Monitoring de la plateforme IOT & SmartCity

Table des matières

[1. Préambule 3](#_Toc511139242)

[2. Contexte 3](#_Toc511139243)

[3. Présentation 4](#_Toc511139244)

[a. Elasticsearch 4](#_Toc511139245)

[b. Kibana 4](#_Toc511139246)

[c. Logstash 4](#_Toc511139247)

[d. Metricbeat 5](#_Toc511139248)

[e. Filebeat 5](#_Toc511139249)

[f. Logging driver 6](#_Toc511139250)

[4. Installation 6](#_Toc511139251)

[a. Logging Driver 6](#_Toc511139252)

[b. Logstash 7](#_Toc511139253)

[c. Metricbeat 9](#_Toc511139254)

[d. Serveur web & redis 10](#_Toc511139255)

[e. Docker-compose 10](#_Toc511139256)

[f. Droit d’accès 10](#_Toc511139257)

# Préambule

Ce document présente les divers outils de monitoring pour la plateforme IOT & SmartCity de la Métropole de Nice Cote d’Azur ainsi que leurs installations et leurs configurations.

# Contexte

La plateforme IOT & SmartCity est mise en place avec Docker Swarm. Il est nécessaire de pouvoir connaître la consommation en ressource de chaque container ainsi que leurs remontées de logs.

Pour remonter les logs, on utilise Filebeat ou logstash qui permet de surveiller des fichiers comme des fichiers de logs.

Pour remonter les informations de consommation de ressource, on utilise Metricbeat qui se connecte au socket Docker.

La suite ELK (Elasticsearch, Logstash, Kibana) permet le stockage et l’affichage des informations



# Présentation

## Elasticsearch

Elasticsearch permet de stocker et d’analyser les données reçues depuis n’importe quelle entrée. Il propose un moteur de recherche intégré ainsi qu’une API REST.

## Kibana

Kibana est inclus dans la suite ELK. Il permet la consultation des données d’Elasticsearch et de les mettre en place dans des Dashboard.



## Logstash

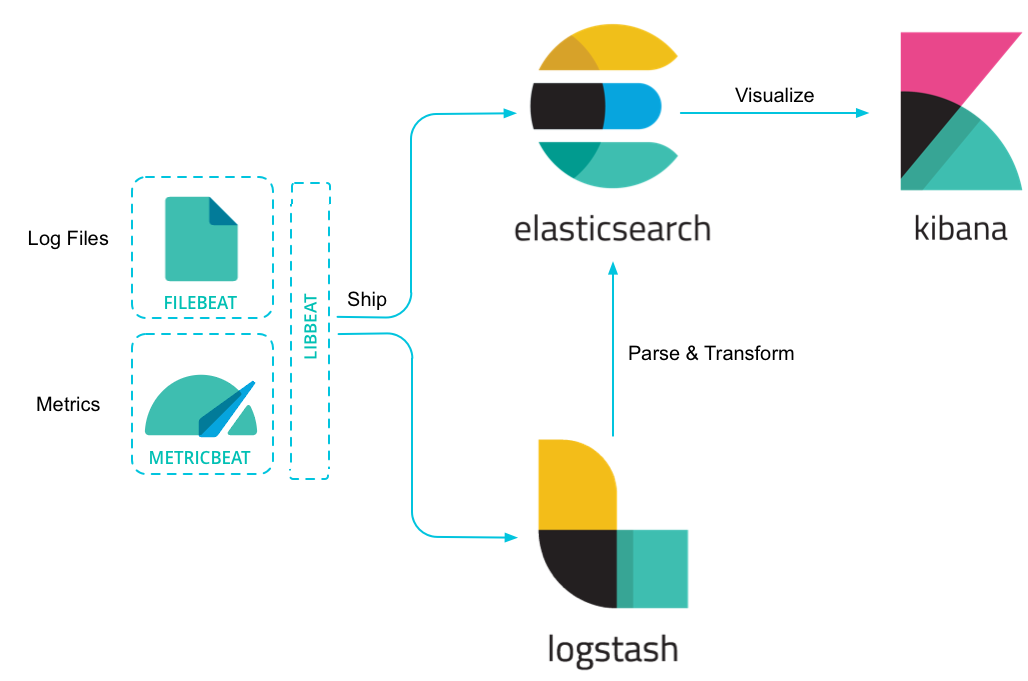
Logstash permet la mise en forme des données reçues d’une multitude de source. Il utilise des Pipelines pour gérer les différentes sources en même temps et peut rediriger les logs vers différentes plateformes de stockage comme Elasticsearch

## Metricbeat

Metricbeat est un collecteur d’indicateurs système (CPU, mémoire, I/O, disks). Il permet dans ce cas, de récupérer les informations machine de chaque container pour connaitre leur consommation de ressources sur les machines physiques.

## Filebeat

Filebeat surveille des fichiers pour envoyer leur contenu vers ELK. Il est alors possible de surveiller le contenu des fichiers de logs de chaque container sur la machine host.



## Logging driver

Filebeat se connecte aux fichiers de logs de chaque container. Mais les logs ne sont identifiés que par l’identifiant du container et non pas son nom, ce qui rend difficile l’identification du container. Filebeat remonte l’information brute contenu dans les fichiers.

Il est possible de ne pas utiliser Filebeat, mais uniquement logstash. Il suffit de rediriger les logs envoyés par les containers, (utilisation du logging driver gelf). Les logs sont alors bien reconnaissables par le nom des containers.

Mais les logs ne sont plus présents dans les fichiers de logs des containers. ELK est le seul système stockant des logs.

# Installation

L’installation a été faite sur une VM de xubuntu (version 16.04) avec Docker (17.12.0-ce) et Docker-compose (1.19.0)

L’installation utilise le Logging driver « gelf » pour collecter les logs, et les pousser vers logstash.

## Logging Driver

Pour changer le Logging Driver de tous les containers, il est nécessaire de changer le fichier « daemon.json »

/etc/docker/daemon.json

On change le driver par défaut de Docker en ajoutant les lignes suivantes

{

"log-driver": "gelf",

"log-opts":{

"gelf-address":"udp://localhost:12201"

}

}

Le fichier est disponible sur le gitlab

<https://gitlab.nicecotedazur.org/SmartCities/monitoring>

## Logstash

On utilise 2 pipelines pour Logstash (logs et metricbeat). On utilise alors 3 fichiers de configurations. « pipelines.yml », « logs.conf » et « beats.conf »

pipelines.yml :

- pipeline.id: logs

path.config: "/usr/share/logstash/pipeline/logs.conf"

- pipeline.id: beats

path.config: "/usr/share/logstash/pipeline/beats.conf"

logs.conf :

input {

gelf {

port => "12201"

}

}

output {

elasticsearch {

hosts => elasticsearch

}

}

beats.conf :

input {

beats {

port => "5044"

}

}

output {

elasticsearch {

hosts => elasticsearch

index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"

}

}

On utilise les ports 12201 et 5044 pour les logs des containers ainsi que les metrics. Il est possible de les changer si besoin.

## Metricbeat

On utilise un fichier de configuration « metricbeat.yml » :

metricbeat.modules:

- module: docker

metricsets:

- info

- container

- cpu

- memory

- diskio

- network

- healthcheck

hosts: ["unix:///var/run/docker.sock"]

# hosts: ["tcp://vmlx2080tech:2376"]

period: 10s

fields:

env: dev

output:

logstash:

hosts: ["logstash:5044"]

setup.kibana.host: "kibana:5601"

setup.dashboards.enabled: true

Ce fichier de config permet de définir quelle information remonter par Metricbeat. Il faut le connecter au « docker.sock » avec « hosts » dans l’exemple il est mis en volume dans le container mais on peut aussi utiliser une connexion TCP.

On le connecte au Logstash. La connexion avec Kibana permet de charger des dashboard pour les données de Metricbeat.

Elasticsearch et Kibana ne demandent pas de configuration supplémentaire.

## Serveur web & redis

Un serveur en python est ajouté pour générer des entrées de log, il n’est pas utile de le mettre en place sur le serveur. L’image du serveur se crée avec le docker-compose qui build à partir du Dockerfile.

## Docker-compose

Le docker compose met en place un Elasticsearch, un Logstash, un Kibana et un Metricbeat sur le même réseau. Il est conseillé de rediriger les logs des composants de monitoring en « json-file » pour éviter des surgénérer du log. Un log est généré à chaque log reçu, une boucle risque d’être créée.

On ajoute à chaque service dans le docker-compose les lignes suivantes.

logging:

driver: json-file

## Droit d’accès

Il peut être nécessaire de données les droits du « docker.sock » dans Metricbeat. Pour cela il faut se connecter au container et entrer la commande suivante

chown metricbeat /var/run/docker.sock

Si « docker.sock » a été mis en local.