**Technical Project Report 1**

**PROMETHEUS DASHBOARD WITH ALERTING**

1. **A discussion about the case**

Sturdy Stirrups Inc is a leading company in the riding equipment across the globe. In the recent past, online competitors have increased significantly. Therefore there is a need in improving the marketing and sales awareness. Majority of the sales of the company are from the stores but the sales from the online store are growing at a steady rate.[1]

The challenge is that they are not able to visualize and keep track of where they are selling the most and from where the most profits are coming from. They are not able to see the real time sales data from across the world at a single place. Because of this lacking in the real time data visualization, they are not able to analyze, monitor and thereby not able to make the necessary business decisions which could enhance their sales growth and profit margins.[1]

Although the company does have access to the sales records and the profit margins from the retailer APIs, but the data is currently not collected in a centralized entity and they do not have the best possible way to display the information to provide useful statistics. For this particular case, the dashboard solution would be ideal for the main offices as well as for the local sales offices to view the real time sales data.[1]

The company has started using the cloud based collaboration between their offices. Also, they use an online chat communication tool to communicate between the offices. Another add on feature which would be helpful is to provide the alert messages via the chat channel when the sales data or profit margins reaches a particular threshold limit. [1]

This case is about the difficulty to display the sales and profits margin data metrics across different offices. There is also inability to view the huge data at a single glance and not able to monitor the data. This relates to the task of collecting the required metrics, displaying the data metrics on the dashboard, monitoring the data on certain thresholds limits or conditions, further sending the alerts to the respective group via different channels.

To solve the problem, a monitoring and dash boarding solution would be designed and build. A dashboard solution provides the visualization of the Key Performance Indicators (KPI) and other organizational and strategic data at a glance. A dashboard is a tool which provides the real information from the statistical point of view to the organization.

The combination of tools such as Docker, Prometheus and Grafana are useful to build up a dashboard with monitoring system. A dashboard solution would display a graph of the sales data in the local offices as well as an overview summary of the sales data in the main offices. The solution would also facilitate the communication of the alert notifications sent via Microsoft Teams chat channel in case if the threshold limits are reached.

**2. A technical design**

For the sake of providing the necessary solution to the above problem statement, an artificial data source is created which would resemble the data which is closely similar to the sales data in context of the problem case. The data source needs to be something which is neither completely static nor fully randomized data and should be a time series data. The data needs to be relevant to the problem and meaningful to the case discussed earlier. The dataset used is the data of currency exchange rates for different currencies, which would be the dataset for each of the local offices. This dataset varies steadily with time. [1]

For prototyping the technical design, there would be a Main Office and three local offices situated in the cities Oslo, Copenhagen and Stockholm. The Dashboard solution would collect the metrics and display them on different dashboards. The main office dashboard would be display the overall summary view of all the local offices. The metrics from the local offices would be extracted and displayed on the local dashboard.

The alerting messages would also be sent to the team through a chat room when the sales data reaches a threshold value in the local offices or the total sales of all the local offices reaches a threshold value. This would be implemented through Microsoft Teams. This would notify the managers or employees about offices where the sales and profit margins were highest or where the sales didn’t do well and need some action.

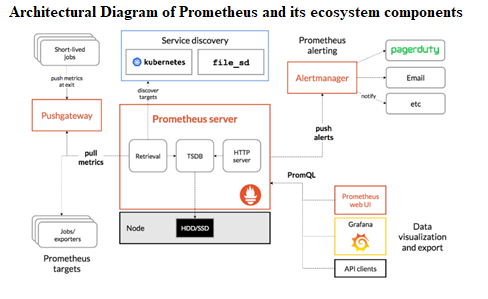
The tools that would be used to build the solution are Docker, Prometheus, Grafana. Below are the background details for each of these tools and also how they would be helpful to address this case problem.

*Docker:* It is a tool which is designed to make it easier to create, deploy and run applications using containers. With the containers, developers can package up an application with all the parts required such as libraries, and other dependencies and ship it all as one package. In this problem case, the Docker would be used to create a packaging container which would contain[2]

*Prometheus:* Prometheus is a free software application used for event monitoring and alerting. It records real-time metrics in a time series database (allowing for high dimensionality) built using a HTTP pull model, with flexible queries and real-time alerting.[4]

The main features of Prometheus are as follows[4]:

* It is a data model with time series data which is identified by metric name and key/value pairs and the data retrieval happens through a pull request over HTTP.
* The PromQL is a flexible query language which facilitates this functionality.
* There is an intermediate gateway which supports the push of the time series data.
* The identification of the targets happens through service discovery or static configuration.



**Fig 1: Architectural diagram of Prometheus and its ecosystem components. Source [3]**

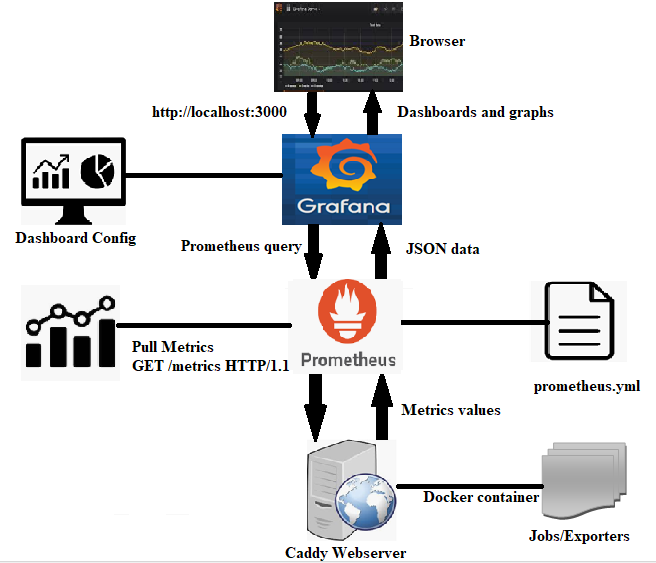
As in shown in the Fig 1, Prometheus ecosystem contains a Prometheus server which scrapes the data and stores time series data. There are client libraries in Prometheus which help in matching the language of the application and this helps in defining and exposing the internal metrics through HTTP endpoint on the application’s instance. Examples of the client libraries include Go, Java, Scala, Python, Ruby. There is also a push gateway handling the short lived jobs. There are other special purpose exporter services such as HAProxy, StatsD , Graphite, etc. A component alertmanager is for managing the alerts.[4]

*Grafana:* Grafana is an open source metric analytics & visualization suite. It is most commonly used for visualizing time series data for infrastructure and application analytics.[7]

The Fig2 below illustrates the interaction of various components in the technical design of the prototype solution for prometheus dashboard with monitoring. A Docker container which is created serves the data from the metrics file in the exporter\_data directory to the caddy webserver. The prometheus.yml file is configured with a new job\_name in the scrape\_configs section to scrape the metrics data to Grafana. A scrape\_config section defines a set of targets and parameters describing how to scrape them. [6]

A Prometheus webserver is configured with the new version of the prometheus.yml file which pulls the metrics data from the caddy webserver through GET HTTP query. Grafana which runs on the <http://localhost:3000> sends a Prometheus query to receive the metrics to the Prometheus server. The Prometheus server sends the metrics data received from the caddy webserver in the form of JSON data to Grafana. Grafana is configured with the data source as Prometheus <http://prometheus:9090>. A dashboard is configured in Grafana with graph panels providing metrics name and legend format name along with alert notifications. The graph for the time series data is displayed on Grafana dashboard.[8]

Interaction between Various Technical Components



**Figure 2 Technical flow design of the proposed prototype, Ref [5]**

The below implementation logic flowchart in Fig 3 demonstrates the steps that need to be performed for creation of Prometheus dashboard with monitoring solution for the case of Sturdy Stirrups Inc. The below steps are involved:[9][10][11][12]

* *Installing Docker on Ubuntu 18.04, initialize swarm and download swarmprom*: Swarmprom is a starter kit for Docker swarm monitoring with Prometheus, Grafana, node exporter, Alert manager.

The Prometheus runs as a Docker service on a Docker swarm. Multiple docker engines are joined into a docker swarm using docker swarm init on one manager and docker swarm join on other managers and worker nodes. [https://docs.docker.com/thirdparty]

* *Starting the entire slack in swarmprom directory:* This would install all the components :alert\_manager, caddy, cadvisor, dockerd-exporter, grafana, node-exporter, prometheus and unsee.
* *Creating a custom metrics file:*  This shell file would contain the script for extracting the currency exchange rates information from the <http://transferwise.com> for three different currency exchange rates euro-to-inr ,gbp-to-inr, usd-to-inr. This would be the artificial data source corresponding to the Local Offices in Oslo, Copenhagen and Stockholm. This file would be made executable.
* *Create a simple exporter service:* A simple exporter service or the target would be created which would collect the metrics and view the data on the Grafana dashboard
* *Creating a directory for exporter service/target and metrics file:* A directory folder named “exporter\_data” is created in /var/lib/. A metrics file is created inside the folder and it is made executable.

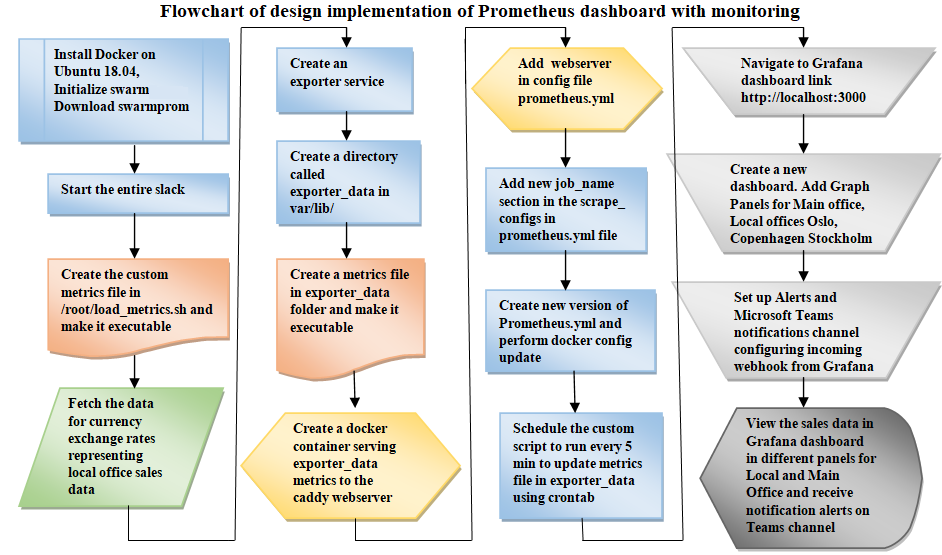


Figure 3 Flowchart of implementation process steps for creating Prometheus dashboard with monitoring

* *Create a docker container serving metrics file in exporter\_data to caddy webserver:* A docker container is created which would collect the metrics data from exporter\_data folder and serve it to caddy webserver.
* *Adding new job\_name in the scrape\_configs section in prometheus.yml:*  A new job name would be inserted which would scrape the metrics data based on the parameters defined in the prometheus.yml config file to the prometheus server.
* *Perform docker config update:*  A new version of prometheus.yml.v1 file is created and docker config is updated with the source as prometheus.yml.v1 and target given for prometheus server.
* *Schedule script to run every 5 min to update metrics file in exporter\_data using crontab:*Using crontab, the output contents of the script would be updated in the metrics file in exporter\_data every 5 min. This would fetch and update custom metrics data in metrics file after every 5 min.
* *Create Dashboard and Graph Panels in Grafana:* The Grafana dashboard is opened with link <http://localhost:3000> and a dashboard named Sturdy Stirrups Inc. is created. Four graph panels are added with the metrics and legend information. One graph for Main Office Sales data which would display the Sales data overview summary of the Local Offices. The other three graph panels would display the sales data in local offices of Oslo, Copenhagen and Stockholm. A playlist can be created which would rotate the different dashboards to show the metrics effectively to the team members.
* *Create Alerts in Dashboard:*  The alerts would be created for the Local Offices with upper and lower threshold limits respectively for each Local Office. For Main office, the alerts would be set similarly considering the total sales from all the Local offices. If the sales data value is beyond upper threshold limit, it indicates the high number of sales and higher profit margins. If sales data value goes lesser than the lower threshold limit, it indicates the sales and profit margins were less and it needs certain attention.
* *Sending Notification Alerts in Microsoft Teams Channels:* A incoming webhook would be configured in Micorosft Teams channel to receive notification alerts from Grafana. In Grafana, this webhook would be configured in the notification channel send the alerts to Microsoft Teams channel.

With the above set up, Grafana dashboard would display the different graphs for Main office overview summary data and local offices sales data with the notification alerts sent to Microsoft Teams channel.

**3. Pilot Implementation**

For demonstrating the implementation prototype for problem of inability to view the huge data effectively, a sample data is created and not the real company data. Currency exchange rates data for different currency conversion is used to orchestrate the implementation of the proposed solution. A currency exchange rates data is selected for this discussion case which depicts the dynamic behaviour of the change in the currency conversion rates between two different currencies with time. The data used here is just sample data for implementing the prototype, which further can be replaced with the real data for testing the real company scenario. The data is taken is neither completely static nor completely randomized which is the case with the currency exchange rates as they change steadily with time. This currency data was chosen since it closely resembles the company sales data which also changes dynamically with time period.

The Grafana dashboard would display the sales data metrics for company Sturdy Stirrups Inc. for Local Offices in Oslo, Copenhagen and Stockholm and Main office displaying overview summary report and send notification alert messages in Teams channel.

Below are the implementation steps performed in the system, based on the tutorial references [9][10][11][12]:

***Step 1 Installing Docker on Ubuntu 18.04 server***

The first step is to install Docker on Ubuntu 18.04 server. Following are the commands to do the same. The command sudo su would switch to the root user.

**sudo su**

**apt-get update**

**apt-install docker-ce**

To verify whether Docker is installed successfully or not, run the below command to test it.

**docker ps**

***Step 2 Initialize a swarm***

**docker swarm init**

This command will initiate this server to be the manager of a swarm of docker servers. Despite having only one server, this swarm-mode is required to open up more features of docker.

***Step 3 Download Swarmprom***

**git clone** <https://github.com/stefanprodan/swarmprom.git>

This command would download the swarmprom components from the given github directory link

***Step 4 Start the entire slack***

The command cd swarmprom would change to the swarmprom directory and the next command would start the entire slack for the first time

**cd swarmprom**

**docker stack deploy –c docker-compose.yml mon**

It takes few minutes until all components are getting downloaded and running. The progress can be checked using the below command

**docker service ls**

The output of the above command is displayed on the screen as below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **NAME** | **MODE** | **REPLICAS** | **IMAGE** | **PORTS** |
| **dauwj6byl8ud** | **mon\_alertmanager** | **replicated** | **1/1** | **stefanprodan/swarmprom-alertmanager:v0.14.0** | **\*:3000->3000/tcp, \*:9090->9090/tcp, \*:9093-9094->9093-9094/tcp** |
| **z0a3sj48yn2e** | **mon\_caddy** | **replicated** | **1/1** | **stefanprodan/caddy:latest** |  |
| **uzj7z6d2lky7** | **mon\_cadvisor** | **global** | **1/1** | **google/cadvisor:latest** |  |
| **5qb8lcso5gx0** | **mon\_dockerd-exporter** | **global** | **1/1** | **stefanprodan/caddy:latest** |  |
| **yy0ic8mhww2x** | **mon\_grafana** | **replicated** | **1/1** | **stefanprodan/swarmprom-grafana:5.3.4** |  |
| **i9rj7sk6t3r8** | **mon\_node-exporter** | **global** | **1/1** | **stefanprodan/swarmprom-node-exporter:v0.16.0** |  |
| **4nrc16afac97** | **mon\_prometheus** | **replicated** | **1/1** | **stefanprodan/swarmprom-prometheus:v2.5.0** |  |
| **rl64qvaqtwxb** | **mon\_unsee** | **replicated** | **1/1** | **cloudflare/unsee:v0.8.0** |  |

***Step 5 Connect to the Grafana user interface***

To verify if the Grafana is running properly on the port 3000 can be checked in two ways.

The URL link to navigate to is of the format - https://IP address : Port number.

If the IP of the open stack instance is 128.39.121.133 then the URL to check and verify would be <https://128.39.121.133:3000>. This URL can be navigated in the web browser to check the status.

If it is on the local machine, then the link can be accessed via the URL http://localhost:3000

The Grafana dashboard can be accessed for the first time with the initial default username which is admin and password which is admin. The user would be prompted to change the password.

***Step 6 Create a custom metrics file***

For getting the custom metrics data for currency exchange rates of different currencies to create artificial data source, a shell script file named load\_metrics.sh is created in the directory /root with the below command.

**nano /root/load\_metrics.sh**

This would collect the custom metrics data pertaining to this discussion case about Sturdy Stirrup Inc.

A code snippet of shell script file contents is as follows:

1. **#!/bin/sh**
2. **EUR\_INR=$(curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "EUR-INR" |sed -e 's/.\*"EUR-INR">//'|sed -e 's/<.\*//')**
3. **GBP\_INR=$(curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "GBP-INR" |sed -e 's/.\*"GBP-INR">//'|sed -e 's/<.\*//')**
4. **USD\_INR=$(curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "USD-INR" |sed -e 's/.\*"USD-INR">//'|sed -e 's/<.\*//')**

**Figure 4: Code snippet of custom shell script**

The statement 2 above would get the data website link <https://transferwise.com/gb/currency-converter/nok-to-inr-rate>” with the curl command.

For the highlighted statement in red above, the shell command until grep “EUR-INR” command extracts the entire line for currency exchange rate for euros to inr.

**curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "EUR-INR" |sed -e 's/.\*"EUR-INR">//'|sed -e 's/<.\*//')**

The output is shown as below.

**<a class=”js**-**TopCurrenciesLink currency-table\_\_link" href="/gb/currency-converter/eur-to-inr-rate" data-top-currencies= EUR-INR">78.74150</a>**

But, only a single numeric value for the exchange rates is required, so the rate 78.74150 needs to be extracted. And the string before and after the number 78.74150 needs to be trimmed.

The below command highlighted in yellow would trim the string before 78.74150

**curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "EUR-INR" |sed -e 's/.\*"EUR-INR">//'|sed -e 's/<.\*//')**

Output is as follows:

**78.74150</a>**

For trimming the string part after 78.74150, the below command is executed.

**curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "EUR-INR" |sed -e 's/.\*"EUR-INR">//'|sed -e 's/<.\*//')**

The output of the above query would give a single value.

**78.74150**

Similarly, the line statements 3 and 4 in Fig 4 of code snippet would extract the currency exchange rates for gbp-to-inr, usd-to-inr.

The code statements from 1 to 8 in the Fig 5 would declare the metrics variables sales for Main office, and sales1, sales2 and sales3 for Local offices.

1. **echo "# TYPE sales gauge"**
2. **echo "# HELP sales Company sales data in Main office"**
3. **echo "# TYPE sales1 gauge"**
4. **echo "# HELP sales1 Company sales data in Oslo office"**
5. **echo "# TYPE sales2 gauge"**
6. **echo "# HELP sales2 Company sales data in Copenhagen office"**
7. **echo "# TYPE sales3 gauge"**
8. **echo "# HELP sales3 Company sales data in Stockholm office"**
9. **echo "sales{timeval=\"OsloSales\"} $EUR\_INR"**
10. **echo "sales{timeval=\"CopenhagenSales\"} $GBP\_INR"**
11. **echo "sales{timeval=\"StockholmSales\"} $USD\_INR"**
12. **echo "sales1{timeval1=\"SalesData\"} $EUR\_INR"**
13. **echo "sales2{timeval2=\"SalesData\"} $GBP\_INR"**
14. **echo "sales3{timeval3=\"SalesData\"} $USD\_INR"**

**Figure 5: Code snippet of custom metrics shell file**

The code statements in the Fig 5 from 9 to 10 would output the respective exchange currency rates.

The load\_metrics.sh script file is made executable with the below command.

**chmod +x /root/load\_metrics.sh**

Execute the script by running it with the below command

**./load\_metrics.sh**

Output of the script execution is as below

**# TYPE sales gauge**

**# HELP sales Company sales data across different local offices displayed in Main office**

**# TYPE sales1 gauge**

**# HELP sales1 Company sales data in Oslo office**

**# TYPE sales2 gauge**

**# HELP sales2 Company sales data in Copenhagen office**

**# TYPE sales3 gauge**

**# HELP sales3 Company sales data in Stockholm office**

**sales{timeval="OsloSales"} 78.74150**

**sales{timeval="CopenhagenSales"} 94.33480**

**sales{timeval="StockholmSales"} 70.79800**

**sales1{timeval1="SalesData"} 78.74150**

**sales2{timeval2="SalesData"} 94.33480**

**sales3{timeval3="SalesData"} 70.79800**

***Step 7 Create a simple exporter service***

To be able to collect the metrics and view the custom data on the Grafana dashboard, a target needs to be created. To set up this target, a simple Prometheus exporter service is created following the below steps.

Create a folder named “exporter\_data” under the directory /var/lib/ with the below command.

**mkdir /var/lib/exporter\_data**

A metrics file is created where the data would be exported. The below command creates the metrics file under the folder /var/lib/exporter\_data/ and opens the file in the nano editor to add the file contents.

**nano /var/lib/exporter\_data/metrics**

The file permissions are checked with the below command and the metrics file is made executable.

**ls -l /var/lib/exporter\_data/metrics  
  
chmod +x /var/lib/exporter\_data/metrics**

**Then create a docker container which would collect data from the exporter\_data folder and serve the data to the caddy webserver.**

**docker run -–name=myexporter -d --network=mon\_net –v /var/lib/exporter\_data:/srv abiosoft/caddy**

This command would make the metrics file available to the caddy webserver. With this command, a container with caddy webserver is started and it would start reporting the contents it finds in the folder /var/lib/exporter\_data in the local server. The network of this docker container “mon\_net” created is same as the network to which Prometheus container is connected. The Prometheus container uses the name “myexporter” given in the parameter –name=myexporter.

The container “myexporter” created is verified with the below command to check that it is running in active status along with other containers.

**docker ps**

**9beb30c977b4 abiosoft/caddy "/bin/parent caddy -…" 2 months ago Up 2 months 80/tcp, 443/tcp, 2015/tcp myexporter**

Verify the output of the logs generated for “myexporter” with the below command

**docker logs myexporter**

This would display the result something as below

**"GET /metrics HTTP/1.1" 200 133**

***Step 8 Modify init script and update prometheus with it***

The init script needs to be modified only once to add the multiple targets later.

Navigate to the /swarmprom/prometheus/conf/ folder

Edit the docker-entrypint.sh in nano editor, scroll down and search for the line as below

**mv /tmp/prometheus.yml /etc/prometheus/prometheus.yml**

Comment the above line by placing # at the beginning of the line.

**# mv /tmp/prometheus.yml /etc/prometheus/prometheus.yml**

A special docker config file is created with the below command.

**docker config create prometheus-docker-entrypoint.sh.v1 docker-entrypoint.sh**

Update the running prometheus container with the new version of the init script with the below command

**docker service update --config-add source=prometheus-docker-entrypoint.sh.v1,target=/etc/prometheus/docker-entrypoint.sh,mode=0755 mon\_prometheus**

The above command notifies docker, that one service will get a file inserted into all containers of that service. The source file mentioned would locally be available as /etc/prometheus/docker-entrypoint.sh with the correct permission code of 755. This approach works irrespective of number of containers and size of swarm.

***Step 9: Adding new target to prometheus***

A new webserver needs to be added to the Prometheus configs. To do that edit the Prometheus.yml file in the directory /swarmprom/prometheus/conf with the below commands.

**cd swarmprom/prometheus/conf/**  
  
**nano prometheus.yml**

Scroll to the section named “scrape\_configs:” and insert a new job\_name section.

**- job\_name: 'myexporter'**  
  
**scrape\_interval: 60s**  
  
**static\_configs:**  
  
 **- targets: [myexporter:2015]**

The next step is to create a new version of the Prometheus configuration and execute the docker config update with the below commands.

**docker config create prometheus.yml.v1 prometheus.yml**   
  
**docker service update --config-add source=prometheus.yml.v1,target=/etc/prometheus/prometheus.yml,mode=0644 mon\_prometheus**

After running the above docker service update command, success message is displayed as “Service Converged”.

***Step 10: Scheduling the script run***

This custom script output needs to continuously update the metrics file in the exporter\_Data folder automatically so that the Prometheus target is updated with the latest metrics data.

For this the script needs to be scheduled to run after every five minutes

This is done using the tool cron.

The cron service helps to schedule the execution of commands in Linux and Unix.

To ensure that the editor used is nano, below command is executed in shell.

export EDITOR=nano

**export EDITOR=nano**

Next command is executed as below. This would start a cron editing session.

**Crontab -e**

The below entry is added inside the nano editor just opened as follows: This entry would schedule the script every 5 minutes to run and would redirect the contents of the script to be stored in the metrics file under the exporter\_data folder . In this folder the “ myexporter webserver” is serving.

**\*/5 \* \* \* \* /root/load\_metric.sh > /var/lib/exporter\_data/metrics**

Once 5 minutes have passed, the metrics file can be verified to see if the contents are updated from the custom shell script.

**ls –l /var/lib/exporter\_data/metrics**

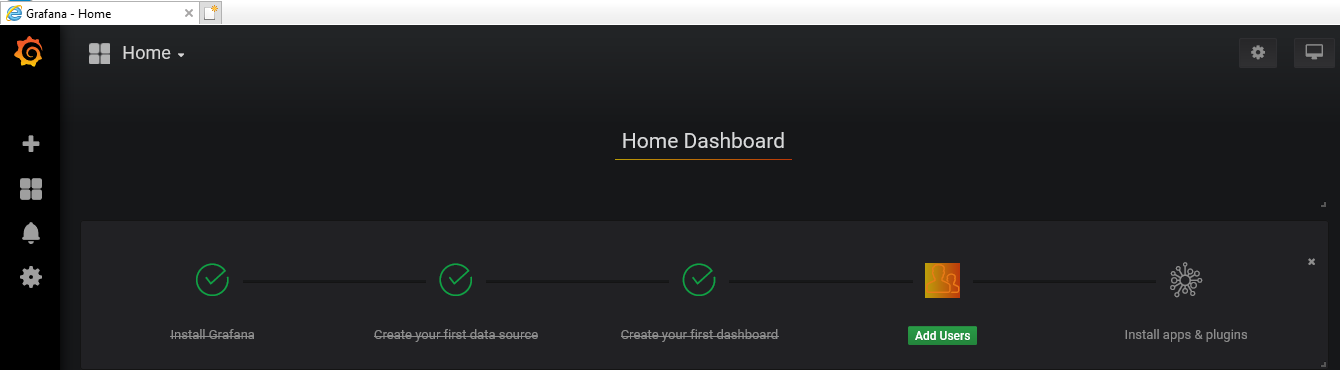
For this problem case, where the metrics data of revenue needs to be displayed on the dashboard from the local companies, the exporters created above would only run from the local machines. But here, the exporters should also run on other machines and Prometheus should scrape for the metrics data.

***Step 11 Adding the metrics to Grafana Dashboard*:**

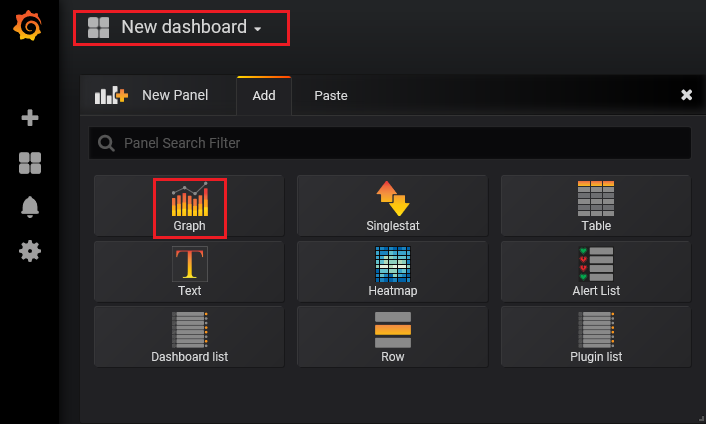
Following steps are carried out.

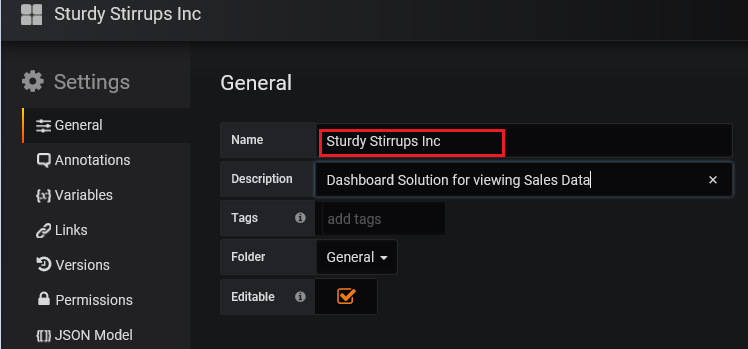
Go to the Grafana by navigating to the link http:// IP address : Port number 3000

In this case, it is <http://128.39.121.133:3000>



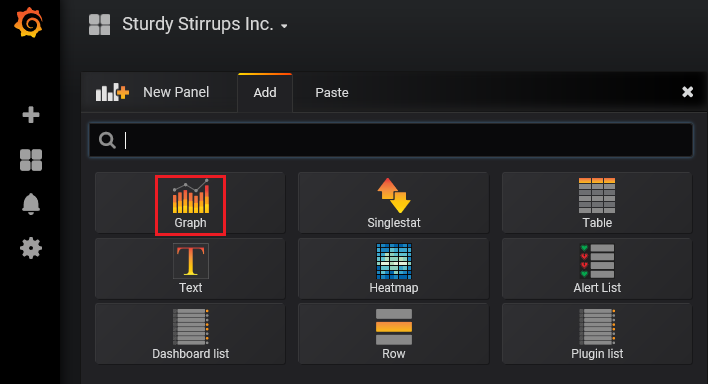
Create a new Dashboard with the name “Sturdy Stirrups Inc.” as below



****

*Add Graph Panel for displaying Overview data in Main Office:*

Then click on Add Panel and select Graph Panel by clicking on the Graph icon as in the below screenshot.





In the metrics field, **sales{}** is entered and in the Legend format, **{{timeval}}** is entered. The Graph Panel is named as Main Office Sales Data and the Panel is saved. The graph is then displayed with the timeline of last 6 hours as in the Figure 6 below. The Fig 6 shows the graph of Main Office displaying the overall summary of sales data across Local Offices situated in Oslo, Copenhagen and Stockholm.

Sturdy Stirrups Inc: Main Office Sales Data

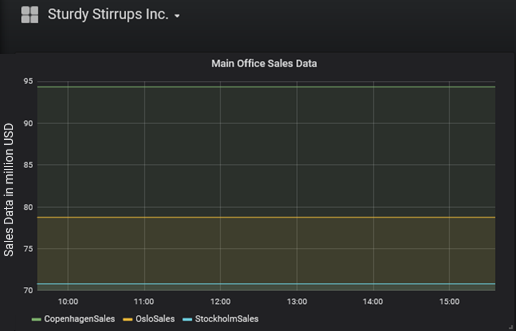
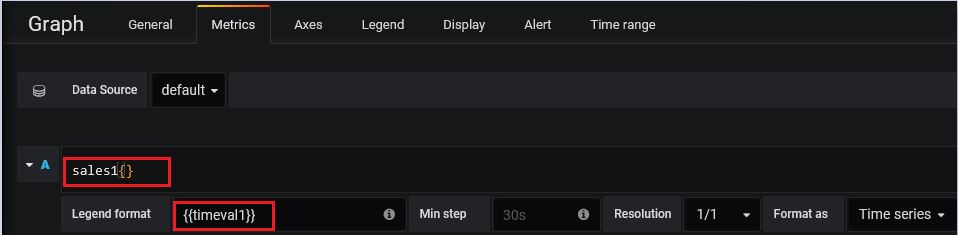


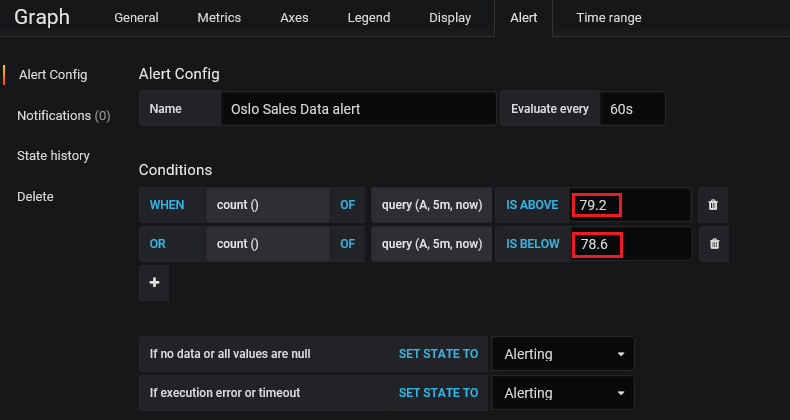
Figure 6 Graph displaying Main Office Sales Data Overview Summary

*Add Graph Panel for displaying Sales Data in Oslo*

Add the metrics as **sales1{}** and Legend format as **{{timeval1}}**

****

Setting up the Alerts: Alert config is set up on value of Oslo sales data going above 79.2 indicating higher number of sales and sales data going below 78.6 indicating the sales were not meeting the minimum expectations.

****

The Fig 7 shows the graph for Oslo Sales data in million dollars as a time series data over a period cycle of 6 hours and would trigger an alert when the value reaches above 79.2 or below 78.6.

Sturdy Stirrups Inc. Local Office: Oslo Sales Data

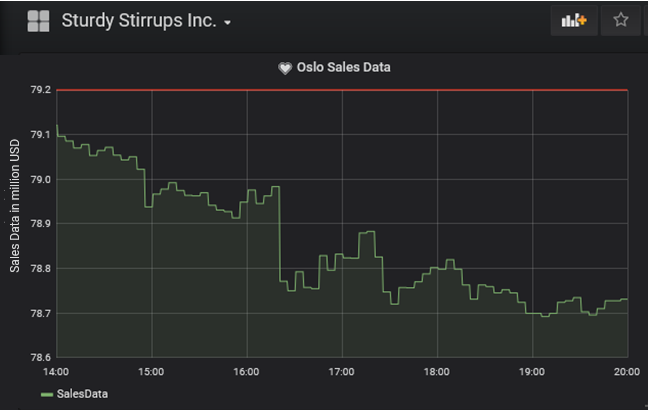
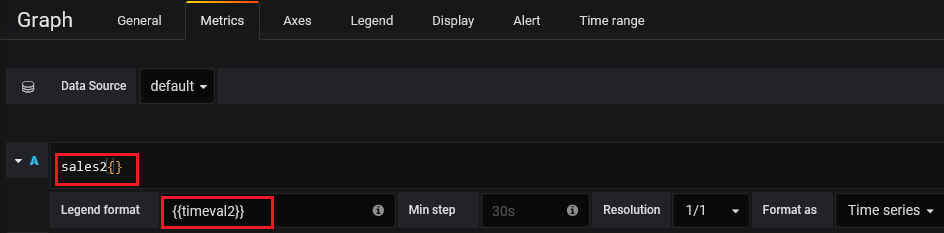


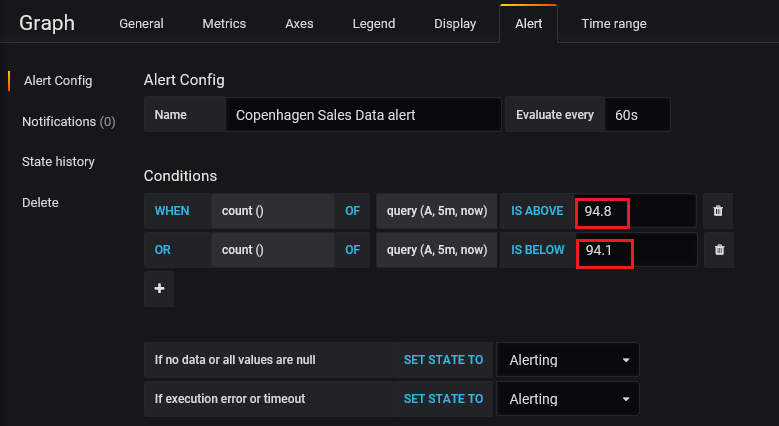
Figure 7 Graph displaying Local Office Oslo Sales Data

*Graph Panel for displaying Sales Data in Copenhagen*

Add the metrics as **sales2{}** and Legend format as **{{timeval2}}**

**

Setting up the Alerts: Alert config is set up on value of Copenhagen sales data going above 94.8 indicating higher number of sales and sales data going below 94.1 indicating the sales were lower

**

The Fig 8 shows the graph for Copenhagen Sales data in million dollars as a time series data over a period cycle of 6 hours and would trigger an alert when the value reaches above 94.8 or below 94.1

Sturdy Stirrups Inc. Local Office: Copenhagen Sales Data

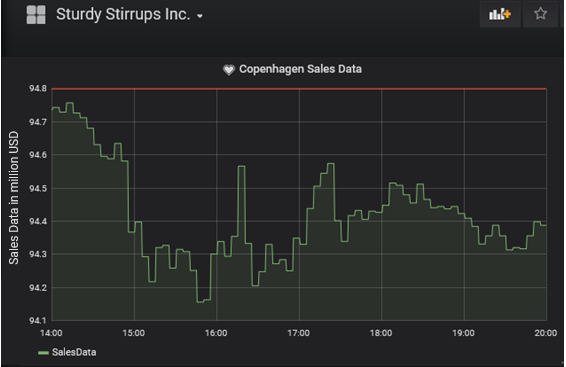
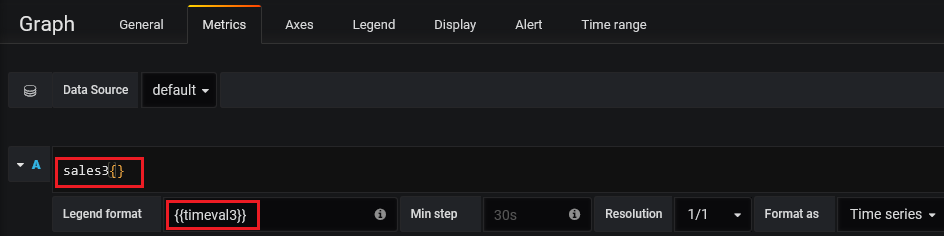
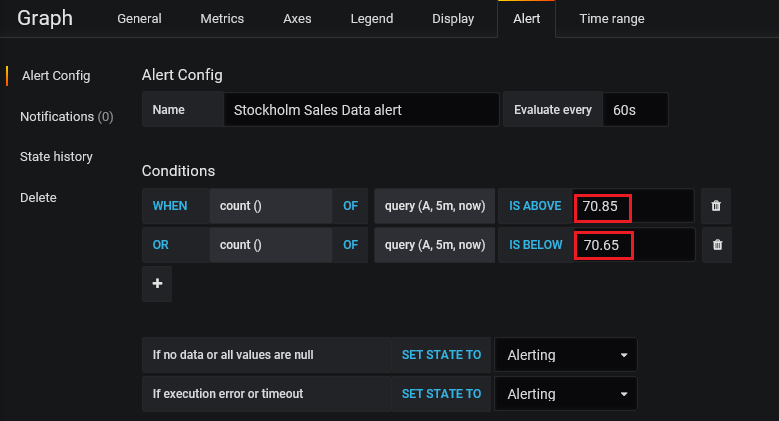


Figure 8 Graph displaying Local Office Copenhagen Sales Data

*Graph Panel for displaying Sales Data in Stockholm*

Add the metrics as **sales3{}** and Legend format as **{{timeval3}}**

**

****

The Fig 9 shows the graph for Stockholm Sales data in million dollars as a time series data over a period cycle of 6 hours and would trigger an alert when the value reaches above 70.85 or below 70.65

Sturdy Stirrups Inc. Local Office: Stockholm Sales Data

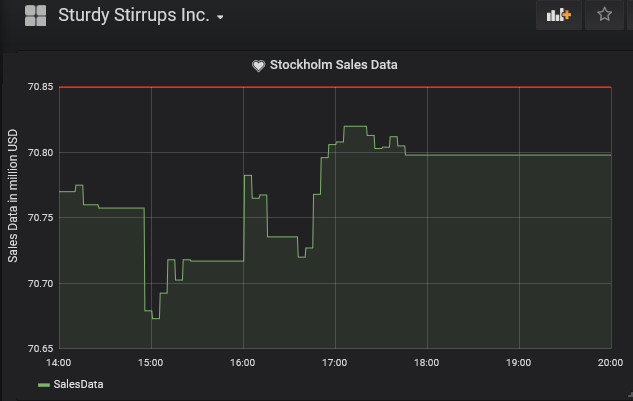
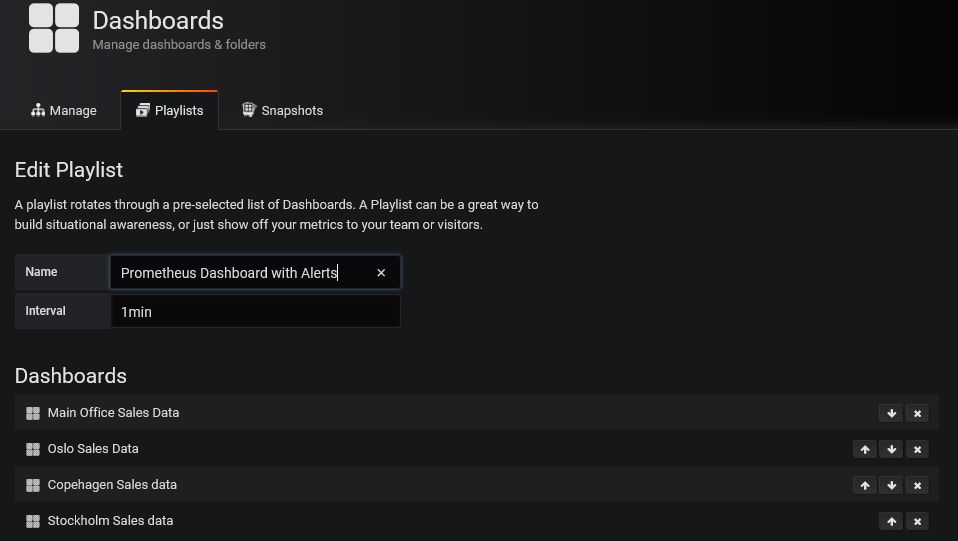


Figure 9 Graph displaying Local Office Stockholm Sales Data

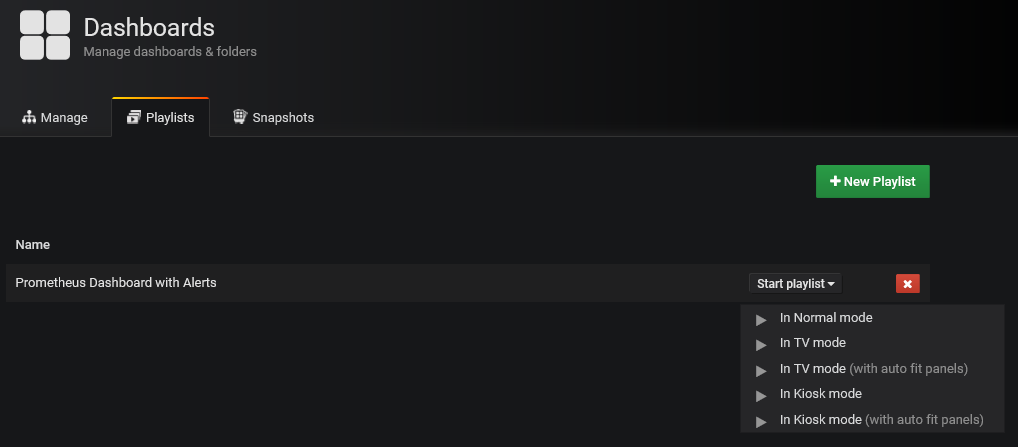
*Creating a PlayList:*

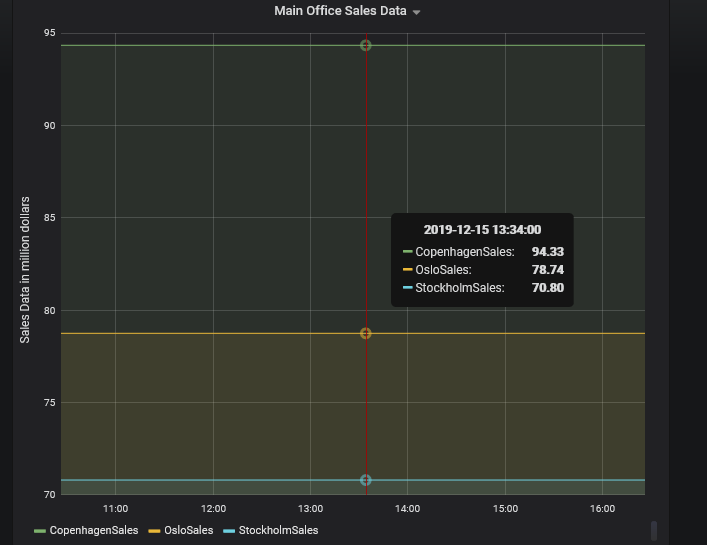
A playlist can be created which rotates through the list of selected dashboards. It is a nice way to display the metrics to the team members.[13]

In the Dashboard>Playlists option, a playlist can be created giving the interval time and selecting the list of dashboards to display on a rotating basis as in the below screenshot.[13]



The playlist can be started in various modes such as Normal, TV mode, Kiosk mode as below

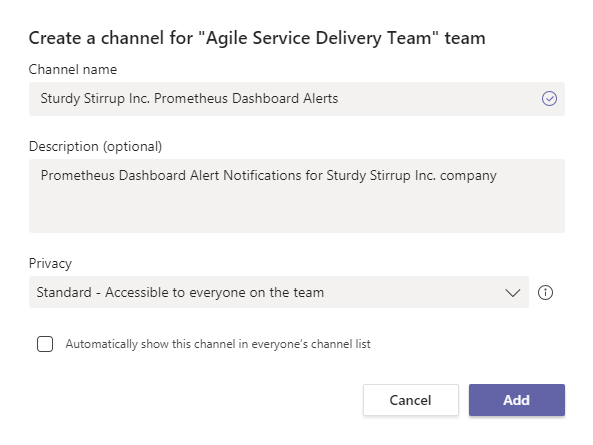




***Step 12 Sending Alert Notifications***

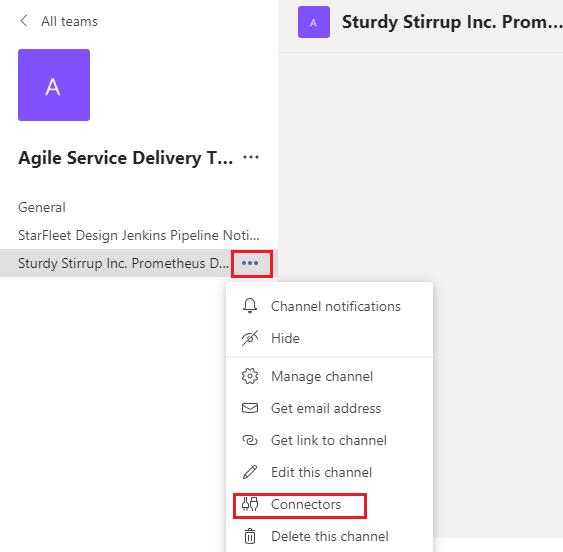
*Setting Up the Microsoft Teams Notification Channel[11][12]*

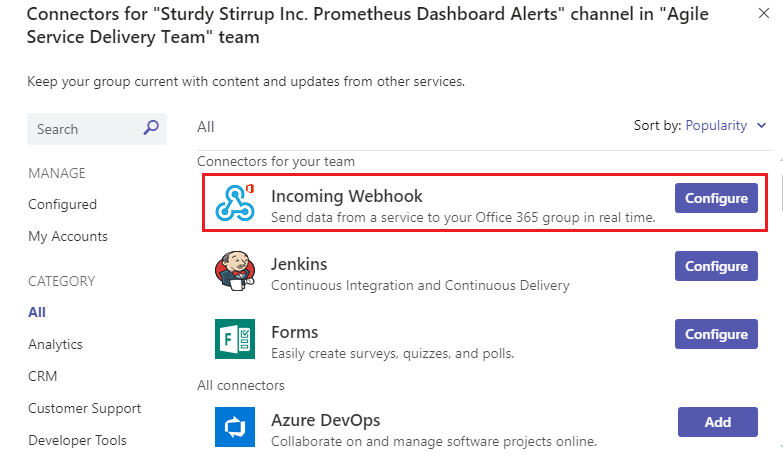
In Microsoft teams, create a Team named “Agile Service Delivery Team” and create a channel called “Sturdy Stirrup Inc. Prometheus Dashboard Alerts” as below

****

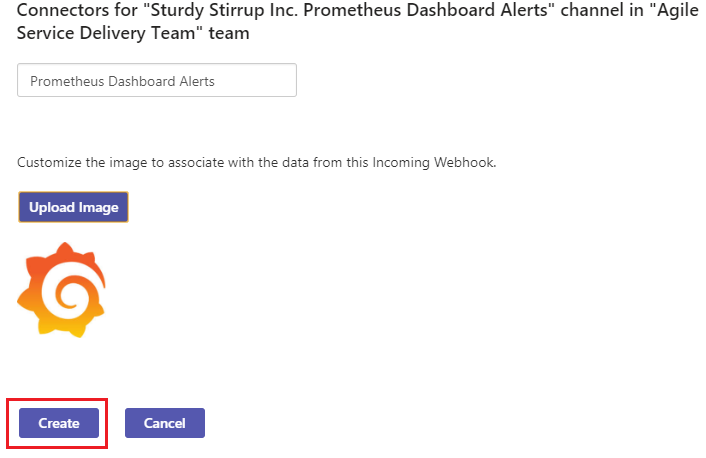
*Configuring Webhook*

Create connectors to the channel by selecting the option as below and create an Incoming Webhook.

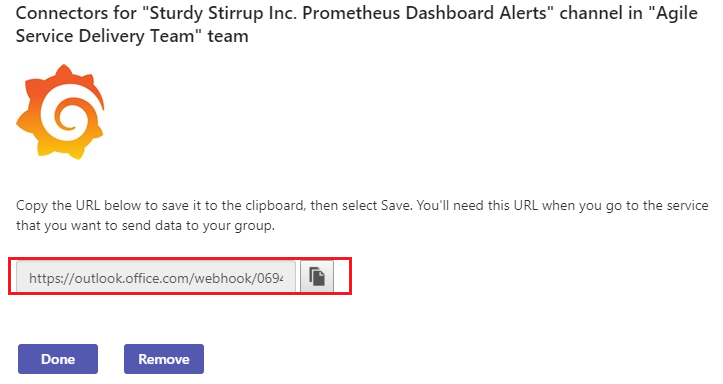
****

****

Give the name of the connector, upload an image and click on Create button.

****

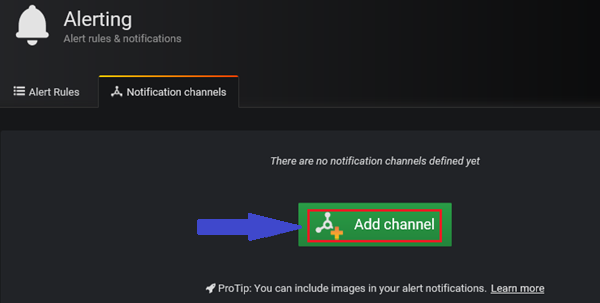
Once the connector is created, copy the URL below as highlighted in the below screenshot to clipboard.

****

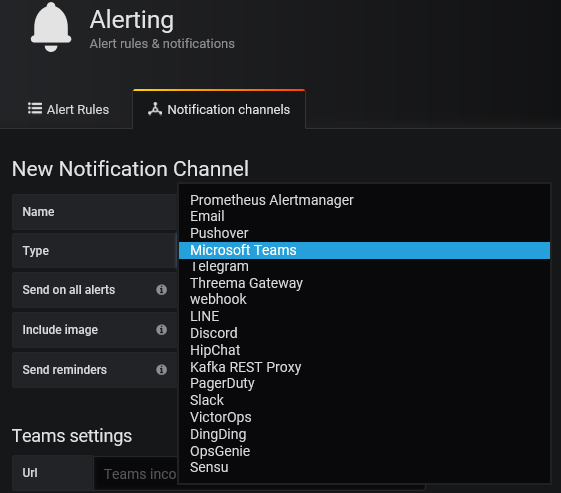
A unique URL is generated as highlighted in the above screenshot with the Incoming Webhook. This URL would be used in Grafana . The URL is saved in the clipboard.

*Configuring Notifications in Grafana*

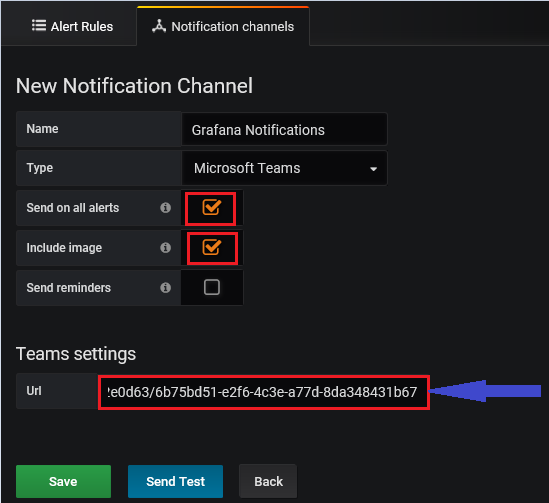
Under Alerting > Notifications, click on Add channel button as in the below screenshot.

******

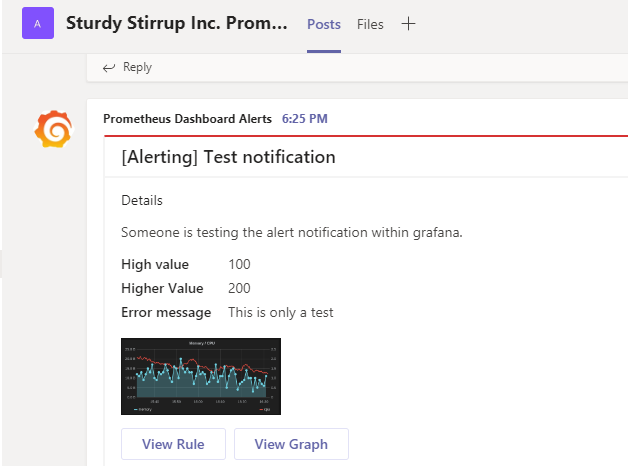
Select Microsoft Teams as the Notification channel

******

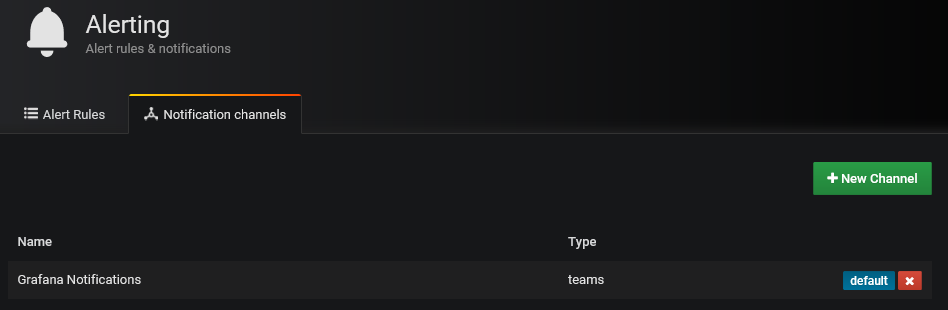
Provide the copied webhook URL in the below Team setting URL field as in the below screenshot.

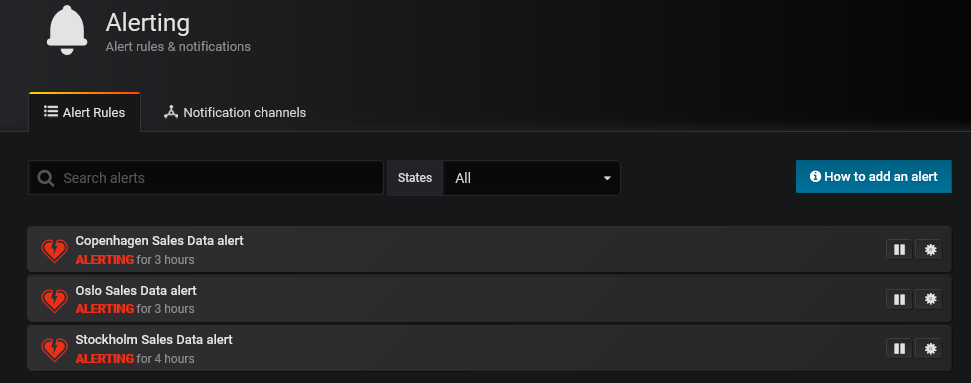
******

A Test Alert can be sent to test the above set up and the alert is received in the Teams channel as in the below screenshot.



After click on the Create button, a notification channel is created in Grafana as below. The Alert rules created for the different dashboards are displayed in the Alert rules section.



****

A Prometheus dashboard with alerting is implemented successfully which displays the sales data as a time series data in a graphical view for Main Office and Local Offices with alert notifications being sent.

**4. An evaluation**

The prototype solution implemented provides the ability to the sales managers or employees in the company to view the vast data effectively at a glance in a dashboard. This enables the managers to analyse the sales data of riding equipments and their profit margins.

The overview summary of the local offices in a dashboard for Main office managers helps them to take appropriate decisions to increase their sales in the offices where the profit margins are maximum. The offices where the sales are not doing well, corrective actions can be taken to improve the sales in those areas to gain maximum profits.

The alert notifications received via the Microsoft Teams channel, provides the notifications to the team members in the chat channel about the sales and profits going beyond the threshold, and this ensures corresponding actions would be taken to address those alert notifications.

The dashboard solution using Prometheus, Grafana and Docker tools is implemented successfully and can be extended with additional features. The solution can further be improved by sending the communication alerts also via other communication modes such as emails. A dashboard can also be created for each of the Local offices and Main office displaying the revenue data, sales and profit margins in a single snapshot which would further help in detailed data analytics and improved data readability.

**Appendix**

**Configuration Files:**

Custom metrics script file **load\_metrics.sh**

**#!/bin/sh**

**echo "# TYPE sales gauge"**

**echo "# HELP sales Company sales data across different local offices displayed in Main office"**

**echo "# TYPE sales1 gauge"**

**echo "# HELP sales1 Company sales data in Oslo office"**

**echo "# TYPE sales2 gauge"**

**echo "# HELP sales2 Company sales data in Copenhagen office"**

**echo "# TYPE sales3 gauge"**

**echo "# HELP sales3 Company sales data in Stockholm office"**

**EUR\_INR=$(curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "EUR-INR" |sed -e 's/.\*"EUR-INR">//'|sed -e 's/<.\*//')**

**GBP\_INR=$(curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "GBP-INR" |sed -e 's/.\*"GBP-INR">//'|sed -e 's/<.\*//')**

**USD\_INR=$(curl -s GET https://transferwise.com/gb/currency-converter/nok-to-inr-rate | grep "USD-INR" |sed -e 's/.\*"USD-INR">//'|sed -e 's/<.\*//')**

**echo "sales{timeval=\"OsloSales\"} $EUR\_INR"**

**echo "sales{timeval=\"CopenhagenSales\"} $GBP\_INR"**

**echo "sales{timeval=\"StockholmSales\"} $USD\_INR"**

**echo "sales1{timeval1=\"SalesData\"} $EUR\_INR"**

**echo "sales2{timeval2=\"SalesData\"} $GBP\_INR"**

**echo "sales3{timeval3=\"SalesData\"} $USD\_INR"**

**Set of shell commands to be executed:**

**apt-get update**

**apt-get install –y curl**

**apt install docker**

**docker swarm init**

**git clone https://github.com/stefanprodan/swarmprom.git**

**cd /root/swarmprom**

**docker stack deploy –c docker-compose.yml mon**

**docker service ls**

**mkdir /var/lib/exporter\_data**

**docker run --name=myexporter -d --network=mon\_net -v /var/lib/exporter\_data:/srv abiosoft/caddy**

**docker config create prometheus-docker-entrypoint.sh.v1 docker-entrypoint.sh #Create new version of prometheus config file**

**docker service update –config-add source=prometheus-docker-entrypoint.sh.v1, target=/etc/Prometheus/docker-entrypoint.sh, mode=0755 mon\_prometheues**

**docker service update –-config-add source=prometheus.yml.v1, target=/etc/prometheus/prometheus.yml, mode=0644 mon\_prometheus**

**#The above command replaces the configuration file with the latest version**

**docker config create prometheus.yml.v1 prometheus.yml**

**docker service update --config-add source=prometheus.yml.v1,target=/etc/prometheus/prometheus.yml,mode=0644 mon\_prometheus**

**docker logs myexporter**

**export EDITOR=nano**

**crontab -e**

**Reference to sources:**

[1] Prometheus dashboard with alerting <https://oslomet.instructure.com/courses/16478/pages/prometheus-dashboard-with-alerting?module_item_id=119333>

[2] <https://opensource.com/resources/what-docker>

[3] <https://github.com/prometheus/prometheus>

[4] <https://medium.com/htc-research-engineering-blog/build-a-monitoring-dashboard-by-prometheus-grafana-741a7d949ec2>

[5**]** Fn Project – Display Fn runtime metrics using Prometheus and Grafana <https://fnproject.io/tutorials/grafana/>

[6] <https://prometheus.io/docs/prometheus/latest/configuration/configuration/>

[7] <https://grafana.com/docs/grafana/v4.3/>

[8] Prometheus Overview <https://prometheus.io/docs/introduction/overview/>

[9] Prometheus tutorial<https://git.cs.hioa.no/kyrre.begnum/swarmprom-setup>

[10] Setting up a simple prometheus export service <https://oslomet.instructure.com/courses/16478/pages/setting-up-a-simple-prometheus-export-service>

[11] Notifications set up for Microsoft Teams <https://jorgedelacruz.uk/2019/09/27/grafana-using-microsoft-teams-for-our-notifications-when-established-thresholds-are-exceeded/>

[12] <https://dzone.com/articles/grafana-alerting-and-email-notifications>

[13] <https://grafana.com/docs/grafana/v6.2/reference/playlist/>

[14] <https://prometheus.io/docs/prometheus/latest/configuration/configuration/>

[15] Web Server icon [**https://www.google.co.in/search?q=web+server+icon&sxsrf=ACYBGNQo6N1xKZKx6-eIaXf5K-AHrAwgyQ:1576184086942&tbm=isch&source=iu&ictx=1&fir=FgwmkQi86CLRAM%253A%252CxIYO1exJt9CQyM%252C\_&vet=1&usg=AI4\_-kQ2FONzLTQBHAp9oXzby6pn4u\_rqw&sa=X&ved=2ahUKEwi99-ik\_7DmAhXlpIsKHRH2BjoQ\_h0wDHoECA0QBg#**](https://www.google.co.in/search?q=web+server+icon&sxsrf=ACYBGNQo6N1xKZKx6-eIaXf5K-AHrAwgyQ:1576184086942&tbm=isch&source=iu&ictx=1&fir=FgwmkQi86CLRAM%253A%252CxIYO1exJt9CQyM%252C_&vet=1&usg=AI4_-kQ2FONzLTQBHAp9oXzby6pn4u_rqw&sa=X&ved=2ahUKEwi99-ik_7DmAhXlpIsKHRH2BjoQ_h0wDHoECA0QBg)

[16] Dashboard icon [https://www.google.co.in/search?biw=1242&bih=597&tbm=isch&sxsrf=ACYBGNRTA2D8F0JL0OFLBFJU0zSQ6D1BeA%3A1576184406564&sa=1&ei=VqryXfmNIu\_IrgTMo5zYCQ&q=dashboard+icon&oq=dashboard+&gs\_l=img.3.0.0i67l2j0l6j0i67j0.61395.65517..66713...0.0..0.189.1108.5j5......0....1..gws-wiz-img.......35i39.rzckNA3hxBA#](https://www.google.co.in/search?biw=1242&bih=597&tbm=isch&sxsrf=ACYBGNRTA2D8F0JL0OFLBFJU0zSQ6D1BeA%3A1576184406564&sa=1&ei=VqryXfmNIu_IrgTMo5zYCQ&q=dashboard+icon&oq=dashboard+&gs_l=img.3.0.0i67l2j0l6j0i67j0.61395.65517..66713...0.0..0.189.1108.5j5......0....1..gws-wiz-img.......35i39.rzckNA3hxBA)

[17] File icon <https://www.vectorstock.com/royalty-free-vector/file-icon-paper-symbol-vector-22537726>

[18] Prometheus icon <https://www.google.co.in/search?tbm=isch&sxsrf=ACYBGNQWHFdyLA7wYvzUrQ5RaB9jJo8vPw%3A1576429668901&sa=1&ei=ZGj2XdjTNvSCk74PisKTiAo&q=prometheus+icon&oq=prometheus+icon&gs_l=img.3..35i39j0l2j0i30j0i5i30l2j0i8i30l3j0i24.62128.64246..64406...0.0..1.499.3311.0j2j7j1j2......0....1..gws-wiz-img.......0i67.U_eatmcgg3Q&ved=0ahUKEwjYlbaTkrjmAhV0wcQBHQrhBKEQ4dUDCAc&uact=5#>

[19] Grafana icon

<https://www.google.co.in/search?tbm=isch&sxsrf=ACYBGNRbK5tJz8V7wCmRtUAmxdw5P9QKpQ%3A1576429788144&sa=1&ei=3Gj2XeSlCLHQmwXl9IKIBQ&q=grafana+icon+blue&oq=grafana+icon+blue&gs_l=img.3...56087.56764..56913...0.0..0.177.607.2j3......0....1..gws-wiz-img.......0j0i30j0i5i30j0i8i30.pObKcu3K3Bw&ved=0ahUKEwjkg6TMkrjmAhUx6KYKHWW6AFEQ4dUDCAc&uact=5#>

[20] Grafana Graph dashboard icon

<https://www.google.co.in/search?tbm=isch&sxsrf=ACYBGNTnmjBvAZ3Q3rmYr1_dFE2S9PQnKA%3A1576429846236&sa=1&ei=Fmn2XfKCDomg6ATh8JzwAg&q=grafana+graph&oq=grafana+graph&gs_l=img.3..0l5j0i8i30l5.51036.53864..54063...1.0..0.104.1145.14j1......0....1..gws-wiz-img.......35i39j0i30j0i5i30j0i24.1cpZ8SMTOdc&ved=0ahUKEwjy5f3nkrjmAhUJEJoKHWE4By4Q4dUDCAc&uact=5#>