Co to jest?

CO TO JEST?

RDF RIOTECH DATA FACTORY

Przykład:

Aplikacja - wyszukiwarka przepisów

- **Wymagania biznesowe**: utworzyć aplikację będącą wyszukiwarką przepisów kulinarnych. Aplikacja powinna dawać możliwość “inteligentnego” wyszukiwania - być odporna na literówki, zawierać zaawansowane filtrowanie, znać synonimy słów itd.

Przykład:

Aplikacja - wyszukiwarka przepisów

- Potencjalny kształt systemu:

✓ Architektura: **ETL** (ze źródłami w postaci różnych blogów kulinarnych)

✓ **HDFS** - do zapisywania surowych danych

✓ **HBase** - do zapisywania przetworzonych, ustrukturyzowanych danych

✓ **Spark** - do przetwarzania danych (np. Czyszczenia, strukturyzacji, zapisu do HBase)

✓ …

✓ **Aplikcja webowa**, która jest wyszukiwarką

Przykład:

Aplikacja - wyszukiwarka przepisów

- **Skąd** aplikacja webowa **ma czerpać dane?**
  - PostgreSQL?
  - HBase?
  - Data warehouse?
- Czy zaawansowane wyszukiwanie **ma być zaimplementowane *“od 0”* w aplikacji webowej?**

Przykład:

Aplikacja - wyszukiwarka przepisów

- Technologia **full text search** (przeszukiwanie pełno-tekstowe) odpowiedzią na oba problemy

- To technologia, która pozwala **przechowywać dane** oraz **wyszukiwać je w zaawansowany sposób**.

- Solr to jedna z dwóch najpopularniejszych technologii full text search (obok Elasticsearch).

- Solr bazuje na Lucene.

Cechy Solr

- API RESTowe - żeby korzystać z Solr niekoniecznie trzeba znać Javę, można wysyłać query na API.

- Przeszukiwanie pełnotekstowe (full-text search) - możliwość przeszukiwania po tekście oraz łatwej implementacji takich rzeczy jak wykrywanie literówek.

- Baza NoSQL

- Admin Interface

- Skalowalność (dzięki pracy rozproszonej)

- Wiele możliwości analizy tekstu i zaawansowanego przeszukiwania, takiego jak filtry, statystyki, synonimy, wymiany walut itd.

Budowa Solr

RDF RIOTECH DATA FACTORY

Podstawowa terminologia

- **Instancja** (*instance*) - pojedynczy serwer zainstalowany na danej maszynie JVM
- **Węzeł** (*node*) - pojedyncza instancja solr, jeden serwer.
- **Klaster** (*cluster*) - zestaw nodów powiązanych ze sobą

Elementy Solr

- **Document** - podstawowa jednostka informacji. Odpowiednik rekordu w RDB.

- **Pole** (*field*) - element dokumentu, który przechowuje klucz-wartość. Klucz to nazwa pola, wartość to dane. Pole może posiadać wiele typów: float, long, double, date, text, Integer, boolean itd.

- **Kolekcja** (*collection*) - Zestaw dokumentów. Odpowiednik tabeli (lub indeksu).

Architektura - elementy

- **Request Handler** - przyjmuje requesty od klientów. Są różne request handlery w zależności od tego co chcemy zrobić (np. query lub index update)

- **Query parser** - parsuje query na taką strukturę, którą zrozumie Lucene. Jednocześnie szuka także potencjalnych błędów składniowych.

- **Response Writer** - komponent, który generuje ustrukturyzowany output. Solr wspiera różne formaty - np. XML, JSON czy CSV.

- **Analyzer/tokenizer** - Lucene rozpoznaje dane w formie tokenów. Solr analizuje dane które przychodzą i dzieli je na zestaw tokenów, które potem przetwarza Lucene.

- **Update Request Processor** - zestaw narzędzi, które służą do updatu danych. Ten komponent jest odpowiedzialny za modyfikacje takie jak usunięcie pola, dodanie pola itd.

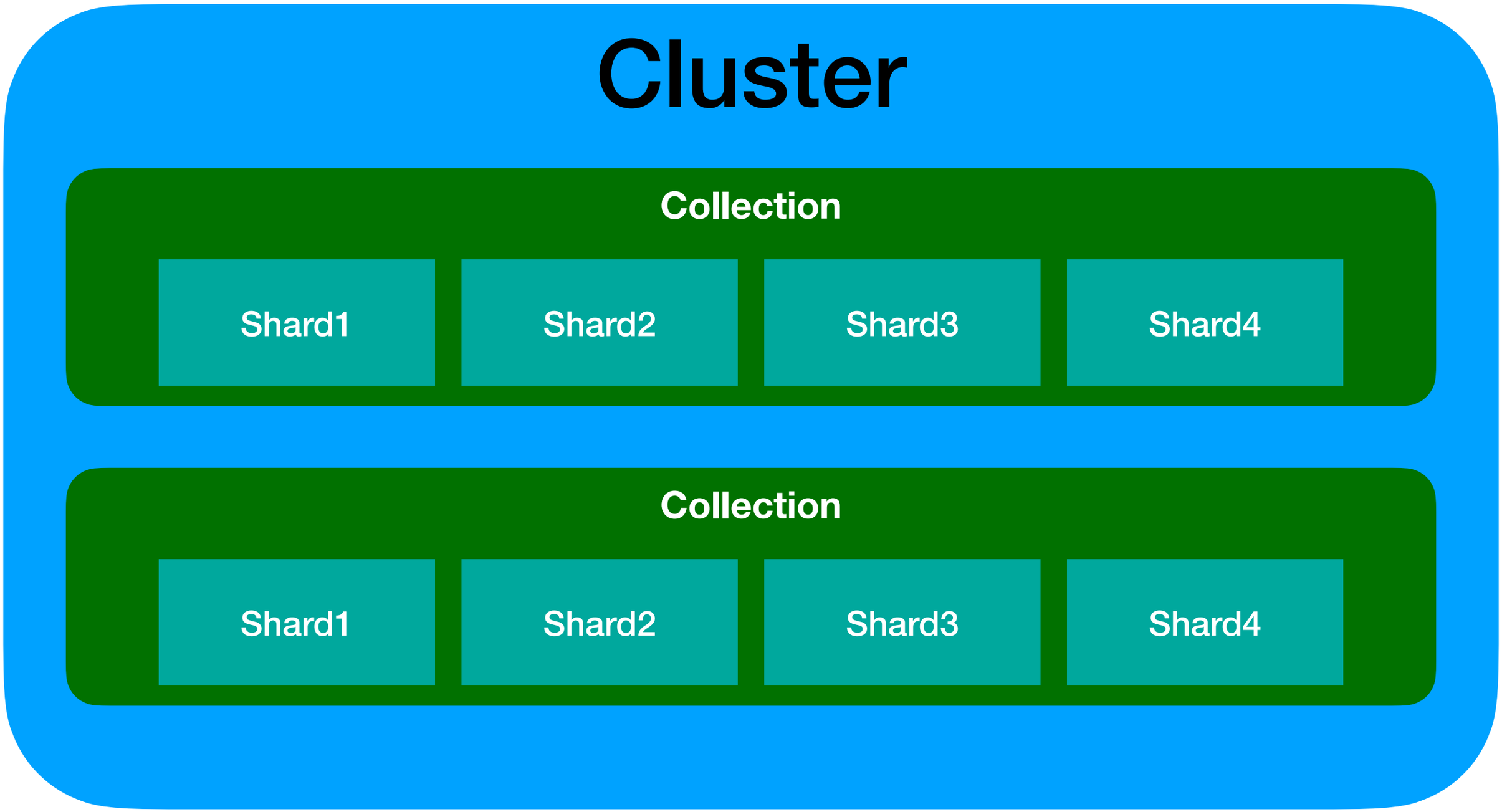
Budowa kolekcji

- **Shard/core** - element kolekcji.

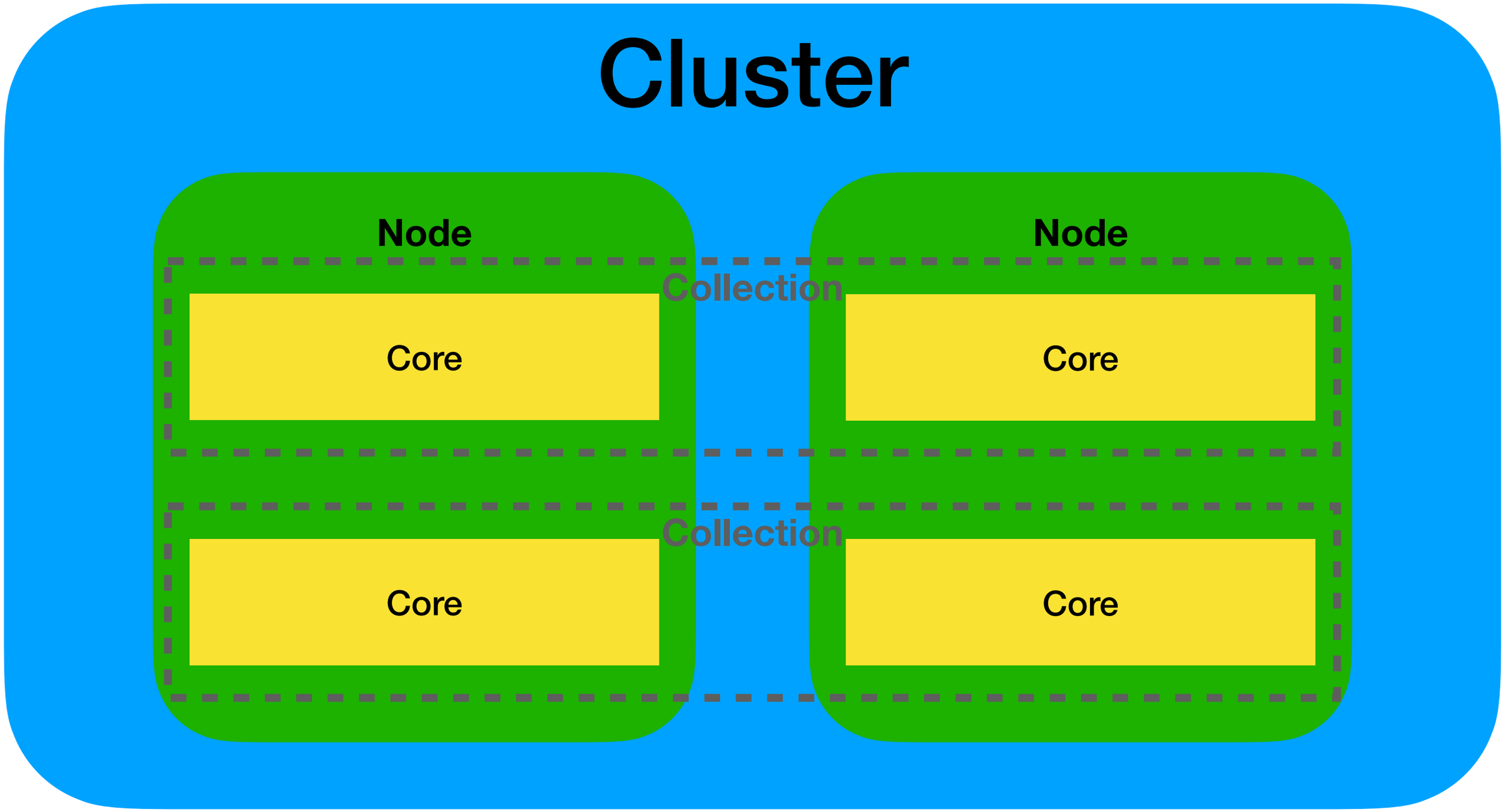
- **Replica** - kopia sharda.

- **Leader** - shard, który jest wyznaczony jako leader decyduje o zapisach, updatach itd.

Budowa logiczna



Budowa fizyczna



RDF RIOTECH DATA FACTORY

Solr Cloud

- Solr Cloud - obecnie “domyślna” konfiguracja solr.
- Zbudowany, aby można było pracować w trybie rozproszonym.
- W przeciwieństwie do większości, **architektura skupia się na rozproszeniu kolekcji**, a nie Solr jako takiego (i nie jest Master-Slave).

Solr Cloud

- Każdy shard może mieć wiele replik i musi mieć dokładnie jednego lidera. Gdy lider padnie, zookeeper wybierze następnego.
- Solr Cloud wspiera zarówno rozproszony indexing jak i rozproszone query.
- Zookeeper przechowuje konfiguracje

Solr Cloud

- To na jaki shard trafą jakie dokumenty, zależy od “shard strategy” całej kolekcji.

- Można określić implementację routingu poprzez pole “router name” przy tworzeniu kolekcji.

- Domyślnie: compositeID.

Zookeeper

- Koordynuje pracę nodów
- Tam znajdują się ustawienia (configset)
  - solrconfig.xml - konfiguracja solr w niektórych kwestiach (np. Mapowania typów)
  - managed-schema - schemat naszej kolekcji
  - inne
- Configset można wysłać poprzez polecenie *bin/solr zk upconfig -n [nazwa_konfiguracji] -d [ścieżka_do_konfiguracji] -z [adres_zookeepera]:[port_zookeepera]*

Configset

RDF RIOTECH DATA FACTORY

Czym jest?

- Zestaw konfiguracyjny potrzebny do stworzenia kolekcji.
- “Forma do rzeźby”
- 1 configset może być użyty do stworzenia wielu kolekcji
- 2 podstawowe pliki:
  - managed-schema (plik xml, pomimo braku rozszerzenia)
  - solrconfig.xml
- Inne potrzebne pliki.

solrconfig.xml

- Request Handlers
- Listeners
- Request dispatcher
- Admin Web Interface
- Parametry związane z replikacją i duplikatami
- Wczytywanie dodatkowych bibliotek z dysku serwera

```
-->
<config>

  <luceneMatchVersion>8.6.0</luceneMatchVersion>
  <dataDir>${solr.data.dir:}</dataDir>
  <!--<lib path="/technologies/solr/bins/solr-8.11.2/dist/solr-langid-8.11.2.jar"/>
  <lib path="/technologies/solr/bins/solr-8.11.2/contrib/langid/lib/langdetect-1.1-20120112.jar"/>
  -->
  <lib dir="/technologies/solr/bins/solr-8.11.2/dist" regex=".*\.jar"/>
  <lib dir="/technologies/solr/bins/solr-8.11.2/contrib/langid/lib" regex=".*\.jar"/>
  <lib dir="/technologies/solr/bins/solr-8.11.2/contrib/analysis-extras/lucene-libs" regex=".*\.jar"/>
  <lib dir="/technologies/solr/bins/solr-8.11.2/contrib/analysis-extras/lib" regex=".*\.jar"/>
  <directoryFactory name="DirectoryFactory"
                    class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}"/>

  <codecFactory class="solr.SchemaCodecFactory"/>

  <indexConfig>

    <lockType>${solr.lock.type:native}</lockType>

  </indexConfig>

  <jmx />
  <updateHandler class="solr.DirectUpdateHandler2">

    <updateLog>
      <str name="dir">${solr.ulog.dir:}</str>
      <int name="numVersionBuckets">${solr.ulog.numVersionBuckets:65536}</int>
    </updateLog>
    <autoCommit>
      <maxTime>3600000</maxTime>
      <openSearcher>false</openSearcher>
    </autoCommit>

    <autoSoftCommit>
      <maxTime>6000</maxTime>
    </autoSoftCommit>
```


Fragment solrconfig.xml

managed-schema

- Schemat kolekcji
- Zawiera:
  - Pola (Fields: nazwy i ich parametry)
  - Typy (Field Types: analyzers)
  - Kopiowanie pól i pola dynamiczne (copy fields; dynamic fields)

managed-schema

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<schema name="books-simple-schema" version="1.6">
        <field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />
        <uniqueKey>id</uniqueKey>
        <field name="title" type="text_short" indexed="true" stored="true" required="false" multiValued="false" />
        <field name="author" type="text_short" indexed="true" stored="true" required="false" multiValued="false" />
        <field name="price" type="currency" indexed="true" stored="true" required="false" multiValued="false" />
        <field name="description" type="text_general" indexed="true" stored="false" required="false" multiValued="false" />
        <field name="category" type="text_short" indexed="true" stored="false" required="false" multiValued="false" />
        <field name="language" type="string" indexed="true" stored="false" />
        <field name="languages" type="string" indexed="true" stored="false" multiValued="true"/>
        <field name="rating" type="double" indexed="true" stored="true" required="false"/>
```

Początek managed-schema

Configset

Wysyłanie na solr krok po kroku

1. Tworzymy configset (min. managed-schema + solrconfig.xml)

2. Wysyłamy cały configset na zookeepera

3. Tworzymy kolekcję z wykorzystaniem configsetu dostępnego w Solr.

managed-schema

Modyfikowanie schematu danych

- managed-schema możemy modyfikować zdalnie, za pomocą requestów http.

- Możemy także modyfikować lokalnie i wysyłać do zookeepera zmienioną wersję.

- Moja opinia: nie polecam po http!

- Po zmianie schematu należy przeindeksować kolekcję.

managed-schema

Types & analyzers

- W Solr, w managed-schema pola mają typy. Te typy muszą być jednak wcześniej stworzone.
- Definiujemy tu nie tylko typ typu (np. “short_text” typu “string”) ale także analizę.
- Analiza to proces któremu zostaje poddana fraza.
- Analiza odbywa się zarówno w kontekście indexingu jak i query

managed-schema

Types & analyzers

- Typ może zawierać 4 rodzaje informacji:
  - Nazwę pola (obowiązkowo)
  - Nazwę klasy, która implementuje (obowiązkowo)
  - Opis analizy, jeśli to “TextField”
  - Field type properties - w zależności od klasy implementacyjnej (niektóre właściwości mogą być obowiązkowe)

managed-schema

Analyzers

- Analyzer to zestaw tokenizera i filtrów.
- W efekcie ma przetworzyć podaną frazę
- Token dostaje na wejściu frazę, a zwraca zestaw tokenów
- Filtr dostaje na wejściu zestaw (listę) tokenów i zwraca także listę tokenów.
- Tokenizer musi zawsze być na samym początku i jest 1.
- Filtry ułożone są po tokenie i możemy mieć ich dowolną liczbę.
- Wyjście filtra jest wejściem kolejnego

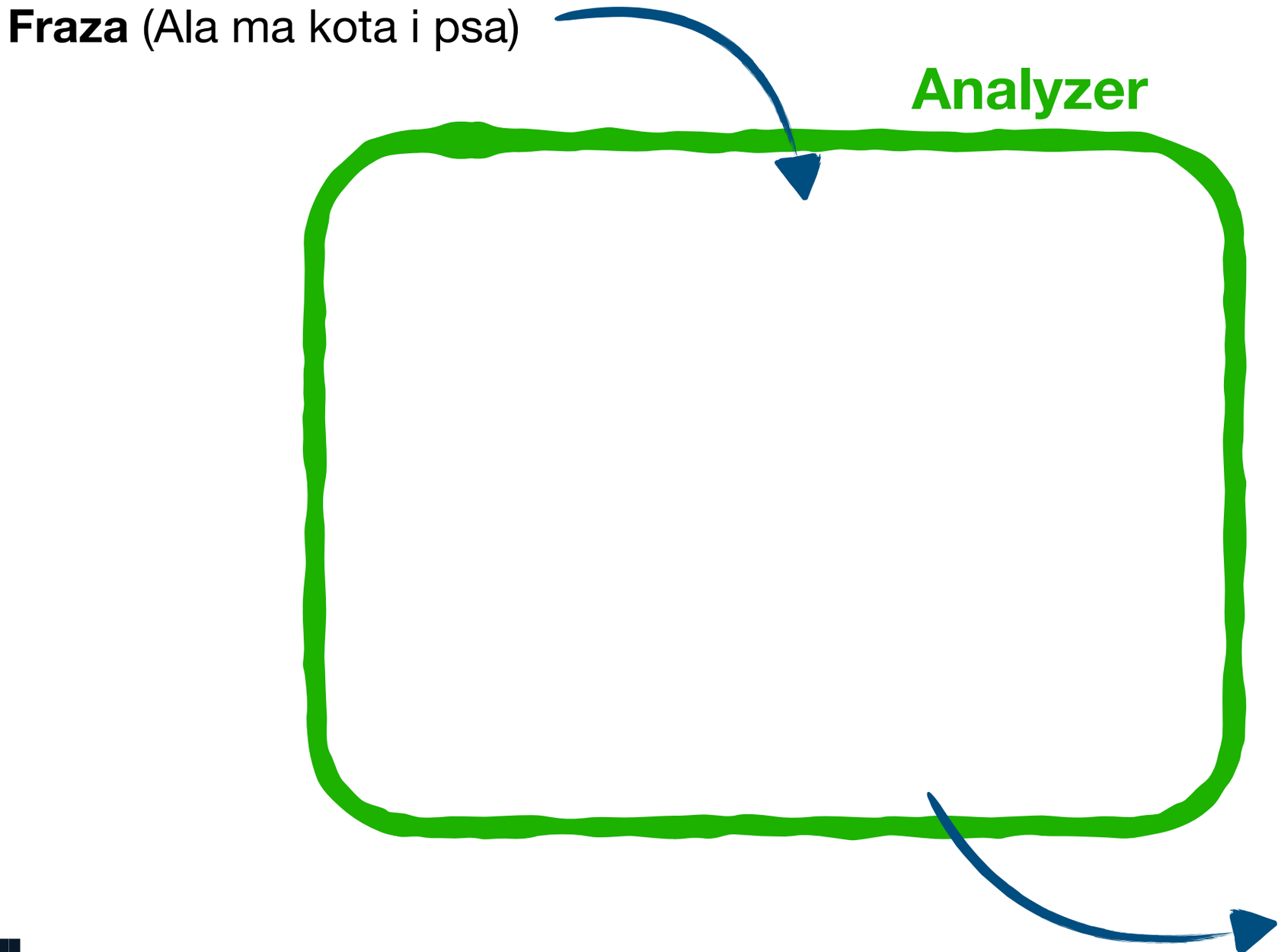
managed-schema

Analyzer

Fraza (Ala ma kota i psa)

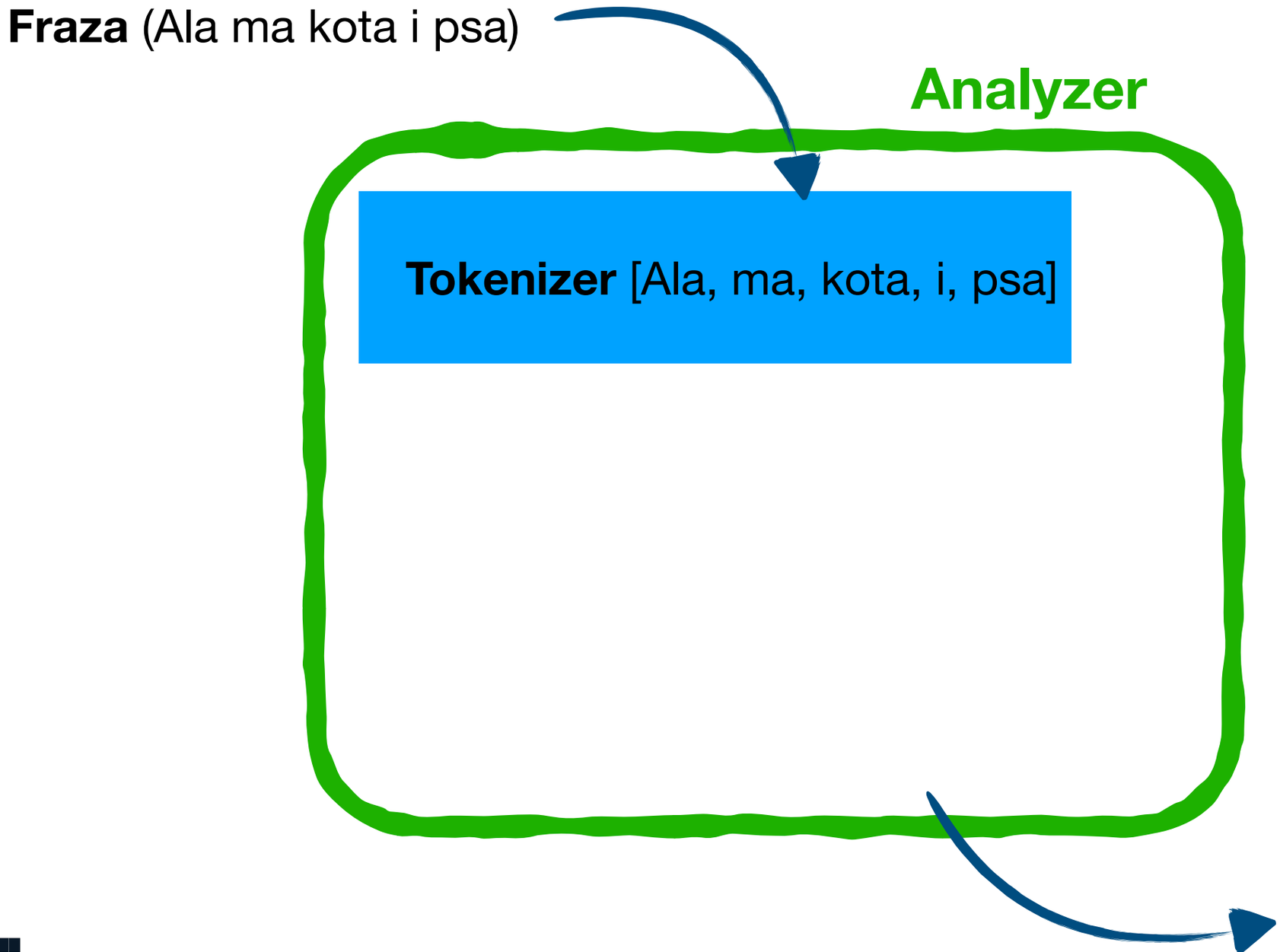
managed-schema

Analyzer



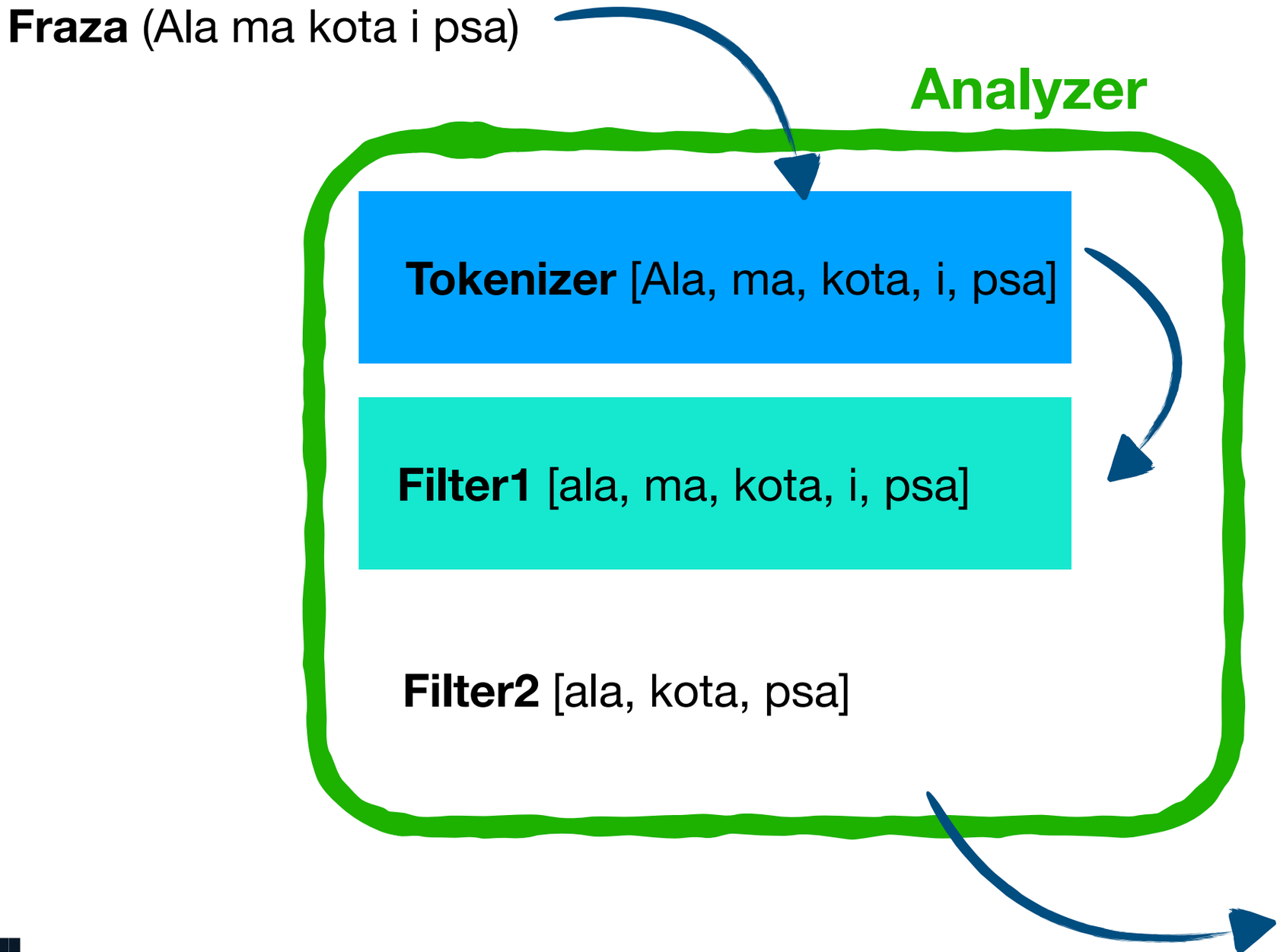
managed-schema

Analyzer



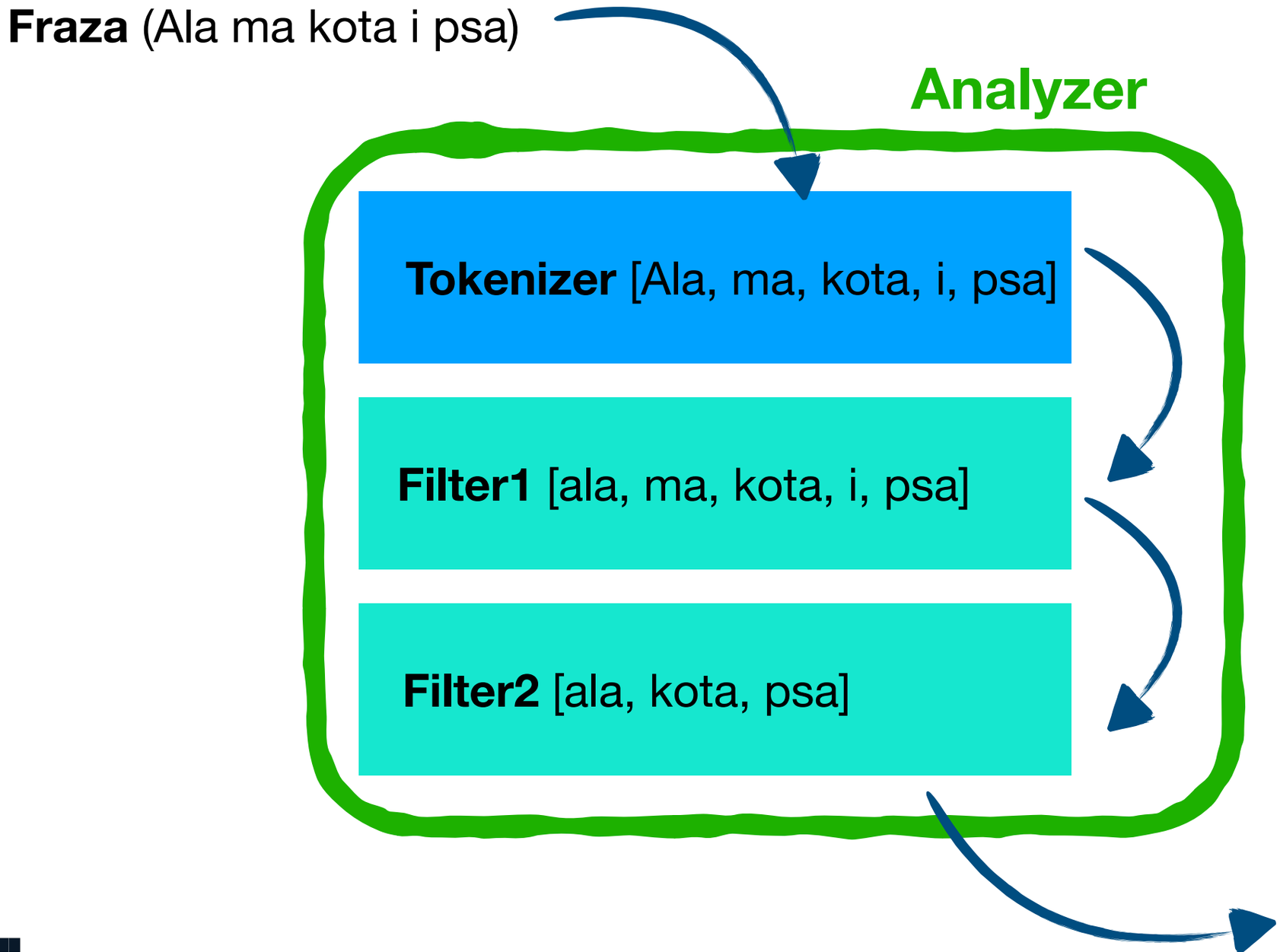
managed-schema

Analyzer



managed-schema

Analyzer



managed-schema

Types & analyzers [przykład]

```
<fieldType name="string" class="solr.StrField" sortMissingLast="true" />
<fieldType name="long" class="solr.LongPointField" docValues="true"/>
<fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>
```

managed-schema

Types & analyzers [przykład]

```
<fieldType name="text_short" class="solr.TextField" positionIncrementGap="100">
        <analyzer>
                <tokenizer class="solr.StandardTokenizerFactory" />
                <filter class="solr.LowerCaseFilterFactory"/>
        </analyzer>
</fieldType>
```

managed-schema

Types & analyzers [przykład]

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">
        <analyzer type="index">
                <tokenizer class="solr.StandardTokenizerFactory"/>
                <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
                <filter class="solr.LowerCaseFilterFactory"/>
        </analyzer>
        <analyzer type="query">
                <tokenizer class="solr.StandardTokenizerFactory"/>
                <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
                <filter class="solr.SynonymGraphFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
                <filter class="solr.LowerCaseFilterFactory"/>
        </analyzer>
</fieldType>
```

managed-schema

Types & analyzers

- Typ może zawierać 4 rodzaje informacji:
  - Nazwę pola (obowiązkowo)
  - Nazwę klasy, która implementuje (obowiązkowo)
  - Opis analizy, jeśli to “TextField”
  - Field type properties - w zależności od klasy implementacyjnej (niektóre właściwości mogą być obowiązkowe)

managed-schema

Najpopularniejsze tokenizery

- Standard Tokenizer
- Classic Tokenizer
- White Space Tokenizer
- Keyword Tokenizer
- Letter Tokenizer
- Lower Case Tokenizer
- Path Hierarchy Tokenizers

managed-schema

Pola

```
<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />
<uniqueKey>id</uniqueKey>
<field name="title" type="text_short" indexed="true" stored="true" required="false" multiValued="false" />
<field name="author" type="text_short" indexed="true" stored="true" required="false" multiValued="false" />
<field name="price" type="currency" indexed="true" stored="true" required="false" multiValued="false" />
<field name="description" type="text_general" indexed="true" stored="false" required="false" multiValued="false" />
<field name="category" type="text_short" indexed="true" stored="false" required="false" multiValued="false" />
```

managed-schema

Indexed vs Stored

- Tworząc schemę kolekcji możemy określić dwa pola: indexed oraz stored. Oba będą wskazywały na to jak pole jest postrzegane przez Solr.

- **Indexed** - to pole może posłużyć do wyszukiwania dokumentu.

- **Stored** - to pole może zostać wyciągnięte z wyszukanego dokumentu.

managed-schema

Indexed vs Stored

- Przykład: mamy kolekcję People.
- **Pole firstName:**
  - stored=true, indexed=false.
    - **Możemy zobaczyć to pole** w wyszukanych dokumentach, ale **nie możemy przeszukiwać** szukając np. firstName=marek.
- **Pole lastName:**
  - stored=false, indexed=true.
  - Odwrotna sytuacja: **wyszukamy wpisując *“lastName:Czuma”*** ale nie znajdziemy nazwiska w wyszukanych dokumentach.

managed-schema

Pola dynamiczne

- Zwykłe pola, tyle że zdefiniowane z gwiazdką.
- Gdy któreś pole które chcemy zaindeksować nie pasuje do żadnego ze zdefiniowanych, może być dopasowane do zmiennych dynamicznych.
- Przydatne, gdy mamy setki pól, których nie chcemy deklarować.

managed-schema

Pola dynamiczne

- Przykład:
  <dynamicField name="*_i" type="double" indexed="true" stored="true" />

- Wystarczy dodać dokument z polem “price_i” i zostanie to zaindeksowane jako pole double.

managed-schema

Pola dynamiczne

- Obserwacje:
  - Gdy wyszukamy po nieistniejącym polu które mogłoby istnieć, zwróci nam pustą listę wyników
  - Trzeba uważać, by nie zrobić śmietnika.
  - Problemy z wydajnością (Lucene przeznacza miejsce na każdą istniejącą unikalną kolumnę).

managed-schema

Kopiowanie pól

- Możemy w Solr kopiować pola.
- Dzieje się to **PRZED ANALIZĄ!**
- Przykład:
  <copyField source="cat" dest="cat2" maxChars="30000" />
- Wartość pola “cat” kopiowana jest do pola “cat2”.

Configset do pobrania

- Można pobrać przykładowy schemat ze strony RDF:

  http://riotechdatafactory.com/wp-content/uploads/2022/07/rdf_configset_template.tar.gz

Co dalej

1. Wchodzimy na serwer
2. Przejrzymy configset
3. Wysyłamy go na zookeepera.

Gratulacje!

Good job;-)

RDF RIOTECH DATA FACTORY