Table of Contents

Splunk Homework Assignment

Objective:

Demonstrate your ability to set up and use Splunk with Docker containers for log analysis.

Task 1: Docker Setup

1. Install Docker on your local machine if you haven't already.
2. Pull the official Splunk Docker image from Docker Hub.
3. Create a Docker container with Splunk running. Ensure that the container exposes necessary ports.

Task 2: Log Ingestion with Docker

1. Ingest sample log data into your Splunk container. You can use any sample log data available online or create your own.
2. Verify that the logs are successfully ingested and searchable in Splunk.

Task 3: Basic Search and Visualization

1. Use Splunk search commands to find the count of events with a specific keyword in the logs.
2. Create a basic visualization (chart or graph) based on a search query of your choice.

Task 4: Dockerized Deployment

1. Dockerize a custom application (could be a simple Python script generating logs) and send its logs to your Splunk container.
2. Confirm that the logs from your custom application are being indexed and searchable in Splunk.

Task 5: Alerts and Monitoring

1. Set up an alert to notify you if there are more than 5 error events in the last hour.
2. Monitor the real-time logs for any events containing the word "warning" and display them in real-time.

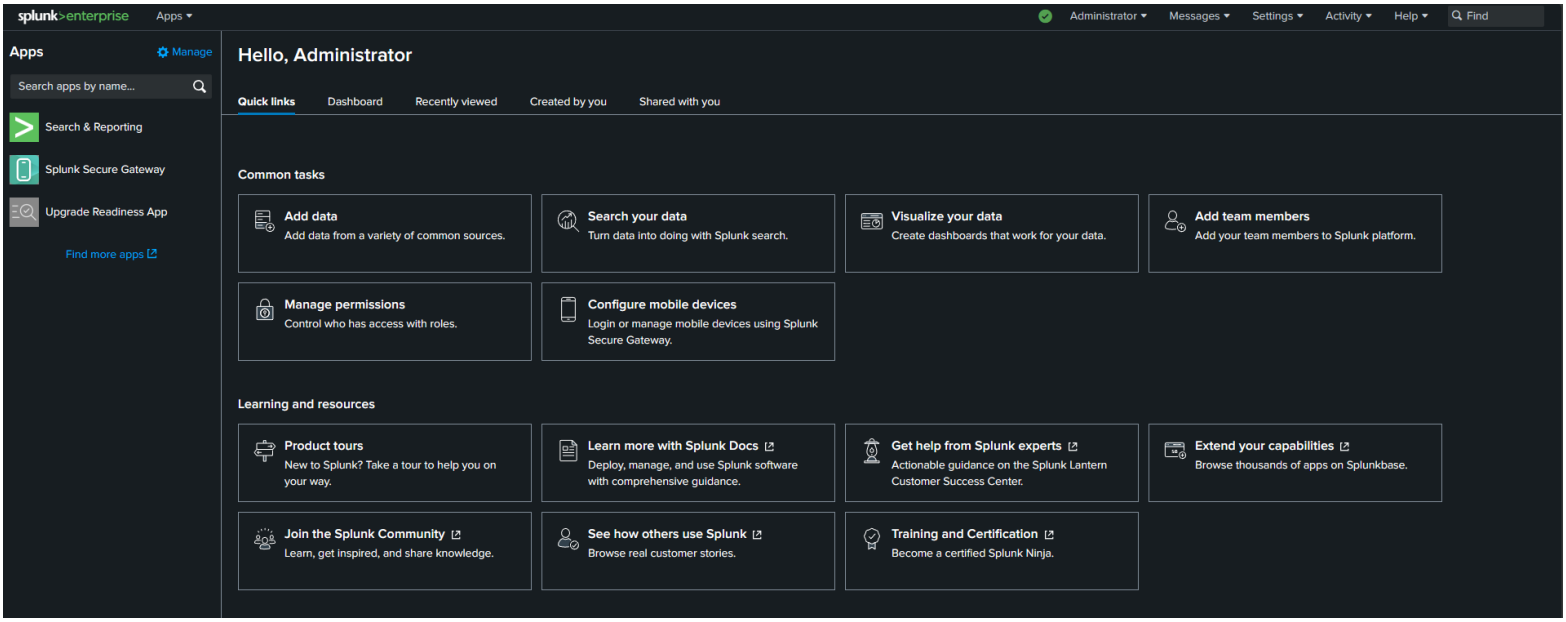
I've started with Windows and switched to Kali 2023 – explanation below

Task 1

1. Download the Splunk image, using the command: docker pull splunk/splunk:latest
2. Run the image, using the command: docker run -d -p 8000:8000 -p 8080:8080 -e SPLUNK_START_ARGS='--accept-license' -e SPLUNK_PASSWORD='password' --name splunk splunk/splunk:latest
3. The container is initializing, wait for it to complete. You can use the command: docker ps - To see all the containers data including status

```
┌──(root㉿kali)-[/home/kali]
└─# docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED        STATUS                  PORTS                                                                                   NAMES
f86be3a526aa   splunk/splunk:latest   "/sbin/entrypoint.sh…"   13 minutes ago   Up 6 minutes (healthy)   8065/tcp, 8088-8089/tcp, 8191/tcp, 9887/tcp, 0.0.0.0:8000→8000/tcp, :::8000→8000/tcp, 9997/tcp   splunk
```

4. The Splunk container is up and running <Kali_IP:8000> or from the Kali machine <localhost:8000>



Problems:

I had a problem with the stage 2 when running the container on Windows environment:

```
TASK [splunk_standalone : Setup global HEC] ***********************************
fatal: [localhost]: FAILED! => {
    "cache_control": "private",
    "changed": false,
    "connection": "Close",
    "content_length": "130",
    "content_type": "text/xml; charset=UTF-8",
    "date": "Thu, 11 Jan 2024 16:25:03 GMT",
    "elapsed": 0,
    "redirected": false,
    "server": "Splunkd",
    "status": 401,
    "url": "https://127.0.0.1:8089/services/data/inputs/http/http",
    "vary": "Cookie, Authorization",
    "warnings": [
        "Module did not set no_log for password"
    ],
    "www_authenticate": "Basic realm=\"/splunk\"",
    "x_content_type_options": "nosniff",
    "x_frame_options": "SAMEORIGIN"
}

MSG:

Status code was 401 and not [200]: HTTP Error 401: Unauthorized
```

This is because the Splunk user does not have permission to edit the /var directory. I switched to Kali 2023 on VMware player, and ran it as root, the problem was solved. There is a quick fix in the (2) resource ("ANSIBLE_EXTRA_FLAGS=-vv"), but it's mainly for DEBUG proposes, so I want with the Kali solution. Also, my computer is preforming better with this technique.

Resources:

1. Deploy and run Splunk Enterprise inside a Docker container - Splunk Documentation
2. Unable to install Standalone Splunk when using doc... - Splunk Community
3. Data Storage ## | docker-splunk
4. Navigation | docker-splunk

Note:

In the second stage I've exposed two ports. One for the Splunk Enterprise and one for Task 4, where I will need an additional port to send data to Splunk.

To stop the container for later use, enter the following command in the Terminal:

docker container stop <container_name>

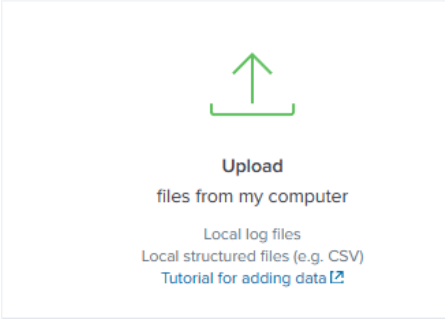
To start the container, enter the following command in the Terminal:

docker container start <container_name>

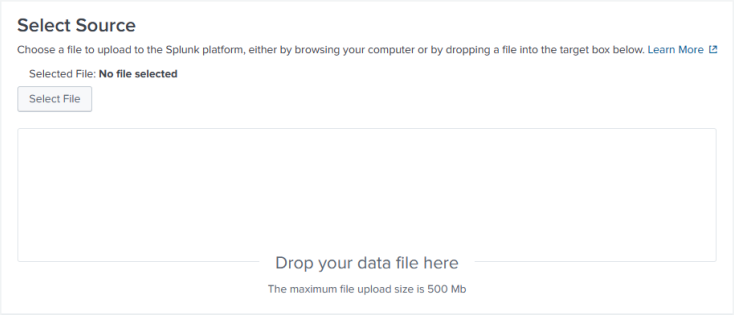
Task 2

1. At the home page select “Add data” option

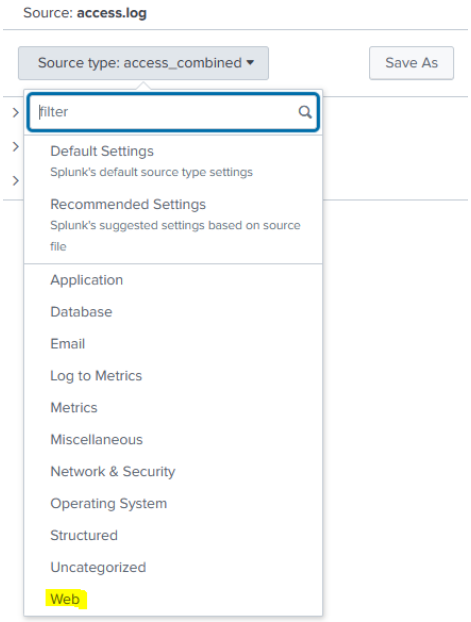
2. Select “Upload” option



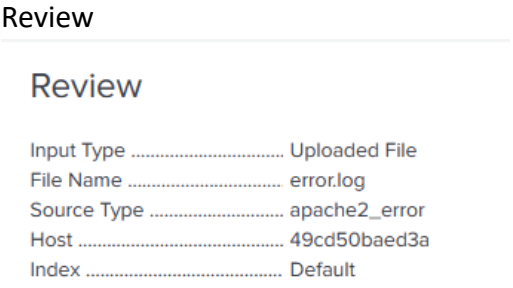
3. In this page select the log files you want to add. I’ve used dummy logs from apache2 web server. Location on Linux machine /var/log/apache2. (access.log)



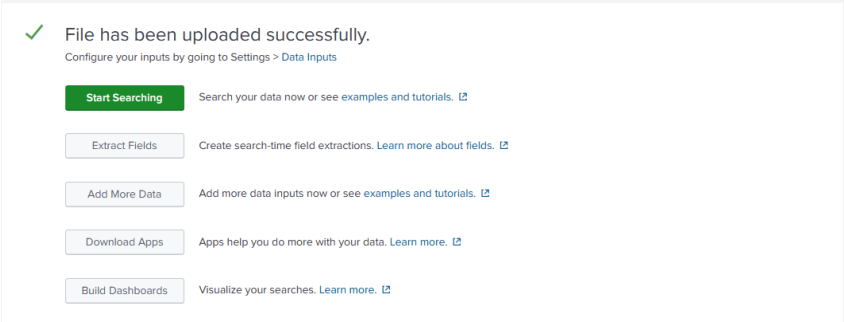
4. Select the source type for the log file. I’ve selected a type from the web option.



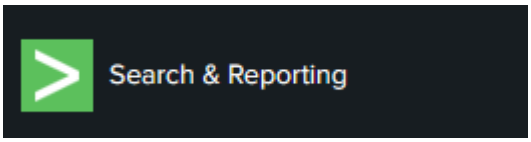
5. Review



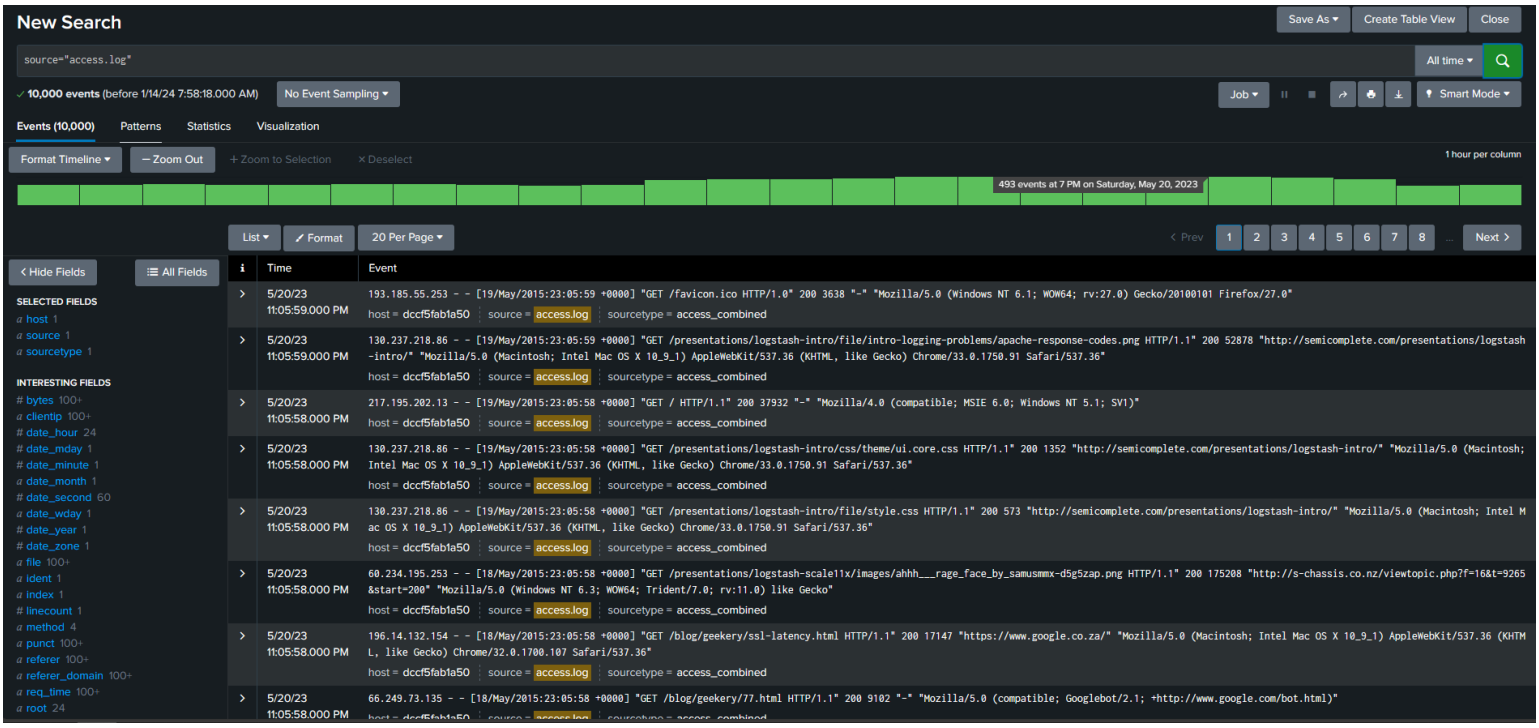
6. Complete



7. I've uploaded a file named access.log, record the access to the web server.
8. To search your data at the home page, select "Search & Reporting" option



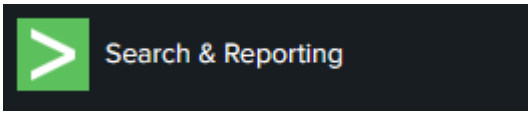
9. You can search your events by host, sourcetype, source or any other data you have. for example: IP address



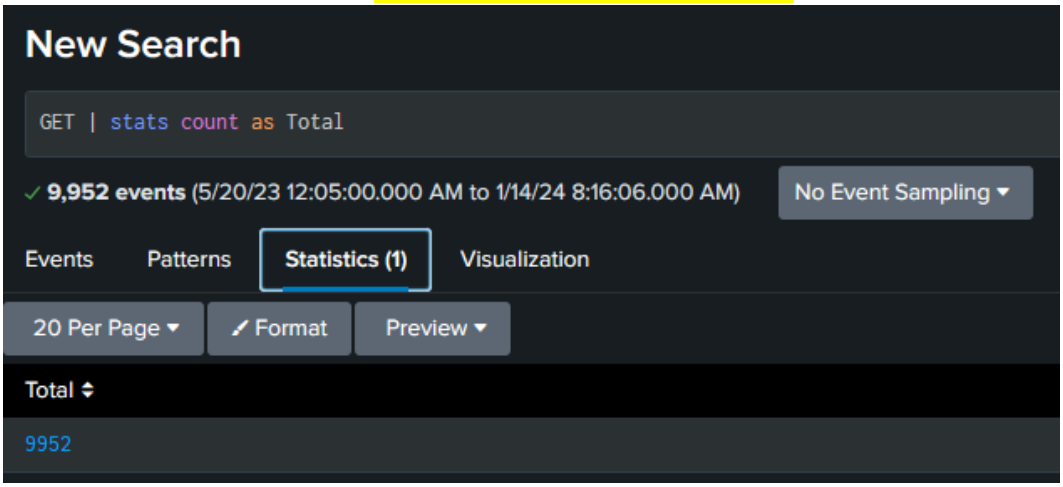
Task 3

Please take a look at the image from Task 2 stage 8. We can see a keyword GET. Let's return the count of events that have this keyword in them.

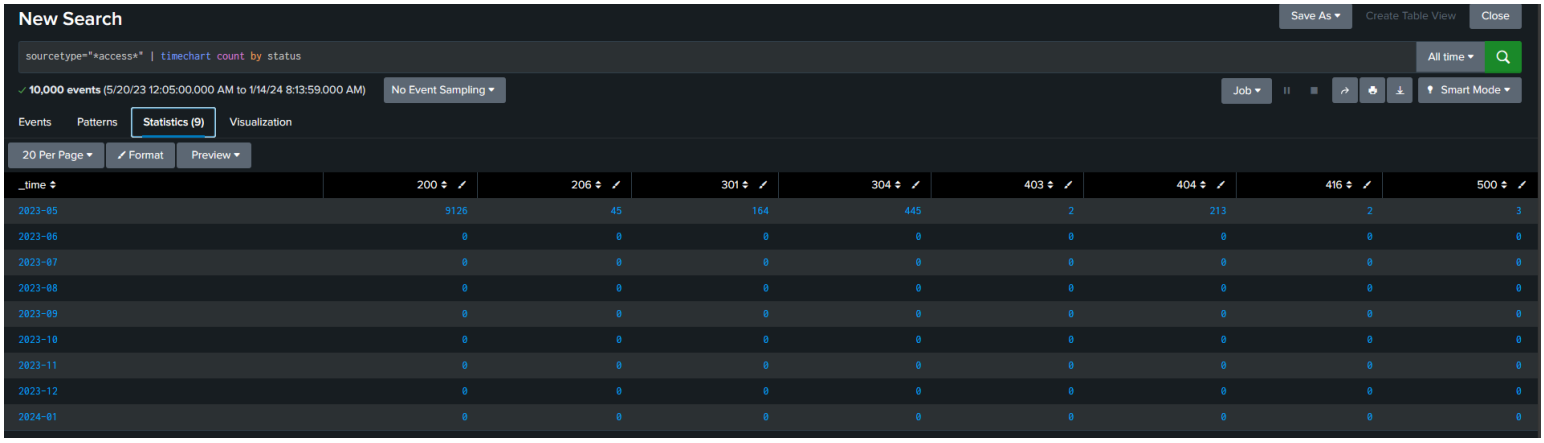
1. In Splunk, from the main page, click the search button



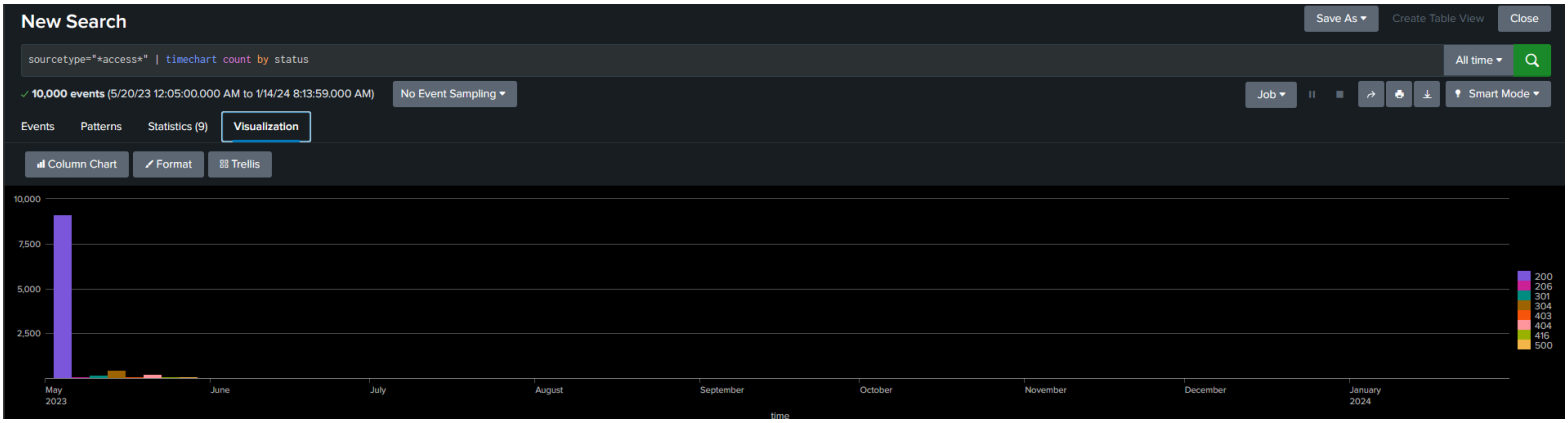
2. To count the number of events that have this keyword use the following command in the search bar: <keyword> | stats count as Total



3. Now we want to visualize our data. I will show two different ways to create a basic visualization for the data:
   I. Using the following command: sourcetype="*access*" | timechart count by status. sourcetype="*access*" will find all source types with the word access in them. (* means - I don't care what comes before and after the word). The second part of the command will generate a time chart form the data using the status field.



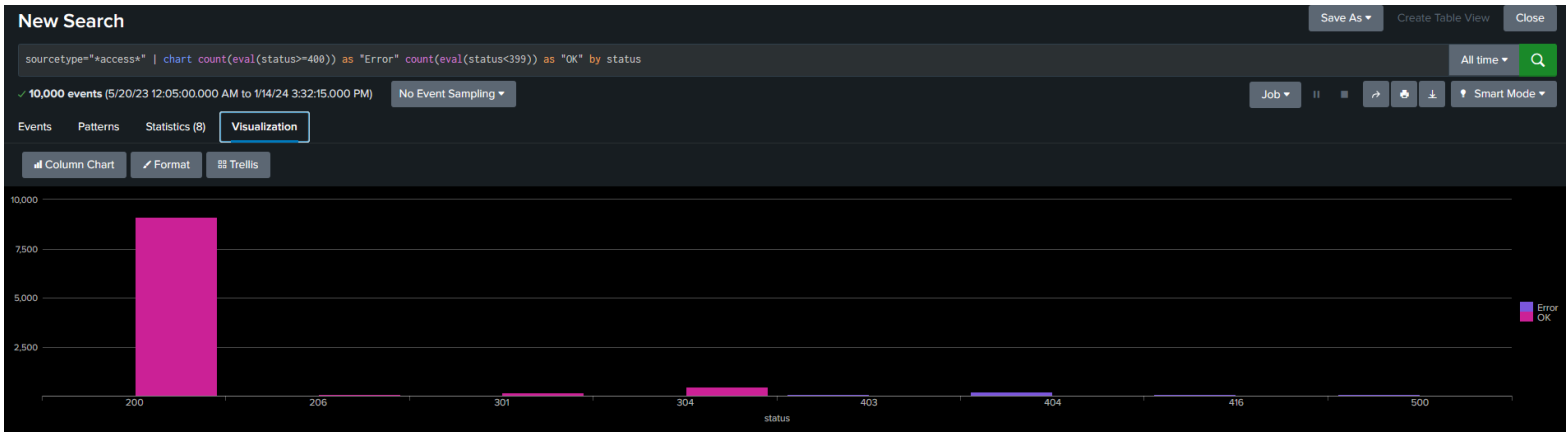
_time	200	206	301	304	403	404	416	500
2023-05	9126	45	164	445	2	213	2	3
2023-06	0	0	0	0	0	0	0	0
2023-07	0	0	0	0	0	0	0	0
2023-08	0	0	0	0	0	0	0	0
2023-09	0	0	0	0	0	0	0	0
2023-10	0	0	0	0	0	0	0	0
2023-11	0	0	0	0	0	0	0	0
2023-12	0	0	0	0	0	0	0	0
2024-01	0	0	0	0	0	0	0	0



II. To show the status codes without time stamp use the commend: sourcetype="*access*" | chart count(eval(status>=400)) as "Error" count(eval(status<399)) as "OK" by status

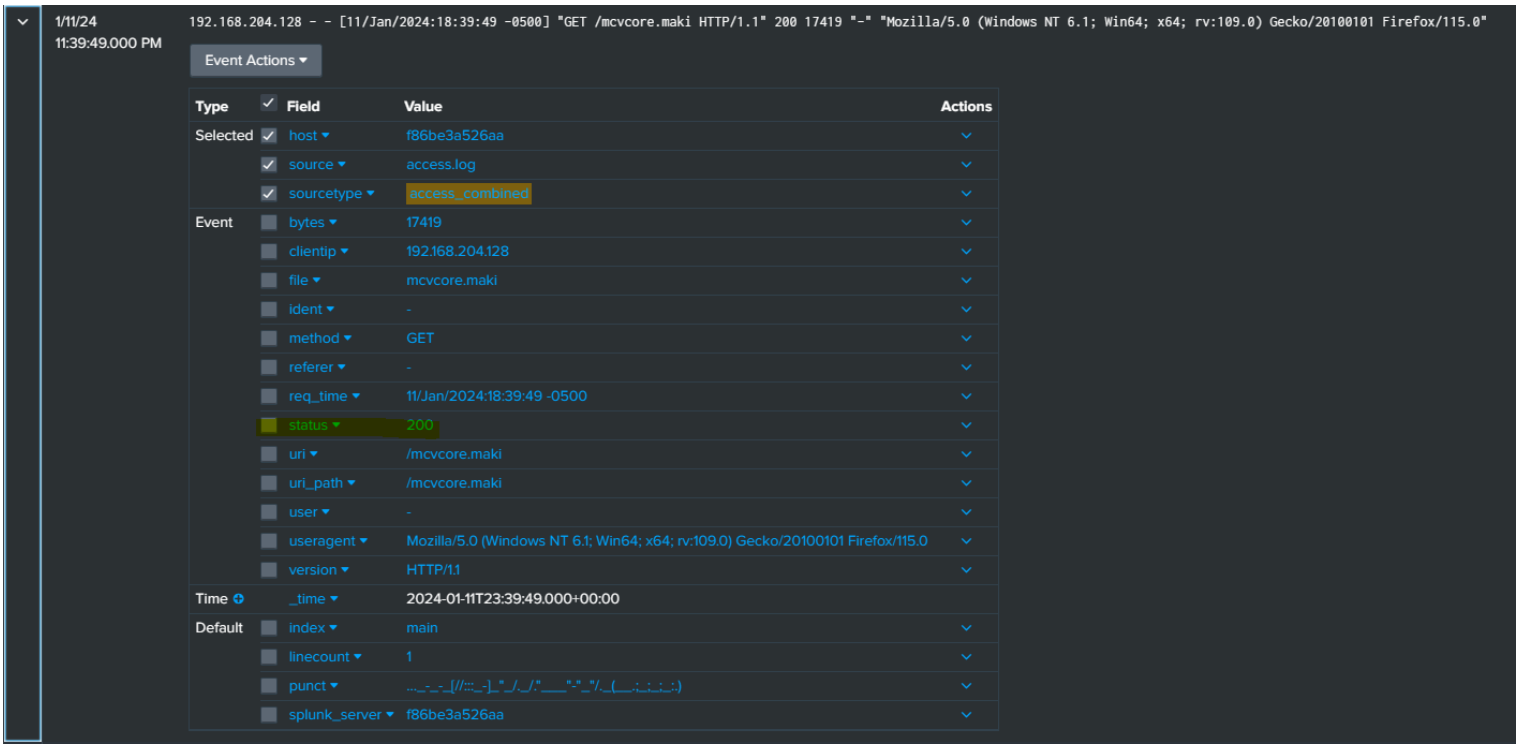
The second part of the command splits the status codes in two – error and ok. The eval command calculates an expression and puts the resulting value into a field. Next set all the error codes to a label named “Error” and the successful codes in to the “OK” label to display on the chart.

status	Error	OK
200	0	9126
206	0	45
301	0	164
304	0	445
403	2	0
404	213	0
416	2	0
500	3	0



Problems:

I had a problem with the visualization of the data I've uploaded to Splunk. I saw online that some peoples use status as a variable in the search bar, so I've came to the conclusion that the way I've uploaded the data was incorrect. I've uploaded the data again but now used the propre type for the data and got a status field for each event. Finally, I've updated Task 2 in this document.



Note:

I've deleted the old data with the command: <search_query> | delete

For this command to execute successfully you need to add a can_delete permission to the user you are working with. Find it in Settings -> Roles.

After deleting the data I've removed this permission for security reasons.

Resources:

Solved: How to count number of events in a search result? - Splunk Community

Solved: How to chart a daily count of HTTP status codes di... - Splunk Community

Solved: How to display table of total error status code an... - Splunk Community

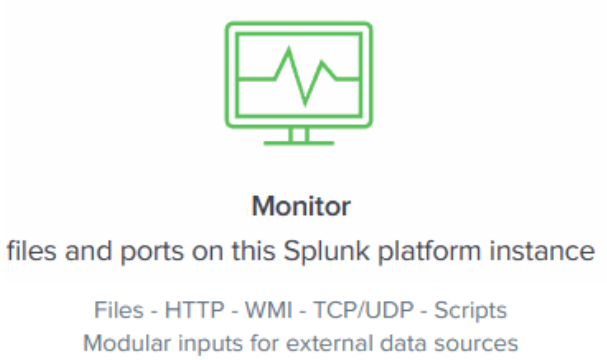
Create a basic chart - Splunk Documentation

Task 4

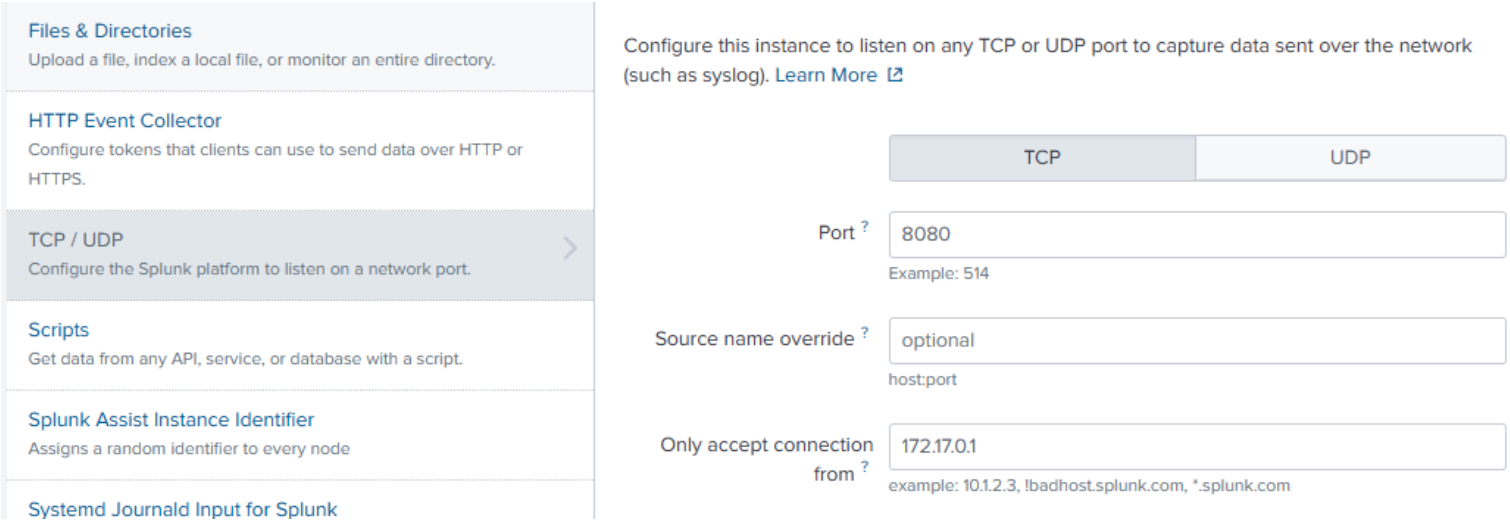
The python script reads from a dummy log file and send the data to the Splunk Container.

1. let's set up a TCP connection to the Splunk container. Remember we have one more exposed port (8080) to use for this purpose. In the main page of Splunk click on add data

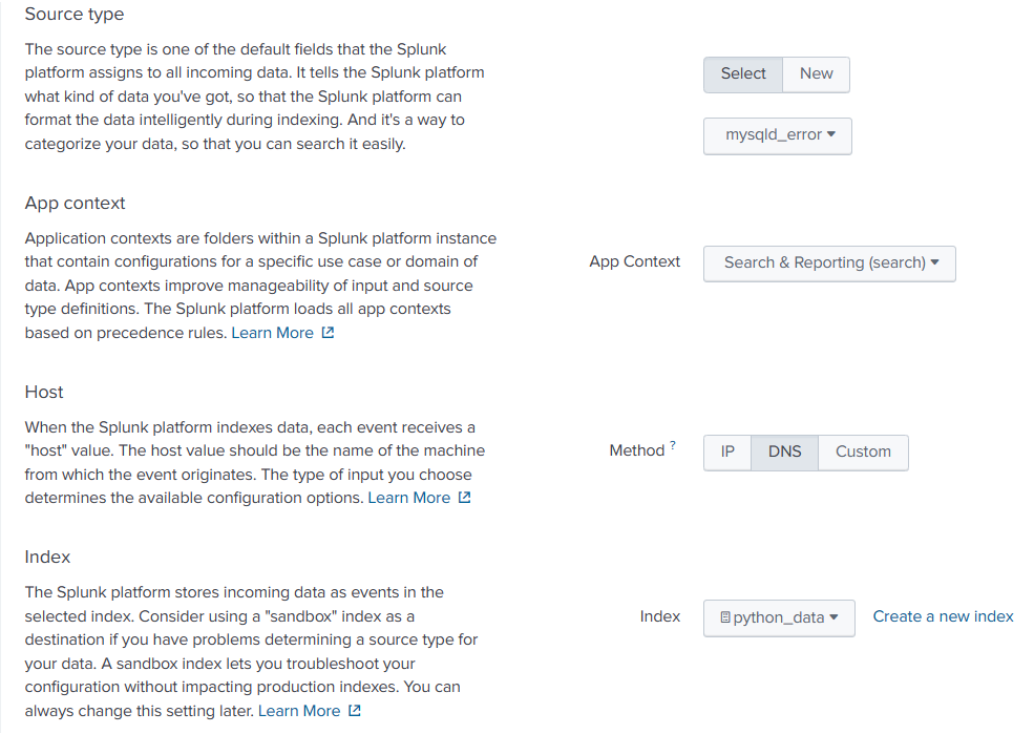
2. Click on the monitor option



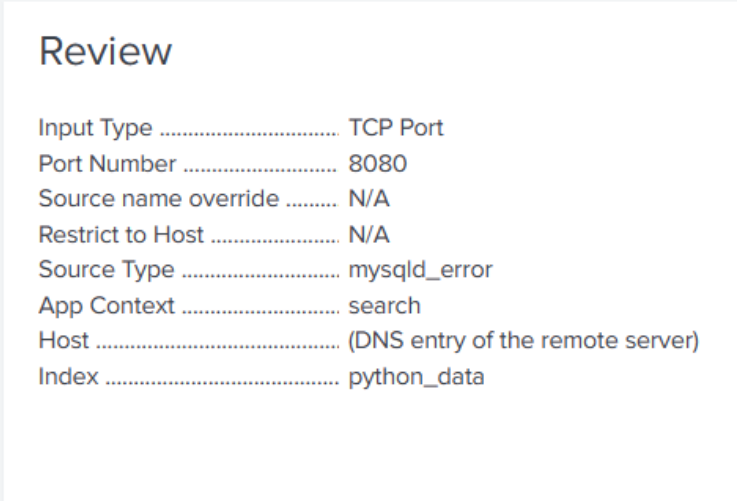
3. Then click on the TCP/UDP option and enter a port (8080). For security reasons set up the docker IP as the only accept connection.



4. Source type should be mysql_error. I've created a new index – Python_data, a repository for all the data (events) sent by Python.



5. Finally, review the TCP input



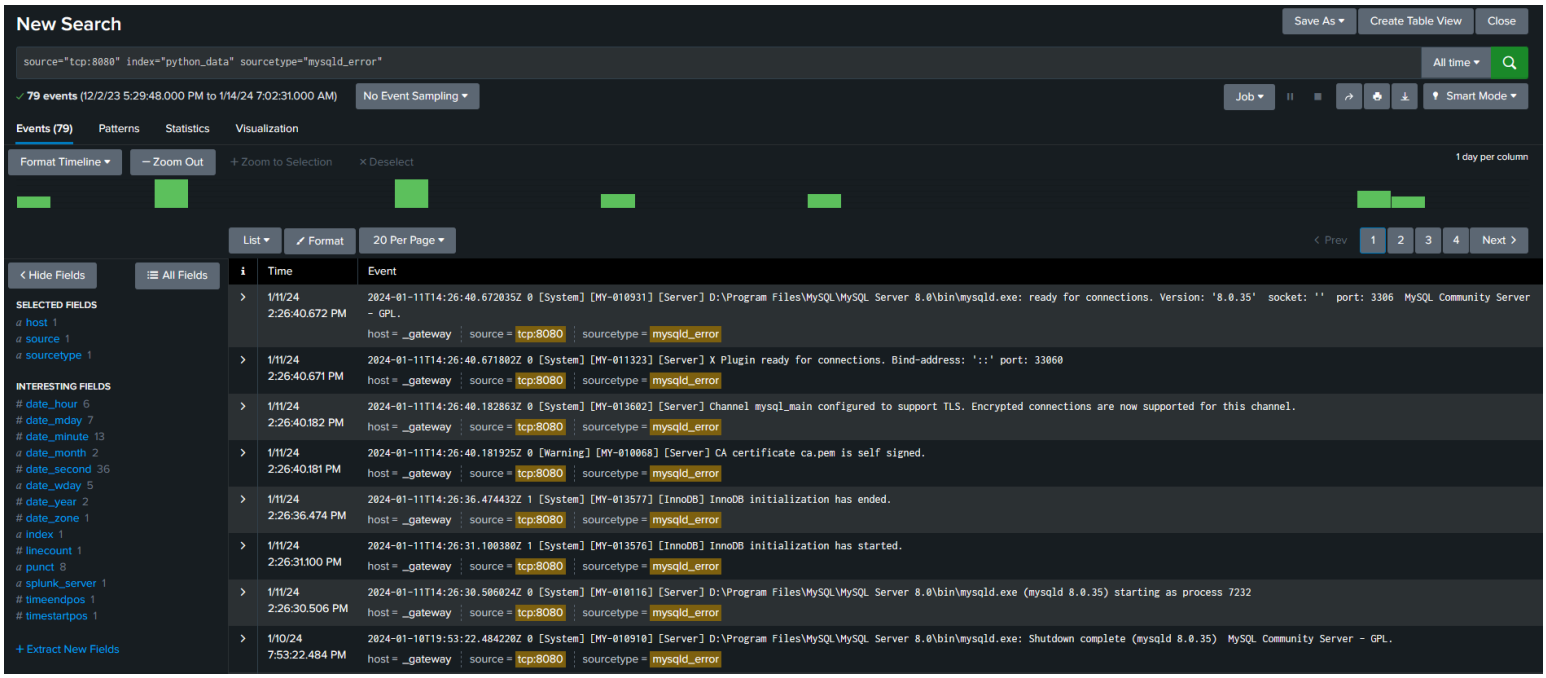
Nex step is to set up a container with our simple application –

6. Copy the Python script and the log file to the Kali machine, to a directory.
7. We need to create a Dockerfile to create an image out of our script. Enter this command in the terminal: touch Dockerfile
8. Use the text editor or vim (Linux command) to edit the file (I'm including the files in the email). We used a base image of Alpine Linux running Python, a minimalist Linux distro, which helps keep the images for Docker small. COPY will move the application into the container image, WORKDIR sets the working directory.

9. Now we need to build an image with the command: docker build --tag send_data_to_splunk .
   The name of the image is send_data_to_splunk
10. Finally, we can start the application as a container, using the command: docker run --name python-app send_data_to_splunk

```
┌──(root㉿kali)-[/home/kali]
└─# docker run --name python-app send_data_to_splunk
Connection has been made
Sending error number 0
Connection has been made
Sending error number 1
```

11. On the Splunk search we can search for those events



<u>Problems</u>:

When I was trying to send all the file log without closing the socket, I got one long event. This is not a desired behavior, so I've closed the socket after every log (line in the file) sent and we got 79 events from a 79-line log file. To solve this problem, I've used common sense.

<u>Note:</u>

For this task I'm using dummy logs form MySQL Database. After research online I found that a quick setup of TCP on Splunk will give us the desirable solution to this task.

<u>Recurses</u>:

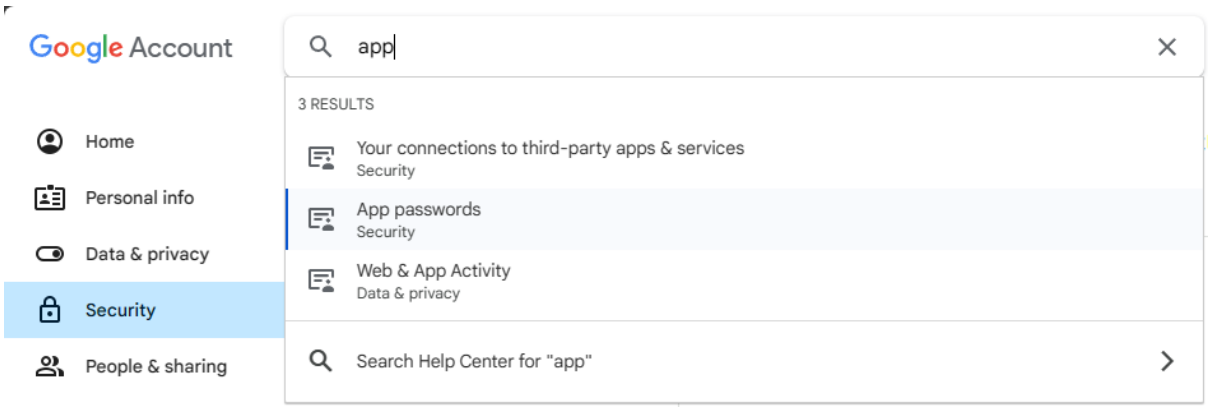
[how to send logs to splunk - חיפוש ב-Google]

[Containerize a Python App in 5 Minutes - Atmosera]

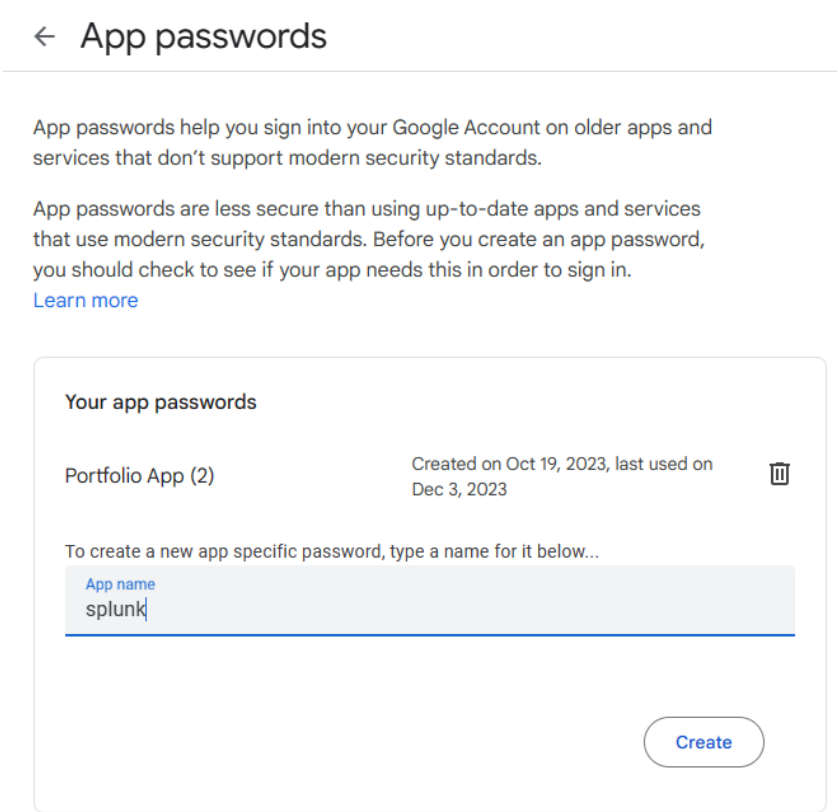
Task 5

In this task we want to set up real-time alerts for the incoming data from Task 4.

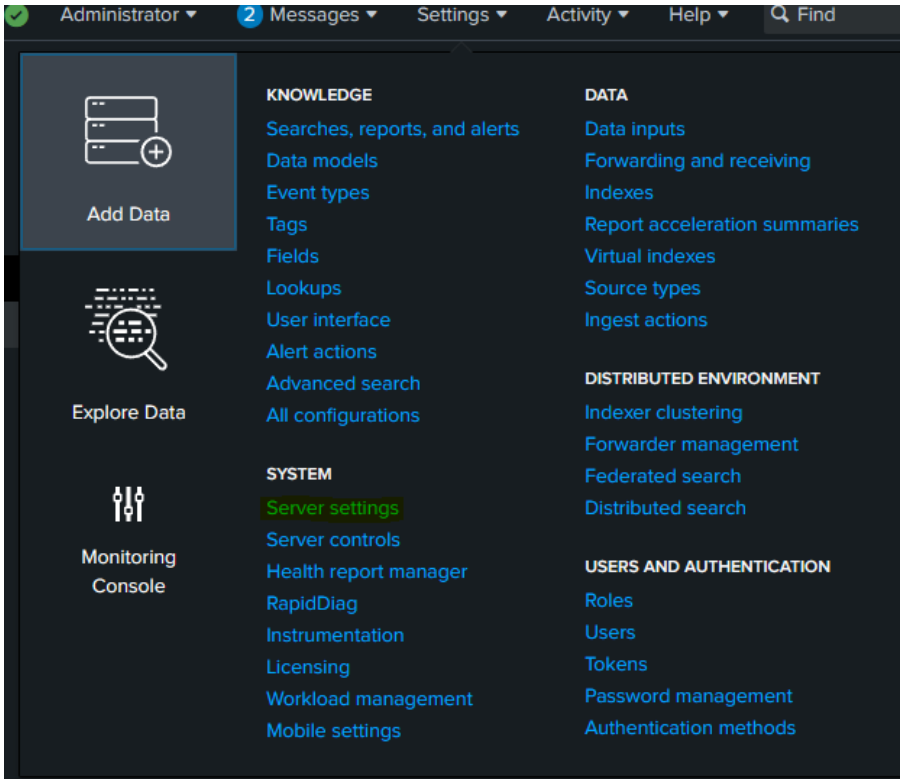
1. First, we want the receive an email for every event containing the word “warning”. For this purpose, I’ve used Gmail. To do so we need to set a new application. Go to Google account and search for app. Click on the app passwords.



2. Then you will need to set up a new app. You will get a password, please save it for later.

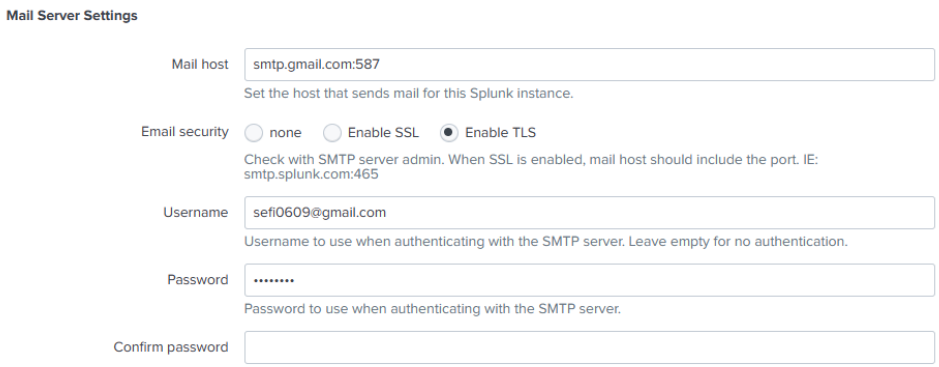


3. We need to setup the email setting on Splunk. Go to Settings -> Server settings

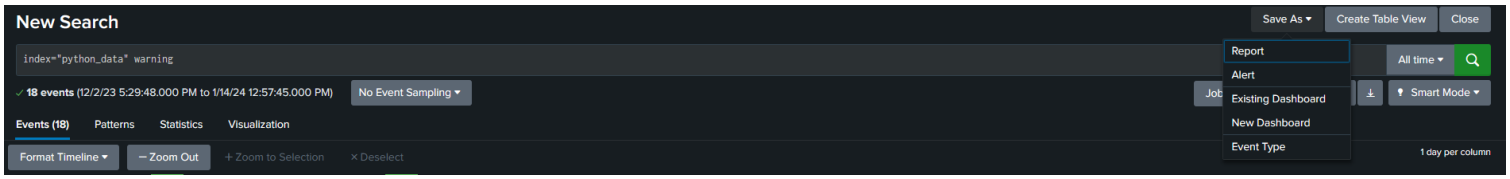


4. Click on Email Settings

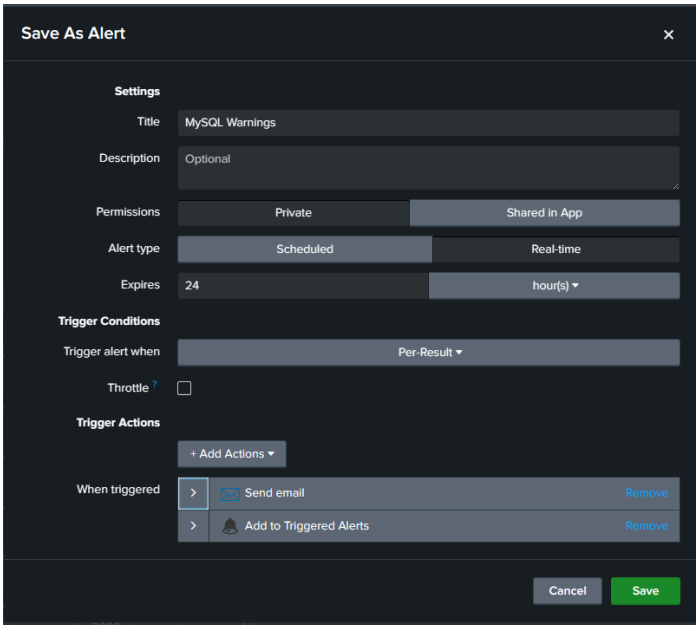
5. Set up the Mail host, username and password (the app password we created). Click Save.



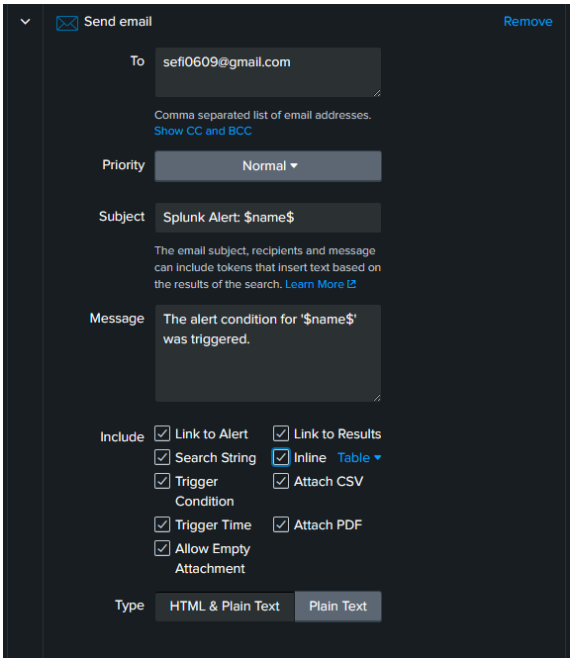
6. Now we can set up an alert. Go to search and enter your query to the search bar. Then click Save As -> Alert. My search query is: index="python_data" warning



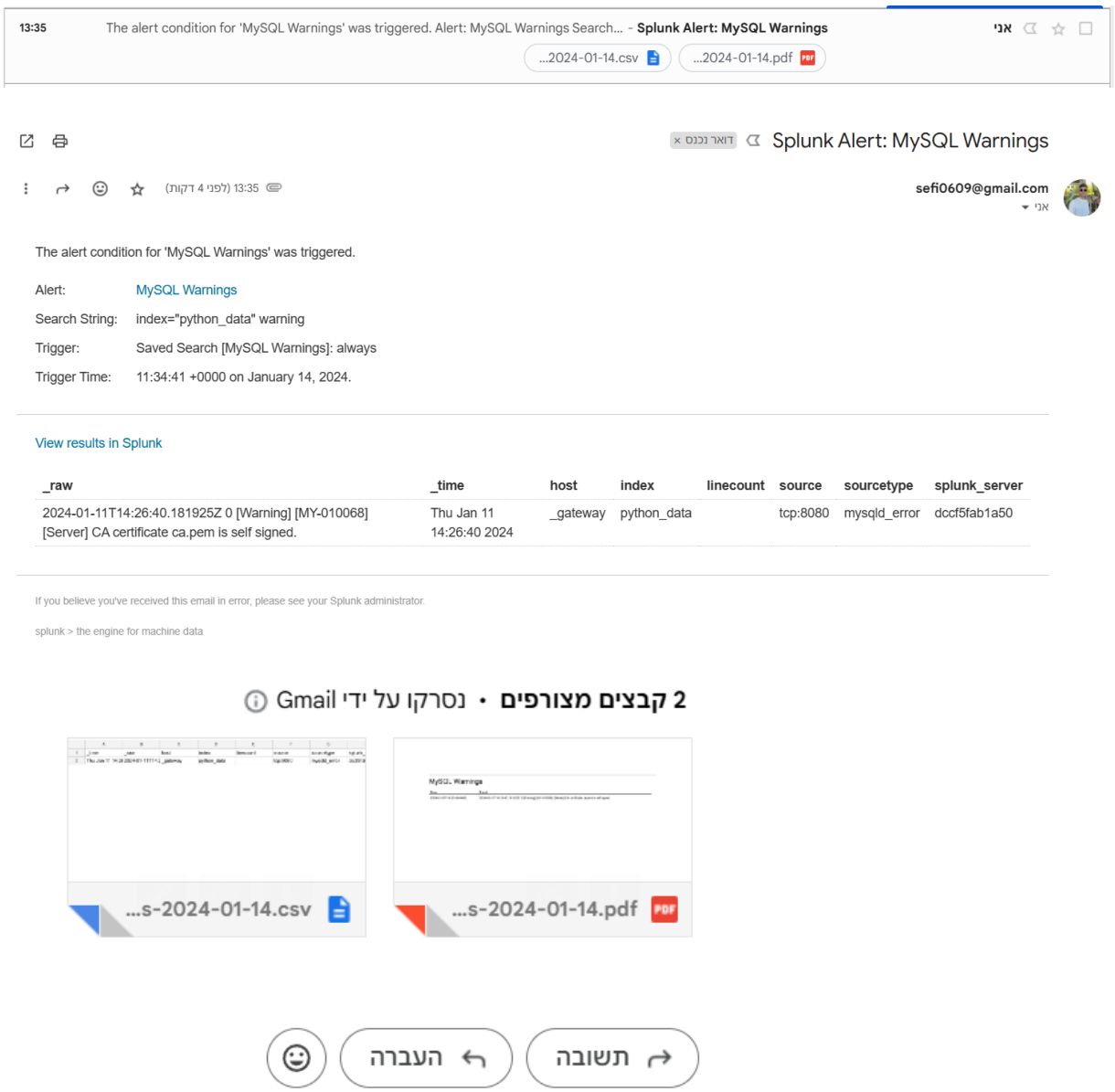
7. Enter a title, click on Real-time and add actions to when the alert is triggered. I've selected send email and add to triggered alerts. In the triggered alert action select the severity of the alert.



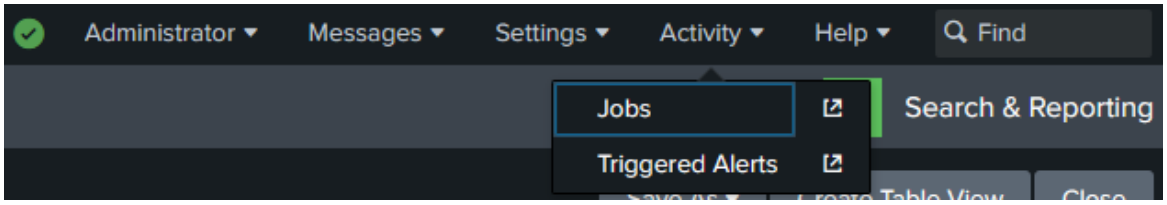
8. In the send email enter the email address you want to alert. Check all the checkboxes for a detailed email about the event.



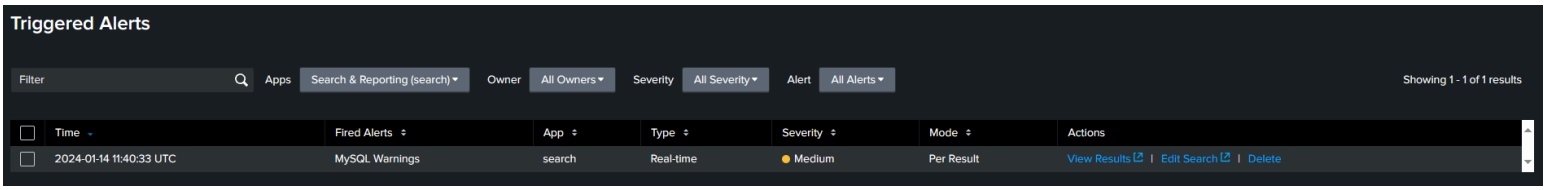
9. Snapshots of the email: (I’ve started the Python container again)



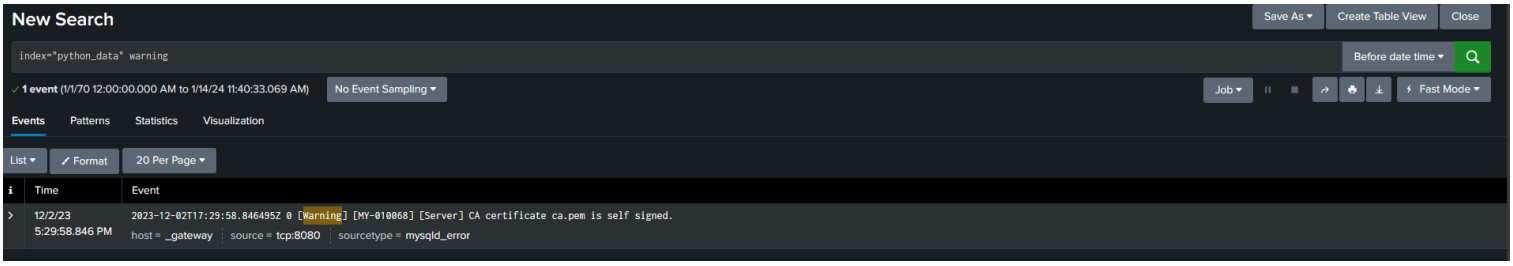
10. On the top right click on the activity option



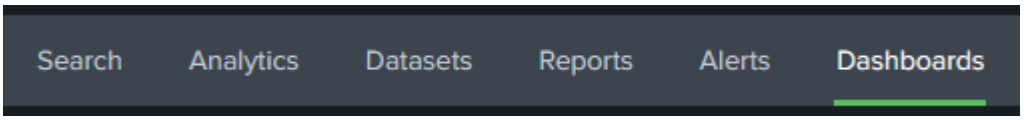
11. Click on the triggered alerts to see if the alert was triggered



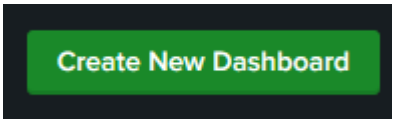
12. You can click on view results for more detail about the event



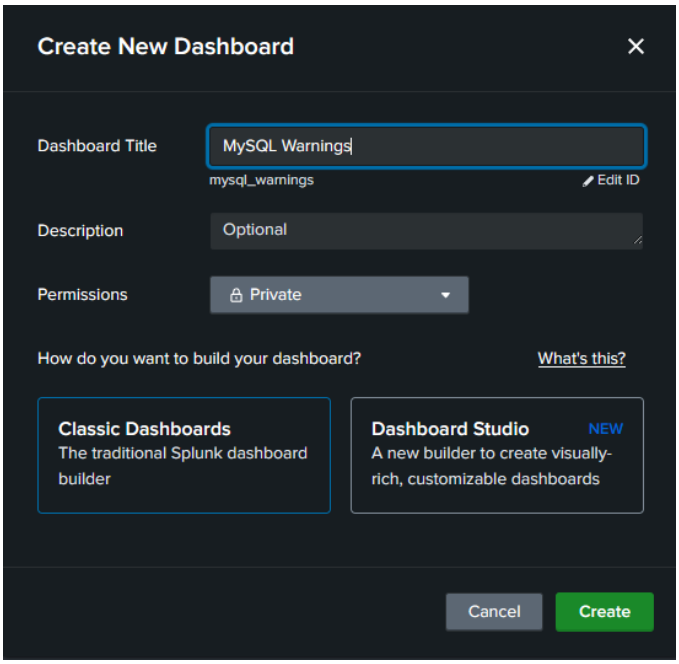
13. For a more real-time experience we will set an auto-refresh dashboard. From the search & reporting app click on dashboards.



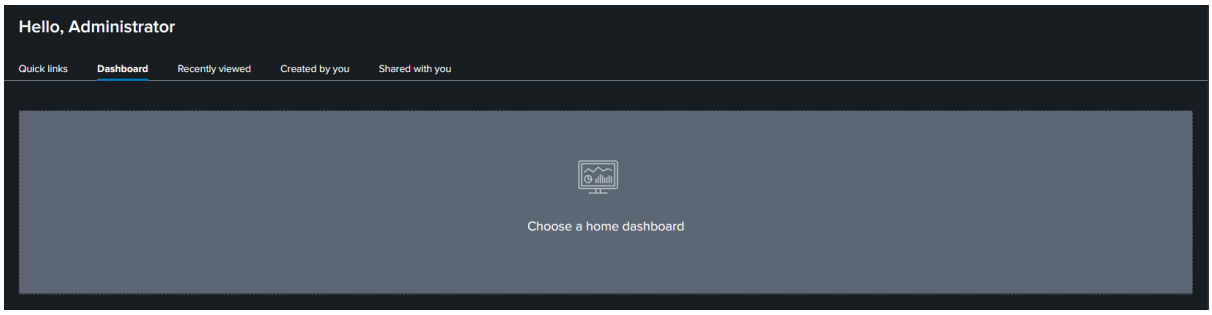
14. Add a new dashboard by clicking the create new dashboard button.

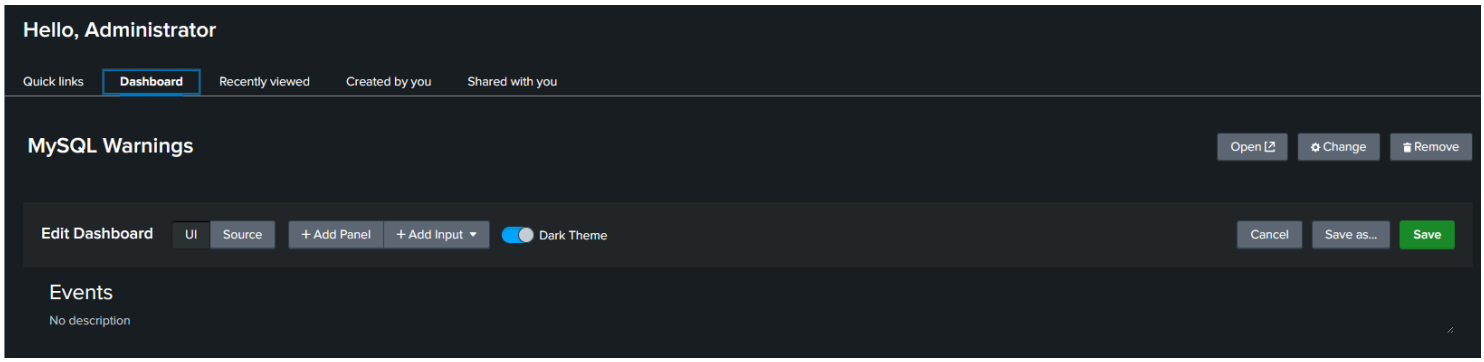


15. Enter a title and press the classic dashboard

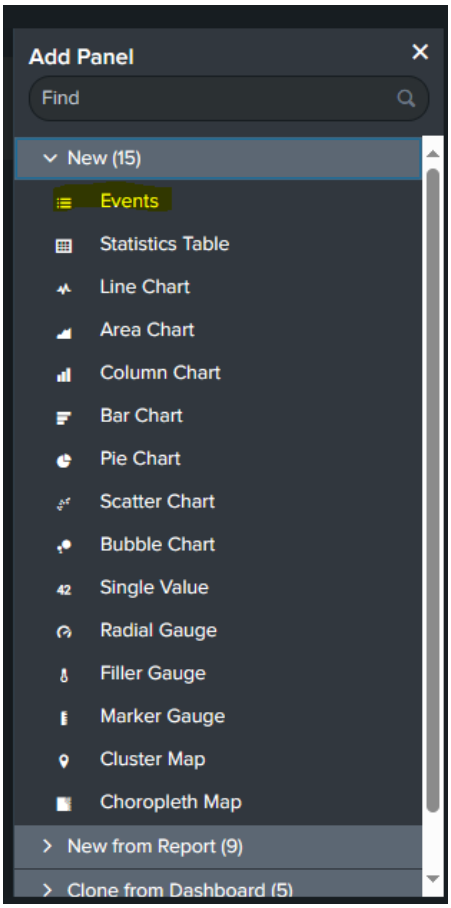


16. After the dashboard is created you can access it from the main page by clicking on the dashboard option. Add your custom new dashboard.

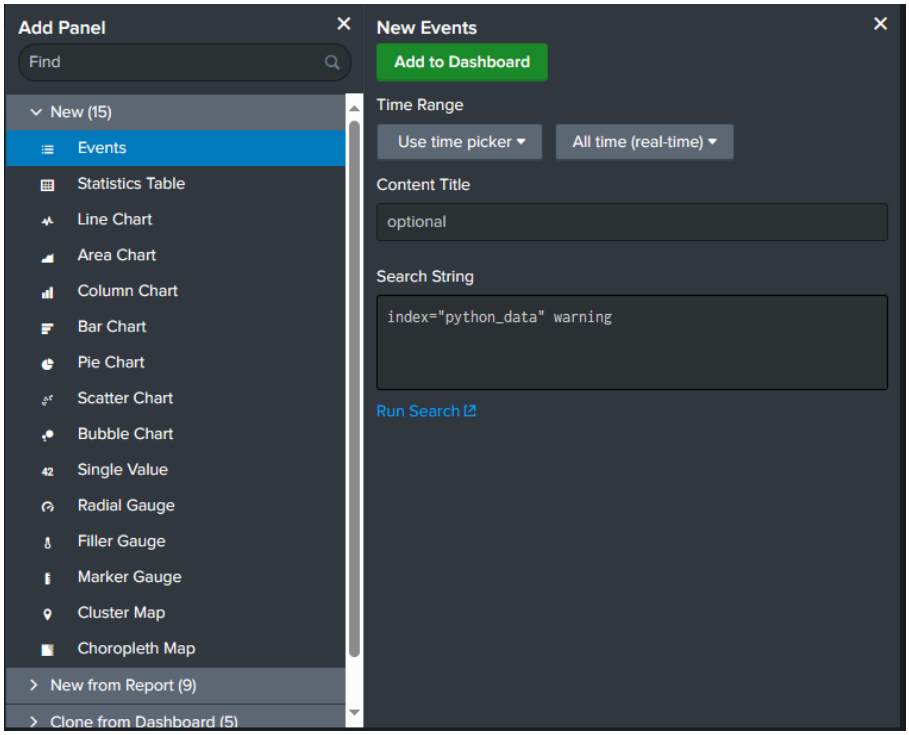




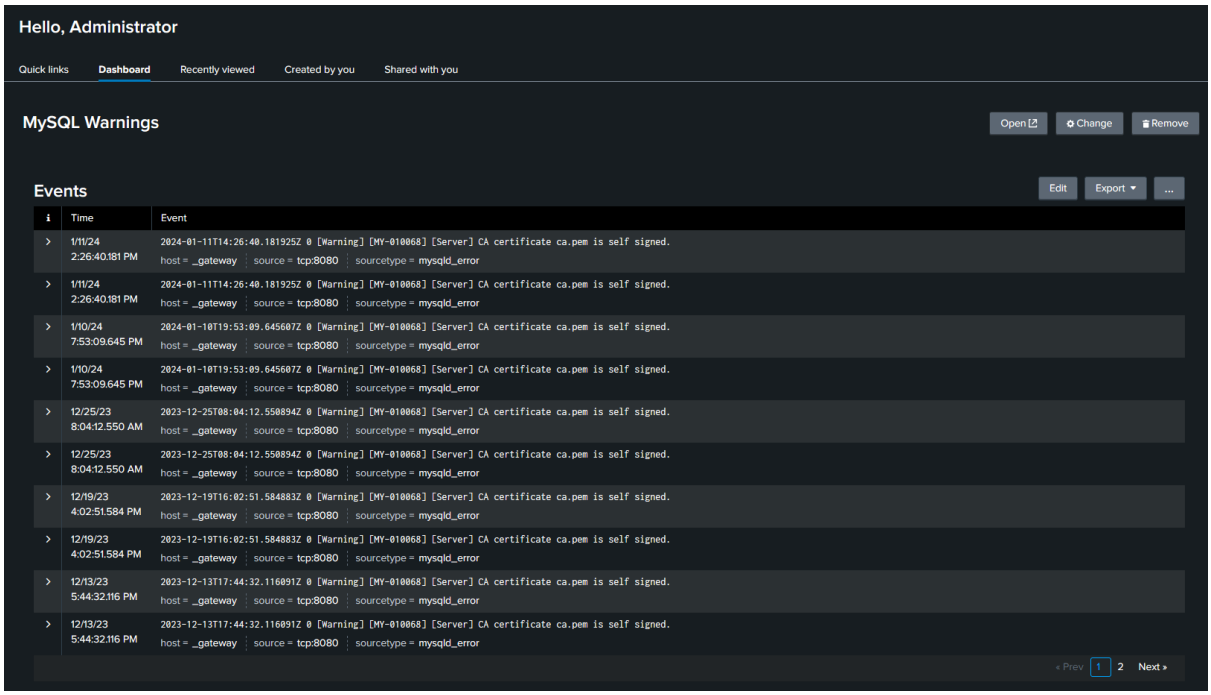
17. Add a new panel to the dashboard. Click on Events



18. Enter your search query and select all time(Real-time) in time range

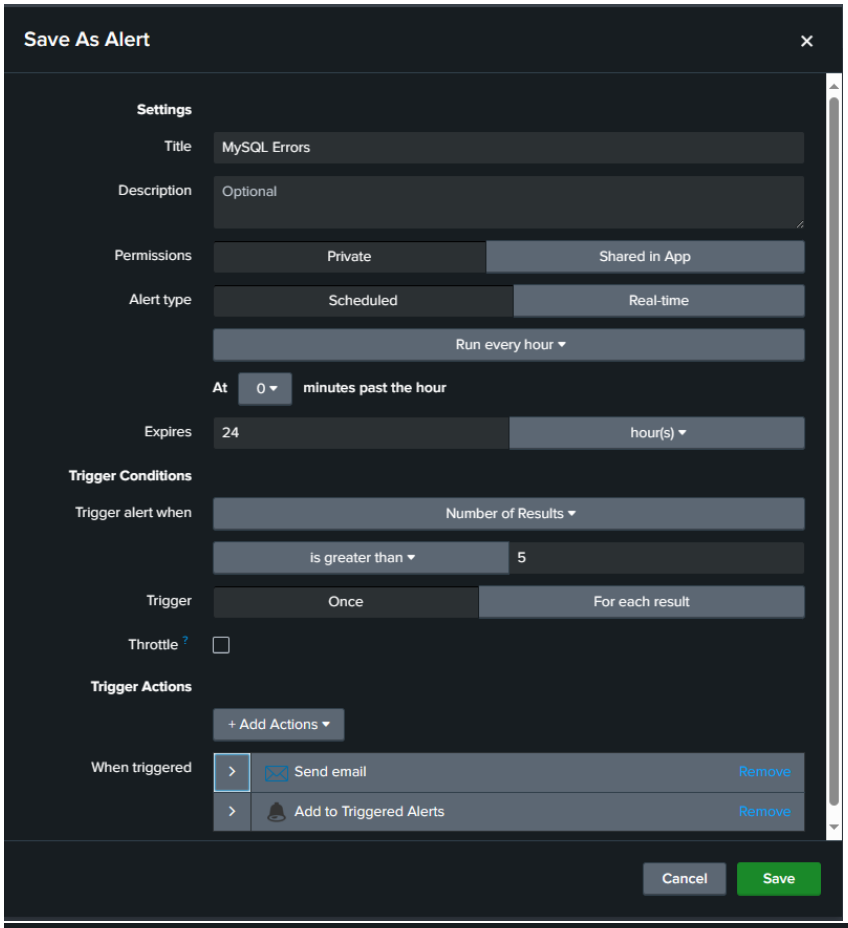


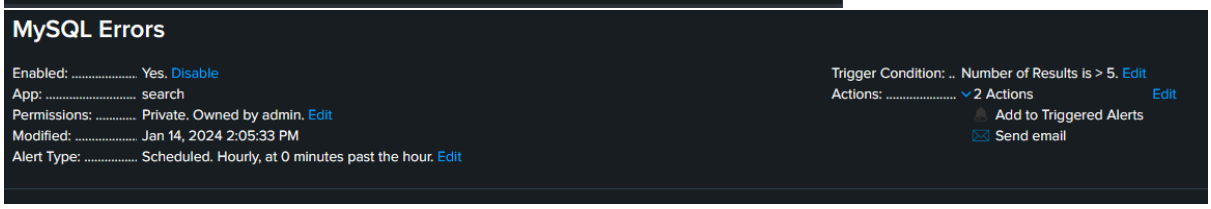
Finally, the dashboard will look like this. When python sends new events (logs) with the word “warning” in them, the dashboard will automatically refresh.



Let's set up an alert for error events.

1. Go to the search app and follow the steps from the previous alert setup. My query is: index="python_data" error
2. Real time alerts are always running in the background. We want to know if there are more than 5 errors in the last hour. We want to avoid utilizing the machine memory unless absolutely necessary, so we can set this alert to run every hour. Set the trigger to greater than 5. Add the two actions from the last alert.





The process will run every hour and if there are more than 5 error it will record on the triggered alert page and we will get an email as shown in the previous alert setup.

Resources:

Create real-time alerts - Splunk Documentation

Email notification action - Splunk Documentation

Create detectors to trigger alerts — Splunk Observability Cloud documentation