Documentation

Welcome to the Apache Drill Documentation. If you do not see the navigation bar for the documentation, click the menu icon on the left at the top of any page.



If you're new to Drill, try out one of these resources:

* [Drill Introduction](http://drill.apache.org/docs/drill-introduction)
* [Tutorials](http://drill.apache.org/docs/tutorials-introduction)
* [Getting Started](javascript:%20void(0);)
* [Architecture](javascript:%20void(0);)
* [Tutorials](javascript:%20void(0);)
* [Install Drill](javascript:%20void(0);)
* [Configure Drill](javascript:%20void(0);)
* [Connect a Data Source](javascript:%20void(0);)
* [ODBC/JDBC Interfaces](javascript:%20void(0);)
* [Query Data](javascript:%20void(0);)
* [Performance Tuning](javascript:%20void(0);)
* [Log and Debug](javascript:%20void(0);)
* [SQL Reference](javascript:%20void(0);)
* [Data Sources and File Formats](javascript:%20void(0);)
* [Develop Custom Functions](javascript:%20void(0);)
* [Getting Started](javascript:%20void(0);)

# Drill Introduction

Jul 31, 2017

Drill is an Apache open-source SQL query engine for Big Data exploration(大数据检索的sql查询引擎). Drill is designed from the ground up（以下） to support high-performance analysis on the semi-structured（对半结构话的数据的高效分析） and rapidly evolving data coming from modern Big Data applications（以及从当今大数据应用中迅速的获取数据）, while still providing the familiarity and ecosystem of ANSI SQL（同时仍然提供我们熟悉的SQL生态系统这一行业标准的查询语言）, the industry-standard query language. Drill provides plug-and-play integration with existing Apache Hive and Apache HBase deployments（drill提供了与apache的hbase和hive等的集成插件）.

## What's New in Apache Drill 1.11

Drill 1.11 provides the following new features and improvements:

* Cryptography-related functions. (DRILL-5634)
* Spill to disk for the hash aggregate operator. (DRILL-5457)
* Format plugin support for PCAP files. (DRILL-5432)
* Ability to change the HDFS block Size for Parquet files. (DRILL-5379)
* Ability to store query profiles in memory. (DRILL-5481)
* Configurable CTAS directory and file permissions option. (DRILL-5391)
* Support for network encryption. (DRILL-4335)
* Relative paths stored in the metadata file. (DRILL-3867)
* Support for ANSI\_QUOTES. (DRILL-3510)

## What's New in Apache Drill 1.10

Drill 1.10 provides the following new features and improvements:

* Support for the [CREATE TEMPORARY TABLE AS (CTTAS)](http://drill.apache.org/docs/create-temporary-table-as-cttas/) command.
* A [JDBC connection option](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-jdbc-url-format-for-a-direct-drillbit-connection) that improves fault tolerance when connecting directly to a Drill node from a client.
* The [Web Console](http://drill.apache.org/docs/identifying-multiple-drill-versions-in-a-cluster) displays the Drill version and additional query profile statistics.
* Drill implicitly interprets the [INT96](http://drill.apache.org/docs/parquet-format/#about-int96-support/) timestamp data type in Parquet files.
* Support for Kerberos authentication between the client and drillbit.

## What's New in Apache Drill 1.9

Drill 1.9 provides the following new features:

* Asynchronous Parquet reader
* Parquet filter pushdown
* Dynamic UDF support
* HTTPD format plugin

## What's New in Apache Drill 1.8

Drill 1.8 provides the following new features and changes:

* Metadata cache pruning
* IF EXISTS parameter with the DROP TABLE and DROP VIEW commands
* DESCRIBE SCHEMA command
* Multi-byte delimiter support
* New parameters for filter selectivity estimates
* Changes to the configuration and launch scripts - See [Configuration and Launch Script Changes](http://drill.apache.org/docs/apache-drill-1-8-0-release-notes/#configuration-and-launch-script-changes)

## What's New in Apache Drill 1.7

Drill 1.7 provides the following new features:

* Monitoring via JMX
* Hive CHAR data type support
* HBase 1.x support

## What's New in Apache Drill 1.6

Drill 1.6 provides the following new features:

* Inbound impersonation
* Additional custom window frames

## What's New in Apache Drill 1.5

Drill 1.5 provides the following new features:

* Authentication and security for the Web interface and REST API
* Experimental query support for Apache Kudu (incubating)
* An improved memory allocator
* Configurable caching for Hive metadata

## What's New in Apache Drill 1.4

Drill 1.4 introduces the following improvements:

* [select with options](http://drill.apache.org/docs/plugin-configuration-basics/#using-the-formats-attributes-as-table-function-parameters) that you use in queries to change storage plugin settings
* Improved behavior when parsing CSV file header names
* A variable to set non-pretty, such as compact, printing of JSON
* Better drillbit.log files that include query text

Drill 1.4 fixes an error that occurred when you query a Hive table using the HBaseStorageHandler ([DRILL-3739](https://issues.apache.org/jira/browse/DRILL-3739)). To successfully query a Hive table using the HBaseStorageHandler, you need to configure the Hive storage plugin as described in the [Hive storage plugin documentation](http://drill.apache.org/docs/hive-storage-plugin/#connect-drill-to-the-hive-remote-metastore).

## What's New in Apache Drill 1.3

This releases fix issues and add a number of enhancements, including the following ones:

* [Enhanced Amazon S3 support](http://drill.apache.org/docs/s3-storage-plugin/)
* Hetrogeneous types  
  Support for columns that evolve from one data type to another over time.
* [Text file headers](http://drill.apache.org/docs/text-files-csv-tsv-psv/#using-a-header-in-a-file)
* [Sequence files support](http://drill.apache.org/docs/querying-sequence-files/)
* Enhancements related to querying Hive tables, MongoDB collections, and Avro files

## What's New in Apache Drill 1.2

This release of Drill fixes [many issues](http://drill.apache.org/docs/apache-drill-1-2-0-release-notes/) and introduces a number of enhancements, including the following ones:

* Support for JDBC data sources, such as MySQL, through a [new JDBC Storage plugin](https://issues.apache.org/jira/browse/DRILL-3180)
* Improvements in the Drill JDBC driver including inclusion of Javadocs and better application dependency compatibility
* Enhancements to Avro file formats  
  + [Support for complex data types](https://issues.apache.org/jira/browse/DRILL-3565), such as UNION and MAP
  + [Optimized Avro file processing](https://issues.apache.org/jira/browse/DRILL-3720) (block-wise)
* Partition pruning improvements
* A number of new [SQL window functions](http://drill.apache.org/docs/sql-window-functions)  
  + NTILE
  + LAG and LEAD
  + FIRST\_VALUE and LAST\_VALUE
* [HTTPS support](http://drill.apache.org/docs/configuring-web-console-and-rest-api-security/) for Web Console operations
* Performance improvements for [querying HBase](http://drill.apache.org/docs/querying-hbase/#querying-big-endian-encoded-data), which includes leveraging [ordered byte encoding](http://drill.apache.org/docs/querying-hbase/#leveraging-hbase-ordered-byte-encoding)
* [Optimized reads](http://drill.apache.org/docs/querying-hive/#optimizing-reads-of-parquet-backed-tables) of Parquet-backed, Hive tables
* Read support for the [Parquet INT96 type](http://drill.apache.org/docs/parquet-format/#about-int96-support) and a new TIMESTAMP\_IMPALA type used with the [CONVERT\_FROM](http://drill.apache.org/docs/supported-data-types/#data-types-for-convert_to-and-convert_from-functions) function decodes a timestamp from Hive or Impala.
* [Parquet metadata caching](http://drill.apache.org/docs/optimizing-parquet-metadata-reading/) to improve query performance on a large number of files
* DROP TABLE command
* Improved correlated subqueries
* Union Distinct
* Improved LIMIT processing

## What's New in Apache Drill 1.1

Many enhancements in Apache Drill 1.1 include the following key features:

* [SQL window functions](http://drill.apache.org/docs/sql-window-functions)
* [Partitioning data](http://drill.apache.org/docs/drill-introduction/) using the new [PARTITION BY](http://drill.apache.org/docs/partition-by-clause) clause in the CTAS command
* [Delegated Hive impersonation](http://drill.apache.org/docs/configuring-user-impersonation-with-hive-authorization/)
* Support for UNION and UNION ALL and better optimized plans that include UNION.

## What's New in Apache Drill 1.0

Apache Drill 1.0 offers the following new features:

* Many performance planning and execution [improvements](http://drill.apache.org/docs/performance-tuning-introduction/).
* Updated [Drill shell](http://drill.apache.org/docs/configuring-the-drill-shell) now formats query results.
* [Query audit logging](http://drill.apache.org/docs/getting-query-information/) for getting the query history on a Drillbit.
* Improved connection handling.
* New Errors tab in the Query Profiles UI that facilitates troubleshooting and distributed storing of profiles.
* Support for a new storage plugin input format: [Avro](http://avro.apache.org/docs/current/spec.html)

In this release, Drill disables the DECIMAL data type, including casting to DECIMAL and reading DECIMAL types from Parquet and Hive. You can [enable the DECIMAL type](http://drill.apache.org/docs/drill-introduction/docs/supported-data-types/#enabling-the-decimal-type), but this is not recommended.

## Apache Drill Key Features

Key features of Apache Drill are:

* Low-latency SQL queries 低延迟的sql查询
* Dynamic queries on self-describing data in files (such as JSON, Parquet, text) and HBase tables, without requiring metadata definitions in the Hive metastore. 对于基于文件的自描述数据和hbase的表的动态查询，不需要在hive的metastore中的metadata定义
* ANSI SQL
* Nested data support 支持嵌套
* Integration with Apache Hive (queries on Hive tables and views, support for all Hive file formats and Hive UDFs) 与apache的hive集成（如查询hive的表和视图，支持所有的hive文件格式和hive的UDFs）
* BI/SQL tool integration using standard JDBC/ODBC drivers 集成了标准的JDBC和ODBC驱动的商业智能/sql 工具

## Quick Links

If you've never used Drill, visit these links to get a jump start:

* [Drill in 10 Minutes](http://drill.apache.org/docs/drill-in-10-minutes/)
* [Query Files](http://drill.apache.org/docs/querying-a-file-system)
* [Query HBase](http://drill.apache.org/docs/querying-hbase)
* [SQL Support](http://drill.apache.org/docs/sql-reference-introduction/)
* [Drill Tutorials](http://drill.apache.org/docs/tutorials-introduction)

# Why Drill

## Top 10 Reasons to Use Drill

## 1. Get started in minutes

It takes just a few minutes to get started with Drill. Untar（解压） the Drill software on your Linux, Mac, or Windows laptop and run a query on a local file. No need to set up any infrastructure or to define schemas. Just point to the data, such as data in a file, directory, HBase table, and drill（例如文件数据、文件目录、hbase的表和drill等）.

$ tar -xvf apache-drill-<version>.tar.gz

$ <install directory>/bin/drill-embedded

0: jdbc:drill:zk=local> SELECT \* FROM cp.`employee.json` LIMIT 5;

+--------------+----------------------------+---------------------+---------------+--------------+----------------------------+-----------+----------------+-------------+------------------------+----------+----------------+----------------------+-----------------+---------+-----------------------+

| employee\_id | full\_name | first\_name | last\_name | position\_id | position\_title | store\_id | department\_id | birth\_date | hire\_date | salary | supervisor\_id | education\_level | marital\_status | gender | management\_role |

+--------------+----------------------------+---------------------+---------------+--------------+----------------------------+-----------+----------------+-------------+------------------------+----------+----------------+----------------------+-----------------+---------+-----------------------+

| 1 | Sheri Nowmer | Sheri | Nowmer | 1 | President | 0 | 1 | 1961-08-26 | 1994-12-01 00:00:00.0 | 80000.0 | 0 | Graduate Degree | S | F | Senior Management |

| 2 | Derrick Whelply | Derrick | Whelply | 2 | VP Country Manager | 0 | 1 | 1915-07-03 | 1994-12-01 00:00:00.0 | 40000.0 | 1 | Graduate Degree | M | M | Senior Management |

| 4 | Michael Spence | Michael | Spence | 2 | VP Country Manager | 0 | 1 | 1969-06-20 | 1998-01-01 00:00:00.0 | 40000.0 | 1 | Graduate Degree | S | M | Senior Management |

| 5 | Maya Gutierrez | Maya | Gutierrez | 2 | VP Country Manager | 0 | 1 | 1951-05-10 | 1998-01-01 00:00:00.0 | 35000.0 | 1 | Bachelors Degree | M | F | Senior Management |

## 2. Schema-free JSON model

Drill is the world's first and only distributed SQL engine that doesn't require schemas（不需要scheme，即不需要数据结构）. It shares the same schema-free JSON model as MongoDB and Elasticsearch（它与mongodb和elasticsearch 一样使用无scheme的json）. No need to define and maintain schemas or transform data (ETL)（不需要保存表schem或者对数据进行转换）. Drill automatically understands the structure of the data（drill能够自动理解数据结构）.

## 3. Query complex, semi-structured data in-situ

Using Drill's schema-free JSON model, you can query complex, semi-structured data in situ. No need to flatten or transform the data prior to or during query execution（不需要事先对数据进行转换）. Drill also provides intuitive extensions to SQL to work with nested data. Here's a simple query on a JSON file demonstrating how to access nested elements and arrays（嵌套的元素和数组）:

SELECT \* FROM (SELECT t.trans\_id,

t.trans\_info.prod\_id[0] AS prod\_id,

t.trans\_info.purch\_flag AS purchased

FROM `clicks/clicks.json` t) sq

WHERE sq.prod\_id BETWEEN 700 AND 750 AND

sq.purchased = 'true'

ORDER BY sq.prod\_id;

## 4. Real SQL -- not "SQL-like"

Drill supports the standard SQL:2003 syntax. No need to learn a new "SQL-like" language or struggle with a semi-functional BI tool. Drill supports many data types including DATE, INTERVAL, TIMESTAMP, and VARCHAR, as well as complex query constructs such as correlated sub-queries and joins in WHERE clauses. Here is an example of a TPC-H standard query that runs in Drill:

### TPC-H query 4

SELECT o.o\_orderpriority, COUNT(\*) AS order\_count

FROM orders o

WHERE o.o\_orderdate >= DATE '1996-10-01'

AND o.o\_orderdate < DATE '1996-10-01' + INTERVAL '3' month

AND EXISTS(

SELECT \* FROM lineitem l

WHERE l.l\_orderkey = o.o\_orderkey

AND l.l\_commitdate < l.l\_receiptdate

)

GROUP BY o.o\_orderpriority

ORDER BY o.o\_orderpriority;

## 5. Leverage standard BI tools

Drill works with standard BI tools. You can use your existing tools, such as Tableau, MicroStrategy, QlikView and Excel.

## 6. Interactive queries on Hive tables

Apache Drill lets you leverage your investments in Hive. You can run interactive queries with Drill on your Hive tables and access all Hive input/output formats (including custom SerDes). You can join tables associated with different Hive metastores（可以连接不同的hive元数据存储）, and you can join a Hive table with an HBase table or a directory of log files. Here's a simple query in Drill on a Hive table:

SELECT `month`, state, sum(order\_total) AS sales

FROM hive.orders

GROUP BY `month`, state

ORDER BY 3 DESC LIMIT 5;

## 7. Access multiple data sources

Drill is extensible（可扩展的）. You can connect Drill out-of-the-box to file systems (local or distributed, such as S3 and HDFS), HBase and Hive. You can implement a storage plugin to make Drill work with any other data source. Drill can combine data from multiple data sources on the fly in a single query, with no centralized metadata definitions（无需中心metadata的预先定义，drill即可连接多个数据源）. Here's a query that combines data from a Hive table, an HBase table (view) and a JSON file:

SELECT custview.membership, sum(orders.order\_total) AS sales

FROM hive.orders, custview, dfs.`clicks/clicks.json` c

WHERE orders.cust\_id = custview.cust\_id AND orders.cust\_id = c.user\_info.cust\_id

GROUP BY custview.membership

ORDER BY 2;

## 8. User-Defined Functions (UDFs) for Drill and Hive

Drill exposes a simple, high-performance Java API to build [custom user-defined functions](http://drill.apache.org/docs/develop-custom-functions/) (UDFs) for adding your own business logic to Drill（drill为用户定义自己的业务逻辑的构建提供了以非常简单的java api）. Drill also supports Hive UDFs(drill也支持hive的用户自定义函数，无需修改即可迁移到drill中). If you have already built UDFs in Hive, you can reuse them with Drill with no modifications.

## 9. High performance

Drill is designed from the ground up for high throughput and low latency（高产出低延迟）. It doesn't use a general purpose execution engine like MapReduce, Tez or Spark. As a result, Drill is flexible (schema-free JSON model) and performant（弹性的和表现好的）. Drill's optimizer leverages rule- and cost-based techniques, as well as data locality and operator push-down, which is the capability to push down query fragments into the back-end data sources. Drill also provides a columnar and vectorized execution engine（也提供了针对列和垂直列的执行引擎）, resulting in higher memory and CPU efficiency（导致了高效的内存和CPU效率）.

## 10. Scales from a single laptop to a 1000-node cluster

Drill is available as a simple download you can run on your laptop. When you're ready to analyze larger datasets, deploy Drill on your Hadoop cluster (up to 1000 commodity servers). Drill leverages the aggregate memory in the cluster to execute queries using an optimistic pipelined model, and automatically spills to disk when the working set doesn't fit in memory（使用了一个优化的管道模型，利用集群中的很大内存来执行查询，并且当内存不足时自动将数据写到磁盘）.

* [Architecture](javascript:%20void(0);)

# Architecture Introduction

Apache Drill is a low latency distributed query engine for large-scale datasets, including structured and semi-structured/nested data. Inspired by Google’s Dremel, Drill is designed to scale to several thousands of nodes and query petabytes of data at interactive speeds that BI/Analytics environments require.

Drill is also useful for short, interactive ad-hoc queries on large-scale data sets. Drill is capable of querying nested data in formats like JSON and Parquet and performing dynamic schema discovery. Drill does not require a centralized metadata repository.

## High-Level Architecture

Drill includes a distributed execution environment, purpose built for large- scale data processing. At the core of Apache Drill is the "Drillbit" service, which is responsible for accepting requests from the client, processing the queries, and returning results to the client.

A Drillbit service can be installed and run on all of the required nodes in a Hadoop cluster to form a distributed cluster environment. When a Drillbit runs on each data node in the cluster, Drill can maximize data locality during query execution without moving data over the network or between nodes. Drill uses ZooKeeper to maintain cluster membership and health-check information.

Though Drill works in a Hadoop cluster environment, Drill is not tied to Hadoop and can run in any distributed cluster environment. The only pre-requisite for Drill is Zookeeper.

See [Drill Query Execution](http://drill.apache.org/docs/drill-query-execution/).

## Drill Clients

You can access Drill through the following interfaces:

* [Drill shell](http://drill.apache.org/docs/configuring-the-drill-shell/)
* [Drill Web Console](http://drill.apache.org/docs/monitoring-and-canceling-queries-in-the-drill-web-console)
* [ODBC/JDBC](http://drill.apache.org/docs/interfaces-introduction/#using-odbc-to-access-apache-drill-from-bi-tools)
* C++ API

### Dynamic schema discovery

Drill does not require schema or type specification for data in order to start the query execution process. Drill starts data processing in record-batches and discovers the schema during processing. Self-describing data formats such as Parquet, JSON, AVRO, and NoSQL databases have schema specified as part of the data itself, which Drill leverages dynamically at query time. Because the schema can change over the course of a Drill query, many Drill operators are designed to reconfigure themselves when schemas change.

### Flexible data model

Drill allows access to nested data attributes, as if they were SQL columns, and provides intuitive extensions to easily operate on them. From an architectural point of view, Drill provides a flexible hierarchical columnar data model that can represent complex, highly dynamic and evolving data models. Relational data in Drill is treated as a special or simplified case of complex/multi-structured data.

### No centralized metadata

Drill does not have a centralized metadata requirement. You do not need to create and manage tables and views in a metadata repository, or rely on a database administrator group for such a function. Drill metadata is derived through the storage plugins that correspond to data sources. Storage plugins provide a spectrum of metadata ranging from full metadata (Hive), partial metadata (HBase), or no central metadata (files). De-centralized metadata means that Drill is NOT tied to a single Hive repository. You can query multiple Hive repositories at once and then combine the data with information from HBase tables or with a file in a distributed file system. You can also use SQL DDL statements to create metadata within Drill, which gets organized just like a traditional database. Drill metadata is accessible through the ANSI standard INFORMATION\_SCHEMA database.

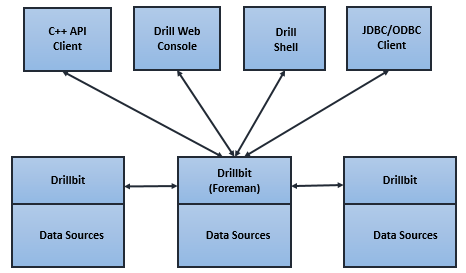
### Extensibility

Drill provides an extensible architecture at all layers, including the storage plugin, query, query optimization/execution, and client API layers. You can customize any layer for the specific needs of an organization or you can extend the layer to a broader array of use cases. Drill uses classpath scanning to find and load plugins, and to add additional storage plugins, functions, and operators with minimal configuration.

# Drill Query Execution

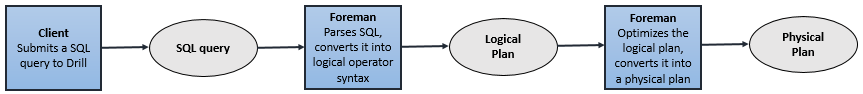
When you submit a Drill query, a client or an application sends the query in the form of an SQL statement to a Drillbit in the Drill cluster. A Drillbit is the process running on each active Drill node that coordinates, plans, and executes queries, as well as distributes query work across the cluster to maximize data locality.

The following image represents the communication between clients, applications, and Drillbits:

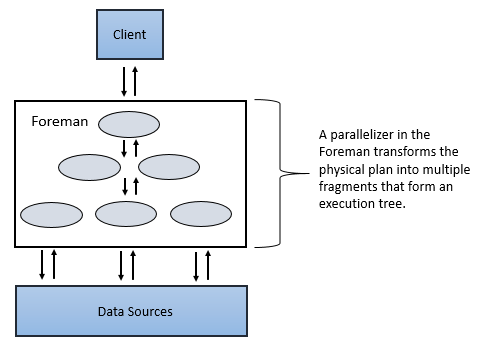


The Drillbit that receives the query from a client or application becomes the Foreman for the query and drives the entire query. A parser in the Foreman parses the SQL, applying custom rules to convert specific SQL operators into a specific logical operator syntax that Drill understands. This collection of logical operators forms a logical plan. The logical plan describes the work required to generate the query results and defines which data sources and operations to apply.

The Foreman sends the logical plan into a cost-based optimizer to optimize the order of SQL operators in a statement and read the logical plan. The optimizer applies various types of rules to rearrange operators and functions into an optimal plan. The optimizer converts the logical plan into a physical plan that describes how to execute the query.



A parallelizer in the Foreman transforms the physical plan into multiple phases, called major and minor fragments. These fragments create a multi-level execution tree that rewrites the query and executes it in parallel against the configured data sources, sending the results back to the client or application.



## Major Fragments

A major fragment is a concept that represents a phase of the query execution. A phase can consist of one or multiple operations that Drill must perform to execute the query. Drill assigns each major fragment a MajorFragmentID.

For example, to perform a hash aggregation of two files, Drill may create a plan with two major phases (major fragments) where the first phase is dedicated to scanning the two files and the second phase is dedicated to the aggregation of the data.



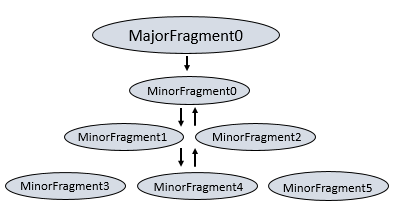
Drill uses an exchange operator to separate major fragments. An exchange is a change in data location and/or parallelization of the physical plan. An exchange is composed of a sender and a receiver to allow data to move between nodes.

Major fragments do not actually perform any query tasks. Each major fragment is divided into one or multiple minor fragments (discussed in the next section) that actually execute the operations required to complete the query and return results back to the client.

You can work with major fragments within the physical plan by capturing a JSON representation of the plan in a file, manually modifying it, and then submitting it back to Drill using the SUBMIT PLAN command. You can also view major fragments in the query profile, which is visible in the Drill Web Console. See [EXPLAIN](http://drill.apache.org/docs/explain/)and [Query Profiles](http://drill.apache.org/docs/query-profiles/) for more information.

## Minor Fragments

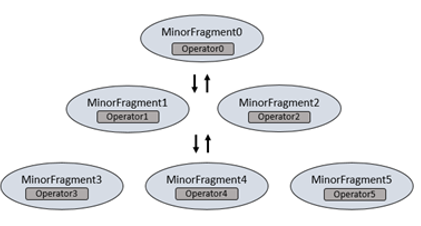
Each major fragment is parallelized into minor fragments. A minor fragment is a logical unit of work that runs inside a thread. A logical unit of work in Drill is also referred to as a slice. The execution plan that Drill creates is composed of minor fragments. Drill assigns each minor fragment a MinorFragmentID.



The parallelizer in the Foreman creates one or more minor fragments from a major fragment at execution time, by breaking a major fragment into as many minor fragments as it can usefully run at the same time on the cluster.

Drill executes each minor fragment in its own thread as quickly as possible based on its upstream data requirements. Drill schedules the minor fragments on nodes with data locality. Otherwise, Drill schedules them in a round-robin fashion on the existing, available Drillbits.

Minor fragments contain one or more relational operators. An operator performs a relational operation, such as scan, filter, join, or group by. Each operator has a particular operator type and an OperatorID. Each OperatorID defines its relationship within the minor fragment to which it belongs. See [Physical Operators](http://drill.apache.org/docs/physical-operators/).



For example, when performing a hash aggregation of two files, Drill breaks the first phase dedicated to scanning into two minor fragments. Each minor fragment contains scan operators that scan the files. Drill breaks the second phase dedicated to aggregation into four minor fragments. Each of the four minor fragments contain hash aggregate operators that perform the hash aggregation operations on the data.

You cannot modify the number of minor fragments within the execution plan. However, you can view the query profile in the Drill Web Console and modify some configuration options that change the behavior of minor fragments, such as the maximum number of slices. See [Configuration Options](http://drill.apache.org/docs/configuration-options-introduction/).

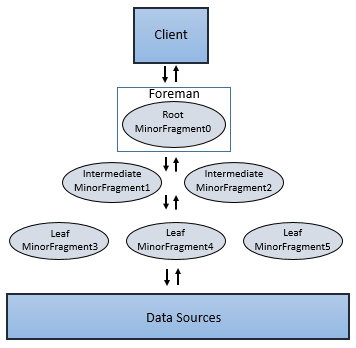
### Execution of Minor Fragments

Minor fragments can run as root, intermediate, or leaf fragments. An execution tree contains only one root fragment. The coordinates of the execution tree are numbered from the root, with the root being zero. Data flows downstream from the leaf fragments to the root fragment.

The root fragment runs in the Foreman and receives incoming queries, reads metadata from tables, rewrites the queries and routes them to the next level in the serving tree. The other fragments become intermediate or leaf fragments.

Intermediate fragments start work when data is available or fed to them from other fragments. They perform operations on the data and then send the data downstream. They also pass the aggregated results to the root fragment, which performs further aggregation and provides the query results to the client or application.

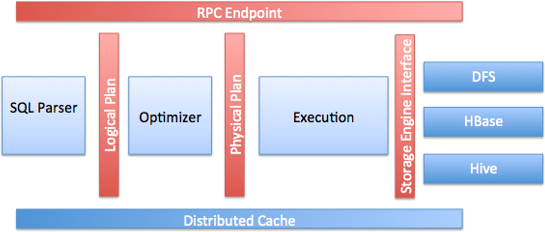
The leaf fragments scan tables in parallel and communicate with the storage layer or access data on local disk. The leaf fragments pass partial results to the intermediate fragments, which perform parallel operations on intermediate results.



Drill only plans queries that have concurrent running fragments. For example, if 20 available slices exist in the cluster, Drill plans a query that runs no more than 20 minor fragments in a particular major fragment. Drill is optimistic and assumes that it can complete all of the work in parallel. All minor fragments for a particular major fragment start at the same time based on their upstream data dependency.

Core Modules

The following image represents components within each Drillbit:



The following list describes the key components of a Drillbit:

* **RPC endpoint**: Drill exposes a low overhead protobuf-based RPC protocol to communicate with the clients. Additionally, C++ and Java API layers are also available for client applications to interact with Drill. Clients can communicate with a specific Drillbit directly or go through a ZooKeeper quorum to discover the available Drillbits before submitting queries. It is recommended that the clients always go through ZooKeeper to shield clients from the intricacies of cluster management, such as the addition or removal of nodes.
* **SQL parser**: Drill uses [Calcite](https://calcite.incubator.apache.org/), the open source SQL parser framework, to parse incoming queries. The output of the parser component is a language agnostic, computer-friendly logical plan that represents the query.
* **Storage plugin interface**: Drill serves as a query layer on top of several data sources. Storage plugins in Drill represent the abstractions that Drill uses to interact with the data sources. Storage plugins provide Drill with the following information:
  + Metadata available in the source
  + Interfaces for Drill to read from and write to data sources
  + Location of data and a set of optimization rules to help with efficient and fast execution of Drill queries on a specific data source

In the context of Hadoop, Drill provides storage plugins for distributed files and HBase. Drill also integrates with Hive using a storage plugin.

When users query files and HBase with Drill, they can do it directly or go through Hive if they have metadata defined there. Drill integration with Hive is only for metadata. Drill does not invoke the Hive execution engine for any requests.

# Performance

Drill is designed from the ground up for high performance on large datasets. The following core elements of Drill processing are responsible for Drill's performance:

**Distributed engine**

Drill provides a powerful distributed execution engine for processing queries. Users can submit requests to any node in the cluster. You can add new nodes to the cluster to scale for larger volumes of data to support more users or improve performance.

**Columnar execution**

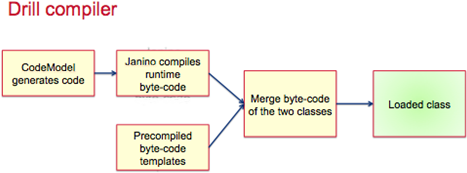
Drill optimizes for both columnar storage and execution by using an in-memory data model that is hierarchical and columnar. When working with data stored in columnar formats such as Parquet, Drill avoids disk access for columns that are not involved in a query. Drill's execution layer also performs SQL processing directly on columnar data without row materialization. The combination of optimizations for columnar storage and direct columnar execution significantly lowers memory footprints and provides faster execution of BI and analytic types of workloads.

**Vectorization**

Rather than operating on single values from a single table record at one time, vectorization in Drill allows the CPU to operate on vectors, referred to as a record batches. A record batch has arrays of values from many different records. The technical basis for efficiency of vectorized processing is modern chip technology with deep-pipelined CPU designs. Keeping all pipelines full to achieve efficiency near peak performance is impossible to achieve in traditional database engines, primarily due to code complexity.

**Runtime compilation**

Runtime compilation enables faster execution than interpreted execution. Drill generates highly efficient custom code for every single query. The following image shows the Drill compilation/code generation process:



**Optimistic and pipelined query execution**

Using an optimistic execution model to process queries, Drill assumes that failures are infrequent within the short span of a query. Drill does not spend time creating boundaries or checkpoints to minimize recovery time. In the instance of a single query failure, the query is rerun. Drill execution uses a pipeline model where all tasks are scheduled at once. The query execution happens in- memory as much as possible to move data through task pipelines, persisting to disk only if there is memory overflow.

* [Tutorials](javascript:%20void(0);)

Tutorials Introduction

If you've never used Drill, use these tutorials to download, install, and start working with Drill. The tutorials include step-by-step procedures for the following tasks:

* [Drill in 10 Minutes](http://drill.apache.org/docs/drill-in-10-minutes)  
  Download, install, and start Drill in embedded mode (single-node cluster mode).
* [Analyzing the Yelp Academic Dataset](http://drill.apache.org/docs/analyzing-the-yelp-academic-dataset)  
  Download and install Drill in embedded mode and use SQL examples to analyze Yelp data.
* [Learn Drill with the MapR Sandbox](http://drill.apache.org/docs/about-the-mapr-sandbox)  
  Explore data using a Hadoop environment pre-configured with Drill.
* [Analyzing Highly Dynamic Datasets](http://drill.apache.org/docs/analyzing-highly-dynamic-datasets)  
  Learn how to handle dynamic data without changing a schema or going through an ETL phase.
* [Analyzing Social Media](http://drill.apache.org/docs/analyzing-social-media)  
  Analyze Twitter data in its native JSON format using Drill.
* [Analyzing Data Using Window Functions](http://drill.apache.org/docs/analyzing-data-using-window-functions) Learn how to use analytic/window functions.
* [Tableau Examples](http://drill.apache.org/docs/tableau-examples)  
  Access Hive tables using Drill and Tableau.
* [Using MicroStrategy Analytics with Apache Drill](http://drill.apache.org/docs/using-microstrategy-analytics-with-apache-drill/)  
  Use the Drill ODBC driver from MapR to analyze data and generate a report using Drill from the MicroStrategy UI.
* [Using Tibco Spotfire Desktop with Drill](http://drill.apache.org/docs/using-tibco-spotfire-desktop-with-drill/)  
  Use Apache Drill to query complex data structures from Tibco Spotfire Desktop.
* [Configuring Tibco Spotfire Server with Drill](http://drill.apache.org/docs/configuring-tibco-spotfire-server-with-drill)  
  Integrate Tibco Spotfire Server with Apache Drill and explore multiple data formats on Hadoop.
* [Using Apache Drill with Tableau 9 Desktop](http://drill.apache.org/docs/using-apache-drill-with-tableau-9-desktop)  
  Connect Tableau 9 Desktop to Apache Drill, explore multiple data formats on Hadoop, and access semi-structured data.
* [Using Apache Drill with Tableau 9 Server](http://drill.apache.org/docs/using-apache-drill-with-tableau-9-server)  
  Connect Tableau 9 Server to Apache Drill, explore multiple data formats on Hadoop, access semi-structured data, and share Tableau visualizations with others.
* [Using Drill to Analyze Amazon Spot Prices](https://github.com/vicenteg/spot-price-history/#drill-workshop---amazon-spot-prices)  
  Use a Drill workshop on github to create views of JSON and Parquet data.
* [Running Drill Queries on S3 Data](http://drill.apache.org/blog/2014/12/09/running-sql-queries-on-amazon-s3/)  
  Step through querying files using Drill and Amazon Simple Storage Service (S3).

# Drill in 10 Minutes

Jul 31, 2017

## Objective

Use Apache Drill to query sample data in 10 minutes. For simplicity, you run Drill in embedded mode rather than distributed mode to try out Drill without having to perform any setup tasks.

## Installation Overview

You can install Drill to run in embedded mode on a machine running Linux, Mac OS X, or Windows. For information about installing Drill to run in distributed mode, see [Installing Drill in Distributed Mode](http://drill.apache.org/docs/installing-drill-in-distributed-mode).

This installation procedure includes how to download the Apache Drill archive file and extract the contents to a directory on your machine. The Apache Drill archive contains sample JSON and Parquet files that you can query immediately.

After installing Drill, you start the Drill shell. The Drill shell is a pure-Java console-based utility for connecting to relational databases and executing SQL commands. Drill follows the SQL:2011 standard with [extensions](http://drill.apache.org/docs/sql-extensions/) for nested data formats and other capabilities.

## Embedded Mode Installation Prerequisites

You need to meet the following prerequisites to run Drill:

* Linux, Mac OS X, and Windows: Oracle JDK [version 7](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html) or [version 8](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) if running Drill 1.6 or later
* Windows only:  
  + A JAVA\_HOME environment variable set up that points to the JDK installation
  + A PATH environment variable that includes a pointer to the bin directory of the JDK installation
  + A third-party utility for unzipping a .tar.gz file

### Java Installation Prerequisite Check

Run the following command in a terminal (Linux and Mac OS X) or Command Prompt (Windows) to verify that Java 7 or 8 is the version in effect:

java -version

The output looks something like this:

java version "1.7.0\_79"

Java(TM) SE Runtime Environment (build 1.7.0\_7965-b15)

Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)

## Install Drill on Linux or Mac OS X

Complete the following steps to install Drill:

1. In a terminal window, change to the directory where you want to install Drill.
2. Download the latest version of Apache Drill [here](http://apache.mirrors.hoobly.com/drill/drill-1.11.0/apache-drill-1.11.0.tar.gz) or from the [Apache Drill mirror site](http://www.apache.org/dyn/closer.cgi/drill/drill-1.11.0/apache-drill-1.11.0.tar.gz) with the command appropriate for your system:  
   * wget http://apache.mirrors.hoobly.com/drill/drill-1.11.0/apache-drill-1.11.0.tar.gz
   * curl -o apache-drill-1.11.0.tar.gz http://apache.mirrors.hoobly.com/drill/drill-1.11.0/apache-drill-1.11.0.tar.gz
3. Copy the downloaded file to the directory where you want to install Drill.
4. Extract the contents of the Drill .tar.gz file. Use sudo if necessary:  
   tar -xvzf <.tar.gz file name>

The extraction process creates an installation directory containing the Drill software.

At this point, you can start Drill.

## Start Drill on Linux and Mac OS X

Start Drill in embedded mode using the drill-embedded command:

1. Navigate to the Drill installation directory. For example:

cd apache-drill-<version>

1. Issue the following command to launch Drill in embedded mode:

bin/drill-embedded

The message of the day followed by the [0: jdbc:drill:zk=local> prompt](http://drill.apache.org/docs/starting-drill-on-linux-and-mac-os-x/#about-the-drill-prompt) appears.

At this point, you can [submit queries](http://drill.apache.org/docs/drill-in-10-minutes/#query-sample-data) to Drill.

### Install Drill on Windows

You can install Drill on Windows. First, set the JAVA\_HOME environment variable, and then install Drill. Complete the following steps to install Drill:

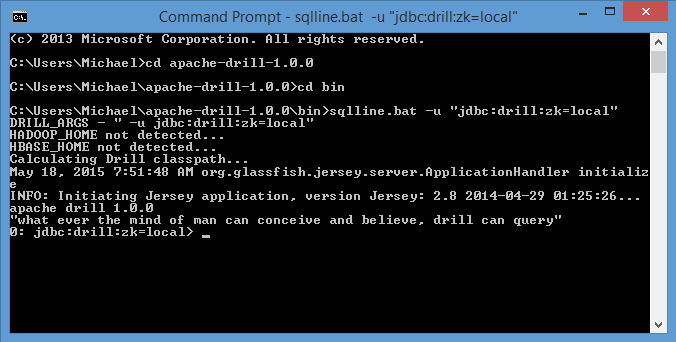
1. Download the latest version of Apache Drill [here](http://apache.mirrors.hoobly.com/drill/drill-1.11.0/apache-drill-1.11.0.tar.gz) or from the [Apache Drill mirror site](http://www.apache.org/dyn/closer.cgi/drill/drill-1.11.0/apache-drill-1.11.0.tar.gz).
2. Move the apache-drill-<version>.tar.gz file to a directory where you want to install Drill.
3. Unzip the TAR.GZ file using a third-party tool. If the tool you use does not unzip the TAR file as well as the TAR.GZ file, unzip the apache-drill-<version>.tar to extract the Drill software. The extraction process creates the installation directory named apache-drill- containing the Drill software.

At this point, you can start Drill.

## Start Drill on Windows

Start Drill by running the sqlline.bat file and typing a connection string, as shown in the following procedure. The zk=local in the connection string means the local node is the ZooKeeper node:

Start the Drill shell using the **sqlline command**. Complete the following steps to launch the Drill shell:

1. Open Command Prompt.
2. Open the apache-drill- folder.
3. Go to the bin directory. For example:  
   cd bin
4. Type the following command on the command line: sqlline.bat -u "jdbc:drill:zk=local"

The zk=local means the local node is the ZooKeeper node. At this point, you can [run queries](http://drill.apache.org/docs/drill-in-10-minutes/#query-sample-data).

## Stopping Drill

Issue the following command when you want to exit the Drill shell:

!quit

## Query Sample Data

At the root of the Drill installation, a sample-data directory includes JSON and Parquet files that you can query. The default dfs storage plugin configuration represents the local file system on your machine when you install Drill in embedded mode. For more information about storage plugin configuration, refer to [Storage Plugin Registration](http://drill.apache.org/docs/connect-a-data-source-introduction).

Use SQL to query the sample JSON and Parquet files in the sample-data directory on your local file system.

### Querying a JSON File

A sample JSON file, [employee.json](http://drill.apache.org/docs/querying-json-files/), contains fictitious employee data. To view the data in the employee.json file, submit the following SQL query to Drill, using the [cp (classpath) storage plugin](http://drill.apache.org/docs/storage-plugin-registration/) configuration to point to the file.

0: jdbc:drill:zk=local> SELECT \* FROM cp.`employee.json` LIMIT 3;

The query output is:

+--------------+------------------+-------------+------------+--------------+---------------------+-----------+----------------+-------------+------------------------+----------+----------------+------------------+-----------------+---------+--------------------+

| employee\_id | full\_name | first\_name | last\_name | position\_id | position\_title | store\_id | department\_id | birth\_date | hire\_date | salary | supervisor\_id | education\_level | marital\_status | gender | management\_role |

+--------------+------------------+-------------+------------+--------------+---------------------+-----------+----------------+-------------+------------------------+----------+----------------+------------------+-----------------+---------+--------------------+

| 1 | Sheri Nowmer | Sheri | Nowmer | 1 | President | 0 | 1 | 1961-08-26 | 1994-12-01 00:00:00.0 | 80000.0 | 0 | Graduate Degree | S | F | Senior Management |

| 2 | Derrick Whelply | Derrick | Whelply | 2 | VP Country Manager | 0 | 1 | 1915-07-03 | 1994-12-01 00:00:00.0 | 40000.0 | 1 | Graduate Degree | M | M | Senior Management |

| 4 | Michael Spence | Michael | Spence | 2 | VP Country Manager | 0 | 1 | 1969-06-20 | 1998-01-01 00:00:00.0 | 40000.0 | 1 | Graduate Degree | S | M | Senior Management |

+--------------+------------------+-------------+------------+--------------+---------------------+-----------+----------------+-------------+------------------------+----------+----------------+------------------+-----------------+---------+--------------------+

3 rows selected (0.827 seconds)

### Querying a Parquet File

Query the region.parquet and nation.parquet files in the sample-data directory on your local file system.

#### Region File

If you followed the Apache Drill in 10 Minutes instructions to install Drill in embedded mode, the path to the parquet file varies between operating systems.

To view the data in the region.parquet file, use the actual path to your Drill installation to construct this query:

SELECT \* FROM dfs.`<path-to-installation>/apache-drill-<version>/sample-data/region.parquet`;

The query returns the following results:

+--------------+--------------+-----------------------+

| R\_REGIONKEY | R\_NAME | R\_COMMENT |

+--------------+--------------+-----------------------+

| 0 | AFRICA | lar deposits. blithe |

| 1 | AMERICA | hs use ironic, even |

| 2 | ASIA | ges. thinly even pin |

| 3 | EUROPE | ly final courts cajo |

| 4 | MIDDLE EAST | uickly special accou |

+--------------+--------------+-----------------------+

5 rows selected (0.409 seconds)

#### Nation File

The path to the parquet file varies between operating systems. Use the actual path to your Drill installation to construct this query:

SELECT \* FROM dfs.`<path-to-installation>/apache-drill-<version>/sample-data/nation.parquet`;

The query returns the following results:

SELECT \* FROM dfs.`Users/drilluser/apache-drill/sample-data/nation.parquet`;

+--------------+-----------------+--------------+-----------------------+

| N\_NATIONKEY | N\_NAME | N\_REGIONKEY | N\_COMMENT |

+--------------+-----------------+--------------+-----------------------+

| 0 | ALGERIA | 0 | haggle. carefully f |

| 1 | ARGENTINA | 1 | al foxes promise sly |

| 2 | BRAZIL | 1 | y alongside of the p |

| 3 | CANADA | 1 | eas hang ironic, sil |

| 4 | EGYPT | 4 | y above the carefull |

| 5 | ETHIOPIA | 0 | ven packages wake qu |

| 6 | FRANCE | 3 | refully final reques |

| 7 | GERMANY | 3 | l platelets. regular |

| 8 | INDIA | 2 | ss excuses cajole sl |

| 9 | INDONESIA | 2 | slyly express asymp |

| 10 | IRAN | 4 | efully alongside of |

| 11 | IRAQ | 4 | nic deposits boost a |

| 12 | JAPAN | 2 | ously. final, expres |

| 13 | JORDAN | 4 | ic deposits are blit |

| 14 | KENYA | 0 | pending excuses hag |

| 15 | MOROCCO | 0 | rns. blithely bold c |

| 16 | MOZAMBIQUE | 0 | s. ironic, unusual a |

| 17 | PERU | 1 | platelets. blithely |

| 18 | CHINA | 2 | c dependencies. furi |

| 19 | ROMANIA | 3 | ular asymptotes are |

| 20 | SAUDI ARABIA | 4 | ts. silent requests |

| 21 | VIETNAM | 2 | hely enticingly expr |

| 22 | RUSSIA | 3 | requests against th |

| 23 | UNITED KINGDOM | 3 | eans boost carefully |

| 24 | UNITED STATES | 1 | y final packages. sl |

+--------------+-----------------+--------------+-----------------------+

25 rows selected (0.101 seconds)

## Summary

Apache Drill supports nested data, schema-less execution, and decentralized metadata. At this point, you know how to create a simple query on a JSON or Parquet file.

## Next Steps

Now that you have an idea about what Drill can do, you might want to:

* [Install Drill on a cluster.](http://drill.apache.org/docs/installing-drill-on-the-cluster)
* [Configure storage plugins to connect Drill to your data sources](http://drill.apache.org/docs/connect-a-data-source-introduction).
* Query [Hive](http://drill.apache.org/docs/querying-hive) and [HBase](http://drill.apache.org/docs/hbase-storage-plugin) data.
* [Query Complex Data](http://drill.apache.org/docs/querying-complex-data)
* [Query Plain Text Files](http://drill.apache.org/docs/querying-plain-text-files)

## More Information

For more information about Apache Drill, explore the [Apache Drill web site](http://drill.apache.org/).

# Analyzing the Yelp Academic Dataset

Apache Drill is one of the fastest growing open source projects, with the community making rapid progress with monthly releases. The key difference is Drill’s agility and flexibility. Along with meeting the table stakes for SQL-on-Hadoop, which is to achieve low latency performance at scale, Drill allows users to analyze the data without any ETL or up-front schema definitions. The data can be in any file format such as text, JSON, or Parquet. Data can have simple types such as strings, integers, dates, or more complex multi-structured data, such as nested maps and arrays. Data can exist in any file system, local or distributed, such as HDFS or S3. Drill, has a “no schema” approach, which enables you to get value from your data in just a few minutes.

Let’s quickly walk through the steps required to install Drill and run it against the Yelp data set. The publicly available data set used for this example is downloadable from [Yelp](http://www.yelp.com/dataset_challenge) (business reviews) and is in JSON format.

## Installing and Starting Drill

### Download Apache Drill onto your local machine

To experiment with Drill locally, follow the installation instructions in [Drill in 10 Minutes](http://drill.apache.org/docs/drill-in-10-minutes/).

Alternatively, you can [install Drill in distributed mode](http://drill.apache.org/docs/installing-drill-in-distributed-mode) if you want to scale your environment.

Let’s try out some SQL examples to understand how Drill makes the raw data analysis extremely easy.

**NOTE**

You need to substitute your local path to the Yelp data set in the angle-bracketed portion of the FROM clause of each query you run.

## Querying Data with Drill

### 1. View the contents of the Yelp business data

0: jdbc:drill:zk=local> !set maxwidth 10000

0: jdbc:drill:zk=local> select \* from

dfs.`<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json`

limit 1;

+------------------------+----------------------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------+--------------------------------+---------+--------------+-------------------+-------------+-------+-------+-----------+--------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+---------------+

| business\_id | full\_address | hours | open | categories | city | review\_count | name | longitude | state | stars | latitude | attributes | type | neighborhoods |

+------------------------+----------------------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------+--------------------------------+---------+--------------+-------------------+-------------+-------+-------+-----------+--------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+---------------+

| vcNAWiLM4dR7D2nwwJ7nCA | 4840 E Indian School Rd Ste 101, Phoenix, AZ 85018 | fill in{"Tuesday":{"close":"17:00","open":"08:00"},"Friday":{"close":"17:00","open":"08:00"},"Monday":{"close":"17:00","open":"08:00"},"Wednesday":{"close":"17:00","open":"08:00"},"Thursday":{"close":"17:00","open":"08:00"},"Sunday":{},"Saturday":{}} | true | ["Doctors","Health & Medical"] | Phoenix | 7 | Eric Goldberg, MD | -111.983758 | AZ | 3.5 | 33.499313 | {"By Appointment Only":true,"Good For":{},"Ambience":{},"Parking":{},"Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} | business | [] |

+------------------------+----------------------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------+--------------------------------+---------+--------------+-------------------+-------------+-------+-------+-----------+--------------------------------------------------------------------------------------------------------------------------------------------------------------+----------+---------------+

**NOTE**

This document aligns Drill output for example purposes. Drill output is not aligned in this case.

You can directly query self-describing files such as JSON, Parquet, and text. There is no need to create metadata definitions in the Hive metastore.

### 2. Explore the business data set further

#### Total reviews in the data set

0: jdbc:drill:zk=local> select sum(review\_count) as totalreviews

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json`;

+--------------+

| totalreviews |

+--------------+

| 1236445 |

+--------------+

#### Top states and cities in total number of reviews

0: jdbc:drill:zk=local> select state, city, count(\*) totalreviews

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json`

group by state, city order by count(\*) desc limit 10;

+------------+------------+--------------+

| state | city | totalreviews |

+------------+------------+--------------+

| NV | Las Vegas | 12021 |

| AZ | Phoenix | 7499 |

| AZ | Scottsdale | 3605 |

| EDH | Edinburgh | 2804 |

| AZ | Mesa | 2041 |

| AZ | Tempe | 2025 |

| NV | Henderson | 1914 |

| AZ | Chandler | 1637 |

| WI | Madison | 1630 |

| AZ | Glendale | 1196 |

+------------+------------+--------------+

#### Average number of reviews per business star rating

0: jdbc:drill:zk=local> select stars,trunc(avg(review\_count)) reviewsavg

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json`

group by stars order by stars desc;

+------------+------------+

| stars | reviewsavg |

+------------+------------+

| 5.0 | 8.0 |

| 4.5 | 28.0 |

| 4.0 | 48.0 |

| 3.5 | 35.0 |

| 3.0 | 26.0 |

| 2.5 | 16.0 |

| 2.0 | 11.0 |

| 1.5 | 9.0 |

| 1.0 | 4.0 |

+------------+------------+

#### Top businesses with high review counts (> 1000)

0: jdbc:drill:zk=local> select name, state, city, `review\_count` from

dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json`

where review\_count > 1000 order by `review\_count` desc limit 10;

+-------------------------------+-------------+------------+---------------+

| name | state | city | review\_count |

+-------------------------------+-------------+------------+---------------+

| Mon Ami Gabi | NV | Las Vegas | 4084 |

| Earl of Sandwich | NV | Las Vegas | 3655 |

| Wicked Spoon | NV | Las Vegas | 3408 |

| The Buffet | NV | Las Vegas | 2791 |

| Serendipity 3 | NV | Las Vegas | 2682 |

| Bouchon | NV | Las Vegas | 2419 |

| The Buffet at Bellagio | NV | Las Vegas | 2404 |

| Bacchanal Buffet | NV | Las Vegas | 2369 |

| The Cosmopolitan of Las Vegas | NV | Las Vegas | 2253 |

| Aria Hotel & Casino | NV | Las Vegas | 2224 |

+-------------------------------+-------------+----------------------------+

#### Saturday open and close times for a few businesses

0: jdbc:drill:zk=local> select b.name, b.hours.Saturday.`open`,

b.hours.Saturday.`close`

from

dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json`

b limit 10;

+----------------------------+------------+------------+

| name | EXPR$1 | EXPR$2 |

+----------------------------+------------+------------+

| Eric Goldberg, MD | 08:00 | 17:00 |

| Pine Cone Restaurant | null | null |

| Deforest Family Restaurant | 06:00 | 22:00 |

| Culver's | 10:30 | 22:00 |

| Chang Jiang Chinese Kitchen| 11:00 | 22:00 |

| Charter Communications | null | null |

| Air Quality Systems | null | null |

| McFarland Public Library | 09:00 | 20:00 |

| Green Lantern Restaurant | 06:00 | 02:00 |

| Spartan Animal Hospital | 07:30 | 18:00 |

+----------------------------+------------+------------+

Note how Drill can traverse and refer through multiple levels of nesting.

### 3. Get the amenities of each business in the data set

Note that the attributes column in the Yelp business data set has a different element for every row, representing that businesses can have separate amenities. Drill makes it easy to quickly access data sets with changing schemas.

First, change Drill to work in all text mode (so we can take a look at all of the data).

0: jdbc:drill:zk=local> alter system set `store.json.all\_text\_mode` = true;

+------------+-----------------------------------+

| ok | summary |

+------------+-----------------------------------+

| true | store.json.all\_text\_mode updated. |

+------------+-----------------------------------+

Then, query the attribute’s data.

0: jdbc:drill:zk=local> select attributes from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` limit 10;

+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| attributes |

+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| {"By Appointment Only":"true","Good For":{},"Ambience":{},"Parking":{},"Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} |

| {"Take-out":"true","Good For":{"dessert":"false","latenight":"false","lunch":"true","dinner":"false","breakfast":"false","brunch":"false"},"Caters":"false","Noise Level":"averag |

| {"Take-out":"true","Good For":{"dessert":"false","latenight":"false","lunch":"false","dinner":"false","breakfast":"false","brunch":"true"},"Caters":"false","Noise Level":"quiet" |

| {"Take-out":"true","Good For":{},"Takes Reservations":"false","Delivery":"false","Ambience":{},"Parking":{"garage":"false","street":"false","validated":"false","lot":"true","val |

| {"Take-out":"true","Good For":{},"Ambience":{},"Parking":{},"Has TV":"false","Outdoor Seating":"false","Attire":"casual","Music":{},"Hair Types Specialized In":{},"Payment Types |

| {"Good For":{},"Ambience":{},"Parking":{},"Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} |

| {"Good For":{},"Ambience":{},"Parking":{},"Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} |

| {"Good For":{},"Ambience":{},"Parking":{},"Wi-Fi":"free","Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} |

| {"Take-out":"true","Good For":{"dessert":"false","latenight":"false","lunch":"false","dinner":"true","breakfast":"false","brunch":"false"},"Noise Level":"average" |

| {"Good For":{},"Ambience":{},"Parking":{},"Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} |

+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

**NOTE**

This document aligns Drill output for example purposes. Drill output is not aligned in this case.

Turn off the all text mode so we can continue to perform arithmetic operations on data.

0: jdbc:drill:zk=local> alter system set `store.json.all\_text\_mode` = false;

+-------+------------------------------------+

| ok | summary |

+-------+------------------------------------+

| true | store.json.all\_text\_mode updated. |

+-------+------------------------------------+

### 4. Explore the restaurant businesses in the data set

#### Number of restaurants in the data set

0: jdbc:drill:zk=local> select count(\*) as TotalRestaurants from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` where true=repeated\_contains(categories,'Restaurants');

+------------------+

| TotalRestaurants |

+------------------+

| 14303 |

+------------------+

#### Top restaurants in number of reviews

0: jdbc:drill:zk=local> select name,state,city,`review\_count` from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` where true=repeated\_contains(categories,'Restaurants') order by `review\_count` desc limit 10;

+------------------------+-------+-----------+--------------+

| name | state | city | review\_count |

+------------------------+-------+-----------+--------------+

| Mon Ami Gabi | NV | Las Vegas | 4084 |

| Earl of Sandwich | NV | Las Vegas | 3655 |

| Wicked Spoon | NV | Las Vegas | 3408 |

| The Buffet | NV | Las Vegas | 2791 |

| Serendipity 3 | NV | Las Vegas | 2682 |

| Bouchon | NV | Las Vegas | 2419 |

| The Buffet at Bellagio | NV | Las Vegas | 2404 |

| Bacchanal Buffet | NV | Las Vegas | 2369 |

| Hash House A Go Go | NV | Las Vegas | 2201 |

| Mesa Grill | NV | Las Vegas | 2004 |

+------------------------+-------+-----------+--------------+

#### Top restaurants in number of listed categories

0: jdbc:drill:zk=local> select name,repeated\_count(categories) as categorycount, categories from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` where true=repeated\_contains(categories,'Restaurants') order by repeated\_count(categories) desc limit 10;

+---------------------------------+---------------+---------------------------------------------------------------------------------------------------------------------------------------------------+

| name | categorycount | categories |

+---------------------------------+---------------+---------------------------------------------------------------------------------------------------------------------------------------------------+

| Binion's Hotel & Casino | 10 | ["Arts &,Entertainment","Restaurants","Bars","Casinos","Event,Planning &,Services","Lounges","Nightlife","Hotels &,Travel","American] |

| Stage Deli | 10 | ["Arts &,Entertainment","Food","Hotels","Desserts","Delis","Casinos","Sandwiches","Hotels,& Travel","Restaurants","Event Planning &,Services"] |

| Jillian's | 9 | ["Arts &,Entertainment","American (Traditional)","Music,Venues","Bars","Dance,Clubs","Nightlife","Bowling","Active,Life","Restaurants"] |

| Hotel Chocolat | 9 | ["Coffee &,Tea","Food","Cafes","Chocolatiers &,Shops","Specialty Food","Event Planning &,Services","Hotels & Travel","Hotels","Restaurants"] |

| Hotel du Vin & Bistro Edinburgh | 9 | ["Modern,European","Bars","French","Wine,Bars","Event Planning &,Services","Nightlife","Hotels &,Travel","Hotels","Restaurants"] |

| Elixir | 9 | ["Arts &,Entertainment","American (Traditional)","Music,Venues","Bars","Cocktail,Bars","Nightlife","American (New)","Local,Flavor","Restaurants"] |

| Tocasierra Spa and Fitness | 8 | ["Beauty &,Spas","Gyms","Medical Spas","Health &,Medical","Fitness & Instruction","Active,Life","Day Spas","Restaurants"] |

| Costa Del Sol At Sunset Station | 8 | ["Steakhouses","Mexican","Seafood","Event,Planning & Services","Hotels &,Travel","Italian","Restaurants","Hotels"] |

| Scottsdale Silverado Golf Club | 8 | ["Fashion","Shopping","Sporting,Goods","Active Life","Golf","American,(New)","Sports Wear","Restaurants"] |

| House of Blues | 8 | ["Arts & Entertainment","Music Venues","Restaurants","Hotels","Event Planning & Services","Hotels & Travel","American (New)","Nightlife"] |

+---------------------------------+---------------+---------------------------------------------------------------------------------------------------------------------------------------------------+

**NOTE**

This document aligns Drill output for example purposes. Drill output is not aligned in this case.

#### Top first categories in number of review counts

0: jdbc:drill:zk=local> select categories[0], count(categories[0]) as categorycount

from dfs.`/<path-to-yelp-dataset>/yelp\_academic\_dataset\_business.json`

group by categories[0]

order by count(categories[0]) desc limit 10;

+----------------------+---------------+

| EXPR$0 | categorycount |

+----------------------+---------------+

| Food | 4294 |

| Shopping | 1885 |

| Active Life | 1676 |

| Bars | 1366 |

| Local Services | 1351 |

| Mexican | 1284 |

| Hotels & Travel | 1283 |

| Fast Food | 963 |

| Arts & Entertainment | 906 |

| Hair Salons | 901 |

+----------------------+---------------+

### 5. Explore the Yelp reviews dataset and combine with the businesses.

#### Take a look at the contents of the Yelp reviews dataset.

0: jdbc:drill:zk=local> select \*

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_review.json` limit 1;

+---------------------------------+------------------------+------------------------+-------+------------+----------------------------------------------------------------------+--------+------------------------+

| votes | user\_id | review\_id | stars | date | text | type | business\_id |

+---------------------------------+------------------------+------------------------+-------+------------+----------------------------------------------------------------------+--------+------------------------+

| {"funny":0,"useful":2,"cool":1} | Xqd0DzHaiyRqVH3WRG7hzg | 15SdjuK7DmYqUAj6rjGowg | 5 | 2007-05-17 | dr. goldberg offers everything i look for in a general practitioner. | review | vcNAWiLM4dR7D2nwwJ7nCA |

+---------------------------------+------------------------+------------------------+-------+------------+----------------------------------------------------------------------+--------+------------------------+

#### Top businesses with cool rated reviews

Note that we are combining the Yelp business data set that has the overall review\_count to the Yelp review data, which holds additional details on each of the reviews themselves.

0: jdbc:drill:zk=local> Select b.name

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` b

where b.business\_id in (SELECT r.business\_id

FROM dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_review.json` r

GROUP BY r.business\_id having sum(r.votes.cool) > 2000

order by sum(r.votes.cool) desc);

+-------------------------------+

| name |

+-------------------------------+

| Earl of Sandwich |

| XS Nightclub |

| The Cosmopolitan of Las Vegas |

| Wicked Spoon |

+-------------------------------+

#### Create a view with the combined business and reviews data sets

Note that Drill views are lightweight, and can just be created in the local file system. Drill in standalone mode comes with a dfs.tmp workspace, which we can use to create views (or you can can define your own workspaces on a local or distributed file system). If you want to persist the data physically instead of in a logical view, you can use CREATE TABLE AS syntax.

0: jdbc:drill:zk=local> create or replace view dfs.tmp.businessreviews as

Select b.name,b.stars,b.state,b.city,r.votes.funny,r.votes.useful,r.votes.cool, r.`date`

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` b, dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_review.json` r

where r.business\_id=b.business\_id

+------------+-----------------------------------------------------------------+

| ok | summary |

+------------+-----------------------------------------------------------------+

| true | View 'businessreviews' created successfully in 'dfs.tmp' schema |

+------------+-----------------------------------------------------------------+

Let’s get the total number of records from the view.

0: jdbc:drill:zk=local> select count(\*) as Total from dfs.tmp.businessreviews;

+------------+

| Total |

+------------+

| 1125458 |

+------------+

In addition to these queries, you can get many deep insights using Drill’s [SQL functionality](http://drill.apache.org/docs/sql-reference). If you are not comfortable with writing queries manually, you can use a BI/Analytics tools such as Tableau/MicroStrategy to query raw files/Hive/HBase data or Drill-created views directly using Drill [ODBC/JDBC drivers](http://drill.apache.org/docs/odbc-jdbc-interfaces).

The goal of Apache Drill is to provide the freedom and flexibility in exploring data in ways we have never seen before with SQL technologies. The community is working on more exciting features around nested data and supporting data with changing schemas in upcoming releases.

The FLATTEN function can be used to dynamically rationalize semi-structured data so you can apply even deeper SQL functionality. Here is a sample query:

#### Get a flattened list of categories for each business

0: jdbc:drill:zk=local> select name, flatten(categories) as category

from dfs.`/<path-to-yelp-dataset>/yelp/yelp\_academic\_dataset\_business.json` limit 20;

+-----------------------------+---------------------------------+

| name | category |

+-----------------------------+---------------------------------+

| Eric Goldberg, MD | Doctors |

| Eric Goldberg, MD | Health & Medical |

| Pine Cone Restaurant | Restaurants |

| Deforest Family Restaurant | American (Traditional) |

| Deforest Family Restaurant | Restaurants |

| Culver's | Food |

| Culver's | Ice Cream & Frozen Yogurt |

| Culver's | Fast Food |

| Culver's | Restaurants |

| Chang Jiang Chinese Kitchen | Chinese |

| Chang Jiang Chinese Kitchen | Restaurants |

| Charter Communications | Television Stations |

| Charter Communications | Mass Media |

| Air Quality Systems | Home Services |

| Air Quality Systems | Heating & Air Conditioning/HVAC |

| McFarland Public Library | Libraries |

| McFarland Public Library | Public Services & Government |

| Green Lantern Restaurant | American (Traditional) |

| Green Lantern Restaurant | Restaurants |

| Spartan Animal Hospital | Veterinarians |

+-----------------------------+---------------------------------+

#### Top categories used in business reviews

0: jdbc:drill:zk=local> select celltbl.catl, count(celltbl.catl) categorycnt

from (select flatten(categories) catl from dfs.`/yelp\_academic\_dataset\_business.json` ) celltbl

group by celltbl.catl

order by count(celltbl.catl) desc limit 10 ;

+------------------+-------------+

| catl | categorycnt |

+------------------+-------------+

| Restaurants | 14303 |

| Shopping | 6428 |

| Food | 5209 |

| Beauty & Spas | 3421 |

| Nightlife | 2870 |

| Bars | 2378 |

| Health & Medical | 2351 |

| Automotive | 2241 |

| Home Services | 1957 |

| Fashion | 1897 |

+------------------+-------------+

Stay tuned for more features and upcoming activities in the Drill community.

To learn more about Drill, please refer to the following resources:

* Download Drill here: <http://getdrill.org/drill/download>
* [10 reasons we think Drill is cool](http://drill.apache.org/docs/why-drill)
* [A simple 10-minute tutorial](http://drill.apache.org/docs/drill-in-10-minutes)
* [More tutorials](http://drill.apache.org/docs/tutorials-introduction/)
* [Learn Drill with the MapR Sandbox](javascript:%20void(0);)

# About the MapR Sandbox

This tutorial uses the MapR Sandbox, which is a Hadoop environment pre-configured with Drill. MapR includes Drill as part of the Hadoop distribution. The MapR Sandbox with Drill is a fully functional single-node cluster that can be used to get an overview of Drill in a Hadoop environment. Business and technical analysts, product managers, and developers can use the sandbox environment to get a feel for the power and capabilities of Drill by performing various types of queries.

Hadoop is not a prerequisite for Drill and users can start ramping up with Drill by running SQL queries directly on the local file system. Refer to [Apache Drill in 10 minutes](http://drill.apache.org/docs/drill-in-10-minutes) for an introduction to using Drill in local (embedded) mode.

# Installing the Apache Drill Sandbox

## Prerequisites

The MapR Sandbox with Apache Drill runs on VMware Player and VirtualBox, free desktop applications that you can use to run a virtual machine on a Windows, Mac, or Linux PC. Before you install the MapR Sandbox with Apache Drill, verify that the host system meets the following prerequisites:

* VMware Player or VirtualBox is installed.
* At least 20 GB free hard disk space, at least 4 physical cores, and 8 GB of RAM is available. Performance increases with more RAM and free hard disk space.
* Uses one of the following 64-bit x86 architectures:
  + A 1.3 GHz or faster AMD CPU with segment-limit support in long mode
  + A 1.3 GHz or faster Intel CPU with VT-x support
* If you have an Intel CPU with VT-x support, verify that VT-x support is enabled in the host system BIOS. The BIOS settings that must be enabled for VT-x support vary depending on the system vendor. See the VMware knowledge base article at <http://kb.vmware.com/kb/1003944> for information about how to determine if VT-x support is enabled.

### VM Player Downloads

For Linux, Mac, or Windows, download the free [VMware Player](https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0) or [VirtualBox](https://www.virtualbox.org/wiki/Downloads). Optionally, you can purchase [VMware Fusion](http://www.vmware.com/products/fusion/) for Mac.

### VM Player Installation

The following list provides links to the virtual machine installation instructions:

* To install the VMware Player, see the [VMware documentation](http://www.vmware.com/support/pubs/player_pubs.html). Use of VMware Player is subject to the VMware Player end user license terms. VMware does not provide support for VMware Player. For self-help resources, see the [VMware Player FAQ](http://www.vmware.com/products/player/faqs.html).
* To install VirtualBox, see the [Oracle VM VirtualBox User Manual](http://dlc.sun.com.edgesuite.net/virtualbox/4.3.4/UserManual.pdf). By downloading VirtualBox, you agree to the terms and conditions of the respective license.

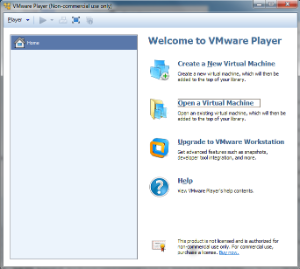
## Installing the MapR Sandbox with Apache Drill on VMware Player/VMware Fusion

Complete the following steps to install the MapR Sandbox with Apache Drill on VMware Player or VMware Fusion:

1. Download the MapR Sandbox with Drill file to a directory on your machine:  
   <https://www.mapr.com/products/mapr-sandbox-hadoop/download-sandbox-drill>
2. Open the virtual machine player, and select the **Open a Virtual Machine** option.

**Tip for VMware Fusion**

If you are running VMware Fusion, select **Import**.



1. Navigate to the directory where you downloaded the MapR Sandbox with Apache Drill file, and select MapR-Sandbox-For-Apache-Drill-<version>-vmware.ova.  
   The Import Virtual Machine dialog appears.
2. Click **Import**. The virtual machine player imports the sandbox.
3. Select MapR-Sandbox-For-Apache-Drill-<version>\_VM, and click **Play virtual machine**. It takes a few minutes for the MapR services to start. After the MapR services start and installation completes, a screen that displays the following message appears:
4. MapR-Sandbox-For-Apache-Drill-<version> installation finished successfully.
5. Please go to http://127.0.0.1:8047 to begin your experience.
6. Open a browser on your host machine and enter the URL in the browser's address field.
7. You can access the host via SSH by ssh mapr@localhost -p 2222
8. Log in to this virtual machine: Linux/Windows <Alt+F2>, Mac OS X <Options+F5>

**Note:** The URL provided corresponds to the Web Console in Apache Drill.

1. Verify that a DNS entry was created on the host machine for the virtual machine. If not, create the entry.
   * For Linux and Mac, create the entry in /etc/hosts.
   * For Windows, create the entry in the %WINDIR%\system32\drivers\etc\hosts file.

Example: 127.0.1.1 <vm\_hostname>

1. Navigate to [localhost:8047](http://localhost:8047/) to experience the Drill Web Console, or log in to the sandbox through the command line.

* Login using ssh as described in ["Getting to Know the Sandbox"](http://drill.apache.org/docs/getting-to-know-the-drill-sandbox). When prompted, enter mapr as the login name and password.
* Alternatively, access the command line on the VM: Press Alt+F2 on Windows or Option+F5 on Mac.

### What's Next

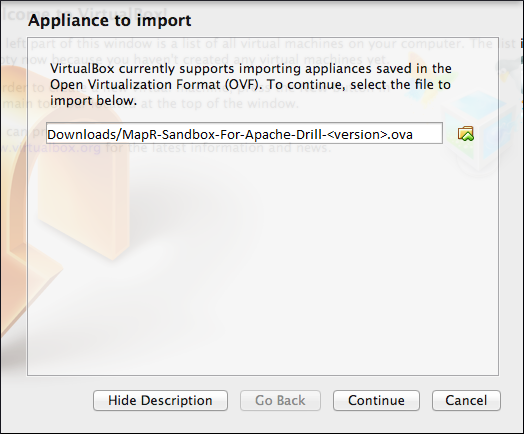
After downloading and installing the sandbox, continue with the tutorial by [Getting to Know the Drill Sandbox](http://drill.apache.org/docs/getting-to-know-the-drill-sandbox/).

## Installing the MapR Sandbox with Apache Drill on VirtualBox

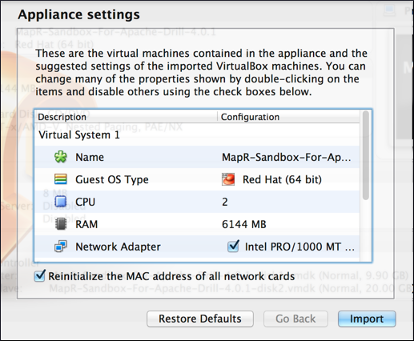
The MapR Sandbox for Apache Drill on VirtualBox comes with NAT port forwarding enabled, which allows you to access the sandbox using localhost as hostname.

Complete the following steps to install the MapR Sandbox with Apache Drill on VirtualBox:

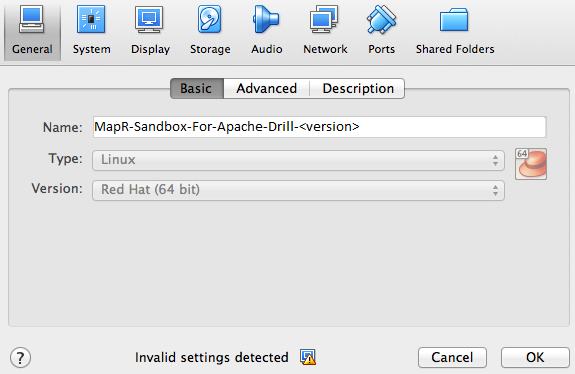
1. Download the MapR Sandbox with Apache Drill file to a directory on your machine:  
   <https://www.mapr.com/products/mapr-sandbox-hadoop/download-sandbox-drill>
2. Open the virtual machine player.
3. Select **File > Import Appliance**. The Import Virtual Appliance dialog appears.



1. Navigate to the directory where you downloaded the MapR Sandbox with Apache Drill, select MapR-Sandbox-For-Apache-Drill-<version>.ova, and click **Next**. The Appliance Settings window appears.

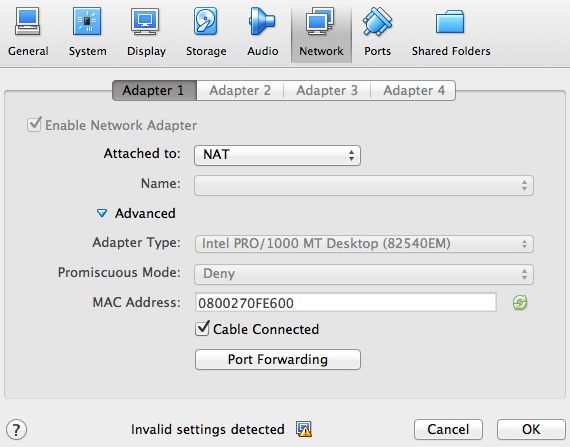


1. Select the check box at the bottom of the screen: **Reinitialize the MAC address of all network cards**, then click **Import**. The Import Appliance imports the sandbox.
2. When the import completes, select **Settings**. The VirtualBox - Settings dialog appears.



1. Select **Network**.

Verify that Adapter 1 is attached to **NAT**. This option should work for most scenarios. However, if you are going to use a wired Ethernet connection, you can select **NAT Network**. If you use ODBC or JDBC on a remote host, select **Bridged Adapter**.



1. Click **OK** to continue.
2. Click **Start**. It takes a few minutes for the MapR services to start. After the MapR services start and installation completes, a screen that displays the following message appears:
3. MapR-Sandbox-For-Apache-Drill-<version> installation finished successfully.
4. Please go to http://127.0.0.1:8047 to begin your experience.
5. Open a browser on your host machine and enter the URL in the browser's address field.
6. You can access the host via SSH by ssh mapr@localhost -p 2222
7. Log in to this virtual machine: Linux/Windows <Alt+F2>, Mac OS X <Options+F5>

**Note:** The URL provided corresponds to the Web Console in Apache Drill.

1. Navigate to [localhost:8047](http://localhost:8047/) to experience the Drill Web Console, or log into the sandbox through the command line.
   * Login using ssh as described in ["Getting to Know the Sandbox"](http://drill.apache.org/docs/getting-to-know-the-drill-sandbox). When prompted, enter mapr as the login name and password.
   * Alternatively, access the command line on the VM: Press Alt+F2 on Windows or Option+F5 on Mac.

### What's Next

After downloading and installing the sandbox, continue with the tutorial by [Getting to Know the Drill Sandbox](http://drill.apache.org/docs/getting-to-know-the-drill-sandbox/).

# Getting to Know the Drill Sandbox

This section covers key information about the Apache Drill tutorial. After [installing the Drill sandbox](http://drill.apache.org/docs/installing-the-apache-drill-sandbox) and starting the sandbox, you can open another terminal window (Linux) or Command Prompt (Windows) and use the secure shell (ssh) to connect to the VM, assuming ssh is installed. Use the following login name and password: mapr/mapr. For example:

$ ssh mapr@localhost -p 2222

Password:

Last login: Mon Sep 15 13:46:08 2014 from 10.250.0.28

Welcome to your Mapr Demo virtual machine.

Using the secure shell instead of the VM interface has some advantages. You can copy/paste commands from the tutorial and avoid mouse control problems.

Drill includes a shell for connecting to relational databases and executing SQL commands. On the sandbox, the Drill shell runs in embedded mode. After logging into the sandbox, use the SQLLine command. The Drill shell appears, and you can run Drill queries.

[mapr@maprdemo ~]$ sqlline

apache drill 1.1.0

"Does your data know the Drill?"

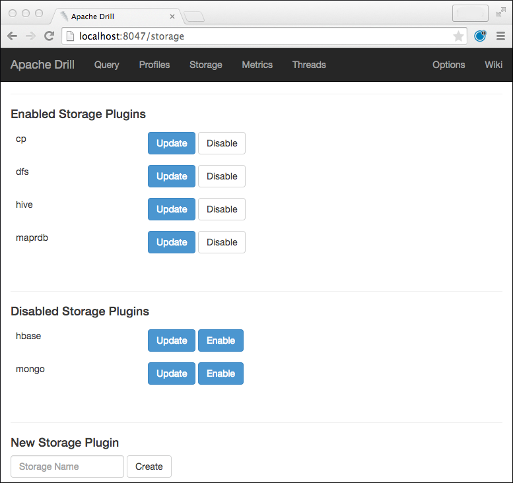
0: jdbc:drill:>

In this tutorial you query a number of data sets, including Hive and HBase, and files on the file system, such as CSV, JSON, and Parquet files. To access these diverse data sources, you connect Drill to storage plugins.

## Storage Plugin Overview

You use a [storage plugin](http://drill.apache.org/docs/connect-a-data-source-introduction) to connect to a data source, such as a file or the Hive metastore. Take a look at the storage plugin definitions by opening the Storage tab in the Drill Web Console. Launch a web browser and go to: http://<IP address>:8047/storage.

The control panel for managing storage plugins appears.



You see the following storage plugin configurations:

* cp
* dfs
* hive
* maprdb
* hbase
* mongo

Click Update to examine a configuration.

If you've used an installation of Drill before using the sandbox, you might notice that a few storage plugin configurations in the sandbox differ from the same storage plugin configurations in a Drill installation. The sandbox configurations of dfs, hive, maprdb, and hbase storage plugins definitions play a role in simulating the cluster environment for running the tutorial.

### dfs

The dfs storage plugin in the sandbox configures a connection to the MapR file system (MapR-FS).

The dfs storage plugin configuration in the sandbox also includes a set of workspaces; each one represents a location in MapR-FS:

* root: access to the root file system location
* clicks: access to nested JSON log data
* logs: access to flat (non-nested) JSON log data in the logs directory and its subdirectories
* views: a workspace for creating views

The dfs configuration includes format definitions.

{

"type": "file",

"enabled": true,

"connection": "maprfs:///",

"workspaces": {

"root": {

"location": "/mapr/demo.mapr.com/data",

"writable": false,

"defaultInputFormat": null

},

"clicks": {

"location": "/mapr/demo.mapr.com/data/nested",

"writable": true,

"defaultInputFormat": "parquet"

},

. . .

"formats": {

. . .

"csv": {

"type": "text",

"extensions": [

"csv"

],

"delimiter": ","

},

. . .

"json": {

"type": "json"

},

"maprdb": {

"type": "maprdb"

}

. . .

### maprdb

The maprdb is a configuration for MapR-DB in the sandbox. You use this format in the sandbox to query MapR-DB/HBase tables.

### hive

The hive configuration for a Hive data warehouse within the sandbox. Drill connects to the Hive metastore by using the configured metastore thrift URI. Metadata for Hive tables is automatically available for users to query.

{

"type": "hive",

"enabled": true,

"configProps": {

"hive.metastore.uris": "thrift://localhost:9083",

"hive.metastore.sasl.enabled": "false"

}

}

Do not use this storage plugin configuration outside the sandbox. Use the configuration for either the [remote or embedded metastore configuration](http://drill.apache.org/docs/hive-storage-plugin/).

## What's Next

Start running queries by going to [Lesson 1: Learn About the Data Set](http://drill.apache.org/docs/lesson-1-learn-about-the-data-set).

# Lesson 1: Learn about the Data Set

## Goal

This lesson is simply about discovering what data is available, in what format, using simple SQL SELECT statements. Drill is capable of analyzing data without prior knowledge or definition of its schema. This means that you can start querying data immediately (and even as it changes), regardless of its format.

The data set for the tutorial consists of:

* Transactional data: stored as a Hive table
* Product catalog and master customer data: stored as MapR-DB tables
* Clickstream and logs data: stored in the MapR file system as JSON files

## Queries in This Lesson

This lesson consists of select \* queries on each data source.

## Before You Begin

### Start the Drill Shell

If the Drill shell is not already started, use a Terminal or Command Prompt to log into the demo VM as mapr, then enter sqlline, as described in ["Getting to Know the Sandbox"](http://drill.apache.org/docs/getting-to-know-the-drill-sandbox):

You can run queries to complete the tutorial. To exit from the Drill shell, type:

0: jdbc:drill:> !quit

Examples in this tutorial use the Drill shell. You can also execute queries using the Drill Web Console.

### Enable the DECIMAL Data Type

This tutorial uses the DECIMAL data type in some examples. The DECIMAL data type is disabled by default in this release, so enable the DECIMAL data type before proceeding:

alter session set `planner.enable\_decimal\_data\_type`=true;

+-------+--------------------------------------------+

| ok | summary |

+-------+--------------------------------------------+

| true | planner.enable\_decimal\_data\_type updated. |

+-------+--------------------------------------------+

1 row selected

### List the available workspaces and databases:

0: jdbc:drill:> show databases;

+---------------------+

| SCHEMA\_NAME |

+---------------------+

| INFORMATION\_SCHEMA |

| cp.default |

| dfs.clicks |

| dfs.default |

| dfs.logs |

| dfs.root |

| dfs.tmp |

| dfs.views |

| hive.default |

| maprdb |

| sys |

+---------------------+

This command exposes all the metadata available from the storage plugins configured with Drill as a set of schemas. The Hive and MapR-DB databases, file system, and other data are configured in the file system. As you run queries in the tutorial, you run the USE command to switch among these schemas. Switching schemas in this way resembles using different database schemas (namespaces) in a relational database system.

## Query Hive Tables

The orders table is a six-column Hive table defined in the Hive metastore. This is a Hive external table pointing to the data stored in flat files on the MapR file system. The orders table contains 122,000 rows.

### Set the schema to hive:

0: jdbc:drill:> use hive.`default`;

+-------+-------------------------------------------+

| ok | summary |

+-------+-------------------------------------------+

| true | Default schema changed to [hive.default] |

+-------+-------------------------------------------+

1 row selected

You will run the USE command throughout this tutorial. The USE command sets the schema for the current session.

### Describe the table:

You can use the DESCRIBE command to show the columns and data types for a Hive table:

0: jdbc:drill:> describe orders;

+-------------+------------+-------------+

| COLUMN\_NAME | DATA\_TYPE | IS\_NULLABLE |

+-------------+------------+-------------+

| order\_id | BIGINT | YES |

| month | VARCHAR | YES |

| cust\_id | BIGINT | YES |

| state | VARCHAR | YES |

| prod\_id | BIGINT | YES |

| order\_total | INTEGER | YES |

+-------------+------------+-------------+

The DESCRIBE command returns complete schema information for Hive tables based on the metadata available in the Hive metastore.

### Select 5 rows from the orders table:

0: jdbc:drill:> select \* from orders limit 5;

+------------+------------+------------+------------+------------+-------------+

| order\_id | month | cust\_id | state | prod\_id | order\_total |

+------------+------------+------------+------------+------------+-------------+

| 67212 | June | 10001 | ca | 909 | 13 |

| 70302 | June | 10004 | ga | 420 | 11 |

| 69090 | June | 10011 | fl | 44 | 76 |

| 68834 | June | 10012 | ar | 0 | 81 |

| 71220 | June | 10018 | az | 411 | 24 |

+------------+------------+------------+------------+------------+-------------+

Because orders is a Hive table, you can query the data in the same way that you would query the columns in a relational database table. Note the use of the standard LIMIT clause, which limits the result set to the specified number of rows. You can use LIMIT with or without an ORDER BY clause.

Drill provides seamless integration with Hive by allowing queries on Hive tables defined in the metastore with no extra configuration. Hive is not a prerequisite for Drill, but simply serves as a storage plugin or data source for Drill. Drill also lets users query all Hive file formats (including custom serdes). Additionally, any UDFs defined in Hive can be leveraged as part of Drill queries.

Because Drill has its own low-latency SQL query execution engine, you can query Hive tables with high performance and support for interactive and ad-hoc data exploration.

## Query MapR-DB and HBase Tables

The customers and products tables are MapR-DB tables. MapR-DB is an enterprise in-Hadoop NoSQL database. It exposes the HBase API to support application development. Every MapR-DB table has a row\_key, in addition to one or more column families. Each column family contains one or more specific columns. The row\_key value is a primary key that uniquely identifies each row.

Drill directly queries MapR-DB and HBase tables. Unlike other SQL on Hadoop options, Drill requires no overlay schema definitions in Hive to work with this data. Drill removes the pain of having to manage duplicate schemas in Hive when you have a MapR-DB or HBase table with thousands of columns typical of a time-series database.

### Products Table

The products table has two column families.

|  |  |
| --- | --- |
| Column Family | Columns |
| details | name category |
| pricing | price |

The products table contains 965 rows.

### Customers Table

The Customers table has three column families.

|  |  |
| --- | --- |
| Column Family | Columns |
| address | state |
| loyalty | agg\_rev membership |
| personal | age gender |

The customers table contains 993 rows.

### Set the workspace to maprdb:

use maprdb;

+-------+-------------------------------------+

| ok | summary |

+-------+-------------------------------------+

| true | Default schema changed to [maprdb] |

+-------+-------------------------------------+

1 row selected

### Describe the tables:

0: jdbc:drill:> describe customers;

+--------------+------------------------+--------------+

| COLUMN\_NAME | DATA\_TYPE | IS\_NULLABLE |

+--------------+------------------------+--------------+

| row\_key | ANY | NO |

| address | (VARCHAR(1), ANY) MAP | NO |

| loyalty | (VARCHAR(1), ANY) MAP | NO |

| personal | (VARCHAR(1), ANY) MAP | NO |

+--------------+------------------------+--------------+

4 rows selected

0: jdbc:drill:> describe products;

+--------------+------------------------+--------------+

| COLUMN\_NAME | DATA\_TYPE | IS\_NULLABLE |

+--------------+------------------------+--------------+

| row\_key | ANY | NO |

| details | (VARCHAR(1), ANY) MAP | NO |

| pricing | (VARCHAR(1), ANY) MAP | NO |

+--------------+------------------------+--------------+

3 rows selected

Unlike the Hive example, the DESCRIBE command does not return the full schema up to the column level. Wide-column NoSQL databases such as MapR-DB and HBase can be schema-less by design; every row has its own set of column name-value pairs in a given column family, and the column value can be of any data type, as determined by the application inserting the data.

A “MAP” complex type in Drill represents this variable column name-value structure, and “ANY” represents the fact that the column value can be of any data type. Observe the row\_key, which is also simply bytes and has the type ANY.

### Select 5 rows from the products table:

0: jdbc:drill:> select \* from products limit 5;

+--------------+----------------------------------------------------------------------------------------------------------------+-------------------+

| row\_key | details | pricing |

+--------------+----------------------------------------------------------------------------------------------------------------+-------------------+

| [B@b01c5f8 | {"category":"bGFwdG9w","name":"U29ueSBub3RlYm9vaw=="} | {"price":"OTU5"} |

| [B@5edfe5ad | {"category":"RW52ZWxvcGVz","name":"IzEwLTQgMS84IHggOSAxLzIgUHJlbWl1bSBEaWFnb25hbCBTZWFtIEVudmVsb3Blcw=="} | {"price":"MTY="} |

| [B@3d5ff184 | {"category":"U3RvcmFnZSAmIE9yZ2FuaXphdGlvbg==","name":"MjQgQ2FwYWNpdHkgTWF4aSBEYXRhIEJpbmRlciBSYWNrc1BlYXJs"} | {"price":"MjEx"} |

| [B@65e93096 | {"category":"TGFiZWxz","name":"QXZlcnkgNDk4"} | {"price":"Mw=="} |

| [B@3074fc1f | {"category":"TGFiZWxz","name":"QXZlcnkgNDk="} | {"price":"Mw=="} |

+--------------+----------------------------------------------------------------------------------------------------------------+-------------------+

5 rows selected

Given that Drill requires no up front schema definitions indicating data types, the query returns the raw byte arrays for column values, just as they are stored in MapR-DB (or HBase). Observe that the column families (details and pricing) have the map data type and appear as JSON strings.

In Lesson 2, you will use CAST functions to return typed data for each column.

### Select 5 rows from the customers table:

+0: jdbc:drill:> select \* from customers limit 5;

+--------------+-----------------------+-------------------------------------------------+---------------------------------------------------------------------------------------+

| row\_key | address | loyalty | personal |

+--------------+-----------------------+-------------------------------------------------+---------------------------------------------------------------------------------------+

| [B@3ed2649e | {"state":"InZhIg=="} | {"agg\_rev":"MTk3","membership":"InNpbHZlciI="} | {"age":"IjE1LTIwIg==","gender":"IkZFTUFMRSI=","name":"IkNvcnJpbmUgTWVjaGFtIg=="} |

| [B@66cbe14a | {"state":"ImluIg=="} | {"agg\_rev":"MjMw","membership":"InNpbHZlciI="} | {"age":"IjI2LTM1Ig==","gender":"Ik1BTEUi","name":"IkJyaXR0YW55IFBhcmsi"} |

| [B@5333f5ff | {"state":"ImNhIg=="} | {"agg\_rev":"MjUw","membership":"InNpbHZlciI="} | {"age":"IjI2LTM1Ig==","gender":"Ik1BTEUi","name":"IlJvc2UgTG9rZXki"} |

| [B@785b6305 | {"state":"Im1lIg=="} | {"agg\_rev":"MjYz","membership":"InNpbHZlciI="} | {"age":"IjUxLTEwMCI=","gender":"IkZFTUFMRSI=","name":"IkphbWVzIEZvd2xlciI="} |

| [B@37c21afe | {"state":"Im1uIg=="} | {"agg\_rev":"MjAy","membership":"InNpbHZlciI="} | {"age":"IjUxLTEwMCI=","gender":"Ik9USEVSIg==","name":"Ikd1aWxsZXJtbyBLb2VobGVyIg=="} |

+--------------+-----------------------+-------------------------------------------------+---------------------------------------------------------------------------------------+

5 rows selected

Again, the table returns byte data that needs to be cast to readable data types.

## Query the File System

Along with querying a data source with full schemas (such as Hive) and partial schemas (such as MapR-DB and HBase), Drill offers the unique capability to perform SQL queries directly on file system. The file system could be a local file system, or a distributed file system such as MapR-FS, HDFS, or S3.

In the context of Drill, a file or a directory is synonymous with a relational database “table.” Therefore, you can perform SQL operations directly on files and directories without the need for up-front schema definitions or schema management for any model changes. The schema is discovered on the fly based on the query. Drill supports queries on a variety of file formats including text, CSV, Parquet, and JSON.

In this example, the clickstream data coming from the mobile/web applications is in JSON format. The JSON files have the following structure:

{"trans\_id":31920,"date":"2014-04-26","time":"12:17:12","user\_info":{"cust\_id":22526,"device":"IOS5","state":"il"},"trans\_info":{"prod\_id":[174,2],"purch\_flag":"false"}}

{"trans\_id":31026,"date":"2014-04-20","time":"13:50:29","user\_info":{"cust\_id":16368,"device":"AOS4.2","state":"nc"},"trans\_info":{"prod\_id":[],"purch\_flag":"false"}}

{"trans\_id":33848,"date":"2014-04-10","time":"04:44:42","user\_info":{"cust\_id":21449,"device":"IOS6","state":"oh"},"trans\_info":{"prod\_id":[582],"purch\_flag":"false"}}

The clicks.json and clicks.campaign.json files contain metadata as part of the data itself (referred to as “self-describing” data). The data elements are complex, or nested. The initial queries below do not show how to unpack the nested data, but they show that easy access to the data requires no setup beyond the definition of a workspace.

### Query nested clickstream data

#### Set the workspace to dfs.clicks:

0: jdbc:drill:> use dfs.clicks;

+-------+-----------------------------------------+

| ok | summary |

+-------+-----------------------------------------+

| true | Default schema changed to [dfs.clicks] |

+-------+-----------------------------------------+

1 row selected

In this case, setting the workspace is a mechanism for making queries easier to write. When you specify a file system workspace, you can shorten references to files in your queries. Instead of having to provide the complete path to a file, you can provide the path relative to a directory location specified in the workspace. For example:

"location": "/mapr/demo.mapr.com/data/nested"

Any file or directory that you want to query in this path can be referenced relative to this path. The clicks directory referred to in the following query is directly below the nested directory.

#### Select 2 rows from the clicks.json file:

0: jdbc:drill:> select \* from `clicks/clicks.json` limit 2;

+-----------+-------------+-----------+---------------------------------------------------+-------------------------------------------+

| trans\_id | date | time | user\_info | trans\_info |

+-----------+-------------+-----------+---------------------------------------------------+-------------------------------------------+

| 31920 | 2014-04-26 | 12:17:12 | {"cust\_id":22526,"device":"IOS5","state":"il"} | {"prod\_id":[174,2],"purch\_flag":"false"} |

| 31026 | 2014-04-20 | 13:50:29 | {"cust\_id":16368,"device":"AOS4.2","state":"nc"} | {"prod\_id":[],"purch\_flag":"false"} |

+-----------+-------------+-----------+---------------------------------------------------+-------------------------------------------+

2 rows selected

The FROM clause reference points to a specific file. Drill expands the traditional concept of a “table reference” in a standard SQL FROM clause to refer to a file in a local or distributed file system.

The only special requirement is the use of back ticks to enclose the file path. This is necessary whenever the file path contains Drill reserved words or characters.

#### Select 2 rows from the campaign.json file:

0: jdbc:drill:> select \* from `clicks/clicks.campaign.json` limit 2;

+-----------+-------------+-----------+---------------------------------------------------+---------------------+----------------------------------------+

| trans\_id | date | time | user\_info | ad\_info | trans\_info |

+-----------+-------------+-----------+---------------------------------------------------+---------------------+----------------------------------------+

| 35232 | 2014-05-10 | 00:13:03 | {"cust\_id":18520,"device":"AOS4.3","state":"tx"} | {"camp\_id":"null"} | {"prod\_id":[7,7],"purch\_flag":"true"} |

| 31995 | 2014-05-22 | 16:06:38 | {"cust\_id":17182,"device":"IOS6","state":"fl"} | {"camp\_id":"null"} | {"prod\_id":[],"purch\_flag":"false"} |

+-----------+-------------+-----------+---------------------------------------------------+---------------------+----------------------------------------+

2 rows selected

Notice that with a select \* query, any complex data types such as maps and arrays return as JSON strings. You will see how to unpack this data using various SQL functions and operators in the next lesson.

## Query Logs Data

Unlike the previous example where we performed queries against clicks data in one file, logs data is stored as partitioned directories on the file system. The logs directory has three subdirectories:

* 2012
* 2013
* 2014

Each of these year directories fans out to a set of numbered month directories, and each month directory contains a JSON file with log records for that month. The total number of records in all log files is 48000.

The files in the logs directory and its subdirectories are JSON files. There are many of these files, but you can use Drill to query them all as a single data source, or to query a subset of the files.

#### Set the workspace to dfs.logs:

0: jdbc:drill:> use dfs.logs;

+-------+---------------------------------------+

| ok | summary |

+-------+---------------------------------------+

| true | Default schema changed to [dfs.logs] |

+-------+---------------------------------------+

1 row selected

#### Select 2 rows from the logs directory:

0: jdbc:drill:> select \* from logs limit 2;

+-------+-------+-----------+-------------+-----------+----------+---------+--------+----------+-----------+----------+-------------+

| dir0 | dir1 | trans\_id | date | time | cust\_id | device | state | camp\_id | keywords | prod\_id | purch\_flag |

+-------+-------+-----------+-------------+-----------+----------+---------+--------+----------+-----------+----------+-------------+

| 2012 | 8 | 109 | 08/07/2012 | 20:33:13 | 144618 | IOS5 | ga | 4 | hey | 6 | false |

| 2012 | 8 | 119 | 08/19/2012 | 03:37:50 | 17 | IOS5 | tx | 16 | and | 50 | false |

+-------+-------+-----------+-------------+-----------+----------+---------+--------+----------+-----------+----------+-------------+

2 rows selected

Note that this is flat JSON data. The dfs.clicks workspace location property points to a directory that contains the logs directory, making the FROM clause reference for this query very simple. You do not have to refer to the complete directory path on the file system.

The column names dir0 and dir1 are special Drill variables that identify subdirectories below the logs directory. In Lesson 3, you will do more complex queries that leverage these dynamic variables.

#### Find the total number of rows in the logs directory (all files):

0: jdbc:drill:> select count(\*) from logs;

+---------+

| EXPR$0 |

+---------+

| 48000 |

+---------+

1 row selected

This query traverses all of the files in the logs directory and its subdirectories to return the total number of rows in those files.

# What's Next

Go to [Lesson 2: Run Queries with ANSI SQL](http://drill.apache.org/docs/lesson-2-run-queries-with-ansi-sql).

# Lesson 2: Run Queries with ANSI SQL

## Goal

This lesson shows how to do some standard SQL analysis in Apache Drill: for example, summarizing data by using simple aggregate functions and connecting data sources by using joins. Note that Apache Drill provides ANSI SQL support, not a “SQL-like” interface.

## Queries in This Lesson

Now that you know what the data sources look like in their raw form, using select \* queries, try running some simple but more useful queries on each data source. These queries demonstrate how Drill supports ANSI SQL constructs and also how you can combine data from different data sources in a single SELECT statement.

* Show an aggregate query on a single file or table. Use GROUP BY, WHERE, HAVING, and ORDER BY clauses.
* Perform joins between Hive, MapR-DB, and file system data sources.
* Use table and column aliases.
* Create a Drill view.

## Aggregation

### Set the schema to hive:

0: jdbc:drill:> use hive.`default`;

+-------+-------------------------------------------+

| ok | summary |

+-------+-------------------------------------------+

| true | Default schema changed to [hive.default] |

+-------+-------------------------------------------+

1 row selected

### Return sales totals by month:

0: jdbc:drill:> select `month`, sum(order\_total)

from orders group by `month` order by 2 desc;

+------------+---------+

| month | EXPR$1 |

+------------+---------+

| June | 950481 |

| May | 947796 |

| March | 836809 |

| April | 807291 |

| July | 757395 |

| October | 676236 |

| August | 572269 |

| February | 532901 |

| September | 373100 |

| January | 346536 |

+------------+---------+

10 rows selected

Drill supports SQL aggregate functions such as SUM, MAX, AVG, and MIN. Standard SQL clauses work in the same way in Drill queries as in relational database queries.

Note that back ticks are required for the “month” column only because “month” is a reserved word in SQL.

### Return the top 20 sales totals by month and state:

0: jdbc:drill:> select `month`, state, sum(order\_total) as sales from orders group by `month`, state

order by 3 desc limit 20;

+-----------+--------+---------+

| month | state | sales |

+-----------+--------+---------+

| May | ca | 119586 |

| June | ca | 116322 |

| April | ca | 101363 |

| March | ca | 99540 |

| July | ca | 90285 |

| October | ca | 80090 |

| June | tx | 78363 |

| May | tx | 77247 |

| March | tx | 73815 |

| August | ca | 71255 |

| April | tx | 68385 |

| July | tx | 63858 |

| February | ca | 63527 |

| June | fl | 62199 |

| June | ny | 62052 |

| May | fl | 61651 |

| May | ny | 59369 |

| October | tx | 55076 |

| March | fl | 54867 |

| March | ny | 52101 |

+-----------+--------+---------+

20 rows selected

Note the alias for the result of the SUM function. Drill supports column aliases and table aliases.

## HAVING Clause

This query uses the HAVING clause to constrain an aggregate result.

### Set the workspace to dfs.clicks

0: jdbc:drill:> use dfs.clicks;

+-------+-----------------------------------------+

| ok | summary |

+-------+-----------------------------------------+

| true | Default schema changed to [dfs.clicks] |

+-------+-----------------------------------------+

1 row selected

### Return total number of clicks for devices that indicate high click-throughs:

0: jdbc:drill:> select t.user\_info.device, count(\*) from `clicks/clicks.json` t

group by t.user\_info.device

having count(\*) > 1000;

+---------+---------+

| EXPR$0 | EXPR$1 |

+---------+---------+

| IOS5 | 11814 |

| AOS4.2 | 5986 |

| IOS6 | 4464 |

| IOS7 | 3135 |

| AOS4.4 | 1562 |

| AOS4.3 | 3039 |

+---------+---------+

6 rows selected

The aggregate is a count of the records for each different mobile device in the clickstream data. Only the activity for the devices that registered more than 1000 transactions qualify for the result set.

## UNION Operator

Use the same workspace as before (dfs.clicks).

### Combine clicks activity from before and after the marketing campaign

0: jdbc:drill:> select t.trans\_id transaction, t.user\_info.cust\_id customer from `clicks/clicks.campaign.json` t

union all

select u.trans\_id, u.user\_info.cust\_id from `clicks/clicks.json` u limit 5;

+-------------+------------+

| transaction | customer |

+-------------+------------+

| 35232 | 18520 |

| 31995 | 17182 |

| 35760 | 18228 |

| 37090 | 17015 |

| 37838 | 18737 |

+-------------+------------+

This UNION ALL query returns rows that exist in two files (and includes any duplicate rows from those files): clicks.campaign.json and clicks.json.

## Subqueries

### Set the workspace to hive:

0: jdbc:drill:> use hive.`default`;

+-------+-------------------------------------------+

| ok | summary |

+-------+-------------------------------------------+

| true | Default schema changed to [hive.default] |

+-------+-------------------------------------------+

1 row selected

### Compare order totals across states:

0: jdbc:drill:> select ny\_sales.cust\_id, ny\_sales.total\_orders, ca\_sales.total\_orders

from

(select o.cust\_id, sum(o.order\_total) as total\_orders from hive.orders o where state = 'ny' group by o.cust\_id) ny\_sales

left outer join

(select o.cust\_id, sum(o.order\_total) as total\_orders from hive.orders o where state = 'ca' group by o.cust\_id) ca\_sales

on ny\_sales.cust\_id = ca\_sales.cust\_id

order by ny\_sales.cust\_id

limit 20;

+------------+------------+------------+

| cust\_id | ny\_sales | ca\_sales |

+------------+------------+------------+

| 1001 | 72 | 47 |

| 1002 | 108 | 198 |

| 1003 | 83 | null |

| 1004 | 86 | 210 |

| 1005 | 168 | 153 |

| 1006 | 29 | 326 |

| 1008 | 105 | 168 |

| 1009 | 443 | 127 |

| 1010 | 75 | 18 |

| 1012 | 110 | null |

| 1013 | 19 | null |

| 1014 | 106 | 162 |

| 1015 | 220 | 153 |

| 1016 | 85 | 159 |

| 1017 | 82 | 56 |

| 1019 | 37 | 196 |

| 1020 | 193 | 165 |

| 1022 | 124 | null |

| 1023 | 166 | 149 |

| 1024 | 233 | null |

+------------+------------+------------+

This example demonstrates Drill support for subqueries.

## CAST Function

### Use the maprdb workspace:

0: jdbc:drill:> use maprdb;

+-------+-------------------------------------+

| ok | summary |

+-------+-------------------------------------+

| true | Default schema changed to [maprdb] |

+-------+-------------------------------------+

1 row selected (0.088 seconds)

### Return customer data with appropriate data types

0: jdbc:drill:> select cast(row\_key as int) as cust\_id, cast(t.personal.name as varchar(20)) as name,

cast(t.personal.gender as varchar(10)) as gender, cast(t.personal.age as varchar(10)) as age,

cast(t.address.state as varchar(4)) as state, cast(t.loyalty.agg\_rev as dec(7,2)) as agg\_rev,

cast(t.loyalty.membership as varchar(20)) as membership

from customers t limit 5;

+----------+----------------------+-----------+-----------+--------+----------+-------------+

| cust\_id | name | gender | age | state | agg\_rev | membership |

+----------+----------------------+-----------+-----------+--------+----------+-------------+

| 10001 | "Corrine Mecham" | "FEMALE" | "15-20" | "va" | 197.00 | "silver" |

| 10005 | "Brittany Park" | "MALE" | "26-35" | "in" | 230.00 | "silver" |

| 10006 | "Rose Lokey" | "MALE" | "26-35" | "ca" | 250.00 | "silver" |

| 10007 | "James Fowler" | "FEMALE" | "51-100" | "me" | 263.00 | "silver" |

| 10010 | "Guillermo Koehler" | "OTHER" | "51-100" | "mn" | 202.00 | "silver" |

+----------+----------------------+-----------+-----------+--------+----------+-------------+

Note the following features of this query:

* The CAST function is required for every column in the table. This function returns the MapR-DB/HBase binary data as readable integers and strings. Alternatively, you can use CONVERT\_TO/CONVERT\_FROM functions to decode the string columns. CONVERT\_TO/CONVERT\_FROM are more efficient than CAST in most cases. Use only CONVERT\_TO to convert binary types to any type other than VARCHAR.
* The row\_key column functions as the primary key of the table (a customer ID in this case).
* The table alias t is required; otherwise the column family names would be parsed as table names and the query would return an error.

### Remove the quotes from the strings:

You can use the regexp\_replace function to remove the quotes around the strings in the query results. For example, to return a state name va instead of “va”:

0: jdbc:drill:> select cast(row\_key as int), regexp\_replace(cast(t.address.state as varchar(10)),'"','')

from customers t limit 1;

+------------+------------+

| EXPR$0 | EXPR$1 |

+------------+------------+

| 10001 | va |

+------------+------------+

1 row selected

## CREATE VIEW Command

0: jdbc:drill:> use dfs.views;

+-------+----------------------------------------+

| ok | summary |

+-------+----------------------------------------+

| true | Default schema changed to [dfs.views] |

+-------+----------------------------------------+

1 row selected

### Use a mutable workspace:

A mutable (or writable) workspace is a workspace that is enabled for “write” operations. This attribute is part of the storage plugin configuration. You can create Drill views and tables in mutable workspaces.

### Create a view on a MapR-DB table

0: jdbc:drill:> create or replace view custview as select cast(row\_key as int) as cust\_id,

cast(t.personal.name as varchar(20)) as name,

cast(t.personal.gender as varchar(10)) as gender,

cast(t.personal.age as varchar(10)) as age,

cast(t.address.state as varchar(4)) as state,

cast(t.loyalty.agg\_rev as dec(7,2)) as agg\_rev,

cast(t.loyalty.membership as varchar(20)) as membership

from maprdb.customers t;

+-------+-------------------------------------------------------------+

| ok | summary |

+-------+-------------------------------------------------------------+

| true | View 'custview' created successfully in 'dfs.views' schema |

+-------+-------------------------------------------------------------+

1 row selected

Drill provides CREATE OR REPLACE VIEW syntax similar to relational databases to create views. Use the OR REPLACE option to make it easier to update the view later without having to remove it first. Note that the FROM clause in this example must refer to maprdb.customers. The MapR-DB tables are not directly visible to the dfs.views workspace.

Unlike a traditional database where views typically are DBA/developer-driven operations, file system-based views in Drill are very lightweight. A view is simply a special file with a specific extension (.drill). You can store views even in your local file system or point to a specific workspace. You can specify any query against any Drill data source in the body of the CREATE VIEW statement.

Drill provides a decentralized metadata model. Drill is able to query metadata defined in data sources such as Hive, HBase, and the file system. Drill also supports the creation of metadata in the file system.

### Query data from the view:

0: jdbc:drill:> select \* from custview limit 1;

+----------+-------------------+-----------+----------+--------+----------+-------------+

| cust\_id | name | gender | age | state | agg\_rev | membership |

+----------+-------------------+-----------+----------+--------+----------+-------------+

| 10001 | "Corrine Mecham" | "FEMALE" | "15-20" | "va" | 197.00 | "silver" |

+----------+-------------------+-----------+----------+--------+----------+-------------+

1 row selected

Once the users know what data is available by exploring it directly from the file system, views can be used as a way to read the data into downstream tools such as Tableau and MicroStrategy for analysis and visualization. For these tools, a view appears simply as a “table” with selectable “columns” in it.

## Query Across Data Sources

Continue using dfs.views for this query.

### Join the customers view and the orders table:

0: jdbc:drill:> select membership, sum(order\_total) as sales from hive.orders, custview

where orders.cust\_id=custview.cust\_id

group by membership order by 2;

+------------+------------+

| membership | sales |

+------------+------------+

| "basic" | 380665 |

| "silver" | 708438 |

| "gold" | 2787682 |

+------------+------------+

3 rows selected

In this query, we are reading data from a MapR-DB table (represented by custview) and combining it with the order information in Hive. When doing cross data source queries such as this, you need to use fully qualified table/view names. For example, the orders table is prefixed by “hive,” which is the storage plugin name registered with Drill. We are not using any prefix for “custview” because we explicitly switched the dfs.views workspace where custview is stored.

Note: If the results of any of your queries appear to be truncated because the rows are wide, set the maximum width of the display to 10000:

Do not use a semicolon for this SET command.

### Join the customers, orders, and clickstream data:

0: jdbc:drill:> select custview.membership, sum(orders.order\_total) as sales from hive.orders, custview,

dfs.`/mapr/demo.mapr.com/data/nested/clicks/clicks.json` c

where orders.cust\_id=custview.cust\_id and orders.cust\_id=c.user\_info.cust\_id

group by custview.membership order by 2;

+------------+------------+

| membership | sales |

+------------+------------+

| "basic" | 372866 |

| "silver" | 728424 |

| "gold" | 7050198 |

+------------+------------+

3 rows selected

This three-way join selects from three different data sources in one query:

* hive.orders table
* custview (a view of the HBase customers table)
* clicks.json file

The join column for both sets of join conditions is the cust\_id column. The views workspace is used for this query so that custview can be accessed. The hive.orders table is also visible to the query.

However, note that the JSON file is not directly visible from the views workspace, so the query specifies the full path to the file:

dfs.`/mapr/demo.mapr.com/data/nested/clicks/clicks.json`

## What's Next

Go to [Lesson 3: Run Queries on Complex Data Types](http://drill.apache.org/docs/lesson-3-run-queries-on-complex-data-types).

# Lesson 3: Run Queries on Complex Data Types

## Goal

This lesson focuses on queries that exercise functions and operators on self- describing data and complex data types. Drill offers intuitive SQL extensions to work with such data and offers high query performance with an architecture built from the ground up for complex data.

## Queries in This Lesson

Now that you have run ANSI SQL queries against different tables and files with relational data, you can try some examples including complex types.

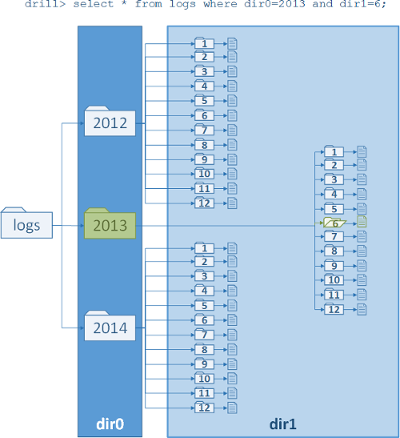
* Access directories and subdirectories of files in a single SELECT statement.
* Demonstrate simple ways to access complex data in JSON files.
* Demonstrate the repeated\_count function to aggregate values in an array.

## Query Partitioned Directories

You can use special variables in Drill to refer to subdirectories in your workspace path:

* dir0
* dir1
* …

Note that these variables are dynamically determined based on the partitioning of the file system. No up-front definitions are required on what partitions exist. Here is a visual example of how this works:



### Set workspace to dfs.logs:

0: jdbc:drill:> use dfs.logs;

+-------+---------------------------------------+

| ok | summary |

+-------+---------------------------------------+

| true | Default schema changed to [dfs.logs] |

+-------+---------------------------------------+

1 row selected

### Query logs data for a specific year:

0: jdbc:drill:> select \* from logs where dir0='2013' limit 10;

+-------+-------+-----------+-------------+-----------+----------+---------+--------+----------+-----------+----------+-------------+

| dir0 | dir1 | trans\_id | date | time | cust\_id | device | state | camp\_id | keywords | prod\_id | purch\_flag |

+-------+-------+-----------+-------------+-----------+----------+---------+--------+----------+-----------+----------+-------------+

| 2013 | 8 | 12104 | 08/29/2013 | 09:34:37 | 962 | IOS5 | ma | 3 | milhouse | 17 | false |

| 2013 | 8 | 12132 | 08/23/2013 | 01:11:25 | 4 | IOS7 | mi | 11 | hi | 439 | false |

| 2013 | 8 | 12177 | 08/14/2013 | 13:48:50 | 23 | AOS4.2 | il | 14 | give | 382 | false |

| 2013 | 8 | 12180 | 08/03/2013 | 20:48:45 | 1509 | IOS7 | ca | 0 | i'm | 340 | false |

| 2013 | 8 | 12187 | 08/16/2013 | 10:28:07 | 0 | IOS5 | ny | 16 | clicking | 11 | false |

| 2013 | 8 | 12190 | 08/10/2013 | 14:16:50 | 9 | IOS5 | va | 3 | a | 495 | false |

| 2013 | 8 | 12200 | 08/02/2013 | 20:54:38 | 42219 | IOS5 | ia | 0 | what's | 346 | false |

| 2013 | 8 | 12210 | 08/05/2013 | 20:12:24 | 8073 | IOS5 | sc | 5 | if | 33 | false |

| 2013 | 8 | 12235 | 08/28/2013 | 07:49:45 | 595 | IOS5 | tx | 2 | that | 51 | false |

| 2013 | 8 | 12239 | 08/13/2013 | 03:24:31 | 2 | IOS5 | or | 6 | haw-haw | 40 | false |

+-------+-------+-----------+-------------+-----------+----------+---------+--------+----------+-----------+----------+-------------+

10 rows selected

This query constrains files inside the subdirectory named 2013. The variable dir0 refers to the first level down from logs, dir1 to the next level, and so on. So this query returned 10 of the rows for February 2013.

### Further constrain the results using multiple predicates in the query:

This query returns a list of customer IDs for people who made a purchase via an IOS5 device in August 2013.

0: jdbc:drill:> select dir0 as yr, dir1 as mth, cust\_id from logs

where dir0='2013' and dir1='8' and device='IOS5' and purch\_flag='true'

order by `date`;

+-------+------+----------+

| yr | mth | cust\_id |

+-------+------+----------+

| 2013 | 8 | 4 |

| 2013 | 8 | 521 |

| 2013 | 8 | 1 |

| 2013 | 8 | 2 |

...

### Return monthly counts per customer for a given year:

0: jdbc:drill:> select cust\_id, dir1 month\_no, count(\*) month\_count from logs

where dir0=2014 group by cust\_id, dir1 order by cust\_id, month\_no limit 10;

+----------+-----------+--------------+

| cust\_id | month\_no | month\_count |

+----------+-----------+--------------+

| 0 | 1 | 143 |

| 0 | 2 | 118 |

| 0 | 3 | 117 |

| 0 | 4 | 115 |

| 0 | 5 | 137 |

| 0 | 6 | 117 |

| 0 | 7 | 142 |

| 0 | 8 | 19 |

| 1 | 1 | 66 |

| 1 | 2 | 59 |

+----------+-----------+--------------+

10 rows selected

This query groups the aggregate function by customer ID and month for one year: 2014.

## Query Complex Data

Drill provides some specialized operators and functions that you can use to analyze nested data natively without transformation. If you are familiar with JavaScript notation, you will already know how some of these extensions work.

### Set the workspace to dfs.clicks:

0: jdbc:drill:> use dfs.clicks;

+-------+-----------------------------------------+

| ok | summary |

+-------+-----------------------------------------+

| true | Default schema changed to [dfs.clicks] |

+-------+-----------------------------------------+

1 row selected

### Explore clickstream data:

Note that the user\_info and trans\_info columns contain nested data: arrays and arrays within arrays. The following queries show how to access this complex data.

0: jdbc:drill:> select \* from `clicks/clicks.json` limit 5;

+-----------+-------------+-----------+---------------------------------------------------+---------------------------------------------------------------------------+

| trans\_id | date | time | user\_info | trans\_info |

+-----------+-------------+-----------+---------------------------------------------------+---------------------------------------------------------------------------+

| 31920 | 2014-04-26 | 12:17:12 | {"cust\_id":22526,"device":"IOS5","state":"il"} | {"prod\_id":[174,2],"purch\_flag":"false"} |

| 31026 | 2014-04-20 | 13:50:29 | {"cust\_id":16368,"device":"AOS4.2","state":"nc"} | {"prod\_id":[],"purch\_flag":"false"} |

| 33848 | 2014-04-10 | 04:44:42 | {"cust\_id":21449,"device":"IOS6","state":"oh"} | {"prod\_id":[582],"purch\_flag":"false"} |

| 32383 | 2014-04-18 | 06:27:47 | {"cust\_id":20323,"device":"IOS5","state":"oh"} | {"prod\_id":[710,47],"purch\_flag":"false"} |

| 32359 | 2014-04-19 | 23:13:25 | {"cust\_id":15360,"device":"IOS5","state":"ca"} | {"prod\_id":[0,8,170,173,1,124,46,764,30,711,0,3,25],"purch\_flag":"true"} |

+-----------+-------------+-----------+---------------------------------------------------+---------------------------------------------------------------------------+

5 rows selected

### Unpack the user\_info column:

0: jdbc:drill:> select t.user\_info.cust\_id as custid, t.user\_info.device as device,

t.user\_info.state as state

from `clicks/clicks.json` t limit 5;

+---------+---------+--------+

| custid | device | state |

+---------+---------+--------+

| 22526 | IOS5 | il |

| 16368 | AOS4.2 | nc |

| 21449 | IOS6 | oh |

| 20323 | IOS5 | oh |

| 15360 | IOS5 | ca |

+---------+---------+--------+

5 rows selected (0.171 seconds)

This query uses a simple table.column.column notation to extract nested column data. For example:

t.user\_info.cust\_id

where t is the table alias provided in the query, user\_info is a top-level column name, and cust\_id is a nested column name.

The table alias is required; otherwise column names such as user\_info are parsed as table names by the SQL parser.

### Unpack the trans\_info column:

0: jdbc:drill:> select t.trans\_info.prod\_id as prodid, t.trans\_info.purch\_flag as

purchased

from `clicks/clicks.json` t limit 5;

+-------------------------------------------+------------+

| prodid | purchased |

+-------------------------------------------+------------+

| [174,2] | false |

| [] | false |

| [582] | false |

| [710,47] | false |

| [0,8,170,173,1,124,46,764,30,711,0,3,25] | true |

+-------------------------------------------+------------+

5 rows selected

Note that this result reveals that the prod\_id column contains an array of IDs (one or more product ID values per row, separated by commas). The next step shows how you to access this kind of data.

## Query Arrays

Now use the [n] notation, where n is the position of the value in an array, starting from position 0 (not 1) for the first value. You can use this notation to write interesting queries against nested array data.

For example:

trans\_info.prod\_id[0]

refers to the first value in the nested prod\_id column and

trans\_info.prod\_id[20]

refers to the 21st value, assuming one exists.

### Find the first product that is searched for in each transaction:

0: jdbc:drill:> select t.trans\_id, t.trans\_info.prod\_id[0] from `clicks/clicks.json` t limit 5;

+------------+------------+

| trans\_id | EXPR$1 |

+------------+------------+

| 31920 | 174 |

| 31026 | null |

| 33848 | 582 |

| 32383 | 710 |

| 32359 | 0 |

+------------+------------+

5 rows selected

### For which transactions did customers search on at least 21 products?

0: jdbc:drill:> select t.trans\_id, t.trans\_info.prod\_id[20]

from `clicks/clicks.json` t

where t.trans\_info.prod\_id[20] is not null

order by trans\_id limit 5;

+------------+------------+

| trans\_id | EXPR$1 |

+------------+------------+

| 10328 | 0 |

| 10380 | 23 |

| 10701 | 1 |

| 11100 | 0 |

| 11219 | 46 |

+------------+------------+

5 rows selected

This query returns transaction IDs and product IDs for records that contain a non-null product ID at the 21st position in the array.

### Return clicks for a specific product range:

0: jdbc:drill:> select \* from (select t.trans\_id, t.trans\_info.prod\_id[0] as prodid,

t.trans\_info.purch\_flag as purchased

from `clicks/clicks.json` t) sq

where sq.prodid between 700 and 750 and sq.purchased='true'

order by sq.prodid;

+------------+------------+------------+

| trans\_id | prodid | purchased |

+------------+------------+------------+

| 21886 | 704 | true |

| 20674 | 708 | true |

| 22158 | 709 | true |

| 34089 | 714 | true |

| 22545 | 714 | true |

| 37500 | 717 | true |

| 36595 | 718 | true |

...

This query assumes that there is some meaning to the array (that it is an ordered list of products purchased rather than a random list).

## Perform Operations on Arrays

### Rank successful click conversions and count product searches for each session:

0: jdbc:drill:> select t.trans\_id, t.`date` as session\_date, t.user\_info.cust\_id as

cust\_id, t.user\_info.device as device, repeated\_count(t.trans\_info.prod\_id) as

prod\_count, t.trans\_info.purch\_flag as purch\_flag

from `clicks/clicks.json` t

where t.trans\_info.purch\_flag = 'true' order by prod\_count desc;

+------------+--------------+------------+------------+------------+------------+

| trans\_id | session\_date | cust\_id | device | prod\_count | purch\_flag |

+------------+--------------+------------+------------+------------+------------+

| 37426 | 2014-04-06 | 18709 | IOS5 | 34 | true |

| 31589 | 2014-04-16 | 18576 | IOS6 | 31 | true |

| 11600 | 2014-04-07 | 4260 | AOS4.2 | 28 | true |

| 35074 | 2014-04-03 | 16697 | AOS4.3 | 27 | true |

| 17192 | 2014-04-22 | 2501 | AOS4.2 | 26 | true |

...

This query uses an SQL extension, the repeated\_count function, to get an aggregated count of the array values. The query returns the number of products searched for each session that converted into a purchase and ranks the counts in descending order. Only clicks that have resulted in a purchase are counted.

## Store a Result Set in a Table for Reuse and Analysis

To facilitate additional analysis on this result set, you can easily and quickly create a Drill table from the results of the query.

### Continue to use the dfs.clicks workspace

0: jdbc:drill:> use dfs.clicks;

+-------+-----------------------------------------+

| ok | summary |

+-------+-----------------------------------------+

| true | Default schema changed to [dfs.clicks] |

+-------+-----------------------------------------+

1 row selected (1.61 seconds)

### Return product searches for high-value customers:

0: jdbc:drill:> select o.cust\_id, o.order\_total, t.trans\_info.prod\_id[0] as prod\_id

from

hive.orders as o

join `clicks/clicks.json` t

on o.cust\_id=t.user\_info.cust\_id

where o.order\_total > (select avg(inord.order\_total)

from hive.orders inord

where inord.state = o.state);

+----------+--------------+----------+

| cust\_id | order\_total | prod\_id |

+----------+--------------+----------+

| 1328 | 73 | 26 |

| 1328 | 146 | 26 |

| 1328 | 56 | 26 |

| 1328 | 91 | 26 |

| 1328 | 74 | 26 |

...

+----------+--------------+----------+

107,482 rows selected (14.863 seconds)

This query returns a list of products that are being searched for by customers who have made transactions that are above the average in their states.

### Materialize the result of the previous query:

0: jdbc:drill:> create table product\_search as select o.cust\_id, o.order\_total, t.trans\_info.prod\_id[0] as prod\_id

from

hive.orders as o

join `clicks/clicks.json` t

on o.cust\_id=t.user\_info.cust\_id

where o.order\_total > (select avg(inord.order\_total)

from hive.orders inord

where inord.state = o.state);

+-----------+----------------------------+

| Fragment | Number of records written |

+-----------+----------------------------+

| 0\_0 | 107482 |

+-----------+----------------------------+

1 row selected (3.488 seconds)

This example uses a CTAS statement to create a table based on a correlated subquery that you ran previously. This table contains all of the rows that the query returns (107,482) and stores them in the format specified by the storage plugin (Parquet format in this example). You can create tables that store data in csv, parquet, and json formats.

### Query the new table to verify the row count:

This example simply checks that the CTAS statement worked by verifying the number of rows in the table.

0: jdbc:drill:> select count(\*) from product\_search;

+---------+

| EXPR$0 |

+---------+

| 107482 |

+---------+

1 row selected (0.155 seconds)

### Find the storage file for the table:

[root@maprdemo product\_search]# cd /mapr/demo.mapr.com/data/nested/product\_search

[root@maprdemo product\_search]# ls -la

total 451

drwxr-xr-x. 2 mapr mapr 1 Sep 15 13:41 .

drwxr-xr-x. 4 root root 2 Sep 15 13:41 ..

-rwxr-xr-x. 1 mapr mapr 460715 Sep 15 13:41 0\_0\_0.parquet

Note that the table is stored in a file called 0\_0\_0.parquet. This file is stored in the location defined by the dfs.clicks workspace:

"location": "http://demo.mapr.com/data/nested"

There is a subdirectory that has the same name as the table you created.

## What's Next

Complete the tutorial with the [Summary](http://drill.apache.org/docs/summary).

# Summary

This tutorial introduced Apache Drill and its ability to run ANSI SQL queries against various data sources, including Hive tables, MapR-DB/HBase tables, and file system directories. The tutorial also showed how to work with and manipulate complex and multi-structured data commonly found in Hadoop/NoSQL systems.

Now that you are familiar with different ways to access the sample data with Drill, you can try writing your own queries against your own data sources.

# Analyzing Highly Dynamic Datasets

Mar 15, 2017

Today’s data is dynamic and application-driven. The growth of a new era of business applications driven by industry trends such as web, social, mobile, and Internet of Things are generating datasets with new data types and new data models. These applications are iterative, and the associated data models typically are semi-structured, schema-less and constantly evolving. Semi-structured data models can be complex/nested, schema-less, and capable of having varying fields in every single row and of constantly evolving as fields get added and removed frequently to meet business requirements.

This tutorial shows you how to natively query dynamic datasets, such as JSON, and derive insights from any type of data in minutes. The dataset used in the example is from the Yelp check-ins dataset, which has the following structure:

check-in

{

'type': 'checkin',

'business\_id': (encrypted business id),

'checkin\_info': {

'0-0': (number of checkins from 00:00 to 01:00 on all Sundays),

'1-0': (number of checkins from 01:00 to 02:00 on all Sundays),

...

'14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),

...

'23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)

}, # if there was no checkin for a hour-day block it will not be in the dataset

}

It is worth repeating the comment at the bottom of this snippet:

# if there was no checkin for a hour-day block it will not be in the dataset.

The element names that you see in the checkin\_info are unknown upfront and can vary for every row. The data, although simple, is highly dynamic data. To analyze the data there is no need to first represent this dataset in a flattened relational structure, as you would using any other SQL on Hadoop technology.

Step 1: First download Drill, if you have not yet done so, onto your machine

http://drill.apache.org/download/

tar -xvf apache-drill-1.10.0.tar

Install Drill locally on your desktop (embedded mode). You don’t need Hadoop.

Step 2: Start the Drill shell.

bin/drill-embedded

Step 3: Start analyzing the data using SQL

First, let’s take a look at the dataset:

0: jdbc:drill:zk=local> SELECT \* FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` limit 2;

+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------------+------------------------+

| checkin\_info | type | business\_id |

+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------------+------------------------+

| {"3-4":1,"13-5":1,"6-6":1,"14-5":1,"14-6":1,"14-2":1,"14-3":1,"19-0":1,"11-5":1,"13-2":1,"11-6":2,"11-3":1,"12-6":1,"6-5":1,"5-5":1,"9-2":1,"9-5":1,"9-6":1,"5-2":1,"7-6":1,"7-5":1,"7-4":1,"17-5":1,"8-5":1,"10-2":1,"10-5":1,"10-6":1} | checkin | JwUE5GmEO-sH1FuwJgKBlQ |

| {"6-6":2,"6-5":1,"7-6":1,"7-5":1,"8-5":2,"10-5":1,"9-3":1,"12-5":1,"15-3":1,"15-5":1,"15-6":1,"16-3":1,"10-0":1,"15-4":1,"10-4":1,"8-2":1} | checkin | uGykseHzyS5xAMWoN6YUqA |

+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------------+------------------------+

**NOTE**

This document aligns Drill output for example purposes. Drill output is not aligned in this case.

You query the data in JSON files directly. Schema definitions in Hive store are not necessary. The names of the elements within the checkin\_info column are different between the first and second row.

Drill provides a function called KVGEN (Key Value Generator) which is useful when working with complex data that contains arbitrary maps consisting of dynamic and unknown element names such as checkin\_info. KVGEN turns the dynamic map into an array of key-value pairs where keys represent the dynamic element names.

Let’s apply KVGEN on the checkin\_info element to generate key-value pairs.

0: jdbc:drill:zk=local> SELECT KVGEN(checkin\_info) checkins FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` LIMIT 2;

+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| checkins |

+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| [{"key":"3-4","value":1},{"key":"13-5","value":1},{"key":"6-6","value":1},{"key":"14-5","value":1},{"key":"14-6","value":1},{"key":"14-2","value":1},{"key":"14-3","value":1},{"key":"19-0","value":1},{"key":"11-5","value":1},{"key":"13-2","value":1},{"key":"11-6","value":2},{"key":"11-3","value":1},{"key":"12-6","value":1},{"key":"6-5","value":1},{"key":"5-5","value":1},{"key":"9-2","value":1},{"key":"9-5","value":1},{"key":"9-6","value":1},{"key":"5-2","value":1},{"key":"7-6","value":1},{"key":"7-5","value":1},{"key":"7-4","value":1},{"key":"17-5","value":1},{"key":"8-5","value":1},{"key":"10-2","value":1},{"key":"10-5","value":1},{"key":"10-6","value":1}] |

| [{"key":"6-6","value":2},{"key":"6-5","value":1},{"key":"7-6","value":1},{"key":"7-5","value":1},{"key":"8-5","value":2},{"key":"10-5","value":1},{"key":"9-3","value":1},{"key":"12-5","value":1},{"key":"15-3","value":1},{"key":"15-5","value":1},{"key":"15-6","value":1},{"key":"16-3","value":1},{"key":"10-0","value":1},{"key":"15-4","value":1},{"key":"10-4","value":1},{"key":"8-2","value":1}] |

+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

Drill provides another function to operate on complex data called ‘Flatten’ to break the list of key-value pairs resulting from ‘KVGen’ into separate rows to further apply analytic functions on it.

0: jdbc:drill:zk=local> SELECT FLATTEN(KVGEN(checkin\_info)) checkins FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` LIMIT 20;

+--------------------------+

| checkins |

+--------------------------+

| {"key":"3-4","value":1} |

| {"key":"13-5","value":1} |

| {"key":"6-6","value":1} |

| {"key":"14-5","value":1} |

| {"key":"14-6","value":1} |

| {"key":"14-2","value":1} |

| {"key":"14-3","value":1} |

| {"key":"19-0","value":1} |

| {"key":"11-5","value":1} |

| {"key":"13-2","value":1} |

| {"key":"11-6","value":2} |

| {"key":"11-3","value":1} |

| {"key":"12-6","value":1} |

| {"key":"6-5","value":1} |

| {"key":"5-5","value":1} |

| {"key":"9-2","value":1} |

| {"key":"9-5","value":1} |

| {"key":"9-6","value":1} |

| {"key":"5-2","value":1} |

| {"key":"7-6","value":1} |

+--------------------------+

You can get value from the data quickly by applying both KVGEN and FLATTEN functions on the datasets on the fly--no need for time-consuming schema definitions and data storage in intermediate formats.

On the output of flattened data, you use standard SQL functionality such as filters , aggregates, and sort. Let’s see a few examples.

### Get the total number of check-ins recorded in the Yelp dataset

0: jdbc:drill:zk=local> SELECT SUM(checkintbl.checkins.`value`) AS TotalCheckins FROM (

. . . . . . . . . . . > SELECT FLATTEN(KVGEN(checkin\_info)) checkins FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` ) checkintbl

. . . . . . . . . . . > ;

+---------------+

| TotalCheckins |

+---------------+

| 4713811 |

+---------------+

### Get the number of check-ins specifically for Sunday midnights

0: jdbc:drill:zk=local> SELECT SUM(checkintbl.checkins.`value`) AS SundayMidnightCheckins FROM (

. . . . . . . . . . . > SELECT FLATTEN(KVGEN(checkin\_info)) checkins FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` ) checkintbl WHERE checkintbl.checkins.key='23-0';

+------------------------+

| SundayMidnightCheckins |

+------------------------+

| 8575 |

+------------------------+

### Get the number of check-ins per day of the week

0: jdbc:drill:zk=local> SELECT `right`(checkintbl.checkins.key,1) WeekDay,sum(checkintbl.checkins.`value`) TotalCheckins from (

. . . . . . . . . . . > select flatten(kvgen(checkin\_info)) checkins FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` ) checkintbl GROUP BY `right`(checkintbl.checkins.key,1) ORDER BY TotalCheckins;

+------------+---------------+

| WeekDay | TotalCheckins |

+------------+---------------+

| 1 | 545626 |

| 0 | 555038 |

| 2 | 555747 |

| 3 | 596296 |

| 6 | 735830 |

| 4 | 788073 |

| 5 | 937201 |

+------------+---------------+

### Get the number of check-ins per hour of the day

0: jdbc:drill:zk=local> SELECT SUBSTR(checkintbl.checkins.key,1,strpos(checkintbl.checkins.key,'-')-1) AS HourOfTheDay ,SUM(checkintbl.checkins.`value`) TotalCheckins FROM (

. . . . . . . . . . . > SELECT FLATTEN(KVGEN(checkin\_info)) checkins FROM dfs.`/users/nrentachintala/Downloads/yelp/yelp\_academic\_dataset\_checkin.json` ) checkintbl GROUP BY SUBSTR(checkintbl.checkins.key,1,strpos(checkintbl.checkins.key,'-')-1) ORDER BY TotalCheckins;

+--------------+---------------+

| HourOfTheDay | TotalCheckins |

+--------------+---------------+

| 3 | 20357 |

| 4 | 21076 |

| 2 | 28116 |

| 5 | 33842 |

| 1 | 45467 |

| 6 | 54174 |

| 0 | 74127 |

| 7 | 96329 |

| 23 | 102009 |

| 8 | 130091 |

| 22 | 140338 |

| 9 | 162913 |

| 21 | 211949 |

| 10 | 220687 |

| 15 | 261384 |

| 14 | 276188 |

| 16 | 292547 |

| 20 | 293783 |

| 13 | 328373 |

| 11 | 338675 |

| 17 | 374186 |

| 19 | 385381 |

| 12 | 399797 |

| 18 | 422022 |

+--------------+---------------+

## Summary

In this tutorial, you surf both structured and semi-structured data without any upfront schema management or ETL.

# Analyzing Social Media

This tutorial covers how to analyze Twitter data in native JSON format using Apache Drill. First, you configure an environment to stream the Twitter data filtered on keywords and languages using Apache Flume, and then you analyze the data using Drill. Finally, you run interactive reports and analysis using MicroStrategy.

## Social Media Analysis Prerequisites

* Twitter developer account
* AWS account
* A MapR node on AWS
* A MicroStrategy AWS instance

## Configuring the AWS environment

Configuring the environment on Amazon Web Services (AWS) consists of these tasks:

* Create a Twitter Dev account and register a Twitter application
* Provision a preconfigured AWS MapR node with Flume and Drill
* Provision a MicroStrategy AWS instance
* Configure MicroStrategy to run reports and analyses using Drill
* Create a Twitter Dev account and register an application

This tutorial assumes you are familiar with MicroStrategy. For information about using MicroStrategy, see the [MicroStrategy documentation](http://www.microstrategy.com/Strategy/media/downloads/products/cloud/cloud_aws-user-guide.pdf).

## Establishing a Twitter Feed and Flume Credentials

The following steps establish a Twitter feed and get Twitter credentials required by Flume to set up Twitter as a data source:

1. Go to dev.twitter.com and sign in with your Twitter account details.
2. Click **Manage Your Apps** under Tools in the page footer.
3. Click **Create New App** and fill in the form, then create the application.
4. On the **Keys and Access Tokens** tab, create an access token, and then click **Create My Access Token**. If you have read-only access, you can create the token.
5. Copy the following credentials for the Twitter App that will be used to configure Flume:
   * Consumer Key
   * Consumer Secret
   * Access Token
   * Access Token Secret

## Provision Preconfigured MapR Node on AWS

You need to provision a preconfigured MapR node on AWS named ami-4dedc47d. The AMI is already configured with Flume, Drill, and specific elements to support data streaming from Twitter and Drill query views. The AMI is publicly available under Community AMIs, has a 6GB root drive, and a 100GB data drive. Being a small node, very large volumes of data will significantly decrease the response time to Twitter data queries.

1. In AWS, launch an instance.  
   The AMI image is preconfigured to use a m2.2xlarge instance type with 4 vCPUs and 32GB of memory.
2. Select the AMI id ami-4dedc47d.
3. Make sure that the instance has been assigned an external IP address; an Elastic IP is preferred, but not essential.
4. Verify that a security group is used with open TCP and UDP ports on the node. At this time, all ports are left open on the node.
5. After provisioning and booting up the instance, reboot the node in the AWS EC2 management interface to finalize the configuration.

The node is now configured with the required Flume and Drill installation. Next, update the Flume configuration files with the required credentials and keywords.

## Update Flume Configuration Files

1. Log in as the ec2-user using the AWS credentials.
2. Switch to the mapr user on the node using su – mapr.
3. Update the Flume configuration files flume-env.sh and flume.conf in the <FLUME HOME>/conf directory using the Twitter app credentials from the first section. See the [sample files](https://github.com/mapr/mapr-demos/tree/master/drill-twitter-MSTR/flume).
4. Enter the desired keywords, separated by a comma.  
   Separate multiple keywords using a space.
5. Filter tweets for specific languages, if needed, by entering the ISO 639-1 [language codes](http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) separated by a comma. If you need no language filtering, leave the parameter blank.
6. Go to the FLUME HOME directory and, as user mapr, type screen on the command line as user mapr:
7. Start Flume by typing the following command:
8. ./bin/flume-ng agent --conf ./conf/ -f ./conf/flume.conf -Dflume.root.logger=INFO,console -n TwitterAgent
9. Enter CTRL+a to exit, followed by d to detach.  
   To go back to the screen terminal, simply enter screen –r to reattach.  
   Twitter data streams into the system.
10. Run the following command to verify volumes:
11. du –h /mapr/drill\_demo/twitter/feed.

You cannot run queries until data appears in the feed directory. Allow 20-30 minutes minimum.

## Provision a MicroStrategy AWS Instance

MicroStrategy provides an AWS instance of various sizes. It comes with a free 30-day trial for the MicroStrategy instance. AWS charges still apply for the platform and OS.

To provision the MicroStrategy node in AWS:

1. On the [MicroStrategy website](http://www.microstrategy.com/us/analytics/analytics-on-aws), click **Get started**.
2. Select some number of users, for example, select 25 users.
3. Select the AWS region. Using a MapR node and MicroStrategy instance in the same AWS region is highly recommended.
4. Click **Continue**.
5. On the Manual Launch tab, click **Launch with EC2 Console** next to the appropriate region, and select **r3.large instance**.  
   An EC2 instance of r3.large is sufficient for the 25 user version.
6. Click **Configure Instance Details**.
7. Select an appropriate network setting and zones, ideally within the same zone and network as the MapR node that you provisioned.

**IMPORTANT**

Make sure that the MicroStrategy instance has a Public IP; elastic IP is preferred but not essential.

1. Keep the default storage.
2. Assign a tag to identify the instance.
3. Select a security group that allows sufficient access to external IPs and open all ports because security is not a concern.
4. In the AWS Console, launch an instance, and when the AWS reports that the instance is running, select it, and click **Connect**.
5. Click **Get Password** to get the OS Administrator password.

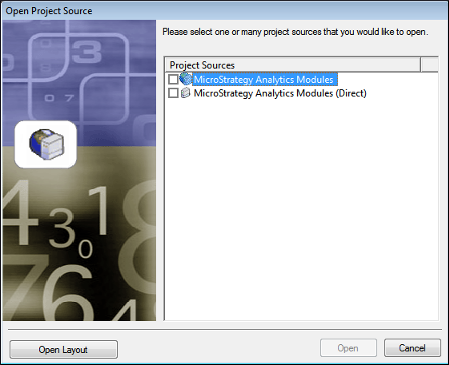
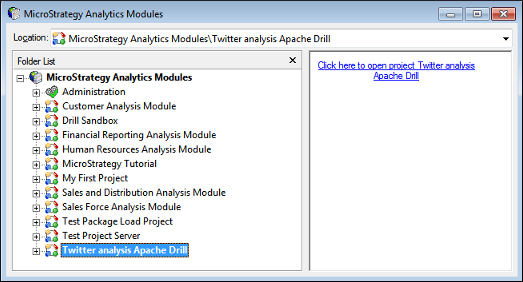
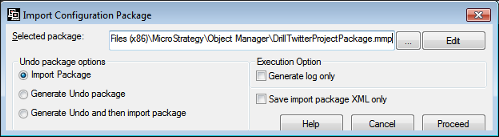
The instance is now accessible with RDP and is using the relevant AWS credentials and security.

## Configure MicroStrategy

You need to configure MicroStrategy to integrate with Drill using the ODBC driver. You install a MicroStrategy package with a number of useful, prebuilt reports for working with Twitter data. You can modify the reports or use the reports as a template to create new and more interesting reports and analysis models.

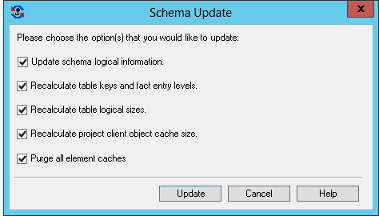
1. Configure a System DSN named Twitter with the ODBC administrator. The quick start version of the MapR ODBC driver requires the DSN.
2. [Download the quick start version of the MapR ODBC driver for Drill](http://package.mapr.com/tools/MapR-ODBC/MapR_Drill/MapRDrill_odbc_v0.08.1.0618/MapRDrillODBC32.msi).
3. [Configure the ODBC driver](http://drill.apache.org/docs/using-microstrategy-analytics-with-apache-drill) for Drill on MicroStrategy Analytics.  
   The Drill object is part of the package and doesn’t need to be configured.
4. Use the AWS Private IP if both the MapR node and the MicroStrategy instance are located in the same region (recommended).
5. Download the [Drill and Twitter configuration](https://github.com/mapr/mapr-demos/blob/master/drill-twitter-MSTR/MSTR/DrillTwitterProjectPackage.mmp) package for MicroStrategy on the Windows system using Git for Windows or the full GitHub for Windows.

## Import Reports

1. In MicroStrategy Developer, select **Schema > Create New Project** to create a new project with MicroStrategy Developer.
2. Click **Create Project** and type a name for the new project.
3. Click **OK**.  
   The Project appears in MicroStrategy Developer.
4. Open MicroStrategy Object Manager.
5. Connect to the Project Source and login as Administrator.  
   
6. In MicroStrategy Object Manager, MicroStrategy Analytics Modules, select the project for the package. For example, select **Twitter analysis Apache Drill**.  
   
7. Select **Tools > Import Configuration Package**.
8. Open the configuration package file, and click **Proceed**.  
   The package with the reports is available in MicroStrategy.

You can test and modify the reports in MicroStrategy Developer. Configure permissions if necessary.

## Update the Schema

1. In MicroStrategy Developer, select **Schema > Update Schema**.
2. In Schema Update, select all check boxes, and click **Update**.  
   

## Create a User and Set the Password

1. Expand Administration.
2. Expand User Manager, and click **Everyone**.
3. Right-click to create a new user, or click **Administrator** to edit the password.

## About the Reports

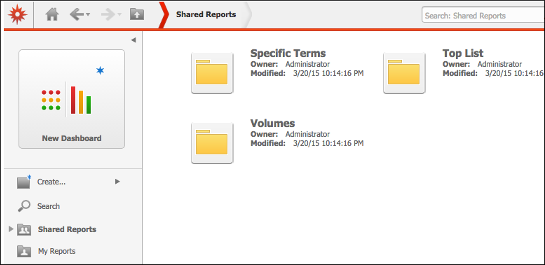
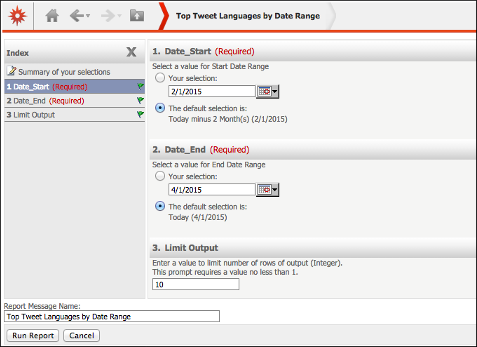
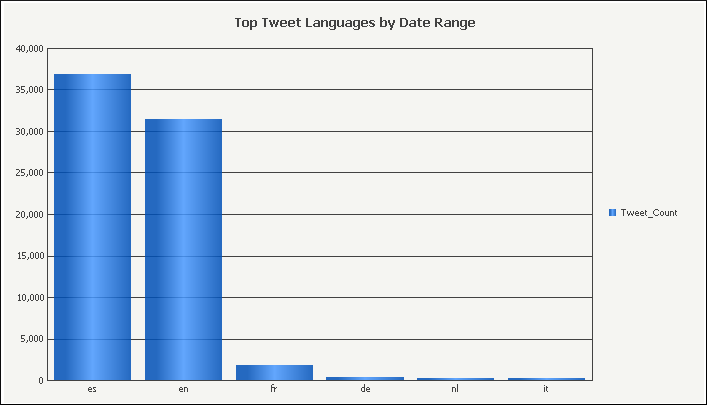
There are 18 reports in the package. Most reports prompt you to specify date ranges, output limits, and terms as needed. The package contains reports in three main categories:

* Volumes: A number of reports that show the total volume of Tweets by different date and time designations.
* Top List: Displays the top Tweets, Retweets, hashtags and users are displayed.
* Specific Terms: Tweets and Retweets that can be measured or listed based on terms in the text of the Tweet itself.

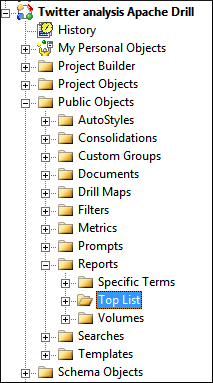
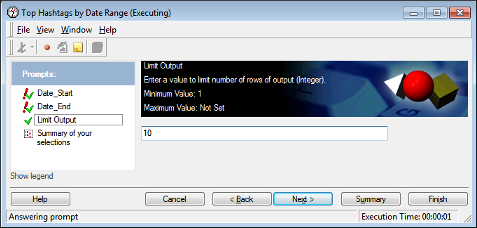
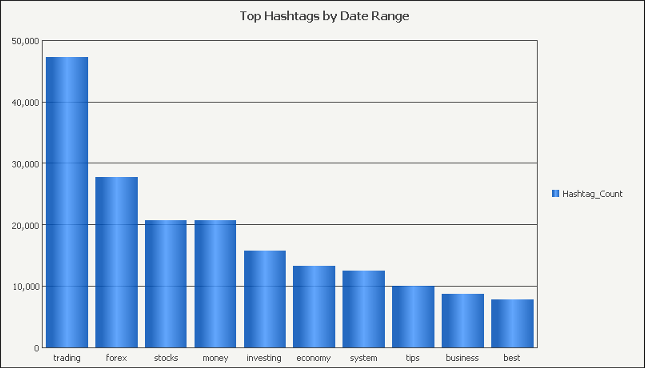
You can copy and modify the reports or use the reports as a template for querying Twitter data using Drill.

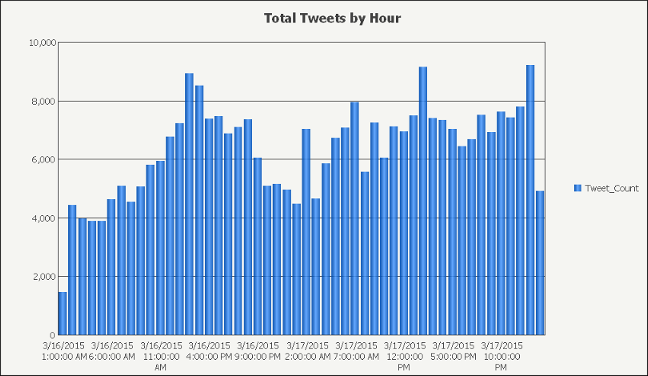
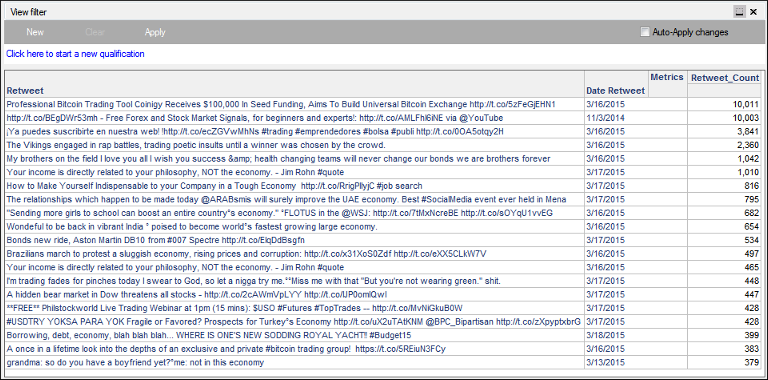
You can access reports through MicroStrategy Developer or the web interface. MicroStrategy Developer provides a more powerful interface than the web interface to modify reports or add new reports, but requires RDP access to the node.

## Using the Web Interface

1. Using a web browser, enter the URL for the web interface:  
   http:///MicroStrategy/asp/Main.aspx
2. Log in as the User you created or as Administrator, using the credentials created initially with Developer.
3. On the Welcome MicroStrategy Web User page, choose the project that was used to load the analysis package: **Drill Twitter Analysis**.  
   
4. Select **Shared Reports**.  
   The folders with the three main categories of the reports appear.
5. Select a report, and respond to any prompts. For example, to run the Top Tweet Languages by Date Range, enter the required Date\_Start and Date\_End.  
   
6. Click **Run Report**.  
   A histogram report appears showing the top tweet languages by date range.
7. To refresh the data or re-enter prompt values, select **Data > Refresh** or **Data > Re-prompt**.

## Browsing the Apache Drill Twitter Analysis Reports

The MicroStrategy Developer reports are located in the Public Objects folder of the project you chose for installing the package.  
Many of the reports require you to respond to prompts to select the desired data. For example, select the Top Hashtags report in the right-hand column. This report requires you to respond to prompts for a Start Date and End Date to specify the date range for data of interest; by default, data for the last two months, ending with the current date is selected. You can also specify the limit for the number of Top Hashtags to be returned; the default is the top 10 hashtags.  
When you click **Finish** a bar chart report with the hashtag and number of times it appeared in the specified data range appears.  


Other reports are available in the bundle. For example, this report shows total tweets by hour:This report shows top Retweets for a date range with original Tweet date and count in the date range.  


## Summary

In this tutorial, you learned how to configure an environment to stream Twitter data using Apache Flume. You then learned how to analyze the data in native JSON format with SQL using Apache Drill, and how to run interactive reports and analysis using MicroStrategy.

# Analyzing Data Using Window Functions

This tutorial briefly introduces the analytics in Drill 1.2, namely ANSI SQL-compliant analytic and window functions. Drill supports the following SQL window functions:

* PARTITION BY and OVER clauses
* A variety of aggregated window functions for Sum, Max, Min, Count, Avg
* Analytic functions such as First\_Value, Last\_Value, Lead, Lag, NTile, Row\_Number, and Rank

Window functions are highly versatile. You can reduce the joins, subqueries, and explicit cursors that you need to write. Window functions solve a variety of use cases with minimal coding effort.

This tutorial builds on previous tutorials, [Analyzing the Yelp Academic Dataset](http://drill.apache.org/docs/analyzing-the-yelp-academic-dataset/) and [Analyzing Highly Dynamic Datasets](http://drill.apache.org/docs/analyzing-highly-dynamic-datasets/), and uses the same Yelp dataset.

## Getting Started

1. To get started, download the [Yelp](http://www.yelp.com/dataset_challenge) (business reviews) now.
2. [Install and start Drill](http://drill.apache.org/docs/analyzing-the-yelp-academic-dataset/#installing-and-starting-drill).
3. List the available schemas in Drill.
4. SHOW schemas;
5. +---------------------+
6. | SCHEMA\_NAME |
7. +---------------------+
8. | INFORMATION\_SCHEMA |
9. | cp.default |
10. | dfs.default |
11. | dfs.root |
12. | dfs.tmp |
13. | dfs.yelp |
14. | sys |
15. +---------------------+
16. 7 rows selected (1.755 seconds)
17. Switch to using the workspace in which Yelp data is loaded.
18. USE dfs.yelp;
19. +-------+---------------------------------------+
20. | ok | summary |
21. +-------+---------------------------------------+
22. | true | Default schema changed to [dfs.yelp] |
23. +-------+---------------------------------------+
24. 1 row selected (0.129 seconds)
25. Start with exploring one of the datasets available in Yelp dataset - the business information.
26. SELECT \* FROM `business.json` LIMIT 1;
27. +------------------------+-----------------------------------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------+--------------------------------+---------+--------------+-------------------+-------------+-------+-------+-----------+-----------------------------------------------------------------------------------------------------------------------------------------------------+----------+---------------+
28. | business\_id | full\_address | hours | open | categories | city | review\_count | name | longitude | state | stars | latitude | attributes | type | neighborhoods |
29. +------------------------+--------------+------+-------------------------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------+--------------------------------+---------+--------------+-------------------+-------------+-------+-------+-----------+-----------------------------------------------------------------------------------------------------------------------------------------------------+----------+---------------+
30. | vcNAWiLM4dR7D2nwwJ7nCA | 4840 E Indian School Rd Ste 101 Phoenix, AZ 85018 | {"Tuesday":{"close":"17:00","open":"08:00"},"Friday":{"close":"17:00","open":"08:00"},"Monday":{"close":"17:00","open":"08:00"},"Wednesday":{"close":"17:00","open":"08:00"},"Thursday":{"close":"17:00","open":"08:00"},"Sunday":{},"Saturday":{}} | true | ["Doctors","Health & Medical"] | Phoenix | 7 | Eric Goldberg, MD | -111.983758 | AZ | 3.5 | 33.499313 | {"By Appointment Only":true,"Good Ambience":{},"Parking":{},"Music":{},"Hair Types Specialized In":{},"Payment Types":{},"Dietary Restrictions":{}} | business | [] |
31. +-------------+--------------+-------+------+------------+------+--------------+-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------+--------------------------------+---------+--------------+-------------------+-------------+-------+-------+-----------+-----------------------------------------------------------------------------------------------------------------------------------------------------+----------+---------------+
32. 1 row selected (0.514 seconds)

## Use Window Functions for Simple Queries

1. Get the top Yelp businesses based on the number reviews in each city and the row number of the business.
2. SELECT name, city, review\_count, row\_number()
3. OVER (PARTITION BY city ORDER BY review\_count DESC) AS rownum
4. FROM `business.json` LIMIT 15;
5. +----------------------------------------+------------+---------------+---------+
6. | name | city | review\_count | rownum |
7. +----------------------------------------+------------+---------------+---------+
8. | Cupz N' Crepes | Ahwatukee | 124 | 1 |
9. | My Wine Cellar | Ahwatukee | 98 | 2 |
10. | Kathy's Alterations | Ahwatukee | 12 | 3 |
11. | McDonald's | Ahwatukee | 7 | 4 |
12. | U-Haul | Ahwatukee | 5 | 5 |
13. | Hi-Health | Ahwatukee | 4 | 6 |
14. | Healthy and Clean Living Environments | Ahwatukee | 4 | 7 |
15. | Active Kids Pediatrics | Ahwatukee | 4 | 8 |
16. | Roberto's Authentic Mexican Food | Anthem | 117 | 1 |
17. | Q to U BBQ | Anthem | 74 | 2 |
18. | Outlets At Anthem | Anthem | 64 | 3 |
19. | Dara Thai | Anthem | 56 | 4 |
20. | Cafe Provence | Anthem | 53 | 5 |
21. | Shanghai Club | Anthem | 50 | 6 |
22. | Two Brothers Kitchen | Anthem | 43 | 7 |
23. +----------------------------------------+------------+---------------+---------+
24. 15 rows selected (0.67 seconds)
25. Check the number reviews for each business compared to the average number of reviews across all business in the city.
26. SELECT name, city,review\_count,
27. Avg(review\_count) OVER (PARTITION BY City) AS city\_reviews\_avg
28. FROM `business.json` LIMIT 15;
29. +----------------------------------------+------------+---------------+---------------------+
30. | name | city | review\_count | city\_reviews\_avg |
31. +----------------------------------------+------------+---------------+---------------------+
32. | Hi-Health | Ahwatukee | 4 | 32.25 |
33. | My Wine Cellar | Ahwatukee | 98 | 32.25 |
34. | U-Haul | Ahwatukee | 5 | 32.25 |
35. | Cupz N' Crepes | Ahwatukee | 124 | 32.25 |
36. | McDonald's | Ahwatukee | 7 | 32.25 |
37. | Kathy's Alterations | Ahwatukee | 12 | 32.25 |
38. | Healthy and Clean Living Environments | Ahwatukee | 4 | 32.25 |
39. | Active Kids Pediatrics | Ahwatukee | 4 | 32.25 |
40. | Anthem Community Center | Anthem | 4 | 14.492063492063492 |
41. | Scrapbooks To Remember | Anthem | 4 | 14.492063492063492 |
42. | Hungry Howie's Pizza | Anthem | 7 | 14.492063492063492 |
43. | Pinata Nueva | Anthem | 3 | 14.492063492063492 |
44. | Starbucks Coffee Company | Anthem | 13 | 14.492063492063492 |
45. | Pizza Hut | Anthem | 6 | 14.492063492063492 |
46. | Rays Pizza | Anthem | 19 | 14.492063492063492 |
47. +----------------------------------------+------------+---------------+---------------------+
48. 15 rows selected (0.395 seconds)
49. Check how the number of reviews for each business contribute to the total number of reviews for all businesses in the city.
50. SELECT name, city,review\_count,
51. Sum(review\_count) OVER (PARTITION BY City) AS city\_reviews\_sum
52. FROM `business.json`limit 15;
53. +----------------------------------------+------------+---------------+-------------------+
54. | name | city | review\_count | city\_reviews\_sum |
55. +----------------------------------------+------------+---------------+-------------------+
56. | Hi-Health | Ahwatukee | 4 | 258 |
57. | My Wine Cellar | Ahwatukee | 98 | 258 |
58. | U-Haul | Ahwatukee | 5 | 258 |
59. | Cupz N' Crepes | Ahwatukee | 124 | 258 |
60. | McDonald's | Ahwatukee | 7 | 258 |
61. | Kathy's Alterations | Ahwatukee | 12 | 258 |
62. | Healthy and Clean Living Environments | Ahwatukee | 4 | 258 |
63. | Active Kids Pediatrics | Ahwatukee | 4 | 258 |
64. | Anthem Community Center | Anthem | 4 | 913 |
65. | Scrapbooks To Remember | Anthem | 4 | 913 |
66. | Hungry Howie's Pizza | Anthem | 7 | 913 |
67. | Pinata Nueva | Anthem | 3 | 913 |
68. | Starbucks Coffee Company | Anthem | 13 | 913 |
69. | Pizza Hut | Anthem | 6 | 913 |
70. | Rays Pizza | Anthem | 19 | 913 |
71. +----------------------------------------+------------+---------------+-------------------+
72. 15 rows selected (0.543 seconds)

## Use Window Functions for Complex Queries

1. List Top 10 cities and their highest ranked businesses in terms of number of reviews. Use Drill window functions such as rank, dense\_rank in these queries.
2. WITH X
3. AS
4. (SELECT name, city, review\_count,
5. RANK()
6. OVER (PARTITION BY city
7. ORDER BY review\_count DESC) AS review\_rank
8. FROM `business.json`)
9. SELECT X.name, X.city, X.review\_count
10. FROM X
11. WHERE X.review\_rank =1 ORDER BY review\_count DESC LIMIT 10;
12. +-------------------------------------------+-------------+---------------+
13. | name | city | review\_count |
14. +-------------------------------------------+-------------+---------------+
15. | Mon Ami Gabi | Las Vegas | 4084 |
16. | Studio B | Henderson | 1336 |
17. | Phoenix Sky Harbor International Airport | Phoenix | 1325 |
18. | Four Peaks Brewing Co | Tempe | 1110 |
19. | The Mission | Scottsdale | 783 |
20. | Joe's Farm Grill | Gilbert | 770 |
21. | The Old Fashioned | Madison | 619 |
22. | Cornish Pasty Company | Mesa | 578 |
23. | SanTan Brewing Company | Chandler | 469 |
24. | Yard House | Glendale | 321 |
25. +-------------------------------------------+-------------+---------------+
26. 10 rows selected (0.49 seconds)
27. Compare the number of reviews for each business with the top and bottom review counts in the city.
28. SELECT name, city, review\_count,
29. FIRST\_VALUE(review\_count)
30. OVER(PARTITION BY city ORDER BY review\_count DESC) AS top\_review\_count,
31. LAST\_VALUE(review\_count)
32. OVER(PARTITION BY city ORDER BY review\_count DESC) AS bottom\_review\_count
33. FROM `business.json` limit 15;
34. +----------------------------------------+------------+---------------+-------------------+----------------------+
35. | name | city | review\_count | top\_review\_count | bottom\_review\_count |
36. +----------------------------------------+------------+---------------+-------------------+----------------------+
37. | My Wine Cellar | Ahwatukee | 98 | 124 | 12 |
38. | McDonald's | Ahwatukee | 7 | 124 | 12 |
39. | U-Haul | Ahwatukee | 5 | 124 | 12 |
40. | Hi-Health | Ahwatukee | 4 | 124 | 12 |
41. | Healthy and Clean Living Environments | Ahwatukee | 4 | 124 | 12 |
42. | Active Kids Pediatrics | Ahwatukee | 4 | 124 | 12 |
43. | Cupz N' Crepes | Ahwatukee | 124 | 124 | 12 |
44. | Kathy's Alterations | Ahwatukee | 12 | 124 | 12 |
45. | Q to U BBQ | Anthem | 74 | 117 | 117 |
46. | Dara Thai | Anthem | 56 | 117 | 117 |
47. | Cafe Provence | Anthem | 53 | 117 | 117 |
48. | Shanghai Club | Anthem | 50 | 117 | 117 |
49. | Two Brothers Kitchen | Anthem | 43 | 117 | 117 |
50. | The Tennessee Grill | Anthem | 32 | 117 | 117 |
51. | Dollyrockers Boutique and Salon | Anthem | 30 | 117 | 117 |
52. +----------------------------------------+------------+---------------+-------------------+----------------------+
53. 15 rows selected (0.516 seconds)
54. Compare the number of reviews with the number of reviews for the previous and following businesses.
55. SELECT city, review\_count, name,
56. LAG(review\_count, 1) OVER(PARTITION BY city ORDER BY review\_count DESC)
57. AS preceding\_count,
58. LEAD(review\_count, 1) OVER(PARTITION BY city ORDER BY review\_count DESC)
59. AS following\_count
60. FROM `business.json` limit 15;
61. +------------+---------------+----------------------------------------+------------------+------------------+
62. | city | review\_count | name | preceding\_count | following\_count |
63. +------------+---------------+----------------------------------------+------------------+------------------+
64. | Ahwatukee | 124 | Cupz N' Crepes | null | 98 |
65. | Ahwatukee | 98 | My Wine Cellar | 124 | 12 |
66. | Ahwatukee | 12 | Kathy's Alterations | 98 | 7 |
67. | Ahwatukee | 7 | McDonald's | 12 | 5 |
68. | Ahwatukee | 5 | U-Haul | 7 | 4 |
69. | Ahwatukee | 4 | Hi-Health | 5 | 4 |
70. | Ahwatukee | 4 | Healthy and Clean Living Environments | 4 | 4 |
71. | Ahwatukee | 4 | Active Kids Pediatrics | 4 | null |
72. | Anthem | 117 | Roberto's Authentic Mexican Food | null | 74 |
73. | Anthem | 74 | Q to U BBQ | 117 | 64 |
74. | Anthem | 64 | Outlets At Anthem | 74 | 56 |
75. | Anthem | 56 | Dara Thai | 64 | 53 |
76. | Anthem | 53 | Cafe Provence | 56 | 50 |
77. | Anthem | 50 | Shanghai Club | 53 | 43 |
78. | Anthem | 43 | Two Brothers Kitchen | 50 | 32 |
79. +------------+---------------+----------------------------------------+------------------+------------------+

15 rows selected (0.518 seconds)