



Guide utilisateur

© Hurence

Version v1.0, 02.07.2020: First book

Table des matières

1. Introduction	1
2. Concepts	2
2.1. Modèle de données	2
2.2. Encodage symbolique SAX	4
3. Installation	6
3.1. Installation standalone	6
3.1.1. Pre-requis pour une installation mono serveur en standalone (pour tester)	6
3.1.2. Installation de l'historian	7
Description du fichier de configuration de l'historian	9
Description des composants installés par le script d'installation	12
3.2. Installation mono noeud de production	12
3.2.1. Pre-requis pour une installation mono serveur en production	12
3.2.2. Installation de Apache SolR	13
3.2.3. Install Apache Spark	13
3.2.4. Installation de Grafana	13
Installation Plugin Grafana pour utiliser le data historian Hurence	13
3.2.5. Installation de l'historian	14
Description du fichier de configuration de l'historian	16
4. Redémarrage du data historian	19
4.1. Redémarrer solr	19
4.2. Redémarrer l'historian	19
4.3. Redémarrer Grafana	19
5. Gestion des données	20
5.1. Import de donnée à partir de fichiers csv avec l'api REST	20
5.2. Requête des données avec l'api REST	21
5.3. Utiliser Spark pour requêter les données	24
5.4. L'injection temps réel avec LogIsland	26
6. Visualisation des données	27
6.1. Configurer la data source grafana	27
6.1.1. Explication sans image	27
6.1.2. Explication avec image	27
6.2. Configurer vos dashboard	29
7. Stopper et détruire les données	33
7.1. Stopper ses instances SolR	33

Chapter 1. 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.

Chapter 2. 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 utilisons 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.

2.1. Modèle de données

Une Measure est un point de donnée dans le temps. On peut y associer des méta-informations sous forme de tags.

```
class Measure {  
    String name;  
    long timestamp;  
    double value;  
    float quality;  
    Map<String, String> tags;  
}
```

Un Chunk est un ensemble de Measures ordonnées chronologiquement groupées par nom ou valeurs de tags. La valeur est un tableau de bytes encodé en protocol buffer, ce qui implique la compaction des données.

```

class Chunk {
    SchemaVersion version = SchemaVersion.VERSION_1;
    String name, id, metricKey;
    byte[] value;
    long start, end;
    Map<String, String> tags;

    long count, first, min, max, sum, avg, last, stdDev;
    float qualityFirst, qualityMin, qualityMax, qualitySum, qualityAvg;

    int year, month;
    String day;
    String origin;
    String sax;
    boolean trend;
    boolean outlier;
}

```

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 avec le schéma suivant :

```

<schema name="default-config" version="1.6">
  ...
  <field name="chunk_avg" type="pdouble"/>
  <field name="chunk_count" type="pint"/>
  <field name="chunk_day" type="string"/>
  <field name="chunk_end" type="plong"/>
  <field name="chunk_first" type="pdouble"/>
  <field name="chunk_hour" type="pint"/>
  <field name="chunk_last" type="pdouble"/>
  <field name="chunk_max" type="pdouble"/>
  <field name="chunk_min" type="pdouble"/>
  <field name="chunk_month" type="pint"/>
  <field name="chunk_origin" type="string"/>
  <field name="chunk_outlier" type="boolean"/>
  <field name="chunk_quality_avg" type="pfloat"/>
  <field name="chunk_quality_first" type="pfloat"/>
  <field name="chunk_quality_max" type="pfloat"/>
  <field name="chunk_quality_min" type="pfloat"/>
  <field name="chunk_quality_sum" type="pfloat"/>
  <field name="chunk_sax" type="ngramtext"/>
  <field name="chunk_start" type="plong"/>
  <field name="chunk_std_dev" type="pdouble"/>
  <field name="chunk_sum" type="pdouble"/>
  <field name="chunk_trend" type="boolean"/>
  <field name="chunk_value" type="text_general" multiValued="false" indexed="false"/>
  <field name="chunk_year" type="pint"/>
  <field name="metric_id" type="string" docValues="true" multiValued="false" indexed=
"true" stored="true"/>
  <field name="metric_key" type="string"/>
  <field name="name" type="string" multiValued="false" indexed="true" required="true"
stored="true"/>
</schema>

```

2.2. Encodage symbolique SAX

On notera que le Chunk comporte des champs agrégés. Outre les statistiques classiques sur la valeur et la qualité du point, on intègre aussi l'encodage symbolique avec SAX.

L'avantage d'utiliser SAX est qu'il est capable d'agir comme un outil de réduction de dimensionnalité, il tolère les séries temporelles de différentes longueurs et facilite la recherche de tendances.

Le codage SAX est une méthode utilisée pour simplifier les séries chronologiques grâce à une sorte de résumé des intervalles de temps. En calculant la moyenne, en regroupant et en représentant symboliquement les périodes, les données deviennent beaucoup plus petites et plus faciles à traiter, tout en capturant ses aspects importants. On peut par exemple s'en servir pour détecter des changements statistiques des tendance et donc des comportements anormaux.

Chapter 3. 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.

IMPORTANT

N'ouvrez pas les ports de votre data historian sur le réseau public, protégez le a minima par un firewall, car cette installation n'est pas sécurisée. Une installation très sécurisée est possible; contactez nous pour plus d'information.

NOTE

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

Dans ce guide nous proposons une installation simple et rapide qui est idéale pour tester, développer et stocker de petits volumes de données peu sensibles. Nous ne recommandons 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.](#)

3.1. 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.

3.1.1. 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 ou Ubuntu ou Mac

- 16 Gigabits de RAM
- 8 vcores de CPU
- 250 Giga octets de disque
- Java 8

3.1.2. 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 :

NOTE

Solr n'aime pas pour des questions de sécurité que vous installiez et lanciez en root. Utilisez donc un utilisateur qui n'est ni sudoer ni root (un utilisateur applicatif ou le vôtre) mais vérifiez bien que vous avez les droits en écriture là où HDH va s'installer (par défaut /opt/hdh).

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

NOTE

le script install.sh doit être exécutable. Si vous obtenez un problème d'autorisation au lancement, vérifiez qu'il a les bonnes propriétés et sinon exécutez cette commande:

```
chmod +x install.sh
```

NOTE

lors de l'installation Solr vous met en garde éventuellement sur votre configuration machine en ce qui concerne le nombre de fichiers ouverts qui est généralement réduit à 1024. Si vous voulez une installation robuste pour accueillir de nombreuses données il vous faudra changer ce paramètre avec la commande ulimit (et l'édition d'un fichier de configuration dont le nom dépend de votre OS). Référez vous à votre documentation système.

Il sera alors demandé 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.

Voici un exemple de l'installation :

```

Os detected is linux : linux-gnu
Wget is already installed
Where do you want to install Hurence Data Historian ?[/opt/hdh]

Do you want us to install an embedded solr (version 8.2.0 required)? (otherwise you need to have one that can be used from this machine)
1) Yes
2) No
#? 1
Which name to use for the solr collection which will be storing time series ?[historian]

Which name to use for the solr report collection ?[historian-report]

Do you want us to add tags names for your time series (you can always add them after installation ?)
1) Add_tags
2) Skip
#? 1

Tag name (STOP when you want stop): tag1
tapped tag tag1
Tag name (STOP when you want stop): stop
array is tag1

Do you want us to install an embedded grafana (version 7.6.3 required)? (otherwise you need to have one that can be used from this machine if you plan to use grafana)
1) Yes
2) No
#? 1
Do you want us to install the historian datasource grafana plugin ? You need it to see data with grafana. We can install it only if grafana is on this machin as single node otherwise you will have to install it manually. If
you choose to install an embedded grafana you can install it as well.
1) Yes
2) No
#? 1
Do you want us to install an embedded spark (this is not required)?
1) Yes
2) No
#? 2

```

NOTE

Attention au chemin où 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.

Et voici un aperçu des derniers logs si tout s'est bien passé lors de l'installation (vérifiez que vous n'avez pas eu d'erreur d'affiché) :

```

Install completed. Starting historian...
==== Start
Started.
check logs at '/home/gregoire/tmp/install_historian/test/historian-1.3.5/bin/./app.log'
The historian server is now running.
You can use ./bin/historian-server.sh [start|stop|restart] to manage the historian server.

```

Vous pouvez vérifier que votre serveur a bien été lancé avec la commande suivante:

```
curl http://localhost:8080/api/grafana/v0
```

Si ce n'est pas le cas, allez voir les logs (dans le répertoire d'installation et le sous répertoire *historian-1.3.5*, il y a un fichier app.log). En général ce sera un problème de droits pour créer des répertoires: notamment dans /tmp (vérifiez bien vos droits en écriture sur ce répertoire).

Pour démarrer ou redémarrer l'historian (en cas de problème), tapez la commande suivante dans le répertoire d'installation :

```
historian-1.3.5/bin/historian-server.sh start
(or historian-1.3.5/bin/historian-server.sh restart)
```

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 8.2.0 installé dans \$HDH_HOME/solr-8.2.0

- Le serveur solr a été démarré par le script d'installation. 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 7.0.3 installé dans \$HDDH_HOME/grafana-7.0.3
- 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. Les binaires du plugin sont installés dans \$HDDH_HOME/grafana-7.0.3/data/plugins/grafana-historian-datasource/dist
- 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 \$HDDH_HOME/historian-1.3.5
- Un répertoire "\$HDDH_HOME/data" qui va contenir les time series de l'historian.

Voici la structure de \$HDDH_HOME à l'issue de l'installation par défaut :

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

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,
    "upload_directory" : "/tmp/historian",
    "max_data_points_maximum_allowed" : 50000
  },
  "historian": {
    "schema_version": "VERSION_1",
    "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 où les fichiers csv seront upload (ils sont effacés une fois l'import de données achevé).
- 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 à 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 où sont stockées les timeseries.
 - `historian/solr/annotation_collection` : Le nom de la collection où 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 `$HDH_HOME/data`.

3.2. Installation mono noeud de production

3.2.1. 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 ou Ubuntu ou Mac
- 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](#)

3.2.2. 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"

3.2.3. Install Apache Spark

Pour installer spark vous pouvez télécharger cette archive : [spark-2.3.4-bin-without-hadoop.tgz](#)

Les commandes suivantes vous permettront d'avoir une installation locale. Sinon veuillez suivre 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/
```

3.2.4. 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](#)

3.2.5. 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 :

NOTE

Solr n'aime pas pour des questions de sécurité que vous installiez et lanciez en root. Utilisez donc un utilisateur qui n'est ni sudoer ni root (un utilisateur applicatif ou le vôtre) mais vérifiez bien que vous avez les droits en écriture là où HDH va s'installer (par défaut /opt/hdh).

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

NOTE

le script install.sh doit être exécutable. Si vous obtenez un problème d'autorisation au lancement, vérifiez qu'il a les bonnes propriétés et sinon exécutez cette commande:

```
chmod +x install.sh
```

NOTE

lors de l'installation Solr vous met en garde éventuellement sur votre configuration machine en ce qui concerne le nombre de fichiers ouverts qui est généralement réduit à 1024. Si vous voulez une installation robuste pour accueillir de nombreuses données il vous faudra changer ce paramètre avec la commande ulimit (et l'édition d'un fichier de configuration dont le nom dépend de votre OS). Référez vous à votre documentation système.

Il sera alors demandé à l'utilisateur plusieurs informations.

Here is an example:

```
Os detected is linux : linux-gnu
wget is already installed
Where do you want to install Hurence Data Historian ?[/opt/hdh]

Do you want us to install an embedded solr (version 8.2.0 required)? (otherwise you need to have one that can be used from this machine)
1) Yes
2) No
#? 2

What is the path to the solr cluster ? We will use the solr REST api to create collection.[localhost:8983/solr]
sysolehost:port/solr
Which name to use for the solr collection which will be storing time series ?[historian]

Which name to use for the solr report collection ?[historian-report]

Do you want to add tags names for your time series (you can always add them after installation ?)
1) Add_tags
2) Skip
#? 2
array is

Do you want us to install an embedded grafana (version 7.8.3 required)? (otherwise you need to have one that can be used from this machine if you plan to use grafana)
1) Yes
2) No
#? 2

Do you want us to install the historian datasource grafana plugin ? You need it to see data with grafana. We can install it only if grafana is on this machin as single node otherwise you will have to install it manually. If
you choose to install an embedded grafana you can install it as well.
1) Yes
2) No
#? 1
[path/to/grafana/data/plugins]
path/to/plugin/dir/for/my/grafana/instance
Do you want us to install an embedded spark (this is not required)?
1) Yes
2) No
#? 2
```


NOTE

Attention au chemin où 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.

Et voici un aperçu des derniers logs si tout s'est bien passé lors de l'installation (vérifiez que vous n'avez pas eu d'erreur d'affiché) :

```
Install completed. Starting historian...
==== Start
Started.
check logs at '/home/gregoire/tmp/install_historian/test/historian-1.3.5/bin/./app.log'
The historian server is now running.
You can use ./bin/historian-server.sh [start|stop|restart] to manage the historian server.
```

Vous pouvez vérifier que votre serveur a bien été lancé avec la commande suivante:

```
curl http://localhost:8080/api/grafana/v0
```

Si ce n'est pas le cas, allez voir les logs (dans le répertoire d'installation et le sous répertoire *historian-1.3.5*, il y a un fichier *app.log*). En général ce sera un problème de droits pour créer des répertoires: notamment dans /tmp (vérifiez bien vos droits en écriture sur ce répertoire).

Pour démarrer ou redémarrer l'historian (en cas de problème), tapez la commande suivante dans le répertoire d'installation :

```
historian-1.3.5/bin/historian-server.sh start
(or historian-1.3.5/bin/historian-server.sh restart)
```

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/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.

Le script \$HDH_HOME/historian-1.3.5/bin/historian-server.sh sert à lancer/arrêter l'api rest de l'historian.

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,
    "upload_directory" : "/tmp/historian",
    "max_data_points_maximum_allowed" : 50000
  },
  "historian": {
    "schema_version": "VERSION_1",
    "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 où les fichiers csv seront upload (ils sont effacés une fois l'import de données achevé).
- 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 à 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 où sont stockées les timeseries.
- `historian/solr/annotation_collection` : Le nom de la collection où 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 d'un fichier de configuration pendant le script d'installation selon les informations renseignées.

Chapter 4. Redémarrage du data historian

Il y a un ordre à respecter pour démarrer ou redémarrer le data historian. Il faut dans l'ordre:

1. [Redémarrer Solr](#). Si il n'est pas complètement stoppé, stoppez le (voir la section Arrêter Solr).
2. [Redémarrer l'historian server](#)
3. [Redémarrer Grafana](#)

4.1. Redémarrer solr

Si vous avez coupé Solr ou bien si vous avez redémarré 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 $HDH_HOME/data/solr/node1 -p 8983
# démarre un second core Solr localement qui va utiliser le serveur zookeeper
précédemment créer.
bin/solr start -cloud -s $HDH_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](#)

4.2. Redémarrer l'historian

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

Si cela ne marche pas car un PID (un process id) existe et le data historian est supposé up, alors faite un stop, puis refaite un 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.

4.3. Redémarrer Grafana

Utilisez restart si c'est un redémarrage. Si votre grafana ne tourne pas déjà utilisez start.

```
$HDH_HOME/grafana-7.0.3/bin/grafana-server start
```

Chapter 5. Gestion des données

Vous pouvez consulter notre documentation sur l'api REST [api rest](#) pour connaître 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.

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

Cette section va nous apprendre à injecter un fichier CSV dans l'historian. Le folder **samples** du data historian contient des exemples. Nous allons injecter les données du sous folder **happiness** ; quelques fichiers qui donnent le score de "happiness" des pays chaque année.

Exécutez la commande suivante (dans ce qui suit remplacez \$HDH_HOME par votre répertoire d'installation et éventuellement l'adresse de votre serveur historian si vous n'avez pas effectué une installation locale):

```
curl -X POST \  
http://localhost:8080/api/historian/v0/import/csv \  
-F 'my_csv_file=@$HDH_HOME/historian-1.3.5/samples/happiness/2015.csv' \  
-F mapping.value="Happiness Score" \  
-F mapping.timestamp=Timestamp \  
-F mapping.tags=Country \  
-F group_by=name \  
-F group_by=tags.Country \  
-F format_date='yyyy-MM-dd HH:mm:ss'
```

et bien sûr vous pouvez appliquer la même requête à tous les fichiers de 2016 à 2020. Cela vous fait un petit jeu de données sur lequel nous allons pouvoir apprendre à faire des choses.

Ces query devraient retourner une réponse de ce type :

```
{
  "tags" : [ "Country" ],
  "grouped_by" : [ "name", "Country" ],
  "report" : [ {
    "name" : "happiness",
    "Country" : "Switzerland",
    "number_of_points_injected" : 1,
    "number_of_point_failed" : 0,
    "number_of_chunk_created" : 1
  },
  ..., {
    "name" : "happiness",
    "Country" : "Togo",
    "number_of_points_injected" : 1,
    "number_of_point_failed" : 0,
    "number_of_chunk_created" : 1
  }
]
}
```

5.2. Requête des données avec l'api REST

Une manière de voir si vos données ont bien été injectées est de les requêter. Essayons de voir la progression du bonheur (métrique happiness) en France sur les années 2015 à 2020. Et essayons de l'obtenir en Json.

Déjà regardons simplement si il existe des métriques happiness dans l'historien avec la requête suivante:

```
curl --header 'Content-Type: application/json' -X POST \
http://localhost:8080/api/grafana/v0/query -d '{
  "names": ["happiness"]
}'
```

Cette query retourne une réponse de ce type :

```
[{
  "name": "happiness",
  "datapoints": [[7.587, 1420070400000], ..., [2.566900015, 1577836800000]]
}]
```

En filtrant sur le tag Country (= France) nous devrions avoir les enregistrements de la France uniquement.

```
curl --header 'Content-Type: application/json' -X POST \
http://localhost:8080/api/grafana/v0/query -d '{
  "names": ["happiness"],
  "tags": {
    "Country": "France"
  }
}'
```

Cette query retourne la réponse suivante :

```
[{
  "name": "happiness",
  "tags": {"Country": "France"},
  "datapoints": [
    [6.575, 1420070400000],
    [6.478, 1451606400000],
    [6.44199991226196, 1483228800000],
    [6.489, 1514764800000],
    [6.592, 1546300800000],
    [6.663799763, 1577836800000]
  ]
}]
```

Obtenir l'ensemble des noms de métriques.

```
curl --X POST 'http://localhost:8080/api/grafana/v0/search' \
--header 'Content-Type: application/json' \
--data-raw '{
  "name": "ha",
  "limit": 100
}'
```

Ici on précise que l'on désire avoir toutes les noms de métrique contenant la string "ha" mais de n'en retrouver qu'un maximum de 100. Si l'on ne renseigne pas le champ "name" alors la requête retourne tout les noms de métrique.

```
["happiness"]
```

Obtenir l'ensemble des tags présent dans l'historien.

```
curl -X POST 'http://localhost:8080/api/grafana/v0/search/tags'
```

Cette query retourne la liste des tags présent en json :


```
["Country"]
```

Obtenir l'ensemble des valeurs d'un tag.

```
curl -X POST 'http://localhost:8080/api/grafana/v0/search/values' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "field": "Country",  
  "query": "F",  
  "limit": 100  
}'
```

Ici on précise que l'on désire avoir toutes les valeurs existante pour le tag "Country" contenant la string "F" mais de n'en retourner qu'un maximum de 100. Si l'on ne renseigne pas le champ "name" alors la requête retourne tous les noms de métrique.

Cette query retourne cette liste :

```
["Burkina Faso", "Finland", "France"]
```

Attention la requête est sensible aux majuscules. La même requête avec un "f" minuscule retournera en effet :

```
["Afghanistan", "Central African Republic", "Hong Kong S.A.R. of China", "South Africa",  
"Taiwan Province of China"]
```

Obtenir des mesures en spécifiant :

- Un intervalle de temps
- La métrique dont on cherche les données est "happiness"
- On veut au maximum 10 points
- Si il y a besoin d'effectuer un sampling nous voulons utiliser un algorithme à base de bucket et en prenant la plus petite valeur à chaque fois.

```
curl -X POST 'http://localhost:8080/api/grafana/v0/query' \
--header 'Content-Type: application/json' \
--data-raw '{
  "from": "2015-01-01T00:00:00.000Z",
  "to": "2016-01-01T00:00:00.000Z",
  "names": ["happiness"],
  "max_data_points": 10,
  "sampling":{
    "algorithm": "MIN",
    "bucket_size" : 100
  }
}'
```

Cette query retourne cette réponse :

```
[{
  "name": "happiness",
  "datapoints": [
    [4.867, 1420070400000],
    [2.839, 1420070400000],
    [3.36, 1451606400000],
    [2.905, 1451606400000]
  ]
}]
```

Nous obtenons bien moins de 10 points. Attention comme nous n'avons pas précisé de tags les valeurs de tous les pays sont mélangés !

note

Il n'y a ici que 4 points de retournés alors que nous avons spécifié un maximum de 10 points. Cela n'est pas optimum car il y a plus de 10 points en tout ! En effet nous devrions ici avoir 10 points. Ce problème sera fixé lors de la prochaine release. En attendant n'hésitez pas à augmenter la valeur de "max_data_points" si cela devait vous arriver.

warn

Attention cependant à ne pas demander trop de points ! En effet si vous voulez exporter la totalité des points cette endpoint n'est pas l'outil adéquat. D'ailleurs si le nombre indiqué est supérieur à 2147483647 la requête ne fonctionnera pas. Dans une future release le maximum pour cette variable sera revu à la baisse.

5.3. 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 interactions 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.

```
$HDH_HOME/spark-2.3.4-bin-hadoop2.7/bin/spark-shell --jars assembly/target/historian-1.3.5-SNAPSHOT/historian-1.3.5-SNAPSHOT/lib/loader-1.3.5-SNAPSHOT.jar,assembly/target/historian-1.3.5-SNAPSHOT/lib/spark-solr-3.6.6-shaded.jar
```

```
import com.hurence.timeseries.model.Chunk
import com.hurence.historian.spark.ml.Chunkyfier
import com.hurence.historian.spark.sql
import org.apache.spark.sql._
import com.hurence.historian.spark.sql.functions._
import com.hurence.historian.spark.sql.reader.{ChunksReaderType, 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/historian-spark/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")).setSaxStringLength(20)
val chunksDS =
chunkyfier.transform(measuresDS).as[Chunk](Encoders.bean(classOf[Chunk]))

val writer = WriterFactory.getChunksWriter(WriterType.SOLR)
writer.write(sql.Options("historian", Map(
```

```

"zkhost" -> "localhost:9983",
"collection" -> "historian",
"tag_names" -> "metric_id,warn,crit"
)), chunksDS)

// 4. Explicit commit to make sure all docs are visible
val solrCloudClient = SolrSupport.getCachedCloudClient("localhost:9983")
solrCloudClient.commit("historian", true, true)

val reader = ReaderFactory.getChunksReader(ChunksReaderType.SOLR)
val solrDF = reader.read(sql.Options("historian", Map(
    "zkhost" -> "localhost:9983",
    "collection" -> "historian",
    "tag_names" -> "metric_id,warn,crit"
))).as[Chunk](Encoders.bean(classOf[Chunk]))

```

5.4. 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.

Chapter 6. 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).

6.1. Configurer la data source grafana

En général vous voulez visualiser vos données plutôt que de les requêter. Allez sur [l'URL de grafana](#). Vous pouvez vous logger en tant que user: admin et pwd: admin (Vous pouvez passer l'écran suivant).

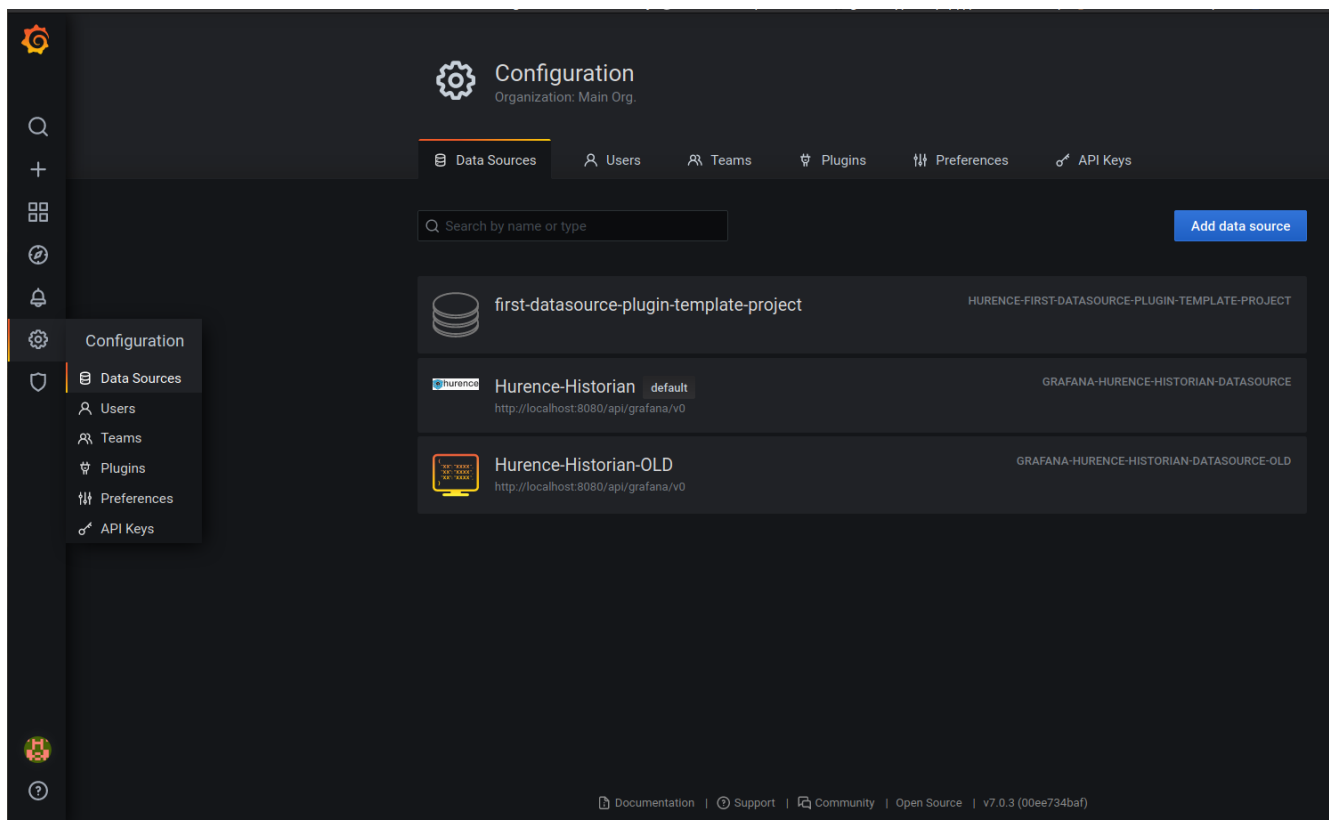
Vous avez une page d'accueil que nous pouvez lire puis faire disparaître. Afin de pouvoir consulter les données du data historian :

6.1.1. Explication sans image

1. Vous allez cliquer sur le bouton de Configuration à gauche puis sur "Data Sources".
2. Cliquez sur le bouton "Add data source".
3. Aller à la fin et cliquez sur celle qui s'appelle "Hurence-Historian" ou bien filtrer dans la barre de recherche pour la faire apparaître.
4. Mettez dans la case "URL" l'URL de votre historian server (l'installation par défaut est <http://localhost:8080/api/grafana/v0/>)
5. Cliquez "Save and Test"
6. Vous devez avoir un message vert vous disant que votre data source est chargée et bien configuré (joignable).

6.1.2. Explication avec image

Allez 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.

⌵⌴ Settings

Name ⓘ

Hurence-Historian

Default

☒

HTTP

URL ⓘ

http://localhost:8080/api/grafana/v0

Access

Server (default) ▾

Help >

Whitelisted Cookies ⓘ

Add Name

Add

Auth

Basic auth

☐

With Credentials ⓘ

☐

TLS Client Auth

☐

With CA Cert ⓘ

☐

Skip TLS Verify

☐

Forward OAuth Identity ⓘ

☐

Custom HTTP Headers

Header

admin

Value

configured

Reset

🗑

+ Add header

Max search metric ⓘ

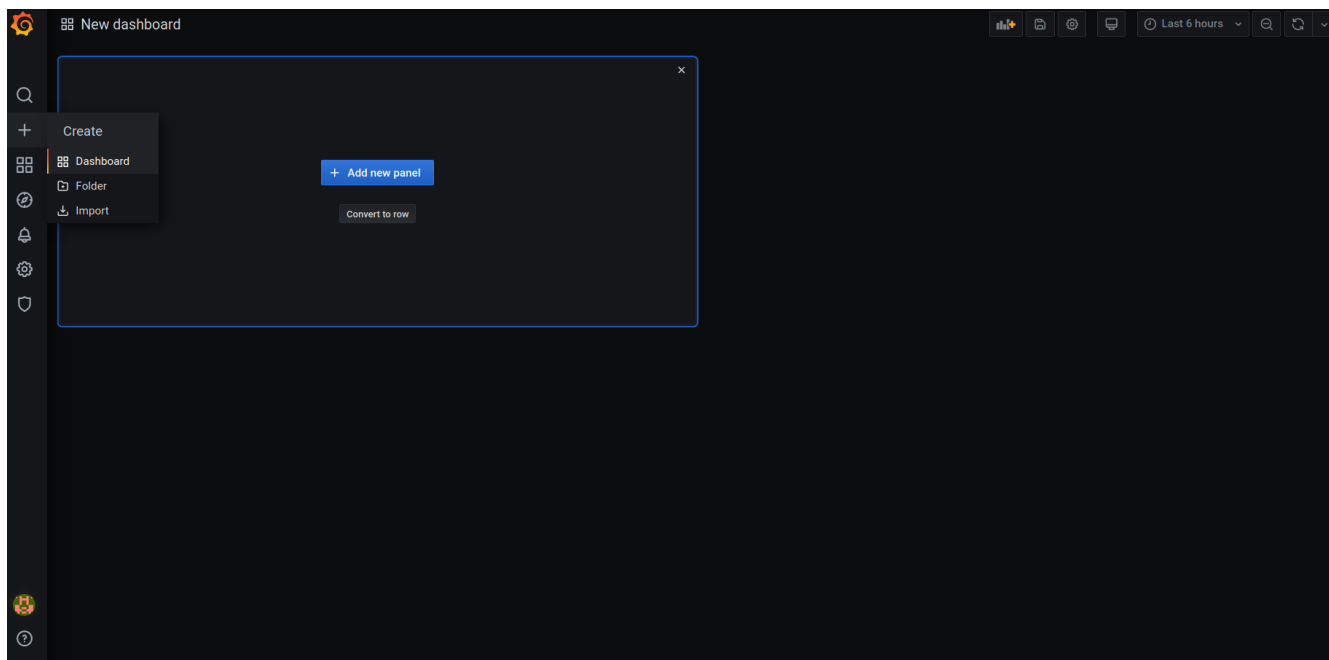
10

Testez la connectivité en cliquant sur "Save & Test" button at the bottom of the page.

6.2. Configurer vos dashboard

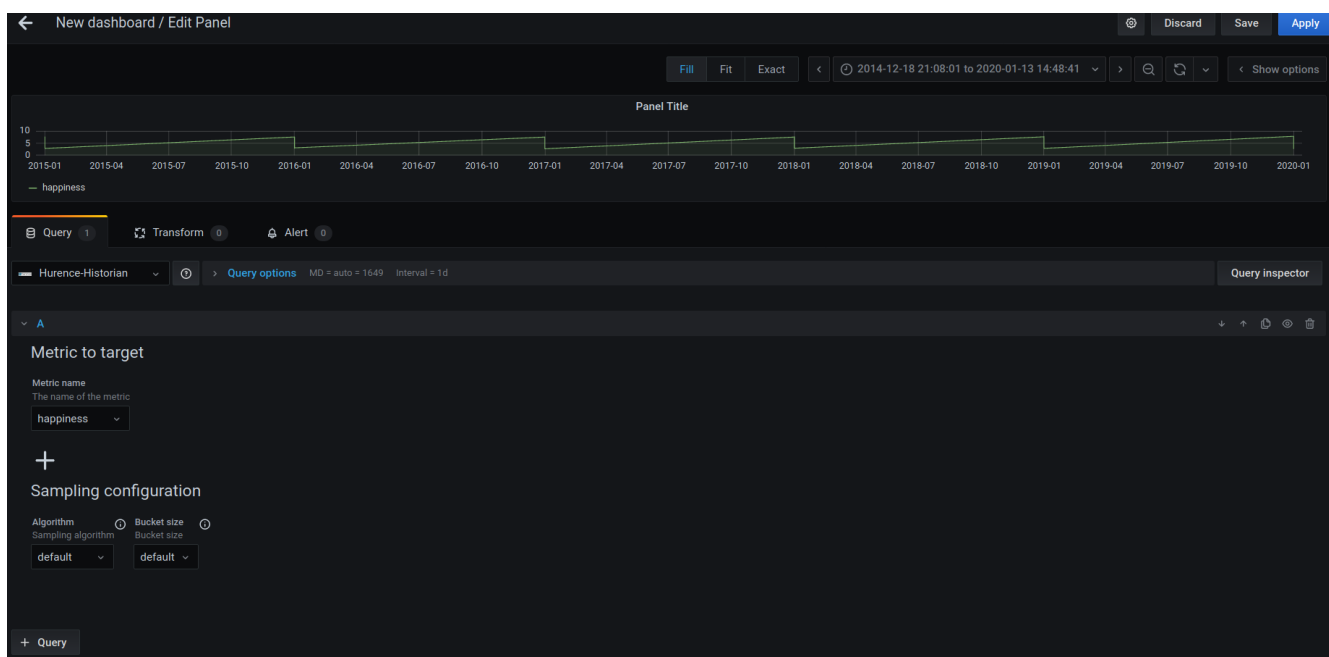
Pour créer des dashboards il faut aller sur l'interface graphique de grafana [l'URL de grafana](#) (localhost étant à remplacer par un nom de machine si vous n'êtes pas en installation standalone).

Ensuite créez un nouveau dashboard.

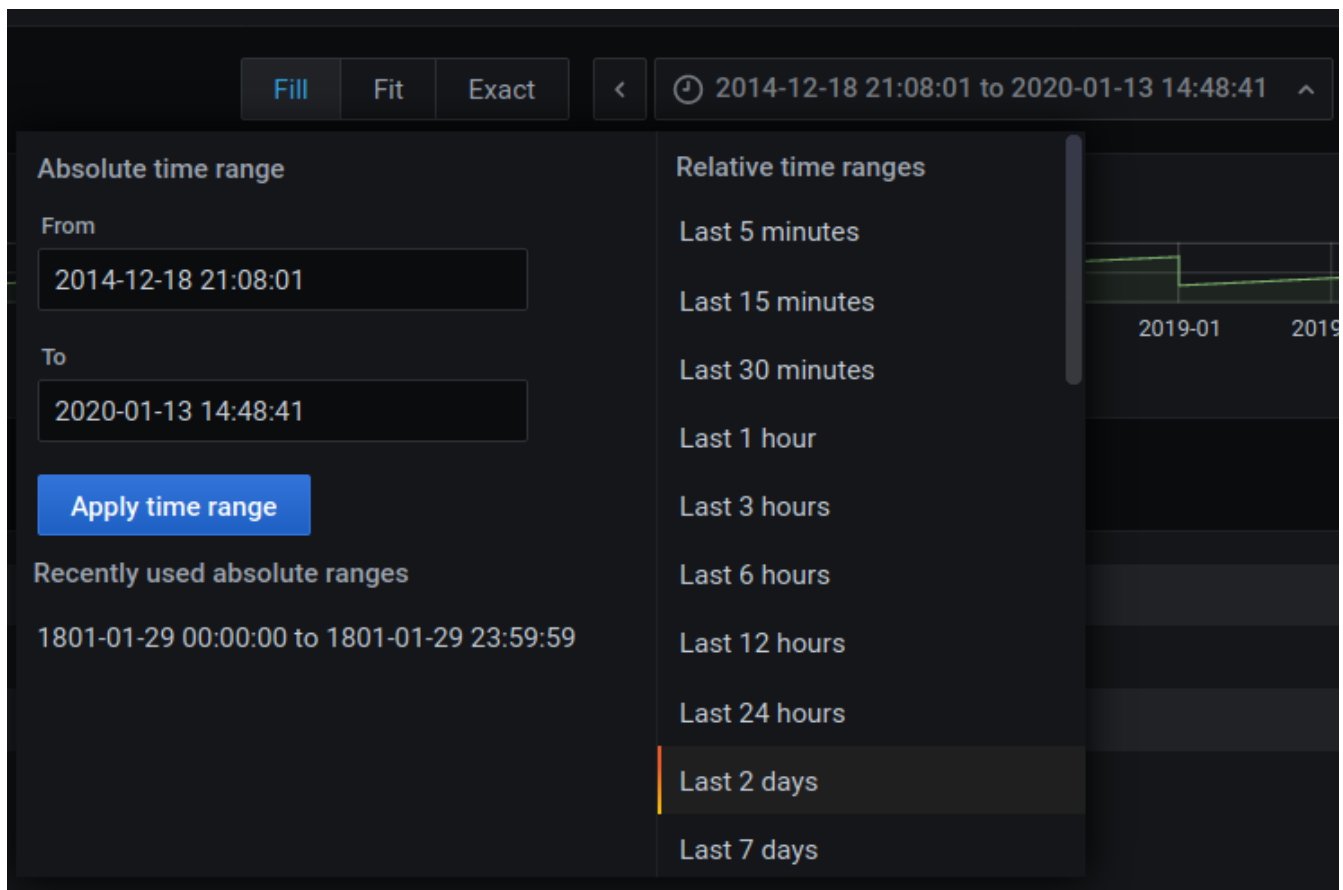


Ajoutez les graphes que vous voulez, un petit exemple ci-dessous :

- Nous avons sélectionné la métrique "happiness" (préalablement injecté).
- La time range choisis est de "2014-12-18 21:08:01" à "2020-01-13 14:48:41".

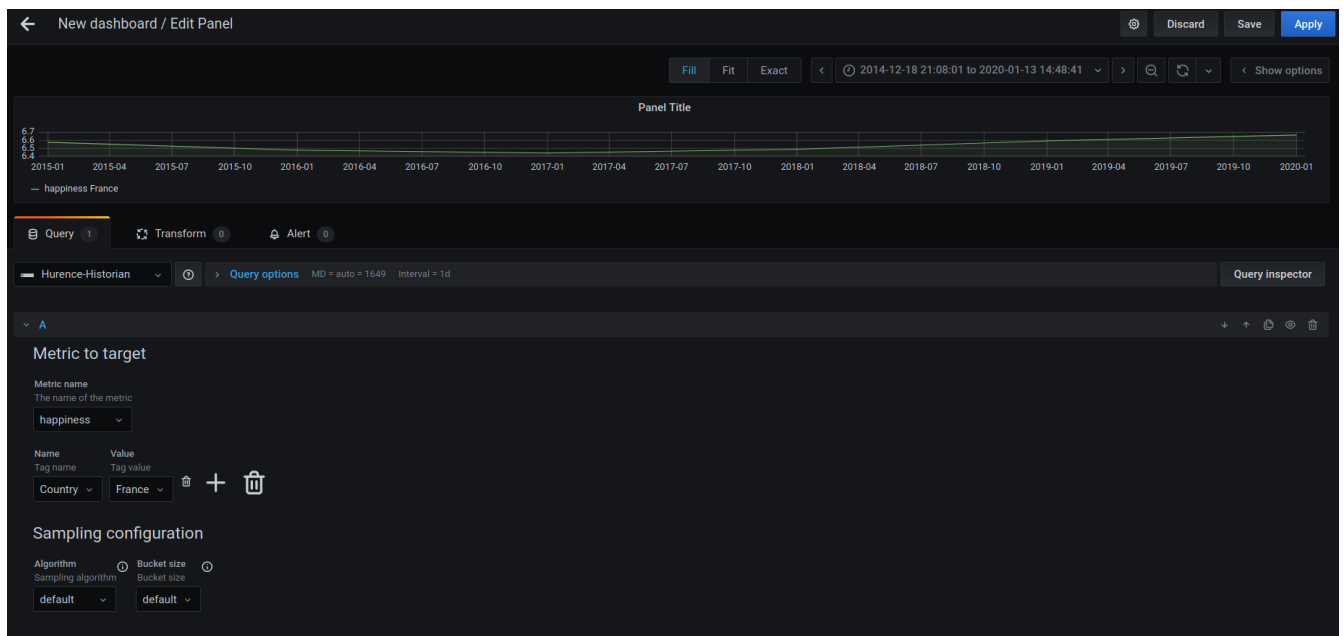


Pour configurer la time range :



TODO remarque sur la visu

Si on filtre sur le tag "Country" avec la valeur France par exemple on obtient :



Pour cela il faut cliquer sur "Add new tag filter". Puis sélectionner la valeur "Country" comme "Tag name". Enfin cliquer sur "Tag value" et commencer à taper le nom du pays désirer puis sélectionner "France" lorsque celui-ci apparaît.

Metric to target

Metric name
The name of the metric

happiness ▾

+ Add new tag filter

Sampling configuration

Algorithm ⓘ
Sampling algorithm

default ▾

Bucket size ⓘ
Bucket size

default ▾

TODO général note

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.

Chapter 7. 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).

7.1. Stopper ses instances Solr

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

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