# Closing the Loop Between Dev and Analytics

AWS User Group
Zach Miller - 4/25/2019
Senior DS @ CreditNinja

# What does closing the loop mean?

- Both Dev and Analytics need to be empowered to be efficient
- Both teams need to share data
- We’re going to talk about why that can be a struggle…
- … and some architecture choices we can make to help

# Who am I?

- Data scientist that also cares about writing software that is palatable

# Who am I?

- Data scientist that also cares about writing software that is palatable
- Math nerd

# Who am I?

- Data scientist that also cares about writing software that is palatable
- Math nerd
- Recovering academic

# Who am I?

- Data scientist that also cares about writing software that is palatable
- Math nerd
- Recovering academic
- Big fan of efficiency when possible

# Who am I?

- Data scientist that also cares about writing software that is palatable
- Math nerd
- Recovering academic
- Big fan of efficiency when possible
- In the data analysis space for more than a decade

# Who am I?

- Data scientist that also cares about writing software that is palatable
- Math nerd
- Recovering academic
- Big fan of efficiency when possible
- In the data analysis space for more than a decade
- I’ve spent a LOT of time doing things in very dumb ways

# Who am I?

- Data scientist that also cares about writing software that is palatable
- Math nerd
- Recovering academic
- Big fan of efficiency when possible
- In the data analysis space for more than a decade
- I’ve spent a LOT of time doing things in very dumb ways
- In my current role, I sit somewhere between analytics and dev

# Let’s start out a bit weirdly

Let’s walk through a bit of my career together, seeing all the ways you shouldn’t organize  dev and analytics teams.

1. This will be an instructive overview of what choices we should be making in designing infrastructure.

# Let’s start out a bit weirdly

Let’s walk through a bit of my career together, seeing all the ways you shouldn’t organize  dev and analytics teams.

1. This will be an instructive overview of what choices we should be making in designing infrastructure.

2. None of this is ground-breaking, but I think it’s something we, as a community, don’t give enough thought to.

# Let's start out a bit weirdly

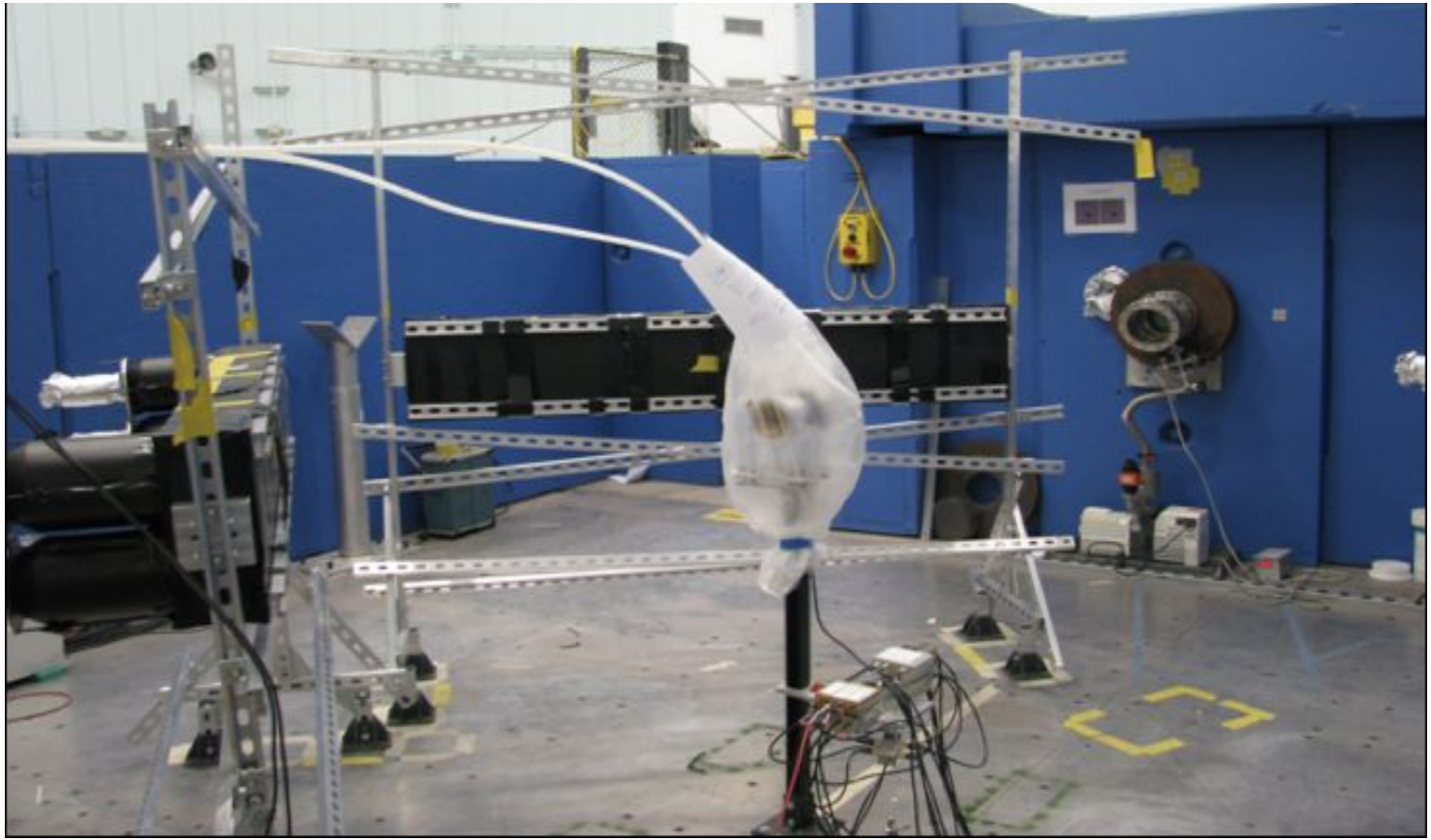
Let's walk through a bit of my career together, seeing all the ways you shouldn't organize  dev and analytics teams.

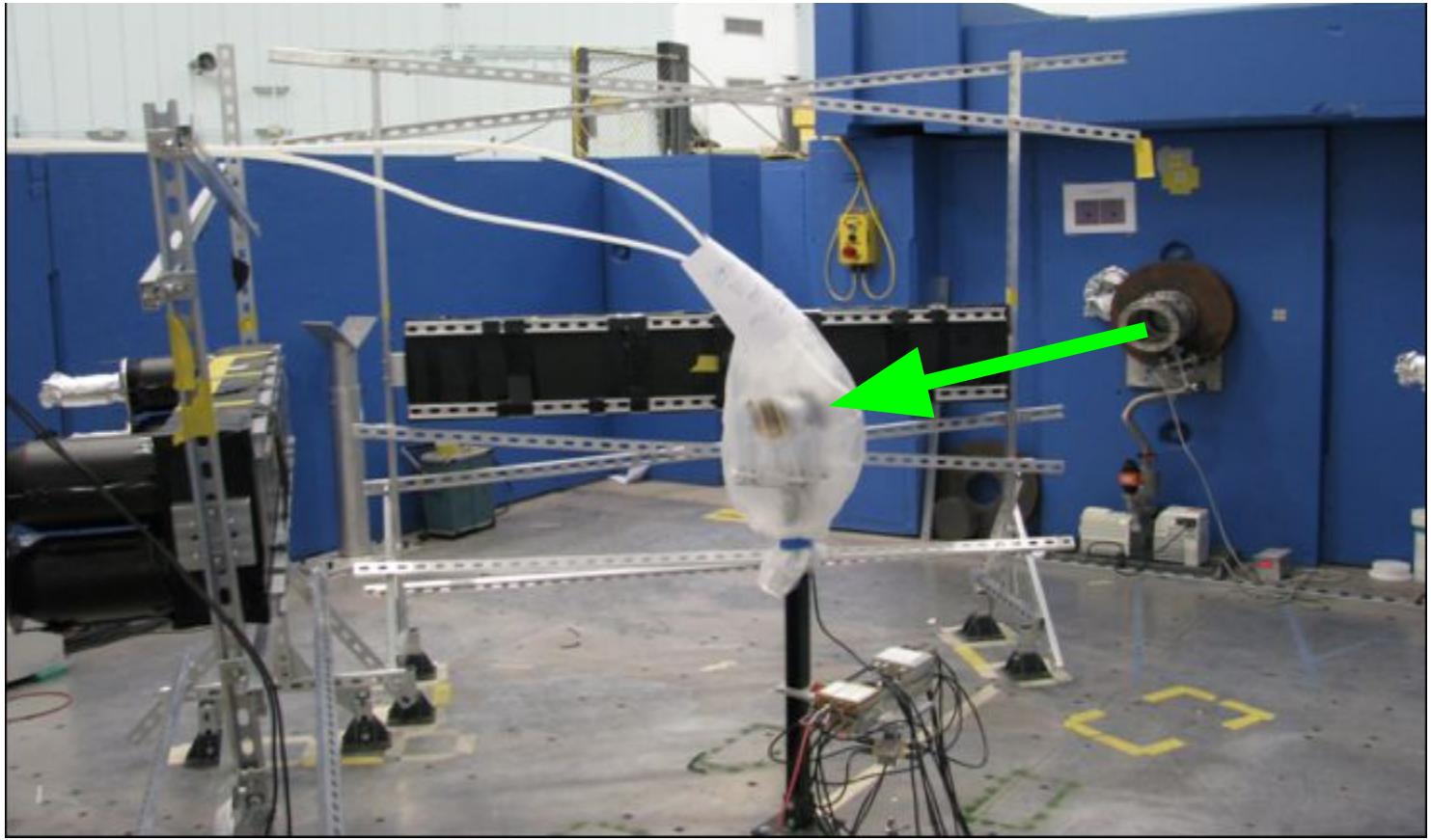
1. This will be an instructive overview of what choices we should be making in designing infrastructure.

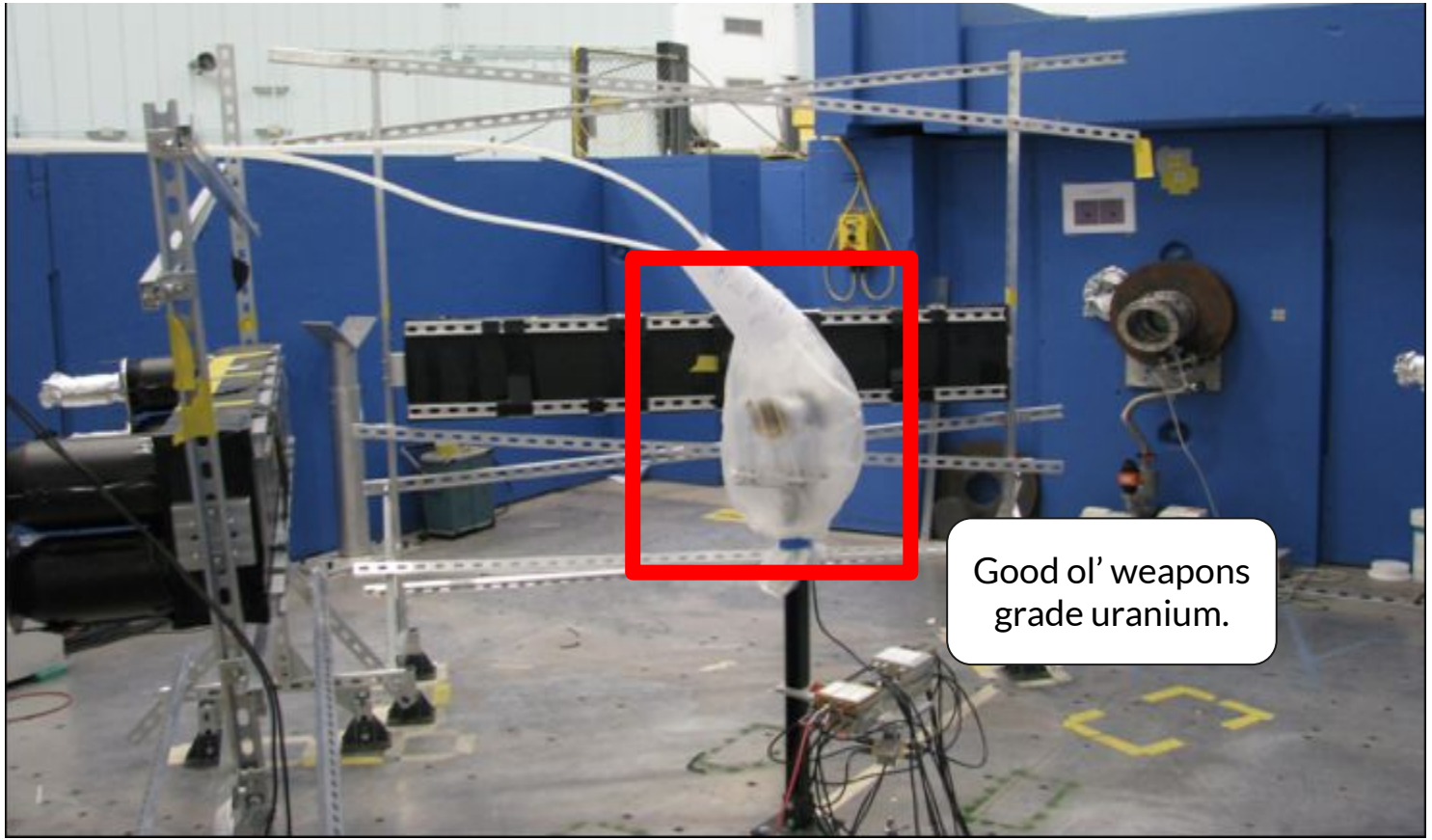
2. None of this is ground-breaking, but I think it's something we, as a community, don't give enough thought to.

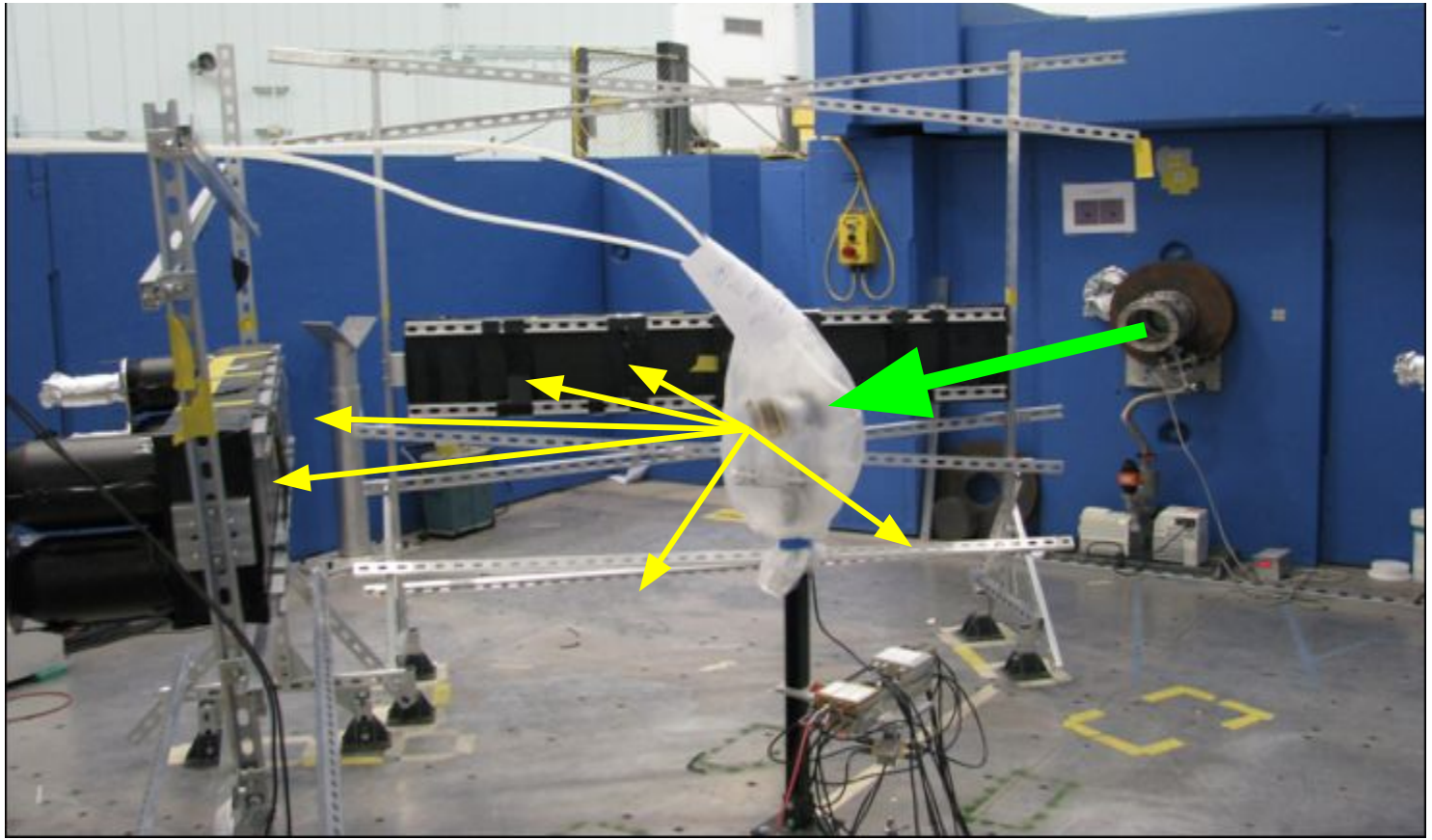
3. You're a captive audience that now has to listen to me talk about my dissertation, whether you want to or not.

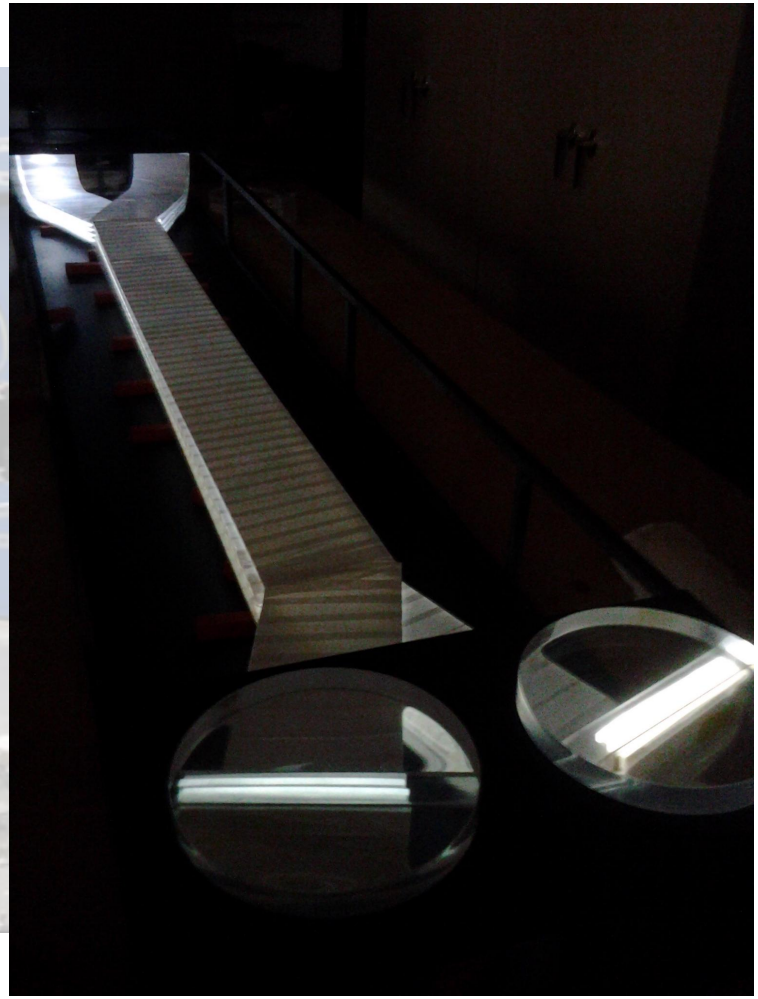
physics

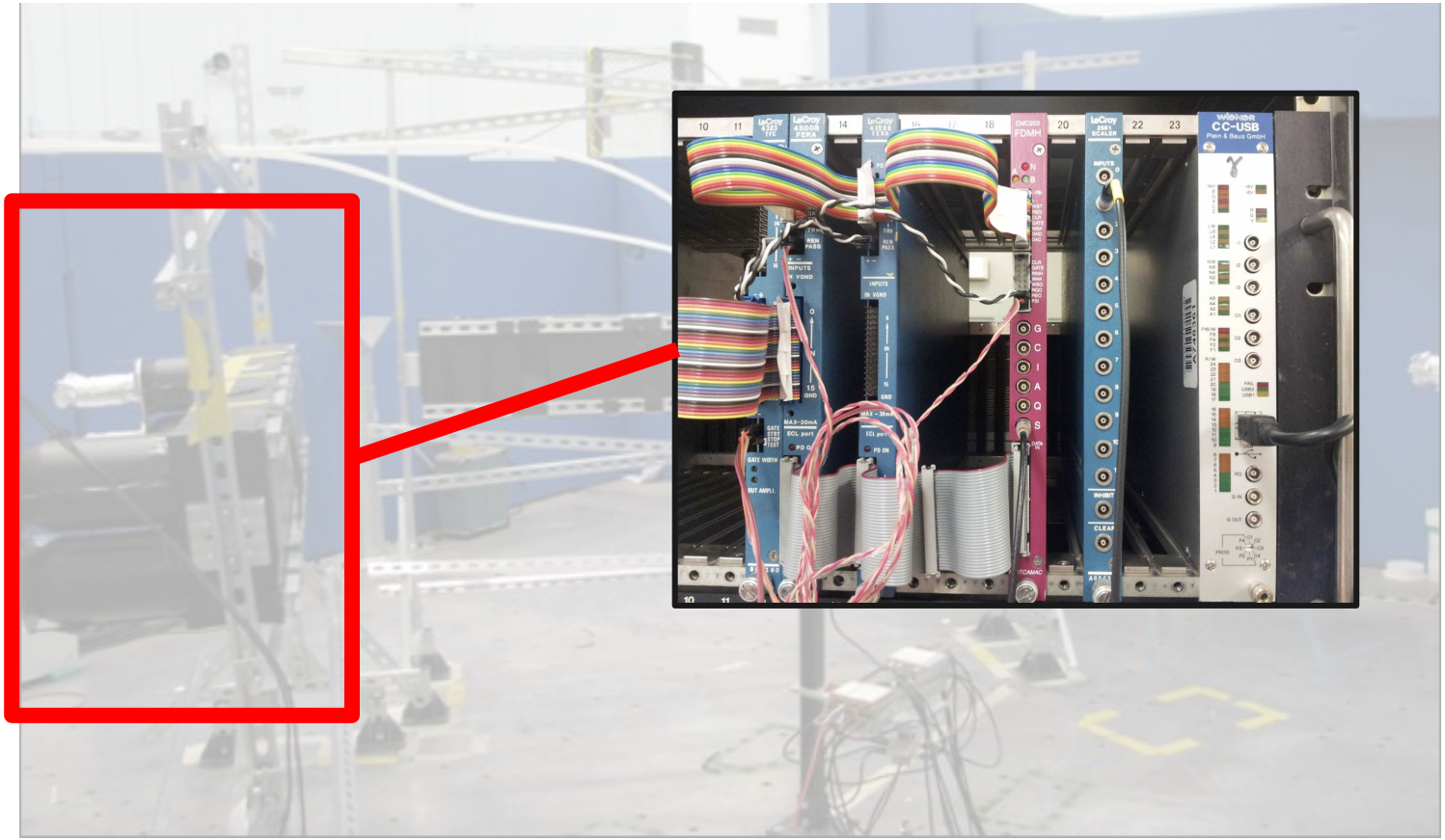





Good ol’ weapons grade uranium.

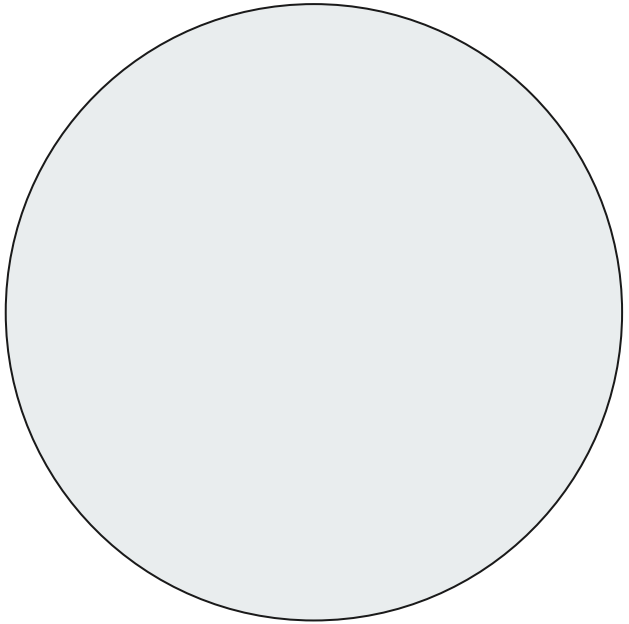





CC-USB

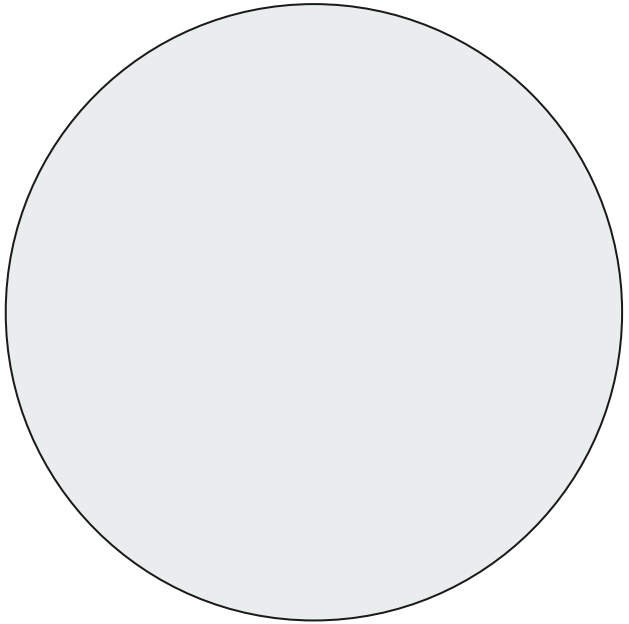
# A Venn Diagram about this research

People who give even the remotest of craps about this data.



# A Venn Diagram about this research

People who give even the remotest of craps about this data.



People who are me

# This allows me to make specific choices

1. All the data will live on a (backed up) hard drive
2. I'm going to store the data in a way that makes sense for my specific analysis
3. Screw metadata
4. Everything is numeric and I'm using an archaic piece of software to run my analysis, so I'm going to store the data how that software likes it

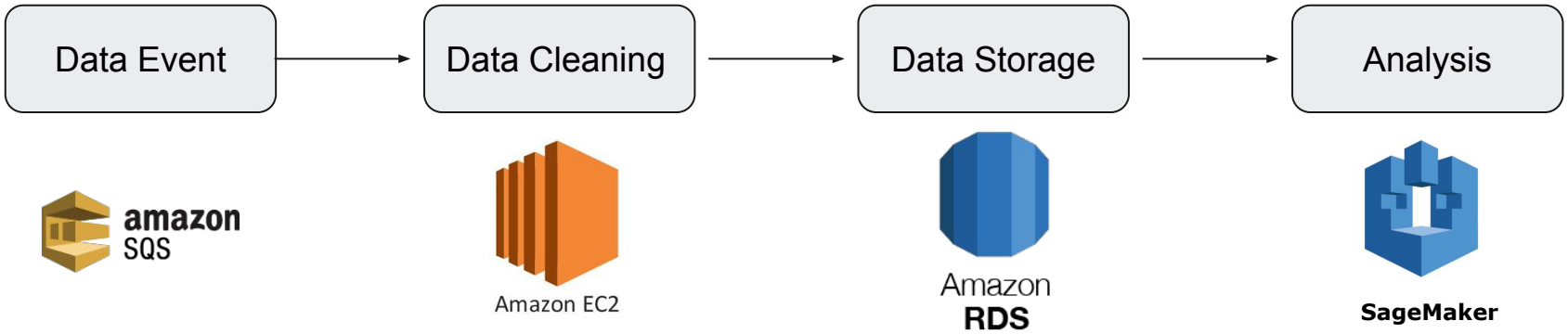
## This is essentially the “Young Business” Scenario

The Dev team is the Analytics team is the Engineering team is the database admin.

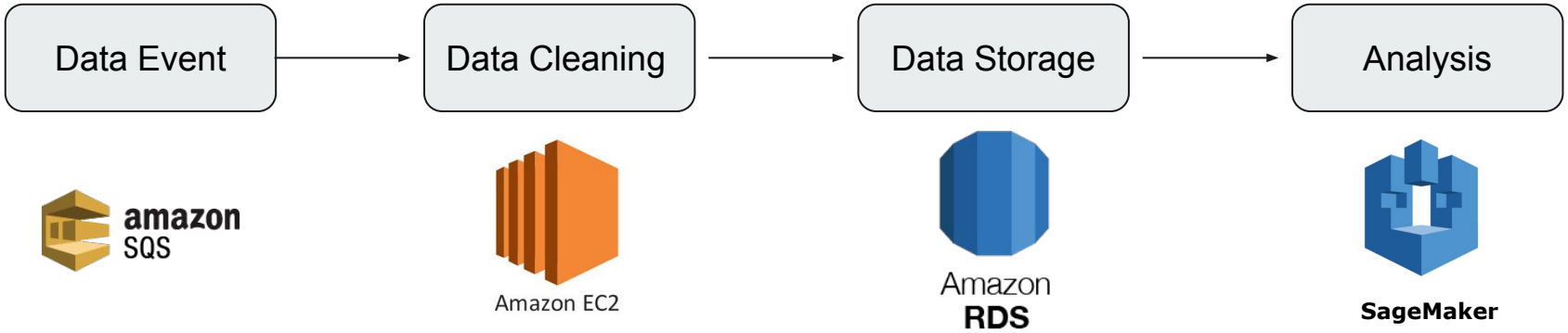
There is no difference between what the devs want and what the analysts want.

# Data Flow is simple in this case

# Data Flow is simple in this case



# Data Flow is simple in this case




This makes me happy inside because it's simple and elegant.

It's also extremely unrealistic for any real organization.

# Pros of this type of system

- Efficient for its purpose
- Little room for hand-shaking errors
- Few rabbit holes
- It has one job, and does that job well

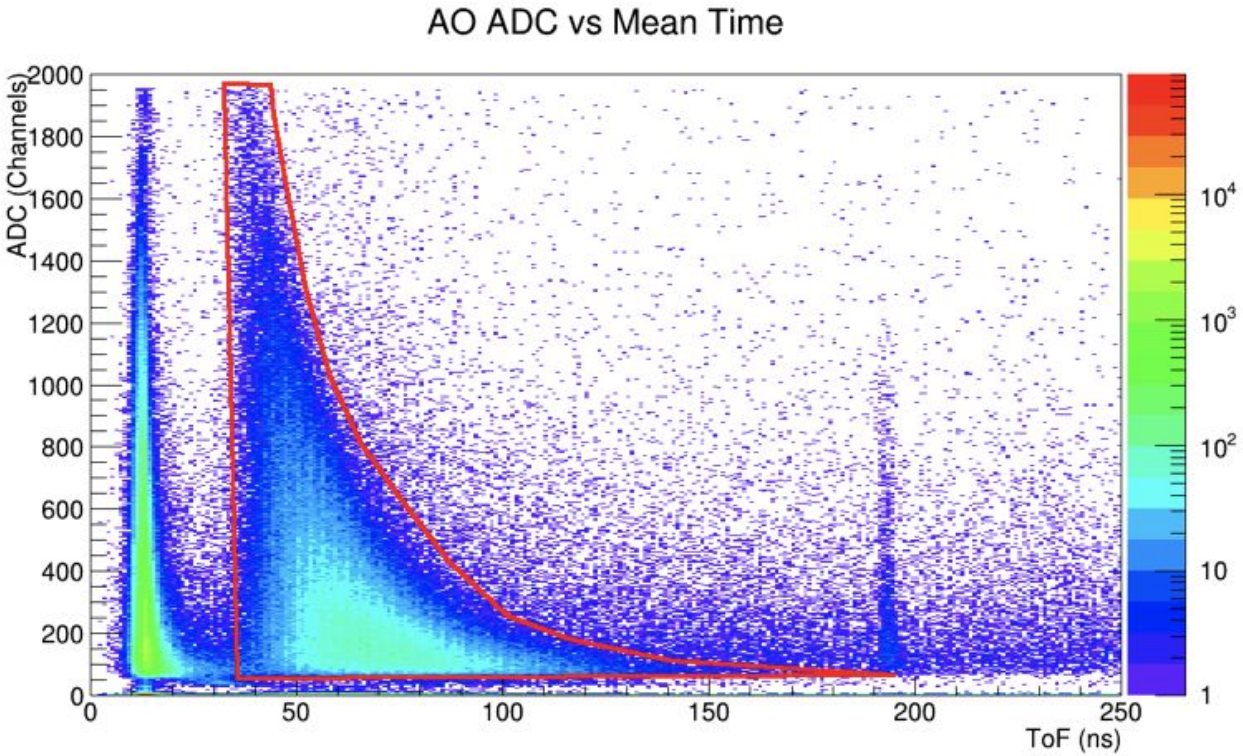
# Cons of this type of system

- Literally everything else

- Not scalable
- No room for flexible data usage
- Can’t exist outside a vacuum

# Before we move on…

# Before we move on… check out this sweet plot.



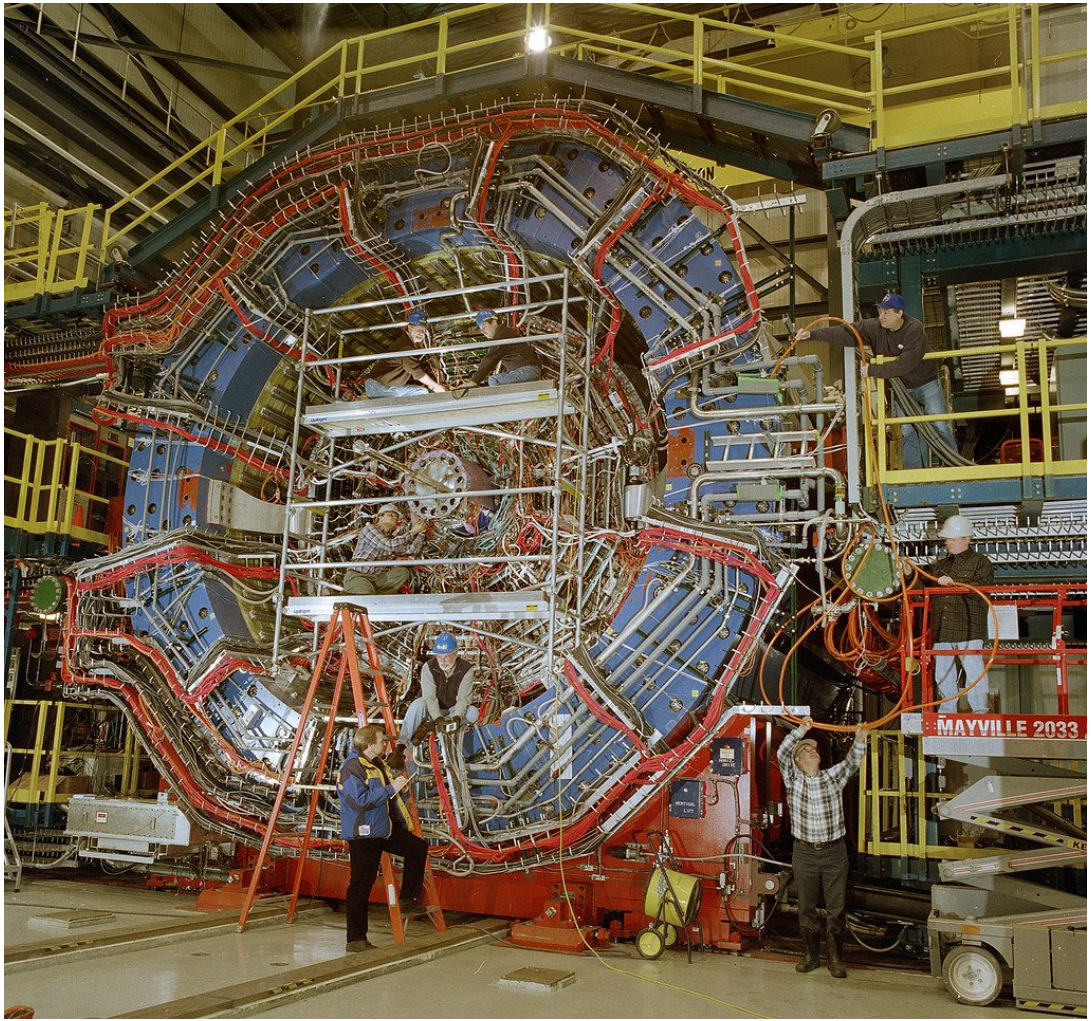

The ones in red are fissions that generated enough energy to be part of a nuclear chain reaction (if we had critical mass)

# At this point, I moved on to a larger organization where I was working with a much larger team

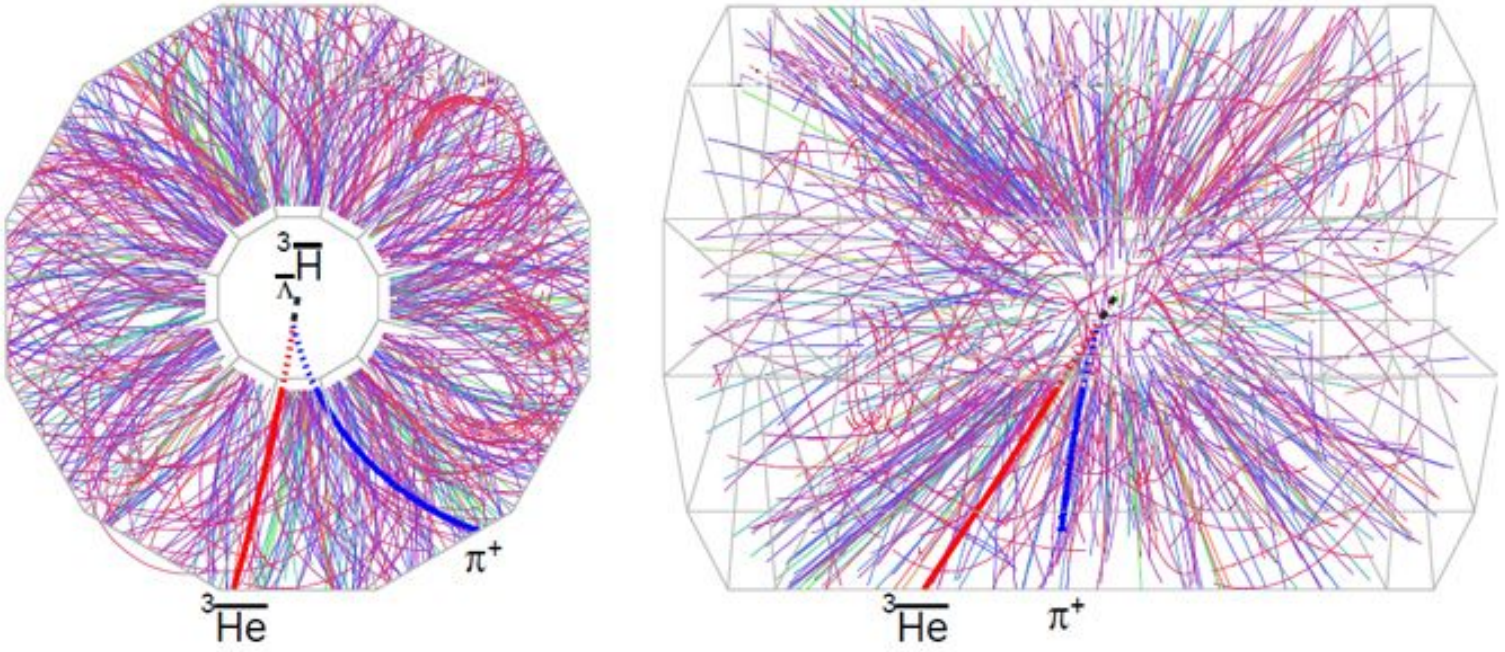
# STAR Detector

- 14+ Systems generating data
- 500+ people analyzing data
- 30+ institutions
- Generates several TB of data per day

- 1 really big nightmare for data management



# STAR Detector




Each line is constructed from about 150 data points. All of these lines happened for a single particle collision and only exist for about 12 ns. We recorded several thousand of these events per second, 24 hours per day, for several months per year.

# Two types of teams

Detector Teams

- Want ability to see live performance
- Need to store data in quick recall locations
- Don’t give a damn about format, just tell me if it works
- More interested in acquiring and actioning on the data than deciding what it means
- REALLY into JSON

# Two types of teams

Detector Teams

- Want ability to see live performance
- Need to store data in quick recall locations
- Don’t give a damn about format, just tell me if it works
- More interested in acquiring and actioning on the data than deciding what it means
- REALLY into JSON

Analysis Teams

- Want the data to be stable long term
- Care a lot about “preserving the integrity of the data”
- PUT IT IN THE FORMAT I NEED OR I’LL CUT YOUR CHILDREN
- Worried about small fluctuations in how the data is stored and whether it’s meaningful
- More concerned about the event itself than the tools used to process it

# Two types of teams

Detector Teams

- Want ability to see live performance
- Need to store data in quick recall locations
- Don’t give a damn about format, just tell me if it works
- More interested in acquiring and actioning on the data than deciding what it means
- REALLY into JSON

Analysis Teams

- Want the data to be stable long term
- Care a lot about “preserving the integrity of the data”
- PUT IT IN THE FORMAT I NEED OR I’LL CUT YOUR CHILDREN
- Worried about small fluctuations in how the data is stored and whether it’s meaningful
- More concerned about the event itself than the tools used to process it

# Two types of teams

***COUGH* DEV *COUGH***

- Want ability to see live performance
- Need to store data in quick recall locations
- Don’t give a damn about format, just tell me if it works
- More interested in acquiring and actioning on the data than deciding what it means
- REALLY into JSON

Analysis Teams

- Want the data to be stable long term
- Care a lot about “preserving the integrity of the data”
- PUT IT IN THE FORMAT I NEED OR I’LL CUT YOUR CHILDREN
- Worried about small fluctuations in how the data is stored and whether it’s meaningful
- More concerned about the event itself than the tools used to process it

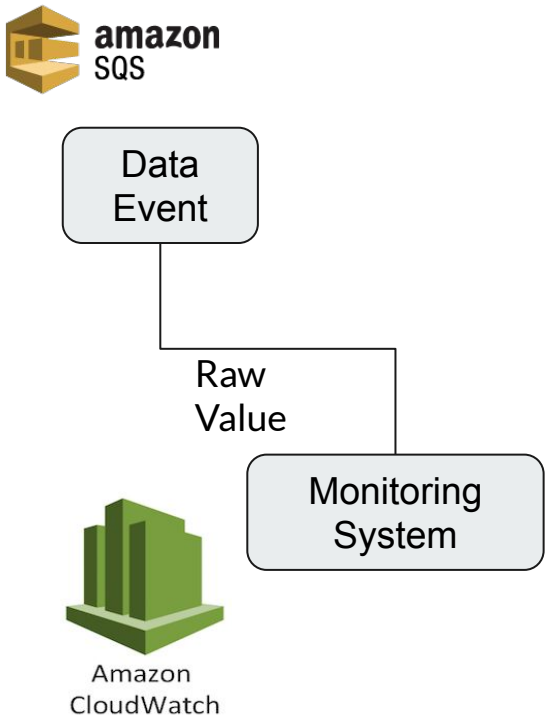
# Knowing full well that our Dev team is in the audience and have a large amount of control over the data I need...

- I'm not saying one side is right or wrong. They do have often competing priorities though. So the rest of this talk is going to be about how systems can be designed to accommodate those priorities, and how AWS gives us a lot of tools to make that happen.

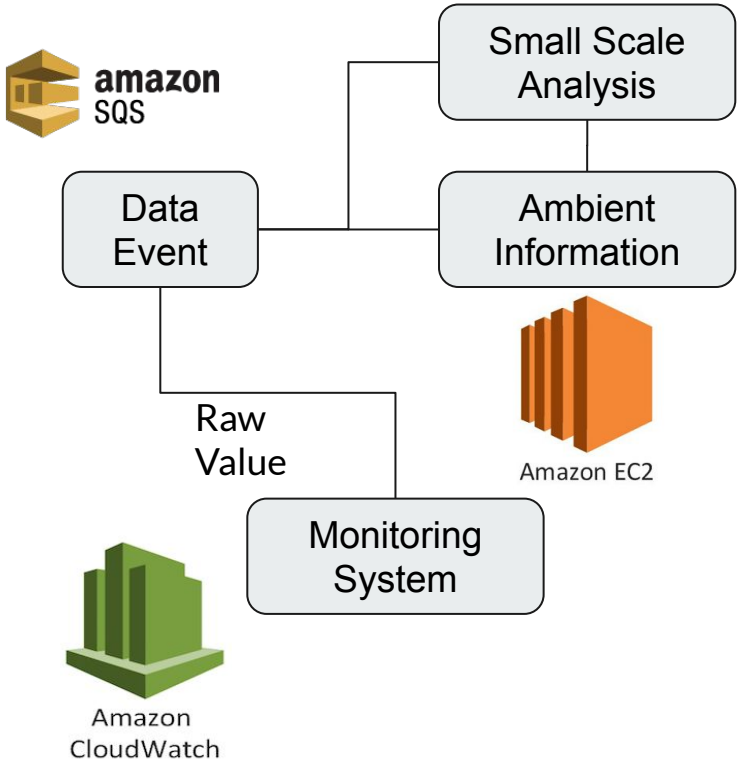
# System A (as designed by detector teams)


Data
Event

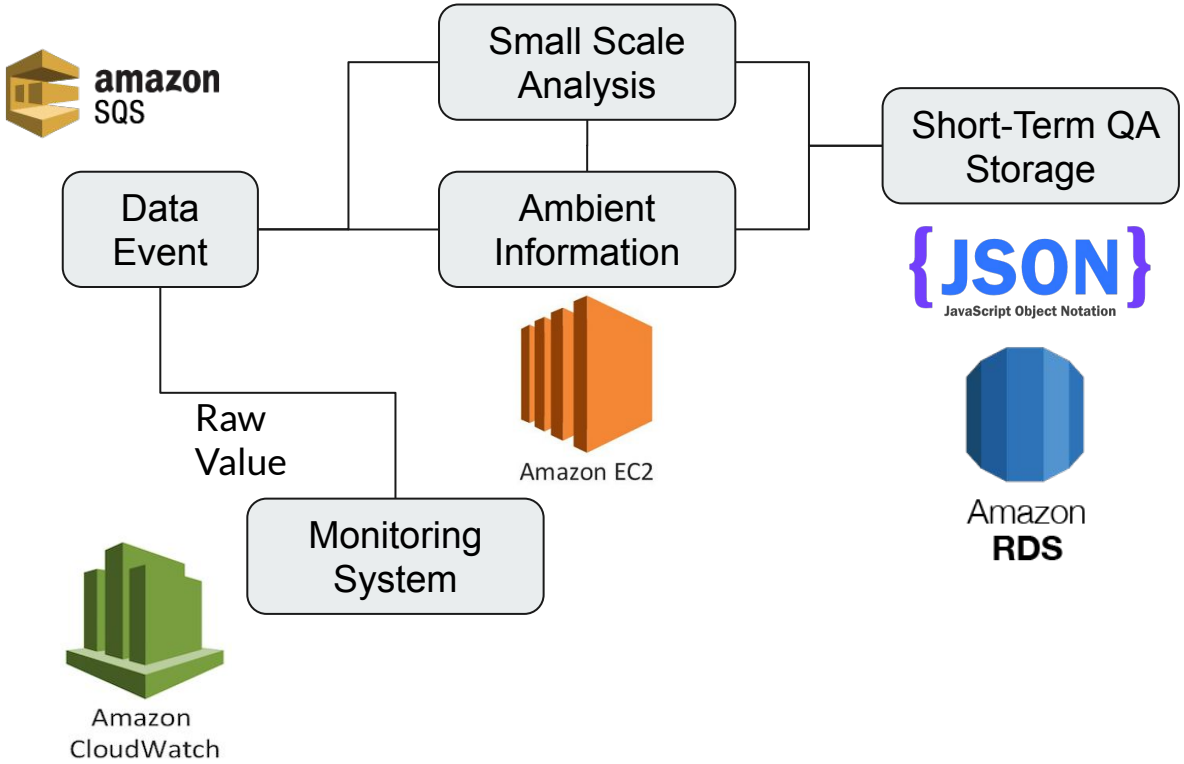
# System A (as designed by detector teams)



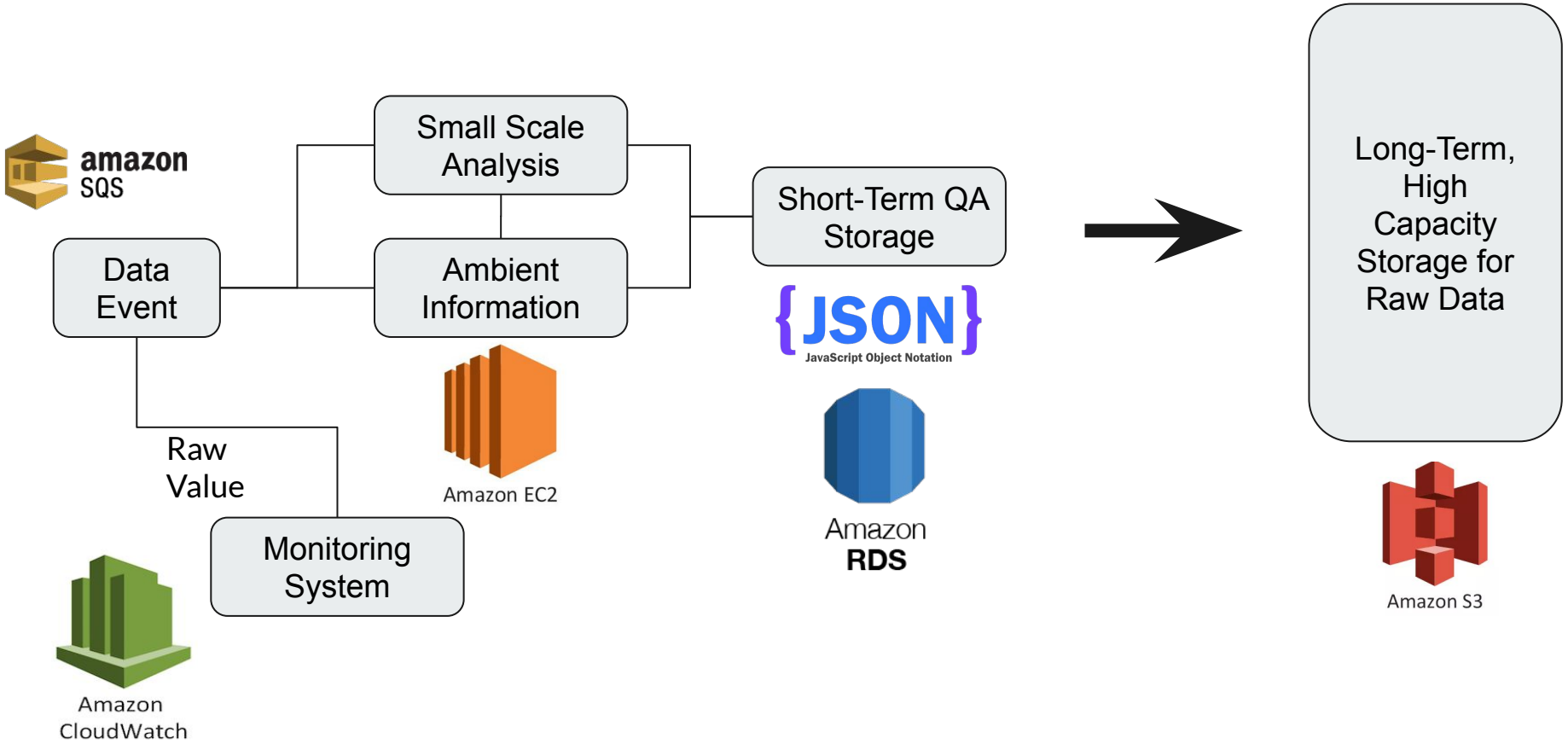
# System A (as designed by detector teams)



# System A (as designed by detector teams)



# System A (as designed by detector teams)



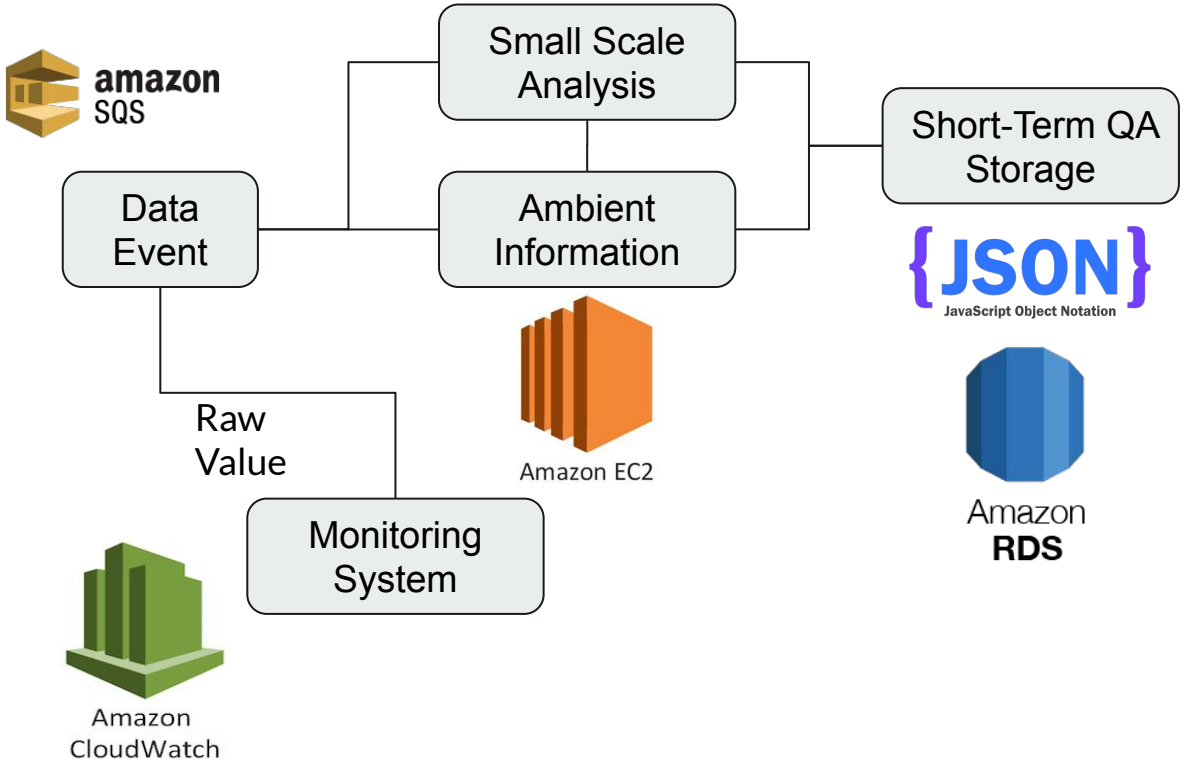
# Data flow becomes complicated

In this type of system, the devs are “creating” or “extracting data” from consumers, and the analysts have to then make sense of that data.

The way the devs handle data is different than the way analysts might want to.

The system is complicated, so it’s not straightforward to just go to a table and say, “tell me what you know.”

# System A (as designed by detector teams)




Dev Concerns:

All of this needs to run in 500ms or so, all I have time for is some cursory checks and then the next event must be processed.

I don't want to maintain 25 different schemas that change whenever the data does.

# System A (as designed by detector teams)

Analytics Concerns:

Does any of this manipulation change the raw data?

Will the data be stored in a format where I can process the data and use it to build models?

Long-Term, High Capacity Storage for Raw Data

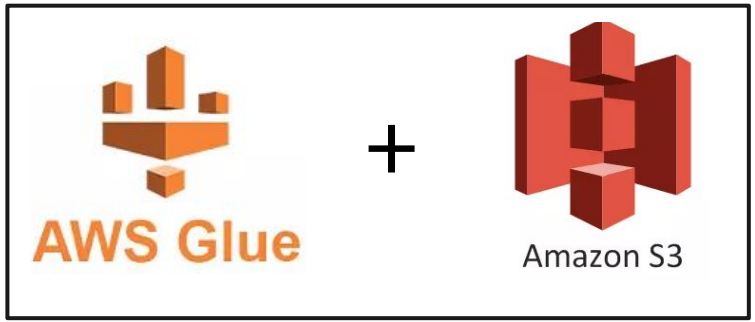
Amazon S3

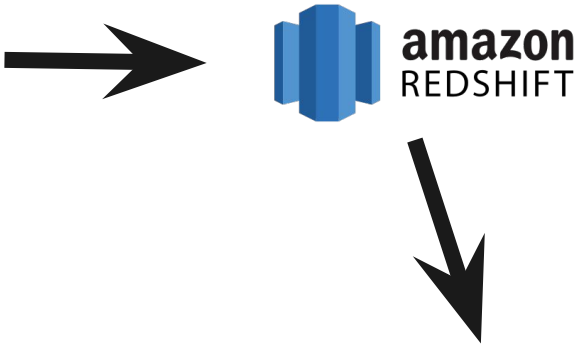
# ETL is your friend


The competing priorities make sense. Neither team should give up the ability to be efficient.

Create a middle man to solve the issue. Glue can read, unpack, and generally do whatever ETL you need in order to translate between the two environments.

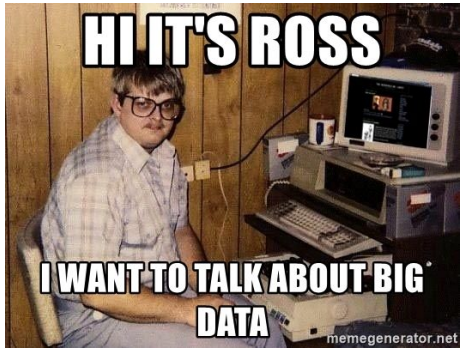
# ETL is your friend




DEV
Team

Analytics
Team

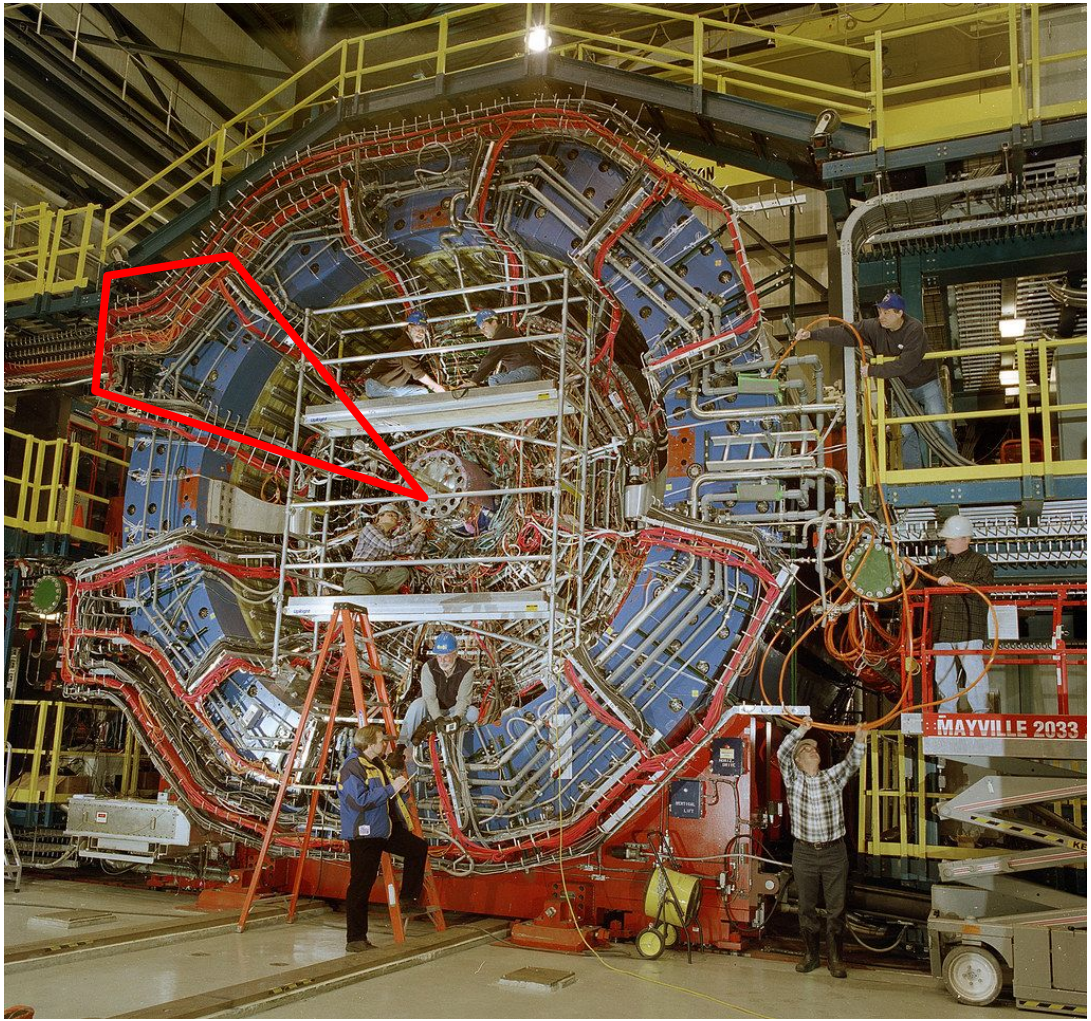


# So far we've talked about a single system

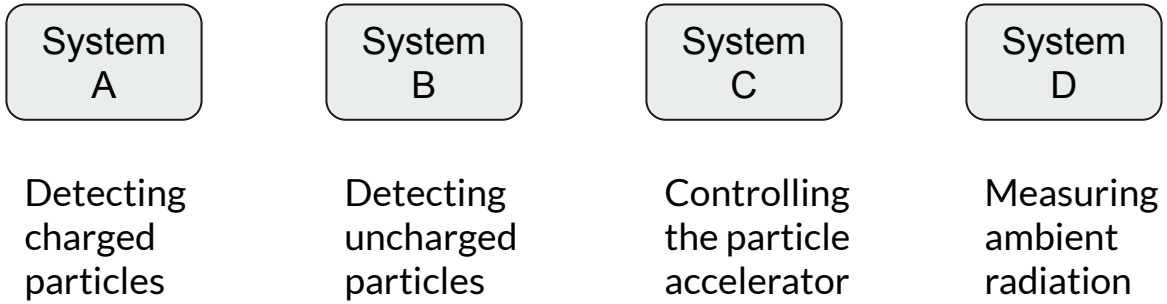
No business (or research group) runs on a single system.

- 14+ systems
- ~1,000,000 sensitive detectors per system

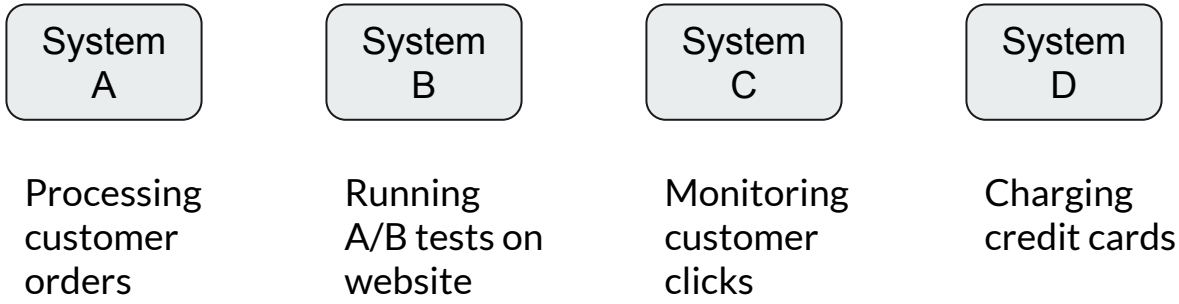
How the hell do we merge all of that?



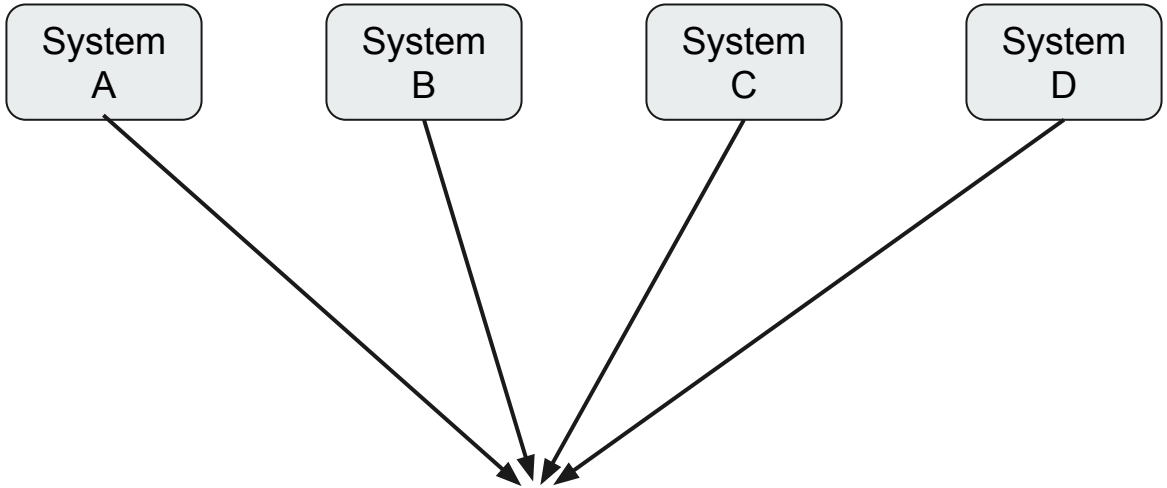
# The whole system



# The whole system

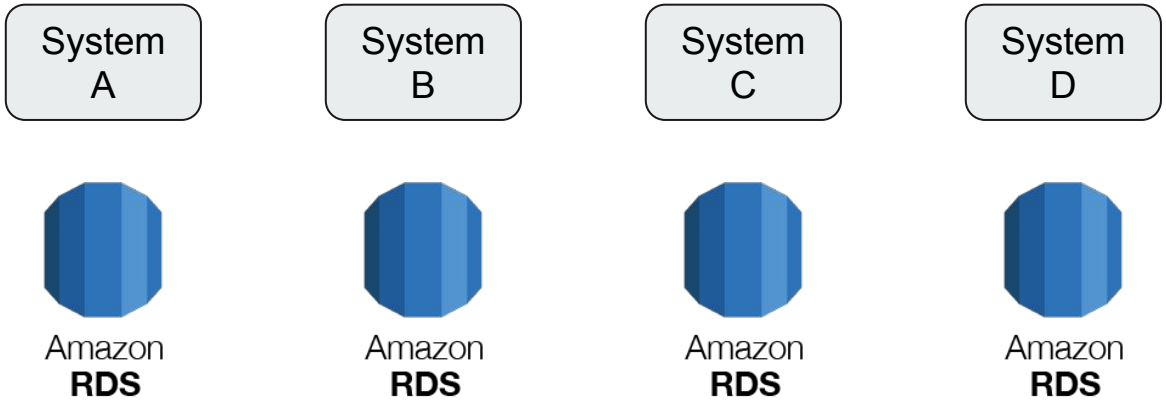


# The whole system (as seen by analytics)

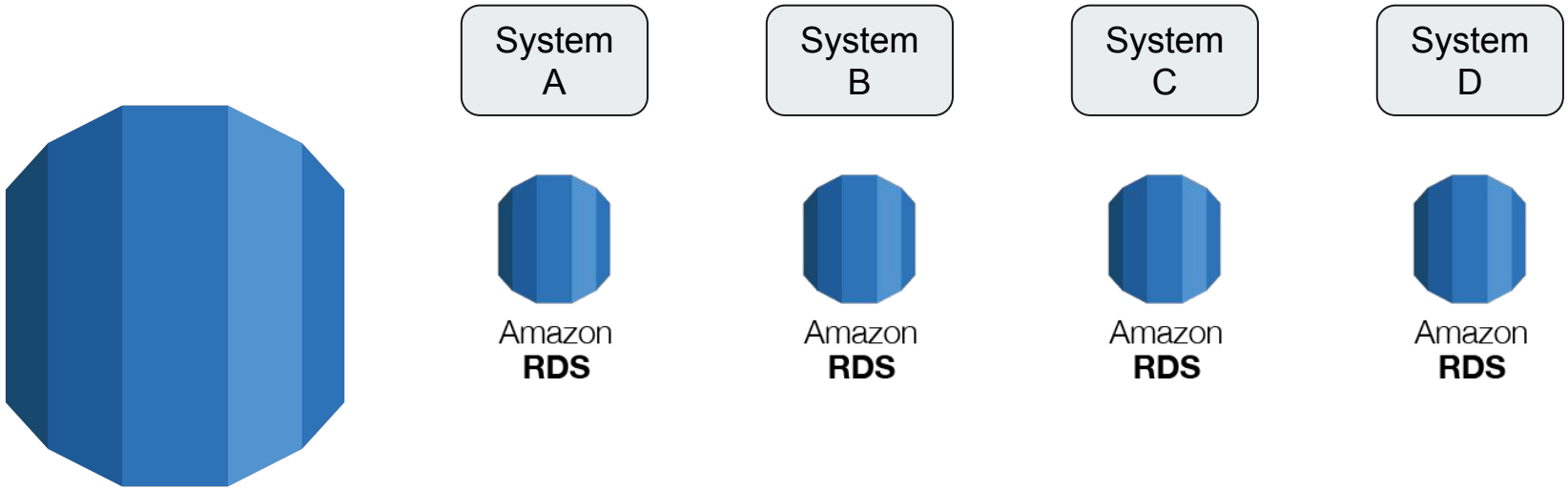



| A_feat1 | A_feat2 | A_feat3 | B_feat1 | B_feat2 | B_feat3 | B_feat4 | ... | D_feat9 |
|---|---|---|---|---|---|---|---|---|
| 12 | 0.5 | 11 | Reginald | 42 | 11teen | 7.5 | ... | True |
| 11 | 0.0001 | NaN | Steven | 42 | C100 | 9.6 | ... | False |

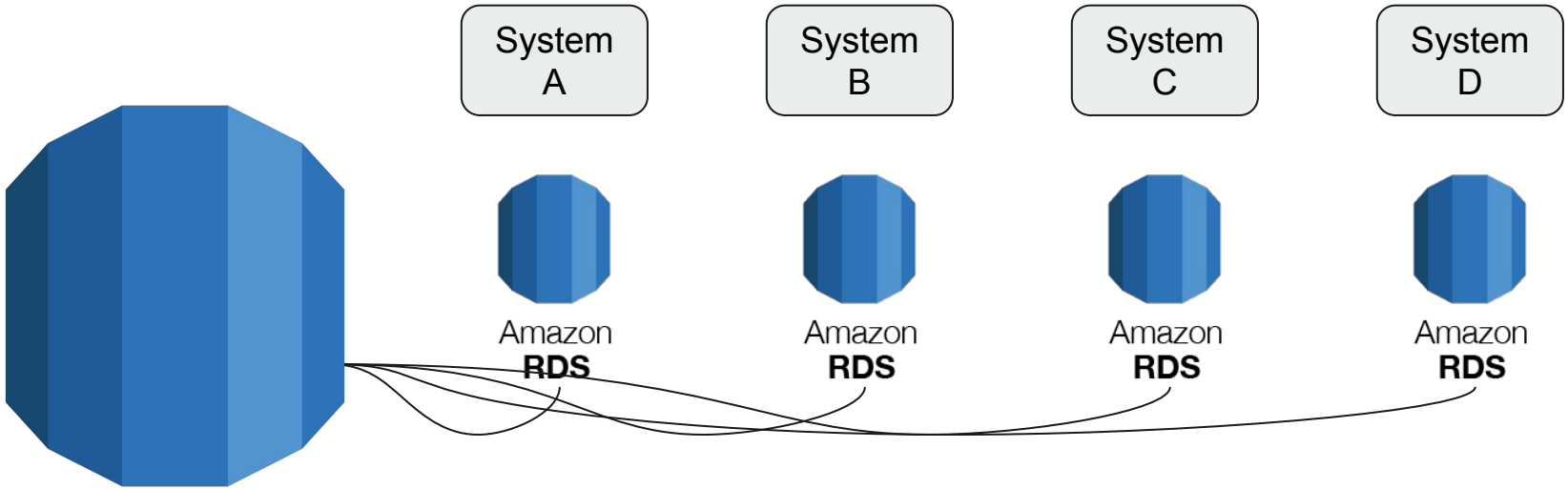
# The whole system



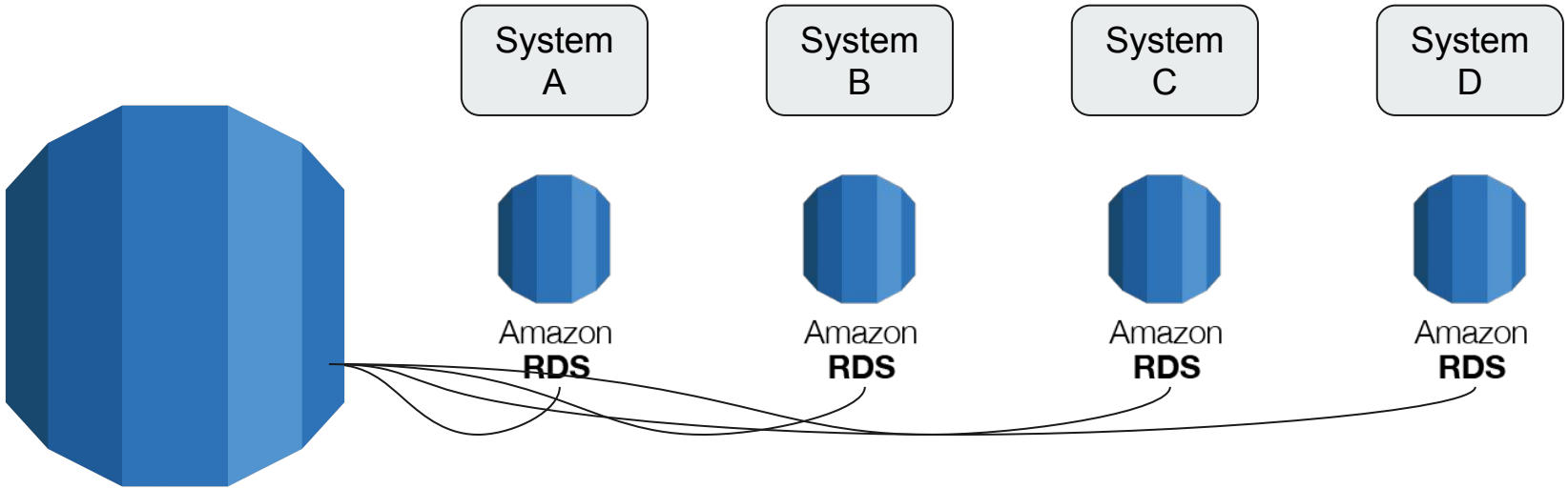
# The whole system



# The whole system



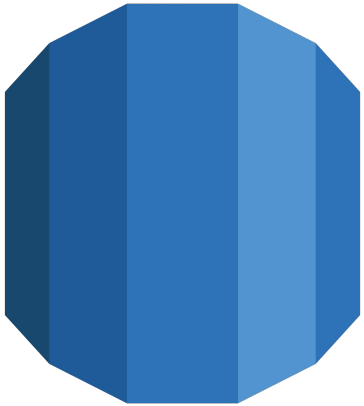
# The whole system

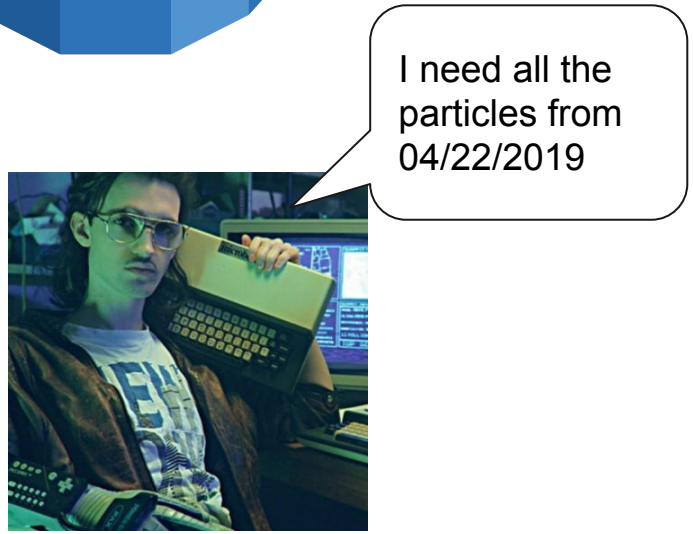



Storage is typically cheap (now-a-days), and many-to-many join systems can make systems infinitely more navigable.

Having a master database (or table) that knows about the doings of all of the systems through clever logging can make a system usable, scalable, and accessible to folks with the ability to SQL.

# Develop Tooling to Maximize Efficiency

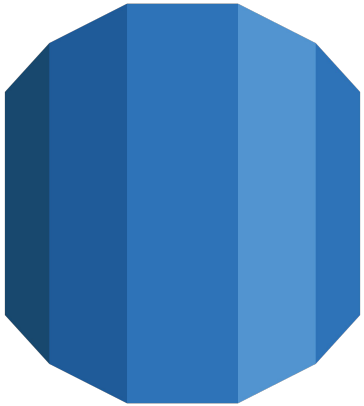




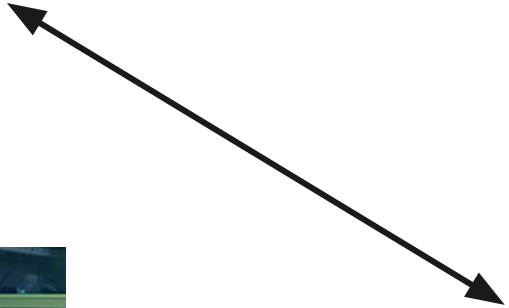
The Data Retrieval Tool That Actually Works Sometimes™

# Develop Tooling to Maximize Efficiency





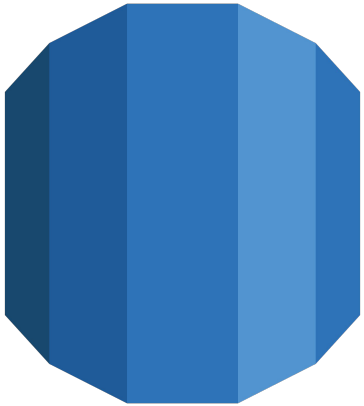


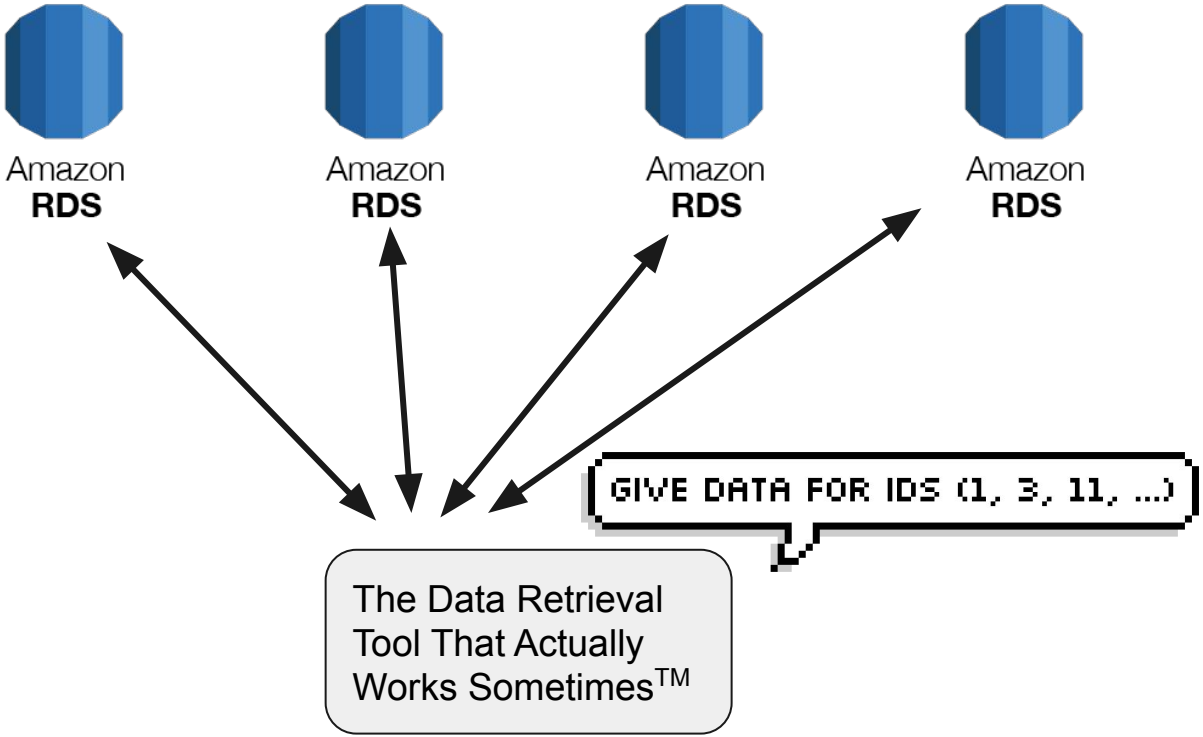


WHICH IDS HAPPENED ON 4/24/19?

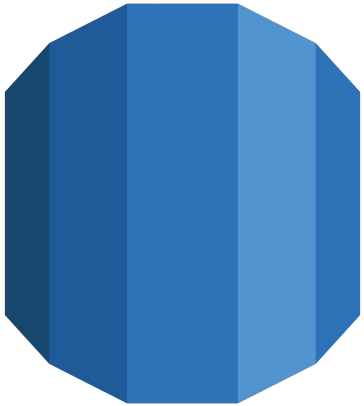
The Data Retrieval Tool That Actually Works Sometimes™

# Develop Tooling to Maximize Efficiency





# Develop Tooling to Maximize Efficiency







| | | |
|---|---|---|
| imp | ort | ant |
| ma | ny | nu |
| mb | ers | here |


The Data Retrieval Tool That Actually Works Sometimes™

# Develop Tooling to Maximize Efficiency

There will be dev work in the TDRTTAWS™ in order to make sure that the analysts aren't spending their time fighting unnecessary battles they aren't the best team to solve.

That is valuable time and work.

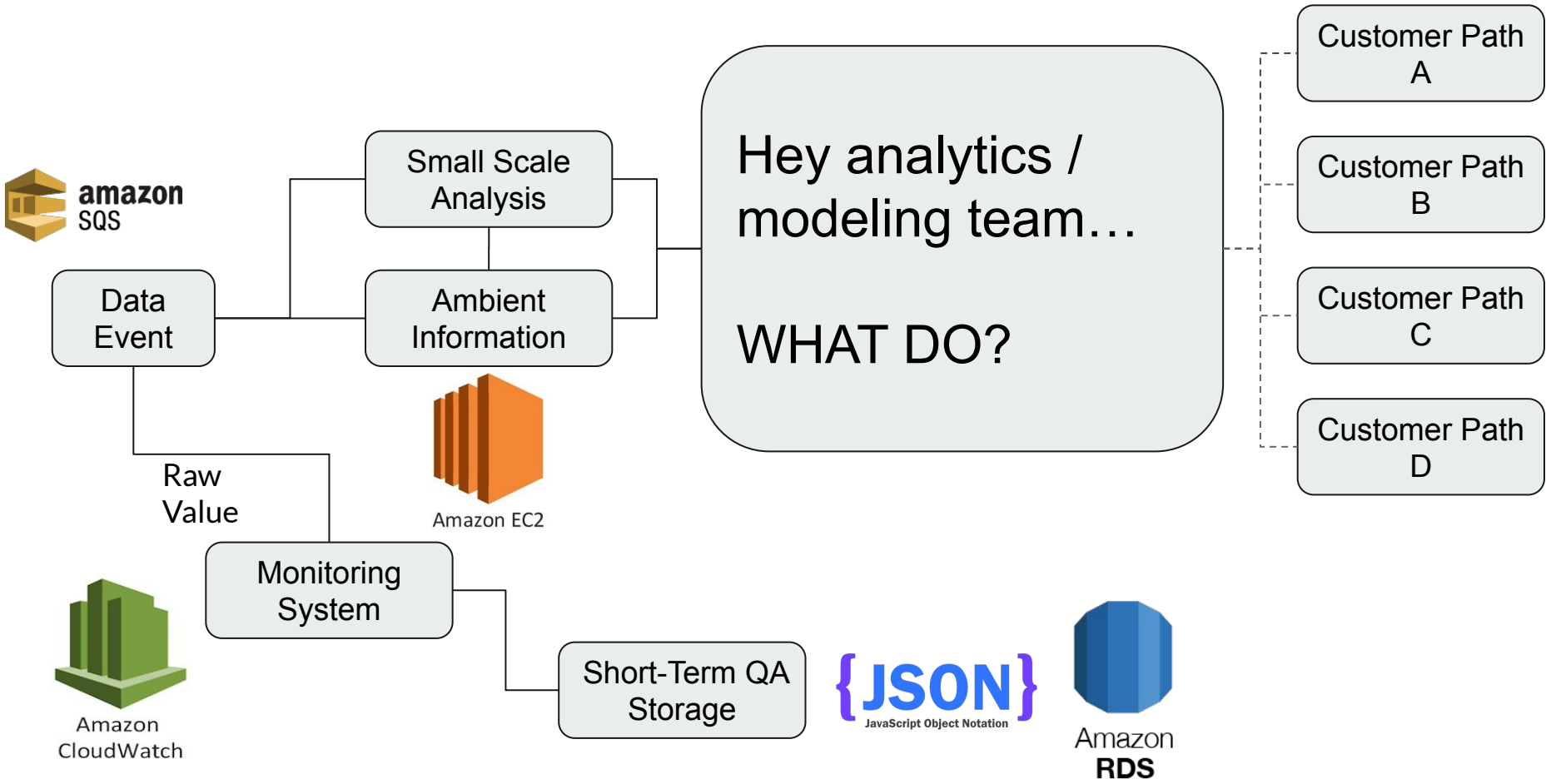
The Data Retrieval Tool That Actually Works Sometimes™

# And what should analytics be doing?

So far, we've largely talked about architecting your system to minimize conflict between dev and analytics from the dev side.

Analytics isn't innocent in this 'battle' though… let's return to our System A and make it a bit less physics-y

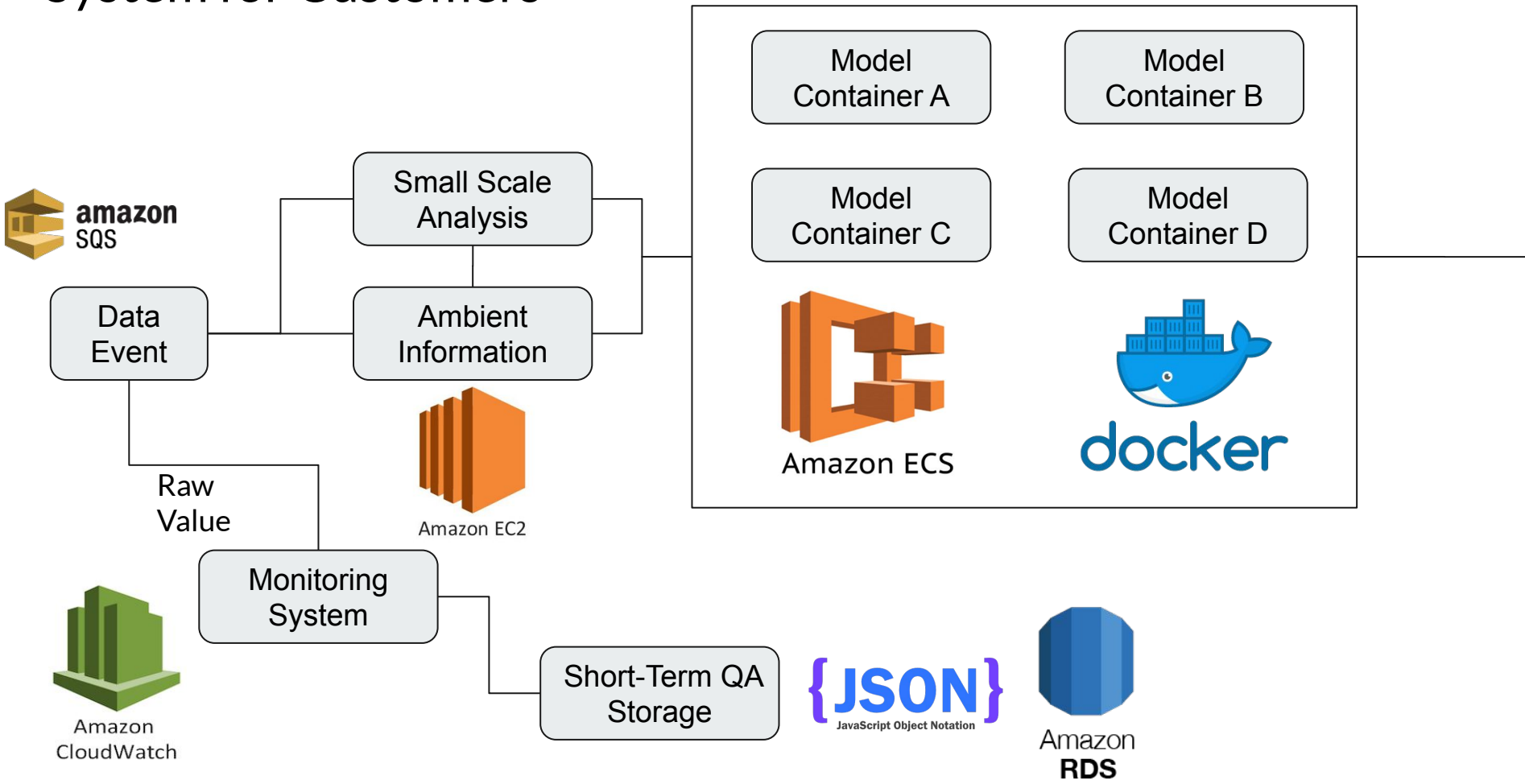
# System for Customers



# Culture Shift

- At some point in the last few years, data scientists have gotten a bit diva-y.
- “I only make the models/decisions, I don’t have to worry about how they get deployed.”
- Analytics teams: meet microservice ideology

# System for Customers



# Microservices

- This style of infrastructure has drawbacks, but it's the best way to make sure that many teams can play together without fighting.
- If every piece is a module, then analytics can design the modules they need, and dev can swap them in and out.
- Each piece of data management becomes a battery pack that one team can own.
- This can be models, ETL, database cleaning, whatever

# Summarizing

Devs want…

The customer to flow through the service/product neatly

Fast response time, no errors

Useful logging

Not to bother with a bunch of finnicky crap to make the data “just right”

Analysts want…

Good, clean data that truly represents the consumer

Data that’s in a format that is easy to use for analysis

The ability to get their discoveries into production

A platform that supports their ability to access and convert data into insights

# Summarizing

Both want…

A platform that isn’t a hassle, and gets “out of the way” to let them do their job.

Data to be collected consistently, and errors to be minimal.

Infrastructure that doesn’t make them weep.

And the business to succeed

# Summarizing

- There are two (or more) sorts of systems:
  - The dev is the analyst
  - The dev ‘creates’ data for the analyst to consume
- Respecting these competing priorities makes for a better system
- Creating ‘middlemen’ to translate between these priorities is wise
- ETL is your friend
- Designing tooling is your friend
- Strong database design is your friend
- Microservices are your friend

# There's way more stuff you can be doing

- Using shared cluster computing environments for ETL and analytics work helps maintain a consistent environment (AWS EMR)
- Maintaining a data lake on S3 with just the raw data seen by Dev means there's always a ground truth (AWS Athena/S3/Glue)
- Long term data storage for rarely accessed data (Glacier)
- Serverless architecture to make the dev requirements lighter on the analytics team (Lambda)
- Using SageMaker deployments to make model deployment self-managed by analytics (Sagemaker)
- Etc etc.

# Questions?

Note: CreditNinja is currently hiring Data Scientists and Devs