4. Keeping Backups

Having established the overall structure of our vault, it's crucial to proceed with creating a backup in the cloud. This step ensures that we safeguard our information against loss due to potential unlucky accidents.

4.1 Plugins Needed for This Methodology

In this guide, we'll explore cloud storage options for your notes, focusing on a free solution. There are two primary methods to consider:

1. **Git Plugin**: This is a free community plugin that enables you to synchronize your data with a GitHub repository. It's a cost-effective way to ensure your notes are backed up and accessible.[1]
2. **Obsidian Sync Plugin**: This built-in plugin offers synchronization services for $4 USD per month.[2] It provides cloud storage and cross-platform capabilities, allowing you to access your vault from Apple devices and other platforms.

Given our focus on affordability and accessibility, we will concentrate on the Git Plugin as our chosen solution for cloud storage.

4.2 Implementing Backups Using GitHub

Let's now discuss the backup implementation.

4.2.1 Installing Git Plugin

Start by creating a new GitHub account and downloading the Git plugin.

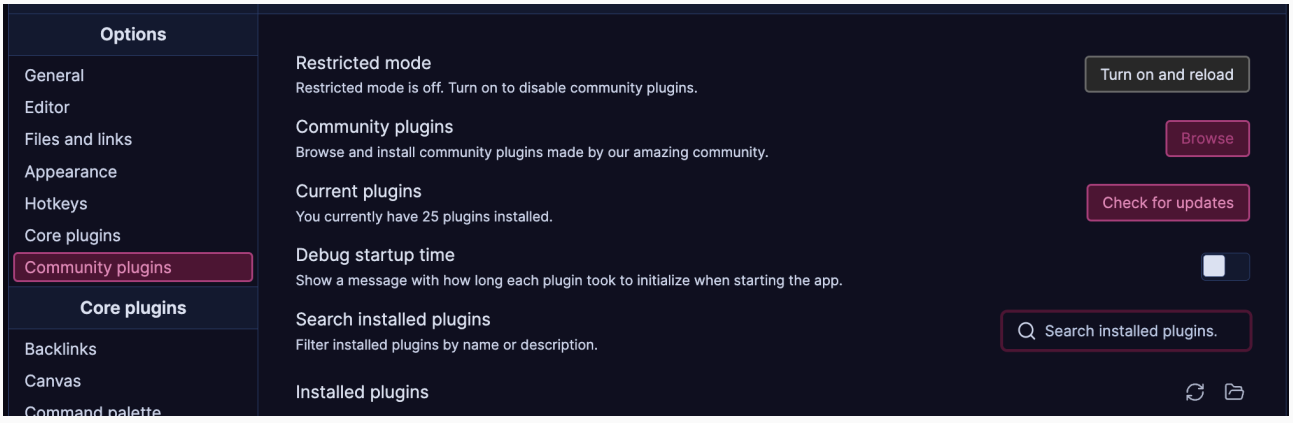
Creating GitHub Account and Downloading Git Plugin

1. *Create a GitHub Account*: If you don't already have one, sign up at GitHub. For detailed instructions on setting up your account, consult this helpful article.

---

2. *Access the Git plugin*:

- Navigate to the settings window by selecting *Options*.
- Go to *Community Plugins* > *Browse* to view available plugins.

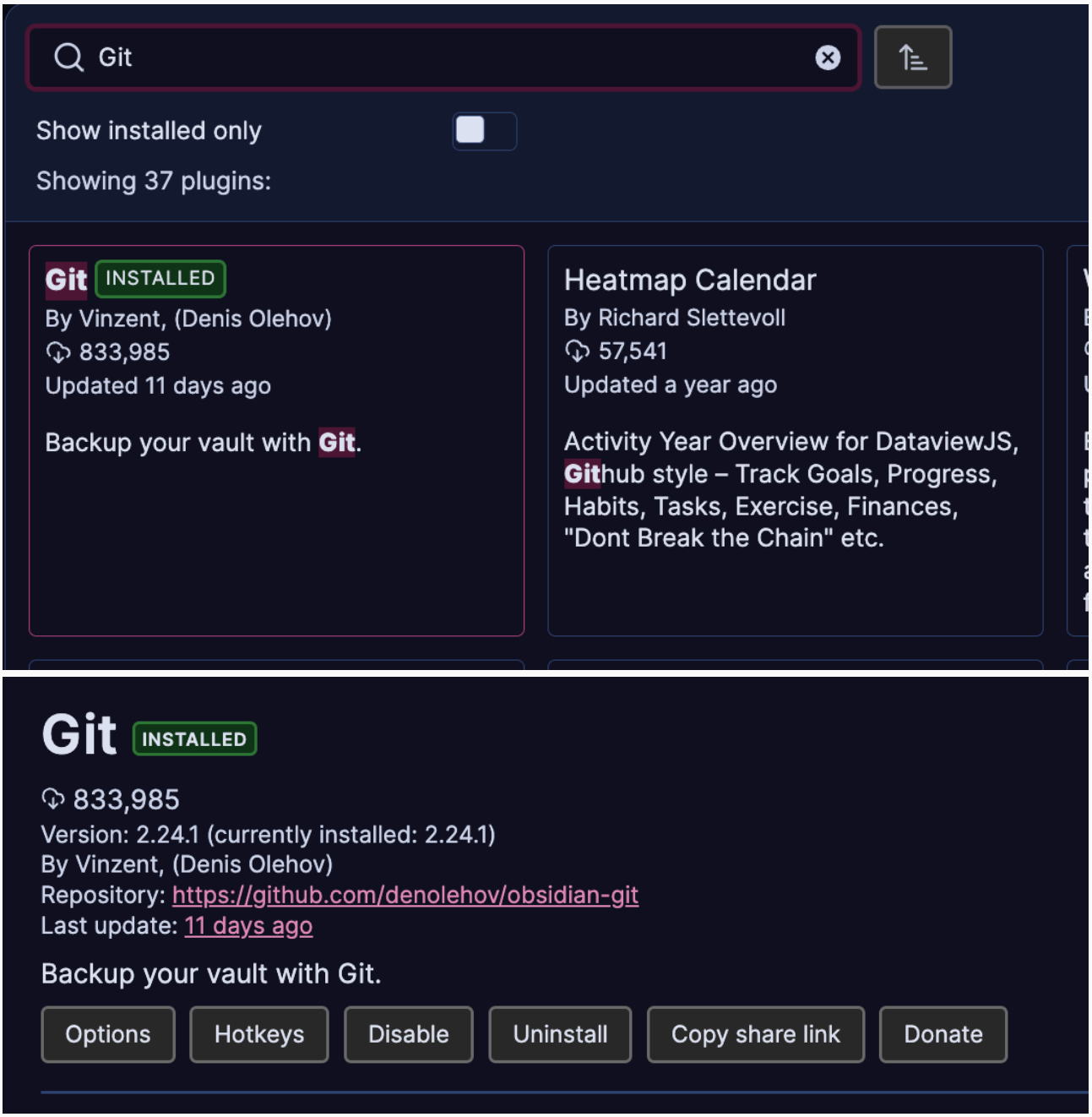


---

3. *Install the Git plugin*:

- In the *Browse* section, use the search bar to find the "Git" plugin.
- Click on the first result to view the plugin's details.
- Select *Install* to download the plugin to your Obsidian vault.

- Once installed, click *Enable* to activate the plugin. This will allow Obsidian to sync your notes with a GitHub repository.



4.2.2 Setting Up Authentication

Next, to enable the Git plugin in Obsidian to synchronize your notes with GitHub, you need to *set up GitHub authentication via SSH Keys*. This step ensures secure, automatic uploads without the need for manual user interaction each time. Here's how to proceed:

Setting Up GitHub Authentication

1. *Generate SSH Keys for GitHub Authentication*: Begin by creating a new SSH key specifically for GitHub. This key will serve as a secure means for your vault to communicate with GitHub, facilitating the automatic syncing of your notes. Follow the detailed instructions provided in the GitHub documentation: Generating SSH Keys for GitHub.

2. *Add Your SSH Key to the ssh-agent*: After generating your SSH key, the next step is to add it to the ssh-agent. The ssh-agent is a program that holds private keys used for SSH authentication, making it easier to manage keys and passphrases. Detailed guidance for this process can be found here: Generating a new SSH key and adding it to the ssh-agent - GitHub Docs.

> Although there are various methods for authenticating with GitHub, using SSH keys is preferred for its simplicity. Once set up, the Git plugin can automatically handle the upload process to GitHub, ensuring your notes are always backed up securely without additional effort on your part. For other authentication methods using the Git plugin, check out the official Git plugin documentation.

4.2.3 Linking Your Vault to GitHub

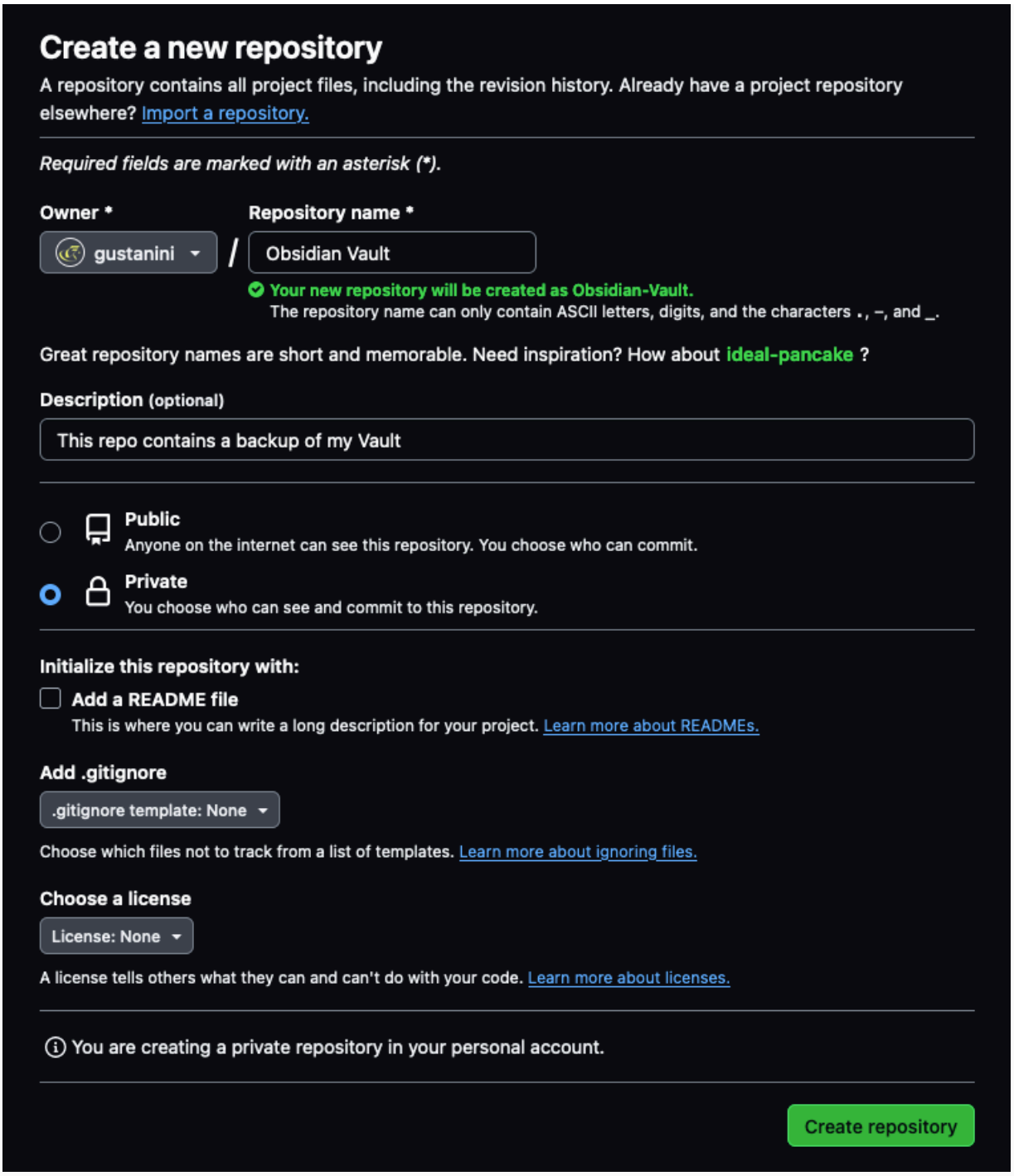
Now that you have an account, authentication is configured and the Git plugin is installed, you can link your vault to your repository.

Linking Your Vault to a GitHub Repository

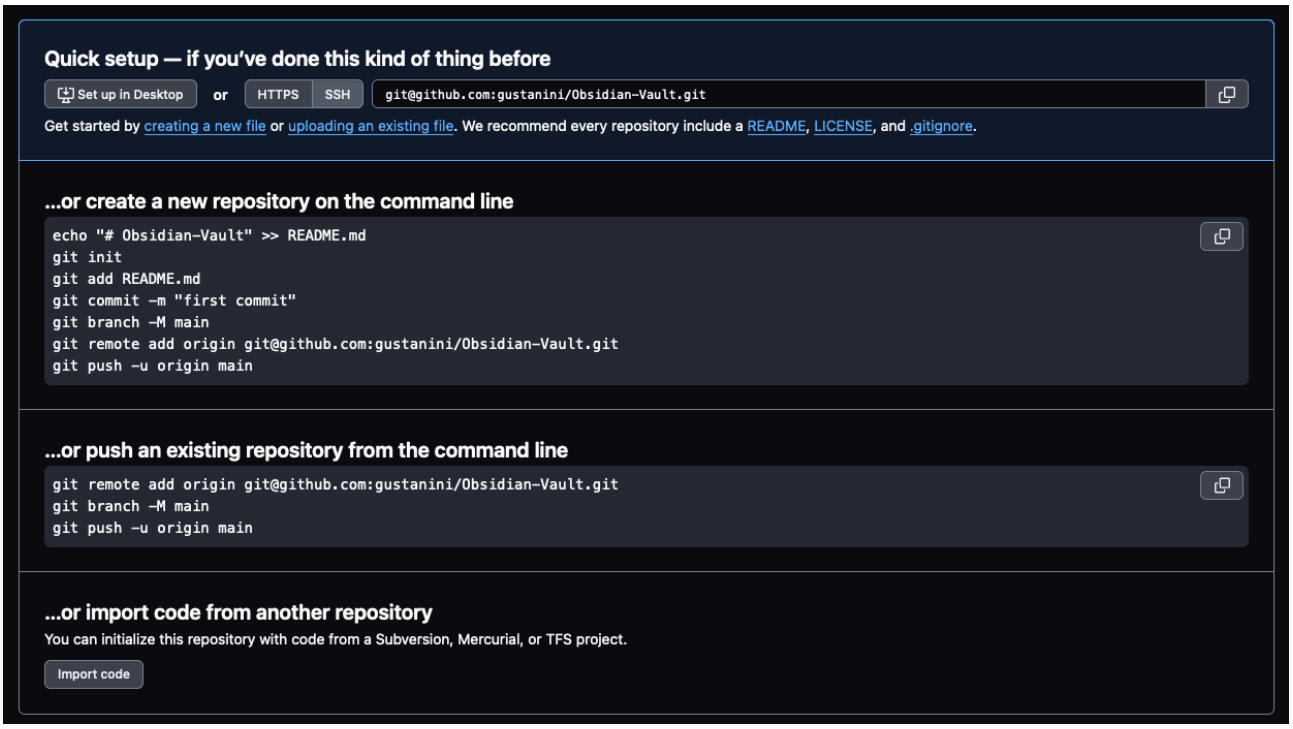
1. *Create a new private repository on GitHub*:

- Navigate to *Your Profile* on GitHub, then click on *Repositories*.

- Select *New* to create a new repository. When filling out the form, ensure that the repository visibility is set to **private** to keep your notes confidential.

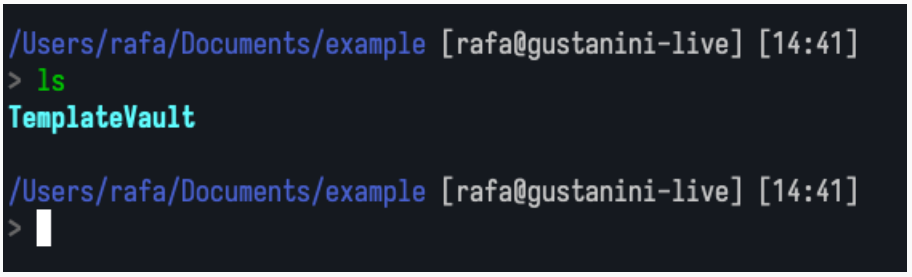


- After filling out the form, click on *Create Repository*. Keep the window open as you'll need the information it provides for later steps.



2. *Initialize Your Local Repository*:

- Open a command prompt or console on your computer.
- Navigate to your desired directory where you wish to store your vault, for example, your Documents folder. You can create a new directory specifically for your vault using the commands `mkdir example; cd example`, replacing "example" with your preferred directory name.
- If your vault is not already in this directory, move it here now.



3. *Set Up Git in Your Vault Directory*:

- In the command prompt or console, within your vault's directory, initialize a new Git repository by running:

```
git init
```

- Add all files in the directory to the Git staging area with:

```
git add *
```

- Make your first commit to record the addition of your files to the repository:

```
git commit -m "first commit"
```

```
/Users/rafa/Documents/example [rafa@gustanini-live] [14:46]
> git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/rafa/Documents/example/.git/

/Users/rafa/Documents/example [git::master] [rafa@gustanini-live] [14:46]
> git add *

/Users/rafa/Documents/example [git::master *] [rafa@gustanini-live] [14:46]
> git commit -m "first commit"
```

- Rename your default branch to `main` using:

```
git branch -M main
```

- Link your local repository to your GitHub repository by using the command provided in the GitHub setup window, replacing the example URL with your repository's URL:

```
git remote add origin git@github.com:username/repository-name.git
```

- Finally, push your local repository to GitHub:

```
git push -u origin main
```

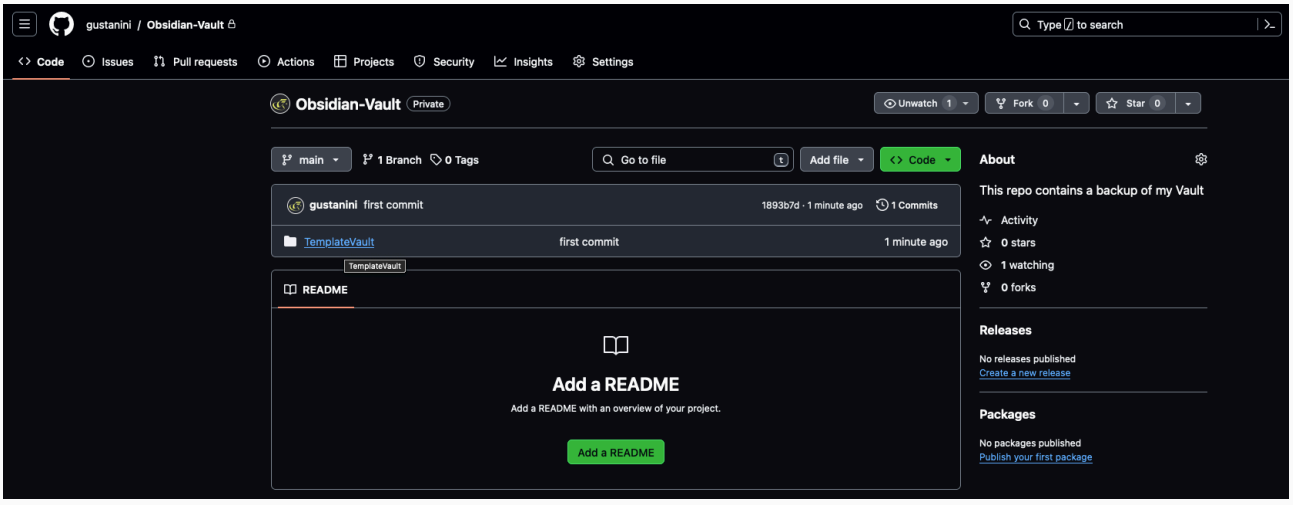
```
/Users/rafa/Documents/example [git::master] [rafa@gustanini-live] [14:46]
> git branch -M main

/Users/rafa/Documents/example [git::main] [rafa@gustanini-live] [14:47]
> git remote add origin git@github.com:gustanini/Obsidian-Vault.git

/Users/rafa/Documents/example [git::main] [rafa@gustanini-live] [14:47]
> git push -u origin main
Enumerating objects: 177, done.
Counting objects: 100% (177/177), done.
Delta compression using up to 8 threads
Compressing objects: 100% (167/167), done.
Writing objects: 100% (177/177), 12.48 MiB | 3.89 MiB/s, done.
Total 177 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To github.com:gustanini/Obsidian-Vault.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

/Users/rafa/Documents/example [git::main] [rafa@gustanini-live] [14:47]
>
```

4. *Verify Your Backup on GitHub*: After completing the push command, visit your GitHub repository's URL to ensure your vault's files are now hosted on GitHub. You should see all the files you've committed, indicating a successful backup.



4.2.4 Setting Up Git Plugin

For the final step in securing and automating your Obsidian vault backup using GitHub, you'll *configure the Git Plugin settings within Obsidian*. This ensures your vault is regularly synced with your GitHub repository, maintaining an up-to-date backup without manual intervention.

Git Plugin Settings

- Navigate to the Git Plugin settings in Obsidian.
- Look for the *Automatic* section, specifically the *Vault Backup Interval (minutes)* setting.
- Adjust this setting to **60 minutes**. This frequency means that your vault will automatically sync with your GitHub repository every hour, ensuring your notes are regularly backed up and up to date.

Automatic

Split automatic commit and push
Enable to use separate timer for commit and push

Vault backup interval (minutes)
Commit and push changes every X minutes. Set to 0 (default) to disable. (See below setting for further configuration!)
60

Auto backup after latest commit
If turned on, set last auto backup time to latest commit

Auto pull interval (minutes)
Pull changes every X minutes. Set to 0 (default) to disable.
0

Specify custom commit message on auto backup
You will get a pop up to specify your message

Commit message on auto backup/commit
Available placeholders: {{date}} (see below), {{hostname}} (see below), {{numFiles}} (number of changed files in the commit) and {{files}} (changed files in commit message)
vault backup: {{date}}

While the backup interval is the crucial setting to adjust for regular syncing, the Git Plugin comes with several other options that you might find useful depending on your specific needs. However, for the purpose of this guide and to ensure a smooth backup process, replicating the settings shown in the previous image should suffice.

Feel free to customize the rest of settings according to your personal needs. The following are pictures from my personal settings (most settings, if not all, are in their default value).

Commit message

Commit message on manual backup/commit
Available placeholders: {{date}} (see below), {{hostname}} (see below), {{numFiles}} (number of changed files in the commit) and {{files}} (changed files in commit message)
vault backup: {{date}}

{{date}} placeholder format
Specify custom date format. E.g. "YYYY-MM-DD HH:mm:ss. See Moment.js for more formats.
YYYY-MM-DD HH:mm:s

{{hostname}} placeholder replacement
Specify custom hostname for every device.

Preview commit message
Preview

List filenames affected by commit in the commit body

Backup

Sync Method
Selects the method used for handling new changes found in your remote git repository.
Merge

Pull updates on startup
Automatically pull updates when Obsidian starts

Push on backup
Disable to only commit changes

Pull changes before push
Commit -> pull -> push (Only if pushing is enabled)

History View

Show Author
Show the author of the commit in the history view

Hide

Show Date
Show the date of the commit in the history view. The {{date}} placeholder format is used to display the date.

Source Control View

Automatically refresh Source Control View on file changes
On slower machines this may cause lags. If so, just disable this option

Source Control View refresh interval
Milliseconds to wait after file change before refreshing the Source Control View

7000

Miscellaneous

Disable notifications
Disable notifications for git operations to minimize distraction (refer to status bar for updates). Errors are still shown as notifications even if you enable this setting

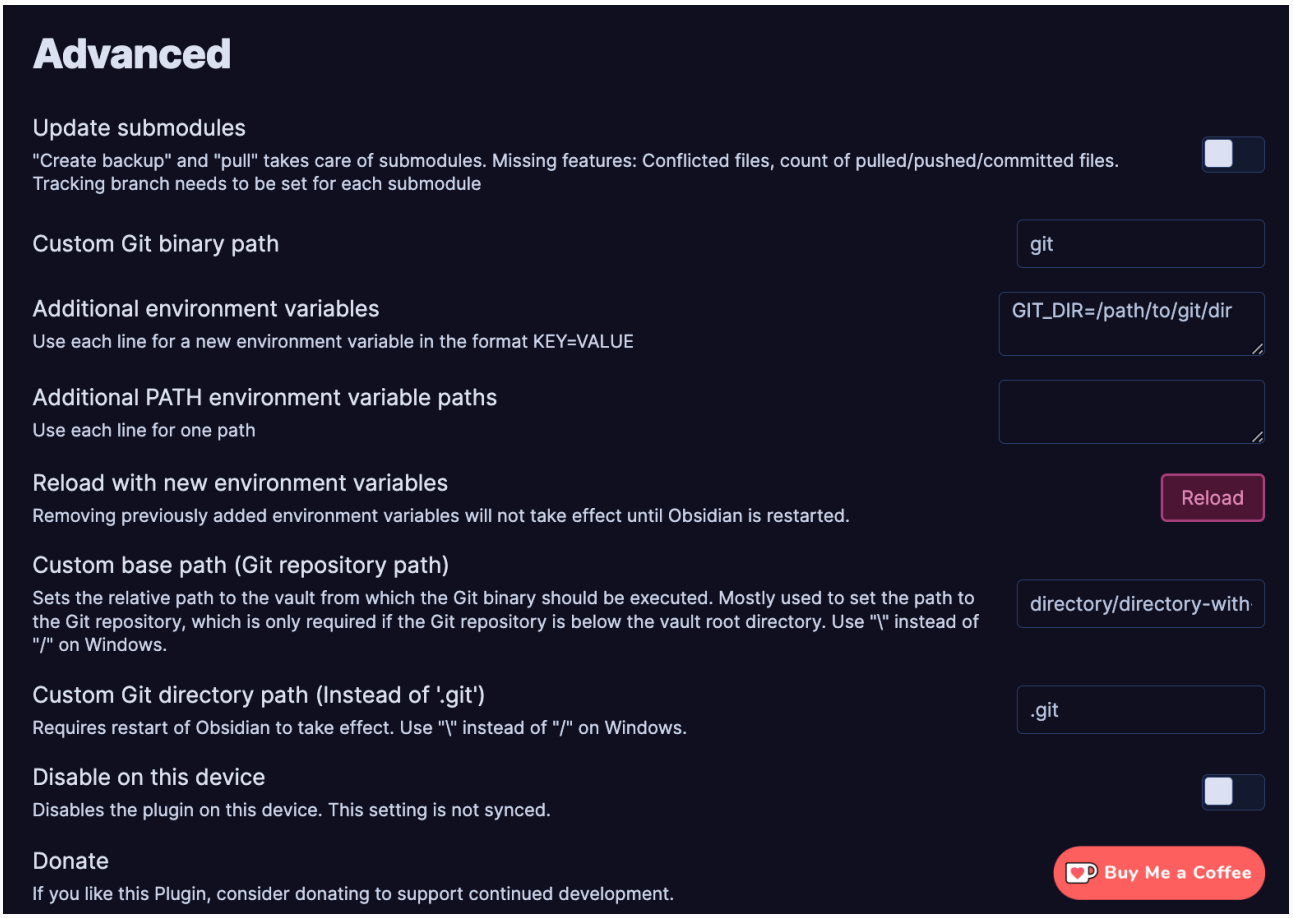
Hide notifications for no changes
Don't show notifications when there are no changes to commit/push

Show status bar
Obsidian must be restarted for the changes to take affect

Show stage/unstage button in file menu

Show branch status bar
Obsidian must be restarted for the changes to take affect

Show the count of modified files in the status bar



4.3 Manually Syncing Your Vault

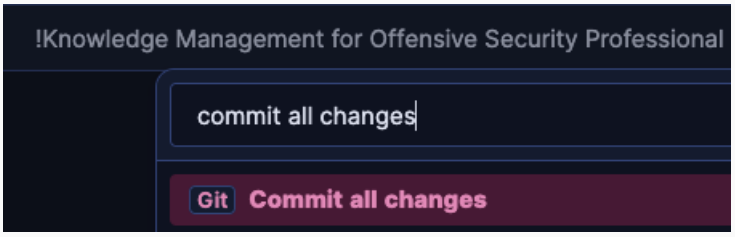
With the backup setup finalized, your vault is now securely hosted in the cloud and configured to automatically sync every hour. However, there may be times when you want to immediately sync your changes without waiting for the next automatic interval. For these situations, the Git Plugin provides an option for manual synchronization.

Manually Syncing Your Vault with GitHub

Access the Git Plugin Commands:

- Use the command palette by pressing `Ctrl+P` (or `Cmd+P` on macOS), then search for the following Git commands:

"Git: Commit all changes"



And "Git: Push".

This feature is especially useful after making significant updates to your notes or before performing operations that could impact your data, providing an additional safety net to ensure your information is always protected.

References

1. What Is GitHub? A Beginner's Introduction to GitHub ↩
2. Obsidian Sync ↩