6/5/2025

Scheduling reports and alerts in Splunk: A Practical Guide

RESEARCH REPORT

Balaji Varaprasad
CYBERSAPIENS

Scheduled Reports and Alerts in Splunk

In Splunk, the overall configuration of scheduled reports and alerts revolves around defining a search query, scheduling its execution, and optionally triggering actions based on the results. Everything begins with creating a search using SPL (Search Processing Language), which filters and analyses the relevant data, for example, failed login attempts or error logs. This search can then be saved as a report, which can be scheduled to run automatically at specified intervals such as hourly, daily, or weekly. The report can be configured to output results in various formats like tables or charts, and it can be emailed, downloaded, or added to dashboards for regular monitoring.

Alerts are a specialized form of scheduled search that not only run periodically but also include trigger conditions. When these conditions are met, such as the number of failed logins exceeding a threshold, Splunk can automatically execute predefined actions. These actions include sending email notifications, running scripts, sending data to a webhook, or alerting via Slack or other integrations. Users can also configure alert throttling or suppression settings to avoid repeated alerts for the same issue. Overall, this configuration ensures that important events in your data are not only captured and summarized but also responded to in a timely and automated manner.

Types of Data Sources Monitored by Scheduled Reports and Alerts in Splunk:

In Splunk, the data sources monitored or analysed by scheduled reports and alerts come from a wide variety of machine-generated data collected across systems, applications, and networks. These sources typically include:

1. **System Logs**

   - Operating system logs (e.g., /var/log/ in Linux, Windows Event Logs)

- Tracks login attempts, system errors, shutdowns, and user activity

2. **Security Logs**

   - Firewall logs, antivirus logs, intrusion detection/prevention systems (IDS/IPS)
   - Useful for detecting threats, failed login attempts, port scans, and policy violations

3. **Application Logs**

   - Logs generated by custom or third-party applications (e.g., Apache, Nginx, MySQL)
   - Monitor error rates, usage patterns, performance issues, and crashes

4. **Network Traffic**

   - Data from routers, switches, and network monitoring tools (e.g., NetFlow, SNMP)
   - Helps analyze bandwidth usage, detect anomalies, or identify malicious traffic

5. **Authentication and Access Logs**

   - Logs from Active Directory, LDAP, Okta, or other identity systems
   - Track who accessed what, when, and from where, key for access control monitoring

6. **Cloud Services Logs**

   - Logs from AWS CloudTrail, Azure Monitor, Google Cloud Logging, etc.
   - Provide insights into cloud resource access, configuration changes, and security events

7. **Web Server Logs**

   - Access logs, error logs from web servers like Apache, IIS, or Nginx
   - Monitor website traffic, detect attacks (e.g., SQL injection), or troubleshoot issues

Scheduled reports and alerts in Splunk typically monitor security events, system performance, user activities, and application behaviour across on-premise, cloud, and hybrid environments. The specific data sources depend on what has been onboarded into Splunk, but the goal is always to identify patterns, detect anomalies, and automate response actions where needed.

Data Sources Monitored in our case:

The primary data source for this use case is, index=_audit: This internal index logs Splunk's authentication attempts, including failed logins.

This data source is essential for tracking authentication behaviour and potential brute-force or unauthorized access attempts within Splunk itself.

Practical: Scheduling reports and creating alerts

Task: Set up a scheduled report that provides a summary of failed login attempts over the last 24 hours and configure an alert to trigger whenever the number of failed login attempts exceeds a predefined threshold. (index=_audit failedlogin).

Failed Login Query to Create a scheduled report on failed logins in 24 hours:

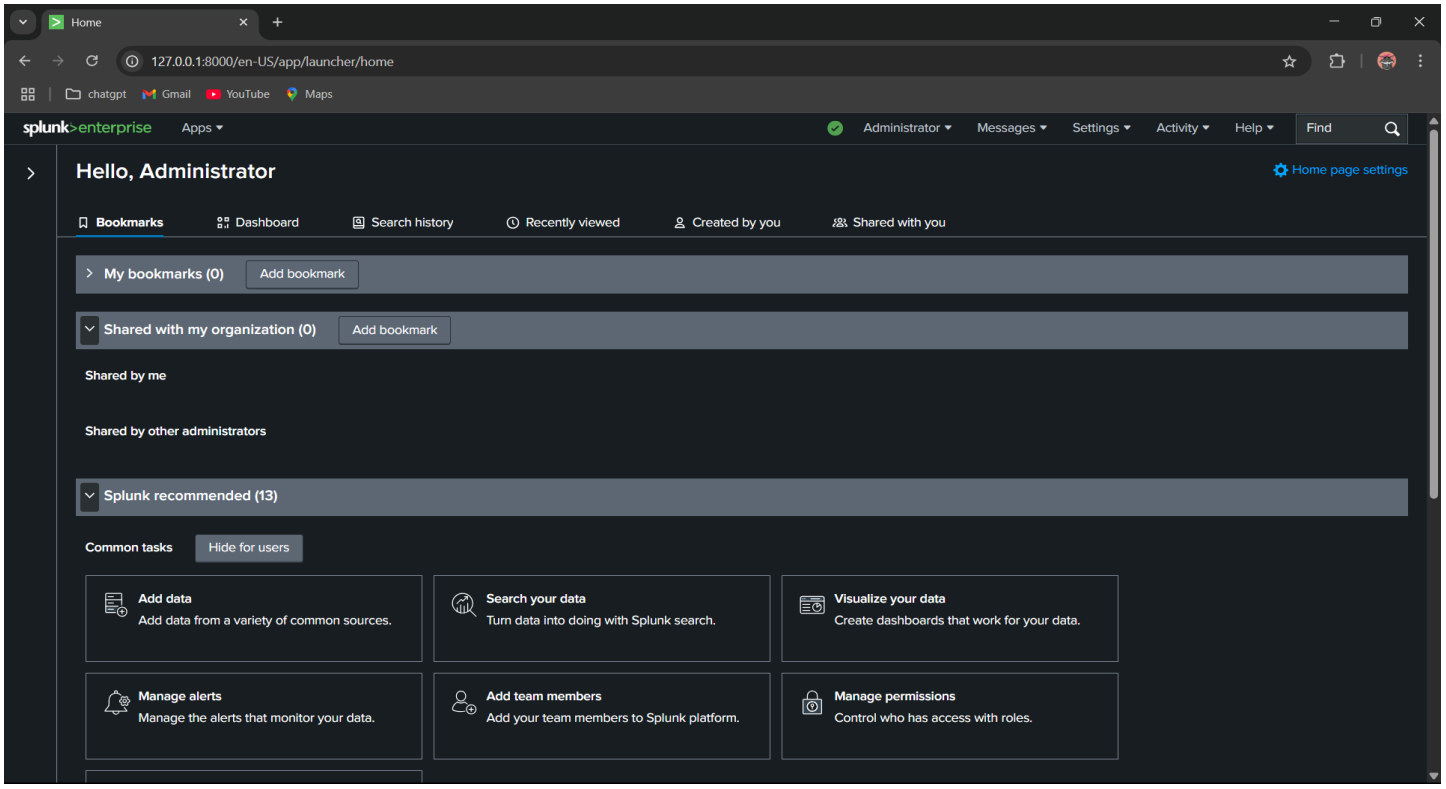
Splunk's internal audit logs (stored in index _audit) record web interface logins and other admin actions. Failed login attempts are captured with fields like action="login attempt" and an info=failed flag. **For example:**

```
index=_audit action="login attempt" info="failed"| stats count by user, _time
```

This query (with a time range of the last 24 hours) returns the count of failed Splunk Web logins per user. You can further refine or group by user if needed. As Splunk docs note, you should search _audit to review login events.

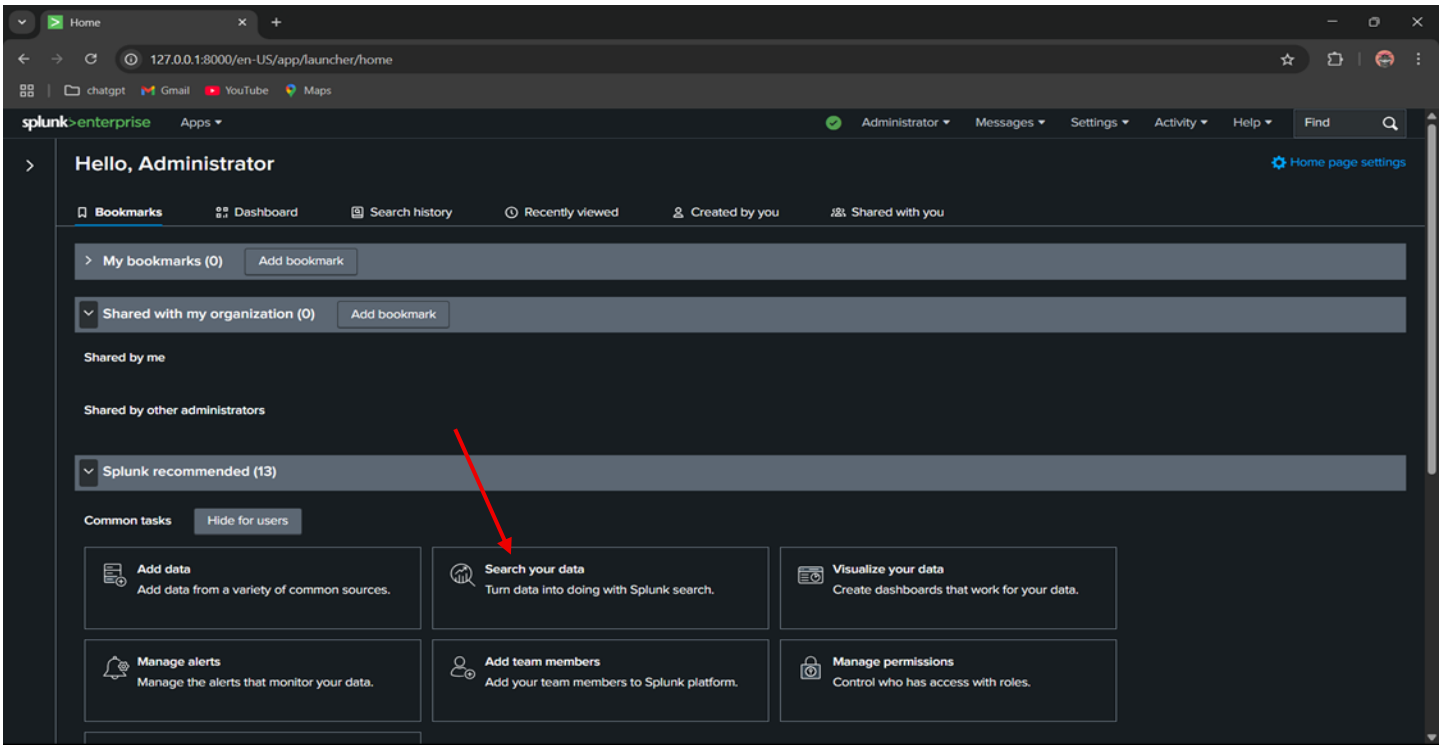
Step 1:

Open the Splunk Enterprise application and log in using your administrative credentials to begin the practical configuration process.

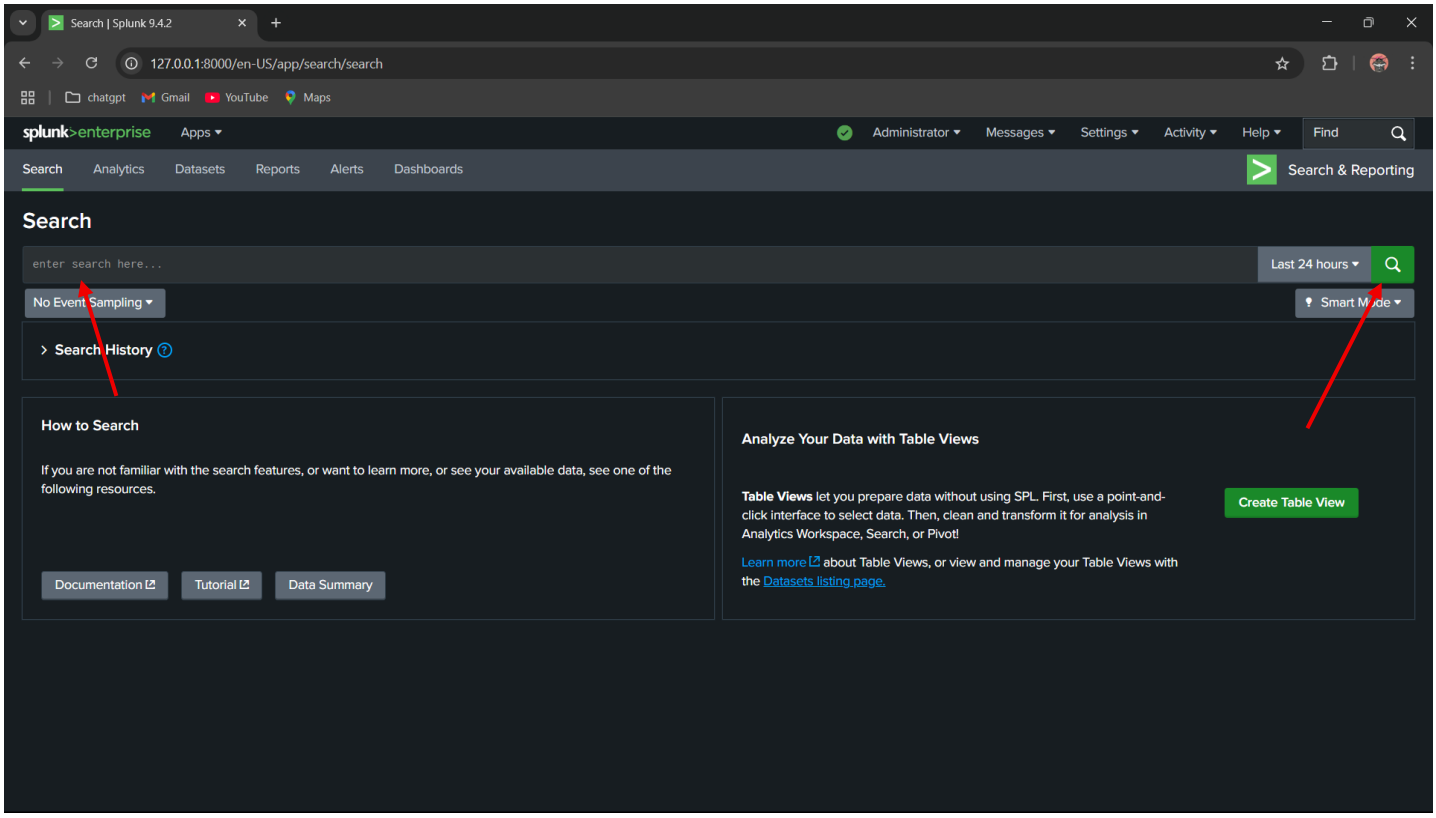


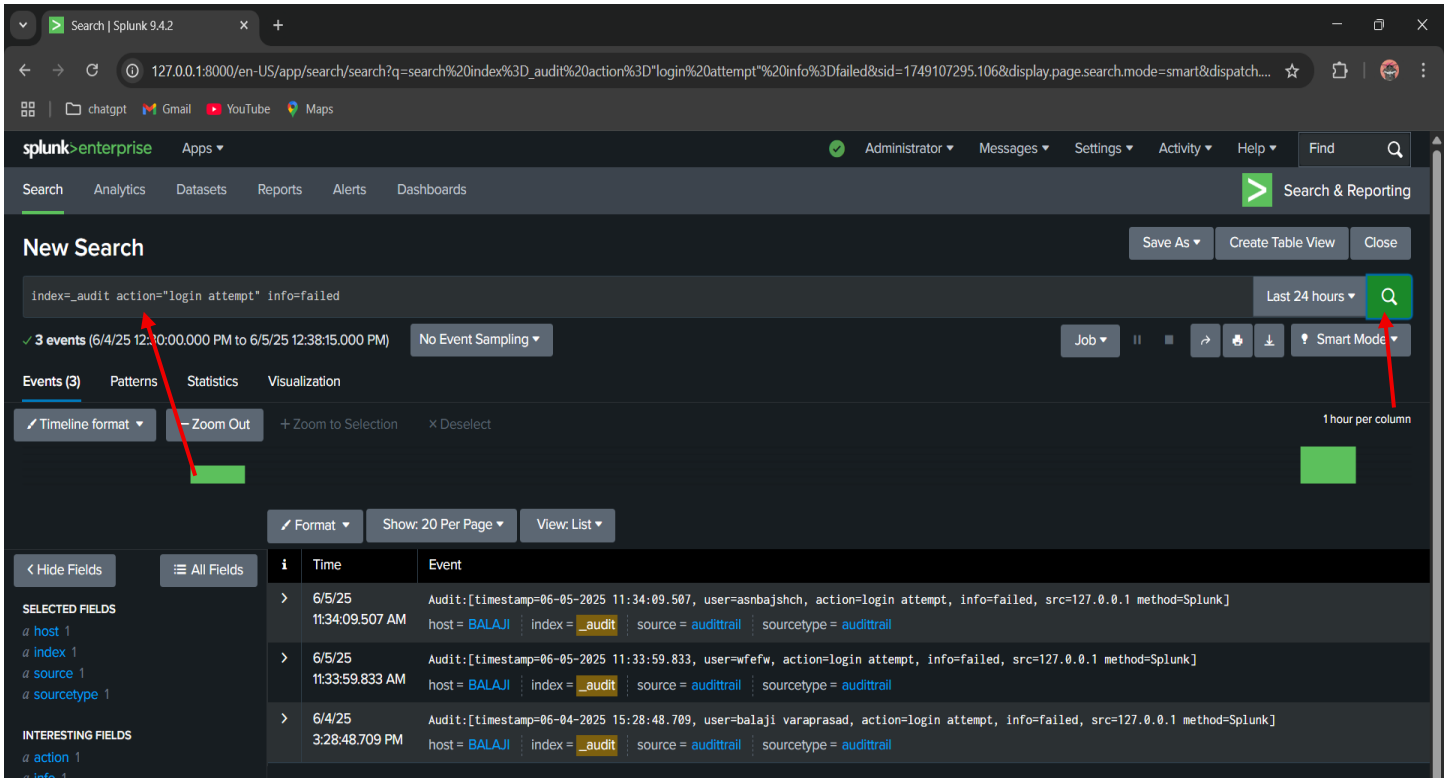
Step 2:

- Navigate to the Search your data. Set the time range filter to Last 24 hours, then enter your SPL (Search Processing Language) query to extract failed login attempts from the internal audit logs.



- index=_audit action="login attempt" info=failed
- This retrieves all failed login attempts recorded in the last 24 hours.





Explanation:

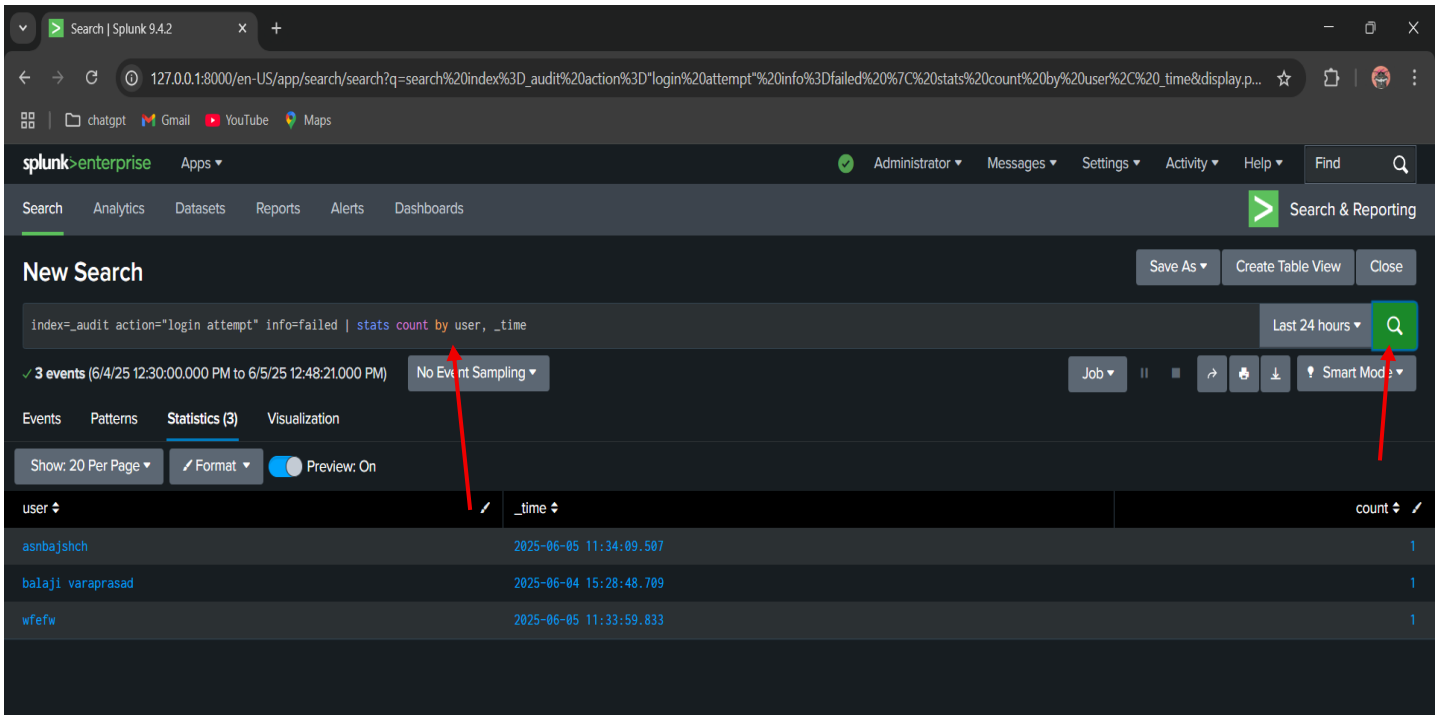
- **index=_audit**
  This specifies that the search should be performed within the **_audit** index. The _audit index in Splunk stores logs related to user activity, authentication, and system-level operations. It is commonly used for auditing and monitoring administrative actions, including logins.

- **action="loginattempt"**
  This filter narrows down the search to events where a user tried to log in. The action field captures the type of event that occurred, in this case, a login attempt.

- **info=failed**
  This condition further filters the results to only include **failed** login attempts. The info field indicates the outcome of the event, and failed shows that the login was not successful.

Step 3:

- Now, let's further filter and organize the data to make it more insightful. We will extract and display only the username, along with the count of failed login attempts and the corresponding time.
- Add, | stats count by user, _time spl query to before query to filter our data.



Explanation:

- stats count by user, _time: This command groups the results by **user** and **timestamp**, and counts the number of failed login attempts per user at specific times.
- This helps in identifying which users are experiencing repeated login failures and when they occurred.

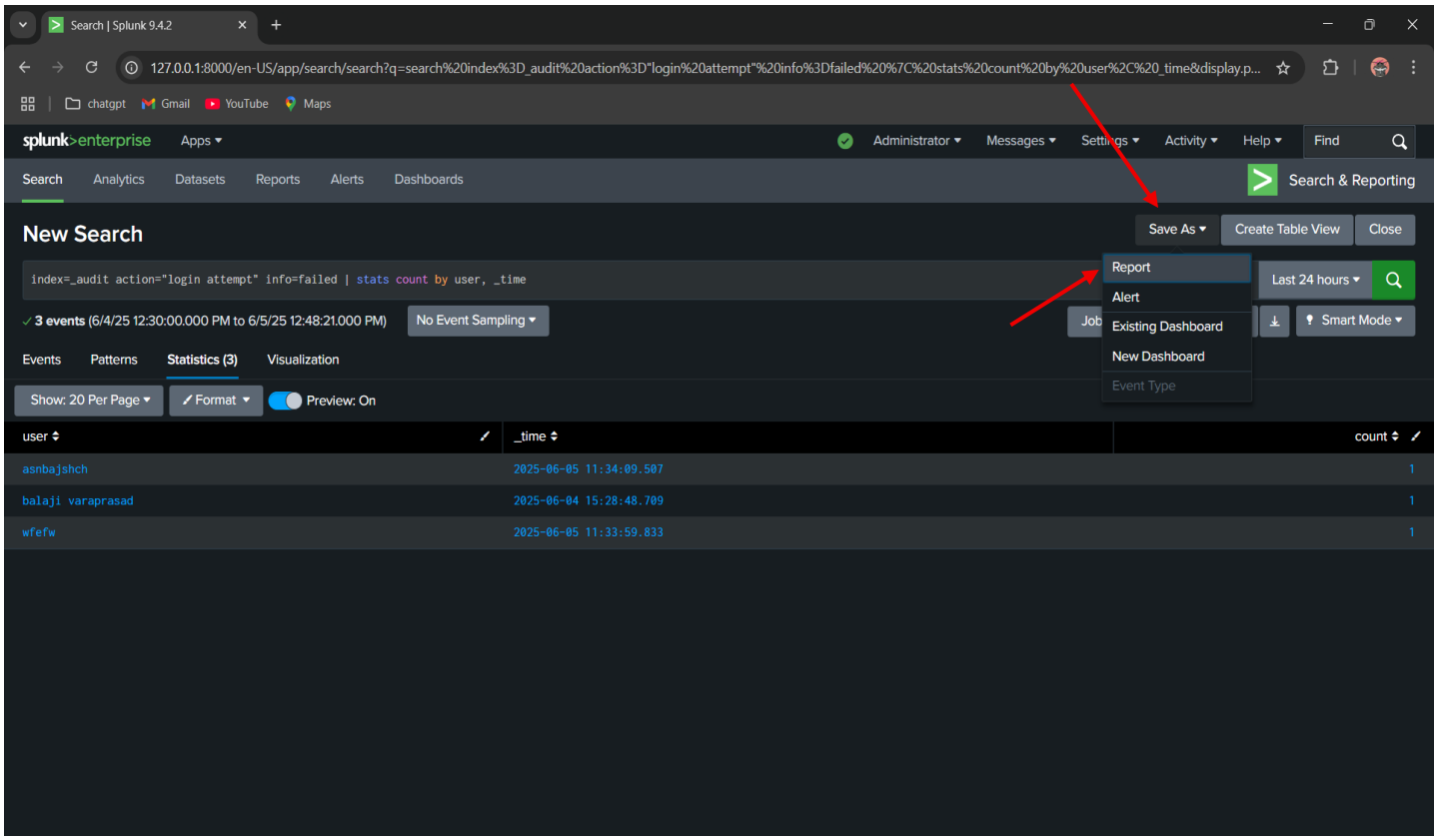
Step 4:

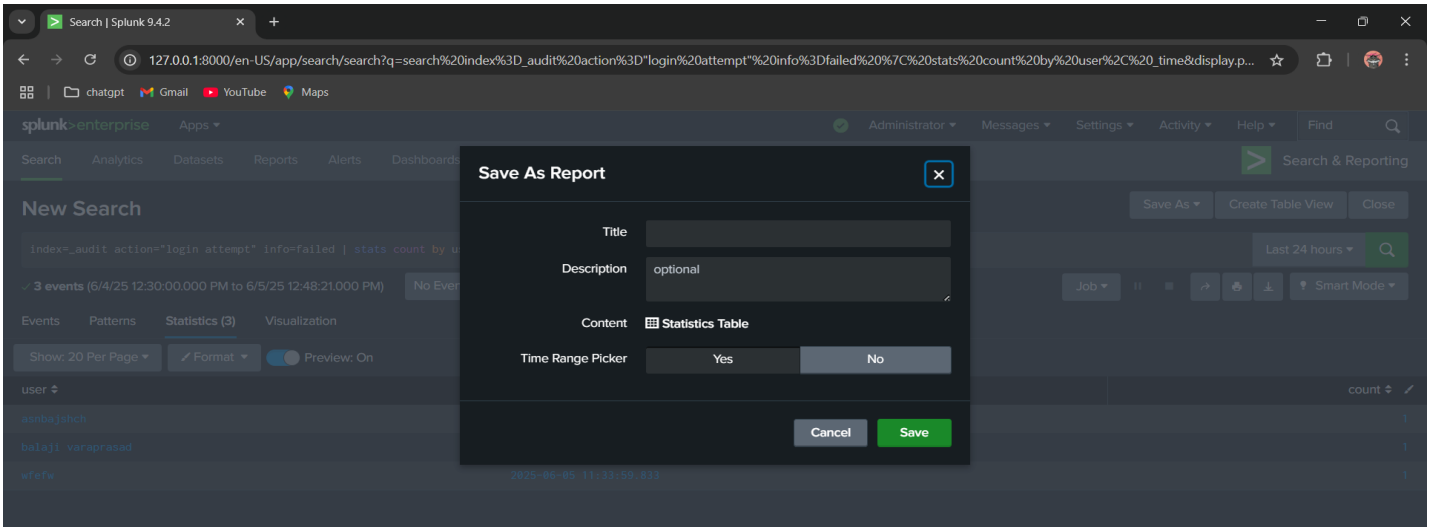
- Once the desired results are generated, proceed to save the search as a report in order to schedule it for regular execution.

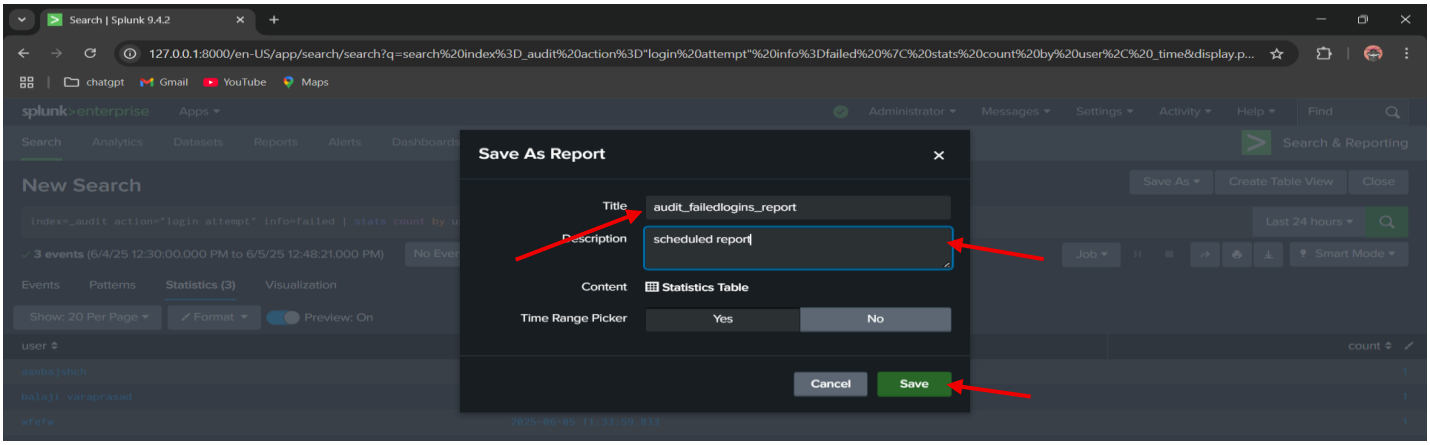
To do this:

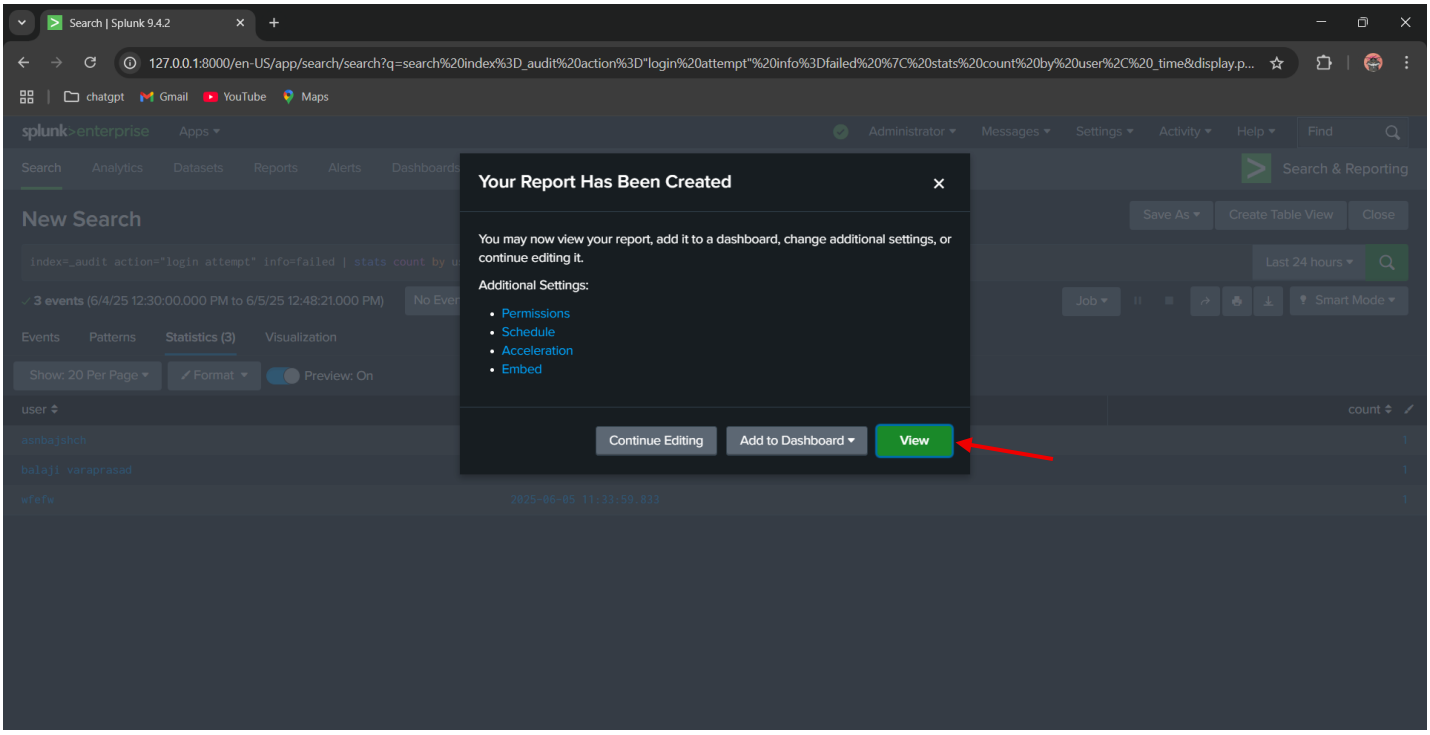
- Click on Save As → Report in the top right corner of the search window.
- Provide a suitable Report Title and optional Description.
- Set the report permissions if needed (private or shared).

Note: *If you are unfamiliar with the steps to save a report in Splunk, you can refer to my previous research report published on LinkedIn for detailed guidance.*





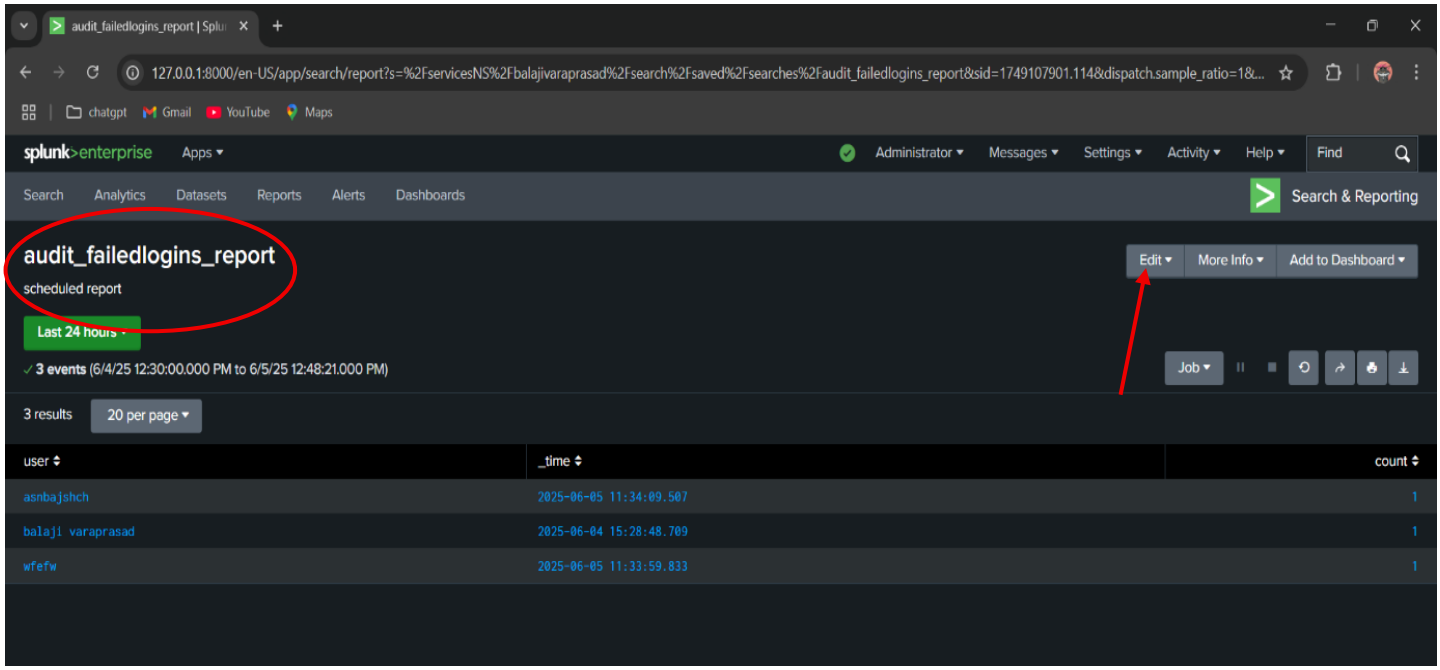


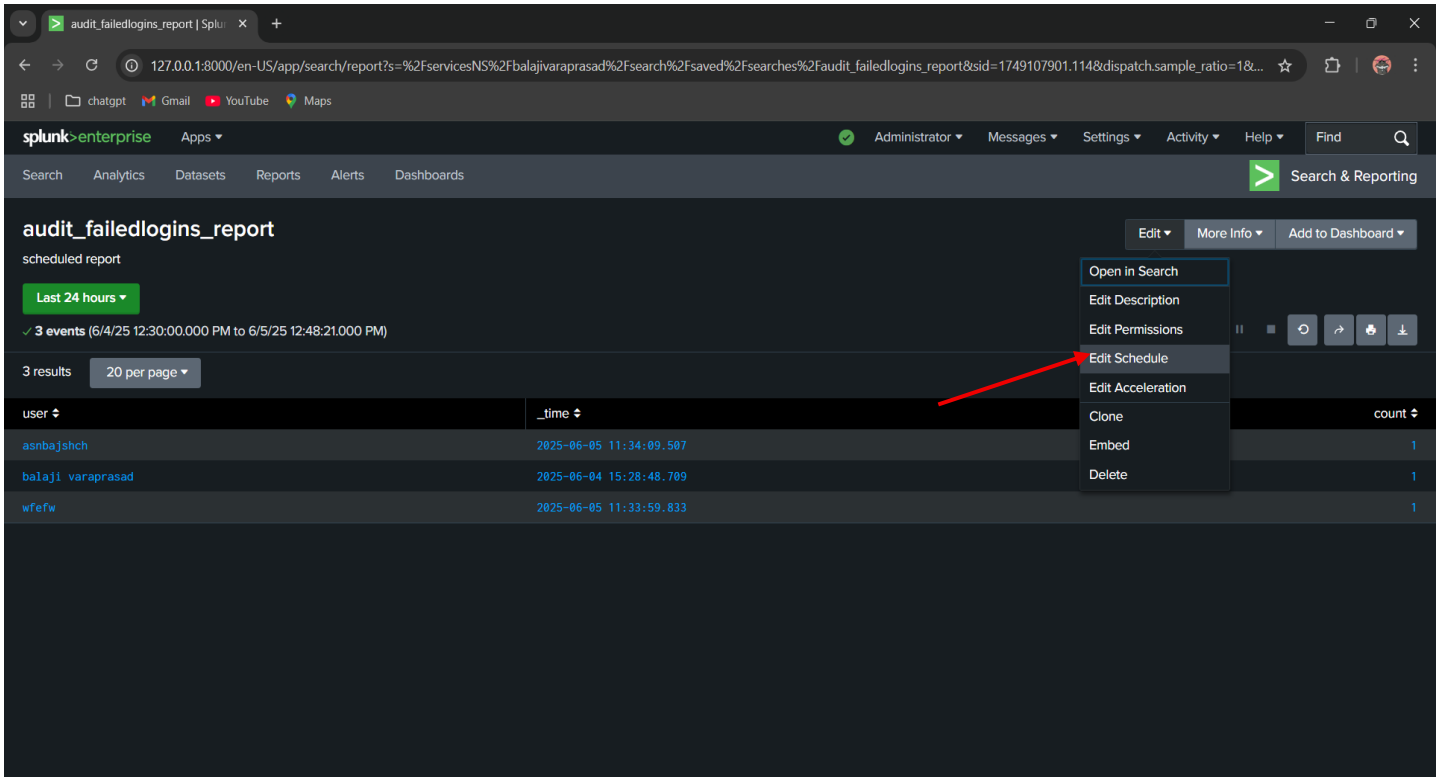


- After creating click on view to open the created report.

Step 5:

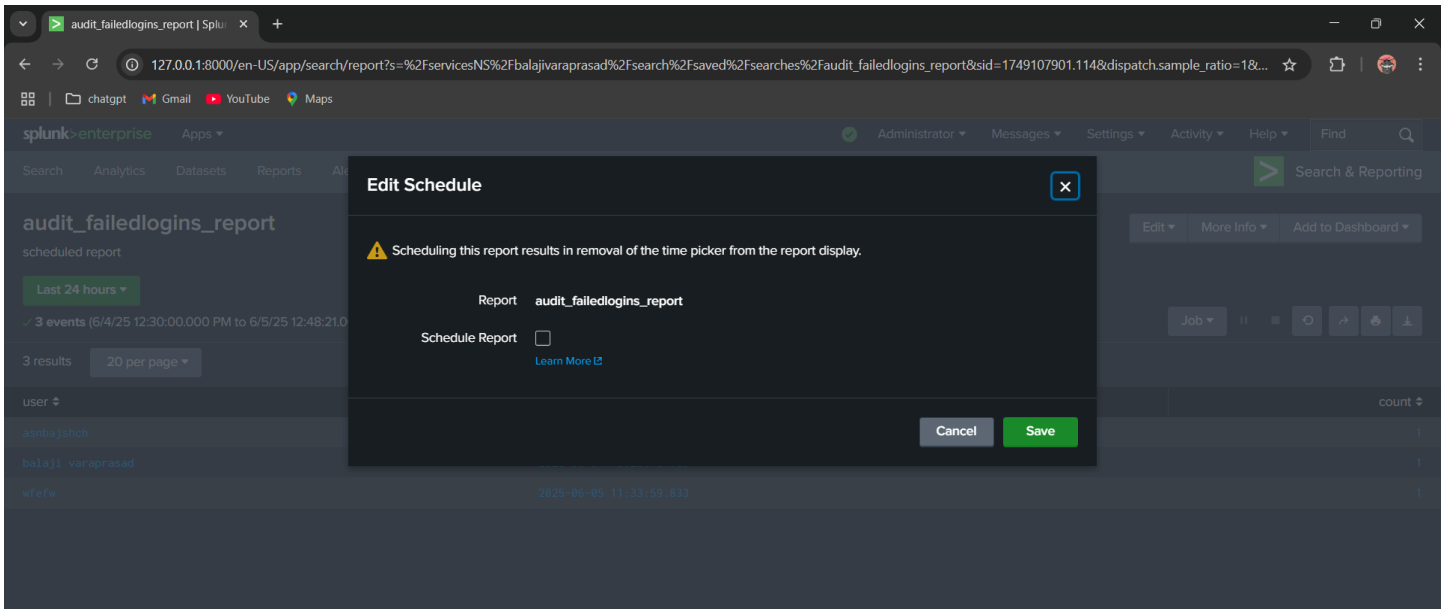
- After opening the report you just created, click on the **Edit** option located at the top-right corner of the report interface. From the dropdown, select **Edit Schedule** to configure the scheduling settings.
- This allows you to automate the report to run at specified intervals (e.g., daily, hourly, or using a custom cron expression), ensuring regular monitoring without manual execution.





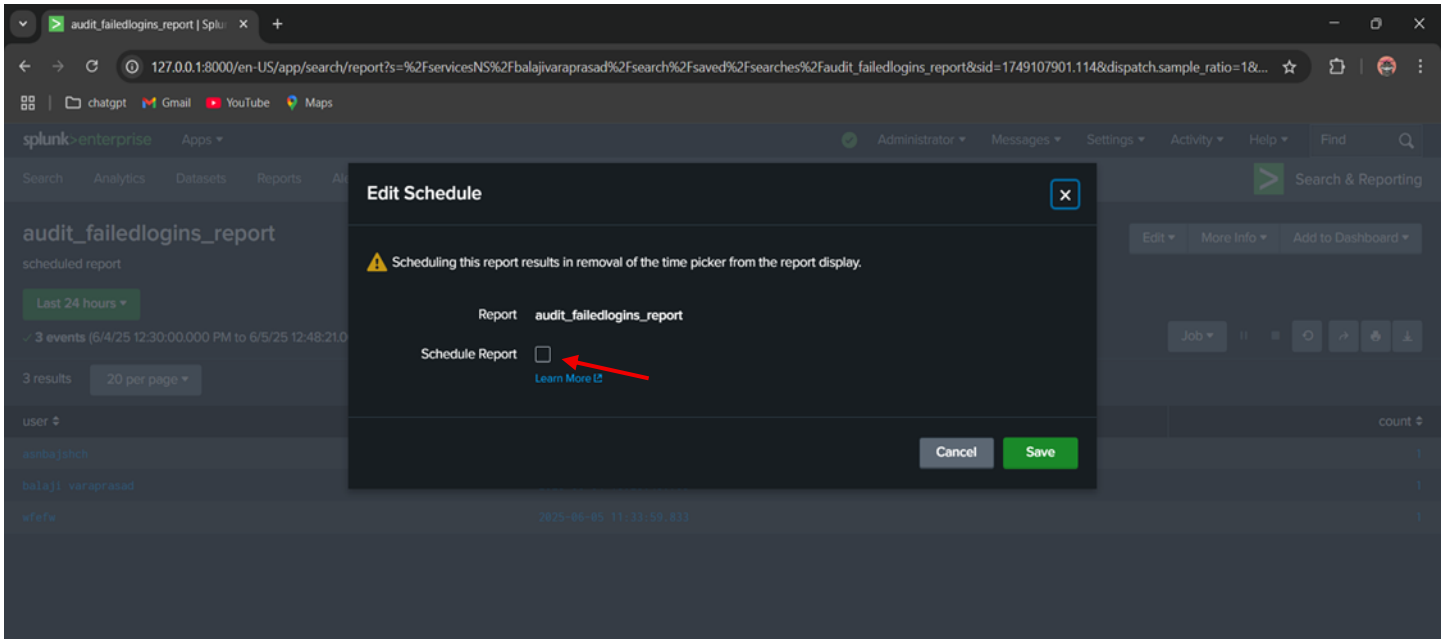
Step 6:

After clicking on Edit Schedule, a pop-up window will appear where you can define the scheduling parameters for the report.



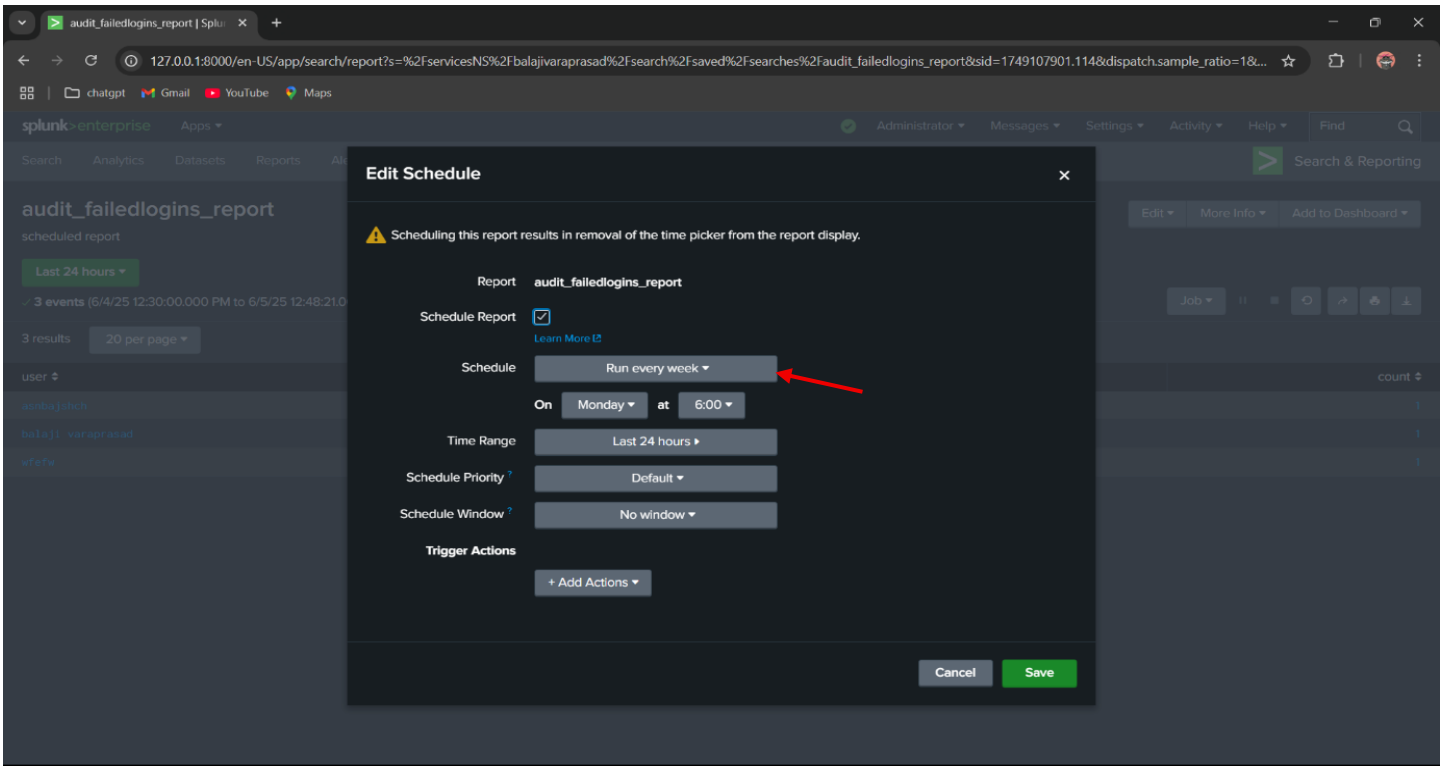
Step 7:

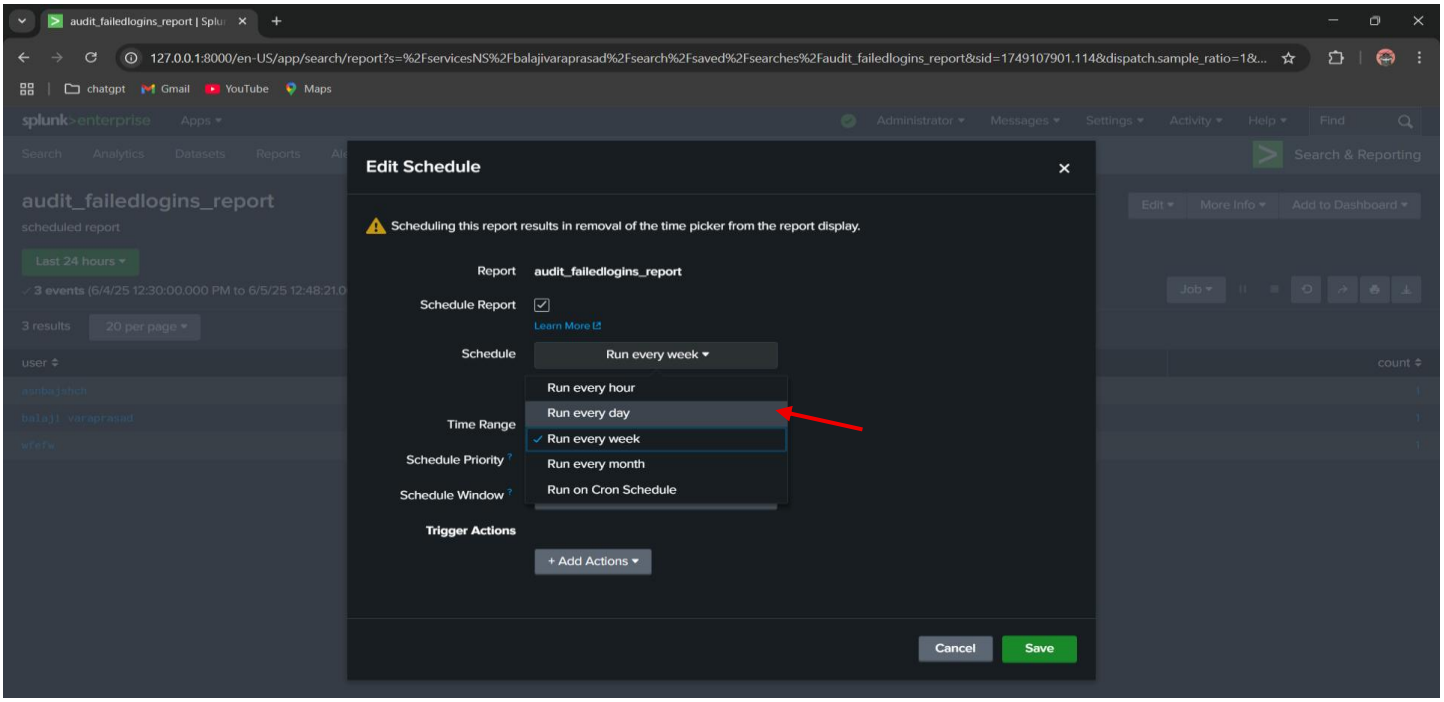
- In the schedule configuration pop-up, click on the checkbox labeled "Schedule Report" to enable the scheduling settings.
- Once the checkbox is selected, the scheduling fields become editable, allowing you to define when and how often the report should run. This step is essential to activate the automated scheduling feature for your report.



Step 8:

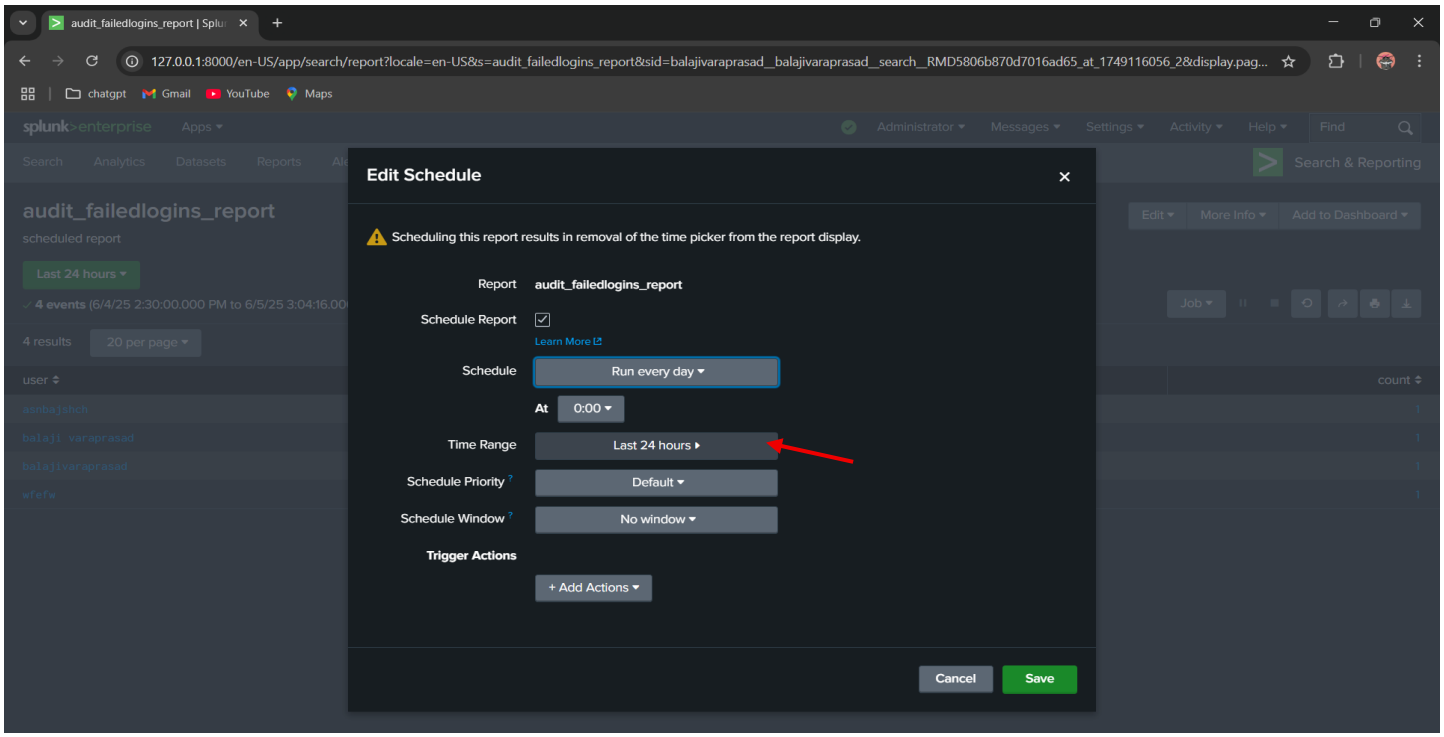
- In the same pop-up window, locate the dropdown menu next to the Schedule label. Click on it to open the options for how frequently the report should run.
- For this example, I selected Daily to ensure the report runs once every day.
- You can select the frequency based on your report's purpose and monitoring needs.

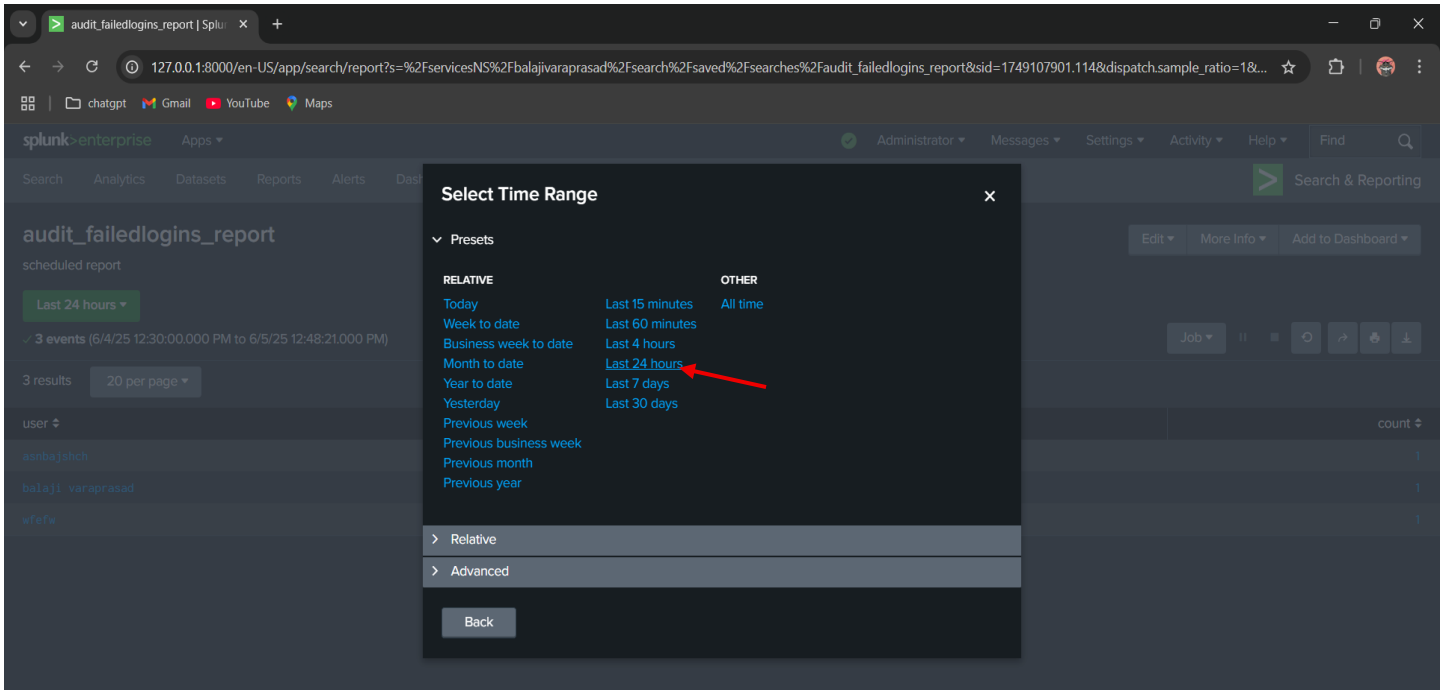




Step 9:

- After selecting the report frequency, configure the Time Range for which the report should retrieve data during each scheduled run.
- For this case, select "Last 24 hours" to ensure the report captures all failed login attempts from the previous day every time it runs.





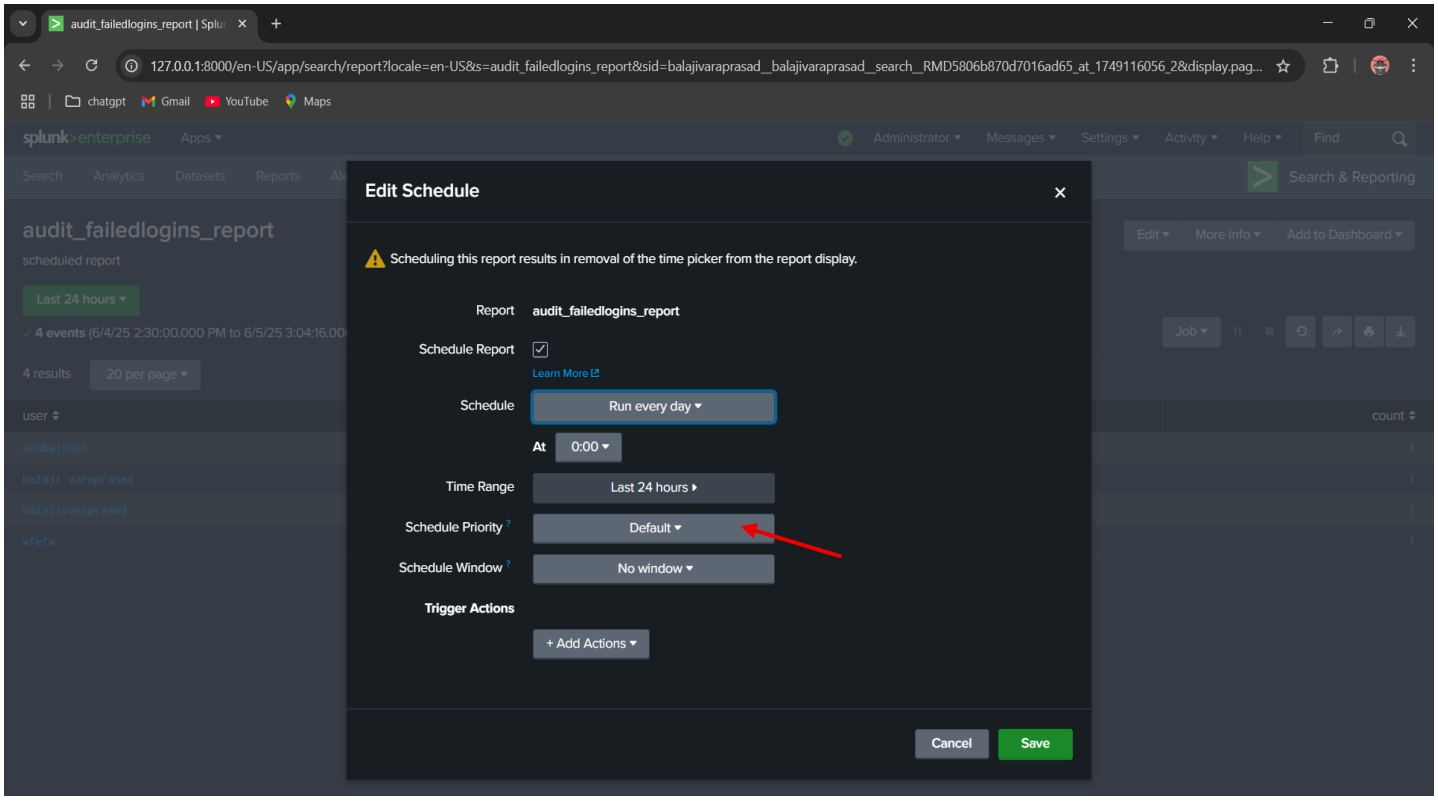
Step 10:

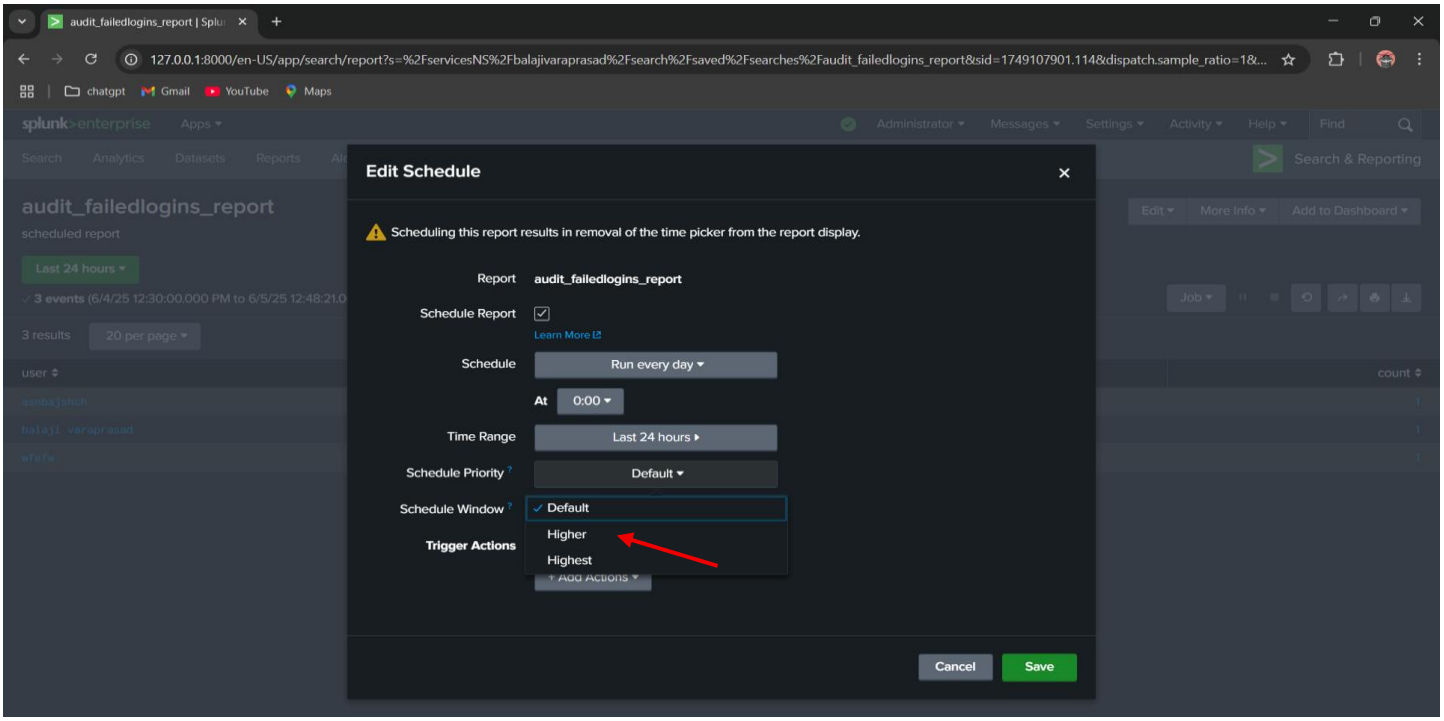
- Once the time range is configured, proceed to set the Schedule Priority for the report.
- In the scheduling window, you'll find a Priority dropdown menu. This determines the execution priority of the report in relation to other scheduled searches in the system.

Options typically include:

- Default
- Higher
- Lower

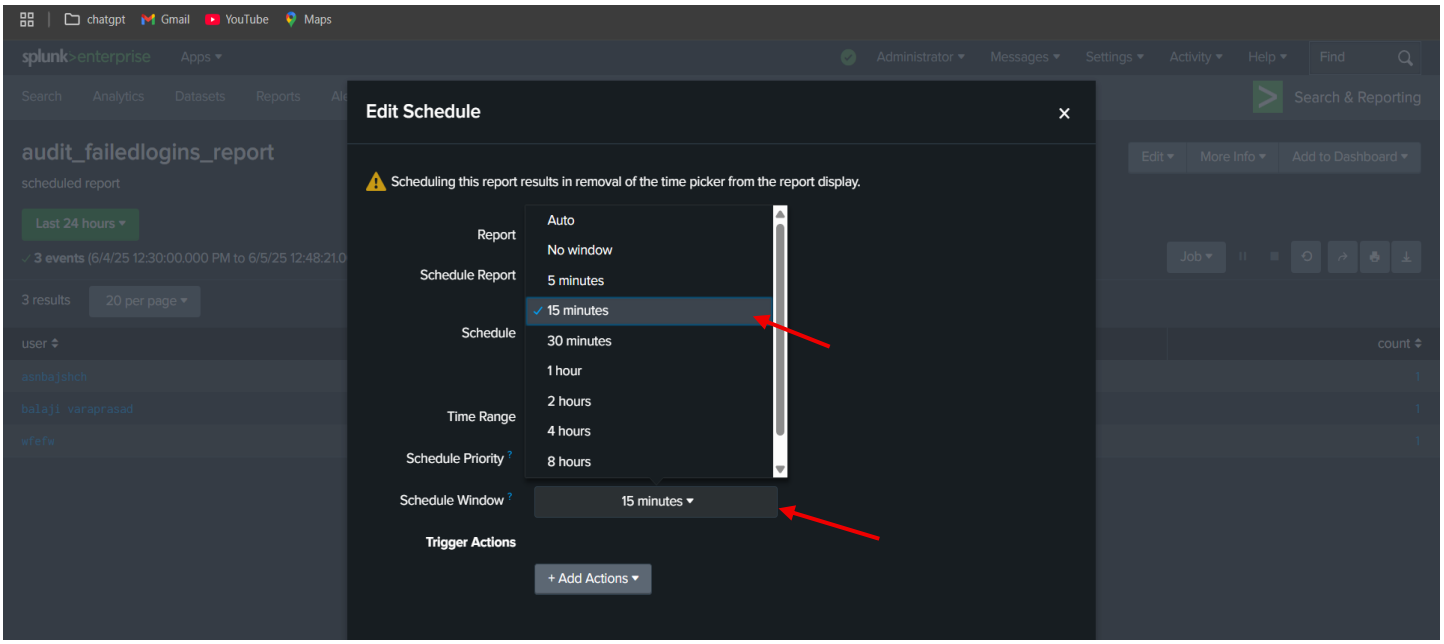
- For most use cases, selecting Default is sufficient. However, if the report is critical and needs to run ahead of others, you may choose Higher priority.
- In this example, I selected High Priority to ensure the report runs with precedence over lower-priority scheduled searches.





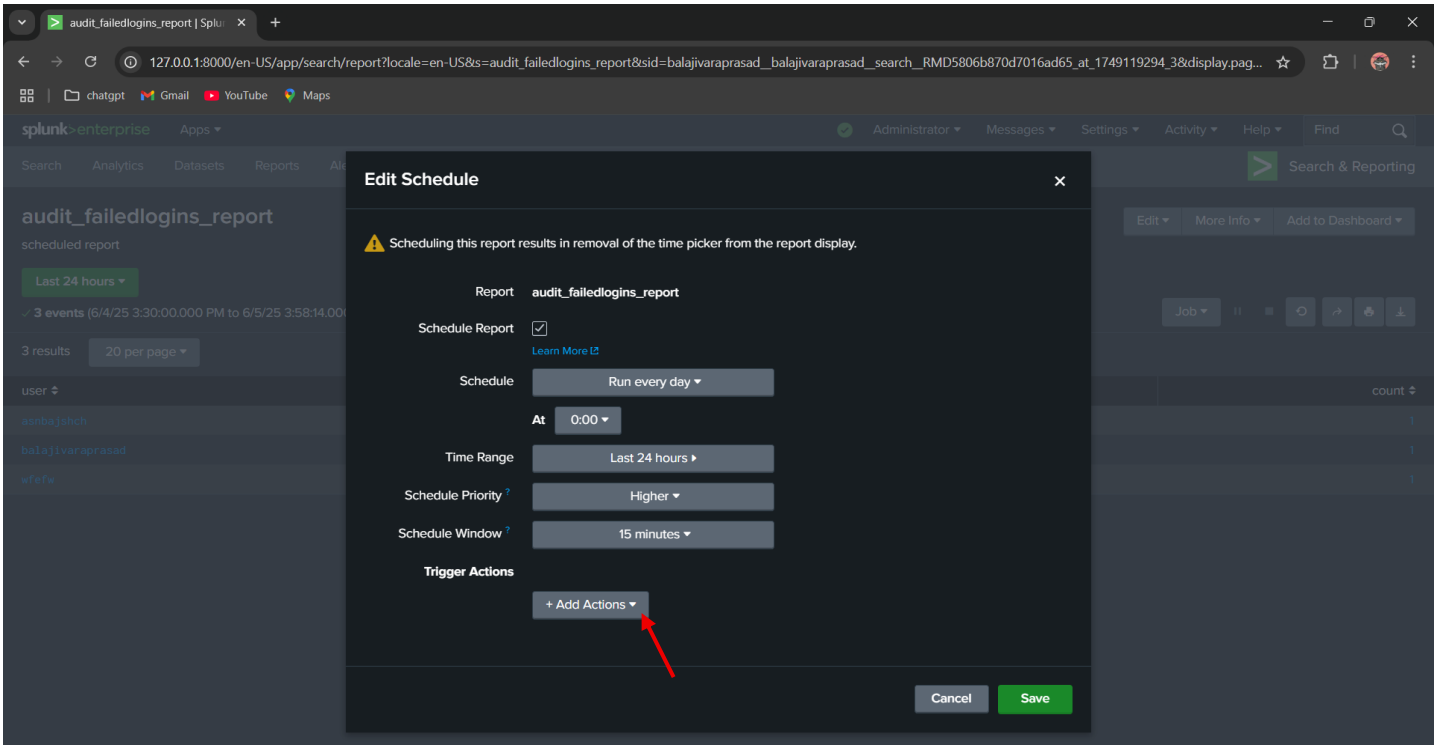
Step 11:

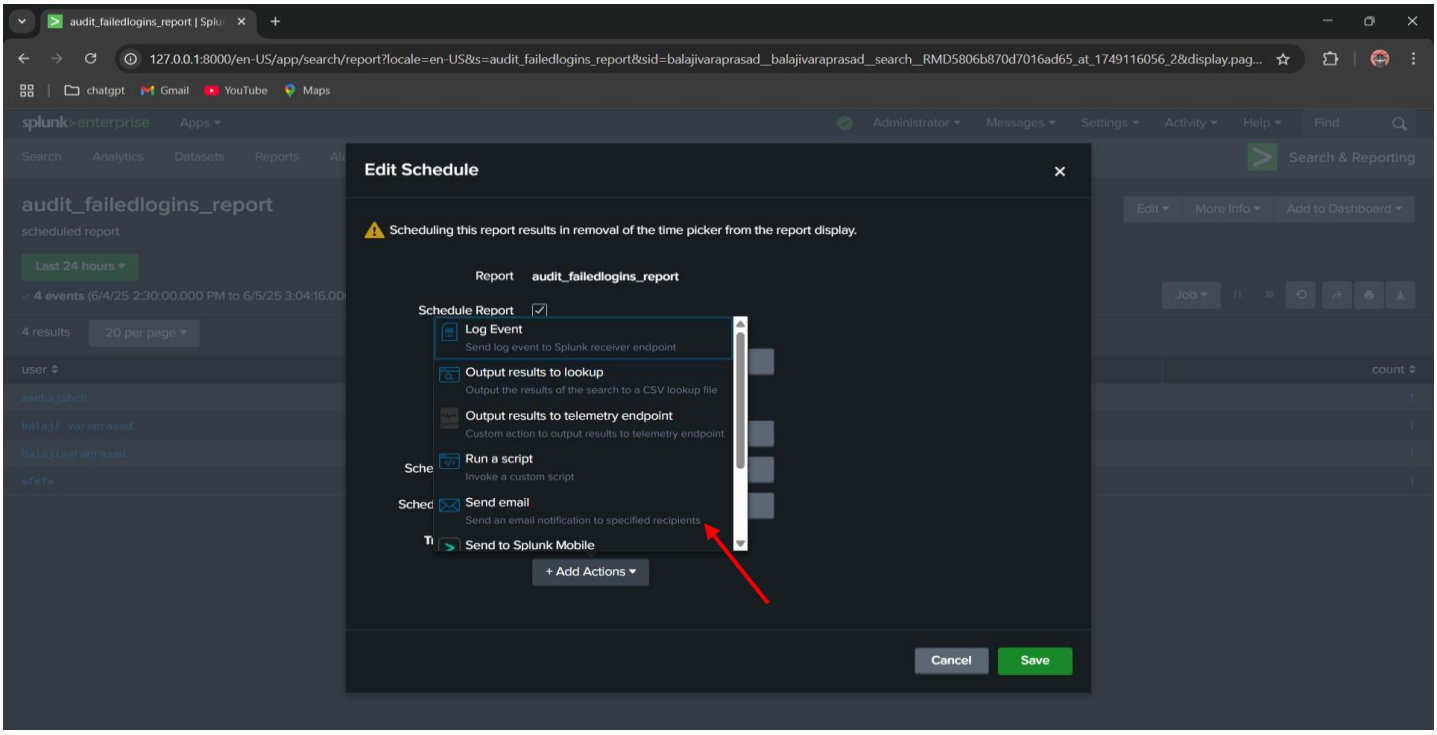
- Next, set the Schedule Window, which defines the allowed time range during which Splunk can run the scheduled report if it misses the exact scheduled time.
- This is useful in environments with limited resources or heavy search loads. For example, if the report is scheduled to run at 6:00 AM but the system is busy, the schedule window allows Splunk to execute the report slightly later (e.g., within 5 or 10 minutes).



Step 12:

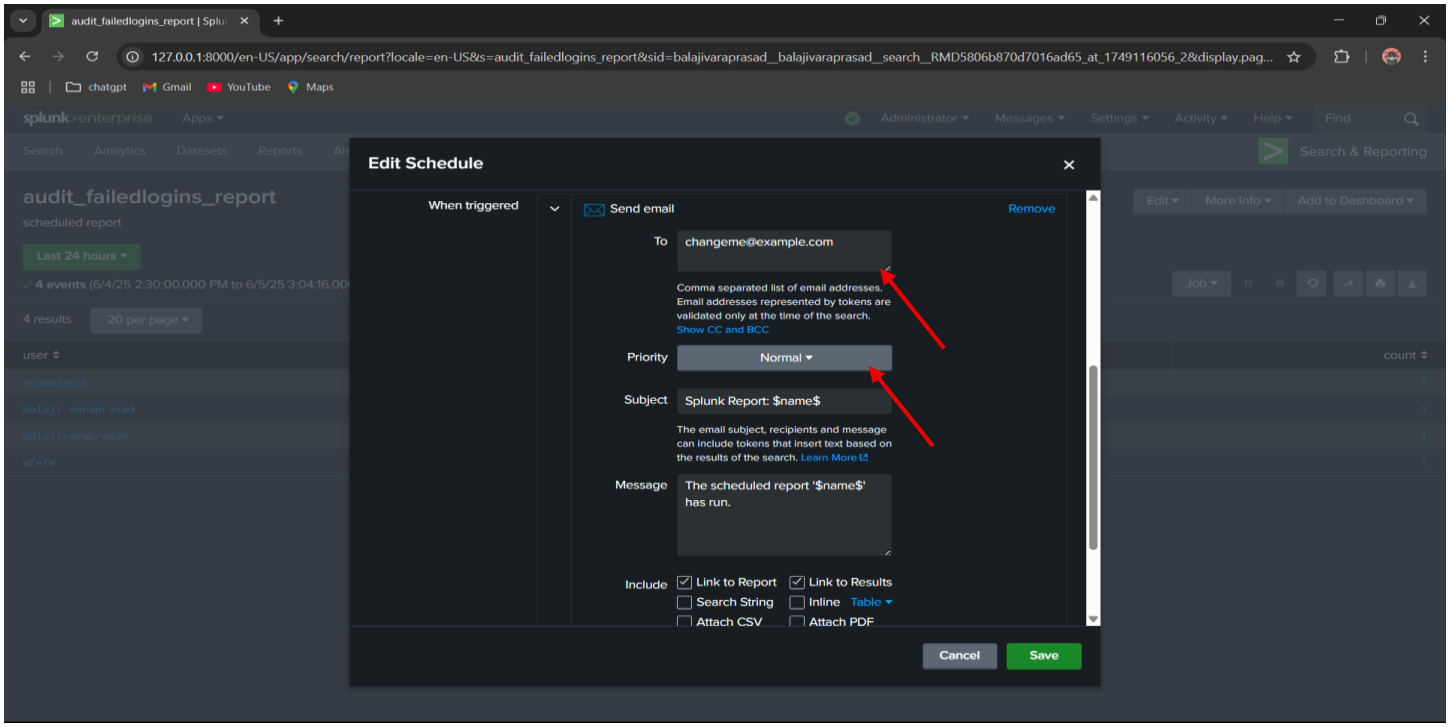
- In the Trigger Actions section of the scheduling window, click on “Add Actions” and select “Send Email” to configure email notifications for the report.
- This allows Splunk to automatically send the generated report to your specified email address upon each scheduled run.

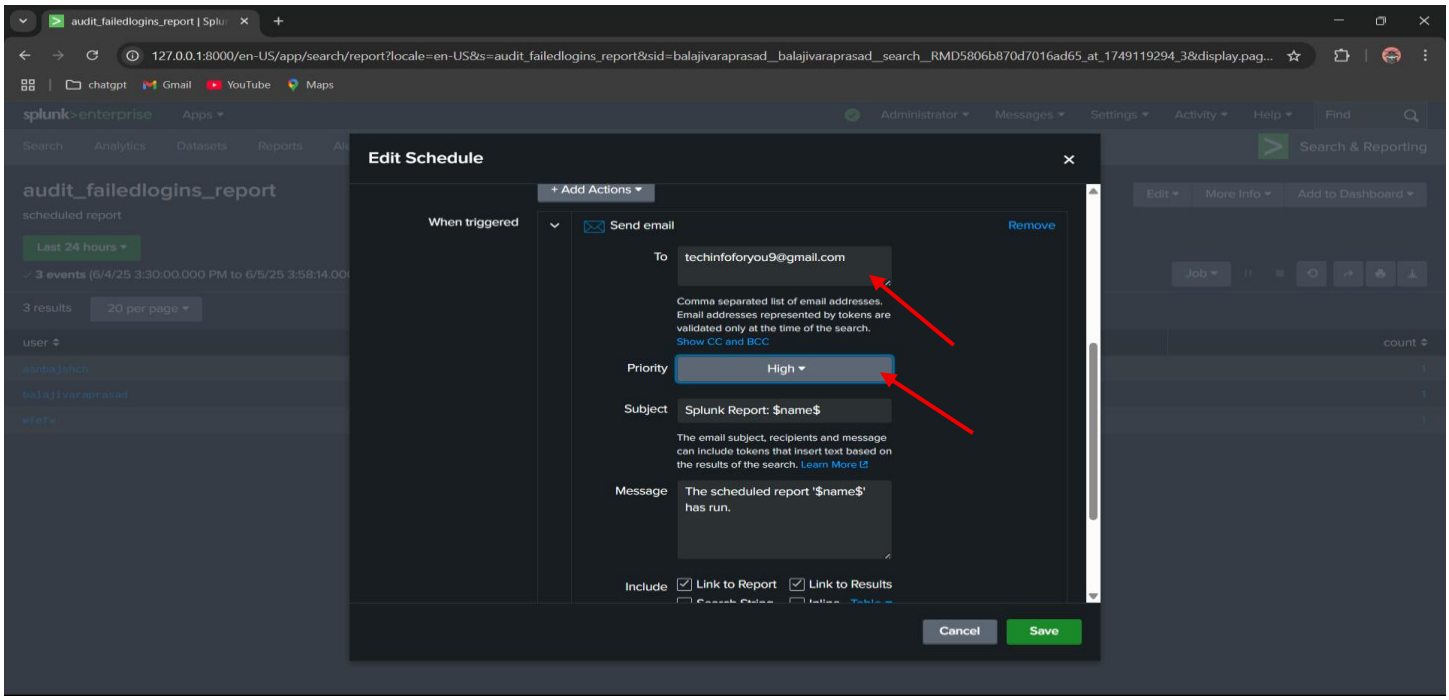




Step 13:

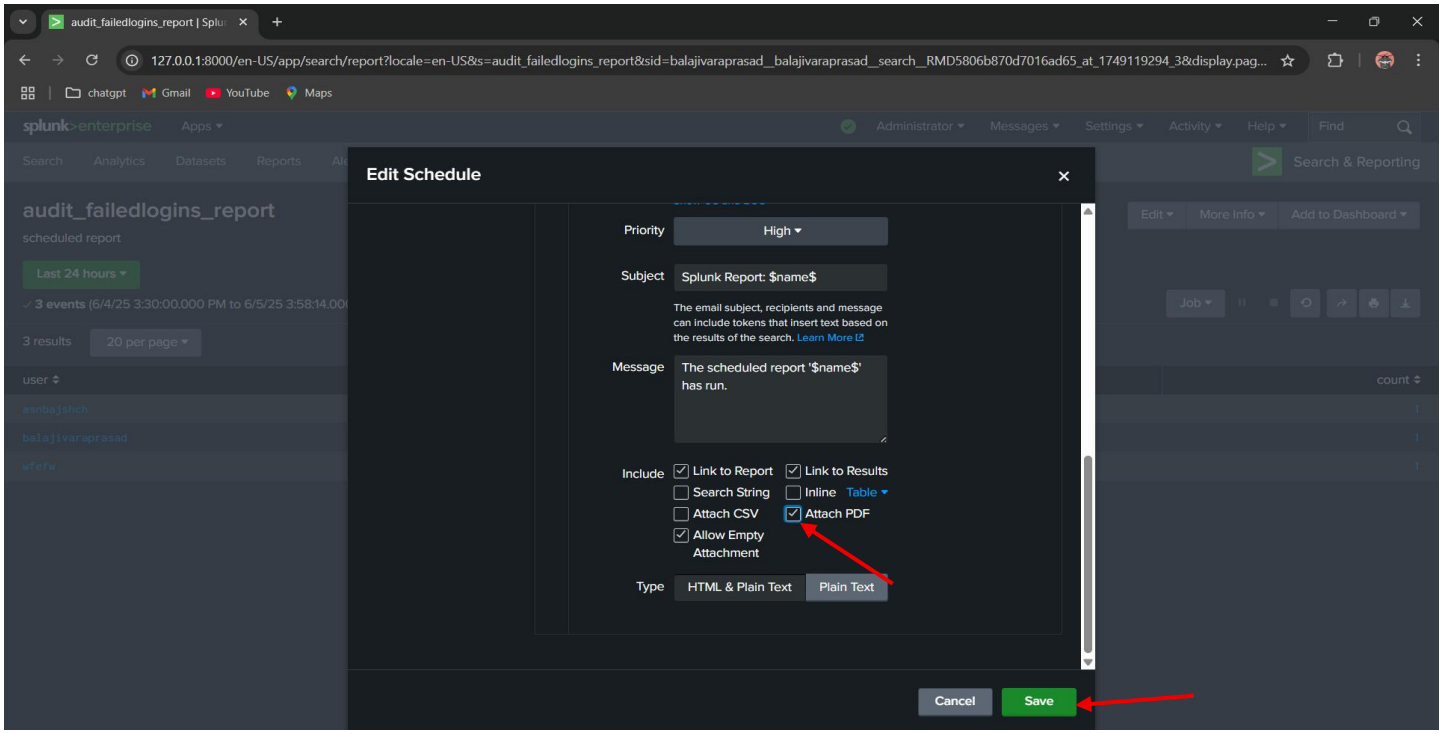
- After selecting the Send Email action, the next step is to configure the email settings. Begin by entering the recipient's email address in the "To" field, this is where the scheduled report will be delivered. You can add multiple addresses if needed, separated by commas.
- Finally, set the email priority to High to ensure the alert is treated with urgency by the recipient's mail system. These settings help ensure that critical security insights are promptly delivered and noticed.





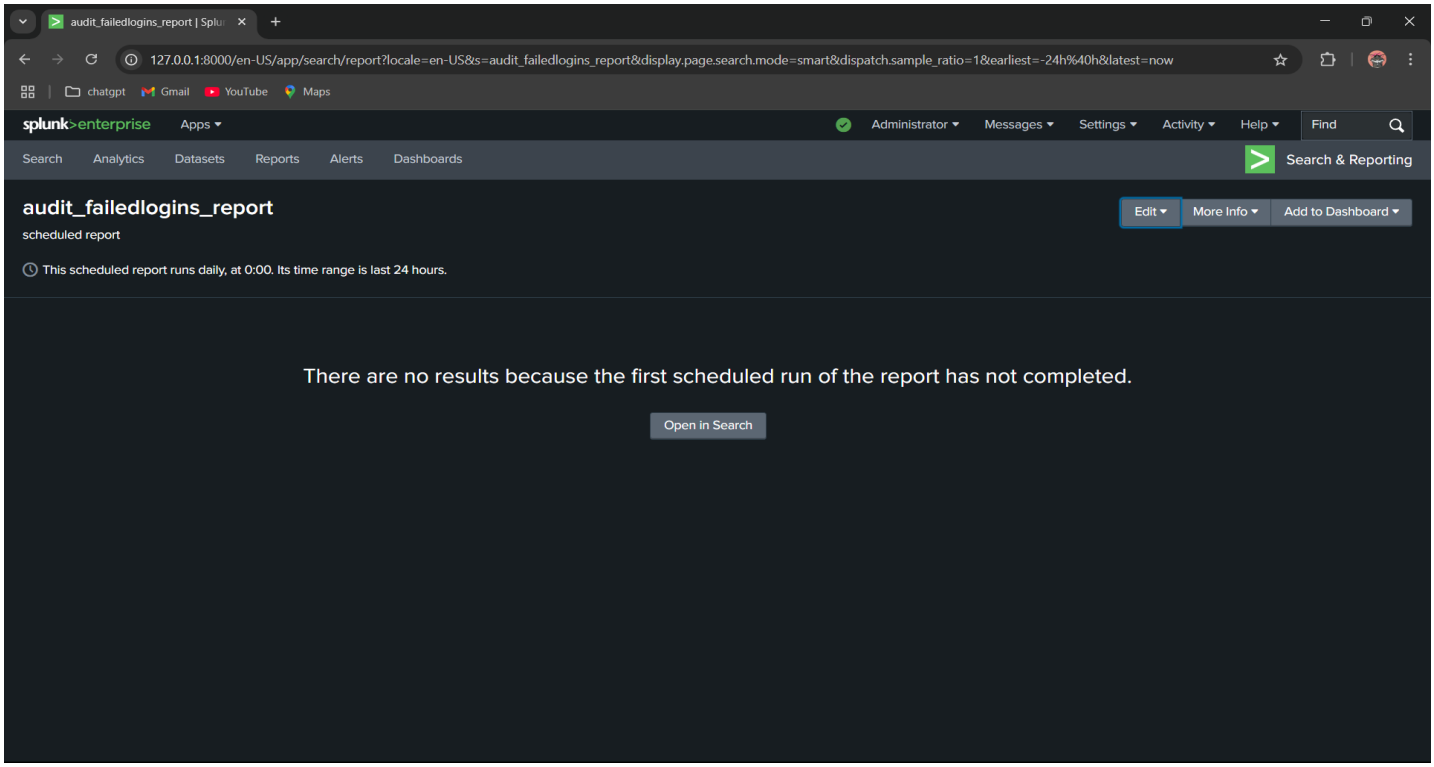
Step 14:

- In this final step, you can customize how the report will be shared via email.
- For this example, I chose to attach the report as a PDF, ensuring the recipient receives a clean, readable summary of failed login attempts.
- You may select whichever format suits your reporting needs. Once all the necessary options are selected, simply click the Save button to complete the configuration. This finalizes the scheduled report setup with automated email delivery.



Step 15:

- Your automated report with email delivery is now fully configured and ready to run as per the defined schedule.
- Below image shows, the scheduled report named audit_failedlogins_report has been successfully configured. It is set to run daily at 00:00 hours with a time range of the last 24 hours.
- Since the first scheduled run has not yet occurred, there are currently no results to display. This message confirms that the setup is complete and the system is waiting for the scheduled time to execute the report for the first time.
- Once the report runs at the scheduled time, Splunk will automatically generate the report and send it via email to the configured recipients.



- This confirms the report scheduling is active and that you'll start receiving automated reports from the next scheduled execution onward.

Alert Criteria and Configuration:

Alert Condition: Configure the alert to fire when failed logins exceed the threshold. For example, run the spl over the last 24h and set "Trigger alert when number of results > 3." This ensures that any surge in failures (beyond 3) raises

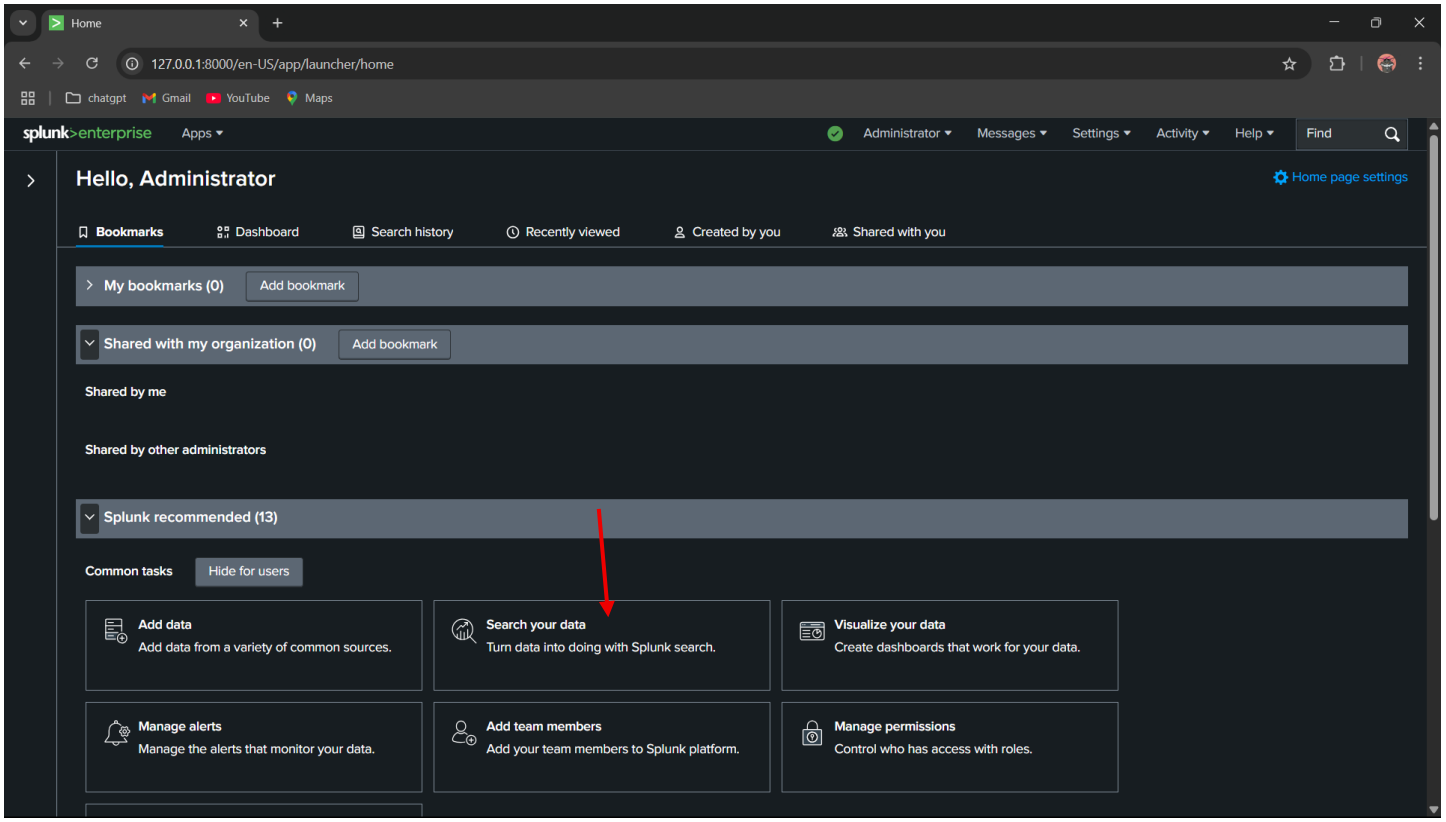
an alarm. Splunk will run the saved search on schedule and count the results, if the count is above 3, the alert triggers.

Example spl query:

index=_audit action="login attempt" info="failed" | stats count as failed_attempts by user, _time | where failed_attempts > 3

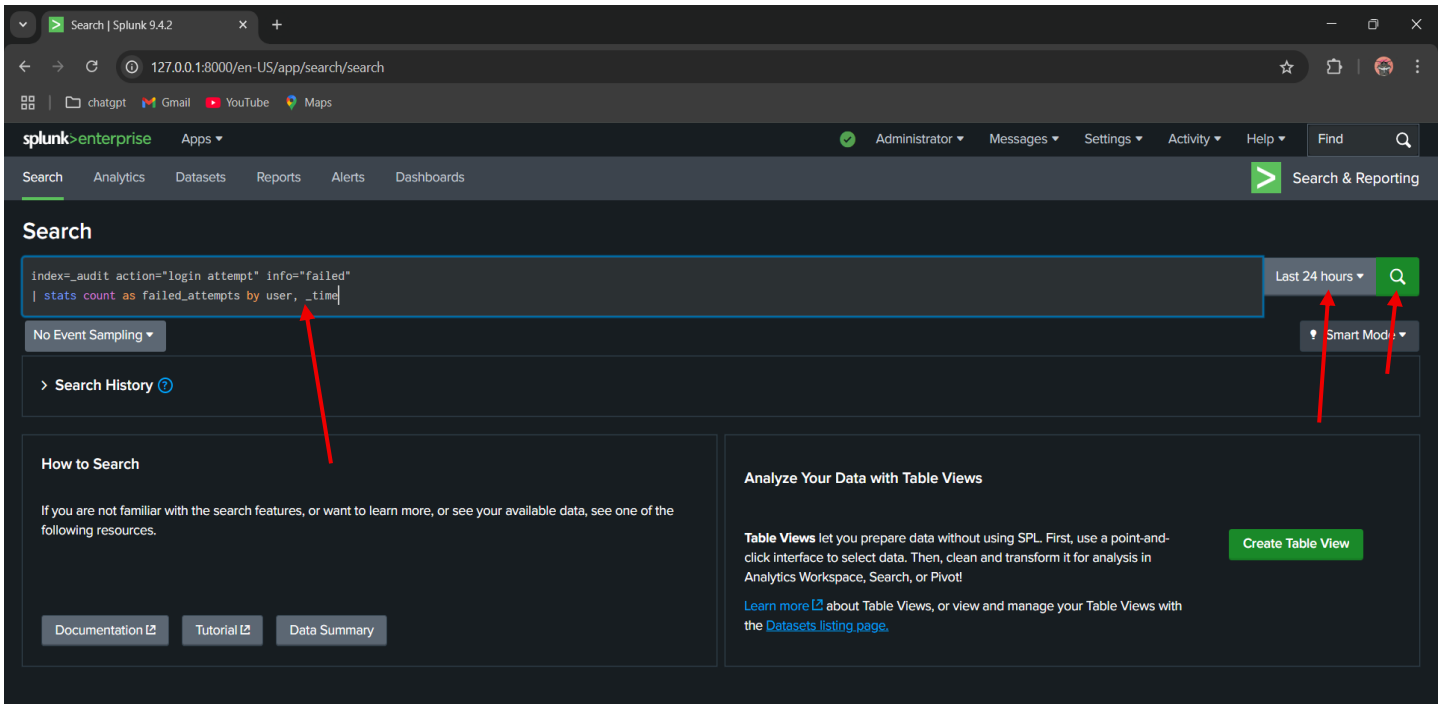
Step 1:

- Start by logging in to your Splunk instance using your credentials. Once logged in, navigate to the Search & Reporting app from the homepage.
- This is where you will run your search query, configure trigger conditions, and begin creating the alert based on specific criteria, in this case, failed login attempts.



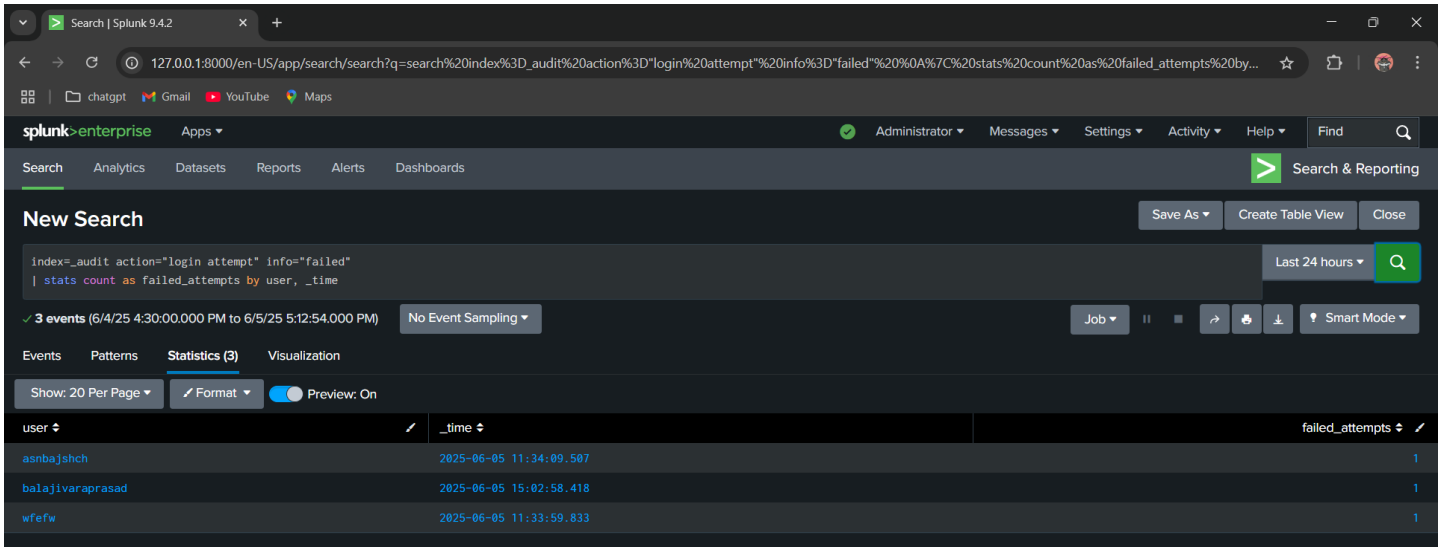
Step 2:

- In the Search & Reporting app, enter the following SPL (Search Processing Language) query to extract failed login attempts from the internal audit index:
- index=_audit action="login attempt" info="failed" | stats count as failed_attempts by user, _time
- This query filters the audit logs to display only failed login attempts. It then uses the stats command to count the number of failed login attempts for each user over time.
- Set time filter to 24hrs and click on search.
- This is the same query used earlier for generating the scheduled report. Reusing it ensures consistency in data analysis and alerting.



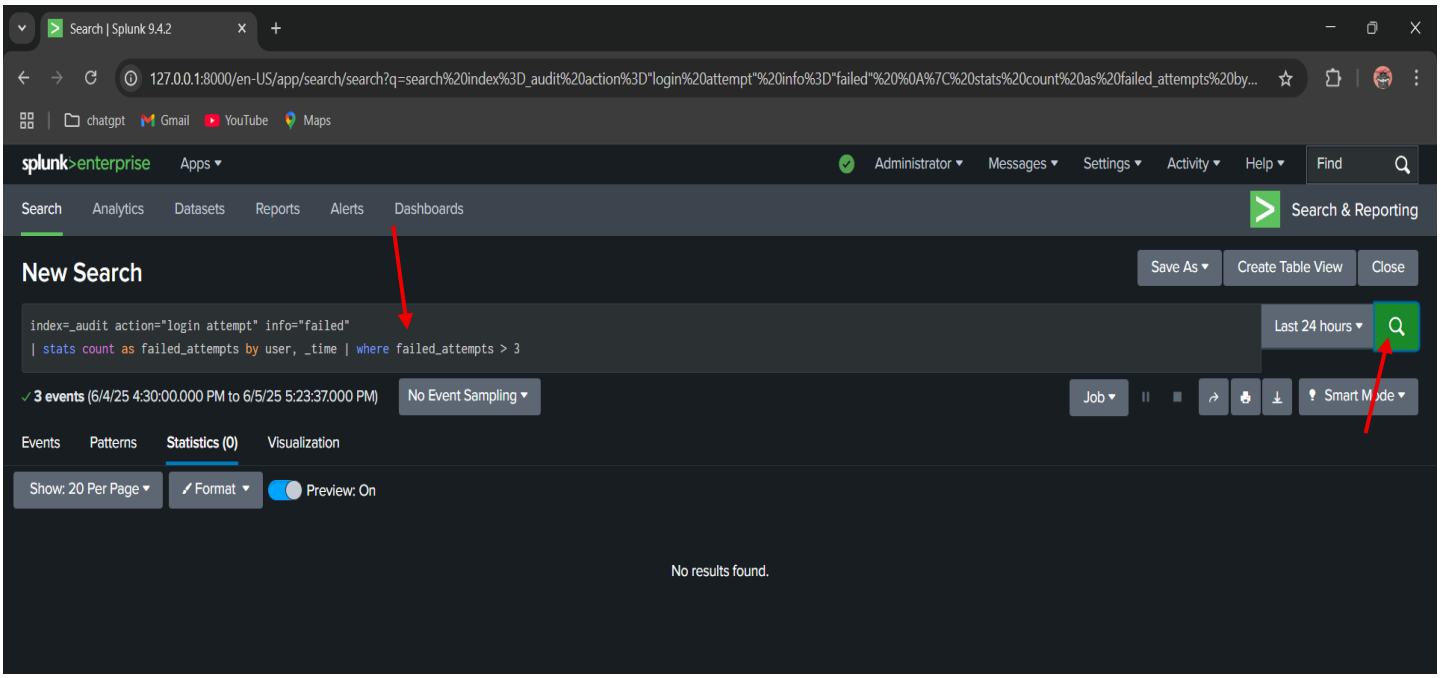
Step 3:

- The "Statistics" tab shows a table with three columns: user, _time, and failed_attempts. Each row represents a failed login event by a specific user along with the timestamp and the count of failed attempts.
- In this case, three users (asnbajshch, balajivaraPrasad, and wfefw) each have one failed login attempt within the last 24 hours.



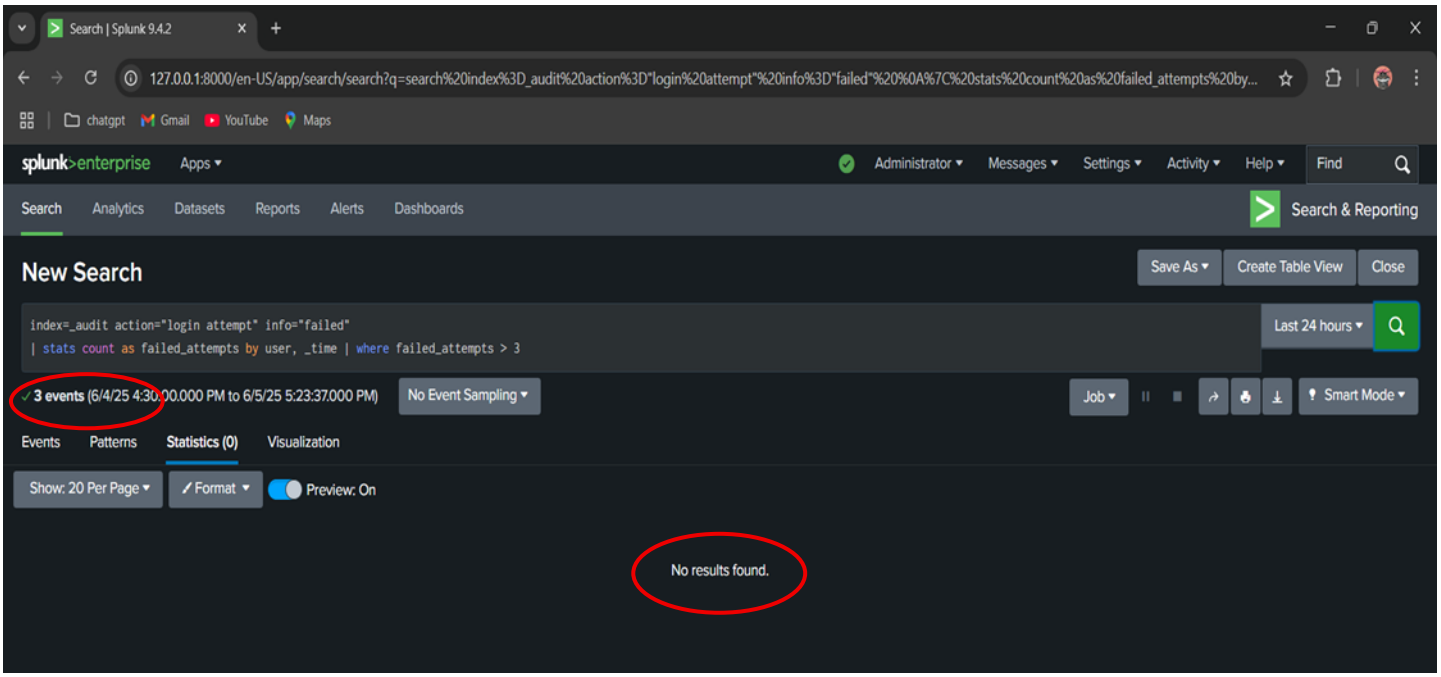
Step 4:

- To configure an alert, we need a condition that defines when the alert should be triggered. In this example, we'll use a threshold value of 3 failed login attempts. This means the alert will only trigger if the number of failed login attempts by any user exceeds 3 within the specified time range.
- Update your previous SPL query by appending a where clause as shown:
- index=_audit action="login attempt" info="failed" | stats count as failed_attempts by user, _time | where failed_attempts > 3
- This refined query filters out all results where failed login attempts are 3 or fewer and retains only the events that exceed the threshold, which is essential for setting up a meaningful and actionable alert.



Step 5:

- Even though there are 3 events displayed initially, we see no results after applying the threshold filter. This is because none of the users in the dataset have more than 3 failed login attempts, which is the threshold we defined in the where clause.
- Since our query now filters results to only include users with more than 3 failed login attempts, and none meet this condition, the final output is empty. This confirms that the alert logic is working as intended, it will only trigger when a genuine threshold breach occurs.

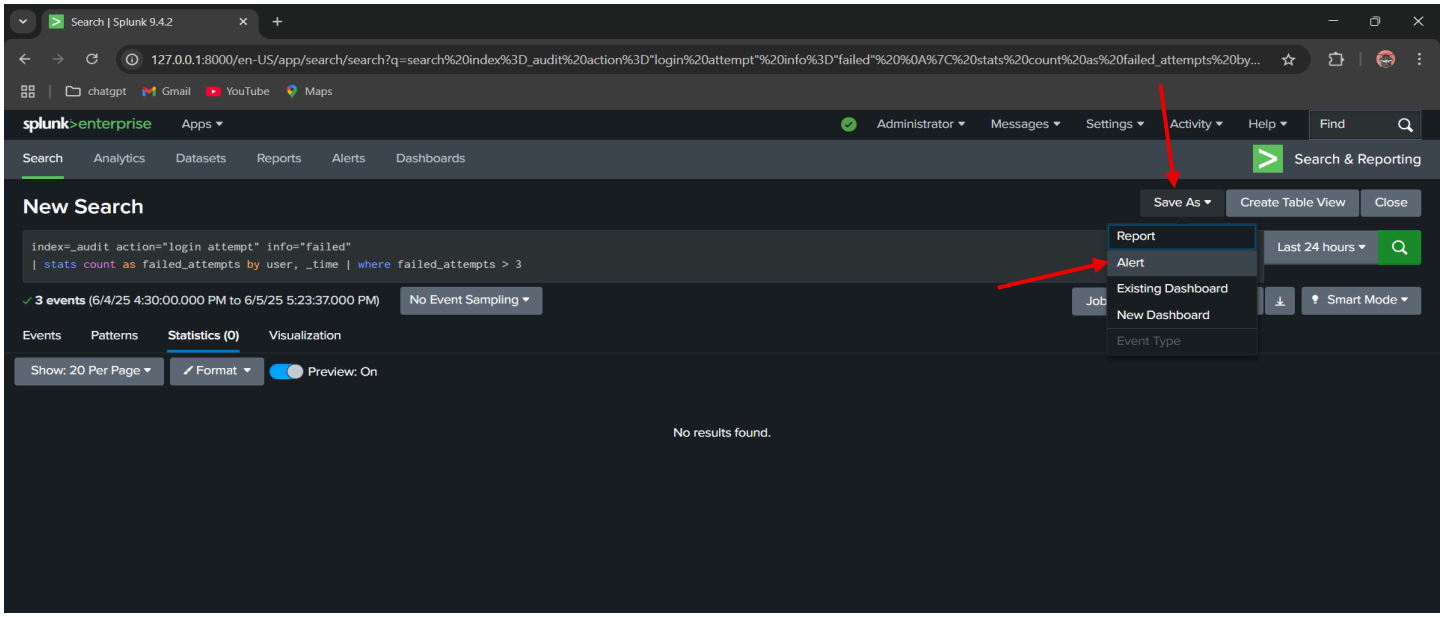


Step 6:

Now that we've applied the threshold condition to detect users with more than 3 failed login attempts, the next step is to configure this search as an alert.

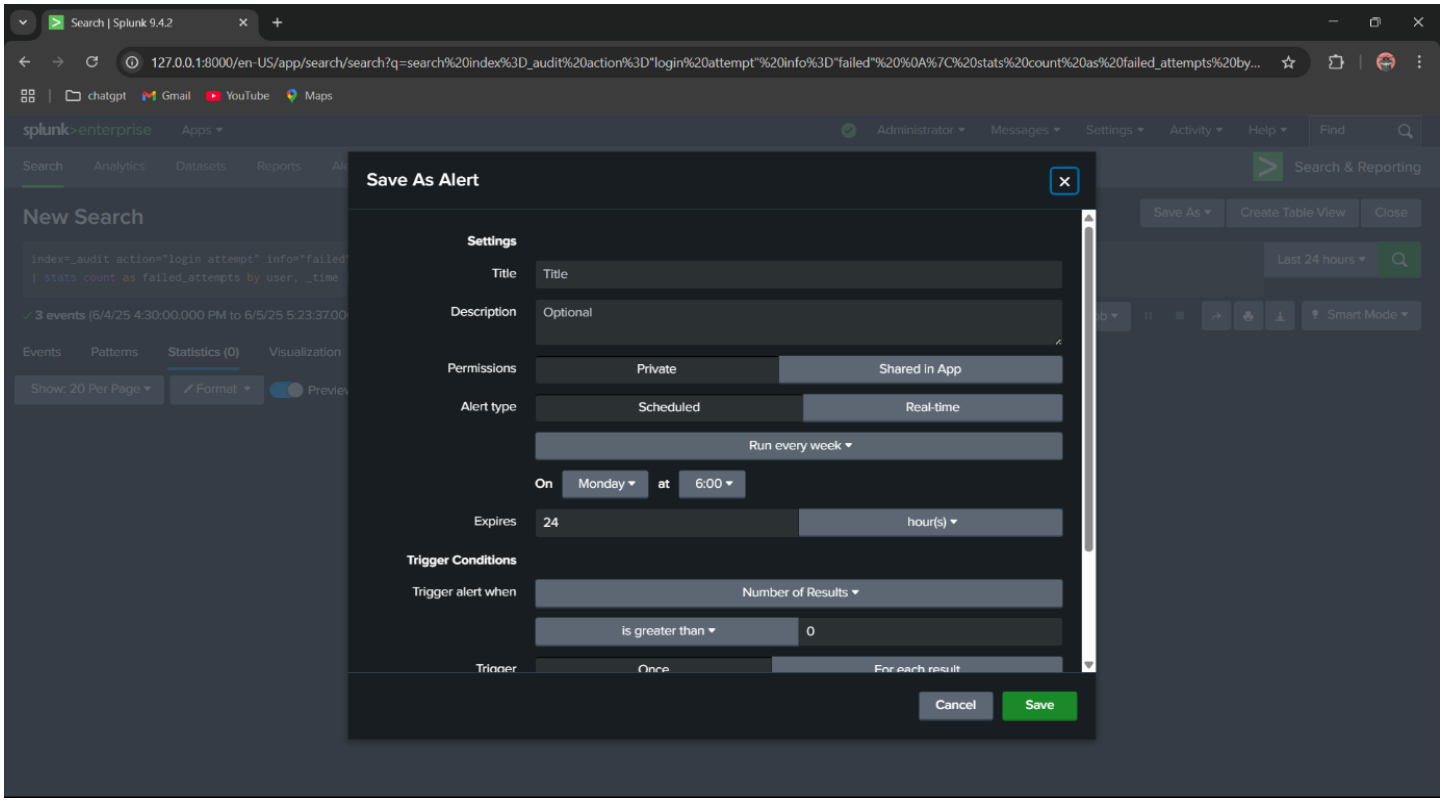
Step 7:

- To do this, click on the "Save As" button at the top right of the search window and select "Alert" from the dropdown.
- This allows us to save the search query and define the conditions under which an alert should be triggered, in this case, whenever failed login attempts exceed our set threshold.



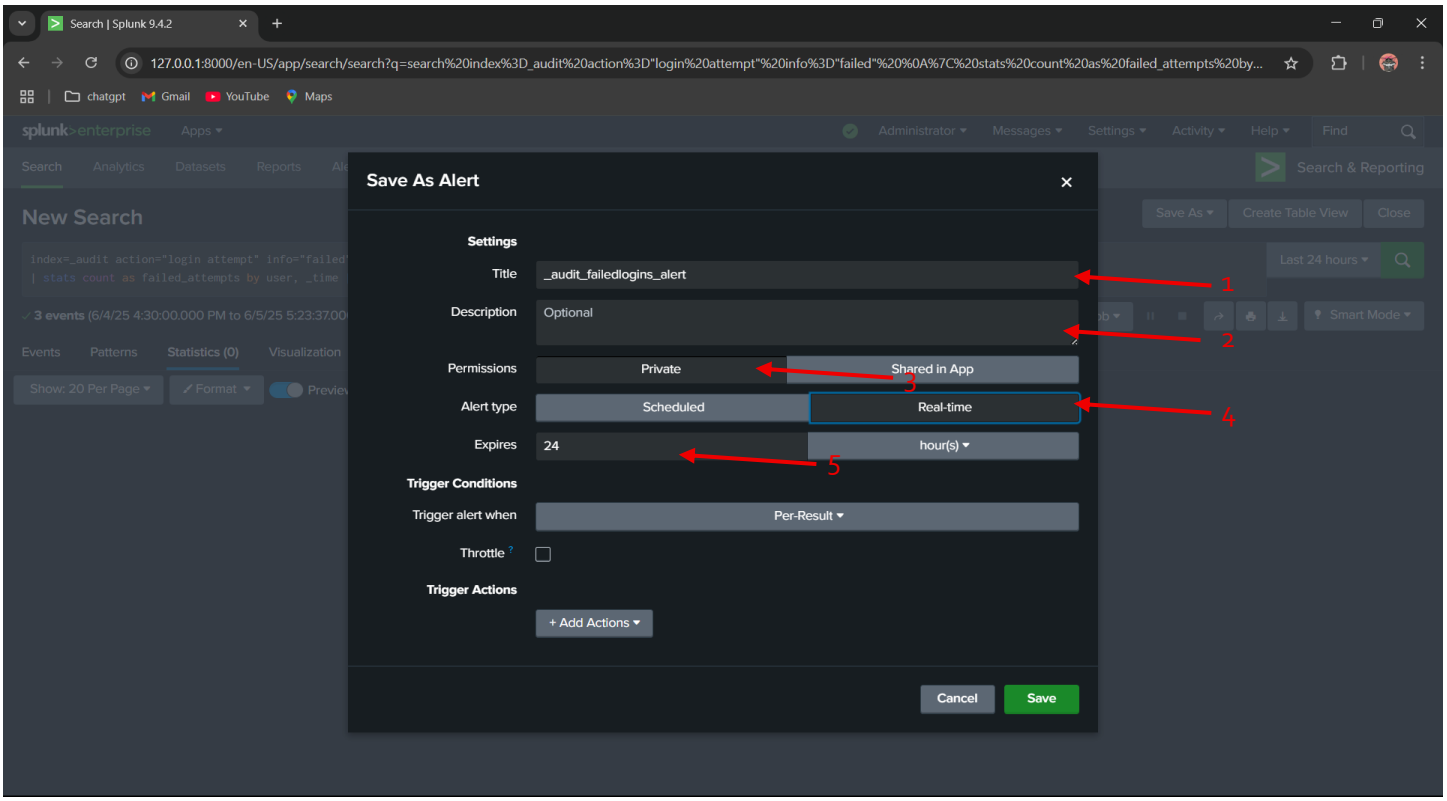
Step 8:

After clicking on “Save As → Alert,” a configuration dialog box appears where you need to provide the necessary details to set up the alert.



Step 9:

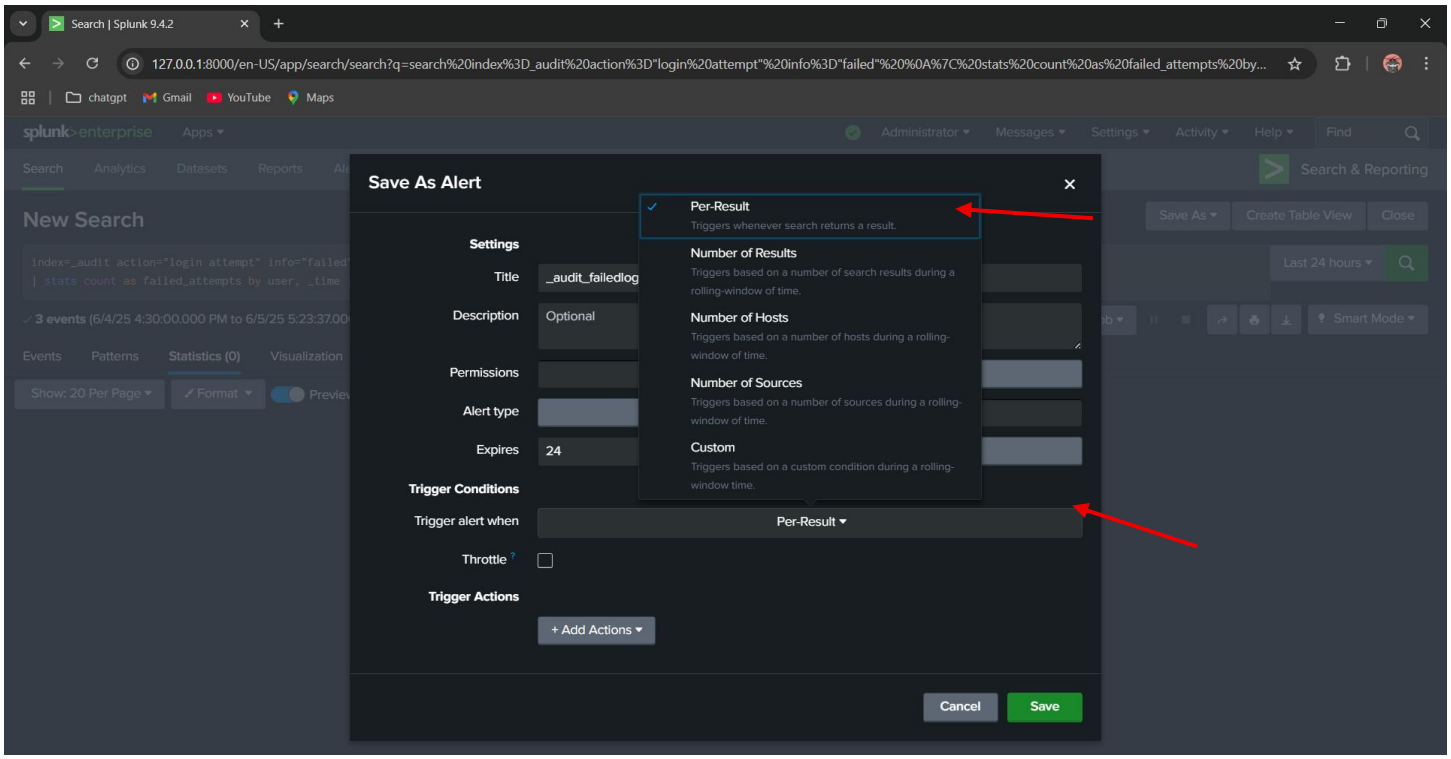
1. **Title**: Enter a meaningful name for the alert. Here, I used _audit_failedlogin_alert.
2. **Description:** This field is optional. You can provide a brief explanation of the alert if needed.
3. **Permissions:** Leave it as Private to restrict visibility to only you.
4. **Alert Type:** Select Real-time to enable continuous monitoring and trigger alerts instantly when conditions are met.
5. **Expiration Time:** You can adjust this based on how long the alert should stay active. In this case, it is left at the default 24 hours, but you can set it to specific days or hours if required**.**

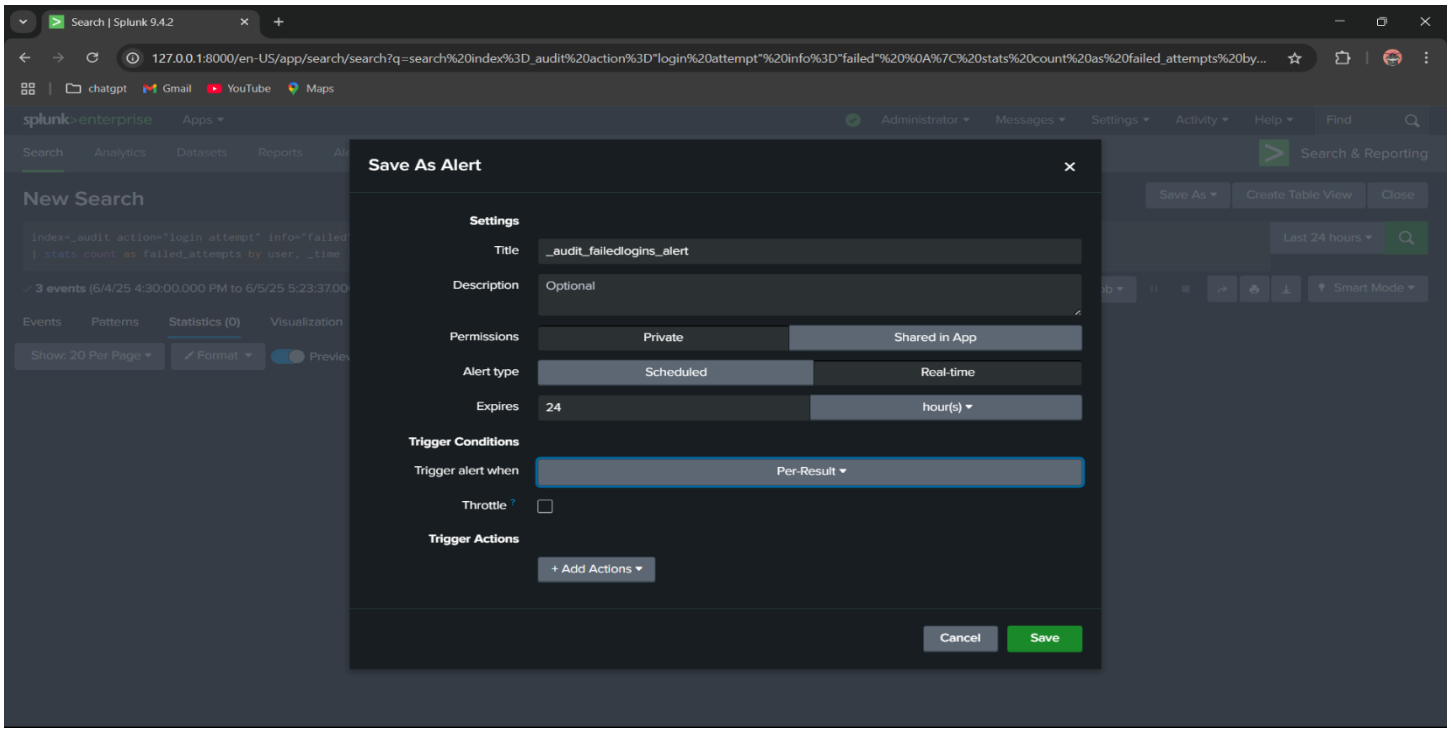


Step 10:

- In the "Trigger alert when" section, choose Per Result as the trigger condition.
- This means the alert will be triggered each time a single search result meets the defined condition, in this case, when a user has more than 3 failed login

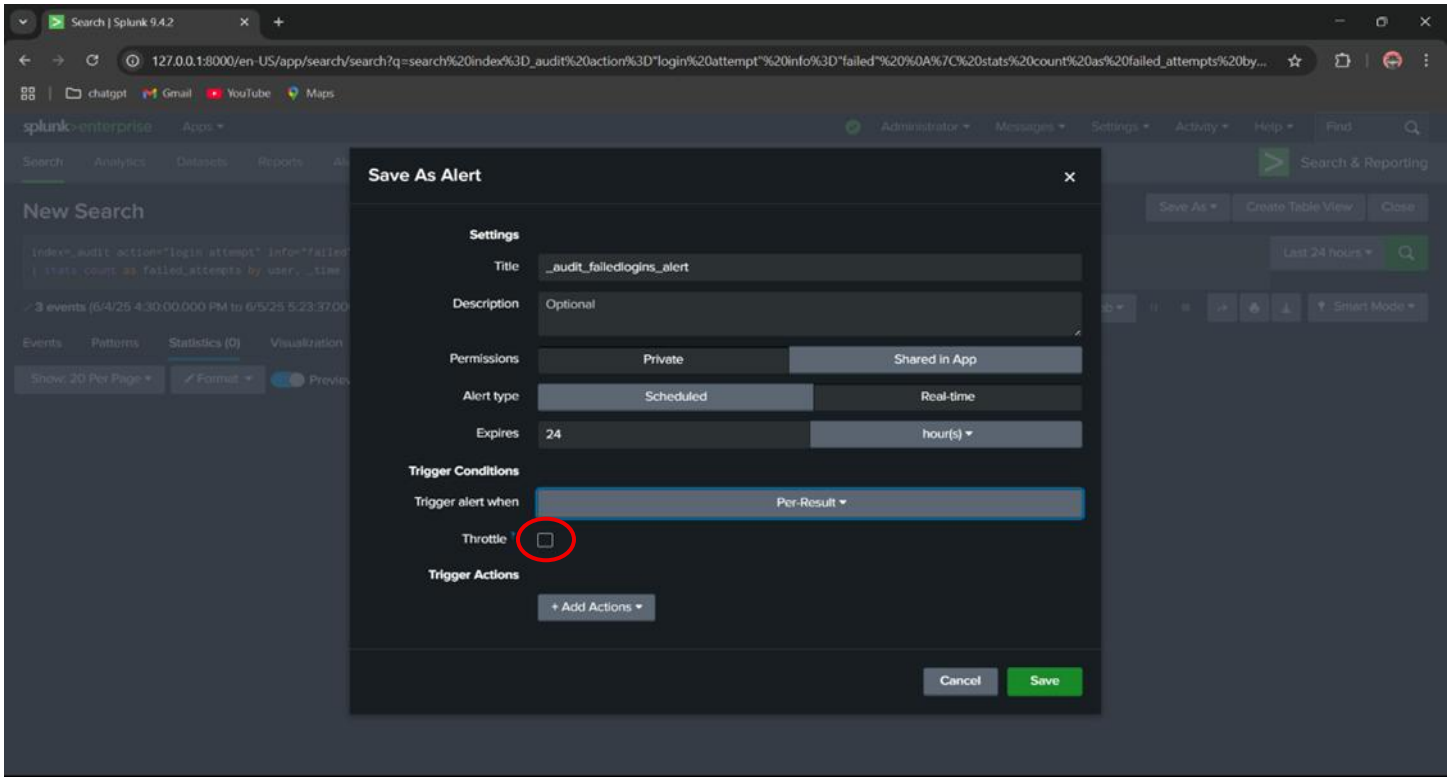
attempts. This setting is ideal for real-time monitoring scenarios where you want to be notified every time a matching event occurs.





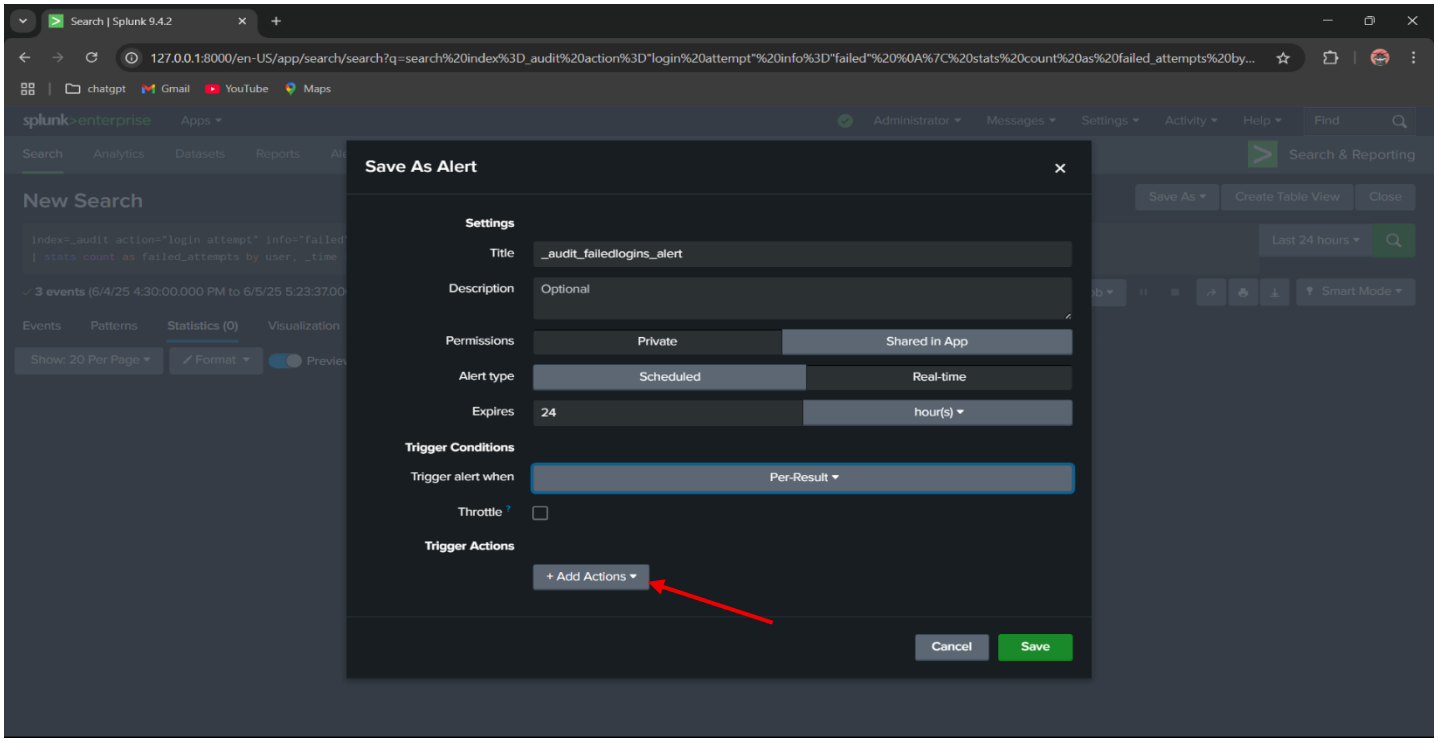
Step 11:

- In this step, I did not enable the throttle option. However, if you prefer, you can configure it based on your alerting needs.
- Throttle is used to prevent duplicate or repeated alerts from being triggered within a short time frame for the same condition. When enabled, it suppresses additional alerts for a specific time period (e.g., 1 hour) even if the condition is still being met.
- This is especially useful to avoid alert fatigue when the same issue continuously triggers the alert. For example, if a user repeatedly fails to log in, throttling ensures you receive just one alert instead of dozens in quick succession.



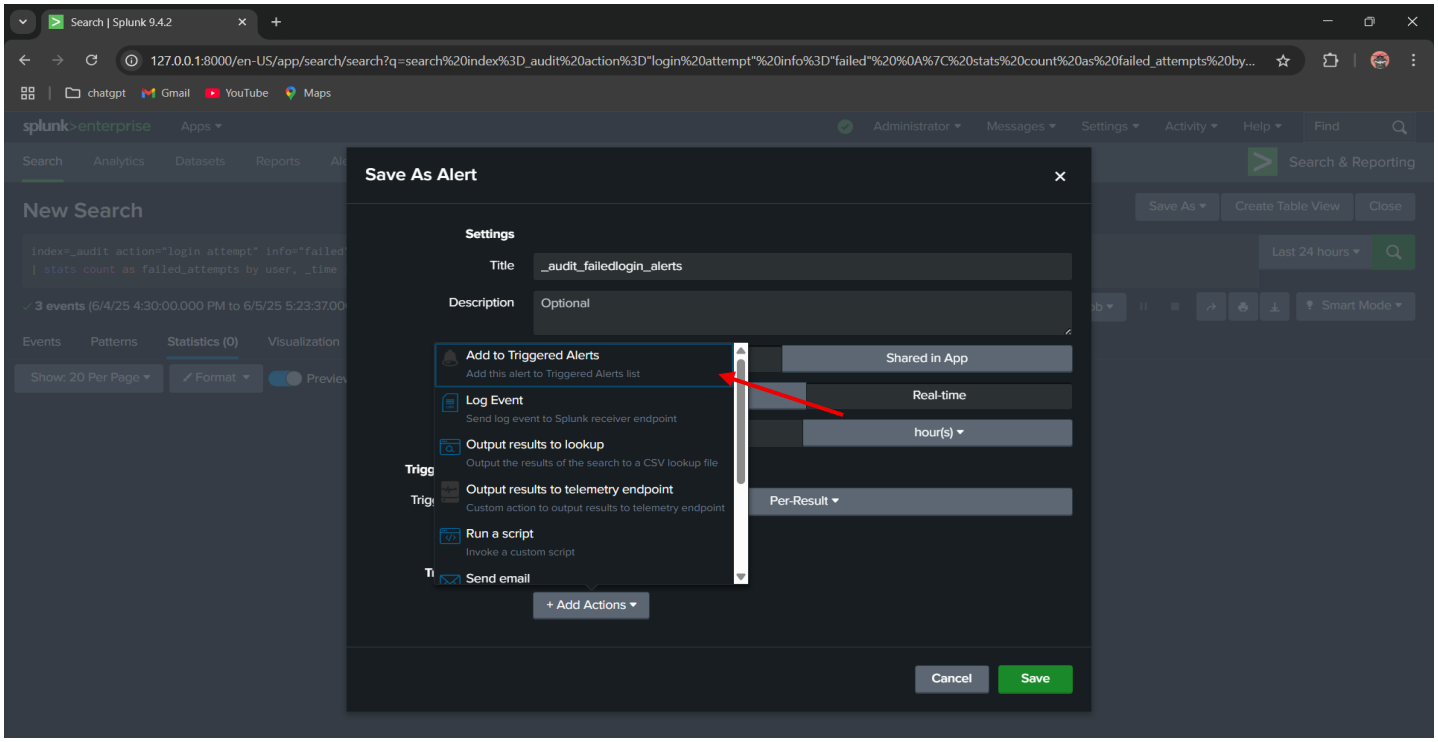
Step 12:

Now it's time to set how the alert should notify you when triggered. Under the Trigger Actions section, click on the "Add Actions" button. A dropdown menu will appear with several available options.



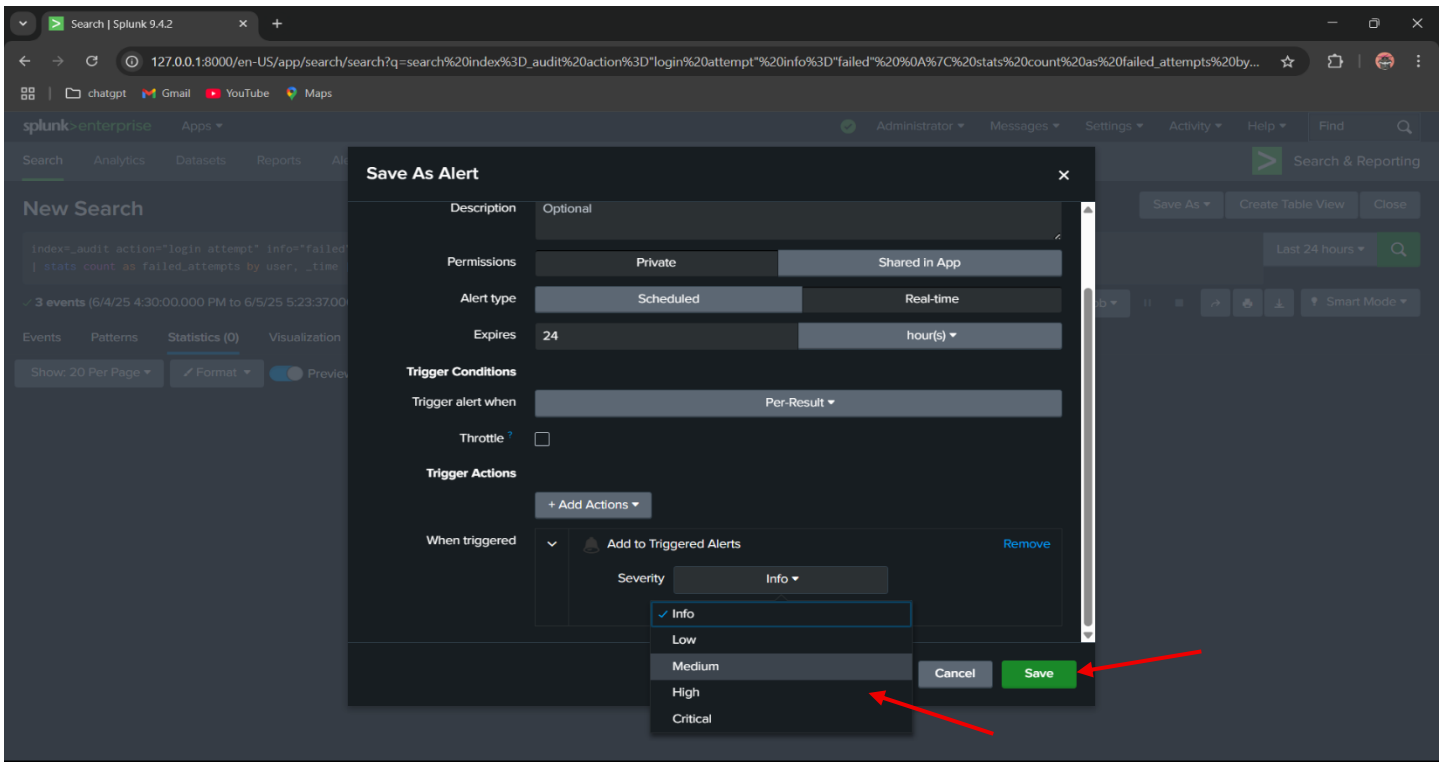
Step 13:

Now simply click on "Add to Trigger Alerts". This confirms and attaches the action to your alert configuration. Once added, the action becomes active and will execute whenever the alert condition is met.



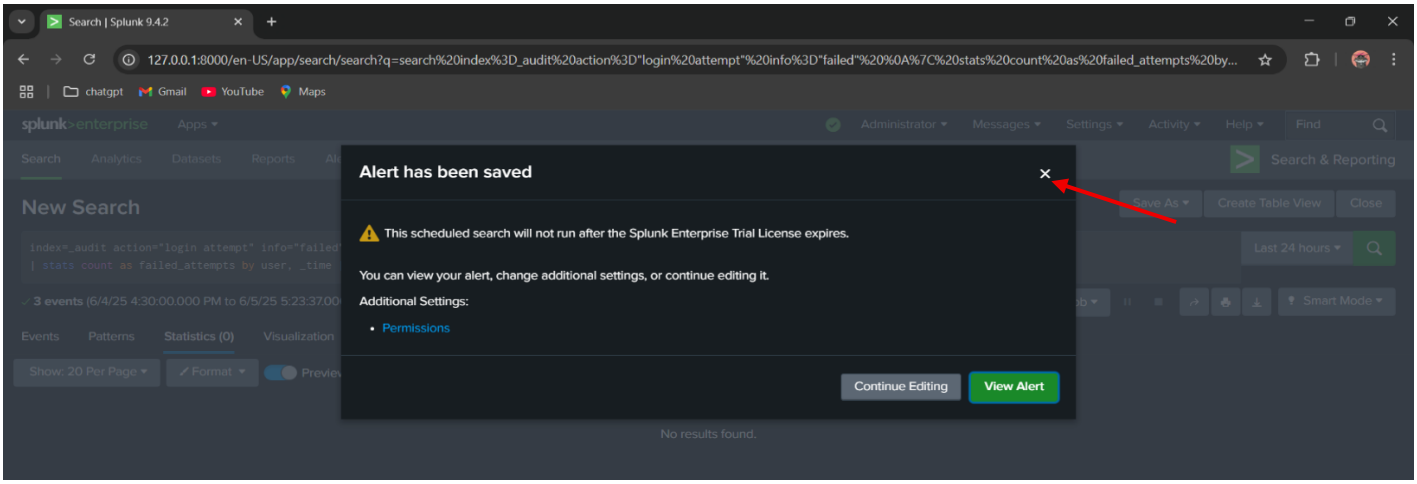
Step 14:

After clicking on Add to configure alert triggers, set the severity level according to your preference. In this case, I selected High. Finally, click Save to apply the settings.



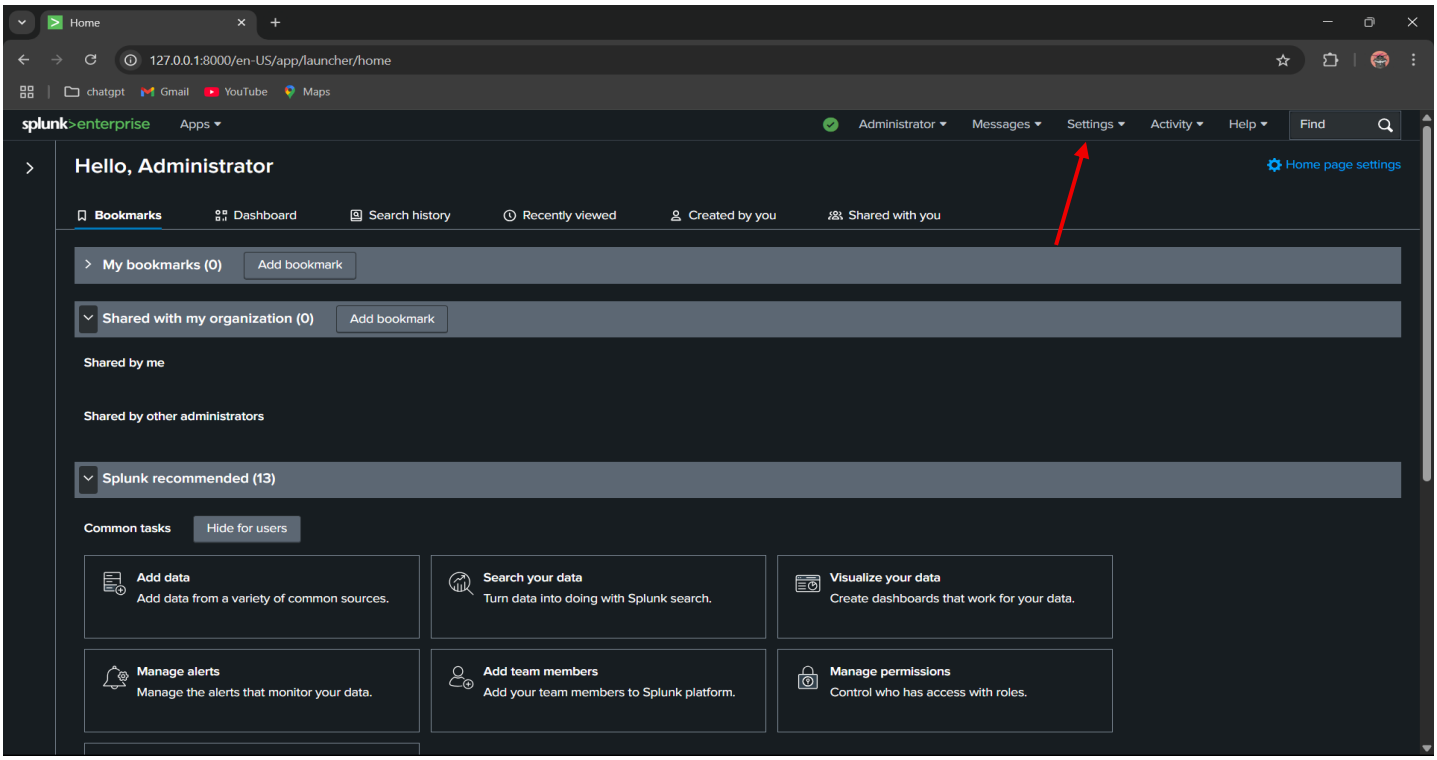
Step 15:

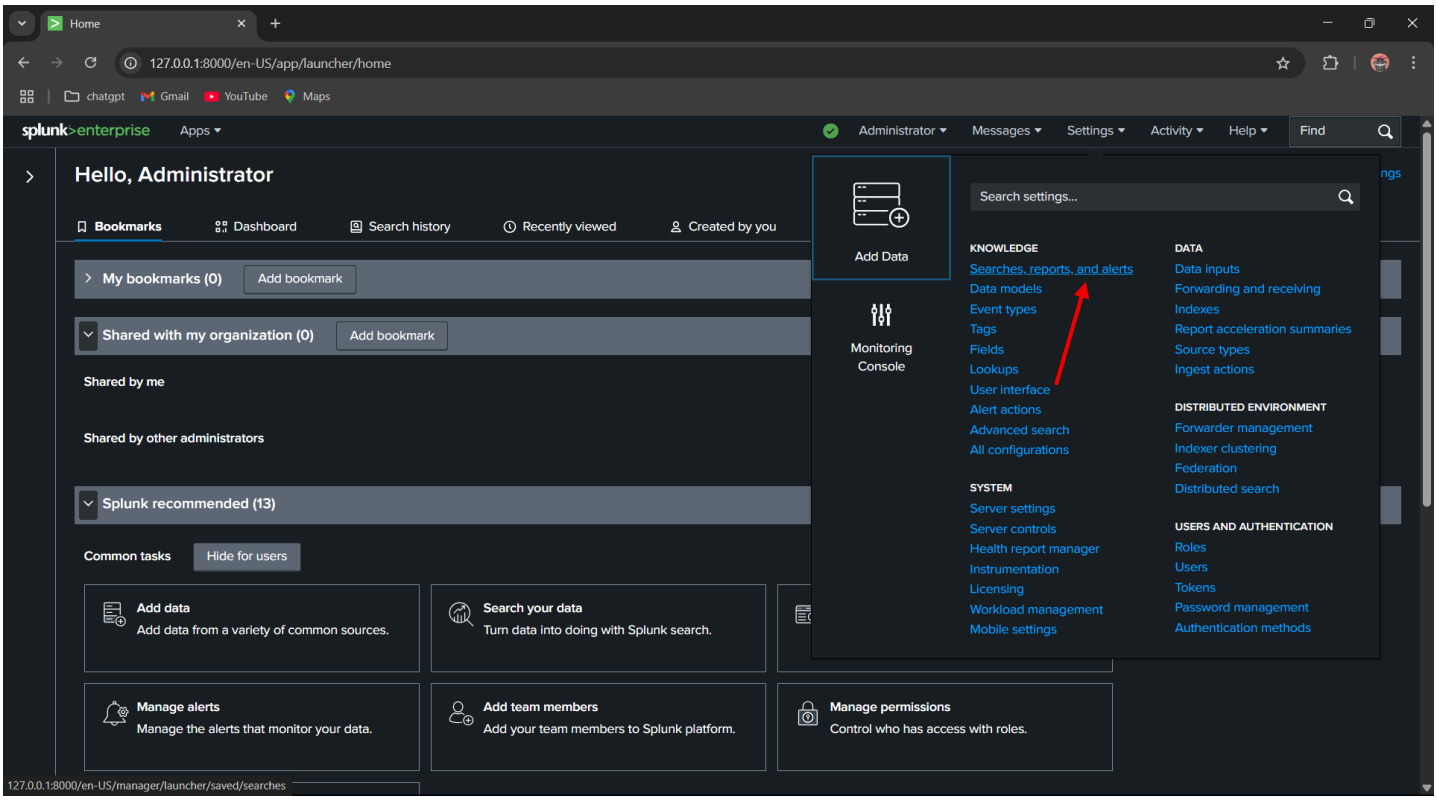
After saving, a pop-up window will appear confirming that the alert has been successfully created. You can either close this window and access the alert later from the dashboard, or click on View in the pop-up to open the alert immediately**.**



Step 16:

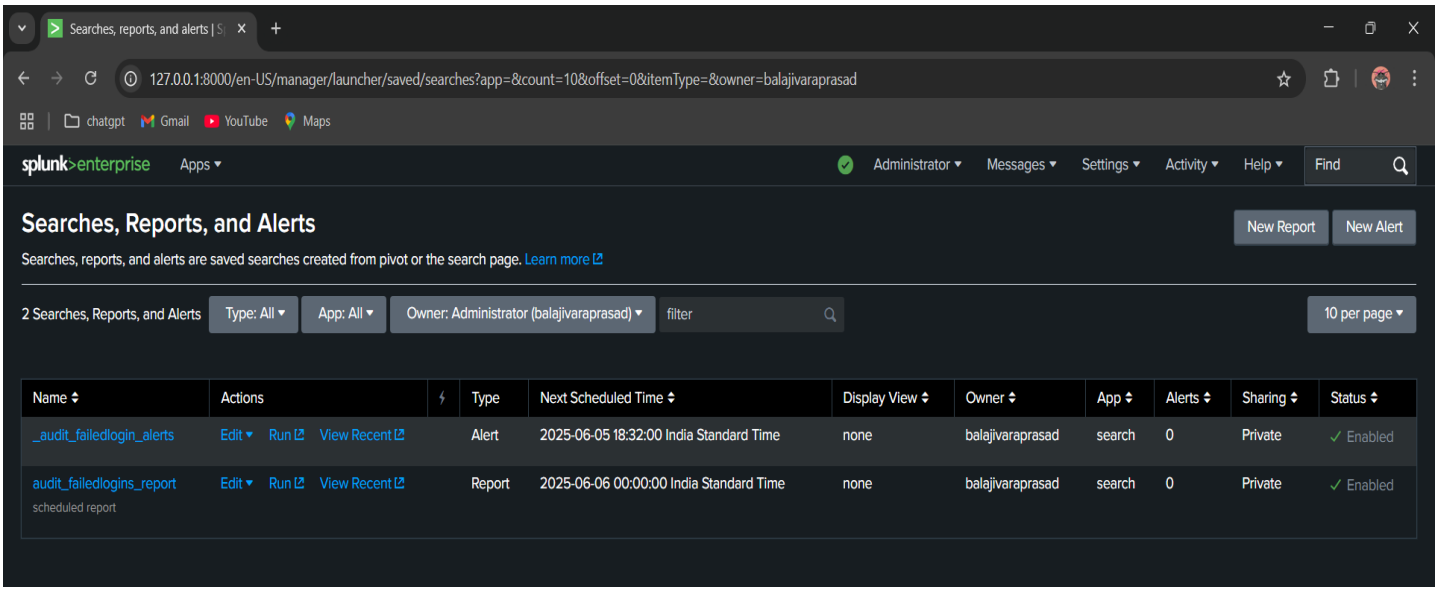
Navigate to the Splunk homepage by clicking on the Splunk icon. Then, go to Settings and under the Knowledge section, click on Searches, Reports, and Alerts.





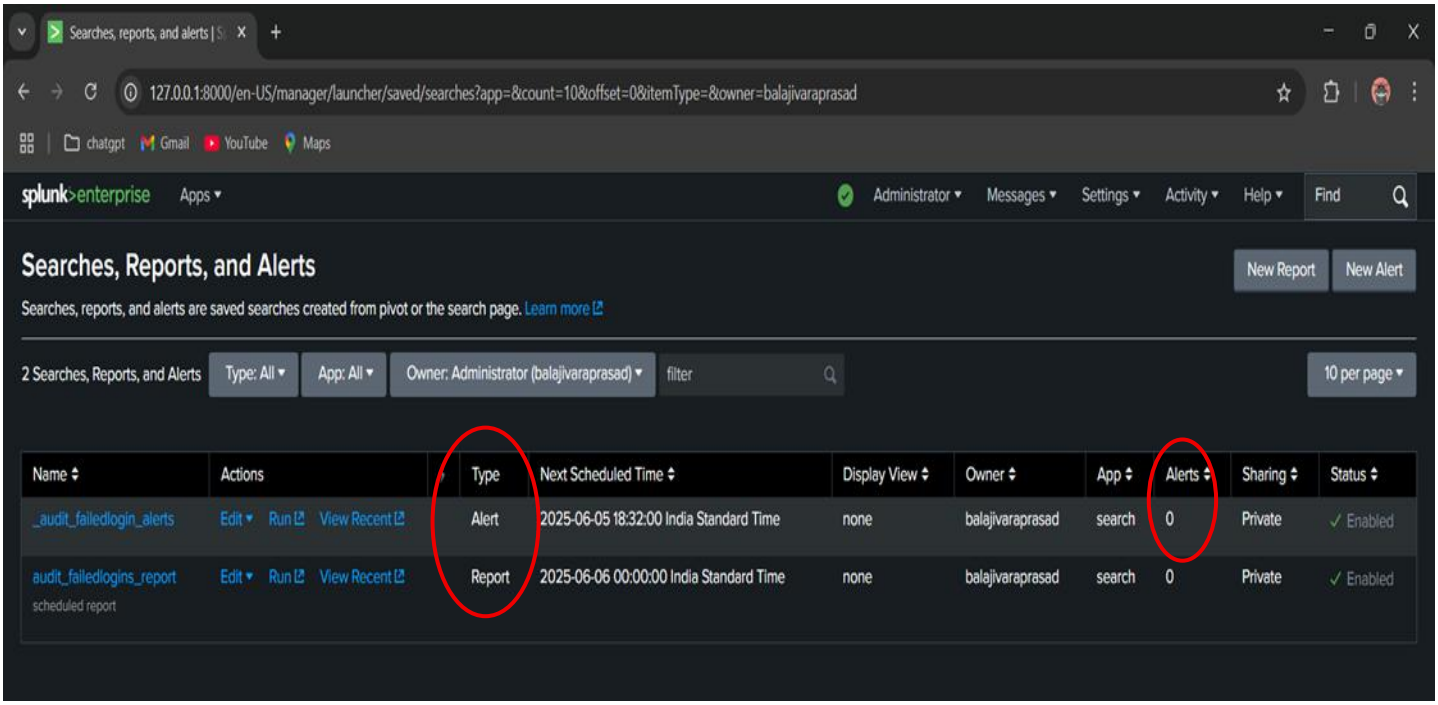
Step 17:

You can now view the alert that was just created. Additionally, the scheduled report created earlier will also be visible in the list.



Step 18:

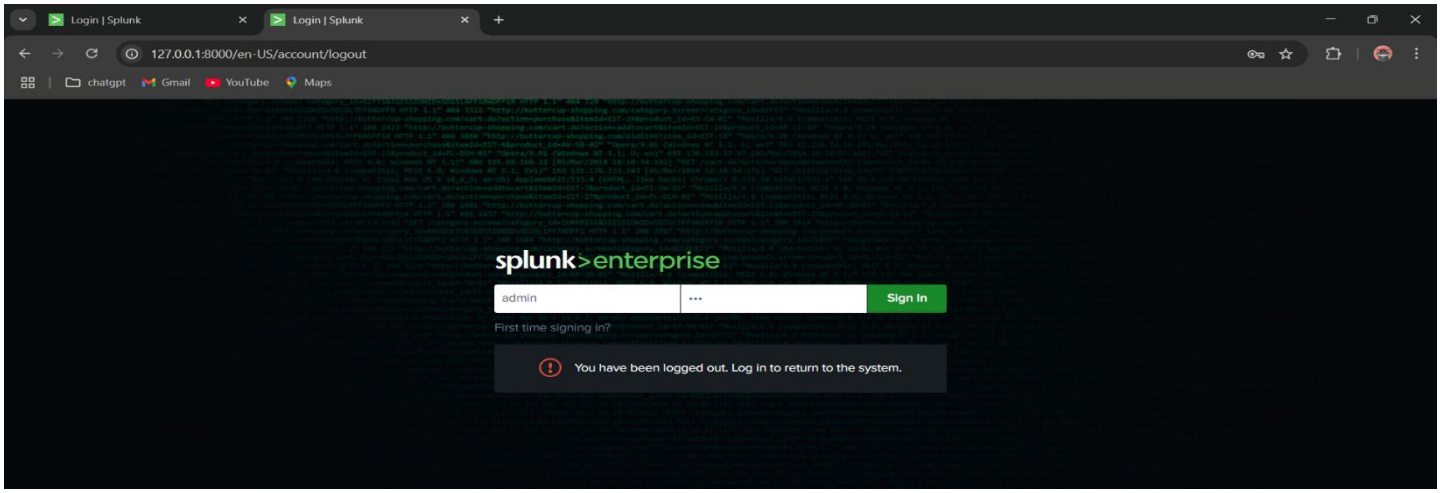
You can view the type of each data and the number of times the alert has been triggered. Currently, the alert count is zero because no failed login attempts have occurred since the alert was created.

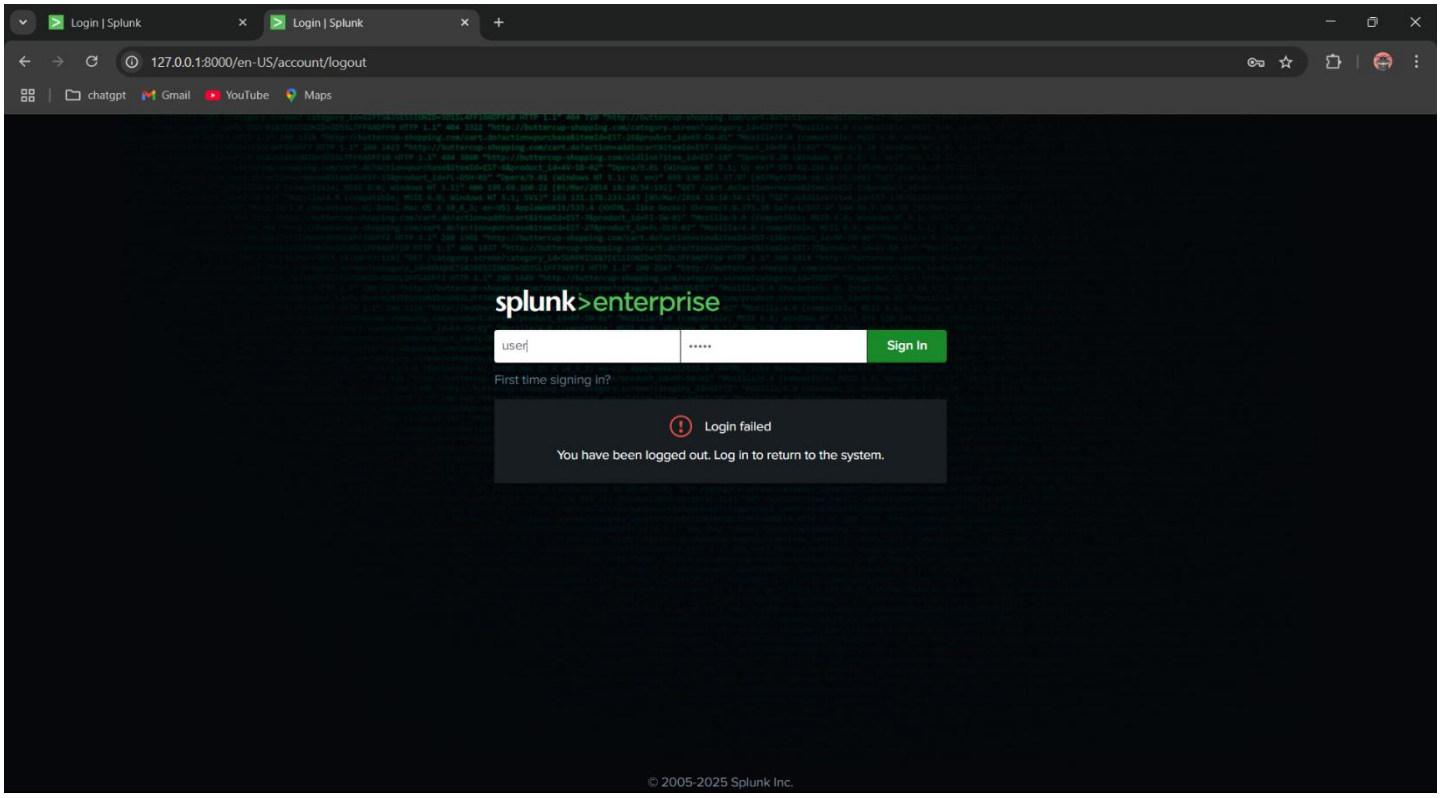


Let’s simulate failed login attempts in Splunk to trigger the alert and verify that it functions as expected**.**

Step 19:

Open Splunk in a new browser tab and navigate to the login page. Enter a username of your choice and intentionally provide an incorrect password. Click the login button repeatedly, around 3 to 5 times or more, to simulate failed login attempts. This action generates failed login events that exceed the threshold value specified in your alert query.





Step 20:

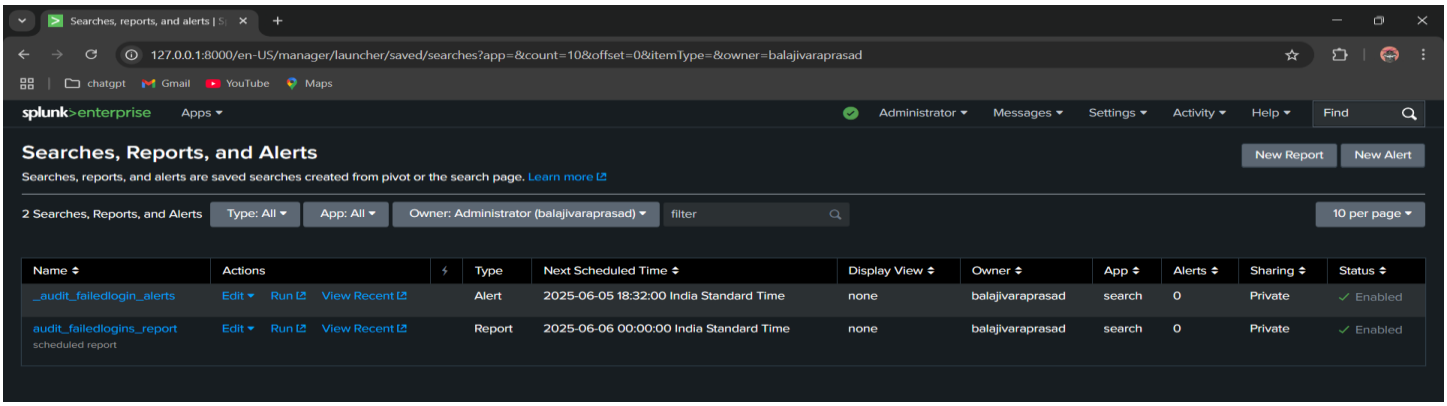
After simulating the failed login attempts, log in to Splunk using your original credentials to access the platform again.

Step 21:

Now, repeat the steps from Step 16 to Step 18 to navigate back to the Searches, Reports, and Alerts section. There, you can verify that the alert has been triggered as a result of the simulated failed login attempts.

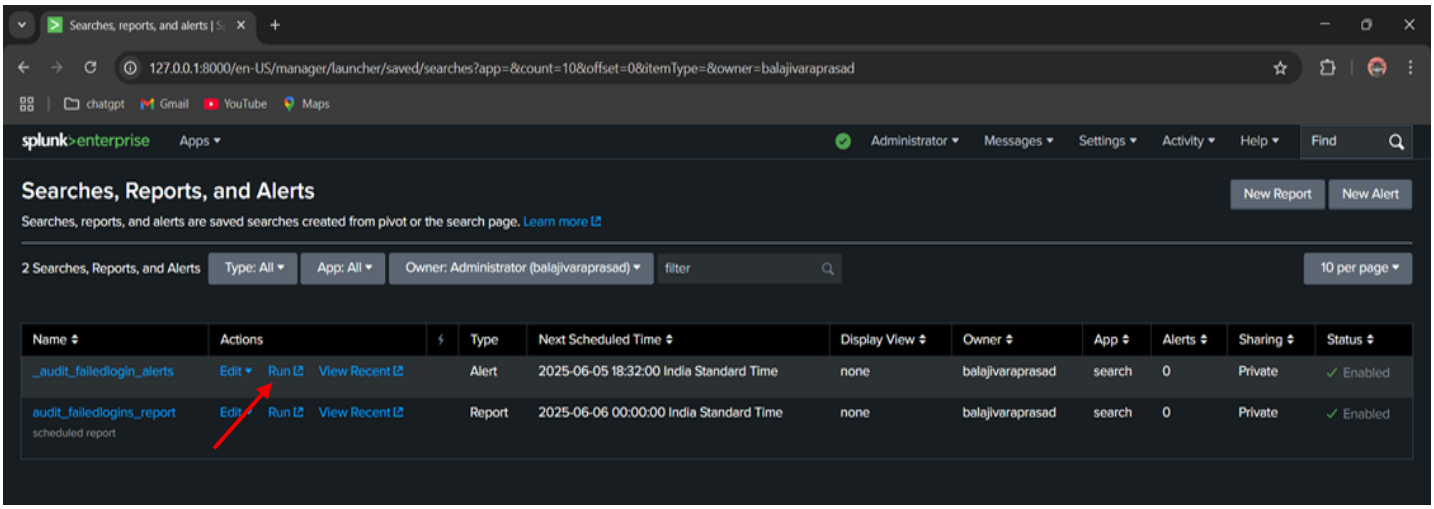
Step 22:

Now, we can still see there are zero alerts.



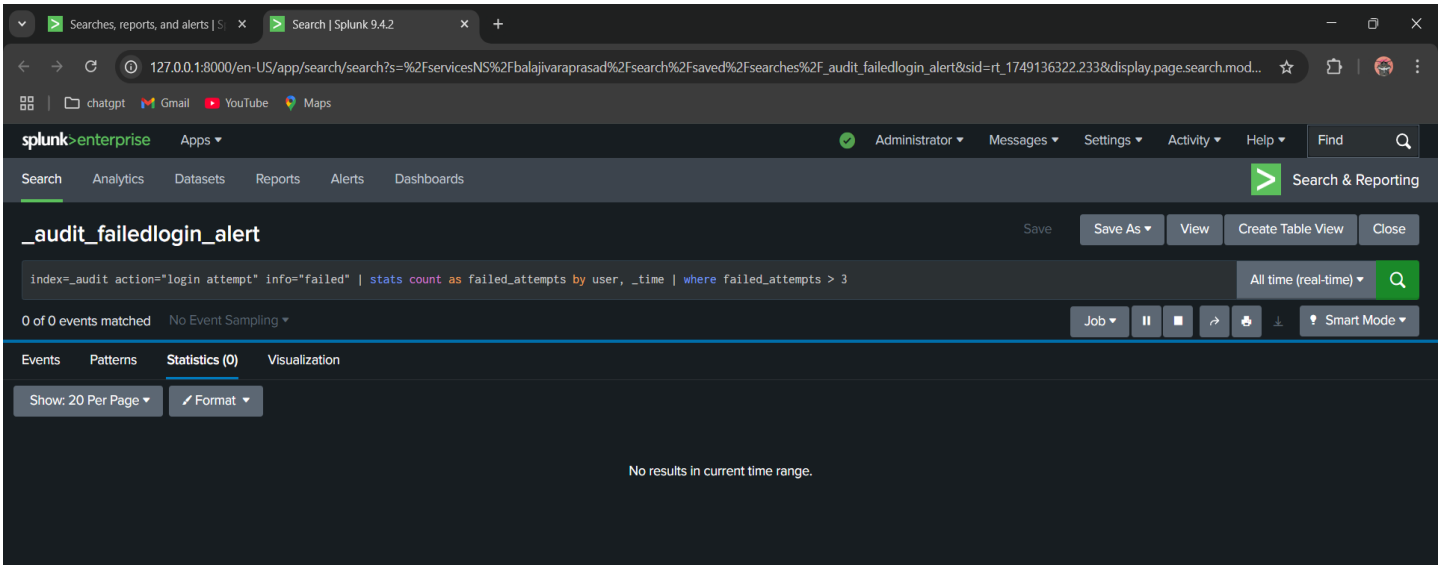
Step 23:

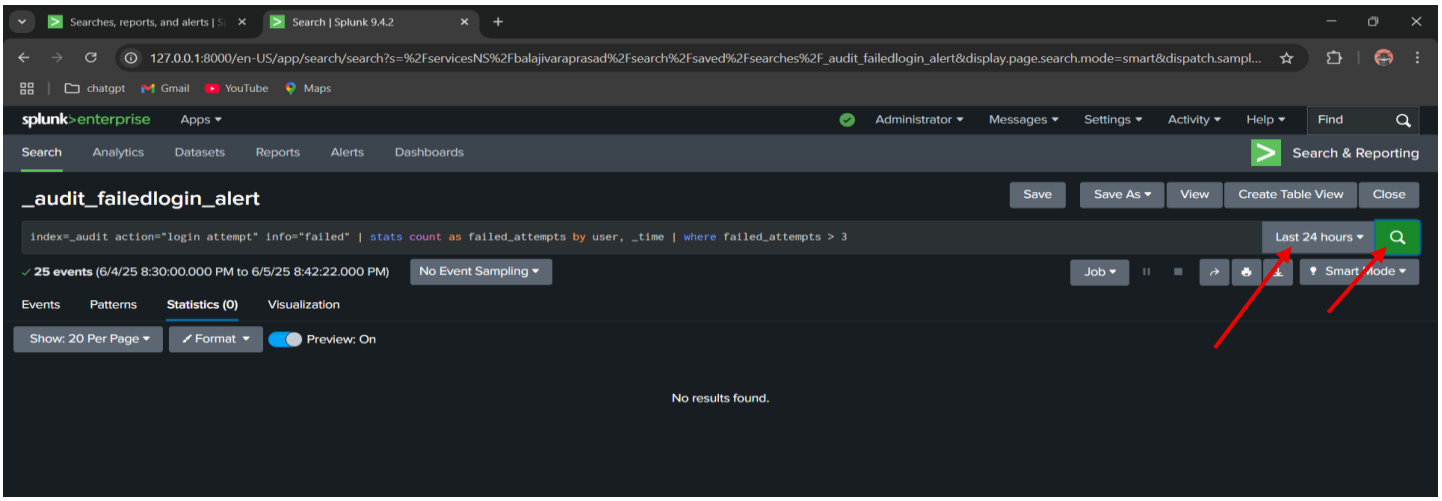
Now click on the run option to update the alerts.



Step 24:

You will now be redirected to the Search page. Set the time filter to Last 24 hours, then click on the Search button to execute the query.

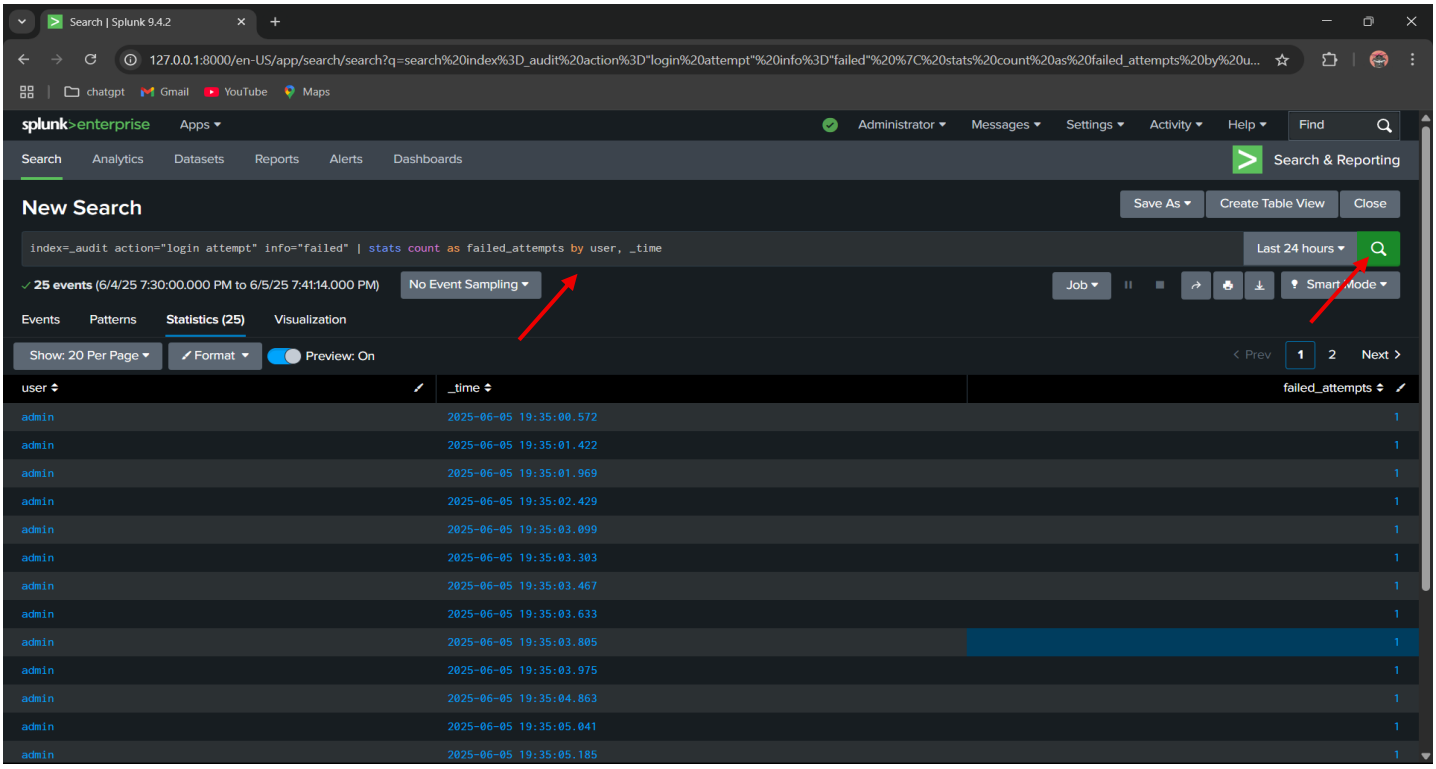




Step 25:

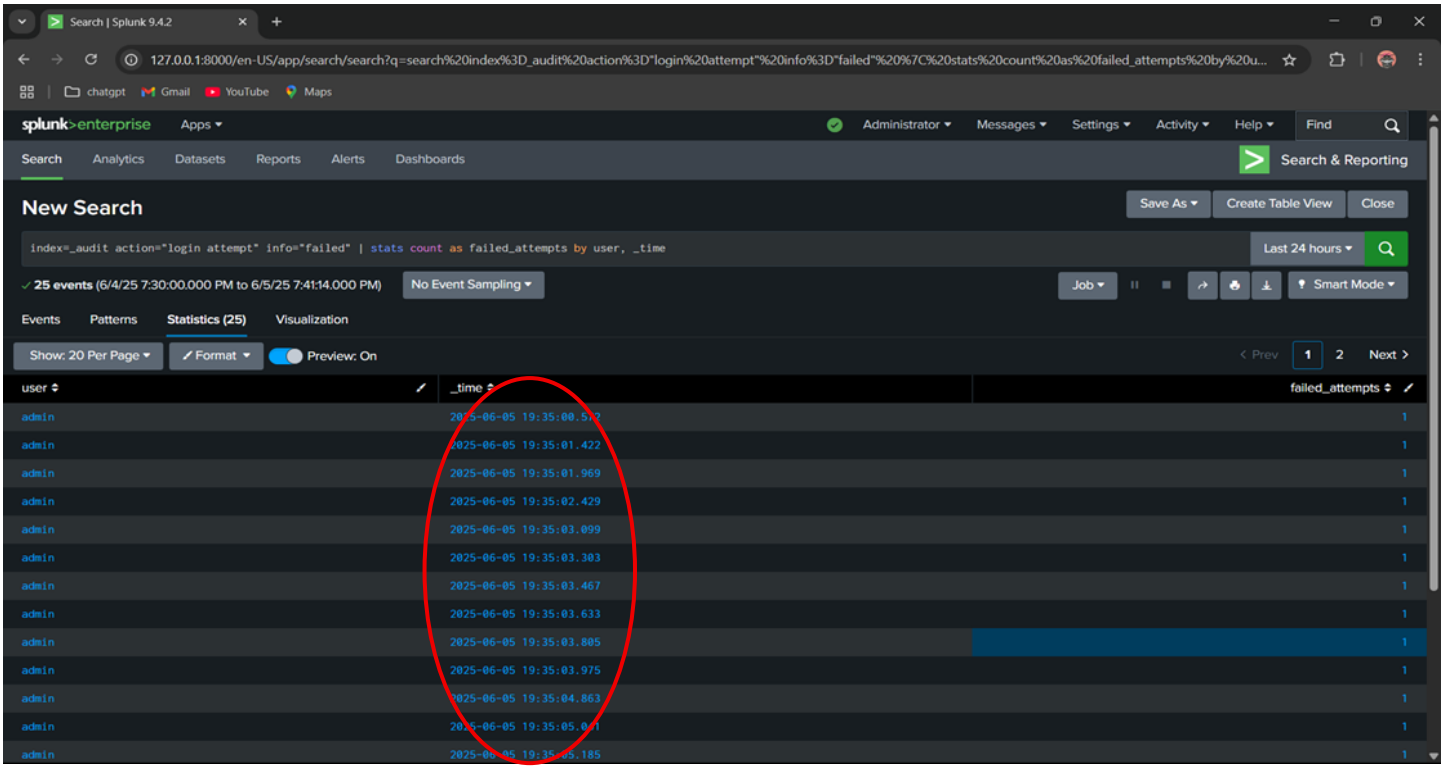
- Even though the time filter is set to Last 24 hours, you may see a message indicating “No results found.”
- Let’s troubleshoot this issue. The most likely causes are:
  1. An error in the SPL (Search Processing Language) query.
  2. Improper filtering or thresholds that are excluding the relevant data.

- However, we can see that the number of events has increased to **25**, which confirms that the failed login attempts we simulated have been successfully captured in the audit logs.
- To verify this, let's simplify the query by temporarily removing the threshold condition from the main SPL query and rerun the search to check if the events appear.



Step 26:

- After running the updated query, the data appears as expected. This confirms that the issue was due to the use of _time (timestamp) in the SPL query.
- When _time is included in the stats or by clause, each failed login attempt is recorded as a separate event, even for the same user, because _time differentiates entries down to the millisecond.
- As a result, the data was fragmented across multiple timestamps, preventing the alert from being triggered, even though there were multiple failed login attempts by the same user.



Step 27:

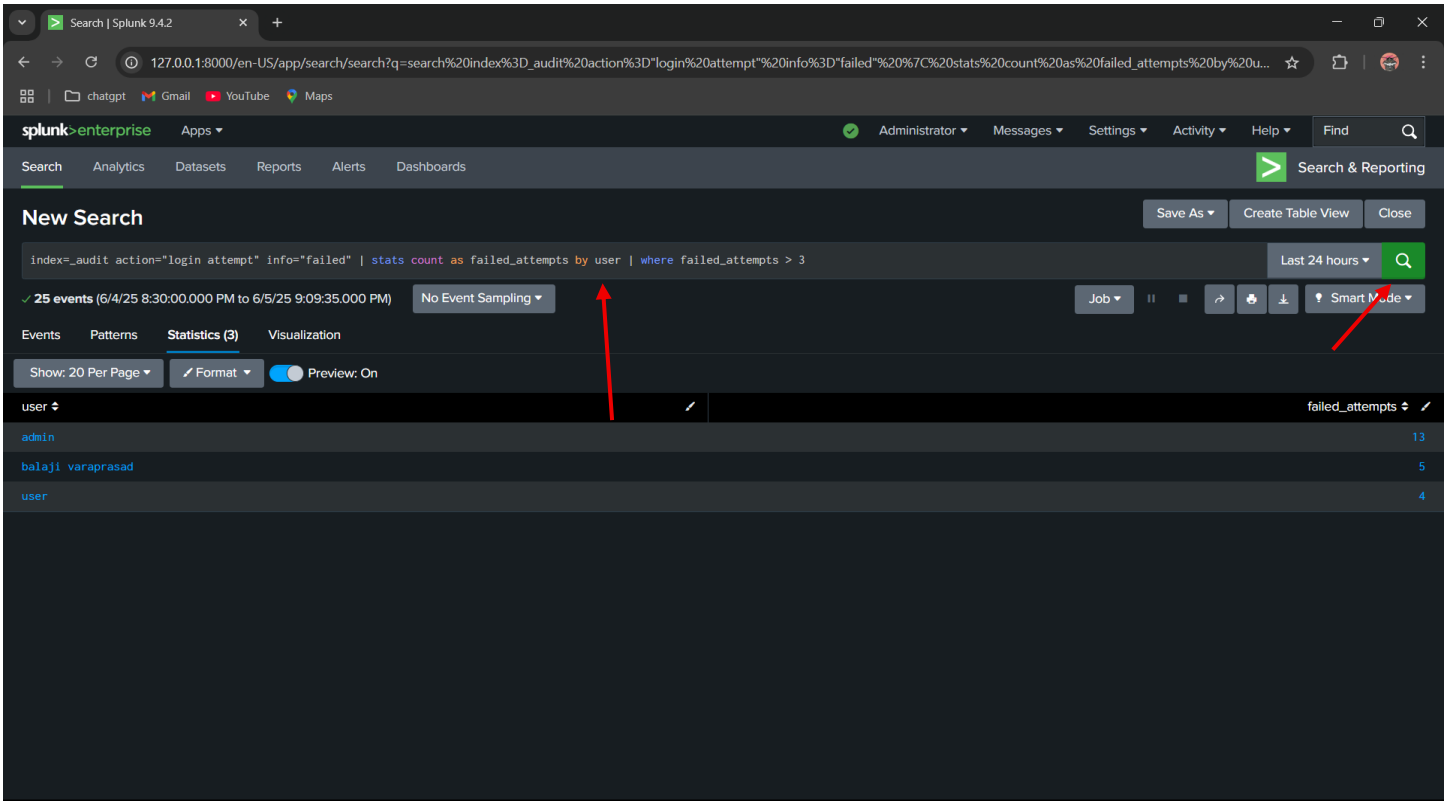
Now, let's correct the SPL query and create a new version that properly groups the failed login attempts. This updated query will ensure that the alert is triggered when the defined threshold is reached.

Step 28:

- Use the following updated SPL query to detect users with more than three failed login attempts**:**
- **index=_audit action="login attempt" info="failed"| stats count as failed_attempts by user| where failed_attempts > 3**
- Enter this query into the search bar and run it.
- The key difference in this version is that we removed _time from the stats clause, which allows the query to group all failed attempts by user, rather than splitting them by timestamp. This ensures the alert will be triggered once the threshold is exceeded.

Step 29:

You can now clearly see that the data has been generated as expected. The query successfully groups failed login attempts by user, making it suitable for setting up an alert based on the defined threshold**.**



Step 30:

- Now, delete the previously created alert to avoid confusion. To delete an alert:

1. Go to Settings and navigate to the Alerts section.
2. Locate the alert you created earlier.
3. Click on the Edit option next to it.
4. From the dropdown, select Delete to remove the alert.

Step 31:

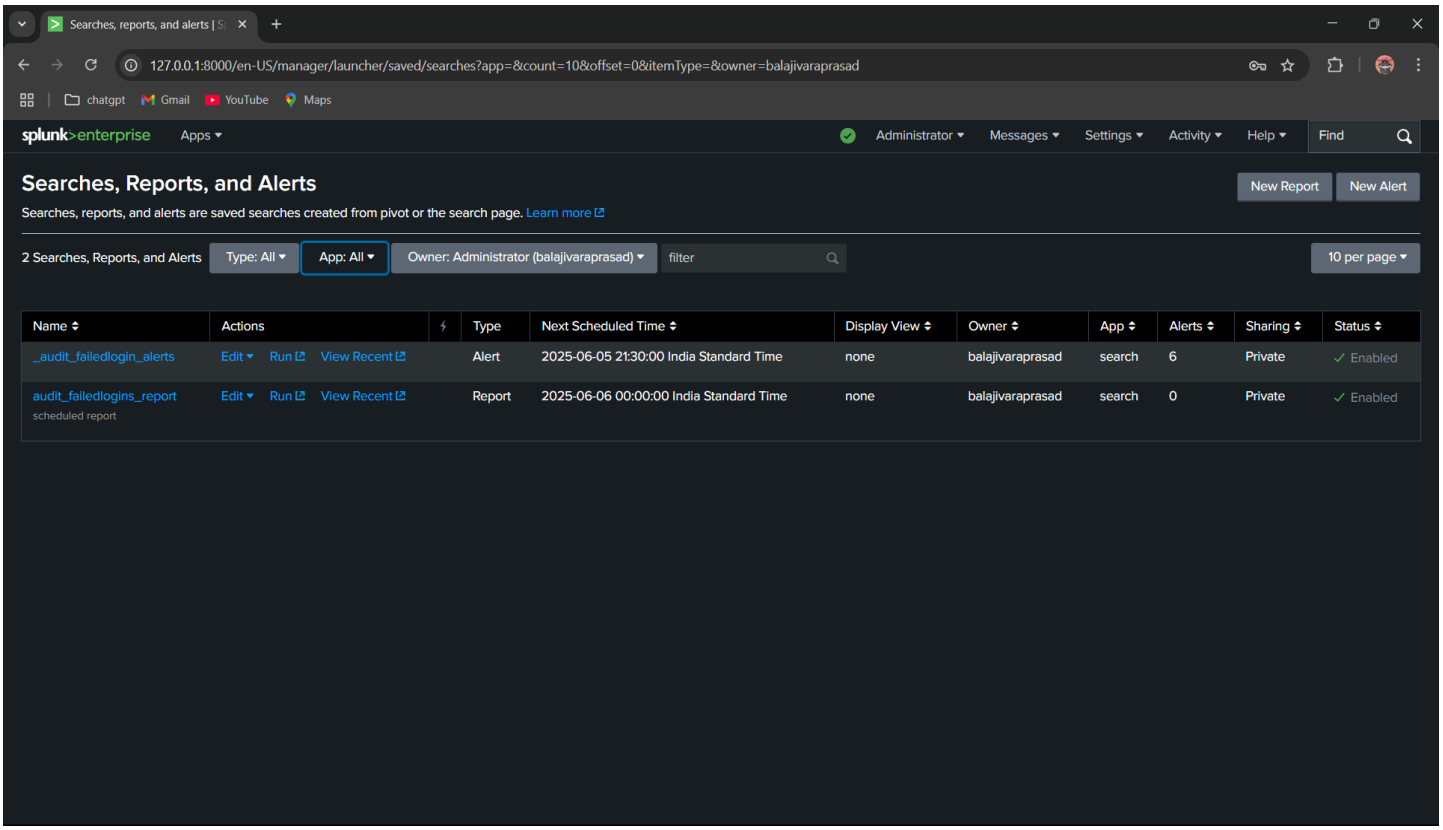
To create the alert, please follow the instructions outlined in Steps 5 through 18.

Step 32:

Now, simulate the failed login attempts again by following the procedure described in Steps 19 through 22. Try to use different usernames this time.

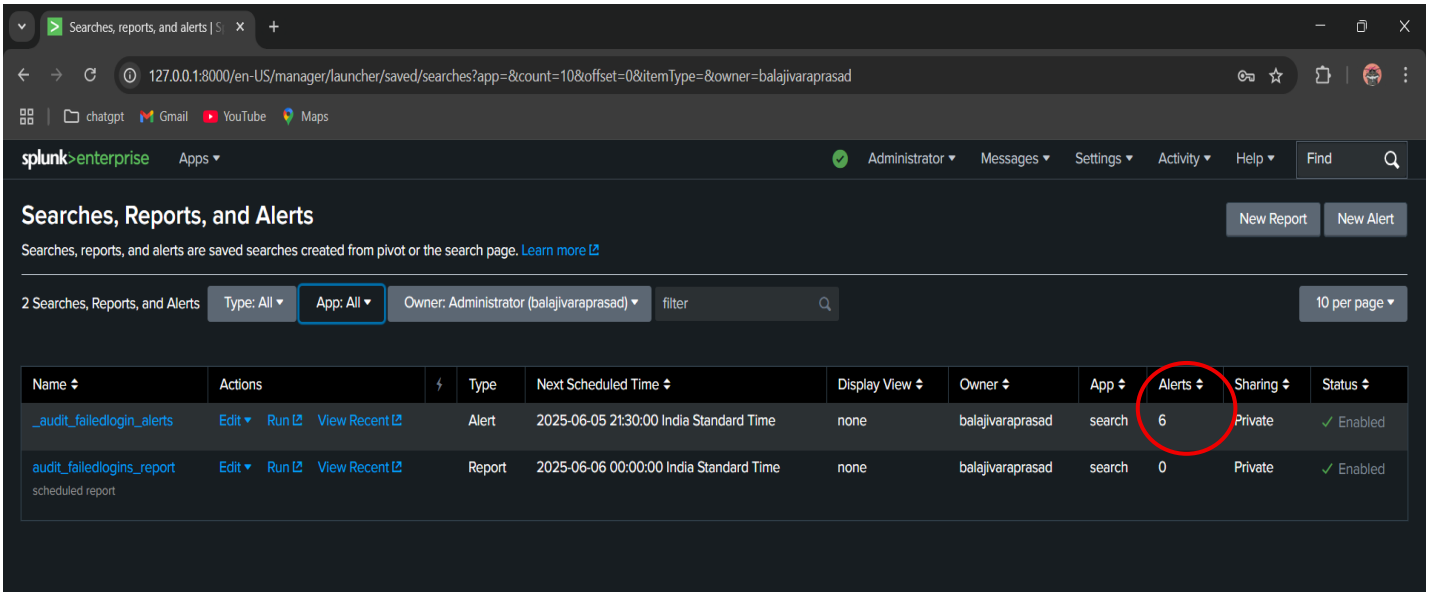
Step 33:

After navigating to Search, Reports, and Alerts in the Settings, you can view the alert created using the new SPL query.



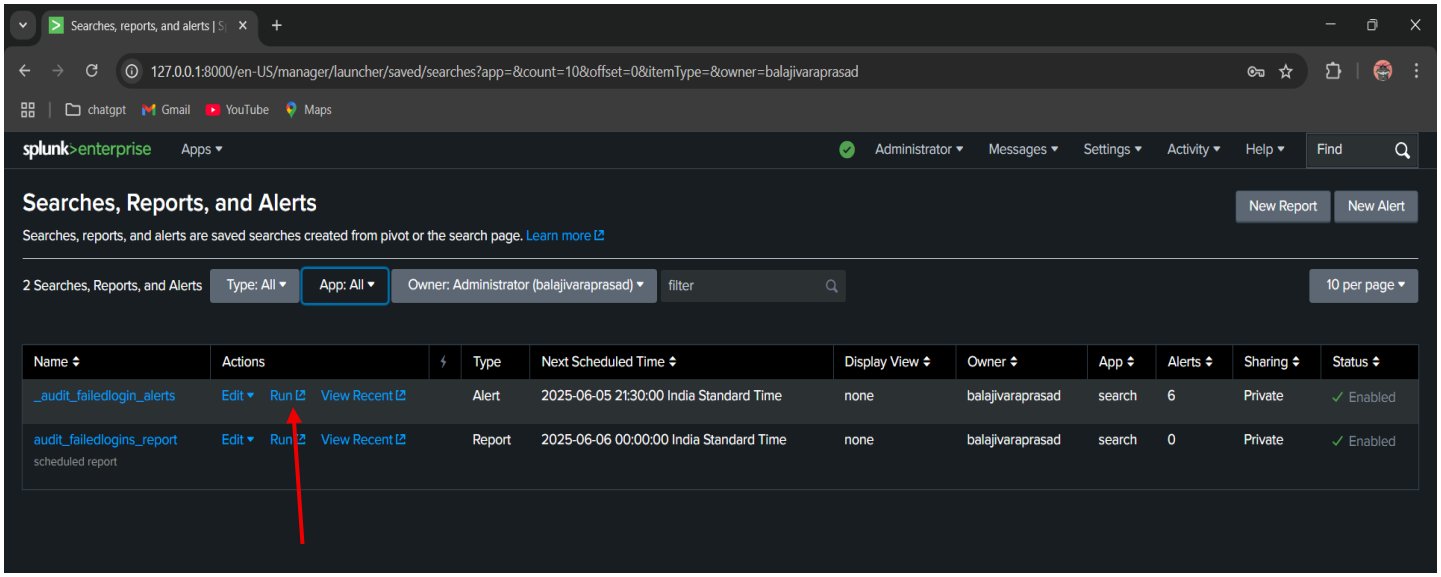
Step 34:

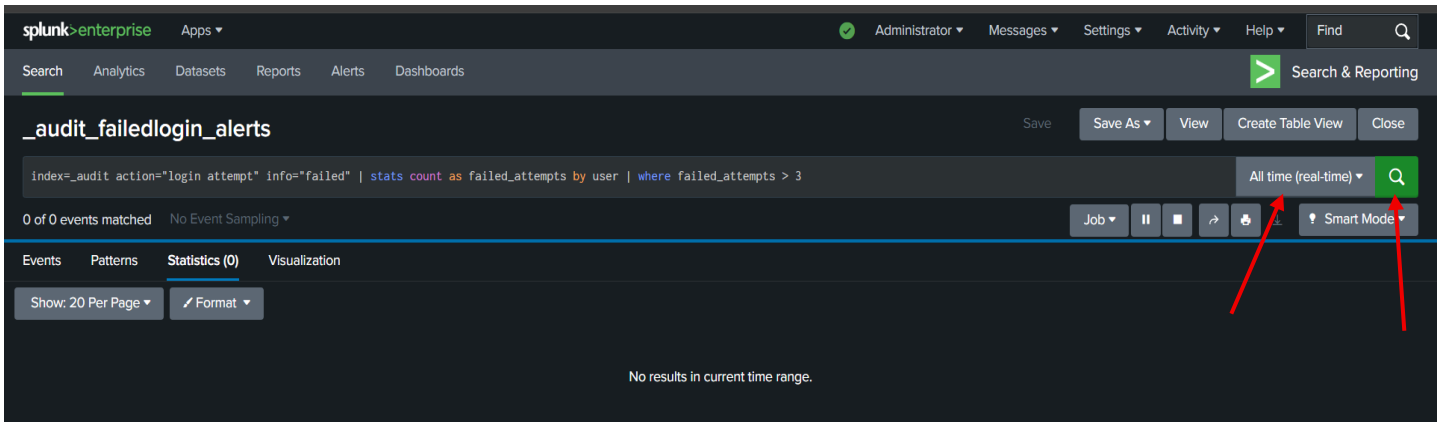
You can now see that six alerts have been triggered in the alert you created.

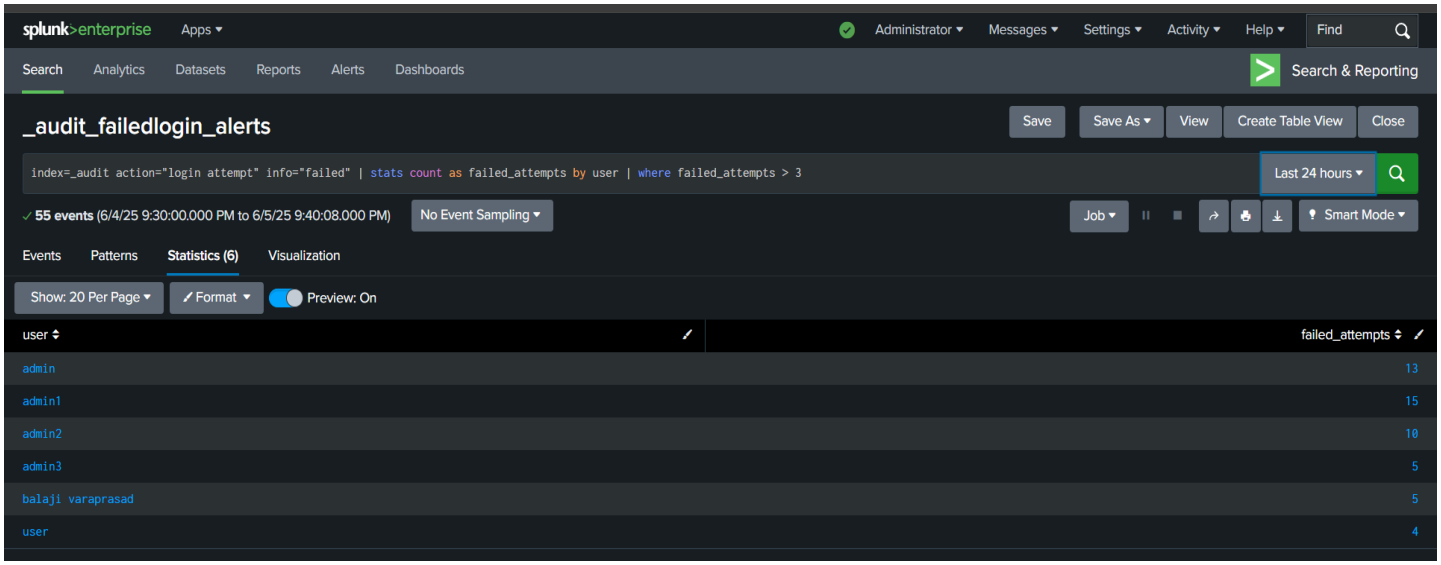


Step 35:

Click on the Run option, set the time filter to the last 24 hours, and then click Search. You will now be able to view the data from the simulated failed login attempts.







We successfully created a real-time alert in Splunk to monitor failed login attempts exceeding a defined threshold. By setting a condition to trigger the alert when any user experiences more than three failed login attempts, the alert helps in promptly identifying potential unauthorized access attempts. The configuration includes specifying a meaningful title, setting the alert type to real-time, and defining the trigger condition as "per result." Though no action was added in this instance, the alert is now active and will execute whenever the condition is met, allowing for quick visibility into critical security events.

Alert Actions and Escalation:

Once triggered, Splunk can notify teams or systems via alert actions. For example:

- **Email Notification:** Splunk can send an email with alert details to specified recipients. (You must configure SMTP in Splunk's Settings ➔ Server Settings ➔ Email.) Emails can include the search results or a summary.
- **Webhooks/Integrations:** Use the webhook or custom alert actions (via Splunkbase apps) to invoke external services. For instance, you might call a ServiceNow API to open an incident ticket, or post to a Slack/PagerDuty webhook. Splunk also offers an official ServiceNow and PagerDuty add-on for automated ticket creation.
- **Custom Scripts**: You can run a script action to perform any automated response (e.g. restart a service, disable an account) when the alert fires.

To handle escalation if the alert is not addressed, consider:

- **Repeated Notifications:** Configure the alert to repeat (e.g. daily) until the condition clears, or schedule a follow-up search. This ensures ongoing issues keep alerting.
- **On-Call Systems:** Route alerts into an on-call/incident management tool (like Splunk On-Call/VictorOps or PagerDuty). These platforms support multi-level escalation policies so that if the first responder doesn't acknowledge the incident, it automatically pages the next person.
- **Ticketing Integration:** Auto-create a ticket in ITSM (ServiceNow, Jira, etc.) with the alert details. The ticket system can then enforce SLA-based escalation (reminders, re-assignments) if the issue remains open.

By combining Splunk's alert actions with external tools, you can ensure that a "failed login" alert not only notifies the team immediately (via email or chat) but also gets tracked in a ticketing system or on-call rotation. Splunk's alert framework makes it easy to plug in these mechanisms, for example, an email alert action will "send an email notification to specified recipients when an alert triggers", and similar actions exist for scripts or webhooks.

Conclusion:

Scheduling reports and alerts in Splunk is a fundamental practice that significantly enhances operational efficiency, security monitoring, and real-time data analysis. By automating the process of generating insights, such as tracking failed login attempts, organizations can ensure continuous visibility into critical events without manual intervention. This not only saves time but also ensures timely detection and response to anomalies or threats, such as unauthorized access attempts. The integration of alerts with trigger conditions and automated actions like email notifications further empowers teams to stay proactive and maintain high system reliability.

Moreover, the detailed step-by-step implementation covered in this guide demonstrates that setting up such monitoring mechanisms in Splunk is both practical and accessible, even for those without deep technical expertise. Splunk's intuitive interface, combined with its powerful Search Processing Language (SPL), enables users to create targeted reports and effective alerts tailored to their operational or security requirements. In today's data-driven world, leveraging such tools ensures that important insights are not only uncovered but also acted upon, paving the way for smarter, faster decision-making.

Reference:

https://docs.splunk.com/Documentation/Splunk/9.4.2/Alert/Definescheduledalerts#:~:text=Create%20scheduled%20alerts

https://docs.splunk.com/Documentation/Splunk/9.4.2/Alert/Definescheduledalerts#:~:text=1,This%20setting%20controls%20the%20lifespan

https://docs.splunk.com/Documentation/Splunk/9.4.2/Alert/Definescheduledalerts#:~:text=8,happen%20when%20the%20alert%20triggers

https://docs.splunk.com/Documentation/Splunk/9.4.2/Security/Searchforauditevents#:~:text=1

https://docs.splunk.com/Documentation/Splunk/9.4.2/Alert/Emailnotification#:~:text=Send%20an%20email%20notification%20to,directly%20in%20a%20search%20command

THANK YOU