

# ESOF

## ElasticSearch



**Project of ESOF 2018 -- MIEIC**

**Head of CU: João Carlos Pascoal Faria**

**Class 3**

**Group 1**

✚ João Pedro Viveiros Franco --- up201604828 --- up201604828@fe.up.pt

✚ Tiago Pinho Cardoso --- up201605762 --- up201605762@fe.up.pt

✚ Tomás Nuno Fernandes Novo --- up201604503 --- up201604503@fe.up.pt

**Supervisor:** Filipe Alexandre Pais de Figueiredo Correia

November, 2018

# Índice

<b>1. ElasticSearch – The Open Source Project</b>	
1.1. How alive is ElasticSearch?.....	3
1.2. How important is ElasticSearch?.....	4
1.3. What is ElasticSearch good for?.....	4
1.4. What are the technologies involved?.....	5
<b>2. ElasticSearch Issues.....</b>	<b>6</b>
2.1. Issue #34127.....	7
2.1.1. Description of the issue.....	7
2.1.2. Requirements.....	7
2.1.3. Source Code Files.....	7
2.1.4. System Architecture.....	8
2.1.5. Design of the fix.....	9
2.1.6. Fix Source Code.....	10
2.1.5. Validate the fix.....	13
2.1.6. Submission of the fix.....	13
2.1.6. Wrap-up.....	14
2.2. Issue #34609.....	14
2.2.1. Description of the issue.....	14
2.2.2. Requirements.....	14
2.2.3. Source Code Files.....	15
2.2.4. System Architecture.....	16
2.2.5. Design of the fix.....	16
2.2.6. Fix Source Code.....	17
2.2.7. Validate the fix.....	19
2.2.8. Submission of the fix.....	19
2.2.9. Wrap-up.....	19

# 1. Elasticsearch - The Open Source Project

## 1.1. How alive is Elasticsearch?

**ElasticSearch** was born in 2004 and since then it has been growing, specially the community that works on it. This community is very active. There are **1,103 contributors** and **1,508 open issues**. Until now, there are already **16.138 closed issues**, confirming that ElasticSearch has been one of the most interesting projects to work on.

The search engine versions have been changing since ElasticSearch was created. It has **different versions**, with **version 6.x** being in **active development** with new features, and **version 5.x is still under support**. Older versions are inactive for around one year.

The **version 5.x is feature-freeze** and it's **only getting bug fixes** at the moment. Although the development for this version is not very active, the latest release, 5.6.12, was released at 19th September, which is recent. Moreover, some work on the next minor version, 5.6.13, is ongoing.

The development of the current public release, 6.4.x, is very active with almost daily commits. The next planned release, 6.5, is even more active, which means the **project contributors** are **actively working to improve the product**, either by tweaking existing functionality or by adding new features.

Overall, there are **over 100 commits weekly** on average.

Although the list of collaborators (core development team) is not known for this particular repository, **the Elastic organization has 129 people**. They have several repositories, thus this number is not a sufficient indicator for ElasticSearch in particular. However, considering the number of merged pull request and closed issues, **we can say there are enough people managing the project**.

Anyone interested in giving their contribution can easily do it. The project has a guideline for **contribution** with project's information and the steps to follow.

Summarizing, ElasticSearch is really alive and its life is sustained thanks to the numerous contributors who keep working on this amazing project, so we can expect an amazing lifetime for it.

## 1.2. How importante is ElasticSearch?

Elastic search is a highly important tool for most companies, as it is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases.

It is particularly relevant because it lets you perform and combine many types of searches — structured, unstructured, geo, metric — any way you want, allowing you to query data in various ways to obtain several important analytics.

For example, with this software you could search for all posts (social media and the likes) containing mentions of your product, then you could take those results and aggregate them by geography; you could then narrow it down to a specific country, even sort things out by date or break it down by the three main states of that region.

Thus, it ends up playing a somewhat vital role in things like datamining, data analysis and many subjects in that general area.

## 1.3. What is ElasticSearch good for?

Elasticsearch can be used to perform a lot of different use cases: "classical" full text search, analytics store, auto completer, spell checker, alerting engine, and as a general purpose document store.

It is especially good for analytics, because it integrates well with the ELK stack, and its interface is good for the REST/JSON driven configuration.

It provides a scalable search, since it scales horizontally to handle many events per second while automatically managing how indices and queries are distributed across many operations.

It also has a near real-time search, as a result of the implementation of inverted indices with finite state transducers for full-text querying, BKD trees for storing numeric and geo data and a column store for analytics.

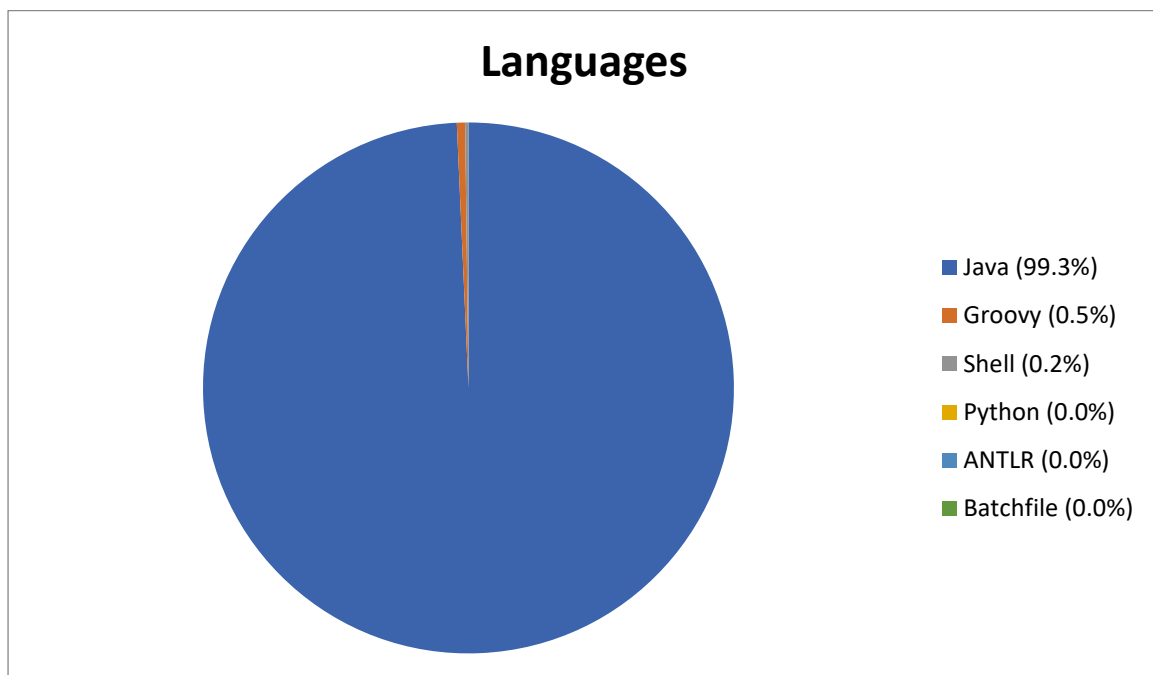
Thus, it's good for those seeking an engine that's effective at quickly and dynamically searching through large amounts of data, as is the case for a search-intensive application.

## 1.4. What are the technologies involved?

**Type:** Software Service. Interaction with a REST API or Java API, with multiple Clients available: Java, Groovy, .Net, PHP, Perl, Python, Ruby.

### Languages:

- **Java** – Main language
- **Groovy** – Used in buildSrc. There's an issue open to replace Groovy with Java
- **AsciiDoc** – Documentation
- **YAML, JSON** – Mainly used on the REST API Specification



## **2. ElasticSearch Issues**

- **Issue #34127 – Include origin information in deprecation logging**
- **Issue #34609 – SQL: add tests for RLIKE**

## 2.1. Issue #34127 - <https://github.com/elastic/elasticsearch/issues/34127>

### 1. Description of the Issue

The Elasticsearch's issue #34127 is the result of the lack of information on the deprecation logging. What happens is that the deprecation logging only logs the timestamp and the message of deprecation. The suggestion to correct it is to add more information to the DeprecationLogger of the project.

### 2. Requirements

Aiming to obtain the solution for this issue, it was required to add more information on the deprecation message, like the hostname or IP address. We also had to know the basics of Java programming.

To solve this issue we improved the deprecation message by adding some useful information besides the information that is given.

An example of a deprecation message is:

```
[WARN ][o.e.d.r.RestController] Content type detection for rest requests is deprecated. Specify the content type using the [Content-Type] header.
[WARN ][o.e.d.c.ParseField]  Deprecated field [inline] used, expected [source] instead
[WARN ][o.e.d.c.ParseField]  Deprecated field [valueType] used, expected [value_type] instead
[WARN ][o.e.d.r.a.RestFieldStatsAction] [_field_stats] endpoint is deprecated! Use [_field_caps] instead or run a min/max aggregations on the desired fields.
[WARN ][o.e.d.c.ParseField]  Deprecated field [type] used, replaced by [match_phrase and match_phrase_prefix query]
```

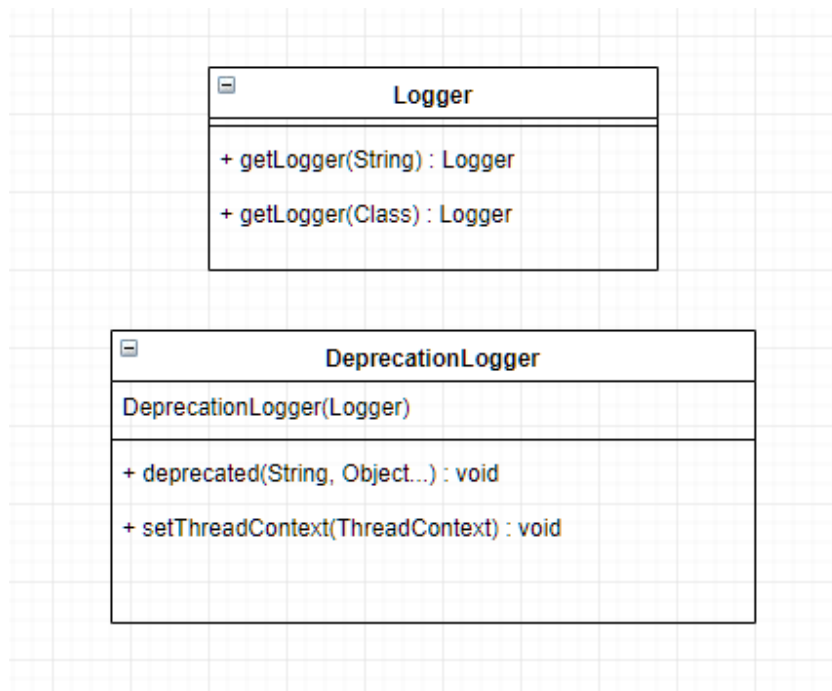
### 3. Source code files

The files that are directly related to #34127 issue are:

- **elasticsearch/server/src/main/java/org/elasticsearch/common/logging/DeprecationLogger.java** – File of the Deprecation Logger where the issue was.
- **elasticsearch/server/src/test/java/org/elasticsearch/common/logging/DeprecationLoggerTests.java** – File where the Deprecation Logger is tested.
- **elasticsearch/server/src/main/java/org/elasticsearch/transport/TransportService.java** – File that has the responsibility to send the port to the Deprecation Logger

## 4. System Architecture

After a lot of research on the open source project ElasticSearch, we achieved some knowledge of how this project works. Specifically, in this issue, the architecture has the following UML:



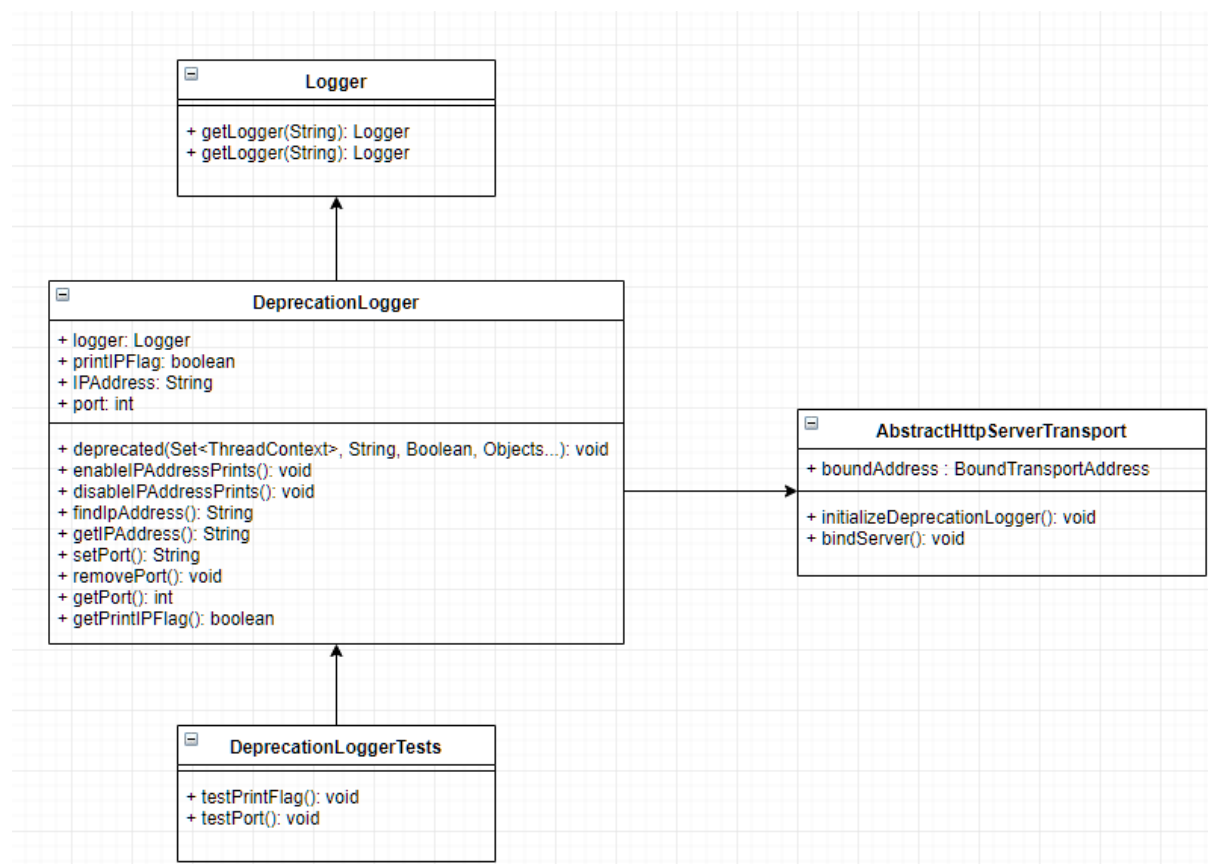
The **DeprecationLogger** allows various Elasticsearch classes to warn the user when they use a method that is going to be removed from the project soon. The **Logger** class allows other classes to easily log actions to the user when deemed necessary, without having to deal with things like identifying the class calling the logger and allows for different types of logging(Error, Information Warning).



## 5. Design of the fix

The solution was to add more information on the deprecation message, like the hostname or IP address. To add this information about the user we used the `NetworkInterface` and the `InetAddress` class of the `java.net` library.

In practice, the fix that we implemented consists to show this information on the Deprecation Logger.



For example, this deprecation message changes from this:

```
[2018-12-06T19:16:07,470][WARN ][o.e.d.s.StoredScriptSource] [node-0] empty scripts should no longer be used
```

To this:

```
[2018-12-06T19:09:03,116][WARN ][o.e.d.s.StoredScriptSource] [node-0] {192.168.1.120:9300} empty scripts should no longer be used
```

## 6. Fix Source Code

The code alterations we implemented are the following, ordered by files:

- server/src/main/java/org/elasticsearch/common/logging/DeprecationLogger.java

1. First we created some new variables

```
private boolean printIPFlag = false;
private static String IPAddress = null;
private static int port = -1;
```

2. Then we changed this function

```
void deprecated(final Set<ThreadContext> threadContexts, final String message, final boolean log, final Object... params) {
    final Iterator<ThreadContext> iterator = threadContexts.iterator();

    if (iterator.hasNext()) {
        final String formattedMessage = LoggerMessageFormat.format(message, params);
        final String warningHeaderValue = formatWarning(formattedMessage);
        assert WARNING_HEADER_PATTERN.matcher(warningHeaderValue).matches();
        assert extractWarningValueFromWarningHeader(warningHeaderValue).equals(escapeAndEncode(formattedMessage));
        while (iterator.hasNext()) {
            try {
                final ThreadContext next = iterator.next();
                next.addResponseHeader("Warning", warningHeaderValue, DeprecationLogger::extractWarningValueFromWarningHeader);
            } catch (final IllegalStateException e) {
                // ignored; it should be removed shortly
            }
        }
    }

    if (log) {
        logger.warn(message, params);
    }
}
```

Antes

```
void deprecated(final Set<ThreadContext> threadContexts, final String message, final boolean log, final Object... params) {
    final Iterator<ThreadContext> iterator = threadContexts.iterator();

    String newMessage;
    if (printIPFlag)
    {
        if (port != -1)
            newMessage = "{" + IPAddress + ":" + port + "}" + message;
        else
            newMessage = "{" + IPAddress + "}" + message;
    }
    else
        newMessage = message;

    if (iterator.hasNext()) {
        final String formattedMessage = LoggerMessageFormat.format(message, params);
        final String warningHeaderValue = formatWarning(formattedMessage);
        assert WARNING_HEADER_PATTERN.matcher(warningHeaderValue).matches();
        assert extractWarningValueFromWarningHeader(warningHeaderValue).equals(escapeAndEncode(formattedMessage));
        while (iterator.hasNext()) {
            try {
                final ThreadContext next = iterator.next();
                next.addResponseHeader("Warning", warningHeaderValue, DeprecationLogger::extractWarningValueFromWarningHeader);
            } catch (final IllegalStateException e) {
                // ignored; it should be removed shortly
            }
        }
    }

    if (log) {
        logger.warn(newMessage, params);
    }
}
```

Depois

3. Then we added some functions: the first two are relatable to prints, the first enables them and the second disables them. The third function finds the user IP address and the four is its get function. The functions **setPort()**, **getPort()** and **removePort()** are related to the port. The last one, **getPrintIPFlag()** returns the Boolean value of the variable we created on step 1.

```
public void enableIPAddressPrints() {
    printIPFlag = true;
    IPAddress = findIpAddress();
}

public void disableIPAddressPrints() {
    printIPFlag = false;
}

private String findIpAddress() {
    try
    {
        Enumeration e;
        e = NetworkInterface.getNetworkInterfaces();
        while (e.hasMoreElements())
        {
            NetworkInterface n = (NetworkInterface) e.nextElement();
            Enumeration ee = n.getInetAddresses();
            while (ee.hasMoreElements())
            {
                InetAddress i = (InetAddress) ee.nextElement();
                if (i.isSiteLocalAddress()) {
                    return i.getHostAddress();
                }
            }
        }
    }
    catch (SocketException e1)
    {
        e1.printStackTrace();
    }

    return null;
}

public static String getIPAddress(){
    return IPAddress;
}

public static void setPort(int newPort) {
    port = newPort;
}

public static int getPort(){
    return port;
}

public static void removePort(){
    port = -1;
}
```

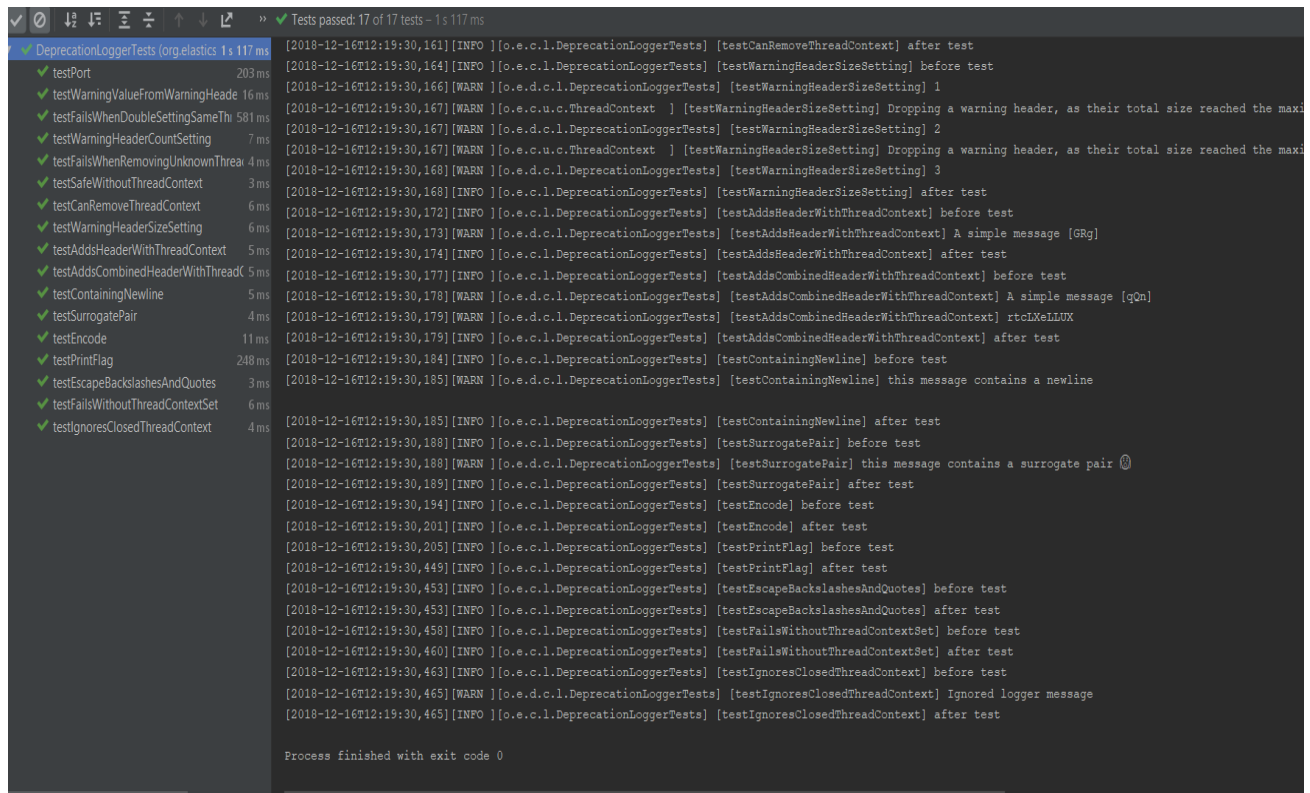
- server/src/main/java/org/elasticsearch/http/AbstractHttpServerTransport.java

1. In this file we created a function that initializes the Deprecation Logger

```
private void initializeDeprecationLogger()
{
    DeprecationLogger.setPort(this.boundAddress.publishAddress().getPort());
}
```

2. After that, we changed the function **bindServer()**, which has the purpose to bind and start to accept incoming connections, adding the function we created in step 1. of this file.

The following print screen shows that our implemented fix works.



## 7. Validate the fix

To test our changes, we created the following tests on the file `server/src/test/java/org/elasticsearch/common/logging/DeprecationLoggerTests.java`

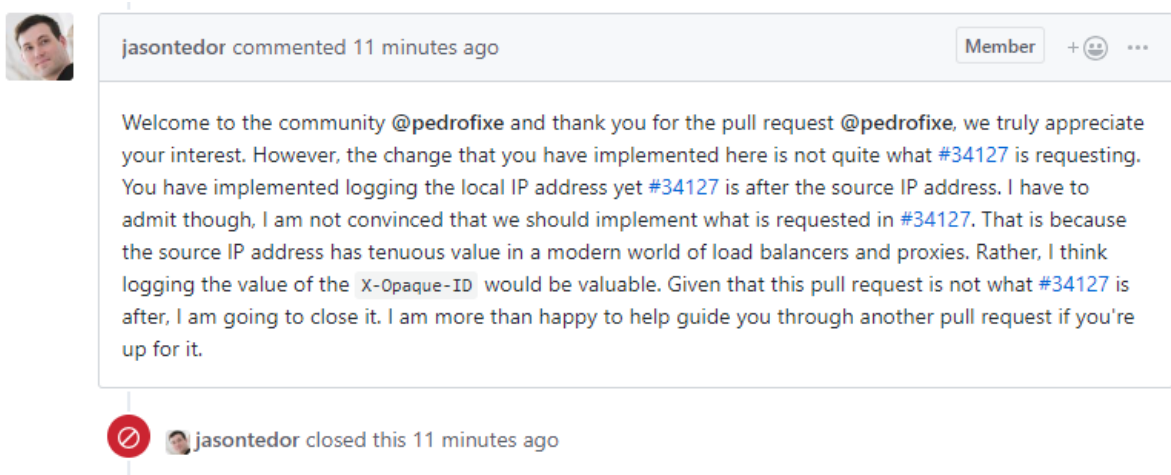
```
public void testPort(){
    assertEquals(logger.getPort(), -1);
    logger.setPort(1234);
    assertEquals(logger.getPort(), 1234);
    logger.removePort();
    assertEquals(logger.getPort(), -1);
}

public void testIP(){
    assertEquals(logger.getIPAddress(), null);
    logger.enableIPAddressPrints();
    assertNotNull(logger.getIPAddress());
    logger.disableIPAddressPrints();
    assertEquals(logger.getIPAddress(), null);
}
```

These tests check validate the initial state of the PrintIPFlag and the Port and the IPAddress of the user.

## 8. Submission of the fix

We tested and submitted our fix of this issue by making a pull request. Unfortunately, it was turned down.



Here is the link to the pull request: <https://github.com/elastic/elasticsearch/pull/36687>

## 9. Wrap-up

To solve the issue we had to divide our work. We consider that the following is a fair work division of each student:

- João Pedro – 50%
- Tomás Novo – 35%
- Tiago Cardoso – 15%

### 2.1. Issue #34609 - <https://github.com/elastic/elasticsearch/issues/34609>

#### 1. Description of the Issue

The #34609 ElasticSearch's issue related to SQL language. In this issue, there is a class called **RLike** that has no tests for and it would be very positive if someone could make tests to ensure that everything runs perfectly in the meantime.

#### 2. Requirements

To solve this issue we had to know the basics of SQL and how tests work in Java.

This issue required the implementation of some tests for **RLike** class.

### 3. Source Code Files

Files directly related to the issue:

- **elasticsearch/x-pack/plugin/sql/src/main/java/org/elasticsearch/xpack/sql/expression/predicate/regex/RLike.java** – Where RLike is implemented

```
public class RLike extends RegexMatch {

    public RLike(Location location, Expression left, Literal right) {
        super(location, left, right);
    }

    @Override
    protected NodeInfo<RLike> info() {
        return NodeInfo.create(this, RLike::new, left(), (Literal) right());
    }

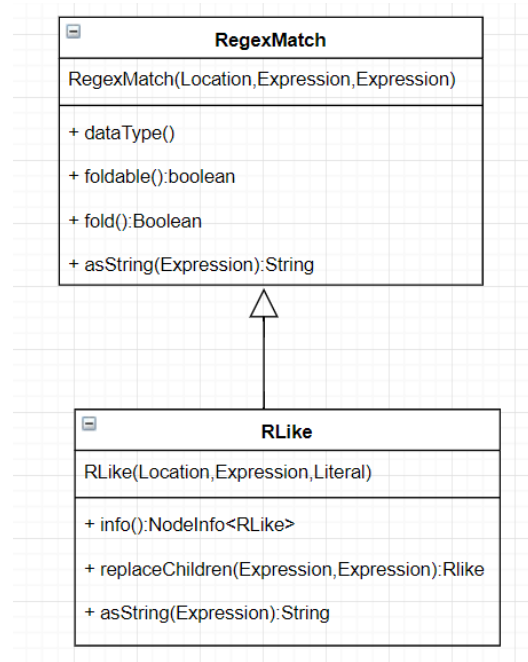
    @Override
    protected RLike replaceChildren(Expression newLeft, Expression newRight) {
        return new RLike(location(), newLeft, (Literal) newRight);
    }

    @Override
    protected String asString(Expression pattern) {
        return pattern.fold().toString();
    }
}
```

- **elasticsearch/x-pack/plugin/sql/src/test/java/org/elasticsearch/xpack/sql/expression/predicate/regex/RLikeTests.java** – New file where we implemented the tests

## 4. System Architecture

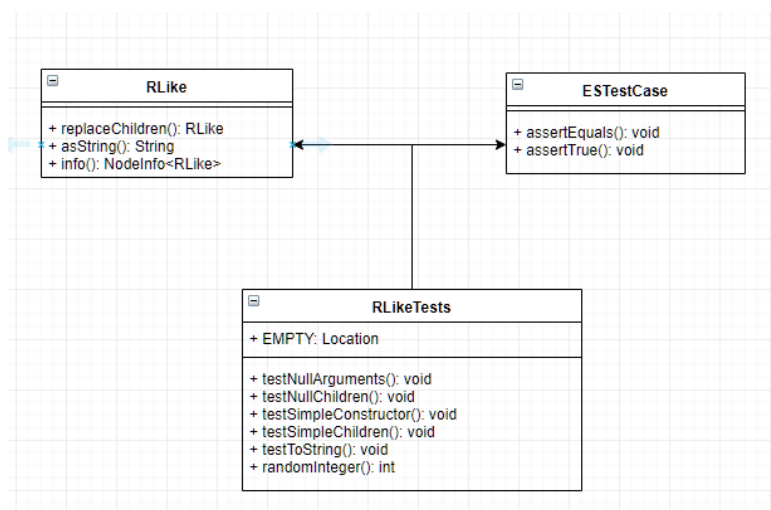
Researching on the open source project ElasticSearch gave us the knowledge of how this issue happens. This issue's architecture has the following UML:



The **RLike** class is the equivalent to **RLike** SQL operator that performs a pattern match of a string expression against a pattern.

## 5. Design of the fix

Our plan to fix this issue is to create some tests for the class **RLike**.



The **ESTestCase** is a generic class to implement tests in Elasticsearch.



## 6. Fix Source Code

With the objective to implement the tests for RLike class, we created the following code:

```
public class RLikeTests extends ETestCase {

    private static final Location EMPTY = new Location(-1, -2);

    public void testNullArguments() {

        boolean exceptionCaught = false;

        try
        {
            new RLike(null, null, null);
        }
        catch (SqlIllegalArgumentException exception)
        {
            exceptionCaught = true;
        }

        assertEquals(exceptionCaught, true);
    }
}
```

```

public void testNullChildren() {

    boolean exceptionCaught = false;

    RLike rlike = new RLike(EMPTY, Literal.of(EMPTY, 1), Literal.of(EMPTY, 1));

    try
    {
        rlike.replaceChildren(null, Literal.of(EMPTY, 1));
    }
    catch (SqlIllegalArgumentException exception)
    {
        exceptionCaught = true;
    }

    assertEquals(exceptionCaught, true);
}

public void testSimpleConstructor() {

    int n1 = randomInteger(), n2 = randomInteger();

    RLike rlike = new RLike(EMPTY, Literal.of(EMPTY, n1), Literal.of(EMPTY, n2));

    List<Expression> children = rlike.children();

    assertEquals(children.size(), 2);

    assertEquals(children.get(0).toString(), Integer.toString(n1));
    assertEquals(children.get(1).toString(), Integer.toString(n2));
}

public void testSimpleChildren() {

    int n1 = randomInteger(), n2 = randomInteger(), n3 = randomInteger(), n4 = randomInteger();

    RLike rlike = new RLike(EMPTY, Literal.of(EMPTY, n1), Literal.of(EMPTY, n2));

    rlike = rlike.replaceChildren(Literal.of(EMPTY, n3), Literal.of(EMPTY, n4));

    List<Expression> children = rlike.children();
    assertEquals(children.size(), 2);

    assertEquals(children.get(0).toString(), Integer.toString(n3));
    assertEquals(children.get(1).toString(), Integer.toString(n4));
}

public void testToString() {

    int n1 = randomInteger(), n2 = randomInteger();

    RLike rlike = new RLike(EMPTY, Literal.of(EMPTY, n1), Literal.of(EMPTY, n2));

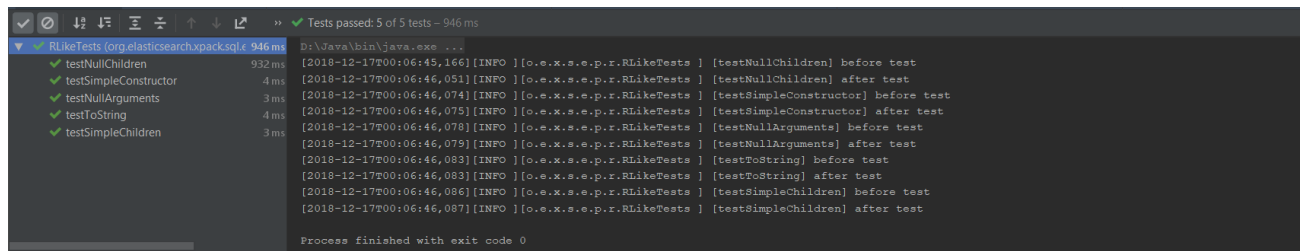
    String testString = rlike.toString().substring(0, rlike.toString().indexOf('#'));

    assertEquals(testString, n1 + " REGEX " + n2);
}

private static int randomInteger()
{
    return ThreadLocalRandom.current().nextInt(-10, 11);
}
}

```

The following print shows that our implemented fix works.



```
Tests passed: 5 of 5 tests - 946 ms
D:\Java\bin\java.exe ...
[2018-12-17T00:06:45,166] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testNullChildren] before test
[2018-12-17T00:06:46,051] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testNullChildren] after test
[2018-12-17T00:06:46,074] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testSimpleConstructor] before test
[2018-12-17T00:06:46,075] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testSimpleConstructor] after test
[2018-12-17T00:06:46,078] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testNullArguments] before test
[2018-12-17T00:06:46,079] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testNullArguments] after test
[2018-12-17T00:06:46,083] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testToString] before test
[2018-12-17T00:06:46,083] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testToString] after test
[2018-12-17T00:06:46,086] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testSimpleChildren] before test
[2018-12-17T00:06:46,087] [INFO] [o.e.x.s.e.p.r.ELikeTests] [testSimpleChildren] after test
Process finished with exit code 0
```

## 7. Validate the fix

The issue itself was to implement tests about this class. Therefore, we tested against null arguments and children, with simple arguments and children and with the toString function.

## 8. Submission of the fix

Like the first issue we worked on, we submitted the fix of this issue by making a pull request and we're currently waiting for answer. The pull request can be accessed by the following link: <https://github.com/elastic/elasticsearch/pull/36688>

## 9. Wrap-up

To solve the issue we had to divide our work. We consider that the following is a fair work division of each student:

- Tomás Novo – 50 %
- João Franco – 35 %
- Tiago Cardoso – 15 %