Twitter Thread by Jamin Ball

Jamin Ball
@jaminball

A trend I'm excited for this year: DataOps & the Analytical Engineer

~10 years ago DevOps was born. The role of system admins and developers merged. Infrastructure became self-serve

Today the role of data engineers and business analysts are merging. Data is becoming self-serve

Data infrastructure is becoming so powerful that the tools today allow non-technical folks to carry out the once complicated / custom code/ huge backlog jobs of data engineers.

Before getting into what this means, let's first discuss how we got here

Before 2012 the data world was dominated by transactional (OLTP) databases like PostgreSQL, MySQL, etc and analytical (OLAP) databases like Oracle, Netezza

Tools like Informatica / Talend were used to batch load (ETL) data into these databases, Tableau used to visualize

As you can imagine, there was heavy engineering work to manage the environment...

Then in 2013 AWS released their cloud data warehouse Redshift, and it was a game changer. Snowflake was founded in 2012, but didn't really pick up steam until a few years later (around 2016)

So why was Redshift a big deal?

1. It was the first cloud-native OLAP warehouse. It reduced the TCO of an OLAP database by orders of magnitude.

2. Speed of processing analytical queries increased dramatically

3. And later on (Snowflake pioneered this) they separated compute & storage. In overly simplified terms, this meant customers could scale their storage and compute resources independently of one another. This was a huge deal

What does this all mean? An EXPLOSION of data

The barriers to maintain a database were completely broken down, and the amount of data that was sent to Redshift / Snowflake / BigQuery skyrocketed.

Now, after this point we still weren't ready for DataOps / Analytical Engineers. What did it take to get there?

In my opinion there were 3 major technologies / shifts that happened that have given rise to DataOps:

1. The shift from ETL to ELT (extract-transform-load to extract-load-transform). Data used to be transformed (joined, aggregated, cleaned, etc) in motion while being loaded into the warehouse. Now, data is being loaded into the warehouse in it's raw form..

...Why is this important? In an ETL process if something goes wrong it's very hard to debug if the issue happened in the "T" or the "L". It was also harder to build these pipelines. With ELT, tools like @fivetran allow you to point & click to connect source data to your warehouse

The big trend here? The data warehouse is starting to subsume the data lake, and the default is becoming: "just send all data to Redshift / Snowflake." Again, the barriers of storing / collecting data is going way down

2. The importance of the cloud data warehouse. We already talked about this, but the one incremental point I want to make is the power of the compute within the warehouse went way up, and the cost of that compute went way down...

...this is fundamentally what enabled the "T" in ELT to happen within the warehouse. The compute horsepower of a Snowflake / Redshift made it possible

3. So who's driving these transformations? Tools like @getdbt @fishtowndata. The big technology advancement of the open source project dbt was representing these data transformations as code (SQL). It allowed anyone who knows SQL (business analysts) to author the transformations

Prior, the transformations were done with custom Python code by data engineers, or GUI based ETL tools. These took forever to build, were inflexible / hard to scale, and a black box

So in summary, the major platforms enabling the rise of DataOps are:

1. Data Movement: @fivetran
2. Data storage / compute: @SnowflakeDB @awscloud @GCPcloud
3. Data Transformations: @getdbt

If you think about what these technologies allow: to get data into a warehouse you just point and click data sources to their destination. The connectors are pre-built. You don't have to manage an on-prem database. And the data transformations are represented as basic SQL

And here is my KEY point (I get I've buried the lead a bit, but I believe the setup is important): Data Engineers used to manage all of the complexities of moving, storing, and transforming data. A lot of it was built with custom code / Python, and managed MANUALLY ■

Today, with the powerful tools I listed above, the data ecosystem can be managed with turnkey tools, removing the need for a lot of the complex work the data engineers handled previously....

...Instead of the BI analysts requesting the data engineer build them a pipeline of data, they can point & click with Fivetran and get data loaded into Snowflake, and write some SQL with dbt to get a materialized view (subset of data) to efficiently run their query against

After quite the lead in I hope that it's clear now how the business analysts (like BI analysts) can now carry out the functionality of data engineers to access data in a turnkey, self-serve manner. I'm describing these analysts as the "Analytica Engineer" and the process DataOps

So why is this important, and why am I excited about DataOps this year?

When data access becomes democratized and self-serve in nature, the need for new tools to manage this "modern data stack" goes up. I think we'll see a TON of huge companies built in the following categories:

1. I think we'll see the data catalogue / lineage systems re-invented. To access data you need to know where it's located / where it came from

2. Monitoring the quality of the data (and pipelines) will become increasingly important in self-serve settings. "Datadog for data"

3. A wave of modern BI tools

4. Governance becomes a bigger deal. With self-serve access, how can you govern who should have access to what?

5. Metadata gains increased importance

6. The way data pipelines were orchestrated (ie Airflow) will completely evolve

6. The "endpoints" for data coming out warehouses will grow. Right now the two primary places data goes from the warehouse are BI ML tools. I think we'll see an explosion of data going into new places, like back into SaaS apps

7. And many more we can't even imagine now