

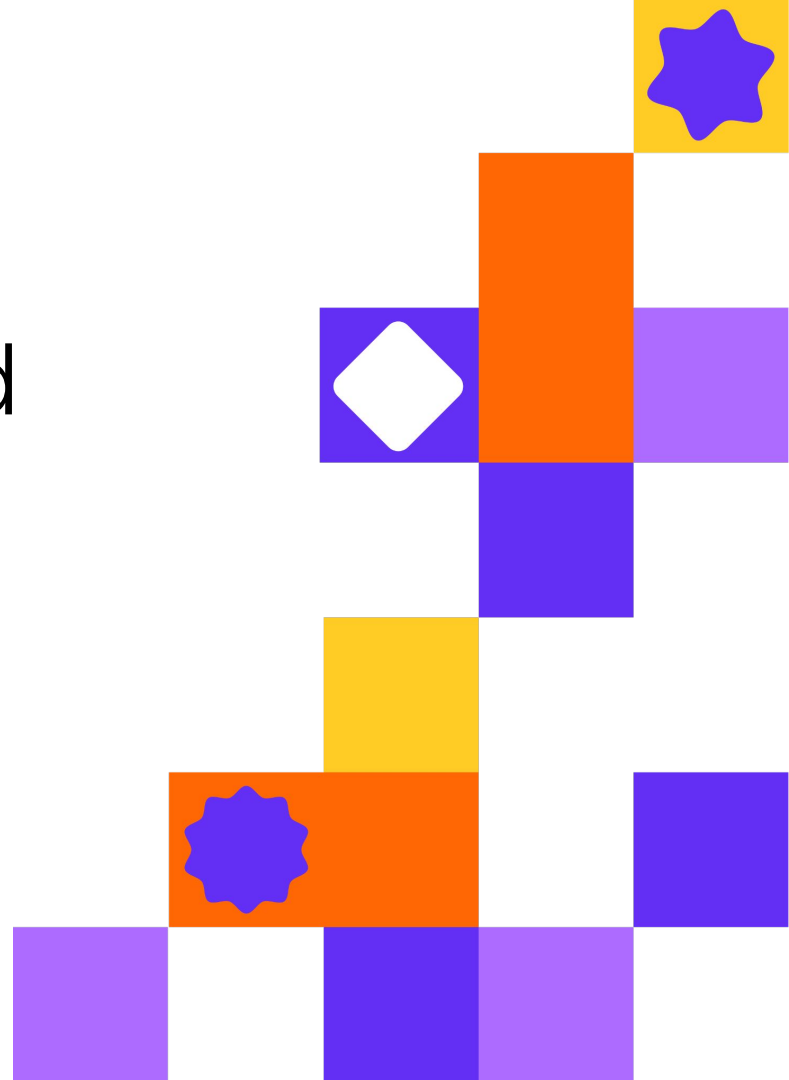


So you want to build a data mesh

Jenna Jordan

October 13-16, 2025

CRAFTED BY





Agenda

1

Introductions

Let's get to know each other

2

What is data mesh?

And why does it matter for implementing dbt Mesh?

3

Domain Ownership

How to go from 1 dbt project to many - the easy way

4

Data as a Product

dbt features you need to build dbt models according to the 8 characteristics of data products

5

Self Serve Data Platform

3 tips for making data product model development easy for your domain teams

6

Federated Computational Governance

CI/CD is your most important feedback loop

7

People & Process

The squishy stuff

8

Questions?

I know you have them



Meet the speaker



Jenna Jordan

Sr Analytics Engineer

Ratio PBC
(formerly Analytics8)

jennajordan.me



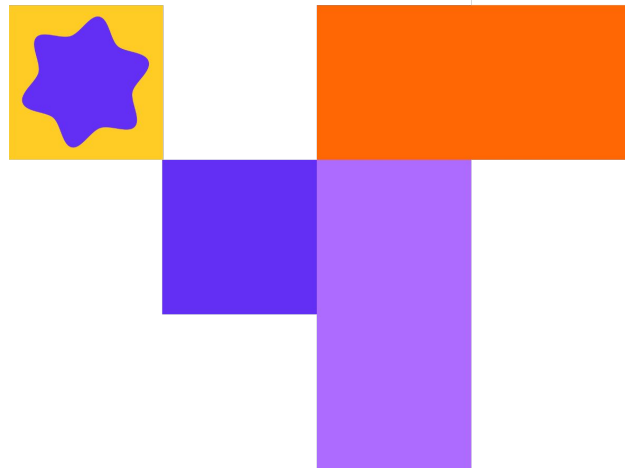
Meet the audience! Raise your hand if...

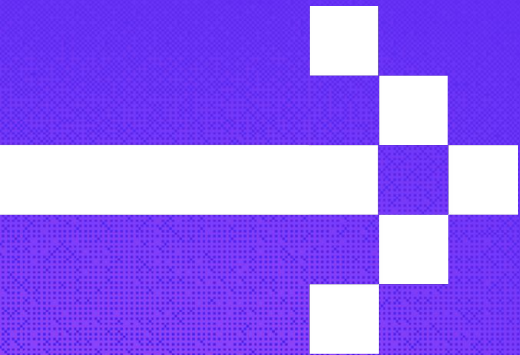
- You have a functioning dbt Mesh in operation/production, with many projects that depend on each other
- You are actively working towards a dbt Mesh, and are in the process of figuring out how to get it fully operational
- You have one dbt project, but you are interested in expanding to more and having a dbt Mesh of many projects
- You are interested in (or curious about) the idea of data mesh and/or dbt mesh
- Your organization has an Enterprise or Enterprise+ plan on dbt Cloud
- Your organization has a team plan on dbt Cloud
- Your organization is sticking to dbt Core only – open source FTW



What is data mesh?

And how does dbt Mesh fit in?





Data mesh, a
socio-technical approach
to **distributed data
management at scale**,
provides a driving
philosophy for how to
evolve your data practice.

A decorative graphic on the left side of the slide, consisting of a horizontal white bar and several white squares of varying sizes arranged in a grid-like pattern.

dbt Mesh, a set of dbt features that empower data teams to work independently and collaboratively, provides tooling for **distributed data management at scale.**

O'REILLY®

Data Mesh

Delivering Data-Driven Value at Scale

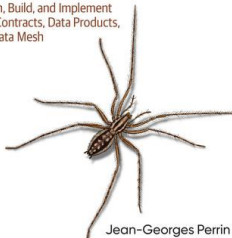


Zhamak Dehghani

O'REILLY®

Implementing Data Mesh

Design, Build, and Implement
Data Contracts, Data Products,
and Data Mesh



Jean-Georges Perrin
& Eric Broda
Foreword by Scott Hileman

Data Mesh IN ACTION

Jacek Michalik
Sven Bolzon
Marin Sivak
with Marina Jurechowa
Foreword by George Ford

HANNA



martinFowler.com

Refactoring Agile Architecture About Thoughtworks

How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh

Many enterprises are investing in their next generation data lake, with the hope of democratizing data at scale to provide business insights and ultimately make automated intelligent decisions. Data platforms based on the data lake architecture have common failure modes that lead to unfulfilled promises at scale. To address these failure modes we need to shift from the centralized paradigm of a lake, or its predecessor data warehouse. We need to shift to a paradigm that draws from modern distributed architecture: considering domains as the first class concern, applying platform thinking to create self-serve data infrastructure, and treating data as a product.

20 May 2019



Zhamak Dehghani

CONTENTS

The current enterprise data platform architecture
Architectural failure modes
Centralized and monolithic
Coupled pipeline decomposition

martinFowler.com

Refactoring Agile Architecture About Thoughtworks

Data Mesh Principles and Logical Architecture

Our aspiration to augment and improve every aspect of business and life with data, demands a paradigm shift in how we manage data at scale. While the technology advances of the past decade have addressed the scale of volume of data and data processing compute, they have failed to address scale in other dimensions: changes in the data landscape, proliferation of sources of data, diversity of data use cases and users, and speed of response to change. Data mesh addresses these dimensions, founded in four principles: domain-oriented decentralized data ownership and architecture, data as a product, self-serve data infrastructure as a platform, and federated computational governance. Each principle drives a new logical view of the technical architecture and organizational structure.

03 December 2020

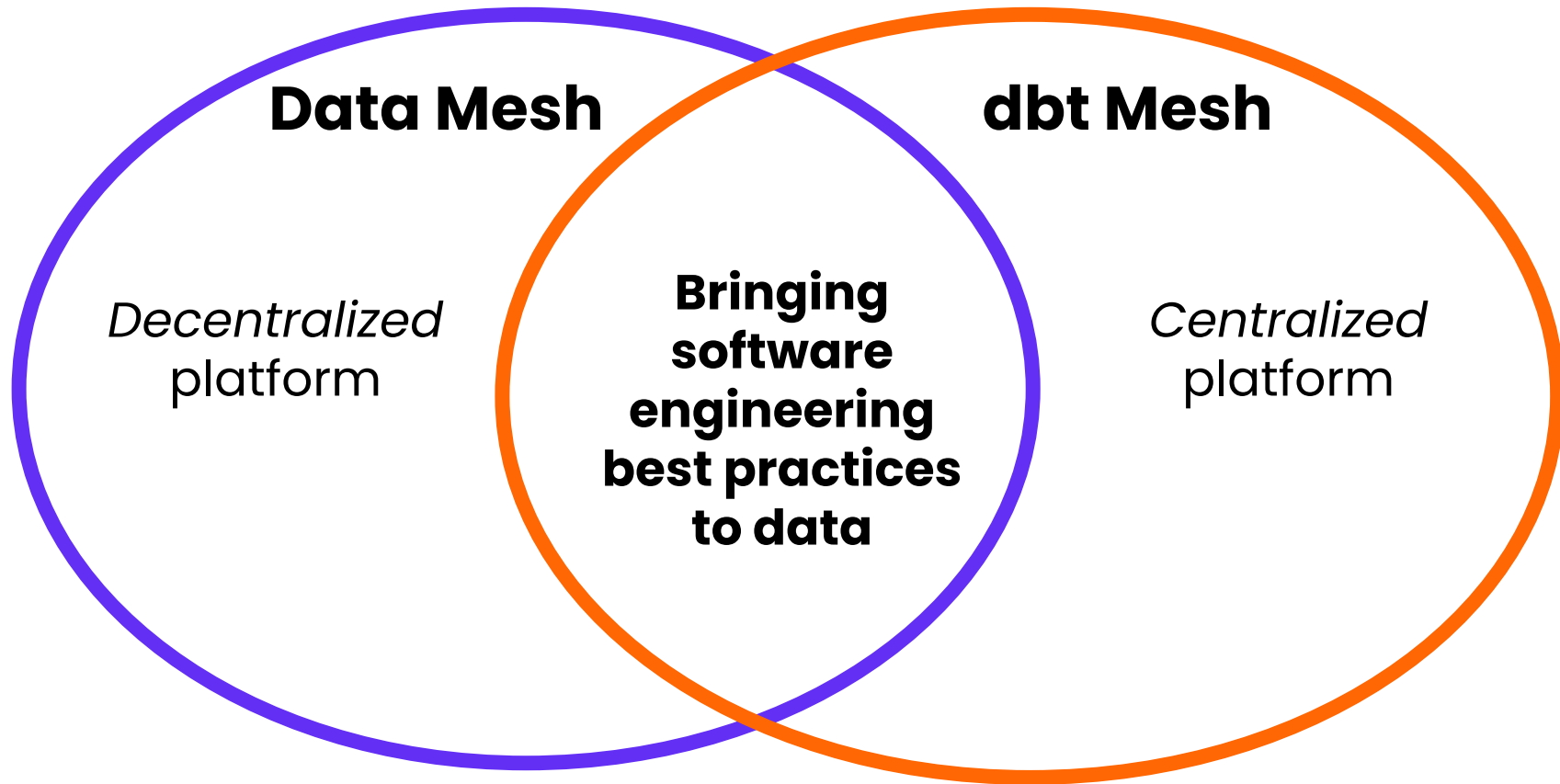


Zhamak Dehghani

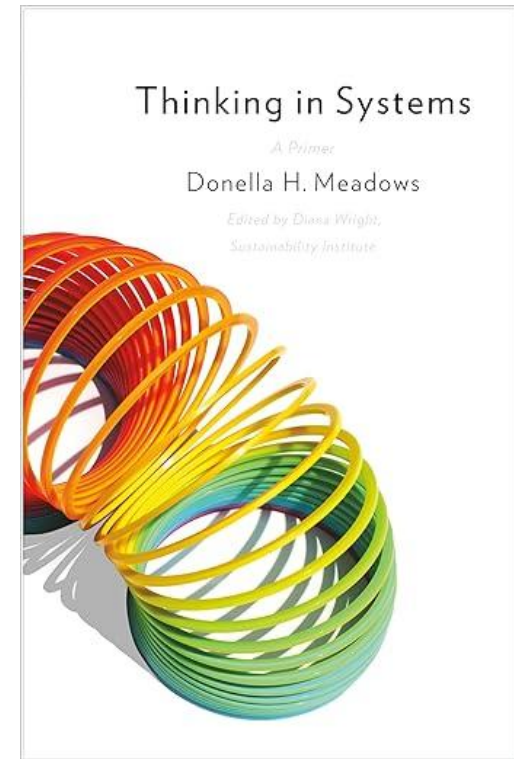
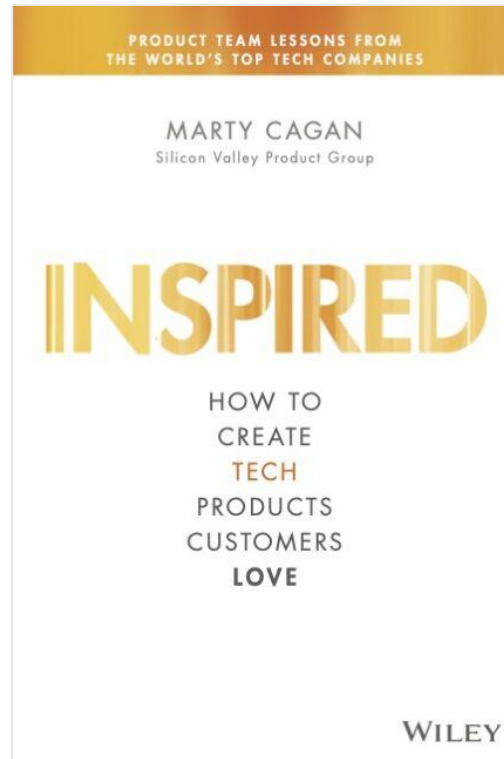
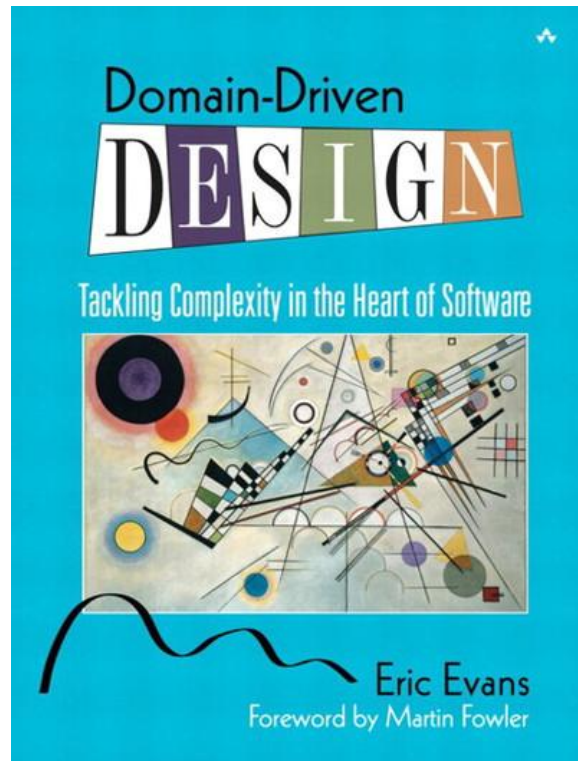
CONTENTS

The great divide of data
Core principles and logical architecture of data mesh
Domain Ownership
Logical architecture: domain-oriented data and compute

Some books and blog posts to read if you want to learn more



Data mesh and dbt are united by a foundational mission

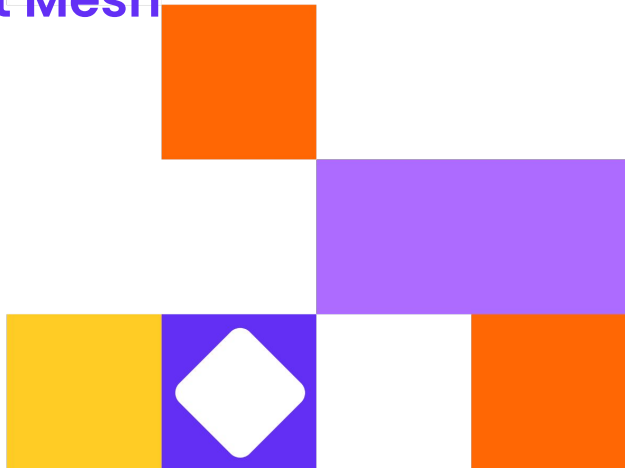


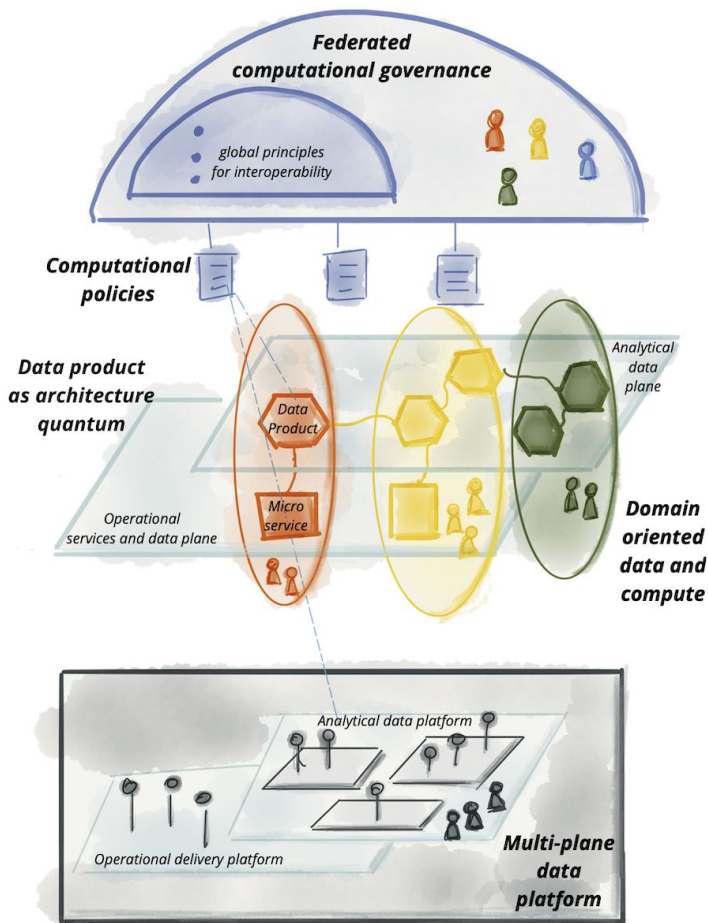
Books that influenced Zhamak Dehghani's thinking for data mesh



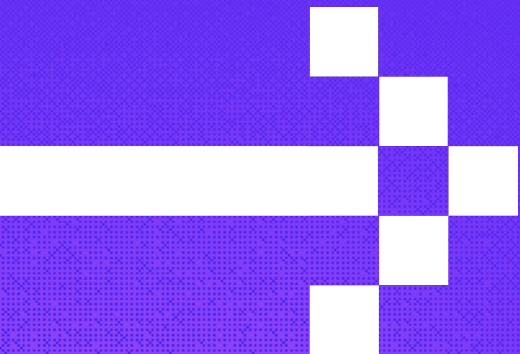
The 4 principles of data mesh

and how to follow them while building a dbt Mesh

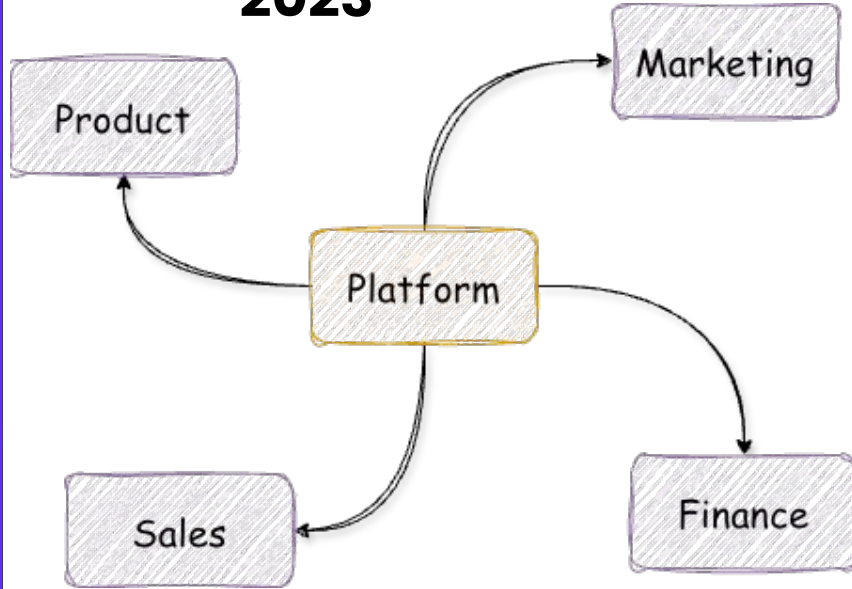
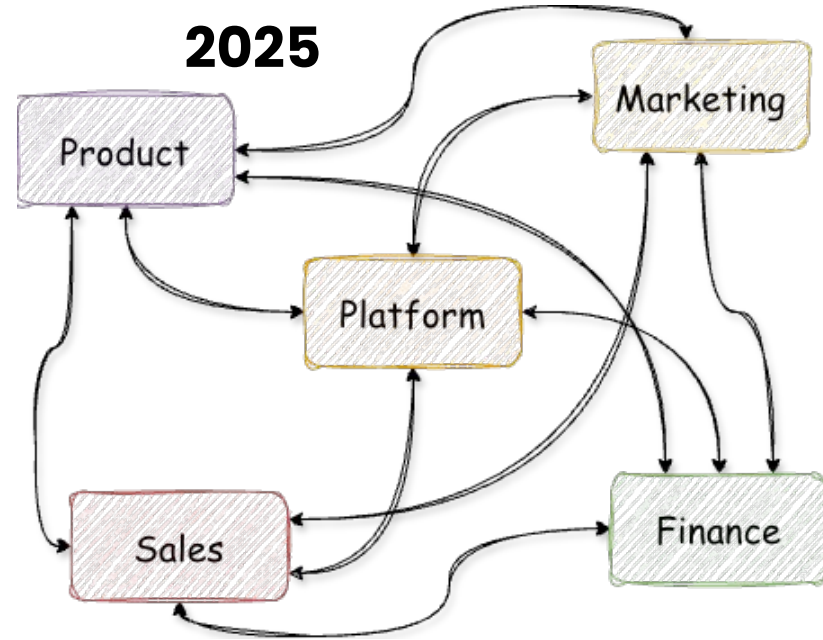




1. Domain Ownership
2. Data as a product
3. Self-serve Data Platform
4. Federated Computational Governance



Principle 1: Domain Ownership

2023**2025**

Hub & Spoke pattern vs... a meshier pattern



Pro Tip: don't start
with multiple dbt
projects!

Where does your
organization fit on
this spectrum?

dbt Mesh Organizational User Profiles



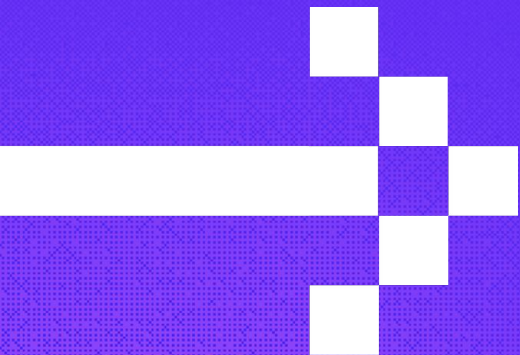
Aligned with the data
mesh approach, but
brand new to dbt

dbt veteran with a
spaghetti monster DAG
ready for change



Transitioning from 1 dbt project to many

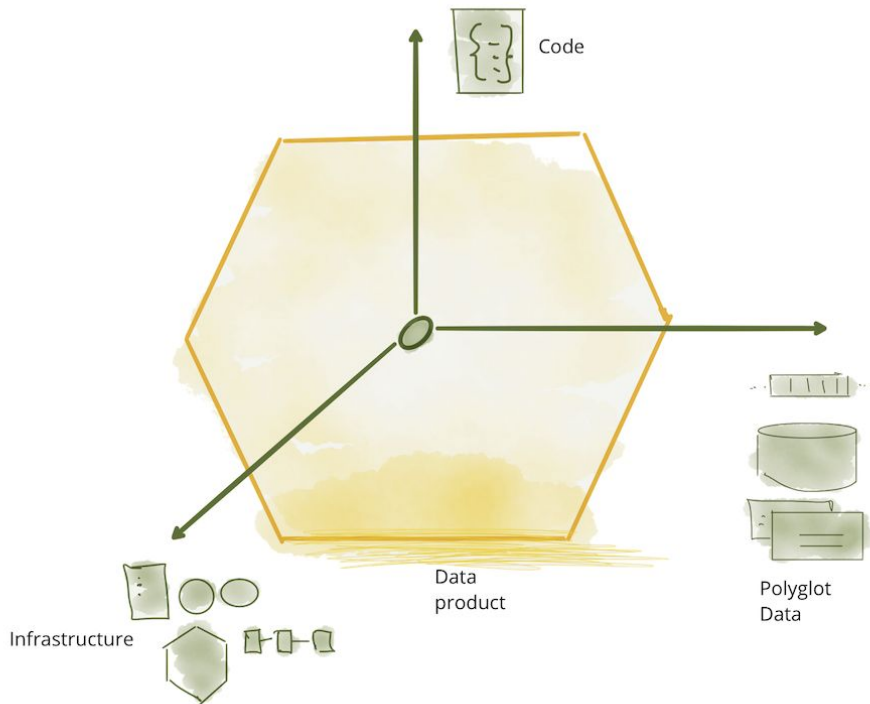
1. dbt **Groups** to draft domains
2. Align project folder structure to the groups
3. Update **model access**:
 - a. default to “private”
 - b. decide which models should be “public”
4. Transition dbt project management responsibilities to domain teams
 - a. Enforce domain ownership in PRs with a codeowners file
 - b. Domain ownership over dbt jobs (orchestration)
5. Domain teams spin out into their own dbt projects as they are ready



Principle 2: Data as a Product

8 Characteristics of Successful Data Products

1. Discoverable
2. Addressable
3. Understandable
4. Trustworthy & Useful
5. Natively Accessible
6. Interoperable
7. Valuable on its own
8. Secure





8 Characteristics of Successful Data Products

1. **Discoverable** meta config, controlled vocabulary
2. Addressable
3. Understandable
4. Trustworthy & Useful
5. Natively Accessible
6. Interoperable
7. Valuable on its own
8. Secure



8 Characteristics of Successful Data Products

1. Discoverable meta config, controlled vocabulary
2. **Addressable** model contracts, versions, deprecation_date
3. Understandable
4. Trustworthy & Useful
5. Natively Accessible
6. Interoperable
7. Valuable on its own
8. Secure



8 Characteristics of Successful Data Products

1. Discoverable meta config, controlled vocabulary
2. Addressable model contracts, versions, deprecation_date
3. **Understandable** doc blocks, enhanced descriptions
4. Trustworthy & Useful
5. Natively Accessible
6. Interoperable
7. Valuable on its own
8. Secure



8 Characteristics of Successful Data Products

1. Discoverable meta config, controlled vocabulary
2. Addressable model `contracts`, `versions`, `deprecation_date`
3. Understandable `doc blocks`, enhanced descriptions
4. **Trustworthy & Useful** SLOs, `data tests`, `freshness checks`
5. Natively Accessible
6. Interoperable
7. Valuable on its own
8. Secure



8 Characteristics of Successful Data Products

1. Discoverable meta config, controlled vocabulary
2. Addressable model **contracts**, **versions**, **deprecation_date**
3. Understandable **doc blocks**, enhanced descriptions
4. Trustworthy & Useful SLOs, **data tests**, **freshness checks**
5. **Natively Accessible** Cross-platform/Iceberg, **Semantic Layer**
6. Interoperable
7. Valuable on its own
8. Secure



8 Characteristics of Successful Data Products

1. Discoverable **meta** config, controlled vocabulary
2. Addressable model **contracts**, **versions**, **deprecation_date**
3. Understandable **doc blocks**, enhanced descriptions
4. Trustworthy & Useful SLOs, **data tests**, **freshness checks**
5. Natively Accessible Cross-platform/Iceberg, **Semantic Layer**
6. **Interoperable** **Semantic Layer**, naming conventions
7. Valuable on its own
8. Secure



8 Characteristics of Successful Data Products

- | | |
|-------------------------------|---|
| 1. Discoverable | meta config, controlled vocabulary |
| 2. Addressable | model contracts, versions, deprecation_date |
| 3. Understandable | doc blocks, enhanced descriptions |
| 4. Trustworthy & Useful | SLOs, data tests, freshness checks |
| 5. Natively Accessible | Cross-platform/Iceberg, Semantic Layer |
| 6. Interoperable | Semantic Layer, naming conventions |
| 7. Valuable on its own | OBT, sql + yml + md + data + lineage |
| 8. Secure | |



8 Characteristics of Successful Data Products

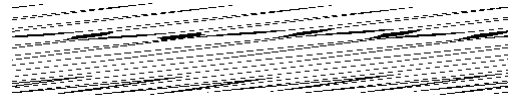
1. Discoverable **meta** config, controlled vocabulary
2. Addressable model **contracts**, **versions**, **deprecation_date**
3. Understandable **doc blocks**, enhanced descriptions
4. Trustworthy & Useful SLOs, **data tests**, **freshness checks**
5. Natively Accessible Cross-platform/Iceberg, **Semantic Layer**
6. Interoperable **Semantic Layer**, naming conventions
7. Valuable on its own OBT, sql + yml + md + data + lineage
8. **Secure** **grants** config, terraform

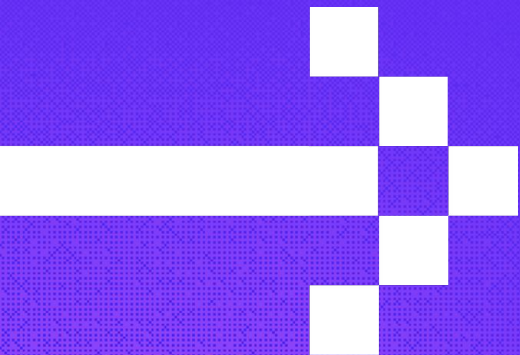


Iteratively grow mart models into data product models

- Model access set to “public”
- Enforce model contracts, use model versions, set deprecation dates
- Doc blocks for DRY and more expansive documentation
- Test data before it hits the data product model
- Add metadata
- One Big Table
- dbt grants to enforce data access policies
- Iceberg materialization if needed/possible

Score your data products on these standards!





Principle 3: Self-serve Data Platform



dbt Studio IDE

dbt Semantic Layer

dbt VSCode Extension

Cross Platform

dbt Canvas

dbt Mesh

dbt Copilot

dbt Catalog

dbt Learn

dbt Insights

Course Catalog

dbt Platform products that make your jobs easier



3 suggestions to make dbt project development easier

Pro-Tip #1: Create an internal dbt package

Pro-Tip #2: Create a template for new dbt projects

Pro-Tip #3: Create dbt projects specifically for learning



3 suggestions to make dbt project development easier

Pro-Tip #1: Create an internal dbt package

Pro-Tip #2: Create a template for new dbt projects

Pro-Tip #3: Create dbt projects specifically for learning

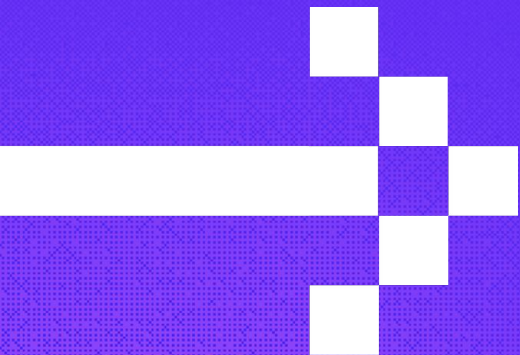


3 suggestions to make dbt project development easier

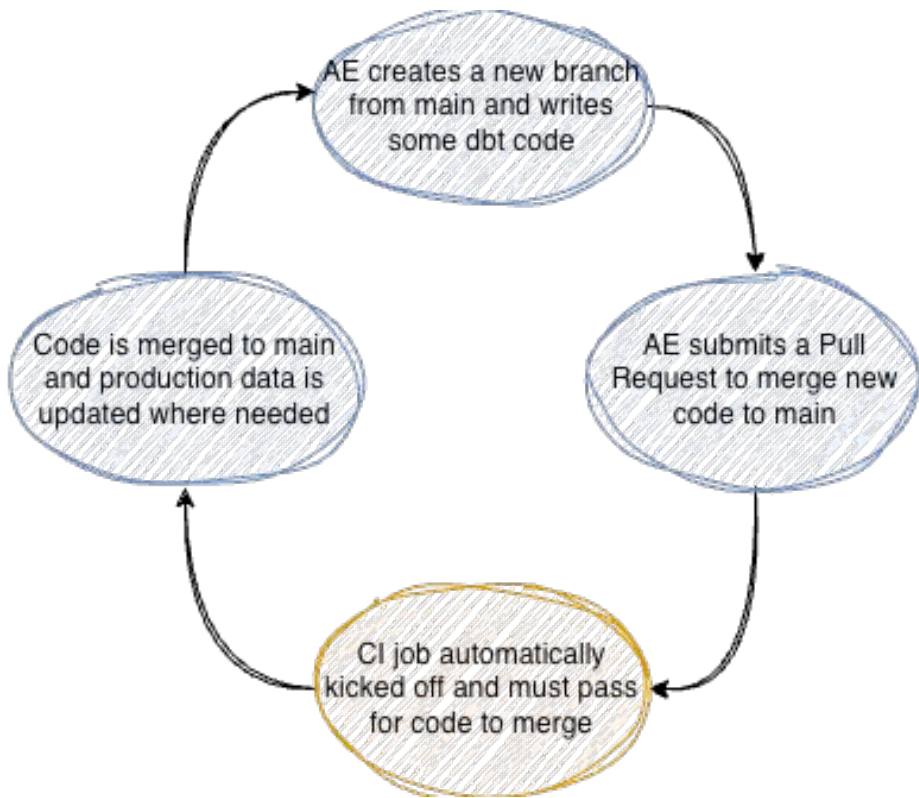
Pro-Tip #1: Create an internal dbt package

Pro-Tip #2: Create a template for new dbt projects

Pro-Tip #3: Create dbt projects specifically for learning



Principle 4: Federated Computational Governance



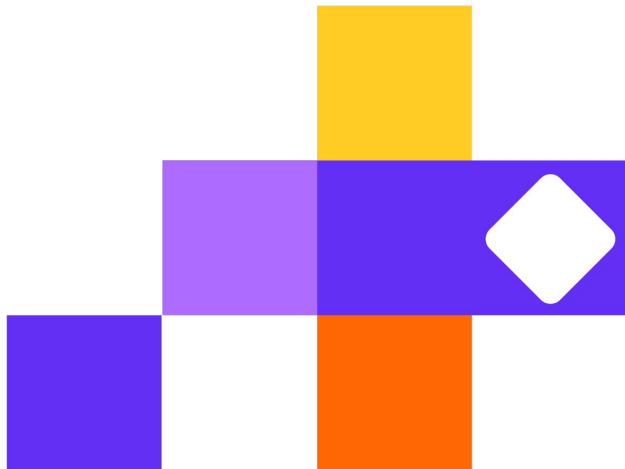
CI jobs are the leverage point in the CI/CD pipeline feedback loop

- Don't break prod by accident!
- Use the **dbt Project Evaluator** package to enforce project standards
- Build on the **Project Evaluator** models to create custom **mesh-wide** tests
- Turn your internal package into mesh monitoring project



People & Process

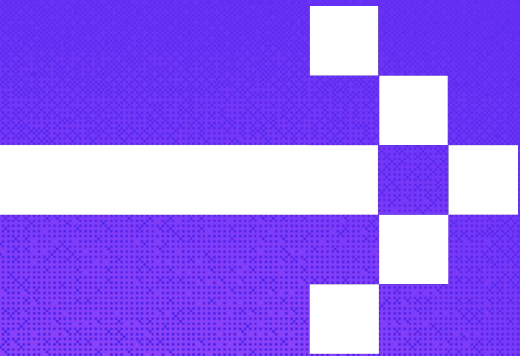
The squishy stuff





You have a lot of decisions to make! Let's review...

- Domain Ownership
- Data as a Product
- Self-serve Data Platform
- Federated Computational Governance



Convene your mesh –
and map out the
process of crafting new
policies



Have a question?

- Find the accompanying blog post at <https://jennajordan.me/>
- Find me in the dbt community slack: @Jenna Jordan
- Find me on social media
 - LinkedIn: /in/jennajordan1
 - Bluesky: @jennajordan.me

Continue conversations in chat
#coalesce-2025



Thank you

Please be sure to take the post-session survey

