IaC Snowflake User RSA key pairs Rotation Terraform module

This Terraform module offers a robust and automated solution for managing the entire lifecycle of RSA key pairs used to authenticate a Snowflake service account. It leverages the Infrastructure-as-Code (IaC) approach with a specialized AWS Lambda function, known as the IaC Snowflake User RSA key pairs and JWT Generator, to automate the generation and rotation of RSA key pairs. The module allows users to define rotation intervals (e.g., every 30 days since the last key generation) to enhance security by regularly renewing cryptographic credentials. Additionally, it integrates seamlessly with AWS Secrets Manager to securely store and manage the generated key pairs, ensuring that the keys remain protected and easily accessible for Snowflake authentication without manual intervention.

Overall, this module is a vital tool for maintaining a secure, efficient, and automated RSA key management process in cloud environments that utilize Snowflake.

> *Snowflake currently enforces a limit of two active RSA key pairs per user account. This restriction is designed to facilitate secure key rotation practices, allowing one key pair to be used actively while the other is kept as a backup or prepared for future use. This dual-key setup ensures continuous access to the Snowflake account while maintaining high security standards by enabling periodic key renewal without downtime.*

To customize this Terraform module for your specific requirements, you can utilize its configurable input and output variables. These variables enable you to adjust key parameters such as the frequency of RSA key pair rotation, the details of the Snowflake user, and other settings relevant to your environment. This approach allows you to adapt the module's behavior to fit different operational needs without directly modifying its underlying code.

By using these variables, you can:

- **Set the Rotation Frequency**: Define how often RSA key pairs should be rotated, aligning with your organization's security policies.
- **Configure Snowflake User Details**: Specify attributes account names, or any other Snowflake-specific settings.
- **Adjust AWS Lambda Settings**: Customize Lambda function parameters, such as memory size and timeout, to optimize performance for your use case.

This flexibility makes the module highly reusable and easier to integrate into various Terraform configurations, enhancing composability and streamlining infrastructure management. It allows for quick adjustments to meet different security requirements or deployment scenarios, reducing the time and effort needed to maintain and adapt infrastructure as code (IaC) practices across multiple projects or environments.

Table of Contents

1.0 Let's get started!

> **Important Notice**
>
> To ensure seamless operation and avoid potential disruptions when using Terraform with time-based rotation, it’s crucial to regularly execute the module within the specified rotation period. Specifically, the execution frequency should match or exceed the configured rotation interval for RSA key pairs. Failing to adhere to this schedule risks the deletion of multiple key pairs in a single execution cycle. Such an occurrence could potentially remove all active RSA keys, thereby disrupting any processes that rely on older keys for accessing Snowflake resources. To maintain uninterrupted access and functionality, it is imperative to keep the module execution timely and consistent with the rotation settings.

These are the steps

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

   > You need to have the following cloud accounts:
   >
   > - AWS Account *with SSO configured*
   > - GitHub Account *with OIDC configured for AWS*
   > - Terraform Cloud Account

2. Clone the repo:

   ```
   git clone https://github.com/j3-signalroom/iac-snowflake-user-rsa_key_pairs_rotation-tf_module.git
   ```

3. Update the cloned Terraform module's main.tf by following these steps:

   a. Locate the `terraform.cloud` block and replace **`signalroom`** with your Terraform Cloud Organization Name.

   b. In the `terraform.cloud.workspaces` block, replace **`snowflake-resources-workspace`** with your Terraform Cloud Organization's Workspaces Name.

4. Deploy your Terraform module to GitHub by following these steps:

   a. **Commit your module:** Ensure all changes to your Terraform module are committed to your local Git repository.

   b. **Push to GitHub:** Push your committed changes to your GitHub repository. This makes the module available for use in other projects.

   c. **Add a Module Block:** In the Terraform configuration where you want to use the module, add a module block. Inside this block, include the following:

   Source: Reference the GitHub repository URL where your module is stored.

Version: Specify the appropriate branch, tag, or commit hash to ensure you're using the correct version of the module.

d. **Pass Input Variables:** Within the same module block, pass the required input variables by defining them as key-value pairs:

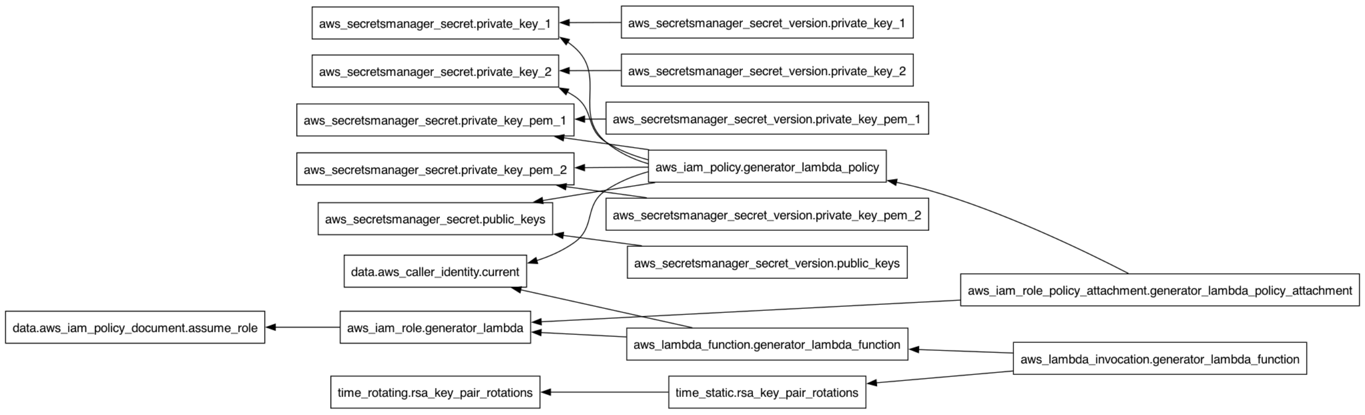
Input Variable	Description
`aws_region`	The AWS Region the Terraform configuration uses.
`snowflake_account`	The Snowflake Account identifer issued to your organization.
`service_account_user`	The Snowflake service account user who is to be assigned the RSA key pairs for its authentication.
`day_count`	***(Optional and defaults to 30 days)*** How many day(s) should the RSA key pairs be rotated for.
`aws_lambda_memory_size`	***(Optional and defaults to 128 MB)*** Lambda allocates CPU power in proportion to the amount of memory configured. Memory is the amount of memory available to your Lambda function at runtime. You can increase or decrease the memory and CPU power allocated to your function using the Memory setting. You can configure memory between 128 MB and 10,240 MB in 1-MB increments. At 1,769 MB, a function has the equivalent of one vCPU (one vCPU-second of credits per second).
`aws_lambda_timeout`	***(Optional and defaults to 900 seconds)*** Lambda runs your code for a set amount of time before timing out. Timeout is the maximum amount of time in seconds that a Lambda function can run. The default value for this setting is 3 seconds, but you can adjust this in increments of 1 second up to a maximum value of 900 seconds (15 minutes).
`aws_log_retention_in_days`	***(Optional and defaults to 7 days)*** Specifies the number of days you want to retain log events in the specified log group. Possible values are: 1, 3, 5, 7, 14, 30, 60, 90, 120, 150, 180, 365, 400, 545, 731, 1096, 1827, 2192, 2557, 2922, 3288, 3653, and 0. If you select 0, the events in the log group are always retained and never expire.

e. **Output Variable(s):** The output variable(s) allow data produced by the module to be exposed to the parent module (the configuration that calls the module), making it possible to use the results of the module's actions elsewhere in the Terraform deployment.

Output Variable	Description
`active_rsa_public_key_number`	Specifies current active RSA public key number.

1.1 Visualizing the Terraform Configuration

Below is the Terraform visualization of the Terraform configuration. It shows the resources and their dependencies, making the infrastructure setup easier to understand.


aws_secretsmanager_secret.private_key_1
aws_secretsmanager_secret_version.private_key_1
aws_secretsmanager_secret.private_key_2
aws_secretsmanager_secret_version.private_key_2
aws_secretsmanager_secret.private_key_pem_1
aws_secretsmanager_secret_version.private_key_pem_1
aws_secretsmanager_secret.private_key_pem_2
aws_iam_policy.generator_lambda_policy
aws_secretsmanager_secret.public_keys
aws_secretsmanager_secret_version.private_key_pem_2
data.aws_caller_identity.current
aws_secretsmanager_secret_version.public_keys
aws_iam_role_policy_attachment.generator_lambda_policy_attachment
data.aws_iam_policy_document.assume_role
aws_iam_role.generator_lambda
aws_lambda_function.generator_lambda_function
aws_lambda_invocation.generator_lambda_function
time_rotating.rsa_key_pair_rotations
time_static.rsa_key_pair_rotations

> **To fully view the image, open it in another tab on your browser to zoom in.**

When you update the Terraform Configuration, to update the Terraform visualization, use the `terraform graph` command with Graphviz to generate a visual representation of the resources and their dependencies. To do this, run the following command:

```
terraform graph | dot -Tpng > .blog/images/terraform-visualization.png
```

2.0 Resources

Snowflake key pair authentication and key pair rotation

Terraform Resource `time_rotating`

Terraform Hidden Gems! Secret Rotation with time_rotating