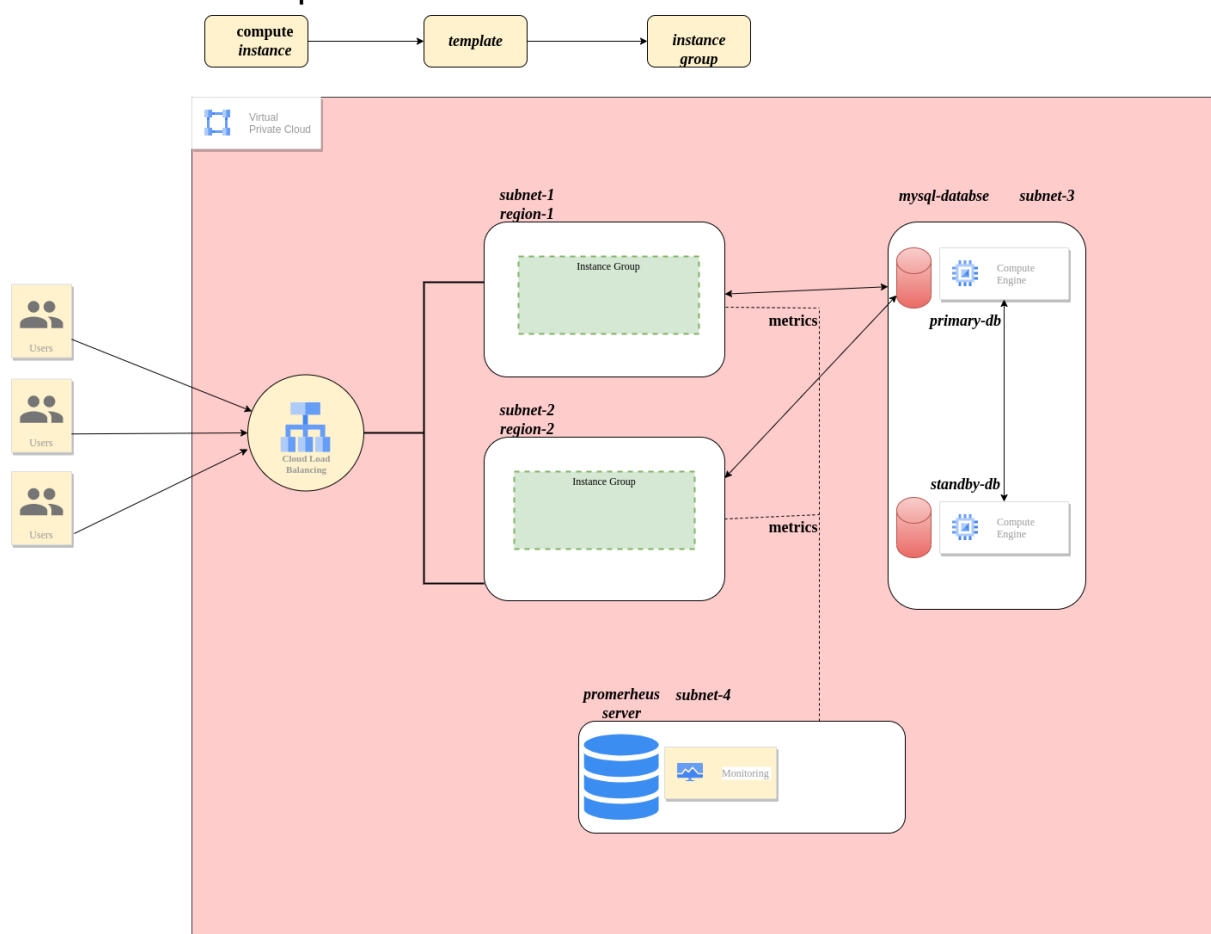




OBJECTIVE: Hosting and monitoring of ESPOCRM website

Architecture of espocrm website:



EspoCRM:

- Its a web application with a frontend designed as a single page application and REST API backend written in PHP
- So we need a webserver (apache or nginx), and all PHP dependencies but if you use NGINX as webserver we need to install PHP-FPM to serve php code

Nginx:

NGINX is open source software used as a web server, reverse proxying, caching, load balancing, media streaming, and more.

Hosting of ESPOCRM in a Linux server:

Nginx installation: to install nginx run below commands

```
sudo apt-get update
sudo apt-get install nginx -y
```

INSTALL PHP7.4: to install php Run below commands

```
sudo apt-get install PHP7.4
```

PHP dependencies: to install php dependencies Run below commands

```
sudo apt-get update
sudo apt-get install php-mysql php-json php-gd php-zip php-imap php-mbstring
php-curl php-exif php-ldap
sudo phpenmod imap mbstring
sudo service nginx restart
```

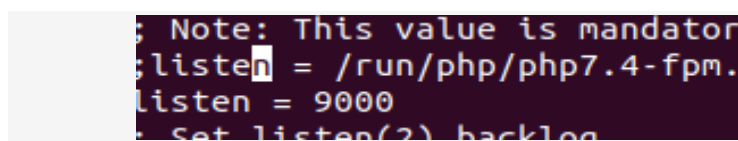
Actually nginx doesn't serve the php code. To serve php code in nginx we need other package called php-fpm

```
Sudo apt-get install php7.4-fpm
```

After installing the php7.4-fpm by default it listen to the socket but in nginx configuration file we mentioned that it is listening to the 9000 port so change the php7.4-fpm configuration file and make it listen on port 9000
To do this edit the /etc/php/7.4/fpm/pool.d/[www.conf](#) file comment the line fpm socket listening and add listen port 9000

```
Vim /etc/php/7.4/fpm/pool.d/www.conf
```

Check the below screenshot and to add the listen port



```
; Note: This value is mandatory
listen = /run/php/php7.4-fpm.
listen = 9000
; Set listen(2) backlog
```

To effect the changes which made earlier in php7.4-fpm restart the php7.4-fpm service

```
Sudo service php7.4-fpm restart
```

In the official espocrm website given the recommended php.ini settings . so set the values as per given in /etc/php/7.4/fpm/php.ini file

```
max_execution_time = 180
```

```
max_input_time = 180
```

```
memory_limit = 256M
```

```
post_max_size = 50M
```

```
upload_max_filesize = 50M
```

Download the espocrm package

```
Wget https://www.espocrm.com/downloads/EspoCRM-7.2.2.zip
```

Unzip the downloaded package and rename EspoCRM-7.2.2 to espocrm then move to /var/www/html/espocrm.

```
unzip EspoCRM-7.2.2.zip
```

```
mv EspoCRM-7.2.2 Espocrm
```

```
mv Espocrm /var/www/html/
```

To host this website add a nginx configuration file in **/etc/nginx/site-available** location

```
vim /etc/nginx/site-available/espocrm.conf
```

Add the below content in the espocrm.conf file To host this website add a nginx configuration file in /etc/nginx/site-available location

```
vim /etc/nginx/site-available/espocrm.conf
```

Add the below content in the espocrm.conf file

```
server {  
    listen 80 default_server;  
    listen [::]:80 default_server;  
  
    server_name localhost; # domain name
```

```

charset utf-8;
index index.html index.php;

client_max_body_size 50M;

keepalive_timeout 300;
types_hash_max_size 2048;

server_tokens off;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;

gzip on;
gzip_types text/plain text/css text/javascript application/javascript
application/json;
gzip_min_length 1000;
gzip_comp_level 9;

root /var/www/html/espocrm/public; # path to public dir

location /client {
    root /var/www/html/espocrm; # path to espocrm root dir
    autoindex off;

    location ~* ^\.+.(js|css|png|jpg|jpeg|gif|ico|tpl)$ {
        access_log off;
        expires max;
    }
}

location = /favicon.ico { access_log off; log_not_found off; }
location = /robots.txt { access_log off; log_not_found off; }

location ~ /\.php$ {
    fastcgi_pass localhost:9000;
    include fastcgi_params;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param QUERY_STRING $query_string;
}

location /api/v1/ {
    if (!-e $request_filename){
        rewrite ^/api/v1/(.*)$ /api/v1/index.php last; break;
    }
}

```

```

location /portal/ {
    try_files $uri $uri/ /portal/index.php?$query_string;
}

location /api/v1/portal-access {
    if (!-e $request_filename){
        rewrite ^/api/v1/(.*)$ /api/v1/portal-access/index.php last; break;
    }
}

location ~ /(\\.htaccess|\\.web.config|\\.git) {
    deny all;
}
}

```

** You need to change Document root to the absolute path of your EspoCRM instance.if you are using other than /var/www/html/espocrm path
Link and activate the file to /etc/nginx/site-enabled:

```
Sudo ln -s /etc/nginx/site-available/espocrm.conf /etc/nginx/site-enable/espocrm.conf
```

Run below command in a terminal to check nginx configuration file are as per required syntax :

```
sudo nginx -t
```

restart nginx server so that the new configuration file comes in to effect :

```
sudo service nginx restart
```

INSTALL MYSQL DATABASE: To install mysql Run below commands

```
sudo apt-get install mysql-server
```

Install the mysql securely to do this run the following command

```
Sudo mysql_secure_installation
```

Remove the test databases and set the root password

Creating EspoCRM Database :

After installing all the packages, now we need to create a Database,run the command to create the EspoCRM database.

Run the below command to logon to the database server. You will get a prompt for a root password. So, type the root password you created above:

```
sudo mysql -u root -p
```

1) To create Database run the the below command:

```
CREATE DATABASE espocrm;
```

2) Then create a database user as vegayser, with new password:

```
CREATE USER 'espo-user'@'localhost' IDENTIFIED BY 'password';
```

3) Grant the user Privileges:

```
GRANT ALL ON espocrm.* TO 'espo-user'@'localhost' IDENTIFIED BY 'password';
```

4) Save the changes and exit

```
FLUSH PRIVILEGES;  
exit ;
```

Go to the browser and type the localhost or <server public ip>

If you prompt to change the document root path ownership and permissions, change the ownership and permissions for document root directory

To set the ownership permissions, execute below commands in the terminal:

```
sudo chown -R www-data:www-data /var/www/espocrm
```

Change access permissions for the directory.

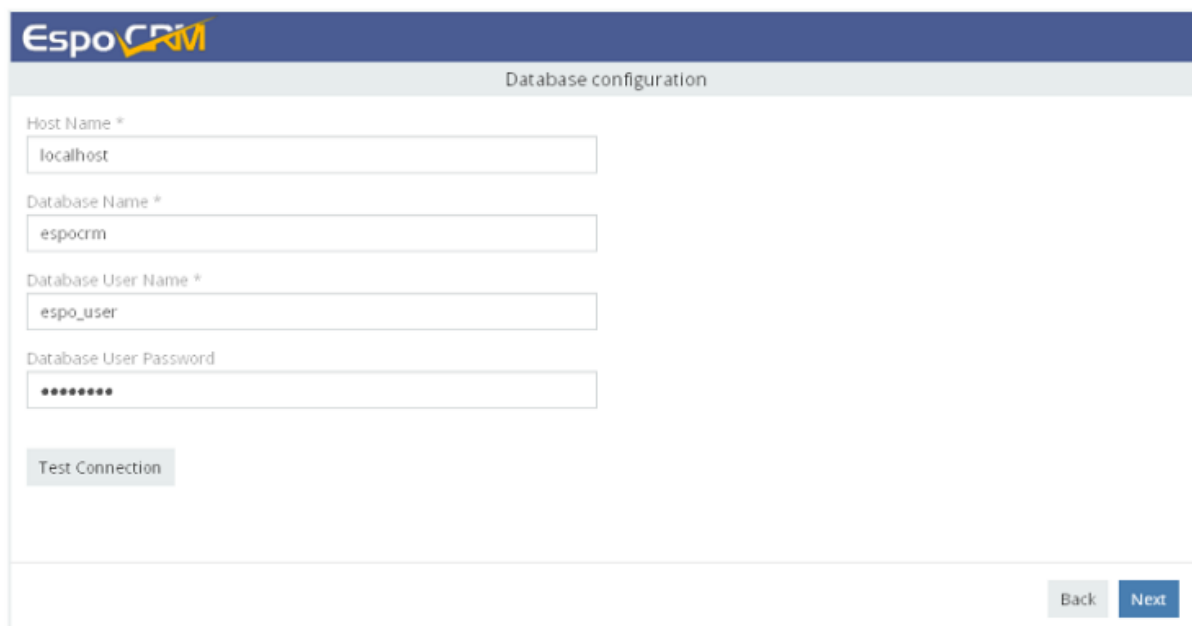
```
sudo chmod -R 755 /var/www/espocrm
```

Then open the browser and enter the domain name / localhost, to view the Espocrm application.

If you see the following screen the installation was successful



- Enter the details for your newly created MySQL database.

The image shows the 'Database configuration' form in EspoCRM. The form has a blue header with the EspoCRM logo. Below the header, the title 'Database configuration' is centered. The form contains four input fields: 'Host Name *' with 'localhost' entered, 'Database Name *' with 'espocrm' entered, 'Database User Name *' with 'espo_user' entered, and 'Database User Password' with a masked password '*****'. Below these fields is a 'Test Connection' button. At the bottom right, there are 'Back' and 'Next' buttons.

- After entering everything correctly. Then, you will have EspoCRM installed and ready to use.
- Enter user name and password to login,

EspoCRM

Username

Password

Login

EspoCRM

System settings

Date Format
MM/DD/YYYY

Time Format
HH:mm

Time Zone
UTC

First Day of Week
Sunday

Default Currency
USD

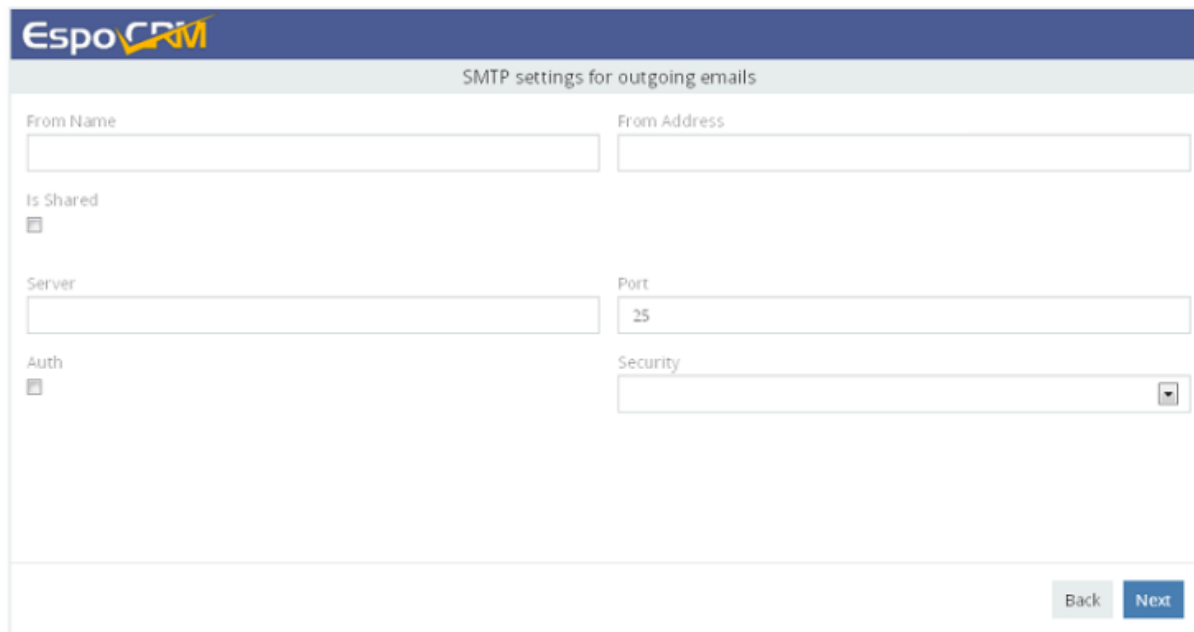
Thousand Separator
.

Decimal Mark *
.

Language
English (United States)

BackNext

- Enter SMTP settings for outgoing emails. This step can be skipped by clicking the *Next* button.



The image shows the 'SMTP settings for outgoing emails' form in EspoCRM. The form has a blue header with the EspoCRM logo. Below the header, the title 'SMTP settings for outgoing emails' is centered. The form contains several input fields: 'From Name' and 'From Address' (text boxes), 'Is Shared' (checkbox), 'Server' (text box), 'Port' (text box with '25' entered), 'Auth' (checkbox), and 'Security' (dropdown menu). At the bottom right, there are 'Back' and 'Next' buttons.

The installation was done , now we need to set up a corn

Set up the crontab expression :

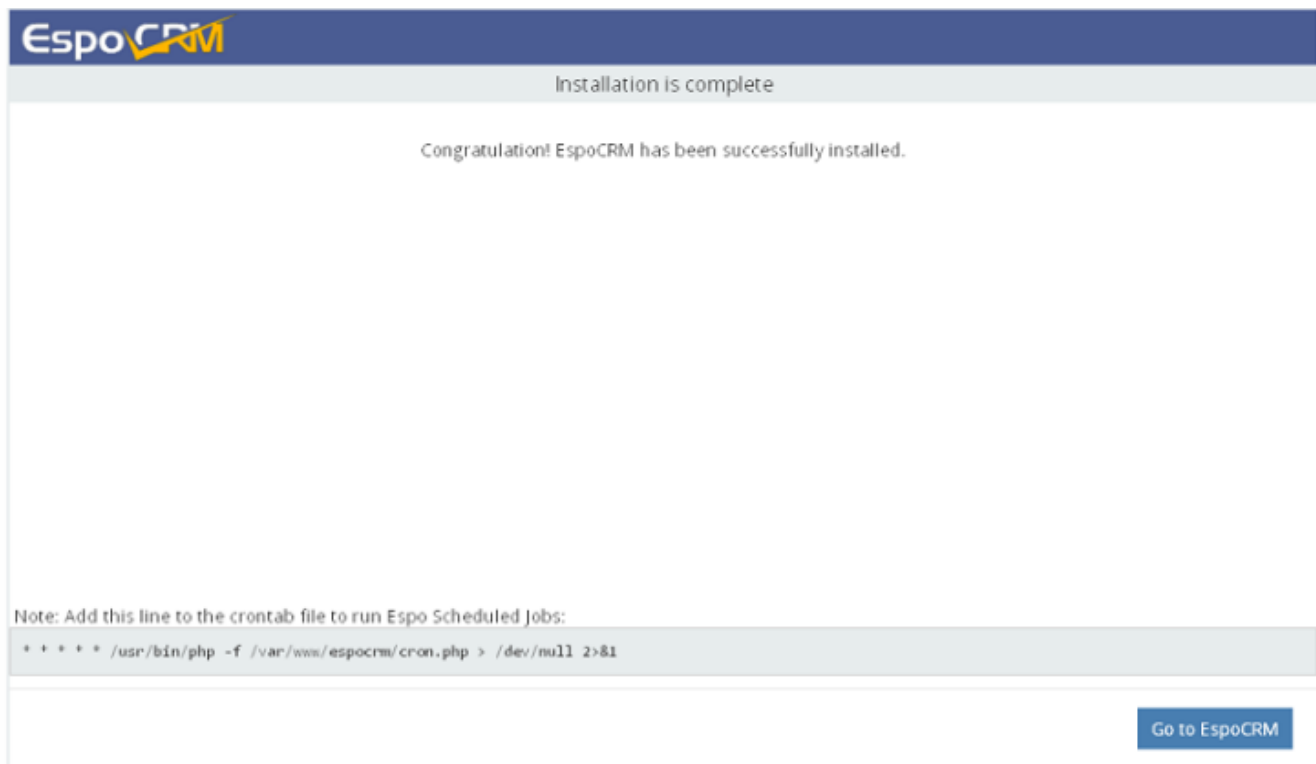
➤ Edit crontab to enable automation of EspoCRM tasks using Crontab.

Open a terminal and run this command

```
crontab -e -u WEBSERVER_USER
```

➤ Copy the below expression and Add the following expression at the bottom of the file. Save and close the file.

```
* * * * */usr/bin/php -f /var/www/html/espocrm/cron.php > /dev/null 2>&1
```



prometheus:

Prometheus is a software application used for event monitoring and alerting. It records real-time metrics in a time series database

For Monitoring of server metrics we use prometheus

For security purposes, we'll begin by creating two new user accounts, prometheus and node_exporter. We'll use these accounts throughout the tutorial to isolate the ownership on Prometheus' core files and directories.

Create these two users, and use the --no-create-home and --shell /bin/false options so that these users can't log into the server.

```
sudo useradd --no-create-home --shell /bin/false prometheus
sudo useradd --no-create-home --shell /bin/false node_exporter
```

Before we download the Prometheus binaries, create the necessary directories for storing Prometheus' files and data. Following standard Linux conventions, we'll create a directory in /etc for Prometheus' configuration files and a directory in /var/lib for its data.

```
sudo mkdir /etc/prometheus
sudo mkdir /var/lib/prometheus
```

Installation of prometheus

Go to the prometheus.io official site and search for the latest version of prometheus

First, download and unpack the current stable version of Prometheus in /opt directory.

```
cd /opt
```

```
wget
```

```
https://github.com/prometheus/prometheus/releases/download/v2.38.0/prometheus-2.38.0.linux-amd64.tar.gz
```

Now, unpack the downloaded archive

```
tar xvfz prometheus-2.38.0.linux-amd64.tar.gz
```

This will create a directory called prometheus-2.38.0.linux-amd64 containing two binary files (prometheus and promtool), consoles and console_libraries directories containing the web interface files, a licence, a notice, and several example files.

Copy the two binaries to the /usr/local/bin directory.

```
sudo cp prometheus-2.38.0.linux-amd64/prometheus /usr/local/bin/  
sudo cp prometheus-2.38.0.linux-amd64/promtool /usr/local/bin/
```

Set the user and group ownership on the binaries to the prometheus user created in Step 1.

```
sudo chown prometheus:prometheus /usr/local/bin/prometheus  
sudo chown prometheus:prometheus /usr/local/bin/promtool
```

Copy the consoles, console_libraries directories and prometheus.yml file to /etc/prometheus.

```
sudo cp -r prometheus-2.38.0.linux-amd64/consoles /etc/prometheus  
sudo cp -r prometheus-2.38.0.linux-amd64/console_libraries /etc/prometheus  
sudo cp prometheus-2.38.0.linux-amd64/prometheus.yml /etc/prometheus
```

Set the user and group ownership on the directories to the prometheus user. Using the -R flag will ensure that ownership is set on the files inside the directory as well.

```
sudo chown -R prometheus:prometheus /etc/prometheus/consoles  
sudo chown -R prometheus:prometheus /etc/prometheus/console_libraries  
sudo chown prometheus:prometheus /etc/prometheus/prometheus.yml
```

Configuring Prometheus:

The file /etc/prometheus/prometheus.yml is used for configuring the prometheus use the vim editor and add the following content

```
global:  
  scrape_interval: 15s
```

```
scrape_configs:
```

```
- job_name: 'prometheus'
  scrape_interval: 5s
  static_configs:
    - targets: ['localhost:9090']
```

Understand the prometheus.yml configuration file:

```
global:
  scrape_interval: 15s
```

These initial two lines tell about the global settings, define the default interval for scraping metrics. Note that Prometheus will apply these settings to every exporter unless an individual exporter's own settings override the globals.

This `scrape_interval` value tells Prometheus to collect metrics from its exporters every 15 seconds, which is long enough for most exporters.

Next comes to the prometheus Now, add Prometheus itself to the list of exporters to scrape from with the following `scrape_configs`

```
scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
```

Prometheus uses the `job_name` to label exporters in queries and on graphs, so be sure to pick something descriptive here.

As Prometheus exports important data about itself that you can use for monitoring performance and debugging, we've overridden the global `scrape_interval` directive from 15 seconds to 5 seconds for more frequent updates.

Lastly, Prometheus uses the `static_configs` and `targets` directives to determine where exporters are running. Since this particular exporter is running on the same server as Prometheus itself, we can use `localhost` instead of an IP address along with the default port, `9090`.

We need install the prometheus as service so we should place a service file in `/etc/systemd/system/` Directory

```
sudo vim /etc/systemd/system/prometheus.service
```

Add the following code to prometheus.service file

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file /etc/prometheus/prometheus.yml \
--storage.tsdb.path /var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries
[Install]
WantedBy=multi-user.target
```

then reload the systemd to make effect of prometheus service

```
sudo systemctl daemon-reload
```

then start the prometheus service using following command :

```
sudo systemctl start prometheus
```

to check the service is started or not run the below command :

```
sudo systemctl status prometheus
```

Install the node exporter:

node exporter exposes a wide variety of hardware and kernel related metrics.

Create a user to node exporter:

```
sudo useradd --no-create-home --shell /bin/false node_exporter
```

To download node exporter run the below command on terminal:
using wget

https://github.com/prometheus/node_exporter/releases/download/v1.4.0-rc.0/node_exporter-1.4.0-rc.0.linux-amd64.tar.gz

to unzip the tar file run below command:

```
tar xvzf node_exporter-1.4.0-rc.0.linux-amd64.tar.gz
```

copy the binary to the /usr/local/bin directory and set the user and group ownership to the node_exporter :

```
sudo cp node_exporter-1.4.0-rc.0.linux-amd64/node_exporter /usr/local/bin
```

```
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter
```

create a node_exporter service file :

```
sudo vim /etc/systemd/system/node_exporter.service
```

Add the following code to node_exporter.service file

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target
[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=multi-user.target
```

then reload the systemd

```
sudo systemctl daemon-reload
```

to start the node_exporter service using following command :

```
sudo systemctl start node_exporter
```

to check the service is started or not run the below command :

```
sudo systemctl status node_exporter
```

Configuring Prometheus to Scrape Node Exporter:

Because Prometheus only scrapes exporters which are defined in the **scrape_configs** portion of its configuration file, we'll need to add an entry for Node Exporter, just like we did for Prometheus itself.

```
vim /etc/prometheus/prometheus.yml
```

At the end of the scrape_configs block, add a new entry called node_exporter.

```
- job_name: 'node_exporter'
  scrape_interval: 5s
  Static_configs:
  - targets: ['localhost:9100']
```

Save the file

Because this exporter is also running on the same server as Prometheus itself, we can use localhost instead of an IP address again along with Node Exporter's default port, 9100. Your whole configuration file should look like this:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'node_exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9100']
```

Finally, restart Prometheus to put the changes into effect.

```
sudo systemctl restart prometheus
```

verify that everything is running correctly with the status command

```
sudo systemctl status prometheus
```

the configuration file looks like this save the file and exit

Finally Restart the prometheus service:

```
sudo systemctl restart prometheus
```

check the service is started or not run the below command

```
sudo systemctl status prometheus
```

GRAFANA:

Grafana is a free, open-source, data visualisation platform. It is used for monitoring, analysis, and visualisation of real-time system data. Its frontend is written in Typescript while the backend is written in Go. It can be used with time series databases such as InfluxDB, Prometheus, and Elasticsearch. It provides a beautiful dashboard that allows users to create and edit both log and data graphs and create metrics

First, install all required dependencies using the following command:

```
apt-get install gnupg2 apt-transport-https software-properties-common -y
```

Next, download and add the Grafana GPG key with the following command:

```
wget -q -O - https://packages.grafana.com/gpg.key | apt-key add -
```

Next, add the Grafana repository to APT using the following command:

```
echo "deb https://packages.grafana.com/oss/deb stable main" | tee -a  
/etc/apt/sources.list.d/grafana.list
```

Once the repository is added to your system, you can update it with the following command:

```
apt-get update -y
```

Install Grafana:

Now, you can install the Grafana by running the following command:

```
apt-get install grafana -y
```

Now, start the Grafana service and enable it to start at system reboot:

```
systemctl start grafana-server  
systemctl enable grafana-server
```

You can now check the status of the Grafana with the following command:

```
systemctl status grafana-server
```

Add the grafana.conf in /etc/nginx/sites-available

```
vim /etc/nginx/sites-available/grafana.conf
```

Add the following lines:

```
server {  
    server_name grafana.example.com;  
    listen 8090 ;  
    access_log /var/log/nginx/grafana.log;  
  
    location / {  
        proxy_pass http://localhost:3000;  
        proxy_set_header Host $http_host;  
        proxy_set_header X-Forwarded-Host $host:$server_port;
```



```
    proxy_set_header X-Forwarded-Server $host;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
  }  
}
```

Save and close the file then verify the Nginx configuration file using the following command:

```
nginx -t
```

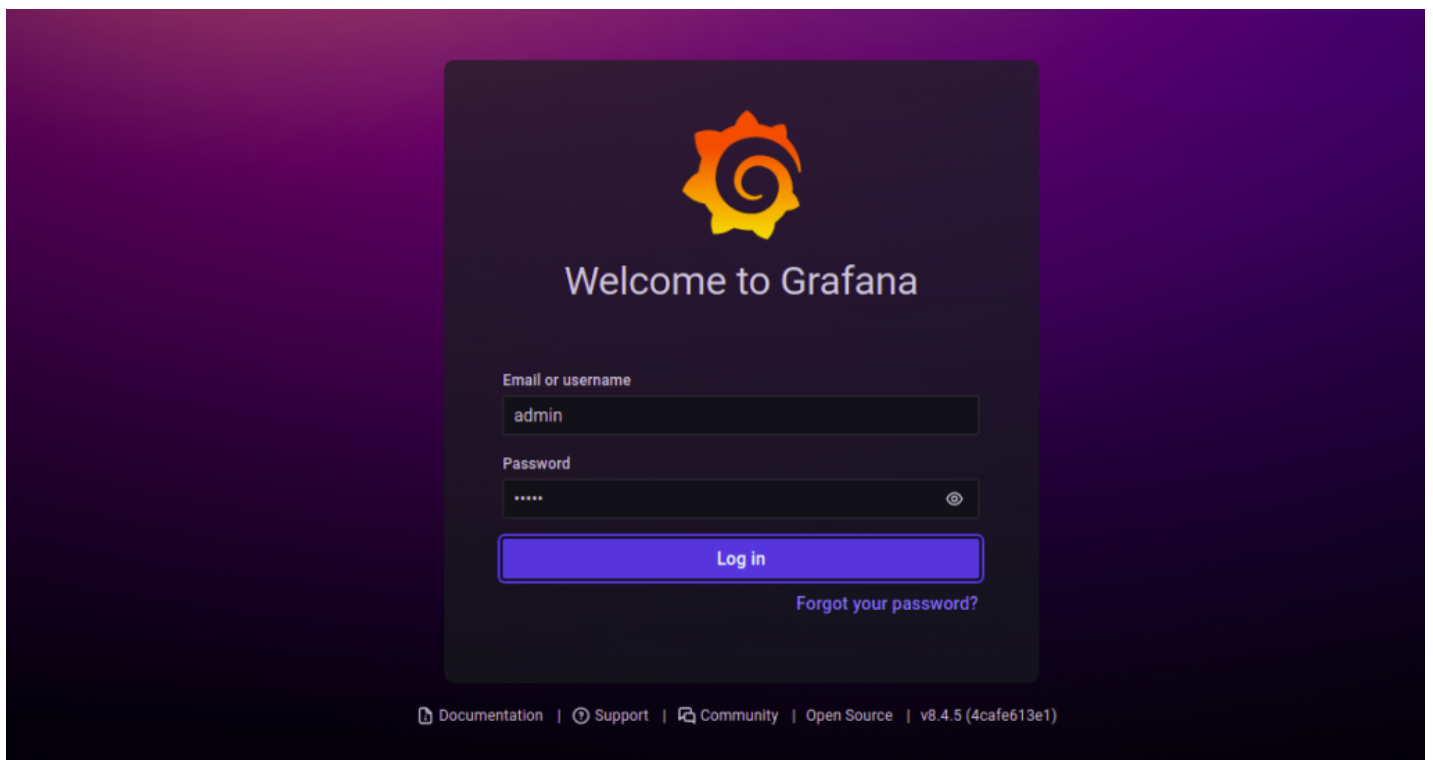
You will get the following output:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

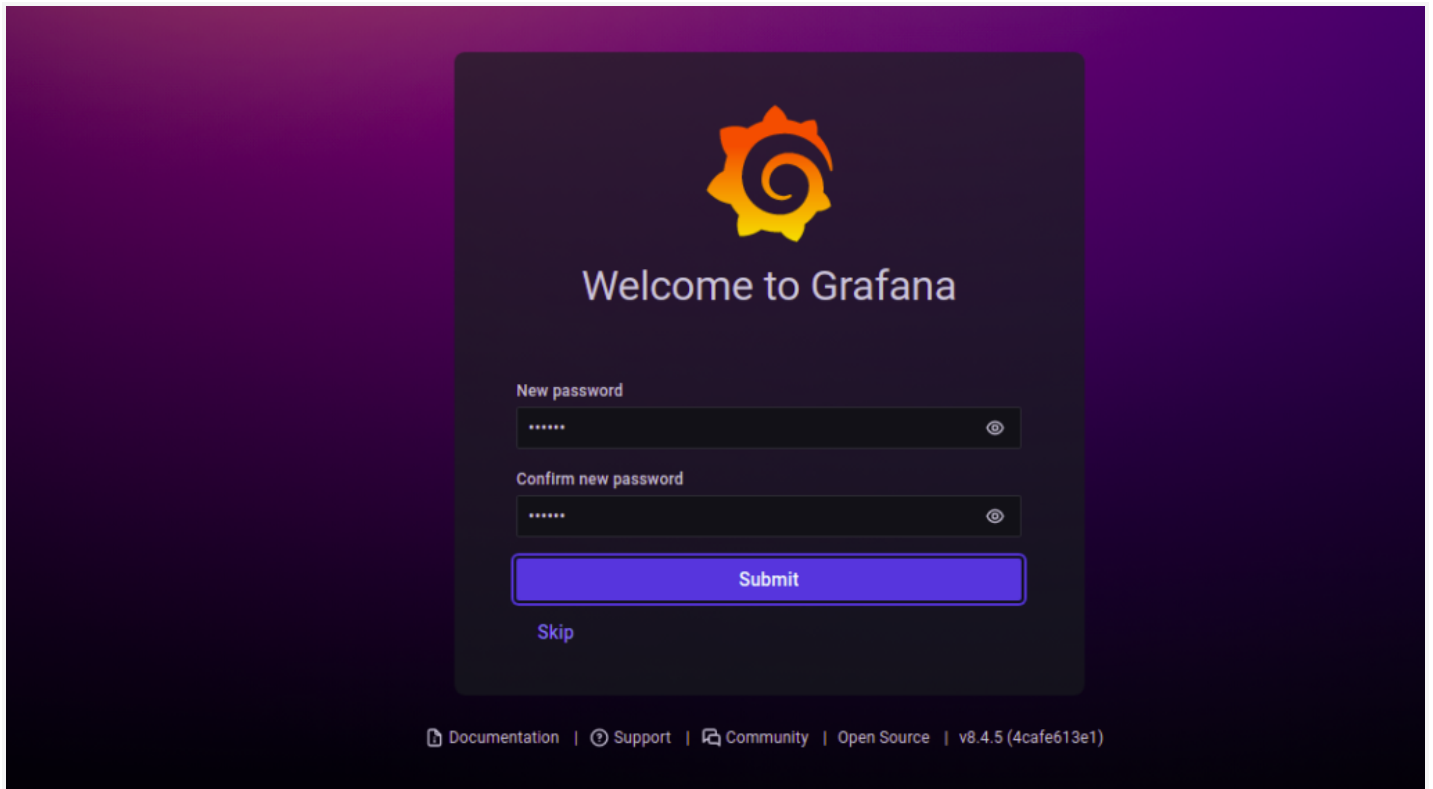
Finally, restart the Nginx service to apply the changes:

```
systemctl restart nginx
```

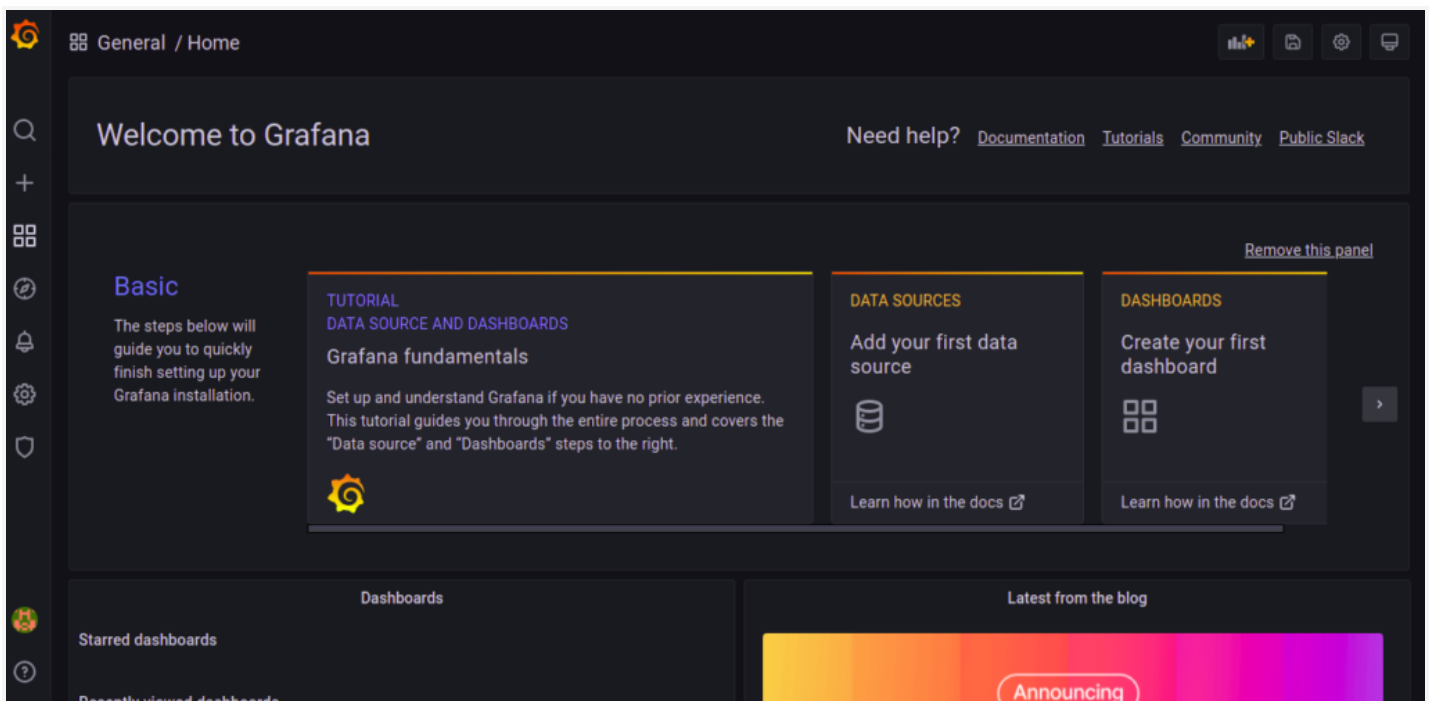
Now, open your web browser and access the Grafana dashboard using the URL **localhost:8090**. You will be redirected to the Grafana login page:



Provide default admin username and password as admin/admin and click on the **Log in** button. You should see the Grafana password change screen:



Change your default password and click on the **Submit** button. You should see the Grafana dashboard on the following screen:



To add data sources follow the below navigation

Then Navigate to dashboard-->datasource-->add datasource--> prometheus



Home



Search dashboards



Dashboards



Explore



Alerting



Configuration



Data sources

Users

Teams

Plugins

Preferences

API keys



Server Admin

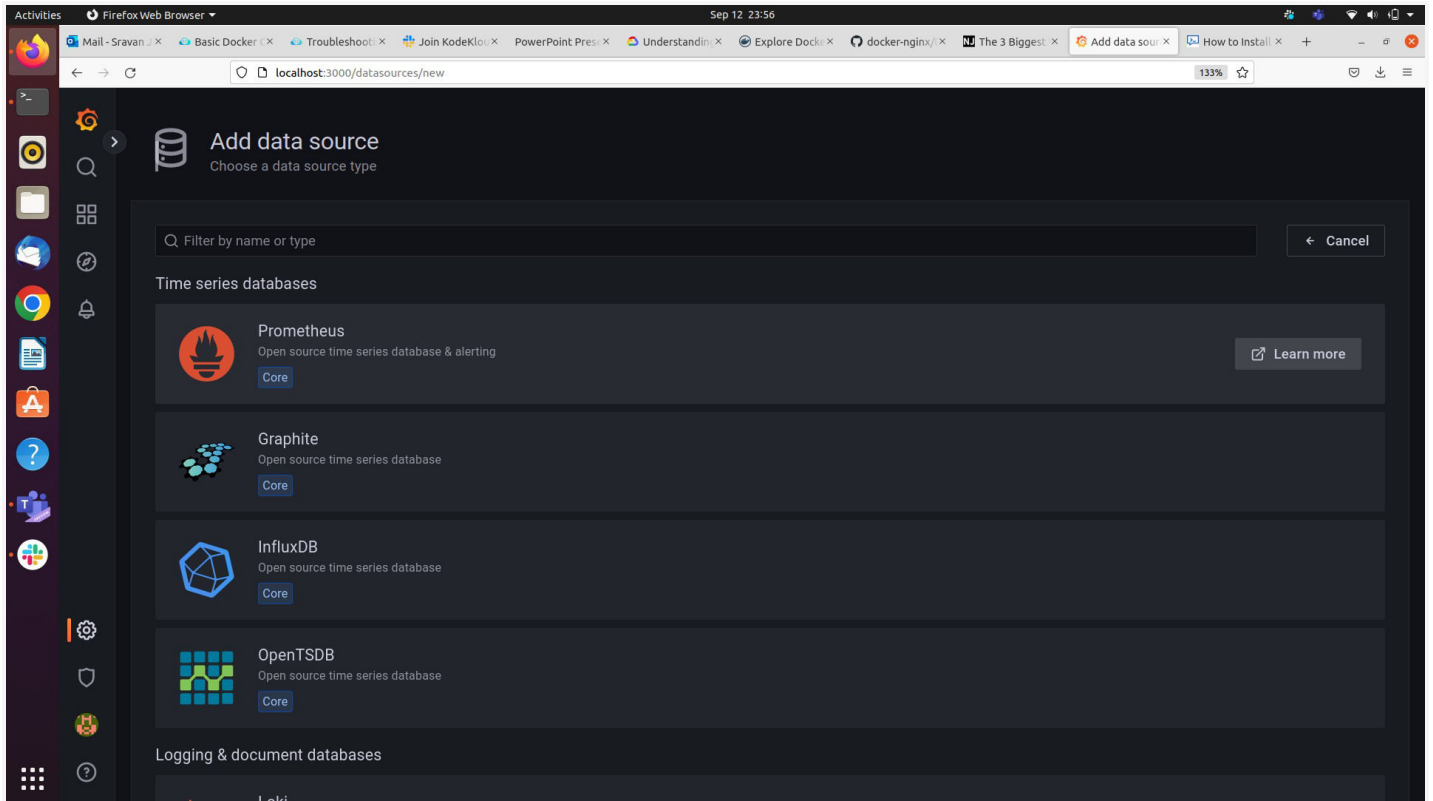


admin



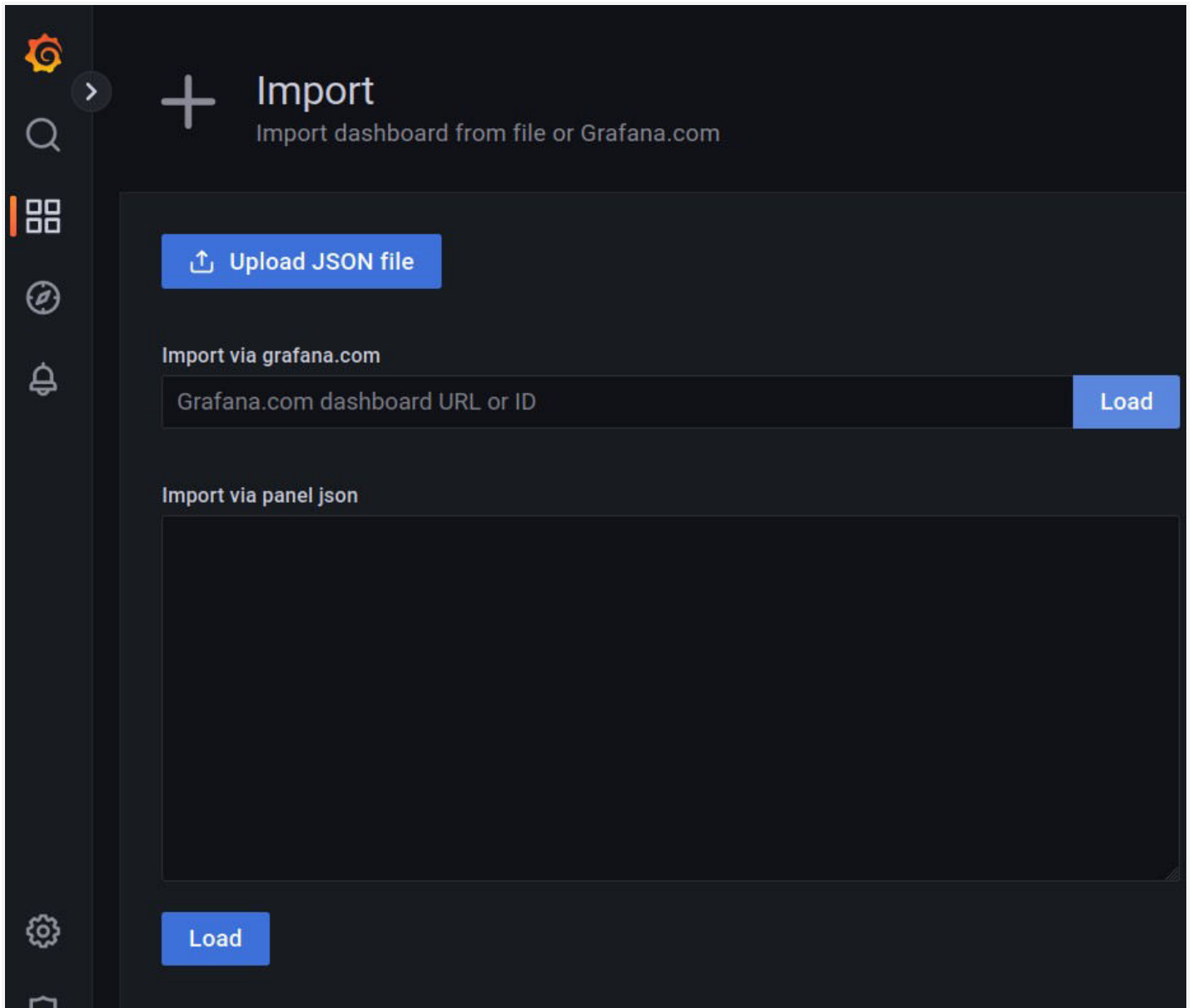
Help



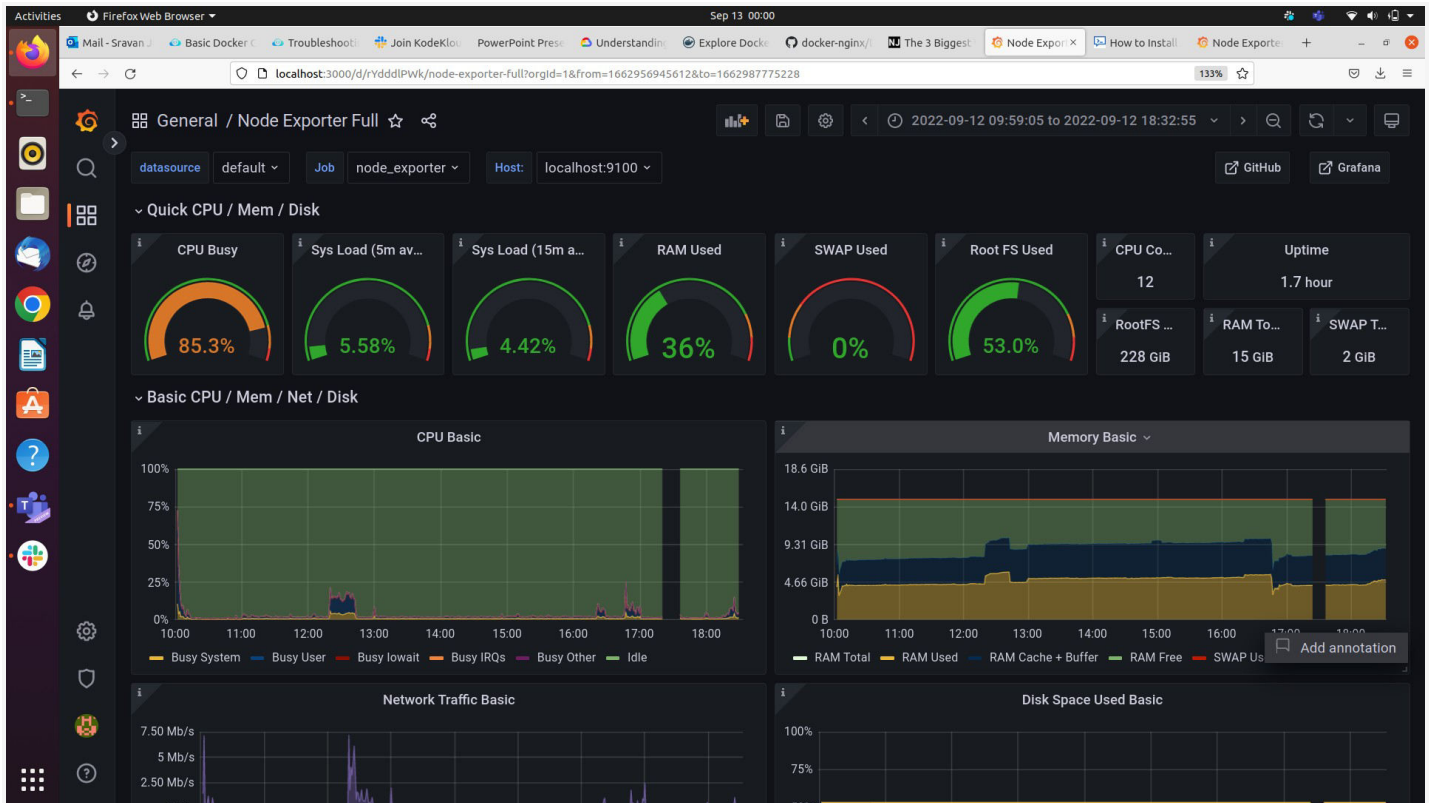


Click on prometheus and then add the promethues url path to it(<http://localhost:9000>).

Then navigate to dashboard-->import
then add the node exporter full id in the and load it.(1860)



After adding the node_exporter full id then we will see the below gauges to show the metrics



Thank you