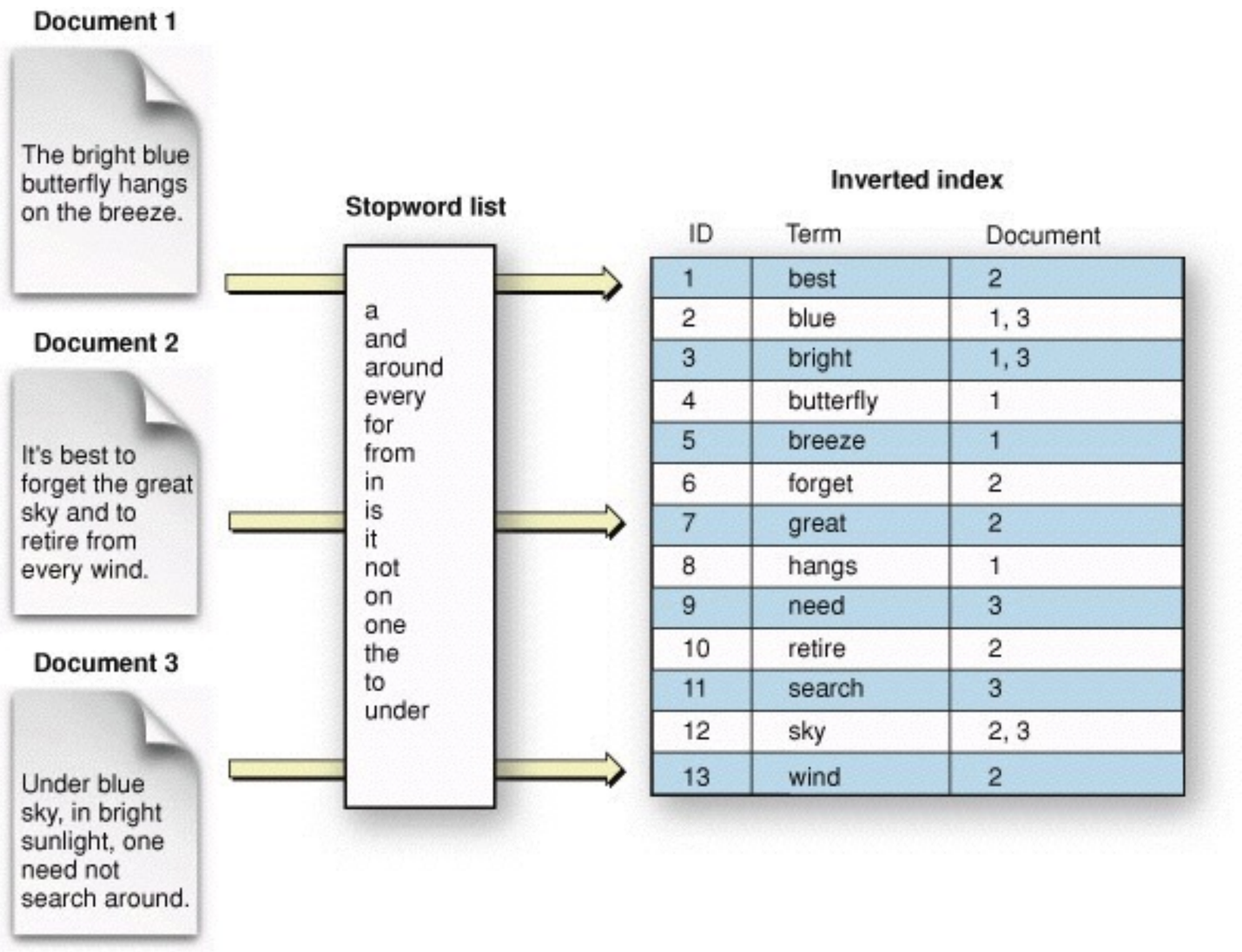
Lucidworks

Fundamentals of Search

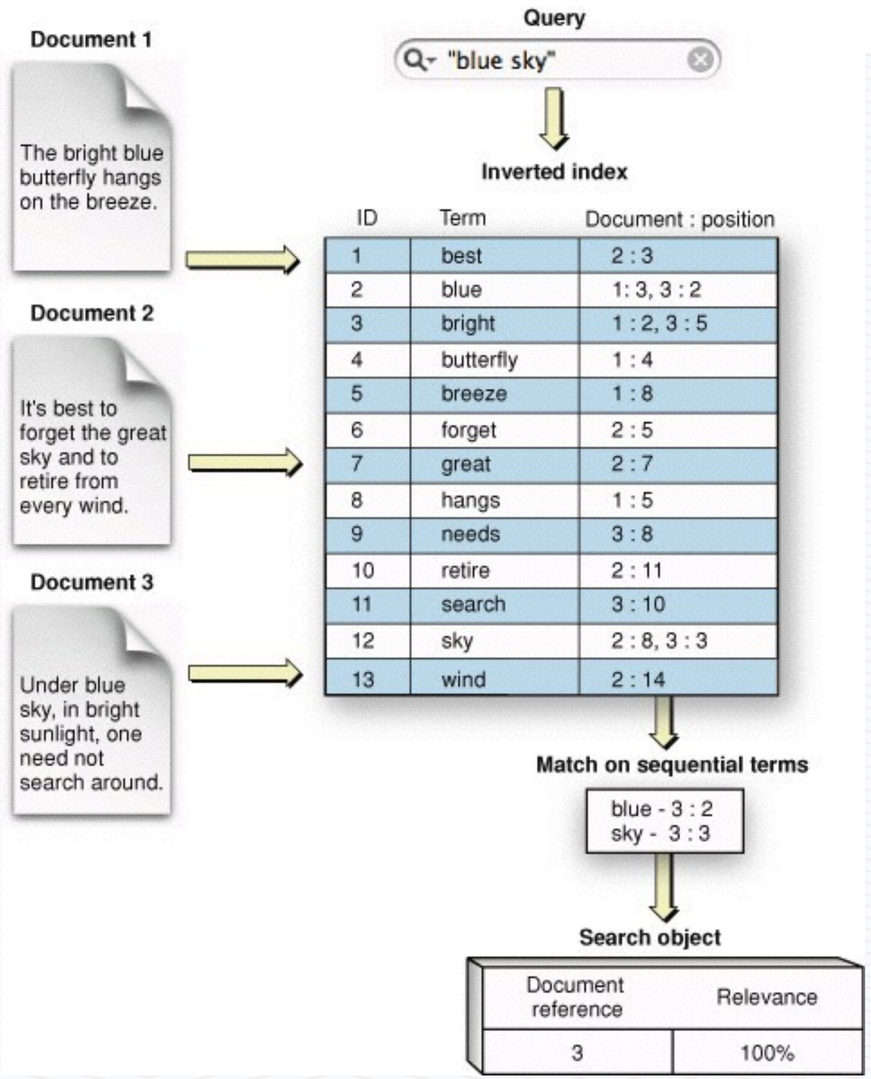
The Inverted Index

Inverted Index with Stop Words



ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

Inverted Index with Term Positions



ID	Term	Document : position
1	best	2 : 3
2	blue	1: 3, 3 : 2
3	bright	1 : 2, 3 : 5
4	butterfly	1 : 4
5	breeze	1 : 8
6	forget	2 : 5
7	great	2 : 7
8	hangs	1 : 5
9	needs	3 : 8
10	retire	2 : 11
11	search	3 : 10
12	sky	2 : 8, 3 : 3
13	wind	2 : 14

Search Engines vs. Databases

Search Requires a Different Data Modeling and Access Paradigm

Traditional Databases	Modern Search Engines	Comments
Store Normalized Data in Tables	Store Denormalized Documents	Need to think differently about the data model
Vertical Scaling	Horizontal Scaling	Solr is built for Hadoop-scale
Searches require Table Scan (slows down dramatically as data and access grows)	Get extremely fast, interactive speeds on “big” data	Optimized for information Retrieval
Does not analyze unstructured text; slow at querying	Optimized for unstructured and semistructured data	Search-first NoSQL store
Results may be sorted by some column	Results ranked by relevance	Many ways to tune relevance in order to provide powerful user experiences

Different data model and horizontal scaling are characteristics of other modern NoSQL databases (Cassandra, HBASE, Couchbase, etc.) but the other three elements are unique to search engines

Solr Documents Do Not Follow the Traditional Normalized Model

User:

Id	UserName	About	Location	Company	LastModified
456	Coco	I'm a real monkey	1	1	2013-06-01 T15:26:37Z
123	John Doe	Senior Software Engineer with 10 years of experience with java, ruby, and .net	2	1	2013-06-05 T12:25:12Z

Location:

Id	City	State
1	Norcross	GA
2	Atlanta	GA
3	Decatur	GA

Company:

Id	CompanyName	CompanyDescription	Location
1	Code Monkeys R Us, LLC	we write lots of code	2

Solr Documents

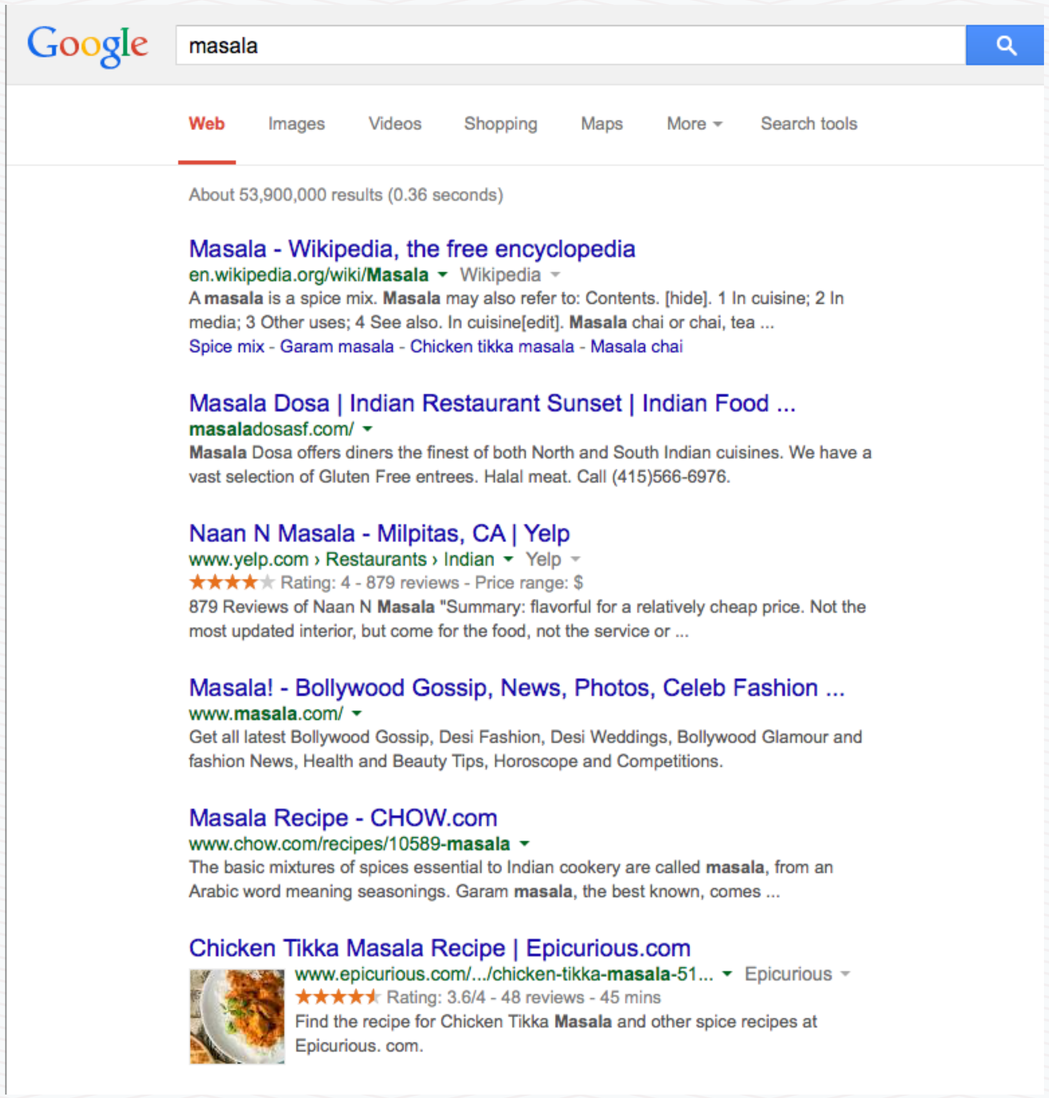
```
<doc>
  <field name="id">123</field>
  <field name="username">John Doe</field>
  <field name="about">Senior Software Engineer with 10 years of
                      experience with java, ruby, and .net
  </field>
  <field name="usercity">Atlanta</field>
  <field name="userstate">Georgia</field>
  <field name="companyname">Code Monkeys R Us, LLC</field>
  <field name="companydescription">we write lots of code</field>
  <field name="companycity">Decatur</field>
  <field name="companystate">Georgia</field>
  <field name="lastmodified">2013-06-05T12:25:12Z</field>
</doc>

<doc>
  <field name="id">456</field>
  <field name="username">Coco</field>
  <field name="about">I'm a real monkey</field>
  <field name="usercity">Norcross</field>
  <field name="userstate">Georgia</field>
  <field name="companyname">Code Monkeys R Us, LLC</field>
  <field name="companydescription">we write lots of code</field>
  <field name="companycity">Decatur</field>
  <field name="companystate">Georgia</field>
  <field name="lastmodified">2013-06-01T15:26:37Z</field>
</doc>
```

1 **Company information for first user.**

2 **The same company information repeated for the second user.**

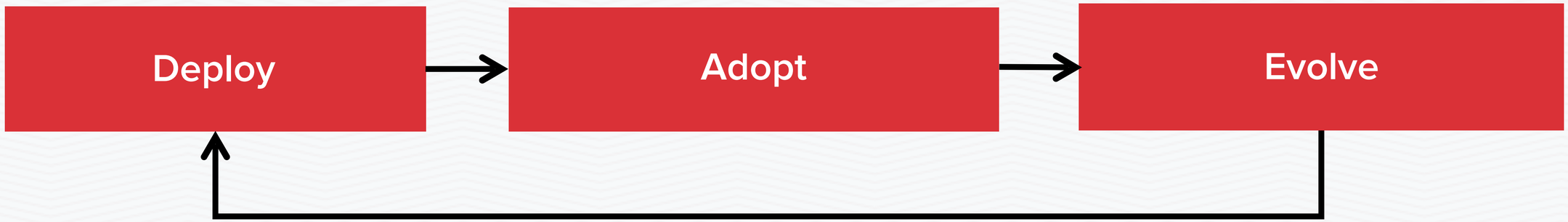
Results Ranked by Relevance

- Does not give you a randomly ordered set of results that matched your query; scores results and attempts to first return items that are more likely to be relevant/useful
- Not just “what matches user query,” but “what is most likely the thing the user wanted”
- Search is Recommendation



Iterative Search App Development Model

Maintaining a Virtuous Cycle....



…that helps customers grow along the Search Deployment Maturity Model

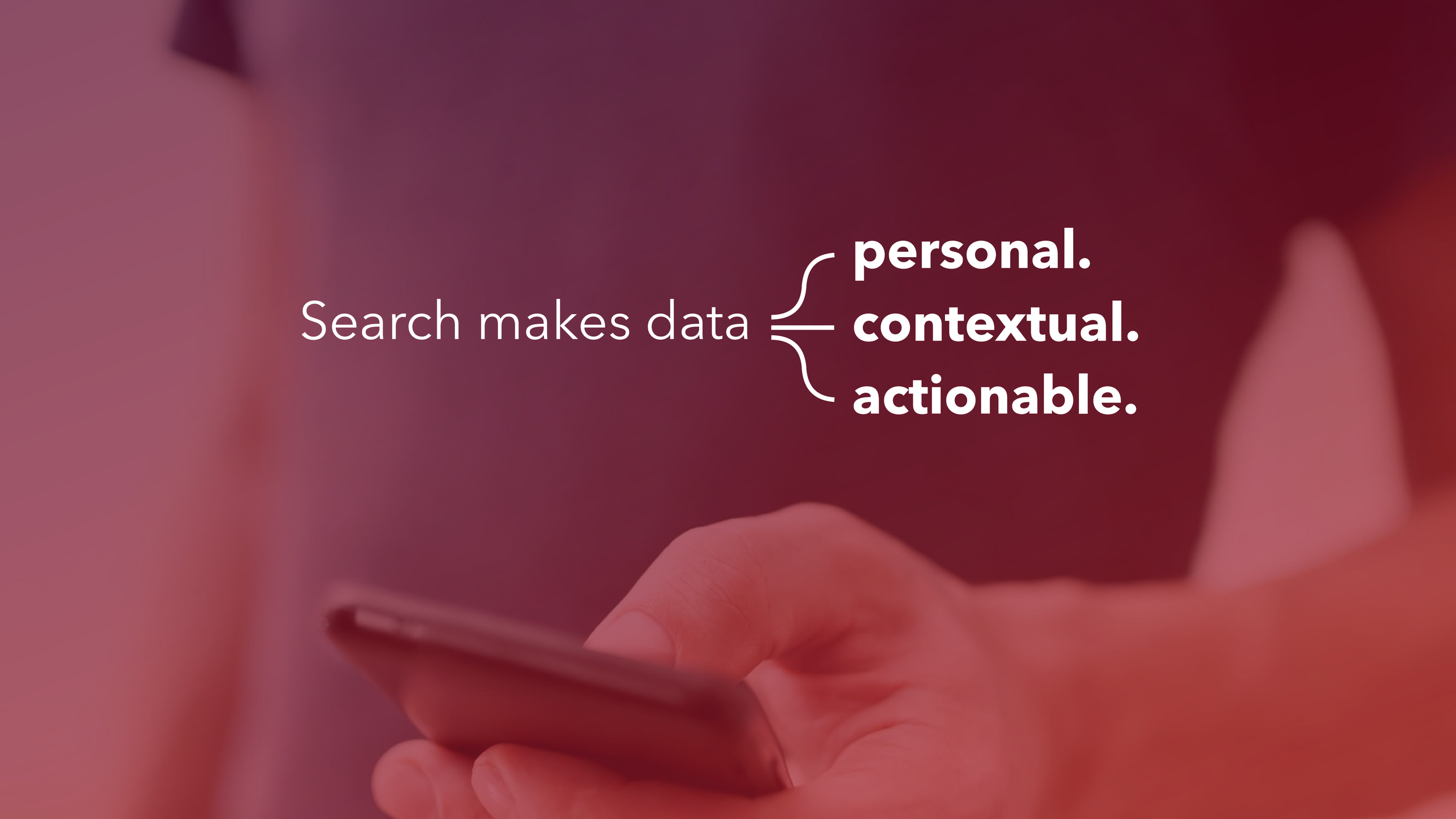
	Early-Stage	Knowledgeable	Experienced	Optimized
Business Driver	Add Search to Application	Improve KPI's (conversion, mean time to resolution, etc.)	Underpin Core Corporate Initiatives	Competitive Advantage
Ownership	Team	Department	Business Unit	Multiple BU's Corporate-Wide
Search Organization	Individual(s)	Team	Competency	Practice and Culture
Applications	Keyword Search	Data Enrichment Complex Queries	Multiple Data Sources/ Federated Search	Search as Experience: Virtuous Cycle between Users and Data
Scale	Low	Medium	High	Massive
Technology Adoption	Add-on tool	Key Part of Solution	Pervasive in IT Stack	Platform as a Service

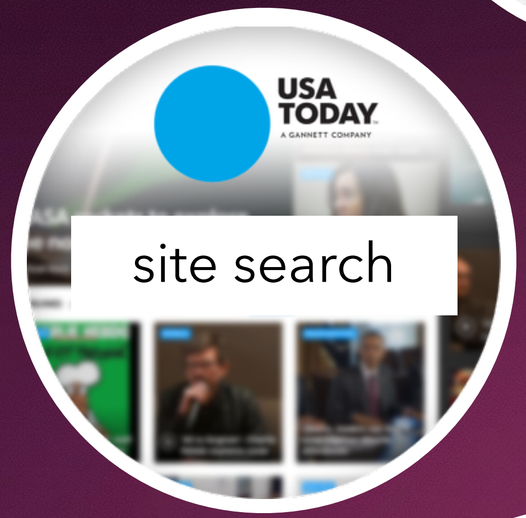
Solr Accelerated

Rapid Search Application Deployment with Lucidworks Fusion

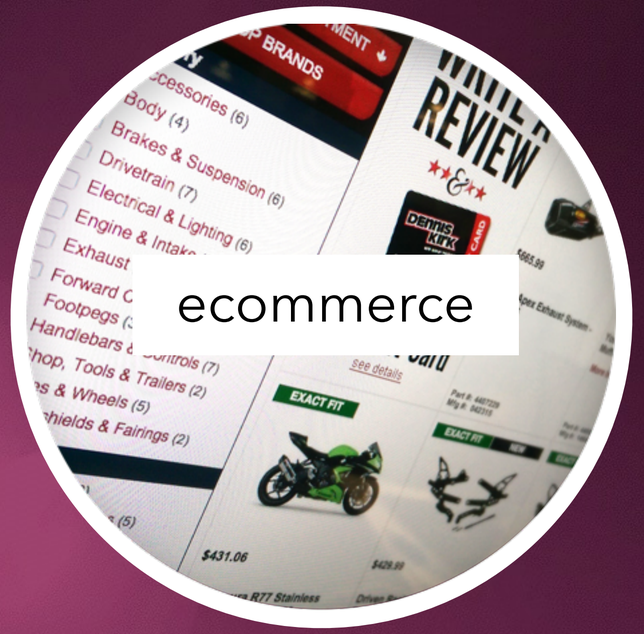
Why Fusion?

Search is **more than just a box.**


Search makes data
personal.
contextual.
actionable.



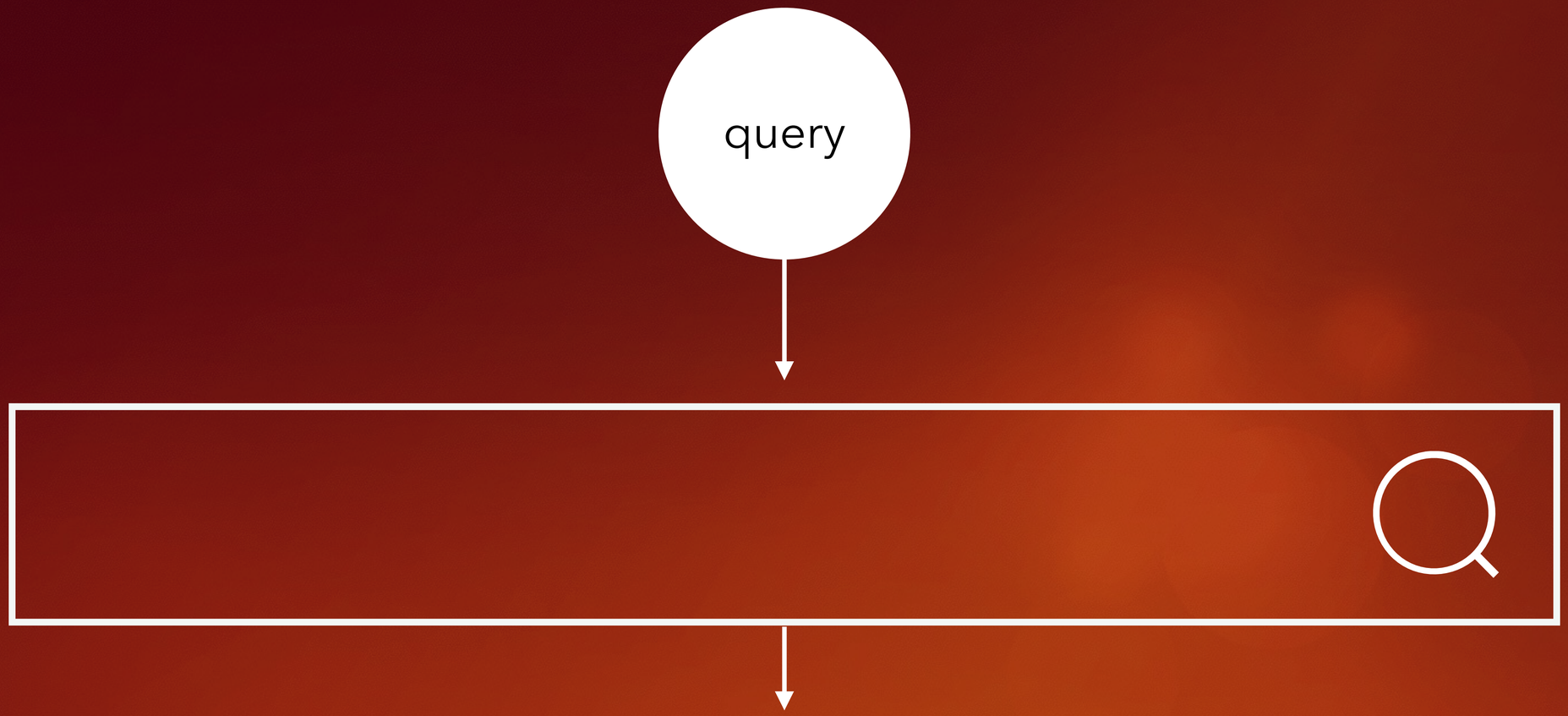
Search is **everywhere.**



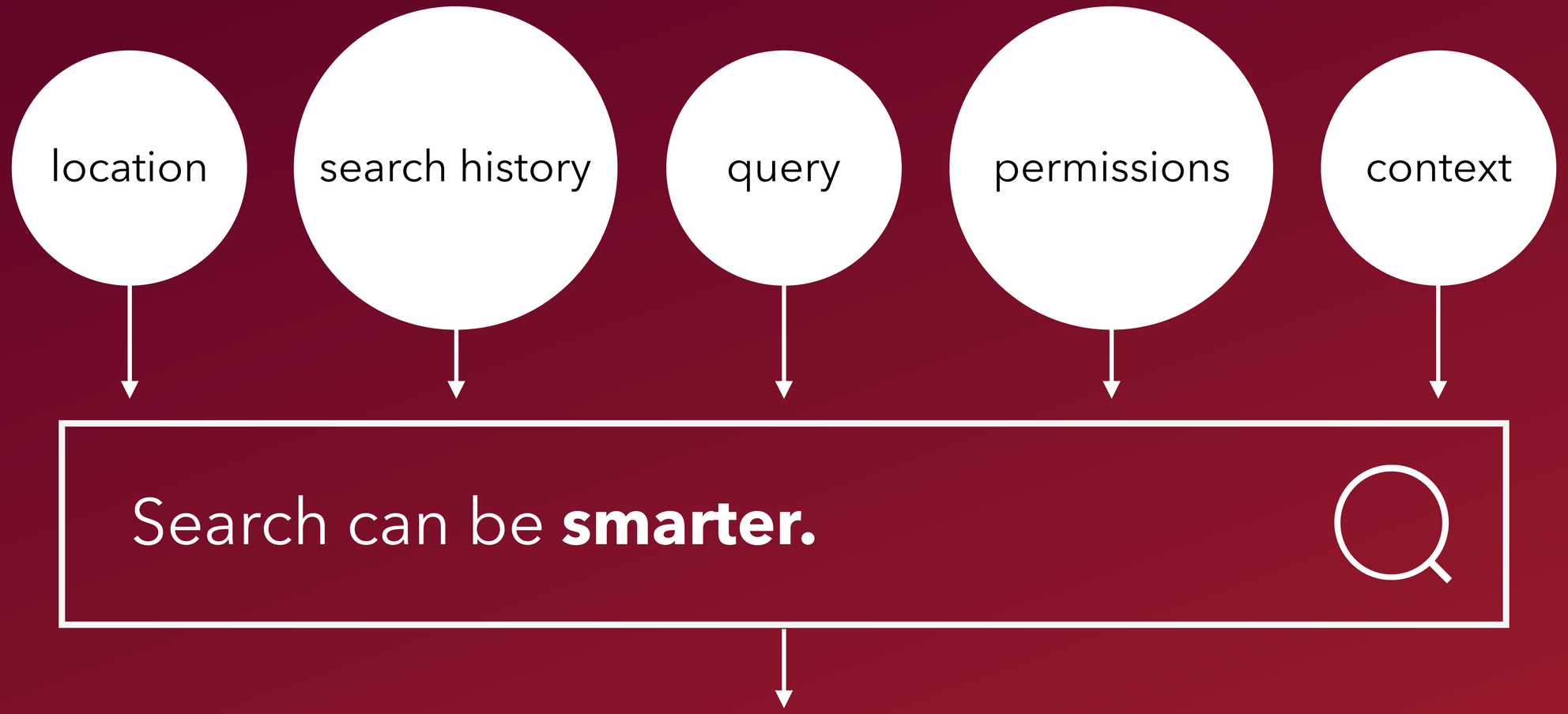
Search is the key to unlocking big data.

Secure access to all your data through one interface, empowering everyone in your organization to access the data they need.

Search anything.



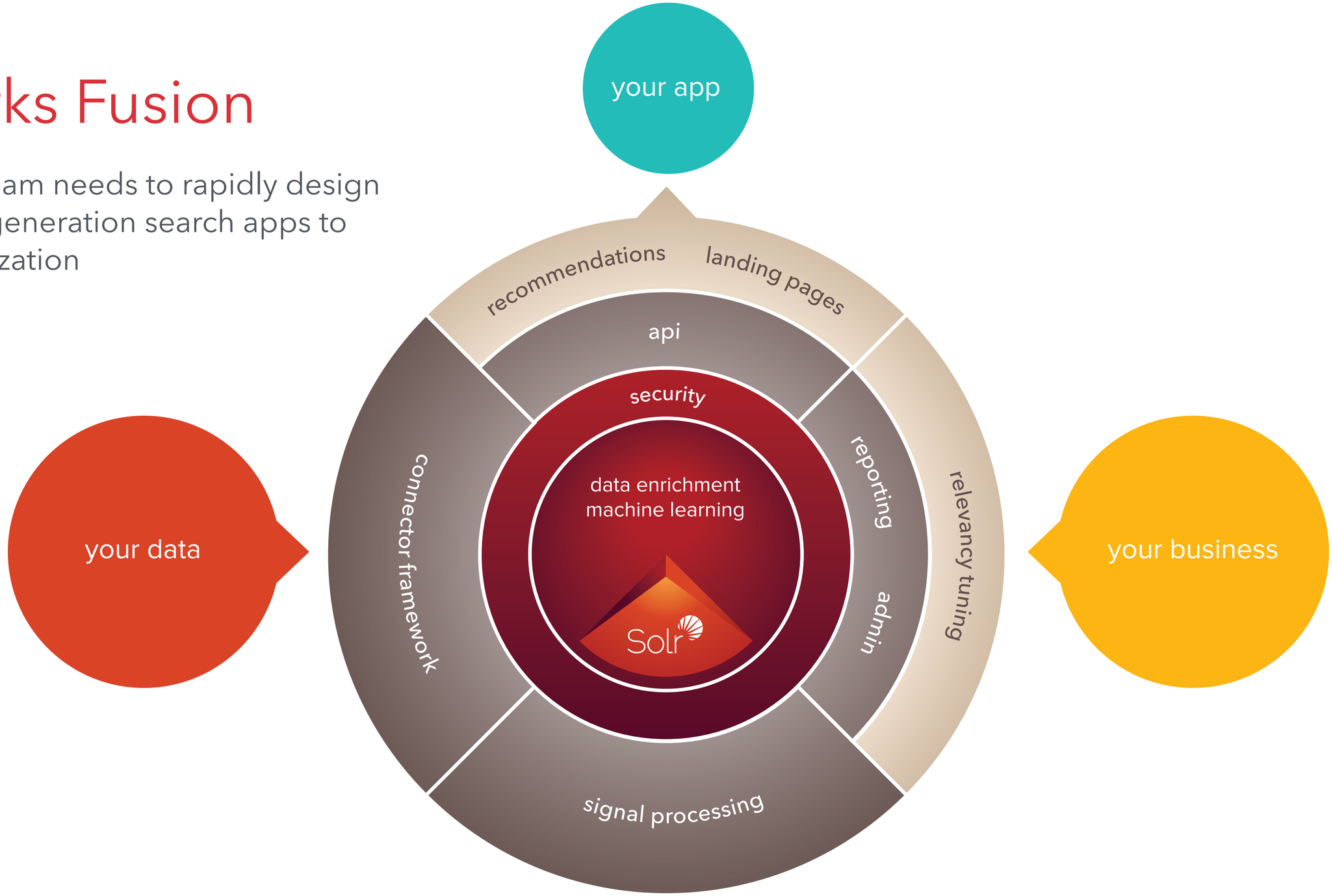
Traditional enterprise search
was all about the query.



Personal, contextual, relevant results: consumer-like simplicity and power **in the enterprise.**

Lucidworks Fusion

Everything your team needs to rapidly design and deploy next-generation search apps to your entire organization



Training Agenda and Learning Goals

Lucidworks Fusion Overview

- **Solr based** - supports multiple versions (4.4 and up, with a few exceptions)
- **Security** - content and connectivity control, integration with LDAP
- **Signals** - Search engine know thyself, and thy users
- **Modification Pipelines** - Input time and query time
- **Connectors and crawlers** - not as creepy as they sound
- **Scales with SolrCloud and Zookeeper** - keep your clouds in the sky and your monkeys well-fed
- **Built-in Log analysis** - Index and report on your Fusion logs, server logs, and other time series data
- **Friendly Admin Interface** - Makes everyone's life easier

Training Agenda

- Introductions
- Why Fusion; Training Goals
- Not your Father's Solr
- Fusion and Solr Deployment
- Getting Started; Navigation Basics
- Fusion and Solr APIs
- How do I get data into Solr?
- Monitoring, Log Analytics and Dashboards
- How do I tailor my Search Results?
- How do I drive more powerful User Experiences?
- Summary, Resources, Feedback

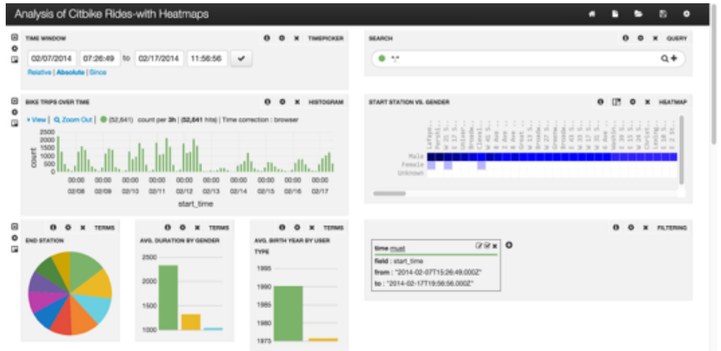
Building Powerful and “Antifragile” Search Applications—Easily

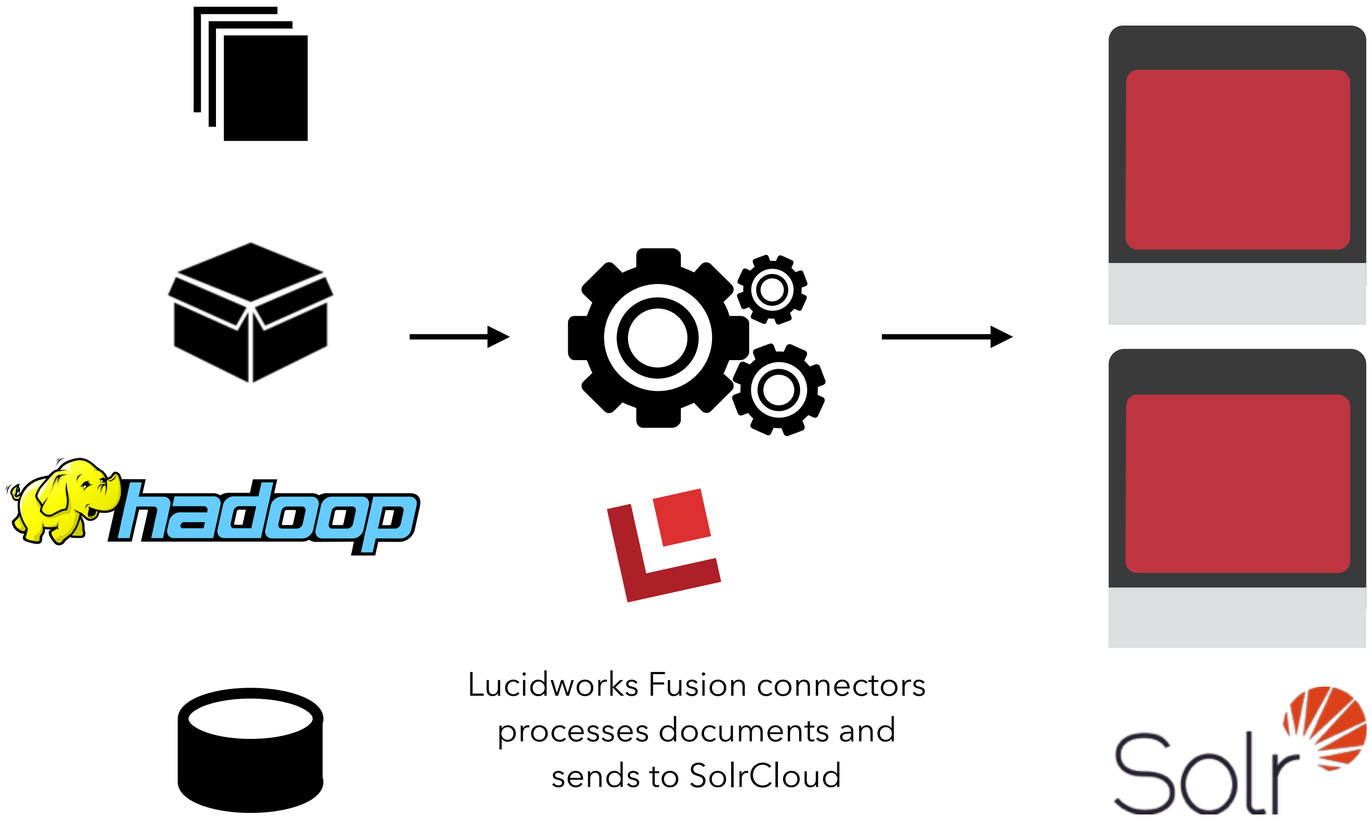
- This course intends to provide a strong foundation in Fusion. Students can use this base to learn advanced concepts from Lucidworks blogs, documentation and webinars
- At the end of this course you can use Fusion to:
  - Create collections and modify their schemas
  - Connect to multiple data sources and ingest content into Solr, modifying and transforming data along the way
  - Administer and monitor a Solr cluster; visualize time series data and build log analytics applications
  - Modify user experiences by tuning relevancy, modifying facets, boosting and blocking documents, etc.
  - Leverage signals to create contextual and personalized recommendations
  - Easily build next generation search apps such as….

Example Search Applications

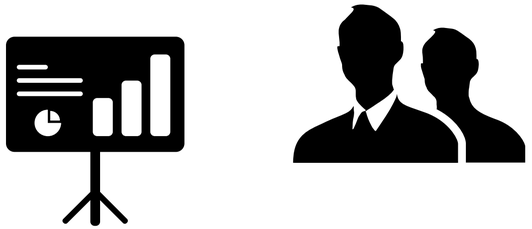
Enterprise Search

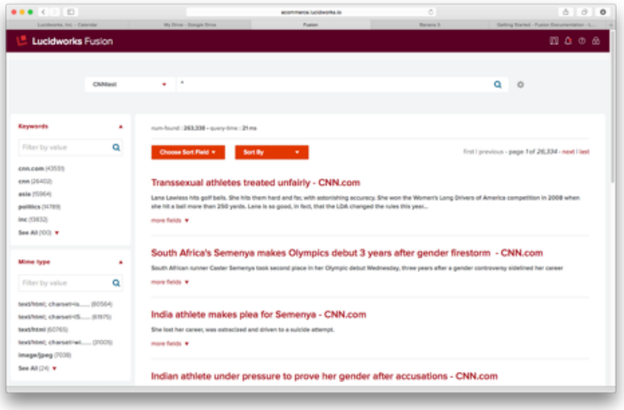
Document storage and federated search



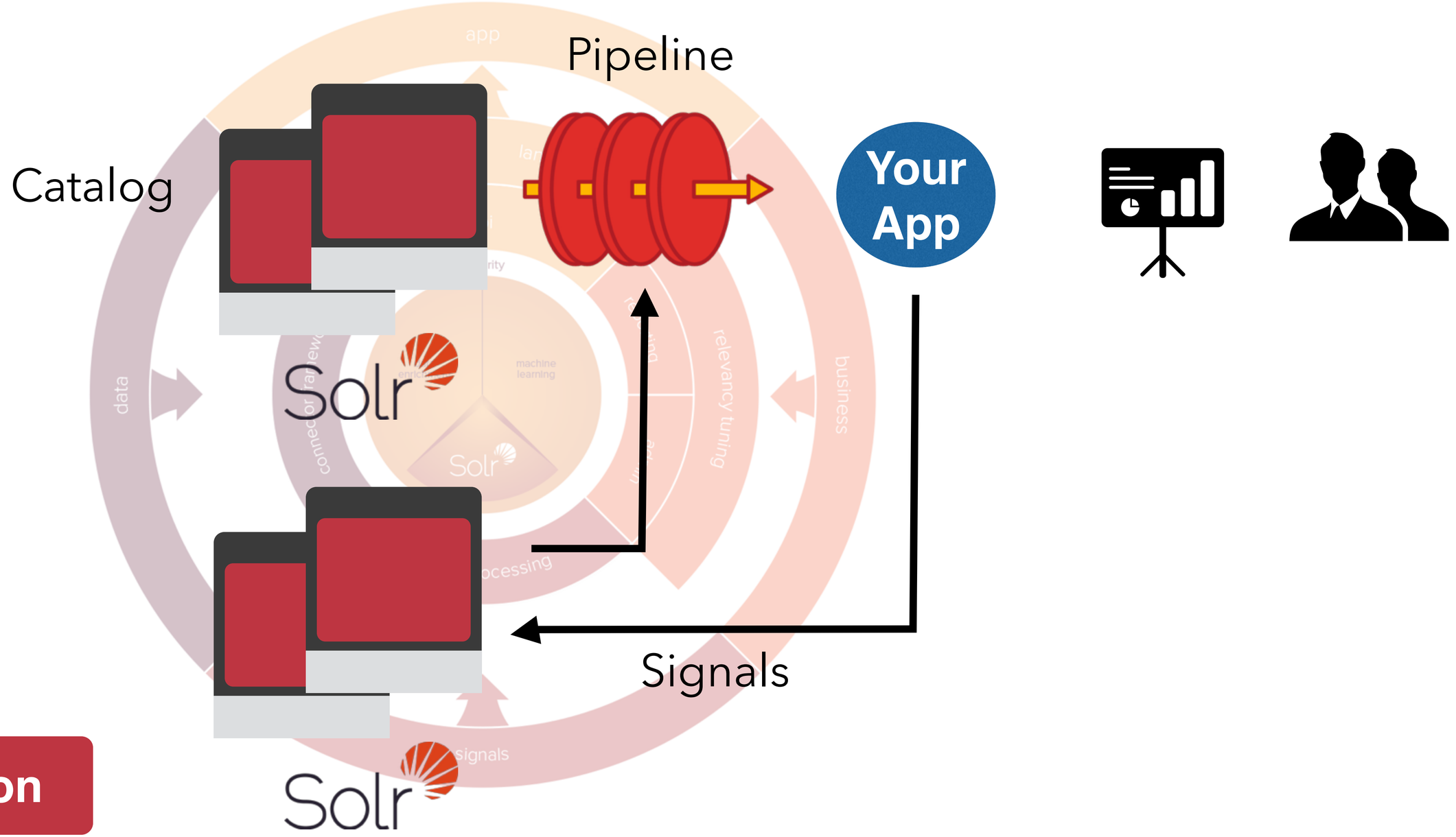


Lucidworks Fusion connectors processes documents and sends to SolrCloud





eCommerce: Search is Recommendation

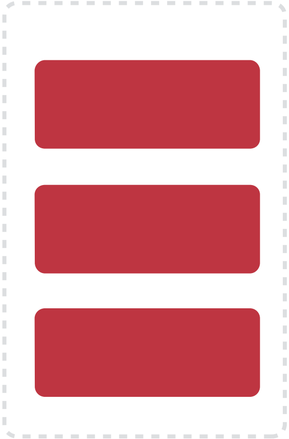


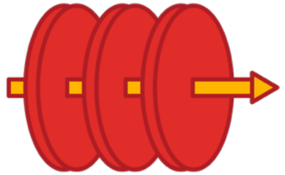
Log record search

High volume indexing of many small records

Machine generated log records
are sent to Flume.

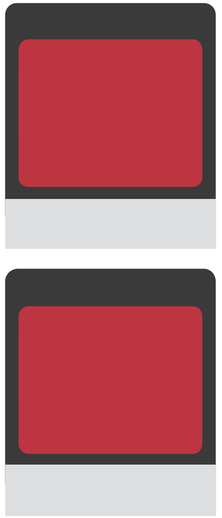
Flume forwards raw log record
to Hadoop for archiving.



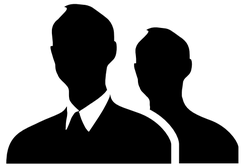


Pipeline

Flume simultaneously parses out
data in record into a Solr document,
forwarding resulting document to Solr



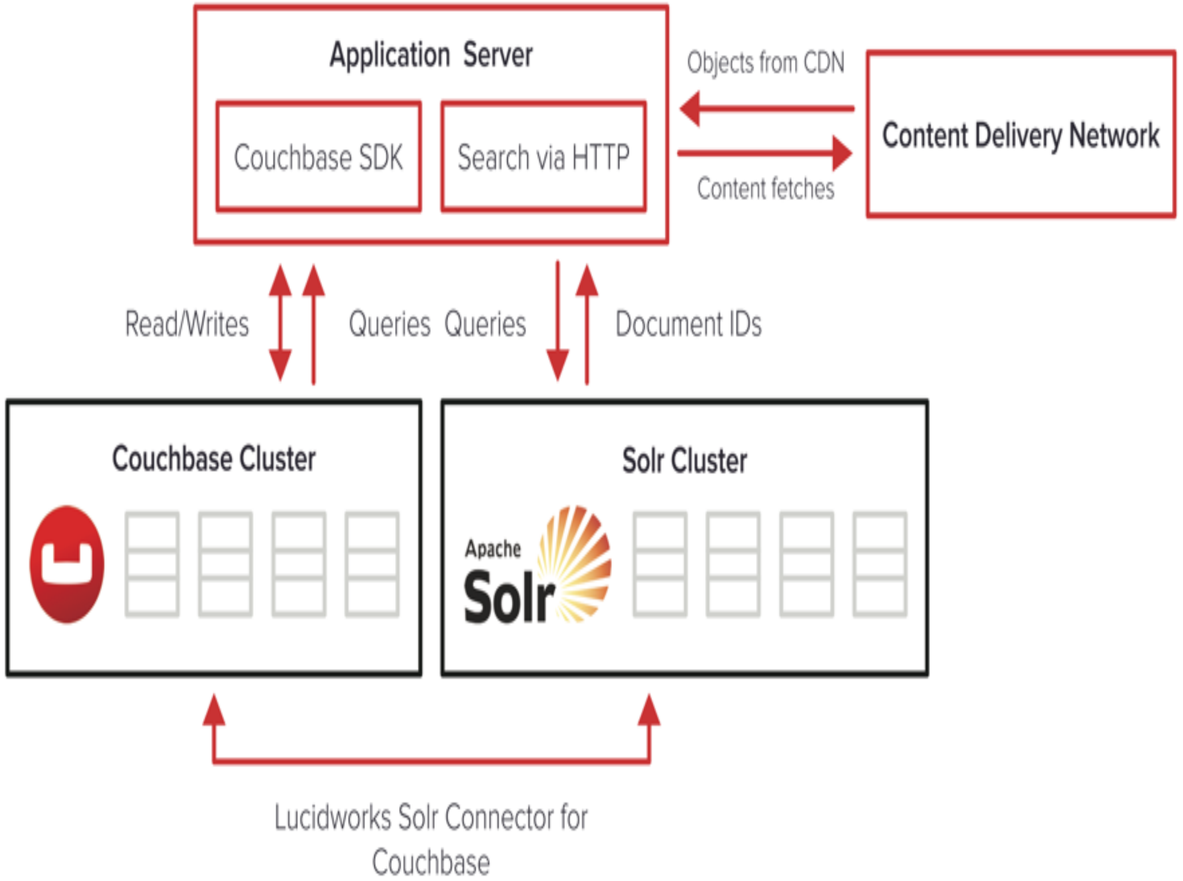
Solr

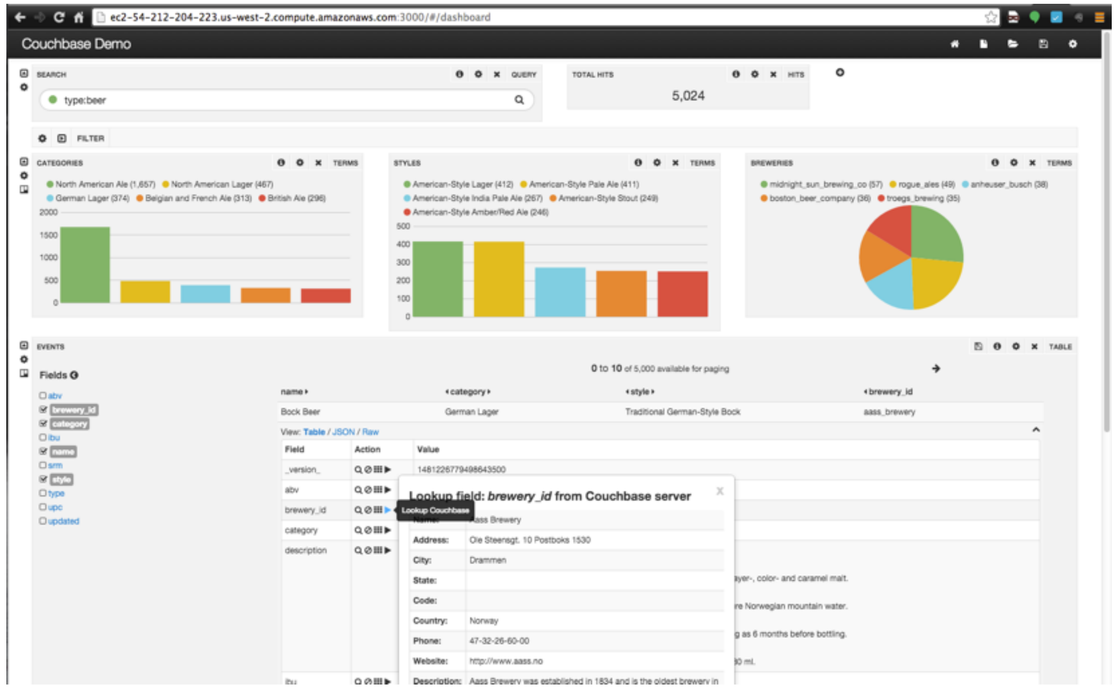


Lucidworks Fusion Dashboards exposes real-time
statistics and analytics to end-users,
as well as full-text search

Platform for Data-driven Applications

Data Access Layer for HDFS and NoSQL

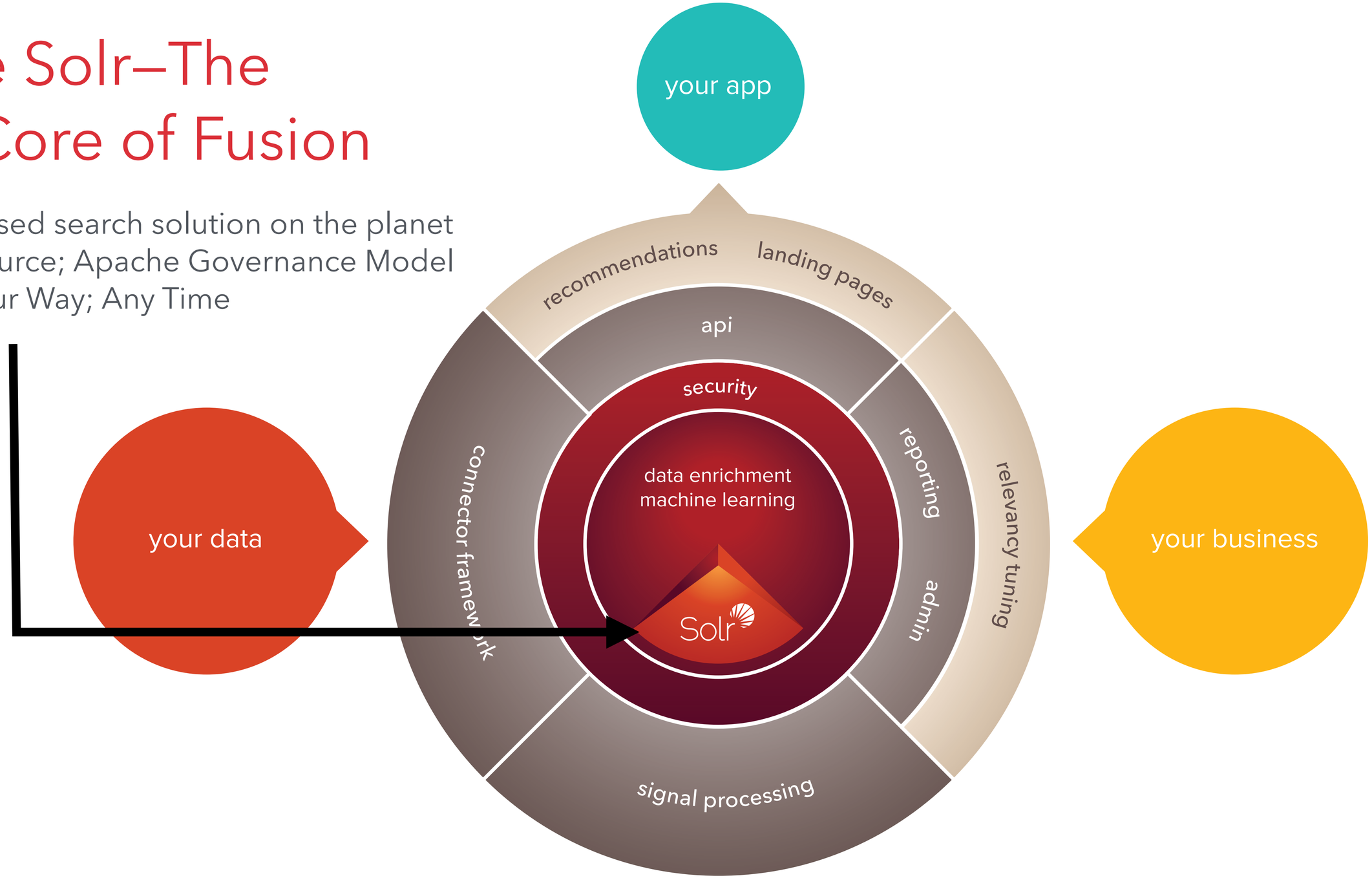




Not your Father’s Solr

Apache Solr–The Open Core of Fusion

Most widely used search solution on the planet
True Open Source; Apache Governance Model
Your Data; Your Way; Any Time



Solr

NEWS FEATURES RESOURCES SEARCH DOWNLOAD LUCENE ⇧ TLP

APACHE SOLR™ 5.0

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

TESTED AND PROVEN

Solr is trusted.

Solr powers some of the most heavily-trafficked websites and applications in the world.

DuckDuckGo Sears StubHub! Zappos.com POWERED by SERVICE® BEST BUY Aol.

Other Notable Users

AT&T	Instagram	Buy.com
Ticketmaster	Netflix	The Echo Nest
Chegg	Disney	Adobe
eBay	Internet Archive	SAP Hybris
Magento	IBM Websphere Commerce	Bloomberg
Comcast	MTV Networks	Travelocity

Why Solr?

- Full-text search with faceting; Near real-time indexing; Dynamic clustering; Rich document (e.g., Word, PDF) handling; Database integration; Hit highlighting; Geospatial search; Multiple language support; ....
- Distributed, Horizontally Scalable, Stable and Robust
- Search-first NoSQL store with Strong Analytics Capabilities
  - Deep Paging
  - Accurate Facets and Stats; Stats on Pivots (5.0)
- Easier to start-up; run as a service on Linux (5.0)

Fusion and Solr Deployment

Fusion Loves Solr

- Fusion makes Solr better!!
- Fusion works with your existing Solr infrastructure - not tied to a single Solr version
- Fusion can work with multiple Solr instances/installs - supports Solr 4.4 and up
- Don’t have Solr yet? Fusion ships with Solr

Fusion Clusters and Scales

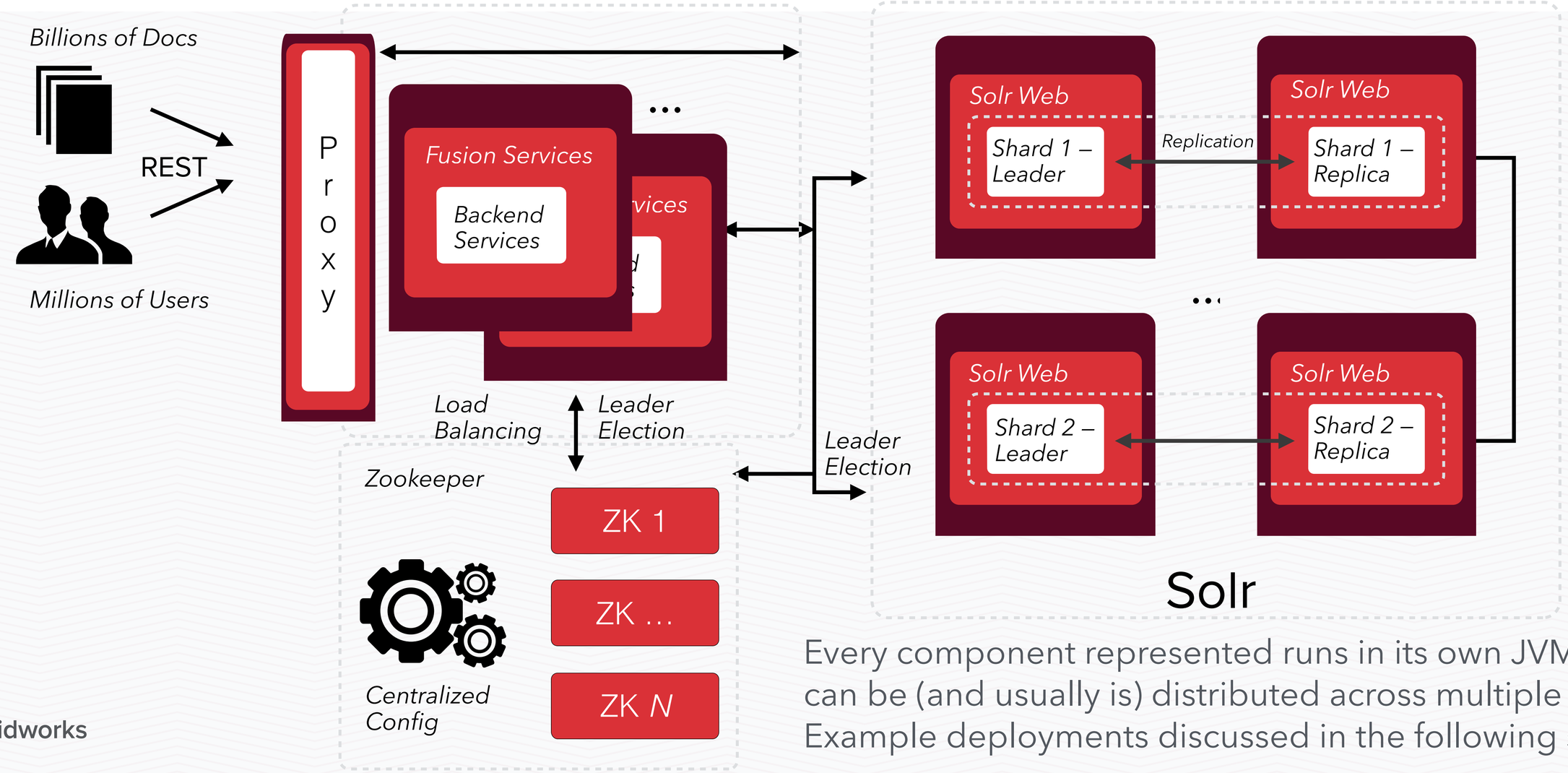
- Fusion leverages Solr in “cloud mode” and Apache Zookeeper for scalability and redundancy
- Fusion + Solr scale linearly with your data
- Our shard-splitting approach means greater control over your scaling needs without having to reindex
- Solr’s maturity, use of Zookeeper and extensive testing minimizes data loss or split brain issues

Apache Solr

+

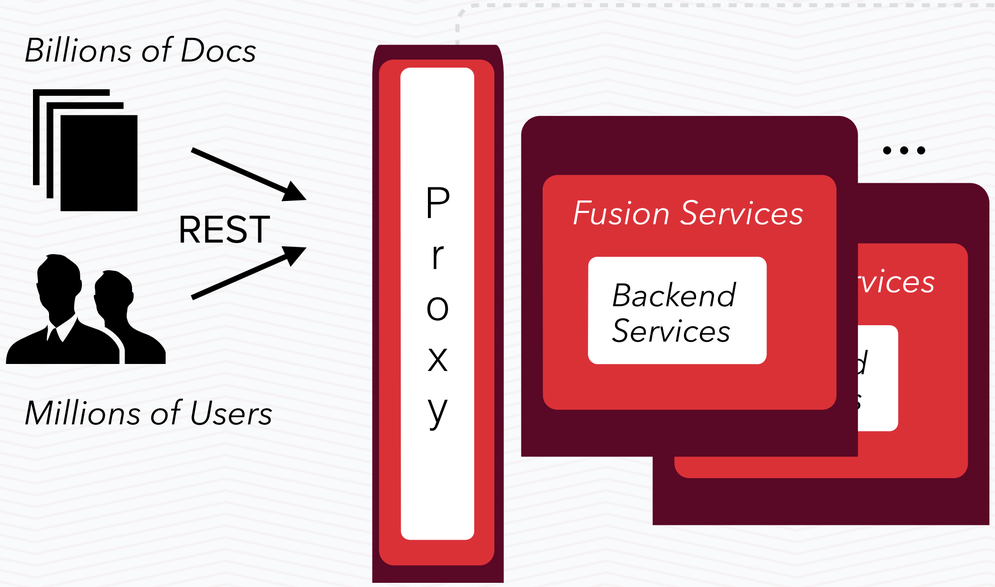
Lucidworks Fusion

Interaction between Fusion, Solr, DataSources and Users



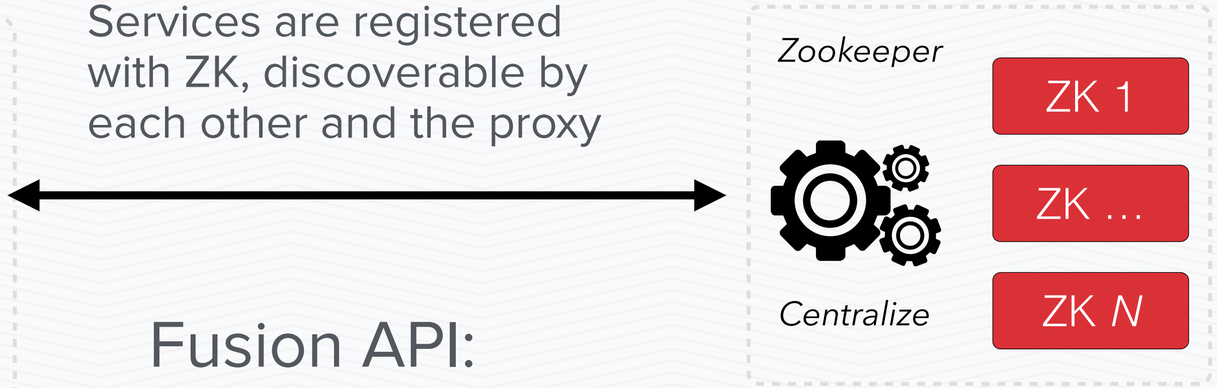
Every component represented runs in its own JVM and can be (and usually is) distributed across multiple servers. Example deployments discussed in the following slides

Fusion's Scalable, Distributed Service-Oriented Architecture



Service discovery and software load balancing among Fusion services happens in the proxy

Services are registered with ZK, discoverable by each other and the proxy



Fusion API: Backend Services

- Connectors
- Pipelines
- Aggregator
- Collections
- Recommender
- Scheduler
- Solr Proxy
- Metrics
- Configuration

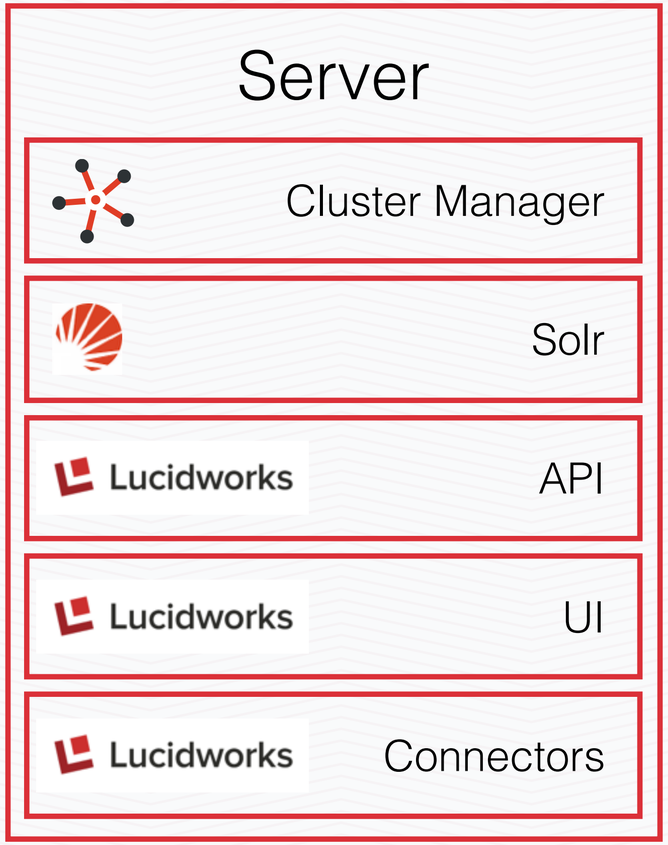
Signals, search logs, application logs and user data is stored in Solr

Fusion Components

Lucidworks Fusion integrates many open source and proprietary components to build a fault-tolerant, flexible search and indexing system.

Lucidworks API	The Fusion API is the heart of the Fusion deployment. All of the Fusion UI and Connectors are controlled through the API, and all communication to Solr is done via the Fusion Proxy which is part of the Fusion API
Lucidworks UI	The Fusion UI presents an intuitive UI to help users manage and monitor their Fusion and Solr deployments
Lucidworks Connectors	The Fusion Connectors enables users to create and modify Fusion Datasources to ingest data from many kinds of sources
Index and Search	Basic indexing and searching is handled by the open source Apache Solr/ Lucene project. Your documents and queries will all eventually be directed to Solr, after being processed and enhanced by Fusion
Cluster Manager	The Cluster Manager - The role of the cluster manager is to coordinate and distribute the operations of the Solr and Fusion clusters. This is implemented by the open source Apache ZooKeeper (ZK) project

Deployment for Prototypes and Dev

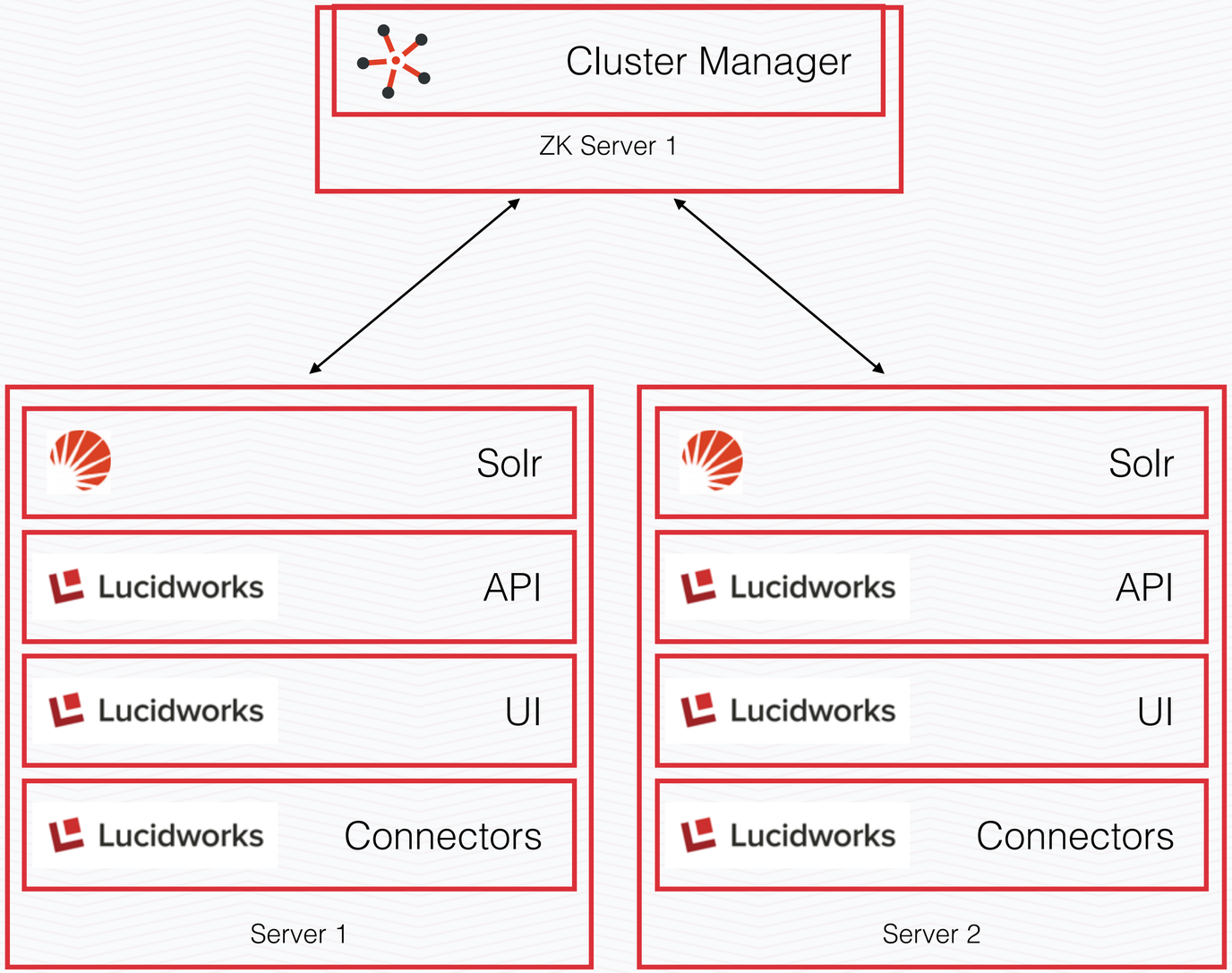


The default Fusion deployment runs all components on a single server, ideal for testing and prototyping.

In this deployment, Fusion still uses the ZooKeeper cluster management service as Fusion deploys in a single-server “cluster”.

Fusion is integrated with Solr and ZooKeeper, storing Fusion index data in Solr and Fusion configuration in ZK.

2-Server Deployment to Test Clustering and Networking



In a 2-server deployment, the full Fusion+Solr stack is deployed on the 2 primary servers, and cluster management lives on its own, independent hardware.

The hardware requirement for the ZK cluster manager is minimal, 1 CPU core, 1GB memory.

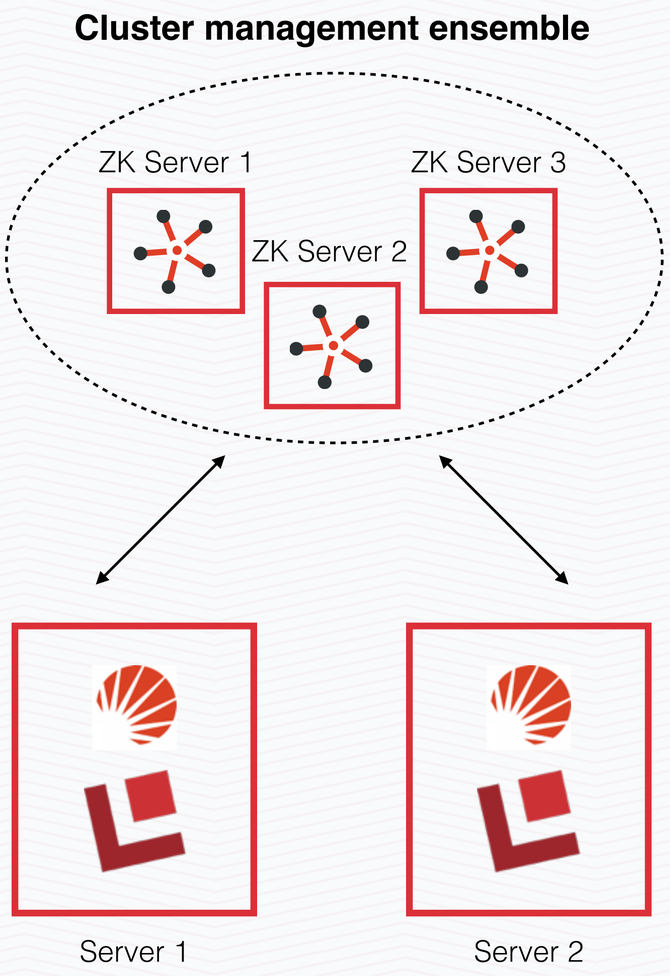
In the event of the ZooKeeper cluster manager failing, queries to Fusion till resolve, but no further indexing is possible.

The Fusion stack has its startup configuration modified to point at your external cluster manager.

After initial startup, ZooKeeper informs Fusion of the other server and they can begin communicating.

Expansion is easy, as all new servers automatically pull configuration data from ZooKeeper.

Highly-Available 2-Server Deployment for Production

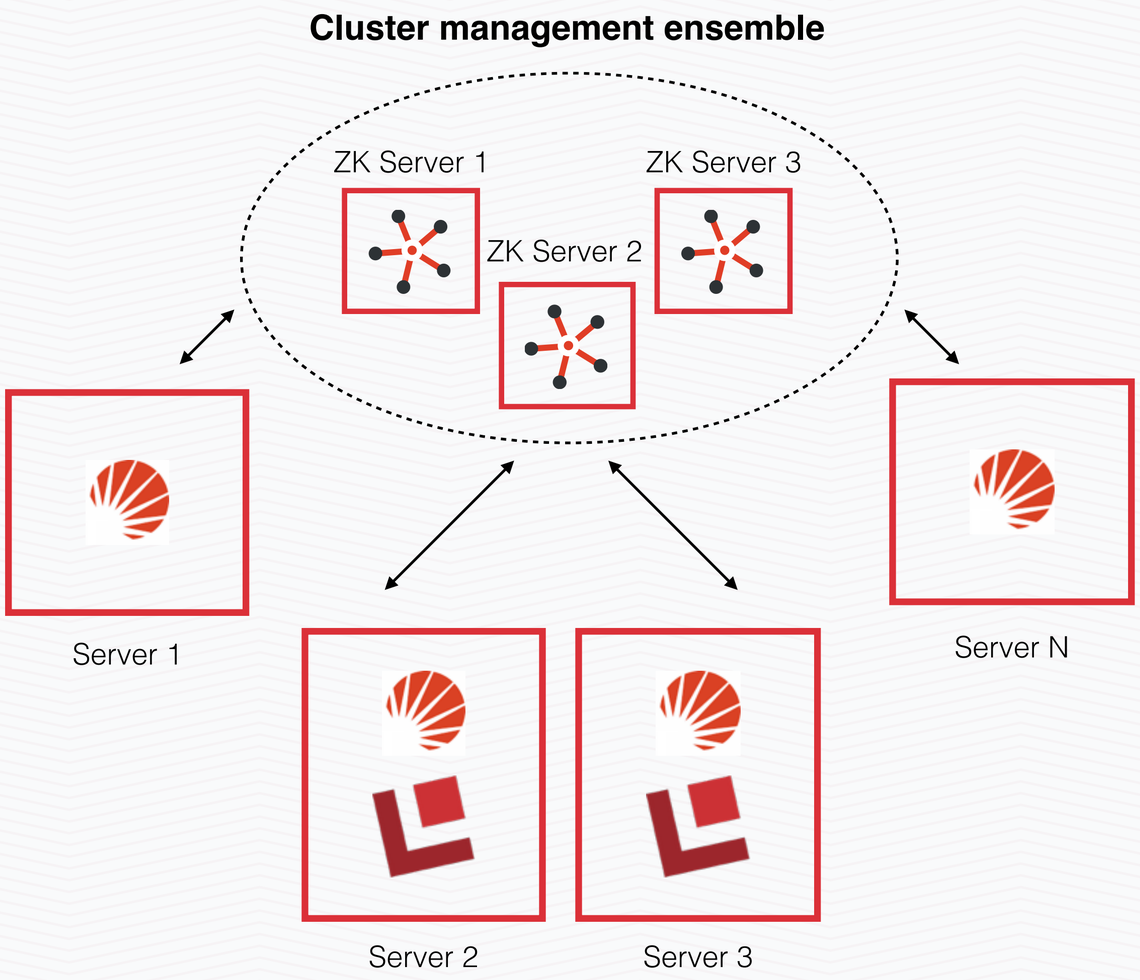


In production, cluster management is set up as an “ensemble”, making both Fusion and cluster management highly-available with no single point of failure.

Fusion is configured with the addresses of all ZK cluster management servers.

With failover for both ZK and Fusion, the failure of any single server will not affect functionality of the cluster, ensuring a highly-available, fault-tolerant Fusion cluster.

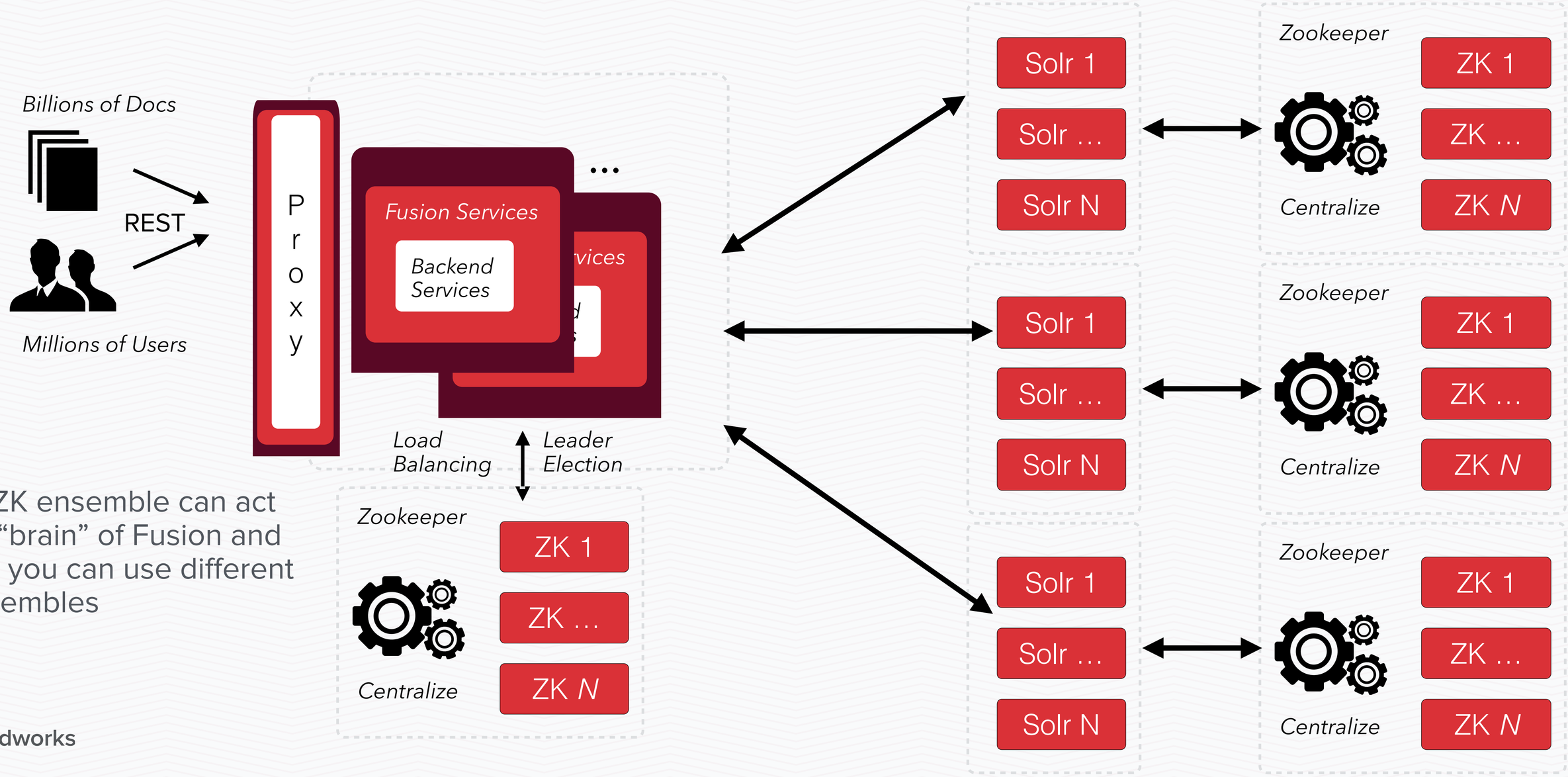
Highly-Available N-Server Deployment for Production



In the N server deployment, you may not need connectors and UI on every server. Additional servers beyond the first two only run Solr, and connect to the ZooKeeper cluster management ensemble just like the first two servers.

ZK and Fusion automatically adds these servers into the cluster, using them to index and serve queries.

Overlaying Multiple SolrCloud Clusters



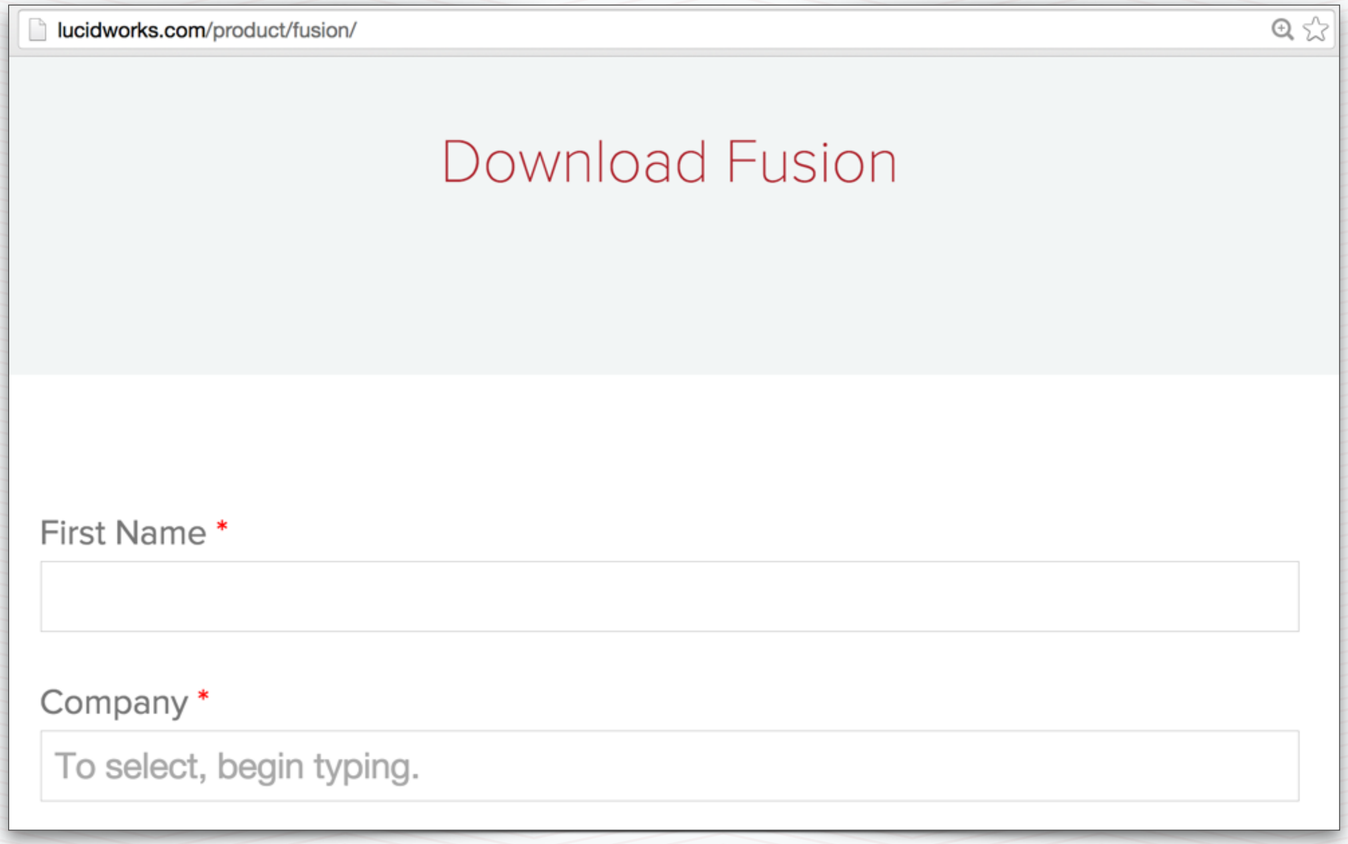
Same ZK ensemble can act as the “brain” of Fusion and Solr, or you can use different ZK ensembles

Lucidworks

How do I get started?

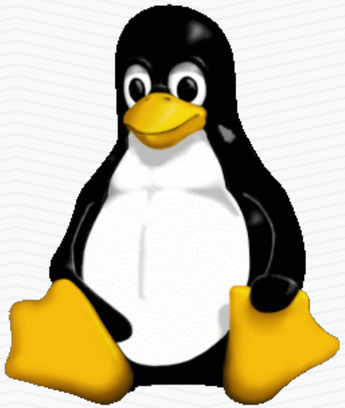
Download Fusion

- New users can download the Fusion install bundle from www.lucidworks.com - registration required
- Existing Lucidworks support customers can download from the Support Portal - login required



Fusion Supported OSs

- Linux distributions that support Java 7 and up - 64 bit
- Windows 7, 8.1, Server 2008, and Server 2012 - 32 and 64 bit
- MacOS 10.7.3 and up
- Download the .zip file for Windows and the .tar.gz for everything else



Java Requirements

- Fusion like Solr is a Java-based application - requires a pre-installed JDK
- Lucidworks recommends Oracle's JDK 1.7 - available here http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
- Prefer Java 1.7u55 or higher to avoid a bug that impacts Lucene indexes (Java 1.7u25 or lower is also acceptable).
- Fusion also supports JDK 1.8
  - JavaScript engines differ between JDK 1.7 and JDK 1.8. Java 1.7 comes with the JavaScript engine "Rhino" from Mozilla, while Java 1.8 comes with JavaScript engine "Nashorn" from Oracle. This difference may impact Fusion Javascript stages. See https://docs.lucidworks.com/display/fusion/Javascript+Index+Stage

Hardware Requirements

- Fusion hardware requirements depend greatly on use case, index size (number of documents), QPS (queries per second) and other factors
- Rules of Thumb:
  - Dev/Testing Environment: minimum 12 GB RAM and 2 CPU cores
  - Small Production: 16 GB RAM and 4 CPU cores
  - Large Production: 32+ GB RAM and 8+ CPU cores
    - Large production environments are likely to be made up of multiple servers with these specs
- These are generalizations. Contact your Lucidworks rep for specific recommendations based on your use case, data load, etc.
- Disk size and number will vary greatly—suffice to say big enough to hold all your indexed data as well any other BLOBS/lookups you wish to store in Fusion

Installing Fusion - Linux/OSX

- Expand Fusion .tar.gz file in the directory of your choice
- For Linux the recommended directory is /opt

```
you@ubuntu:/opt# tar zxvf ./fusion-1.2.0.tar.gz
```

- Same command on Mac

```
mymac:Applications joemac$ tar zxvf ./fusion-1.2.0.tar.gz
```

Starting Fusion

```
user@ubuntu:/opt# cd fusion/bin/
user@ubuntu:/opt/fusion/bin# ./fusion start
2015-02-05 01:11:04Z Starting Fusion Solr on port 8983
2015-02-05 01:11:34Z Starting Fusion API Services on port 8765
2015-02-05 01:11:40Z Starting Fusion UI on port 8764
2015-02-05 01:11:45Z Starting Fusion Connectors on port 8984
```

- Fusion take 5 ports. In addition to the 4 shown above, Zookeeper runs on port 9983
- See https://docs.lucidworks.com/display/fusion/Installing+Lucidworks+Fusion#InstallingLucidworksFusion-RunningFusion for information on starting individual services, how to use Upstart, and run on Windows

Installing and Starting On Windows

- Unzip the package to the directory of your choice
- In the command prompt switch to the install directory and run “fusion.cmd start” from the bin directory

```
C:\fusion\bin>fusion start

Starting Fusion Solr on port 8983

Waiting for 25 seconds, press a key to continue ...

Starting Fusion API Service on port 8765
Using existing C:\fusion\jetty\api\webapps\api

Waiting for 0 seconds, press a key to continue ...

Starting Fusion UI Service on port 8764
Using existing C:\fusion\jetty\ui\webapps\root

Waiting for 0 seconds, press a key to continue ...

Starting Fusion Connectors Service on port 8984
Using existing C:\fusion\jetty\connectors\webapps\connectors

Waiting for 0 seconds, press a key to continue ...

C:\fusion\bin>
```

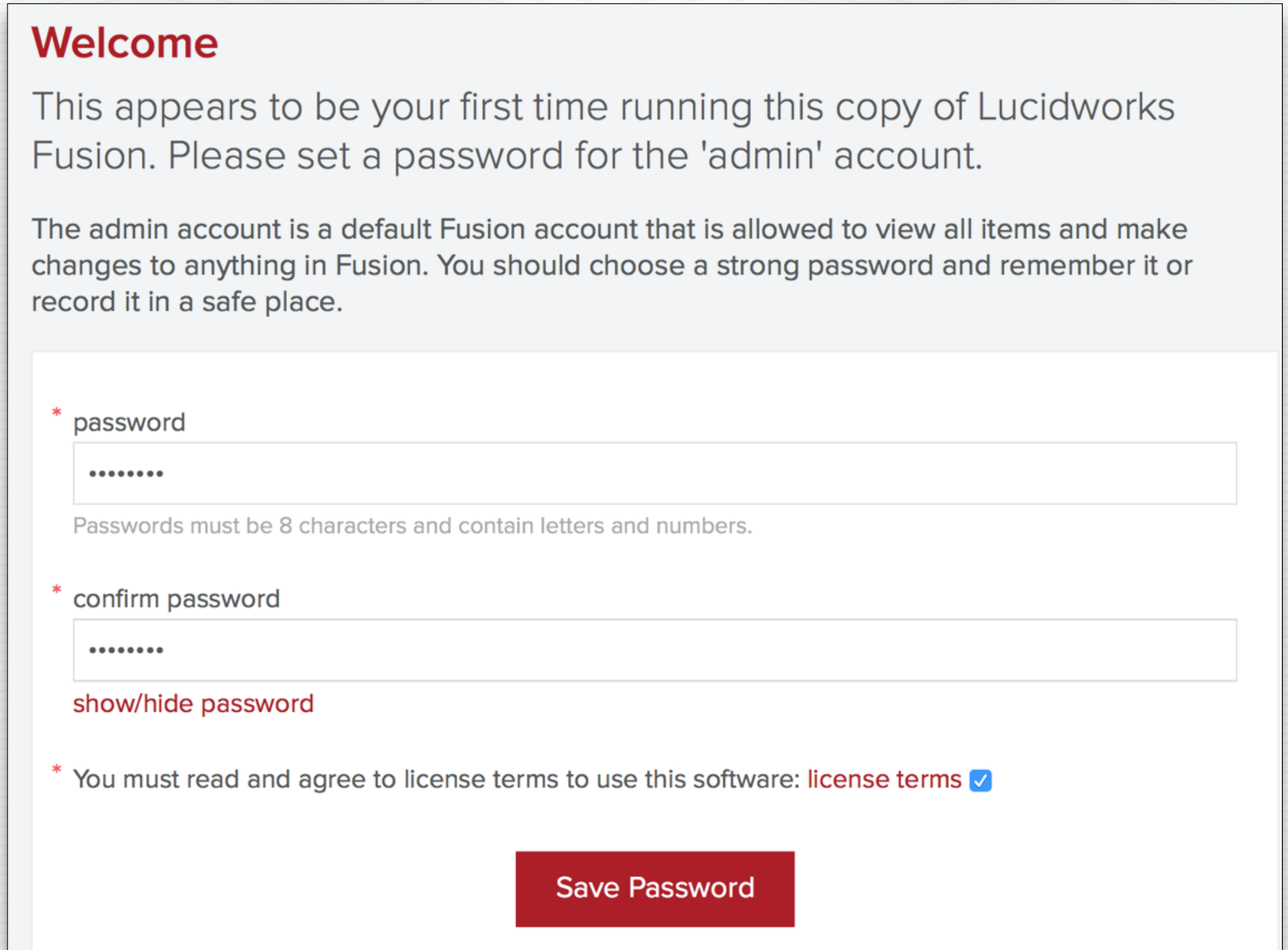
Changing Ports - Fusion and Solr

```
API_PORT=8765
API_STOP_PORT=7765
API_STOP_KEY=fusion

CONNECTORS_PORT=8984
CONNECTORS_STOP_PORT=7984
CONNECTORS_STOP_KEY=fusion

SOLR_PORT=8983
SOLR_STOP_PORT=7983
SOLR_STOP_KEY=fusion

UI_PORT=8764
UI_STOP_PORT=7764
UI_STOP_KEY=fusion
```

- Edit $FUSION/bin/config.sh on Linux/OSX
- Edit $FUSION\bin\config.cm on Windows
- $FUSION=/wherever you installed it

Connecting to Fusion

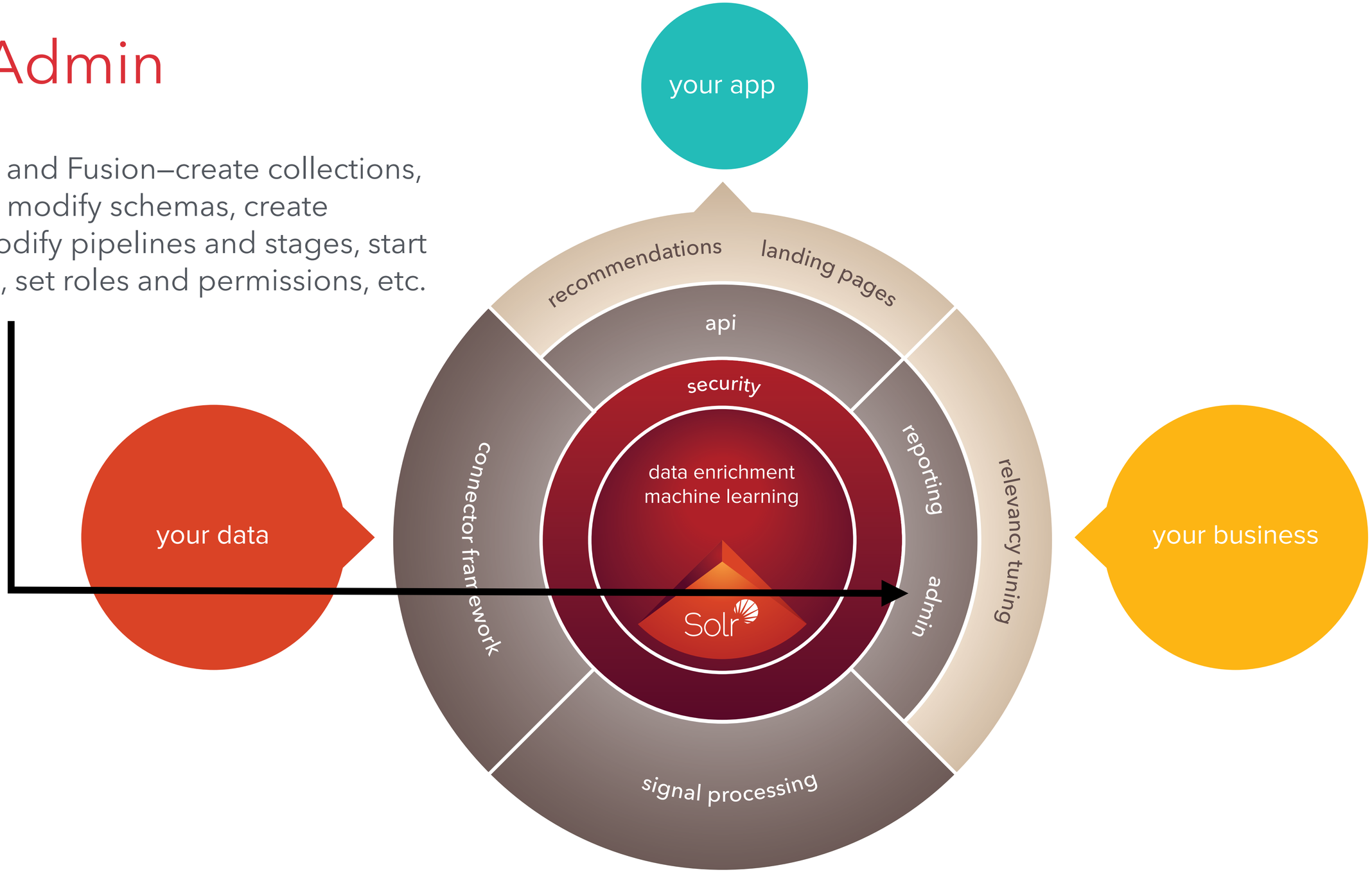
- Connect to http://<fusion_server>:8764 in a web browser
- First time logging in you set the admin password and agree to license terms



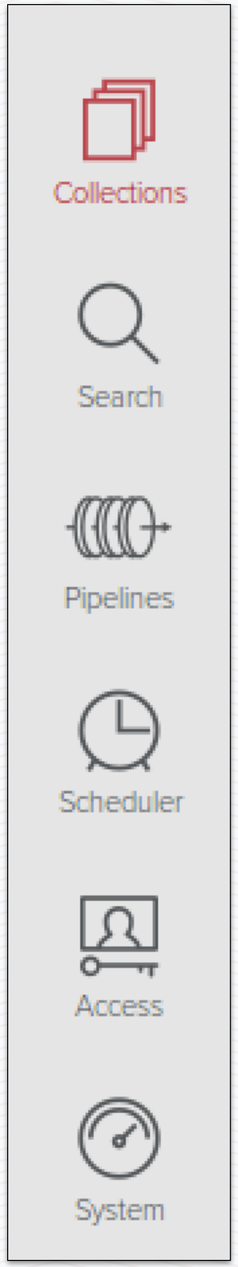
Navigation Basics and Administration

Fusion Admin

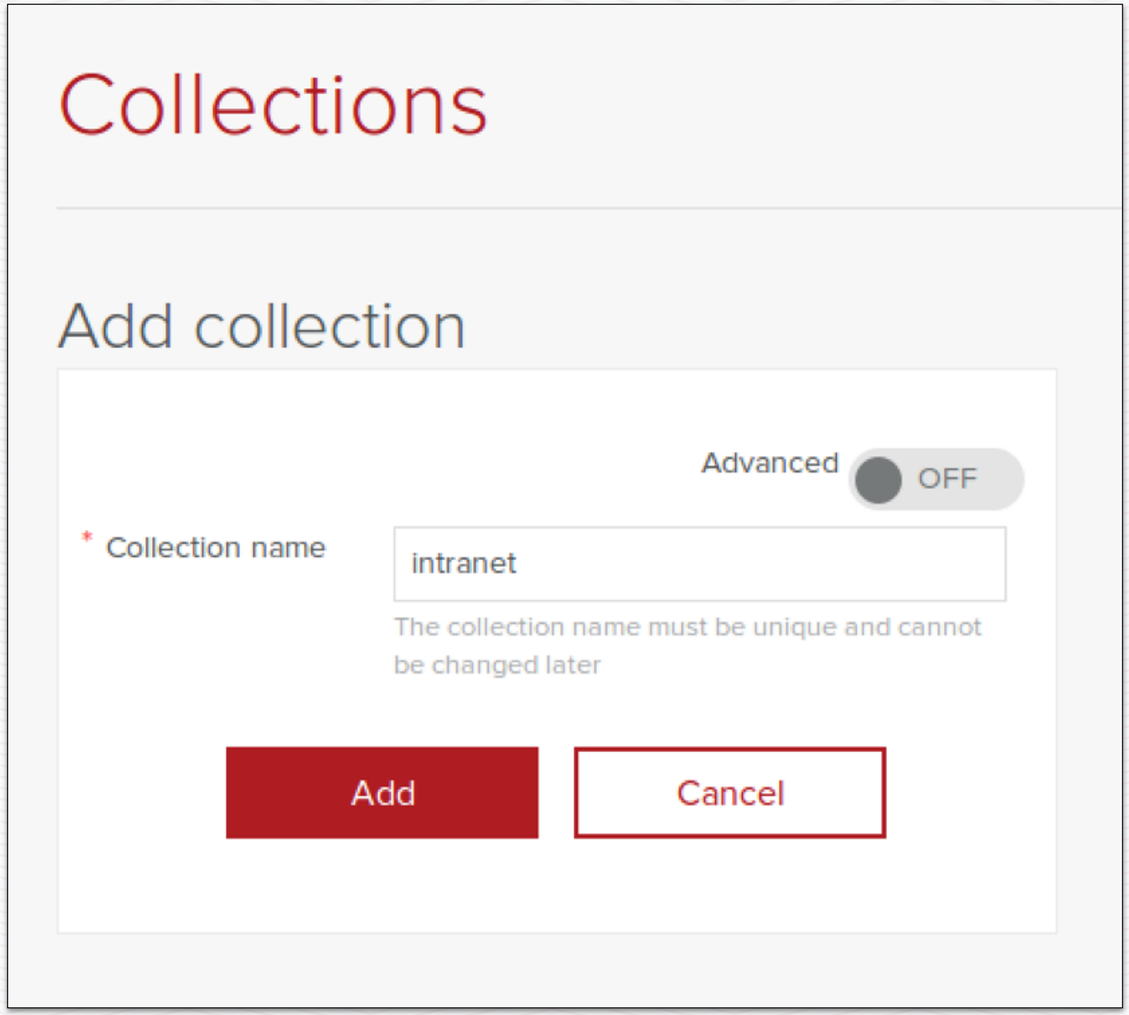
Administer Solr and Fusion—create collections, upload configs, modify schemas, create datasources, modify pipelines and stages, start and stop crawls, set roles and permissions, etc.



Admin UI - Collections

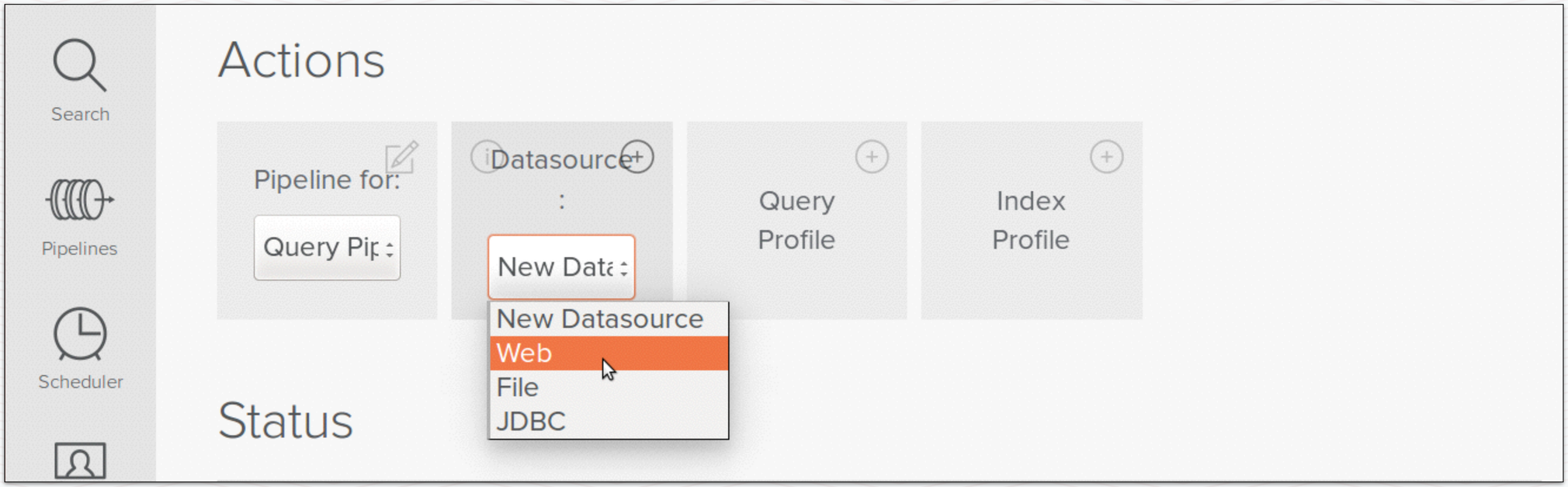


- Add and modify collections
- Give your collection a name and click add



Lucidworks

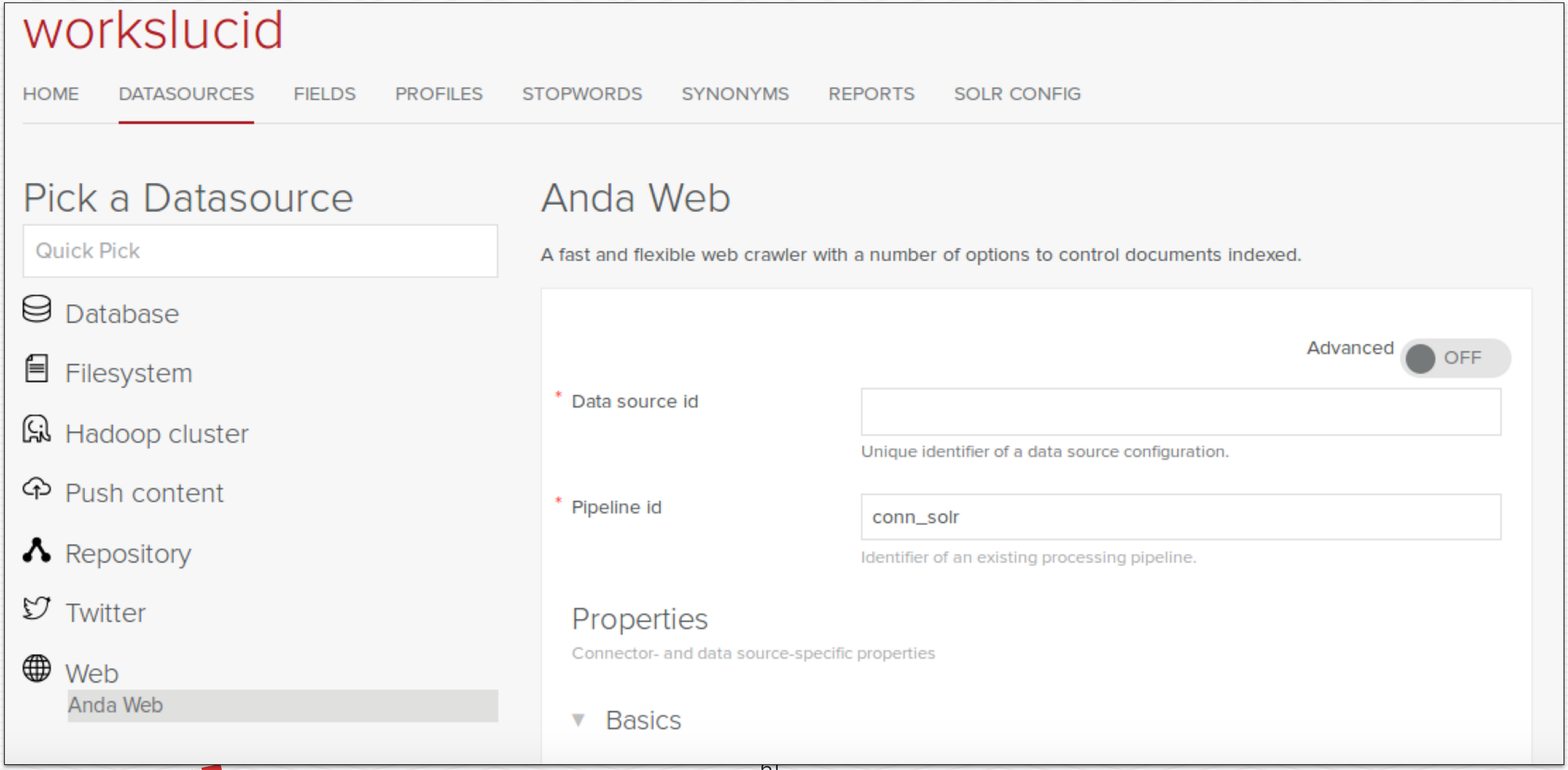
Collections - New Datasource



- You can easily add datasources to your collection
- Select your type from the dropdown, this will bring up the data source interface

Lucidworks

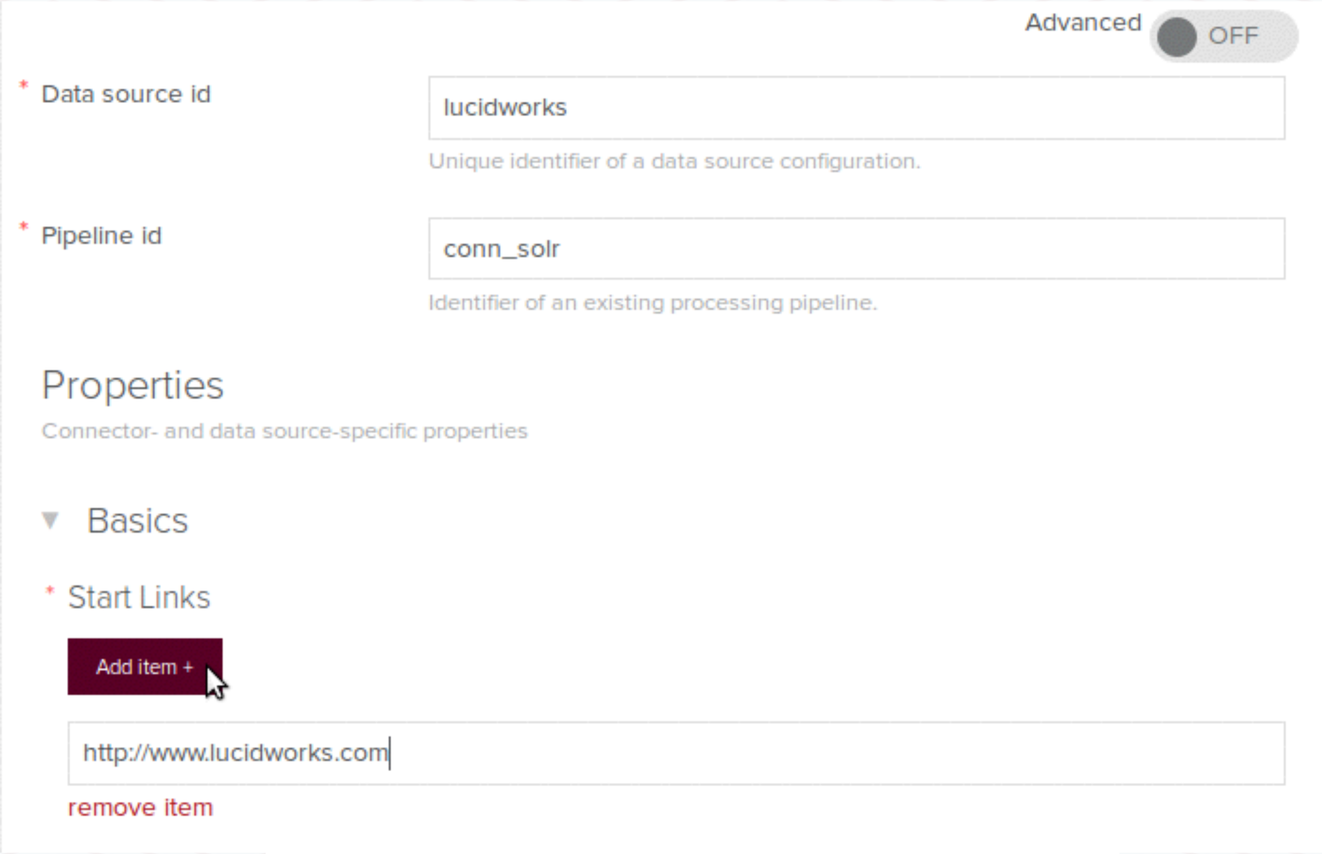
Collections - Datasource UI



Lucidworks

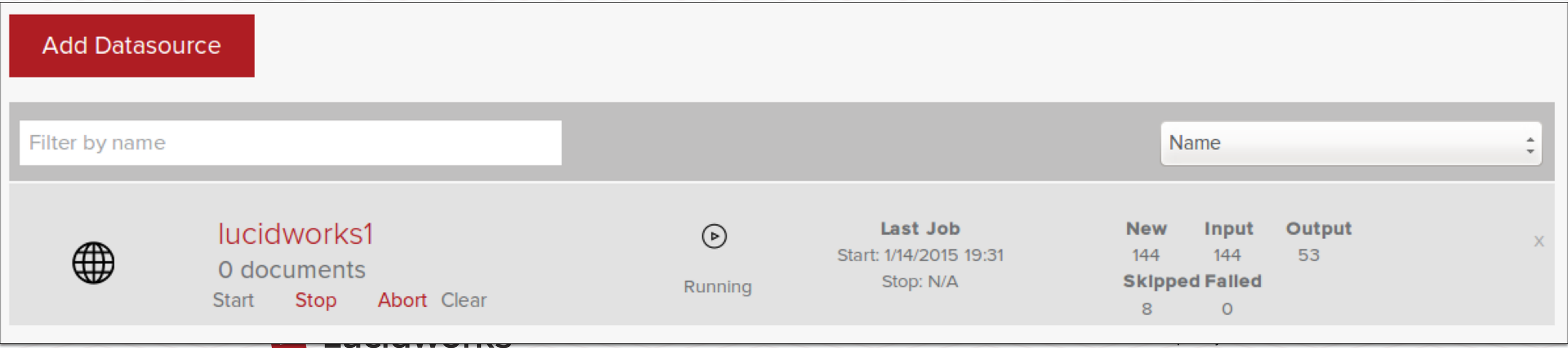
Web Connector

- Give your web crawler a unique ID
- Specify index pipeline
- Click Add item then Add datasource



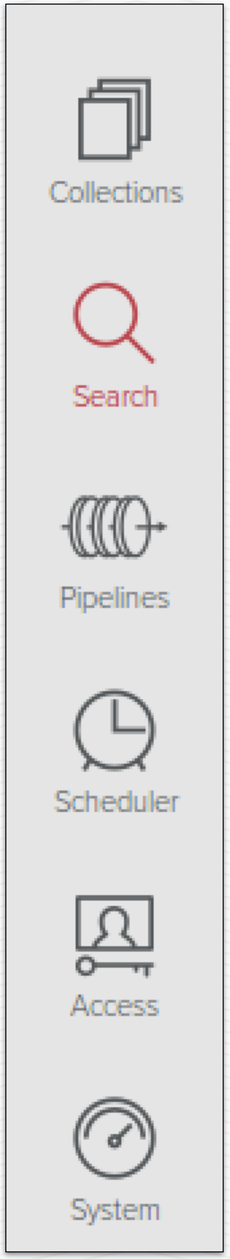
Lucidworks

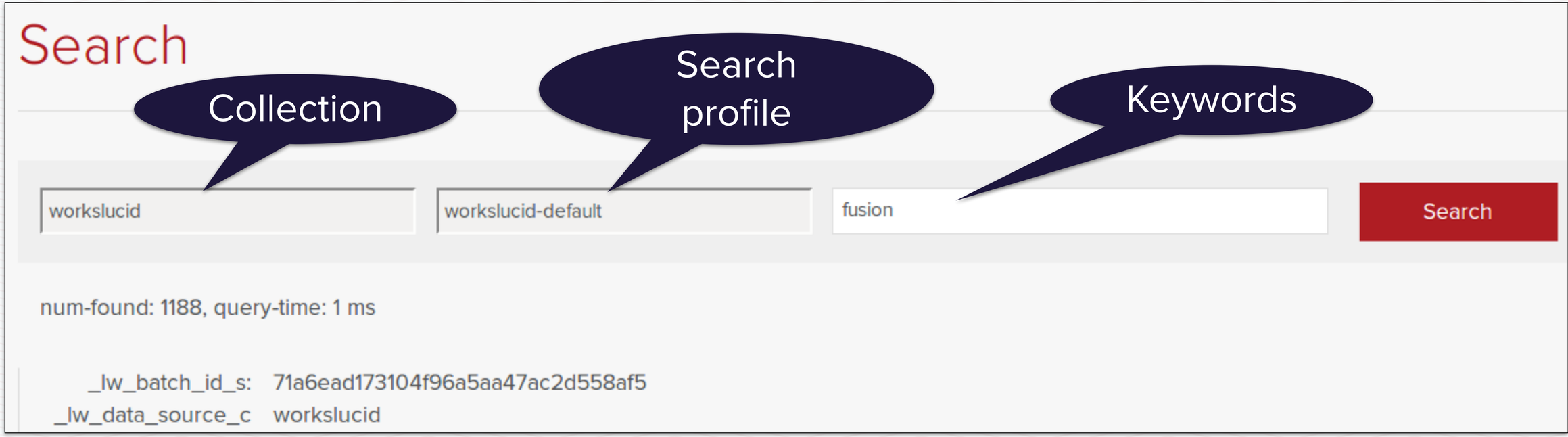
Run Your Connector



Use Search to Test

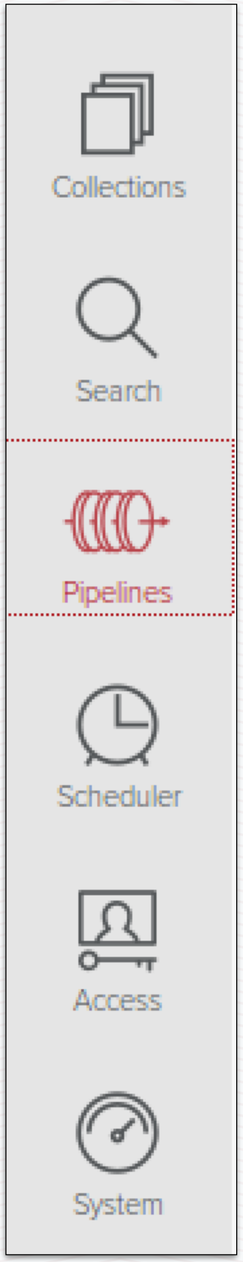
- Specify a collection and search profile

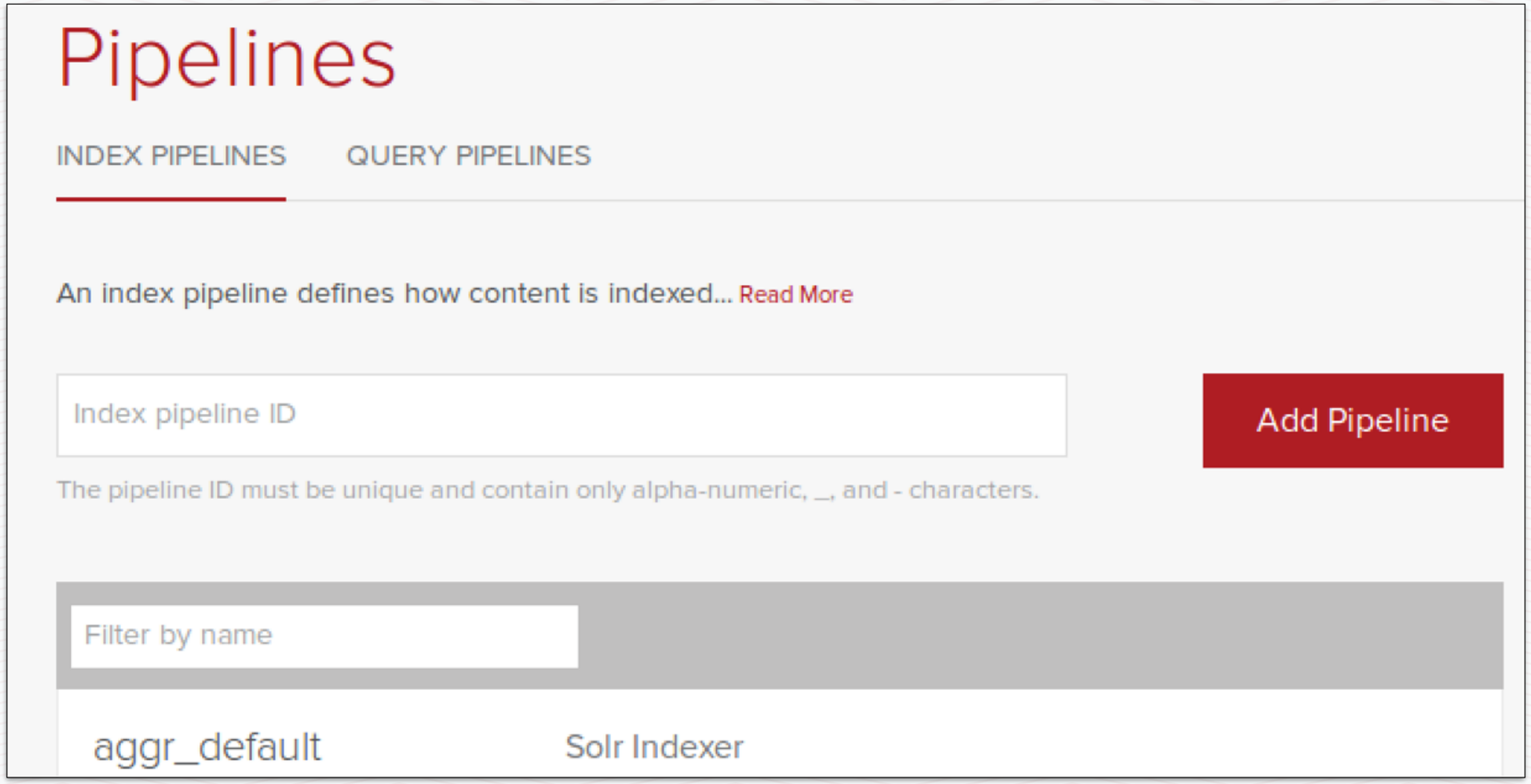




Admin Interface - Pipelines

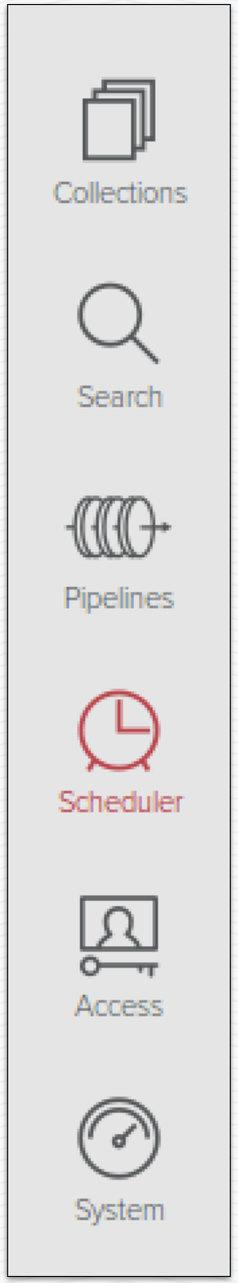
- Edit and configure pipelines - index and query
- We will see more on Pipelines in the Pipelines section



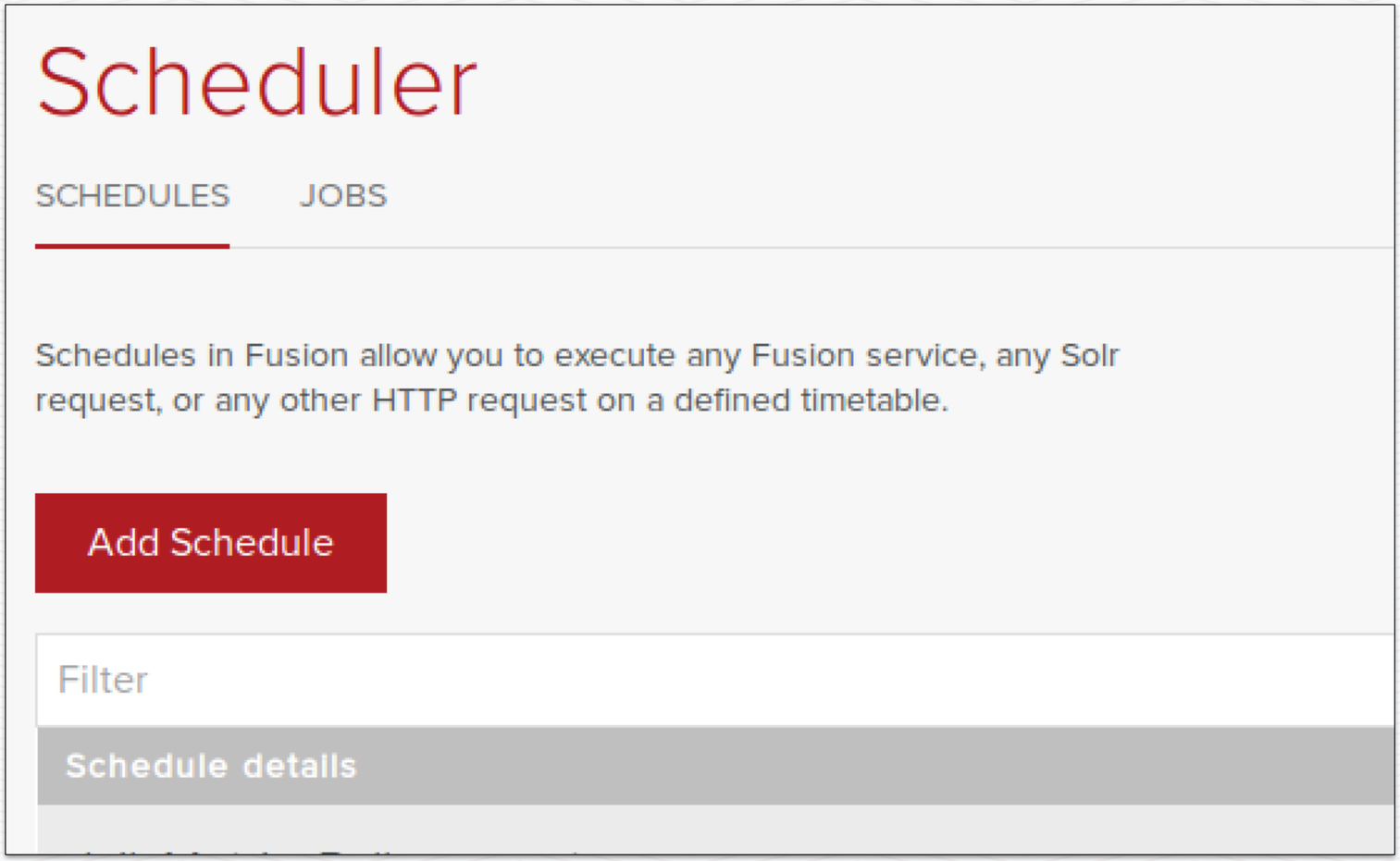


Lucidworks

Admin Interface - Scheduler

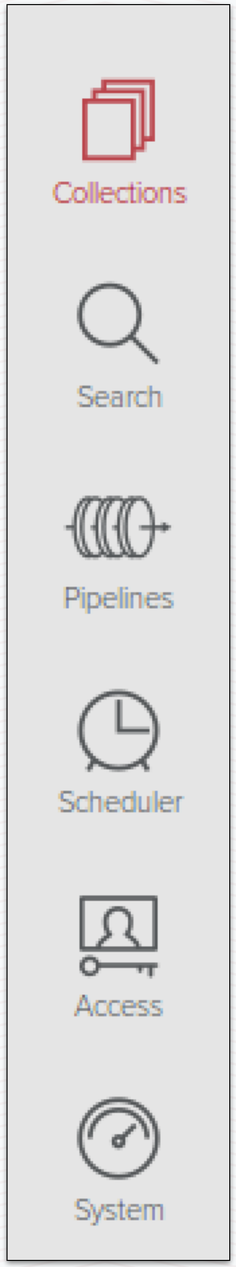


- What the screenshot says
- We will see more on this in the Jobs/Scheduler section

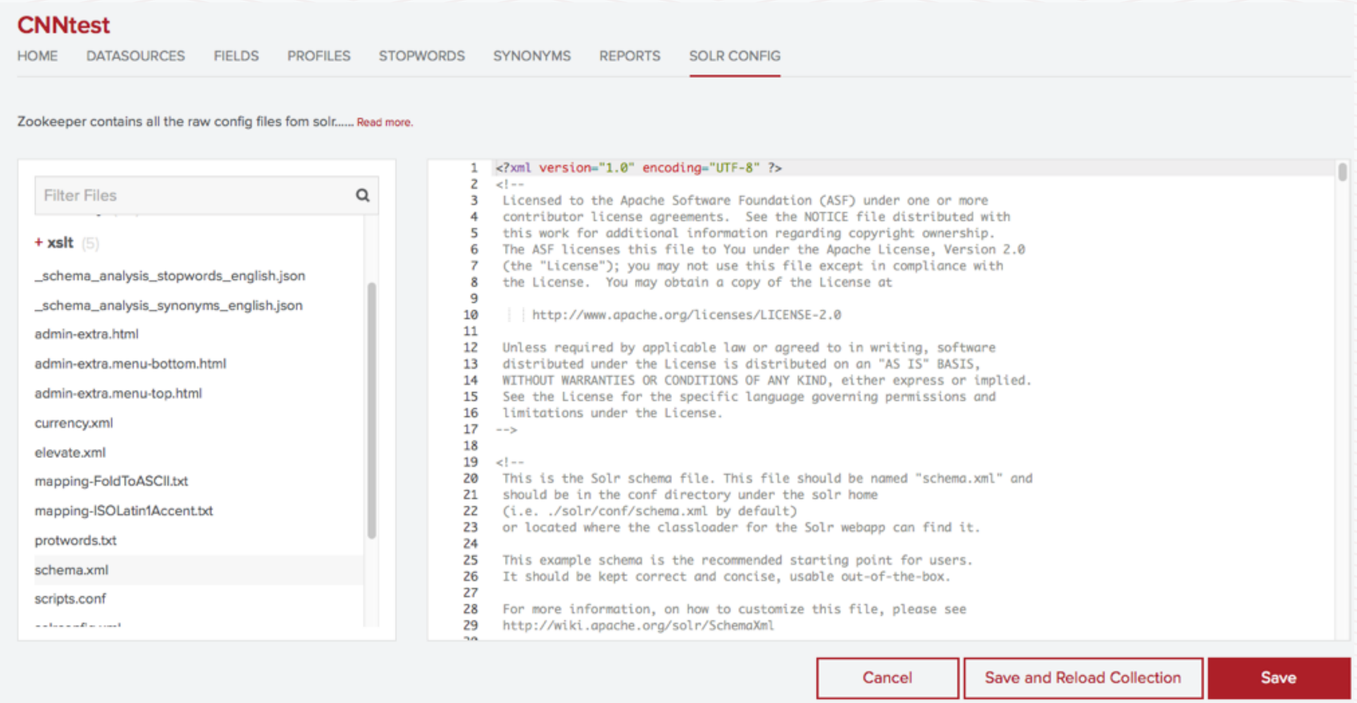


Lucidworks

Admin Interface - Solr Config Editor



- Edit Solr Config Files without shell access to Solr and ZK client



Lucidworks

Demo and Lab 1

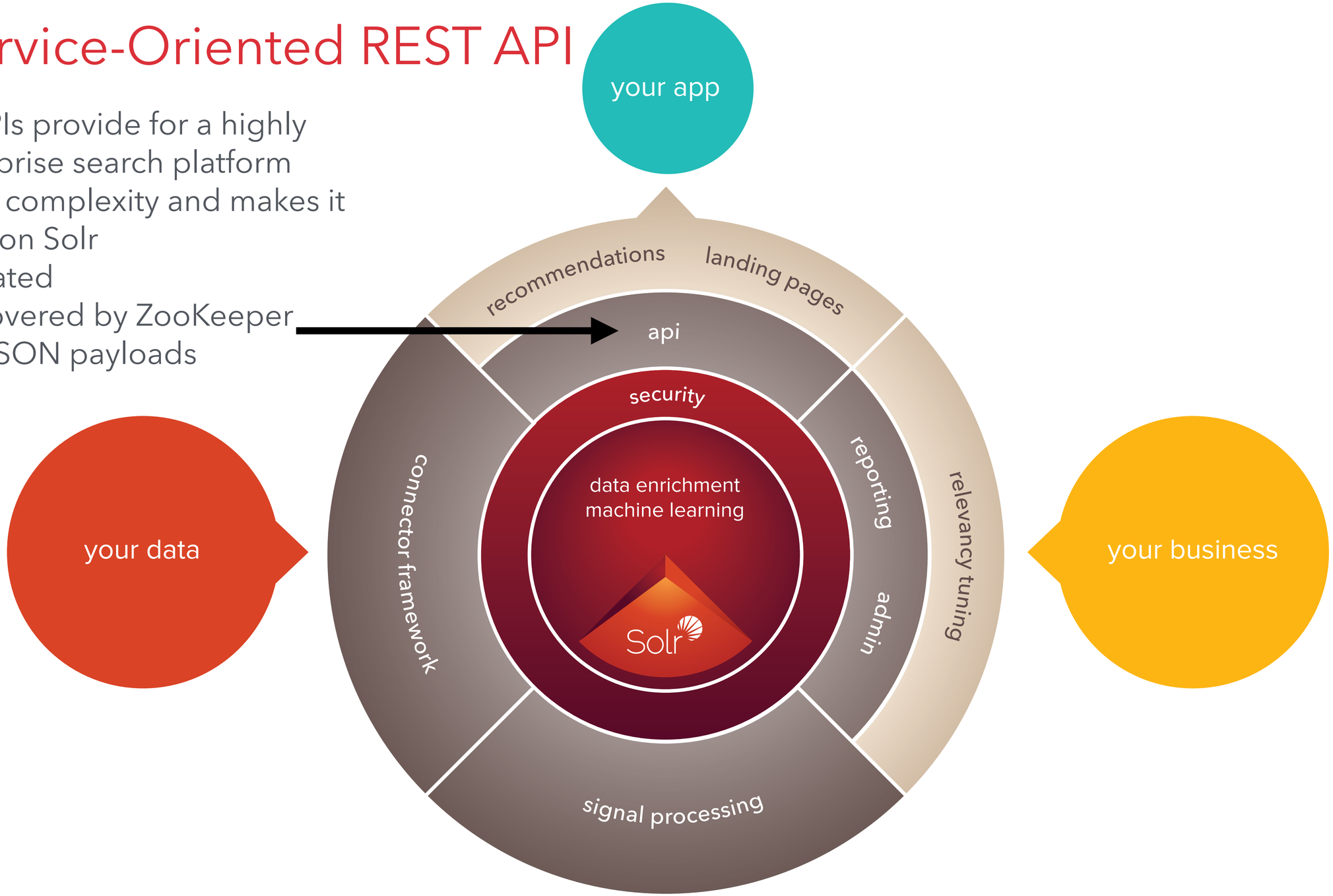
Demo 1 and Hands-on Lab 1

- Demo Harbor Cruise of Fusion
- Hands-on Lab (can be combined with Lab2)
  - Install Fusion (if necessary)
  - Run Fusion
  - Go to the Admin UI
  - Review the various components of Fusion—we will cover them in detail soon
  - Create a collection; crawl a website; view search results

Fusion and Solr APIs

Endpoint/Service-Oriented REST API

- Fusion plus Solr APIs provide for a highly configurable enterprise search platform
- Fusion UI abstracts complexity and makes it easy to build apps on Solr
- Full security integrated
- Distributed & discovered by ZooKeeper
- Human-readable JSON payloads



List of Fusion APIs

- Blob Store API; Collection Features API; Collections API; Configurations API; Connector Datasources API; Connector History API; Connector JDBC API; Connector Jobs API; Connector Plugins API; Connector Status API; Connectors Crawl Database API; History API; Index Pipelines API; Index Profiles API; Index Stages API; Nodes API; Query Pipelines API; Query Profiles API; Query Stages API; Realms API; Recommendations API; Reporting API; Roles API; Scheduler API; Search Cluster API; Sessions API; Signals Aggregator API; Signals API; Solr and SolrAdmin APIs; Stopwords API; Synonyms API; System API; Usage API; User API

- Everything in the Fusion UI uses an API. You can access extended functionality by directly using the REST API. See: https://docs.lucidworks.com/display/fusion/REST+API+Reference

- Introspect API lists all available REST APIs and their endpoints, along with supported methods and any applicable path/query parameters. Usage: `curl -u user:pass http://localhost:8764/api/apollo/introspect`

REST API Examples

Send
two signal events
to record user
clicks

- curl -X POST -H 'Content-type:application/json' -d '[{"params": {"query": "Televisiones Panasonic 50 pulgadas", "filterQueries": ["cat00000", "abcat0100000", "abcat0101000", "abcat0101001"], "docId": "2125233"}, "type":"click", "timestamp": "2011-09-01T23:44:52.533000Z"}, {"params": {"query": "Sharp", "filterQueries": ["cat00000", "abcat0100000", "abcat0101000", "abcat0101001"], "docId": "2009324"}, "type":"click", "timestamp": "2011-09-05T12:25:37.420000Z"}]' http://localhost:8764/api/apollo/signals/docs?commit=true

More REST API Examples

Upload a postgres driver to be used by a collection named docs

```
curl -u user:pass -X POST --form file=Create a new role to allow access to the Admin UI and full control over role definitions and user accounts@/path/postgresql-9.3-1101.jdbc4.jar http://localhost:8764/api/apollo/connectors/plugins/lucid.jdbc/resources/jdbc?collection=docs
```

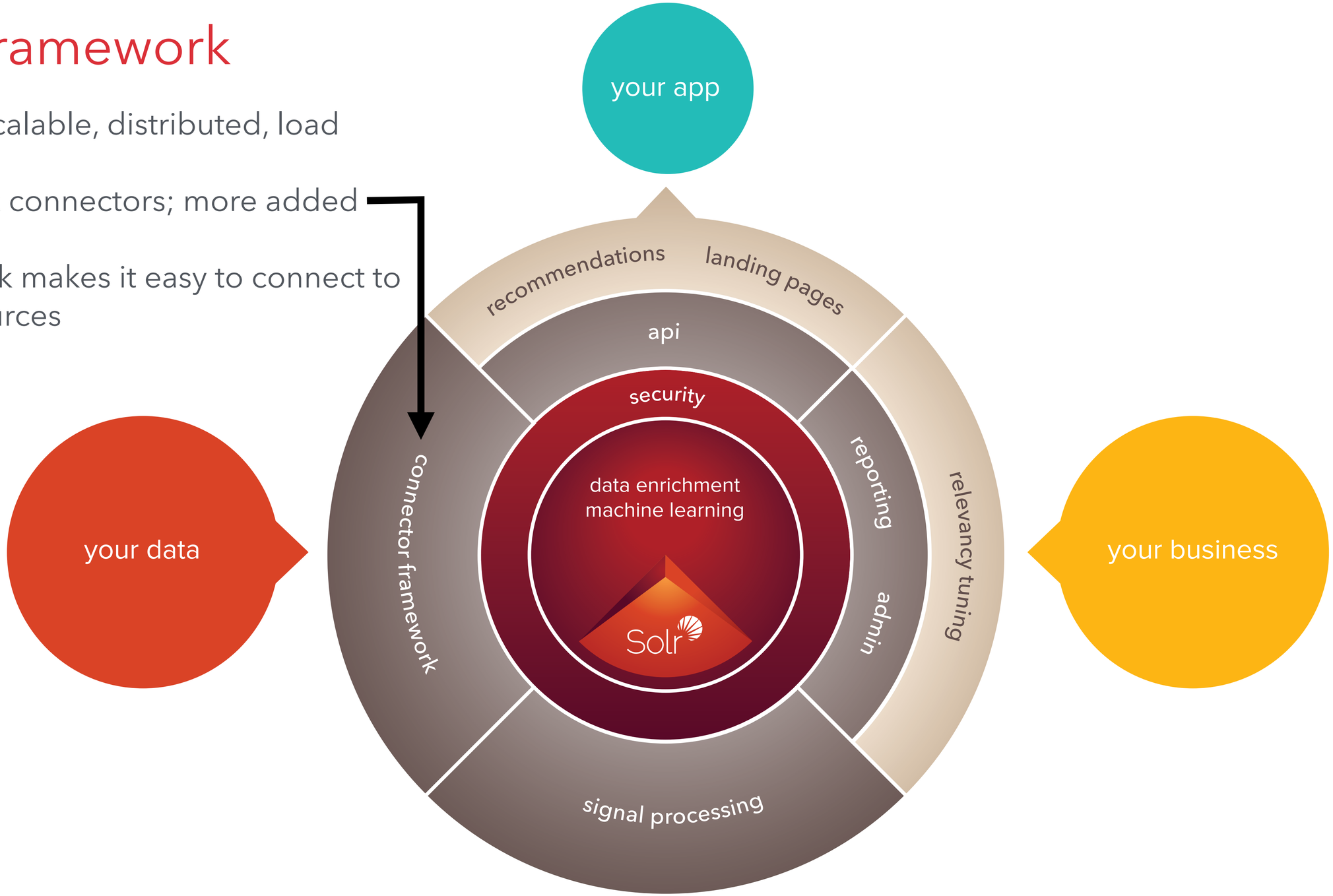
Create a new role to allow access to the Admin UI and full control over role definitions and user accounts

```
curl -u user:pass -X POST -H 'Content-type: application/json' -d '{"name":"userAdmin", "desc":"Gives user update access only", "permissions":["users, roles:*"], "extends":["ui-user"]}' http://localhost:8764/api/roles
```

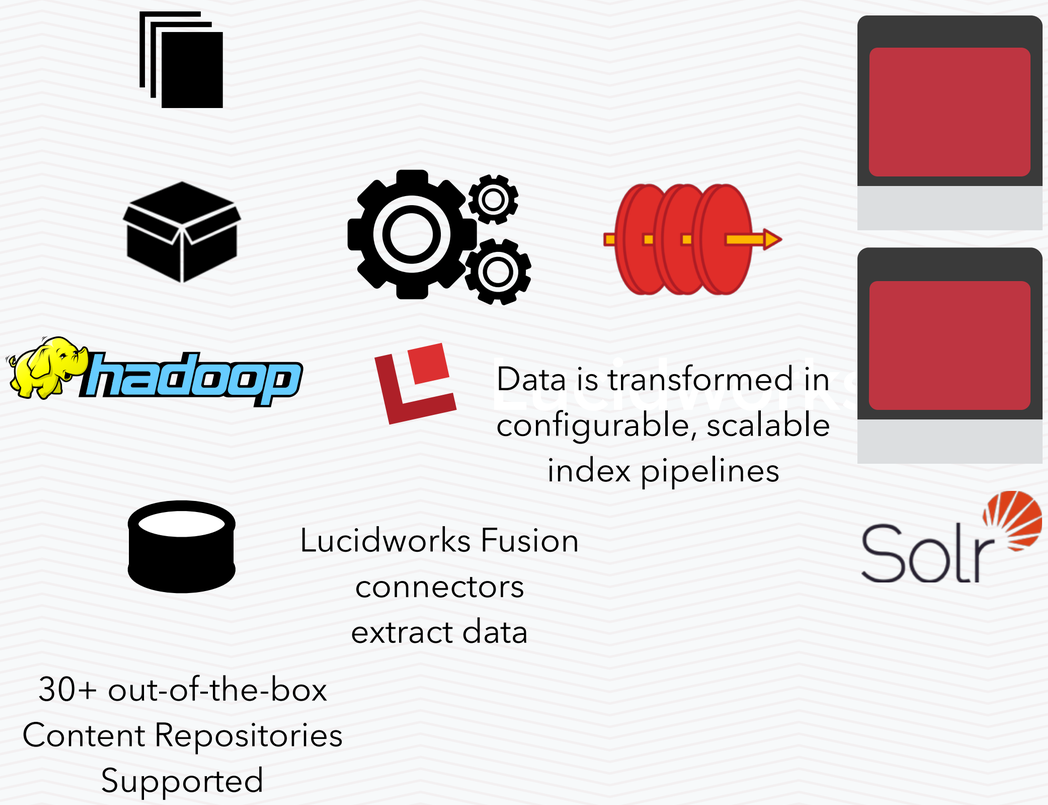
How do I get data into Solr?

Connector Framework

- Backend service, scalable, distributed, load balanced
- 30+ out-of-the-box connectors; more added every release
- Powerful framework makes it easy to connect to additional data sources



Connectors, Datasources and Index Pipelines



- Within Fusion, a datasource is used to create a specific connector instance that is able to connect to a defined repository and collect content for indexing via an index pipeline.
- Datasources are specific to a collection.

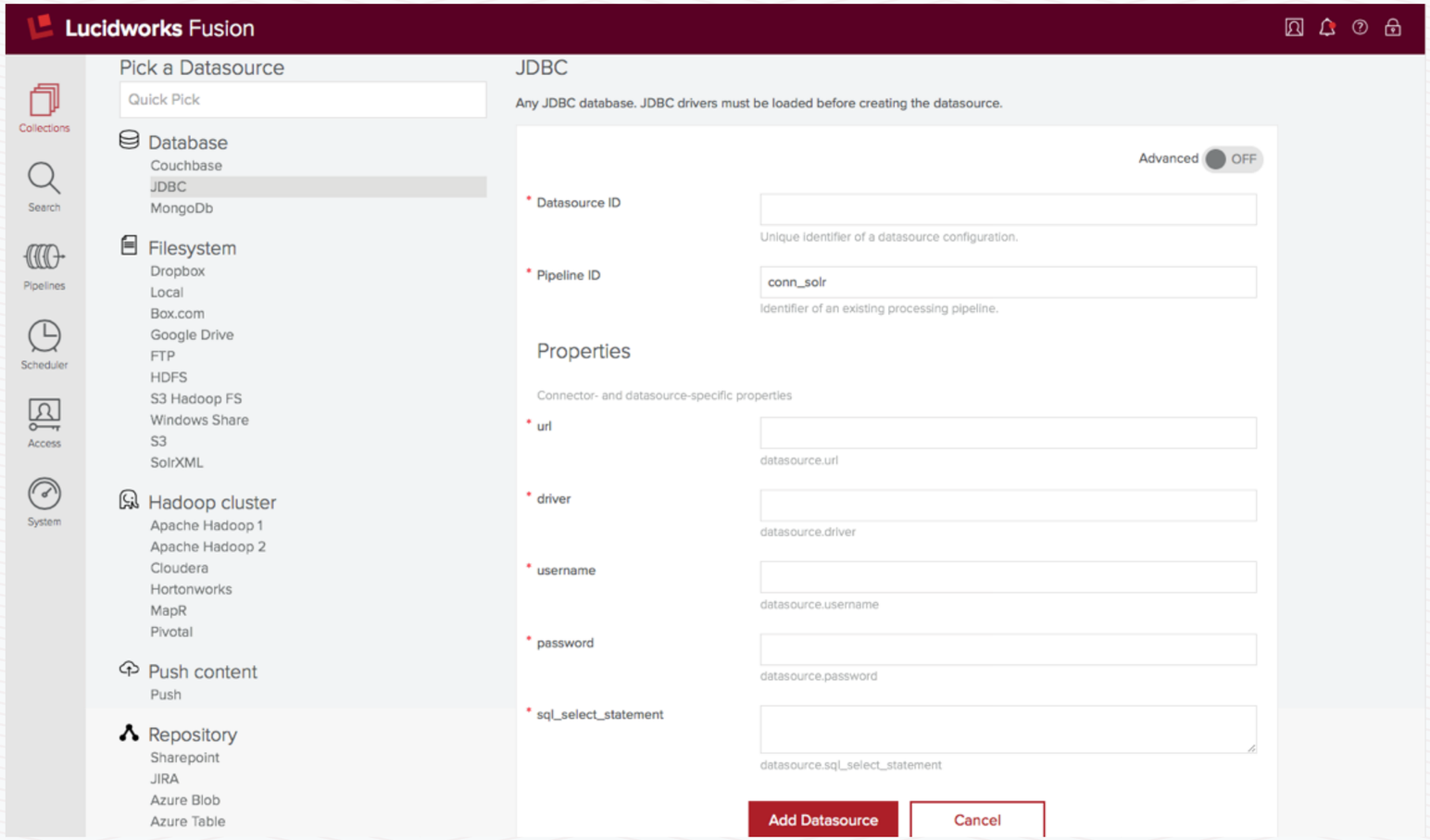
- An index pipeline defines how content is indexed. Every pipeline is made up of a number of stages to perform certain types of transformations or processing on each incoming document.
- Index pipelines and stages are not specific to any particular collection and may be reused across multiple datasources and/or collections.
- Datasource definition associates a specific index pipeline with the datasource

Out-of-the-box Connectors/DataSources

- **Database:** Couchbase, MongoDb, JDBC
- **Filesystem:** DropBox, Local, Box.com, Google Drive, FTP, HDFS, S3 Hadoop FS, Windows Share, S3, SolrXML
- **Hadoop Cluster:** Hortonworks, Cloudera, MapR, Pivotal, Apache Hadoop 1 and 2
- **Push:** Content pushed to Solr
- **Repository:** Sharepoint, JIRA, Azure Blob, Azure Table, Solr Index, Subversion, Drupal (1.3) and Github (1.3)
- **Script:** Javascript
- **Social:** Jive, Slack, Twitter Search, Twitter Stream
- **Web:** Anda

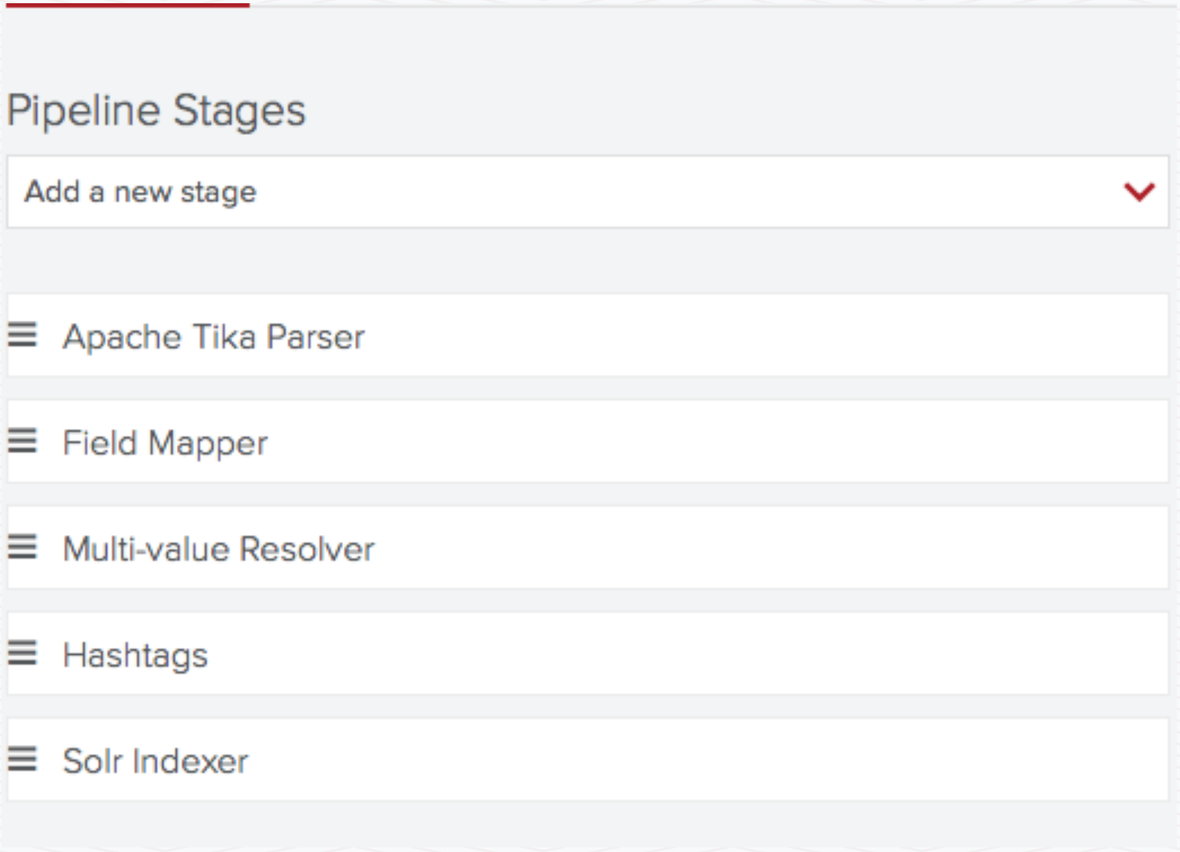
DataSource Definition

- The definition of a datasource includes several details, including:
  - Connector plugin to use
  - Specific plugin type
  - Collection to which documents are indexed
  - Index pipeline used
  - Information on how to connect to the repository and navigate the content.



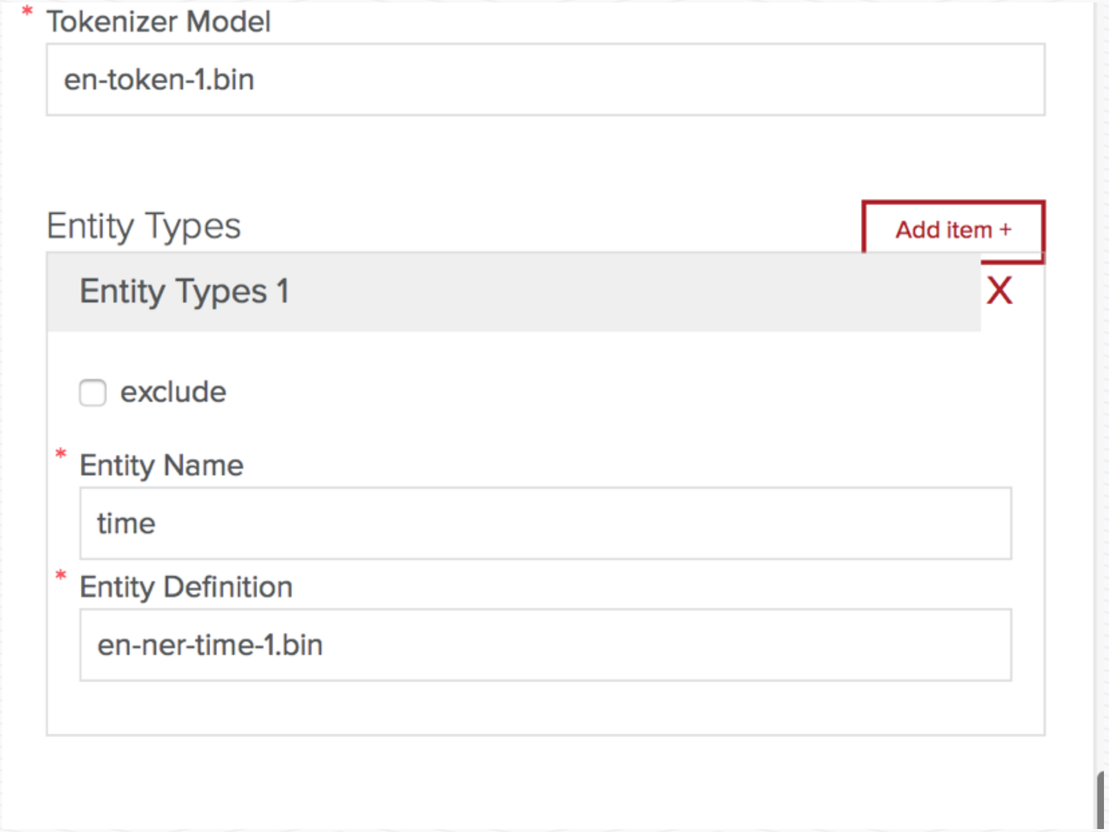
Index Pipelines

- Transform documents that flow through connector
- Separating this from Solr provides enormous flexibility
  - Crawling and parsing eat resources. Complex computations and lookups on external sources (which load network) can be separated from the Solr Cluster
  - Connectors can round-robin between instances
  - Easier to maintain and upgrade



Index Pipeline Stages

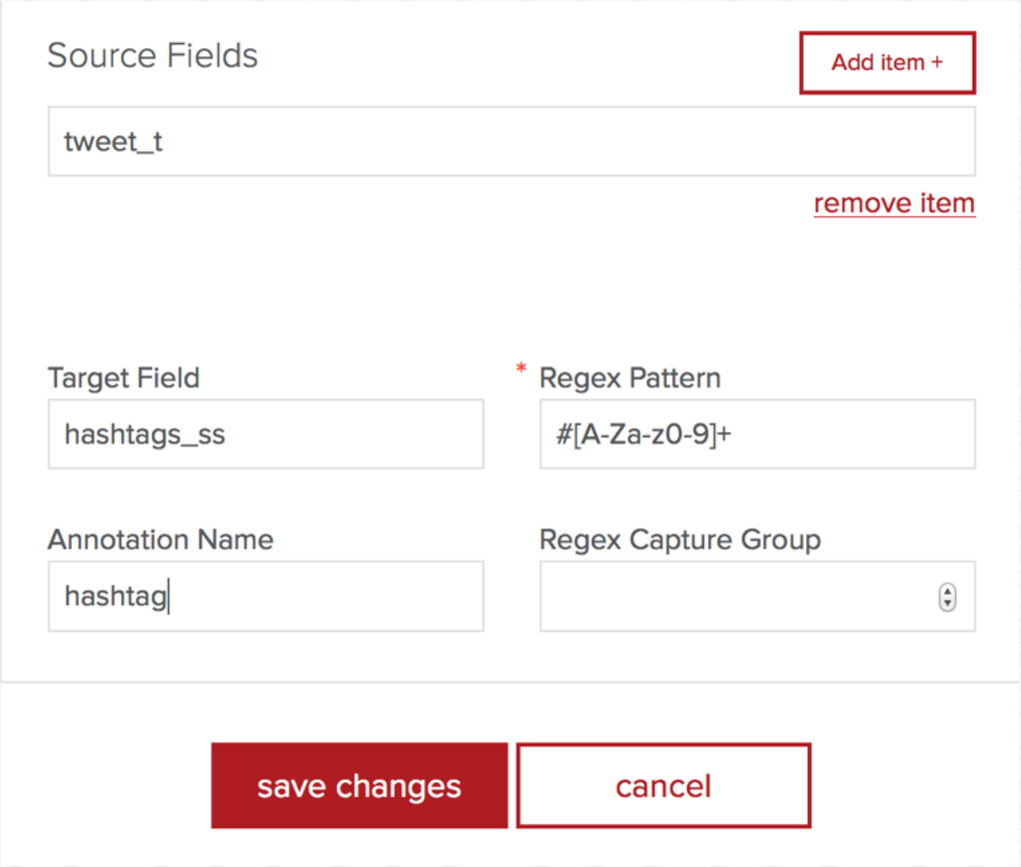
Fusion ships with many out-of-the-box stages that can be used to quickly build and configure your own pipelines



- **Field Mapping Stage:** powerful ability to do advanced mapping of fields from incoming documents to defined fields that exist in the schema.
- **Multi-value Resolver:** resolve multiple field values into a single value based on a set of pre-defined rules (PICK_MAX, PICK_FIRST, etc.)
- **OpenNLP NER Extractor** uses Apache OpenNLP project to extract entities from documents according to pre-trained models stored in Fusion's BLOB store.
- **Indexing RPC Stage** allows calling an external service and merging results retrieved from that service with a document being processed by the pipeline. Calls to the external system are made for each document as it is being processed in the pipeline.

Index Pipeline Stages—continued

- **Regular Expression Extractor** stage type allows extracting entities from documents based on matching regular expressions, and copy them to another field defined in the properties.
- **Regular Expression Filter** allows removing a field based on data found in the field; this filter will ensure the data will not find it's way into the index.
- **Apache Camel Pipeline stage** allows escaping from the pipeline, perhaps to integrate a processing stage in another app, and then returning documents back to the pipeline.
- **Apache Tika Parser** index stage type includes rules for parsing documents with Apache Tika. Fusion uses Tika v1.6; this stage added the ability to parse CSV or TSV files and index rows of these files as individual documents.



Javascript Stage—Swiss Army Knife

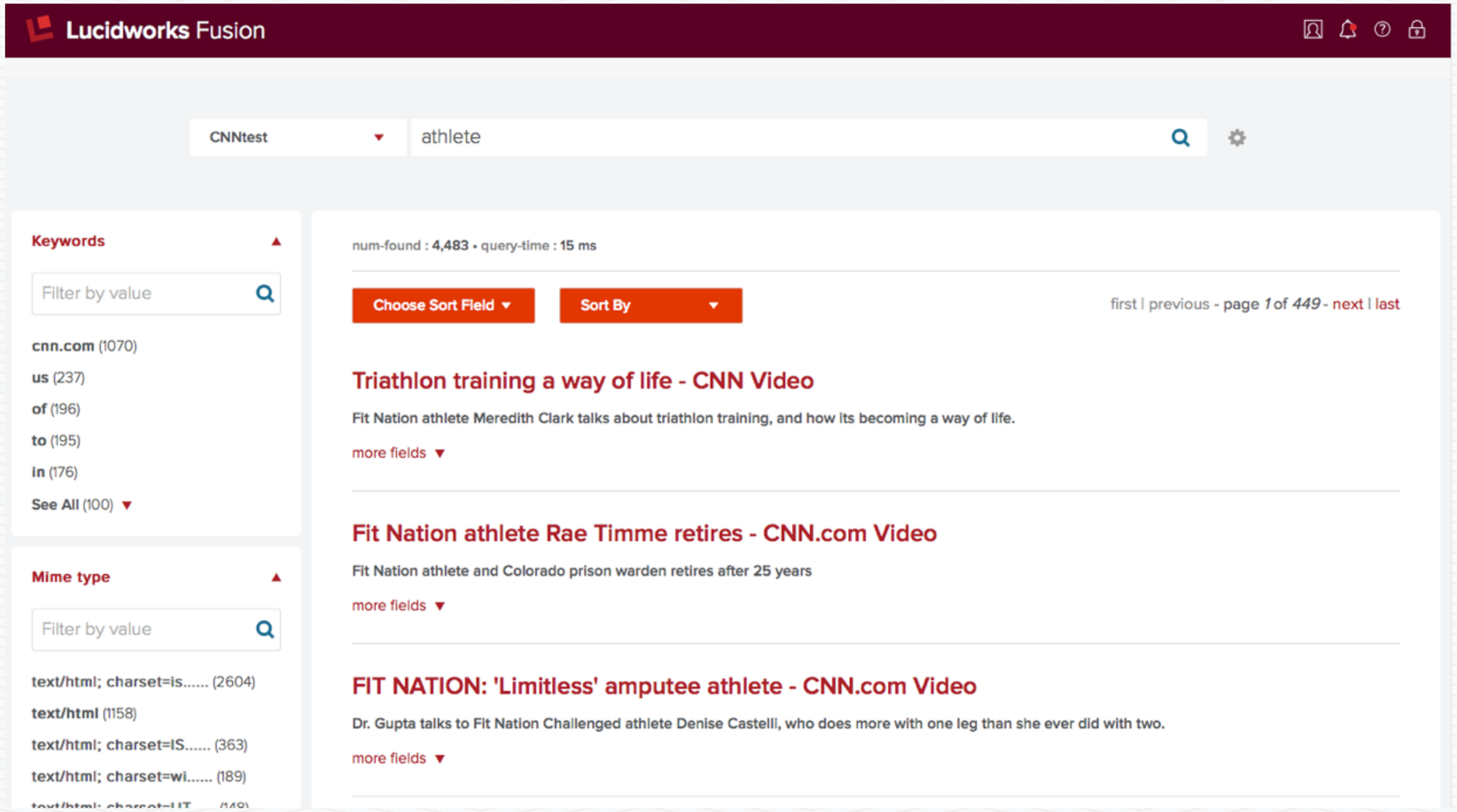
Fusion uses Javascript for running arbitrary scripts. Javascript index stage allows you to run JavaScript functions on your content. When indexing, this may allow you to add or remove content that can't be added with any other available option. Among other things, developers have used this to dedupe, remove disclaimers from emails, conditionally process documents based on datasource, and so on.

You can leverage Java Libs. You can also compile your own generic logic in Java and make them available to Fusion. This provides great programming flexibility.

```
function (doc) {

  // Add a new field
  doc.addField("MyPassion", "Mountaineering");

  // Get a field value.
  var value = doc.getFirstFieldValue("MyPassion");

  // Change a field value:
  doc.setField("MyPassion", "Ice Climbing");

  doc.addField("OldPassion", value);

  // Remove a particular field:
  // If there are multiple instances of a particular field, it will remove all instances.
  doc.removeFields("MyPassion");

  // Another valid method to get a field value.
  var value1 = doc.getFieldValues("Hobby").get(0);

  // There is also an object that persists across several documents called _context
  var count = _context.getProperty("TotalDocCount");

  // You must return the document!
  return doc;
}
```

Fusion’s In-built Search UI

Accessible from the Fusion Launchpad



Demo and Lab 2

Demo and Lab 2

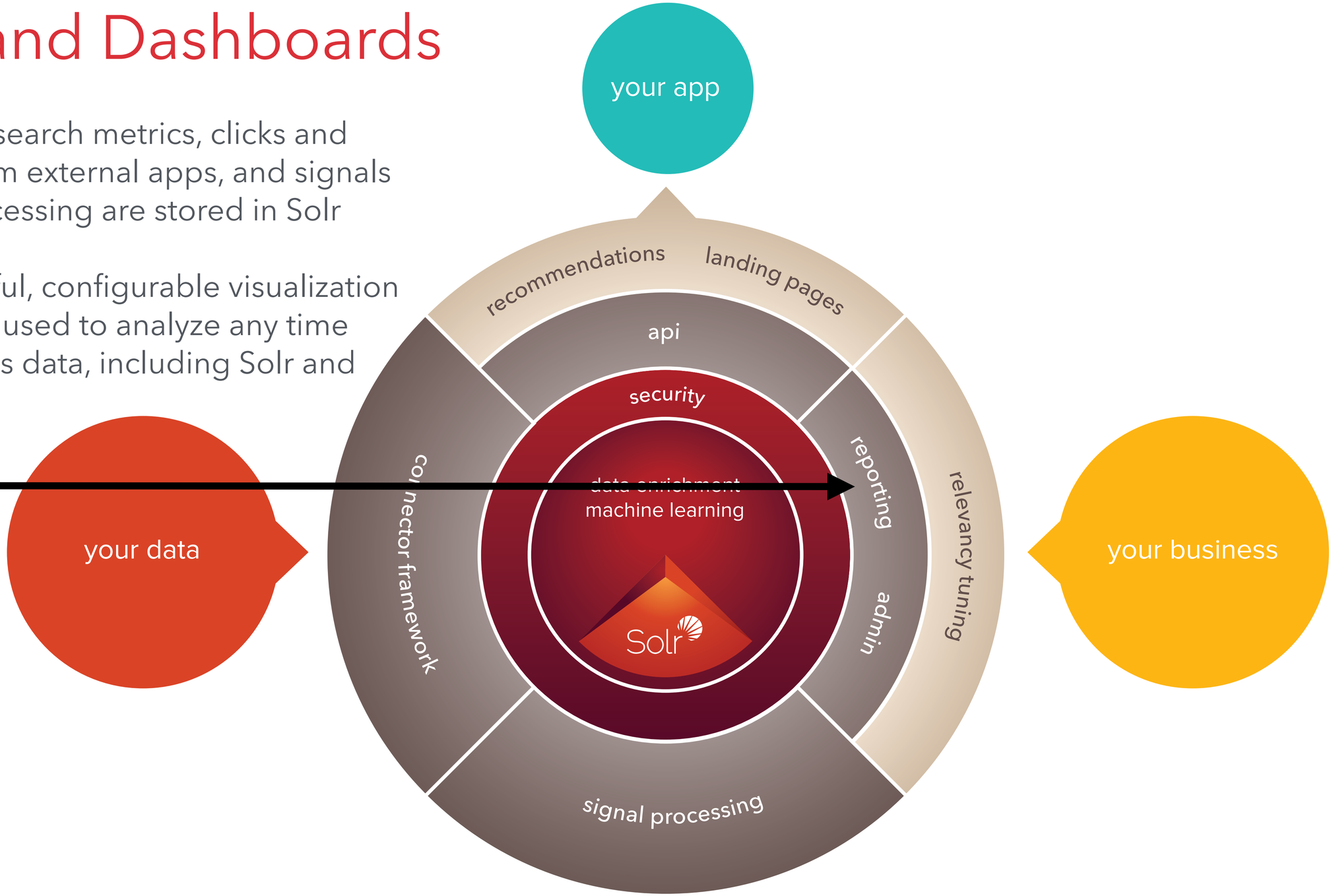
- Demo key out-of-the-box index pipeline stages and the transformation of documents in a pipeline
- Lab: Create a new collection and follow the steps in http://lucidworks.com/blog/noob-notes-fusion-first-look/ to index the Medline dataset. Use the Fusion Search UI to explore your results.
- Challenge lab (optional) If on AWS, or if there is good network to connect to a DB on AWS, also connect to and index from a database.

Monitoring, Log Analytics and Dashboards

Reporting and Dashboards

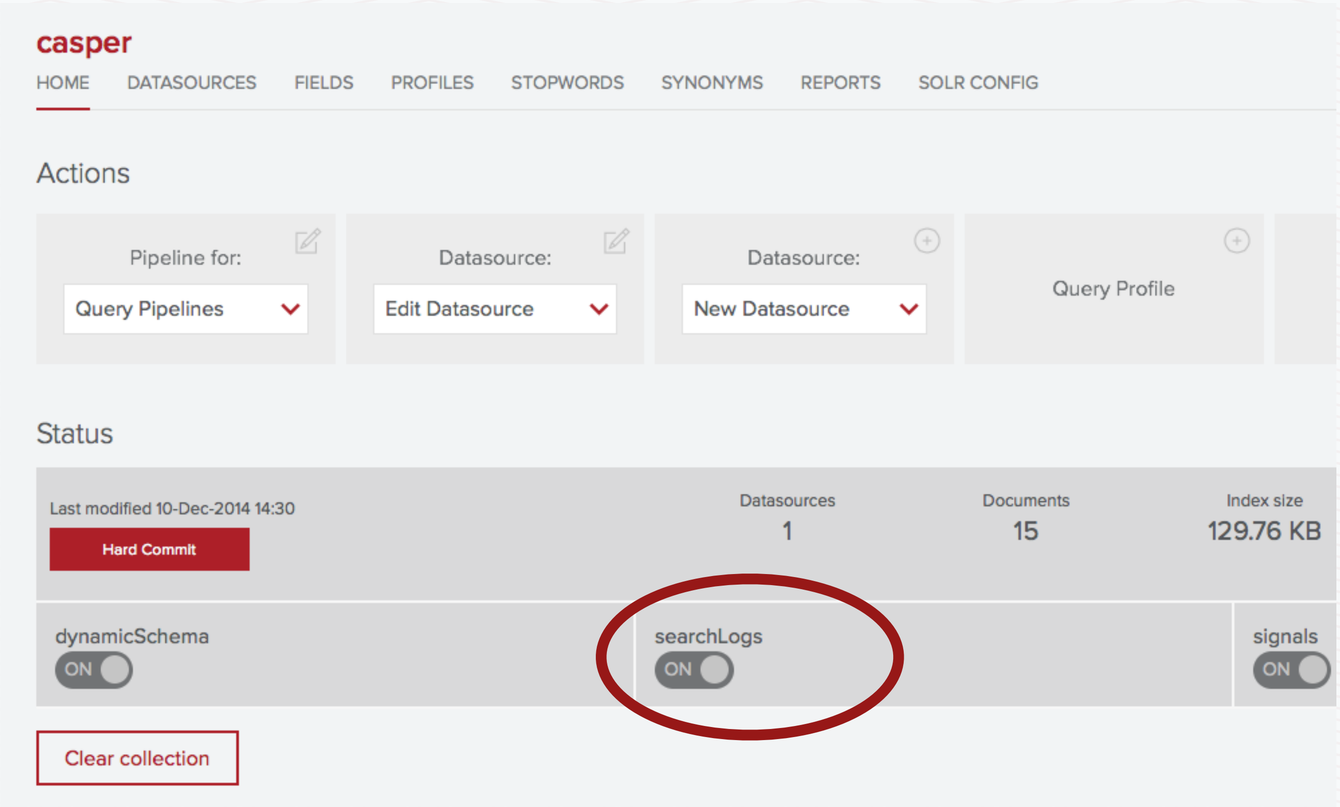
All system metrics and search metrics, clicks and other similar events from external apps, and signals extracted by event processing are stored in Solr

Fusion provides powerful, configurable visualization capabilities that can be used to analyze any time series or non-time series data, including Solr and Fusion logs



The “_logs” collections

- When creating a collection, or editing its parameters, we can turn on search logs. If turned on, search logs are indexed to a collection “<collection_name>_logs”
- Fusion/Solr system logs are stored in a system collection named “logs”



Content of <collection_name>_logs collection

- Contains key parameters of user searches on <collection_name>, such as query term, time taken to execute query, number of hits, etc. By analyzing this, search admins and content creators can understand whether they are providing a responsive interface that is serving relevant results.

View: Table / JSON / Raw

Field	Action	Value
version		1491477278245257200
collection_s		casper
id		8c98d849-87c7-49d9-b570-a1f33e2a27c9
numdocs_l		15
q_s		*
q_txt		*
qtime_l		49
req_defType_ss		edismax
req_facet_ss		true
req_hl.fl_ss		*
req_hl.simple.post_ss		
req_hl.simple.pre_ss		
req_hl.snippets_ss		1
req_hl_ss		true
req_json.nl_ss		arrarr
req_lw.pipelineId_ss		casper-default
req_q_ss		*
req_rows_ss		10
req_sort_ss		score desc
req_start_ss		0
req_wt_ss		json
timestamp_dt		2015-01-27T18:32:53.766Z
totaltime_l		62

Content of system “logs” collection

- Contains details of all system events on Fusion

View: **Table** / JSON / Raw

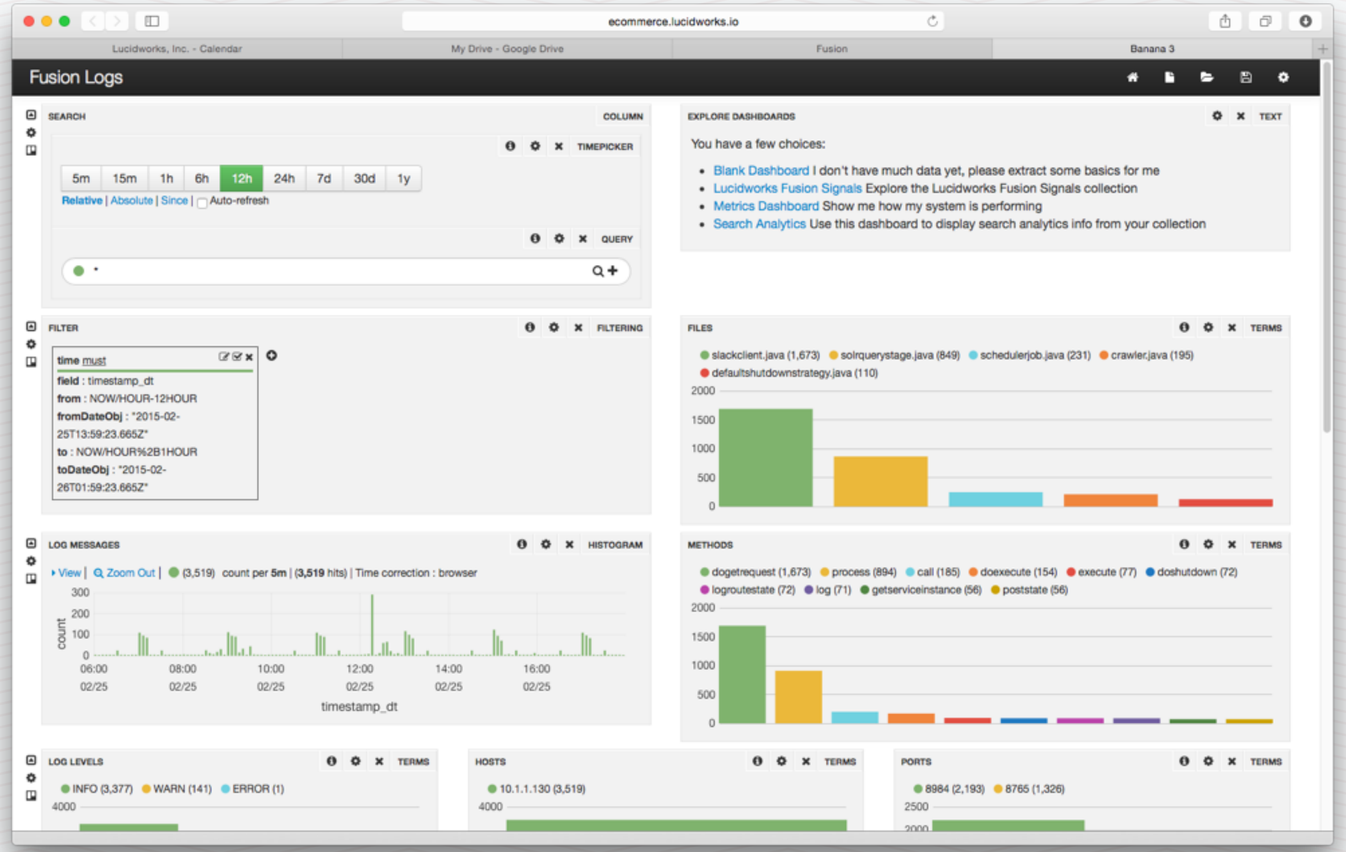
Field	Action	Value
version		1494134656076873700
class_t		com.lucidworks.apollo.pipeline.query.stages.SolrQueryStage
file_t		SolrQueryStage.java
host_s		10.1.1.130
id		0a9d85b6-323e-43fe-bc0d-839005d65fec
level_s		INFO
line_i		272
message_t		Logging search event for collection 'demo'
method_t		process
port_s		8765
thread_t		qtp1981488825-18
timestamp_dt		2015-02-26T02:30:46.711Z

The Reporting API

- Fusion API provides many interesting reports on the searches performed against a collection
- If searchLogs is enabled for a collection, the following reports are available through the reporting API
  - Get a List of Available Reports
  - Find Queries with Less Than 'N' Results
  - Get a List of the Top Queries
  - Get a List of Most Popular Terms
  - Get a List of Most Clicked Documents
  - Get a Histogram of Query Times
  - Get a Date Histogram
  - EXAMPLE: `curl -u user:pass -X POST -H 'Content-type: application/json' -d '{"n":1}' http://localhost:8764/api/apollo/reports/demo/lessThanN` gives us all queries against the collection “demo” that returned less than 1 (i.e. zero) results

Fusion Dashboards

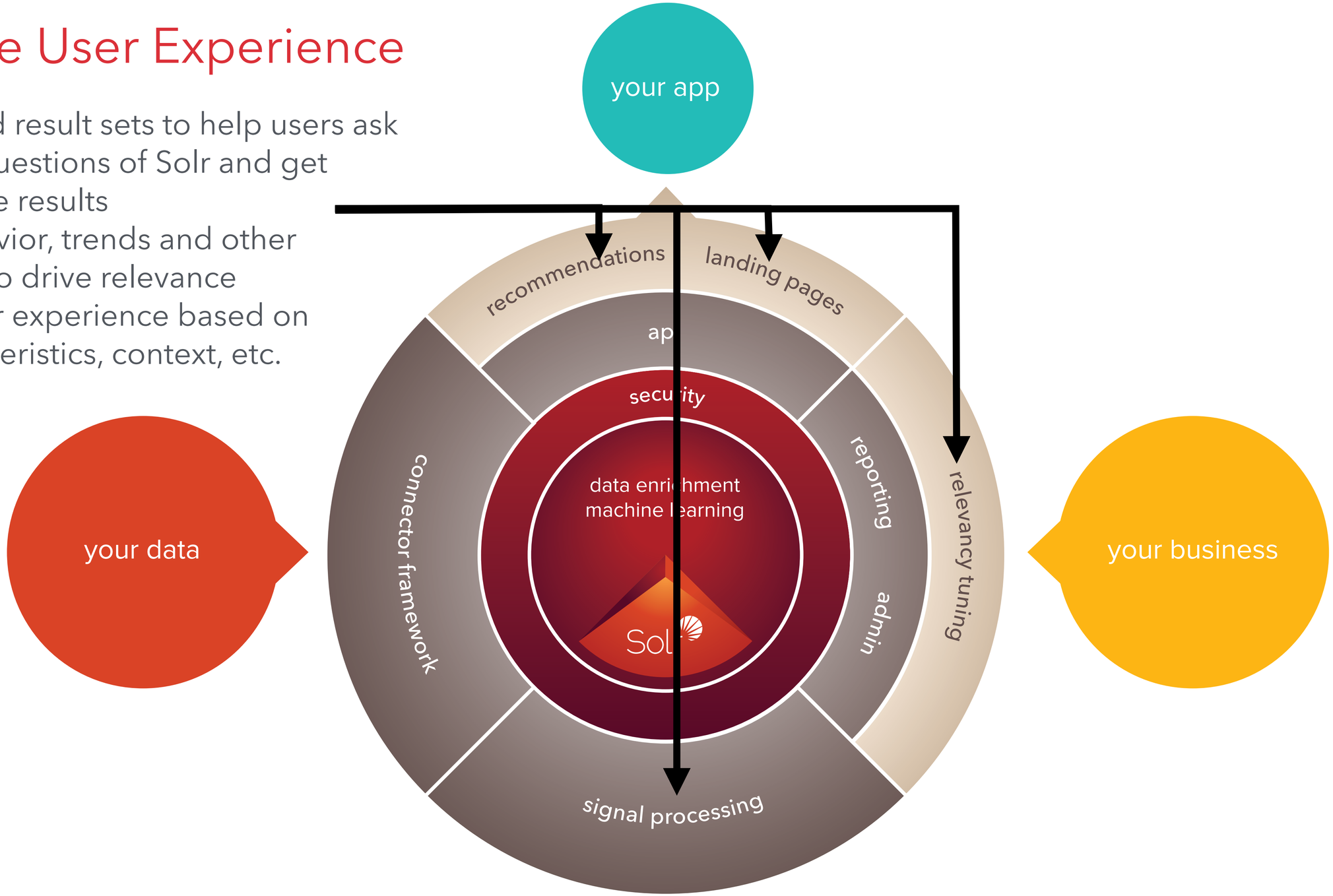
- Integrates the popular open source visualization tool for Solr, Banana (which in turn is a fork of Kibana)
- Dashboards layouts are JSON objects that are stored in Solr
- Visualize the "_logs" collections, as well any other time series or non time series data that you choose to load into Solr



How do I Tailor Search Results?

Fusion and the User Experience

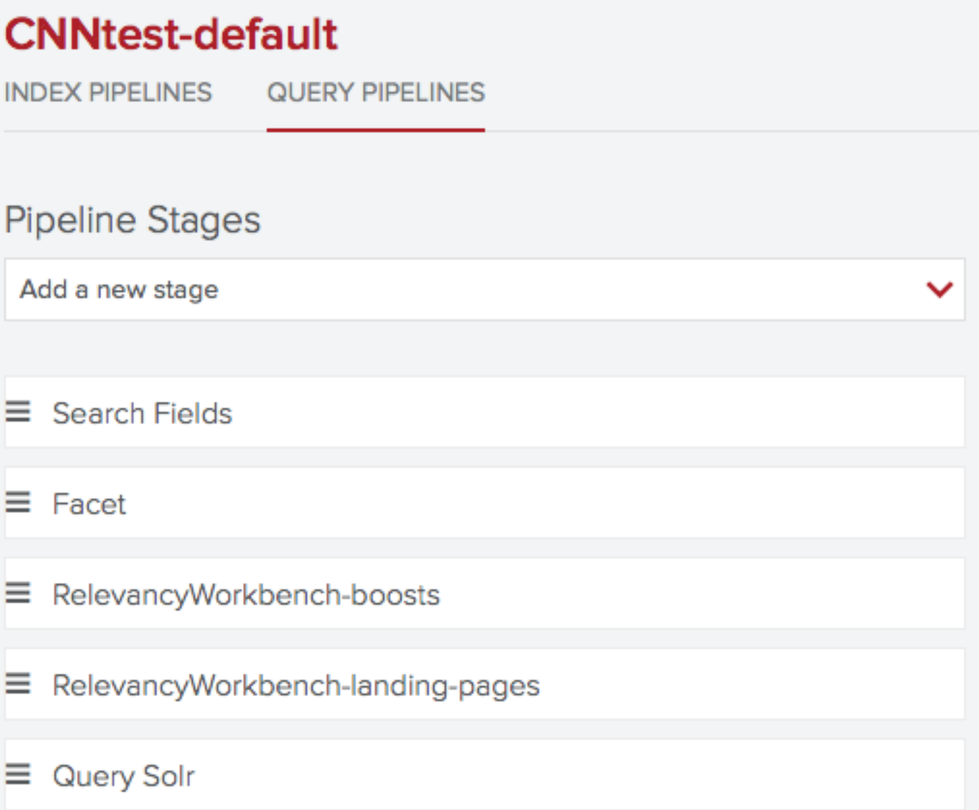
- Modify queries and result sets to help users ask more interesting questions of Solr and get relevant, actionable results
- Capture user behavior, trends and other events, and use it to drive relevance
- Customize the user experience based on query, user characteristics, context, etc.



Query Pipelines and Relevancy Workbench

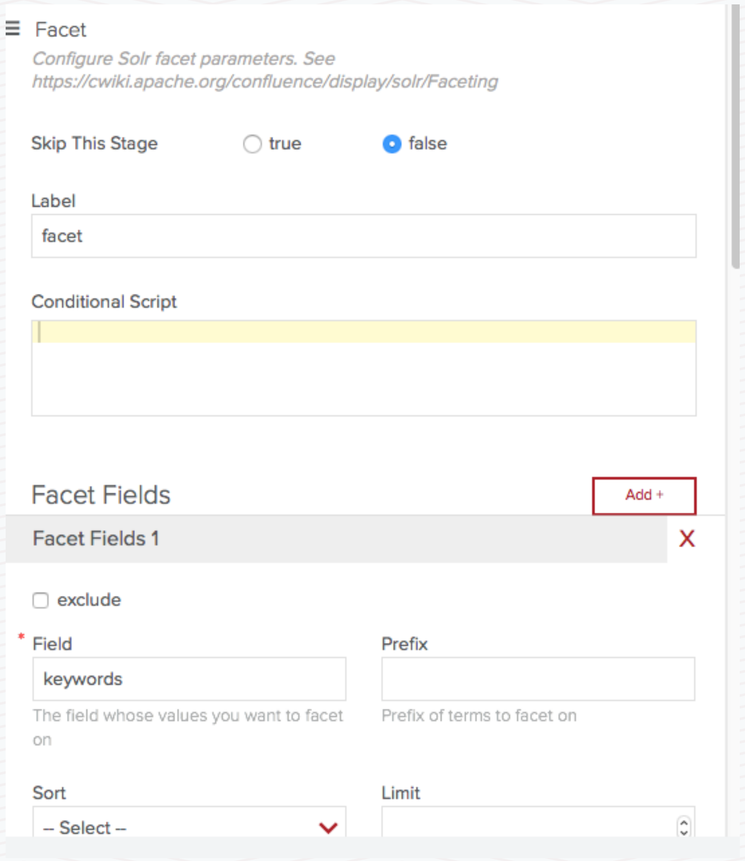
Modifying User Queries and Result Sets

- Query pipelines are similar in concept to Index pipelines; the former transform queries and query results during searching, while the latter modify documents while indexing
- Advantages of maintaining query and result modifications within Fusion query pipelines
  - Scalability, Distributed Deployment and Load Balancing as part of the Fusion Backend API
  - A/B and Multivariate testing—required while tuning/evolving your search application—made easy
  - Ease of maintenance (your front-end app need not change when your query transformation logic changes)



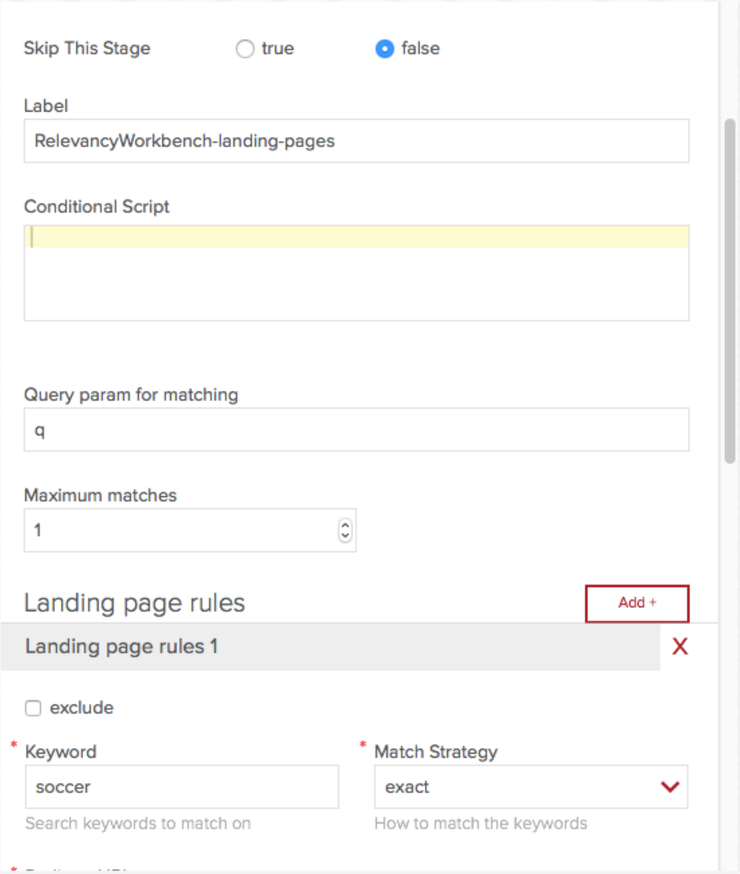
Out-of-the-box Query Pipeline Stages

- **Set Query Params** provides a generic way to specify any Solr query parameter.
- **Facet** stage is used to define a facet.
- **Recommendations Boosting**, **Boost Documents**, and **Block Documents** stage types provide document boosting and blocking capabilities.
  - Recommendations boosting is based on aggregated signals (more on this later), while other two allow defining document boosts/blocks based on the search terms entered.
- **SubQuery Stage**: Solr query to another collection. Returned results can be used to join results or boost main results.
- **Rollup Aggregator:** Rollup stage to aggregate Solr results in the format of List<DocumentResult>. Most commonly used for advanced boosting based on signals, which is performed by the **Advanced Boosting Stage**.



More Query Pipeline Stages

- **Landing Pages:** customize landing pages based on search term. Does not do a redirect, just supplies a URL to the calling application.
- **Logging stage** writes query parameters to the log.
- **Javascript stage**: general transformations. Examples include best bets, forcing exact matches, combining boosts in interesting ways, etc.
- **Security Trimming:** adds capability to apply security restrictions found by crawls to queries as they are being processed.

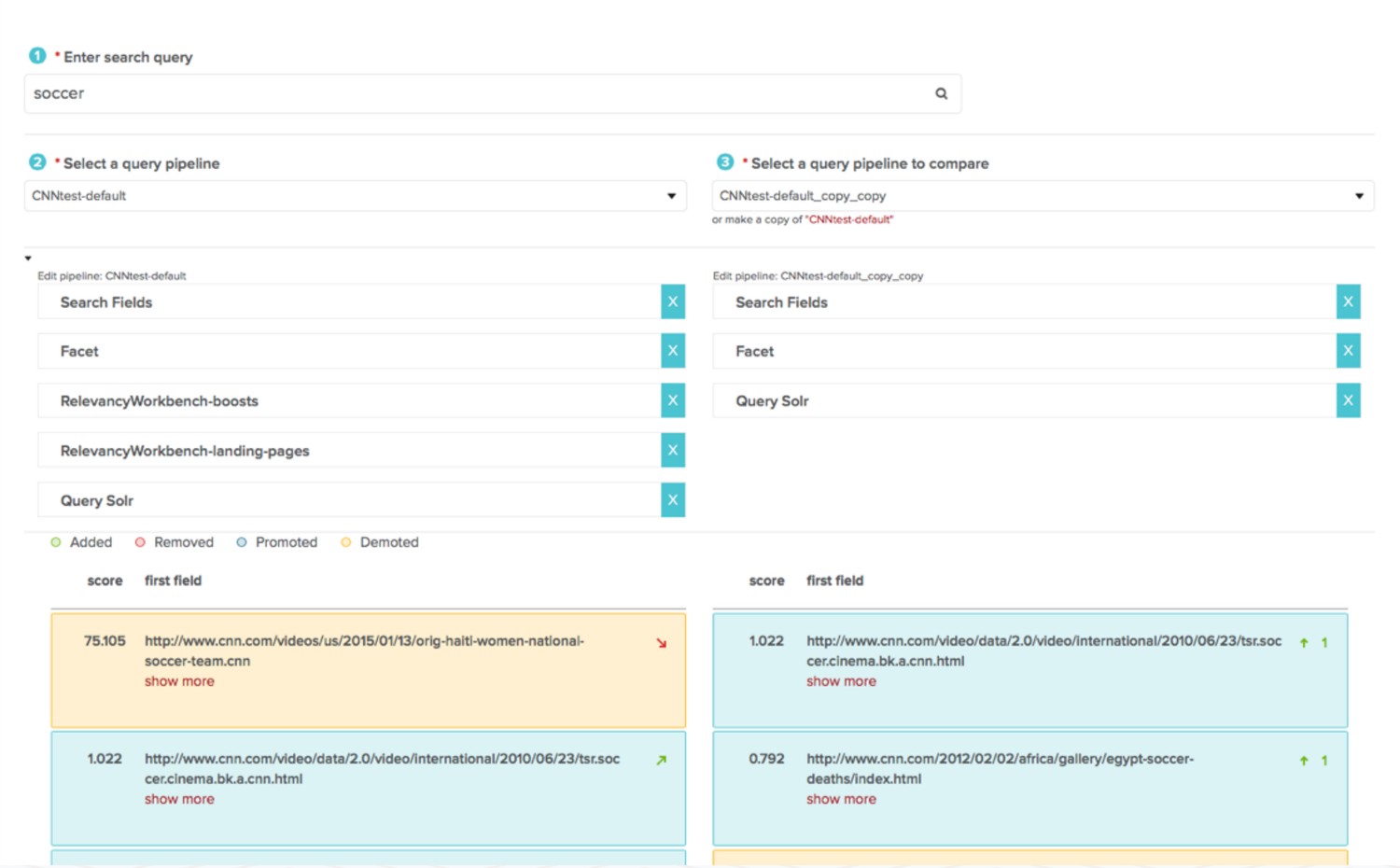


Query and Index Profiles

- In Fusion, query and index pipelines are not connected to a specific collection by default.
  - Provides a great degree of flexibility —pipeline can be created once and re-used in several collections.
  - However it does add some complexity in terms of using a pipeline with a collection.
- Sometimes, for example while using a SolrJ-based push connector (using SolrJ), we need to explicitly tie a pipeline to a collection.
  - Fusion supports a concept called profiles that provide a many-to-many mapping between collections and pipelines.
  - Profiles serve as aliases to a pipeline. Your apps can send docs to one alias and you can change the pipeline and collection that the alias is associated with. That way your front-end apps need not change, you can modify search behavior my modifying the pipeline associated with a profile.
  - Example: `curl -u user:pass -X POST -H "Content-Type: application/vnd.lucidworks-document" -d '[{"id": "myDoc1","fields": [{"name":"title", "value":"My first document"},{"name":"body", "value": "This is a simple document."}]},{"id": "myDoc2","fields": [{"name":"title","value": "My second document"},{"name":"body", "value": "This is another simple document."}]}]' http://localhost:8764/api/apollo/collections/docs/index-profiles/testProfile/index`
    - Sends documents to a profile named testProfile.

Relevancy Workbench

- Tune your search results by comparing query pipelines and editing them as necessary



Demo and Lab 3

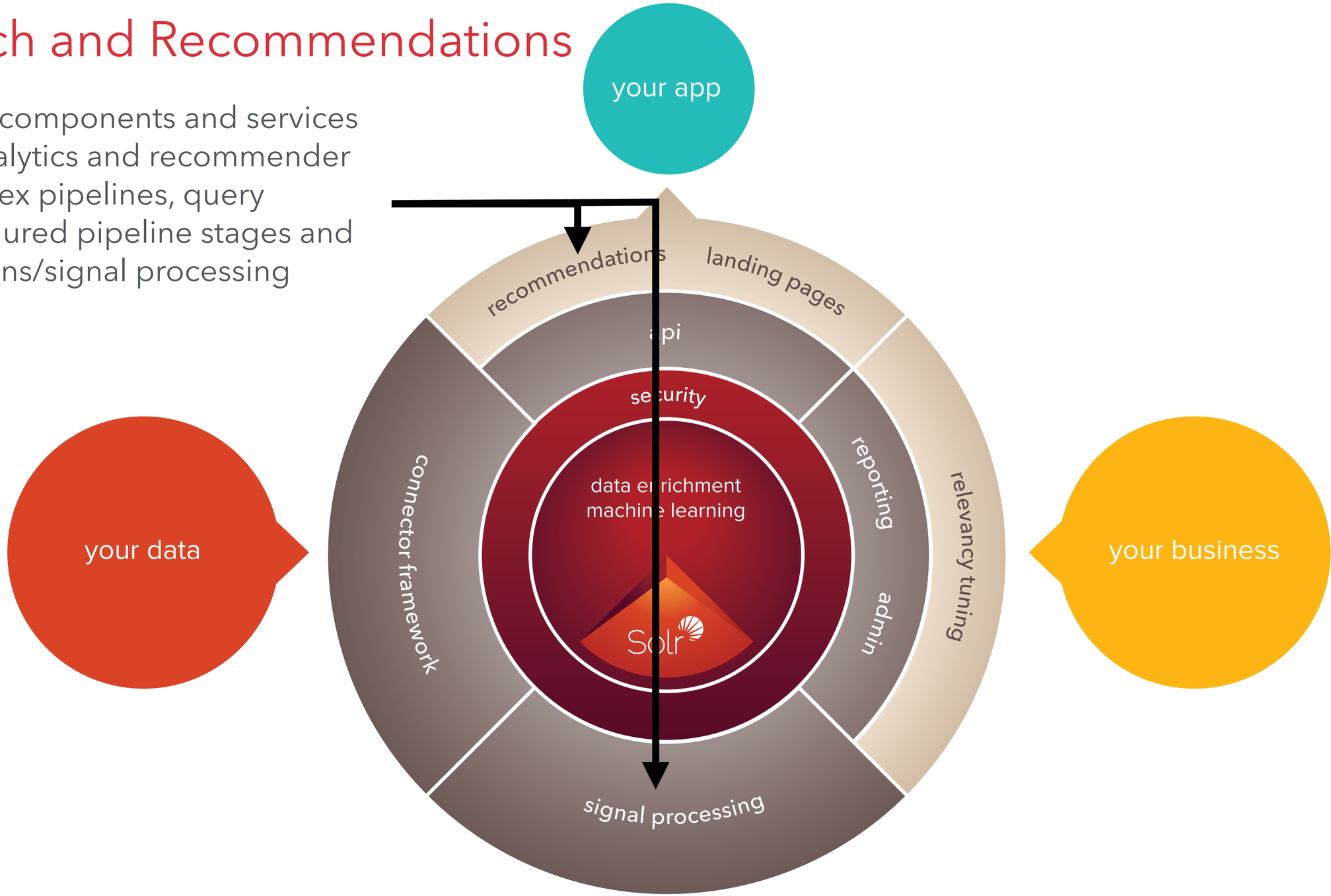
Demo and Lab 3

- Demo
  - Showcase Fusion dashboards, query pipelines and relevancy workbench
- Hands-on Lab (can be combined with Lab 4)
  - Import a product catalog (from “csv” or XML)
  - Use Fusion UI and show simple the effects of configuring query pipeline stages (Facet, Set Query Params)
  - Use Global sources as an example of query pipelines with Javascript stages
  - Use relevancy workbench with two query pipelines defined above to compare results
  - Set up a signals collection, create an index profile that points to it and index signals by pushing to that profile (using logstash say. Fusion 1.3 should have a logstash connector)
  - View it with a dashboard provided
- Hand-out Feedback Forms

How do I drive more powerful user experiences?

Fusion of Search and Recommendations

- Fusion provides key components and services required to build analytics and recommender systems—such as index pipelines, query pipelines, pre-configured pipeline stages and powerful aggregations/signal processing capabilities

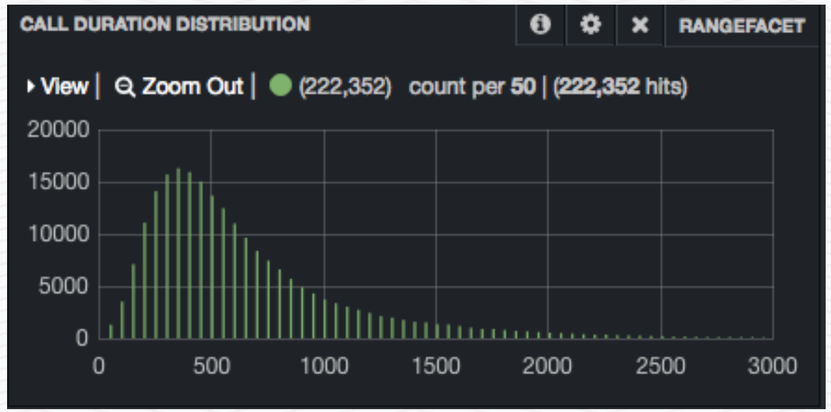


Events Processing and Signals Extraction

Terminology

- **Event:** A data point or measurement with an associated timestamp (and location)
  - Examples: User query, click, add-to-cart, buy, CDR (call data record), sensor data for a given moment in time (eg. temperature reading at 0800:00:00UTC in SFO), etc.
- **Time Series:** A sequence of events (data points), with a natural temporal ordering. Observations close together in time will be more closely related than observations further apart
  - A set of query or click log records from a search engine e.g. a "clickstream," a set of medical claims with claim start dates, a set of CDR data, etc.
- **Streams:** On-going time series with no defined end-point or date
- **Signal:** a function that conveys information about the the behavior or attributes of some system or phenomenon
  - For example, a rising qps (queries-per-second) and a corresponding rise in query response times may indicate the need for more hardware; increased call durations may indicate the need to add cell-tower capacity; the number of searches for the word "flu" or "influenza" in a region might indicate an increased incidence of influenza in that region

query_s	doc_id_s	type_s	timestamp_dt
macbook pro	9637258	click	2011-09-07T15:46:14.108Z
or tax	2462512	click	2011-08-21T14:27:10.327Z
norton	1520158	click	2011-10-07T00:13:41.43Z
hello kitty	1256419	click	2011-10-26T00:20:25.44Z
wireless router	2225056	click	2011-10-12T21:56:34.244Z
samsung 8000	2128142	click	2011-10-30T14:41:17.978Z
earbud straight	8964864	click	2011-08-21T19:32:55.225Z

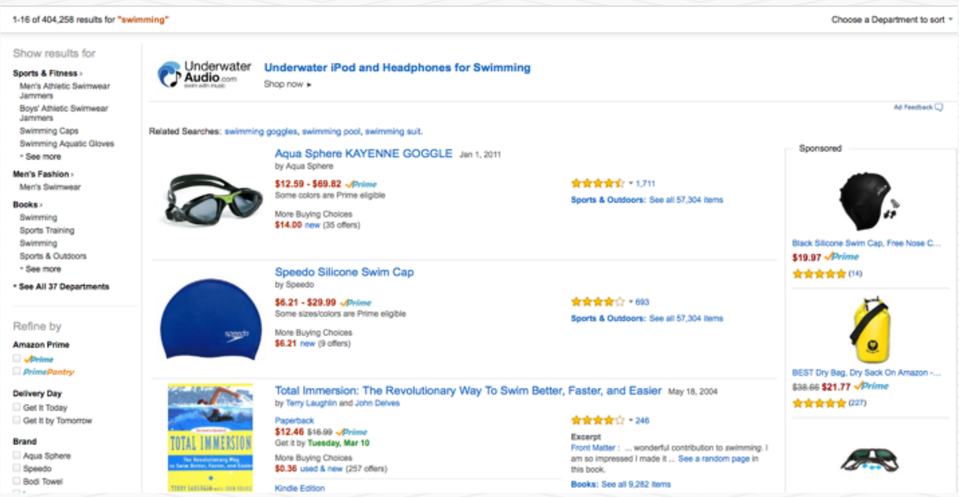


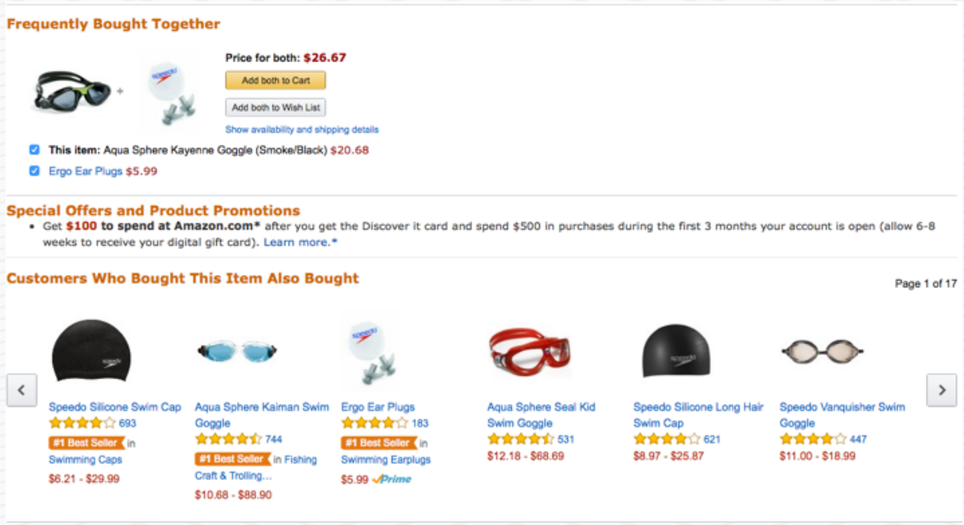
Evolving Concepts

- In Fusion, we have tended to use events and signals interchangeably in the product and documentation
- An event is almost always a signal in the sense that it conveys information about some system or phenomenon
  - Some events are highly significant in themselves (for example a syslog record saying memory utilization is 100%, or a firewall log record indicating a breach)
- However, many signals are computed by analyzing a collection of events
  - Aggregation and analysis of events and event streams typically extracts signals that contain more information than the individual events themselves
  - Example: it is interesting to know that one user searched for “tablet” and clicked on the new iPad; it is far more interesting and actionable if we learnt that 80% of the users who searched for “tablet” clicked on the new iPad). The latter is “actionable” in that we could promote (boost relevancy) of the iPad to all users who searched for “tablet”

Key Fusion/Solr Components for Driving Powerful User Experiences and Presenting Actionable Information

- **Index pipelines** are used to process event streams at ingestion time
- **Solr** stores large quantities of events and signals, and provides a number of on-the-fly analysis and aggregation capabilities (facets, stats, pivot facets, stats on pivots, etc.)
- **Fusion API** extends Solr's analysis capabilities through its aggregations API; used to process large sets of events, extract signals and store these signals in Solr
- **Query pipelines** leverage raw and aggregated data in Solr to tune relevance and the user experience
- Overall, Fusion enables you to **ask more interesting questions of your data and receive timely, predictive and actionable information**





Aggregations

- Events and signals may need to be aggregated in order to be used for analysis, recommendations, etc.
- Aggregator Functions: arithmetic, string, collection, statistical, logical, scripting and special functions
  - Sum, sumOfSquares, mean, min, max, count, decay_sum, etc.
  - Cat, split, replace, etc.
  - Collect, discard, etc.
  - Variance, stddev, cardinality, skewness, kurtosis, quantiles, topK, covariance, correlation, sigmoid, etc.
  - Modify and define aggregation functions using Javascript

Field	Action	Value
version		1477627505717280800
aggr_id_s		db081bf4d3e7483b9cb824367b3f8e7d
aggr_type_s		click@doc_id_s-query_s-filters_s
attr_params.docId_		1232447
attr_params.filterQueries_		cat00000,abcat020
attr_params.indicator_s_		Private,Pub
attr_params.query_		beats,B
attr_params.query_time_dt_		Fri Oct 2
attr_params.userId_		a90c2ae060cc
attr_query_orig_s_		beats,Beats,beats.,beaTs,B
count_d		366
count_i		366
doc_id_s		1232447
expr_t		beats & abcat0200000 $ abcat0204000 $ cat00000 $ pcmcat14 pcmcat144700050004 | beats & abcat0200000 $ abcat020400
filters_orig_ss		abcat0204000,pcmcat144700050004,abcat0200000,cat00000
filters_s		abcat0200000 $ abcat0204000 $ cat00000 $ pcmcat14470005
flag_s		aggr
id		db081bf4d3e7483b9cb824367b3f8e7d-54402
ids_ss		0024b8a8-10af-4907-869f-3228717
params.indicator_s		Private
params.position_s		0
params.query_time_dt		2011-10-09T15:45:30.956Z
query_orig_s		beats
query_s		beats
query_t		beats
script_d		4
script_sum_logs_d		253.69186808494052
timestamp_dt		2011-10-26T22:58:39Z
type_s		click
weight_d		22.049415588378906

366 users clicked on this particular document (docId=1232447) after searching for “beats”

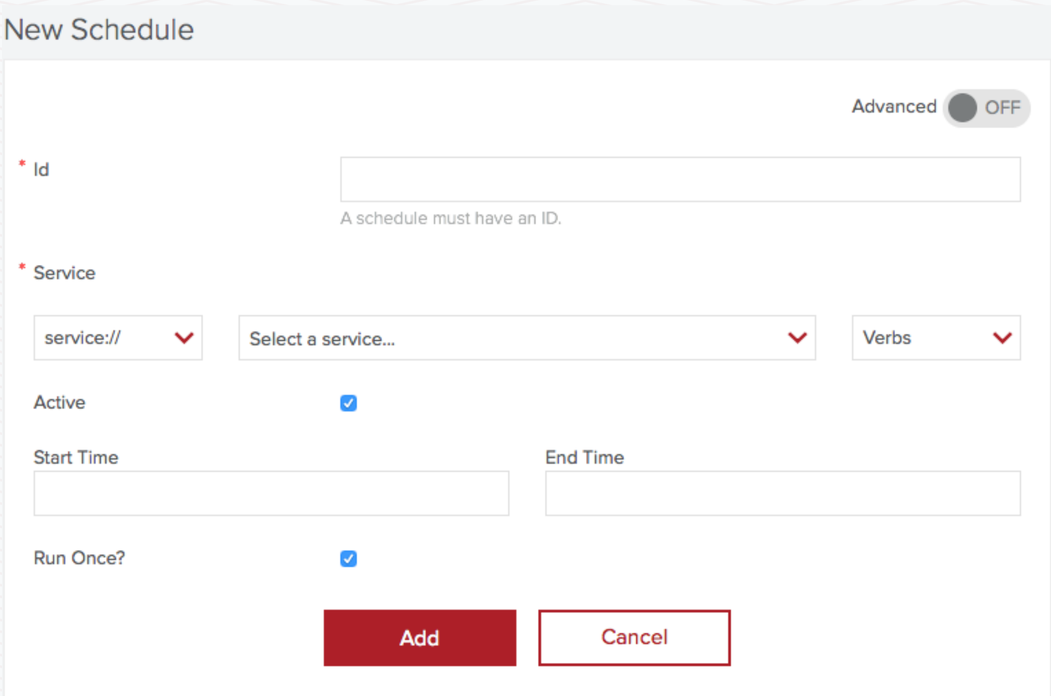
Search term

Weighted

sum of clicks with time decay

Scheduler

- Scheduler API/service allow you to execute any Fusion service, any Solr request, or any other HTTP request on a defined timetable
- Scheduler service does not in itself execute any business logic
  - Defines start time and repeat interval, and an address to an endpoint that will perform the requested actions
- Examples:
  - Run a Solr query at a specified time every day
  - Define a datasource to be re-crawled once a week
  - Define a periodic aggregation of clickstream events



How do I build Recommender Systems?

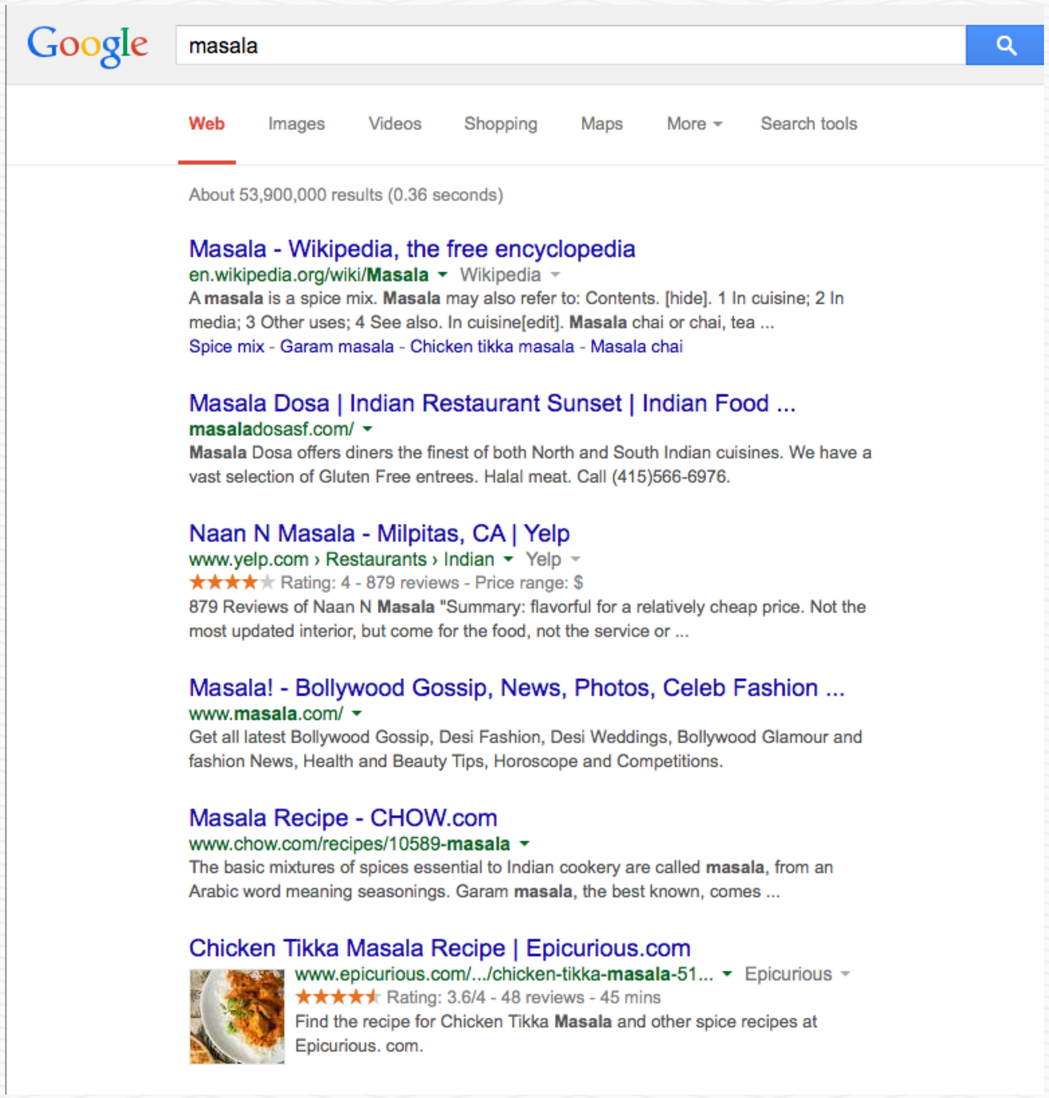
What is a Recommender System?

- Textbook definition: “Software tools and techniques providing users with suggestions for items a user may wish to utilize.”

~Ricci et al., Recommender Systems Handbook. Springer, 2011.

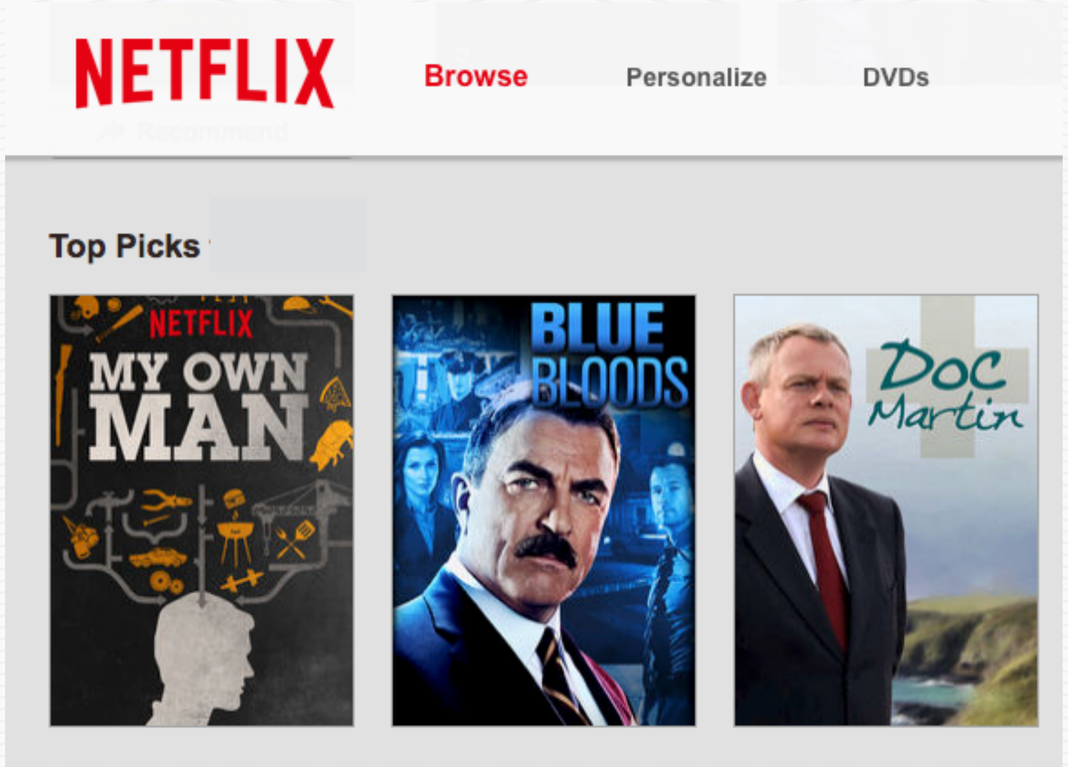
Search is a Recommendation Problem

- Does not give you a randomly ordered set of results that matched your query; scores results and attempts to first return items that are more likely to be relevant/useful
- Not just “what matches user query,” but “what is most likely the thing the user wanted”



Recommendation is a Search Problem

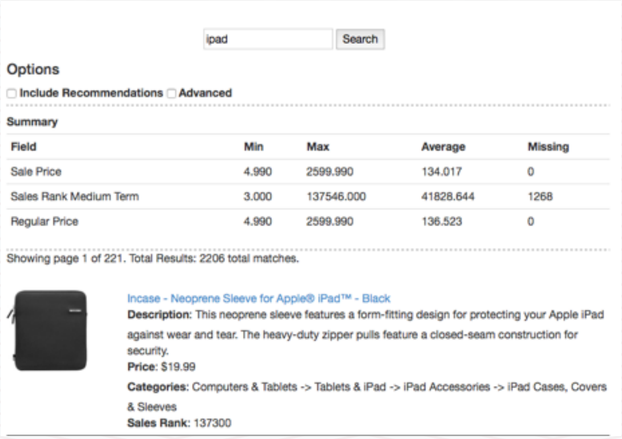
- Recommendation systems generally query an index of possible items in order to find those items that are a best match
- Usually involves storing a large sparse matrix and retrieving quickly
- Search engine plus associated processing provides a powerful, scalable, performant recommender system



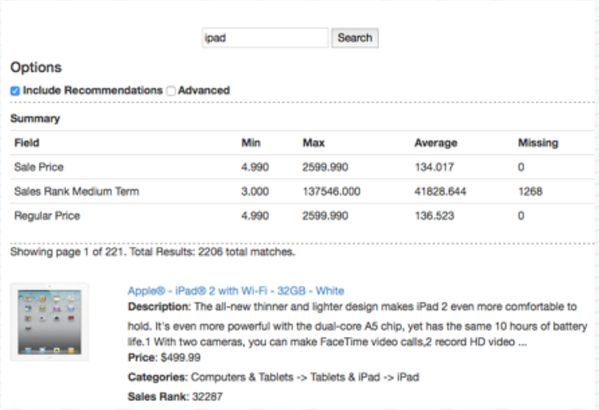
Fusion of Search and Recommendations

- In the traditional view, search is generally “explicit” (i.e. requires user input) while recommendations are usually “implicit” (automatically derive or assume some user intent)

- Fusion provides the tools to flexibly combine recommendations and search

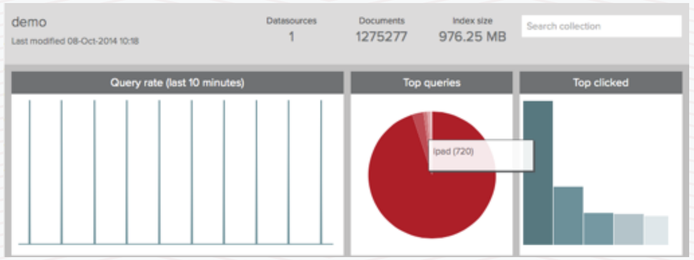


- Without recommendations, top ranked result for the search “ipad” is an iPad case, because the term appears in the title and frequently in the description. However, when we use **click boosting**, the most clicked on item, a white iPad, rises to the top.



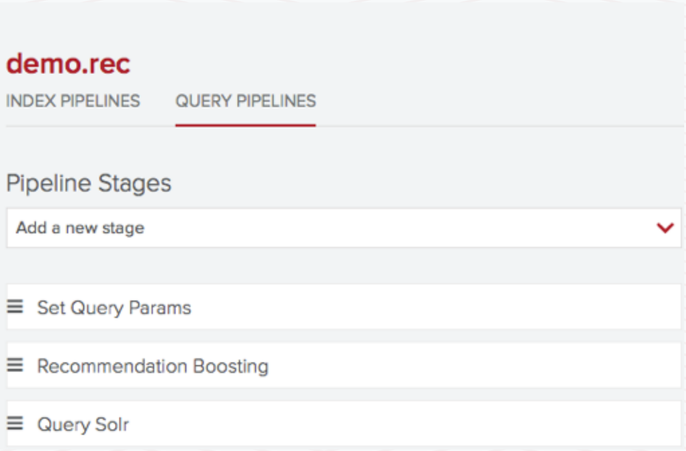
Click Boosting in Fusion

- Click boosting ("users who searched for this item tended to click on ...) achieved as follows:
  - Index incoming stream of click events (sent by the web application) to a "signals" collection (typically named "<collection_name>_signals"
  - Periodically aggregate "signals" collection on docId's and queries and store in aggregation collection. The aggregation function could use a weighted sum, with the weights calculated from a half-life parameter that models the time-decay in the importance of a click (more recent clicks are weighted more than older clicks)
  - Build a query pipeline that looks up the aggregated collection to get most frequently clicked items for the query, and uses this to add boosts to the raw query
  - Associate the query pipeline to a collection and direct all application searches to the endpoint representing that query profile



Collection containing product catalog

Associated events/ signals and aggregation collections



Query Pipeline with Click Boosting

Types of Recommendations

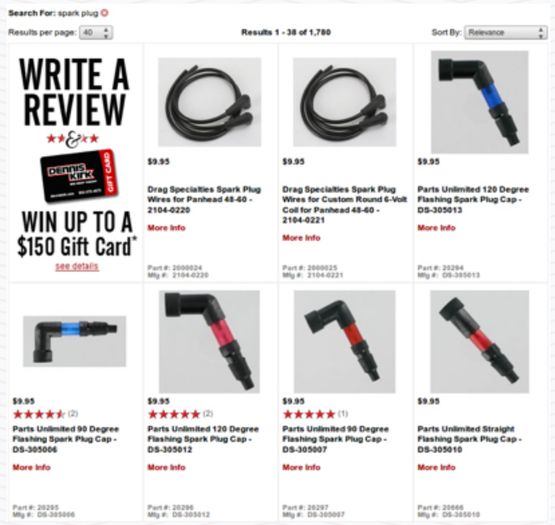
- Non-personalized: same for everyone
  - Editor's picks; most popular; trending now (simple but often very effective)
- Contextual: based on what the user is doing **right now,** but not looking at past behavior
  - "Users who viewed this item also viewed..." (or “ user who bought this item also bought...”, etc.); click boosting; similar searches ("users who searched for this also searched for...")
  - Sometimes called “semi-personalized” or “ephemeral” recommendations in the literature
- Personalized: uses the current user's history to generate recommendations
  - "Recommended for you”; "based on your shopping history”

Implementing Recommendations in Fusion

- Non-personalized
  - Boost stage
  - Aggregate on doc ID, then query the aggregation collection directly
- Contextual
  - Click boosting (subquery + rollup + boost)
  - Aggregate on doc ID and <context>, then query the aggregation collection
  - Click boosting with context
- Personalized
  - Aggregate on user ID and content attributes, then query (or boost using) the aggregation collection
  - Simple collaborative filter (with more coming soon!)

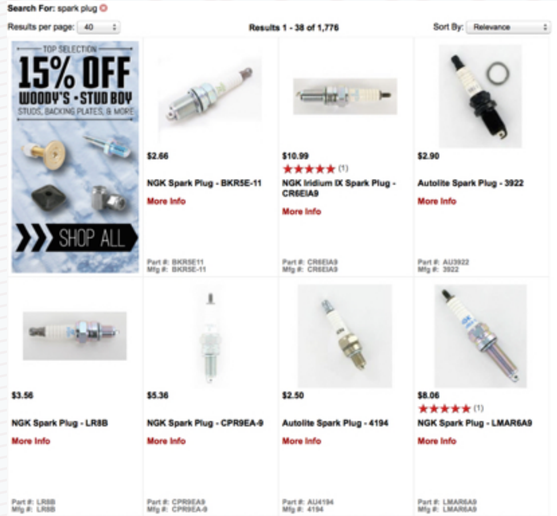
Implementation Considerations

- Your need to track events (clicks, buys, add-to-cart, up-vote, rating, document views, etc.) and send them to Fusion
  - Need sufficient information to implement a particular recommender
  - For contextual recommendations, need to capture and send context
  - For personalized recommendations, you need to reliably track users and send user information
- More expensive at query-time
  - You may need to make multiple queries and some calculations in the query pipeline (true of most recommender systems)
- Choose the Aggregation Intervals Wisely
  - Expensive operation; match the scheduled aggregation interval to the time it takes for user patterns to change
  - “Differential” calculations available for certain types of aggregations
- Cold Start
  - If the site has been in existence for some time, we have successfully used existing clickstream logs, even if they are old



Without recommendations, a search for “spark plug” brings up spark plug wires and accessories

With recommendations, a search for “spark plug” brings up spark plugs, which are items that users actually clicked on after their search



Demo and Lab 4

Demo and Lab 4

- Build a simple recommender using click boosting. Need to use aggregator and query pipelines. Show results in relevancy workbench using two query pipelines, one default and one with the recommender. Also view in the Search UI
- Homework: Build a recommender that uses context (such as user device, age, gender) to customize results

Summary, Resources and Feedback

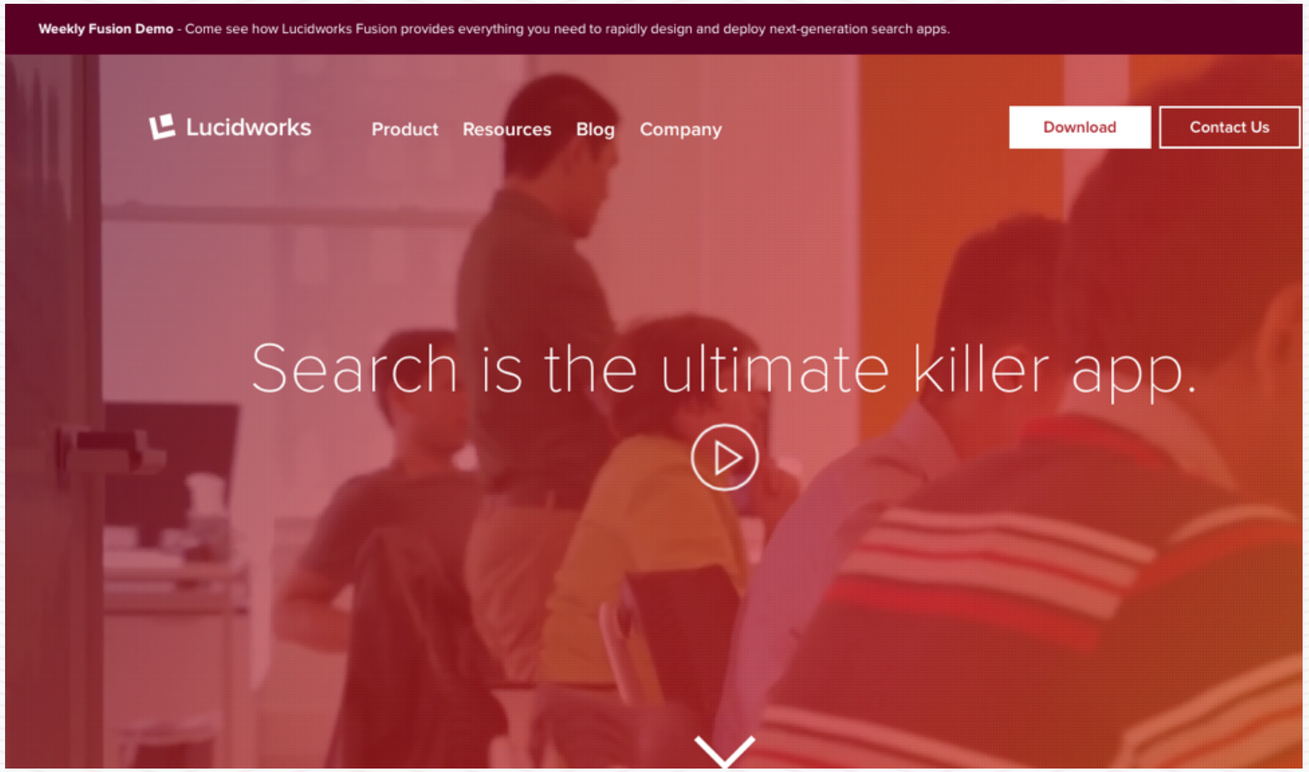
Training Summary

- Introductions
- Why Fusion; Training Goals
- Not your Father's Solr
- Fusion and Solr Deployment
- Getting Started; Navigation Basics
- Fusion and Solr APIs
- How do I get data into Solr?
- Monitoring, Log Analytics and Dashboards
- How do I tailor my Search Results?
- How do I drive more powerful User Experiences?
- Summary, Resources, Feedback

Your Feedback is Important to Us

Resources

- Solr: http://lucene.apache.org/solr
- Company: http://www.lucidworks.com
  - Blog: http://www.lucidworks.com/blog
- Fusion: http://www.lucidworks.com/products/fusion
  - Help: https://docs.lucidworks.com/display/fusion/Lucidworks+Fusion+Documentation



Acknowledgements

Contributors

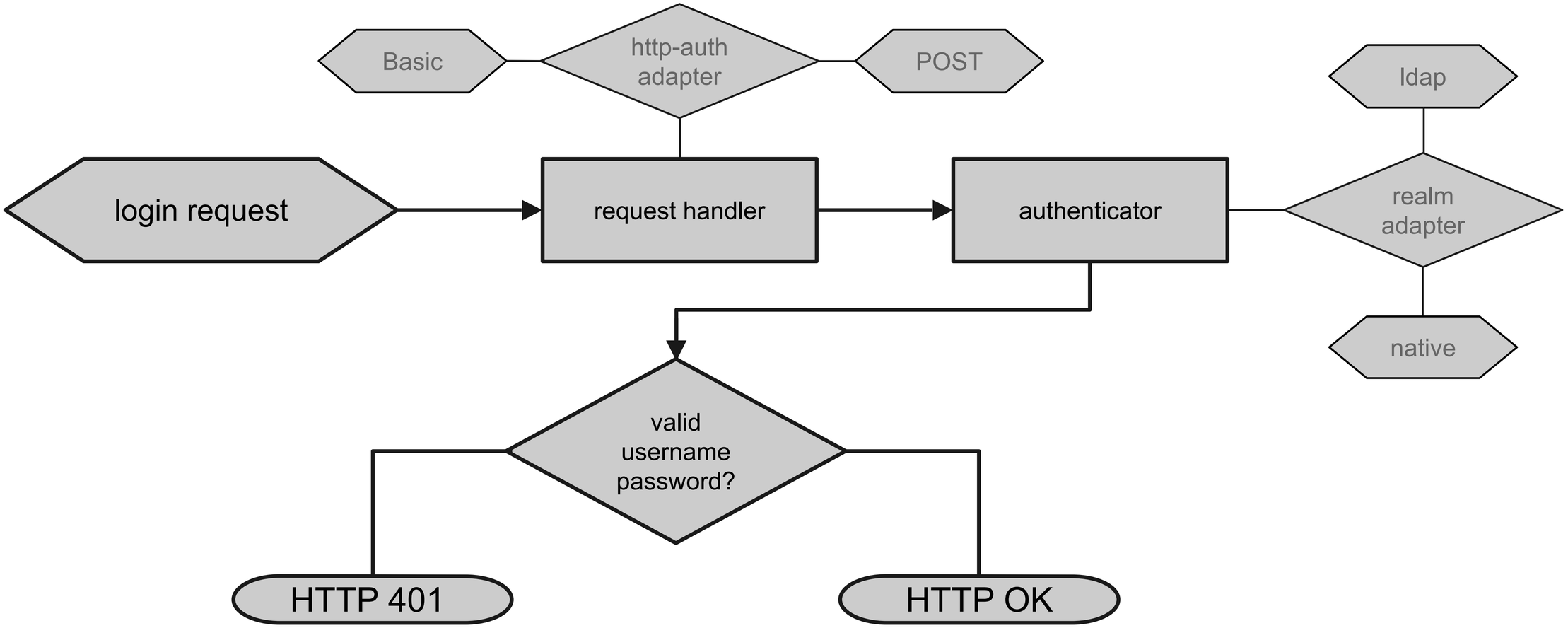
- Material drawn from presentations/blogs/articles/documentation authored by Grant Ingersoll, Cassandra Targett, Mitzi Morris, David Arthur, Matt Hoffman, Jim Walker, Yann Yu, Matt Mitchell, Evan Sayer, Evan Pease, Fran Lukesh, Andy Wibbels, Marcelline Saunders, Drew Oetzel, Ravi Krishnamurthy and many others....

Fusion Security (Optional)

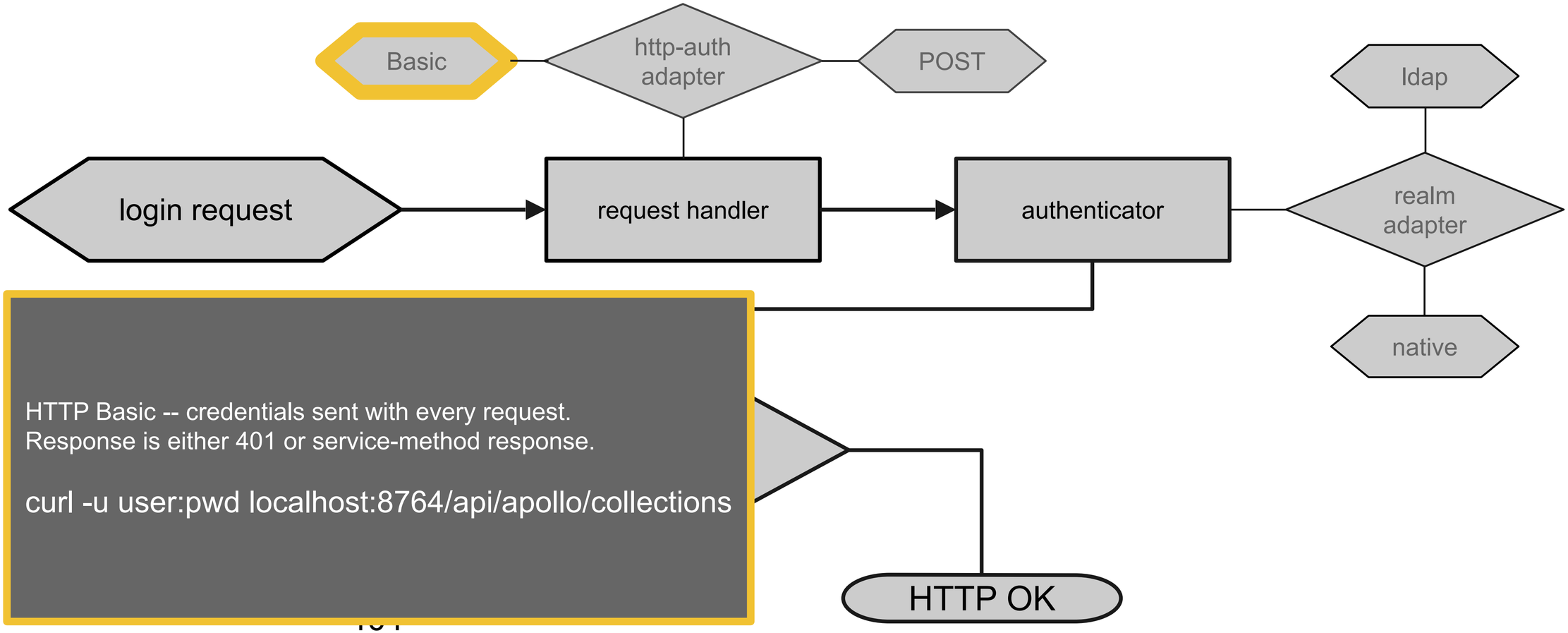
Topics

Authentication
Authorization
Permissions
Roles
Admin UI
Known Issues
Roadmap / Fusion 1.3

Authentication



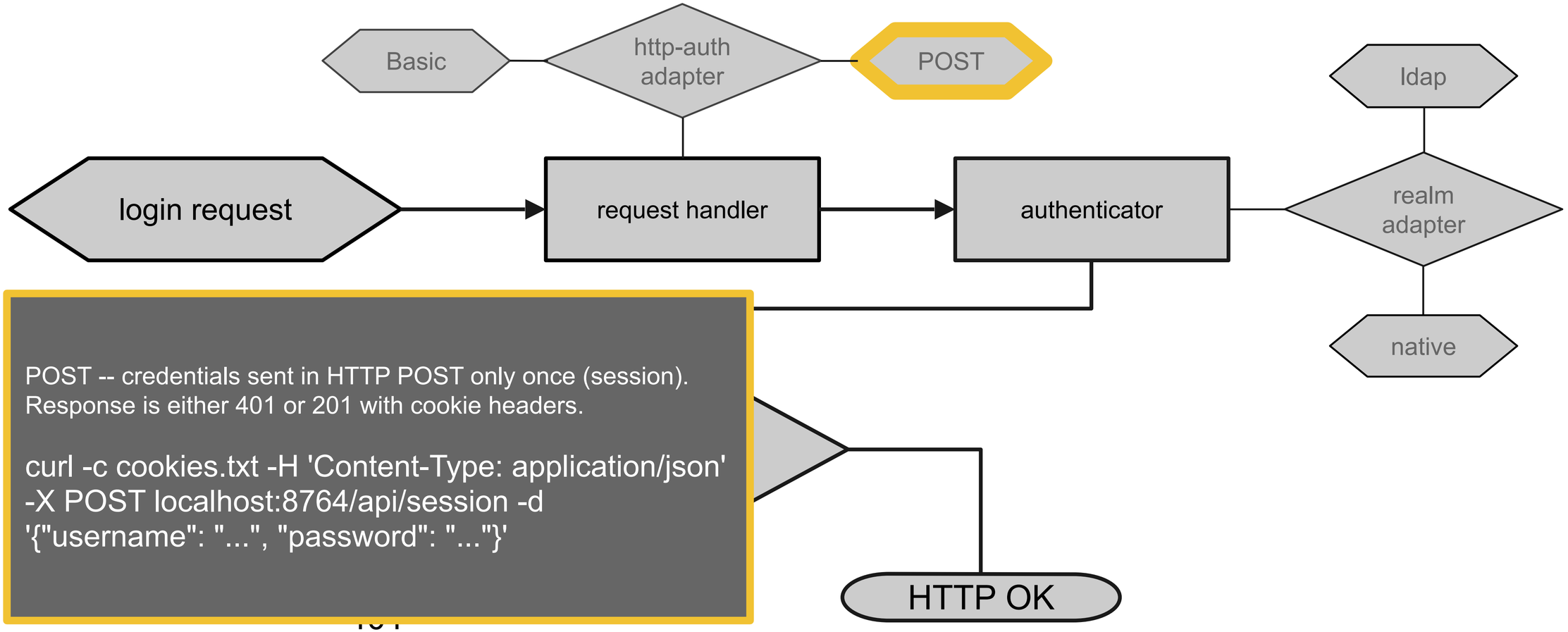
Authentication



HTTP Basic -- credentials sent with every request.
Response is either 401 or service-method response.

curl -u user:pwd localhost:8764/api/apollo/collections

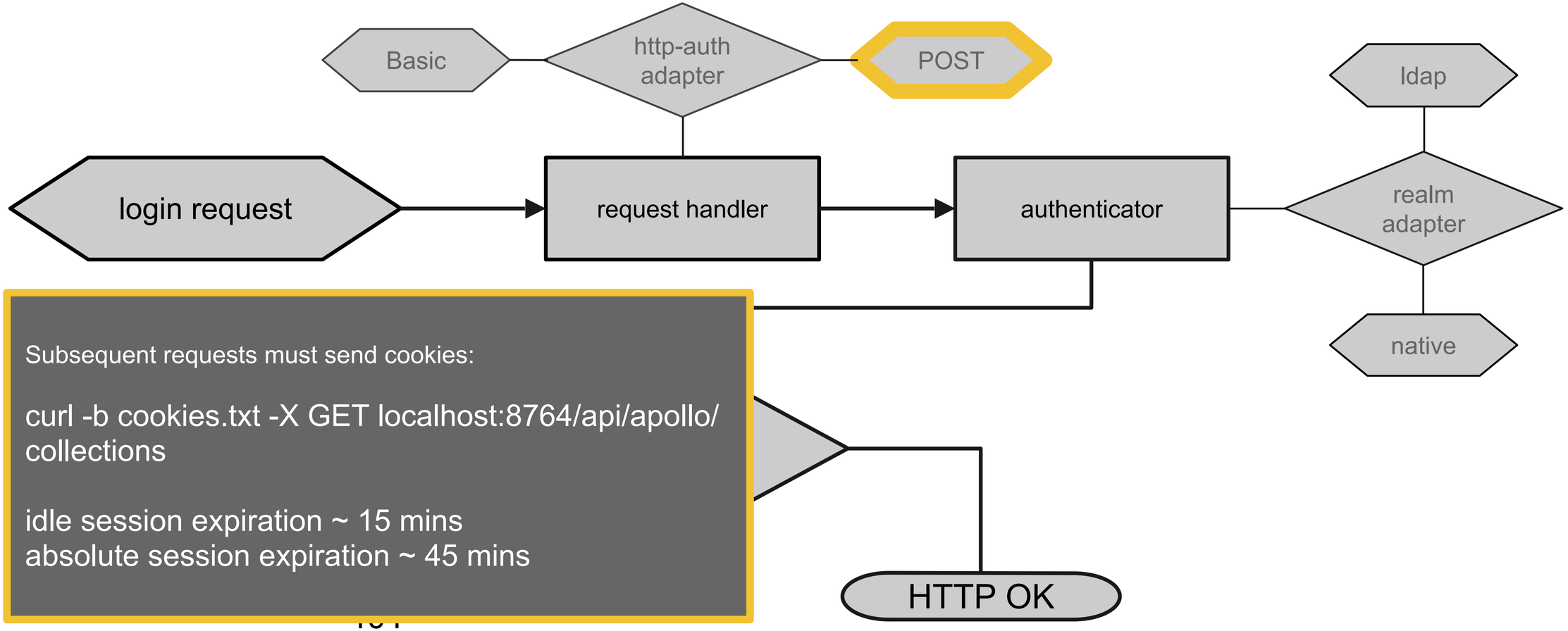
Authentication



POST -- credentials sent in HTTP POST only once (session).
Response is either 401 or 201 with cookie headers.

```
curl -c cookies.txt -H 'Content-Type: application/json'
-X POST localhost:8764/api/session -d
'{"username": "...", "password": "..."}'
```

Authentication

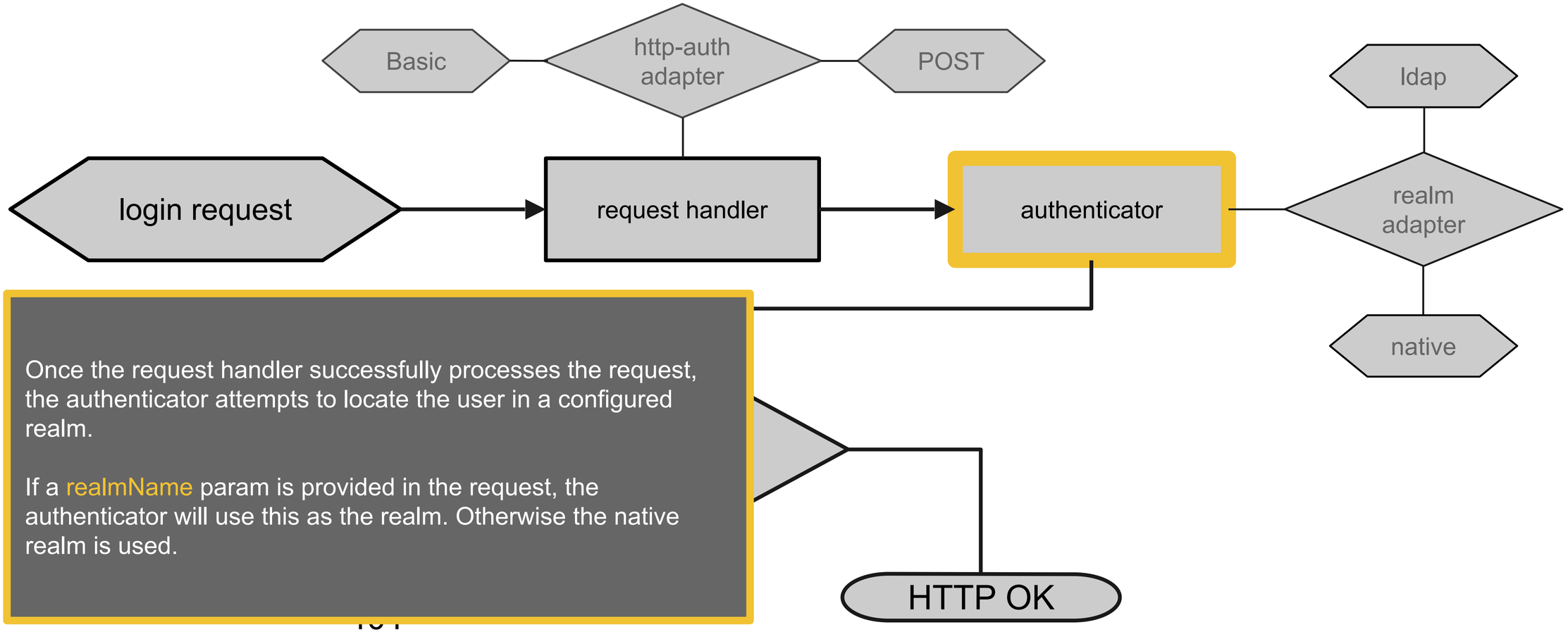


Subsequent requests must send cookies:

curl -b cookies.txt -X GET localhost:8764/api/apollo/collections

idle session expiration ~ 15 mins
absolute session expiration ~ 45 mins

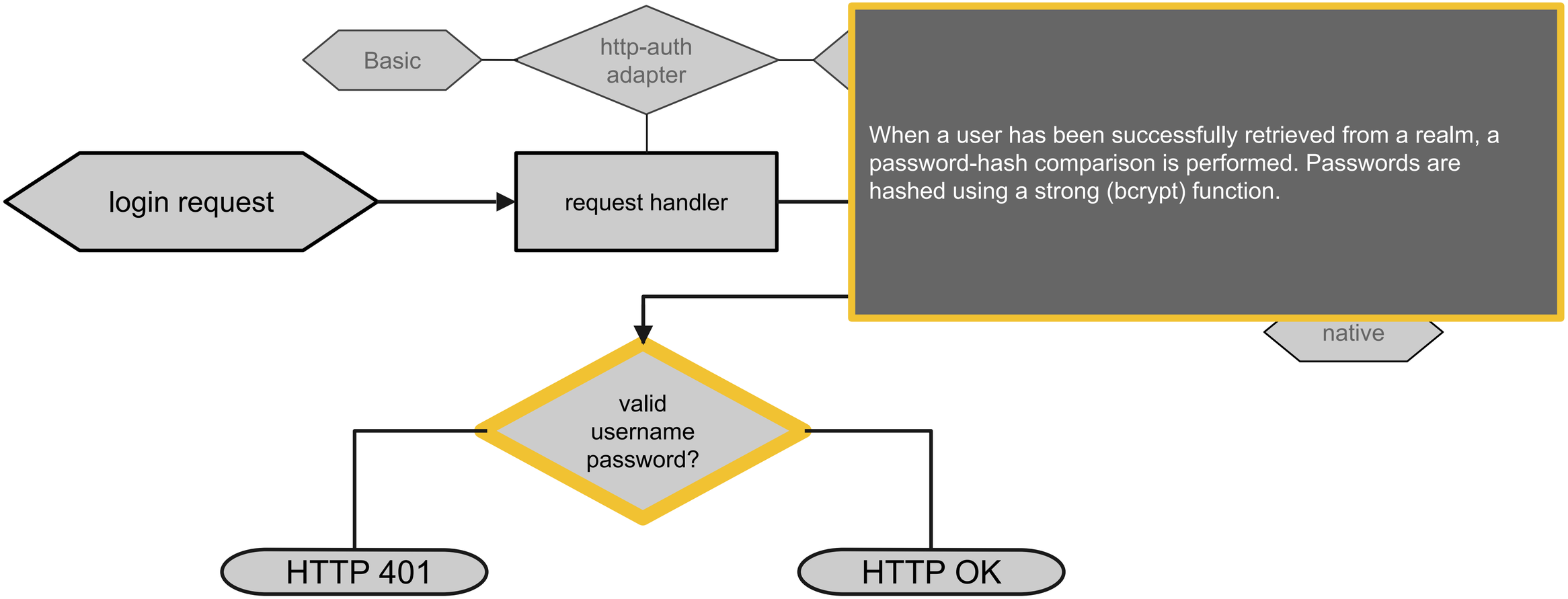
Authentication



Once the request handler successfully processes the request, the authenticator attempts to locate the user in a configured realm.

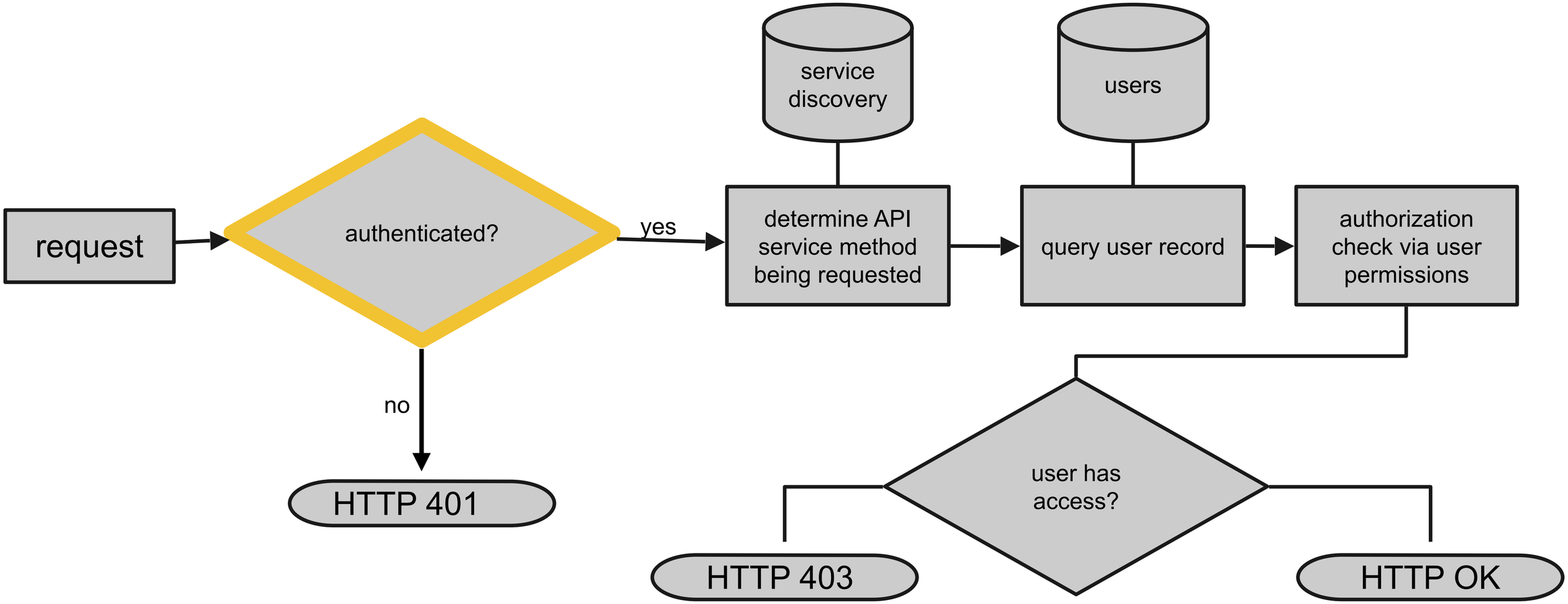
If a realmName param is provided in the request, the authenticator will use this as the realm. Otherwise the native realm is used.

Authentication

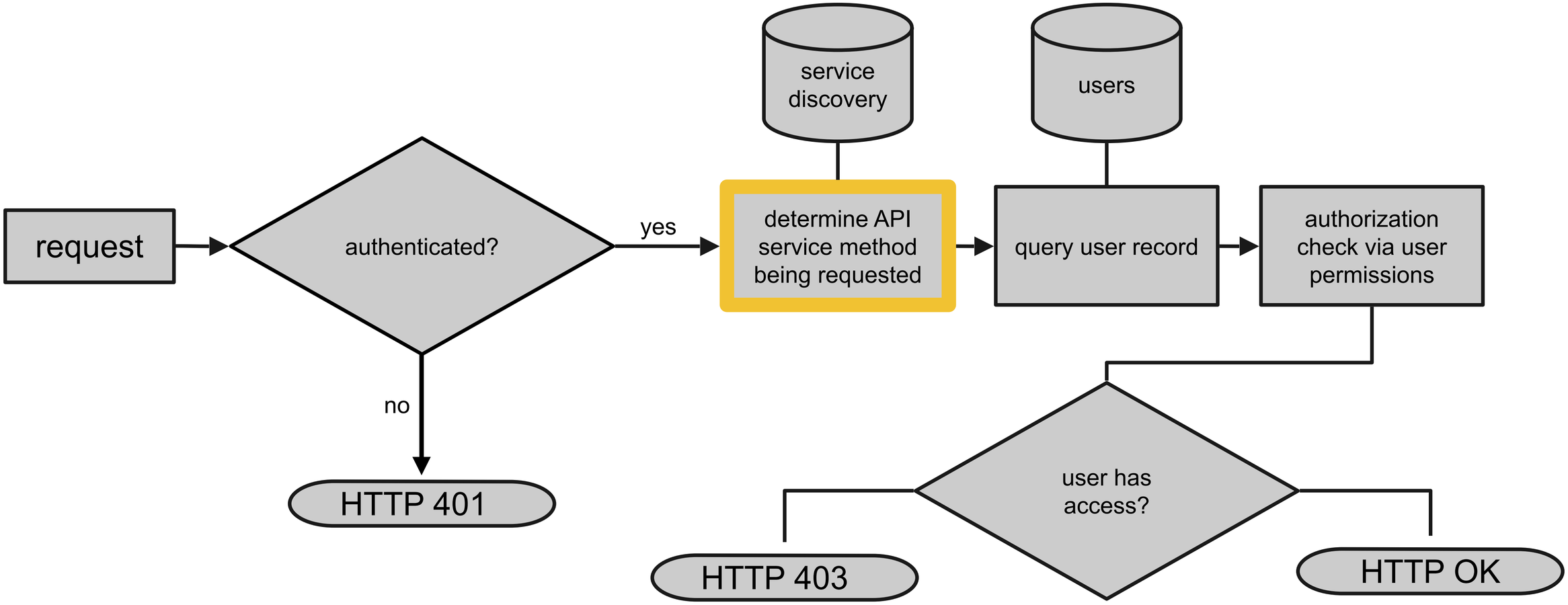


When a user has been successfully retrieved from a realm, a password-hash comparison is performed. Passwords are hashed using a strong (bcrypt) function.

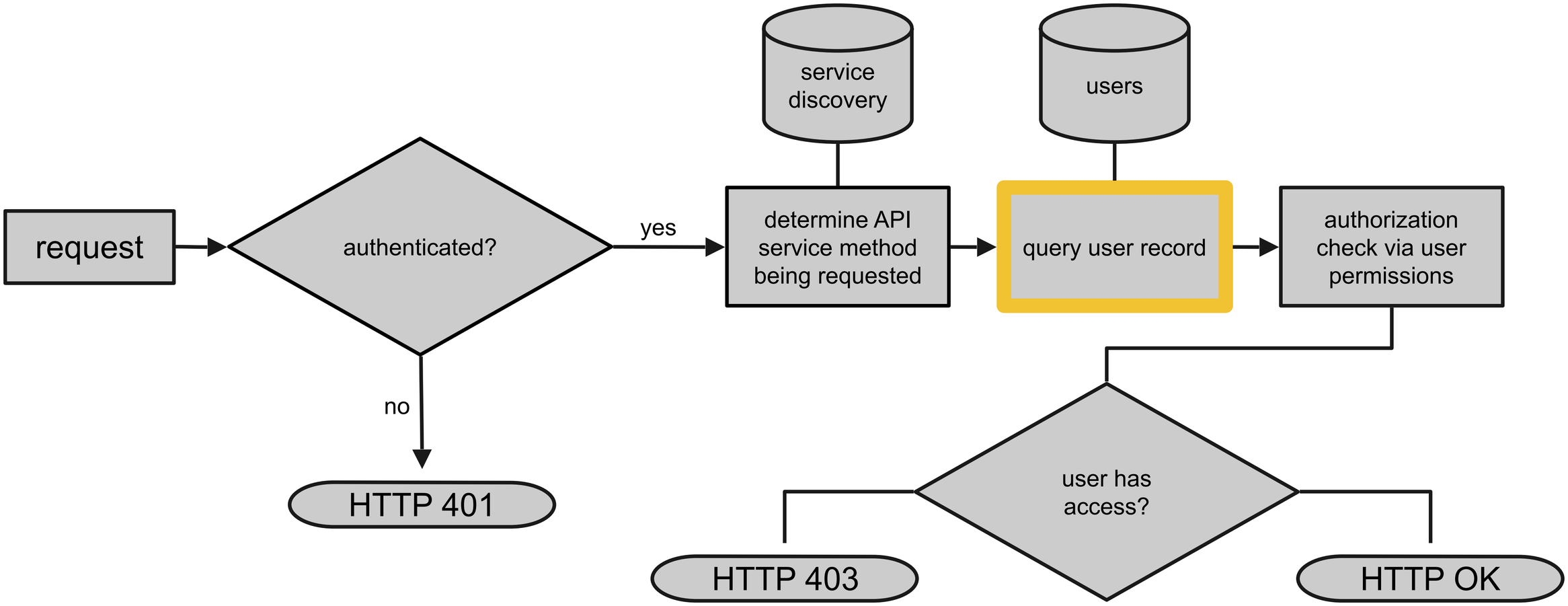
Authorization



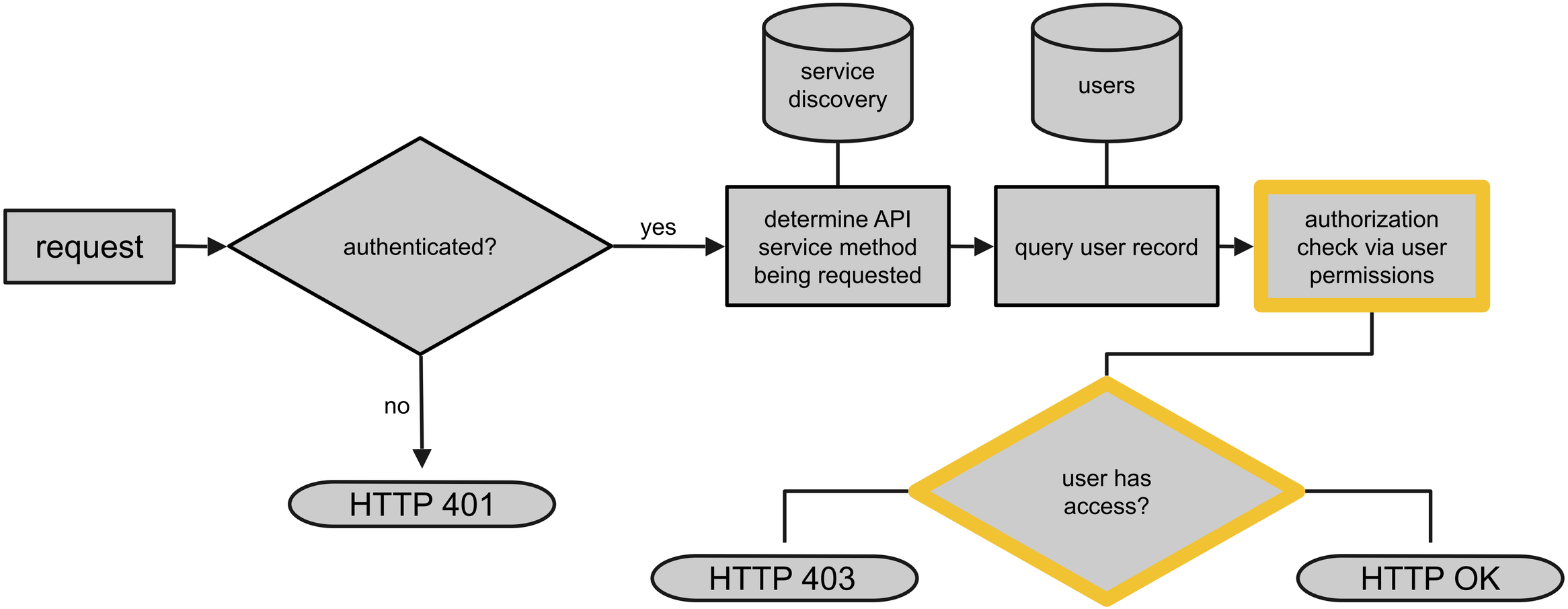
Authorization



Authorization



Authorization



Permissions

- uses Apache Shiro library
- a user can have many permissions
- permissions describe what a user can do, not what a user can’t do
- permissions map directly to service methods and HTTP verbs

Permission Structure

service:method:ID

Permission Structure

The service component of a permission maps directly to an Apollo service name.

collections:method:ID

Permission Structure

The method component of a permission maps to either an HTTP verb (#GET,#POST etc.) or an explicit service method (getCollection etc.).

collections:getCollection:ID

Permission Structure

The ID component of a permission maps to an instance of a resource, represented by the service.

collections:getCollection:foo

Permission Structure

Permissions components can have multiple values

collections:getCollection:foo,logs

Roles

- Unique name
- Named sets of permissions
- Roles can inherit from other roles, but can’t override permissions
- Users link to a role to inherit role permissions - no overrides

Admin UI

- Users and roles CRUD UI
- Role names are used for lightweight UI authz

Known issues/limitations

- API list responses aren’t authz filtered
- Admin UI authorization is hardcoded to preset list of role names, not flexible

Roadmap / Fusion 1.3

- Bug fix for list filtering
- Introduce new user/role permissions for Fusion UI “apps” (search, collections, relevancy workbench etc.)
- Admin UI admin able to assign UI app permissions to users/roles
- Possible high level approach for dealing with authz: resource based (collections) in addition to API (existing)
- Kerberos
- Solr proxy authz (can query, delete, optimize, commit etc.)

Demo and Lab 5 (Optional)

Lab 5 (Optional)

- Create 2 or more collections with different schemas, datasources and data. You can use the collections created in previous labs or quickly create 2 new collections and crawl two different websites.
- Create a user who can search one collection and not the other.
- Create a user who can administer one collection and not the other.

Lucidworks