Saturday, December 7, 2024

**Learning GitHub for MSSQL**

For us as a MSSQL DBA habitual to SSMS, let us take a step back and learn necessary GitHub and its integration with databases in a way that builds on our current expertise. We'll connect the dots between what we already know and what GitHub offers.

**What is GitHub?**

GitHub is an online platform for managing and collaborating on code. Think of it as a tool that tracks changes to scripts and files, much like version history in SSMS or backups in SQL Server, but more collaborative and powerful.

* **Repositories (Repos):** These are like folders or projects that store all our scripts, configurations, or files related to a database or application.
* **Commits:** Each "save" or change to a file is logged as a commit. It tracks:
  + Who made the change.
  + What changed.
  + When it was changed.
* **Branches:** Imagine having a "sandbox" environment for testing changes without affecting production. Branches let us experiment and merge changes back when ready.
* **Pull Requests:** A way to review and discuss changes before they’re applied, like a sign-off process for database changes.
* **GitHub Actions:** Automate tasks such as running scripts, applying changes, or scheduling backups.

**Why Should we learn About GitHub?**

As a DBA, we:

1. Write scripts (SQL, maintenance jobs, etc.).
2. Deploy changes (schema updates, permissions, data transformations).
3. Manage backups and recovery plans.

GitHub helps us in:

* **Organize SQL Scripts**: All our scripts are stored, versioned, and categorized.
* **Collaborate with Developers**: Align with teams that rely on the database for their applications.
* **Automate Repetitive Tasks**: Stop doing the same tasks manually—schedule them using workflows.
* **Trace Changes**: Know exactly who changed a stored procedure or table structure and when.

**How GitHub Compares to SSMS (For DBAs):**

|  |  |
| --- | --- |
| **SSMS** | **GitHub** |
| Manages database connections and queries | Manages SQL scripts and automates tasks |
| Local or server-based | Cloud-based, accessible anywhere |
| Manual backups/jobs | Automated via workflows |
| Script history if saved manually | Built-in version control and history |
| Jobs and schedules in SQL Agent | Automations with GitHub Actions |

**Key Concepts to Focus On as a DBA**

1. **Storing Scripts in GitHub**
   * All the SQL scripts (e.g., table creation, stored procedures, migration scripts) can be stored in a repository.
   * GitHub keeps track of every change, so we never lose your work.
2. **Automating Common DBA Tasks**
   * Use GitHub Actions to:
     + Apply SQL updates.
     + Backup databases.
     + Run scheduled cleanup tasks (like deleting old logs).
3. **Reviewing and Testing Changes**
   * Use "pull requests" in GitHub to have changes reviewed by other team members before they go live.
   * For example, before deploying a stored procedure, we can run it against a test database automatically.
4. **Integrating with Cloud SQL**
   * As a DBA, we already work with a database (SQL Server). GitHub is the layer above it, allowing us to manage our database scripts and automate operations seamlessly.

**Example 1: Schema Changes**

1. We write a script to add a new table:

CREATE TABLE Customers

(CustomerID INT PRIMARY KEY,

Name NVARCHAR(100),

Email NVARCHAR(100));

1. Save it in GitHub (e.g., in a folder like schemas/).
2. GitHub Actions runs the script on our Cloud SQL database automatically.
3. If the script works fine, it’s logged, and the team knows the database has been updated.

**Example 2: Automated Backups**

We currently schedule backups in SQL Server Agent. But Using GitHub:

1. We can Store a script that creates a backup in Cloud SQL.
2. Use GitHub Actions to run the script every night automatically.

**Example 3: Rolling Back Changes**

1. Let’s say a script we ran caused issues (e.g., deleted some important rows).
2. With GitHub, we can view the **history** of our scripts, identify the problem, and roll back to an earlier version of the script.
3. GitHub Actions can run the rollback script .

**Basic commands to learn:**

Git add <nameoffile> 🡪 this will add the file to the staging area

Or

Git add . 🡪 this will add all the files in that directory to the staging area

Git commit -m “Add a message ” 🡪 This will commit the changes made on the file to git

Git status 🡪 shows the current status of files

Git branch 🡪 shows all the branches of git on that repo/folder

Git checkout branchname🡪 moves you to other branch

Git branch -m main 🡪 renames the branch to main

BY default the name initial branch is master, as per new standards you can change it to main also.

Git remote set-url origin new.git.url/here 🡪 sets the url path of git repo

Git remote -v 🡪 shows the origin repo where the local repo is pushing or pulling the files

git config --global user.name "Your Name" 🡪 set the username for your git in local machine

git config --global user.email "you@example.com" 🡪 sets email for your git in local machine

To verify your settings, run git config –list 🡪 gives the config list such as name and emails set

Git push origin-u main 🡪 pushes the changes done on local repo to github repo

Git pull origin 🡪 it is make the local repo same as remote repo

Git merge main 🡪 merges the changes of other branch to main