REACTOR

# Let's See that Replay:

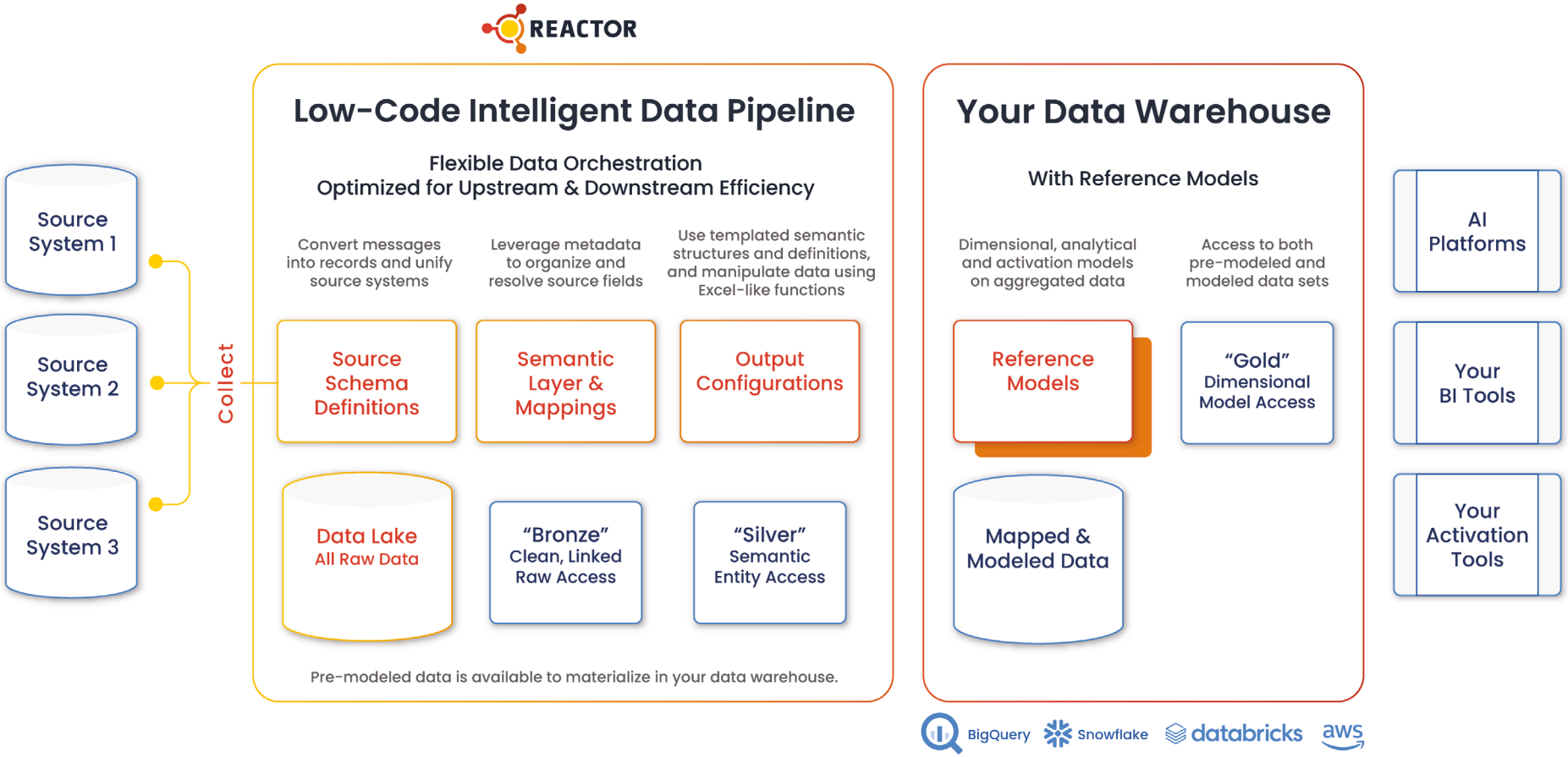
## Remap, Remodel and Reprocess Data for Future-Proof Use Case Flexibility

Jared Stiff, CTO, Co-Founder
Rachel Workman, VP Value Engineering
Eric Best, CEO, Co-Founder

## Introduction: The cloud changes everything

Cloud data warehouses can transform the way you run your business, revealing the drivers and detractors of profitable growth. But cloud data warehouses can also become expensive dumping grounds for unusable data.

A useful and cost effective data infrastructure requires more than just a data warehouse filled with raw data, dependent upon brute-force data engineering to map and model data into useful business output.



In one of the most underrated and better action movies of the last decade, **Edge of Tomorrow,** Bill Cage (played by Tom Cruise, of course) is forced to replay a segment of his life over and over and over. Though this replay isn't voluntary, he quickly realizes he can do things differently on each replay, using trial and error to meet his objectives, ultimately saving the human race from being conquered by an alien civilization.

**Imagine if we could do that for data enablement.**

Though Reactor isn't taking on the preservation of the human race quite yet,
**we are solving for outcome-changing replay in data enablement.**

As we discussed previously, rules applied to a data pipeline are often lossy and destructive (important data are filtered out due to cost or complexity) and brittle in their implementation (rules are hard-coded in ways that make future changes to the pipeline and its outputs complicated, risky and expensive).

## Looking Back to See Ahead

If the world were a static or at least predictable place, this approach wouldn't create problems for data practitioners and consumers.

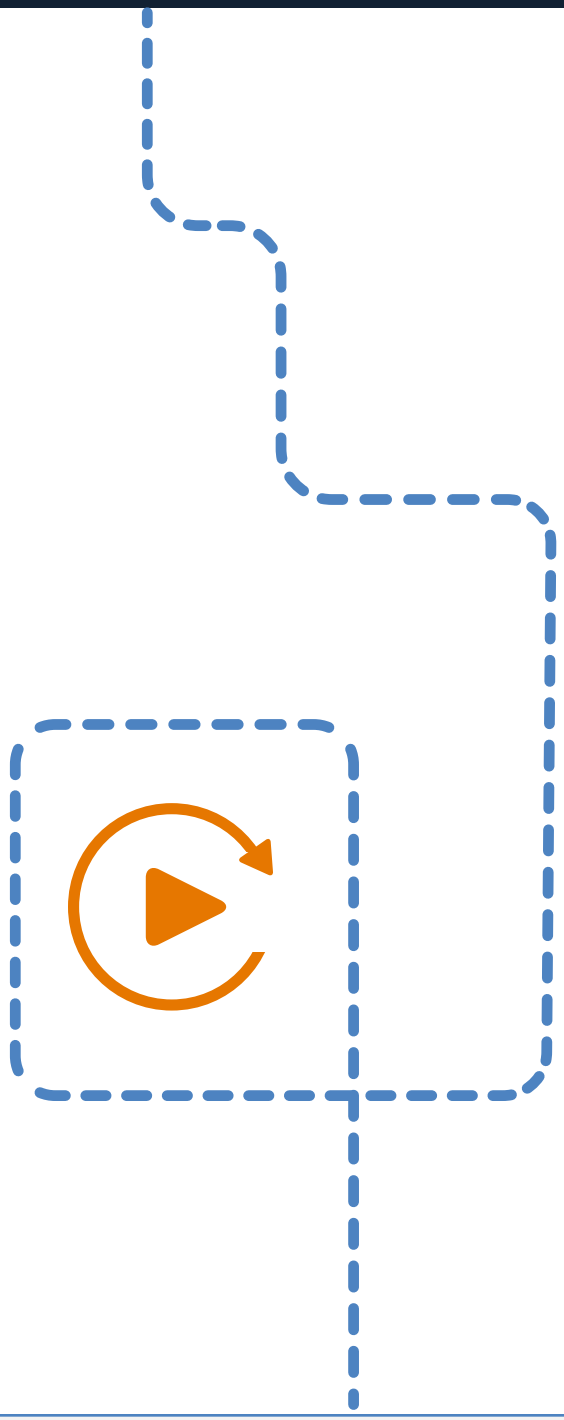
But as we've all seen and know, the world contains constantly changing data systems, schemas, and flows – and decision makers present constantly evolving needs for new outputs, reports, visualizations and analysis. In general, growth in data volume, data set complexity and systems footprints compounds complexity over time.

Since data analysts and engineers can't know in advance what business decision makers will need in terms of use cases and related data models, it makes sense (cloud storage costs notwithstanding) to collect and store as much raw event data as possible – and make it as easy as possible to remodel and reinterpret the data as necessary over time.

This has of course been a key driver in the rise of the modern data lake. But more data without context and especially governance creates new problems. More data can be helpful, but more data exacerbates the complexity problem rather than solving it.

**What if we had a means of reinterpreting the past (especially when new facts or data sets become available, ideally landed in our immutable event log) in ways that enable new use cases and new analysis, without costly data engineering refactoring?**

**What if you could "future-proof" your data flows and transformations — reserving the right to reinterpret history, or "replay the tape" – when stakeholders have new use cases and need new insights?**



## Alternative 1: Early Data Pipelines

ETL-based data flows were the status quo for decades. Take only what you need, move that data to databases and cubes, leave the rest behind. Need something new? Data engineers revisit the entire pipeline from source to output in the form of a major IT project.

Transformations to a normalized model are usually "one-way" - the data has been transformed, the other data you now need (but didn't know back then you would need) is lost due to the transformations. **Some organizations believe they can predict and accommodate the future – and discover later that this is harder than it first appears.**

## Alternative 2: The Current Approach

With the rise of elastic computing and massively scalable storage, the latest generation of pipelines swaps ETL for ELT, loading as much data as possible up front and putting destructive transformations last in the data flow. Per our last blog entry, this is a major advancement toward a future-proof data architecture.

The catch is that modern tooling stores and processes everything through complex logic IN the cloud data warehouse, commonly Snowflake or Google BigQuery, sometimes AWS RedShift or other flavor of cloud warehouse. The result – experienced and written about elsewhere – is that while storage is certainly cheaper in the cloud, ELT promotes spiraling costs in both data engineering and cloud computing. These costs grow exponentially with data volume and complexity.

The goal from here is to preserve the flexibility and other benefits of modern ELT, while simplifying data engineering, governance and processing expense - and to be able to specialize data (and the pipelines that move and model it) without regret.

Why doesn't ELT in the abstract solve this problem? **The dirty secret of ELT-based architectures is that Transformation (with a capital T!) at the end of modern data pipelines is still the bottleneck to buildout, maintenance and scaling.** Just like software development, data transformation projects grow moss. Refactoring transformations are a huge cost burden to data teams. We might call this phenomenon data "modeling-debt" or "transform-debt." We need a way to cost-effectively transform raw data into usable models on an ongoing basis.

ELT also requires reimporting all raw data from source systems every time data consumers present a new requirement – in order to "replay" the data. There is real computing cost and analytical load on source systems to re-run data ingestions for reinterpretation.

## Alternative 3: Our Recommended Approach
## Common Semantic Modeling upon Ingest

**Consider this – What if there were a hybrid model that combines the best of both ETL and ELT principles. ETLT?**

First, we apply a **non-destructive transformation at ingest** to semantically label and ascribe context and meaning to the data on the way in. If we can organize and label data ongoing as we capture it – effectively capturing meaning and context as we go – then all of the data (with its metadata) becomes flexibly usable for any purpose downstream, now and in the future.

Data is transformed – but only for meaning and context – on the way in, with semantic metadata and therefore governable understanding of the data generated as early as possible in the data flow. Common semantic models are rendered in-stream, in-memory as soon as the data is ingested, but these models are only current until new data comes along – refreshed or replaced.

**Transformation for semantic labeling and structure only upon ingest results in governable semantic models.**

This in no way limits the system's ability to apply transformations to outbound data, to accommodate new data models or specific orchestration end points. These outbound (ELT) transformations should be simpler and more manageable as the semantic understanding was already established far upstream at ingestion.

**Outbound transformations inherit and uphold the common semantic layer, while offering flexibility for more complex modeling logic and downstream orchestrations.**

At Reactor, we've been building and experimenting with this hybrid architecture with good results. We keep source data intact in its original and lossless form, stored locally in the cloud rather than captive in source systems.

Streaming transformations run constantly between raw ingest logging and semantically labeled models; and those transformations are expected, and in fact designed to change over time. As the raw data is well-organized and semantically labeled, this enables downstream transformation flexibility to address future use cases.

We are able to accommodate changes to source data schemas, transformations and modeling, and new stakeholder use cases without losing control.

## "Metadata, Not Code"

As we designed our initial architecture for what has essentially become multi-stage or continuous transformation, it became apparent that there are opportunities to optimize for efficiency in the transformation code itself.

In most data pipelines today, transformations are executed through some combination of SQL, Python or UX-driven tools like Zapier. The most popular integration and modeling tools today are squarely aimed at data engineers writing code. Assigning and/or defining semantic metadata labels to data sets at ingest, upstream of the data warehouse, allows us to move much of the transformation logic from custom code to simpler metadata manipulations.

The more accurate and complete the initial semantic labeling, structure and definitions are at ingest, the more responsive downstream transformations can be. A library of semantic handler services can address this need. These services can be applied via fixed schema database or custom code. Fully productionizing these components moves **join- and stitch-logic upstream within the data pipeline and moves mapping logic from compiled code to composable data models.** The end result is a metadata-driven process that vastly increases our reusability and maintainability, with the eventual goal being governed, end-user control (and community sharing of common definitions).

With this approach, schema descriptions and transformations can be updated & modified within our application with no software code recompilation and deployment necessary. This metadata-driven approach supports standardized lifecycles for SCM and release management. And it allows reuse of transformation logic across and between egress points to allow customizations and extensions without breaking the ability to make common upgrades through a multi-tenant product versioning lifecycle. Proprietary extensions to our standard models are easily added to support new BI systems, data warehouses and operational systems.

## Instant Replay Architectures

One of the most important features of this reference architecture is a concept we've dubbed "replay." Once you've properly organized and labeled raw data at ingest (immutably stored locally), we can change the definitions captured in semantic labels and "replay," or reprocess the data to accommodate new downstream models, orchestration schemas and stakeholder use cases and related outputs. The architecture enables the ability to efficiently change the models with little or no code, and few or no breaking changes. This is profoundly different from and superior to prior approaches that require data engineering for any change at any stage.

The result is the ability to truly democratize and future-proof data flows and models. By gathering all of the data up front, you don't need to know now all the ways you might need to use the data in the future, and can instead focus on short-term usability, relevancy and ROI.

## Realized Benefits

How does continuous transformation in Reactor compare to the status-quo approach characterized by SQL or Python modeling in or on a cloud data warehouse?

**Here are a few of the differences and benefits:**

**Specialize and future-proof** your data models and flows without regret, as all raw, lossless data is retained for future use cases and reinterpretation

**Maximize complex data modeling flexibility** without compromising data governance, by separating, defining and maintaining the semantic layer upstream of modeling logic

**Increase your time-to-insights** and lower your cloud compute costs with in-stream processing rather than processing materialized data sets

**Allow all data stakeholders (especially non-engineers) to refine and reprocess** ever-changing and imperfect data models without breaking changes, using continuous transformation and "replay."

**Simplify modeling** and reduce cloud data warehouse processing expense by addressing data prep including semantic labeling during data onboarding

**Limit data reprocessing costs** and latency by applying new or modified transformations to affected data fields only

### Remodel as Needed

With data replay, data practitioners and the business stakeholders that rely on their work have the tools necessary to replay targeted segments of their pipelines at any time and as many times as needed, driving outcomes that meet the fluid and ever-changing needs of the business environments they support.

As Bill Cage discovered, the **flexibility to replay, rewrite and reinterpret history can be a superpower.** The same holds true for stakeholders building modern data pipelines and data models.

# Future Proof your Data Stack with Replay

With data replay, your data practitioners and the business stakeholders that rely on their work have the tools necessary to replay targeted segments of your pipelines at any time and as many times as needed, driving outcomes that meet the fluid and ever-changing needs of the business environments they support.

Find out more about all **nine characteristics of a Future-Proof Cloud Data infrastructure** in our comprehensive exclusive ebook.

Contact Reactor to learn more and get started today!

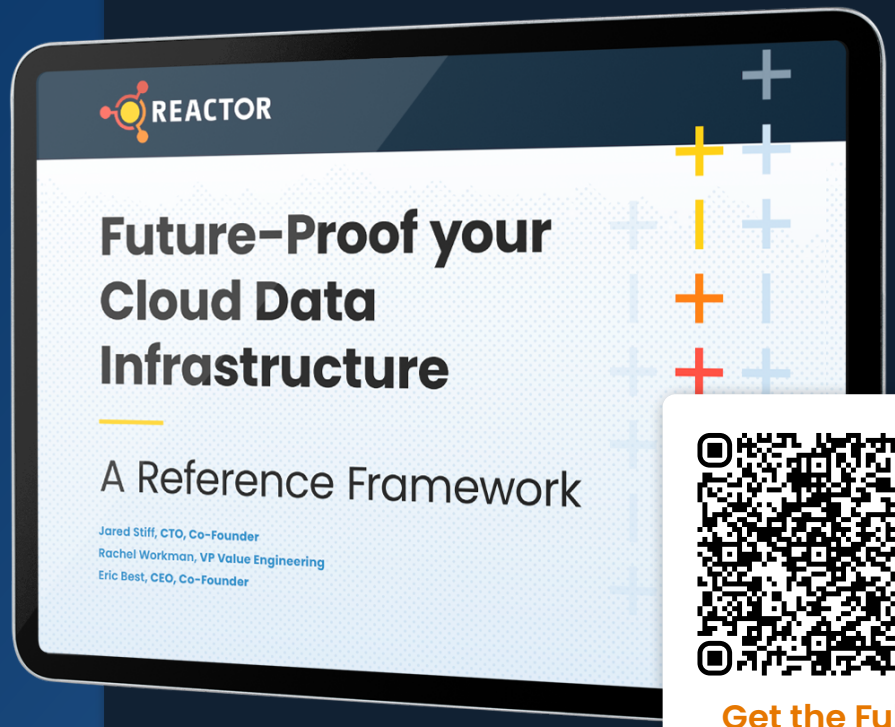
REACTOR

**Reactor** provides the fastest, most efficient path to useful, business-ready data for generative AI, analytics and activation. Built for retailers of any size or complexity, Reactor transforms your unique data infrastructure into an easy-to-use, no-code environment that's accessible to everyone — no engineering degree required. Reactor onboards and ingests data from business critical systems and applications, landing clean, well-defined data modeled directly in your data warehouse.



REACTOR

**Put Your Data to Work.**




**Get the Full E-Book**

www.reactordata.com
1-888-417-6863
Grow@reactordata.com