Making big moves in Big Data with Hadoop, Hive, Parquet, Hue and Docker

Jump and run in this brief introduction to Big Data

Nikolay Dimolarov · Follow

Published in Towards Data Science

10 min read · Jan 25, 2020

What data at most big companies in 2020 looks like. Seriously.

The goal of this article is to introduce you to some key concepts in the buzzword realm of Big Data. After reading this article — potentially with some additional googling — you should be able to (more or less) understand how this whole Hadoop thing works.

So, to be more precise in this article you will:

- Learn a lot of definitions (yay)
- Spin up a Hadoop cluster with a few bells and whistles via docker-compose.
- Understand Parquet files and how to convert your csv datasets into Parquet files.
- Run SQL (technically HiveQL but it is very similar) queries on your Parquet files like it's nobody's business with Hive.
- Also, be expected to have some basic knowledge in Docker & docker-compose, running Python scripts etc. — nothing crazy but it is better if you know in advance.

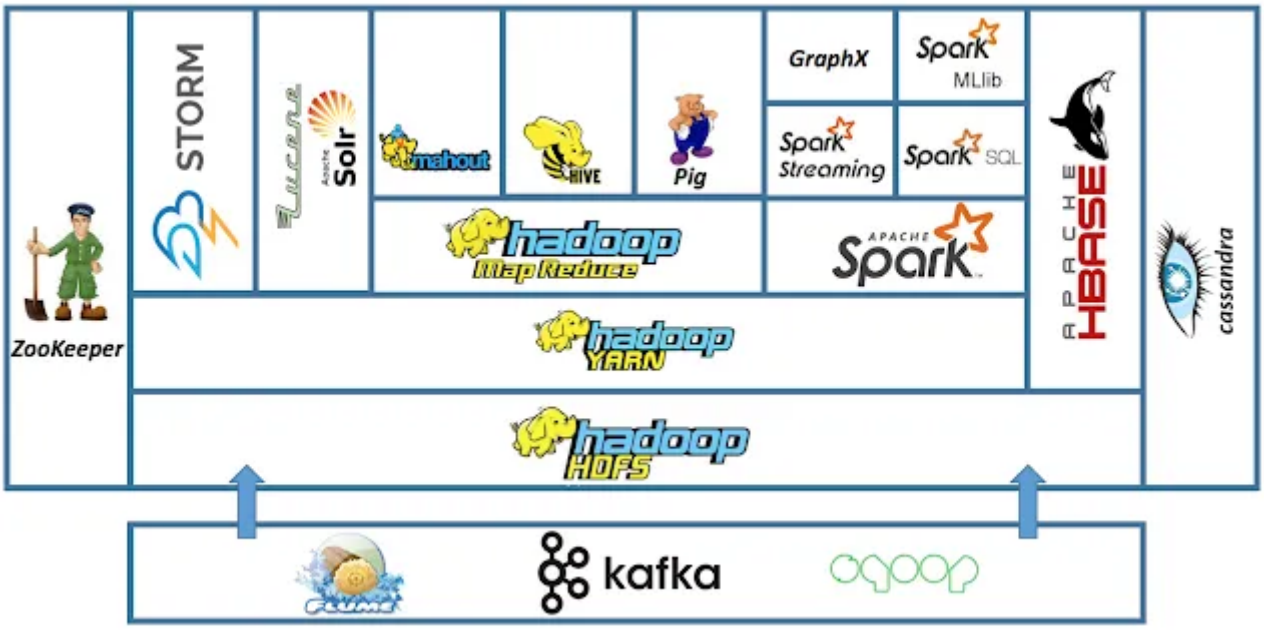
If you are already using an alternative to Hadoop's ecosystem — cool — this article is more geared towards readers that have to get familiar with Hadoop due to work, university etc. and is just one "solution" to the Big Data conundrum with its pros and cons respectively.

Big Data: this is the big one. *Big data usually includes data sets with sizes beyond the ability of commonly used software tools to capture, curate, manage, and process data within a tolerable elapsed time.* Now you might ask the simple question (and a lot of people have): "How big is big data?" Well, to be frank that is a very hard question and depending on how fast technology moves, what one considers "big" data today might be "small" data tomorrow. Nonetheless, the definition above is pretty timeless since it refers to sizes beyond the ability of commonly used tools — this is your reference line; so in 2020 let's bite the bullet and say the following is true: when you start dealing with double digit TB datasets in your DBs and above, you are probably hitting the limits of some of the more run-of-the-mill tools and it is maybe time to look into distributed computing and potentially this article.

Hadoop: a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local

computation and storage. The 3 most important core modules you should know about are:

- (Storage) HDFS: a distributed file system with high-throughput access and redundancy (copies of all files are maintained across the cluster) at the same time
- (Resource Management) YARN a.k.a. Yet Another Ressource Negotiator: a framework for job scheduling and cluster resources management e.g. which nodes are available etc.
- (Processing) MapReduce: a YARN-based system for parallel processing of large datasets. This is the main algorithm used to seamlessly distribute computational tasks across the nodes of the cluster. You can read upon the origins of MapReduce around the Web. Popular alternatives are TEZ and Spark, which were developed later for processing data even more efficiently.



Parts of the Hadoop Ecosystem in one diagram. Focus on HDFS, YARN, MapReduce and Hive for now.

Hive: a data warehouse software that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive. So, basically Hive sits on top of the aforementioned Hadoop stack and it allows you to directly use SQL on your cluster.

Hue: an open source SQL Assistant for Databases & Data Warehouses i.e. an easy GUI for looking into HDFS, Hive etc. Very handy for beginners! It is maintained by

Cloudera and you can find it on GitHub.

Parquet: a columnar storage* format available to any project in the Hadoop ecosystem. You can read why this is a good idea with big data sets in the explanation below. Again, there are a lot of alternatives but this technology is free, open-source and widely used in production across the industry.

*columnar storage: in normal row-based DBs e.g. MySQL you are storing data in rows, which are then distributed across different blocks if you cannot fit your data on one block. Now if you have a lot of columns and rows in your data distributed across multiple blocks, things can get pretty slow. Which is why you could instead store each column in a separate block. In that case you can access *all* the data of a column by just accessing one block. There is a great longer explanation on the concept here. As AWS puts it (unrelated to Parquet but still true): *"Columnar storage for database tables is an important factor in optimizing analytic query performance because it drastically reduces the overall disk I/O requirements and reduces the amount of data you need to load from disk."* Here is also another comparison to CSV, which shows how much storage you can save and what kind of speedups you can expect.

Just a great Jaws reference and this is here solely to brighten your mood if the article is already too much.

Building something

Ok. Time to build something! Namely a Hadoop cluster with Hive on top of it to run SQL queries on Parquet files stored in HDFS all the while visualizing everything in Hue. Does this sentence make more sense now than in the beginning of the article? Cool.

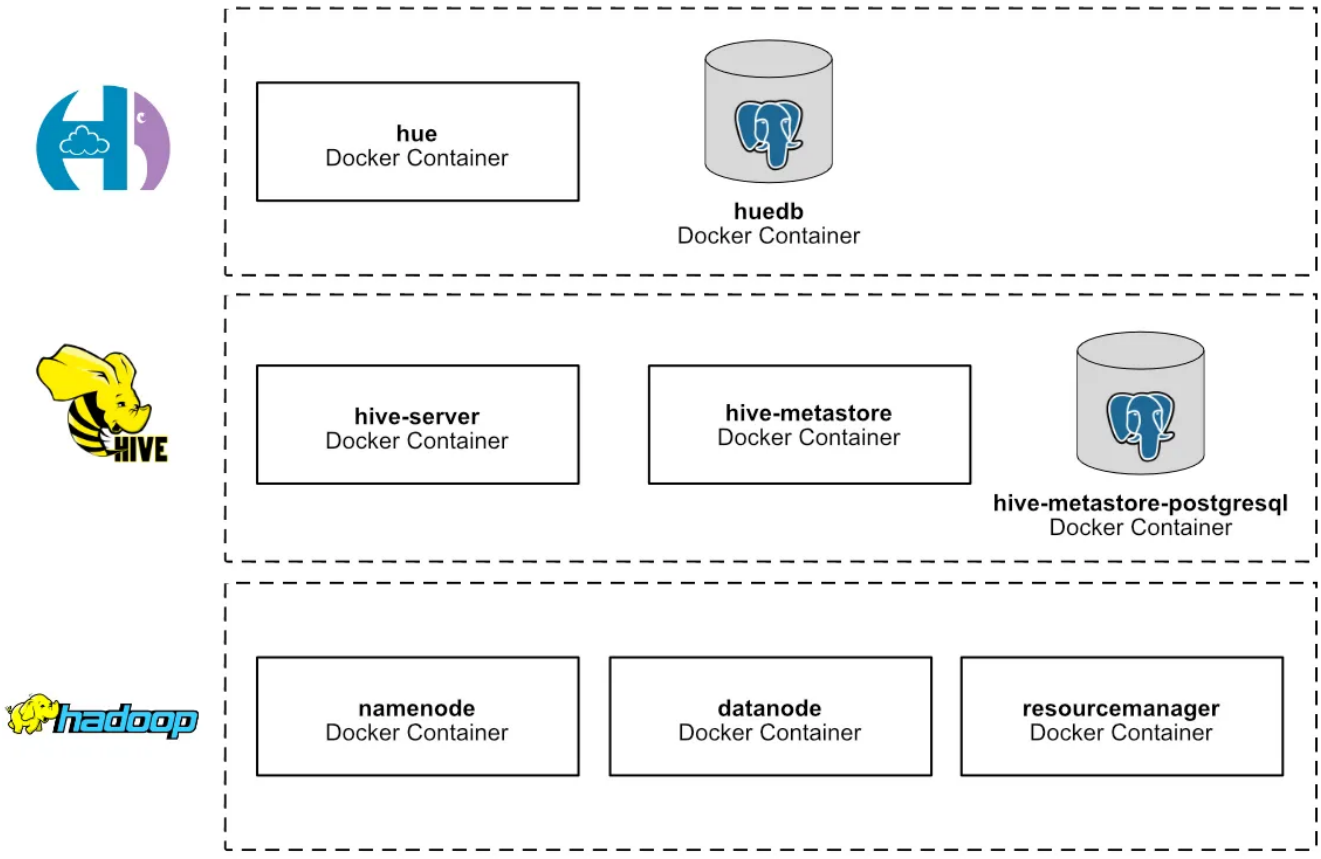
There are a lot of ways to do this and Hadoop is famous for operating computer clusters built from “commodity hardware” but since this is just a learning exercise, it is a bit easier to quickly spin up a small Hadoop cluster with the aforementioned stack in Docker with docker-compose — you can of course do this in Kubernetes too but it is way beyond the scope of this article. This setup here will not be even close to production but then again this article should merely serve as a gateway for your big data journey.

Here is the repo for this article:

tech4242/docker-hadoop-hive-parquet

This project will showcase how to spin up a Hadoop cluster with Hive in order to run SQL queries on Parquet files...

github.com



A nicer view of the docker-compose file in the repository, which should roughly sketch out the architecture of the project.

Some highlights from the implementation in Docker

This was not quite as straightforward as initially thought, so here are some pointers from the development in case you want to customize this further. The focus of this article is not to give you a crash course in Docker and docker-compose either, so this section is brief and only highlights some places where you could get stuck.

- If you want anything to work with Hue you need to figure out to override Hue's default configs by mounting the hue-overrides.ini file (you will find it in the repo and the override in docker-compose). Obvious right? Wink, wink.
- In hue-overrides.ini you should be looking at: [[database]] => this is the internal Hue DB, [[hdfs_clusters]] => connecting to HDFS to view files in Hue, [[yarn_clusters]] => setting up YARN and [beeswax] => connecting to Hive to run SQL queries in Hue.
- If you do not have this line *thrift_version=7* in hue-overrides.ini, Hue will refuse to connect to the Hive (=Thrift) Server because it is defaulting to a Hive Server version that is too high. This took hours.
- If you use Hue's default SQLite DB, you will get the message "database locked" when you try to connect to Hive => that is why there is a db-hue Postgres DB in the docker-compose file. Something about SQLite not being suitable for a multi-threaded environment as described here. Cloudera should work on their error messages...
- POSTGRES_DB, POSTGRES_USER, POSTGRES_PASSWORD in hadoop-hive.env can be used with the official postgres Docker image to directly create the DB user when you spin up your container. Check.
- Watch out with your 5432 port and not exposing it multiple times since PGDB is running more than once for this project (once as a metastore for Hive and once as a DB for hue)

tl;dr on the next steps

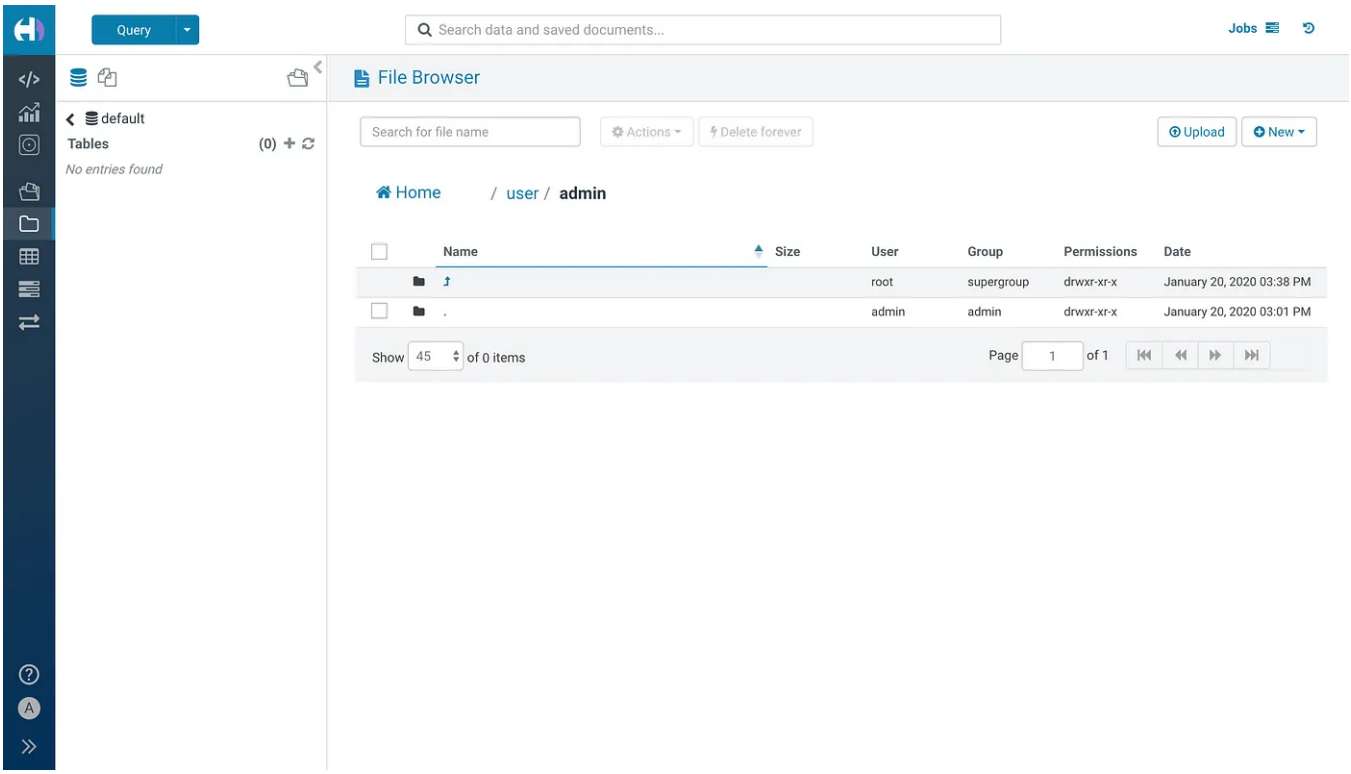
Ok. Short summary on what will happen next for the impatient engineers:

1. Start Hue, Hive and your Hadoop nodes with docker-compose up
2. Download a .csv dataset from Kaggle & convert it with the supplied Python script

3. Import said Parquet file to HDFS via Hue & preview it to make sure things are OK
4. Create empty Hive table with the Parquet file schema after inspecting the schema with the parquet-tools CLI tool
5. Import file from HDFS into Hive's table
6. Run some SQL queries!

Starting the cluster and launching Hue with docker-compose

Well, since everything is already setup, just clone the repository on your computer and type docker-compose up in your terminal. That's it. Then go to localhost:8888 and you should (after setting up the initial password for Hue) see this screen:



This screen shows you your Hadoop cluster's HDFS while the sidebar shows you the DB tables in Hive's metastore — both of which are empty in this case.

Uploading Parquet files on HDFS and previewing them in Hue

When trying to open some (quite a few) Parquet files in Hue you are going to get the following error message:

"Failed to read Parquet file"

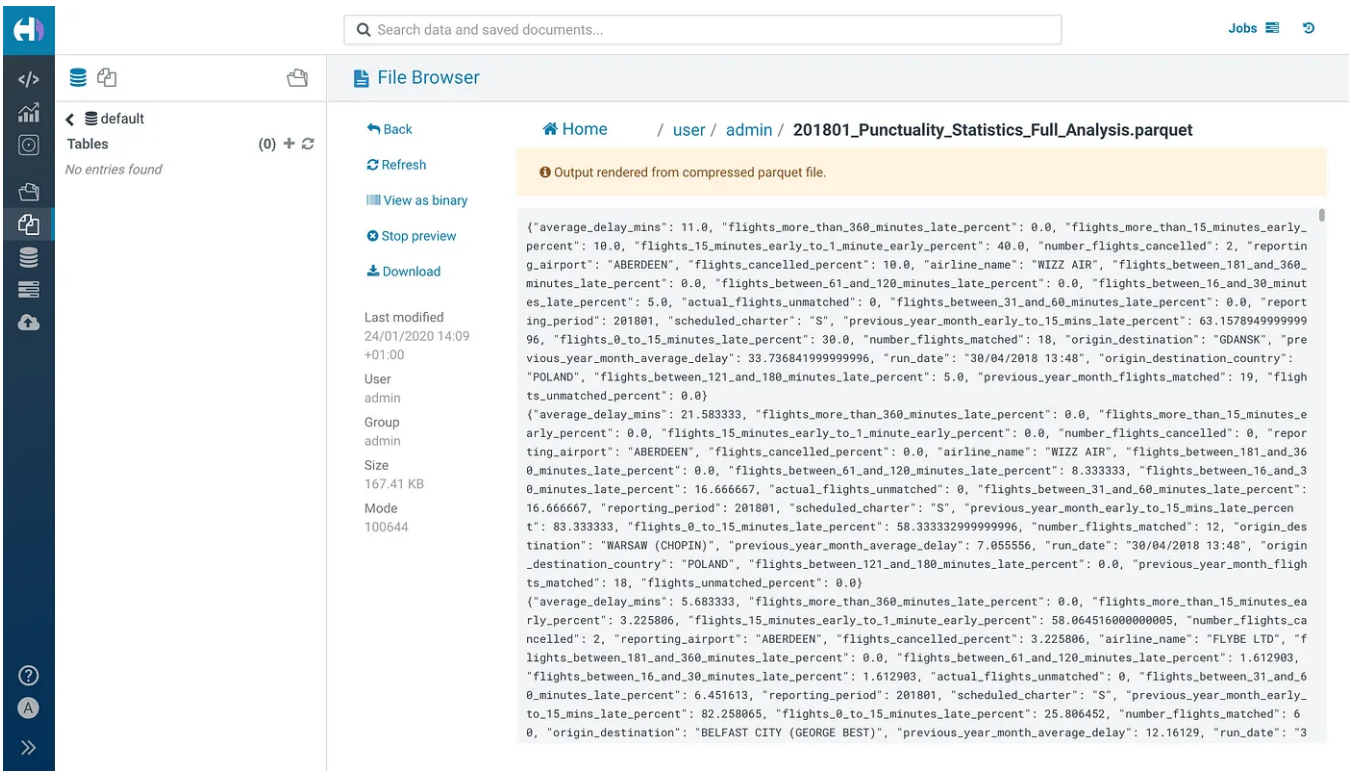
and in your docker-compose logs:

NameError: global name 'snappy' is not defined

Turns out that Hue does not support the snappy compression that is the default for a lot of Parquet converting tools like pandas. There is no workaround for this except for recreating your Parquet files (if they are using snappy). Worst UX ever on Cloudera's side...

In the GitHub repository you will find a parquet_converter.py, which uses pandas and specifies the compression as None, so one does not default to snappy that will subsequently breaking Hue. Meaning you can take any dataset from e.g. Kaggle in .csv format and convert it to Parquet with the provided Python module.

At this point — if you are unfraid of the CLI — the best suggestion is for you to forget Hue and just use Hive and HDFS directly for your Parquet files. But if you stuck around with Hue like me you could see a UK Punctuality Statistics report from Kaggle that was converted with the Python script mentioned above and then uploaded as a file:



The File Browser in Hue when you click on the successfully imported Parquet file. You can access the File Browser from the dark sidebar on the left.

Creating a Hive table from your Parquet file and schema

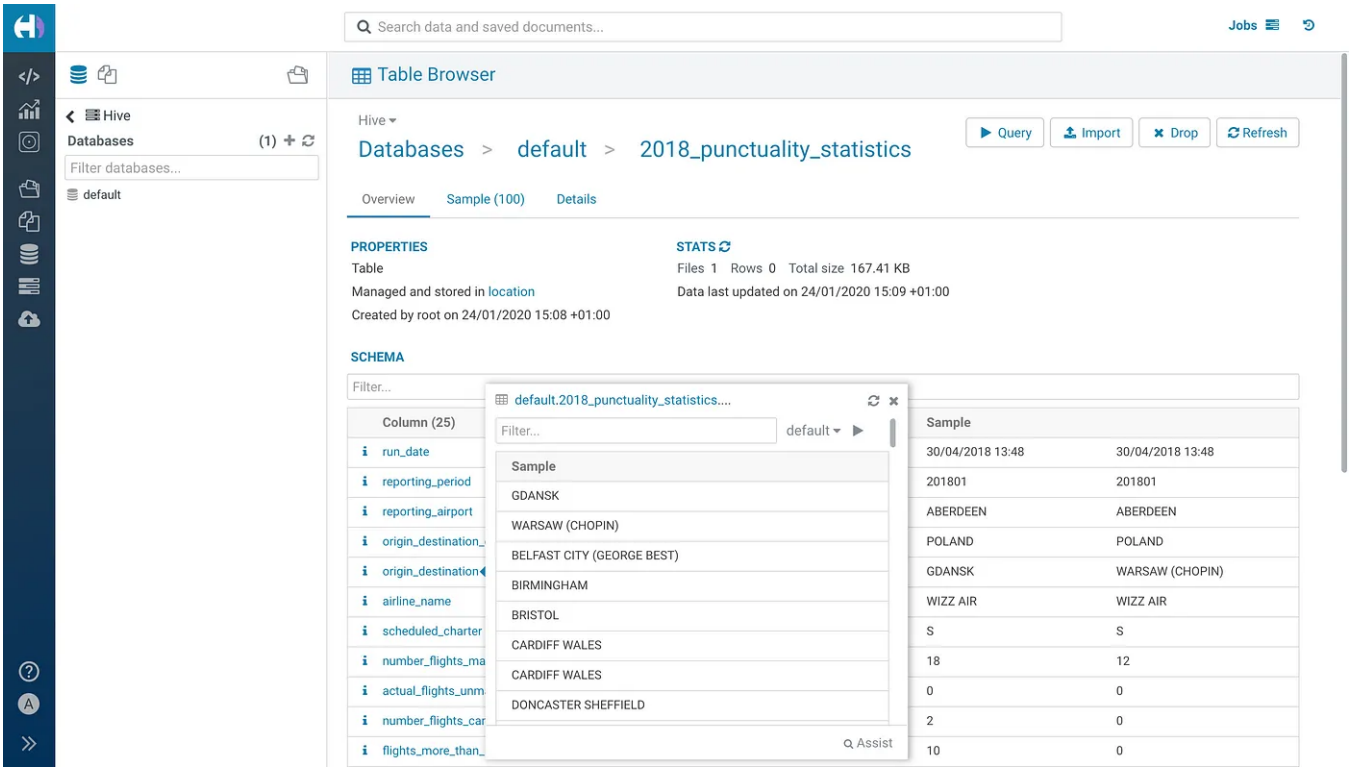
After seeing that your data was properly imported, you can create your Hive table. For this you should run the following command in your command line in the folder where you converted your file (probably /your_github_clone/data):

parquet-tools schema 201801_Punctuality_Statistics_Full_Analysis.parquet

This will output the schema you need to create the table in Hue (UTF8 = string in Hue):

message schema {
optional binary run_date (UTF8);
optional int64 reporting_period;
optional binary reporting_airport (UTF8);
optional binary origin_destination_country (UTF8);

Time to create your table:



The preview of the new table you created. Go the DB icon in the dark sidebar and create a new table manually with the schema as described above. Afterwards click on the import button in the dark sidebar and import your Parquet file into the empty table. Afterwards you should see the above screen.

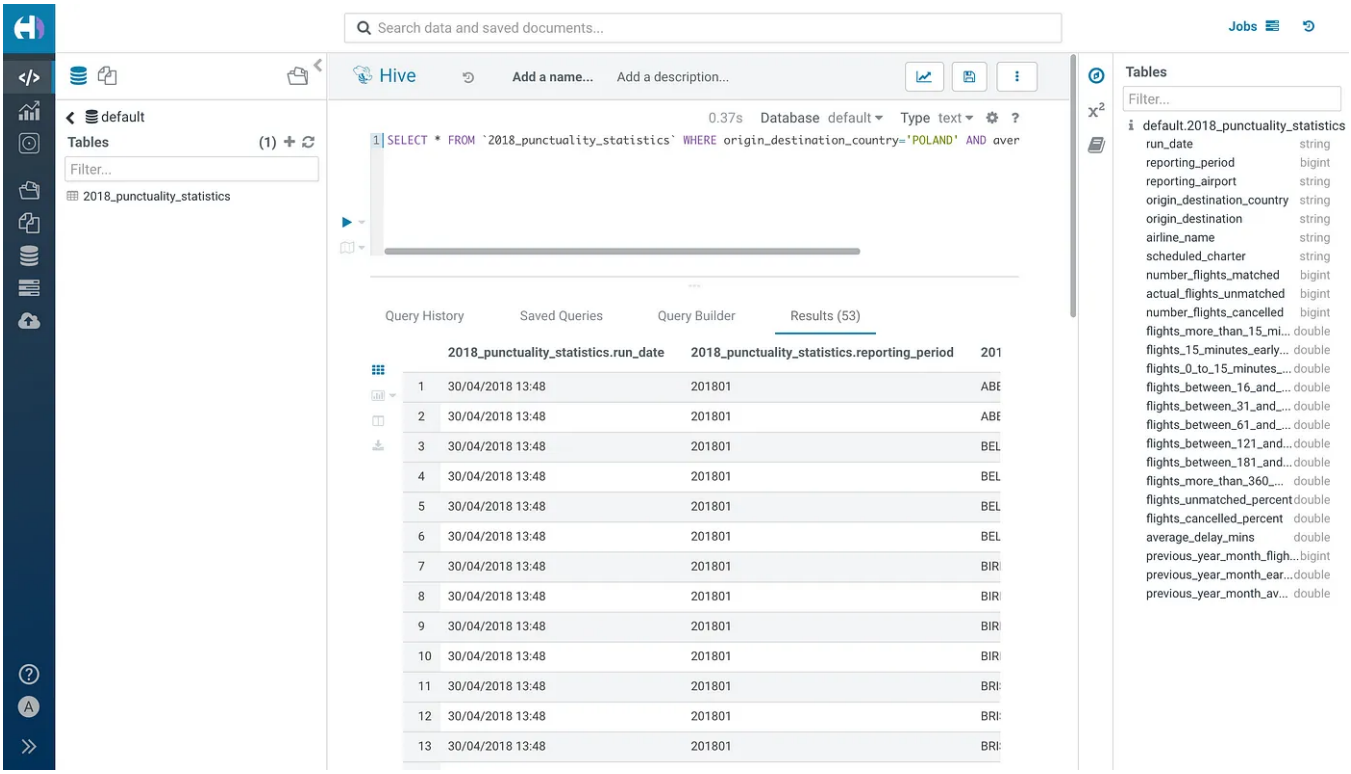
Running SQL queries

Running SQL queries was promised and it shall be delivered. The first icon in Hue'S sidebar is its query editor.

What if you wanted to find out all the flights from Poland that had an average delay of more than 10 minutes?

SELECT * FROM `2018_punctuality_statistics` WHERE origin_destination_country='POLAND' AND average_delay_mins>=10;

The auto-complete feature in the editor is fantastic, so even if you are a SQL novice, you should be able to play around with your data without too much effort.



Finally.

Time to let go

Dear reader, sadly you have reached the end of this article. Please write in the comments below if you feel like this journey should have a sequel. So, to recap you learned how to run a Hadoop cluster with Hive for running SQL queries all the while visualizing everything in Hue with docker-compose. Not bad.

This is of course a very *very* simplified look into what is possible with Hadoop but chances are that you are just starting out in the field, so give yourself some time and build on this knowledge and infrastructure. Furthermore, there are great online courses out there that you can look into next.

Looking into 2020 and beyond

Now, if you have been sort of listening in on Hadoop's ecosystem over the last few years you would have seen that the two biggest players on the market — Cloudera and Hortonworks — merged around a year ago amid a slow down of the Hadoop big data market. Add the fact that people seem way more interested in Kubernetes than in older Hadoop specific technologies like YARN for resource management and orchestration, the fast adoption of DL frameworks like PyTorch and you sort of have a perfect storm forming for the ageing Hadoop stack. Nonetheless projects like Apache Spark are chugging along by e.g. introducing Kubernetes as an alternative to YARN. Exciting times for the ecosystem!

Sources:

- Photo by Rick Mason on Unsplash, 17.01.2020
- Hadoop Architecture Diagram: https://2.bp.blogspot.com/-w7KeAnwWnBQ/WfYBJzgtvQI/AAAAAAAAAMk/D58SpZfK7lkJ8QnKnQZW268mKzRvuOOnACLcBGAs/s640/HadoopStack.png, 23.01.2020
- https://hadoop.apache.org/, 17.01.2020
- https://hive.apache.org/, 17.01.2020
- http://parquet.apache.org/, 17.01.2020
- https://docs.aws.amazon.com/redshift/latest/dg/c_columnar_storage_disk_mem_mgmnt.html, 17.01.2020
- Snijders, C.; Matzat, U.; Reips, U.-D. (2012). "'Big Data': Big gaps of knowledge in the field of Internet". *International Journal of Internet Science*. 7:1–5.
- https://pbs.twimg.com/media/BiZiNmXCAAA2n8U.jpg, 19.01.2020

Hadoop | Big Data | Hive | Analytics | Docker

Follow

Written by Nikolay Dimolarov

161 Followers · Writer for Towards Data Science

I solve problems with software. I am a CTO, software engineer and product guy. https://www.dimolarov.com/

More from Nikolay Dimolarov and Towards Data Science

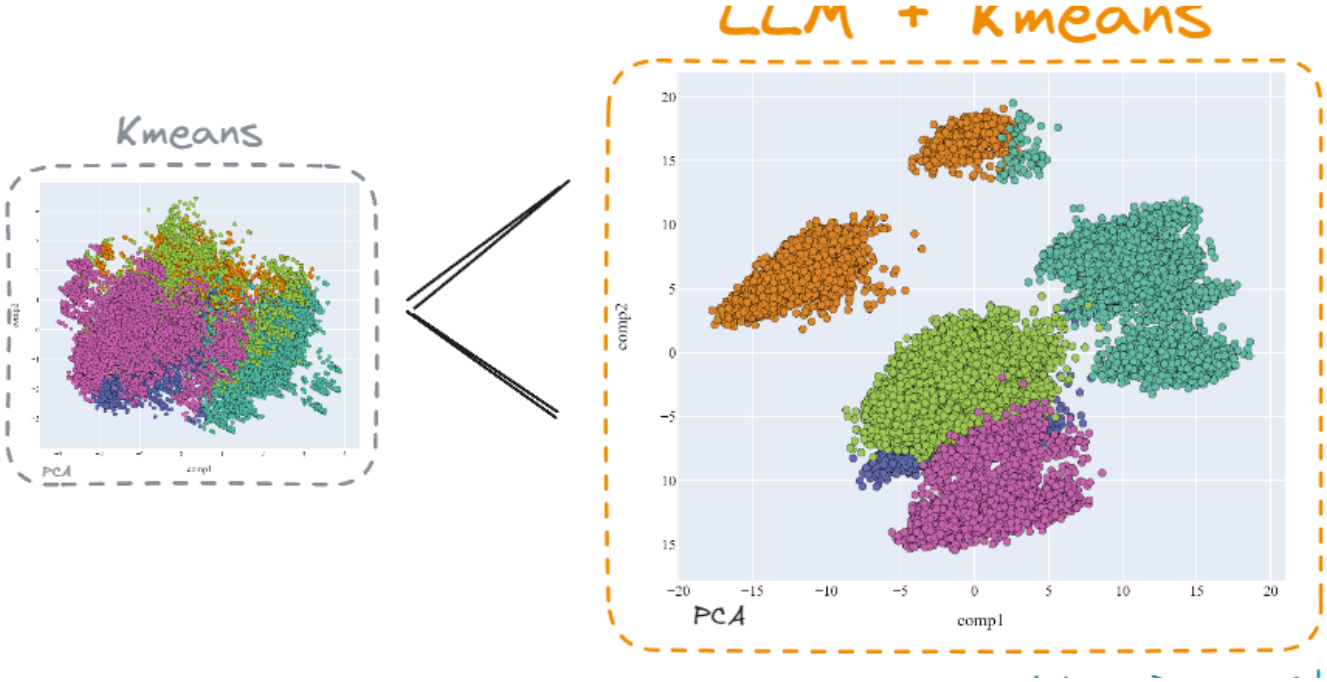
Nikolay Dimolarov in Towards Data Science

What's behind a simple SQL query?

A brief behind the scenes examination of what happens when an app/developer runs a SQL query to fetch data from a PostgresDB database.

7 min read · Aug 6, 2020

137 1



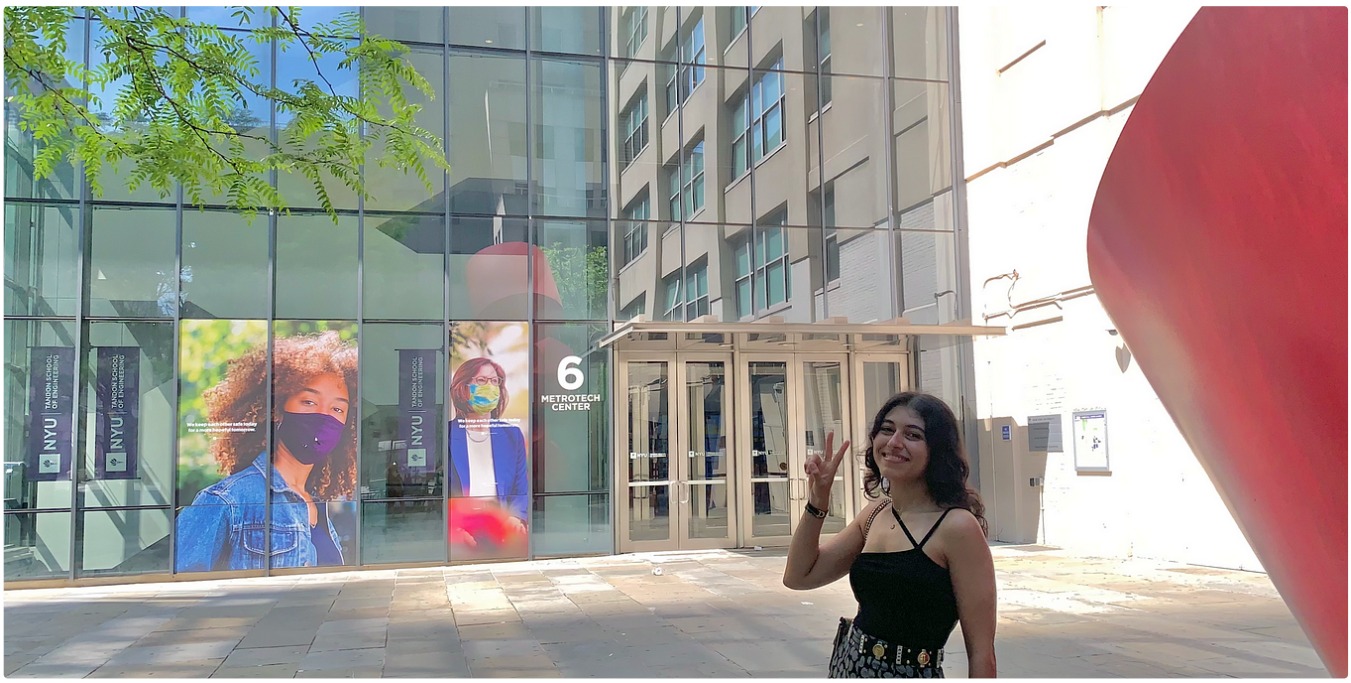
Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

23 min read · Sep 27

2.8K 24



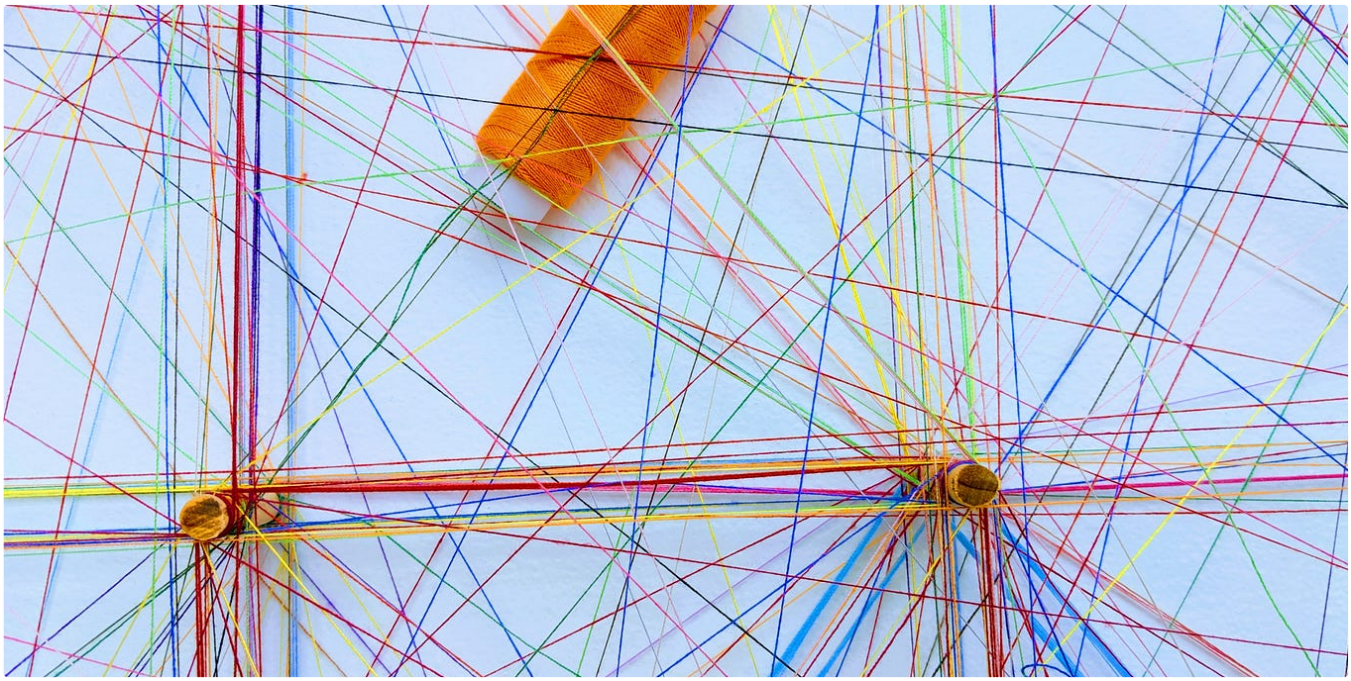
Khouloud El Alami in Towards Data Science

Don’t Start Your Data Science Journey Without These 5 Must-Do Steps From a Spotify Data Scientist

A complete guide to everything I wish I’d done before starting my Data Science journey, here’s to acing your first year with data

✦ · 18 min read · Sep 25

2.1K 21



Nikolay Dimolarov in Towards Data Science

On the state of Deep Learning outside of CUDA’s walled garden

The current state of training artifical neural networks on non-CUDA GPUs.

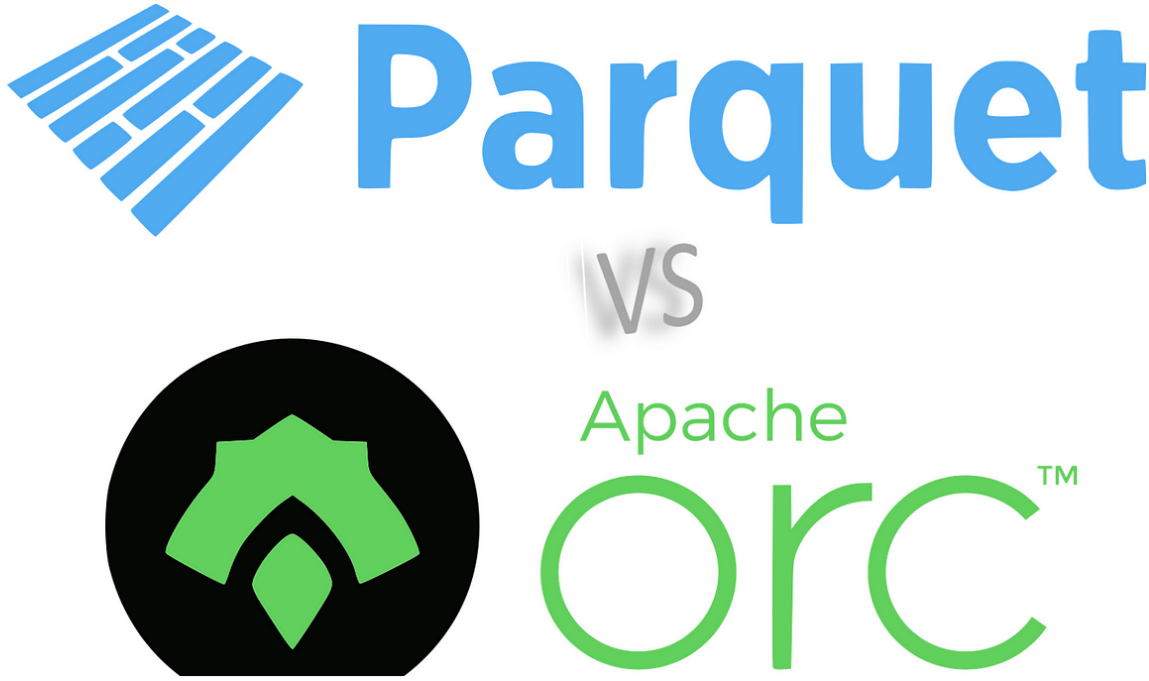
5 min read · Jun 5, 2019

911 6

See all from Nikolay Dimolarov

See all from Towards Data Science

Recommended from Medium



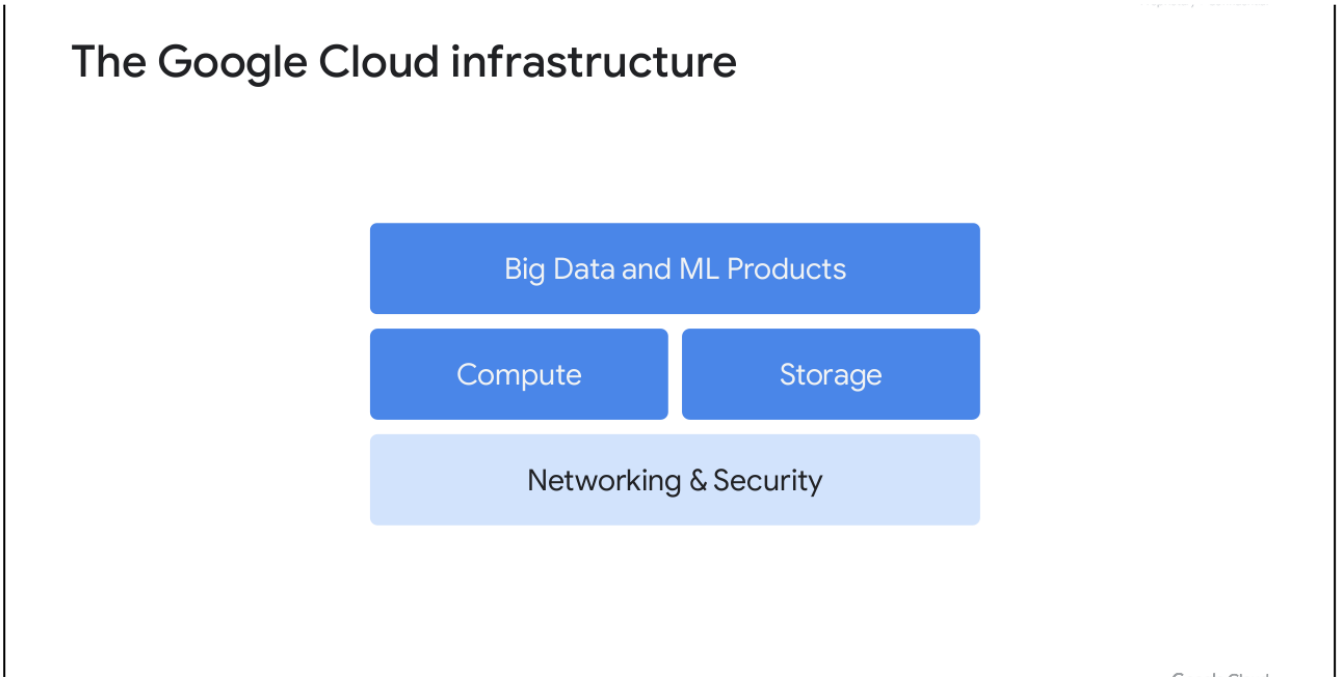
Ankush Singh

Why Parquet vs. ORC: An In-depth Comparison of File Formats

If you work in the field of data engineering, data warehousing, or big data analytics, you're likely no stranger to dealing with large...

5 min read · Jun 5

27



Betul Gurbuz

Creating a Streaming Data Pipeline for a Real-Time Dashboard with Dataflow

Let's start by exploring the Google Cloud infrastructure through three different layers.

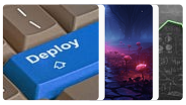
9 min read · Apr 17

34

Lists

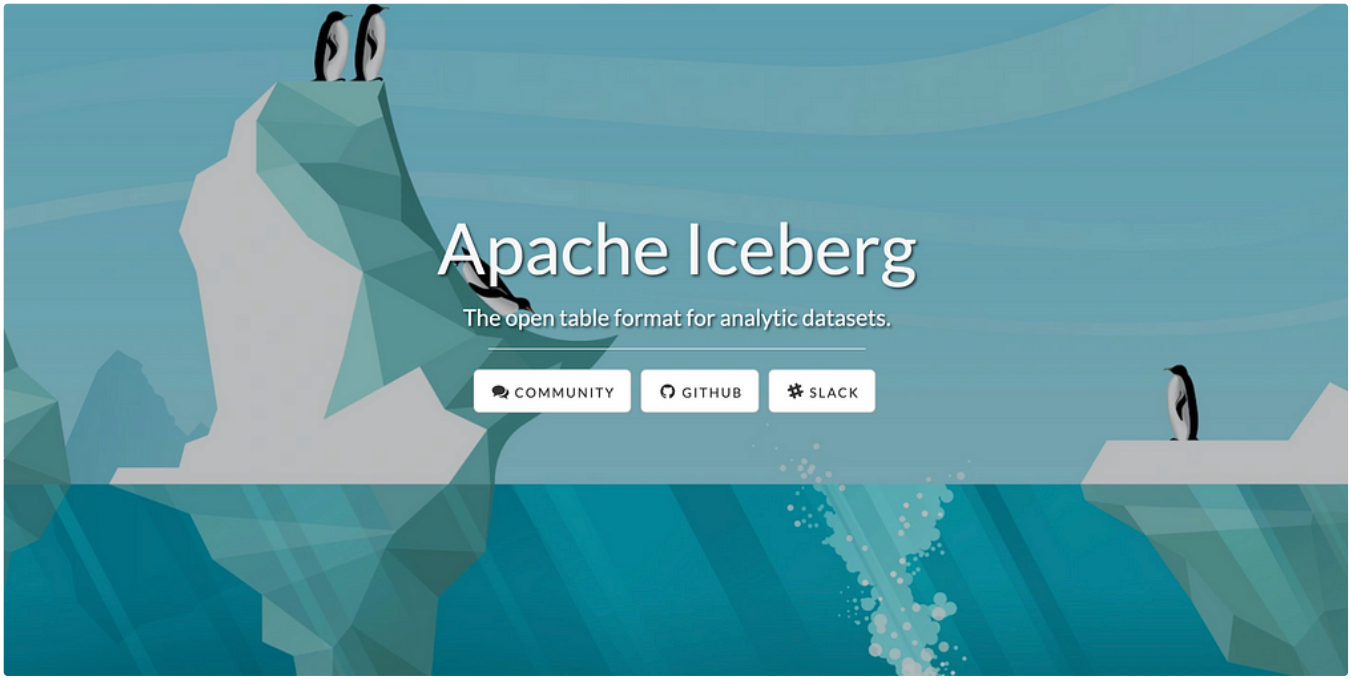
Coding & Development
11 stories · 208 saves

New_Reading_List
174 stories · 144 saves



Predictive Modeling w/ Python
20 stories · 475 saves

Natural Language Processing
689 stories · 305 saves



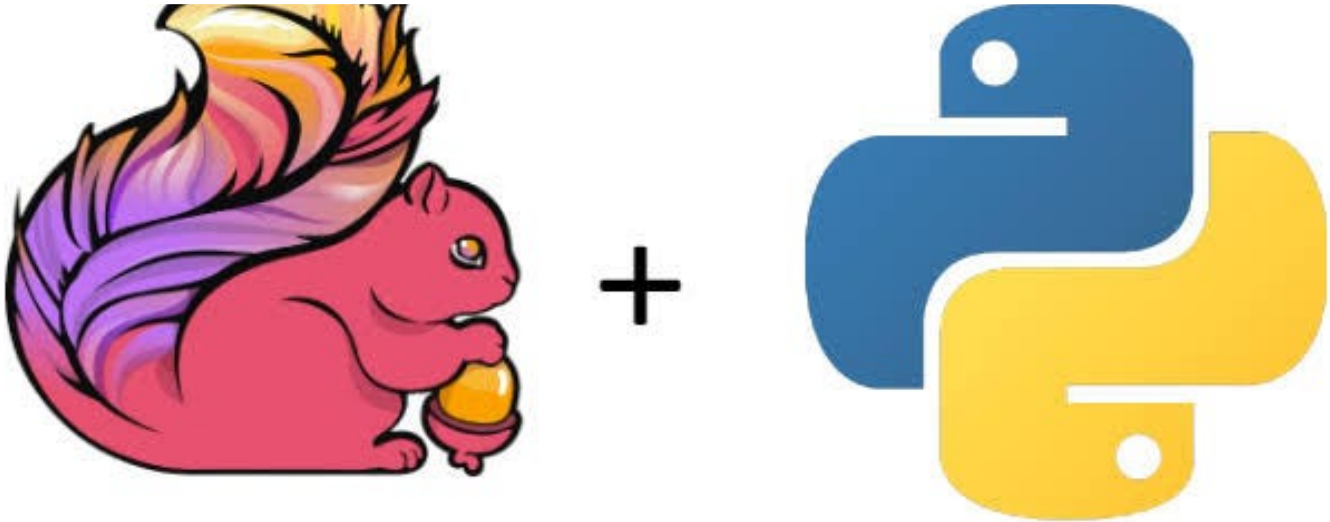
Sree Vaddi

Quickstart Iceberg with Spark and Docker Compose

Introduction

3 min read · Apr 24

3 1



Daniel Santana

Flink + Docker + Kafka

Apache Flink is a powerful stream processing framework that enables real-time data processing. Docker provides an easy way to set up and...

2 min read · Aug 7

9

Ansab Iqbal

Delta Lake Introduction with Examples [using Pyspark]

What is Delta lake

6 min read · Aug 20

6

Amit Joshi

Spark Architecture: A Deep Dive

Apache Spark is an open-source distributed computing system designed for big data processing and analytics. Spark is known for its speed...

5 min read · Jun 1


17