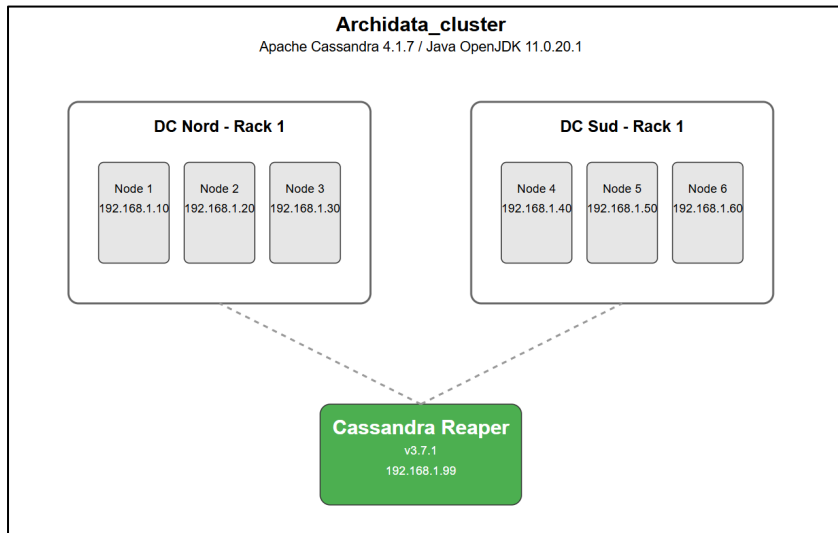


Architecture Cassandra - Archidta_cluster :

Cluster composé de 6 nœuds Apache Cassandra 4.1.7 répartis sur deux datacenters (Nord et Sud). Chaque DC contient 3 nœuds dans un unique rack (rack1).

L'ensemble du cluster s'appuie sur OpenJDK 11.0.20.1 LTS.

La maintenance et la réparation des données sont gérées via Cassandra Reaper 3.7.1, déployé sur une machine dédiée (192.168.1.99).



```
[node6@node6PC conf]$ nodetool -u cassandra -pw cassandra status
Datacenter: dc_nord
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens   Owns (effective)  Host ID                               Rack
UN 192.168.1.30  176,97 KiB  16       28,3%             283f0fbe-e6af-424e-8fd0-b9590d8d42d7 rack1
UN 192.168.1.20  226,11 KiB  16       35,0%             01590d8c-94e6-4b65-889a-e16fbeafceef rack1
UN 192.168.1.10  220,87 KiB  16       37,3%             98093374-2308-4c93-96cc-da81f6e26543 rack1

Datacenter: dc_sud
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens   Owns (effective)  Host ID                               Rack
UN 192.168.1.60  99,38 KiB  16       34,1%             6e81be4f-3c7c-4cad-b486-dfc68d0544b7 rack1
UN 192.168.1.50  176,99 KiB  16       35,7%             ae1f0552-4ad0-4598-9021-d44bbdce237c rack1
UN 192.168.1.40  177,01 KiB  16       29,7%             fb75ae34-a34b-4dd6-9f43-aa87f58dddb5 rack1

[node6@node6PC conf]$
```

Création d'un KEYSPACE pour les tests avec cassandra-stress :

```
cassandra-stress write n=100000 -node 192.168.1.60 -schema
"replication(strategy=NetworkTopologyStrategy, dc_nord=1, dc_sud=2)" -rate threads=50 -log
interval=5
```

Planifier et suivre les réparations via l'interface web de Cassandra-Reaper

Configuration initiale :

```
# See a bit more complete example at:
# src/server/src/test/resources/cassandra-reaper.yaml
segmentCountPerNode: 64
repairParallelism: DATACENTER_AWARE
repairIntensity: 0.9
scheduleDaysBetween: 7
repairRunThreadCount: 15
hangingRepairTimeoutMins: 30
storageType: cassandra
enableCrossOrigin: true
incrementalRepair: false
blacklistTwcsTables: true
enableDynamicSeedList: true
repairManagerSchedulingIntervalSeconds: 10
activateQueryLogger: false
jmxConnectionTimeoutInSeconds: 5
useAddressTranslator: false
maxParallelRepairs: 2
# purgeRecordsAfterInDays: 30
# numberOfRunsToKeepPerUnit: 10
```

```
cassandra:
  clusterName: "archidata_cluster"
  contactPoints: ["192.168.1.10", "192.168.1.30", "192.168.1.60"]
  keyspace: reaper_db_demo
  loadBalancingPolicy:
    type: tokenAware
    shuffleReplicas: true
  subPolicy:
    type: dcAwareRoundRobin
    localDC:
      usedHostsPerRemoteDC: 0
    allowRemoteDCsForLocalConsistencyLevel: false
```

Démarrage

```
[reaper@reaperPC configs]$ sudo systemctl start cassandra-reaper
[reaper@reaperPC configs]$ sudo systemctl status cassandra-reaper
● cassandra-reaper.service - Reaper for Apache Cassandra
  Loaded: loaded (/usr/lib/systemd/system/cassandra-reaper.service; disabled; preset: disabled)
  Active: active (running) since Fri 2025-02-07 22:07:03 CET; 4s ago
    Docs: http://cassandra-reaper.io/
  Main PID: 3533 (java)
    Tasks: 22 (Limit: 22376)
   Memory: 200.7M
      CPU: 4.392s
  CGroup: /system.slice/cassandra-reaper.service
          └─3533 java -ea -Xms2G -Xmx2G -Djava.net.preferIPv4Stack=true -cp ./cassandra-reaper-3.7.1.jar io.cassandra-reaper.ReaperApplication server /etc/cassandra-reaper/

févr. 07 22:07:03 reaperPC systemd[1]: Started Reaper for Apache Cassandra.
févr. 07 22:07:03 reaperPC cassandra-reaper[3533]: Using reaper in target
```

Accès à l'interface web

192.168.1.99:8080/webui/login.html

Reaper for Apache Cassandra

Username:

Password:

Remember Me ☐

Login

← → ↺ 192.168.1.99:8080/webui/index.html ☆ 📄 ⚙️ 🗑️ 📧 ☰

CassandraReaper 3.7.1 Switch theme ▾ Report a bug

- Clusters
- Schedules
- Repairs
- Snapshots
- Live Diagnostic beta
- Logout

Cluster

Add Cluster ▾

Filter:

archidata_cluster

Nodes: 6	dc_nord 1.7 MB	dc_sud 1.5 MB
Total load: 3.3 MB	rack1 1.7 MB	rack1 1.5 MB
Running repairs: 0		

[Forget cluster](#) [info](#)

Le **Repair Type** (type de réparation) dans Apache Cassandra Reaper désigne la méthode utilisée pour réparer les données dans un cluster Cassandra. La réparation est essentielle pour maintenir la cohérence des données et s'assurer que toutes les répliques d'un même morceau de données sont synchronisées.

Voici les principaux types de réparations disponibles dans Cassandra :

Types de Réparation

1. Full Repair

- **Description** : Ce type de réparation vérifie et répare tous les segments de données dans une table. Il s'assure que toutes les répliques de chaque partition sont synchronisées.
- **Exemple** : Utilisé lorsque des données ont été modifiées ou lorsque des nœuds ont été ajoutés ou retirés du cluster, nécessitant une vérification complète de la cohérence des données.
- **Utilisation** : Recommandé pour les tables avec des données critiques où la cohérence est primordiale.

2. Incremental Repair

- **Description** : Ce type de réparation ne traite que les données qui ont été modifiées depuis la dernière réparation. Cela permet de réduire le temps et les ressources nécessaires pour effectuer la réparation.
- **Exemple** : Utilisé dans des environnements où les données changent fréquemment, permettant de ne réparer que les partitions affectées.
- **Utilisation** : Idéal pour les clusters en production où la minimisation de l'impact sur les performances est importante.

3. Subrange Repair

- **Description** : Ce type de réparation se concentre sur une sous-plage spécifique de données, plutôt que sur l'ensemble de la table. Cela est utile pour les tables très volumineuses ou lorsque seuls certains segments nécessitent une attention.
- **Exemple** : Utilisé lorsqu'une partie spécifique de la table a été identifiée comme ayant des incohérences ou des problèmes.
- **Utilisation** : Pratique pour des réparations ciblées, permettant de limiter l'impact sur le reste du cluster.

4. Repair with Blacklist

- **Description** : Ce type de réparation exclut certains nœuds de la réparation, ce qui peut être utile si des nœuds sont connus pour avoir des problèmes ou si l'on souhaite éviter de les solliciter pendant la réparation.
- **Exemple** : Utilisé lorsqu'un nœud est en maintenance ou rencontre des problèmes de performance.
- **Utilisation** : Permet de gérer la réparation de manière plus fine, en évitant les nœuds problématiques.

Conclusion

Le choix du type de réparation dépend des besoins spécifiques du cluster Cassandra, de la taille des données, de la fréquence des modifications et des ressources disponibles. Une bonne stratégie de réparation est cruciale pour maintenir la santé et la performance d'un système Cassandra. Les administrateurs doivent évaluer régulièrement leurs pratiques de réparation pour s'assurer qu'elles répondent aux exigences de cohérence et de performance de leur environnement.

Options de Configuration pour la Réparation

1. Tables

- **Description** : Permet de spécifier les tables sur lesquelles la réparation doit être effectuée. Cela peut inclure une ou plusieurs tables dans le keyspace.
- **Utilisation** : Si vous souhaitez réparer uniquement certaines tables, vous pouvez les ajouter ici.

2. Blacklist

- **Description** : Permet d'ajouter des tables à une liste noire, ce qui signifie qu'elles seront exclues du processus de réparation.
- **Utilisation** : Utile pour éviter de réparer des tables qui ne nécessitent pas d'attention ou qui sont en cours d'utilisation intensive.

3. Nodes

- **Description** : Permet d'ajouter des nœuds spécifiques au processus de réparation. Vous pouvez cibler des nœuds particuliers dans le cluster.
- **Utilisation** : Pratique si vous souhaitez effectuer la réparation sur des nœuds spécifiques ou exclure certains nœuds du processus.

4. Datacenters

- **Description** : Permet de spécifier les centres de données sur lesquels la réparation doit être effectuée. Cela est particulièrement pertinent dans les configurations multi-datacenters.
- **Utilisation** : Utile pour cibler des réparations dans des centres de données spécifiques, par exemple, si un datacenter a été récemment ajouté ou retiré.

5. Segments per node

- **Description** : Définit le nombre de segments que chaque nœud doit traiter lors de la réparation.
- **Utilisation** : Permet de contrôler la charge de travail sur chaque nœud, ce qui peut être important pour éviter la surcharge.

6. Parallelism

- **Description** : Sélectionne le niveau de parallélisme pour la réparation. Cela peut inclure des options comme PARALLEL, SERIAL, ou DATACENTER_AWARE.
 - **PARALLEL** : Répare plusieurs segments en même temps.
 - **SERIAL** : Répare un segment à la fois.
 - **DATACENTER_AWARE** : Répare en tenant compte des répliques dans le même centre de données.
- **Utilisation** : Le choix du parallélisme peut affecter les performances et la durée de la réparation.

7. Repair Intensity

- **Description** : Indique le niveau d'intensité de la réparation, généralement exprimé en pourcentage. Cela détermine combien de données seront réparées par rapport à l'ensemble.
- **Utilisation** : Une intensité plus élevée peut entraîner une utilisation accrue des ressources, tandis qu'une intensité plus faible peut prolonger le temps de réparation.

8. Repair Type

- **Description** : Sélectionne le type de réparation à effectuer (par exemple, Full, Subrange).
- **Utilisation** : Choisir le type approprié en fonction des besoins de réparation spécifiques.

9. Repair Threads

- **Description** : Définit le nombre de threads à utiliser pour le processus de réparation.
- **Utilisation** : Plus de threads peuvent améliorer la vitesse de réparation, mais peuvent également augmenter la charge sur le système.

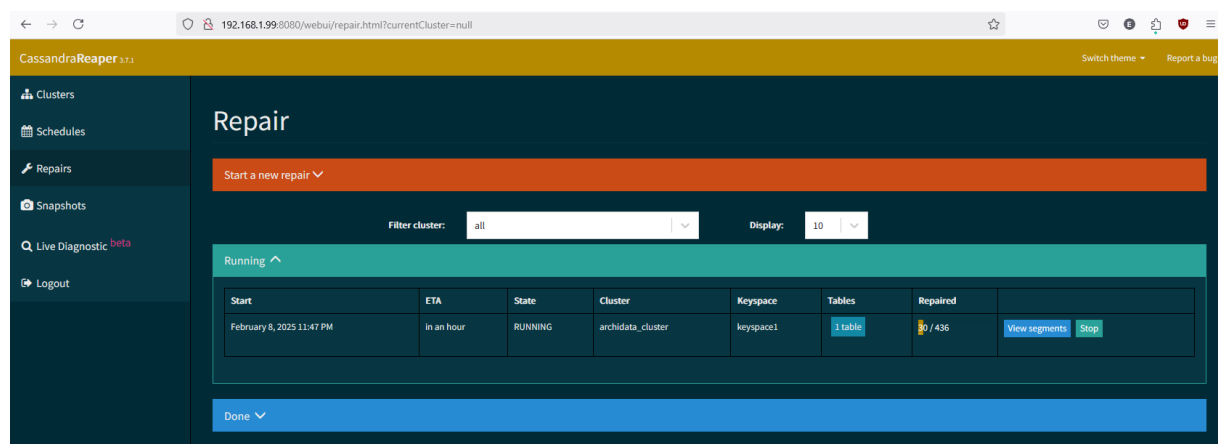
10. Segment Timeout

- **Description** : Indique le temps limite (en minutes) pour le traitement d'un segment avant qu'il ne soit considéré comme échoué.
- **Utilisation** : Cela permet de gérer les situations où un segment prend trop de temps à être réparé, garantissant ainsi que le processus de réparation reste efficace.

Conclusion

Ces options permettent aux administrateurs de personnaliser le processus de réparation en fonction des besoins spécifiques de leur environnement Cassandra. Une configuration appropriée peut améliorer la performance, la cohérence des données et réduire l'impact sur le cluster pendant les opérations de maintenance.

Un repair en cours :



The screenshot shows the CassandraReaper web interface. The top navigation bar includes 'Clusters', 'Schedules', 'Repairs', 'Snapshots', 'Live Diagnostic beta', and 'Logout'. The main content area is titled 'Repair' and features a 'Start a new repair' button. Below this, there are filters for 'Filter cluster:' (set to 'all') and 'Display:' (set to '10'). A table titled 'Running' shows the details of a repair in progress. The table has columns for Start, ETA, State, Cluster, Keyspace, Tables, Repaired, and buttons for 'View segments' and 'Stop'.

Start	ETA	State	Cluster	Keyspace	Tables	Repaired	
February 8, 2025 11:47 PM	In an hour	RUNNING	archidata_cluster	keyspace1	1 table	0 / 436	View segments Stop

Segments

a013fbf4-e66e-11ef-bafa-593ac128ef5d	-240127874779899680	-197751069762977500	0	DONE	192.168.1.50	192.168.1.30 (dc_nord) 192.168.1.50 (dc_sud)	192.168.1.40 (dc_sud)	February 8, 2025 11:47 PM	February 8, 2025 11:47 PM	a few seconds	Replay
a007c5f2-e66e-11ef-bafa-593ac128ef5d	-8743144229993946000	-8702921116670552000	0	DONE	192.168.1.60	192.168.1.40 (dc_sud) 192.168.1.60 (dc_sud)	192.168.1.20 (dc_nord)	February 8, 2025 11:47 PM	February 8, 2025 11:47 PM	a few seconds	Replay
a01115c5-e66e-11ef-bafa-593ac128ef5d	-2728839975366430700	-2682697583965785000	0	DONE	192.168.1.20	192.168.1.40 (dc_sud) 192.168.1.60 (dc_sud)	192.168.1.20 (dc_nord)	February 8, 2025 11:47 PM	February 8, 2025 11:47 PM	a few seconds	Replay

Not started (403)

ID	Start token	End token	Fail count	Replicas	State
a0061940-e66e-11ef-bafa-593ac128ef5d	-9107090699948490000	-9101132382500738000	0	192.168.1.40 (dc_sud) 192.168.1.10 (dc_nord) 192.168.1.50 (dc_sud)	NOT_STARTED
a00778d0-e66e-11ef-bafa-593ac128ef5d	-9101132382500738000	-9076480589825203000	0	192.168.1.40 (dc_sud) 192.168.1.10 (dc_nord) 192.168.1.60 (dc_sud)	NOT_STARTED
a0079fe0-e66e-11ef-bafa-593ac128ef5d	-9076480589825203000	-9051828797149669000	0	192.168.1.40 (dc_sud) 192.168.1.10 (dc_nord) 192.168.1.60 (dc_sud)	NOT_STARTED
a0079fe1-e66e-11ef-bafa-593ac128ef5d	-9051828797149669000	-9007085221510948000	0	192.168.1.40 (dc_sud) 192.168.1.20 (dc_nord) 192.168.1.60 (dc_sud)	NOT_STARTED
a0079fe2-e66e-11ef-bafa-593ac128ef5d	-9007085221510948000	-8962341645872226000	0	192.168.1.40 (dc_sud) 192.168.1.20 (dc_nord) 192.168.1.60 (dc_sud)	NOT_STARTED

```
resolver-dns-4.1.58.Final.10b03e6, netty-resolver-dns-native-macos=netty-resolver-dns-native-macos-4.1.58.Final.10b03e65f1, netty-transport=netty-transport-4.1.58.Final.10b03e6, netty-transport-native-epoll=netty-transport-native-epoll-4.1.58.Final.10b03e6, netty-transport-native-kqueue=netty-transport-native-kqueue-4.1.58.Final.10b03e65f1, netty-transport-native-unix-common=netty-transport-native-unix-common-4.1.58.Final.10b03e6, netty-transport-rxtx=netty-transport-rxtx-4.1.58.Final.10b03e6, netty-transport-sctp=netty-transport-sctp-4.1.58.Final.10b03e6, netty-transport-udt=netty-transport-udt-4.1.58.Final.10b03e6]
INFO [main] 2025-02-08 19:28:07,441 ConfigurationLoader.java:62 - Configuration location: files:/opt/mcac/config/metric-collector.yaml
INFO [main] 2025-02-08 19:28:07,655 CollectdController.java:469 - /opt/mcac/config/collectd.conf.tpl
nodep@nodeP-PC [lib]$ grep -i "Starting DataStax Metric Collector for Apache Cassandra" /var/log/cassandra/debug.log
INFO [main] 2025-02-08 19:28:07,399 CassandraDaemonInterceptor.java:60 - Starting DataStax Metric Collector for Apache Cassandra 0.3.5
nodep@nodeP-PC [lib]$
```

Installation Prometheus

← → ↺

🔒 192.168.1.99:9090/targets?search=

Prometheus

Alerts Graph Status ▾ Help

Targets

All Unhealthy Collapse All

🔍 Filter by endpoint or labels

prometheus (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.1.99:8081/prometheusMetrics	UP	instance="192.168.1.99:8081" job="prometheus"	682.000ms ago	188.035ms	

Configuration prometheus / metrics cassandra & Ajout alertmanager

```
GNU nano 5.6.1 /etc/prometheus/prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        - "192.168.1.99:9093"

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "/etc/prometheus/first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: "reaper"

    metrics_path: '/prometheusMetrics'
    scrape_interval: 5s
    # scheme defaults to 'http'.
    static_configs:
      - targets: ["192.168.1.99:8081"]
    metric_relabel_configs:
      - source_labels: [__name__]
        regex: "io_cassandra_reaper_service_RepairRunner_segments(Done|Total)(.*)_(.*)_(.*)"
        target_label: cluster
        replacement: '${2}'
      - source_labels: [__name__]
        regex: "io_cassandra_reaper_service_RepairRunner_segments(Done|Total)(.*)_(.*)_(.*)"
        target_label: keyspace
        replacement: '${3}'
      - source_labels: [__name__]
        regex: "io_cassandra_reaper_service_RepairRunner_segments(Done|Total)(.*)_(.*)_(.*)"
        target_label: repair_id
        replacement: '${4}'
      - source_labels: [__name__]
        regex: "io_cassandra_reaper_service_RepairRunner_segments(Done|Total).*"
        target_label: __name__
        replacement: 'io_cassandra_reaper_service_RepairRunner_segments${1}'
```

Exemple alerte de cassandra-reaper sur prometheus

192.168.1.99:9090/alerts?search=

Prometheus Alerts Graph Status Help

Inactive (0) Pending (0) Firing (1) Filter by name or labels Show annotations

/etc/prometheus/first_rules.yml > cassandra-reaper-alerts firing (1)

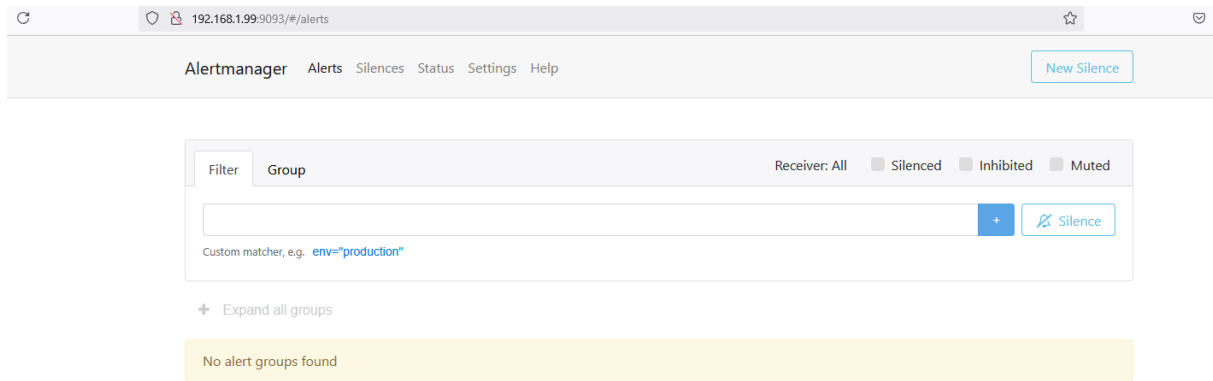
▼ CassandraReaperDown (1 active)

name: CassandraReaperDown
expr: up{job="reaper"} == 0
for: 30s
labels:
severity: critical
annotations:
description: Le service Reaper n'est plus accessible depuis Prometheus.
summary: Cassandra Reaper est down

Labels	State	Active Since	Value
alertname=CassandraReaperDown instance=192.168.1.99:8081 job=reaper severity=critical	FIRING	2025-02-14T21:46:18.141835464Z	0

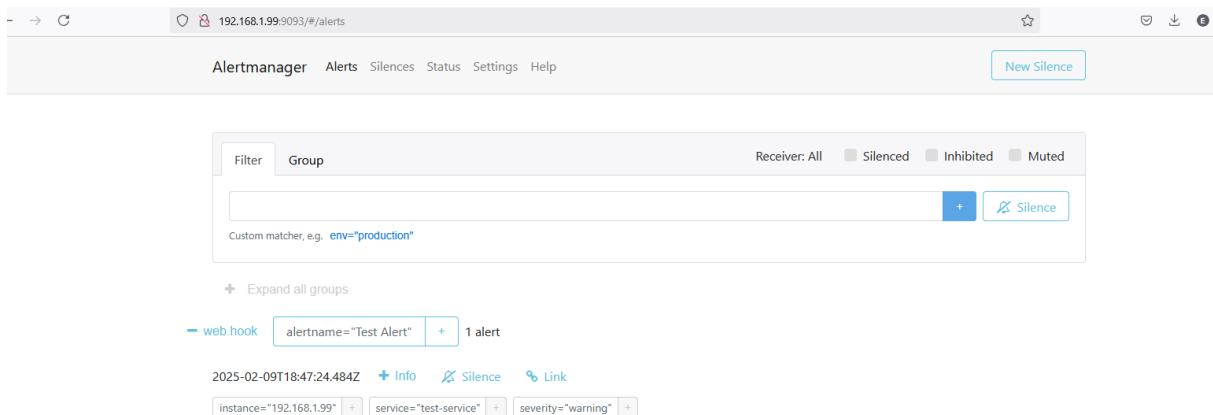
Installation AlertManager

<https://developer.couchbase.com/tutorial-configure-alertmanager>



Tester son fonctionnement

```
curl -i \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-X POST "http://192.168.1.99:9093/api/v2/alerts" \
-d '[
{
  "status": "firing",
  "labels": {
    "alertname": "Test Alert",
    "service": "test-service",
    "severity": "warning",
    "instance": "192.168.1.99"
  },
  "annotations": {
    "summary": "High latency is high!"
  }
}]'
```



Configuration fichier alertmanager.yml pour l'envoi des alertes par email

```

global:
  resolve_timeout: 5m
  smtp_smarthost: 'smtp.office365.com:587'
  smtp_from: 'Mohammed.elmalki@ynov.com' # Adresse qui enverra les alertes
  smtp_auth_username: 'Mohammed.elmalki@ynov.com'
  smtp_auth_password: 'XXXXXXXXXX'
  smtp_require_tls: true

route:
  group_by: ['alertname']
  group_wait: 30s
  group_interval: 1m
  repeat_interval: 1h
  receiver: 'Nouredine'

receivers:
- name: 'XXXXXXXXXX'
  email_configs:
  - to: 'XXXXXXXXXX@ynov.com'
    send_resolved: true

inhibit_rules:
- source_match:
  severity: 'critical'
  target_match:
  severity: 'warning'
  equal: ['alertname', 'dev', 'instance']

```

pour le teste il faut arreter reaper :

```
sudo systemctl stop cassandra-reaper.service
```

Exemple alerte Reçu par email :

[FIRING:1] CassandraReaperDown (192.168.1.99:8081 reaper critical) Boîte de réception x

Mohammed.elmalki@ynov.com 22:47 (il y a 8 minutes) ☆ ☺ ↶ ⋮

À moi ▾

1 alert for alertname=CassandraReaperDown

[View In Alertmanager](#)

[1] Firing

Labels

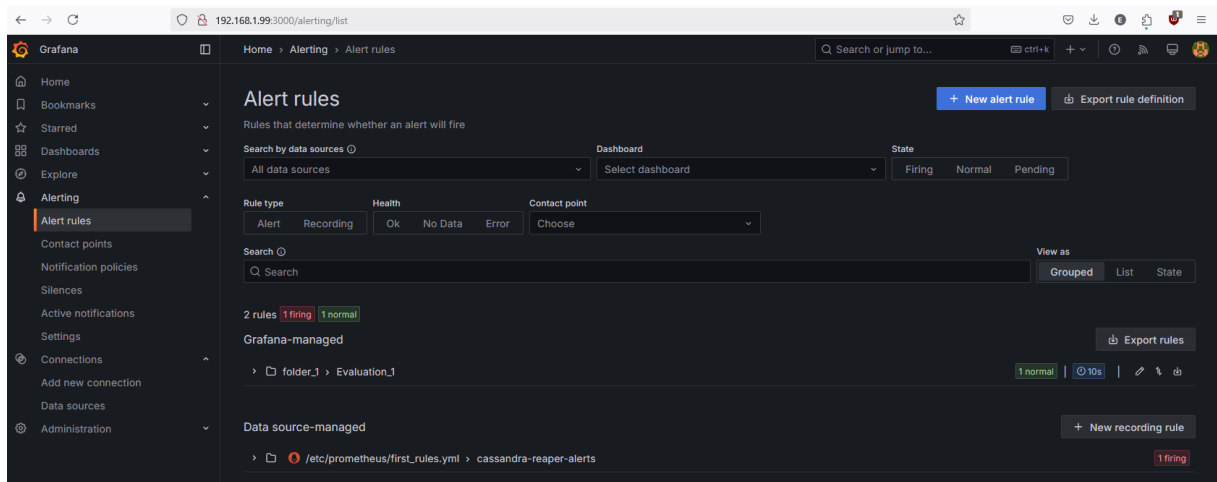
alertname = CassandraReaperDown
instance = [192.168.1.99:8081](#)
job = reaper
severity = critical

Annotations

description = Le service Reaper n'est plus accessible depuis Prometheus.
summary = Cassandra Reaper est down
[Source](#)

Sent by Alertmanager

Exemple d'Alerte créé via l'interface grafana



Exemple dashboard grafana

