FREE eBook

LEARNING

solr

Free unaffiliated eBook created from
Stack Overflow contributors.

#solr

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: solr

It is an unofficial and free solr ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official solr.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with solr

Remarks

This section provides an overview of what solr is, and why a developer might want to use it.

It should also mention any large subjects within solr, and link out to the related topics. Since the Documentation for solr is new, you may need to create initial versions of those related topics.

Examples

Introduction

Solr is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via JSON, XML, CSV or binary over HTTP. You query it via HTTP GET and receive JSON, XML, CSV or binary results. Solr uses the Lucene search library and extends it.

Here are some of the main features that solr provides:

- Advanced Full-Text Search Capabilities

Powered by Lucene™, Solr enables powerful matching capabilities including phrases, wildcards, joins, grouping and much more across any data type

- **Optimized for High Volume Traffic**

Solr is proven at extremely large scales the world over

- **Standards Based Open Interfaces - XML, JSON and HTTP**

Solr uses the tools you use to make application building a snap

- **Comprehensive Administration Interfaces**

Solr ships with a built-in, responsive administrative user interface to make it easy to control your Solr instances

- **Easy Monitoring**

Need more insight into your instances? Solr publishes loads of metric data via JMX

- **Highly Scalable and Fault Tolerant**

Built on the battle-tested Apache Zookeeper, Solr makes it easy to scale up and down. Solr bakes in replication, distribution, rebalancing and fault tolerance out of the box.

- **Flexible and Adaptable with easy configuration**

Solr's is designed to adapt to your needs all while simplifying configuration

- **Near Real-Time Indexing**

Want to see your updates now? Solr takes advantage of Lucene's Near Real-Time Indexing capabilities to make sure you see your content when you want to see it

- **Extensible Plugin Architecture**

Solr publishes many well-defined extension points that make it easy to plugin both index and query time plugins. Of course, since it is Apache-licensed open source, you can change any code you want!

Some solr cool features:

- **Schema when you want, schemaless when you don't**

Use Solr's data-driven schemaless mode when getting started and then lock it down when it's time for production.

- **Powerful Extensions**

Solr ships with optional plugins for indexing rich content (e.g. PDFs, Word), language detection, search results clustering and more

- **Faceted Search and Filtering**

Slice and dice your data as you see fit using a large array of faceting algorithms

- **Geospatial Search**

Enabling location-based search is simple with Solr's built-in support for spatial search

- **Query Suggestions, Spelling and More**

Solr ships with advanced capabilites for auto-complete (typeahead search), spell checking and more

- **Rich Document Parsing**

Solr ships with Apache Tika built-in, making it easy to index rich content such as Adobe PDF, Microsoft Word and more.

Installing SOLR

Introduction

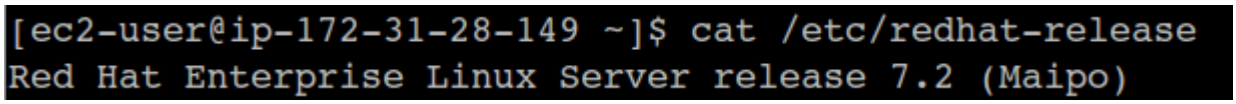
The following procedure was tested on a test instance in AWS, with Redhat and `Solr 6.1.0`. You may need to adjust the process to your operating system and environment accordingly.

Prerequisites

1. Make sure you use RedHat or a similar (Fedora-based) OS.

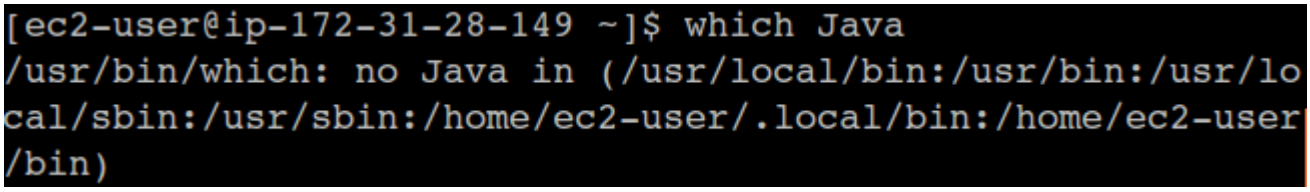
   `cat /etc/redhat-release`

   displays your OS version.

   
   ```
   [ec2-user@ip-172-31-28-149 ~]$ cat /etc/redhat-release
   Red Hat Enterprise Linux Server release 7.2 (Maipo)
   ```

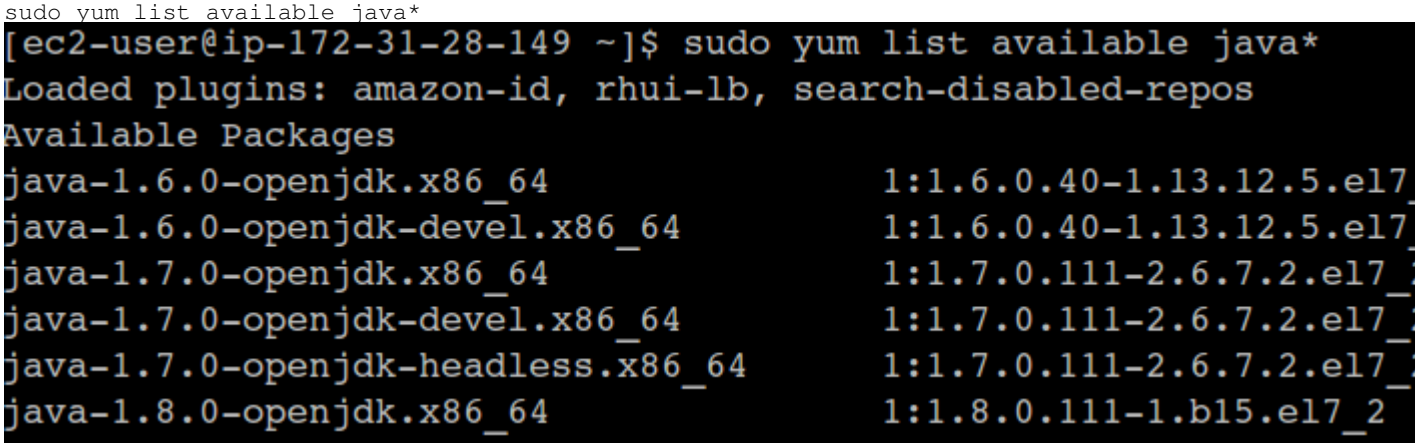
2. Check if java 1.6 or higher is installed

   `which java`

   
   ```
   [ec2-user@ip-172-31-28-149 ~]$ which Java
   /usr/bin/which: no Java in (/usr/local/bin:/usr/bin:/usr/lo
   cal/sbin:/usr/sbin:/home/ec2-user/.local/bin:/home/ec2-user
   /bin)
   ```

3. Install Java if necessary

   `sudo yum list available java*`

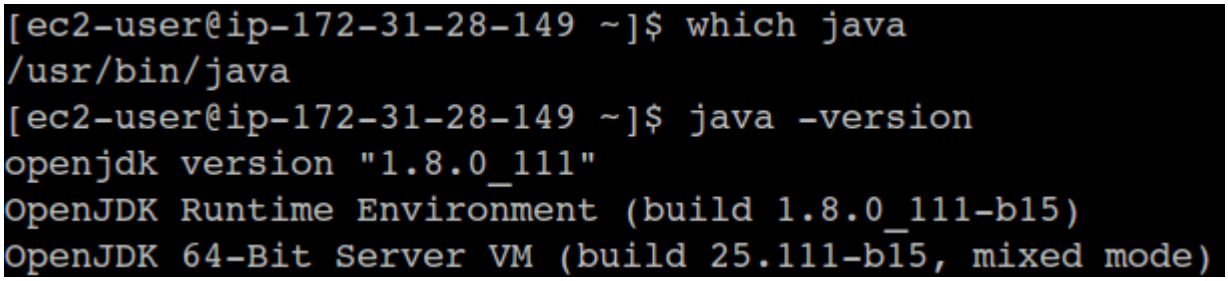
   
   ```
   [ec2-user@ip-172-31-28-149 ~]$ sudo yum list available java*
   Loaded plugins: amazon-id, rhui-lb, search-disabled-repos
   Available Packages
   java-1.6.0-openjdk.x86_64                1:1.6.0.40-1.13.12.5.el7
   java-1.6.0-openjdk-devel.x86_64          1:1.6.0.40-1.13.12.5.el7
   java-1.7.0-openjdk.x86_64                1:1.7.0.111-2.6.7.2.el7_
   java-1.7.0-openjdk-devel.x86_64          1:1.7.0.111-2.6.7.2.el7_
   java-1.7.0-openjdk-headless.x86_64       1:1.7.0.111-2.6.7.2.el7_
   java-1.8.0-openjdk.x86_64                1:1.8.0.111-1.b15.el7_2
   ```

   `sudo yum install java-1.8.0-openjdk.x86_64`

4. Check if it is installed correctly

   `which java` displays the Java home

   `java -version` displays the Java version

   
   ```
   [ec2-user@ip-172-31-28-149 ~]$ which java
   /usr/bin/java
   [ec2-user@ip-172-31-28-149 ~]$ java -version
   openjdk version "1.8.0_111"
   OpenJDK Runtime Environment (build 1.8.0_111-b15)
   OpenJDK 64-Bit Server VM (build 25.111-b15, mixed mode)
   ```

5. Create a SOLR user.

   `sudo adduser solr`

6. Add a password for the user.

   `sudo passwd solr`

```
[ec2-user@ip-172-31-28-149 ~]$ sudo adduser solr
[ec2-user@ip-172-31-28-149 ~]$ sudo passwd solr
Changing password for user solr.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

7. Enable sudo on the SOLR user, run visudo
   `sudo visudo`

8. Find the following lines:

```
## Allows people in group wheel to run all commands
# %wheel ALL=(ALL) ALL
```

9. If `%wheel` is commented out, uncomment the second line by removing the # character.
   `%wheel ALL=(ALL) ALL`

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

10. If you made a change use `:wq` otherwise use `:q` to quit.

11. Add the solr user to the wheel group.
    `sudo usermod -aG wheel solr`

12. Switch over to the `solr` user an check if you have root privileges:
    `su solr -`
    `sudo whoami`

```
[ec2-user@ip-172-31-28-149 ~]$ sudo usermod -aG wheel solr
[ec2-user@ip-172-31-28-149 ~]$ su solr
Password:
[solr@ip-172-31-28-149 ec2-user]$ groups
solr wheel
[solr@ip-172-31-28-149 ec2-user]$ sudo whoami

We trust you have received the usual lecture from the local
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for solr:
root
```

Downloading SOLR

13. Find your local mirror at: `http://www.apache.org/dyn/closer.lua/lucene/solr/`

14. Change your directory location to opt:
    ```
    cd /opt/
    ```

15. Download a copy of the package from the mirror:
    ```
    sudo curl -O http://www.trieuvan.com/apache/lucene/solr/6.1.0/solr-6.1.0.tgz
    ```
    ```
    [solr@ip-172-31-28-149 opt]$ sudo curl -O http://www.trieuvan.co
    m/apache/lucene/solr/6.2.0/solr-6.2.0.tgz
      % Total    % Received % Xferd  Average Speed   Time    Time
      Time  Current
                                     Dload  Upload   Total   Spent
      Left  Speed
      0     0    0     0    0     0      0      0 --:--:-- --:--:--
      3  142M    3 5343k    0     0  5501k      0  0:00:26 --:--:--
      9  142M    9 13.5M    0     0  7041k      0  0:00:20  0:00:01
     13  142M   13 19.3M    0     0  6656k      0  0:00:21  0:00:02
     16  142M   16 23.4M    0     0  6028k      0  0:00:24  0:00:03
     19  142M   19 27.4M    0     0  5656k      0  0:00:25  0:00:04
    ```

16. Untar the package:
    ```
    sudo tar zxvf solr-6.1.0.tgz
    ```

17. Copy the installer script to your folder:
    ```
    sudo cp /opt/solr-6.1.0/bin/install_solr_service.sh .
    ```

18. Remove the unnecessary files:
    ```
    sudo rm -rf solr-6.1.0
    ```

Install

1. Run the install script:
   ```
   sudo ./install_solr_service.sh solr-6.1.0.tgz
   ```

```
Started Solr server on port 8983 (pid=9607). Happy searching!


Found 1 Solr nodes:

Solr process 9607 running on port 8983
{
  "solr_home":"/var/solr/data",
  "version":"6.2.0 764d0f19151dbff6f5fcd9fc4b2682cf934590c5 - mi
ke - 2016-08-20 05:41:37",
  "startTime":"2016-10-20T04:02:37.028Z",
  "uptime":"0 days, 0 hours, 0 minutes, 15 seconds",
  "memory":"85.4 MB (%17.4) of 490.7 MB"}

Service solr installed.
```

2. Make SOLR service autostart when the server is rebooted.

```
sudo chkconfig --add solr
chkconfig | grep solr
```

```
[solr@ip-172-31-28-149 ec2-user]$ sudo chkconfig --add solr
[sudo] password for solr:
[solr@ip-172-31-28-149 ec2-user]$ chkconfig |grep solr

Note: This output shows SysV services only and does not inc
lude native
      systemd services. SysV configuration data might be ov
erridden by native
      systemd configuration.

      If you want to list systemd services use 'systemctl l
ist-unit-files'.
      To see services enabled on particular target use
      'systemctl list-dependencies [target]'.

solr            0:off   1:off   2:on    3:on    4:on    5:o
n       6:off
```

3. Change service owner

```
sudo chown -R solr:solr /var/solr/
```

Testing your installation

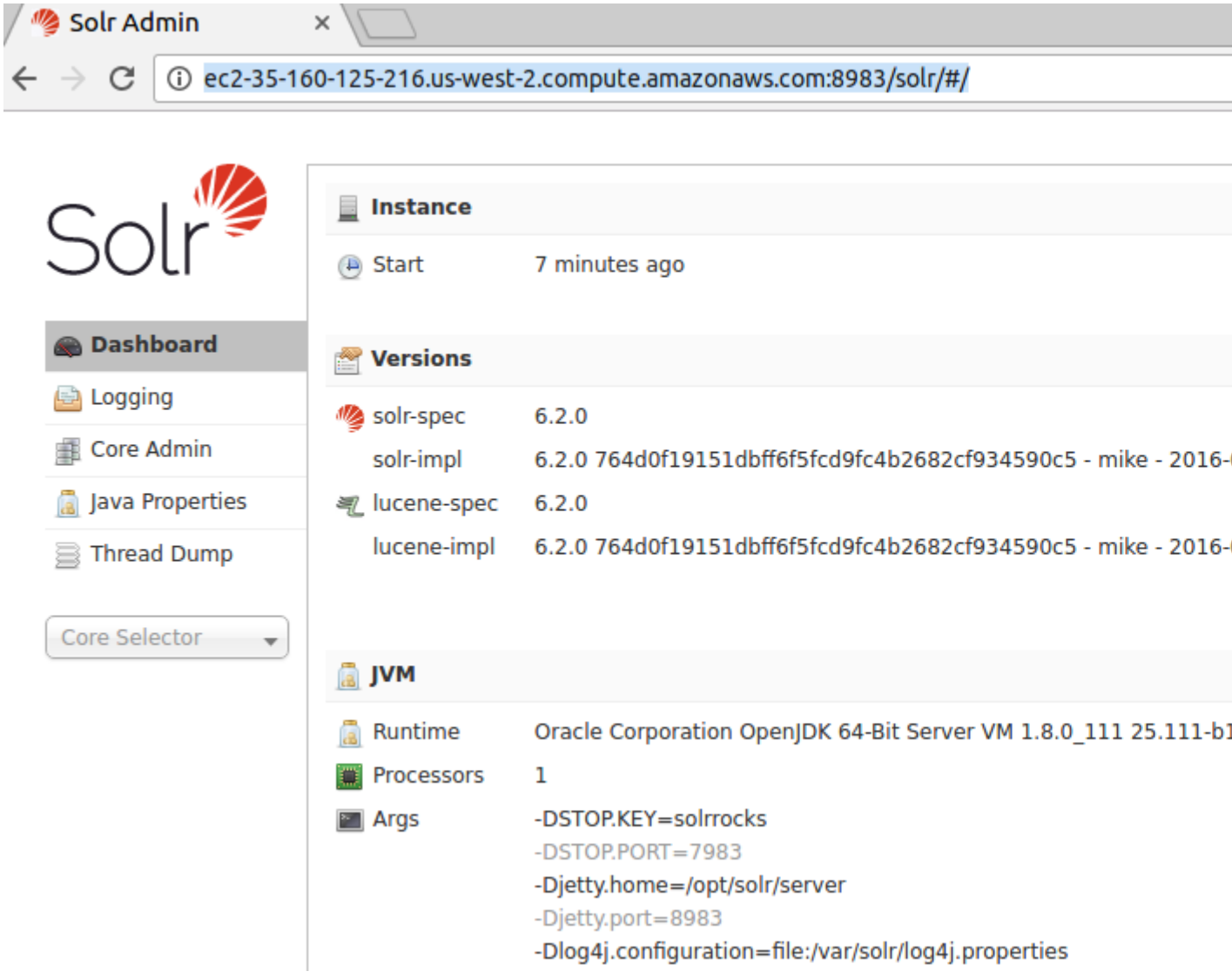
1. Create a core from command line:

```
sudo su - solr -c "/opt/solr/bin/solr create -c NewCore1 -n data_driven_schema_configs"
```

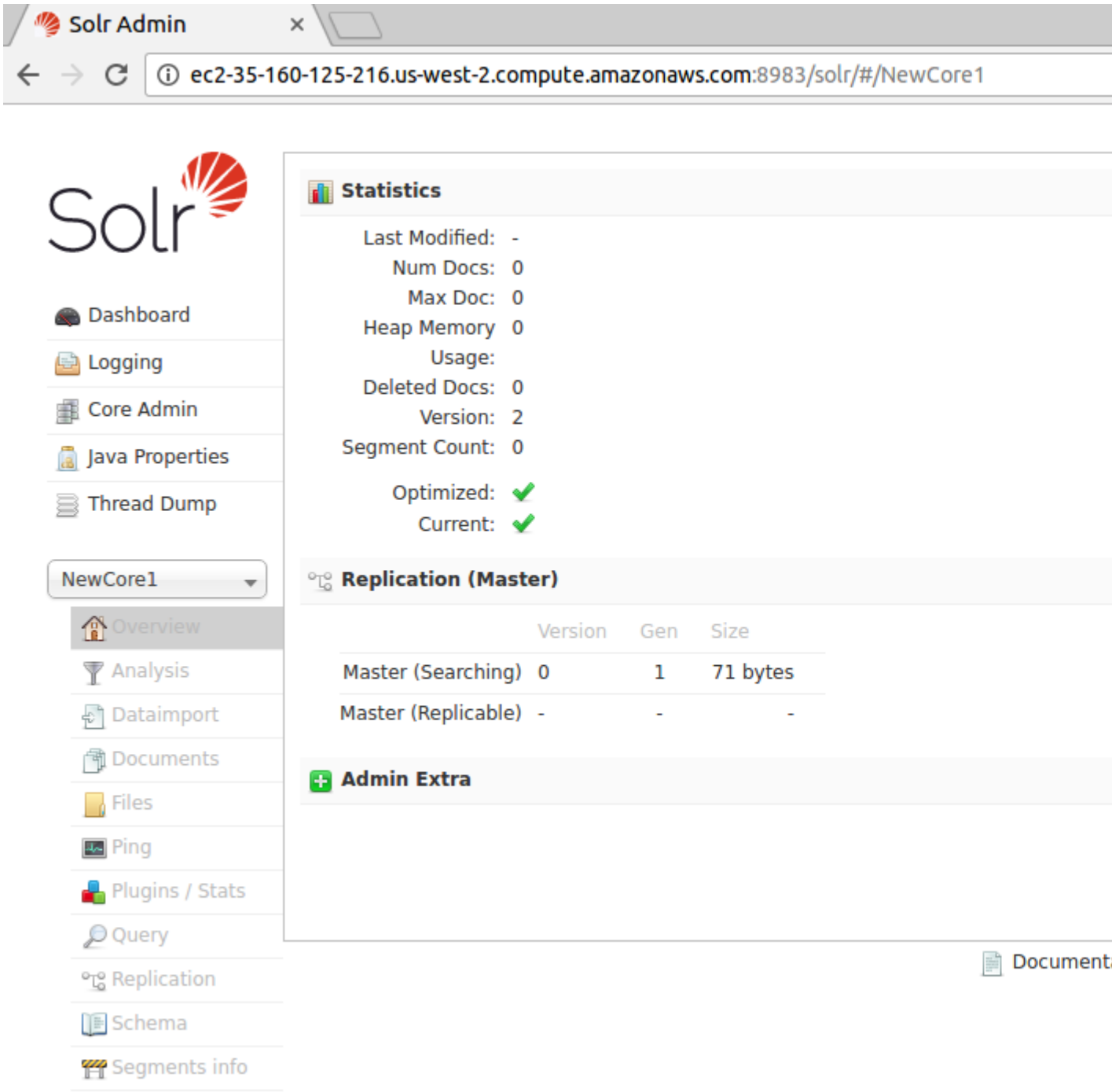
```
[solr@ip-172-31-28-149 ec2-user]$ sudo su - solr -c "/opt/s
olr/bin/solr create -c NewCore1 -n data_driven_schem_config
s"

Copying configuration to new core instance directory:
/var/solr/data/NewCore1

Creating new core 'NewCore1' using command:
http://localhost:8983/solr/admin/cores?action=CREATE&name=N
ewCore1&instanceDir=NewCore1

{
  "responseHeader":{
    "status":0,
    "QTime":2856},
  "core":"NewCore1"}
```

2. Open the Admin in a browser:

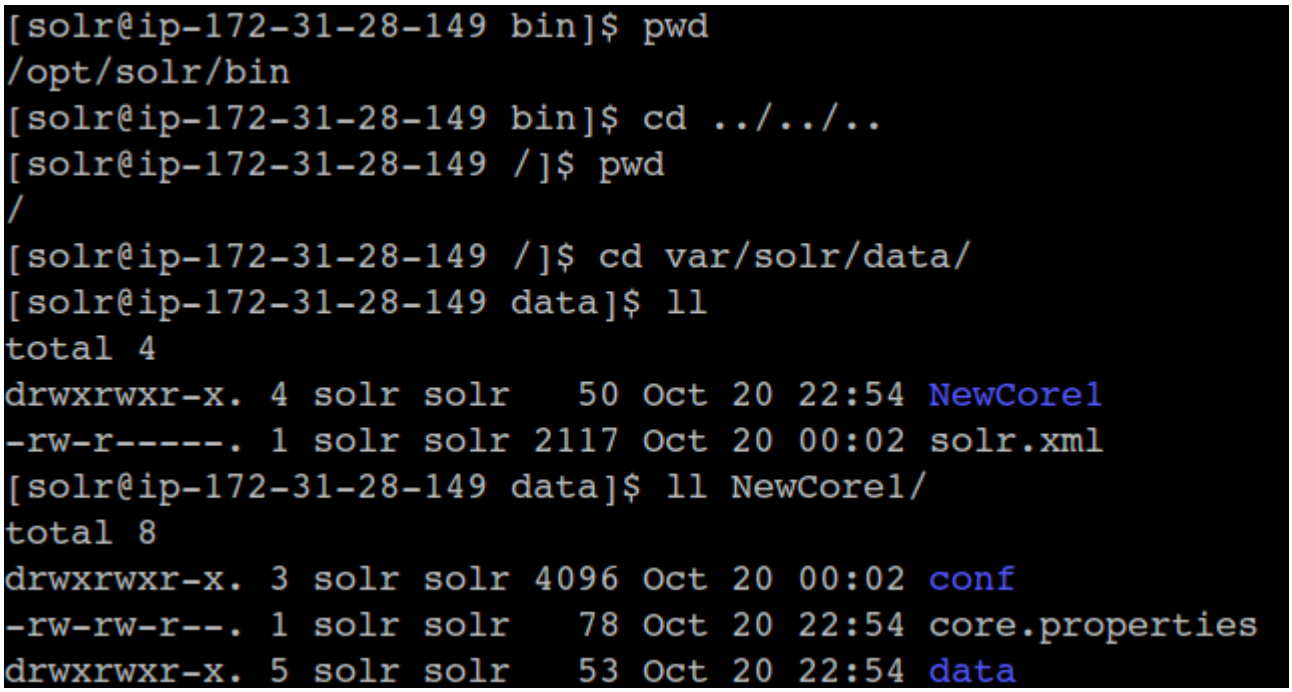
`http://<solr_server>:8983/solr/#/`



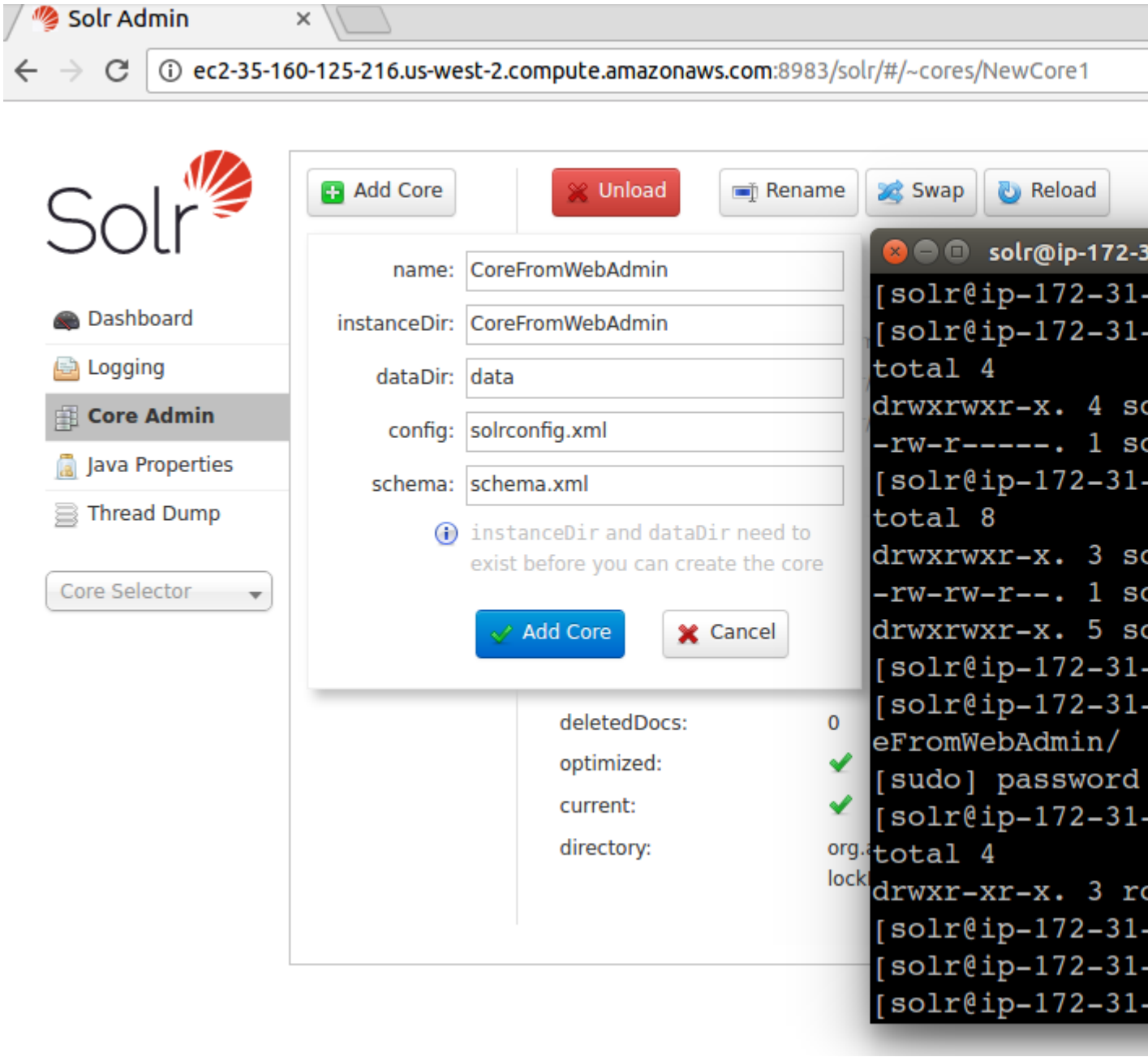
3. Open the list of cores in the menu to see the NewCore1 core.



4. Test if cores are sticky:
   `sudo service solr restart`
5. Refresh the Admin in a browser:
   `http://<solr_server>:8983/solr/#/` Make sure the Admin page reloads and the core reappears after the reboot.
6. View server status in the command line:
   `sudo service solr status`
7. Prepare a new core config for core creation in the Web Admin, by changing your directory location to data:
   `cd var/solr/data/`



8. This is where the new cores are stored:
   `ll`
9. The newly created core's conf folder can be used as a template:
   `ll NewCore1/`
10. Create a folder for another core you will create in the Web Admin:
    `mkdir CoreFromWebAdmin`
11. Copy the conf directory over to the new location:
    `sudo cp -R NewCore1/conf/ CoreFromWebAdmin`
12. Switch to the Web Admin interface in your browser
13. Click Add Core
14. Add CoreFromWebAdmin as the name and the folder for the new core.

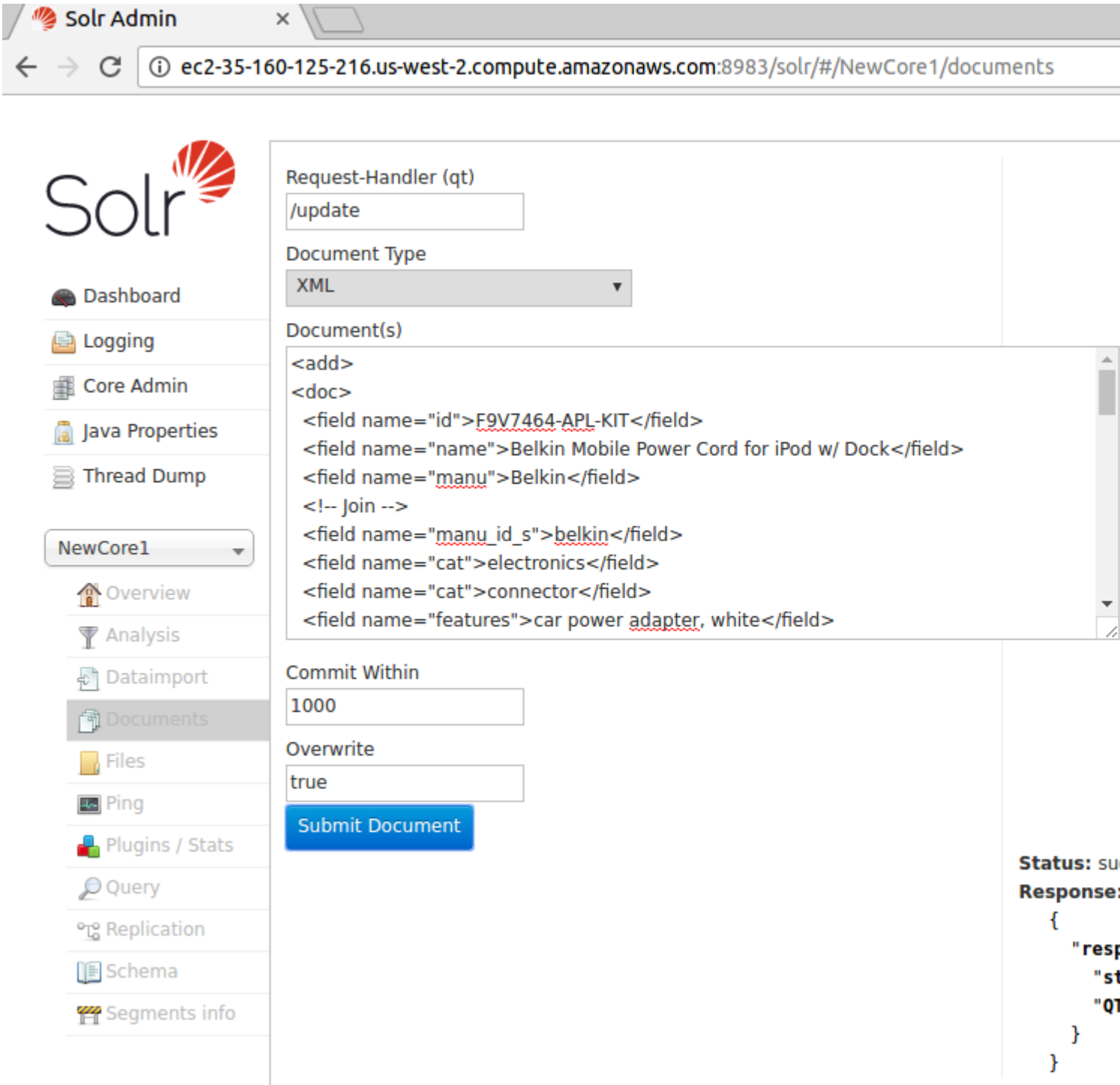


15. Open the new core.
16. Click documents to add docs.
17. Select XML format and paste the code below:

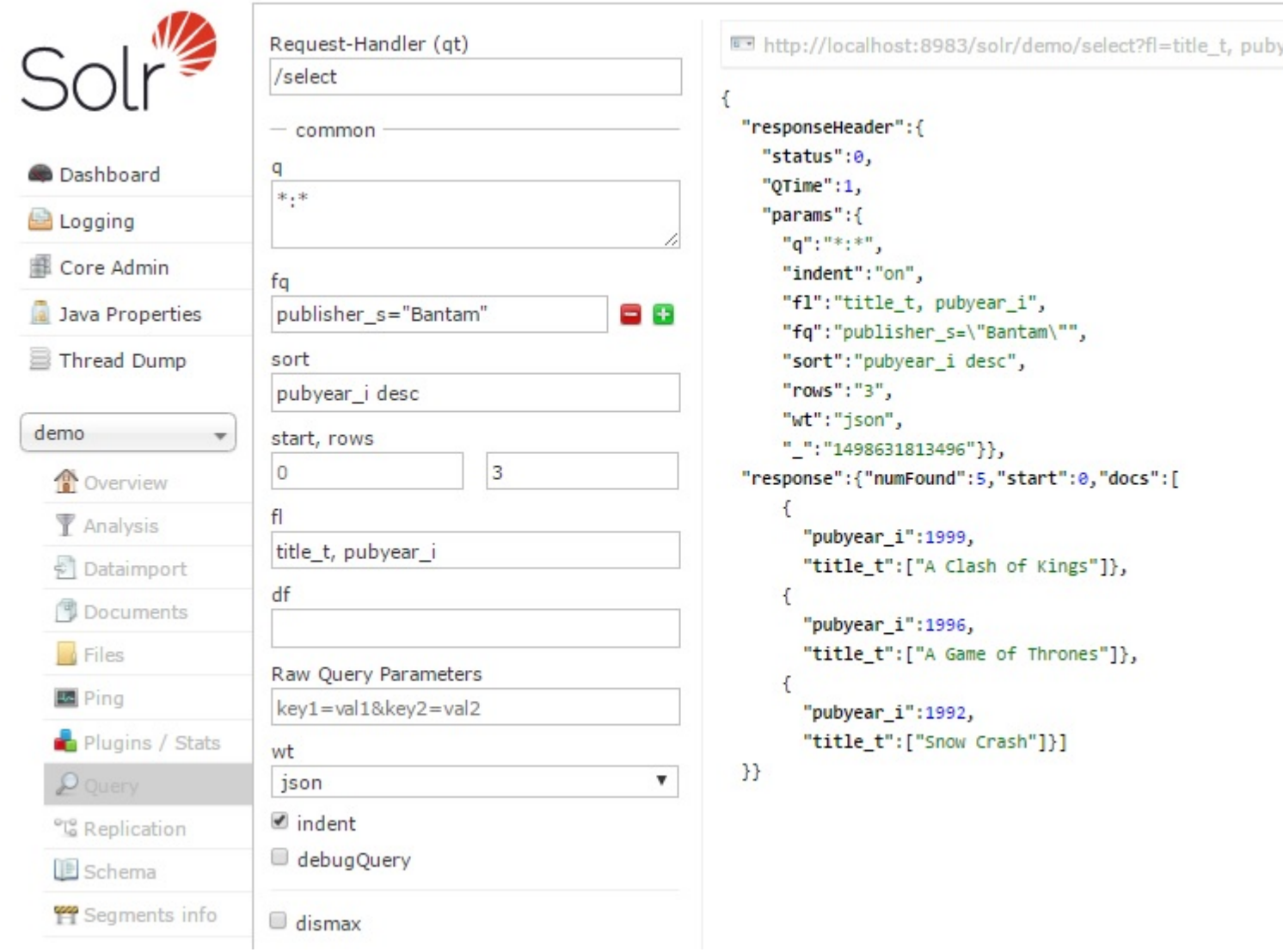
```
<add><doc>
  <field name="id">F9V7464-APL-KIT</field>
  <field name="name">Belkin Mobile Power Cord for iPod w/ Dock</field>
  <field name="manu">Belkin</field>
  <!-- Join -->
  <field name="manu_id_s">belkin</field>
  <field name="cat">electronics</field>
  <field name="cat">connector</field>
  <field name="features">car power adapter, white</field>
  <field name="weight">4.0</field>
  <field name="price">19.95</field>
  <field name="popularity">1</field>
  <field name="inStock">false</field>
  <!-- Buffalo store -->
  <field name="store">45.18014,-93.87741</field>
  <field name="manufacturedate_dt">2005-08-01T16:30:25Z</field>
```

```
</doc>

<doc>
  <field name="id">IW-032</field>
  <field name="name">iPod &amp;amp; iPod Mini USB 2.0 Cable</field>
  <field name="manu">Belkin</field>
  <!-- Join -->
  <field name="manu_id_s">belkin</field>
  <field name="cat">electronics</field>
  <field name="cat">connector</field>
  <field name="features">car power adapter for iPod, white</field>
  <field name="weight">2.0</field>
  <field name="price">11.50</field>
  <field name="popularity">1</field>
  <field name="inStock">false</field>
  <!-- San Francisco store -->
  <field name="store">37.7752,-122.4232</field>
  <field name="manufacturedate_dt">2006-02-14T23:55:59Z</field>
</doc>
<doc>
  <field name="id">F887464-APL-KIT</field>
  <field name="name">Belkin Mobile Power Cord for iPod w/ Dock</field>
  <field name="manu">Belkin</field>
  <!-- Join -->
  <field name="manu_id_s">belkin</field>
  <field name="cat">electronics</field>
  <field name="cat">connector</field>
  <field name="features">car power adapter, black</field>
  <field name="weight">4.0</field>
  <field name="price">19.95</field>
  <field name="popularity">1</field>
  <field name="inStock">true</field>
  <!-- Buffalo store -->
  <field name="store">45.18014,-93.87741</field>

</doc>
<doc>
  <field name="id">FAV7464-APL-KIT</field>
  <field name="name">Belkin Mobile Power Cord for iPod w/ Dock</field>
  <field name="manu">Belkin</field>
  <!-- Join -->
  <field name="manu_id_s">belkin</field>
  <field name="cat">electronics</field>
  <field name="cat">connector</field>
  <field name="features">car power adapter, blue</field>
  <field name="weight">4.0</field>
  <field name="price">15.95</field>
  <field name="popularity">2</field>
  <field name="inStock">true</field>
  <!-- Buffalo store -->
  <field name="store">45.18014,-93.87741</field>
  <field name="manufacturedate_dt">2015-09-21T16:30:25Z</field>
</doc></add>
```

If your response returns a success, you have successfully installed SOLR and verified your installation.



Example of Solr Search



Read Getting started with solr online: https://riptutorial.com/solr/topic/1015/getting-started-with-solr

Chapter 2: Apache Solr

Examples

Solr installation

You can install Solr in any system where a suitable Java Runtime Environment (JRE) is available, as detailed below. Currently this includes Linux, OS X, and Microsoft Windows. You will need the Java Runtime Environment (JRE) version 1.8 or higher. At a command line, check your Java version like this:

$ java -version

Solr is available from the Solr website at http://lucene.apache.org/solr/. Extract the Solr distribution archive to your local directory

Solr quick start (start and stop solar)

You can start Solr by running bin/solr from the Solr directory and this will start Solr in the background, listening on port 8983.

$ bin/solr start

To change the port Solr listens on, you can use the -p parameter when starting, such as:

$ bin/solr start -p 8984

Since Solr is a server, it is more common to run it in the background, especially on Unix/Linux. However, to start Solr in the foreground, simply do:

$ bin/solr start -f

When running Solr in the foreground (using -f), then you can stop it using Ctrl-c. However, when running in the background, you should use the stop command, such as:

$ bin/solr stop -p 8983

The stop command requires you to specify the port Solr is listening on or you can use the -all parameter to stop all running Solr instances.

If you're not sure if Solr is running locally, you can use the status command:

$ bin/solr status

Read Apache Solr online: https://riptutorial.com/solr/topic/9635/apache-solr

Chapter 3: How to create a custom fieldType

Remarks

Remarks on custom Java class based field:

This is a small section of a large article custom sorting in Solr using external field written to sort Solr documents based on custom field comparator.

Remarks on custom field created from existing Solr fields:

Apache has created a detailed documentation on this topic - Understanding Analyzers, Tokenizers, and Filters.

Examples

Create a custom Solr field type from own custom Java class

Schema changes:

You will need to define a new field type in your solr schema file and then you can create fields of that type. Example schema snippet:

```
<!-- Source: solr/example/.../conf/schema.xml -->
<?xml version="1.0" encoding="UTF-8" ?>
<schema name="adam" version="1.3">
  <types>
    ...
    <fieldType name="rank_t" class="org.apache.solr.schema.ext.RankFieldType"/>
  </types>
 <fields>
   ...
   <field name="rank" type="rank_t" indexed="true" stored="true"/>
 </fields>
 ...
</schema>
```

Java class for custom field type:

```
// Source: src/java/org/apache/solr/schema/ext/RankFieldType.java
package org.apache.solr.schema.ext;

import java.io.IOException;

import org.apache.lucene.document.Fieldable;
import org.apache.lucene.search.SortField;
import org.apache.solr.response.TextResponseWriter;
import org.apache.solr.schema.FieldType;
import org.apache.solr.schema.SchemaField;
import org.apache.solr.search.ext.RankFieldComparatorSource;
```

```
public class RankFieldType extends FieldType {

    @Override
    public SortField getSortField(SchemaField field, boolean top) {
        return new SortField(field.getName(), new RankFieldComparatorSource(), top);
    }

    @Override
    // copied verbatim from GeoHashField method
    public void write(TextResponseWriter writer, String name, Fieldable f) throws IOException
{
         writer.writeStr(name, f.stringValue(), false);
    }
}
```

Create custom field type from available field types

Let's get some theoretical knowledge before moving to the example. There are three important terms being used here Analyzers, Tokenizers, and Filters. To create such custom field you will need to create an analyzer with one tokenizer and one or more filters. As mentioned here, you can have only one tokenizer per analyzer but there are ways to overcome this limitation.

```
<fieldType name="alphaOnlySort" class="solr.TextField" sortMissingLast="true"
omitNorms="true">
  <analyzer>
    <tokenizer class="solr.KeywordTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.TrimFilterFactory"/>
    <filter class="solr.PatternReplaceFilterFactory" replace="all" replacement=""
pattern="([^a-z])"/>
  </analyzer>
</fieldType>
```

Another example:

```
<fieldType name="lowercase_text" class="solr.TextField" positionIncrementGap="150">
  <analyzer>
     <tokenizer class="solr.KeywordTokenizerFactory" />
     <filter class="solr.LowerCaseFilterFactory" />
  </analyzer>
</fieldType>
```

One more example with description:

```
<fieldType name="text_stem" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPorterFilterFactory"/>
  </analyzer>
</fieldType>
```

This example starts with Solr's standard tokenizer, which breaks the field's text into tokens. Those

tokens then pass through Solr's standard filter, which removes dots from acronyms, and performs a few other common operations. All the tokens are then set to lowercase, which will facilitate case-insensitive matching at query time. The last filter in the above example is a stemmer filter that uses the Porter stemming algorithm. A stemmer is basically a set of mapping rules that maps the various forms of a word back to the base, or stem, word from which they derive. For example, in English the words "hugs", "hugging" and "hugged" are all forms of the stem word "hug". The stemmer will replace all of these terms with "hug", which is what will be indexed. This means that a query for "hug" will match the term "hugged", but not "huge".

Example usage of such custom field:

```
<field name="keywords" type="text_stem" indexed="true" stored="true" />
```

List of available tokenizer types: list of tokenizer types

List of available filter types: list of filter types

Read How to create a custom fieldType online: https://riptutorial.com/solr/topic/7409/how-to-create-a-custom-fieldtype

Chapter 4: INDEXING HIVE2 TABLE IN SOLR USING SOLR DIH

Introduction

This documentation provides a way to connect to hive using SOLR Data Import Handler and index the data in SOLR. This is an interesting documentation because I couldn't find it over internet.

The handler basically handles more than 80 million records which means a strong infrastructure with good CPUs and Memory is definitely needed.

Remarks

Please feel free to reach out to us and we will try to help as much as possible.

Examples

Steps

We got the hive2 jars first and made it working on through java to check the connectivity. Then we realized the jars to be used are :

1. hadoop-common-2.7.0-mapr-1703.jar
2. hive-common-2.1.1-mapr-1703-r1.jar
3. hive-jdbc-2.1.1-mapr-1703-r1-standalone.jar

If you are using SOLR Cloud then these jars are to be transferred to the VM where SOLR is installed and then referenced in solrconfig.xml like this:

Import Part in solrconfig.xml

< lib dir="/users/path_to_folder_with_jar" regex=".*.jar" />

Then this is the most important part: Your hive connection string:

Connection Part

< dataConfig > < dataSource name="ABC" driver="org.apache.hive.jdbc.HiveDriver" url="jdbc:hive2://....connectionString" user="username" password="password" />

< document name="collection_name">

< entity name="collection_lookup" query="select unique_key as id from table_name">

< /entity>

< /document>

< /dataConfig>

Push config through zookeeper

server/scripts/cloud-scripts/zkcli.sh -zkhost host1:2181,host2:2181 -cmd upconfig -confname configName -confdir server/solr/configsets/folder/

Go to http://host:8983/solr/#/collection_name/dataimport//dataimport then check debug and first check with 10 or 20 records.

You will see the data flowing. CHEERS !! I can help if you want to discuss further but I am assuming this should do. It is working for me.

Read INDEXING HIVE2 TABLE IN SOLR USING SOLR DIH online: https://riptutorial.com/solr/topic/10697/indexing-hive2-table-in-solr-using-solr-dih

Chapter 5: Lucene Query Syntax

Examples

Proximity search

```
name:"john doe"~1
```

Searches for multiple terms within a specific term distance (**~1**), i.e will find text containing **john anonymous doe** but not **john second name doe**

Basic search

```
name:john
```

Searches for a single term (joe) in a single field (name)

Boolean search

```
+firstname:john +surname:doe
```

Matches documents where firstname is john and surname is doe. **+** predix indicates that the search term *must* occur (AND).

```
+firstname:john -surname:doe
```

Matches documents where firstname is john and surname is not doe. **-** predix indicates that the search term *must not* occur (NOT).

```
+firstname:john surname:(doe bloggs)
```

Matches documents where firstname is john and surname is either doe or bloggs. No prefix indicates that the surname *should* occur (OR)

Phrase search

```
name:"john doe"
```

Searches for multiple terms in specific order.

Boosting search terms

```
name:(john doe^5)
```

The **^** indicator can be used to boost a search term to increase it's relevance level meaning that documents containing *doe* are more relevant than ones containing *john*

Wildcard search

```
name:john*
```

The * indicator allows you to do a wildcard search matching 0 or more characters after the search term *john*, will return documents containing john, johnson, john's, johnny and so on.

```
name:do?
```

The ? indicator allows you to do a wildcard search with a single character in the search term, will return documents containing doe, dog, dot and so on.

Range search

```
age:[50 TO 60]
```

Matches documents where age is between 50 and 60 including 50 and 60

```
age:{50 TO 60}
```

Matches documents where age is between 50 and 60 excluding 50 and 60

```
age:[* TO 60]
```

Matches documents where age is less than or equal to 60

```
age:[50 TO *]
```

Matches documents where age is greater than or equal to 50

```
age:{50 to 60]
```

You can mix curly and square brackets. Matches documents where age is greather than 50 but less than or equal to 60

Join across cores

```
{!join from=personid to=id fromIndex=AddressCore}address:Address1
```

So if you have two cores that look like this:

PersonCore - id, name

AddressCore - id, address, personid

This will find all PersonCore documents at a specific address

Read Lucene Query Syntax online: https://riptutorial.com/solr/topic/2242/lucene-query-syntax

Chapter 6: Troubleshooting Solr

Remarks

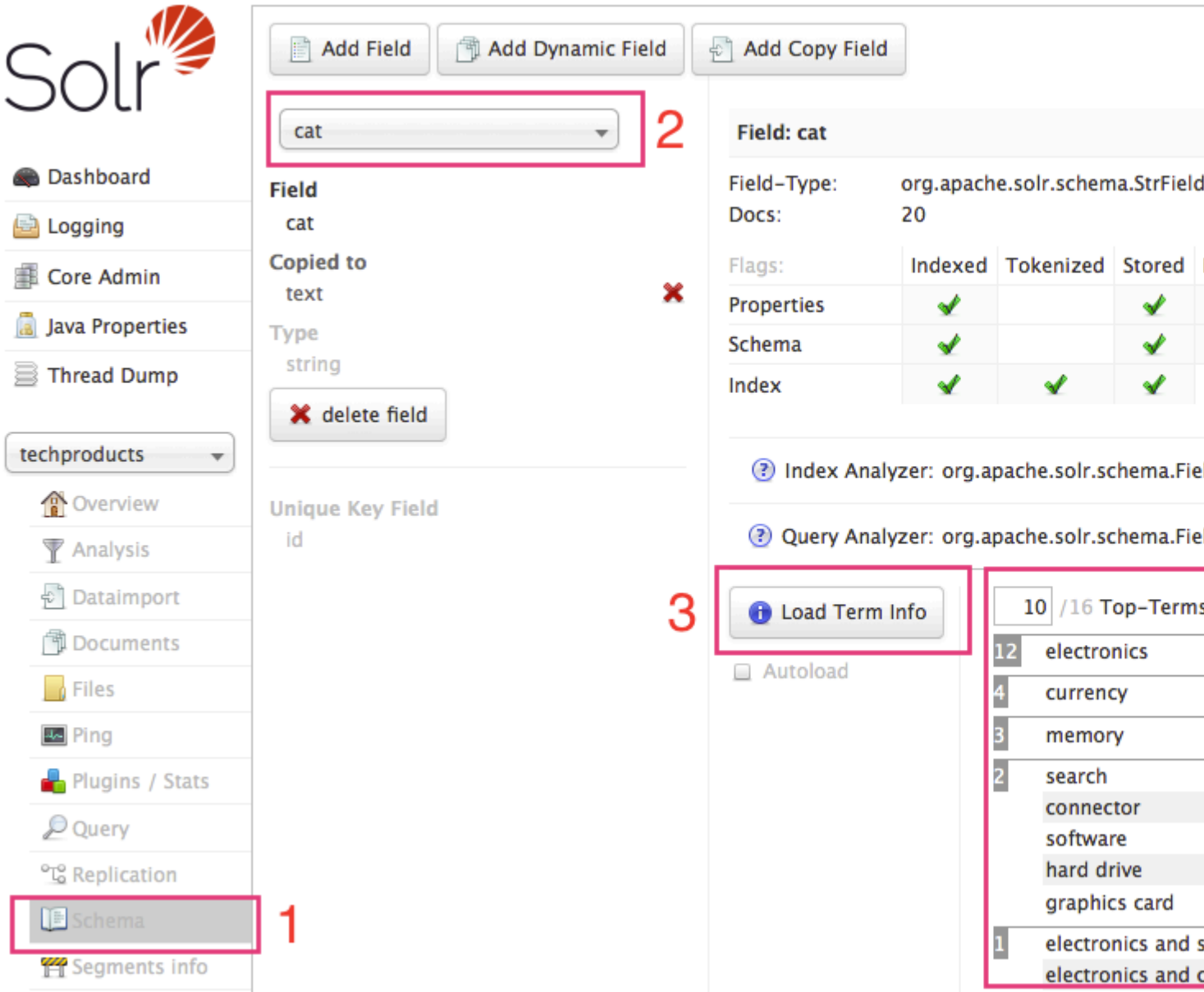
When searching against the target of a copyField instructions, the analyzer stack used is the one of the copyField target, not source. This is especially important to remember with default configuration that searches against generic **text** field.

Examples

Has my content actually indexed?

Often people try to index some content and then find it. If they cannot see the expected result, they try to troubleshoot the whole end-to-end process. The better way is to see whether the content actually indexed in the expected fields. This way it splits the problem into two: indexing and searching.

The easiest way to verify what indexed is in the Admin UI's Schema screen (1). Just select the relevant field (2) and load its Term info (3). That's all the indexed terms in that field. The list (4) could be quite long, but you can change number of items shown and/or index into a separate test field just for debugging.



If you see nothing or no expected content, it usually means the indexing failed. If you see the content here, but not in the query, then it is the search you need to troubleshoot.

Read Troubleshooting Solr online: https://riptutorial.com/solr/topic/1969/troubleshooting-solr

Credits

S. No	Chapters	Contributors
1	Getting started with solr	Community, esiprogrammer, Lefty G Balogh, mfatihk, SimplyInk
2	Apache Solr	Prahlad Awasthi
3	How to create a custom fieldType	RamenChef, saurav
4	INDEXING HIVE2 TABLE IN SOLR USING SOLR DIH	Khalid Imam
5	Lucene Query Syntax	BunkerMentality, Fuu, MatsLindh
6	Troubleshooting Solr	Alexandre Rafalovitch