# grafana/grafana:

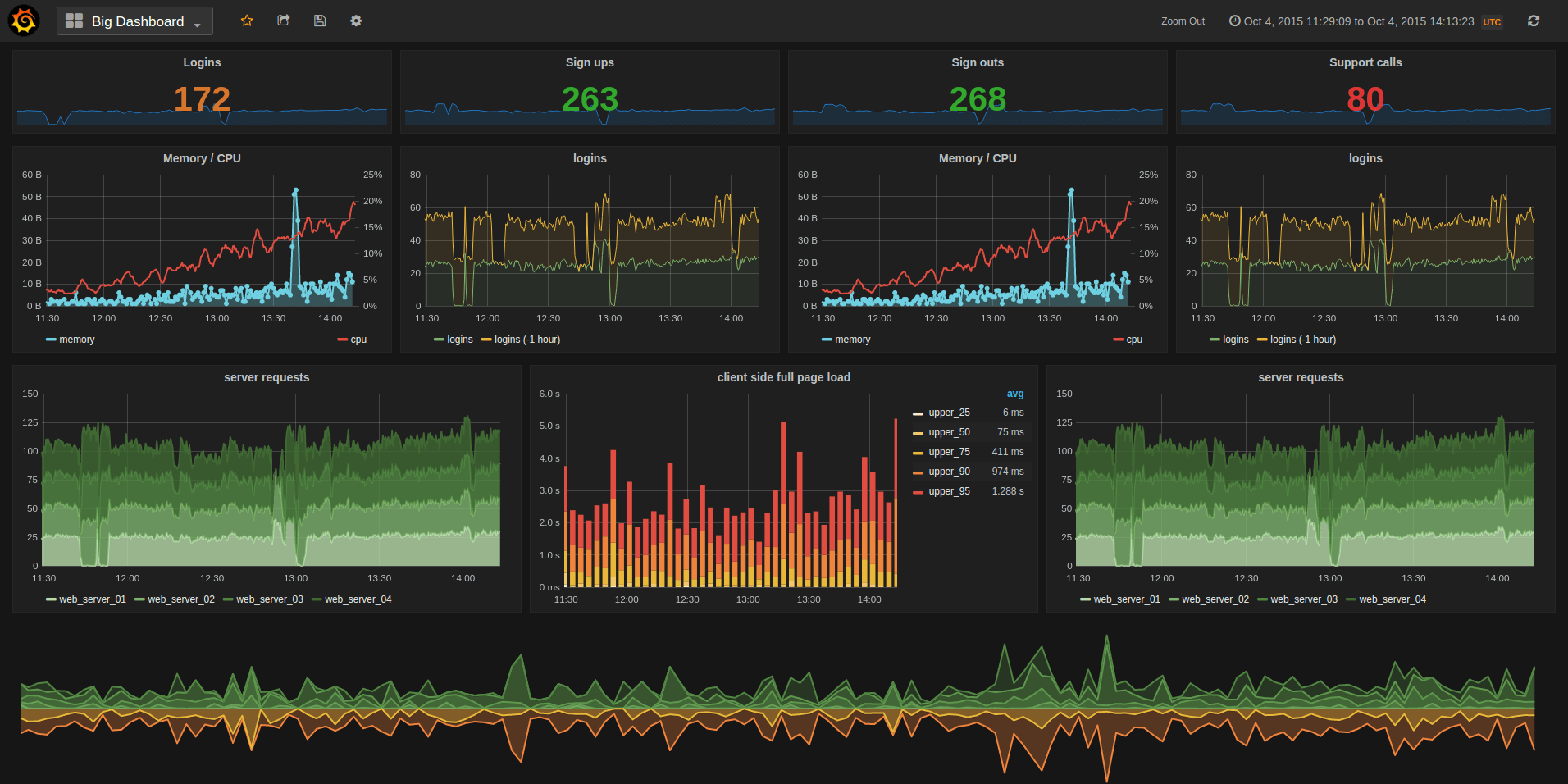
https://github.com/grafana/grafana

The tool for beautiful monitoring and metric analytics & dashboards for Graphite, InfluxDB & Prometheus & More

[Grafana](https://grafana.com/)

[Website](https://grafana.com/) | [Twitter](https://twitter.com/grafana) | [Community & Forum](https://community.grafana.com/)

Grafana is an open source, feature rich metrics dashboard and graph editor for Graphite, Elasticsearch, OpenTSDB, Prometheus and InfluxDB.

[](https://camo.githubusercontent.com/d010ea19c70677a0bfd8a64fc01d2b0948e1ffc1/687474703a2f2f646f63732e67726166616e612e6f72672f6173736574732f696d672f66656174757265732f64617368626f6172645f6578312e706e67)

* [Install instructions](http://docs.grafana.org/installation/)
* [What's New in Grafana 2.0](http://docs.grafana.org/guides/whats-new-in-v2/)
* [What's New in Grafana 2.1](http://docs.grafana.org/guides/whats-new-in-v2-1/)
* [What's New in Grafana 2.5](http://docs.grafana.org/guides/whats-new-in-v2-5/)
* [What's New in Grafana 3.0](http://docs.grafana.org/guides/whats-new-in-v3/)
* [What's New in Grafana 4.0](http://docs.grafana.org/guides/whats-new-in-v4/)
* [What's New in Grafana 4.1](http://docs.grafana.org/guides/whats-new-in-v4-1/)
* [What's New in Grafana 4.2](http://docs.grafana.org/guides/whats-new-in-v4-2/)

Features

Graphing

* Fast rendering, even over large timespans
* Click and drag to zoom
* Multiple Y-axis, logarithmic scales
* Bars, Lines, Points
* Smart Y-axis formatting
* Series toggles & color selector
* Legend values, and formatting options
* Grid thresholds, axis labels
* [Annotations](http://docs.grafana.org/reference/annotations/)
* Any panel can be rendered to PNG (server side using phantomjs)

Dashboards

* Create, edit, save & search dashboards
* Change column spans and row heights
* Drag and drop panels to rearrange
* [Templating](http://docs.grafana.org/reference/templating/)
* [Scripted dashboards](http://docs.grafana.org/reference/scripting/)
* [Dashboard playlists](http://docs.grafana.org/reference/playlist/)
* [Time range controls](http://docs.grafana.org/reference/timerange/)
* [Share snapshots publicly](http://docs.grafana.org/v2.0/reference/sharing/)

InfluxDB

* Use InfluxDB as a metric data source, annotation source
* Query editor with field and tag typeahead, easy group by and function selection

Graphite

* Graphite target expression parser
* Feature rich query composer
* Quickly add and edit functions & parameters
* Templated queries
* [See it in action](http://docs.grafana.org/datasources/graphite/)

Elasticsearch, Prometheus & OpenTSDB

* Feature rich query editor UI

Alerting

* Define alert rules using graphs & query conditions
* Schedule & evalute alert rules, send notifications to Slack, Hipchat, Email, PagerDuty, etc.

Requirements

There are no dependencies except an external time series data store. For dashboards and user accounts Grafana can use an embedded database (sqlite3) or you can use an external SQL data base like MySQL or Postgres.

Installation

Head to [grafana.org](http://docs.grafana.org/installation/) and [download](https://grafana.com/get) the latest release.

If you have any problems please read the [troubleshooting guide](http://docs.grafana.org/installation/troubleshooting/).

Documentation & Support

Be sure to read the [getting started guide](http://docs.grafana.org/guides/gettingstarted/) and the other feature guides.

Run from master

If you want to build a package yourself, or contribute. Here is a guide for how to do that. You can always find the latest master builds [here](https://grafana.com/grafana/download)

Dependencies

* Go 1.8.1
* NodeJS LTS

Get Code

go get github.com/grafana/grafana

Since imports of dependencies use the absolute path github.com/grafana/grafana within the $GOPATH, you will need to put your version of the code in $GOPATH/src/github.com/grafana/grafana to be able to develop and build grafana on a cloned repository. To do so, you can clone your forked repository directly to $GOPATH/src/github.com/grafana or you can create a symbolic link from your version of the code to $GOPATH/src/github.com/grafana/grafana. The last options makes it possible to change easily the grafana repository you want to build.

go get github.com/\*your\_account\*/grafana

mkdir $GOPATH/src/github.com/grafana

ln -s $GOPATH/src/github.com/\*your\_account\*/grafana $GOPATH/src/github.com/grafana/grafana

Building the backend

cd $GOPATH/src/github.com/grafana/grafana

go run build.go setup

go run build.go build

Building frontend assets

To build less to css for the frontend you will need a recent version of **node (v6+)**, npm (v2.5.0) and grunt (v0.4.5). Run the following:

npm install -g yarn

yarn install --pure-lockfile

npm run build

To build the frontend assets only on changes:

sudo npm install -g grunt-cli # to do only once to install grunt command line interface

grunt watch

Recompile backend on source change

To rebuild on source change.

go get github.com/Unknwon/bra

bra run

Running

./bin/grafana-server

Open grafana in your browser (default: http://localhost:3000) and login with admin user (default: user/pass = admin/admin).

Dev config

Create a custom.ini in the conf directory to override default configuration options. You only need to add the options you want to override. Config files are applied in the order of:

1. grafana.ini
2. dev.ini (if found)
3. custom.ini

Create a pull request

Before or after you create a pull request, sign the [contributor license agreement](http://docs.grafana.org/project/cla/).

Contribute

If you have any idea for an improvement or found a bug do not hesitate to open an issue. And if you have time clone this repo and submit a pull request and help me make Grafana the kickass metrics & devops dashboard we all dream about!

License

Grafana is distributed under Apache 2.0 License. Work in progress Grafana 2.0 (with included Grafana backend)

# Docs Home | Grafana Documentation

http://docs.grafana.org/

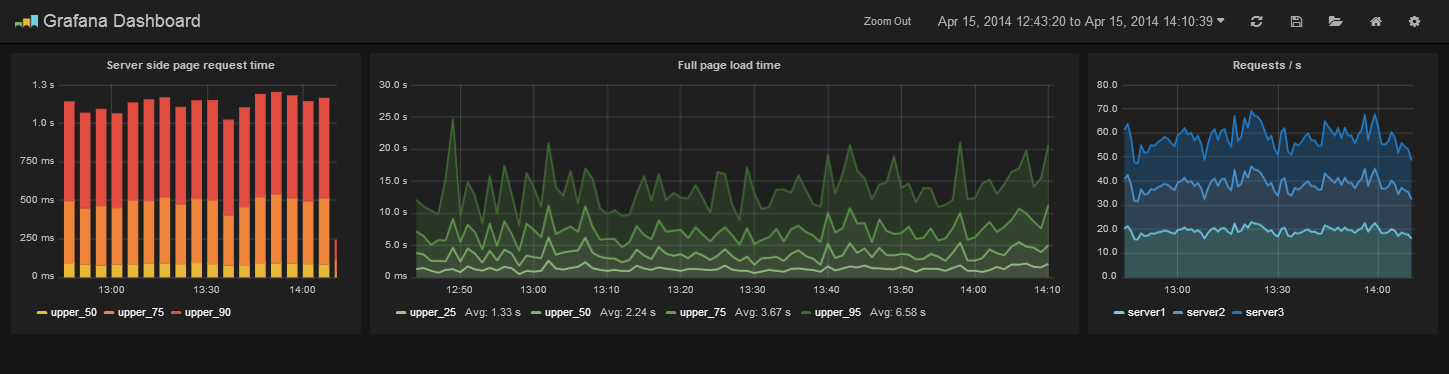
# 用Grafana为Elasticsearch做日志分析 - chszs的专栏 - 博客频道 - CSDN.NET

http://blog.csdn.net/chszs/article/details/50373453

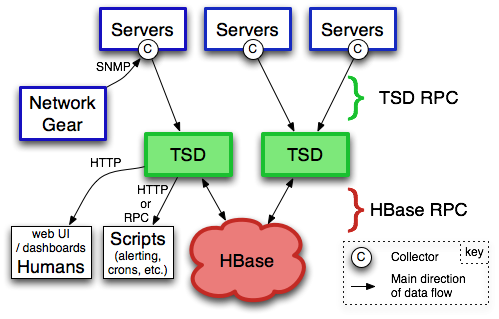
用Grafana为Elasticsearch做日志分析

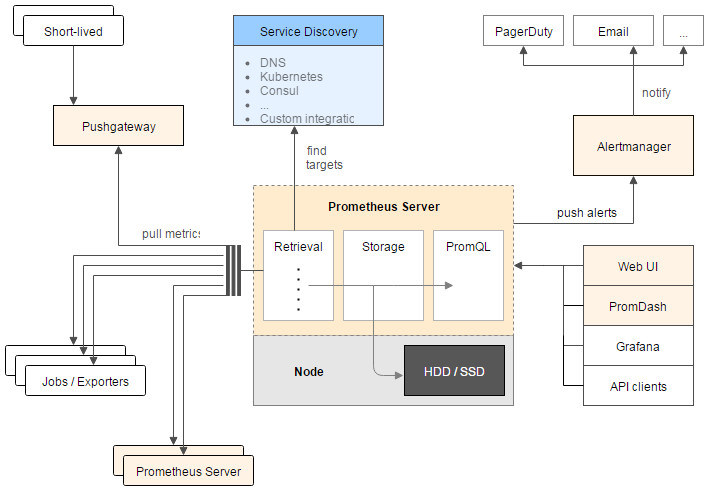
作者：chszs，未经博主允许不得转载。经许可的转载需注明作者和博客主页：[**http://blog.csdn.net/chszs**](http://blog.csdn.net/chszs)

Grafana是一个开源的、功能强大的指标仪表板和图形编辑器工具，它面向Graphite、Elasticsearch、OpenTSDB、Prometheus和InfluxDB等数据源。目前Grafana的最新版本为2.6版。

Grafana仪表板界面如下：   


Graphite：Graphite是一个可扩展的实时图表，最新版本为0.9.10，地址： <http://graphite.wikidot.com/>   


OpenTSDB：OpenTSDB是一个基于[**Hbase**](http://lib.csdn.net/base/hbase)的分布式、可扩展的、基于时间序列的实时监控信息收集和展示平台。它支持秒级数据采集metrics，使用HBase进行永久存储，可以做容量规划，并很容易的接入到现有的监控系统里。OpenTSDB可以从大规模的设备中获取相应的metrics并进行存储、索引以及服务，从而使得这些数据更容易让人理解，如web化，图形化等。<http://opentsdb.net/index.html>   


Prometheus：是一个开源的系统和服务的监控系统。地址： <https://github.com/prometheus/prometheus>   


InfluxDB：InfluxDB是一个开源的、分布式的、基于时间序列的[**数据库**](http://lib.csdn.net/base/mysql)，它没有外部依赖。InfluxDB可用于记录测量、事件以及性能分析。地址：<https://influxdata.com/>

Grafana可以作为Kibana的替代品。Grafana最令人称道的是它提供的可视化仪表板工具，可以汇集各种数据源（比如InfluxDB）的测量数据并以图形方式显示。Grafana是Kinana的一个分支，但它没有提供对以Elasticsearch作为数据源的支持。还好在Grafana 2.5版以后，增加了对以Elasticsearch作为数据源的支持。在Sematext数据分析网站有介绍。

Sematext网站：<https://sematext.com/>

Elasticsearch通常不用于存储测量数据，而是常常用于存储随时间不停记录的数据，比如日志数据或事件数据（可以想想物联网IoT）。Grafana 2.5在显示方面只支持数值类型的显示，但在2.6版开始支持文本数据的表格显示。

Grafana的安装很简单，以Debian安装为例：

执行命令：

$ wget https://grafanarel.s3.amazonaws.com/builds/grafana\_2.6.0\_amd64.deb

$ sudo apt-get install -y adduser libfontconfig

$ sudo dpkg -i grafana\_2.6.0\_amd64.deb

启动服务器：

$ sudo service grafana-server start

# Grafana

grafana/grafana: Gorgeous metric viz, dashboards & editors for Graphite, InfluxDB & OpenTSDB

https://github.com/grafana/grafana

open-falcon/grafana-openfalcon-datasource: openfalcon plugin for grafana

https://github.com/open-falcon/grafana-openfalcon-datasource

# Open-falcon上搭建Grafana - blueswind8306 - ITeye技术网站

http://blueswind8306.iteye.com/blog/2287561

背景需求

Open-falcon本身自带的dashboard感觉功能不够强大，希望能够接入Grafana做更加丰富、灵活的图形展现。整个安装过程由于在公司服务器上安装，一些依赖包被墙，弄的比较折腾，所以把整个安装过程记录下来。希望对更多的人有帮助。

系统环境及软件版本

操作系统：CentOS 6.6

内核版本：2.6.32-504.el6.x86\_64

Open-falcon版本：0.1.0

Grafana版本：PR#3787的patch版本，基于Grafana-2.6

参考资料

Open-falcon支持Grafana的帮助文档

Grafana官方安装文档

Go的安装

在以下地址可以下载到Go，注意Grafana-2.6需要的Go最低版本是1.5，我安装的版本是Go1.5.3

http://golangtc.com/download

我将go安装到/opt/gohome/go目录下，并且指一个软链/opt/gohome/default到这个目录，方便未来升级go版本：

Shell代码 收藏代码

ln -s /opt/gohome/go /opt/gohome/default

创建工作目录：

Shell代码 收藏代码

mkdir /opt/gohome/workspace

增加环境变量：

Shell代码 收藏代码

vi /etc/profile：

export GOROOT=/opt/gohome/default

export GOPATH=/opt/gohome/workspace

export PATH=$GOROOT/bin:$GOPATH/bin:$PATH

source /etc/profile

检查安装好以后的go版本：

Shell代码 收藏代码

$ go version

go version go1.5.3 linux/amd64

nodejs的安装

nodejs安装很简单，直接通过官网下载二进制包，解压并指一下环境变量就好了

Grafana的安装

由于需要安装的Grafana必须是PR#3787的patch才能支持open-falcon，所以我直接通过github下载了这个PR提交者fork的分支代码（因为这个分支的merge请求被拒绝了）。

zip包下载地址：

https://github.com/hitripod/grafana/archive/feature-openfalcon.zip

下载成功后，将zip文件放到以下目录并解压：

Shell代码 收藏代码

cd /opt/gohome/workspace/src/github.com/grafana

unzip grafana-feature-openfalcon.zip

mv grafana-feature-openfalcon.zip grafana

由于下载的zip包的Godeps目录下已经包含了所有的依赖包，所以不需要下载依赖包，可以直接build：

Shell代码 收藏代码

cd /opt/gohome/workspace/src/github.com/grafana/grafana

go run build.go setup

go run build.go build

后续build前端代码的过程和Grafana官网安装文档基本一致就不再赘述了，注意npm install可能由于网络问题依赖下载不完整，可以多试几遍就好了

装好后，启动grafana-server，浏览器访问3000端口，默认管理员账号admin,admin就可以登陆了，之后按照open-falcon相关文档接入数据源就好了

# 利用Grafana为OpenStack搭建监控系统

- 其他网络技术 - 红黑联盟

http://www.2cto.com/net/201701/582106.html

**利用Grafana为OpenStack搭建现代化监控**[**系统**](http://www.2cto.com/os/)**：**首先简单说下为什么会用grafana，最近公司在新机房上了300多台物理机，其中有60台分配给我做OpenStack的私有云环境。  
  
OpenStack部署很快，三下五除二很快就上线了Mitaka的版本。但是在分给各个运维使用一段时间后，暴露一些问题。

场景一

某个干坏事的小朋友在[虚拟机](http://www.2cto.com/os/xuniji/)里面搞性能测试，直接打满cpu使用率和物理机的网络IO，影响这个物理机上的其他虚拟机。

OpenStack的Cinder卷采用的LVM + Iscsi方式提供，线上虚拟机的某个在某个时间点同步大量数据到数据卷，造成打满存储网络。直接表现的现象就是虚拟机里面的/dev/vdb设备掉线，造成异常。

种种现象表明，我需要接入一个监控系统，实现快速定位到某个物理机或者虚拟机的异常指标。之前打算采用公司现有的zabbix监控框架，但是有两个点让我放弃使用zabbix的方案。其一，zabbix在绘图制表以及对指标排序方面相比grafana来还是有一定差距；二来，部门负责监控的同事太忙了,没时间开发我提出的需求，没办法只有自己动手（苦笑脸）。

选择

既然要自己动手，那就要选择合适自己监控系统。目前网上的方案太多，刚开始找的时候简直一脸懵逼，直到我看到一片文章，标题我忘了，大概内容是利用Collectd + Influxdb + Grafana来做虚拟机监控。遗憾的是，那篇文章讲得很短，几乎没有实质性的内容。好吧，既然有人提出过方案，那我没理由不试一下。

Collectd

简单来说Collectd是用C开发一套高性能的监控指标采集agent，官网上已经有丰富的插件，实现各种监控指标。同时也支持通过Shell、[Python](http://www.2cto.com/kf/web/Python/)、Ruby、Perl等一些[编程](http://www.2cto.com/kf)语言实现扩展的监控指标。这里需要注意的是，自定义的监控指标是没有在Collectd默认的类型[数据库](http://www.2cto.com/database/)里面（/usr/share/collectd/types.db）,所以如果要自定义监控，需要创建一个自己的types.db，然后在collectd的配置文件里面加上

TypesDB "/usr/local/share/types.db.custom"

Github上有个collectd-rabbitmq插件，可以让我们很好的理解collectd的采集机制。

另外，无意间发现一位大神的博客bolg.kankanan.com，里面Collectd相关的文章相当不错。

Influxdb

又一个用go语言写出来的时序数据库神器，广泛用于监控系统的后端存储，对计算大量数据的指标有着不俗的表现。同时提供丰富的查询函数。最重要的是提供collectd的插件，可以让collectd直接将采集到的指标通过udp协议发往数据库。

Grafana

不用多说，炫酷的监控前端数据展示工具。支持多种数据源接入，以及多种插件。

开工

既然选好工具，那就开始动手吧。首先我选择CentOS7系统，添加EPEl源。

1.安装Collectd

|  |  |
| --- | --- |
| 1 | yuminstallcollectdcollectd-lvmcollectd-virt |

collectd 采集普通指标

collectd-lvm 用来采集cinder-volumes指标

collectd-virt 用来采集虚拟机指标

配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | #cat/etc/collectd.conf|grep-v^#|sed'/^$/d'  LoadPluginsyslog  LoadPlugincpu  LoadPlugindf  LoadPlugindisk  LoadPlugininterface  LoadPluginload  LoadPluginlvm  LoadPluginmemory  LoadPluginnetwork  <pluginnetwork>  server"<influxdbip>""25826"    Include"/etc/collectd.d"</influxdbip></pluginnetwork> |

libvirt插件配置

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | cat/etc/collectd.d/libvirt.conf    LoadPluginvirt  <pluginvirt>  Connection"[qemu:///system](qemu://system)"  RefreshInterval60  #Domain"name"  #BlockDevice"name:device"  #InterfaceDevice"name:device"  #IgnoreSelectedfalse  HostnameFormatuuid  #InterfaceFormatname  PluginInstanceFormatuuid  </pluginvirt> |

配好后启动collectd即可

2.安装InfluxDB

|  |  |
| --- | --- |
| 1  2 | #axel-<n20wgethttps://dl.influxdata.com/influxdb/releases/influxdb-1.1.1.x86_64.rpm>  #yumlocalinstall-yinfluxdb-1.1.1.x86\_64.rpm |

配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | /etc/influxdb/influxdb.conf  [[collectd]]  enabled=true  bind-address="10.16.30.47:25826"  database="collectd"  retention-policy=""  typesdb="/usr/share/collectd/types.db" |

3.安装Grafana

|  |  |
| --- | --- |
| 1  2 | #axel-<n20https://grafanarel.s3.amazonaws.com/builds/grafana-4.0.2-1481203731.x86_64.rpm>  #yumlocalinstall-ygrafana-4.0.2-1481203731.x86\_64.rpm |

4.构建OpenStack的Influx索引表

由于通过Collectd采集上来的指标中，没有宿主机AZ，也没有物理机与虚拟的映射关系，更没有租户和虚拟机的信息。这个时候就需要自己动手去构建索引表了。我这里写了一个很Low逼的Shell脚本，来帮助我定时向InfluxDB里面Post最新的OpenStack信息。

脚本.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | #!/bin/bash  influxdb\_url="[http://](NULL)<inluxdbip>:8086"  database="collectd"  write\_data="${influxdb\_url}/write?db=${database}"  log\_file="/var/log/mitaka/mapping\_hosts.log"  log\_time=`date+%Y-%m-%d\%T`  tmp\_file="/tmp/.measurement.tmp"  map\_file="/tmp/.map.tmp"  tenant\_file="/tmp/.tenant\_map"  functiondelete\_mensurement\_hosts(){  echo"$(date+%Y-%m-%d\%T)Deletemeasurementfromdatabasescollectd.">>${log\_file}  curl-Ss-POST${influxdb\_url}/query?db=collectd--data-urlencode"q=DROPmeasurementhosts">>${log\_file}  if[[$?-eq0]];then  echo"$(date+%Y-%m-%d\%T)Deletesuccessfully">>${log\_file}  return0  else  echo"$(date+%Y-%m-%d\%T)Deletefaild,unknownerror.">>${log\_file}  return1  fi  }  functionget\_mapping(){  touch${tmp\_file}  source/root/.keystonerc\_admin  openstackserverlist--all--long-fvalue2>&1>>${tmp\_file}  echo"$(date+%Y-%m-%d\%T)Getmappinghost.">>${log\_file}  novalist--all-tenants|grepvlan|awk-F'|''{print$2$4}'>${map\_file}  foruuidin`cat${tenant\_file}|awk'{print$1}'`;  do  user=`cat${tenant\_file}|grep$uuid|awk'{print$2}'`  sed-i"s/${uuid}/${user}/g"${map\_file}  done  }  functionPost\_measurements\_hosts(){  foruuidin`cat${tmp\_file}|awk'{print$1}'`;  do  availability\_zone=`cat${tmp\_file}|grep-w$uuid|awk'{print$7}'`  instance\_name=`cat${tmp\_file}|grep-w$uuid|awk'{print$2}'`  host=`cat${tmp\_file}|grep-w$uuid|awk'{print$8}'`  ipaddress=`cat${tmp\_file}|grep-w$uuid|awk'{print$6}'`  tenant=`cat${map\_file}|grep-w$uuid|awk'{print$2}'`  echo"$(date+%Y-%m-%d\%T)Post$write\_data$uuid$availability\_zone$instance\_name$host${ipaddress#\*=}${tenant}.">>${log\_file}  curl-Ss-i-XPOST"${write\_data}"--data-binary"hosts,uuid=${uuid},instance=${instance\_name},ip=${ipaddress#\*=},availability\_zone=${availability\_zone},host=${host},tenant=${tenant}value=1">>${log\_file}  done  }  get\_mapping  delete\_mensurement\_hosts  Post\_measurements\_hosts  rm-rf${tmp\_file}  rm-rf${map\_file}</inluxdbip> |

然后实现的效果就如下



现在有了这个表，就可以愉快的在Grafana上创建templating了。

5.绘图

DashBoard

dashboard主要汇总OpenStack的资源使用情况，同时对物理机和虚拟机的使用情况做排序。这样在单位时间内，我就能知道是哪台物理机或虚拟机占用的资源最多。



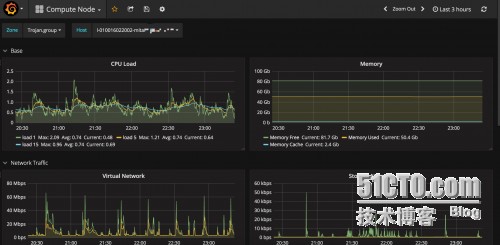
计算节点

templating

计算节点当然需要索引availabilit\_zone和host信息了。所以templating的配置如下

|  |  |
| --- | --- |
| 1  2 | showtagvaluesfromhostswithkey="availability\_zone"  showtagvaluesfromhostswithkey="host"whereavailability\_zone=~/$Zone$/ |

*指标*

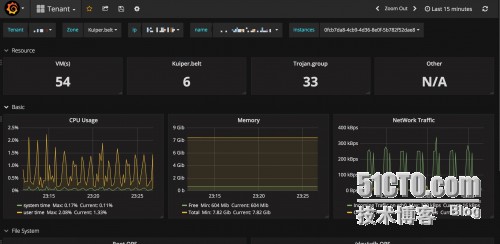


租户信息

*templating*

|  |  |
| --- | --- |
| 1  2  3  4 | showtagvaluesfromhostswithkey="tenant"  showtagvaluesfromhostswithkey="availability\_zone"wheretenant=~/$Tenant$/  showtagvaluesfromhostswithkey="ip"wheretenant=~/$Tenant$/andavailability\_zone=~/$Zone$/  showtagvaluesfromhostswithkey="instance"whereip=~/$ip$/showtagvaluesfromhostswithkey="uuid"whereip=~/$ip$/ |

*指标*



# 2@OpenStack General System Performance dashboard for Grafana | Grafana Labs

https://grafana.com/dashboards/1021

***OpenStack General System Performance***

by[**akrzos**](https://grafana.com/akrzos)

DASHBOARD

Browbeat dashboard "General OpenStack System Performance Dashboard" for viewing Performance Metrics collected from Tripleo deployed OpenStack Clouds.

Last updated: 4 months ago

Downloads: 85

[Overview](https://grafana.com/dashboards/1021)[Revisions](https://grafana.com/dashboards/1021/revisions)

* 

This dashboard is used to view System Performance with "typical" OpenStack Nodes (Controllers, Computes, Deployment Nodes (Undercloud)). It is built for a collectd/graphite metrics environment. It is best to actually deploy the dashboards (and collectd configurations) using Browbeat (https://github.com/openstack/browbeat), however you can use the json in here and by modifying the included collectd configuration.

**Collector Configuration Details**

# Installed by Browbeat Ansible Installer

# Config type: controller

# Interval default is 10s

Interval 10

# Hostname for this machine, if not defined, use gethostname(2) system call

Hostname "overcloud-controller-0"

# Loaded Plugins:

LoadPlugin "logfile"

<Plugin "logfile">

LogLevel "info"

File "/var/log/collectd.log"

Timestamp true

</Plugin>

LoadPlugin write\_graphite

LoadPlugin cpu

LoadPlugin conntrack

LoadPlugin df

LoadPlugin disk

LoadPlugin exec

LoadPlugin interface

LoadPlugin irq

LoadPlugin load

LoadPlugin match\_regex

LoadPlugin memory

LoadPlugin mysql

LoadPlugin numa

LoadPlugin processes

LoadPlugin swap

LoadPlugin tail

LoadPlugin turbostat

LoadPlugin unixsock

LoadPlugin uptime

# Open unix domain socket for collectdctl

<Plugin unixsock>

SocketFile "/var/run/collectd-unixsock"

SocketGroup "collectd"

SocketPerms "0770"

DeleteSocket true

</Plugin>

PreCacheChain "PreCache"

<Chain "PreCache">

<Rule "ignore\_tap">

<Match "regex">

Plugin "^interface$"

PluginInstance "^tap\*"

</Match>

Target "stop"

</Rule>

<Rule "ignore\_interfaces\_q">

<Match "regex">

Plugin "^interface$"

PluginInstance "^q.\*"

</Match>

Target "stop"

</Rule>

Target "return"

</Chain>

# Graphite Host Configuration

<Plugin write\_graphite>

<Carbon>

Host "{{graphite\_host}}"

Port "2003"

Prefix "{{graphite\_prefix}}."

Protocol "tcp"

LogSendErrors true

StoreRates true

AlwaysAppendDS false

EscapeCharacter "\_"

</Carbon>

</Plugin>

<Plugin df>

ValuesPercentage true

</Plugin>

<Plugin disk>

Disk "/^[hsv]d[a-z]+[0-9]?$/"

IgnoreSelected false

</Plugin>

<Plugin mysql>

<Database "overcloud">

Host "localhost"

User "root"

Password "{{mysql\_root\_password.stdout}}"

Socket "/var/lib/mysql/mysql.sock"

InnodbStats true

</Database>

</Plugin>

# (akrzos) Including the version of OpenStack that the process was verified as running after

# OpenStack Installation with a comment at the end of each Process/ProcessMatch statement.

# A Minus before the version means the process was not found in that version. (Ex -9)

<Plugin processes>

# Aodh (OpenStack Installed)

ProcessMatch "aodh-evaluator" "aodh-evaluator" # 9,10

ProcessMatch "aodh-listener" "aodh-listener" # 9,10

ProcessMatch "aodh-notifier" "python.+aodh-notifier" # 9,10

# Ceilometer (OpenStack Installed)

ProcessMatch "ceilometer-agent-central" "python.+ceilometer-agent-central" # -9,-10

ProcessMatch "ceilometer-agent-compute" "python.+ceilometer-agent-compute" # -9,-10

ProcessMatch "ceilometer-agent-notification" "ceilometer-agent-notification" # 9,10

ProcessMatch "ceilometer-alarm-evaluator" "python.+ceilometer-alarm-evaluator" # -9,-10

ProcessMatch "ceilometer-alarm-notifier" "python.+ceilometer-alarm-notifier" # -9,-10

ProcessMatch "ceilometer-api" "python.+ceilometer-api" # 9,10

ProcessMatch "ceilometer-collector" "ceilometer-collector" # 9,10

ProcessMatch "ceilometer-polling" "ceilometer-polling" # 9,10

# Ceph (OpenStack Installed)

ProcessMatch "ceph-mon" "^/usr/bin/ceph-mon" # -9,-10

ProcessMatch "ceph-osd" "^/usr/bin/ceph-osd" # -9,-10

ProcessMatch "diamond" "python.+diamond" # -9,-10

ProcessMatch "salt-minion" "python.+salt-minion" # -9,-10

# Cinder (OpenStack Installed)

ProcessMatch "cinder-api" "python.+cinder-api" # 9,10

ProcessMatch "cinder-scheduler" "python.+cinder-scheduler" # 9,10

ProcessMatch "cinder-volume" "python.+cinder-volume" # 9,10

# Collectd (Browbeat Installed)

ProcessMatch "collectd" "/usr/sbin/collectd"

# Glance (OpenStack Installed)

ProcessMatch "glance-api" "python.+glance-api" # 9,10

ProcessMatch "glance-registry" "python.+glance-registry" # 9,10

# Gnocchi (OpenStack Installed)

ProcessMatch "gnocchi-metricd" "gnocchi-metricd" # 9,10

ProcessMatch "gnocchi-statsd" "python.+gnocchi-statsd" # 9,10

# Everything Else (OpenStack Installed)

# (Processes displayed under "Everything Else" on Grafana Dashboards)

ProcessMatch "dnsmasq" "^dnsmasq.+" # -9,-10

ProcessMatch "haproxy" "/usr/sbin/haproxy.+/etc/haproxy/haproxy.cfg" # 9,10

Process "httpd" # 9,10

Process "memcached" # 9,10

Process "mongod" # 9,10

ProcessMatch "mysqld" "/usr/libexec/mysqld" # 9,10

ProcessMatch "rabbitmq" "/usr/lib64/erlang/erts-.+/bin/beam.smp" # 9,10

Process "redis-server" # 9,10

ProcessMatch "tuskar-api" "python.+tuskar-api" # -9,-10

# Heat (OpenStack Installed)

ProcessMatch "heat-api" "python.+heat-api --config-file" # 9,10

ProcessMatch "heat-api-cfn" "python.+heat-api-cfn" # 9,10

ProcessMatch "heat-api-cloudwatch" "python.+heat-api-cloudwatch" # 9,10

ProcessMatch "heat-engine" "python.+heat-engine" # 9,10

# Keystone (OpenStack Installed)

ProcessMatch "keystone-all" "python.+keystone-all" # -9,-10

ProcessMatch "keystone-admin" "keystone-admin.\*-DFOREGROUND" # 9,10

ProcessMatch "keystone-main" "keystone-main.\*-DFOREGROUND" # 9,10

# Neutron (OpenStack Installed)

ProcessMatch "neutron-dhcp-agent" "python.+neutron-dhcp-agent" # 9,10

ProcessMatch "neutron-l3-agent" "python.+neutron-l3-agent" # 9,10

ProcessMatch "neutron-metadata-agent" "python.+neutron-metadata-agent" # 9,10

ProcessMatch "neutron-ns-metadata-proxy" "python.+neutron-ns-metadata-proxy" # -9,-10

ProcessMatch "neutron-openvswitch-agent" "python.+neutron-openvswitch-agent" # 9,10

ProcessMatch "neutron-server" "python.+neutron-server" # 9,10

# Nova (OpenStack Installed)

ProcessMatch "nova-api" "python.+nova-api" # 9,10

ProcessMatch "nova-cert" "python.+nova-cert" # -9,-10

ProcessMatch "nova-compute" "python.+nova-compute" # -9,-10

ProcessMatch "nova-conductor" "python.+nova-conductor" # 9,10

ProcessMatch "nova-consoleauth" "python.+nova-consoleauth" # 9,10

ProcessMatch "nova-novncproxy" "python.+nova-novncproxy" # 9,10

ProcessMatch "nova-scheduler" "python.+nova-scheduler" # 9,10

# OVS (OpenStack Installed)

ProcessMatch "ovs-vswitchd" "ovs-vswitchd.+openvswitch" # 9,10

ProcessMatch "ovsdb-client" "/bin/ovsdb-client" # 9,10

ProcessMatch "ovsdb-server" "ovsdb-server.+openvswitch" # 9,10

ProcessMatch "ovn-northd" "ovn-northd.+openvswitch" # 9,10

ProcessMatch "ovn-controller" "ovn-controller.+openvswitch" # 9,10

ProcessMatch "ovn-controller-vtep" "ovn-controller-vtep.+openvswitch" # 9,10

# Sahara (OpenStack Installed)

ProcessMatch "sahara-api" "python.+sahara-api" # 9,-10

ProcessMatch "sahara-engine" "python.+sahara-engine" # 9,-10

# Swift (OpenStack Installed)

ProcessMatch "swift-account-auditor" "python.+swift-account-auditor" # 9,10

ProcessMatch "swift-account-reaper" "python.+swift-account-reaper" # 9,10

ProcessMatch "swift-account-replicator" "python.+swift-account-replicator" # 9,10

ProcessMatch "swift-account-server" "python.+swift-account-server" # 9,10

ProcessMatch "swift-container-auditor" "python.+swift-container-auditor" # 9,10

ProcessMatch "swift-container-updater" "python.+swift-container-updater" # 9,10

ProcessMatch "swift-container-replicator" "python.+swift-container-replicator" # 9,10

ProcessMatch "swift-container-server" "python.+swift-container-server" # 9,10

ProcessMatch "swift-object-auditor" "python.+swift-object-auditor" # 9,10

ProcessMatch "swift-object-updater" "python.+swift-object-updater" # 9,10

ProcessMatch "swift-object-replicator" "python.+swift-object-replicator" # 9,10

ProcessMatch "swift-object-server" "python.+swift-object-server" # 9,10

ProcessMatch "swift-proxy-server" "python.+swift-proxy-server" # 9,10

# Pacemaker / Corosync (OpenStack Installed)

ProcessMatch "attrd" "/usr/libexec/pacemaker/attrd" # 9,10

ProcessMatch "cib" "/usr/libexec/pacemaker/cib" # 9,10

Process "corosync" # 9,10

ProcessMatch "crmd" "/usr/libexec/pacemaker/crmd" # 9,10

ProcessMatch "lrmd" "/usr/libexec/pacemaker/lrmd" # 9,10

ProcessMatch "pacemakerd" "/usr/sbin/pacemakerd" # 9,10

ProcessMatch "pcsd" "^/usr/bin/ruby.+/usr/lib/pcsd" # 9,10

ProcessMatch "pengine" "/usr/libexec/pacemaker/pengine" # 9,10

ProcessMatch "stonithd" "/usr/libexec/pacemaker/stonithd" # 9,10

</Plugin>

<Plugin swap>

ReportBytes true

ValuesPercentage true

</Plugin>

# Tail plugin configuration

<Plugin "tail">

<File "/var/log/keystone/keystone.log">

Instance "keystone"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/httpd/keystone\_wsgi\_admin\_error.log">

Instance "keystone-wsgi-admin"

<Match>

Regex ":error"

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/httpd/keystone\_wsgi\_main\_error.log">

Instance "keystone-wsgi-main"

<Match>

Regex ":error"

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/nova/nova-api.log">

Instance "nova-api"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/nova/nova-conductor.log">

Instance "nova-conductor"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/nova/nova-consoleauth.log">

Instance "nova-consoleauth"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/nova/nova-novncproxy.log">

Instance "nova-novncproxy"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/nova/nova-scheduler.log">

Instance "nova-scheduler"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/neutron/dhcp-agent.log">

Instance "neutron-dhcp-agent"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/neutron/l3-agent.log">

Instance "neutron-l3-agent"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/neutron/metadata-agent.log">

Instance "neutron-metadata-agent"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/neutron/openvswitch-agent.log">

Instance "neutron-openvswitch-agent"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/neutron/server.log">

Instance "neutron-server"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/cinder/api.log">

Instance "cinder-api"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/cinder/scheduler.log">

Instance "cinder-scheduler"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/cinder/volume.log">

Instance "cinder-volume"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/glance/api.log">

Instance "glance-api"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/glance/registry.log">

Instance "glance-registry"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

# Swift logs all into the same file

<File "/var/log/swift/swift.log">

Instance "swift"

<Match>

Regex "account-server: ERROR "

DSType "CounterInc"

Type "counter"

Instance "account-server"

</Match>

<Match>

Regex "container-server: ERROR "

DSType "CounterInc"

Type "counter"

Instance "container-server"

</Match>

<Match>

Regex "object-server: ERROR "

DSType "CounterInc"

Type "counter"

Instance "object-server"

</Match>

</File>

<File "/var/log/ceilometer/agent-notification.log">

Instance "ceilometer-agent-notification"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/ceilometer/central.log">

Instance "ceilometer-central"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/ceilometer/collector.log">

Instance "ceilometer-collector"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/httpd/ceilometer\_wsgi\_error.log">

Instance "ceilometer-wsgi-api"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/gnocchi/app.log">

Instance "gnocchi-app"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/gnocchi/metricd.log">

Instance "gnocchi-metricd"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/gnocchi/statsd.log">

Instance "gnocchi-statsd"

<Match>

Regex " ERROR "

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

<File "/var/log/httpd/gnocchi\_wsgi\_error.log">

Instance "gnocchi-wsgi-api"

<Match>

Regex ":error"

DSType "CounterInc"

Type "counter"

Instance "total"

</Match>

</File>

</Plugin>

# Include other collectd configuration files

Include "/etc/collectd.d"

# Grafana - 好脑袋和烂笔头 - 开源中国社区

http://my.oschina.net/guol/blog/515126

最近在调研InfluxDB的时候接触到了Grafana，Grafana是什么？一个类似Kibana的东西，也是对后端的数据进行实时展示，那么Grafana和Kibana有什么区别？在我看来区别不大，不过在大家的日常使用中Kibana是跟着Logstash、ElasticSearch等组件一起使用做日志展示、索引、分析的，造成了一种假象就是Kibana就只有这种用法了，Kibana也可以接入其他数据源的，不过大家最长用的还是展示日志，Grafana是什么呢？该项目你可能没听过，也比较年轻，他一般是和一些时间序列数据库进行配合来展示数据的，例如：Graphite、OpenTSDB、InfluxDB等。下面看看官方是怎么解释Grafana的：

grafana是用于可视化大型测量数据的开源程序，他提供了强大和优雅的方式去创建、共享、浏览数据。dashboard中显示了你不同metric数据源中的数据。

grafana最常用于因特网基础设施和应用分析，但在其他领域也有机会用到，比如：工业传感器、家庭自动化、过程控制等等。

grafana有热插拔控制面板和可扩展的数据源，目前已经支持Graphite、InfluxDB、OpenTSDB、Elasticsearch。

下面我们就练练手，先安装：

    grafana官方已经有打包好的deb和rpm包以及编译好的二进制包，同时grafana还支持Mac OS X、windows等等，下载页面在：[download](http://grafana.org/download/)

    作为第一次尝试，我们选择编译好的二进制包：grafana-2.1.3.linux-x64.tar.gz，后面熟练后可选择deb或者rpm。二进制包解压后，进入其中的conf目录，里面有一个defaults.ini文件，在grafana的conf目录下，把defaults.ini配置文件重命名为custom.ini。进入grafana的bin目录下，有一个grafana-server二进制文件，直接执行./grafana-server web即可启动应用程序。更详细的安装在这里：[here](http://docs.grafana.org/installation/)

grafana和LDAP集成：

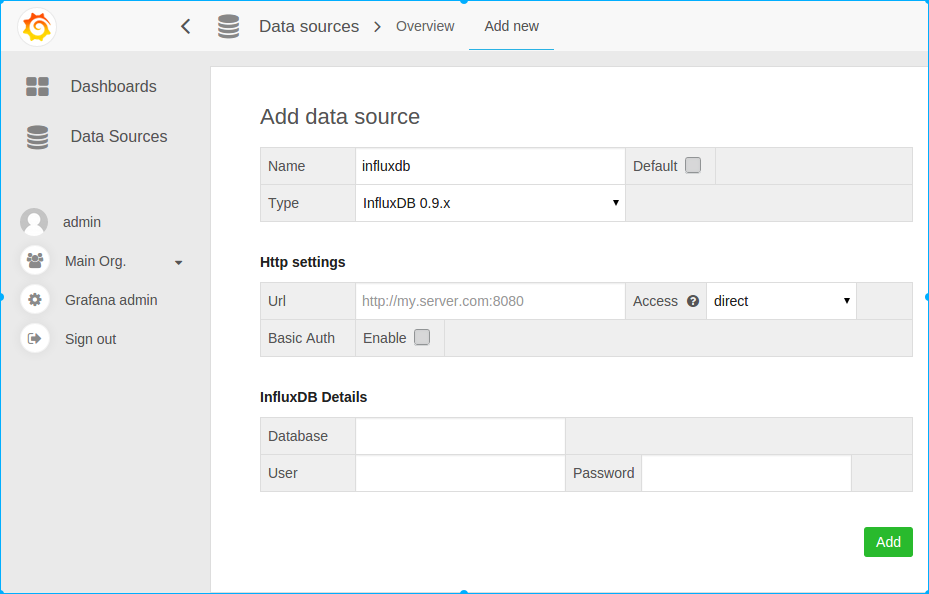
    可惜我们这公司竟没有使用AD/LDAP的习惯，在这里看过程：[here](http://docs.grafana.org/installation/ldap/)

批量安装配置：

    grafana支持puppet、ansible、docker、chef等安装，感觉没啥用就不介绍了

grafana后端数据源配置：

    grafana主要是支持一些时间序列数据库，我们调研的是influxdb，所以就以influxdb为例了，界面上点点就配置好了，主要过程看这里：[here](http://docs.grafana.org/datasources/overview/)



grafana的配置文件：

    grafana后端的配置文件可以是多个以.ini结尾的配置文件，主要从三个配置文件读取配置：默认是$WORKING\_DIR/conf/defaults.ini，其次用户配置是$WORKING\_DIR/conf/custom.ini，用户配置则可以在命令行启动grafana时通过--config参数重新指定配置文件来覆盖。如果你是以deb或者rpm安装的，则默认的配置文件是/etc/grafana/grafana.ini，这个文件是在init.d的启动脚本中通过--config参数指定的。

    所有在配置文件中的配置都可以通过环境变量来覆盖，使用的语法如下：GF\_<SectionName>\_<KeyName>，例如：

[security]

admin\_user = admin

[auth.google]

client\_secret = 0ldS3cretKey

    如果使用环境变量，则是如下：

export GF\_SECURITY\_ADMIN\_USER=true

export GF\_AUTH\_GOOGLE\_CLIENT\_SECRET=newS3cretKey

下面具体看看每个配置段的配置：

app\_mode：应用名称，默认是production

[path]

data：一个grafana用来存储sqlite3、临时文件、回话的地址路径

logs：grafana存储logs的路径

[server]

http\_addr：监听的ip地址，，默认是0.0.0.0

http\_port：监听的端口，默认是3000

protocol：http或者https，，默认是http

domain：这个设置是root\_url的一部分，当你通过浏览器访问grafana时的公开的domian名称，默认是localhost

enforce\_domain：如果主机的header不匹配domian，则跳转到一个正确的domain上，默认是false

root\_url：这是一个web上访问grafana的全路径url，默认是%(protocol)s://%(domain)s:%(http\_port)s/

router\_logging：是否记录web请求日志，默认是false

cert\_file：如果使用https则需要设置

cert\_key：如果使用https则需要设置

[database]

grafana默认需要使用数据库存储用户和dashboard信息，默认使用sqlite3来存储，你也可以换成其他数据库

type：可以是mysql、postgres、sqlite3，默认是sqlite3

path：只是sqlite3需要，定义sqlite3的存储路径

host：只是mysql、postgres需要，默认是127.0.0.1:3306

name：grafana的数据库名称，默认是grafana

user：连接数据库的用户

password：数据库用户的密码

ssl\_mode：只是postgres使用

[security]

admin\_user：grafana默认的admin用户，默认是admin

admin\_password：grafana admin的默认密码，默认是admin

login\_remember\_days：多少天内保持登录状态

secret\_key：保持登录状态的签名

disable\_gravatar：

[users]

allow\_sign\_up：是否允许普通用户登录，如果设置为false，则禁止用户登录，默认是true，则admin可以创建用户，并登录grafana

allow\_org\_create：如果设置为false，则禁止用户创建新组织，默认是true

auto\_assign\_org：当设置为true的时候，会自动的把新增用户增加到id为1的组织中，当设置为false的时候，新建用户的时候会新增一个组织

auto\_assign\_org\_role：新建用户附加的规则，默认是Viewer，还可以是Admin、Editor

[auth.anonymous]

enabled：设置为true，则开启允许匿名访问，默认是false

org\_name：为匿名用户设置组织名称

org\_role：为匿名用户设置的访问规则，默认是Viewer

[auth.github]

针对github项目的，很明显，呵呵

enabled = false

allow\_sign\_up = false

client\_id = some\_id

client\_secret = some\_secret

scopes = user:email

auth\_url = https://github.com/login/oauth/authorize

token\_url = https://github.com/login/oauth/access\_token

api\_url = https://api.github.com/user

team\_ids =

allowed\_domains =

allowed\_organizations =

[auth.google]

针对google app的，呵呵

enabled = false

allow\_sign\_up = false

client\_id = some\_client\_id

client\_secret = some\_client\_secret

scopes = https://www.googleapis.com/auth/userinfo.profile https://www.googleapis.com/auth/userinfo.email

auth\_url = https://accounts.google.com/o/oauth2/auth

token\_url = https://accounts.google.com/o/oauth2/token

api\_url = https://www.googleapis.com/oauth2/v1/userinfo

allowed\_domains =

[auth.basic]

enabled：当设置为true，则http api开启基本认证

[auth.ldap]

enabled：设置为true则开启LDAP认证，默认是false

config\_file：如果开启LDAP，指定LDAP的配置文件/etc/grafana/ldap.toml

[auth.proxy]

允许你在一个HTTP反向代理上进行认证设置

enabled：默认是false

header\_name：默认是X-WEBAUTH-USER

header\_property：默认是个名称username

auto\_sign\_up：默认是true。开启自动注册，如果用户在grafana DB中不存在

[analytics]

reporting\_enabled：如果设置为true，则会发送匿名使用分析到stats.grafana.org，主要用于跟踪允许实例、版本、dashboard、错误统计。默认是true

google\_analytics\_ua\_id：使用GA进行分析，填写你的GA ID即可

[dashboards.json]

如果你有一个系统自动产生json格式的dashboard，则可以开启这个特性试试

enabled：默认是false

path：一个全路径用来包含你的json dashboard，默认是/var/lib/grafana/dashboards

[session]

provider：默认是file，值还可以是memory、mysql、postgres

provider\_config：这个值的配置由provider的设置来确定，如果provider是file，则是data/xxxx路径类型，如果provider是mysql，则是user:password@tcp(127.0.0.1:3306)/database\_name，如果provider是postgres，则是user=a password=b host=localhost port=5432 dbname=c sslmode=disable

cookie\_name：grafana的cookie名称

cookie\_secure：如果设置为true，则grafana依赖https，默认是false

session\_life\_time：session过期时间，默认是86400秒，24小时

以下是官方文档没有，配置文件中有的

[smtp]

enabled = false

host = localhost:25

user =

password =

cert\_file =

key\_file =

skip\_verify = false

from\_address = admin@grafana.localhost

[emails]

welcome\_email\_on\_sign\_up = false

templates\_pattern = emails/\*.html

[log]

mode：可以是console、file，默认是console、file，也可以设置多个，用逗号隔开

buffer\_len：channel的buffer长度，默认是10000

level：可以是"Trace", "Debug", "Info", "Warn", "Error", "Critical"，默认是info

[log.console]

level：设置级别

[log.file]

level：设置级别

log\_rotate：是否开启自动轮转

max\_lines：单个日志文件的最大行数，默认是1000000

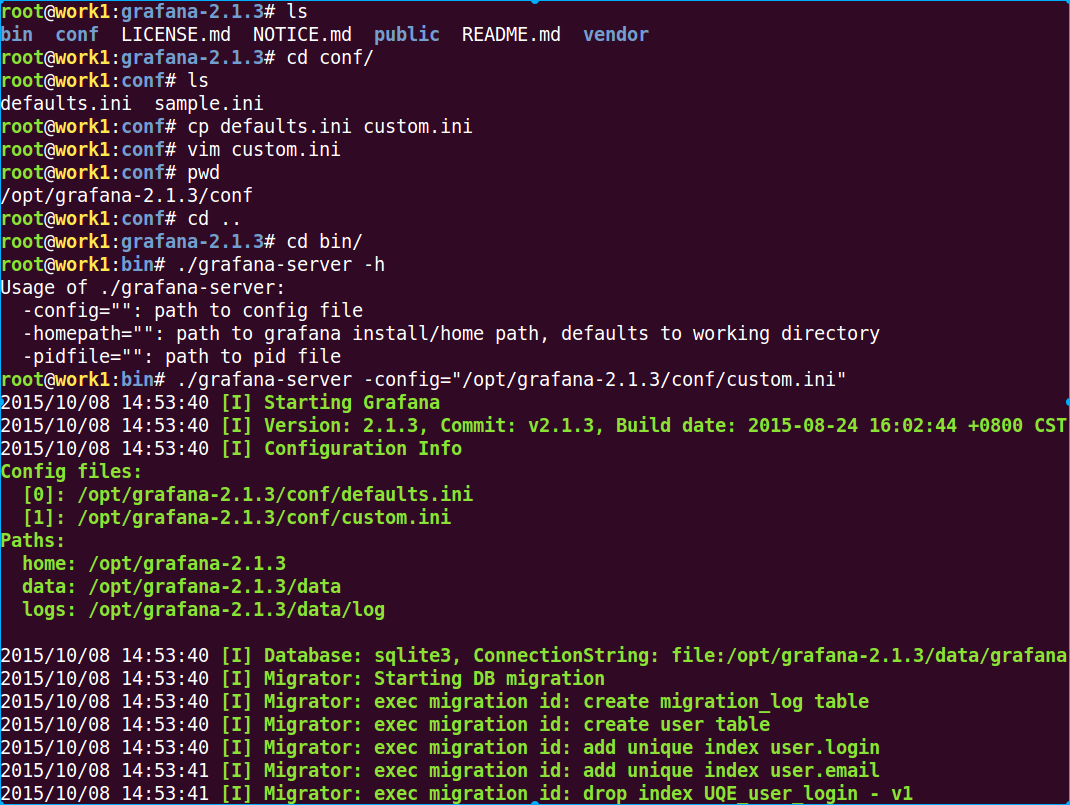
max\_lines\_shift：单个日志文件的最大大小，默认是28，表示256MB

daily\_rotate：每天是否进行日志轮转，默认是true

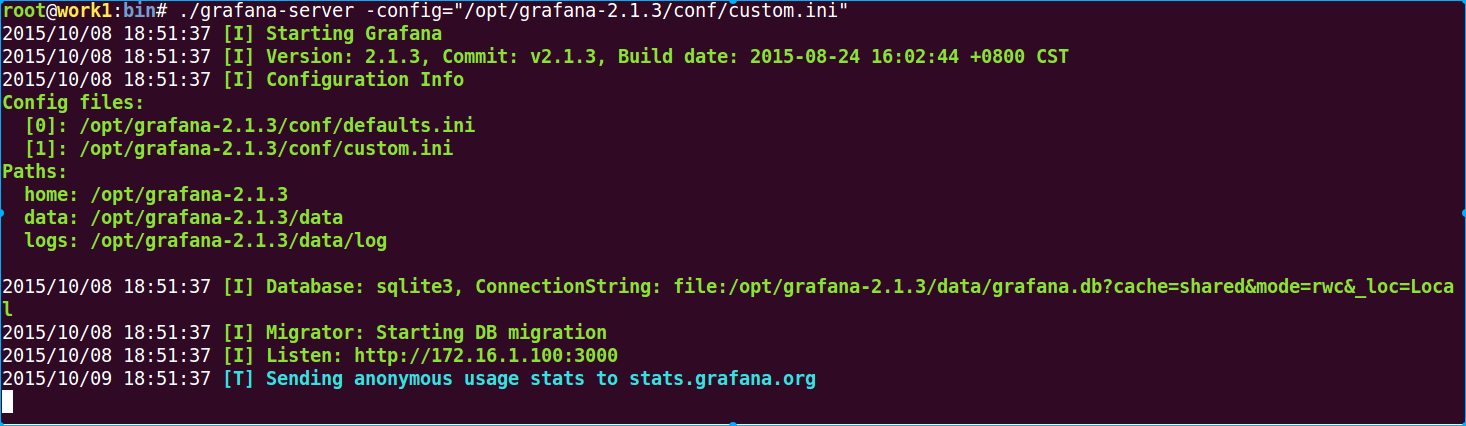
max\_days：日志过期时间，默认是7,7天后删除

启动grafana：

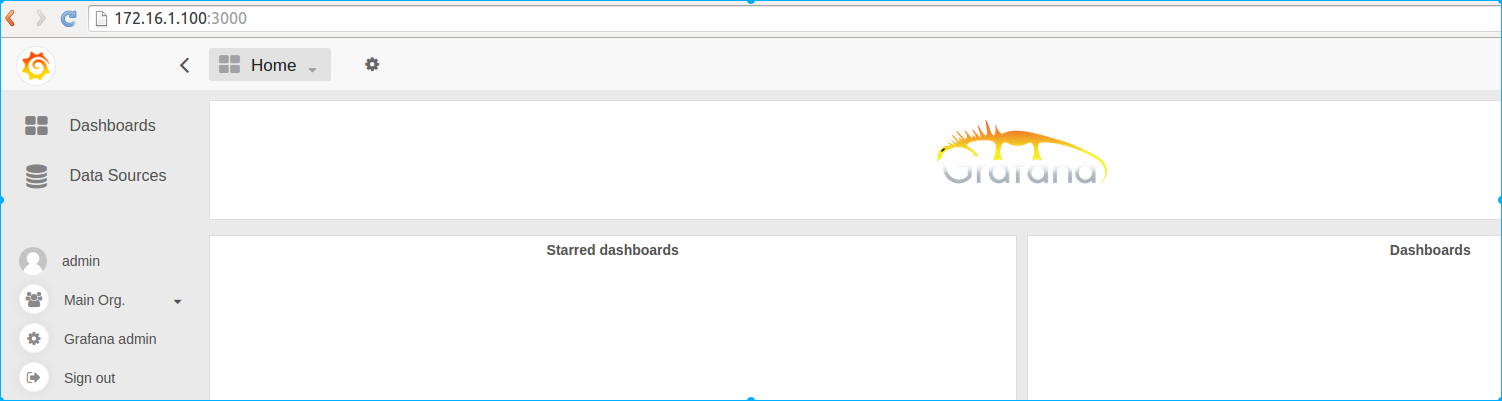
./grafana-server -config="/opt/grafana-2.1.3/conf/custom.ini"

    下面的就是初始化过程了，会有大片的输出，再次重启grafana的时候则不会进行初始化：



下面访问grafana看看：



© 著作权归作者所有

分类：[Linux操作技巧](http://my.oschina.net/guol/blog?catalog=129500)

字数：1980

# 从kibana迁移到grafana作为elasticsearch的前端展现 | 峰云就她了

http://xiaorui.cc/2016/01/07/%E4%BB%8Ekibana%E8%BF%81%E7%A7%BB%E5%88%B0grafana%E4%BD%9C%E4%B8%BAelasticsearch%E7%9A%84%E5%89%8D%E7%AB%AF%E5%B1%95%E7%8E%B0/

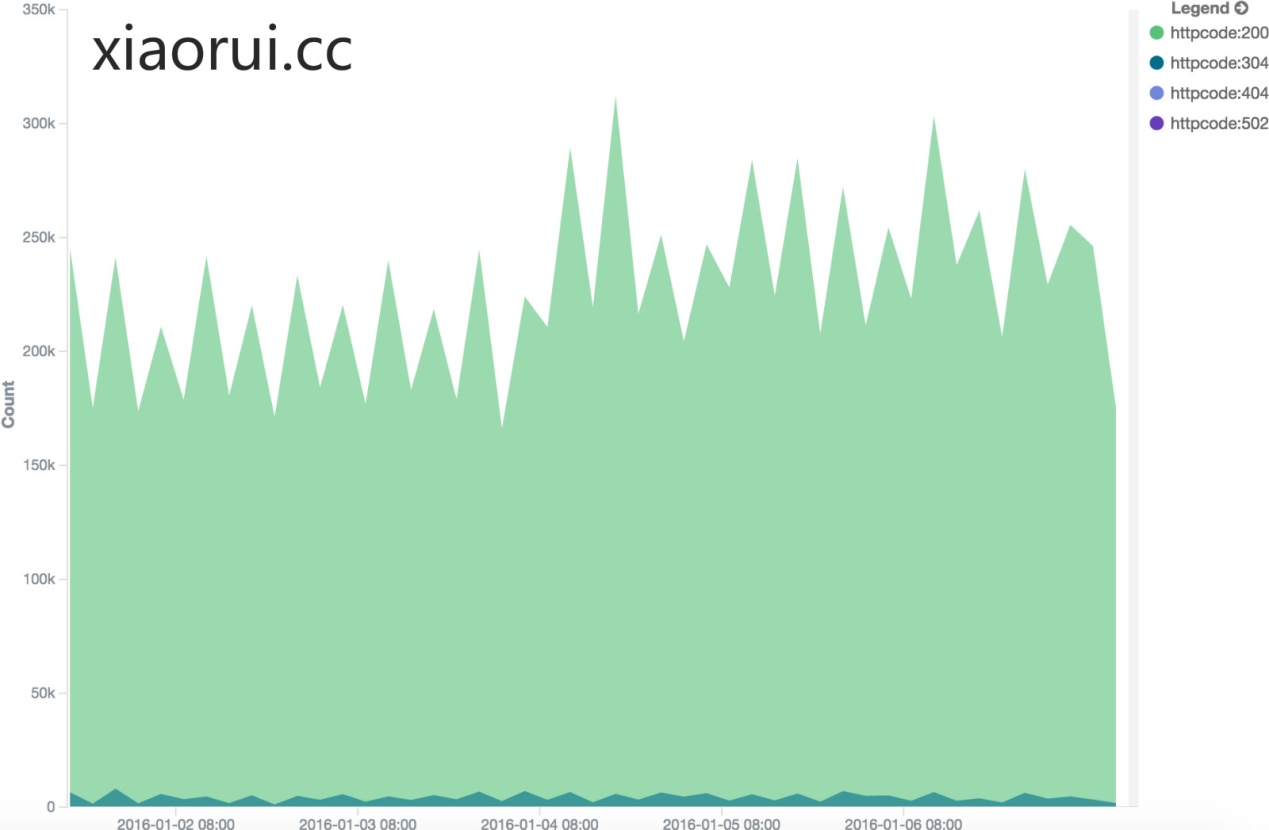
今天说说，我们为什么会选择从kibana4迁移到grafana，并选用grafana作为elasticsearch的图表展现工具。 文章中关于kinaba和grafana的对比会有些片面，勿喷.

最一开始使用kibana ElasticSearch的组合是为了集中式收集应用及系统日志.  后来由于业务方面的原因，现在各个业务的多数模块也选择依赖elasticsearch。除此之外，现在监控的数据也从opentsdb hbase迁移到elasticsearch里面。 目的 ？ 只为更好的实时聚合计算，并快速的展现业务报表，展现监控数据。

文章写的不是很严谨，欢迎来喷，另外该文后续有更新的，请到原文地址查看更新.

<http://xiaorui.cc/2016/01/07/%E4%BB%8Ekibana%E8%BF%81%E7%A7%BB%E5%88%B0grafana%E4%BD%9C%E4%B8%BAelasticsearch%E7%9A%84%E5%89%8D%E7%AB%AF%E5%B1%95%E7%8E%B0/>

先说下kibana方面的事，kibana往往在展现一条数据的时候效果是完美的，尤其是kibana4那种清淡的绿色让人心旷神怡。 但很多时候我们要做多维度数据图表展现, 这地方kibna貌似没有做图表样式的优化。当很多条数据拥挤在一起时，很难区分出每个点的数值，换句话说很不直观。  另外kibana更加适合日志类型的展现， 虽然他也可以kv结构，但配置起来有些麻烦.



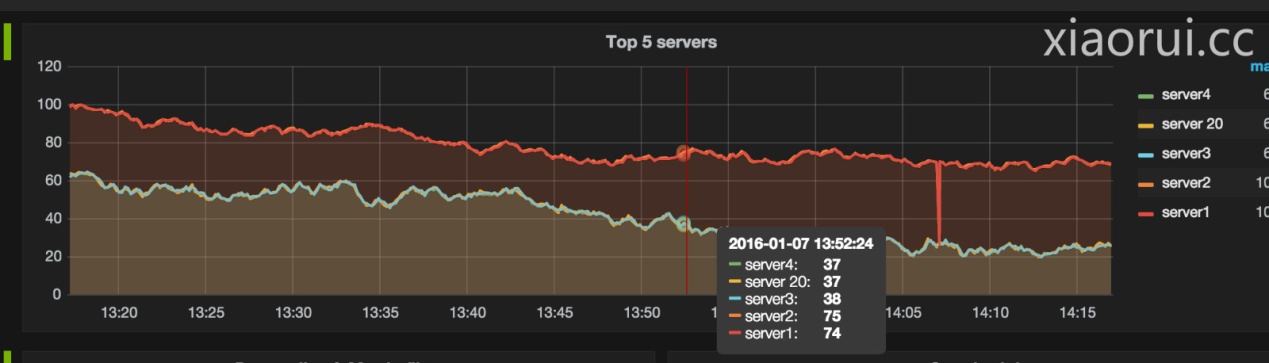
另外kibana没有管理的api,  只能点来点去.  在kibana4.x的版本里集成了一个node服务端，但就最新的版本来说，他只是被当成一个静态服务器使用，没有更多的动态功能，比如权限管理。

去年写过一个文章专门描述kibana批量操作的问题. 当时的需求是有很多的图表及dashboard面板.  如果靠着手动点击不怎么现实.  后来发现Kibana的数据都存在elasticsearch里，

所以写了个较完整的python模块去直接操作elasticsearch，其实就是个.kibana的配置文件.  因为api跟公司的业务耦合一起了，所以暂不方便开源出来。 kibana es批量操作，<http://xiaorui.cc/?p=1570> 有兴趣看看.

kibana没有权限管理，用户管理。  大多数人都是使用nginx做为kibana的基本密码认证（HTTP Auth Basic），这样做也能保证一定的安全，但毕竟不和规矩.

刚才说的kibana的槽点, 在grafana里是可以解决的。对比kibana4来说，grafana支持更多数据源.  grafana以前的1.x版本就支持不少的时序数据库，比如最风光的influxdb，适合大集群大存储的opentsdb， 单机存储的graphite . 后有老外把zabbix扩展到grafana里。现在新版的grafana也支持elasticsearch了。   感叹，elasticsearch是越来越火了，功能也越来越丰富.



grafana官方有不少语言的控制api, 只是没有python的。 在github中找到一个老外开源的grafana python api , grafana\_api\_client。  grafana docs有详细的api使用文档，有兴趣的朋友看看.

Python

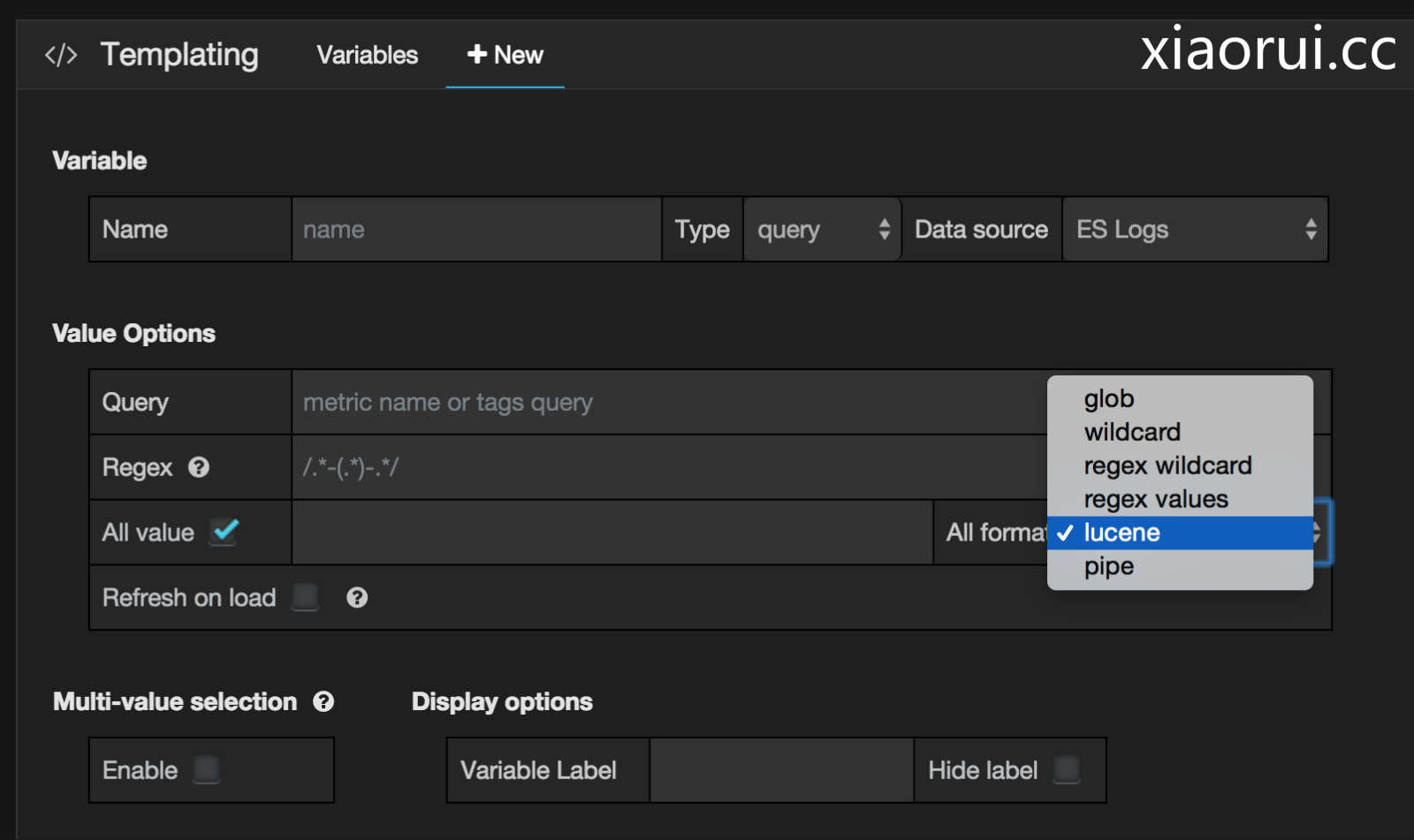
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | from grafana\_api\_client import GrafanaClient  client = GrafanaClient(("admin", "admin"), host="127.0.0.1", port=3000) # or, alternatively:  client = GrafanaClient("yourapikey", host="127.0.0.1", port=3000)  client.org()  #{"id":1,"name":"Main Org."}  client.org.replace(name="Your Org Ltd.")  #{"id":1,"name":"Your Org Ltd."}  client.dashboards.db.create(dashboard={...}, overwrite=False)  #{"dashboard": {...}, "overwrite": False} |

kibana vs grafana功能对比:

单纯的日志，我还是推荐大家把日志报表放在elk （ kibana ）里面的，因为kibana的模板有针对日志的search语法，有的query\_string ,match 全文匹配。 如果kibana本身的搜索需求不能满足你，你可以使用JSON Input来实现自定义的搜索语法。 grafana用的则都是类似term这样的精确匹配.

监控方面的数据，尤其是那种metrics监控类型数据，非常适合用Elasticsearch Grafana组合.  题外说下，对于监控数据的落地我们走了不少的坑，从rrdtool, mysql, Influxdb, OpenTsdb到最后的Elasticsearch，其中的db选型失败也有我们自身能力的原因.  Opentsdb没有想象中的那么好.

修正上面的说法，grafana的新增图表里面，会有elasticsearch和ES LOG两项. 对于metric names字段也是有正则匹配的模式，对于具体的value是可以模糊匹配，正则匹配，lucene全文索引匹配规则.



原谅我桀骜不羁说kibana的不是，我这也是因爱生恨.  我用过kibana的好多版本,  我曾经在Kibana Issue问过他们为什么不把kibana的node.js后端做的丰富一点，作者给我的答复是, 他们本来就不想做太复杂，只是想单纯的做前端图表显示.

END.

问: 不知道监控数据存储为什么最后选择的是Elasticsearch 不是Influxdb

答: 首先我没有否定influxdb的意思. 我曾经多方面测试过influxdb可能会存在的瓶颈，当他的数据在超过几百G的时候，绕开cache做随机range聚合查询会发现性能有明显的下滑，有些难以忍受.

对Python及运维开发感兴趣的朋友可以加QQ群 ： 478476595 !!!   
{ 2000人qq大群内有各厂大牛，常组织线上分享及沙龙，对高性能及分布式场景感兴趣同学欢迎加入该QQ群 }

# 新一代监控平台整合telegraf、influxdb、garafana - 康建华 - 51CTO技术博客

http://michaelkang.blog.51cto.com/1553154/1759877

随着容器时代到来，公司对devops等概念理解越来越深刻，业务系统架构也在不断向微服务架构调整。业务系统也不断的提供更多的api接口，实现方便灵活的调用。在这种业务背景下，感觉业务系统的监控系统（zabbix、cacti）等，略显笨重，在使用灵活度上感觉也略显欠佳。

    那么有没有新的监控系统能够在满足监控的需求的基础上，让使用变得更灵活、调用更加方便那？

答案是肯定的，有，个人查了一些资料，简单整理一下，下面进行简单描述：

监控系统一般分几块：

数据采集

数据存储

数据展示

告警 （后期探讨）

对应找到服务如下：

采集数据（telegraf）-> 存储数据（InfluxDB) -> 显示数据（Grafana）

下面对各个服务的特点进行介绍：

1：telegraf

    Telegraf 是一个用 Go 编写的代理程序，可收集系统和服务的统计数据，并写入到 InfluxDB 数据库。

Telegraf 具有内存占用小的特点，通过插件系统开发人员可轻松添加支持其他服务的扩展。

个人感受：

1：监控插件满足主流监控需求；

2：添加监控项方式简单、灵活；

3：监控项模块化，方便自动程序调用；

官网参考地址

<https://docs.influxdata.com/telegraf/v0.11/>

2：influxdb

    InfluxDB 是一个开源分布式时序、事件和指标数据库。使用 Go 语言编写，无需外部依赖。其设计目标是实现分布式和水平伸缩扩展。

1、它有三大特性：

1. Time Series （时间序列）：你可以使用与时间有关的相关函数（如最大，最小，求和等）

2. Metrics（度量）：你可以实时对大量数据进行计算

3. Eevents（事件）：它支持任意的事件数据

2、特点

schemaless(无结构)，可以是任意数量的列

Scalable

min, max, sum, count, mean, median 一系列函数，方便统计

Native HTTP API, 内置http支持，使用http读写

Powerful Query Language 类似sql

Built-in Explorer 自带管理工具

3、API

InfluxDB 支持两种api方式

HTTP API

Protobuf API

个人感受：

1：数据为为监控而生；

2：方便扩容；

3：管理、使用都很爽；

3：grafana

    Grafana是一个纯粹的html/js应用，访问InfluxDB时不会有跨域访问的限制。只要配置好数据源为InfluxDB之后就可以，剩下的工作就是配置图表。Grafana 功能非常强大。

使用ElasticsSearch保存DashBoard的定义文件，也可以Export出JSON文件(Save ->Advanced->Export Schema)，然后上传回它的/app/dashboards目录。

个人感受：

1：gafana 太强了，展示监控数据就是小菜一碟；

2：丰富的数据源接口，各种数据都能接入，在gafana进行展示；

3：丰富的API接口，方便自动化程序调用；

4：监控dashboard 导入 、导出，这个很棒，做好一个比较满意的展示面板，导出后主要修改一下里面的IP等信息，通过导入，其它主机的展示全部搞定。

以上仅仅是个人总结的一点，优缺点还需要大家各自评点，下面是我的安装部署的一些资料，提供参考，不足之处请指正。

[**telegraf安装部署详解**](http://michaelkang.blog.51cto.com/1553154/1759864)

[**influxdb 安装配置详解**](http://michaelkang.blog.51cto.com/1553154/1759865)

[**grafana 安装配置详解**](http://michaelkang.blog.51cto.com/1553154/1759867)

# Collectd+Influxdb+Grafana打造监控系统 - 运维之路

http://www.361way.com/collectd-influxdb-grafana/5296.html

[InfluxDB](https://www.influxdata.com/downloads/) 是 Go 语言开发的一个开源分布式时序数据库，非常适合存储指标、事件、分析等数据（同类型的数据库还有：OpenTSDB、KairosDB、MonnetDB、druid）；[collectd](https://collectd.org/) 是C 语言写的一个系统性能采集工具；[Grafana](http://grafana.org/)是纯 Javascript 开发的前端工具，用于访问 InfluxDB，自定义报表、显示图表等。三者结合是这样的：采集数据（collectd）-> 存储数据（InfluxDB) -> 显示数据（Grafana）。

一、软件安装

influxdb和grafana官方提供的都有rpm包，下载安装即可，这里以最新版本的为例：

1. wget https://dl.influxdata.com/influxdb/releases/influxdb-1.1.0.x86\_64.rpm
2. rpm -ivh influxdb-1.1.0.x86\_64.rpm
3. wget https://grafanarel.s3.amazonaws.com/builds/grafana-3.1.1-1470047149.x86\_64.rpm
4. rpm -ivh grafana-3.1.1-1470047149.x86\_64.rpm

collectd软件官方提供的rpm包感觉有点老了，这里给主机上配置上epel 源，epel 源里可以直接yum  -y install collectd 安装。

二、整合配置

1、collectd配置

collectd有许多模块默认是注释的，可以根据自己需要开启即可。当然还有一些扩展插件如mysql、drdb、ceph、amqp、rrdtool、virt等的，也可以根据自己需要安装相应的rpm包即可。由于这些和本篇关系不大，这里主要还是说如何将数据发往influxdb 。编辑collectd的配置文件，开启如下内容：

1. # vim /etc/collectd/collectd.conf
2. ...
3. LoadPlugin network
4. ...
5. <Plugin network>
6. Server "10.212.52.253" "25826"
7. </Plugin>

这里的server地址是influxdb服务器的地址和端口，如是需要influxdb主机开启了认证，也可以使用如下方式进行配置：

1. <Server "10.212.52.253" "25826">
2. SecurityLevel Encrypt
3. Username "user"
4. Password "secret"
5. Interface "eth0"
6. ResolveInterval 14400
7. </Server>

2、influxdb配置

InfluxDB 现在自带一个 collectd 插件来获取 collectd 客户端发来的数据，如果是0.8.4 以前的版本只能通过 influxdb-collectd-proxy 这样的第三方程序来连接 collectd 和 InfluxDB 。具体修改其配置开启如下部分即可：

1. # vim /etc/influxdb/influxdb.conf
2. [[collectd]]
3. enabled = true
4. bind-address = ":25826"
5. database = "collectd"

这里的数据库无需在influxdb里事先创建好，collectd在向influxdb发送数据的时候会自动创建该数据库。这里重启influxdb服务，会发现其会开启一个UDP的25826端口用来接收数据 。

三、grafana配置

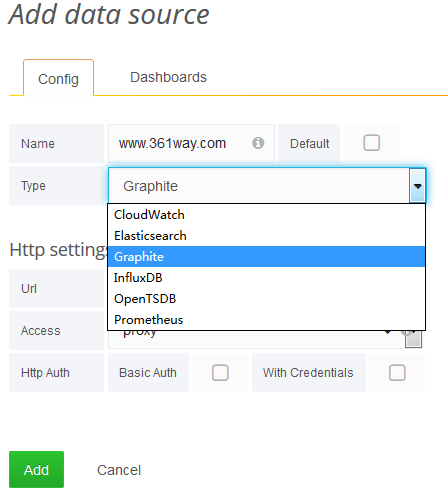
按如下操作，启动三个服务。通过http://IP:3000可以进入grafana的配置界面。

1. systemctl enable collectd
2. systemctl start collectd
3. systemctl enable influxdb
4. systemctl start influxdb
5. systemctl enable grafana
6. systemctl start grafana

确认influxdb中有数据写入后，开始登录grafana界面自定义图形配置 。

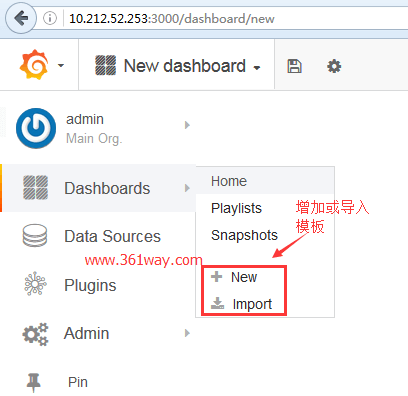
1、配置数据源

grafana不仅支持influxdb，还支持zabbix等其他数据源的导入。具体可以通过data source进行配置，见下图：



2、自定义dashboard或导入

通过dashboards －－－new create或import模板，如下图：



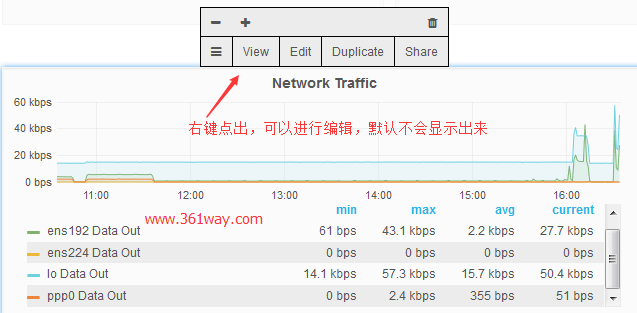
导入的模板为json格式 ，官方也提代了一些常用模板：<https://grafana.net/dashboards>，可以直接下载import导入使用。如果使用新建模板，需要使用add row设计界面、图形及influxdb查询的select语句。

3、自定义模板

以流量为例，对应的sql 模板语句如下：

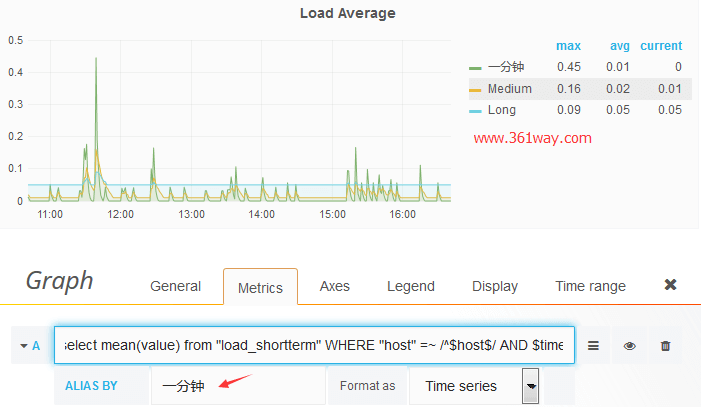
1. SELECT derivative(mean(value), 1s)\*8 FROM "interface\_rx" WHERE "type"='if\_octets' AND "host" =~ /^$host$/ AND $timeFilter GROUP BY time($interval), instance fill(null)
2. SELECT derivative(mean(value), 1s)\*8 FROM "interface\_tx" WHERE "type"='if\_octets' AND "host" =~ /^$host$/ AND $timeFilter GROUP BY time($interval), instance fill(null)

这里derivative和mean为数学计算函数，具体可以参看grafana官方文档。上面的SQL语句分别查询了了各个接口收、发的网络并进行展示，对应显示的图形界面如下：



4、别名设置

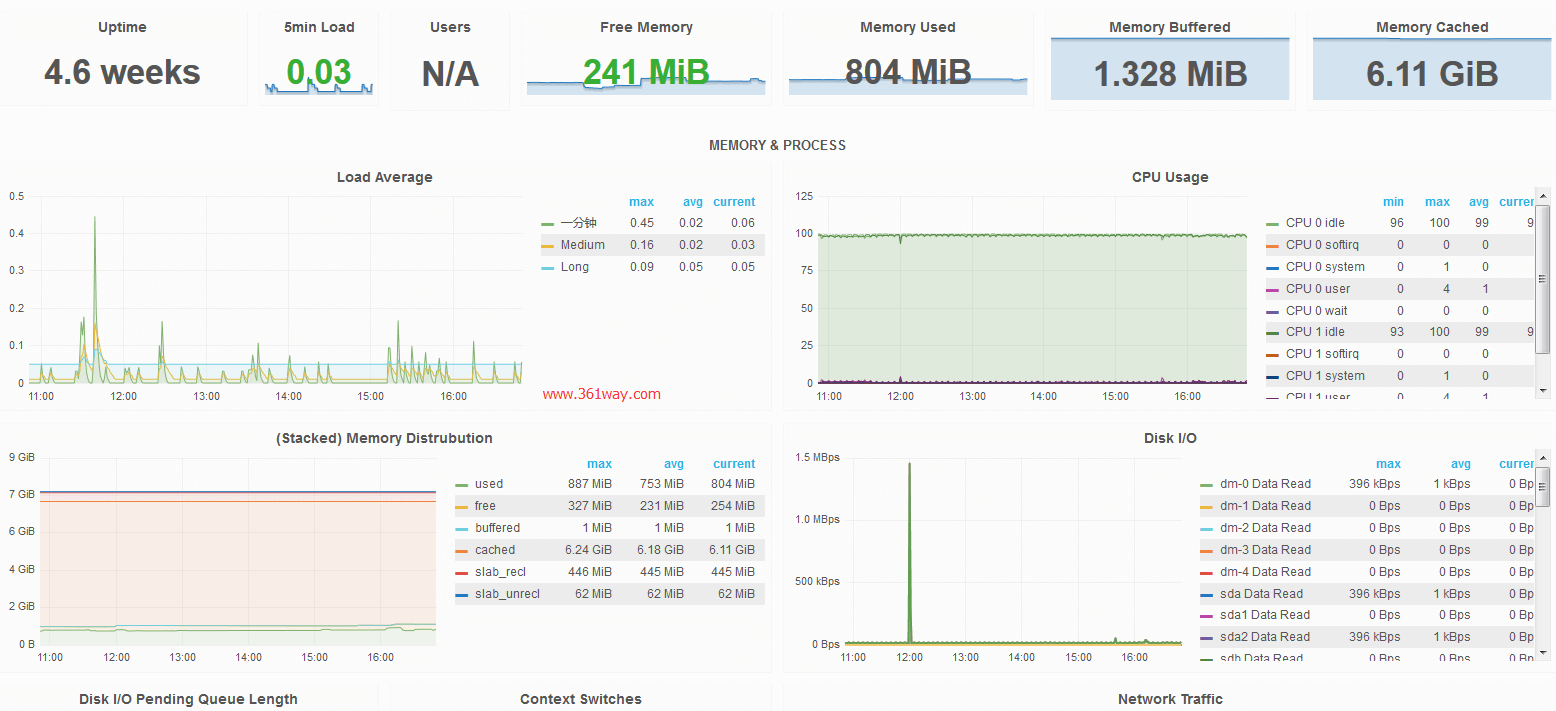
默认显示出的名称都是influxdb对应的字段名称，由于influxdb相对应的字段大都是英文显示的，显得不够友好，也可以通过别名设置显示中文标记，如下图所示，配置1\5\15分钟的load：



同样，出图的样式，如柱状图、点图、线图等都可以在display里进行设置和选取。也可以将多个指标值放在一起。如左边Y轴显示的是IO使用的百分比，右边Y轴用于显示具体的读写速度 xxx KB/S 。

四、总结

在总结之前，先给一个整体显示的图看下效果(点击图片查看大图)：

[[](http://www.361way.com/wp-content/uploads/2016/11/grafana-system.png)](http://www.361way.com/wp-content/uploads/2016/11/grafana-system.png)

三者在监控和展示方面的结合是比较强大的，而且也可以自由定制多个dashboard界面，而且grafana可以支持多数据源的导入 。出图界面也比较漂亮快捷 。其还支持通过[grafana-influx-dashboard](https://github.com/anryko/grafana-influx-dashboard)插件将多台主机的某个指标在一屏展示出来。但是三个结合起来有一个不完美的地方，就是无法及时进行告警输出，虽然influxdb公司有kapacitor组件可以进行查询并告警，不过感觉TICKscript与我们经常接触到的语言差距比较大，使用起来感觉不够顺手。

参考文档：

[grafana官方手册](http://docs.grafana.org/datasources/influxdb/)

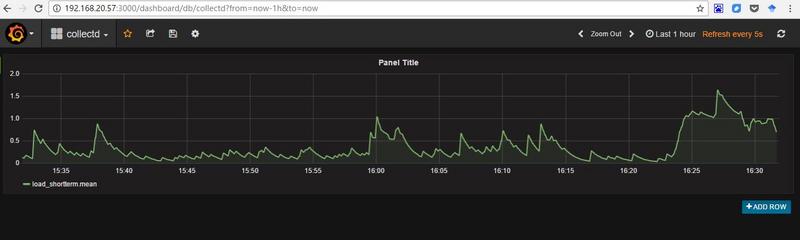
[利用collectd, influxdb和grafana进行简单的负载预警](https://segmentfault.com/a/1190000007527469)

# 2@利用collectd, influxdb和grafana进行简单的负载预警 - mrchenatu - SegmentFault

https://segmentfault.com/a/1190000007527469

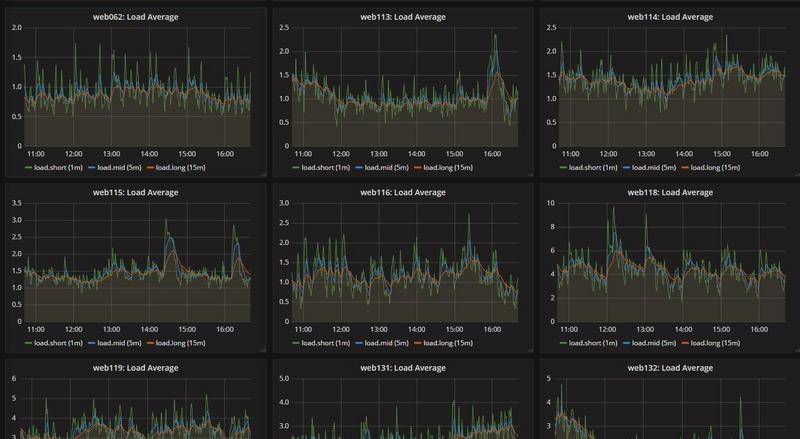
本文仅仅是对负载预警的简单尝试，能够预测的场景也比较有限，但是作为预测工作的开始已经比较能够说明问题了。

利用collectd, influxdb和grafana进行监控系统搭建可以参考这篇文章[Monitoring hosts with CollectD, InfluxDB and Grafana](http://jansipke.nl/monitoring-hosts-with-collectd-influxdb-and-grafana/) grafana的操作比nagios和cacti真的友好很多，可定制的能力也强很多。



虽然grafana有一定的报警能力（在grafana4.0版本之后,alert模块直接继承进来，所以推荐4.0的版本），但是能够提前十几分钟对于集群负载超标进行预警，一直是我们的一个小目标。所以在这里，我们就开始为这个小目标做了一点小努力。

在实际运行中，我们安装了这个[小插件](https://github.com/anryko/grafana-influx-dashboard)。它能够对应用负载进行批量的显示，节约了好多体力活。

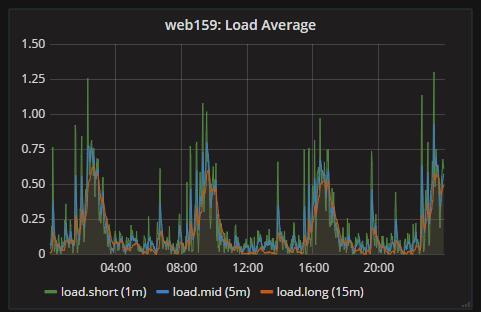


通过分析，我们可以看到以下几种负载

简单趋势型



周期型



规律不明显型

而对于周期型我们也可以看到在短期是可以看到一定趋势的。

在这里我们仅采用线性回归对简单趋势进行预测。这点对于python很好实现

**from** datetime **import** datetime

**from** influxdb **import** DataFrameClient

**import** numpy **as** np

**from** scipy **import** stats

**if** \_\_name\_\_ == "\_\_main\_\_":

host = 'localhost'

port = 8086

user = 'root'

password = 'root'

dbname = 'collectd'

client = DataFrameClient(host, port, user, password, dbname)

print("Create pandas DataFrame")

start\_time = datetime(2016, 11, 17, 3,10).timestamp()

end\_time = datetime(2016, 11, 17, 8,50).timestamp()

query = "select \* from load\_midterm where host='web153' and time > " + str(int(start\_time)) + "s and time < " \

+ str(int(end\_time)) + "s"

df = client.query(query)

slope, intercept, r\_value, p\_value, std\_err = stats.linregress(range(df['load\_midterm'].shape[0]), df['load\_midterm']['value'].values)

print(slope)

print(intercept)

print(r\_value)

print(p\_value)

在这里我们用的是统计模块得到斜率和截距。需要注意的是，我们还要对R2，p值以及方差进行评价。一般来说r^2 >0.7, p<0.05, std\_err<0.1

在这里，为什么没有采用一些更复杂的机器学习方法呢？

无论采用什么模型，关键是要提取一些关键feature。我观察过简单的应用服务器的其他指标，包括tcp连接数，cpu，磁盘io等，与load的协同性比较高，很难成为15min后load的先导预测feature。如果从服务器集群的整体角度去找feature的话，也许是可以做的，未来会关注这块。

在具体实现上，后端基于python flask编写一个预测服务器，而前端开发可以基于grafana的clock插件进行开发:[clock-panel](https://github.com/grafana/clock-panel)

* [2016年11月18日发布](https://segmentfault.com/a/1190000007527469)

* [更多](javascript:void(0);)

# Monitoring hosts with CollectD, InfluxDB and Grafana - jansipke.nl

http://jansipke.nl/monitoring-hosts-with-collectd-influxdb-and-grafana/

In this article we will install a monitoring solution on CentOS 7 consisting of [CollectD](https://collectd.org/), [InfluxDB](https://influxdb.com/)and [Grafana](http://grafana.org/).

The first step is to install CollectD:

*yum -y install epel-release*

*yum -y install collectd*

At the time of writing, this resulted in the installation of CollectD version 5.5.0. If you are using an older version of CentOS, make sure that you do not install version 4 or lower.

The configuration can be tweaked to your liking, but should contain at least something like this in /etc/collectd.conf:

*FQDNLookup true*

*BaseDir "/var/lib/collectd"*

*PIDFile "/var/run/collectd.pid"*

*PluginDir "/usr/lib64/collectd"*

*TypesDB "/usr/share/collectd/types.db"*

*LoadPlugin syslog*

*LoadPlugin interface*

*LoadPlugin load*

*LoadPlugin network*

*<Plugin interface>*

*Interface "eth0"*

*IgnoreSelected false*

*</Plugin>*

*<Plugin load>*

*ReportRelative true*

*</Plugin>*

*<Plugin network>*

*Server "127.0.0.1" "25826"*

*</Plugin>*

The most important part is the “network” plugin, where we define that the measurements should be sent to 127.0.0.1 on port 25826. We will instruct InfluxDB shortly to listen on that port for incoming packets from CollectD.

Now start the daemon and make sure that it starts at boot as well:

*systemctl start collectd.service*

*systemctl enable collectd.service*

Installation of InfluxDB can be achieved like this:

*yum -y install http://influxdb.s3.amazonaws.com/influxdb-0.9.4.2-1.x86\_64.rpm*

Edit the InfluxDB configuration file /etc/opt/influxdb/influxdb.conf and change the lines near the[collectd] heading as follows:

*[collectd]*

*enabled = true*

*bind-address = "127.0.0.1:25826"*

*database = "collectd"*

*typesdb = "/usr/share/collectd/types.db"*

And then start the daemon:

*systemctl start influxdb.service*

You can now check if the measurements from CollectD are received by InfluxDB as follows:

*# /opt/influxdb/influx*

*Connected to http://localhost:8086 version 0.9.4.2*

*InfluxDB shell 0.9.4.2*

*> use collectd*

*Using database collectd*

*> show measurements*

*name: measurements*

*------------------*

*name*

*interface\_rx*

*interface\_tx*

*load\_longterm*

*load\_midterm*

*load\_shortterm*

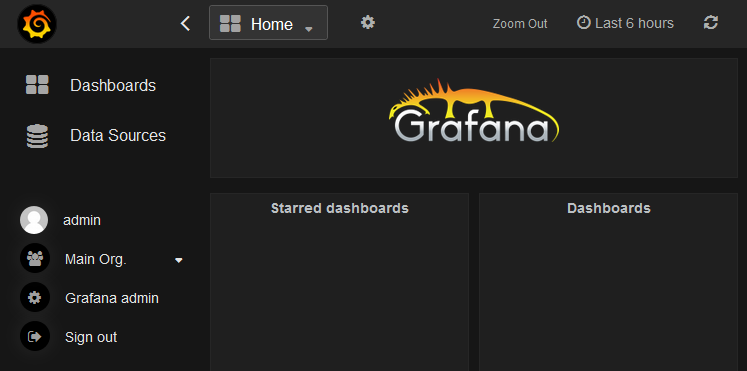
The final step is installing and starting Grafana:

*yum -y install https://grafanarel.s3.amazonaws.com/builds/grafana-2.5.0-1.x86\_64.rpm*

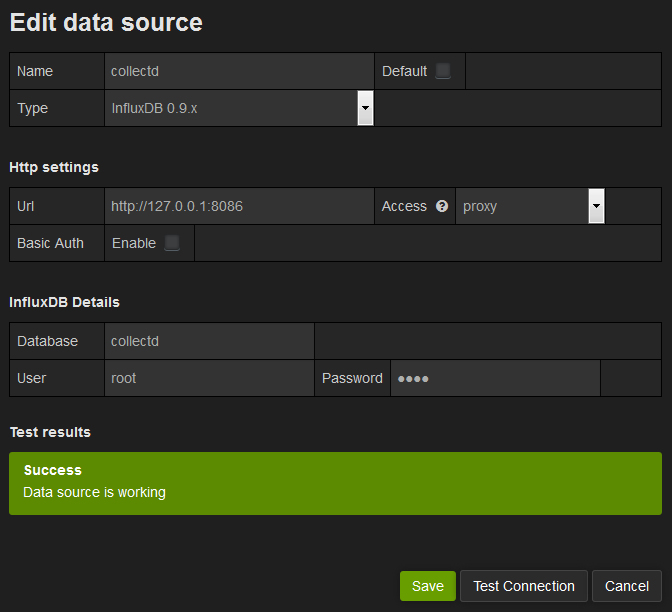
*systemctl start grafana-server.service*

*systemctl enable grafana-server.service*

Start a webbrowser and navigate to http://the\_host\_running\_grafana:3000. Login with username admin and password admin. You will be presented with the following page:

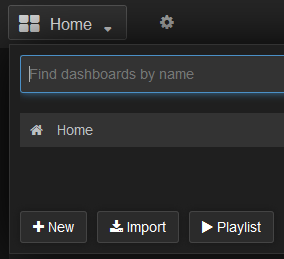
[](http://jansipke.nl/wp-content/uploads/grafana1.png)

Now click on Data Sources and then Add new. Fill in the resulting screen as follows:

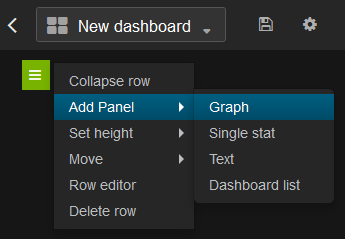
[](http://jansipke.nl/wp-content/uploads/grafana2.png)

The default user for InfluxDB is root and the password is also root. Test the connection and then save.

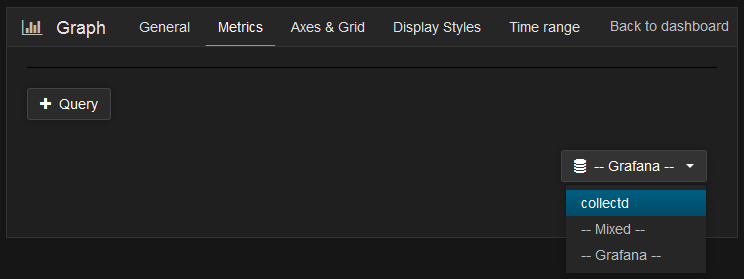
Click on Dashboards, then on the triangle pointing down next to Home, then onNew:

[](http://jansipke.nl/wp-content/uploads/grafana3.png)

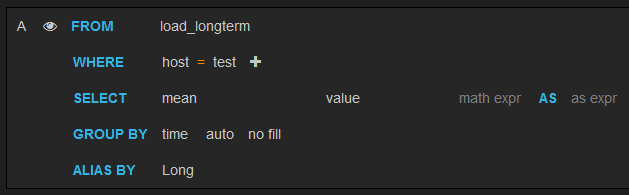
The screen now shows a small green rectangle just below New dashboard. Hover over this rectangle and select Add Panel and then Graph:

[](http://jansipke.nl/wp-content/uploads/grafana4.png)

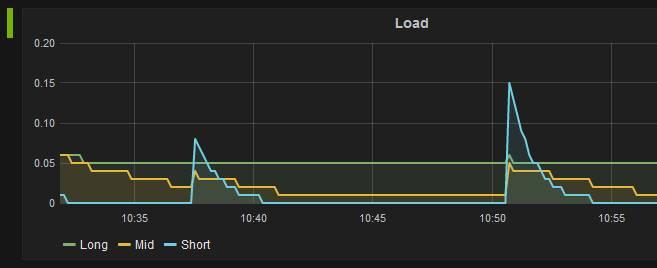
A graph is now shown with test data. Click on the title of this graph and choose Edit. Click on— Grafana — and select collectd as the datasource:

[](http://jansipke.nl/wp-content/uploads/grafana5.png)

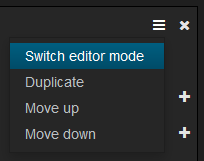
An edit window now allows you to change the appearance and data of the graph. On theGeneral tab, change the title of the graph into Load. On the Metrics tab, click on select measurement next to FROM and select load\_longterm. Click on the plus sign next toWHERE and select host. Then click on select tag value and choose the name of the host you want to display the load from. Finally, change the value of ALIAS BY to Long:

[](http://jansipke.nl/wp-content/uploads/grafana6.png)

Click on + Query to add similar queries for the mid term and short term load values. Finally, click on the save button on the top of the screen. The graph should now look like this:

[](http://jansipke.nl/wp-content/uploads/grafana7.png)

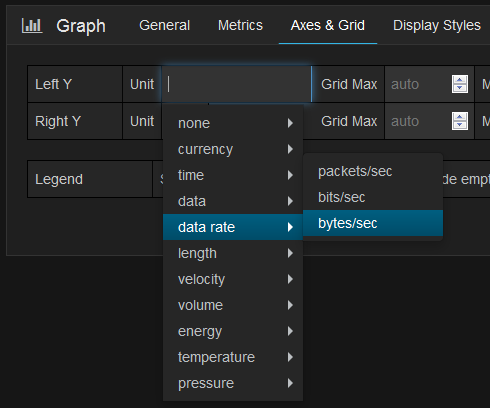
For showing the network traffic statistics for our host, we will add a new graph to the dashboard. Click on + ADD ROW, then on the green rectangle and select Add Panel andGraph again. Select the collectd datasource. This time, we will not be able to use the visual query editor. The measurements from interface\_rx and interface\_tx are numbers that show the total number of packets or bytes, instead of the numbers per time unit. Click on the three horizontal lines on the right side of the graph editor and choose Switch editor mode:

[](http://jansipke.nl/wp-content/uploads/grafana8.png)

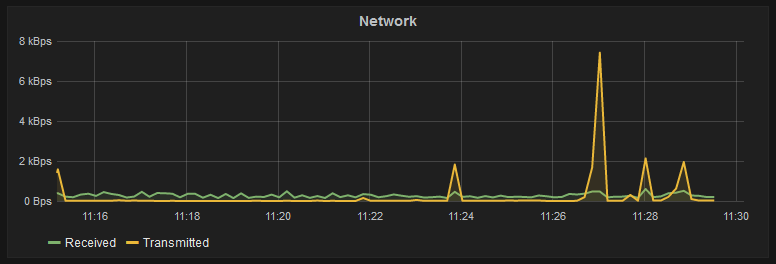
Use the following text in the input box:

*SELECT derivative("value") AS "value" FROM "interface\_rx" WHERE "host" = 'test' AND "type" = 'if\_octets' AND "instance" = 'eth0'*

Make sure to change the value of host and instance to the correct values. Also make sure to use single quotes for the values and not double quotes. On the Axes & Grid tab, change the unit of Left Y to bytes per second:

[](http://jansipke.nl/wp-content/uploads/grafana9.png)

Click on + Query to add a similar query for interface\_tx. Add aliases for both queries:Received for interface\_rx and Transmitted for interface\_tx. Now the graph looks like this:

[](http://jansipke.nl/wp-content/uploads/grafana10.png)

THIS ENTRY WAS POSTED IN [PROGRAMMING](http://jansipke.nl/category/programming/) AND TAGGED [CENTOS](http://jansipke.nl/tag/centos/), [COLLECTD](http://jansipke.nl/tag/collectd/), [GRAFANA](http://jansipke.nl/tag/grafana/), [INFLUXDB](http://jansipke.nl/tag/influxdb/). BOOKMARK THE [PERMALINK](http://jansipke.nl/monitoring-hosts-with-collectd-influxdb-and-grafana/).

# Using Grafana on Top of Elasticsearch - DZone Big Data

https://dzone.com/articles/using-grafana-on-top-of-elasticsearch

*Need to build an application around your data?*[*Learn more*](https://dzone.com/go?i=200129&u=http%3A%2F%2Fhubs.ly%2FH06Pr9h0)*about dataflow programming for rapid development and greater creativity.*

Now, why would I want to do that?

If I’m using ELK, I already have Kibana — and since [version 5.x](http://logz.io/blog/kibana-5/), Timelion is provided out-of-the-box so I can [use that for analyzing time-series data](http://logz.io/blog/kibana-timelion-time-series-analysis/), right?

Well, yes and no.

While very similar in terms of what can be done with the data itself within the two tools. The main differences between Kibana and Grafana lie in configuring how the data is displayed. Grafana has richer display features and more options for playing around with how the data is represented in the graphs.

While it takes some time getting accustomed to building graphs in Grafana — especially if you’re coming from Kibana — the data displayed in Grafana dashboards can be read and analyzed more easily.

Here are some instructions on setting up the integration with Elasticsearch and getting started with your first Grafana dashboard.

Installing Grafana

This article assumes you have an [ELK Stack](http://logz.io/learn/complete-guide-elk-stack/) up and running already, so the first step is to install Grafana.

The instructions below are for Ubuntu/Debian. If you’re using a different OS, refer to Grafana’s excellent docs [here](http://docs.grafana.org/) (if you’re using Docker, that’s probably the [easiest way](http://docs.grafana.org/installation/docker/) to get Grafana up and running).

To do this, first add the following line to your /etc/apt/sources.list file (don’t worry about the version name, keep it as jessie even if you’re using a more recent version:

deb https://packagecloud.io/grafana/stable/debian/ jessie main

Next, add the Package Cloud key so you can install a signed package:

curl https://packagecloud.io/gpg.key | sudo apt-key add -

Update your repos and install Grafana with:

sudo apt-get update && sudo apt-get install grafana

Last but not least, start Grafana:

sudo service grafana-server start

Open your browser at http://:3000 and use admin/admin as the credentials to access Grafana:



Connecting to Elasticsearch

Once installed, your next step is to set up the integration with a data source — in our case, Elasticsearch.

Click on the **Add data source** button displayed in your Grafana Home Dashboard, and configure the connection with Elasticsearch.

A few pointers.

You will be required to enter the name of the Elasticsearch index with which you want to integrate. Use this cURL on the host on which Elasticsearch is installed to get a list of all Elasticsearch indices:

curl -XGET 'localhost:9200/\_cat/indices?v&pretty'

An example output:

health status index                 uuid                   pri rep docs.count docs.deleted store.size pri.store.size

yellow open   metricbeat-2017.01.23 VPzuOuthQtSxsmo9bEscGw   5   1      12309            0      3.4mb          3.4mb

yellow open   .kibana               nIq0NUcRT1ejxw4-MwHXWg   1   1          2            0     68.2kb         68.2kb

In the HTTP settings section, you will be required to select the type of access to use. Just to clarify, in the direct access, the URL that you provide is accessed directly from the browser, whereas, in the proxy access, the Grafana backend acts as a proxy and routes requests from the browser to Elasticsearch.

Here are the settings that I used to connect with an Elasticsearch installed on an AWS EC2 instance:

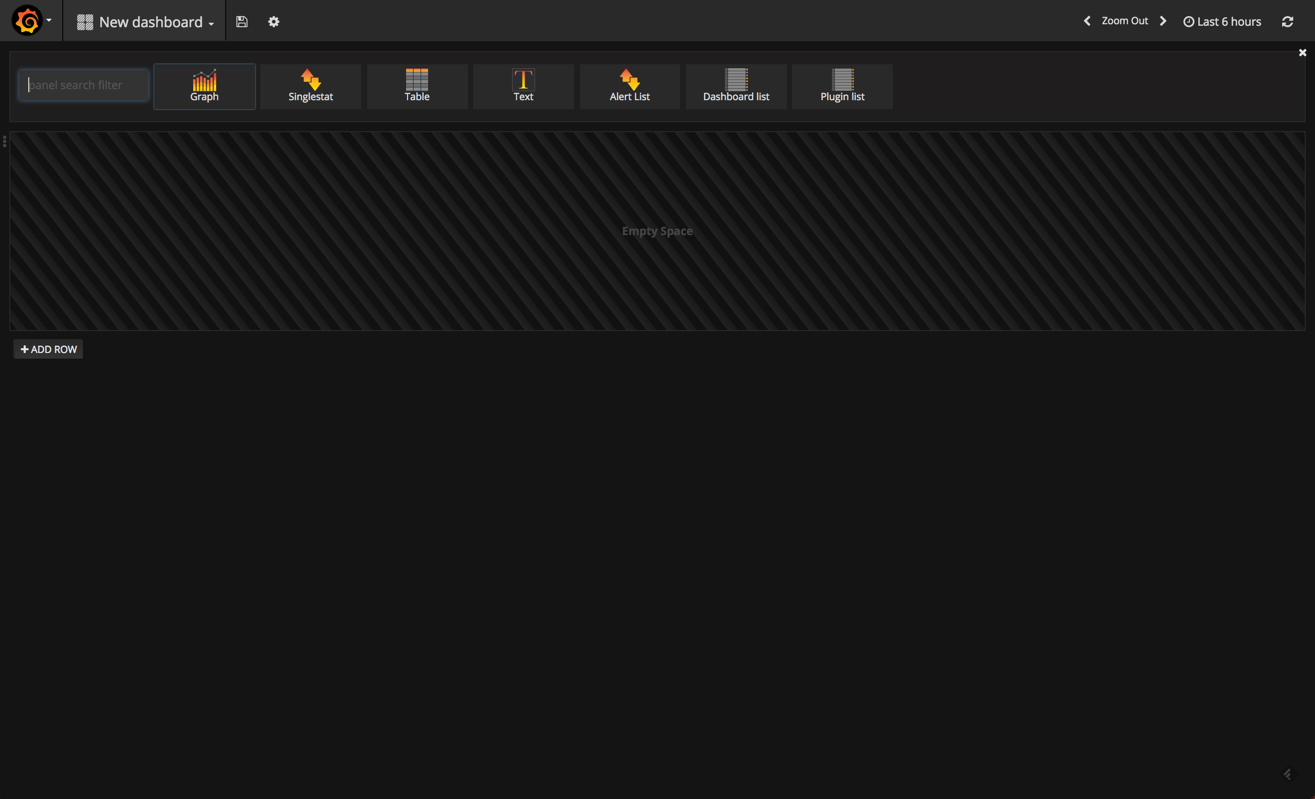


Click **Save & Test**. A green success message means that Elasticsearch was connected successfully.

Creating a Grafana Dashboard

For this tutorial, I defined two data sources for two different Elasticsearch indices — one for Apache logs shipped using [Filebeat](http://logz.io/blog/filebeat-wizard/) and the other for server performance metrics to Elasticsearch using[Metricbeat](http://logz.io/blog/metricbeat-elastic-stack-5-0/).

We’ll start by creating a new dashboard. This is done by clicking on the Grafana icon in the top-left corner and selecting **Dashboards**>**New**.



In Grafana 4.1, you have the selection of different visualizations — or “panels,” as they are called in Grafana — to choose from at the top of the dashboard.

We’re going to select the Graph panel, which is the most frequently-used panel type. By default, a nice panel is displayed showing some sort of data over time. Don’t get too excited — this is not your Elasticsearch data but some fake data source Grafana that is using to help us get started.

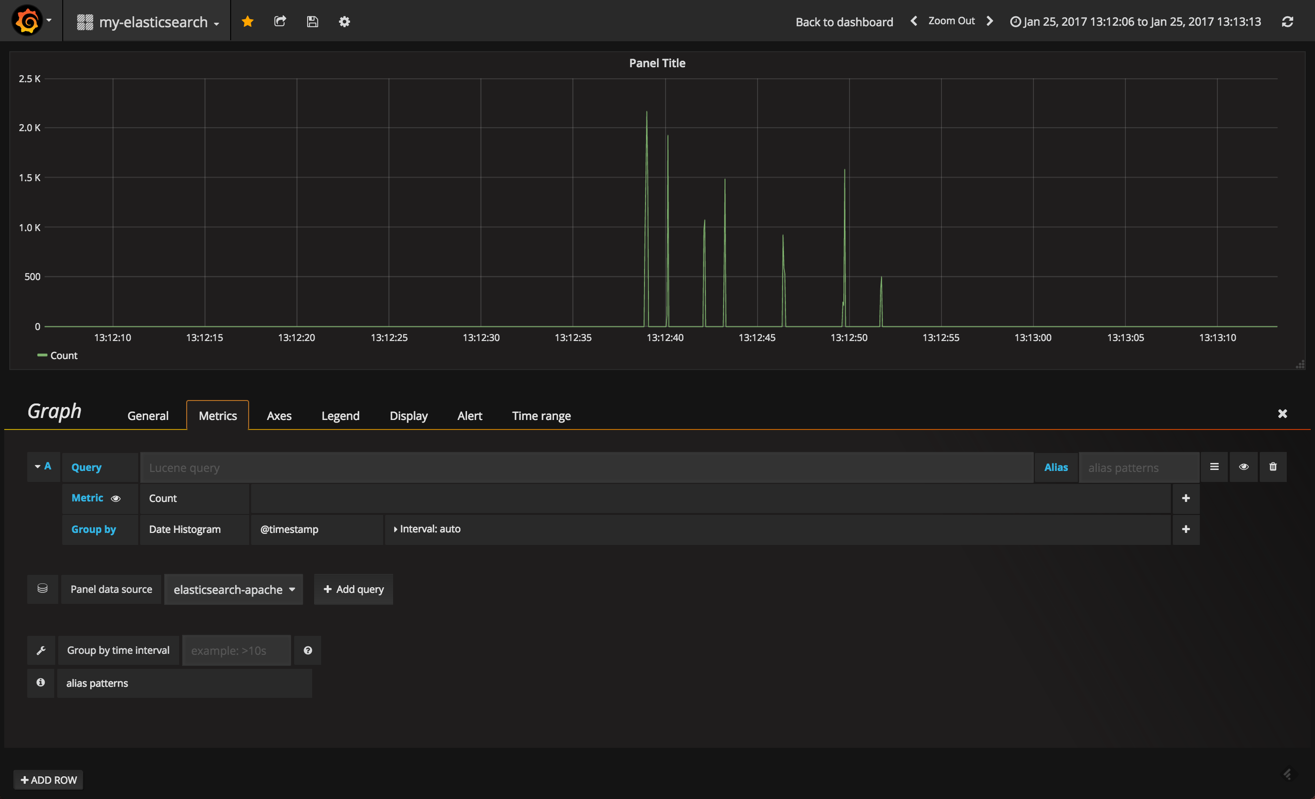
To edit the graph, you need to click the panel title and then **Edit**.



Our graph is opened in edit mode, with the Metrics tab open. This tab is the most important one because it defines what data to display. Of course, and like in Kibana, the options on display here will vary based on the data source and data type.

Start by removing the fake data source and adding your Elasticsearch data source.

Then click **+ Add query**.



The options for defining what and how the data is to be cut is similar to Kibana — in the query field, define your Lucene query and then select an aggregation type for both the Y (metric) and X (Group by) axes.

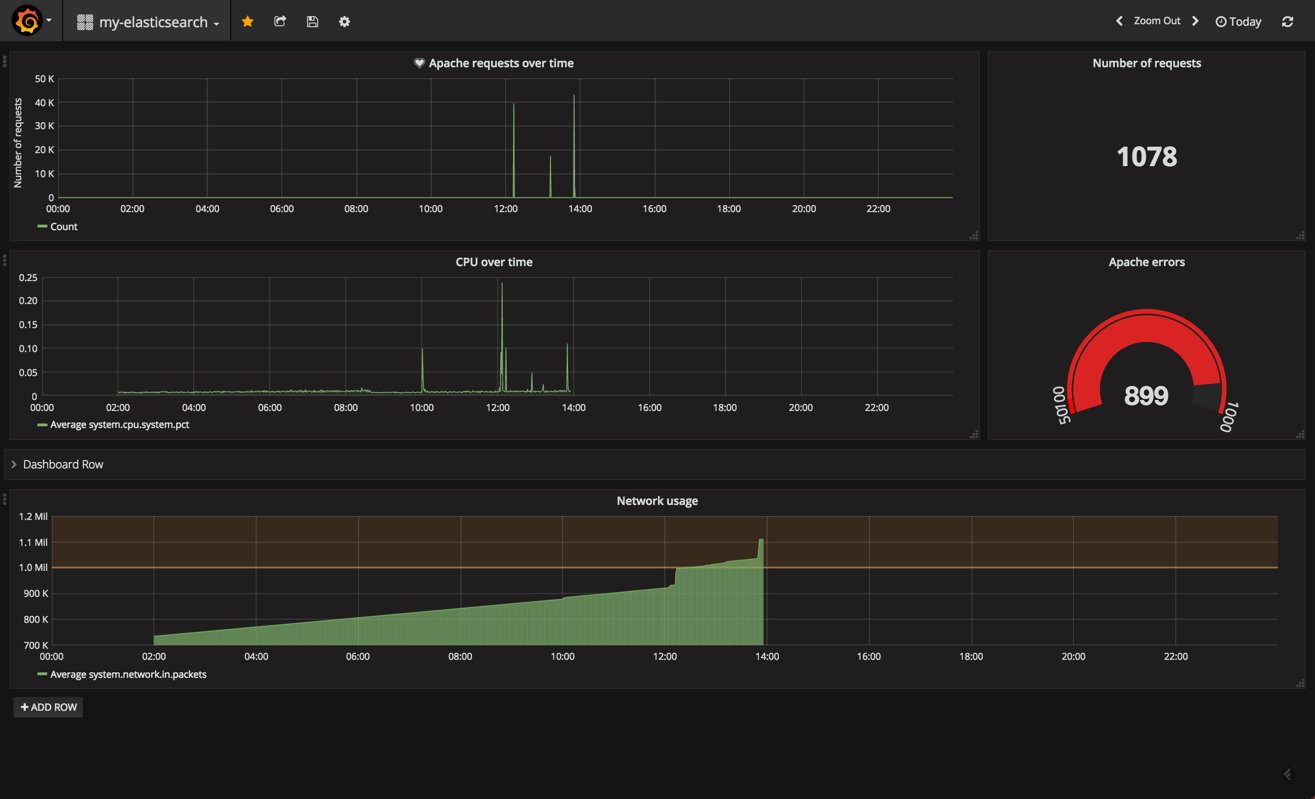
In the other tabs, the richness in display options comes to the fore.

In the**General** tab, you define the title and description for the panel. You can also add dynamic links to the panel that can link to other dashboards or URLs.

In the **Axes** tab you can play around with the units and scales for the X and Y axes and add custom labels for each axis.

We can continue to build our panels in a similar way. Grafana has three main panel types on offer — which is a bit limiting, compared to Kibana — but you will find that the three main types (graph, table, single stat) cover most of your monitoring needs.

In no time, you can have a dashboard up and running. Here is an example of an Apache and server performance monitoring dashboard using the two Elasticsearch indices as data sources. Of course, you could hook in any other data source that is supported by Grafana to create a more comprehensive dashboard:



Summary

From a functionality perspective, it’s hard to point out a critical parity between the two tools. So, if you’ve got your own ELK up and running, there may be no pressing need of abandoning Kibana.

Still, Grafana is well worth exploring for two main reasons. First, it’s extremely easy to set up. It took me just a matter of minutes to get the integration up and running. Second, from a mere usability perspective, Grafana has a much nicer UI and UX.

There are some compatibility issues with integrating Elasticsearch 5.x that you should be aware of — alerting, one of Grafana’s more recent features — does not seem to work well, for example.

If you’re interested in a more detailed comparison between these two great visualization tools, I recommend reading both [our high-level comparison](http://logz.io/blog/grafana-vs-kibana/) and [this more technical breakdown](https://www.rittmanmead.com/blog/2017/01/time-series-visualisations-kibana-or-grafana/) by the folks at Rittman Mead.

[*Check out*](https://dzone.com/go?i=200130&u=http%3A%2F%2Fhubs.ly%2FH06Pr9h0)*the Exaptive data application Studio. Technology agnostic. No glue code. Use what you know and rely on the community for what you don't.*[*Try the community version*](https://dzone.com/go?i=200130&u=https%3A%2F%2Fexaptive.city%2F%23%2Flanding%3Freferrer%3DGeneral)*.*

# How to effectively use the Elasticsearch data source in Grafana

and solutions to common pitfalls |

Grafana Labs Blog

https://grafana.com/blog/2016/03/09/how-to-effectively-use-the-elasticsearch-data-source-in-grafana-and-solutions-to-common-pitfalls/

Covered in this article:

* [Lucene Query Format](https://grafana.com/blog/2016/03/09/how-to-effectively-use-the-elasticsearch-data-source-in-grafana-and-solutions-to-common-pitfalls/#lucene)
* [Templated Queries](https://grafana.com/blog/2016/03/09/how-to-effectively-use-the-elasticsearch-data-source-in-grafana-and-solutions-to-common-pitfalls/#templated)
* [Sawtooth-Like Graphs](https://grafana.com/blog/2016/03/09/how-to-effectively-use-the-elasticsearch-data-source-in-grafana-and-solutions-to-common-pitfalls/#sawtooth)
* [Incomplete data at the beginning and the end of a graph](https://grafana.com/blog/2016/03/09/how-to-effectively-use-the-elasticsearch-data-source-in-grafana-and-solutions-to-common-pitfalls/#incomplete)
* [Sum function broken](https://grafana.com/blog/2016/03/09/how-to-effectively-use-the-elasticsearch-data-source-in-grafana-and-solutions-to-common-pitfalls/#sum)

About one year after I created an [issue](https://github.com/grafana/grafana/issues/1034) at Grafana’s GitHub page, we finally have support for using Elasticsearch as a time series database! At that time, I was trying to lower the burden of adopting the open source Java performance monitoring tool [stagemonitor](http://www.stagemonitor.org/)which I’m the project lead of. As stagemonitor already requires Elasticsearch for its request analysis feature, someone suggested that I should try to also store the metrics data into Elasticsearch so that stagemonitor users only have to install one database. At first, I was like ‘are you nuts? Elasticsearch is no TSDB, it’s used for logs and stuff but not for metrics!’. But the idea was interesting and I could not stop thinking about it. Also, around that time I first read about the [metrics 2.0](http://metrics20.org/) approach which basically means that you don’t have ‘classic’ metric names that are dot-separated but rather a document of key-value pairs. This has several advantages e.g. it makes the metrics more self-descriptive and lets you add tags (aka dimensions like datacenter) afterwards without changing the metric identity (which would otherwise break your existing dashboards). To me it seemed like Elasticsearch actually would be a great fit for this data model and its [aggregation framework](https://www.elastic.co/guide/en/elasticsearch/reference/master/search-aggregations.html) could replace the most important functions of graphite.

I have already [blogged](https://www.elastic.co/blog/elasticsearch-as-a-time-series-data-store) about how to set up the data model so that it fits most naturally into Elasticsearch and how to tune the mapping to minimize disk overhead. In this post, I want to focus on how to effectively use the Grafana query editor and how to avoid common pitfalls.

When using the Elasticsearch data source, you have to keep in mind that this feature is relatively new. So there are some rough edges and not all functions (aggregations) Elasticsearch provides are currently available in the editor. When coming from Graphite, you first have to get familiar with the way aggregations work. But still, you might miss some functions especially if it comes to interacting with multiple series like dividing or subtracting two series. So before migrating all your metrics to Elasticsearch, you should do a proof of concept whether you will still be able to construct your most important graphs. In general, the Elasticsearch data source is more similar to InfluxDB or OpenTSDB which are based on tags and values than to Graphite which is based on dot-separated metric names.

One big problem of the TSDB market is that there are no standards (like SQL) and that there is no clear leader. There are so many TSDBs out there that all have some pros and cons. That’s why Grafana supports a lot of [data sources](http://docs.grafana.org/datasources/overview/) which is great, but switching from one to another TSDB requires a lot of effort because the fundamental concepts are different and the graphs and dashboards have to be rebuilt without an easy migration path. However, there are [plans](https://github.com/grafana/grafana/issues/3677) to support functions within Grafana that can be applied to all data sources. That means that only a basic set of functions have to be provided by the data sources. In particular, those which require access to all data points like the percentiles function. Other functions like the difference between two series can just as well be performed within Grafana.

**Lucene query format**

The biggest difference to the query editor for InfluxDB is that there is no input for FROM and WHERE but only a Lucene query input field. The [Lucene query syntax](https://lucene.apache.org/core/2_9_4/queryparsersyntax.html) is a quite powerful language for constructing a search query. But don’t despair, you don’t have to know all the nuances of that syntax. Most use cases can be handled with a basic set of instructions. Also, in the future, there might be a [more convenient way to enter Elasticsearch queries](https://github.com/grafana/grafana/issues/3806) which will probably work more like the InfluxDB WHERE input so you don’t have to mess with the actual Lucene query for simple cases.

https://grafana.com/blog/assets/img/blog/migrated/influx_query.png Grafana Query editor for InfluxDB

https://grafana.com/blog/assets/img/blog/migrated/es_query.png Grafana Query editor for Elasticsearch

The fundamental difference of these databases is that in InfluxDB you have a dedicated name for the series that is like a table name in relational databases. In Elasticsearch there is no dedicated series name but you have tags and values you can use to filter, for example, to show only metrics of a certain host. So when you are using Elasticsearch, it is advisable that you always have a tag called name, that acts like the series name in InfluxDB. Of course you can also use a different tag like series, @metric or something else. Just make sure that this name is consistent throughout all your metrics.

Let’s say we want to migrate the InfluxDB query FROM logins.cout WHERE hostname= $Hostname to Elasticsearch. At first we have to translate the FROM clause into a Lucene query like this:

name:logins.count

The second step is to apply the templated hostname filter

name:logins.count AND hostname:$Hostname

The important thing is that you explicitly have to combine the two filters with the boolean operator AND. If you forget the AND, the default operator OR will be applied which would result in a mess because it would mean: either collect the logins.count metric or return any metric where hostname is $Hostname. Remember: the series name is just another tag. There is another important point: if a value you want to filter by contains a whitespace you have to quote it.

name:logins.count AND hostname:“my awesome host”

Best practice is that you don’t have white spaces in tag values. I’ll tell you why in a moment.

###Templated queries [Templated queries](http://docs.grafana.org/reference/templating/) are a great way to make dashboards more dynamic by allowing users to slice and dice the data. For example, you could make a dashboard where you get an overview of all data across all hosts and use a templated query to be able to see the data of one or more individual hosts. To do this with the Elasticsearch data source, we currently have to enter raw JSON data when creating a template. The following collects all hosts that are in the eastern datacenter:

{“find”: “terms”, “field”: “host”, “query”: “datacenter: east”}

When a certain host in the dropdown gets selected, the value of the template variable $Host changes. But that does not mean that all graphs are automatically filtered by the currently selected host like you may be used to if you have used Kibana before. To apply the host filter to a graph you have to explicitly add it to the Lucene query like this: https://grafana.com/blog/assets/img/blog/migrated/templated_query.png

But beware of white spaces in template values. Because the templating system works with simple string replacement, the values get inserted into the raw Lucene query and the query fails. Looking closer at the [Lucene query reference](https://lucene.apache.org/core/2_9_4/queryparsersyntax.html#Fields) you could put the template variable in quotes like host:“$Host” but that in turn fails if you select multiple hosts. In that case $Host gets expanded to (“host1” OR “host2”). So quoting $Host results in “(“host1” OR “host”)” which leads to an invalid query again. For the same reason inserting a template variable into a template query fails:

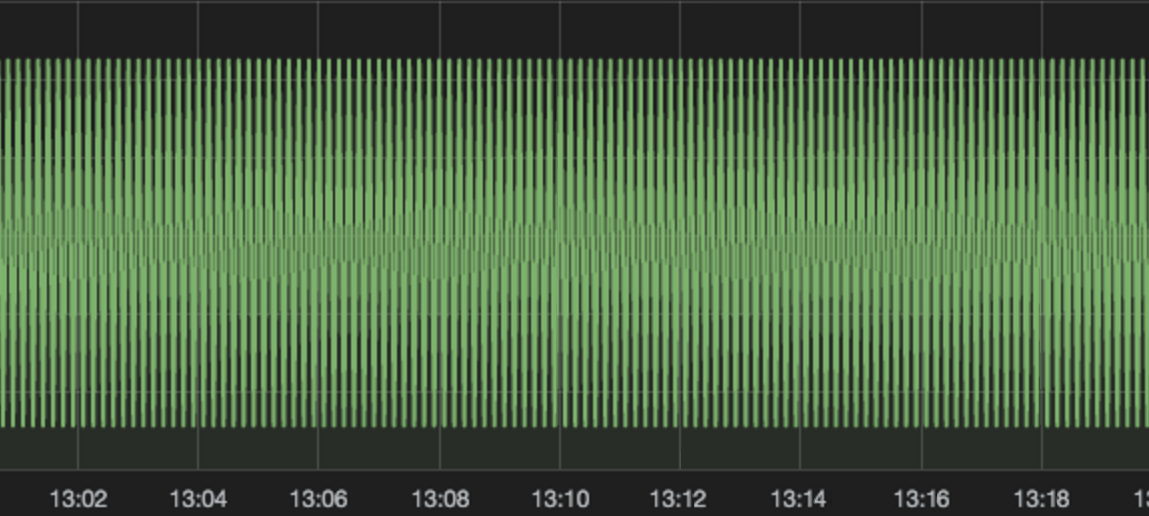
{“find”: “terms”, “field”: “host”, “query”: “datacenter:$Datacenter”}

If only one data center is selectable everything is fine, but as soon as you select more than one, the query becomes invalid.

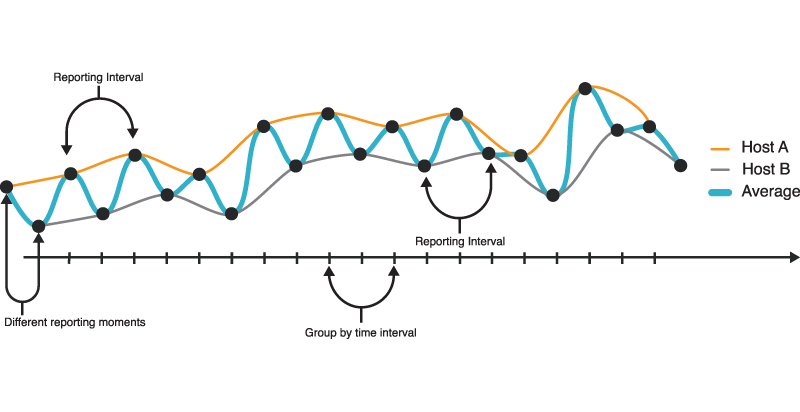
Long story short: avoid white spaces in tag values.

**Sawtooth-like graphs**

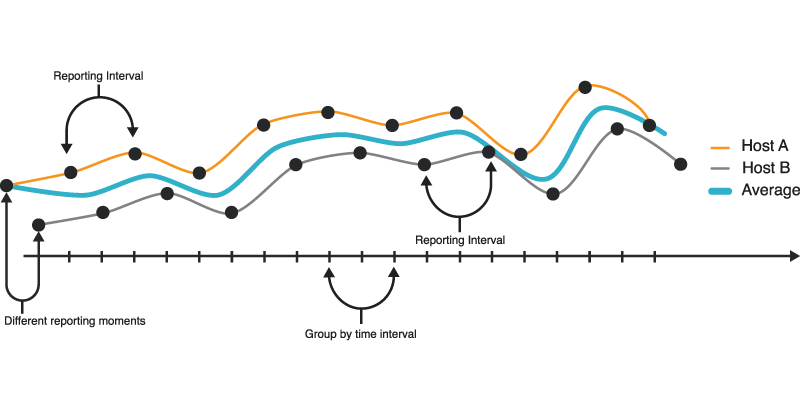
This problem can appear when you have multiple reporters that are sending metrics to Elasticsearch, for example, if you have two instances of a web application that are monitored by [stagemonitor](http://www.stagemonitor.org/). Typically you configure them so that they report in a uniform interval (like each 60 seconds). But the moment they are reporting depends on when the servers are started.

If you want to graph the average response time across your web servers, that difference in the reporting moment can cause these ugly sawtooth-like graphs (don’t look too long at it - it gets kinda hypnotic ;) 

The problem is more than one data point (the group by time interval) per 60 seconds (the reporting interval) gets requested. So the average function does not really take the average value of the values of ‘Host A’ and ‘Host B’ but rather hops between their values, because they don’t fall into the same bucket.



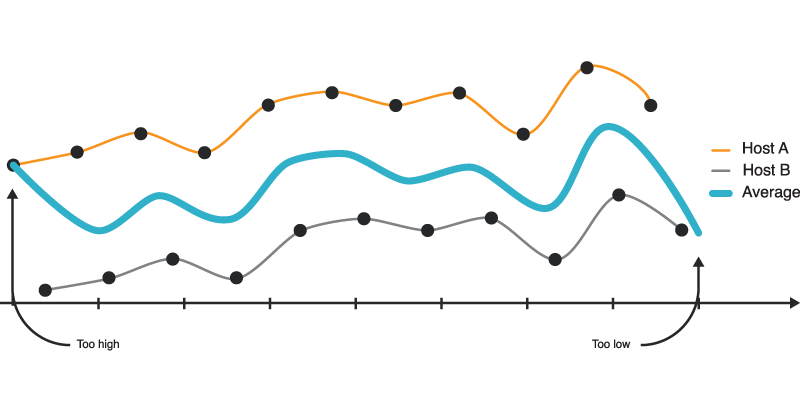
The solution to this is to make sure that the values always fall into the same bucket regardless of the exact moment the different hosts reported their values.



In Grafana this is quite easy as you can set a lower bound for the group by time interval in the Metrics tab of the panel editor like this: https://grafana.com/blog/assets/img/blog/migrated/groupby3.png

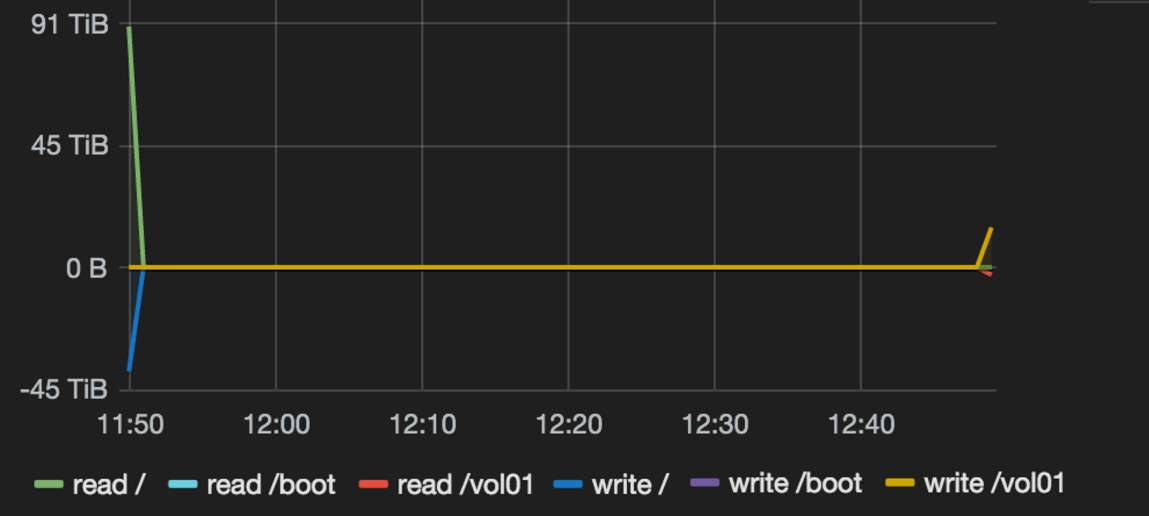
If you are using [stagemonitor](http://www.stagemonitor.org/) you don’t have to do this manually as it already knows the reporting interval and sets the group by time interval accordingly on all graphs when importing the dashboards.

**Incomplete data at the beginning and the end of a graph**

This problem also results from the different reporting moments that occur when multiple agents push metrics into Elasticsearch. The group by time ‘trick’ works for most cases but at the very beginning of a graph where there might be one data point of ‘Host A’ that falls into a bucket where there is none for ‘Host B’ because B’s data point (that should be in the same bucket) is not within the requested timeframe. The same problem can also occur at the very end of a graph. 

Most of the times this might not be a real issue, but in some cases it totally screws up your graph. A good example of that is when you want to monitor Disk I/O. From /proc/diskstats you get the total bytes written since the last reboot. That number can diverge quite massively between different servers. To get a meaningful graph, you probably want to apply a derivative function so that you get the bytes written per second. That together makes a missing data point at the end/beginning fatal as it causes a massive spike so that the data in the middle is not visible. This is why:

First, the average of the values of all hosts are calculated. If there is a massive difference in the values of the hosts, the average is a lot different when there is a missing datapoint. Things get worse if you also apply a derivative function. For example let’s assume the value of Host 1 is 1 and the value of Host 2 is 1,000,000 and their values increase by 1 each time they report. In a happy world, the fist average is 500,000.5 the second one 500,001.5 and so on and the derivative is always 1. But let’s say the first value of Host 1 is missing. The first average now is 1,000,000 and the second one 500,001,5. That makes the graph look a bit quirky but that’s something we can live with. But when applying the derivative function everything starts to fall apart. The fist derivative is 499,998.5 (1,000,000-500,001,5) and the consequent ones are always 1 again. That leads to a massive spike at the beginning of the graph so that the real information (that the derivative is always 1) is not visible. This is what it looks like:

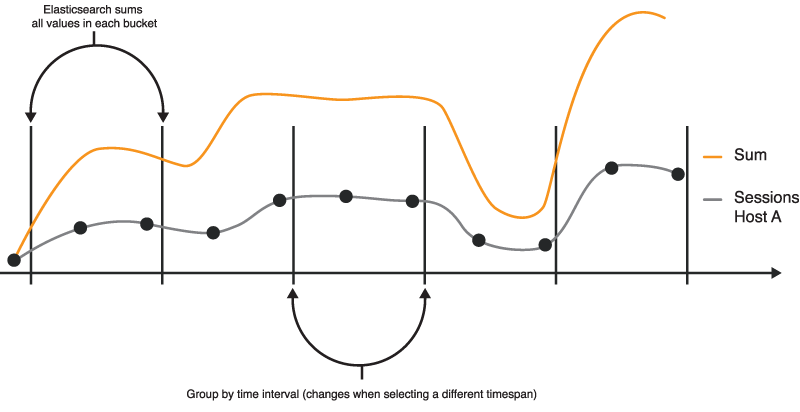


One thing you can do to avoid this issue is to perform a group by on the tag that identifies a reporter. In this case I would do a group by on the host tag because there is one reporter on each host that sends metrics. So each host has an individual graph, thus I avoid the problem.

One thing that Grafana could do to avoid this issue is to ask for a larger timeframe than requested and then truncate the graph client side. But this only fixes the start of the graph, as you obviously can’t look into the future to include the not-yet-reported value of ‘Host A’.

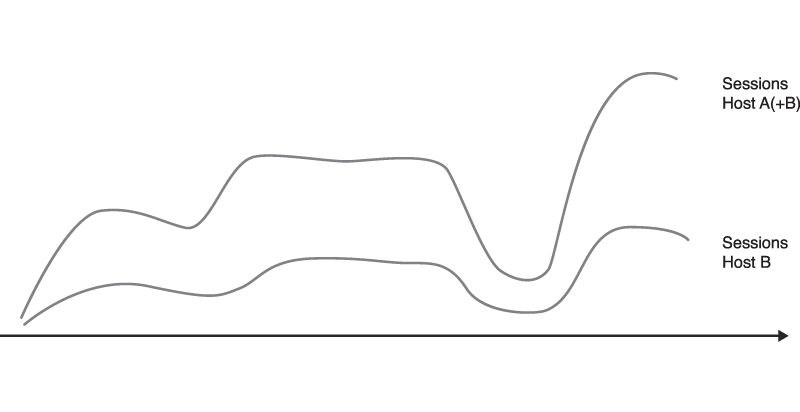
**Sum function broken**

A common question when monitoring a clustered web application is to find out how many users are currently online. One simple way would be to just sum all sessions across all hosts. Sounds easy. Sounds like I just have to apply the sum function of Elasticsearch and I’m good. Well, unfortunately, that’s wrong. 

This time, the problem doesn’t have to do with multiple reporters or different reporting moments. It even occurs when you want to sum the sessions of only one application. To understand the issue we first have to understand how the sum function in Elasticsearch works. To display a graph, we always have to apply the date histogram aggregation, which creates buckets for a certain timeframe (the group by time interval). Grafana auto-selects the bucket size that is appropriate for the current timeframe. The larger the time span of the graph gets (for example last 24h) the bigger the buckets get and the more individual data points fall within a single bucket. For example, when we report each 30 seconds and the bucket size is 60 seconds there are two data points per bucket. The sum function works by summing all values that are in a single bucket. That means that the calculated sum is higher than we expect. 

That renders the sum and also the count function quite useless for most cases and you should rather work with the average andmax functions. There, unfortunately, is not an easy fix for that. I came up with two attempts to solve that problem.

The first attempt is based on the fact that the sum is accurate when the reporting interval is equal to the size of the buckets. So if we divide the sum by bucket size/reporting interval in a script, it would show the correct sum regardless of the bucket size. But that attempt comes with big flaws which doesn’t make that a viable solution. We would somehow have to know what the current group by time interval is. Grafana could, for example, expose a template variable for that. But this seems like an ugly hack to me. The other reason is that when you want to change the reporting interval you would also have to change every single graph that is built that way - a maintenance nightmare.

A more pragmatic solution is to group the sessions by Hosts and then stack the values. 

The problem with that approach is that it can quickly get quite messy. Especially if you have a lot of hosts and you are also grouping by other dimensions, the number of individual graphs quickly add up. For example, let’s say you want to stack the read and write I/O of all your hosts for each mount point. Depending on how many hosts you are monitoring and how many different mount points they have, you might end up with thousands of graphs.

**Conclusion**

I hope you found this Elasticsearch article useful and will give the data source a shot. If you come up with a better approach on how to solve a particular problem, or you came across another challenge, please let me know in the comments. I should also mention that most of the problems are not specific to Elasticsearch and will also occur if you are using InfluxDB or other data sources that work in a similar way.

**Recent Posts**

[**Grafana 4.2 Release**](https://grafana.com/blog/2017/03/22/grafana-4.2-release/)

[**We are now Grafana Labs**](https://grafana.com/blog/2017/03/13/we-are-now-grafana-labs/)

[**Grafana 4.2 Beta Release**](https://grafana.com/blog/2017/02/27/grafana-4.2-beta-release/)

[**GrafanaCon 2016 Videos Available**](https://grafana.com/blog/2017/01/13/grafanacon-2016-videos-available/)

[**Grafana 4.1 Release**](https://grafana.com/blog/2017/01/11/grafana-4.1-release/)

**Categories**

[**Company**](https://grafana.com/blog/categories/company/)

[**Data Visualization**](https://grafana.com/blog/categories/data-visualization/)

[**Design**](https://grafana.com/blog/categories/design/)

[**DevOps**](https://grafana.com/blog/categories/devops/)

[**Elasticsearch**](https://grafana.com/blog/categories/elasticsearch/)

[**Engineering**](https://grafana.com/blog/categories/engineering/)

[**Grafana**](https://grafana.com/blog/categories/grafana/)

[**GrafanaCon**](https://grafana.com/blog/categories/grafanacon/)

[**Graphite**](https://grafana.com/blog/categories/graphite/)

[**Monitoring**](https://grafana.com/blog/categories/monitoring/)

[**Open Source**](https://grafana.com/blog/categories/open-source/)

[**Prometheus**](https://grafana.com/blog/categories/prometheus/)

[**Release**](https://grafana.com/blog/categories/release/)

[**Snap Telemetry**](https://grafana.com/blog/categories/snap-telemetry/)

[**Time Series Databases**](https://grafana.com/blog/categories/time-series-databases/)

[**Tutorials**](https://grafana.com/blog/categories/tutorials/)

[**UX**](https://grafana.com/blog/categories/ux/)

[**WorldPing**](https://grafana.com/blog/categories/worldping/)

[comments powered by Disqus](http://disqus.com/)

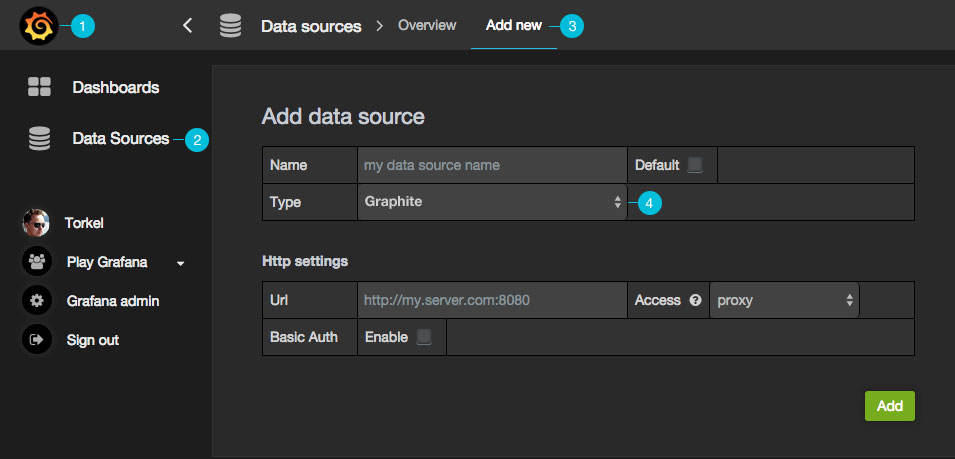
# Using Elasticsearch in Grafana | Grafana Documentation

http://docs.grafana.org/features/datasources/elasticsearch/

**Using Elasticsearch in Grafana**

Grafana ships with advanced support for Elasticsearch. You can do many types of simple or complex elasticsearch queries to visualize logs or metrics stored in elasticsearch. You can also annotate your graphs with log events stored in elasticsearch.

**Adding the data source**



1. Open the side menu by clicking the the Grafana icon in the top header.
2. In the side menu under the Dashboards link you should find a link named Data Sources.

NOTE: If this link is missing in the side menu it means that your current user does not have theAdmin role for the current organization.

1. Click the Add new link in the top header.
2. Select Elasticsearch from the dropdown.

| **Name** | **Description** |
| --- | --- |
| Name | The data source name, important that this is the same as in Grafana v1.x if you plan to import old dashboards. |
| Default | Default data source means that it will be pre-selected for new panels. |
| Url | The http protocol, ip and port of you elasticsearch server. |
| Access | Proxy = access via Grafana backend, Direct = access directly from browser. |

Proxy access means that the Grafana backend will proxy all requests from the browser, and send them on to the Data Source. This is useful because it can eliminate CORS (Cross Origin Site Resource) issues, as well as eliminate the need to disseminate authentication details to the Data Source to the browser.

Direct access is still supported because in some cases it may be useful to access a Data Source directly depending on the use case and topology of Grafana, the user, and the Data Source.

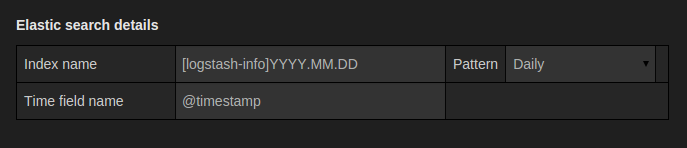
**Direct access**

If you select direct access you must update your Elasticsearch configuration to allow other domains to access Elasticsearch from the browser. You do this by specifying these to options in your **elasticsearch.yml** config file.

http.cors.enabled: true

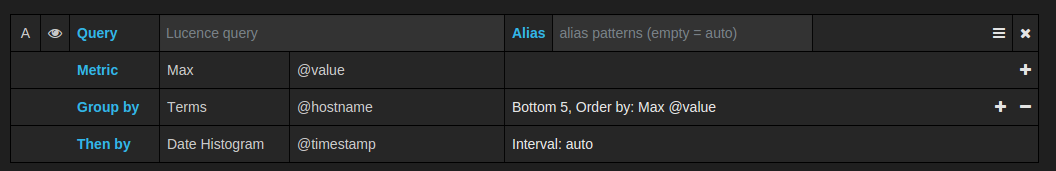
http.cors.allow-origin: "\*"

**Index settings**



Here you can specify a default for the time field and specify the name of your elasticsearch index. You can use a time pattern for the index name or a wildcard.

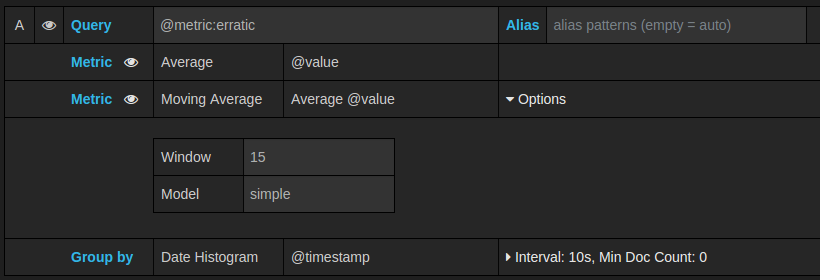
**Metric Query editor**



The Elasticsearch query editor allows you to select multiple metrics and group by multiple terms or filters. Use the plus and minus icons to the right to add / remove metrics or group bys. Some metrics and group by have options, click the option text to expand the the row to view and edit metric or group by options.

**Pipeline metrics**

If you have Elasticsearch 2.x and Grafana 2.6 or above then you can use pipeline metric aggregations like **Moving Average** and **Derivative**. Elasticsearch pipeline metrics require another metric to be based on. Use the eye icon next to the metric to hide metrics from appearing in the graph. This is useful for metrics you only have in the query to be used in a pipeline metric.



**Templating**

The Elasticsearch datasource supports two types of queries you can use to fill template variables with values.

**Possible values for a field**

{"find": "terms", "field": "@hostname"}

**Fields filtered by type**

{"find": "fields", "type": "string"}

**Fields filtered by type, with filter**

{"find": "fields", "type": "string", "query": <lucene query>}

**Multi format / All format**

Use lucene format.

# Grafana 和 Elasticsearch - 推酷

http://www.tuicool.com/articles/VfqiIjz

时间 2016-12-28 11:56:08  [伪架构师](http://www.tuicool.com/sites/EjEFJfN)

原文  [http://blog.fleeto.us/content/grafana-he-elasticsearch](http://blog.fleeto.us/content/grafana-he-elasticsearch?utm_source=tuicool&utm_medium=referral)

主题 [Grafana](http://www.tuicool.com/topics/11040110)[ElasticSearch](http://www.tuicool.com/topics/11020003)

容器化和微服务，让世界花枝招展，又支离破碎。一个典型的运行在容器云之上的微服务架构的应用，通常是由多种服务和基础设施的支撑而来的。这对运维工作提出一个很大的挑战 —— 一个应用后端的众多系统，究竟是怎样的工作状况？

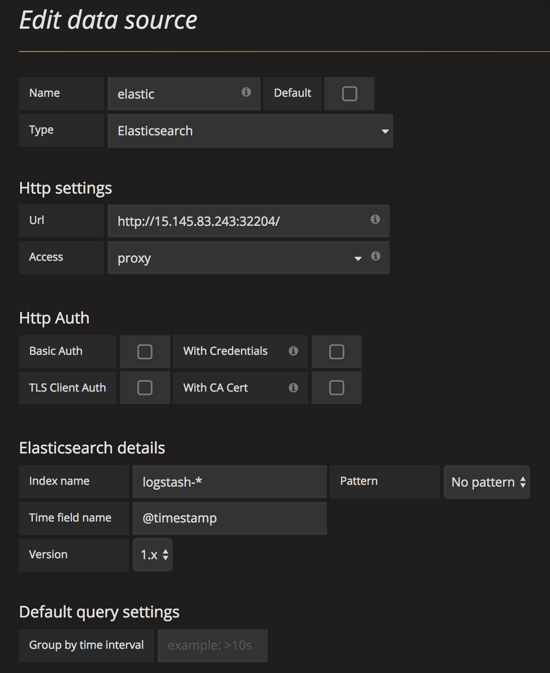
事实上，所有构成这一应用的微服务以及支撑这一应用的所有基础设施，都会有各自的日志、指标数据，以及构建在上游的监控、日志系统。各处分散的数据和系统，会给支持团队造成极大的负担，也最终成为开发运维工作的拦路虎。

之前的经验中，可以把自家应用的各种业务、技术指标通过 Zabbix 或者 influxDB 进行存储，经由 Grafana 的插件系统进行整合展示，目前流行的容器云支撑系统 Kubernetes，也能够通过 influxDB 在 Grafana 上展示 Heapster 搜集到的数据。指标的事情解决了，下面自然就是日志了。

Grafana 也提供了针对 Elasticsearch 的数据源插件。下面用 Kubernetes 中正在运行的日志收集实例来展示 Grafana 对 ES 的支持。

建立数据源

首先是建立一个 Elasticsearch 类型的数据源。这里使用了一个 Kubernetes 集群的 ES 日志。Type 部分选择 Elasticsearch，然后填写 URL 地址、认证方式。Index name 填写 **logstash-\*** 。大致如图所示。



点击下面的 **Test & Save** ，测试成功后完成数据源设置。

Access 一般来说都是选择 Proxy，也就是服务器间通信。

建立指标

接下来进入展示环节。利用新建的 ES 数据源，来建立展示单元。

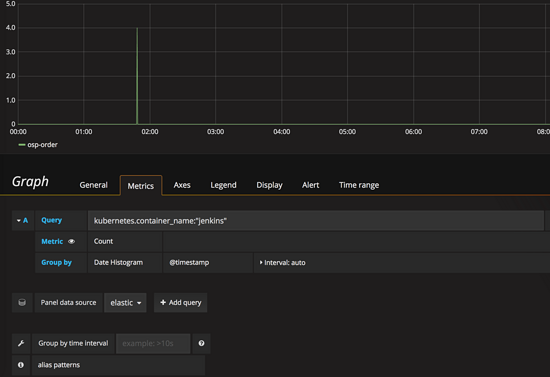
曲线图

首先建立一个 Graph Panel，数据源选择上面新建的 ES。

Query 一栏需要按照 Lucene 语法进行查询。这里我们选择 **kubernetes.container\_name: "jenkins"** ，也就是容器名称为 "jenkins" 的日志项。

其他可以保持缺省即可。

配置结束之后，稍等几秒钟，就会出现数据点。



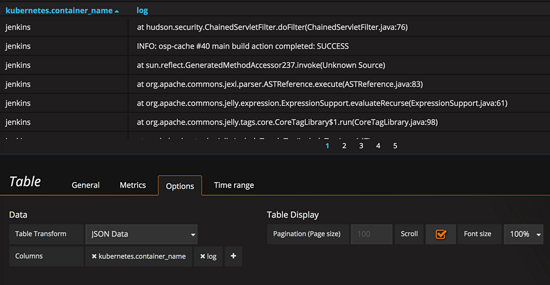
日志表格

个人感觉上面的的数字在日志来说用处并不大，我们的目的还是在同一界面下查看日志。

建立一个 Table Panel。配置数据源为 ES。 **Metric 选择 Raw Document**。

经过短暂等候，上面表格会出现一堆的 JSON 文本，我们要通过 Options 来进行展现配置。

在 Columns 中我们简单的新加两列："kubernetes.container\_name" 和 "log"，就会以表格的形式把这两列展示出来。



告一段落

至此，Grafana 就成为一个集成了众多来源的运维入口。经过进一步的加工和配置（ **其实非常大量非常琐碎** ），仅从这一个入口就能够完成很多的日常巡检任务；更重要的是，因为数据的统一展示，在业务、服务和基础设施之间建立了直观的联系，为事故的处理甚至预测，提供了更多的便利条件。

# Grafana + Zabbix --- 部署分布式监控系统 - ﹏猴子请来的救兵 - 博客园

httpwww.cnblogs.comyyhharchive201509084792830.html#l10

阅读目录：

[1. 关闭防火墙](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html" \l "l01)

[2. 安装Zabbix下载源](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l02)

[3. ZabbixClient  --- 安装zabbix-agent代理](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l03)

[4. ZabbixServer --- 安装zabbix-server服务](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l04)

[5. ZabbixServer --- 配置zabbix-server的数据库](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l05)

[6. ZabbixServer --- 配置zabbix-server的服务参数](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l06)

[7. ZabbixServer --- 使用zabbix-server](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l07)

[8. 登录Zabbix的Web端](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l08)

[9. 在Web配置zabbix监控agent](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l09)

[10. 安装Grafana以及Zabbix插件](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l10)

[11. 登录Grafana并且配置](http://www.cnblogs.com/yyhh/archive/2015/09/08/4792830.html#l11)

序章：

        Zabbix的一个很优秀的分布式监控服务器， 它有两部分组成：

1. “zabbix-server”用来收集并且在web端展示数据

2. “zabbix-agent”用来采集数据，发送给server

        在安装Zabbix时，用了3台虚拟机来测试监控的数据是否正常采集、展示：

**Server：**

ZabbixServer --- IP : 192.168.246.133

**Client：**

ZabbixClient\_1 --- IP : 192.168.246.134

ZabbixClient\_2 --- IP : 192.168.246.135

1. 关闭防火墙

        Linux的防火墙是咱们新手的噩梦，很多情况会出现能ping通，但是访问不了Web页面。所以开始就干掉它！

// 关闭防火墙

[root@localhost ~]# /etc/init.d/iptables stop

iptables: Setting chains to policy ACCEPT: filter [ OK ]

iptables: Flushing firewall rules: [ OK ]

iptables: Unloading modules: [ OK ]

// 开机关闭防火墙

[root@localhost ~]# chkconfig iptables off

2. 安装Zabbix下载源

        zabbix官方提供了下载源，直接执行下面命令，下载安装就可以了

# rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/6/x86\_64/zabbix-release-2.4-1.el6.noarch.rpm

        提示安装成功后，查看Linux下载源，是否正确安装，如果安装成功，Linux下会有zabbix.repo源文件

[root@localhost Desktop]# ll /etc/yum.repos.d/

total 28

-rw-r--r--. 1 root root 1991 Aug 4 00:13 CentOS-Base.repo

-rw-r--r--. 1 root root 647 Aug 4 00:13 CentOS-Debuginfo.repo

-rw-r--r--. 1 root root 289 Aug 4 00:13 CentOS-fasttrack.repo

-rw-r--r--. 1 root root 630 Aug 4 00:13 CentOS-Media.repo

-rw-r--r--. 1 root root 6259 Aug 4 00:13 CentOS-Vault.repo

-rw-r--r--. 1 root root 401 Sep 11 2014 zabbix.repo

3. ZabbixClient --- 安装zabbix-agent代理

       分别在ZabbixClient\_1与ZabbixClient\_2安装并配置如下设置：

       关闭防火墙

[root@localhost Desktop]# /etc/init.d/iptables stop

[root@localhost Desktop]# chkconfig iptables off

        安装Zabbix官方下载源

[root@localhost Desktop]# rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/6/x86\_64/zabbix-release-2.4-1.el6.noarch.rpm

        安装zabbix-agent，在下载的时候，会自动安装zabbix-agent的基础依赖包

[root@localhost Desktop]# yum install zabbix-agent

        进入zabbix-agent的Config文件

[root@localhost Desktop]# vim /etc/zabbix/zabbix\_agentd.conf

        配置zabbix-agent的参数，主要修改Server的IP和Hostname，

Server=192.168.246.133

Hostname=localhost

      启动zabbix-agent

[root@localhost Desktop]# /etc/init.d/zabbix-agent start

4. ZabbixServer --- 安装zabbix-server服务

       关闭防火墙

[root@localhost Desktop]# /etc/init.d/iptables stop

[root@localhost Desktop]# chkconfig iptables off

        安装Zabbix官方下载源

[root@localhost Desktop]# rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/6/x86\_64/zabbix-release-2.4-1.el6.noarch.rpm

        通过zabbix提供的下载源，下载并安装zabbix-server-mysql和zabbix-web-mysql，这个两个mysql数据库是用来存放采集到的数据，Zabbix的web端会通过这些数据，来展示绘图。而zabbix-get则是用来收集数据的指令。

# yum install zabbix-server-mysql zabbix-web-mysql zabbix-get

        在下载这两个包的时候，会提示你安装基础依赖包，其中这些基础包中，还有zabbix-server的服务端的包，选择“y”继续即可。

// 以下是需要安装的基础依赖包

================================================================================

Package Arch Version Repository Size

================================================================================

Installing:

zabbix-server-mysql x86\_64 2.4.6-1.el6 zabbix 1.5 M

zabbix-web-mysql noarch 2.4.6-1.el6 zabbix 14 k

Installing for dependencies:

OpenIPMI-libs x86\_64 2.0.16-14.el6 base 473 k

fping x86\_64 2.4b2-16.el6 zabbix-non-supported 31 k

iksemel x86\_64 1.4-2.el6 zabbix-non-supported 47 k

net-snmp x86\_64 1:5.5-54.el6\_7.1 updates 308 k

php-bcmath x86\_64 5.3.3-46.el6\_6 updates 39 k

php-mbstring x86\_64 5.3.3-46.el6\_6 updates 459 k

php-mysql x86\_64 5.3.3-46.el6\_6 updates 86 k

zabbix x86\_64 2.4.6-1.el6 zabbix 161 k

zabbix-server x86\_64 2.4.6-1.el6 zabbix 22 k

zabbix-web noarch 2.4.6-1.el6 zabbix 4.5 M

Updating for dependencies:

net-snmp-libs x86\_64 1:5.5-54.el6\_7.1 updates 1.5 M

php x86\_64 5.3.3-46.el6\_6 updates 1.1 M

php-cli x86\_64 5.3.3-46.el6\_6 updates 2.2 M

php-common x86\_64 5.3.3-46.el6\_6 updates 529 k

php-gd x86\_64 5.3.3-46.el6\_6 updates 111 k

php-pdo x86\_64 5.3.3-46.el6\_6 updates 79 k

php-xml x86\_64 5.3.3-46.el6\_6 updates 107 k

Transaction Summary

================================================================================

5. ZabbixServer --- 配置zabbix-server的数据库

        启动mysql，我的CentOS在安装的时候，就把mysql的软件给安装了，如果没有mysql，则需要去安装mysql，我这里就直接启动mysql就可以了

# /etc/init.d/mysqld start

        成功启动mysql的话，输入mysql，就可以进入mysql的命令行了。进入命令行，创建zabbix数据库

[root@localhost ~]# mysql

mysql> create database zabbix;

Query OK, 1 row affected (0.00 sec)

        在mysql中，查看刚刚创建的zabbix是否创建成功，如下图数据库中，有zabbix代表已经创建成功

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| test |

| zabbix |

+--------------------+

4 rows in set (0.00 sec)

        接下要要执行zabbix官方提供是sql语句。在安装基础依赖包的时候，已经在文件里面了。现在可以直接执行

[root@localhost ~]# cd /usr/share/doc/zabbix-server-mysql-2.4.6/create/

[root@localhost create]# mysql -uroot zabbix < schema.sql

[root@localhost create]# mysql -uroot zabbix < images.sql

[root@localhost create]# mysql -uroot zabbix < data.sql

6. ZabbixServer --- 配置zabbix-server的服务参数

        配置在以下路径里面 :

# vim /etc/zabbix/zabbix\_server.conf

        如果没有特殊需要的话，使用默认配置就可以了，我的mysql账户是root，所以需要修改DBuser

LogFile=/var/log/zabbix/zabbix\_server.log

LogFileSize=0

PidFile=/var/run/zabbix/zabbix\_server.pid

DBName=zabbix

DBUser=root

DBSocket=/var/lib/mysql/mysql.sock

StartPollers=5

CacheSize=256M

SNMPTrapperFile=/var/log/snmptt/snmptt.log

AlertScriptsPath=/usr/lib/zabbix/alertscripts

ExternalScripts=/usr/lib/zabbix/externalscripts

        修改zabbix的时区，有些系统并不是北京时间，如果是国外的时间，在展示出来的图的时间就是错的，为了保守起见，设置一下zabbix的时区

# vim /etc/httpd/conf.d/zabbix.conf

        修改以下字段

php\_value date.timezone Asia/Shanghai

        修改zabbix的php的时区

# vim /etc/php.ini

        修改以下字段

date.timezone = Asia/Shanghai

        启动Zabbix服务，启动httpd：

[root@localhost create]# /etc/init.d/zabbix-server start

[root@localhost create]# /etc/init.d/httpd start

7. 使用zabbix-get

        安装好了zabbix-agent和zabbix-server。在Server端试试能不能收到agent端的数据，可以使用zabbix-get。在安装mysql的时候，已经将zabbix-get安装过了。可以使用zabbix\_get -help命令来查看该命令怎么使用

[root@localhost create]# zabbix\_get -help

Zabbix get v2.4.6 (revision 54796) (10 August 2015)

usage: zabbix\_get [-hV] -s <host name or IP> [-p <port>] [-I <IP address>] -k <key>

Options:

-s --host <host name or IP> Specify host name or IP address of a host

-p --port <port number> Specify port number of agent running on the host. Default is 10050

-I --source-address <IP address> Specify source IP address

-k --key <key of metric> Specify key of item to retrieve value for

-h --help Display help information

-V --version Display version number

Example: zabbix\_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]"

        测试是否能收到数据，使用以下命令，发现是可以收到的。

// 测试是否可以收到系统的信息

[root@localhost create]# zabbix\_get -s 192.168.246.134 -p 10050 -k "system.uname"

Linux localhost.localdomain 2.6.32-573.el6.x86\_64 #1 SMP Thu Jul 23 15:44:03 UTC 2015 x86\_64

// 测试是否可以收到CPU的数据

[root@localhost create]# zabbix\_get -s 192.168.246.134 -p 10050 -k "system.cpu.load"

0.020000

8. 登录Zabbix的Web端

        在Server端，启动Zabbix服务，启动httpd：

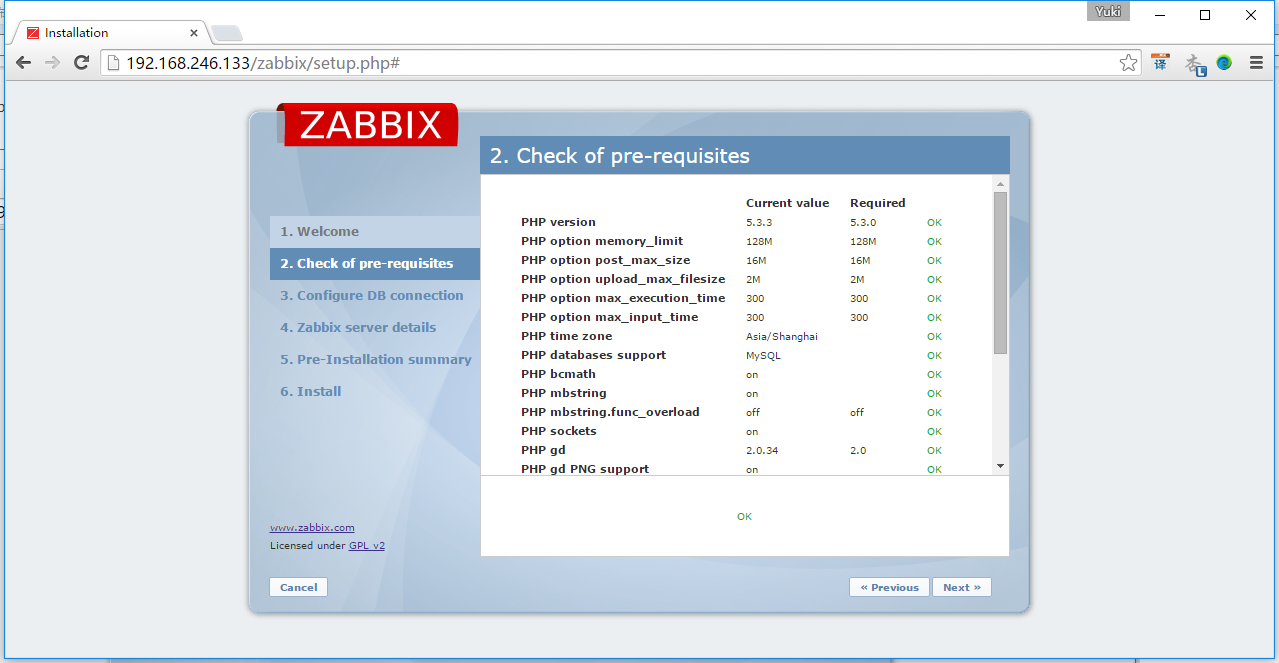
[root@localhost create]# /etc/init.d/zabbix-server start

[root@localhost create]# /etc/init.d/httpd start

        登录Zabbix的Web客户端，在浏览器中输入“<http://192.168.246.133/zabbix/index.php>”，默认会跳转至Zabbix的setup引导页面：

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192443169-2147459964.png)

        点击Next

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192446340-931916244.png)

        点击Next

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192449669-775174622.png)

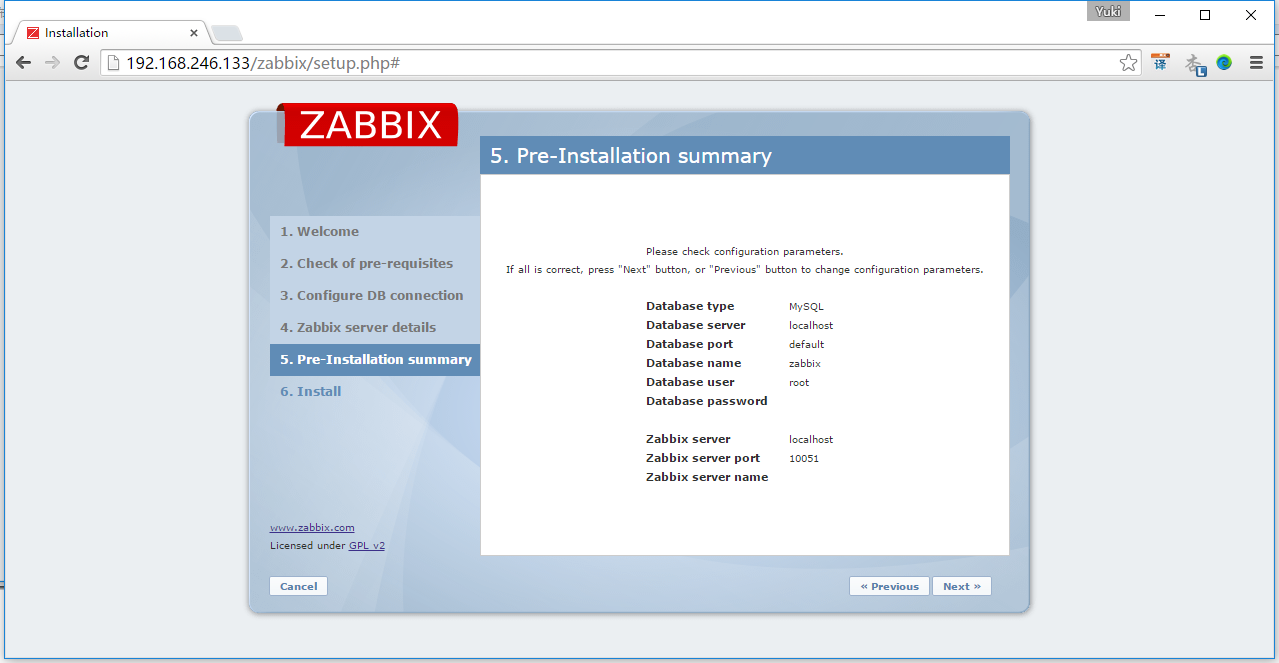
        点击Test connection，否则无法点击Next，  测试连接成功后，点击Next

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192452669-1057308292.png)

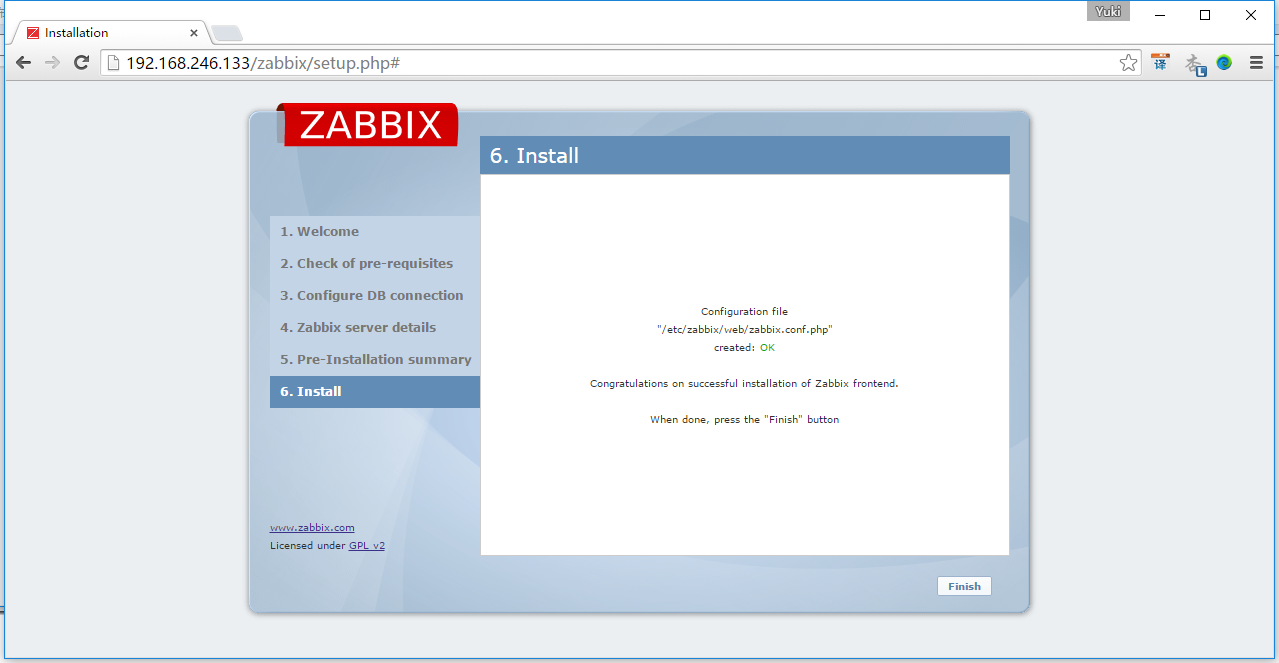
        设置端口，可以直接使用默认设置的端口10051

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192455684-931733296.png)

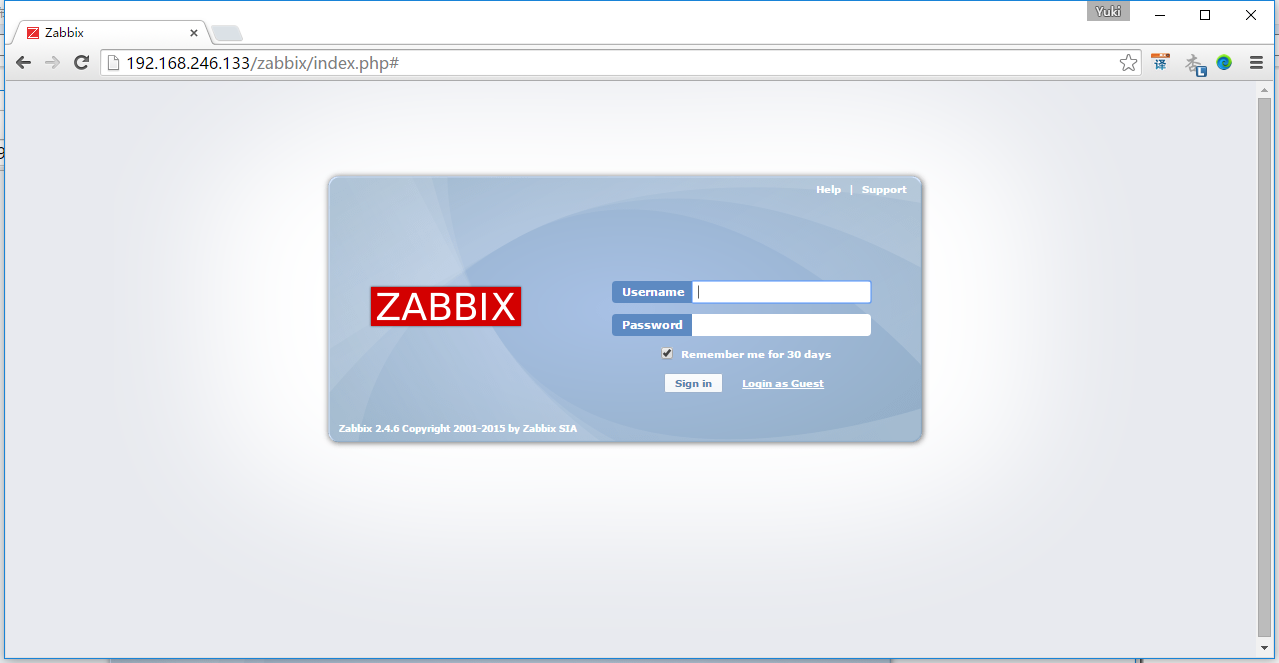
        点击Next

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192458981-1057581296.png)

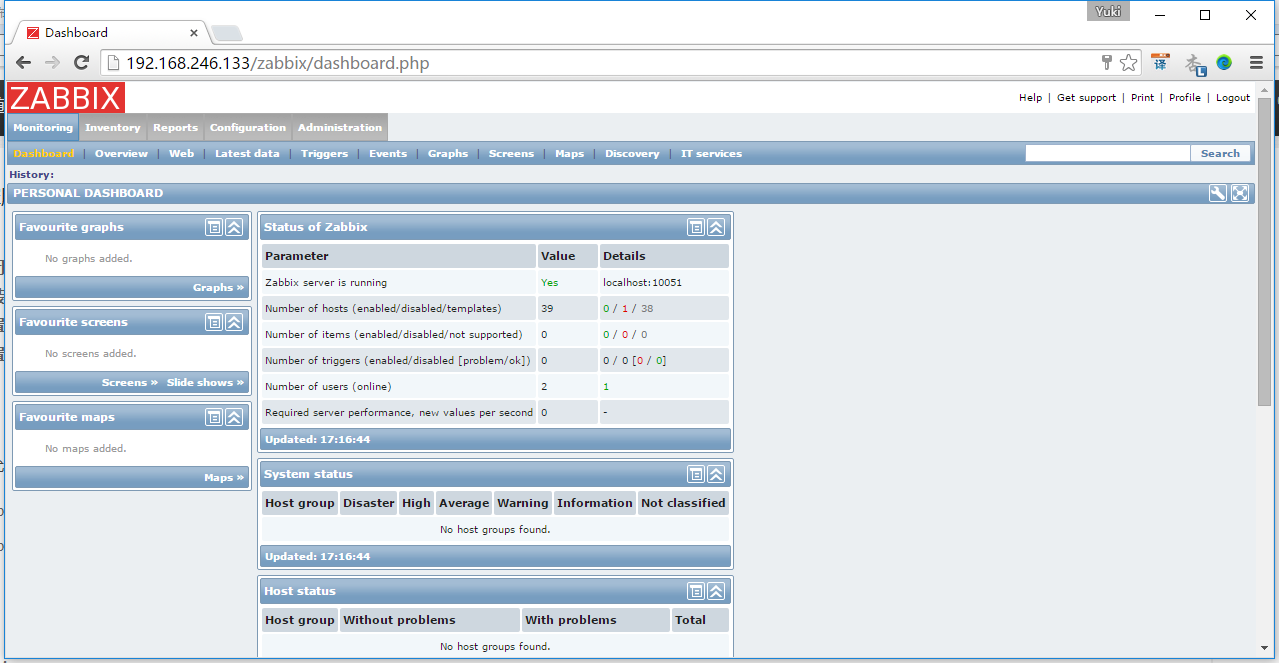
        点击Next

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192502278-851822042.png)

        点击Finish，完成初始化引导

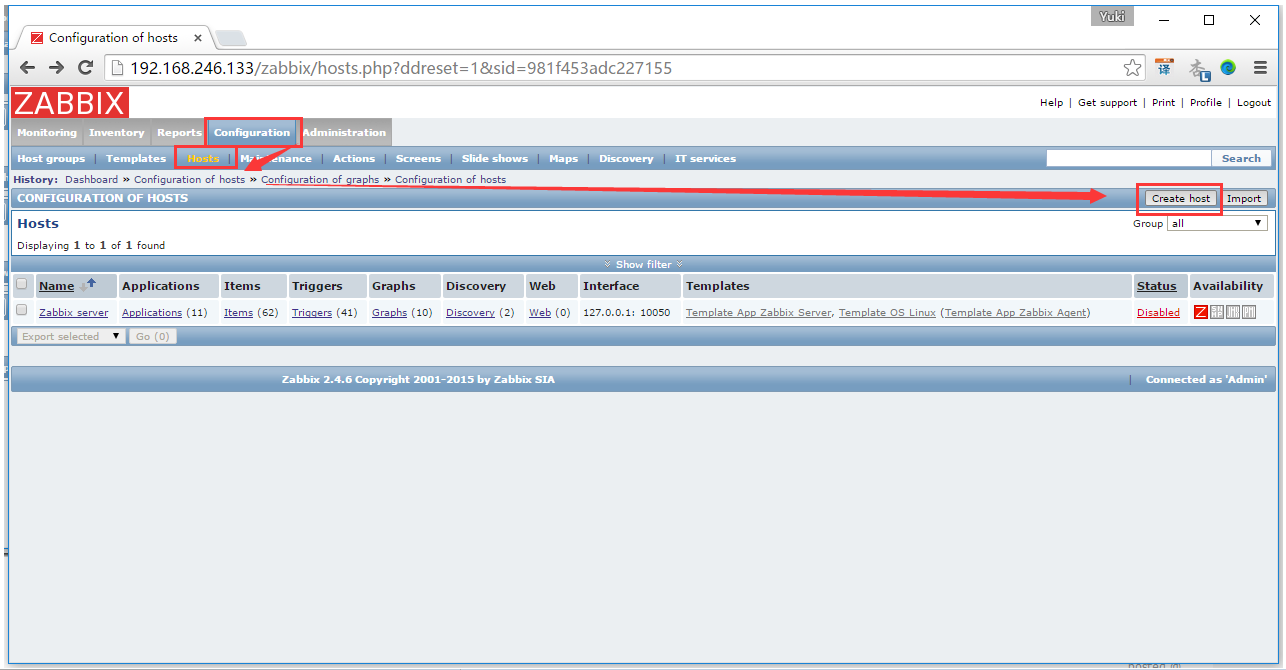
[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192505122-1309629252.png)

        输入默认的账号密码，账号：admin， 密码zabbix

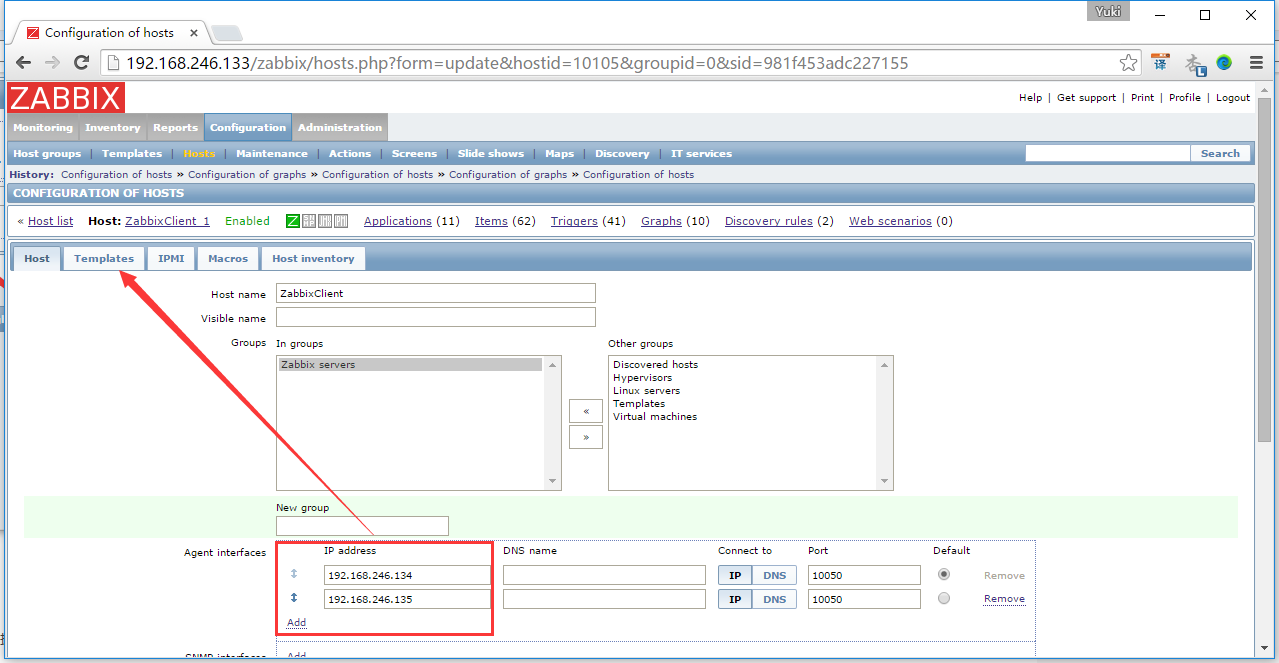
[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192508497-1146541381.png)

9. 在Web配置zabbix监控agent

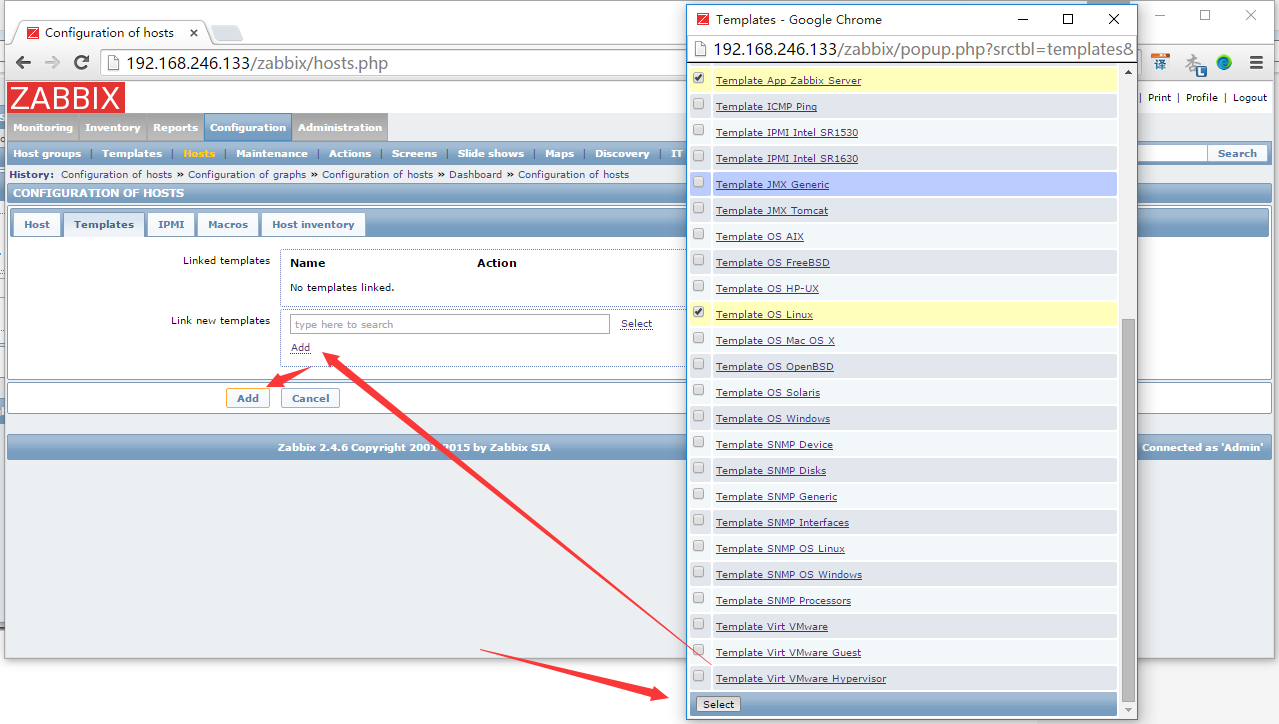
        点击下图的Create host

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908202233997-630987813.png)

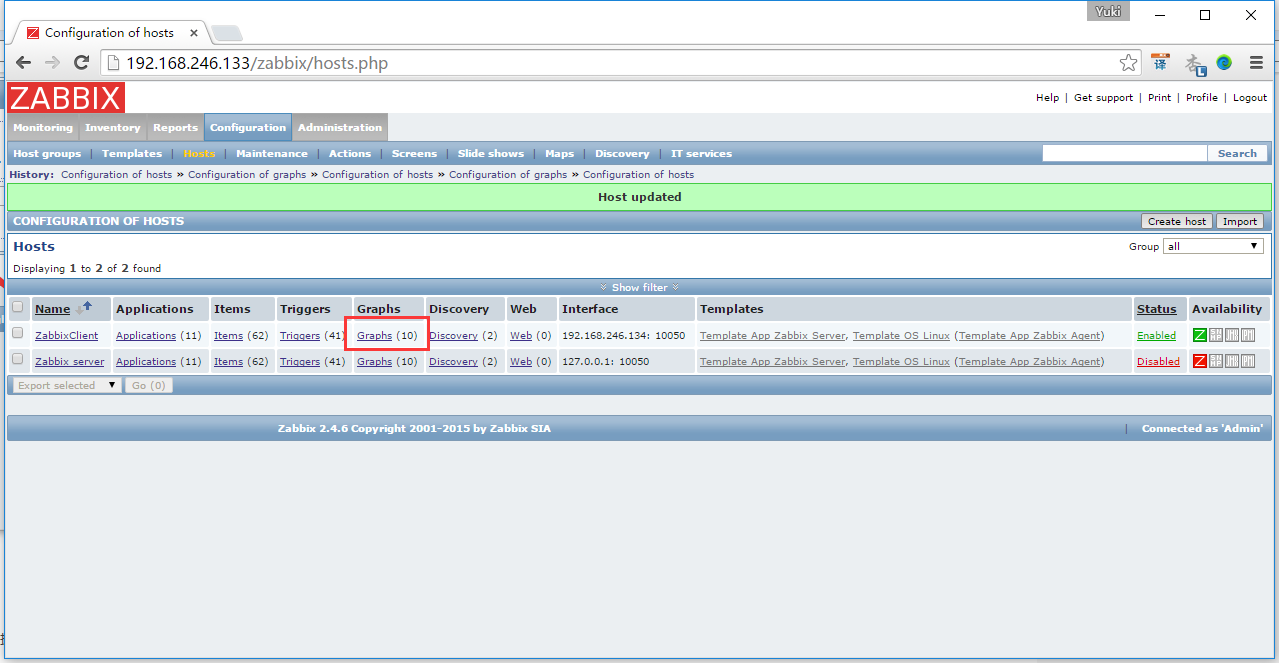
        按下图配置，主意要输入agent的IP和端口，端口默认是10050

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192513325-1569415391.png)

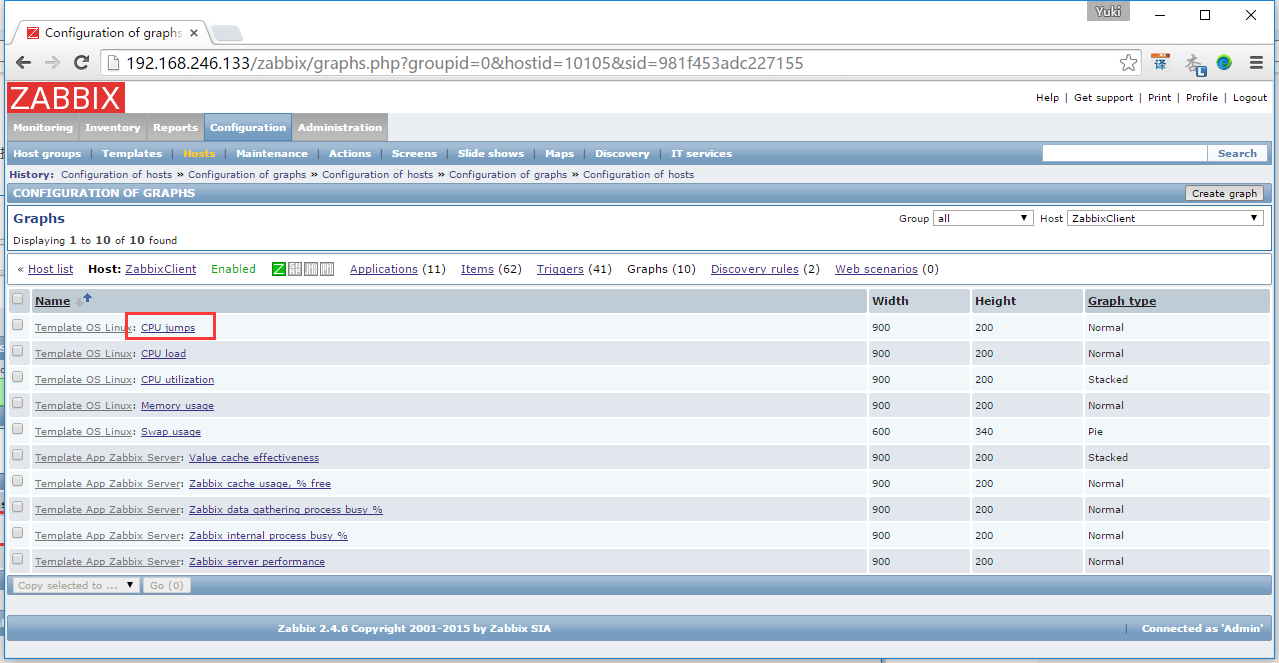
        点击select，选择Linux OS和zabbix Server的模板，点击Select->Add->Add，如下图

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192516231-2079582123.png)

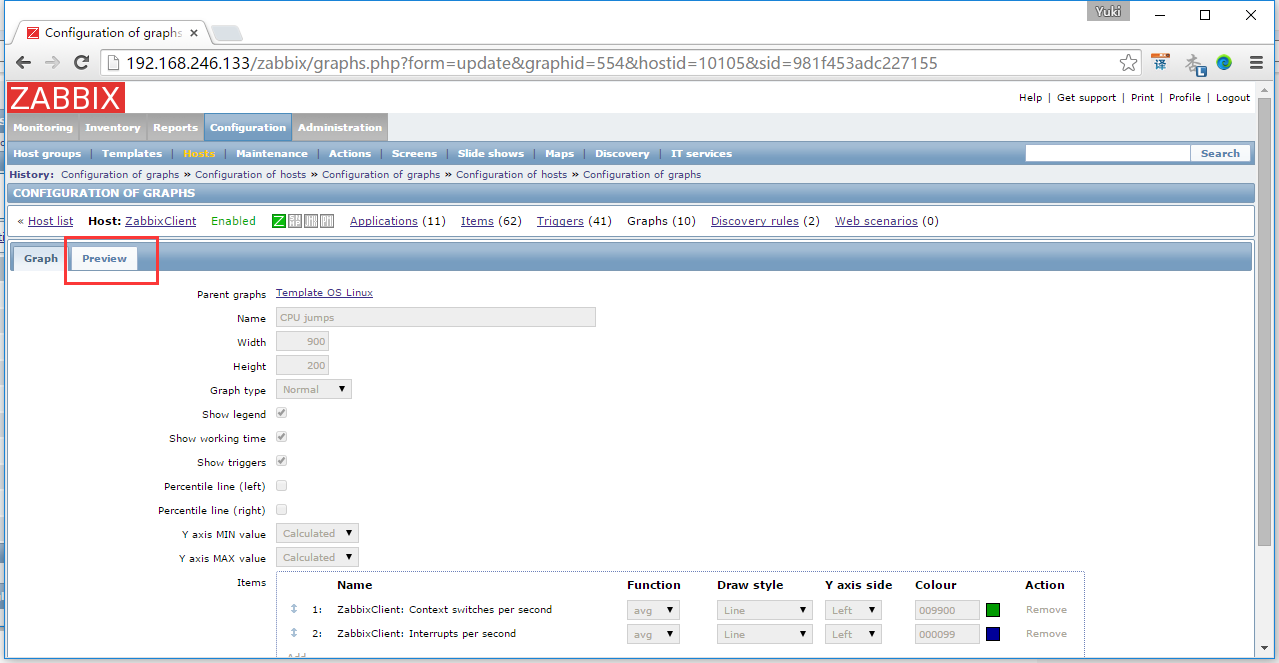
        添加了以后就已经自动启动了，可以点击Graphs，查看是否有数据过来

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192519106-559881840.png)

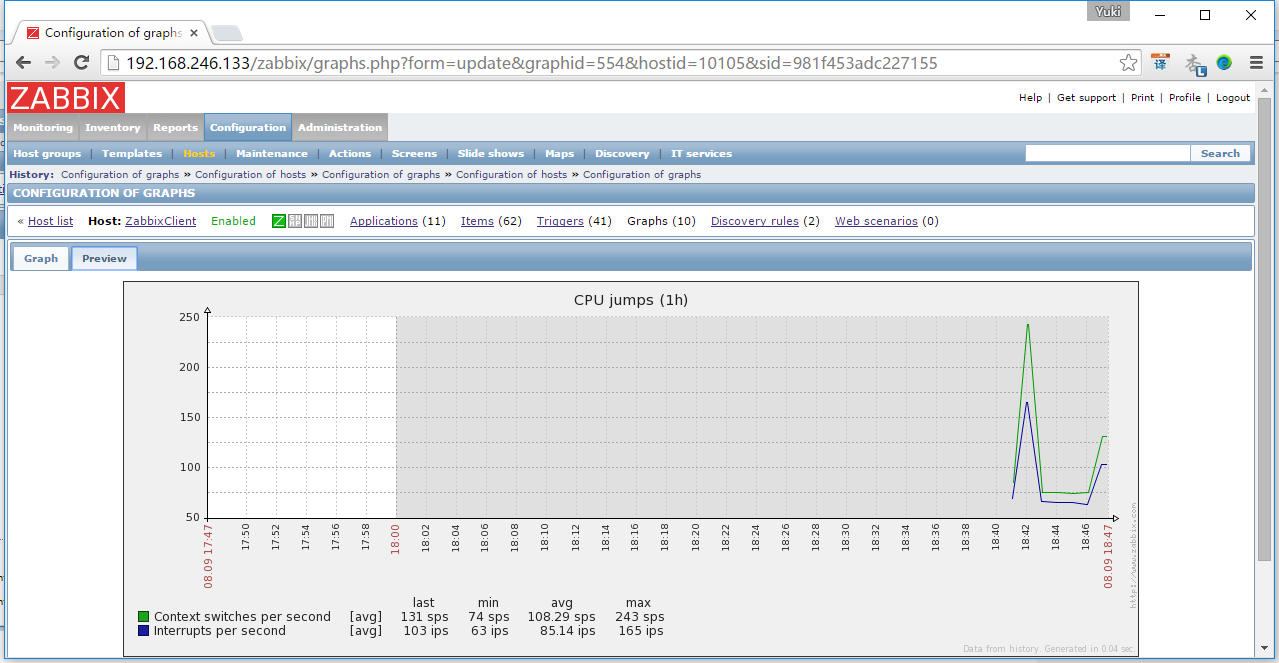
        点击CPU jumps

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192522262-1189461886.png)

        点击preview

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192525059-1575511719.png)

        就可以看到agent那边有数据传过来了

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192527450-785718562.png)

10. 安装Grafana

        在ZabbixServer服务器上，安装Grafana

yum install https://grafanarel.s3.amazonaws.com/builds/grafana-2.1.1-1.x86\_64.rpm

        下载Grafana的Zabbix插件，在git上下载，如果无法运行下面命令的话，可以先“# yum install git”安装这个命令

[root@localhost ~]# git clone https://github.com/linglong0820/grafana-zabbix

        将插件复制到Grafana文件下

[root@localhost ~]# cp -r grafana-zabbix/zabbix/ /usr/share/grafana/public/app/plugins/datasource/

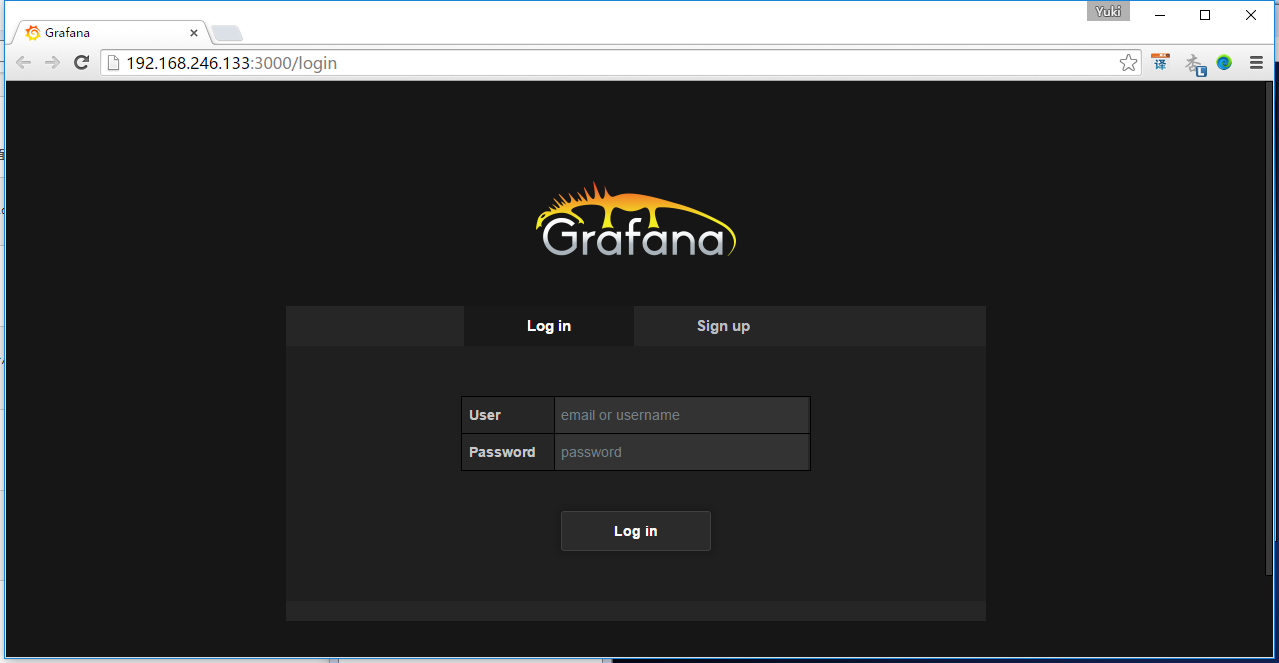
        启动Grafana

[root@localhost ~]# service grafana-server start

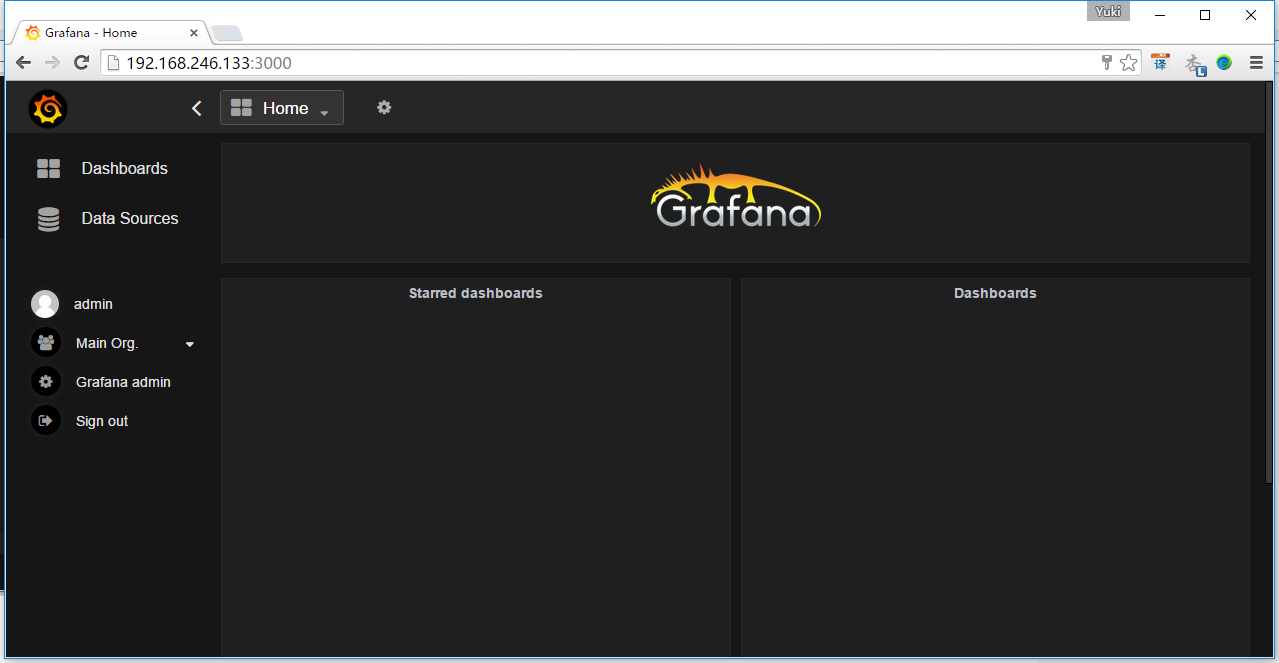
[root@localhost ~]# chkconfig grafana-server on

11. 登录Grafana并且配置

        启动Grafana后，在浏览器中输入“[http://192.168.246.133:3000/login](http://www.cnblogs.com/yyhh/archive/2015/09/08/192.168.246.133:3000/login)”就可以看到页面了。

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192530028-1294314319.png)

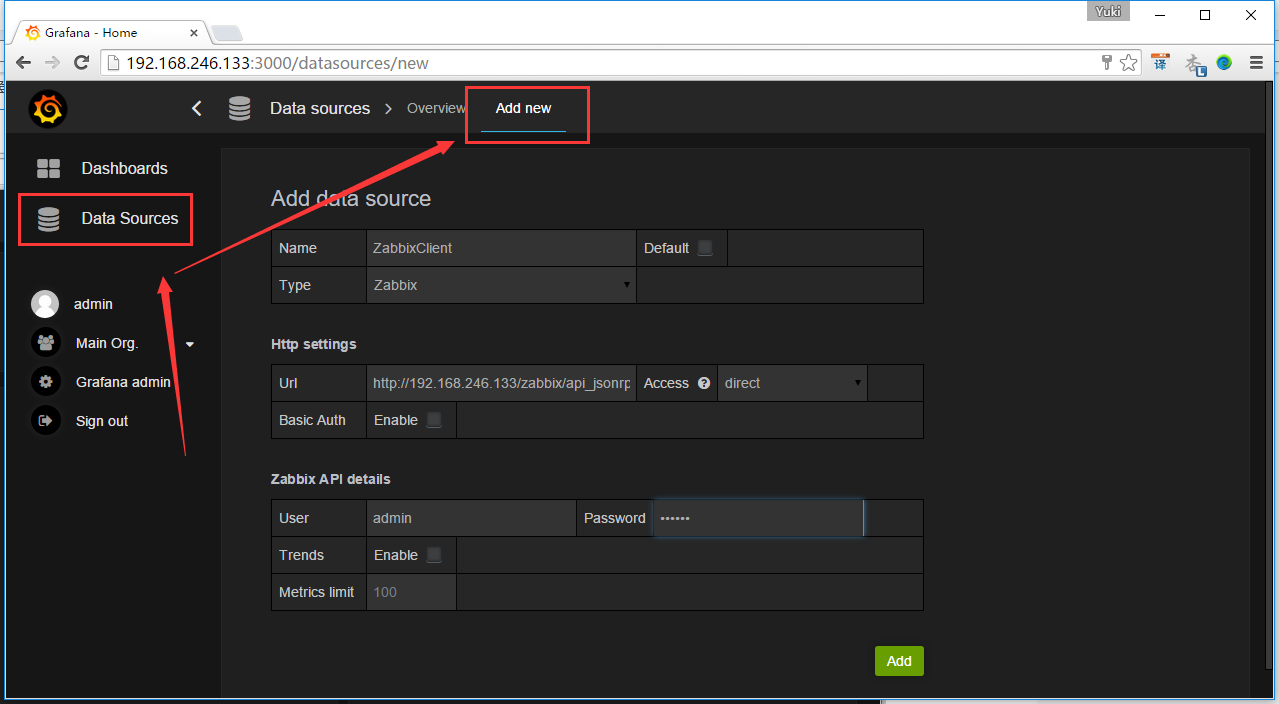
        输入默认用户名和密码，用户名：admin， 密码：admin， 点击登录

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192531653-2046354514.png)

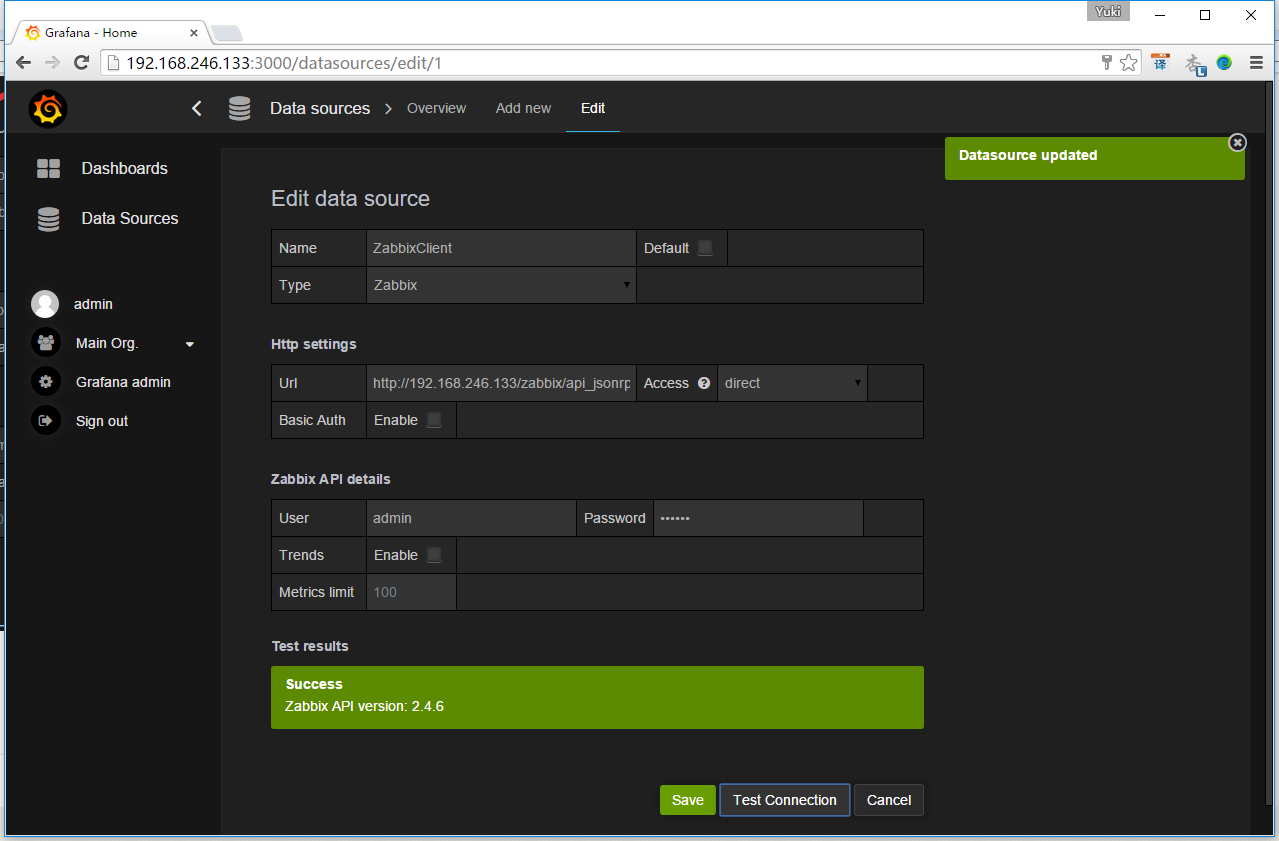
        按如下配置

        【主意1】Zabbix的API地址为“<http://192.168.246.133/zabbix/api_jsonrpc.php>”，其中IP的安装zabbix-server的服务器IP

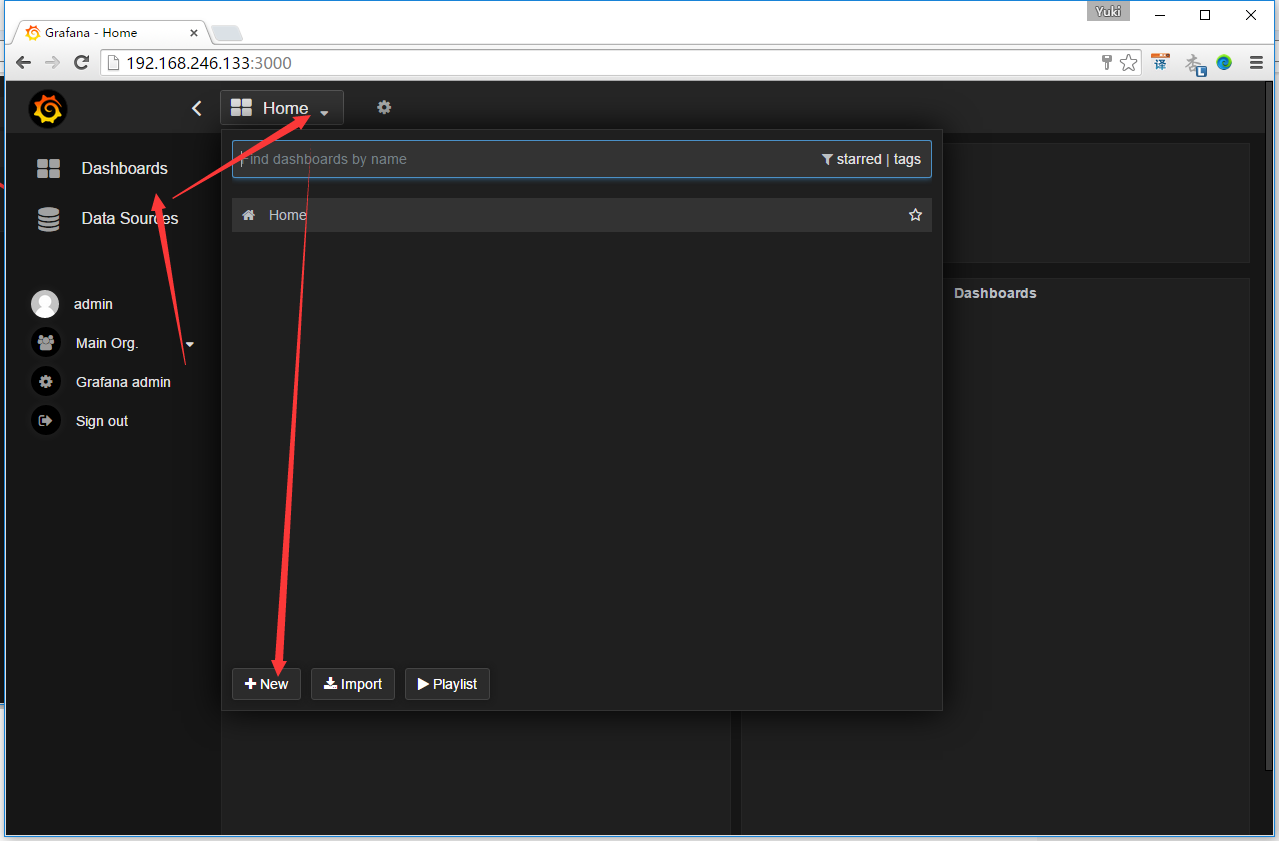
        【主意2】Zabbix的API账号密码就是Zabbix的Web端登录账号和密码，账号：admin， 密码zabbix

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192533465-1704730152.png)

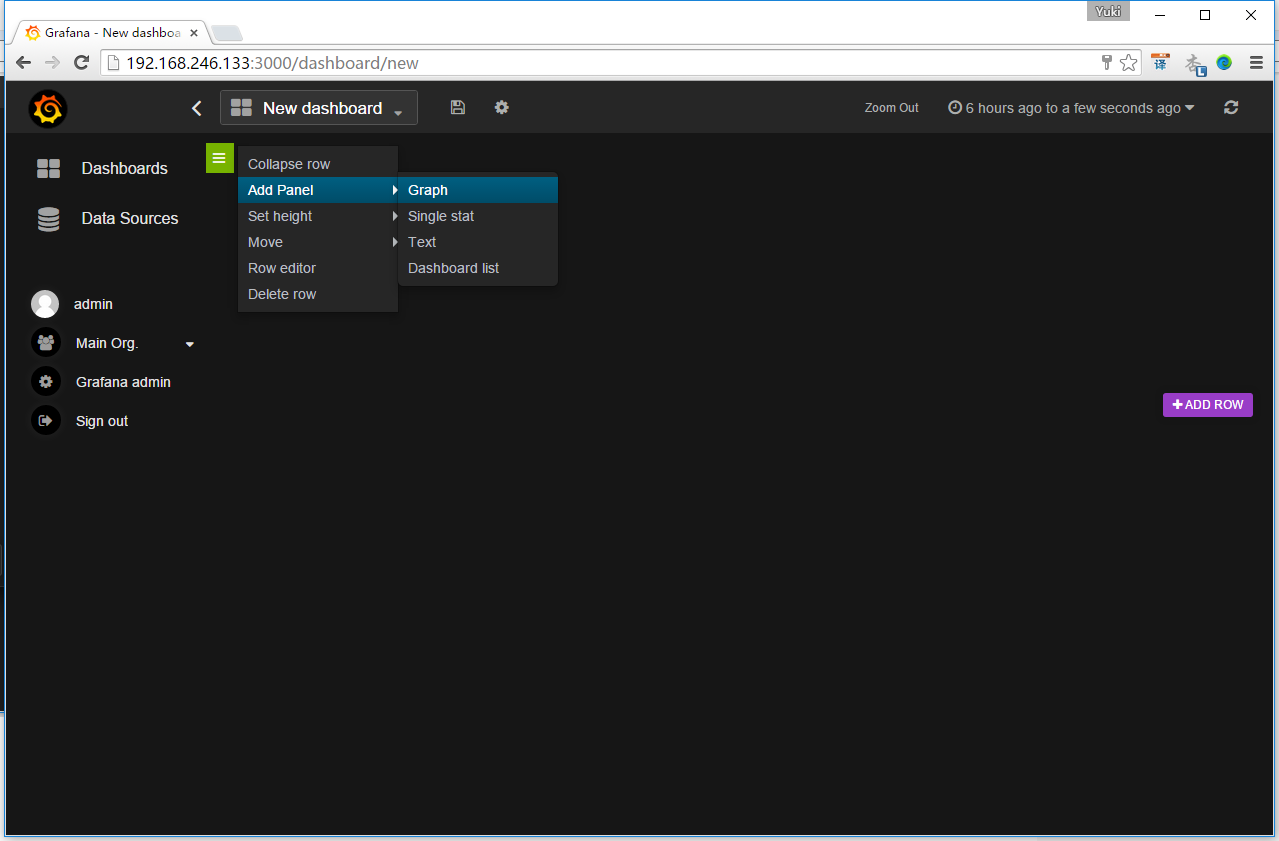
        点击Add后，点击Test Connection测试是否可以连接。提示“Success”说明连接成功，点击Save

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192535778-1246823532.png)

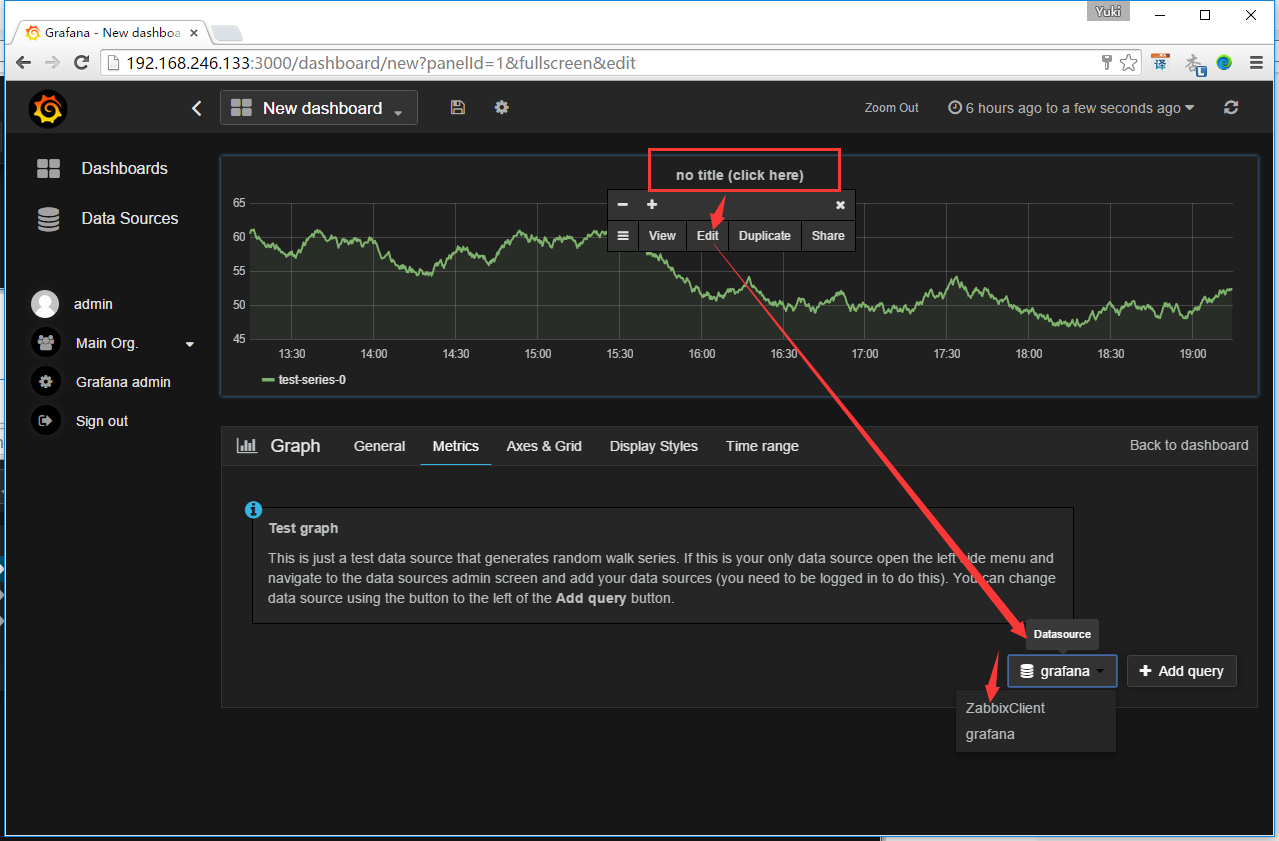
        点击Dashboard，去主目录设置， 点击new，新建一个Home

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192538356-1595970052.png)

        新建一个Graph

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192540559-2087262480.png)

        点击下列红框处->选择Edit->选择数据库ZabbixClient

[](http://images2015.cnblogs.com/blog/626593/201509/626593-20150908192543372-1685300580.png)

        然后配置一下，就可以出现华丽丽的数据了，下图都是官方的示意图：

分类: [Zabbix](http://www.cnblogs.com/yyhh/category/732409.html)