


Build Solr Apps Faster With Fusion

Comparing Apache Solr to Lucidworks Fusion



*There's a lot of data in your organization.
And it's all over the place.
And it's constantly growing.
And **everybody** wants access to it.*

The promise of search was that we could break down the silos between teams and business units. Employees would go to one place to find the information and insights crucial to doing their jobs. Customers would easily find the products they are searching for and quickly buy them.

Instead the silos got bigger and more unwieldy. Each business unit built their own search solution. The systems aren't talking to each other. Securing the data is a nightmare. And if we're talking ecommerce, your traffic reports show how customers get lost in your online store and simply just give up and go somewhere else.

Along with the deluge of data is the tidal shift in user expectations. Your users expect Google. Your customers expect Amazon. Users want to type just a few words and get back relevant, personalized search results that give them exactly what they wanted—and only showing them documents or data they have permission to access. They expect it to work on every device, every screen, every time.

Then there's IT. Overworked, overwhelmed, and understaffed. IT is trying to serve various internal customers who have numerous initiatives. For most IT organizations, search isn't their strong suit. They don't have internal expertise in building search applications. It just simply isn't a core competency.

And there's you stuck in the middle—trying to build a search app.

You feel you've got a pretty good handle on the above challenges—exponentially increasing amounts of data, ever-increasing user expectations, and limited IT resources—along with your technical requirements. That's why you chose to build your search app with Apache Solr.

Solr's Great, But Solr's Not Enough

Solr's a solid choice. It is used by some of the world's biggest companies for applications ranging from enterprise search and log analytics to ecommerce search and site search. It is open source, extensible, scalable, and high performance. Solr's open source pedigree has helped drive technical innovation faster than legacy search vendors over the past couple decades.

As you start scoping out your search app with Solr, you find that your project has a lot of expectations to meet that go far and beyond the capabilities of vanilla Solr. Some of these requirements are related to what we mentioned above—the consumerization of technology. Some are features of how an enterprise-grade search app should perform or function.

Meeting these expectations means writing a lot of custom code. You'll need to build several more components to get a Solr app up to enterprise quality:

1. Write a polished, easy-to-use UI that works across all browsers and all devices.
2. Connect to security infrastructure such as Active Directory via LDAP or Kerberos.
3. Secure the data so users only see the documents, collections, fields and basically just the data they should have permission to access.
4. Implement ETL scripts and data processing to index and enrich data in a timely and efficient manner.
5. Connect to RDBMS (Oracle, SQL Server, DB2), Big Data (Hadoop, HDFS) and other less traditional infrastructure.
6. Setup a web crawler to crawl any documents and data from the public web or your company's intranet.
7. Deploy cluster management and other administrative maintenance including analytics, reporting, and alerts.

So you start calculating the time and staff needed to build all of those components. You haven't even started to scope out the effort to build the parts of the app unique to your business requirements. We're talking about just the effort to get Solr up to par with enterprise expectations.

Not to mention relevancy. When users can't find what they need, you need to figure out how to adjust relevance. Aside from fine-tuning Solr for scalability, security, and performance, you also need to implement methods to increase relevancy. As you work on these challenges, you'll need more processing power to index and search in a timely manner. This might call for connecting to a distributed processing framework like Apache Spark. With improved relevance in mind, you hand-code scripts and send all the processing through Apache Spark.



Solr is used by 90% of
the Fortune 500

And so you build all that stuff.

It's many months later. Older and wiser, you have a homegrown search app built with Apache Solr and several custom coded components or open source packages and modules stitched together into a working search application. You still have to maintain all these bits and pieces (and hope the expertise that helped you assemble them doesn't leave the organization).

What if all that madness was already done? You could leapfrog months of development time and focus your resources and time on the part of your app that are unique to your business requirements.

That's why we built Fusion.

Fusion Begins Where Solr Leaves Off

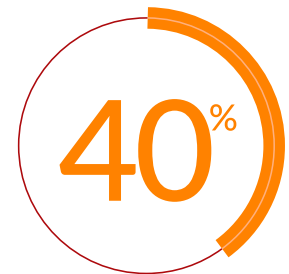
Lucidworks Fusion is built on top of Solr and even runs on top of your existing Solr deployment. You have full access to all of the power, speed, and scalability that Apache Solr brings to an app. But Fusion augments it with tools for improving relevance as well as out of the box solutions for data acquisition, security and administration—all organized with a friendly admin interface. Additionally, we fuse the processing power of Apache Spark into our platform to perform operations at scale.

Lucidworks Fusion is built on top of Solr and even runs on top of your existing Solr deployment. You have full access to all of the power, speed, and scalability that Apache Solr brings to an app.

Lucidworks is the longtime principal commercial supporter of Solr, employing one-third of Solr committers and contributing over half the code with each new release. We have years of experience building and deploying enterprise search apps for some of the world's biggest brands.

We built all this expertise right into Lucidworks Fusion.

Think of Fusion as Solr++.



Lucidworks employs over 40% of the active committers on the Solr project



Lucidworks contributes over 70% of Solr's open source codebase

Fusion Features

Amplifying the power of Apache Solr, Fusion provides a number of additional capabilities:



Over 60 connectors to connect to your data wherever it lives



Simplified ETL with flexible, scalable indexing pipelines for easy data acquisition



Better, more relevant results through extensive support for capturing signals like clicks, views, purchases, and shares



Recommendation engine for using signals straight out of the box



Alerts with integration for email, Slack, PagerDuty, and more



Graphical admin UI to simplify all parts of development and deployment



Analytics with fast and flexible charting and reporting dashboards for in-depth insights into users, results, and data



Advanced natural language processing (NLP) services for parsing for key concepts and named entities and lower order functions like part-of-speech tagging



Blob storage service for storing large objects like machine learning models or arbitrary customer content



Security with fine-grained authorization controls, authentication options, and encrypted communications that integrates with common systems like LDAP, Active Directory, and Kerberos and more



Native support for big data frameworks like HDFS, Apache Spark, and HBase



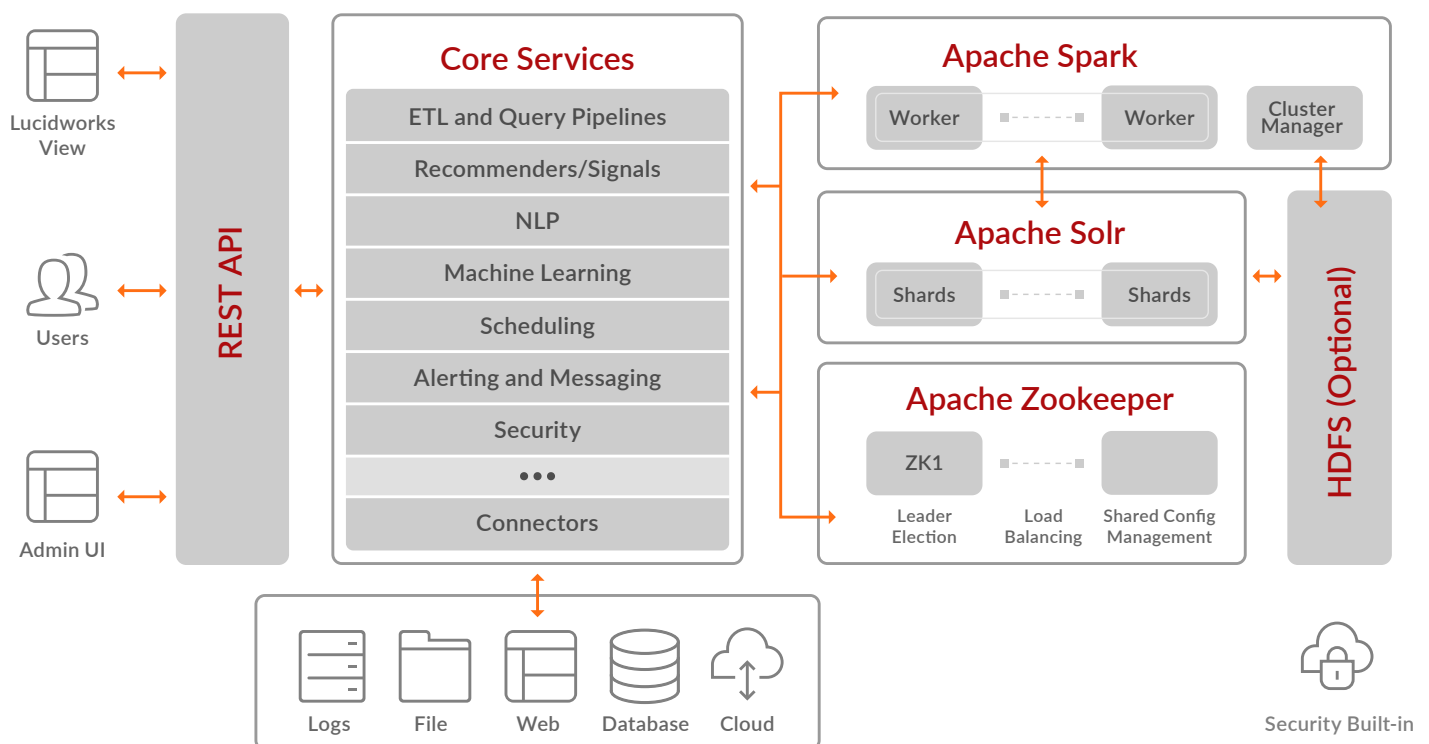
A multitude of extensions to Solr's request and query capabilities via our query pipelines



A UI framework for rapid prototyping of your search app

Fusion Architecture

The diagram below shows the architecture for a typical Fusion deployment



6 Weeks or 6 Months?

Here's a look at some of the key features that most search apps have to support and compare how long they would take to do-it-yourself with Solr versus using the capabilities of Lucidworks Fusion:

Feature	Time to Build and Deploy	<div> <div></div> DIY Solr <div></div> With Fusion </div>
Multi-browser, multi-device end user interface	<div> <div></div> 4 weeks minimum, to several months <div></div> 1 day to 2 weeks </div>	
Security/authentication	<div> <div></div> 3 months <div></div> 1 day </div>	
ETL and index time enrichment	<div> <div></div> 2 weeks, to several months <div></div> 1 day to 2 weeks depending on complexity </div>	
Creating new connectors for data sources	<div> <div></div> 1 to 3 months per connector <div></div> Out of the box </div>	
Web crawler	<div> <div></div> 4 months or more <div></div> Out of the box </div>	
Cluster management/ops	<div> <div></div> 6 weeks <div></div> 3-6 days </div>	
UI manage rules, boosts, and blocks	<div> <div></div> 4 months <div></div> Out of the box with 1 week to customize </div>	
Notifications and alerts	<div> <div></div> 2 weeks or more <div></div> Out of the box and a few hours of customization </div>	
Analytics and reporting	<div> <div></div> 1 week or more <div></div> Out of the box with pre-configured reports and a few hours of customization </div>	
Admin UI	<div> <div>X</div> You'd probably just rely on engineers to implement changes to rules and relevancy directly in the code <div></div> Out of the box and ready for use by non-technical business users </div>	
NLP parsing, ML integration, entity extraction, taxonomies	<div> <div></div> 2 months to over a year <div></div> Out of the box with pre-configured reports and a few hours of customization </div>	
Big data framework integration (Hadoop, etc.)	<div> <div></div> 4-6 months <div></div> Out of the box </div>	

And that just gets you to deployment; the app shipped and out the door. After that, your development team has to stay current with new releases of Solr, any changes to data sources, packages, and connectors. But with Fusion, you have less code for your team to maintain, and enjoy regular releases that come with new capabilities and improve existing features.

And we haven't even talked about staffing.

Fusion Keeps Teams Small and Agile

There's also a big difference between the number of engineers needed for building a Solr search app versus building one in Fusion.

For a moderately sized organization with a typical amount of data and complexity, a typical Apache Solr app will need 4-5 full-time Java developers. At least two of those devs will need to be familiar with Apache Solr. This is not always easy expertise to find. You'll also need at least 1 full-time front-end developer to build your end user or admin interface. Post-deployment, you'll need 1-2 Java developers on the operations side to deploy changes to relevancy scripts and updates to connectors. One of those developers will need to be Solr-savvy.

With Lucidworks Fusion, you don't need to find Java developers with specific Apache Solr knowledge. You'll need 1 or 2 full-time Java engineers for development. You'll still need 1 full-time front-end developer to build your end user interface, but they'll work faster since Fusion includes a search-UI framework for quick prototyping and iteration. Post-deployment, you'll still need one general purpose admin but the business users can execute updates to rules and relevancy. Fusion has much more modest staffing requirements than going with just Solr.

Fusion is Cheaper, Faster, Better

You could still go it alone and roll-your-own. Spend months recreating the wheel, building the standard components every search app needs. Cobble together all the bits and pieces and packages that you need to make the app work and hope it provides responsive, relevant results.

Or you could just use Lucidworks Fusion.

Fusion gives you the power and reliability of Apache Solr with faster time to value, less code to maintain, a better operational experience, and fewer resources to hire. Cheaper, faster, better.

Time to give Fusion a test drive

To get started with Fusion, go to lucidworks.com/download and download the latest version. Fusion is free for development purposes and is easy to setup and run locally. All you need is a recent version of Java.

You can be up and searching data in less than 5 minutes.