ON the evolution of Data Engineering

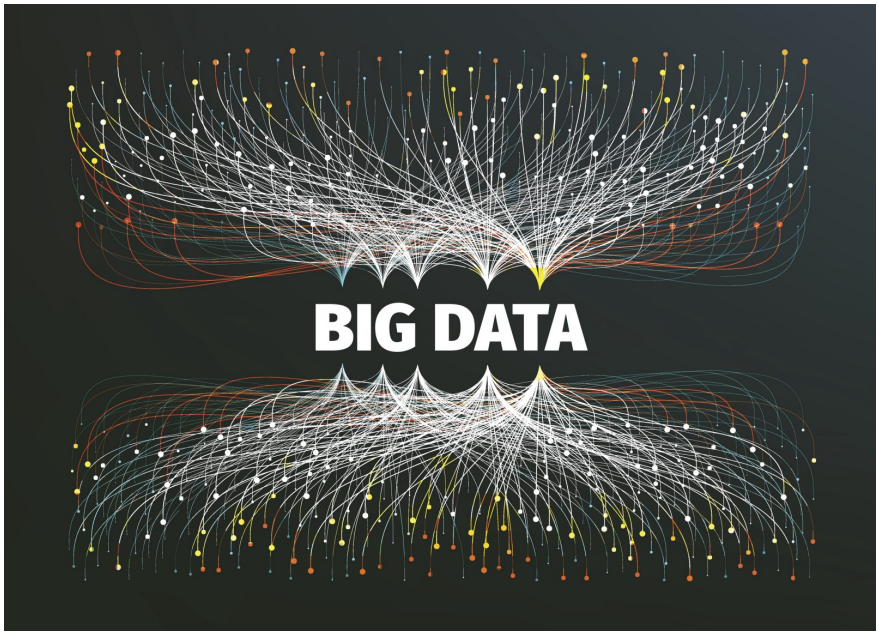
Julien Kervizic Follow

Oct 8, 2018

A few years ago being a data engineer meant managing data in and out of a database, creating pipelines in SQL or Procedural SQL and doing some form of ETL to load data in a data-warehouse, creating data-structures to unify, standardize and (de)normalize datasets for analytical purpose in a non-realtime manner. Some companies were adding to that a more front facing business components that involved building analytic cubes and dashboard for business users.

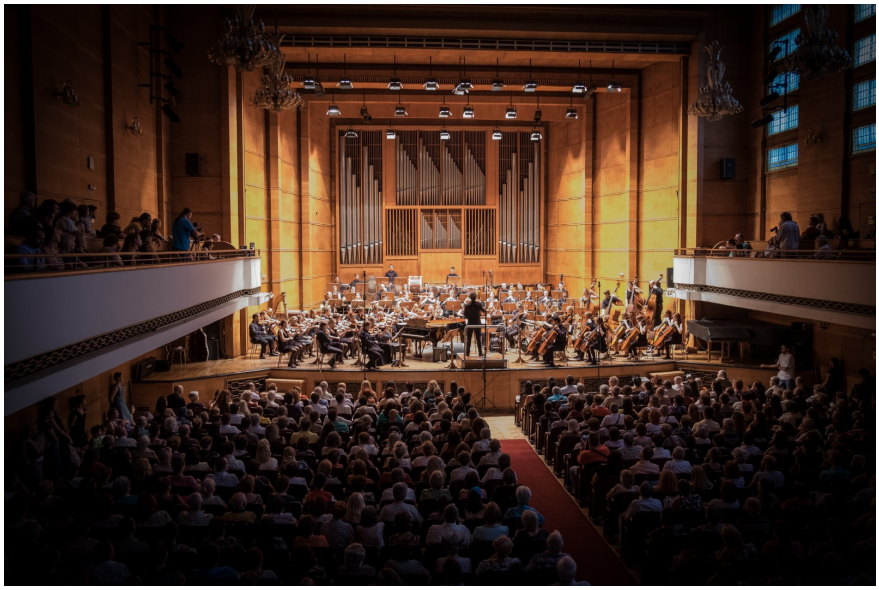
In 2018 and beyond the role and scope of the data engineers has changed quite drastically. The emergence of data products has created a gap to fill which required a mix of skills not traditionally embedded within typical development teams, the more Software Development Oriented data engineers and the more data oriented Backend Engineers were in a prime role to fill this gap.

This evolution was facilitated by a growing number of technologies that helped to bridge the gap both for those of Data Engineering and those of a more Backend Engineering background.



Big Data: The emergence of Big Data and the associated technologies that can with it drastically changed the data landscape

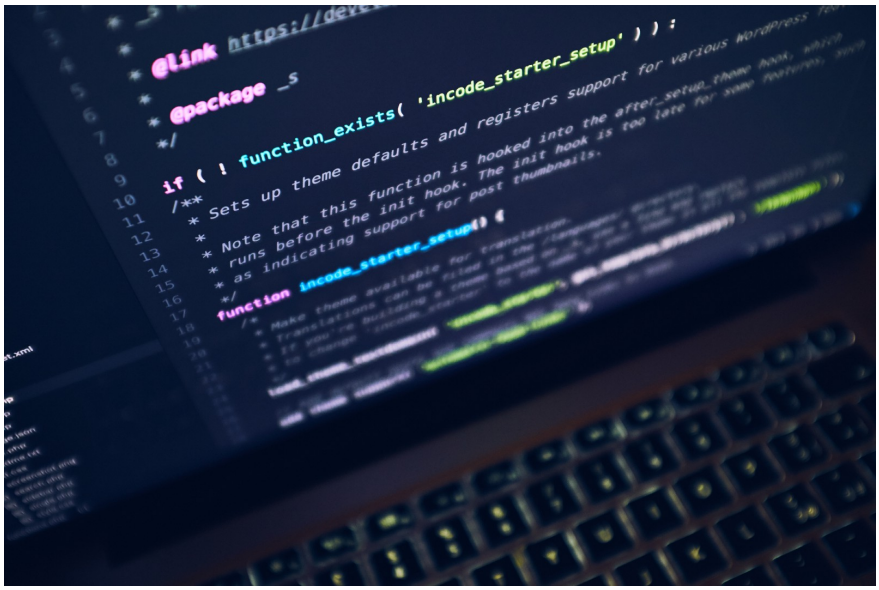
with Hadoop open-sourced in 2006, it became easier and cheaper to store large amount of data, Hadoop unlike traditional RDBMS databases did not require a lot of structuring in order to be able to process the data. The complexity to develop on Hadoop was initially quite high, requiring the development of Map Reduce jobs in Java. The challenges of processing big data forced the emergence of Backend Engineers working on analytical data workflow. It was not until Hive was open sourced in 2010 that the more traditional data engineers could get an easy bridge to get on boarded in this era of Big data.



"people watching orchestra" by Manuel Nägeli on Unsplash

Data Orchestration Engines: With the development of Big data, large internet companies were faced with a challenge to operate complex data flow without any tools such as SSIS used for more traditional RDBMS working in this ecosystem. Spotify opened sourced Luigi in 2012 and Airbnb Airflow (inspired by a similar system at facebook) in 2015. Coming from a heavy engineering driven background, these orchestration engines were essentially data-flows as code.

Python being the language most of these orchestration engine were built on helped them gain ground benefiting based on the traction in the PyData ecosystem and from the increase use of python among Production engineers. Traditional Data Engineers coming into this ecosystem needed to adapt and up-skill in software engineering.



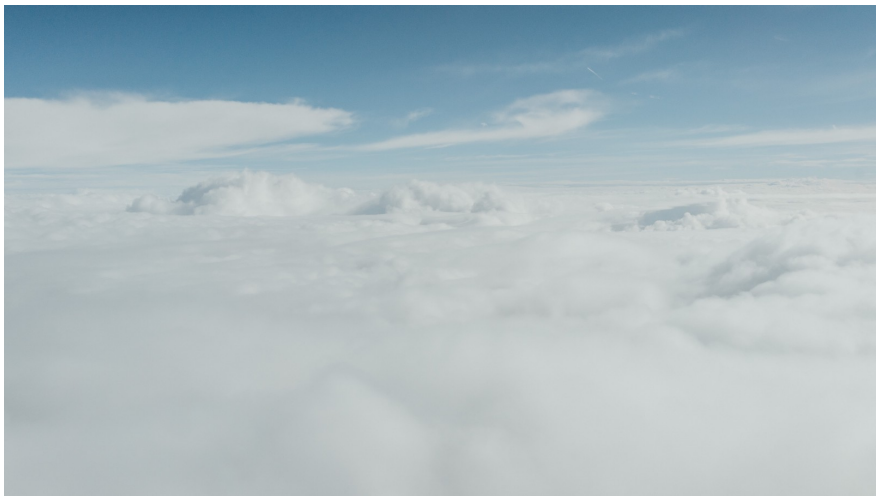
"turned on gray laptop computer" by Luca Bravo on Unsplash

Machine Learning: The trove of data that was now possible to collect from the internet, Machine learning quickly gained traction. Until the advent of Hadoop, Machine Learning models were usually trained on a single machine and usually applied on a very ad-hoc manner. For large internet companies, in the early days of Hadoop leverage machine learning models required some advanced software development knowledge in order to train and apply models into production, with the use of frameworks such as Mahout leveraging upon MapReduce.

Some Backend engineers started to specialize in this area to become Machine Learning Engineers, very production focused Data Scientists. For a lot of startup this kind of development was however overkill. Improvement in SKLearn, a python project open started in 2007, and the popularization of orchestration engine made it fairly easy to go from a proof of concept by a Data Scientist to production ready workflows by Data Engineers for moderately sized datasets.

"three flaming sparkler sticks" by David von Diemar on Unsplash

Spark & Real-time: It was the release of Spark's MLlib for python in 2014, that democratized machine learning computation on Big data. The API was fairly similar to the one Data-scientists were used to from the PyData ecosystem and further development of Spark further helped bridged the gap. Spark further offered a way for data engineers to easily process streaming data, offering a window towards real-time processing. Spark enabled an increased contribution of data engineers towards data products.



"clouds" by MILKOVÍ on Unsplash

Cloud development & Serverless: AWS was officially launched in 2006, its storage layer S3 had been built upon Hadoop the traditional big data platform. Elastic Map Reduce was launched in 2009 making it easier to dynamically spin up and scale Hadoop clusters for processing purpose.

The move to the cloud had multiple implication for data engineers. The cloud abstracted physical limitations, for most users it meant that storage and compute was essentially infinite provided one can pay for it. Optimization previously done to keep business running waiting for new servers to be installed or upgraded needed not to be done anymore. So was the work previously done tasks scheduling to allocate the load across time due to ressource constraint. The cloud by allowing for scaling up and down ressources made it much easier to handle high peak batch jobs typical in data engineering. This however came at the cost of having to manage infrastructure and the scaling process through code.

The introduction of Lambda function on AWS in late 2014 kicked off the serverless movement. From a data perspective data could be easily ingested without managing infrastructure. The release Athenalaunching in late 2016 pushed things further allowing to query directly onto s3 without the need to setup a cluster. This is freeing data engineers from managing infrastructure scaling based on requests allowing them to spend more time on development,

The role of the data engineer is no longer to just provide support for analytics purpose but to be the owner of data-flows and to be able to serve data both to production and for analytics purpose.

To that end Data Engineering has been looking more towards a software engineering perspective. Maxime Beauchemin's post on functional data engineering advocates advocates for borrowing patterns of functional programming and apply them to data engineering. The emerging data ops movement and its manifesto in turn borrows from the DevOps movement in software engineering.