Presented By
Chad Sanderson

dataquaiity.camp/slack

DATA CONTRACTS: ACCOUNTABLE DATA QUALITY

ABOUT ME

+++

Hello,
I am Chad Sanderson

Owner, Data Quality Camp

- AI Platform Team at Microsoft
- Led Data Platform Team at Convoy
- Co-wrote "The Engineer's Guide to Data Contracts"
- Created the Data Quality Camp Slack Group

— dataquaiity.camp/slack

WHAT I CAME TO DO

- Build the data platform team! (Specifically ML & Experimentation)
- Help develop novel applications of AI/ML on behalf of product teams
- Launch the infrastructure to support ML at massive scale

WHAT I ACTUALLY DID...

- Constant firefighting
- Maintaining 100s of lines of spaghetti SQL
- Yelled at by data consumers for things breaking (not my fault)
- Explain why ML & AI initiatives weren't scaling
- Try (and fail) to hire more data engineers

OBSERVED PROBLEMS

Constant Failures	Breaking changes (and more insidious logic changes) were endless
Replication	Data was exponentially duplicating. Teams didn’t trust each other’s work
Alert Fatigue	Lots of monitors, very little action. Testing was rarely if ever useful
Painful On-call	Data engineering on-call was a ngihtmare- constant unexpected outages
Bad Relationships	Data engineering was always to blame, even when it wasn’t their fault.

ROOT CAUSE #1: VISIBILITY

- Data producers had no visibility into the downstream usage of their data.
- Data consumers had very little visibility into upstream changes
- DE became a bottleneck
- **No one understood how each team was affected by the other**

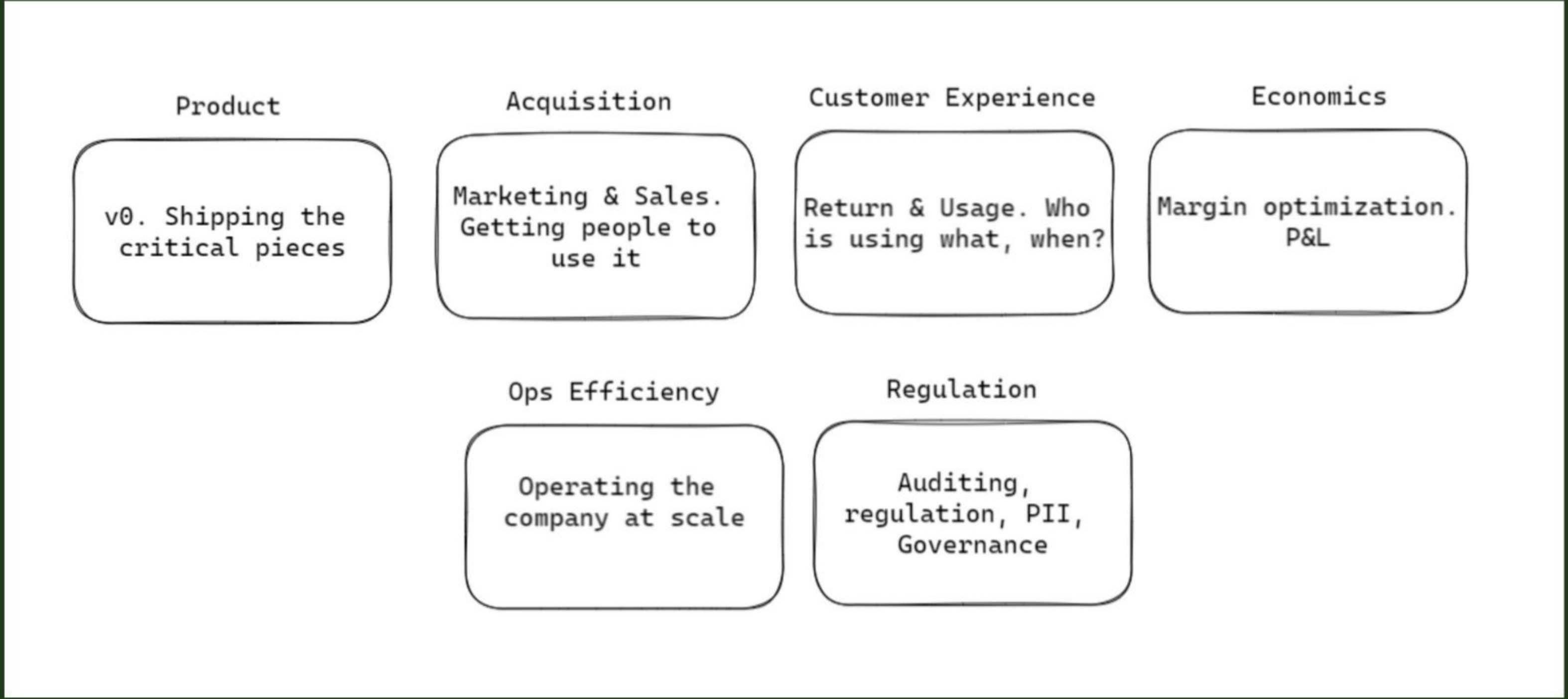
ROOT CAUSE #2: PIPELINE EVOLUTION

- As data changed, complex filters are added on top existing queries
- The data model often failed to evolve iteratively with the business
- These two issues resulted in growing spaghetti SQL
- **This SQL was painful to parse and resulted in recreating wheels**

ROOT CAUSE #3: OWNERSHIP (OR LACK THEREOF)

- Producers didn’t truly own the data they emitted from analytics/ML sense
- Due to point #2, OBT often emerged which also lacked ownership
- High value datasets were built leveraging unowned tables
- **When something important broke, no one raised their hand to fix it**

BUSINESS MATURITY CURVE



Product

v0. Shipping the critical pieces

Acquisition

Marketing & Sales. Getting people to use it

Customer Experience

Return & Usage. Who is using what, when?

Economics

Margin optimization. P&L

Ops Efficiency

Operating the company at scale

Regulation

Auditing, regulation, PII, Governance

JUST GETTING STARTED

Data teams are initially created to support product and growth functions. Teams desperately needed some data, but not everything was equally important.

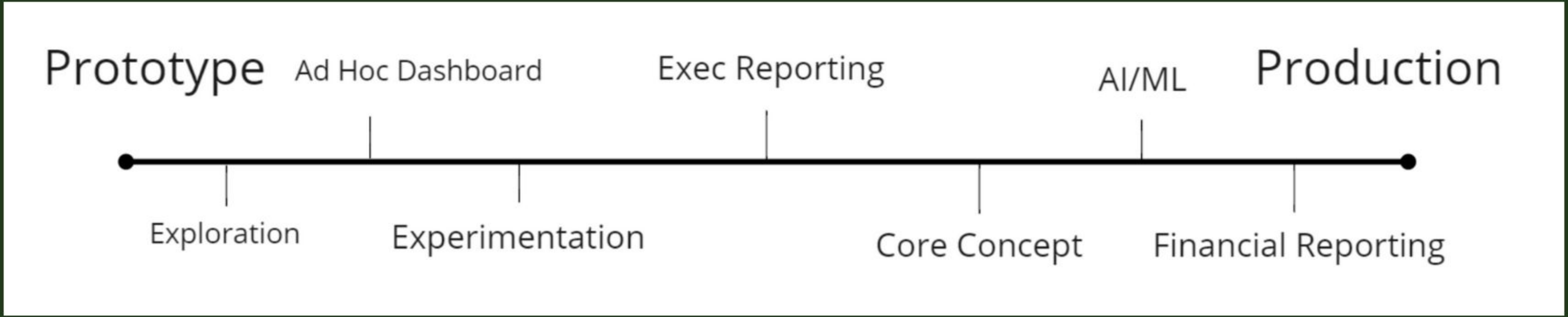
- Data Quality is useful, but low direct ROI
- Directional data is OK
- The data is team is small
- No need for a comprehensive data model
- The tool sets and processes are scrappy! (just plug and go)

AT SCALE

Data infrastructure becomes a significant cost. Teams spend big on headcount, compute resources, and tools. The ROI of this architecture is questionable. Teams look for more meaningful use cases to make data investments.

- Directional data is no longer acceptable
- The scrappy data model does not scale
- The maintenance cost is massive for a support function
- Data Quality is a necessity, not a nice to have
- If things break, the business suffers

DATA MATURITY CURVE



Teams often use the modern data stack to get started quickly spinning up data pipelines for ad hoc analysis, but it is challenging to scale to production grade use cases.

SUMMARY

+++

Data as Support
BI & Analytics

- Extremely important early on
- Helps justify product & marketing spend
- Must be created for speed
- Directionality is acceptable
- (At first) doesn't require model complexity
- Ideally inexpensive
- Challenging to justify large cost at scale

Data as Product
AI/ML & more...

- Extremely important later on
- Can generate ROI alone, not support
- Must be built for durability and scale
- Directionality is not acceptable
- Definitely requires a complex data model
- Expensive, but with measurable value
- Easy to justify large cost at scale

THE MODERN DATA STACK

dbt Apache Airflow tableau

Fivetran snowflake Metabase

Move fast and
break things!

Teams often use the modern data stack to get started quickly spinning up data pipelines for ad hoc analysis, but it is challenging to scale to production grade use cases.

PROS VS CONS

+++

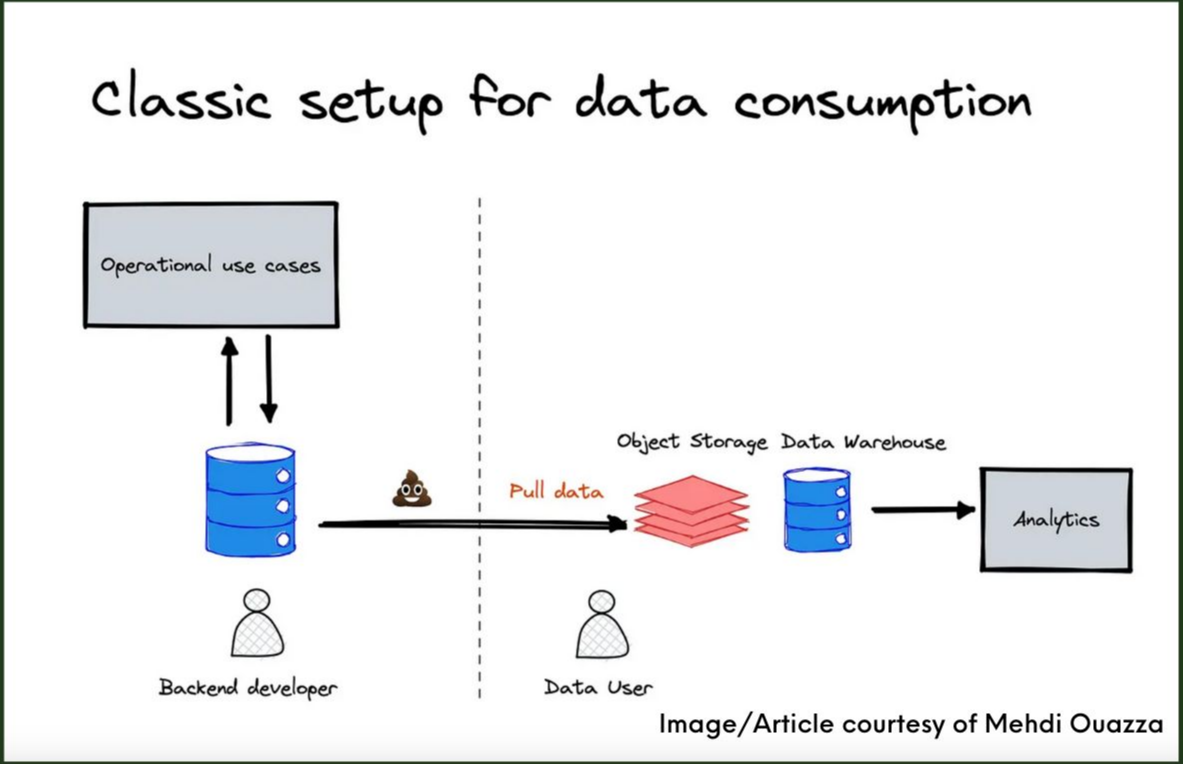
The Pros

- Self-Service
- Pretty easy to learn
- Easy to implement
- Not expensive (at first)
- Allows teams to move quickly

The Cons

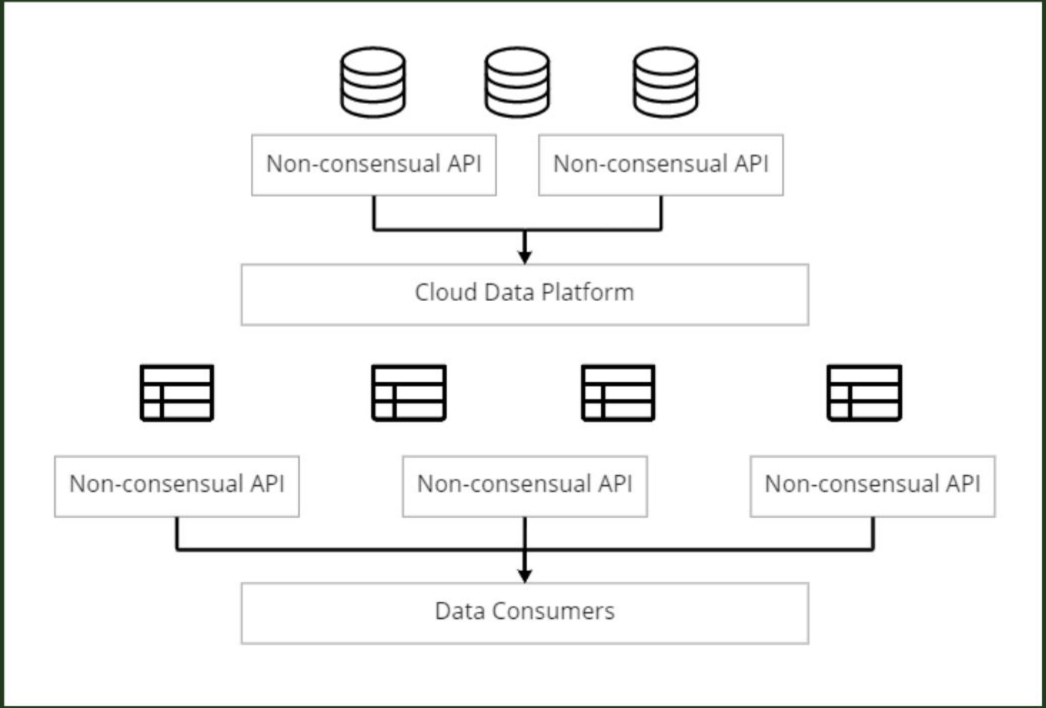
- Lack of ownership
- Garbage In/Garbage Out
- Tech debt
- Missing semantics/context
- Lack of data quality

THE COST OF MOVING FAST...



Image/Article courtesy of Mehdi Ouazza

THE COST OF MOVING FAST...



DATA AS A PRODUCT

Data ownership should look like the implementation of typical software engineering best practices. API generation, CI/CD, and Monitoring are critical to ensuring the stability of data products.

- Data quality is the responsibility of everyone, not only data teams
- Consumers and producers must conversate without DE in the loop
- Common engineering standards exist and should be followed for data
- Data products must leverage production-grade data pipelines

DATA AS A PRODUCT

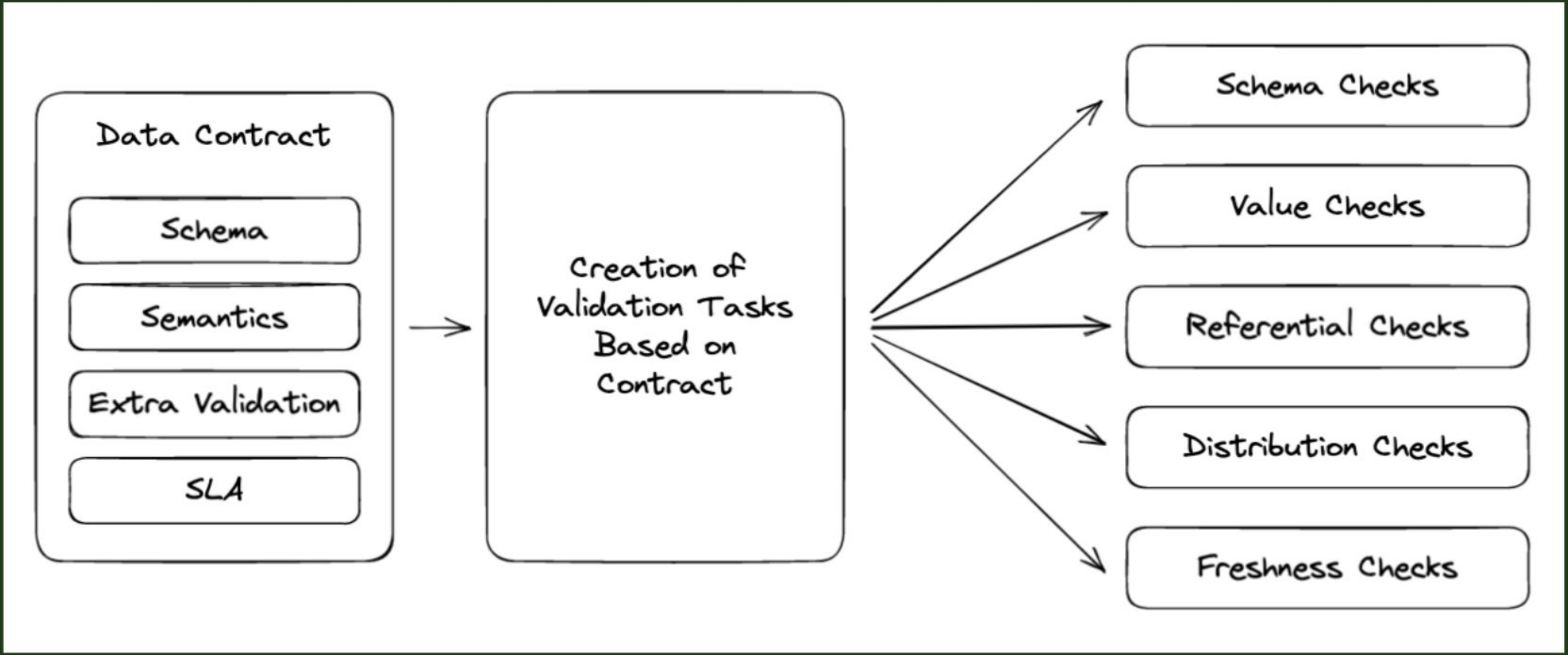
“A verbal contract isn’t worth the paper it’s been written on.” - Samuel Goldwyn

WHAT ARE DATA CONTRACTS?

Data Contracts are API-based agreements between producers and consumers that capture the schema, semantics, distributions and enforcement policies of the data. **They are the technical vehicle necessary to implement data products.**

- Provide a single surface for collaboration on data in a shared language
- Allow the data model to evolve in an Agile, iterative way
- Apply data governance incrementally where it’s needed

THE DATA CONTRACT



WHERE TO APPLY CONTRACTS

Data Contracts are API-based agreements between producers and consumers that capture the schema, semantics, distributions and enforcement policies of the data. **They are the technical vehicle necessary to implement data products.**

- Provide a single surface for collaboration on data in a shared language
- Allow the data model to evolve in an Agile, iterative way
- Apply data governance incrementally where it's needed

PREVENTION + DETECTION

+++

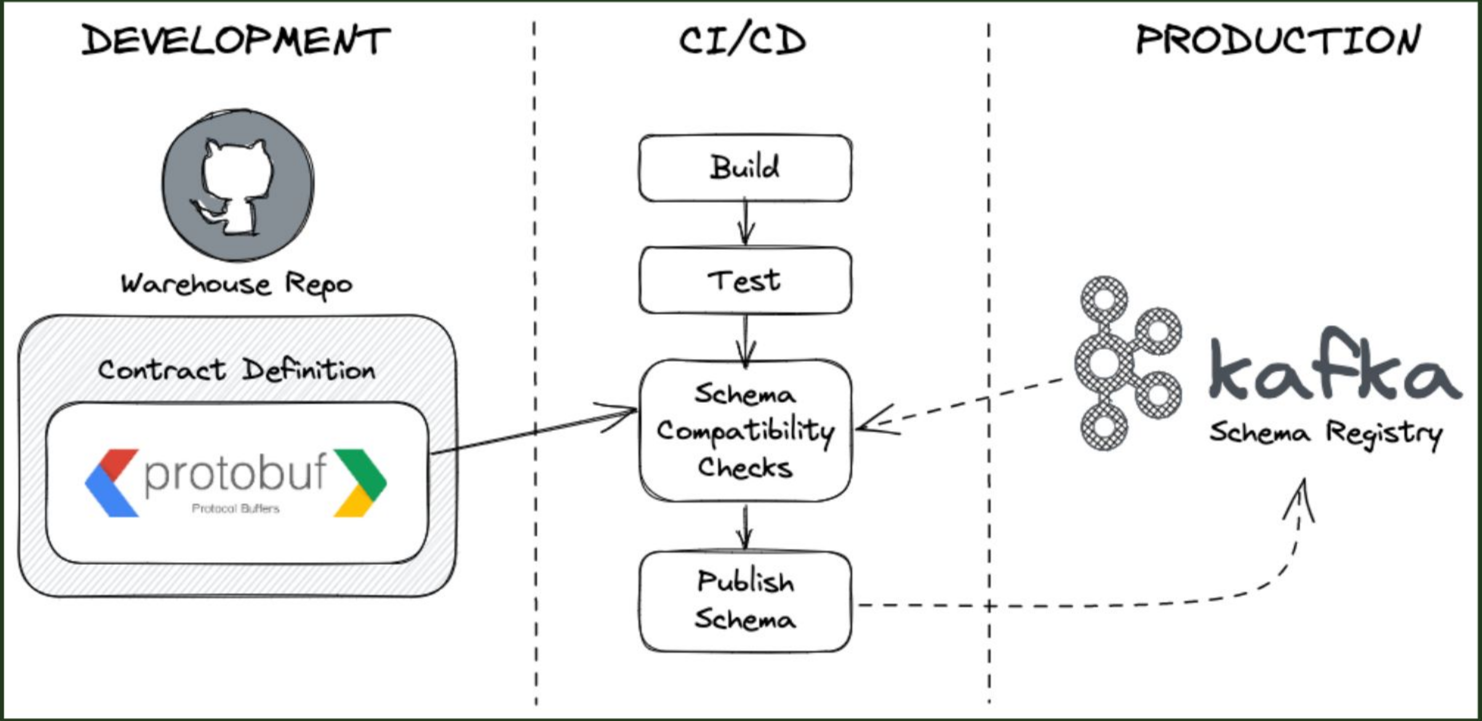
Prevention
Keeping things from breaking

- Inserted in the CI/CD workflow
- Schema & Value enforcement
- Feedloops not walls
- Lineage & Diffs

Detection
Understanding when things break

- Monitoring on data and then taking action
- Push data to a DLQ or holding table
- Alert both producers and consumers
- Capture and surface SLAs

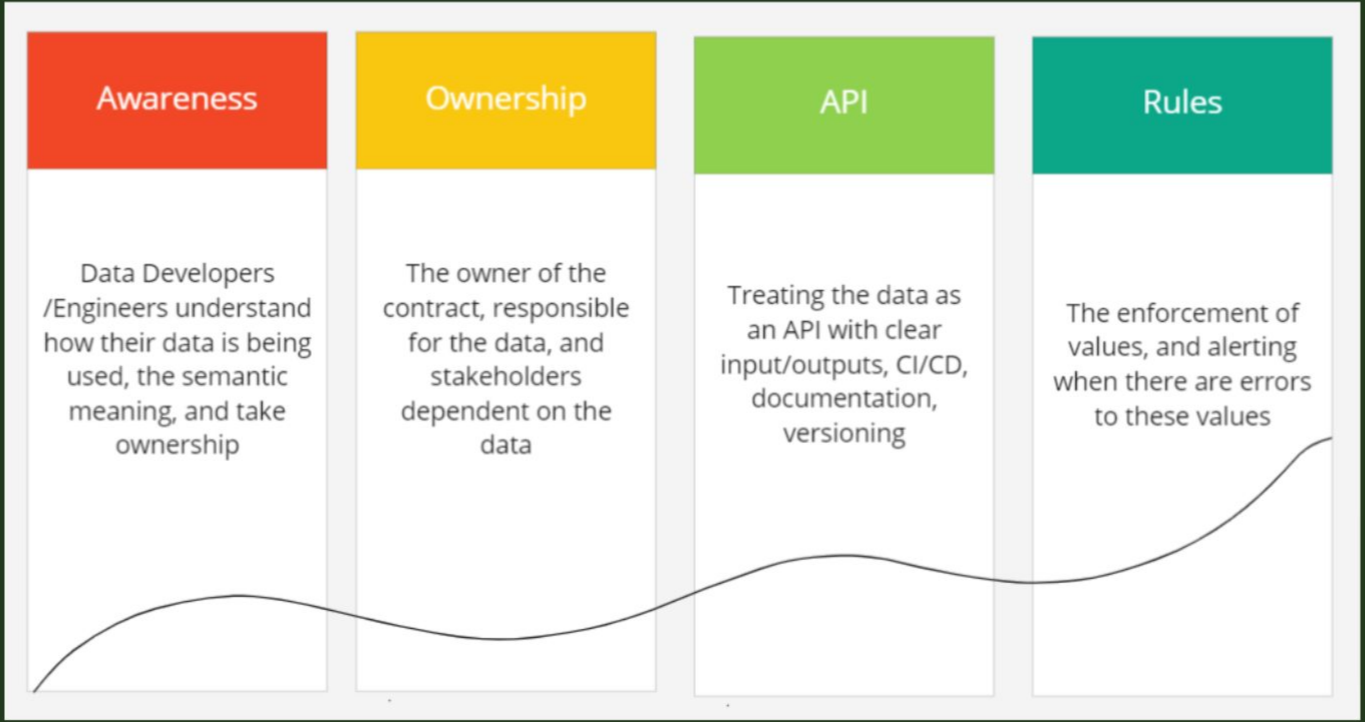
THE HOW


DEVELOPMENT
CI/CD
PRODUCTION
Warehouse Repo
Contract Definition
protobuf
Protocol Buffers
Build
Test
Schema Compatibility Checks
Publish Schema
kafka
Schema Registry

IMPLEMENTATION

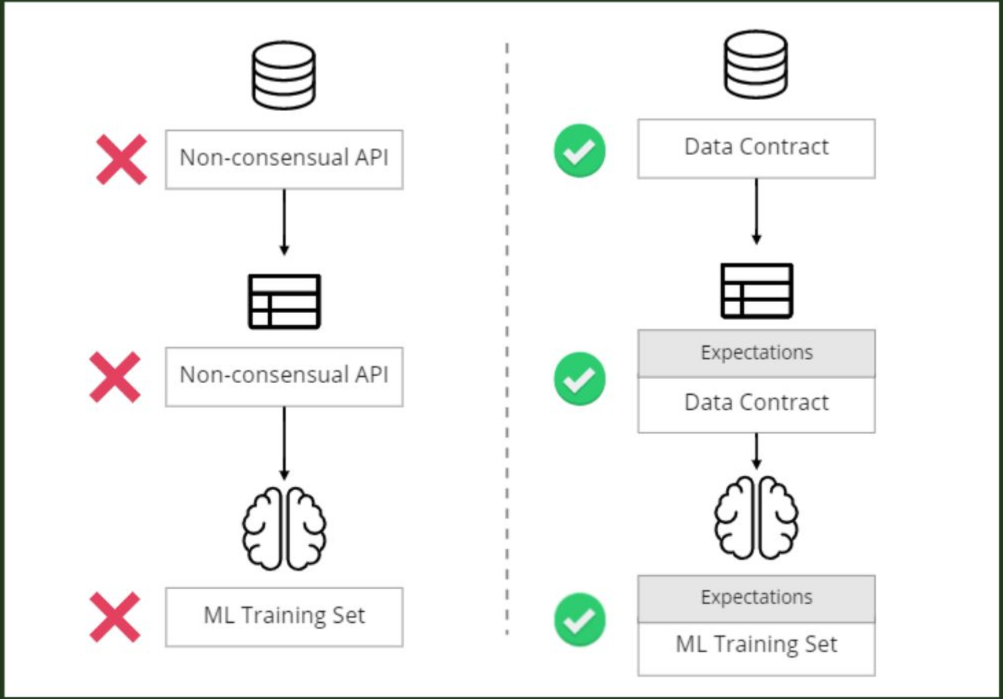
- Contract stored in a **schema registry**
- Checks run during **CI/CD**
- Lineage to provide context for producers
- For stream: Semantic monitors defined in YAML, checks run in stream processing layer (Flink, kSQL)
- For batch: Monitors defined in SaaS/OSS tools, can be rolled back
- **Catalog** to provide context for consumers

THE MATURITY CURVE OF CONTRACTS



Awareness	Ownership	API	Rules
Data Developers /Engineers understand how their data is being used, the semantic meaning, and take ownership	The owner of the contract, responsible for the data, and stakeholders dependent on the data	Treating the data as an API with clear input/outputs, CI/CD, documentation, versioning	The enforcement of values, and alerting when there are errors to these values

FROM OLD TO NEW...



INCENTIVES FOR ENGINEERING +++

Simplicity
Existing Tools

Awareness
Tactile Feedback

Value
High ROI

Data Contracts shouldn’t be applied for all data. Just the data products that matter.

WHERE (AND HOW) TO START

Find the most critical use cases for data in the company and start there. A good rule of thumb: If incremental data quality does not result in incremental business value, it probably isn't a good place to begin!

- Find an ROI-generating use case
- Understand the impact poor data has on revenue
- Implement contracts throughout the pipeline
- Demonstrate the increase in robustness as it translate to business value
- Rollout elsewhere

Data Contracts Over Time

Starting

- Awareness
- Data Quality
- $$$$
- Simplicity
- Incrementality

Iterating

- Data Modeling
- Additional Constraints
- Abstractions
- Alerting
- Violation Reporting

Maturing

- Data governance
- Privacy
- Policy enforcement
- Security

IMPACT AT CONVOY

What we did	Implemented 100s of data contracts
What happened	Huge impacts to business bottom line, cut NULL values in pricing models by double digits!
What changed	Data teams and Engineering teams began talking/collaborating regularly about data
Other benefits	Radically better data documentation, clear ownership
What's next	Add more contracts and monitors incrementally

KEY TAKEAWAYS

- Data contracts are Data APIs which cover both schema and semantics
- Contracts are critical to decentralized data product development
- Invest in contracts where incremental quality results in directly measurable business value

THANK YOU!

DATA QUALITY CAMP

ESTD 2022

JOIN OUR SLACK

dataquality.camp/Slack