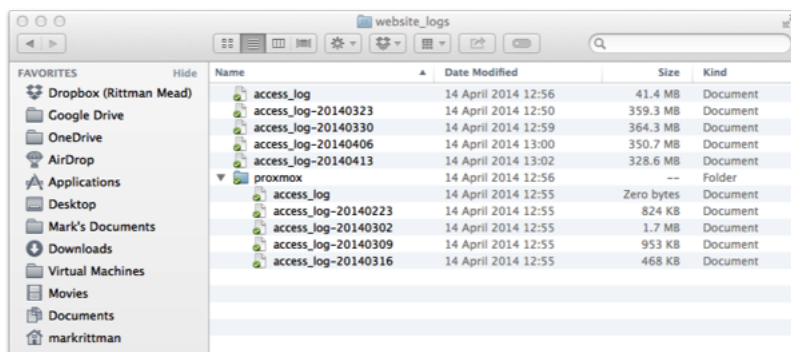


Simple Data Manipulation and Reporting using Hive, Impala and CDH5

24 APRIL 2014

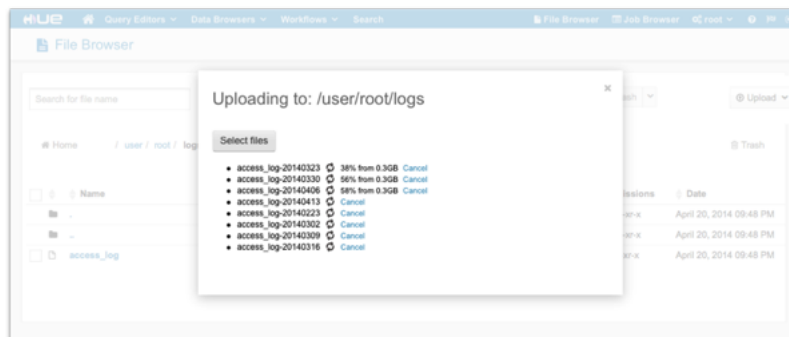
Although I'm pretty clued-up on OBIEE, ODI, Oracle Database and so on, I'm relatively new to the worlds of Hadoop and Big Data, so most evenings and weekends I play around with Hadoop clusters on my home VMWare ESXi rig (<http://www.rittmanmead.com/blog/2013/08/inside-my-home-office-development-lab-vmware-os-x-server/>) and try and get some experience that might then come in useful on customer projects. A few months ago I went through an example of loading-up flight delays data into Cloudera CDH4 and then analysing it using Hive and Impala (<http://www.rittmanmead.com/blog/2014/01/obiee-11-1-1-7-cloudera-hadoop-hiveimpala-part-2-load-data-into-hivecatalog-analyze-using-impala/>), but realistically it's unlikely the data you'll analyse in Hadoop will come in such convenient, tabular form. Something that's more realistic is analysing log files from web servers or other high-volume, semi-structured sources, so I asked Robin to download the most recent set of Apache log files from our website, and I thought I'd have a go at analysing them using Pig and Hive, and maybe the visualise the output using OBIEE (if possible, later on).

As I said, I'm not an expert in Hadoop and the Cloudera platform, so I thought it'd be interesting to describe the journey I went through, and also give some observations from myself on when to use Hive and when to use Pig; when products like Cloudera Impala (<http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/>) could be useful, and also the general state-of-play with the Cloudera Hadoop platform. So the files I started off with were Apache weblog files, with 10 in total and sizes ranging from 350MB to around 2MB.

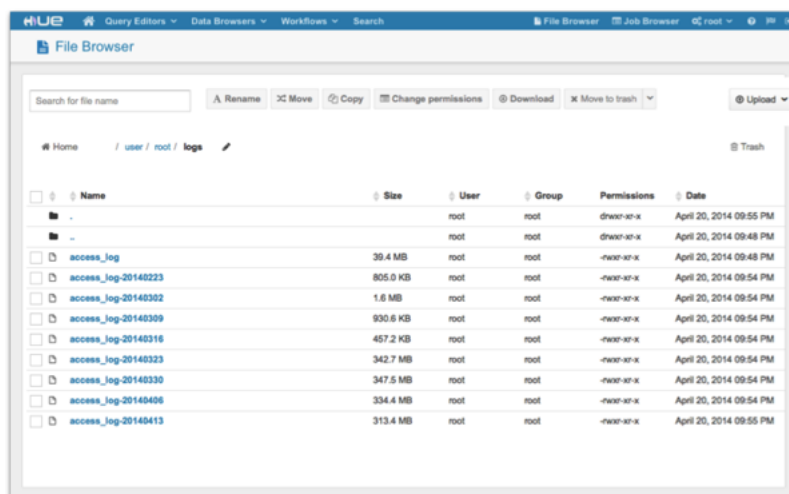


Looking inside one of the log files, they're in the standard Apache log file format (or "combined log format"), where the visitor's IP address is recorded, the date of access, some other information and the page (or resource) they requested:

What I'm looking to do is count the number of visitors on a day, which was the most popular page, what time of day are we most busy, and so on. I've got a Cloudera Hadoop CDH5.0 6-node cluster running on a VMWare ESXi server at home, so the first thing to do is log into Hue, the web-based developer admin tool that comes with CDH5, and upload the files to a directory on HDFS (Hadoop Distributed File System), the Unix-like clustered file system that underpins most of Hadoop.



You can, of course, SFTP the files to one of the Hadoop nodes and use the “hadoop fs” command-line tool to copy the files into HDFS, but for relatively small files like these it's easier to use the web interface to upload them from your workstation. Once I've done that, I can then view the log files in the HDFS directory, just as if they were sitting on a regular Unix filesystem.



At this point though, the files are still “unstructured” - just a single log entry per line - and I'll therefore need to do something before I can count things like number of hits per day, what pages were requested and so on. At this beginners level, there's two main options you can use - Hive, a SQL interface over HDFS that lets you select from, and do set-based transformations with, files of data; or Pig, a more procedural language that lets you manipulate file contents as a series of step-by-step tasks. For someone like myself with a relational data warehousing background, Hive is probably easier to work with but it comes with some quite significant limitations compared to a database like Oracle - we'll see more on this later.

Whilst Hive tables are, at the most simplest level, mapped onto comma or otherwise-delimited files, another neat feature in Hive is that you can use what's called a "SerDe (<https://cwiki.apache.org/confluence/display/Hive/SerDe>)", or "Serializer-Deserializer", to map more complex file structures into regular table columns. In the Hive DDL script below, I use this SerDe feature to have a regular expression parse the log file into columns, with the data source being an entire directory of files, not just a single one:

```
CREATE EXTERNAL TABLE apachelog (
  host STRING,
  identity STRING,
  user STRING,
  time STRING,
  request STRING,
  status STRING,
  size STRING,
  referer STRING,
  agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  "input.regex" = "([^\ ]*) ([^\ ]*) ([^\ ]*) (-|\\[[^\\]]*\\]) ([^\\"]*|\"[^\"]*\\") (-|[0-9]*) (-|[0-9]*)?(?: ([^\\"]*|\"[^\"]*\\") ([^\\ ]*))",
  "output.format.string" = "%1$s %2$s %3$s %4$s %5$s %6$s %7$s %8$s %9$s"
)
STORED AS TEXTFILE
LOCATION '/user/root/logs';
```

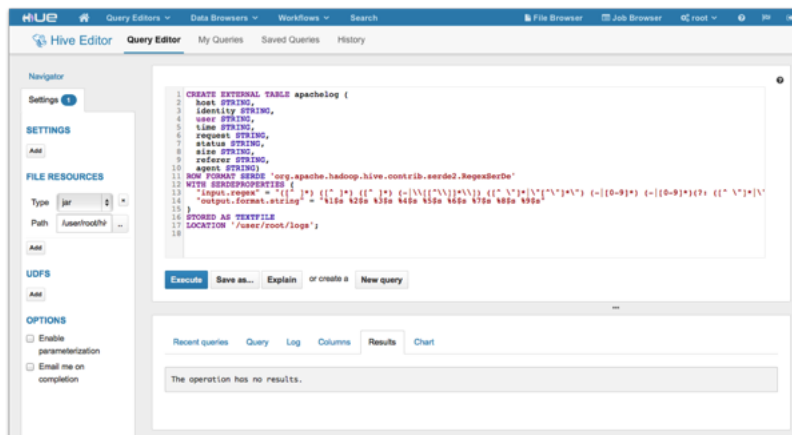
Things to note in the above DDL are:

- **EXTERNAL** table means that the datafile used to populate the Hive table sits somewhere outside Hive's usual `/user/hive/warehouse` directory, in this case in the `/user/root/logs` HDFS directory.
- **ROW FORMAT SERDE** 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe' tells Hive to use the Regular Expressions Serializer-Deserializer to interpret the source file contents, and
- **WITH SERDEPROPERTIES ...** gives the SerDe the regular expression to use, in this case to decode the Apache log format.

Probably the easiest way to run the Hive DDL command to create the table is to use the Hive query editor in Hue, but there's a couple of things you'll need to do before this particular command will work:

1. You'll need to get hold of the JAR file in the Hadoop install that provides this SerDE (hive-contrib-0.12.0-cdh5.0.0.jar) and then copy it to somewhere on your HDFS file system, for example /user/root. In my CDH5 installation, this file was at opt/cloudera/parcels/CDH/lib/hive/lib/, but it'll probably be at /usr/lib/hive/lib if you installed CDH5 using the traditional packages (rather than parcels) route. Also if you're using a version of CDH prior to 5, the filename will be renamed accordingly. This JAR file then needs to be accessible to Hive, and whilst there's various more-permanent ways you can do this, the easiest is to point to the JAR file in an entry in the query editor File Resources section as shown below.

2. Whilst you're there, un-check the "Enable Parameterization" checkbox, otherwise the query editor will interpret the SerDe output string as parameter references.



Once the command has completed, you can click over to the Hive Metastore table browser, and see the columns in the new table.

The screenshot shows the Hive Metastore Manager interface. On the left, there are 'ACTIONS' like 'Import Data', 'Browse Data', 'Drop Table', and 'View File Location'. The main area displays the 'Databases > default > apachelog' table. It has columns: host, identity, user, time, and request. The data shows various IP addresses and HTTP requests.

host	identity	user	time	request
100.43.83.150	-	-	[13/Apr/2014:03:41:11 +0000]	"GET /images/olap_model_10_map_data.jpg HTTP/1.1"
103.255.250.7	-	-	[13/Apr/2014:03:38:56 +0000]	"GET / HTTP/1.0"
103.255.250.7	-	-	[13/Apr/2014:03:39:20 +0000]	"GET / HTTP/1.0"
103.255.250.7	-	-	[13/Apr/2014:03:39:56 +0000]	"GET / HTTP/1.0"
103.255.250.7	-	-	[13/Apr/2014:03:40:20 +0000]	"GET / HTTP/1.0"
103.255.250.7	-	-	[13/Apr/2014:03:40:57 +0000]	"GET / HTTP/1.0"
107.170.72.117	-	-	[13/Apr/2014:03:39:51 +0000]	"GET /feed/ HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:32 +0000]	"GET /2011/11/05/11p-scripting-%E2%80%93-generate-kudrn-and-chang HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:33 +0000]	"GET /wp-content/plugins/wordpress-syntax-highlighter/css/admin/wordpress.min.css HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:34 +0000]	"GET /wp-content/plugins/wordpress-syntax-highlighter/css/admin/wordpress.min.css HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:34 +0000]	"GET /wp-content/plugins/wordpress-syntax-highlighter/css/admin/wordpress.min.css HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:34 +0000]	"GET /wp-content/plugins/wordpress-syntax-highlighter/css/admin/wordpress.min.css HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:34 +0000]	"GET /wp-content/plugins/wordpress-syntax-highlighter/css/admin/wordpress.min.css HTTP/1.1"
123.2.34.215	-	-	[13/Apr/2014:03:40:34 +0000]	"GET /wp-content/plugins/wordpress-syntax-highlighter/css/admin/wordpress.min.css HTTP/1.1"

Behind the scenes, Hive maps its table structure onto all the files in the /user/root/logs HDFS directory, and when I run a SELECT statement against it, for example to do a simple row count, MapReduce mappers, shufflers and sorters are spun-up to return the count of rows to me.

The screenshot shows the Hive Editor interface. On the left, there are 'SETTINGS', 'FILE RESOURCES', 'UDFS', and 'OPTIONS'. The main area shows a query: 'select count(*) from apachelog;'. Below the query, there are buttons: 'Execute', 'Save as...', 'Explain', 'or create a', and 'New query'. At the bottom, there is a 'Results' tab showing the query result: '5341613'.

But in its current form, this table still isn't all that useful - I've just got raw IP addresses for page requesters, and the request date is a format that's not easy to work with. So let's do some further manipulation, creating another table that splits out the request date into year, month, day and time, using Hive's CREATE TABLE AS SELECT command to transform and then load in one command:

```
CREATE TABLE apachelog_date_split_parquet
ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'parquet.hive.DeprecatedParquetInputFormat'
OUTPUTFORMAT 'parquet.hive.DeprecatedParquetOutputFormat'
AS
SELECT host,
       identity,
       user,
       substr(time,9,4) year,
       substr(time,5,3) month,
       substr(time,2,2) day,
       substr(time,14,2) hours,
       substr(time,17,2) secs,
       substr(time,20,2) mins,
       request,
       status,
       size,
       referer,
       agent
FROM   apachelog
;
```

Note the ParquetHive SerDe I'm using in this table's row format definition - Parquet (<http://blog.cloudera.com/blog/2013/03/introducing-parquet-columnar-storage-for-apache-hadoop/>) is a compressed, column-store file format developed by Cloudera originally for Impala (more on that in a moment), that from CDH4.6 is also available for Hive and Pig. By using Parquet, we potentially take advantage of speed and space-saving advantages compared to regular files, so let's use that feature now and see where it takes us. After creating the new Hive table, I can then run a quick query to count web server hits per month:

The screenshot shows the Hive Editor interface. On the left, the 'Navigator' pane shows a tree view of the database structure. The main area displays a query: `select concat(year, '-', month), count(*) from apache_log_date_split_parquet group by concat(year, '-', month)`. Below the query, there are buttons for 'Execute', 'Save as...', 'Explain', 'or create a', and 'New query'. At the bottom, the 'Results' tab is active, showing a table with 4 rows and 2 columns: `concat(year, '-', month)` and `count(*)`.

	concat(year, '-', month)	count(*)
0	NULL	207
1	2014-Apr	2281489
2	2014-Feb	11677
3	2014-Mar	3068240

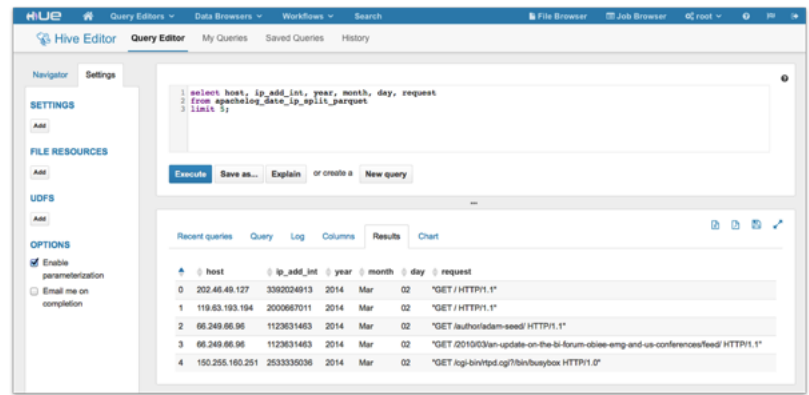
So - getting more useful, but it'd be even nicer if I could map the IP addresses to actual countries, so I can see how many hits came from the UK, how many from the US, and so on. To do this, I'd need to use a lookup service or table to map my IP addresses to countries or cities, and one commonly-used such service is the free GeoIP database provided by MaxMind, where you turn your IP address into an integer via a formula (<http://dev.maxmind.com/geoip/legacy/csv/>), and then do a BETWEEN to locate that IP within ranges defined within the database. How best to do this though?

There's several ways that you can enhance and manipulate data in your Hadoop system like this. One way, and something I plan to look at on this blog later in this series, is to use Pig, potentially with a call-out to Perl or Python to do the lookup on a row-by-row (or tuple-by-tuple) basis - this blog article on the Cloudera site (<https://blog.cloudera.com/blog/2009/06/analyzing-apache-logs-with-pig/>) goes through a nice example. Another way, and again something I plan to cover in this series on the blog, is to use something called "Hadoop Streaming" - the ability within MapReduce to "subcontract" the map and reduce parts of the operation to external programs or scripts, in this case a Python script that again queries the MaxMind database (<http://tech.iheart.com/post/40684867607/hadoop-streaming-logfiles-into-hdfs-hive>) to do the IP-to-country lookup.

But surely it'd be easiest to just calculate the IP address integer and just join my existing Hive table to this GeoIP lookup table, and do it that way? Let's start by trying to do this, first by modifying my final table design to include the IP address integer calculation defined on the MaxMind website:

```
CREATE TABLE apache_log_date_ip_split_parquet
ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT "parquet.hive.DeprecatedParquetInputFormat"
OUTPUTFORMAT "parquet.hive.DeprecatedParquetOutputFormat"
AS
SELECT host,
       (cast(split(host, '\\.')[0] as bigint) * 16777216)
       + (cast(split(host, '\\.')[1] as bigint) * 65535)
       + (cast(split(host, '\\.')[2] as bigint) * 256)
       + (cast(split(host, '\\.')[3] as bigint)) ip_add_int,
       identity,
       user,
       substr(time,9,4) year,
       substr(time,5,3) month,
       substr(time,2,2) day,
       substr(time,14,2) hours,
       substr(time,17,2) secs,
       substr(time,20,2) mins,
       request,
       status,
       size,
       referer,
       agent
FROM
  apache_log
;
```

Now I can query this from the Hive query editor, and I can see the IP address integer calculations that I can then use to match to the GeoIP IP address ranges.



I then upload the IP Address to Countries CSV file from the MaxMind site to HDFS, and define a Hive table over it like this:

```
create external table geo_lookup (
  ip_start      string,
  ip_end        string,
  ip_int_start  int,
  ip_int_end    int,
  country_code  string,
  country_name  string
)
row format DELIMITED
FIELDS TERMINATED BY '|'
LOCATION '/user/root/lookups/geo_ip';
```

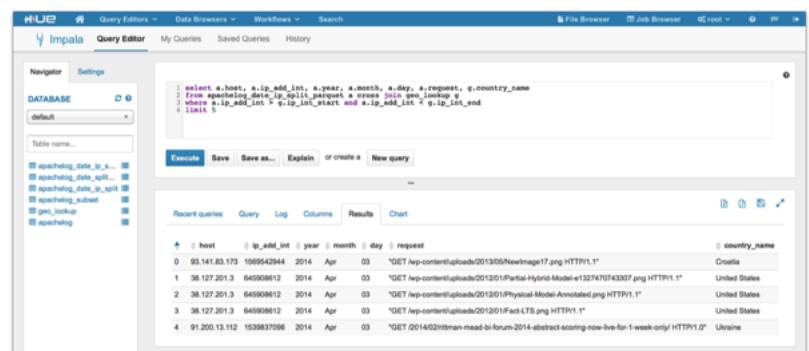
Then I try some variations on the BETWEEN clause, in a SELECT with a join:

```
select a.host, l.country_name
from apachelog_date_ip_split a join geo_lookup l
on (a.ip_add_int > l.ip_int_start) and (a.ip_add_int < l.ip_int_end)
group by a.host, l.country_name;
```

```
select a.host, l.country_name
from apachelog_date_ip_split_parquet a join geo_lookup l
on a.ip_add_int between l.ip_int_start and l.ip_int_end;
```

.. which all fail, because Hive only supports equi-joins

(<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Joins#LanguageManualJoins-JoinSyntax>). One option is to use a Hive UDF (user-defined function) such as this one here (<https://github.com/edwardcapriolo/hive-geoip>) to implement a GeoIP lookup, but something that's probably a bit more promising is to switch over to Impala, which has the ability to do non-equality joins (http://www.cloudera.com/content/cloudera-content/cloudera-docs/Impala/latest/Installing-and-Using-Impala/ciiu_langref_sql.html?scroll=joins_unique_2) through the crossjoin feature (Hive can in fact also use cross-joins, but they're not very efficient). Impala also has the benefit of being much faster for BI-type queries than Hive, and it's also designed to work with Parquet, so let's switch over to the Impala query editor, run the "invalidate metadata" command to re-sync its table view with Hive's table metastore, and then try the join in there:






Not bad. Of course this is all fairly simple stuff, and we're still largely working with relational-style set-based transformations. In the next two posts in the series though I want get a bit more deep into Hadoop-style transformations - first by using a feature called "Hadoop Streaming" to process data on its way into Hadoop, done in parallel, by calling out to Python and Perl scripts; and then take a look at Pig, the more "procedural" alternative to Hive - with the objective being to enhance this current dataset to bring in details of the pages being requested, filter out the non-page requests, and do some work with authors, tag and clickstream analysis.

We were unable to load Disqus. If you are a moderator please see our [troubleshooting guide](#).

Mark Rittman (/blog/author/mark-rittman/)

Read more posts (/blog/author/mark-rittman/) by this author.

Share this Post

 (<https://twitter.com/intent/tweet?text=Simple%20Data%20Manipulation%20and%20Reporting%20using%20Hive%2C%20Impala%20and%20data-manipulation-and-reporting-using-hive-impala-and-cdh5/>)  (<https://www.facebook.com/sharer/sharer.php?u=https://www.rittmanmead.com/blog/2014/04/simple-data-manipulation-and-reporting-using-hive-impala-and-cdh5/>)  (<https://plus.google.com/share?url=https://www.rittmanmead.com/blog/2014/04/simple-data-manipulation-and-reporting-using-hive-impala-and-cdh5/>)

TECHNICAL INSIGHTS (/BLOG/TAG/TECHNICAL)

BUSINESS INSIGHTS (/BLOG/TAG/BUSINESS-INSIGHTS)

RITTMAN MEAD LIFE (/BLOG/TAG/RITTMAN-MEAD-LIFE)

Recent Posts

- OA Summit 2020: OA Roadmap Summary (/blog/2020/06/oa-summit-2020-oracle-analytics-roadmap-summary/)
- Data Virtualization: What is it About? (/blog/2020/06/data-virtualization-what-is-it/)
- Getting Smart View to work with OAC (/blog/2020/05/getting-smart-view-to-work-with-oac/)
- Oracle Analytics: Everything you always wanted to know (But were afraid to ask) (/blog/2020/02/oracle-analytics-everything-you-always-wanted-to-know-but-were-afraid-to-ask/)
- Oracle Data Science - Accelerated Data Science SDK Configuration (/blog/2020/02/accelerated-data-science-sdk-configuration/)

Sign Up for Our Newsletter

SUBSCRIBE

(/blog/2014/04/previewing-bi-forum-atlanta-2014/)

READ THIS NEXT

YOU MIGHT ENJOY

(/blog/2014/04/previewing-three-oracle-data-visualization-sessions-at-)

Preview of the Rittman Mead BI Forum in Atlanta

Mark has done a great job of previewing the upcoming content for both BI Forums, the one running locally...

Previewing Three Oracle Data Visualization Sessions at the Atlanta US BI Forum 2014

Many of the sessions at the UK and US Rittman Mead BI Forum 2014 events in May focus on...

About Us

Rittman Mead is a data and analytics company who specialise in data visualisation, predictive analytics, enterprise reporting and data engineering.

<http://www.rittmanmead.com/feed/> <http://twitter.com/rittmanmead>

<https://www.facebook.com/rittmanmead/> <http://www.linkedin.com/company/rittman-mead/>

Contact Us

Rittman Mead Consulting Ltd.

Platform, Hove Town Hall
Tisbury Road,
Brighton, BN3 3BQ
United Kingdom

Tel: (Phone) +44 1273 053956

Email: (Email) info@rittmanmead.com (mailto:info@rittmanmead.com)