Rapport du projet CNAM-NFE204 - Etude d'un système de base de données NOSQL

InfluxDB

Fabrice DUNAN <fabrice.dunan@laposte.net>

Table des matières

https://gitlab.com/logrus_fr/CNAM-projets/NFE204

Fabrice DUNAN <fabrice.dunan@laposte.net>

v1, 2018-05-18

tuteur: P.Rigaux

Préface : Quel SGBD NOSQL étudier ???

Partir des données…

En tentant naïvement l'analyse de données ouvertes, on constate souvent l'existence d'une variable de date voire d'horodatage.
L'objet des séries temporelles ou séries chronologiques est l'étude des variables au cours du temps.
Les algorithmes étudiés dans les autres UE CNAM que l'on a suivi ne se focalisent pas sur l'étude des séries temporelles…Cela n'empêche pas de s'y intéresser !

Des sources de données "Time series" communes : les mesures physiques (des données météo vont être exploitées dans la suite de ce rapport), les données produites par des objets connectés mais aussi la surveillance d'un système d'exploitation de serveur où l'on veut mesurer des métriques système d'un composant comme la CPU, la RAM, le stockage (quantité ou I/O)…tout ce qui fait un serveur peut être ingéré, traité, interrogé au sein d'une base "Time series" et représenté graphiquement pour surveillance, suivi, "capacity planning", etc, ….
Même une application informatique "serveur" comme Gitlab, challenger de GitHub dans le domaine de la gestion de dépots Git, embarque InfluxDB pour surveiller l'état de sa JVM comme les mesures sur le "Garbage collecting" et d'autres encore. Cette DB "time series" est donc particulièrement adaptée aux outils "devOps". Cette utilisation peut se généraliser à toute surveillance métier pour peu que le métier en question émette périodiquement des données chronologiques.
Les DB "time series" s'adaptent très bien à l'analytique temps réel où les données sont produites et exploitées en temps réel.

… un moteur de recherche ? …

Ces données horodatées sont parfois immuables. C'est le cas des prises de mesure (physique, météo, ressources système informatique..) ou des journaux informatiques.

Les moteurs de recherche étaient de bons candidats.

Pour autant, les SGBD NOSQL vus en cours NFE204 sont exclus des candidats possibles, y compris les moteurs d'indexation comme ElasticSearch ou Solr.

Plutôt qu'étudier un 3ème moteur d'indexation, tout aussi adapté à la gestion des données horodatées, on se tournerait bien vers l'originalité de leur compagnon habituel, les DB clé valeur qui sont souvent utilisés comme cache dans un contexte streaming.

… ou leur compagnon ?

Hélas, leurs langages de requête étaient trop pauvres, leurs schémas quasi absents et leurs fonctionnalités trop spécialisées. On a finalement écarté l'étude de "SGBD clé valeur" type Riak ou ReDis.

Les SGBD NOSQL "time series"

On revient donc sur l'idée initiale d'étudier un des nombreux SGBD de la famille "time series" étudiés pour ce genre de données.

Comparatif

L’idée initiale était d’étudier deux membres de cette famille, un par binôme, dans des projets NFE204 distincts, pour ne retenir que le "meilleur" dans les autres projets.
Faute de temps et de coordination, ce comparatif ne sera pas traité.

1. Objectif du document

Ce document est le rapport d'étude du SGBD NOSQL "time series" **InfluxDB**. Il a été élaboré dans le cadre de l'UE NFE204, visant à l'obtention du certificat de données massives du CNAM.

Son objectif est à la fois de découvrir le SGBD NOSQL pré-cité, la famille de SGBD à laquelle il appartient, de constituer une base de données documentaire et enfin d'approfondir un ou plusieurs des aspects de ce SGBD.

Il sera constitué tout d'abord d'une première partie de présentation du SGBD, telle qu'on aurait pu la faire en se documentant dans les détails, sans manipulation ni test.

Cette partie sera suivie d'une partie dédiée à l'utilisation unitaire et simpliste de la base.

Les deux points que l'on souhaite approfondir dans InfluxDB, son langage de requêtes et son fonctionnement en tant que cluster, seront couverts par les deux dernières parties :

On expérimentera InfluxDB et son langage de requêtes plus en profondeur et autour d'une thématique de données massives. Seule la partie interaction avec la base de donnée de la problématique sera traitée dans ce document.

Enfin, on survolera la partie administration de la base dans une dernière partie, mais en précisant les aspects cluster d'InfluxDB.

2. InfluxDB "sur le papier"

A l'heure de la rédaction de ce document, InfluxDB se maintient selon certaines sources comme la TSDB leader.

Nous allons tenter de décortiquer ce système le plus possible pour comprendre pourquoi il est tant plébiscité. On commence cette enquête par une étude documentaire.

2.1. TimeSeries DB, kesako ?

Dans le cas de stockage de résultats de mesures, les SGBDR classiques ne sont pas forcément la panacée.
Un certain nombre d'outils s'y substituaient déjà, comme RRDTool.

Mais l'absence des caractéristiques des SGBD NOSQL comme la gestion de (très) gros volumes de données, parler JSON, une interface REST, un langage de requêtes, la réplication, le partitionnement…ont laissé une place à de nouveaux système capables également de gérer les séries chronologiques, les événements ou les métriques: "Time Series DataBases" (abrégé *TSDB* par la suite).

2.2. Historique

Le projet InfluxDB a démarré en **octobre 2013** par une société du nom d'Errplane fondée par *Paul Dix* et *Todd Persen*.

Cette société a été initialement financée par un fond d'investissement de la Silicon Valley nommé Y-combinator, connu également pour avoir financé Docker, entre autres.
Après une levée de fonds de 8M$ en 2014, Errplane change de nom en 2015 pour devenir InfluxData inc. Fin 2016, InfluxData lève 16M$.

La dernier version en date, publiée en **avril 2018**, est la **1.5.2**.

2.3. Modèle commercial, licence et support

Le modèle économique d'InfluxDB est répandu : la base du SGBD est open source, les fonctionnalités propriétaires (HA, scalabilité, backup/restore avances...) sont contenues dans la version commerciale **InfluxEntreprise**.

La licence d'utilisation est gratuite pour un unique serveur.
Le code de cette version "community" dite **influxDB OSS** est Open source (licence MIT).

Pour accéder aux fonctionnalités des cluster InfluxDB, une licence commerciale est obligatoire.
Le test du cluster peut être réalisé gratuitement sur une période limitée sur la plateforme Cloud d' Influxdata. Ce test est couvert dans un chapitre ultérieur du projet.
L'offre cloud **InfluxCloud** hébergée par AWS.
Pour l'offre cloud :
- entre 250 et 500$/mois cloud standard
- 900 et 1500$/mois pour le cloud avance/professionnel
la distinction se faisant sur les ressources allouées.

Les gammes de prix des différents services sont détaillés dans le lien précédent.

Le support est fourni par Influxdata.

2.4. Utilisateurs et domaines d'utilisation notoires

InfluxDB est utilisé par eBay, Cisco pour ne citer qu'eux...

On résume ci-après les domaines communiqués par InfluxData et déjà mentionnés plus haut :

- IoT et capteurs
- "monitoring DevOps"
- "Analytics" temps réel

2.5. Les entrailles d'InfluxDB

Dans ce chapitre, on survole les caractéristiques internes d'InfluxDB.

2.5.1. Fonctions & limites

InfluxDB comporte les fonctions listées ci-après, certaines étant la conséquence d'autres :

SGBD NoSQL "moderne":

- schéma des données
- langage d'interrogation
- protocoles :
  - HTTP API REST
  - JSON over UDP

Performances et tolérance aux pannes

- réplication
- partitionnement
- passage à l'échelle
- HA au sein d'un cluster

Certaines des limites sont listées ici, d'autres là. En synthèse ci-dessous, la liste des fonctionnalités qu' InfluxDB n'a pas :

- index secondaires
- procédures stockées
- Fonctions de Map&Reduce
- Clés étrangères
- Transaction
- Jointure

Néanmoins, il parait nécessaire de rester attentif aux évolutions car dans le monde des SGBD NOSQL, les fonctionnalités tendent à s'harmoniser et le projet InfluxDB est très actif !

2.5.2. Systèmes d'exploitation supportés

Les systèmes d'exploitation sur lesquels peuvent fonctionner influxDB sont les distributions Linux commerciales principales et leurs contreparties libres, certains UNIX libres et un OS commercial.

Table 1. Tableau des OS supportés

nom OS	
Ubuntu	Debian
RHEL	CentOS
SLES	OpenSuse
FreeBSD/PC-BSD	
MacOSX	

Dans la partie mise en pratique de ce rapport, on utilisera exclusivement InfluxDB sur Ubuntu.

2.5.3. Choix de conception

Langage avec lequel InfluxDB a été développé

InfluxDB est écrit en Go, le langage crée par Google.

Moteur d'influxDB

InfluxDB était initialement fondé sur le moteur clé/valeur **LevelDB de Google**. D'après la documentation, et pour des raisons approfondies dans le lien suivant, il utilise maintenant le moteur BoltDb écrit en Go.

Architecture

InfluxDB OSS est d'un point de vue OS un processus multithreadé qui écoute sur le port **8086**.

On verra plus tard dans ce document, dans le chapitre ultérieur, un focus sur l'architecture cluster d'InfluxDB Entreprise.

2.6. Ecosystème client

2.6.1. Client CLI

Le client en ligne de commande est présent dans l'installation d'influxdb. On privilégiera son utilisation dans la suite de l'étude.

On trouvera les détails techniques en annexe

2.6.2. API REST

L'API REST complète les moyens pour interagir avec InfluxDB. Elle permet l'exploitation des codes retour HTTP, de l'authentification HTTP, des token JWT pour le SSO, ce, en recevant et envoyant des documents JSON.

Peu utilisée directement dans ce projet, on pourra consulter des détails en annexe. Néanmoins, il faut garder à l'esprit que le CLI ou les bibliothèques clientes exploitent cette API.

2.6.3. Langages et bibliothèques InfluxDB

Ci-dessous, le tableau résumant les langages de développement fournissant des bibliothèques pour inter-opérer avec InfluxDB:

Table 2. Tableau des langages

langags					
.Net	Clojure	Erlang	Go	Haskell	Java
JavaScript	JavaScript (Node.js)	Lisp	Perl	PHP	*Python*
R	Ruby	Rust	Scala		

2.6.4. La "stack TICK"

InfluxDB propose également une suite logicielle "compagnon", composée de 4 éléments remplissant des fonctionnalités complémentaires à InfluxDB.

Elle sera décrite en annexe car finalement peu ou pas utilisée dans cette étude, ce qui sera d'ailleurs justifié lorsque l'on abordera le choix sur la méthode d'import de données.

2.6.5. Clients IHM

Le client d'administration web embarqué est déprécié depuis la 1.1, absent depuis la 1.3 au profit de Chronograf, qui est décrit plus loin.

Il existe aussi un IDE sous Windows, en bêta, non testé dans cette étude.

2.7. Le modèle de données d'InfluxDB

On verra dans cette partie les spécificités de la représentation de la donnée dans InfluxDB.

La ligne de la table, appellée **"point"** dans InfluxDB, suit la grammaire du "Line Protocol" que l'on résumera ci dessous.

Une ligne de la table nommée "point"

```
<measurement>[,<tag-key>=<tag-value>...] <field-key>=<field-value>[,<field2-
key>=<field2-value>...] [unix-nano-timestamp]
```

Les détails du line protocol ainsi qu'un tutorial sur le line protocol sont dans la documentation officielle.

Dans les chapitres suivants, on décrira les types d'éléments constitutifs du point respectant le Line Protocol mentionnés ci dessus.

2.7.1. Premier élément : les "measurements" ou mesures

La mesure ou "measurement" est la table d'InfluxDB : Elle joue le rôle de container colonnes des tags, "fields" et "timestamps".

2.7.2. Second élément "optionnel" : les "tags"

Les *tags* sont les métadonnées et à ce titre leurs valeurs sont indexées. InfluxDB distingue *tag key* le nom de la colonne, de *tag value*, sa valeur. Leur type est contraint aux chaines de caractères.

Les tags sont optionnels mais importants car c'est sur eux que vont porter les critères des requêtes.

Pour simplifier, on parlera de tag pour designer le couple (tag key, tag, value).

2.7.3. Troisième élément : les "fields" ou champs

Ce sont les données contenues dans InfluxDB. Elles sont donc à ce titre obligatoires. Par contre, elles ne sont pas indexées. En cas de requête, le moteur scannera toute la mesure !

Les valeurs des fields sont typées et peuvent être :
* des chaines de caractères
* des entiers, avec un suffixe pour indiquer ce type (10i)
* des nombres réels
* des booléens sans sensibilité à la casse pour la syntaxe (true, FALSE…)

Pour simplifier, on parlera de field ou champ pour désigner le couple (field key, field, value).

2.7.4. Dernier mais non le moindre : le "timestamp" ou estampille

Par défaut, le timestamp est indiqué en nombre de nanosecondes écoulées depuis le 01/01/1970.

Sans mention de la date dans une insertion, c'est la date courante qui sera fournie.

Lors d'une requête, sans contrainte sur la valeur de time, les données retournées seront comprises entre 1/1/70 et l'instant de la requête.

A dire vrai, si ce format est adapté à des capteurs, à l'utilisation lors de tests unitaires humains, c'est un peu ennuyant.

Mais le CLI et l'API Influx DB permettent de donner une précision plus confortable. On verra comment par la suite.

Par contre, seule l'API **permet d'insérer un timestamp avec un format de date particulier…**

2.7.5. La "retention policy"

Trois points concernant ce paramètre important pour la fiabilité des données:

1. Dans la retention policy d'InfluxDB, on trouvera naturellement la durée de conservation des données. Cette partie sera reprise dans le chapitre administration.
2. Dans le contexte d'un cluster, on aura aussi les paramètres de réplication, c'est à dire sur combien de noeuds les "shard" seront répliqués. (Se référer au chapitre clustering pour un approfondissement du sujet)
3. Enfin l'objet retention policy comprend également le paramètre "Shard group duration" qui stipule l'intervalle de temps dont est responsable un shard group. Autrement dit et par exemple, un shard group distinct sera crée tous les 7 jours par défaut pour contenir les mesures liées à cet intervalle de temps. C'est un mode de fonctionnement propre à InfluxDB.

Par défaut, InfluxDB crée une RP nommée '*autogen*' dont les caractéristiques sont les suivantes :

- Rétention : durée infinie
- Facteur de réplication = 1
- "Shard Group duration" d'une semaine

On pourra obtenir plus d'informations dans la documentation sur les retention policy.

2.7.6. Les Séries

Une série est une collection de données qui a en commun :

- la mesure
- le ou les mêmes valeurs de tags
- la politique de rétention

Les points qui ont les mêmes tags dans une même mesure appartiennent à la même série de points.

```
point is uniquely identified by the measurement name, tag set, and timestamp. If you
submit Line Protocol with the same measurement, tag set, and timestamp, but with a
different field set, the field set becomes the union of the old field set and the new
field set, where any conflicts favor the new field set.
```

2.7.7. Les bonnes pratiques sur la création de schémas

Les bonnes pratiques sur la création de schémas InfluxDB sont décrites dans la documentation.

On en retiendra quelques éléments :

- Les tags ne doivent pas avoir trop de modalités sous peine d'une consommation excessive de mémoire.

IMPORTANT	C'est un écueil que l'on a expérimenté lors des tests en fixant comme tag la température ou la pression. La quantité de mémoire consommée par InfluxDB devient telle que la machine de tests devient inutilisable (swap)…

- On choisit un type tag à une variable à laquelle on veut appliquer un GROUPBY.
- On choisit un type "field" à une variable à laquelle on veut appliquer une fonction influxQL.

2.8. Le langage d'interrogation d'InfluxDB : InfluxQL (ou IQL)

InfluxDB propose un langage de requête complet qui ressemble au SQL, dénommé InfluxQL que l'on abrègera par IQL par la suite.

La documentation référence d'influxQL est ici.

2.8.1. Un langage de requêtes à priori similaire au SQL…

On retrouve les mnémoniques et qualifieurs classiques et des originalités :

- *INSERT*

- *SELECT*
- *WHERE*
- *GROUP BY*
- *INTO* qui permet de stocker les résultats d'une requête dans une autre base; cela peut être utilisé dans le chapitre continuous query.
- *LIMIT*
- *OFFSET*
- *ORDER BY*
- *DESC / ASC*
- *AS*
- *FILL* permet de retourner une valeur par défaut pour un intervalle vide résultant d'une requête de groupBy.

On peut également effectuer des *calculs* sur les fields et des *expressions régulières* sur les tags. (cf exemples infra)

On ne décrira pas ici le fonctionnement de chacune des ces instructions. Le lecteur avec une expérience du SQL se rendra immédiatement compte des fonctions. Pour les autres, des exemples d'utilisation seront présentés dans le chapitre approfondissement.

Par contre, on ne trouve ni UPDATE, ni DELETE.

2.8.2. … mais CR-ud pas CRUD …

Si jusque là, InfluxDB possède de nombreuses caractéristiques d'un SGBD relationnel classique, il s'en distingue car il ne comporte pas d'UPDATE et qu'un DELETE au comportement spécifique existe.

```
"written once, rarely updated"
```

La signification des données est rarement unitaire mais prend son sens sur un large groupe de mesures (les times series)

Par conséquent, InfluxDB privilégie la création et les requêtes sur la mise à jour et la destruction; le tout pour focaliser sur la performance des 2 premiers critères. La documentation résume cela par CR-ud au lieu de CRUD pour les SGBDR SQL.

Il faut donc utiliser des contournements pour faire des opérations de mise à jour :

- Pour mettre à jour un point, il faut l'insérer à nouveau avec le même ensemble measurement, tag(s) et timestamp.
- On peut supprimer une série mais pas directement des points. Le contournement consistant à trouver le timestamp d'une valeur de "field" donnée et le supprimer.

2.8.3. ...et sans jointure

Pour des raisons liées au domaine des TSDB, au choix de conception et à la performance, IQL ne propose pas de jointure, comme expliqué qans le lien précédent.

Par contre InfluxData semble proposer un nouveau langage, **IFQL**, pour un plan de requête moins coûteux et pour pouvoir exprimer ces requetes "select-project-join" puissantes, mais complexes.

2.8.4. Les fonctions IQL

IQL propose 4 familles de fonctions, riches, très bien décrites dans la documentation.

On propose des tableaux de synthèse des fonctions IQL en annexe.

2.9. Protocoles et formats

Outre le "LineProtocol" sur TCP, InfluxDb est également compatible avec les protocoles suivants :

- CollectD
  Le format est celui de collectd, le protocole transport est UDP.
- Graphite
  Grâce à un plugin, Influx accepte le format Graphite.
- OpenTSDB
  InfluxDB supporte les protocoles telnet and HTTP d'OpenTSDB. InfluDB peut donc remplacer OpenTSDB !
- Prometheus
  InfluxDB fournit l'API Prometheus pour la lecture et l'écriture par un processus de conversion en format InfluxDB.

3. Utilisation superficielle

L'objectif de ce chapitre est d'illustrer ce que l'on peut faire avec InfluxDB concrètement, sachant les informations théoriques du chapitre précédent.

On va tout d'abord travailler avec un jeu de données minimal, un sous-ensemble de données physiques (température, pression), l'exploitation en profondeur avec des données météo plus complètes étant dédiée à un chapitre ultérieur.

La partie Installation et paramétrage est reportée en annexe car non nécessaire à la compréhension de la suite du document. Le lecteur souhaitant mettre en place un environnement de tests pourra s'y reporter.

3.1. Création (C)

Dans ce chapitre, on explore les capacités de création d'objets InfluxDB.

3.1.1. Base

Création par CLI d'une base

```
> create database test_mesures
> show databases
name: databases
name
---
_internal
test_mesures
```

Alternative : Création par API REST

```
user@M40474:~$ curl -i -XPOST http://localhost:8086/query --data-urlencode "q=create
database test_mesures_REST"
HTTP/1.1 200 OK
Content-Type: application/json
Request-Id: a1367134-0803-11e8-8175-000000000000
X-Influxdb-Build: OSS
X-Influxdb-Version: 1.5.2
X-Request-Id: a1367134-0803-11e8-8175-000000000000
Date: Fri, 02 Feb 2018 10:27:11 GMT
Transfer-Encoding: chunked

{"results":[{"statement_id":0}]}
```

Chaque commande CLI a son équivalent API. On donnera des exemples CLI exclusivement à partir de ce point.

3.1.2. "measurements" ou tables

La table est crée à la première insertion en suivant la grammaire "line Protocol". Il n'a pas de commande au sens propre pour en créer.

3.1.3. Insertion

Dans cette partie, on va voir les différentes façons de (bien) insérer des "tags", "fields" et éventuellement timestamp dans un "measurement".

Ligne par ligne par CLI (avec la politique de rétention par défaut)

```
> USE test_mesures
Using database test_mesures
> INSERT mesure_station,id_station=zzzz,altitude_station=500i
temperature=15,pression=1.0 1421100000000000
SELECT * FROM mesure_station WHERE id_station=='zzzz'
> SELECT * FROM mesure_station WHERE id_station='zzzz'

name: mesure_station
time             altitude_station id_station pression temperature
----             ---------------- ---------- -------- -----------
1421100000000000 500i             zzzz       1        15
```

Ligne par ligne par API REST

```
curl -XPOST 'http://localhost:8086/write?db=test_mesures' \
--data-binary 'mesure_station,id_station=xxxx,altitude_station=200i
temperature=20,pression=1.2 1421000000000000'
```

En batch

Pour insérer massivement des données, le CLI avec option -import permet de le faire pour peu que l'on se plie au Line protocol InfluxDB.

un exemple tiré de la documentation

```
# DDL
CREATE DATABASE pirates
CREATE RETENTION POLICY oneday ON pirates DURATION 1d REPLICATION 1

# DML
# CONTEXT-DATABASE: pirates
# CONTEXT-RETENTION-POLICY: oneday

treasures,captain_id=dread_pirate_roberts value=801 1439856000
treasures,captain_id=flint value=29 1439856000
treasures,captain_id=sparrow value=38 1439856000
treasures,captain_id=tetra value=47 1439856000
treasures,captain_id=crunch value=109 1439858880
```

DDL signifiant Data definition language, la partie dédiée à la création des database et des politiques de rétention.

DML signifiant Data Manipulation Language, partie qui sélectionne Database et Politique de rétention puis insère les points.

Une alternative pour des données "brutes" consiste à utiliser Telegraf, plus aisée si l'on ne veut pas faire de développement, mais contrainte sur les formats de données, comme on le verra plus tard.

3.2. Interrogation (R)

3.2.1. Lister les bases

```
> show databases
name: databases
name
---
_internal
```

_internal contient les métadonnées internes et les données de profiling.

3.2.2. Informations sur les "measurements" et ses colonnes

Pour afficher les mesures

```
> use test_mesures
Using database test_mesures
> show measurements
name: measurements
name
---
mesure_station
```

Lister les "fields

```
> SHOW FIELD KEYS
name: mesure_station
fieldKey    fieldType
--------    ---------
pression    float
temperature float
```

Lister les tags

```
> SHOW TAG KEYS
name: mesure_station
tagKey
------
altitude_station
id_station
```

Lister les séries

```
> show series
key
---
mesure_station,altitude_station=200i,id_station=xxxx
```

3.2.3. Interrogations "classiques"

Lister tout le contenu de la mesure "mesure_station"

```
> select * from mesure_station
name: mesure_station
time             altitude_station id_station pression temperature
----            ---------------- ---------- -------- -----------
1421000000000000 200i             xxxx       1.2      20
```

La même chose en spécifiant directement database, retention policy et measurement

```
> SELECT * FROM "test_mesures"."autogen"."test_mesures"
name: test_mesures
time                           altitude_station id_station pression temperature
----                           ---------------- ---------- -------- -----------
2018-02-03T17:21:54.574244521Z 2000i            yyyy       0.8      30
```

Le format de la date est également plus lisible.

Filtrer une valeur ... en parcourant toute la mesure

```
>select * from mesure_station where temperature < 50
name: mesure_station
time             altitude_station id_station pression temperature
----             ---------------- ---------- -------- -----------
1421000000000000 200i             xxxx       1.2      20
```

Filtrer une valeur en utilisant un tag donc un index

```
>select * from mesure_station where id_station = 'xxxx'
name: mesure_station
time             altitude_station id_station pression temperature
----             ---------------- ---------- -------- -----------
1421000000000000 200i             xxxx       1.2      20
```

IMPORTANT Utiliser la syntaxe suivante pour avoir des dates lisibles par un être humain.

```
>precision rfc3339
>select * from mesure_station where id_station = 'xxxx'
name: mesure_station
time                 altitude_station id_station pression temperature
----                 ---------------- ---------- -------- -----------
1970-01-17T10:43:20Z 200i             xxxx       1.2      20
```

3.3. Mises à jour (U)

Comme déjà expliqué, la mise à jour et la suppression ont un comportement spécifique dans InfluxDB.

Un point est identifié de manière unique par son "measurement", l'ensemble de ses tags et son estampille.

D'après la documentation et après expérimentation, le résultat de la "mise à jour" d'un point, c'est à dire l'insertion d'un point avec les mêmes identifiants uniques résultera par l*union* des deux points, le plus récent prenant le pas avec ses valeurs le cas échéant.

Un exemple illustrant la mise à jour adaptée de la documentation officielle

```
> INSERT mesure_station,id_station=zzzz,altitude_station=500i temperature=15

> select * from mesure_station  where time = 1525808295545544168
name: mesure_station
time                 altitude_station id_station pression temperature
----                 ---------------- ---------- -------- -----------
1525808295545544168 500i             zzzz                15
```

On remarque que la valeur de pression n'est pas renseignée.

insertion du "même" point avec un field supplémentaire

```
> INSERT mesure_station,id_station=zzzz,altitude_station=500i
temperature=15,pression=1.0 1525808295545544168
```

le point "union" résultant

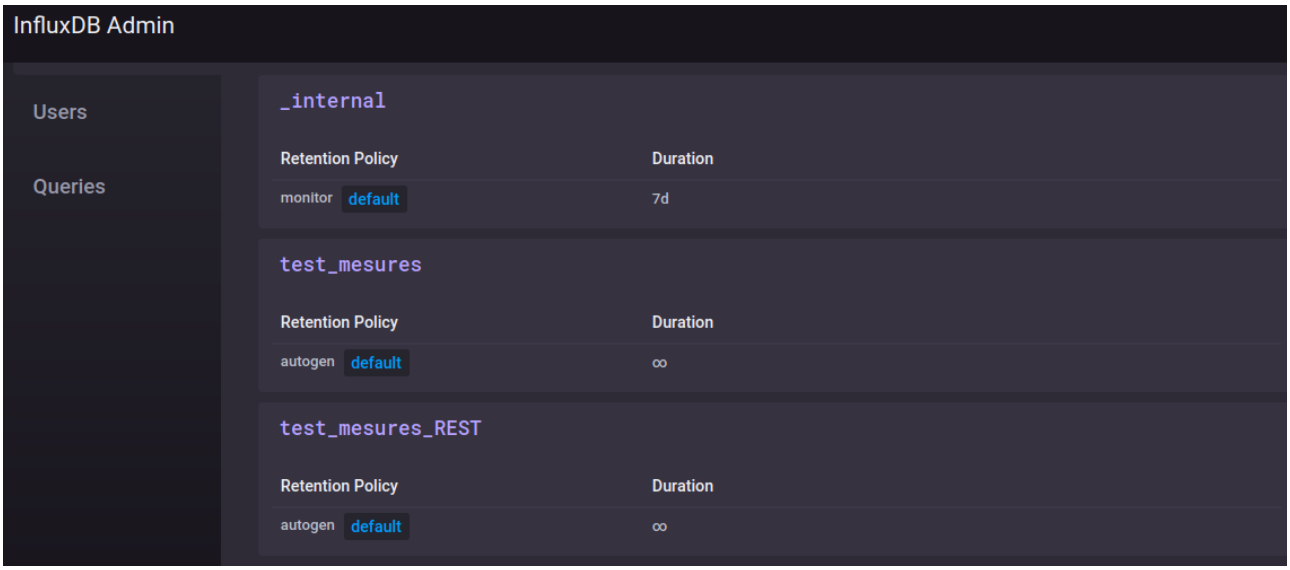
```
> select * from mesure_station  where time = 1525808295545544168
name: mesure_station
time                altitude_station id_station pression temperature
----                ---------------- ---------- -------- -----------
1525808295545544168 500i             zzzz       1        15
```

3.4. Suppression… (D)

InfluxDB permet de supprimer, via des commandes d'administration, un ensemble de points :

3.4.1. …de bases

- Suppression de base `DROP DATABASE <database_name>`
- Alternative suppression par chronograf



3.4.2. …de séries

C'est la manière de supprimer ses essais /erreurs :

- Suppression de séries `DROP SERIES FROM <measurement_name[,measurement_name]> WHERE <tag_key>='<tag_value>'`

exemple :

```
> select * from synop
name: synop
time                -10     -520    -60  -666666 0       0.000000 0.200000 07005
101620 102510  170 19290   2.000000 2.900000 20171101000000 278.050000 280.050000
281.550000 3.200000 7 87
----                ---     ----    ---  ------- -       -------- -------- -----
------ ------  --- -----   -------- -------- -------------- ---------- ----------
---------- -------- - --
1525937705274858290 -10     -510    -60  -666666 0       0        -0.1     7015
101890 102470  190 11470   3.6      4.4      20171101000000 278.25     279.95
279.75     5.1      7 89
...
1525937705494431967 -10     -666666 -160 -666666 21      -666666  -666666  7460  98050
101940  100 10000   1        -666666  20040601000000 285.35     285.95     -666666
2.1      6 96
> DROP SERIES FROM synop
> SELECT * FROM synop
```

Une alternative :

- suppression des points de la série contrairement à précédemment:
  - sans supprimer la série de l'index
  - supporte le where sur le timestamp

```
DELETE FROM <measurement_name> WHERE [<tag_key>='<tag_value>'] | [<time interval>]
```

3.4.3. ...de "measurements"

Suppression de tables `DROP MEASUREMENT <measurement_name>`

3.4.4. Et les points ?

Par contre, il est impossible de supprimer un point autrement que par une politique de rétention.

TIP	Il existe un contournement mentionné par la documentation qui consiste à rechercher la valeur du field, puis le timestamp correspondant puis effectuer un DELETE fondé sur le timestamp.

4. Approfondissement et fonctionnalités SGBD pour une problématique de données massives

4.1. La problématique de données "massives"

Ayant étudié principalement des données catégorielles en STA211, et pour préparer InfluxDB à une utilisation "massive", on s'est fixé quelques objectifs :

- Traiter un sujet ayant trait à des mesures physiques
- Que ces mesures puissent être importanttes en terme de variables et d'individus
- Que le sujet produise des mesures compatibles avec InfluxDB...

Après quelques échanges lors des jalons NFE204, le visionnage des arrière-plans des vidéos de ce cours et la mine rejouie des enseignants STA211 lors de leurs déplacements, l'objectif d'une étude météorologique et, en particulier, de la météo de communes plus méridionales que Paris comme Nice, s'est imposé.

4.2. Choix des données

Le site de Météofrance fournit des données publiques, gratuites et librement utilisables pour peu, d'après la licence d'utilisation que l'on mentionne sa source et paternité ... ce qui est chose faite conformément aux instructions du lien précédent dans la rubrique "Condition d'accès".

Ces données contiennent des informations intelligibles au profane de la météorologie: température, pression, lieu de mesure, vitesse du vent, enneigement...

Descriptif des données choisies issu du site de meteoFrance

```
Données d'observations issues des messages internationaux d'observation en surface
(SYNOP) circulant sur le système mondial de télécommunication (SMT) de l'Organisation
Météorologique Mondiale (OMM). Paramètres atmosphériques mesurés (température,
humidité, direction et force du vent, pression atmosphérique, hauteur de pr
écipitations) ou observés (temps sensible, description des nuages, visibilité) depuis
la surface terrestre. Selon instrumentation et spécificités locales, d'autres param
ètres peuvent être disponibles (hauteur de neige, état du sol, etc.)
```

Nous découvrons ainsi une émanation des Nations Unies, impliquée dans les problématiques écologiques actuelles, et ses moyens techniques de partage de mesures météorologiques.

Quelques caractéristiques des données SYNOP mises à disposition dans les diverses rubriques de la page leur étant dédiée:

- 1 fichier par mois
- données qui couvrent du 1/1996 au 11/2017 donc 22 ans

- 1 mois = 14000 lignes en moyenne donc près de 4M de lignes
- format CSV
- fréquence de production d'un point par station : toutes les 3h

En [annexe_SYNOP], le tableau décrivant les mnémoniques des colonnes du fichier CSV, leur types et unités :

Ci-après, un aperçu de quelques colonnes :

numer_sta	date	pmer	tend	cod_tend	dd	ff	t	td	u
7005	20170501000000	100160	60	0	150	2.700000	282.450000	280.850000	90
7015	20170501000000	100270	190	2	200	4.600000	283.450000	281.850000	90
61996	20170501030000	100890	-80	5	300	12.300000	290.350000	286.050000	76
61997	20170501030000	101850	-120	8	240	15.400000	276.650000	275.150000	90
7181	20170501060000	101190	210	3	190	5.200000	279.650000	278.450000	92
7190	20170501060000	101210	390	3	270	3.000000	280.950000	277.850000	81
7335	20170506120000	101210	160	3	240	9.600000	287.850000	282.850000	72
7434	20170506120000	101350	190	1	250	6.100000	285.750000	281.450000	75

On trouvera en annexe ou dans la rubrique information sur les stations du site dédié aux données les diverses correspondances entre codes numer_sta et noms de villes, paramètres géodésiques et altitude

4.3. Récupération des données

Les données sont disponibles sur le site internet de MétéoFrance via un formulaire.

Après une rapide analyse du fonctionnement de ce formulaire, on développe un script pour récupérer l'ensemble des données en une passe pour constituer un jeu conséquent.

Ce code du script de récupération des données meteofrance est fourni en annexe.

4.4. Préparation des données

Après exploration des données, on découvre que les valeurs manquantes sont marquées 'mq', ce qui est facile pour les distinguer de données numériques.

InfluxDB acceptant les valeurs manquantes, une amélioration du script d'import décrit ci-après permettra de résoudre ce problème.

Par contre, le choix est d'ignorer les lignes où *tous les champs valent mq*, car InfluxDB impose au moins un "field" non vide.

4.5. Les choix de modélisation

Une fois les données choisies et découvertes, se pose la question de leur représentation dans InfluxDB : Quelle variable indexer? Que considérer comme "field" ?

On décide que seuls l'horodatage et le code de la station de mesure nous permettront d'obtenir les données météo utiles (température, pression, vent, humidité…). Ces dernières seront considérées comme des "field".

En effet, et l'on découvre une caractéristique importante d'InfluxDB, on ne peut pas obtenir dans un `SELECT` les "tags values" sans préciser un "field".
Par exemple, la réquête `SELECT numer_sta FROM synop` … ne renvoie aucune valeur !
Par contre, `SELECT temperature,numer_sta FROM synop` renverra bien les informations espérées (en annexe, les références à cette "découverte").

Manifestement, la question est sujette à débats récents ici et surtout là

4.6. Quel outil choisir pour importer ?

Une fois le schéma choisi, il faut maintenant importer les données. Pour ce faire, plusieurs solutions se présentent. La suite de ce chapitre éclairera le lecteur sur le choix effectué.

4.6.1. Logstash

Logstash est un outil d'import transformation et export de données, composant de la suite compagnon d'ElastiSearch, versatile car fondé sur de multiples plugins…et déjà connu de l'auteur !
C'est donc naturellement la première solution qui émerge, d'autant que la documentation de Telegraph référence Logstash comme exemple de son parseur Grok.
On imagine alors une solution du type "plugin input file" plus un "filter csv" combiné avec le plugin "output influxdb"…mais l'objectif étant de découvrir InfluxDB et son environnement, si l'on doit utiliser un ETL autant utiliser la stack TICK (et ainsi avoir la bonne stack TICK…) .

4.6.2. Telegraf

On se tourne donc vers Telegraf, décrit en annexe.
La configuration par défaut, utilisable immédiatement, permet la capture d'un nombre impressionnant de métriques, du système d'exploitation ou d'outils classiques.

Néanmoins, le format CSV n'est pas supporté directement et ce, même si CSV est un format répandu et que la demande d'évolution a été faite auprès d'InfluxData.

C'est regrettable car JSON est supporté nativement dans InfluxDB … mais les données ne sont pas en JSON !

On a trouvé a posteriori les mêmes données météo en JSON mais sur un site tiers : Faute de temps pour accréditer ces données, on préfèrera l'original à la copie.

4.6.3. API InfluxDB par bibliothèque Python

Plutôt que se contraindre à étudier une configuration et l'adapter hypothétiquement à son besoin, on s'oriente vers le développement d'un script autour de la bibliothèque Python InfluxDB. Si c'est une approche qui s'adaptera au besoin, elle peut consommer du temps de mise au point.

Pour ne pas réinventer la roue, on évalue des codes disponibles sur Internet. Si le premier essai est infructueux, le second aboutit moyennant quelques modifications.

On choisit d'importer l'ensemble des données. On rappelle la répartition des colonnes de la table :
* Estampille
* Un unique tab (numer_sta)
* 57 fields

On va importer 22 ans de données (de 1996 à 2017), c'est à dire 264 mois soit, si un mois contient approximativement 15000 lignes, **4 millions de lignes**.

WARNING	Quand tous les "fields values" sont manquantes, on a le message d'erreur suivant à l'import. Le "point" sera simplement ignoré.

Message d'erreur "lignes sans fields"

```
influxdb.exceptions.InfluxDBClientError: 400: {"error":"partial write: unable to parse
'synop,numer_sta=81415,pres=0,t=304.850000  1487689200000000000': invalid field format
dropped=0"}
```

Import des données météo dans un conteneur InfluxDB

```
21:08:30-user@M40474:~/Bureau/CNAM/git/CNAM-projets/NFE204(master)$ for i in
donnees/extractions/*.csv;do python3 ../../csv-to-influxdb/csv-to-influxdb.py -d';' -s
'172.17.0.2:8086' -m 'synop' -tc 'date' -tf '%Y%m%d%H%M%S' --fieldcolumns
pmer,tend,dd,ff,td,u,vv,hbas,tnN,txN,rafper,pres,t,cod_tend,td,vv,ww,w1,w2,n,nbas,cl,c
m,ch,niv_bar,geop,tend24,tn12,tn24,tx12,tx24,tminsol,sw,tw,raf10,rafper,per,etat_sol,h
t_neige,ssfrai,perssfrai,rr1,rr3,rr6,rr12,rr24,phenspe1,phenspe2,phenspe3,phenspe4,nnu
age1,ctype1,hnuage1,nnuage2,ctype2,hnuage2,nnuage3,ctype3,hnuage3,nnuage4,ctype4,hnuag
e4  --tagcolumns numer_sta --dbname meteo --input $i;done
Read 5000 lines
Inserting 5000 datapoints...
Wrote 5000, response: True
Read 10000 lines
...
```

Les 59 champs importés

```
> SHOW FIELD KEYS FROM synop
name: synop
fieldKey fieldType
-------- ---------
ch       float
cl       float
...
w1       float
w2       float
ww       float
```

La récupération de lignes

```
> SELECT * FROM synop LIMIT 3
name: synop
time                  ch cl cm cod_tend ctype1 ctype2 ctype3 ctype4 dd  etat_sol ff
geop hbas hnuage1 hnuage2 hnuage3 hnuage4 ht_neige n   nbas niv_bar nnuage1 nnuage2
nnuage3 nnuage4 numer_sta per perssfrai phenspe1 phenspe2 phenspe3 phenspe4 pmer
pres  raf10 rafper rr1 rr12 rr24 rr3 rr6 ssfrai sw t      td     tend tend24 tminsol
tn12   tn24 tw tx12   tx24 u  vv    w1 w2 ww
----                  -- -- -- -------- ------ ------ ------ ------ --  -------- --
---- ---- ------- ------- ------- ------- -------- -   ---- ------- ------- -------
------- ------- --------- --- --------- -------- -------- -------- -------- ----
----  ----- ------ --- ---- ---- --- --- ------ -- -      --     ---- ------ -------
----   ---- -- ----   ---- -  --    -- -- --
1996-01-01T00:00:00Z          1                                  170          2
07005                                                  100030 99090
0             276.04 275.95 100                                          99
1996-01-01T00:00:00Z 11 30 23 4                                  100          2
2250                                          25  1
89642                                                  98820  98290
271.15 261.65 10                  265.65         271.65      44 40000
1996-01-01T00:00:00Z 60 62 61 1                                  0            0
101                                           07015
100020 99380                              0            275.95 275.35 100
96 300   4  4  45

...
```

WARNING	On constate dès cette première requête de test...une augmentation importante (2Go) de la mémoire occupée par le container InfluxDB

Nombre de lignes insérées

```
> SELECT COUNT(*) FROM synop
...
```

IMPORTANT	La commande fait swapper la machine et elle devient inutilisable.

TIP	Pour tuer une requête longue, cf deux solutions: par le CLI `KILL QUERY` ou simplement `CTRL+C`

4.7. Résoudre les problèmes d'import

4.7.1. Réduire le nombre de points

Après une lecture de la documentation des paramètres InfluxDB, on décide de :
⇒ stopper la base de surveillance _internal
⇒ modifier la configuration du cache pour lancer le serialisation sur disque plus souvent (10m a 1m)
⇒ baisser la valeur maximale du cache
⇒ réduire la mémoire occupée par le cache de l'index

...et réduire la quantité de données...

4.7.2. Réduire le nombre de fields

On finit par choisir de limiter le nombre de colonnes pour pouvoir importer plus d'individus sur l'infrastructure de tests.

Enumération des colonnes

```
> SHOW FIELD KEYS
name: synop
fieldKey fieldType
-------- ---------
dd       float
ff       float
hbas     float
pmer     float
pres     float
rafper   float
t        float
td       float
tend     float
u        float
vv       float
```

Import d'un sous-ensemble des colonnes (13) sur l'intégralité des individus

```
21:57:11-user@M40474:~/Bureau/CNAM/git/CNAM-projets/NFE204(master)$ for i in
donnees/extractions/*.csv;do python3 ../../csv-to-influxdb/csv-to-influxdb.py -d';' -s
'172.17.0.2:8086' -m 'synop' -tc 'date' -tf '%Y%m%d%H%M%S' --fieldcolumns
pmer,tend,dd,ff,td,u,vv,hbas,tnN,txN,rafper,pres,t  --tagcolumns numer_sta --dbname
meteo --input $i;done
Read 5000 lines
Inserting 5000 datapoints...
Wrote 5000, response: True
Read 10000 lines
...
```

Récupération de quelques lignes

```
> SELECT * FROM synop LIMIT 5
name: synop
time                  dd  ff  hbas numer_sta pmer   pres  rafper t      td     tend u
vv
----                  --  --  ---- --------- ----   ----  ------ -      --     ---- -
--
1996-01-01T00:00:00Z 170 2         07005     100030 99090        276.04 275.95 100  99
1996-01-01T00:00:00Z 100 2   2250 89642     98820  98290        271.15 261.65 10   44
40000
1996-01-01T00:00:00Z 0   0         07015     100020 99380        275.95 275.35 100  96
300
1996-01-01T00:00:00Z 120 4.1 250  07027     99940  99130        279.04 279.04 0    100
1200
1996-01-01T00:00:00Z 110 1   70   07130     99900  99380        281.25 280.65 -10  96
800
```

Comptage des lignes

```
> SELECT COUNT(*) FROM synop
name: synop
time                  count_dd count_ff count_hbas count_pmer count_pres count_rafper
count_t count_td count_tend count_u count_vv
----                  -------- -------- ---------- ---------- ---------- ------------
------- -------- ---------- ------- --------
1970-01-01T00:00:00Z 3323727  3324681  1998697    3252545    2891372    1776188
3380581 3366987  2840858    3365351 2479722
```

Il semble que ce sous-ensemble de donnée soit utilisable dans le contexte de l'infrastructure de tests mis à disposition.

Avant de tirer des informations des données, on interroge InfluxDB sur sa manière de les stocker.

Si InfluxDB partitionne les données en shards comme de nombreux SGBD NOSQL, une originalité est qu'InfluxDB les regroupe en groupes d'intervalles de temps finis.

Constater le regroupement de shards par intervalles de temps : les "shard groups"

```
> SHOW SHARDS
...
name: meteo
id   database retention_policy shard_group start_time           end_time
expiry_time          owners
--   -------- ---------------- ----------- ----------           --------
-----------          ------
3021 meteo    autogen          3021        1996-01-01T00:00:00Z 1996-01-08T00:00:00Z
1996-01-08T00:00:00Z
3022 meteo    autogen          3022        1996-01-08T00:00:00Z 1996-01-15T00:00:00Z
1996-01-15T00:00:00Z
...
4109 meteo    autogen          4109        2017-12-18T00:00:00Z 2017-12-25T00:00:00Z
2017-12-25T00:00:00Z
4110 meteo    autogen          4110        2017-12-25T00:00:00Z 2018-01-01T00:00:00Z
2018-01-01T00:00:00Z
```

Par contre, il ne faut pas se fier à la colonne expiry_time, les données de 1996 étant fort heureusement toujours disponibles :

Requêter les données antérieures à 1997

```
> SELECT * FROM synop WHERE time < '1997-01-01'
name: synop
time                 dd  ff   hbas numer_sta pmer   pres   t      td     tend  u   vv
----                 --  --   ---- --------- ----   ----   -      --     ----  -   --
1996-01-01T00:00:00Z 170 2         07005     100030 99090  276.04 275.95 100   99
1996-01-01T00:00:00Z 100 2    2250 89642     98820  98290  271.15 261.65 10    44
40000
...
```

On peut aller plus loin avec des commandes telles que influx_inspect.

La commande SHOW STATS nous donne entre autres l'occupation mémoire :

```
name: runtime
Alloc     Frees     HeapAlloc HeapIdle   HeapInUse HeapObjects HeapReleased HeapSys
Lookups Mallocs   NumGC NumGoroutine PauseTotalNs Sys        TotalAlloc
-----     -----     --------- --------   --------- ----------- ------------ -------
------- -------   ----- ------------ ------------ ---        ----------
110615032 206060142 110615032 1774403584 190300160 366717      1599184896   1964703744
86306   206426859 166   17           110406088    2078956024 17800933456
...
```

Le nombre de requêtes et leur temps d'exécution

```
...
name: queryExecutor
queriesActive queriesExecuted queriesFinished queryDurationNs recoveredPanics
------------- -------------- --------------- --------------- ---------------
1             13              12              14139604864     0
...
```

Le nombre de séries dans la base météo

```
...
name: database
tags: database=meteo
numMeasurements numSeries
--------------- ---------
1               62
...
```

Et ce qui explique la consommation mémoire:

Le cache par shard

```
...
name: shard
tags: database=meteo, engine=tsm1, id=3021,
path=/var/lib/influxdb/data/meteo/autogen/3021, retentionPolicy=autogen,
walPath=/var/lib/influxdb/wal/meteo/autogen/3021
diskBytes fieldsCreate seriesCreate writeBytes writePointsDropped writePointsErr
writePointsOk writeReq writeReqErr writeReqOk
--------- ------------ ------------ ---------- ------------------ --------------
------------- -------- ----------- ----------
145804    27913        59           0          0                  0              3090
1         0           1

...

name: tsm1_cache
tags: database=meteo, engine=tsm1, id=3021,
path=/var/lib/influxdb/data/meteo/autogen/3021, retentionPolicy=autogen,
walPath=/var/lib/influxdb/wal/meteo/autogen/3021
WALCompactionTimeMs cacheAgeMs cachedBytes diskBytes memBytes snapshotCount
writeDropped writeErr writeOk
------------------- ---------- ----------- --------- -------- -------------
------------ -------- -------
25                  996        462000      0         0        0              0
0         1
...
```

4.8. Traitement

On propose dans cette partie un certain nombre de requêtes qui permettront d'illustrer les fonctionnalités d'InfluxDB. Au fil de ces requêtes, on découvrira des informations sur la problématique des données météo.
Dans une étude de données complètes, ces informations fourniraient la base à une exploitation par des algorithmes de traitement de données massives et surtout une analyse plus approfondie; un exemple ambitieux pouvant être de produire un modèle de prédiction…

4.8.1. Quelques requêtes simples

Décaler la sélection de points

```
> SELECT * FROM synop LIMIT 3 OFFSET 15
name: synop
time                  dd  ff  hbas    numer_sta pmer   pres   rafper t          td
tend tnN txN u  vv
----                  --  --  ----    --------- ----   ----   ------ -          --
---- --- --- -  --
2017-01-01T00:00:00Z 300 5.7 -999999 07661     102770 101120 9.6    281.750000 277.35
-100 0   0   74 -999999
2017-01-01T00:00:00Z 330 6.5 -999999 07690     102750 102420 8.2    279.150000 270.05
-130 0   0   52 50000
2017-01-01T00:00:00Z 0   0   30      07037     102830 100870 0      268.650000 267.65
-190 0   0   93 130
```

Ordonner la sélection par critère chronologique décroissant

```
> SELECT * FROM synop ORDER BY time DESC LIMIT 3
name: synop
time                  dd  ff  hbas numer_sta pmer   pres   rafper t      td     tend u
vv
----                  --  --  ---- --------- ----   ----   ------ -      --     ---- -
--
2017-12-31T21:00:00Z 50  3.8      81415     101230 100020        298.45 295.95 10   86
2017-12-31T21:00:00Z 230 7.2      07005     100030 99160  22.2   280.85 277.45 120  79
20000
2017-12-31T21:00:00Z 0   0   450  81405     101220 101130 1.5    298.85 297.65 -30  93
28540
```

Enumération en créant une colonne résultat

```
> SELECT COUNT(pmer) AS nb FROM synop
name: synop
time                  nb
----                  --
1970-01-01T00:00:00Z 3252545
```

Nombre de mesures de températures à Nice (07690)

```
> SELECT COUNT(t) FROM synop WHERE numer_sta = '07690'
name: synop
time                count
----                -----
1970-01-01T00:00:00Z 58596
```

Donner la force du vent en m/s, sa direction en °, la vitesse des rafales en m/s, la pression au niveau de la mer en Pa à Nice à quelques heures de 2018

```
> SELECT dd,ff,rafper,pmer FROM synop WHERE time > '2017-12-31T00:00:00Z' AND
numer_sta = '07690'
name: synop
time                dd  ff  rafper pmer
----                --  --  ------ ----
2017-12-31T03:00:00Z 330 4   5.4    101880
2017-12-31T06:00:00Z 350 4.4 6.7    101950
2017-12-31T09:00:00Z 330 6.5 8.2    102070
2017-12-31T12:00:00Z 320 3   7.2    101980
2017-12-31T15:00:00Z 180 2.1 3.1    101810
2017-12-31T18:00:00Z 320 3.7 5.4    101750
2017-12-31T21:00:00Z 330 4.3 6.8    101670
```

Mêmes mesures quand la vitesse du vent moyen excède 72 km/s vitesse du vent

```
> SELECT numer_sta,dd,ff,rafper,pmer FROM synop WHERE ff > 20 LIMIT 5
name: synop
time                numer_sta dd  ff   rafper pmer
----                --------- --  --   ------ ----
1996-01-10T00:00:00Z 07661     70  21          100790
1996-01-15T15:00:00Z 61998     270 22.1        100280
1996-01-18T12:00:00Z 61997     220 21.6        100320
1996-01-20T15:00:00Z 89642     180 25.2        97820
1996-01-20T15:00:00Z 61997     320 20.5        99660
```

IMPORTANT	La requête est beaucoup plus lente car le where porte sur un "field" ce qui implique un "full scan" du "measurement".

TIP	L'endroit de mesure est sans doute situé outremer, en tout cas ce code n'est pas référencé dans le fichier de correspondance fourni par MétéoFrance.

4.8.2. "Jouer" avec les "temps"

La température à Nice fin juin

```
>  SELECT t -273 FROM synop WHERE time > '2017-06-29T09:00:00Z' AND time < '2017-06-
30' AND numer_sta = '07690'
name: synop
time                  td     numer_sta
----                  --     ---------
2017-06-29T12:00:00Z 288.55 61972
2017-06-29T15:00:00Z 288.35 61972
2017-06-29T18:00:00Z 289.15 61972
2017-06-29T21:00:00Z 289.15 61972
```

La température à Nice à 5h du jour de l'an 2018

```
> SELECT td,numer_sta FROM synop WHERE time > '2017-12-31T09:00:00Z' - 5h AND
numer_sta = '07690'
name: synop
time                  td     numer_sta
----                  --     ---------
2017-12-31T06:00:00Z 277.15 07690
2017-12-31T09:00:00Z 276.95 07690
2017-12-31T12:00:00Z 278.75 07690
2017-12-31T15:00:00Z 281.25 07690
2017-12-31T18:00:00Z 278.55 07690
2017-12-31T21:00:00Z 278.25 07690
```

La température de rosée à Nice le jour de la naissance de mon fils après 7h, 133m et 1456 secondes

```
> SELECT td,numer_sta FROM synop WHERE time > '2011-05-15' +7h + 133m +1456s AND
numer_sta = '07690' LIMIT 3
name: synop
time                  td     numer_sta
----                  --     ---------
2011-05-15T12:00:00Z 284.05 07690
2011-05-15T15:00:00Z 282.25 07690
2011-05-15T18:00:00Z 281.35 07690
```

En Martinique l'humidité en % autour du 28/05/2017 à 3h

```
> SELECT numer_sta,u FROM synop WHERE numer_sta =~ /78.../ AND time = '2017-05-28' +
3h
name: synop
time                  numer_sta u
----                  --------- -
2017-05-28T03:00:00Z 78897     93
2017-05-28T03:00:00Z 78922     87
2017-05-28T03:00:00Z 78925     81
```

4.8.3. Agrégations

Humidité moyenne à Nice en février 2017

```
> SELECT MEAN(u) FROM synop WHERE time > '2017-02-01' AND time < '2017-02-28' AND
numer_sta = '07690'
name: synop
time                           mean
----                           ----
2017-02-01T00:00:00.000000001Z 72.57309941520468
> SELECT MEAN(u) FROM synop WHERE time > '2017-02-01' AND time < '2017-02-28' AND
numer_sta = '07690'
```

Humidité moyenne en % par station

```
> SELECT MEAN(u) FROM synop GROUP BY numer_sta
name: synop
tags: numer_sta=07005
time                 mean
----                 ----
1970-01-01T00:00:00Z 82.10907789818033

name: synop
tags: numer_sta=07015
time                 mean
----                 ----
1970-01-01T00:00:00Z 79.44397775774993
...
```

Hauteur des nuages moyenne à Nice par semaine entre avril et mai

```
> SELECT MEAN(hbas) FROM synop WHERE numer_sta = '07690' AND time >= '2017-04-
06T00:00:00Z' AND time <= '2017-05-06T00:00:00Z' GROUP BY time(7d)
name: synop
time                 mean
----                 ----
2017-04-06T00:00:00Z 772.5
2017-04-13T00:00:00Z 719.2307692307693
2017-04-20T00:00:00Z 611.1111111111111
2017-04-27T00:00:00Z 1057.4285714285713
2017-05-04T00:00:00Z 855
```

4.8.4. Fonctions de sélection

Les 5 températures les plus chaudes en °C à Nice pendant l'été 2017

```
> SELECT TOP(t,5)-273 FROM synop WHERE numer_sta = '07690' AND time >= '2017-06-01'
AND time <= '2017-09-01'
name: synop
time                 top
----                 ---
2017-06-17T15:00:00Z 30.25
2017-06-17T18:00:00Z 29.94999999999999
2017-07-24T12:00:00Z 34.05000000000001
2017-08-01T12:00:00Z 30.350000000000023
2017-08-06T12:00:00Z 31.55000000000001
```

Les températures les plus froides à Nice

```
 SELECT BOTTOM(t,3)-273 FROM synop WHERE numer_sta = '07690'
name: synop
time                 bottom
----                 ------
1999-01-31T03:00:00Z -1.1499999999999773
1999-01-31T06:00:00Z -1.75
2005-01-29T06:00:00Z -1.5500000000000114
```

La hauteur minimale des nuages à Nice

```
> SELECT MIN(hbas) FROM synop WHERE numer_sta = '07690'
name: synop
time                 min
----                 ---
2000-04-22T18:00:00Z 2
```

La dernière mesure de hauteur des nuages à Nice

```
> SELECT LAST(hbas) FROM synop WHERE numer_sta = '07690'
name: synop
time                 last
----                 ----
2017-12-31T21:00:00Z 450
```

4.8.5. Transformation

Différence entre deux valeurs successives

```
> SELECT hbas FROM synop WHERE numer_sta = '07690' AND time >= '2017-04-06T00:00:00Z'
AND time <= '2017-04-06' + 2d
name: synop
time                 hbas
----                 ----
2017-04-06T12:00:00Z 800
2017-04-06T15:00:00Z 800
2017-04-06T18:00:00Z 1250
2017-04-07T06:00:00Z 450
2017-04-07T09:00:00Z 450
2017-04-07T12:00:00Z 250
2017-04-07T15:00:00Z 450

> SELECT DIFFERENCE(hbas) FROM synop WHERE numer_sta = '07690' AND time >= '2017-04-
06T00:00:00Z' AND time <= '2017-04-06' + 2d
name: synop
time                 difference
----                 ----------
2017-04-06T15:00:00Z 0
2017-04-06T18:00:00Z 450
2017-04-07T06:00:00Z -800
2017-04-07T09:00:00Z 0
2017-04-07T12:00:00Z -200
2017-04-07T15:00:00Z 200
```

Ecart type d'humidité à Nice au mois de février

```
> SELECT STDDEV(u) FROM synop WHERE time > '2017-02-01' AND time < '2017-02-28' AND
numer_sta = '07690'
name: synop
time                          stddev
----                          ------
2017-02-01T00:00:00.000000001Z 10.307574672600488
```

5. Un aperçu de l'administration d'InfluxDB

Quelques éléments pour pouvoir anticiper les modalités d'administration d'un influxDB en conditions réelles.
Ce qui suit est extrait de la documentation et inspiré d'un article paru sur Gnu Linux Magazine France. Dans cet article, les exemples sont ceux de la documentation en ligne InfluxDB. La version traitée était la 0.13... Il va de soi que les exemples suivants sont de notre cru.

5.1. Sauvegarde et restauration

Il est nécessaire de sauvegarder :

- les métadonnées contenues dans le metastore

Commande de sauvegarde du metastore

```
sudo docker exec -it fdu-influxdb mkdir /var/lib/influxdb/backup
sudo docker exec -it fdu-influxdb influxd backup /var/lib/influxdb/backup
```

- toutes les bases

Commandes de sauvegarde des databases

```
sudo docker exec -it fdu-influxdb backup -database test_mesures
/var/lib/influxdb/backup
sudo docker exec -it fdu-influxdb backup -database telegraf /var/lib/influxdb/backup
etc...
```

Pour la restauration :

Commandes de restauration

```
sudo docker exec -it fdu-influxdb influxd restore -metadir /var/lib/influxdb/meta
/var/lib/influxdb/backup
influxd restore -database /var/lib/influxdb/data /var/lib/influxdb/backup
sudo docker exec -it fdu-influxdb chown -R influxdb /var/lib/influxdb
```

puis relance du container.

5.2. Gestion des logs

Sur un binaire classique, les logs sont stockées dans `/var/log/influxdb/influxd.log`.

Dans un container, on les retrouve sur la sortie standard, récupérable si le container est détaché via la commande `docker logs <nom_du_container`.

Il y a 3 catégories de logs par défaut, `[meta]`, `[data]` et `[http]` configurables via fichier de

configuration transmissible au moment du `docker run` pour le container.

5.3. Authentification et autorisations

Toutes les expérimentations contenues dans ce rapport sont réalisées sans authentification.

La synthèse de la documentation est la suivante: la politique d'authentification et d'habilitations ressemble à celle des SGBDR.

Activation

Il faut créer un utilisateur "DB" d'administration, modifier le paramètre **auth-enabled** de la section [http] du fichier de configuration.

Ci-dessous des commandes classiques ressemblantes aux SGBDR ordinaires :

Gestion des utilisateurs

- **CREATE USER**
- **DROP USER**
- **SHOW USERS**

Les habilitations

- **READ**
- **WRITE**
- **ALL**

Gestion des droits

- **GRANT**
- **REVOKE**

5.4. Politique de rétention

InfluxDB propose une fonctionnalité de suppression de la donnée au bout d'un certain temps.

Quand on ne précise pas de politique de rétention à la création du measurement, on récupère la politique par défaut :

```
> SHOW RETENTION POLICIES ON test_mesures
name    duration shardGroupDuration replicaN default
----    -------- ------------------ -------- -------
autogen 0s       168h0m0s           1        true
```

Si duration vaut 0 alors la rétention est infinie.

Création d'une politique de rétention spécifique

```
> CREATE RETENTION POLICY trente_jours ON test_mesures DURATION 30d REPLICATION 1
```

Pour utiliser cette politique, il faut préciser le paramètre **rp** lors de l'insertion des points :

Utilisation de la retention policy "trente_jours"

```
> INSERT INTO trente_jours mesure_station,id_station=tttt,altitude_station=500i
temperature=15,pression=1.0
```

On note que l'on n'a pas précisé de timestamp pour se situer par défaut à la date actuelle.

Par contre, pour obtenir les points insérés, il faudra préciser la politique de rétention :

SELECT sans paramètre RP

```
> select * from mesure_station
name: mesure_station
time                           altitude_station id_station pression temperature
----                           ---------------- ---------- -------- -----------
1970-01-17T10:43:20Z           200i             xxxx       1.2      20
1970-01-17T10:45:00Z           500i             zzzz       1        15
2018-02-03T18:44:00.139560729Z 2000i            yyyy       0.9      30
```

SELECT avec paramètre RP

```
> select * from trente_jours.mesure_station
name: mesure_station
time                           altitude_station id_station pression temperature
----                           ---------------- ---------- -------- -----------
2018-05-08T18:31:40.40178337Z  500i             tttt       1        15
2018-05-08T18:56:48.314248631Z 500i             tttt       1        15
```

5.5. Requêtes en continu ou Continuous request

Cette fonctionnalité permet d'ordonnancer périodiquement une requête d'agrégation pour éventuellement stocker le résultat dans une autre base.
On retrouve le principe du streaming présent dans d'autres "frameworks" ou SGBD NOSQL.

5.6. Clustering

Il est quasiment systématique de faire rimer production avec cluster pour un SGBD.
C'est ce que l'on se propose de découvrir dans cette partie.

Comme dit précédemment, la partie clustering d'InfluxDB n'est disponible que dans sa version commerciale donc payante, **InfluxEntreprise**.

Fort heureusement pour le projet, InfluxData propose la possibilité de tester gratuitement pendant une période limitée (14 jours), son produit complet. InfluxData, nous impose de le faire au travers de son cloud **InfluxCloud**, méthode que l'on décrira ci après.

Mais avant cela, nous allons synthétiser les informations disponibles dans la documentation officielle publique.

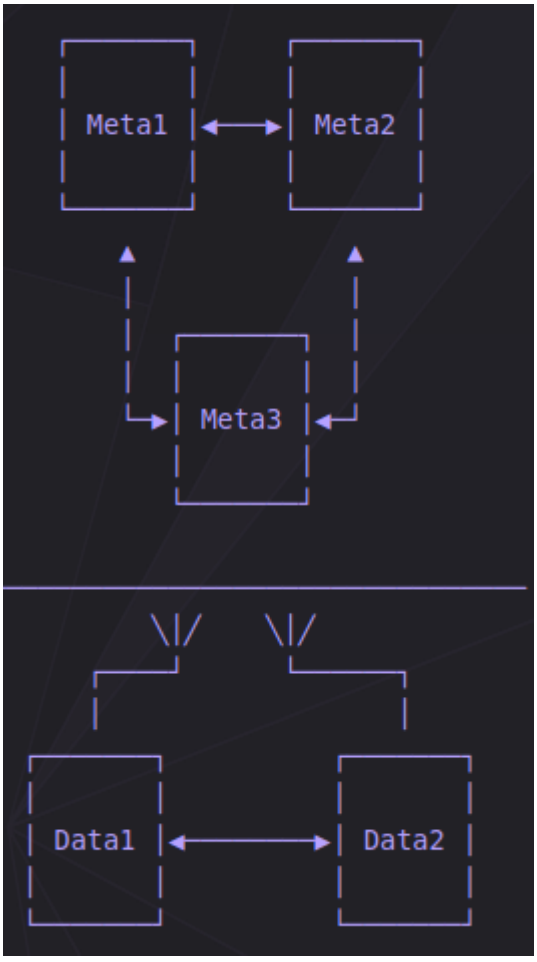
5.6.1. L'architecture d'un cluster InfluxDB

Une installation standard d'un cluster InfluxEntreprise consiste en 3 composants :

- Datanodes
- Metanodes
- Un serveur web, permettant la répartition de la charge

Un cluster influxDB ne requiert vraiment que les deux premiers pour fonctionner. C'est d'ailleurs pour cela que le serveur web est laissé au choix de l'utilisateur. Par contre, à sa charge de le configurer pour gérer une partie fournie dans d'autres SGBD NOSQL, la répartition des requêtes sur les noeuds.

Un schéma de l'architecture du cluster InfluxDB



Les ports du cluster

- entre datanodes (Protobuf) : **8088**
- entre metanodes (Raft) : **8089**

- datanodes et metanodes : **8091**

Les metanodes InfluxdDB

Leur rôle est la gestion du metastore donc de l'ensemble des informations partagées de gestion du cluster :

- les informations relatives aux noeuds du cluster
- les "databases"
- les "retention policies"
- L'index des shard group
- La sécurité ("users", habilitations…)
- Les "continuous queries"
- Les "subscriptions" (Abonnements à destination de consommateurs tels Kapacitor, sujet non approfondi dans ce projet)

Il doit toujours y avoir au minimum 3 metanodes dans un cluster, et dans tous les cas, toujours un nombre impair.

Il n'y a pas forcément de correspondance metanode / serveur mais il est déconseillé par la documentation de faire cohabiter sur un même OS deux metanodes.

Le protocole Raft

Pour gérer l'état du cluster, les metanodes utilisent le *protocole de consensus "Raft"*.
Comme on peut le voir dans cette référence de la documentation officielle, l'objectif de ce protocole est de privilégier La **consistance et le partitionnement** sur la tolérance aux pannes.

Son fonctionnement en ce qui concerne le rôle des metanodes est de les restreindre à 3 :

- suiveur (*follower*)
- candidat (*candidate*)
- dirigeant (*leader*)
  en fonction du vote des autres noeuds.

Le candidat devenu *leader* seul reçoit les requêtes clients, et réplique les "logs entries" dans les machines à état des noeuds du cluster. C'est le second principe clé de Raft. Il est semblable à Paxos en terme de performances.

Le protocole Raft est décrit plus en détails ici :
https://ramcloud.stanford.edu/wiki/download/attachments/11370504/raft.pdf

Un cluster de 3 noeuds peut tolérer une panne. Un cluster de 5 noeuds peut en tolérer deux. Ce sont les deux valeurs avec le meilleur compromis performance/disponibilité recommandées .

Installation du meta node

L'installation est décrite en détails dans la documentation officielle ci-après.
InfluxCloud propose une installation du cluster déjà faite.

Les datanodes influxDB

Datanodes gèrent :

- les measurements
- les tags, clés et valeurs
- les champs, clés et valeurs
- les groupes de shard
- les réponses aux requêtes transmises
- Ils requêtent également des informations (index) auprès du metagroup

Le nombre minimal : 2 datanodes, le nombre recommandé étant un multiple quelconque du facteur de réplication.

Leur protocole de fonctionnement est le *protocole buffers* ou _protobuf de Google décrit plus en détails ici :
https://developers.google.com/protocol-buffers/

TIP	les données se trouvent par défaut dans /var/lib/influxdb/data/<database>/<retention_policy>/<shard_id>

5.6.2. Création d'un cluster InfluxDB dans le cloud InfluxData / AWS

Afin d'obtenir un environnement de tests InfluxEntreeprise, à part s'offrir la licence, il n'y a pas d'autre possibilité que de souscrire, moyennant transmission de ses coordonnées bancaires (...), à l'offre cloud d'InfluxData. L'infrastructure est fournie par le cloud d'Amazon, AWS.

Les caractéristiques du cluster cloud sont décrites en annexe.

Au point de cette étude, et sans moyen de vérifier, on espère pouvoir manipuler les serveurs constituant le cluster pour pouvoir effectuer les tests de failover automatique, répartition des shards etc...

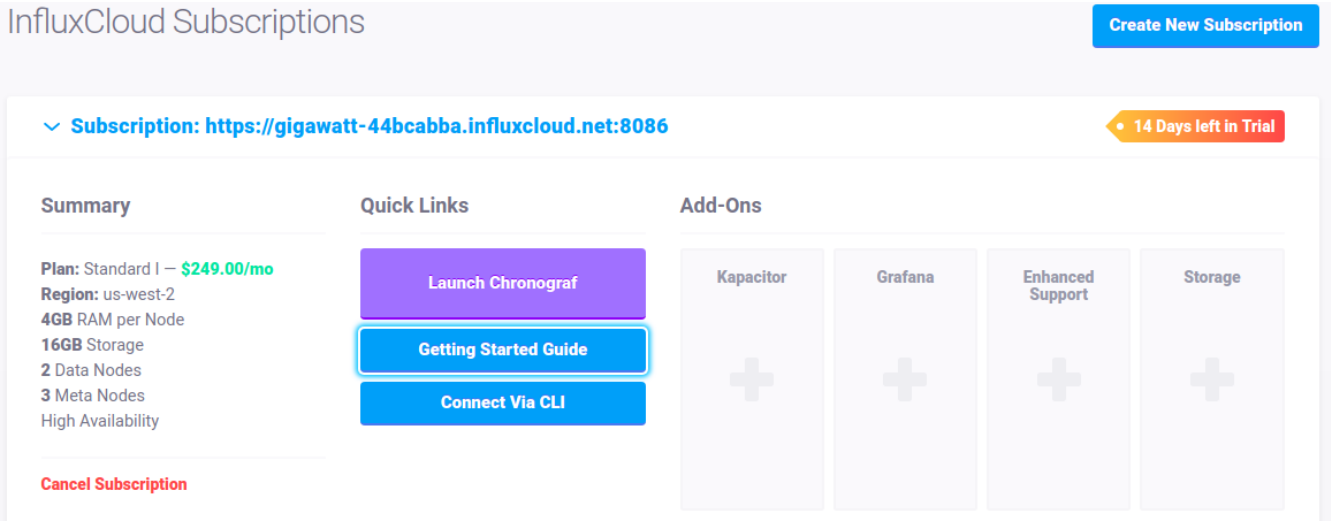


Figure 1. Eléments de configuration du cluster

A la première connexion via le CLI, on se rend compte que ce n'est pas la dernière version (1.5.2)

qui est disponible, mais l'avant dernière (1.5.1). Il est dommage que cela ne figure nulle part, car il est conseillé d'avoir les mêmes niveaux de version entre client et service…

Deuxième connexion de test via CLI influx

```
15:08:07-user@M40474:~/Bureau/CNAM/git/CNAM-projets(master)$ sudo docker run --rm
--name influx_cloud -it influxdb:1.5.1 influx -precision rfc3339 -username
f.dunan@probtp.com -password xxxxxxxxx -host gigawatt-44bcabba.influxcloud.net -ssl
-unsafeSsl -port 8086
Unable to find image 'influxdb:1.5.1' locally
1.5.1: Pulling from library/influxdb
c73ab1c6897b: Pull complete
1ab373b3deae: Pull complete
b542772b4177: Pull complete
2546b96148b0: Pull complete
56b54f39c74f: Pull complete
5fbdf2fbf43e: Pull complete
ef4784e7446b: Pull complete
c9ff778fe79c: Pull complete
Digest: sha256:3facbd59e92e90a9ef8dd646fae64576b6500b7b7ab0fb7cf98d50216a2cdcc2
Status: Downloaded newer image for influxdb:1.5.1
Connected to https://gigawatt-44bcabba.influxcloud.net:8086 version 1.5.1-c1.5.1
InfluxDB shell version: 1.5.1
> SHOW DATABASES
name: databases
name
---
_internal
state_fair_db
meteo
```

Quelques informations supplémentaires

```
> SHOW DIAGNOSTICS
...
name: config
bind-address gossip-frequency hostname
reporting-disabled
------------ ---------------- --------
------------------
:8088        3s               prod-44bcabba-us-west-2-data-4.influxdata.local true
...
name: config-httpd
access-log-path bind-address enabled https-enabled max-connection-limit max-row-limit
--------------- ------------ ------- ------------- -------------------- -------------
                :8086        true    false         0                    0
```

Un autre diagnostic rend le 2e datanode

```
...
name: config
bind-address gossip-frequency hostname
reporting-disabled
------------ ---------------- --------
------------------
:8088        3s               prod-44bcabba-us-west-2-data-3.influxdata.local true
```

5.6.3. InfluxCloud est insuffisant pour se faire une idée du cluster !

Malheureusement, seul le port d'interaction avec InfluxDB est ouvert. Impossible de contacter metanodes et datanodes directement.

La commande suivante, censée donner l'état des shard, échoue :

Tentatives d'utilisation d'influx-ctl, le CLI du cluster

```
00:17:26-user@M40474:~/Bureau/CNAM/git/CNAM-projets(master)$ sudo docker run --rm
--name influx_cloud_ctl -it influxdb:meta influxd-ctl -auth-type basic -user
f.dunan@probtp.com -pwd xxxxxxxx -bind-tls -k -bind gigawatt-
44bcabba.influxcloud.net:8091 copy-shard-status

00:17:56-user@M40474:~/Bureau/CNAM/git/CNAM-projets(master)$ sudo docker run --rm
--name influx_cloud_ctl -it influxdb:meta influxd-ctl -auth-type basic -user
f.dunan@probtp.com -pwd xxxxxxxxx -bind-tls -k -bind gigawatt-
44bcabba.influxcloud.net:8091 show-shards

00:17:56-user@M40474:~/Bureau/CNAM/git/CNAM-projets(master)$ sudo docker run --rm
--name influx_cloud_ctl -it influxdb:meta influxd-ctl -auth-type basic -user
f.dunan@probtp.com -pwd xxxxxxxx -bind-tls -k -bind gigawatt-
44bcabba.influxcloud.net:8091 show
```

La documentation des commandes cluster n'aura hélas pas beaucoup servi.

On ne pourra donc pas "jouer" avec l'extinction des datanodes et metanodes, constater la re-répartition… c'est une déception de taille car lors du choix d'InfluxDB, si le fait de ne pouvoir bénéficier de toutes les fonctionnalités pouvait sembler redhibitoire a proiori, la présence d'un cloud de test de la version compléte avait rendu l'étude d'InfluxDB intéressante…
Comment évaluer le mode de failover automatique d'InfluxDB autrement que sur le papier ?
Et dans une moindre mesure, on a pris le risque d'un achat internet avec ses propres moyens de paiement….

Partitionnement

Sur cet aspect crucial, on devra se contenter d'un aspect documentaire…et il semble que l'on aille de charybde en scylla : D'après la documentation sur le repartitionnement il semblerait que la re-répartition des shard dans un cluster influxdb se fasse manuellement ….

5.6.4. Exploitation du cluster avec l'ensemble des données de l'approfondissement

Pour autant, on va pouvoir bénéficier de la puissance de calcul de la mémoire pour charger l'intégralité des 23 années de mesures météo avec toutes les variables !

Essai avec le même nombre de variables

```
12:38:17-user@M40474:~/Bureau/CNAM/git/CNAM-projets/NFE204(master)$ for i in
donnees/extractions/*;do python3 ../../csv-to-influxdb/csv-to-influxdb.py -d';' -s
'gigawatt-44bcabba.influxcloud.net:8086' -m 'synop' -tc 'date' -tf '%Y%m%d%H%M%S'
--fieldcolumns pmer,tend,dd,ff,td,u,vv,hbas,tnN,txN,rafper,pres,t  --tagcolumns
numer_sta --dbname meteo -u f.dunan@probtp.com -p xxxxxxxx --input $i;done
```

Impact sur les shards

```
name: meteo
id   database retention_policy shard_group start_time           end_time
expiry_time          owners
--   -------- ---------------- ----------- ----------           --------
-----------          ------
5    meteo    autogen          3           1996-01-01T00:00:00Z 1996-01-08T00:00:00Z
1996-01-08T00:00:00Z 4,5
...
1093 meteo    autogen          1091        2017-12-18T00:00:00Z 2017-12-25T00:00:00Z
2017-12-25T00:00:00Z 4,5
1094 meteo    autogen          1092        2017-12-25T00:00:00Z 2018-01-01T00:00:00Z
2018-01-01T00:00:00Z 5,4
```

On note l'ordre des datanodes dans la colonne owners

Commande insertion complète

```
14:46:49-user@M40474:~/Bureau/CNAM/git/CNAM-projets/NFE204(master)$ for i in
donnees/extractions/*;do python3 ../../csv-to-influxdb/csv-to-influxdb.py -d';' -s
'gigawatt-44bcabba.influxcloud.net:8086' -m 'synop' -tc 'date' -tf '%Y%m%d%H%M%S'
--fieldcolumns
pmer,tend,dd,ff,td,u,vv,hbas,tnN,txN,rafper,pres,t,cod_tend,td,vv,ww,w1,w2,n,nbas,cl,c
m,ch,niv_bar,geop,tend24,tn12,tn24,tx12,tx24,tminsol,sw,tw,raf10,rafper,per,etat_sol,h
t_neige,ssfrai,perssfrai,rr1,rr3,rr6,rr12,rr24,phenspe1,phenspe2,phenspe3,phenspe4,nnu
age1,ctype1,hnuage1,nnuage2,ctype2,hnuage2,nnuage3,ctype3,hnuage3,nnuage4,ctype4,hnuag
e4  --tagcolumns numer_sta --dbname meteo -u f.dunan@probtp.com -p xxxxxxxxx --input
$i;done
```

Nombre....et bug !

```
> SELECT * FROM synop LIMIT 5
ERR: %!s(<nil>)
```

La description du bug de requête sur un count trop important que l'on expérimente là.

pourtant :

```
name: meteo
id   database retention_policy shard_group start_time           end_time
expiry_time          owners
--   -------- ---------------- ----------- ----------           --------
-----------          ------
1095 meteo    autogen          1093        1996-01-01T00:00:00Z 1996-01-08T00:00:00Z
1996-01-08T00:00:00Z 5,4


...

2226 meteo    autogen          2224        2017-12-18T00:00:00Z 2017-12-25T00:00:00Z
2017-12-25T00:00:00Z 4,5
2227 meteo    autogen          2225        2017-12-25T00:00:00Z 2018-01-01T00:00:00Z
2018-01-01T00:00:00Z 5,4
```

et

```
> SELECT * FROM synop LIMIT 5
name: synop
time                  ch cl cm cod_tend ctype1 ctype2 ctype3 ctype4 dd  etat_sol ff
geop hbas hnuage1 hnuage2 hnuage3 hnuage4 ht_neige n   nbas niv_bar nnuage1 nnuage2
nnuage3 nnuage4 numer_sta per perssfrai phenspe1 phenspe2 phenspe3 phenspe4 pmer
pres  raf10 rafper rr1 rr12 rr24 rr3 rr6 ssfrai sw t      td     tend tend24 tminsol
tn12   tn24 tw tx12   tx24 u   vv    w1 w2 ww
----                  -- -- -- -------- ------ ------ ------ ------ --  -------- --
---- ---- ------- ------- ------- ------- -------- -   ---- ------- ------- -------
------- ------- --------- --- --------- -------- -------- -------- -------- ----
----  ----- ------ --- ---- ---- --- --- ------ -- -      --     ---- ------ -------
----   ---- -- ----   ---- -   --    -- -- --
1996-01-01T00:00:00Z          1                                  170          2
07005                                                  100030 99090
0             276.04 275.95 100                                          99
1996-01-01T00:00:00Z 11 30 23 4                                  100          2
2250                                          25  1
89642                                                  98820  98290
271.15 261.65 10                  265.65         271.65      44  40000
1996-01-01T00:00:00Z 60 62 61 1                                  0            0
101                                              07015
100020 99380                                0             275.95 275.35 100
96  300   4  4  45
1996-01-01T00:00:00Z 60 36 61 0        7      6                  120          4.1
250  180     1500                             90  7           3       6
07027                                                  99940  99130
0   0             279.04 279.04 0                                        100
1200  5  4  28
1996-01-01T00:00:00Z 60 36 61 8        7                         110          1
70   60                                       100 8           8
07130                                                  99900  99380
0             281.25 280.65 -10                                      96  800   8
6  46
```

On a quand même réussi à insérer l'ensemble des lignes ~ **23 ans x 15000 lignes x 59 colonnes** !

Ci après un exemple de ce que l'on peut faire avec avec le chronograph mis à disposition de l'utilisateur du cloud : Ici l'évolution des températures moyennes mensuelles sur plusieurs années.

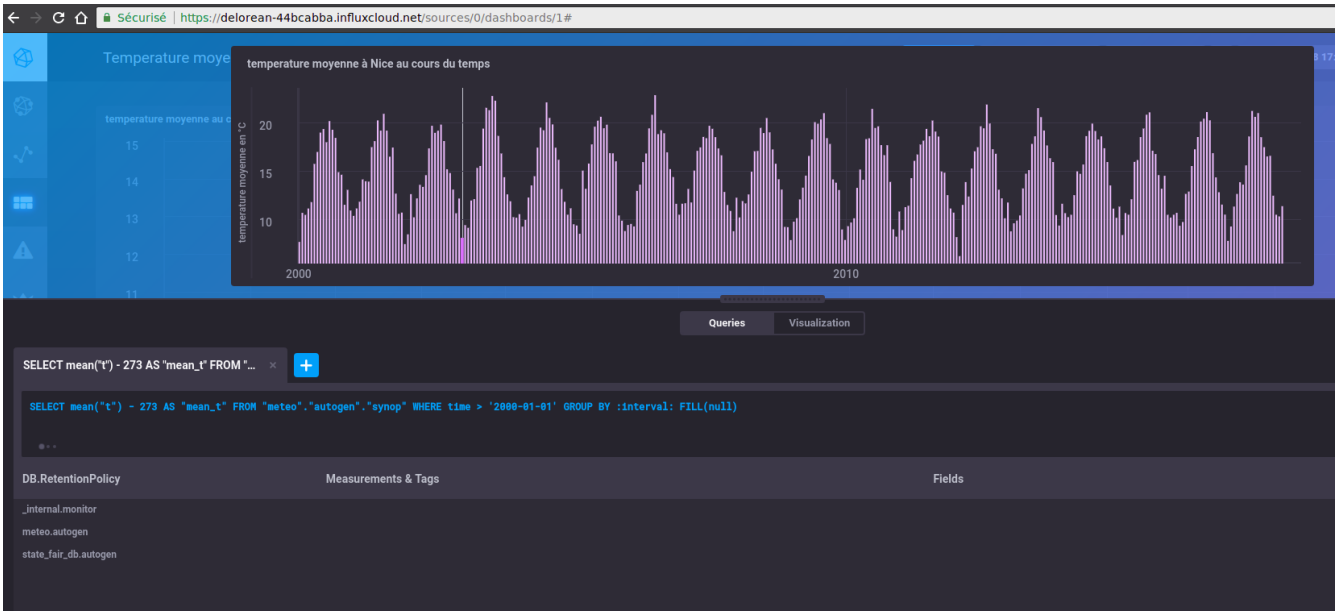


Figure 2. élaboration de la requête dans chronograf

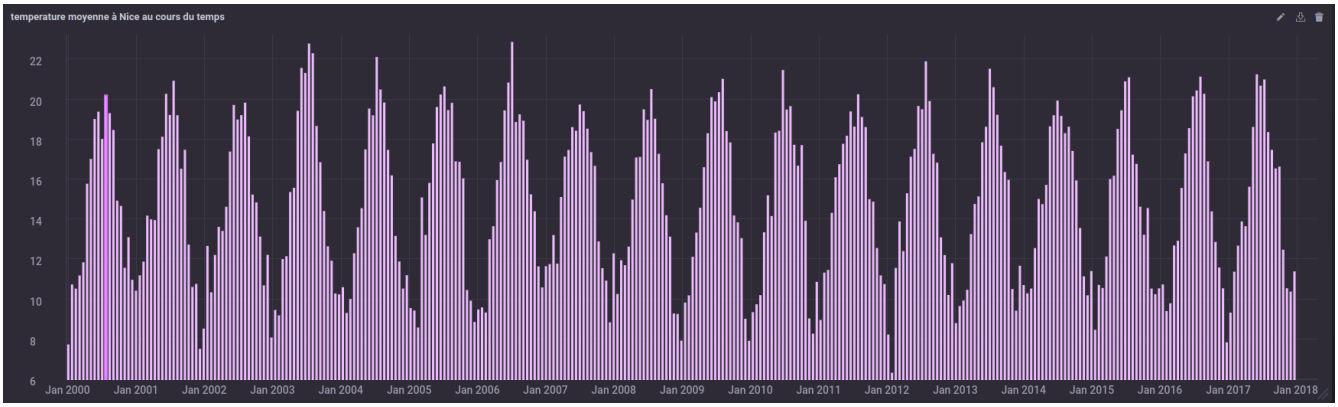


Figure 3. zoom sur la visualisation

6. Conclusion et approfondissements envisageables

A l'usage simple et rapide, ayant un écosystème, un langage de requêtes et une documentation ultra-riches, plébiscité par des utilisateurs, dynamique en terme d'activité, on comprend mieux pourquoi InfluxDB est un leader en terme de TSDB.

Néanmoins, l'appréhension du modèle de données est exigeante et "perd" parfois ses utilisateurs. Certains choix de conception sont aussi discutables. A l'usage, il faut être attentif aux problématiques d'utilisation de la mémoire, surtout si on est limité en cette ressource.

Enfin le fait que le clustering soit payant et qu'il n'y en n'ait qu'un vague aspect est à la fois une des faiblesses de cette étude mais aussi un gros point d'interrogation sur l'efficacité d'InfluxDB en production notamment pour ses administrateurs : Si le re-partitionnement après ajout d'un noeud est manuel, on sera loin des canons d'elasticité automatique du moment dans le domaine du SGBD NoSQL.

On aura durant cette étude, appris les nombreuses notions originales d'InfluxDB....notamment en solutionnant les problémes rencontrés : choisir un tag n'ayant pas une grande cardinalité, faire des projections sur des informations bien choisies...On aura également effleuré le domaines de la mesure physique, générateurs d'un gros volume de données.

Malgré cela, les qualités d'InfluxDB offrent des perspectives de prolongation d'étude. Ci-après quelques points :

- Utiliser les données contenues dans InfluxDB pour être le support d'un modèle de classification ou de prédiction.
- Construire une ou plusieurs variables géographiques afin de proposer des tags supplémentaires et ainsi, obtenir des informations par latitudes ou par régions. Un approche relationnelle classique aurait été de construire une table dédiée, mais l'absence de jointure et le fait que la table n'aurait rien d'une time, justifirait la création d'une colonne de type 'tag'.
- Approfondir les "continuous requests".
- Lors de l'utilisation du cluster, modifier le niveau de consistance des écritures au CLI (consistency <level> sets write consistency level: any, one, quorum, or all) et différencier les timings d'insertion
- Explorer **IFQL**
- Acquerir une licence commerciale pour finir l'évaluation d'un cluster.

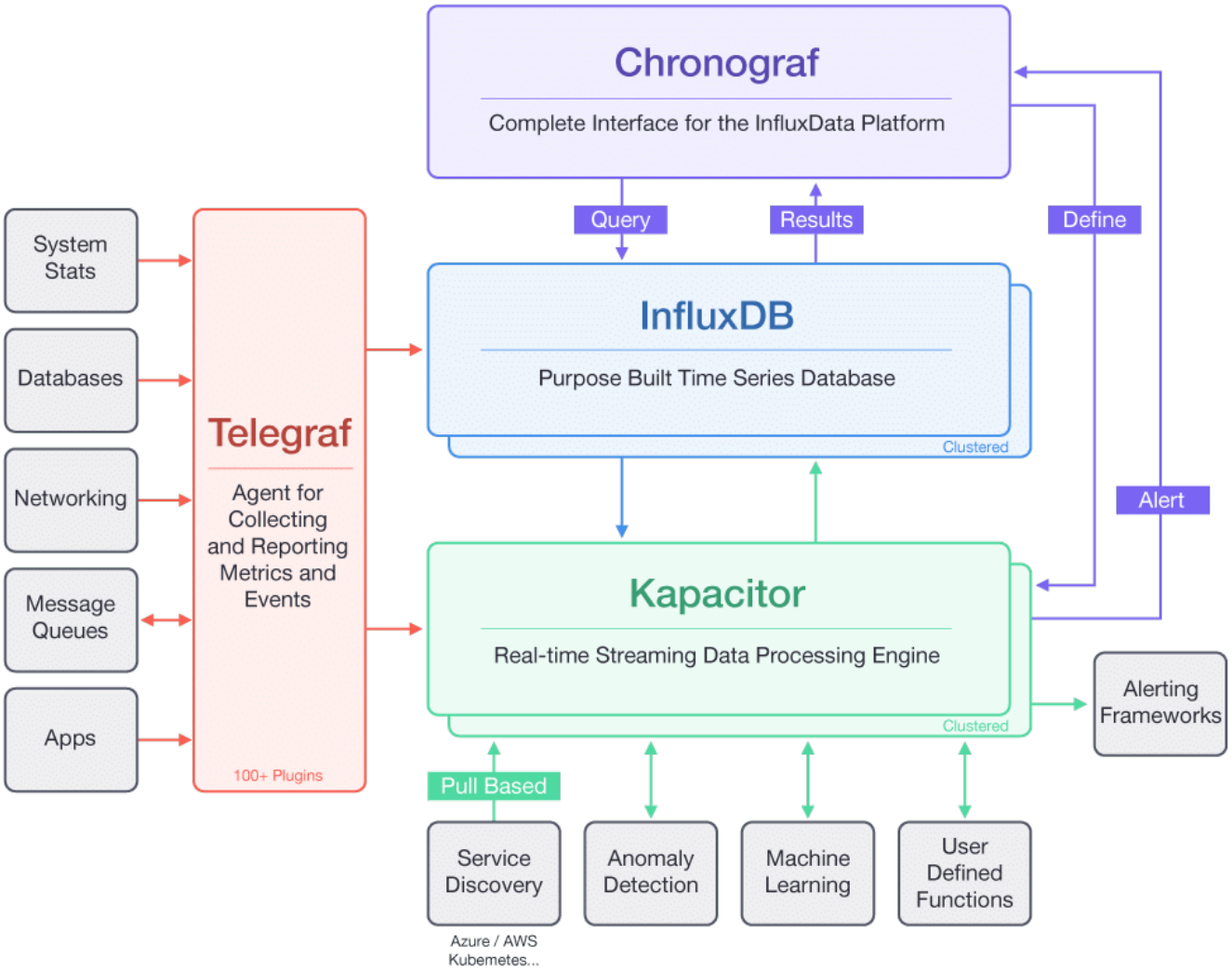
Et sans doute bien d'autres choses encore !

Appendix A: Annexes

A.1. En savoir plus sur la stack TICK

A.1.1. Architecture

Ci-dessous, le schéma des interactions entre InfluxDB et ses compagnons.



A.1.2. Telegraf

C'est l'agent pour collecter les métriques et les insérer dans InfluxDB.

Une configuration de Telegraf

```
[agent]
interval = "1s"
round_interval = true
metric_batch_size = 1000
metric_buffer_limit = 10000
collection_jitter = "0s"
flush_interval = "1s"
flush_jitter = "0s"
```

```
precision = ""
debug = true
quiet = false
#logfile = "/tmp/telegraf.log"

hostname = ""
omit_hostname = false

[[inputs.logparser]]
  ## file(s) to tail:
  files = ["/home/user/Bureau/CNAM/git/CNAM-projets/donnees/meteoFrance/meteo-2017_11-
sample.csv"]
  from_beginning = true
  ## For parsing logstash-style "grok" patterns:
  [inputs.logparser.grok]
    #patterns = ['%{WORD:numer_sta:tag};%{INT:date:ts-
"20060102150405"};%{INT:pmer_Pa};%{INT:tend_Pa};%{INT:cod_tend:drop};%{INT:dd_deg:drop
};%{BASE10NUM:ff_m-
s};%{BASE10NUM:temp_K};%{BASE10NUM:td_K:drop};%{INT:u_%};%{BASE10NUM:vv_m:drop};%{INT:
ww:drop};%{INT:w1:drop};%{INT:w2:drop};%{BASE10NUM:n_%:drop};%{INT:nbas:drop};%{INT:hb
as:drop};%{INT:cl:drop};%{INT:cm:drop};%{INT:ch:drop};%{INT:pres_Pa};%{INT:niv_bar_Pa:
drop};%{INT:geop:drop};%{INT:tend24_Pa};%{BASE10NUM:tnN_K;%{BASE10NUM:txN_K};%{BASE10N
UM:tminsol_K};%{INT:sw_code:drop};%{BASE10NUM:tw_K:drop};%{BASE10NUM:raf10_m-
s};%{BASE10NUM:rafper_m-
s};%{BASE10NUM:per_min};%{INT:etat_sol_code:drop};%{BASE10NUM:ht_neige_m};%{BASE10NUM:
ssfrai_m};%{BASE10NUM:perssfrai_01m};%{BASE10NUM:rrN_mm};%{BASE10NUM:phenspeN:drop};%{
INT:nnuageN_octa:drop};%{INT:ctypeN_code};%{INT:hnuageN_m};%{WORD:drop:drop}']
      patterns = ['%{WORD:numer_sta};%{INT:date:ts-
"20060102150405"};%{INT:pmer_Pa:int};%{INT:tend_Pa:int};%{INT:cod_tend:drop};%{INT:dd_
deg:drop};%{NUMBER:ff_m_s:float};%{NUMBER:temp_K:float};%{NUMBER:td_K:drop};%{INT:u_p1
00:int};%{NUMBER:vv_m:drop};%{INT:ww:drop};%{INT:w1:drop};%{INT:w2:drop};%{NUMBER:n_p1
00:drop};%{INT:nbas:drop};%{INT:hbas:drop};%{INT:cl:drop};%{INT:cm:drop};%{INT:ch:drop
};%{INT:pres_Pa:int};%{INT:niv_bar_Pa:drop};%{INT:geop_code:drop};%{INT:tend24_Pa:int}
;%{NUMBER:tnN_K:float};%{NUMBER:txN_K:float};%{NUMBER:tminsol_K:float};%{INT:sw_code:d
rop};%{NUMBER:tw_K:drop};%{NUMBER:raf10_m_s:float};%{NUMBER:rafper_m_s};%{NUMBER:per_m
in:float};%{INT:etatsol:drop};']

[[outputs.file]]
  ## Files to write to, "stdout" is a specially handled file.
  files = ["stdout"]

#[[outputs.influxdb]]
## The full HTTP or UDP endpoint URL for your InfluxDB instance.
#urls = ["http://localhost:8086"] # required
## The target database for metrics (telegraf will create it if not exists).
#database = "mesures_stations_meteo" # required
## Write timeout (for the InfluxDB client), formatted as a string.
#timeout = "5s"
```

A.1.3. Chronograf

C'est l'outil de visualisation pour les données temporelles dans InfluxDB.

Il permet également de gérer InfluxDB à l'aide d'une interface web.

A.1.4. Kapacitor

Ce client est dédié à la supervision et à la gestion des alertes.

A.2. Tableaux de synthèse des fonctions IQL

Par convention, l'argument 'champs' signifiera un ou plusieurs "field key".

A.2.1. Les fonctions d'agrégation

Les fonctions d'agrégation classiques suivantes ont cours avec InfluxDB :

Nom fonction	Effet
COUNT(champs)	Rend l'effectif
DISTINCT(champs)	Rend les modalités d'un champ
INTEGRAL(champ,time)	Calcule l'aire sous la courbe représentée par les points selectionnés
MEAN(champs)	Calcule la moyenne arithmétique
MEDIAN(champs)	Rend la médiane des champs
MODE(champs)	Rend la valeur la plus fréquente
SPREAD(champs)	Rend la différence entre min et max
STDDEV(champs)	Rend l'écart type
SUM(champs)	Rend la somme des champs

A.2.2. Les fonctions de sélection

Nom fonction	Effet
BOTTOM(champs)	Rend les plus petites valeurs
FIRST(champs)	Rend la valeur avec l'estampille la plus ancienne
LAST(champs)	Rend la valeur avec l'estampille la plus récente
MAX(champs)	Rend la valeur du champ maximum
MIN(champs)	Rend la valeur du champ minimum
PERCENTILE(champs, N)	Rend la valeur du champ supérieure aux N% des valeurs du champ
SAMPLE(champs, N)	Rend un échantillon aléatoire de N valeurs
TOP(champs,N)	Rend les N plus grandes valeurs

A.2.3. Les fonctions de transformation

Elles permettent d'obtenir un résultat entre deux valeurs d'une même mesure.

Nom fonction	Effet
CEILING()	Non implémentée cf documentation.
CUMULATIVE_SUM(champs)	Calcule la somme des champs sur une période.
DERIVATIVE(champs,N)	Rend le taux de variation entre 2 mesures qui se suivent.
DIFFERENCE(champs)	
ELAPSED(champs)	Rend la différence entre 2 timestamps dont les points se suivent
FLOOR()	Non implémentée cf documentation.
HISTOGRAM()	Non implémentée cf documentation. (ne fonctionne pas cf https://github.com/influxdata/influxdb/issues/5930)
MOVING_AVERAGE(champs, N)	Rend la moyenne mobile sur une fenêtre de N points
NON_NEGATIVE_DERIVATIVE(champs,N)	Idem DIFFERENCE mais ne rend que la valeur absolue
NON_NEGATIVE_DIFFERENCE()	Idem DIFFERENCE mais ne rend que la valeur absolue

A.2.4. Prédicteur

La fonction `HOLT_WINTERS()` est disponible pour :

- Prédire quand les valeurs vont passer un seuil donné
- Comparer les données prédites et réelles pour détecter des anomalies.

cf https://docs.influxdata.com/influxdb/v1.5/query_language/functions/#examples-18 pour un exemple, elle n'est pas testée dans cette étude.

A.3. Eléments techniques (Installation, paramétrage...) d'InfluxDB

A.3.1. Matériel de tests

Dans cette étude, on utilisera comme plateforme un laptop Lenovo T430s avec ubuntu 17.10 installé.

A.3.2. InfluxDB et docker

- Installation et démarrage

On dispose de deux méthodes :

Installation du paquet Ubuntu

```
sudo apt-get influxdb
```

Cependant, la version du package ubuntu d'influxDb est obsolète (1.1.1)

On préfèrera la dernière version officielle de l'image docker (**1.5.2**)

```
sudo mkdir  /var/opt/influxdb

docker network create influxdb

sudo docker run -d -p 8086:8086 \
      -v /home/user/Bureau/CNAM/git/CNAM-
projets/NFE204/etc/influxdb.conf:/etc/influxdb/influxdb.conf:ro \
      -v /var/opt/influxdb:/var/lib/influxdb \
      -e INFLUXDB_REPORTING_DISABLED=true \
      --name fdu-influxdb \
      influxdb -config /etc/influxdb/influxdb.conf
```

Pour ajouter un réseau link (deprecated)

```
--net=influxdb \
```

On vérifie que InfluxDB a bien fonctionné par un "ping" REST.

Ping REST

```
user@M40474:~/Bureau/CNAM/git/CNAM-projets/donnees/meteoFrance$ curl -sl -I `sudo
docker inspect -f '{{.NetworkSettings.IPAddress}}' fdu-influxdb`:8086/ping
HTTP/1.1 204 No Content
Content-Type: application/json
Request-Id: 7be68cbe-0853-11e8-860a-000000000000
X-Influxdb-Build: OSS
X-Influxdb-Version: 1.5.2
X-Request-Id: 7be68cbe-0853-11e8-860a-000000000000
Date: Fri, 02 Feb 2018 19:58:48 GMT
```

A.3.3. Logs de démarrage et extinction influxDB

Premier run sans -d

```
[I] 2018-01-30T20:44:25Z InfluxDB starting, version 1.5.2, branch 1.4, commit
6d2685d1738277a1c2672fc58df7994627769be6
[I] 2018-01-30T20:44:25Z Go version go1.9.2, GOMAXPROCS set to 4

 8888888           .d888 888                   8888888b.  888888b.
   888            d88P"  888                   888  "Y88b 888  "88b
   888            888    888                   888    888 888  .88P
   888   88888b.  888888 888 888  888 888  888 888    888 8888888K.
   888   888 "88b 888    888 888  888  Y8bd8P' 888    888 888  "Y88b
   888   888  888 888    888 888  888   X88K   888    888 888    888
   888   888  888 888    888 Y88b 888 .d8""8b. 888  .d88P 888   d88P
 8888888 888  888 888    888  "Y88888 888  888 8888888P"  8888888P"

[I] 2018-01-30T20:44:25Z Using configuration at: /etc/influxdb/influxdb.conf
[I] 2018-01-30T20:44:26Z Using data dir: /var/lib/influxdb/data service=store
[I] 2018-01-30T20:44:26Z opened service service=subscriber
[I] 2018-01-30T20:44:26Z Starting monitor system service=monitor
[I] 2018-01-30T20:44:26Z 'build' registered for diagnostics monitoring service=monitor
[I] 2018-01-30T20:44:26Z 'runtime' registered for diagnostics monitoring
service=monitor
[I] 2018-01-30T20:44:26Z 'network' registered for diagnostics monitoring
service=monitor
[I] 2018-01-30T20:44:26Z 'system' registered for diagnostics monitoring
service=monitor
[I] 2018-01-30T20:44:26Z Starting precreation service with check interval of 10m0s,
advance period of 30m0s service=shard-precreation
[I] 2018-01-30T20:44:26Z Starting snapshot service service=snapshot
[I] 2018-01-30T20:44:26Z Starting continuous query service service=continuous_querier
[I] 2018-01-30T20:44:26Z Starting HTTP service service=httpd
[I] 2018-01-30T20:44:26Z Authentication enabled:false service=httpd
[I] 2018-01-30T20:44:26Z Listening on HTTP:[::]:8086 service=httpd
[I] 2018-01-30T20:44:26Z Starting retention policy enforcement service with check
interval of 30m0s service=retention
[I] 2018-01-30T20:44:26Z Listening for signals
[I] 2018-01-30T20:44:26Z Storing statistics in database '_internal' retention policy
'monitor', at interval 10s service=monitor
```

Shutdown

```
[I] 2018-01-30T20:51:45Z Signal received, initializing clean shutdown...
[I] 2018-01-30T20:51:45Z Waiting for clean shutdown...
[I] 2018-01-30T20:51:45Z snapshot listener closed service=snapshot
[I] 2018-01-30T20:51:45Z shutting down monitor system service=monitor
[I] 2018-01-30T20:51:45Z terminating storage of statistics service=monitor
[I] 2018-01-30T20:51:45Z Precreation service terminating service=shard-precreation
[I] 2018-01-30T20:51:45Z continuous query service terminating
service=continuous_querier
[I] 2018-01-30T20:51:45Z Retention policy enforcement service closing.
service=retention
[I] 2018-01-30T20:51:45Z closed service service=subscriber
[I] 2018-01-30T20:51:45Z server shutdown completed
```

A.3.4. Ports exposés

- **8086** port API HTTP
- *8083* port interface administration, supprimé a partir de la 1.3.0
- *2003* Graphite support, si activé

A.3.5. Processus

Nom du processus et du fichier binaire "serveur" InfluxDB

```
influxd
```

Fichier de configuration

```
/etc/influxd/influxdb.conf
```

Principaux réglages:

- reporting des données à influxData
- répertoire de stockage des données
- répertoire de stockage des meta-données
- configuration du frontal embarqué (nginx)
- plugins de protocoles liés à d'autres outils
  (graphite, collectd, opentsdb, UDP...)

Bref, influxDB est riche...

A.3.6. Bibliothèques python clientes

Les "bindings" python sont disponibles sous forme de paquet ubuntu :

```
apt-get install python-influxdb python3-influxdb
```

A.3.7. CLI & CLI docker

Le client en ligne de commande est présent dans l'installation d'influxdb.

binaire CLI infludb

```
/usr/bin/influx
```

Celui-ci est disponible également sous forme de paquet Ubuntu.

```
user@M40474:~$ apt-get install influxdb-client

user@M40474:~$ influx
Visit https://enterprise.influxdata.com to register for updates, InfluxDB server
management, and monitoring.
Connected to http://localhost:8086 version 1.5.2
InfluxDB shell version: 1.1.1
```

NOTE Le CLI et InfluxDB devraient avoir des versions identiques. Sinon il y a un risque d'erreur d'analyse syntaxique des requêtes, cf documentation sur ce point.

On privilégiera donc l'image docker du CLI influx plus récente.

```
user@M40474:~$ sudo docker run --rm --name influx --link=fdu-influxdb -it influxdb
influx -precision rfc3339 -host `sudo docker inspect -f
'{{.NetworkSettings.IPAddress}}' fdu-influxdb`
Connected to http://172.17.0.2:8086 version 1.5.2
InfluxDB shell version: 1.5.2
```

Portion de log sur le serveur dès la connection du client:

```
[httpd] 172.17.0.4 - - [02/Feb/2018:09:21:39 +0000] "GET /ping HTTP/1.1" 204 0 "-"
"InfluxDBShell/1.5.2" 796914a1-07fa-11e8-8001-000000000000 196
```

TIP Pour le paramétrage de la précision des dates, on utilisera rfc3339 au travers du paramètre -precision.

Paramètres de -précision

```
'rfc3339|h|m|s|ms|u|ns' Specifies the format/precision of the timestamp: rfc3339
(YYYY-MM-DDTHH:MM:SS.nnnnnnnnnZ), h (hours), m (minutes), s (seconds), ms
(milliseconds), u (microseconds), ns (nanoseconds). Precision defaults to nanoseconds.
```

A.3.8. Endpoints REST d'influxDB

Ci-dessous, une synthèse des méthodes ou "endpoints" proposées :

Table 3. "Endpoints" exposés

/debug/requests	Utiliser /debug/requests/ pour tracer les requêtes HTTP aux "endpoints" /write et /query.
/ping	Permet de vérifier le statut de l'instance InfluxDB. Donne également la version.
/query	/query permet d'émettre des requêtes mais aussi de gérer les DB, utilisateurs et politiques de rétention.
/write	/write permet d'insérer et dans une moindre mesure de mettre à jour dans une DB pré-existante.

Inutile de paraphraser la documentation de l'API d'InfluxDB qui est déjà très complète.

A.3.9. Lancer Telegraf

Le besoin étant de faire fonctionner Telegraf comme injecteur à la demande, l'utilisation d'un container est moins pratique qu'un binaire. En effet, les modifications de configuration peuvent être nombreuses.
Le paquet ubuntu étant absent, on procède à une installation du paquet debian fourni par InfluxData.

Téléchargement et installation de telegraf

```
wget https://dl.influxdata.com/telegraf/releases/telegraf_1.5.2-1_amd64.deb
sudo dpkg -i telegraf_1.5.2-1_amd64.deb

service telegraf stop
```

Un fichier de configuration est fourni en annexe.

On notera que dans la configuration du plugin InfluxDB de Telegraf, on peut régler les règles de commit dans un cluster InfluxDB :

```
## Write consistency (clusters only), can be: "any", "one", "quorum", "all"
```

D'ores et déjà, on voit que le client est maître, comme Cassandra, de la politique de consistance de l'écriture.

A.3.10. Lancer Chronograf

Ici, on utilise un container pour obtenir un GUI web issu de l'image docker officielle de Chronograf.

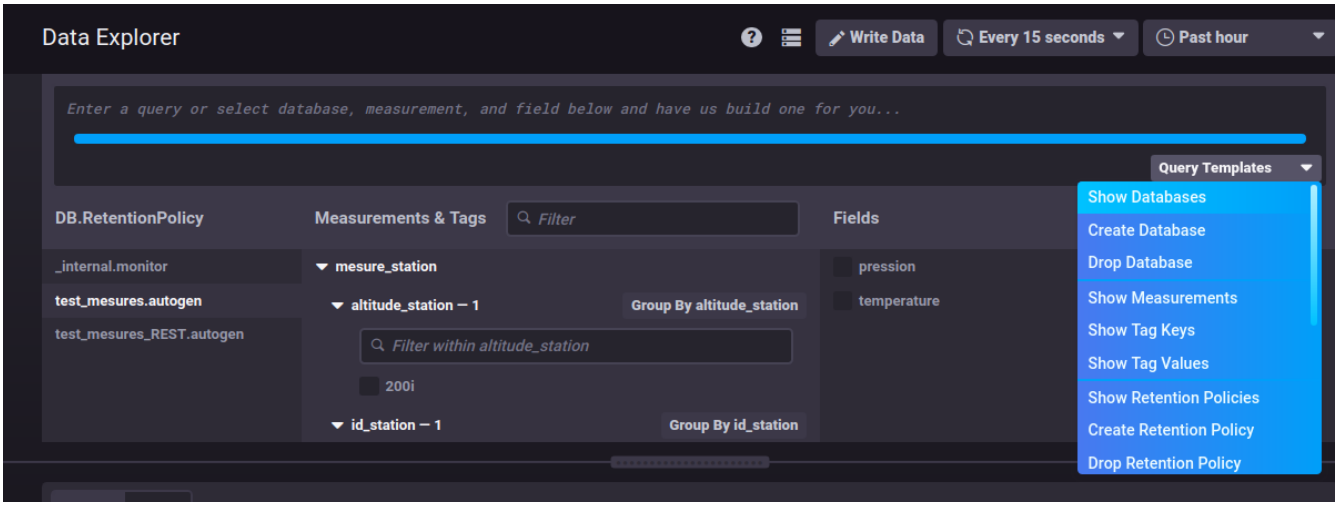
Lancement du container chronograf

```
sudo docker run -p 8888:8888 chronograf \
      --influxdb-url=http://`sudo docker inspect -f '{{.NetworkSettings.IPAddress}}'
fdu-influxdb`:8086
```

L'IHM web est atteignable sur le port 8888 du container via un navigateur.

Il faut d'abord procéder à une brève configuration :
Menu de gauche, item "configuration", "add-source" et renseigner l'URL de connexion. Un exemple : http://172.17.0.2:8086.

Ce qui nous permet, entre autres, d'obtenir à l'aide du menu "Data explorer", la liste des tables et des données.



A.3.11. Lancer Kapacitor

Le monitoring et la gestion d'alerte pour un SGBD NoSQL dépassent le périmètre de l'étude. Pour aborder le sujet, consulter la documentation officielle.

A.4. Description des variables des données météo

METEO FRANCE
Toujours un temps d'avance

Paramètres inclus dans les fichiers de données SYNOP

Descriptif	Mnémonique	type	unité
Indicatif OMM station	numer_sta	car	
Date (UTC)	date	car	AAAAMMDDHHMISS
Pression au niveau mer	pmer	int	Pa
Variation de pression en 3 heures	tend	int	Pa
Type de tendance barométrique	cod_tend	int	code (0200)
Direction du vent moyen 10 mn	dd	int	degré
Vitesse du vent moyen 10 mn	ff	réel	m/s
Température	t	réel	K
Point de rosée	td	réel	K
Humidité	u	int	%
Visibilité horizontale	vv	réel	m
Temps présent	ww	int	code (4677)
Temps passé 1	w1	int	code (4561)
Temps passé 2	w2	int	code (4561)
Nebulosité totale	n	réel	%
Nébulosité des nuages de l'étage inférieur	nbas	int	octa
Hauteur de la base des nuages de l'étage inférieur	hbas	int	mètre
Type des nuages de l'étage inférieur	cl	int	code (0513)
Type des nuages de l'étage moyen	cm	int	code (0515)
Type des nuages de l'étage supérieur	ch	int	code (0509)
Pression station	pres	int	Pa
Niveau barométrique	niv_bar	int	Pa
Géopotentiel	geop	int	m2/s2
Variation de pression en 24 heures	tend24	int	Pa
Température minimale sur N heures	tnN	réel	K
Température maximale sur N heures	txN	réel	K
Température minimale du sol sur 12 heures	tminsol	réel	K
Méthode mesure tw	sw	int	code (3855)
Température du thermomètre mouillé	tw	réel	K
Rafales sur les 10 dernières minutes	raf10	réel	m/s
Rafales sur une période	rafper	réel	m/s
Période de mesure de la rafale	per	réel	min
Etat du sol	etat_sol	int	code (0901)
Hauteur totale de la couche de neige, glace, autre au sol	ht_neige	réel	m
Hauteur de la neige fraîche	ssfrai	réel	m
Periode de mesure de la neige fraiche	perssfrai	réel	1/10 heure
Précipitations dans les N dernières heures	rrN	réel	mm
Phénomène spécial	phenspeN	réel	code (3778)
Nébulosité cche nuageuse N	nnuageN	int	octa
Type nuage N	ctypeN	int	code (0500)
Hauteur de base N	hnuageN	int	m

car : caractère ASCII, int : nombre entier, réel : nombre réel (avec décimale).
Les nombres entre parenthèses après le mot "code" sont les numéros de table de code de l'OMM (Organisation Mondiale de la Météorologie).
Vous pouvez cliquer sur le mot code pour accéder directement a ces tables sur le site de l'OMM

Figure 4. Description des 59 colonnes du fichier

A.5. Correspondances numer_sta et communes des stations de mesure

Table 4. Correspondance entre codes et valeurs de la colonne numer_sta

ID	Nom	Latitude	Longitude	Altitude
07005	ABBEVILLE	50.136000	1.834000	69
07015	LILLE-LESQUIN	50.570000	3.097500	47
07020	PTE DE LA HAGUE	49.725167	-1.939833	6
07027	CAEN-CARPIQUET	49.180000	-0.456167	67
07037	ROUEN-BOOS	49.383000	1.181667	151
07072	REIMS-PRUNAY	49.209667	4.155333	95
07110	BREST-GUIPAVAS	48.444167	-4.412000	94
07117	PLOUMANAC'H	48.825833	-3.473167	55
07130	RENNES-ST JACQUES	48.068833	-1.734000	36
07139	ALENCON	48.445500	0.110167	143
07149	ORLY	48.716833	2.384333	89
07168	TROYES-BARBEREY	48.324667	4.020000	112
07181	NANCY-OCHEY	48.581000	5.959833	336
07190	STRASBOURG-ENTZHEIM	48.549500	7.640333	150
07207	BELLE ILE-LE TALUT	47.294333	-3.218333	34
07222	NANTES-BOUGUENAIS	47.150000	-1.608833	26
07240	TOURS	47.444500	0.727333	108
07255	BOURGES	47.059167	2.359833	161
07280	DIJON-LONGVIC	47.267833	5.088333	219
07299	BALE-MULHOUSE	47.614333	7.510000	263
07314	PTE DE CHASSIRON	46.046833	-1.411500	11
07335	POITIERS-BIARD	46.593833	0.314333	123
07434	LIMOGES-BELLEGARDE	45.861167	1.175000	402
07460	CLERMONT-FD	45.786833	3.149333	331
07471	LE PUY-LOUDES	45.074500	3.764000	833
07481	LYON-ST EXUPERY	45.726500	5.077833	235

ID	Nom	Latitude	Longitude	Altitude
07510	BORDEAUX-MERIGNAC	44.830667	-0.691333	47
07535	GOURDON	44.745000	1.396667	260
07558	MILLAU	44.118500	3.019500	712
07577	MONTELIMAR	44.581167	4.733000	73
07591	EMBRUN	44.565667	6.502333	871
07607	MONT-DE-MARSAN	43.909833	-0.500167	59
07621	TARBES-OSSUN	43.188000	0.000000	360
07627	ST GIRONS	43.005333	1.106833	414
07630	TOULOUSE-BLAGNAC	43.621000	1.378833	151
07643	MONTPELLIER	43.577000	3.963167	2
07650	MARIGNANE	43.437667	5.216000	9
07661	CAP CEPET	43.079333	5.940833	115
07690	NICE	43.648833	7.209000	2
07747	PERPIGNAN	42.737167	2.872833	42
07761	AJACCIO	41.918000	8.792667	5
07790	BASTIA	42.540667	9.485167	10
61968	GLORIEUSES	-11.582667	47.289667	3
61970	JUAN DE NOVA	-17.054667	42.712000	9
61972	EUROPA	-22.344167	40.340667	6
61976	TROMELIN	-15.887667	54.520667	7
61980	GILLOT-AEROPORT	-20.892500	55.528667	8
61996	NOUVELLE AMSTERDAM	-37.795167	77.569167	27
61997	CROZET	-46.432500	51.856667	146
61998	KERGUELEN	-49.352333	70.243333	29
67005	PAMANDZI	-12.805500	45.282833	7
71805	ST-PIERRE	46.766333	-56.179167	21
78890	LA DESIRADE METEO	16.335000	-61.004000	27
78894	ST-BARTHELEMY METEO	17.901500	-62.852167	44
78897	LE RAIZET AERO	16.264000	-61.516333	11
78922	TRINITE-CARAVEL	14.774500	-60.875333	26
78925	LAMENTIN-AERO	14.595333	-60.995667	3

ID	Nom	Latitude	Longitude	Altitude
81401	SAINT LAURENT	5.485500	-54.031667	5
81405	CAYENNE-MATOURY	4.822333	-52.365333	4
81408	SAINT GEORGES	3.890667	-51.804667	6
81415	MARIPASOULA	3.640167	-54.028333	106
89642	DUMONT D'URVILLE	-66.663167	140.001000	43

A.6. Script de récupération des données Météofrance

```
import requests
import time
import gzip
import shutil

#
https://donneespubliques.meteofrance.fr/donnees_libres/Txt/Synop/Archive/synop.201801.
csv.gz
URL_base =
"https://donneespubliques.meteofrance.fr/donnees_libres/Txt/Synop/Archive/synop."
URL_suffixe = ".csv.gz"
OUTPUT_DIR="extractions"
OUTPUT_FILENAME_PREFIX="meteo-"
OUTPUT_FILENAME_EXTENSION=".csv"

for year in range(1996, 2018):
    for month in ["%.2d" % i for i in range(1, 13)]:
        URL_fichier = URL_base + str(year) + str(month) + URL_suffixe
        rep = requests.get(URL_fichier)
        status = rep.status_code
        print("URL <" + rep.url + "> code retour <" + str(status) + ">\n")
        if (status == 200):
            downloaded_filename = OUTPUT_DIR+"/" + OUTPUT_FILENAME_PREFIX + str(year)
+ "_" + str(month) + OUTPUT_FILENAME_EXTENSION
       #     downloaded_filename_gz = downloaded_filename + ".csv.gz"
        with open(downloaded_filename, "wb") as outfile:
            outfile.write(rep.content)
       # with gzip.open(downloaded_filename_gz, 'rb') as f_in,
open(downloaded_filename, 'wb') as f_out:
        #    shutil.copyfileobj(f_in, f_out)
    # http://www.meteofrance.com/robots.txt
    time.sleep(1)
#TODO multithread & temporisation
#TODO temps passe
```

A.7. Scripts import des données MétéoFrance dans InfluxDB

A.7.1. Le script lauréat modifié pour les besoins du projet

```
import requests
import json
import gzip
import argparse
import csv
import datetime

from influxdb import InfluxDBClient

epoch = datetime.datetime.utcfromtimestamp(0)
def unix_time_millis(dt):
    return int((dt - epoch).total_seconds() * 1000)

def loadCsv(inputfilename, servername, user, password, dbname, metric, timecolumn,
timeformat, tagcolumns, fieldcolumns, usegzip, delimiter, batchsize):
    host = servername[0:servername.rfind(':')]
    port = int(servername[servername.rfind(':')+1:])
    #pour le cloud
    #client = InfluxDBClient(host='gigawatt-44bcabba.influxcloud.net', port=8086,
username='f.dunan@probtp.com', password='Tototo11!', database=dbname, ssl=True)
    client = InfluxDBClient(host, port, user, password, dbname)
    #print('Deleting database %s'%dbname)
    #client.drop_database(dbname)
    #print('Creating database %s'%dbname)
    #client.create_database(dbname)
    client.switch_user(user, password)

    # format tags and fields
    if tagcolumns:
        tagcolumns = tagcolumns.split(',')
    if fieldcolumns:
        fieldcolumns = fieldcolumns.split(',')

    # open csv
    datapoints = []
    inputfile = open(inputfilename, 'r')
    count = 0
    with open(inputfilename, 'r') as csvfile:
        reader = csv.DictReader(csvfile, delimiter=delimiter)
        for row in reader:
            timestamp =
unix_time_millis(datetime.datetime.strptime(row[timecolumn],timeformat)) * 1000000 #
in nanoseconds
```

```
            tags = {}
            for t in tagcolumns:
                v = 0
                if t in row:
                    if(row[t] != 'mq'):#FDU
                        v = row[t]
                tags[t] = v

            fields = {}
            for f in fieldcolumns:
                v = 0
                if f in row:
                    if(row[f] != 'mq'):#FDU
                        v = float(row[f])
            #        else:
            #            v = -999999.0
                        fields[f] = v


            point = {"measurement": metric, "time": timestamp, "fields": fields,
"tags": tags}

            datapoints.append(point)
            count+=1

            if len(datapoints) % batchsize == 0:
                print('Read %d lines'%count)
                print('Inserting %d datapoints...'%(len(datapoints)))
                response = client.write_points(datapoints)

                if response == False:
                    print('Problem inserting points, exiting...')
                    exit(1)

                print ("Wrote %d, response: %s" % (len(datapoints), response))


                datapoints = []


    # write rest
    if len(datapoints) > 0:
        print( 'Read %d lines'%count)
        print( 'Inserting %d datapoints...'%(len(datapoints)))
        response = client.write_points(datapoints)

        if response == False:
            print ('Problem inserting points, exiting...')
            exit(1)

        print ("Wrote %d, response: %s" % (len(datapoints), response))
```

```
    print ('Done')

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Csv to influxdb.')

    parser.add_argument('-i', '--input', nargs='?', required=True,
                        help='Input csv file.')

    parser.add_argument('-d', '--delimiter', nargs='?', required=False, default=',',
                        help='Csv delimiter. Default: \',\'.')

    parser.add_argument('-s', '--server', nargs='?', default='localhost:8086',
                        help='Server address. Default: localhost:8086')

    parser.add_argument('-u', '--user', nargs='?', default='root',
                        help='User name.')

    parser.add_argument('-p', '--password', nargs='?', default='root',
                        help='Password.')

    parser.add_argument('--dbname', nargs='?', required=True,
                        help='Database name.')

    parser.add_argument('-m', '--metricname', nargs='?', default='value',
                        help='Metric column name. Default: value')

    parser.add_argument('-tc', '--timecolumn', nargs='?', default='timestamp',
                        help='Timestamp column name. Default: timestamp.')

    parser.add_argument('-tf', '--timeformat', nargs='?', default='%Y-%m-%d %H:%M:%S',
                        help='Timestamp format. Default: \'%%Y-%%m-%%d %%H:%%M:%%S\'
e.g.: 1970-01-01 00:00:00')

    parser.add_argument('--fieldcolumns', nargs='?', default='value',
                        help='List of csv columns to use as fields, separated by
comma, e.g.: value1,value2. Default: value')

    parser.add_argument('--tagcolumns', nargs='?', default='host',
                        help='List of csv columns to use as tags, separated by comma,
e.g.: host,data_center. Default: host')

    parser.add_argument('-g', '--gzip', action='store_true', default=False,
                        help='Compress before sending to influxdb.')

    parser.add_argument('-b', '--batchsize', type=int, default=5000,
                        help='Batch size. Default: 5000.')

    args = parser.parse_args()
    loadCsv(args.input, args.server, args.user, args.password, args.dbname,
        args.metricname, args.timecolumn, args.timeformat, args.tagcolumns,
```

```
        args.fieldcolumns, args.gzip, args.delimiter, args.batchsize)
```

source

https://github.com/escalate/influxdb-csv-importer

prérequis

- installation InfluxDB par pip

modifications

- conversion python3
- gestion des des 'mq'
- désactivation de la création/suppression database automatique
- ajout gestion HTTPS et authentification pour le cloud

A.7.2. Un essai d'import avec un script utilisant docker non retenu car limité

```
Création database
> create database météo
> use météo

Ajout des données dans container
09:19:47-user@M40474:~/Bureau/CNAM/git/influxdb-csv-importer(master)$ sudo docker
build --tag=csvimporter .

Vérification: header est bien pris en compte
10:07:04-user@M40474:~/Bureau/CNAM/git/influxdb-csv-importer(master)$ sudo docker  run
csvimporter --delimiter ';' --server '172.17.0.2' --port 8086 --database meteo
--measurement synop --locale fr_FR.UTF-8 --print-columns --verbose meteo-2017_11-
sample.csv
DEBUG: CSV filename is set to "meteo-2017_11-sample.csv"
DEBUG: CSV delimter is set to ";"
DEBUG: InfluxDB sever address is set to "172.17.0.2"
DEBUG: InfluxDB sever port is set to "8086"
DEBUG: InfluxDB database is set to "meteo"
DEBUG: InfluxDB measurement is set to "synop"
DEBUG: Timestamp format is set to "epoch"
DEBUG: Timestamp timezone is set to "UTC"
DEBUG: Locale for ctype values is set to "('fr_FR', 'UTF-8')"
DEBUG: Locale for numeric values is set to "('fr_FR', 'UTF-8')"
DEBUG: Locale for monetary values is set to "('fr_FR', 'UTF-8')"
DEBUG: Toggle for int to float conversion is set to "True"
[
    "",
    "ch",
    "cl",
    "cm",
```

```
    "cod_tend",
    "ctype1",
    "ctype2",
    "ctype3",
    "ctype4",
    "date",
    "dd",
    "etat_sol",
...
    "w2",
    "ww"
]

sudo docker  run csvimporter --delimiter ';' --server '172.17.0.2' --port 8086
--timestamp-column "date" --timestamp-format "fab_datetime" --database meteo
--measurement synop --locale fr_FR.UTF-8 --column-ignorelist
cod_tend,td,vv,ww,w1,w2,n,nbas,hbas,cl,cm,ch,niv_bar,geop,tend24,tn12,tn24,tx12,tx24,t
minsol,sw,tw,raf10,rafper,per,etat_sol,ht_neige,ssfrai,perssfrai,rr1,rr3,rr6,rr12,rr24
,phenspe1,phenspe2,phenspe3,phenspe4,nnuage1,ctype1,hnuage1,nnuage2,ctype2,hnuage2,nnu
age3,ctype3,hnuage3,nnuage4,ctype4,hnuage4,date --write-data --verbose meteo-2017_11-
sample.csv
```

source

https://github.com/fabio-miranda/csv-to-influxdb

modifications

- contrainte d'import des données dans le container
- modification du format de date

limite

pas de gestion des tags influxDB

A.8. Les caractéristiques du cluster cloud

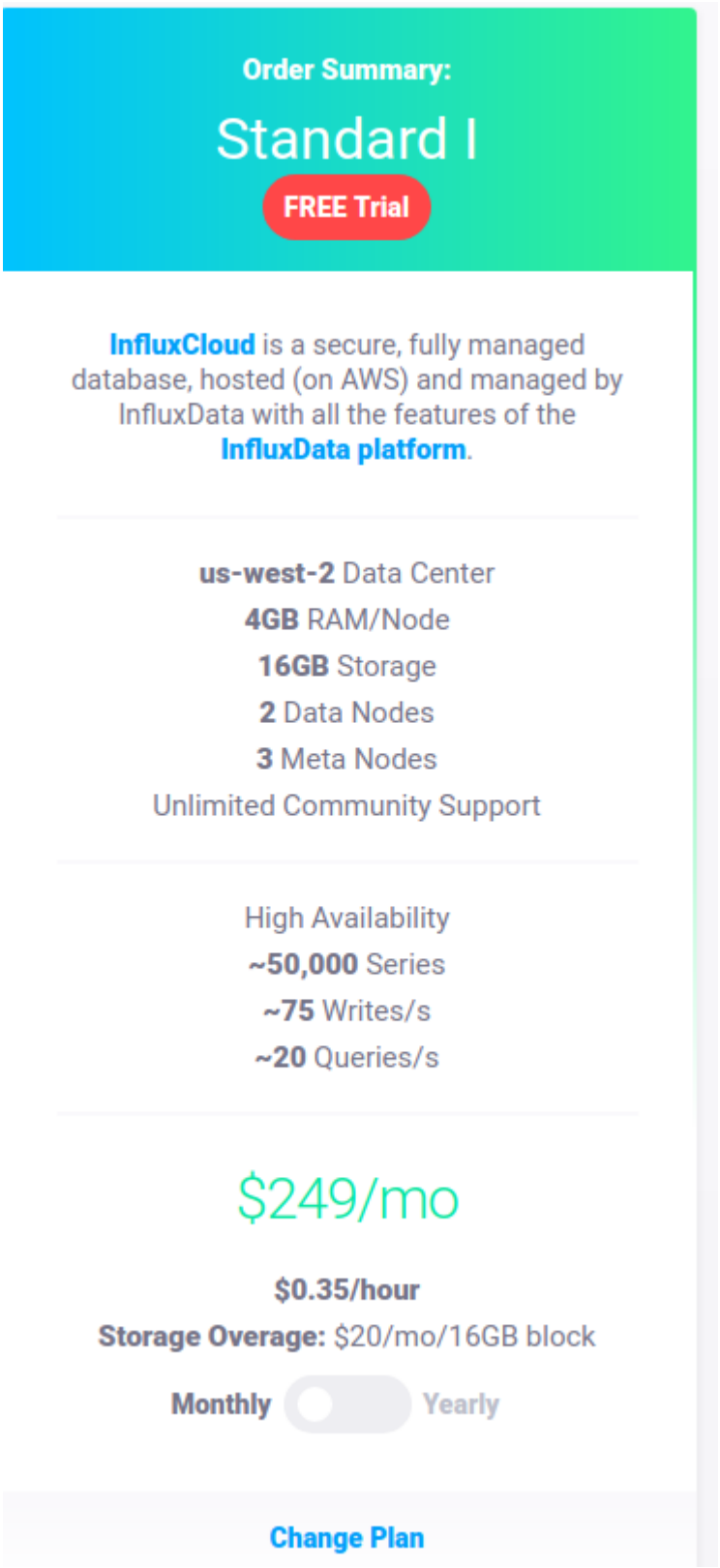


Figure 5. Caractéristiques du cluster cloud

A.9. Des commandes de diagnostic

Commandes

```
influx -execute "show shards" > "shards-$(date +%s).txt"
influx -execute "show stats" > "stats-$(date +%s).txt"
influx -execute "show diagnostics" > "diagnostics-$(date +%s).txt"
curl -o "goroutine-$(date +%s).txt"
"http://localhost:8086/debug/pprof/goroutine?debug=1"
curl -o "heap-$(date +%s).txt" "http://localhost:8086/debug/pprof/heap?debug=1"
```

source : https://github.com/influxdata/influxdb/issues/7810

Références

Bibliographiques

- J.Delamarche, "Le stockage de Series chronologiques avec InfluxDB", GLMF n°198, novembre 2016, ed. diamond.

Web

Générales sur InfluxDB

- Un inventaire de SGBD NOSQL : https://db-engines.com/en/ranking/time+series+dbms.
- Y-combinator : https://en.wikipedia.org/wiki/Y_Combinator_(company)
- Erplane : https://techcrunch.com/2013/03/18/errplane-performance-monitoring-and-alert-service-for-web-apps/
- Limites InfluxDB vue par un site tiers: https://db-engines.com/en/system/InfluxDB
- limites InfluxDB vues par InfluxData : https://docs.influxdata.com/influxdb/v1.2/concepts/insights_tradeoffs/

Données

- le site de l'organisation mondiale de la météorologie : https://public.wmo.int/fr et son extranet :https://www.wmo.int/pages/index_fr.html
- Source des données météoFrance : https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32

Techniques

- Pourquoi InfluxDB n'est pas CRUD : https://docs.influxdata.com/influxdb/v1.4/concepts/crosswalk/#a-note-on-why-influxdb-isn-t-crud
- kapacitor : https://www.influxdata.com/time-series-platform/kapacitor/
- BoltDB : https://docs.influxdata.com/influxdb/v1.4/concepts/storage_engine/#boltdb-and-mmap-b-trees
- IDE : https://github.com/CymaticLabs/InfluxDBStudio
- Paquet Ubuntu InfluxDB : https://packages.ubuntu.com/artful/influxdb

- Image docker officielle influxDB : https://hub.docker.com/_/influxdb/
- Image docker officielle chronograf : https://hub.docker.com/_/chronograf/
- CLI influx : https://docs.influxdata.com/influxdb/v1.4/tools/shell/
- Import batch en CLI : https://docs.influxdata.com/influxdb/v1.4/tools/shell/#import-data-from-a-file-with-import
- L'implémentation Raft d'hashiCorp : https://github.com/hashicorp/raft
- Procédure d'installation des meta nodes : https://docs.influxdata.com/enterprise_influxdb/v1.5/production_installation/meta_node_installation/#step-1-modify-the-etc-hosts-file
- Le select et les tags dans la documentation officielle : https://docs.influxdata.com/influxdb/v1.5/query_language/data_exploration/#common-issues-with-the-select-statement et https://docs.influxdata.com/influxdb/v1.5/troubleshooting/frequently-asked-questions/#tag-keys-in-the-select-clause