

**Vivekanand Education Society's Institute of Technology**  
**Department of Computer Engineering**



# **SERVER MONITORING USING PROMETHEUS AND GRAFANA**

**SUBJECT:**  
CLOUD COMPUTING

**PROJECT BY:**  
SAURABH FEGADE (D17C - 14)  
RAJAT GUPTA (D17C - 16)  
JAYA HINDUJA (D17C - 19)

**DATE:**  
12/04/2017

## **TABLE OF CONTENTS:**

1. Abstract	1
2. Introduction	1
3. Implementation Details	3
4. Result	8
5. Screenshots	8
6. References	13

## 1. ABSTRACT

Server Monitoring is a process to monitor server's system resources like CPU Usage, Memory Consumption, I/O, Network, Disk Usage, Process etc. Server Monitoring helps understanding server's system resource usage which can help you better your capacity planning and provide a better end-user experience.

Your application's health depends on a large part on the health of the underlying server. Server Monitoring ensures that your server machine is capable of hosting your applications. Server Monitoring provides you with data relating to your operating system and when used in conjunction with other monitoring data from the application you get a true glimpse into the working of your system.

## 2. INTRODUCTION

Prometheus is an open source monitoring system and time series database. It addresses many aspects of monitoring such as the generation and collection of metrics, graphing the resulting data on dashboards, and alerting on anomalies. To achieve this, it offers a variety of components that are run separately but used in combination.

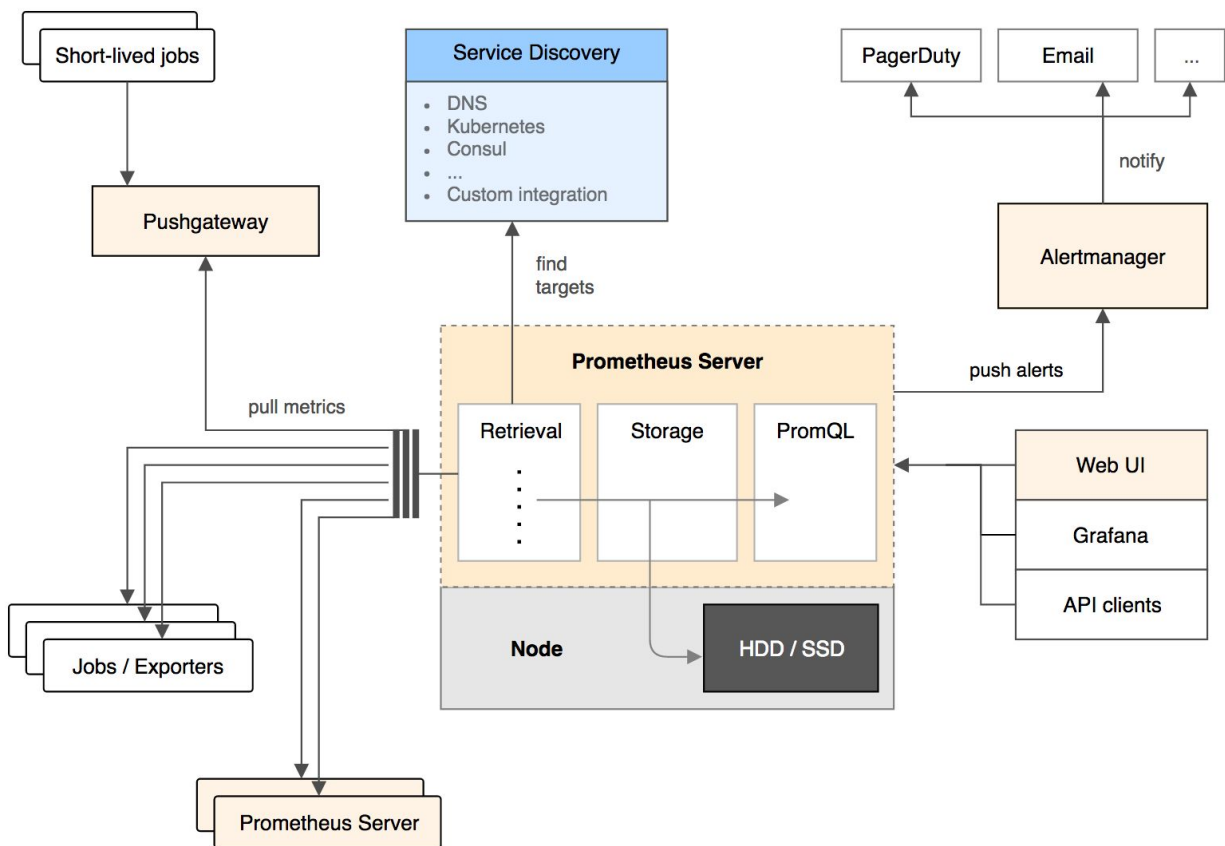
Docker provides a way for you to encapsulate server processes using Linux containers (or other encapsulation technologies) so that they are more easily managed and isolated from each other. Each Docker container is a local instance of a Docker image. You can think of a Docker image as a complete Linux installation. In this tutorial, we will learn how to install three key components for using Prometheus on Docker. These are:

- A Prometheus server to collect metrics and query them
- A Node Exporter to export system metrics in a Prometheus-compatible format
- Grafana, a web-based graphical dashboard builder that supports Prometheus among other backends

There are many more components in the Prometheus ecosystem, but these three provide a good starting point for using Prometheus.

Prometheus scrapes metrics from instrumented jobs, either directly or via an intermediary push gateway for short-lived jobs. It stores all scraped samples locally and runs rules over this data to either record new time series from existing data or

generate alerts. Grafana or other API consumers can be used to visualize the collected data.



**Overall architecture of Prometheus and some of its ecosystem components**

Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. In a world of microservices, its support for multi-dimensional data collection and querying is a particular strength.

Prometheus is designed for reliability, to be the system you go to during an outage to allow you to quickly diagnose problems. Each Prometheus server is standalone, not depending on network storage or other remote services. You can rely on it when other parts of your infrastructure are broken, and you do not have to set up complex infrastructure to use it.

Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data driven culture.

### 3. IMPLEMENTATION DETAILS

#### Step 1: What do you need?

To follow this tutorial you will need:

- an Ubuntu 16.04 droplet on DigitalOcean;
- a non-root user with sudo access; and
- Docker installed on the server droplet.

#### Step 2: Initial Server Setup. Creating the non-root user and giving it sudo access

Login to your server using the following command:

```
$ ssh root@45.55.147.132
```

If this is the first time you are logging in, you will be prompted to change the password.

Create a new user using the following command:

```
$ adduser rajat
```

As root, run this command to add your new user to the *sudo* group

```
$ gpasswd -a rajat sudo
```

If you do not already have an SSH key pair, which consists of a public and private key, you need to generate one.

To generate a new key pair, enter the following command at the terminal of your local machine

```
local$ ssh-keygen
```

After generating an SSH key pair, you will want to copy your public key to your new server.

Run the ssh-copy-id script by specifying the user and IP address of the server that you want to install the key on, like this:

```
local$ ssh-copy-id rajat@45.55.147.132
```

Then ssh into the non-root user: `$ ssh rajat@45.55.147.132`

### Step 3: Installing Docker

The quickest way to install Docker is to download and install their installation script.

```
$ sudo wget -qO- https://get.docker.com/ | sh
```

Working with Docker is a pain if your user is not configured correctly, so add your user to the docker group with the following command.

```
$ sudo usermod -aG docker rajat
```

### Step 4: Installing Prometheus

The Prometheus server is the central piece of the Prometheus ecosystem and is responsible for collecting and storing metrics as well as processing expression queries and generating alerts.

Docker container images for all Prometheus components are hosted under the [prom](#) organization on Docker Hub.

In this tutorial, we will choose to pass in a configuration file from the host system.

First, create a minimal Prometheus configuration file on the host filesystem at `~/prometheus.yml`:

```
$ nano ~/prometheus.yml
```

Add the following contents to the file:

```
# A scrape configuration scraping a Node Exporter and
the Prometheus server
# itself.
```

```
scrape_configs:
```

```
  # Scrape Prometheus itself every 5 seconds.
```

```
  - job_name: 'prometheus'
```

```
    scrape_interval: 5s
```

```
    static_configs:
```

```
      - targets: ['localhost:9090']
```

```
  # Scrape the Node Exporter every 5 seconds.
```

```
  - job_name: 'node'
```

```
    scrape_interval: 5s
```

```
static_configs:
  - targets: ['45.55.147.132:9100']
```

Start the Prometheus Docker container with the external configuration file:

```
$ sudo docker run -d -p 9090:9090 -v
~/prometheus.yml:/etc/prometheus/prometheus.yml
prom/prometheus
-config.file=/etc/prometheus/prometheus.yml
-storage.local.path=/prometheus
-storage.local.memory-chunks=10000
```

The first time you run this command, it will pull the Docker image from the Docker Hub. This command is quite long and contains many command line options. Let's take a look at it in more detail:

- The `-d` option starts the Prometheus container in detached mode, meaning that the container will be started in the background and will not be terminated by pressing CTRL+C.
- The `-p 9090:9090` option exposes Prometheus's web port (9090) and makes it reachable via the external IP address of the host system.
- The `-v [...]` option mounts the `prometheus.yml` configuration file from the host filesystem into the location within the container where Prometheus expects it (`/etc/prometheus/prometheus.yml`).
- The `-config.file` option is set accordingly to the location of the Prometheus configuration file *within in the container*.
- The `-storage.local.path` option configures the metrics storage location *within the container*.
- Finally, the `-storage.local.memory-chunks` option adjusts Prometheus's memory usage to the host system's very small amount of RAM (only 512MB) and small number of stored time series in this tutorial (just under 1000). It instructs Prometheus to keep only 10000 sample chunks in memory (roughly 10 chunks per series), instead of the default of 1048576. This is a value you will definitely need to tune when running Prometheus on a machine with more RAM and when storing more time series.

## Step 5: Setting up Node Exporter

The Node Exporter is a server that exposes Prometheus metrics about the host machine (node) it is running on. This includes metrics about the machine's filesystems, networking devices, processor usage, memory usage, and more.

To start the Node Exporter on port 9100 using Docker:

```
$ sudo docker run -d -p 9100:9100 -v
"/proc:/host/proc" -v "/sys:/host/sys" -v ":/rootfs"
--net="host" prom/node-exporter -collector.procfs
/host/proc -collector.sysfs /host/proc
-collector.filesystem.ignored-mount-points
"^/(sys|proc|dev|host|etc) ($|/)"
```

The following Docker and Node Exporter flags are used to provide a reasonable approximation for the *host* metrics:

- On Linux, the Node Exporter gathers most of its metrics from the `/proc` and `/sys` filesystems. These filesystems are mounted from the host into the container underneath a `/host` directory, using Docker's `-v` flag.
- Via the Node Exporter's `-collector.procfs` and `-collector.sysfs` flags, we instruct the Node Exporter to look for the `/proc` and `/sys` filesystems in a non-standard location.
- To report host filesystem metrics, we also mount the entire root (`/`) filesystem into the container (at `/rootfs`), again using Docker's `-v` flag.
- Use Node Exporter's `-collector.filesystem.ignored-mount-points` flag to ignore any other filesystems within the container that do not belong to the host system. This option takes a regular expression of mount points to exclude from the reported metrics.
- Using the `--net=host` Docker flag, we place the container into the same network stack as the host, so that reading from files such as `/proc/net/dev` will yield the same results as on the host (reading from the `/proc` filesystem mounted in from the host is not sufficient).



## Step 6: Setting up Grafana

Finally, we will set up Grafana. Grafana is a graphical dashboard builder that supports Prometheus as a backend to query for data to graph.

Grafana stores its dashboard metadata (such as which dashboards exist and what graphs they should show) in a configurable SQL-based database. Grafana supports using a local file-backed SQLite3 database as well as external database servers such as MySQL or PostgreSQL for this. In this tutorial, we will use a SQLite3 database backed by a Docker data volume. Launch Grafana as a Docker container with an administrator password of your choosing:

```
$ sudo docker run -d -p 3000:3000 -e
"GF_SECURITY_ADMIN_PASSWORD=admin" -v
~/grafana_db:/var/lib/grafana grafana/grafana
```

This will download the Grafana Docker image from the Docker Hub and create a new Docker volume placed at `~/grafana_db` on the host system and at `/var/lib/grafana` in the container filesystem.

- In the container, Grafana will then automatically create and initialize its SQLite3 database at `/var/lib/grafana/grafana.db`.
- The `-e` flag allows passing environment variables to the process launched inside the Docker container.
- Here, we use it to set the `GF_SECURITY_ADMIN_PASSWORD` environment variable to the desired dashboard administrator password, overriding the default password of admin.

## Step 7: Checking System Status

One can check all running docker containers using the following command:

```
$ sudo docker ps
```

You should now be able to reach your Prometheus server at `http://45.55.147.132:9090/`. Verify that it is collecting metrics about itself by heading to `http://45.55.147.132:9090/status`

To verify that Grafana is running correctly, head to

```
http://45.55.147.132:3000/.
```

The administrator username is admin and the password is the one you chose when starting the Docker container previously.

## 4. RESULTS

We have set up a Prometheus server, a Node Exporter, and Grafana — all using Docker. Even though these are currently all running on the same machine, this is only for demonstration purposes. In production setups, one would usually run the Node Exporter on every monitored machine, multiple Prometheus servers (as needed by the organization), as well as a single Grafana server to graph the data from these servers.

## 5. SCREENSHOTS

```
rajat@ubuntu-2gb-nyc3-01:~$ sudo docker run -d -p 9090:9090 -v ~/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus --config.file=/etc/prometheus/
prometheus.yml --storage.local.path=/prometheus --storage.local.memory-chunks=10000
[sudo] password for rajat:
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
557a0c95bfcd: Pull complete
a3ed95caeb02: Pull complete
caf4d0cf9832: Pull complete
ee054001e2db: Pull complete
b95bf6c4c81b: Pull complete
86503a6ba368: Pull complete
ff27c7b0b50e: Pull complete
534e30a17a42: Pull complete
475d41733562: Pull complete
Digest: sha256:06d9e878fba272c5f55ebc2863694b5076ff01f810c6d187767cd53891ba453
Status: Downloaded newer image for prom/prometheus:latest
53eca4ef2c216a1979654e917aab982a199cb887182b0e57eaeafcb185661d14
rajat@ubuntu-2gb-nyc3-01:~$ sudo docker run -d -p 9100:9100 -v "/proc:/host/proc" -v "/sys:/host/sys" -v "/:/rootfs" --net="host" prom/node-exporter --collec
tor.procfs /host/proc --collector.sysfs /host/proc --collector.filesystem.ignored-mount-points '^(/sys|proc|dev|host|etc)($|/)'
Unable to find image 'prom/node-exporter:latest' locally
latest: Pulling from prom/node-exporter
df74c0cea8ad: Pull complete
a3ed95caeb02: Pull complete
ce984171bf74: Pull complete
5844de687528: Pull complete
Digest: sha256:ffc3e0adf58771cb2097aa0b074a6fe68b4925ba75c2e1c41f41ae656eebee11
Status: Downloaded newer image for prom/node-exporter:latest
4dbfb378c8b8154be11dbbfa3a429979bbf496408c2ef25f30acf6c40e4fa433
rajat@ubuntu-2gb-nyc3-01:~$ sudo docker run -d -p 3000:3000 -e "GF_SECURITY_ADMIN_PASSWORD=admin" -v ~/grafana_db:/var/lib/grafana grafana/grafana
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
6d827a3ef358: Pull complete
a3ed95caeb02: Pull complete
6c68d80aeafa: Pull complete
64d300d2f19d: Pull complete
Digest: sha256:5f5faa0c2919deb128f8ea42d71fc249fb8d352cf8a346110631956dd56c7a03
Status: Downloaded newer image for grafana/grafana:latest
8128f94e85fb30185a8edfb3ef7d096494e8a064ec305c1b9d0f9cd287b73e60
rajat@ubuntu-2gb-nyc3-01:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS              PORTS                               NAMES
8128f94e85fb        grafana/grafana    "/run.sh"                10 seconds ago    Up 9 seconds       0.0.0.0:3000->3000/tcp              hungry_booth
4dbfb378c8b8        prom/node-exporter "/bin/node_exporte..." 56 seconds ago    Up 55 seconds      9100->9100/tcp                      loving_poitras
53eca4ef2c21        prom/prometheus    "/bin/prometheus -..." About a minute ago Up About a minute  9090->9090/tcp                      peaceful_fermat
```

**Docker commands to install Prometheus, Node Exporter and Grafana and then view all active containers**



ubuntu-2gb-nyc3-01

2 GB Memory / 40 GB Disk / NYC3 - Ubuntu 16.04.2 x64



ipv4: 45.55.147.132

ipv6: [Enable now](#)

Private IP: [Enable now](#)

Floating IP: [Enable now](#)

[Console](#):

## Graphs

[Access](#)

[Power](#)

[Volumes](#) NEW!

[Resize](#)

[Networking](#)

[Backups](#)

[Snapshots](#)

[Kernel](#)

[History](#)

[Destroy](#)

[Tags](#)

Last updated just now. [Create alert policy.](#)

6 hours

### CPU ?



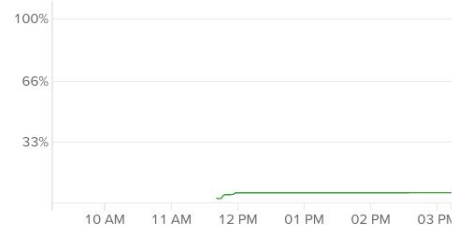
### Memory ?



### Disk I/O ?



### Disk usage ?



### Bandwidth public ?



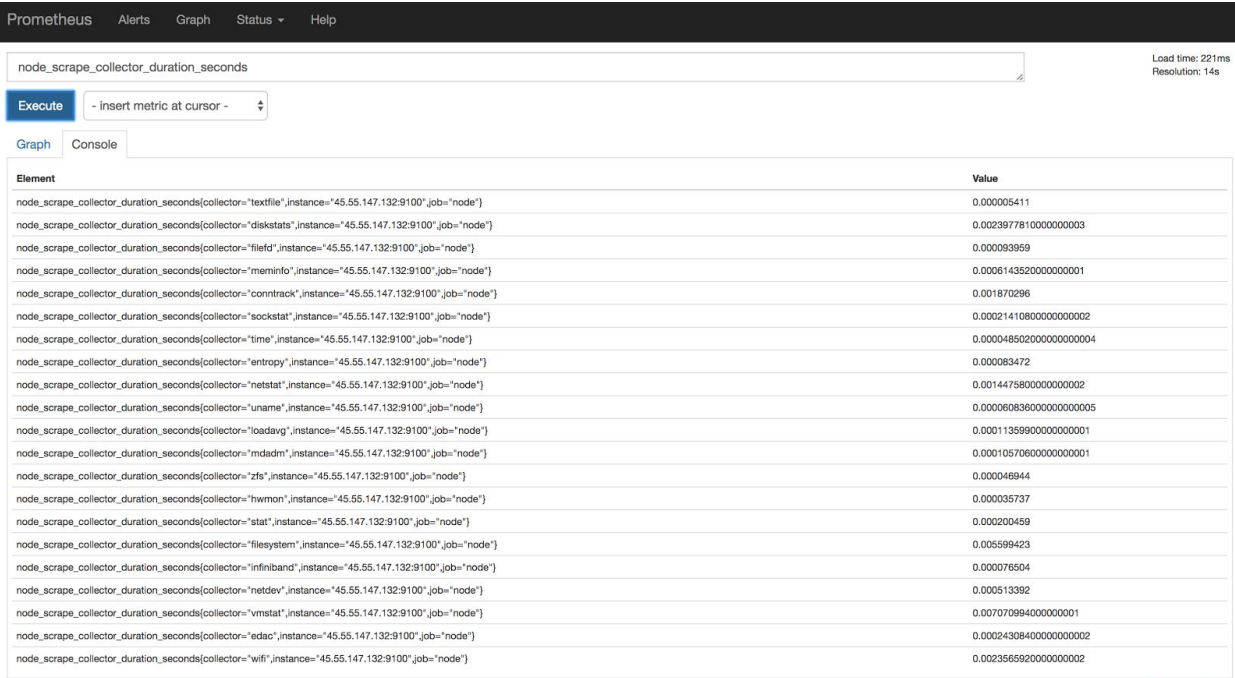
### Top processes ?

Last updated: just now

CPU Memory

prometheus	5.37%
dockerd	2.71%
docker-containe	1.49%
grafana-server	1.38%
node_exporter	1.05%
snapt	1.01%
do-agent	0.68%

Active view of the dashboard for the DigitalOcean droplet



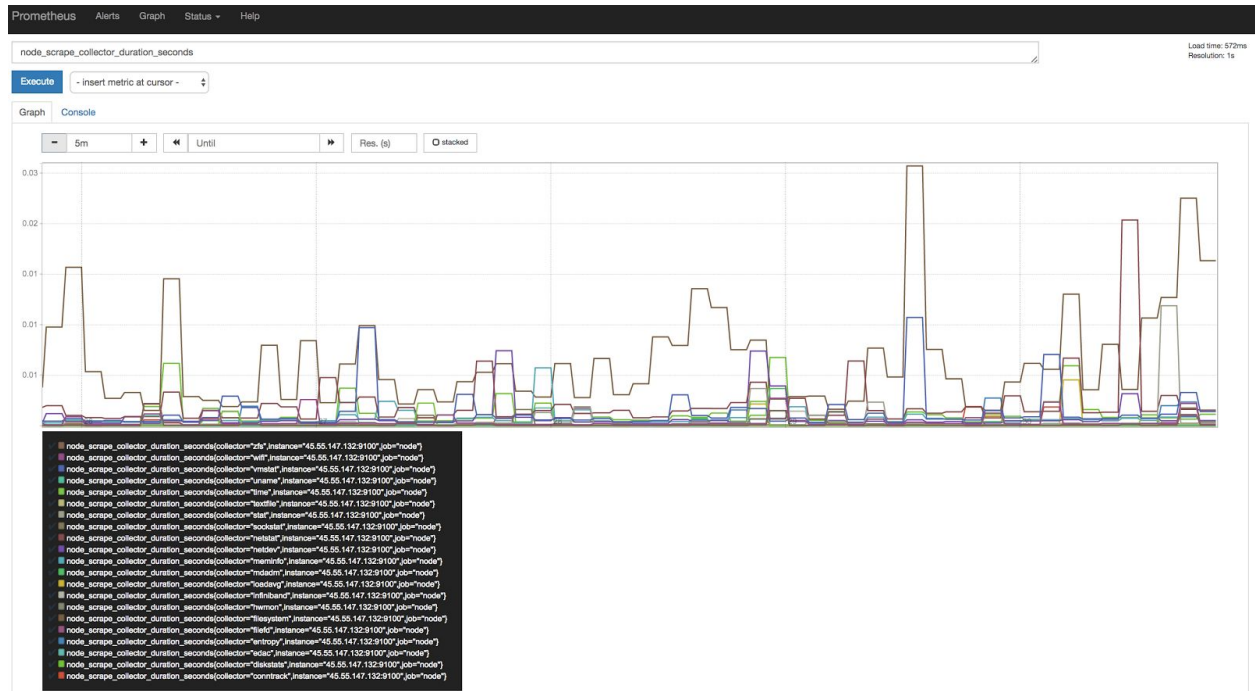
Console view of an attribute in Prometheus


Prometheus Alerts Graph Status Help

### Targets

node				
Endpoint	State	Labels	Last Scrape	Error
http://45.55.147.132:9100/metrics	UP	Instance="45.55.147.132:9100"	4.302s ago	
prometheus				
Endpoint	State	Labels	Last Scrape	Error
http://localhost:9090/metrics	UP	Instance="localhost:9090"	2.303s ago	

Status of Node Exporter and Prometheus Server



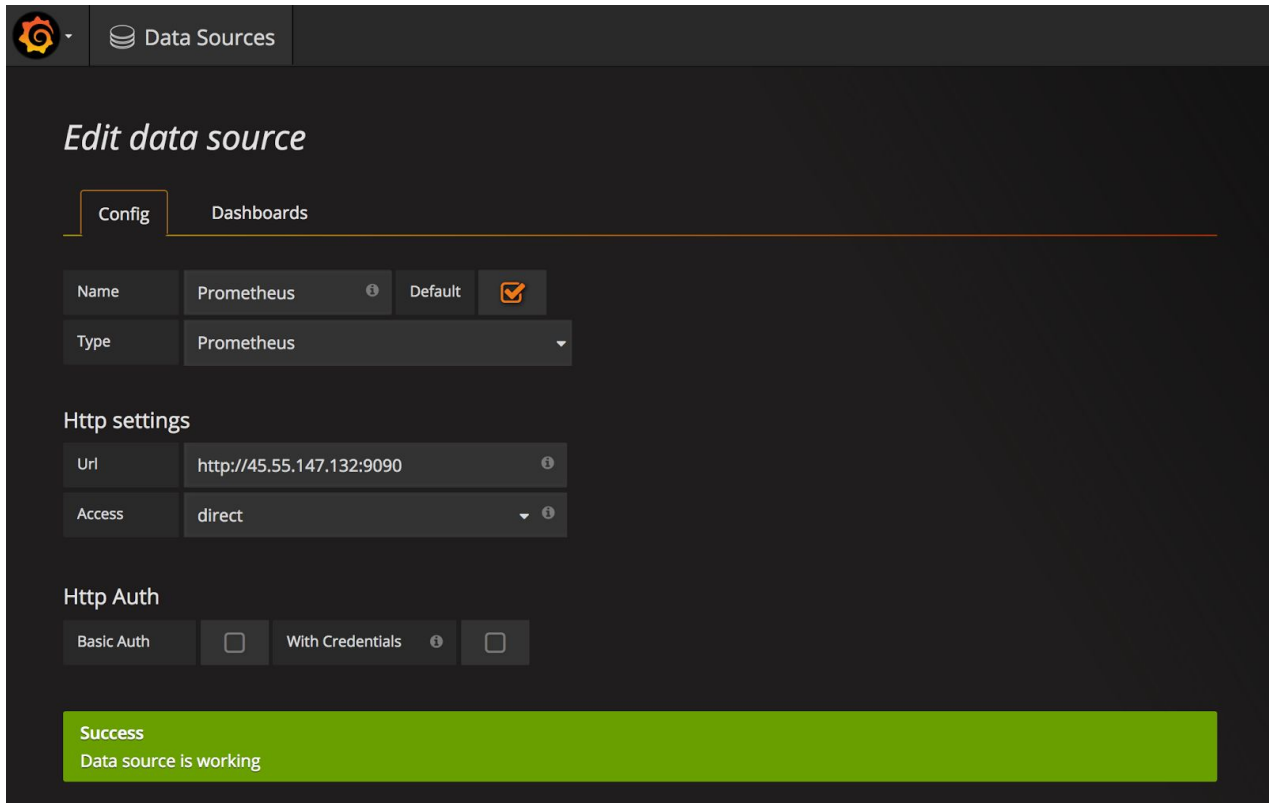


Log in

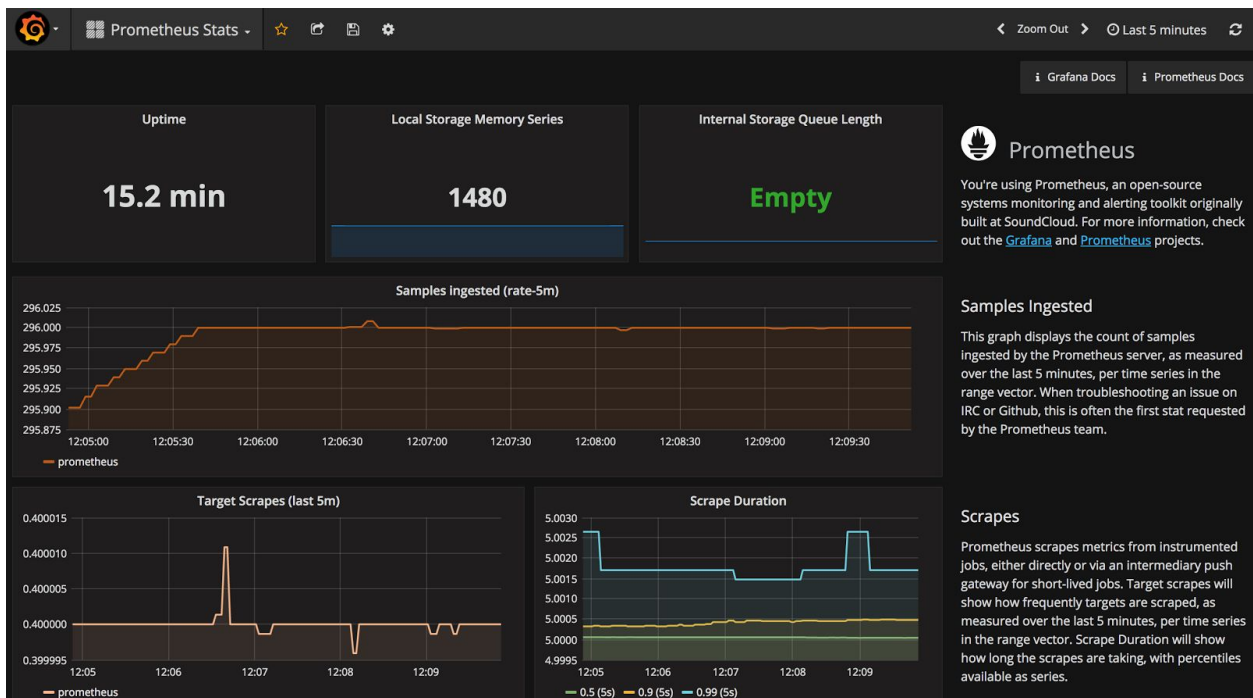
Sign up

User	admin
Password	.....

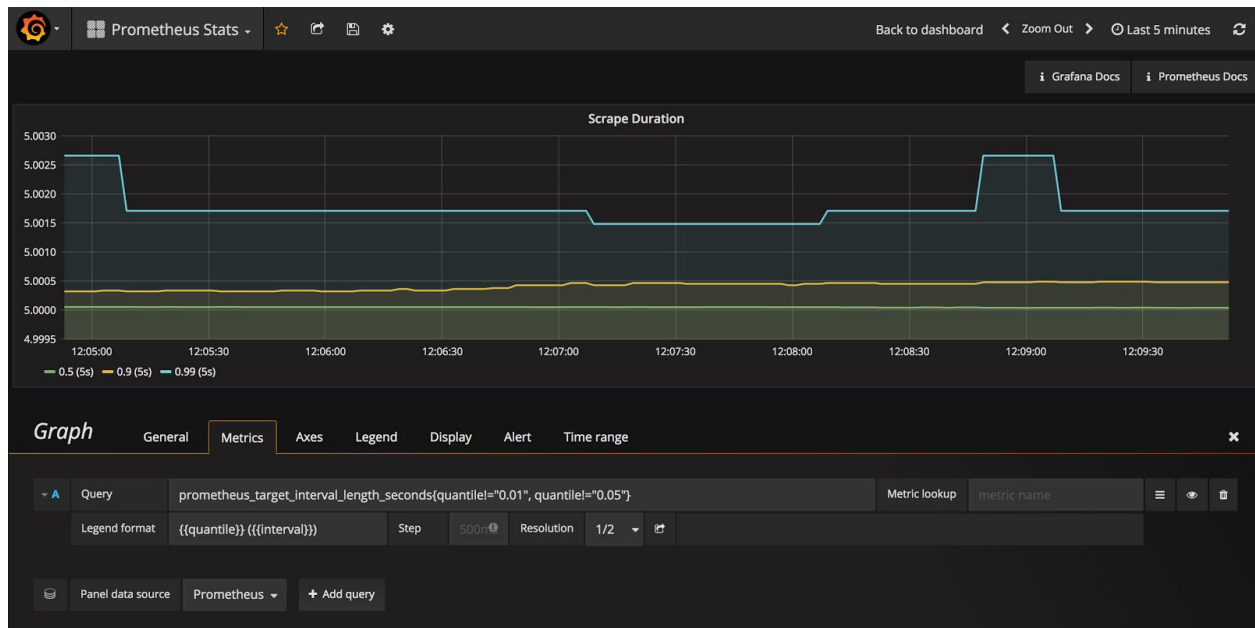
Log in



## Setting up of Data Source in Grafana



## Dashboard view after setting up the Data Source



**Editing the attributes in dashboard in Grafana**

## 6. REFERENCES

- [1]<https://www.digitalocean.com/community/tutorials/how-to-install-prometheus-using-docker-on-ubuntu-14-04>
- [2]<https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-14-04>
- [3]<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-14-04>
- [4]<https://prometheus.io/>
- [5]<https://grafana.com/>