JUIN 14, 2016

Monitoring de votre serveur avec Telegraf, InfluxDB et Grafana

Guillaume LINUX / TUTORIELS, TUTORIELS 18 Comments

Utilisation du disque

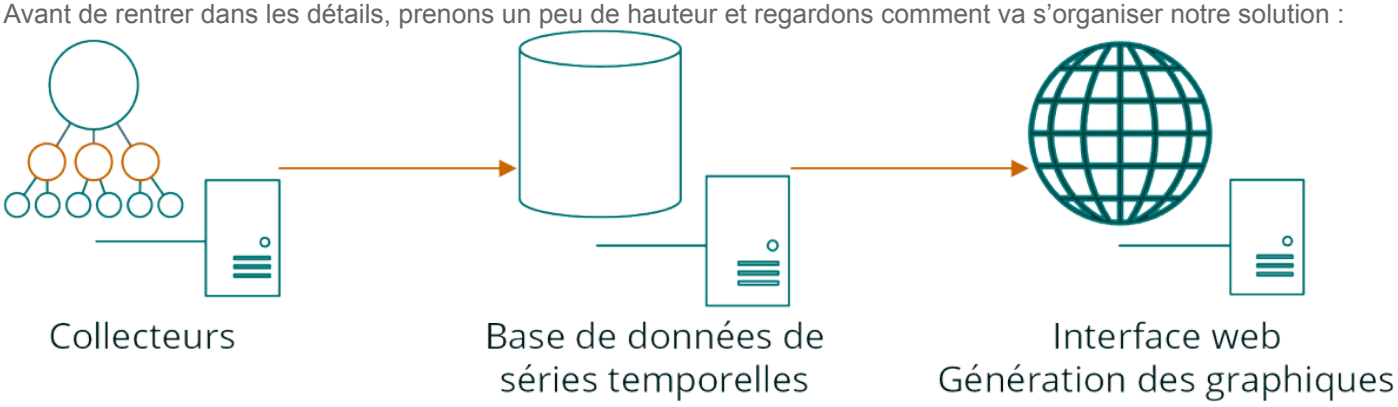
J'ai récemment essayé une nouvelle solution pour surveiller ma petite Kimsufi, et cette solution s'appuie sur une base de données de série temporelle (*Time Series Database*). Je n'a jamais abordé ce sujet sur le blog alors qu'on entend de plus en plus parler de ce type de base de données, notamment avec l'émergence récente de l'Internet des Objets (ou encore IoT, *Internet of Things*). C'est donc l'occasion de voir ce qu'il en est.

Je vous propose donc d'apprendre à surveiller votre serveur avec Telegraf, InfluxDB et Grafana.

Avant de commencer

ARCHITECTURE DE LA SOLUTION

Avant de rentrer dans les détails, prenons un peu de hauteur et regardons comment va s'organiser notre solution :



1. A gauche, on trouve les collecteurs. Leur rôle va être de relever un certains nombre de métriques sur différents éléments de notre machine à intervalles réguliers : charge CPU, consommation de mémoire… Ici on reste en administration système, mais si je reviens sur le thème *IoT*, on pourrait imaginer une sonde de température, en domotique, qui relèverait la température d'une pièce toutes les minutes par exemple. J'ai choisi d'utiliser Telegraf, nous y reviendrons
2. Toutes ces métriques vont être envoyées vers une base de données optimisée pour le stockages de séries temporelles. J'ai choisi InfluxDB, nous y reviendrons
3. Enfin, c'est bien beau d'accumuler des métriques mais si on n'en fait rien, l'utilité reste limitée. L'outil très en vogue en ce moment pour générer des graphiques ou des diagrammes à partir de bases de données de séries temporelles, c'est Grafana. Vous allez voir, c'est très joli

VOUS AVEZ DIT BASE DE DONNÉES DE SÉRIES TEMPORELLES ?

Une série temporelle est une série de points qui représente une série de mesures identifiées par l'instant auquel elles ont été prises, notamment grâce au timestamp. Le timestamp est la clé d'un point, tout comme un ID entier le serait pour une série d'articles dans la base de données d'un blog.
D'ailleurs, un ensemble de points, dans InfluxDB, ressemble à ceci :

```
name: ping_average_response_ms
------------------------------
time                    host                          url             value
1447160880026027773     ns370781.ip-91-121-193.eu     ping.ovh.net    4.346
1447160880032568398     ns370781.ip-91-121-193.eu     lafibre.info    19.286
1447160890021953406     ns370781.ip-91-121-193.eu     ping.ovh.net    4.113
1447160890035702810     ns370781.ip-91-121-193.eu     lafibre.info    19.339
1447160900018458303     ns370781.ip-91-121-193.eu     ping.ovh.net    4.1
1447160900031992843     ns370781.ip-91-121-193.eu     lafibre.info    18.911
1447160910021806894     ns370781.ip-91-121-193.eu     ping.ovh.net    4.116
1447160910033598267     ns370781.ip-91-121-193.eu     lafibre.info    19.256
1447160920019408740     ns370781.ip-91-121-193.eu     ping.ovh.net    4.147
1447160920034487812     ns370781.ip-91-121-193.eu     lafibre.info    19.115
```

La série précédente représente le temps de réponse des sites ping.ovh.net et lafibre.info lorsque ma Kimsufi leur envoi un ping. Comme je vous le disais, chaque point est unique et identifié grâce au timestamp de l'instant où la mesure a été effectuée, ici dans la colonne time.

La colonne de droite, identifiée par la clé `value` est ce qu'on appelle un champ. Les champs de valeurs sont nos données à proprement parler. Ces valeurs peuvent être de type strings, floats, entiers, ou booléens et je le répète seront toujours liés à un timestamp. Avec la valeur mesurée, ce sont les deux champs obligatoires lorsqu'on ajoute des données dans une telle base de données.

Enfin, les deux colonnes du milieu sont ce qu'on appelle des tags. Identifiés là aussi par des clés (`host` et `url`), ce sont des chaînes de caractères qui représentent les métadonnées de nos mesures. Ici j'ai une valeur pour `host` : `ns370781.ip-91-121-193.eu`, soit le nom d'hôte de ma machine et deux pour `url` : `ping.ovh.net` et `lafibre.info`, qui sont les adresses auxquelles le collecteur envoie un ping.

Enfin, vous l'aurez deviné, un point est représenté par une de ces lignes, par exemple :

```
name: ping_average_response_ms
------------------------------
time                    host                          url             value
1447160880026027773     ns370781.ip-91-121-193.eu     ping.ovh.net    4.34
```

Voilà dans les grandes lignes ce qu'il faut savoir sur ce type de base de données. Voyons maintenant comment installer InfluxDB.

Installation et configuration de InfluxDB

Ici l'installation reste relativement classique puisque InfluxData propose un dépôt Debian. Nous allons donc l'ajouter aux listes de sources APT :

```
# On fait confiance à la clé de InfluxData
$ curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -

# On ajoute le dépôt
$ echo "deb https://repos.influxdata.com/debian jessie stable" > /etc/apt/sources.list.d/influxdb.list
```

On met ensuite à jour APT et on lance l'installation :

```
$ sudo apt update
$ sudo apt install influxdb
```

Vient ensuite la configuration. J'ai gardé le fichier par défaut, j'ai simplement désactivé les sections `[monitor]` et `[admin]` parce que je n'en ai pas besoin. Après, les goûts et les couleurs…

Ensuite il va falloir préparer une base de données, dans laquelle Telegraf enverra les données qu'il collectera. Assurez-vous que le service est lancé, et connectez-vous au shell Influx avec la commande `influx`. Puis ensuite :

```
> CREATE DATABASE telegraf
> SHOW DATABASES;
name: DATABASES
---------------
name
_internal
telegraf
```

```
> CREATE USER telegraf WITH PASSWORD 'un_joli_password'
> GRANT ALL ON telegraf TO telegraf
```

Voilà, normalement tout est prêt. Personnellement j'ajoute une dernière chose : une politique de rétention de données limitée à 30 jours, parce que je n'ai pas besoin de plus.

```
> CREATE RETENTION POLICY thirty_days ON telegraf DURATION 30d REPLICATION 1 DEFAULT
> SHOW RETENTION POLICIES ON telegraf
name            duration        replicaN        DEFAULT
DEFAULT         0               1               FALSE
thirty_days     720h0m0s        1               TRUE
```

Mais dans le cas d'une sonde de température d'une maison par exemple, il pourrait être intéressant de comparer les valeurs d'une année sur l'autre… Bref. Ma base de données est prête, nous pouvons passer à l'étape suivante : installer Telegraf et commencer à envoyer des données dans InfluxDB.

Installation et configuration de Telegraf

Telegraf faisant partie du projet InfluxData, et comme nous avons déjà ajouté leur dépôts à APT, il suffit d'un simple :

```
$ sudo apt install telegraf
```

En revanche pour la configuration, je vais carrément faire une sauvegarde du fichier d'origine et repartir à zéro.

```
$ sudo mv /etc/telegraf/telegraf.conf /etc/telegraf/telegraf.conf.orig
$ sudoedit /etc/telegraf/telegraf.conf
```

Et dans `/etc/telegraf/telegraf.conf` je mets :

```
[tags]

# Configuration for telegraf agent
[agent]
  debug = false
  flush_buffer_when_full = true
  flush_interval = "15s"
  flush_jitter = "0s"
  hostname = "hostname_de_ma_machine"
  interval = "15s"
  round_interval = true
```

Là encore c'est à adapter en fonction de vos besoins. Les miens étant minimalistes, ma configuration l'est tout autant. Ici par exemple j'ai simplement configuré l'agent Telegraf. Il faut ensuite configurer un ou plusieurs « outputs », vers lesquels Telegraf enverra ses métriques, collectées depuis des « inputs ».

Notre « output » étant InfluxDB, la configuration, que je place dans `/etc/telegraf/telegraf.d/outputs.conf` :

```
[[outputs.influxdb]]
  database = "telegraf"
  precision = "s"
  urls = [ "http://127.0.0.1:8086" ]
  username = "telegraf"
  password = "un_joli_password"
```

Il faut savoir que Telegraf est aussi capable d'envoyer des données vers tout un tas d'autres outputs comme Kafka, Graphite, OpenTSDB…

En ce qui concerne les inputs, nous allons pour le moment collecter des données à propos du CPU. Ma configuration va donc être très courte, et je la place dans `/etc/telegraf/telegraf.d/inputs_system.conf` :

```
# Read metrics about CPU usage
[[inputs.cpu]]
  percpu = false
  totalcpu = true
```

Telegraf nous propose de tester cette configuration, grâce à la commande suivante :

```
sudo telegraf -test -config /etc/telegraf/telegraf.conf -config-directory /etc/telegraf/telegraf.d
* Plugin: cpu, Collection 1
> cpu,cpu=cpu-total time_guest=0,time_guest_nice=0,time_idle=2360477.02,time_iowait=26617.74,time_irq=3.62,tim
* Plugin: cpu, Collection 2
> cpu,cpu=cpu-total time_guest=0,time_guest_nice=0,time_idle=2360478.97,time_iowait=26617.74,time_irq=3.62,tim
```

Qui nous donne un bel aperçu des données que Telegraf va envoyer dans InfluxDB. Si vous le souhaitez vous pouvez filtrer les tags ou les champs que Telegraf va envoyer. Par exemple si je ne suis intéressé que par les champs de type `usage_*`, je vais pouvoir utiliser la directive suivante :

```
# Read metrics about CPU usage
[[inputs.cpu]]
  percpu = false
  totalcpu = true
  fieldpass = [ "usage*" ]
```

À vous de fouiller dans les inputs Telegraf pour trouver votre bonheur. En ce qui concerne ma Kimsufi, voici ce que je relève en ce qui concerne le système :

```
# Read metrics about CPU usage
[[inputs.cpu]]
  percpu = false
  totalcpu = true
  fieldpass = [ "usage*" ]

# Read metrics about disk usagee
[[inputs.disk]]
  fielddrop = [ "inodes*" ]
  mount_points=["/","/home"]

# Read metrics about diskio usage
[[inputs.diskio]]
  devices = ["sda2","sda3"]
  skip_serial_number = true

# Read metrics about network usage
[[inputs.net]]
  interfaces = [ "eth0" ]
  fielddrop = [ "icmp*", "ip*", "tcp*", "udp*" ]

# Read metrics about memory usage
[[inputs.mem]]
  # no configuration

# Read metrics about swap memory usage
[[inputs.swap]]
  # no configuration

# Read metrics about system load & uptime
[[inputs.system]]
  # no configuration
```

C'est vraiment personnel, mais ça peut donner quelques pistes… Une fois que c'est prêt, redémarrez ou démarrez le service telegraf. Au bout de 15 secondes, les données devraient commencer à apparaître dans InfluxDB.

```
> USE telegraf
USING DATABASE telegraf
> SHOW MEASUREMENTS
name: measurements
------------------
name
cpu
disk
diskio
mem
net
swap
system
> SELECT TIME,host,usage_guest,usage_idle FROM cpu LIMIT 5
name: cpu
---------
TIME                    host                            usage_guest     usage_idle
1457987760000000000     ns370781.ip-91-121-193.eu       0               96.5863453814776
```

```
1457987775000000000     ns370781.ip-91-121-193.eu       0               95.00670241287492
1457987790000000000     ns370781.ip-91-121-193.eu       0               98.36202573962869
1457987805000000000     ns370781.ip-91-121-193.eu       0               98.76150627654624
1457987820000000000     ns370781.ip-91-121-193.eu       0               97.75393898785175
```

Et voilà le travail ! Les données commencent à arriver dans InfluxDB ! Il ne reste plus qu'à les exploiter et les mettre en forme. Et là, c'est le rôle de Grafana.

Grafana : installation et configuration

Ici encore pour l'installation je ne vais pas me prendre la tête, je vais utiliser le dépôt proposé par l'équipe du projet.

```
$ echo 'deb https://packagecloud.io/grafana/stable/debian/ jessie main' > /etc/apt/sources.list.d/grafana.list
$ curl https://packagecloud.io/gpg.key | sudo apt-key add -
$ sudo apt update; sudo apt install grafana
```

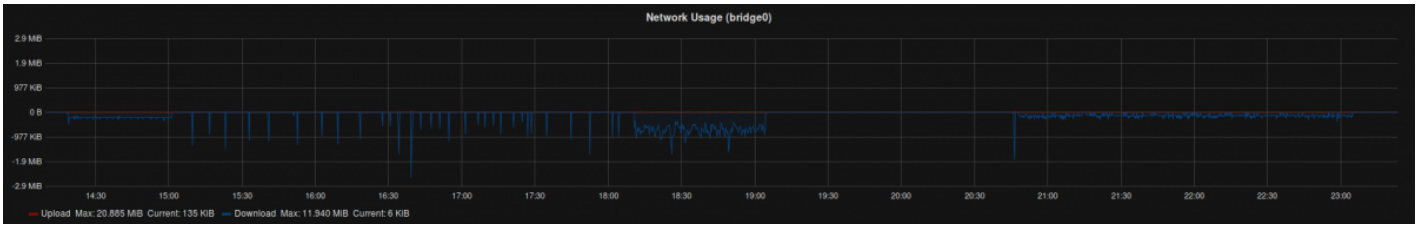
> Il est possible qu'APT ait besoin du paquet `apt-transport-https` pour contacter le dépôt de Grafana via HTTPS !

J'ai ensuite fait le choix de connecter Grafana à une base MySQL pour le stockage de ses données (utilisateurs, préférences…). Configuration classique via `/etc/grafana/grafana.ini`

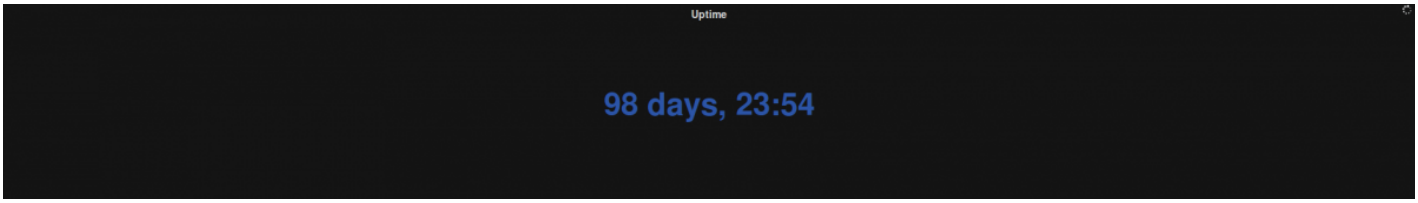
À noter aussi que j'ai fait le choix de mettre mon Grafana derrière un proxy-pass Nginx afin d'avoir de belles URLs. Côté configuration c'est censé fonctionner « out-of-the-box », sinon c'est comme d'habitude dans `/etc/grafana/grafana.ini` qu'il faut aller fouiller.

Reste à configurer nos tableaux de bords et nos graphiques, et alors quoi de mieux qu'une vidéo pour se faire une idée ?!

Voilà j'espère que ça vous aura plu, amusez-vous bien avec Grafana. Pour finir voici quelques graphiques générés avec les données d'une machine perso (cliquez pour agrandir) :

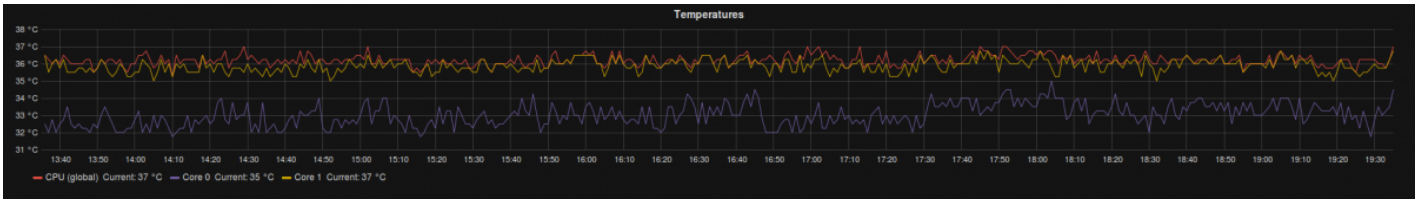


Consommation de la bande passante



Uptime, ici avec le panel Singlestat

Utilisation du disque



Température du CPU

Annexes

PROXY-PASS NGINX POUR GRAFANA

Cette configuration rend Grafana accessible sur l'adresse `grafana.mon-domaine.com` :

```
server {
    listen 80;
    server_name grafana.mon-domaine.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header Authorization "";
    }

    access_log off;
    error_log /var/log/nginx/monitorix-error.log;
}
```

Seboss666 nous propose une mise en œuvre très intéressante du triplet Telegraf, InfluxDB et Grafana pour surveiller l'état de sa connexion ADSL.

Cet article vous a plu ? Partagez-le sur les réseaux sociaux !