6. Metadata

Metadata is an important aspect of organization. Obsidian includes a built-in feature called "properties." Before continuing with this tutorial, I suggest getting familiar with it by reading the properties documentation. Properties are your notes' YAML Frontmatter[1] and it contains identifying information for each note.

So, how is metadata useful for you? In a nutshell, adding metadata at the top of your notes helps keep them organized using dates, tags, custom links, and other custom parameters.

The interesting part of using metadata is that you can easily search in your vault for specific topics. Every note is linked to its parent note type, and there is an extremely useful plugin called Dataview that allows you to automatically create dynamic tables, lists, and more by leveraging tags in your metadata.

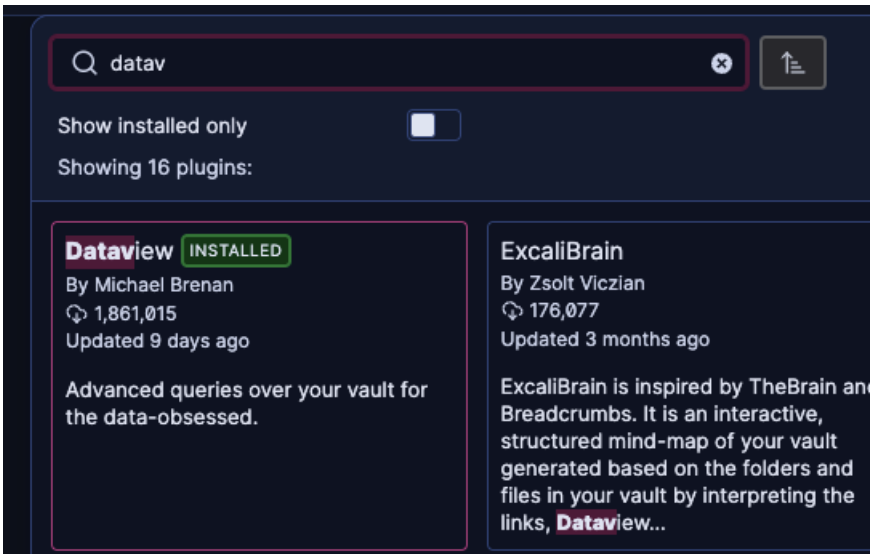
6.1 Plugins Needed for This Methodology

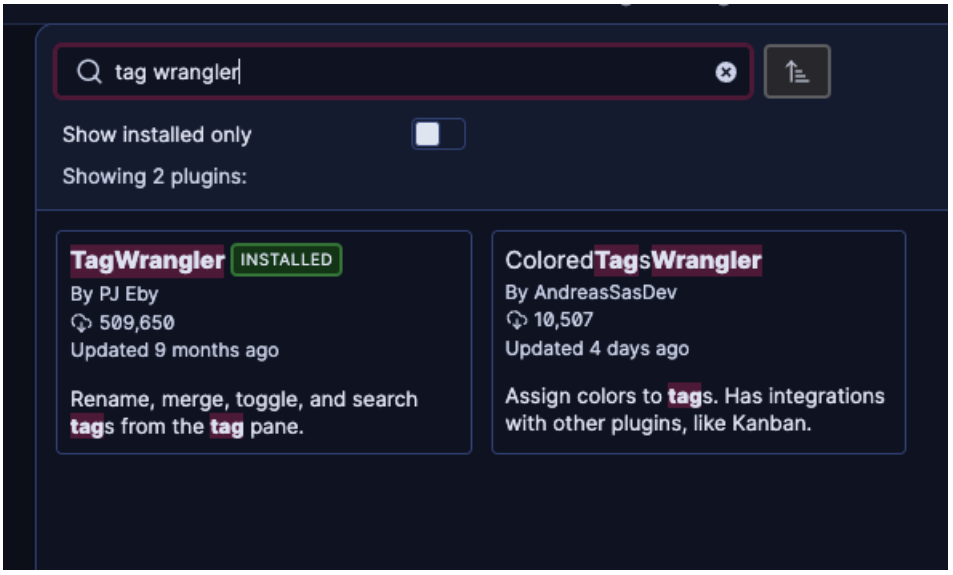
For this methodology, you will need the following plugin(s):

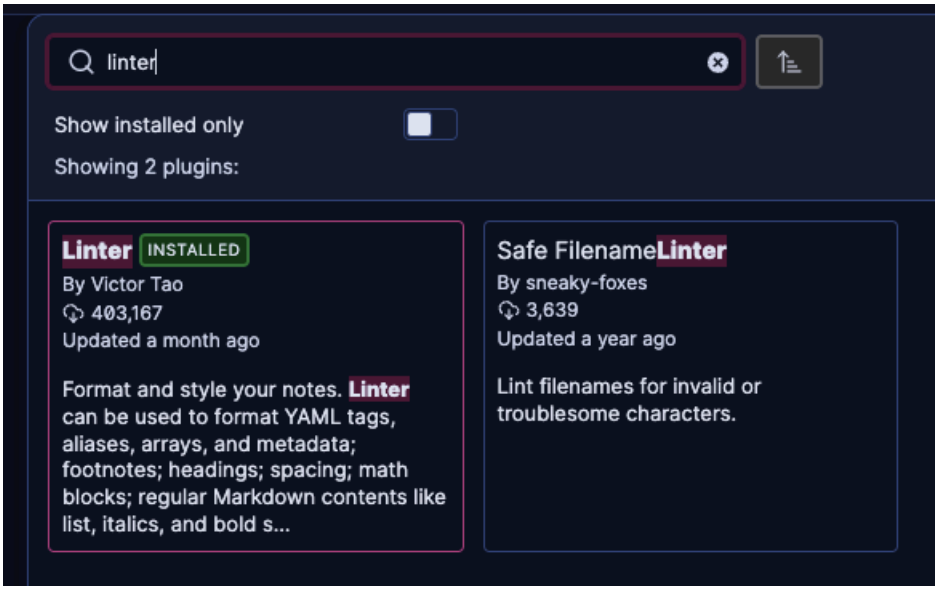
1. **Dataview**: Dataview is a powerful plugin that lets you query your whole vault for notes that match certain criteria. It uses a easy to learn SQL-like syntax for this purpose. This plugin is essential when moving onto more advanced methodologies like creating diagrams, custom cheatsheets and others.
2. **Linter**: Linter is a formatting plugin with several interesting features. I use this plugin all the time when writing notes to keep a consistent style throughout all my notes. It contains a metadata formatting feature that will automatically create or populate your YAML Frontmatter with custom data saving you lots of time.
3. **TagWrangler**: TagWrangler is a tag managing plugin. It will allow you to view and edit all tags present throughout your vault. If for any reason you need to change a tag's name to something else (very common in my experience) this plugin's got you covered.

6.2 How to Implement This Methodology

Proceed to installing these three plugins from the community plugins tab.





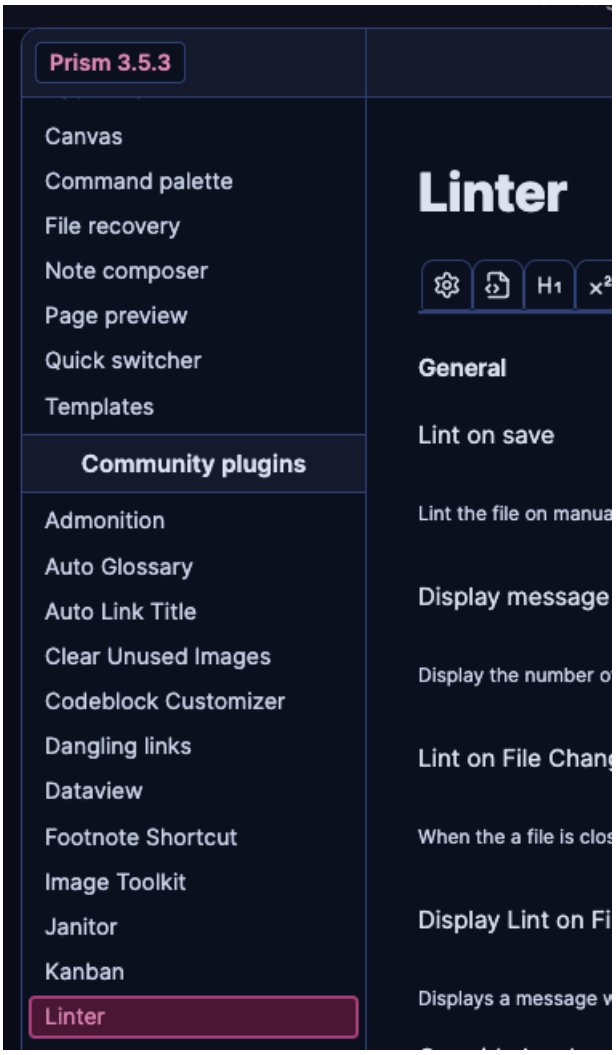


Once our plugins are installed, we can move on to setting them up. You can leave TagWrangler (which does not support customization) and Dataview with their default settings. That leaves us with Linter.

6.3.1 Setting Up Linter

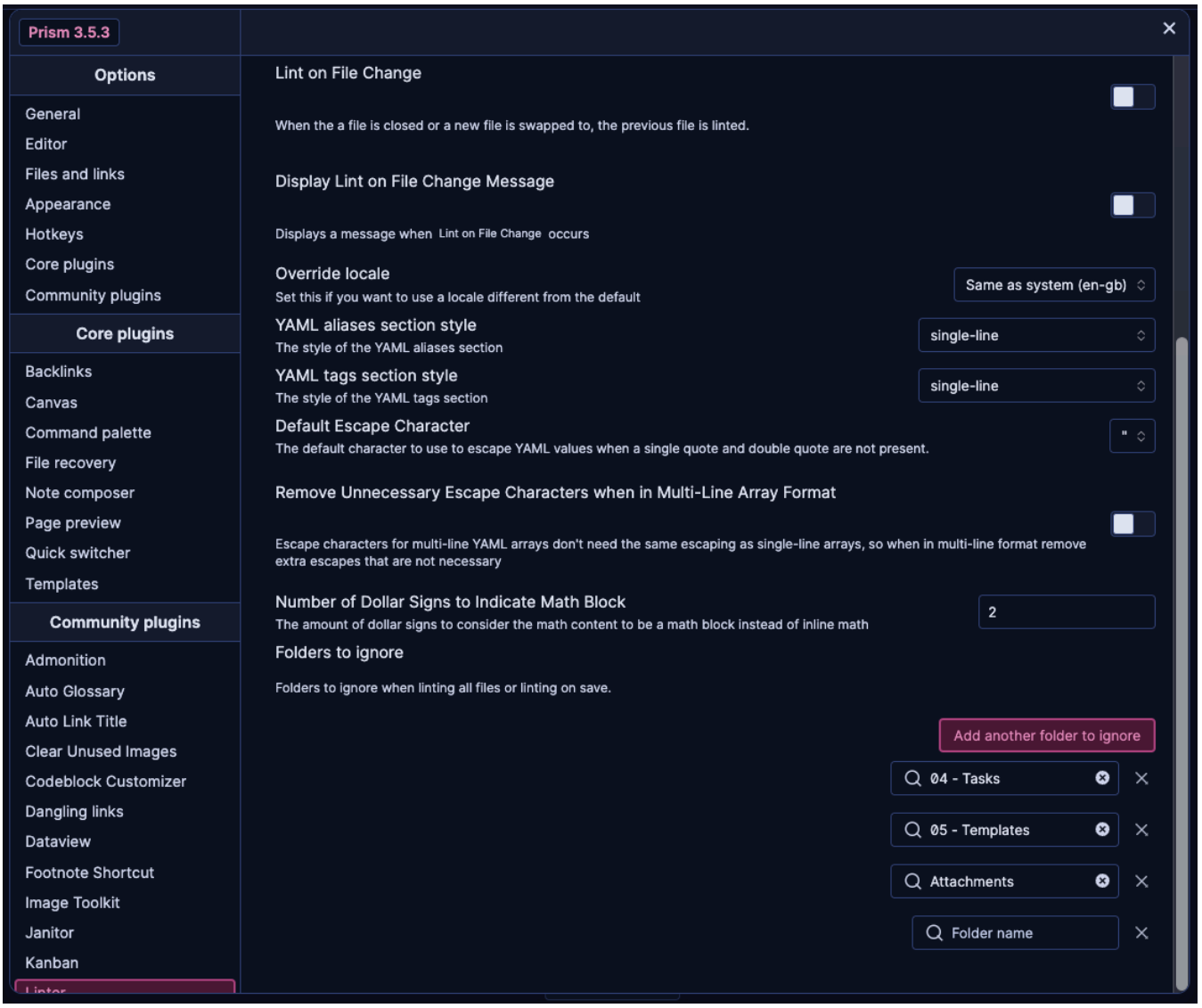
The setup is very straightforward, and you will want to use settings that format notes to your personal liking. I will be showing screenshots of the most important settings as there are too many to list here.

> If you want to see all my custom settings, feel free to open the Obsidian settings window and click on Linter under Community Plugins on the left.



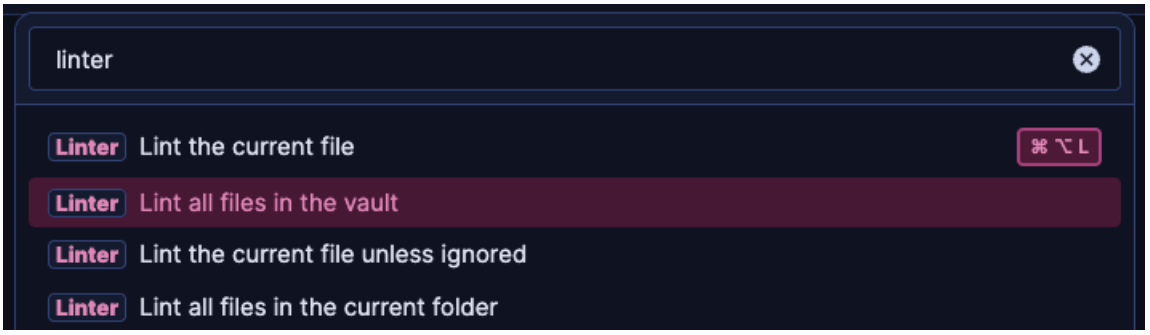
6.3.1.1 General Tab

Under *General*, scroll to the bottom and go to *Folders to Ignore*. Add your "Tasks", "Templates", "Attachments", and any other miscellaneous folders you might have.



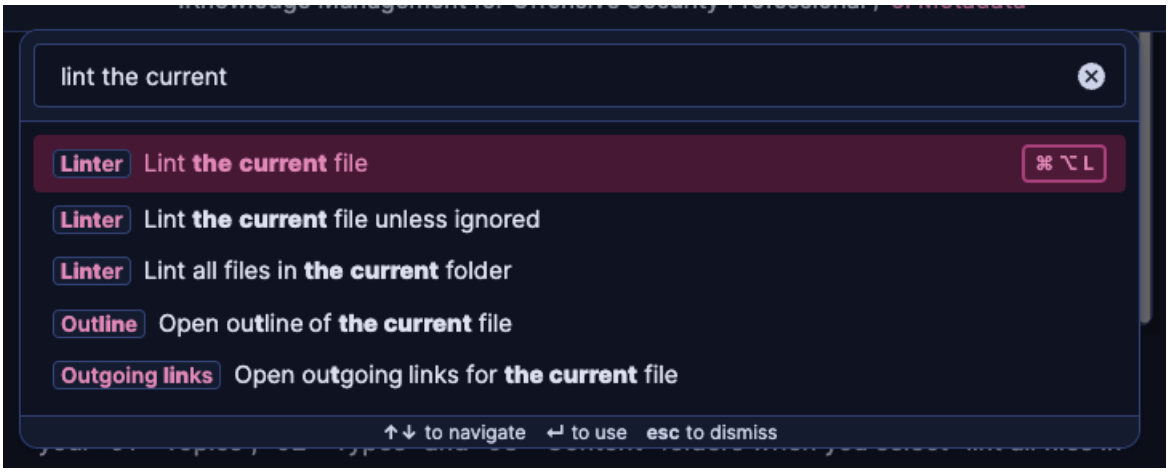
This setting will prevent adding metadata and formatting settings to notes that are outside your "01 - Topics", "02 - Types", and "03 - Content" folders when you select "lint all files in the current folder" or "lint all files in the vault" options.

Remember, these three folders contain all your notes that you want to keep formatted and properly classified, while other folders contain templates and miscellaneous files that you probably don't want to edit.

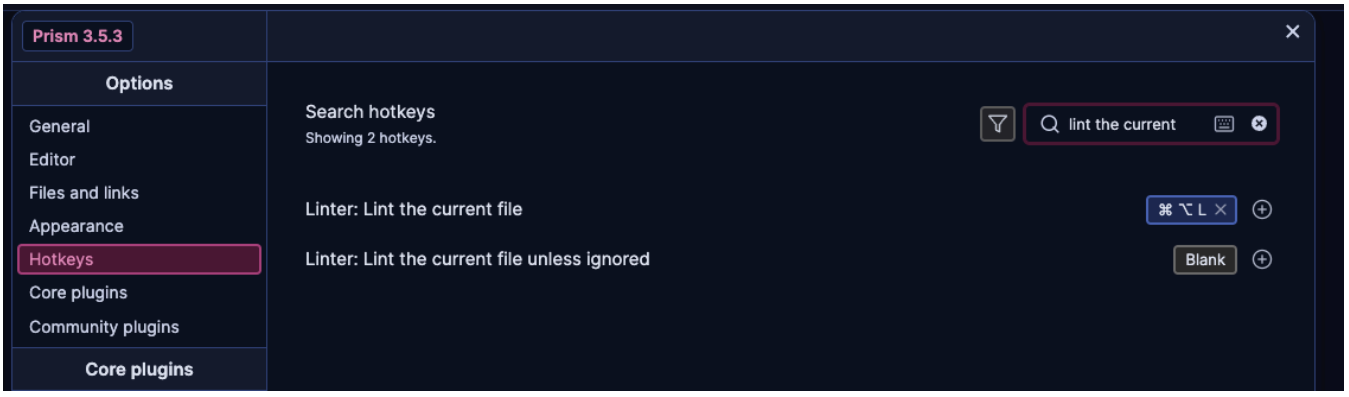


6.3.1.2 Linter Hotkey

I suggest adding a HotKey for the "Lint the Current File" command. Hit Command + P or Ctrl + P to see the command palette and find all available commands.



As you can see, since formatting my notes and adding metadata is a feature I use extensively, I added this hotkey to save time. This can be done by opening the settings menu, going to Options > Hotkeys, and adding a hotkey for the desired command.



In my case I use Command + Alt + L (Mac).

6.3.1.3 YAML Frontmatter Tab

This is the most relevant tab for this entire methodology because it contains the YAML Frontmatter settings.

Here are my current settings.

Linter

Search all settings

YAML

Add Blank Line After YAML

Adds a blank line after the YAML block if it does not end the current file or it is not already followed by at least 1 blank line

Dedupe YAML Array Values

Removes duplicate array values in a case sensitive manner.

Dedupe YAML aliases section

Turns on removing duplicate aliases.

Dedupe YAML tags section

Turns on removing duplicate tags.

Dedupe YAML array sections

Turns on removing duplicate values for regular YAML arrays

YAML Keys to Ignore

A list of YAML keys without the ending colon on their own lines that are not meant to have duplicate values removed from them.

Escape YAML Special Characters

Escapes colons with a space after them (:), single quotes ('), and double quotes (") in YAML.

Try to Escape Single Line Arrays

Tries to escape array values assuming that an array starts with "[", ends with "]", and has items that are delimited by ",".

Force YAML Escape

Escapes the values for the specified YAML keys.

Force YAML Escape on Keys

Uses the YAML escape character on the specified YAML keys separated by a new line character if it is not already escaped. Do not use on YAML arrays.

Format Tags in YAML

Remove Hashtags from tags in the YAML frontmatter, as they make the tags there invalid.

Format YAML Array

Allows for the formatting of regular YAML arrays as either multi-line or single-line and `tags` and `aliases` are allowed to have some Obsidian specific YAML formats. **Note: that single string to single-line goes from a single string entry to a single-line array if more than 1 entry is present. The same is true for single string to multi-line except it becomes a multi-line array.**

Insert YAML Attributes will add any number of custom entries to your YAML Frontmatter. In my case, since I use topics and types to classify my notes, I added those two entries that contain a list of note links.

> You could also add a new entry named something like "Ready" and set it to yes or no. This could be useful for tracking your progress on individual notes while writing them.

Another useful setting is *Move Tags to YAML*. I have not experimented with it yet, but it could prove useful if you tend to use many tags within your notes.

Insert YAML attributes

Inserts the given YAML attributes into the YAML frontmatter. Put each attribute on a single line.

Text to insert

Text to insert into the YAML frontmatter

```
Topics:
Types:
```

Move Tags to YAML

Move all tags to YAML frontmatter of the document.

Body tag operation

What to do with non-ignored tags in the body of the file once they have been moved to the frontmatter

Nothing

Tags to ignore

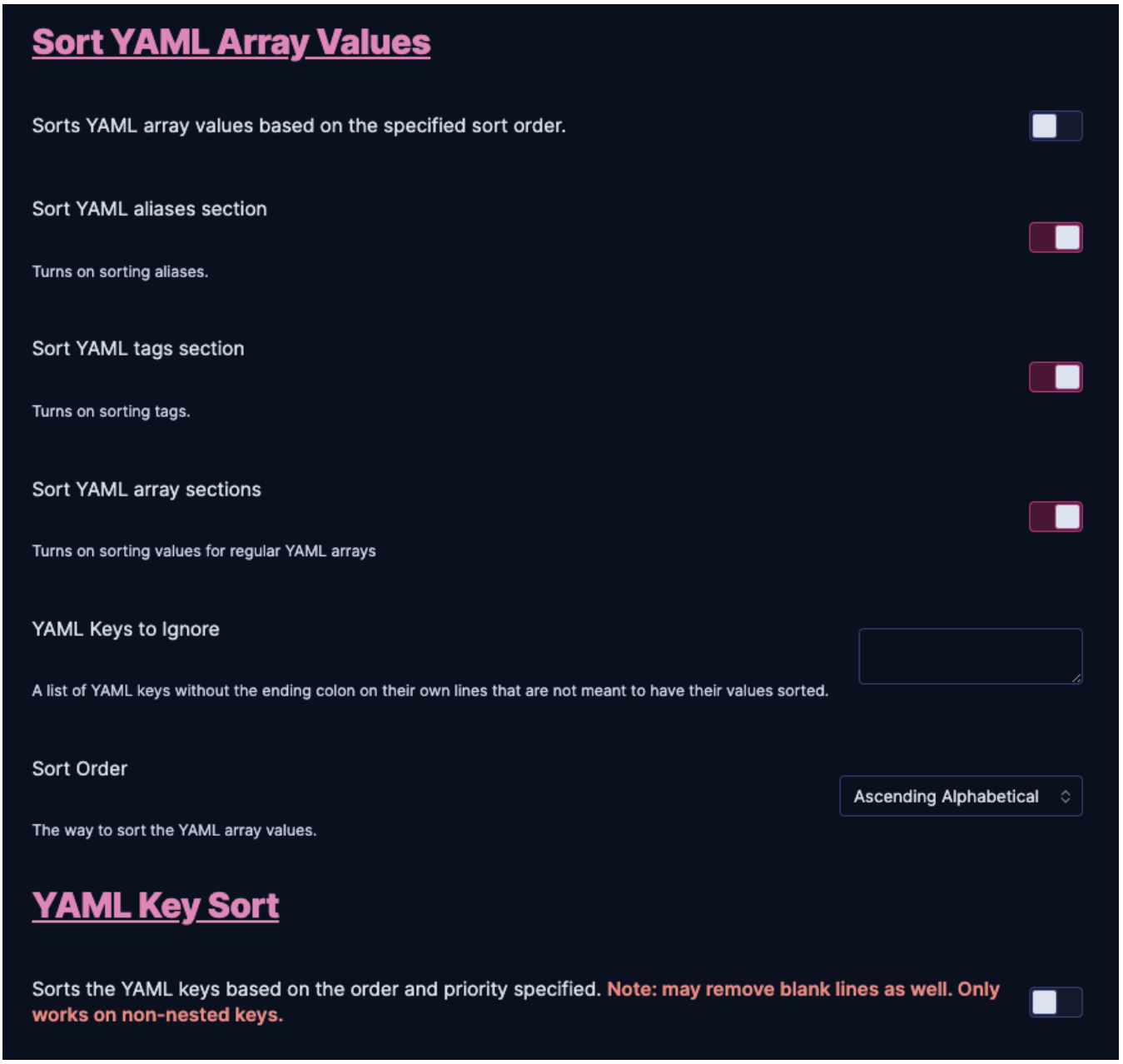
The tags that will not be moved to the tags array or removed from the body content if Remove the hashtag from tags in content body is enabled. Each tag should be on a new line and without the #. **Make sure not to include the hashtag in the tag name.**

Remove YAML Keys

Removes the YAML keys specified

YAML Keys to Remove

The YAML keys to remove from the YAML frontmatter with or without colons



YAML Timestamp adds date metadata at the top of your note. I personally like adding the date created and modified data.

YAML Timestamp

Keep track of the date the file was last edited in the YAML front matter. Gets dates from file metadata.

Date Created

Insert the file creation date

Date Created Key

date created

Which YAML key to use for creation date

Force Date Created Key Value Retention

Reuses the value in the YAML frontmatter for date created instead of the file metadata which is useful for preventing file metadata changes from causing the value to change to a different value.

Date Modified

Insert the date the file was last modified

Date Modified Key

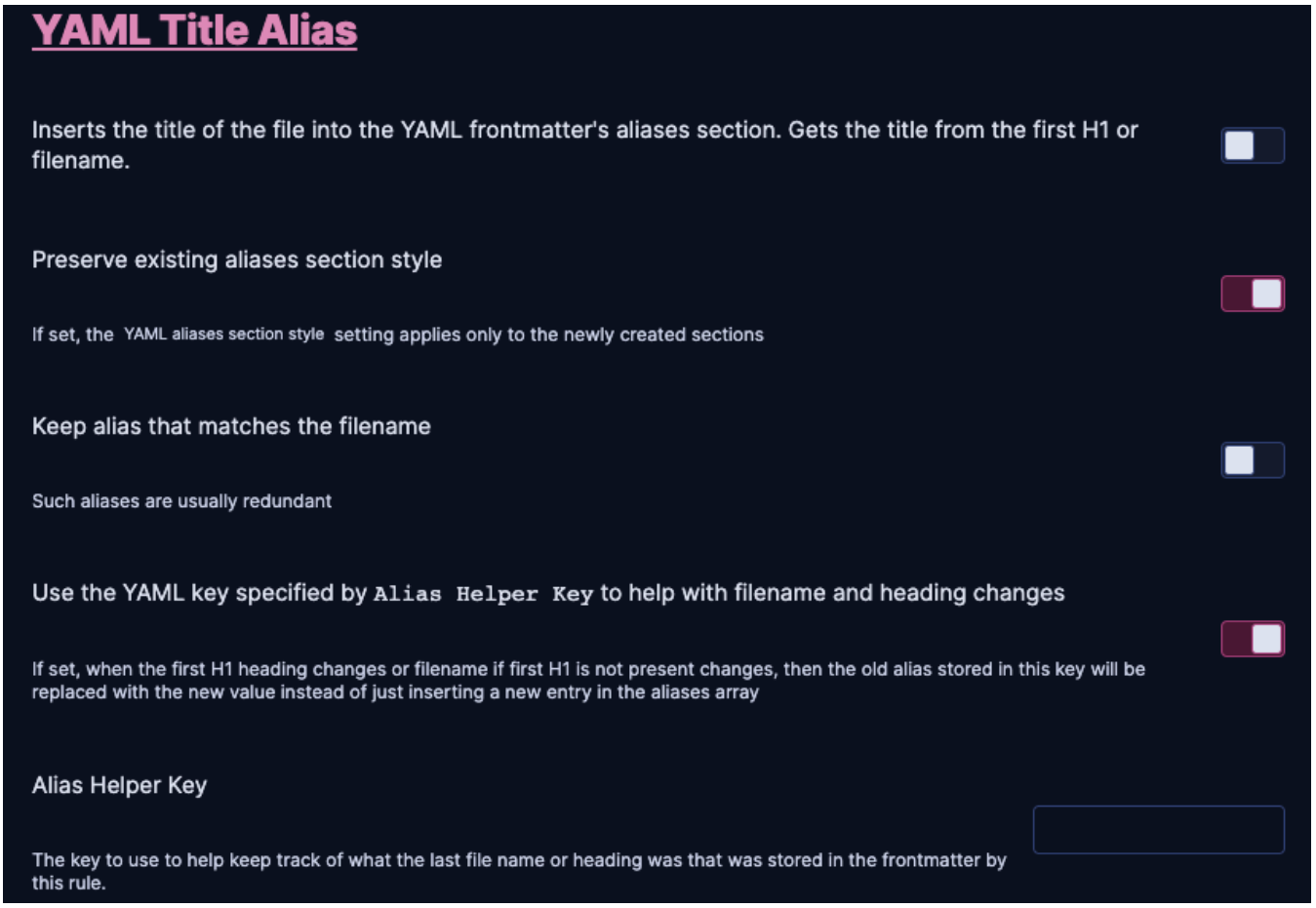
date modified

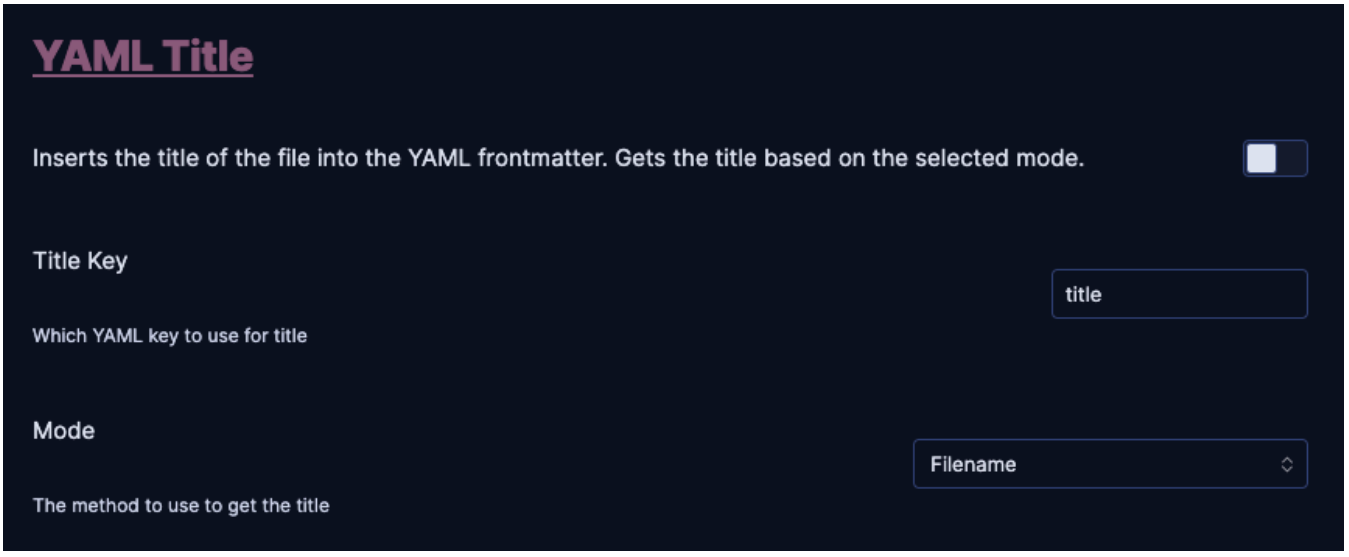
Which YAML key to use for modification date

Format

dddd, MMMM Do YYYY

Moment date format to use (see Moment format options)





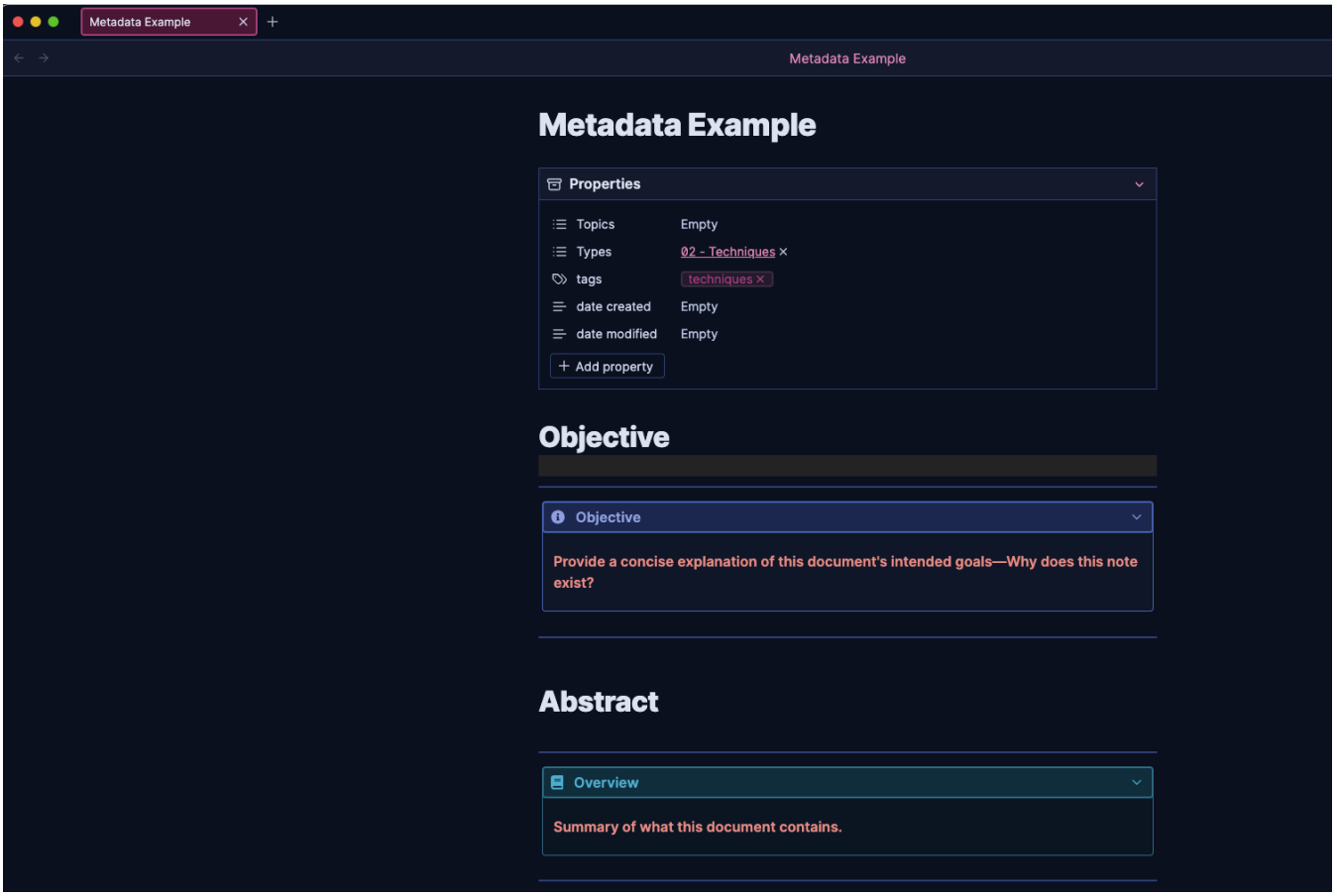
6.3.1.4 Other Settings

That is pretty much it. I suggest going through each setting and tweaking them to your liking. Feel free to copy my settings by checking each individual tab within the settings.

6.3.2 Adding Metadata

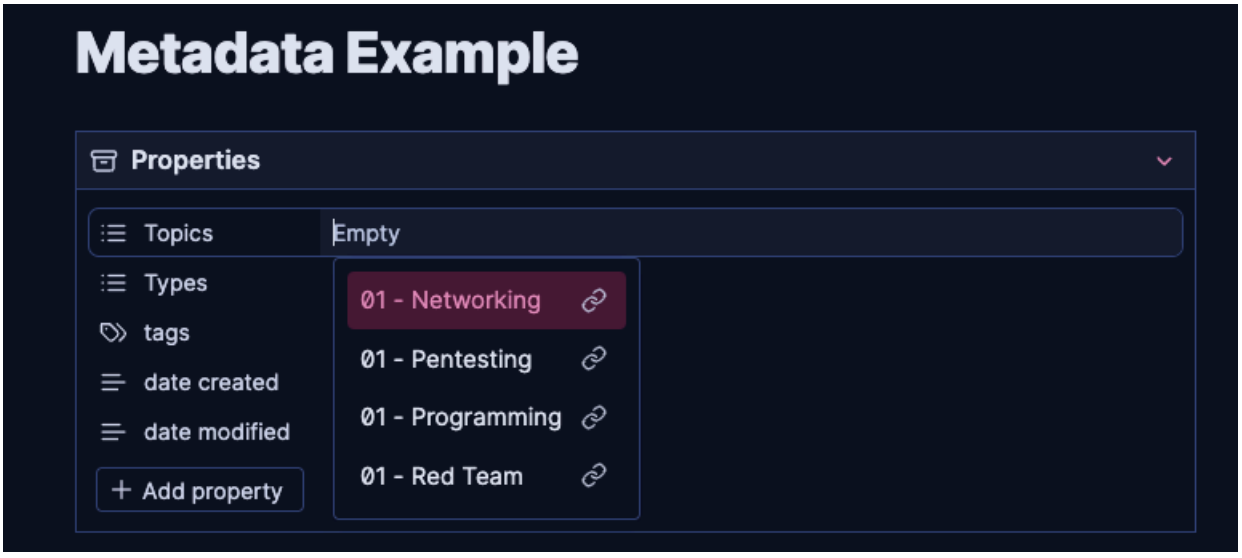
Now, I'll showcase how I add metadata to new notes. This is the first step I take whenever I create a new note. It only takes a few seconds and is essential.

After pressing Command + N (Ctrl + N), the Templater plugin triggers and asks me what type of note I want. I select a new "template" note, and a new note is created with that type of structure.

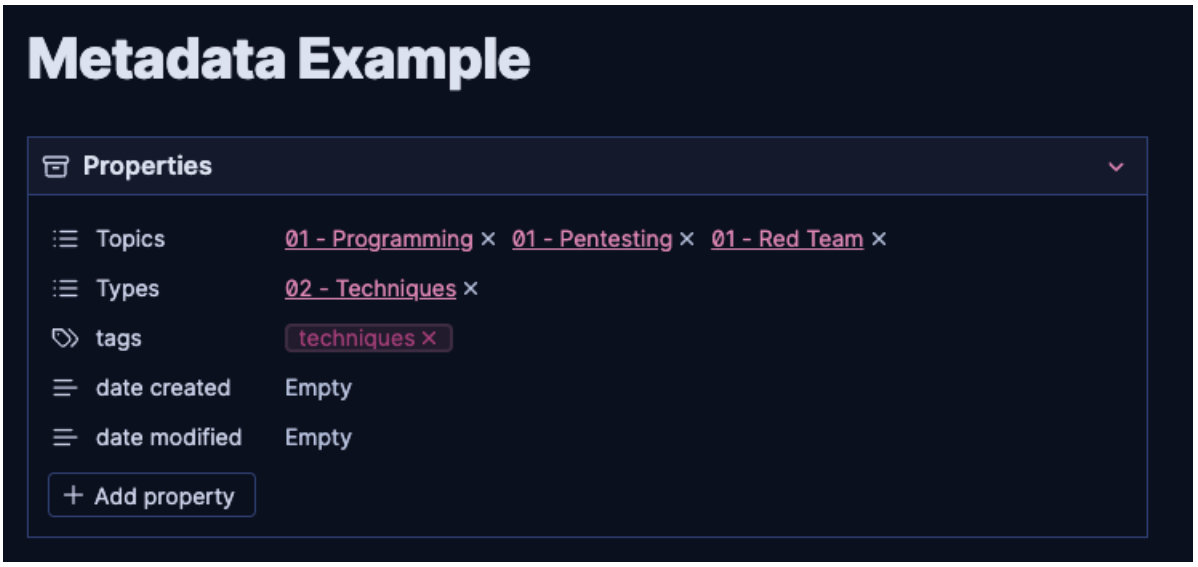


As you can see, our template has already selected the correct note type (02 - Techniques) and added a "techniques" tag. This is by design and saves time compared to entering it manually. For this example, let's say this note will document a new Shellcode Runner technique[2].

The first step would be to add the correct "Topic" for this note. Click on the first entry (Topic), where you can choose or write the name of the topic from our four topics.



In this case, we could add Programming, Pentesting and/or Red Team since this technique is relevant to those three topics.



Then, I would execute the "Lint Current File" command using my hotkey Command + Alt + L.

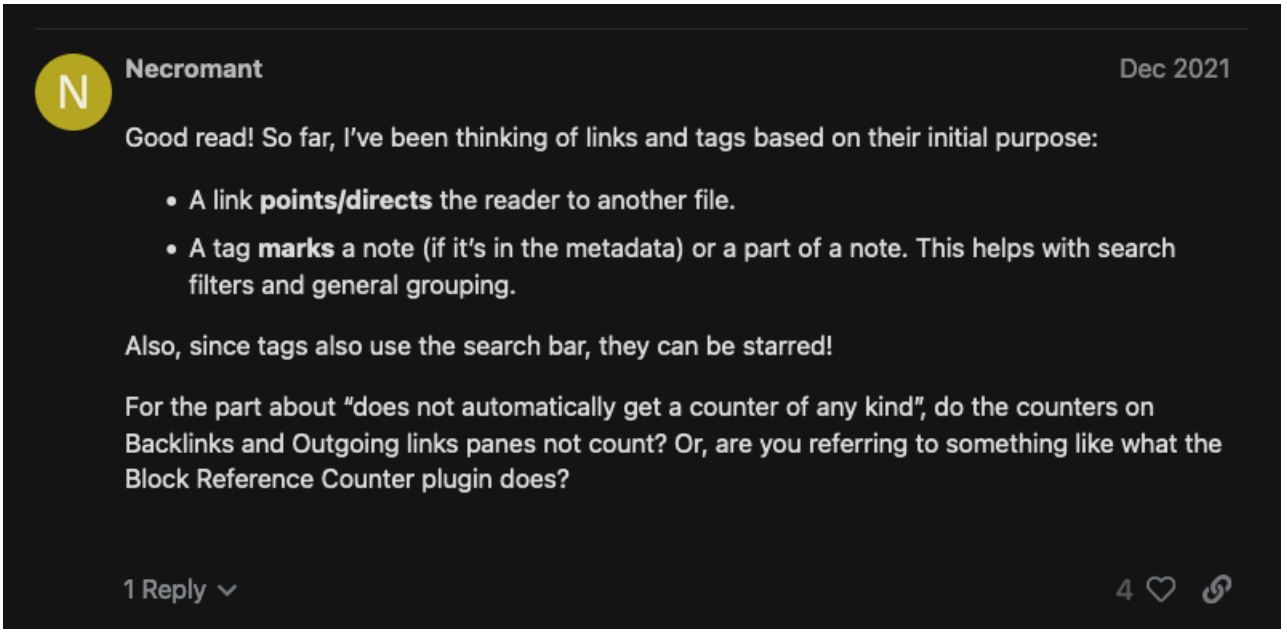
As you can see, the date is now populated automatically using the file's metadata without much hassle.

Finally, I would add any relevant tags to this note. In this case, I would probably add "csharp" and "shellcoderunner", since this tradecraft is written in C# and we are discussing shellcode runners.

I try not to use too many tags as it can become convoluted and hard to track.

Some people prefer using tags exclusively instead of links, but I have stuck with my current configuration since I started learning and continue to use it today. There are different opinions on this topic, so I recommend reading about it and choosing your own methodology to stick with.[3]

I use tags and links in the manner described by this user. I recommend using at least tags because Dataview uses them to create custom lists.

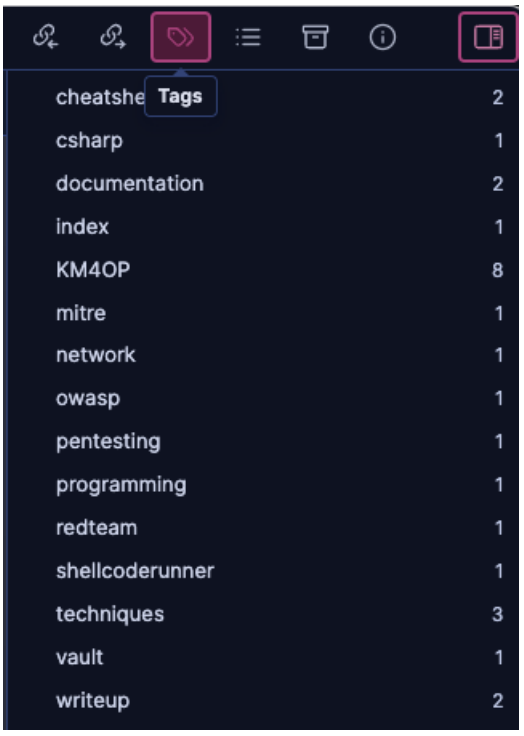


6.3.3 Managing Tags with Tag Wrangler

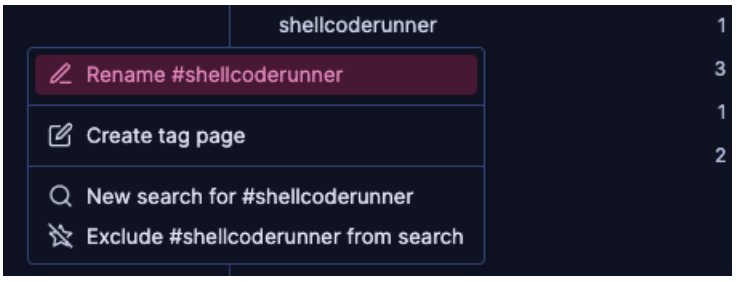
Now that you know how to add metadata to your notes, it's important to learn how to manage your tags using TagWrangler.

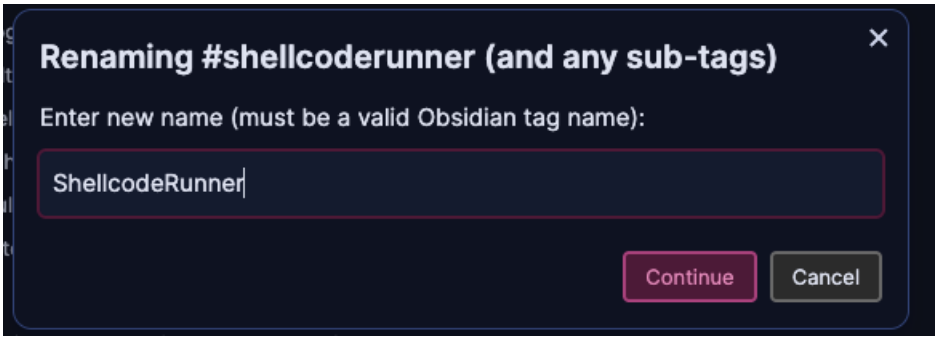
After writing several notes using the "shellcoderunner" tag, you might want to change the tag's name to make it more readable, such as "ShellcodeRunner".

To do this, go to the right dashboard and click on "Tags".



Right-click your desired tag and click on "rename tag".

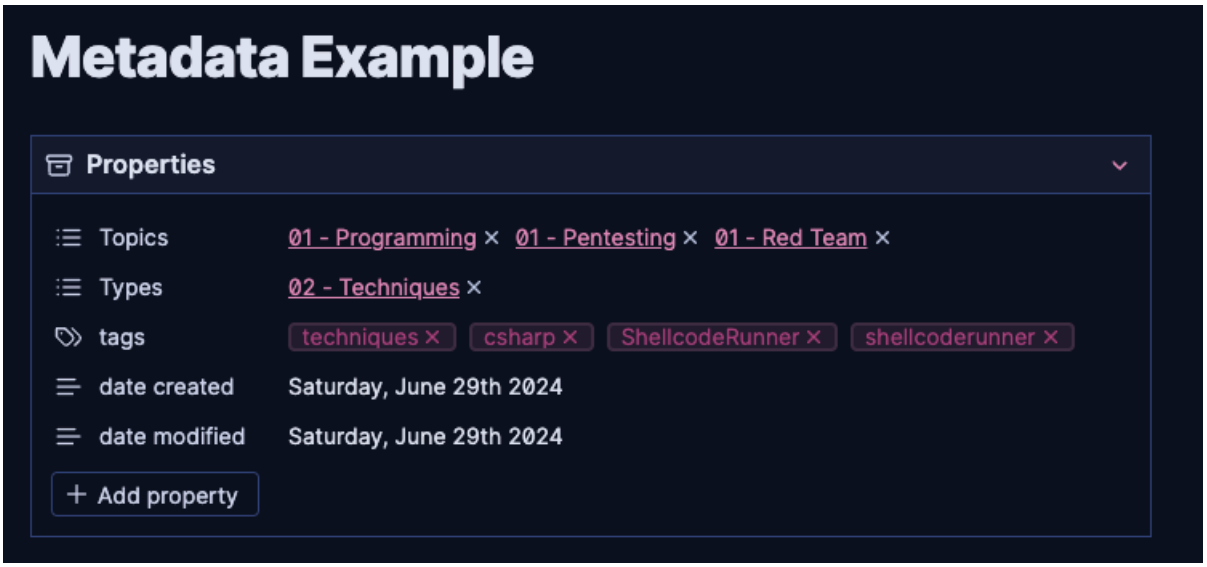




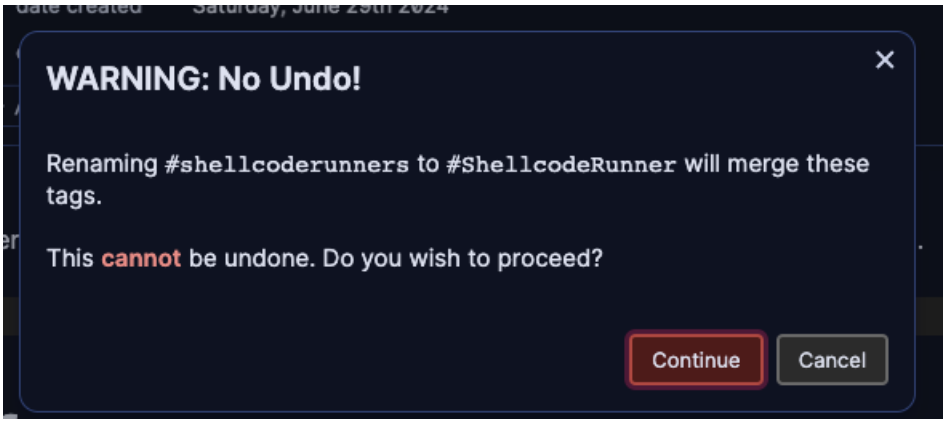
Click "Continue" and the tag will be renamed in all notes where it's used.

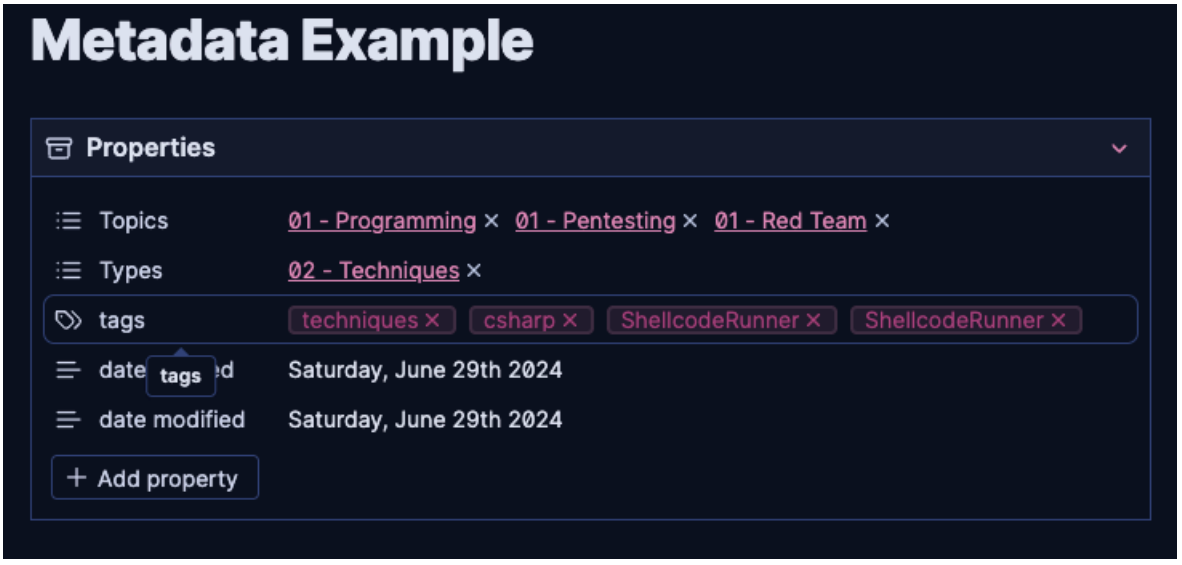
You can also use this feature to merge tags.

For example, if you added a new tag called "shellcoderunners" but later realized that a "ShellcodeRunner" tag already existed, you can merge them using TagWrangler.



To resolve this issue, simply rename the incorrect tag to match the correct one. Accept the prompt, and the tags will be merged.



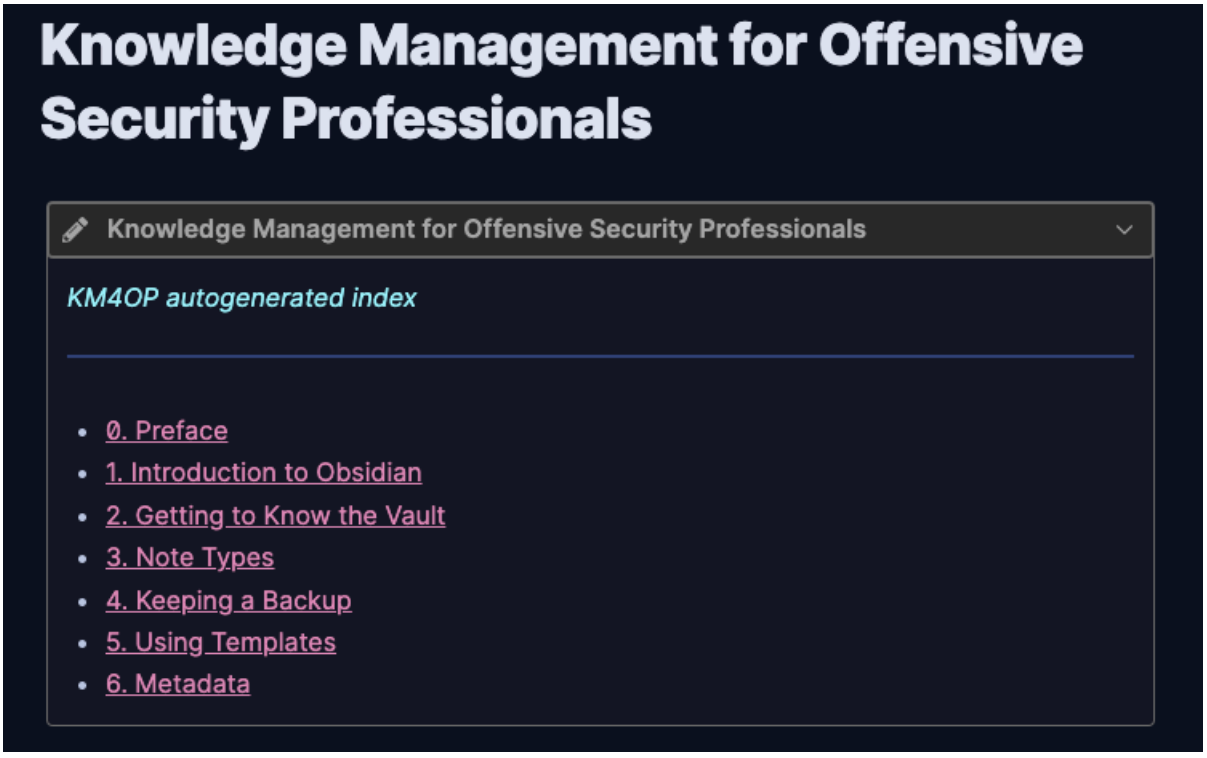


6.3.4 Using Dataview

Now that you know how to use tags and metadata, we can leverage them to create custom lists and tables using Dataview. I recommend referencing the official documentation.

For example, in the Knowledge Management for Offensive Security Professionals note, you will find an admonition block containing all notes related to this course. This is a dynamic list that automatically populates

with every note tagged with "KM4OP".



This is the source code for the block.

```
```ad-note
title: Knowledge Management for Offensive Security Professionals

*KM4OP autogenerated index*
___
~~~dataview
LIST
FROM #KM4OP
WHERE file.name != "Knowledge Management for Offensive Security Professionals"
SORT file.name asc
~~~

```
```

As you can see, I first created an Admonition Block using three backticks and the "ad-note" keyword. I added some text inside, and then I created a new code block using three wave dashes (you could use six backticks as well) and the "dataview" keyword.

To configure this list, I:

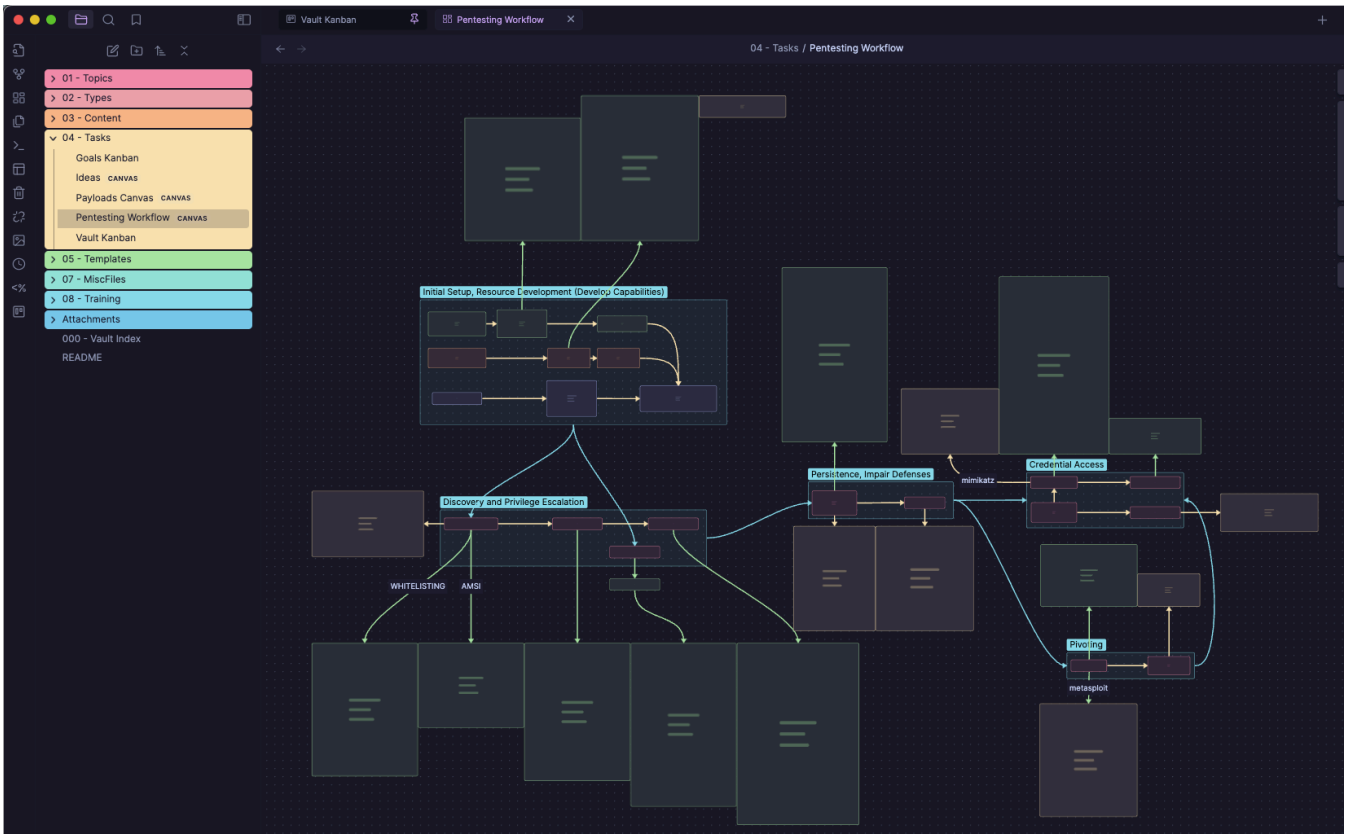
- Used the **LIST** keyword to create a new list.
- Used the **FROM** keyword to select notes with a specific tag.
- Used the **WHERE** keyword to exclude the current note by its file name.
- Used the **SORT** keyword to keep the list sorted in ascending order.

This syntax resembles SQL, and I was able to create this list by referencing the documentation.

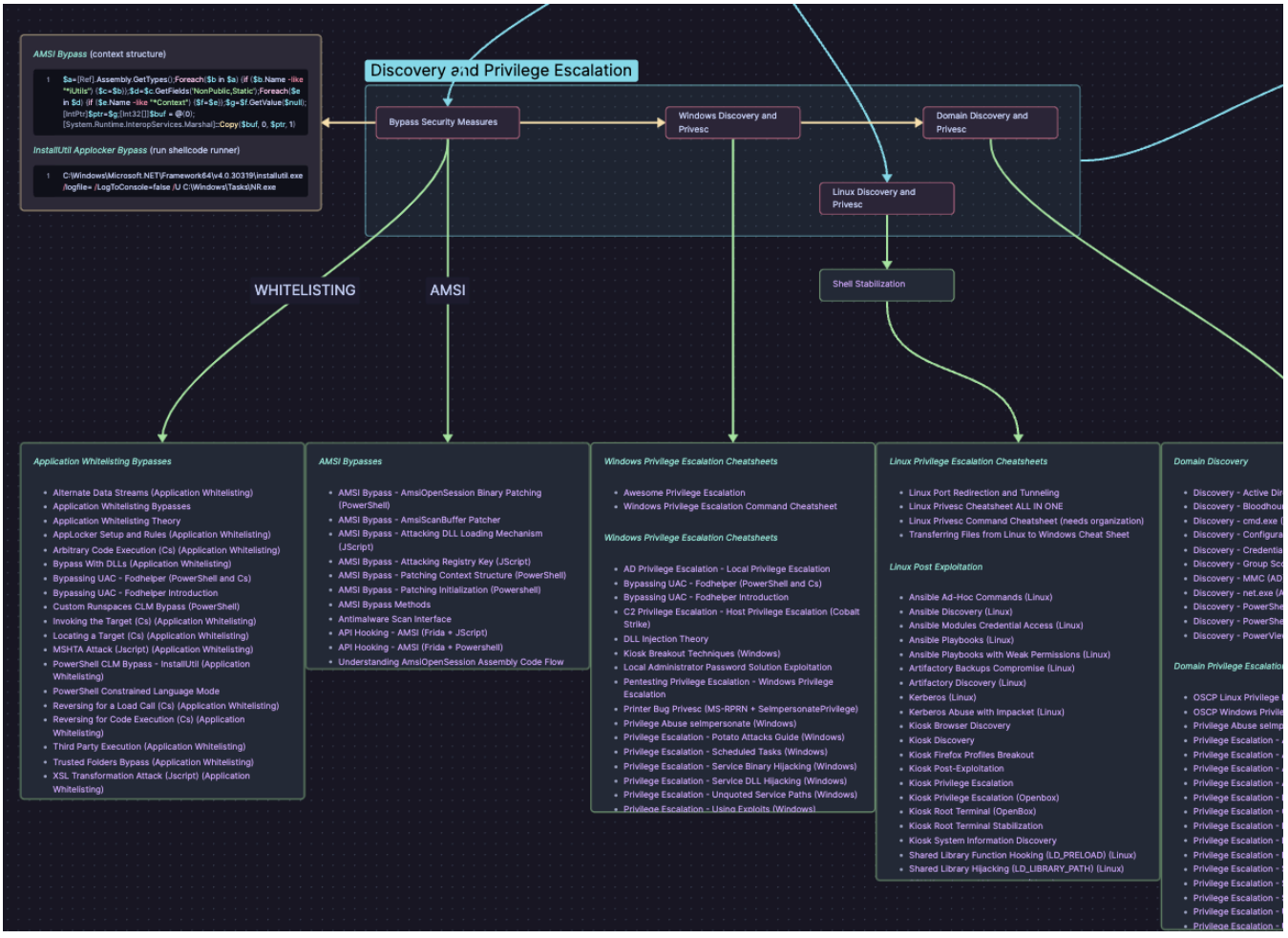
6.3.5 Example Diagram

I will now provide an example from my personal vault to demonstrate this methodology in action.

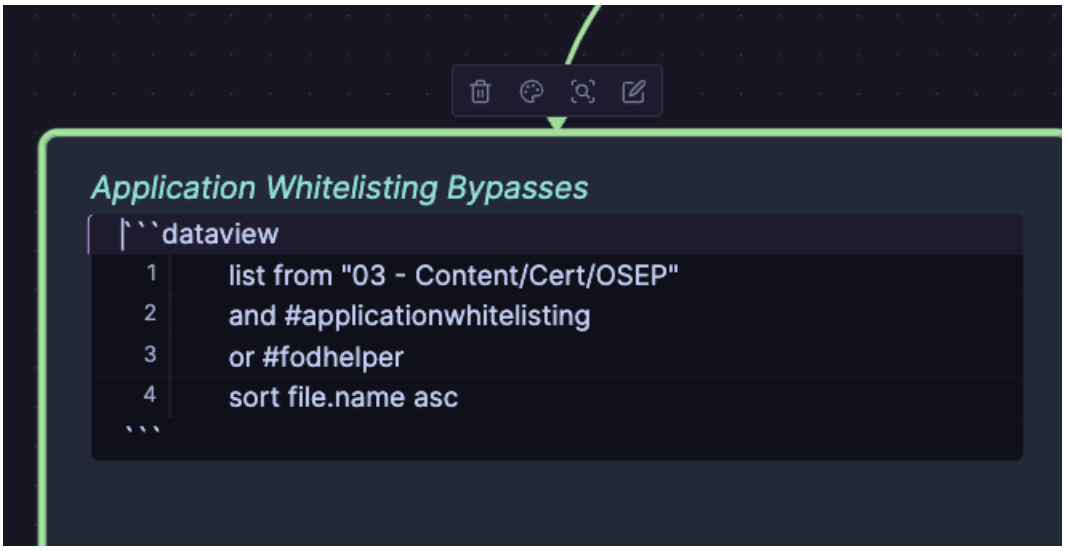
In my Tasks folder, I have a canvas note which is a Pentesting Workflow diagram that I used while preparing for my most recent certification, OSEP.



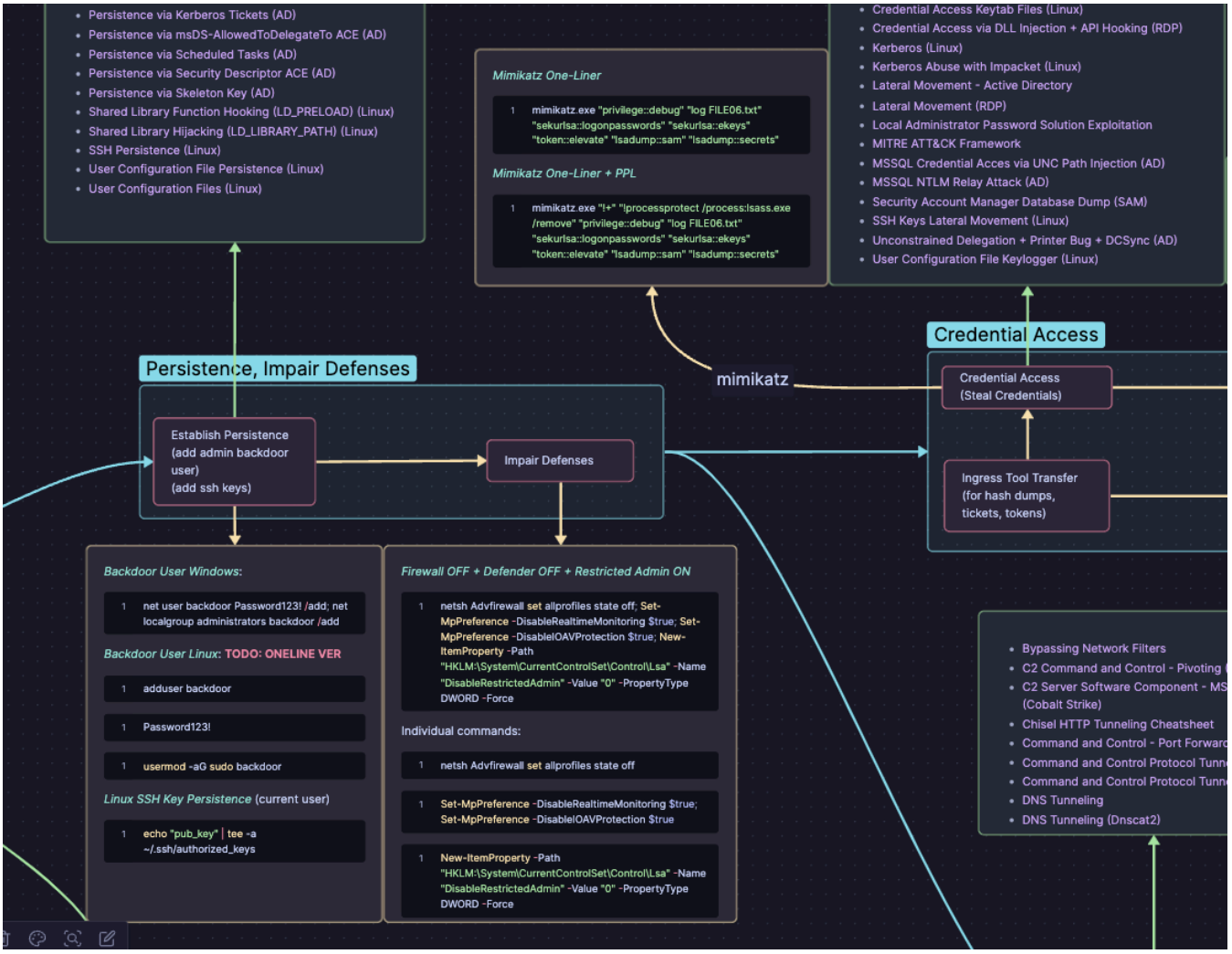
This diagram contains various steps, code blocks, and notes that I would follow each time I hacked a machine.



Creating each of these lists manually would have taken an insane amount of time and effort. Instead, I used admonition to leverage the metadata already present on my notes.



Using metadata enabled me to create a comprehensive step-by-step hacking methodology diagram.



References

1. YAML Frontmatter - Fork My Brain ↩
2. Shellcode Runners | Pentester's Promiscuous Notebook ↩
3. https://forum.obsidian.md/t/a-guide-on-links-vs-tags-in-obsidian/28231 ↩