

# 学位論文

## 学内のElasticsearchシステムのデータ移行と 冗長化

提出年月日 令和 4 年 X 月 X 日

改定日 令和 年 月 日

指導教員 都築 伸二 教授

入学年度 令和 6 年

学科名 電子情報工学専攻

論文提出者 祖父江 匠真

# 内容梗概

本論文は、筆者が愛媛大学大学院理工学研究科電子情報工学専攻電気電子工学コースに在学中に行った、太陽光発電の計測データの補正と Elasticsearch のデータ移行と冗長化についてまとめたものであり、以下の 5 章から構成されている。

## 第 1 章 緒論

本研究を行うに至った経緯及び、本研究の目的について述べている。

## 第 2 章 太陽光発電の計測データの補正

ここでは太陽光発電の計測データの補正について述べる。

## 第 3 章 学内ゾーンにおける Elasticsearch クラスタへのデータ移行

ここでは学内ゾーンにおける Elasticsearch クラスタへのデータ移行について述べる。

## 第 4 章 サーバーゾーンでのクラスタ構築

ここでは、サーバーゾーンでのクラスタ構築における仮想環境を使用した事前検証について述べている。

## 第 5 章 結論

本研究によって明らかになった事項や今後の研究課題について簡単にまとめている。

# 目次

内容梗概	I
第 1 章 緒論	1
第 2 章 太陽光発電の計測データの補正	2
2.1 節 緒言	2
2.2 節 太陽光発電の計測データの問題点について	2
2.3 節 日射量の計算式の導出	2
2.4 節 実測データと計算データの比較	4
2.5 節 相互相関によるずれ時間の特定	4
2.6 節 日射量の計算データの予測精度改善	5
2.7 節 pvlib の概要	5
2.8 節 実測データと pvlib を使って求めた計算データの比較	6
2.9 節 pvlib を用いて計算した日射量を使用した, 相互相関に よるずれ時間の特定	7
2.10 節 前処理の追加による相互相関の計算結果の改善	7
2.11 節 結言	9
第 3 章 学内ゾーンにおける Elasticsearch クラスタへのデータ移行	10
3.1 節 緒言	10
3.2 節 Elasticsearch の概要	10
3.3 節 kibana の概要	11
3.4 節 学内ゾーンで稼働している Elasticsearch システムの状況	12
3.5 節 データ移行対象の Elasticsearch インデックスについて	12
3.6 節 CO <sub>2</sub> データの移行手順について	12

3.6.1	データのエクスポート . . . . .	12
3.6.2	データの重複削除 . . . . .	13
3.6.3	データのインポート . . . . .	15
3.7 節	一度目のデータ移行で移行できなかった CO <sub>2</sub> データの 移行について . . . . .	15
3.8 節	kibana によるデータの可視化 . . . . .	16
3.9 節	LEAF の運行日誌に関するデータの移行について . . . .	17
3.10 節	LEAF の運行日誌に関するデータの移行手順について .	20
3.10.1	データのエクスポート . . . . .	20
3.10.2	データのインポート . . . . .	20
3.11 節	結言 . . . . .	20
第 4 章	サーバーゾーンでのクラスタ構築における仮想環境を使用した事 前検証 . . . . .	21
4.1 節	緒言 . . . . .	21
4.2 節	サーバーゾーンで稼働している Elasticsearch システム の状況 . . . . .	21
4.3 節	Docker とは . . . . .	22
4.3.1	コンテナとは . . . . .	22
4.3.2	Docker イメージとは . . . . .	23
4.4 節	Docker Compose とは . . . . .	23
4.5 節	検証環境のセットアップ . . . . .	23
4.5.1	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) . . .	23
4.5.2	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) . . . . .	28
4.6 節	異なる Elasticsearch クラスタへのノード参加検証 . . .	29
4.7 節	手順 . . . . .	30
4.7.1	単一ノードで稼働するクラスタ A の構築 . . . .	30
4.7.2	クラスタ B の構築 . . . . .	31

4.7.3 クラスタ B への参加試行 . . . . .	33
4.8 節 Elasticsearch のバージョンアップ . . . . .	35
4.9 節 バージョンアップ手順 . . . . .	35
4.9.1 インストール方法の特定 . . . . .	35
4.9.2 apt によるバージョンアップ . . . . .	36
4.10 節 kibana のバージョンアップ . . . . .	36
4.11 節 バージョンアップ後の動作確認 . . . . .	38
4.12 節 サーバーゾーンにおけるクラスタの構築 . . . . .	38
4.13 節 結言 . . . . .	38
<b>第 5 章 結論と今後の課題 . . . . .</b>	<b>39</b>
5.1 節 結論 . . . . .	39
5.2 節 今後の課題 . . . . .	39
<b>謝 辞 . . . . .</b>	<b>41</b>
<b>参考文献 . . . . .</b>	<b>42</b>

# 第1章

## 緒論

太陽光発電は、再生可能エネルギー源として世界中で注目されており、その効率的な運用と管理には正確な計測データが不可欠である。しかしながら、実際の計測データには様々な課題が存在する。特に、PCの内部時計のずれによる時間情報の不正確さは、データの信頼性を損なう重要な問題である。

本研究では、これらの問題に対処するため、太陽光発電の計測データの補正方法を提案する。具体的には、相互相関を用いて計測データの時間的ずれを特定し、`pvlib` ライブラリを活用して日射量の計算データの予測精度を向上させる。これにより、太陽光発電データの信頼性の向上と、データベースの効率的な管理を目指す。

本論文は以下の構成となっている。第2章では、太陽光発電の計測データの時間的ずれの特定に焦点を当て、相互相関を用いた手法を提案する。第3章では、`Elasticsearch` を用いたデータ管理と移行について詳述する。第4章では、仮想環境を使用したクラスタ構築の検証について述べ、特にバージョンの異なる `Elasticsearch` ノード間でのクラスタリングに関する検証を行う。さらに、`Elasticsearch` および `Kibana` のバージョンアップとその後の動作確認についても詳細に説明する。これらの章を通じて、太陽光発電データの精度と、`Elasticsearch` の最適化を目指す本研究の成果と課題を明らかにする。

## 第2章

# 太陽光発電の計測データの補正

### 2.1 節 緒言

本章では太陽光発電の計測データの補正について述べる。

### 2.2 節 太陽光発電の計測データの問題点について

CSV データなどで保存された太陽光発電の環境データはオフライン環境でファイル書き込みを行っているため、PC の内部時計がずれており、計測データの日時情報が正確な日時とは異なっている。

そこで相互相関を用いて、実測した日射量の時系列データと、計算式により求まる日射量の時系列データとの時間的遅延の秒数を検出することで、実測データの計測日時のずれ時間を特定する。

### 2.3 節 日射量の計算式の導出

任意の緯度経度、日時における日射量  $Q$  は、任意の緯度  $\phi$ 、経度  $\lambda$  の地点における任意の日時、太陽高度  $\alpha$  から求めることができる。

まず、次式により元旦からの通し日数  $dn$  に基いて定めた  $\theta$  を用いて、当該日

の太陽赤緯  $\delta$ , 地心太陽距離  $\frac{r}{r^*}$ , 均時差  $E_q$  をそれぞれ以下の式により求める.

$$\theta = \frac{2\pi(dn - 1)}{365} \quad (2.1)$$

$$\begin{aligned} \delta = & 0.006918 - 0.399912 \cos \theta + 0.070257 \sin \theta - 0.006758 \cos 2\theta \\ & + 0.000907 \sin 2\theta - 0.002697 \cos 3\theta + 0.001480 \sin 3\theta \end{aligned} \quad (2.2)$$

$$\begin{aligned} \frac{r}{r^*} & \quad (2.3) \\ = & \frac{1}{\sqrt{1.000110 + 0.034221 \cos \theta + 0.001280 \sin \theta + 0.000719 \cos 2\theta + 0.000077 \sin 2\theta}} \end{aligned}$$

$$\begin{aligned} E_q = & 0.000075 + 0.001868 \cos \theta - 0.032077 \sin \theta \\ & - 0.014615 \cos 2\theta - 0.040849 \sin 2\theta \end{aligned} \quad (2.4)$$

日本標準時間から, 太陽の時角  $h$  を求める.

$$h = \frac{(\text{日本標準時間} - 12)\pi}{12} + \text{標準子午線からの経度差} + E_q \quad (2.5)$$

$\delta, \phi, h$  の値が既知となったので  $\alpha$  は

$$\alpha = \arcsin(\sin \phi \sin \delta + \cos \phi \cos \delta \cos h) \quad (2.6)$$

として求めることができる.

最後に,  $Q$  を

$$Q = 1367 \left( \frac{r^*}{r} \right)^2 \sin \alpha \quad (2.7)$$

により求めることができる. また,  $1367\text{W}/\text{m}^2$  は太陽定数である.

式 (2.1) ~ 式 (2.7) を用いることで, 任意の緯度経度, 日時における日射量が求まる.



## 2.4 節 実測データと計算データの比較

Elasticsearch サーバーから取得した日射量データと、計算式から求めた日射量データをプロットしたものを図 2.1 に示す。図 2.1 は Elasticsearch サーバーから取得した 2022 年 6 月 2 日の日射量データと、リサイクル館の緯度経度と日付情報より求めた日射量の値をプロットしている。

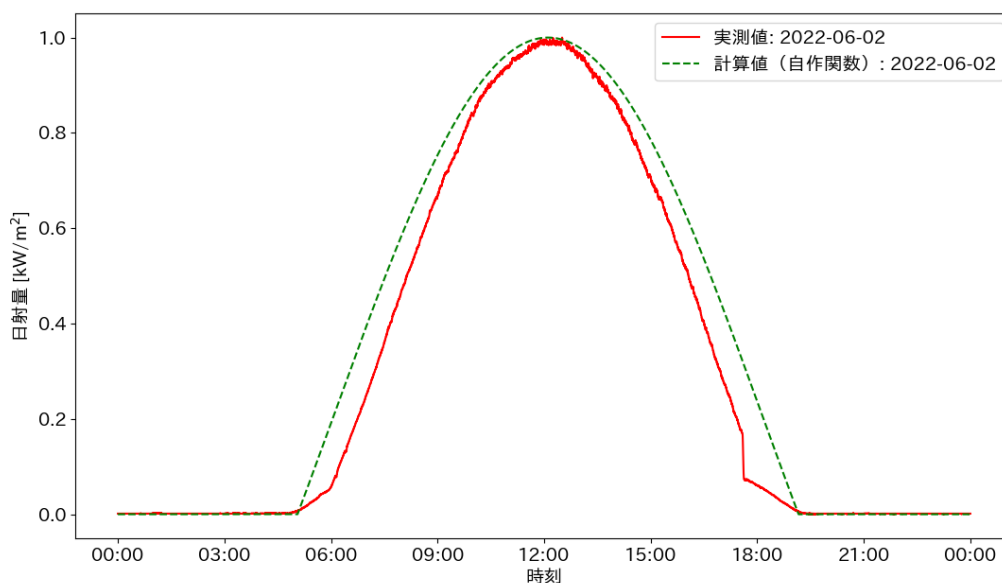


図 2.1 2022 年 6 月 2 日の日射量の実測データと計算データをプロットしたもの

## 2.5 節 相互相関によるずれ時間の特定

今回、相互相関の計算に使用する実測データの範囲を、2022 年 6 月 2 日 0 時 0 分から 2022 年 6 月 2 日 23 時 59 分まで期間とする。

2022 年 6 月 2 日を選定した理由として、図 2.1 より、2022 年 6 月 2 日の実測データの概形は計算データの概形と類似していたためである。

実測データの計測日時の情報をもとに、計算データを求め、これらの値を入力として相互相関を計算する。

相互相関の計算結果より、計算データの日時を実測データより 124 秒進めた際に、相互相関の値が最大となった。

しかし、今回使用した実測データには計測日時のずれは殆どないため、計算データの日時を進めていない際に実測データとの相関が最大となるのが正しい。

これは、日射量の計算式の予測精度が低いことが原因であると考えられる。

## 2.6 節 日射量の計算データの予測精度改善

日射量の計算データの予測精度を改善するため、式 (2.1) ~ 式 (2.7) を使った方法ではなく、pvlib というライブラリを使用して、任意の緯度経度と日時における日射量を求めて相互相関を計算する。

## 2.7 節 pvlib の概要

pvlib は、太陽光発電システムの性能シミュレーションや関連するタスクを実行するための関数とクラスのセットを提供する、コミュニティが開発したツールボックスである。

以下は、pvlib の主な特徴である。

- 太陽位置計算: pvlib は、地球上の任意の場所における太陽の位置を計算する機能を提供する。これは、太陽の方位角や高度角を求めるのに使用される。
- 大気透過モデル: 大気を通過する太陽放射の量や質を推定するモデルが含まれている。
- 太陽光発電システムの性能モデリング: 太陽光発電モジュールやインバーターの性能モデルが含まれており、異なる条件下での太陽光発電システムの出力をシミュレートできる。

## 2.8 節 実測データとpvlibを使って求めた計算データの比較

Elasticsearch サーバーから取得した日射量データと, pvlib を用いて求めた日射量データをプロットしたものを図 2.2 に示す. 図 2.2 は Elasticsearch サーバーから取得した 2022 年 6 月 2 日の日射量データと, リサイクル館の緯度経度と日付情報を入力として pvlib より求めた日射量の計算データをプロットしている.

図 2.1 と比較して, pvlib より求めた日射量が実測データにより近い概形となっていることが分かる.

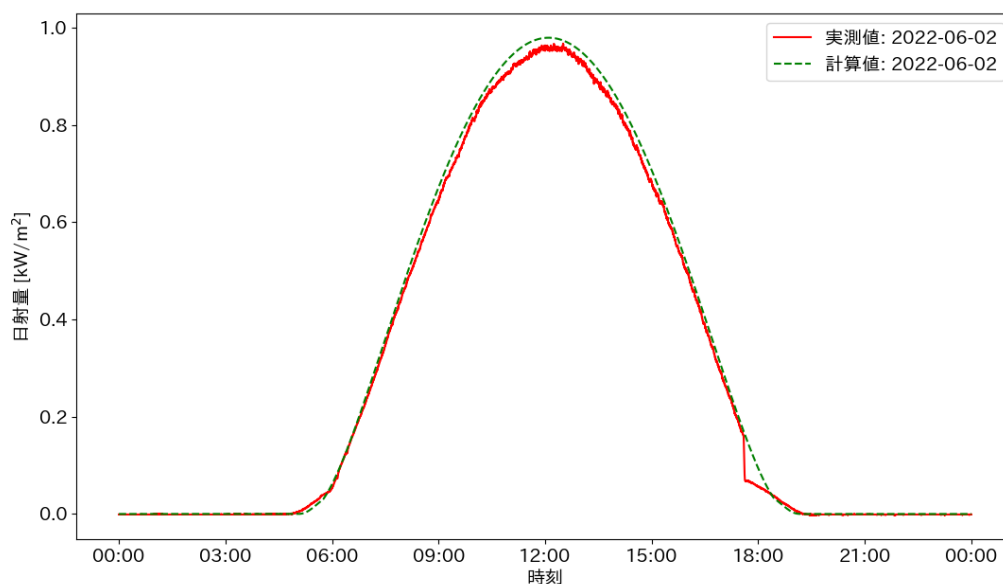


図 2.2 2022 年 6 月 2 日の日射量の実測データと計算データをプロットしたもの

## 2.9 節 pvlib を用いて計算した日射量を使用した, 相互相関によるずれ時間の特定

相互相関の計算に使用する実測データの範囲を, 2022 年 6 月 2 日 0 時 0 分から 2022 年 6 月 2 日 23 時 59 分まで期間とする.

実測データの計測日時の情報をもとに, pvlib より計算データを求め, これらの値を入力として相互相関を計算する.

相互相関を求めた結果, 計算データの日時を実測データより 74 秒進めた際に, 相互相関の値が最大となることが分かった.

式 (2.1) ~ 式 (2.7) を用いて求めた日射量データを用いて相互相関を計算した時と比較して, 124 秒から 74 秒へと 50 秒改善した.

## 2.10 節 前処理の追加による相互相関の計算結果の改善

図 2.2 では, 日没の辺りにおいて, 実測データと計算データの概形が大きくことなっている.

このような実測データに影響を与える外部要因を事前に除去した上で, 相互相関を計算する前処理を追加することで相互相関の計算結果が改善するか検証する.

前処理を含めた相互相関の計算方法は以下のステップで行う.

1. 実測データの日射量を  $0 \text{ kW/m}^2$  と見なすしきい値の指定: まず, 実測データをフィルタリングするために日射量のしきい値を設定する. 今回は,  $q=0.2 \text{ kW/m}^2$  をしきい値として設定する.
2. しきい値に該当するタイムスタンプの特定: 続いて, 実測データの各データ点から  $0.2 \text{ kW/m}^2$  を減算して絶対値を取った際に最も 0 に近い値を取るタイムスタンプを午前と午後でそれぞれ一点ずつ特定する.
3. 特定したタイムスタンプを使った実測データのフィルタリング: 前のステップで得たタイムスタンプを使用して, 2 点のタイムスタンプの外側にある実測データの日射量を  $0 \text{ kW/m}^2$  とする.

4. 計算データとの相互相関の計算: 実測データをフィルタリングした後, 計算データとの相互相関を計算する.

図 2.3 に上述の前処理によってフィルタリングされた実測データと, 計算データをプロットしたものを示す.

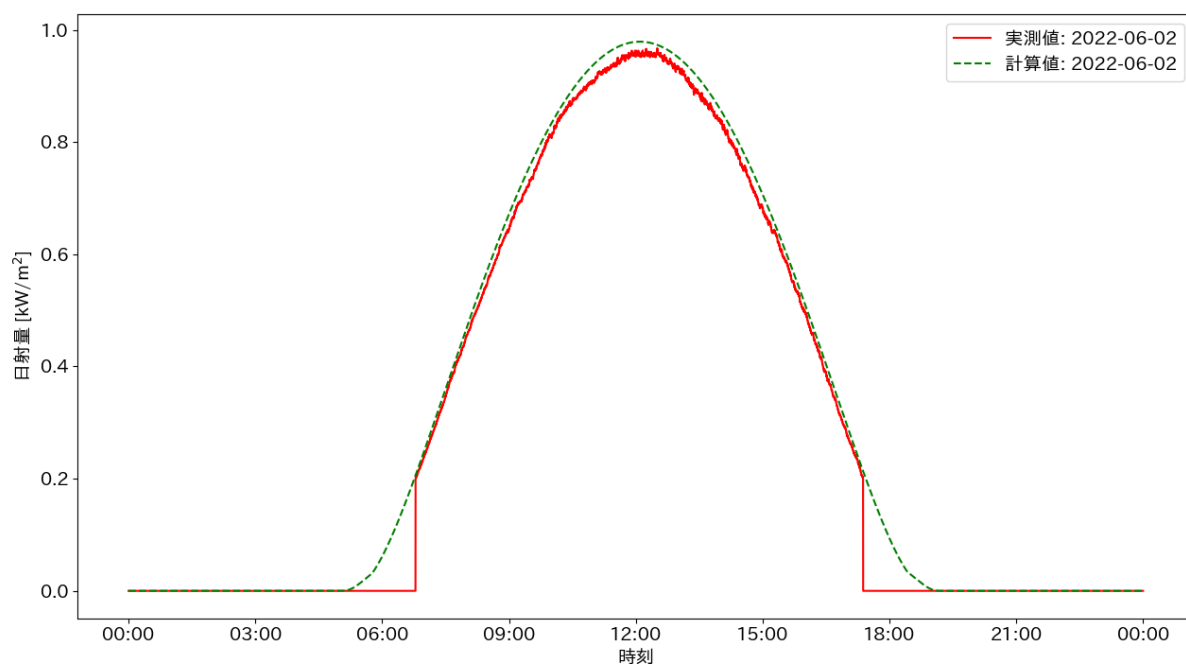


図 2.3 前処理によってフィルタリングされた実測データと, 計算データをプロットしたもの

図 2.3 にプロットしたフィルタリングされた実測データと, 計算データを入力として相互相関を計算する.

相互相関を求めた結果, 計算データの日時を実測データより 14 秒進めた際に, 相互相関の値が最大となることが分かった.

pvlib を使って求めた日射量データを入力として相互相関を計算した時と比較して, 74 秒から 14 秒へと 60 秒改善した.

## 2.11 節 結言

本章では太陽光発電の計測データの補正について述べた。次章では学内ゾーンで稼働している Elasticsearch クラスタへのデータ移行について述べる。

## 第3章

# 学内ゾーンにおける Elasticsearch クラスタへのデータ移行

### 3.1 節 緒言

本章では学内ゾーンで稼働している Elasticsearch クラスタへのデータ移行について述べる。

### 3.2 節 Elasticsearch の概要

Elasticsearch は、分散処理に対応した全文検索エンジンである。主な特徴は、以下の通りである。

- 高速な検索性能: ビッグデータなどの巨大で複雑なデータの集合にも対応可能
- 部分一致検索が可能: 検索キーワードの一部に一致するドキュメントも検索可能
- ほぼリアルタイムの検索: ドキュメントにインデックスを付けてから検索可能になるまで約 1 秒程度
- スケーラビリティ: サーバー数を増やすことで、検索性能と処理能力を拡張可能

これらの特徴から、Elasticsearch は、以下のような用途に適している。

- ログ分析: Web サイトやアプリケーションのログから、アクセス状況やエラー情報を分析する
- セキュリティインテリジェンス: ネットワークやシステムから、セキュリティ脅威を検知する
- ビジネス分析: 顧客データや販売データから、トレンドや傾向を分析する

### 3.3 節 kibana の概要

Kibana は、Elasticsearch に保存されたデータを可視化するためのツールである。主な特徴は、以下の通りである。

- 直感的な操作性: ドラッグ＆ドロップで簡単に可視化を作成できる
- 豊富な可視化機能: グラフ、表、地図など、さまざまな可視化機能を提供
- 高度なフィルタリング機能: 条件を指定して、データを詳細に絞り込むことができる

これらの特徴から、Kibana は、以下のような用途に適している。

- ログ分析: Web サイトやアプリケーションのログから、アクセス状況やエラー情報を可視化する
- セキュリティインテリジェンス: ネットワークやシステムから、セキュリティ脅威を可視化する
- ビジネス分析: 顧客データや販売データから、トレンドや傾向を可視化する



### 3.4 節 学内ゾーンで稼働している Elasticsearch システムの状況

学内ゾーンでは, 133.71.106.168 で単一ノードの Elasticsearch と, 133.71.106.170, 133.71.106.141, 133.71.106.136 の Elasticsearch ノードによって構成された Elasticsearch クラスタが稼働している.

133.71.106.168 の Elasticsearch には, CO<sub>2</sub> 濃度監視システムによって計測されたデータと LEAF の運行日誌に関するデータが保存されている.

### 3.5 節 データ移行対象の Elasticsearch インデックスについて

133.71.106.168 で稼働している単一ノードの Elasticsearch に保存された CO<sub>2</sub> データと LEAF の運行日誌に関するデータを, 学内ゾーンで稼働している Elasticsearch クラスタへ移行する.

### 3.6 節 CO<sub>2</sub> データの移行手順について

CO<sub>2</sub> のデータ移行を行う上で, タイムスタンプと部屋番号の組み合わせが重複しているデータが一部存在しており, この重複データを取り除いた上でデータ移行を行う必要がある. そこで一度, 移行元の Elasticsearch サーバーのデータをローカルマシンにエクスポートして, 重複データを取り除いた上で, 移行先の Elasticsearch サーバーにデータをインサートする.

#### 3.6.1 データのエクスポート

移行元の Elasticsearch サーバーのデータのローカルマシンへのエクスポートには, elasticdump ライブラリを使用して, JSON 形式でエクスポートした. その際, co2 という文字列を含むインデックスのデータのみをエクスポートした.

### 3.6.2 データの重複削除

重複データの削除は SQLite データベースを用いて行った。

SQLite は、軽量で自己完結型のデータベースエンジンです。これは、多くのアプリケーションに組み込まれており、以下の特徴を持っています：

軽量: SQLite は非常に小さく、リソースの少ない環境でも動作します。例えば、組み込みシステムやモバイルアプリケーションなどです。自己完結型: SQLite はサーバーレスで、設定や管理が不要です。これは、データベースが単一のファイルとして存在し、外部の依存関係がないことを意味します。トランザクション: SQLite は ACID トランザクションをサポートしており、データの整合性を保ちます。言語のサポート: 多くのプログラミング言語で使用できます。幅広い用途: デSKTOP、モバイル、ウェブアプリケーションなど、さまざまな環境で利用されています。フリーかつオープンソース: SQLite はパブリックドメインに属し、誰でも自由に使用、変更、配布できます。SQL 標準準拠: 多くの SQL 標準機能をサポートしていますが、全てではありません。

SQLite は、その単純さと効率性から、広く使われているデータベースエンジンの一つです。特に、大規模なデータベースシステムのオーバーヘッドが不要な場合や、独立したアプリケーションでの使用に適しています。

SQLite データベースはリレーショナルデータベースの一種であり、複合主キーを使って複数のテーブルカラムの組み合わせを一意的識別子として扱うことができる。これにより、同じ組み合わせのデータを重複して挿入しようとした場合、データベースエンジンがコンフリクトエラーを発生させ、重複データの挿入を阻止する。そのため、今回の重複データ削除には適していると判断した。

今回使用した SQLite データベースでは、部屋番号 (number) とタイムスタンプ (utctime) を一意のキーとして設定した。以下のリスト 4.1, リスト 3.2 に示すように、移行元の Elasticsearch サーバーに保存されている co2 インデックスのドキュメントは、フィールドのメンバーが統一されておらず、一部センサー情報が存在しない場合がある。そのため、データの挿入時にコンフリクトエラーが発生した場合は、既存のレコードと挿入しようとしたレコードを比較し、既存レコードの値が NULL であるカラムにおいて、挿入しようとしているレ

コードの値が非 NULL である場合には、既存レコードのカラムの値を更新するようにした。これにより、重複データ削除時に一部センサー情報などが欠けてしまう問題を解決した。

Listing 3.1 `_source` フィールドのメンバー数が少ないドキュメント

```
{
  "_index": "co2_e411",
  "_type": "_doc",
  "_id": "nEi2nnoB2-iFXnrMOobM",
  "_score": 1,
  "_source": {
    "utctime": "2020-10-09T05:09:06+00:00",
    "number": "E411",
    "PPM": "481",
    "data": "Thingspeak"
  }
}
```

Listing 3.2 `_source` フィールドのメンバー数が多いドキュメント

```
{
  "_index": "co2_e411",
  "_type": "_doc",
  "_id": "YKBqU4QBugDzeydA2gyi",
  "_score": 1,
  "_source": {
    "RH": 26.98,
    "PPM": 423,
    "JPtime": "2022-11-06T22:45:30.080925",
    "ip": "172.23.68.19/16",
    "utctime": "2022-11-06T13:45:30.080895",
  }
}
```

```
    "TEMP": 24.47 ,  
    "index_name": "co2_e411" ,  
    "ms": "" ,  
    "number": "E411"  
  }  
}
```

### 3.6.3 データのインポート

重複データ削除後のデータが保存された SQLite テーブルからすべてのレコードを読み出して、ターゲットの Elasticsearch サーバーに移行した。

その際、python の elasticsearch ライブラリを使用し、co2\_modbus という名前のインデックスに保存した。

## 3.7 節 一度目のデータ移行で移行できなかった CO<sub>2</sub> データの移行について

実装したデータ移行プログラムを使用して 133.71.201.197 から 133.71.106.141 の Elasticsearch サーバーへ CO<sub>2</sub> データを移行したのが 2023 年 5 月中旬頃であり、CO<sub>2</sub> 濃度監視システムを開発、運用している高木君が、移行先である 133.71.106.141 の Elasticsearch サーバーに対してラズベリーパイから CO<sub>2</sub> データのインサートを行うよう対応したのが 2023 年 7 月中旬であったため、2023 年 5 月中旬から 2023 年 7 月中旬までの間の約 2ヶ月間の CO<sub>2</sub> データが移行先の Elasticsearch サーバーに移行出来ていなかった。そこで、追加の移行作業を行った。

移行方法は以下のとおりである。

1. まず、2023 年 5 月中旬に移行した際の全移行データの中で最も最新の `utctime` フィールドの値を検索する。
  - 検索した結果、2023 年 5 月中旬に移行した際の全移行データの中で最も最新の `utctime` は「2023-05-16T05:48:30.081305」であった。

2. 次に、移行先 ElasticSearch サーバーに対してラズベリーパイからインサートされた全データの中で最も古い utctime フィールドの値を検索する。
  - 検索した結果、ラズベリーパイからインサートされた全データの中で最も古い utctime は「2023-07-20T07:15:39.314008」であった。
3. 前回の CO<sub>2</sub> データの移行は 2023 年 5 月中旬頃に行ったため、2023 年 5 月 1 日 0 時 0 分 0 秒以降の utctime を持つドキュメントを、移行元 ElasticSearch サーバーのインデックス名に co2 という文字列を含むインデックスから elasticdump ライブラリを使用してローカルマシンにエクスポートする。
4. 部屋番号 (number) とタイムスタンプ (utctime) の組み合わせがユニークになるようにエクスポートしたデータをフィルタリングする。
5. 更に、1 と 2 で得られた utctime の範囲に含まれる utctime を持つドキュメントのみになるようフィルタリングする。
6. フィルタリング後のデータを移行先 ElasticSearch サーバーにバルクインサートする。

### 3.8 節 kibana によるデータの可視化

計 2 回の CO<sub>2</sub> データを移行した後の co2\_modbus インデックスについて、横軸をタイムスタンプ (utctime) とし、縦軸を PPM, RH, TEMP としてそれぞれプロットしたものを図 3.1 ~ 図 3.3 に示す。

2 回目の CO<sub>2</sub> データの移行によって、2023 年 5 月中旬から 2023 年 7 月中旬までの期間とその前後の期間において、図 3.1 ~ 図 3.3 より、連続的にデータが変化していることが目視で確認できるので、データ移行は正常に出来たと判断できる。

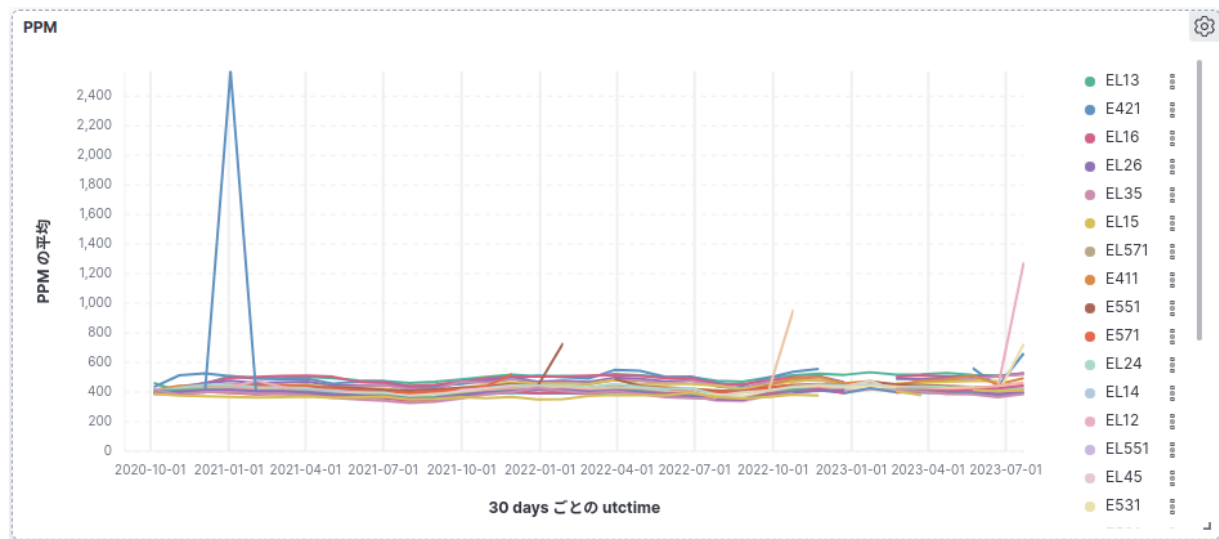


図 3.1 co2\_modbus の PPM

### 3.9 節 LEAF の運行日誌に関するデータの移行について

LEAF の運行日誌に関するデータが保存されたインデックスは以下の 2 つである。

- movement\_diary
- movement\_diary01

これらのインデックスのデータ移行は、同名のインデックスを移行先の ElasticSearch サーバーに作成して、作成したインデックスにデータを挿入することで行った。

次に、上記のインデックスに保存されているデータについて説明する。

以下に movement\_diary と movement\_diary01 のドキュメントの違いを列挙する。

#### 1. driver フィールド:

- movement\_diary のドキュメントでは、driver フィールドは文字列である。

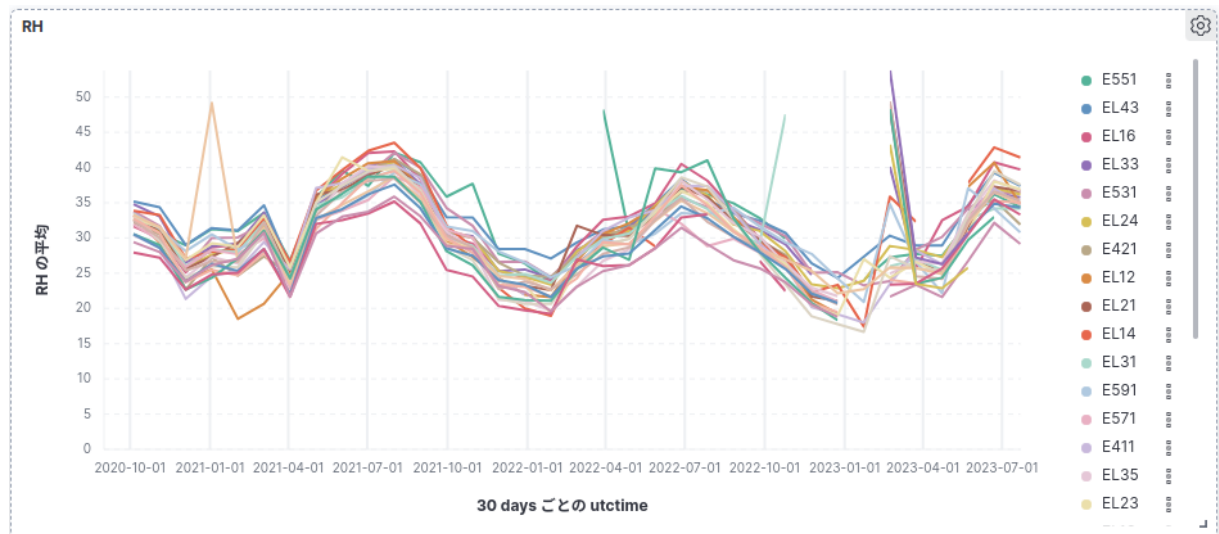


図 3.2 co2\_modbus の RH

- movement\_diary01 のドキュメントでは, driver フィールドは配列で, その中に文字列と 2 つの null 値が含まれている.
2. “destination” フィールド:
- movement\_diary のドキュメントでは, “destination” フィールドは単一の文字列である.
  - movement\_diary01 のドキュメントでは, “destination” フィールドは配列で, その中に 2 つの文字列が含まれている.
3. “charge\_place” フィールド:
- movement\_diary のドキュメントには, “charge\_place” フィールドは存在しない.
  - movement\_diary01 のドキュメントでは, “charge\_place” フィールドが追加されているが, その値は空文字列である.
4. “battery\_rate” フィールド:
- movement\_diary のドキュメントには, “battery\_rate” フィールドは存在しない.

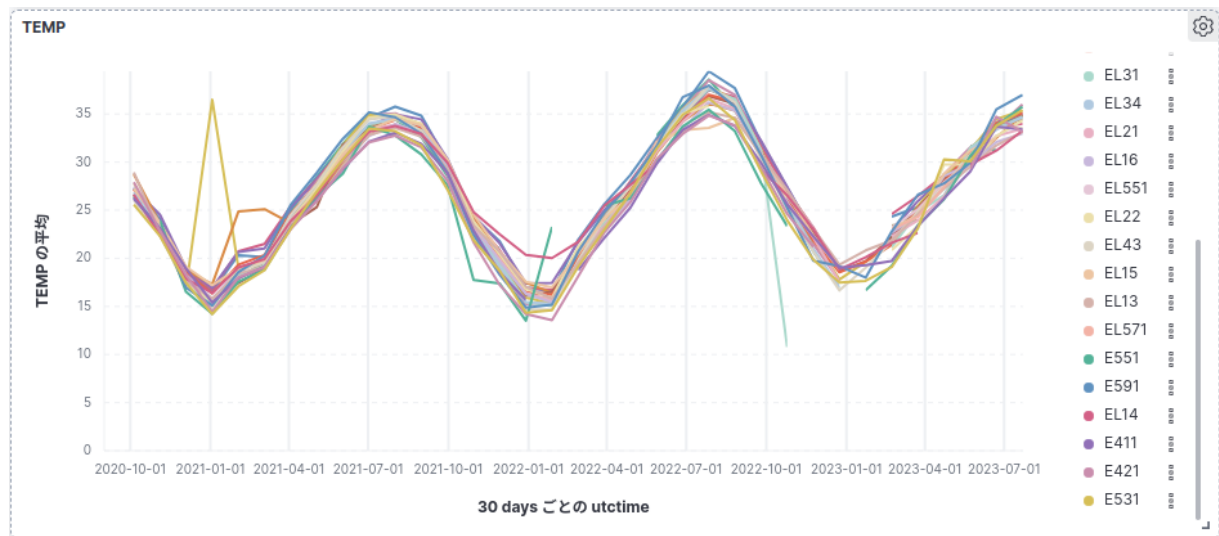


図 3.3 co2\_modbus の TEMP

- movement\_diary01 のドキュメントでは, “battery\_rate” フィールドが追加されており, その値は数値である.

5. “battery\_rate\_distance” フィールド:

- movement\_diary のドキュメントには, “battery\_rate\_distance” フィールドは存在しない.
- movement\_diary01 のドキュメントでは, “battery\_rate\_distance” フィールドが追加されており, その値は数値である.

movement\_diary と movement\_diary01 のドキュメントの違いより, movement\_diary01 は movement\_diary のもつ情報量を全て保持しており, その上で追加のフィールドを持っていることから, 移行するのは movement\_diary01 インデックスのみで十分であることが分かった.



## 3.10 節 LEAF の運行日誌に関するデータの移行手順について

### 3.10.1 データのエクスポート

移行元の ElasticSearch サーバーのデータのローカルマシンへのエクスポートには, `elasticsearch-dump` ライブラリを使用して, `movement_diary01` インデックスの全ドキュメントを JSON 形式でエクスポートした.

### 3.10.2 データのインポート

`python` の `elasticsearch` ライブラリを使用し, 移行先の Elasticsearch に `movement_diary01` という名前のインデックスを作成して, エクスポートしたデータを全てインサートした.

## 3.11 節 結言

本章では学内ゾーンで稼働している Elasticsearch クラスタへのデータ移行について述べた.

次章ではサーバーゾーンでのクラスタ構築における仮想環境を使用した事前検証について述べる.

## 第4章

# サーバーゾーンでのクラスタ構築における仮想環境を使用した事前検証

### 4.1 節 緒言

本章では、サーバーゾーンでのクラスタ構築における仮想環境を使用した事前検証について述べる。

### 4.2 節 サーバーゾーンで稼働している Elasticsearch システムの状況

サーバーゾーンでは、133.71.201.197 で単一ノードの Elasticsearch が稼働しており、リサイクル館の太陽光パネルの計測データが保存されている。

133.71.201.197 の Elasticsearch のバージョン 7.17.6 であり、学内ゾーンで稼働している Elasticsearch クラスタに参加している Elasticsearch ノードのバージョンは 7.17.9 である。

本研究室では現在、バージョン 7.17.9 の Elasticsearch を採用しているため、サーバーゾーンで構築しようとしているクラスタの Elasticsearch のバージョンも、学内ゾーンで稼働している Elasticsearch クラスタと同様、バージョン 7.17.9 を採用する。

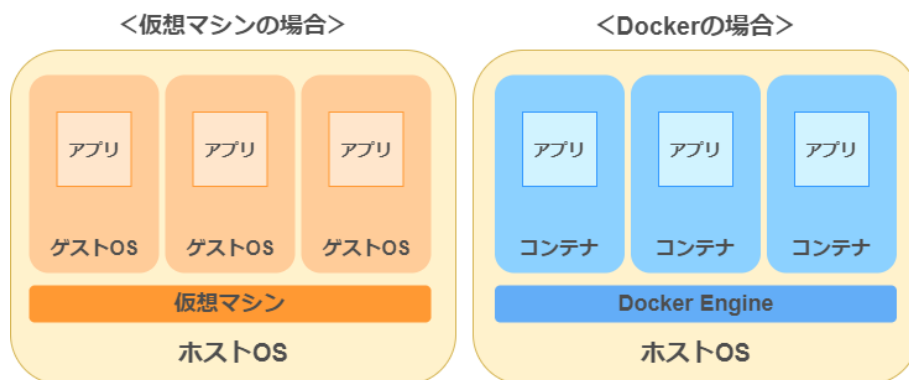


図 4.1 仮想マシンと Docker の違い [3]

そこで、Docker, 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。

## 4.3 節 Docker とは

Docker は、軽量で独立したコンテナ型仮想環境用のプラットフォームである。従来の仮想化では、VMWare などの仮想化ソフトウェアを用いて、ホスト OS 上にゲスト OS を構築する形式だった。しかし、Docker はホスト OS 上にゲスト OS なしで独立したコンテナ型の仮想環境として構築される。Docker コンテナを利用する場合は、Docker Engine をインストールすることでコンテナの立ち上げ、停止、削除といった操作を行うことができる。

### 4.3.1 コンテナとは

コンテナは、アプリケーションとそのすべての依存関係（ライブラリ、実行環境など）をカプセル化した軽量な実行単位である。Docker の場合、コンテナの作成には Docker イメージが必要となる。

### 4.3.2 Docker イメージとは

Docker イメージとは, Docker コンテナを作成するためのテンプレートであり, Docker イメージの中には, Docker コンテナの実行に必要な Linux ファイルシステムとメタ情報を含む.

Linux ファイルシステムというのは, / ディレクトリ以下の /etc /bin /sbin /usr などのディレクトリ階層およびファイルである.

Docker では, コンテナとして動かしたいアプリケーションが必要とする, 最小限のファイルを Docker イメージの中に入れる.

さらに, そのアプリケーションを動かすために必要なデフォルトのコマンドや引数の指定, 外に公開するポート番号の情報などの情報がある. これらをメタ情報として, 同じく Docker イメージの中に入れられる

Docker イメージは Docker Hub やその他のレジストリで共有されており, これらのサービスから取得することが可能である.

今回は Elasticsearch の開発元である Elastic 社が提供している Elasticsearch の Docker イメージを使って検証を行う.

## 4.4 節 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, 複数のコンテナで協調して動作するアプリケーションの開発を単純化する.

## 4.5 節 検証環境のセットアップ

### 4.5.1 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 4.1 に 7.17.9 バージョンの Elasticsearch のみを使用してクラスタを構築した時の docker-compose.yml を示す.

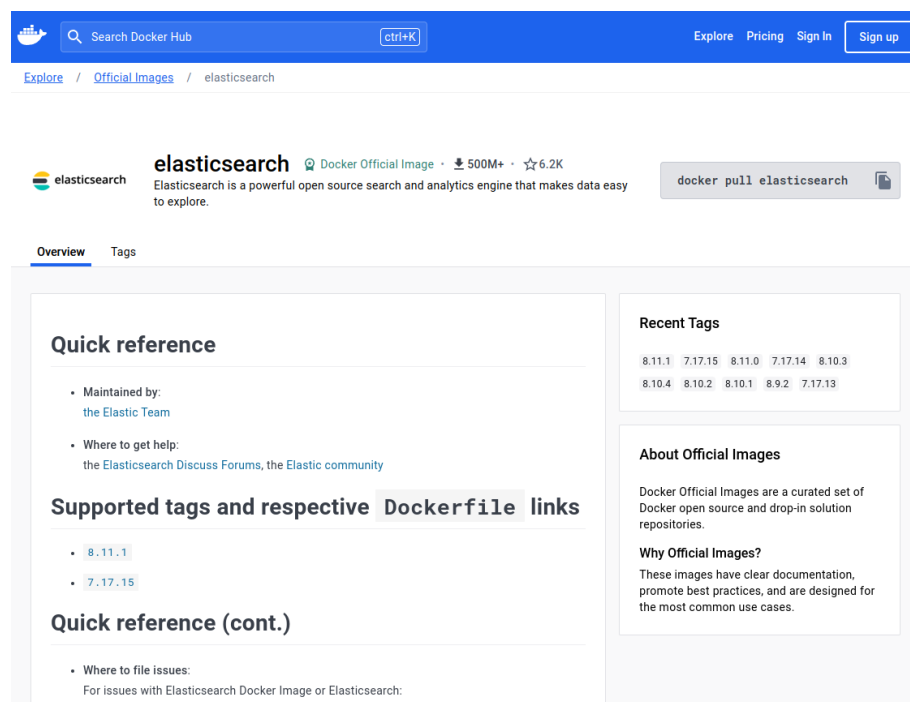


図 4.2 Elasticsearch の Docker イメージ

Listing 4.1 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

```
version: '2.2'

services:
  es01:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
    container_name: es01
    environment:
      - node.name=es01
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es02,es03
      - cluster.initial_master_nodes=es01,es02,es03
    ports:
      - 9200:9200
    networks:
      - elastic
  es02:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
    container_name: es02
    environment:
      - node.name=es02
      - cluster.name=es-docker-cluster
      - discovery.seed_hosts=es01,es03
      - cluster.initial_master_nodes=es01,es02,es03
    networks:
      - elastic
  es03:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.9
    container_name: es03
    environment:
      - node.name=es03
```

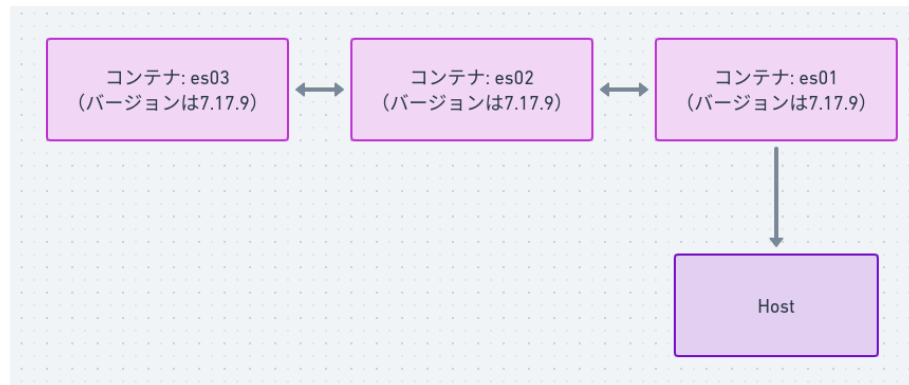


図 4.3 docker-compose.yml を図で表現したもの

```
- cluster.name=es-docker-cluster
- discovery.seed_hosts=es01, es02
- cluster.initial_master_nodes=es01, es02, es03
networks:
- elastic
```

```
networks:
  elastic:
    driver: bridge
```

また、図 4.3 に Listing 4.1 の docker-compose.yml を図で表現したものを示す。  
Listing 4.1 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 のクラスタ設定を含む.
- **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' はホストマシンの 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

- **networks:** デフォルトのドライバである bridge ドライバを使用する elastic ネットワークを定義している. 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.

以降, 説明を簡単にするため, docker-compose.yml を図で表現したもののみを掲示する.

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 ?? に示す.

図 4.4 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.



```
$ 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 4.4 クラスタに参加しているノードを一覧表示した結果

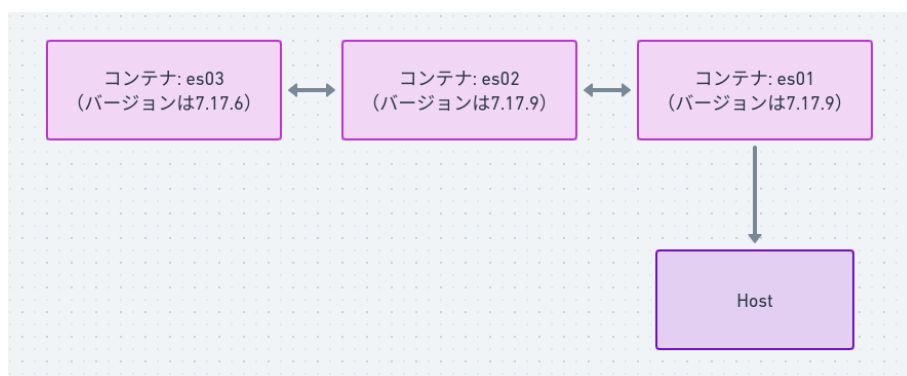


図 4.5 変更後の docker-compose.yml を図で表現したもの

#### 4.5.2 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)

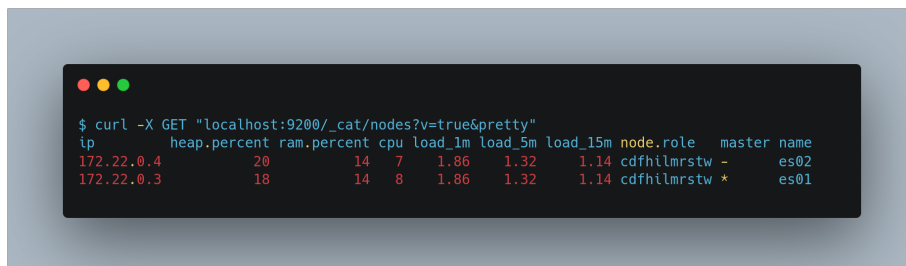
図 4.3 の docker-compose.yml の es03 のコンテナが使用する Docker イメージを変更して, es03 のノードで使用する Elasticsearch のバージョンを 7.17.9 から 7.17.6 に変更する.

図 4.5 に変更後の docker-compose.yml を図で表現したものを示す.

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.

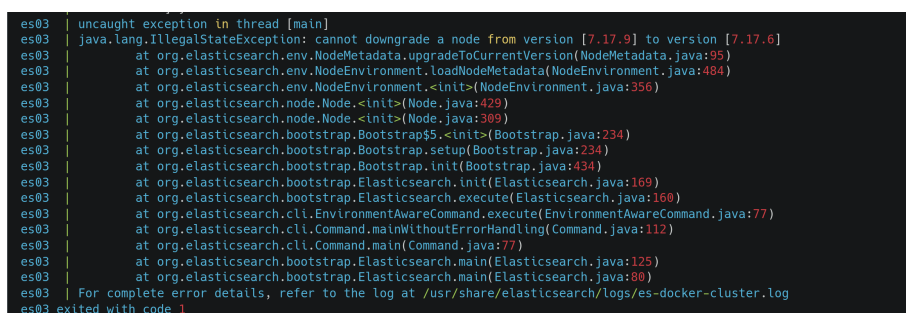
クラスタの起動後, curl コマンドを使用してクラスタに参加しているノードを一覧表示した結果を図 4.6 に示す.

図 4.6 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.



```
$ 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
172.22.0.4 20         14       7    1.86   1.32   1.14   cdfhilmrstw -   es02
172.22.0.3 18         14       8    1.86   1.32   1.14   cdfhilmrstw *   es01
```

図 4.6 クラスタに参加しているノードを一覧表示した結果



```
es03 | java.lang.IllegalStateException: cannot downgrade a node from version [7.17.9] to version [7.17.6]
es03 |   at org.elasticsearch.env.NodeMetadata.upgradeToCurrentVersion(NodeMetadata.java:95)
es03 |   at org.elasticsearch.env.NodeEnvironment.loadNodeMetadata(NodeEnvironment.java:484)
es03 |   at org.elasticsearch.env.NodeEnvironment.<init>(NodeEnvironment.java:356)
es03 |   at org.elasticsearch.node.Node.<init>(Node.java:429)
es03 |   at org.elasticsearch.node.Node.<init>(Node.java:389)
es03 |   at org.elasticsearch.bootstrap.Bootstrap$5.<init>(Bootstrap.java:234)
es03 |   at org.elasticsearch.bootstrap.Bootstrap.setup(Bootstrap.java:234)
es03 |   at org.elasticsearch.bootstrap.Bootstrap.init(Bootstrap.java:434)
es03 |   at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:169)
es03 |   at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:160)
es03 |   at org.elasticsearch.cli.EnvironmentAwareCommand.execute(EnvironmentAwareCommand.java:77)
es03 |   at org.elasticsearch.cli.Command.mainWithoutErrorHandling(Command.java:112)
es03 |   at org.elasticsearch.cli.Command.main(Command.java:77)
es03 |   at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:125)
es03 |   at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:80)
es03 | For complete error details, refer to the log at /usr/share/elasticsearch/logs/es-docker-cluster.log
es03 exited with code 1
```

図 4.7 es03 のログ

また, Elasticsearch 起動時に出力されたログを確認したところ, 図 4.7 に示すように, クラスタに参加できなかった es03 のコンテナで Elasticsearch がエラーログを出力して終了していることが分かった。

## 4.6 節 異なる Elasticsearch クラスタへのノード参加検証

サーバーゾーンでのクラスタ構築において, リサイクル館の太陽光パネルの計測データを保存している Elasticsearch ノードを新たなノードとして構築したクラスタに参加できるか, Docker を用いて検証した。

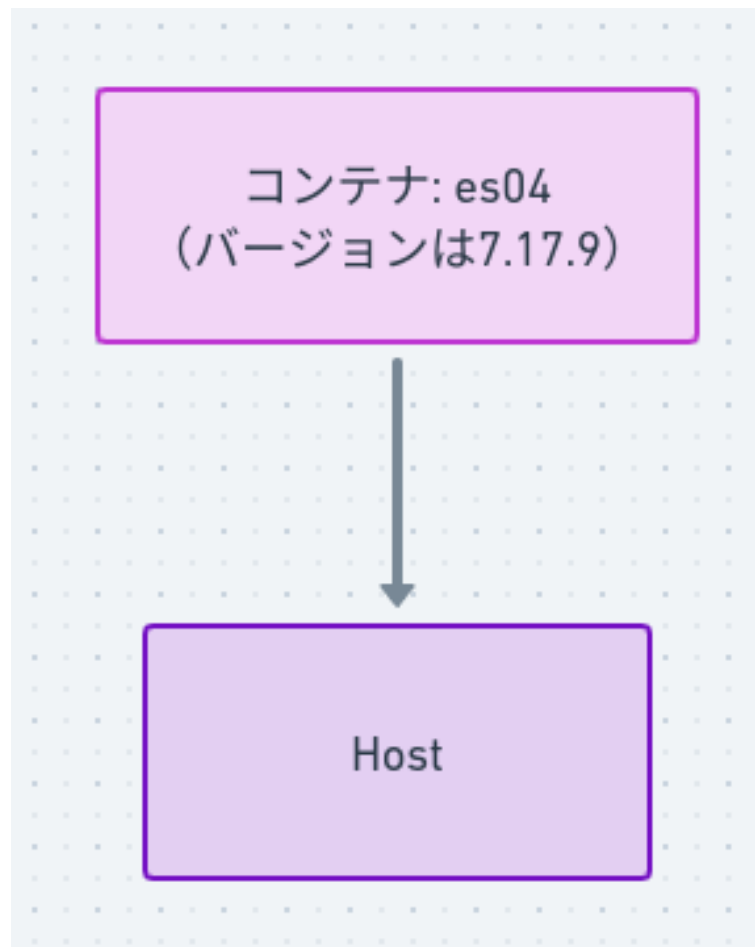


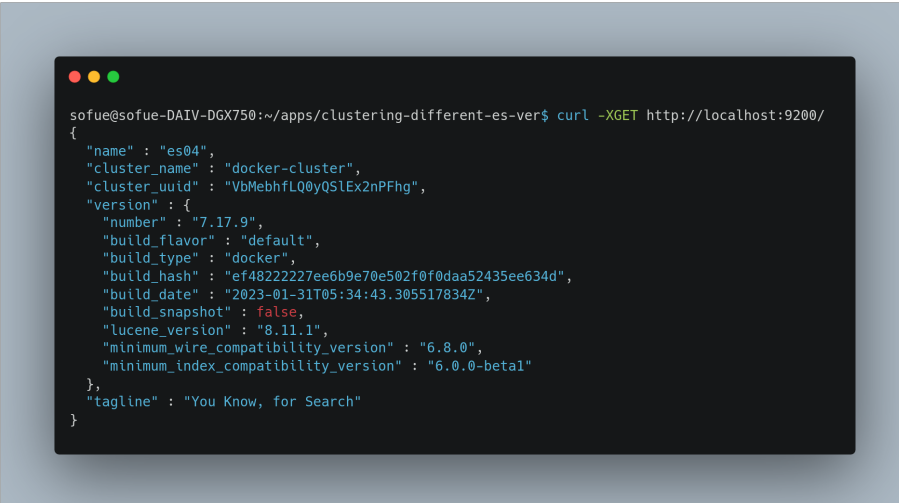
図 4.8 クラスタ A の構築の際に使用した docker-compose.yml を図で表現したもの

## 4.7 節 手順

### 4.7.1 単一ノードで稼働するクラスタ A の構築

まず, docker-compose を用いて単一ノード (コンテナ名は es04) でクラスタ  
以後このクラスタをクラスタ A と呼ぶを構築する. 以後このクラスタをクラ  
スタ A と呼ぶ.

図 4.8 にクラスタ A の構築の際に使用した docker-compose.yml を図で表現  
したものを示す.



```
sofue@sofue-DAIV-DGX750:~/apps/clustering-different-es-ver$ curl -XGET http://localhost:9200/
{
  "name" : "es04",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "VbMebhfLQ0yQSLEx2nPfhg",
  "version" : {
    "number" : "7.17.9",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "ef48222227ee6b9e70e502f0f0daa52435ee634d",
    "build_date" : "2023-01-31T05:34:43.305517834Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

図 4.9 クラスタの情報について問い合わせた結果

docker-compose を用いてノードを起動した後、クラスタの情報について問い合わせた結果を図 4.9 に示す。

クラスタの情報について問い合わせた後、Docker コンテナを停止してノードをシャットダウンした。

#### 4.7.2 クラスタ B の構築

次に、クラスタ A に使用したノードとは別の 3 ノード (コンテナ名はそれぞれ es01, es02, es03) でクラスタを構築する。以後このクラスタをクラスタ B と呼ぶ。

図 4.10 にクラスタ B の構築の際に使用した docker-compose.yml を図で表現したものを示す。

docker-compose を用いて 3 つのノードを起動した後、クラスタの情報について問い合わせた結果を図 4.11 に示す。

図 4.9, 4.11 より、クラスタ A とクラスタ B はそれぞれ異なるクラスタ ID を付与されたことが分かる。

クラスタの起動後、クラスタに参加しているノードの一覧を取得した結果を図 4.12 に示す。

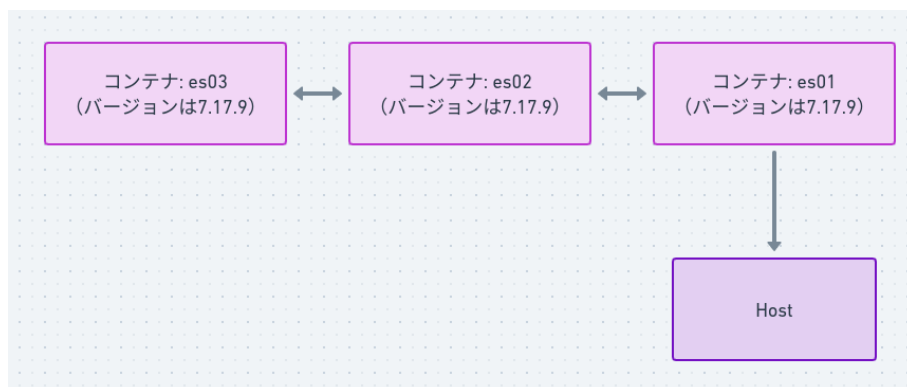


図 4.10 クラスタ B の構築の際に使用した docker-compose.yml を図で表現したもの

```
sofue@sofue-DAIV-DGX750:~/apps/clustering-different-es-ver$ curl -XGET http://localhost:9200/
{
  "name" : "es01",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "XNQVSIKyTkuytnT_rKQt4w",
  "version" : {
    "number" : "7.17.9",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "ef4822227ee6b9e70e502f0f0daa52435ee634d",
    "build_date" : "2023-01-31T05:34:43.305517834Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

図 4.11 クラスタの情報について問い合わせた結果

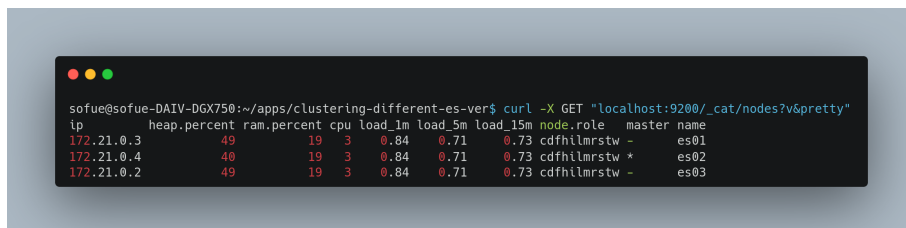


図 4.12 クラスタ B の起動後, クラスタに参加しているノードの一覧を取得した結果

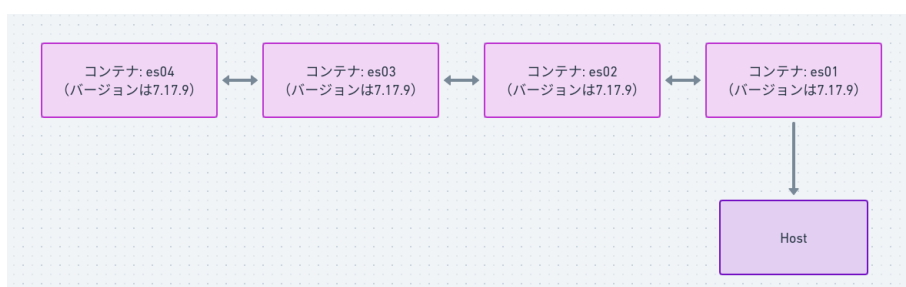


図 4.13 合計 4 ノードでクラスタ B の起動を試みた際に使用した docker-compose.yml を図で表現したもの

図 4.12 より, es01, es02, es03 ノードが全てクラスタ B に参加できていることが分かる。

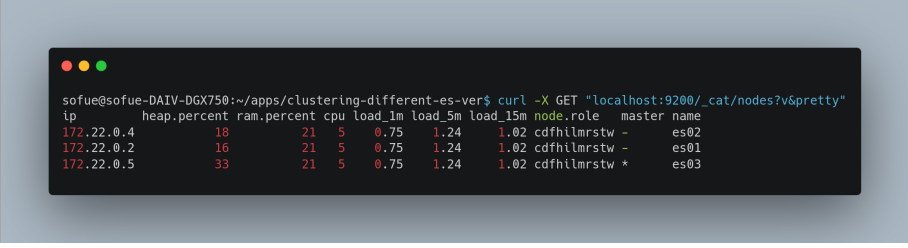
クラスタに参加しているノードの一覧を取得した後, 全ての Docker コンテナを停止してノードを全てシャットダウンした。

### 4.7.3 クラスタ B への参加試行

次に, 図 4.10 の docker-compose.yml に対して, クラスタ A のノード (es04 コンテナ) を追加し, 合計 4 ノードでのクラスタ B の起動を試みる。

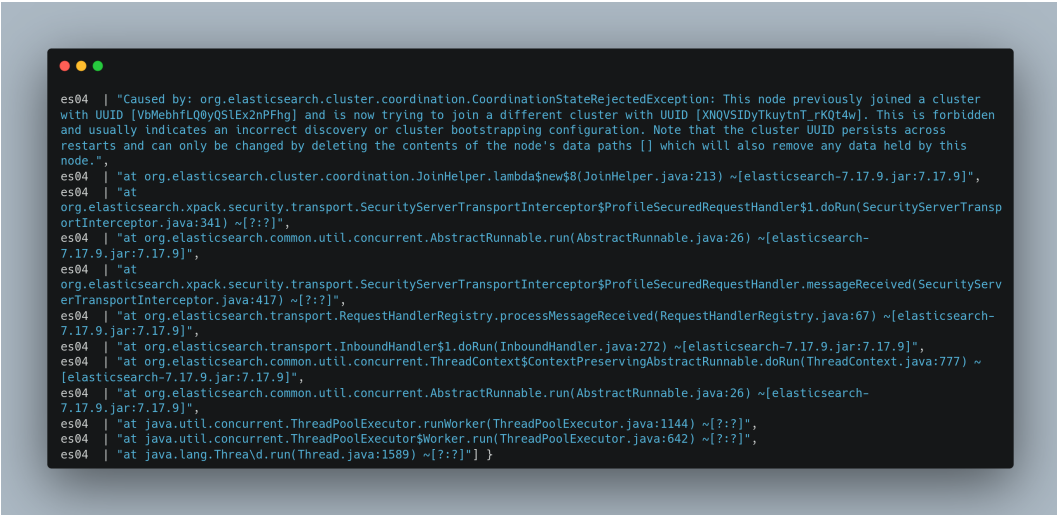
図 4.13 に, 合計 4 ノードでクラスタ B の起動を試みた際に使用した docker-compose.yml を図で表現したものを示す。

クラスタの起動後, クラスタに参加しているノードの一覧を取得した結果を図 4.14 に示す。



```
sofue@sofue-DAIV-DGX750:~/apps/clustering-different-es-ver$ curl -X GET "localhost:9200/_cat/nodes?v&pretty"
ip          heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
172.22.0.4  18           21      5    0.75   1.24    1.02  cdfhlmrstw -   es02
172.22.0.2  16           21      5    0.75   1.24    1.02  cdfhlmrstw -   es01
172.22.0.5  33           21      5    0.75   1.24    1.02  cdfhlmrstw *   es03
```

図 4.14 合計4ノードでクラスタの起動を試みた後、クラスタに参加しているノードの一覧を取得した結果



```
es04 | "Caused by: org.elasticsearch.cluster.coordination.CoordinationStateRejectedException: This node previously joined a cluster with UUID [VbMebhFL0yQSLEx2nPfhg] and is now trying to join a different cluster with UUID [XNQV5IDyTkuytnT_rKQt4w]. This is forbidden and usually indicates an incorrect discovery or cluster bootstrapping configuration. Note that the cluster UUID persists across restarts and can only be changed by deleting the contents of the node's data paths [] which will also remove any data held by this node."
es04 | "at org.elasticsearch.cluster.coordination.JoinHelper.lambda$new$8(JoinHelper.java:213) ~[elasticsearch-7.17.9.jar:7.17.9]",
es04 | "at org.elasticsearch.xpack.security.transport.SecurityServerTransportInterceptor$ProfileSecuredRequestHandler$1.doRun(SecurityServerTransportInterceptor.java:341) ~[?:?]",
es04 | "at org.elasticsearch.common.util.concurrent.AbstractRunnable.run(AbstractRunnable.java:26) ~[elasticsearch-7.17.9.jar:7.17.9]",
es04 | "at org.elasticsearch.xpack.security.transport.SecurityServerTransportInterceptor$ProfileSecuredRequestHandler.messageReceived(SecurityServerTransportInterceptor.java:417) ~[?:?]",
es04 | "at org.elasticsearch.transport.RequestHandlerRegistry.processMessageReceived(RequestHandlerRegistry.java:67) ~[elasticsearch-7.17.9.jar:7.17.9]",
es04 | "at org.elasticsearch.transport.InboundHandler$1.doRun(InboundHandler.java:272) ~[elasticsearch-7.17.9.jar:7.17.9]",
es04 | "at org.elasticsearch.common.util.concurrent.ThreadContext$ContextPreservingAbstractRunnable.doRun(ThreadContext.java:777) ~[elasticsearch-7.17.9.jar:7.17.9]",
es04 | "at org.elasticsearch.common.util.concurrent.AbstractRunnable.run(AbstractRunnable.java:26) ~[elasticsearch-7.17.9.jar:7.17.9]",
es04 | "at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144) ~[?:?]",
es04 | "at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:642) ~[?:?]",
es04 | "at java.lang.Thread.run(Thread.java:1589) ~[?:?]" }
```

図 4.15 es04 コンテナのログ

図 4.14 より、クラスタ A のノードがクラスタ B に参加できていないことが分かる。

es04 コンテナ（クラスタ A のノード）で出力されたログの一部を図 4.15 に示す。

図 4.15 には、異なるクラスタ ID を持つクラスタにノードが参加することは禁止されており、これを行うためにはインデックスやドキュメント情報などが格納されているデータパス配下のフォルダ、ファイルを削除する必要があると書かれている。

以上の検証結果から、既に稼働しているノードを別のクラスタに新しいノード

ドとして参加させることは出来ないことが分かった。

したがって、リサイクル館の太陽光パネルの計測データが保存された Elasticsearch ノードをクラスタに参加させるには以下の2通りの方法が考えられる。

- リサイクル館の太陽光パネルの計測データが保存された Elasticsearch ノードのバックアップを取り、ノードに保存されたインデックスやドキュメントのデータを削除した上で、CO<sub>2</sub> データなどが保存されたクラスタに新しいノードとして参加させる
- CO<sub>2</sub> データなどが保存されたクラスタとは別で、サーバーゾーンに新たにクラスタを構築する。クラスタの構築にはリサイクル館の太陽光パネルの計測データが保存された Elasticsearch ノードが所属するクラスタを使用する。

## 4.8 節 Elasticsearch のバージョンアップ

3章の検証結果より、異なるバージョンの Elasticsearch ノードでクラスタを構築することは出来ないため、リサイクル館の太陽光パネルの計測データを保存している Elasticsearch(133.71.201.197) をバージョンアップする必要がある。そこで、133.71.201.197 にインストールされた Elasticsearch のバージョンアップを行う。

## 4.9 節 バージョンアップ手順

### 4.9.1 インストール方法の特定

バージョンアップを行うためには、133.71.201.197 の UbuntuPC にどのように Elasticsearch をインストールしたか特定する必要がある。

図 4.16 に、apt によってインストールされたパッケージの中に elasticsearch という文字列を含むパッケージが存在するか調べた結果を示す。図 4.16 より、apt によってインストールされたことが分かった。





```
matsubara@iris:~$ apt list --installed | grep elasticsearch
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
elasticsearch/stable,now 7.17.6 amd64 [installed,upgradable to: 7.17.15]
```

図 4.16 apt によって elasticsearch がインストールされたか調べた結果

次に, apt でインストール可能な elasticsearch のバージョンを一覧表示した結果を図 4.17 に示す. 図 4.17 にターゲットである 7.17.9 が含まれているため, apt を使用してバージョンアップできることが確認できた.

#### 4.9.2 apt によるバージョンアップ

まず, `sudo systemctl stop elasticsearch.service` コマンドを実行して elasticsearch ノードをシャットダウンする.

次に, `sudo apt install elasticsearch=7.17.9` コマンドを実行して elasticsearch パッケージをバージョンアップする.

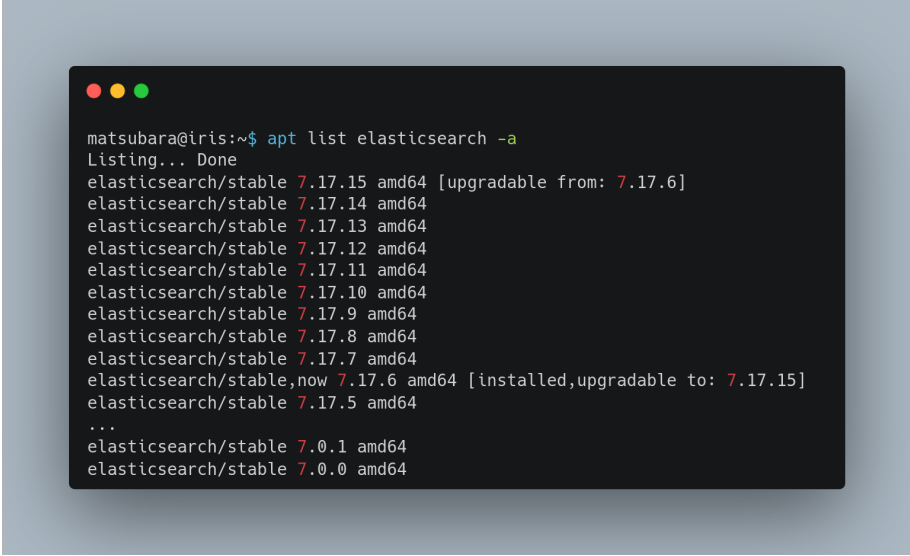
elasticsearch をバージョンアップ後, `sudo systemctl start elasticsearch` コマンドを実行して Elasticsearch ノードを起動する.

ノードの起動後, Elasticsearch のバージョンを確認した結果を図 4.18 に示す.

図 4.18 より, Elasticsearch のバージョンが 7.17.9 にバージョンアップ出来たことが確認できた.

### 4.10 節 kibana のバージョンアップ

kibana も elasticsearch と同様, apt を使用してインストールされていたため, `sudo systemctl stop kibana.service` コマンド, `sudo apt install kibana=7.17.9` コマンド, `sudo systemctl start kibana` コマンドをそれぞれ



```
matsubara@iris:~$ apt list elasticsearch -a
Listing... Done
elasticsearch/stable 7.17.15 amd64 [upgradable from: 7.17.6]
elasticsearch/stable 7.17.14 amd64
elasticsearch/stable 7.17.13 amd64
elasticsearch/stable 7.17.12 amd64
elasticsearch/stable 7.17.11 amd64
elasticsearch/stable 7.17.10 amd64
elasticsearch/stable 7.17.9 amd64
elasticsearch/stable 7.17.8 amd64
elasticsearch/stable 7.17.7 amd64
elasticsearch/stable,now 7.17.6 amd64 [installed,upgradable to: 7.17.15]
elasticsearch/stable 7.17.5 amd64
...
elasticsearch/stable 7.0.1 amd64
elasticsearch/stable 7.0.0 amd64
```

図 4.17 apt でインストール可能な elasticsearch のバージョンを一覧表示した結果



```
matsubara@iris:~$ curl -u takenaka:takenaka -s -XGET http://localhost:9200/ | grep number
"number" : "7.17.9",
```

図 4.18 ノードの起動後, Elasticsearch のバージョンを確認した結果

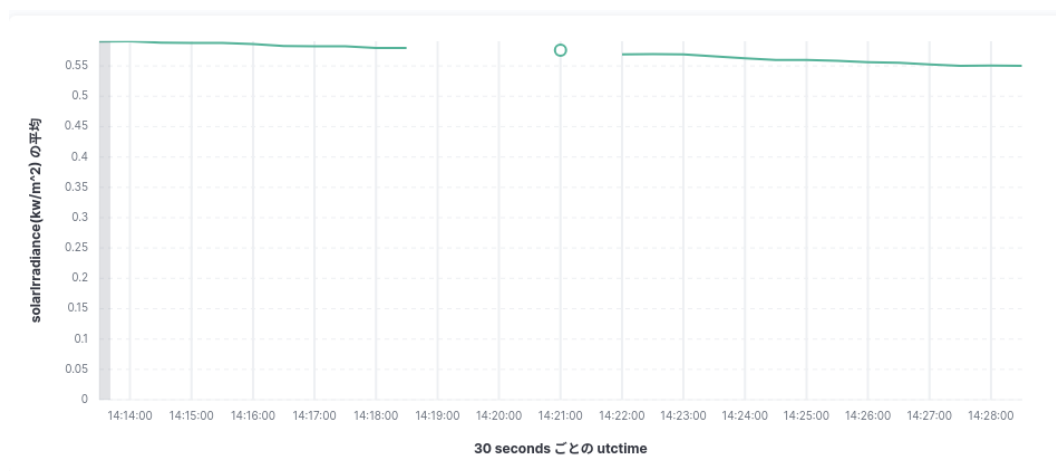


図 4.19 Elasticsearch のバージョンアップ後, 太陽光パネルの計測データが保存されているか kibana 上で確認した結果

れ実行して, kibana のバージョンアップも行った.

#### 4.11 節 バージョンアップ後の動作確認

Elasticsearch のバージョンアップ後, 太陽光パネルの計測データが Elasticsearch に保存されているか kibana 上で確認した結果を図 ??に示す.

図 4.19 より, バージョンアップ後の Elasticsearch ノードを起動した 14:22:00 以降にドキュメントがインサートされていることが確認できた.

#### 4.12 節 サーバゾーンにおけるクラスタの構築

#### 4.13 節 結言

本章では, サーバゾーンでのクラスタ構築における仮想環境を使用した事前検証について述べた.

次章では結論と今後の課題について述べる。

## 第5章

### 結論と今後の課題

#### 5.1 節 結論

本研究では、太陽光発電の計測データ補正と Elasticsearch のデータ移行および冗長化について詳細に検討し、以下の主要な成果を達成した。

- 相互相関を用いた太陽光発電計測データの時間的ずれの特定手法の提案。
- Elasticsearch クラスタへのデータ移行に関する具体的な手順の確立と成功による、データ管理とアクセスの効率化。
- サーバーゾーンでの Elasticsearch クラスタ構築に向けた仮想環境を使用した事前検証を通じて、バージョンアップの重要性と手順の確立。

#### 5.2 節 今後の課題

本研究の成果を踏まえ、今後の研究の方向性として以下の課題が考えられる。

- 太陽光発電計測データの補正手法のさらなる改善。
- 学内ゾーンとサーバーゾーンでそれぞれ稼働しているクラスタごとに kibana が存在しており、本研究室で管理する Elasticsearch に保存されたデータを一元的に管理、閲覧することが出来ないため、kibana の統合による一元管理の実現。

- システムの継続的なモニタリングと定期的なメンテナンスの実施.

## 謝 辞

本研究を行うにあたり、終始、懇切丁寧な御指導と適切な御助言を賜りました本学工学部電気電子工学科通信システム工学研究室の都築伸二教授に深甚なる感謝の意を表します。

## 参考文献

- [1] 中川清隆, ”太陽方位、高度、大気外日射量の計算”,  
[http://es.ris.ac.jp/nakagawa/met\\_cal/solar.html](http://es.ris.ac.jp/nakagawa/met_cal/solar.html), 参照 May 23, 2022.
- [2] 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 20,2023.
- [3] RAKUS Developers Blog, ”Docker とは一体何なんだ?【初心者向け】 - RAKUS Developers Blog — ラクス エンジニアブログ”, <https://tech-blog.rakus.co.jp/entry/20221007/docker>, 参照 Nov 20,2023.