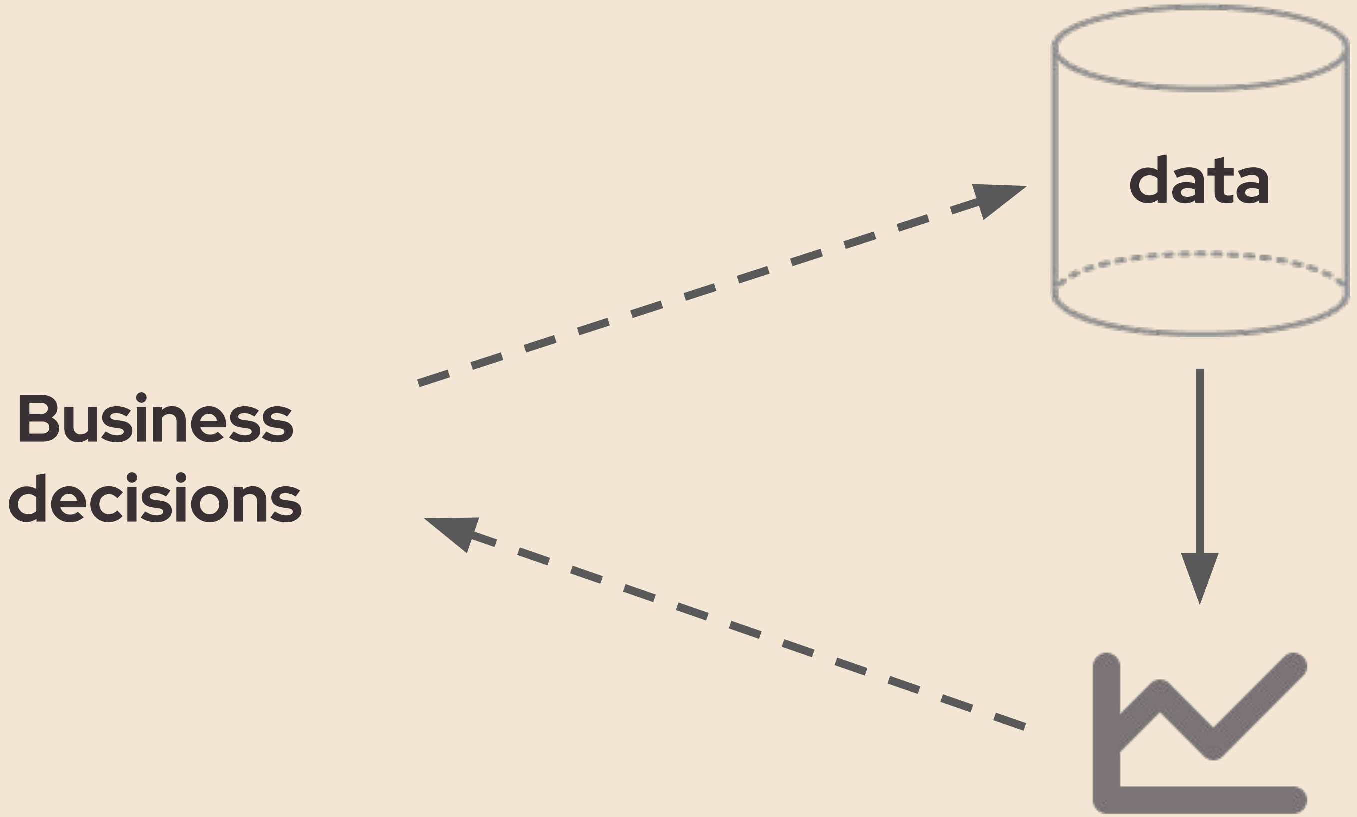
DataOps for Business Intelligence

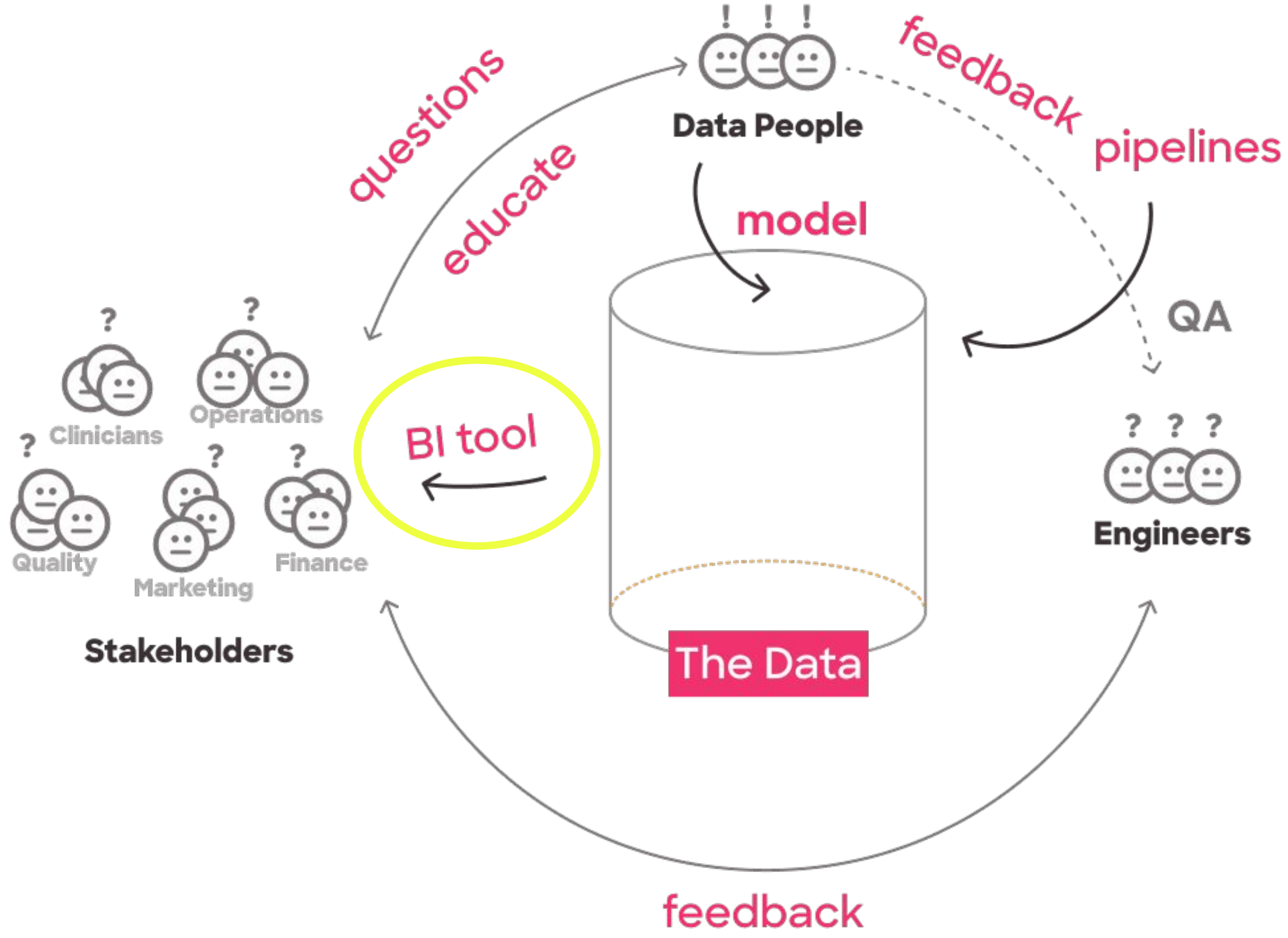
How “Dashboards as Code” can help you develop and validate your analytics

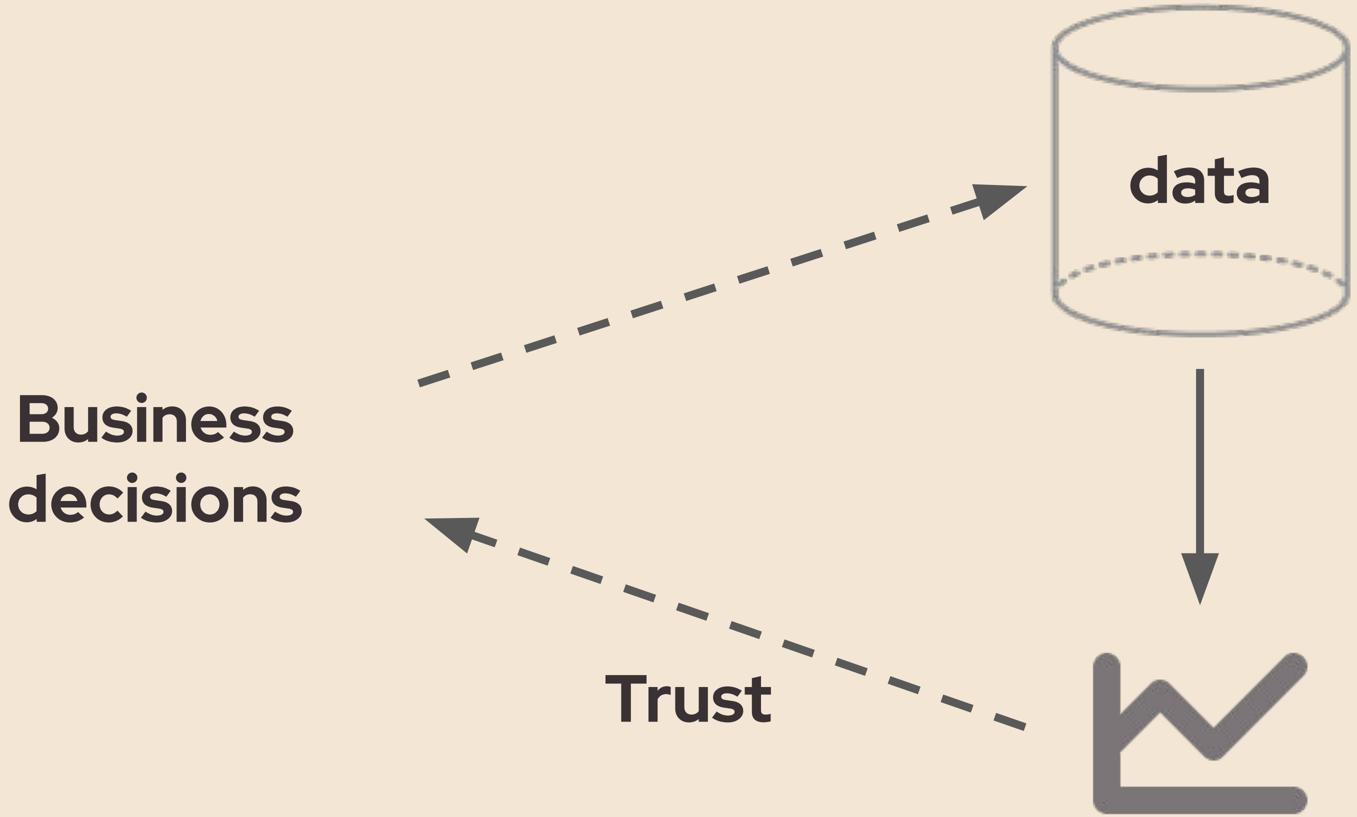
March 29, 2023

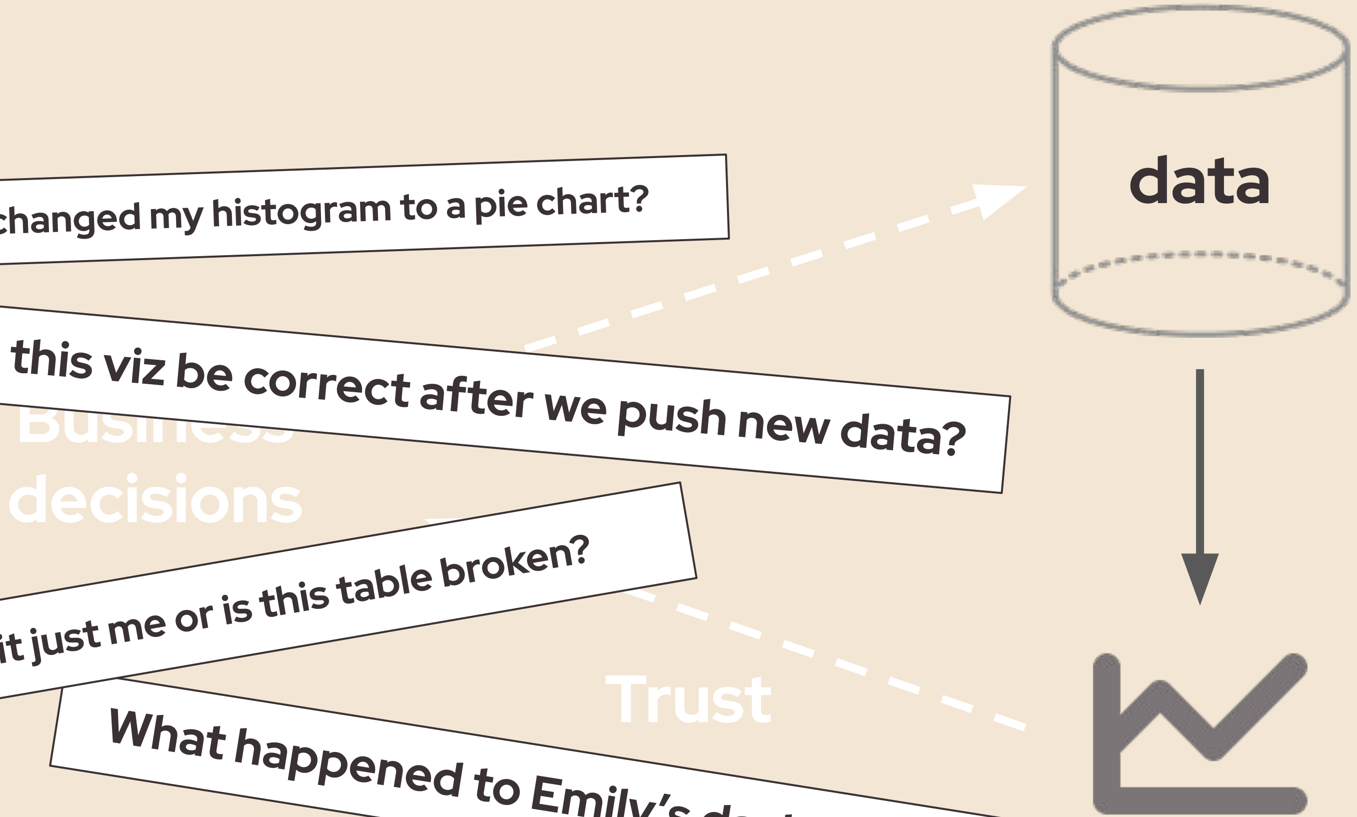
Dan Eisenberg, VP of Technology @ glean

“...what is DataOps?”


data
Business decisions


Data People
feedback
pipelines
questions
educate
model
QA
Clinicians
Operations
Quality
Marketing
Finance
Stakeholders
BI tool
The Data
Engineers
feedback


data
Business decisions
Trust



Who changed my histogram to a pie chart?

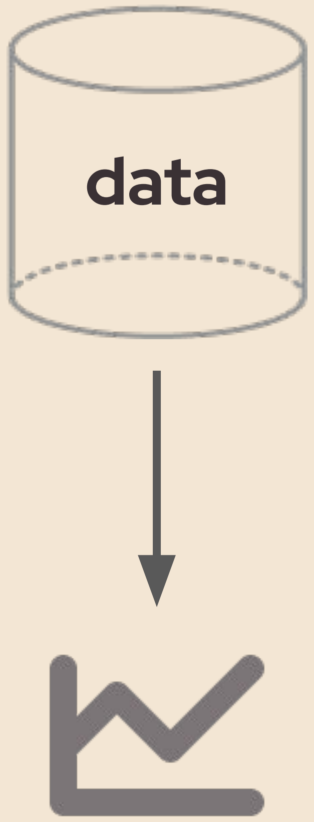
Will this viz be correct after we push new data?

Is it just me or is this table broken?

What happened to Emily's dashboard???

DataOps:

Using `code` to make BI development better



This talk:

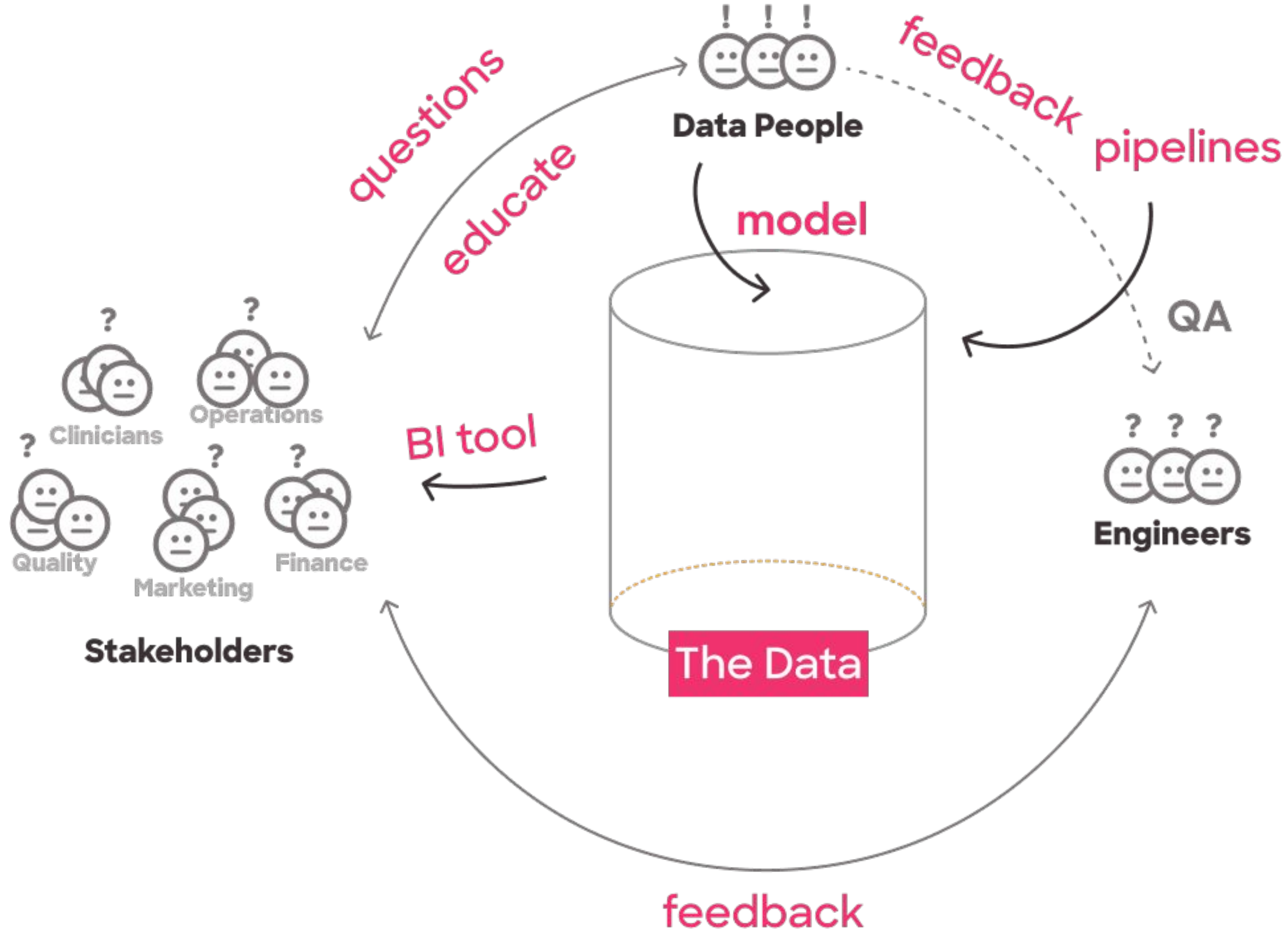
→ What’s hard about BI development?

→ How DataOps can fix it

→ Some practical DataOps strategies and techniques

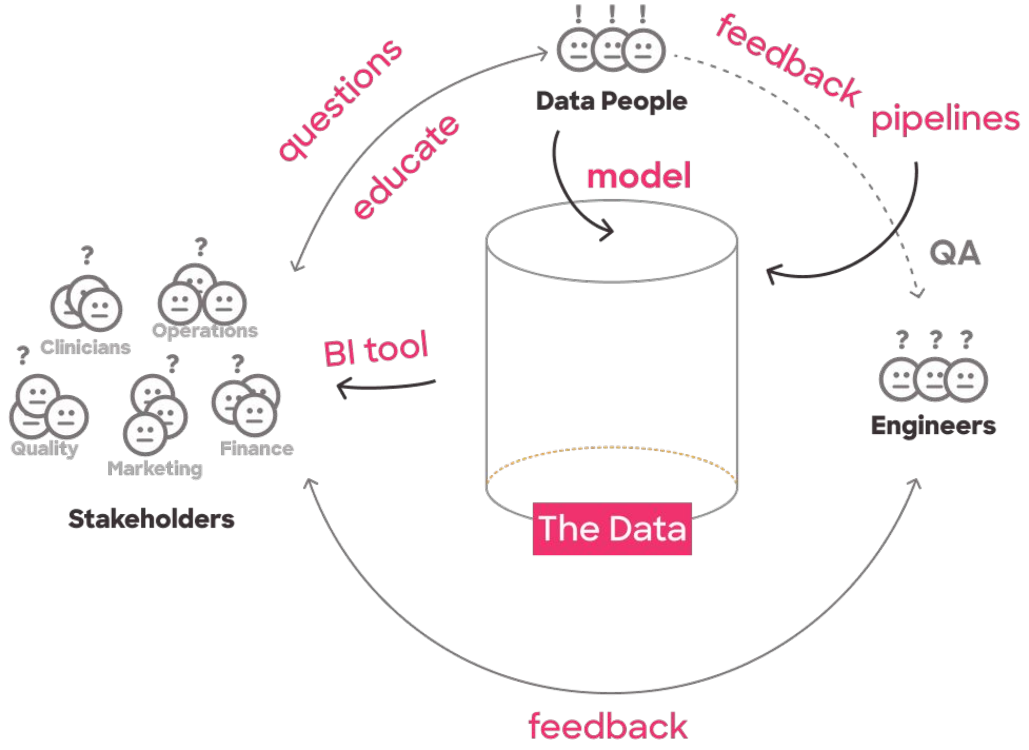
Who I am

Google Analytics $\longrightarrow$ flatiron®


questions
educate
Data People
feedback
pipelines
model
QA
Clinicians
Operations
Quality
Marketing
Finance
Stakeholders
BI tool
Engineers
The Data
feedback

$$\begin{tikzcd} \text{Google Analytics} \arrow[r] & \text{flatiron®} \arrow[r] & \text{glean} \end{tikzcd}$$

What makes this hard?



Challenges today

Hard to version control.

“My dashboard looks different and I don’t know why.”

Limited integration tests.

“This chart was broken for weeks and I didn’t know about it.”

No automated deployments.

“Why aren’t we seeing the latest data here?”

Slooooow development cycle.

“This ticket from last quarter still isn’t done...”

These problems sound familiar…

Let’s jump in the time machine…

It’s 2014 and everybody is moving their infrastructure to the cloud…

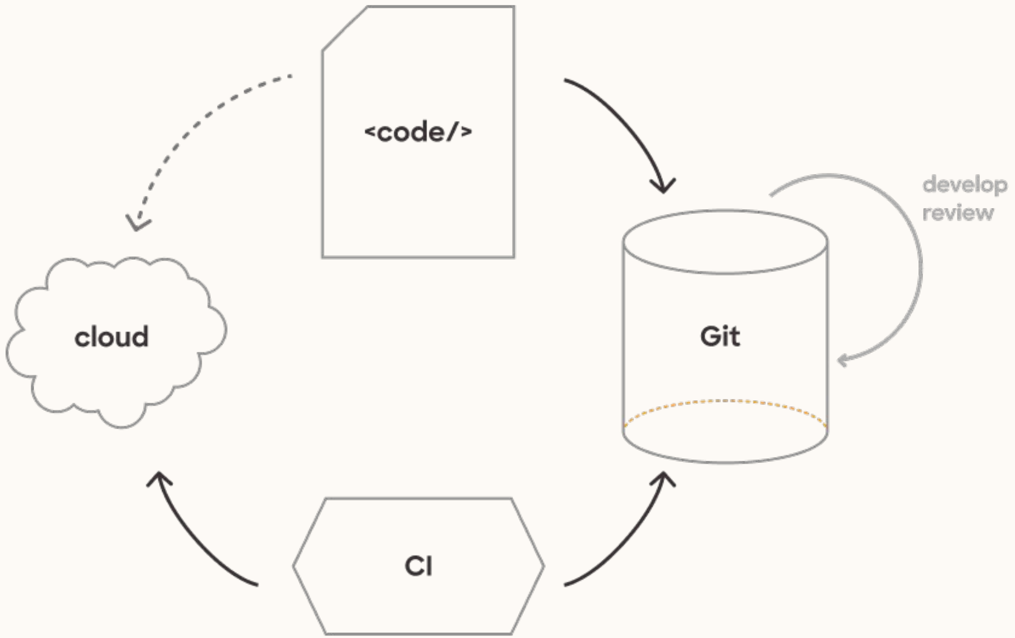
It’s 2014 and everybody is moving their infrastructure to the cloud, **but managing that infrastructure has become a challenge:**

→ No version control
→ No testing
→ No automated deployment
→ Ad hoc collaboration / high communication overhead

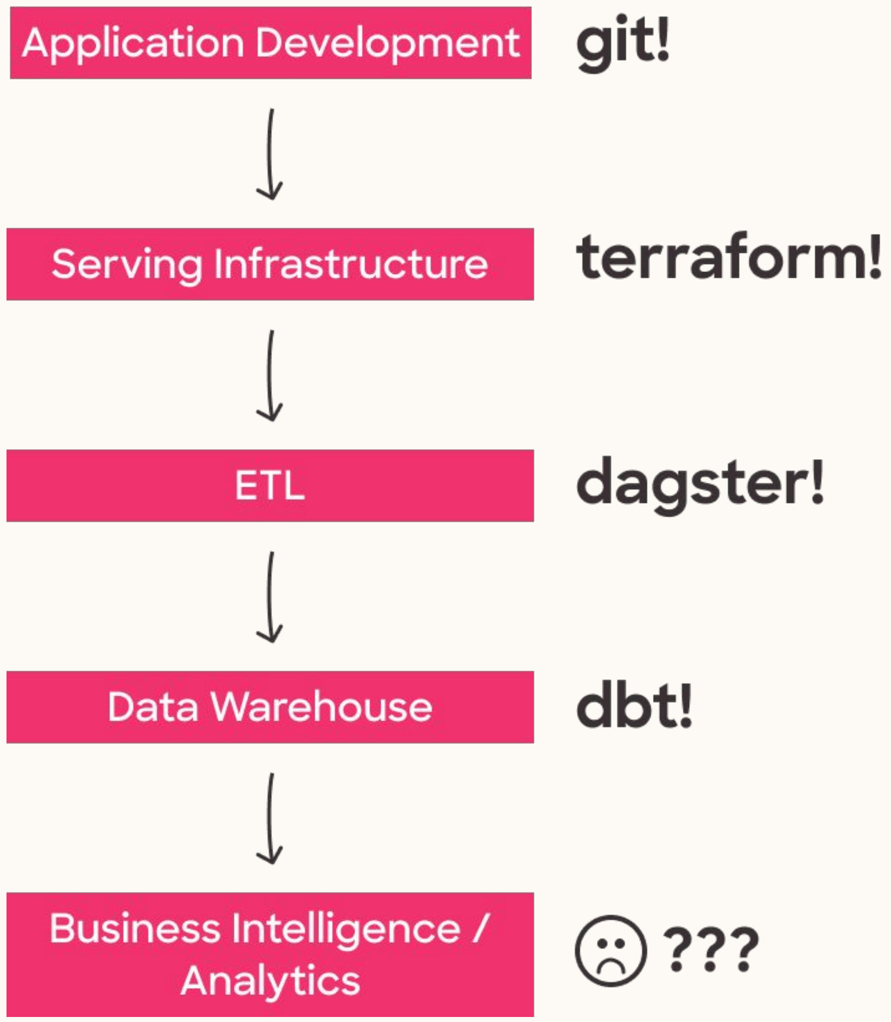
Enter... DevOps

Fundamental innovation:

Treat infra ops just like software development.



“X as code”

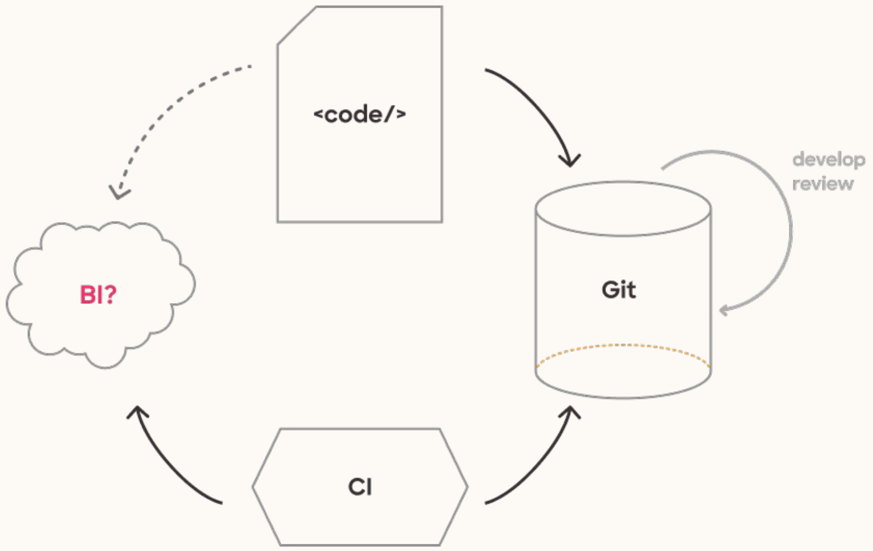

Application Development
git!
Serving Infrastructure
terraform!
ETL
dagster!
Data Warehouse
dbt!
Business Intelligence / Analytics
???

DevOps → DataOps

Ok, code can be good!

But BI is different from infrastructure in some important ways.

How do we make this work?



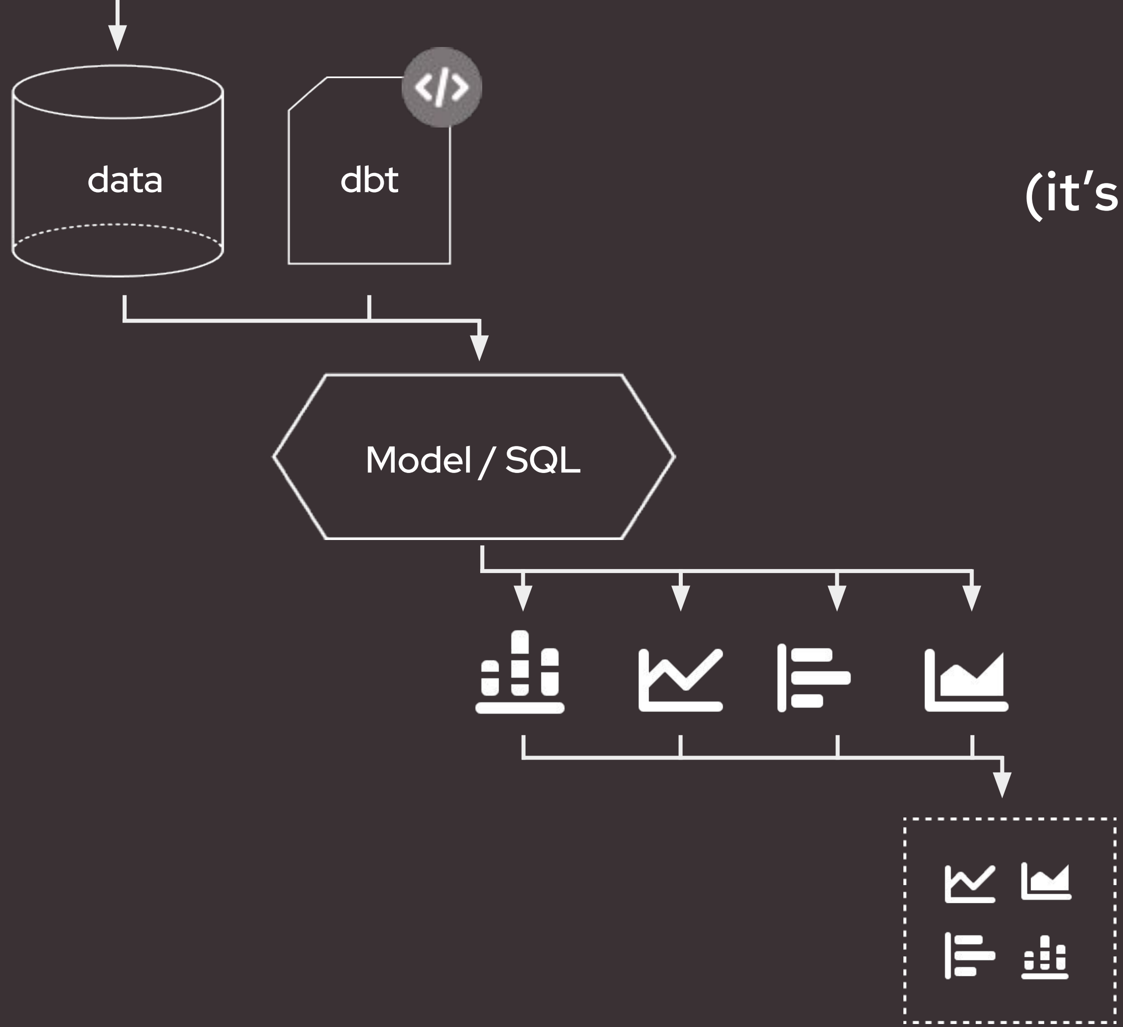
DataOps challenges

1. Wide range of use cases:
   a. Governed metrics & dashboards (prioritize stability)
   b. Adhoc, exploratory analysis (prioritize flexibility)

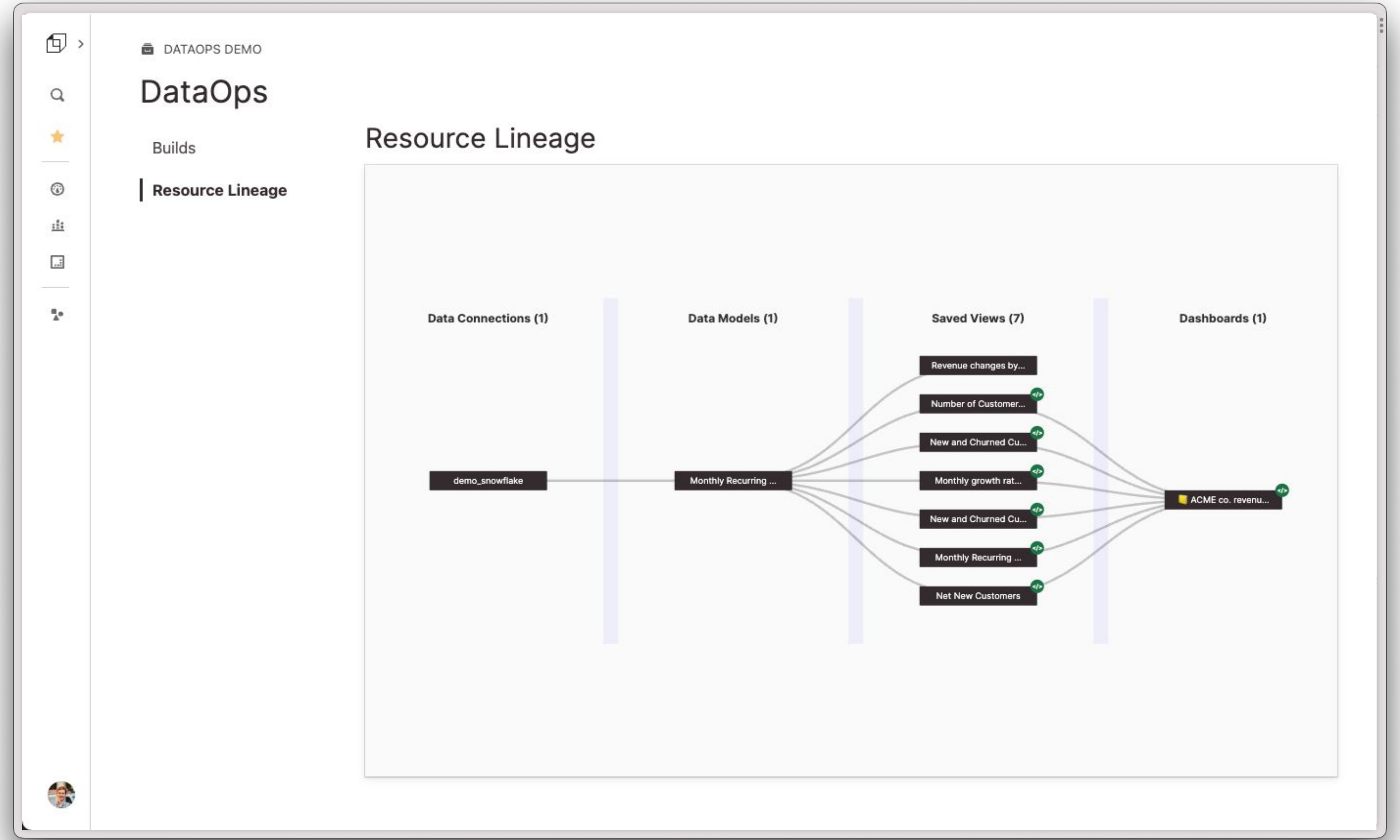
2. BI is inherently visual

3. Interdisciplinary collaboration

4. High number of dependencies



(it's DAGs all the way down)



DATAOPS DEMO

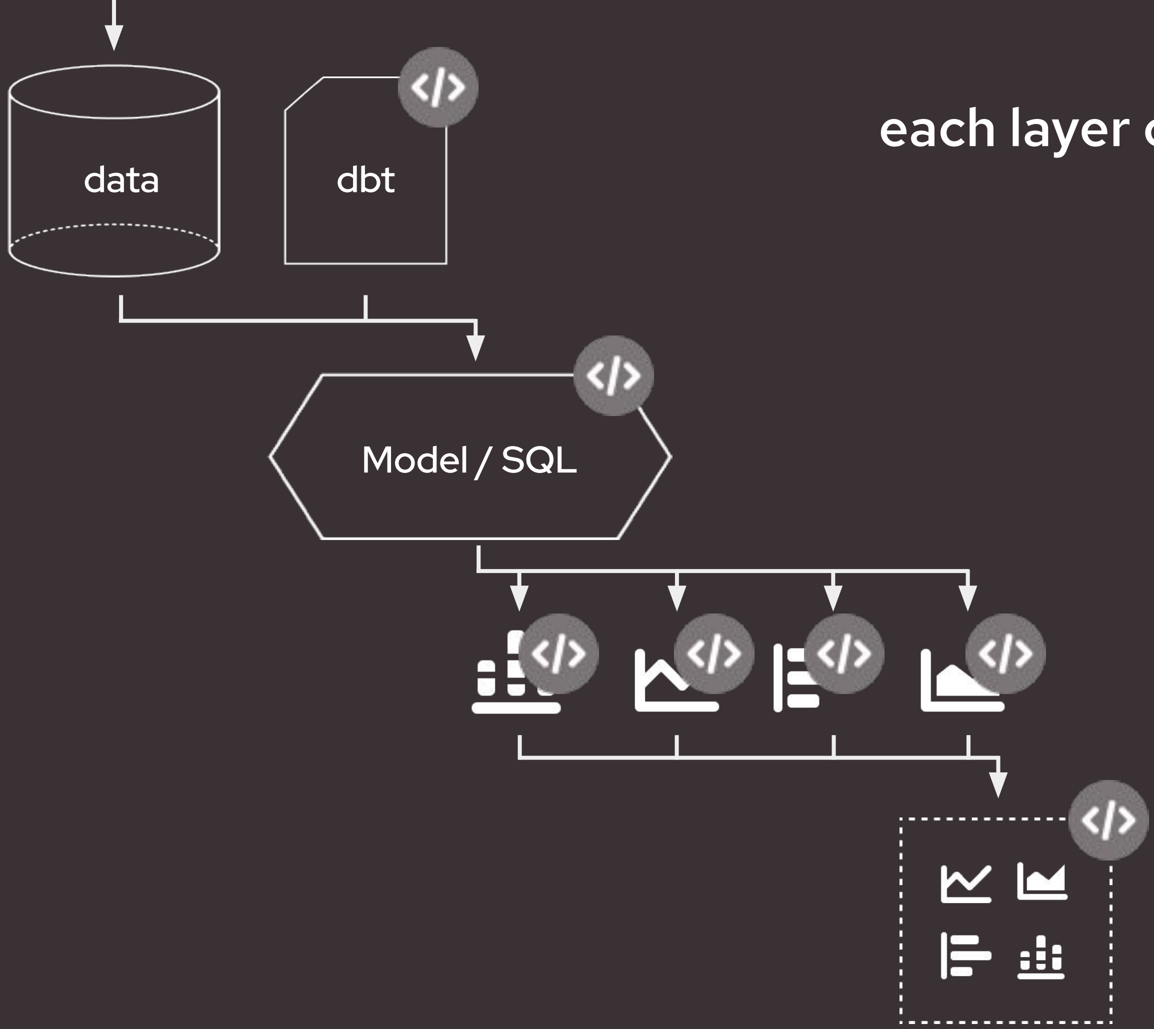
DataOps

Builds

Resource Lineage

Resource Lineage

each layer can be defined with code



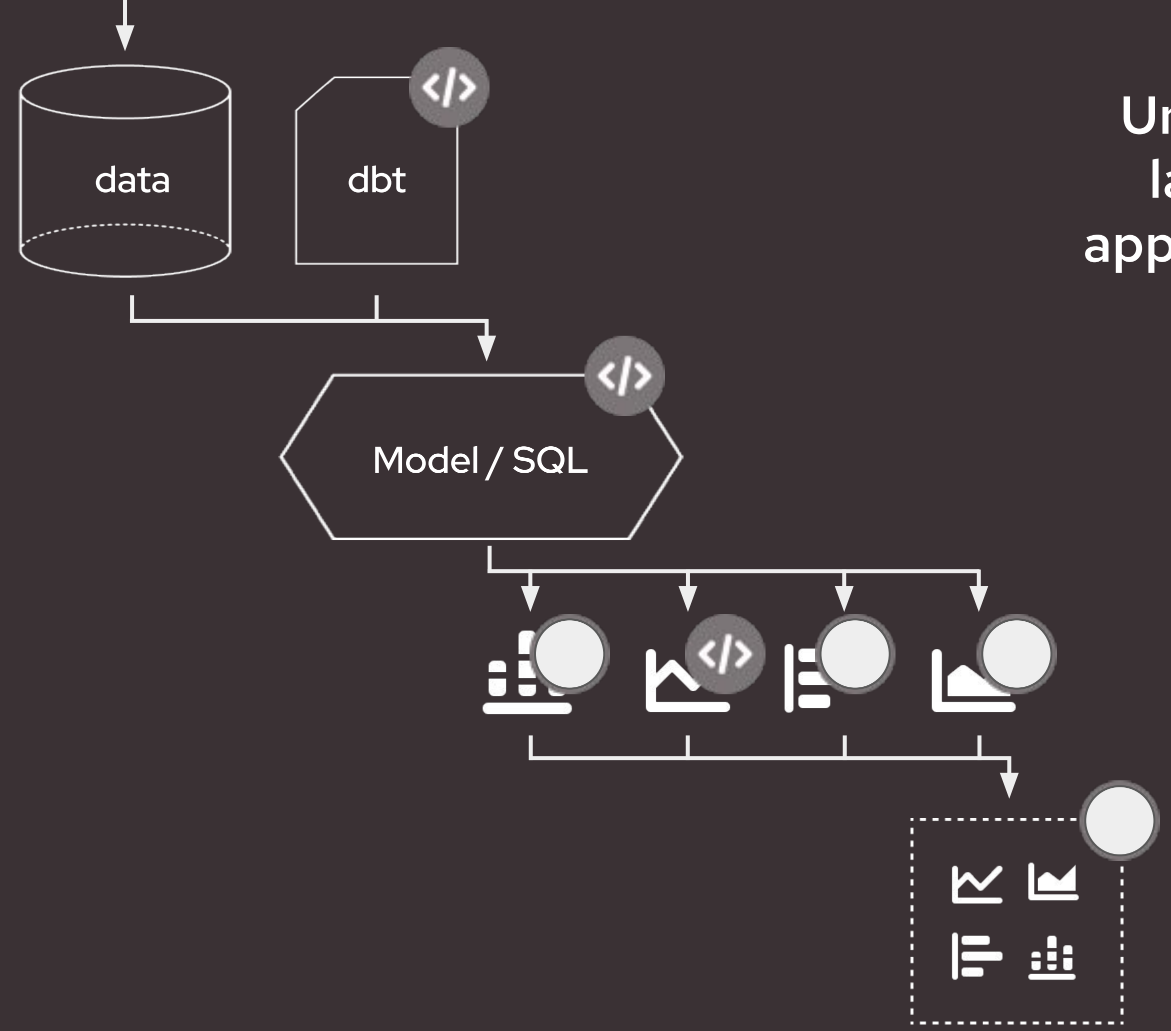
DataOps challenges

1. **Wide range of use cases:**
   a. **Governed metrics & dashboards (prioritize stability)**
   b. **Adhoc, exploratory analysis (prioritize flexibility)**

2. BI is inherently visual

3. Interdisciplinary collaboration

4. High number of dependencies



Unlike DevOps, some BI layers might never be appropriate for defining as code

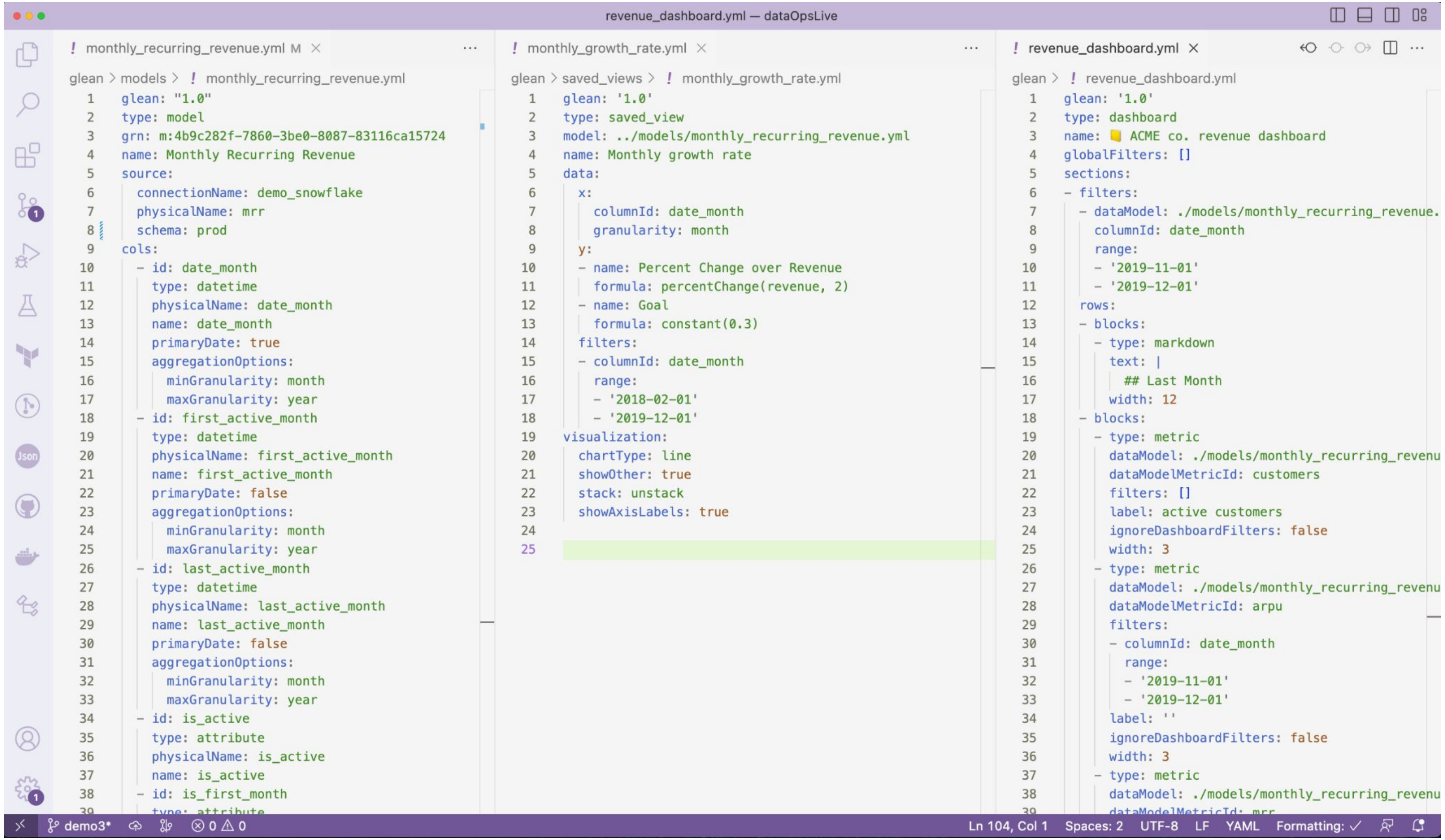
DataOps challenges

1. Big range of use cases:
   a. Governed metrics & dashboards (prioritize stability)
   b. Adhoc, exploratory analysis (prioritize flexibility)

2. **BI is inherently visual**

3. Interdisciplinary collaboration

4. High number of dependencies

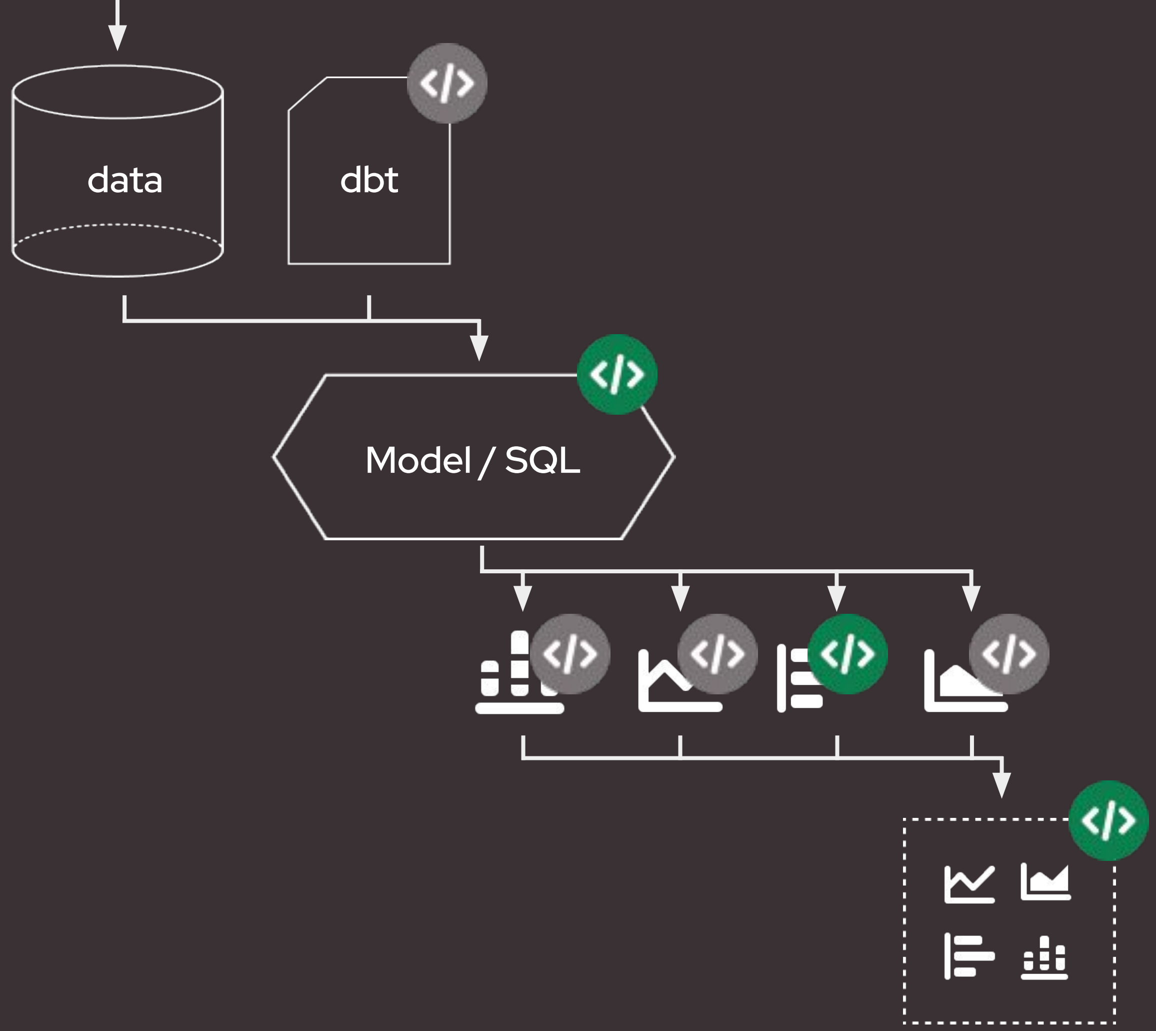

revenue_dashboard.yml — dataOpsLive
monthly_recurring_revenue.yml M
glean > models > monthly_recurring_revenue.yml
monthly_growth_rate.yml
glean > saved_views > monthly_growth_rate.yml
revenue_dashboard.yml
glean > revenue_dashboard.yml
demo3*
0 0
Ln 104, Col 1
Spaces: 2
UTF-8
LF
YAML
Formatting: ✓

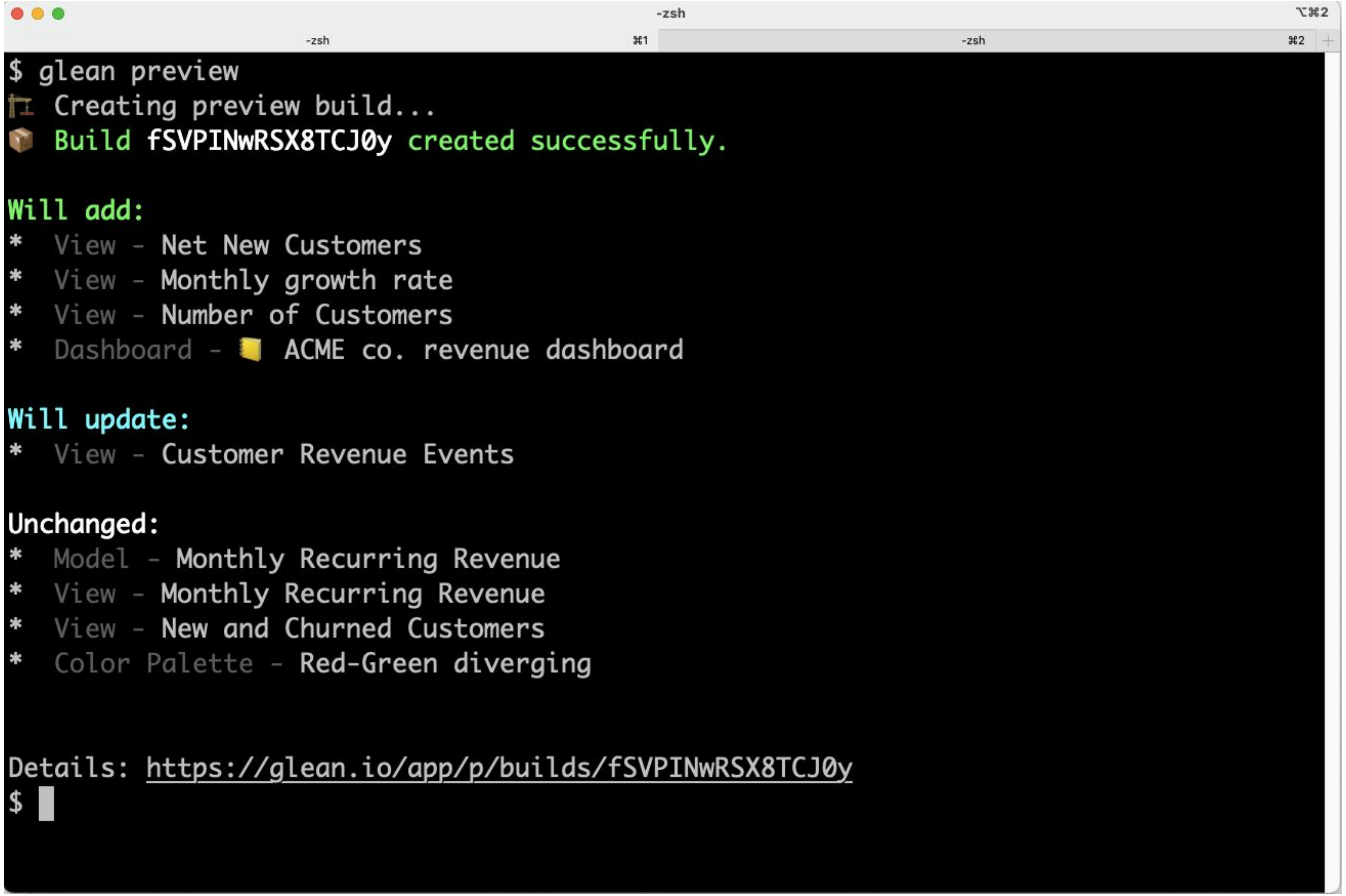
```yaml
glean: "1.0"
type: model
grn: m:4b9c282f-7860-3be0-8087-83116ca15724
name: Monthly Recurring Revenue
source:
  connectionName: demo_snowflake
  physicalName: mrr
  schema: prod
cols:
  - id: date_month
    type: datetime
    physicalName: date_month
    name: date_month
    primaryDate: true
    aggregationOptions:
      minGranularity: month
      maxGranularity: year
  - id: first_active_month
    type: datetime
    physicalName: first_active_month
    name: first_active_month
    primaryDate: false
    aggregationOptions:
      minGranularity: month
      maxGranularity: year
  - id: last_active_month
    type: datetime
    physicalName: last_active_month
    name: last_active_month
    primaryDate: false
    aggregationOptions:
      minGranularity: month
      maxGranularity: year
  - id: is_active
    type: attribute
    physicalName: is_active
    name: is_active
  - id: is_first_month
    type: attribute
```

```yaml
glean: '1.0'
type: saved_view
model: ../models/monthly_recurring_revenue.yml
name: Monthly growth rate
data:
  x:
    columnId: date_month
    granularity: month
  y:
  - name: Percent Change over Revenue
    formula: percentChange(revenue, 2)
  - name: Goal
    formula: constant(0.3)
  filters:
  - columnId: date_month
    range:
    - '2018-02-01'
    - '2019-12-01'
visualization:
  chartType: line
  showOther: true
  stack: unstack
  showAxisLabels: true

```

```yaml
glean: '1.0'
type: dashboard
name: 📒 ACME co. revenue dashboard
globalFilters: []
sections:
- filters:
  - dataModel: ./models/monthly_recurring_revenue.
    columnId: date_month
    range:
    - '2019-11-01'
    - '2019-12-01'
  rows:
  - blocks:
    - type: markdown
      text: |
        ## Last Month
      width: 12
  - blocks:
    - type: metric
      dataModel: ./models/monthly_recurring_revenu
      dataModelMetricId: customers
      filters: []
      label: active customers
      ignoreDashboardFilters: false
      width: 3
    - type: metric
      dataModel: ./models/monthly_recurring_revenu
      dataModelMetricId: arpu
      filters:
      - columnId: date_month
        range:
        - '2019-11-01'
        - '2019-12-01'
      label: ''
      ignoreDashboardFilters: false
      width: 3
    - type: metric
      dataModel: ./models/monthly_recurring_revenu
      dataModelMetricId: mrr
```

change tracking





```
$ glean preview
🏗 Creating preview build...
📦 Build fSVPINwRSX8TCJ0y created successfully.

Will add:
*  View - Net New Customers
*  View - Monthly growth rate
*  View - Number of Customers
*  Dashboard - 📒 ACME co. revenue dashboard

Will update:
*  View - Customer Revenue Events

Unchanged:
*  Model - Monthly Recurring Revenue
*  View - Monthly Recurring Revenue
*  View - New and Churned Customers
*  Color Palette - Red-Green diverging


Details: https://glean.io/app/p/builds/fSVPINwRSX8TCJ0y
$ 
```

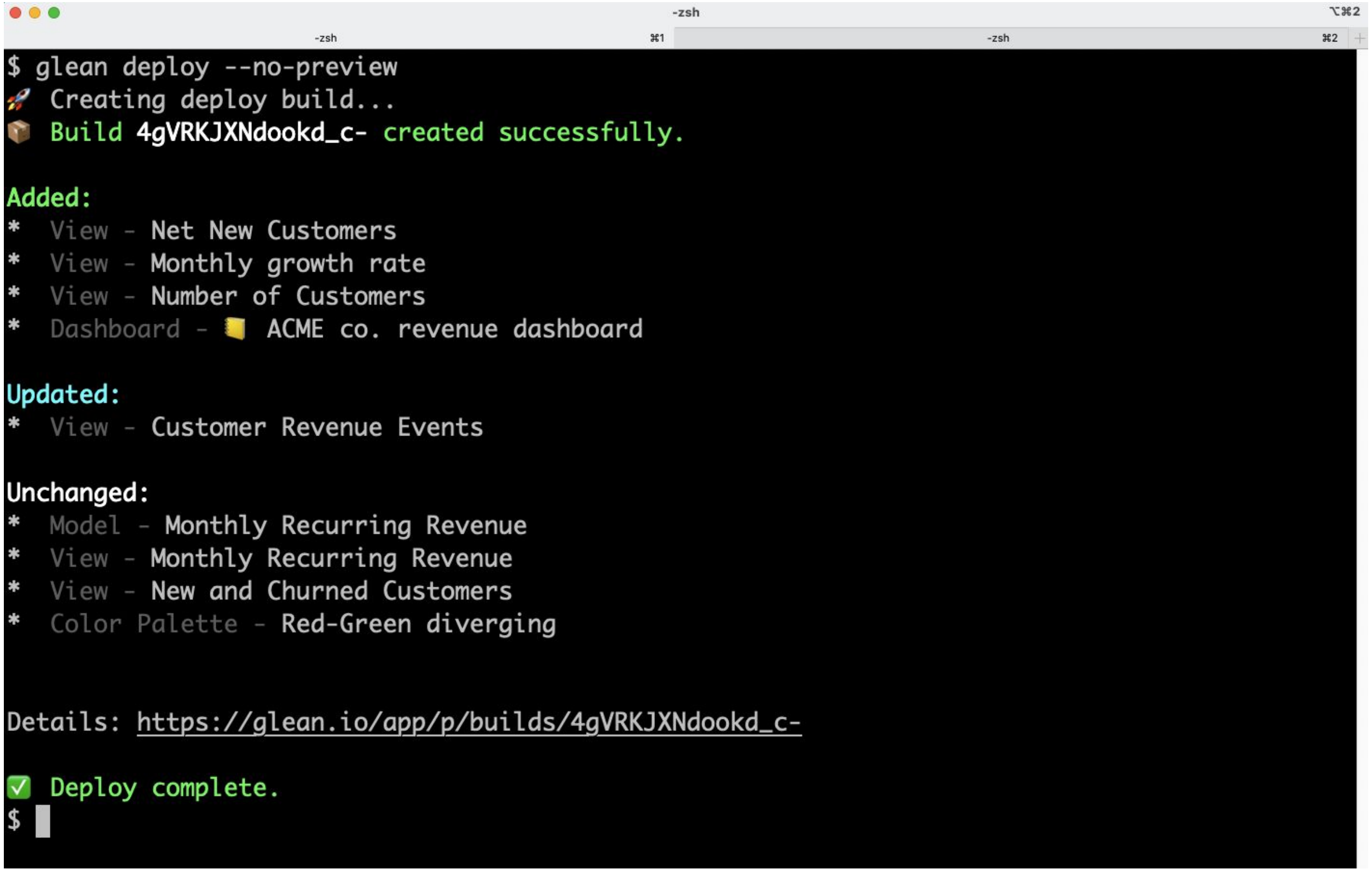
DataOps challenges

1. Big range of use cases:
   a. Governed metrics & dashboards (prioritize stability)
   b. Adhoc, exploratory analysis (prioritize flexibility)

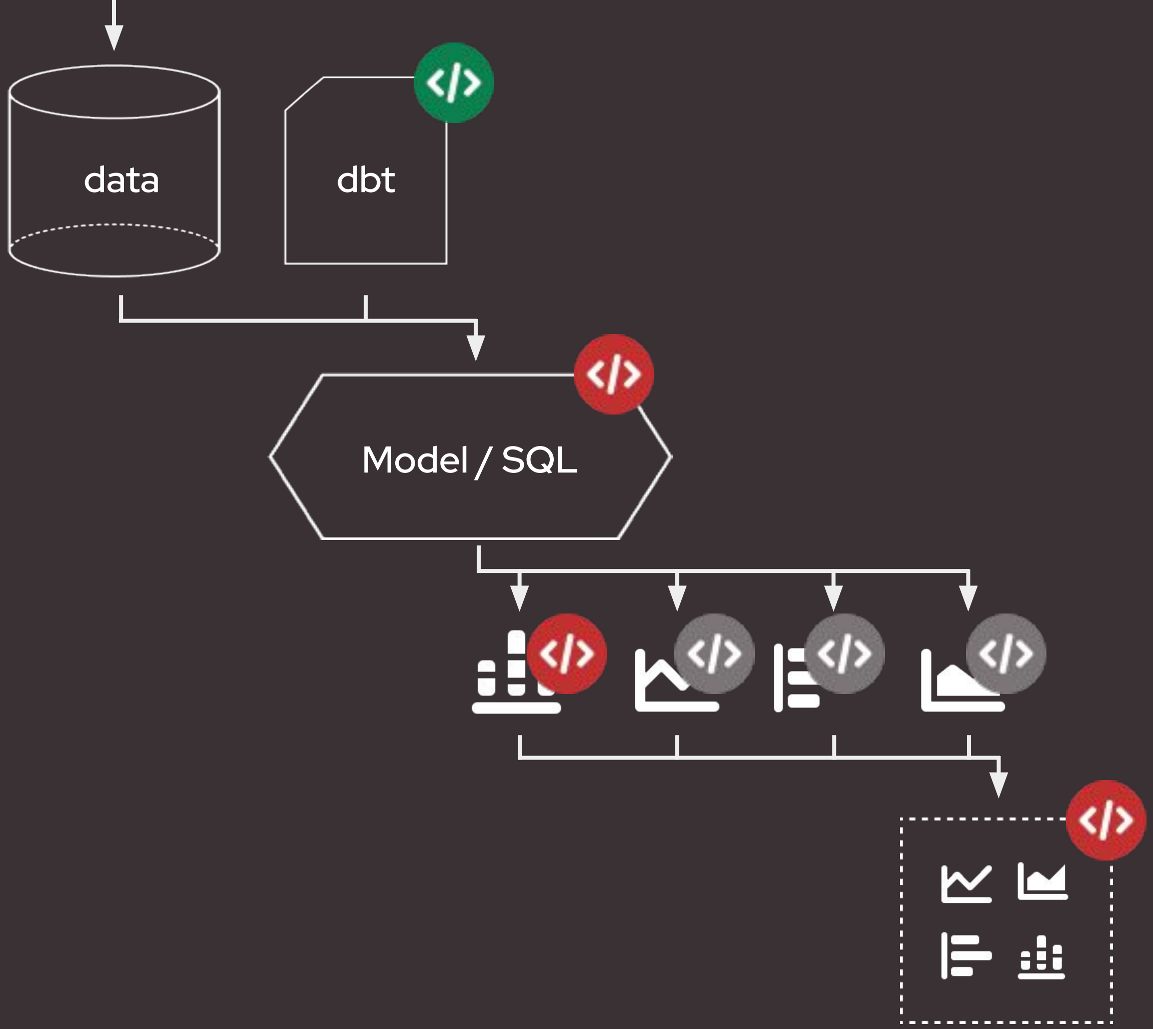
2. BI is inherently visual

3. **<u>Interdisciplinary collaboration</u>**

4. High number of dependencies



```
$ glean deploy --no-preview
🚀 Creating deploy build...
📦 Build 4gVRKJXNdookd_c- created successfully.

Added:
*  View - Net New Customers
*  View - Monthly growth rate
*  View - Number of Customers
*  Dashboard - 📒 ACME co. revenue dashboard

Updated:
*  View - Customer Revenue Events

Unchanged:
*  Model - Monthly Recurring Revenue
*  View - Monthly Recurring Revenue
*  View - New and Churned Customers
*  Color Palette - Red-Green diverging


Details: https://glean.io/app/p/builds/4gVRKJXNdookd_c-

✅ Deploy complete.
$
```


data
dbt
Model / SQL

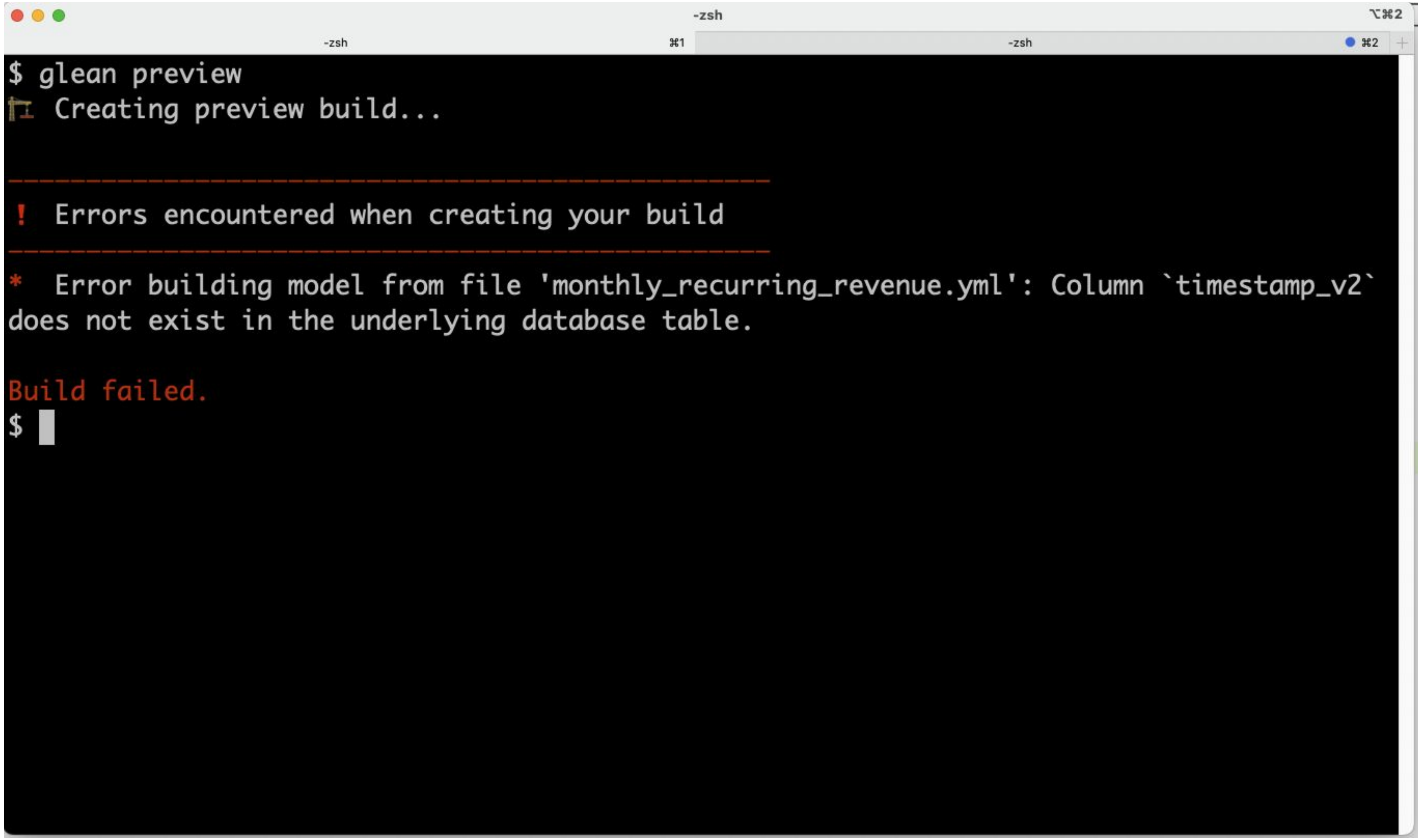
DataOps challenges

1. Big range of use cases:
   a. Governed metrics & dashboards (prioritize stability)
   b. Adhoc, exploratory analysis (prioritize flexibility)

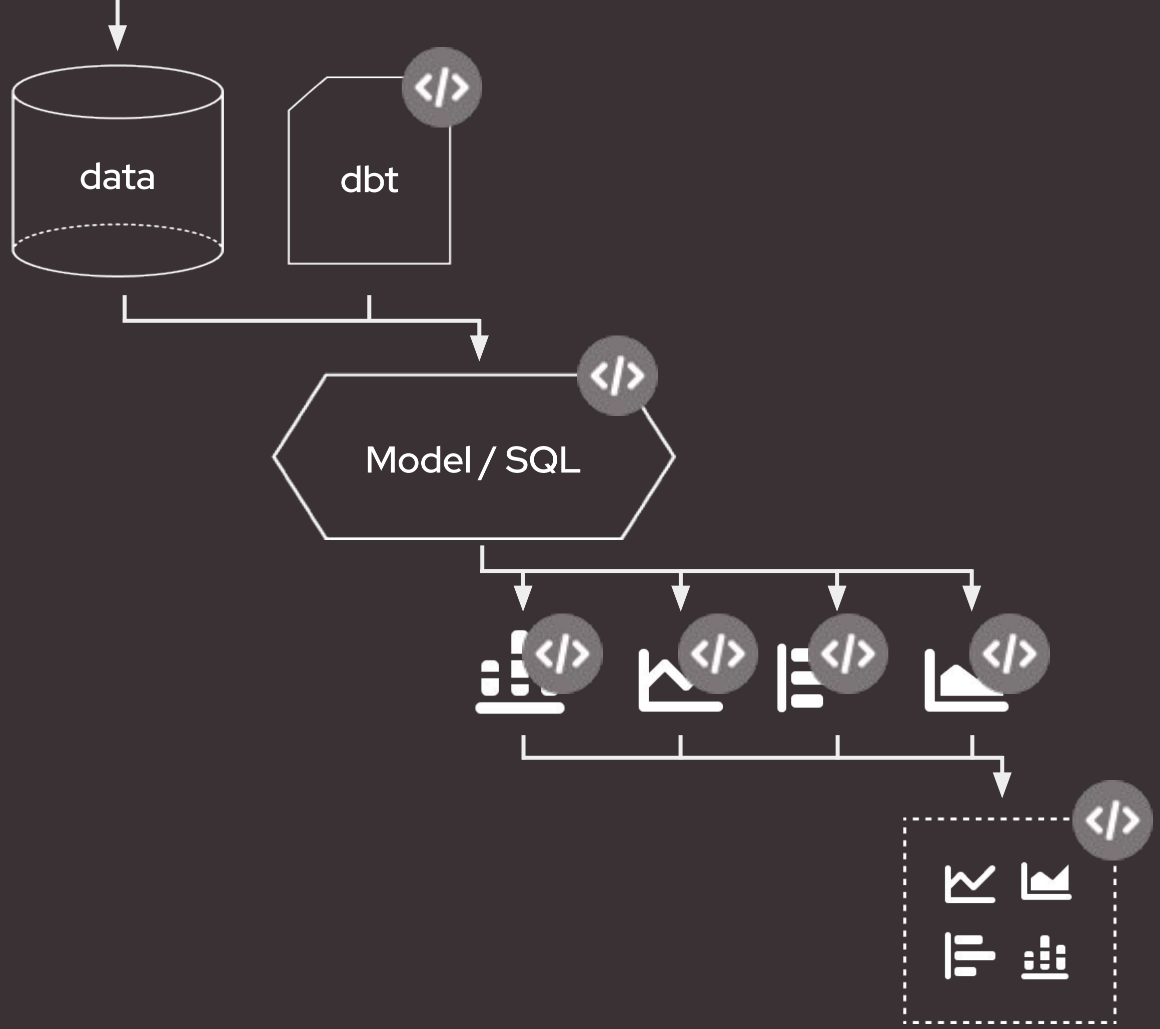
2. BI is inherently visual

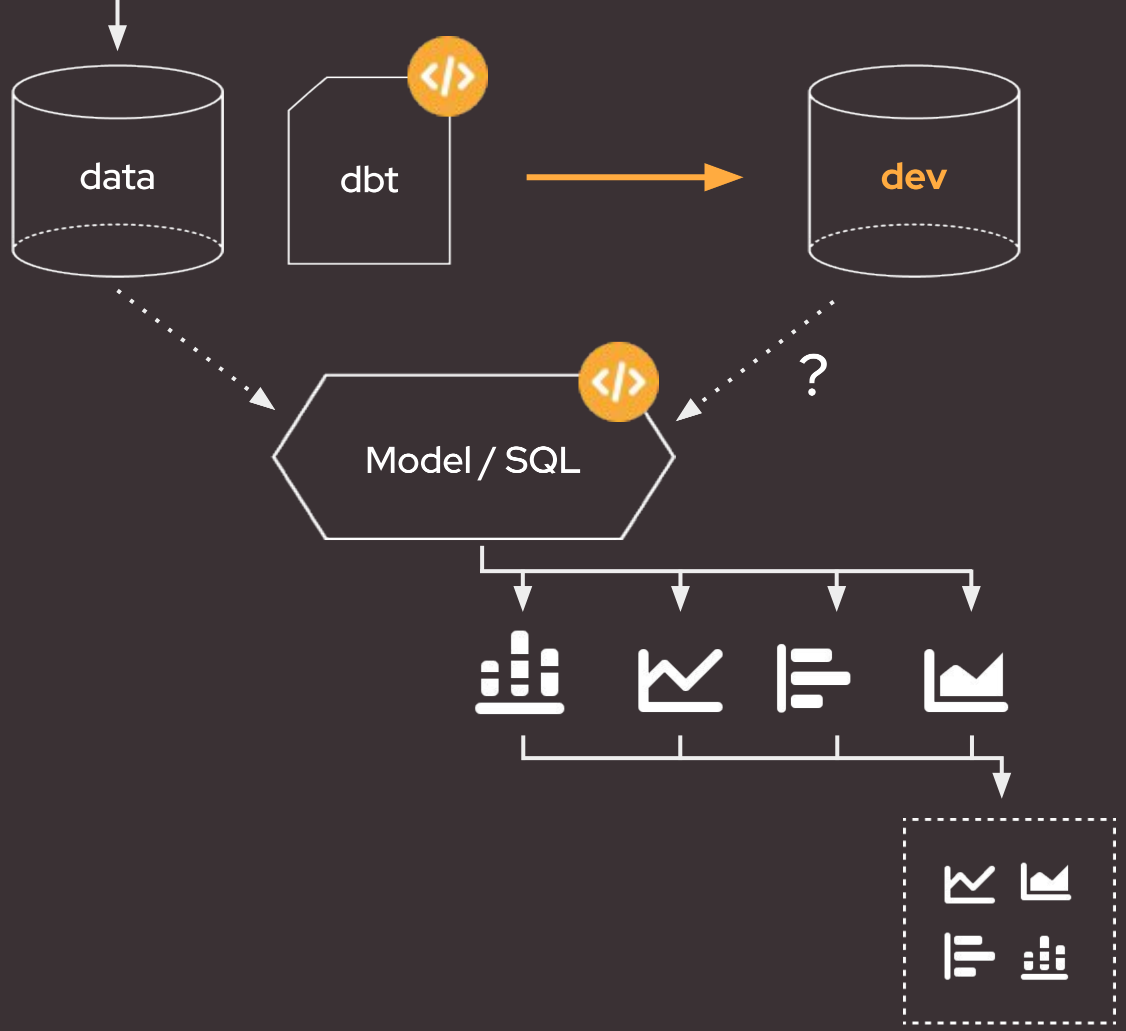
3. Interdisciplinary collaboration

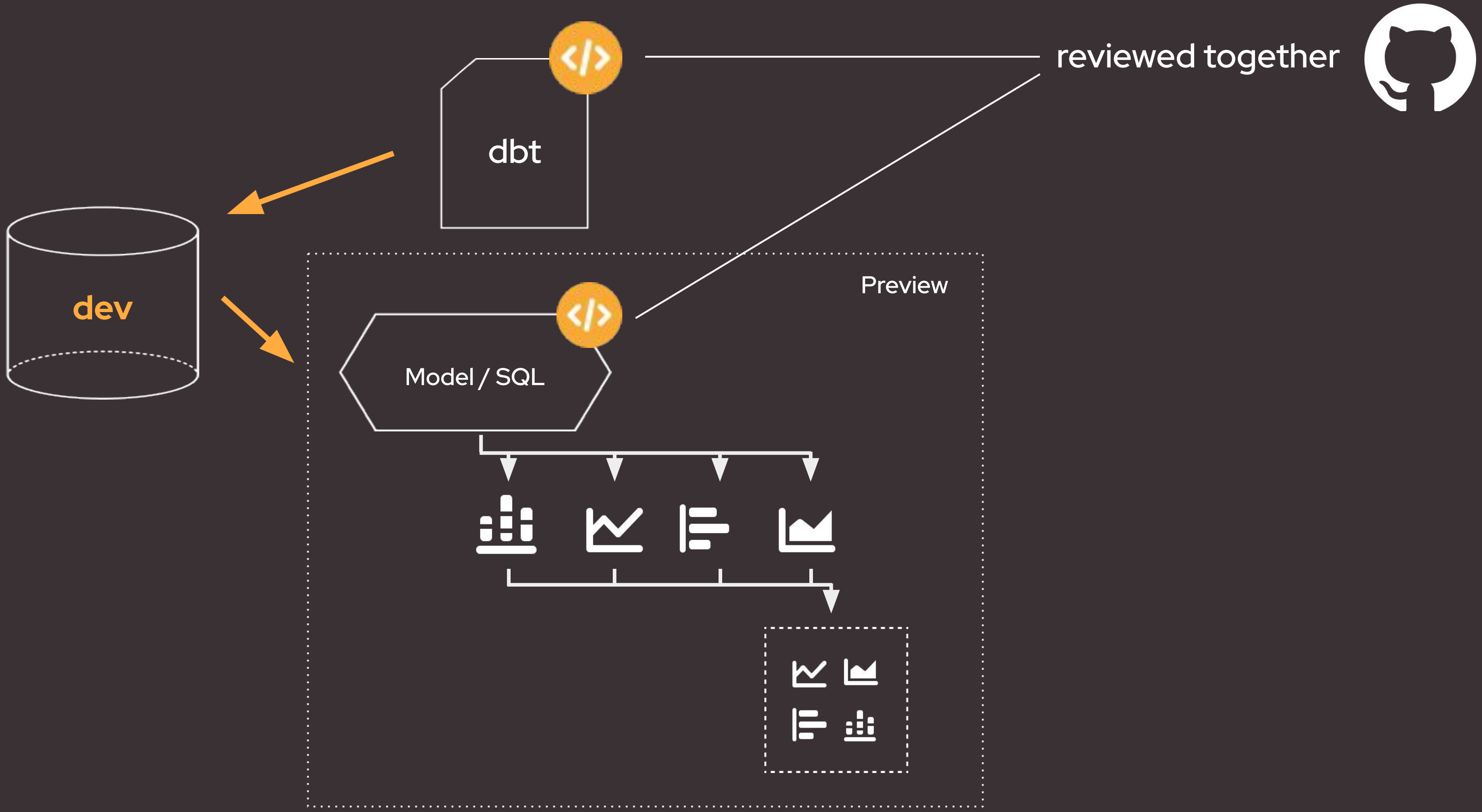
4. **<u>High number of dependencies</u>**

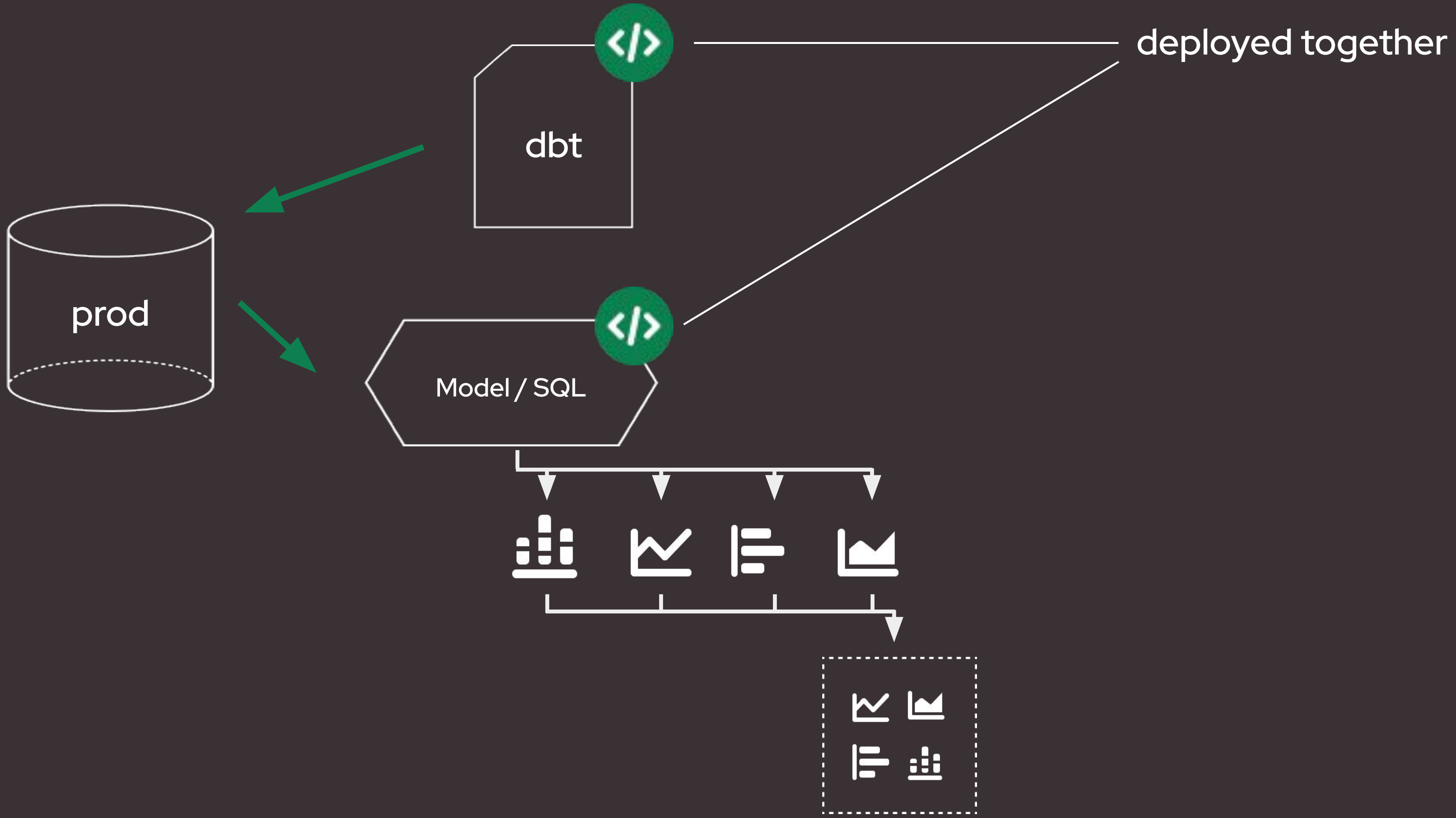


```
$ glean preview
🏗 Creating preview build...

------------------------------------------------------
!  Errors encountered when creating your build
------------------------------------------------------
*  Error building model from file 'monthly_recurring_revenue.yml': Column `timestamp_v2`
does not exist in the underlying database table.

Build failed.
$
```


data
dbt
Model / SQL


data
dbt
dev
?
Model / SQL


reviewed together
dbt
dev
Preview
Model / SQL


deployed together
dbt
prod
Model / SQL

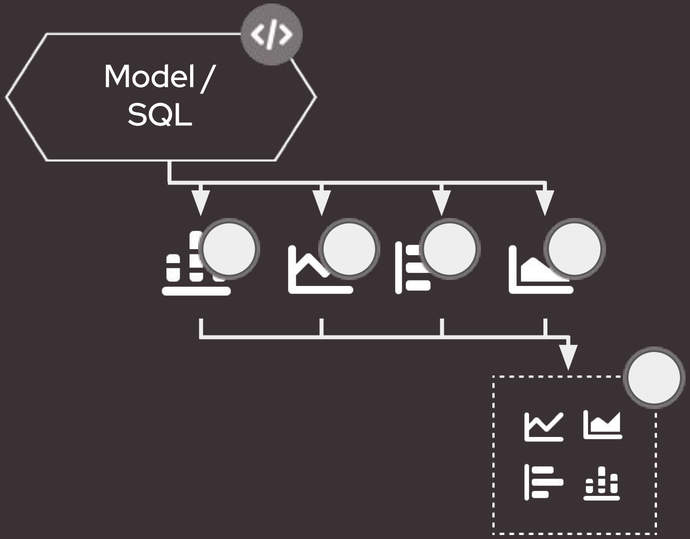
Choosing a DataOps Strategy

What’s your use case?

- Governed metrics + adhoc exploration
- Ensuring important dashboards don’t break
- Improving dev <> stakeholder collaboration
- Deploying changes more frequently

DataOps Strategy 1

Define your model as code, leave downstream dashboards and analyses as user-controlled



1. **Model as Code**

Advantages

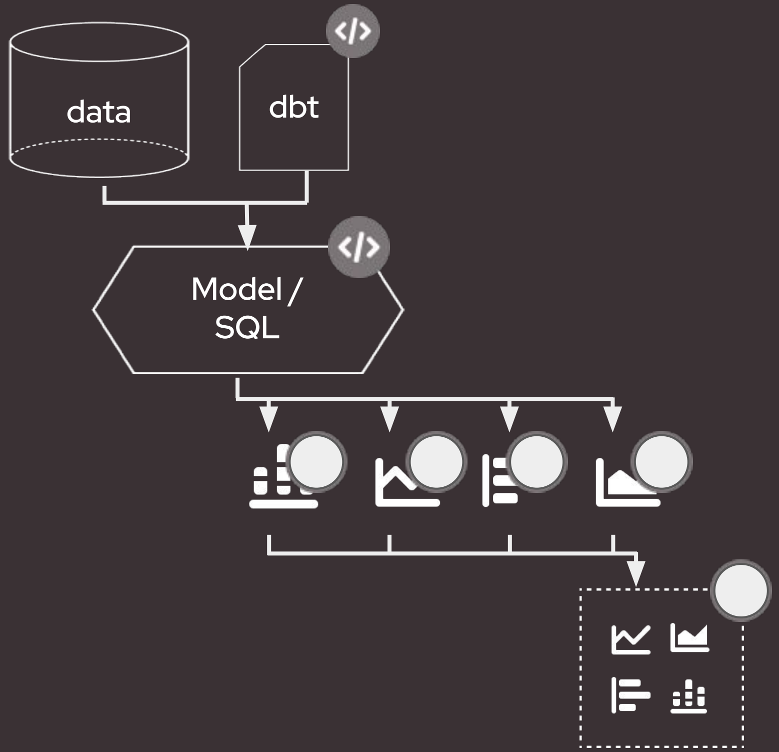
→ Keep your data pipelines and BI modeling layer in sync.

→ You can change and validate both in the same pull request.

Considerations

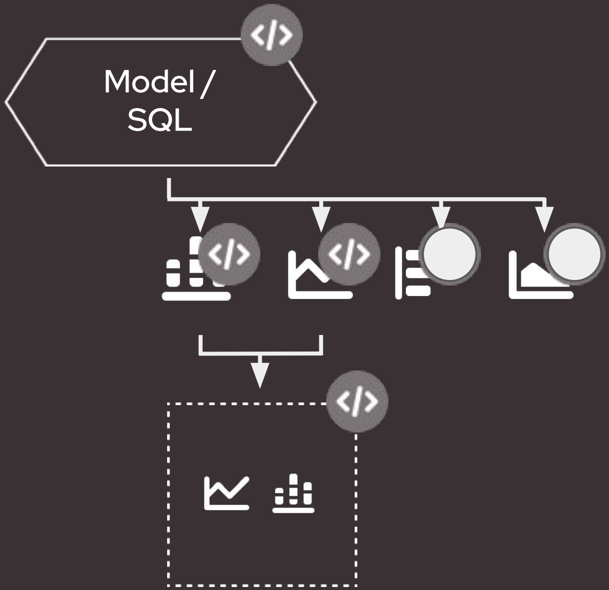
→ End users can change dashboards without review from the data team

→ Need a mature modeling layer so that it's hard to build a bad / incorrect chart



DataOps Strategy 2

Define an entire vertical slice as code: models, charts, dashboards.



2. Vertical slice

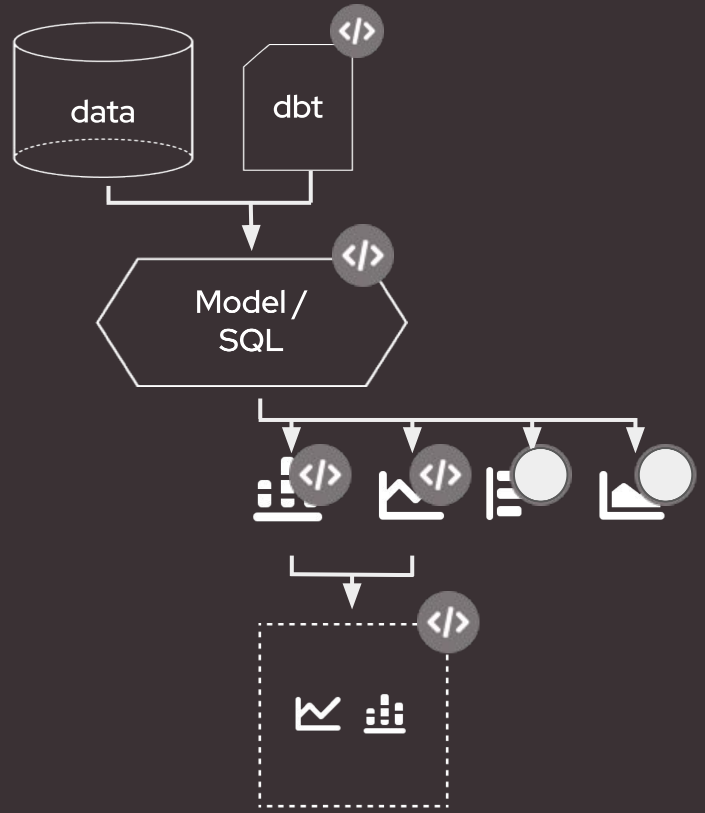
Advantages

→ Tighter control over the end result – strong validation that your dashboard will not be broken.

Considerations

→ Works best for visualizations / dashboards that won't change very often.

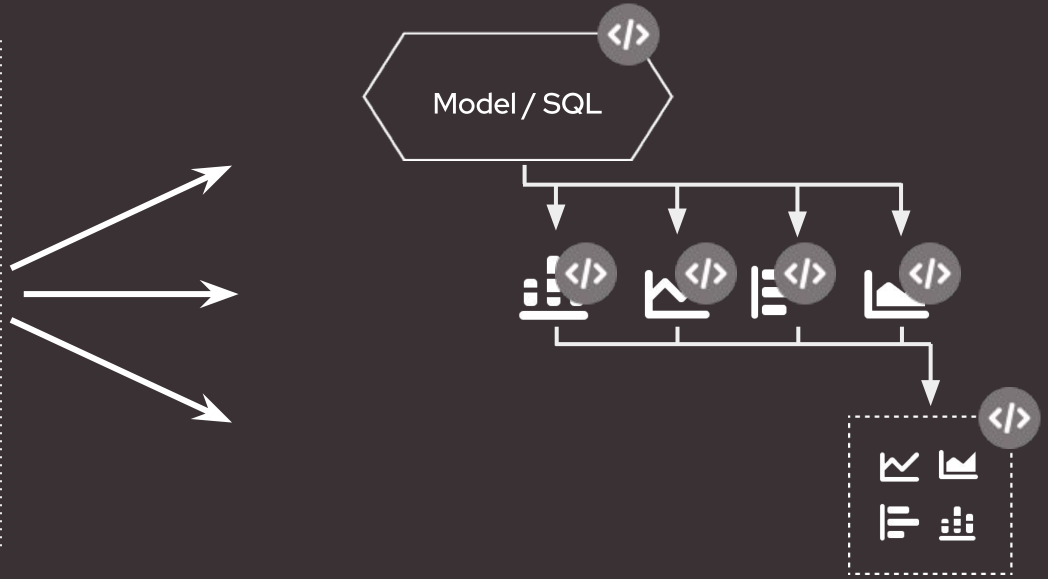
→ Adds friction to updating the end product



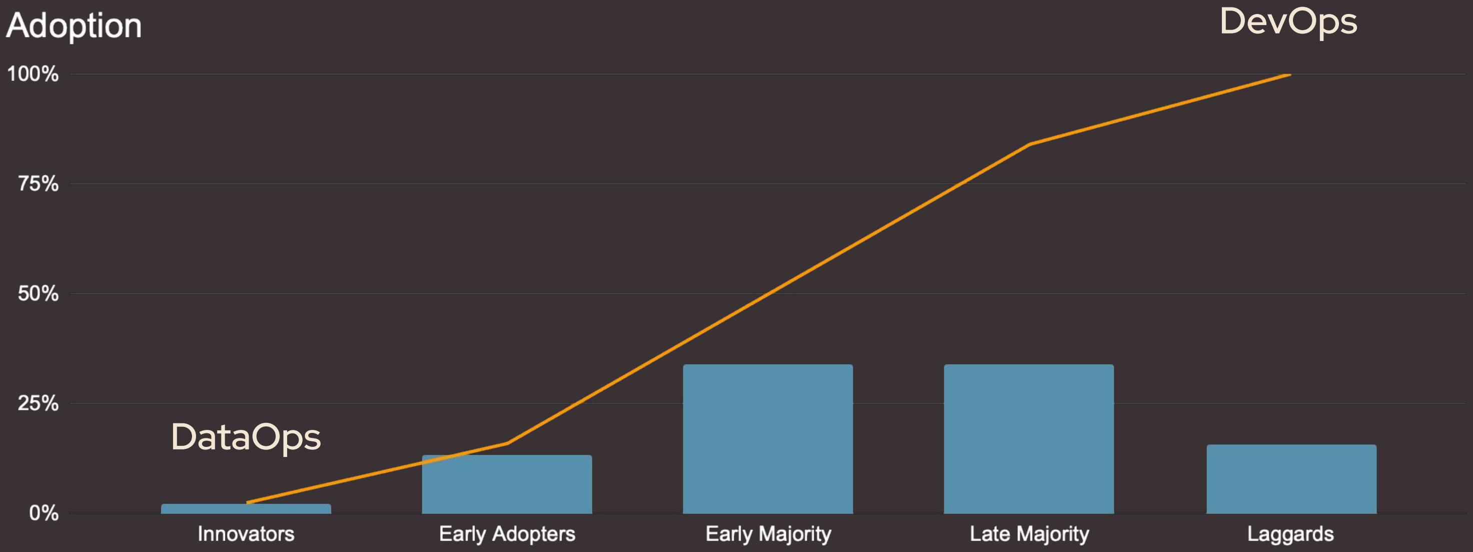
DataOps Strategy 3

Use code to define templates that can be deployed and customized as needed.

```
template(
   schema,
   data)
```



Still a lot to invent here...!



Summary

- Fragile BI leads to lack of trust
- DataOps helps by applying engineering best practices to BI development
- To be effective, these strategies and tools need to bridge the gap between UI & code and support a diverse set of use cases

Thank you!

glean

https://glean.io

hashboard

https://hashboard.com