

CLOUDERA DEPLOYMENT GUIDE

Getting Started with Hadoop Tutorial

CLOUDERA

 Microsoft Azure

Table of contents

Setup	2-10
Showing big data value	11-15
Showing data hub value	16
Advanced analytics on the same platform	17-29
Data governance and compliance	30-37
The End Game	38

Setup

For the remainder of this tutorial, we will present examples in the context of a fictional corporation called DataCo. Our mission is to help this organization get better insight by asking bigger questions.

SCENARIO:

Your Management: is talking euphorically about Big Data.

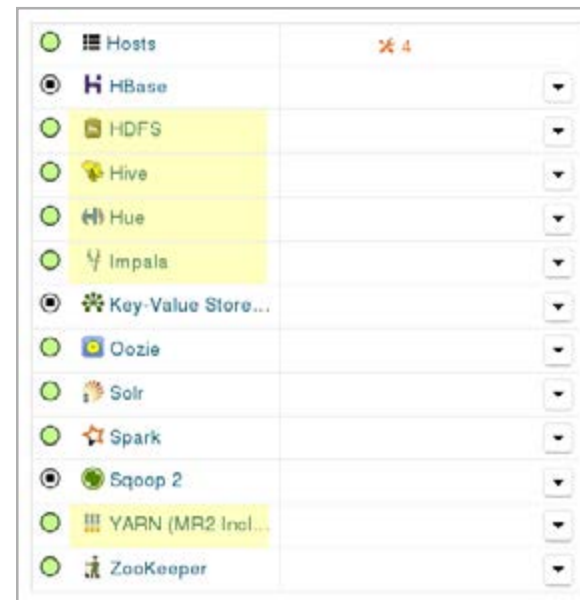
You: are carefully skeptical, as it will most likely all land on your desk anyway. Or, it has already landed on you, with the nice project description of: Go figure this Hadoop thing out.

PREPARATION:

Verify your environment. Go to Cloudera Manager in your demo environment and make sure the following services are up and running (have a green status dot next to them in the Cloudera Manager HOME Status view):

- **Apache Impala** - You will use this for interactive query
- **Apache Hive** - You will use for structure storage (i.e. tables in the Hive metastore)
- **HUE** - You will use for end user query access
- **HDFS** - You will use for distributed data storage
- **YARN** - This is the processing framework used by Hive (includes MR2)

If any of the services show yellow or red, restart the service or reach out to this discussion forum for further assistance.



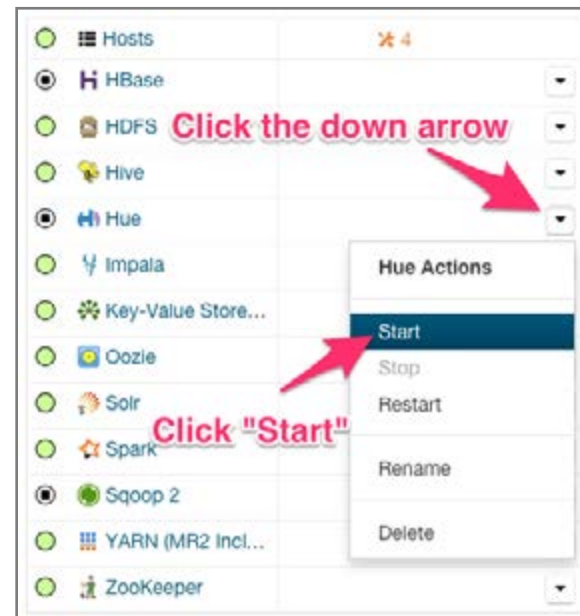
Setup

For the remainder of this tutorial, we will present examples in the context of a fictional corporation called DataCo. Our mission is to help this organization get better insight by asking bigger questions.

STARTING / RESTARTING A SERVICE:

1. Click on the dropdown menu to the right of the service name.
2. Click on Start or Restart.
3. Wait for your service to turn to green

Now that you have verified that your services are healthy and showing green, you can continue.



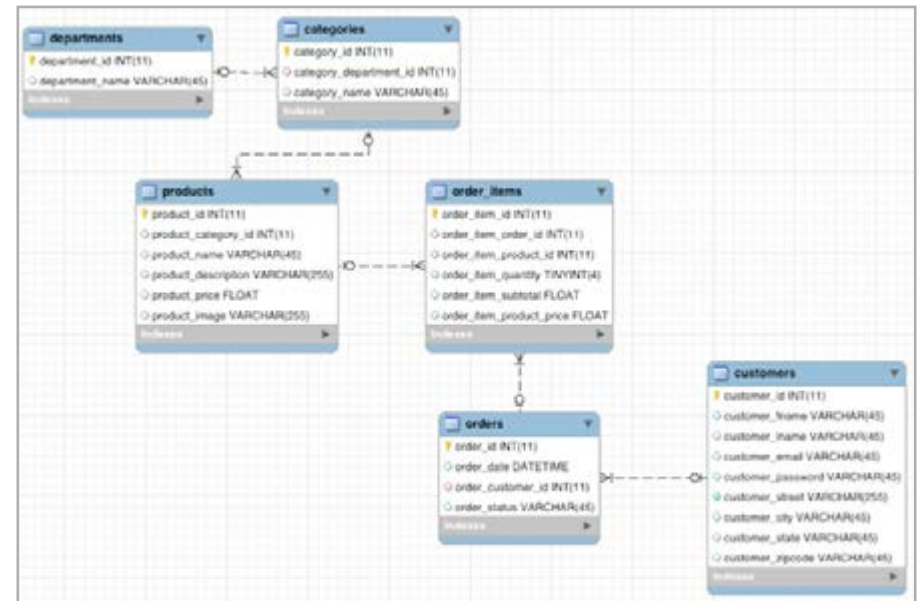
Exercise 1: Ingest and query relational data

In this scenario, DataCo's business question is: What products do our customers like to buy? To answer this question, the first thought might be to look at the transaction data, which should indicate what customers do buy and like to buy, right?

This is probably something you can do in your regular RDBMS environment, but a benefit of Apache Hadoop is that you can do it at greater scale at lower cost, on the same system that you may also use for many other types of analysis.

What this exercise demonstrates is how to do the same thing you already know how to do, but in CDH. Seamless integration is important when evaluating any new infrastructure. Hence, it's important to be able to do what you normally do, and not break any regular BI reports or workloads over the dataset you plan to migrate.

To analyze the transaction data in the new platform, we need to ingest it into the Hadoop Distributed File System (HDFS). We need to find a tool that easily transfers structured data from a RDBMS to HDFS, while preserving structure. That enables us to query the data, but not interfere with or break any regular workload on it.



Exercise 1: Ingest and query relational data

Apache Sqoop, which is part of CDH, is that tool. The nice thing about Sqoop is that we can automatically load our relational data from MySQL into HDFS, while preserving the structure. With a few additional configuration parameters, we can take this one step further and load this relational data directly into a form ready to be queried by Apache Impala, the MPP analytic database included with CDH, and other workloads.

You should first log in to the Master Node of your cluster via a terminal. Then, launch the Sqoop job:

```
> sqoop import-all-tables \  
-m {{cluster_data.worker_node_hostname.length}} \  
  --connect jdbc:mysql://{{cluster_data.manager_  
node_hostname}}:3306/retail_db \  
  --username=retail_dba \  
  --password=cloudera \  
  --compression-codec=snappy \  
  --as-parquetfile \  
  --warehouse-dir=/user/hive/warehouse \  
  --hive-import
```

This command may take a while to complete, but it is doing a lot. It is launching MapReduce jobs to pull the data from our MySQL database and write the data to HDFS in parallel, distributed across the cluster in Apache Parquet format. It is also creating tables to represent the HDFS files in Impala/ Apache Hive with matching schema.

Parquet is a format designed for analytical applications on Hadoop. Instead of grouping your data into rows like typical data formats, it groups your data into columns. This is ideal for many analytical queries where instead of retrieving data from specific records, you're analyzing relationships between specific variables across many records. Parquet is designed to optimize data storage and retrieval in these scenarios.

Exercise 1: Ingest and query relational data

VERIFICATION

When this command is complete, confirm that your data files exist in HDFS.

```
> hadoop fs -ls /user/hive/warehouse/  
> hadoop fs -ls /user/hive/warehouse/categories/
```

These commands to your right will show the directories and the files inside them that make up your tables.

Note: The number of .parquet files shown will be equal to what was passed to Sqoop with the -m parameter. This is the number of 'mappers' that Sqoop will use in its MapReduce jobs. It could also be thought of as the number of simultaneous connections to your database, or the number of disks / Data Nodes you want to spread the data across. So, on a single-node you will just see one, but larger clusters will have a greater number of files.

```
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=26328  
HDFS: Number of bytes written=67816  
HDFS: Number of read operations=9  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=27  
  
Job Counters  
  Launched map tasks=3  
  Other local map tasks=3  
  Total time spent by all mappers in occupied slots (ms)=27336  
  Total time spent by all reducers in occupied slots (ms)=0  
  Total time spent by all map tasks (ms)=27336  
  Total vcore-seconds taken by all map tasks=27336  
  Total megabyte-seconds taken by all map tasks=29012544  
  
Map-Reduce Framework  
  Map input records=1345  
  Map output records=1345  
  Input split bytes=364  
  Spilled Records=0  
  Failed Shuffles=0  
  Merged Map outputs=0  
  GC time elapsed (ms)=201  
  CPU time spent (ms)=12339  
  Physical memory (bytes) snapshot=1844811776  
  Virtual memory (bytes) snapshot=4748362538  
  Total committed heap usage (bytes)=1366827948  
  
File Input Format Counters  
  Bytes Read=0  
  
File Output Format Counters  
15-06-22 14:32:35 280 MapReduce: ImportJobInput: Transferred 51.4992 MB in 36.9812 seconds (1.3889 MB/sec)  
15-06-22 14:32:35 280 MapReduce: ImportJobInput: Retrieved 1345 records.  
[root@cloudera1 ~]# hadoop fs -ls /user/hive/warehouse/  
Found 5 items  
drwxrwxrwt - root hive          0 2015-06-22 14:29 /user/hive/warehouse/categories  
drwxrwxrwt - root hive          0 2015-06-22 14:29 /user/hive/warehouse/customers  
drwxrwxrwt - root hive          0 2015-06-22 14:29 /user/hive/warehouse/departments  
drwxrwxrwt - root hive          0 2015-06-22 14:31 /user/hive/warehouse/order_items  
drwxrwxrwt - root hive          0 2015-06-22 14:32 /user/hive/warehouse/orders  
drwxrwxrwt - root hive          0 2015-06-22 14:32 /user/hive/warehouse/products  
[root@cloudera1 ~]#  
[root@cloudera1 ~]# hadoop fs -ls /user/hive/warehouse/categories/  
Found 8 items  
drwxrwxrwt - root hive          0 2015-06-22 14:29 /user/hive/warehouse/categories/.metadata  
-rw-r--r--  2 root supergroup    1295 2015-06-22 14:29 /user/hive/warehouse/categories/282-8770-6a55-41b4-0137-1476d41d6d14.parquet  
-rw-r--r--  2 root supergroup    1281 2015-06-22 14:29 /user/hive/warehouse/categories/92580bc0-9b7b-4258-b747-bd96e0557aa1.parquet  
-rw-r--r--  2 root supergroup    1234 2015-06-22 14:29 /user/hive/warehouse/categories/d714a8b0-d86d-44a0-b364-b34ed03d7523.parquet  
[root@cloudera1 ~]#
```


Exercise 1: Ingest and query relational data

Hive and Impala also allow you to create tables by defining a schema over existing files with 'CREATE EXTERNAL TABLE' statements, similar to traditional relational databases. But Sqoop already created these tables for us, so we can go ahead and query them.

We're going to use Hue's Impala app to query our tables. Hue provides a web-based interface for many of the tools in CDH and can be found on port 8888 of your Manager Node. In the QuickStart VM, the administrator username for Hue is 'cloudera' and the password is 'cloudera'.

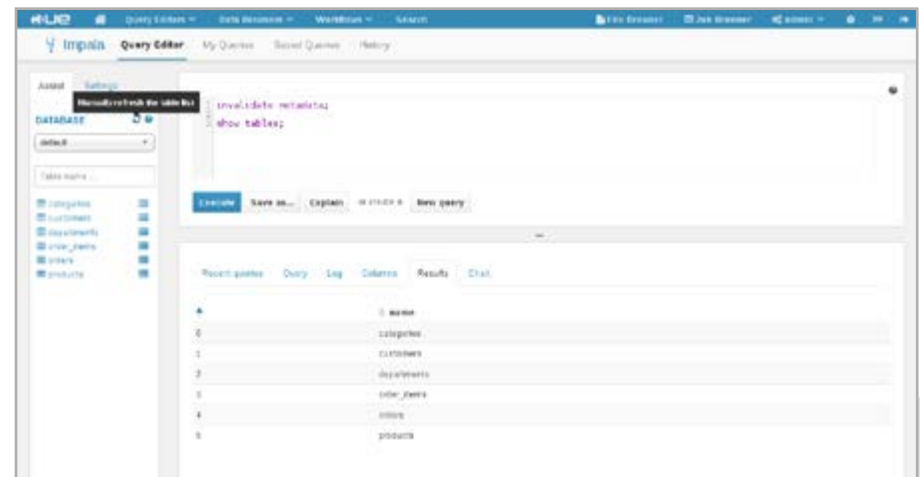
Once you are inside of Hue, click on Query Editors, and open the Impala Query Editor.



To save time during queries, Impala does not poll constantly for metadata changes. So, the first thing we must do is tell Impala that its metadata is out of date. Then we should see our tables show up, ready to be queried:

```
invalidate metadata;  
show tables;
```

You can also click on the "Refresh Table List" icon on the left to see your new tables in the side menu.

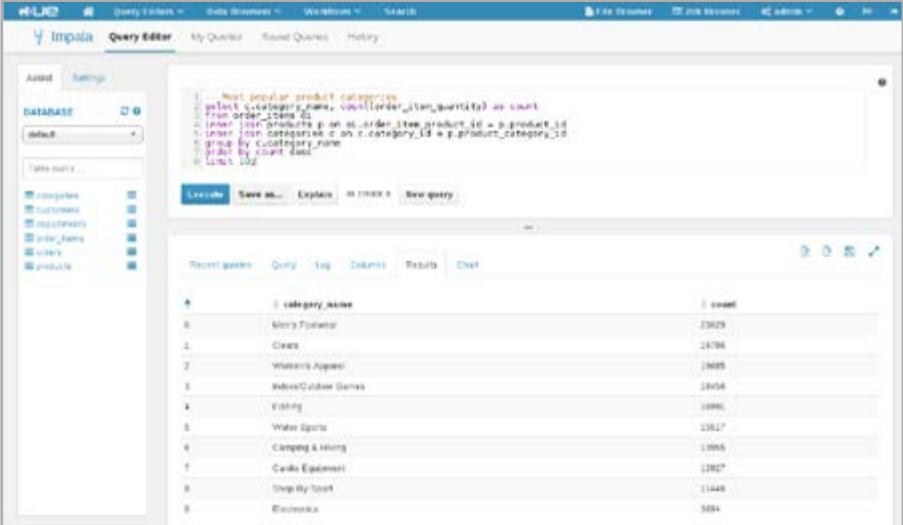


Exercise 1: Ingest and query relational data

Now that your transaction data is readily available for structured queries in CDH, it's time to address DataCo's business question. Copy and paste or type in the following standard SQL example queries for calculating total revenue per product and showing the top 10 revenue generating products:

```
-- Most popular product categories
select c.category_name, count(order_item_quantity) as count
from order_items oi
inner join products p on oi.order_item_product_id =
p.product_id
inner join categories c on c.category_id = p.product_
category_id
group by c.category_name
order by count desc
limit 10;
```

You should see results of the following form:



The screenshot shows the Cloudera Impala Query Editor interface. The query editor contains the following SQL query:

```
1: -- Most popular product categories
2: select c.category_name, count(order_item_quantity) as count
3: from order_items oi
4: inner join products p on oi.order_item_product_id = p.product_id
5: inner join categories c on c.category_id = p.product_category_id
6: group by c.category_name
7: order by count desc
8: limit 10;
```

The results are displayed in a table with the following columns: category_name and count. The results are as follows:

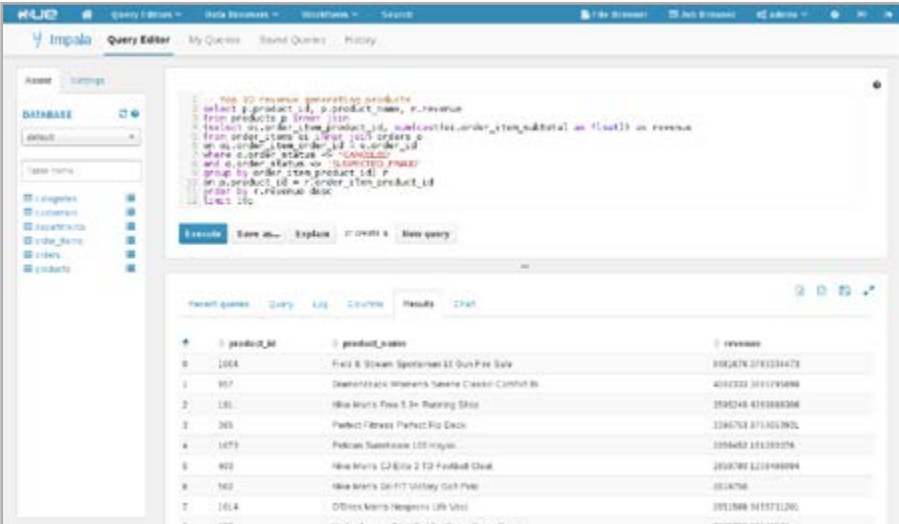
category_name	count
Men's Footwear	23629
Class	21796
Women's Apparel	19885
Indoor/Outdoor Games	18054
Fishing	16996
Water Sports	13527
Camping & Hiking	12995
Card Games	12827
Travel Kits	11448
Electronics	3894

Exercise 1: Ingest and query relational data

Clear out the previous query, and replace it with the following:

```
-- top 10 revenue generating products
select p.product_id, p.product_name, r.revenue
from products p inner join
(select oi.order_item_product_id, sum(cast(oi.order_item_subtotal as float)) as revenue
from order_items oi inner join orders o
on oi.order_item_order_id = o.order_id
where o.order_status <> 'CANCELED'
and o.order_status <> 'SUSPECTED_FRAUD'
group by order_item_product_id) r
on p.product_id = r.order_item_product_id
order by r.revenue desc
limit 10;
```

You should see results similar to this:



The screenshot shows the Impala Query Editor interface. The query editor contains the same SQL query as shown in the previous block. The results pane displays the top 10 revenue-generating products. The results are as follows:

product_id	product_name	revenue
1004	Free & Screen Spectacles 10 Sun Free Sale	8182476.378334478
997	Deamontack Women's Tennis Classic Comfort B	4092333.391279888
1001	Hila Men's Free 5 On Running Shoe	3595245.476388336
1003	Perfect Fitness Perfect Fit Deck	3366753.471603902
1079	Polcom Sunscreen 100 Spray	3366682.113380076
999	Hila Men's 12.5oz 2 Top Football Cleat	2595780.521848894
1002	Hila Men's 10-FIT Victory Soft Pad	2514756
1014	O'Drick Men's Hengens 100 Volo	2011848.919771260
991	Under Armour Girls' Tumble Soccer Soccer Ball	1285729.511123242

You may notice that we told Sqoop to import the data into Hive but used Impala to query the data. This is because Hive and Impala can share both data files and the table metadata. Hive works by compiling SQL queries into MapReduce jobs, which makes it very flexible, whereas Impala executes queries itself and is built from the ground up to be as fast as possible, which makes it better for interactive analysis. We'll use Hive later for an ETL (extract-transform-load) workload.

Exercise 1: Ingest and query relational data

CONCLUSION

Now that you have gone through the first basic steps to Sqoop structured data into HDFS, transform it into Parquet file format, and create hive tables for use when you query this data.

You have also learned how to query tables using Impala and that you can use regular interfaces and tools (such as SQL) within a Hadoop environment as well. The idea here being that you can do the same reports you usually do, but where the architecture of Hadoop vs traditional systems provides much larger scale and flexibility.

Showing big data value

SCENARIO:

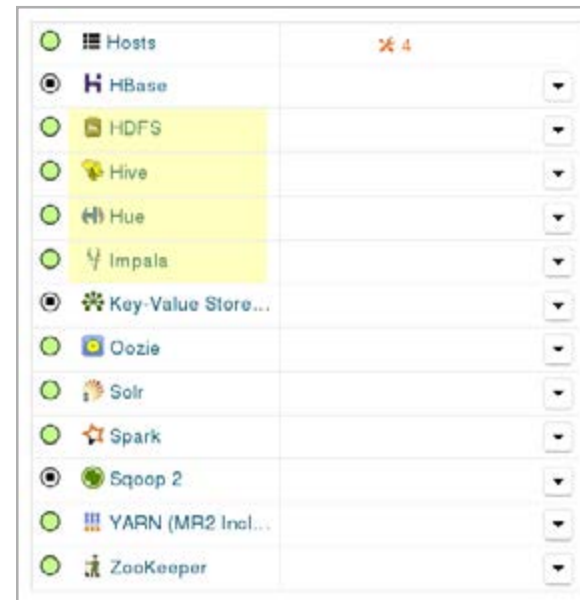
Your Management: is indifferent, you produced what you always produce - a report on structured data, but you really didn't prove any additional value.

You: are either also indifferent and just go back to what you have always done... or you have an ace up your sleeve.

PREPARATION:

Go to Cloudera Manager's home page and verify the following services are up:

- Impala
- Hive
- HDFS
- Hue



Exercise 2: Correlate structured data with unstructured data

Since you are a pretty smart data person, you realize another interesting business question would be: are the most viewed products also the most sold? Since Hadoop can store unstructured and semi-structured data alongside structured data without remodeling an entire database, you can just as well ingest, store, and process web log events. Let's find out what site visitors have viewed the most.

For this, you need the web clickstream data. The most common way to ingest web clickstream is to use Apache Flume. Flume is a scalable real-time ingest framework that allows you to route, filter, aggregate, and do "mini-operations" on data on its way in to the scalable processing platform.

In Exercise 4, later in this tutorial, you can explore a Flume configuration example, to use for real-time ingest and transformation of our sample web clickstream data. However, for the sake of tutorial-time, in this step, we will not have the patience to wait for three days of data to be ingested. Instead, we prepared a web clickstream data set (just pretend you fast forwarded three days) that you can bulk upload into HDFS directly.

BULK UPLOAD DATA

For your convenience, we have pre-loaded some sample access log data into /opt/examples/log_data/access.log.2. Let's move this data from the local filesystem, into HDFS.

```
> sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse/  
original_access_logs  
> sudo -u hdfs hadoop fs -copyFromLocal /opt/examples/  
log_files/access.log.2 /user/hive/warehouse/original_  
access_logs
```

The copy command may take several minutes to complete. Verify that your data is in HDFS by executing the following command:

```
> hadoop fs -ls /user/hive/warehouse/original_access_logs
```

Exercise 2: Correlate structured data with unstructured data

Now you can build a table in Hive and query the data via Apache Impala and Hue. You'll build this table in 2 steps. First, you'll take advantage of Hive's flexible SerDes (serializers / deserializers) to parse the logs into individual fields using a regular expression. Second, you'll transfer the data from this intermediate table to one that does not require any special SerDe. Once the data is in this table, you can query it much faster and more interactively using Impala.

We'll use the Hive Query Editor app in Hue to execute the following queries:

```
CREATE EXTERNAL TABLE intermediate_access_logs (  
  ip STRING,  
  date STRING,  
  method STRING,  
  url STRING,  
  http_version STRING,  
  code1 STRING,  
  code2 STRING,  
  dash STRING,  
  user_agent STRING)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.  
serde2.RegexSerDe'  
WITH SERDEPROPERTIES (  

```

```
  'input.regex' = '([^ ]*) - - \\[[([\\^\\]]*)\\]  
"([\\^\\ ]*) ([\\^\\ ]*) ([\\^\\ ]*)" (\\d*) (\\d*) "([\\^"]*)"  
"([\\^"]*)"' ,  
  'output.format.string' = "%1$s %2$s %3$s %4$s  
%5$s %6$s %7$s %8$s %9$s")  
LOCATION '/user/hive/warehouse/original_access_logs';  
  
CREATE EXTERNAL TABLE tokenized_access_logs (  
  ip STRING,  
  date STRING,  
  method STRING,  
  url STRING,  
  http_version STRING,  
  code1 STRING,  
  code2 STRING,  
  dash STRING,  
  user_agent STRING)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
LOCATION '/user/hive/warehouse/tokenized_access_logs';  
ADD JAR {{lib_dir}}/hive/lib/hive-contrib.jar;  
INSERT OVERWRITE TABLE tokenized_access_logs SELECT *  
FROM intermediate_access_logs;
```

Exercise 2: Correlate structured data with unstructured data

The final query will take a minute to run. It is using a MapReduce job, just like our Sqoop import did, to transfer the data from one table to the other in parallel. You can follow the progress in the log below, and you should see the message 'The operation has no results.' when it's done.

Again, we need to tell Impala that some tables have been created through a different tool. Switch back to the Impala Query Editor app, and enter the following command:

```
invalidate metadata;
```

Now, if you enter the 'show tables;' query or refresh the table list in the left-hand column, you should see the two new external tables in the default database. Paste the following query into the Query Editor:

```
select count(*),url from tokenized_access_logs  
where url like '%\product\%'  
group by url order by count(*) desc;
```

By introspecting the results you quickly realize that this list contains many of the products on the most sold list from previous tutorial steps, but there is one product that did not show up in the previous result. There is one product that seems to be viewed a lot, but never purchased. Why?

Well, in our example with DataCo, once these odd findings are presented to your manager, it is immediately escalated. Eventually, someone figures out that on that view page, where most visitors stopped, the sales path of the product had a typo in the price for the item. Once the typo was fixed, and a correct price was displayed, the sales for that SKU started to rapidly increase.

Exercise 2: Correlate structured data with unstructured data

CONCLUSION

If you had lacked an efficient and interactive tool enabling analytics on high-volume semi-structured data, this loss of revenue would have been missed for a long time. There is risk of loss if an organization looks for answers within partial data. Correlating two data sets for the same business question showed value and being able to do so within the same platform made life easier for you and for the organization. If you'd like to dive deeper into Hive, Impala, and other tools for data analysis in Cloudera's platform, you may be interested in Data Analyst Training.

For now, we'll explore some different techniques.

Showing data hub value

SCENARIO:

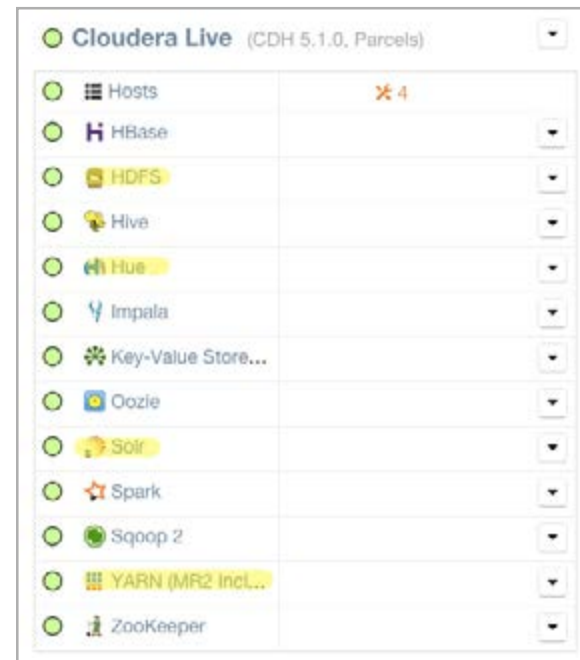
Your Management: can't believe the magic you do with data and is about to promote you and invest in a new team under your lead... when all hell breaks loose. You get an emergency call - as you are now the go-to person - and your manager is screaming about the loss of sales over the last three days.

You: from slightly excited to under the gun in seconds...well, lucky for you, there might be a quick way to find out what is happening.

PREPARATION:

Go to Cloudera Manager and verify these services are running:

- HDFS
- Hue
- Solr
- YARN



Advanced analytics on the same platform

SCENARIO:

Your Management: is of course thrilled with the recent discoveries you helped them with—you basically saved them a lot of money! They start giving you bigger questions, and more funding (we really hope the latter!)

You: are excited to dive into more advanced use cases, but you know that you'll need even more funding by the organization. You decide to really show off!

PREPARATION:

Go to Cloudera Manager and verify these services are running:

- HDFS
- Spark
- YARN / MR2



Exercise 3: Explore log events interactively

Since sales are dropping and nobody knows why, you want to provide a way for people to interactively and flexibly explore data from the website. We can do this by indexing it for use in Apache Solr, where users can do text searches, drill down through different categories, etc. Data can be indexed by Solr in batch using MapReduce, or you can index tables in Apache HBase and get real-time updates. To analyze data from the website, however, we're going to stream the log data in using Flume.

The web log data is a standard web server log which may look something like this:

Solr organizes data similarly to the way a SQL database does. Each record is called a 'document' and consists of fields defined by the schema: just like a row in a database table. Instead of a table, Solr calls it a 'collection' of documents. The difference is that data in Solr tends to be more loosely structured. Fields may be optional, and instead of always matching exact values, you can also enter text queries that partially match a field, just like you're searching for web pages. You'll also see Hue refer to 'shards' - and that's just the way Solr breaks collections up to spread them around the cluster so you can search all your data in parallel.

[illegible]

Exercise 3: Explore log events interactively

Here is how you can start real-time-indexing via Cloudera Search and Flume over the sample web server log data and use the Search UI in Hue to explore it:

CREATE YOUR SEARCH INDEX

Ordinarily when you are deploying a new search schema, there are four steps:

1. Creating an empty configuration

```
> solrctl --zk {{zookeeper_connection_string}}/solr
instancedir --generate solr_configs
```

The result of this command is a skeleton configuration that you can customize to your liking via the conf/schema.xml.

2. Edit your schema

The most likely area in conf/schema.xml that you would be interested in is the <fields></fields> section. From this area you can define the fields that are present and searchable in your index.

3. Uploading your configuration

```
> cd /opt/examples/flume
> solrctl --zk {{zookeeper_connection_string}}/solr
instancedir --create live_logs ./solr_configs
```

4. Creating your collection

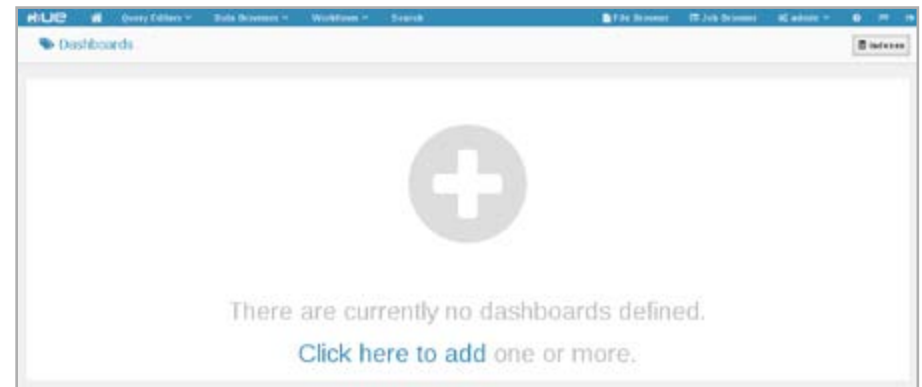
```
solrctl --zk {{zookeeper_connection_string}}/solr
collection --create live_logs -s {{ number of solr
servers }}
```

Exercise 3: Explore log events interactively

You may need to replace the IP addresses with those of your three data nodes. You can verify that you successfully created your collection in Solr by going to Hue, and clicking **Search** in the top menu

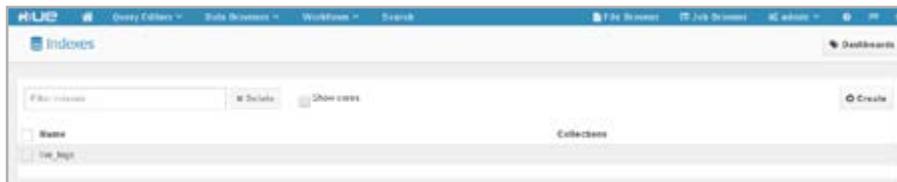


Then click on **Indexes** from the top right to see all the indexes/collections.



Exercise 3: Explore log events interactively

Now you can see the collection that we just created, **live_logs**, click on it.



You are now viewing the fields that we defined in our schema.xml file.

A screenshot of the HUE web interface showing the schema for the 'live_logs' collection. The left sidebar has 'ACTIONS' and 'Indexes' sections. The main area is titled 'Indexes > live_logs' and displays a table with the following fields and properties:

Name	Type	Unique key field	Required	Indexed	Stored
category	string		✓	✓	✓
product	string		✓	✓	✓
received_date	date		✓	✓	✓
id	int_general		✓	✓	✓
received	int_general		✓	✓	✓
department	string		✓	✓	✓
action	string		✓	✓	✓
_version	long		✓	✓	✓
it	string	✓	✓	✓	✓

At the bottom of the table, there are 'Add' and 'Delete' buttons.

Exercise 3: Explore log events interactively

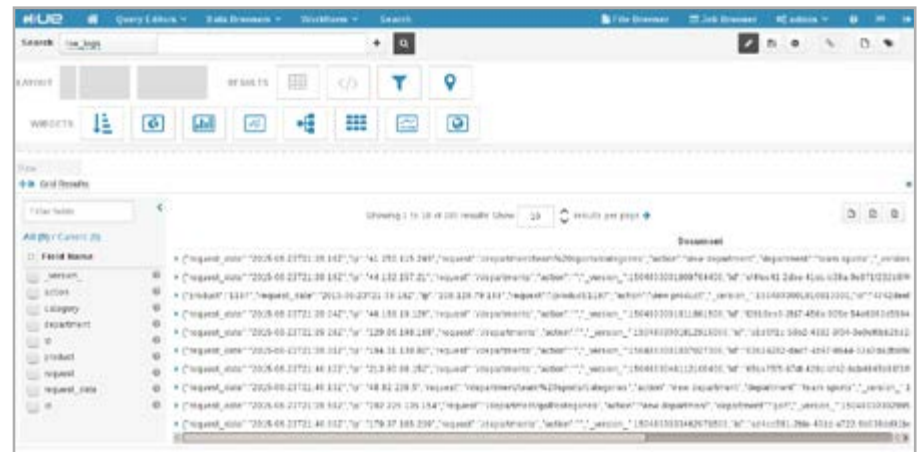
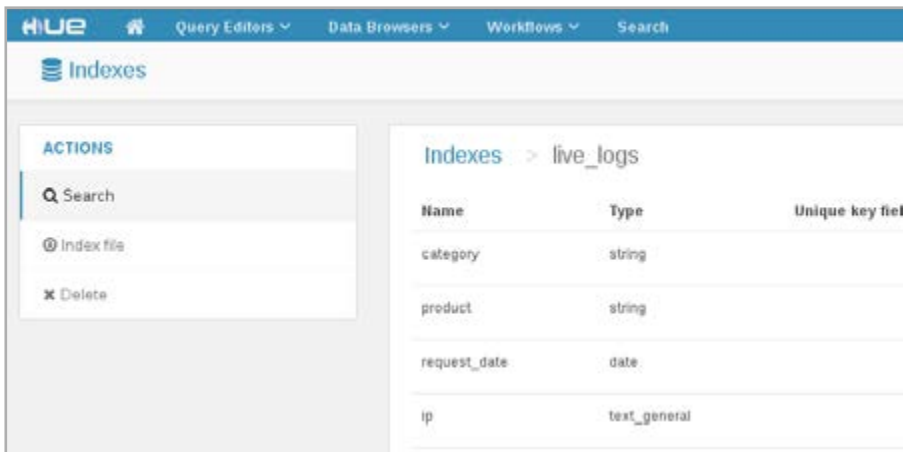
Now that you have verified that your search collection/index was created successfully, we can start putting data into it using Flume and Morphlines. Flume is a tool for ingesting streams of data into your cluster from sources such as log files, network streams, and more. Morphlines is a Java library for doing ETL on-the-fly, and it's an excellent companion to Flume. It allows you to define a chain of tasks like reading records, parsing and formatting individual fields, and deciding where to send them, etc. We've defined a morphline that reads records from Flume, breaks them into the fields we want to search on, and loads them into Solr (You can read more about Morphlines [here](#)).

This example Morphline is defined at `/opt/examples/flume/conf/morphline.conf`, and we're going to use it to index our records in real-time as they're created and ingested by Flume.

Exercise 3: Explore log events interactively

Now you can go back to the Hue UI, and click 'Search' from the collection's page:

You'll be able to search, drill down into, and browse events that have been indexed.

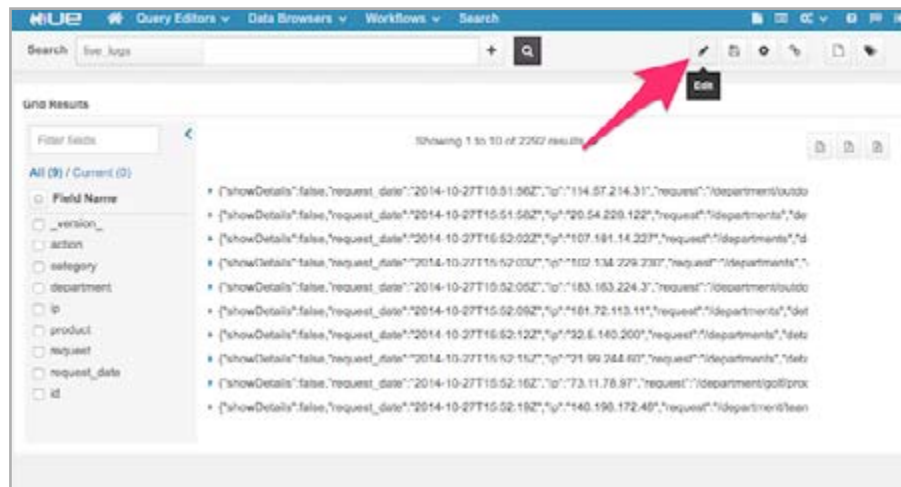


If one of these steps fails, please reach out to the Discussion Forum and get help. Otherwise, you can start exploring the log data and understand what is going on.

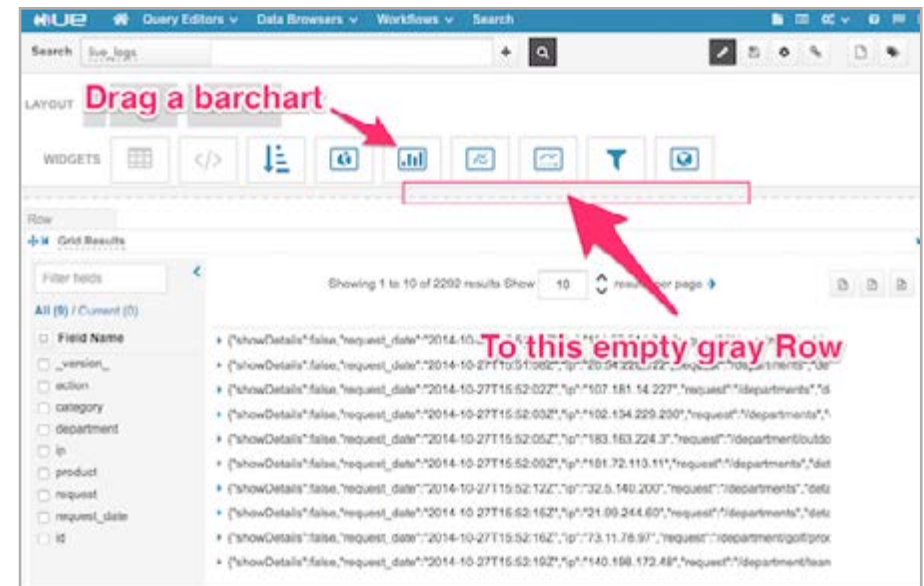
For our story's sake, we pretend that you started indexing data the same time as you started ingesting it (via Flume) to the platform, so that when your manager escalated the issue, you could immediately drill down into data from the last three days and explore what happened. For example, perhaps you noted a lot of DDOS events and could take the right measures to preempt the attack. Problem solved! Management is fantastically happy with your recent contributions, which of course leads to a great bonus or something similar. :D

Exercise 4: Building a dashboard

To get started with building a dashboard with Hue, click on the pen icon.

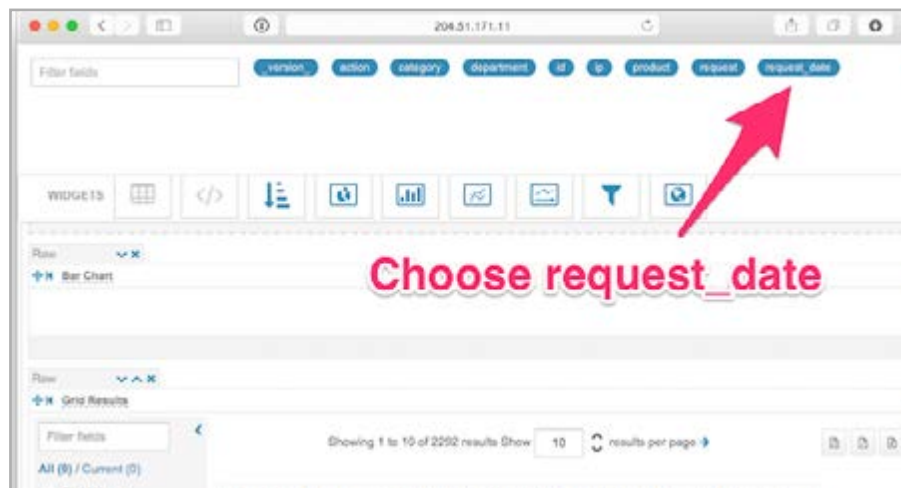


This will take you into the edit-mode where you can choose different widgets and layouts that you would like to see. You can choose a few options and configurations here, but for now, just drag a barchart into the top gray row.

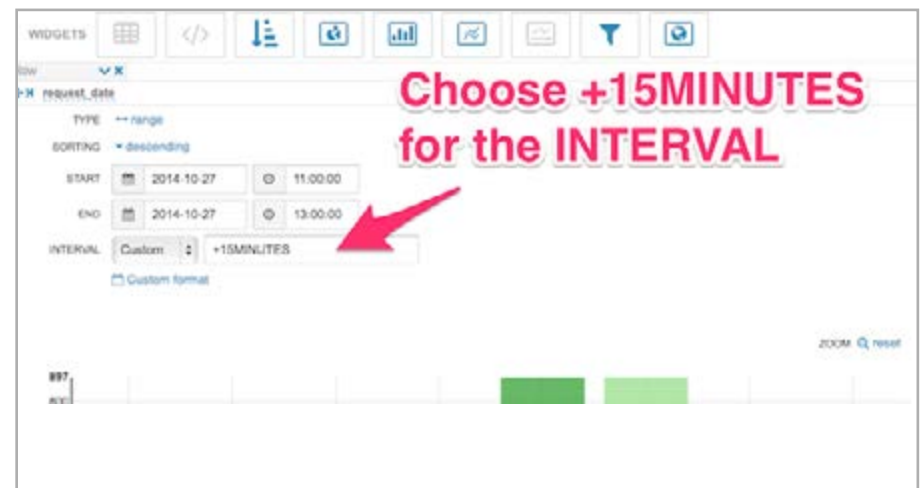


Exercise 4: Building a dashboard

This will bring up the list of fields that are present in our index so that you can choose which field you would like to group by. Let's choose **request_date**.



For the sake of our display, choose +15MINUTES for the INTERVAL.

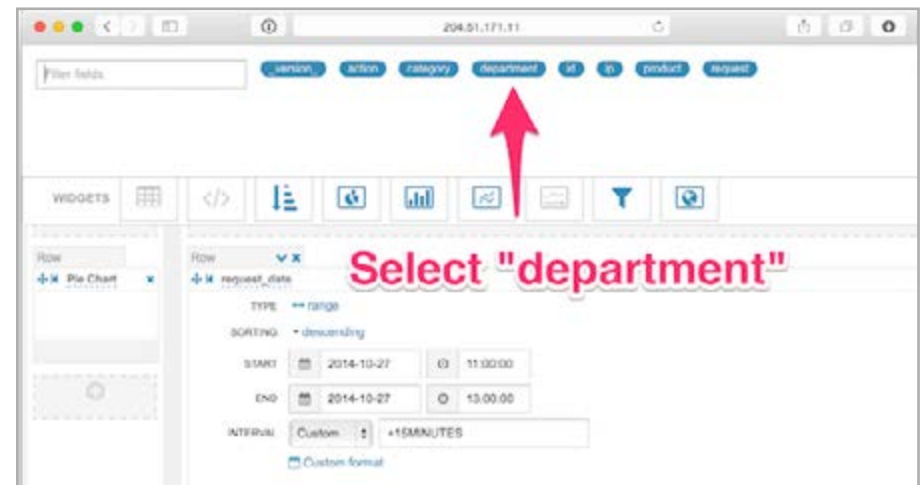
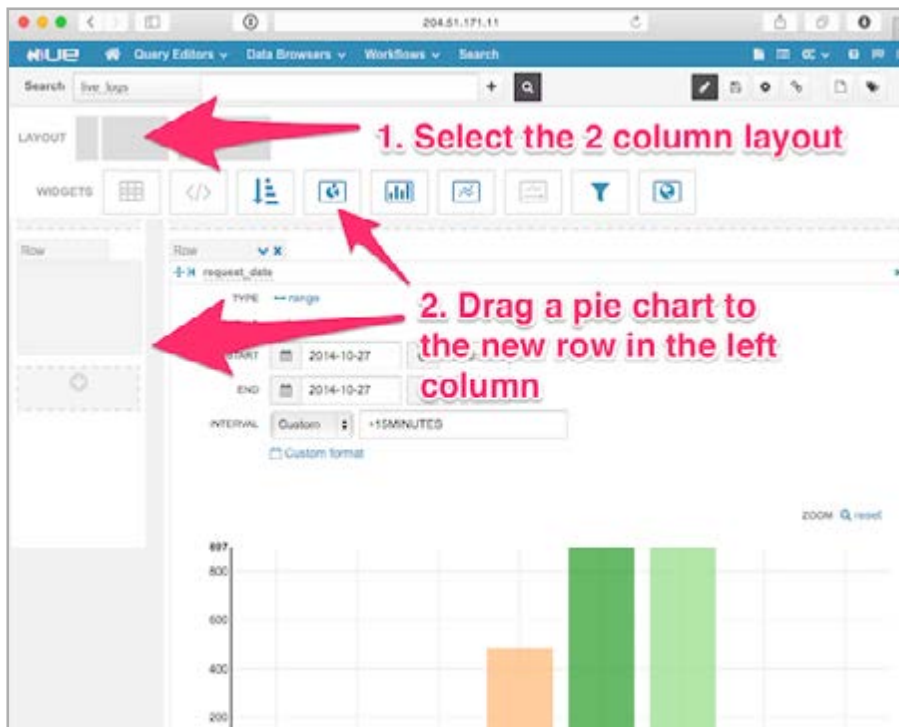


Exercise 4: Building a dashboard

You aren't limited to a single column; you can view this as a two-column display as well. Select the two-column layout from the top left.

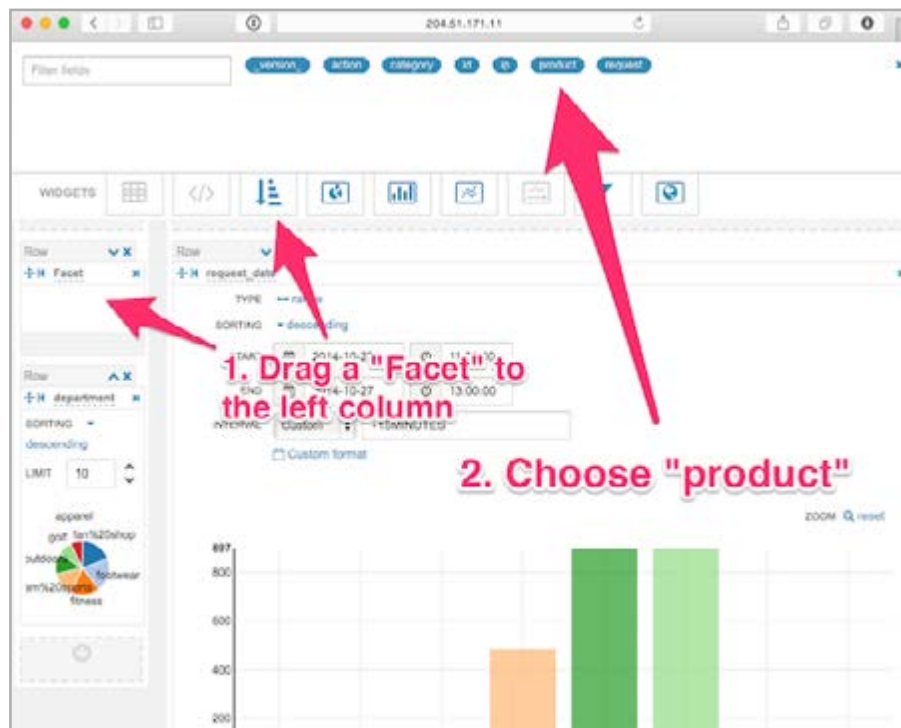
While you're here, let's drag a pie chart to the newly created row in the left column.

This time, let's choose department as the field that we want to group by for our pie chart.

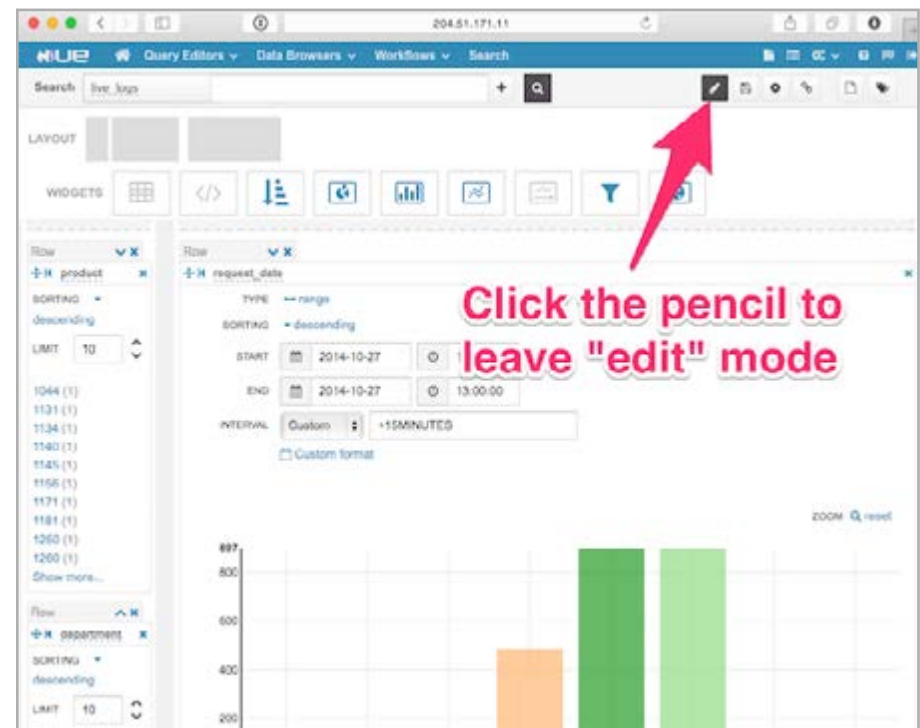


Exercise 4: Building a dashboard

Things are really starting to take shape! Let's add a **Facet filter** to the left-hand side and select product as the facet.

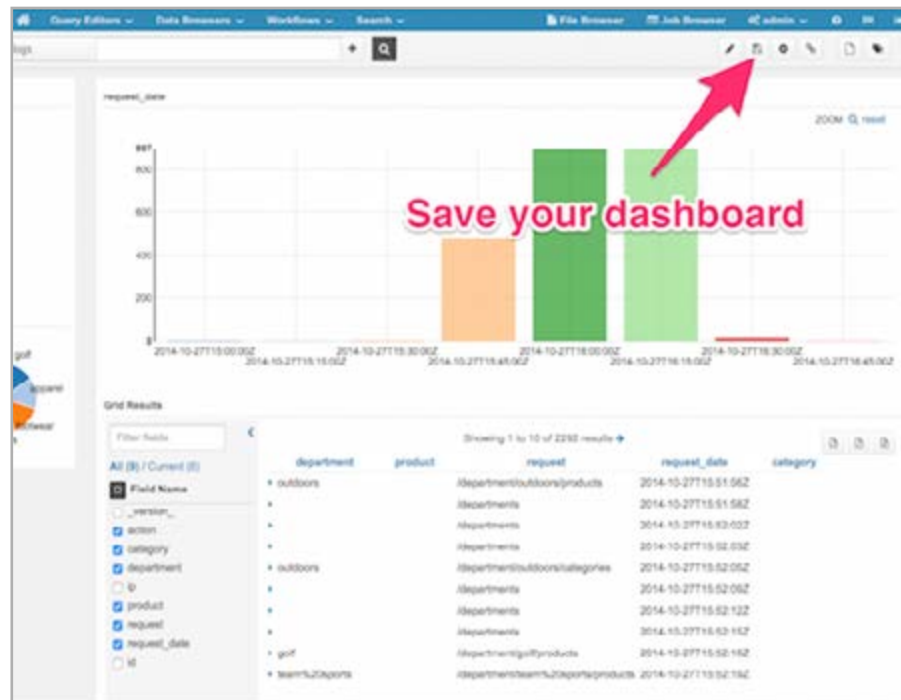


Now that we are satisfied with our changes, let's click on the pencil icon to exit edit mode.



Exercise 4: Building a dashboard

And save our dashboard.



At the [Hue project's blog](#) you can find a wide selection of video tutorials for accomplishing other tasks in Hue. For instance, you can watch a video of a similar Search dashboard to this example being created [here](#).

You may also be interested in more advanced training on the technologies used in this tutorial, and other ways to index data in real-time and query it with Solr in Cloudera's [Search Training course](#).

Data governance and compliance

DataCo has moved into bigger business thanks to the Big Data projects you've contributed to. As more and more users start using the Enterprise Data Hub you built, it starts getting more complicated to manage and trace data and access to data. In addition, as your previous deliveries created such success, the company has decided to build out a full EDH strategy, and as a result lots of sensitive data is headed for your cluster too: credit card transactions, social security data, and other financial records for DataCo.

Your Management: is worried about security controls and ability to audit the access for compliance.

You: need to resolve their concerns and you want to make it easier to manage who does what on your cluster for back-charging purposes too.

Some people are questioning exactly how the decision was made to change the pricing on the website. You realize this is the perfect chance to prove yourself again.

You need to demonstrate that you can:

- Easily show who has queried the data
- Show exactly what has been done with it since it was created
- Enforce policies about how it gets managed in the future

Cloudera Navigator provides a solution to all these problems. If using Cloudera Live, you can find a link in the Govern Your Data section of your Guidance Page, which your welcome email will direct you to. In using the QuickStart VM, the username is 'cloudera' and the password is 'cloudera'.

Exercise 5: Cloudera Navigator

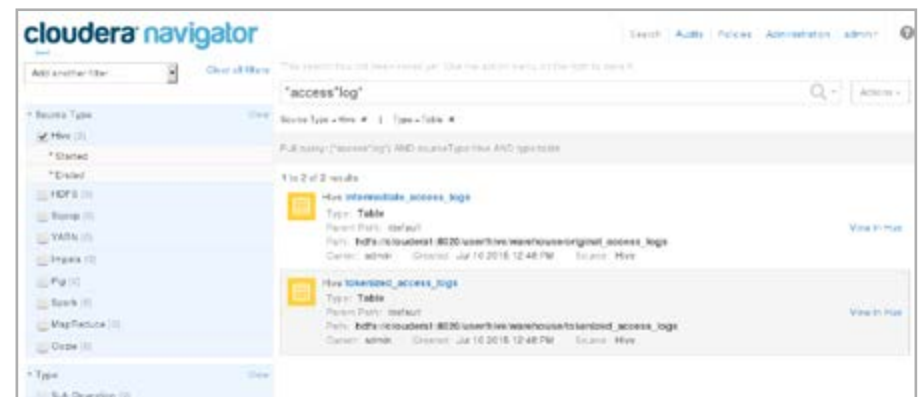
DISCOVERY

The first thing you see when you log into Cloudera Navigator is a search tool. It's an excellent way to find data on your cluster, even when you don't know exactly what you're looking for. Go ahead and click the link to 'explore your data'.

You know that the old web server log data you analyzed was a Hive table, so select 'Hive' under 'Source Type', and 'Table' under 'Type'. You're also pretty sure it had 'access log' in the name, so enter this search query at the top and hit enter:

```
*access*log*
```

When the results appear, you immediately recognize the `tokenized_access_log` table. That must be the one you queried!

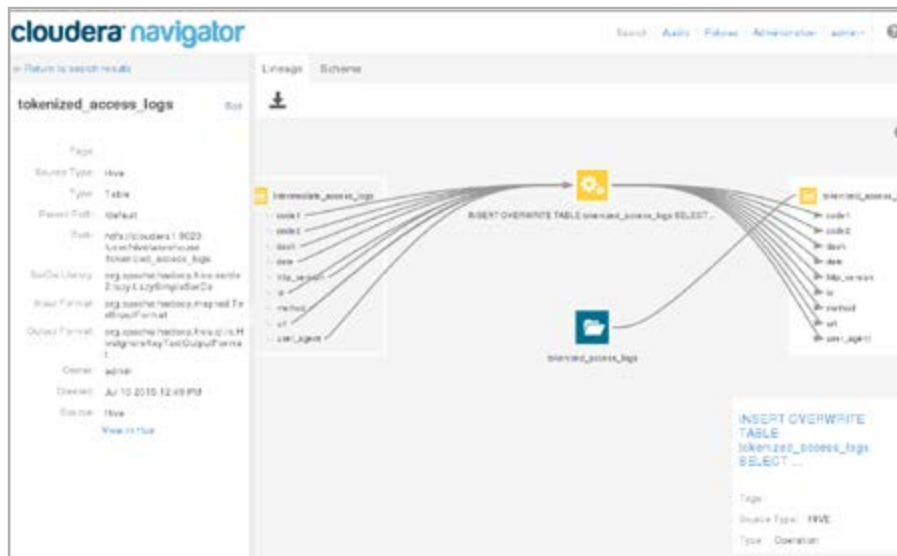


Exercise 5: Cloudera Navigator

LINEAGE

Now that you've found the data you were looking for, click on the table and you'll see a graph of the data's lineage. You'll see the **tokenized_access_logs** table on the right and the underlying file with the same name in HDFS in blue. You'll also see the other Hive table you created from the original file and the query you ran to transform the data between the two. (The different colors represent different source types: yellow data comes from Hive, blue data comes directly from HDFS.)

As you click on the nodes in this graph, more detail will appear. If you click on the **tokenized_access_logs** table and the **intermediate_access_logs** table, you'll see arrows for each individual field running through that query. You can see how quickly you could trace the origin of datasets even in a much busier and more complicated environment!



Exercise 5: Cloudera Navigator

AUDITING

Now you've shown where the data came from, but we still need to show what's been done with it. Go to the **'Audits'** tab, using the link in the top-right corner.

As you can see, there are hundreds of events that have been recorded, each with details of what was done, by whom, and when. Let's narrow down what we're looking for again. Open the "Filters" menu from below the **"Audit Events"** heading.

Find a report...

Audit Events

Recent Deleted Resources

Create New Report +

Audit Events

Filter

Export

Jul 10 2015 12:48 PM - Jul 10 2015 1:48 PM

1 - 60

	Timestamp	Username	IP Address	Service Name	Operation	Resource
+	Jul 10 2015 1:47 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:47 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:45 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:45 PM	admin	60.163.136.240	Navigator	manipula	
+	Jul 10 2015 1:45 PM	admin	60.163.136.240	Navigator	manipula	
+	Jul 10 2015 1:45 PM	admin	60.163.136.240	Navigator	manipula	
+	Jul 10 2015 1:45 PM	code r	10.0.0.114 r	hdfs r	setStatus r	user@code@hadoop
+	Jul 10 2015 1:44 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:43 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:42 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:41 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:41 PM	admin	60.163.136.240	Navigator	manipula	
+	Jul 10 2015 1:40 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:40 PM	admin	60.163.136.240	Navigator	manipula	
+	Jul 10 2015 1:40 PM	admin	60.163.136.240	Navigator	savedSearch	
+	Jul 10 2015 1:39 PM	code	10.0.0.114	hdfs	setStatus	user@code@hadoop
+	Jul 10 2015 1:39 PM	admin	60.163.136.240	Navigator	manipula	
+	Jul 10 2015 1:39 PM	admin	60.163.136.240	Navigator	savedSearch	

The screenshot displays the Cloudera Navigator interface for viewing audit events. At the top, the Cloudera Navigator logo is on the left, and navigation links (Search, Audit, Policies, Administration, admin) are on the right. Below the logo is a search bar. The main heading is "Audit Events". On the right, there's a link "Show As Report" and a date range filter "Jul 10 2015 12:48 PM - Jul 10 2015 1:48 PM".

Below the heading, there are filter sections: "Recent Denied Accounts" and "Create New Report". The main filter area includes dropdowns for "Username" (set to "admin") and "Operation" (set to "QUERY"). There's also an "Add New Filter" button and an "Apply" button.

Below the filters, there's a "Report" section with a table of audit events. The table has the following columns: Timestamp, Username, IP Address, Service Name, Operation, and Resource. The events listed are all queries executed by the "admin" user from IP "10.10.0.114" against the "ingate" service, with a resource of "null".

Timestamp	Username	IP Address	Service Name	Operation	Resource
Jul 10 2015 12:53 PM	admin	10.10.0.114	ingate	QUERY	null
Jul 10 2015 12:53 PM	admin	10.10.0.114	ingate	QUERY	default/normalized_access_logs
Jul 10 2015 12:53 PM	admin	10.10.0.114	ingate	QUERY	null
Jul 10 2015 12:52 PM	admin	10.10.0.114	ingate	QUERY	null
Jul 10 2015 12:52 PM	admin	10.10.0.114	ingate	QUERY	null
Jul 10 2015 12:53 PM	admin	10.0.0.114	hive	QUERY	default/normalized_access_logs
Jul 10 2015 12:50 PM	admin	10.0.0.114	hive	QUERY	default/normalized_access_logs

Exercise 5: Cloudera Navigator

Click the + icon twice to add two new filters. For the first filter, set the property to '**Username**' and fill in 'admin' as the value. For the second filter, set the property to 'Operation' and fill in 'QUERY' as the value. Then click '**Apply**'.

As you click on the individual results, you can see the exact queries that were executed and all related details.

You can also view and create reports based on the results of these searches on the left-hand corner. There's already a report called "Recent Denied Accesses". If you checked that report now, you may see that in the course of this tutorial, some tools have tried to access a directory called '/user/anonymous' that we haven't set up, and that the services don't have permission to create.

The screenshot shows the Cloudera Navigator interface. On the left, there's a sidebar with 'Audit Events' and 'Recent Denied Accesses'. The main area displays a table of audit events. The table has columns: Timestamp, Username, IP Address, Service Name, Operation, and Resource. Below the table, there's a detailed view of a selected event, showing its details like Object Type, Database Name, Query ID, and the actual query text.

Timestamp	Username	IP Address	Service Name	Operation	Resource
Jul 10 2015 12:53 PM	admin	89.16.0.0 114	impala	QUERY	hdfs://
Jul 10 2015 12:53 PM	admin	89.16.0.0 114	impala	QUERY	default:denied_access_logs

Details for the selected event (Timestamp: Jul 10 2015 12:53 PM):

- Object Type: TABLE
- Database Name: default
- Query ID: ST421150a750a3a781159d733a0a6
- Query Text: select count(*) from denied_access_logs where uid like 'hadoop%' group by uid order by count(*) desc
- Result: Query ST421150a750a3a781159d733a0a6 expired due to client inactivity (timeout is 10s)
- Package: HDFS
- Table Name: denied_access_logs
- Query ID: ST421150a750a3a781159d733a0a6

Exercise 5: Cloudera Navigator

POLICIES

It's a relief to be able to audit access to your cluster and see there's no unexpected or unauthorized activity going on. But wouldn't it be even better if you could automatically apply policies to data? Let's open the policies tab in the top-right hand corner and create a policy to make the data we just audited easier to find in the future.



Click the + icon to add a new policy, name your policy "Tag Insecure Data". Check the box to enable the policy, and enter the following as the search query:

```
(permissions:"rwxrwxrwx") AND (sourceType:hdfs) AND  
(type:file OR type:directory) AND (deleted:false)
```


Exercise 5: Cloudera Navigator

This query will detect any files in HDFS that allow anyone to read, write, and execute. It's common for people to set these permissions to make sure everything works, but your organization may want to refine this practice as you move into production or implement more strict practices for some data sets.

To apply this tag on existing data, set the schedule to "Immediate", and check the box "Assign Metadata". Under tags, enter "insecure", and then click **"Add Tag"**. Save the policy.

The screenshot shows the 'Tag Insecure Data' policy configuration page in Cloudera Navigator. The interface includes a search bar at the top left with the text 'Find a policy...' and a 'Create New Policy +' button. The main content area is titled 'Tag Insecure Data' and contains several sections: 'Create Policy' (checked), 'Search Query' (containing the query 'gs://<OR type=directory AND (chmod|ls|cat)'), 'Policy Description' (containing the text 'What does the policy do...'), 'Schedule' (set to 'Immediate'), 'Assign Metadata' (checked), 'Name' (with an 'Expression' button and a text input field), 'Description' (with an 'Expression' button and a text input field), and 'Tags' (containing the tag 'insecure' and an 'Add Tag' button). The 'Save' and 'Cancel' buttons are located at the top right of the main content area.

Exercise 5: Cloudera Navigator

CONCLUSION

You've now experienced how to use Cloudera Navigator for discovery of data and metadata. This powerful tool makes it easy to audit access, trace data lineage, and enforce policies.

With more data, and more data formats available in a multi-tenant environment, data lineage and governance are getting challenging. Cloudera Navigator provides enterprise-grade governance that's built into the foundation of Apache Hadoop.

You can learn more about the various management features provided by Cloudera Manager in the [Cloudera Administrator Training for Apache Hadoop](#).

The End Game

We hope you have enjoyed this basic tutorial, and that you:

- Have a better understanding of some of the popular tools in CDH
- Know how to setup some basic and familiar BI use cases, as well as web log analytics and real-time search
- Can explain to your manager why you deserve a raise!

NEXT STEPS

If you're ready to install Cloudera's platform on your own cluster (on premise or in the public cloud), there are a few options:

- Try the [AWS Quick Start](#) for easy deployment of Cloudera's platform on AWS clusters via your own account (promo credit available)
- Try the [Azure Test Drive](#) for Cloudera Director (3-hour sandbox to provision EDH clusters on Azure)