**Grafana Installation**

**Need to install & configure below services:**

1. [**Influx DB Installation**](#Influx_db_installation)

* **Configure repos & installation steps:**
* **Configure Database**
* **Configure CoLLECTD in influxdb.conf**

1. [**Collectd Installation**](#collectd_installation)

* **Installation Steps**
* **Configure collectd.conf**
* **Configure collectd.conf with common plugins**
* **Service Enable & Restart**

1. [**Grafana Installation**](#grafana_installation)

* **Installation Steps**
* **Service Enable & Start**
* **Default Local URL & Port**
* **Default web User & Password**
* **Add Data Source on Grafana Setting**
* **Configure Influxdb on Grafana**

1. [**Virtual Hosting – Apache & Nginx**](#virtual_hosting)

* **Configure Proxy on Apache**
* **Configure Proxy on Nginx**

1. [**Grafana Dashboard & Panel**](#dashboard_panel)

* **Adding Dashboard**
* **Adding Panel**
* **Panel – Mentioning Title**
* **Select Data Source**
* **Configure Legend**

1. [**Plugins & Queries**](#plugins)

* **CPU Alert**
* **Memory Alert**
* **Disk Size Alert**
* **Nginx Alert**
* **MySQL Alert**
* **Average Load Alert**
* **RabbitMQ Alert**
* **Redis Alert**
* **Mongo Alert**
* **Apache Alert**

1. **Retention Policy on Influx DB**
2. **Configure SMTP**
3. **Configure Alert**

**Influx DB Installation**

**Configure repos & installation steps:**

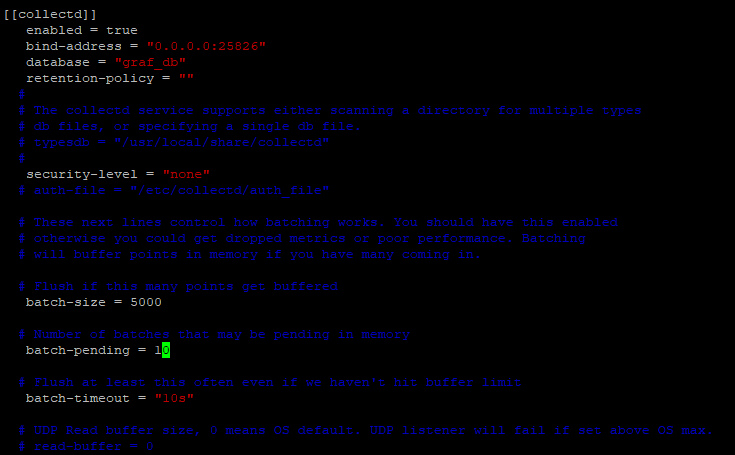
* + curl -sL <https://repos.influxdata.com/influxdb.key> | apt-key add -
  + source /etc/lsb-release
  + echo "deb https://repos.influxdata.com/${DISTRIB\_ID,,} ${DISTRIB\_CODENAME} stable" | tee /etc/apt/sources.list.d/influxdb.list
  + apt-get update
  + apt-get install influxdb
  + systemctl start influx

**Configure Database**

* + Influx (Go to Influx DB)
  + Create database graf\_db (Create database , e.g graf\_db)
  + Show series (Returns a list of [series](https://docs.influxdata.com/influxdb/v1.8/concepts/glossary/#series) for the specified [database](https://docs.influxdata.com/influxdb/v1.8/concepts/glossary/#database).)
  + Exit

**Configure CoLLECTD in influxdb.conf**

Uncomment below lines from /etc/influxdb/influxdb.conf and mention the database name which you been created “graf\_db”.



**Note: Restart Influx dB after Collectd installation.**

**Collectd Installation**

**Collectd daemon task is to gather and store data statistics on the system that it runs on.**

**Installation Steps**

* + apt-get install collectd

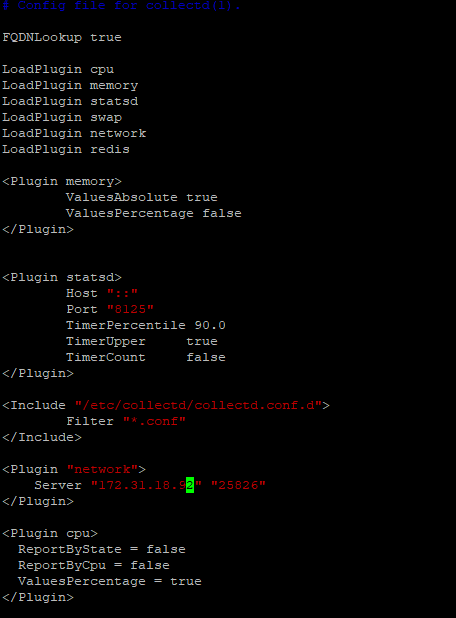
**Configure collectd.conf**

* + vim /etc/collecd/collectd.conf

Server – Add influxdb Server IP



**Configure collectd.conf with common plugins**



**Note: Restart Influx dB & Collectd service and enable as well.**

**Service Enable & Restart**

* + systemctl restart collectd.service
  + systemclt restart influx.service
  + systemctl enable collectd.service
  + systemctl enable influx.service

**Grafana Installation**

**Installation Steps**

* 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 –

**Add this repository for stable releases**

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

**Add the repository update**

* sudo apt-get update
* sudo apt-get install Grafana

**Service Enable & Start**

* sudo systemctl daemon-reload
* sudo systemctl start grafana-server
* sudo systemctl enable grafana-server.service

**Default Local URL & Port**

* http://Server\_IP:3000/

**Default web User & Password**

**Use below default user & password at First Time.**

* Username: admin
* Password: admin

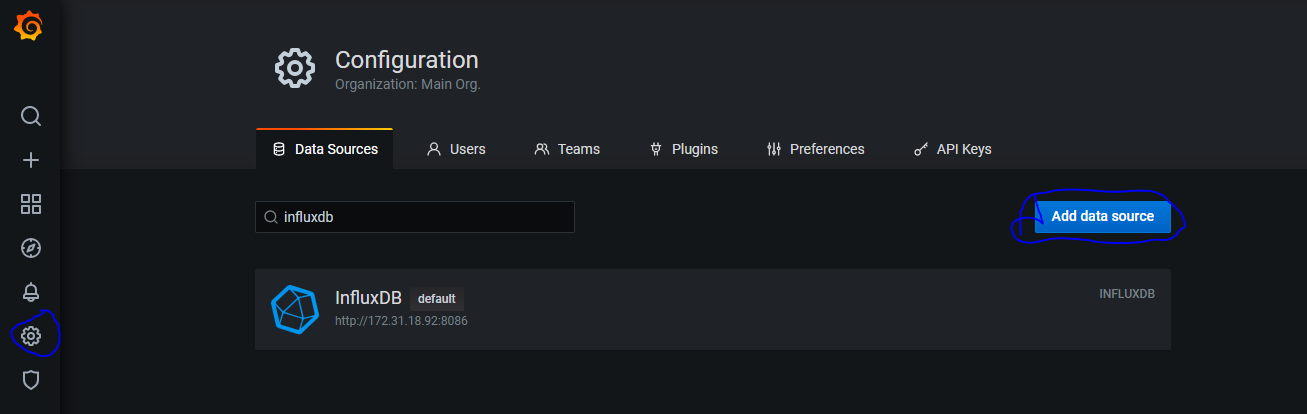
**After Login First time it will prompt to enter New Password.**

* Change the password.

**Add Data Source on Grafana Setting**

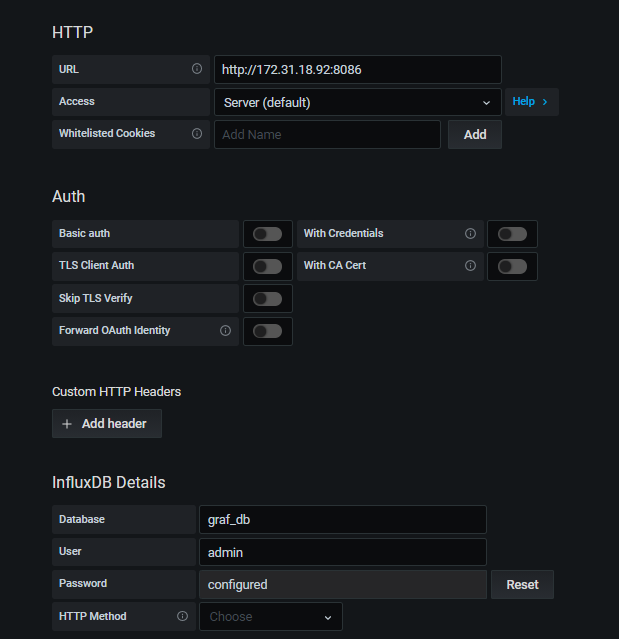
**Go to setting and add data source**

* Choose Influxdb



Configure Influxdb on Grafana

* URL : Server IP where Influx DB installed and default influx port
* Database: Enter database created in Influx DB
* Username : admin
* Password : admin
* Save & Apply



**Virtual Hosting Configuration**

Configure Proxy on Apache

<VirtualHost \*:80>

ServerAdmin admin@emailid.com

ServerName domain/subdomain\_name

ServerAlias www.domain.name

RequestHeader set Host "domain\_name"

DocumentRoot /var/www/html/jenkins

<Directory "/var/www/html/jenkins">

AllowOverride all

Options -Indexes +FollowSymLinks

Require all granted

</Directory>

ProxyPass / http://localhost:3000/

ProxyPassReverse / http://localhost:3000/

ErrorLog /var/log/apache2/grafana\_error.log

CustomLog /var/log/apache2/grafana\_access.log combined

</VirtualHost>

Configure Proxy on Nginx

server {

listen 443 ssl http2;

listen [::]:443 ssl http2;

ssl\_certificate /etc/ssl/certs/ssl\_surbo\_prod\_cert.crt;

ssl\_certificate\_key /etc/ssl/private/ssl\_surbo\_prod\_pvt.key;

server\_name domain\_name; # substitute your machine's IP address or FQDN

charset utf-8;

# max upload size

client\_max\_body\_size 75M; # adjust to taste

access\_log /var/log/nginx/grafana-access.log;

error\_log /var/log/nginx/grafana-error.log;

location / {

proxy\_pass http://127.0.0.1:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

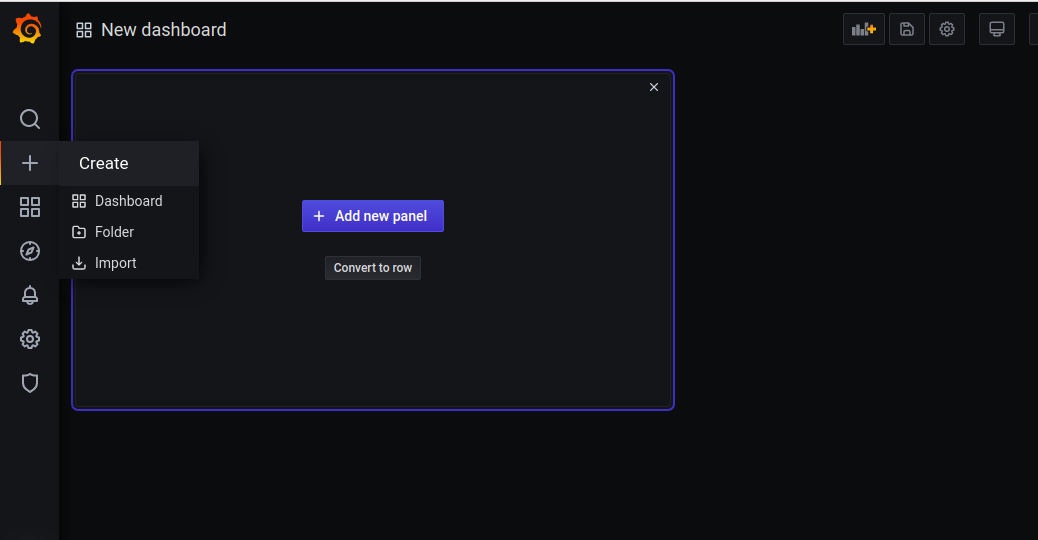
proxy\_cache\_bypass $http\_upgrade;

}

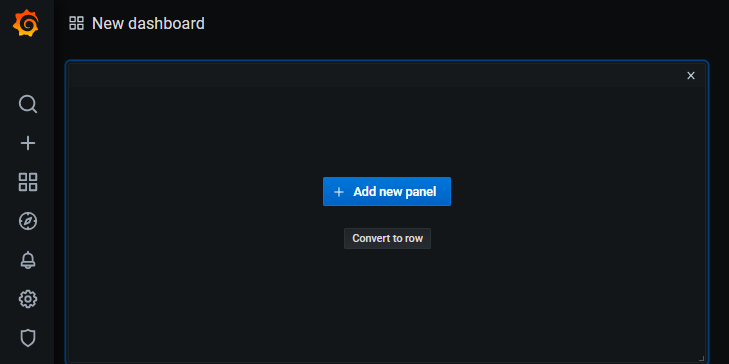
}

**Grafana DashBoard & Panel**

Add Dashboard

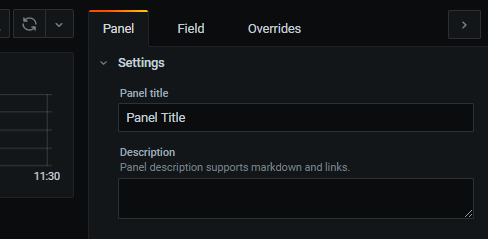


Add Panel



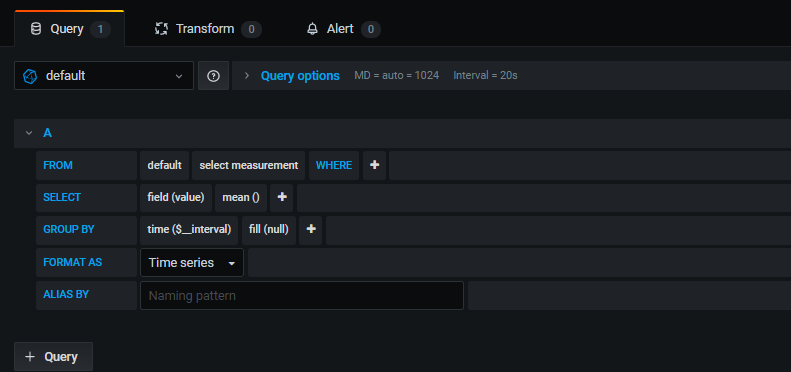
Panel Name

* Set panel name you want to show



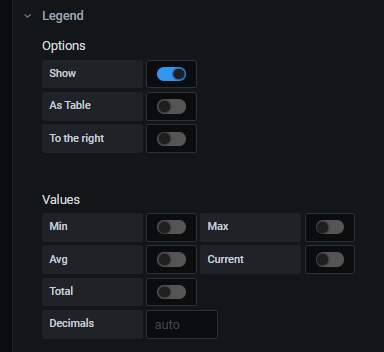
Select Data Source

* select Data Source you want to show stats on Grafana  
  **Query for getting data and show stats on grafana**
* **from** : select measurement from influxdb
* **host** : which server stats you want to show
* **type\_instance** : which type of activity you want to show
* **select** : select value on the server
* **alias-by** : name define of activity



Configure Legend

**Choose as per requirement, It will show as table and in the right as well it showing the Min, Avg, Max values as per your query.**



**Plugins & Queries**

**Need to be add on /etc/collectd/collectd.conf**

**CPU Alert**

* **Plugins:**

LoadPlugin cpu

<Plugin cpu>

ReportByState = false

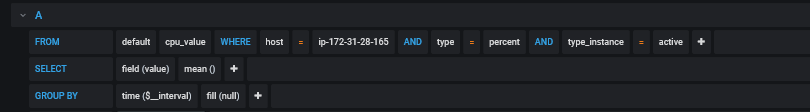
ReportByCpu = false

ValuesPercentage = true

</Plugin>

* **Query:**

SELECT mean("value") FROM "cpu\_value" WHERE ("host" = 'ip-172-31-28-165' AND "type" = 'percent' AND "type\_instance" = 'active') AND $timeFilter GROUP BY time($\_\_interval) fill(null)



**Memory Alert**

* **Plugins:**

LoadPlugin memory

<Plugin memory>

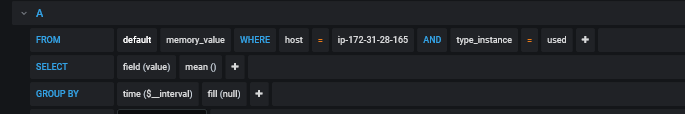
ValuesAbsolute true

ValuesPercentage false

</Plugin>

* **Query:**

SELECT mean("value") FROM "memory\_value" WHERE ("host" = 'ip-172-31-28-165' AND "type\_instance" = 'used') AND $timeFilter GROUP BY time($\_\_interval) fill(null)



**Disk Size Alert**

* **Plugins:**

LoadPlugin df

<Plugin "df">

Device "/dev/xvda1" #(Partition Name)

MountPoint "/"

FSType "ext4"

IgnoreSelected false

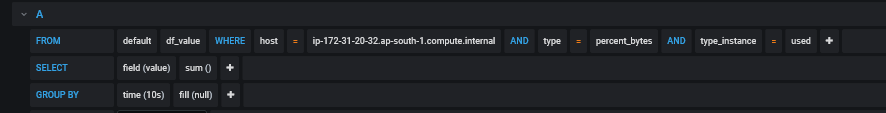
ReportInodes false

ValuesPercentage true

</Plugin>

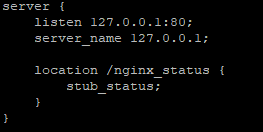
* **Query:**

SELECT sum("value") FROM "df\_value" WHERE ("host" = 'ip-172-31-20-32.ap-south-1.compute.internal' AND "type" = 'percent\_bytes' AND "type\_instance" = 'used') AND $timeFilter GROUP BY time(10s) fill(null)



**Nginx Alert**

**Note: Configure Nginx Status configuration on Virtual Host – /etc/nginx/conf.d directory**



* **Plugins:**

LoadPlugin nginx

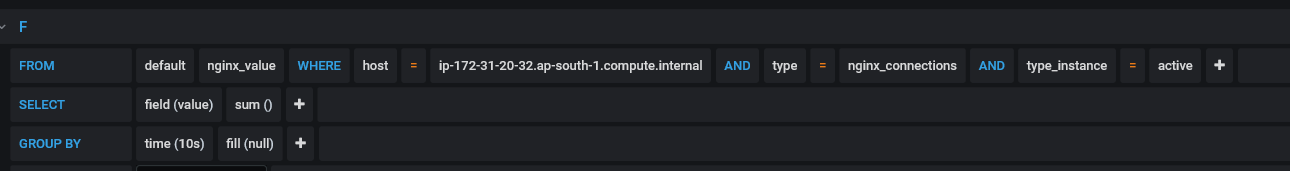
<Plugin "nginx">

URL "http://localhost/nginx\_status"

</Plugin>

* **Query:**

SELECT sum("value") FROM "nginx\_value" WHERE ("host" = 'ip-172-31-20-32.ap-south-1.compute.internal' AND "type" = 'nginx\_connections' AND "type\_instance" = 'active') AND $timeFilter GROUP BY time(10s) fill(null)



**MySQL Alert**

* **Plugins:**

LoadPlugin mysql

<Plugin mysql>

<Database "database\_name"> #(For Spedific Database)

Host "IP\_address" # (Private IP)

User "username"

Password "password"

Port XXXX

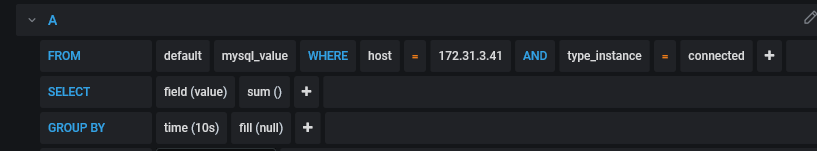
MasterStats false

</Database>

</Plugin>

* **Query:**

SELECT sum("value") FROM "mysql\_value" WHERE ("host" = '172.31.3.41' AND "type\_instance" = 'connected') AND $timeFilter GROUP BY time(10s) fill(null)



**Average Load Alert**

* **Plugins:**

LoadPlugin load

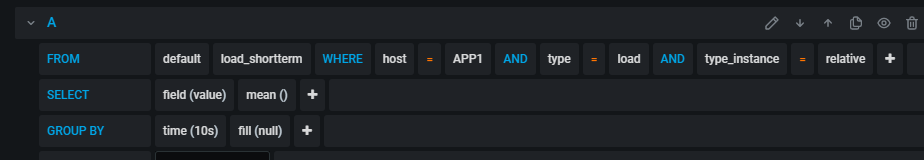
<Plugin load>

ReportRelative true

</Plugin>

* **Query:**

SELECT mean("value") FROM "load\_shortterm" WHERE ("host" = 'APP1' AND "type" = 'load' AND "type\_instance" = 'relative') AND $timeFilter GROUP BY time(10s) fill(null)



**RabbitMQ Connections Alert**

**Note: Need to install before running rabbitmq\_monitoring from below command**

#pip install collectd-rabbitmq-monitoring

#systemctl restart collectd

* **Plugins:**

<LoadPlugin python>

Globals true

</LoadPlugin>

<Plugin python>

LogTraces true

Interactive false

Import "collectd\_rabbitmq\_monitoring"

<Module collectd\_rabbitmq\_monitoring>

# Adjust these parameters for your install:

interval 10

host "192.168.XXX.XXX"

port 156??

username "????????"

password "??????????"

# Omit message\_count if you do not want to count any messages on

# specific queues.

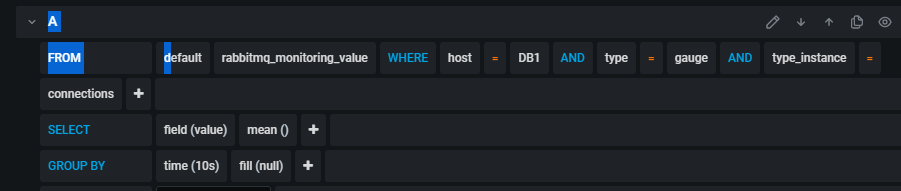
message\_count "metering.sample" "notifications.info"

</Module>

</Plugin>

* **Query:**

SELECT mean("value") FROM "rabbitmq\_monitoring\_value" WHERE ("host" = 'DB1' AND "type" = 'gauge' AND "type\_instance" = 'connections') AND $timeFilter GROUP BY time(10s) fill(null)



**Redis Alert**

* **Plugins:**

<Plugin redis>

<Node "mynode">

Host "192.168.XXX.XXX"

Port "78??"

Timeout 2000

</Node>

</Plugin>

* **Query:**

SELECT mean("value") FROM "load\_shortterm" WHERE ("host" = 'APP1' AND "type" = 'load' AND "type\_instance" = 'relative') AND $timeFilter GROUP BY time(10s) fill(null)

**Apache2 Alert**

### \*\* Collectd File Logs

LoadPlugin "logfile"

<Plugin "logfile">

LogLevel "info"

File "/var/log/collectd.log"

Timestamp true

</Plugin>

This will add Default Log location of Collectd.

### \*\* MongoDB Alert:

To Add mongodb alert first add collectd plugin:

<Plugin python>

# mongodb.py is at path /opt/collectd-plugins/mongodb.py

ModulePath "/opt/collectd-plugins/"

Import "mongodb"

<Module mongodb>

Host "172.31.3.41"

Password "suRb05MnG01"

Database "admin" "botplatform"

</Module>

</Plugin>

### Add mongodb.py file:

#

# Plugin to collectd statistics from MongoDB

#

import collectd

from pymongo import MongoClient

from pymongo.read\_preferences import ReadPreference

from distutils.version import LooseVersion as V

class MongoDB(object):

def \_\_init\_\_(self):

self.plugin\_name = "mongo"

self.mongo\_host = "172.31.3.41"

self.mongo\_port = 17901

self.mongo\_db = ["admin", ]

self.mongo\_user = "admin1"

self.mongo\_password = "suRb05MnG01"

self.lockTotalTime = None

self.lockTime = None

self.accesses = None

self.misses = None

def submit(self, type, instance, value, db=None):

if db:

plugin\_instance = '%s-%s' % (self.mongo\_port, db)

else:

plugin\_instance = str(self.mongo\_port)

v = collectd.Values()

v.plugin = self.plugin\_name

v.plugin\_instance = plugin\_instance

v.type = type

v.type\_instance = instance

v.values = [value, ]

v.dispatch()

def get\_db\_and\_collection\_stats(self):

con = MongoClient(host=self.mongo\_host, port=self.mongo\_port, read\_preference=ReadPreference.SECONDARY)

db = con[self.mongo\_db[0]]

if self.mongo\_user and self.mongo\_password:

db.authenticate(self.mongo\_user, self.mongo\_password)

server\_status = db.command('serverStatus')

version = server\_status['version']

at\_least\_2\_4 = V(version) >= V('2.4.0')

# operations

for k, v in server\_status['opcounters'].items():

self.submit('total\_operations', k, v)

# memory

for t in ['resident', 'virtual', 'mapped']:

self.submit('memory', t, server\_status['mem'][t])

# connections

self.submit('connections', 'current', server\_status['connections']['current'])

if 'available' in server\_status['connections']:

self.submit('connections', 'available', server\_status['connections']['available'])

if 'totalCreated' in server\_status['connections']:

self.submit('connections', 'totalCreated', server\_status['connections']['totalCreated'])

# network

if 'network' in server\_status:

for t in ['bytesIn', 'bytesOut', 'numRequests']:

self.submit('bytes', t, server\_status['network'][t])

# locks

if 'lockTime' in server\_status['globalLock']:

if self.lockTotalTime is not None and self.lockTime is not None:

if self.lockTime == server\_status['globalLock']['lockTime']:

value = 0.0

else:

value = float(server\_status['globalLock']['lockTime'] - self.lockTime) \* 100.0 / float(server\_status['globalLock']['totalTime'] - self.lockTotalTime)

self.submit('percent', 'lock\_ratio', value)

self.lockTime = server\_status['globalLock']['lockTime']

self.lockTotalTime = server\_status['globalLock']['totalTime']

# indexes

if 'indexCounters' in server\_status:

accesses = None

misses = None

index\_counters = server\_status['indexCounters'] if at\_least\_2\_4 else server\_status['indexCounters']['btree']

if self.accesses is not None:

accesses = index\_counters['accesses'] - self.accesses

if accesses < 0:

accesses = None

misses = (index\_counters['misses'] or 0) - (self.misses or 0)

if misses < 0:

misses = None

if accesses and misses is not None:

self.submit('cache\_ratio', 'cache\_misses', int(misses \* 100 / float(accesses)))

else:

self.submit('cache\_ratio', 'cache\_misses', 0)

self.accesses = index\_counters['accesses']

self.misses = index\_counters['misses']

for mongo\_db in self.mongo\_db:

db = con[mongo\_db]

if self.mongo\_user and self.mongo\_password:

con[self.mongo\_db[0]].authenticate(self.mongo\_user, self.mongo\_password)

db\_stats = db.command('dbstats')

# stats counts

self.submit('counter', 'object\_count', db\_stats['objects'], mongo\_db)

self.submit('counter', 'collections', db\_stats['collections'], mongo\_db)

self.submit('counter', 'num\_extents', db\_stats['numExtents'], mongo\_db)

self.submit('counter', 'indexes', db\_stats['indexes'], mongo\_db)

# stats sizes

self.submit('file\_size', 'storage', db\_stats['storageSize'], mongo\_db)

self.submit('file\_size', 'index', db\_stats['indexSize'], mongo\_db)

self.submit('file\_size', 'data', db\_stats['dataSize'], mongo\_db)

# collection stats

collections = db.collection\_names()

for collection in collections:

collection\_stats = db.command('collstats', collection)

if 'wiredTiger' in collection\_stats:

if 'cursor' in collection\_stats['wiredTiger']:

for k, v in collection\_stats['wiredTiger']['cursor'].items():

self.submit('collection\_stats', (collection + '-' + k), v, mongo\_db)

con.close()

def config(self, obj):

for node in obj.children:

if node.key == 'Port':

self.mongo\_port = int(node.values[0])

elif node.key == 'Host':

self.mongo\_host = node.values[0]

elif node.key == 'User':

self.mongo\_user = node.values[0]

elif node.key == 'Password':

self.mongo\_password = node.values[0]

elif node.key == 'Database':

self.mongo\_db = node.values

else:

collectd.warning("mongodb plugin: Unkown configuration key %s" % node.key)

mongodb = MongoDB()

collectd.register\_read(mongodb.get\_db\_and\_collection\_stats)

collectd.register\_config(mongodb.config)

Now move to the (/usr/share/collectd) Directory and cat the types.db:

Add the missing parameter:

|  |
| --- |
| cache\_ratio value:GAUGE:0:100 |
|  | connections value:COUNTER:0:U |
|  | counter value:COUNTER:U:U |
|  | file\_size bytes:GAUGE:0:U |
|  | memory value:GAUGE:0:281474976710656 |
|  | percent percent:GAUGE:0:100.1 |
|  | total\_operations value:DERIVE:0:U |
|  | collection\_stats value:COUNTER:U:U |