**PROMETHEUS AND GRAFANA**

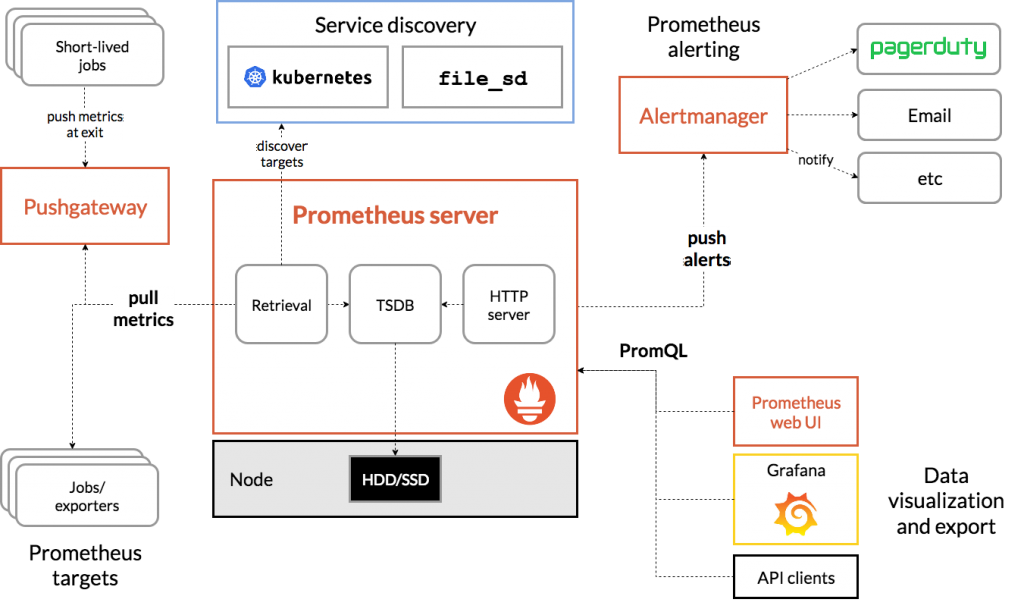
**What is Prometheus?**

[Prometheus](https://newrelic.com/solutions/prometheus-monitoring) is an open-source monitoring solution written in Go that collects metrics data and stores that data in a time series database. It was originally built by SoundCloud in 2012 and became part of the [Cloud Native Computing Foundation](https://www.cncf.io/) (CNCF) in 2016. It uses PromQL, a powerful query language for querying your time series data.

**Features of Prometheus**

* A multi-dimensional data model with time series data identified by metric name and key/value pairs
* PromQL, a flexible query language to leverage this dimensionality
* No reliance on distributed storage; single server nodes are autonomous
* Time series collection happens via a pull model over HTTP
* Pushing time series is supported via an intermediary gateway
* Targets are discovered via service discovery or static configuration
* Multiple modes of graphing and dashboarding support

**Architecture of Prometheus**

****

**What is Grafana?**

Grafana helps us by allowing us to query, visualize, alert and explore our metrics no matter where they are stored, it also helps in providing us with the tools for turning our time series database data into beautiful graphs and visualizations.

**Grafana Dashboard**

Grafana Dashboard contains data form the plugged-in data sources like Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL, etc. The dashboard contains a gamut of visualization options such as geo maps, heat maps, histograms, all the variety of charts & graphs which a business typically requires to study data.

**Uses of Grafana in Devops**

Grafana is an open source, feature-rich metrics dashboard and graph editor. It supports Graphite, Elasticsearch, OpenTSDB, Prometheus, and InfluxDB analytics services. Use the Grafana service hook to annotate Grafana dashboards upon completion of Azure Pipelines deployments.

**Need Prometheus and Grafana**

Prometheus collects rich metrics and provides a powerful querying language; Grafana transforms metrics into meaningful visualizations. Both are compatible with many, if not most, data source types.

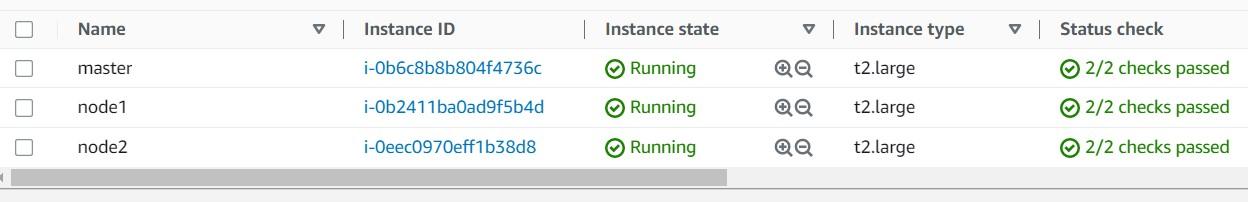
Prometheus is an open-source data monitoring tool. The combination of [Prometheus and Grafana](http://docs.grafana.org/features/datasources/prometheus/) is the de-facto combination leveraged in the industry for deploying a data visualization setup. Grafana dashboard is used for visualizing the data whereas the backend is powered by [Prometheus](https://prometheus.io/).

**How to install Prometheus and Grafana in Amazon linux 2**

Here First we will launch the three AWS ec2-instances and we will create Kubernetes cluster and then in the next step will install Prometheus and Grafana to monitor the ec2 machines.

**Step1 : Launch 3 Amazon Linux 2 ec2 machines(Master, node1 and node2) with t2.large Instance Type**

Lanch the required ec2-instances with required configurations



**Step2 : Installing Docker and Kubernetes on master**

Connect to the master machine shell execute the following commands

* Switch to root user and create a file called master.sh with docker and Kubernetes installation commands in it.
* Create a new file with a name of master.sh

vi master.sh

* Paste following content in the file

hostnamectl set-hostname K8s-Master

yum update -y

yum install git -y

amazon-linux-extras install java-openjdk11 -y

yum install docker -y

systemctl enable docker

systemctl restart docker

systemctl status docker

free -mh

sestatus

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch

enabled=1

gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kubelet kubeadm kubectl

EOF

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

systemctl enable kubelet

systemctl restart kubelet

kubeadm init

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

export KUBECONFIG=/etc/kubernetes/admin.conf

kubeadm token create --print-join-command

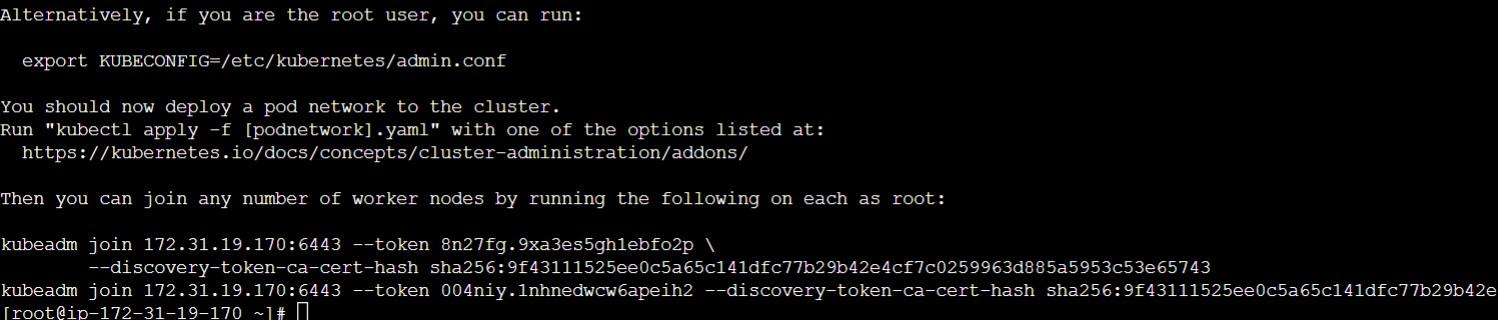
* Give execution permission to master.sh file

chmod +x master.sh

* Run the above master.sh file

./master.sh

* After running master.sh file you will get kubeadm join command copy and keep, we will use that join command to join our nodes to master.

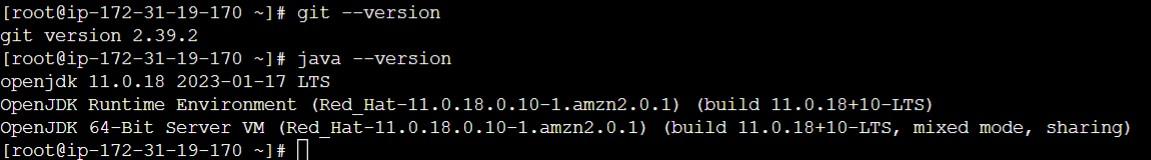


Check git version

git –version

Check java version

java --version



**Step 3 : Adding node1 and node2 to Kubernetes master**

* Connect to node1 ec2 machine and Switch to root user

sudo -i

* Create a file called node1.sh and put following commands in it. And give execute permission to node1.sh file.
* Create a file name with node1.sh

vi node1.sh

* Insert the following commands in it.

hostnamectl set-hostname node1

yum update -y

amazon-linux-extras install java-openjdk11 -y

yum install git -y

yum install docker -y

systemctl enable docker

systemctl restart docker

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch

enabled=1

gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kubelet kubeadm kubectl

EOF

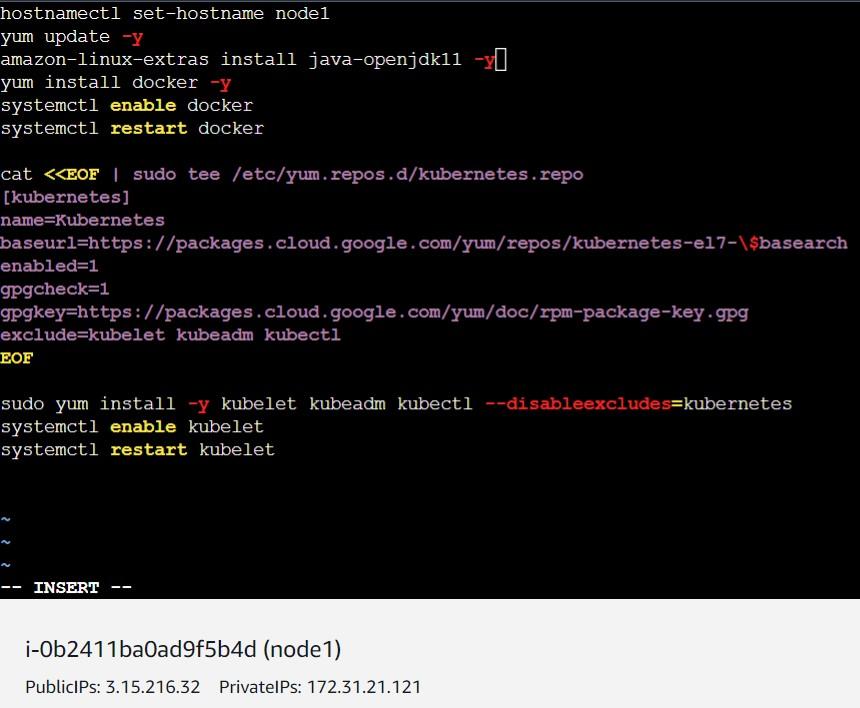
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

systemctl enable kubelet

systemctl restart kubelet

* Your file will look like this. Give execute permission to this file and run it.

chmod +x node1.sh



* Execute the following command

./node1.sh

**Step4: Add node2 to Kubernetes Cluster by following same steps.**

* Open node2 ec2 machine and Switch to root user

sudo -i

* Create a file called node2.sh and put following commands in it. And give execute permission to node2.sh file.

vi node2.sh

* Put these commands in the file and run it.

hostnamectl set-hostname node2

yum update -y

amazon-linux-extras install java-openjdk11 -y

yum install git -y

yum install docker -y

systemctl enable docker

systemctl restart docker

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearchenabled=1

gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kubelet kubeadm kubectl

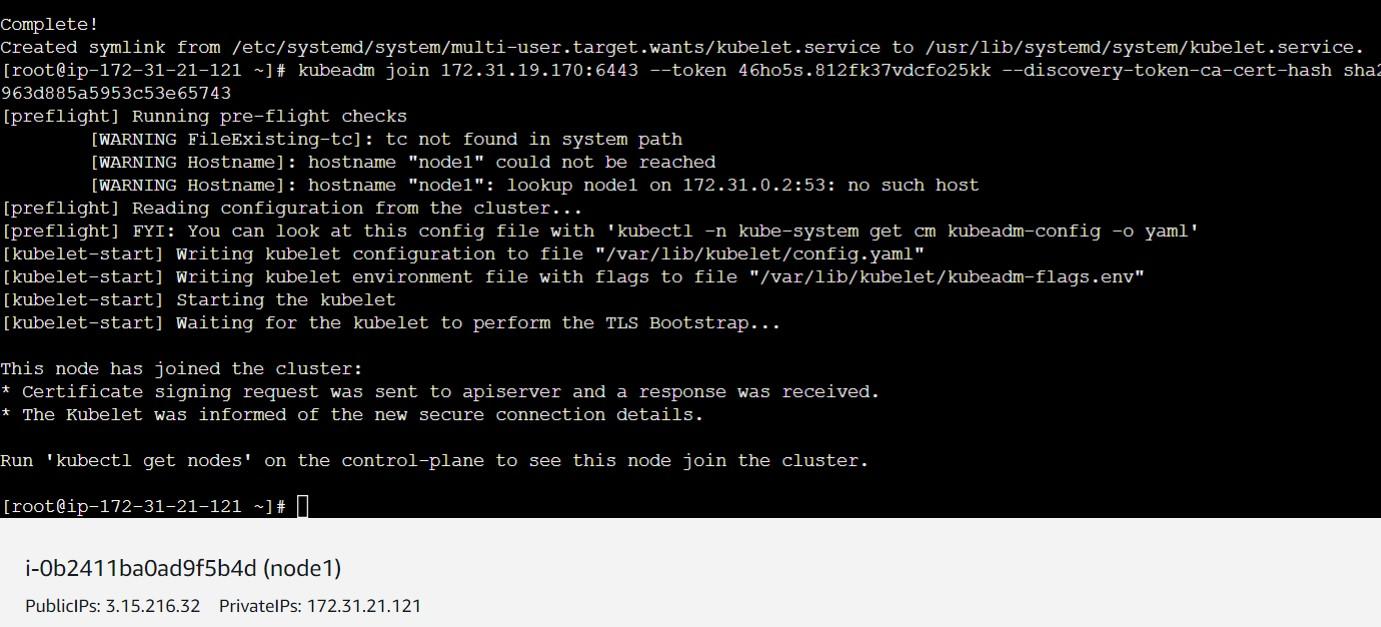
EOF

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

systemctl enable kubelet

systemctl restart kubelet

* Now run that Kubeadm join command on node1 and node2. **(Execute the join command only on node1 and node 2)**
* After running join command on node1 you will get this.



* After running join command on node2 you will get this.



* Execute the command to get the nodes but it shows nodes are **NotReady**

kubectl get nodes

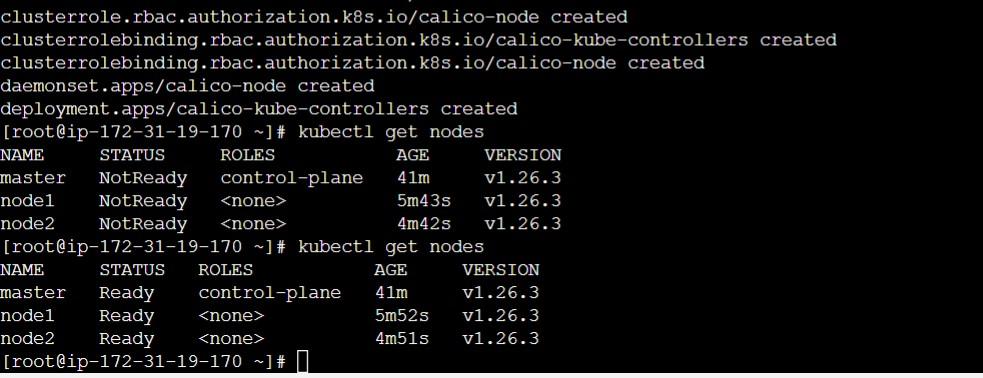
* Run the following commands on master node only. By adding this CNI the all the nodes get ready in our cluster

curl https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml -O

kubectl apply -f calico.yaml

* If all above steps followed correctly the cluster will be created and status will be changed to ready.

kubectl get nodes



**Step5 : Now lets clone our project and deploy it on Kubernetes cluster.**

* On Master node clone the git repository using the command
* Create a project directory and run git clone command inside it

git clone <https://github.com/ramdassbhanage/fitpro-k8s-project-new-git2.git>

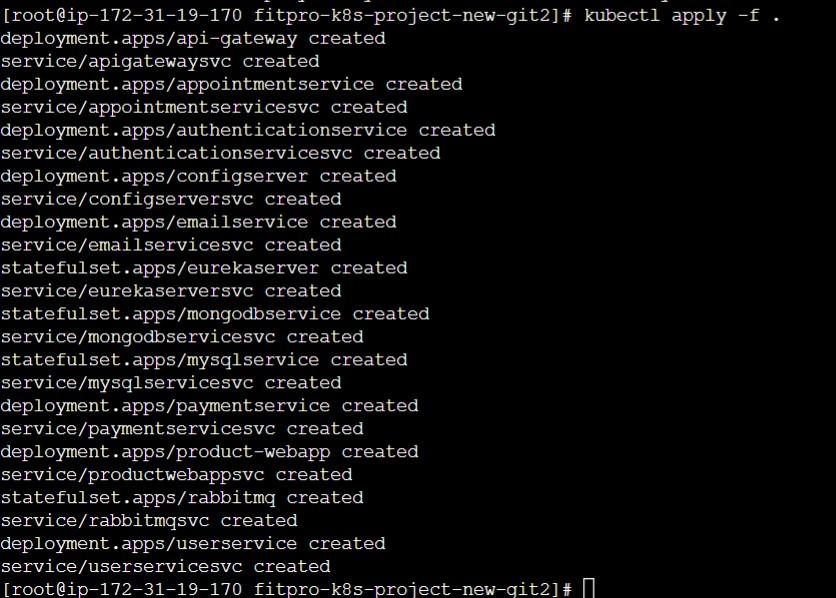
* After cloning you will get ready microservices yaml files. You just need to run them.



* Run the following command to deploy all the microservices to Kubernetes cluster

kubectl apply -f .

* After running apply command project is deployed



* Run the kubectl get pods command to list all pods

kubectl get pods

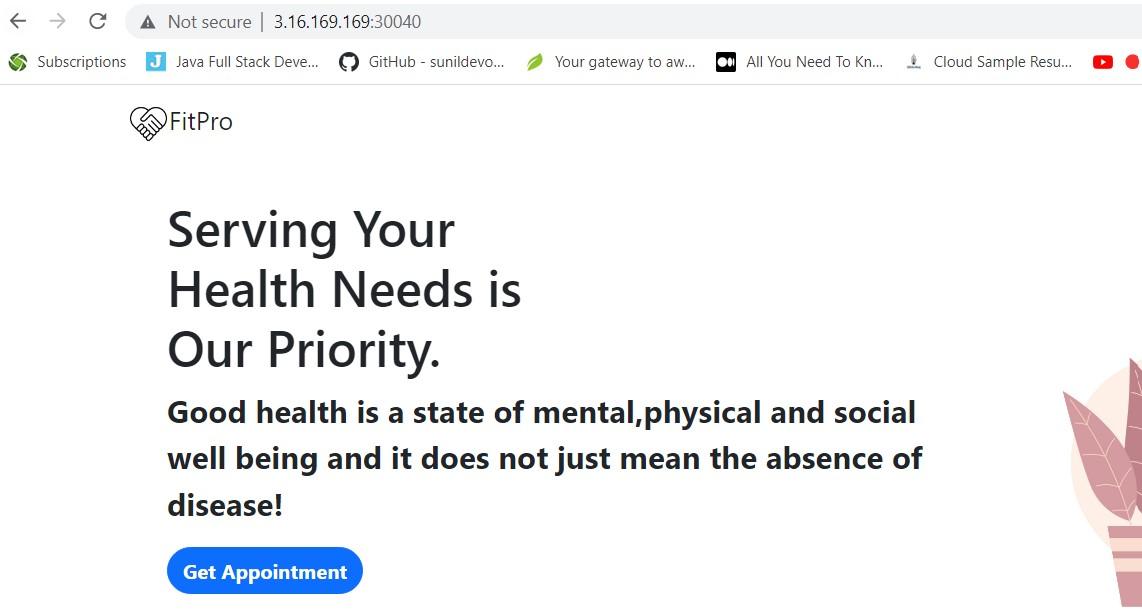
* Run the kubectl get all command to list all the pods and services

kubectl get all

* Now run kubecl get svc command to list all ports information

kubectl get svc

* Copy 5 digit port of apigateway service and paste it in the browser with <public ip> of your master machine.
* Example : (<http://3.16.169.169:30040/>)
* You will get this output in the browser



**Installing Prometheus and Grafana**

**Steps to install Prometheus**

**Step 1: Run these commands on master machine to install Prometheus**

* Add the repository

sudo tee /etc/yum.repos.d/prometheus.repo <<EOF

[prometheus]

name=prometheus

baseurl=https://packagecloud.io/prometheus-rpm/release/el/7/x86\_64

repo\_gpgcheck=1

enabled=1

gpgkey=https://packagecloud.io/prometheus-rpm/release/gpgkey

https://raw.githubusercontent.com/lest/prometheus-rpm/master/RPM-GPG-KEY-prometheus-rpm

gpgcheck=1

metadata\_expire=300

EOF

* We will the get the output like this after adding the repo



Run these commands

* Install Prometheus service

sudo yum -y install prometheus2

* Enable the Prometheus service

systemctl enable prometheus

* Start the Prometheus service

systemctl restart Prometheus

* Check the Prometheus service

systemctl status prometheus

* You can use rpm command to check for the version of Prometheus installed

rpm -qi prometheus2

* Execute the command to edit the file

vim /etc/prometheus/prometheus.yml

**write this all in the file**

**(paste the below lines in the file)**

# my global config

global:

  scrape\_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

  evaluation\_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.

  # scrape\_timeout is set to the global default (10s).

# Alertmanager configuration

alerting:

  alertmanagers:

  - static\_configs:

    - targets:

      # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation\_interval'.

rule\_files:

  # - "first\_rules.yml"

  # - "second\_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:

# Here it's Prometheus itself.

scrape\_configs:

  # The job name is added as a label `job=<job\_name>` to any timeseries scraped from this config.

  - job\_name: 'K8s-master'

    # metrics\_path defaults to '/metrics'

    # scheme defaults to 'http'.

    static\_configs:

    - targets: ['localhost:9090']

  # Pull host metrics with node exporter

  - job\_name: ‘Worker-node-1’

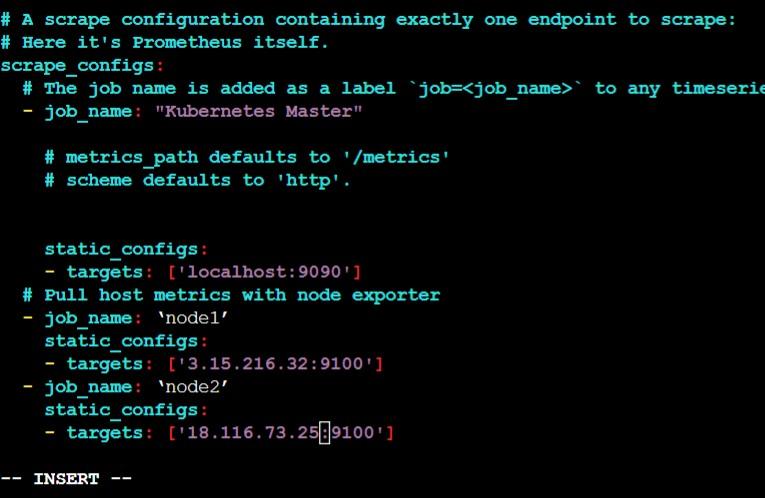
    static\_configs:

      - targets: ['<Replace with node1 public IP>:9100']

  - job\_name: ‘Worker-node-1’

    static\_configs:

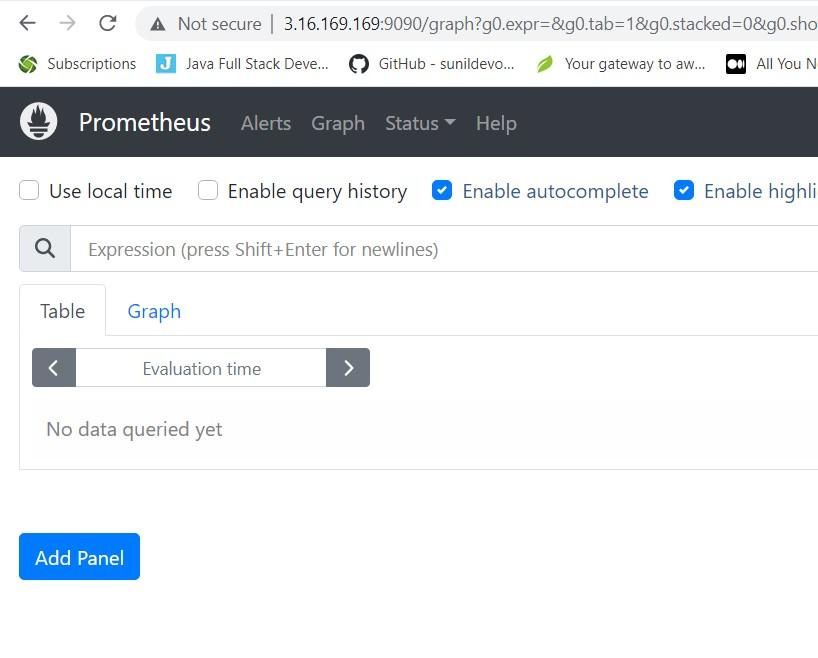
      - targets: ['<Replace with node2 public IP>:9100']



* Put public Ip address of node1 and node2 in local host
* After editing Prometheus.yml file restart it.

systemctl restart Prometheus

* Copy master node Public IP : 9090 and paste it in browser.



**Step 2 : Install node exporter on node1 and node2**

* Connect to node1 ec2 machine
* Add the repository

sudo tee /etc/yum.repos.d/prometheus.repo <<EOF

[prometheus]

name=prometheus

baseurl=https://packagecloud.io/prometheus-rpm/release/el/7/x86\_64

repo\_gpgcheck=1

enabled=1

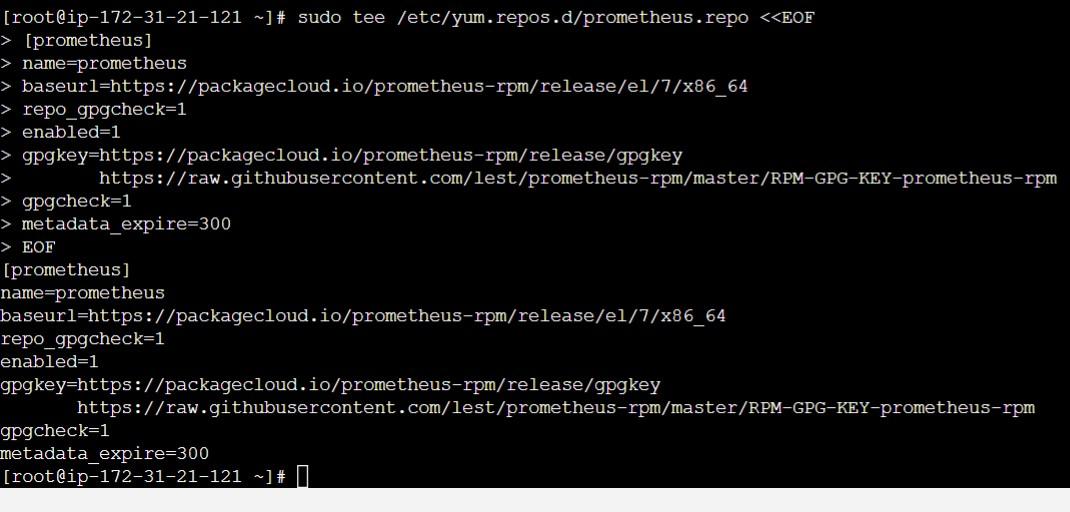
gpgkey=https://packagecloud.io/prometheus-rpm/release/gpgkey

https://raw.githubusercontent.com/lest/prometheus-rpm/master/RPM-GPG-KEY-prometheus-rpm

gpgcheck=1

metadata\_expire=300

EOF



* Run following commands
* Install the node exporter

yum -y install node\_exporter

* Enable the node exporter service

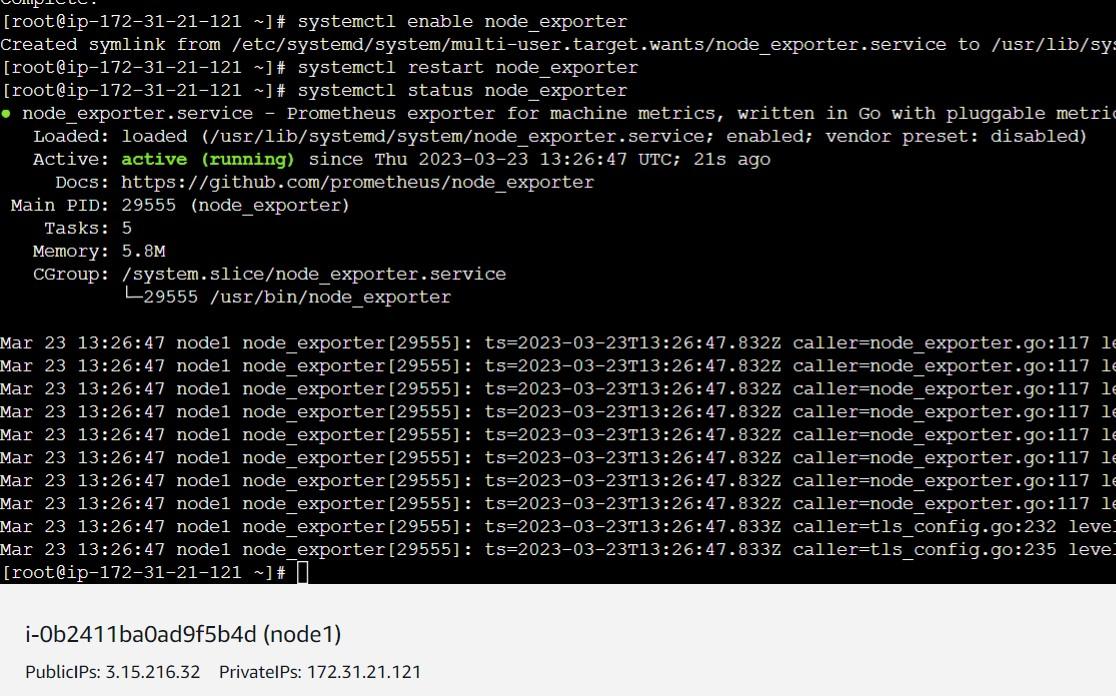
systemctl enable node\_exporter

* Start the node exporter service

systemctl restart node\_exporter

* Check the status of node exporter service

systemctl status node\_exporter



* Connect to node2 ec2 machine
* Add the repository

sudo tee /etc/yum.repos.d/prometheus.repo <<EOF

[prometheus]

name=prometheus

baseurl=https://packagecloud.io/prometheus-rpm/release/el/7/x86\_64

repo\_gpgcheck=1

enabled=1

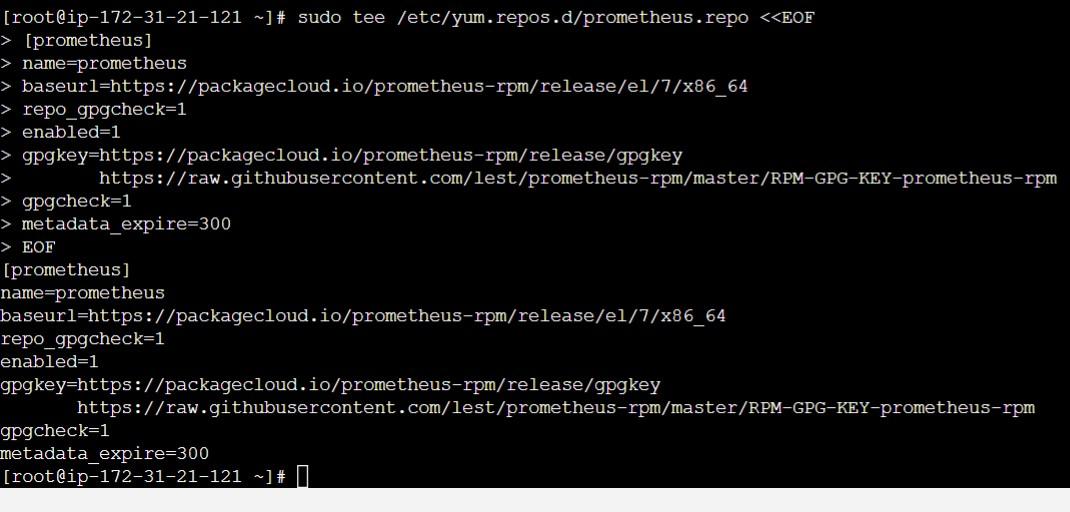
gpgkey=https://packagecloud.io/prometheus-rpm/release/gpgkey

https://raw.githubusercontent.com/lest/prometheus-rpm/master/RPM-GPG-KEY-prometheus-rpm

gpgcheck=1

metadata\_expire=300

EOF



* Run following commands
* Install the node exporter

yum -y install node\_exporter

* Enable the node exporter service

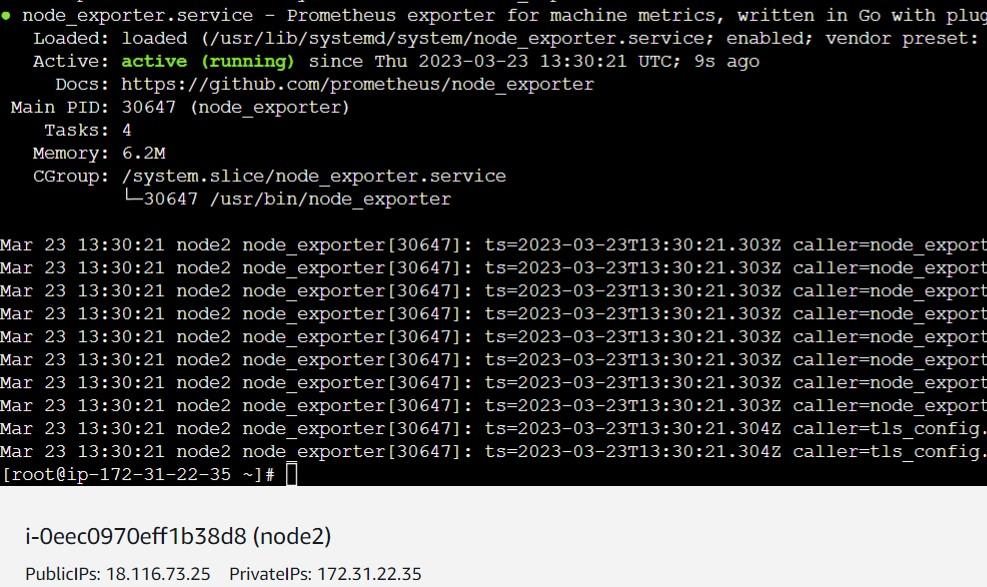
systemctl enable node\_exporter

* Start the node exporter service

systemctl restart node\_exporter

* Check the status of node exporter service

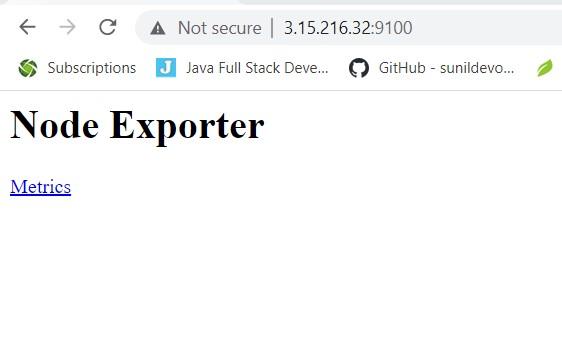
systemctl status node\_exporter



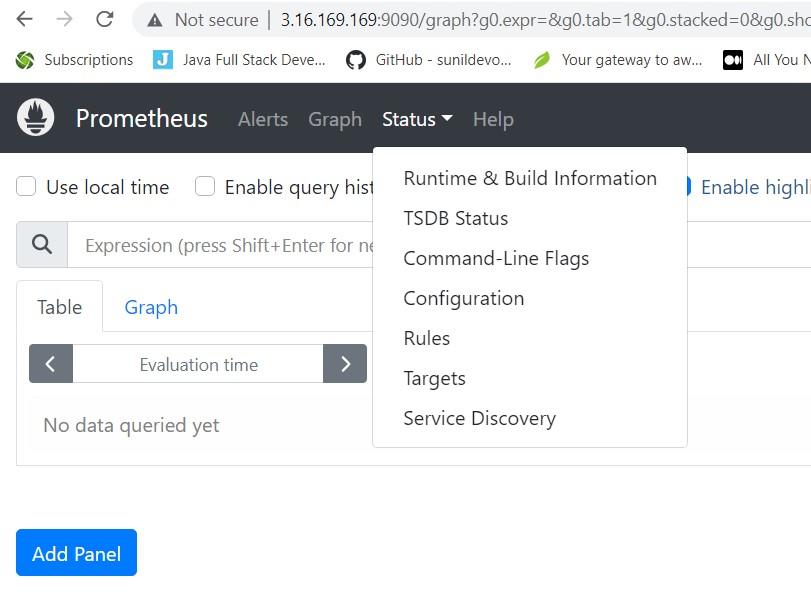
* Copy Node1 Public Ip and open it with 9100 port

Example: http://3.15.216.32:9100/

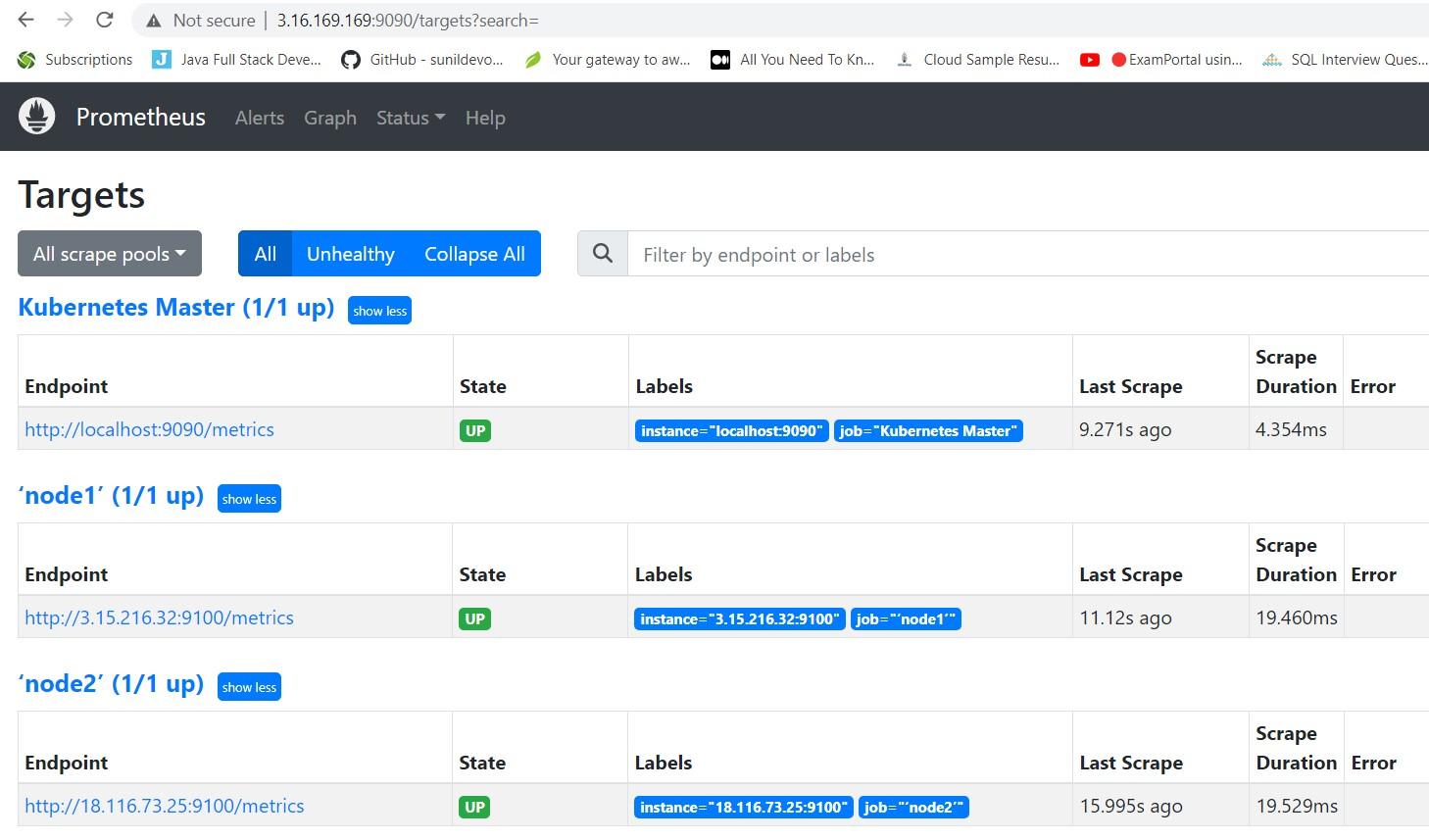
* You will get this in browser



* Now copy master node puplic ip and open it with 9090 port and goto to targets



* Click on targets. It will display you node1 and node2 machine



**Step 4: Installing Grafana on Master machine**

* Create **grafana.repo** file to install grafana in amazon linux 2

sudo vi /etc/yum.repos.d/grafana.repo

* Add following content in **grafana.repo** file

[grafana]

name=grafana

baseurl=https://rpm.grafana.com

repo\_gpgcheck=1

enabled=1

gpgcheck=1

gpgkey=https://rpm.grafana.com/gpg.key

sslverify=1

sslcacert=/etc/pki/tls/certs/ca-bundle.crt

* After adding the data file look like this



Run following commands to install grafana

* Install the Grafana service on master machine

sudo yum install grafana -y

* Start the Grafana service

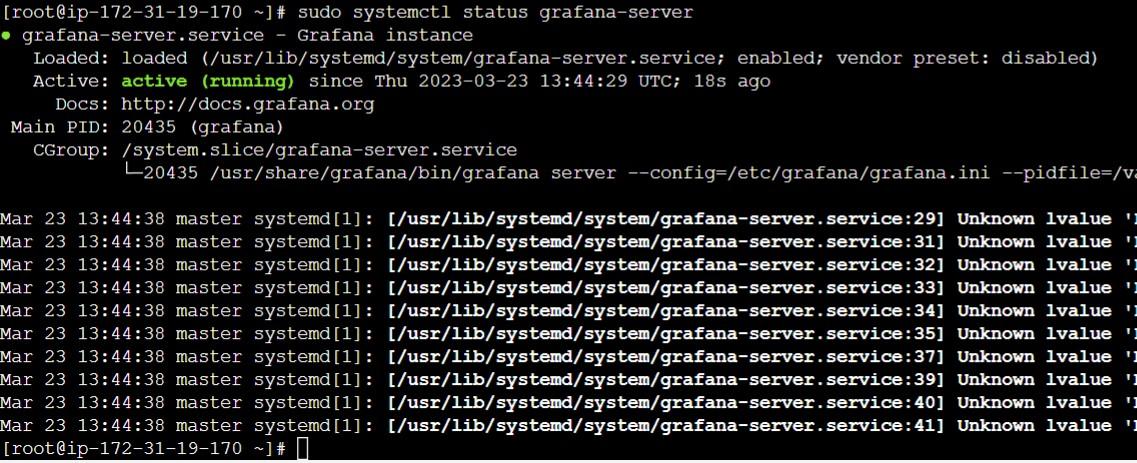
sudo systemctl start grafana-server

* Enable the Grafana service

sudo systemctl enable grafana-server

* Check the Status Grafana service

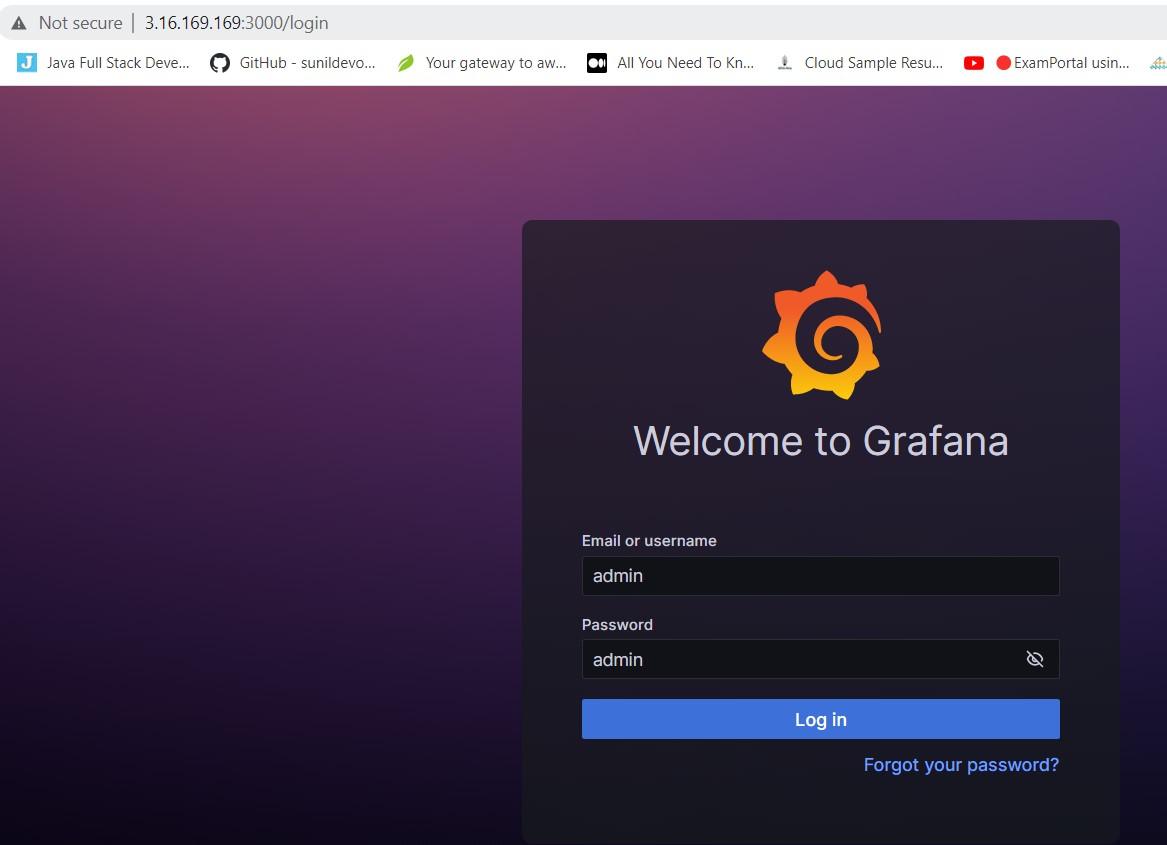
sudo systemctl status grafana-server

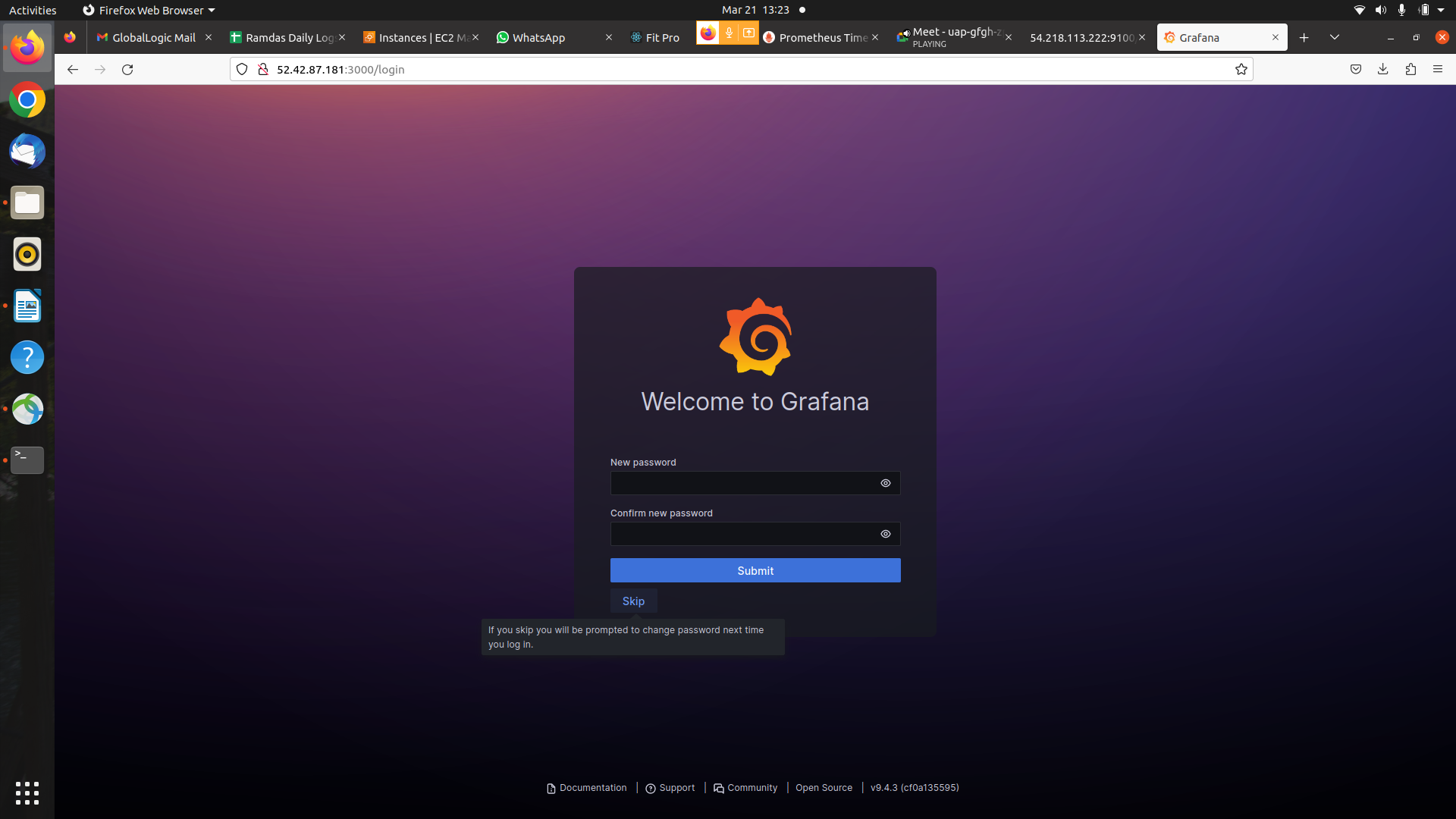


* Goto browser copy paste public IP of Master and 3000 port

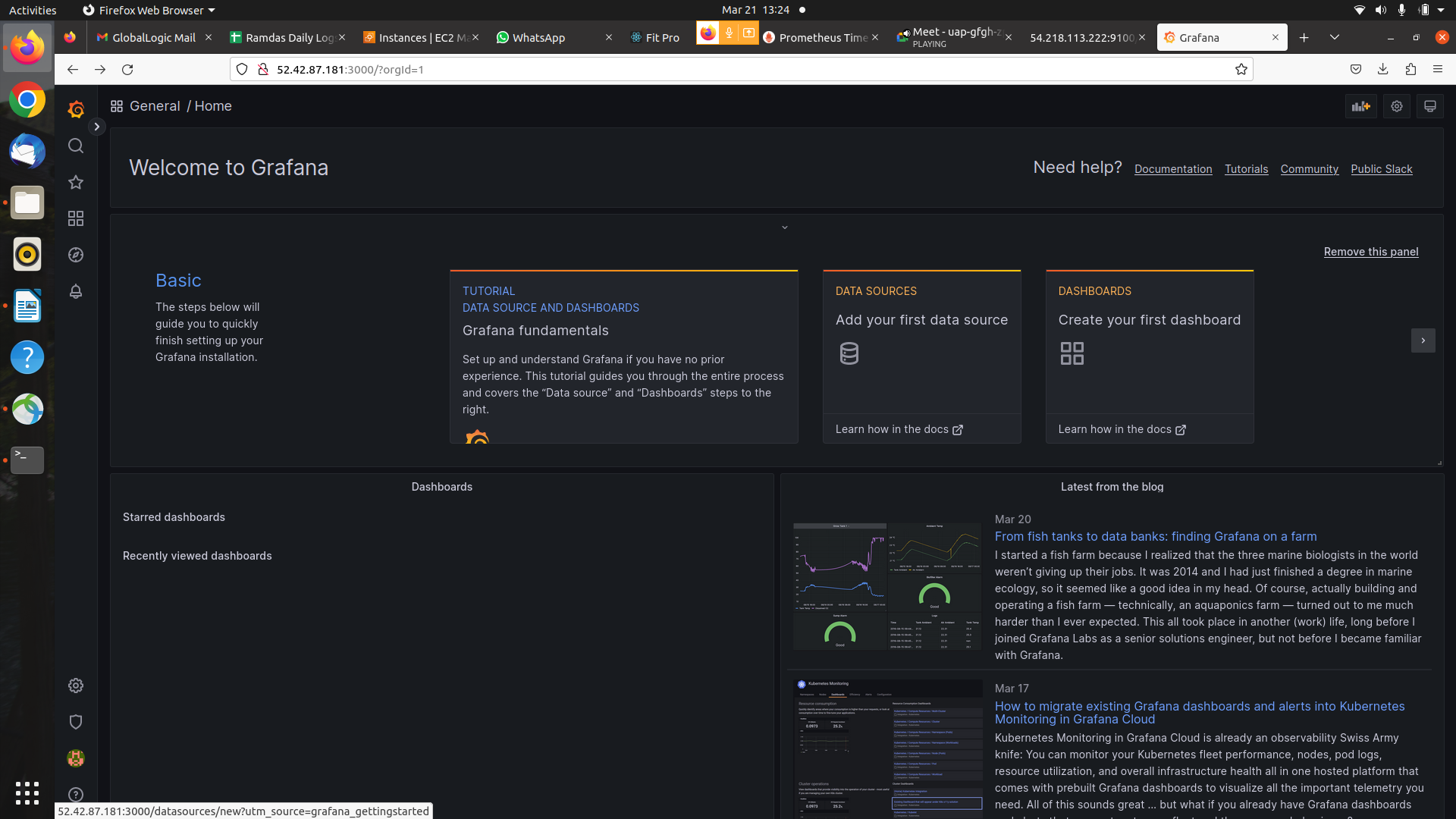
Example: <http://3.16.169.169:3000/>

* You will see Grafana dashboard use admin and admin for username and password

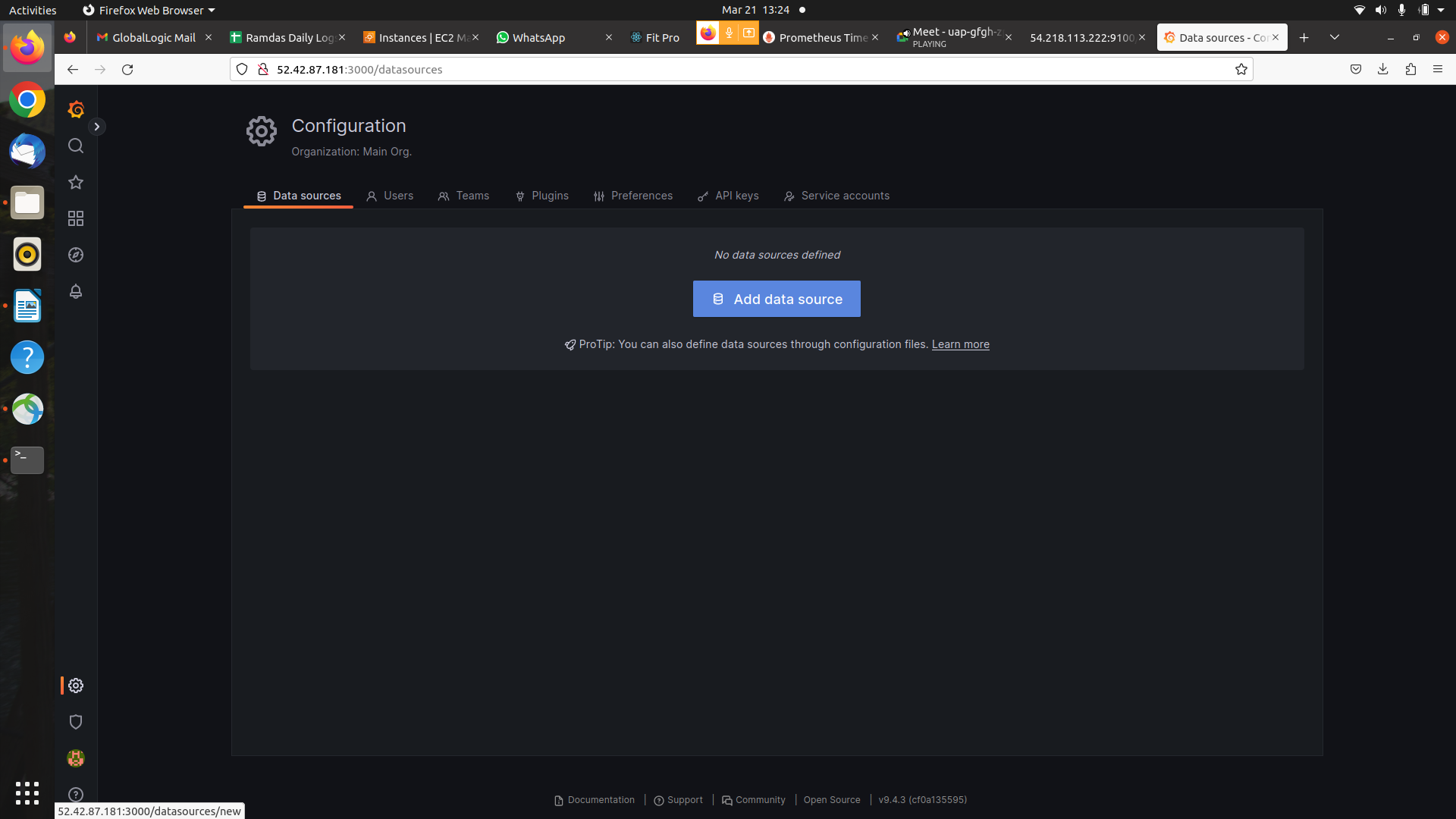


* Click skip if it ask to change password

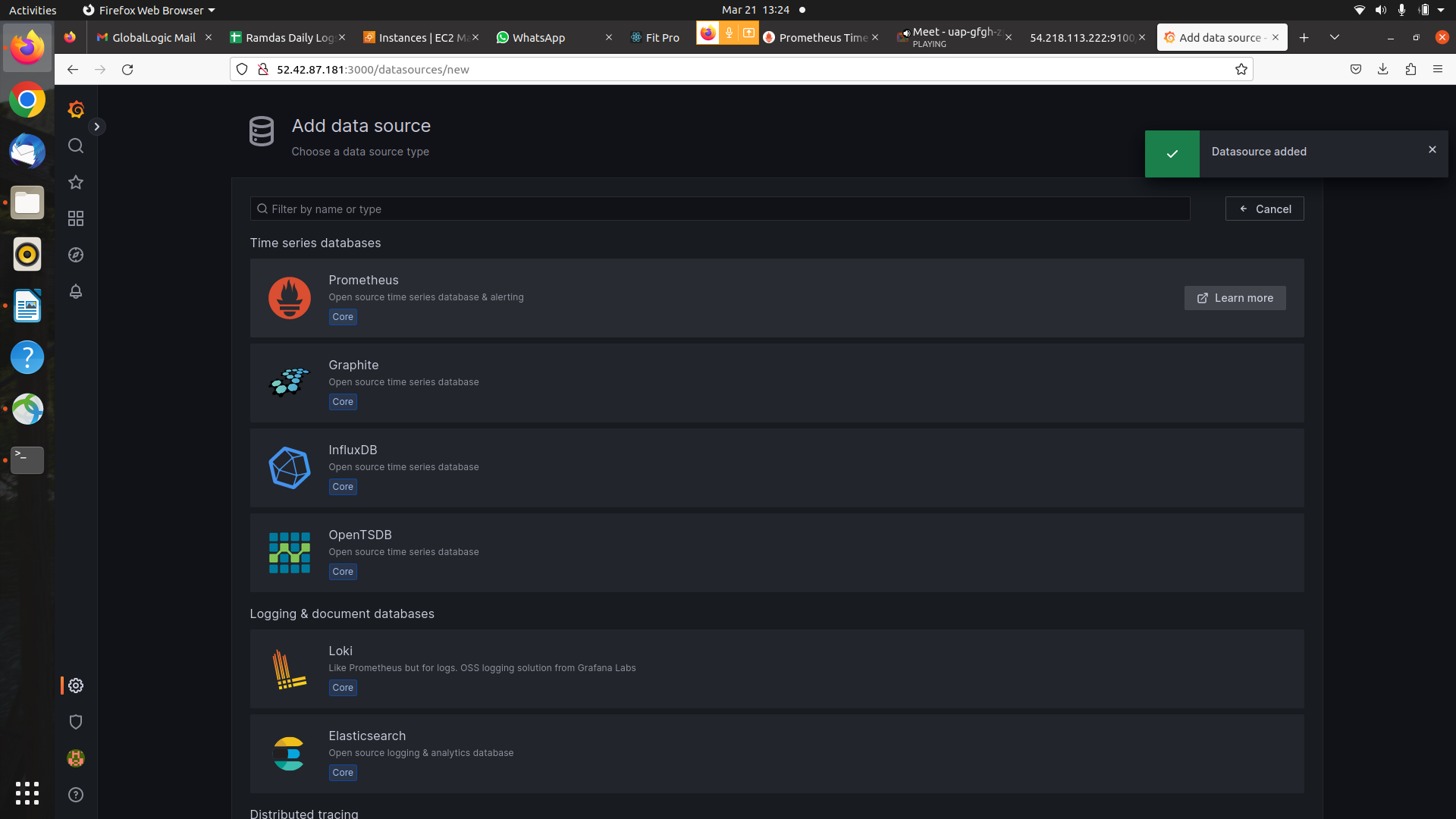
* You will see Grafana dashboard & Data Souces



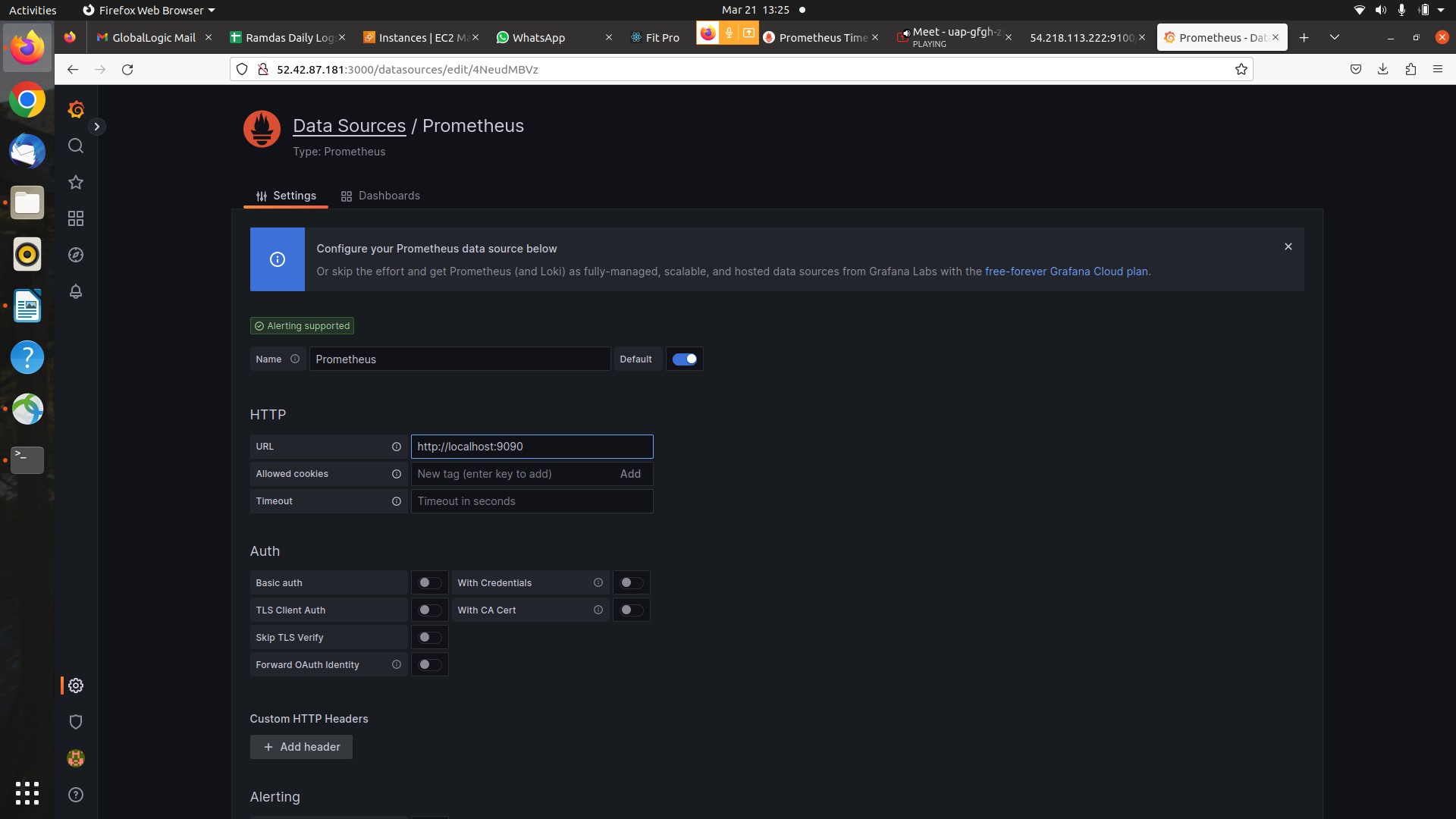
* You will see Add Data Source



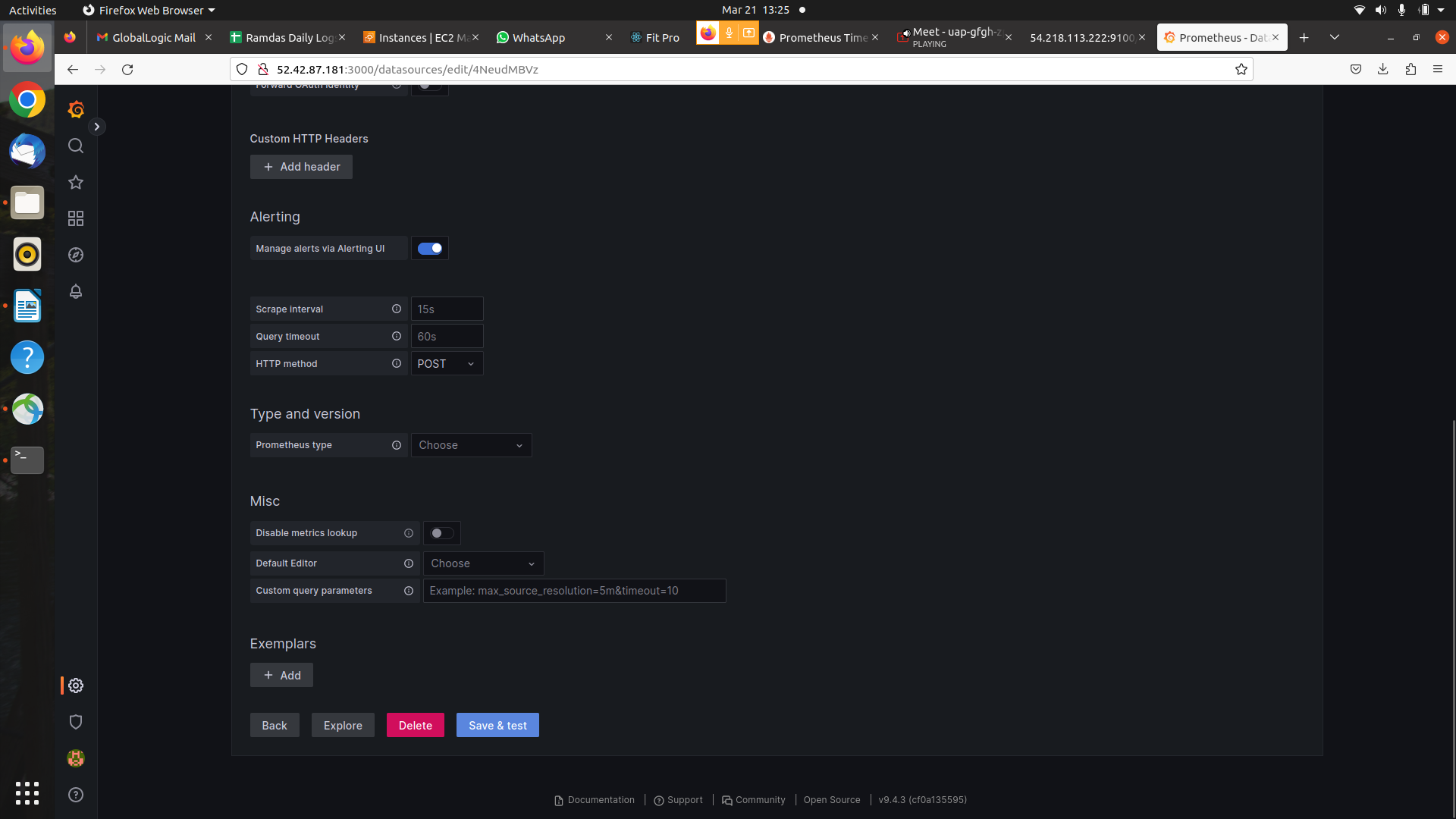
* You can select Prometheus Data Source and Click it.



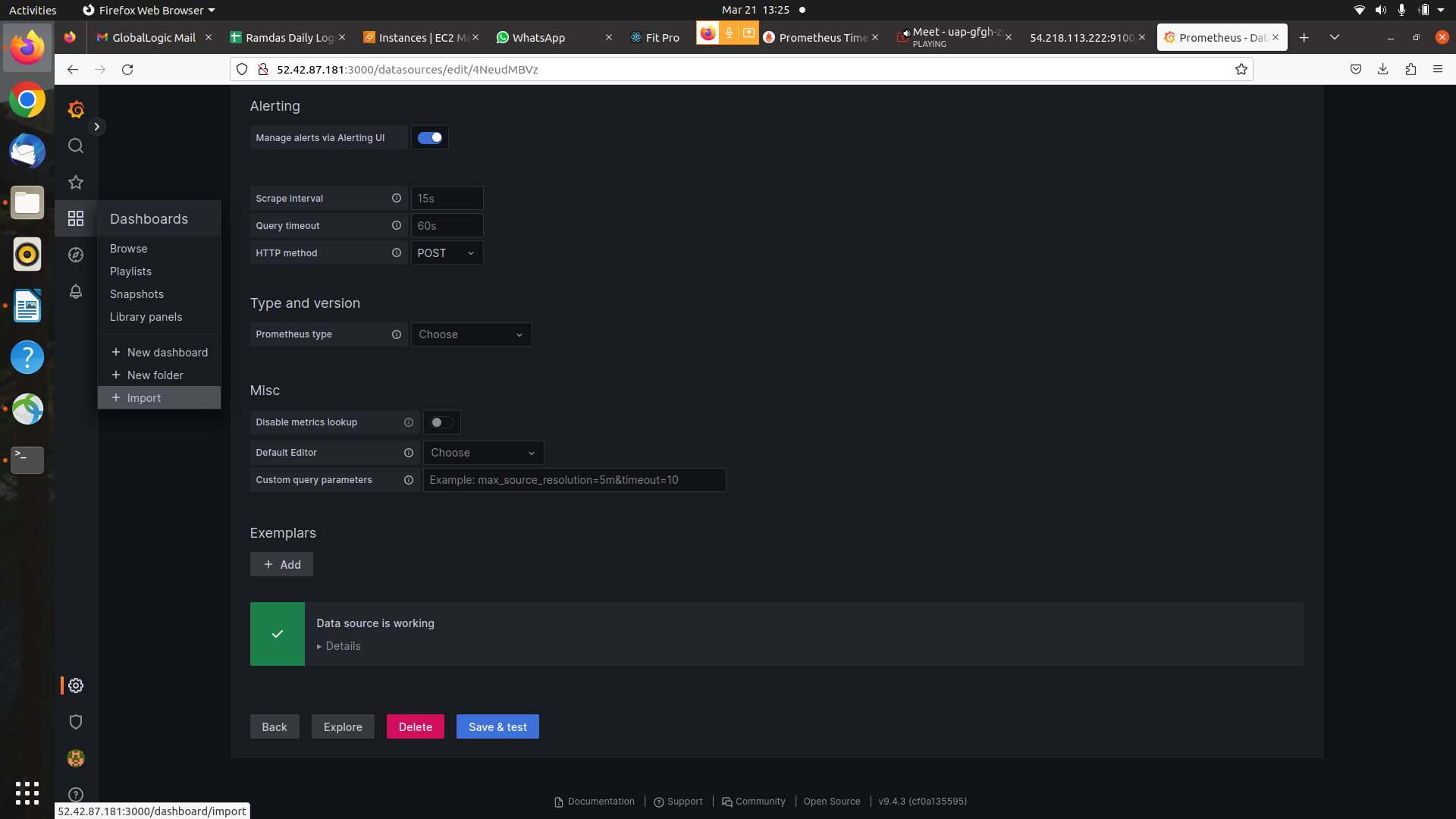
* after change Configuration Add url i.e.[http://localhost:9090](http://localhost:9090/)



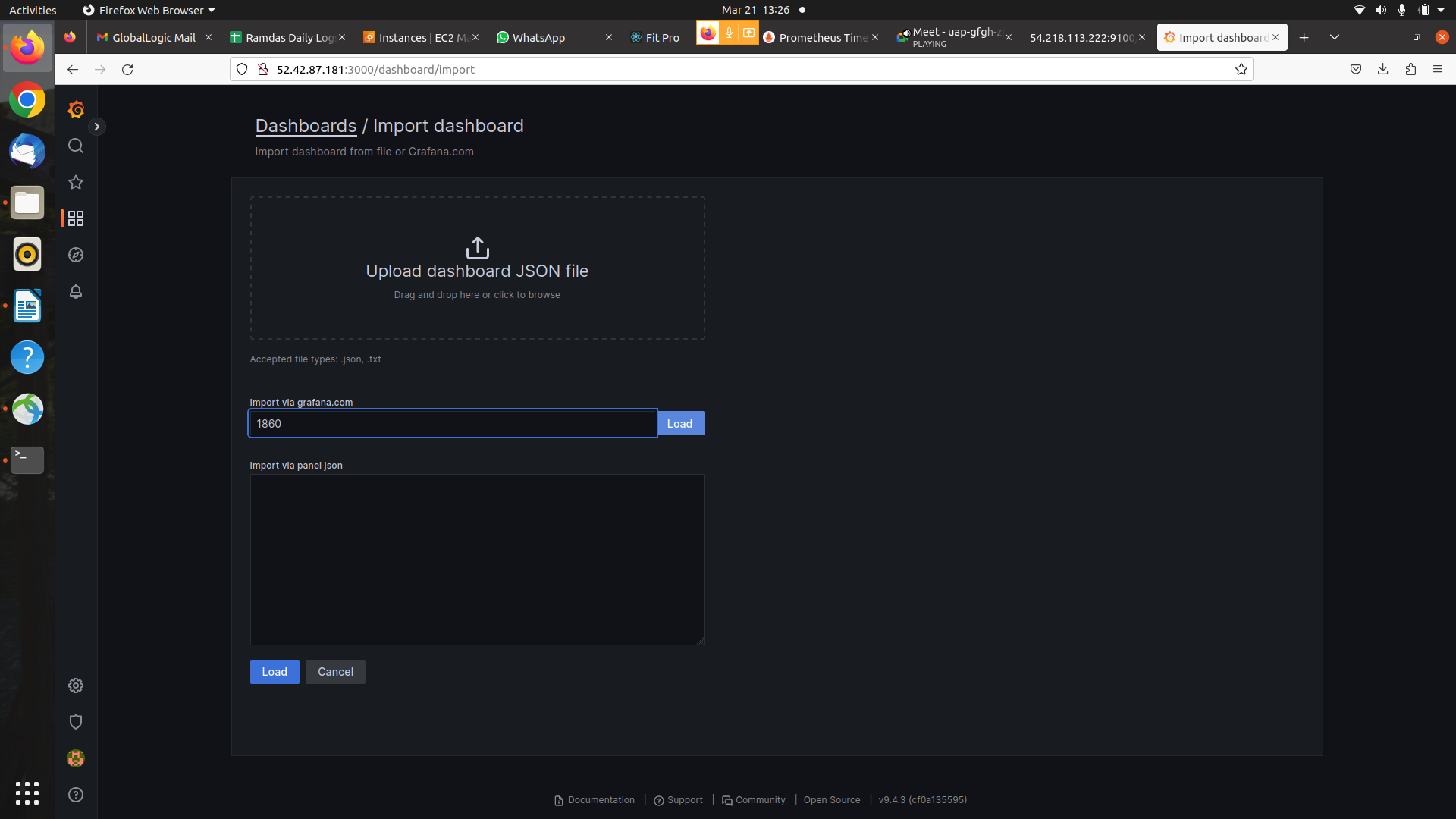
* Then click save & Test.



* You will see DialogBox DataSource Added and its working.



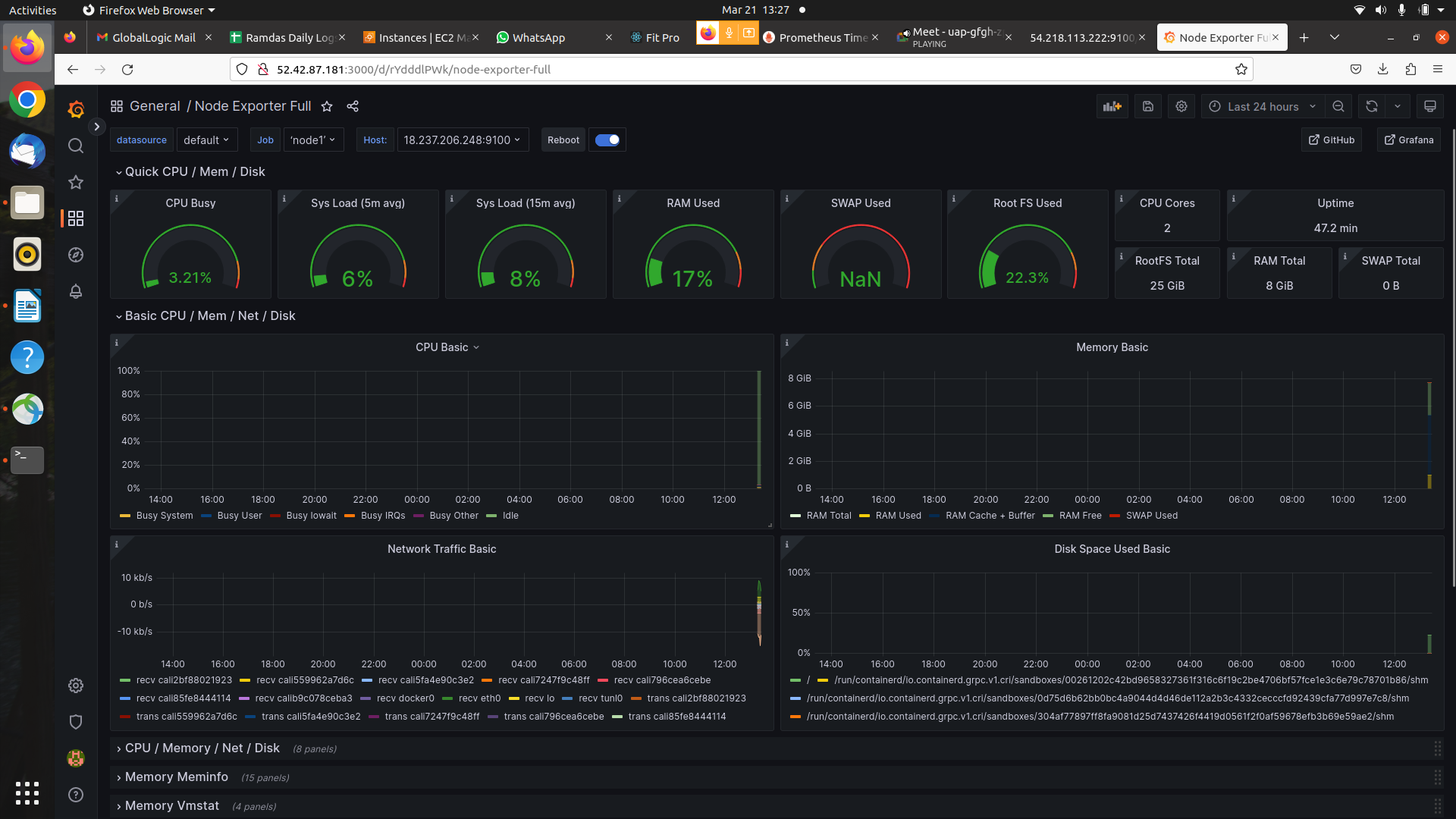
* you will see Dashobards /import dashboard add Grafana ID to import Prometheus 1860 click Load.



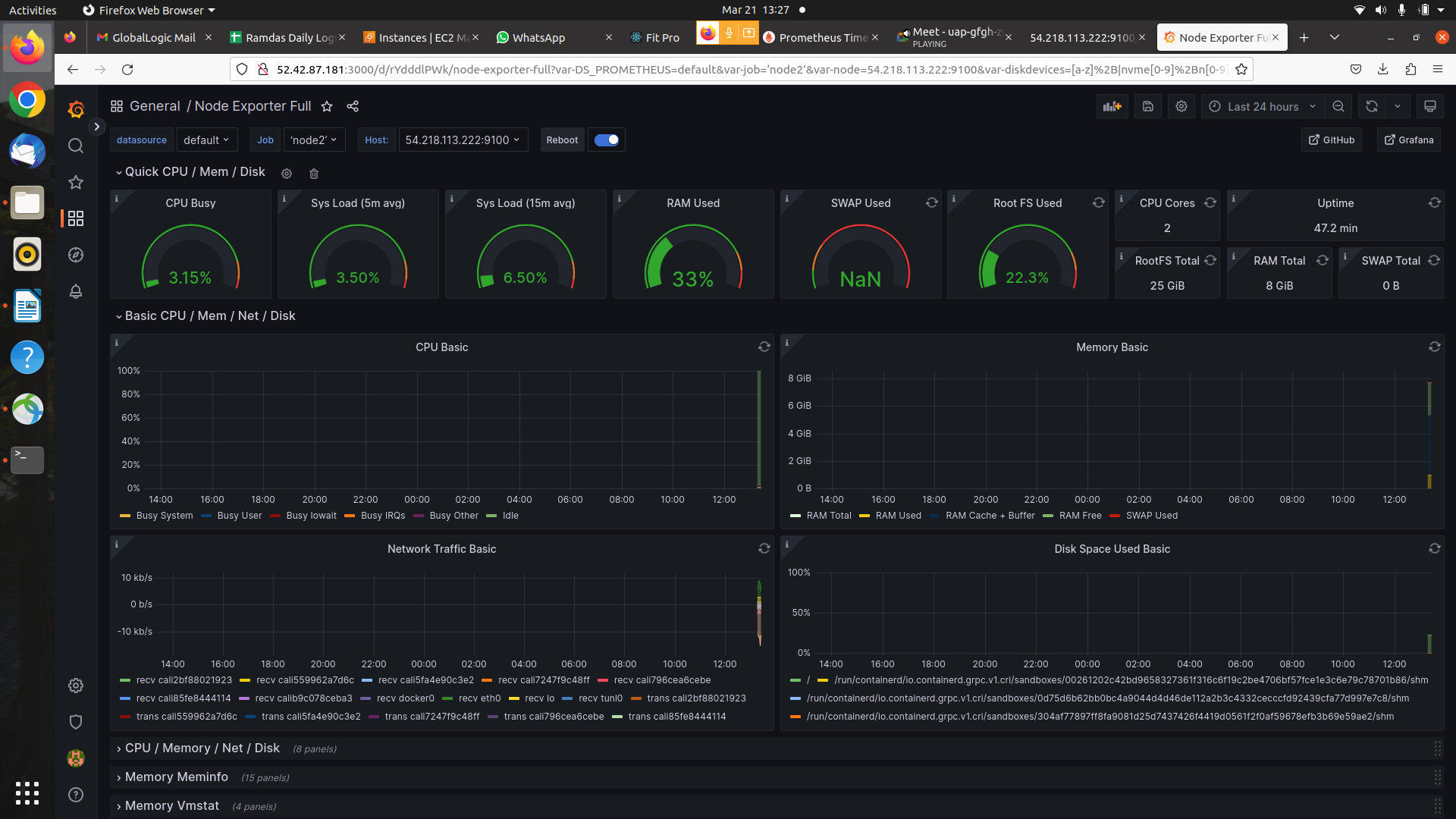
* you will see Dashboards /import dashboard select Data Source.



* you will see Node1 Dashboard Output.



* You will see Node2 Dashboard Output.



* You can see node1 and node2 Dashboard Output for Time series last 5 min.

