**Outline**

**Introduction**

* talk about the Build vs. Buy decision making process.
  + Build
  + Buy
* talk about why it's important to make this decision correctly.
  + **Bigeye's position on whether to build or buy:**
    - Most problems are not value additive to core business. In particular, data observability, data warehouse, ETL orchestrator, none of these basic data tools are a differentiator for your company. Our recommendation is that you should therefore buy them.

**Major Considerations**

* is it something that is core to your business/critical to your business success. If not, go towards a buy approach
  + caveat is that if you're a large business with special data needs
* if you build in-house, what IP are you actually adding to the company? Is this actually what your company is selling?
* what are the ongoing costs?
  + teams are generally good at initial costs but forget the ongoing costs
* does the philosophy of the tool the thinking was do the tools fit the organization, or does the organization have to adapt? Every tool brought its own philosophy; the philosophies were so different from Lyft, and led them to create something from scratch
* are you in a highly regulated industry, vetting the vendor may take a long time
* is it onerous for you to be bound to a vendor's roadmap?
  + if you build, you can build exactly the integrations that you need, and you're not bound to a third party vendor's road map and promises they have made to their other customers
* people are moving from company to company more rapidly than ever before, which makes buying a more attractive option
  + no data program can afford to have to reverse engineer everything that has been built
* don't assume with the vendor — maybe they won't be able to have 500 people log into the tool at once
* if you choose to buy, make sure you understand what it takes from you to make the integration successful

**Case Studies**

**Perpay**

* just under 50 employees
* when they were evaluating how to get data into their marketing tool (reverse ETL tool)
* initially implemented everything in house
* maintaining their own solution was very cumbersome
* ended up choosing Census — had the integration they needed
* how much time were they spending on integration

**Lyft**

* data was store in the s3 data lake with Hive/Presto
* had so much data that no one knew what to search for/what's trustworthy
* solutions that were data catalogs and data governance products
  + data steward goes in and enter information
    - Atlassian and Calibra (buy)
    - Atlas (adopt) (open source)
    - Lyft's own data catalog
  + ended up going the build route because none of the tools fit Lyft's philosophy

**Cost/Benefit Analysis Framework**

* assuming there is an apples to apples comparison
  + Initial cost + ongoing cost
  + Initial cost:
    - how do we know if this build effort is successful
    - define success metrics
      * adoption — people who use it
      * NPS - measure how satisfied you are
      * when do we need to get there
      * this determines how many people/energy
    - the tradeoff with the vendor side is that you still have to do some implementation
    - the build path gives you more control
* list all these requirements — what is the minimum viable product

**Changing Course**

* cost/benefit analysis needs to be done continually, not only once up front, but probably something like once a year
  + eventually the market catches up, and you make the transition
    - very difficult to introduce risks
    - people get married to something and don't want to let it go
* when do you pull the plug on legacy build solutions?
* every year, you should pull the requirements back and up and validate whether current solution still is the best solution

**Considering Open Source Solutions**

* open source is more flexible option
* a nice middle ground of control
* insurance policy — if the vendor goes out of business, you still have the code
* make sure you really understand what's required when choosing open source
  + it costs money to deploy the solution somewhere
  + cost in terms of employees' time to manage the open source solution
  + need a certain skillset
  + somewhat at the mercy of contributors to the project