**INSTALL PROMETHEUS ON VM/EC2/GOOGLECLOUD INSTANCE**

**Prerequisites**

* VM/EC2/GC instance with ubuntu 18.04 OS
* The instances should have security groups/firewall rules open for 9090 and 9100
* Need root access on the VM. All steps which are outlined below should be run as root

**Prometheus Installation:**

Prometheus can be installed in different ways as pods containers and a plain installation via the binaries which we are going to see in the below steps outlined

* ***sudo su - # switch to root***
* ***useradd --no-create-home prometheus # create a user called prometheus on server***
* ***mkdir /etc/prometheus # create a config directory for prometheus***
* ***mkdir /var/lib/prometheus # create a storage path for prometheus***
* ***wget https://github.com/prometheus/prometheus/releases/download/v2.38.0/prometheus-2.38.0.linux-amd64.tar.gz # downloading prometheus from official website***
* ***tar xvfz prometheus-2.38.0.linux-amd64.tar.gz # untar the release bundle***
* ***cd prometheus-2.38.0.linux-amd64/ # navigate to prometheus release bundle***
* ***cp -p prometheus /usr/local/bin/ # Install prometheus binary on server. This will help us to run promtheus binary from anywhere on server. We use this to start prometheus on server***
* ***cp -p promtool /usr/local/bin/ # Install promtool binary on server. This binary is used to validate rules in prometheus config file .Not a manadatory binary***
* ***cp -pR consoles/ /etc/prometheus/ ## This directory hold html files responsible for prometheus UI***
* ***cp -pR console\_libraries/ /etc/prometheus/ ## this directory holds the library files of Prometheus***

***PROMETHEUS CONFIG YAML***

For testing purpose let’s add some content into prometheus.yml. This is the config file prometheus server reads and understands what to monitor using prometheus

* ***vi /etc/prometheus/prometheus.yml*** ***## we configure the same path during the startup of promtheus as config file***



Detailed explanation of each field is given below

## The global configuration specifies parameters that are valid in all other configuration contexts. They also serve as defaults for other configuration sections.

global:

## How frequently to scrape targets by default.Targets are the services which we wish to monitor

scrape\_interval: 15s

## How long until a scrape request times out.If the target doesnt respond back with data within the below interval.Prometheus marks the target as unhealthy until next scrape\_interval

scrape\_timeout: 15s

## These labels will be used when communicating with external systems like alerting..Its completely optional

external\_labels:

monitor: 'prometheus'

## A list of scrape configurations. This is typically a section where we ask prometheus server to monitor particular URL's or metrics endpoints.In the general case, one scrape configuration specifies a single job. In advanced configurations, this may change. Targets can be static or dynamic.In this example we are asking prometheus to montior itself to test the installation

scrape\_configs:

## The job name assigned to scraped metrics by default.It has to be unique for different scrape configs.This can be seen in prometheus graph UI when we exceute a query by selecting a required metric

- job\_name: 'prometheus'

## key word which designates static configs. There can also be dynamic\_configs using servicediscovery mechanisms depending on what targets we wish to monitor

static\_configs:

## target to monitor.Here in this example, its listening to prometheus metrics end point hence it can scrape prometheus metrics. The default metrics path in prometheus is /metircs and to view the targets we can launch prometheus appending prometheus URL with a path /targets

## We can also change the global configs which are mentioned above under each specific job

- targets: ['localhost:9090']

* ***vi /etc/systemd/system/prometheus.service ## this will help us create prometheus as a service so that we can enable it at startup***

******

* ***## change permissions of prometheus related files to user prometheus***
  + ***chown -R prometheus:prometheus /etc/prometheus/***
  + ***chown -R prometheus:prometheus /etc/prometheus/console\_libraries***
  + ***chown -R prometheus:prometheus /var/lib/prometheus***
  + ***chown prometheus:prometheus /usr/local/bin/prometheus***
  + ***chown prometheus:prometheus /usr/local/bin/promtool***
* ***## Enable prometheus service on server startup. This will bring up prometheus along with reboot***
  + ***systemctl daemon-reload***
  + ***systemctl enable prometheus***
  + ***service prometheus start***
* ***Prometheus server will now be accessible on port 9090 on the installed server***
  + ***http:// {external IP}/{publicDNS}/{HOSTIP}:9090***
* ***Other links which you can explore are***

***## Prometheus metrics***

* + ***http:// {external IP}/{publicDNS}/{HOSTIP}:9090/metrics***

***## Targets - Prometheus is only target we have so far and it should be healthy at this point***

* + ***http:// {external IP}/{publicDNS}/{HOSTIP}:9090/targets***

***Exporters:***

There are many exporters and integrations available for Prometheus through which it can gather metrics of various resources.

In the below instructions we are now going to install Node exporter which will help collect all the node related metrics

The ***node\_exporter*** is designed to monitor the host system.

***Installation of Node\_exporter***

***Note***: For demo purpose we are installing node\_exporter on same server where promethus is installed. It can be installed on any server which you wish to monitor and can be mention in the config file to scrape the metrics

* ***useradd --no-create-home node\_exporter # create a user called node\_exporter on server***
* ***wget https://github.com/prometheus/node\_exporter/releases/download/v1.3.1/node\_exporter-1.3.1.linux-amd64.tar.gz # downloading node\_exporter from official website***
* ***tar xvzf node\_exporter-1.3.1.linux-amd64.tar.gz # untar the release bundle***
* ***cd node\_exporter-1.3.1.linux-amd64/ # navigate to node\_exporter release bundle***
* ***cp -p node\_exporter /usr/local/bin/ # Install prometheus binary on server. This will help us to run node\_exporter binary from anywhere on server. We use this to start node\_exporter on server***
* ***vi /etc/systemd/system/node-exporter.service ## this will help us create node\_exporter as a service so that we can enable it at startup***

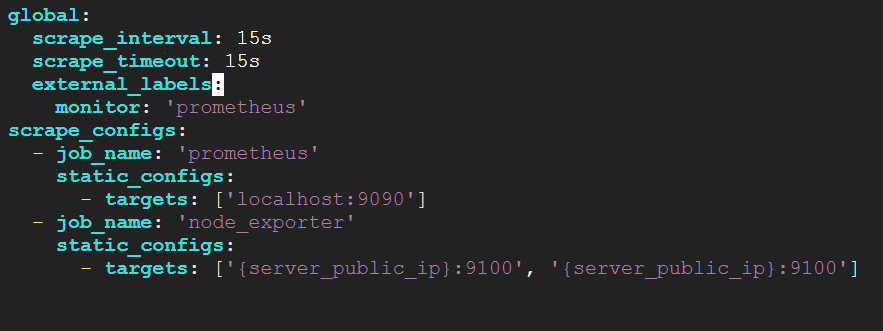
******

* ***## Enable node\_exporter service on server startup. This will bring up node\_exporter along with reboot***
  + ***systemctl daemon-reload***
  + ***systemctl enable node-exporter***
  + ***systemctl start node-exporter (starting the node\_exporter)***
* ***At this point you should be able to access metrics page of node\_exporter. These are the same metrics Prometheus server will scrape after we do some changes in Prometheus.yml***
  + ***http:// {external IP}/{publicDNS}/{HOSTIP}:9100/metrics***
* ***Add the below content under scrape configs in Prometheus.yml..We are defining prometheus configuration to read the metrics collected by node\_exporter..By default for any job the metrics endpoint is /metrics so we don’t explicitly mention it..In case there is different endpoint we override it in scrape\_configs using this key word metrics\_path: <path>***

***vi /etc/prometheus/prometheus.yml***

******

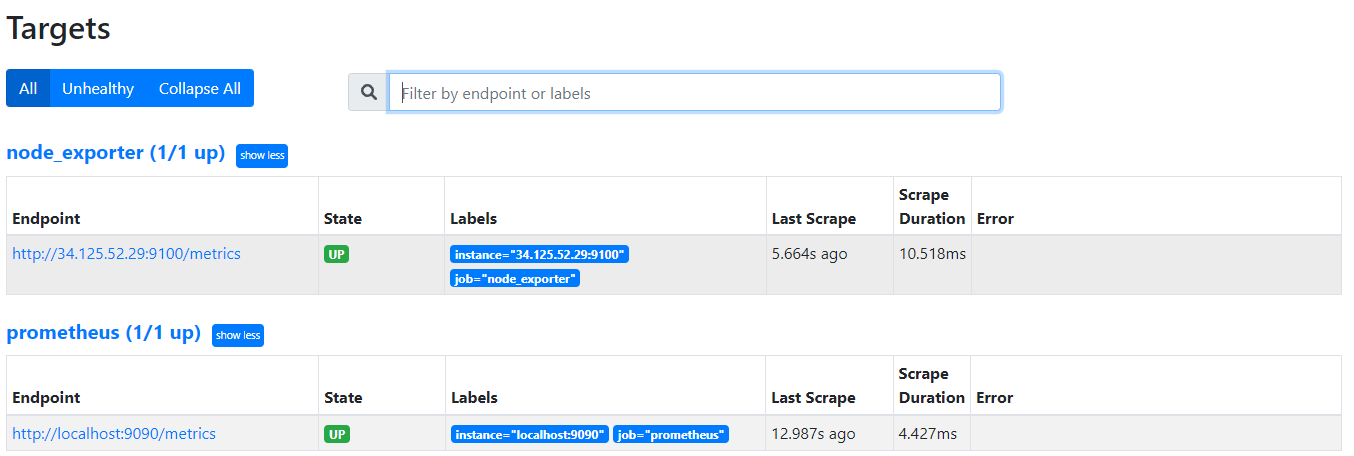
* ***At this point your promethus.yml should look like below>Please make sure you are updating the node public IP where node exporter is installed and you wihs to monitor.As you can see we can give multiple targets.i.e multiple nodes in this case with comma separated values in targets***



* ***We restart Prometheus server as there is new configuration given so that server picks the config***

***systemctl restart Prometheus***

* ***Once the Prometheus server is restarted you would see the targets endpoint showing up two targets configured as below***
  + ***http:// {external IP}/{publicDNS}/{HOSTIP}:9090/targets***



***Grafana Installation:***

To visualize the node exported collected metrics more efficiently lets install Grafana on the same node and use a community Grafana dashboard to visualize the node metrics

Same as Prometheus we can install Grafana in different ways in pods container etc.. In this case we are doing a standalone installation

**Note**: Im using the same server to install grafana

* + ***sudo apt-get install -y apt-transport-https***
  + ***sudo apt-get install -y software-properties-common wget***
  + ***wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -***
  + ***echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list***
  + ***sudo apt-get update***
  + ***sudo apt-get install grafana***
  + ***systemctl daemon-reload***
  + ***systemctl start grafana-server***
  + ***systemctl status grafana-server***
  + ***systemctl enable grafana-server.service***

***Once all the steps are complete you should be able to access Grafana on below URL***

* + ***http:// {external IP}/{publicDNS}/{HOSTIP}:3000***

***username/password: admin/admin (we can change this later if needed)***

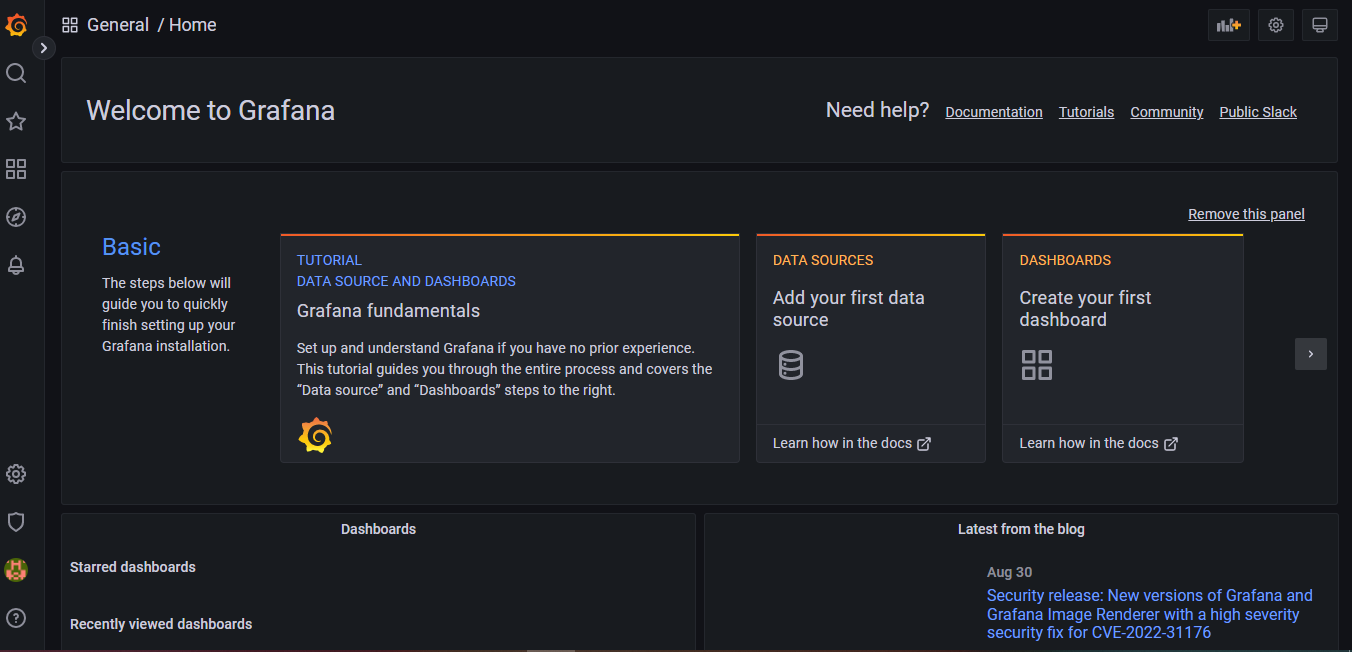
***Configuring Datasource in Grafana:***

Grafana is just a avisualization tools which gets the data from various sources and represents that data in a good visualization format.

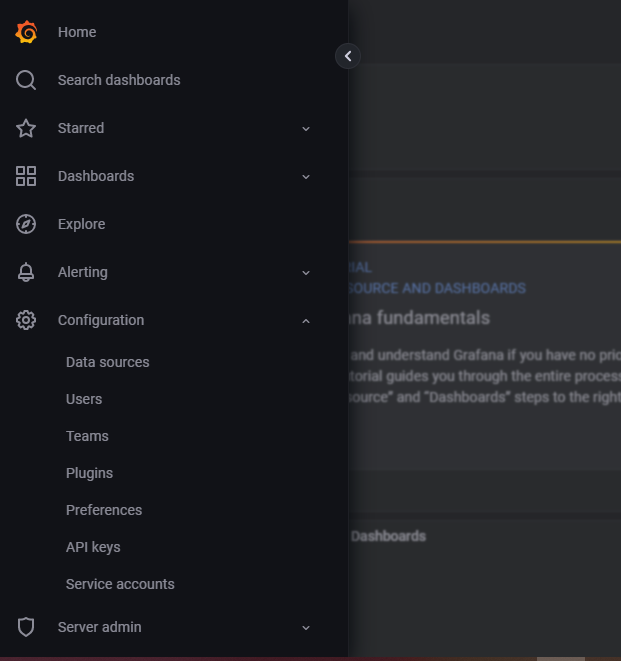
We are now going to configure the datasource as Prometheus which we just installed.So the data whichever is in Prometheus with the help of promql queries in Grafana dashboard will be displayed in the Grafana UI

Let’s configure the datasource

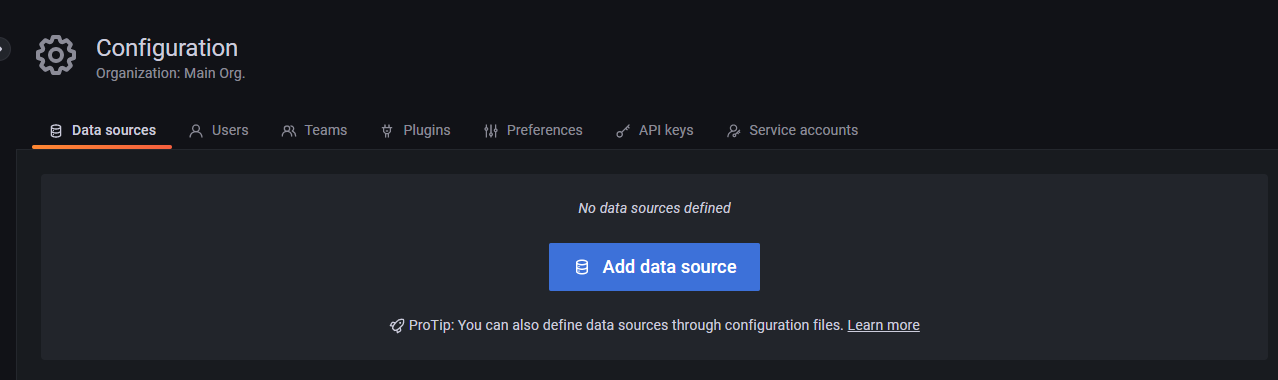
* As soon as we login using the credentials below would be landing page

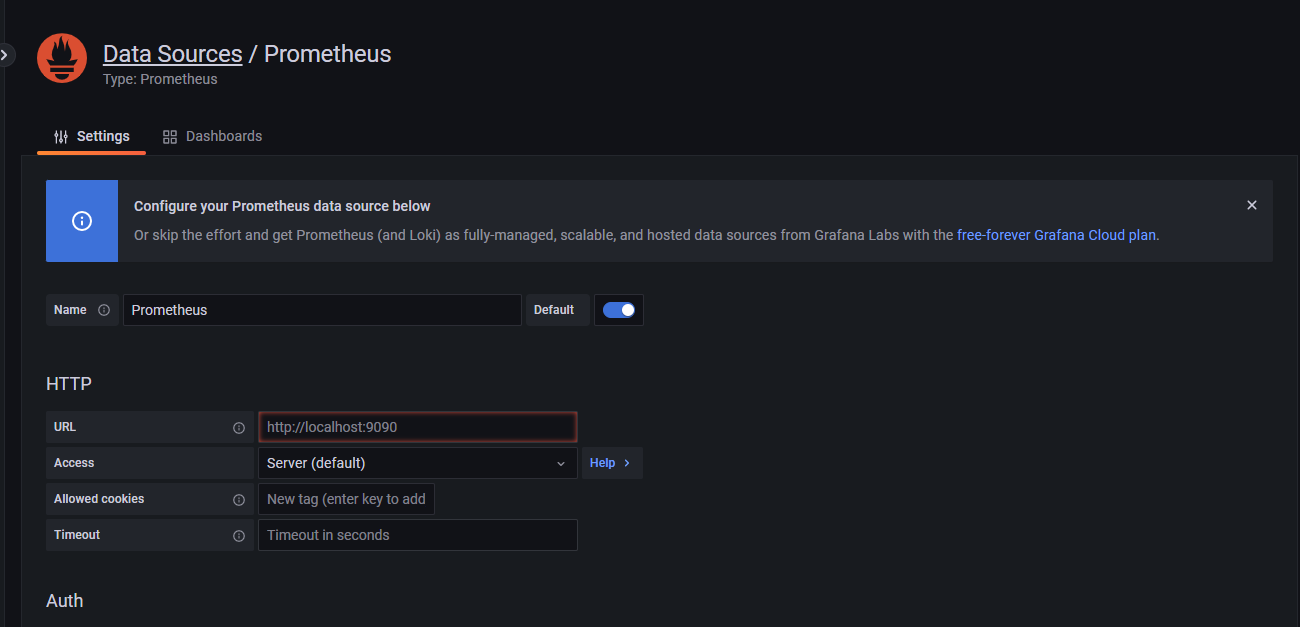


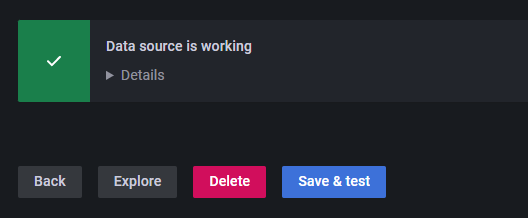
* On the left Navigation there are multiple options select configurations -> Datasources



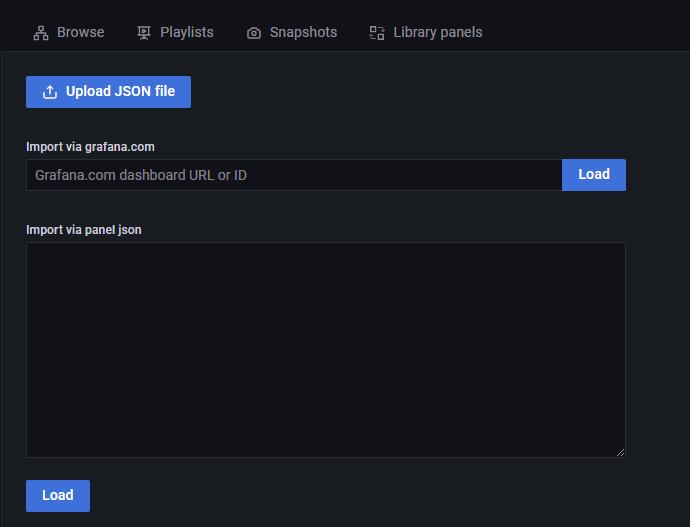
* Click on Add the data source



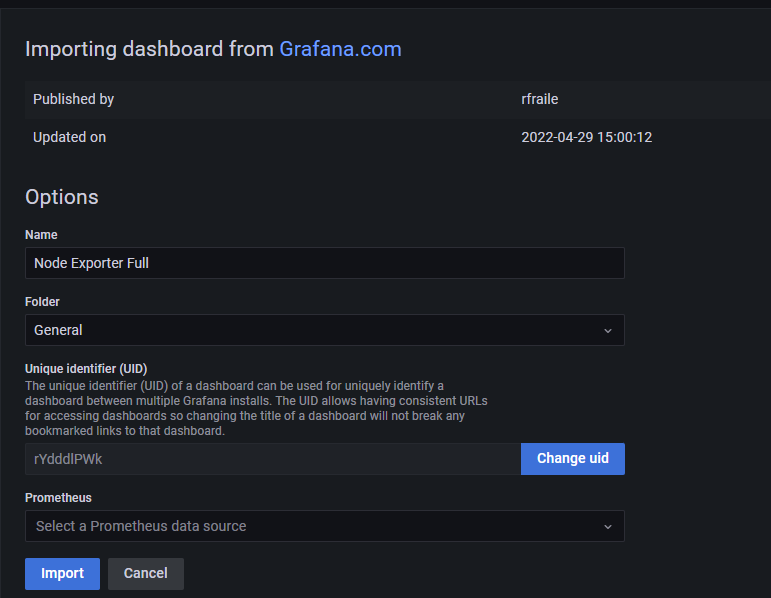
* There are many datasources Grafana upports by default lets click on Prometheus
* IT takes you to the Prometheus details page where we provide the URL of Prometheus
* 
* Under URL update it as <http://localhost:9090> ( our Prometheus endpoint)
* Click on Save and test and it should show as below which means Grafana can now connect to prometheus and query it



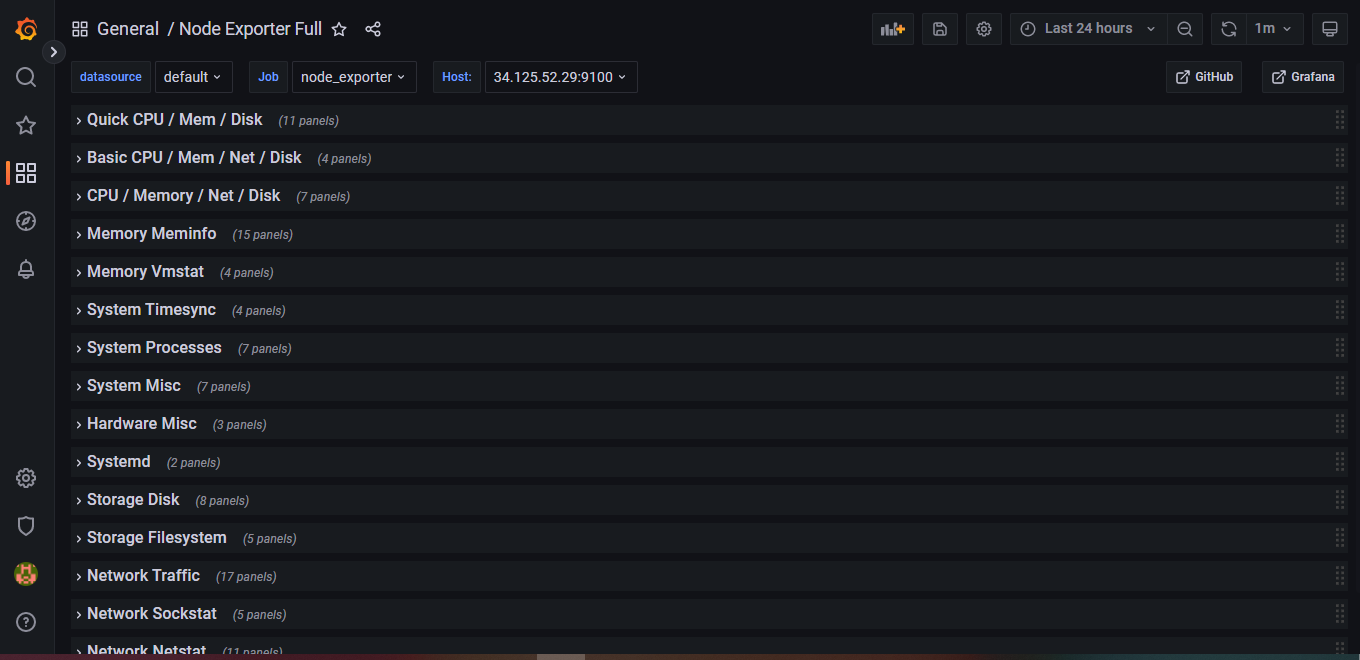
* Let’s now copy the community Grafana dashboard for node\_exporter and see the data which Prometheus is collecting
* On the left Nav of Grafana select Dashboards and Import option



* Type the ID 1860 in the first box and click on load



* Make sure you are selecting the data source at the bottom. Its Prometheus. You can select from dropdown and click on Import
* That’s it you will be able to see all the node level metrics which node exporter is collecting in great visualization



* If you expand the panels, you would be able to see the metrics related to each block

