IaC Snowflake Resources Terraform Configuration

Terraform, an open-source Infrastructure as Code (IaC) tool developed by HashiCorp, uses a declarative approach for managing infrastructure resources. Unlike imperative programming languages like Java, which require explicit, sequential commands to achieve a specific outcome, Terraform enables users to define their desired infrastructure state through configuration files using a YAML-like syntax. This approach abstracts the complexity of manual infrastructure management by allowing users to focus on "what" the final state should be rather than "how" to achieve it.

With Terraform, users can efficiently manage a wide range of Snowflake resources—including Warehouses, Databases, Schemas, Tables, and Roles/Grants—by defining their desired state in configuration files. Terraform maintains a detailed record of the current state of these resources and compares it against the desired state specified by the user. Based on this comparison, Terraform automatically generates a reconciliation plan to bring the existing infrastructure into alignment with the desired configuration. This process involves creating, updating, or deleting resources as needed, enabling consistent, repeatable, and predictable management of infrastructure components.

The configuration leverages the IaC Snowflake Service User RSA key pairs Rotation Terraform module to automate the creation and rotation of RSA key pairs for a Snowflake service account user. This module ensures that each RSA key pair is securely rotated based on a defined schedule, reducing the risk of credential compromise and improving the overall security of the data streaming environment.

To protect sensitive credentials, the configuration securely stores the generated RSA key pairs for both resources in AWS Secrets Manager, ensuring that only authorized users and services have access to these credentials. This secure storage method prevents unauthorized access and minimizes the risk of key exposure.

Table of Contents

1.0 Let's get started!

These are the steps

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

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

    > You need to have the following installed on your local machine:
    >
    > - Terraform CLI version 1.12 or newer

2. Clone the repo:

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

   Then refer to `iac-snowflake-admin_service_user-rsa_key_credentials_creation-script README` to set it up.

3. Clone the repo:

   ```
   git clone https://github.com/j3-signalroom/iac-snowflake-service_user-
   rsa_key_pairs_and_jwt_generator-lambda.git
   ```

   Then refer to `iac-snowflake-service_user-rsa_key_pairs_and_jwt_generator-lambda README` to set it up.

4. Clone the repo:

   ```
   git clone https://github.com/j3-signalroom/iac-snowflake-resources-
   tf.git
   ```

5. 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 **`iac-snowflake-resources-workspace`** with your Terraform Cloud Organization's Workspaces Name.

6. To run repo's Terraform configuration locally, follow these steps:

   a. Navigate to the root folder of the `iac-snowflake-resources-tf/` repository that you cloned.

   b. Open a terminal in this directory.

   c. Execute the following script:

   ```
   ./deploy.sh <create | delete> --profile=<SSO_PROFILE_NAME> \
                                 --snowflake_warehouse=
   <SNOWFLAKE_WAREHOUSE> \
                                 --snowflake-service-user=
   <SNOWFLAKE_SERVICE_USER> \
                                 --secrets-path=<SECRETS_PATH> \
                                 --lambda-function-name=
   <LAMBDA_FUNCTION_NAME> \
   ```

```
                              --admin-service-user-secrets-root-path=
<ADMIN_SERVICE_USER_SECRETS_ROOT_PATH>
```

Argument placeholder	Replace with
`<SSO_PROFILE_NAME>`	your AWS SSO profile name for your AWS infrastructue that host your AWS Secrets Manager.
`<SNOWFLAKE_WAREHOUSE>`	the Snowflake warehouse (or "virtual warehouse") you choose to run the resources in Snowflake.
`<SNOWFLAKE_SERVICE_USER>`	the Snowflake service account user who is to be assigned the RSA key pairs for its authentication.
`<SECRETS_PATH>`	the AWS Secrets Manager path where the RSA key pairs for the Snowflake service account user will be stored.
`<LAMBDA_FUNCTION_NAME>`	the name of the AWS Lambda function that will handle the RSA key pair rotation for the Snowflake service account user.
`<ADMIN_SERVICE_USER_SECRETS_ROOT_PATH>`	the root path for the Snowflake admin service user secrets in AWS Secrets Manager.

7. Or, to run the repository's Terraform configuration from GitHub, follow these steps:

   a. **Deploy the Repository**: Ensure that you have cloned or forked the repository to your GitHub account.

   b. **Set Required Secrets and Variables**: Before running any of the GitHub workflows provided in the repository, you must define at least the `AWS_DEV_ACCOUNT_ID` variable (which should contain your AWS Account ID for your development environment). To do this:

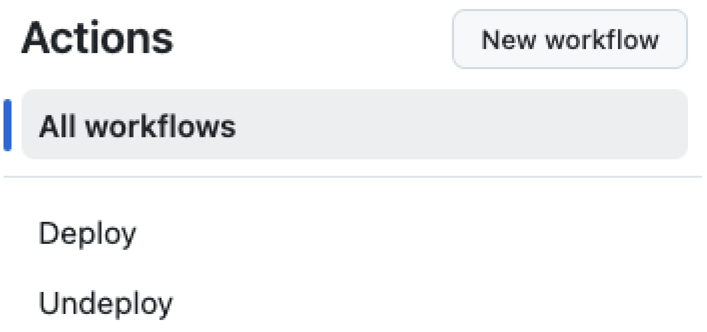
   - Go to the **Settings** of your cloned or forked repository in GitHub.
   - Navigate to **Secrets and Variables** > **Actions**.
   - Add the `AWS_DEV_ACCOUNT_ID` and any other required variables or secrets.

   c. **Navigate to the Actions Page**:

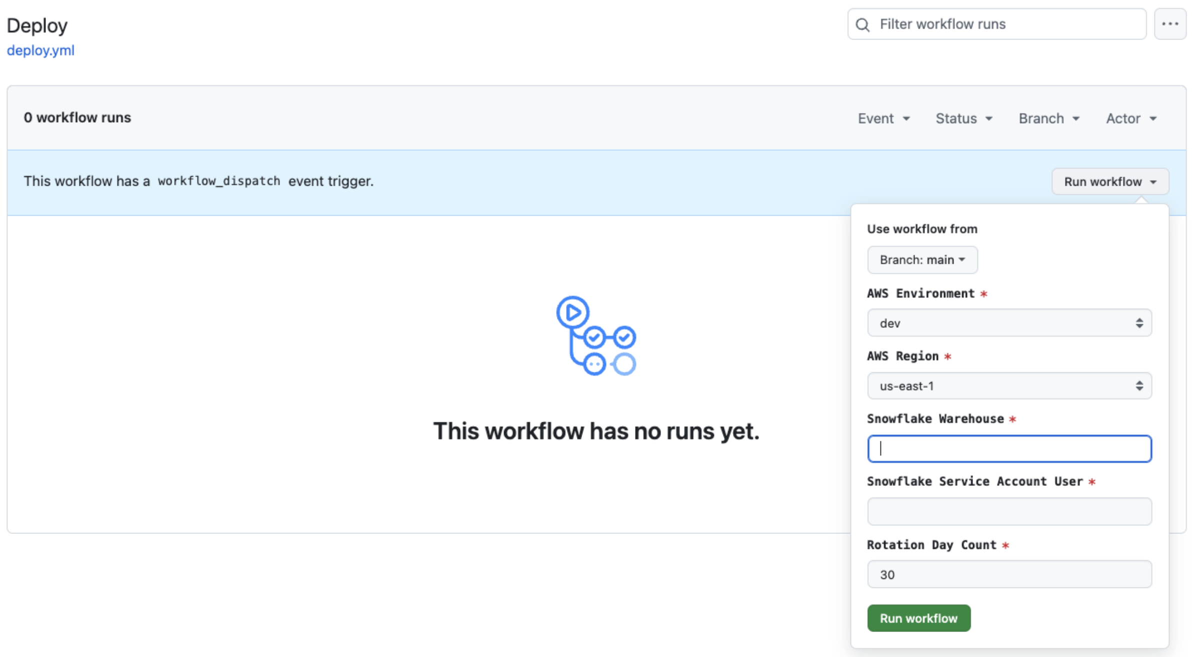
   - From the cloned or forked repository on GitHub, click on the **Actions** tab.

   d. **Select and Run the Deploy Workflow**:

   - Find the **Deploy workflow** link on the left side of the Actions page and click on it.



- On the **Deploy workflow** page, click the **Run workflow** button.
- A workflow dialog box will appear. Fill in the necessary details and click **Run workflow** to initiate the Terraform deployment process.



This workflow has a `workflow_dispatch` event trigger.

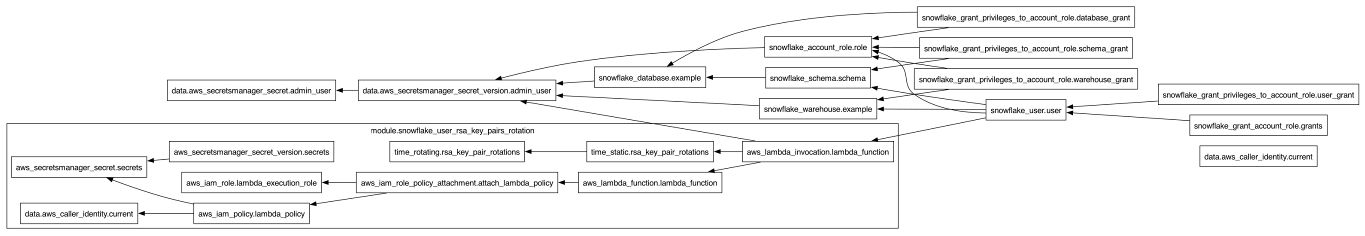
Use workflow from
Branch: main

AWS Environment * : dev
AWS Region * : us-east-1
Snowflake Warehouse * :
Snowflake Service Account User * :
Rotation Day Count * : 30

By following these steps, you will run the Terraform configuration directly from GitHub, leveraging GitHub Actions for automation and deployment.

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.


snowflake_grant_privileges_to_account_role.database_grant
snowflake_account_role.role
snowflake_grant_privileges_to_account_role.schema_grant
snowflake_database.example
snowflake_schema.schema
snowflake_grant_privileges_to_account_role.warehouse_grant
data.aws_secretsmanager_secret.admin_user
data.aws_secretsmanager_secret_version.admin_user
snowflake_grant_privileges_to_account_role.user_grant
snowflake_warehouse.example
snowflake_user.user
snowflake_grant_account_role.grants
data.aws_caller_identity.current
module.snowflake_user_rsa_key_pairs_rotation
aws_secretsmanager_secret_version.secrets
time_rotating.rsa_key_pair_rotations
time_static.rsa_key_pair_rotations
aws_lambda_invocation.lambda_function
aws_secretsmanager_secret.secrets
aws_iam_role.lambda_execution_role
aws_iam_role_policy_attachment.attach_lambda_policy
aws_lambda_function.lambda_function
data.aws_caller_identity.current
aws_iam_policy.lambda_policy

> **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 Terraform Provider
- Terraforming Snowflake Quickstart