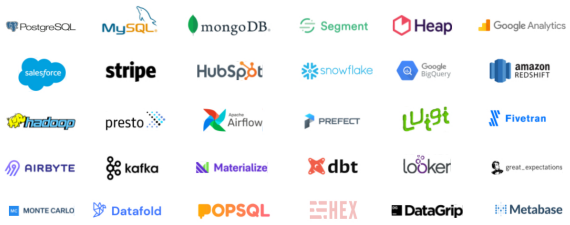
WHAT YOUR DATA TEAM IS USING: THE ANALYTICS STACK

February 16th, 2021

ANALYTICS DATA WAREHOUSE



Not many people know this, but before Tolstoy released *War and Peace* in its entirety in 1869, he originally titled it: *Analytics: what is it good for*. And here at Technically, we often wonder the same thing. Since Snowflake broke the internet with a *checks notes* $80B market cap, everyone – in tech, finance, and misc. industries alike – have been paying closer attention to the analytics stack. And they have questions:

- What are these data analysts using?
- What is this "analytics engineer" title that keeps popping up?
- Why would you use Snowflake over BigQuery or Redshift?
- What does Segment do? And what is Fivetran?
- What happened to Looker, and what's dbt?

Fear not - while I may not have learned much during my Data Science major, a few years working on analytics teams has given me the requisite knowledge to illuminate this thick darkness of harmful jargon.

SOME DISCLAIMERS

As always, a few notes in advance so that I won't have to deal with angry comments:

–> Analytics stacks are highly company dependent

I've worked with data at 5+ different companies, all with very different analytics stacks. The infrastructure you choose depends **highly** on your budget, technical history, stakeholder needs, and types of people on staff. It is **not** one size fits all.

–> Analytics stacks are very different from Data Science stacks

Data *Science* – or in my personal definition, the practice of creating and utilizing predictive models to drive business value – is distinct from analytics, and that distinction is beyond the scope of this post. Suffice it to say that Machine Learning is its own beast with its own tools, and I won't be covering those tools here. But, having said that, Machine Learning stacks will usually **overlap** with analytics stacks, and rely on some of the same tools (e.g. data warehouses).

–> The universe is constantly changing - don't read this in the future

In my few, scant years in the workforce, tooling for analytics has undergone drastic change. Hell, the entire analytics practice – especially team organization and roles – is completely different than it was when I graduated undergrad. I believe it was Yogi Berra who said that "change is the only constant," and his wisdom extends to this domain as well.

–> This content is not exhaustive

There are thousands of available tools in the data ecosystem, and it's impossible to cover them all. I have also not used them all, and am not aware of them all. So if you think I missed something, just get in touch!

THE ANALYTICS STACK - 10K FEET UP

The goal of any analytics stack is to be able to **answer questions about the business with data**. Those questions can be simple:

- How many active users do we have?
- What was our recurring revenue last month?
- Did we hit our goal for sales leads this quarter?

But they can also be complex and bespoke:

- What behavior in a user's first day indicates they're likely to sign up for a paid plan?
- What's our net dollar retention for customers with more than 50 employees?
- Which paid marketing channel has been most effective over the past 7 days?

Answering these questions requires a whole stack of tools and instrumentation to make sure the right data is in place. You can think of the **end product** of a great analytics stack as a nicely formatted, useful data warehouse full of tables like "user_acquisition_facts" that make answering the above questions as simple as a basic SQL query. \

But the road to getting there is unpaved and treacherous. The actual data you need is **all over the place**, siloed in different tools with different interfaces. It's **dirty**, and needs reformatting and cleaning. It's **constantly changing**, and needs maintenance, care, and thoughtful monitoring. The analytics stack and its associated work is all about getting that data in the format and location you need it.

The basics:

1. **Where data comes from**: production data stores, instrumentation, SaaS tools, and public data
2. **Where data goes**: managed data warehouses and homegrown storage
3. **How data moves around**: ETL tools, SaaS connectors, and streaming
4. **How data gets ready**: modeling, testing, and documentation
5. **How data gets used**: charting, SQL clients, sharing

For each section, we'll note a few sample tools that are popular for getting the job done.

WHERE THE DATA COMES FROM

All of that nice, formatted, joined data in the data warehouse – the data that makes up the dashboards, the models, and the slide decks – has to originate from somewhere. And it usually comes from one of four places:

1. PRODUCTION DATA STORES

Pretty much every app is backed by a database, a series of databases, a stream; just data of some sort. If you're using an app that lets you log in, there's a database to store user information. Your emails in Gmail are stored in a database. Your tweets are stored in a database. In many cases, this data is actually pretty useful for analytics. You may be able to answer "how many active users do we have?" from the users database, and so on and so forth.

In terms of tools, there are literally thousands of databases on the market, from relational ones (Postgres, MySQL, MSSQL, etc.) to NoSQL ones (MongoDB, Firebase, DynamoDB) to streaming ones (Kafka, although apparently Kafka is not a database).

2. INSTRUMENTATION

Much more in vogue these days is **instrumenting** your product, or firing little "events" every time a user does something in the product. Those events go into a database, and can be used to answer questions down the road. If a user clicks on the "settings" tab in your app, you might automatically fire an event like this:

```
{
  name: "settings_tab_clicked",
  user_id: "e2uirh2837yrfh4ur3f34",
  user_email: "justin@technically.dev",
  created_at: "2021-01-01",
}
```

Now, if I'm curious how often users click on the "settings" tab, I can just aggregate these events with basic SQL. The move towards getting information out of events instead of production databases is getting called *Event Driven Analytics* and it's

picking up steam.

Instrumentation is mostly done in code, and teams will often (usually?) do it manually. But Segment also provides a library and infrastructure for firing and managing these events, and it's getting quite popular. We're also seeing new startups like Iteratively that help teams manage and test their product events.

3. SAAS TOOLS

Businesses run operations on tools like Salesforce, Hubspot, and Stripe, and those tools output valuable data that's essential to answering key business questions. If you're running your billing through Stripe, *that's* where your revenue data is going to come from. Salesforce has interesting information on how your deals pass through the sales process, demographic data on your leads, etc.

The hard part is getting data **out** of these Saas tools and **into** a place where you can analyze it along with all of your other data, i.e. the warehouse. More on that in a later section.

4. PUBLIC PLACES / DEMOGRAPHIC DATA

It's less common, but teams will sometimes source data from public sources or use enrichment providers like Clearbit to get information on their customers or site visitors. We're also starting to see domain-specific providers like Iggy that provide high quality data as a service.

WHERE THE DATA GOES TO

Common knowledge today is that for analytics, you want a **data warehouse** with all of the data you need centralized in **one location**, and optimized for **large queries** to **run fast** (for more on why you can't just query what already exists, read up here).

Generally, building and maintaining warehouses was pretty annoying, and required specialized infrastructure and expertise. But today (thankfully) we're in a golden age of managed data warehouses, where you can pay Snowflake or BigQuery to manage that tedium for you (for a hefty price, of course).

1. MANAGED DATA WAREHOUSES

The three big players in the space are Snowflake, BigQuery (from Google), and Redshift (from Amazon), with a new entrant (Firebolt) getting some press. These solutions are expensive, but they make the analytics stack **a lot** simpler by obviating the need to manage servers and infrastructure.

I often get asked why teams choose Snowflake over BigQuery, and to analyze the respective merits of these different products. I've personally used all 3 of the big ones, and performance varied, but that was mostly based on architecture and data types. There is **no one size fits all answer** to this question: companies generally evaluate solutions based on cost, feature set, integrations, and portability (e.g. Snowflake is multi-cloud, sort of).

2. HOME ROLLED STUFF

If you want to build your own data warehouse – because of cost, security, or feature considerations – there's a bunch of open source software out there to use. A popular stack is Hive on top of HDFS, which are both part of the Hadoop ecosystem. I've also seen Presto used here as a query engine, running on top of something like Mesos for compute management.

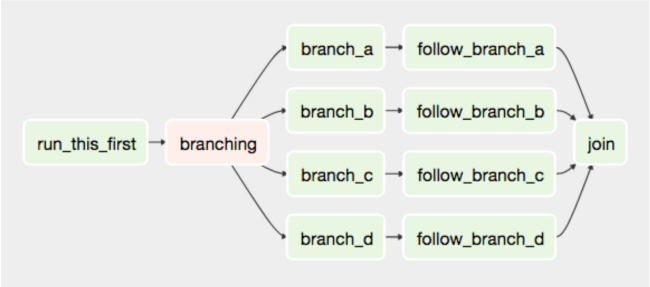
HOW THE DATA MOVES AROUND

Once you've got a clean handle on where your data comes from and what warehouse it's going to go to, you need to start thinking about how to move it from place to

place. The goal of this part of the stack is to get data **out of siloed systems** and **into your centralized warehouse**, where you can query it as you place.

1. WORKFLOW ORCHESTRATORS

Getting data out of your source systems and into your warehouse isn't usually a simple "pick it up and move it" kind of operation – there are usually a bunch of things you'll want to do to that data while it's on the way, like transform it and clean it (see next section). For larger companies with complex datasets, you might see pipelines that rely on 20+ different source tables just to build one little piece of the warehouse. Pipelines like that usually get represented with diagrams like this:



Workflow orchestrators help teams build these complex pipelines, make sure they run on time, and alert you when things go wrong. The most popular open source tool in the space is Airflow, with Luigi still hanging on and up and comers Prefect and Dagster making some waves.

> **WORKPLACE EXAMPLE**
>
> At DigitalOcean, we used Airflow to author these complex data pipelines. The idea was to take data out of source locations like our production database, move it into the warehouse, and aggregate it in a way that made it easy to report on our invoices and revenue.
>
> **WORKPLACE EXAMPLE**

I wrote more in depth on this space here. Note that some architectures may copy source data into the warehouse before transforming it, instead of doing that in flight: this is the difference between ETL (extract - transform - load) and ELT (extract - load - transform). Although I've mostly seen it in machine learning contexts, some teams do use Apache Spark to author distributed data pipelines. If you've heard of Databricks, that's how they make their money - selling Spark.

2. SAAS TOOLS

In an answer to the prayers of analytics teams around the globe, there are now SaaS tools you can use to move data around without having to write a ton of code. These tools are generally oriented to data sources that sit in SaaS tools, like some of the ones we mentioned earlier (Salesforce, Hubspot, etc.).

Data that sits in SaaS tools like Salesforce or Stripe have the benefit of a **fixed schema** – the way the data is stored is basically identical for everyone. Every company that uses Stripe has (give or take a bit) identical table names, column names, and database structure – because Stripe set everything up. The same is mostly true for Salesforce and Hubspot. And that consistency makes it much simpler to move data around without custom code.

The most popular tool for this kind of data movement is Segment, which helps you move data from sources like Salesforce into your warehouse, or even other tools you want your Salesforce data in. Fivetran is another popular option, with open source competitor Airbyte coming out of stealth recently, and Meltano offering a more full stack option.

3. STREAMING

The model of moving data from source to destination over and over again isn't the only way you can architect your systems. As more consensus around Event-Driven Analytics hits the ground, the idea that all events should just be streamed to a central place where teams can **subscribe** to it continuously is starting to gain popularity. Jay Kreps (founder of Confluent + co-creator of Apache Kafka) wrote the manifesto for this way of thinking here.

Streaming is a larger decision than just analytics, will usually involve engineering teams, and generally is oriented towards more real time use cases (i.e. user facing analytics, Machine Learning, etc.). The default solution for streaming like such is the open source Apache Kafka, which can pay for as a managed service via Confluent, as well as closed source alternatives on AWS and GCP. New entrant Materialize is taking a SQL-centric approach.

> 🚨 **CONFUSION ALERT** 🚨
>
> Overlap in this space can be very confusing – teams can use *all three* of the above categories, one of them, or none at all. In developer tools, what it means for companies to be "competitive" is a lot more nuanced than in other fields.
>
> 🚨 **CONFUSION ALERT** 🚨

TRANSFORMING AND MODELING DATA

Data in its source format is rarely organized in the way that you need it to be for your analyses. As you use your ETL tools to move that data from source to warehouse, you'll usually want to apply transformations that get the data in a better format. Beyond that, **how to build and organize a data warehouse** is an art in itself: which kinds of tables should we have? How do we document what different columns mean? How do we test that data types are correct and things haven't been duplicated? This is where transformation and modeling tools come in.

1. MODELING

Effective analytics teams have what's called a **data model** – it's how they map the domain of the business (users, dollars, activity) to the actual underlying data. Modeling covers a few important pieces of warehouse design:

- **Definitions** – what does a *user* mean? What does *active* mean? What does *recurring* mean? I.e. how you map concepts to your data
- **Intermediate tables** – taking common queries, scheduling them, and materializing them as tables in your warehouse (e.g. daily_active_users)
- **Performance and structure** – how to organize longer and wider tables in a schema that optimizes query speed and cost

Modeling is a discipline as old as time, and to this day, teams still pay consultants to look at their business domain and intelligently map it to how data should look. The Star Schema is one approach to warehouse design that's pretty popular.

Typically, modeling has been relegated to disparate SQL files and scheduling via ETL tools. But dbt, an open source package for data modeling via templated SQL, has been all the rage over the past few quarters. They're riding the wave of "the emergence of the Analytics Engineer" and offer a full solution for building, testing, and documenting a warehouse. I would say the incumbent here is Looker, which (among other things) provides a markup language for data modeling called LookML.

2. TESTING AND MONITORING

Modern application codebases are checked into version control, rigorously tested (hopefully) before deploying, and monitored closely for performance issues. Teams are starting to come to the conclusion that data should be no different – pipelines and tables should be tested regularly to avoid data mishaps and inaccuracies, and continuously monitored to keep uptime as close to 100% as possible.

dbt has native support for building basic testing into your warehouse, but open source Great Expectations, along with Monte Carlo and Datafold, are tackling the testing / monitoring problem as standalone solutions.

3. DOCUMENTATION

A warehouse is only useful if your team can actually understand it, and without a set of documentation to explain which tables mean what and how they're built, you'll get stuck quickly. Again, dbt comes in the clutch here with their documentation feature, while newcomers like Atlan and Metaphor are working on making metadata more accessible. Open source Amundsen has been around for a bit. You can also just keep things in a Google Doc if you want.

VISUALIZING AND USING DATA

Alright – so you've got your source data in order, you've set up systems to move and transform that data, you've got a beautifully organized and documented data warehouse, and you've set up monitoring and testing to make sure things stay that way. Now what? The final step is actually writing queries against and visualizing the data that your team has worked so hard to curate.

1. WRITING QUERIES

If you have the credentials for your data warehouse, you can write SQL pretty much anywhere. In addition to just running SQL files through your command line, analysts and scientists have literally hundreds of options for software to write SQL in, from modern startup-y places like PopSQL to OGs like DBeaver. I cannot stress how many of these exist, it's actually ridiculous. Most data warehouses even let you write queries via their own tools.

2. VISUALIZATION AND SHARING

For analytics folks, making charts and graphs can end up being the most annoying part of the whole data workflow. It's usually some disjointed combination of querying data, copying and pasting it into Google Sheets, and then screenshotting that graph into a slideshow (or something like that). There are basically 4 ways to tackle this:

1. **The tools you love** – get data out of your warehouse and into a familiar tool, like Excel or Google Sheets
2. **Visualization + modeling** – platforms like Looker and Tableau have visualization built in, but you have to buy into their whole ecosystem
3. **In your SQL client** – visualize the SQL you write directly, like in PopSQL or Numeracy (RIP)
4. **Standalone dashboarding tools** – write SQL and get charts, like Metabase and Redash (RIP) (there is overlap here with #3)

And then you've got tools like Mode, that sort of let you do all of this, and also add in Python and R. Companies like Hex help you visualize *and* share your work too.

In practice, you'll see teams use all four of these in consonance (I certainly have in past roles). The general climate is that visualization is a "need to have" for tools in the space, but not necessarily a point of differentiation.

PRACTICAL EXAMPLE: HOW IT ALL WORKS TOGETHER

To make all of the above more concrete, let's walk through an imaginary example. Suppose years down the road, more and more of you loyal subscribers continue to flock to Technically, and we suddenly become a big company with a data team. We're looking for **basic visibility** into the business:

- How many subscribers do we have?
- What's the ratio of free:paid subscribers?
- What are our most valuable referral channels?
- Which types of content are most popular?

The team is tasked with creating a few dashboards so the management team can look at this data regularly, stacked with charts and backed by SQL. Here's the stack we might build:

→ Where data comes from

Technically is hosted on a web server. We have a production **Postgres database** with users and subscriptions, and fire events via **Segment** every time a user visits a page or reads a piece of content. We also use **Salesforce** for tracking leads and **Stripe** for billing and subscriptions.

→ Where data goes

We went with **BigQuery** as our data warehouse because we had a bunch of Google startup credits lying around.

→ Moving data around

Technically uses **Segment** to move data from Salesforce to Hubspot and vice versa, as well as for getting Salesforce data into our BigQuery warehouse. We evaluated using **Airflow** to author more complex pipelines, but don't have the right use case quite yet.

→ How data gets ready

The Technically warehouse is modeled and documented via **dbt**, and we use **Great Expectations** to make sure each pipeline run returns reasonable, clean data.

→ How data gets used

Our analysts write their queries in the **BigQuery** console, across their own clients like **DBeaver**, and in our shared **PopSQL** instance. We use **Metabase** for dashboarding and visualization.

To sum that all up, this is the stack:

- Postgres / Salesforce / Stripe
- Segment
- BigQuery
- dbt
- Great Expectations
- DBeaver / PopSQL
- Metabase

This is state of the art now, but check back in two years and it will probably not be.

OTHER NOT-INCLUDED TOOLS

There are a bunch of other tools in the space that don't fit cleanly into the analytics stack, but are relevant and worth looking into. A few examples:

→ Warehouse to SaaS sync

A well organized data warehouse is actually an asset for the rest of the business, beyond just the data team. Sync tools like Census and Hightouch help non-technical end users sync warehouse data into the SaaS tools they need, like (paradoxically) Salesforce.

→ Insights for everyone

Further along the theme of making data accessible to non-technical end users at the company, tools like Glean help take your modeled warehouse and expose it in an intuitive way to your sales, marketing, product, etc. teams. Think click on a graph to drill in, etc. Note that this is part of the value proposition of Looker, if implemented correctly. The OG here is Amplitude, which is specifically focused on product analytics.

📲 If you liked or hated this, share it on Twitter, Reddit, or HackerNews.

(Thanks to Taylor Murphy, Claire Carroll, and Barrald for edits 😏)
