Graphite: définition de la précision du stockage des données

Sommaire

Concept

Les bases de métrologie comme **Graphite** doivent stocker des données en quasi-temps réel (*avec une fréquence de l'ordre de la minute*), et ce, sur plusieurs années.

- Bien que Graphite permette de conserver toutes ces données sur disque, une telle approche mènerait rapidement à une surcharge de l' espace de stockage.
- Graphite repose sur une idée simple : **plus les données sont anciennes, moins leur précision est cruciale, au profit de la mise en évidence des tendances**.
- Graphite intègre un **mécanisme de rétention des données**.
  - Une rétention correspond à une précision associée à une période de couverture.
  - L'objectif est de définir plusieurs niveaux de rétention, avec une granularité qui diminue au fil du temps (*une donnée toutes les minutes pendant une journée, une donnée toutes les cinq minutes durant une semaine ...*).

Lorsqu'une donnée change de rétention, **Graphite effectue une agrégation** (*une moyenne*) afin de réduire le volume tout en conservant l'information essentielle.

Cela a deux avantages :

- **Stockage optimisé** : possibilité de conserver des données de métrologie sur le long terme sans saturer le disque.
- **Prévisibilité de l'espace disque** : la taille occupée par les métriques est connue dès l'insertion de la première donnée, car l'allocation de l' espace disque est anticipée.

Les **règles de précision** s'appliquent à des **chemins de métriques**.

Chaque métrique — par exemple `cpu_idle` correspondant au check CPU de l'hôte `serveur-1` — dispose de son propre fichier `.wsp`, stocké dans le répertoire : /opt/graphite/storage/whisper/.

- Ce fichier est placé dans un sous-dossier structuré selon le modèle suivant : HOST-UUID/CHECK-UUID/NOM-METRIQUE.wsp.

Il est donc possible de définir des **règles de précision** à différents niveaux :

- **Globalement**, pour tous les éléments de supervision gérés par Shinken.
- **Par hôte**, pour appliquer des règles spécifiques à un serveur ou équipement.
- **Par check**, pour affiner la rétention en fonction du type de supervision (*CPU, mémoire, ...*).

Configurer la précision du stockage des données

La configuration des précisions de stockage des données s'effectue dans le fichier **/opt/graphite/conf/storage-schemas.conf**.

Exemple de fichier de configuration :

```
# Schema definitions for Whisper files. Entries are scanned in order,
# and first match wins. This file is scanned for changes every 60 seconds.
#
# Definition Syntax:
#
#    [name]
#    pattern = regex
#    retentions = timePerPoint:timeToStore, timePerPoint:timeToStore, ...
#
# name: Arbitrary unique name for the rule
# pattern: Regular expression to match against the metric name. For syntax see:
#          https://docs.python.org/3/library/re.html#regular-expression-syntax
# retentions: Retention schema for the metric
#
# Remember: To support accurate aggregation from higher to lower resolution
#           archives, the precision of a longer retention archive must be
#           cleanly divisible by precision of next lower retention archive.
#
#           Valid:    60s:7d,300s:30d (300/60 = 5)
#           Invalid:  180s:7d,300s:30d (300/180 = 3.333)
#
# This retention is set at the time the first metric is sent.
# Changing this file will not affect already-created .wsp files.
# Use whisper-resize.py to change existing data files.

# Carbon's internal metrics. This entry should match what is specified in
# CARBON_METRIC_PREFIX and CARBON_METRIC_INTERVAL settings
[carbon]
pattern = ^carbon\.
retentions = 60:90d

[default]
pattern = .*
retentions = 60s:1d,5m:30d,1h:3y
```

Pour la configuration carbon, **Il faut** :

- la laisser en premier.
- ne pas la modifier.

```
[carbon]
pattern = ^carbon\.
retentions = 60:90d
```

Chaque règle de précision correspond à une section du fichier, dans laquelle sont définis les paramètres de stockage.

Exemple de section :

```
[everything_1min]
pattern = .*
retentions = 60s:7d,300s:30d,1800s:180d,86400s:547d
```

Les règles du fichier de configuration :

- Ce fichier peut contenir plusieurs sections.
- Les sections sont appliquées dans l'ordre, de la première à la dernière.
- Les patterns sont des expressions régulières.
- Le premier pattern qui correspond au chemin de la métrique est utilisé.
- La précision est définie au moment où la première métrique est envoyée.
- La modification de ce fichier n'aura aucune incidence sur les fichiers .wsp déjà créés.

- Utiliser whisper-resize.py pour les modifier.

> Les modifications du fichier **/opt/graphite/conf/storage-schemas.conf** doivent être effectuées sur la machine du carbon-cache.
>
> Dans le cas d'un cluster Graphite, il faut modifier le fichier sur toutes les machines avec un carbon-cache.

> Le fichier de configuration /opt/graphite/conf/storage-schemas.conf est automatiquement analysé par Graphite toutes les minutes.
> Pour appliquer les modifications immédiatement, il est toutefois possible de redémarrer le service carbon-cache.

Définition d'une section

La règle de précision à appliquer sur une métrique correspond à une section.

Exemple de section :

```
[everything_1min]
pattern = .*
retentions = 60s:7d,300s:30d,1800s:180d,86400s:547d
```

Une section correspond à :

- Un nom
  - à placer entre crochets
  - doit être unique
- Un pattern qui contient une expression régulière correspondant aux métriques visées.
  - pattern = .* correspond à tous les éléments
- Un schéma de rétentions qui décrit les différentes précisions temporelles et durées de conservation appliquées aux données de la métrique.
  - 60s:7d un point toutes les minutes pour 7 jours

Pattern : comment appliquer une précision aux éléments de Shinken

Un **pattern** est une **expression régulière** utilisée pour faire correspondre le nom des nouvelles métriques. Les caractères spéciaux des expressions régulières sont donc utilisables (*par exemple : ^ pour indiquer le début d'une chaîne, $ pour la fin ...*).

Cas particulier des points dans les noms de métriques :

- Graphite interprète les **points (.)** dans les noms de métriques comme des **séparateurs hiérarchiques**.
  - Par exemple : serveur.mongo.ram sera stocké logiquement sous la forme : /serveur/mongo/ram
- Dans Shinken, toutes les métriques sont enregistrées selon le schéma HOST-UUID.CHECK-UUID.METRIC-NAME et donc stockées sous la forme hiérarchique HOST-UUID/CHECK-UUID/METRIC-NAME.
- Il faut ainsi échapper les points dans les expressions régulières.
  - Dans une **expression régulière**, le caractère `.` correspond à **n'importe quel caractère**.
  - Pour faire correspondre un **point littéral**, il faut ainsi **l'échapper avec un antislash (\)**.
    - Exemple : pour matcher la métrique serveur.mongo.ram : serveur\.mongo\.ram

Sur tous les éléments de supervision gérés par Shinken

Pour appliquer les mêmes précisions de stockage à l'ensemble des éléments de Shinken, il faut :

- Placer la section **après** la section `carbon`, qui correspond au schéma de rétention des données des métriques internes à Graphite.
- Utilisez le pattern suivant : pattern = .*

Ce comportement correspond à la configuration par défaut fournie avec Shinken.

Exemple de section permettant d'appliquer la même précision à tous les éléments de Shinken :

```
[everything_1min]
pattern = .*
retentions = 60s:7d,300s:30d,1800s:180d,86400s:547d
```

Pour appliquer les modifications immédiatement :

```
service carbon-cache restart
```

À tous les checks d'un hôte

Il est possible de définir une précision de stockage spécifique pour un hôte dans Graphite.

Pour cela, procéder comme suit :

- Placer la section **avant** la section générale qui matche toutes les données (*c'est-à-dire avant la section avec* `pattern=.*`) et toutes autres sections qui pourraient correspondre à l'hôte.
- Utiliser un pattern qui cible l'HOST-UUID de l'hôte au début du chemin, au format :

```
pattern = ^HOST-UUID.*
```

Exemple de section pour appliquer une précision uniforme à toutes les métriques d'un hôte

```
[specifique]
pattern = ^d723f22cb01111f0a8260800276e8d1e.*
retentions = 1m:1h,5m:2h,1h:30d,1d:1y,8640m:3y
```

Pour appliquer les modifications immédiatement :

```
service carbon-cache restart
```

> **La précision est définie à la création d'un métrique**
>
> - La précision d'un fichier de métrique est fixée lors de la **création du fichier**, soit à la réception de la première métrique, soit lors du lancement de la commande `whisper-resize.py`.
> - Modifier la configuration Graphite ne redimensionnera pas automatiquement les fichiers `.wsp` existants, mais s'appliquera uniquement aux nouveaux fichiers créés.

Pour appliquer un nouveau schéma de précisions sur des métriques existantes, il faut :

- **`whisper-resize`Supprimer le dossier de l'hôte contenant les métriques.**
  - Cela entraînera la régénération des fichiers avec le nouveau schéma, mais **engendre une perte de données**.
  - Le chemin des données d'un hôte est : /opt/graphite/storage/whisper/HOST-UUID/
- Utiliser le script `whisper-resize.py`
  - Ce script permet de redimensionner les fichiers .wsp existants pour appliquer un nouveau schéma de rétentions
  - Exemple d'utilisation de la commande (*1m:1h,5m:2h,1h:30d,1d:1y,8640m:3y correspond à la nouvelle définition des précisions*) :

```
/usr/bin/whisper-resize.py /opt/graphite/storage/whisper/HOST-UUID/*/*.wsp 1m:1h,5m:2h,1h:30d,
1d:1y,8640m:3y
```

Une fois les fichiers créés ou modifiés, il est possible vérifier la prise en compte du schéma dans le fichier de logs /opt/graphite/storage/log/carbon-cache/carbon-cache-a/creates.log:

Chercher des entrées indiquant la création des fichiers `.wsp` avec la précision attendue, par exemple :

```
29/06/2021 16:41:44 :: new metric b6f742865f074584a605d31116833f3d.92182da3c28c5193c0e36b4a68b5e896.
connexion_time matched schema specifique
29/06/2021 16:41:44 :: new metric b6f742865f074584a605d31116833f3d.92182da3c28c5193c0e36b4a68b5e896.
connexion_time matched aggregation schema default_average
29/06/2021 16:41:44 :: creating database file /opt/graphite/storage/whisper/b6f742865f074584a605d31116833f3d
/92182da3c28c5193c0e36b4a68b5e896/connexion_time.wsp (archive=[(60, 60), (300, 24), (3600, 720), (86400,
365), (518400, 182)] xff=0.01 agg=average)
```

À un check, quel que soit l'hôte sur lequel il est accroché

Il est possible de définir une précision de stockage spécifique à un check quelque soit l'hôte sur lequel il est accroché.

Pour cela, procéder comme suit :

- Placer la section **avant** la section générale qui matche toutes les données (*c'est-à-dire avant la section avec* `pattern = .*`) et toutes autres sections qui pourraient matchées le check.
- Utiliser un pattern qui cible l'UUID du check, au format :

```
pattern = .*\.CHECK-UUID\..*
```

Exemple de section pour appliquer une précision uniforme à un check, quelque soit l'hôte auquel il est accroché :

```
[specifique]
pattern = .*\.d723f22cb01111f0a8260800276e8d1e\..*
retentions = 1m:1h,5m:2h,1h:30d,1d:1y,8640m:3y
```

Pour appliquer les modifications immédiatement :

```
service carbon-cache restart
```

La précision est définie à la création d'un métrique

- La précision d'un fichier de métrique est fixée lors de la **création du fichier**, soit à la réception de la première métrique, soit lors du lancement de la commande `whisper-resize.py`.
- Modifier la configuration Graphite ne redimensionnera pas automatiquement les fichiers `.wsp` existants, mais s'appliquera uniquement aux nouveaux fichiers créés.

Pour appliquer un nouveau schéma de précisions sur des métriques existantes, il faut :

- **Supprimer les dossiers du ckeck contenant les métriques.**
  - Cela entraînera la régénération des fichiers avec le nouveau schéma, mais **engendre une perte de données**.
  - Le chemin des données d'un check est : /opt/graphite/storage/whisper/*/CHECK-UUID/*
- Utiliser le script `whisper-resize.py`
  - Ce script permet de redimensionner les fichiers .wsp existants pour appliquer un nouveau schéma de rétentions
  - Exemple d'utilisation de la commande (*1m:1h,5m:2h,1h:30d,1d:1y,8640m:3y correspond à la nouvelle définition des précisions*) :

```
/usr/bin/whisper-resize.py /opt/graphite/storage/whisper/*/CHECK-UUID/*.wsp 1m:1h,5m:2h,1h:30d,
1d:1y,8640m:3y
```

Une fois les fichiers créés ou modifiés, il est possible vérifier la prise en compte du schéma dans le fichier de logs /opt/graphite/storage/log/carbon-cache/carbon-cache-a/creates.log:

Chercher des entrées indiquant la création des fichiers `.wsp` avec la précision attendue, par exemple :

```
29/06/2021 16:41:44 :: new metric b6f742865f074584a605d31116833f3d.92182da3c28c5193c0e36b4a68b5e896.
connexion_time matched schema specifique
29/06/2021 16:41:44 :: new metric b6f742865f074584a605d31116833f3d.92182da3c28c5193c0e36b4a68b5e896.
connexion_time matched aggregation schema default_average
29/06/2021 16:41:44 :: creating database file /opt/graphite/storage/whisper/b6f742865f074584a605d31116833f3d
/92182da3c28c5193c0e36b4a68b5e896/connexion_time.wsp (archive=[(60, 60), (300, 24), (3600, 720), (86400,
365), (518400, 182)] xff=0.01 agg=average)
```

À une métrique en particulier

Shinken ne permet pas de définir une rétention appliquée à une métrique uniquement.

Rétentions : comment définir les précisions de stockage des métriques

Une ligne de rétentions peut définir plusieurs rétentions. Chaque rétention à le format "**précision:période de couverture**", séparée par une virgule.

Les unités possibles pour la précision et la période de couverture sont :

- s : seconde
- m : minute
- h : heure
- d : jour
- w : semaine
- y : année

> Ne pas mettre une précision en dessous d'une minute.

Exemple de rétentions :

```
[apache_busyWorkers]
pattern = *\..*\.cpu$
retentions = 60s:7d,5m:21d,15m:5y
```

Ce schéma stocke une donnée par minute pendant une semaine, puis une donnée toutes les cinq minutes sur 21 jours et enfin une donnée toutes les 15 minutes durant 5 ans.

Lorsque l'on définit une ligne de rétentions, il faut suivre deux règles :

- définir les rétentions de la plus précise à la moins précise :
  - 15m:5y,5m:21d,60s:7d est incorrect,
  - 60s:7d,5m:21d,15m:5y est correct
- Pour garantir une agrégation des données de précision plus élevée vers celles de précision plus faible, une précision doit être un multiple exact de la précision précédente.
  - Une précision de 60 secondes **peut être suivie** d'une de 300 secondes (*5 minutes*).
  - Une précision de 180 secondes **ne peut pas être suivie** d'une précision de 600 secondes (*600 ne pouvant pas être divisible pas 180*)

Rétentions livrées par défaut à partir de la v02.06.02

La précision livrée par défaut depuis la 02.06.02 est la suivante :

```
[everything_1min]
pattern = .*
retentions = 60s:7d,300s:30d,1800s:180d,86400s:547d
```

Va appliquer pour tous les métriques la règle de stockage suivante :

- un point toutes les 60s, sur 7 jours
- un point toutes les 5min, sur 30 jours
- un point par 30min, sur 3 mois
- un point par jour, sur 1 an et demi

À noter qu'un point occupe 12 octets:

- 4 pour la date (*taille d'un int32*)
- 8 pour la valeur (*taille d'un double64*)

Ici un fichier créé avec cette rétention aura donc la taille :

- 60s:7d:
  - nombres de points: 7 j * 86400s / 60s = 10080 points
  - taille= 12o * 10080 = 120ko
- 300s:30d:
  - nombre de points : 30 j*86400 / 300s = 8640 points
  - taille = 12o * 8640 = 101Ko
- 1800s:180d:
  - nombre de points : 180 j * 86400 / 1800s = 8640points
  - taille = 12o * 8640 = 101Ko
- 86400s:547d
  - nombre de points : 547 j * 86400 / 86400s = 547 points
  - taille = 12o * 547 = 6Ko

Soit une taille totale de **328Ko**, par métrique présente sur le disque (*le fichier .wsp*).

> **La précision n'est pas modifiée lors d'une mise à jour**
>
> La précision n'est pas modifiée lors d'une mise à jour de Shinken Enterprise.

Rétentions livrées par défaut avant la v02.06.02

La précision livrée par défaut avant la 02.06.02 était la suivante :

```
[everything_1min]
pattern = .*
retentions = 60s:7d,300s:30d,3600s:180d,86400s:650d
```

Va appliquer pour tous les métriques la règle de stockage suivante:

- un point toutes les 60s, sur 7 jours
- un point toutes les 5min, sur 30 jours
- un point par heure, sur 6 mois
- un point par jour, sur 2 ans

À noter qu'un point occupe 12 octets:

- 4 pour la date (*taille d'un int32*)
- 8 pour la valeur (*taille d'un double64*)

Ici un fichier créé avec cette rétention aura donc la taille :

- 60s:7d:
  - nombres de points: 7 j * 86400s / 60s = 10080 points
  - taille= 12o * 10080 = 120ko
- 300s:30d:
  - nombre de points : 30 j*86400 / 300s = 8640 points
  - taille = 12o * 8640 = 103Ko
- 3600s:180d:
  - nombre de points : 180 j * 86400 / 3600s = 4320points
  - taille = 12o * 4320 = 52Ko
- 86400s:650d

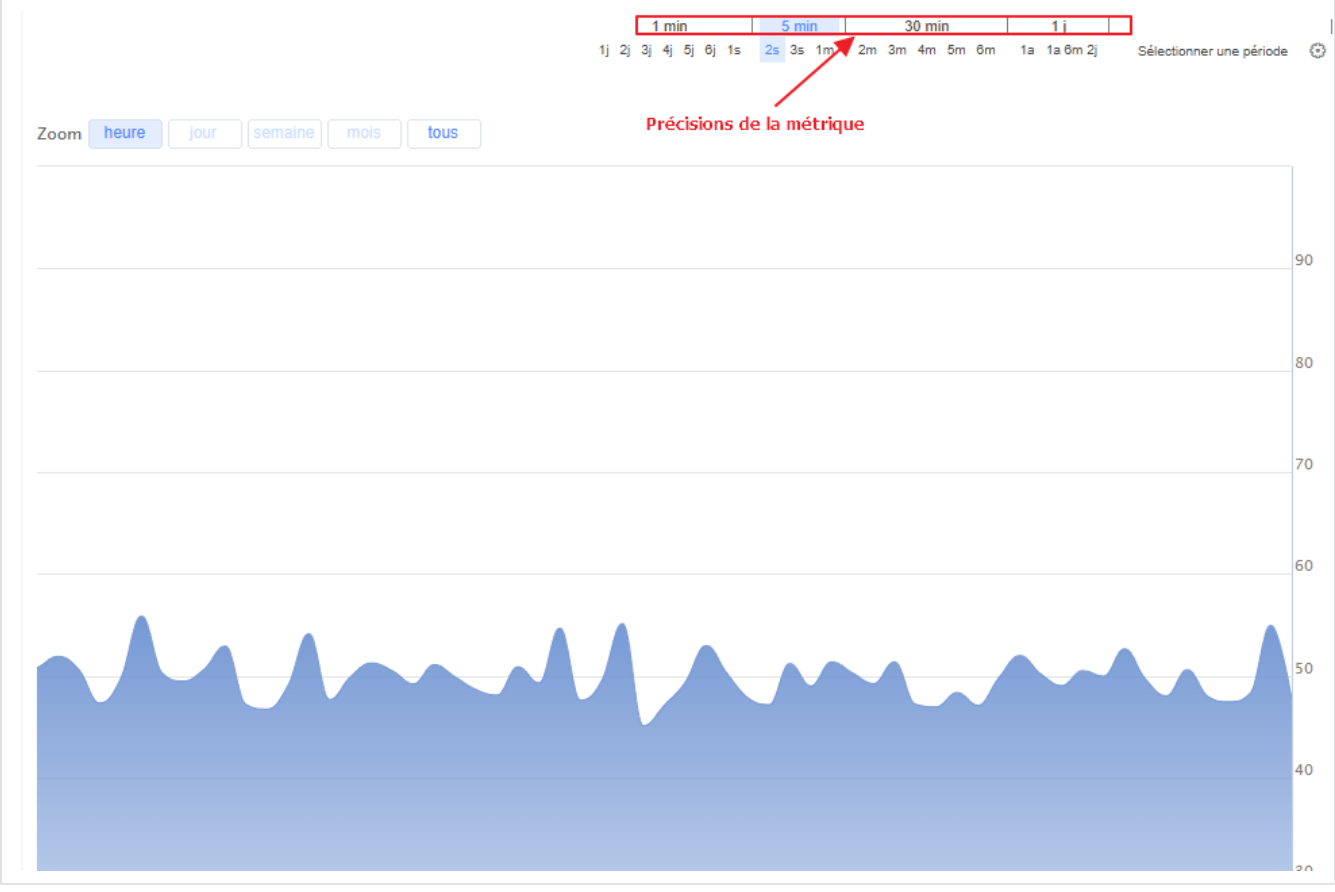
- nombre de points : 650 j * 86400 / 86400s = 650 points
- taille = 12o * 650 = 8Ko

Soit une taille totale de **283Ko** par métrique présente sur le disque (*le fichier .wsp*).

> **La précision n'est pas modifiée lors d'une mise à jour**
>
> La précision n'est pas modifiée lors d'une mise à jour de Shinken Enterprise.

Visualiser les précisions dans l'interface de Visualisation

Lorsque l'on affiche les métriques dans l'Interface de Visualisation, les précisons des données de la métrique apparaitront en haut à droite (*voir la page* Onglet Graphiques *pour plus d'information sur l'affichage des métriques dans l'Interface de Visualisation*).



Modifier la précision sur tous les éléments d'une installation de Shinken en cours de fonctionnement

Script de migration massif

Pour modifier la précision des métriques sans perdre les anciennes valeurs, utiliser le script ***whisper-resize.py***

Dans un premier temps, modifier les retentions dans le fichier */opt/graphite/conf/storage-schemas.conf*

Pour que le paramètre soit pris en compte, il faut redémarrer carbon-cache :

```
service carbon-cache restart
```

Une fois *carbon-cache* redémarré,

- créer le script *mass_modif_retention_graphite.sh* qui va modifier tous les fichiers .wsp présent sur la machine :

```
#!/bin/bash

FOLDER="/opt/graphite/storage/whisper"
RETENTION="XXXXXX"

for i in $(find $FOLDER/$1 -iname "*.wsp"); do
    if [ -a $i ];
        then
        /usr/bin/whisper-resize.py $FOLDER/$i $RETENTION;
        /usr/bin/chown apache:apache $FOLDER/$i
    fi;
done
```

Il faut remplacer la variable RETENTION par celle mise dans le fichier storage-schemas.conf avant exécution du script

Rendre le script exécutable

```
chmod +x mass_modif_retention_graphite.sh
```

Et le lancer en mettant une faible priorité tant au CPU qu'à la consommation disque :

```
ionice nice ./mass_modif_retention_graphite.sh
```

Suppression des anciennes données après migration

Le script va créer pour chaque fichier converti un fichier .bak qui est une sauvegarde de l'ancien fichier, ce qui signifie que la taille nécessaire à cette conversion va nécessiter beaucoup d'espace disque.

- Il est possible d'annuler la création de ce fichier en ajoutant le paramètre "–nobackup" au script. L'ancien fichier sera supprimé uniquement en cas de succès de la conversion.
- Dans le script, remplacer cette ligne :

```
/usr/bin/whisper-resize.py $FOLDER/$i $RETENTION;
```

Par cette ligne :

```
/usr/bin/whisper-resize.py $FOLDER/$i $RETENTION --nobackup;
```

En cas d'erreur

Erreur dans le format du fichier storage-schemas.conf

Lorsque la configuration du fichier **storage-schemas.conf** est incorrect, le démon carbon-cache échouera à démarrer et affichera un message d'erreur Python.

Exemple d'erreur provoquée par un fichier de configuration incorrect :

```
 service carbon-cache restart
Stopping carbon-cache...                                   [  OK  ]
Starting carbon-cache...Traceback (most recent call last):
  File "/opt/graphite/bin/carbon-cache.py", line 32, in <module>
    run_twistd_plugin(__file__)
  File "/opt/graphite/lib/carbon/util.py", line 140, in run_twistd_plugin
    runApp(config)
  File "/opt/shinken/shinken_venv_graphite_redhat8/lib64/python3.6/site-packages/twisted/scripts/twistd.py",
line 29, in runApp
    runner.run()
  File "/opt/shinken/shinken_venv_graphite_redhat8/lib64/python3.6/site-packages/twisted/application/app.
py", line 370, in run
    self.application = self.createOrGetApplication()
  File "/opt/shinken/shinken_venv_graphite_redhat8/lib64/python3.6/site-packages/twisted/application/app.
py", line 432, in createOrGetApplication
    ser = plg.makeService(self.config.subOptions)
  File "/opt/graphite/lib/twisted/plugins/carbon_cache_plugin.py", line 21, in makeService
    return service.createCacheService(options)
  File "/opt/graphite/lib/carbon/service.py", line 114, in createCacheService
    setupPipeline(['write'], root_service, settings)
  File "/opt/graphite/lib/carbon/service.py", line 89, in setupPipeline
    setupWriterProcessor(root_service, settings)
  File "/opt/graphite/lib/carbon/service.py", line 198, in setupWriterProcessor
    from carbon.writer import WriterService
  File "/opt/graphite/lib/carbon/writer.py", line 35, in <module>
    SCHEMAS = loadStorageSchemas()
  File "/opt/graphite/lib/carbon/storage.py", line 81, in loadStorageSchemas
    config.read(STORAGE_SCHEMAS_CONFIG)
  File "/opt/graphite/lib/carbon/conf.py", line 157, in read
    result = ConfigParser.read(self, path)
  File "/usr/lib64/python3.6/configparser.py", line 697, in read
    self._read(fp, filename)
  File "/usr/lib64/python3.6/configparser.py", line 1092, in _read
    fpname, lineno)
configparser.DuplicateOptionError: While reading from '/opt/graphite/conf/storage-schemas.conf' [line 23]:
option 'pattern' in section 'specifique' already exists
```

Erreur dans le format des précisions de rétention

```
service carbon-cache restart
Stopping carbon-cache...                                   [FAILED]
Starting carbon-cache...'too many values to unpack (expected 2) in section [Everything_1Min] in /opt/graphite
/conf/storage-schemas.conf'
```