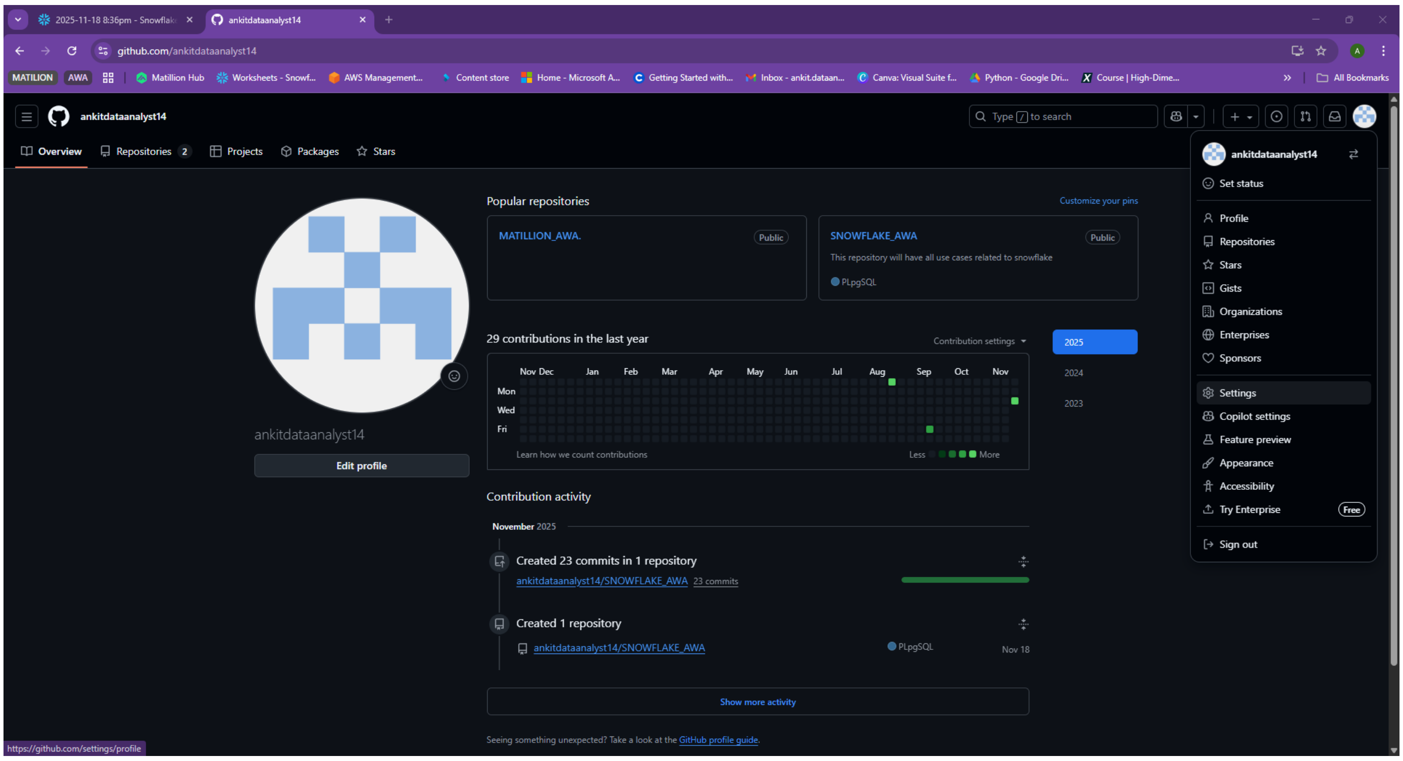
GitHub → Snowflake Integration Guide (Final Version)

This guide shows how to generate a GitHub Personal Access Token (classic) and integrate your GitHub repository with Snowflake. Each step is aligned with the correct screenshot, with detailed notes below each screenshot. The full Snowflake SQL script for GitHub integration is included at the end.

Step 1 — Open GitHub Profile Settings

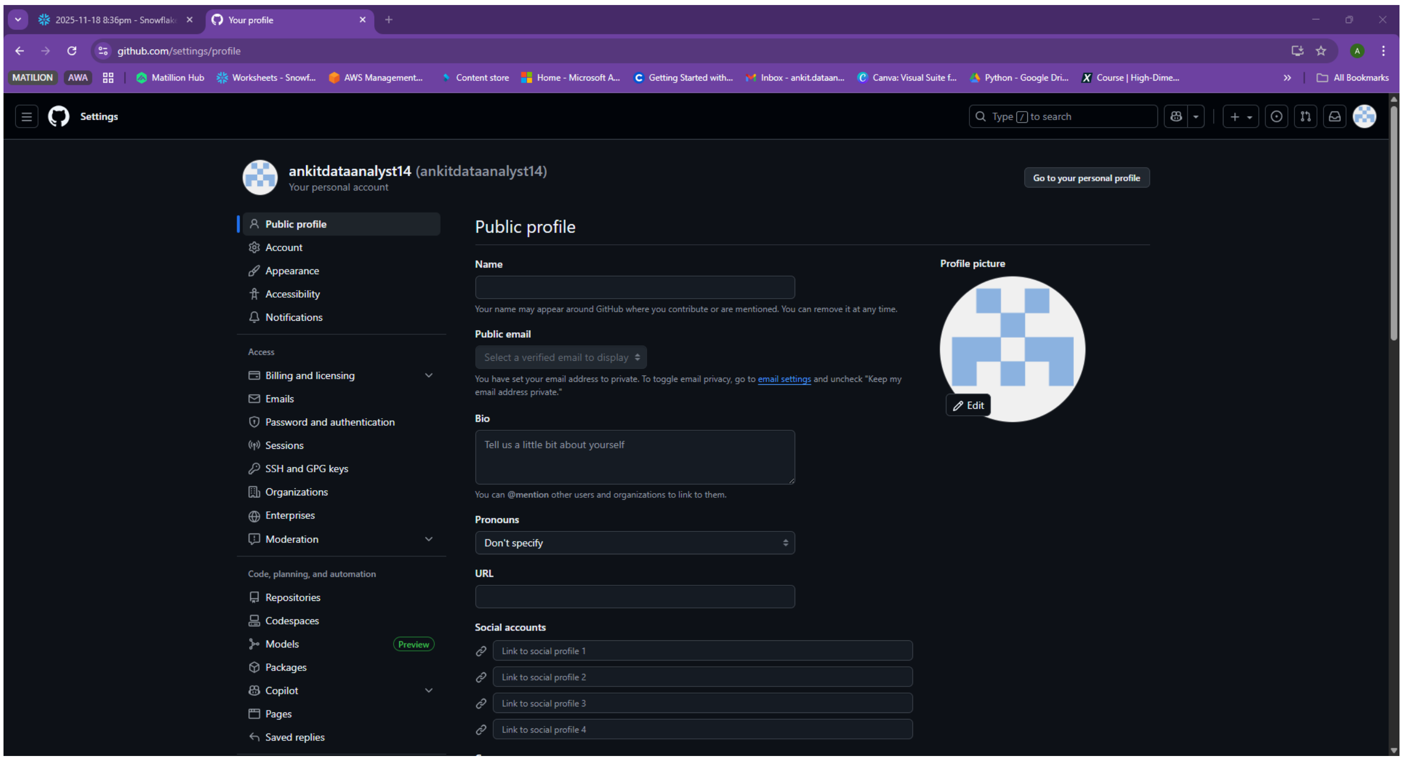
Log into GitHub and click your profile icon in the top-right corner, then choose 'Settings' from the dropdown.



On this page, you begin navigating toward developer settings.

Step 2 — Public Profile Page Opens

This page displays your public profile settings. No action required here.



github.com/settings/profile

Settings

ankitdataanalyst14 (ankitdataanalyst14)
Your personal account

Go to your personal profile

Public profile
Account
Appearance
Accessibility
Notifications

Access
Billing and licensing
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation

Code, planning, and automation
Repositories
Codespaces
Models Preview
Packages
Copilot
Pages
Saved replies

Public profile

Name

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email

Select a verified email to display

You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

Bio

Tell us a little bit about yourself

You can @mention other users and organizations to link to them.

Pronouns

Don't specify

URL

Social accounts

Link to social profile 1
Link to social profile 2
Link to social profile 3
Link to social profile 4

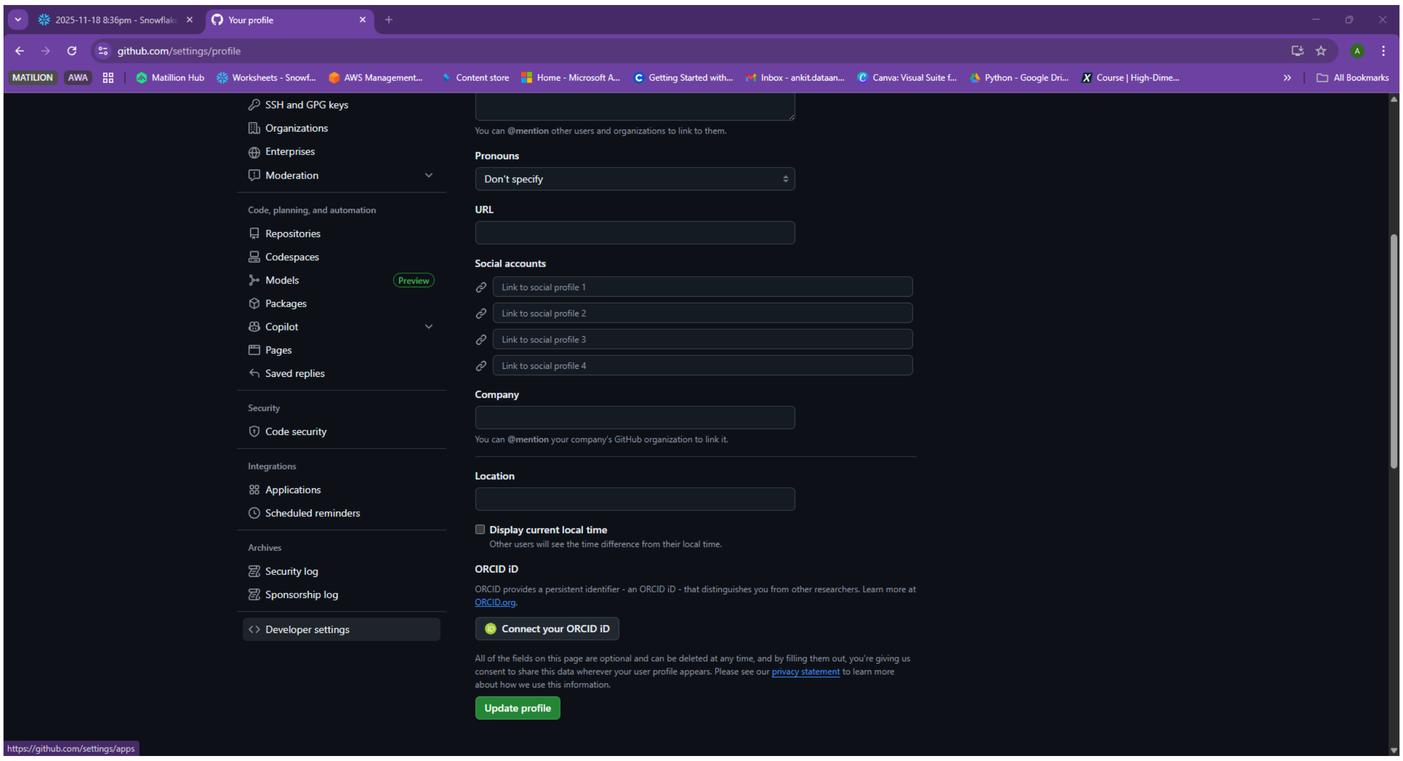
Profile picture

Edit

Scroll downward to reach the Developer Settings section.

Step 3 — Scroll Public Profile Settings

Continue scrolling the sidebar to reach the bottom.



You can @mention other users and organizations to link to them.

Pronouns

Don't specify

URL

Social accounts

Link to social profile 1

Link to social profile 2

Link to social profile 3

Link to social profile 4

Company

You can @mention your company's GitHub organization to link it.

Location

Display current local time

Other users will see the time difference from their local time.

ORCID iD

ORCID provides a persistent identifier - an ORCID iD - that distinguishes you from other researchers. Learn more at ORCID.org.

Connect your ORCID iD

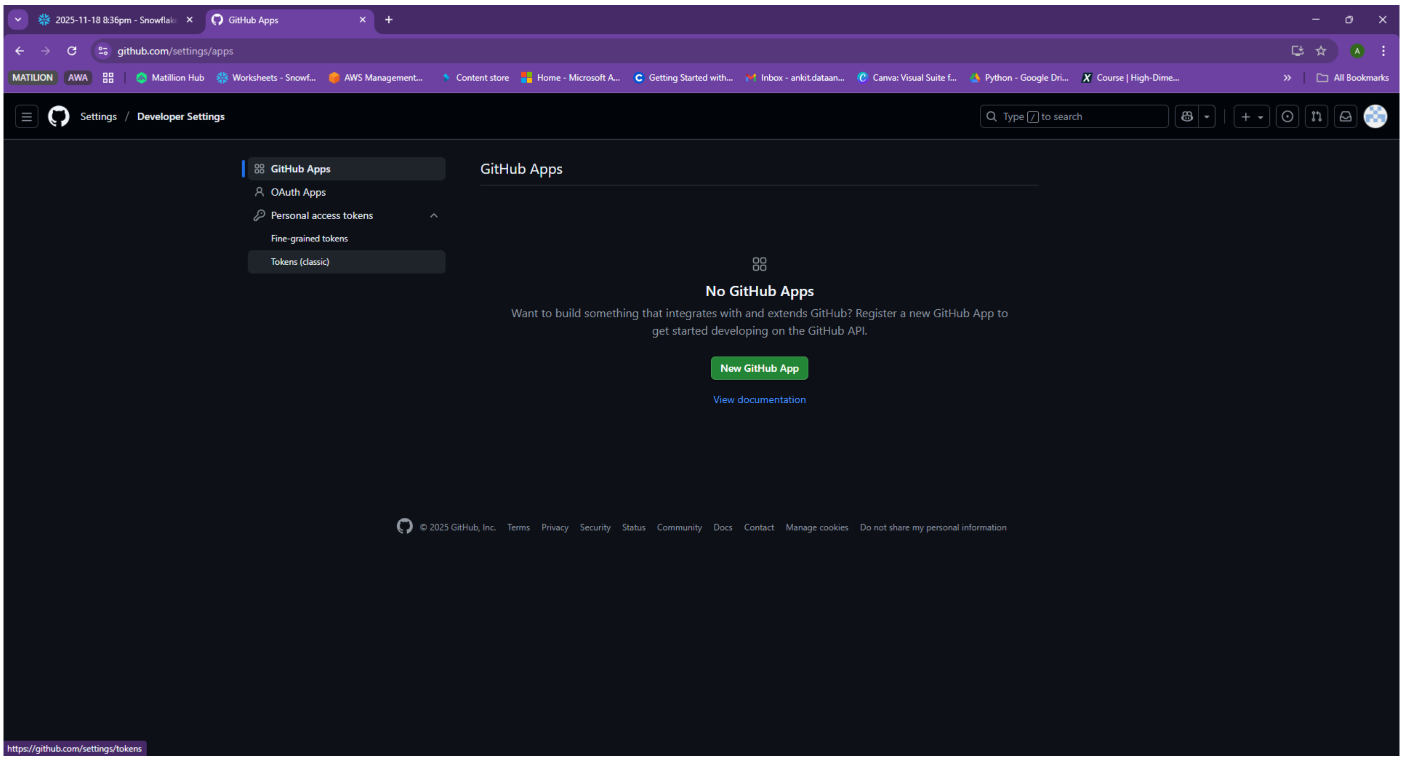
All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.

Update profile

Developer Settings is located at the bottom of the left menu.

Step 4 — Navigate to Developer Settings

Click 'Developer settings' located in the sidebar.


2025-11-18 8:36pm - Snowflake
GitHub Apps
github.com/settings/apps
Settings / Developer Settings
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)

GitHub Apps

No GitHub Apps

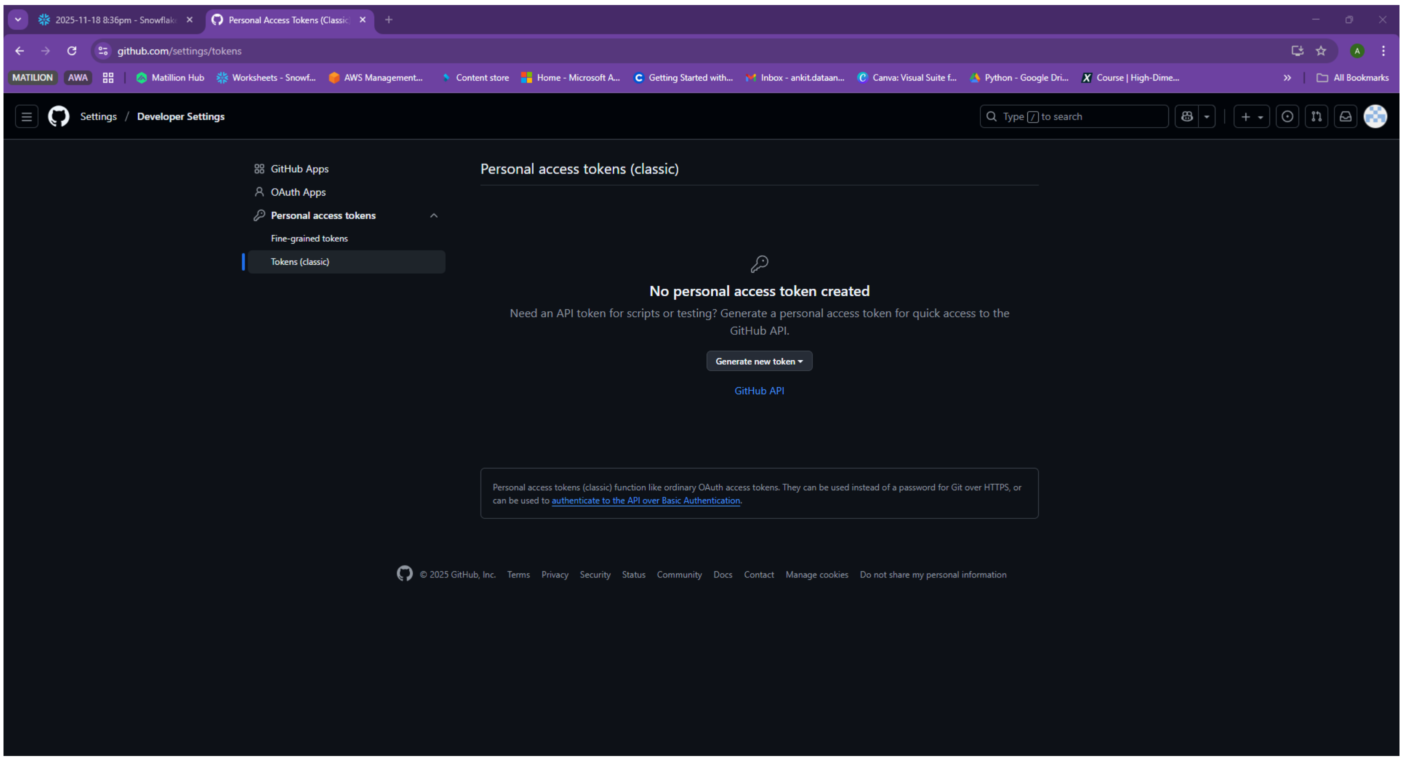
Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started developing on the GitHub API.

New GitHub App
View documentation
© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information
https://github.com/settings/tokens

This is the required section to access Personal Access Tokens.

Step 5 — Open Personal Access Tokens → Tokens (classic)

Click ‘Personal access tokens’, then select ‘Tokens (classic)’.



Personal access tokens (classic)

No personal access token created

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

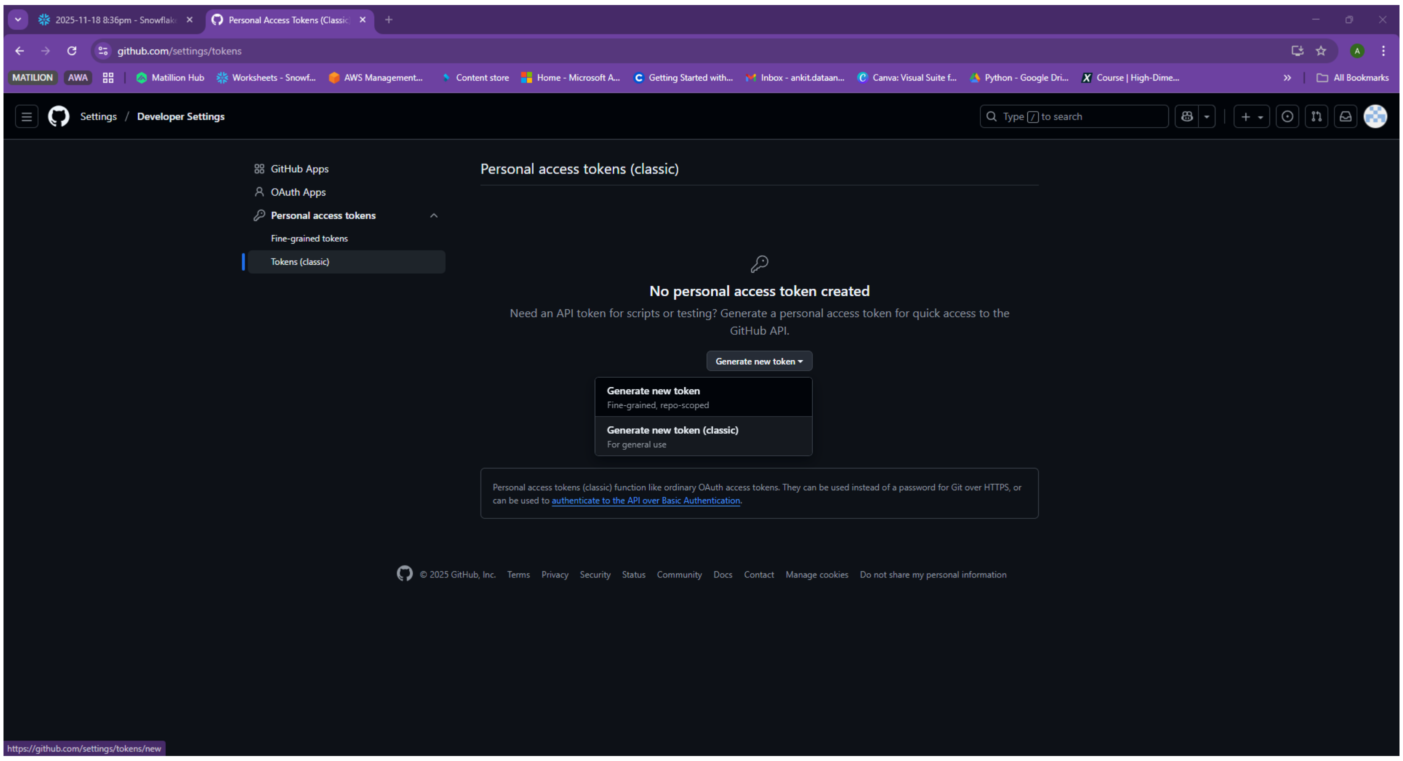
Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

You will see a screen showing zero tokens created (if first time).

Step 6 — Generate a new token (classic)

Click 'Generate new token' and choose 'Generate new token (classic)'.



Personal access tokens (classic)

No personal access token created

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

Generate new token — Fine-grained, repo-scoped

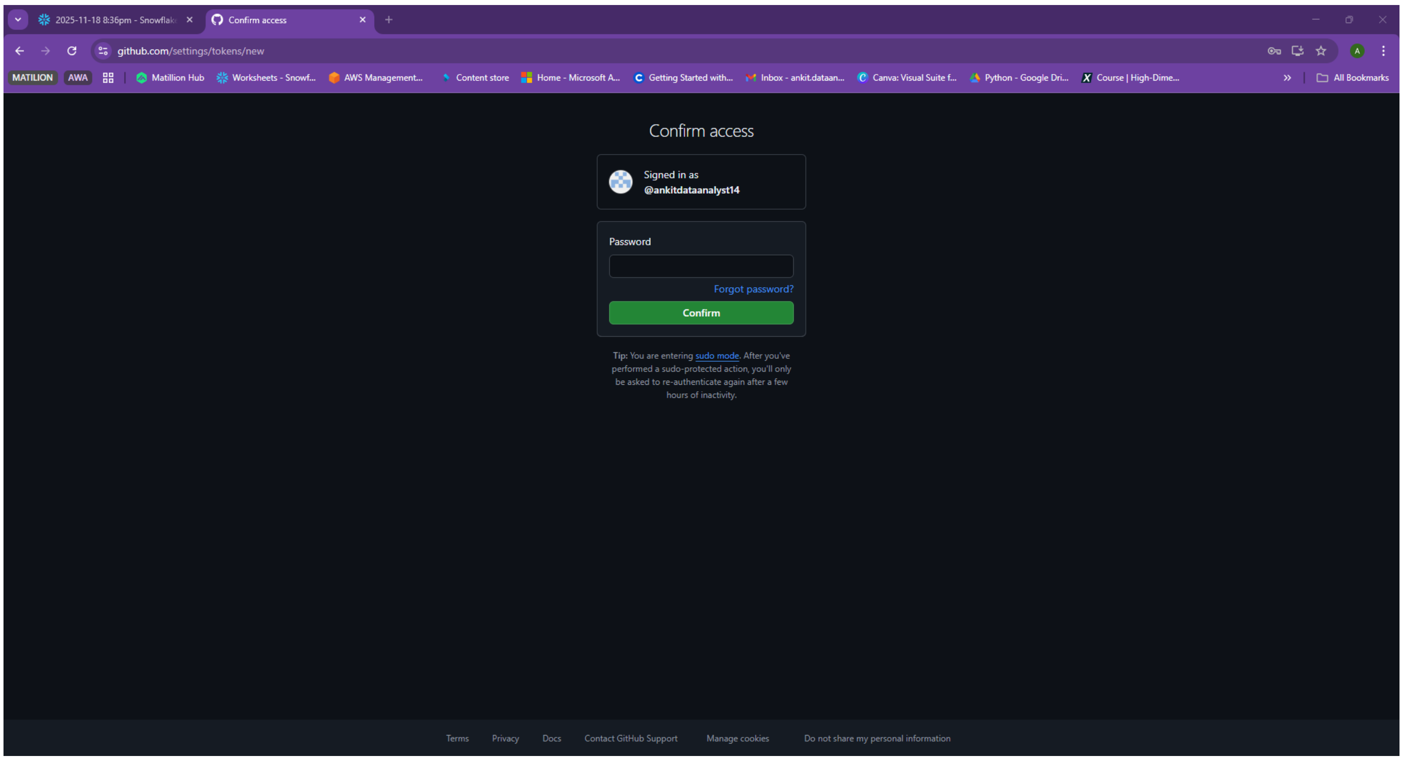
Generate new token (classic) — For general use

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

This begins the PAT creation workflow.

Step 7 — Re-authenticate (Password Confirmation)

GitHub requires password re-entry for security.



Confirm access

Signed in as
@ankitdataanalyst14

Password

Forgot password?

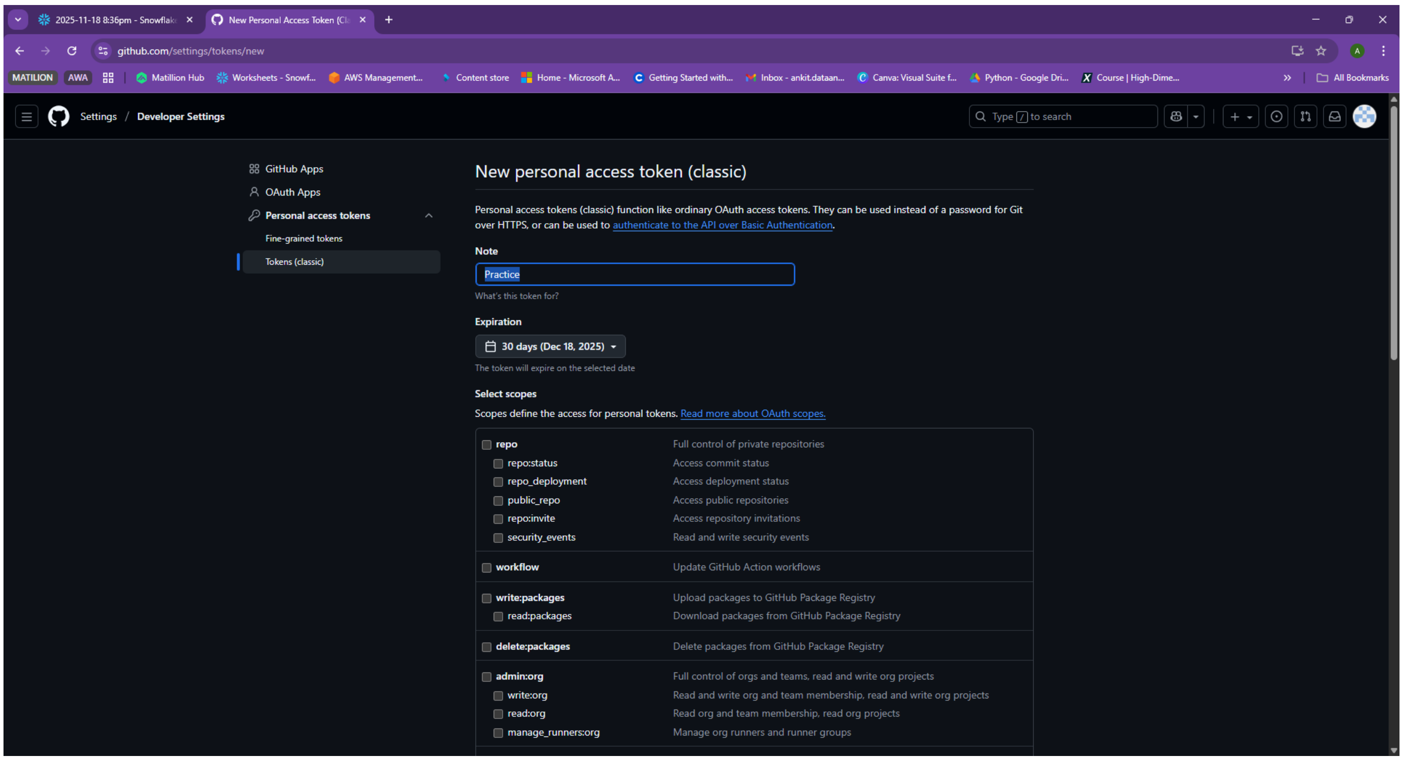
Confirm

Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.

Enter your password to continue.

Step 8 — Provide token name and expiration

Enter a token name (example: Practice Token). Choose expiration duration.



New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

Practice

What's this token for?

Expiration

30 days (Dec 18, 2025)

The token will expire on the selected date

Select scopes

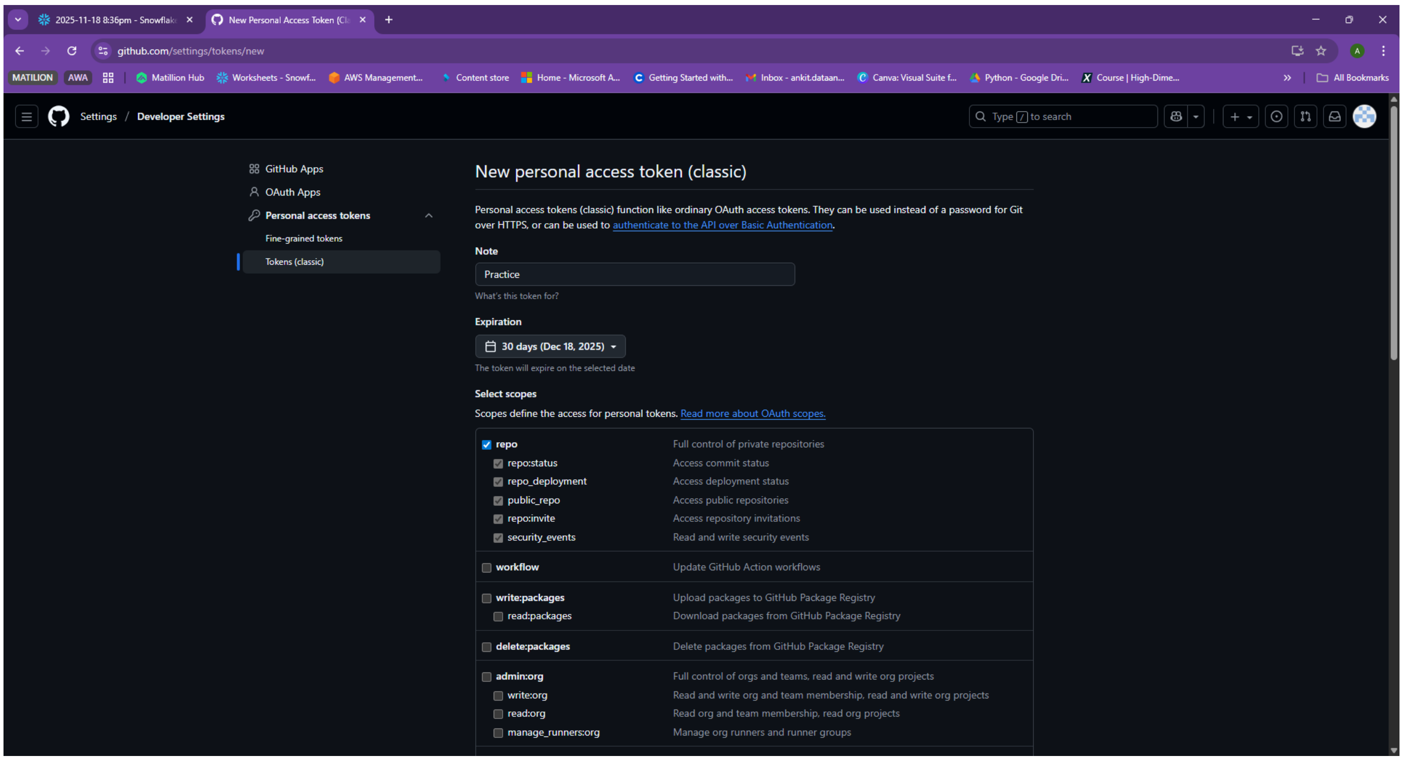
Scopes define the access for personal tokens. Read more about OAuth scopes.

☐ **repo**	Full control of private repositories
☐ repo:status	Access commit status
☐ repo_deployment	Access deployment status
☐ public_repo	Access public repositories
☐ repo:invite	Access repository invitations
☐ security_events	Read and write security events
☐ **workflow**	Update GitHub Action workflows
☐ **write:packages**	Upload packages to GitHub Package Registry
☐ read:packages	Download packages from GitHub Package Registry
☐ **delete:packages**	Delete packages from GitHub Package Registry
☐ **admin:org**	Full control of orgs and teams, read and write org projects
☐ write:org	Read and write org and team membership, read and write org projects
☐ read:org	Read org and team membership, read org projects
☐ manage_runners:org	Manage org runners and runner groups

Scroll down to configure scopes.

Step 9 — Select required scopes

Check 'repo' and related repository permissions. These authorize Snowflake to access your repo.



New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

Practice

What's this token for?

Expiration

30 days (Dec 18, 2025)

The token will expire on the selected date

Select scopes

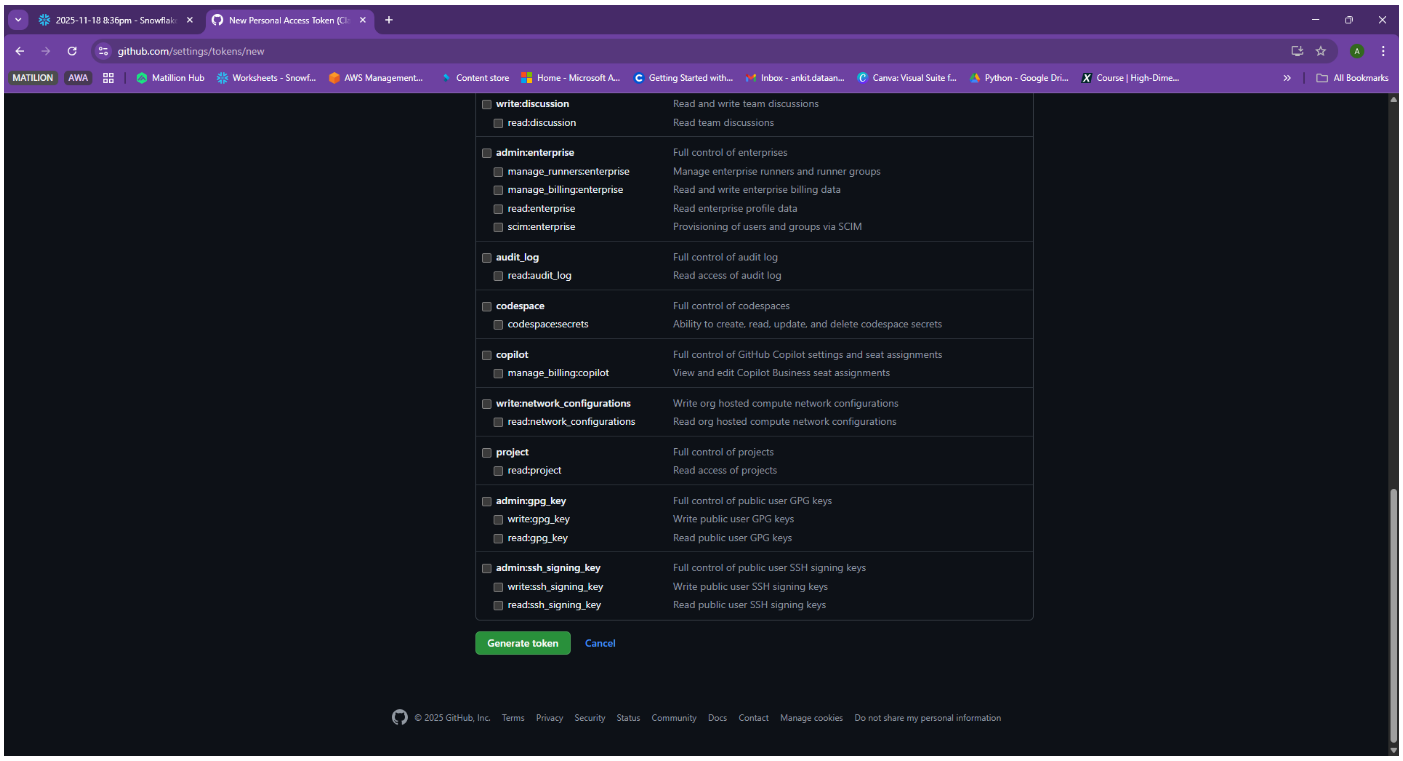
Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ **repo**	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events
☐ **workflow**	Update GitHub Action workflows
☐ **write:packages**	Upload packages to GitHub Package Registry
☐ read:packages	Download packages from GitHub Package Registry
☐ **delete:packages**	Delete packages from GitHub Package Registry
☐ **admin:org**	Full control of orgs and teams, read and write org projects
☐ write:org	Read and write org and team membership, read and write org projects
☐ read:org	Read org and team membership, read org projects
☐ manage_runners:org	Manage org runners and runner groups

Avoid enabling unnecessary scopes beyond repo permissions.

Step 10 — Click Generate Token

Scroll down and click the green 'Generate token' button.



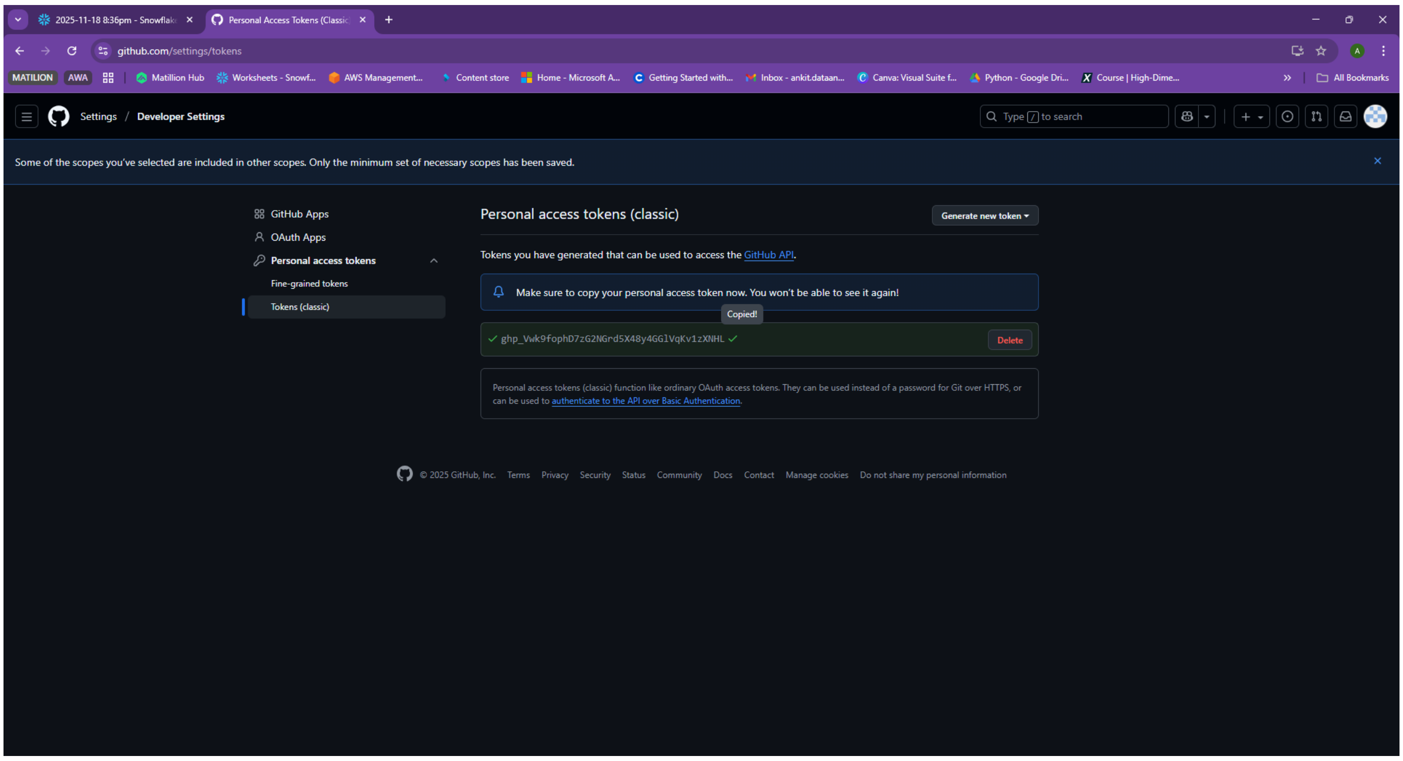
write:discussion	Read and write team discussions
read:discussion	Read team discussions
admin:enterprise	Full control of enterprises
manage_runners:enterprise	Manage enterprise runners and runner groups
manage_billing:enterprise	Read and write enterprise billing data
read:enterprise	Read enterprise profile data
scim:enterprise	Provisioning of users and groups via SCIM
audit_log	Full control of audit log
read:audit_log	Read access of audit log
codespace	Full control of codespaces
codespace:secrets	Ability to create, read, update, and delete codespace secrets
copilot	Full control of GitHub Copilot settings and seat assignments
manage_billing:copilot	View and edit Copilot Business seat assignments
write:network_configurations	Write org hosted compute network configurations
read:network_configurations	Read org hosted compute network configurations
project	Full control of projects
read:project	Read access of projects
admin:gpg_key	Full control of public user GPG keys
write:gpg_key	Write public user GPG keys
read:gpg_key	Read public user GPG keys
admin:ssh_signing_key	Full control of public user SSH signing keys
write:ssh_signing_key	Write public user SSH signing keys
read:ssh_signing_key	Read public user SSH signing keys

Generate token
Cancel
© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information
github.com/settings/tokens/new

The token will appear only once—copy and store safely.

Step 11 — Copy and Secure Your Token

Copy your generated token immediately; GitHub will not show it again.



Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

Copied!

```
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
```

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

github.com/settings/tokens
Settings / Developer Settings
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
Generate new token
Delete
© 2025 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

Use Snowflake Secret Manager to store this token securely.

Snowflake SQL Integration Script

```
-- FULL SQL SCRIPT (user-provided)

-- ####################################################################
-- 00 - PREP: NOTES (do not run)
-- ####################################################################
-- 1) Create a Snowflake SECRET to store your GitHub PAT securely.
-- 2) Never paste PATs in SQL – use Secrets.
-- 3) Repo origin: https://github.com/ankitdataanalyst14/SNOWFLAKE_AWA.git

CREATE OR REPLACE SCHEMA GITHUB;

-- SECRET creation is recommended via UI for security.
-- Example (admin):
-- CREATE OR REPLACE SECRET GIT_SECRET TYPE=PASSWORD USERNAME='<USERNAME>' PASSWORD=

CREATE OR REPLACE API INTEGRATION GIT_INT
  API_PROVIDER=GIT_HTTPS_API
  API_ALLOWED_PREFIXES=('https://github.com/ankitdataanalyst14/')
  ENABLED=TRUE
  ALLOWED_AUTHENTICATION_SECRETS=(GIT_SECRET);

CREATE OR REPLACE GIT REPOSITORY SNOWFLAKE_AWA
  API_INTEGRATION=GIT_INT
  GIT_CREDENTIALS=GIT_SECRET
  ORIGIN='https://github.com/ankitdataanalyst14/SNOWFLAKE_AWA.git';

SHOW GIT REPOSITORIES;
DESCRIBE GIT REPOSITORY SNOWFLAKE_AWA;
SHOW GIT BRANCHES IN GIT REPOSITORY SNOWFLAKE_AWA;

CREATE OR REPLACE STAGE GITHUB.REPO_CLONE FILE_FORMAT=(TYPE='AUTO');

COPY FILES INTO @GITHUB.REPO_CLONE
  FROM @SNOWFLAKE_AWA/branches/main
  OVERWRITE=TRUE;

LIST @GITHUB.REPO_CLONE;

EXECUTE IMMEDIATE FROM @SNOWFLAKE_AWA/branches/main/sql/deploy_objects.sql;

CREATE OR REPLACE PROCEDURE GITHUB.RUN_RFM_FROM_GIT()
  RETURNS STRING
  LANGUAGE PYTHON
  RUNTIME_VERSION='3.10'
  HANDLER='handler'
  IMPORTS=('@SNOWFLAKE_AWA/branches/main/python/rfm_snowpark.py')
  PACKAGES=('snowflake-snowpark-python')
AS
$$
import rfm_snowpark
def handler(session):
    return "OK: " + str(rfm_snowpark.run_rfm(session))
$$;

CREATE OR REPLACE FUNCTION GITHUB.CALC_SCORE(x FLOAT)
```

```
  RETURNS FLOAT
  LANGUAGE PYTHON
  RUNTIME_VERSION='3.10'
  HANDLER='calc_score'
  IMPORTS=('@SNOWFLAKE_AWA/branches/main/utils/calc_score.py')
AS
$$
import calc_score
def calc_score(x): return calc_score.calc_score(x)
$$;

SHOW GIT COMMITS IN GIT REPOSITORY SNOWFLAKE_AWA;
LIST @SNOWFLAKE_AWA/branches/main;
LIST @SNOWFLAKE_AWA/tags;
```