is a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository .Below are the links of snowflake repositories created for Supply and Demand teams.

[https://dev.azure.com/CLIENT X/BIDW/\_git/repo\_supply\_snowflake](https://dev.azure.com/BCBSRI/BIDW/_git/repo_supply_snowflake)  
[https://dev.azure.com/CLIENT X/BIDW/\_git/repo\_demand\_snowflake](https://dev.azure.com/BCBSRI/BIDW/_git/repo_demand_snowflake)

#### Branching Strategy

Below diagram illustrates the GIT branching strategy for Snowflake



#### Types of Branches

1. ***Feature branch*** – Developer creates feature branch as per standard naming convention to check in the code. Feature branch naming should <TOPAZID>\_<TOPAZ DESC>
2. ***Release branch*** – Developer creates release branch as per standard naming convention to merge the code. Release branch naming should be rel\_<TOPAZID>\_<TOPAZ DESC>
3. ***Main branch*** – Clean version of code deployed to Prod is maintained in Main branch

* The feature and release branches are cloned from main branch to contain production copy of code
* Developers develop their code and commit it to Feature branch i.e., feature1, feature2, feature3 etc. under users folder with respective project/rss names
* Developer creates Release branch under release folder with RSS/project name and then creates pull request to merge the code from feature branch to release branch
* From release branch the code gets deployed into Dev, Minor, Major and Prod with approval process implemented using approval gates
* On successful deployment to Prod, the code gets synchronized to the Main branch. There are no manual direct commits into release and main branch. Main branch contains the latest code in Prod environment.

#### Branch Policies

In the Main branch, the restrictions are applied through build validation file. The purpose of this file is to make sure that the PR raised from release/\* will get merged to main branch.

A screenshot of a computer

Description automatically generated with medium confidence

Pull Request(PR) raised from any branch other than Release branch to merge into Main branch is restricted

# Starter pipeline

# Start with a minimal pipeline that you can customize to build and deploy your code.

# Add steps that build, run tests, deploy, and more:

# https://aka.ms/yaml

trigger:

- none

pool:

    name: 'CLIENT X\_WIN\_Agt\_Pool'

steps:

- task: PowerShell@2

  inputs:

    targetType: 'inline'

    script: |

      $sourceBranch = "$(System.PullRequest.SourceBranch)"

      Write-Host  "SourceBranch : $sourceBranch"

      if($sourceBranch -like "refs/heads/users\*")

      {

        Write-Host "Cant merge the code directly from users branch($(System.PullRequest.SourceBranch)) to $(System.PullRequest.TargetBranch)" -ForegroundColor Red -BackgroundColor Yellow

          exit 1

      }

#### Branch reviewers

Lead group id is added as Peer Reviewer of code.

A screenshot of a computer

Description automatically generated with low confidence

#### Build Service Permission on Repository

The below diagram represents the permissions on repository with BIDW Build service group.

A screenshot of a computer

Description automatically generated with medium confidence

### Continuous Delivery

Continuous delivery (CD) is the process of automating build and deployment from a build to a production environment. A release pipeline can create staging environments to automate deploy new builds.

In the this project the code is fetched from the repository as artifact and then the code is mapped to the dev environment first and through release pipeline task the code can be deployed to the Minor, Major and Prod environments.

#### Schemachange

Schemachange is a Database Change Management (DCM) tool. DCMs are used to manage different objects in a database via some tools or processes. When combined with a version control system and a CI/CD tool, database changes can be approved and deployed through a pipeline using modern software delivery practices.

Schemachange is Python-based DCM tool. It follows an imperative-style approach to managing changes within Snowflake database. In imperative style, all scripts are put in one script file for deployment unlike declarative approach where we have individual script for each object. Advantage of imperative style is that there is no need of separate schema comparison tool. The disadvantage of imperative style is that it is unable to maintain individual script versions. To get advantages of declarative style while using in Schemachange, in current implementation, we have defined sub folders for object types and process of creating one script per object; so that delta changes can be identified, and script versions can be maintained.

Azure Pipeline and Azure Repos are used with Schemachange as DCM to manage Snowflake Account. The following architecture states the flow:

A diagram of a diagram

Description automatically generated

#### Release pipeline

A diagram of a computer process

Description automatically generated

**Step 1:**

The developers commit code to feature branch and the code then pushed to release branch via pull request. As per the approval process defined within the project, the code from release branch is deployed to Test Minor, Major and Production databases with stages and gate approvals.

**Step 2:**

The release branches i.e., release/\* are mapped to release pipeline with repository information as artifact to the release pipeline. The below diagram depicts the pipeline configuration set up for reference.

A screenshot of a computer

Description automatically generated  
  
**Step 3:**

The Linux machine i.e., **bazeu2dlsoapp03** has been configured with the installation of python 3.8 version and Schemachange tool. Below screenshot of agent mapping to the release pipeline.

A screenshot of a computer

Description automatically generated  
  
**Step 4:**

The release pipeline includes three tasks i.e., Fetch private key from KeyVault, Replace tokens for databases in SQL Scripts and then Deploy scripts to Dev using Schema change CLI. The below link provides the details of the the release pipeline configured for the project.

[https://dev.azure.com/CLIENT X/BIDW/\_releaseDefinition?definitionId=16&\_a=definition-tasks&environmentId=48](https://dev.azure.com/BCBSRI/BIDW/_releaseDefinition?definitionId=16&_a=definition-tasks&environmentId=48)  
  
  
**Step 5:**

The service connection for the release pipeline is **azsc-eus2-data-Matillion-001** and It is for resource group(s) which holds key vault resources.

In Azure portal for respective keyvault App registration of the Azuredevops service connection needs to be setup. so that keyvault can be access through pythonscript. The keyvault name for the task is **azkv-eus2-dev-pythn-01**.

The below screenshots are for reference.

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text, software, number, computer icon

Description automatically generated  
  
  
  
**Step 6:**

Link the key vault key’s directly into a variable to be used by the pipeline

A screenshot of a computer

Description automatically generated

**Step 7:**

**Replace tokens for databases and schema in SQL Scripts task** – This task will check for migration folder by following the project structure. In the release pipeline variables have provided with respect to instances. As per the variables it will take DB Name and schema and perform the operation for replacement only with \*.sql and \*\*/\*.sql files i.e., it will check for folders and sub folders to update.

A screenshot of a computer

Description automatically generated with medium confidence

Below variables are used with the release pipeline to make it configurable

A screenshot of a computer

Description automatically generated

**Step 8:**

**Deploy scripts to Dev using Schema change CLI** – This task is to run Schemachange CLI. As a prerequisite, it checks for python installation, snowflake python driver installation. It runs Schemachange using warehouse credentials and deploys database objects and change history table using the role specified. Upon successful completion, all DB scripts are deployed to the instance.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

#### Build Service Release Pipeline Permission:

Below diagram showsthe Permissions for release pipeline for BIDW build service(CLIENT X) and Project collection build service(CLIENT X) group

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Appendix



# Snowflake CMDB Discovery:

As per ServiceNow team, Auto-discovery of snowflake databases in CMDB is not feasible currently. However, we are working with the team to create a snowflake classification in service Now. Once the class is ready, we have opened a request with ServiceNow team to update the CMDB with snowflake database details.

