Time Solution Tutorial Environment

By Eugene Asahara

Last Updated: June 4, 2025

This document walks you through setting up the development environment required to follow along with the tutorials in the book, *Time Molecules*. The environment includes SQL Server, Neo4j, and , Visual Code, and Python. Follow the instructions step-by-step.

Notes:

- Timesolution SQL Server Database (timesolution.bak).
    - The only material that isn't directly observable (i.e. code and data you can directly read) is the SQL Server database backup file, TimeSolution.bak.
    - TimeSolution.bak is around 50 MB, too big for Github (especially when updating). The file is hosted on a Onedrive storage specified in this document. Be sure to download this file only from that location.
    - The TimeSolution database comprises the vast majority of the tutorials in the book. The Neo4j and Python aspects of the tutorial are not required. This could simplify the installation to just SQL Server—which would be very convenient for those who are not software developers.

Prerequisites

This tutorial assumes you are working on a personal or work laptop. Most of the setup will take place locally.

Minimum Requirements

- Local admin rights – Needed to install:
  - SQL Server Developer Edition
  - Neo4j Desktop
  - Python (3.10.2 or later)
  - Git for Windows (includes Git Bash)
- **Internet access** – Required for downloading installers, extensions, cloning the GitHub repo, and accessing the sample database files.
- **Disk space** – At least 10 GB free for SQL Server, database files, Neo4j, and Python packages.
- **GitHub account**– For cloning the Time Molecules GitHub repository and participating in any future updates.
  - GitHub Desktop (optional) – A graphical interface for cloning and managing repositories.
  - **Git Bash (optional)** – Used to validate the authenticity of the provided sample .bak file using GPG. *Installed automatically with Git for Windows*.

Alternative Arrangements

If you're working on a **restricted corporate laptop** or already have parts of the stack available, you may not need to install everything:

- **Python:**
  - If Python is already installed and you're using an IDE like PyCharm, Anaconda, or JupyterLab, you can use that instead of VS Code.
- **SQL Server:**
  - If you already have access to a SQL Server instance (either local or remote), you can restore the TimeSolution.bak database there.
  - **Important:** You must have permission to create or restore databases.
- **Neo4j:**
  - If you're already using Neo4j (Desktop, Server, or Aura), you may skip Neo4j Desktop installation.
  - **Important:** You must be able to load plugins (e.g., APOC and n10s) and import data into the database.

This tutorial is modular—most of the core work happens in SQL Server. Python and Neo4j provide optional enhancements, analytics, and visualizations.

Clone the Time Molecules Repository

Before setting up Python and VS Code, you'll need to clone the Time Molecules GitHub repository to your local machine. This will give you access to all the scripts, notebooks, and configuration files used throughout the setup.

a. Create a Local Folder

I recommend creating a base directory to hold the project: *C:\MapRock*

If this folder doesn't exist, create it manually or let Git do it during the clone step.

b. Install Git (if not already installed)

Download Git for Windows, which includes Git Bash:

- https://git-scm.com/download/win

Follow the installation prompts. Leave most options at their defaults.

c. Clone the Repository

Open **Git Bash** (or Command Prompt if Git is on your PATH), and run:

git clone https://github.com/MapRock/TimeMolecules.git C:/MapRock/TimeMolecules

This will download the full repository contents into C:\MapRock\TimeMolecules.

SQL Server Setup

a. Install SQL Server Developer Edition (latest version)

The Developer Edition of SQL Server provides the full feature set of the Enterprise Edition, but it's licensed for development and testing only.

Steps:

1. Open a browser and navigate to: https://www.microsoft.com/en-us/sql-server/sql-server-downloads
2. Under "Developer edition", click the **Download now** button.
3. When prompted, choose to run the installer or save it and run later.
4. The SQL Server Installation Center will open. Choose **Basic** installation for quick setup.
5. Accept the license terms, then click **Install**.
6. Wait for the installation to complete. It will show you the instance name (default is SQLEXPRESS or MSSQLSERVER) and a summary.
7. Note the SQL Server instance name, since you will use this when connecting via Management Studio or scripts.
8. Optionally, you may want to open **SQL Server Configuration Manager** and ensure TCP/IP is enabled if remote access or ports are involved.

b. Install SQL Server Management Studio (SSMS)

- Visit: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms
- Download and install the latest version of SSMS.

c. Download and Validate the Time Solution Database

This step is optional, but recommended if you want to explore *Time Molecules* hands-on. The .bak file is a standard SQL Server backup file, which is not an executable and poses very little risk from a virus perspective. However, as with any downloaded file—especially one that restores a database—it's wise to validate its authenticity.

That's why I've included a digital signature you can verify using GPG. It's a simple way to confirm the file hasn't been tampered with and that it came from me. While a .bak file is unlikely to contain malicious code, databases can contain unexpected data, settings, or objects—so this extra step helps ensure integrity and trust.

- Download TimeSolution.bak from the provided OneDrive link:
    - https://1drv.ms/u/c/7d94c9ab48b30303/EQDqhzDQZ4RGnjKh-NSXabsBhVUG3QSfMlAcqmz_0CF4Kw?e=hyogeA
    - Three files:
        - Timesolution.bak

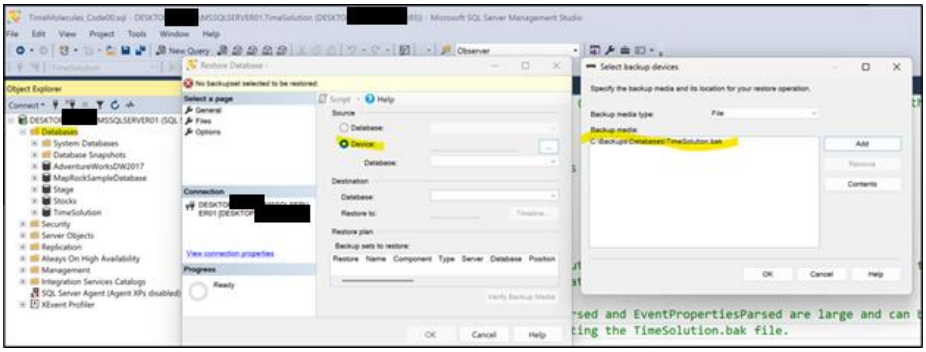
        - publickeytm.asc
        - timesolution.bak.asc
    - Assuming a download of three files to:
        - C:\MapRock\TimeMolecules\data\
- Validate with Git Bash.
    - Open Git Bash.
    - Navigate to data directory: ***cd /c/MapRock/TimeMolecules/data***
    - Import public key: ***gpg --import publickeytm.asc***
    - Run validation command: ***gpg --verify timesolution.bak.asc timesolution.bak***
        - You Should see message: **`Good signature from "Eugene Asahara (For Time Molecules) <eugene@softcodedlogic.com>"`**

```
easah@EAA2024 MINGW64 /c/maprock/timemolecules/data
$ gpg --verify timesolution.bak.asc timesolution.bak
gpg: Signature made Fri May 30 06:04:12 2025 MDT
gpg:                using RSA key DDFF0614BF5C3F81FFBC08BA64A0A449AC63537D
gpg: Good signature from "Eugene Asahara (For Time Molecules) <eugene@softcodedlogi
c.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: DDFF 0614 BF5C 3F81 FFBC  08BA 64A0 A449 AC63 537D
```

        - You may see the error: *gpg: WARNING: This key is not certified with a trusted signature!*
            - This isn't an error, just a *trust model warning*. It means GPG verified the signature cryptographically, but you haven't explicitly trusted the key in your local keyring yet. GPG doesn't know if you really trust that *this* public key belongs to Eugene Asahara—because no one has signed it in a "web of trust."

d. Restore the TimeSolution Database

- Open SSMS and connect to your SQL Server instance.
- Right-click on "Databases" > "Restore Database..."
- Choose **Device**, click **Add**, and locate the TimeSolution.bak file.
- Select the backup, ensure the restore paths are correct, and proceed.



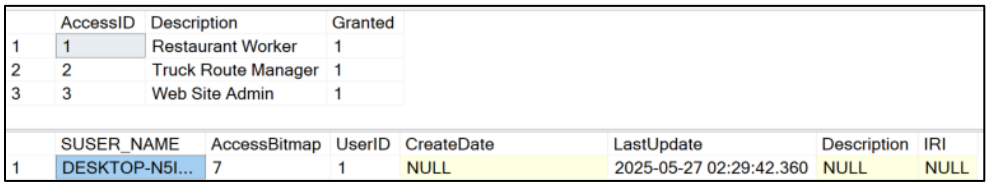
d. Initialization Script

Execute the TSQL script: *c:\MapRock\TimeMolecules\book_code\sql\TimeMolecules_Code00.sql*

This is also a good way to ensure your SQL Server is set up. The script does two things:

1. Adds you as a user to the dbo.Users table.
2. Parses case and event properties—the CasePropertiesParsed and EventPropertiesParsed tables, respectively. I truncated them in order to significantly reduce the size of the TimeSolution.bak file. This part takes a few minutes.

You should see a result like this:



	AccessID	Description	Granted
1	1	Restaurant Worker	1
2	2	Truck Route Manager	1
3	3	Web Site Admin	1

	SUSER_NAME	AccessBitmap	UserID	CreateDate	LastUpdate	Description	IRI
1	DESKTOP-N5I...	7	1	NULL	2025-05-27 02:29:42.360	NULL	NULL

e. Securing Access to User Access Rights (Bitmap Access Model)

The Time Solution database includes a security model that restricts direct access to the dbo.Users table. Each user's access is defined by a bitmap, which encodes their permissions across rows in the dbo.Access table.

To enforce access control while preserving query simplicity and performance, all permission lookups are exposed via a secure scalar function: dbo.UserAccessBitmap.

Function-Based Access Pattern:

- Function: dbo.UserAccessBitmap
- Purpose: Returns the access bitmap for the current login, based on their row in dbo.Users.
- Security Enforcement:
  The function encapsulates access to dbo.Users using SQL Server's ownership chaining model. This allows a user to execute the function without needing direct access to the underlying table.

Key Implementation Details:

- dbo.UserAccessBitmap uses SUSER_NAME() (or optionally SUSER_SID()) to resolve the current user's identity and locate the corresponding access bitmap.
- The function and dbo.Users must share the same schema owner (typically dbo) to ensure ownership chaining applies.
- Direct access to the dbo.Users table should be revoked or denied for all non-administrative users.

Permissions Setup

Object	Type	Permission	Notes

dbo.UserAccessBitmap	Scalar Func	GRANT EXECUTE	Grant to users or roles that need access evaluation
dbo.Users	Table	DENY SELECT or REVOKE	Prevents direct access; enforced via function layer

Applications or queries needing to evaluate access should use this function:

SELECT dbo.UserAccessBitmap() AS UserAccessMask;

This will return a BIGINT bitmap that maps to access entries in the dbo.Access table (e.g., bit 1 = row 1, bit 2 = row 2, etc.).

f. Restore AdventureWorksDW2017 (optional)

1. Go to the official Microsoft GitHub repository for sample databases: https://github.com/microsoft/sql-server-samples/releases/tag/adventureworks
    1. Scroll down to the Assets section under "AdventureWorks 2017 OLAP".
    2. Download the file: AdventureWorksDW2017.bak
2. Restore the Database in SSMS
    1. Open SQL Server Management Studio (SSMS).
    2. Connect to your SQL Server instance.
    3. Right-click on Databases → choose Restore Database...
    4. Select Device → Click the "..." → Add
       → Locate and select AdventureWorksDW2017.bak.
    5. In the Files tab, verify the restore paths (or change to a writable directory).
    6. Click OK to restore.

Neo4j Setup

a. Install Neo4j Desktop

- Visit: https://neo4j.com/download/
- Download and install Neo4j Desktop.
- Create a new database project and set the base folder to C:\MapRock\Neo4j

b. Configure Neo4j Plugins

- Add the APOC plugin.
- Add the Semantic Web plugin (RDF Plugin). To do this:
    - In Neo4j Desktop, click on the database > "Plugins"
    - Install **APOC** and **Neosemantics (n10s)**

c. Start the database. Ensure it is accessible from bolt://localhost:7687

Python and VS Code Setup

a. Install Python (version 3.10.2 or later)

- Visit: https://www.python.org/downloads/
- Download and install version 3.10.2 or later.
- Add Python to PATH during installation.

b. Install Visual Studio Code (VS Code)

- Visit: https://code.visualstudio.com/
- Download and install VS Code.

c. Add VS Code Extensions

- Launch VS Code.
- Click the Extensions icon in the Activity Bar on the side (or press Ctrl+Shift+X).
- In the Extensions pane, type the following names one by one into the search bar:
    - Jupyter by Microsoft: Enables support for Jupyter notebooks (.ipynb files).
    - Neo4j for VS Code by neo4j.com: Provides database integration, query execution, and exploration for Neo4j.
    - Cypher Query Language: Adds syntax highlighting for Cypher if not already covered by the Neo4j extension.
- Click Install on each relevant result.
- Reload VS Code when prompted after installation.

d. Clone the GitHub Repository

- Run: git clone https://github.com/MapRock/TimeMolecules.git C:\MapRock\TimeMolecules

e. Open Project in VS Code

- Launch VS Code.
- Open the folder: C:\MapRock\TimeMolecules\book_code\src

f. Set Up the Environment Variables

- Select the .env file in the src folder and set the following contents:
    - OPENAI_API_KEY="Your OpenAI API Key"
    - CYPHER_LOAD_DIR="C:/Databases/Neo4j/relate-data/dbmss/dbms-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/import/"
        - **To find the correct CYPHER_LOAD_DIR path**:
            1. In Neo4j Desktop, go to the running database.

2. Click the three dots > "Open Folder"
3. Navigate to import folder within the dbms-* directory.
4. Copy the full path and replace xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx with the actual ID.

With this environment, you're ready to run the Time Solution components across SQL Server, Neo4j, and Python.