# 镜像

https://hub.alauda.cn/repos/googlelib/cadvisor

This repo is synced from <https://hub.docker.com/r/google/cadvisor/>

You can pull the images in this doc by using:

docker pull index.alauda.cn/googlelib/cadvisor

开发者平台 https://dev.aliyun.com/detail.html?spm=5176.1972343.2.2.cahx64&repoId=2401

docker pull registry.cn-hangzhou.aliyuncs.com/acs-sample/google-cadvisor:v0.23



# tutum/influxdb - Docker Hub

https://hub.docker.com/r/tutum/influxdb/

tutum-docker-influxdb

InfluxDB image

**Usage**

To create the image tutum/influxdb, execute the following command on tutum-docker-influxdb folder:

docker build -t tutum/influxdb .

You can now push new image to the registry:

docker push tutum/influxdb

**Tags**

tutum/influxdb:latest -> influxdb 0.9.6.1

tutum/influxdb:0.9 -> influxdb 0.9.6.1

tutum/influxdb:0.8.8 -> influxdb 0.8.8

**Running your InfluxDB image**

Start your image binding the external ports 8083 and 8086 in all interfaces to your container. Ports 8090 and8099 are only used for clustering and should not be exposed to the internet:

docker run -d -p 8083:8083 -p 8086:8086 tutum/influxdb

Docker containers are easy to delete. If you delete your container instance and your cluster goes offline, you'll lose the InfluxDB store and configuration. If you are serious about keeping InfluxDB data persistently, then consider adding a volume mapping to the containers /data folder:

docker run -d --volume=/var/influxdb:/data -p 8083:8083 -p 8086:8086 tutum/influxdb

**Note**: influxdb:0.9 is **NOT** backwards compatible with 0.8.x. If you need version 0.8.x, please run:

docker run -d -p 8083:8083 -p 8086:8086 tutum/influxdb:0.8.8

**Configuring your InfluxDB**

Open your browser to access localhost:8083 to configure InfluxDB. Fill the port which maps to 8086. There is no default user anymore in version 0.9 but you can set*auth-enabled: true*in the config.toml.

Alternatively, you can use RESTful API to talk to InfluxDB on port 8086. For example, if you have problems with the initial database creation for version 0.9.x, you can use the new influx cli tool to configure the database. While the container is running, you launch the tool with the following command:

docker exec -ti influxdb-container-name /usr/bin/influx

Visit https://enterprise.influxdata.com to register for updates, InfluxDB server management, and monitoring.

Connected to http://localhost:8086 version 0.9.6.1

InfluxDB shell 0.9.6.1

>

**Initially create Database**

Use -e PRE\_CREATE\_DB="db1;db2;db3" to create database named "db1", "db2", and "db3" on the first time the container starts automatically. Each database name is separated by ;. For example:

docker run -d -p 8083:8083 -p 8086:8086 -e ADMIN\_USER="root" -e INFLUXDB\_INIT\_PWD="somepassword" -e PRE\_CREATE\_DB="db1;db2;db3" tutum/influxdb:latest

Alternatively, create a database and user with the InfluxDB 0.9 shell:

> CREATE DATABASE db1

> SHOW DATABASES

name: databases

---------------

name

db1

> USE db1

> CREATE USER root WITH PASSWORD 'somepassword' WITH ALL PRIVILEGES

> GRANT ALL PRIVILEGES ON db1 TO root

> SHOW USERS

user admin

root true

For additional Administration methods with the InfluxDB 0.9 shell, check out the [Administration](https://influxdb.com/docs/v0.9/administration/administration.html) guide on the InfluxDB website.

**Initially execute influxql script (Available only in influxdb:0.9)**

Use -v /tmp/init\_script.influxql:init\_script.influxql:ro if you want that script to been executed on the first time the container starts automatically. Each influxql command on separated line. For example:

* Docker run command
* docker run -d -p 8083:8083 -p 8086:8086 -e ADMIN\_USER="root" -e INFLUXDB\_INIT\_PWD="somepassword" -v /tmp/init\_script.influxql:init\_script.influxql:ro tutum/influxdb:latest
* The influxdb script
* CREATE DATABASE mydb
* CREATE USER writer WITH PASSWORD 'writerpass'
* CREATE USER reader WITH PASSWORD 'readerpass'
* GRANT WRITE ON mydb TO writer
* GRANT READ ON mydb TO reader

**SSL support (Available only in influxdb:0.8.8)**

By default, Influx DB uses port 8086 for HTTP API. If you want to use SSL API, you can set SSL\_SUPPORT totrue as an environment variable. In that case, you can use HTTP API on port 8086 and HTTPS API on port 8084. Please do not publish port 8086 if you want to only allow HTTPS connection.

If you provide SSL\_CERT, system will use user provided SSL certificate. Otherwise the system will create a self-signed certificate, which usually has an unauthorized certificate error, not recommend.

The cert file should be a combination of Private Key and Public Certificate. In order to pass it as an environment variable, you need specifically convert newline to \n(two characters). In order to do this, you can simply run the command awk 1 ORS='\\n' <your\_cert.pem>. For example:

docker run -d -p 8083:8083 -p 8084:8084 -e SSL\_SUPPORT="True" -e SSL\_CERT="`awk 1 ORS='\\n' ~/cert.pem`" tutum/influxdb:latest

**Graphite API support**

InfluxDB has plugin to support the [Graphite Carbon API](http://graphite.readthedocs.org/en/1.0/feeding-carbon.html). This can be customized via the following variables:

* GRAPHITE\_DB: name of the database the graphite plugin shall write the incoming metrics to
* GRAPHITE\_BINDING: by default the graphite plugin listens on ':2003'. You can provide any ipaddress:port
* GRAPHITE\_PROTOCOL: 'udp' or 'tcp' (default)
* GRAPHITE\_TEMPLATE: By default the template is set to instance.profile.measurement\* which will parse a metric and create tags from it

docker run -d -p 8083:8083 -p 8086:8086 -p 2015:2015 -e ADMIN\_USER="root" -e INFLUXDB\_INIT\_PWD="somepassword" -e PRE\_CREATE\_DB=my\_db -e GRAPHITE\_DB="my\_db" -e GRAPHITE\_BINDING=':2015' -e GRAPHITE\_PROTOCOL="udp" -e GRAPHITE\_template="tag1.tag2.tag3.measurement\*" tutum/influxdb

More details on the configuration of InfluxDB's graphite plugin can be found at:<https://github.com/influxdb/influxdb/blob/master/services/graphite/README.md>

**Collectd support**

InfluxDB has a plugin to support the [collectd network plugin](https://collectd.org/wiki/index.php/Plugin:Network). This can be customized via the following variables:

* COLLECTD\_DB: name of the database the collectd plugin shall write the incoming metrics to
* COLLECTD\_BINDING: by default the collectd plugin listens on ':25826'. You can provide anyipaddress:port
* COLLECTD\_RETENTION\_POLICY: custom retention policy
* types.db: default types.db from collectd version 5.5.0 is provided. For custom types consider adding a volume mapping for /usr/share/collectd/types.db

docker run -d -p 8083:8083 -p 8086:8086 -p 25826:25826/udp -e ADMIN\_USER="root" -e INFLUXDB\_INIT\_PWD="somepassword" -e PRE\_CREATE\_DB=my\_db -e COLLECTD\_DB="my\_db" -e COLLECTD\_BINDING=':25826' -e COLLECTD\_RETENTION\_POLICY="mypolicy" tutum/influxdb

More details on the configuration of InfluxDB's collectd plugin can be found at:<https://github.com/influxdb/influxdb/blob/master/services/collectd/README.md>

**UDP support**

If you provide a UDP\_DB, influx will open a UDP port (4444 or if provided UDP\_PORT) for reception of events for the named database.

docker run -d -p 8083:8083 -p 8086:8086 -p 4444:4444/udp --expose 8090 --expose 8099 --expose 4444 -e UDP\_DB="my\_db" tutum/influxdb

**Clustering (Available in influxdb:0.9.4.2-1)**

# (make sure firewall allows ports 8088, 8089)

docker run -p 8088:8088 -e FORCE\_HOSTNAME=192.168.0.1:8088 -t tutum/influxdb

docker run -p 8089:8088 -e FORCE\_HOSTNAME=192.168.0.1:8089 -e JOIN=192.168.0.1:8088 -t tutum/influxdb

**Clustering (Available in influxdb:0.8.8)**

Use :

* -e SEEDS="host1:8090, host2:8090" to pass seeds nodes to your container.
* -e REPLI\_FACTOR=x where x is the replicator factor of shards through the cluster (defaults to 1)
* -e FORCE\_HOSTNAME="auto" to force the hostname in the config file to be set to the container IPv4 eth0 address (useful to test clustering on a single docker host)
* -e FORCE\_HOSTNAME="<whatever>" to force the hostname in the config file to be set to 'whatever'

Example on a single docker host:

* Launch first container:
* docker run -p 8083:8083 -p 8086:8086 --expose 8090 --expose 8099 \
* -e FORCE\_HOSTNAME="auto" -e REPLI\_FACTOR=2 \
* -d --name masterinflux tutum/influxdb
* Then launch one or more "slaves":
* docker run --link masterinflux:master -p 8083 -p 8086 --expose 8090 --expose 8099 \
* -e SEEDS="master:8090" -e FORCE\_HOSTNAME="auto" \

-d tutum/influxdb

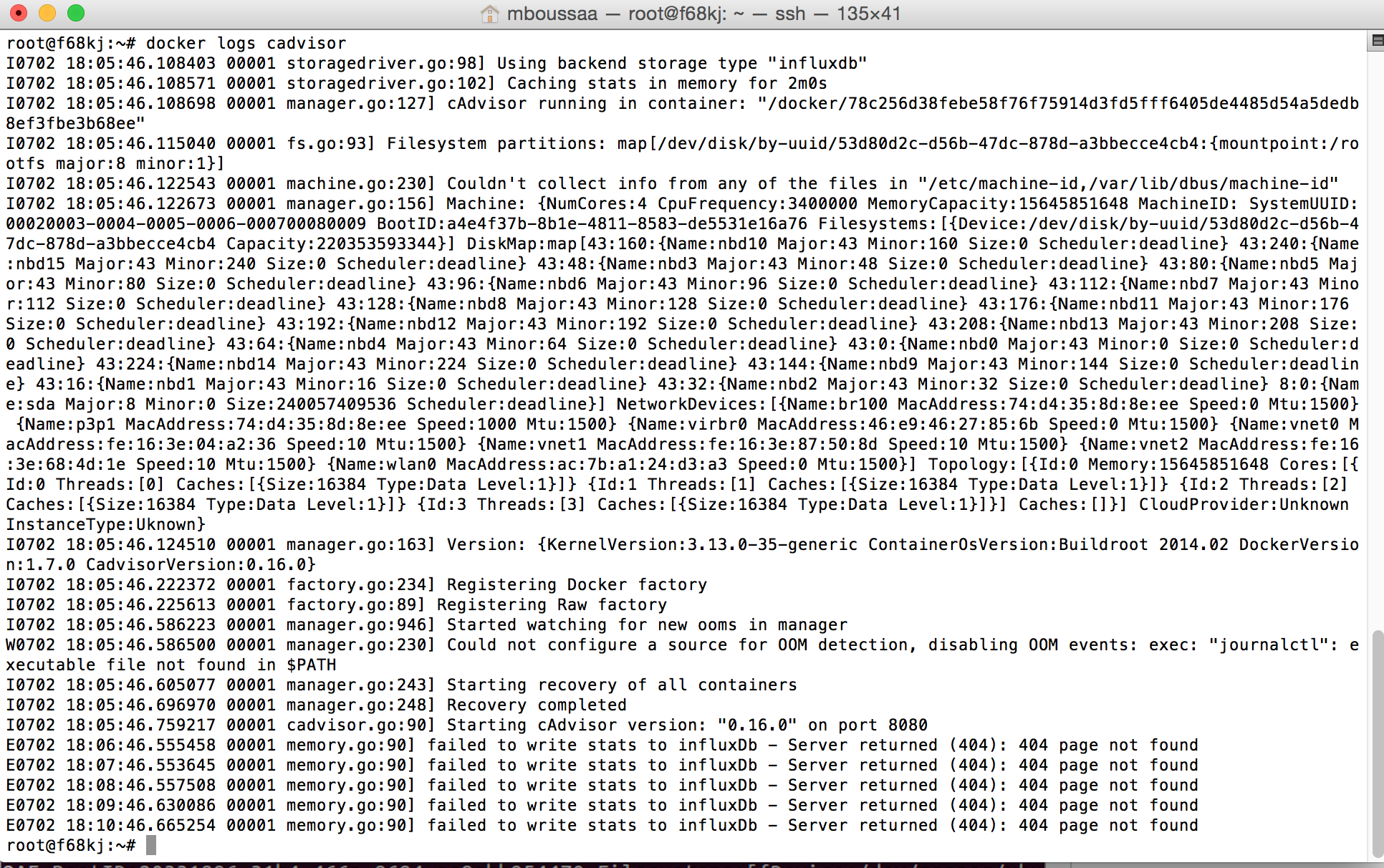
# Cadvisor fails to write stats to influxDB · Issue #794 · google/cadvisor

https://github.com/google/cadvisor/issues/794

docker pull registry.cn-hangzhou.aliyuncs.com/acs-sample/google-cadvisor:v0.23

v0.23已经支持了

Hi,  
Cadvisor fails to write stats into influxDB

Here the logs:  
[](https://cloud.githubusercontent.com/assets/5878565/8483984/0d1c49f4-20f5-11e5-881f-5accffa94063.png)

I am starting Cadvisor like this:  
docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro --publish=8080:8080 --detach=true --name=cadvisor --restart=always google/cadvisor:latest --logtostderr -storage\_driver=influxdb -storage\_driver\_host=10.0.0.22:8086 -storage\_driver\_db=cadvisorDB

I am starting influxDB like this:  
docker run -d --name=influxdb -p 8083:8083 -p 8086:8086 --restart=always -expose 8090 --expose 8099 tutum/influxdb

When I execute a sample request on InfluxDB like this "select \* from stats" it shows:  
ERROR: {"error":"error parsing query: found STATS, expected identifier at line 1, char 15

Thank you for help.

You are using influxdb v0.9 that is not currently supported by cadvisor, see [#743](https://github.com/google/cadvisor/issues/743).  
You can use influxdb 0.8 just by using tutum/influxdb:0.8.8

It's really look like you use influxdb 0.9 and not 0.8 because on the 0.8.8 web ui, when you create a database you have Shard Spaces box, but it's more a problem with influxdb than cadvisor IMO.

I juste made a test with by running influxdb with:  
docker run -d --name=influxdb -p 8083:8083 -p 8086:8086 --restart=always -expose 8090 --expose 8099 tutum/influxdb

And with a force refresh (remove my browser cache).

[**@svenmueller**](https://github.com/svenmueller) I tested a PR and it is now supported... [#1040 (comment)](https://github.com/google/cadvisor/pull/1040#issuecomment-176015272)

I built an image to help others at <https://hub.docker.com/r/marcellodesales/google-cadvisor/>... I will maintain it until the team releases a new version with it.

FYI: google/cadvisor:v0.20.5 should include support for InfluxDB v0.9

On Wed, Feb 3, 2016 at 5:30 PM, Marcello de Sales [notifications@github.com](mailto:notifications@github.com)  
wrote:

# Panamax Docker Application Template with cAdvisor, ElasticSearch, Grafana, and InfluxDB | Tutum Blog

https://blog.tutum.co/2014/08/25/panamax-docker-application-template-with-cadvisor-elasticsearch-grafana-and-influxdb/

Posted on [August 25, 2014](https://blog.tutum.co/2014/08/25/panamax-docker-application-template-with-cadvisor-elasticsearch-grafana-and-influxdb/) by [Bryan Lee](https://blog.tutum.co/author/kickingthetv/) — [11 Comments](https://blog.tutum.co/2014/08/25/panamax-docker-application-template-with-cadvisor-elasticsearch-grafana-and-influxdb/#comments)

[Panamax](http://panamax.io/) just released from Lucas Carlson and his team at CenturyLink Labs. They’ve created a contest for putting together the best Panamax templates, with various categories being judged. They’re giving over $100,000 worth of prizes. Definitely check it out when you have a chance! Today is the last day to submit!

I had recently talked about [using cAdvisor to monitor Docker containers](https://blog.tutum.co/2014/08/07/using-cadvisor-to-monitor-docker-containers/). But as I mentioned, running cAdvisor provides ephemeral data for your containers. For this data to be truly valuable, it needs to be logged in a database. And once we have that data logged in a database, wouldn’t it be nice to view it in a nicely charted graph? My thoughts exactly

The power of Panamax comes from application templates. Templates are similar to [fig](http://www.fig.sh/) files, but once you define your application template it can be shared with anyone. You can even search the database to find publicly available templates built by others. Now all of your hard work configuring and tweaking each of your images won’t be in vain. You just have to get it working once, and now you can reproduce the exact same technology stack on demand.

cAdvisor doesn’t record data historically. We modified it so that when it’s first launched with a Docker link it will automatically connect to InfluxDB and use it as its datastore backend.

Our modified version of cAdvisor saves the step of having to normally pass flags from cAdvisor to InfluxDB to connect the two.

Why use Grafana? Doesn’t InfluxDB have an interface? Grafana let’s us store our searches, and display the results in pretty looking graphs.

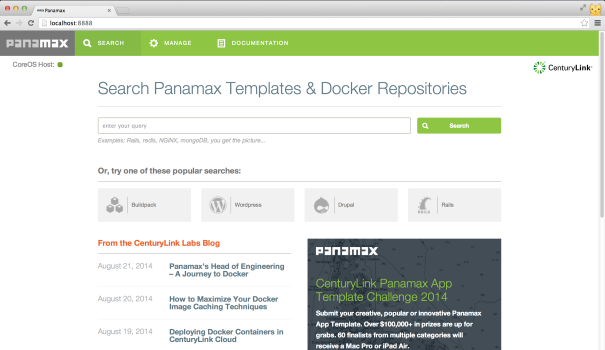
So the first step is to fire up Panamax. You can find the install directions [here](http://panamax.io/get-panamax/).

Get started with:

panamax init

Once it has successfully launched, a new browser window will be automatically opened to:

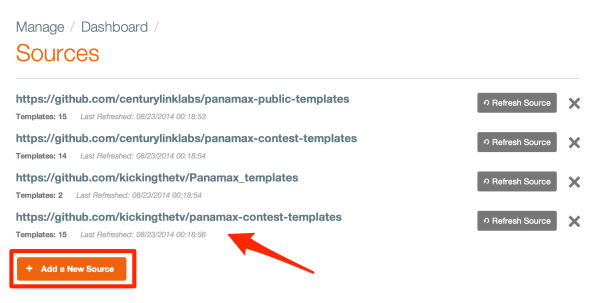
localhost:8888

[](https://tutumcloud.files.wordpress.com/2014/08/panamax_dash.png)

Now we need to load the proper source repositories. Just click on the **Manage** tab and type in the repository name.

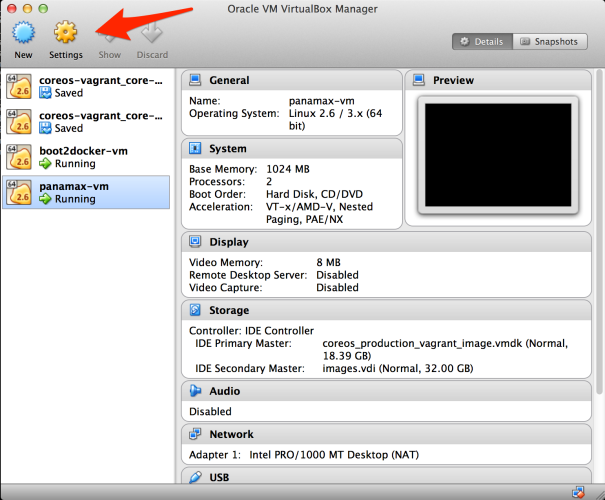
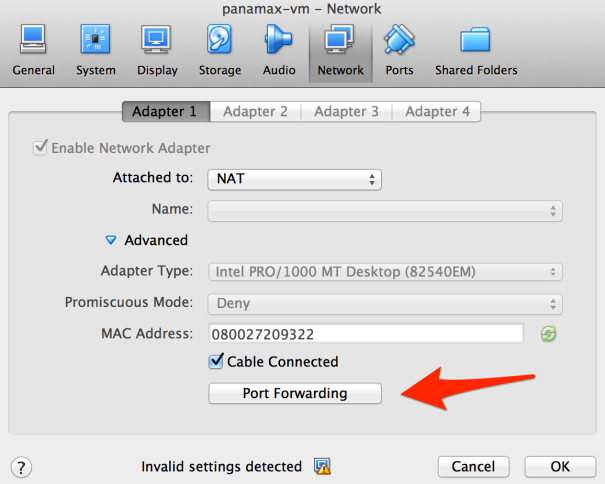
We’ll be adding:

kickingthetv/panamax-contest-templates

[](https://tutumcloud.files.wordpress.com/2014/08/panamax_sources.png)

Next we’re going to set up our ports. We’ll be forwarding our virtual box ports to match the ports**ElasticSearch**, **Grafana**, and **InfluxDB**listen on.

I’ll launch virtualbox by just typing in the terminal: virtualbox

[](https://tutumcloud.files.wordpress.com/2014/08/virtualbox_dash.png) We’ll head over to panamax-vm and click **Settings** then **Network** then **Port Forwarding.**  
[](https://tutumcloud.files.wordpress.com/2014/08/panamax_vm.png)

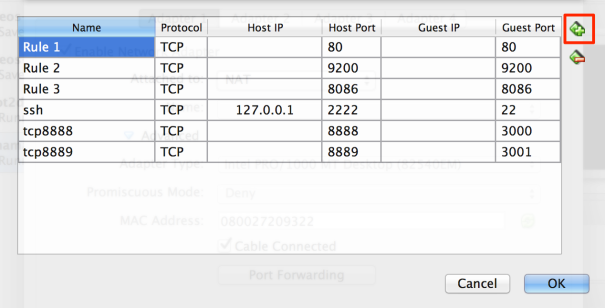
From there click on the little green + icon and we’ll be adding in three port mappings:

**ElasticSearch**: 9200:9200

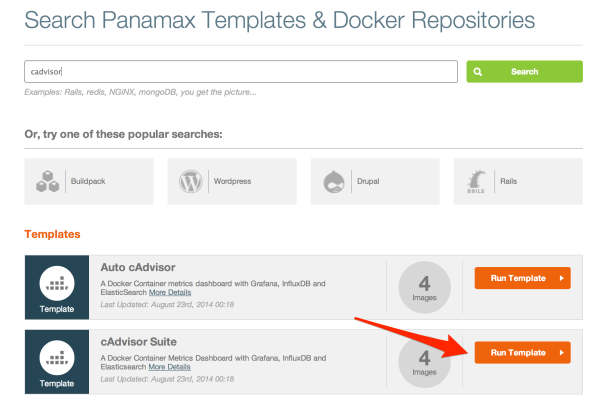
**Grafana**: 80:80

**InfluxDB**: 8086:8086

Once you’re done it should look like this:

[](https://tutumcloud.files.wordpress.com/2014/08/port_forwarding.png)

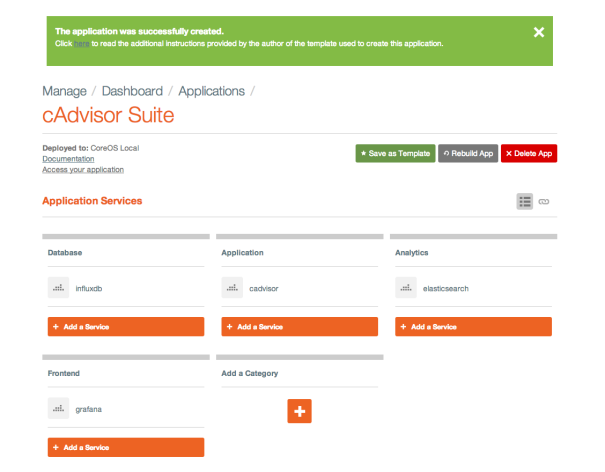
Now that we’re all setup, we’ll head back over to our browser where we have Panamax running. Now we have to search and find the template that I created. It’s called cAdvisor Suite.

[](https://tutumcloud.files.wordpress.com/2014/08/panamax_search.png)

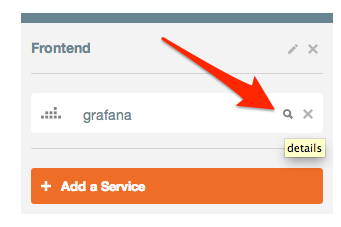
Go ahead and click on Run Template

You’ll be taken to the Application Services Dashboard. Another neat aspect of Panamax is the ability to drag and drop services into different categories that you can custom name.

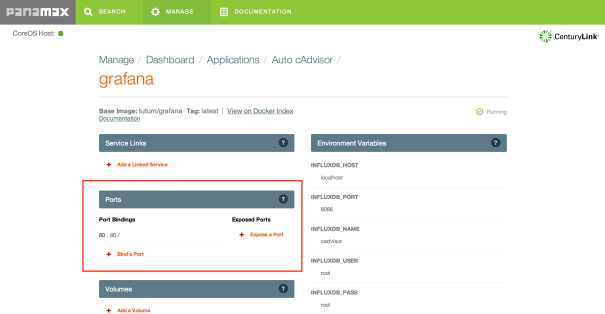
I went ahead and assigned **InfluxDB** to **Database**, **cAdvisor** to **Application**, **ElasticSearch** to**Analytics** and **Grafana** to **Frontend**.

[](https://tutumcloud.files.wordpress.com/2014/08/panamax_cadvisor.png)

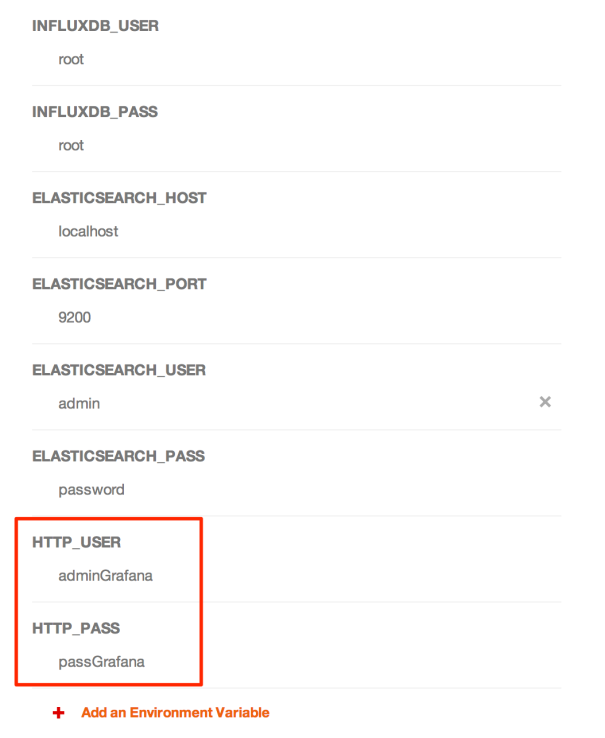
We’re going to mouse over **Grafana** and click on the little magnifying glass on the right-hand side.

[](https://tutumcloud.files.wordpress.com/2014/08/panamax_cadvisor2.png)

From here we can see which port our service is listening on.

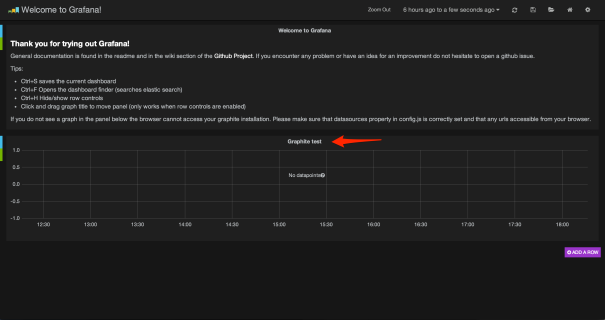
[](https://tutumcloud.files.wordpress.com/2014/08/panamax_grafana_ports.png)

Scrolling down on the right hand side, we can see all of the passwords for each of our services. Make sure to take note of the **Grafana** username and password.

[](https://tutumcloud.files.wordpress.com/2014/08/panamax_grafana_pass.png)

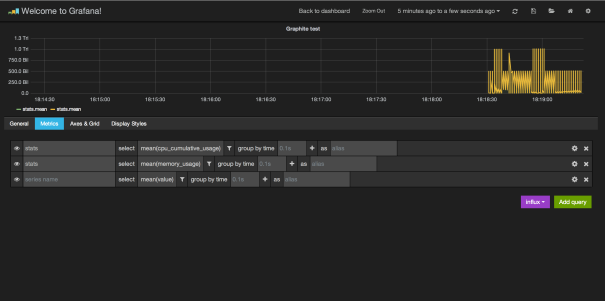
Next we’re going to access **Grafana**. Go to your web browser and type in: localhost:80

You’ll be prompted for your **Grafana** username and password. Once you successfully login you’ll be brought to **Grafana’s** Dashboard. The dashboard may not initially show up and the screen will remain blank. If this happens just hit refresh on your browser and it should pop right up.

[](https://tutumcloud.files.wordpress.com/2014/08/grafana.png)

Let’s test to see if **Grafana** is collecting data. Click on the name of the graph, it should beGraphite test, and from the dropdown select Edit.

Click on the Series field and select stats, click on (value) and choose which data you’d like to view. Soon enough your graph will become populated with the data **cAdvisor** is collecting.

[](https://tutumcloud.files.wordpress.com/2014/08/grafana_graph.png)

I hope you found this helpful! If you have any questions make sure to check out the [readme](https://github.com/KickingTheTV/panamax-contest-templates/blob/master/cadvisor_kickingthetv.pmx) file, and feel free to ask me any questions here!

# How to setup Docker Monitoring

https://www.brianchristner.io/how-to-setup-docker-monitoring/

How To Setup Docker Monitoring

Posted on May 20, 2015 in [**docker**](https://www.brianchristner.io/tag/docker/), [**monitoring**](https://www.brianchristner.io/tag/monitoring/)



Docker monitoring of servers and containers is becoming necessary the more Docker hosts and containers we provision. This tutorial will walk you through how to glue together several different components in order to achieve Docker monitoring.

Components For Docker Monitoring

First things first. We assume that Docker is installed, configured, and running on your host before we begin. Please ensure you can connect to your Docker host with a Web Browser either locally or over a Public IP. The rest of the Tutorial we will refer to this as the DockerIP The below components will be used to create our Docker Monitoring solution.

[**cAdvisor**](https://registry.hub.docker.com/u/google/cadvisor/) - Google has been using containers for quite sometime and created cAdvisor to help monitor their infrastructure. This single tool alone is an amazing monitoring tool. It not only monitors your Docker containers but the Docker host as well without any configuratio by just running the cAdvisor container on your Docker host. Be sure to check out the [**cAdvisor GitHub**](https://github.com/google/cadvisor) for more documentation on the API and different configuration options.

[**InfluxDB**](https://registry.hub.docker.com/u/tutum/influxdb/) - InfluxDB is a distributed time series database. cAdvisor only displays realtime information and doesn't store the metrics. We need to store the monitoring information which cAdvisor provides in order to display a time range other than realtime.

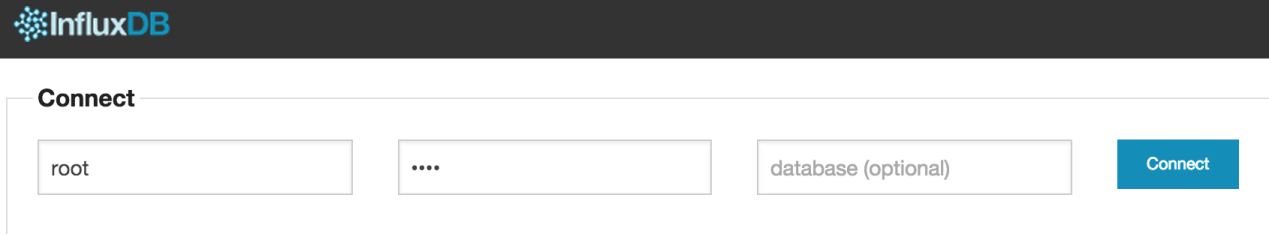
[**Grafana Metrics Dashboard**](https://registry.hub.docker.com/u/grafana/grafana/) - The Grafana Dashboard allows us to pull all the pieces together visually. This powerful Dashboard allows us to run queries against the InfluxDB and chart them accordingly in a very nice layout.

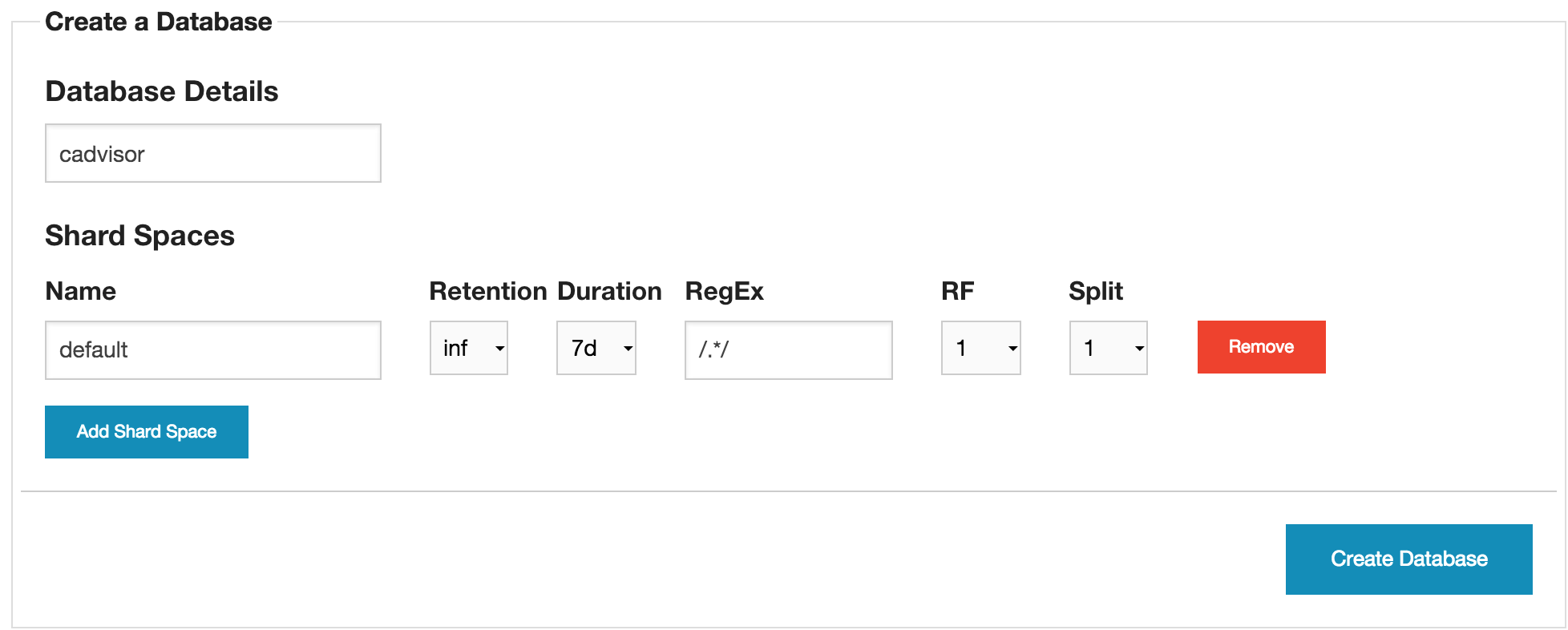
Installation Of Docker Monitoring

Now that we have an overview of the different components involved in our Docker Monitoring setup let's get started pulling it all together.

We will start with the InfluxDB first and work our way towards connecting the cAdvisor.

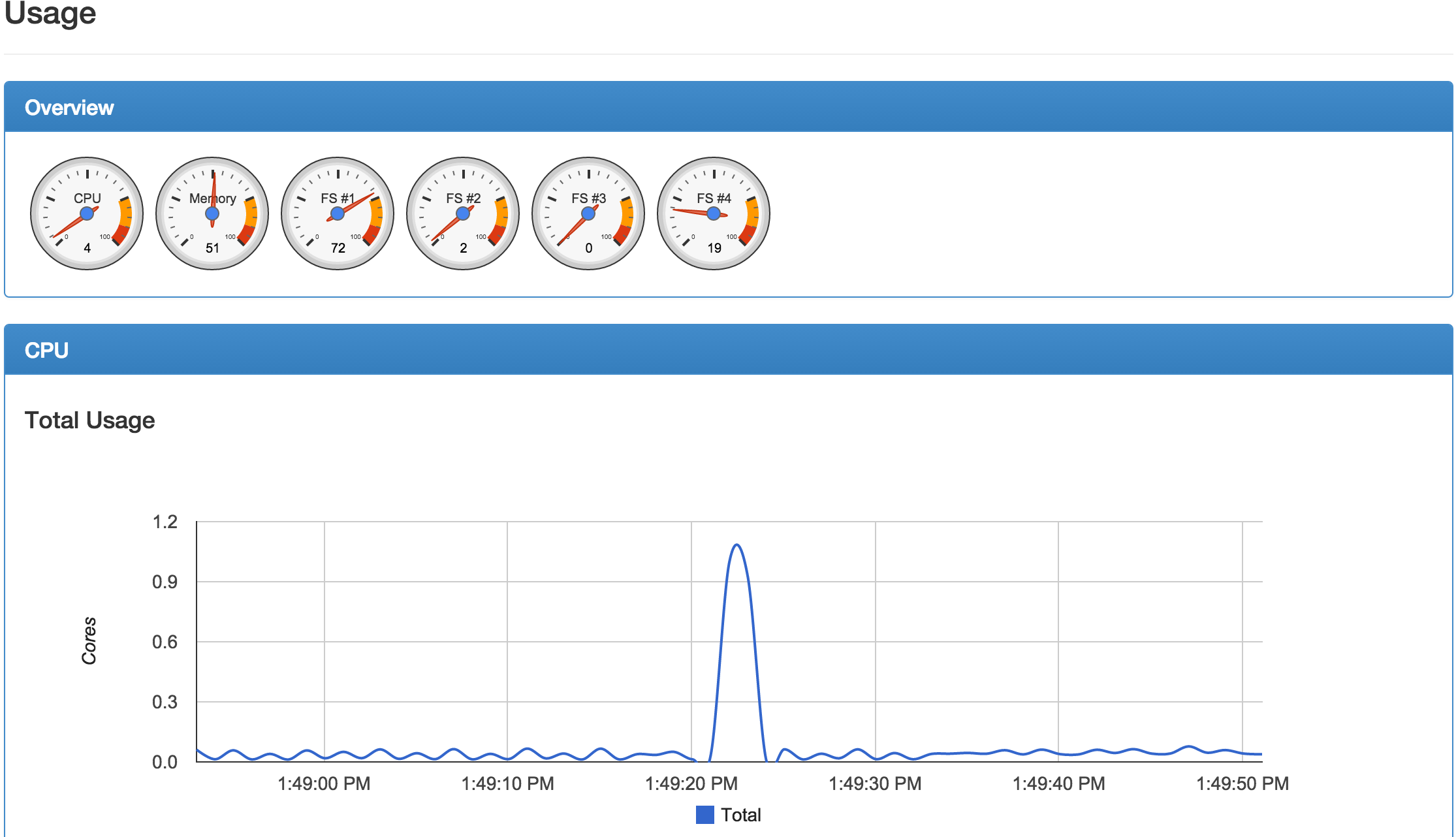
1) Install the InfluxDb. We use the default settings below and name the container influxsrv which we will use later on for linking.   
sudo docker run -d -p 8083:8083 -p 8086:8086 --expose 8090 --expose 8099 --name influxsrv tutum/influxdb

Let's test quickly that our InfluxDB installed correctly. Navigate to your [**http://DockerIP:8083**](http://dockerip:8083/) Use the credentials below to login to InfluxDB.   
Username - root   
Password - root   


2) Create the cadvisor Database   
After logging into InfluxDB click on the Databases link at the top of the screen. Type the name cadvisor for the Database name and click Create Database   


3) Install the cAdvisor container and link it to the InfluxDB container.

sudo docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro --publish=8080:8080 --detach=true --link influxsrv:influxsrv --name=cadvisor google/cadvisor:latest -storage\_driver\_db=influxdb -storage\_driver\_host=influxsrv:8086

Once the cAdvisor container has been installed and running you can now navigate to the [**http://DockerIP:8080**](http://dockerip:8080/) For example,http://192.168.10.1:8080 You should now see the cAdvisor gathering statistics on your Docker host and containers.   


4) Install the Grafana Dashboard and link it to the InfluxDB container:   
sudo docker run -d -p 3000:3000 -e INFLUXDB\_HOST=localhost -e INFLUXDB\_PORT=8086 -e INFLUXDB\_NAME=cadvisor -e INFLUXDB\_USER=root -e INFLUXDB\_PASS=root --link influxsrv:influxsrv --name grafana grafana/grafana

5) Login to Grafana and configure the Data Sources.   
Navigate to [**http://DockerIP:3000**](http://dockerip:3000/)   
Username - admin   
Password - admin

6) Connect the InfluxDB to the Grafana Dashboard:   
Once logged in click on the Grafana icon(Fireball) in the upper left hand corner of the GUI. This should pop out a sidebar menu. Click on Data Sources.

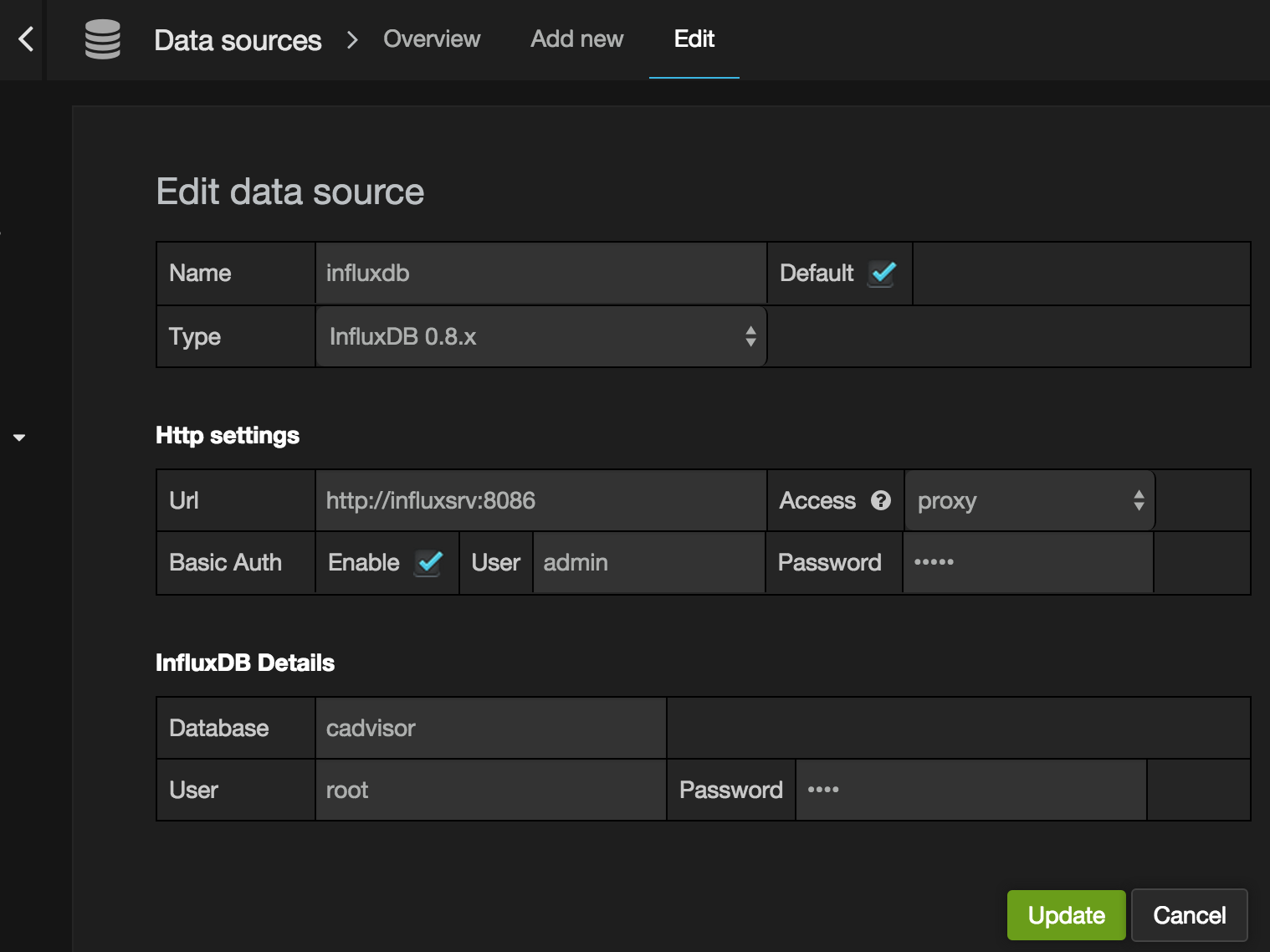
Next, click on Add New Data Source at the top of the screen.   
Fill in the following information in the Data Source screen:

**Data Source Settings**

Name: influxdb   
Type: InfluxDB 0.8.x   
Be sure to check default box.

Http settings   
Url: [**http://influxsrv:8086**](http://influxsrv:8086/) (This is the name we specified when createing the link on the Grafana container)   
Access: proxy   
Basic Auth: Enabled   
User: admin   
Password: admin

InfluxDB Details   
Database: cadvisor (Or the name you specified when creating the database in step 2)   
User: root   
Password: root

You should now have an established connection to the InfluxDB which we will test in the next section.   


Configuring Grafana For Docker Monitoring

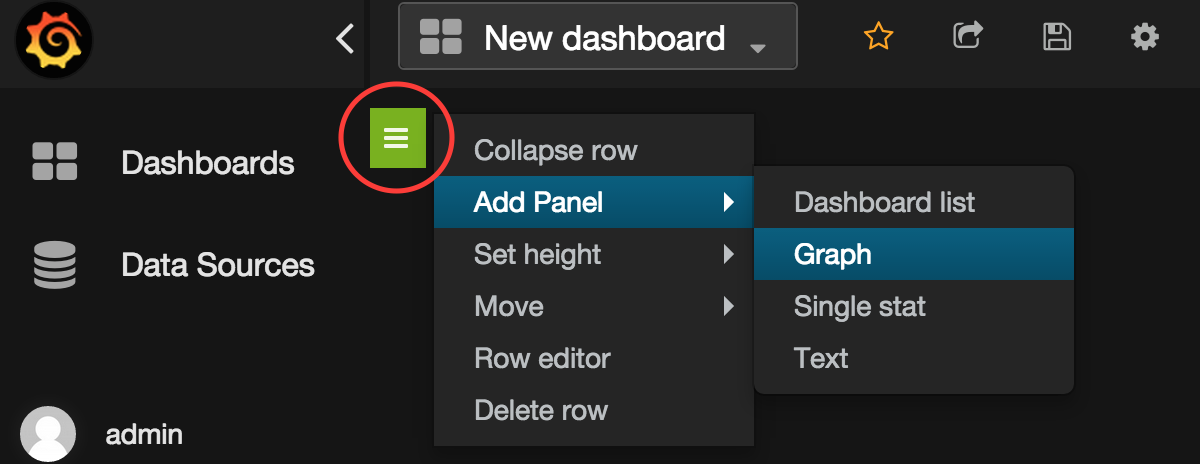
Now comes the fun part. Let's setup our first Dashboard with Grafana and visualize the data coming from the cAdvisor.

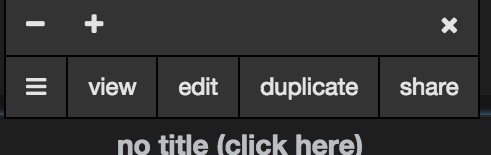
1) Click on the Grafana icon once again (The Fireball icon upper left corner)

2) Open the Dashboard menu --> Expand the Home Menu drop Down --> Click +New

3) We've now created a new Dashboard inside of Grafana. Let's create our first graph inside this Dashboard. Click the green vertical line as seen below in the screenshot circled in Red. This expands the row options for the Dashboard.

4) Click Add Panel --> Graph



5) Click the Title area of the new Graph you created where it says "no title (click here)" and click Edit   


6) It's time to write our first query for our graph. We will create a graph displaying the Filesystem storage limit and usage.   
**Query 1** - Fill in the following information inside the Graph screen: series: stats   
Click on "value" which will present you a drop down list of available series available inside of the InfluxDB.   
select: mean(fs\_limit)   
Alias: Limit

**Query 2** - At the Bottom of Graph screen is an +Add Query button which allows us to add another metric to our graph. series: stats   
select: mean(fs\_usage)   
Alias: Usage



7) Click on the General Menu and Change the Title of your Graph

8) Click on the Axis & Grid Menu   
Left Y Unit: Bytes   
Your Chart should now display with the correct units.

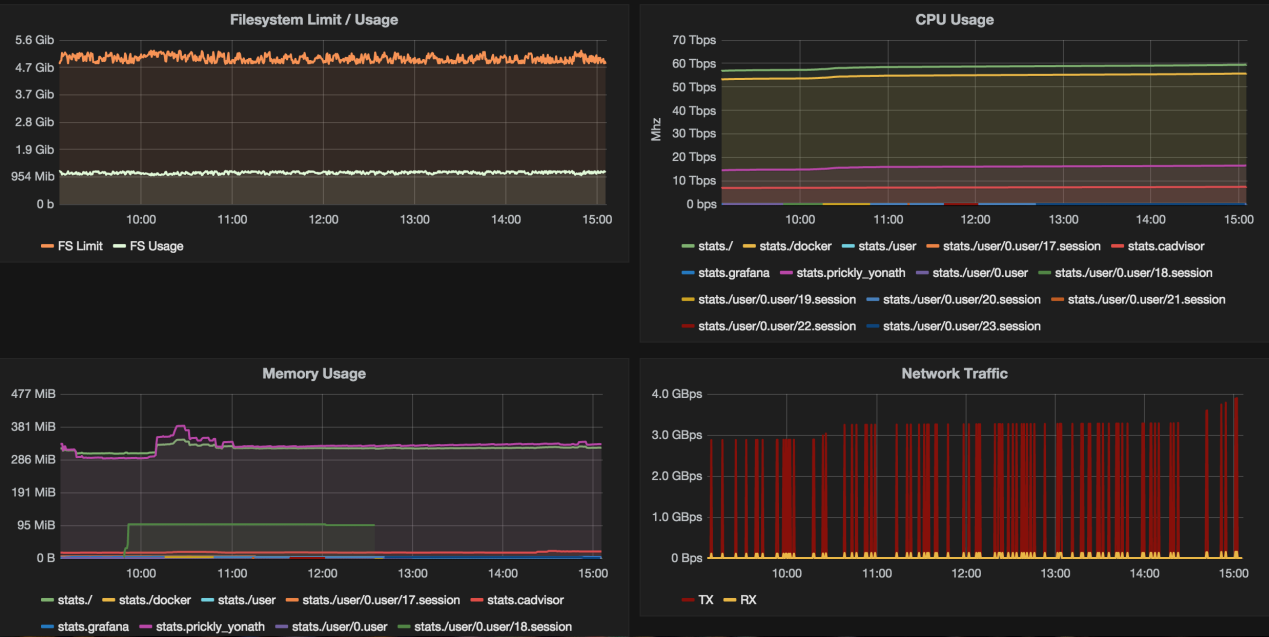
9) Once finished like any other project be sure to save your work. Hit the Save icon a the top of the screen.

The queries seen in the sample screenshot can be found here - [**Docker Monitoring Queries**](https://github.com/vegasbrianc/docker-monitoring/tree/master/Queries)

Docker Monitoring Conclusion

We have now built a single Grafana Dashboard with a Graph containing our Filesystem statistics. As you can see it's extremely simple to create multiple graphs to monitor our Docker Host and Containers.

Be sure to check out the [**Grafana Docs**](http://docs.grafana.org/) to dive deeper with the queries and functionality of Grafana. Take a look below at the screenshot which shows the possibilities for creating some really interesting graphs (Bandwidth, CPU Usage per Container, Memory Usage, and Filesystem Limit/Usage).



Leave a comment below if you have any issues or questions.

Good Luck!

Troubleshooting

In the event you have troubles this is for you. It took me quite sometime to figure out all the settings and where the problems were with the connection from Grafana to InfluxDB. In the event you have issues with your Graphs I highly recommend a Development Console in your Browser of choice.

With the development console it is really easy to see problems with your queries or connections to the InfluxDB container. For Example: Chrome Development Tools -> More Tools -> Javascript Console

Another workaround is using the IP address of the containers to resolve connection issues. However, if you restart the container the IP address changes so this is only a temp fix.

docker inspect <container name>

Search the output for the IP address which is under the Network Settings section as seen below:   
"NetworkSettings": { "Bridge": "docker0", "Gateway": "172.17.42.1", "GlobalIPv6Address": "", "GlobalIPv6PrefixLen": 0, "IPAddress": "172.17.0.54",

You can then replace the name that we used in Data Source settings we used above as a workaround. This worked for me until I fixed the links between containers.

Update

Thanks to everyone for the overwhelming response to this article. Since this article was published Dale Kate-Murray and Ross Jimenez created a[**Docker Monitoring Compose file**](http://bit.ly/1LFek7L) to help everyone get started.

Another question that came up in the comments is how to build the different dashboards. So here is the JSON file from the Dashboard -[**https://github.com/vegasbrianc/docker-monitoring**](https://github.com/vegasbrianc/docker-monitoring)

# Docker集群监控平台 cAdvisor InfluxDB Grafana

- 云计算技术频道 - 红黑联盟 http://www.2cto.com/net/201701/582103.html

基础概念

cAdvisor

? **cAdvisor 为Docker容器用户提供了了解运行时容器资源使用和性能特征的工具。cAdvisor的容器抽象基于Google的lmctfy容器栈，因此原生支持Docker容器并能够“开箱即用”地支持其他的容器类型。cAdvisor部署为一个运行中的daemon，它会收集、聚集、处理并导出运行中容器的信息。这些信息能够包含容器级别的资源隔离参数、资源的历史使用状况、反映资源使用和网络统计数据完整历史状况的柱状图。**

InfluxDB

**InfluxDB 是一个开源分布式时序、事件和指标**[**数据库**](http://www.2cto.com/database/)**。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.**

其主要特色功能  
基于时间序列，支持与时间有关的相关函数（如最大，最小，求和等） 可度量性：你可以实时对大量数据进行计算 基于事件：它支持任意的事件数据

**?InfluxDB的主要特点**

无结构（无模式）：可以是任意数量的列可拓展的 支持min, max, sum, count, mean, median 等一系列函数，方便统计 原生的HTTP支持，内置HTTP API 强大的类SQL语法 自带管理界面，方便使用

Grafana

**Graphite 是一款开源的监控绘图工具。可以实时收集、存储、显示时间序列类型的数据（time series data），有些类似Kibana的东西。**

?- **以下是官方的说明**

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

镜像列表

[root@master workdir]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

registry latest 182810e6ba8c 36 hours ago 37.6 MB

grafana/grafana 4.0.2 757396810071 2 weeks ago 268.2 MB

jevic.io/nginx alpine d964ab5d0abe 4 weeks ago 54.89 MB

google/cadvisor fejta fe153dd2defc 5 weeks ago 736.7 MB

jevic.io/cadvisor fejta fe153dd2defc 5 weeks ago 736.7 MB

jevic.io/influxdb 0.13 39fa42a093e0 5 months ago 290.3 MB

tutum/influxdb 0.13 39fa42a093e0 5 months ago 290.3 MB

提示：请勿下载使用 influxdb:latest 镜像

启动脚本：

[root@master workdir]# cat influxdb.sh

#!/bin/bash

docker service create \

--network jevic-io \

-p 8083:8083 \

-p 8086:8086 \

--mount source=influxdb-vol,type=volume,target=/var/lib/influxdb \

--name=influxdb \

--constraint 'node.hostname==node01' \

jevic.io/influxdb:0.13

[root@master workdir]# cat cadvisor.sh

#!/bin/bash

docker service create \

--network jevic-io \

--name cadvisor \

-p 8080:8080 \

--mode global \ #为每个节点创建一个服务,收集节点docker性能数据

--mount source=/var/run,type=bind,target=/var/run,readonly=false \

--mount source=/,type=bind,target=/rootfs,readonly=true \

--mount source=/sys,type=bind,target=/sys,readonly=true \

--mount source=/var/lib/docker,type=bind,target=/var/lib/docker,readonly=true \

jevic.io/cadvisor:fejta -storage\_driver=influxdb -storage\_driver\_host=influxdb:8086 -storage\_driver\_db=cadvisor

[root@master workdir]# cat grafana.sh

#!/bin/bash

docker service create \

--network jevic-io \

--name grafana \

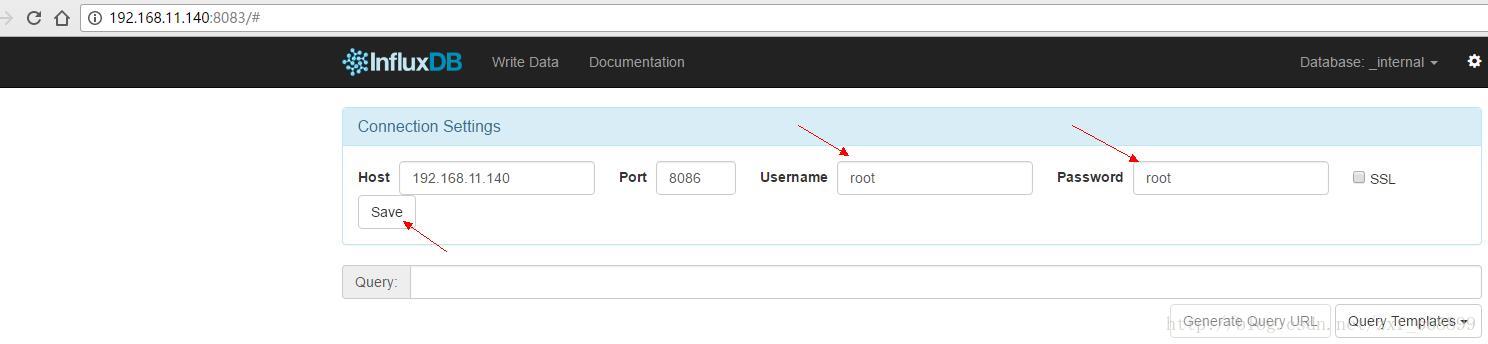
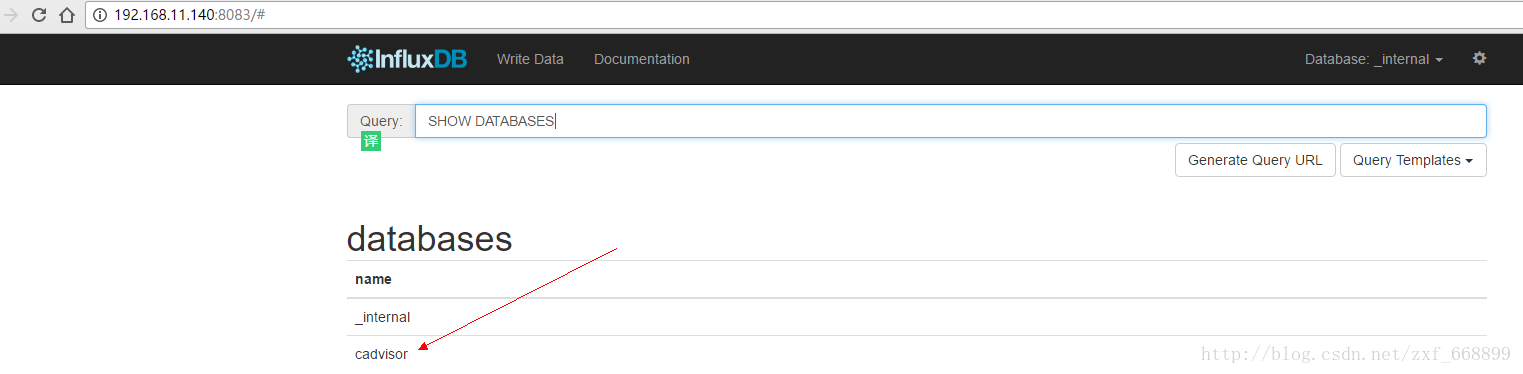
#-e "GF\_SECURITY\_ADMIN\_PASSWORD=passwd" \

--constraint 'node.hostname==master' \

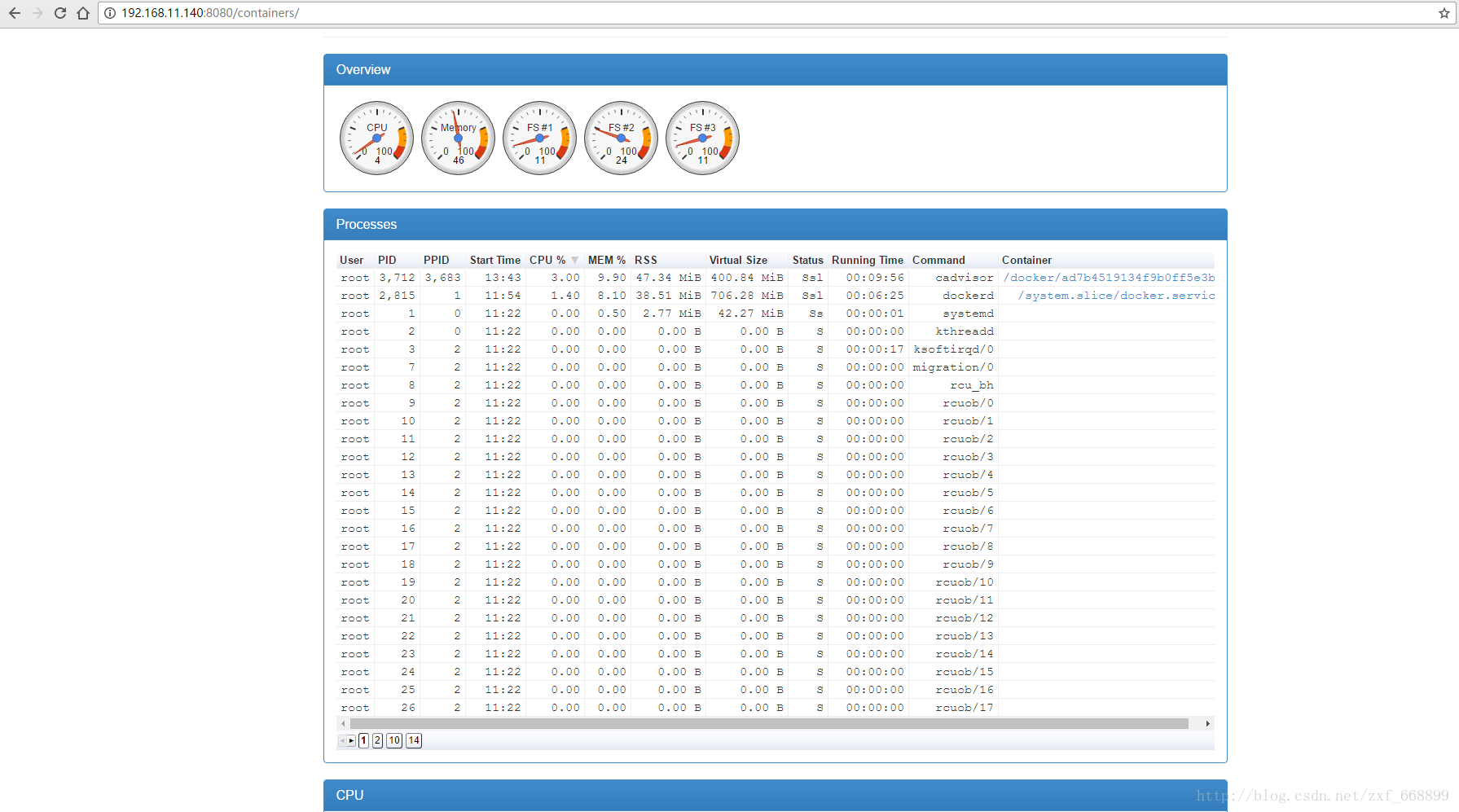
-p 3000:3000 \

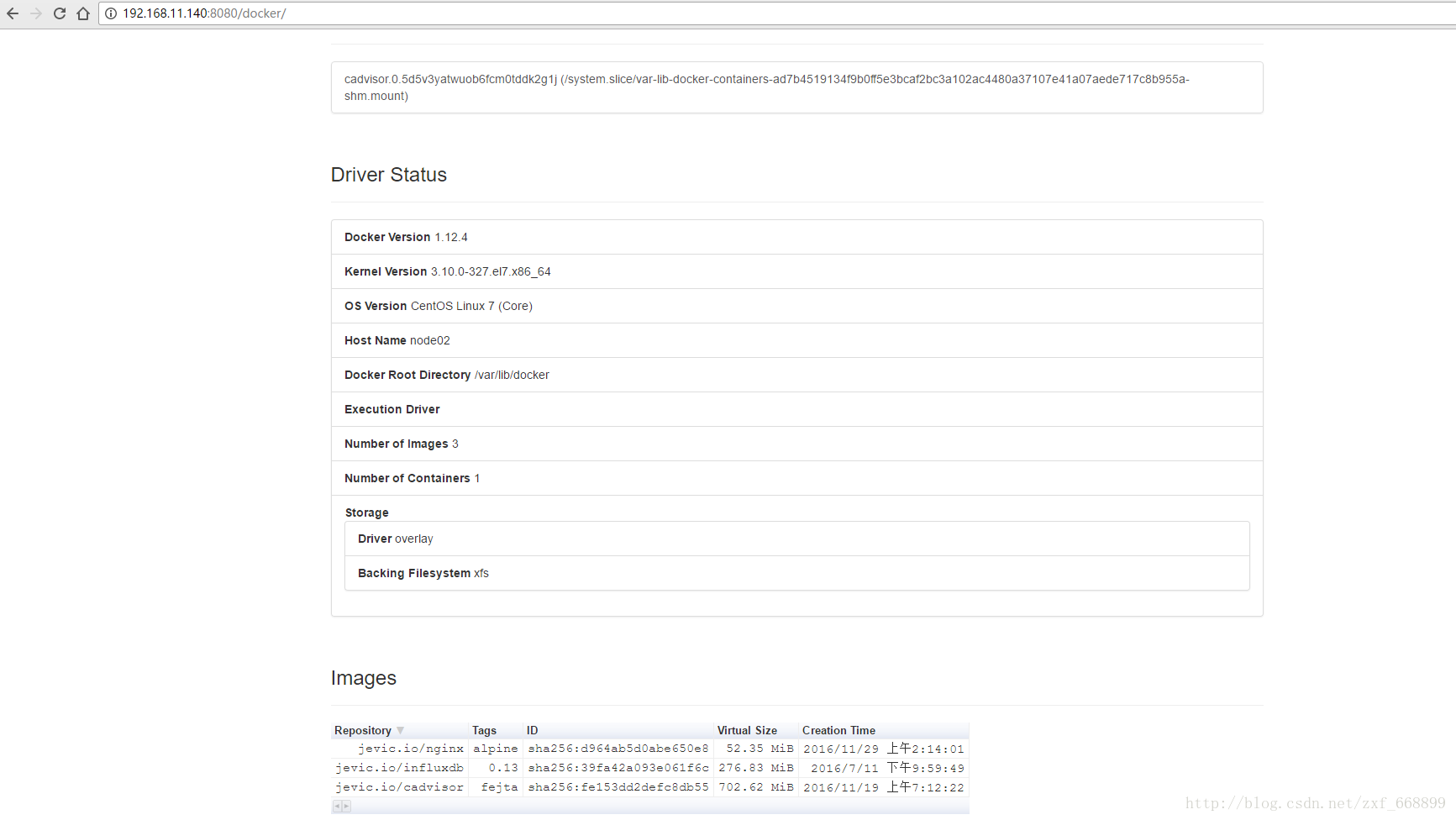
grafana/grafana:4.0.2

influxdb 设置

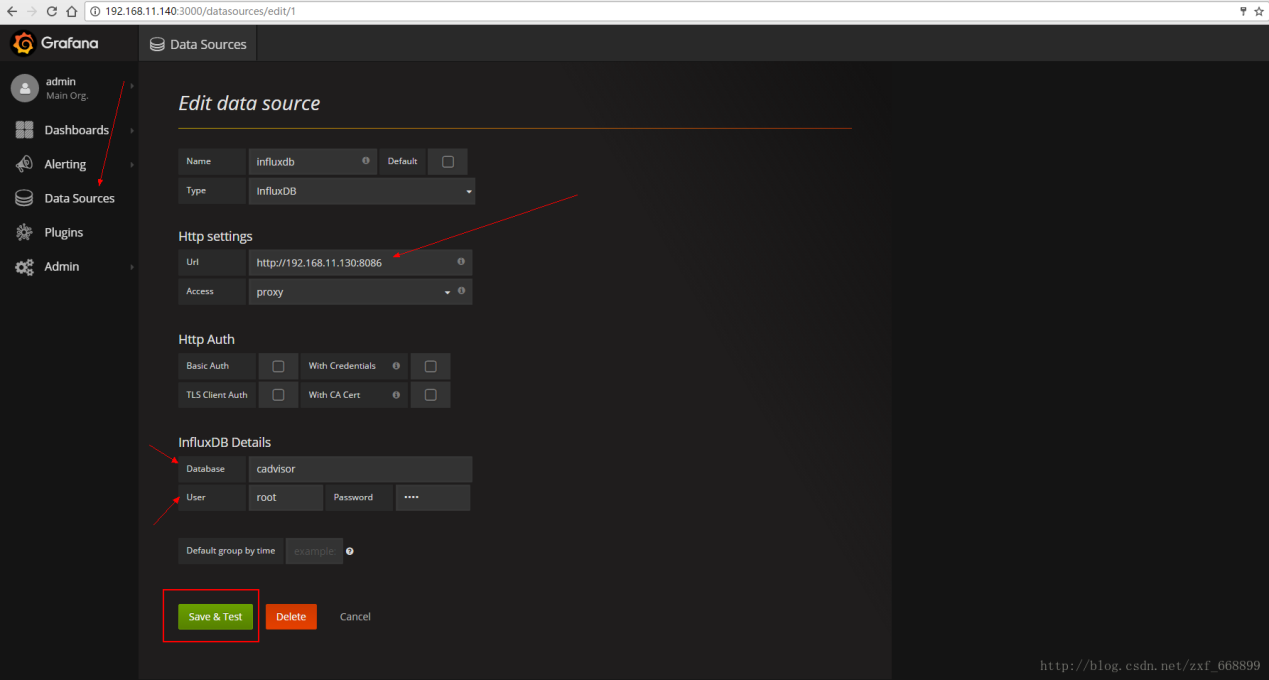
访问 8083端口 用户名密码默认都为 root  
创建数据库并查看  
CREATE DATABASE “cadvisZ喎�"http://www.2cto.com/kf/ware/vc/" target="\_blank" class="keylink">vciZyZHF1bzsgU0hPVyBEQVRBQkFTRVMNCjxwPjxpbWcgYWx0PQ=="这里写图片描述" src="http://www.2cto.com/uploadfile/Collfiles/20161230/20161230093013251.png" title="\" />  


cAdvisor

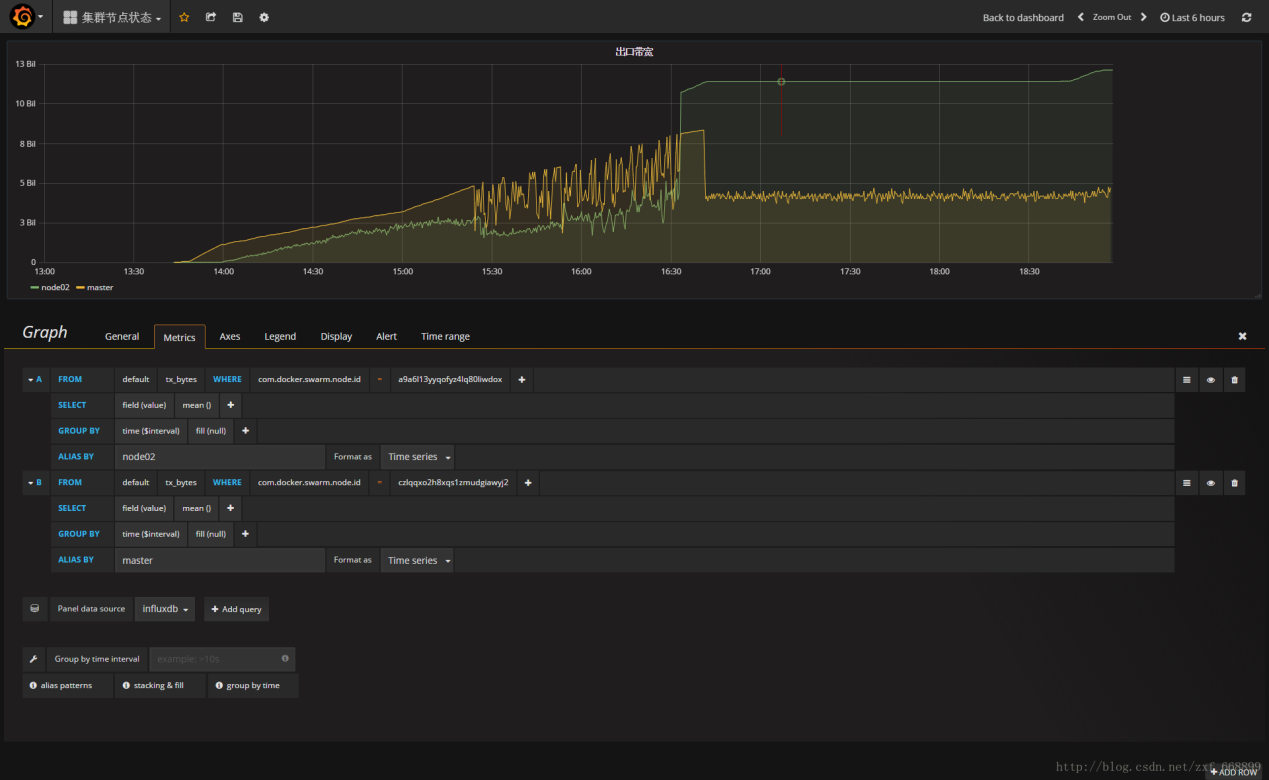
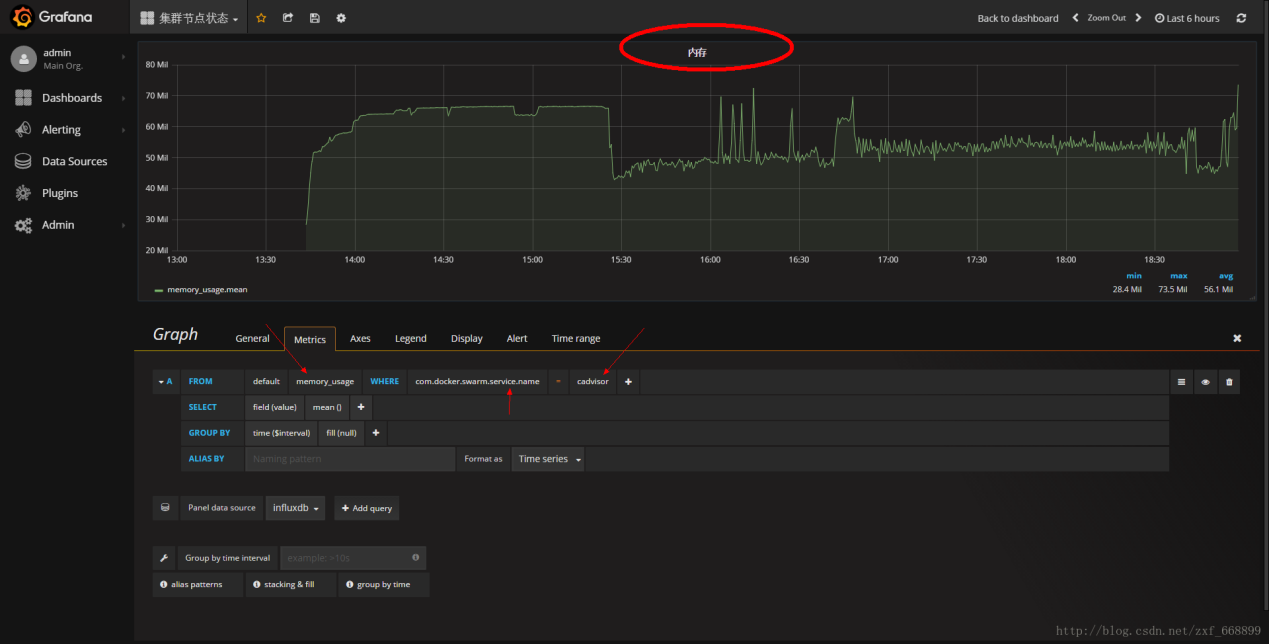
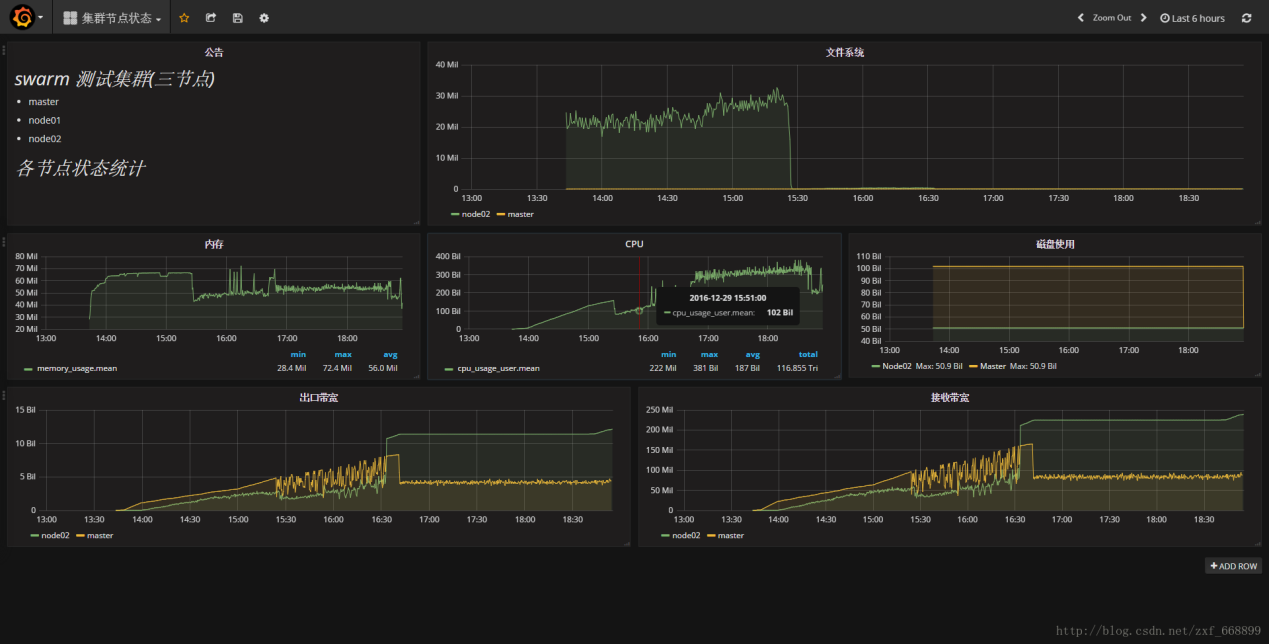
cdvisor运行以后，可以通过<http://192.168.11.140:8080/> 查看到Docker运行的机器和容器状态  


通过<http://192.168.11.140:8080/docker/>，可以看到Docker服务器的基本信息，如Host、镜像数据、窗口数据等情况  


Grafana 图形配置

运行Grfana容器，通过[浏览器](http://www.2cto.com/os/liulanqi/)打开[http://192.168.11.130:3000](http://192.168.11.130:3000/)，用户名admin，密码admin 配置数据源  


示例图【请右键打开新标签查看原图】

# 利用谷歌开源工具cAdvisor 结合influxdb存储＋Grafana前端展示进行Docker容器的监控

- 飞走不可 - 博客园 http://www.cnblogs.com/hanyifeng/p/6233851.html

一、Docker 监控方式

1.利用docker 的 docker stats API

命令： docker stats [容器ID/容器名称]

[复制代码](javascript:void(0);)

[root@docker ~]# docker stats --help

Usage: docker stats [OPTIONS] [CONTAINER...]

Display a live stream of container(s) resource usage statistics

-a, --all Show all containers (default shows just running)

--help Print usage

--no-stream Disable streaming stats and only pull the first result

[复制代码](javascript:void(0);)

参数：-a, --all 　　表示查看所有容器包括已经exit状态的

　　   --no-stream　　表示仅拉取第一次的请求结果后就结束

示例：查看下某个容器的状态

[root@docker ~]# docker stats --no-stream cadvisor

CONTAINER CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O

cadvisor 1.48% 62.48 MB / 3.977 GB 1.57% 4.902 MB / 378 MB 18.93 MB / 0 B

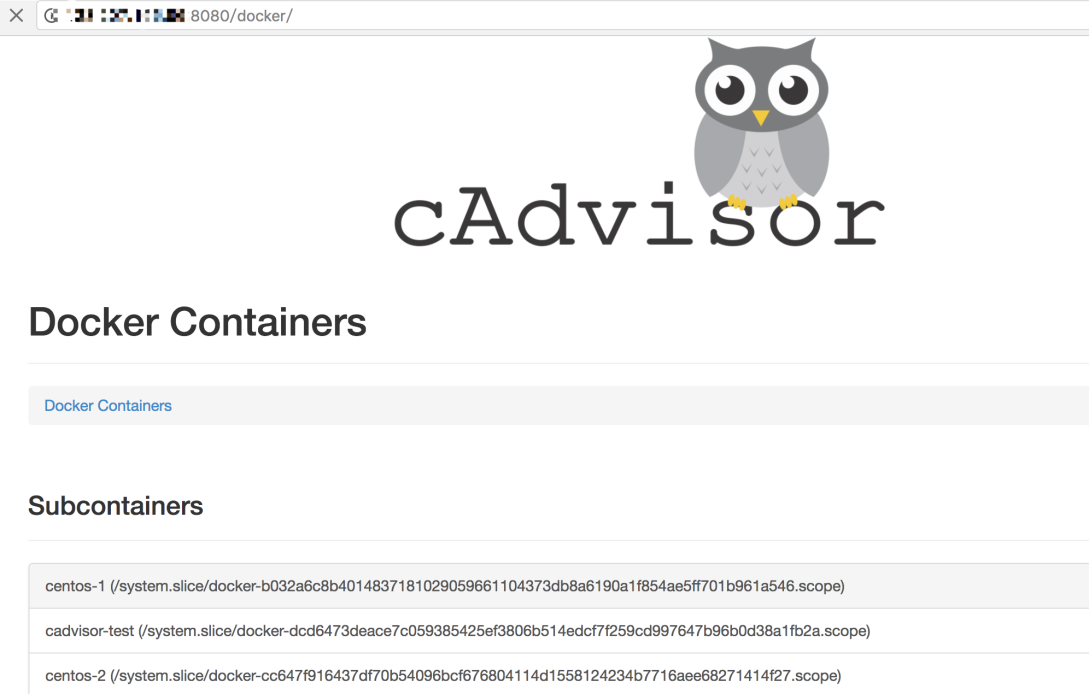
从上面可以看到容器的资源使用，包括CPU、内存、内存使用率、网络入口IO、磁盘IO、及内存LIMIT。该方式比较简单明了，就像系统自带的top命令一样。

2.利用谷歌开源工具cAdvisor（[官网](https://github.com/google/cadvisor)）

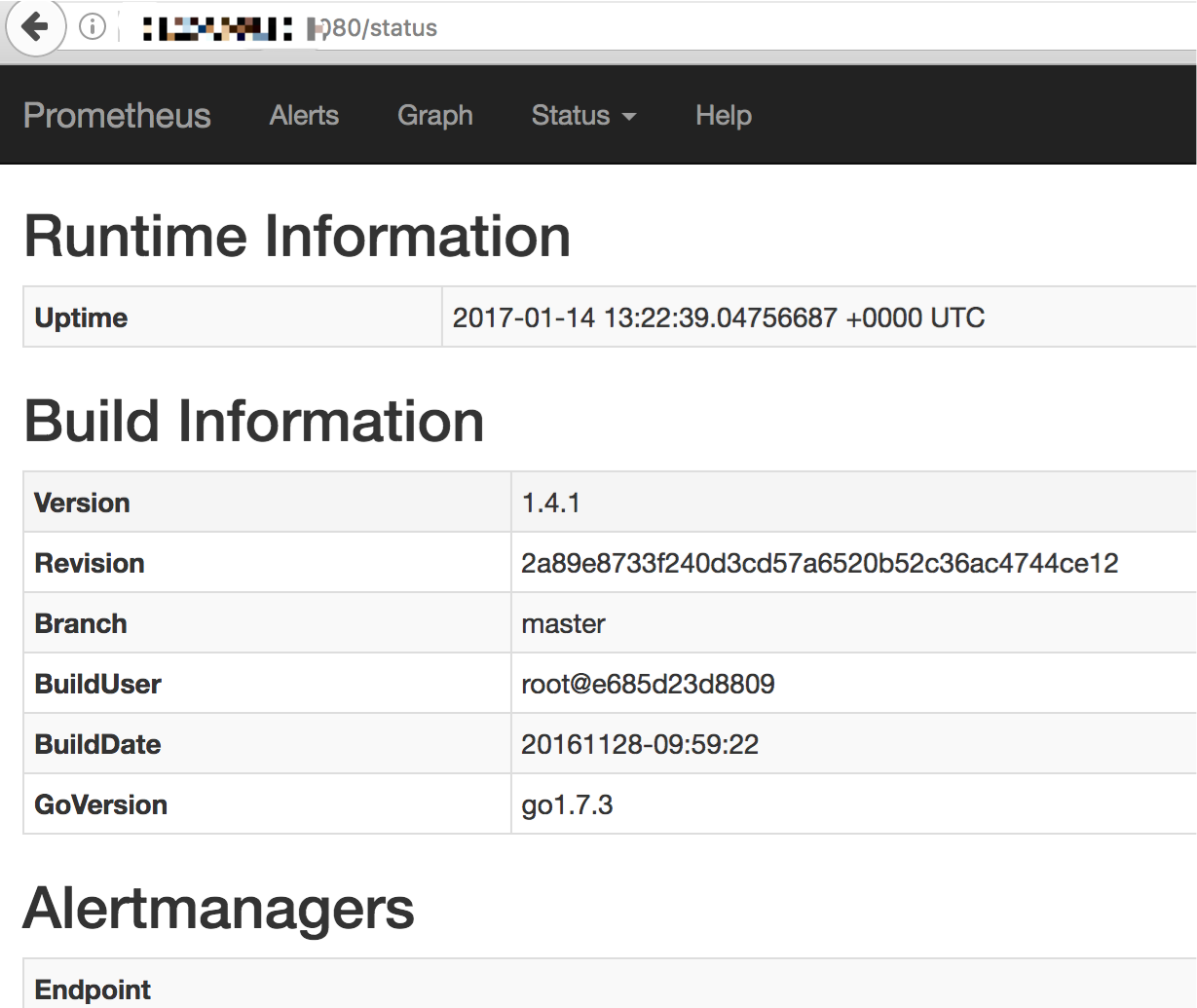
创建容器，如果docker主机上没有该容器，可能需要喝杯热茶的时间（去docker hub上下载镜像，视网速）

[root@docker ~]# docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro --publish=8080:8080 --detach=true --name=cadvisor-test google/cadvisor:latest

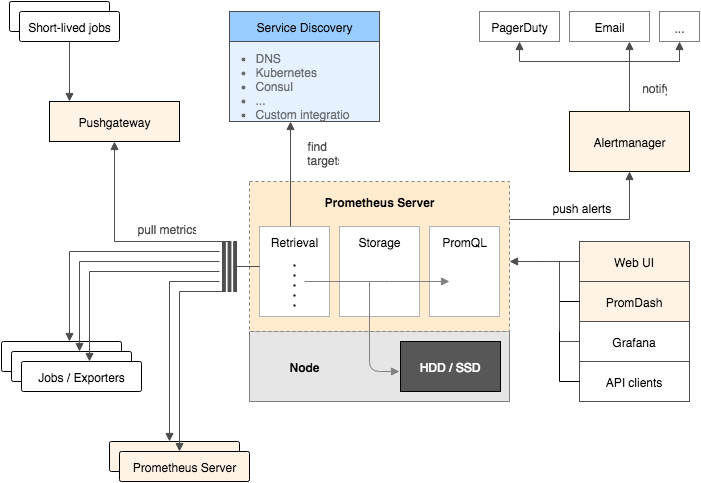
创建完成后，可以访问http://主机ip:8080/,如下图

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

3.利用独立开源项目Prometheus监控工具（[官网](https://prometheus.io/)），后续有时间会专门对其进行系统学习下。

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

来自官网资料：Prometheus及其一些生态系统组件的总体架构

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

二、本文案例开源工具介绍

1.cAdvisor (Container Advisor)该程序是由Google 开源的一个项目。提供了给使用容器的用户对其主机上运行容器的资源使用情况和性能的了解，它是一个以容器方式运行的守护进程，用来数据采集、汇聚、可视化和导出运行中容器的信息。具体来说，对于每个容器，它保留资源的隔离参数，历史资源使用，完整历史资源使用的直方图，和网络统计。此数据由机器上的容器输出。

2.InfluxDB（[官网](https://www.influxdata.com/time-series-platform/influxdb/)）是一个由Go编写的开源数据库，专门用于处理具有高可用性和高性能要求的时间序列数据。InfluxDB安装没有外部依赖，可以在几分钟内完成，然而它的灵活性、可扩展性足够复杂的部署。具有实时分析、历史分析、预测分析、异常检测等特点。

3.Grafana（[官网](http://grafana.org/)）提供了一个强大而优雅的方式来创建、探索，并可以共享仪表板和数据。也是用于查询、可视化时间序列和指标的主要工具。支持的数据源有Graphite、Prometheus、Elasticsearch、InfluxDB、OpenTSDB、AWS CloudWatch等，并且在4.0及更高版本中加入了警报功能（but现在只支持一些数据源。它们包括Graphite，Prometheus，InfluxDB和OpenTSDB）。目前报警规则只有Graph面板支持，Table和Singlestat会在未来版本中出现。支持的通知方式有Email、即时通讯工具Slack、Webhook等。

三、监控部署

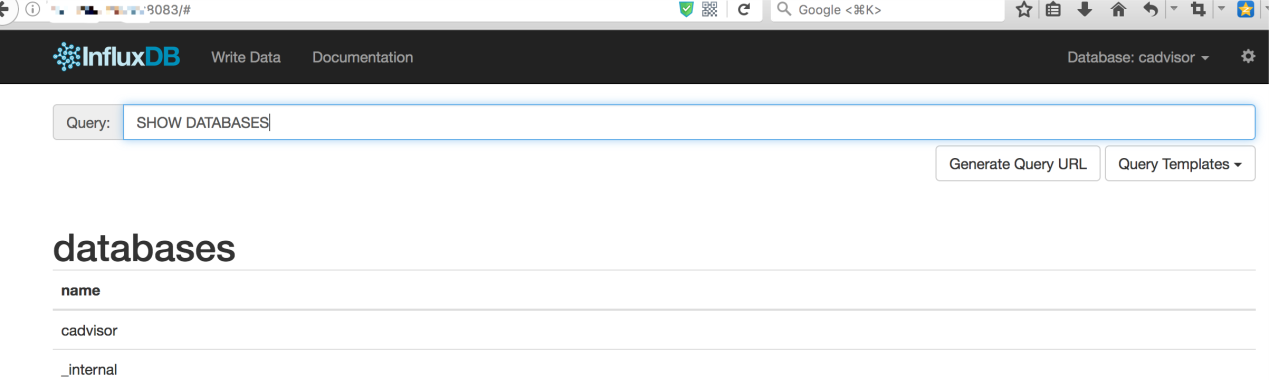
1.创建第一个容器influxdb

[root@docker ~]# docker run -d -p 8083:8083 -p 8086:8086 --expose 8090 --expose 8099 --name influxsrv -e PRE\_CREATE\_DB=cadvisor tutum/influxdb:0.13

注：默认情况下，InfluxDB使用以下网络端口：

　　TCP端口8083用于InfluxDB的管理面板  
　　TCP端口8086用于通过InfluxDB的HTTP API进行客户端 - 服务器的通信

创建完成后，打开浏览器，访问http://ip:8083，默认用户名，密码是root、root，如下图

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

2.创建cAdvisor容器并将其link到InfluxDB容器

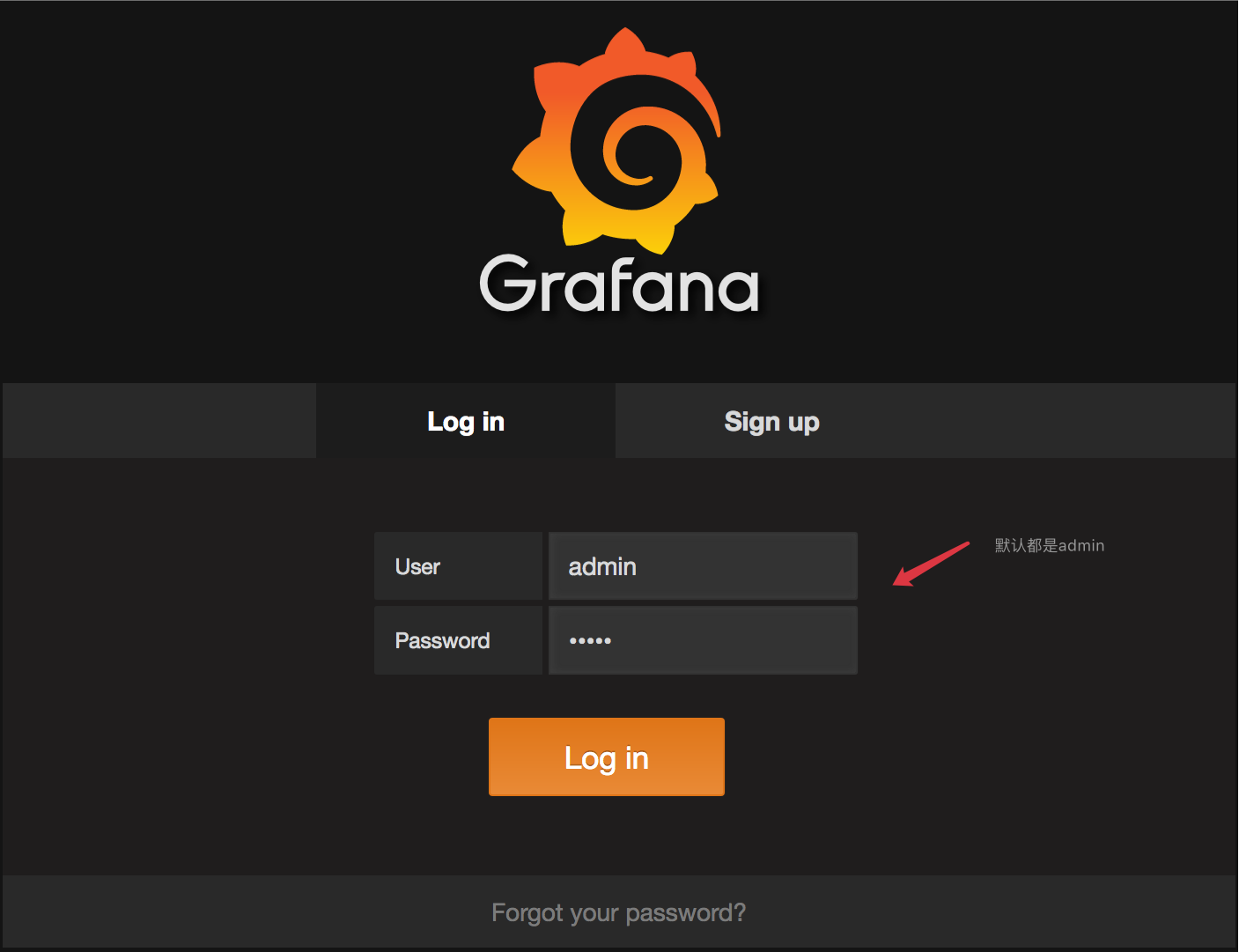
[root@docker ~]# docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro --publish=8080:8080 --detach=true --link influxsrv:influxsrv --name=cadvisor google/cadvisor:v0.24.1 -storage\_driver=influxdb -storage\_driver\_db=cadvisor -storage\_driver\_host=influxsrv:8086

3.创建Grafana容器并将其链接到InfluxDB容器：

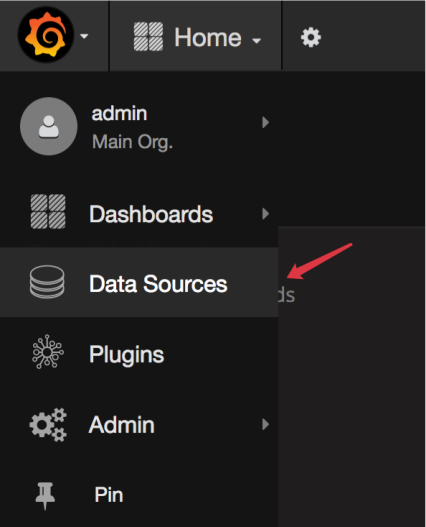
docker run -d -p 3000:3000 -e INFLUXDB\_HOST=localhost -e INFLUXDB\_PORT=8086 -e INFLUXDB\_NAME=cadvisor -e INFLUXDB\_USER=root -e INFLUXDB\_PASS=root --link influxsrv:influxsrv --name grafana grafana/grafana:3.1.1

4.打开浏览器，访问http://ip:3000/ Grafana界面

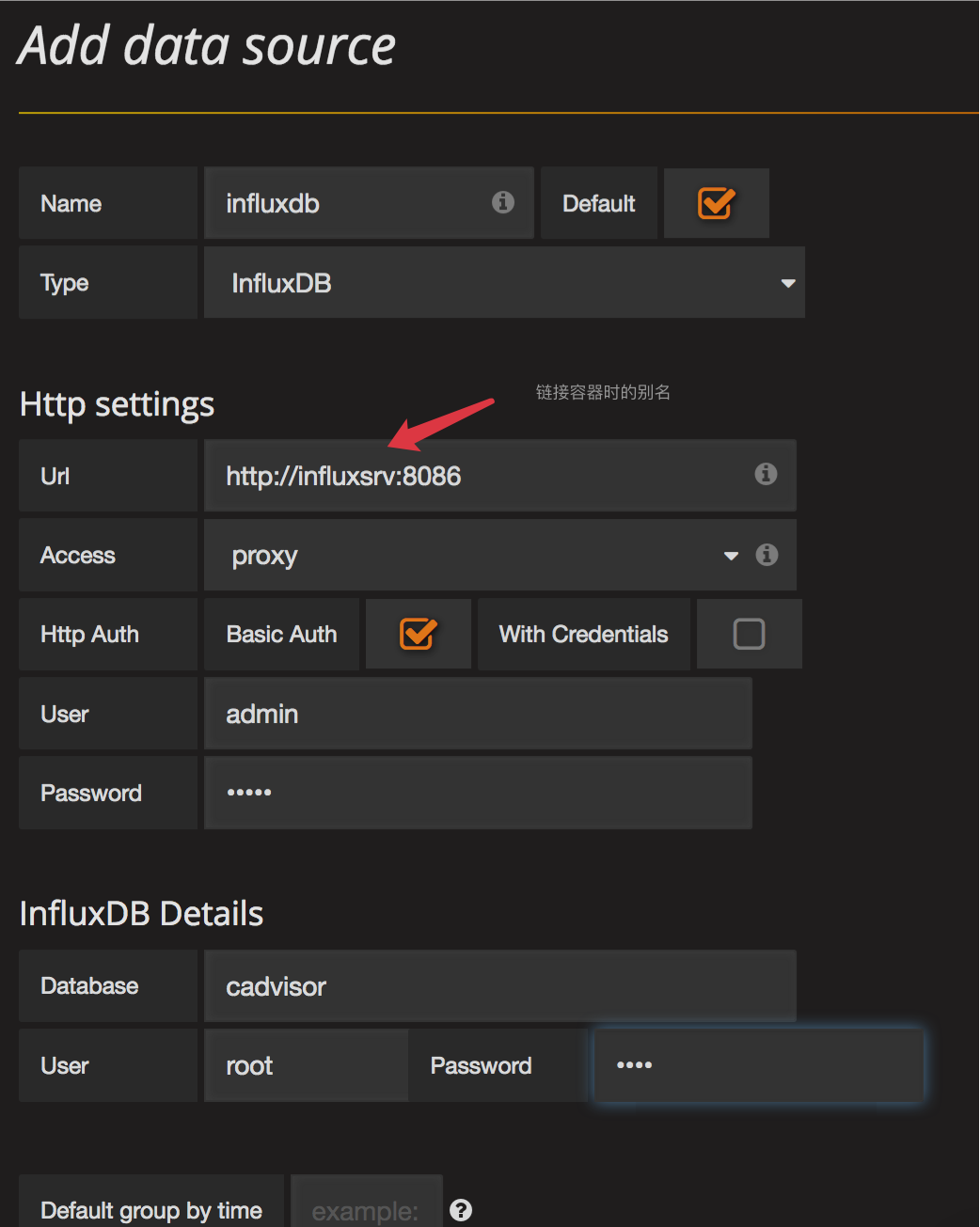
*如有转载，请注明原文出处。*[*飞走不可*](http://www.cnblogs.com/hanyifeng/p/6233851.html)*：*[*http://www.cnblogs.com/hanyifeng/p/6233851.html*](http://www.cnblogs.com/hanyifeng/p/6233851.html)

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

5.登录上去后，将Influxdb设置为Grafana的数据源，按照下图添加：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

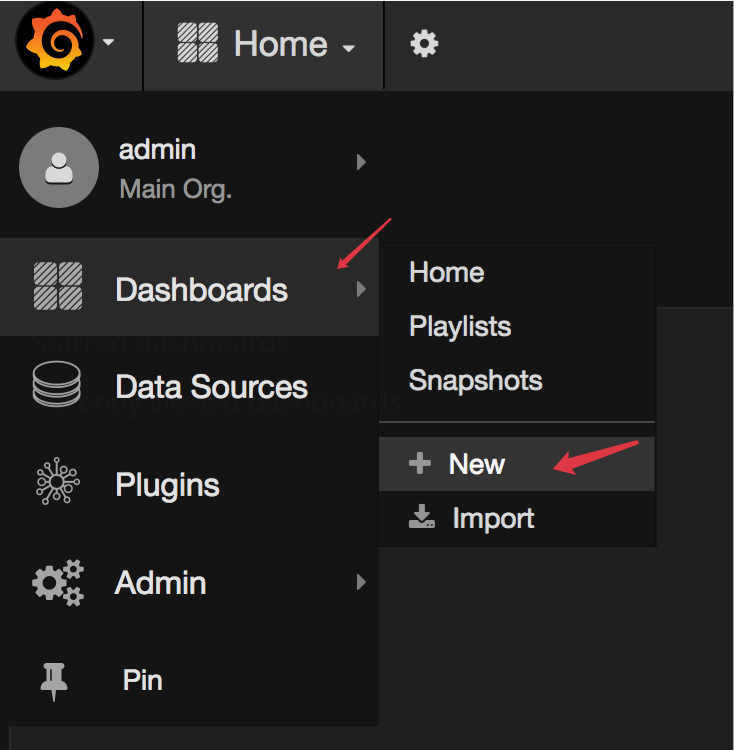
之后点击Add data source，add增加。信息如下：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

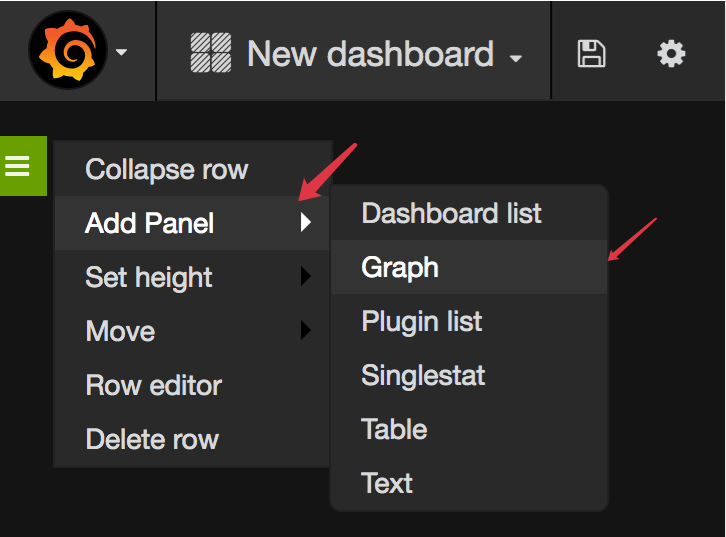
最后保存测试连接。

6.我们使用Grafana设置我们的第一个Dashboard，并可视化来自cAdvisor的数据。

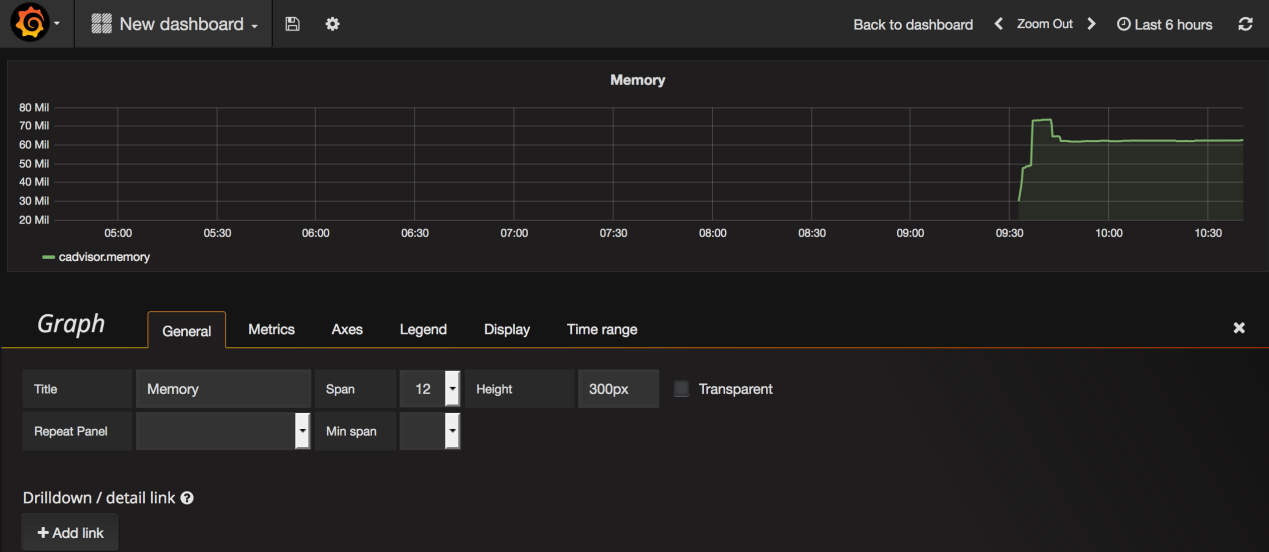
首先，添加一个Dashboard：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

然后新建一个graph，如下图：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

先设置Graph中的每个标题选项，如下：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

General：

　　Title：Memory

　　Height：300px

Metrics：输入查询语句及选择数据源，default 已经是influxdb，可以不用设置（点击http://images2015.cnblogs.com/blog/790056/201701/790056-20170117143307536-798072549.png 可切换模式 ）。

　　query 1: SELECT mean("value") FROM "memory\_usage" WHERE container\_name='cadvisor' AND container\_name='cadvisor' AND $timeFilter GROUP BY time($interval), "container\_name" fill(previous)

　　ALIAS BY：$tag\_container\_name.memory

　　Format as：Time series

Axes：主要修改一下Y轴的 显示单位

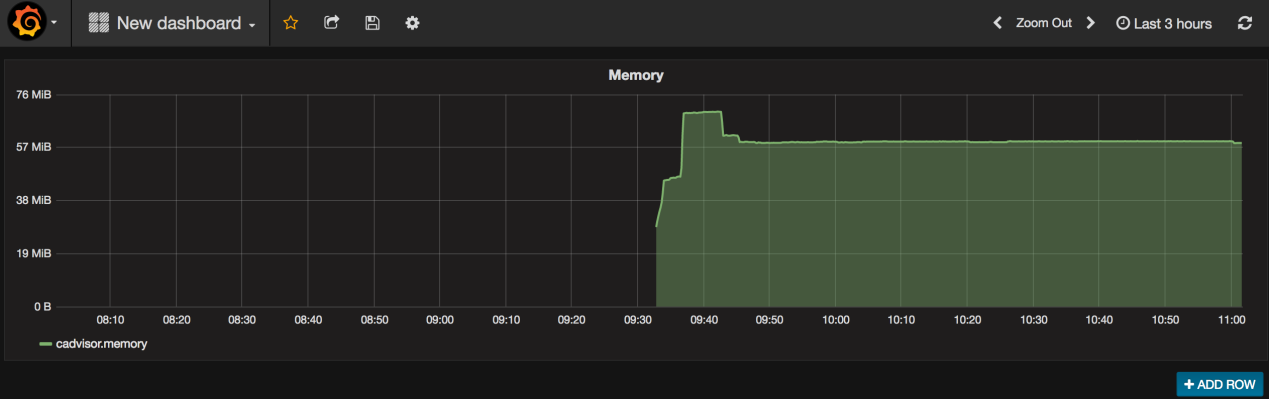
　　Left Y-->Unit：bytes、Y-Min：0

Legend：可以在图标左下方显示统计的最大、最小、及平均值等。

Display：主要修改下显示的波浪线或者点，还可以修改区域的显示深度（可以按个人喜好）

　　Mode Options-->Fill：4、LineWidth：2

修改设置完成后，记得点击正上面的保存http://images2015.cnblogs.com/blog/790056/201701/790056-20170115105712760-535502828.png按钮。完整的一个dashboard就出来了。如下图：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

点击ADD ROW ，可以增加其它监控Graph、Table、Text等。剩下需要监控的如CPU、磁盘IO、网络IO。创建方式和上面基本一致，除了左Y轴那里需要注意下单位。

7.完整的几个图，以及查询语句如下：

[](http://www.cnblogs.com/hanyifeng/p/6233851.html)

cpu 查询：

SELECT derivative(mean("value"), 10s) FROM "cpu\_usage\_total" WHERE container\_name='cadvisor' AND $timeFilter GROUP BY time($interval), container\_name fill(previous)

网络 i/o 查询：  
tx：

SELECT derivative(mean("value"), 10s)/8 FROM "tx\_bytes" WHERE container\_name='cadvisor' AND $timeFilter GROUP BY time($interval), container\_name fill(previous)

rx：

SELECT derivative(mean("value"), 10s)/8 FROM "rx\_bytes" WHERE container\_name='cadvisor' AND $timeFilter GROUP BY time($interval), container\_name fill(previous)

文件系统使用及limit：

fs

SELECT mean("value") FROM "fs\_usage" WHERE container\_name='cadvisor' AND container\_name='cadvisor' AND $timeFilter GROUP BY time($interval), "container\_name" fill(previous)

limit

SELECT mean("value") FROM "fs\_limit" WHERE container\_name='cadvisor' AND $timeFilter GROUP BY time($interval), "container\_name" fill(previous)

四、总结一下

 　　Docker 容器的监控不容忽视，毕竟里面是有业务程序在跑的，有了监控，还要有相关预警规则，及报警的处理动作，就像zabbix 一样。不知道cAdvisor监控Docker 集群有木有其它的开源技术方案，毕竟Dcoker 服务器是集群存在的。下次我找到好的技术方案，再来分享。

　　如果文中有错误，还望大家不吝赐教。

如有转载，请注明原文出处。[飞走不可](http://www.cnblogs.com/hanyifeng/p/6233851.html)：<http://www.cnblogs.com/hanyifeng/p/6233851.html>

参考资料：

https://segmentfault.com/a/1190000002527178

https://www.brianchristner.io/how-to-setup-docker-monitoring/

每个人都应是守望者，守望我们的心智，我们的理想，以防它在生活中不知不觉地坠落、被操控和被自己遗忘。。。

分类: [云计算](http://www.cnblogs.com/hanyifeng/category/758360.html)

标签: [docker monitor](http://www.cnblogs.com/hanyifeng/tag/docker%20monitor/), [docker](http://www.cnblogs.com/hanyifeng/tag/docker/), [influxdb](http://www.cnblogs.com/hanyifeng/tag/influxdb/), [grafana](http://www.cnblogs.com/hanyifeng/tag/grafana/), [cAdvisor](http://www.cnblogs.com/hanyifeng/tag/cAdvisor/), [docker 监控](http://www.cnblogs.com/hanyifeng/tag/docker%20%E7%9B%91%E6%8E%A7/), [docker cAdvisor](http://www.cnblogs.com/hanyifeng/tag/docker%20cAdvisor/), [docker 容器监控](http://www.cnblogs.com/hanyifeng/tag/docker%20%E5%AE%B9%E5%99%A8%E7%9B%91%E6%8E%A7/), [容器监控](http://www.cnblogs.com/hanyifeng/tag/%E5%AE%B9%E5%99%A8%E7%9B%91%E6%8E%A7/)