Introduction au Monitoring

Analyser les données et alerter avant un désastre[1]

Alexandre « erdnaxe » Iooss

CR@NS

Lundi 20 janvier 2020

1. Ou « *Comment bien dormir la nuit sur ses deux oreilles ?* ».

1. Collecter et stocker les données
2. Analyser les données et alerter

Section 1

Collecter et stocker les données

Collecter les données

Comment collecter les données ?

Architecture push ou pull. On travail avec des « `exporters` » installés sur les périphériques où l'on souhaite collecter des données.
Penser à la sécurité et l'anonymisation des données.

À vous de jouer

Installez `prometheus-node-exporter` et allez sur http://localhost:9100/.
Sinon allez sur `zamok` et tapez
`curl zamok.adm.crans.org:9100/metrics`
Vous observez des « `metrics` ».

Stocker les données

L'équivalent de SQL pour des séries temporelles

Prometheus est une base de données de séries temporelles.
On le configure pour qu'il aille régulièrement interroger des « `exporters` » et sauvegarder leurs « `metrics` » dans une base de données.

À vous de jouer

Installez `prometheus` puis écrivez une configuration minimale dans `/etc/prometheus/prometheus.yml` :

```
scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets: ['127.0.0.1 :9090']
  - job_name: node
    static_configs:
      - targets: ['127.0.0.1 :9100']
```

Rechargez le service et allez sur http://localhost:9090/.

Section 2

Analyser les données et alerter

Analyser les données

C'est bien beau d'avoir les données mais maintenant faut les utiliser.

Outils classiques d'analyse

Grafana (au Cr@ns), Kibana, directement prometheus (peu avancé).

À vous de jouer

Identifiez-vous sur https://grafana.crans.org/, puis allez dans « Explore ».
Que font ces requêtes ?

- `node_cpu_frequency_hertz/1e9` ;
- `apt_upgrades_pending == 0` ;
- `sum(node_systemd_unit_state{state="failed"}) by (instance)`
- `upsInputVoltage` ;
- `unifi_vap_num_stations` ;
- `unifi_vap_num_stations{instance=~"k.*"}` ;

Alerter les administrateurs

Avec prometheus

Dans `/etc/prometheus/prometheus.yml` on peut indiquer un fichier contenant les règles d'alertes puis un webhook pour envoyer les alertes :

```
rule_files:
  - "alert.rules.yml"  # Monitoring alerts definitions

alerting:
  alertmanagers:
  - static_configs:
    - targets: ['localhost :9093']
```

Exemple de définition

Au Cr@ns : https://gitlab.crans.org/nounous/ansible/blob/master/roles/prometheus/templates/prometheus/alert.rules.yml.j2

```
groups:
- name: alert.rules
  rules:

  # Alert for any instance that is unreachable
  - alert: InstanceDown
    expr: up == 0
    for: 3m
    labels:
      severity: critical
    annotations:
      summary: "{{ $labels.instance }} est invisible !"
```

Autre exemple de définition

```
groups:
- name: alert.rules
  rules:

  # [...]

  # Check systemd unit
  - alert: SystemdServiceFailed
    expr: node_systemd_unit_state{state="failed"} == 1
    for: 10m
    labels:
      severity: warning
    annotations:
      summary: "{{ $labels.name }} a échoué"
```