Orchestrator 管理者のための ミドルウェア運用設定ガイド Ⅱ

- Elastic Stack,
- HAA/Redis,
- Load Balancer 編

Version 1.0

UiPath Robotic Process Automation ®

免責事項

- 本ガイドの内容は 2020 年 8 月現在の情報であり、下記の製品リリースに基づいております。
  UiPath Orchestrator v2020.4 (FTS)
  製品の新しいリリース、修正プログラム などによって、動作・仕様が変わる可能性がありますので、予めご留意ください。
- 本ガイドで扱う 3rd Party 製のミドルウェアは特に明記しない限り以下の通りです：
  - CentOS {7, 8}/ Red Hat Enterprise Linux {7, 8}
  - Elastic Stack = {Elasticsearch, Kibana, Curator} version 7.x
  - Redis = {UiPath High Availability Add-on (HAA), Redis Enterprise, Redis OSS}
  - Load Balancer = F5 BIG-IP
- 本ガイドに含まれる情報は可能な限り正確を期しておりますが、UiPath 株式会社の正式なレビューを受けておらず、本ガイドに記載された内容に関して UiPath 株式会社は何ら保証するものではございません。従って、本ガイドに含まれる情報の利用またはこれらの技法の実施は使用者の責任においてなされるものであり、ガイドの内容によって受けたいかなる被害に関して一切の補償をするものではございません。
- 本ガイドをコピー等で複製する場合は、UiPath 株式会社および執筆者の承諾なしではできません。

商標について

- UiPath、UiPath Orchestrator、UiPath Robot、UiPath Studio および UiPath ロゴは、世界の多くの国で登録された UiPath の米国およびその他の国における商標です。
- Microsoft、Windows、Windows Server、SQL Server、Active Directory および Windows ロゴ は Microsoft Corporation の米国およびその他の国における商標です。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその他の関連会社の米国およびその他の国における商標または登録商標です。
- Elastic、および関連するロゴとマークは、Elastic N.V.及びその関連会社の商標または登録商標です。
- Redis は、Redis Labs Ltd の商標です。
- BIG-IP は、F5 Networks の商標です。
- その他、記載されている製品名、会社名およびサービス名はそれぞれの各社の商標または登録商標です。

Revision History

Date	Version	Author	Description
2020/09/02	1.0	Hiroaki Mishima (UiPath)	1st version.

Table of Contents

1. はじめに

1.1. 概要と注意事項

本ガイドは「UiPath Orchestrator（以下、OCと略記）の運用・保守に従事する管理者」を読者として想定しており、OC稼働のための**任意の**構成要素（ソフトウェア）であるミドルウェア（Elastic Stack［特にElasticsearch〈以下、ESと略記〉, Kibana, Curator］、Redis［UiPath High Availability Add-on〈以下、HAAと略記〉, Redis Enterprise, Redis OSS］, Load Balancer［以下、LBと略記］）に対して推奨される設定項目、監視観点などのベストプラクティスを解説しています。

本ガイドで紹介している設定には優先度として「高」と「低」のいずれかが割り振られています。「優先度：高」は、OCや各ミドルウェアのパフォーマンスや運用に影響を与える可能性が高い設定ですが、未設定でもOCの機能は損なわないことを意味します。「優先度：低」は、OCや各ミドルウェアのパフォーマンスに与える影響が軽微であることを意味します。また、最適な設定はお客様のOCや各ミドルウェアの構成環境・利用方法に依存するため、「優先度：低」に設定されている項目は弊社として現時点で一意な推奨設定をご提案することが難しい項目も含まれます。

特に明記しない限り、本ガイドに記載されている内容はオンプレミスで構築する場合を前提としています。クラウド等の他のサービスで構築する場合は、利用者が参照や変更ができない内容があります。設定できない内容に関しまして、その設定値や変更方法についてはそれぞれのサービサーにお問い合わせください。

本ガイドで提示されている設定は、記載されている回避策を管理者が自己の判断で使用することを前提に提供されているものであり、自己の責任においてご使用ください。

本ガイドのWebリソースは下記のリンクにございます。
https://www.uipath.com/ja/resources/knowledge-base/middleware-config-maintenance-guide-vol2

1.2. UiPath の用語

UiPath 製品に関わる用語とその定義を次の表に示します。

用語	略語	定義
Orchestrator	OC	一連の Robot にライセンスを付与し、配置、管理及び監視を行うことができる Web アプリケーションです。Robot が参照するリソースを管理したり、動作状況を監視したり、Robot を実行環境に展開したりできます。
Studio	ST	Robot に実行させるための自動化処理（Workflow）を作成するための統合開発環境です。
Robot	(N/A)	Studio で作成された Workflow に従って、コンピュータ上でプロセス（実際の自動化処理）を実行するソフトウェアです。
Attended Robot	AR	Robot の種類の一つです。ユーザーの監視下において、直接操作をした結果として起動し、Workflow 等のデベロップメント・アウトプットを実行します。
Unattended Robot	UR	Robot の種類の一つです。ユーザーの監視が無い状態で Workflow 等のデベロップメント・アウトプットを実行します。
Package	(N/A)	Workflow の集まり（プロジェクト）を zip 圧縮したデータです。拡張子は.nupkg です。
Heartbeat	HB	Robot (/Studio) から Orchestrator に 30 秒間隔で送信されます。Robot (/Studio) のステータス確認のための HTTP リクエストです。

2. OS の共通設定

本ガイドで取り扱う OS は Red Hat 系 Linux（CentOS、RHEL 等）のみです。
UiPath による Redis Enterprise の OEM 製品である UiPath High Availability Add-on (HAA)では Debian 系 Linux の Ubuntu が要件外のため、**Ubuntu は本ガイドの範囲外**とします。また、Redis OSS も Windows 版の開発が止まっていることもあり、**Windows も本ガイドの範囲外**とします。

- 本章のチェックリスト
  検討項目は下記の通りです。

§	検討項目	優先度	検討タイミング	推奨	分類
2.1.1	SELinux の無効化	低	運用後	社内ポリシーに反しないのであれば、無効化します。	設定

2.1. Linux カーネルの設定

Linux カーネルは、Linux OS の主要コンポーネントであり、コンピュータのハードウェアとプロセスの間のコアインタフェースです。これら 2 つの間で通信し、可能な限り効率的にリソースを管理する役割を担っています。カーネルには次の 4 つの機能があります：

- メモリ管理: メモリが、何をどこに保存するためにどれくらい使用されたのかを追跡します。
- プロセス管理: どのプロセスがいつ、どれだけの期間、CPU を使用できるかを定めます。
- デバイスドライバー: ハードウェアとプロセスの間の仲介および通訳として機能します。
- システムコールとセキュリティ: プロセスからサービス要求を受け取ります。

正常に実装されているカーネルはユーザーからは認識されず、カーネル空間と呼ばれる専用の場所で動作します。この空間でメモリを割り当て、データが保存されているすべての場所をトレースします。Web ブラウザやファイルなど、ユーザーから認識できるものはユーザー空間と呼びます。これらのアプリケーションは、システムコール・インタフェース（略: SCI）を通じてカーネルと連携します。

参考

[1] Linux カーネルとは（Red Hat 社公式ガイド）

2.1.1. SELinux の無効化

SELinux は Linux の監査やセキュリティを向上させる機能です。

優先度：低
検討タイミング：運用後
推奨

お客様のポリシーに反しないのであれば、無効にしておくことを推奨します。

理由

SELinux が有効の場合、サービスの動作や設定内容が大幅に制限されます。

方法

1) SELinux の状態（既定で有効）を確認します。以下のように getenforce コマンドを実行して「Enforcing」と表示された場合には selinux が有効になっています。

```
[<username>@<hostname> ~]$ getenforce
Enforcing
```

2) Root ユーザーで/etc/selinux/config ファイルを修正します。vi コマンドで対象の設定ファイルを開きます。

```
[<username>@<hostname> ~]$ vi /etc/selinux/config
```

3) 下記の赤字の箇所を「enforcing」から「disabled」に修正します。修正箇所のスペルを間違えると OS が起動しなくなる場合がありますのでご注意ください。

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX= disabled
# SELINUXTYPE= can take one of three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

4) SELinux が無効になっていることを確認します。

```
[<username>@<hostname> ~]$ getenforce
Disabled
```

参考

[2] Security-Enhanced Linux

3. Elastic Stack – Elasticsearch の設定

Elastic Stack は Elastic 社が提供するデータの収集・加工・集計・分析のソフトウェア群です。本ガイドでは特に Elasticsearch と Kibana を扱っています。

UiPath Robot の実行ログは、既定では SQL Server の [UiPath] データベースの [Logs] テーブルに記録されますが、Orchestrator のアプリケーション設定（%PROGRAMFILES(X86)%¥UiPath¥Orchestrator¥Web.config の変更）により Elasticsearch にも実行ログを記録させることができます。また SQL Server にログを送らず、Elasticsearch にのみログを格納させることができます。

実行ログの格納先を Elasticsearch に限定させることで Orchestrator の必須コンポーネントである SQL Server の負荷を軽減させることができます。変更を伴わないデータの格納先としては、SQL Server のようなリレーショナルデータベースよりも「データのバケツ」としての特徴が強い Elasticsearch の方が動作も軽量です。Elasticsearch に保存された実行ログは OC 管理画面からも確認することができ、Kibana によって多角的に分析して可視化することもできます。

Elasticsearch の公式用語のうち、本ガイドに関連する用語を下記に列挙します：

用語	定義
Node	Elasticsearch のプロセス（≒サーバー）です。基本的には 1 つのサーバーに、1 つの Elasticsearch が稼働するため、Node 数≒サーバー数となります。
Cluster	データ全体を一緒に保持する 1 つ以上の Node（サーバー）の集合です。すべての Node に亘って統合したインデクシング機能と検索機能を提供します。名前の既定値は "elasticsearch" です。Cluster は一意の名前で識別されます。複数の環境がある場合は重複しないようにします（"logging-dev", "logging-stage", "logging-prod", etc.）。
Document	Index を構成し、JSON 形式で表現可能な情報です。RDBMS で云うところのレコードに例えることもできます。
Index	Document の集合です。RDB で云うところの Table のような論理的な概念です。
Shard	Index を各 Elasticsearch Node へ分散して配置するための物理的な概念です。
Segment	Document が格納されています。一度作成されると中身の Document を編集したり削除したりできないファイルとなります。Document が削除される場合は Segment に削除マークが付与されます。
Type	Document の型です。※v7.x 系より廃止されました。
Primary Shard	Index の書き込みと参照処理で使用される Shard です。Primary Shard の数は Index 作成時に固定されます。Index 作成後に Primary Shard の数は変更できません。Elasticsearch v7.x より、Index 当たりの既定の Primary Shard 数は 5 から 1 に変更されました。
Replica Shard	Index の参照処理で使用される Shard であり、Primary Shard のコピーです。Primary Shard を保持する Node とは別の Node に保持されます。 Replica Shard の作成は必須ではありません。Replica Shard の数は動的に変更することができます。Primary Shard がきえてしまった消失した場合、該当の Replica Shard が昇格されフェールオーバーに使用されます。
Index Aliases	Index（及びそれらの集合）に対する別名です。単に別名を与えるだけではなく、自動的に追加処理（ルーティング／フィルタリングなど）の論理的なプロキシとしても機能します。

- 本章のチェックリスト
  検討項目は下記の通りです。

§	検討項目	優先度	検討タイミング	推奨	分類
3.1.1	Elasticsearch の OS 選定	高	運用前	CentOS または Red Hat Enterprise Linux (RHEL)	設定
3.1.2	仮想メモリマップ領域	高	運用後	vm.max_map_count≥262,144	設定
3.2.1	Split Brain 対策	高	運用前	※詳細は本文を参照してください。	設定
3.2.2	ノード追加	低	運用後	※詳細は本文を参照してください。	設定
3.3.1	Swap の無効化	高	運用後	起動時にメモリを確保し、Swap しないようにする設定します。	設定
3.3.2	Fielddata Cache	低	運用後	Fielddata Cache に使用可能なメモリのサイズを制限します。	設定
3.3.3	Process/Thread の数	低	運用後	※詳細は本文を参照してください。	設定
3.3.4	Thread Pool のサイズ	低	運用後	threadpool.index.queue_size: -1 threadpool.bulk.queue_size: -1 threadpool.search.queue_size: -1	設定
3.3.5	Refresh Interval	低	運用後	Index.refresh_interval: 30s Index.refresh_interval: -1	設定
3.3.6	File Descriptors	低	運用後	MAX_OPEN_FILES≥65,536=2^{16}	設定
3.3.7	Merge Thread 数	低	運用後	※詳細は本文を参照してください。	設定
3.3.8	Search.max_buckets ﾊﾟﾗﾒｰﾀ	低	運用後	search.max_buckets: 65,535 (v7.9), 10,000 (v7.0~7.8), 10,000 (v6.2~6.8)	設定
3.3.9	実行ログの最大表示件数	低	運用後	※詳細は本文を参照してください。	設定
3.4.1	Heap サイズ	高	運用後	物理メモリの半分以下かつ 30GB 以下にします。	設定
3.4.2	Metaspace (Off-Heap)	高	運用後	※詳細は本文を参照してください。	設定
3.4.3	GC ログ	低	運用後	GC ログを適切に取得するように設定します。	設定
3.4.4	Heap Dump	低	運用後	OOME 発生時に Heap ダンプを自動的に取得させます。	設定
3.5.1	Primary Shard 数	高	運用後	Shard 数≒Index サイズ[GB]/30GB	設定
3.5.2	Replica Shard 数	低	運用後	Elasticsearch が冗長構成の場合、1 つ以上とします。	設定
3.5.3	累積 Shard のパージ	低	運用後	※詳細は本文を参照してください。	設定
3.6.1	ストレージ構成	高	運用前	※詳細は本文を参照してください。	設定
3.6.2	パーティション	低	運用前	データはデータ領域専用のパーティションにストアします。	設定
3.6.3	ディスクサイズ	高	運用前	※詳細は本文を参照してください。	設定
3.6.4	ディスク空き容量の閾値	高	運用後	※詳細は本文を参照してください。	設定
3.7.1	_all フィールド	低	運用後	全フィールドを跨いで検索しない無い場合は無効にします。	設定
3.7.2	_source フィールド	低	運用後	※詳細は本文を参照してください。	設定
3.7.3	stored_fields	低	運用後	※詳細は本文を参照してください。	設定
3.7.4	not_analyzed	低	運用後	※詳細は本文を参照してください。	設定
3.7.5	doc_values	低	運用後	※詳細は本文を参照してください。	設定
3.7.6	Index Aliases	低	運用後	※詳細は本文を参照してください。	設定
3.7.7	ignore_above	低	運用後	ignore_above の値を 1024 以上に設定します。	設定
3.7.8	Index Template	高	運用後	利用ケースを踏まえた Template を策定して設定します。	設定
3.8.1	ディスクの空き容量	高	運用後	※詳細は本文を参照してください。	監視
3.8.2	Cluster のｽﾃｰﾀｽとﾉｰﾄﾞの可用性	低	運用後	※詳細は本文を参照してください。	監視
3.8.3	Index のパフォーマンス	高	運用後	※詳細は本文を参照してください。	監視

3.8.4	検索のパフォーマンス	低	運用後	※詳細は本文を参照してください。	監視
3.8.5	Heap の監視	高	運用後	※詳細は本文を参照してください。	監視
2.8.6	ネットワークの Thread Pool	低	運用後	※詳細は本文を参照してください。	監視
3.9.1	Index の close	高	運用後	参照しなくなった古い Index の close を定期的に実施します。	運用
3.9.2	Index の削除	高	運用後	不要な Index の削除を定期的に実施します。	運用
3.9.3	リポジトリの作成	高	運用後	スナップショットの運用を開始する前に実施します。	運用
3.9.4	月次スナップショット整理	高	運用後	月初に一度だけスナップショットの作成を実施します。	運用
3.9.5	日次スナップショット取得	高	運用後	日次スナップショットの作成を可能な限り毎日実施します。	運用
3.9.6	スナップショットのリストア	低	運用後	リストアは必ず Curator または API 経由で実施します。	運用

参考

[3] Elasticsearch リファレンス [5.4] » 始めてみよう » 基本概念（Elastic 社公式ガイド）

3.1. Elasticsearch の OS 設定

3.1.1. Elasticsearch の OS 選定

インストール先の OS として、Linux と Windows があります。

優先度：高
検討タイミング：運用前
推奨

Linux（CentOS、RedHat Enterprise Linux）が推奨されます。

理由

Linux と Windows を動作環境として比較した場合に、評価軸（実績、機能、性能、運用）に対するポイントを下記の表に示します。

評価軸	説明
実績	- Elastic 社としては Linux への導入実績は豊富ですが、Windows への導入実績が少ないようです。フォーラム等でのセットアップや運用面での Q&A においても Linux ベースが多く、Windows 固有の問題については相対的に情報が少ないのが実情です。 - UiPath 社としては Linux と Windows の両方で導入実績があります。
機能	機能面での差はありません。お客様の環境、運用管理コスト、ポリシーなどを踏まえてご選択ください。
性能	お客様の利用環境に依りますが、大きな差はありません。
運用	一般に、Linux 環境の方が Windows 環境に比べて動作が軽量なため HW 要件を節約できる傾向があります。

3.1.2. 仮想メモリマップ領域

Linux 環境の Elasticsearch は既定で mmapfs (memory mapped file system) ディレクトリを使用して Index を保存します。Linux OS による mmap カウントの既定の制限は低すぎる可能性があります。この制限は vm.max_map_count パラメータで指定され、プロセスが使用可能なメモリマップ領域の最大数を意味します。

優先度：高
検討タイミング：運用後
推奨

vm.max_map_count パラメータの値を「262144」以上に増やしてください。既定値は「65530」です。

理由

メモリ不足の例外の発生を抑制します。

方法

< 非永続的かつマシン再起動が不要な方法 >

5) 次のコマンドを実行して、現在のシステムに設定されているパラメータの設定値を確認します。

```
[<username>@<hostname> ~]$ /sbin/sysctl -a
```

例えば、下記のように出力されます：

```
... (略) ...
vm.max_map_count = 65530
... (略) ...
```

6) root 権限で次のコマンドを次のように実行することで、mmap カウントの制限を増やすことができます。

```
[root@<hostname> ~]# sysctl -w vm.max_map_count=262144
```

< 永続的な方法 >
永続的に設定を保持するには、/etc/sysctl.conf の vm.max_map_count パラメータの値を更新します。sysctl vm.max_map_count を実行して、現在値を確認できます。

i) root 権限で/etc/sysctl.conf を開きます。

```
[root@<hostname> ~]# vi /etc/sysctl.conf
```

ii) 「vm.max_map_count=262144」の一行を追記します。

```
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).

vm.max_map_count=262144
```

iii) 次のコマンドを実行して設定を反映します。システムの再起動は不要です。

```
[root@<hostname> ~]# /sbin/sysctl -p /etc/sysctl.conf
```

備考

< エラー例 >
本設定が適切でない場合、Elasticsearch 起動時に以下のようなエラーが発生する恐れがあります。

```
{"type": "server", "timestamp": "2020-06-04T17:34:31,796Z", "level": "INFO", "component":
"o.e.b.BootstrapChecks", "cluster.name": "docker-cluster", "node.name": "mishimahr-es01", "message":
"bound or publishing to a non-loopback address, enforcing bootstrap checks" },
ERROR: [1] bootstrap checks failed,
[1]: max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144],
{"type": "server", "timestamp": "2020-06-04T17:34:31,810Z", "level": "INFO", "component":
"o.e.n.Node", "cluster.name": "docker-cluster", "node.name": "mishimahr-es01", "message":
"stopping ..." },
{"type": "server", "timestamp": "2020-06-04T17:34:31,835Z", "level": "INFO", "component":
"o.e.n.Node", "cluster.name": "docker-cluster", "node.name": "mishimahr-es01", "message": "stopped" },
... (略) ...
```

参考

[4] Elasticsearch Reference [7.8] » Set up Elasticsearch » Important System Configuration » Virtual memory（Elastic 社公式ガイド）

3.2. Elasticsearch の高可用構成

3.2.1. Split Brain 対策

優先度：高

検討タイミング：運用前

推奨

A) 高可用 (High Availability; HA) 構成の Elasticsearch Cluster を組む場合は Master ノード候補が 3 台以上の奇数台数でなければなりません。

※ 2 台構成の Cluster はデータの欠損という障害に対しては有効ですが、1 台が故障しても Cluster が正常に動作するという意味では、**Split Brain** などの問題を内包してしまいます。

B) Elasticsearch 6 系より以前は minimum_master_nodes の値を **(master_eligible_nodes/2)+1** で設定することが推奨されます。

※ **7 系以降では** minimum_master_nodes は廃止されました。Master ノードの一覧を内部で管理するようになり、かつこの一覧が自動で更新されるようになったため、**設定は不要**です。

理由

< Elasticsearch 6 系より以前にあった問題 >

Elasticsearch は分散システムであり、その状態を管理するために Master ノードと呼ばれる特殊なノードが Cluster 内のノードの中から 1 台だけ選出されます。2 台構成の Cluster では、この Master ノードの選出過程で問題が発生します。Elasticsearch が Master ノードを選出するプロセスは、次のようなものになる。

i) 全ての Master ノード候補が、Master ノードになるべきノードに対して投票します（自分自身の場合もある）。ただし、投票は minimum_master_nodes 以上の Master ノード候補が確認できた場合にのみ可能です。

ii) 過半数 (minimum_master_nodes) の票を得た Master ノード候補が Master ノードになります。票がちょうど半分に割れた場合は再投票となります。

2 台しか存在しない Cluster の場合、次のような問題が顕在化します。

- minimum_master_nodes を 2 に設定した場合、1 台でも故障した場合は Master ノードを選び出すことができなくなります。よって、Cluster は Master ノードが存在しない状態になります。その結果、Index を新規作成したり、Cluster の設定を変更することができなくなってしまいます。
- minimum_master_nodes を 1 に設定した場合、ネットワーク上でノード同士が互いに見えなくなった際に Split Brain と呼ばれる状態に陥ります。Master ノードが複数になってしまい、Cluster として正常に機能しなくなってしまいます。

方法

< Elasticsearch 6 系より以前の方法 >

cURL コマンドから設定する例（6 台構成の場合）

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/_cluster/settings -d '{
 "persistent": {"discovery.zen.minimum_master_nodes": 4}
}'
```

備考

Elasticsearch 7 系以降における、アクティブな Master ノードの一覧の管理は下記のようになります。

- Cluster の初回起動時は、elasticserach.yml ファイル内の、cluster.initial_master_nodes に定義したノードが Master 適格ノードとなります。
- Master ノードの台数が変わってからしばらく経つと、一覧が更新されます。例えば、7 台の Master ノードのうち 4 台を同時に落とすと機能しなくなりますが、2 台停止後にその状態をしばらく維持して別の 2 台を停止すれば問題ありません。
- Master ノードを 2 台から 1 台に減らしたい場合は Voting Configuration Exclusions API で、Master ノードの内の 1 台を予め除外します。

参考

[5] 新世代 Elasticsearch クラスターコーディネーション（Elastic 社公式ブログ）
[6] Elasticsearch Reference [7.8] » Modifying your data » Adding and removing nodes（Elastic 社公式ガイド）

3.2.2. ノード追加

優先度：低
検討タイミング：運用後
推奨

- ノード数が増えたら Master ノード兼 Data ノードではなく、Master 専用ノードと Data 専用ノードに分けます。分ける場合は、Master ノード候補のスペックは Data ノードより低くても問題ありません。
- データ量が増えただけであれば、Data ノードを増やします。

備考

高可用構成の場合には Master ノードの候補は 3 台以上の奇数台必要ですが、通常は 5 台や 7 台に増やす必要はありません。

3.3. Elasticsearch インスタンスの設定

主に /etc/elasticesearch/elasticsearch.yml を修正することで設定できます。

3.3.1. Swap の無効化

Kibana 経由で Elasticsearch を利用する場合、ファセット（データ集計のための検索軸フィルタ）を利用するため、検索処理にメモリを使用します。OS のメモリ管理機能の一つで、メインメモリ（RAM）とストレージ（外部記憶装置）の同容量の領域間でデータの交換を行う動作をメモリスワップ（英: Swap）と呼びます。これにより、物理的なメモリ容量より広いメモリ空間を扱えるようになりますが、ディスク I/O の遅延や再配置の処理負荷等で Elasticsearch のパフォーマンスに影響します。

優先度：高

検討タイミング：運用後

推奨

起動時にメモリを確保し、**Swap しない**ようにする設定します。既定では Swap の利用に制限が掛かっていません。

※　他のアプリとの相乗りサーバーであるなどの何らかの理由で問題にならない限り、OS レベルで Swap を無効化しても構いません。問題となる場合には Elasticsearch として利用するメモリアドレスを実メモリに限定します。

理由

Elasticsearch は、物理メモリが十分な場合、Swap を使わないようにする方が性能を発揮します。

方法

Swap を使用しない場合、Linux の Swap を無効にするか、Elasticsearch の設定を変更して Elasticsearch が Swap を使用しないようにする方法があります。現在の設定は SSH クライアント上で以下の cURL コマンドを実行して確認できます。

```
[<username>@<hostname> ~]$ curl http://localhost:9200/_nodes/process?pretty
```

実行結果は以下のようになり、「"mlockall" : true」になっていれば Swap を使わない設定になっています。ここでは「"mlockall" : false」（既定）となっていることが確認できます。

```
{
 ... (略) ...
    "process" : {
      "refresh_interval_in_millis" : 1000,
      "id" : 37152,
      "mlockall" : false
    }
... (略) ...
}
```

< elasticsearch.yml から設定する方法 >

1) "mlockall" を true にするために、例えば vi コマンドなどを利用して設定ファイルを開きます。

```
[<username>@<hostname> ~]$ vi /etc/elasticsearch/elasticsearch.yml
```

2) 先頭の"#"を削除して、bootstrap.memory_lock のコメントアウトを外し、設定を有効にします。

```
# ---------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
#
bootstrap.memory_lock: true
#
# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
```

3) 修正後、変更が反映するように elasticsearch を再起動します。

```
[<username>@<hostname> ~]$ systemctl restart elasticsearch.service
```

備考

< systemd の設定 >

elasticsearch.yml を修正して再起動しても "mlockall" が false のままの場合は systemd の設定も行います。

i) 以下のディレクトリが存在するかを確認します。 yum でインストールしただけの場合はこのディレクトリが存在しないため、その場合は root 権限で作成します。

```
[root@<hostname> ~]# mkdir /etc/systemd/system/elasticsearch.service.d
```

ii) このディレクトリ内に override.conf ファイルを作成します。

```
[<username>@<hostname> ~]$ vi /etc/systemd/system/elasticsearch.service.d/override.conf
```

iii) override.conf ファイルに下記の内容を記述して保存します。

```
[Service]
LimitMEMLOCK=infinity
```

iv) 以下のコマンドを実行して設定ファイルを再読込します。

```
[username@<hostname> ~]$ systemctl daemon-reload
```

v) Elasticsearch を再起動します。再起動後に "mlockall" が true になっていることを確認します。

参考

[7] Elasticsearch Reference [7.8] » Set up Elasticsearch » Important System Configuration » Disable swapping（Elastic 社公式ガイド）
[8] Elasticsearch Reference [7.8] » Set up Elasticsearch » Important System Configuration » Configuring system settings（Elastic 社公式ガイド）

3.3.2. Fielddata Cache

Elasticsearch では各 field のデータをメモリに載せることで高速な検索（Sort や Aggregation）を実現しています。これを Fielddata Cache と呼びます。

優先度：低
検討タイミング：運用後
推奨

Fielddata Cache に使用可能なメモリのサイズを制限します。
※ §3.7.5 の doc_values を使うのであれば設定は不要です。doc_values:true を設定すると Fielddata Cache をメモリではなくディスクに乗せることができますが、メモリに乗せる場合よりも 20%程低速になります。

理由

既定ではデータが無制限にメモリに載るため、OutOfMemoryError（略: OOME）を助長させる可能性があります。本設定は、異常にコストの高いクエリに対して OOME になる前にエラーレスポンスが返されるケースを想定して、安全装置の役割を果たします。

方法

elasticsearch.yml に下記設定を追記してサービスを再起動します。適正値は運用しながら調整する必要があります。下記の例では indices.fielddata.cache.size を Heap サイズの 75%に設定しています。

```
indices.fielddata.cache.size: 75% #default 無制限
```

具体的な値も設定できます。例えば Heap メモリが 16GB の場合、例えば「12gb」のように設定できます。

備考

< 現在の Fielddata Cache サイズの確認 >
SSH クライアントから、下記の cURL コマンドで確認できます。fielddata.memory_size_in_bytes が該当のサイズを表しています。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_stats/fielddata?pretty
```

< Fielddata Cache の手動パージ >
Fielddata Cache を手動でクリアするには下記の API（cURL コマンドなど）を利用します。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/_cache/clear?fielddata=true
```

例えば、「{"_shards":{"total":123,"successful":123,"failed":0}}」のような結果が返ってきます。

< エラー例 >
下記のような CircuitBreaker が関わるエラー（Circuit Breaking Exception）が散見される場合は本節の設定の適用を検討してください。

```
ElasticsearchException[org.elasticsearch.common.breaker.CircuitBreakingException:
Data too large, data for field [id] would be larger than limit of [19206242304/17.8gb]];
```

CircuitBreaker は OutOfMemory をある程度抑止してくれる機構です。あくまで評価、必ずしも高コストのクエリを的確にブロックできるとは限らず、完全に制御できる訳ではありません。クエリ実行時に必要とするメモリを見積もり、その値が閾値を超えたら実行させないという機能を果たしますが、実際にはこの見積もりはうまく行かないことも多く、Circuit Breaking Exception が発生することなくクエリが実行されてしまい、OOME が発生する場合があります。

< indices.fielddata.cache.expire パラメータについて>
同様のパラメータに indices.fielddata.cache.expire がありますが、こちらの利用または変更は Elastic 社により非推奨とされています。

参考

[9] Elasticsearch Reference [7.8] » Set up Elasticsearch » Configuring Elasticsearch » Field data cache settings（Elastic 社公式ガイド）
[10] Elasticsearch: The Definitive Guide [2.x] » Aggregations » Doc Values and Fielddata » Limiting Memory Usage > Circuit Breaker（Elastic 社公式ガイド）
[11] Real Memory Circuit Breaker でノードの回復性が大きく向上（Elastic 社公式ブログ）

3.3.3. Process/Thread の数

Elasticsearch は、異なる複数の操作を処理するために、Thread Pool を使用します。

優先度：低
検討タイミング：運用後
推奨

Elasticsearch 実行ユーザーが作成できるプロセス数を、少なくとも「4096」以上に設定します。
理由

Elasticsearch を実行するユーザーは、十分な数の Process (Thread) の作成を行える必要があります。

方法

/etc/security/limits.conf を編集して設定します。

1) プロセス数の現在の設定値を確認します。一般ユーザーと root ユーザーの両方で確認しましょう。
   <一般ユーザー>

```
[<username>@<hostname> ~]$ ulimit -u
4096
```

   < root ユーザー >

```
[root@<hostname> ~]# ulimit -u
64021
```

   ここでは、一般ユーザーの設定値が「4096」、root ユーザーの設定値が「64021」だと分かります。既に「4096」以上の値の場合は設定を変更する必要はありません。

2) vi コマンドで/etc/security/limits.conf を開きます。

```
[root@<hostname> ~]# vi /etc/security/limits.conf
```

3)　下記の赤字の項目を追記または修正します。値は「4096」以上の値を設定します。

```
# /etc/security/limits.conf
... (略) ...
#
#<domain>          <type>      <item>       <value>
#

#*                 soft        core         0
#*                 hard        rss          10000
#@student          hard        nproc        20
#@faculty          soft        nproc        20
#@faculty          hard        nproc        50
#ftp               hard        nproc        0
#@student          -           maxlogins    4
*                  soft        nofile       4096
*                  hard        nofile       4096
root               soft        nproc        4096
root               hard        nproc        4096

# End of file
```

参考

[12]　Number of threads（Elastic 社公式ガイド）

3.3.4. Thread Pool のサイズ

Elasticsearch の内部実装として、indexing や検索などの処理毎に Thread Pool を持っています。この Thread Pool は、Elasticsearch 起動時に CPU 数を元に自動で最適なサイズに設定されます。ただし、プロセッサ数が 32 を越える場合は、プロセッサ数を手動で設定する必要があります。

　内部で起動される Thread Pool の Queue サイズには上限があり、indexing されるデータ量が多い場合は Queue サイズを溢れてしまい、データが欠落する可能性があります。

優先度：低
検討タイミング：運用後
推奨

「threadpool.index.queue_size」「 threadpool.bulk.queue_size」「threadpool.search.queue_size」を無制限の「-1」または十分大きな値（例えば既定値の 10 倍程度）に設定してください。

理由

Orchestator から bulk API で一度に大量のデータを Elasticsearch に投入すると、 Elasticsearch が処理し切れない場合があります。これは例えば、内部 Queue の Thread 数の上限に達したときに生じます。内部 Queue (bulk.queue) の既定値は 50 です。内部 Queue の Thread 数の上限に達して Queue が溢れた場合には 429 エラー「Too Many Request」が返り、 送信したドキュメントは破棄されてしまいます。Queue の許容量を増やし、リクエストが破棄されることを抑制します。 Orchestrator の利用方法では、Queue サイズの設定を大きくすることで発生するデメリットの存在は考え難いと思われます。

方法

< elasticsearch.yml による設定>

下記の項目を追記または修正します。「-1」は無制限を意味します。

```
threadpool.index.queue_size: -1 # デフォルト 200
threadpool.bulk.queue_size: -1 # デフォルト 50
threadpool.search.queue_size: -1 # デフォルト 1000
```

< API による設定 >
SSH クライアントから API 経由でも設定できます。設定値は上記と同様です。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/_cluster/settings -d '
{
  "transient":{
    "threadpool.index.queue_size": -1,
    "threadpool.bulk.queue_size": -1,
    "threadpool.search.queue_size": -1
  }
}'
```

備考

下記の cURL コマンドおよび出力例で「現在処理待ちの内部 Queue に溜まっているリクエスト数」や「これまでに拒否されたリクエスト数」などを確認できます。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_nodes/stats
```

出力は下記のようになります。

```
... (略) ...
“bulk”:{
 “threads”: 4,
 “queue”: 17, // 現在処理待ちの内部 Queue に溜まっているリクエスト数
 "active": 4,
 "rejected": 333, // これまでに拒否されたリクエスト数
 "largest": 5,
 "completed": 42342
},
... (略) ...
```

3.3.5. Refresh Interval

Elasticsearch は NRT (Near Real-Time Search) 機能を搭載しています。既定ではソフトコミットが毎秒実行され、Elasticsearch に登録された document 等のデータの更新結果が検索に反映されます。

Elasticsearch はデータが登録／更新されるとメモリ上に Index を作成しますが、それとは別に一定時間毎に transaction log としてファイルに出力しています。この transaction log は既定では「30m」すなわち 30 分毎、またはファイルサイズが 200MB に達した場合にディスクへの書き込みが発生します。

Elasticsearch へのデータの登録はバッファに蓄積され、 定期的に Index への反映処理（ソフトコミット）が行われています。 Flush Interval（ソフトコミットによる更新処理間隔）は既定では「1s」すなわち毎秒 1 回実行されます。

優先度：低
検討タイミング：運用後
推奨

推奨 A)または B)のいずれかを設定します。

A) Index 作成中に更新結果をリアルタイムに反映する必要がない場合は、ソフトコミットの実行頻度は「30s」すなわち 30 秒程度に設定します。
※ 単位に「s」を付けない場合は「msec（既定）」になってしまうため注意してください。

B) Bulk API により大量のデータの登録／更新時にファイル I/O のよるパフォーマンス劣化が懸念される場合は、 更新間隔を「-1」すなわち無制限にします。バッファの上限になるまで反映処理を行わないように設定し、Bulk API の実行後に元に戻します。
※ ここで云うところの Bulk API の想定されるユースケースは、Orchestrator からのログ投入が主ではなく、バックアップや検証目的で Index をコピーする等の一時的な操作のことを指します。

理由

Elasticsearch へのドキュメント登録はバッファに蓄積されるのみで、 定期的に Index への反映処理が行われています。 更新処理は既定では毎秒 1 回ですが、特に時間の掛かる Bulk API 実行時にはこの頻度で行う必要はありません。本設定により、高速でログが投入されている時間帯にソフトコミットの実行頻度を下げることで全体のリソース消費を抑えることが期待されます。

方法

< elasticsearch.yml による設定 >

下記の記述を修正または追加します。

A) `Index.refresh_interval: 30s # default 1s`

B) `Index.refresh_interval: -1 # default 1s`

< API による設定 >

下記の cURL コマンドを SSH クライアントから実行します。前者が A)、後者が B)です。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "index" : {
      "refresh_interval" :  "30s"
   }
}'
```

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "index" : {
      "refresh_interval" : -1
   }
}'
```

備考

Elastic 社の公式では、1 つの bulk Import のサイズは 5~15MB 程度にすることを推奨していますが、Orchestrator の運用の観点では特に設定の必要は無いと思われます。

参考

[13] Elasticsearch Reference [7.8] » Set up Elasticsearch » Configuring Elasticsearch » Thread pools
（Elastic 社公式ガイド）

3.3.6. File Descriptors

ファイル記述子（英: File Descriptor）は、OS で開かれているファイルを一意に識別する番号です。File Descriptor の値はリソース（ファイル等）を利用することで増えていきます。この値には上限値が設定されており、プロセス毎に制限されています。

本設定は Linux でのみ考慮すべき設定事項であり、Windows 上で Elasticsearch を実行している場合は無視できます。

優先度：低

検討タイミング：運用後

推奨

Elasticsearch は多数のファイルを扱うため、Elasticsearch の実行ユーザーの開くことができるファイル数（File Descriptor の値）の上限を 65,536 (=2^{16}) 以上に設定します。既定値は 65535 となっています。

理由

File Descriptors の枯渇を防止し、「java.io.IOException: Too many open files」といったエラーの抑制を図ります。

方法

1) SSH クライアントから root 権限で Elasticsearch サーバーにログインします。
2) vi コマンドで起動スクリプト (/etc/init.d/elasticsearch) を開きます。

```
[root@<hostname> ~]# vi /etc/init.d/elasticsearch
```

3) 起動スクリプト内の MAX_OPEN_FILES の値を編集します。ここでは「131071」に設定しています。File Descriptors の枯渇が疑われるエラーに頻繁に遭遇する場合は、この値を徐々に増やします。

```
... (略) ...
# Sets the default values for elasticsearch variables used in this script
ES_HOME="/usr/share/elasticsearch"
MAX_OPEN_FILES=131071
MAX_MAP_COUNT=262144
ES_PATH_CONF="/etc/elasticsearch"

PID_DIR="/var/run/elasticsearch"
... (略) ...
```

備考

< 現在の File Descriptors の上限値の確認方法 >

1) Elasticsearch の PID を確認します。ここでは「37152」でした。

```
[<username>@<hostname> ~]$ cat /var/run/elasticsearch/elasticsearch.pid
37152
```

2) 下記のコマンドから現在の File Descriptor の上限値を確認できます。ここでは ${elasticsearch_PID} に「37152」を代入します。

```
[<username>@<hostname> ~]$ cat /proc/${elasticsearch_PID}/limits
Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
Max data size             unlimited            unlimited            bytes
Max stack size            8388608              unlimited            bytes
Max core file size        0                    unlimited            bytes
Max resident set          unlimited            unlimited            bytes
Max processes             4096                 4096                 processes
Max open files            65535                65535                files
Max locked memory         65536                65536                bytes
Max address space         unlimited            unlimited            bytes
Max file locks            unlimited            unlimited            locks
Max pending signals       64021                64021                signals
Max msgqueue size         819200               819200               bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited            us
```

<現在の File Descriptor 数（開いているファイル数）の確認方法>

i) root 権限で Elasticsearch サーバーにログインします。

ii) 下記のコマンドを実行します。適切な ${elasticsearch_PID} の値を代入してください。ここでは「12345」個のファイルが開かれていることが確認できます。

```
[<username>@<hostname> ~]$ [root@mishimahr-es01 ~]$ ls -l /proc/${elasticsearch_PID}/fd/ | wc
-l
12345
```

参考

[14] Elasticsearch Reference [7.8] » Set up Elasticsearch » Important System Configuration » File Descriptors（Elastic 社公式ガイド）

3.3.7. Merge Thread 数

Elasticsearch の Shard は Lucene Index であり、Lucene Index は Segment に分割されます。Lucene Index では、複数の Document をまとめて Lucene Segment に格納しています。Segment は、一度作成されると中身の Document を編集／削除したりすることのない Immutable なファイルになります。Document に削除があった場合、Segment に削除マークを付与し、一定のタイミングで Merge 処理を実行して Segment の変更／削除を反映させています。

Merge 処理では、小さな Segment をより大きな Segment に Merge したり、削除マークが付与された Document を削除したりしています。Merge 処理は、他の Merge 処理や検索処理等と HW リソースを調整して Merge scheduler で自動実行されます。Merge Scheduler は Merge 操作を必要に応じて自動実行します。Merge 処理は別々の Thread で実行され、Thread 数の上限に達した状態で、それ以上に Merge 処理を実行しようとする場合は、Merge Thread が使用可能になるまで待機します。Merge に使用する Thread 数については、設定を動的に変更することができます。

優先度：低
検討タイミング：運用後
推奨

並列 I/O ができない HDD のような回転するディスクを使用している場合は、Multi-Thread で処理できないため、Merge Thread 最大数を「1」に設定します。

※ SSD を利用している場合は既定で最適化されているため変更は基本的には不要ですが、効果的にチューニングできれば Merge Thread 使用可能待ちを減らすことができ、Merge 処理の実行頻度が向上する可能性があります。既定値は CPU の数に応じて 1～4 が設定されます：

```
Math.max(1, Math.min(4, Runtime.getRuntime().availableProcessors() / 2))
```

方法

cURL コマンドから update-index-settings API を使用して動的に変更することができます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "index" : {
      "index.merge.scheduler.max_thread_count" : 1
   }
}'
```

※ 実運用でこの設定を Index 毎に投入するのは作業が煩雑になる可能性がありますので、§3.7.8 の Index Template の利用を検証しましょう。

参考

[15] Elasticsearch Reference [7.7] » Index modules » Merge（Elastic 社公式ガイド）

3.3.8. search.max_buckets パラメータ

search.max_buckets は v6.2 から実装されたパラメータです。単発のレスポンスにおける aggregation bucket の最大数を定めることができます。v6.2 でこの設定値を有限（例えば 10,000）にしてしまうと、v6.1 まで実行できていたクエリが急に実行できなくなる可能性があったため、v6.2~v6.8 では既定値が「-1」で、search.max_buckets による制限の機能は既定で無効化されている代わりに、10,000 を超える bucket を作る aggregation が実行されると deprecation ログにその旨が記録されます。

優先度：低
検討タイミング：運用後
推奨

下記のような設定が推奨されます。

version	推奨値	既定値	備考
7.9 以降	**65,535**	65,535	Circuit Breaker によるメモリ使用量の見積もり精度が向上したことや、一部の aggregation の実行時のメモリ使用量が少なくなったことで、search.max_buckets の既定値が 10,000 から 65,535 に増加しています。
7.0~7.8	**10,000**	10,000	クエリが実行できない弊害が発生した際に、設定値を 10,000 以上に緩和することを検討します。
6.2~6.8	**10,000**	-1（無制限）	既に稼働中の Elasticsearch クラスタの設定を変える場合は、deprecation ログが出力されていないことを確認します。

理由

設定値が過少な場合、本来は問題ないクエリが実行できなくなってしまう可能性があります。設定値が過大な場合、OOME 発生のリスクが上昇します。

方法

ここでは search.max_buckets を「15,000」に設定する方法を示しています。

1) Elasticsearch サーバーから root 権限で/etc/elasticsearch/elasticsearch.yml を開きます。

```
[root@<hostname> ~]# vi /etc/elasticsearch/elasticsearch.yml
```

2) 下記の一行を追記して保存します。

```
search.max_buckets: 15000
```

備考

§3.3.2 > 備考でも述べていますが、Circuit Breaker の方は、「クエリ実行時に必要とするメモリを見積もり、その値が閾値を超えたら実行させない」という機構で、単に bucket の数に制限を設ける search.max_buckets パラメータの設定は本来不要の筈です。しかし、実際はこの見積もりは難しく、Circuit Breaking Exception が発生することなくクエリが実行されてしまい、OOME が発生することがありました。Circuit Breaker の機構は search.max_buckets よりも先に Elasticsearch に実装されています。

参考

[16] Elasticsearch Reference [7.9] » Set up Elasticsearch » Configuring Elasticsearch » Search settings（Elastic 社公式ガイド）

[17] Elasticsearch Reference [7.9] » What's new in 7.9 > Improve speed and memory usage of multi-bucket aggregations（Elastic 社公式ガイド）

[18] Aggregations enhancement - better memory usage estimates to avoid circuit breaking #28220（GitHub – Elastic 社リポジトリ）

3.3.9. 実行ログの最大表示件数

Orchestrator の[Job] > [Logs]画面での検索結果の表示について、該当の Job の実行ログが 10,000 件（既定）を超えると、Orchestrator にログを表示することができなくなります。ダウンロードの容量とは無関係です。ここでは特に、Orchestrator が実行ログを SQL Server からではなく Elasticsearch から参照している状況を前提とします。

優先度：低
検討タイミング：運用後
推奨

10,000 件以上の実行ログを Orchestrator で参照したい場合は、次のパラメータの両方の値を「10,000」以上に変更します。

A) Elasticsearch の elasticsearch.yml で index.max_result_window パラメータ
B) Orchestrator の Web.config の Logs.Elasticsearch.MaxResultWindow パラメータ

上限を引き上げる場合、JVM Heap の消費量が増えますので変更の際には Heap 使用量を確認しながら調整してください。SQL Server から実行ログを参照している場合、設定項目は B)のみです。

方法

下記の例では、Orchestrator で実行ログを 50,000 件まで参照できるように設定しています。

A1) Elasticsearch サーバーから root 権限で/etc/elasticsearch/elasticsearch.yml を開きます。

```
[root@<hostname> ~]# vi /etc/elasticsearch/elasticsearch.yml
```

A2) 下記の一行を追記して保存します。

```
index.max_result_window: 50000 # default 10000
```

B1) Orchestrator サーバーから%ProgramFiles(x86)%¥UiPath¥Orchestrator¥Web.config を開きます。
B2) 下記の箇所を修正して保存します。

```
  </appSettings>
... (略) ...
    <!-- Logs -->
    <add key="Logs.Elasticsearch.MaxResultWindow" value="50000" />
... (略) ...
  </appSettings>
```

B3) IIS マネージャーから、UiPath Orchestrator サイトを再起動して設定を反映させます。

参考

[19] Elasticsearch Reference [7.8] » Index modules > dynamic index settings（Elastic 社公式ガイド）

3.4. Elasticsearch の JVM 設定

3.4.1. Heap サイズ

Elasticsearch が使う Java の Heap サイズを設定します。既定では、JVM が最大／最小で 1GB の Heap を使用するよう設定されています。

優先度：高
検討タイミング：運用後
推奨

A) 物理メモリの**半分以下かつ 30GB 以下**にします。例えば、32GB メモリの場合には 16GB を、64GB メモリの場合は 28~29GB を Heap メモリを設定します。
B) 最大／最小（初期値）の Heap サイズを等しい値で設定します。同時に、§3.3.1 で述べた Swap を無効にし、JVM の起動時の Heap サイズの値で固定させます。

理由

A) 過剰に大きな値を設定すると Java の GC が走るまでに時間が掛かってしまいます。32GB より大きな容量の物理メモリは「UseCompressedOops によるパフォーマンスの問題」により、ポインタのサイズが 2 倍になり、逆にメモリ消費量が増えてしまいます。
B) 最大／最小の Heap サイズが等しくない JVM を起動すると、システム使用中に Heap サイズが変更され一時停止する可能性があります。

方法

< /etc/elasticsearch/jvm.options による設定>

1) SSH クライアントから Elasticsearch サーバーにログインします。
2) Root 権限の昇格して、vi コマンドで jvm.option ファイルを開きます。

```
[root@<hostname> ~]# vi /etc/elasticsearch/jvm.options
```

3) jvm.option ファイルの該当箇所を編集して保存します。この例では 16GB に設定しています。

```
## JVM configuration
... (略) ...
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms16g
-Xmx16g
... (略) ...
```

備考

< JVM Heap の枯渇 >
JVM Heap が枯渇すると GC により Heap が解放されますが、幾度の GC でも解放が間に合わない場合には Elasticsearch が停止します。このため、JVM Heap サイズは定常的に監視しましょう。Heap が枯渇する要因として、例えば Index を開き続けている、開いている Index のログが多すぎる、JVM の最大メモリサイズが小さいなどが考えられます。GC による回収が十分に行われず Heap が枯渇する場合は、JVM の最大メモリサイズを拡張し、不要な Index を Close/Delete を実施してください。
　Index を Close することでメモリを開放することができ、メモリリソースの逼迫を改善します。Close した Index は Open することで再度利用可能になります。Index を Close することで、存在はするが検索できない状態になります。

< 32GB の補足 >
32GB は-XX:-UseCompressedOops により圧縮 Oops を明示的に無効にした場合の最大 Default Heap サイズで、何もオプションを付けずに起動した場合は 29GB が上限です。

< Elasticsearch 7.7 以降の JVM Heap >
Elasticsearch 7.7 から更に効率的に Heap 領域が利用できるようになり、多くのデータを抱えられるようになりました。

参考

[20] Elasticsearch: The Definitive Guide [master] » Administration, Monitoring, and Deployment » Production Deployment » Heap: Sizing and Swapping（Elastic 社公式ガイド）
[21] Elasticsearch Reference [7.8] » REST APIs » Index APIs » Open index API（Elastic 社公式ガイド）
[22] Coming in 7.7: Significantly decrease your Elasticsearch heap memory usage（Elastic 社公式ブログ）

3.4.2. Metaspace (Off-Heap)

Elasticsearch のような Java アプリケーションは基本的に複数のファイルに分かれた数多のクラスによって成り立っています。Java アプリケーションを動かすためには、これらのクラスを JVM が読み込まなければなりません。Java では JVM のクラスローダと呼ばれる機構がこの処理を担います。クラスローダは、読み込むべきクラスを判断してストレージから当該クラスファイルを探し出し、ファイルを開いてクラスをロードします。
　Java アプリケーションである Elasticsearch も、ロードしたクラスデータを利用します。Java8 以降ではこれらクラスのメタデータを **Metaspace** と呼ばれる「**Off-Heap 領域（Native メモリ領域）**」で使用します。Elasticsearch において Off-Heap が利用される具体的な処理としては、Bulk API 利用や Garbage Collection 実行時の JIT(Just-In-Time)コンパイラの **Code Cache** があります。Metaspace はクラスのメタデータを格納する場所であって、インスタンス情報は Heap 領域に格納されます。
　Metaspace の既定の最大値は 18446744073709547520 (Bytes) = 18 (EB; Exa Byte)であり、実質的に上限はありません。

優先度：高
検討タイミング：運用後

推奨

A) 利用環境に依存しますが、後述の Permanent 領域の規定値である 60～80MB 以上かつ Heap 最大値未満に設定します。Metaspace の枯渇によって JVM で OutOfMemoryError が発生する場合は徐々に Metaspace の最大値を増やします。

B) Elasticsearch を長期に亘って再起動しない運用を前提とする場合、Code Cache の Flush を無効にして、 Code Cache サイズ（**ReservedCodeCacheSize**）を 128MB 以上で設定します。利用環境によってパフォーマンスを監視しながら、更に増やしていくかを検討します。

理由

A) Metaspace 領域は、Java 8 より以前では Permanent 領域と呼ばれていました。Permanent 領域は「-XX:MaxPermSize」で設定する必要があり、既定で最大値（最大容量）が有限でした。しかし、Java 8 以降の Metaspace 領域は既定ではメモリが許す限り際限なく消費されてしまうため、最大値を指定してキャップすることが推奨されます。

B) Java VM は一定の規則に従って JIT を使用し、実行時にバイトコードからマシン語へコンパイルを行っています。これにより高速な処理が可能になっています。このコンパイル結果を保持しているのが Code Cache です。既定では、Code Cache を格納できる最大サイズである **ReservedCodeCacheSize** が **CodeCacheMinimumFreeSpace** （既定では 500KB）を下回って枯渇しそうになると JIT コンパイルが停止され、Code Cache の利用領域が一定以上確保できるまで不要なコンパイル結果を削除します（**CodeCacheFlusing 機能**）。そのため、JIT コンパイルが再開されるまでインタプリタによる動作になるため非常に低速な処理となります。ReservedCodeCacheSize は OpenJDK 64-Bit1.8.9_232 では既定で 240 MByte が割り当てられています。最大でこの値まで Metaspace（Off-heap 領域）が必要になる可能性があります。

CodeCacheFlusing 機能は既定で有効になっており、その挙動は大まかに下記の流れになります：

i) ReservedCodeCacheSize が CodeCacheMinimumFreeSpace 以下になると JIT コンパイラを停止します。

ii) 不要と判断されたコンパイル済みコードを破棄します。

iii) CodeCache が CodeCacheFlushingMinimumFreeSpace 以下まで削減されると JIT コンパイラが活動を再開します。

方法

1) SSH クライアントから Elasticsearch サーバーにログインします。
2) Root 権限の昇格して、vi コマンドで jvm.option ファイルを開きます。

```
[root@<hostname> ~]# vi /etc/elasticsearch/jvm.options
```

3) jvm.option ファイルの該当箇所に、例えば下記のように追記または編集して保存します。
   - 「-XX:MaxMetaspaceSize=16g」を追記して Metaspace の最大サイズ を 16GB します。
   - 「-XX:-UseCodeCacheFlushing」を追記します。JIT コンパイラのシャットダウン前に CodeCache の Flush されることを禁止し、事前に JIT コンパイラを停止させます。JIT コンパイラ停止時にログが出力されるようになります。
   - 「-XX:ReservedCodeCacheSize=128m」を追記して、JIT コンパイルされた Code の最大 Code Cache サイズを 128MB にます。InitialCodeCacheSize の値（既定値は 500KB）以上かつ ReservedCodeCacheSize の上限値である 2GB 以下で設定してください。

```
## JVM configuration
... (略) ...
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms16g
-Xmx16g
-XX:MaxMetaspaceSize=16g
-XX:-UseCodeCacheFlushing
-XX:ReservedCodeCacheSize=128m
... (略) ...
```

※ 「-XX:MaxMetaspaceSize」オプションで指定できる範囲は以下になります：

JDK/JRE Bitness	最小値	最大値（上限）
32bit	256KB	4GB
64bit	256KB	8,589,934,592GB

オプションが指定されなかった場合は実質無制限（OS が領域を獲得できれば上限なし）に設定されます。

備考

< 現在の MaxMetaspaceSize の確認方法 >

SSH クライアントから下記のコマンドを実行します。

```
[<username>@hostname ~]$ java -XX:+PrintFlagsFinal | grep Metaspace*
```

下記のような出力が得られます。「size_t MaxMetaspaceSize」が現在の MaxMetaspaceSize を表します。

```
size_t InitialBootClassLoaderMetaspaceSize  = 4194304                 {product}     {default}
size_t MaxMetaspaceExpansion                = 5451776                 {product}     {default}
uintx MaxMetaspaceFreeRatio                 = 70                      {product}     {default}
size_t MaxMetaspaceSize                     = 18446744073709547520    {product}     {default}
size_t MetaspaceSize                        = 21807104                {pd product}  {default}
size_t MinMetaspaceExpansion                = 339968                  {product}     {default}
uintx MinMetaspaceFreeRatio                 = 40                      {product}     {default}
bool UseLargePagesInMetaspace               = false                   {product}     {default}
... (略) ...
```

< FullGC を発生させる閾値となる Metaspace サイズ >
既定値および指定できる範囲は下記の通りです。オプションが指定されなかった場合は実質無制限（OS が領域を獲得できれば上限なし）に設定されます。ただし、指定値が「-XX:MaxMetaspaceSize」オプションで指定された値よりも大きな値の場合、「-XX:MaxMetaspaceSize」オプションで指定された値となります。

JDK/JRE Bitness	JVM タイプ	既定値	最小値	最大値（上限）
32bit	Java HotSpot Client VM	12MB	256KB	4GB
	FJVM（既定）	16MB		
64bit	FJVM（既定）	20.796875MB	256KB	8,589,934,592GB

Metaspace サイズを 4GB に設定する場合、“-XX:MetaspaceSize=4g”と指定します。この場合、Metaspace サイズが初めて 4GB に達した段階で FullGC が走ります。

64bit の JDK/JRE 8 以降では、圧縮オブジェクトポインタ機能が有効な場合 (-XX:+UseCompressedOops)、Metaspace 領域内に Compressed Class Space 領域が作成されます。この領域に、Java Heap 内のオブジェクトから参照されるクラス情報が配置され、JVM 起動時に 1GB だけ確保されます。

参考

[23] Honey, you have changed quite a bit lately, haven't you?（Elastic 社公式ブログ）
[24] https://docs.oracle.com/javase/jp/8/docs/technotes/tools/unix/java.html（Oracle 社公式ページ）
[25] Using Native Memory by JVM（Classmethod 社公式ブログ）

[26] https://software.fujitsu.com/jp/manual/manualfiles/m170006/b1ws1303/01z200/b1303-00-11-04-01.html#chart_metaspace_size8

3.4.3. GC ログ

ガーベジコレクション（英: Garbage Collection; GC）が走った際の情報はログ出力させることができます。出力された GC ログは、例えば、GCViewer ツールを利用して分析することができます。

優先度：低
検討タイミング：運用後
推奨

GC ログを適切に取得するように設定します。

理由

GC ログを取得して確認すると、Heap メモリが足りなくなった原因を調査することができます。Heap 使用量の上昇の緩急とその原因を知るための手掛かりになります。緩やかな上昇であった場合、オブジェクト参照の解放漏れなどが原因なので解析は少し難しくなる傾向にあります。急激な上昇が見られた場合は、一般にデータベースやファイルから一度に大量のデータを読み込んだことが原因と思われます。

方法

< /etc/elasticsearch/jvm.options による設定>
jvm.options ファイルに下記の内容を追記または修正します。

A) -verbose:gc
起動時に本オプションを付与することで、GC ログが標準出力されます。
B) -Xloggc（取得先パス）
GC ログを単独で取得したい場合は、本オプションで GC ログの出力先を指定します。例えば /var/log/elasticsearch/gc.log.`date +%Y%m%d%H%M%S` を指定します。ここでは GC ログファイル名の末尾に日付情報を付与することで、GC ログが上書きされるのを防いでいます。
C) -XX:+PrintGCDetails
GC ログに詳細な情報を吐き出すようにします。GC アルゴリズムや JVM バージョンによって追加される情報は変わりますが、例えば GC スループットの情報が出力されます。
D) -XX:+PrintGCDateStamps
GC ログを絶対時間で出力します。「-XX:+PrintGCTimeStamps」オプションは JVM 起動時の時間からの相対時間で出力されます。
E) -XX:+PrintTenuringDistribution
オブジェクトの年齢情報を出力します。すなわち、各オブジェクトが 経験した Scavenge GC の回数の統計情報を出力します。
F) -XX:+UserGCLogFileRotation
GC ログを 1 つのファイルに延々吐き続けるのではなく、ログローテーションを有効にします。
G) -XX:NumberOfGCLogFiles
ログローテーションさせる際に最大何個の GC ログファイルでローテーションさせるかを設定します。下記の例では 5 個に設定しています。

H)　-XX:GCLogFileSize
　　ローテーションが発生するログサイズを設定します。下記の例では 1 GB に設定しています。

設定例は以下のようになります。

```
... (略) ...
## JDK 8 GC logging
-verbose:gc
-XX:+PrintGCDetails
-XX:+PrintGCDateStamps
-XX:+PrintTenuringDistribution
-XX:+PrintGCApplicationStoppedTime
-Xloggc:/var/log/elasticsearch/gc.log.`date +%Y%m%d%H%M%S`
-XX:+UseGCLogFileRotation
-XX:NumberOfGCLogFiles=5
-XX:GCLogFileSize=1024M
... (略) ...
```

参考

[27]　How do I enable Java garbage collection logging?（Red Hat 公式ページ）
[28]　https://software.fujitsu.com/jp/manual/manualfiles/m190005/b1ws1318/03z200/b1318-a-06-03.html

3.4.4. Heap Dump

Java で FullGC が頻発したり、メモリリークしてたり、busy Thread で逼迫している状況が発生する場合があります。その際、調査に必要な Dump を取得してから再起動する対応が一般的です。Heap Dump を取得することで、Heap に割り当てられたオブジェクト情報（メモリマップ）を取得することができます。

優先度：低
検討タイミング：運用後
推奨

OutOfMemoryError (OOME) が発生したタイミングで、自動的に Heap ダンプが取得されるように設定します。定期的に削除して、ストレージ容量が逼迫しないように注意してください。出力先のストレージ容量に余裕が無い場合は、出力先を変更するか、再現検証などの場合にのみ有効にするようにしてください。

※　Heap Dump 取得中は Elasticsearch が停止します。また Heap サイズによっては数十分の時間が掛かることがあります。

理由

Heap 起因の詳細なトラブルシュートに活用します。

方法

< /etc/elasticsearch/jvm.options による設定 >

1) -XX:+HeapDumpOnOutOfMemoryError を追記します。
2) -XX:HeapDumpPath（Heap ダンプの出力先パス）で指定したディレクトリに java_pid<pid>.hprof の名前で Heap ダンプが出力されます。出力先パスを指定しない場合、Java を起動したカレントディレクトリに出力されるので、-XX:HeapDumpPath により明示的に指定しましょう。

```
... (略) ...
## heap dumps

# generate a heap dump when an allocation from the Java heap fails
# heap dumps are created in the working directory of the JVM
-XX:+HeapDumpOnOutOfMemoryError

# specify an alternative path for heap dumps; ensure the directory exists and
# has sufficient space
-XX:HeapDumpPath=/var/lib/elasticsearch
... (略) ...
```

備考

< 任意のタイミングで Heap Dump を出力する方法 >

Heap ダンプを任意に取得する場合は、Java JDK に同梱されている jmap を実行します。

```
[root@<hostname> ~]# vi jmap -F -dump:format=b,file=<file name> (pid)
```

実行できない場合は、環境変数を確認するか、フルパスで実行してください。

< Heap Dump が出力されないケース >

OutOfMemoryError はユーザのプロセス生成数上限 (ulimit -u) に達して Native Thread が生成できなくなったときにもスローされますが、この場合は Heap Dump は出力されません。プロセス生成数上限に達した場合は以下のようなメッセージが出力されます。

```
java.lang.OutOfMemoryError: unable to create new native thread
```

3.5. Elasticsearch の Shard 設定

3.5.1. Primary Shard 数

適切な Primary Shard（以下、単に Shard と記述します）数の設定は、Elasticsearch の性能に大きく影響しますので、必ず考慮してください。最新の Elasticsearch 7.x 系では既定の Shard 数が 5 -> 1 に修正されました。

バージョン	既定値
Elasticsearch 7.x	# of Shards = 1
Elasticsearch 6.x 以前	# of Shards = 5

優先度：高

検討タイミング：運用後

推奨

- 1 Index が 30GB（目安）を超える場合には Shard 数を 1 増やします。すなわち、「Shard 数≒Index サイズ[GB]/30GB」となるように Shard 数を調整します。
- Index の作成が日次で、1 Shard 辺りのサイズが 3 GB 以下の場合、月次での作成に変更します。Daily であれば削除対象が細かくなり、運用の柔軟性が上がります。

理由

Elasticsearch 6.x 以前での既定の Shard 数である「5」は非常に大きな値です。特筆して大規模な UiPath Robot の運用をしている環境でもない限り、Elasticsearch の性能に悪影響を与える可能性があります。Elasticsearch の内部実装として、Shard 数毎にインスタンス生成があり、Shard が増えるほど Java Heap の消費量が増えるため、Heap 枯渇の危険性が高まります。また、Shard を配置できずに Cluster health が絶えず Yellow となる可能性が高いです。

逆に Shard 数が少なく、例えば 1 つの Shard サイズが 40GB を超える場合は、Elasticsearch の検索の性能が発揮されなくなる可能性があります。

方法

< cURL コマンドによる設定 >

SSH クライアントから下記の cURL コマンドにより Shard 数を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "index" : {
      "number_of_shards" : 2
   }
}'
```

備考

< Shard の確認 >
SSH クライアントから下記の cURL コマンドにより Shard 数を確認できます。Kibana の DevTools を使う場合は「GET _cat/shards」で確認できます。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_cat/shards?v
```

例えば下記のような出力結果が得られます。一行は Index 名、Shard 数、プライマリ(p)／レプリカ(r)、ステータス、Document 数、サイズ、IP アドレス、node 名の順に並んでいます。

Index	shard	prirep	state	docs	store	ip	node
default-2020.04	0	p	STARTED	0	283b	127.0.0.1	mishimahr-es01
default-2020.03	1	p	STARTED	10	50.8kb	127.0.0.1	mishimahr-es01
.monitoring-es-7-2020.04.02	0	p	STARTED	81495	43.7mb	127.0.0.1	mishimahr-es01
.apm-agent-configuration	0	p	STARTED	0	283b	127.0.0.1	mishimahr-es01
.kibana_1	0	p	STARTED	13	56kb	127.0.0.1	mishimahr-es01
... (略) ...							

参考

[29] Elasticsearch Reference [7.8] » Index modules > static index settings（Elastic 社公式ガイド）
[30] Elasticsearch クラスターのシャード数はいくつに設定すべきか？（Elastic 社公式ブログ）

3.5.2. Replica Shard 数

Replica Shard は Primary Shard のコピーになります。Replica Shard は Cluster 構成の場合のみ作成することができます。
Replica Shard を各ノードに配置することで負荷分散により検索性能を向上させることができます。ただし、その分だけディスク容量が必要になる点には注意が必要です。

優先度：低
検討タイミング：運用後
推奨

Elasticsearch が冗長構成の場合には、Replica Shard 数を 1 以上に変更します。

方法

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより Replica Shard 数を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "index" : {
      "number_of_replicas" : 1
   }
}'
```

備考

巨大な Index の構築中は Replica Shard を作らないようにすることで、Index 構築時の負荷を軽減して処理が高速化されることが期待できます。

3.5.3. 累積 Shard のパージ

ノードに保持できる Shard 数は、利用できる JavaHeap 量に比例しますが、Elasticsearch によって強制される固定の上限はありません。

優先度：低
検討タイミング：運用後
推奨

ノード毎の累積 Shard 数は、構成した JavaHeap 1 GB 当たり 20 個未満に維持するように定期的に削除またはバックアップします。

理由

例えば、30GB の Heap では累積最大 600 Shards となりますが、この上限よりも大幅に下回る数に維持させることで Cluster を良好な状態に維持できます。

方法

古い index を定期的に削除することで、累積 Shard 数を抑えます。

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより Index を削除できます。

```
[<username>@<hostname> ~]$ curl -X DELETE http://localhost:9200/<index_name>
```

3.6. Elasticsearch のストレージ設定

Elasticsearch のストレージに関わる推奨設定を説明します。

3.6.1. ストレージ構成

Index が保存される Data Node のストレージタイプは indexing の性能に多大な影響を及ぼします。

優先度：高

検討タイミング：運用前（／運用後）

推奨

下記の点に留意してください。

a) HDD を使用する場合は 15kRPM 以上が強く推奨されますが、可能な限り SSD を利用します。
b) SSD について、特に DAS(Direct Attached ストレージ)で NVMe 形式が推奨されます。AWS EC2 の場合と i3 インスタンスが該当します。
c) Index をリモートマウントされたファイルシステム（例えば NFS や SMB/CIFS）上に配置せず、ローカルストレージを利用します。
d) 仮想化されたストレージ（Amazon の Elastic Block ストレージなど）の利用に注意しましょう。
e) 複数の SSD を複数の path.data ディレクトリに Index をストライピング（RAID0 のように）させます。

理由

b) SSD は高性能な HDD よりも確実に高速です。ランダムアクセスによる消費電力が低いだけでなく、シーケンシャルな I/O アクセス性能も高いです。indexing、Merge や検索による並列的な I/O も高速のため、パフォーマンスの向上を図ることができます。
d) 仮想化されたストレージでも Elasticsearch で十分に動作しますが、ローカルストレージと比較すると本質的に低速です。最高性能でプロビジョニングされた EBS でさえ、ローカルインスタンスにある SSD よりも低速のようです。ローカルインスタンスにある SSD は物理マシン上のすべての仮想マシンから共有されてアクセスされます。もし他の仮想マシンに急に I/O が集中した場合、突発的なスローダウンとなる恐れもあります。
e) I/O 性能の高速化を図れる一方で、ストライピング先の故障により Index が壊れて Shard を失うリスクを増加させることに注意しましょう。単一の Shard でパフォーマンスを最適化し、異なるノード間でレプリカを追加すると、ノードの故障への冗長化ができます。

参考

[31] Performance Considerations for Elasticsearch Indexing（Elastic 社公式ブログ）

3.6.2. パーティション

HDD や SSD の記憶媒体において論理的に分割された単位領域をパーティションといいます。Linux ではドライブレター（Windows で云うところの「C:」「D:」などの英字 1 文字）という概念は無く、パーティションを区切った場合は任意の場所にマウントして使用します。デバイスは sd{a,b,c,…,z,A,B,…}のように末尾のアルファベット順で認識され、パーティションには更にその末尾に番号が振られます。

1 つの記憶媒体を最大 4 つのパーティションに分割できます。これらのパーティション情報は MBR（英: Master Boot Record; ディスクの一番先頭のセクター）中のパーティションテーブルに格納されます。Linux におけるパーティションの種類は下記表の通りです。

パーティションの種類	説明
基本パーティション	基本パーティションは、1 台のディスクに必ず 1 つ以上が存在します。1 つの物理ディスクは最大で 4 つの基本パーティションに分割することができます。基本パーティションのデバイスファイル名は、SCSI ディスク(/dev/sda)の場合、「/dev/sda{1,2,3,4}」のようになります。
拡張パーティション	基本パーティションのうち、1 つだけを拡張パーティションとして使用することができます。拡張パーティションにはファイルシステムを作成することができない代わりに、論理パーティションが格納されます。
論理パーティション	論理パーティションは、拡張パーティション内に作成されたパーティションです。作成できる論理パーティションの数はハードディスクのタイプにより異なります。論理パーティションのデバイスファイル名は SCSI の場合、作成済みの基本パーティションの数に関係なく、「/dev/sda{5,6,…}」のようになります。

優先度：低
検討タイミング：運用前
推奨

Elasticsearch サーバーのストレージはパーティションをシステム領域とデータ領域に分割し、データはデータ領域にストアします。

※　パーティション設定は、インストールやセットアップ段階で設定された後は基本的に変更できません。厳密には不可能ではないですが非推奨のため、計画性をもってパーティションを設定する必要があります。

備考

パーティションの分割は、主に以下の目的のために利用されます。

- ファイルシステム障害の局所化
- ディスク容量不足によるトラブル防止
  特に Elasticsearch の場合、不意に大量のログが投入されてディスクの空き容量が足りなくなった場合でも、パーティション内に被害を限定でき、システム全体に対する影響を最小限に抑えられます。
- 性能劣化の防止
- 複数 OS の共存

3.6.3. ディスクサイズ

優先度：高

検討タイミング：運用前（／運用後）

推奨

Cluster 全体およびノード 1 台あたりに必要なディスクサイズを下記の計算式で見積もります。

- 必要ディスクサイズ（Cluster 全体）= 1GB x ログ件数/100 万 x (1 + Replica Shard 数)
- 必要ディスクサイズ（ノード 1 台あたり）= 必要ディスクサイズ（Cluster 全体）/ ノード数

恒久的なサイズについて検証環境にてリソースのモニタリングを実施し、見積もっていただくことを推奨します。

理由

目安として 1000 万件のログの格納に約 10GB 必要になります。また、Cluster 構成で Replica Shard を用意する場合は、Cluster 全体としては Replica Shard 数分だけ同じ容量が必要になります。

3.6.4. ディスク空き容量の閾値

Elasticsearch はノードに新しい Shard を割り当てるか、そのノードから Shard を積極的に再配置するかを決定する前に、ノードの使用可能なディスク領域を考慮しています。

優先度：高

検討タイミング：運用後

推奨

Elasticsearch がディスクを効率的に消費するために、環境に応じて適切な「cluster.routing.allocation.disk.watermark.*」オプションと「cluster.info.update.interval」オプションを設定します。各オプションの規定値とその値の意味を下記の通りです。

オプション名	既定値	説明
cluster.routing.allocation.disk.watermark.low	85%	ディスク使用率がこの割合を越えたノードには新規 Shard を割り当てません。
cluster.routing.allocation.disk.watermark.high	90%	ディスク使用率がこの割合を越えたノードは、保持している Shard を別のノードに再配置しようとします。
cluster.routing.allocation.disk.watermark.flood_stage	95%	ディスク使用率がこの割合を越えたノードに保持されているデータが読み取り専用になり、書き込みが禁止されます。これはノードがディスク領域を使い果たすのを防ぐための最終関門です。Index 作成を続行できる十分なディスク領域を確保してから、手動で「読み取り専用」状態を解放する必要があります。
cluster.info.update.interval	30s	Elasticsearch が Cluster 内の各ノードのディスク使用状況をチェックする頻度を指定できます。

理由

1TB ディスクの場合は 85%では 150GB しか空きがありませんが、10TB ディスクの場合は 85%だと 1.5TB も空きができてしまいます。利用しているディスクサイズによって推奨値は異なります。マージ前の Segment サイズや、Shard 再配置に必要な分はディスクに余裕が必要です。また耐障害性も考える場合、1 台が停止しても運用を継続できる余裕は必要です。

　ディスク容量が小さい場合はディスク使用状況をチェック頻度は短くても良いですが、大容量のディスクを利用している場合には長めに設定することもできます。

方法

< elasticsearch.yml から設定する方法 >
以下の設定を追加して Elasticsearch を再起動してください。下記の例では、「cluster.routing.allocation.disk.watermark.*」オプションを絶対値で表現し、チェック頻度は 1 分です。値は割合(%)でも絶対値（例えば 500mb）でも良いですが、混ぜて使うことはできません。

```
cluster.routing.allocation.disk.watermark.low: 850mb
cluster.routing.allocation.disk.watermark.high: 900mb
cluster.routing.allocation.disk.watermark. flood_stage: 950mb
cluster.info.update.interval: 1m
```

備考

< エラー例 >
ストレージの使用量が flood_stage 以上になった場合、例えば下記のようなエラーがスローされます。

```
TransportError(403, u'cluster_block_exception', u'blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];')
```

```
blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];: [cluster_block_exception] blocked by: [FORBIDDEN/12/index read-only / allow delete (api)];
```

< ディスクの閾値超過からの復旧 >
ディスクの閾値 (flood_stage) を超過した場合、「読み取り専用」状態となり、ディスク使用率が閾値未満に回復しても、この「読み取り専用」状態を解除するまでログを格納することができなくなります。SSH クライアントから Elasticsearch サーバーにログインして、下記の cURL コマンドにより「書き込み禁止」状態を解除します。別のサーバーからアクセスする場合は URL を変える必要があります。

```
[<username>@<hostname> ~]$ curl -X PUT -H "Content-Type: application/json" http://localhost:9200/_all/_settings -d '{"index.blocks.read_only_allow_delete": null}'
```

「{"acknowledged":true}」と表示されれば、コマンド処理は成功しています。

参考

[32] Elasticsearch Reference [7.8] » Modifying your data » Disk-based shard allocation（Elastic 社公式ガイド）

3.7. Elasticsearch のスキーマ設定

Elasticsearch は事前のスキーマ定義を経ることなくデータを indexing できる機能（スキーマレス）を持ちますが、データの投入前に厳密にスキーマを定義することが推奨されます。理由としては、スキーマを定義せずに投入されたデータは Elasticsearch が型を判断するためエラーの元になり易いことが挙げられます。

3.7.1. _all フィールド

既定で全フィールドのデータが入る_all という Meta Fields があります。**_all フィールドは Elasticsearch 6.x より廃止**されました。_all フィールドを有効にする方法もありますが、既定で無効になっています。

優先度：低
検討タイミング：運用後
推奨

全フィールドを跨いだ検索を行う予定が無い場合は無効にします。

理由

ディスク容量（Index サイズ）の節約や indexing の高速化を図ります。

方法

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより設定を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "mappings" : {
      "_all" : {"enabled" : false}
   }
}'
```

参考

[33] Elasticsearch Reference [6.8] » Mapping » Meta-Fields » _all field（Elastic 社公式ガイド）

3.7.2. _source フィールド

_source は送信されたドキュメントの元データ（JSON 化される前のログ）を格納する Meta Fields です。既定で有効になっています。

優先度：低
検討タイミング：運用後
推奨

_source はハイライトや Update API、Re-indexing などの用途に利用されますが、このような用途に利用しないのであれば、無効にします。

理由

ディスク容量（Index サイズ）の節約を図ります。ディスク空き容量が切迫していないのであれば、特に無効にする必要はありません。また、_source が無くなると、Discover で生データが無くなり、Elasticsearch から直接データを参照しようとしても検索結果に現れなくなります。よって_source の無効化は恒久的な推奨設定ではなく、データ逼迫時の救済措置として一時的に利用するケースが想定されます。

方法

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより設定を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "mappings" : {
      "_source" : {"enabled" : false}
   }
}'
```

備考

< Index サイズの削減率の目安 >
ログ 300 万件 3GB、Replica Shard 無しの場合に各フィールドの有効／無効による Index サイズの削減率の一例を下記の表に示します。

_all	_source	Index サイズ	削減率
enable	enable	4.1GB	(N/A)
disable	enable	3.1GB	約 24.4%
enable	disable	3.2GB	約 22.0%
disable	disable	2.2GB	約 46.3%

参考

[34] Elasticsearch Reference [7.8] » Mapping » Meta-Fields » _source field（Elastic 社公式ガイド）

3.7.3. stored_fields

既定ではフィールドは indexing されても store はされず、meta の_source としてオリジナルの JSON が store されています。サイズの大きなフィールドがあるなど、選んで store する場合は true にします。

優先度：低
検討タイミング：運用後
推奨

検索のみに使用し、値を取得する必要がないフィールドは store:false とすることでディスク容量を節約します。

方法

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより設定を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "mappings" : {
      "store" : false
   }
}'
```

3.7.4. not_analyzed

string フィールドは既定では Index 時に形態素解析などの処理が行われてしまうため、完全一致検索時に想定しない結果が返ってきたり、Index 処理が重くなる可能性がある。

優先度：低
検討タイミング：運用後
推奨

完全一致の検索のみを行いたい string フィールドに対しては、マッピング時に not_analyzed を付与します。

方法

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより設定を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "string_field" : {
      "index" : "not_analyzed"
   }
}'
```

3.7.5. doc_values

Mapping の際に doc_values:true を設定すると、cache をメモリではなくディスクに乗せることができる。メモリに乗せる場合よりも 20%程低速になるが、Heap メモリに影響されなくなり、挙動が安定する。尚、not_analyzed ではない string フィールドには適用できないので注意されたい。

優先度：低
検討タイミング：運用後
推奨

Mapping の際に各フィールドに doc_values:true を設定しておくと、field data cache をメモリではなくディスクに乗せることができます。メモリに乗せる場合よりも 20%程低速になりますが、Heap メモリに影響されなくなるのでメモリを増やせない場合に有効です。設定後は Re-indexing が必要です。データ量が多い場合に検討します。

方法

< cURL コマンドによる設定 >
SSH クライアントから下記の cURL コマンドにより設定を変更できます。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/<index_name>/_settings -d '
{
   "string_field" : {
      "doc_values" : true
   }
}'
```

備考

doc_values を使わない場合は、OutOfMemoryError を避けるために§3.3.2 の indices.fielddata.cache.size を設定してください。

参考

[35] Disk-Based Field Data a.k.a. Doc Values（Elastic 社公式ブログ）

3.7.6. Index Aliases

Elasticsearch を運用する際、OS のアップデート、Orchestrator のバージョンアップ、運用ルール変更などに伴い、運用中であっても Index や検索クエリを変更していく作業は日々発生します。この場合、Index を再構築する必要性も出てきますが、再構築して整合性のあるデータセットを準備するには時間を要します。その都度メンテナンスを行うのは運用コストを増大させる恐れがあります。

優先度：低

検討タイミング：運用後

推奨

Elasticsearch で Orchestrator から投入された Index を参照する際に、Index 名を直接参照するのではなく Alias を設定し、それらを参照することで、Elasticsearch サービスを停止させることなく Index をメンテナンスできるようにします。

理由

Index Aliases を利用することで、Index の無停止再構築を行うことができます。直接 Index を利用するのではなく、Index のメンテナンス時に Alias を切り替えることで Index の無停止再構築を実現します。

方法

< cURL コマンドによる設定 >

SSH クライアントから下記の cURL コマンドにより設定を変更できます。以下に 2 つの例を示します。

A) test_index に紐付いていた Alias を test_index_2 に切り替える際のコマンドです。Elasticsearch/Kibana から常に test_index_alias を参照するようにしておけば、シームレスに新しい Index を参照できるようになります。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/_aliases -d '
{
 "actions" : [
  {"remove": {"index": "test_index", "alias": "test_index_alias" }},
  {"add": {"index": "test_index2", "alias": "test_index_alias" }}
 ]
}'
```

B)　直近 3 日間の Index を「log-recent」という Alias で指定します。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/_aliases -d '
{
 "actions" : [
  {"add": {"index": "log-2020.03.17", "alias": "log-recent" }},
  {"add": {"index": "log-2020.03.18", "alias": "log-recent" }},
  {"add": {"index": "log-2020.03.19", "alias": "log-recent" }}
 ]
}'
```

参考

[36]　Update index alias API（Elastic 社公式ガイド）
[37]　Changing Mapping with Zero Downtime（Elastic 社公式ブログ）

3.7.7.　ignore_above

ignore_above の設定値より長い文字列は、indexing も格納もされません。文字列の配列の場合、ignore_above は各配列要素に個別に適用され、ignore_above より長い文字列要素は indexing されず、格納されません。ignore_above の既定値は 256 です。

優先度：低
検討タイミング：運用後
推奨

ignore_above の値を 1024 以上に設定します。

理由

ignore_above の設定値より長い文字列は aggregate 等の処理に使うことができません。

参考

[38]　Elasticsearch Reference [7.9] » Mapping » Mapping parameters » ignore_above

3.7.8.　Index Template

Index Template は Index の定義を指定します。共通の構成要素を自動で適用できます。

優先度：高
検討タイミング：運用後
推奨

お客様の利用ケースを踏まえた Template を策定して設定します。詳細は『Orchestrator 導入ステップバイステップガイド [2020.4 対応版]』を参照してください。

理由

Index 作成の度に明示的に field type の設定をする手間を省くことができます。

方法

Kibana から登録できます。詳細は参考のリンクを参照ください。

参考

[39] Elasticsearch Index Template のバージョニング（Classmethod 社公式記事）

3.8. Elasticsearch の監視

様々なメトリックス情報を監視することで Elasticsearch が正常に動作しているか、将来障害が発生し得ないかを推察することができます。また、Index で消費しているストレージサイズを確認することで次月（設定によっては次週や翌日など）の予測を立て、Shard 数などの再設計などにも役立てることができます。

3.8.1. ディスクの空き容量

優先度：高

検討タイミング：運用後

推奨

ディスクの空き容量が枯渇する（ディスクフルになる）と Elasticsearch はログデータを格納することができなくなります。またシステム全体にも影響を与えますので、ディスクフルにならないように空きディスク容量を必ず監視し、定期的に Index 削除などのメンテナンスを行ってください。

また、Elasticseach ではディスクの閾値設定を行うことで、ディスクフルになることを防ぐ機能があります。閾値の設定については、§3.6.3 を参照してください。

3.8.2. Cluster のステータスとノードの可用性

優先度：低

検討タイミング：運用後

推奨

Cluster の状態を把握するには、すべてのノードの CPU 使用率メモリ使用率ディスク I/O 等、各 Elasticsearch ノードのステータスをリアルタイムで監視することが推奨されます。CPU 使用率が急上昇した場合は、まず JVM を調べます。Elasticsearch は JVM で実行されるため、Elasticsearch のメモリ使用量を監視するには、JVM メモリと GC の統計を確認しましょう。

3.8.3. Index のパフォーマンス

優先度：高

検討タイミング：運用後

推奨

indexing レートの乱高下が発生した場合は、データソースに問題が発生している可能性があります。また、Cluster 全体のパフォーマンスは、Refresh 時間と Merge 時間の影響を受けること場合があります。各ノードの平均の Query レイテンシを監視していくと良いでしょう。

3.8.4. 検索のパフォーマンス

優先度：低

検討タイミング：運用後

推奨

Elasticsearch の監視で重要な検索のパフォーマンス指標は以下の 2 つです：

クエリレイテンシ時間とリクエストレート

クエリのパフォーマンスに影響を与える可能性のある要素として、例えば、誤った内容のクエリ、非効率なクエリ、Elasticsearch Cluster、JVM メモリ、GC 等が挙げられます。クエリレイテンシは、ユーザーに直接影響を与える要素です。状態を常に監視し、異常事態の発生時に

は直ぐに対処します。クエリレイテンシと共にリクエストレートを監視することで、システムがどの程度使用されているかも把握できます。

フィルタキャッシュ

一般に、フルテキスト検索か関連するスコアに影響する条件がある場合にクエリを使用し、それ以外はフィルタを使います。検索結果にどのくらい関連性が高いかを示すスコアが重要な場合は クエリを使用し、LIKE 検索であれば フィルタを使用しましょう。

Elasticsearch のフィルタは既定でキャッシュされます。フィルタを使用してクエリを実行すると、Elasticsearch はフィルタに一致する Document を検索し、その情報を使用して「BitSet」と呼ばれる構造を作成します。その後のクエリ実行時に、同じフィルタが設定されている場合は、BitSet に格納されている情報が再利用されます。これにより、I/O 処理と CPU サイクルが節約され、クエリの実行速度が向上します。フィルタキャッシュの状況を把握し、実行速度改善の仕組みが機能しているかを確認できることが望ましい。

参考

[40] All About Elasticsearch Filter BitSets（Elastic 社公式ブログ）
[41] Frame of Reference and Roaring Bitmaps（Elastic 社公式ブログ）

3.8.5. Heap の監視

優先度：高
検討タイミング：運用後
推奨

Heap の枯渇によるパフォーマンスの著しい劣化を事前に検知できるように定常的に監視することが推奨されます。

備考

Elasticsearch ログに下記のような一行が出力される場合があります。下記の例では Heap の枯渇が強く疑われます。2019-07-09T14:26:17 の時点で「45.5 秒かかる Full GC によって、合計 15.9GB の Heap の内 15.5GB から 15.7GB までしか Heap 使用量を減らすことができなかった」ことが分かります。Full GC を試みたものの、Heap 使用量の削減が間に合わなかったことを意味します。

```
[2019-07-09T14:26:17,688][WARN ][o.e.m.j.JvmGcMonitorService] [APLRPALOG000002] [gc][old][4102487][17] duration [45.5s], collections [1]/[45.9s], total [45.5s]/[6.4m], memory [15.5gb]->[15.7gb]/[15.9gb], all_pools {[young] [245mb]->[364.2mb]/[532.5mb]}{[survivor] [0b]->[0b]/[66.5mb]}{[old] [15.3gb]->[15.3gb]/[15.3gb]}
```

また、下記のような別の例では「Java Heap メモリが枯渇し始め、GC のオーバヘッドが高い」状態に陥っていることを意味します。

```
[2019-07-09T13:38:49,214][WARN ][o.e.m.j.JvmGcMonitorService] [APLRPALOG000002] [gc][4100044] overhead, spent [631ms] collecting in the last [1s]
```

Elasticsearch の既定の GC は CMS (Concurrent-Mark & Sweep) GC です。Full GC は Heap が完全に枯渇し始めるとトリガーされることから、ログに記録されている時間帯では常に Heap が逼迫していたと推測できます。

3.8.6. ネットワークと Thread Pool

優先度：低

検討タイミング：運用後

推奨

メモリ使用量が乱高下している場合や、長時間の GC が発生している場合に確認します。特に、ボトルネックとして GC が疑われる場合に確認します。GC が多発する原因として下記のような場合が考えられます。

- ある 1 つの特定のプールがストレスを受けている。
- JVM のメモリ不足、等。

3.9. Elasticsearch のメンテナンス

Index データファイルを直接コピーしてバックアップした場合、正しくリストアされませんのでご注意ください。Curator または API 経由でバックアップを行ってください。Curator は Elastic 社が提供する Python 実装の運用支援ツールです。

Elasticsearch が稼働するサーバーを安定的に運用するには、Index のメンテナンスが欠かせません。Elasticsearch は検索対象の Document をメモリに展開します。長期間の運用でメンテナンスを考慮しないと、メモリを過剰に消費して Heap Dump を吐いてサービスが落ちる場合があります。定期的に古いログを削除するなどのメンテナンスが必要です。

参考

[42] Curator Reference [5.8] » About（Elastic 社公式ガイド）
[43] Curator Reference [5.8] » Installation（Elastic 社公式ガイド）
[44] UiPath Orchestrator システムの基盤設計・運用ガイド[第 2 版]（UiPath 公式ガイド）

3.9.1. Index の close

優先度：高
検討タイミング：運用後
推奨

月の切り替わりなどで、参照しなくなった古い Index については close を行ってください。Index を close すると、そのデータは参照できなくなります。

理由

Index を open していると Elasticsearch はデータをメモリに保持し続けるため、Java Heap の枯渇を誘引します。閉じた Index は読み取り／書き込み操作が禁止されます。Document に Index を付けたり、close した Index 内の Document を検索することはできません。close した Index は、Document の Index 作成／検索のために内部データ構造を維持する必要がなくなり、cluster のオーバーヘッドが抑えられます。

方法

< Curator による設定 >

1) テキストエディタで例えば以下の内容の Action ファイルを作成します。ファイルの拡張子は .yml とします。 なお日数にはログを open したい月数 x31 を指定してください。 Index の単位を日単位にしている場合でも close の単位が月単位であればそのまま利用できますが、任意の日数単位で削除したい場合は timestring: を '%Y.%m.%d' に変更する必要があります。例えば 2 カ月より前の Index を close させる場合は以下のように作成します。 日数を表す unit_count は 31x2=62（２ヵ月）です。このファイルで 2020 年 4 月に実行した場合は、 default-2020.04 と default-2020.03 の Index を残し、それより古い Index が close されます。

```
actions:
  1:
    action: close
```

```
  description: "Close indices"
  options:
    ignore_empty_list: True
    continue_if_exception: False
  filters:
    - filtertype: pattern
      kind: prefix
      value: "default-"
    - filtertype: age
      source: name
      direction: older
      timestring: '%Y.%m'
      unit: days
      unit_count: 62
```

2) ファイルを作成したら以下のコマンドを実行します。

```
Curator <作成した Action ファイルのパス>
```

以上で Index の close は終了です。

< cURL コマンドによる API での設定 >

7 系より cURL コマンドによる Index の close が可能になりました。Curator を利用しない場合は SSH クライアントからの cURL コマンドや Kibana の Dev Tools 等で直接 Elasticsearch の API を利用します。

i) 以下の API で Index の一覧を取得します。

```
[<username>@<hostname> ~]$ curl -X GET
http://localhost:9200/_cat/indices/default*?s=index&h=index
```

ii) 以下は出力例です。

```
default-2020.01
default-2020.02
default-2020.03
default-2020.04
```

iii) Open 状態である必要が無い Index があれば以下の API で Index を close します。open したい場合は「/_close」を「/_open」に読み替えます。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/<Index_name>/_close
```

iv) API の応答を確認して、該当の Index が close されたことを確認します。

```
{
  "acknowledged": true,
  "shards_acknowledged": true,
  "indices": {
    " default-2020.01": {
      "closed": true
    }
  }
}
```

以上で Index の close は終了です。

参考

[45] Elasticsearch Reference [7.x] » REST APIs » Index APIs » Close index API（Elastic 社公式ガイド）

3.9.2. Index の削除

優先度：高

検討タイミング：運用後

推奨

不要な Index の削除を定期的に行ってください。

理由

Index を close しても、Index を残し続けるとディスクの枯渇を誘引します。

方法

< Curator による設定 >

1) テキストエディタで例えば以下の内容の Action ファイルを作成します。ファイルの拡張子は .yml とします。なお日数にはログを削除せずに残したい月数×31 を指定してください。Index の単位を日単位にしている場合でも削除の単位が月単位であればそのまま利用できますが、任意の日数単位で削除したい場合は timestring: を '%Y.%m.%d' に変更する必要があります。例えば 2 カ月より前の Index を close させる場合は以下のように作成します。日数を表す unit_count は 31x2=62（2 ヵ月）です。このファイルで 2020 年 4 月に実行した場合は、default-2020.04 と default-2020.03 の Index を残し、それより古い Index が削除されます。

```
actions:
  1:
    action: delete_indices
    description: "Delete indices"
    options:
      ignore_empty_list: True
      continue_if_exception: False
    filters:
      - filtertype: pattern
        kind: prefix
        value: "default-"
      - filtertype: age
        source: name
        direction: older
        timestring: '%Y.%m'
        unit: days
        unit_count: 62
```

3) ファイルを作成したら以下のコマンドを実行します。

```
Curator <作成したファイルのパス>
```

以上で Index の削除は終了です。

< cURL コマンドによる API での設定 >
Curator を利用しない場合は SSH クライアントからの cURL コマンドや Kibana の Dev Tools 等で直接 Elasticsearch の API を利用します。

i) 以下の API で Index の一覧を取得します。

```
[<username>@<hostname> ~]$ curl -X GET
http://localhost:9200/_cat/indices/default*?s=index&h=index
```

ii) 以下は出力例です。

```
default-2020.01
default-2020.02
default-2020.03
default-2020.04
```

iii) 不要な Index があれば以下の API で Index を削除します。Index 名はカンマ区切りやワイルドカード (*) で複数指定することができます。

```
[<username>@<hostname> ~]$ curl -X DELETE http://localhost:9200/<Index_name>
```

例えば default-2020.01 と default-2020.02 の 2 つの Index を削除する場合は、下記のように Index 名をカンマで区切ることで指定できます。

```
[<username>@<hostname> ~]$ curl -X DELETE http://localhost:9200/default-2020.01,default-
2020.02
```

default-2019 で始まる Index をすべて削除する場合は以下のように指定できます。

```
[<username>@<hostname> ~]$ curl -X DELETE http://localhost:9200/default-2019*
```

iv) Index を削除したら再度一覧を表示して、削除した Index が表示されなくなっていることを確認してください。

```
[<username>@<hostname> ~]$ curl -X DELETE
http://localhost:9200/_cat/indices/default*?s=index&h=index
```

以上で Index の削除は終了です。

備考

Kibana を利用している場合、Elasticsearch v6.6 から導入された Index Lifecycle Management (ILM) 機能を使用し、Index を自動的に定期削除することが可能です。詳細は Elastic 社公式ページをご参照ください。

参考

[46] Elasticsearch Reference [7.8] » ILM: Manage the index lifecycle（Elastic 社公式ガイド）

3.9.3. リポジトリの作成

優先度：高

検討タイミング：運用後

推奨

スナップショットの運用を開始する前に実施します。

理由

Elasticsearch では、スナップショットを利用するために事前にリポジトリと呼ばれる領域が必要になります。

方法

< Curator による設定 >

1) 以下のコマンドを実行してリポジトリを作成します。

```
es_repo_mgr create fs --repository <リポジトリ名> --location <リポジトリ格納パス>
```

リポジトリ格納パスは絶対パスで指定することもできますが、Elasticearch で設定している path.repo を基準に相対パスで指定してください。上記コマンドが実行されると格納パスの配下にリポジトリ名のフォルダーが作成されます。

< cURL コマンドによる API での設定 >

Curator を利用しない場合は SSH クライアントからの cURL コマンドや Kibana の Dev Tools 等で直接 Elasticsearch の API を利用します。

i) 下記のような API を実行します。

```
[<username>@<hostname> ~]$ curl -X PUT http://localhost:9200/_snapshot/<リポジトリ名> -d '
{
  "type": "fs",
  "settings": {
    "location": "<リポジトリ格納パス>"
  }
}'
```

備考

リポジトリを設定した場合、リポジトリのパスもしくはその Disk にアクセスできない場合、Elasticsearch が起動できなくなることがあるため注意が必要です。ただし、Elasticsearch の稼働中にリポジトリにアクセスできなくなっても、Elasticsearch がダウンすることはありません。

3.9.4. 月次スナップショット整理

優先度：高

検討タイミング：運用後

推奨

月初に一度だけスナップショットの作成を実施します。

方法

< Curator による設定 >

1) テキストエディタで例えば以下の内容の Action ファイルを作成します。ファイルの拡張子は .yml とします。

```
actions:
  1:
    action: snapshot
    description: "Create snapshot of previous month"
    options:
      repository: "backup"
      name: '<monthly-{now/M-1M{YYYY.MM}}>'
      continue_if_exception: False
      wait_for_completion: True
    filters:
      - filtertype: pattern
        kind: prefix
        value: "default-"
      - filtertype: period
        period_type: relative
        source: name
        timestring: "%Y.%m"
        range_from: -1
        range_to: -1
        unit: months
  2:
    action: delete_snapshots
    description: "Delete daily snapshot of previous month"
    options:
      repository: "backup"
```

```
    ignore_empty_list: True
    continue_if_exception: False
  filters:
    - filtertype: pattern
      kind: prefix
      value: "daily-"
    - filtertype: period
      period_type: relative
      source: name
      range_from: -1
      range_to: -1
      timestring: "%Y.%m"
      unit: months
```

2) ファイルを作成したら以下のコマンドを実行します。

```
Curator <作成したファイルのパス>
```

以上で月次スナップショット整理は終了です。

< cURL コマンドによる API での設定 >

Curator を利用しない場合は SSH クライアントからの cURL コマンドや Kibana の Dev Tools 等で直接 Elasticsearch の API を利用します。

i) 以下の API で先月分の月次スナップショットを作成します。例えば 2020 年 4 月の月初に実施する場合、先月分の Index 名を指定して、以下のように記述します。

```
[<username>@<hostname> ~]$ curl -X PUT
http://localhost:9200/_snapshot/backup/%3Cmonthly-%7Bnow%2FM-
1M%7BYYYY.MM%7D%7D%3E?wait_for_completion=true -d '
{
  "indices": "default-2020.03*",
  "ignore_unavailable": true,
  "include_global_state": false
}'
```

ii) 月次スナップショットを作成した後は不要な日次スナップショットを削除します。まず、削除対象のスナップショットを調べるために以下の API を利用します。赤字の YYYY.MM 部分は先月分のスナップショットに合致するように指定します。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_snapshot/backup/daily-YYYY.MM.*?filter_path=snapshots.snapshot
```

例えば、2020 年 4 月の月初に実施する場合は以下のように指定すれば 2020 年 3 月の日次スナップショットの一覧が得られます。

```
GET _snapshot/backup/daily-2020.03.*?filter_path=snapshots.snapshot
```

以下は出力例です。

```
{
 "snapshots" : [
  {
   "snapshot" : "daily-2020.03.01"
  },
  {
   "snapshot" : "daily-2020.03.02"
  },
  ... (略) ...
 ]
}
```

iii) 一覧表示された各スナップショットについて、以下の API で削除します。複数のスナップショットを一括で削除はできないため、赤字の YYYY.MM.dd を日付指定して繰り返し実行する必要があります。

```
[<username>@<hostname> ~]$ curl -X DELETE http://localhost:9200/_snapshot/backup/daily-YYYY.MM.dd
```

例えば、2020 年 3 月 18 日に取得した日次スナップショットを削除する場合は以下のように指定します。

```
DELETE _snapshot/backup/daily-2020.03.18
```

以上で月次スナップショット整理は終了です。

備考

v7.4 より Snapshot の新機能として Snapshot lifecycle management (SLM) が導入されています。Kibana コンソール画面の Management タブから、snapshot の自動取得および削除などを設定できるようになりました。

参考

[47] Curator Reference [5.8] » Configuration » Action File（Elastic 社公式ガイド）
[48] Elasticsearch Reference [7.4] » Managing the index lifecycle » Getting started with snapshot lifecycle management（Elastic 社公式ガイド）

3.9.5. 日次スナップショット取得

優先度：高
検討タイミング：運用後
推奨

日次スナップショットの作成を可能な限り毎日実施します。
方法

< Curator による設定 >
1) テキストエディタで例えば以下の内容の Action ファイルを作成します。ファイルの拡張子は .yml とします。

```
actions:
  1:
    action: snapshot
    description: "Create daily snapshot of this month"
    options:
      repository: "backup"
      name: 'daily-%Y.%m.%d'
      continue_if_exception: False
      wait_for_completion: True
    filters:
      - filtertype: pattern
        kind: prefix
        value: "default-"
      - filtertype: period
        period_type: relative
        source: name
        timestring: "%Y.%m"
        range_from: 0
        range_to: 0
```

```
      unit: months
```

2) ファイルを作成したら以下のコマンドを実行します。

```
Curator <作成したファイルのパス>
```

3) 以上で日次スナップショットの取得は終了です。

< cURL コマンドによる API での設定 >

Curator を利用しない場合は SSH クライアントからの cURL コマンドや Kibana の Dev Tools 等で直接 Elasticsearch の API を利用します。

i) 例えば、2020 年 4 月に実施する場合は以下のように指定します。

```
[<username>@<hostname> ~]$ curl -X PUT
http://localhost:9200/_snapshot/backup/%3Cdaily-%7Bnow%2Fd%7BYYYY.MM.dd%7D%7D%3E?wait
_for_completion=true -d '
{
  "indices": "default-2020.04*",
  "ignore_unavailable": true,
  "include_global_state": false
}'
```

以上で日次スナップショットの取得は終了です。

3.9.6. スナップショットのリストア

優先度：低

検討タイミング：運用後

推奨

リストアは必ず Curator または API 経由で実施します。

方法

< Curator による設定 >

1) 以下のコマンドでスナップショットの一覧を取得し、リストアしたいスナップショットの名前を確認します。

```
[<username>@<hostname> ~]$ Curator_cli show_snapshots --repository <リポジトリ名>
```

2) スナップショットに含まれる Index 等を確認したい場合は Elasticsearch の以下の API を直接利用してください。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_snapshot/<リポジトリ名>/<スナップショット名>
```

上記 API を呼び出すと JSON 形式でスナップショットの情報が返却されます。 返却された JSON の indices フィールドにスナップショットに含まれる Index 名の一覧が記載されています。

3) リストアするスナップショットを決定したら以下の Action ファイルを作成します。ファイルの拡張子は .yml とします。

```
actions:
  1:
    action: close
    description: "Close indices before restoring snapshot"
    options:
      continue_if_exception: True
      ignore_empty_list: True
    filters:
      - filtertype: pattern
        kind: prefix
        value: "Index 名のプレフィクス"
  2:
    action: restore
    description: "Restore snapshot"
    options:
      repository: "リポジトリ名"
      name: "スナップショット名"
      wait_for_completion: True
    filters:
      - filtertype: state
        state: SUCCESS
  3:
```

```
  action: open
  description: "Open indices after restoring snapshot"
  filters:
    - filtertype: pattern
      kind: prefix
      value: "Index 名のプレフィクス"
```

3) ファイルを作成したら以下のコマンドを実行します。

```
Curator <作成したファイルのパス>
```

以上でスナップショットのリストアは終了です。

< cURL コマンドによる API での設定 >

Curator を利用しない場合は SSH クライアントからの cURL コマンドや Kibana の Dev Tools 等で直接 Elasticsearch の API を利用します。

i) 以下の API を利用してスナップショットの一覧を取得して、リストアするスナップショットの名前を確認します。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_snapshot/<リポジトリ名>/_all?filter_path=snapshots.snapshot
```

レスポンス例を以下に示します。snapshot フィールドの値がスナップショットの名前です。

```
{
 "snapshots" : [
   {
    "snapshot" : "daily-2020.01.30"
   },
   {
    "snapshot" : "daily-2020.01.31"
   },
   {
    "snapshot" : "daily-2020.02.01"
   },
   {
    "snapshot" : "monthly-2020.01"
   },
```

```
  {
    "snapshot" : "daily-2020.02.02"
  }
 ]
}
```

ii) スナップショットに含まれる Index 等を確認したい場合は Elasticsearch の以下の API を利用してください。

```
[<username>@<hostname> ~]$ curl -X GET http://localhost:9200/_snapshot/<リポジトリ名>/<スナップショット名>
```

上記 API を呼び出すと JSON 形式でスナップショットの情報が返却されます。 返却された JSON の indices フィールドにスナップショットに含まれる Index 名の一覧が記載されています。

iii) 次に以下の API で Index を close 状態にします。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/<Index_name>/_close
```

Index 名は default-* のようにワイルドカードで複数の Index を指定することができます。API が成功した場合は以下のような JSON が返却されます。

```
{
 "acknowledged" : true
}
```

iv) Index が close 状態となっているかを以下の API で確認してください。

```
[<username>@<hostname> ~]$ curl -X GET
http://localhost:9200/_cat/indices?v&h=health,status,index
```

レスポンス例を以下に示します。中央のカラムがオープン状態かクローズ状態かを示しています。

```
health status index
       close  default-2019.01
green  open   default-2019.02
green  open   .kibana_1
```

v) Index を close 状態にしたことを確認したら、以下の API でスナップショットをリストアします。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/_snapshot/<リポジトリ名>/<スナップショット名>/_restore
```

リストア操作が受け付けられると以下のような JSON が返却されます。

```
{
  "accepted" : true
}
```

vi) リストアの進捗は以下の API で確認できます。

```
[<username>@<hostname> ~]$ curl -X GET
http://localhost:9200/_cat/recovery?v&h=index,shard,time,stage,snapshot,bytes_percent
```

レスポンス例を示します。Index「default-2020.03」をスナップショット「daily-2020.03.30」からのリストアが完了した後の状態で実行した例です。

```
index          shard time  stage snapshot         bytes_percent
…(略)…
default-2020.03 0     74ms  done  daily-2020.03.30 100.0%
…(略)…
```

vii) リストアが完了すると Index は自動的に open 状態となります。スナップショットに含まれていないリストア対象外の Index の状態は変更されないので、close 状態のもので open 状態にする必要がある Index については以下の API で open してください。

```
[<username>@<hostname> ~]$ curl -X POST http://localhost:9200/<Index_name>/_open
```

以上でスナップショットのリストアは終了です。

4. Elastic Stack – Kibana の設定

Kibana は Elastic 社が開発しているデータ可視化のためのツールです。Elasticsearch に入っているデータを様々な形式で描画することができます。

- 本章のチェックリスト
  検討項目は下記の通りです。

§	検討項目	優先度	検討タイミング	推奨	分類
4.1.1	Kibana の OS 選定	高	運用前	CentOS または Red Hat Enterprise Linux (RHEL)	設定
4.2.1	Elasticsearch ﾕｰｻﾞｰの最小権限	低	運用後	※詳細は本文を参照してください。	設定
4.2.2	Kibana アクセスの暗号化	低	運用後	HTTP レイヤーを TLS 化します。	設定
4.3.1	X-Pack.Monitoring の有効化	高	運用後	X-Pack.Monitoring プラグインを有効化します。	監視
4.3.2	Kibana ログの有効化	高	運用後	Kibana ログを収集します。	監視
4.4.1	Kibana ログのローテーション	高	運用後	Kibana ログを自動的にローテーション（世代管理）させます。	運用
4.4.2	space	低	運用後	space を利用して保存済みオブジェクトを管理します。	運用

4.1. Kibana の OS 設定

4.1.1. Kibana の OS 選定

Elasticsearch と同様にインストール先の OS として、Linux と Windows があります。

優先度：高
検討タイミング：運用前
推奨

Linux（CentOS、RedHat Enterprise Linux）が推奨されます。

理由

Linux と Windows を動作環境として比較した場合に、評価軸（実績、機能、性能、運用）に対するポイントを下記の表に示します。

評価軸	説明
実績	- Elastic 社としては Linux への導入実績は豊富ですが、Windows への導入実績が少ないようです。フォーラム等でのセットアップや運用面での Q&A においても Linux ベースが多く、Windows 固有の問題については相対的に情報が少ないのが実情です。 - UiPath 社としては Linux と Windows の両方で導入実績があります。
機能	機能面での差はありません。お客様の環境、運用管理コスト、ポリシーなどを踏まえてご選択ください。
性能	お客様の利用環境に依りますが、大きな差はありません。
運用	- 一般に、Linux 環境の方が Windows 環境に比べて動作が軽量なため HW 要件を節約できる傾向があります。 - Kibana ログ の Log Rotation 実装を検討される際は、Linux の方が容易です。elasticsearch.log は Log Rotation されるものの、kibana.log をファイルに出力する設定をしている場合は、既定では Log Rotation されずに

	蓄積され続けます。その場合、Linux であれば LogRotate 機能を使用することで対応できますが、Windows にそのような機能は無いため作り込みが必要になります。 - Windows Server に Kibana を導入する場合、弊社の『Orchestrator 導入ステップバイステップガイド』でも記載がある通り、Non-Sucking Service Manager (NSSM) を使用しなければ Windows サービスとして登録できません。

参考

[49] Orchestrator 導入ステップバイステップガイド [2020.4 対応版]（UiPath ナレッジベース）

4.2. Kibana インスタンスの設定

4.2.1. Elasticsearch ユーザーの最小権限

本設定は、Elasticsearch のセキュリティ機能を有効にした場合にのみ検討します。

優先度：低
検討タイミング：運用後
推奨

Elasticsearch に対するログの読み取りと書き込みに必要な最小権限である下記権限を付与したユーザーを作成し、Orchestrator の Web.config 内にユーザー情報を記載します。

- read
- create
- create_index

Cluster Privileges は「Monitor」権限が必要です。

理由

デフォルトで作成される管理者ユーザーに与えられた既定の権限は通常の Orchestrator 運用に対して過剰です。セキュリティの統制の観点から、適切な権限に絞ることが推奨されます。既定のままでも機能面では差し支えありません。

4.2.2. Kibana アクセスの暗号化

Elastic Stack 6.8 および 7.1 より一部のセキュリティ機能が Basic ライセンスにて無償提供されるようになりました。新たに無料化された機能に、SSL を使用したネットワークトラフィックの暗号化、ユーザーの作成と管理、インデックスとクラスターレベルのアクセスを保護するロール定義、Kibana の安全な制御などがあります。

優先度：低
検討タイミング：運用後
推奨

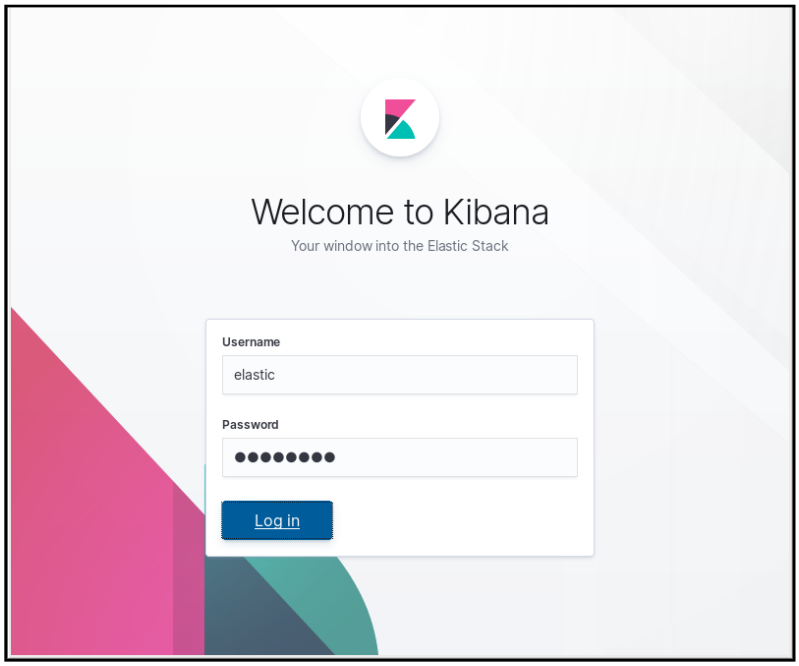
Kibana の Web インターフェースへのインターネットアクセスをセキュアに制限したい場合に、HTTP レイヤーを TLS 化します。

理由

HTTP レイヤーでの TLS 有効化は必須ではありませんが、データをエンドツーエンドで保護するためには強く推奨されます。特に、ユーザー名／パスワード情報を盗難から保護し、クラスターへのウイルス感染を防止する上で重要です。

方法

方法については Elastic 社の公式ブログ[50,51]に詳細の記載がありますので、そちらをご参照ください。



参考

[50] 無料の暗号化とユーザー認証で、Elasticsearch クラスターを安全に保つ（Elastic 社公式ブログ）
[51] セキュリティ機能のはじめ方（Elastic 社公式ブログ）

4.3. Kibana の監視

4.3.1. X-Pack.Monitoring の有効化

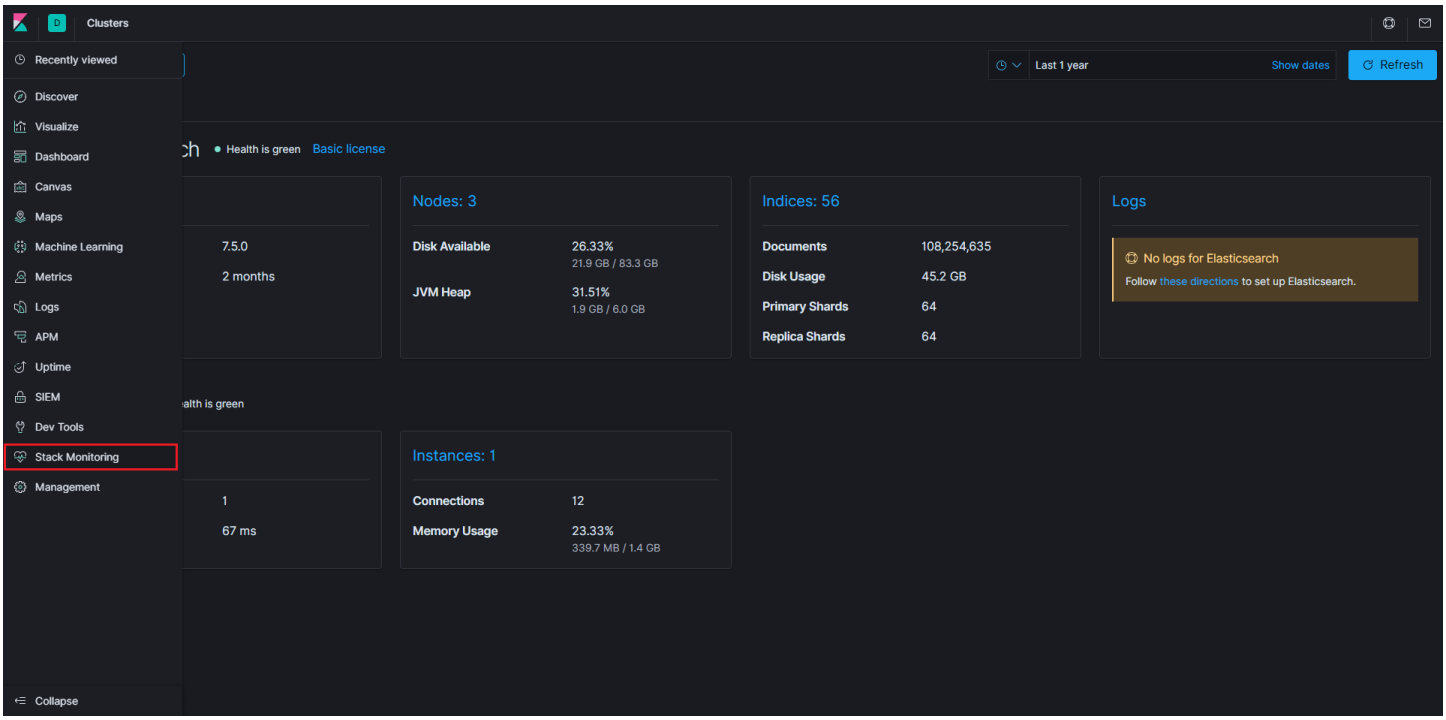
X-Pack.Monitoring は Kibana コンソールの GUI から Elastic Stack の監視を行うことができるプラグインです。X-Pack の機能の内、唯一無償で利用することができる機能です。ただし、無償利用は 1Cluster までに限るのでご注意ください。また、Basic License で利用可能ですが、Basic License の場合は 1 週間しかデータが残りません。

優先度：高
検討タイミング：運用後
推奨

X-Pack.Monitoring を有効にして、Elasticsearch や Kibana の状態を Kibana コンソールから確認できるようにする。既定では、Monitoring プラグインは有効になっていますが、データ収集は無効になっています。

方法

elasticsearch.yml と kibana.yml の両方で「xpack.monitoring.enabled: true」を末尾に追記します。Monitoring ページは Kibana コンソール > サイドバー > Elastic Monitoring からアクセスできます（下図を参照）。



備考

< Monitoring の監視項目 >

ツール名	監視項目
Elasticsearch	• Cluster Overview：Cluster 単位のステータス。 • Nodes：ノード単位のパフォーマンス、Shard 情報。 • Indices：Index 単位のパフォーマンス、Shard 情報。 • Logs：エラーログ数や警告ログ数などの情報。7 系より追加された項目です。
Kibana	• Overview：アクセス状況 • Instances：ノード単位のパフォーマンス

参考

[52] Kibana Guide [7.8] » Set up Kibana » Configure Kibana » Monitoring settings in Kibana（Elastic 社公式ガイド）

4.3.2. Kibana ログの有効化

優先度：高

検討タイミング：運用後

推奨

Kibana ログを収集します。

理由

Kibana サーバーのトラブルシュートに Kibana ログを利用します。

方法

1) SSH クライアントで Kibana サーバーにログインします。
2) root 権限で Kibana 設定ファイル (kibana.yml) を編集します。

```
[root@<hostname> ~]# vi /etc/kibana/kibana.yml
```

3) Kibana ログが保存されるディレクトリを作成します。mkdir コマンドのオプション「-p」は、引数に指定した各ディレクトリで存在しないディレクトリも含めて作成するという意味です。

```
[root@<hostname> ~]# mkdir -p /var/log/kibana
```

4) Elasticsearch/Kibana のアドレスとログ有効化を設定します。

```
…(略)…
# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: "<host-ip>"
…(略)…
# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["http://<host-ip>:9200"]
…(略)…
# Enables you specify a file where Kibana stores log output.
logging.dest: /var/log/kibana/kibana.log
…(略)…
```

5) Kibana ログに対する読み取りと書き込みの権限を付与します。

```
[root@<hostname> ~]# chmod 664 /var/log/kibana
```

「664」は「ls -l」コマンドで云うところの「-rw- rw- r--」となります。これは、所有者と所有グループに読み取り(r)と書き込み(w)の権限を与え、その他の一般ユーザーには読み取り(r)の権限のみを与えることを意味します。実行(x)の権限は誰にも与えられません。「実行」は、ディレクトリの場合は cd コマンドでそのディレクトリに入ることも含みます。

4.4. Kibana の運用

4.4.1. Kibana ログのローテーション

優先度：高

検討タイミング：運用後

推奨

Kibana ログを自動的にローテーション（世代管理）させるように設定します。

理由

Kibana ログは既定ではローテーションされず、蓄積される一方です。長期的に運用する場合、Kibana サーバーのディスク容量を逼迫させる恐れがあります。

方法

ここでは Kibana の運用環境として Linux を想定しています。Windows の場合は別途スクリプト等を作り込む必要がありますが、本ドキュメントの範疇を超えるため触れていません。

1) Kibana.yml から Process ID ファイル (*.pid) の出力を設定する。

```
# Specifies the path where Kibana creates the process ID file.
pid.file: /var/run/kibana.pid
```

2) logrotate サービスが適切にインストールされているか確認します。基本的には logrotate サービスは標準で入っています。

```
[<username>@<hostname> ~]$ logrotate --version
```

出力例は下記の通りです。

```
logrotate 3.8.6
```

3) logrotate サービスの管理ファイルに logrotate.d ディレクトリが指定されていることを確認します。/etc/logrotate.d/* は個々のログファイルの設定ファイルです。

```
[<username>@<hostname> ~]$ cat /etc/logrotate.conf
```

内容は下記の通りです。

```
…(略)…
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
…(略)…
```

4) 新規に logrotate サービスの管理ファイルを Kibana 用に作成します。

```
[<username>@<hostname> ~]$ cat << EOF > /etc/logrotate.d/kibana
/var/log/kibana/*.log {
  missingok
  daily
  size 128M
  create 0644 kibana kibana
  rotate 7
  notifempty
  sharedscripts
  notifempty
  compress
  postrotate
    /bin/kill -HUP $(cat /var/run/kibana/kibana.pid 2>/dev/null) 2>/dev/null
  endscript
}
EOF
```

<オプションについて>

オプション	説明
daily / weekly / monthly	ログをローテーションするサイクルです。 • daily -> 04:02 • weekly -> 日曜日 04:22 • monthly -> 毎月 1 日 04:42
size <filesize>	ローテーションが走るログファイルサイズの閾値
rotate <世代数>	保持する世代数
dateext	日付型をファイル末尾に付与します。
dateformat	末尾に付与する日付フォーマットを変更します。既定では「-%Y%m%d」です。
compress / nocompress	ログファイルを gzip で圧縮、または圧縮しません。
delaycompress	ログの圧縮作業を次回のローテーション時まで遅らせます。Compress オプションと一緒に指定します。
copytruncate	ログファイルをコピーし、元のログファイルの内容を削除します。
create <permission> <username> <group> / nocreate	ローテーション後に空のログファイルを新規作成するか否かを指定します。ファイルのパーミッション、ユーザー名、グループ名を指定できます。
ifempty / notifempty	ログファイルがない場合にローテーションするか否かを指定します。
missingok / nomissingok	ログファイルが存在しない場合にエラーをスローするか否かを指定します。
olddir <directory_path> / noolddir	古いログファイルを別のディレクトリに移行するか否かを指定します。
prerotate / postrotate <script> endscript	prerotate / postrotate と endscript の間に記述されたコマンドをログローテーション前／後に実行します。
sharedscripts / nosharedscripts	ログファイルをワイルドカードで指定して、1 つのサービスにつき複数個のログがある場合、スクリプトを各々実行する (noshared) か 1 回だけ実行する (shared) かを指定します。
su <username> <group>	CentOS 7 以降、指定したユーザーとしてローテーションを実行します。

5) 次のコマンドでファイルの構文 (syntax) を確認します。

```
[<username>@<hostname> ~]$ logrotate -vd /etc/logrotate.d/kibana
```

6) 5) でエラーが無ければ、下記のコマンドでログローテーションをマニュアル実行してみましょう。

```
[<username>@<hostname> ~]$ logrotate -vf /etc/logrotate.d/kibana
```

備考

< エラー例 >

- ユーザーに権限が無い場合に下記のようなエラーが出ることがあります。su オプションを付けるなどして適切なユーザーに変更しましょう。

> error: skipping ~ because parent directory has insecure permissions (It's world writable or writable by group which is not "user") Set "su" directive in config file to tell logrotate which user/group should be used for rotation.

参考

[53] http://www.linuxfromscratch.org/blfs/view/svn/general/logrotate.html

4.4.2. space

Kibana v6.5 より追加された space 機能を利用することで、保存済みオブジェクト（ダッシュボードや可視化など）をカテゴリ別（部署やプロジェクト、ユーザーなど）に整理することができます。各 space は独立しており、1 つの space に入れたオブジェクトは、他の space から参照されることはありません。

優先度：低
検討タイミング：運用後
推奨

space を利用して保存済みオブジェクトを管理します。
理由

保存済みオブジェクトに対して細かいアクセス制御を施すことができます。
方法

参考の[51]を参照ください。
参考

[54] Kibana の"space"で業務コンテンツ整理する（Elastic 社公式ブログ）

5. HAA/Redis の設定

Orchestrator (IIS) サーバーで Active/Active な冗長構成を組む場合、Redis は必須コンポーネントです。Orchestrator ノード間でのキャッシュと共有に使用されるインメモリデータベースとして機能します。ノード間の同期はほぼ即時反映されます。ここで想定される Redis は Redis {OSS, Enterprise}および **UiPath High Availability Add-on (HAA)**です。UiPath の公式サポートの対象となるのは Redis Enterprise の UiPath による OEM 製品である HAA のみです。**特に断わらない限り、紹介される推奨設定は HAA およびこれら Redis に共通する**ものです。以下の本文では HAA を含めて Redis と呼称します。HAA 特有の特記事項などがある場合に限り、その旨を明記します。

- 本章のチェックリスト
  検討項目は下記の通りです。

§	検討項目	優先度	検討タイミング	推奨	分類
5.1.1	Redis の OS 選定	高	運用前	CentOS または Red Hat Enterprise Linux (RHEL)	設定
5.1.2	Overcommit Memory	高	運用後	vm.overcommit_memory=1	設定
5.1.3	Socket の設定	高	運用後	net.core.somaxconn=511 net.ipv4.tcp_max_syn_backlog=511 net.core.netdev_max_backlog=65536	設定
5.2.1	abortConnect 値	高	運用後	abortConnect=false	設定
5.2.2	responseTimeout 値	高	運用後	responseTimeout=5000	設定
5.3.1	Redis サービス	低	運用後	※詳細は本文を参照してください。	監視
5.3.2	レイテンシ	低	運用後	※詳細は本文を参照してください。	監視

5.1. Redis の OS 設定

Redis が動作する OS の推奨設定について説明します。

5.1.1. Redis の OS 選定

優先度：高

検討タイミング：運用前

推奨

本ガイドで推奨している OS は CentOS または Red Hat Enterprise Lunix (RHEL)です。

備考

Windows 版 Redis は、Redis や RedisLabs ではなく、MSOpenTech によって作成されています。2016 年 7 月以降は更新されておらず、サポートされていません。

5.1.2. Overcommit Memory

Linux は、メモリが不足してシステムが停止する恐れがある際、メモリリソースを多く消費しているプロセスを強制終了させます。これは OOM Killer という機構です。重要なプロセスであっても等しく強制終了の対象となります。Overcommit Memory はメモリが使用可能かどうかをチェックするカーネルパラメータです。

優先度：高
検討タイミング：運用後
推奨

vm.overcommit_memory 値を「1」に設定します。

理由

OOM Killer による Redis の突発的なダウンを抑止します。

方法

1) SSH クライアントから Redis サーバーに root 権限でログインし、/etc/sysctl.conf を編集します。

```
[root@<hostname> ~]# vi /etc/sysctl.conf
```

2) ファイルの末尾に下記の内容を追記または編集します。

```
vm.overcommit_memory = 1
```

3) Redis サーバーを再起動します。

```
[root@<hostname> ~]# shutdown -r now
```

備考

< vm.overcommit_memory 値について >

vm.overcommit_memory 値	説明
0（既定）	メモリ要求があったときに空き容量が無い場合、実行中のプロセスを強制終了してメモリを強引に確保します。
1	メモリを使い切るまでは十分なメモリがあるように振る舞います。
2	メモリ要求があったときに空き容量がない場合、メモリ確保ができないエラーを発生させます。

参考

[55] Redis Administration > Redis setup hints（Redis Labs 公式ページ）

5.1.3. Socket の設定

優先度：高

検討タイミング：運用後

推奨

Redis が受け取る SYN パケットを格納するキューサイズを可能な限り余裕をもって設定します。

パラメータ	説明	既定値	推奨値
net.core.somaxconn	TCP Socket が受け付けた接続要求を格納する Queue の最大長です。Redis で設定可能な上限値は 511 です。	128	511（上限値）
net.ipv4.tcp_max_syn_backlog	Socket 当たりの SYN を受け付けて ACK を受け取っていない状態のコネクションの保持可能数です。ただし backlog で somaxconn より大きな数字を定義しても、somaxconn の値が優先されます。	1024	511
net.core.netdev_max_backlog	パケット受信時に Queue に繋ぐことができるパケットの最大数です。	1000	65536

理由

net.ipv4.tcp_max_syn_backlog の設定値の 75%を超過すると SYN パケットが Drop してしまいます。

方法

1) SSH クライアントから Redis サーバーにログインして、/etc/sysctl.conf を編集します。

```
[<username>@<hostname> ~]$ vi /etc/sysctl.conf
```

2) ファイルの末尾に下記の内容を追記または編集します。

```
net.core.somaxconn = 511
net.ipv4.tcp_max_syn_backlog = 511
net.core.netdev_max_backlog = 65536
```

5.2. Redis インスタンスの設定

5.2.1. abortConnect 値

優先度：高

検討タイミング：運用後

推奨

abortConnect=false に設定し、Orchestrator と Redis 間の接続が再試行されるようにします。

理由

Orchestrator と Redis 間ではいくつもの接続が張られていますが、ネットワークの接続不良などの問題で接続が切れると、既定では「abortConnect=true」で再接続されません。再接続されない不要な接続はエラーとしてイベントログに出力され続けます。

※ Azure 環境は既定で abortConnect=false となっており、再接続は有効です。

方法

1) Orchestrator サーバー（IIS サーバー）にログインします。
2) Web.config を開きます。
3) 自動再接続の有効化（abortConnect=false）するように
4) 設定します。

```
<add key="LoadBalancer.Redis.ConnectionString" value="<redis-server-endpoint>:6379,abortConnect=false" />
```

※ HAA の場合、既定の Port 番号が「6379」ではなく「10000」です。

5) IIS サイトを再起動します。

5.2.2. responseTimeout 値

優先度：高

検討タイミング：運用後

推奨

Redis 接続の Timeout 値（ミリ秒）を 5000 ミリ秒（5 秒）に延長します。

理由

Orchestrator から Redis 接続のタイムアウト時間は既定で 1 秒のため、ネットワーク遅延などにより RedisConnectionException の例外が発生するケースがあります。

方法

1) Orchestrator サーバー（IIS サーバー）にログインします。
2) Web.config を開きます。
3) 自動再接続の有効化（abortConnect=false）と responseTimeout 値を 5 秒（responseTimeout=5000）に設定します。

```
<add key="LoadBalancer.Redis.ConnectionString" value="<redis-server-endpoint>:6379,abortConnect=false,responseTimeout=5000" />
```

※ HAA の場合、既定の Port 番号が「6379」ではなく「10000」です。

4) IIS サイトを再起動します。

備考

本設定に関連して、例えば下記のようなエラーが発生する場合があります。

```
Error updating job state:System.TimeoutException: Timeout performing HGETALL AbpCommonHub.Clients, inst: 5, mgr: checkForStaleConnections, err: never, queue: 13, qu: 0, qs: 13, qc: 0, wr: 0, wq: 0, in: 0, ar: 0, clientName: {0}, IOCP: IOCP: (Busy=0, Free=1000, Min=12, Max=1000), WORKER: (Busy=12, Free=32755, Min=12, Max=32767), Local-CPU: unavailable
```

```
Error updating job state:StackExchange.Redis.RedisConnectionException: SocketFailure on EVAL
```

5.3. Redis の監視

5.3.1. Redis サービス

優先度：低

検討タイミング：運用後

推奨

Redis サービスが開始されているかを確認します。

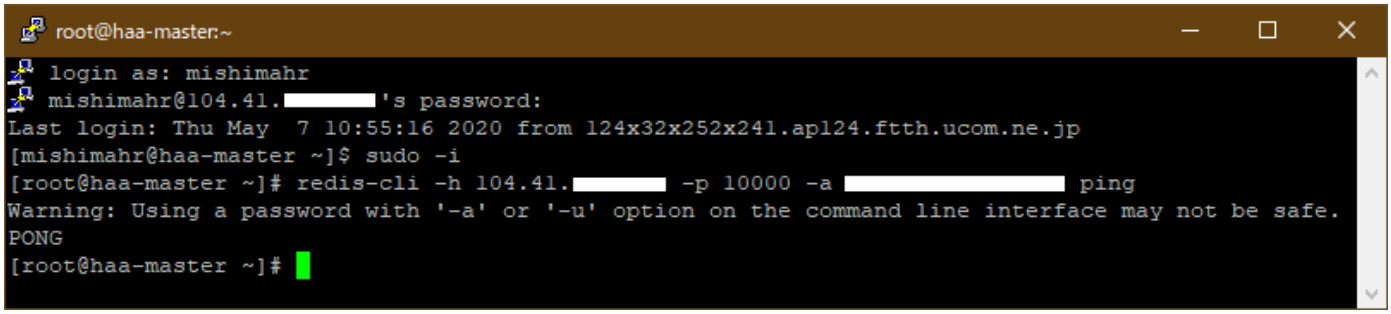
方法

1) Redis の Master ノードに SSH 接続し、下記のコマンドを root 権限で実行します。

```
[root@<Master ノードのホスト名> ~]$ redis-cli -h <HAA/Redis サーバー名> -p <ポート番号> -a <パスワード> ping
```

パラメータ	説明
-h	HAA/Redis サーバー名または IP アドレス。
-p	HAA/Redis サーバーのポート番号。省略時には Redis サーバーの既定ポート番号 6379 が使用されますが、HAA サーバーの既定ポート番号は 10000 のため、明示的に 10000 を指定します。
-a	HAA/Redis サーバーのパスワード。HAA に対してはインストール時に設定したパスワードを指定します。

2) 正常に動作している場合、「PONG」が返されます。



5.3.2. レイテンシ

Redis 2.8.13 から処理の遅かったコマンドの分布や傾向を知るための latency monitoring framework が追加されています。既定では無効化されています。

優先度：低

検討タイミング：運用後

推奨

インスタンスの遅延度を別プロセスで分析したい場合に有効化します。正常系では無効にしておきます。

方法

1) Redis の Master ノードに SSH 接続します。例えば最大許容レイテンシが 100 ミリ秒の場合、100 ミリ秒以上に亘ってサーバーをブロックしているすべてのイベントを記録するには、下記のコマンドを実行します。

```
[<username>@<Master ノードのホスト名> ~]$ CONFIG SET latency-monitor-threshold 100
```

※ ここではレイテンシの閾値は 100 ミリ秒に設定しましたが、既定では「0（無効）」となっています。

備考

Redis の疎通確認向けのコマンド である PING をクライアントが実行すると、Redis サーバーは PONG を返すだけです。このコマンドに対する Redis の処理は極めて軽微なため、PING を実行してから PONG が返ってくるまでの時間をレイテンシとみなして計測することもできます。例えば下記のようなコマンドです。

```
[<username>@<Master ノードのホスト名> ~]$ redis-cli --latency -h <hostname> -p <port_number>
min: 0, max: 2, avg: 0.17 (249 samples)
```

上の例で言えば、249 回 PING を実行し、レイテンシの最短(min)=0ms、最長(max)=2ms、平均(avg)=0.17ms であったことが分かります。

参考

[56] Redis レイテンシ監視フレームワーク（Redis Labs 公式ページ）

6. Load Balancer の設定

本ガイドで対象とする Load Balancer は F5 BIG-IP です。

- 本章のチェックリスト
  検討項目は下記の通りです。

§	検討項目	優先度	検討タイミング	推奨	分類
6.1.1	connectionTimeout 値	高	運用後	connectionTimeout="00:00:20"	設定

6.1. IIS サーバーに関わる設定

6.1.1. connectionTimeout 値

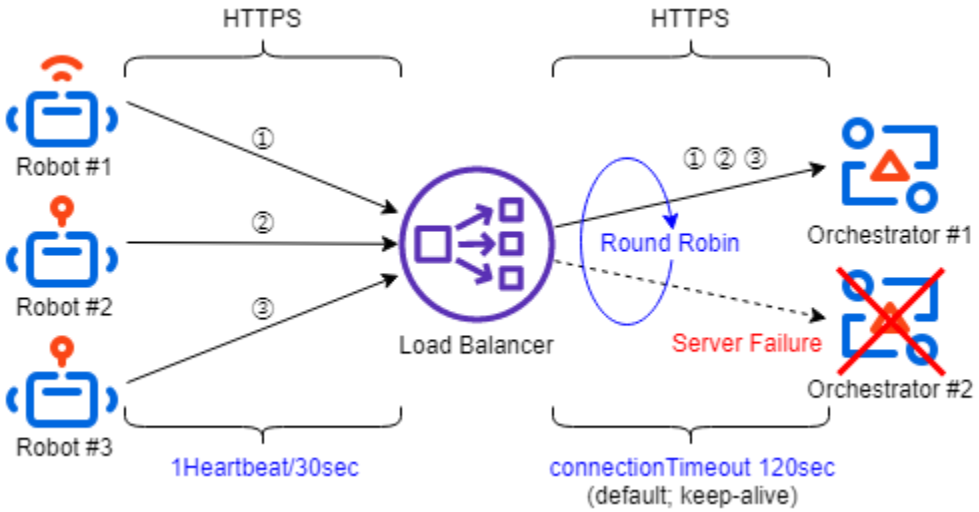
Orchestrator に関わる [Timeout] プロパティは複数ありますが、ここでは Orchestrator の環境に応じて設定の変更が推奨される項目を紹介します。

優先度：高
検討タイミング：運用後
推奨

複数の Orchestrator ノードを Load Balancer で Round Robin にてバランスさせる構成の場合、クライアントと Orchestrator 間の [接続のタイムアウト(秒)] プロパティを 120 秒（既定）から 20 秒に変更します。

理由

Load Balancer 利用環境において、複数の Orchestrator へ張られるセッションのストライピングを平準化します。
　Robot から Orchestrator への Heartbeat は既定で 30 秒毎に送信されます。一方、HTTP keep-alive により既定値のタイムアウト 120 秒が発生する前に次の Heartbeat が送信されるため、Robot からの TCP セッションは張られ続けることになります。その結果、メンテナンス時などに一方の Orchestrator を停止したときには他方の Orchestrator に接続が偏ってしまいます。その後 Orchestrator #2 が復旧しても、TCP セッションが Orchestrator #1 に張りっぱなしとなるため、平準化されません。

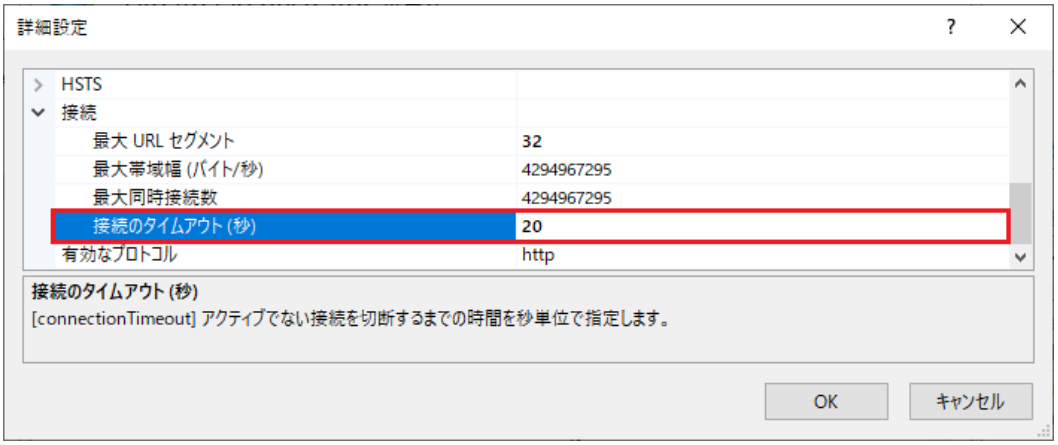


connectionTimeout 値を 20 秒に変更すると、次の Heartbeat が来る前にセッションが一旦終了し、再度張り直すため、この偏りを自動的に解消することができます。

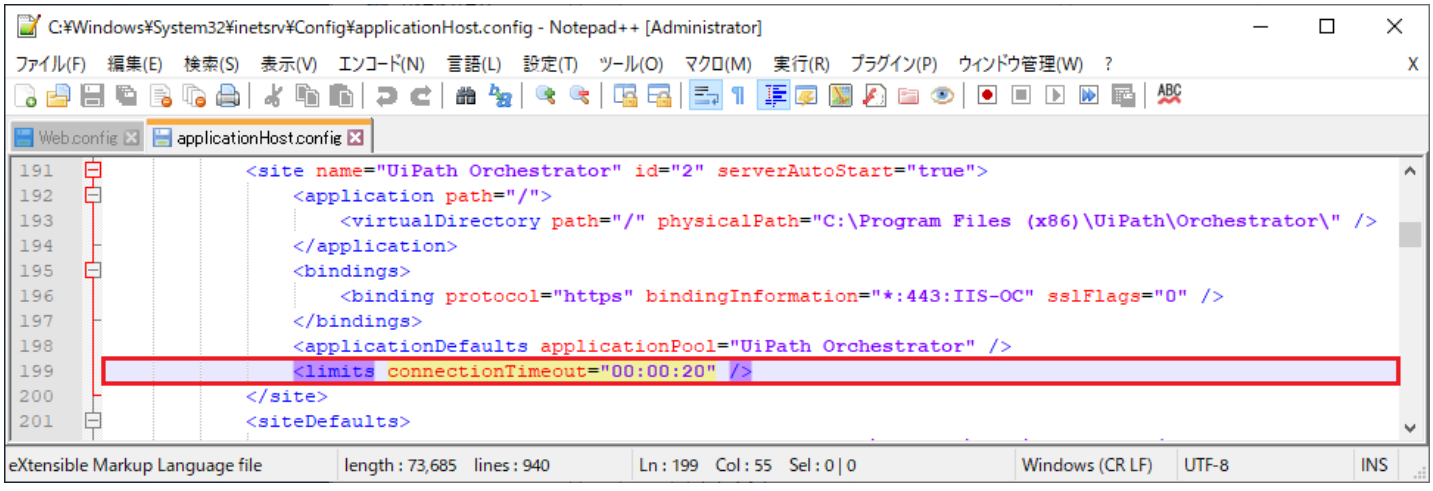
方法

以下のいずれかの方法で設定することができます。

<GUI から設定>
[IIS マネージャー] > [サイト] > [UiPath Orchestrator] > [詳細設定] > [接続] > [接続のタイムアウト (秒)] から、設定値を「120（既定）」から「20」に変更します。



<applicationHost.config ファイルから設定>
このファイルは %Windir%¥System32¥inetsrv¥Config¥applicationHost.config にあります。connectionTimeout 値を赤枠の箇所のように変更します。




```
<limits connectionTimeout="00:00:20" />
```

備考

ASP.NET で SQL Server に重いクエリコマンドを投げると、例えば「Timeout に達しました。操作が完了する前にタイムアウト期間が過ぎたか、またはサーバーが応答していません。」といったエラーメッセージがログに記録される場合があります。

7. 付録

♦ 本章のチェックリスト
検討項目は下記の通りです。

§	検討項目	優先度	検討タイミング	推奨	分類
7.1.1	DFS レプリケーション	高	運用前	ﾚﾎﾟｼﾞﾄﾘﾀｲﾌﾟが Legacy の場合、*.bin ﾌｧｲﾙを同期除外。	設定

7.1. Nuget Package ディレクトリ

NuGet Package (*.nupkg) ディレクトリには、Studio から Publish された NuGet Package と、インストール時にコピーされる Activity Package が配置されます。Orchestrator（IIS サーバー）が Single 構成の場合には NuGet Package はローカルディレクトリに配置しても問題ありません。

Orchestrator が冗長構成の場合の Nuget Package ディレクトリの構成として、ファイルサーバーを利用する方法と、ローカルディレクトリ間で DFS レプリケーションを設定する方法（§7.1.1 を参照）の 2 つが考えられます。基本的には、いずれか 1 つの方法を選択してください。

NuGet Package ディレクトリの構成は v2019 FT 以降、仕様が変更されています。そのため Orchestrator バージョンアップ時には互換性を維持するか、新しい構成（レポシトリタイプ）に移行するかを検討します。

レポシトリタイプ

< Legacy レポジトリタイプ >
この方法は初期の Orchestrator から採用されている方式です。旧バージョンから v2019 FT/LTS にバージョンアップした場合にも互換性維持のために同じ方式が引き継がれます。v2018.4 以前、または v2019 FT 以降において Legacy レポジトリタイプを指定している場合に使用されます。Orchestrator を v2018.4 以前からバージョンアップする際には、バージョンアップ後、Web.config で NuGet.Repository.Type 設定が Legacy となっていることを確認します。

- NuGet Package 本体は Web.config の NuGet.Packages.Path 設定 (既定値では相対パス: ~/NuGetPackages , 絶対パス: C:¥Program Files (x86)¥UiPath¥Orchestrator¥NuGetPackages) に応じて、テナント毎／Package 毎／バージョン毎に保存されます。
- NuGet Package のメタデータは テナント毎に <hostname>.cache.bin というファイル名で作成されます。このファイルは NuGet Package が新たにアップロードされたタイミングなどで再構成され、Orchestrator 管理画面上で Package 一覧を表示する処理や Robot に NuGet フィードを提供する処理などを高速化するために使用されます。DFS レプリケーション（§7.1.2 参照）の使用時には *.cache.bin を同期除外するように設定し、ファイル更新が無限プールに陥ることを防止します。
- Activity Package 本体は NuGet.Activities.Path 設定に応じて、ホスト毎またはテナント毎に保存されます。既定では Orchestrator インストーラーに含まれる Activity Package がコピーされます。
- NuGet Package ディレクトリを移行する際には、NuGet Package 本体 (*.nupkg) および Activity Pacage を既存ディレクトリから新規ディレクトリにコピーすることによって移行が完了します。<hostname>.cache.bin ファイルは自動的に再作成されるためコピーする必要はありません。

< Composite レポジトリタイプ >
v2019 FT/LTS を新規インストールした場合、または Orchestrator バージョンアップ後、Web.config を明示的に NuGet.Repository.Type 設定を Composite と変更した場合、NuGet Package ディレクトリは新しい構成となります。

- NuGet Package 本体は Web.config のストレージ.Location 設定に応じてテナント毎に保存されます。既定値は相対パス: ".¥ストレージ"、絶対パス: "%ProgramFiles(x86)%¥UiPath¥Orchestrator¥ストレージ"です。
- NuGet Package のメタデータは "UiPath" データベースのテーブル内に保存されます。
- Activity Package 本体は Web.config のストレージ.Location 設定に応じてホスト共通で保存されます。設定変更によってテナント毎に保存することもできます。
- NuGet Package ディレクトリを移行する際には、単純なファイルコピーでは移行できません。Orchestrator 管理画面の Package または Library 画面を使用して *.nupkg ファイルを個別にアップロードします。多数の Package を一括してアップロードするには、UiPath 公式サイト [58] で公開されている PowerShell スクリプトを使用します。

参考

[57] UiPath Orchestrator Guide > テナント設定を構成する > [ライブラリ (Libraries)] セクション（UiPath 公式ページ）
[58] NuGet パッケージアップロードスクリプト（UiPath 公式ページ）

ファイルサーバー

Orchestrator が冗長構成の場合の Nuget Package ディレクトリの構成方法の一つとして、すべての Orchestrator ノードがアクセス可能なファイルサーバーを UNC パスで指定します。ファイルサーバーにアクセス可能なユーザーを Orchestrator のアプリケーションプール ID として指定し、適切な権限（変更／読み取りと実行／フォルダーの内容の一覧／読み取り) を付与します。

既存ファイルサーバーも利用できるため実装し易いですが、ファイルサーバーそのものが冗長化されていない場合、単一障害点となります。またファイルサーバーに対するアクセス権限に注意します。

7.1.1. DFS レプリケーション

Orchestrator が冗長構成の場合の Nuget Package ディレクトリの構成方法の一つとして、各 Orchestrator ノードで各ローカルディレクトリを NuGet Package ディレクトリとして指定し、DFS レプリケーションを使用して各ノードのディレクトリを同期します。DFS を使用するには全ての Orchestrator ノードを同じ Active Directory Domain に参加させる必要があります。

優先度：高
検討タイミング：運用前
推奨

レポジトリタイプが Legacy の場合には *.bin ファイルを同期除外します。

※ レポジトリタイプが Composite の場合には cache.bin がデータベース内に格納されるため、上記の考慮は不要です。

理由

Orchestrator が 2 台 (oc01, oc02 とします) があった場合、それぞれのキャッシュファイル (<hostname>.cache.bin) が同期されると下記の通り更新処理がループする可能性があるためです。

i) oc01 に NuGet Package ディレクトリに Package がアップロードされる。
ii) oc01 が NuGet Package ディレクトリの変更を検知して oc01.cache.bin を再作成する。
iii) DFS レプリケーションにより oc01.cache.bin が oc02 の NuGet Package ディレクトリにコピーされる。

iv)　oc02 が NuGet Package ディレクトリの変更を検知して oc02.cache.bin を再作成する。
v)　DFS レプリケーションにより oc02.cache.bin が oc01 の NuGet Package ディレクトリにコピーされる。
vi)　ii) に戻る。

*.bin を同期除外することにより iii)の処理が走らないため上記の無限ループを防止することができます。

参考

[59]　DFS Replication overview（Microsoft 社公式ページ）

以上