Amazon OpenSearch Service 機能解説 – 検索編

AWS Black Belt Online Seminar

Takayuki Enomoto

Solutions Architect, Analytics

2023/01

AWS Black Belt Online Seminarとは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、AWSの各サービスやソリューションについてテーマごとに動画を公開します
- 動画を一時停止・スキップすることで、興味がある分野・項目だけの聴講も可能、スキマ時間の学習にもお役立ていただけます
- 以下のURLより、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/

内容についての注意点

- 本資料では 2023 年 01 月時点のサービス内容および価格について
ご説明しています。最新の情報は AWS 公式ウェブサイト
(https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式
ウェブサイト記載の価格に相違があった場合、
AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税を
ご請求させていただきます

自己紹介

名前：榎本 貴之 (Enomoto, Takayuki)

所属：アマゾンウェブサービスジャパン
　アナリティクス事業本部
　ソリューションアーキテクト部
　アナリティクスソリューションアーキテクト

経歴：インフラエンジニア @システムインテグレーター
-> インフラエンジニア @ゲーム会社
-> Cloud Support Engineer @AWS
-> **Solution Architect @AWS**

好きなAWSサービス: **Amazon OpenSearch Service**,
　Amazon QuickSight, Amazon Neptune,
　Amazon Kinesis, AWS Config,
　Amazon CloudWatch, **AWS Support**

アジェンダ

1. OpenSearch における全文検索
2. 検索システム構成
3. 高度な検索機能

Amazon OpenSearch Service について

OpenSearch

オープンソースの分散型検索・分析スイート

OpenSearch Project によって開発され、Apache 2.0 ライセンスで提供されている

データストア、検索エンジンの **OpenSearch**、
可視化、UI ツールの **OpenSearch Dashboards** から構成されている

セキュリティ、パフォーマンス分析、機械学習など
様々なプラグインによる機能拡張が可能

Amazon OpenSearch Service

OpenSearch を簡単にデプロイ・管理、
スケール可能なフルマネージドサービス

フルマネージド: リソースのデプロイ、
管理に費やす時間を削減

セキュリティ: 認証、認可、暗号化、監査、
およびコンプライアンスのための高度な
セキュリティを維持

データ分析・オブザーバビリティ:
潜在的な脅威を体系的に検出し、機械学習、
アラート、可視化を活用して対処

コスト最適化: 各種リソースを最適化し、
戦略的な作業に注力

Amazon OpenSearch Service 概要

サービス概要については
“AWS Black Belt Online Seminar Amazon OpenSearch Service” を参照のこと

https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_Amazon-OpenSearch-Service-Basic_0131_v1.pdf

検索領域における OpenSearch のユースケース

半構造化データ、非構造化データの様々な側面や属性から、最適な製品、サービス、ドキュメント、回答を素早く発見する

コスト、セキュリティ、規模の要件に合わせて、関連する検索結果をリアルタイムで取得する

INDUSTRY USE CASES

e コマースプラットフォーム: 適切な製品をすばやく見つける

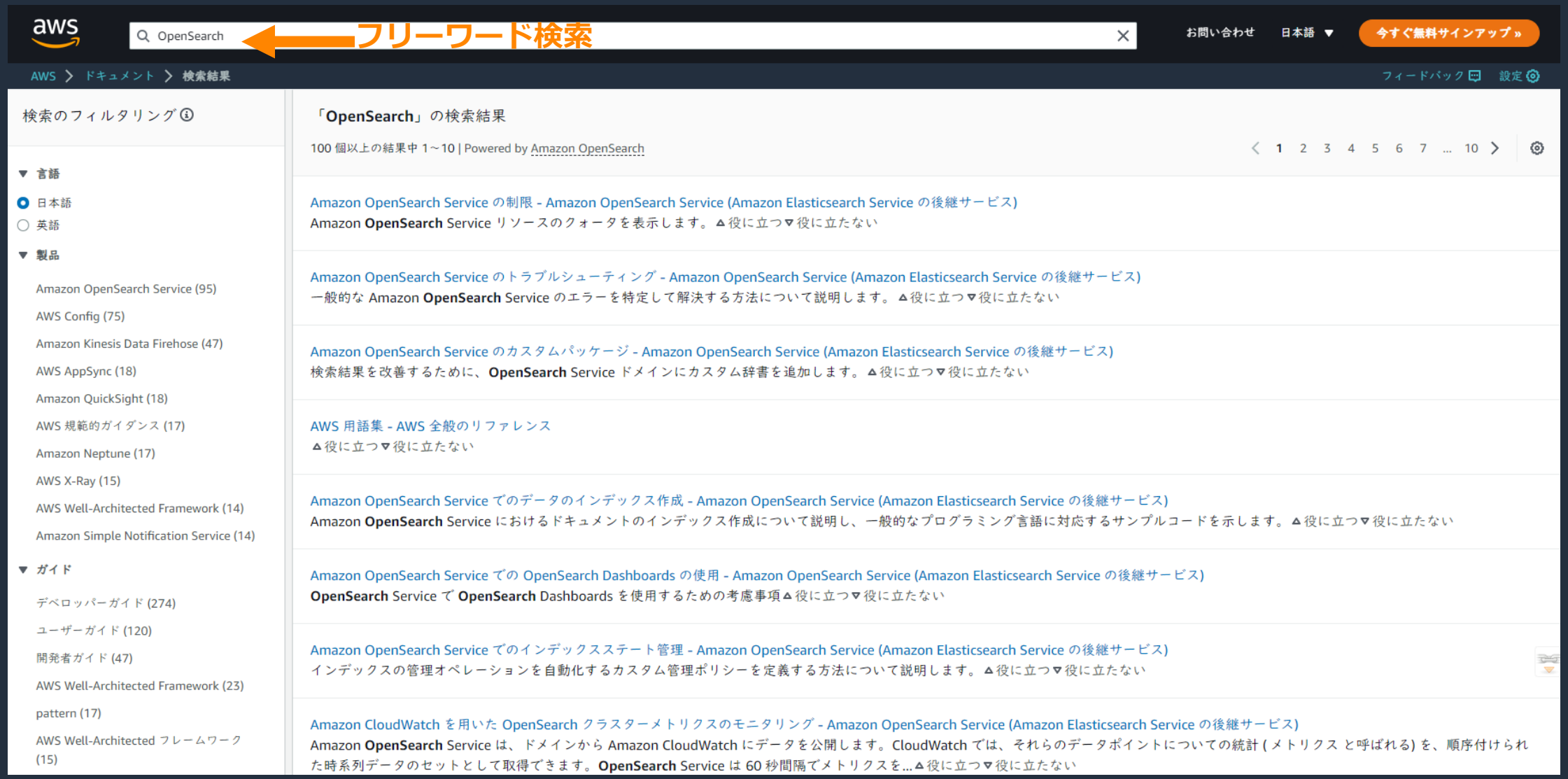
ドキュメントポータル (科学研究記事、投資分析、または診療録(カルテ)): スピーディーかつ関連性の高いドキュメント検索体験

レコメンデーションエンジン (ウィークリープレイリスト、レシピ): パーソナライズされたレコメンデーションを提供することで、ユーザーエンゲージメントを高める

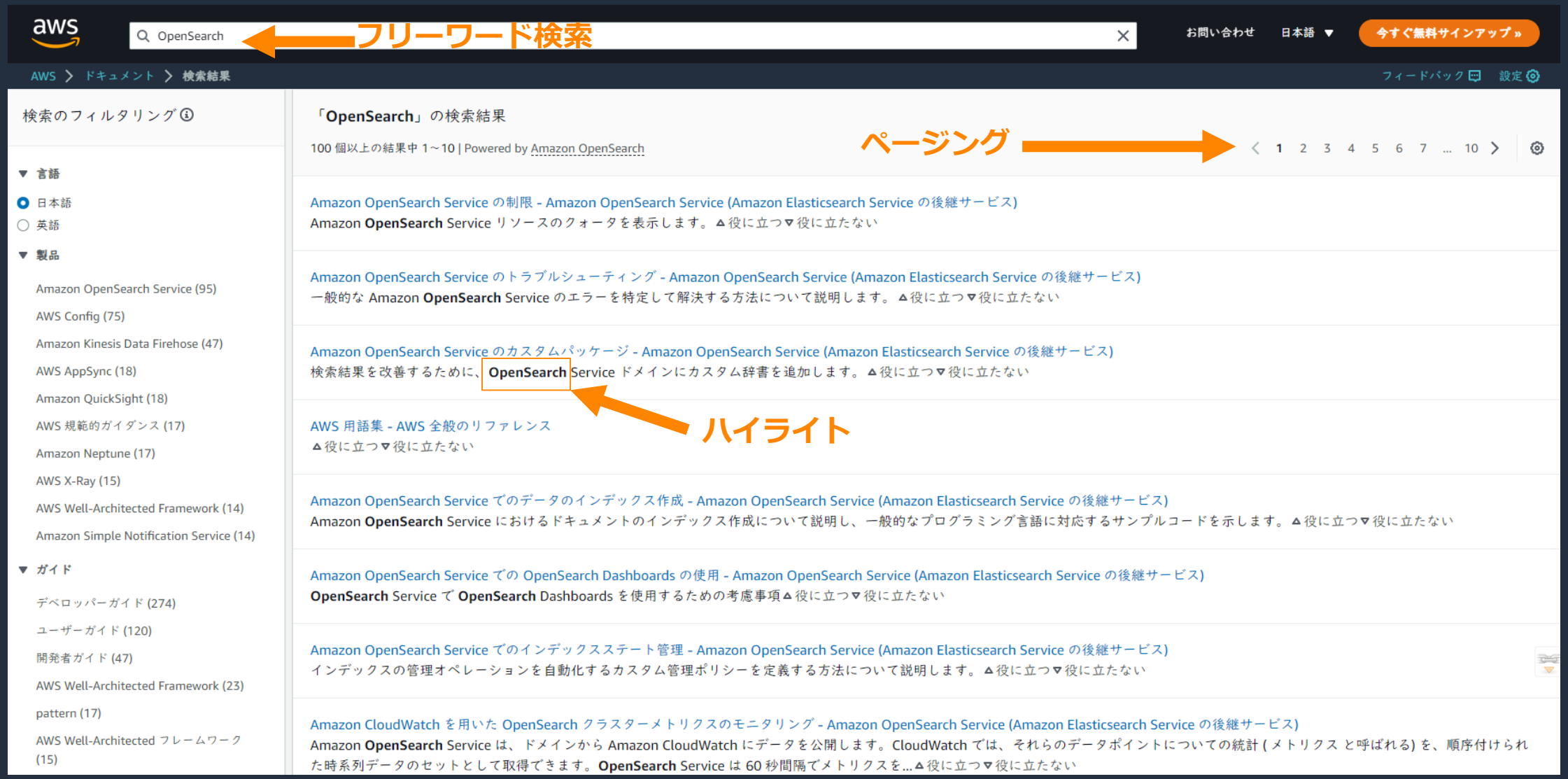
プラットフォーム検索サービス: 機械学習機能を備えた、使いやすくスピーディな検索体験

OpenSearch における全文検索

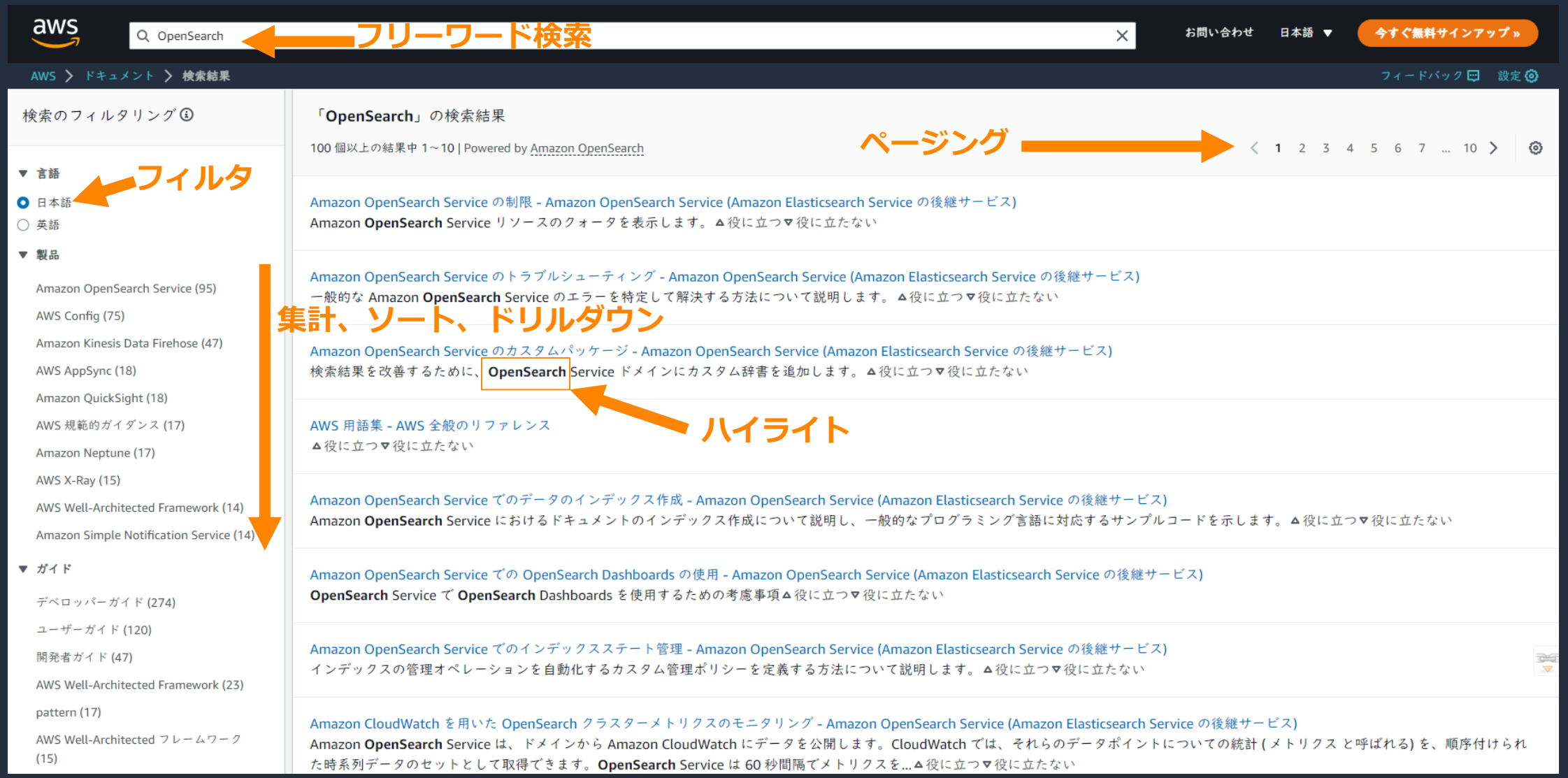
検索？



検索？



検索？



OpenSearch の代表的なデータ構造

目的に応じた複数のデータ構造を採用することで処理を高速化

転置インデックス(全文検索)

ID=1

吾輩は猫である

ID=2

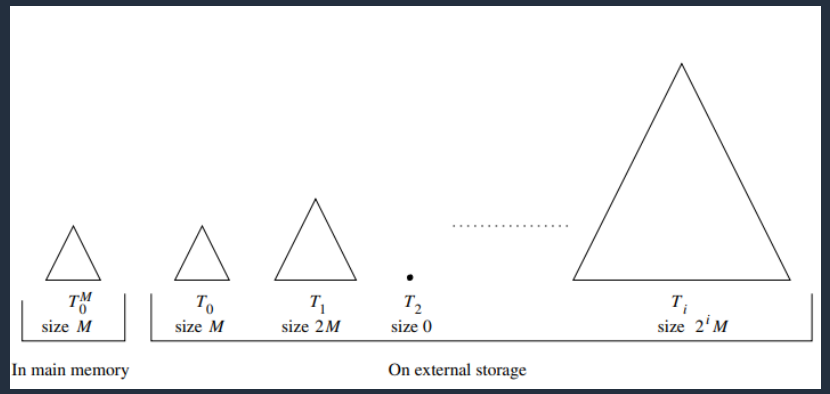
長靴を履いた猫

Term	Doc1	Doc2
吾輩	1	
猫	3	5
長靴		1
履い		3

カラム指向データ構造(集計・ソート)

Userid	username	score
1	alice	1000
2	bob	3000
3	carol	4000
4	dave	2000

Bkd-Tree (数値/地理検索)



https://users.cs.duke.edu/~pankaj/publications/papers/bkd-sstd.pdf

順次検索による全文検索の課題

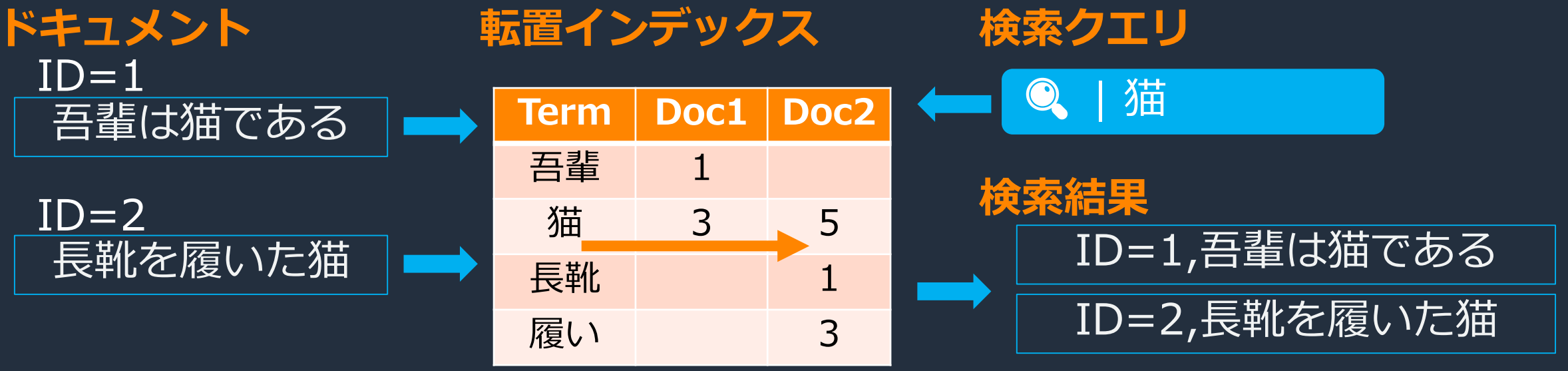
- Unix の grep コマンドのように上から全てのデータを照合していく
- 文書量に応じて処理時間が長くなる

```
$ grep -n Lucene README.txt

1:# Apache Lucene README file
5:Lucene is a Java full-text search engine. Lucene is not a complete
9: * The Lucene web site is at: http://lucene.apache.org/
10: * Please join the Lucene-User mailing list by sending a message to:
18:  The compiled core Lucene library.
23:To build Lucene or its documentation for a source distribution, see BUILD.txt
```

転置インデックス

- OpenSearch は全文検索で転置インデックスを使用している
- キーワードがどの文書に存在するかを索引付ける方式。本の索引に類似
- ドキュメントは単語に分割され、転置インデックスに登録される
- 単語の位置情報も転置インデックスに登録されるため、単語が一定の順番に並んでいる時のみ検索にヒットする、“フレーズ検索” にも対応



転置インデックス > インデックスの流れ

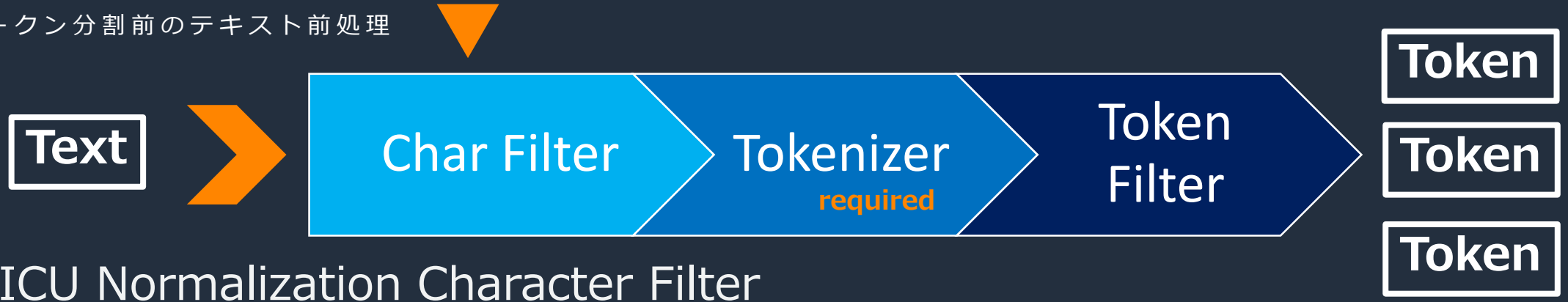
OpenSearch では 一連の処理を行うコンポーネントをアナライザーと呼ぶ。

アナライザーによるドキュメント処理の流れは以下の通り

1. Character Filter: テキスト内の文字の置換、削除 (オプション)
2. Tokenizer: テキストの分割 (必須)
3. Token Filter: テキスト分割後の後処理 (オプション)

アナライザーの構成要素 > Character Filter

トークン分割前のテキスト前処理



ICU Normalization Character Filter

記号->全角カナ変換	ｱﾟｰﾄ	→	アパート
半角カナ->全角カナ変換	ｱﾊﾟｰﾄ	→	アパート
丸数字->半角数字変換	①	→	1
全角数字->半角数字変換	８	→	8

kuromoji_iteration_mark character filter

学問のすゝめ	→	学問のすすめ
時々	→	時時

アナライザーの構成要素 > Tokenizer

テキストを複数のトークンに分割

- Tokenizer は、文字列から複数のトークン(単語)を切り出すためのコンポーネント
- 英語圏では以下のような Standard Analyzer といったスペースで単語を区切るトーカナイザーが使われるが、日本語検索ではスペースで単語が区切られないため形態素解析や N-Gram が用いられる

Original text	吾輩は猫である
形態素解析 (Kuromoji)	吾輩 / は / 猫 / である
Bi-Gram	吾輩 /輩は / は猫 / 猫で / であ / ある

アナライザーの構成要素 > Tokenizer > 形態素解析

- OpenSearch では、Japanese (kuromoji) Analysis と呼ばれる日本語のプラグインが利用可能
- 形態素解析エンジンの Kuromoji がベースとなっている
- 辞書および構文解析ルールに基づき自然な形で文章が分割される
- 分割の精度は辞書に依存する。標準辞書に加えて、追加の辞書の作成・利用が可能

```
GET
_analyze?filter_path=detail.tokenizer.tokens.token,detail.tokenizer.tokens.partOfSpeech,detail.tokenizer.tokens.reading,detail.tokenizer.tokens.baseForm
{ "tokenizer" : "kuromoji_tokenizer", "text" : "吾輩は猫である", "explain": true }
{
  "detail": {
    "tokenizer": {
      "tokens": [
        { "token": "吾輩", "baseForm": null, "partOfSpeech": "名詞-代名詞-一般", "reading": "ワガハイ" },
        { "token": "は",    "baseForm": null, "partOfSpeech": "助詞-係助詞",       "reading": "ハ" },
        { "token": "猫",    "baseForm": null, "partOfSpeech": "名詞-一般",         "reading": "ネコ" },
        { "token": "で",    "baseForm": “だ”, "partOfSpeech": "助動詞",            "reading": "デ" },
        { "token": "ある", "baseForm": null, "partOfSpeech": "助動詞",            "reading": "アル" }
      ]
    }
  }
}
```

アナライザーの構成要素 > Tokenizer > N-Gram

テキストを N 文字ずつのトークンに分割する方式

メリット

- トークン分割が辞書に依存しないため、検索時のヒット率は上がりやすい

デメリット

- トークン数が増加するため、形態素解析と比べるとインデックスサイズが大きくなる
- 検索ノイズが増加しやすい(例: **京都** で検索した際に 東**京都** や 東**京都**庁 もヒット)
- 検索ノイズを削減するために、N を増やす、フレーズ検索を入れる、などの対応が発生。これらの対処はヒット率とのトレードオフを生む

Bi-Gram(2 文字区切り)

Tri-Gram(3 文字区切り)

アナライザーの構成要素 > Token Filter

分割されたトークンに対する後処理を実施

Synonym Token Filter

表記ゆれの吸収	ice, アイス, アイスクリーム	→	アイス
展開	じゃがいも	→	メークイン, 男爵いも, きたあかり, じゃがいも, ジャガイモ

kuromoji_part_of_speech

品詞除去、基本形変換	寿司, が, おいしいね	→	寿司, おいしい

ja_stop

ストップワード除去	'てにをは'は消える	→	消える

アナライザーの設定

- アナライザーはインデックス作成時に定義する。後から追加することも可能
- フィールドごとに個別のアナライザーを選択可能
- 任意の Char Filter、Tokenizer、Token Filter を組み合わせたカスタムアナライザーも利用可能

```
PUT kuromoji_sample
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "custom_kuromoji_analyzer": {
            "tokenizer": "kuromoji_tokenizer",
            "filter": ["kuromoji_baseform", "ja_stop"],
            "char_filter": ["icu_normalizer"]
          }
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "id":  { "type": "keyword" },
      "message":  { "type": "text", "analyzer": "custom_kuromoji_analyzer"}
    }
  }
}
```

アナライザーの動作確認

- _analyze API で動作を確認可能

```
POST kuromoji_sample/_analyze
{
  "text": "ＯｐｅｎｓｅａｒＣｈは①⓪⓪㌫ｵｰﾌﾟﾝｿｰｽの検索／分析スイートです",
  "analyzer": "custom_kuromoji_analyzer"
}

{
  "tokens": [
    {"token": "opensearch", "start_offset": 0, "end_offset": 10, "type": "word", "position": 0},
    {"token": "100", "start_offset": 11, "end_offset": 14, "type": "word", "position": 2},
    {"token": "パーセント", "start_offset": 14, "end_offset": 15, "type": "word", "position": 3},
    {"token": "オープン", "start_offset": 15, "end_offset": 20, "type": "word", "position": 4},
    {"token": "ソース", "start_offset": 20, "end_offset": 23, "type": "word", "position": 5},
    {"token": "検索", "start_offset": 24, "end_offset": 26, "type": "word", "position": 7},
    {"token": "分析", "start_offset": 27, "end_offset": 29, "type": "word", "position": 8},
    {"token": "スイート", "start_offset": 29, "end_offset": 33, "type": "word", "position": 9}
  ]
}
```

カスタム辞書

- 形態素解析の際に、固有名詞やドメイン用語などを正しく判別するために、カスタム辞書を利用可能。
- インデックス設定にカスタム辞書を直接記述することが可能
- 辞書定義はインデックス作成後に変更不可。変更時にはインデックスの再作成が必要

```
PUT my_index
{
  "mappings" : {
    "properties" : {
      "content" : {
        "type" : "text",
        "analyzer": "my_analyzer"
      }
    }
  },
  "settings": {
    "index": {
      "analysis": {
        "tokenizer": {
          "my_kuromoji_tokenizer": {
            "type": "kuromoji_tokenizer",
            "mode": "search",
            "user_dictionary_rules": [
              "高輪ゲートウェイ,高輪 ゲートウェイ,タカナワ ゲートウェイ,カスタム名詞"
            ]
          }
        }
      }
    }
  }
}
```

カスタムパッケージ

- S3 に配置したテキストファイルをノードにインポートする機能
- 以下のファイルに対応
  - シノニムリスト (検索アナライザーでは**動的更新**が可能)
  - ストップワードリスト
  - カスタム辞書(Kuromoji Analysis, IK Analysis)
- 作成したパッケージは複数ドメインに適用可能
  - クロスリージョン、クロスアカウントは非サポート


```
PUT my-index
{
  "settings": {
    "index": {
      "analysis": {
        "analyzer": {
          "my_analyzer": {
            "type": "custom",
            "tokenizer": "standard",
            "filter": ["my_filter"]
          }
        },
        "filter": {
          "my_filter": {
            "type": "synonym",
            "synonyms_path": "analyzers/F234516907",
            "updateable": true
          }
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "standard",
        "search_analyzer": "my_analyzer"
      }
    }
  }
}
```


https://docs.aws.amazon.com/ja_jp/opensearch-service/latest/developerguide/custom-packages.html

ノーマライザーによる keyword フィールドの正規化

- アナライザーは “text” 型のフィールドでのみ利用可能
- keyword 型のフィールドに対して Character Filter、Token Filterに相当する Filter を適用する場合は、アナライザーではなくノーマライザーを設定する
- 1 文字ずつ変換処理を行うフィルターのみが使用可能。ICU Normalization Character Filter を使用した文字の正規化などはできるが、ストップワード除去などのステミングなどはできない

ノーマライザーの設定

- アナライザーと同様に
  インデックス作成時に定義
- フィールドごとに個別の
  ノーマライザーを選択可能
- 任意の Char Filter、
  Token Filter を組み合わせた
  カスタムノーマライザーを設定可

```
PUT kuromoji_sample
{
  "settings": {
    "analysis": {
      "normalizer": {
        "custom_normalizer": {
          "type": "custom",
          "char_filter": ["icu_normalizer", "kuromoji_iteration_mark"]
        }
      }
    }
  },
    "mappings": {
    "properties": {
      "name": {
        "type": "keyword",
        "normalizer": "custom_normalizer"
      }
    }
  }
}
```

ノーマライザーの動作確認

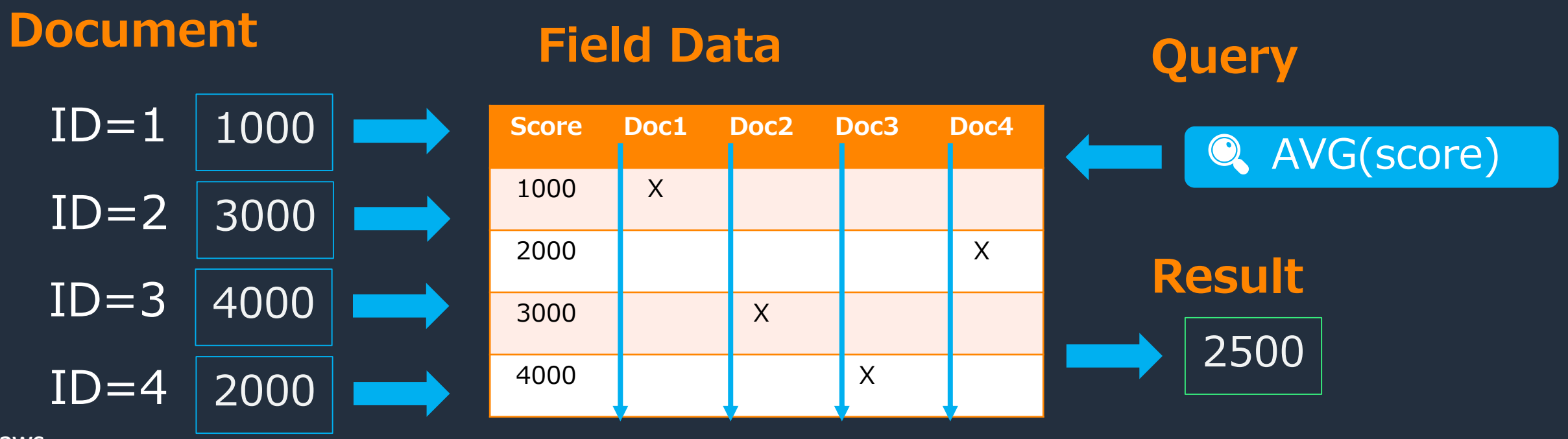
- アナライザーと同様 _analyze API で動作を確認可能
- Keyword フィールドはトークン分割されないため、返却されるトークンは一つだけとなる

```
POST kuromoji_sample/_analyze
{
  "text": "ＯｐｅｎｓｅａｒＣｈは①⓪⓪ﾊﾟｰｾﾝﾄｵｰﾌﾟﾝｿｰｽの検索／分析スイートです",
  "normalizer": "custom_normalizer"
}

{
  "tokens": [
    {
      "token": "opensearchは100パーセントオープンソースの検索/分析スイートです",
      "start_offset": 0,
      "end_offset": 35,
      "type": "word",
      "position": 0
    }
  ]
}
```

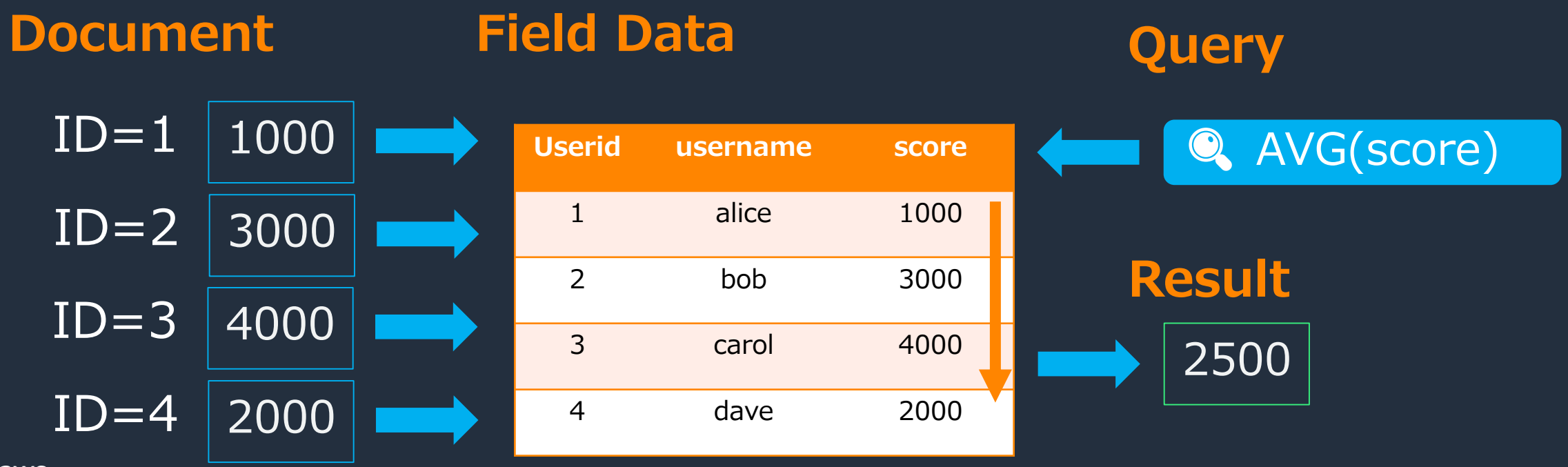
集計・ソートにおける課題

- 転置インデックスはソートや集計を行う上では非効率
- 例えば集計を行う場合、集計対象の全ドキュメントのフィールド値を探索してから集計を行うことになる



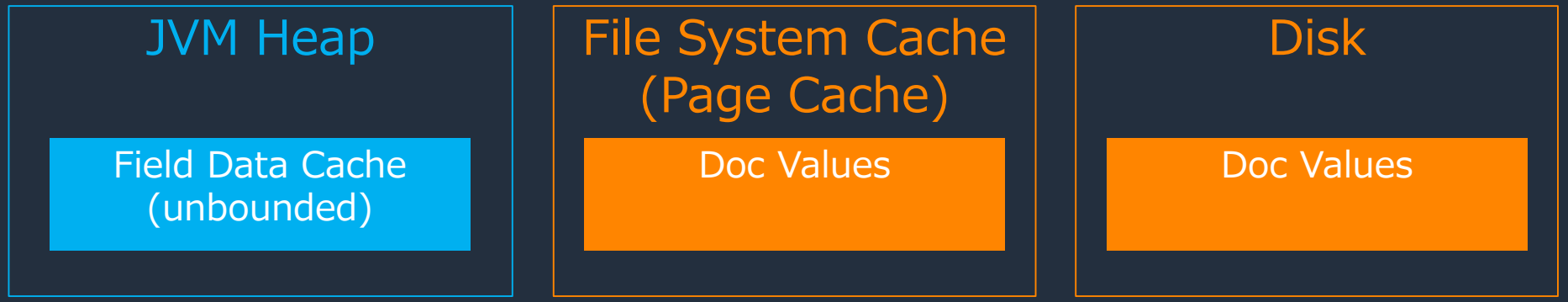
列指向データ構造によるソート・集計の高速化

- 数値型やキーワード型などソートや集計に使われるフィールドについては、列指向型のデータ構造を採用することで処理を高速化している。このデータをフィールドデータと呼ぶ
- 集計は単なる計算処理だけではなく、検索結果をドリルダウンする場面でも活用されている



列指向データ型の格納先

- 数値型、キーワード型など集計やソートに用いられることが多いフィールドについては、 **Doc Values** と呼ばれるデータ構造に格納され、ヒープ領域の使用率を下げるため**ディスク上**に保存される
  - ファイルシステムキャッシュを活用することでレイテンシを短縮している
- テキスト型については、デフォルトで列指向データは無効化されている。明示的に有効化することは可能。ただし、ディスクではなく**ヒープ領域上**の **Field Data Cache** に格納される。データ量によってはヒープ領域を大幅に占有するため、文章中のワードカウントを取りたいなど、必要な場合にのみ有効化する



列指向データ構造にかかる設定

- text 型の列指向データは、**fielddata** オプションで有効化できる

- 集計やソートに使わないフィールドについては、**doc_values** オプションで列指向データ構造を無効化することが可能。
これによりディスク領域を節約できる

- 逆に、対象フィールドを検索に使わない場合は、**index** オプションを無効化することで、
列指向データ構造にのみデータを格納することも可能。これもディスク領域の節約につながる

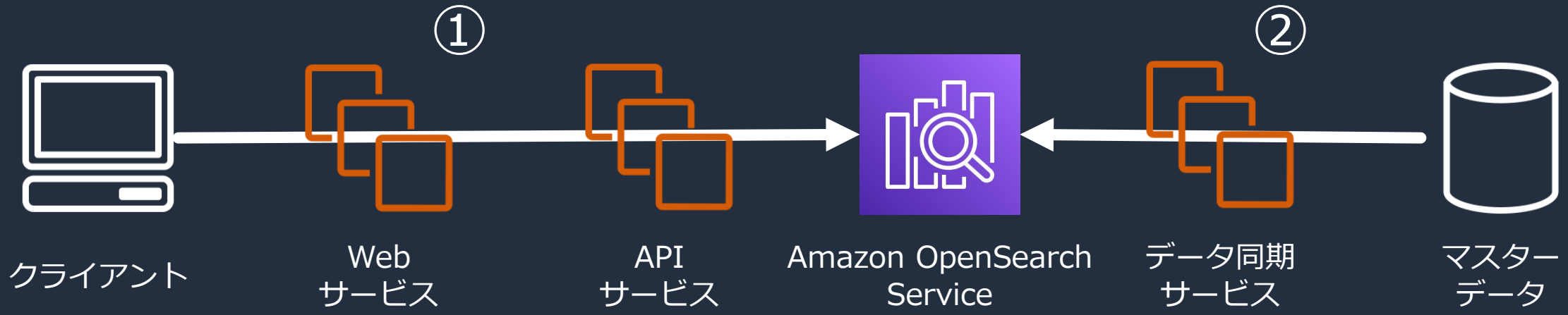
```
PUT message-index
{
  "mappings": {
    "properties": {
      "message": {
        "type":  "text",
        "fielddata": true
      },
      "user_id": {
        "type":  "keyword",
        "doc_values": false
      },
      "message_id": {
        "type":  "keyword",
        "index": false
      }
    }
  }
}
```

検索システム構成

基本的な全文検索アーキテクチャー

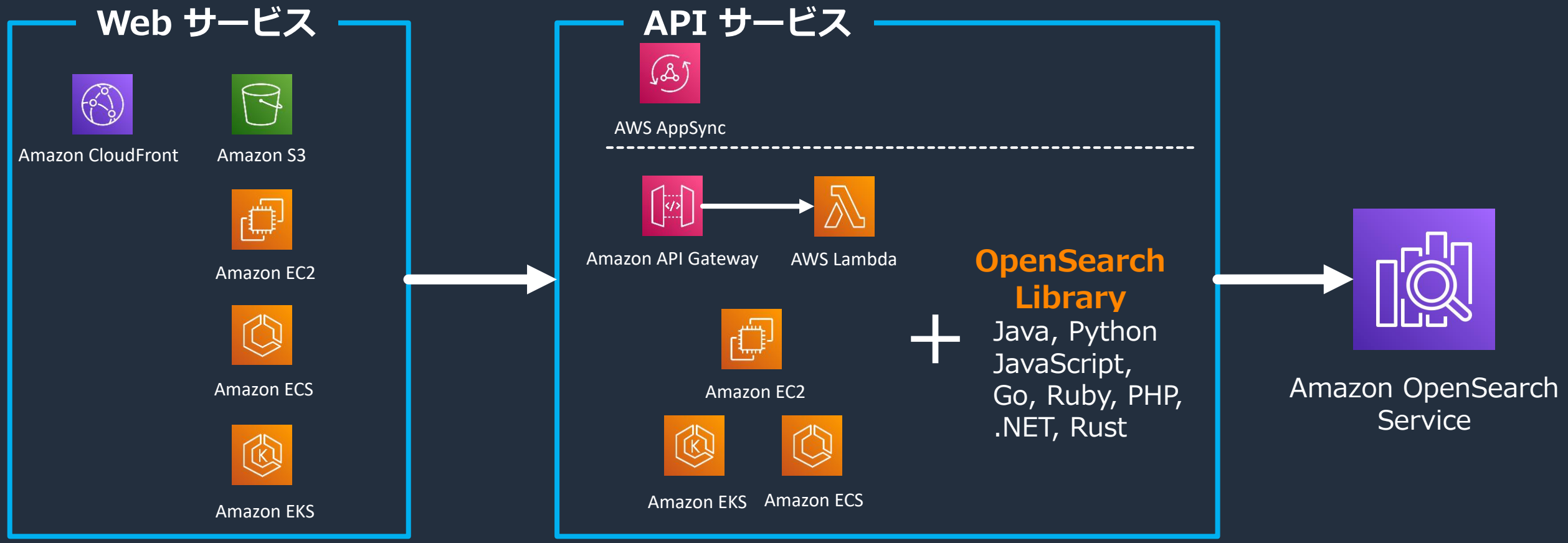
① API サービスは、クライアントから渡された検索条件を
クエリ文字列に変換し、 OpenSearch に対してクエリを発行。
得られた結果をクライアントに返却

② データ同期サービスは、データベースやファイルなどの
データをロードし、OpenSearch へ書き込む

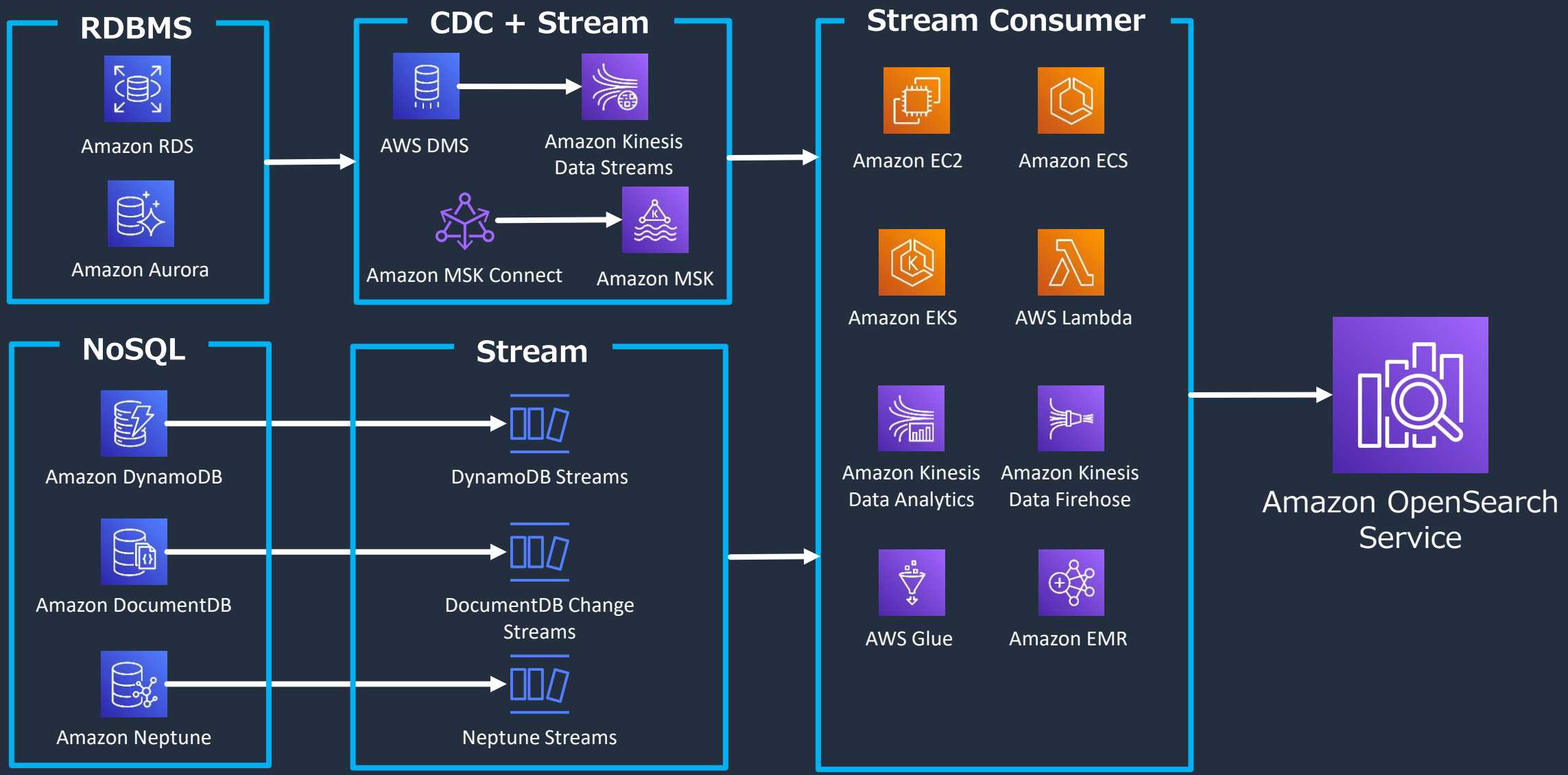


Web サービス / API サービス

- 一般的な Web 三層やサーバーレスサービスとして実装。各言語に対応した **OpenSearch クライアントライブラリ**を使用可能
- AppSync など OpenSearch クライアントを内包したサービスも利用可能

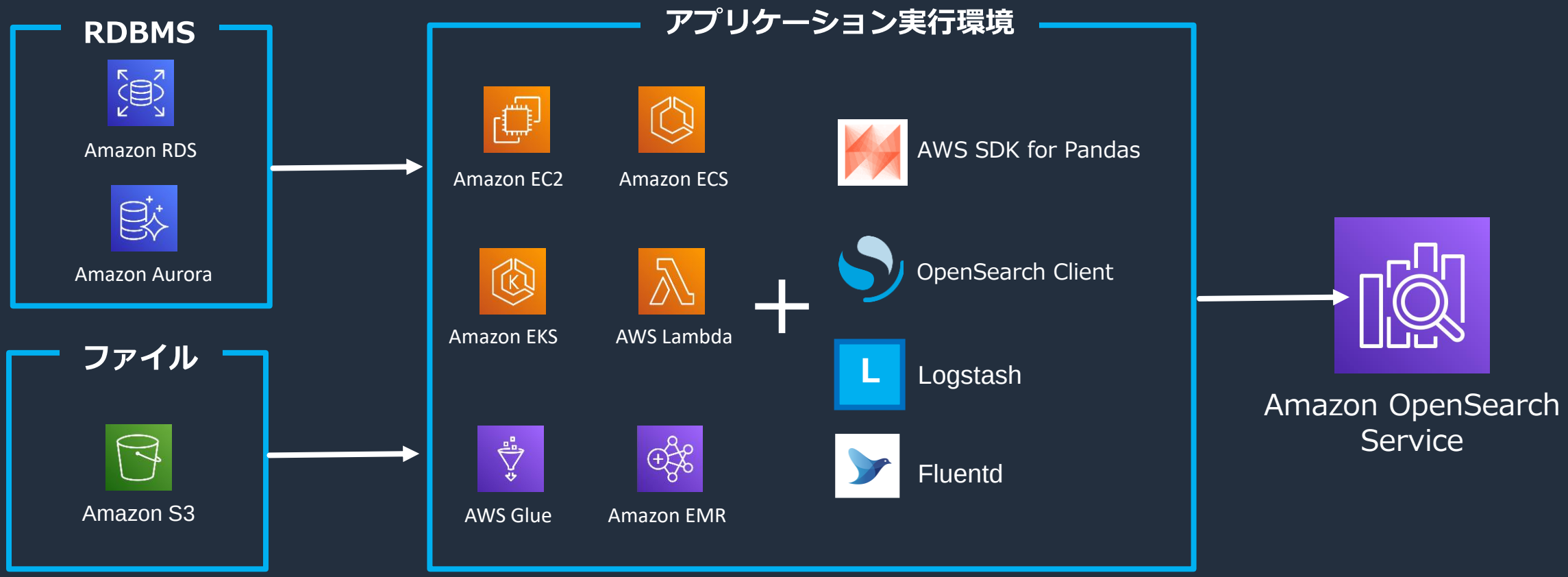


リアルタイムなデータ同期



バッチによるデータ同期

ツール、サービスによっては差分同期が可能。DB がデータソースの場合は、
レコード更新日付を格納するカラムなど、WHERE 句で前回実行時からの
更新分のみを取得できるような情報が必要



高度な検索機能

非同期検索

- OpenSearch では非同期な検索リクエストを発行することも可能
- 大量データの集計、クラスター間をまたがった検索結果の集約など、時間がかかるクエリを実行する際のタイムアウトを回避することができる
- 検索処理はバックグラウンドで実行され、非同期検索リクエスト実行時に発行された ID を使用することで結果を取得可能。
- 検索結果は任意の期間保存することが可能

1. POST _plugins/_asynchronous_search
2. Return “Asynchronous search id”
3. GET _plugins/_asynchronous_search/“Asynchronous search id”
4. Return partial or total result

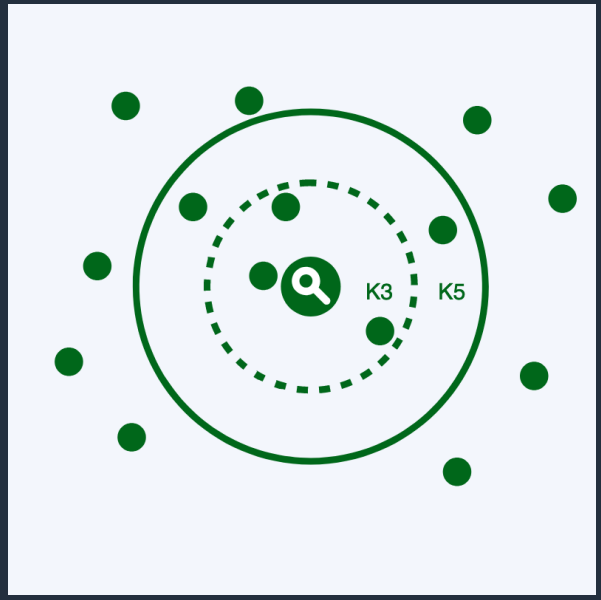
https://opensearch.org/docs/latest/search-plugins/async/index/

ベクトル検索

- ベクトル空間内の最も近い k 個の点を高速に探すための手法。
  類似検索などで使用される

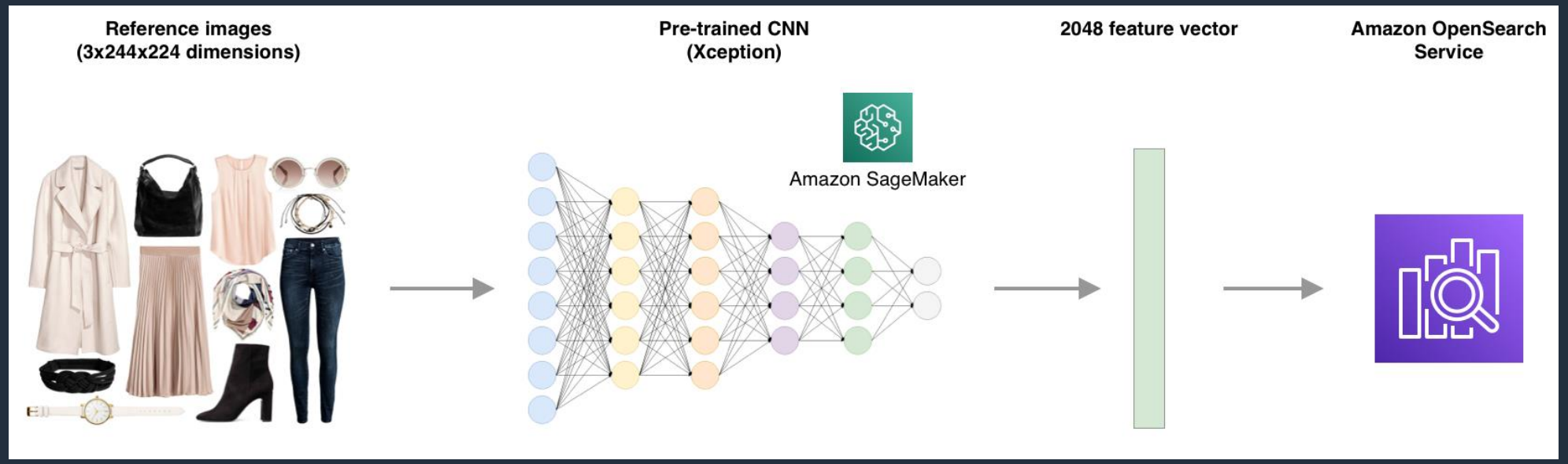
- OpenSearch では Exact kNN および Approximate kNN を利用可能。
  以下のライブラリ・アルゴリズムの組み合わせに対応
  - nmslib (HNSW)
  - Faiss (HNSW, IVF, PQ) * OpenSearch 1.2 以降
  - Lucene (HNSW) * OpenSearch 2.3 以降

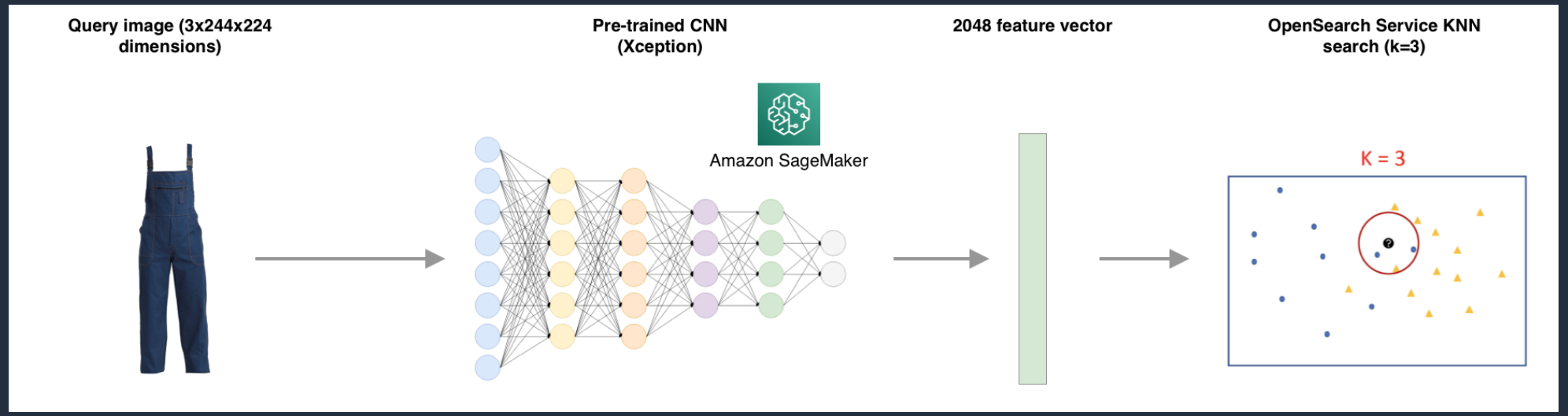
- 10 億ベクトル規模のユースケースにも対応可能
  - 参考: OpenSearch における 10 億規模のユースケースに適した k-NN アルゴリズムの選定



https://opensearch.org/docs/latest/search-plugins/knn/index/

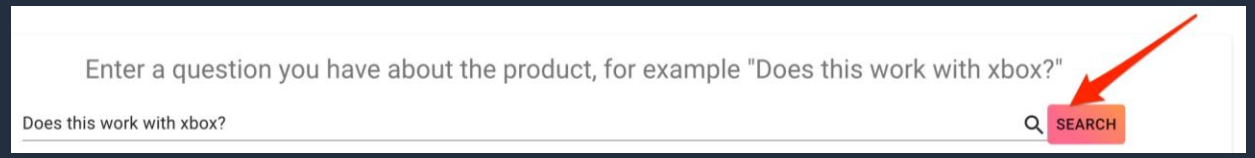
ベクトル検索実装サンプル > 類似画像検索

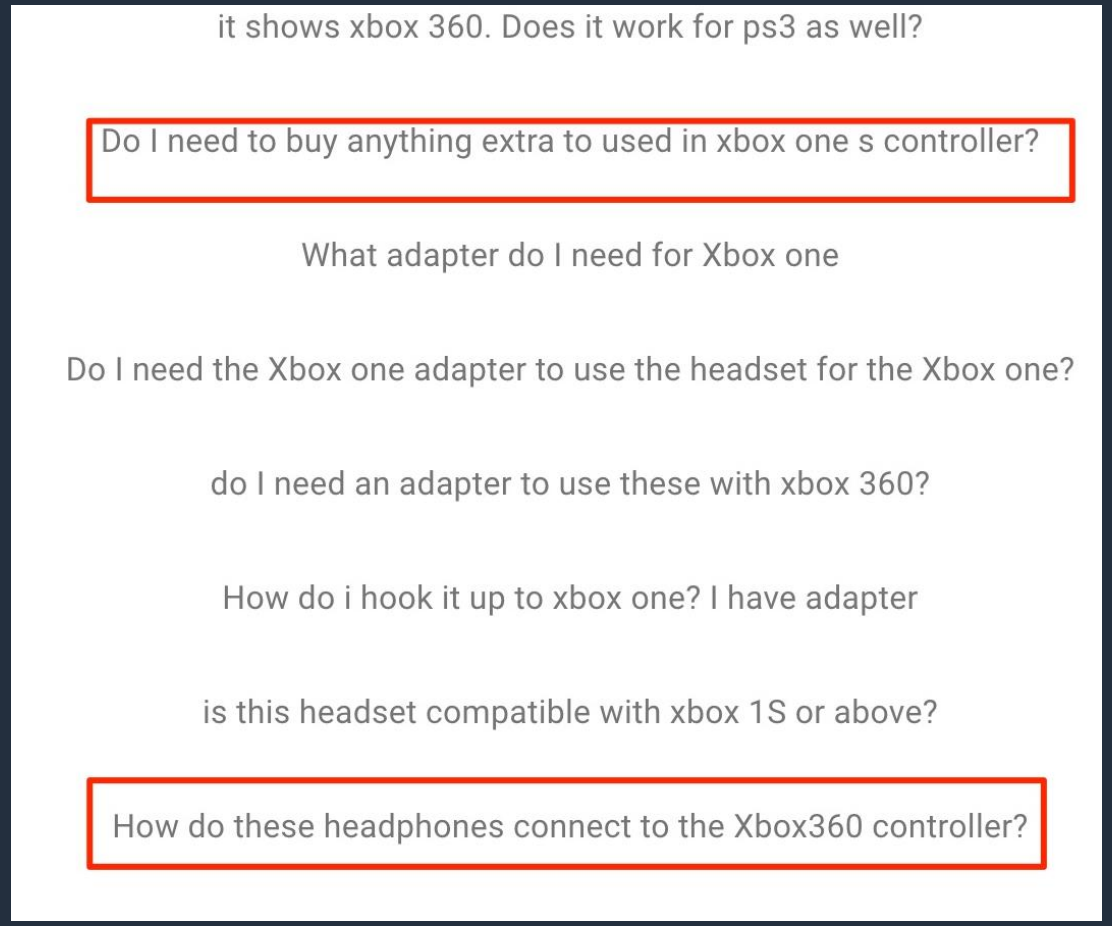




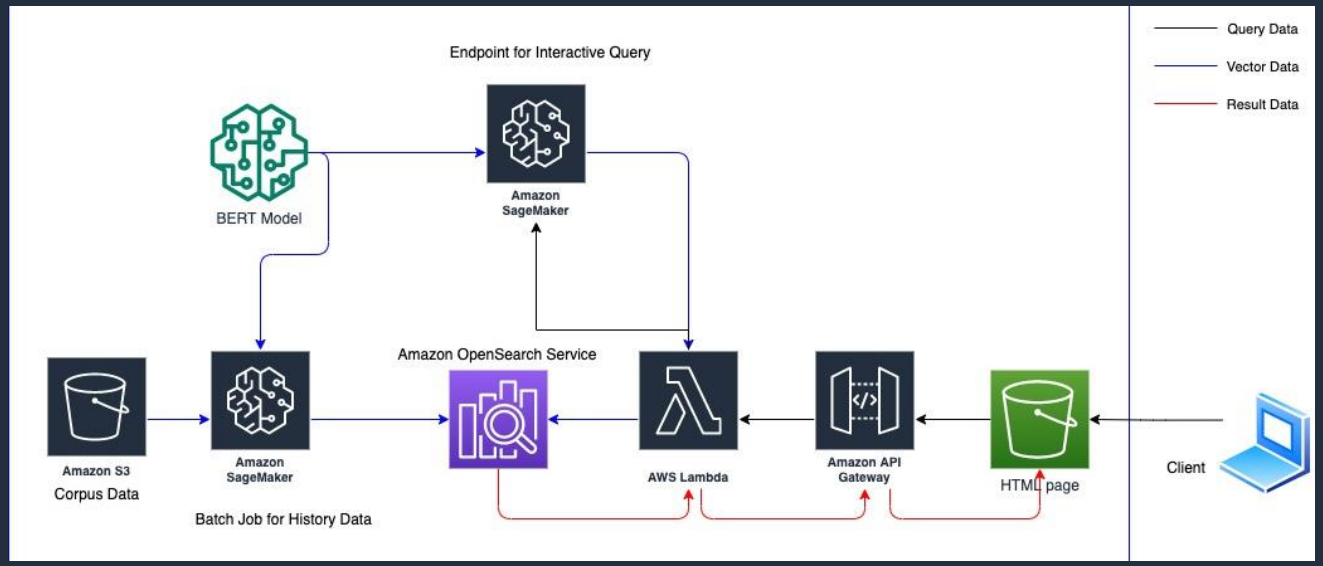
https://github.com/aws-samples/deep-visual-search-workshop

ベクトル検索実装サンプル > セマンティック検索





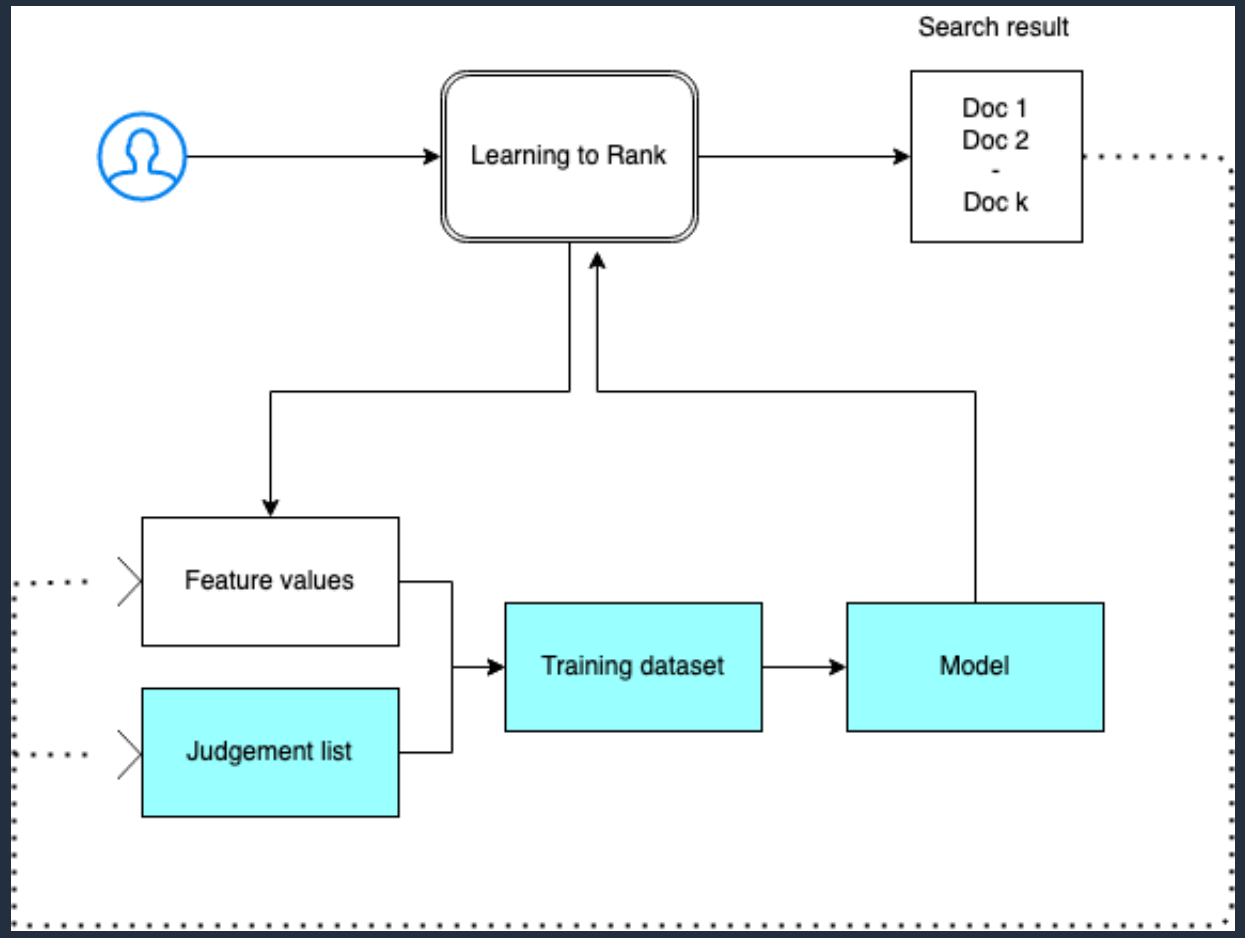
- 検索キーワードそのものに合致する検索結果ではなく、キーワードの意図や意味を読み取った検索結果を返す
- トレーニング済みの BERT モデルを使用している



https://github.com/aws-samples/semantic-search-with-amazon-opensearch

Learning to rank

- 機械学習および行動データを利用してドキュメントの関連性をスコアリング
- XGBoost および Ranklib ライブラリを使用し, 検索結果の再スコアリングを行う
- オープンソースの Learning to Rank プラグインを使用



https://github.com/o19s/elasticsearch-learning-to-rank
https://docs.aws.amazon.com/ja_jp/opensearch-service/latest/developerguide/learning-to-rank.html

SQL

- SQL による検索をサポート
- SELECT 文のみをサポートしており、データの書き込みと削除はできない
- 独自に JOIN やサブクエリなど複雑な処理もサポートしている。
  - これらの処理は OpenSearch コアエンジン外、SQL モジュールで処理されるため、通常の検索と比べてパフォーマンスは低下する点に注意
- Query Workbench、API や CLI、JDBC Driver、ODBC Driver など複数のインターフェイスに対応

```
SELECT
  a.account_number, a.firstname, a.lastname,
  e.id, e.name
FROM accounts a
JOIN employees_nested e
 ON a.account_number = e.id
```

```
SELECT a1.firstname, a1.lastname, a1.balance
FROM accounts a1
WHERE a1.account_number IN (
  SELECT a2.account_number
  FROM accounts a2
  WHERE a2.balance > 10000
)
```

https://docs.aws.amazon.com/ja_jp/opensearch-service/latest/developerguide/sql-support.html

https://opensearch.org/docs/latest/search-plugins/sql/sql/index/

PPL (Piped Processing Language)

- PPL とは、パイプ | でコマンドを繋いで処理を記述する言語

- 検索だけでなく、フィールドの値をパースし複数のフィールドに分割するなど複雑な操作も可能

```
search source=accounts | eval doubleAge = age * 2 | fields age, doubleAge;
```

age	doubleAge
32	64
36	72
28	56
33	66

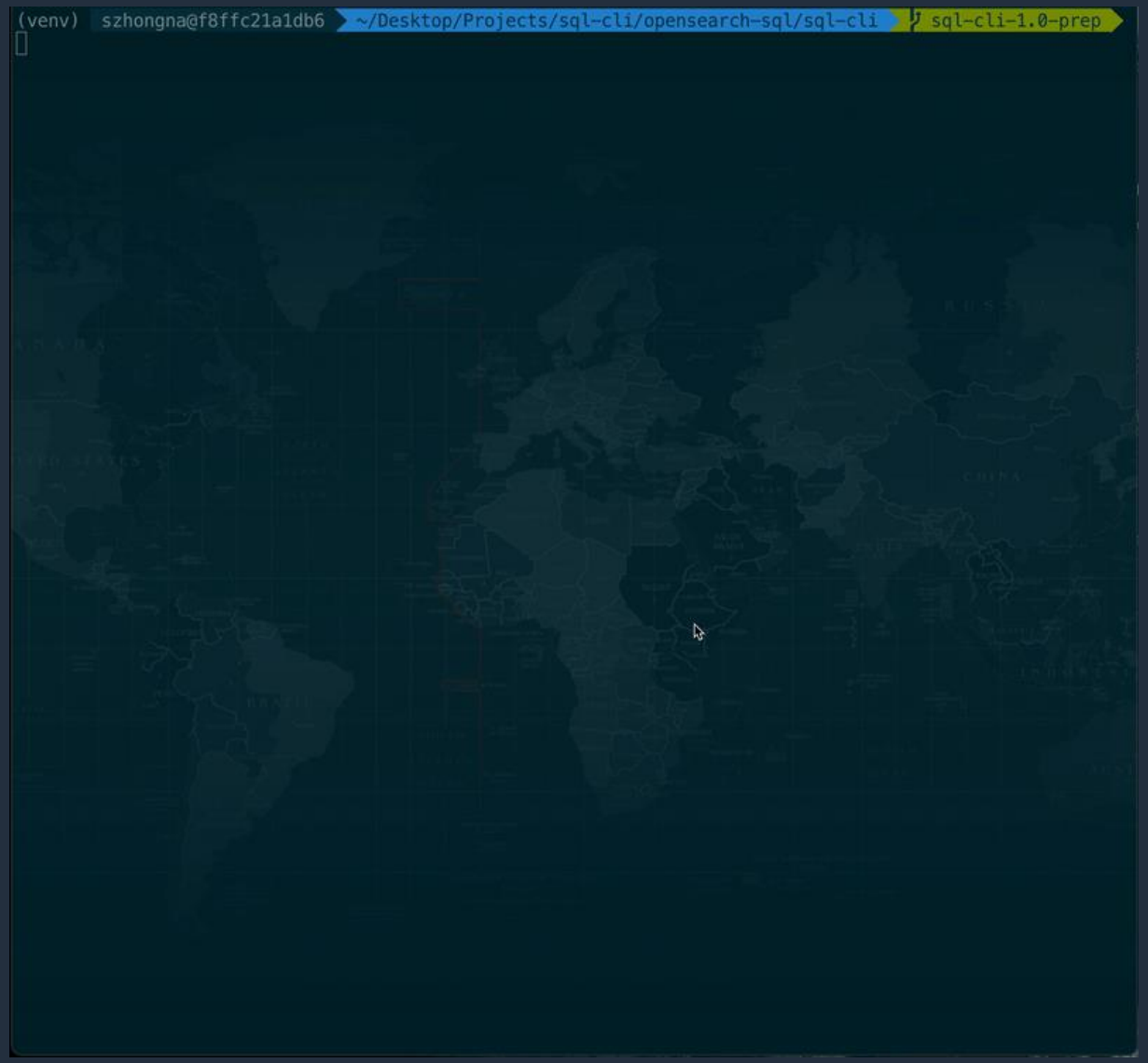
```
os> source=accounts | parse email '.+@(?<host>.+)' | fields email, host ;
fetched rows / total rows = 4/4
```

email	host
amberduke@pyrami.com	pyrami.com
hattiebond@netagy.com	netagy.com
null	null
daleadams@boink.com	boink.com

https://opensearch.org/docs/latest/search-plugins/sql/ppl/functions/
https://docs.aws.amazon.com/ja_jp/opensearch-service/latest/developerguide/ppl-support.html

SQL and PPL CLI

- コマンドラインベースの分析ツール
- SQL および PPL を使用したクエリを実行可能
- Python で実装されており、様々な環境で実行可能



https://opensearch.org/docs/latest/search-plugins/sql/cli/

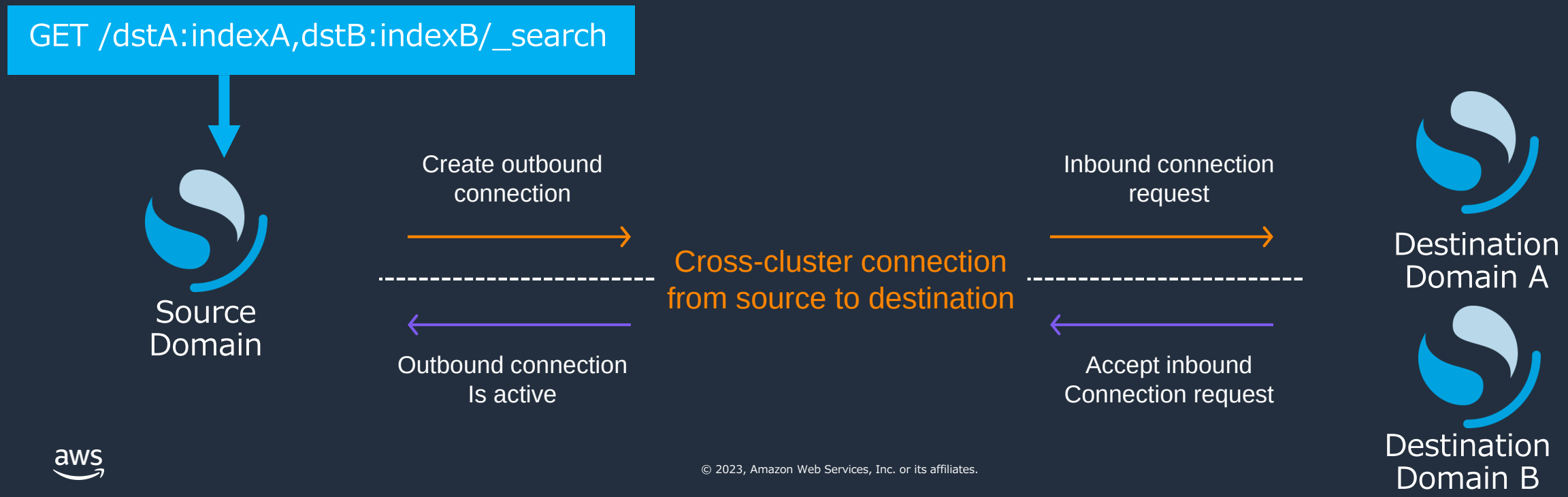
https://github.com/opensearch-project/sql/tree/main/sql-cli

Cross Cluster Search

- 複数クラスターに格納されたデータを横断的に検索、可視化
- 異なるワークロードをドメイン単位で分離することで、ワークロードに応じた適切なリソース割り当て、特定のワークロードによって発生する問題を隔離

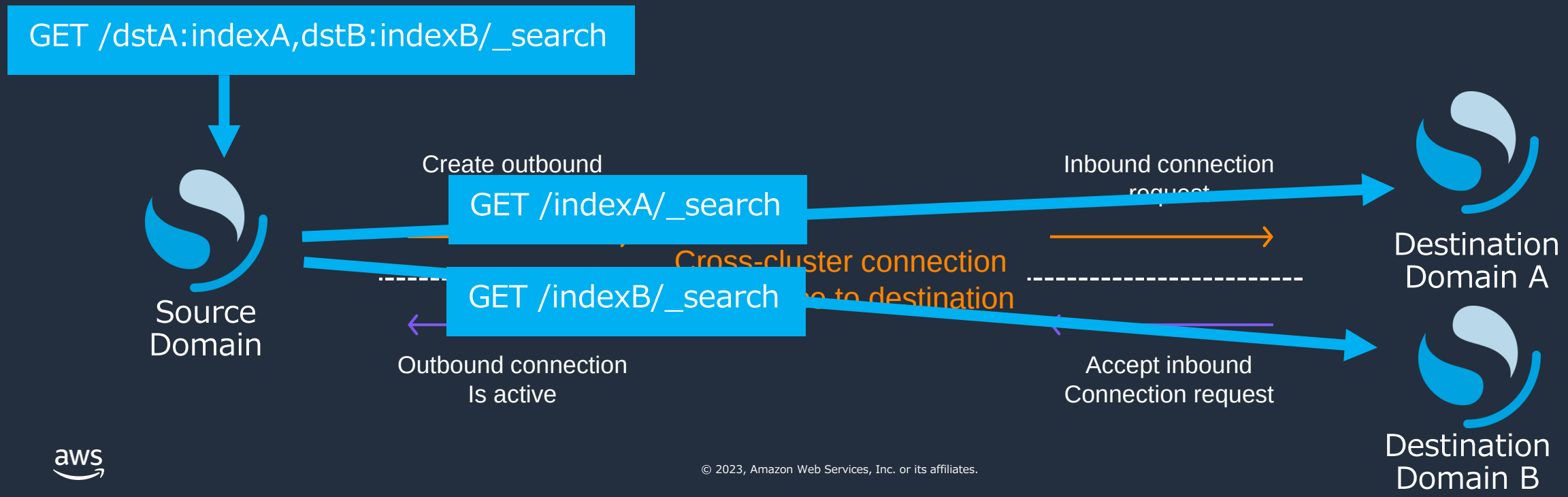
Cross Cluster Search

- 複数クラスターに格納されたデータを横断的に検索、可視化
- 異なるワークロードをドメイン単位で分離することで、ワークロードに応じた適切なリソース割り当て、特定のワークロードによって発生する問題を隔離

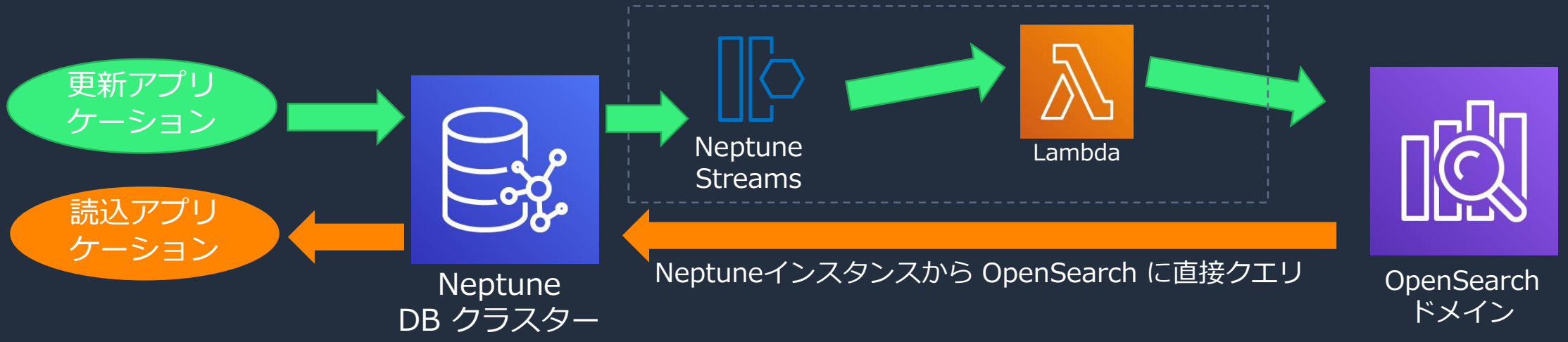


Cross Cluster Search

- 複数クラスターに格納されたデータを横断的に検索、可視化
- 異なるワークロードをドメイン単位で分離することで、ワークロードに応じた適切なリソース割り当て、特定のワークロードによって発生する問題を隔離



Amazon Neptune との連携



- Amazon Neptune の索引の代わりに OpenSearch を使用して起点ノードを決定
  - Neptune に組み込まれた拡張構文でフェデレーションクエリとして動作する
- Neptune Streams を用いた更新アーキテクチャを利用して連携することも可能

その他補足事項

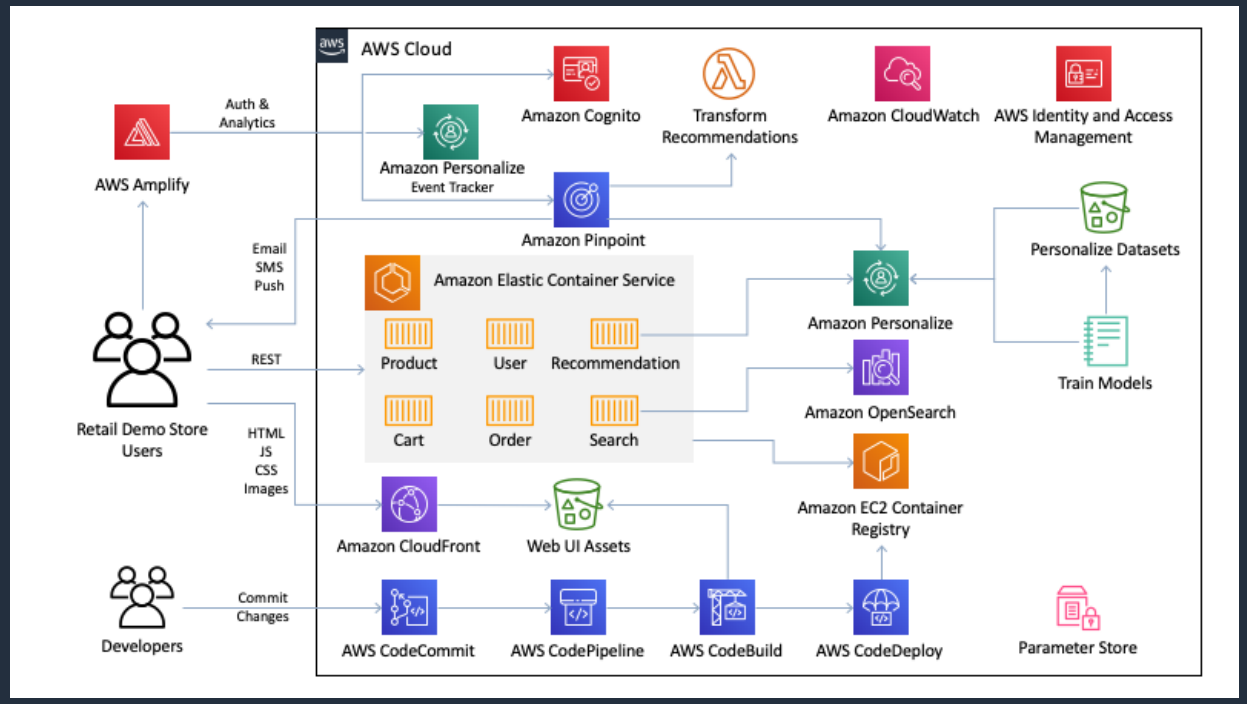
OpenSearch 入門ワークショップの紹介

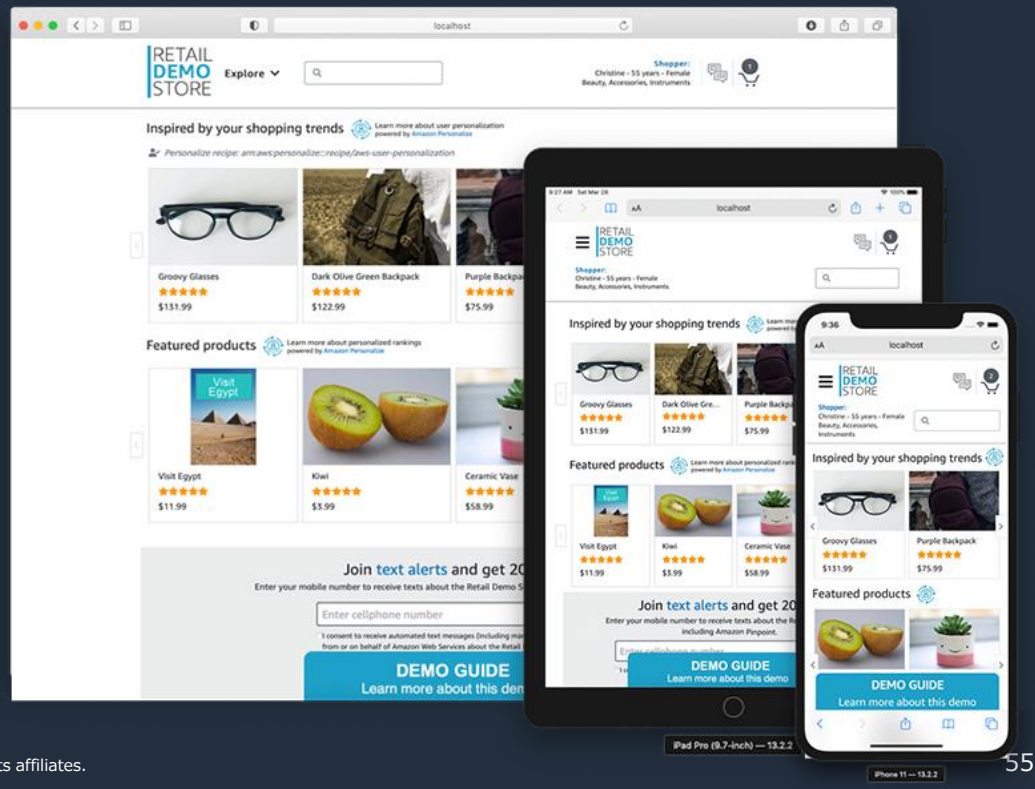
- 実際に手を動かしながら OpenSearch の基本概念や、検索の基礎について学習できるワークショップ
- 一部のラボについては Docker 等のローカル環境にインストールした OpenSearch でも進めることができる

https://catalog.us-east-1.prod.workshops.aws/workshops/26c005b2-b387-454a-b201-9b8f37f92f92/ja-JP/opensearch-concepts
https://catalog.us-east-1.prod.workshops.aws/workshops/26c005b2-b387-454a-b201-9b8f37f92f92/ja-JP/full-text-search

Retail Demo Store ワークショップ

- Amazon OpenSearch Service を含む AWS サービスから構成されたショッピングサイトのデモアプリケーション
- 特定サービスに特化したワークショップコンテンツも提供





https://github.com/aws-samples/retail-demo-store

リファレンス

よくある質問:
https://aws.amazon.com/jp/opensearch-service/faqs/

トラブルシューティング:
https://docs.aws.amazon.com/ja_jp/opensearch-service/latest/developerguide/handling-errors.html

ナレッジセンター:
https://aws.amazon.com/jp/premiumsupport/knowledge-center/#Amazon_OpenSearch_Service

料金:
https://aws.amazon.com/jp/opensearch-service/pricing/

本資料に関するお問い合わせ・ご感想

技術的な内容に関しましては、有料のAWSサポート窓口へ
お問い合わせください

https://aws.amazon.com/jp/premiumsupport/

料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください

ご感想はTwitterへ！ハッシュタグは以下をご利用ください
#awsblackbelt

その他コンテンツのご紹介

ウェビナーなど、AWSのイベントスケジュールをご参照いただけます

https://aws.amazon.com/jp/events/

ハンズオンコンテンツ

https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/

AWS 個別相談会

AWSのソリューションアーキテクトと直接会話いただけます

https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html

Thank you!