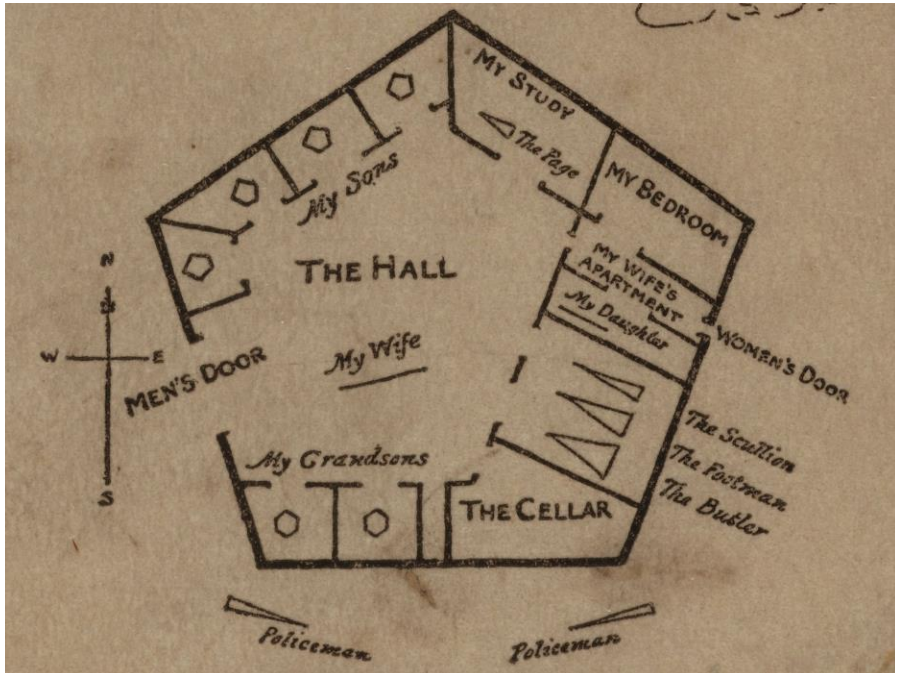
Starburst Galaxy: A Romance of Many Architectures

Benjamin Jeter
Staff Data Architect
Datto EDR

Why this, why now?

A study in reference architectures...

"Oh day and night, but this is wondrous strange"



Edwin Abbot Abbot's *Flatland: A Romance of Many Dimensions*

Agenda:

- Hello (you are here now)!
- Some Design Goals
  - Considerations and Musings
  - Why Trino (on Galaxy)?
- For your Reference
  - What am I (not) Solving?
  - Here *could* be *Diagrams*[1]
- Closing Thoughts

[1]This is where there would be diagrams, but security didn't like them. Sensitive product, attack surface area, that kind of thing.

datto

What do you mean, "reference architecture"?

A brief definition for this talk…

"Distress yourself not if you cannot at first understand the deeper mysteries of Spaceland. By degrees they will dawn upon you."

It is:

- A pattern for making arbitrary data available to end users in a reproducible and modular way.

- It's an opinionated representation of what best practices look like for a given class of use cases.

- A conceptual tool for thinking critically about why we use a particular pattern.

- A pragmatic balance of simplicity and effectiveness.

It's Not:

- A hammer.

- Necessarily the best solution (for your use case). It could be, though!

- A full systems design overview for your data platform.

    - Policy, access control, BI tooling, and the like are out of scope today.

- Going to make you toast.

Some Design Goals

Considerations and musings…

Primary:

- *Facilitate near real-time data access*
- Use only Trino and an orchestrator of your choosing

Secondary:

- *Simplicity*
  - Easy to understand, not simplicity for its own sake
- Modular, Manageable, Flexible, and Adaptable
  - Business needs change, your design should reflect this *a priori*
- Architecture is more than a design diagram
  - Processes are just as important and something less often discussed

Tertiary:

- *Smallest Viable Stack*
  - Also, no Spark
- Trino / Starburst Galaxy
  - I <3 Trino, but this pattern is portable if you want.
- Orchestrator
  - *ORCHESTRATES*
  - Tells your query engine what SQL to execute or calls another service
  - Save the Pandas for later

Facilitate near real-time data access

What does this look like in practice…?

Ingest:

- Data is landed to a "landing zone" in a particular format, let's say JSON for the sake of argument and the file layout is standard Hive
- Use Trino to Query It
- Rest of the Owl :)

Transform:

- Daily batch transform
  - JSON -> Iceberg
  - Serves T-1 data
- External Table
  - Unpartitioned Hive table leverages scan-on-query semantics to query data as it lands
- UNION ALL
  - Iceberg serves T-1 long tail analytics
  - External serves *intraday* data

Access:

- View Abstraction Pattern
  - Users *never* directly query a physical table
  - Leverage INVOKER rights as defined by your security policy
  - Reduces management overhead
    - Transparently re-plumb your pipeline
    - Update your partition / bucket strategy
      - Etc…
- Change to a *whole new pattern* and announce:
  - Data products have been improved by <xyz>, *"no action required"*

datto

Show me some code!

Oh, that’s how you do it…

```
create or replace view <your_catalog>.<production_schema>.<view>
security invoker
as
select
x
, y -- don't @ me about leading commas
, z
from <your_catalog>.<hidden_schema>.iceberg_table where created_date < current_date
union all
select
x
, y
, z
from <your_catalog>.<hidden_schema>.external_table -- this is rebuilt daily and only defined on a single hive partition
```

datto

Thank you!

Benjamin Jeter
Staff Data Architect
Datto EDR