Setup guide: Monitoring FIO QoS parameters using Prometheus, InfluxDB and Grafana

Sumit Shatwara

Contents

[Revision History 2](#_Toc8808511)

[Problem Statement 3](#_Toc8808512)

[Requirements 3](#_Toc8808513)

[Technical Overview 3](#_Toc8808514)

[Setup Procedure 4](#_Toc8808515)

[Steps Involved to configure everything and get you started 5](#_Toc8808516)

[Steps involved to configure InfluxDB as a primary storage for Prometheus metrics 10](#_Toc8808517)

## Revision History

|  |  |  |
| --- | --- | --- |
| Revision | Date | Remarks |
| v0.1 | May 10, 2019 | Initial Draft |
| v0.2 | May 15, 2019 | Added the details of InfluxDB (primary database for Prometheus) |
|  |  |  |
|  |  |  |
|  |  |  |

## Problem Statement

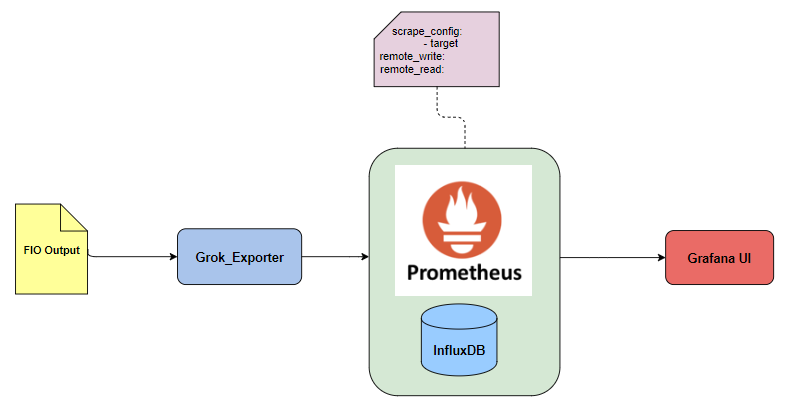
Implement performance-monitoring process of Flexible IO tester (FIO) at client servers using Prometheus and visualize the time-series metrics of QoS parameters (IOPS, Latency and Bandwidth) from FIO output generated on real-time basis, using Grafana dashboard.

## Requirements

* One/Two nodes with CentOS 7.6 or Ubuntu 18.04.
* FIO 3.13 or later
* Prometheus 2.9.1 or later
* Grafana 6.1.6 or later
* Grok Exporter v0.2.7 or later
* InfluxDB 1.7.4 or later

## Technical Overview

Flexible IO tester (FIO) is an open-source synthetic benchmark tool. FIO can generate various IO workloads: sequential reads or random writes, synchronous or asynchronous, all based on the options provided by the user. FIO is the easiest and versatile tool to quickly perform IO performance tests on storage system, and allows you to simulate different types of IO loads.



Prometheus is an open source monitoring solution that stores all its data in a time series database. Prometheus has a multi-dimensional data-model and a powerful query language that is used to generate reports of the resources being monitored. Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. [Grok](https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html) is a tool to parse crappy unstructured log data into something structured and queryable to Prometheus server.

FIO tool generates an unstructured output that is not recognized by Prometheus. That is where Grok-exporter appears, which will the help fetching the required QoS parameters (IOPS, Latency and Bandwidth), and transform them into metric name and key/value pairs. Once the data is queryable to Prometheus and it can be visualized with the help of Grafana dashboard.

However, Prometheus storage engine is designed for keeping mainly the short-term data. This gap can be filled in by InfluxDB time-series database having it to store the long-term data and perform monitoring from Grafana dashboard utilizing InfluxDB as a data source.

## Setup Procedure

Follow below steps for a complete guide on how to set this up.

**Step 1**: Install FIO using yum (EPEL repository must be installed beforehand).

**Step 2**: Install Prometheus Server, InfluxDB and Grafana:

Use these links for how to install Prometheus, InfluxDB and Grafana. Also, enable the InfluxDB http authentication, which is mentioned as an optional step in below link.

[Install Prometheus Server on CentOS 7](https://computingforgeeks.com/install-prometheus-server-on-centos-7/) and [Install Grafana and InfluxDB on CentOS 7](https://computingforgeeks.com/install-grafana-and-influxdb-on-centos-7/).

**Step 3**: Download grok\_exporter-$ARCH.zip for your operating system from the [releases](https://github.com/fstab/grok_exporter/releases) page, extract the archive, cd grok\_exporter-$ARCH

**Step 4**: Configure Grok Exporter ([Gihub link](https://github.com/fstab/grok_exporter)).

**Step 5**: Configure Prometheus server with Scrap jobs

**Step 6**: Configure Prometheus as a data source in Grafana

**Step 7**: Add Dashboards with PromQL queries to Grafana

**Step 8**: Start visualizing FIO output metrics on Grafana

**Step 9**: Configure Prometheus with Remote Read and Write API natively to store metrics in InfluxDB

**Step 10**: Configure InfluxDB as a data source in Grafana

**Step 11**: Add Dashboards with InfluxSQL queries to Grafana

**Step 12**: Start visualizing FIO output metrics on Grafana

## Steps Involved to configure everything and get you started

1. Add Prometheus system user

sudo groupadd --system prometheus

sudo useradd -s /sbin/nologin --system -g prometheus prometheus

We added a system user called prometheus whose default group is prometheus. This user account will be used to run nod exporter service. It is safe since it doesn’t have access to the interactive shell and home directory

1. Add grok-exporter job to Prometheus config file (/etc/prometheus/prometheus.yml):

# Grok-Exporter

- job\_name: grok\_exporter

static\_configs:

- targets: ['<IP of Machine>:9144']

labels:

alias: grok\_fio1

Mention the server IP where the grok-exporter instance is running.

1. Run the FIO command to simulate required type of workload. Sample FIO command to simulate random write workload is mentioned as follows:

$ fio --name=randwrite --ioengine=libaio --iodepth=1 --rw=randwrite --bs=4k --direct=0 --size=800M --numjobs=1 --runtime=1 --status-interval=1 --group\_reporting --time\_based

Output:

randwrite: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1

fio-3.1

Starting 1 process

Jobs: 1 (f=1)

randwrite: (groupid=0, jobs=1): err= 0: pid=4896: Tue May 7 17:20:26 2019

write: IOPS=15.3k, BW=58.8MiB/s (61.6MB/s)(52.9MiB/901msec)

slat (usec): min=3, max=19286, avg=45.04, stdev=462.26

clat (nsec): min=863, max=11404k, avg=8473.32, stdev=110288.04

lat (usec): min=4, max=19293, avg=55.34, stdev=485.29

clat percentiles (usec):

……………..

……………..

……………..

From the above sample FIO output, we can target few values highlighted in the output.

1. Navigate to grok\_exporter path as mentioned in Step 3 of last section. Modify the ./example/config.yml to fetch the values of IOPS, Bandwidth or Avg Latency as below:

**Global Section**:

The global section is as follows:

global:

config\_version: 2

retention\_check\_interval: 53s

The config\_version specifies the version of the config file format. Specifying the config\_version is mandatory, it has to be included in every configuration file. The current config\_version is 2

The retention\_check\_interval is the interval at which grok\_exporter checks for expired metrics. By default, metrics don't expire so this is relevant only if retention is configured explicitly with a metric.

**Input Section**

Grok exporter currently supports two input types: file and stdin. The following section describe the stdin input type:

input:

type: stdin

This is useful if you want to pipe log data to the grok\_exporter command, for example if you want to monitor the output of FIO command, which is dumped into sample.txt text file:

tail -f sample.txt | grok\_exporter -config config.yml

**Grok Section:**

The grok section configures the available Grok patterns, and also additional patterns. An example configuration is as follows:

grok:

patterns\_dir: ./patterns

additional\_patterns:

- 'FIO\_IOPS [0-9]\*[.][0-9]k$|[0-9]\*'

**Metrics Section:**

The metrics section contains a list of metric definitions, specifying how log lines are mapped to Prometheus metrics. Four metric types are supported: Counter, Gauge, Histogram and Summary.

metrics:

- type: gauge

name: iops\_randwrite

help: FIO IOPS Random Write Gauge Metrics

match: ' write: IOPS=%{NUMBER:val1}%{GREEDYDATA:thsd}, %{GREEDYDATA}'

value: '{{if eq .thsd "k"}}{{multiply .val1 1000}}{{else}}{{.val1}}{{end}}'

labels:

parameter: 'IOPS'

cumulative: false

retention: 1s

The configuration is as follows:

* type is gauge. We require gauge because the values of IOPS, BW and Latency are fluctuating and can go up and down.
* name is the name of the metric. Metric names are described in the [Prometheus data model documentation].
* help is a comment describing the metric.
* match is the Grok expression. The sample expression will match the sixth line from the same FIO output mentioned above and it is utilizing the patterns mentioned in ./patterns/grok-patterns. See the [Grok documentation](https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html) for more info.
* labels is an optional map of name/template pairs, as described above.
* value is an conditional expression to evaluate if IOPS value contains “K” as a suffix and then it will multiple by 1000 to maintain consistency across all observed values.
* cumulative is set as false, means it will only display the last observed value. If it is set as true, it will display the sum of all the observed values.
* retention will wipe out all the records of metrics with labels (parameter: ‘IOPS’) after 1 second of last observed value. For the format of the retention value, see [How to Configure Durations](https://github.com/fstab/grok_exporter/blob/master/CONFIG.md#how-to-configure-durations)

**Server Section:**

The server section configures the HTTP(S) server for exposing the metrics:

server:

host: <IP of Machine>

port: 9144

* host can be a hostname or an IP address. If host is specified, grok\_exporter will listen on the network interface with the given address. If host is omitted, grok\_exporter will listen on all available network interfaces.
* port is the TCP port to be used. Default is 9144.

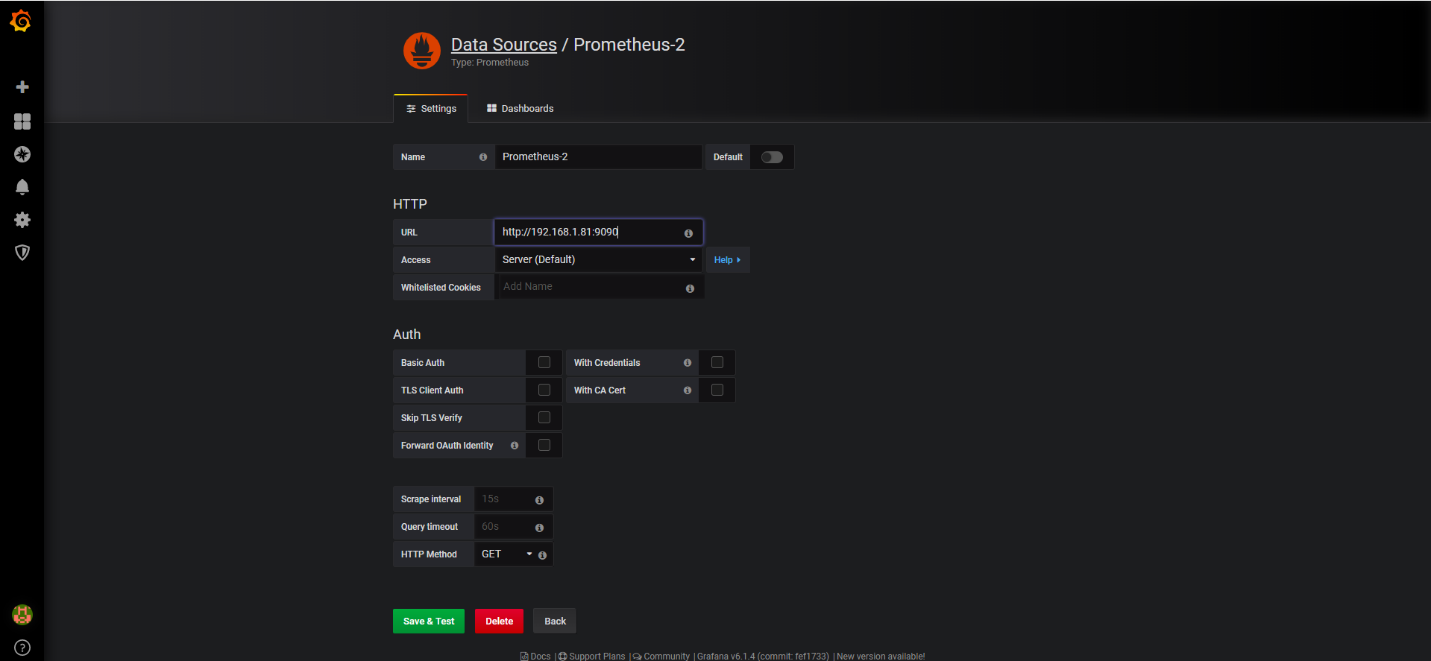
1. Create any file e.g. fio.txt and run the grok-exporter instance using tailer for fio.txt file with following command:

$ tail –f fio.txt | ./grok\_exporter -config ./example/config.yml

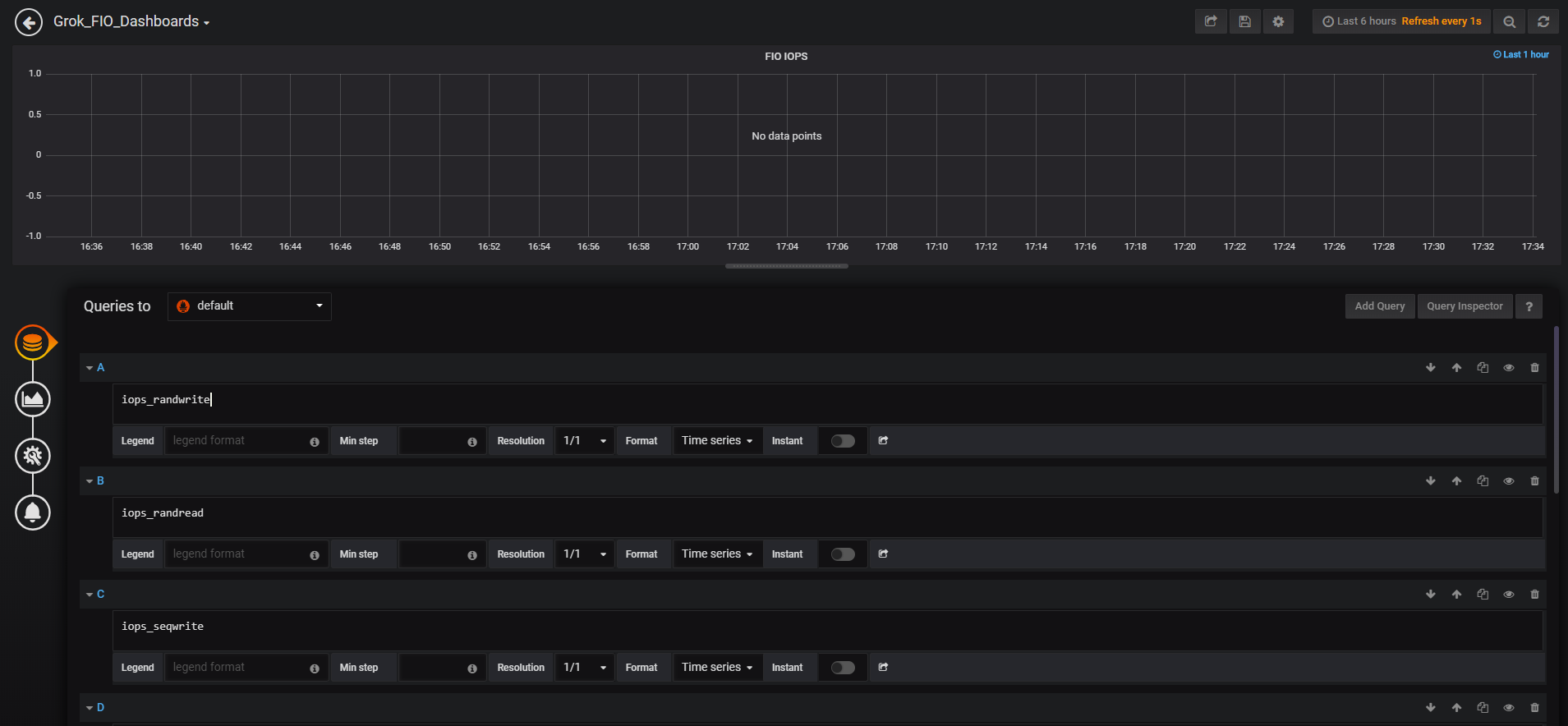
1. Run the FIO command with desired workload and parameters as below:

$ fio --name=randwrite --ioengine=libaio --iodepth=1 --rw=randwrite --bs=4k --direct=0 --size=100M --numjobs=1 --runtime=100 --status-interval=1 --group\_reporting --time\_based >> fio.txt

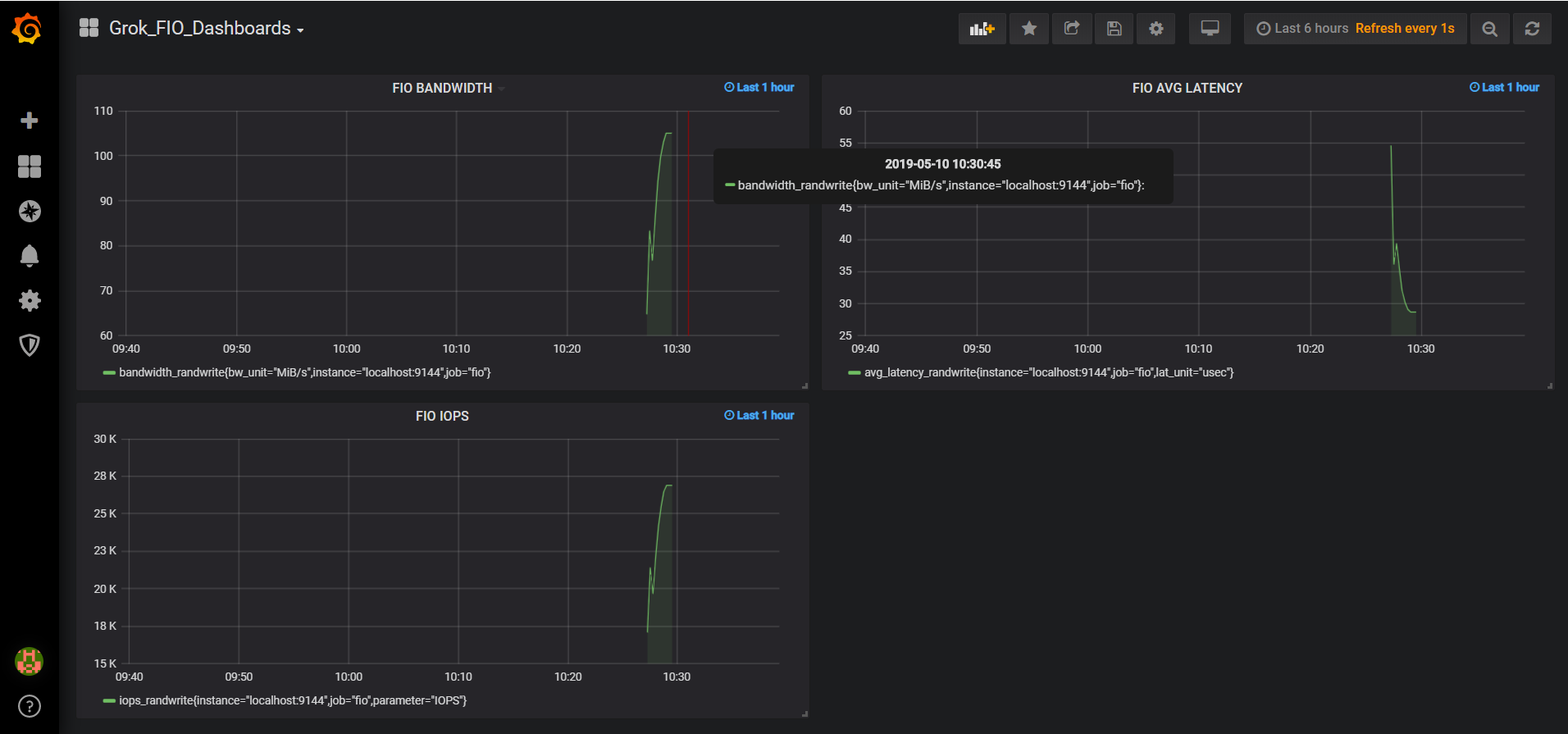
1. Launch Grafana using “http://<IP of Machine>:3000” url address. Login using valid credentials. Add Prometheus server as a Data Source in Grafana. See the sample screenshot below:



1. Create a dashboard in Grafana and add a graph for QoS parameter i.e IOPS, Bandwidth or Latency in the dashboard. Add PromQL queries in the graph with the valid metrics name i.e. iops\_randwrite as defined in above grok-exporter config.yml file. See the sample screenshot below:



1. Adjust the ‘Refresh’ with value 1 second or 5 seconds, and there you go seeing the graph in the dashboard with the time-series values of QoS parameters i.e. IOPS, Bandwidth or Latency which is being generated from FIO output on a real-time basis. You might want to adjust the status-interval and runtime parameters of FIO command to observe the time-series values as per your required duration. See the sample screenshot below:



## Steps involved to configure InfluxDB as a primary storage for Prometheus metrics

Since Prometheus has a limit of storing metrics of 15 days only, next steps is to integrate Prometheus server with passive time-series database InfluxDB so that historical metrics data, which are older than 15 days, can be visualized using Grafana dashboards. Steps for the same as follows:

* 1. Create a target database in InfluxDB instance.

Create a database in your InfluxDB instance to house data sent from Prometheus. In the examples provided below, prometheus is used as the database name, but you’re welcome to use the whatever database name you like.

CREATE DATABASE "prometheus"

* 1. Configure Endpoints in Prometheus config file (/etc/prometheus/prometheus.yml):

To enable the use of the Prometheus remote read and write APIs with InfluxDB, add URL values to the following settings in the [Prometheus configuration file](https://prometheus.io/docs/prometheus/latest/configuration/configuration/):

* [remote\_write](https://prometheus.io/docs/prometheus/latest/configuration/configuration/#%3Cremote_write%3E)
* [remote\_read](https://prometheus.io/docs/prometheus/latest/configuration/configuration/#%3Cremote_read%3E)

The URLs must be resolvable from your running Prometheus server and use the port on which InfluxDB is running (8086 by default). Also include the database name using the db= query parameter.

[Example: Endpoints in Prometheus configuration file](https://docs.influxdata.com/influxdb/v1.7/supported_protocols/prometheus/#example-endpoints-in-prometheus-configuration-file):-

remote\_write:

- url: "http://localhost:8086/api/v1/prom/write?db=prometheus"

remote\_read:

- url: "http://localhost:8086/api/v1/prom/read?db=prometheus"

[Read and write URLs with authentication](https://docs.influxdata.com/influxdb/v1.7/supported_protocols/prometheus/#read-and-write-urls-with-authentication)

If [authentication is enabled on InfluxDB](https://docs.influxdata.com/influxdb/v1.7/administration/authentication_and_authorization/), pass the username and password of an InfluxDB user with read and write privileges using the u= and p= query parameters respectively.

Examples of endpoints with authentication enabled:-

remote\_write:

- url: "http://localhost:8086/api/v1/prom/write?db=prometheus&u=username&p=password"

remote\_read:

- url: "http://localhost:8086/api/v1/prom/read?db=prometheus&u=username&p=password"

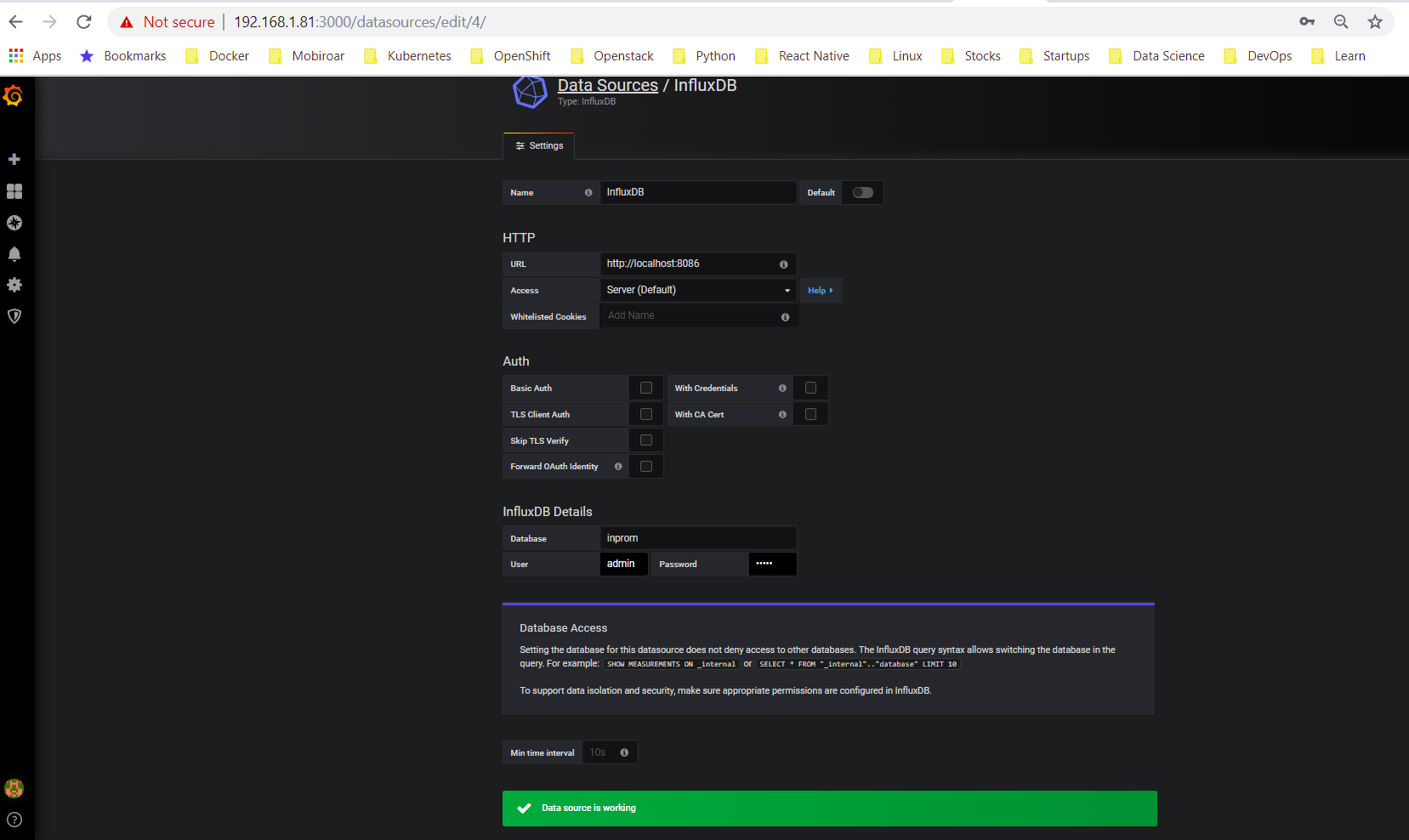
* 1. Create any file e.g. fio.txt and run the grok-exporter instance using tailer for fio.txt file with following command:

$ tail –f fio.txt | ./grok\_exporter -config ./example/config.yml

* 1. Run the FIO command with desired workload and parameters as below:

$ fio --name=seqwrite --ioengine=libaio --iodepth=1 --rw=seqwrite --bs=4k --direct=0 --size=100M --numjobs=1 --runtime=100 --status-interval=1 --group\_reporting --time\_based >> fio.txt

* 1. Launch Grafana using “http://<IP of Machine>:3000” url address. Login using valid credentials. Add InfluxDB database as a Data Source in Grafana. See the sample screenshot below:



* 1. Create a dashboard in Grafana and add a graph for QoS parameter i.e IOPS, Bandwidth or Latency in the dashboard. Add InfluxSQL queries in the graph with the valid metrics name i.e. iops\_seqwrite as defined in above grok-exporter config.yml file. See the sample screenshot below:



* 1. Adjust the ‘Refresh’ with value 1 second or 5 seconds, and there you go seeing the graph in the dashboard with the time-series values of QoS parameters i.e. IOPS, Bandwidth or Latency which is being generated from FIO output on a real-time basis. You might want to adjust the status-interval and runtime parameters of FIO command to observe the time-series values as per your required duration. See the sample screenshot below:

