Guide utilisateur

Version v1.0, 02.07.2020: First book

Table des matières

Introduction

Ce guide aidera l'utilisateur du data historian dans la compréhension de la plateforme, de son installation, de son utilisation. Le data historian ayant été conçu dans l'optique d'un déploiement sur une infrastructure Big Data, qui sont par nature des environnements complexes à mettre en oeuvre, nous n'aborderons dans ce guide que les déploiements sur une seule machine et de manière progressive. Pour un déploiement sur une infrastructure Big Data multi-serveurs et sa sécurisation, il faudra se référer à d'autres documentations ou contacter Hurence à contact@hurence.com.

Concepts

Hurence Data Historian est une solution open source pour gérer d'énormes quantités de time series (séries temporelles). Il se base sur les moteurs de recherche (comme Apache Solr). Les principaux concepts sont les suivants :

- **Measure** C'est un point dans le temps avec une valeur de type "double" identifiée par un nom et des tags (clés valeurs).
- **Chunk** Est un ensemble continu de Measures dans un intervalle de temps donné, groupées par date, name et éventuellement des tags.

Le but principal de cet outil est d'aider à la création, au stockage et au requêtage de ces chunks de time series. Nous utilison le "chunking" à la place du stockage brut des données afin d'économiser de l'espace de stockage et afin d'améliorer les performances pour de grandes volumétries de points. En effet le chunking nous permet de pré-calculer des agrégations pour faciliter le sampling par exemple.

Modèle de données

Une Measure est un point dans le temps Identifié par les champs suivants :

```
case class Measure(name: String,
                   value: Double,
                   timestamp: Long,
                   year: Int,
                   month: Int,
                   day: String,
                   hour: Int,
                   tags: Map[String,String])
```

Un Chunk est identifié par les champs suivants :

```
case class Chunk(name: String,
     day:String,
     start: Long,
     end: Long,
     chunk: String,
     count: Long,
     avg: Double,
     stddev: Double,
     min: Double,
     max: Double,
     first: Double,
     last: Double,
     sax: String,
     tags: Map[String,String])
```

Comme vous pouvez le constater lors du passage entre la Measure et le Chunk de Measure, le "timestamp" a été remplacé par "start" et "stop" qui définissent l'intervalle de temps, la valeur est, elle, passée d'un double dans le champs "value" à une string en base64 appelée "chunk". Cette string contient l'ensemble des valeurs du chunk, compressées par un algorithme.

Dans solr les chunks sont stockés en format XML sous la forme du schéma suivant :

```
<schema name="default-config" version="1.6">
    ...
    <field name="chunk_avg" type="pdouble" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_value" type="string" multiValued="false" indexed="false"/>
    <field name="chunk_count" type="pint" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_day" type="string" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_end" type="plong" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_first" type="pdouble" multiValued="false" indexed="true"
stored="true"/>
    <field name="chunk_hour" type="pint" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_last" type="pdouble" multiValued="false" indexed="true" stored
="true"/>
    <field name="chunk_max" type="pdouble" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_min" type="pdouble" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_month" type="pint" multiValued="false" indexed="true" stored=
"true"/>
    <field name="name" type="string" multiValued="false" indexed="true" stored="true
"/>
    <field name="chunk_outlier" type="boolean" multiValued="false" indexed="true"
stored="true"/>
    <field name="chunk_sax" type="ngramtext" multiValued="false" indexed="true"
stored="true"/>
    <field name="chunk_start" type="plong" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_stddev" type="pdouble" multiValued="false" indexed="true"
stored="true"/>
    <field name="chunk_sum" type="pdouble" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_trend" type="boolean" multiValued="false" indexed="true"
stored="true"/>
    <field name="chunk_year" type="pint" multiValued="false" indexed="true" stored=
"true"/>
    <field name="chunk_origin" type="string" multiValued="false" indexed="true"
stored="true"/>
    <field name="compactions_running" type="string" multiValued="true" indexed="true"
stored="true"/>
</schema>
```

Installation

Cette section décrit une installation simple du data historian sur un seul serveur. Le Data Historian étant conçu pour le Big Data, il est possible de réaliser une installation avec des possibilités de stockage en très grandes volumétries et avec des performances incomparables sur plusieurs racks de serveurs. Ce type d'installation n'est pas décrite ici par simplification car elle concerne des experts en urbanisation d'infrastructures Big Data.

NOTE	Hurence propose de l'accompagnement pour installer le data historian pour des volumétries importantes.

Dans ce guide nous proposing une installation simple et rapide qui est idéale pour tester, développer et stocker de petits volumes de données peu sensibles. Nous ne recommendons pas cette installation en production car pour la production il faut se préparer pour une infrastructure qui deviendra à terme multi noeuds, et aura la nécessaire redondance des données et des mécanismes de fail over. La seconde installation est plus souhaitable si vous planifiez de faire passer à l'échelle ce que vous voulez installer.

- Installation standalone
- Installation de production mono noeud.

Installation standalone

Cette installation est celle à privilégier si vous êtes nouveau dans l'utilisation du data historian et que vous voulez essentiellement le tester. L'installation n'est pas faite pour la production (ou les noeuds seront redondés) ni pour des larges volumes de données (ce n'est qu'une seule machine). Mais cette installation peut évoluer vers une installation de production si nécessaire.

Note

Hurence propose de l'accompagnement pour faire évoluer vos installations mono noeuds vers une infrastructure clusterisée de production.

Pre-requis pour une installation mono serveur en standalone (pour tester)

Comme expliqué plus haut cette installation est la plus rapide et simple pour installer HDH. Elle est idéale pour un test ou si vous n'avez pas de gros volumes de données.

La configuration minimale du serveur pour une installation de test du data historian est:

- un OS CentOS ou Redhat 7.x
- 16 Gigabits de RAM
- 8 vcores de CPU
- 250 Giga octets de disque
- Java 8

Installation de l'historian

Hurence Data Historian est composé de scripts et fichiers binaires qui permettent de travailler avec les time series et les chunks. Téléchargez le script d'installation de l'historian pour votre version install.sh. Lancez l'installation en lançant ce script :

```
bash ./install.sh
```

Il sera alors demander à l'utilisateur plusieurs informations, comme nous allons procéder à une installation de test en mode standalone vous pouvez laisser les valeurs par défaut et juste taper sur la touche entrée pour chaque question. Pour les questions sur les installations standalone tapez "1". Sauf pour l'installation de spark qui ne nous sera pas utile. Le programme va télécharger et installer des versions embarquées de Solr et Grafana.

Attention cependant au chemin ou vous installez l'historian (par défaut /opt/hdh). En effet l'espace disque doit être suffisant pour stocker les binaires, les scripts ainsi que les données de time series que vous allez injecter.

Voici un aperçu de l'installation :

Dans la suite, on appellera le chemin indiqué pour l'installation de l'historian '$HDH_HOME'.

A l'issue de ce script, si vous avez suivi l'exemple, vous aurez :

- Un serveur solr 8.2 installé dans $HDH_HOME/solr-8.2.0
- Le script à démarrer le serveur solr, vous pouvez vérifier cela à l'adresse suivante : solr UI. Vous pouvez regarder la documentation solr pour interagir avec Solr (notamment pour démarrer le cluster et l'éteindre).
- Un serveur Grafana version-7.0.3 installé dans $HDH_HOME/grafana
- Le plugin datasource Grafana de l'historian a été installé aussi sur ce serveur Grafana. Ce plugin permet la visualisation des time series du data historian dans Grafana.
- Le serveur Grafana a été lancé par le script, vous pouvez vérifier à l'adresse suivante : http://localhost:3000/. Vous pouvez regarder la documentation Grafana pour interagir avec

Grafana (notamment pour démarrer le serveur et l'éteindre).

- Le serveur historian installé dans $HDH_HOME/historian-1.3.5
- Un répertoire "$HDH_HOME/data" qui va contenir les time series de l'historian.

Voici la structure de $HDH_HOME à l'issue de l'installation par défaut : * $HDH_HOME/data : Dossier contenant les données Solr * $HDH_HOME/solr-8.2.0 : Dossier contenant l'installation de Solr 8.2.0 * $HDH_HOME/solr-8.2.0/bin/solr : Script permettant de démarrer et arrêter Solr * $HDH_HOME/historian-1.3.5/bin/historian-server.sh : Permet de lancer et arrêter l'api REST de l'historian. * $HDH_HOME/historian-1.3.5/conf/log4j.properties : Fichier pour contrôler le niveau des logs en mode production (défaut). * $HDH_HOME/historian-1.3.5/conf/log4j-debug.properties : Fichier pour contrôler le niveau des logs en mode debug. * $HDH_HOME/historian-1.3.5/conf/historian-server-conf.json : Le fichier de configuration du serveur fournissant l'api rest de l'historian. * $HDH_HOME/application.log : Le fichier de log du data historian. * $HDH_HOME/grafana-7.0.3 : Dossier contenant l'installation de Grafana * $HDH_HOME/grafana-7.0.3/bin/grafana-server : Script permettant de démarrer et arrêter grafana

Si l'installation se passe correctement, tous les services ont été lancés et le data historian est prêt pour utilisation.

Les commandes suivantes sont données, pour information dans le cas où vous devriez arrêter / redémarrer votre serveur.

Pour démarrer l'historian :

```
$HDH_HOME/historian-1.3.5/bin/historian-server.sh start
```

Pour arrêter l'historian taper la commande suivante :

```
$HDH_HOME/historian-1.3.5/bin/historian-server.sh stop
```

Attention ces commandes n'affectent ni Grafana ni Solr qui sont des services indépendants.

Description du fichier de configuration de l'historian

Voici le fichier de configuration par défaut, il contient toutes les informations possibles, certaines ne sont pas obligatoires :

```
{
  "web.verticles.instance.number": 1,
  "historian.verticles.instance.number": 2,
  "http_server" : {
    "host": "localhost",
    "port" : 8080,
    "historian.address": "historian",
    "debug": false,
    "max_data_points_allowed_for_ExportCsv" : 10000,
    "upload_directory": "/tmp/hurence-historian"
  },
  "historian": {
    "schema_version": "VERSION_0",
    "address" : "historian",
    "limit_number_of_point_before_using_pre_agg" : 50000,
    "limit_number_of_chunks_before_using_solr_partition" : 50000,
    "api": {
      "grafana": {
        "search" : {
          "default_size": 100
        }
      }
    },
    "solr" : {
      "use_zookeeper": true,
      "zookeeper_urls": ["localhost:9983"],
      "zookeeper_chroot" : null,
      "stream_url" : "http://localhost:8983/solr/historian",
      "chunk_collection": "historian",
      "annotation_collection": "annotation",
      "sleep_milli_between_connection_attempt" : 10000,
      "number_of_connection_attempt" : 3,
      "urls" : null,
      "connection_timeout" : 10000,
      "socket_timeout": 60000
    }
  }
}
```

- General conf :
  - web.verticles.instance.number : Le nombre d'instances de verticles à déployer pour répondre aux appels http des clients. Un verticle est capable de gérer un grand nombre de requêtes (au moins 1000, voir la documentation de vertx pour plus d'information).
  - historian.verticles.instance.number : Le nombre d'instances de verticles à déployer pour le service de l'historian qui s'occupe du sampling et des interactions avec le backend. C'est ce paramètre qui va être clé. Il y a de grandes chances que ce soit ce paramètre qu'il faille augmenter en cas de problème de performances.
- Http server conf :

- http_server/host : le nom du serveur http à déployer.
- http_server/port : le port sur lequel est déployé l'api rest.
- http_server/historian.address : le nom du service historian vertx déployé. Ne pas modifier sauf si vous hébergez d'autres services vertx. Il est essentiel de bien connaître vertx pour toucher à ce paramètre.
- http_server/max_data_points_allowed_for_ExportCsv : Ici vous pouvez modifier le maximum de points que l'historian accepte de retourner lorsqu'un client utilise l'api rest d'export dans le format csv. Attention de ne pas choisir un maximum trop grand car il faut que cela tienne en mémoire. Si vous avez besoin d'un gros export il vous faudra utiliser un outil comme spark pour réaliser l'export.
- http_server/upload_directory : Le répertoire ou les fichiers csv seront upload (ils sont effacer une fois l'import de données achever).

- Historian service conf :
  - general conf
    - historian/address : le nom du service historian vertx déployé. Ne pas modifier sauf si vous hébergez d'autres services vertx et que vous maîtrisez vertx. Doit être identique à la valeur de 'http_server/historian.address'.
    - historian/limit_number_of_point_before_using_pre_agg : Une option pour optimiser les performances. Attention à ne pas mettre un nombre trop grand.
    - historian/limit_number_of_chunks_before_using_solr_partition : Une option pour optimiser les performances. Attention à ne pas mettre un nombre trop grand.
    - historian/api/grafana/search/default_size : Une option pour modifier le nombre maximum de nom de métriques à retourner par défaut pour l'endpoint search.
    - historian/schema_version : La version du schéma a utiliser. (Attention ne pas modifier cette valeur manuellement !)
  - solr conf
    - historian/solr/connection_timeout : Le timeout lors de la connection au serveur Solr en millisecondes.
    - historian/solr/socket_timeout : Le timeout pour tous les sockets de lecture avec Solr en millisecondes.
    - historian/solr/stream_url : l'url de la collection solr à utiliser pour l'api stream de solr. Il est recommandé de créer une collection dédiée (avec les ressources suffisantes).
    - historian/solr/chunk_collection : Le nom de la collection ou sont stockées les timeseries.
    - historian/solr/annotation_collection : Le nom de la collection ou sont stockées les annotations.
    - historian/solr/sleep_milli_between_connection_attempt : Le nombre de millisecondes à attendre entre chaque tentatives de ping du serveur solr au démarrage de l'historian.
    - historian/solr/number_of_connection_attempt : Le nombre de tentatives pour tester la connectivité au serveur solr au démarrage de l'historian.
    - historian/solr/use_zookeeper : Si vous utilisez solr cloud (avec un serveur zookeeper ou

sans)

- option si utilisation de zookeeper
  - historian/solr/zookeeper_urls : une liste d'au moins un serveur zookeeper (ex: ["zookeeper1:2181"]).
  - historian/solr/zookeeper_chroot : Le chemin root zookeeper qui contient les données solr. Ne pas renseigner ou utiliser null si il n y a pas de chroot (voir documentation zookeeper).
- option si zookeeper n'est pas utilisé
  - historian/solr/urls : Les urls http pour faire des requêtes à solr. Par exemple ["http://server1:8983/solr", "http://server2:8983/solr"].

Le script d'installation (install.sh) génère un fichier de configuration selon les informations renseignées.

Description des composants installés par le script d'installation

Installation de Apache SolR

Apache SolR est la base de donnée utilisée par l'historian, elle peut être remplacée par un autre moteur de recherche.

Le script d'installation a installé Solr au chemin '$HDH_HOME/solr-8.2.0' que nous appelleront '$SOLR_HOME'. Il a également lancé deux cores Solr localement dans le répertoire `$SOLR_HOME/data`.

Pour démarrer solr

Si vous avez coupé Solr ou bien si vous avez redémarrer votre ordinateur vous pouvez relancer ce Solr avec les commandes suivantes :

```
cd $SOLR_HOME
# démarre un core Solr localement ainsi qu'un serveur zookeeper standalone.
bin/solr start -cloud -s $SOLR_HOME/data/solr/node1 -p 8983
# démarre un second core Solr localement qui va utiliser le serveur zookeeper
précédamment créer.
bin/solr start -cloud -s $SOLR_HOME/data/solr/node2/ -p 7574 -z localhost:9983
```

Vérifiez que votre instance Solr fonctionne correctement en allant sur l'interface graphique à l'adresse suivante : (local solr UI

Installation mono noeud de production

Pre-requis pour une installation mono serveur en production

La configuration minimale du serveur pour une installation mono serveur (en production) du data historian est:

- un OS CentOS ou Redhat 7.x

- 32 Gigabits de RAM
- 32 vcores de CPU
- 2 Tera octets de disque
- Java 8
- Spark 2.3.4
- Solr 8.2.0
- Grafana 7.0.3

Dans cette section nous avons prévu des petits guides pour vous aider a installer solr 8.2.0, spark 2.3.4 et grafana 7.0.3. Mais nous recommandons de vous référer à la documentation officielle de ces outils pour des installations de production. Si vous avez déjà des serveurs existants vous pouvez directement passer à cette section

Installation de Apache SolR

Apache SolR est la base de donnée utilisée par l'historian, elle peut être remplacée par un autre moteur de recherche.

Vous pouvez télécharger la version 8.2.0 de solr sur le lien suivant : solr-8.2.0.tgz ou via le site officiel. Nous vous invitons également à suivre la documentation officielle pour installer un cluster solr de production.

Vérifiez que votre instance solr fonctionne correctement en allant sur l'interface graphique à l'adresse suivante : "http://<solrhost>:8983/solr/#/~cloud"

Install Apache Spark

Pour installer spark vous pouvez téléchargez cette archive : spark-2.3.4-bin-without-hadoop.tgz

Les commandes suivantes vous permettront d'avoir une installation locale. Sinon veuillez suuivre les indications officiels pour un cluster de production.

```
# get Apache Spark 2.3.4 and unpack it
cd $HDH_HOME
wget https://archive.apache.org/dist/spark/spark-2.3.4/spark-2.3.4-bin-without-
hadoop.tgz
tar -xvf spark-2.3.4-bin-without-hadoop.tgz
rm spark-2.3.4-bin-without-hadoop.tgz

# add two additional jars to spark to handle our framework
wget -O spark-solr-3.6.6-shaded.jar
https://search.maven.org/remotecontent?filepath=com/lucidworks/spark/spark-
solr/3.6.6/spark-solr-3.6.6-shaded.jar
mv spark-solr-3.6.6-shaded.jar $HDH_HOME/spark-2.3.4-bin-without-hadoop/jars/
cp $HDH_HOME/historian-1.3.4-SNAPSHOT/lib/loader-1.3.4-SNAPSHOT.jar $HDH_HOME/spark-
2.3.4-bin-without-hadoop/jars/
```

Installation de Grafana

Installez Grafana pour votre plateforme comme décrit ici : https://grafana.com/docs/grafana/latest/installation/requirements/. Une fois l'installation de votre cluster grafana effectuée nous allons passer à l'installation de notre plugin grafana datasource nécessaire pour consulter des dashboard basés sur notre historian.

Il est nécessaire d'avoir au minimum la version 7.0.3 de Grafana.

Installation Plugin Grafana pour utiliser le data historian Hurence

Pour consulter les données de l'historian via des dashboard nous utilisons Grafana. Dans ce but nous avons développé nos propres plugins Grafana que nous ferons évoluer avec l'historian.

Pour installer le plugin datasource suivez les instruction sur le projet correspondant

Installation de l'historian

Hurence Data Historian est composé de scripts et fichiers binaires qui permettent de travailler avec les time series et les chunks. Téléchargez le script dinstallation de l'historian pour votre version install.sh. Lancez l'installation en lançant ce script :

```
bash ./install.sh
```

Il sera alors demander à l'utilisateur plusieurs informations.

Voici un exemple :

Dans la suite, on appellera le chemin indiqué pour l'installation de l'historian '$HDH_HOME'.

A l'issue de ce script, vous aurez :

- l'historian installé au chemin indiqué $HDH_HOME.
- Le plugin datasource grafana de l'historian installé sur le serveur Grafana indiqué lors de l'installation

Voici la structure de $HDH_HOME à l'issue de l'installation par défaut : * $HDH_HOME/bin/historian-server.sh : Permet de lancer et arrêter l'api REST de l'historian. * $HDH_HOME/conf/log4j.properties : Fichier pour contrôler le niveau des logs en mode production (défaut). * $HDH_HOME/conf/log4j-debug.properties : Fichier pour contrôler le niveau des logs en mode debug. * $HDH_HOME/conf/historian-server-conf.json : Le fichier de configuration du serveur fournissant l'api rest de l'historian.

Le script $HDH_HOME/bin/historian-server.sh sert à lancer/arrêter l'api rest de l'historian.

Pour lancer l'historian taper la commande suivante :

```
./bin/historian-server.sh start
```

Pour arrêter l'historian taper la commande suivante :

```
./bin/historian-server.sh stop
```

Attention ces commandes n'affectent ni Grafana ni Solr qui sont des services indépendants.

Description du fichier de configuration de l'historian

Voici le fichier de configuration par défaut, il contient toutes les informations possibles, certaines ne sont pas obligatoires :

```
{
  "web.verticles.instance.number": 1,
  "historian.verticles.instance.number": 2,
  "http_server" : {
    "host": "localhost",
    "port" : 8080,
    "historian.address": "historian",
    "debug": false,
    "max_data_points_allowed_for_ExportCsv" : 10000,
    "upload_directory": "/tmp/hurence-historian"
  },
  "historian": {
    "schema_version": "VERSION_0",
    "address" : "historian",
    "limit_number_of_point_before_using_pre_agg" : 50000,
    "limit_number_of_chunks_before_using_solr_partition" : 50000,
    "api": {
      "grafana": {
        "search" : {
          "default_size": 100
        }
      }
    },
    "solr" : {
      "use_zookeeper": true,
      "zookeeper_urls": ["localhost:9983"],
      "zookeeper_chroot" : null,
      "stream_url" : "http://localhost:8983/solr/historian",
      "chunk_collection": "historian",
      "annotation_collection": "annotation",
      "sleep_milli_between_connection_attempt" : 10000,
      "number_of_connection_attempt" : 3,
      "urls" : null,
      "connection_timeout" : 10000,
      "socket_timeout": 60000
    }
  }
}
```

- General conf :
  - web.verticles.instance.number : Le nombre d'instances de verticles à déployer pour répondre aux appels http des clients. Un verticle est capable de gérer un grand nombre de requêtes (au moins 1000, voir la documentation de vertx pour plus d'information).
  - historian.verticles.instance.number : Le nombre d'instances de verticles à déployer pour le service de l'historian qui s'occupe du sampling et des interactions avec le backend. C'est ce paramètre qui va être clé. Il y a de grandes chances que ce soit ce paramètre qu'il faille augmenter en cas de problème de performances.
- Http server conf :

- http_server/host : le nom du serveur http à déployer.
- http_server/port : le port sur lequel est déployé l'api rest.
- http_server/historian.address : le nom du service historian vertx déployé. Ne pas modifier sauf si vous hébergez d'autres services vertx. Il est essentiel de bien connaître vertx pour toucher à ce paramètre.
- http_server/max_data_points_allowed_for_ExportCsv : Ici vous pouvez modifier le maximum de points que l'historian accepte de retourner lorsqu'un client utilise l'api rest d'export dans le format csv. Attention de ne pas choisir un maximum trop grand car il faut que cela tienne en mémoire. Si vous avez besoin d'un gros export il vous faudra utiliser un outil comme spark pour réaliser l'export.
- http_server/upload_directory : Le répertoire ou les fichiers csv seront upload (ils sont effacer une fois l'import de données achever).

- Historian service conf :
  - general conf
    - historian/address : le nom du service historian vertx déployé. Ne pas modifier sauf si vous hébergez d'autres services vertx et que vous maîtrisez vertx. Doit être identique à la valeur de 'http_server/historian.address'.
    - historian/limit_number_of_point_before_using_pre_agg : Une option pour optimiser les performances. Attention à ne pas mettre un nombre trop grand.
    - historian/limit_number_of_chunks_before_using_solr_partition : Une option pour optimiser les performances. Attention à ne pas mettre un nombre trop grand.
    - historian/api/grafana/search/default_size : Une option pour modifier le nombre maximum de nom de métriques à retourner par défaut pour l'endpoint search.
    - historian/schema_version : La version du schéma a utiliser. (Attention ne pas modifier cette valeur manuellement !)
  - solr conf
    - historian/solr/connection_timeout : Le timeout lors de la connection au serveur Solr en millisecondes.
    - historian/solr/socket_timeout : Le timeout pour tous les sockets de lecture avec Solr en millisecondes.
    - historian/solr/stream_url : l'url de la collection solr à utiliser pour l'api stream de solr. Il est recommandé de créer une collection dédiée (avec les ressources suffisantes).
    - historian/solr/chunk_collection : Le nom de la collection ou sont stockées les timeseries.
    - historian/solr/annotation_collection : Le nom de la collection ou sont stockées les annotations.
    - historian/solr/sleep_milli_between_connection_attempt : Le nombre de millisecondes à attendre entre chaque tentatives de ping du serveur solr au démarrage de l'historian.
    - historian/solr/number_of_connection_attempt : Le nombre de tentatives pour tester la connectivité au serveur solr au démarrage de l'historian.
    - historian/solr/use_zookeeper : Si vous utilisez solr cloud (avec un serveur zookeeper ou

sans)

- option si utilisation de zookeeper
  - historian/solr/zookeeper_urls : une liste d'au moins un serveur zookeeper (ex: ["zookeeper1:2181"]).
  - historian/solr/zookeeper_chroot : Le chemin root zookeeper qui contient les données solr. Ne pas renseigner ou utiliser null si il n y a pas de chroot (voir documentation zookeeper).
- option si zookeeper n'est pas utilisé
  - historian/solr/urls : Les urls http pour faire des requêtes à solr. Par exemple ["http://server1:8983/solr", "http://server2:8983/solr"].

Generation un fichier de configuration pendant le script d'install selon les informations renseignées

Gestion des données

Vous pouvez consulter notre documentation sur l'api REST 'rest-api.pdf' pour connaitre les détails de chaque fonction de l'API (endpoint). Maintenant que nous avons installé l'historian. Vous pouvez jouer avec des données, il y a plusieurs manières d'interagir avec l'historian selon votre culture et vos besoins.

L'historian ne contient initialement aucune données. Il existe plusieurs moyens d'injecter des données (voir le guide sur l'api rest), dans ce guide nous allons utiliser l'import de données à partir de fichiers csv.

Import de donnée à partir de fichiers csv avec l'api REST

TODO

Requêter des données avec l'api REST

```
curl --location --request POST 'http://localhost:8080/api/grafana/query' \
--header 'Content-Type: application/json' \
--data-raw '{
  "panelId": 1,
  "range": {
    "from": "2019-03-01T00:00:00.000Z",
    "to": "2020-03-01T23:59:59.000Z"
  },
  "interval": "30s",
  "intervalMs": 30000,
  "targets": [
    {
      "target": "\"ack\"",
      "type": "timeserie"
    }
  ],
  "format": "json",
  "maxDataPoints": 550
}'
```

Obtenir l'ensemble des noms de métriques

```
curl --location --request POST 'http://localhost:8080/api/grafana/search' \
--header 'Content-Type: application/json' \
--data-raw '{
    "target": "*"
}'

# ["ack", ... ,"messages","cpu"]
```

Obtenir des mesures avec en spécifiant un intervalle de temps

```
curl --location --request POST 'http://localhost:8080/api/grafana/query' \
--header 'Content-Type: application/json' \
--data-raw '{
     "range": {
          "from": "2019-11-25T23:59:59.999Z",
          "to": "2019-11-30T23:59:59.999Z"
      },
      "targets": [
        { "target": "ack" }
      ]
}'
```

Utiliser Spark pour requêter les données

Apache Spark est une plateforme open source pour traiter de grandes quantités de données en parallèle sur un cluster de machines.

Hurence Data Historian est très intégré avec Spark de telle sorte que vous puissiez gérer des intéractions avec les données (en entrées pour en injecter ou en requêtage pour les obtenir sur des critères), et ceci avec une API Spark simple.

Les commandes suivantes vous montrent comment prendre un dataset en CSV depuis HDFS (le système de fichiers de Hadoop) ou depuis un système de fichier local, et le charger dans HDH.

./_setup_historian/spark-2.3.4-bin-hadoop2.7/bin/spark-shell --jars assembly/target/historian-1.3.4-SNAPSHOT/historian-1.3.4-SNAPSHOT//lib/loader-1.3.4-SNAPSHOT.jar,assembly/target/historian-1.3.4-SNAPSHOT/historian-1.3.4-SNAPSHOT/lib/spark-solr-3.6.6-shaded.jar

```
import com.hurence.historian.model.ChunkRecordV0
import com.hurence.historian.spark.ml.Chunkyfier
import com.hurence.historian.spark.sql
import com.hurence.historian.spark.sql.functions._
import com.hurence.historian.spark.sql.reader.{MeasuresReaderType, ReaderFactory}
import com.hurence.historian.spark.sql.writer.{WriterFactory, WriterType}
import com.lucidworks.spark.util.SolrSupport
import org.apache.commons.cli.{DefaultParser, Option, Options}
import org.apache.spark.sql.SparkSession
import org.slf4j.LoggerFactory

 val filePath =
"/Users/tom/Documents/workspace/historian/loader/src/test/resources/it-data-
4metrics.csv.gz"
 val reader = ReaderFactory.getMeasuresReader(MeasuresReaderType.GENERIC_CSV)
    val measuresDS = reader.read(sql.Options(
      filePath,
      Map(
        "inferSchema" -> "true",
        "delimiter" -> ",",
        "header" -> "true",
        "nameField" -> "metric_name",
        "timestampField" -> "timestamp",
        "timestampDateFormat" -> "s",
        "valueField" -> "value",
        "tagsFields" -> "metric_id,warn,crit"
      )))



val chunkyfier = new Chunkyfier().setGroupByCols(Array(  "name", "tags.metric_id"))
val chunksDS = chunkyfier.transform(measuresDS).as[ChunkRecordV0]


val writer = WriterFactory.getChunksWriter(WriterType.SOLR)
    writer.write(sql.Options("historian", Map(
      "zkhost" -> "localhost:9983",
      "collection" -> "historian",
      "tag_names" -> "metric_id,warn,crit"
    )), chunksDS)
```

L’injection temps réel avec LogIsland

Le logiciel Open Source de Hurence permettant de faire du stream processing (donc du traitement temps réel de données) permet bien sûr d’injecter des données "à la volée", notamment celles que

vous aurez pu "pousser" dans un bus de message Mqtt ou Kafka.

Hurence Data Historian est donc capable de traiter des données de capteurs et de les stocker et grapher en temps réel.

Pour mettre en place une chaîne d’injection temps réel, le mieux est de contacter Hurence pour un petit accompagnement car cela devient du vrai Big Data temps réel.

Visualisation des données

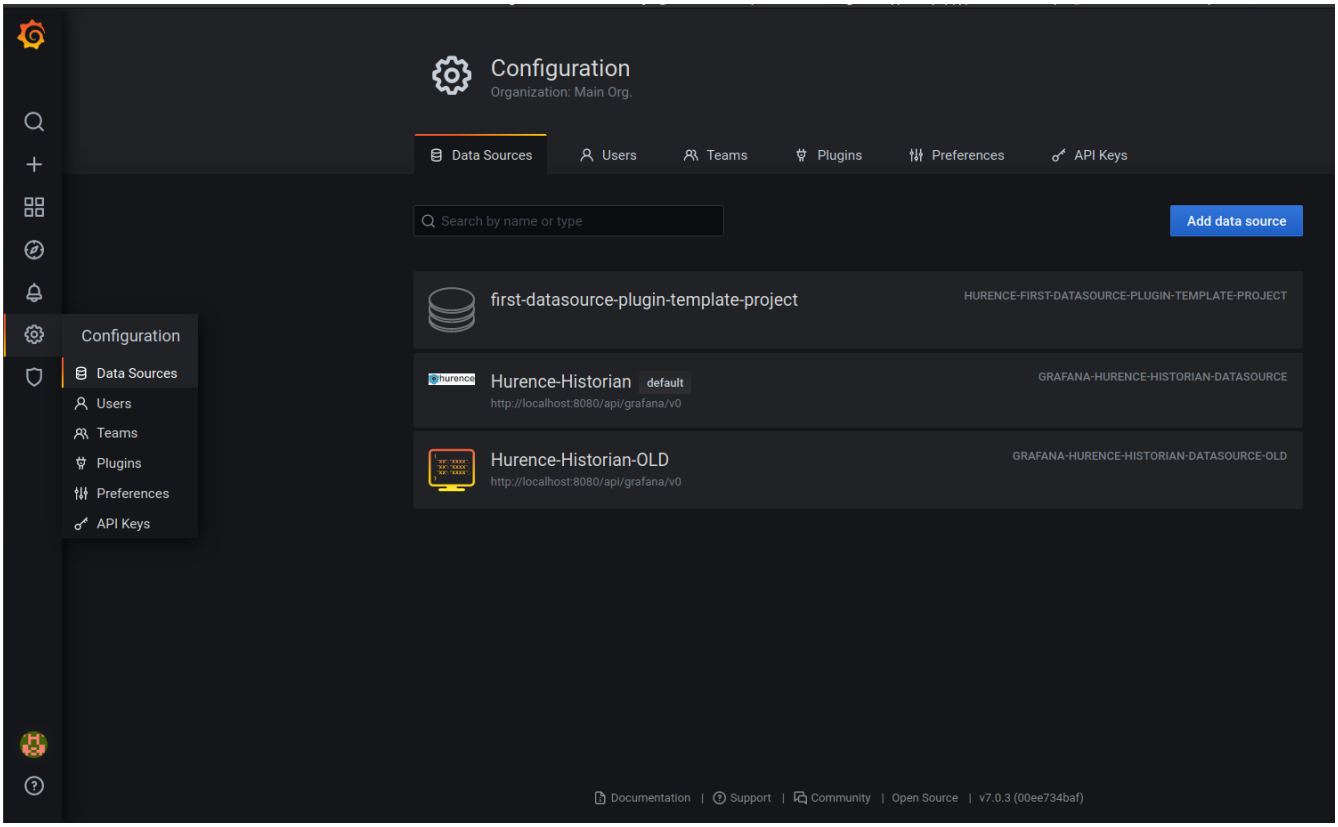
HDH fournit donc un plugin Grafana pour visualiser les données sous formes de graphes.

La manipulation des données depuis les data sources de votre data historian se fait comme pour les autres sources. Vous fournissez votre requête, un intervalle de temps si nécessaire, et un algorithme de sampling si vous requêtez sur une grande quantité de points (pour que la courbe soit bien lisible).

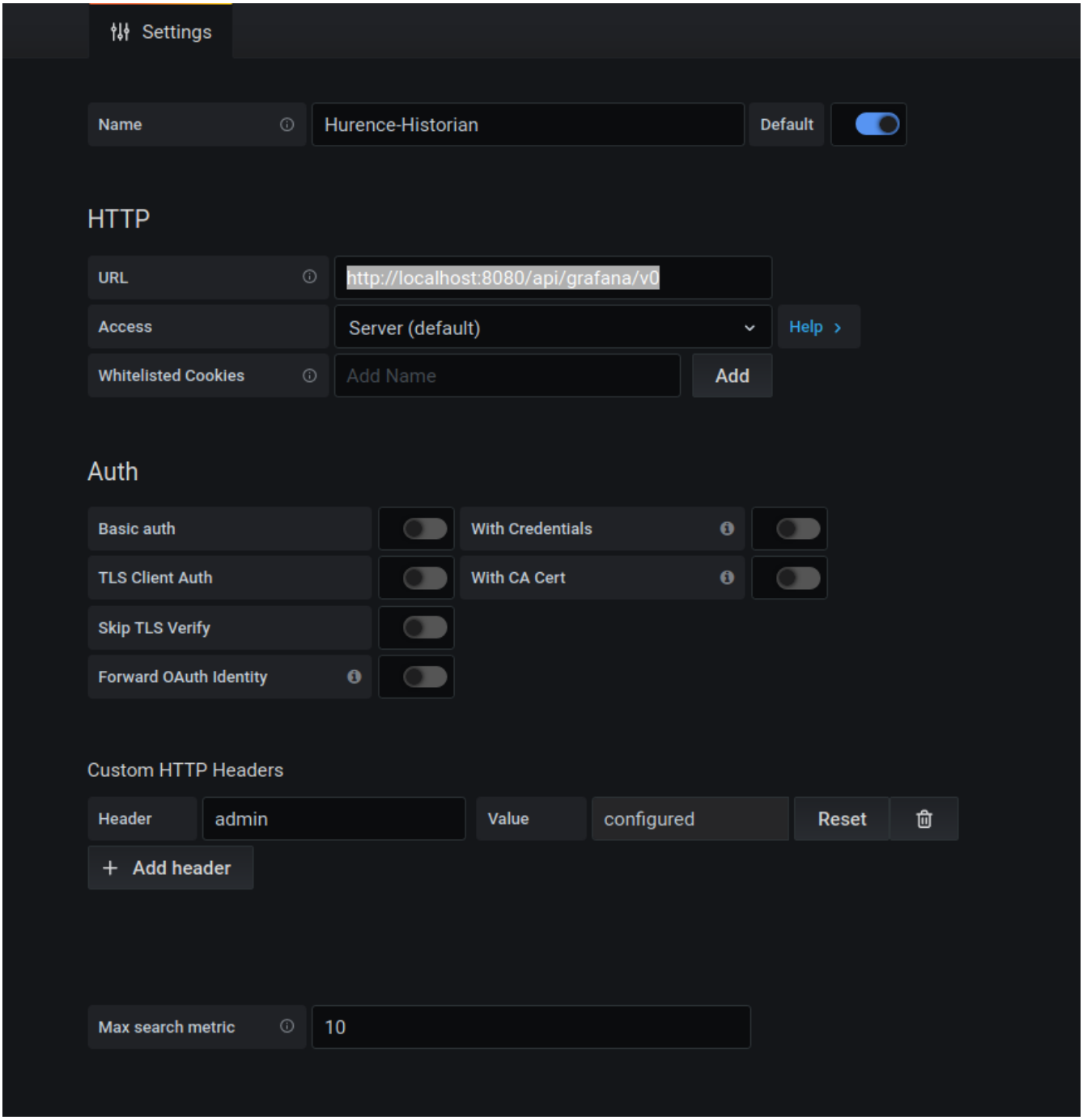
Configurer Grafana et vos dashboard

Pour créér des dashboard il faut aller sur l'interface graphique de grafana (http://localhost:3000/ (localhost étant à remplacer par un nom de machine si vous n'êtes pas en installation standalone).

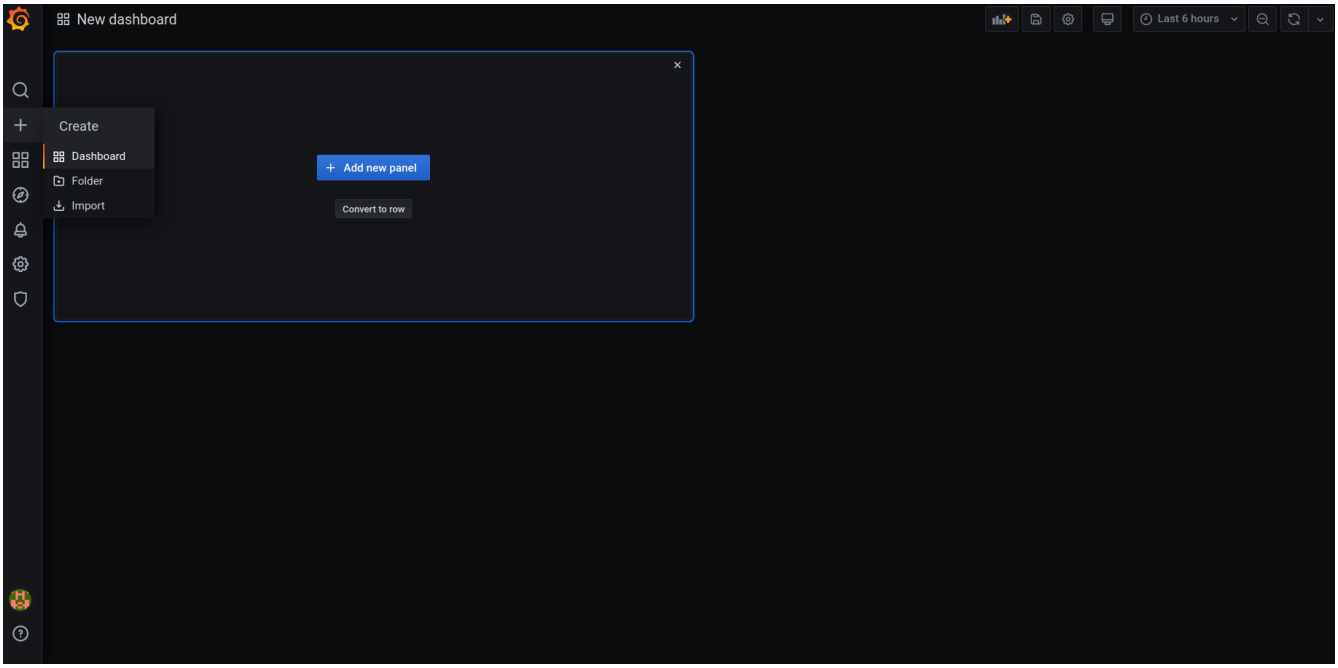
Allez ensuite dans le menu des datasources :



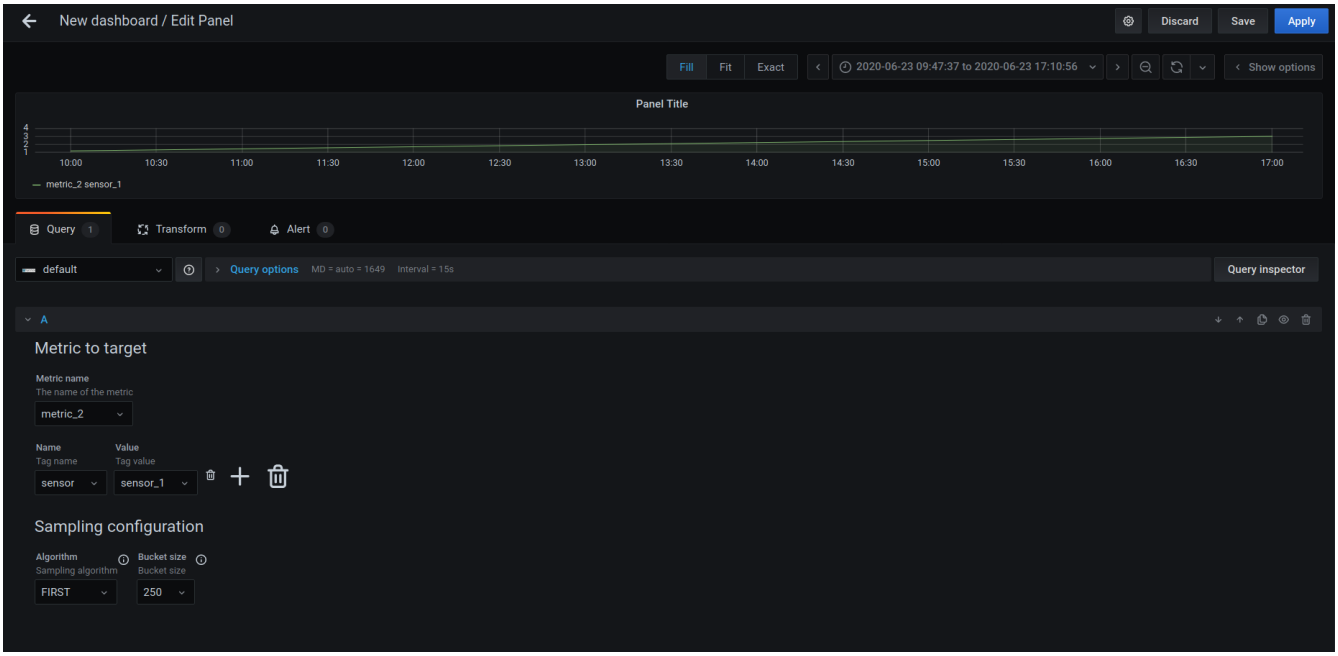
Cherchez la datasource "Hurence-Historian" et sélectionnez la. Il y a juste a renseigner l'URL http://localhost:8080/api/grafana/v0 dans le cas de l'installation standalone.



Testez la connectivité en cliquant sur "Save & Test" button at the bottom of the page. Ensuite créez un nouveau dashboard.



Ajoutez les graphes que vous voulez, un petit exemple ci-dessous :



Une "query" (requête) consiste à renseigner un nom de métrique, et des paires de tag name/tag value ainsi que des options pour l'algorithme de sampling.

note

les options de sampling de la première requête sont utilisées pour toutes les autres.

note

si vous n'avez pas de données d'injectées, suivez le tutorial pour injecter des données.

note

si vous n'avez pas ajouté de tag name à l'installation vous ne pourrez pas utiliser de tags. Il est toujours possible de rajouter des tags manuellement après installation en rajoutant un champ dans le schéma.

Stopper et détruire les données

Cette section vous apprend à stopper les services et à détruire si nécessaire les données (après avoir testé vous voudrez alimenter avec vos données réelles).

Stopper ses instances SolR

Cette section vous apprend à stopper les intances SolR. Attention cette commande va éteindre Solr (cela pourrait impacter d'autres service utilisant Solr).

```
cd $SOLR_HOME
bin/solr stop -all
```