Manuel utilisateur

(C)Hurence

Version v1.0, 18.02.2020: First draft

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 des énormes quantités de time series. 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" identifié par un nom et des tags (clés valeurs).
- **Chunk** Est un ensemble continue de Measures dans un intervalle de temps données groupé 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'epace dans le disque dur et d'améliorer les performances pour de grande volumétries de points. En effet le chunking nous permet de pré-calculer des agrégations pour faciliter le sampling par exemple.

Data modèle

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 elle passé d'un double dans le champs "value" à une string en base64 compresser avec un algorithm stocké dans le champs "chunk".

Dans solr les chunks seront stockés 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.

- Installation standalone(à utiliser seulement pour tester)
- Installation de production mono noeud. Cette installation se base sur des serveurs solr, grafana et spark (spark et grafana étant optionnels) installés de manières indépendantes. Ce guide contient des sections pour installer ces outils.

Installation standalone

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

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

- un OS CentOS ou Redhat 7.x
- 8 Gigabits de RAM
- 6 vcores de CPU
- 50 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 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, 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 tapé "1". Sauf pour l'installation de spark qui ne nous sera pas utile. Le programme va télécharger et installer des versions embarqués 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 installer sur ce serveur grafana.
- Le serveur grafana a du être lancé par le script, vous pouvez vérifier à l'adresse suivante : http://localhost:3000/. Vous pouvez regarder la documentation solr 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 a 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 sont lancé à la fin de cette installation.

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 obligatoire :

```
{
  "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,
  },
  "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 appelles http des clients. Un verticle est capable de gérer un grand nombre de requête (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 grande chance 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 lequelle 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 et que vous savez ce que vous faites
    - 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 utiliser 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.
- 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 savez ce que vous faites. 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 métrique 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 socket de lecture avec Solr en millisecondes.
        - historian/solr/stream_url : l'url de la collection solr a utiliser pour l'api stream de solr. Il est recommandé de créer une collection dédié (avec les ressources suffisantes).
        - historian/solr/chunk_collection : Le nom de la collection ou sont stockés les timeseries.
        - historian/solr/annotation_collection : Le nom de la collection ou sont stockés 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 : If you use solr cloud (using a zookeeper server) or not.
            - 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 renseigné 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'install génère un fichier de configuration selon les informations renseignées.

Description des composants installer par le script d'installation

Installation de Apache SolR

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

Le script d'installation a installer 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 couper solr ou bien si vous avez rédémarrer votre ordinateur vous pouvez relancer le 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 officiel 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 utilisé 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 a suivre la documentation officiel 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 effectué nous allons passer à l'installation de notre plugin grafana datasource nécessaire pour consulter des

dashboard basé sur notre historian.

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

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 développer 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 installer 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 a 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 obligatoire :

```
{
  "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,
  },
  "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 appelles http des clients. Un verticle est capable de gérer un grand nombre de requête (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 grande chance 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 lequelle 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 et que vous savez ce que vous faites
- 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 utiliser 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.

- 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 savez ce que vous faites. 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 métrique 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 socket de lecture avec Solr en millisecondes.
    - historian/solr/stream_url : l'url de la collection solr a utiliser pour l'api stream de solr. Il est recommandé de créer une collection dédié (avec les ressources suffisantes).
    - historian/solr/chunk_collection : Le nom de la collection ou sont stockés les timeseries.
    - historian/solr/annotation_collection : Le nom de la collection ou sont stockés 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 : If you use solr cloud (using a zookeeper server) or not.
      - option si utilisation de zookeeper
        - historian/solr/zookeeper_urls : une liste d'au moins un serveur zookeeper (ex: ["zookeeper1:2181"]).

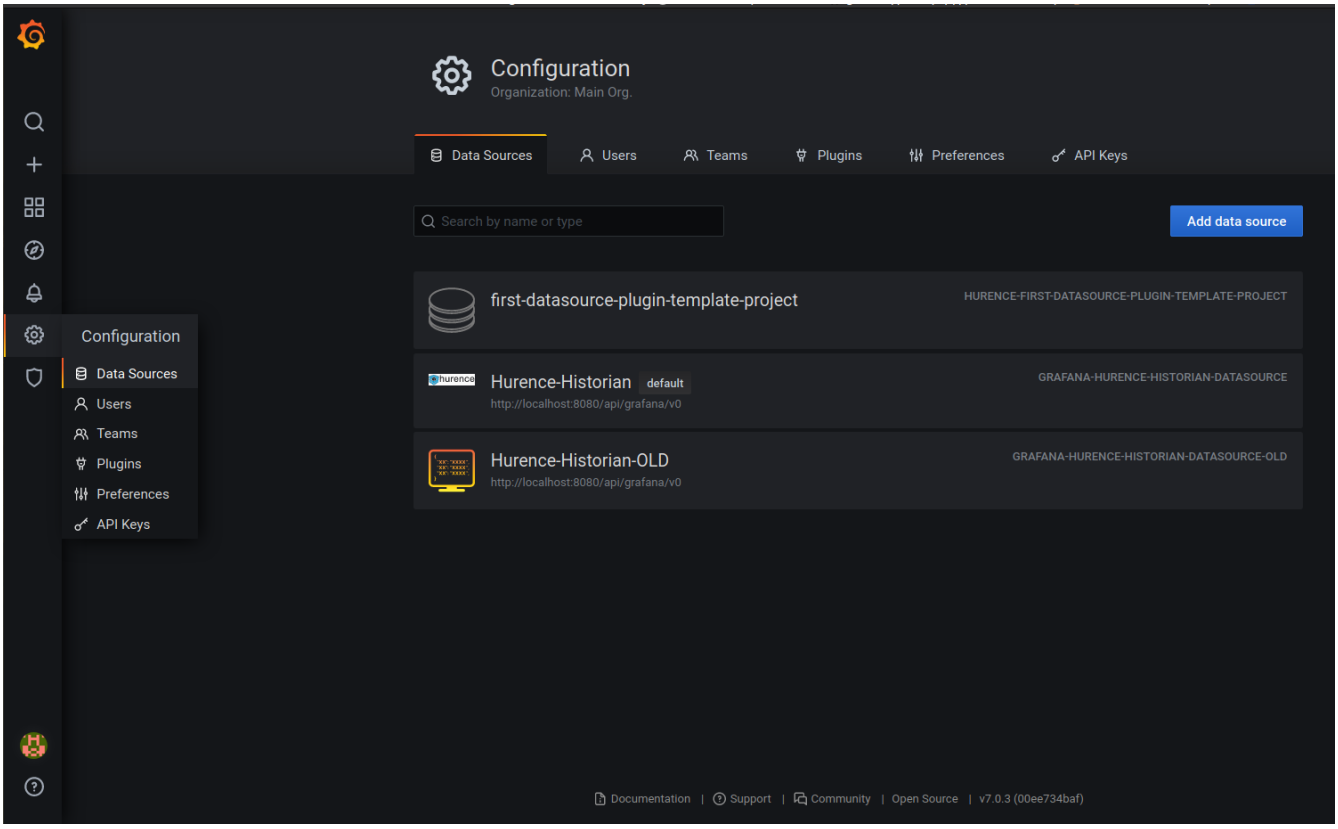
        - historian/solr/zookeeper_chroot : Le chemin root zookeeper qui contient les données solr. Ne pas renseigné ou utiliser null si il n y a pas de chroot (voir documentation zookeeper).
    - option si zookeeper n'est pas utilisé
        - historian/solr/urls : Les urls http pour faire des requêtes à solr. Par exemple ["http://server1:8983/solr", "http://server2:8983/solr"].

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

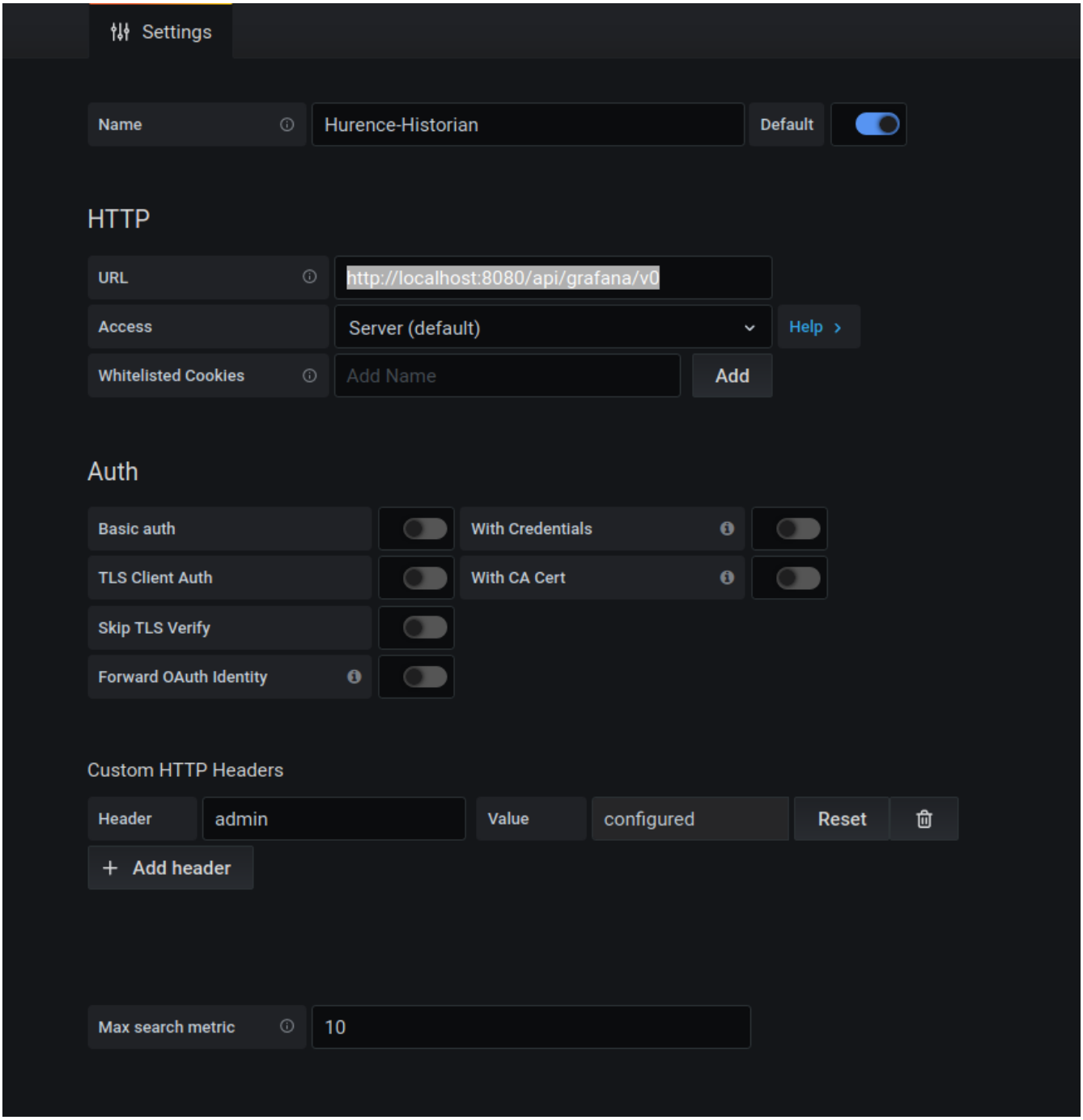
Configure grafana and your dashboard

aller sur l'interface graphique de grafana (http://localhost:3000/ pour une isntallation standalone).

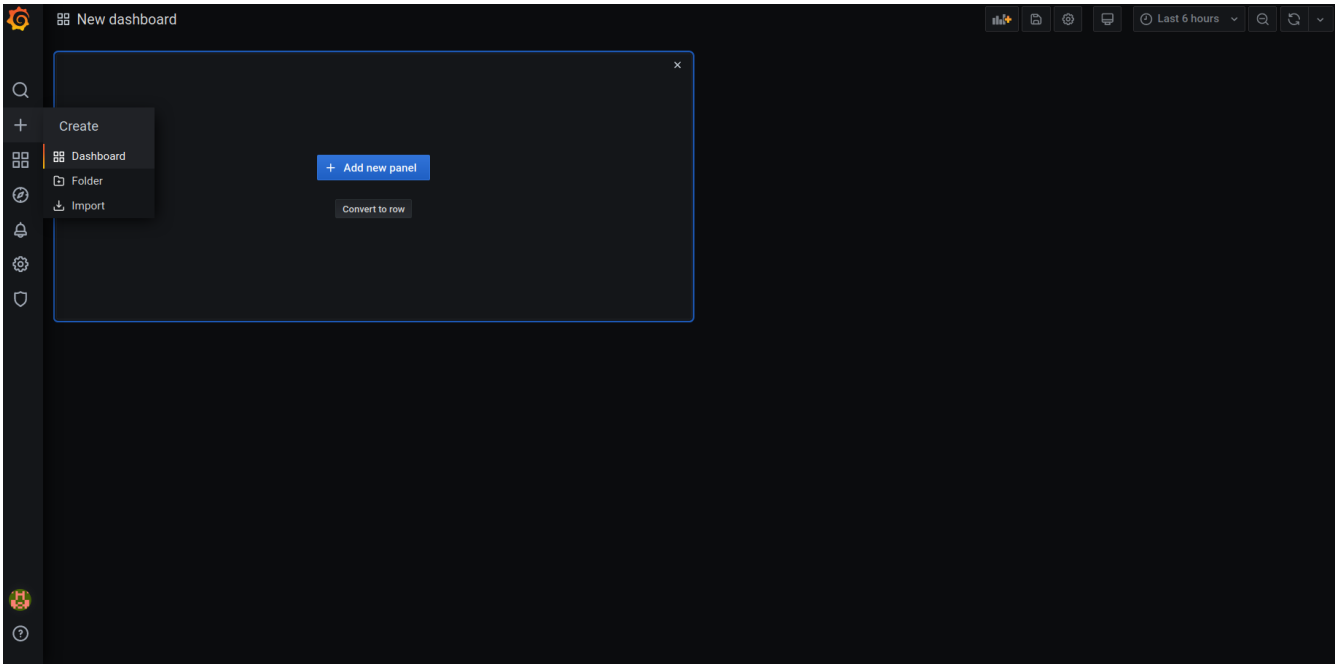
Ensuite allez dans le menu des datasources :



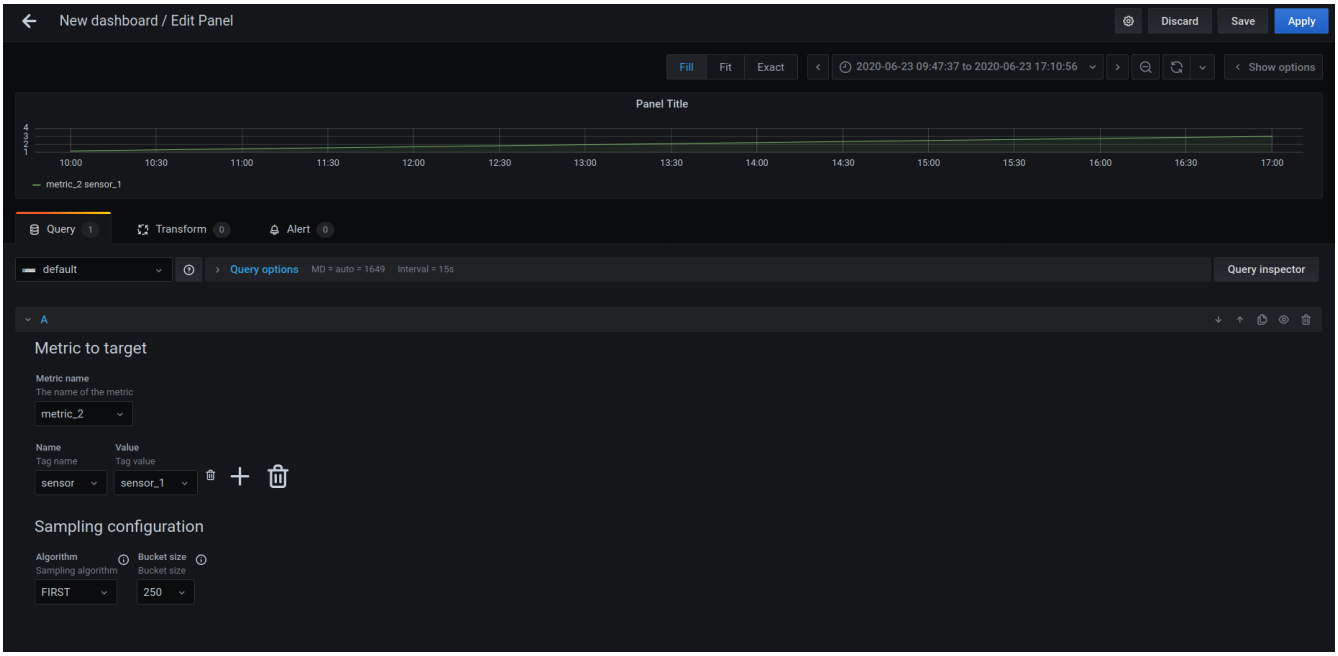
Cherchez la datasource "Hurence-Historian" et sélectionnez la. Il y a juste a renseigné l'URL http://localhost:8080/api/grafana/v0 dans le cas de l'installation standalone.



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" consiste d'un nom de métrique, et de pairs de tag name/tag value ainsi que des options pour l'algorithme de sampling.

note

Pour le moment les options de sampling de la première query fait fois pour toutes les autres.

note

Pour l'instant vous n'avez pas de données d'injecté. Suivez le tutorial pour injecter des données. Aussi si vous n'avez pas ajouter de tag name a l'installation vous ne pourrez pas utiliser de tags. Il est toujours possible de rajouter des tag manuellement après installation en rajoutant un champ dans le schéma.

Tear down and cleanup

This section shows how to stop the services and cleanup data if needed.

Stop your SolR instances

when you’re done you can stop your SolR cores. Attention cette commande va éteindre Solr (cela pourrait impacter d’autres service utilisant solr).

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

Data management

Vous pouvez consulter notre documentation sur l'api REST 'rest-api.pdf' pour connaitre les détails de chaque endpoint. Maintenant que nous avons installer 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

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
}'
```

Get all metrics names

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

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

Get some measures points within a time range

```
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" }
      ]
}'
```

Retrieve data from REST API

TODO

Use Spark to get data

Apache Spark is an Open Source framework designed to process hudge datasets in parallel on computing clusters. Hurence Data Historian is highly integrated with Spark so that you can handle dataset interactions in both ways (input and output) through a simple API.

The following commands show you how to take a CSV dataset from HDFS or local filesystem, load it as a 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)
```

Visualize data through grafana

Realtime data ingestion with logisland