ESILV LÉONARD DE VINCI | ÉCOLE D'INGÉNIEURS PARIS-LA DÉFENSE

IoT Cloud
TP4 – Presentation

1 Introduction

Dans ce TP, vous allez jouer avec un outil de *Presentation* afin d'en comprendre l'intérêt, les avantages et les limites. Vous allez en particulier découvrir *Grafana* [Grab].

Pour rappel, vous devez rendre un rapport individuel à la fin du semestre. Vous devez y mettre toutes les informations qui vous semblent pertinentes. Notez que les réponses aux questions encadrées en gris seraient à considérer comme un minimum à faire apparaitre dans ce rapport.

2 Présenter... mais quoi ?

C'est cool, on a des "objets connectés", ils produisent des données, on a une architecture qui claque et permet d'absorber une quantité impressionante de données (*RabbitMQ* [Rab]), de les router en fonction des besoins spécifiques (*Nifi* [Apa]) et de les stocker (*MongoDB* [Mon]).

Et après ? On paie des serveurs de plus en plus gros et on s'enorgueillit d'avoir plein plein de données ? Ou on en fait quelque chose ?

Vous allez aujourd'hui travailler sur un *Dashboard* permettant de suivre les données collectées. Dans un premier temps, vous devez :

1. Définir et expliquer les différentes informations pertinentes à présenter et la façon de les présenter (vous devez faire une maquette quoi...) qui vous permettra de mettre en avant les données des clients d'après les informations du *Fil Rouge*.

Pour vous aider, quelques questions à vous poser :

— Si je suis client, quelles sont les informations pertinentes qui je voudrais voir tous les jours ? (*Data storytelling*...)
— Si je vois une information étrange, comment je voudrais pouvoir fouiller la donnée, quelles représentations m'aideraient à comprendre une situation ? (Fouille de données)
— Si je suis un gestionnaire (energie ? sécurité ?) et que j'ai plusieurs clients, quelles informations seraient pertinentes à voir ?

3 Déployer... *InfluxDB* [Infa]

L'outil que nous allons utiliser, *Grafana*, ne permet pas de se connecter simplement à une base de données *MongoDB* [Osg]. Nous allons donc commencer par déployer une nouvelle base de données orientée *Time Series*, *InfluxDB* [Infa]. Pour cela, vous devez :

2. Déployer un serveur *InfluxDB* [Infc] avec une interface graphique *Chronograf* [Infb] à partir du *docker-compose* proposé à la Figure 1.
3. Se connecter à l'interface *Chronograf* pour créer une base de données : `CREATE DATABASE "iot"`
4. Les données générées durant le TP1 qui sont envoyées sur une *queue RabbitMQ* sont sous format JSON (exemple : `{"DATE": "1570985503", "SENSOR": "SDBPuiss", "VALUE": "201.54"}`), or *InfluxDB* ne peut recevoir de fichiers JSON mais peut recevoir des données suivant le *InfluxDB line protocol* [Infe] (par exemple : `client1 SDBPuiss=201.54 1570985503`). Nous allons utiliser un *flow Nifi* pour réaliser cette conversion. Ajouter un processeur *Nifi ExecuteScript* [Nifa] avec `ECMAScript` comme *ScriptEngine* et basé sur le code proposé en Figure 2.
5. Finalement, comme présenté en Figure 3, ajouter un processeur *Nifi PutInfluxDB* [Nifb] permettant d'envoyer les données vers une base de données *InfluxDB*.
6. Sinon, vous pouvez télécharger le *flow Nifi* mis à votre disposition sur votre LMS... Enfin, si vous lisez bien le sujet en entier avant de commencer...
7. Une fois que vous aurez envoyé quelques données à partir du générateur proposé au TP1 puis lancé votre *flow Nifi*, vous devriez donc avoir des données sur votre base de données *InfluxDB*. Pour vérifier, retournez sur l'interface *Chronograf* puis testez la commande `SELECT * FROM "iot"."autogen"."client1"` (si votre base de données est nommée `"iot"` et votre *measurement* [Infd] est nommé `"client1"`.), comme présenté en Figure 4.

```yaml
version: '3.7'
services:

    influxdb:
        image: "influxdb"
        hostname: "influxdb"
        ports:
            - "8086:8086"
        networks:
            - iot-labs
        labels:
            NAME: "influxdb"

    influxdbchrono:
        image: "chronograf"
        hostname: "chronograf"
        ports:
            - "8087:8888"
        networks:
            - iot-labs
        labels:
            NAME: "chronograf"
        command: "--influxdb-url=http://influxdb:8086"

networks:
    iot-labs:
        external: true
```

FIGURE 1 – Exemple de *docker-compose* permettant de déployer un serveur *InfluxDB* [Infc] avec une interface graphique *Chronograf* [Infb].

```javascript
flowFile = session.get();
if (flowFile != null) {
    var StreamCallback = Java.type("org.apache.nifi.processor.io.StreamCallback");
    var IOUtils = Java.type("org.apache.commons.io.IOUtils");
    var StandardCharsets = Java.type("java.nio.charset.StandardCharsets");
    var error = false;
    var measure = "client1"
    var line = "";
    var sep = "\n"

    // Get attributes
    flowFile = session.write(flowFile, new StreamCallback(function (inputStream, outputStream) {
        var content = IOUtils.toString(inputStream, StandardCharsets.UTF_8); // message or content
        var message_content = {};
        var sensor = "";
        var date = "";
        var value = "";

        try {
        message_content = JSON.parse(content);
        for (key in message_content){
            if (key == 'SENSOR') {
                sensor = message_content[key]
            } else if (key == 'DATE') {
            date = message_content[key] * 1000 * 1000 * 1000
            } else if (key == 'VALUE') {
            value = message_content[key]
            }
        }
        line = measure + " " + sensor + "=" + value + " " + date + sep

        // Write output content
        outputStream.write(line.getBytes(StandardCharsets.UTF_8));

        } catch (e) {
        error = true;
        log.error('Something went wrong', e)
        outputStream.write(content.getBytes(StandardCharsets.UTF_8));
        }
    }));

    if (error) {
        session.transfer(flowFile, REL_FAILURE)
    } else {
        session.transfer(flowFile, REL_SUCCESS)
    }
}
```

FIGURE 2 – Code *ECMAScript* utilisable sur un processeur *Nifi ExecuteScript* [Nifa] permettant de convertir un JSON (exemple : *{"DATE" : "1570985503", "SENSOR" : "SDBPuiss", "VALUE" : "201.54"}*) vers un texte compatible avec le *InfluxDB line protocol* [Infe] (par exemple : *client1 SDBPuiss=201.54 1570985503*).

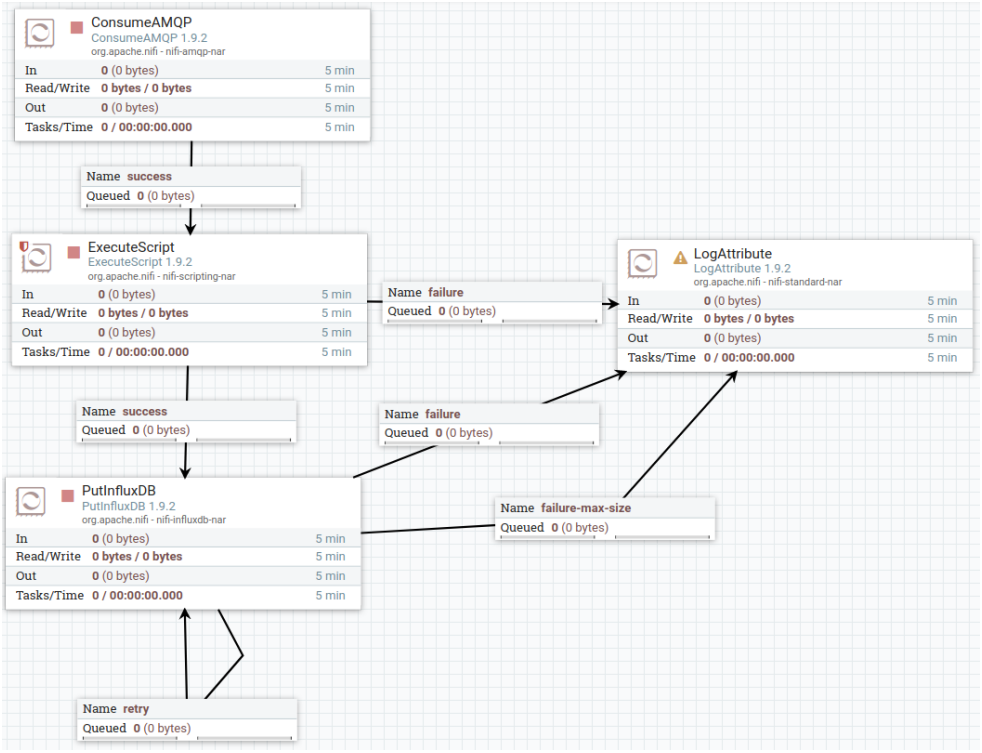

FIGURE 3 – *Flow Nifi* permettant de récupérer des données JSON sur une *queue RabbitMQ*, de les convertir avec un code comme présenté en Figure 2 puis de les envoyer vers une base de données *InfluxDB*.

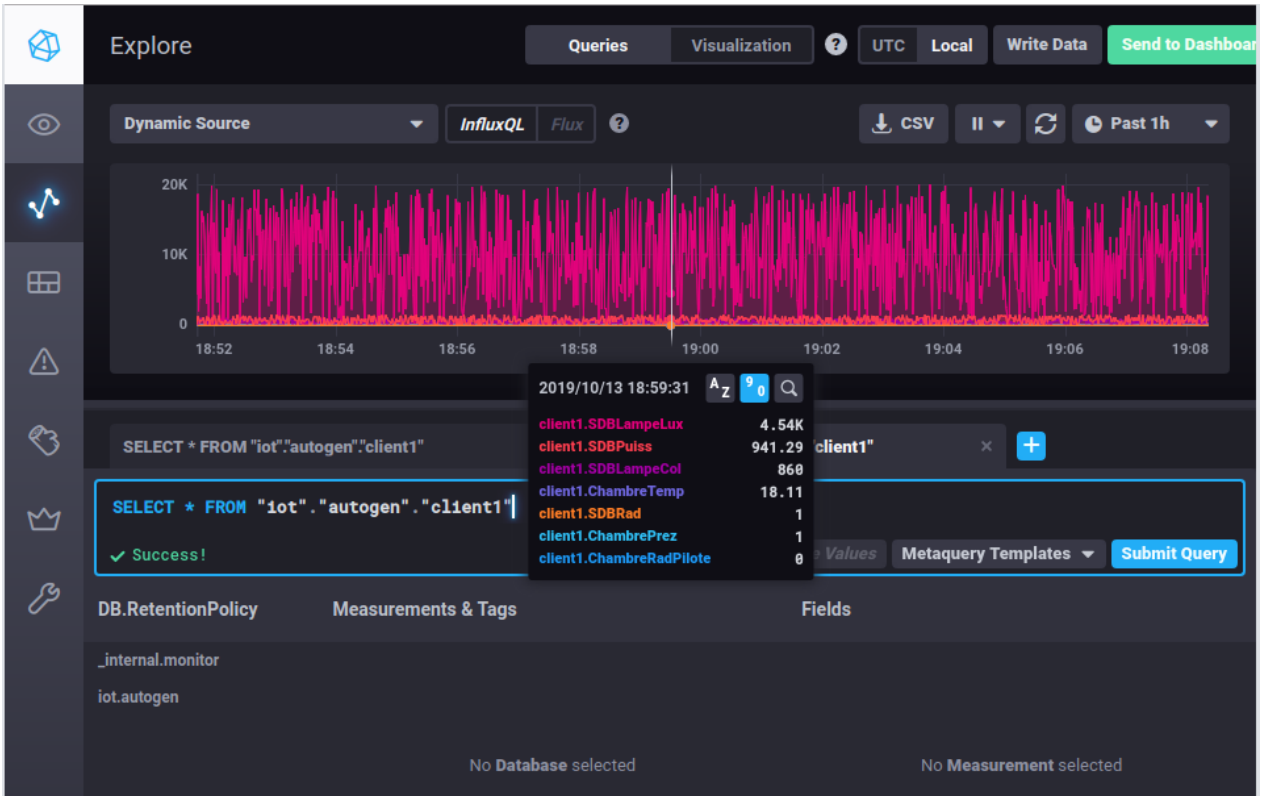

FIGURE 4 – Résultat d'une requête sur *Chronograf* permettant de vérifier les données d'une base de données *InfluxDB*.

4 Déployer... *Grafana* [Grab]

Vous devez maintenant :

8. Déployer un serveur *Grafana* [Grab] à partir du *docker-compose* proposé à la Figure 5.
9. Une fois connecté à l'interface, ajouter une source de données *InfluxDB* en configurant bien l'*URL* et la *Database*.
10. Nous considérons seulement deux capteurs, un capteur de puissance instantanée et un capteur de température. Créer un dashboard comme présenté en Figure 6 contenant :
    - Une gauge indiquant la température actuelle (en vert jusqu'à 20°C, orange jusqu'à 25°C puis en rouge jusqu'à 35°C).
    - Une gauge indiquant la température moyenne.
    - Un *singlestat* indiquant la consommation moyenne à l'heure en kWh.
    - Un graphique contenant deux courbes :
        - La courbe de puissance instantanée indiquant en légende les valeurs min/max/moyenne/courante. Vous utiliserez l'axe Y de gauche pour cette courbe.
        - La courbe de la température avec la même légende. Vous utiliserez l'axe Y de droite pour cette courbe en le limitant entre 18°C et 30°C.
        - Vous ajouterez à la courbe de puissance un seuil à 1000kW permettant de voir rapidement les dépassements.

```yaml
version: '3.7'
services:

    grafana:
        image: "grafana/grafana"
        hostname: "grafana"
        environment:
            GF_SECURITY_ADMIN_PASSWORD: "secret"
        ports:
            - "8084:3000"
        networks:
            - iot-labs
        labels:
            NAME: "grafana"

networks:
    iot-labs:
        external: true
```

FIGURE 5 – Exemple de *docker-compose* permettant de déployer un serveur *Grafana* [Graa].

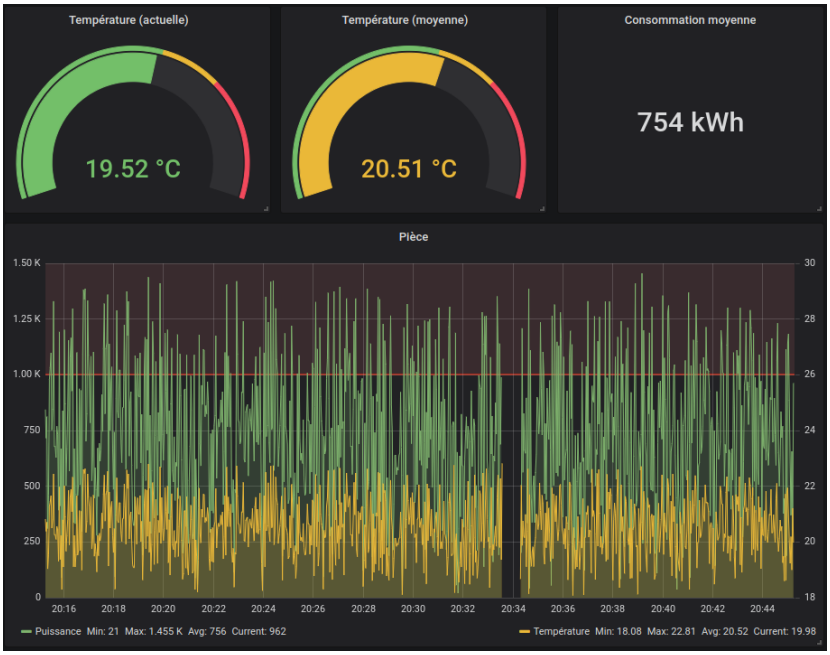

FIGURE 6 – *Dashboard* à reproduire sur *Grafana*.

Références

[Apa] Apache. *Nifi*. URL : https://nifi.apache.org/.

[Graa] Grafana. *Image Docker officielle de Grafana*. URL : https://hub.docker.com/r/grafana/grafana/.

[Grab] Grafana. *Site web principal*. URL : https://grafana.com/.

[Infa] InfluxDB. *Site web principal*. URL : https://www.influxdata.com/.

[Infb] InfluxData. *Image Docker officielle de Chronograph*. URL : https://hub.docker.com/_/chronograf.

[Infc] InfluxData. *Image Docker officielle de InfluxDB*. URL : https://hub.docker.com/_/influxdb.

[Infd] InfluxData. *InfluxDB key concepts : measurement*. URL : https://docs.influxdata.com/influxdb/v1.7/concepts/key_concepts/#measurement.

[Infe] InfluxData. *InfluxDB line protocol tutorial*. URL : https://docs.influxdata.com/influxdb/v1.7/write_protocols/line_protocol_tutorial/.

[Mon] *MongoDB*. URL : https://www.mongodb.com/.

[Nifa] Nifi. *ExecuteScript documentation*. URL : https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-scripting-nar/1.6.0/org.apache.nifi.processors.script.ExecuteScript.

[Nifb] Nifi. *PutInfluxDB documentation*. URL : https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-influxdb-nar/1.6.0/org.apache.nifi.processors.influxdb.PutInfluxDB/.

[Osg] James Osgood. *MongoDB datasource for Grafana*. URL : https://github.com/JamesOsgood/mongodb-grafana.

[Rab] *RabbitMQ*. URL : https://www.rabbitmq.com/.