

Part 5 Examples, Instructions and Exercises

Step by step guide for creating a virtual machine with Plaso, Nginx, OpenSearch and OpenSearch Dashboards followed by exercises for visualization of log2timeline data using OpenSearch Dashboards.

Python for Data Analysis introduction

Master of Advanced Studies in Digital Forensics & Cyber Investigation

Data Analytics and Visualization for Digital Forensics

(c) Hans Henseler, 2024

1. Introduction

We are using VirtualBox from Oracle to work with a virtual machine that runs Plaso tools, OpenSearch, and OpenSearch Dashboards. You will need to download and install Oracle Virtual Box. Here are instructions for installing on Windows:

<https://download.virtualbox.org/virtualbox/7.0.20/VirtualBox-7.0.20-163906-Win.exe>

After finishing the installation you should also install VirtualBox extension pack on your windows host. This executable file can be downloaded here:

https://download.virtualbox.org/virtualbox/7.0.20/Oracle_VM_VirtualBox_Extension_Pack-7.0.20.vbox-extpack

You can try step 2 first. If that works you continue with step 3. If step 2 doesn't work (possibly because of hardware differences or network issues) I recommend to install a fresh VM. This is described in a separate appendix "Part 5 appendix Installing Plaso OpenSearch and OpenSearch Dashboards.pdf".

***** note To perform this exercise you need at least 10GB of free space on your harddrive.*****

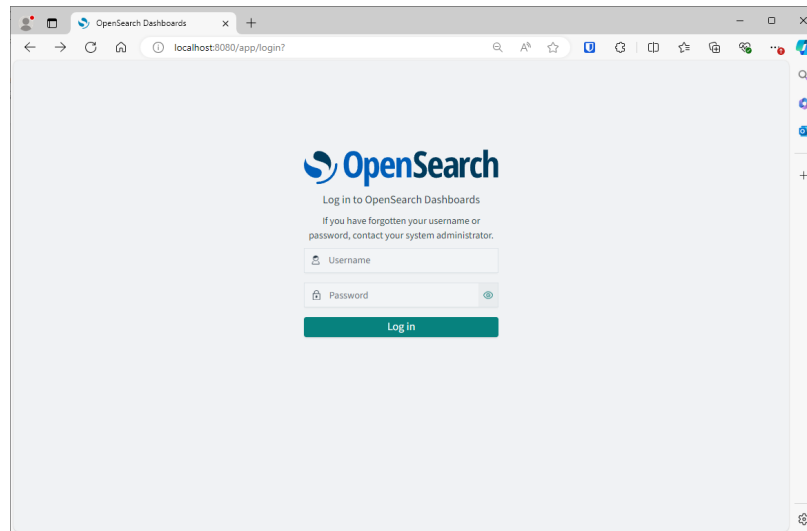
If you are using VirtualBox on a Mac or other OS please check the Oracle website for instructions.

2. Create VM based on the davday3.ova appliance

With these steps you can create the davday3 vm based on an appliance file (davday3.ova):

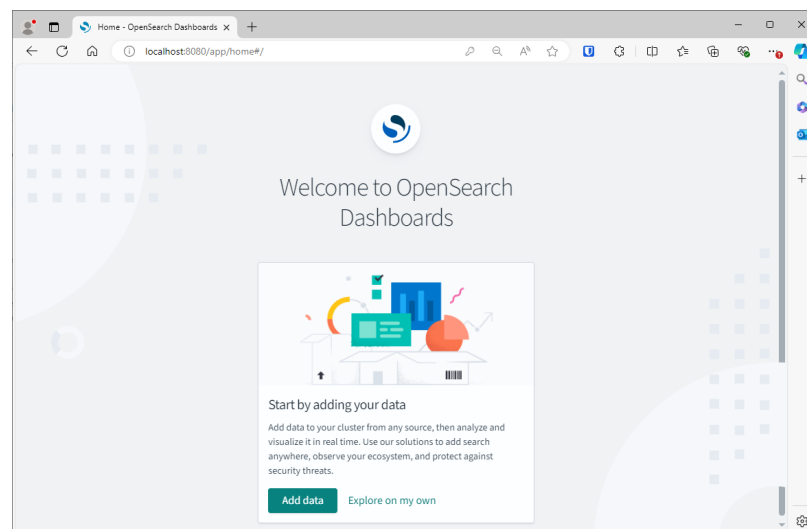
1. Download the davday3_v2.ova appliance (4.5GB). This file can be downloaded via a two-step process. First access the following link:
<https://drive.google.com/drive/folders/1TTJGxGCFBdtYGCH9crdATFlanOriHEzj?usp=sharing>
2. You have no permission to access this folder. If you Ask Permission, the owner (me) will receive an email and I will grant you permission. **Please do this on Tuesday** so you have enough time to download the .ova file. Once you have received permission, go ahead and download the file.
3. Start virtualbox (Download and install virtual box if not already done so)
4. Under the file menu select import appliance. Select the davday3_v2.ova file. Click next. Check location. Click import. Change #cpu and #ram if you do not have enough (default 4 cpu and 16GB RAM). This will take 1-5 minutes. The resulting Ubuntu2204Opensearch VDI disk is 9.2GB.

5. Start the davday3 vm and login with user plaso and password plaso
6. start a web browser on you windows (host) and go to the following url:
<http://127.0.0.1:8080/>. You should see this:



If you get a message the “Opendashboard is not ready”, you may have to wait a few minutes for openserach to get started.

7. login with user admin and password P?ssw0rd1. After successful login you should see:



If you can login that means the VM is working as expected and you can continue with the exercises in section 3.

3. Exercises

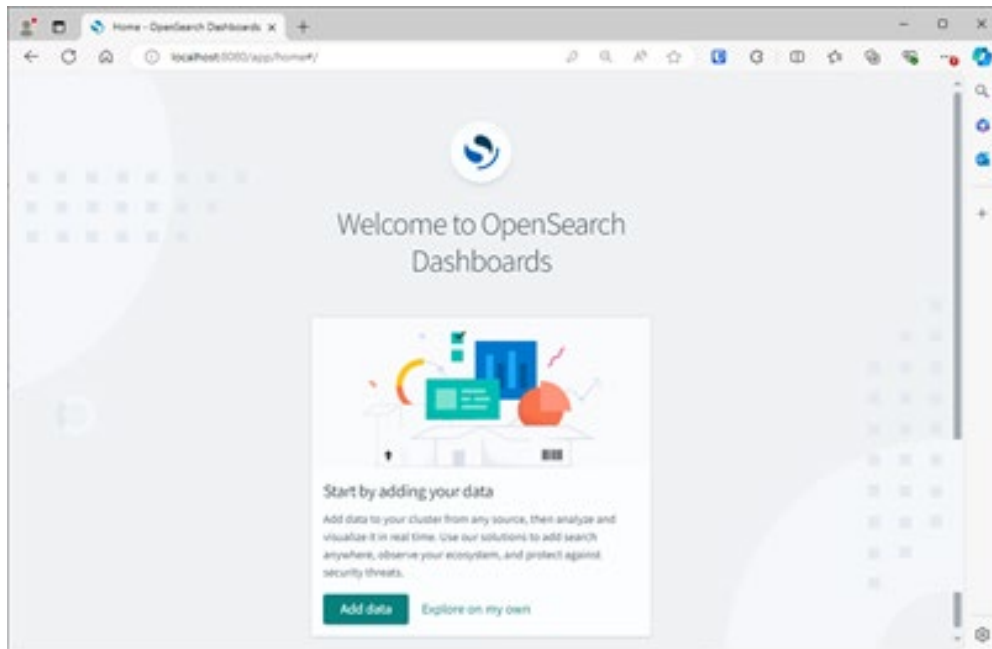
When you get to this point you should have a working Openserch Dashboard VM. Let’s start loading some data. We use psort.py to ingest data to OpenSearch similar to the Colab Notebook in part 4 (day 2 afternoon). In order to use psort.py we need to get a plaso file on the VM. This can be one of the plaso files you generated in part 3 on day 2 in the morning.

3.2 Access the OpenSearch Dashboard webinterface

Open a browser at the VM host (your computer) and enter:

<http://127.0.0.1:8080>

The browser will prompt you for credentials. You can login with user admin and password P?ssw0rd!. There is no data yet in the system and you should see the following:



Before we Add Data we will upload our own data in OpenSearch. This is described in exercise 3.2.

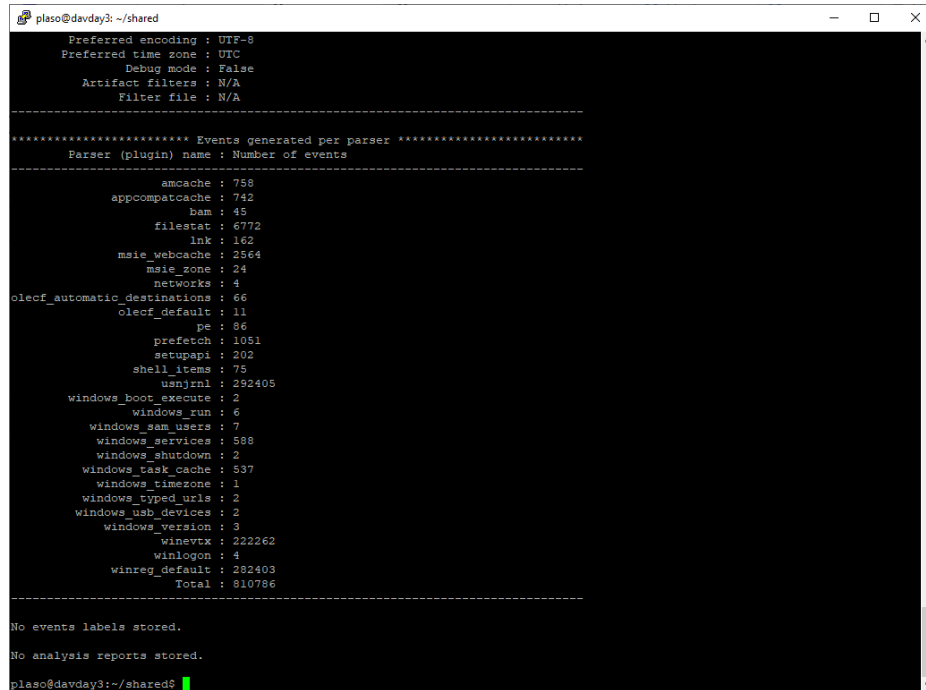
3.3 Set up a shared folder with the host

If you want to use data is on your windows machine you need access that data. To get files on from your windows machine you can share a folder and mount it in your VM. Here is an example how to mount a folder 'shared' on your host computer in the VM.

1. Open VirtualBox
2. Right-click your VM, then click Settings
3. Go to Shared Folders section
4. Add a new shared folder
5. On Add Share prompt, select the Folder Path in your host that you want to be accessible inside your VM. Create a new folder with the name 'day3_data' and select the folder.
6. The Folder Name field automatically is filled with 'day3_data'
7. In mount point enter: ~/shared
8. Uncheck Read-only, check Auto-mount and Make Permanent and click OK
9. Login to the virtual machine with user plaso and password plaso. Rember to use your CMD window with the ssh command so you can copy and paste commands from this document to the shell prompt: `ssh -p 2222 plaso@127.0.0.1`
10. There should be a folder 'shared' in the home directory of plaso. You can check this with the 'ls' command. If it is not there then create it with: `mkdir shared`
11. Mount the shared folder from the host to your ~/shared directory:
`sudo mount -t vboxsf day3_data ~/shared`
12. The host folder should now be accessible inside the VM.
13. `cd ~/shared`
14. At the command prompt list the contents of this folder with 'ls -l'. It should be empty.

15. On your host computer: download the 'mus2019ctf.plaso' which should be in your Google drive from the Colab exercise in part 3 (Tuesday morning) or on your computer. Move mus2019ctf.plaso to the 'day3_data' folder.
16. The 'ls -l' command should know list the 'mus2019ctf.plaso' file with size is app. 493MB.
17. Check with pinfo.py the contents of the 'mus2019ctf.plaso' file. Just check the sessions report:

```
pinfo.py --sections "events,reports,sessions" mus2019ctf.plaso
```



```

plaso@day3: ~/shared
Preferred encoding : UTF-8
Preferred time zone : UTC
Debug mode : False
Artifact filters : N/A
Filter file : N/A

***** Events generated per parser *****
Parser (plugin) name : Number of events
-----
amcache : 758
appcompatcache : 742
bam : 45
filesrat : 6772
lnk : 162
msie_webcache : 2564
msie_zone : 24
networks : 4
olecf_automatic_destinations : 66
olecf_default : 11
pe : 86
prefetch : 1051
setupapi : 202
shell_items : 75
usnjrnl : 292405
windows_boot_execute : 2
windows_run : 6
windows_sam_users : 7
windows_services : 588
windows_shutdown : 2
windows_task_cache : 537
windows_timezone : 1
windows_typed_urls : 2
windows_usb_devices : 2
windows_version : 3
winetx : 222262
winlogon : 4
winreg_default : 282403
Total : 810786

No events labels stored.
No analysis reports stored.
plaso@day3:~/shared$

```

3.4 Extract events for the plaso file and output to Elasticsearch

Next step is the export of events from the plaso file into elasticsearch. Execute the following command:

```
psort.py -o opensearch --server localhost --port 9200 --opensearch-user admin --opensearch-password P?ssw0rd1 --opensearch_mappings /usr/share/plaso/opensearch.mappings --use_ssl --ca_certificates_file_path ~/root-ca.pem --index_name mus2019ctf mus2019ctf.plaso
```

```
plaso@day3: ~/shared
plaso - psort version 20240308

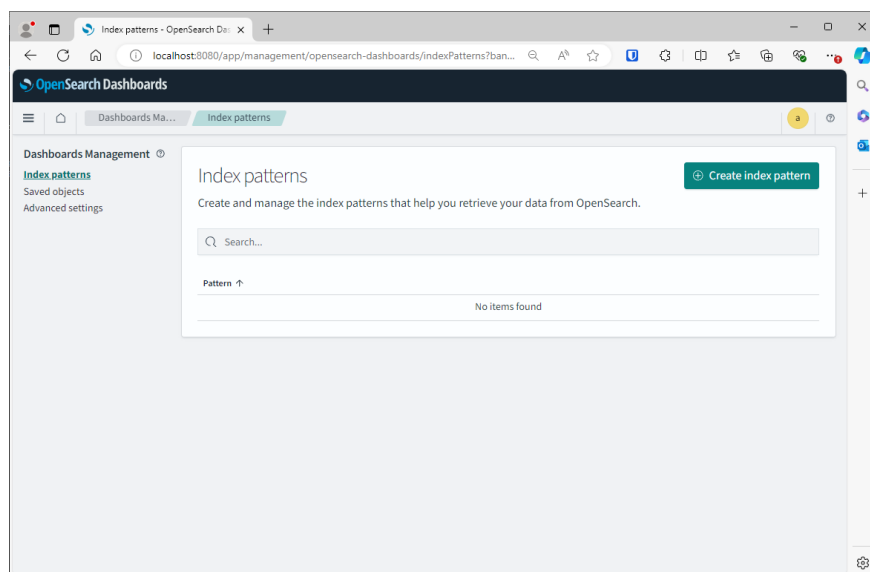
Storage file      : mus2019ctf.plaso
Processing time    : 00:07:06

Events:
  Filtered      0
  In time slice 0
  Duplicates    5097
  MACB grouped  802826
  Total         810786

Identifier  PID  Status  Memory  Events  Tags  Reports
Main       2176 completed 254.8 MiB 810786 (0) 0 (0) 0 (0)

Processing completed.
plaso@day3:~/shared$
```

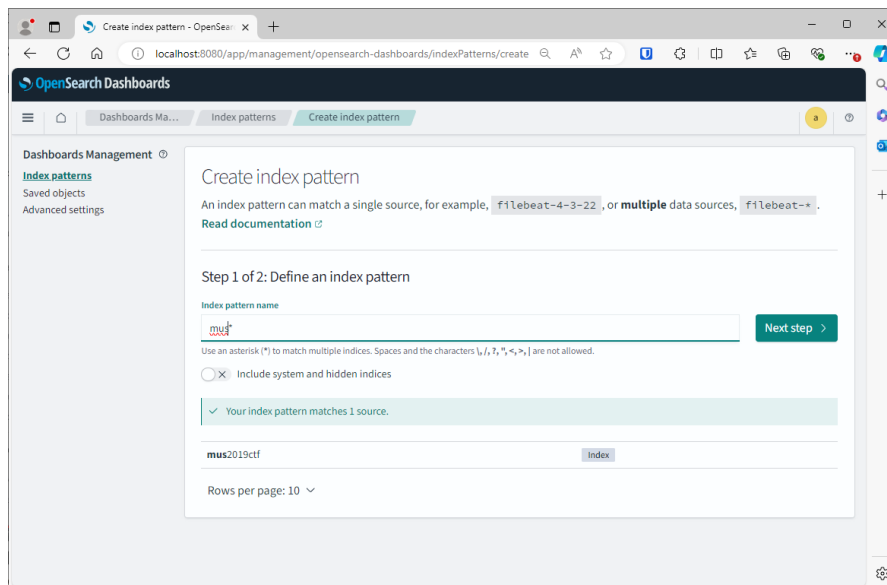
Psort.py starts running and is displaying statistics. Depending on the host computer this can take some time (5-10 minutes for 810.786 events). While Psort.py is busy, go back in OpenSearch Dashboard and select Discover under the OpenSearch Dashboards menu. You need to create an Index pattern.



3.5 Create an index pattern

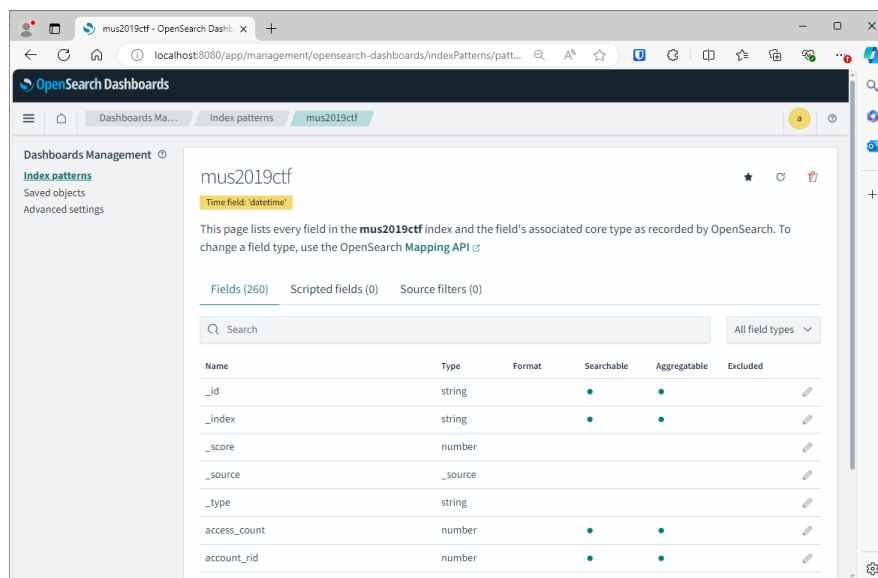
Create an Index Pattern by clicking on the green button:





Creating an index pattern is a 2 step process:

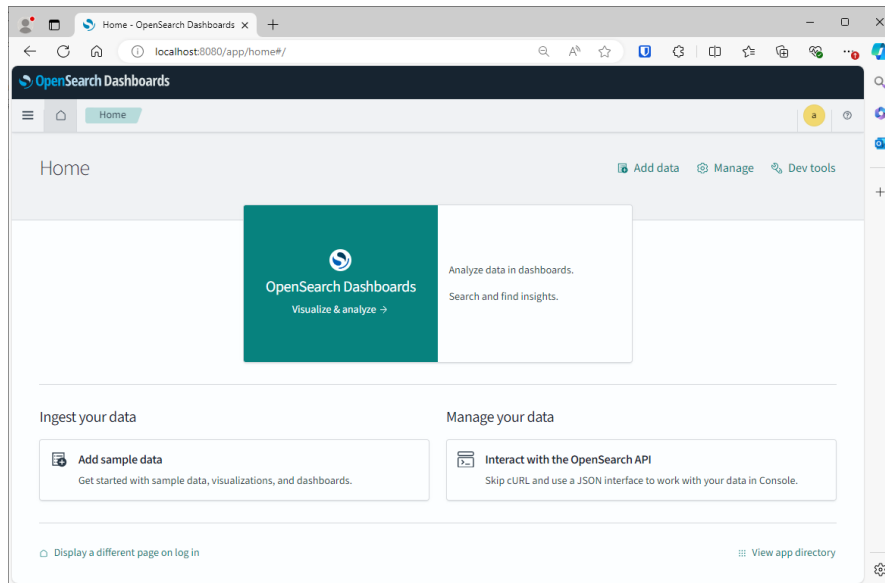
1. Define an index pattern
 - a. Type the complete index name (so that would be mus2019ctf) and select next.
2. After selecting the index you have to configure settings
 - a. Select the Time field. This is easy since there is only field available: datetime. Then select Create index pattern



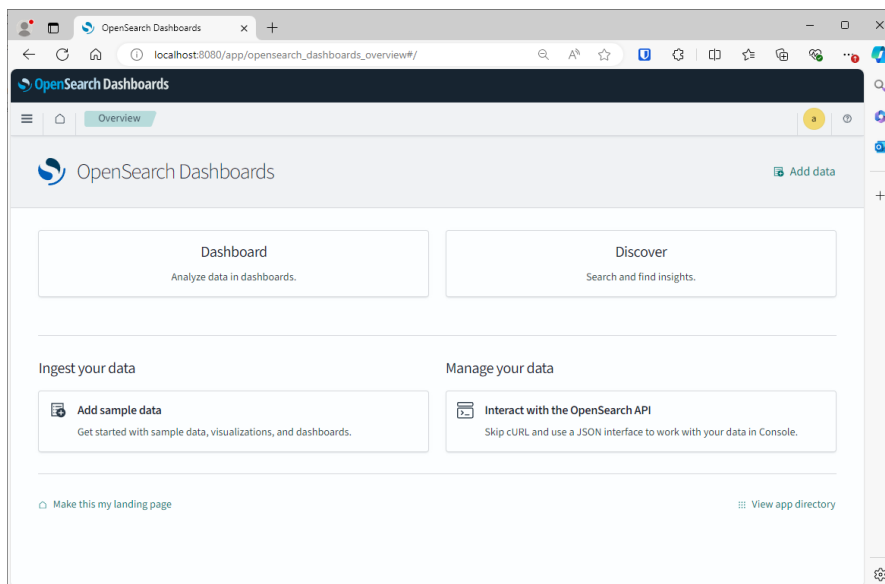
You can edit Field properties. Note that there are 260 fields in our index! For now we won't change anything. We will leave the Management section and go back to home (button right to three horizontal bars in the top left).

3.6 Discover data with OpenSearch dashboard

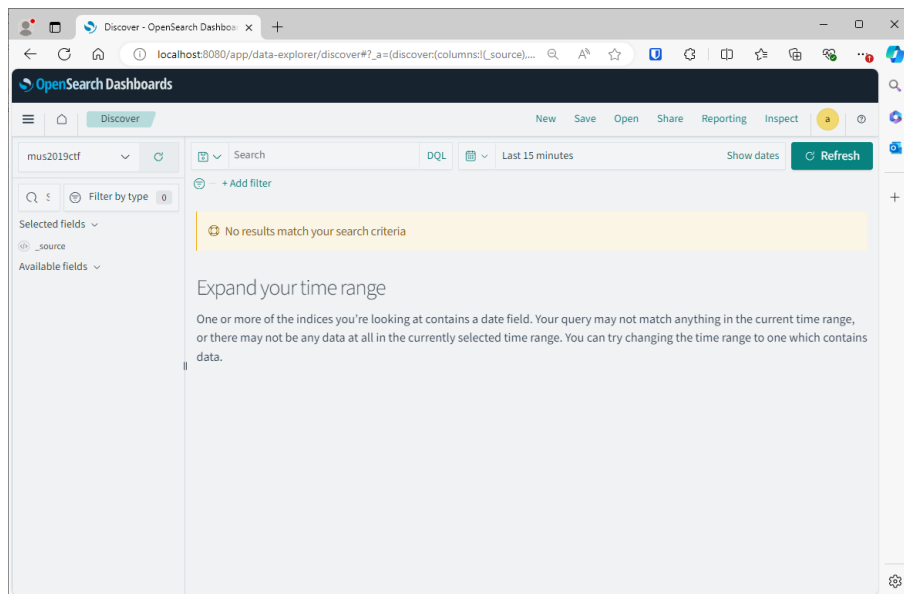
From the Home page dashboard (<http://127.0.0.1:8080/app/home#/>) we select OpenSearch Dashboard/Discover.



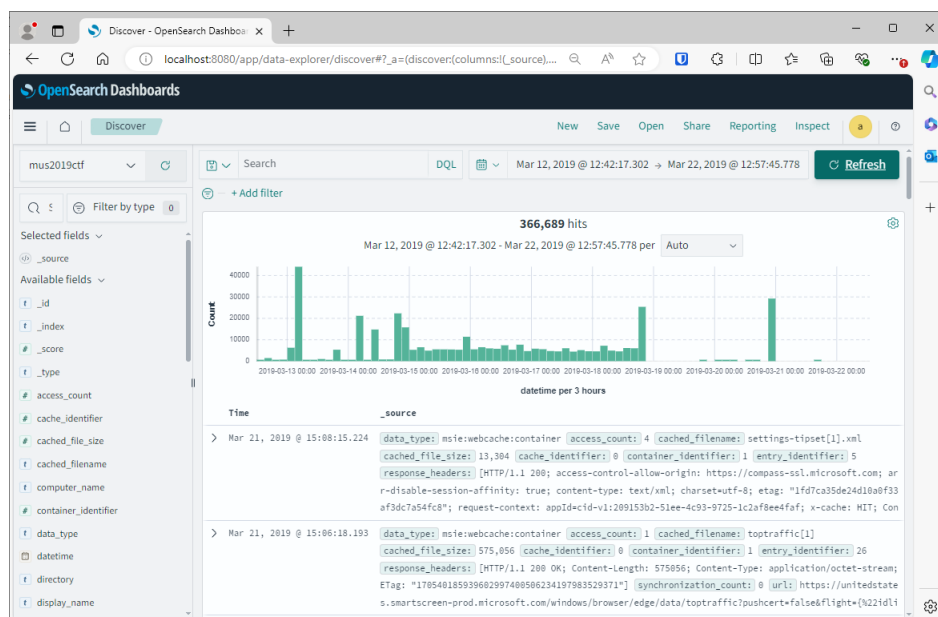
Then select OpenSearch Dashboards



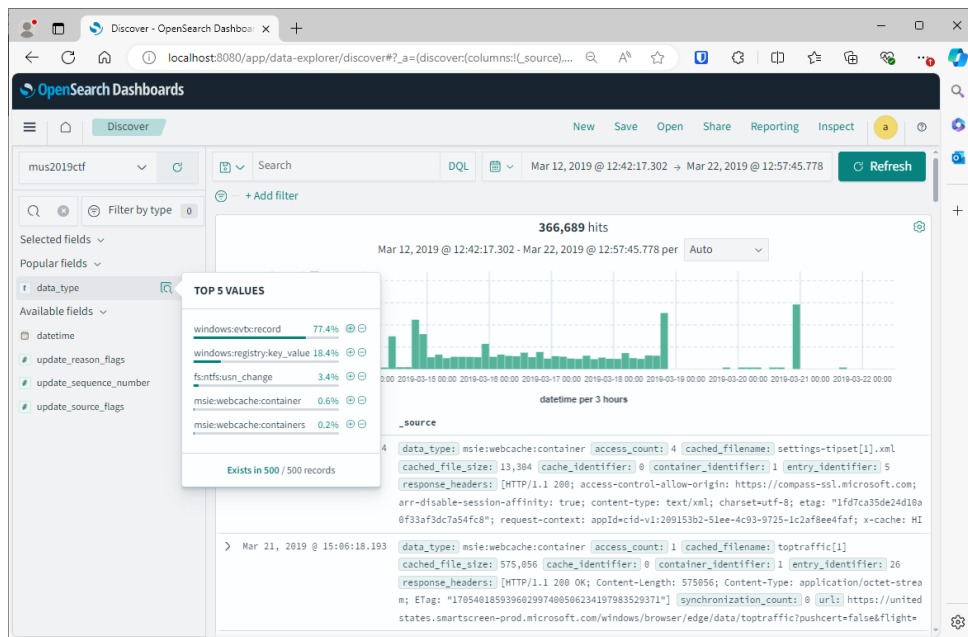
By default OpenSearch Dashboards is filtering for recent events (Last 15 minutes) which are not present in our data.



You can change the date filter in the upper right corner with 'Show dates'. If you click on the period you can set absolute dates. Let's pick the 12-03-2019 until 22-03-2019 period. You should now see that the windows becomes populated with data. If not you may have to wait or select an earlier date.



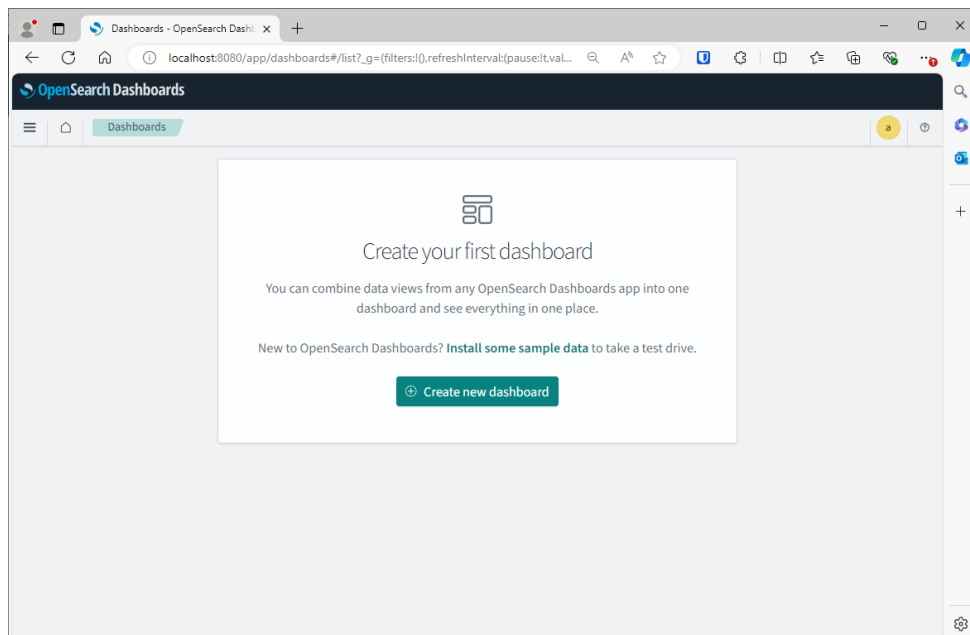
On the left side you see there are 47 different fields available in this time frame. You can search for names. Type in date and select the date_type field. There is a pop-up showing you the values for this field.



This works so much easier than the Elasticsearch json query syntax that we tried on Day 2 😊 Click on the – sign to eliminate a type from our dashboard. You will see it is added to the top left at NOT data_type: windows:evtx:record

3.7 Visualize data with OpenSearch Dashboards

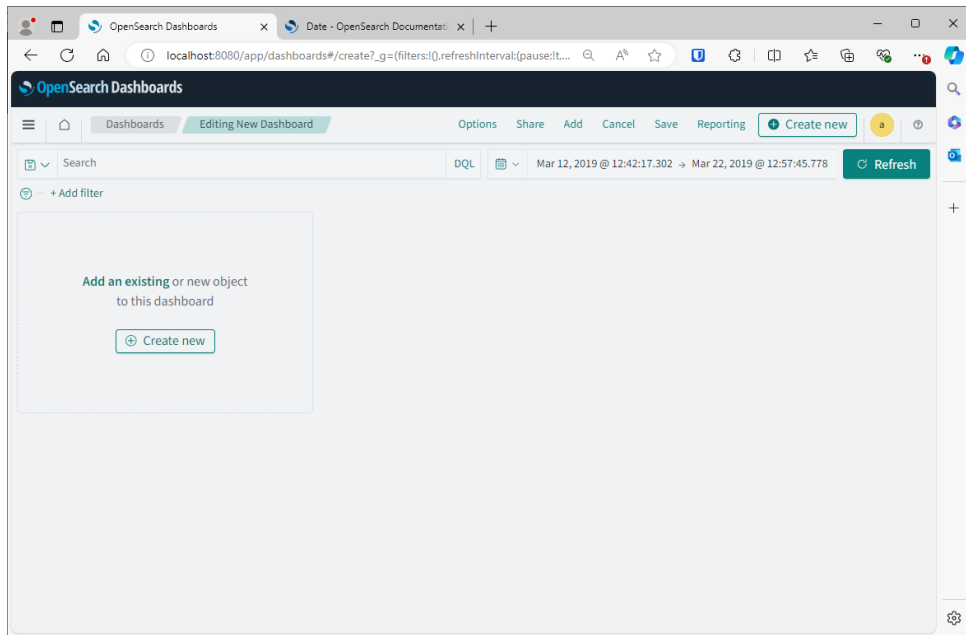
Go back to Homes and choose OpenSearch Dashboards. Then select Dashboard (in stead of Discover). OpenSearch Dashboard will prompt you to create your first dashboard:



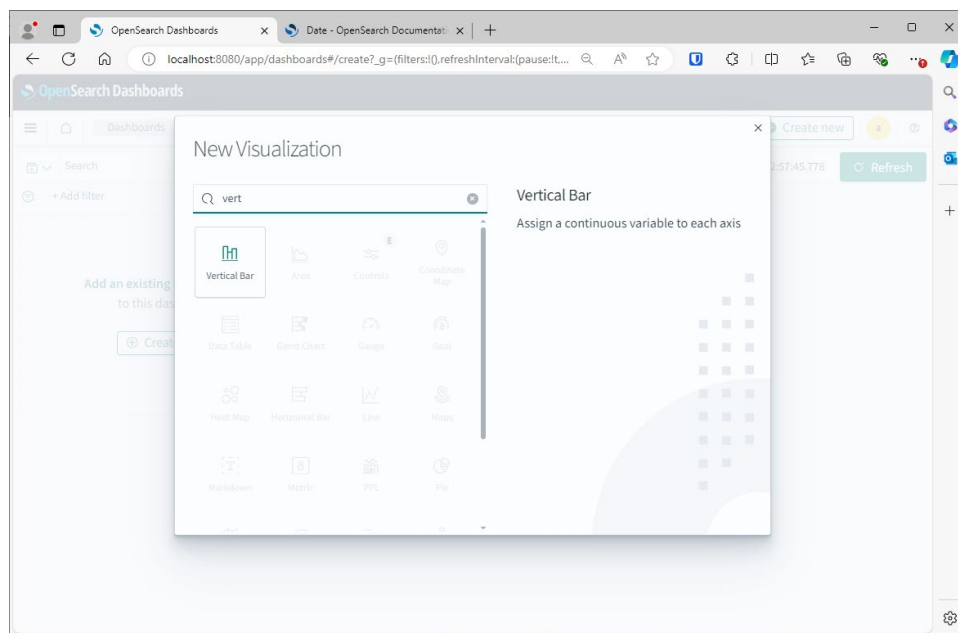
Now try to create some visualisations. Create a dashboard with:

A: Vertical bar chart

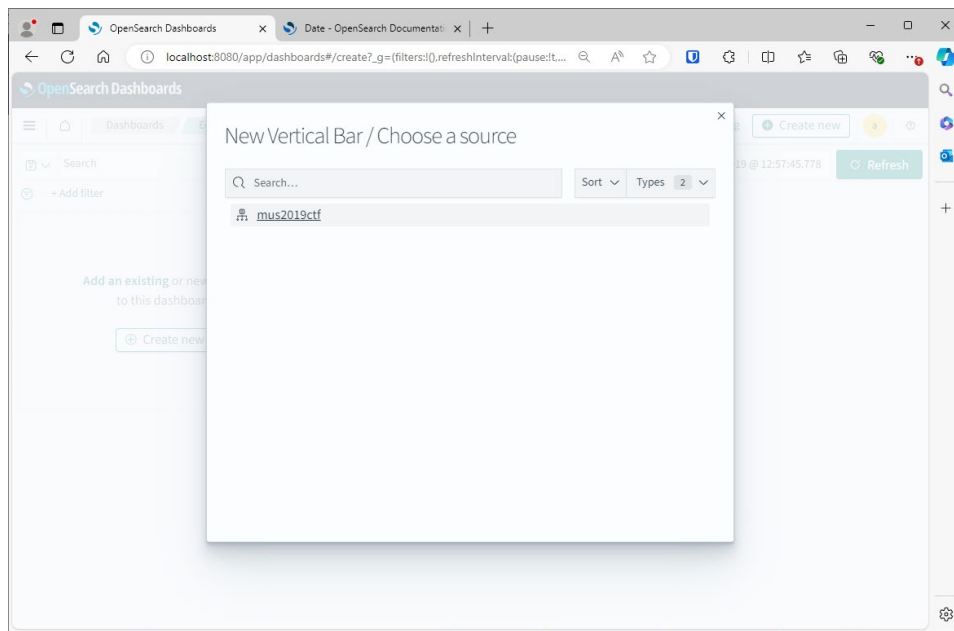
Create a vertical bar chart with date of time for the filtered time frame and filtered against message:http*



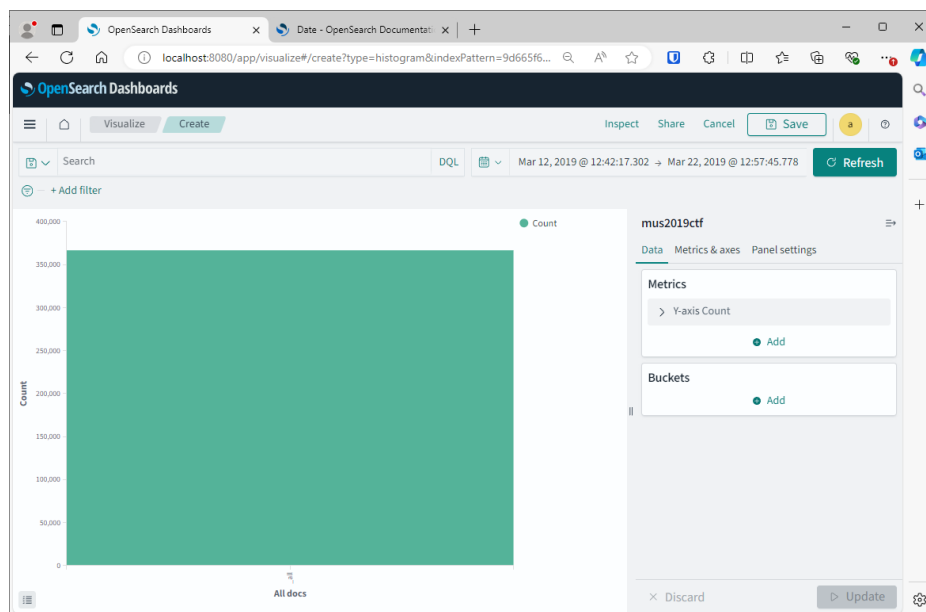
1. Click +Create new



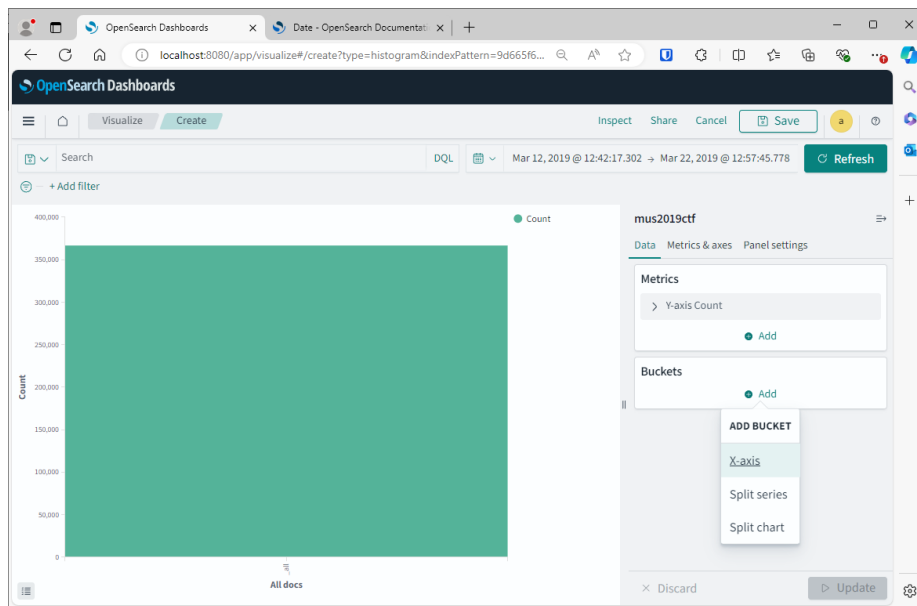
2. Select vertical bar



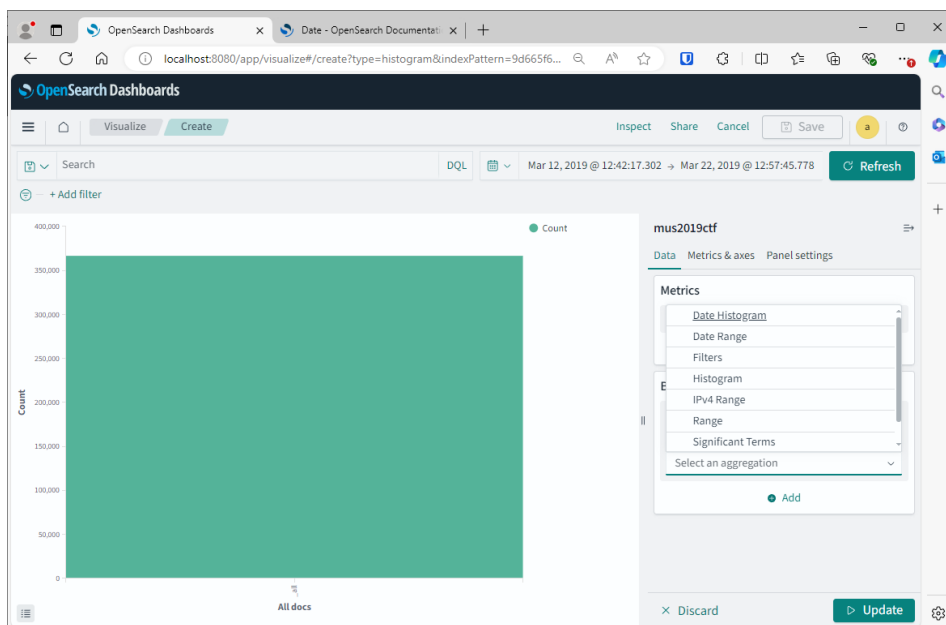
3. Select mus2019ctf index



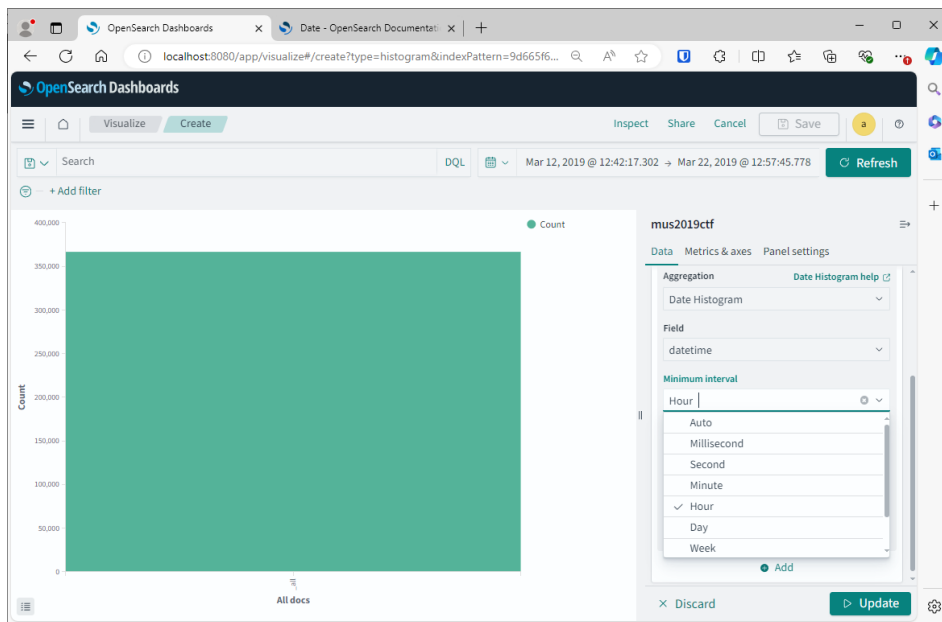
4. Under buckets click Add



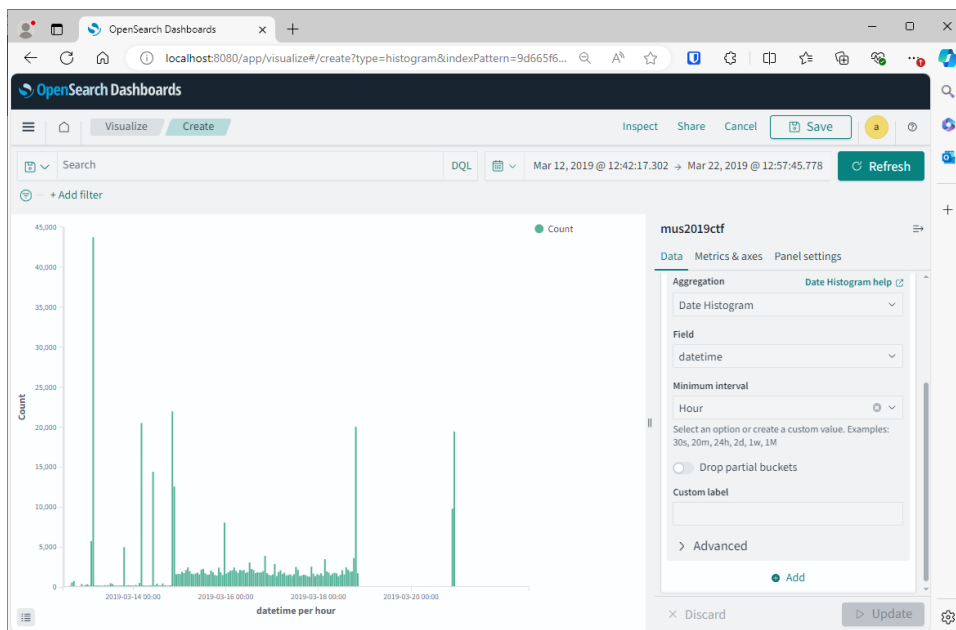
5. Select Aggregation=Date Histogram



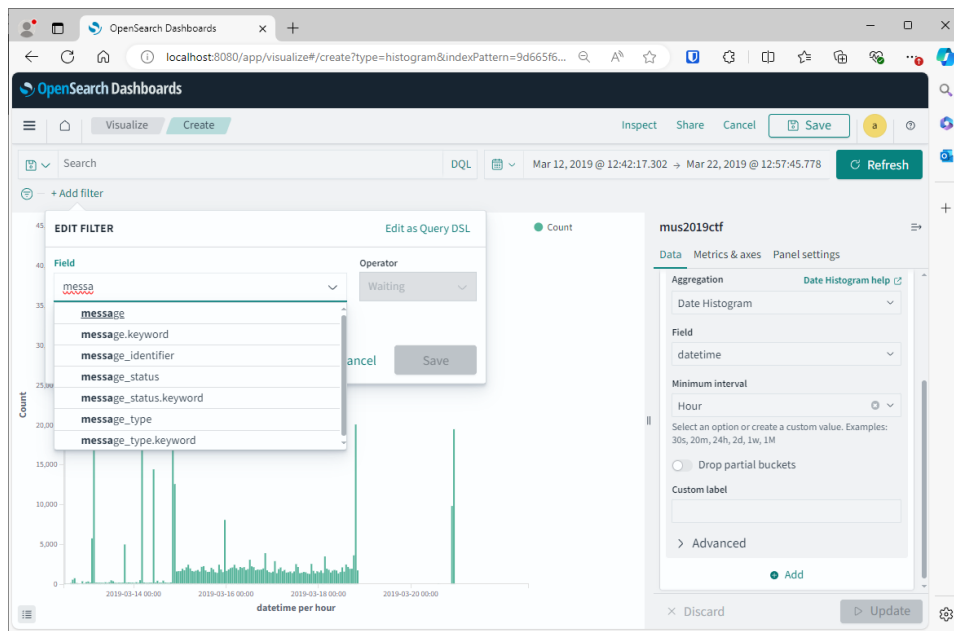
6. Select Hour as minimum time interval



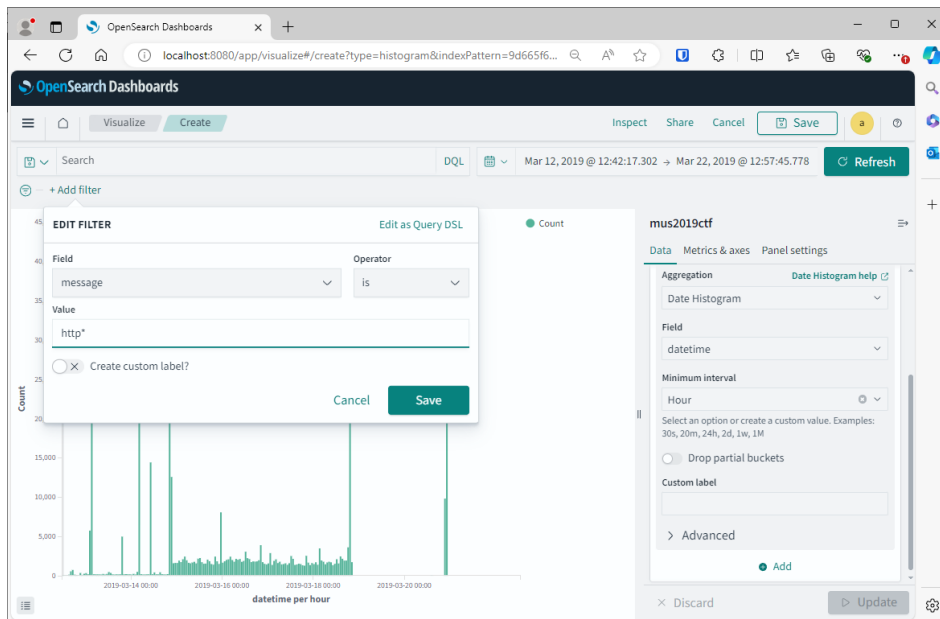
7. Click Update to refresh the graph



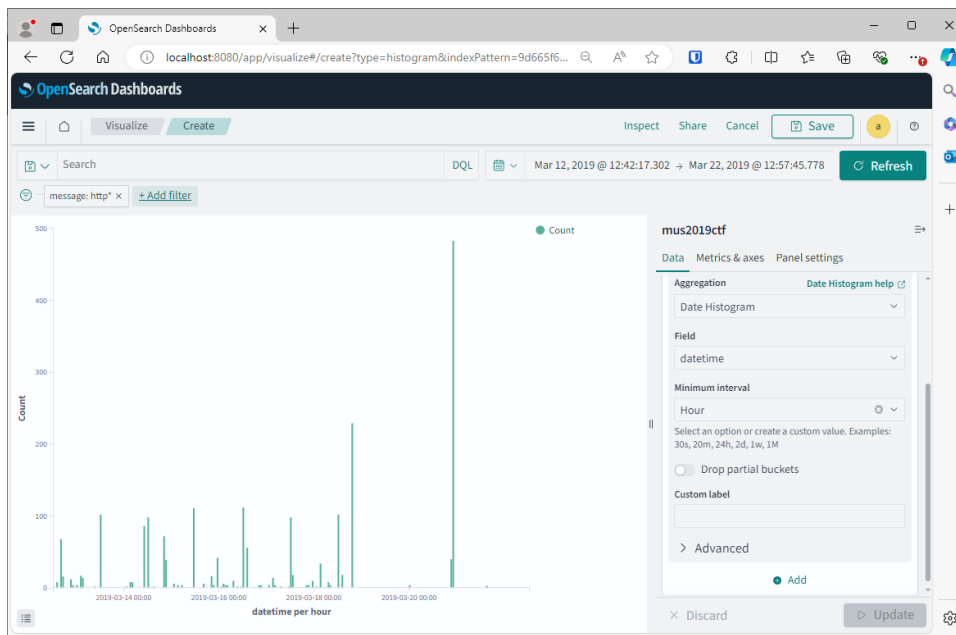
8. Click Add filter and choose Message



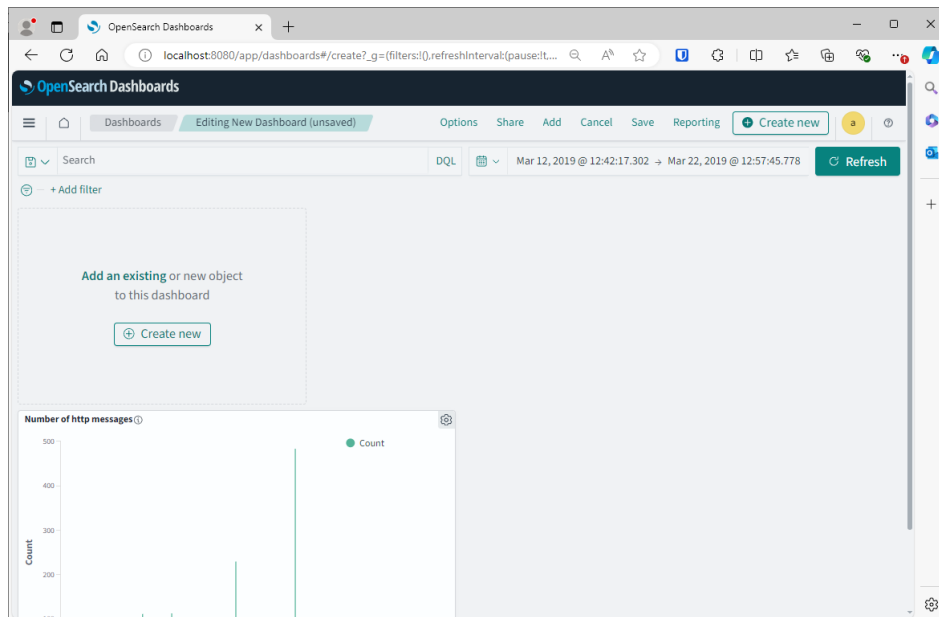
9. Filter on message:http*



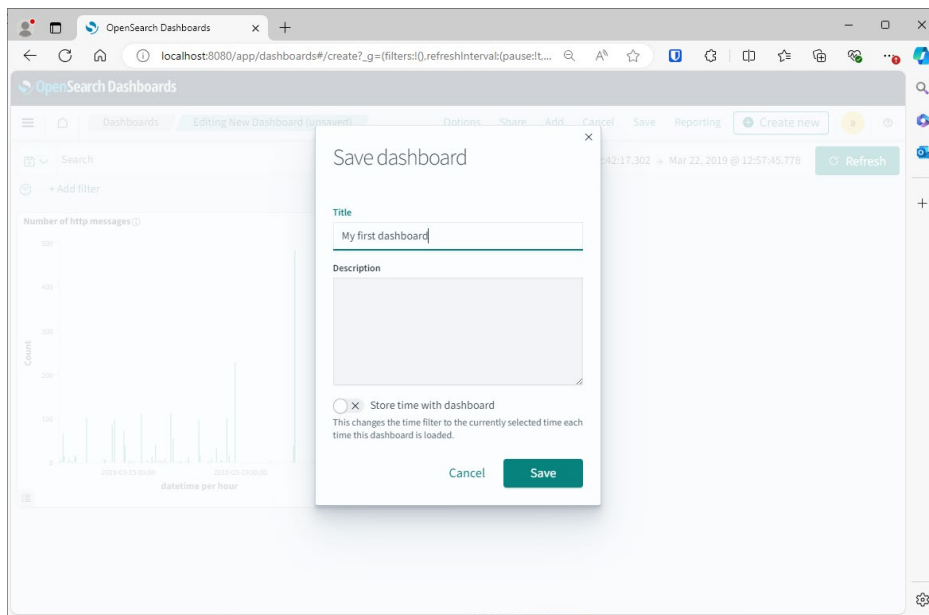
10. Click Save and see the final result



11. Click save in the top right corner to save this Dashboard widget. Give a name and description



12. Save dashboard

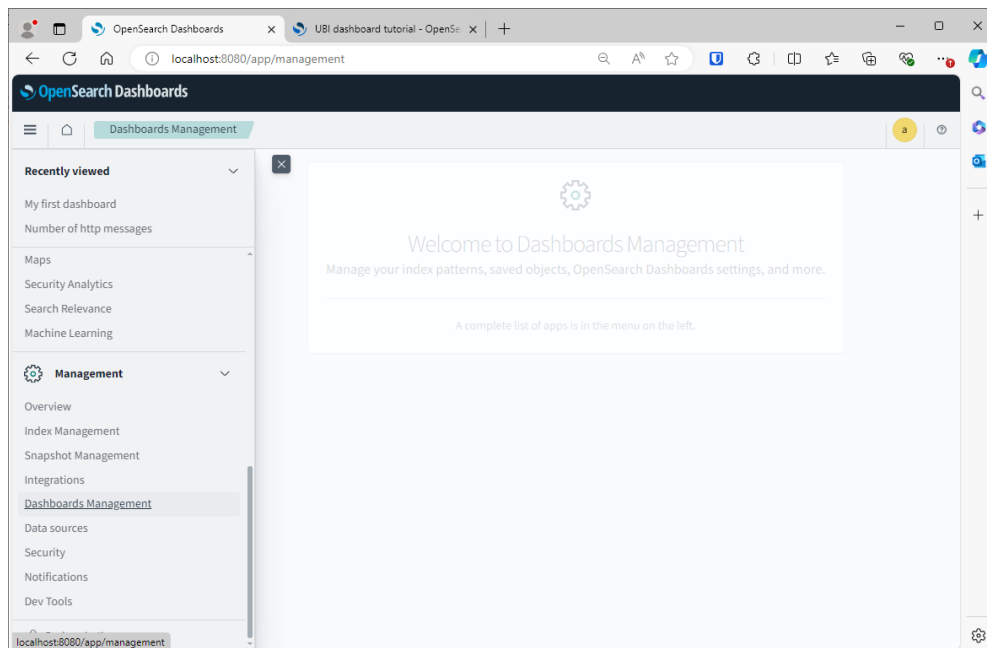


B: Heatmap

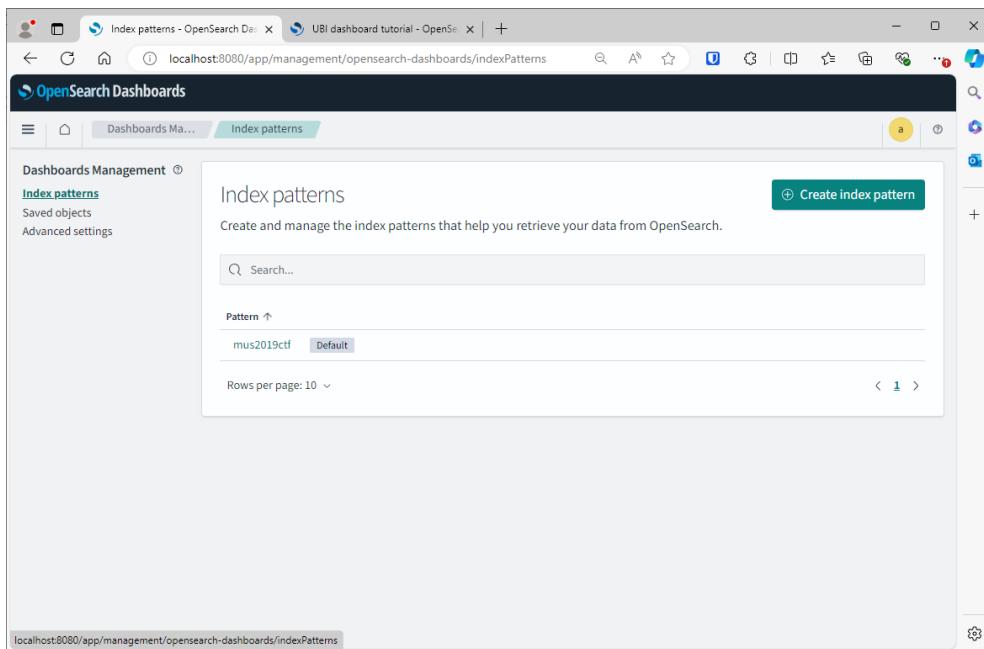
Create a heatmap with number of events per Day of Week vs Hour of Day. In order to do this you need to define scripted values under Index Patterns. See hint below.

Hint for Heatmap:

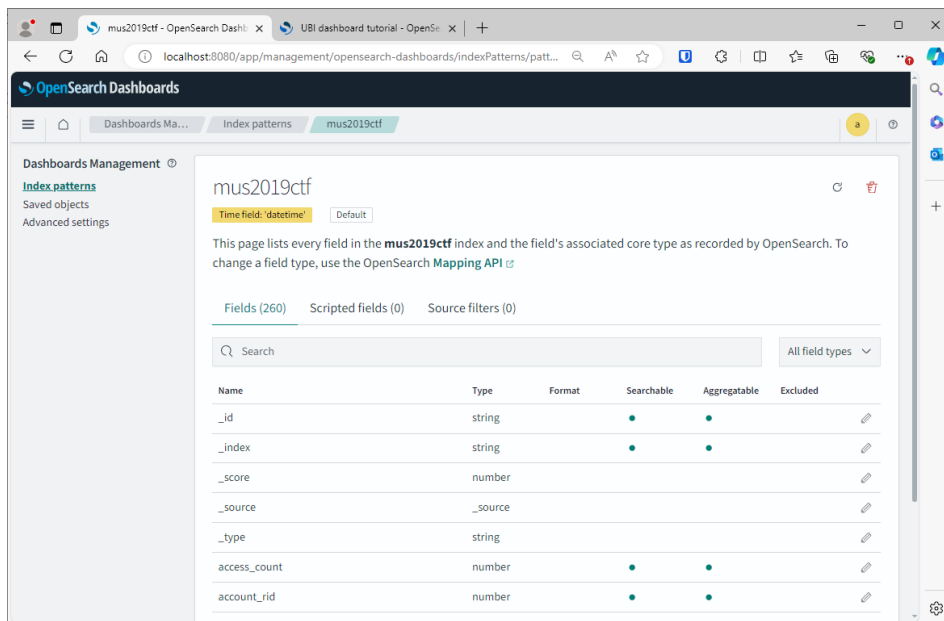
Scripted fields can be defined in the Index Patterns section under Management as follows: open the Home menu, scroll down to the Management section, select Dashboards Management.



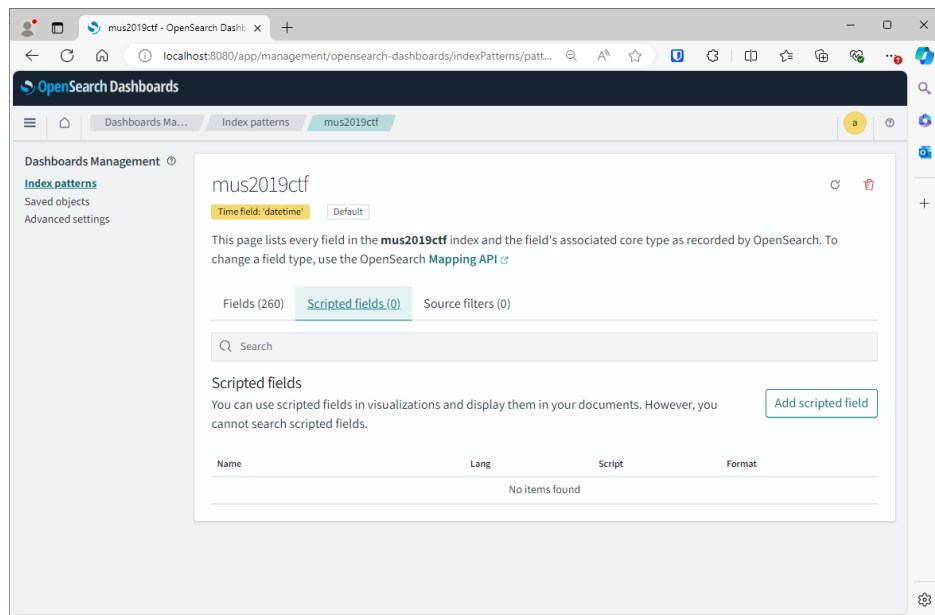
1. In the Management menu select Index Pattern



2 select the mus2019ctf index.

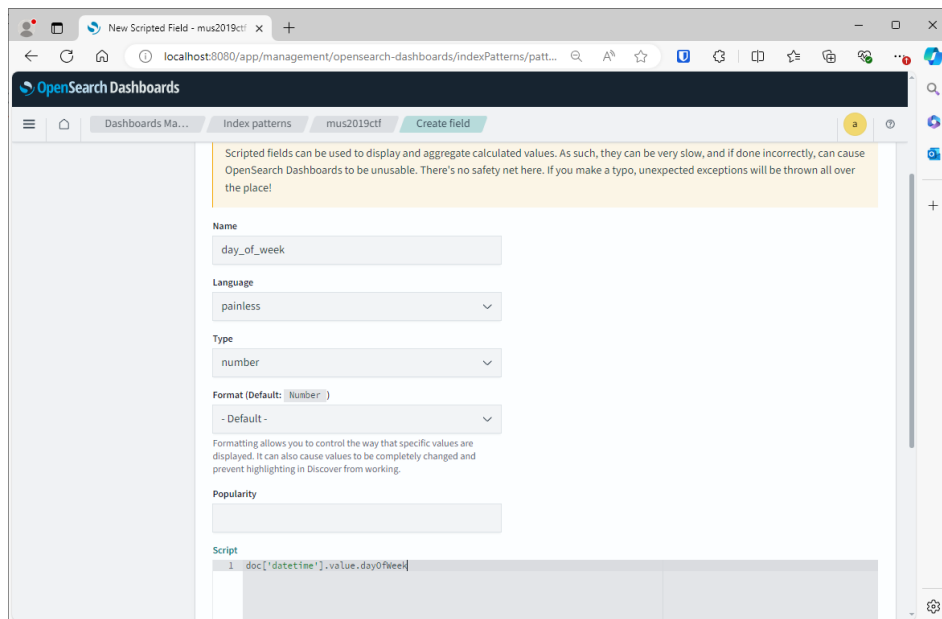


3 Goto scripted fields which is then still empty.



The day_of_week field has the following script:

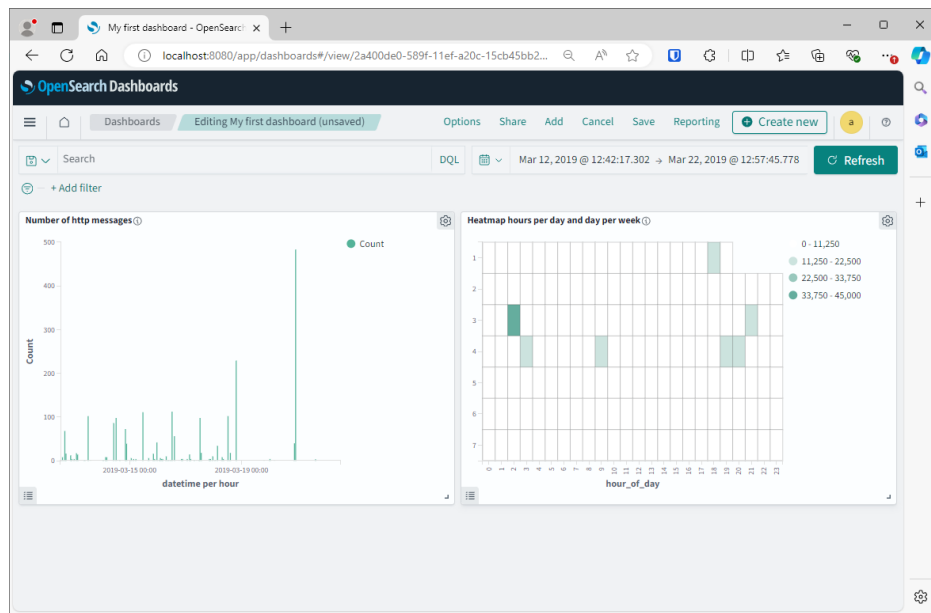
```
doc['datetime'].value.dayOfWeek
```



Same for hour_of_day field has the following script:

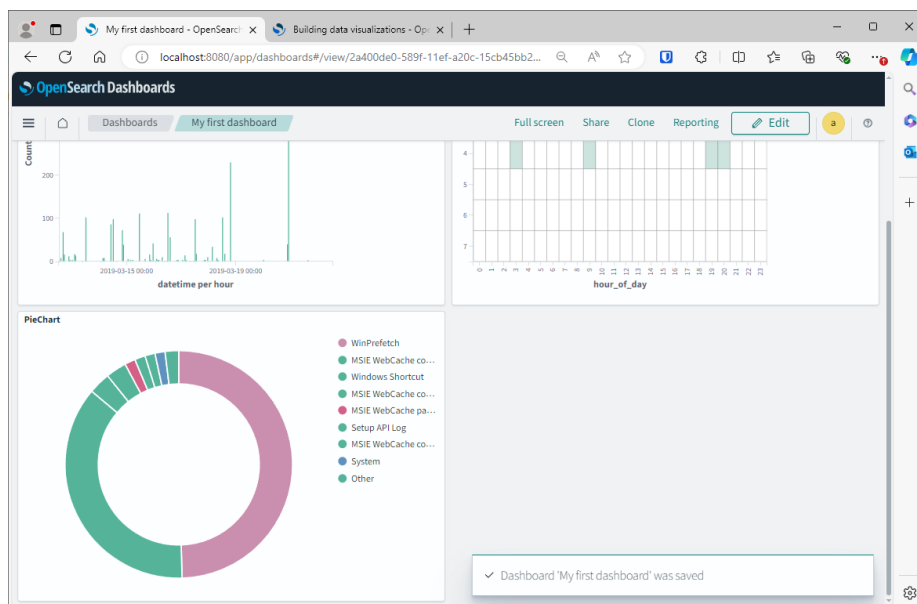
```
doc['datetime'].value.hourOfDay
```

The final result can look like this:



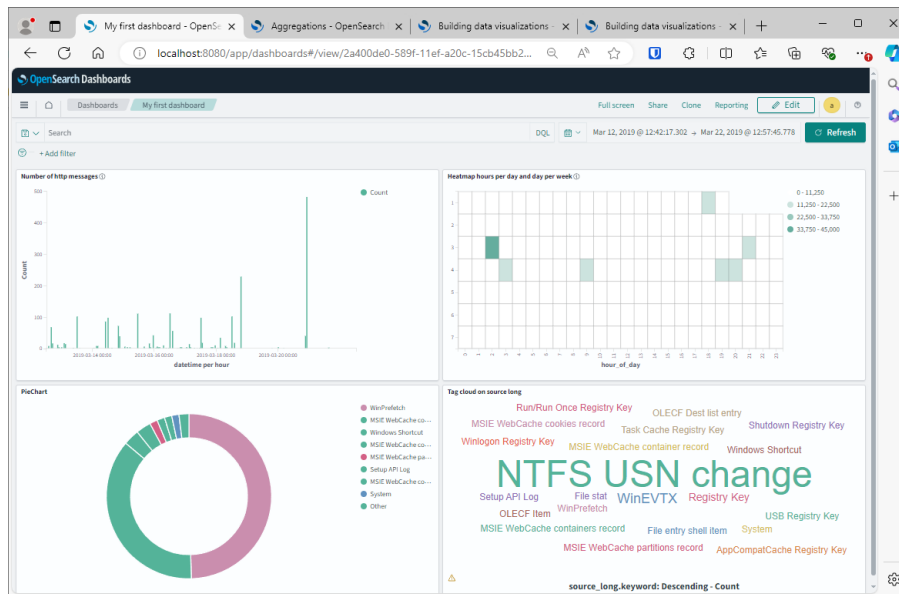
C: PieChart

Create a PieChart based on source_long with count values filtering out NTFS, WinEvtx and Windows registry key.



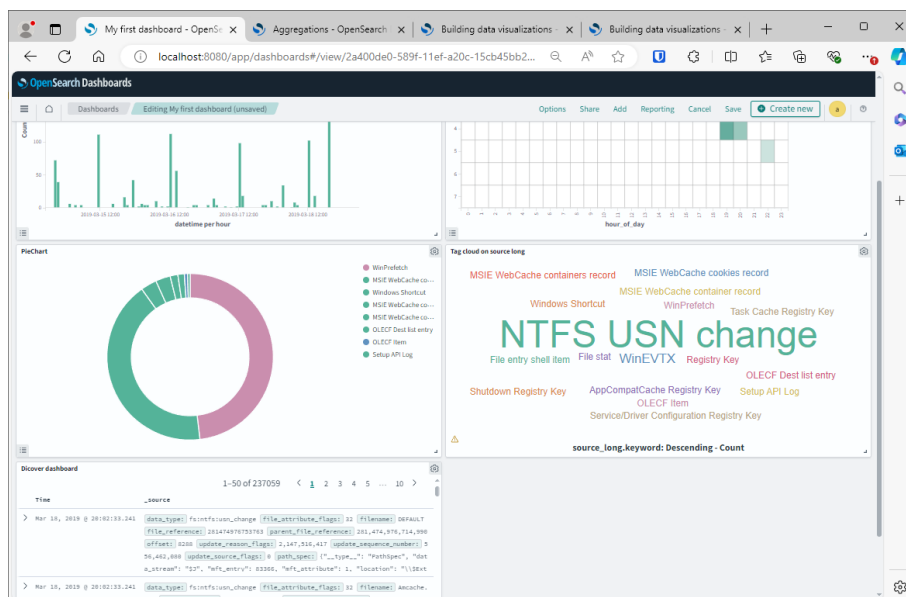
D: Tag Cloud

Create a Tag Cloud based on source_long



E: Add the discover dashboard

It is possible to save the table in the Discover dashboard as a separate visualisation so that we can use it in a custom dashboards. Go to the Discover page and click Save. Then open your own dashboard and add it to your dashboard.



F. Experiment

Experiment a little by selecting value ranges in one visualization and see how the other visualisations are updated simultaneously.

G. More examples

See <https://opensearch.org/docs/2.15/dashboards/visualize/viz-index/> for more examples of visualisations in OpenSearch Dashboards.