Spring 2015

Enterprise Search Technology Using Solr and Cloud

Padmavathy Ravikumar
Governors State University

ENTERPRISE SEARCH TECHNOLOGY USING SOLR AND CLOUD

By

Padmavathy Ravikumar
[redacted]

Masters Project

Submitted in partial fulfillment of the requirements

For the Degree of Master of Science,
With a Major in Computer Science

Governors State University
University Park, IL 60484

Fall 2014

Abstract

Solr is the popular, blazing fast open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database in9tegration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites.

Databases and Solr have complementary strengths and weaknesses. SQL supports very simple wildcard-based text search with some simple normalization like matching upper case to lower case. The problem is that these are full table scans. In Solr all searchable words are stored in an "inverse index", which searches orders of magnitude faster.

Solr is a standalone/cloud enterprise search server with a REST-like API. You put documents in it (called "indexing") via XML, JSON, CSV or binary over HTTP. You query it via HTTP GET and receive XML, JSON, CSV or binary results. The project will be implemented using Amazon/Azure cloud, Apache Solr, Windows/Linux, MS-SQL server and open source tools.

The following are the software requirements of my project:

Content	**Description**
OS	**Windows/ Linux/ Unix**
Database	**MS-SQL server 2012**
Technologies	**Apache Solr**
Cloud	**Amazon, Azure**
Browser	**IE , Chrome, Firefox**

Contents

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

OVERVIEW

Apache Solr is a fast open source Java search server. Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene.

The initial method of implementing search operations was by building databases. In the backend, every search request was compared to each and every record in the database to find a match and return the result. When the database was small, the retrieval was manageable. The problem arose as the database size increased. To combat this problem, indexing process was introduced in databases. This also did not solve the problem entirely. It is exactly in this scene that the Lucene, Solr, and other similar search engine concepts were born. In my project, I will use Solr.

Solr facilitates fast search responses because, instead of searching the text directly, it searches an index instead. The benefits of Solr include advanced full-text search capabilities, scalability to handle high volume traffic, comprehensive administrative interfaces, easy monitoring, high fault tolerance, supportive of many file formats, flexibility, easy adaptability, near real-time indexing, and extensible plugin architecture.

My project aims to addresses the basic concepts of the open source Apache Solr platform that is specifically designed for indexing documents and executing searches.

At the end of my project, the following learning objectives would have been accomplished:

- Describe the basic workings of the Apache Solr search engine.
- Add/update/delete data to be searched using Apache Solr.
- Execute simple queries utilizing concepts of estimated relevance, sorting, facets, etc.

- Describe how queries are parsed by Apache Solr.

Data and Information Technology

This is the age of Information Technology. Data collection, storage, search, retrieval, and availability are just the key functions made possible by the advent of Information technology coupled with many sciences. Today, if we are able to complete our entire course study by just gathering information in the computer, it is a big thanks to the IT. If we are able to wish our once-upon-a-time childhood friend on her birthday, it is a big thanks to Facebook. Similarly, we owe thanks to twitter to get our messages across even before the person who is behind the original message blinks an eyelid. And on and on and on………..the list goes on to the point where today, we are all dependent on information technology. All these mean big time data and information management.

Searching and Search Engines

Search Engines work with this principle: search engines consist of certain components that search the web for links and keywords. The components send the data to indexing software which categorizes the information and send the links to the database along with the keywords. When a search request is made, the engine just extracts the related links and keywords from the database. Search engines can use different concepts such web search, smart, string, etc. There are different types of searches and search engines.

Overall, there are the following different types of search engines such as the follows:

- **Crawler based search engines**

  Crawler based search engines such as Google, AllTheWeb, AltaVista create their listings automatically by using a piece of software to "crawl" or "spider" the web and then index what it finds to build the search base. Web page changes can be dynamically caught by crawler-based search engines and will affect how these web pages get listed in the search results.

- **Human powered search engines**

  Human-powered directories, such as the Yahoo directory, Open Directory and LookSmart, depend on human editors to create their listings. Typically, webmasters submit a short description to the directory for their websites, or editors write one for the sites they review, and these manually edited descriptions will form the search base. Therefore, changes made to individual web pages will have no effect on how these pages get listed in the search results.

- **Meta Search Engines:**

  Metasearch Data such as Dogpile, Mamma, and Metacrawler, transmit user-supplied keywords simultaneously to several individual search engines to actually carry out the search. Search results returned from all the search engines can be integrated, duplicates can be eliminated and additional features such as clustering by subjects within the search results can be implemented by meta-search engines.

  Meta-search engines are good for saving time by searching only in one place and sparing the need to use and learn several separate search engines. "But since meta-search engines do not allow for input of many search variables, their best use is to find hits on obscure items or to see if something can be found using the Internet."

  - There are different web search engines such as Archie, Veronica, The Lone Wanderer, Excite, Yahoo, Lycos. The following are the different categories of search engines such as Web Search Engine, Crawl, Link map, Database Search Engines, Mixed Search Engines. The different types of search are conventional, text based, Multimedia, Q/A, Clustering Systems, Research Systems, Enterprise Search Systems, Web Based Search Systems.

All of the search engines perform the following basic tasks:

- Find full or partial text content based on keywords provided.
- Maintain index of content and references to locations they find
- Facilitate users to search for words or a combination of words in the index.

Database Approach to Searching

Consider the case of the following search engine.

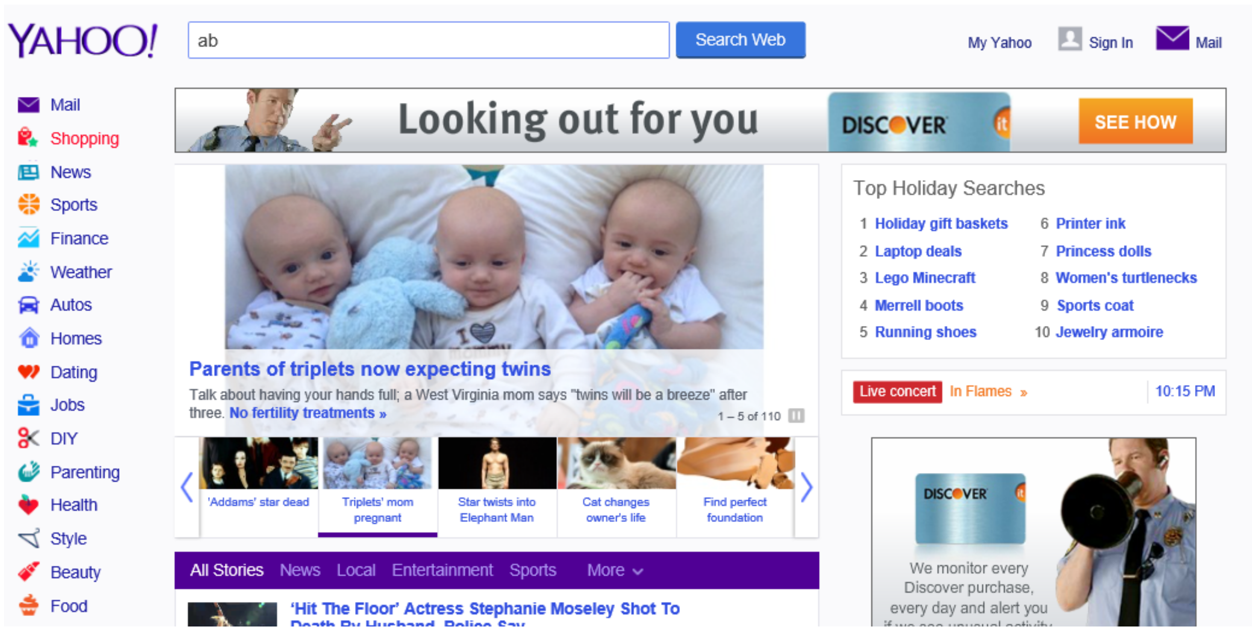


Fig 1. Yahoo Website

As you can see in the above search engine; there is a search box or a query box where you can type any expression or string or text string to be searched. There are different options next to the search box also allows the user to filter the search criteria. These features are the search components of the search application.

To build a search application or an application with search requirements, one always considers the database implementation. The database facilitates search based results. In the backend, this is what happens whenever a search request is made by the user. When the user provides a search string, the terms in the string is searched with every record or each and every record in the entire database table. When a match is found, the results are returned back to the user.

The issue with this type of searching arises when the database size is very large. In case, the number of records in the database is small, the results can be retrieved faster. But as the number of records increases in their size, and there are millions and millions of records, it is very difficult to handle data searches.

As an alternative, architects introduced the indexing process. Indexes were built in the databases initially. Although, this process temporarily provided a performance gain but was still a tedious task to handle data and it was also difficult to handle re-indexing of data in the database.

Again as an alternative, architects decided to shift the indexing process away from the database in a structure. This way, there was no need for the search request to hit the database, therefore ensuring that quicker faster retrieval of information.

Apache Solr and Lucene Approach to Searching

Apache Lucene is a search engine library for full text. It is written entirely in Java. Lucene is embedded with Solr. It is free open-source information retrieval software library. Some of its uses include full text indexing, searching capability, and its utility in implementation of Internet Search agents, site searching.

The concept of Lucene works like this: at the center of lucenes logical architecture is a document that has fields of text. This flexibility in architecture allow the lucenes API to be independent of the file formats. Hence files of different formats such as PDFs, HTML, Microsoft Word, OpenDocument documents can be indexed and the textual information can be extracted.

Lucene, in itself is just an indexing and search library and does not contain other functionalities such as crawling and HTML parsing functionalities. There are several projects that are based on luscene such as the follows. Apache Solr is one of them.

Apache Nutch, Apache Solr, ElasticSearch, Compass, DocFetcher, Swicthtype, KinoSearch, Apache Lucy, Luke.

Apche Solr is an enterprise search platform. It is also written in Java. It serves web services that can manage a document's lifecycle in the index. Solr is an enterprise search platform from the Apche Lucene project. It runs as a full-text search server within a servlet container such as Apache Tomcat or Jetty. Solr works by Lucene as its core or rather it works on top of Lucene and provides services such as full text indexing. Solr has HTML/XML that like Rest and JSON APIs that make it one of very popular languages. Solr's other useful feature are 1: powerful external configuration that allows it to be tailored to many application types without java coding; 2: it has a plugin architecture that supports more advanced customizations.

Lucene is used to create a search index and Solr uses this index to perform searches. Solr works above Lucene. Lucene is a powerful search engine framework that lets us add any search capability to our application. Lucene provides an easy-to-use API while performing or running the entire search related complex operation in the background. Any application can use search library of the Lucene feature. Solr is built around Lucene. It provides arsenal around Lucene. Solr is ready-to-use and is out-of-box. It offers related infrastructure and a lot more features in addition to Solar Lucene.

When to use Lucene?	**When to use Solr?**

You are a search engineer and a programmer have full control of Lucene, requirements demand to customize Lucene, and a willingness to take care of infrastructure elements	You need to use Solr in either of the following cases: 1: You want to use a software tool that does not require knowledge of Java 2: You infrastructure requirements outweigh the customization requirements

Table 1. Solr and Lucene

Lucene and Solr are like car and engine. Lucene is like engine and Solr is like car. You can't drive an engine but you can drive a car. Lucene is a programmatic library that you can't use as-is. Solr is a complete application that you can use out-of-box.

FOCUS

As mentioned in the previous section, the focus of the project is to addresses the basic concepts of the open source Apache Solr platform that is specifically designed for indexing documents and executing searches.

The learning objective of my project is to describe the basic workings of the Apache Solr search engine; add/update/delete data to be searched using Apache Solr; execute simple queries utilizing concepts of estimated relevance, sorting, facets, etc; and describe how queries are parsed by Apache Solr.

This paper will focus on the following features of Apache Solr:

- Connecting to the Amazon Web Services server
- Create an instance in the cloud server
- View the running instances in the cloud

- Stop or run the processes in the cloud
- Access the users in the running instances of the cloud
- Access the folders of the users
- Index the documents in the folder
- Search for indexed data

The above are some of the basic functions of Apache Solr. We can also implement examples to this project implementation. For example, we have provided the following two examples in our project:

Indexing the file containing twitter feeds of Governors State University and then searching for specific tweets filtered by content or person. Another example that I will be providing is with reference to an airport application. Both these examples show you how files are indexed using Apache Solr and how the terms are searched and retrieved from the Indexed Document.

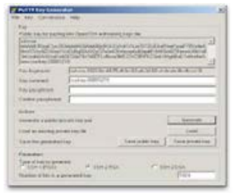
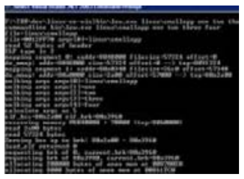
Search and Indexing are the two major features of any search engine and so is it for our project. We are trying to explain in our project what differentiates Apache Solr based search from the regular and traditional database based searches. And the differentiating factor lies in the way Apache Solr searches and Indexes data. Hence, a little light will also be thrown on how the content will be searched and indexed.

Fig 2. Indexing and Searching

Since Apache Solr is a highly technical concept, I will also explain the common terms and concepts that one needs to know beforehand in order to understand how Apache Solr works. I will also be providing insight into the other tools used in the project and give references of all the books and sites that I referenced for completing my project as well as my documentation.

TOOLS

The following is the list of tools that my project uses:

APACHE SOLR	Apache Solr
APACHE LUCENE SOLR	Lucene Apache Solr
PUTTYgen	 [illegible]
LINUX	 [illegible]
FILE ZILLA	FZ

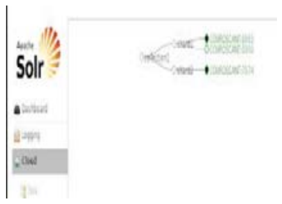
SOLR CLOUD	

Table 2. Tools of the Trade

CONCEPTS AND TERMINOLOGIES

Search library:

A search library consists of documents and texts structured in such a way that facilitates easy and full-text indexing, search and retrieval of any matching terms and data.

Document

A document in Solr context is a container or a primary storage unit that contains a flat collection of fields

Field

A Field is a definition that consists of 2 parts: **name** and **configurable type** (text, integer, double, date).

Core

Core is a separate index and configuration. Cores are used for data partitioning and supporting multiple applications. A single server can support multiple cores.

SolrCloud

SolrCloud in a nutshell refers to a group of features that facilitates scaling of your search index across a cluster of nodes.

Synonyms

This is a query expansion feature For Example MB => megabyte

Stop Words

These are words that should be filtered out from index storage and queries

Structured Content

This refers to content that has been richly tagged with metadata.

Unstructured Content

This includes MS Office, PDF documents, emails, instant messages, etc.

ACL

This is the abbreviation of access control list that keeps track document permission information.

Early Binding

This is an authorization enforcement model where the document ACLs are stored in the index.

Late Binding

This is an authorization enforcement model where document authorization is not determined up until query time.

ETL

This is the abbreviation of extract (content source), transform (normalize the data), load (into index)

Searchased Application

This is built above the search platforms and they are designed to deliver information access that is unified.

Index:

Index is an indirect short cut that is pointing to a great volume of data, information, or knowledge.

Hadoop:

The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

API:

In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising each other. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.

Cloud

Cloud is a network of servers. Cloud services are the opposite of having dedicated server for storage of data. Some of the benefits of cloud include **reliability, scalability, accessibility, and cost**.

INDEXING

Searching and Indexing are the two aspects that any search engine must focus. Search engines indexes, collects, parses, and stores data for speedy data access and retrieval. The purpose of storing an index is to optimize speed and performance in finding related documents for a search query. If there is no index, the search engine would scan each and every record in the database or corpus before finding and returning the search result, if any match. When the size of the database is large, indexing is the best option. There are different ways of indexing.

Database indexing is inefficient as it does not use tokenization method and it cannot handle large volume of data. Indexing is similar to that used at the back in the book. The same technique that is used in books is that which is adapted by the Lucene and since Solr is based on Lucene, the same applies to Solr as well. Index is like a look-up table.

Indexing Process

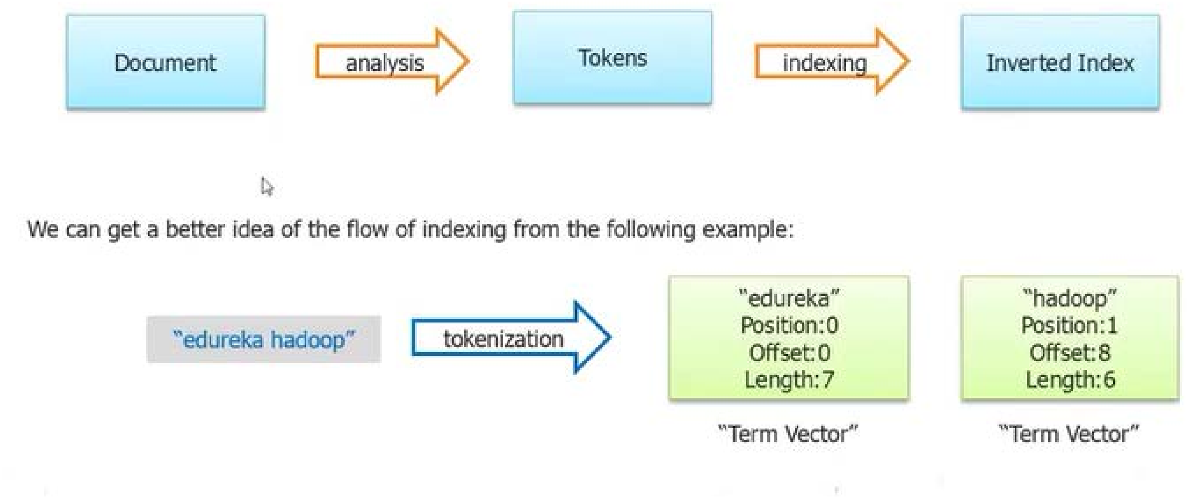

Document
analysis
Tokens
indexing
Inverted Index
We can get a better idea of the flow of indexing from the following example:
"edureka hadoop"
tokenization
"edureka" Position:0 Offset:0 Length:7
"hadoop" Position:1 Offset:8 Length:6
"Term Vector"
"Term Vector"

Fig 3. Indexing

As you can see from the above figure. Indexing is a series of the following steps.

A document which is a record can consists of various information contained in the form of fields and records. A document is first submitted for Analysis. The analyzer analysis the document fields and records for the type of information each unit contains: numerical info, binary info, and alphanumeric text, and other such information. Any type of file of any format can be indexed and searched. Next, each string of text will be split into tokens. For example, consider a string of text Hello World. The string is broken into two tokens: Hello and World. The two terms are saved along with their attributes such as position, offset, and length. All these properties will be used to search and retrieve any of the tokens which in our example is Hello or world or both.

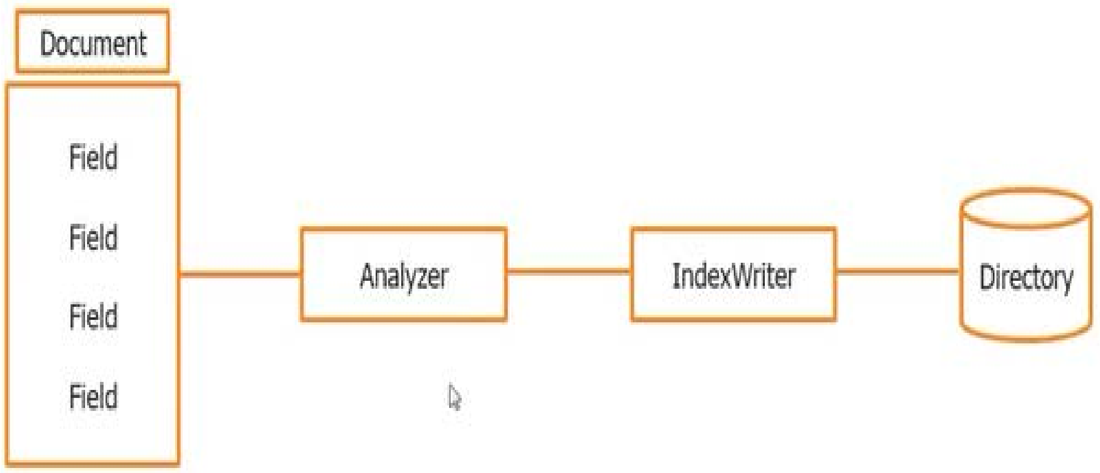


Fig 4. Indexing Process

As you can see from the above figure, the document is first submitted to the Analyzer. Analyzer does the tokenization and filtering of the documents. Analysis is the most difficult part of the Indexing process. After that, the document is broken into vectors and tokens. These tokens and term vectors are then sent to the Index Writer. The directory is again nothing but an Index that contains the Indexed terms.

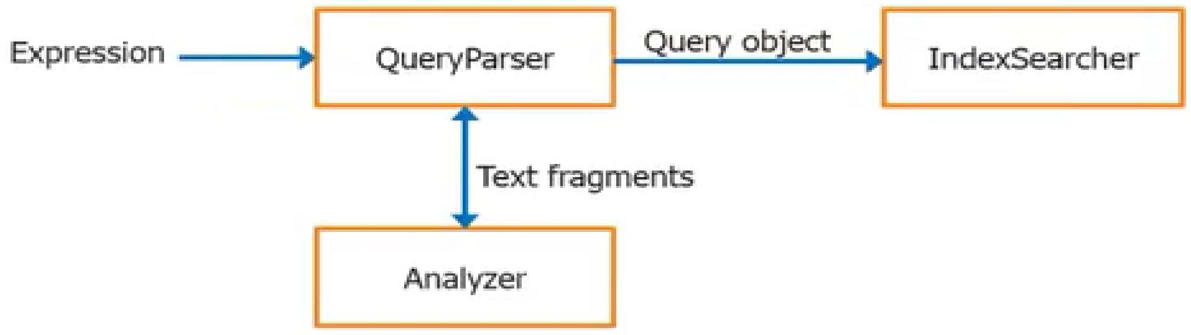


Fig 5. Querying

In the above figure, data stored in the index format has to be retrieved. There are different approaches to do this. One such method is by using SQL. Expression are used to retrieve indexed data. Query Parser translates the textual expression into an arbitrary complex expression for the search. The Query Parser and Analyzer work in sync and decide whether the expression fits into the indexed data and format. If yes, the units (Query Parser and Analyzer) together convert the expression into a Query Object. The Query Object is submitted to the Index Searcher for the search process to begin.

Inverted Indexing

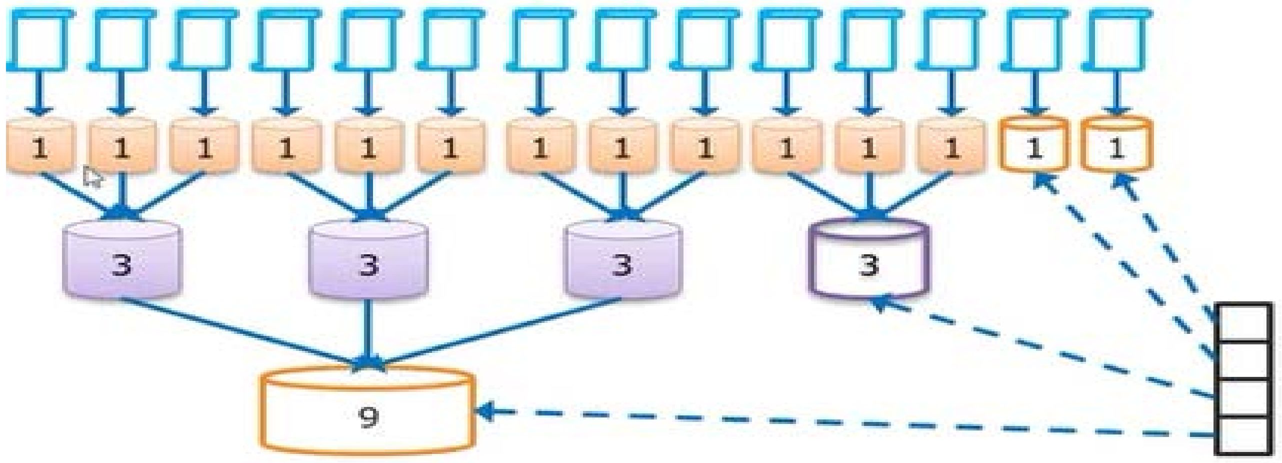


Fig 6. Inverted Indexing

Inverted Index is a technique of reversing the normal index pattern. These kind of indexes are faster to read. All documents are organized in segments. It is cost effective process, reduces the restructuring effort.

For example, we want to add all of the documents and save it in an efficient and organized manner. Segments are created to hold chunks of indexed documents. When a search request is made, a lookup is made into a segment which is most likely to hold the group of related terms. This way, time is saved as well as there is more probabilities of getting a hit or match.

Merge Sort concepts is used in the Inverted Indexing process. After a threshold of maximum document updates, the documents are merged and changes are made to the main index. It is basically a method of accumulating changes before updating the Index with the changes. This way, the index is not restructured or modified every single time that the search or updated is required. Instead, changes are recorded in the local segments, merged after only some point of time. But care must be made to reference the global and local segments to ensure retrieval of the latest data. The following figure surmises the Indexing process.

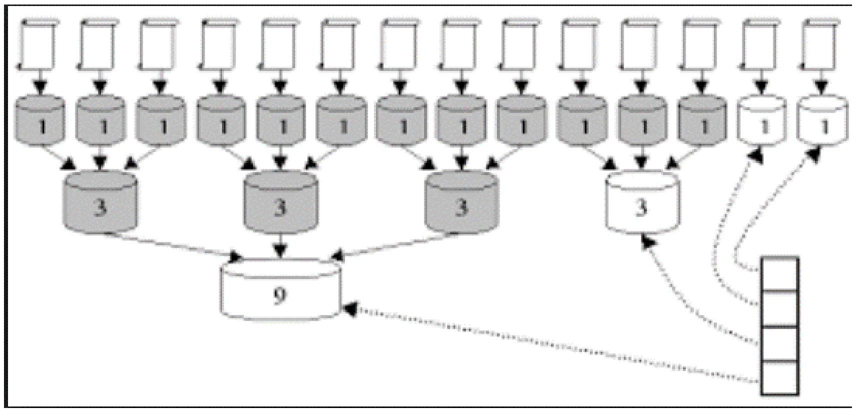


Fig 7. Merge Sort

TOOLS OF THE TRADE

Although we will be looking using the tools such as Linux, Solr, Amazon Web Server, I will discuss Apache Solr as it is the only major concept on which our entire concept is based.

LUCENE

Since Solr inevitably works on the lines of Lucene, I am just highlighting the Lucene architecture.

Lucene Architecture

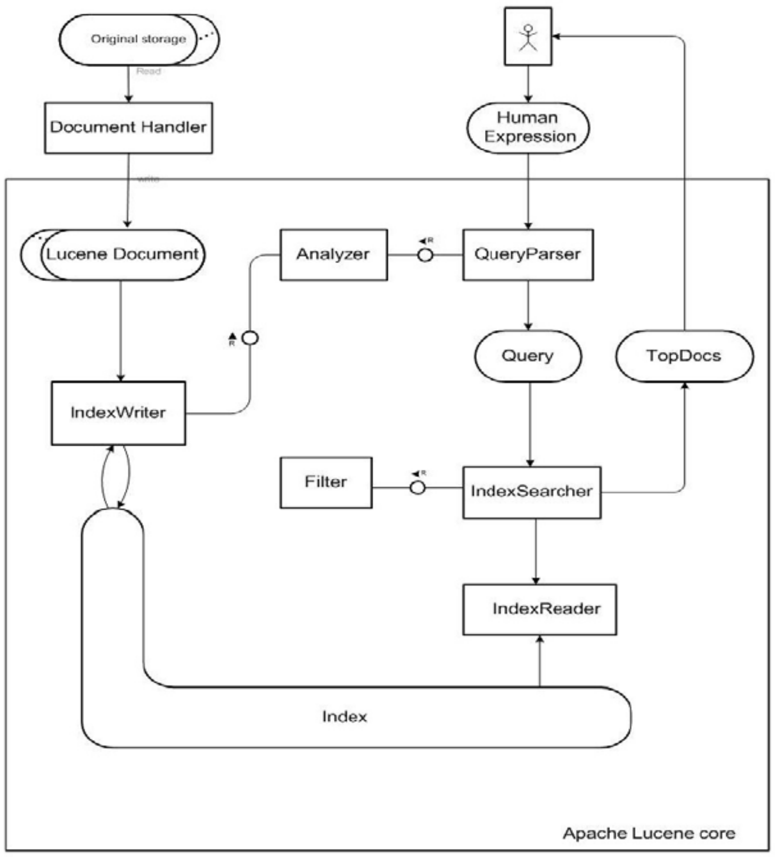


Fig 8. Lucene Architecture

Document Handler

Any application using Lucene must transfer all their original documents into Lucene documents. To do all the conversion; a document handler is needed and all of this is done by a Lucene contribution library. A document handler allows for extraction of certain types of information from the original document and converts them into Lucene documents which are used in the future processing like indexing and search.

Some documents like HTML, PDF, XML and so on needs specific document parsers. Parsers are available in the market for free and can be purchased and integrated into the system.

Index Writer

This is the next phase here the Index Writer analyses and stores Lucene documents into an index. The indexing is done according to specific techniques. The index writer uses the help of one or more analyzers as a strategy for index writing.

Analyzer

The analyzer removes unneeded expressions from the content such as spaces, hyphen, stop words, and many more depending on the analyzer type. At the end of analysis step, the document contains just text that are terms and that can be used for search.

Query Parser

To search inside the index, the user has to provide a query which is a human readable expression which has to be converted into an object of type Query. And this is the job of the Query Parser. The Query object has to be analyzed and then assigned to the Index Searcher.

Index Searcher

This is the core of the searching process. It uses the **Index Reader** to access the Index and retrieve all the terms that match the terms in the Query Object; and then return all of the hints and topdocs as the search results.

Filter

You can use a filter to permit or prohibit one or more terms in the search results.

Lucene Functions:

- Provides an index of data
- Parses user query string
- Searches the index for query terms
- Computes statistics over indexed terms
- Process management
- Files selection and display of search queries.

SOLR

Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene™. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites. Solr powers some of the most heavily-trafficked websites and applications in the world. (Whitehouse.gov, FCC.gov, CNET, Ebay, Apple, Nasa, Sears, Bestbuy, AOL, AT&T, Netflix, Disney, Bloomberg.)

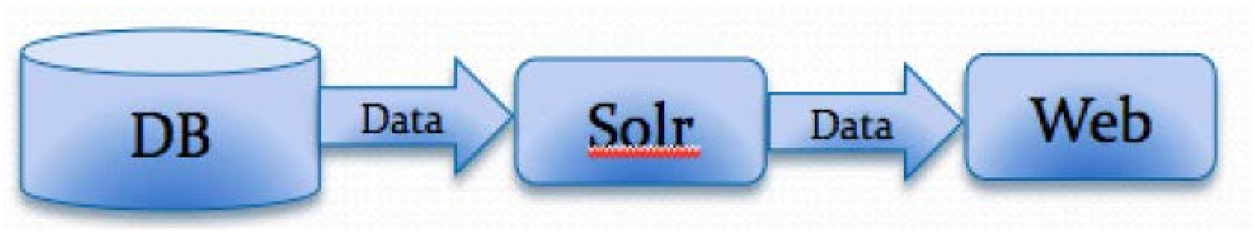


Fig 9. Solr

The core technology underlying Apache Solr is Lucene. Lucene was developed and open sourced by Doug Cutting in 2000 and has evolved and matured since then with a strong online community. In order to use Lucene directly, one writes code to store and query an index stored on a disk. Solr is considered as the server-ization of Lucene.

There are many other open source search projects like Katta, Elastic-Search. Katta is an index serving tool (not an indexer), so we have to perform indexing operation using Lucene running on hadoop and manage indexes. ElasticSearch is an open source search engine like Solr, currently lacking on some of the search features.

Because of these reasons we chose Apache Solr over these products. Solr and Lucene is a different story altogether. Lucene is a very compact and powerful search library while Solr is an enterprise search engine built on top of the Lucene library. Lucene gives information retrieval core technology in a compact package, and Solr builds out features on top, including: a platform independent interface, replication, caching, large scale distributed search, and much more.

Solr Features

The following are some of the features of Solr.

- 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.

- Advanced Full-Text Search Capabilities: 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: Near Real-Time Indexing

  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!

  We use Solr for benefits such as the following:
- Performance
- Optimized for search
- Distributed Search / Index Replication

- Reduce Load
- Faster Results
- Facets: From the user perspective, faceted search (also called faceted navigation, guided navigation, or parametric search; Microsoft calls them "Refiners") breaks up search results into multiple categories, typically showing counts for each, and allows the user to "drill down" or further restrict their search results based on those facets.
- Document Handeling (PDF/Word, etc)
- Better Search Algorithms
- Full Text Search
- Word Streaming, Splitting, etc.
- Highly Configurable

Solr Concepts and Terminologies

Core

Core is a separate index and configuration. Cores are used for data partitioning and supporting multiple applications. A single server can support multiple cores.

Shard

A shard is a chunk of a larger index. It is a logical slice of a collection of search indexes that are distributed across many nodes. They are used for horizontal scaling of an index horizontally. Documents are assigned to only one shard per collection using a hash based document rounding strategy.

SolrCloud

SolrCloud in a nutshell refers to a group of features that facilitates scaling of your search index across a cluster of nodes.

ZooKeeper:

Zookeeper is a distributed coordination service that provides services such as leader election, centralized configuration, and cluster state management.

Node:

Node refers to the Java Virtual Machine that is bound to a specific port on a machine and that hosts the Solr web application.

Collection:

This refers to the search index which is distributed across many nodes. Each collection has a name, count, and a replication factor.

Replication Factor:

This refers to the number of copies of a document in a collection.

Replica:

This is a Solr index that hosts a copy of a shard in a collection. Each replica is implemented as a single Solr core.

Leader:

Replica in the shard that has special duties that are required to support distributed indexing in Solr. Each Shard has one and only one leader at any time. The zookeepers elect the leaders.

Solr Cloud Architecture

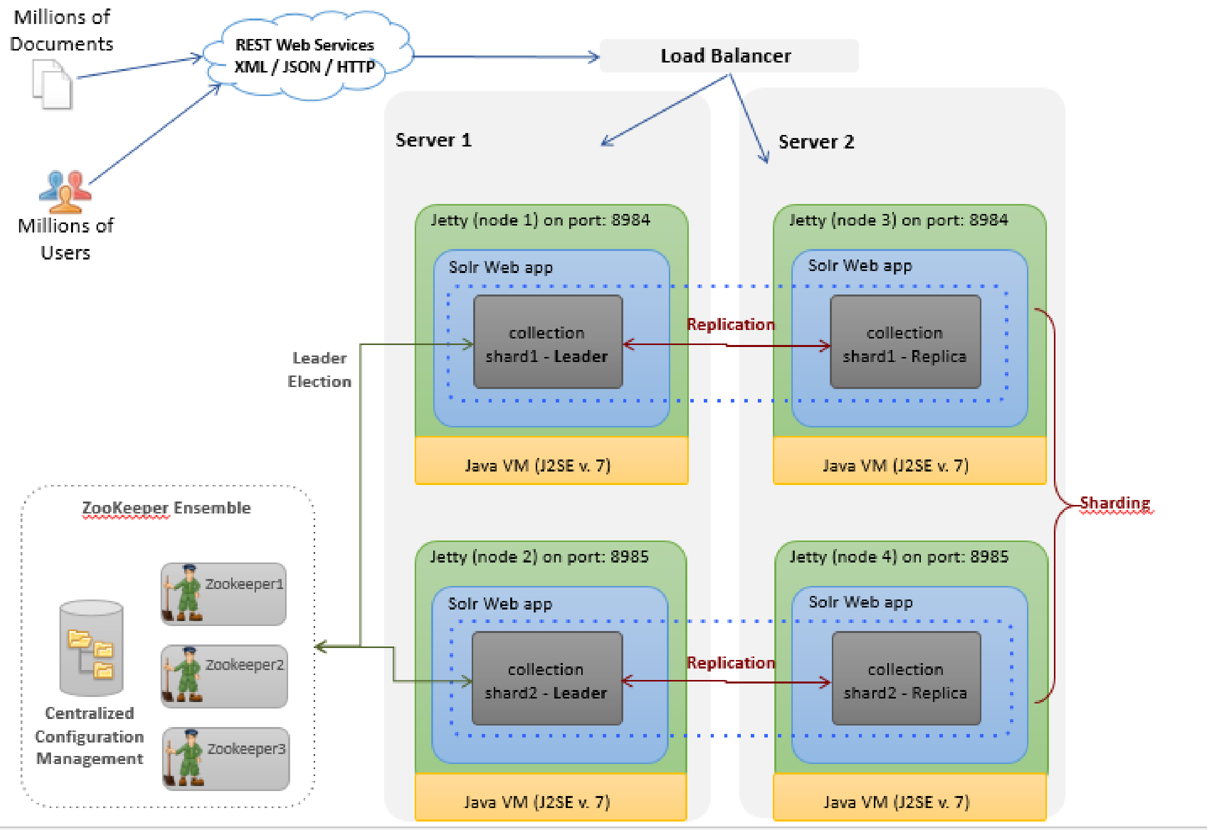


Fig 10. Solr Cloud Architecture

As you can see from the above architecture, there are four nodes that are distributed across two servers. There is a collection in each node of the server. One collection batch serves the role of the leader while the other serves the role of the replica. The ZooKeeper manages this centralized configuration.

Solr/Lucene Architecture

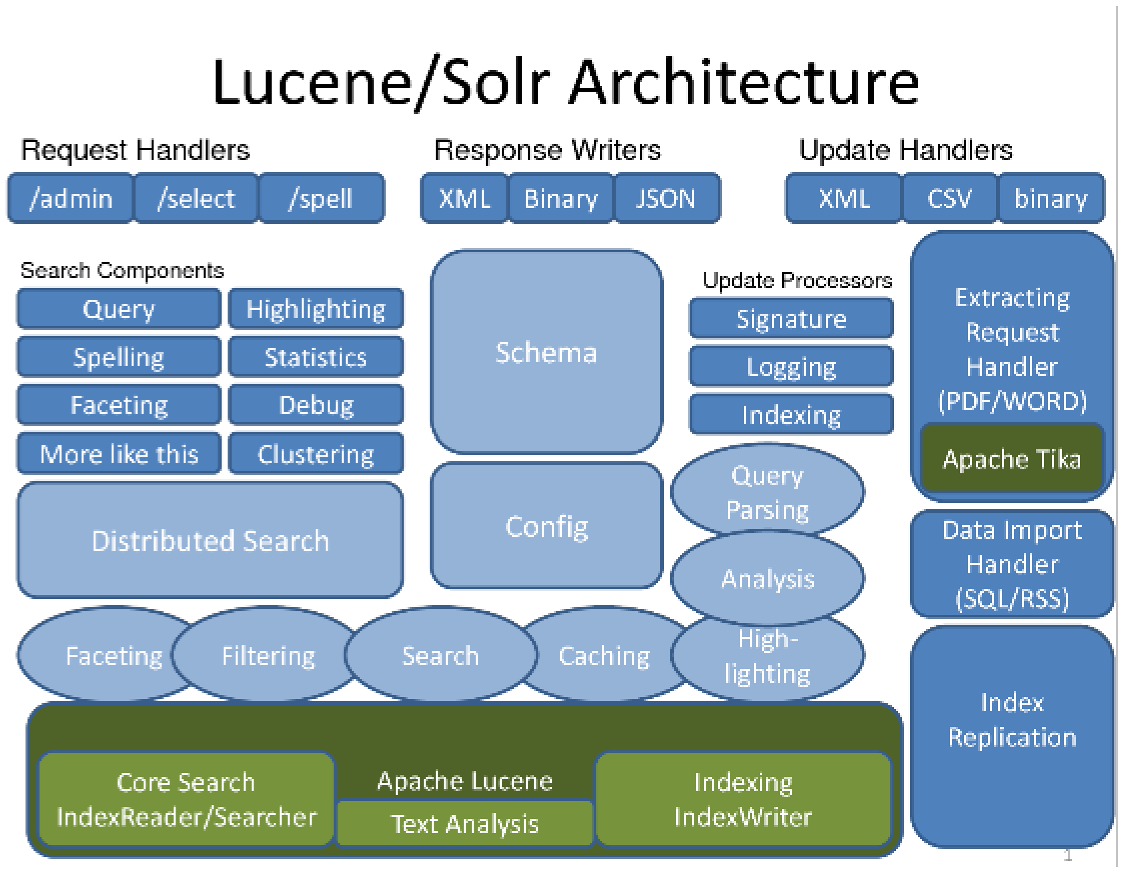


Fig 11. Lucene/Solr Architecture

Request Handlers intercepts all of the incoming requests coming over the https. By default, a lot of request handlers are already configured and available in the architecture. You can however plug-in many handlers in addition to those that are available. For example, in the above figure, **admin** request handler intercepts all of the admin requests. **Select** handler is similar to the servlet and it intercepts different url contexts. The **spell** handler takes care of the spell checking aspect. Whenever the user searches, the input query can be handled by any of the request handler

After interception by the request handlers, the request are then diverted to the **Search Components.** There are different search components, each one taking care of varied functions.

For example, each component handles different core functions such as query, spellings, faceting, highlighting, faceting, statistics, debugging, clustering, and more like this.

The next step is the **distributed search** unit that searches for information and returns it back to the user.

In parallel to these units, there are the **Config** and the **Schema** units in the architecture.

The **Request Writers** and the **Update Handlers** play a different role from the **Request Handlers.** They handle the Indexing process. They update the index. For example, the XML handler processes the submitted xml content. The Jason request writer processes the Jason format of the document. The Database Import request writer imports data directly from the database tables and writes it to the Index.

Solr History

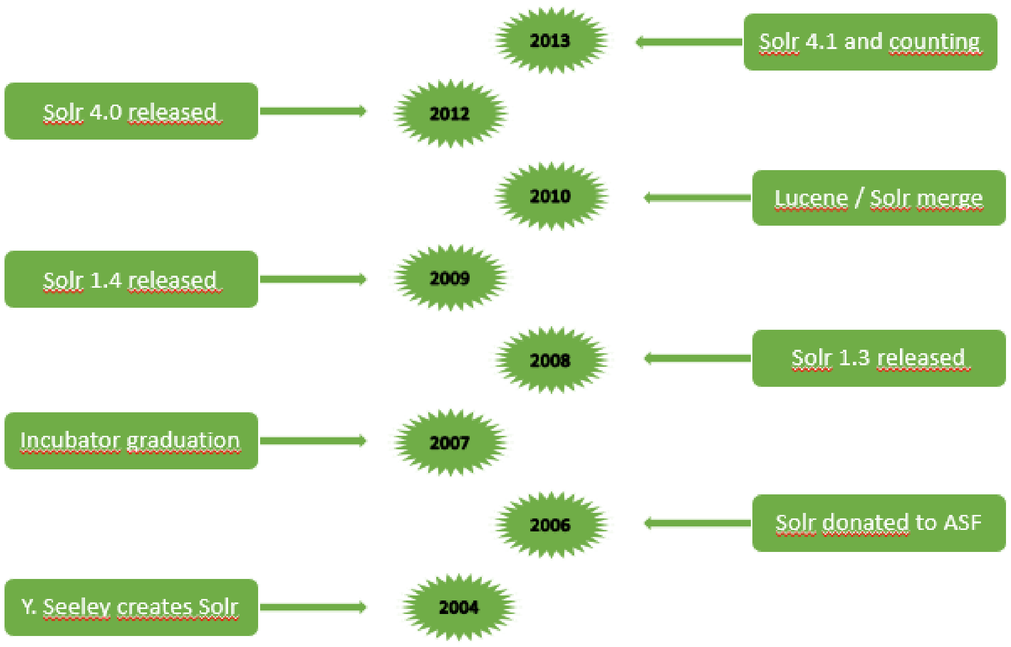


Fig 12. Solr History

Solr introduced in 2004 and it was introduced by Yonik Seelay at CNET Networks.

The Initial version of Solr used Master Slave architecture wherein there is a master for performing all of the write operations and there is a reader to perform all of the read operations. This architecture posed lot of challenges: you have to use all of your read and write operations in a stagnated manner. It can be used only for limited data sets and it is not capable for scaling to huge data sets. The newer versions of Solr was redesigned to handle huge data sets.

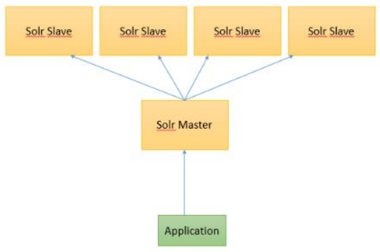


Fig 13. Solr Master Slave Architecture

Getting started with Solr

The example home directory of Solr is example/Solr.

Solr's home directory includes the following:

i. *bin: Files for more advanced setup are placed here.*

ii. *conf: Contains files which help set the Solr configurations.*

iii. *conf/schema.xml: This is the schema for the index including field type definitions for given dataset.*

iv. *conf/solrconfig.xml: This is the primary Solr configuration file.*

v. *data: It contains the actual Lucene index data in binary format.*

vi. *lib: The additional Solr plug in jar files can be placed here.*

The following are the functions that you can execute in Solr:

- Deploying and running Solr

- Adding/Updating/Deleting Data
- Indexing Data in Solr
- Querying data
- Analyzing text
- Other features include distributed search, numeric field statistics, search result clustering, function queries, boosting, and more like this.

The major features of Solr include:

- Full text search:
- Hit highlighting:
- Faceted search:
- Dynamic clustering:
- Database integration:
- Rich document handling

  The other features of Solr include Structures and textual search, HTML Administrative Interface, Extensible through plugins, Caching, Field Collapsing, New AutoSuggest component, More function queries, Geo Spatial Search, Flexible relevance – boost through function queries, Distributes search through Sharding, Embedded in Java Application, Automated management of large clusters through zookeeepers, Search Results clustering based on Carrot 2, Support over various other output formats over HTTP, Loose schema to define types and fields, Mature product search for public sites**.**

SOLR in Web Architecture

The following diagram show Solr Search engine and also depict the search process.

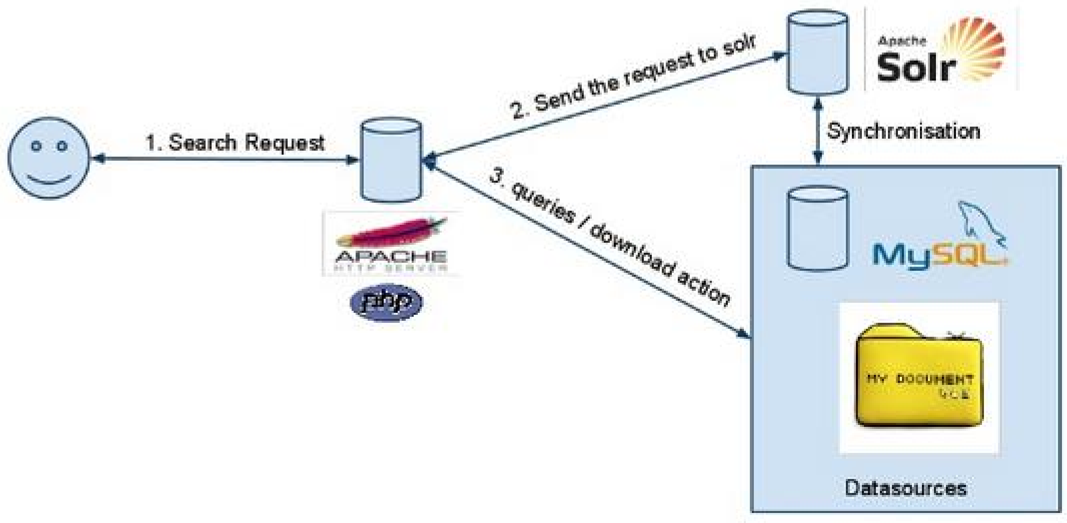


Fig 14. Solr in Web Architecture

Enterprise Search Architecture:

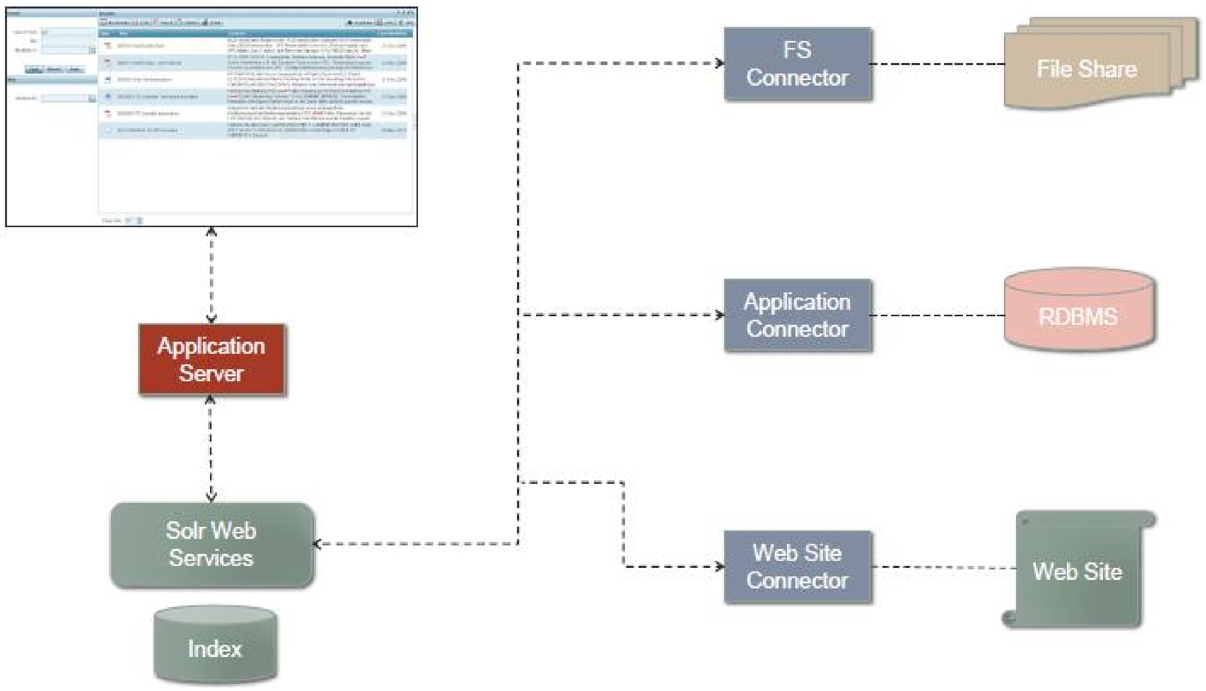
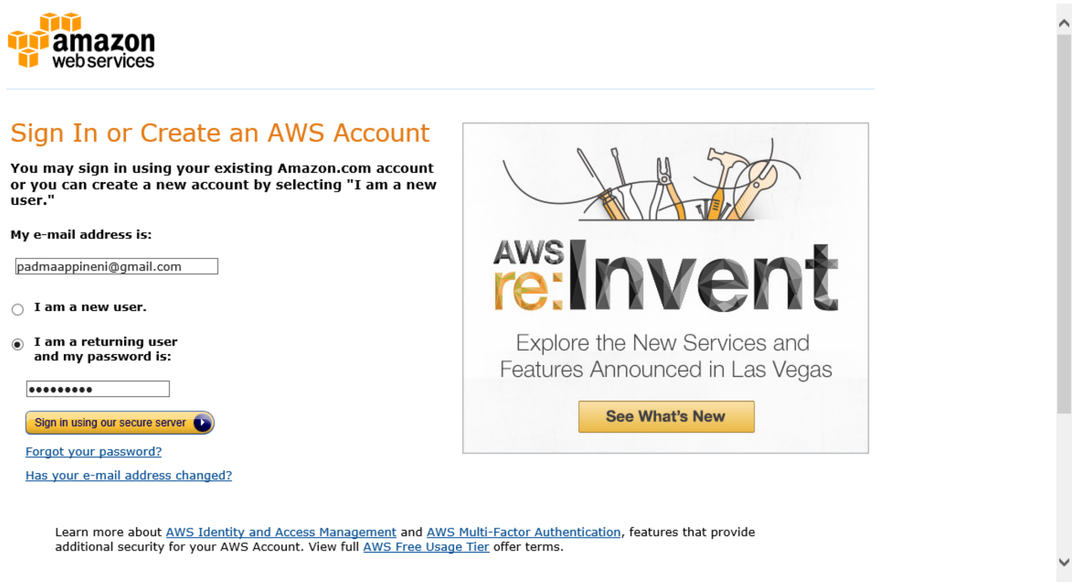


Fig 15. Enterprise Search Architecture

CHAPTER 3: GETTING STARTED

VIEWING INSTANCES IN SERVER

To view an instance in server, first log in to Amazon AWS EC2.



Sign In or Create an AWS Account

You may sign in using your existing Amazon.com account or you can create a new account by selecting "I am a new user."

My e-mail address is: padmaappineni@gmail.com

I am a new user.

[● I am a returning user and my password is:] ●●●●●●●●●

Sign in using our secure server

Forgot your password?

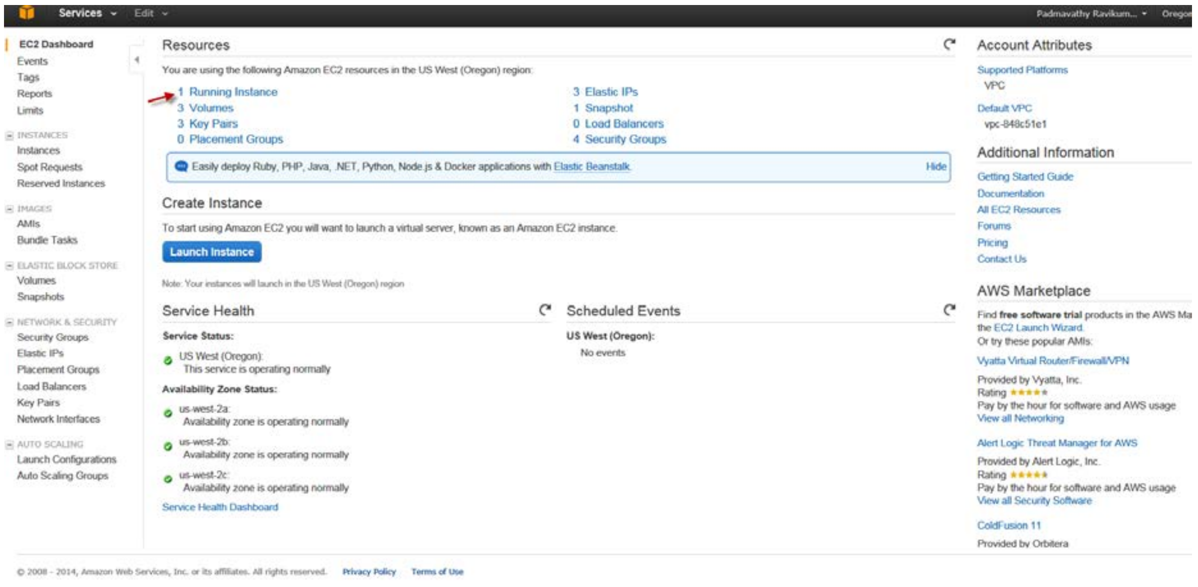
Has your e-mail address changed?

Learn more about AWS Identity and Access Management and AWS Multi-Factor Authentication, features that provide additional security for your AWS Account. View full AWS Free Usage Tier offer terms.

amazon web services
AWS re:Invent
Explore the New Services and Features Announced in Las Vegas
See What's New

Fig 16. Amazon AWS EC2

After login, click for Running RedHat 64-Bit Linux Instances for Amazon EC2 servers.



Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

1 Running Instance	3 Elastic IPs
3 Volumes	1 Snapshot
3 Key Pairs	0 Load Balancers
0 Placement Groups	4 Security Groups

Easily deploy Ruby, PHP, Java, .NET, Python, Node.js & Docker applications with Elastic Beanstalk. Hide

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

US West (Oregon): This service is operating normally

Availability Zone Status:

us-west-2a: Availability zone is operating normally

us-west-2b: Availability zone is operating normally

us-west-2c: Availability zone is operating normally

Service Health Dashboard

Scheduled Events

US West (Oregon): No events

Account Attributes

Supported Platforms: VPC

Default VPC: vpc-848c51e1

Additional Information

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

AWS Marketplace

Find **free software trial** products in the AWS Ma the EC2 Launch Wizard. Or try these popular AMIs:

Vyatta Virtual Router/Firewall/VPN
Provided by Vyatta, Inc.
Rating ★★★★★
Pay by the hour for software and AWS usage
View all Networking

Alert Logic Threat Manager for AWS
Provided by Alert Logic, Inc.
Rating ★★★★★
Pay by the hour for software and AWS usage
View all Security Software

ColdFusion 11
Provided by Orbitera

© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Fig 17. View Instance

The AWS instance named RedHat_Solr1 has already been created.

Instance Name: RedHat_Solr1

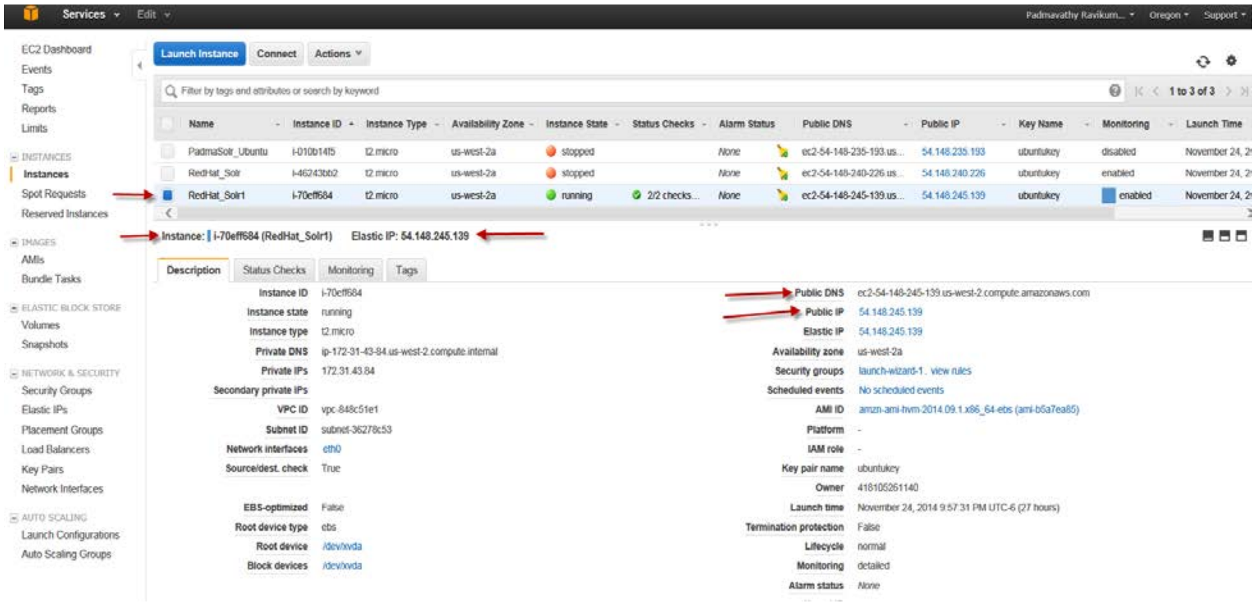
Public IP: 54.148.245.139

Similarly, the following instances have also been created:

Instance Name	**Public IP**
RedHat_Solr1	**54.148.245.139**
RedHat_Solr2	**54.69.244.228**
RedHat_Solr3	**54.148.240.226**

Table 3. Instances

You can also verify this in the server as follows:



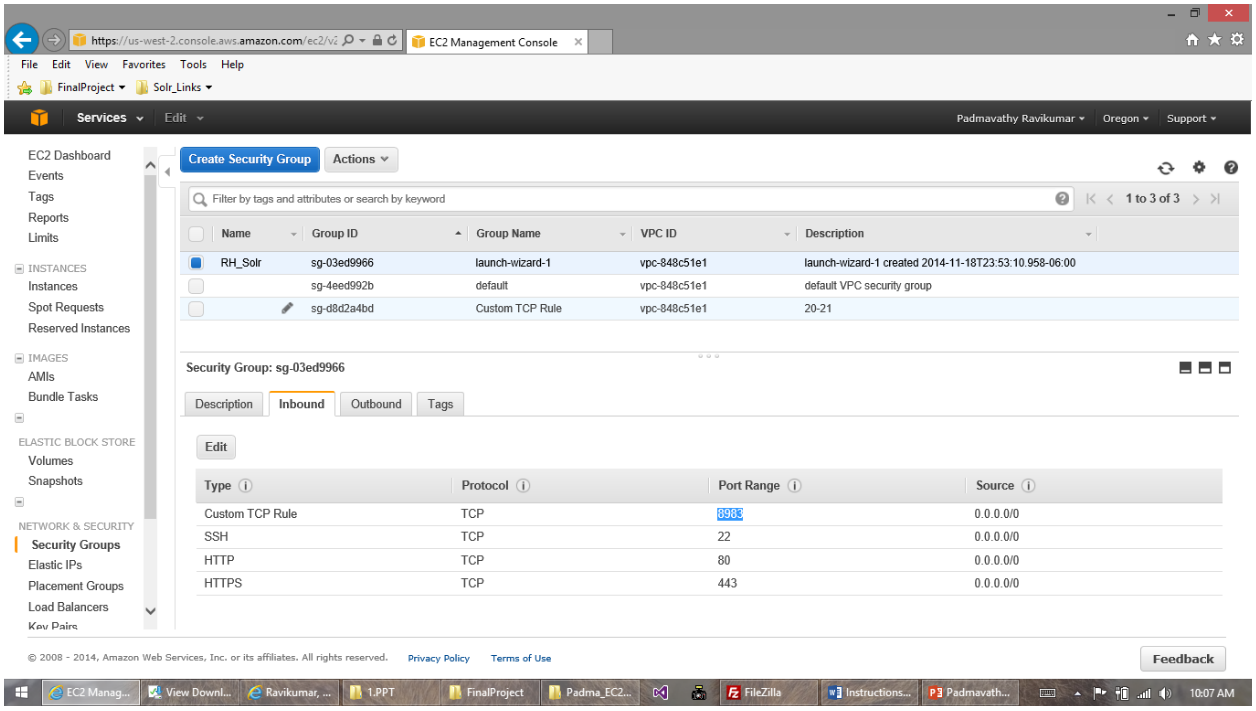
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name	Monitoring	Launch Time
PadmaSolr_Ubuntu	i-010b14f5	t2.micro	us-west-2a	stopped		None	ec2-54-148-235-193.us...	54.148.235.193	ubuntukey	disabled	November 24, 2
RedHat_Solr	i-46243bb2	t2.micro	us-west-2a	stopped		None	ec2-54-148-240-226.us...	54.148.240.226	ubuntukey	enabled	November 24, 2
RedHat_Solr1	i-70eff684	t2.micro	us-west-2a	running	2/2 checks...	None	ec2-54-148-245-139.us...	54.148.245.139	ubuntukey	enabled	November 24, 2

Instance: i-70eff684 (RedHat_Solr1) Elastic IP: 54.148.245.139

Instance ID	i-70eff684	Public DNS	ec2-54-148-245-139.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	54.148.245.139
Instance type	t2.micro	Elastic IP	54.148.245.139
Private DNS	ip-172-31-43-84.us-west-2.compute.internal	Availability zone	us-west-2a
Private IPs	172.31.43.84	Security groups	launch-wizard-1 . view rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-848c51e1	AMI ID	amzn-ami-hvm-2014.09.1.x86_64-ebs (ami-b5a7ea85)
Subnet ID	subnet-36278c53	Platform	-
Network interfaces	eth0	IAM role	-
Source/dest. check	True	Key pair name	ubuntukey
		Owner	418105261140
EBS-optimized	False	Launch time	November 24, 2014 9:57:31 PM UTC-6 (27 hours)
Root device type	ebs	Termination protection	False
Root device	/dev/xvda	Lifecycle	normal
Block devices	/dev/xvda	Monitoring	detailed
		Alarm status	None

Fig 18. Instance Properties

Make sure to add the Custom TCP Rule Port 8983 in Security Group to access SOLR Dashboard.



Name	Group ID	Group Name	VPC ID	Description
RH_Solr	sg-03ed9966	launch-wizard-1	vpc-848c51e1	launch-wizard-1 created 2014-11-18T23:53:10.958-06:00
	sg-4eed992b	default	vpc-848c51e1	default VPC security group
	sg-d8d2a4bd	Custom TCP Rule	vpc-848c51e1	20-21

Security Group: sg-03ed9966

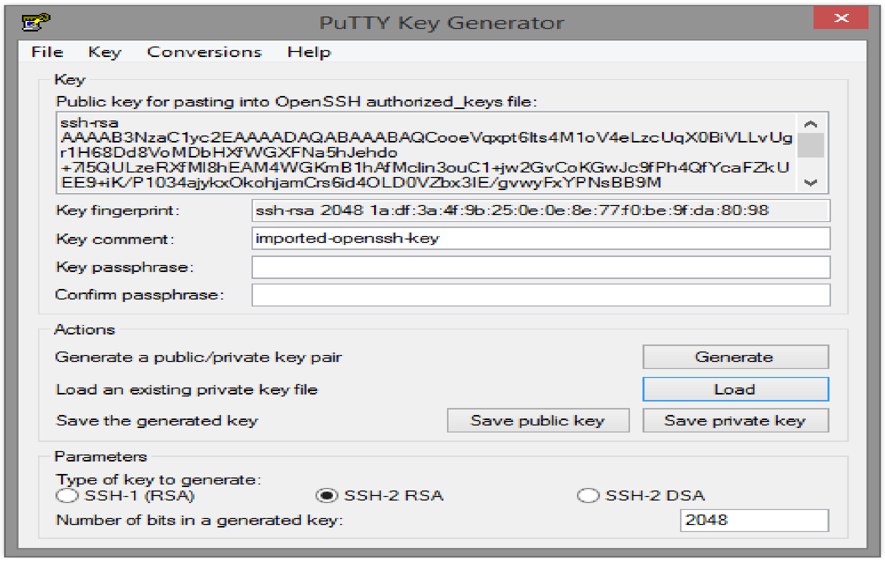
Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	8983	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

Fig 19. RedHat_Solr1

After you verify the AWS instance is running you need to login to the RedHat_Solr1 instance using Putty software or Cygwin tool or openssh or Apple Computer.

AWS login instructions to the server

- Amazon AWS allows you to create a "Security Key Pair" while setup a Linux instance.
- I am attaching the security key pair in this document and also in this email. Please download this package "ubuntukey.pem" in to your computer. This will help you to login to AWS instance using putty.



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCooeVqxpt6lts4M1oV4eLzcUqX0BiVLLvUg
r1H68Dd8VoMDbHXfWGXFNa5hJehdo
+7l5QULzeRXfMI8hEAM4WGKmB1hAfMclin3ouC1+jw2GvCoKGwJc9fPh4QfYcaFZkU
EE9+iK/P1034ajykxOkohjamCrs6id4OLD0VZbx3IE/gvwyFxYPNsBB9M
```

Key fingerprint: ssh-rsa 2048 1a:df:3a:4f:9b:25:0e:0e:8e:77:f0:be:9f:da:80:98

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

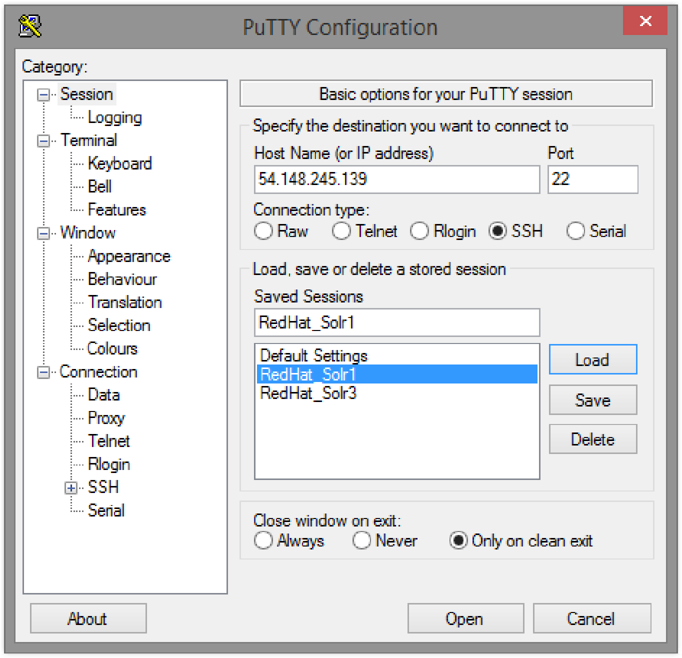
Parameters

Type of key to generate: ( ) SSH-1 (RSA) [✓ SSH-2 RSA] ( ) SSH-2 DSA

Number of bits in a generated key: 2048

Fig 20. Putty Key Generator

Next, you can use the Putty as follows to load and log into any one of the instances that you created.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 54.148.245.139 Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Load, save or delete a stored session

Saved Sessions: RedHat_Solr1

Default Settings
RedHat_Solr1
RedHat_Solr3

Close window on exit: ( ) Always ( ) Never [✓ Only on clean exit]

Fig 21. Putty Configuration

ACCESSING INSTANCES

- You can also login in the following method. Now you can login to AWS instance from three ways (Cygwin or Putty in Windows or MAC)

    i. Open terminal window in Apple/ Cygwin in Windows PC

    ii. Go to the folder path where you saved ubuntukey.pem key

    iii. Use the **chmod** command to make sure your private key file isn't publicly viewable. ***chmod 400 ubuntukey.pem***

    iv. Run this command ***ssh -i ubuntukey.pem ec2-user@54.148.245.139*** on the terminal window (Please see the below screenshots in cygwin)

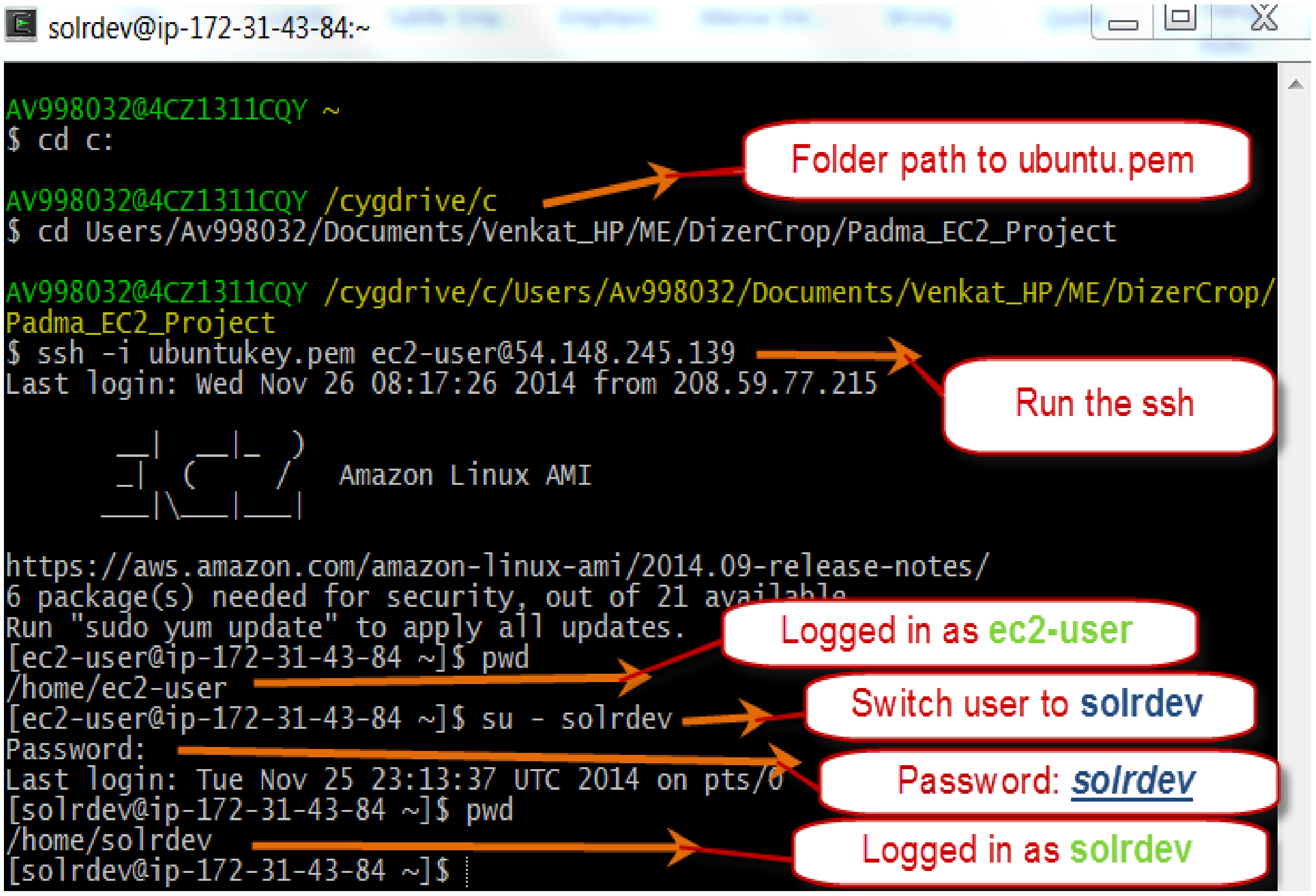
    v. Now you logged in as **ec2-user**

    vi. Switch user to ***solrdev***

    Type command: ***su – solrdev***

    Password: ***solrdev***

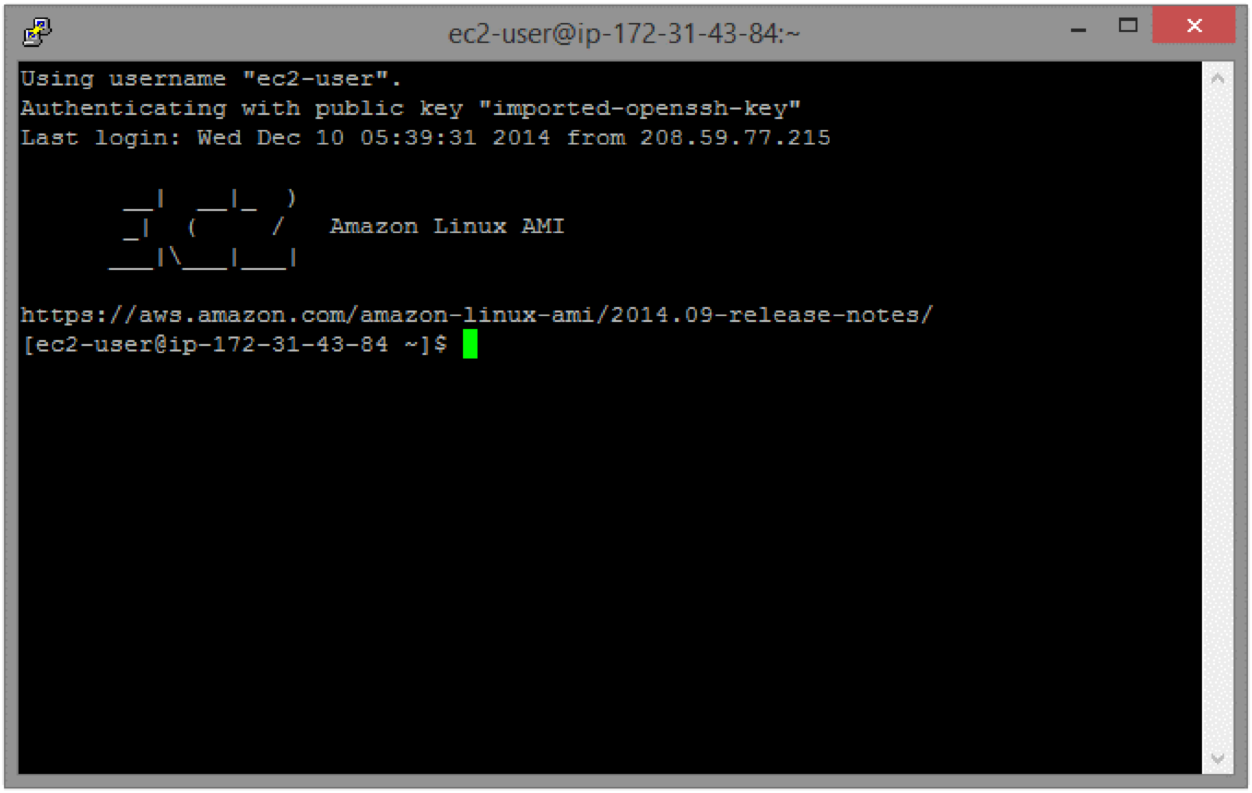
    vii. Now you logged in as **solrdev**



```
solrdev@ip-172-31-43-84:~

AV998032@4CZ1311CQY ~
$ cd c:

AV998032@4CZ1311CQY /cygdrive/c
$ cd Users/Av998032/Documents/Venkat_HP/ME/DizerCrop/Padma_EC2_Project

AV998032@4CZ1311CQY /cygdrive/c/Users/Av998032/Documents/Venkat_HP/ME/DizerCrop/
Padma_EC2_Project
$ ssh -i ubuntukey.pem ec2-user@54.148.245.139
Last login: Wed Nov 26 08:17:26 2014 from 208.59.77.215

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
6 package(s) needed for security, out of 21 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-43-84 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-43-84 ~]$ su - solrdev
Password:
Last login: Tue Nov 25 23:13:37 UTC 2014 on pts/0
[solrdev@ip-172-31-43-84 ~]$ pwd
/home/solrdev
[solrdev@ip-172-31-43-84 ~]$
```

Fig 22. Terminal Window 1

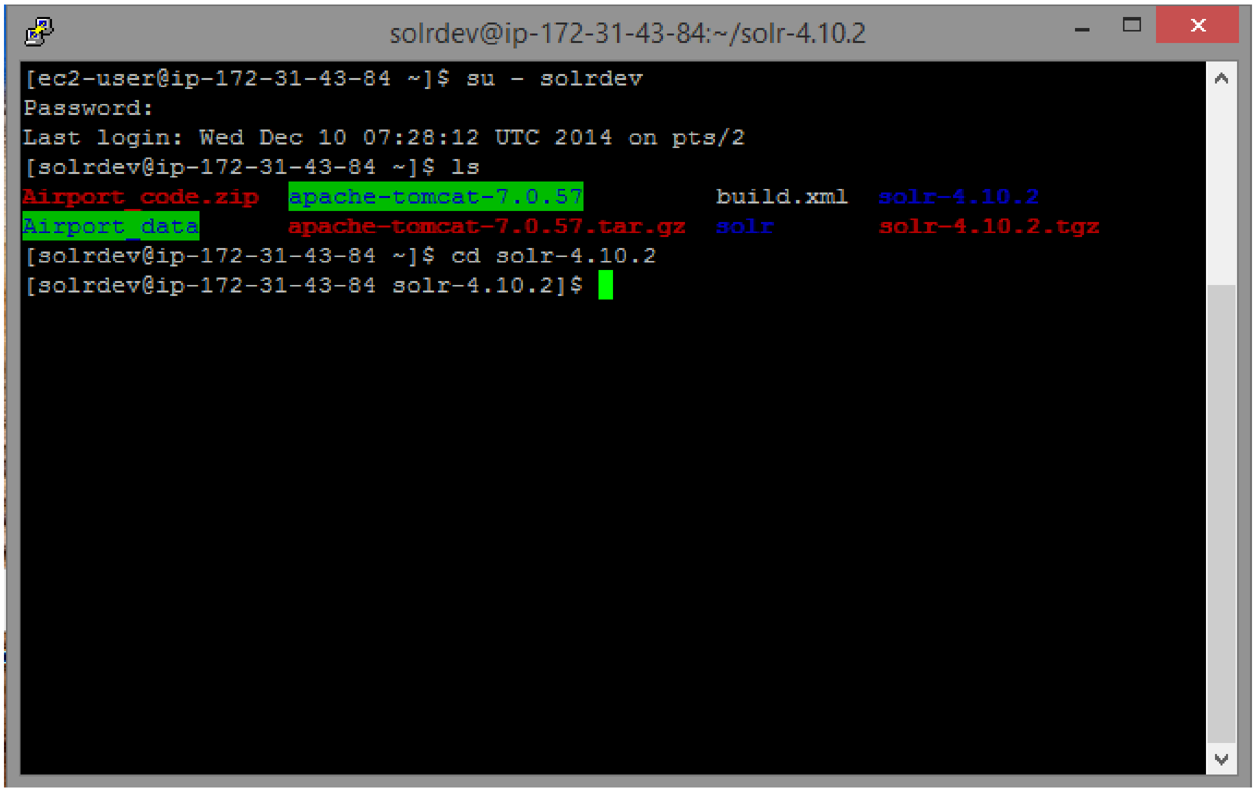
Assuming that we used putty, we get to our login window which is shown below:



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Wed Dec 10 05:39:31 2014 from 208.59.77.215

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
[ec2-user@ip-172-31-43-84 ~]$
```

Fig 23. Terminal Window 2

To clear the screen, login as solrdev, list the directories present in the logged in user, and finally access the users' folder; we execute the following set of commands and the resultants window appears as shown below.

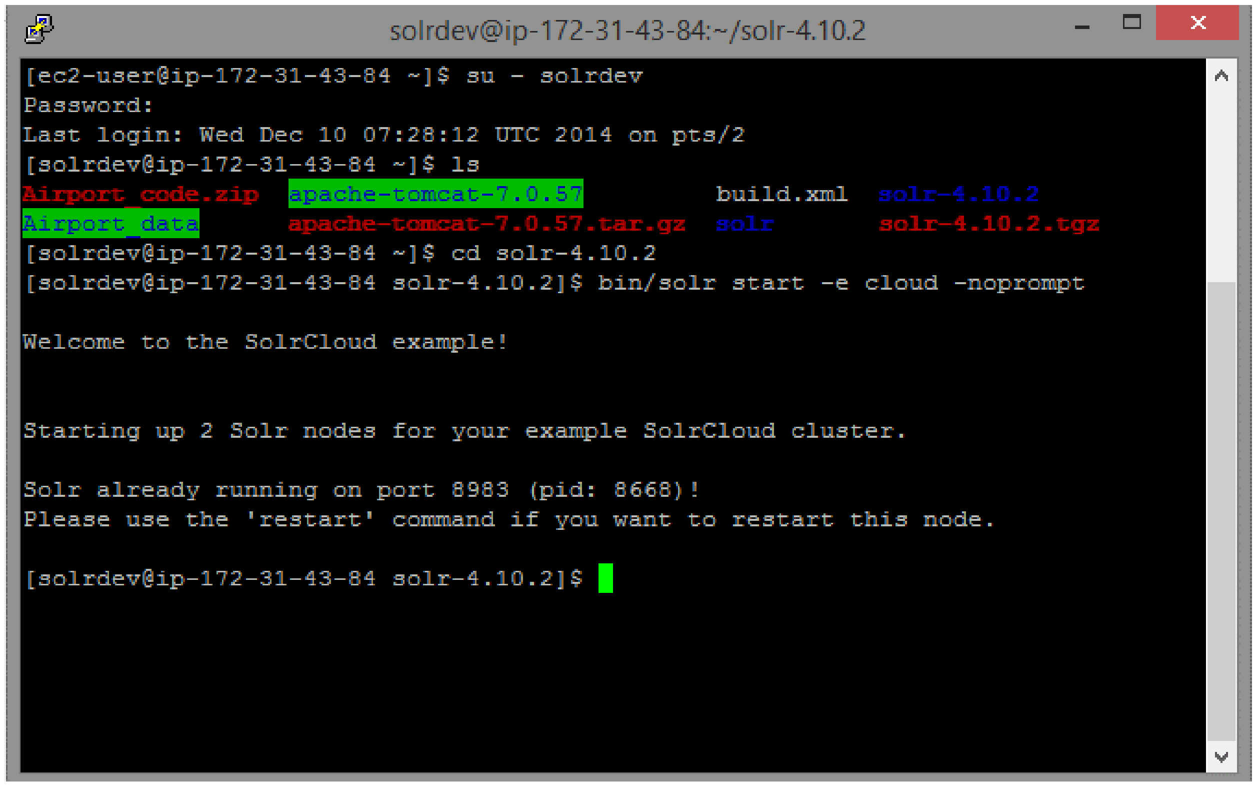


```
[ec2-user@ip-172-31-43-84 ~]$ su - solrdev
Password:
Last login: Wed Dec 10 07:28:12 UTC 2014 on pts/2
[solrdev@ip-172-31-43-84 ~]$ ls
Airport_code.zip  apache-tomcat-7.0.57         build.xml  solr-4.10.2
Airport_data      apache-tomcat-7.0.57.tar.gz  solr       solr-4.10.2.tgz
[solrdev@ip-172-31-43-84 ~]$ cd solr-4.10.2
[solrdev@ip-172-31-43-84 solr-4.10.2]$
```

Fig 24. File Access

To launch the Solr cloud, run the command: bin/Solr start -e cloud –noprompt

The resultant window appears as shown below:



```
[ec2-user@ip-172-31-43-84 ~]$ su - solrdev
Password:
Last login: Wed Dec 10 07:28:12 UTC 2014 on pts/2
[solrdev@ip-172-31-43-84 ~]$ ls
Airport_code.zip  apache-tomcat-7.0.57         build.xml  solr-4.10.2
Airport_data      apache-tomcat-7.0.57.tar.gz  solr       solr-4.10.2.tgz
[solrdev@ip-172-31-43-84 ~]$ cd solr-4.10.2
[solrdev@ip-172-31-43-84 solr-4.10.2]$ bin/solr start -e cloud -noprompt

Welcome to the SolrCloud example!


Starting up 2 Solr nodes for your example SolrCloud cluster.

Solr already running on port 8983 (pid: 8668)!
Please use the 'restart' command if you want to restart this node.

[solrdev@ip-172-31-43-84 solr-4.10.2]$
```

Fig 25. Start Process

INDEXING DATA

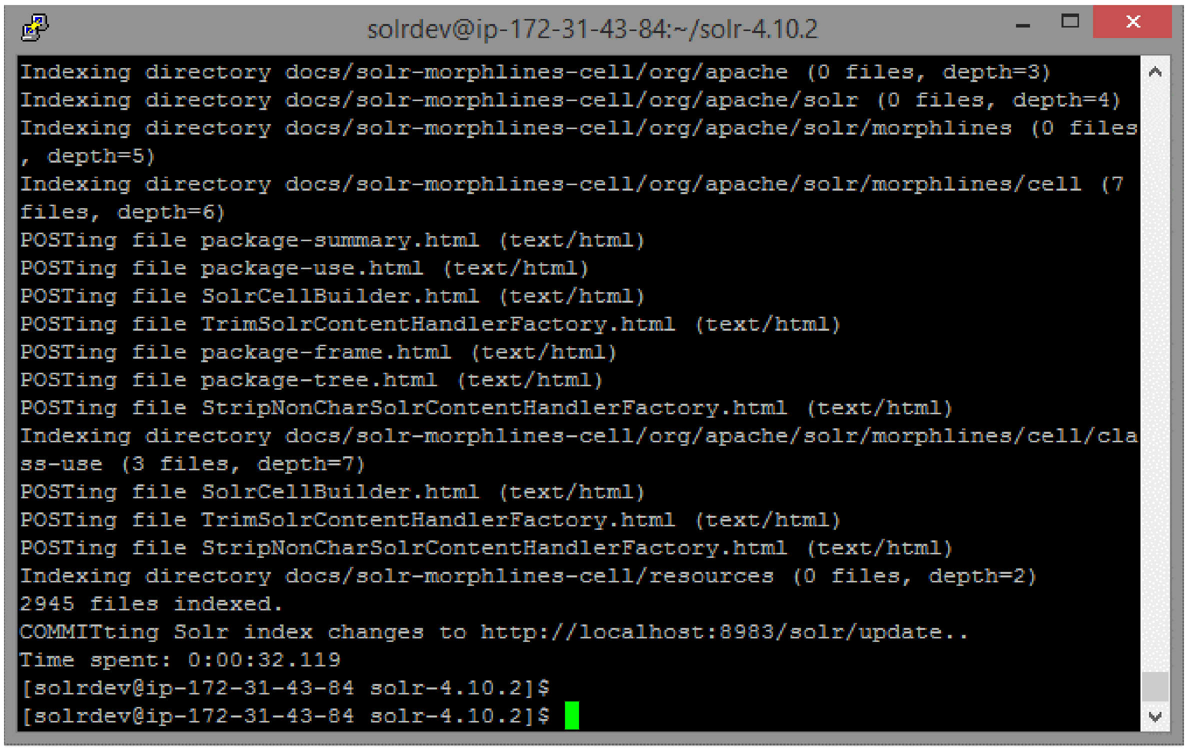
We can index files of different formats using the following commands. To do all of the indexing, we must first execute the command export CLASSPATH=dist/Solr-core-4.10.2.jar

File Format	**Command**
Java File	java -Dauto -Drecursive org.apache.Solr.util.SimplePostTool docs/
Solar XML	java org.apache.Solr.util.SimplePostTool example/exampledocs/*.xml

Jason	java -Dauto org.apache.Solr.util.SimplePostTool example/exampledocs/books.json
CSV	java -Dauto org.apache.Solr.util.SimplePostTool example/exampledocs/books.csv

Table 4. Commands for Data Indexing

The resultant window at the terminal appears as shown below:

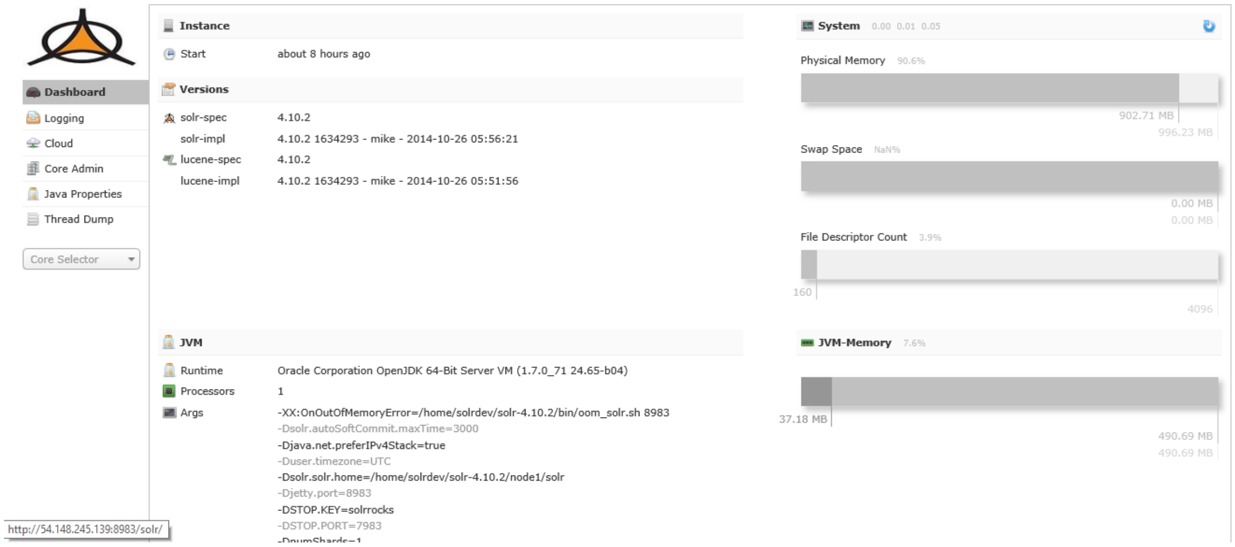


```
Indexing directory docs/solr-morphlines-cell/org/apache (0 files, depth=3)
Indexing directory docs/solr-morphlines-cell/org/apache/solr (0 files, depth=4)
Indexing directory docs/solr-morphlines-cell/org/apache/solr/morphlines (0 files
, depth=5)
Indexing directory docs/solr-morphlines-cell/org/apache/solr/morphlines/cell (7
files, depth=6)
POSTing file package-summary.html (text/html)
POSTing file package-use.html (text/html)
POSTing file SolrCellBuilder.html (text/html)
POSTing file TrimSolrContentHandlerFactory.html (text/html)
POSTing file package-frame.html (text/html)
POSTing file package-tree.html (text/html)
POSTing file StripNonCharSolrContentHandlerFactory.html (text/html)
Indexing directory docs/solr-morphlines-cell/org/apache/solr/morphlines/cell/cla
ss-use (3 files, depth=7)
POSTing file SolrCellBuilder.html (text/html)
POSTing file TrimSolrContentHandlerFactory.html (text/html)
POSTing file StripNonCharSolrContentHandlerFactory.html (text/html)
Indexing directory docs/solr-morphlines-cell/resources (0 files, depth=2)
2945 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/update..
Time spent: 0:00:32.119
[solrdev@ip-172-31-43-84 solr-4.10.2]$
[solrdev@ip-172-31-43-84 solr-4.10.2]$
```

Fig 26. Index Data

SEARCHING DATA

To search for indexed data, we first launch the Solr dashboard. The Solr dashboard appears as below:



Instance

Start about 8 hours ago

Versions

```
solr-spec     4.10.2
solr-impl     4.10.2 1634293 - mike - 2014-10-26 05:56:21
lucene-spec   4.10.2
lucene-impl   4.10.2 1634293 - mike - 2014-10-26 05:51:56
```

JVM

```
Runtime      Oracle Corporation OpenJDK 64-Bit Server VM (1.7.0_71 24.65-b04)
Processors   1
Args         -XX:OnOutOfMemoryError=/home/solrdev/solr-4.10.2/bin/oom_solr.sh 8983
             -Dsolr.autoSoftCommit.maxTime=3000
             -Djava.net.preferIPv4Stack=true
             -Duser.timezone=UTC
             -Dsolr.solr.home=/home/solrdev/solr-4.10.2/node1/solr
             -Djetty.port=8983
             -DSTOP.KEY=solrrocks
             -DSTOP.PORT=7983
             -DnumShards=1
```

System 0.00 0.01 0.05

Physical Memory 90.6% — 902.71 MB / 996.23 MB

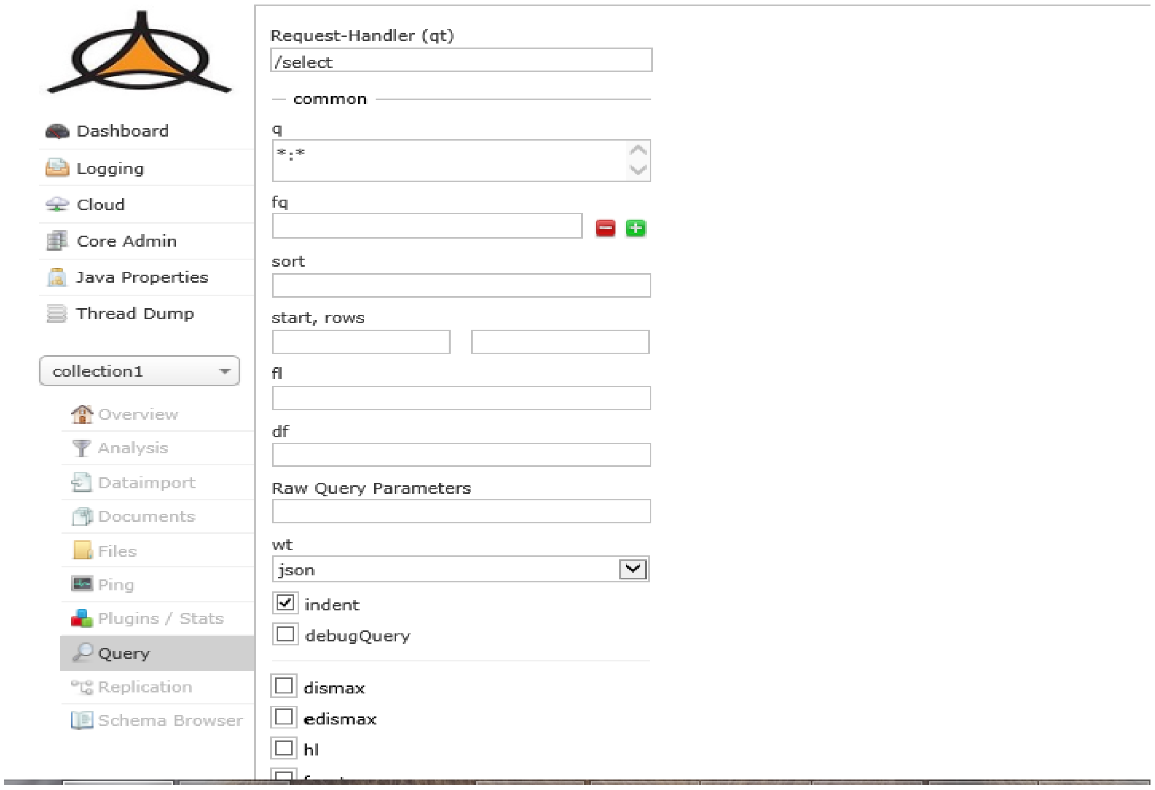
Swap Space NaN% — 0.00 MB / 0.00 MB

File Descriptor Count 3.9% — 160 / 4096

JVM-Memory 7.6% — 37.18 MB / 490.69 MB / 490.69 MB

Fig 27. Solr Dashboard

On the right side of the dashboard, click **Core Selector** and select **Collection 1** from the drop-down menu. After that, click **Query.** The screen appears as shown below:



Request-Handler (qt): /select

q: *:*

wt: json

[✓] indent

[ ] debugQuery

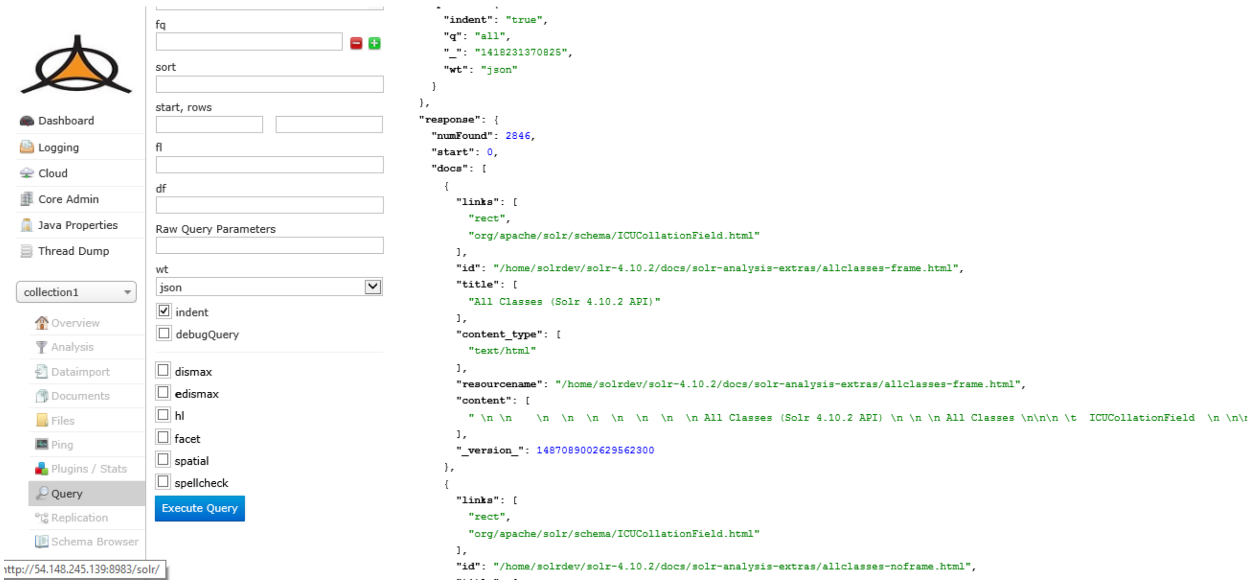
[ ] dismax

[ ] edismax

[ ] hl

Fig 28. Solr Dashboard – Execute Query

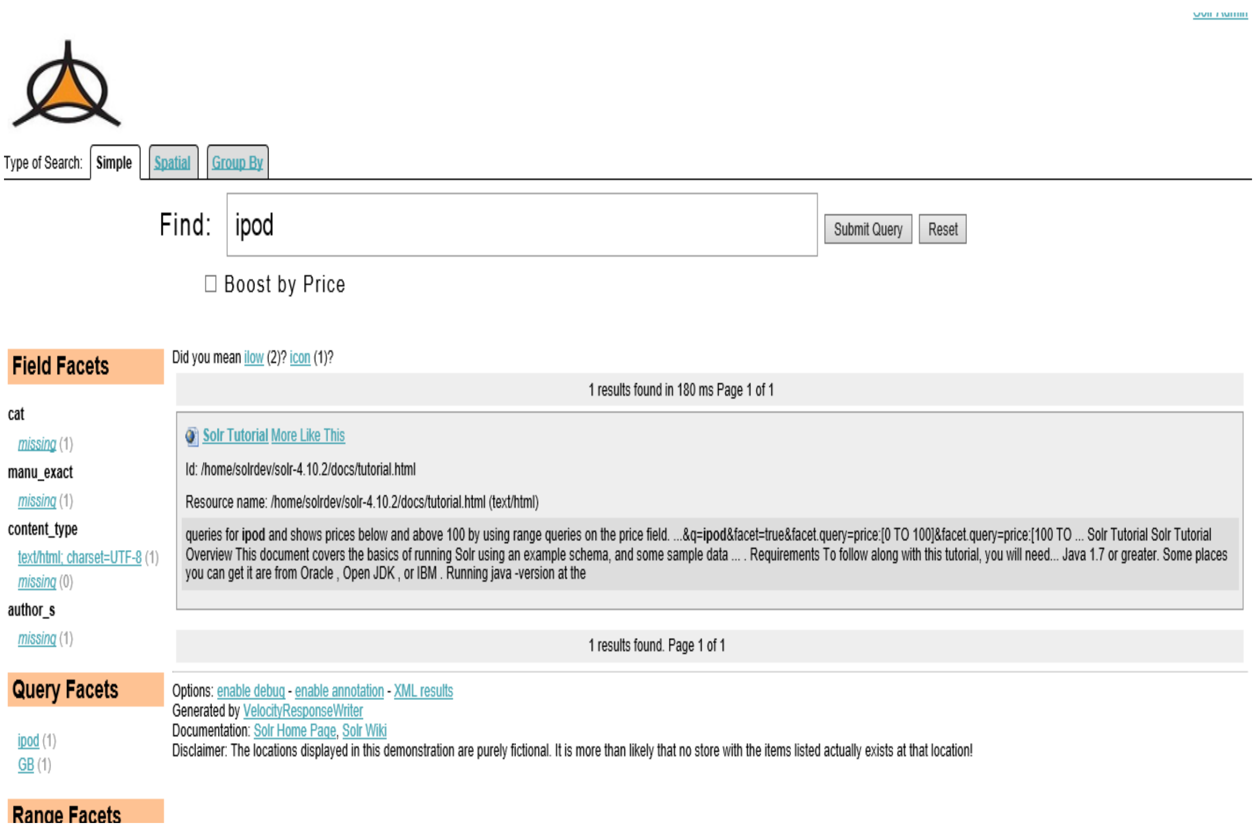
Enter the search terms in the query box and then click on the Execute Query button at the end of the screen. The resultant screen appears as below:



```
      "indent": "true",
      "q": "all",
      "_": "1418231370825",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 2846,
    "start": 0,
    "docs": [
      {
        "links": [
          "rect",
          "org/apache/solr/schema/ICUCollationField.html"
        ],
        "id": "/home/solrdev/solr-4.10.2/docs/solr-analysis-extras/allclasses-frame.html",
        "title": [
          "All Classes (Solr 4.10.2 API)"
        ],
        "content_type": [
          "text/html"
        ],
        "resourcename": "/home/solrdev/solr-4.10.2/docs/solr-analysis-extras/allclasses-frame.html",
        "content": [
          " \n \n    \n  \n  \n  \n  \n  \n  \n All Classes (Solr 4.10.2 API) \n \n \n All Classes \n\n\n \t  ICUCollationField  \n \n\
        ],
        "_version_": 1487089002629562300
      },
      {
        "links": [
          "rect",
          "org/apache/solr/schema/ICUCollationField.html"
        ],
        "id": "/home/solrdev/solr-4.10.2/docs/solr-analysis-extras/allclasses-noframe.html",
```

Fig 29. Solr Dashboard – Query Results

You can also open the screen in browse mode and make queries and get results as shown in the screen shown below.



Did you mean ilow (2)? icon (1)?

1 results found in 180 ms Page 1 of 1

Solr Tutorial More Like This

Id: /home/solrdev/solr-4.10.2/docs/tutorial.html

Resource name: /home/solrdev/solr-4.10.2/docs/tutorial.html (text/html)

queries for **ipod** and shows prices below and above 100 by using range queries on the price field. ...&q=**ipod**&facet=true&facet.query=price:[0 TO 100]&facet.query=price:[100 TO ... Solr Tutorial Solr Tutorial Overview This document covers the basics of running Solr using an example schema, and some sample data ... . Requirements To follow along with this tutorial, you will need... Java 1.7 or greater. Some places you can get it are from Oracle , Open JDK , or IBM . Running java -version at the

1 results found. Page 1 of 1

Options: enable debug - enable annotation - XML results
Generated by VelocityResponseWriter
Documentation: Solr Home Page, Solr Wiki
Disclaimer: The locations displayed in this demonstration are purely fictional. It is more than likely that no store with the items listed actually exists at that location!

Fig 30. Solr Dashboard – Browse

EXAMPLE 1

AIRPORT APPLICATION

Now that you have seen the Solr features, you will now see a modern data application that uses Solr. Airport websites often consists of so much data about flights. Their arrivals, departures, cancellation, delays, and so much more information. All of this information is never constant and keeps changing and getting updated constantly. Also, this is the kind of information that also constantly gets queried and searched.

Solr can deal with both sparse as well ambiguous data which are the basis of all modern applications today. This facilitates the concept of leveraging a search engine to go beyond search. Not only that Lucene and Solr can crunch numbers, answer complex geospatial questions, support several types of joins and grouping options, column-oriented storage, special ways to deal with enumerated and numerical data types, and do many more. They also enable you to define your own complex data types, storage, ranking and do other analytical functions.

The following example shows demonstrates how Solr search engine can go beyond search.

- The application first queries the data
- The application then processes the data with D3 JavaScript Library
- The data sets belong to Research and Innovative Technology Administration (RITA) of the U.S. Department of Transportation's Bureau of Transportation Statistics and from OpenFlights.
- The data includes details such as originating airport, destination airport, time delays causes of delays, and airline information for all flights in a particular time period.

- You can use the data to analyze delays between particular airports, traffic growth at specific airports, and much more.

Starting the application

To start, run the application and then look at some of its interfaces. As you go along, ensure that the application interacts with the data by interrogating Solr in different ways.

Requirements

- Lucene and Solr
- Java 6 or higher.
- A modern web browser such as Google Chrome and Firefox.
- 4GB of disk space depending on the flight data
- Terminal access with a `bash` (or similar) shell on *nix. For Windows, you need Cygwin. I only tested on OS X with the `bash` shell.
- `wget` if you choose to download the data by using the download script that's in the samplecode package. You can also download the flight data manually.
- Apache Ant 1.8+ for compilation and packaging purposes, if you want to run any of the Javacode examples.

Running the application

To run the application, you follow these steps:

1. Download the sample code zip file. Unzip it to the directory of your choice. Name the directory as $SOLR_AIR.
2. At the command prompt, type the following:

```
cd $SOLR_AIR
```

This command changes the path to the $SOLR_AIR

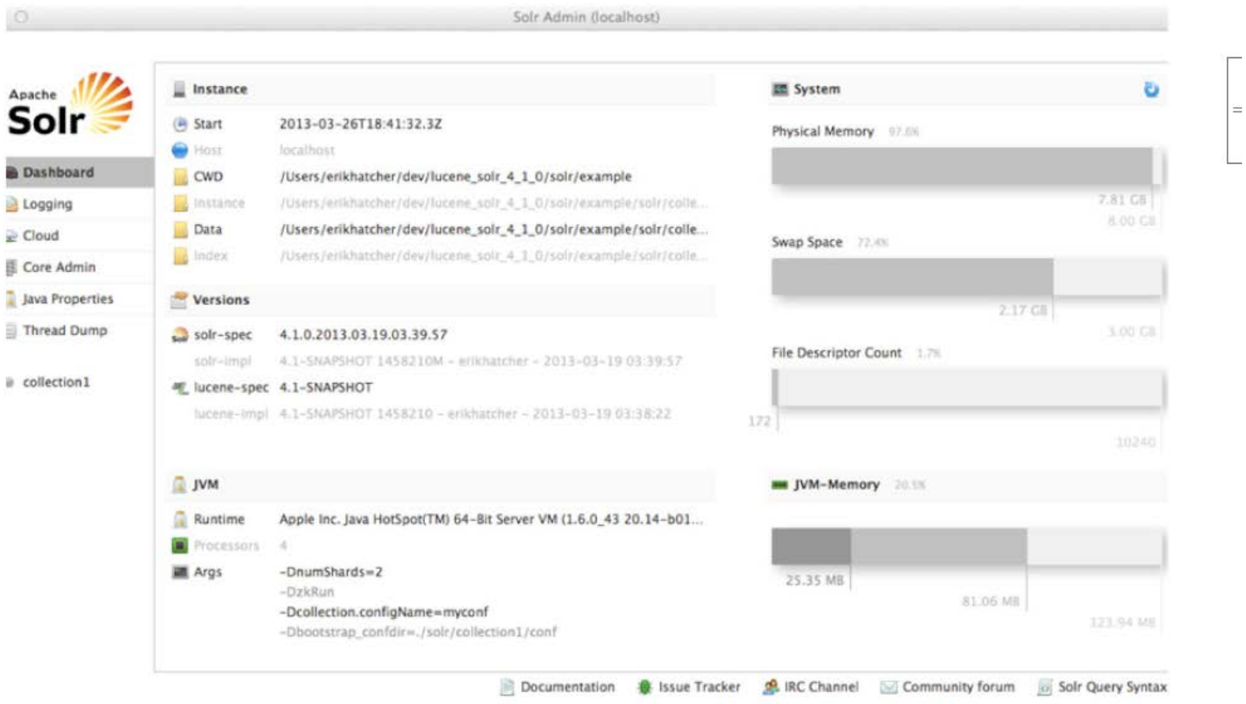
3. Start Solr

   ./bin/start-Solr.sh

4. Run the script that will create the needed fields to model the data.

   ./bin/setup.sh

5. Direct your browser to http://localhost:8983/Solr/#/ to open the new Solr Admin UI. The following window opens.



Instance

Start	2013-03-26T18:41:32.3Z
Host	localhost
CWD	/Users/erikhatcher/dev/lucene_solr_4_1_0/solr/example
Instance	/Users/erikhatcher/dev/lucene_solr_4_1_0/solr/example/solr/colle...
Data	/Users/erikhatcher/dev/lucene_solr_4_1_0/solr/example/solr/colle...
Index	/Users/erikhatcher/dev/lucene_solr_4_1_0/solr/example/solr/colle...

Versions

solr-spec	4.1.0.2013.03.19.03.39.57
solr-impl	4.1-SNAPSHOT 1458210M - erikhatcher - 2013-03-19 03:39:57
lucene-spec	4.1-SNAPSHOT
lucene-impl	4.1-SNAPSHOT 1458210 - erikhatcher - 2013-03-19 03:38:22

JVM

Runtime	Apple Inc. Java HotSpot(TM) 64-Bit Server VM (1.6.0_43 20.14-b01...
Processors	4
Args	-DnumShards=2 -DzkRun -Dcollection.configName=myconf -Dbootstrap_confdir=./solr/collection1/conf

System

Physical Memory 97.6% — 7.81 GB / 8.00 GB

Swap Space 72.4% — 2.17 GB / 3.00 GB

File Descriptor Count 1.7% — 172 / 10240

JVM-Memory 20.5% — 25.35 MB / 81.06 MB / 123.94 MB

Fig 30. Solr Admin UI

6. At the terminal, view contents of the script bin/download-data.sh for details about Open Flights and the data type to be downloaded from RITA. You can download data either manually or by using the script./bin/download-data.sh. The download might take time according to the data bandwidth.

7. After the data has been downloaded, it is time to index all or some data. To index all data, execute the following command:

   bin/index.sh

8. To index some data, execute the following command:

   bin/index.sh 1987

9. Indexing might take some time depending on the data requirements. Meanwhile, open the url http://localhost:8983/Solr/collection1/travel. The User Interface appears as shown below.

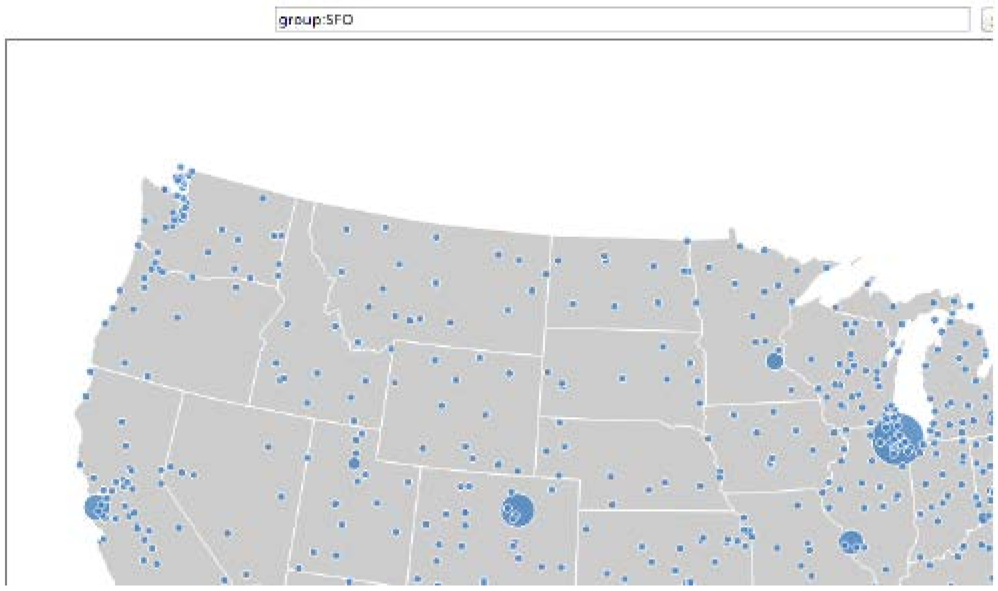


Fig 31. User Interface

Exploring Data

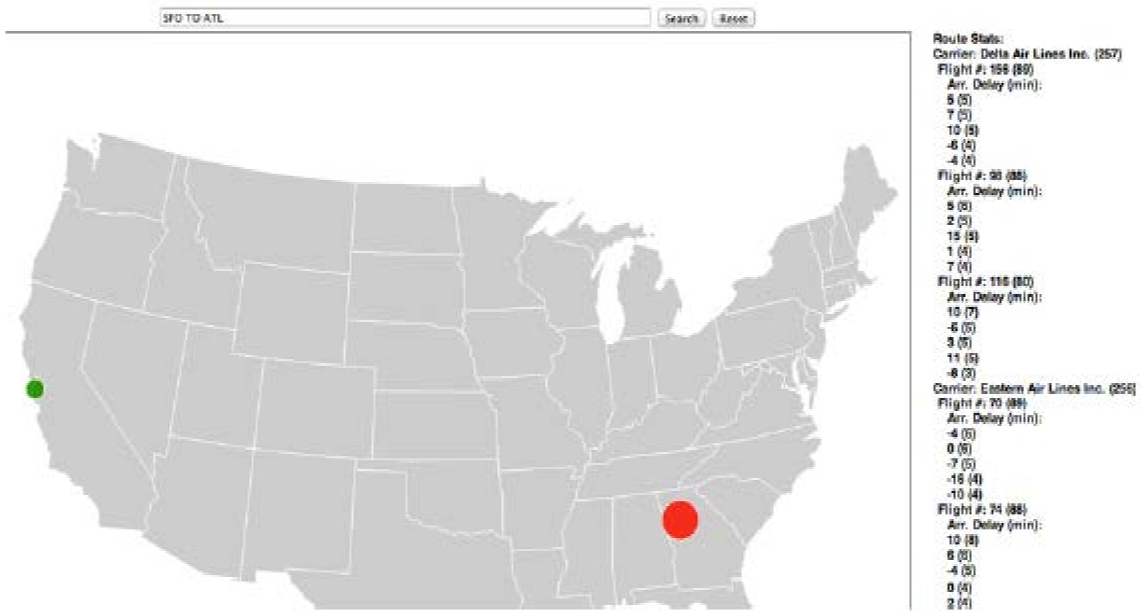
- With the running Solr application and the UI, you can determine the type of questions that you want to probe.
- The browser has two main components: **map** and **search box**.
- The map shows all data or airport information from Solr and not just restricted from the example CSV file. The map also facilities mouse hover-over. That is when the user hovers over any airport on the map, the information about the airport is displayed.
- The search box is designed with functionalities that helps with the sophisticated search and analytics application.
- The key focus areas of search and analysis is:
  - ✓ Pivot Facets
  - ✓ Statistical Functionality
  - ✓ Grouping
  - ✓ Solr's expanded Geospatial support
- All of the above functionalities can facilitate provision of the following information:
  - ✓ Average delay of arriving airplanes at a specific airport
  - ✓ the most common delay times for an aircraft that's flying between two airports
  - ✓ The most common delay times for an aircraft that's flying between two airports per airline or between a certain starting airport and all of the nearby airports).
- The application uses statistical functionality and longstanding faceting capability of Solr to draw initial map of airport dots that will be needed for generation basic information such as total flights, average, maximum, and minimum delay times. This capability can be sufficient to find big data or extreme outliners.

- To facilitate all of the above, there is also some light weight JavaScript code that does the following:
  - ✓ Parses the Query
  - ✓ Creates different Solr requests
  - ✓ Display the queried results
- The different types of request types are as follows:
  - ✓ Lookup per three-letter airport code, such as `RDU` or `SFO`.
  - ✓ Lookup per route, such as `SFO TO ATL` or `RDU TO ATL`. (Multiple hops are not supported.)
  - ✓ Clicking the search button when the search box is empty to show various statistics for all flights.
  - ✓ Finding nearby airports by using the `near` operator, as in `near:SFO` or `near:SFO TO ATL`
  - ✓ Finding likely delays at various distances of travel (less than 500 miles, 500 to 1000, 1000 to2000, 2000 and beyond), as in `likely:SFO`.
  - ✓ Any arbitrary Solr query to feed to Solr's `/travel` request handler, such as&`q=AirportCity:Francisco`
- The first three request types in the aforementioned list are all variations of the same type. These variants highlight Solr's pivot faceting capabilities to show, for instance, the most common arrival delay times per route (such as `SFO TO ATL`) per airline per flight number.

- The `near` option leverages the new Lucene and Solr spatial capabilities to perform significantly enhanced spatial calculations such as complex polygon intersections.

- The `likely` option showcases Solr's grouping capabilities to show airports at a range of distances from an originating airport that had arrival delays of more than 30 minutes.
- All of these request types augment the map with display information through a small amount of D3 JavaScript.
- For the last request type in the list, I simply return the associated JSON. This request type enables you to explore the data on your own. If you use this request type in your own applications, you naturally would want to use the response in an application-specific way.

Let us now see the following query in our example. Suppose if you search all queries from SFO to ATL, you get the results as shown below:



```
Route Stats:
Carrier: Delta Air Lines Inc. (257)
 Flight #: 156 (80)
   Arr. Delay (min):
   5 (5)
   7 (5)
   10 (5)
   -6 (4)
   -4 (4)
 Flight #: 98 (88)
   Arr. Delay (min):
   5 (8)
   2 (5)
   15 (5)
   1 (4)
   7 (4)
 Flight #: 116 (80)
   Arr. Delay (min):
   10 (7)
   -6 (5)
   3 (5)
   11 (5)
   -8 (3)
Carrier: Eastern Air Lines Inc. (256)
 Flight #: 70 (88)
   Arr. Delay (min):
   -4 (5)
   0 (5)
   -7 (5)
   -16 (4)
   -10 (4)
 Flight #: 74 (88)
   Arr. Delay (min):
   10 (8)
   6 (8)
   -4 (5)
   0 (4)
   2 (4)
```

Fig 32. Query Results 1

As you can see in the figure above, the 2 airports are highlighted: one in green and one in red. The list on the right is the Route Stats list and it shows the most common arrival delay times per flight per airline. This is specific to the number of years that you download. This information tells you that the Delta flight has been delayed five minutes before its arrival in Atlanta. This was for five times. And also, the flight had managed to arrive early by six minutes during 4 different occasions.

The Solr request is as follows:

Solr/collection1/travel?&wt=json&facet=true&facet.limit=5&fq=Origin:SFOAND Dest:ATL&q=*:*&facet.pivot=UniqueCarrier,FlightNum,ArrDelay&f.UniqueCarrier.facet.limit =10&f.FlightNum.facet.limit=10

The key functionality in this request is facet.pivot parameter. facet.pivot pivots from the airline (called UniqueCarrier) to FlightNum through to ArrDelay, thereby providing the nested structure that's displayed in Figure 3's Route Stats.

If you try a near query, as in near:JFK, your result should look similar to the following figure.

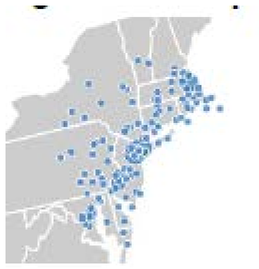


Fig 32. Query Results 2

The Solr request for the above query is:

&fq=source:Airports&q=AirportLocationJTS:"IsWithin(Circle(40.639751,-73.778925 d=3))"...

This request looks for all airports that fall within a circle whose center is at 40.639751 degrees latitude and -73.778925.

References

Chandrasekaran, P., & V, S. (n.d.). *Distributed Search API*. Retrieved from Edu website: http://www.cs.ucsb.edu/~prakash/projects/cs290c/Distributed_Search_API.pdf

Data Import Handler. (n.d.). Retrieved from Solr Wiki: http://wiki.apache.org/solr/DataImportHandler

Fox, E. (n.d.). *Indexing and Searching.* Retrieved from Solr website: http://curric.dlib.vt.edu/modDev/package_modules/MidtermModuleTeam1-Solr.pdf

Getting Started with Amazon EC2 Linux Instances . (n.d.). Retrieved from Amazon AWS website: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

Grainger, T., & Potter, T. (n.d.). *Solar in Action.* Manning Publications.

Kuc, R. (n.d.). *Scaling Solr with Solr Cloud*. Retrieved from Slideshare website: http://www.slideshare.net/sematext/scaling-solr-with-solr-cloud?qid=e759cd50-8ca6-4ff6-9c7f-2e9f257faf84&v=qf1&b=&from_search=2

Nridge, A. (n.d.). *Open Source Search Platform.* Retrieved from Apache Solr website: http://www.nridge.com/presentations/ApacheSolrPresentation.pdf

Rafalovitch, A. (n.d.). *Instant Apache Solr for Indexing Data How-to.*

Smiley, D., & E, P. (n.d.). *Apache Solr 3 Enterprise Search Server.* Packt.

Solr Quick Start. (n.d.). Retrieved from Solr web site: http://lucene.apache.org/solr/quickstart.html

Solr Tutorial. (n.d.). Retrieved from Solr Guide: http://lucene.apache.org/solr/4_2_1/tutorial.html

Tran, M. (n.d.). *Enterprise search with Solr*. Retrieved from http://www.slideshare.net/minhkiller/apache-Solr-8590156