IaC Snowflake Admin Service User RSA Key Credentials Creation Script

This script greatly enhances the efficiency and security of an enterprise's operations. It aims to simplify the process of creating Snowflake admin service users who use RSA key pair authentication. These admin service users will eventually be responsible for creating Snowflake User or Service Accounts. **Especially since as of November 2025, Snowflake announced in 2024 that it will block single-factor password authentication for user and service accounts!**

So, I am glad you are reading this! 😉 Below is a list of the key benefits of this script:

1. **Automated RSA Key Pair Generation:**
    - The script automates the creation of RSA key pairs, which are essential for authenticating the Snowflake user. By handling this automatically, the script eliminates manual steps, making it easier for developers to integrate and manage Snowflake resources through Terraform or other Snowflake clients.
    - This automation streamlines the authentication process, reducing setup time and potential errors, thereby enabling faster and more reliable deployment of Snowflake services.
2. **Minimal required permissions:**
    - The script grants the smallest set of privileges that the admin service user needs to perform its required actions. This approach is part of the principle of *least privilege*, a security best practice that minimizes the potential for unauthorized access or accidental modifications by limiting permissions to only what is necessary. Below is the list of roles that will be granted to the admin service user:

Role	Description
`ACCOUNTADMIN`	The `ACCOUNTADMIN` role in Snowflake is the highest-level administrative role within a Snowflake account. It has full control over all objects, resources, and configurations within the account. This role is responsible for managing all aspects of the Snowflake environment, including user access, resource allocation, and security settings.
`SECURITYADMIN`	The `SECURITYADMIN` role in Snowflake is a built-in system role designed to manage security-related tasks, primarily concerning user and role management. The `SECURITYADMIN` role has elevated privileges that allow it to control access within a Snowflake account, making it one of the key roles for maintaining the security posture of a Snowflake environment.
`SYSADMIN`	The `SYSADMIN` role in Snowflake is one of the predefined system roles that comes with a broad set of administrative privileges. It is designed to provide comprehensive control over most Snowflake resources, such as databases, schemas, warehouses, and other objects within an account. The `SYSADMIN` role is typically used for database administrators who manage the creation and configuration of Snowflake resources and control access to them.

3. **Secure Storage in AWS Secrets Manager:**
    - User information and RSA key pairs are securely stored in AWS Secrets Manager. This ensures that sensitive data is protected while remaining easily accessible for future use without needing to compromise security.
    - The integration with AWS Secrets Manager supports secure key management practices, safeguarding against unauthorized access and simplifying the retrieval of credentials when needed.
4. **Support for Key-Pair Rotation:**
    - To adhere to best practices in security, the script creates two RSA key pairs for each Snowflake user. This approach supports seamless key rotation, allowing one key to be replaced while the other remains active.
    - The decision to generate only two key pairs aligns with Snowflake's current limitation, which allows associating a maximum of two RSA public keys per user. This ensures compliance with Snowflake's capabilities while maintaining robust security protocols.
5. **Support for all the Snowflake clients:**
    - Click here for a list of supported Snowflake clients.

Table of Contents

1.0 Let's get started!

1. Take care of the cloud and local environment prequisities listed below:

   > You need to have the following cloud accounts:
   >
   > - AWS Account *with SSO configured*
   > - `aws2-wrap` utility
   > - Snowflake Account

   > You need to have the following installed on your local machine:
   >
   > - AWS CLI version 2
   > - Snowflake CLI

2. Clone the repo:

   ```
   git clone https://github.com/j3-signalroom/iac-snowflake-admin_service_user-rsa_key_credentials_creation-script.git
   ```

3. From the root folder of the `iac-snowflake-admin_service_user-rsa_key_credentials_creation-script/` repository that you cloned, run the script in your Terminal to create the Snowflake service user:

   ```
   ./provision-snowflake-admin-credentials.sh <create | delete> --profile=<SSO_PROFILE_NAME> \
                                                                --account_identifier=<ACCOUNT_IDENTIFIER> \
                                                                --snowflake_admin_user=<SNOWFLAKE_ADMIN_USER> \
                                                                --snowflake_password=<SNOWFLAKE_PASSWORD> \
                                                                --snowflake_warehouse=<SNOWFLAKE_WAREHOUSE> \
                                                                --secrets_root_path=<SECRETS_ROOT_PATH> \
                                                                --new_admin_service_user=<NEW_ADMIN_SERVICE_USER>
   ```

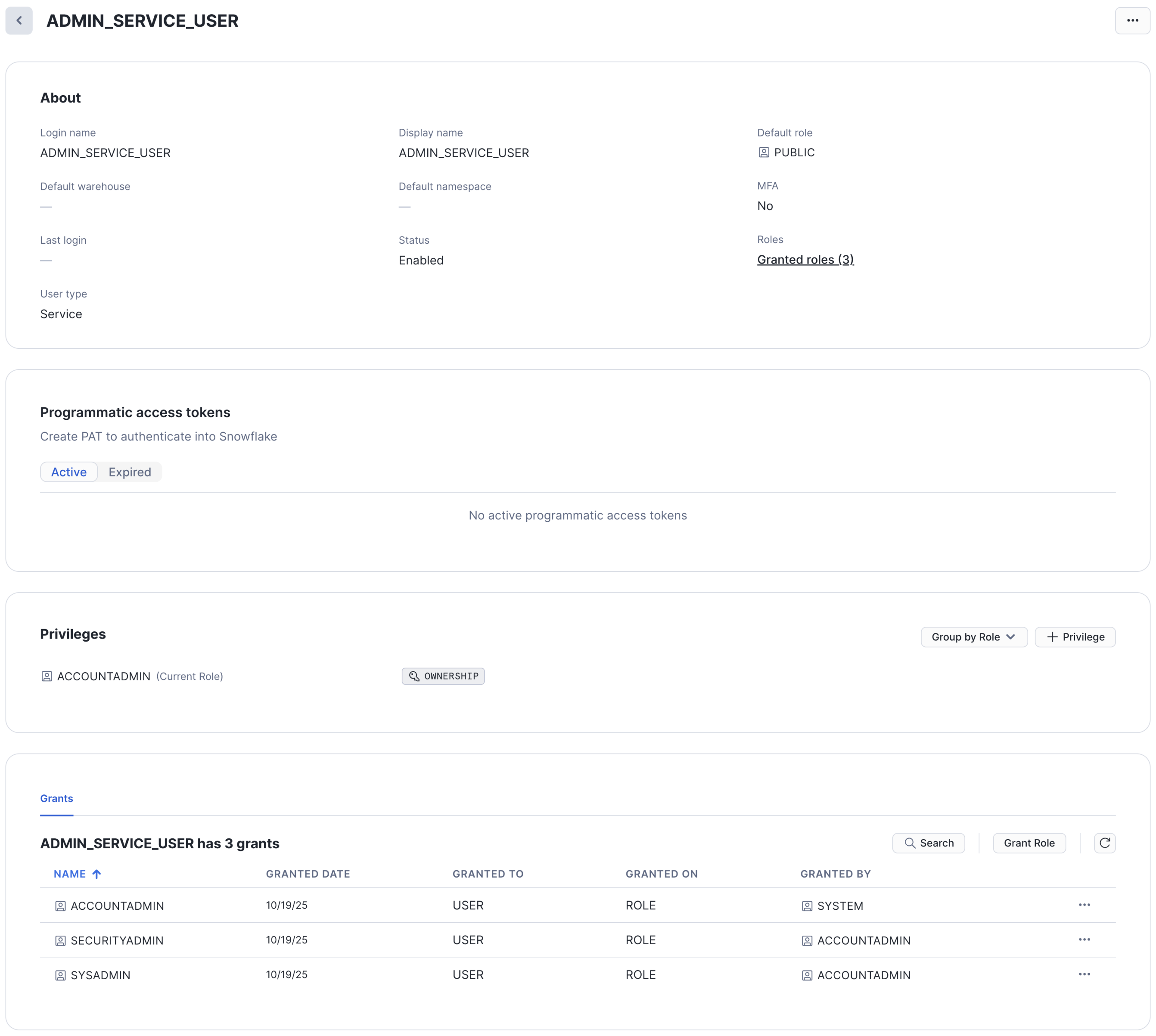
Argument placeholder	Replace with
`<SSO_PROFILE_NAME>`	your AWS SSO profile name for your AWS infrastructue that houses your AWS Secrets Manager.

Argument placeholder	Replace with
`<ACCOUNT_IDENTIFIER>`	your organization's Snowflake account identifier.
`<SNOWFLAKE_ADMIN_USER>`	your Snowflake username that has been granted `ACCOUNTADMIN` privileges.
`<SNOWFLAKE_PASSWORD>`	your Snowflake password of the `<SNOWFLAKE_ADMIN_USER>`.
`<SNOWFLAKE_WAREHOUSE>`	your Snowflake warehouse is the virtual cluster of compute resources that provides CPU, memory, and temporary storage to perform DML (Data Management Language) operations.
`<SECRETS_ROOT_PATH>`	the root path in AWS Secrets Manager where the secrets will be stored.
`<NEW_ADMIN_SERVICE_USER>`	the name of the new Snowflake ACCOUNTADMIN service user to be created or updated.

After the script successfully runs it creates the following in Snowflake and the AWS Secrets Manager for you:

1.1 Snowflake

Below is a picture of an example Snowflake admin service user created with the `ACCOUNTADMIN` role granted by the script:



ADMIN_SERVICE_USER

About

Login name ADMIN_SERVICE_USER	Display name ADMIN_SERVICE_USER	Default role PUBLIC
Default warehouse —	Default namespace —	MFA No
Last login —	Status Enabled	Roles Granted roles (3)
User type Service		

Programmatic access tokens

Create PAT to authenticate into Snowflake

Active Expired

No active programmatic access tokens

Privileges

ACCOUNTADMIN (Current Role) OWNERSHIP

Grants

ADMIN_SERVICE_USER has 3 grants

NAME	GRANTED DATE	GRANTED TO	GRANTED ON	GRANTED BY
ACCOUNTADMIN	10/19/25	USER	ROLE	SYSTEM
SECURITYADMIN	10/19/25	USER	ROLE	ACCOUNTADMIN
SYSADMIN	10/19/25	USER	ROLE	ACCOUNTADMIN

1.2 AWS Secrets Manager Secrets

Here is the list of secrets generated by the Terraform script:

Key	Description
`snowflake_account_identifier`	Your organization's Snowflake account identifier.
`snowflake_organization_name`	The name of your Snowflake organization, which is the part of the account identifier before the hyphen.
`snowflake_account_name`	The name of your Snowflake account, which is the part of the account identifier after the hyphen.
`new_admin_service_user`	The name of the new Snowflake admin user to create and manage future Snowflake resources.
`active_key_number`	The current active RSA public key number.
`snowflake_rsa_public_key_1_pem`	The `new_admin_service_user` Snowflake RSA Public Key 1 PEM, which is encoded in **base64**.

Key	Description
`snowflake_rsa_public_key_2_pem`	The `new_admin_service_user` Snowflake RSA Public Key 2 PEM, which is encoded in **base64**.
`snowflake_rsa_private_key_1_pem`	The `new_admin_service_user` Snowflake RSA Private Key 1 PEM, which is encoded in **base64**.
`snowflake_rsa_private_key_2_pem`	The `new_admin_service_user` Snowflake RSA Private Key 2 PEM, which is encoded in **base64**.

2.0 How the Script Works

This script automates the creation of Snowflake admin users with RSA key pair authentication. It generates two RSA key pairs for each user, ensuring a secure and efficient authentication method. The script also manages the storage of these keys in AWS Secrets Manager, making it easier to handle sensitive information. Below is a sequential diagram of the workflow:

snowflake-rsa-key-authentication-workflow

2.1 Script Sequence Diagram

```
sequenceDiagram
    participant Script as Bash Script
    participant OpenSSL as OpenSSL
    participant FS as File System
    participant Snow as Snowflake CLI
    participant AWS as AWS Secrets Manager

    Note over Script: RSA Key Pair 1 Generation
    Script->>OpenSSL: genrsa 2048
    OpenSSL-->>Script: RSA private key (raw)
    Script->>OpenSSL: pkcs8 -topk8 -inform PEM -nocrypt
    OpenSSL->>FS: private_key_1.p8

    Script->>OpenSSL: rsa -pubout -outform DER
    OpenSSL-->>Script: Public key (DER format)
    Script->>OpenSSL: base64 -A
    OpenSSL->>FS: public_key_1.pub

    Script->>OpenSSL: base64 -A (private key)
    OpenSSL->>FS: private_key_1.b64

    Note over Script: RSA Key Pair 2 Generation
    Script->>OpenSSL: genrsa 2048
    OpenSSL-->>Script: RSA private key (raw)
    Script->>OpenSSL: pkcs8 -topk8 -inform PEM -nocrypt
    OpenSSL->>FS: private_key_2.p8

    Script->>OpenSSL: rsa -pubout -outform DER
    OpenSSL-->>Script: Public key (DER format)
    Script->>OpenSSL: base64 -A
    OpenSSL->>FS: public_key_2.pub

    Script->>OpenSSL: base64 -A (private key)
    OpenSSL->>FS: private_key_2.b64

    Note over Script: Snowflake Service User Creation
    Script->>FS: cat public_key_1.pub
    FS-->>Script: Public key content
    Script->>Snow: CREATE USER TYPE=SERVICE with RSA_PUBLIC_KEY
    Snow-->>Script: Service User created successfully

    Script->>Snow: GRANT ROLE ACCOUNTADMIN
    Snow-->>Script: Role granted

    Script->>Snow: GRANT ROLE SECURITYADMIN
    Snow-->>Script: Role granted

    Script->>Snow: GRANT ROLE SYSADMIN
    Snow-->>Script: Role granted

    Note over Script: AWS Secret Creation
    Script->>FS: cat public_key_1.pub
    FS-->>Script: Public key 1 content
    Script->>FS: cat public_key_2.pub
    FS-->>Script: Public key 2 content
    Script->>FS: cat private_key_1.b64
    FS-->>Script: Private key 1 content
    Script->>FS: cat private_key_2.b64
    FS-->>Script: Private key 2 content

    Script->>AWS: create-secret with JSON payload
    Note right of AWS: Secret contains:
- Account info
- User info
- Both public keys
- Both private keys
- Active key: 1
    AWS-->>Script: Secret created successfully
```

3.0 Resources

- Snowflake Configuring key-pair authentication