IaC Snowflake User RSA Key Pairs and JWT Generator

This AWS Lambda function, developed in Python, automates the creation of two RSA key pairs and a JWT, which are essential for enabling secure, public-key authentication for a Snowflake user or service account. (The reason why only two RSA key pairs are created is that Snowflake currently limits each user to a maximum of two.) Having at least two RSA key pairs helps facilitate key rotation to maintain security and compliance. After generating the RSA key pairs, the function securely stores them in AWS Secrets Manager, using encryption and detailed access controls to protect the keys from unauthorized access. This process not only allows for seamless retrieval and management of the RSA key pairs for future authentication by the Snowflake service account user but also ensures that the keys are handled according to best practices for cloud security and data protection.

> **Important Note**: *If you are reading this before 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! 😉

Table of Contents

1.0 Let's get started!

1. **Prequisities**: 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

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

2. **Get the repo**: Clone the repo:

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

3. **Navigate to the Root Directory**: Open your Terminal and navigate to the root folder of the `iac-snowflake-service_user-rsa_key_pairs_and_jwt_generator-lambda/` repository that you have cloned. You can do this by executing:

```
cd path/to/iac-snowflake-service_user-rsa_key_pairs_and_jwt_generator-lambda/
```

   Replace `path/to/` with the actual path where your repository is located.

4. **Run the Script to Create or Delete the ECR Repository**: Execute the `deploy.sh` script to create an AWS Elastic Container Registry (ECR) repository, build the AWS Lambda Docker container, and publish it to the newly created ECR repository. This will make the container image available for future deployments.

   Use the following command format:

```
./deploy.sh <create | delete> --profile=<SSO_PROFILE_NAME>
```

5. **Replace Argument Placeholders**:

   - `<create | delete>`: Specify either `create` to create the ECR repository or `delete` to remove it.
   - `<SSO_PROFILE_NAME>`: Replace this with your AWS Single Sign-On (SSO) profile name, which identifies your hosted AWS infrastructure.

   For example, to create the ECR repository, use the following command:

```
./deploy.sh create --profile=my-aws-sso-profile
```

   Replace `my-aws-sso-profile` with your actual AWS SSO profile name.

6. **Or, deploy via GitHub**: To run 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:

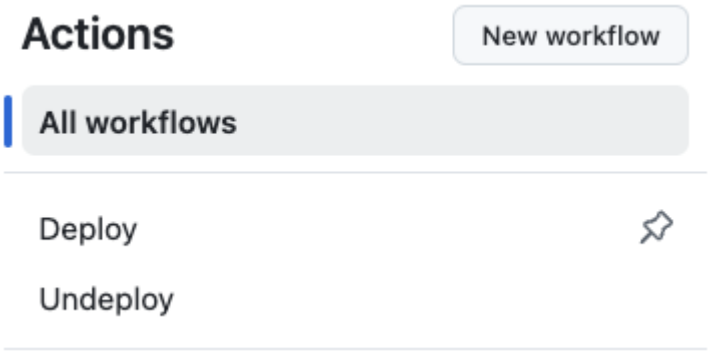
   - Go to the Settings of your cloned or forked repository in GitHub.
   - Navigate to **Secrets and Variables > Actions**.
   - Add the `AWS_DEV_ACCOUNT_ID` and any other required variables or secrets.

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

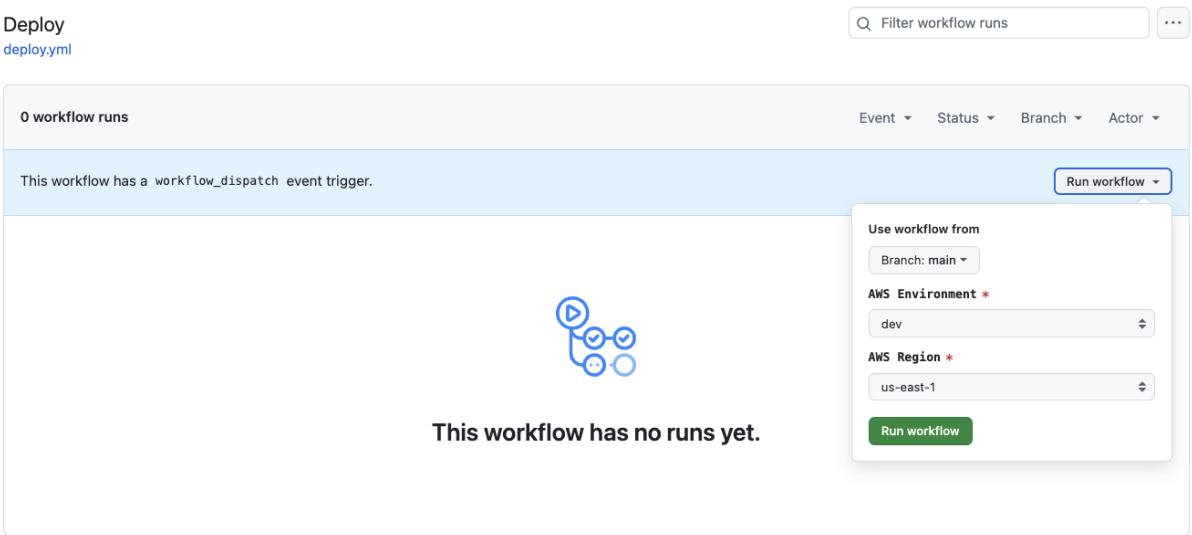
   - From the cloned or forked repository on GitHub, click on the **Actions tab**.

   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 building and publishing the Lambda docker container to ECR.



1.1 Deployment Summary

By following the steps above, you will effectively establish the infrastructure needed to build and deploy the Lambda function container for secure RSA key pair generation in Snowflake. The process involves creating an AWS Elastic Container Registry (ECR) repository, building the Lambda Docker container, and publishing it to the ECR repository. This setup guarantees that the RSA key pairs are securely generated and stored, enabling public-key authentication for your Snowflake service account user.

2.0 Testing

To test the generation of RSA key pairs, you can use the provided test script located in the `tests/` directory. The script will help you verify that the key pairs are generated correctly, meet the required specifications, and can be successfully stored in AWS Secrets Manager. To use the test script, create a `.env` file in the root directory of the project. Then, configure it with the appropriate AWS SSO Profile Name and settings, as follows:

```
SNOWFLAKE_ACCOUNT_IDENTIFIER=<SNOWFLAKE_ACCOUNT_IDENTIFIER>
SNOWFLAKE_ADMIN_USER=<SNOWFLAKE_ADMIN_USER>
SECRETS_PATH=<SECRETS_PATH>
SSO_PROFILE_NAME=<SSO_PROFILE_NAME>
```

Environment Variable	Description
SNOWFLAKE_ACCOUNT_IDENTIFIER	The Snowflake account identifier.
SNOWFLAKE_ADMIN_USER	The Snowflake user name.
SECRETS_PATH	The path to the AWS Secrets Manager secrets.
SSO_PROFILE_NAME	Your AWS SSO profile name.

You can execute the test script with this command:

```
pytest -s tests/test_app.py
```

3.0 Resources

- RSA API