Snowflake Security and Identity & Access Management for Data Applications

This document provides security and data governance control options for anyone who is or will be developing a multi-tenant data application on Snowflake.

By Seth Youssef

Introduction

As shown in *Design Patterns for Building Multi-Tenant Applications on Snowflake*, multi-tenant Snowflake® applications typically conform to one of three design patterns:

- **Multi-Tenant Table (MTT)**: MTT consolidates tenants within a shared table or warehouse. Centralizing tenants in single, shared objects enables tenants to share compute and other resources efficiently.
- **Object Per Tenant (OPT):** OPT isolates tenants into separate tables, schemas, databases, and warehouses. Although this approach allocates individual objects to tenants, the application still operates within a single Snowflake account.
- **Account Per Tenant (APT):** APT isolates tenants into separate Snowflake accounts. Unlike OPT, each tenant within the application has its own dedicated Snowflake account.

The Snowflake Security and Data Governance section provides security and data governance controls for these patterns, including identity and access management (IAM), and data protection policies to protect your application's data tier.

Notes:

- Application providers can choose to build the access control logic within their applications; however, this white paper focuses on how to build the access control logic within Snowflake, leveraging Snowflake data security and data governance capabilities.
- The term "application provider" (AP) refers to **Powered by Snowflake**™ partners and other Snowflake customers who build their SaaS application on Snowflake.

Using this document

Before you use this document to secure your data application, please refer to the Design Patterns for Building Multi-Tenant Applications on Snowflake white paper for more details about what design pattern fits your requirements.

The document is intended for security practitioners who are looking for various security design options and the capabilities available in Snowflake to protect their data applications that use Snowflake as a data tier.

This document is intended as a reference. Readers may selectively read the sections of interest without necessarily reading the entire document.

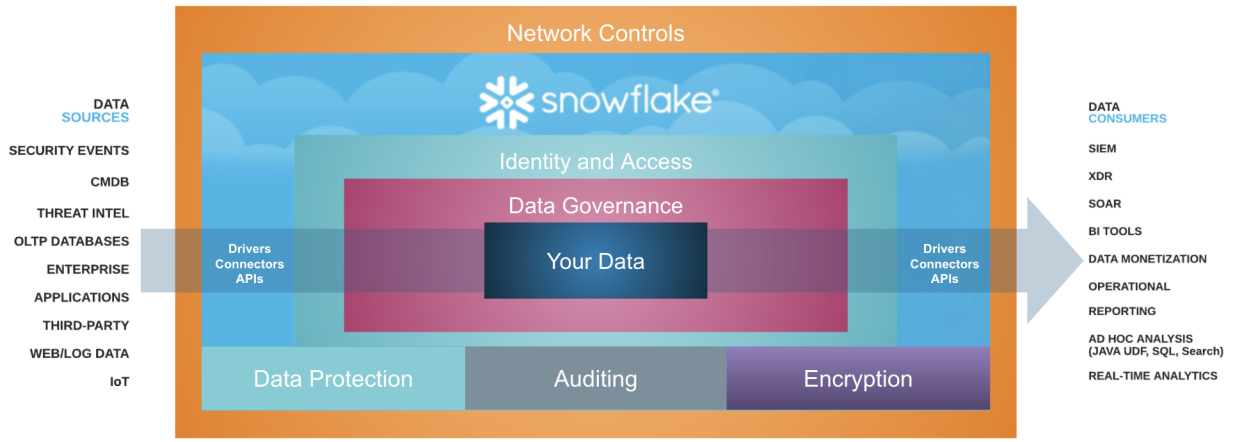
Snowflake Security and Data Governance Overview

Before we discuss the security controls for each pattern, we need to explore the capabilities that make Snowflake fit to protect your data tier.

Data security capabilities

Snowflake is built with security from the ground up. Snowflake's security team maintains the security of the Snowflake Data Cloud™ from the back end and has earned industry certifications that give Snowflake customers confidence in the platform. In addition, Snowflake provides customers with a wide range of security controls to further protect and control access to their data. Figure 1 shows Snowflake defense-in-depth security controls. Data sources and data consumers must go through all the policies and controls configured at each layer.

> **Note:** Please visit *Security Overview and Best Practices* for more information about defense in depth in Snowflake.



Achieve Compliance benchmarks and Privacy goals

Figure 1. Snowflake security defense in depth

Data governance capabilities

Snowflake provides a growing set of data governance capabilities to help customers know their data and apply protection policies according to their security, compliance, and regulatory requirements (for example, column-level encryption, tokenization, masking, anonymization, and so on) before they share or deliver data to data consumers as ready-to-query data products. All this is provided without the need to move or copy the data and while keeping visibility, security, and data governance policies enforced across the data pipeline in Snowflake and while providing auditing, monitoring, and lineage for data in Snowflake.

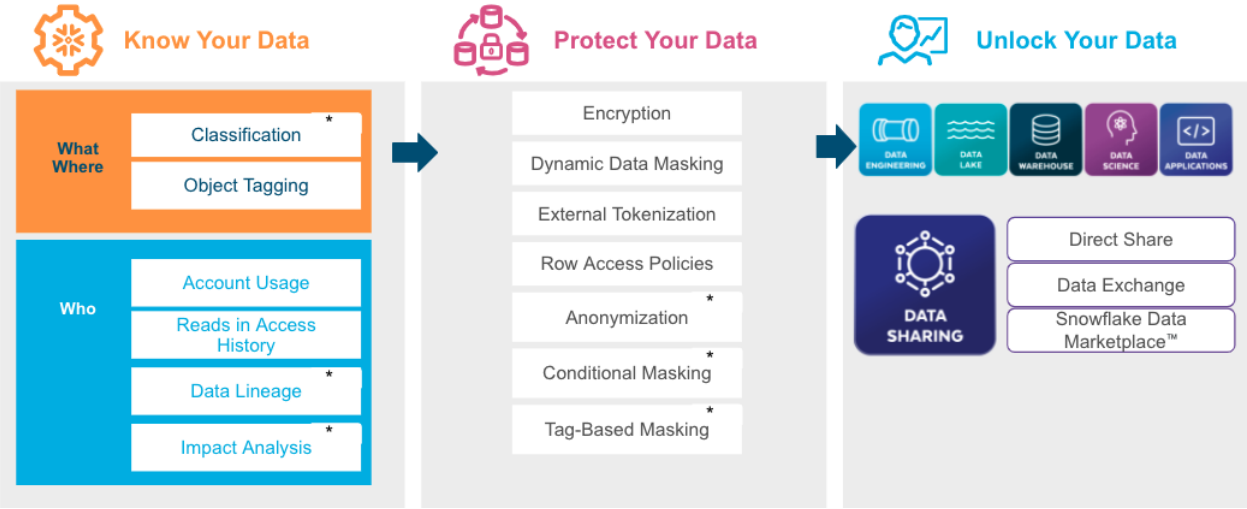


Figure 2. Snowflake data governance capabilities

Note: Capabilities marked with an asterisk (*) are in preview as of this paper's publishing.

Data privacy

Combining Snowflake platform security, customer security controls, data governance policies, and auditing and monitoring gives customers the ability to identify PII and sensitive information, and apply adequate tags and policies such as row access policies, hashing, encryption, tokenization, masking, or anonymization to the application data tier to achieve their organization's privacy and compliance goals. For more information, please visit *The Journey to Processing PII in the Data Cloud*.

Defense in Depth for Multi-Tenant Data Applications

In all the design patterns, a data app has the three main components shown in Figure 3.

- ***Snowflake*** represents the Snowflake account that provides the data tier for the data application.
- ***Application Provider*** represents the direct customer of Snowflake that owns the Snowflake account. The application provider's administrators and DevOps users need to access the Snowflake account for the provisioning and operation of the data application.
- ***End Customer*** represents the application provider's customer that consumes the data application. The end customer's users may or may not have direct access to the underlying Snowflake account.

As you can see, when the application needs to access data from Snowflake, the underlying Snowflake account can be configured with a combination of security, data governance, and privacy policies to satisfy the end customer's requirements. The remaining parts of this paper discuss in detail how those controls apply to each design pattern and their limitations, if any.

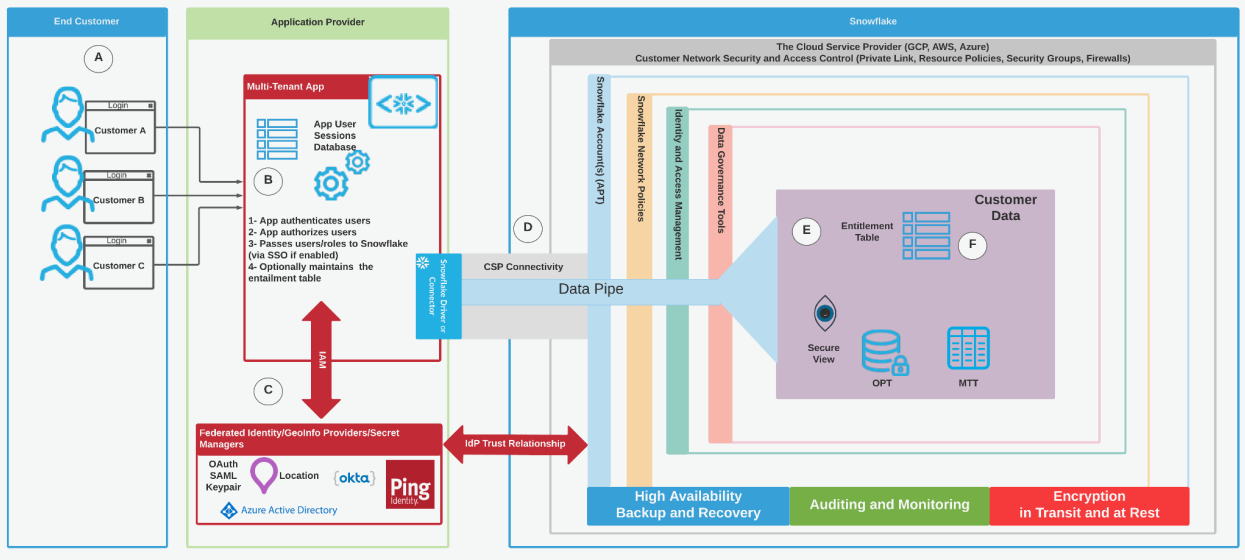


Figure 3. Snowflake data app security architecture

Figure 3 shows the following high-level workflow:

- **A:** The end customer connects to the data application via public or private connections.
- **B:** The data application authenticates to Snowflake using the following options:
  - **Option 1:** The data application connects to Snowflake as a single service account user (independent of who the end customer user is). The service account could be one account across all the tenants or it could be one service account per tenant (APT). In this case, the queries will run in Snowflake under that service account.
  - **Option 2:** The data application connects to Snowflake using single sign-on. In this case, the application provider manages the end customer's users and must provision them along with their roles into Snowflake; this can be done via system for cross-domain identity management (SCIM) or via SQL. In this scenario, the queries will run in Snowflake under the end customer's user identities and can leverage fine-grained data policies for each customer's end user.
  - **Option 3:** In the case of APT, the end customer has the option to use its own identity provider (IdP) for authentication if this is arranged with the AP.

- **C:** The authentication and authorization is handled by the data application, and there are at least two options:
    - **Option 1:** The data application has built-in native components (code to process authentication and authorization) to handle the end customer authentication and authorization.
    - **Option 2:** The data application can use an external authentication and authorization service from an external IdP that the application provider chooses—for example: Okta, Azure Active Directory (AD), Auth0, and PingFederate. The IdP provides authentication and authorization via Security Assertion Markup Language (SAML) and/or OAuth. In addition, some IdPs (such as Okta and Azure AD) can identify the end customer's geographical location, which can be shared with Snowflake to be used with location-based data policies, for example.
- **D:** The data application processes the end customer's request and establishes or uses pooled sessions over TLS 1.2 to Snowflake to retrieve the needed data. This connection could be over the internet or over the private connectivity options of the cloud service provider (CSP)—GCP, Azure, or AWS—as we will see in the Data application connectivity section.
- **E:** Snowflake processes the request based on configured network policies, IAM users/roles, and data policies.
- **F:** The entitlement table is maintained and updated by the data application and/or Snowflake external functions.

Data application connectivity

All communications with Snowflake follow the same basic patterns and channels of communication. Customer data is always encrypted in transit using TLS 1.2.

When an application provider connects to Snowflake via drivers, connectors, APIs, and Snowflake's web UI, it must connect to the following service endpoints to be able to use Snowflake:

1. **Snowflake service endpoint**, to specify the account URLs that could be public or private.
2. **Online Certificate Status Protocol (OCSP) cache endpoint**, to validate the TLS certificate. This is optional but recommended to validate the certificate.

3. **Internal stage (Snowflake managed storage)**, which is used to return the large results to the data application as well as for put/get operations on the internal stage; this connectivity can be public or private.

The communications have two main patterns: public and private connectivity options. Both are discussed in the next sections.

Public connectivity

Figure 4 shows the two options for how a data application can communicate with Snowflake's public URL over internet or over the CSP's backbone without traversing the internet:

- **Option 1, over the internet**: If the application provider's data application lives in another CSP platform than their Snowflake account(s), communication between the data application and Snowflake will be over the public internet.
- **Option 2, over the CSP's backbone**: If the application provider's data application lives in the same CSP platform as their Snowflake account(s), even though Snowflake's public URL and public IP address are used, the communication between the data application and Snowflake will be over the CSP's backbone, as stated by GCP, AWS, and Azure.

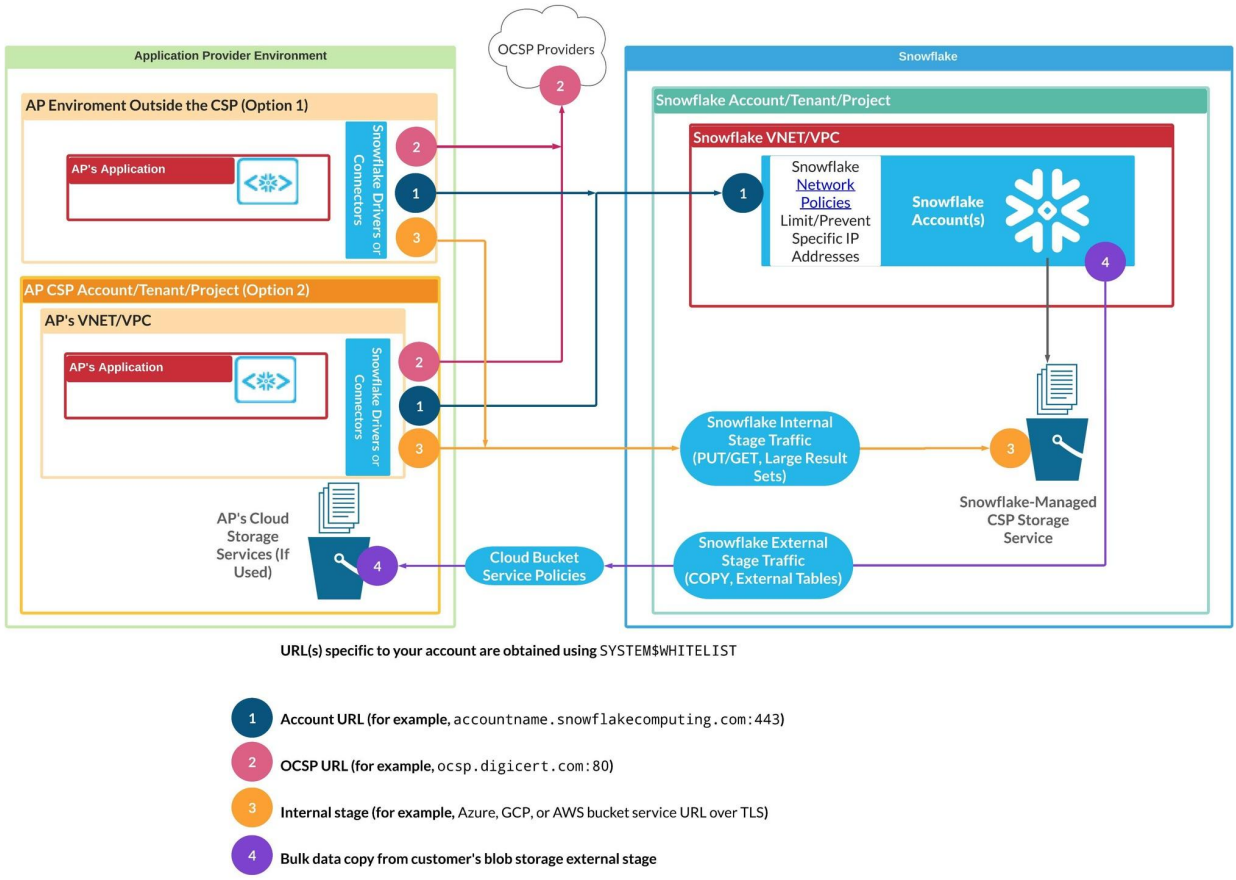


Figure 4. Public connectivity to Snowflake

The data application leverages Snowflake driver/connectors, and it uses the communication channels shown in Table 1.

Table 1: Snowflake public communication channels

Channel	Usage	Required
Channel 1	Used to access Snowflake services.	Yes
Channel 2	Used to make a certificate-revocation check via an OCSP URL.	No, but recommended
Channel 3	Used to GET/PUT files and/or download query large results sets via Snowflake-managed CSP storage services.	Yes
Channel 4	Used if the customer chooses to integrate their Snowflake account with their CSP storage service bucket(s). This bucket could be provided by the same CSP that provides their Snowflake account(s) or by another CSP (GCP, AWS, Azure).	No

Private connectivity

Snowflake's Business Critical Edition offers private connectivity access by integrating with the underlying CSP's features (AWS PrivateLink, Azure Private Link, and GCP Private Service Connect). Application providers can support this enhanced security option by creating a network topology to support it.

Figure 5 shows how a data application can communicate with Snowflake's private URL. The configurations require the application provider data application and Snowflake to be in the same CSP platform, because AWS PrivateLink, Azure Private Link, and GCP Private Service Connect do not support cross-cloud communications.

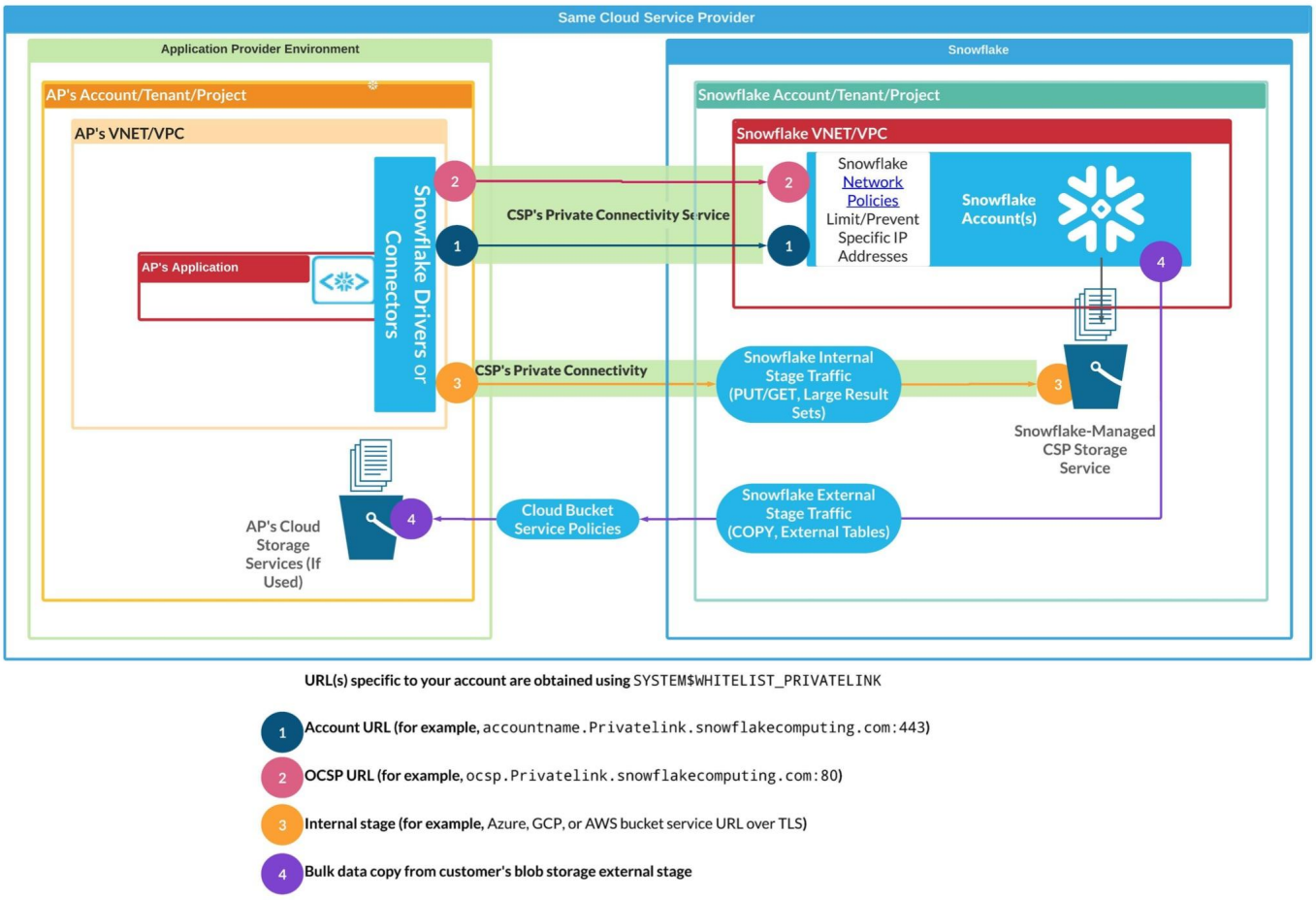


Figure 5. Private Connectivity to Snowflake

The data application leverages Snowflake driver/connectors, and it uses the communication channels shown in Table 2.

Table 2: Snowflake private communication channels

Channel	Usage	Required
Channel 1	Used to access Snowflake services over a private URL; must be in `accountname.privatelink.snowflakecomputing.com` format. Azure Private Link AWS PrivateLink GCP Private Service Connect	Yes
Channel 2	Used to make a certificate-revocation check via an OCSP private URL.	No, but recommended
Channel 3	Used to GET/PUT files and/or download query large results sets via Snowflake-managed CSP storage services. Customers can choose to enable private connectivity to the internal stage.	Yes
Channel 4	Used if the customer chooses to integrate their Snowflake account with their CSP storage service bucket(s). This bucket could be provided by the same CSP that provides their Snowflake account(s) or by another CSP (GCP, AWS, Azure).	No

Private connectivity considerations

Table 3 highlights a few private connectivity considerations seen in the field.

Table 3: Private connectivity considerations

Consideration	Notes
Public URLs Versus Private URLs	Enabling private connectivity with Snowflake does not disable the public URL; the application provider can still use the public URLs to provide access for the application provider's tools that do not support private connectivity. Application providers can cut off public URL access by enabling network policies denying access from public IP addresses and by only allowing connections via the private IP address ranges.

SCIM	SCIM providers that run inside the application provider's virtual private clouds (VPCs) or virtual networks (VNets) or on premises can use the private connectivity options. However, SCIM SaaS providers such as Azure AD and Okta cannot run from inside the application provider's VNets/VPCs; therefore, they have to leverage the public URL to provide automatic users/roles provisioning. The application provider can restrict this public access by using SCIM-level integration network policies.
SAML	SAML can be configured with either a Snowflake public endpoint or a private endpoint, but not both. However, this limitation does not apply to the application/tools that use External OAuth to access Snowflake.
SaaS-to-Snowflake Communications	As mentioned with SCIM, SaaS and client tools that do not live in the application provider's VPCs/VNets or on premises can always use Snowflake public URLs.
APT-Based End Customer Access	Application providers using APT can provision the private endpoints to connect privately to Snowflake account(s). If the end customers want and are allowed to access privately the underlying Snowflake account(s), they can work with the application provider to enable direct private connectivity from the end customer accounts/tenants/projects, or they can access Snowflake via the application provider's VNets/VPCs if it is allowed by the application provider.
DNS	When using private connectivity, the application provider must create the private DNS zones for the domain names, as shown in each application provider's Snowflake accounts' SYSTEM$WHITELIST_PRIVATELINK.
AWS PrivateLink, Azure Private Link, and GCP Private Service Connect (PSC) Connectivity	AWS PrivateLink, Azure Private Link, and GCP Private Service Connect services work within the same cloud provider only, which means, for instance, to use AWS PrivateLink with Snowflake , the application provider must have a VPC in AWS and the application provider's Snowflake account must be in AWS.
AWS PrivateLink	AWS private link is a regional service, which means the application provider's VPC and Snowflake must be in the same region to use AWS PrivateLink. However, the application provider can use VPC peering and Amazon Transit Gateway to provide cross-regional access , or even a VPN to provide cross-cloud access to the application provider's VPC that hosts the Snowflake private link endpoints.

Azure Private Link	In the case of Azure, the private link is cross-regional, which means the application provider's VNet and Snowflake account can be in different regions.
GCP Private Service Connect	GCP PSC is regional and does not support on-premises access. However, GCP offers solutions to provide such a proxy or source NAT.
Private Connectivity with VPNs	If you have a cross-cloud environment and would like to leverage private connectivity, VPNs (AWS, GCP, Azure) help to connect on-premises environments and provide cross-cloud connectivity over the public internet, for instance, between the application provider's AWS VPC and the application provider's Azure VNet or between the application provider's on premises environment and the CSP.
Private Connectivity on Premises	AWS Direct Connect, Azure ExpressRoute, or GCP Cloud Interconnect can connect the CSP without traversing the public internet. For instance, the application provider's on-premises network and AWS can be connected using AWS Direct Connect.

Note: The aspect and design considerations of each of the connectivity options are outside the scope of this document. Please consult your CSP's design guides and best practices if such design options are required.

Data application network policies

Snowflake network policies can be applied at three different levels:

- **Account-level network policies:** These policies apply at the account level and affect all the sessions connected to that particular Snowflake account.
- **Integration-level network policies:** These policies apply at the integration object level, for example, to SCIM and Snowflake OAuth integration objects.
- **User-level network policies:** These policies apply to the user that is accessing the Snowflake account(s).

Table 4 provides more details.

Table 4: Network policy considerations

Design Patterns	Account-Level Policies	Integration-Level Policies	User-Level Policies
MTT	One active policy per Snowflake account; the application provider usually maintains the network policies.	One active policy per integration object; the application provider usually maintains the network policies.	One active policy per user object; the application provider usually maintains the network policies.
OPT	One active policy per Snowflake account; the application provider usually maintains the network policies.	One active policy per integration object; the application provider usually maintains the network policies.	One active policy per user object; the application provider usually maintains the network policies.
APT	One active policy per Snowflake account/tenant. The application provider can configure per-tenant network policies or delegate policy creation to the end customers and end customers maintain their own network policies if they are allowed to access the underlying Snowflake account(s).	One active policy per integration object. The application provider can configure per-tenant network policies or delegate policy creation to the end customers and end customers maintain their own network policies if they are allowed to access the underlying Snowflake account(s).	One active policy per user object. The application provider can configure per-tenant network policies or delegate policy creation to the end customers and end customers maintain their own network policies if they are allowed to access the underlying Snowflake account(s).

Network policies IP address considerations

The IP addresses that are used in any network policies are related to the tools that connect to Snowflake and to the networking and security solutions between the tools and Snowflake:

- Most of the time, the network policies' IP addresses are:
  - The IP addresses of the application provider's data application.
  - For user and role provisioning, the integration network policies' IP addresses are the IP addresses of the SaaS, SCIM provider, and so on that need to be integrated with the Snowflake account(s).
- In a few cases:
  - The network policies' IP addresses are the IP addresses of the other application provider's tools that access Snowflake directly for auditing and monitoring, for example.
  - The network policies' IP address can be the IP addresses of the end customer's tools if the tools are allowed to access Snowflake directly.
- In some scenarios, the application providers or the end customers use NAT gateways, firewalls, or proxy servers when they access Snowflake; in that case, the network policies must consider the IP addresses of the security solutions that sit between the data tools and the Snowflake accounts.

Data application IAM design options

Snowflake drivers and connectors support multiple authentication methods, including native username and password, public/private key pairs, and federated authentication and authorization (for example, single sign-on) via OAuth and SAML.

When building a data application with Snowflake, application providers can choose from two IAM design patterns:

- **Service account:** In this case, the data application uses a service account to access Snowflake. From Snowflake's perspective, all the queries are done with service account(s). The end customer user's identity is not visible to Snowflake.
- **Single sign-on (SSO):** In this case, the data application uses the end customer user's identity via OAuth to connect to Snowflake and run the queries. The end customer user's identity is visible to Snowflake.

Note: Snowflake supports multiple authentication methods per user. For instance, the service account user can use username/password, key pair, and OAuth authentication methods at the same time.

User and role provisioning

With both options, service accounts and/or SSO, the users and the roles must exist in the Snowflake user database. The application provider can provision users and roles manually, via SCIM, or via SQL using Snowflake drivers and connectors.

Snowflake supports multiple SCIM providers per Snowflake account adding flexibility for the application provider to provision users and roles from different sources, for instance:

- The application provider can use one SCIM provider to provision users and roles that are required to operate the platform.
- Each tenant can use their own SCIM provider to provision any additional users and roles if required for their tenant application functionality.

It is advisable to create two sets of logical roles:

- **Functional roles:** A functional role hierarchy contains only users or other functional roles.
- **Access roles:** Access roles contain only privileges. Access roles bundle sets of permissions to be assigned to the functional roles.

Then you can grant access roles to the functional roles. This strategy allows the functional role hierarchy to mimic the IdP groups hierarchy and it simplifies the RBAC design.

For more details, visit Snowflake Security Overview and Best Practices.

For more information about using RBAC with multi-tenant applications, refer to Design Patterns for Building Multi-Tenant Applications on Snowflake.

Data application with service account

The service account in Snowflake is a normal user account that is selected to be used by the data application to access the Snowflake account. The service account usually is used for programmatic access without human interaction; therefore, the authentication options are typically one or a combination of the following (see Table 5):

- Snowflake built-in username/password
- Public-Private key pair
- OAuth

The data application can use one or more service accounts per tenant, and each tenant can have one or more roles assigned to them depending on t6he application's access requirements.

Table 5: Data application with service account authentication options

Authentication Method	Requirements	Considerations
Native Username and Password	Provision service accounts and their roles in Snowflake.	Not recommended unless the data application or the tool that connects to Snowflake does not support any other stronger form of authentication. If a password is required because no other option is available, that password should be very complex (strong) and be rotated as often as possible.
Key pair	Provision service accounts and their roles in Snowflake. Generate the key pairs and assign the public key to each user.	It is recommended to use a secret manager to rotate the keys regularly. See Snowflake Service Account Security, Part 1 for more information on how to protect service accounts in Snowflake.
OAuth	Provision service accounts and their roles in Snowflake. The external identity provider (IdP) must support OAuth and must be integrated with Snowflake. Snowflake supports multiple OAuth IdPs per Snowflake account at the same time.	Snowflake supports OAuth over both private and public URLs at the same time. In this case, OAuth authenticates for the service account. It does not authenticate the end customer's users (that will be discussed next).

Note: There are some cases, especially in the case of MTT, where the application provider will use a single service account across all the tenants to optimize resource sharing. In this case, creating data policies based on tenant/roles won't be possible.

Data application with single sign-on

With this option, the data application passes the end customer's user's identity to Snowflake via OAuth. The application can authenticate the end customer's user via any means of authentication supported by the data application; then the data application requests the token from the OAuth IdP on behalf of the authenticated end customer's user.

This design pattern has the following considerations:

- The data application and Snowflake must trust the same OAuth IdP provider. For more information, refer to this information on integrating the Snowflake account(s) with the OAuth provider.
- The end customer's users and their roles must be provisioned in Snowflake.
- Snowflake supports OAuth over both private and public URLs at the same time.
- Snowflake supports multiple OAuth providers per Snowflake account. Therefore, each tenant can bring their own IdP.

Note: In the majority of current use cases, the application provider manages and provisions the IdP. There are some use cases, especially in the case of the APT pattern, when end customers can bring their IdP to integrate directly with Snowflake.

Data application and IAM summary

Table 6 compares using service accounts versus SSO from a security point of view.

Table 6: Data application IAM design option comparison

Considerations	Service Accounts	Single Sign-On
User Provisioning	One or more service accounts per tenant need to be provisioned. You can create policies per service account(s)/per tenant.	All the end customer's users in addition to any additional required service accounts need to be provisioned. You can create policies per the end customer's user attributes.
Role Provisioning	One or more roles per tenant need to be provisioned.	One or more roles per tenant need to be provisioned.
Row Access Policies	These are per-tenant policies. The entitlement table could provide additional granularity.	SSO provides more-granular authentication because control is per end customer user rather than per end customer, and the entitlement table could provide additional granularity.
Column-Level Security	These are per-tenant policies. The entitlement table could provide additional granularity.	SSO provides more-granular authentication because control is per end customer user rather than per end customer , and the entitlement table could provide additional granularity.
Auditing and Monitoring	All the queries are run with the service account(s) identities.	The queries run with the end customer's user identity.
Authentication Methods	• Username and password • Key pairs • OAuth (for the service accounts)	OAuth
MTT	Snowflake can provide data isolation between tenants via RBAC, row access policies, dynamic data masking and secure views/UDFs while the application provides isolation	Snowflake can provide data isolation between tenants as well as between the end customer's users via RBAC, row access policies, dynamic data masking, and secure

	between end customer's users.	views/UDFs.
OPT	Snowflake provides object-level and data-level isolation between tenants via RBAC, row access policies, dynamic data masking, and secure views/UDFs while the application provides isolation between the end customer's users.	Snowflake can provide object-level and data-level isolation between tenants as well as between the end customer's users via RBAC, row access policies, dynamic data masking, and secure views/UDFs.
APT	Snowflake provides account-level isolation in addition to object-level and data-level isolation between tenants while the application itself provides isolation between the end customer's users.	Snowflake can provide account-level, object-level, and data-level isolation between tenants as well as between the end customer's users.

The entitlement table

The entitlement table (see Figure 6) is an option that application providers can use for fine-grained data access and data protection policies. The entitlement table can be used by Snowflake row access policies and column-level security policies to control access to customer data based on the attributes collected and stored in the entitlement table.

Note: Entitlement table considerations for each data application design pattern will be discussed in their respective sections.

Adopting the entitlement table involves the following tasks:

- Selecting the appropriate data sources for the entitlement table.
- Maintaining and updating the entitlement table.
- Building user-defined functions that use the entitlement table to match conditions and return values to be used in row access policies and column-level security.

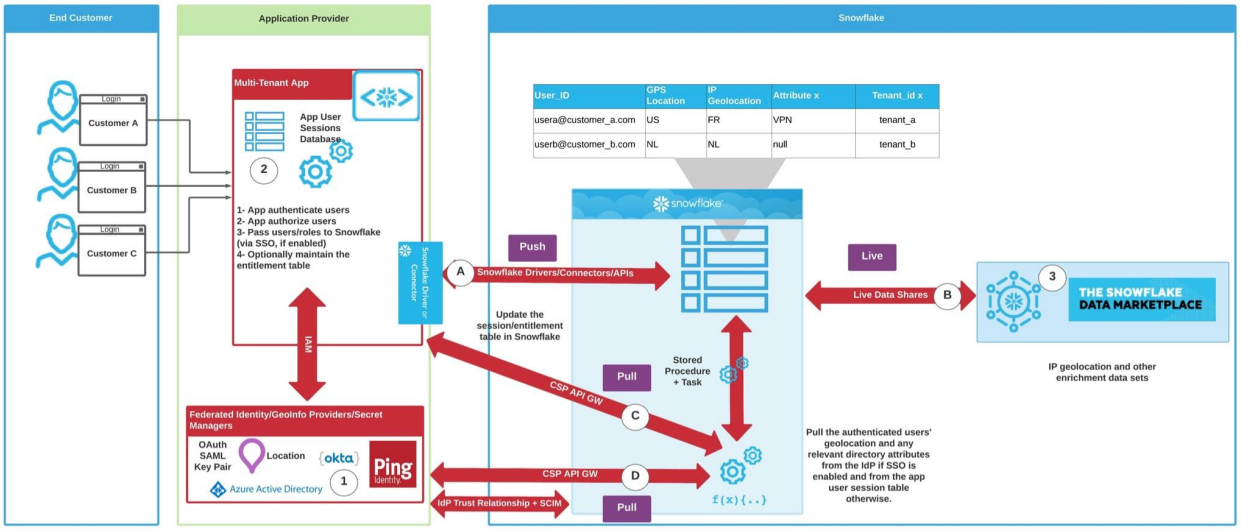


Figure 6. Entitlement table

Entitlement table data sources

As shown in Figure 6, the entitlement table data sources can be (but are not limited to) one or a combination of the following:

1. **The identity provider (IdP):** If the data application is integrated with an IdP to provide end customer user authentication and authorization, the IdP stores the end customer user's authentication details. If the application uses Okta or Azure AD, these details include user IDs, the authentication method, IP addresses, IP address

geolocation data, and GPS geolocation data. If you use a different identity provider (on premises or in the cloud), the same concepts apply as long as your IdP provides the required information via a callable API interface.

2. **The application provider's data application users session database:** Depending on the data application design, the session database may store the end customer user's information for example, `tenant_id`, `user_id`, `location`, and any other attributes that are needed for data application functionality.
3. **Snowflake Data Marketplace™:** The data sets from Snowflake Data Marketplace are shared live, up to date, and are maintained and updated by the data providers. Here some examples of data sets:
    - To create network policies and/or apply column-level security/row access policies based on the session IP address geolocation (aka GeoIP), use IPinfo's IP Geolocation data set.
    - To detect attempts to bypass the GeoIP restrictions, you could use IPinfo's Address Privacy Detection data set, which helps with detecting whether a device's real IP address is hidden behind a public VPN, proxy, TOR server, and/or hosting provider. Use this information to block that IP address from using network policies and/or to apply column-level security/row access policies.
    - To build a cyberfence around the Snowflake account(s), use the Red Sky Alliance CTAC Cyber Threat Intelligence data set, which helps with detecting a known attack bot's source IP addresses, email addresses, and so on. Use this information to block those IP addresses from using network policies or to disable matching email addresses if they exist in the system. For more information about CTAC, see the CTAC Guide.

Note: Most of the time, Snowflake will see the data application IP addresses, not the end customer's user IP address, as detailed in the Data application network policies section.

You can add any data set that fits your security and compliance requirements. Please refer to the Data application network policies section to learn more about IP address considerations.

Maintaining and updating the entitlement table

After selecting the appropriate data sources, the next step is to select the appropriate entitlement table maintenance process(es), which could be one or a combination of the following as shown, in Figure 6.

- **A (Push):** The data application updates the entitlement table once the end customer users are created via signup, subscription, or any end user

provisioning processes. The data application will use Snowflake's established connection to push the entitlement table updates.

- **B (Live):** Via Snowflake Data Marketplace, this is a live share and does not need maintenance or to be updated.
- **C: (Pull):** The application provider's Snowflake account(s) will use a combination of:
    - External functions to call the data application APIs (if they exist) and collect the end customer user's session database.
    - A stored procedure to read and clean the user session database and update the relevant rows in the entitlement table.
    - A task to call the stored procedure automatically every specified period of time, for example, every minute.
- **D (Pull):** This works the same way as (C), but instead of calling the data application API, the external functions call the IdP's API. This option is useful if the data application uses an external IdP to provide user authentication and authorization services.

Using the entitlement table

After selecting the appropriate data source(s) and the table maintenance procedure(s), the entitlement table can be used to create policies to provide data protection and access control based on end customer users.

Data policies

You can create column-level security and row access policies based on the entitlement table attributes or, even better, you can create UDFs to match the user's attributes and then apply the appropriate actions (see Figure 7), for example:

- Row access policies to return the rows or not
- Hashing
- Encryption
- Tokenization
- Dynamic data masking

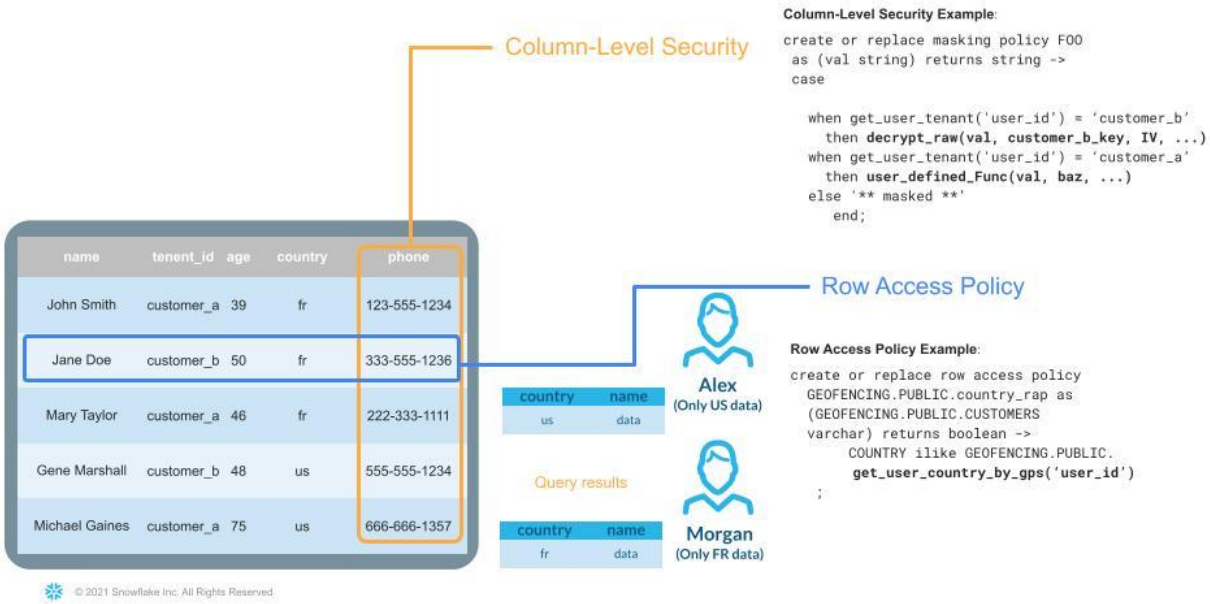


Figure 7. Row access policies and column-level security

Now the policy granularity will depend on the chosen IAM design; the two choices were described in the Data application with service account and Data application with single sign-on sections. If you choose to use SSO, the policies can be applied at the end customer's user

level, where each user can, for example, use the encrypt()/encrypt_raw() or decrypt()/decrypt_raw() functions on their related data rows using their unique encryption keys, as described in Data policy examples section. The same concept applies for other data protection methods such as hashing, tokenization, dynamic data masking , and so on.

> **Note:** Data encryption with the encrypt()/encrypt_raw() or decrypt()/decrypt_raw() functions is on top of the default transparent encryption provided by Snowflake.

Data policy examples

Figure 8 shows a policy architecture example where the application provider's Snowflake account(s) will:

1. Call the end customer or the application provider's secret manager to programmatically fetch the encryption key materials:
    a. In the case of a data application with a service account, the key will be the tenant-level key.
    b. In the case of a data application with single sign-on, the key could be either the tenant key or the specific user encryption key.
2. Snowflake row access policies will use the entitlement table to filter out the rows based on the tenant, the end customer's user ID, the GPS location, or any other criteria stored in the entitlement table.
3. Snowflake column-level security will decrypt the specific rows using the key specified in step 1.

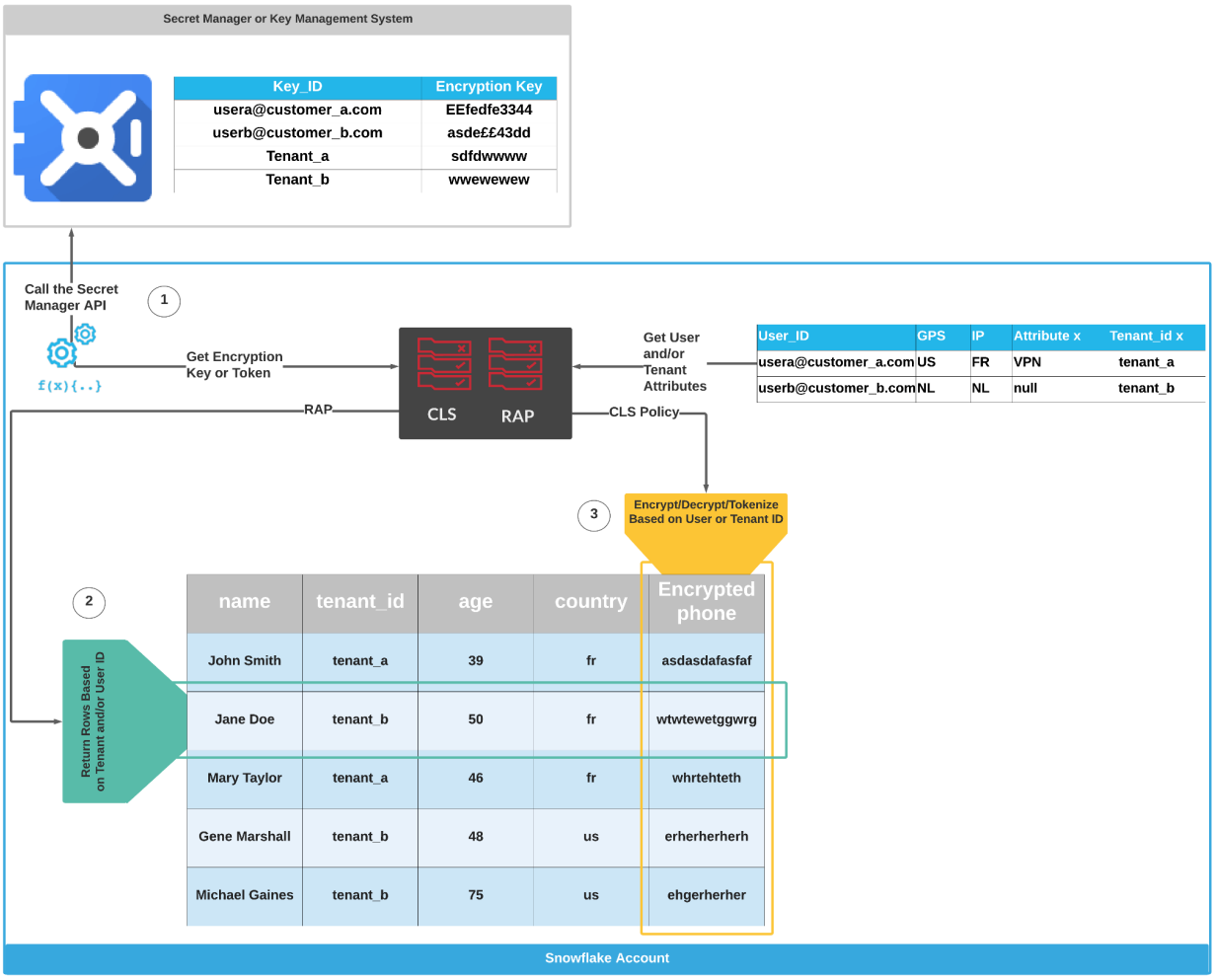


Figure 8. Row access policy (RAP) and column-level security (CLS) architecture example

Example: End customer's user location-based data policies

This section shows how to use the entitlement table to build geo-based data policies. This example uses the following:

A. When loading the data into Snowflake, the customer email address column is encrypted with a unique user encryption key (see Figure 9).

B. In this example, the data application uses single sign-on; therefore, the end customer's user identity is visible to the underlying Snowflake account(s).

C. Snowflake updates the entitlement table regularly with the user's GPS location via a stored procedure, a task, and then an external function that connects to Azure AD to pull the latest user audit logs, including the GPS location of the authenticated user (see Figure 10).

D. The Snowflake row access policy filters out the data based on the GPS location; for instance, if the end customer's user is in France, then the user in France will see only the French data (Figure 11).

E. The Snowflake dynamic data masking policy decrypts the email address using the user encryption key (see Figure 12). The encryption materials are fetched from a secret manager that lives in the end customer's environment or in the application provider's environment by a Snowflake external function.

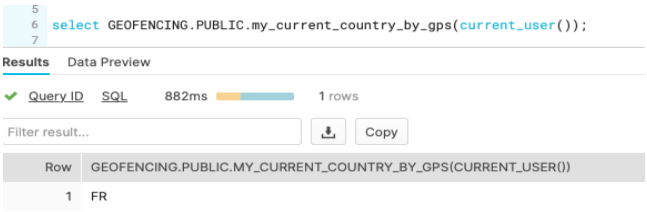
F. Figure 13 shows sample results of the decrypted email addresses.

Query ID SQL 3.66s 914,790 rows

Filter result... Copy

Row	COUNTRY	C_BIRTH_YEAR	C_SALUTATION	ENCRYPTED
1	FR	1958	Sir	BC9BD817BAEFF02BBDE78D4E49F9285080BBCD3147A14938528A23E8982A736AD38D1CB63A51229EC2AA567A68D5D3CD2F
2	FR	1957	Ms.	E5A8F50B4B9C750840B266E38A09FB4A3BBEBCCCAA3E03C62C352886F7DDCC9AFFAA3A32B72468BC37E452026E854544B27BD66E
3	FR	1965	Mr.	87BFDAFB63FE6638153A00BCF6DD2ECDB032C7D455AF464C0491A93AA955404BC1D08653F02A155F53E72A481DC48374AD8C13DD260C15092ECE
4	FR	1965	Dr.	89C96E2B267D4418701D6E184357BEA1EF3E37EE237F60D620A8CBD5E0605E3321EA184AA17B6E315D2E79A63B0E8BE0
5	FR	1930	Mr.	7C4033FD3C99520939EB4EDFB495BBFFA5870BE6F2BFB21882002EB36EBCE69528448B59166695953FA3285447838252034238 8CB9002D428FADEBF427CCE5
6	FR	1991	Dr.	318B74D49D6332B7DB9D810FD06BC3172F129D0848DAB14A401216A30F044FC758DC158BE7F064F221C864F4AB839ADAFBEBAC86AEA3895997893EC22B6C4A

Figure 9. Column-level encryption when (or before) data is loading



```
select GEOFENCING.PUBLIC.my_current_country_by_gps(current_user());
```

Results Data Preview

Query ID SQL 882ms 1 rows

Filter result... Copy

Row	GEOFENCING.PUBLIC.MY_CURRENT_COUNTRY_BY_GPS(CURRENT_USER())
1	FR

Figure 10. Getting the current user's location

```
create or replace row access policy GEOFENCING.PUBLIC.PBS_SEC_RAP as (COUNTRY varchar) returns boolean ->
            COUNTRY ilike  GEOFENCING.PUBLIC.my_current_country_by_gps(current_user())
   ;
alter table GEOFENCING.PUBLIC.CUSTOMERS_MTT add row access policy GEOFENCING.PUBLIC.PBS_SEC_RAP on (country);
```

Figure 11. Geo-based row access policies

```
create or replace masking policy GEOFENCING.PUBLIC.PBS_SEC_DDM_ENC
   as (val VARCHAR) returns VARCHAR ->
   case
      when GEOFENCING.PUBLIC.my_current_country_by_gps(current_user()) = 'FR'
           then to_varchar(DECRYPT(to_binary(val),GEOFENCING.PUBLIC.GET_MY_ENCR_KEY(current_user())),'utf-8')
      when GEOFENCING.PUBLIC.my_current_country_by_gps(current_user()) = 'US'
           then to_varchar(DECRYPT(to_binary(val),GEOFENCING.PUBLIC.GET_MY_ENCR_KEY(current_user())),'utf-8')
   end;

alter table if exists GEOFENCING.PUBLIC.CUSTOMERS_MTT modify column ENCRYPTED set masking policy GEOFENCING.PUBLIC.PBS_SEC_DDM_ENC;
```

Figure 12. Dynamic data masking to decrypt data based on the current user's GPS location

```
16
17 SELECT COUNTRY, C_BIRTH_YEAR,C_SALUTATION, ENCRYPTED  FROM GEOFENCING.PUBLIC.CUSTOMERS_MTT
18
```

esults Data Preview

Query ID SQL 1.34s 457,289 rows

Filter result... Copy

Row	COUNTRY	C_BIRTH_YEAR..	C_SALUTATION	ENCRYPTED
1	FR	1954	Sir	Ian.Briggs@DzbMQC.org
2	FR	1959	Mrs.	Jacqueline.Chapman@ud1BkpTGIfal80.edu
3	FR	1931	Miss	Nicole.Brewer@DI98nH5d.com
4	FR	1983	Dr.	Fannie.Stine@S4T5NE0lkS0.com
5	FR	1950	Miss	Kristin.Daniels@qfqxvL4.com

Figure 13. Testing the policies

> **Note:** In this example, the encrypt()/encrypt_raw() functions were used with the decrypt()/decrypt_raw() functions; however, you can use any other built functions, external tokenization, and user-defined functions.

Per-tenant data policies

The data applications in this scenario use service account(s) to connect to the underlying Snowflake account(s). Therefore, all the data application queries will run under the service account(s) identity. The entitlement table has the service account IDs. The policies granularity will be at the level of the service account/role.

This case is similar to location-based end customer's user data policies example. The difference here is instead of creating the end customer's user-level policies, the policies will be applied at the tenant level via its related service account(s) and roles. When the tenant data is loaded into Snowflake, the tenant's data rows in the sensitive column will be encrypted using a unique tenant key.

Snowflake data encryption

All editions of Snowflake encrypt customer data automatically using AES 256 out of the box. By default, Snowflake creates, manages, uses, and rotates the encryptions keys and their hierarchy, as shown in Figure 14. However, application providers can choose to bring their own key to participate in data encryption at rest, which is discussed in the Bring your own key (Tri-Secret Secure) section.

> **Note:** Snowflake encryption covers micro-partition encryption. On top of that, the application provider can choose to use extra hash, encryption, or tokenization for specific columns and rows using different encryption keys outside of the default encryption key hierarchy shown in Figure 14, as discussed in the Example: End customer's user location-based data policies section.

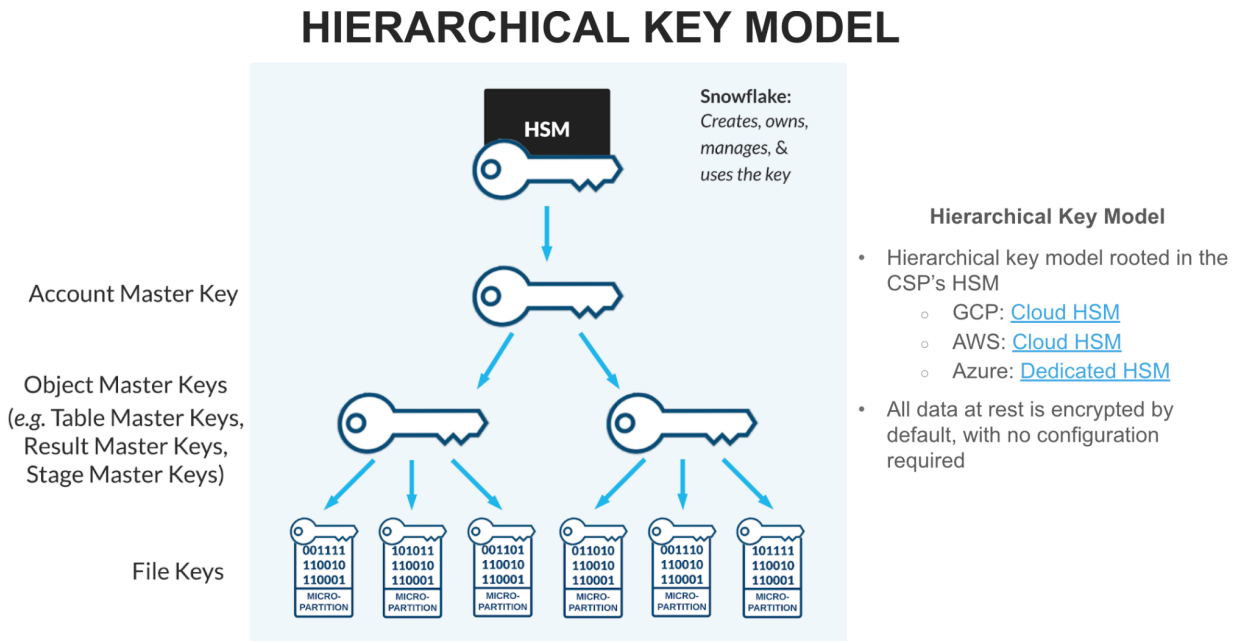


Figure 14. Snowflake default encryption for the hardware security model (HSM)

Bring your own key (Tri-Secret Secure)

Snowflake's Business Critical Edition and above support bringing your own key (BYOK) to encrypt the data at rest; in Snowflake this is called Tri-Secret Secure (TSS). TSS is a Snowflake account-level feature, which means you can turn on TSS at the account level only. Table 7 lists some design considerations.

The end customer's or the application provider's master key(s) must be in the CSP's key management system (AWS KMS, Azure Key Vault, Google Cloud KMS) and must be in the same CSP platform as the application provider's Snowflake account(s).

In this case, the application provider owns, creates, and manages the customer master key(s) that will be used to generate the new account master key, as shown in Figure 15.

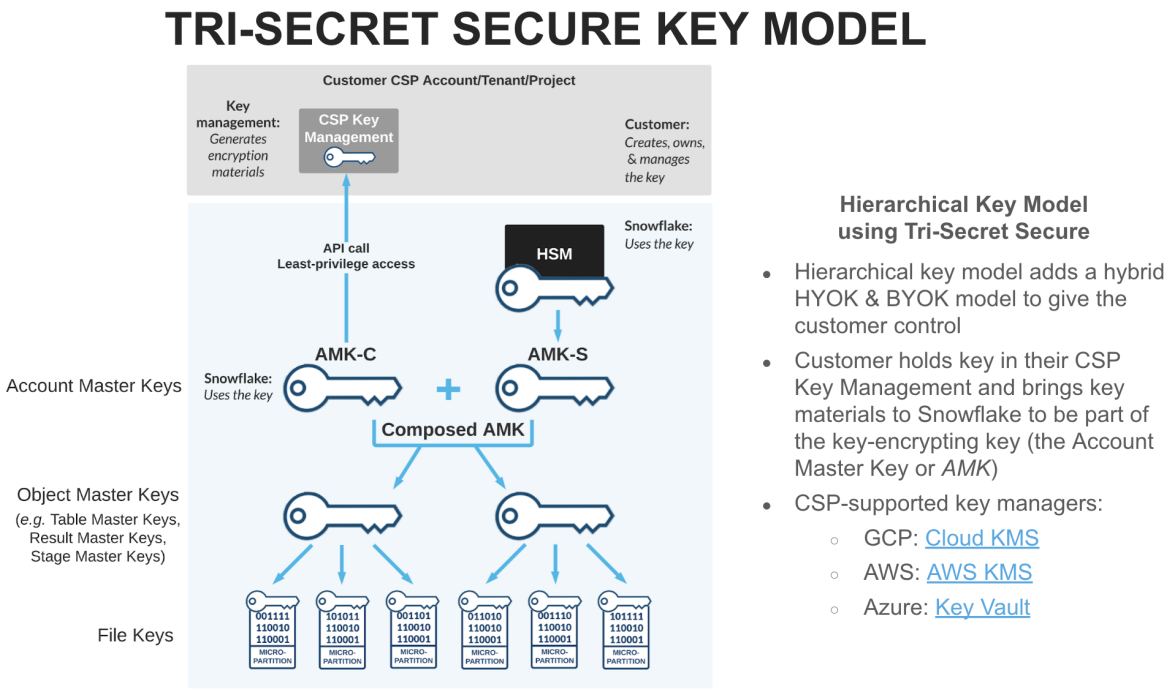


Figure 15. Bring your own Key and hold your own key (HYOK) with Tri-Secret Secure

Table 7: Data application design patterns and TSS considerations

Design Pattern	Considerations	Per-Tenant TSS Key
MTT	All of the data application's tenants can use TSS at the account level. Usually the KMS/key vault lives in the application provider's CSP account/tenant/project.	No
OPT	All of the data application's tenants will use one single BYOK at the account level. Usually the KMS/key vault lives in the application provider's CSP account/tenant/project.	No
APT	Each Snowflake application provider account may or may not have TSS enabled. There two options for the adopters of APT: ● Option 1: The key is owned and managed by the AP.	Yes

	• Option 2: The key is owned and managed by the end customer. In this case, the end customer must have an environment in the same CSP environment as the application provider's Snowflake account. The end customer should work with the application provider to enable the feature for them.	

Note: Even with the MTT and OPT design patterns, the end customers can use column-level security to encrypt or tokenize their records in the application provider's Snowflake account(s), as discussed in the Data policies section.

Auditing and monitoring

When building multi-tenant applications, it is recommended that you have full security visibility over your application tiers. This section covers how you can integrate the application provider's Snowflake accounts with the overall auditing and monitoring solutions you may have in place.

Snowflake logs all the user interactions with Snowflake in a set of history objects (Account_Usage) that you can access and use in multiple ways, as shown in Figure 16.

Your SIEM; Security Orchestration, Automation, and Response (SOAR); or extended detection and response (XDR) technology can be integrated directly into the Snowflake accounts via one of the Snowflake drivers or connectors to access the history objects, or you can export the audit logs to a cloud storage service to be ingested into your auditing and monitoring tools.

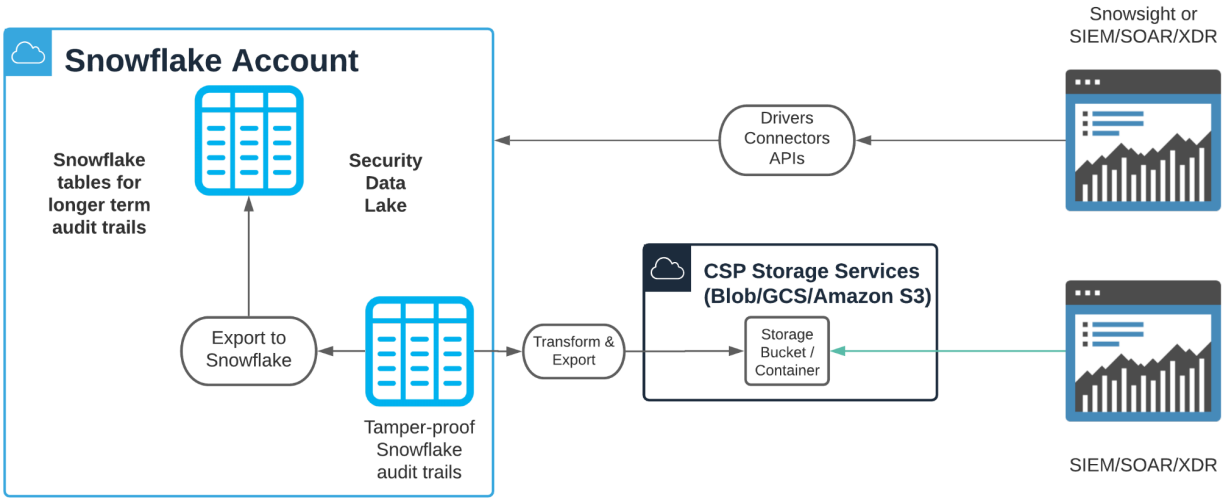


Figure 16. Integrating Snowflake with auditing and monitoring solutions

Account usage and information schema

Account usage history objects have some latency, while information_schema table functions do not have latency. The application provider can create views to vertically join data based on event time stamps from both data sources and provide full real-time visibility, as shown in the following example, which combines the login_history account usage view with the login_history information schema table function:

```
CREATE OR REPLACE VIEW GEOFENCING.PUBLIC.REAL_TIME_LOGIN_HISTORY AS
WITH cst_is AS
(
SELECT event_id
    , event_timestamp
    , event_type
    , user_name
    , client_ip
    , reported_client_type
    , reported_client_version
    , first_authentication_factor
    , second_authentication_factor
    , is_success
    , error_code
    , error_message
    , related_event_id
    , 'Information Schema' record_source
 FROM TABLE(snowflake.information_schema.login_history())
)
, cst_au AS
(
SELECT event_id
    , event_timestamp
    , event_type
    , user_name
    , client_ip
    , reported_client_type
    , reported_client_version
    , first_authentication_factor
    , second_authentication_factor
    , is_success
    , error_code
    , error_message
    , related_event_id
    , 'Account Usage' record_source
 FROM snowflake.account_usage.login_history
WHERE (event_timestamp, event_id) NOT IN (SELECT event_timestamp, event_id FROM cst_is)
)
SELECT * FROM cst_is
UNION ALL
SELECT * FROM cst_au
;
```

Snowflake's ACCOUNT_USAGE has many views to store audit logs for various aspects of the user activities with the Snowflake account. The following ACCESS_HISTORY view, LOGIN_HISTORY view, and QUERY_HISTORY view sections describe some of the views that stand out from a security perspective.

ACCESS_HISTORY view

The ACCESS_HISTORY view provides the application provider with the following capabilities (see Figure 17):

- Discovering unused data to determine whether to archive or delete the data
- Validating data changes to notify users prior to dropping or altering a given table or view
- Auditing data access to comply with regulatory requirements and data governance initiatives

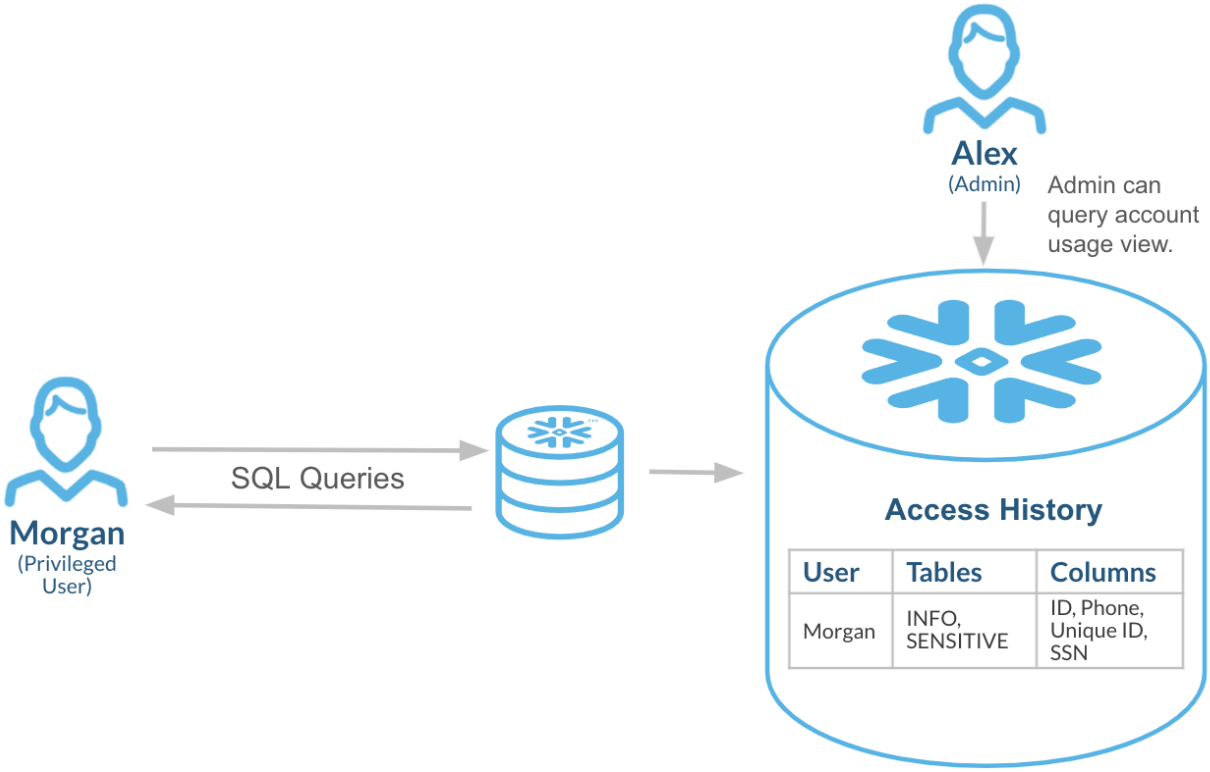


Figure 17. Access History view

LOGIN_HISTORY view

The LOGIN_HISTORY view records login attempts to Snowflake accounts including the time, login IP address, driver version, authentication methods, and more. This helps the application provider to audit and monitor the application provider and end-customer users access to Snowflake. Further, using a three-way JOIN between LOGIN_HISTORY, sessions, and QUERY_HISTORY, you can determine how a user session that was used for a given query was authenticated.

Figure 18 shows an example of a dashboard built using LOGIN_HISTORY audit logs.

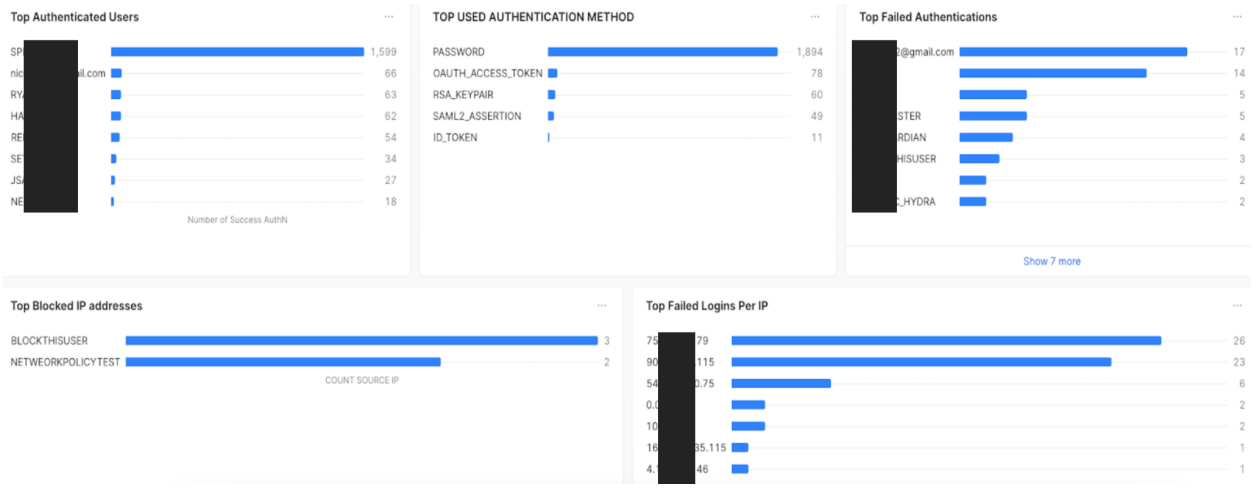


Figure 18. Dashboard built from data from the LOGIN_HISTORY audit logs

QUERY_HISTORY view

The QUERY_HISTORY view provides audit logs for queries executed in a Snowflake account including time range, session, user, role, warehouse, and so on. For instance, ACCOUNTADMIN roles should be used for daily account operations. Figure 19 shows an example of a dashboard built in Snowsight® that highlights that.

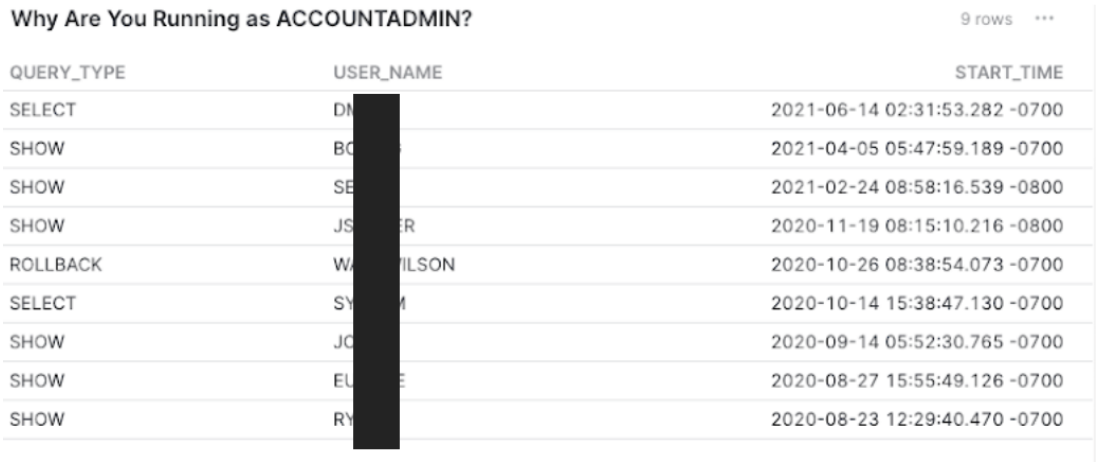


Figure 19. Dashboard built from QUERY_HISTORY view

Auditing and monitoring options

The auditing and monitoring solution can be done at two different levels (see Figure 20):

A. Every Snowflake account in the application provider's ecosystem can be integrated individually with the application provider's auditing and monitoring tools.

B. The application provider can use a dedicated Snowflake account for aggregating all the audit logs for all the data application's Snowflake accounts. With this option, the application provider creates secure views on the top of the account usage and information schema to provide a real-time audit trail and then uses Snowflake's secure data sharing to share the auditing and monitoring secure views across accounts.

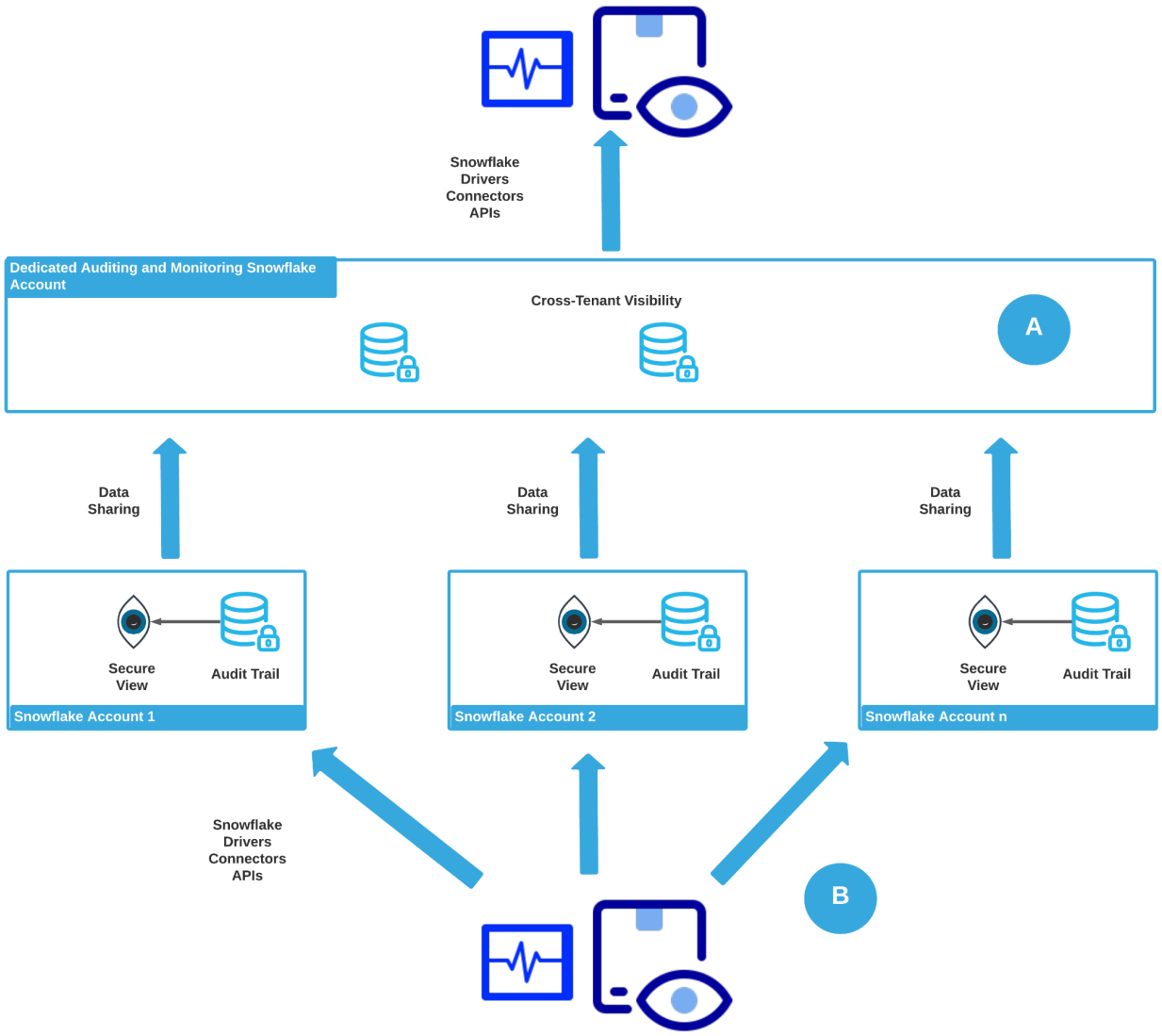


Figure 20. Auditing and monitoring

Table 8 compares options A and B. The application provider can choose to mix the two options to satisfy some situations where the end customer needs access to part of the audit logs directly via their own auditing and monitoring tools, while the application provider uses the dedicated auditing and monitoring of Snowflake to provide auditing and monitoring across all the Snowflake accounts in the scope of the data application(s).

Table 8: Auditing and monitoring options

Considerations	Option A	Option B
Monitoring Tool Connectivity	The auditing and monitoring tools connect to each Snowflake account.	The auditing and monitoring tools connect to the dedicated auditing and monitoring Snowflake account.
Visibility Scope	One Snowflake account.	The dedicated auditing and monitoring Snowflake account provides full visibility across all of the application provider's Snowflake accounts.
Audit Trail Aggregation	The auditing and monitoring tools are responsible for aggregating the audit trails from all the Snowflake accounts in the scope.	The dedicated auditing and monitoring Snowflake account aggregates the audit trails from all other Snowflake accounts via secure views and data sharing and provides a single source of truth.
Data App Design Pattern*	MTT and OPT	APT

***Note:** Option A and B can be used with the data application design patterns; however, option B could make sense with the APT design pattern, where each end customer can monitor their own Snowflake account(s) while the application provider can still use option A to monitor the overall solution.

High availability

Snowflake resilience is automatically distributed across a minimum of three availability zones in each of the supported cloud regions. Application providers can use Snowflake's replication and failover capabilities to achieve cross-region and even cross-cloud high availability for a data application. These capabilities can be combined with the Client Redirect feature available in Snowflake's Business Critical Edition, so the data application can use one global URL (private or public flavors) to connect to Snowflake account(s) in different regions or in different cloud provider platforms. For more details, refer to the Client Redirect documentation.

Provisioning and operational access for the application provider's Snowflake accounts

The application provider can automate aspects of the Snowflake accounts' security controls in addition to other Snowflake capabilities that are required for the data applications.

In the case of MTT and OPT, usually the application provider provisions all the security controls and operates the underlying Snowflake accounts. In case of APT, the application provider can delegate some of the underlying Snowflake accounts to the end customers (tenants). For instance, there are use cases where the application provider is responsible for the following and possibly more:

- Account provisioning
- Account-level security and network controls such as private connectivity, SAML, and stage integrations
- Provisioning users and roles that are needed by the application provider to operate the account
- Auditing and monitoring the Snowflake accounts

At the same time, the application provider can delegate database and schema objects as well as their related users and roles provisioning and operations and security to the end customers using Snowflake access control.

The application provider will need access to the underlying Snowflake accounts, as discussed previously, for provisioning and operating the accounts in addition to integrating Snowflake with the components of the overall data application data pipeline (such as ETL tools, data sources, and so on).

Application providers and their tools can authenticate to Snowflake using any of the following supported authentication methods:

- SAML
- OAuth
- Native usernames and passwords
- Key pairs
- External browser (to support some legacy tools that do not support any form of SSO)

As shown in Figure 21, the method of the authentication and authorization depends on the tools' capability to support one or more of the above methods.

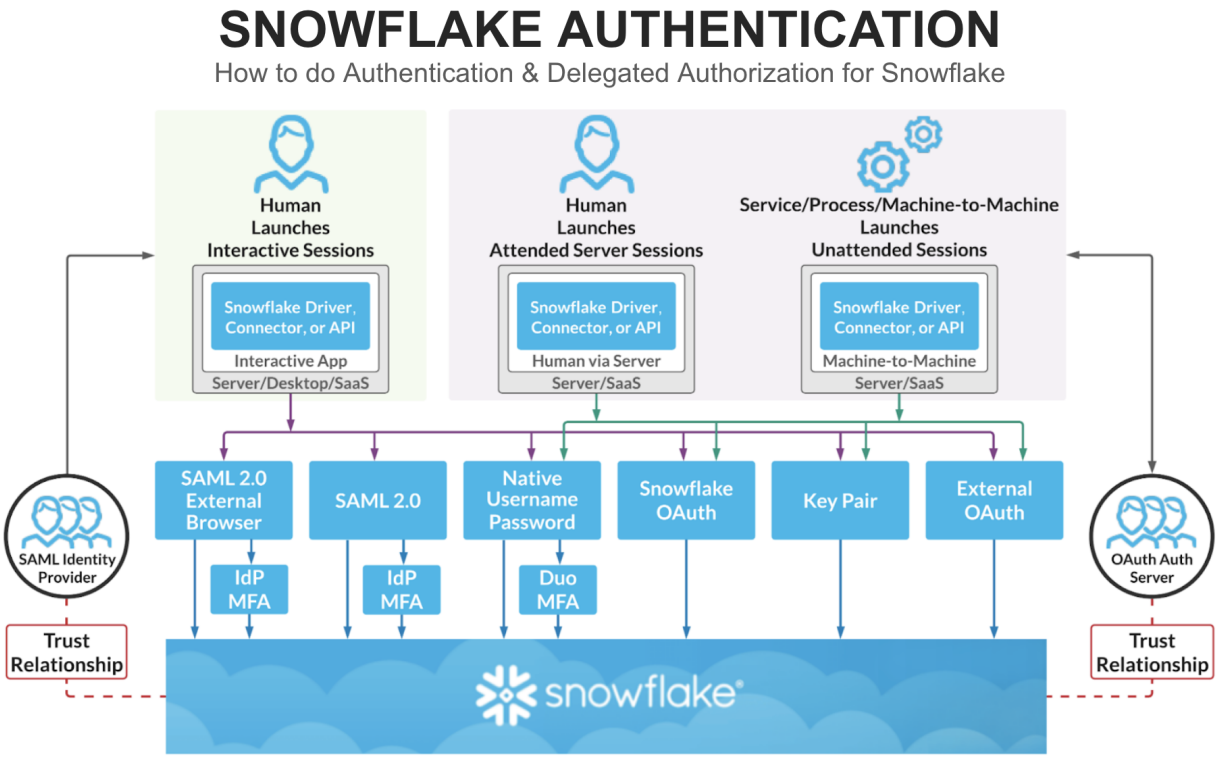


Figure 21. Snowflake authentication and authorization methods

Security for the MTT Design Pattern

The application provider has one (or more, if cross-cloud or cross-region high availability is required) Snowflake accounts configured to service multiple end customers. In this pattern, multiple end customers will use the same Snowflake account and the same data table (see Figure 22).

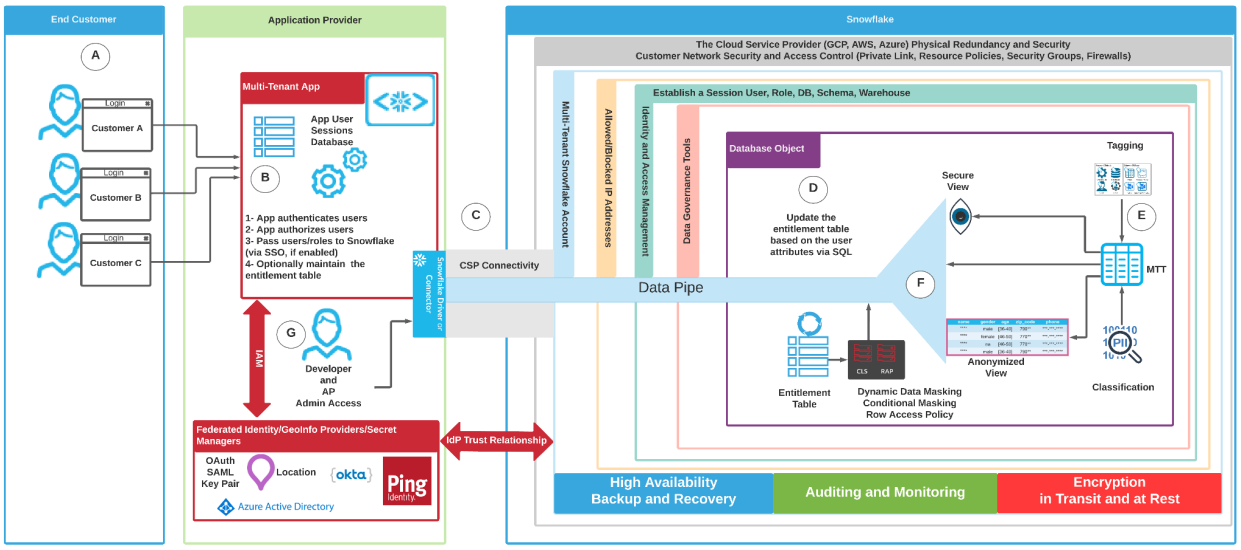


Figure 22. MTT security architecture

Figure 22 shows the MTT design pattern from a security perspective:

- **A:** End customers connect to the multi-tenant application and then get authenticated via the application. The application provider can give the end customers access to the underlying Snowflake account (if that is required). Without exception, the application provider must apply the same policies and security controls discussed in this document to the end customer's user accounts.
- **B:** The data application authenticates the end customer's users via a built-in mechanism or it may use an external IdP to provide an SSO experience. The application creates and maintains the end customer's user sessions.
    - The application can use a service account or it can use SSO to connect to Snowflake.
    - The application optionally can update the entitlement table to be used later in data policies.
- **C:** The application connects to Snowflake via public or private connectivity passing through the network policies and the identity and access management layer.

- **D:** The underlying Snowflake account updates and maintains the entitlement table according to the options discussed in the Maintaining and updating the entitlement table section.
- **E:** The underlying Snowflake account can do the following:
    - Use tagging to tag objects in Snowflake including users, warehouse, tables, and views.
    - Use classification to identify personally identifiable information (PII).
    - As discussed in the Using tags section, the resulting tags can be used to apply additional data policies to protect the end customer's PII if it exists.
    - Optionally, the application provider can use anonymized views on the multi-tenant table to provide analytical reports on the data without exposing sensitive data.
- **F:** At this step, the underlying Snowflake account has all the contextual information to apply data policies on the multi-tenant table directly or on the relevant secure views to isolate data between different end customers.
- **G:** This step shows application provider admin and developer access to the underlying Snowflake accounts. This access is subject to all the security and data policies per the data application functionality and security requirements, as described in the previous steps.

Security for the OPT Design Pattern

The application provider has one (or more, if cross-cloud or cross-region high availability is required) Snowflake accounts configured to service multiple end customers. In this pattern, each end customer will use the dedicated database object in the same Snowflake account (see Figure 23).

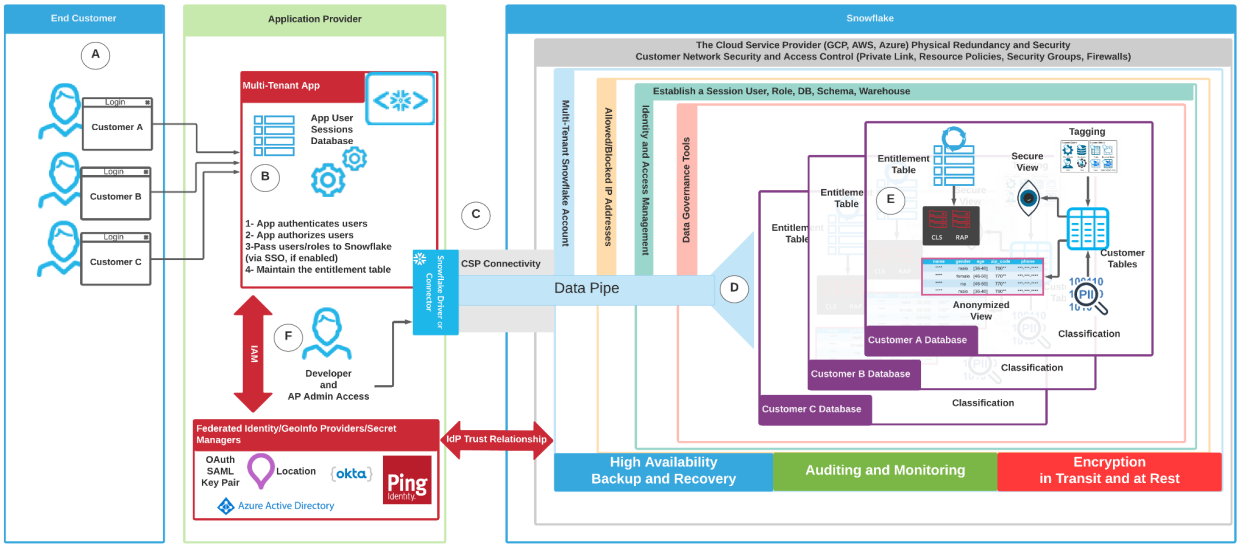


Figure 23. OPT security architecture

Figure 23 shows the OPT design pattern from a security perspective:

- **A:** End customers connect to the multi-tenant application and then get authenticated via the application. The application provider can give the end customers access to the underlying Snowflake account (if that is required). Without exception, the application provider must use access controls such as RBAC and apply the same policies and security controls discussed in this document to the end customer's user accounts.
- **B:** The data application authenticates the end customer's users via a built-in mechanism or it may use an external IdP to provide an SSO experience. The application creates and maintains the end customer's user sessions.
    - The application can use a service account or it can use SSO to connect to Snowflake.
    - The application optionally can update the entitlement table in the relevant end customer's database object to be used later in data policies within the end customer's database object, if needed.

- **C:** The application connects to Snowflake via public or private connectivity passing through the network policies and the identity and access management layer.
- **D:** The underlying Snowflake account uses Snowflake access controls such as RBAC to segregate object access between tenants. Each tenant will have access to its own account objects such as databases, warehouses, users, roles, and so on.
- **E:** Similar to the case with the MTT pattern, within each database, controls can be put in place to define granular data policies.
- **F:** This step shows application provider admin and developer access to the underlying Snowflake accounts. This access is subject to all the security and data policies per the data application functionality and security requirements, as described in the previous steps.

Security for the APT Design Pattern

The application provider provisions one (or more, if cross-cloud or cross-region high availability is required) Snowflake account per tenant (see Figure 24).

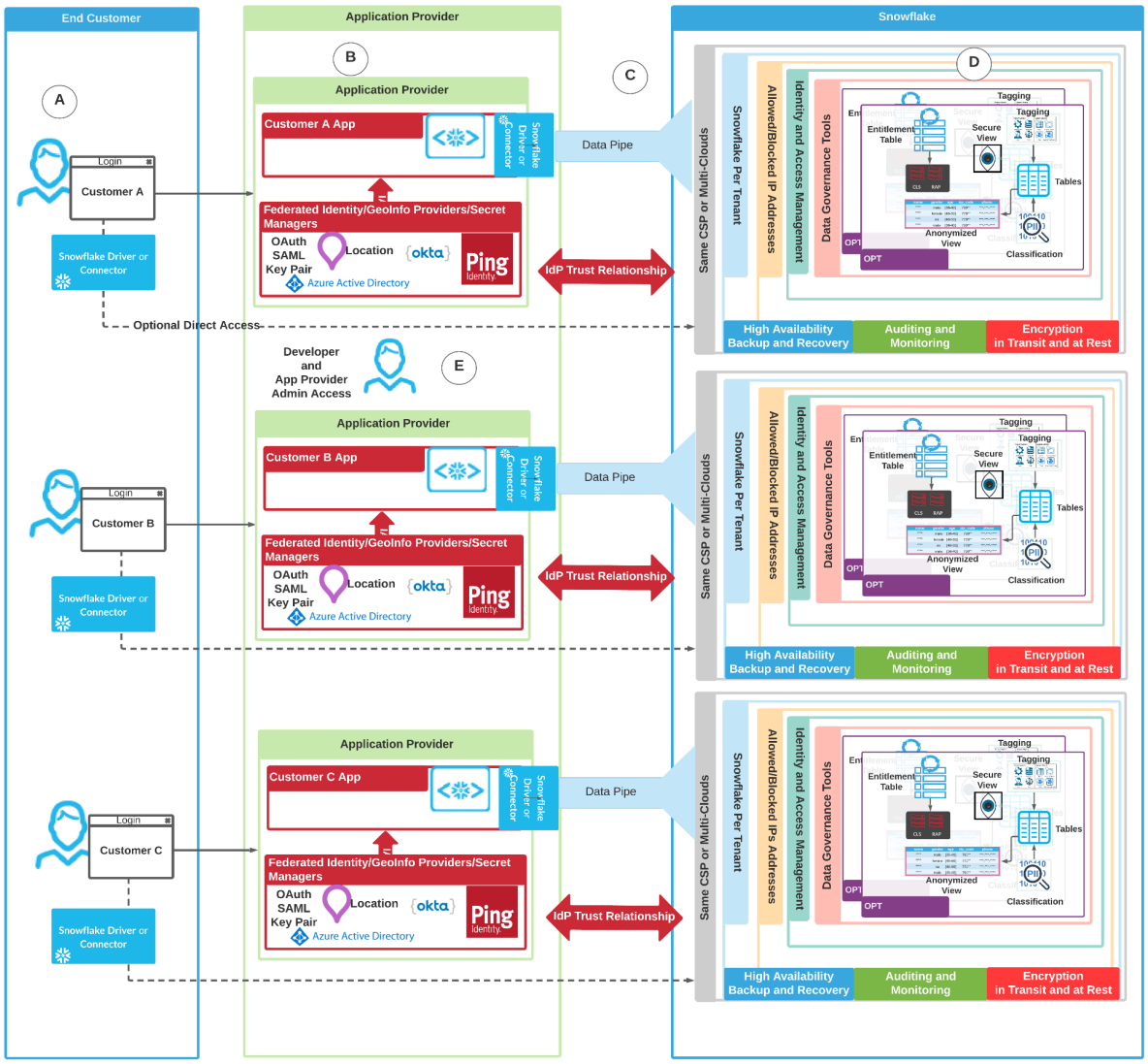


Figure 24. APT security architecture

Figure 24 shows the APT design pattern from a security perspective:

- **A:** End customers connect to the multi-tenant application and then get authenticated via the application. The application provider can give the end customers access to their underlying Snowflake account. The application provider can delegate some of the administrative tasks to the tenant using Snowflake access controls such as

RBAC. For instance, each tenant can bring their own IdP and their SCIM provider to their Snowflake account.

- **B:** The data application authenticates the end customer's users via a built-in mechanism or it may use an external IdP to provide an SSO experience. The application creates and maintains the end customer's user sessions.
    - The application can use a service account or it can use SSO to connect to Snowflake.
    - The application optionally can update the entitlement table in the relevant end customer's Snowflake account(s) to be used later in data policies within the end customer's Snowflake account, if needed.
- **C:** The application connects to Snowflake via public or private connectivity passing through the network policies and the identity and access management layer.
- **D:** Within each tenant Snowflake account, similar controls to what was discussed in the Security for the MTT Design Pattern and Security for the OPT Design Pattern sections can be still used to define granular data policies within the same Snowflake account.
- **E:** This step shows application provider admin and developer access to the underlying Snowflake accounts. This access is subject to all the security and data policies per the data application functionality and security requirements described in the previous steps.

Conclusion

Snowflake provides data applications with different levels of isolation that can be at the account, object, and table levels to satisfy the functionality and security requirements of end customers. Table 9 provides a summary.

Table 9: Data application security summary

	MTT	OPT	APT
Shared Objects	Yes; shared account warehouse, databases, and tables	Yes; shared accounts, but dedicated databases and warehouse	No; dedicated account per tenant
Row-Level Isolation	Yes	Yes; it is possible on top of object-level isolation	Yes; it is possible on top of account and object-level isolation
Object-Level Isolation	No; the objects are shared	Yes, via Snowflake access control	Yes; every tenant has a dedicated Snowflake account (or accounts)
Account-Level Isolation	No; the account (or accounts) is shared	No; the account (or accounts) is shared	Yes; dedicated account per tenant
User Provisioning	Manual or via SCIM per tenant; shared user database	Manual or via SCIM per tenant; shared user database	Manual or via SCIM per tenant; shared user database
Role Provisioning	Manual or via SCIM per tenant; shared role database	Manual or via SCIM per tenant; shared role database	Manual or via SCIM per tenant; shared role database
Connectivity	Public or private	Public or private	Public or private
Network Policy	Refer to the Data application network policies section	Refer to the Data application network policies section	Refer to the Data application network policies section
IAM	One SAML IdP per Snowflake account	One SAML IdP per Snowflake account	One SAML IdP per Snowflake account

	Multiple OAuth IdPs per Snowflake account Multiple SCIM providers per Snowflake account Key pairs Usernames/ password (not recommended)	Multiple OAuth IdPs per Snowflake account Multiple SCIM providers per Snowflake account Key pairs Usernames/ password (not recommended)	Multiple OAuth IdPs Per Snowflake account Multiple SCIM providers per Snowflake account Key pairs Usernames/ password (not recommended)
Bring Your Own Key Per Tenant	No	No	Yes
Row-/Column-Level Encryption Per Tenant	Yes	Yes	Yes
Auditing and Monitoring	Refer to the Auditing and monitoring section	Refer to the Auditing and monitoring section	Refer to the Auditing and monitoring section
High Availability	Refer to the High availability section	Refer to the High availability section	Refer to the High availability section