Monitoring d’application web avec Librato et statsd

# Introduction

Librato est un service de monitoring de performances.

A la différence de services hébergés comme cacti, munin, graphite, il affiche tout type de mesure que vous lui envoyez. Donc pas seulement des mesures hardwares, mais aussi fonctionnelles.  
Les mesures sont de type : counter (visites d’un site par ex.) et gauge (mesure à un moment précis).  
  
Son principal concurrent est New Relic.

On peut vouloir surveiller plusieurs indicateurs, afin de détecter les tendances en temps réel.  
De la santé de plusieurs serveurs (cpu, ram), aux métriques fonctionnelles.  
Cela peut être le cas du nombre de parties jouées d’un jeu en lignes, ou, dans le cas qui va nous servir d’exemple, du suivi du nombre de commandes passées par rapport au nombre de visiteur.  
Afin de détecter tout problème applicatif le plus rapidement possible.  
C’est de l’analyse de tendance.

# Librato

Présentation de l’interface.

Metric : un nom, une valeur : une sonde  
Instrument : un graphique représentant une ou plusieurs métriques à mettre en corrélation.  
Dashboard : Regroupement d’un ou plusieurs instruments afin de pouvoir être visualisés en un coup d’œil.

L’exemple va porter sur la création d’une mesure portant sur cpu\_temp. C’est une gauge puisqu’elle indiquera la valeur à un moment donnée.

http://labs.local/librato/site/post/cpu\_temp/65.4  
Retrouver la metric  
Créer un instrument  
Ajouter au dashboard

Voir le dashboard  
Explication sur le fonctionnement (curl)  
Appel des fonctions d’envoi (plusieurs fois, en changeant la valeur)

Retour sur la vue du dashboard

AB et vue du dashboard en même temps :

Configurer home, cart et order pour etre de type gauge, sum, server-side toutes les 10sec.

$ ab –n 3000 <http://labs.local/librato/site/home>

$ ab –n 2000 <http://labs.local/librato/site/cart>

$ ab –n 1000 <http://labs.local/librato/site/order>

Astuce :

Pour suivre les tendances, il également possible de placer un marqueur, par exemple lorsqu’une mise en prod à eu lieu.

<http://labs.local/librato/site/annotation/home>

Problématique : si on fait une requête en Curl à chaque visite, le site ne sera pas scalable.  
Pour pallier à cela, on utilise un collecteur de statistiques.

# Statsd

L’installation de statsd se fait très facilement en suivant les indications données par l’auteur.

Après avoir installé statsd , on installe un backend statsd pour librato.

A Mediastay, sur une application nodejs, on utilisait statsd.  
Le principe : statsd est un démon en nodejs qui écoute un port UDP.

On crée un socket UDP en javascript, ou en php, et on envoie les informations à statds.  
Pour les besoins de l’exemple, j’ai trouvé une bibliothèque PHP permettant d’envoyer des données en UDP à statsd.

Avec UDP, on accepte de perdre des paquets, mais en contrepartie et on peut traiter des milliers d’appels concurrents sans aucun ou très peu de problème de charge.

$ ab –n 2000 <http://labs.local/librato/site/home>

$ ab –n 2000 <http://labs.local/librato/site/home>d

Comparer les deux métriques sur Librato.  
Comparer la durée d’exécution des deux ab.

# Conclusion

On peut répondre à des problématiques de monitoring en temps réel grâce à ces deux outils, sans perte de performances liées à la montée en charge.

Pour en savoir plus :

<http://www.getapp.com/metrics-application>

<https://github.com/etsy/statsd/>

<https://github.com/librato/statsd-librato-backend>

<https://metrics.librato.com/metrics>