304 - Measurement

DEVOPS

WIK-DEVOPS-302

Intervenant : Jeremy Trufier <jeremy@wikodit.fr>

ynov

wikodit

WIK-DEVOPS

Programme DEVOPS

301 – Introduction, Kaizen et CAMS	1 séance
302 – La culture DevOps	1 séance
303 – Automatisation / Déploiement continu	4 séance
304 – Mesure & Monitoring	2 séance
305 – Partage	2 séance

Introduction

Pourquoi mesurer ?

- Est-ce qu'on s'est amélioré ? (Kaizen)
- Est-ce que ce qu'on déploit pose problème ?
- Quels sont les bottlenecks de nos services ?
- Comment améliorer nos process ?
- Sans donnée, rien ne peut–être prouvé ni fait

Que peut-on mesurer ?

- Marketing
- Support
- Infrastructure
- Application
- Development

Mesurer le Marketing

- Nombre d'utilisateurs (inscription, connection, durées, ...)
- Revenus générés
- Coûts d'exploitations
- Coûts des features non prévues

Mesurer le Support

- Nombre de tickets ouverts
- Nombre de tickets refusés (par raison)
- Temps de support moyen

Mesurer l'infrastructure

- Charge des serveurs
- Charge du réseau
- Temps d'interruption
- SLA et Service Availability (nombre de 9's : 99.9% = 43min/mois)
- Santé des services

Mesurer l'application

- Health Check (et informations)
- Analytics divers
- Erreurs utilisateurs

Mesurer le développement

- Temps de développement et de tests
- Fréquence de déploiement / Volume de changement
- % de travail utile / non prévu
- Vélocité des équipes (ratio entre temps réel et estimé)
- Burnchart, et toute méthode Agile de mesure

Les KPI de vélocité

- MLT (Mean Lead Time) : Temps entre le moment ou une feature est demandée et est accessible aux utilisateurs (build, test, deploiement)
- DCR (Daily Change Rate) : Nombre de changement commités et testés par jours
- MTTD (Mean Time To Detect) : Combien de temps un bug introduit reste avant d'être détecté
- MTTR (Mean Time To Recovery/Resolve) : Combien de temps entre le moment ou un bug et détecté et est corrigé
- MTTA (Mean Time To Approve) : Combien de temps entre le moment ou le développement d'une release est terminée, et elle arrive en production

Les KPI de qualité

- BFR (Build Failure Rate) : % de builds qui échouent
- DFR (Deployment Failure Rate) : % de déploiements qui échouent
- IRFR (Infrastructure-Related Failure Rate) : % de buil ou de déploiements qui échouent à cause de l'opérationnel (infrastructure)
- RWR (Rework Rate) : % de tickets qui sont réouverts
- ADR (Automated Detection Rate) : % de problèmes détectés automatiquement (unit test)
- UWR (Unplanned Work Rate) : % de problèmes non prévus qui surviennent

Instrumentalisation

Push Model vs Pull Model

- Pull Model
  - Le serveur va chercher les infos
  - Protocole SNMP (Réseaux, Routeurs, ...)
  - Le serveur doit connaitre tous les noeuds
- Push Model
  - Des agents envoie leur donnée sur un serveur qui centralise
  - Le serveur n'a pas besoin de connaitre tous les noeuds
  - Parfait pour les parcs dynamiques (Docker, Lambda, etc...)

Dashboard

- Informations en un coup d'oeil
- Accoutumance au "normal" et détection des comportement anormaux
- Mettre en avant les métriques importantes
- Utilisation couplée avec les Information Radiators
- Mais
  - Difficile d'avoir toujours un oeil sur nos dashboards
  - Compliqué à Scaler pour suivre l'infrastructure (100 métriques OK, mais 1000 métriques...)

Alertes

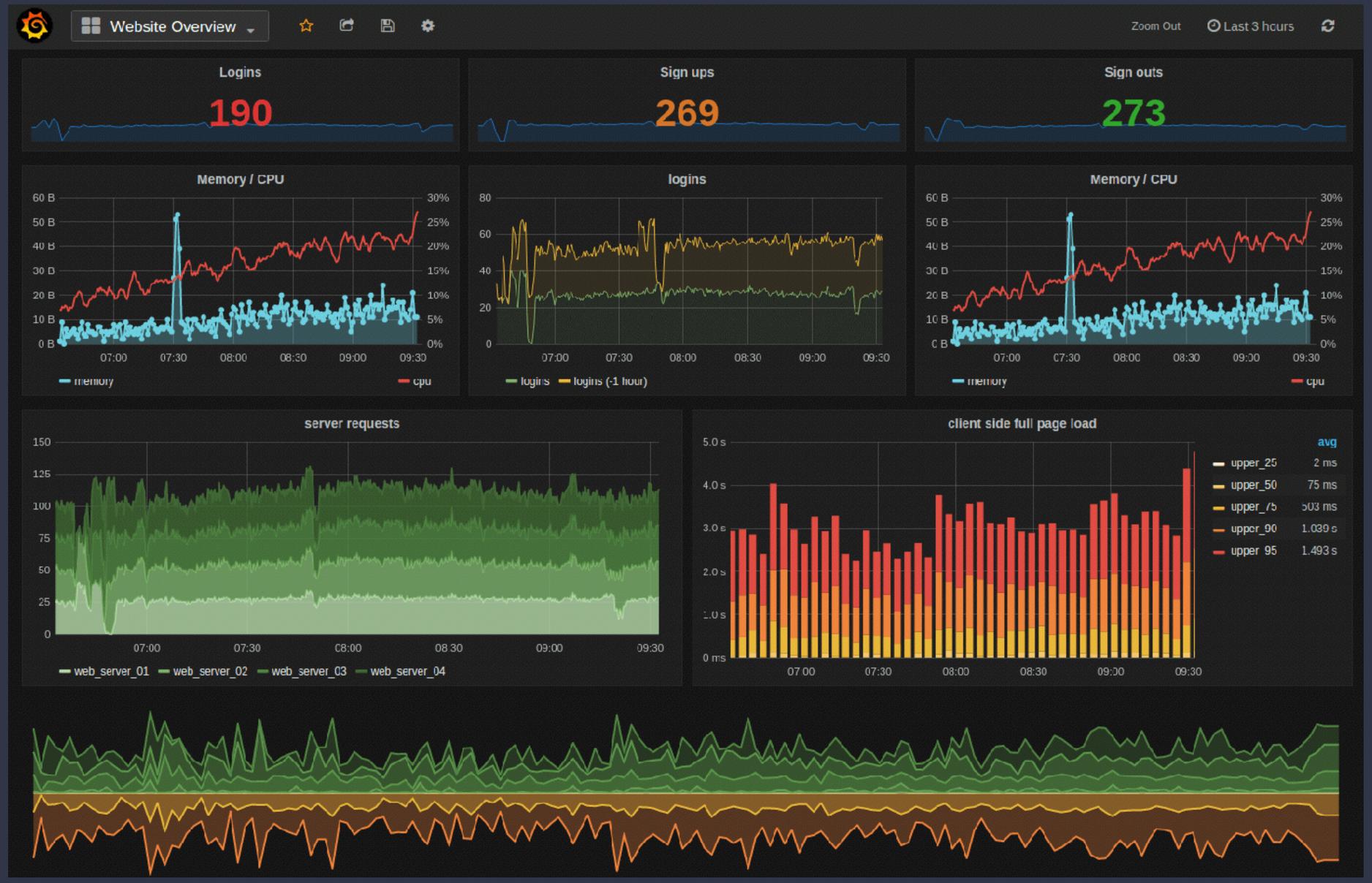
- Pro-active monitoring
- Aide à monitorer nos métriques (monitoring de monitoring)
- Déclencher des évènements selon des triggers (limites)
  - envoi de mails
  - envoi de notifications
  - résolution automatique : auto-scale, ...

Système de monitoring

Graphite

- Système de stockage de métriques
- Puissant service de centralisation
- Carbon : daemon qui reçoit des données, puissant cache
- Whisper : Base de donnée de stockage de données temporalisées
- Web-app qui permet de visualiser les métriques
- Graphite ne collecte rien

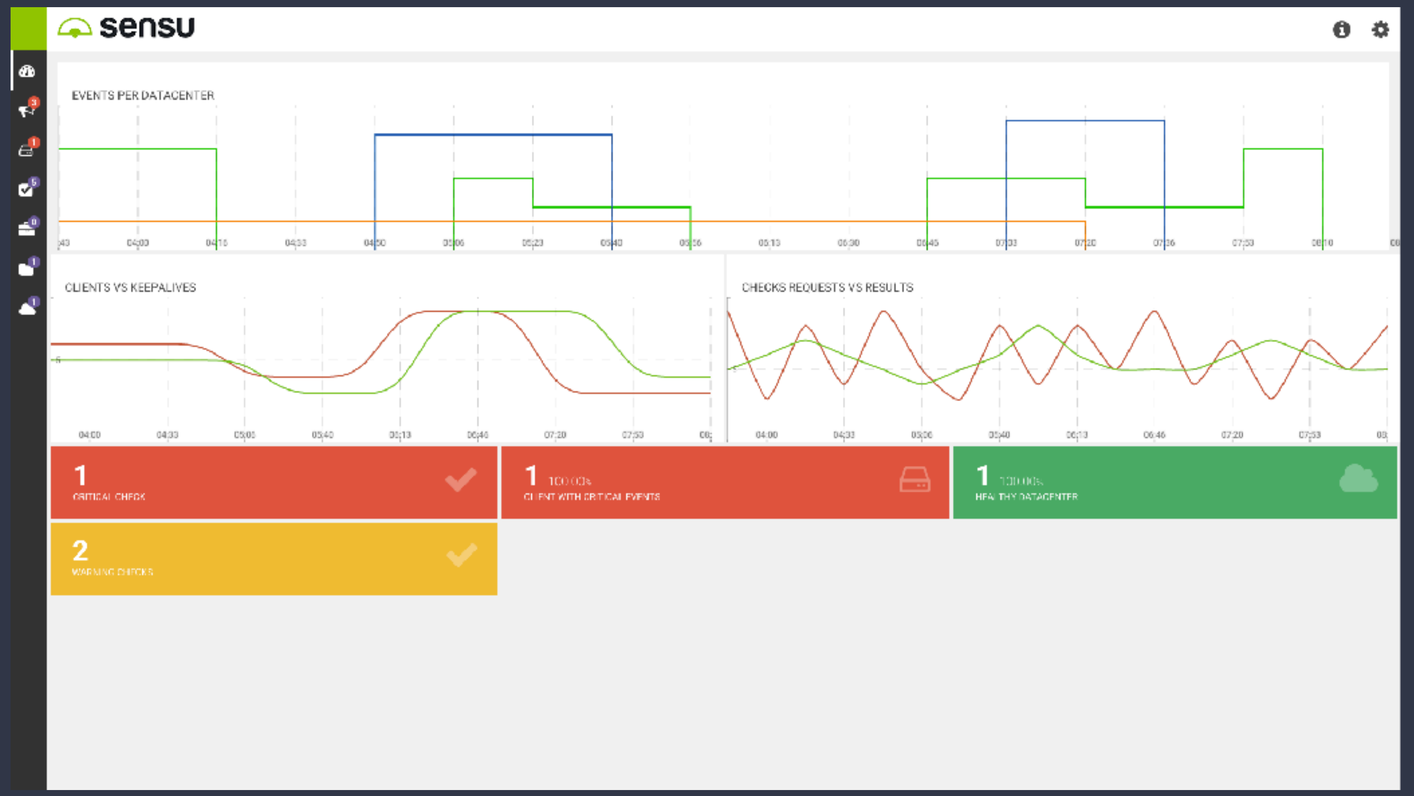
Grafana



- Dashboard pour graphite
- Très visuel (et joli)
- Très personnalisable
- Peut-être sourcé par Elastic Search

sensu

- Outil de collecte de métrics
- similaire à nagios ou cacti pour la partie de collecte
- Basé sur un serveur, un client et une API
- Grand nombre de plugins disponibles
- Version entreprise avec dashboard



statsd

- Service de collecte et d'aggregation (counter, timer, gauges)
- basé sur nodeJS
- Push model
- UDP (light)
- construit pour Graphite

collectd

- Système de collecte de statistiques
- Pull model
- Possède de nombreux plugins
- Peut écrirer les données dans Graphite

Zoom sur statsd

En gros

- Possible d'envoyer manuellement des packets UDP à statsd
- Mais possède de nombreux client sous tout language
- Notre application peut donc envoyer des métriques

Envoyez des métriques

```
echo "some_counter:1|c" | nc -u -w0 127.0.0.1 8125
echo "some_time:300|ms" | nc -u -w0 127.0.0.1 8125
echo "some_gauge:41|g" | nc -u -w0 127.0.0.1 8125
echo "some_gauge:+5|g" | nc -u -w0 127.0.0.1 8125
```

En nodeJS : Counters

npm install --save node-statsd

- Peut-être incrémenté ou décrémenté
- Représente une valeur d'évènement dans un interval de temps (flush)
- Chaque flush vide le compteur

```
const StatsD = require('node-statsd'),
      stats = new StatsD();

// On peut envoyer des métriques
let onRequest = (req, res) => {
  // ...
  stats.increment('api_call')
  // ...
}

let onRequestCanceled = (req) => {
  // ...
  stats.decrement('api_call')
  // ...
}
```

Les gauges

- Données constantes
- System load
- États actuels

```
const StatsD = require('node-statsd'),
      stats = new StatsD();

// Gestion de gauge (donnée statique)
let onClientLogin = (req, res) => {
  // ...
  server.userCount++;
  stats.gauge('active_users', server.userCount);
  // ...
}

let onClientLogout = (req, res) => {
  // ...
  server.userCount--;
  stats.gauge('active_users', server.userCount);
  // ...
}
```

timers / histogrammes

- Timer : en ms
- Histogramme : toute info à un instant T

```
const StatsD = require('node-statsd'),
      stats = new StatsD();

let onRequest = (req, res) => {
  // ...
  stats.histogram('payload_size', req.body.length)
  // ...
}

let onClientLogout = (req, res) => {
  // ...
  stats.timing('user_session_duration', user.connectedAt
- (new Date()).getTime());
  // ...
}
```

TP : Mise en place d'une solution de Monitoring

Cf. PDF correspondant

Félicitations !!

Cours WIK-DEVOPS-304 burned :)