

異なるバージョンの Elasticsearch ノードのクラスタリング

祖父江匠真

1 概要

今回は, Docker Compose を使用して, 異なるバージョンである 7.17.6 と 7.17.9 の Elasticsearch ノードをクラスタリングすることが可能かどうかを確認するために実施した実験について説明する.

クラスタの構築には, Docker Compose を使用した.

2 Docker とは

Docker は, アプリケーションをコンテナと呼ばれる隔離された環境で実行するためのツールである. コンテナは軽量で, システムリソースを共有し, それぞれが独自のファイルシステム, プロセス空間, メモリなどを持っている. これにより, 異なる環境で同じアプリケーションを一貫して実行できる.

3 Docker Compose とは

Docker Compose は, 複数のコンテナを定義し, 実行するためのツールである. これは YAML ファイルを使用して設定され, 複数のコンテナを使用したシステムの構築を単純化する.

4 実験セットアップ

4.1 全て同じバージョンの Elasticsearch を使用したクラスタ構成 (全ノード バージョン 7.17.9)

Listing 1 にクラスタを構成するのに使用した docker-compose.yml ファイルの内容を記載する.

Listing 1: 全て同じバージョンの Elasticsearch を使用したクラスタを構成する docker-compose.yml

```
1 | version: '2.2'
```

```

2 services:
3   es01:
4     image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
5     container_name: es01
6     environment:
7       - node.name=es01
8       - cluster.name=es-docker-cluster
9       - discovery.seed_hosts=es02,es03
10      - cluster.initial_master_nodes=es01,es02,es03
11      - bootstrap.memory_lock=true
12      - "ES_JAVA_OPTS=-Xms2048m,-Xmx2048m"
13     ulimits:
14       memlock:
15         soft: -1
16         hard: -1
17     volumes:
18       - data01:/usr/share/elasticsearch/data
19     ports:
20       - 9200:9200
21     networks:
22       - elastic
23   es02:
24     image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
25     container_name: es02
26     environment:
27       - node.name=es02
28       - cluster.name=es-docker-cluster
29       - discovery.seed_hosts=es01,es03
30       - cluster.initial_master_nodes=es01,es02,es03
31       - bootstrap.memory_lock=true
32       - "ES_JAVA_OPTS=-Xms2048m,-Xmx2048m"
33     ulimits:
34       memlock:
35         soft: -1
36         hard: -1
37     volumes:
38       - data02:/usr/share/elasticsearch/data
39     networks:
40       - elastic
41   es03:
42     image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
43     container_name: es03
44     environment:
45       - node.name=es03
46       - cluster.name=es-docker-cluster
47       - discovery.seed_hosts=es01,es02
48       - cluster.initial_master_nodes=es01,es02,es03

```

```

49     - bootstrap.memory_lock=true
50     - "ES_JAVA_OPTS=-Xms2048m-Xmx2048m"
51     ulimits:
52         memlock:
53             soft: -1
54             hard: -1
55     volumes:
56         - data03:/usr/share/elasticsearch/data
57     networks:
58         - elastic
59
60 volumes:
61     data01:
62         driver: local
63     data02:
64         driver: local
65     data03:
66         driver: local
67
68 networks:
69     elastic:
70         driver: bridge

```

Listing 1の docker-compose.yml ファイルで記述している内容について説明する。

サービスの定義

- **es01, es02, es03:** これらは Elasticsearch のノード（サーバー）である。各ノードは異なるコンテナとして定義されている。‘es01‘, ‘es02‘, ‘es03‘はそれぞれ異なるコンテナ名で, Elasticsearch の異なるインスタンスを実行する。

各ノードの設定

- **image:** 使用する Docker イメージ。ここでは Elasticsearch の 7.17.9 バージョンを使用している。
- **container_name:** コンテナに割り当てられる名前。
- **environment:** 環境変数の設定。Elasticsearch のクラスタ設定, メモリ設定などを含みます。
- **ulimits:** メモリロックの限界を設定。これにより Elasticsearch がメモリを効率的に利用できる。

- **volumes:** データ永続化のためのボリュームマウント。データをコンテナの外に保存する。
- **ports:** ホストマシンとコンテナ間のポートマッピング。例えば, '9200:9200' はホストの 9200 ポートをコンテナの 9200 ポートにマッピングする。
- **networks:** コンテナ間通信のためのネットワーク設定。ここでは 'elastic' ネットワークが使用されている。

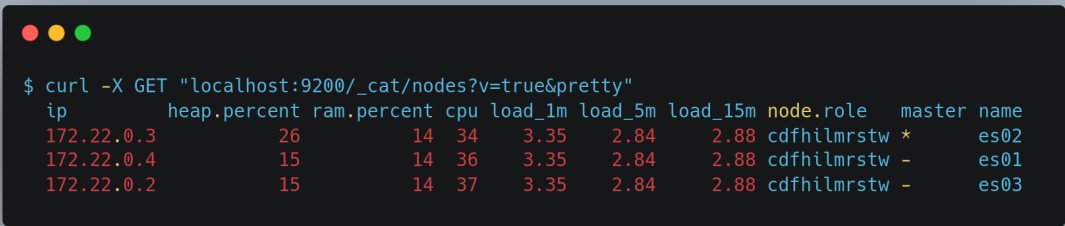
ボリュームとネットワークの設定

- **volumes:** 'data01', 'data02', 'data03' はデータの永続化に使用されるボリュームである。これによりコンテナが削除されてもデータが保持される。
- **networks:** 'elastic' ネットワークはブリッジドライバを使用している。これにより, 異なるコンテナが相互に通信できるようになる。

この設定により, Elasticsearch の 3 ノードを含むクラスタが Docker 上で動作するようセットアップされる。

クラスタの起動には, docker compose up -d コマンドを使用する。

docker compose up -d コマンドを実行した後, curl コマンドを使用してクラスタに参加しているノードを一覧表示した結果を図 1 に示す。



```
$ curl -X GET "localhost:9200/_cat/nodes?v=true&pretty"
ip             heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
172.22.0.3      26           14   34    3.35   2.84    2.88 cdfhilmrstw *   es02
172.22.0.4      15           14   36    3.35   2.84    2.88 cdfhilmrstw -   es01
172.22.0.2      15           14   37    3.35   2.84    2.88 cdfhilmrstw -   es03
```

図 1: クラスタに参加しているノードを一覧表示した結果

図 1 より, 3 つのノード ('es01', 'es02', 'es03') すべてが正常にクラスタに参加できていることが確認できる。

4.2 異なるバージョンの Elasticsearch を使用したクラスタ構成 (2 ノード バージョン 7.17.9, 1 ノード バージョン 7.17.6)

Listing 1 の docker-compose.yml を Listing 2 のように変更することで, es03 のノードで使用する Elasticsearch のバージョンを 7.17.9 から 7.17.6 に変更する.

Listing 2: Listing 1 の docker-compose.yml から変更を加えた箇所

```
1 version: '2.2'
2 services:
3   ...
4   es03:
5     image: docker.elastic.co/elasticsearch/elasticsearch:7.17.6
6     ...
7   ...
8   ...
```

変更後, docker compose up -d コマンドを実行してクラスタを起動する.

クラスタの起動後, curl コマンドを使用してクラスタに参加しているノードを一覧表示した結果を図 2 に示す.

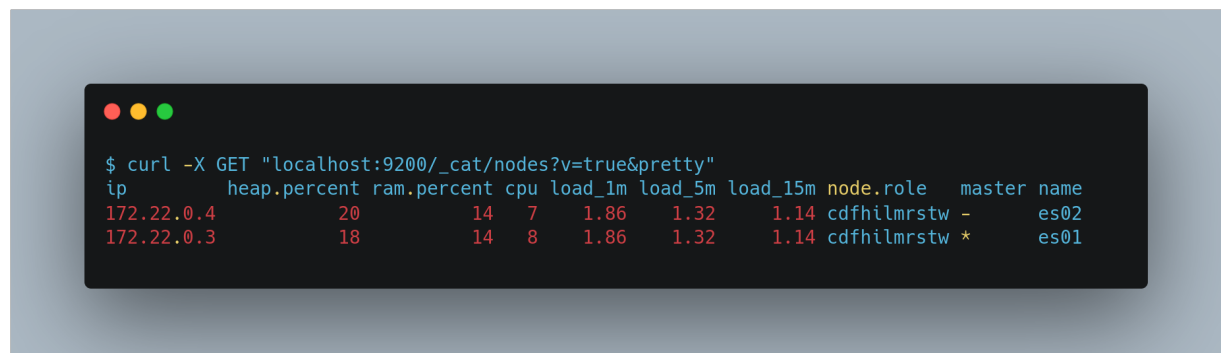


図 2: クラスタに参加しているノードを一覧表示した結果

図 2 より, バージョンが 7.17.9 である 2 つのノード ('es01', 'es02') のみが正常にクラスタに参加できていることが確認できる.

5 まとめ

今回は, Docker Compose を使用して, 異なるバージョンである 7.17.6 と 7.17.9 の Elasticsearch ノードをクラスタリングすることが可能かどうかを確認した.

今回の実験により, Docker Compose を使用した Elasticsearch ノードのクラスタリングは, 全てのノードが同じバージョンであれば行えることが実証された. しか

し、異なるマイナーバージョン（今回は7.17.6と7.17.9）のノードをクラスタ化しようとする、古いバージョンのノードはクラスタに参加できなかった。

そのため、サーバーゾーンでのリサイクル館の測定データを保存している ElasticSearch をバージョンアップする必要があり、今回は ElasticSearch のバージョンアップ方法について調査した結果を報告する。

参考文献

- [1] Elasticsearch B.V., "Install Elasticsearch with Docker — Elasticsearch Guide [7.17] — Elastic ", <https://www.elastic.co/guide/en/elasticsearch/reference/7.17/docker.html>, 参照 Nov 17, 2023.