How Rittman Analytics delivers the semantic layer today with Cube

Olivier Dupuis
CPO at Rittman Analytics

Lewis Baker
COO at Rittman Analytics

Brian Bickell
Head of Partnerships at Cube

Igor Lukanin
Head of Developer Relations at Cube

Code of Conduct

We want to foster an open and welcoming environment where everyone feels that they belong in the Cube community

The full text of our Code of Conduct is available at github.com/cube-js/cube.js/blob/master/CODE_OF_CONDUCT.md

Any instances of inappropriate or unacceptable behavior shall be reported to conduct@cube.dev

Quick notes

- If you have any questions, please type them in the “Q&A” section on Zoom
- We will be using Cube Cloud for demos
- Recording of the event will be available at the events page
- All attendees will receive a post-event survey and we’d appreciate your feedback to help us with future events

What we will discuss today

- How Rittman Analytics approaches the data platform engineering and, specifically, building semantic layers
- What is a semantic layer and how Cube implements it
- Deep-dive demos
- Q&A session

Cube Partner Network

Cube Partner Network

Connect with Cube Partners—Cube experts who'll help you build applications powered by consistent, fast, secure, and accessible data.

rittman analytics

> Partnering with Cube gives us an in-depth view into the future of the semantic layer, thanks to their team we can be confident our deployments will be the best-of-breed for our clients.

Olivier Dupuis

Chief Product Officer at Rittman Analytics

Delivering semantic layers

What is the *purpose* of a semantic layer?

- Derive meaning from data
- Allow formulating domain-specific questions on top of data
- Return meaningful answers

What is a semantic layer?

A tool that superimposes an abstraction on top your data and exposes that abstraction to data consumers.

Components of the abstraction:

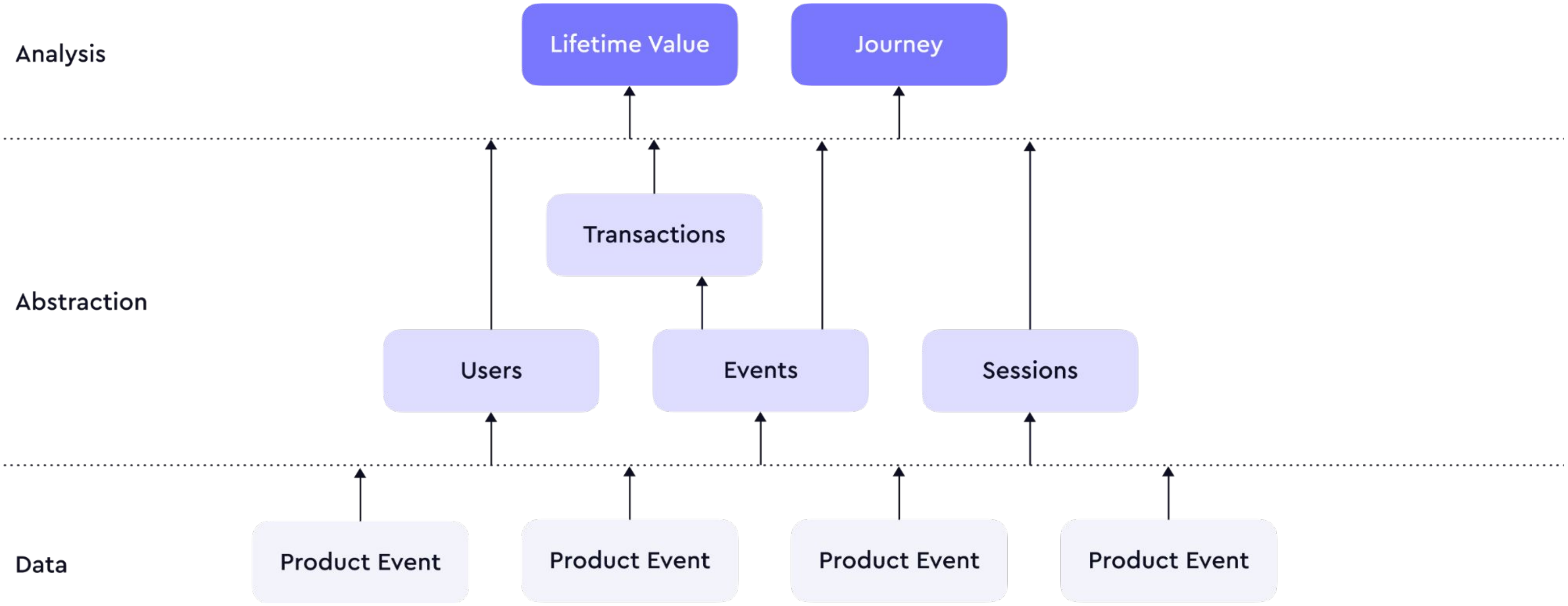
- Domain-specific entities
- Relationships between those entities
- Entity attributes — can loosely be classified as either keys, dimensions, and measures/metrics
- Descriptions of the entities and their attributes

Why build a semantic layer?

You probably already have one!

- As data apps multiply in an organization, it’s important for uniform representation of key entities, relationships, and metrics
- Value is in having consistent results when asking similar questions
- Improved collaboration and communication
- Faster query performance and development
- Simplified data governance

Simple abstraction for product analytics



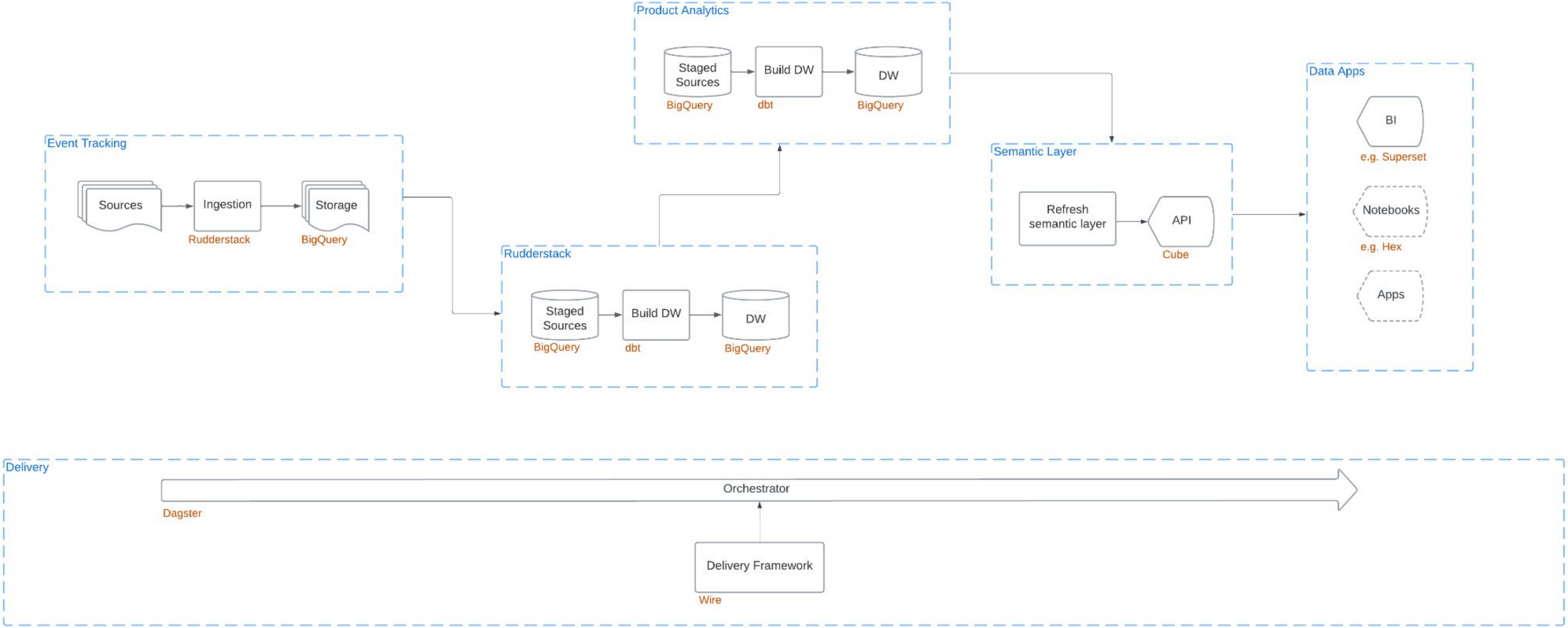
Our approach to delivering semantic layers

- After years of consulting, we’ve developed opinions on engineering data platforms for scalability, flexibility, efficiency, and quality
- Opinions formed around architecture, design principles, processes, and technologies
- Same applies to semantic layers
- We’re still forming our opinion

Rittman Analytics

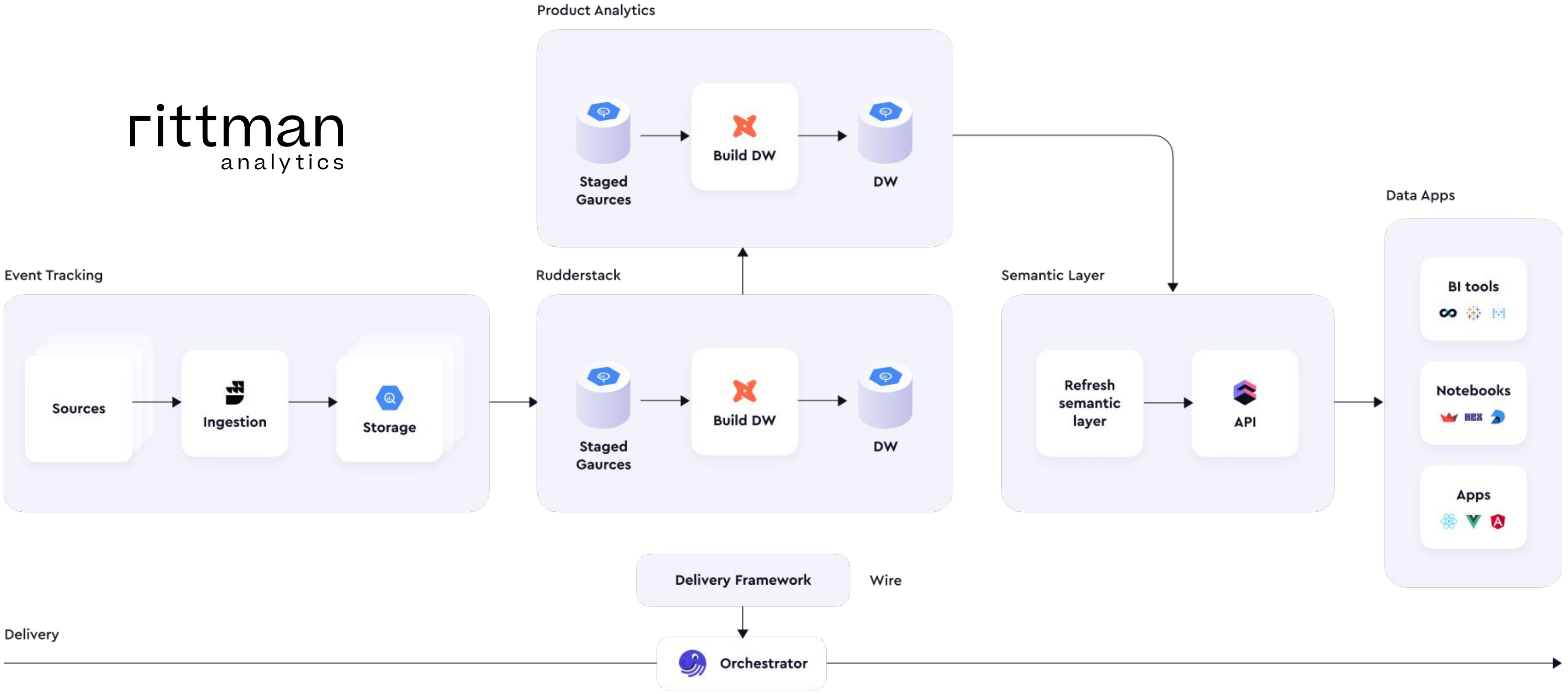
Our approach to building semantic layers

Example: Simple product analytics



Example — Simple product analytics

rittman analytics



Layers of our example

Data products

- Rudderstack as our raw product events data
- Product analytics as a simple abstraction on top of that data

Orchestration

- Dagster to materialize our assets in sequence

Semantic layer

- Cube to expose our simple abstraction, including a single entity (`Events`) and attributes
- Caching layer for performant queries

Data apps

- Cube Playground to test our abstraction and cache
- Superset to interact with the abstraction

Demo 👨‍💻

Delivering a simple abstraction of product data

Backup recording

Cube as the engine for your abstraction

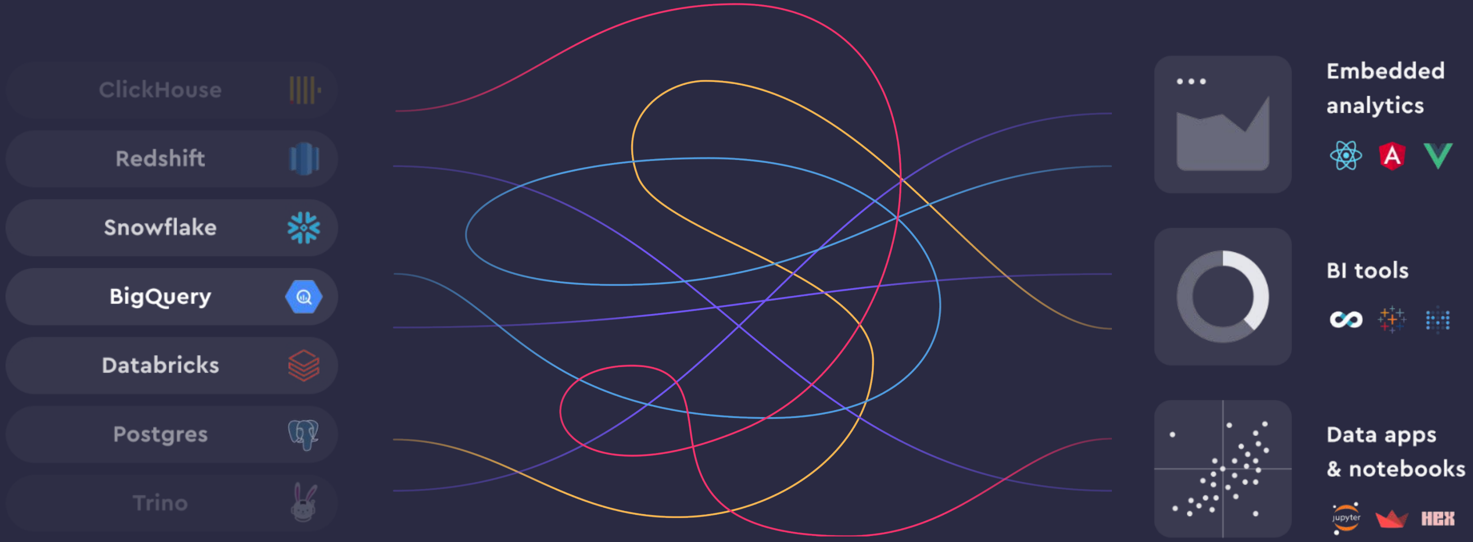
- Translates requests into SQL queries
- Caching and pre-aggregations
- Suite of API endpoints
- Cube Playground
- Access controls

How Cube implements the semantic layer

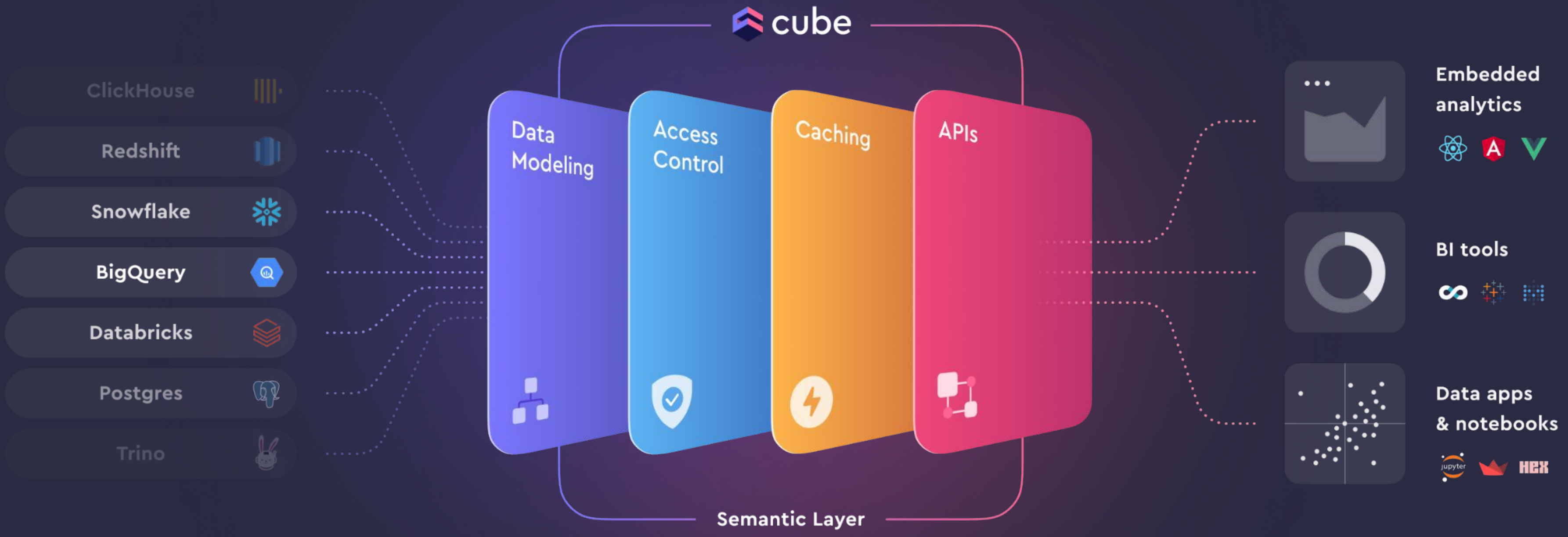
Many-to-Many = Data Chaos

Data engineers are juggling
ALL the sources and outputs

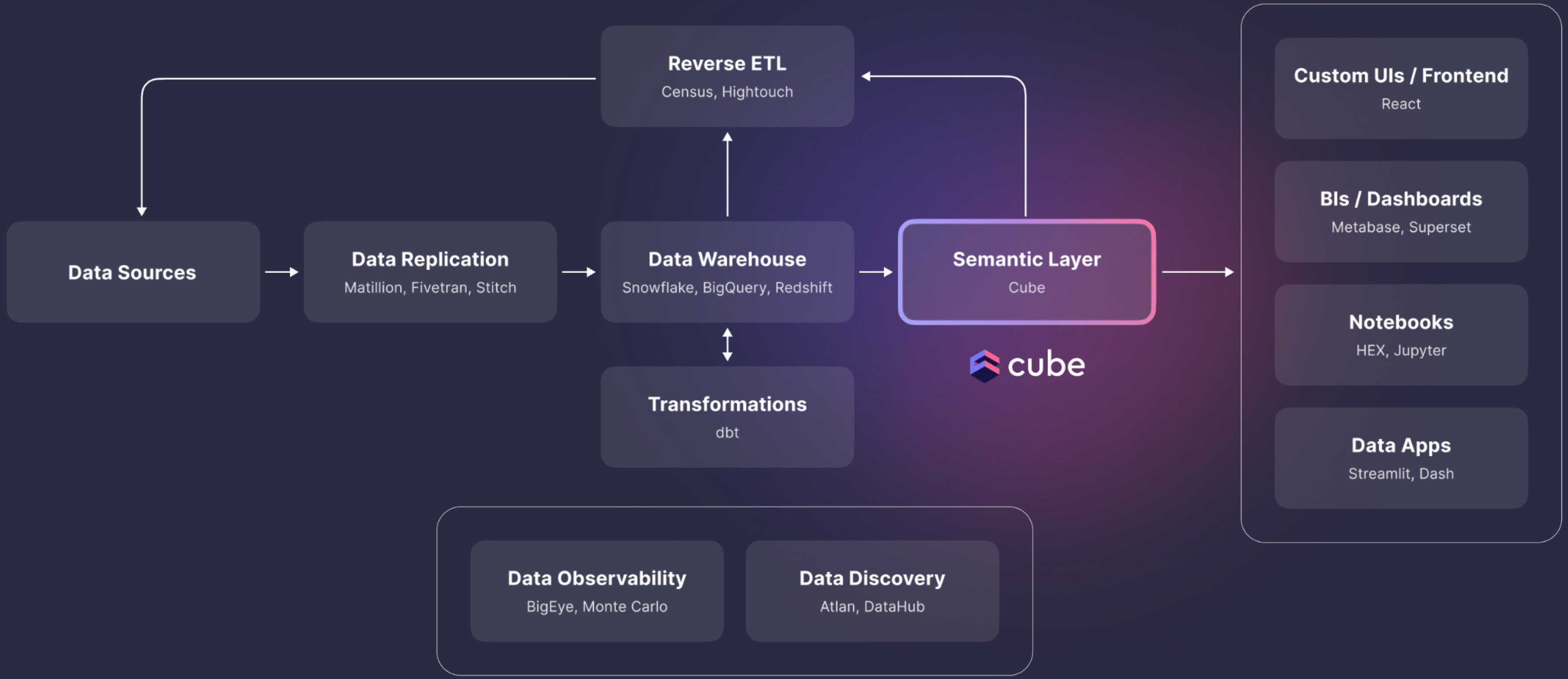
Data consumers are demanding
ALL the data apps



Cube — The Semantic Layer for Data Apps



Modern Data Stack Needs a Semantic Layer



Reverse ETL
Census, Hightouch

Data Sources

Data Replication
Matillion, Fivetran, Stitch

Data Warehouse
Snowflake, BigQuery, Redshift

Semantic Layer
Cube

Transformations
dbt

Custom UIs / Frontend
React

BIs / Dashboards
Metabase, Superset

Notebooks
HEX, Jupyter

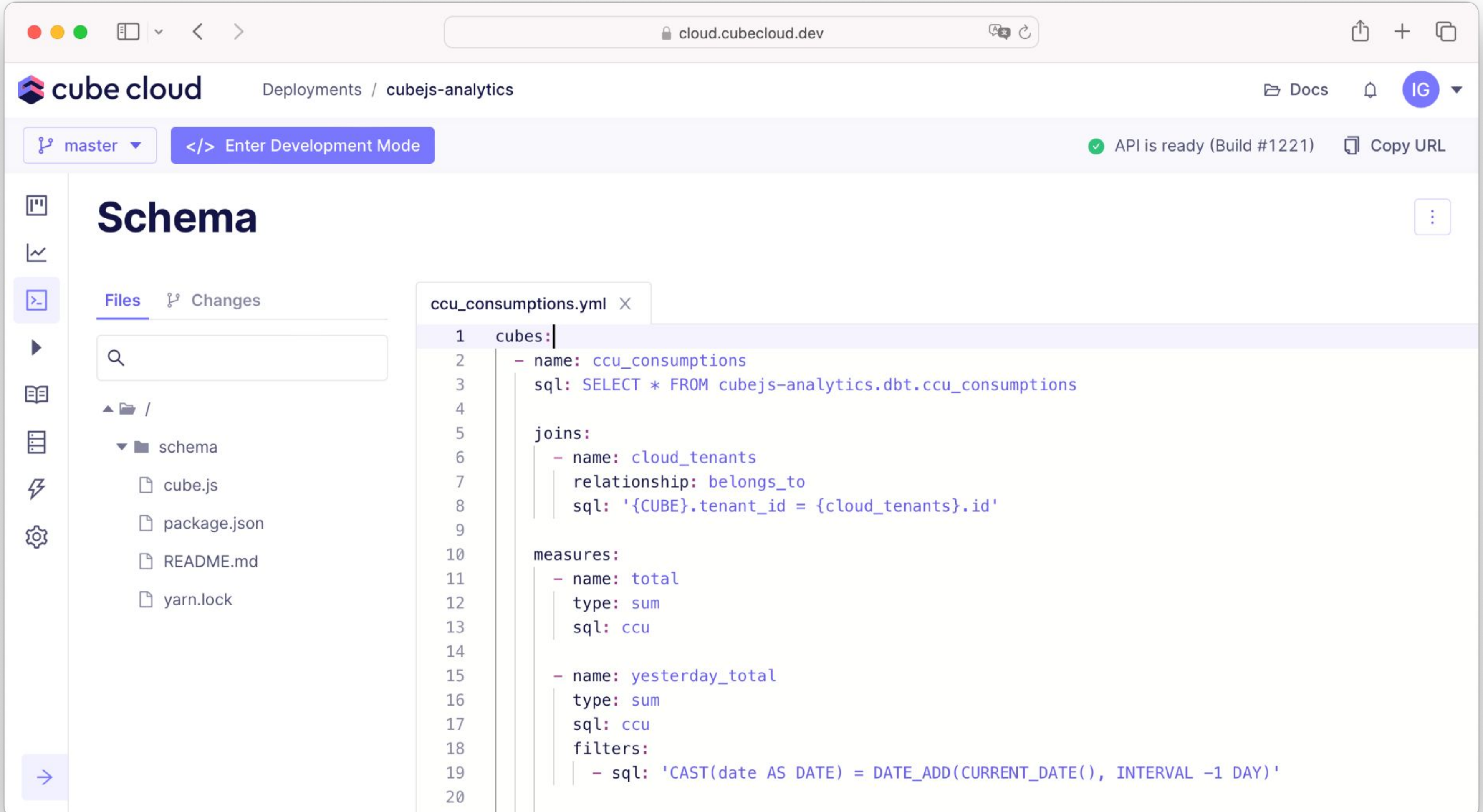
Data Apps
Streamlit, Dash

Data Observability
BigEye, Monte Carlo

Data Discovery
Atlan, DataHub

Data Modeling

- Cube provides means to define the “components of the abstraction”
    - cubes, views, and joins
    - measures and dimensions
- Defined in a declarative, LookML-like models
- Reused by all downstream data apps

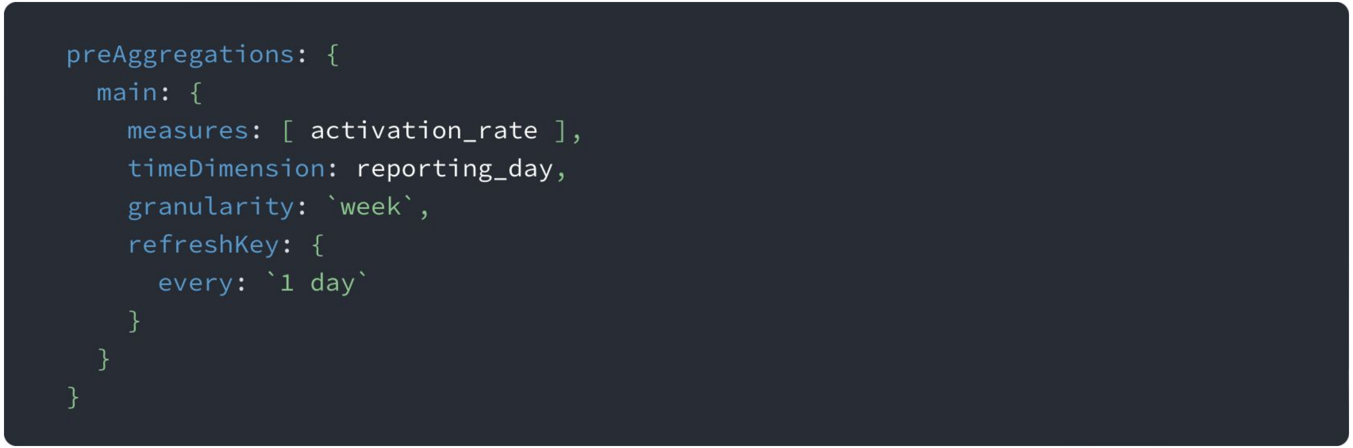


```yaml
cubes:
  - name: ccu_consumptions
    sql: SELECT * FROM cubejs-analytics.dbt.ccu_consumptions

    joins:
      - name: cloud_tenants
        relationship: belongs_to
        sql: '{CUBE}.tenant_id = {cloud_tenants}.id'

    measures:
      - name: total
        type: sum
        sql: ccu

      - name: yesterday_total
        type: sum
        sql: ccu
        filters:
          - sql: 'CAST(date AS DATE) = DATE_ADD(CURRENT_DATE(), INTERVAL -1 DAY)'

```

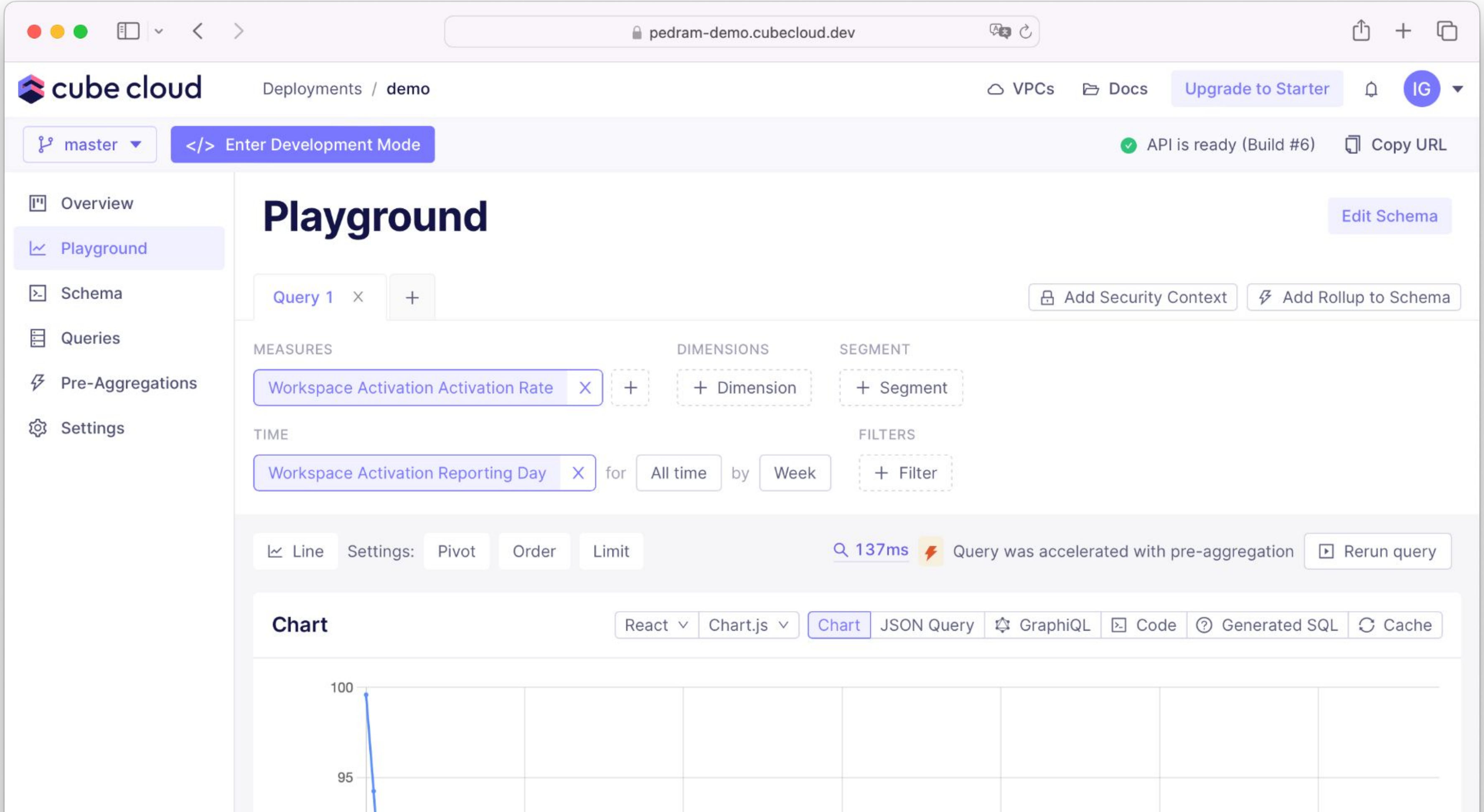
Caching

- Cube provides two-level caching:
  - in-memory cache — deduplicates identical queries
  - pre-aggregations — accelerates queries to sub-second latency
- You have full control over caching and sound default configuration
  - queries always hit the upstream data source — slow, costly
  - some queries are accelerated, other hit the upstream data source
  - all queries are accelerated, queries never go upstream

Caching — Simple configuration



```
preAggregations: {
  main: {
    measures: [ activation_rate ],
    timeDimension: reporting_day,
    granularity: `week`,
    refreshKey: {
      every: `1 day`
    }
  }
}
```



pedram-demo.cubecloud.dev

API is ready (Build #6)

Playground

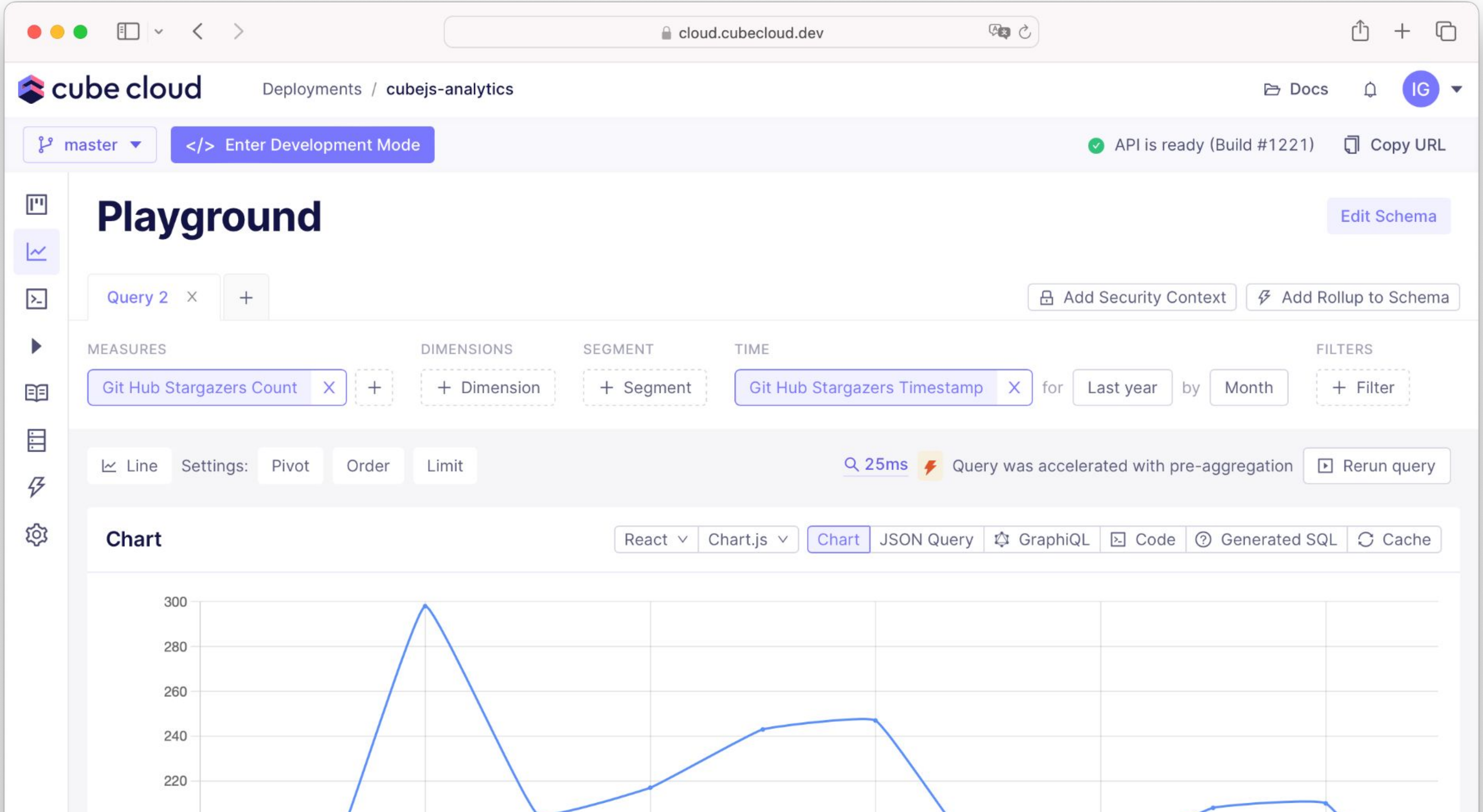
MEASURES: Workspace Activation Activation Rate

TIME: Workspace Activation Reporting Day for All time by Week

137ms Query was accelerated with pre-aggregation

APIs

- Cube provides a set of APIs to deliver data to downstream applications
  - SQL API — for BI tools, data notebooks, etc.
  - REST API and GraphQL API — for front-end applications
- Regardless of the API flavor, queries yield same results
- There’s also Cube Playground, a UI to compose queries



API is ready (Build #1221)

Playground

MEASURES: Git Hub Stargazers Count

DIMENSIONS

SEGMENT

TIME: Git Hub Stargazers Timestamp for Last year by Month

FILTERS

Settings:

25ms Query was accelerated with pre-aggregation

Chart

APIs — Simple queries

```sql
SELECT reporting_day, activation_rate
FROM WorkspaceActivation;
```

```json
{
  "measures": [
    "WorkspaceActivation.activation_rate"
  ],
  "timeDimensions": [ {
    "dimension": "WorkspaceActivation.reporting_day"
  } ]
}
```

APIs — Less simple queries

```
SELECT
  DATE_TRUNC('WEEK', reporting_day) AS reporting_week,
  activation_rate
FROM WorkspaceActivation
ORDER BY 1 DESC;
```

```
{
  "measures": [
    "WorkspaceActivation.activation_rate"
  ],
  "timeDimensions": [ {
    "dimension": "WorkspaceActivation.reporting_day",
    "granularity": "week"
  } ],
  "order": {
    "WorkspaceActivation.reporting_day": "desc"
  }
}
```

Practical considerations

Modeling challenges

- Cube allows remodeling your data warehouse as you wish
- That flexibility requires additional design choices that will need to be enforced to ensure quality of that layer
- Lewis will cover this, but we choose to do an almost exact replication of our data warehouse schema
- But, I’ve also seen different approaches, such as exposing what would have been Views in older approaches (or XA, extended aggregates)

Caching

- Cube’s pre-aggregations are a very flexible solution. Of course, there are alternatives if that better fits your architectural considerations
- Simplest approach is to not have caching and always directly query the data warehouse. Obviously, there are costs associated to this
- You could rely on your BI’s caching as well. But then acceleration and freshness/consistency would be specific to a single tool, not multi-tool
- A client decided to rely on BigQuery’s BI Engine to cache specific tables

Harvesting metadata

- Semantic layer is a central repository of metadata that describes your entities, relationships, attributes, etc. So there's value in harvesting that metadata from upstream systems
- Multiple sources, e.g., database metadata, dbt manifest file
- Lewis to share experience how Droughty sources metadata from database's `information_schema`
- Client working on consuming metadata from dbt jobs, including metric definitions

Q & A

Modeling semantic definitions

Agenda

- What problems does our approach solve?
- How are we doing it?— Concepts
- How are we doing it? — Tooling
- How quick (and accurately) can we do this?

What problems does our approach solve?

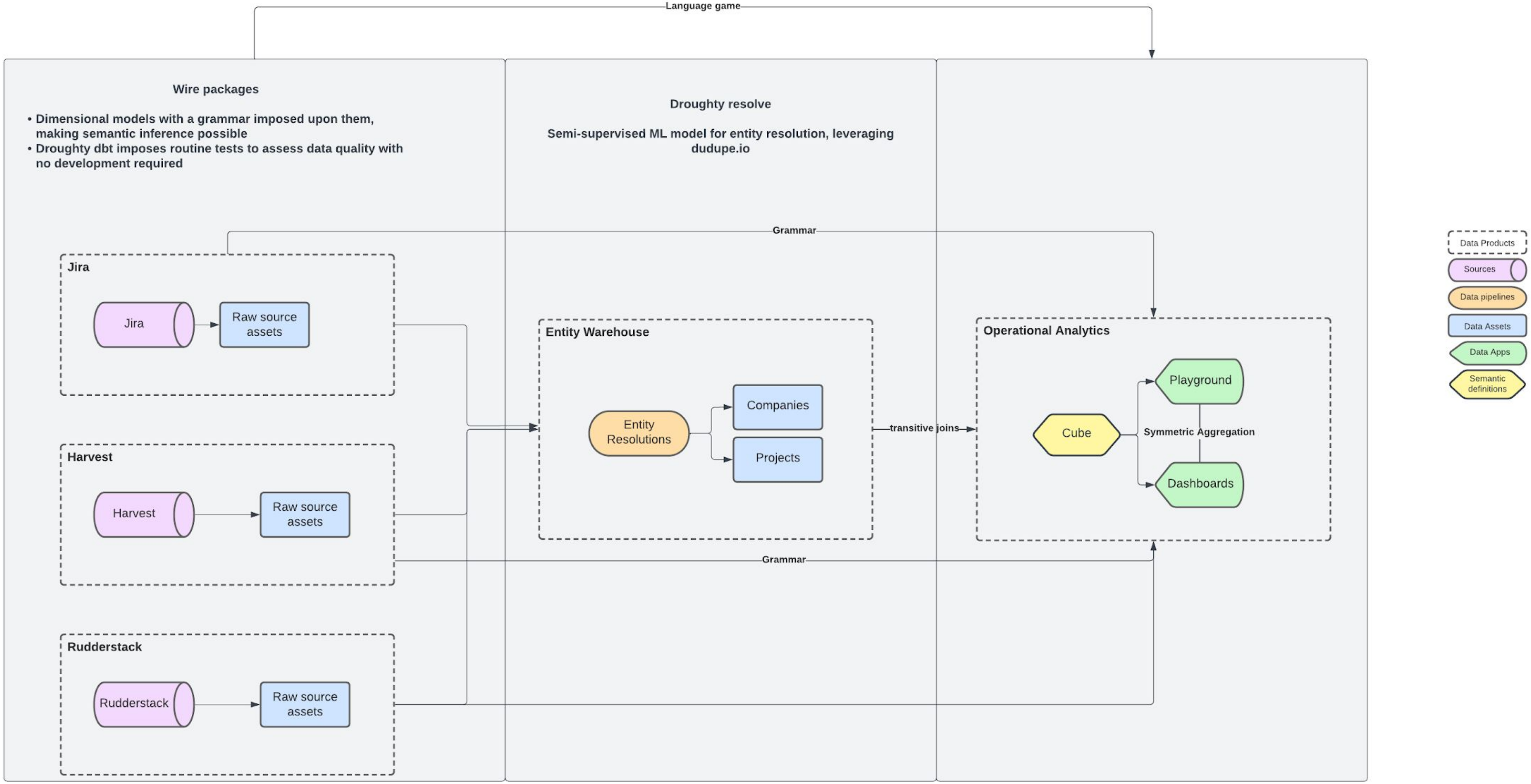
- Moving beyond the unidirectional data model and making them flexible
- Solving the issues of explores ‘running out of road’
- Solving the sometimes poor integration between the decoupled tooling of the modern data stack
- Making the cumbersome, DRY
- Speeding up delivery so that the analysis and activation of data is prioritised over the organisation of it
- Imposes systematic testing and decreases errors through human input
- Embedding meaning into our models, not just ontologies. Semantics are contextual and this offers better comprehension. Not to be confused with metrics!

How are we doing it — Concepts

- Headless semantic layers
- Coupling the semantic layer to the warehouse
  - Using semantic inference to do this automatically
- Leveraging transitive joins for bi-directional queries
- Imposing symmetric aggregates
- Using warehouse metadata to enrich the meanings of entities
- Using semi-supervised ML models to resolve entities

How are we doing it — Tooling

- dbt
- DWH
- Droughty (it’s open source!)
- Cube



Language game

Wire packages

- Dimensional models with a grammar imposed upon them, making semantic inference possible
- Droughty dbt imposes routine tests to assess data quality with no development required

Droughty resolve

Semi-supervised ML model for entity resolution, leveraging dudupe.io

Grammar

Jira

Jira

Raw source assets

Harvest

Harvest

Raw source assets

Rudderstack

Rudderstack

Raw source assets

Entity Warehouse

Entity Resolutions

Companies

Projects

transitive joins

Operational Analytics

Cube

Playground

Symmetric Aggregation

Dashboards

Grammar

Data Products

Sources

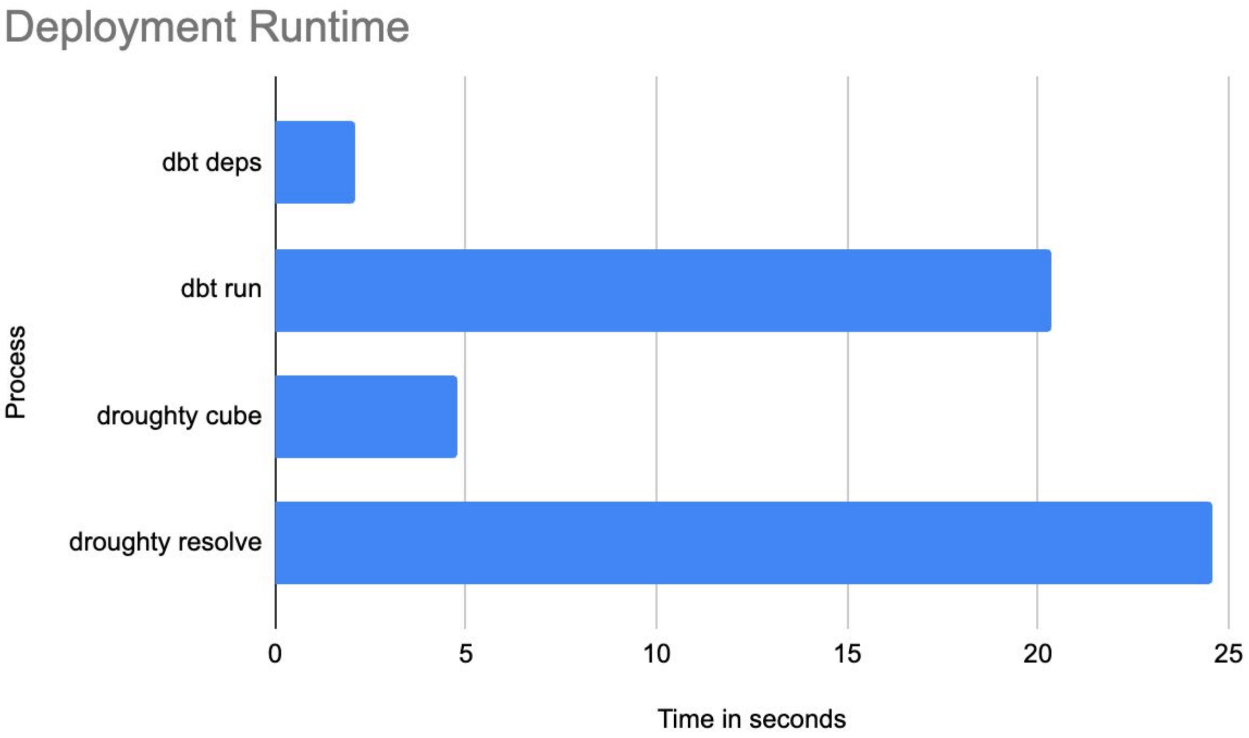
Data pipelines

Data Assets

Data Apps

Semantic definitions

How quick can we do this?



Total: 51.85 seconds

Demo 👨‍💻

Wrapping up / Q&A

- Thanks to Olivier and Lewis from Rittman Analytics 🙌
- Check “What the heck is a semantic layer”
- Check “Building up a semantic layer with dbt Metrics, Cube and Droughty”
- Learn more about Droughty: github.com/lewischarlesbaker/droughty
- Learn more about Cube: cube.dev
- Join slack.cube.dev — the community of 8,000 data practitioners
- **Consider implementing a semantic layer in your organization today**

cube