



Four Data Architecture Mega-Patterns for Agility

Agenda

Four Data Architecture Mega-Patterns for Agility

1. **DataOps**
2. Data Fabric
3. Data Mesh
4. Functional Data Engineering

An Example that Combines all Four Patterns

Conclusion and More Information

DataOps

Data Fabric

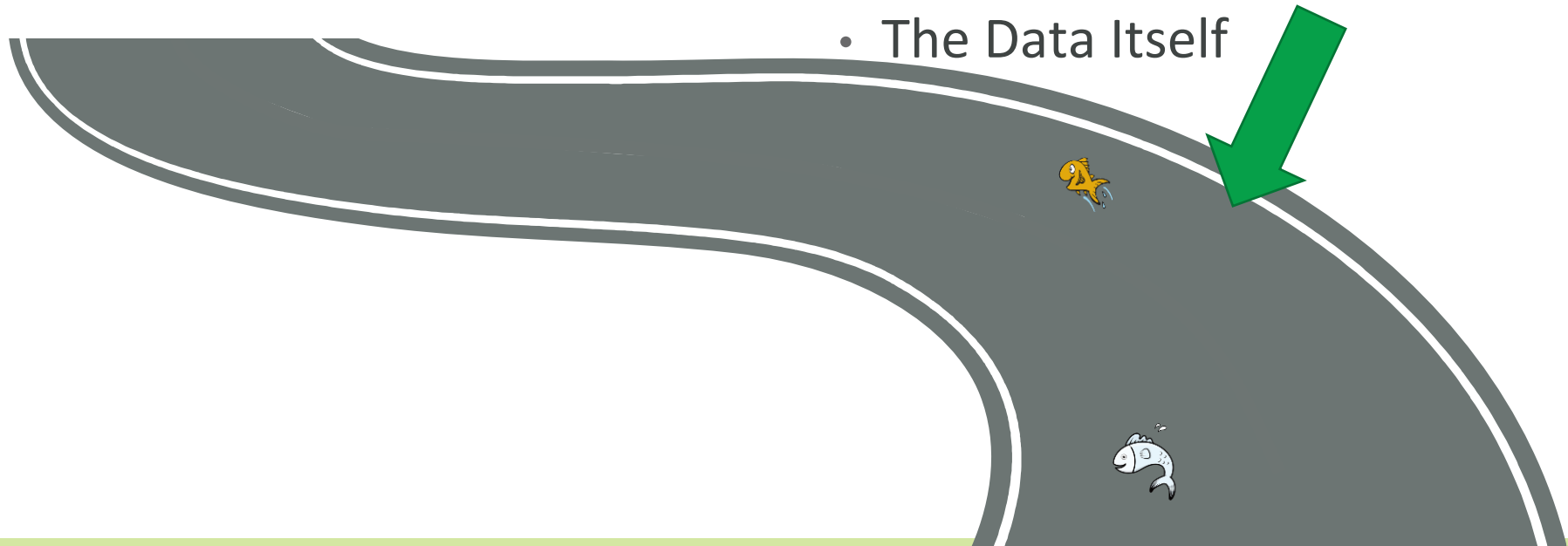
Data Mesh

Functional
Data
Engineering

Our Focus Is The River Of Work Right In Front Of

What is my next task?

- The Model,
- The Algorithm,
- The Data Pipeline,
- The Data Visualization,
- The Governance,
- The Data Itself



Next Task Focus Is Making Us Blind To Failure

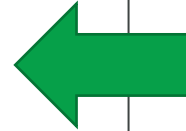
Task Focus Not Working

Data Analytic Projects



The Results are In...

- ❌ 60% fail altogether – Gartner
- ⚠️ 79% have too many errors – Eckerson
- 👋 87% of data science projects never get to production – Gartner



- The Model,
- The Algorithm,
- The Data Pipeline,
- The Data Visualization,
- The Governance,
- The Data Itself

Look **Upstream** At The Source Of The Problem



How You Do It

- Develop
- Deploy
- Iterate
- Monitor
- Test
- Collaborate



How? Focus On Four Key Upstream Processes



Lower Error Rates: Increasing Customer Data Trust



Decrease The Cycle Time: Continuously Deploy Innovation



Measure Your Team: And show everyone your success



Improve Collaboration: Less Meetings & Bureaucracy



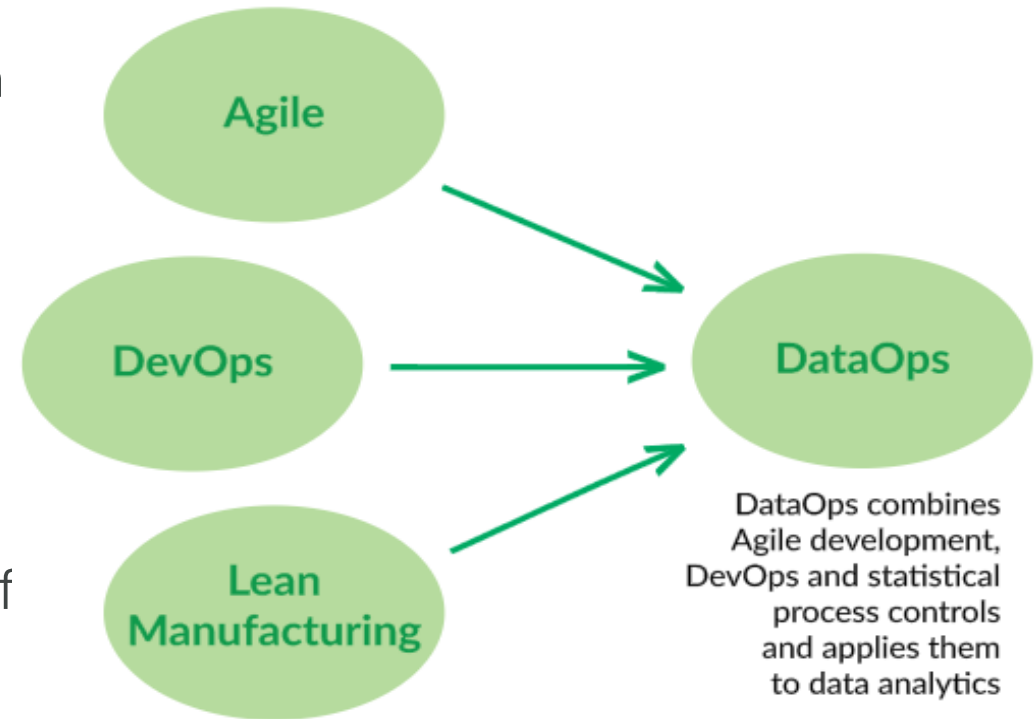
DataOps Aligns People, Processes, and Technology

Rapid experimentation and innovation enables faster delivery

Low error rates

Collaboration across complex sets of people, technology, and environments

Clear measurement and monitoring of results



Agenda

What Problems Do We Need To Solve With
Architecture for AI and Data Analytics?

Four Data Architecture Mega-Patterns for Agility

1. DataOps
- 2. Data Fabric**
3. Data Mesh
4. Functional Data Engineering

An Example that Combines all Four Patterns

Conclusion and More Information

DataOps

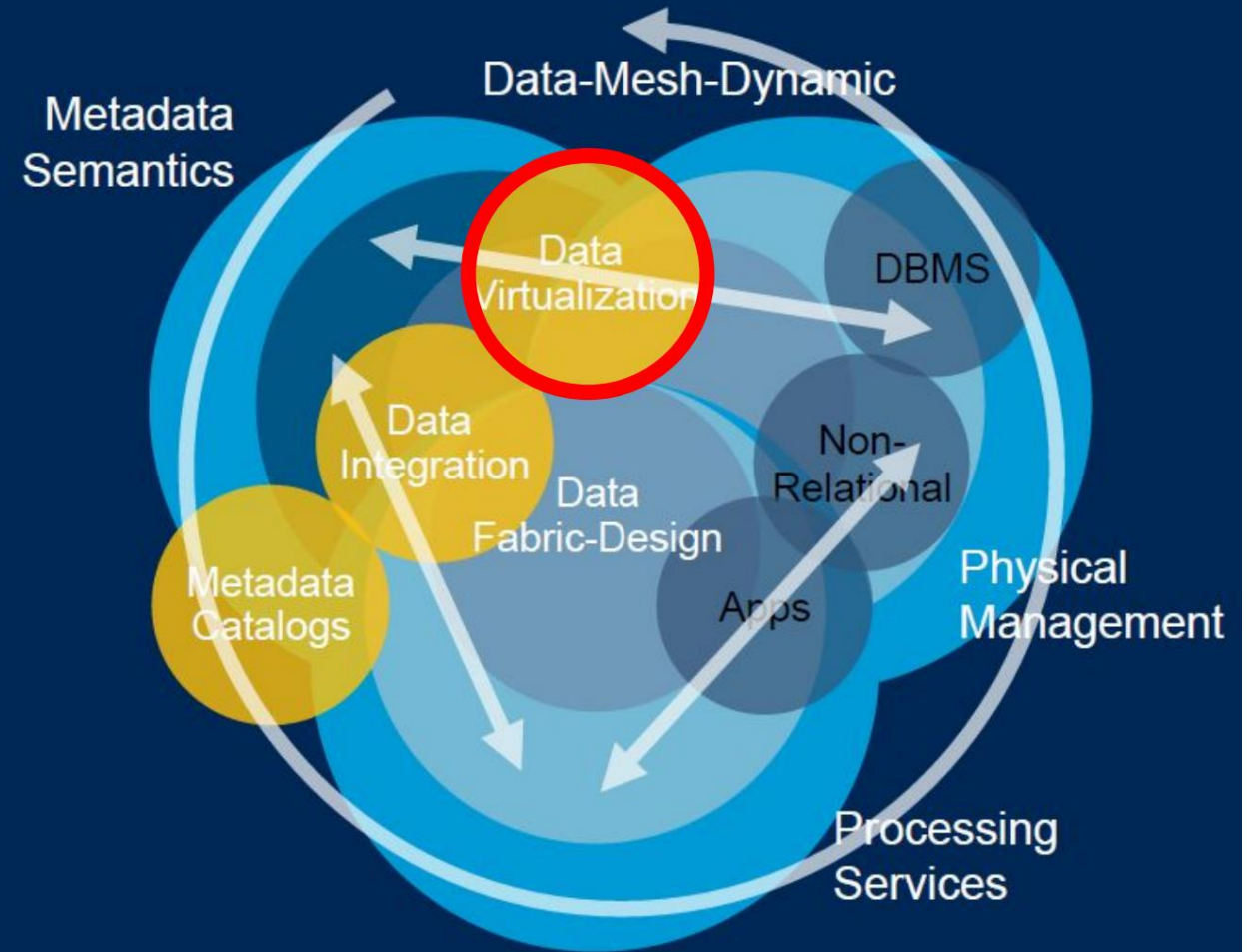
Data Fabric

Data Mesh

Functional
Data
Engineering

Trend No. 6: Data Fabric

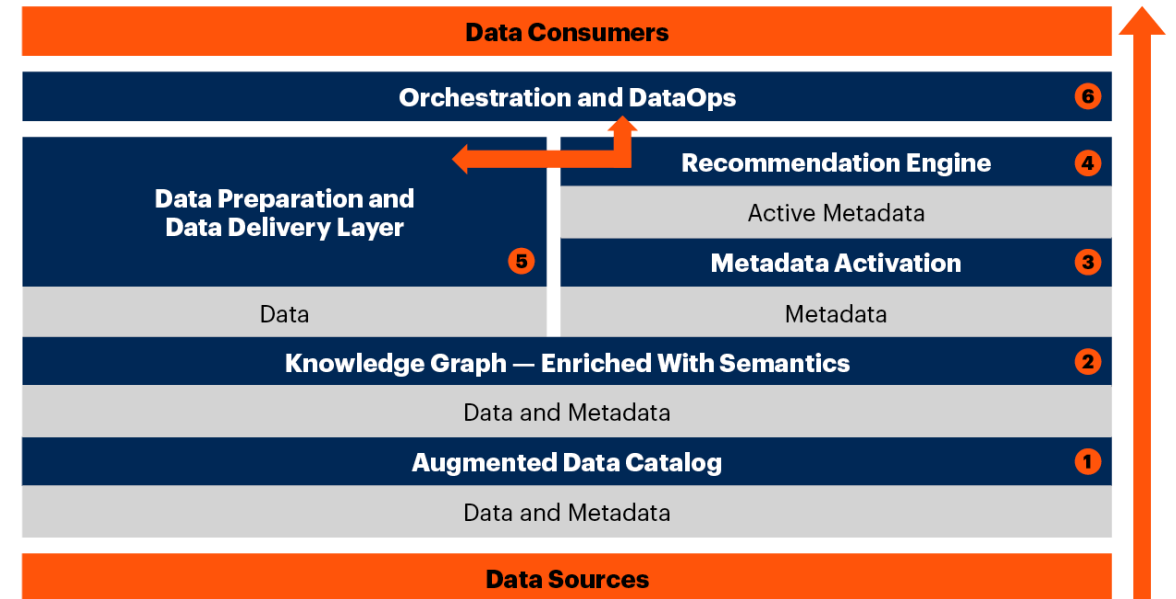
Through 2022, custom-made **data fabric** designs will be deployed as static infrastructure, forcing a new wave of cost to completely redesign for more dynamic approaches.



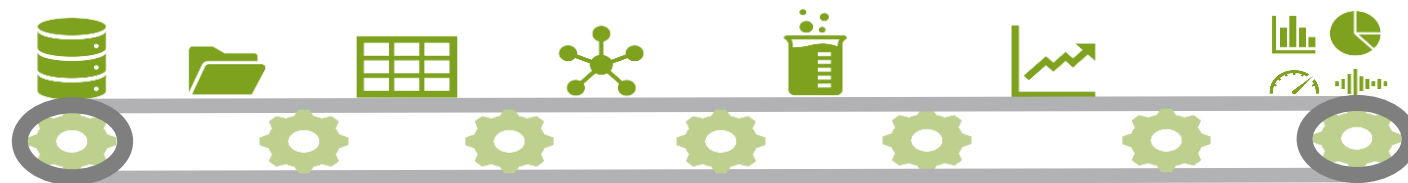
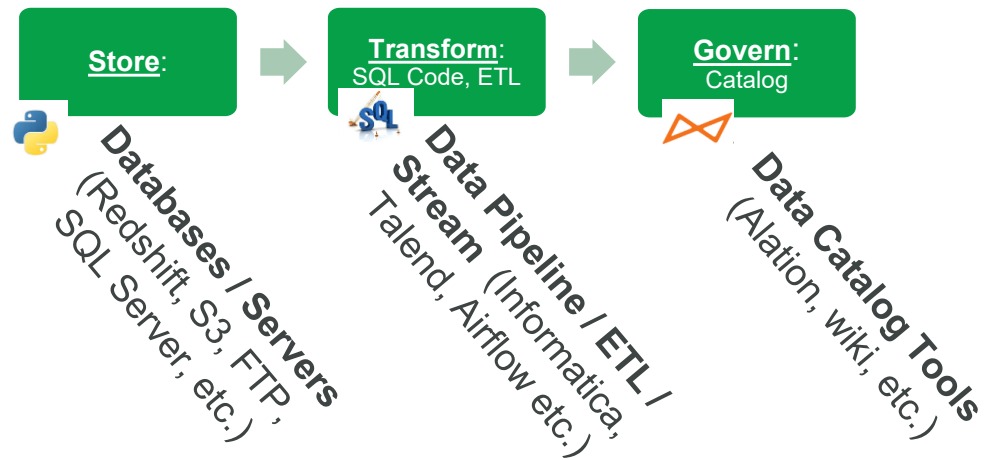
Gartner Data Fabric

“Data fabric focuses on composability, allowing users to build a flexible, agile, scalable architecture that will be able to supply data to humans or machine users.

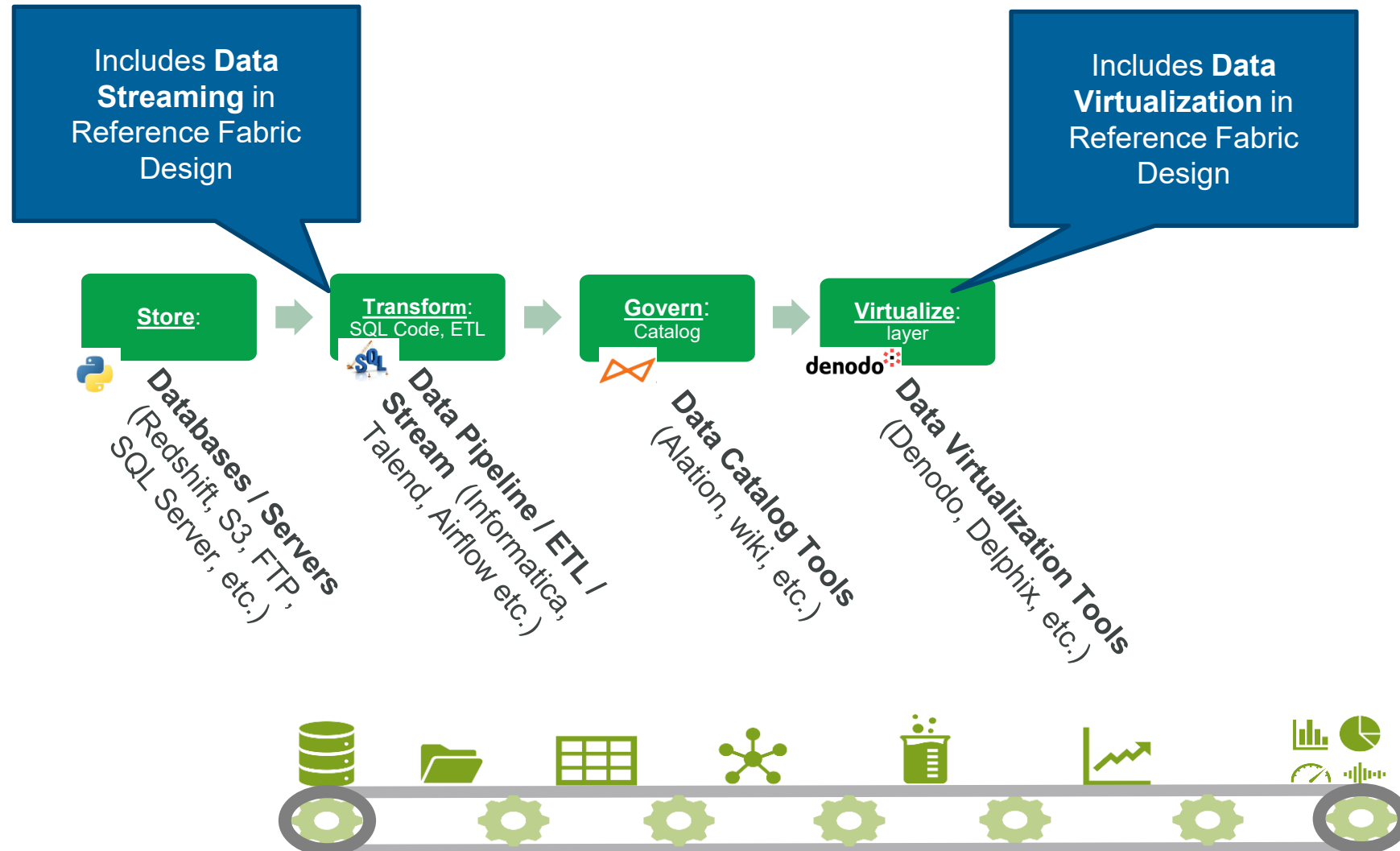
Data fabric is a design concept, not just a set of technology components. “



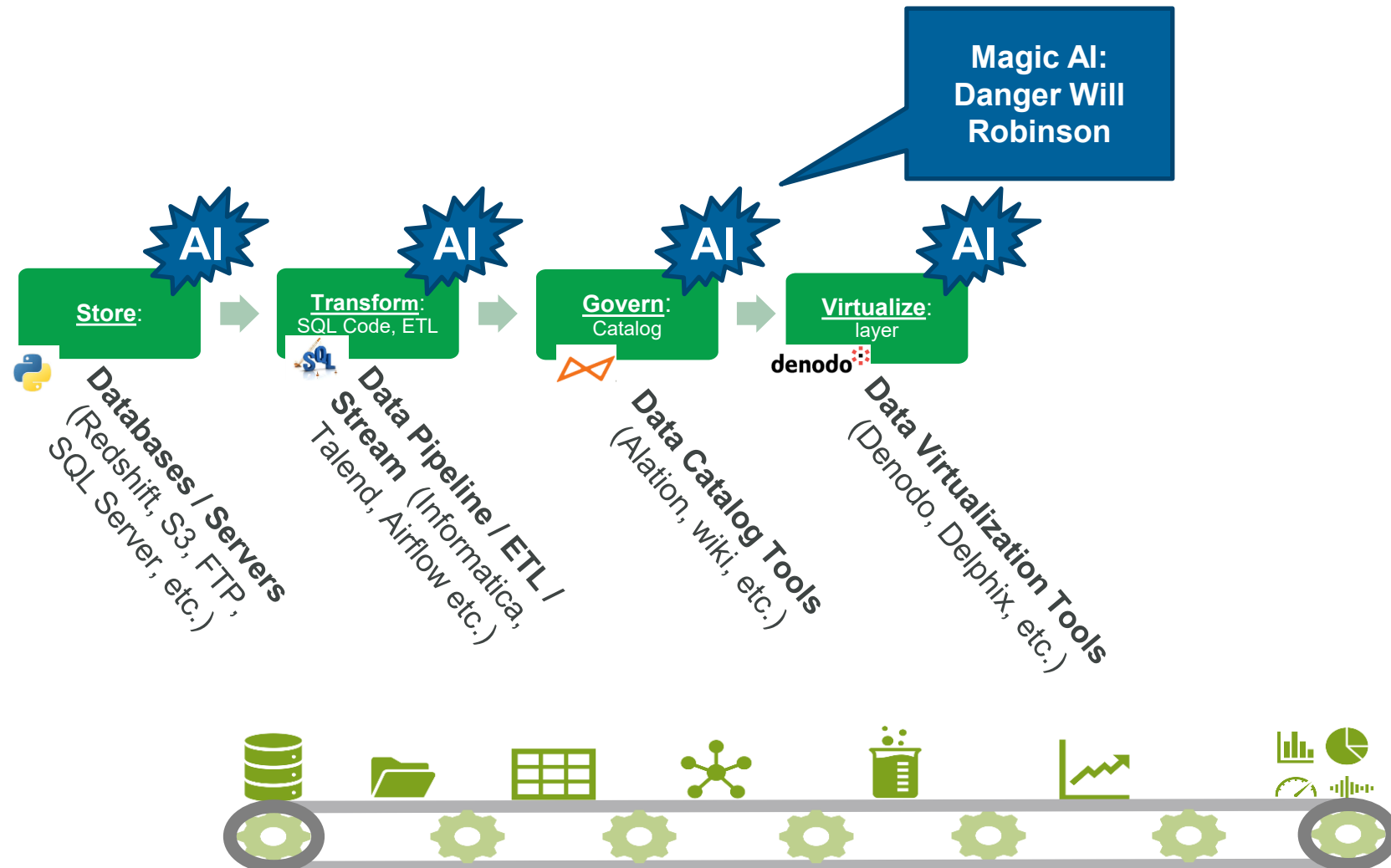
Data Fabric Toolchain Elements



Data Fabric Toolchain Elements



Data Fabric: Beware Magic of 'AI Inside'

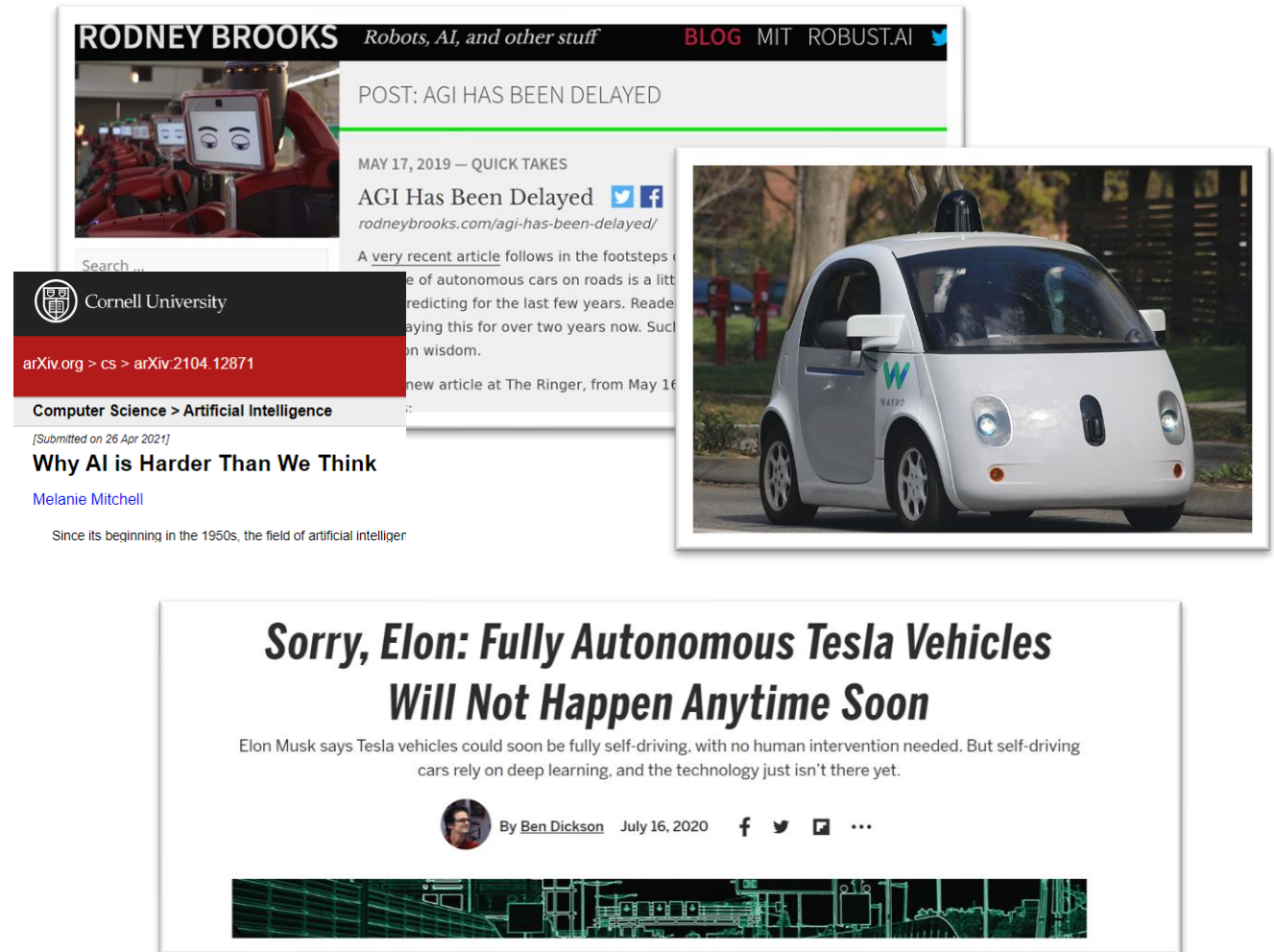


Data Fabric: Beware Magic of 'AI Inside'

Think of 'AI Inside' of Data Fabric like autonomous driving:

- **Level 1:** Simple, keep your hands on wheel
- **Level 5:** Cross Boston, in the snow, at night

We are at Level 1 of AI in the Data Fabric



AI + New Tools $\neq$ Agility

AI + New Tools $\neq$ Agility

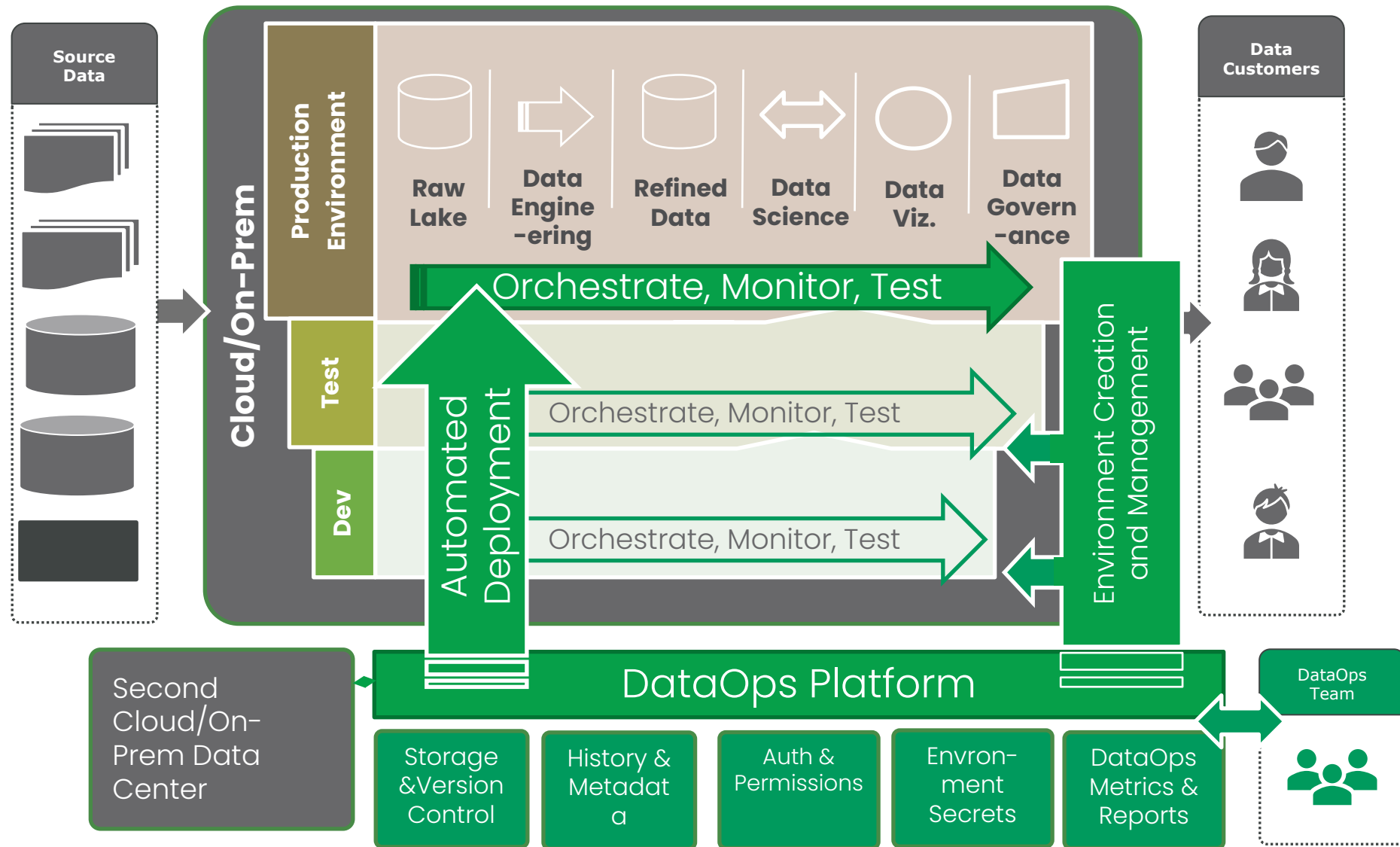


**People & Tools in a
DataOps
Architecture**

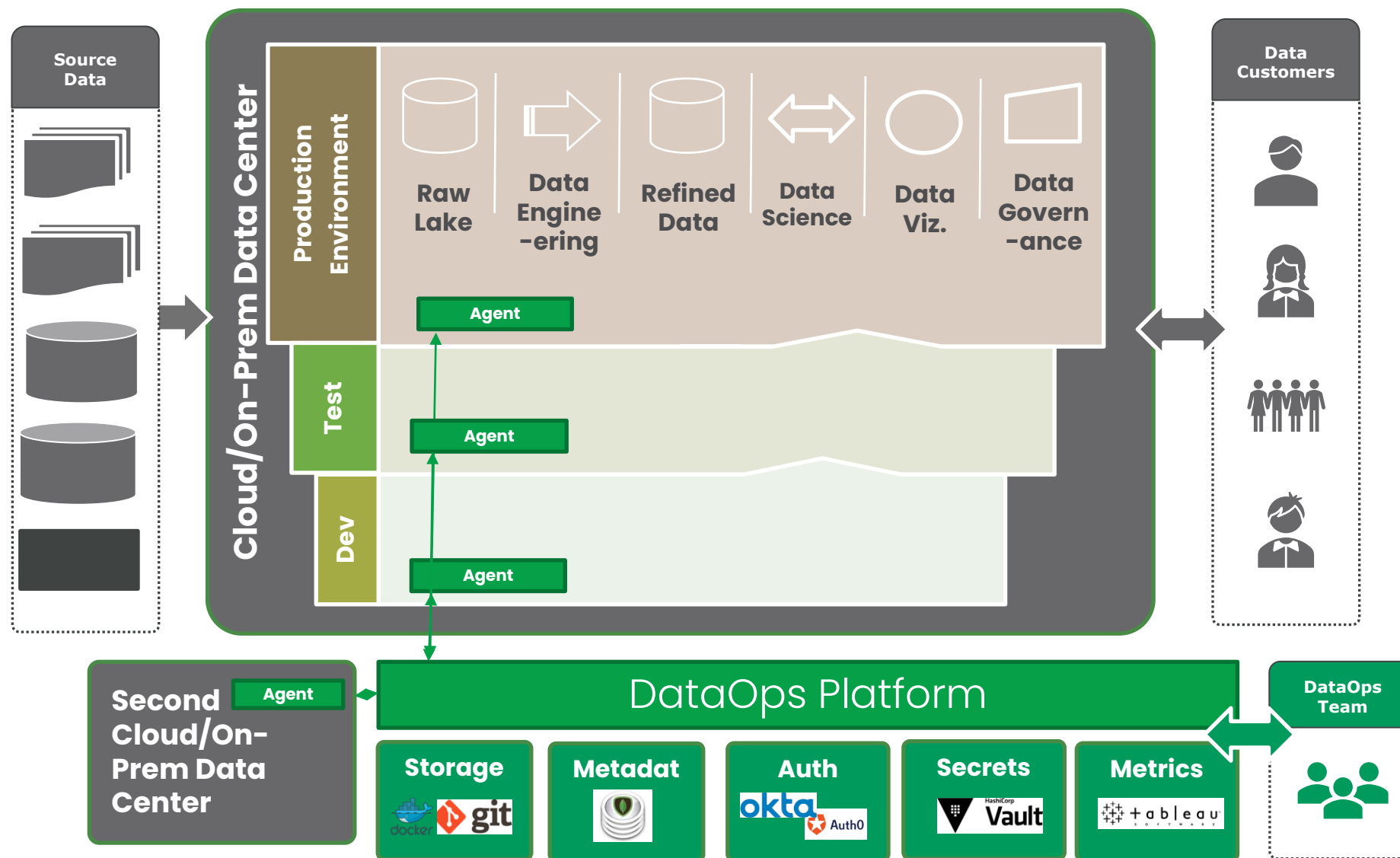
Canonical 'Factory' Data Architecture / Fabric



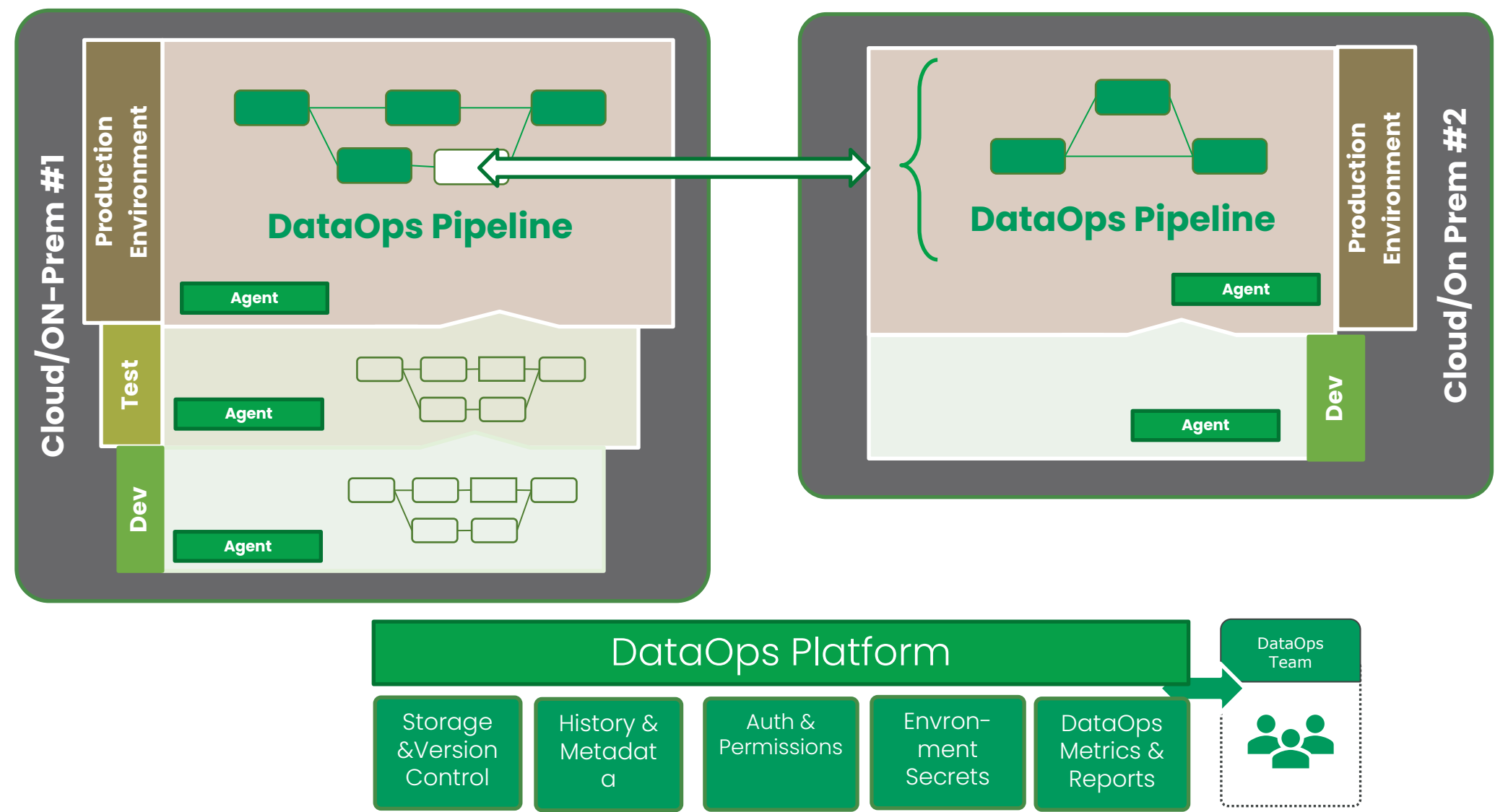
DataOps Functional Architecture



DataOps Physical Architecture



DataOps Spans Environments



Data Fabric – A New Fashion Trend?

- **It's Hot Stuff:**
Gartner View, Forrester View. Top 10 downloaded report 2020, top inquiry
- **What is a data fabric?:**
 - All the stuff you do with centralized data infrastructure:
ETL, DB, governance, store, lake, warehouse, stream/batch transformation.
 - Plus, some fancy new stuff
 1. AI component - magic pixie dust of self-driving data
 2. Data virtualization/semantic layer
 - **However**, it is missing other parts of the data value chain:
models, visualizations, self service. It's more 'hub' than 'spoke'
- **Why?** Moniker that covers the latest trends in data management.
- **Caveat:** The goal of implementing a data fabric is agility - agility is a second-order effect from better tools. The primary driver is people & process following DataOps.

Agenda

Four Data Architecture Mega-Patterns for Agility

1. DataOps
2. Data Fabric
- 3. Data Mesh**
4. Functional Data Engineering

An Example that Combines all Four Patterns

Conclusion and More Information

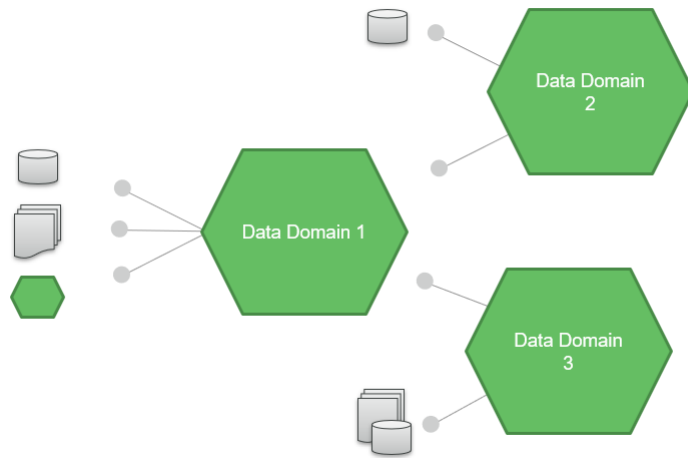
DataOps

Data Fabric

Data Mesh

Functional
Data
Engineering

Data Mesh 101



Why Data Mesh?

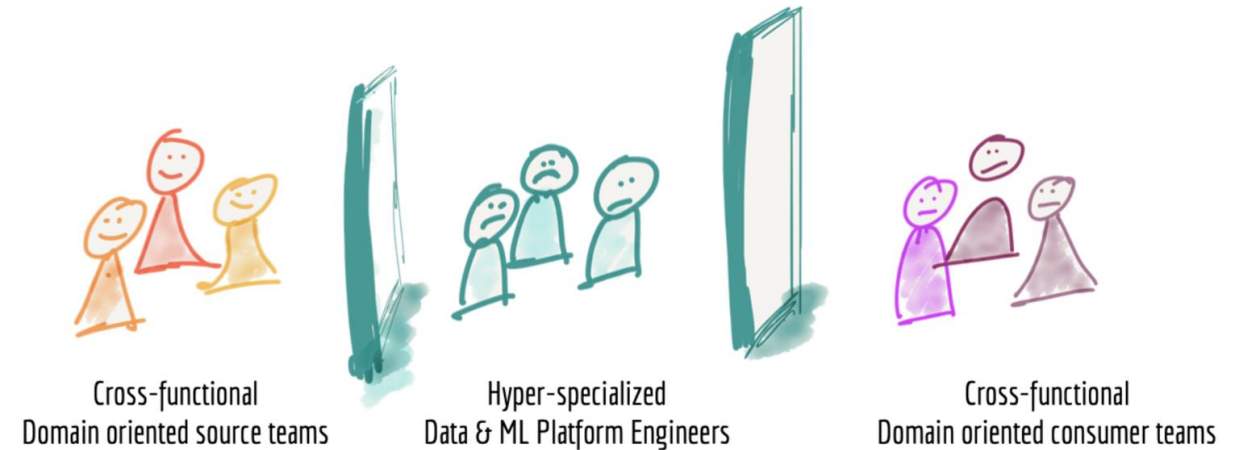
- Centralized Systems Fail
- Skill-based roles are unable to respond to rapid customer needs
- Data domain knowledge matters
- Universal, one size fits all patterns fail
- General Data Analytic Project Failure
- Inspired by domain driven design (DDD) in software

**The main idea is to take a best practice from developing software & apply them to data analytics.
(Sound familiar?)**

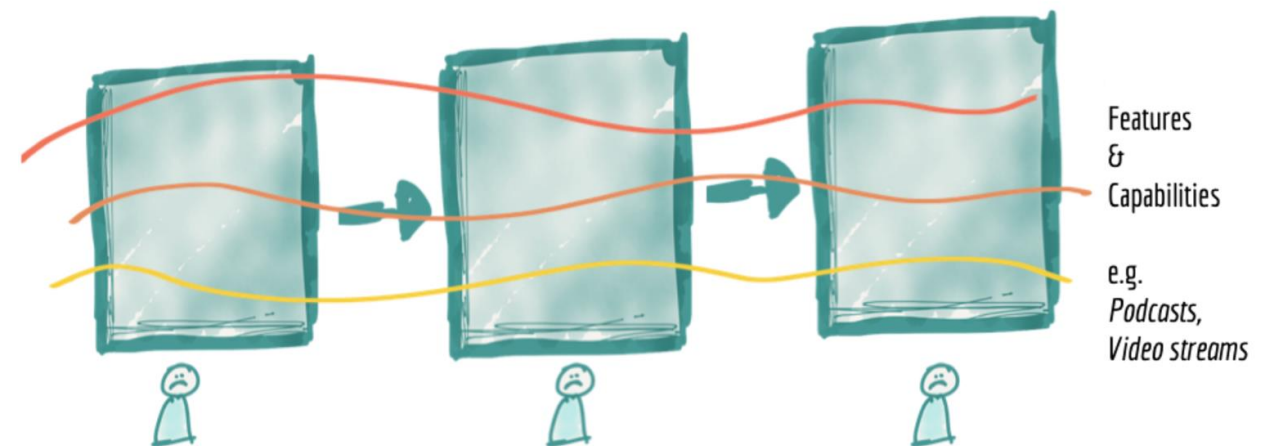
The Human Side of a Data Mesh: Main Idea

The organization creates walls & changes need to cross the traditional organizational boundaries

- The organization structure builds walls & barriers to the changes



- When you make a change, you need to update each component & coordinate between several different teams



No, Data Engineers Are Not Perfectly Fungible



Data Mesh = Organization Mesh

The use of domain-driven / data mesh design as the primary means:

1. Assignment of full end-to-end ownership of a domain to **one cross-functional team that gets the necessary support to fulfil that responsibility.**
2. Structure data
3. Build composable systems

Data Organization Keys

Let the small team continually own the data set & not move for project to project is key

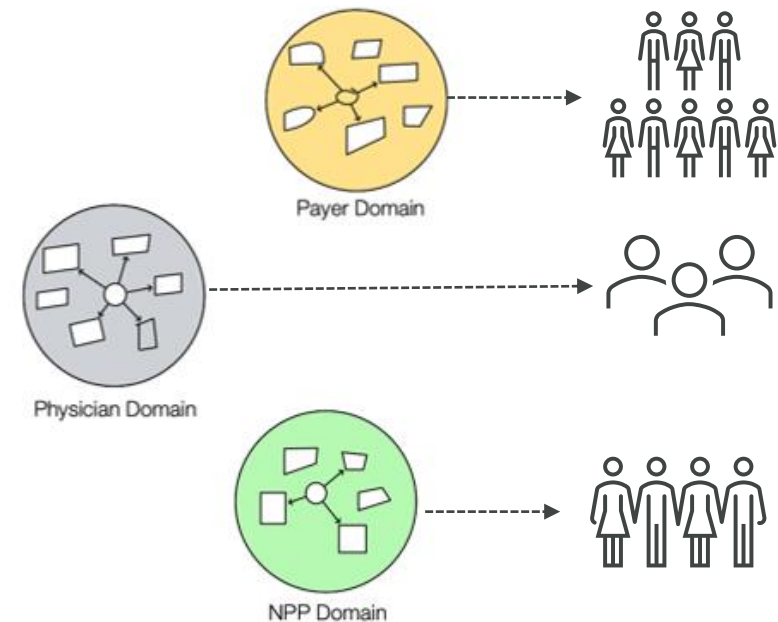
‘You own the product’ thinking provides the right incentives between the producers & consumers

Source: thoughtworks.com/insights/blog/data-mesh-its-not-about-tech-its-about-ownership-and-communication

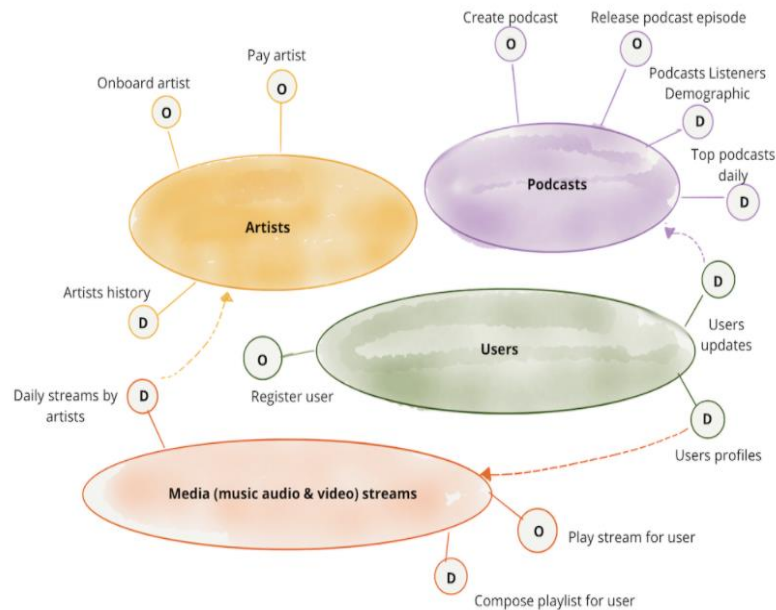
The Human Side of a Data Mesh: Main Idea

The organization creates walls & changes need to cross the traditional organizational boundaries

- Take the ideas of microservices where a team owns the dev, test, deploy & running of the microservice (5-9 people)
- Organize around the domain, not the technology
- The Operational & Data products are created by the same team
- Domain data as a product - domain data teams must consider their data assets & artifacts as their products & others as their customers
- Data Engineers must live, work & understand a finite number of data sets to really add value



What Data is in a Domain?



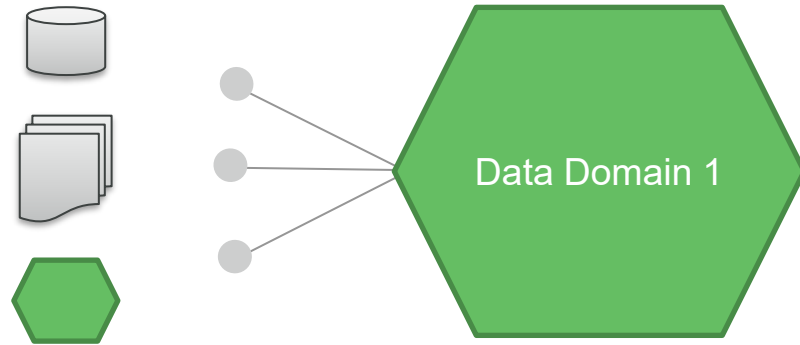
Domains Aligned with Sources / Types of Data

- 'Mastered' Data:
 - Entities of business / subject areas
 - Customers, products, etc.
- 'Sources' of Data:
 - Business reality: facts on the ground
 - Weblogs, user interaction history

Domains Aligned with Consumption of Data

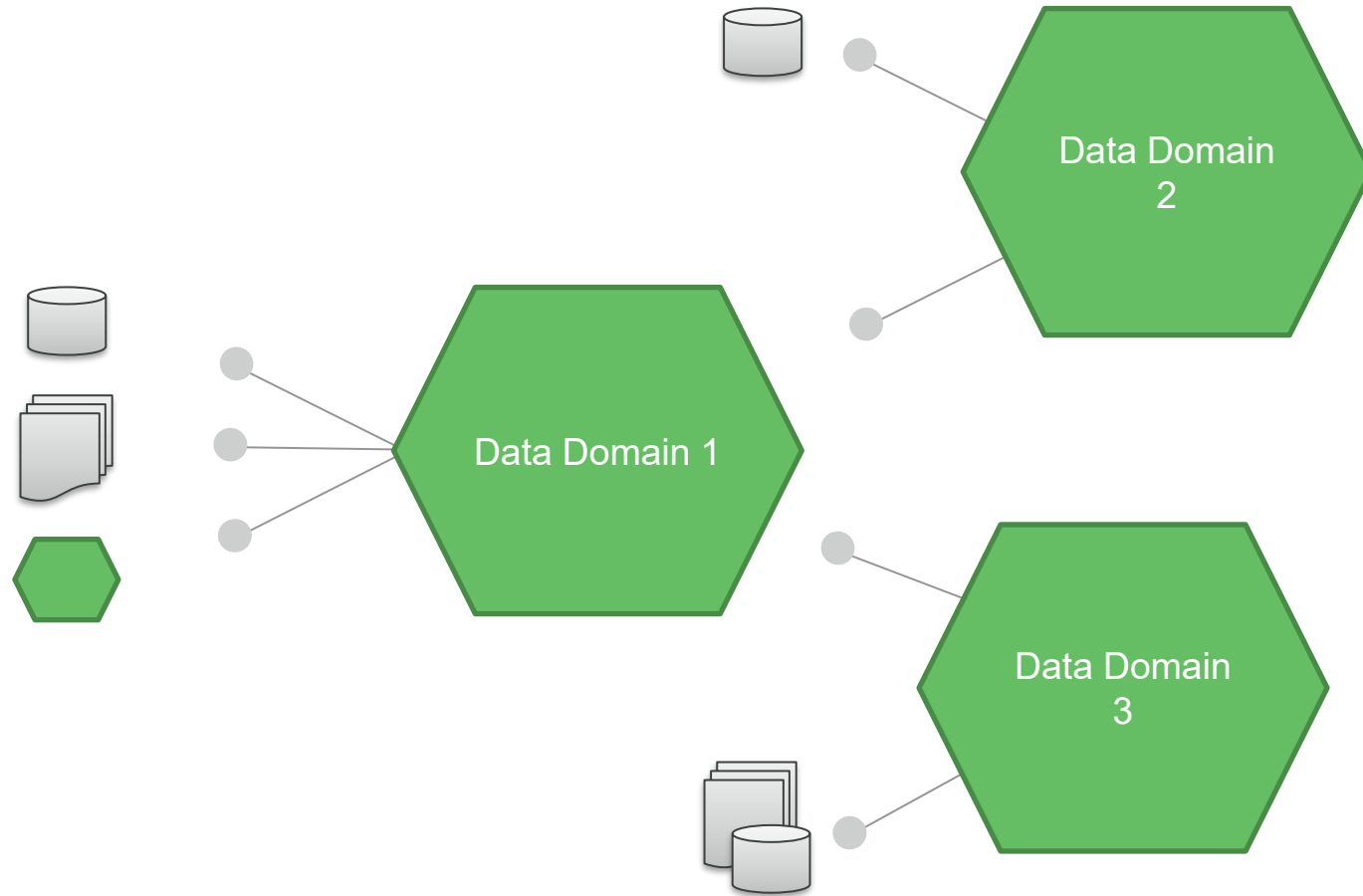
- Integrated Data / Ready for Consumption
 - Facts / Dimensions / Star Schemas
 - Aggregated Views
- Product View
 - Never Done, Always Improving
 - Customer Usage Focus

What are the Domain's Components?

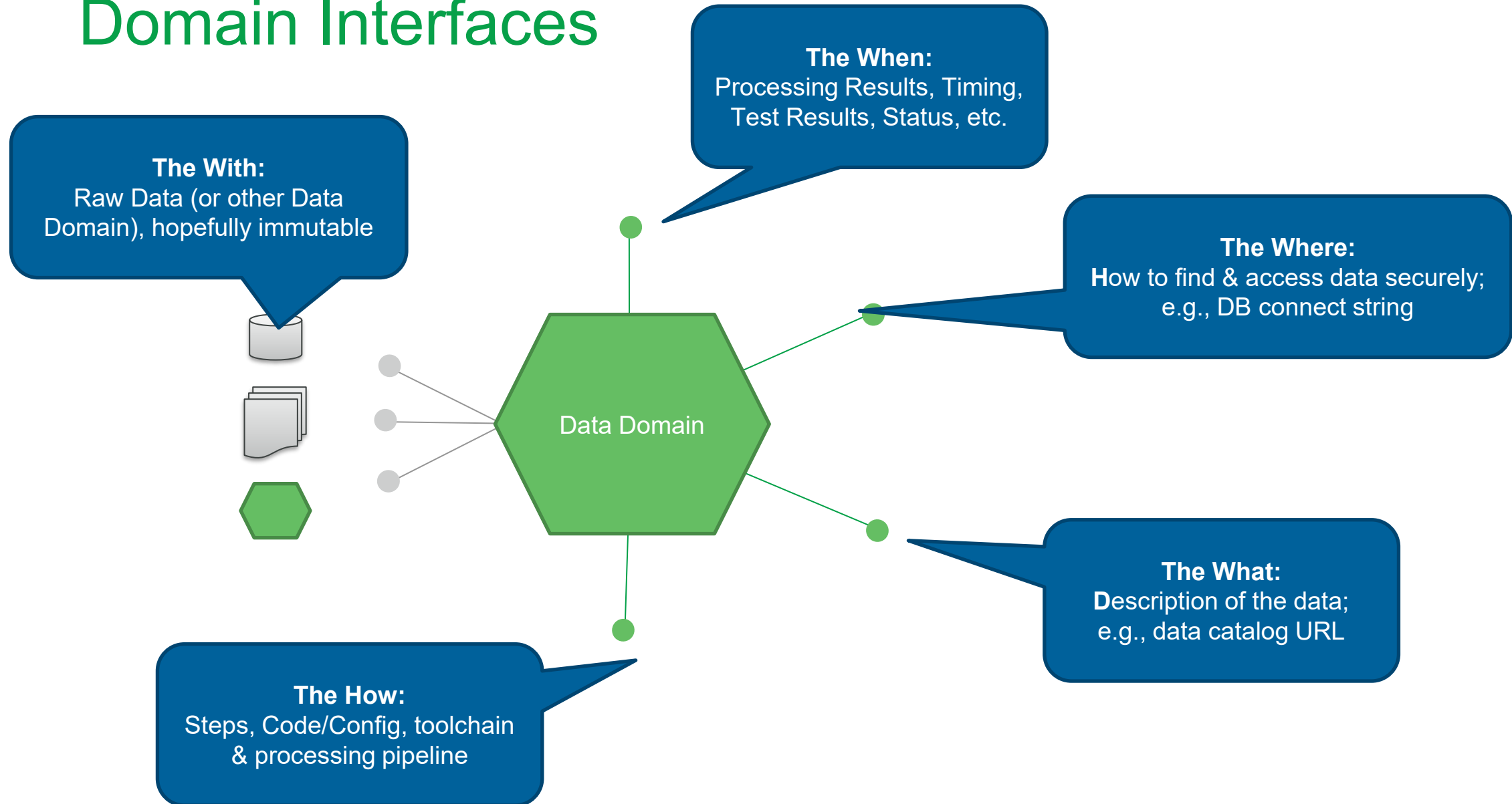


1. Data
2. Artifacts created from that data: models, views, reports, dashboards, etc.
3. Code that acts upon that data: pipelines, toolchains, etc.
4. Team used to create/update/run that Domain
5. Metadata: catalogs, lineage, test results, processing history, etc.

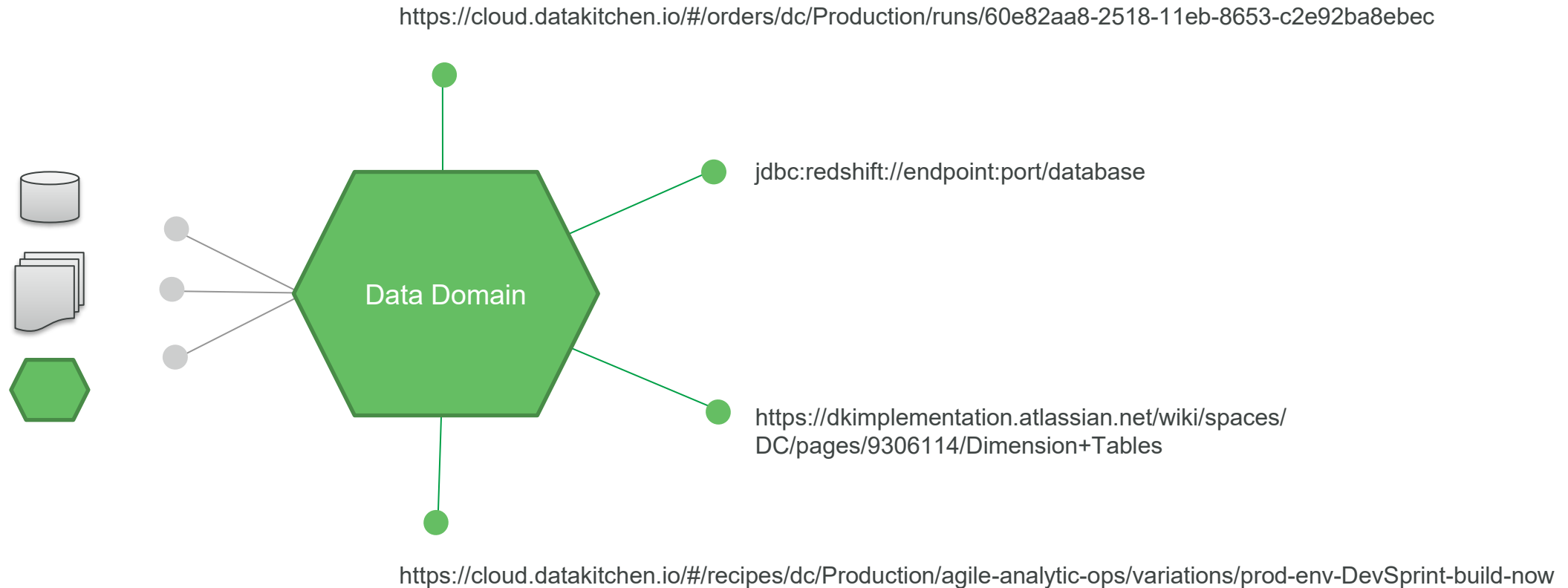
Domain Must Be Composable & Controllable



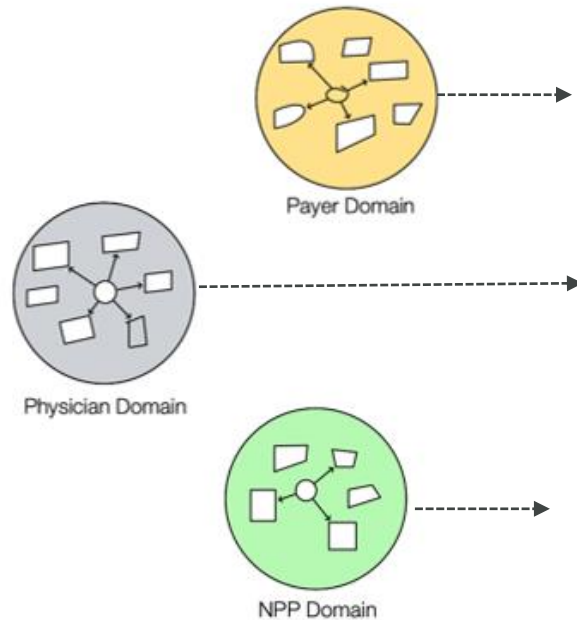
Domain Interfaces



Domain Interfaces as URLs



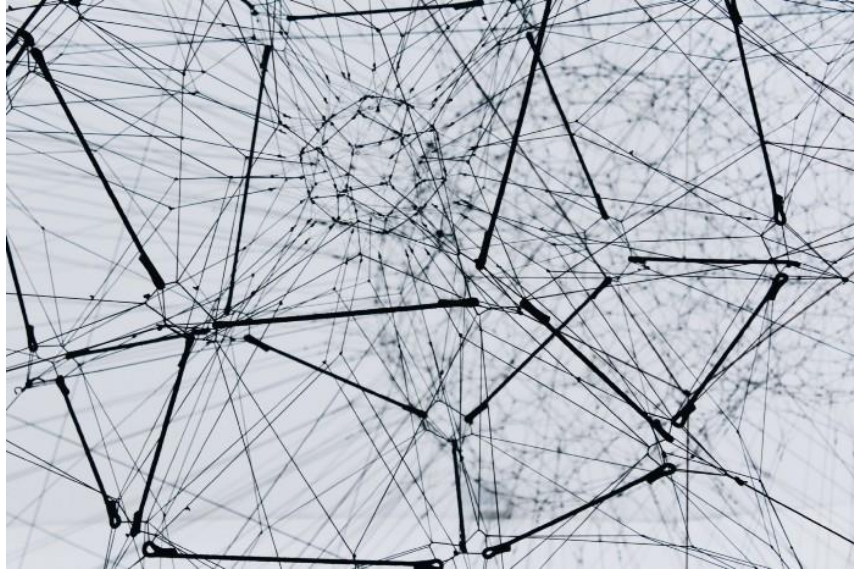
What Do You Want Out of a Domain?



A series of independent domains of data that are:

1. Trusted
2. Usable by the teams' customer
3. Discoverable / Findable
4. Understandable & well-described
5. Secure & permissioned
6. URL/API Driven: & can inter-operate with other domains
7. Have 'single throat to choke' for the customer to easily:
 - Report problem & get updates on fixes
 - Ask for new insights / improvements & get them into production quickly

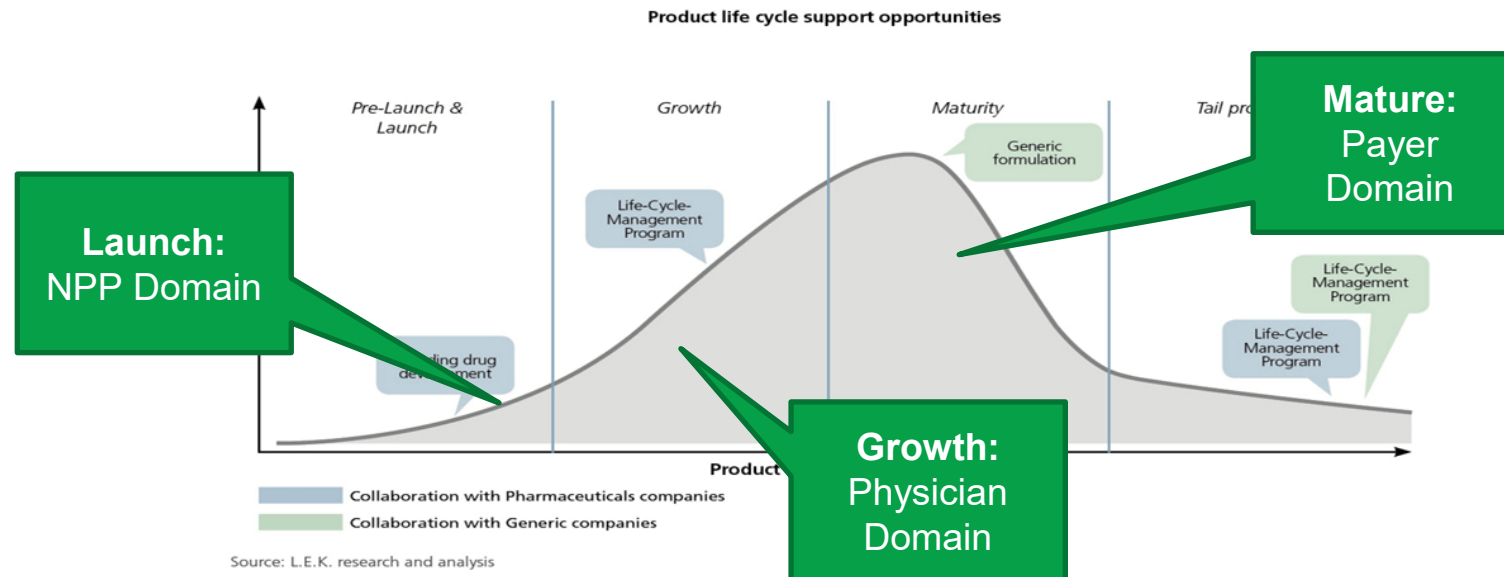
Data Mesh Change in Focus



1. **Domains** & the grouping of your work into small teams & partitions over 'one platform to rule them all'
2. What **services** you are providing your customer, rather than what data you are loading
3. **Discovering** & using over extracting & loading
4. **Decentralization** & the freedom to innovate over central control
5. **Ecosystem** of data products linked together over a centralized lake / warehouse

An Example of Domains

Commercial Pharma Analytics



US Commercial Pharma Domains

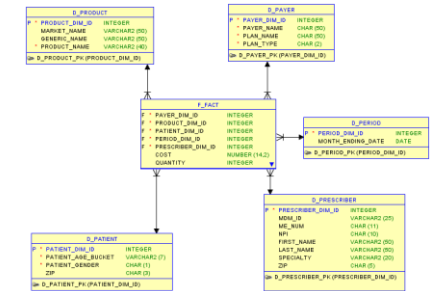
- **NPP** (Non-Personal Promotion): emails, web site visits, even radio ads
- **Physician**: doctor (& other outlets) sales, claims data, anonymized patient data
- **Payer**: Payer/Plan, rebates, formulary

What About the Data?

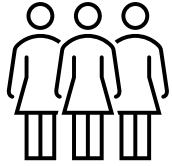
Sub-national Weekly data
Sub-national Payer Data
Sub-national Institutional (DDD) Data
National Prescription Audit Data
Sales Force Alignment Data
Longitudinal Patient Data
Sub-national Profit and Loss Data
Sub-national Claims and Co-pay Data
Payer and Plan Formulary Data
Census Data
Stocking Data
Source of Business
AMA Data
Retail OTC Data
Buy and Bill Data
Field Calls and Promotional Activity Data
Rep Expenses and Vacancy Data
Hotline Verification Data
Contract and Payer Rebates Data
Veeva CRM Data
ERP Data
NPP Data
Forecast Data
Primary Research Data

What about the data in each domain?

- Each domain has separate data sources
- Overlapping entities (e.g., physicians) exist in each domain
- Each domain has different cycle times of product (i.e., daily, weekly, hourly, etc.)
- Each data domain has its unique characteristics.
 - For instance, subnational physician data from IQVIA - purchased by pharma companies - may not 1:1 match claims data, which may not match payer data. This is due to data supplier issues & timing projection algorithms.

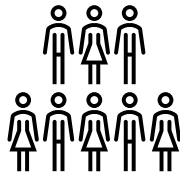


Pharma Sales & Marketing Teams



NPP Domain Marketing & Sales Team

One part of the pharma brand team focused on ads, digital & other non-personal promotions. This team matters most pre-launch & during the growth phase of a product



Physician Domain Marketing & Sales Team

Another part of the pharma team focused on in-person sales. Those are the good-looking people you see in doctors waiting rooms. Sales calls, samples, doctor visits, messages, call alignments, etc. This team matters the most during the first years of a pharma launch.



Payer Domain Marketing & Sales Team

A third part is focused on Payer Marketing. This part is - in essence - controlling the price of a pharmaceutical product due to the rebate given to any payer. They are concerned about the rebate contract, being on formulary & tier & copays. Payer Marketing matters more during the 'mature' phase of a pharma product lifecycle.

Domain Layers

Mastered Data Sets
(IT)

Integrated Data Sets
(Data Engineers)

Self Service Tools
(Analyst)

1. **Mastering & small files foundation files are a domain layer**

There are 1M physicians in the US, but the company master of physicians is only 40K. This work is done by separate teams working independently.

2. **Of course, the main data warehouse is a domain layer**

There are facts & dimensions, along with multiple tables used for specific analysts needed.

3. **Self/Service & Data Science are a domain layers**

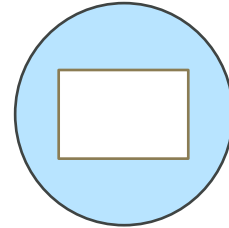
They can keep their owned cached data sets (e.g., tableau extract) or have their own small data sets that they mix with the central data in Alteryx (or other) tools. Data Science teams have their own segmentation models dependent on specific views or extracts of data.

Domain Layers

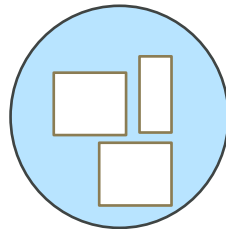
Raw, Sourced Data (Various)

Sub-national Weekly data
Sub-national Payer Data
Sub-national Institutional (DDD) Data
National Prescription Audit Data
Sales Force Alignment Data
Longitudinal Patient Data
Sub-national Profit and Loss Data
Sub-national Claims and Co-pay Data
Payer and Plan Formulary Data
Census Data
Stocking Data
Source of Business
AMA Data
Retail OTC Data
Buy and Bill Data
Field Calls and Promotional Activity Data
Rep Expenses and Vacancy Data
Hotline Verification Data
Contract and Payer Rebates Data
Veeva CRM Data
ERP Data
NPP Data
Forecast Data
Primary Research Data

Mastered Data Sets (IT)

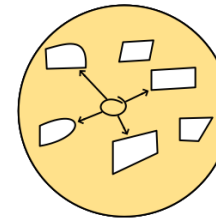


Mastering Domain:
Physician MDM

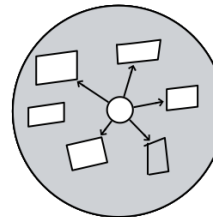


Mastering Domain:
Target Lists, Product
Market Baskets

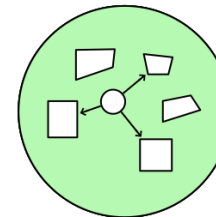
Integrated Data Sets (Data Engineers)



Payer Domain

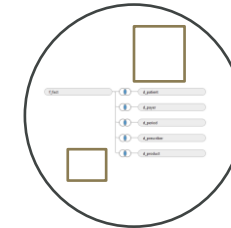


Physician Domain

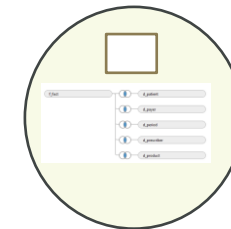


NPP Domain

Self Service Tools (Analyst)



Brand Team
Reporting Domain

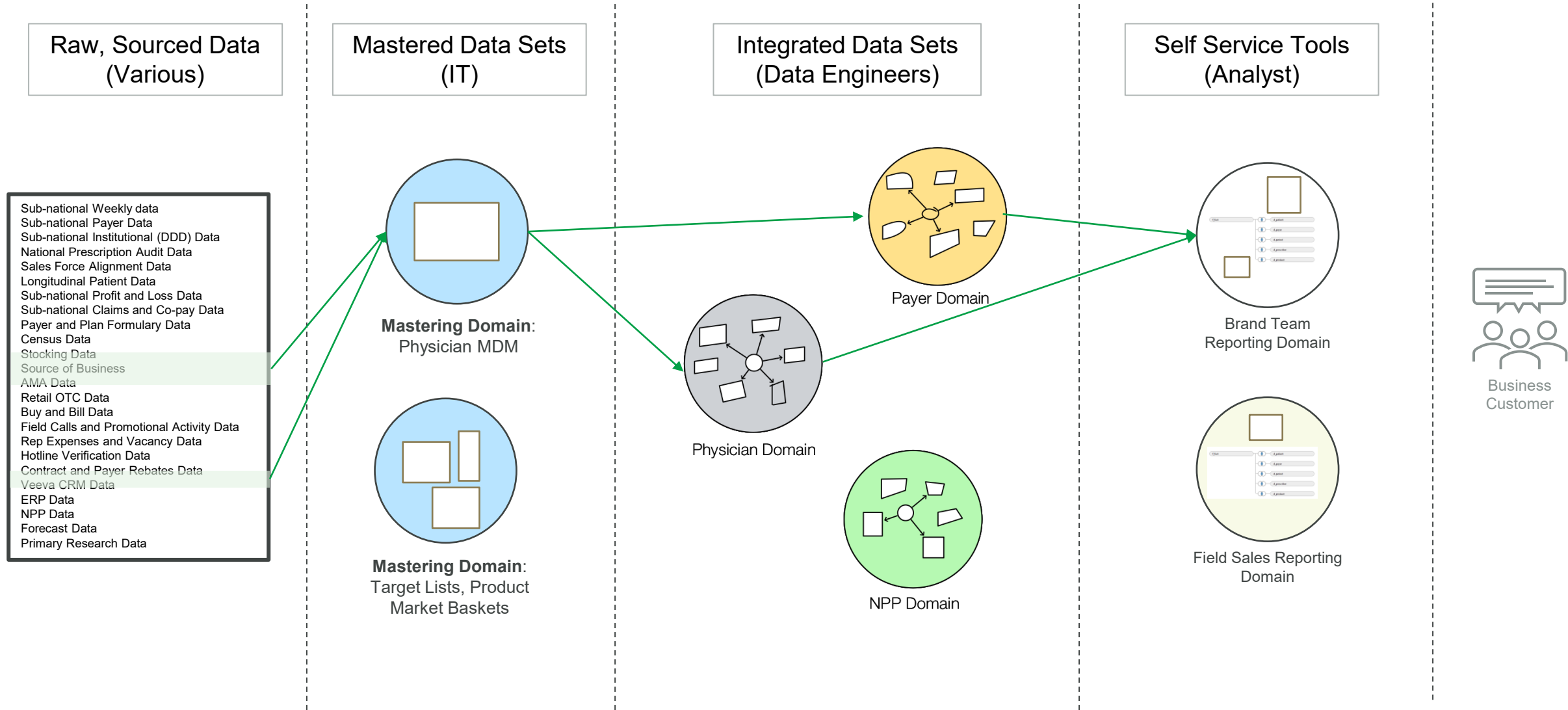


Field Sales Reporting
Domain

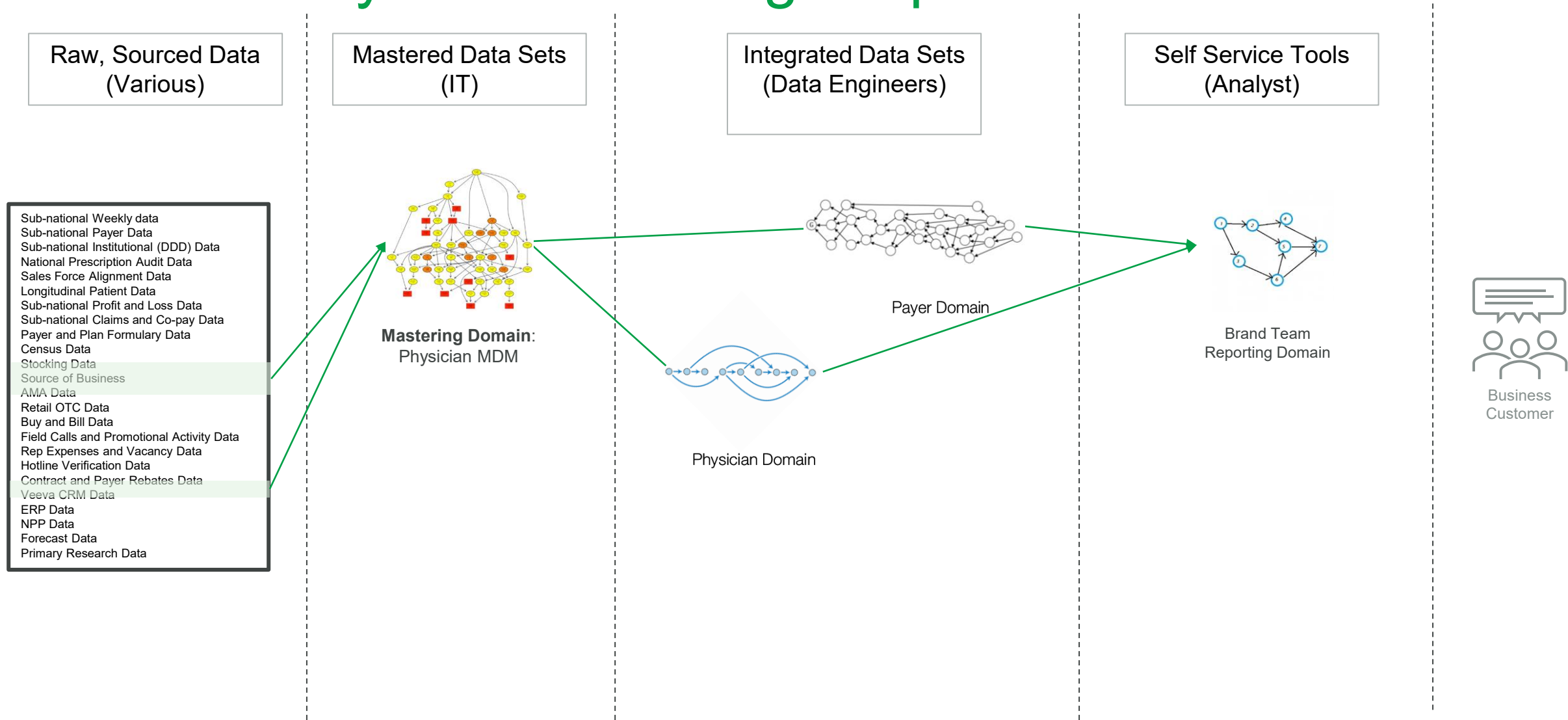


Business
Customer

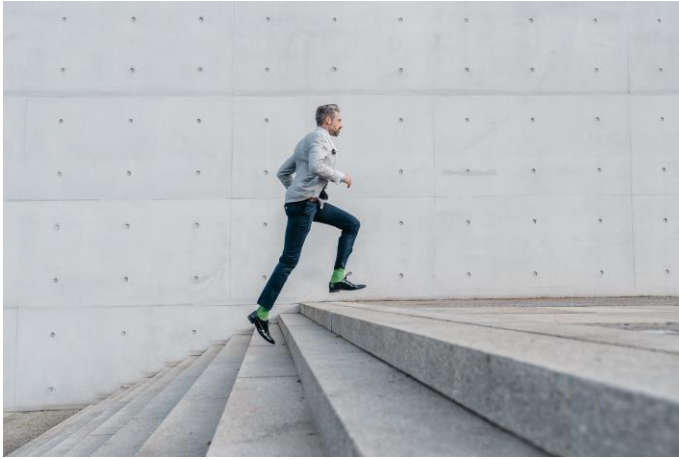
Domain Layers Processing Relationships



Domain Layers Processing Steps



Benefits of Approach



- Yes, you can do all these four Data Architecture Mega-Patterns for Agility!
- Benefits
 - Support over \$10 Billion in sales
 - Integrated 100s of data sets
 - Very, very few errors or missed SLAs
 - > 50,000 automated tests
 - > 100 of schema/data changes per week
 - Staff of seven data and DataOps engineers
 - Low total yearly costs
hardware/hosting/software/staffing
- DataKitchen software enables those four patterns: Recipes, Tests, Kitchens and Especially Ingredients can handle all the needs

Agenda

Four Data Architecture Mega-Patterns for Agility

1. DataOps
2. Data Fabric
3. Data Mesh

4. Functional Data Engineering

An Example that Combines all Four Patterns

Conclusion and More Information

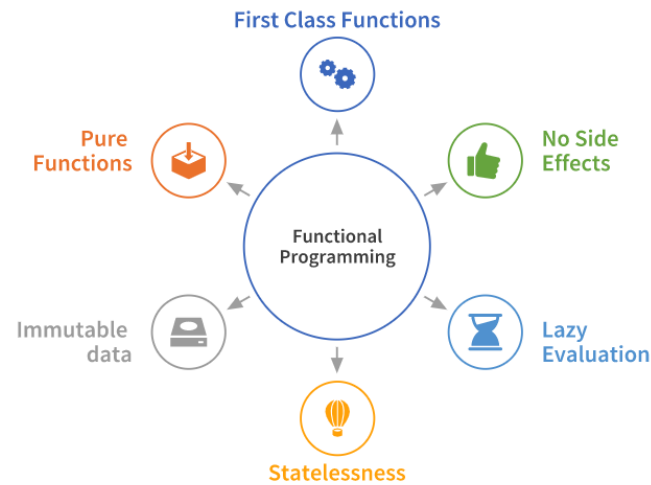
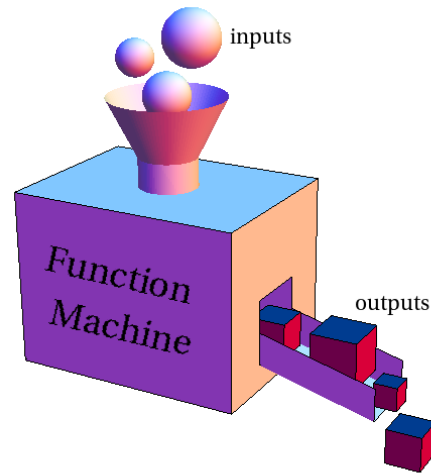
DataOps

Data Fabric

Data Mesh

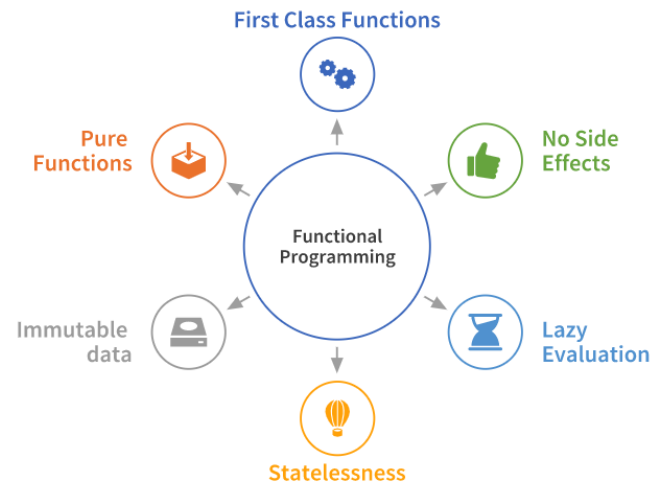
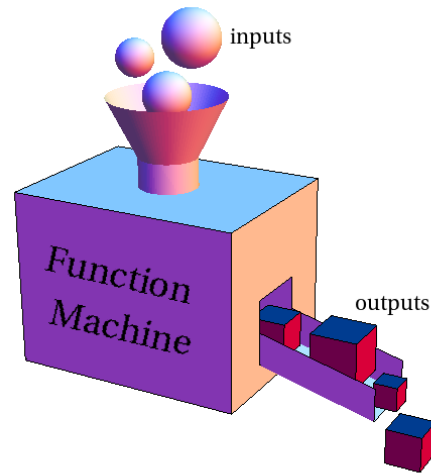
Functional
Data
Engineering

Built With Functional Programming



- Start with immutable (never changing) data
- Pure functions (you put some data in & get some data out)
- Idempotency (you can run it over again & get the same thing)
- No side effects

Functional Approach Benefits



Reproducibility

- Foundational to the scientific method and data science / AI
- Critical from a legal standpoint and sanity standpoint

Complexity Reduction

Cloud Native

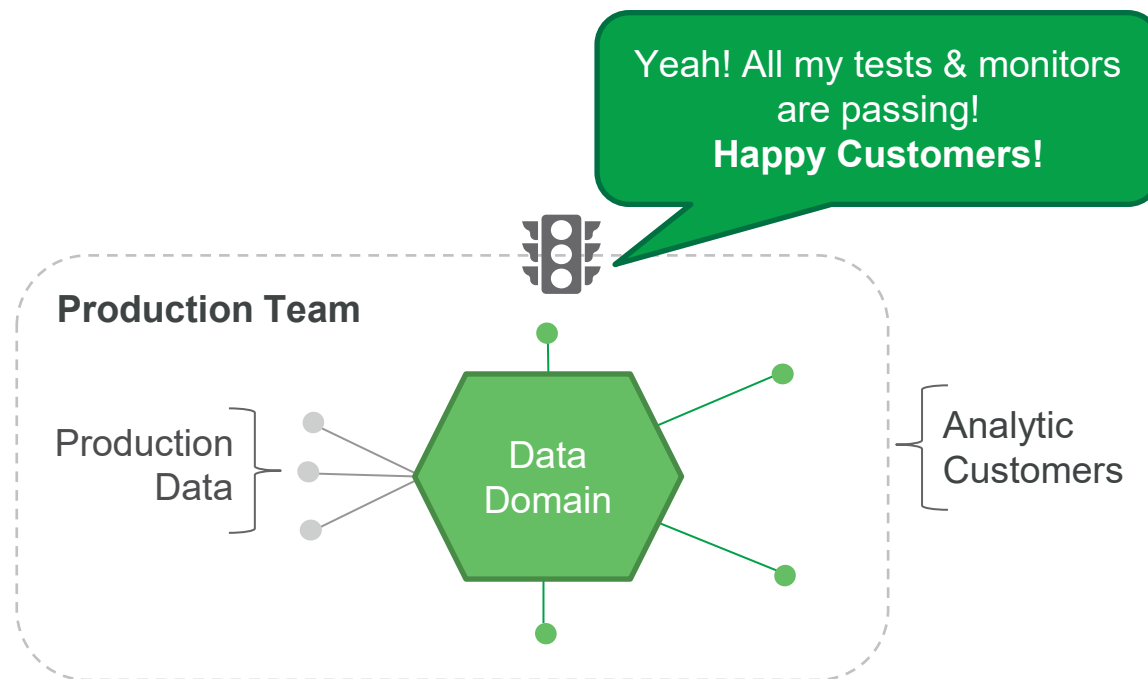
- Storage and compute are cheap

Faster Time To Value

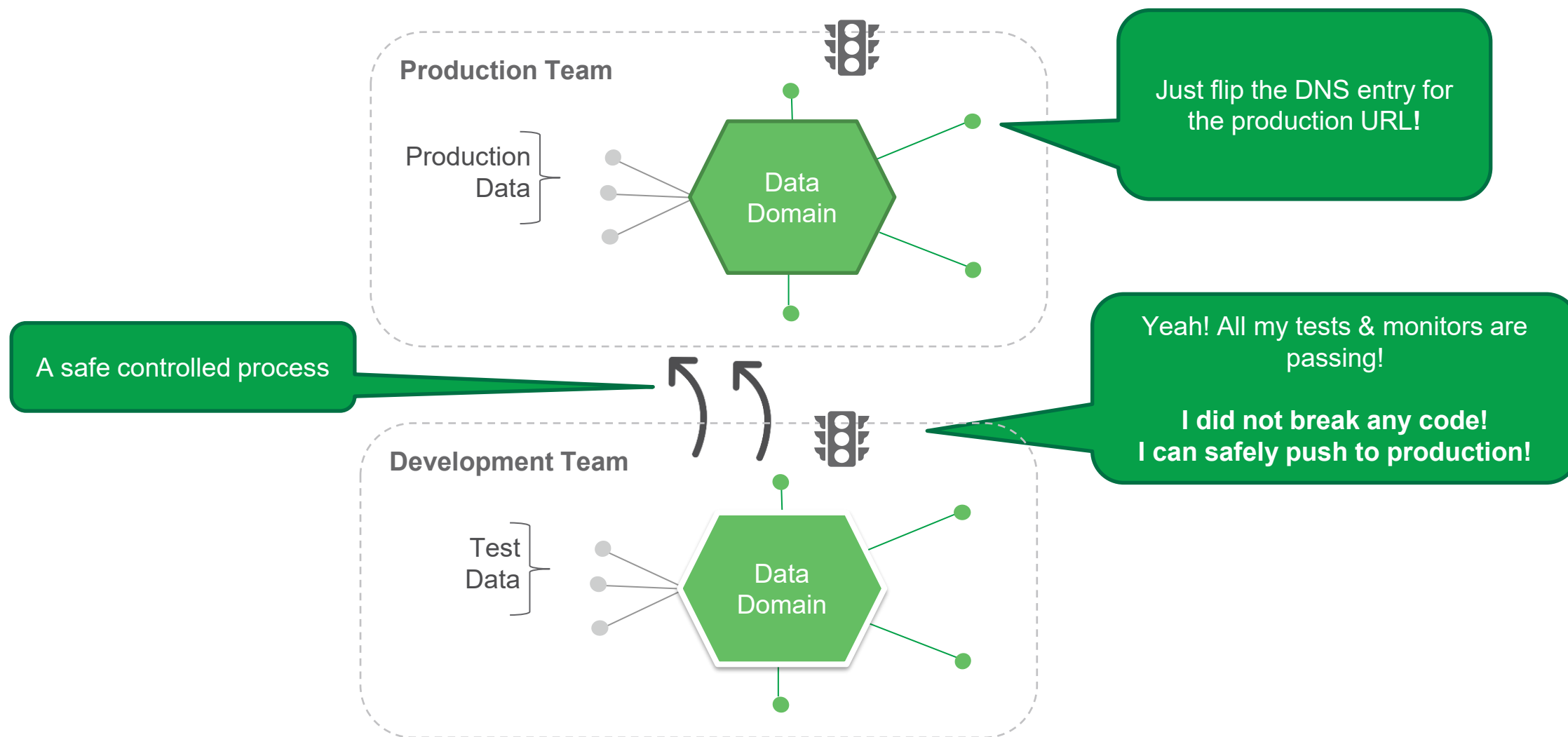
Functional Data Mesh Systems

Think of all your data & analytic work as a “Big Function” in domain

- In that function are your data & AI toolchain
- Everybody works that function (whether they know it or not!)
- Re-running a task for the same date should always produce same output
- Data can be repaired by rerunning the new code
- A ‘big red/green light’ on the system telling you everything is OK



Functional Data Systems Are Easier to Test & Deploy



Agenda

Four Data Architecture Mega-Patterns for Agility

1. DataOps
2. Data Fabric
3. Data Mesh
4. Functional Data Engineering

An Example that Combines all Four Patterns

Why DataKitchen supports these four patterns easily!

DataOps

Data Fabric

Data Mesh

Functional
Data
Engineering

Domain Layers Processing Relationships

How do we update the data?

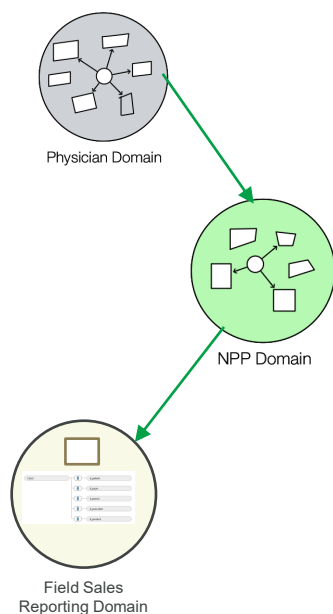
- Each Domain layer its own domain update processing
- Each layer has their own toolchain (i.e., SQL, Python, Informatica, etc.)
- Each layer has a series of sub-steps (i.e., a 'DAG')
- Each layer wants to know if the build is completed, the test applied & if the data is data is correct

What causes the update of each domain?

- Time / Schedule
- Order of operations, a meta-orchestrated coupling of each Domain, one part may need to be done before the other or after.
- Event-orchestrated coupling. When new data arrives, kick off a change.

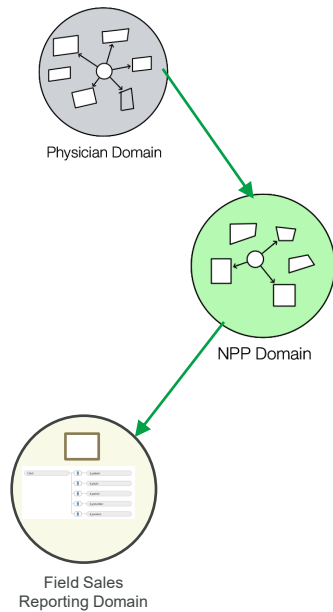
You Need a 'Master DAG' to run them all

Inter-Domain Communication Links



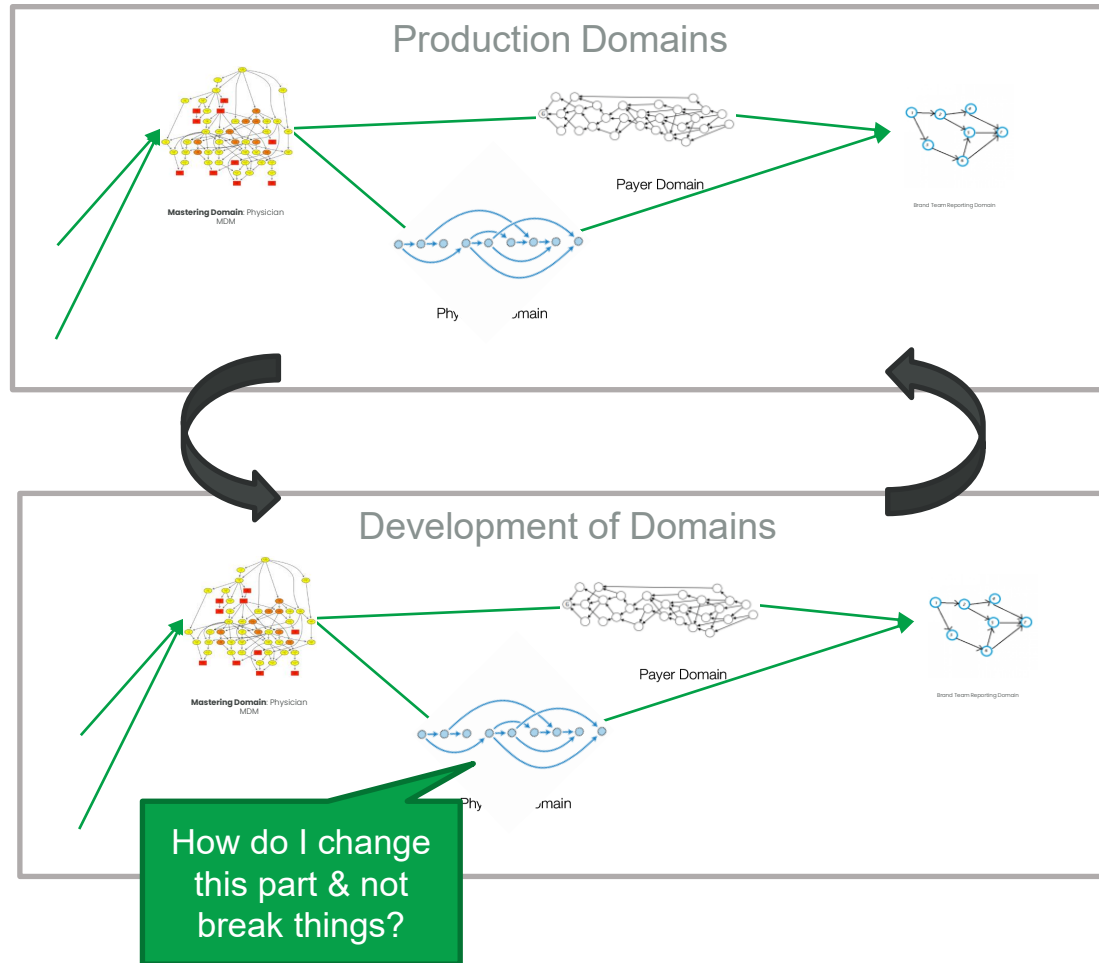
Inter-Domain Communication		Question / Steps Asked
	Domain Query	"When was the last time you were updated?" Successful or failure? Warnings?
	Domain Query	"Is the data or artifacts in your domain good?" Can you prove it with some test results?"
	Process Linkage	"Ok, you start. I am done."
	Process Linkage	"Ok, you start. I am done & here are a bunch of parameters you need to keep going."
	Event Linkage	"Here is an event: e.g., processing completed, error, warnings, etc."
	Data Linkage	"We share a common table (e.g., a dimension table) in our domain."
	Development Linkage	"Can I re-create your domain in development?" "Can I see the code you used to create it?" "Can I modify that code in development?" "Is there a path to production?"

DataKitchen Supported Inter-Domain Communication Links



Inter Domain Communication		DataKitchen Support
	Domain Query	YES
	Domain Query	YES
	Process Linkage	YES
	Process Linkage	YES
	Event Linkage	YES
	Data Linkage	NO
	Development Linkage	YES

Domain Development Process



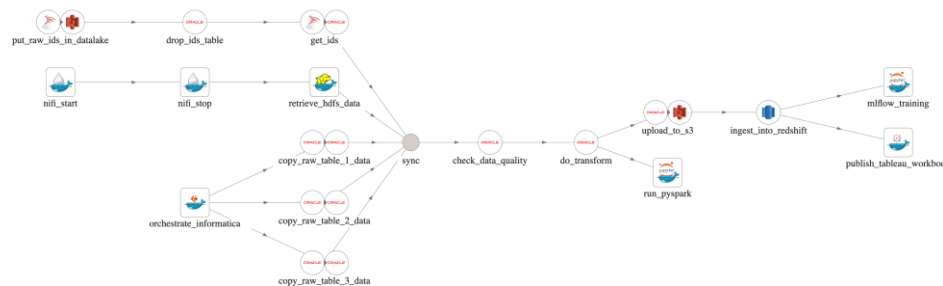
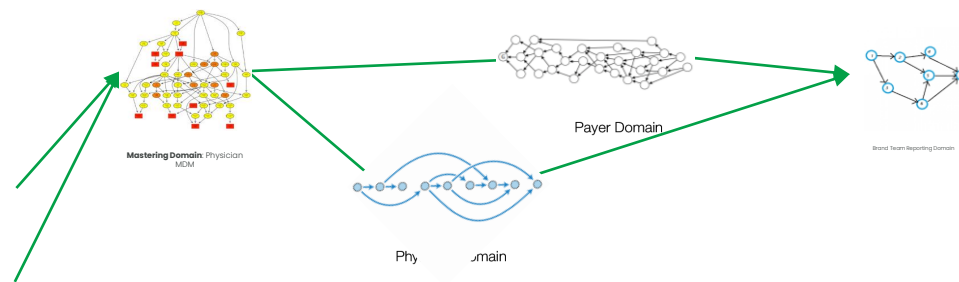
The development process is essential.

- Code changes or new data sets may affect downstream parts of the mesh.
- DataKitchen encapsulates the development & production environments

Key Questions

- How does a developer change one part & not break things?
- How do you allow local change to a domain & global governance & control?

DataKitchen Software's Role (Recipes)



DataKitchen Recipe

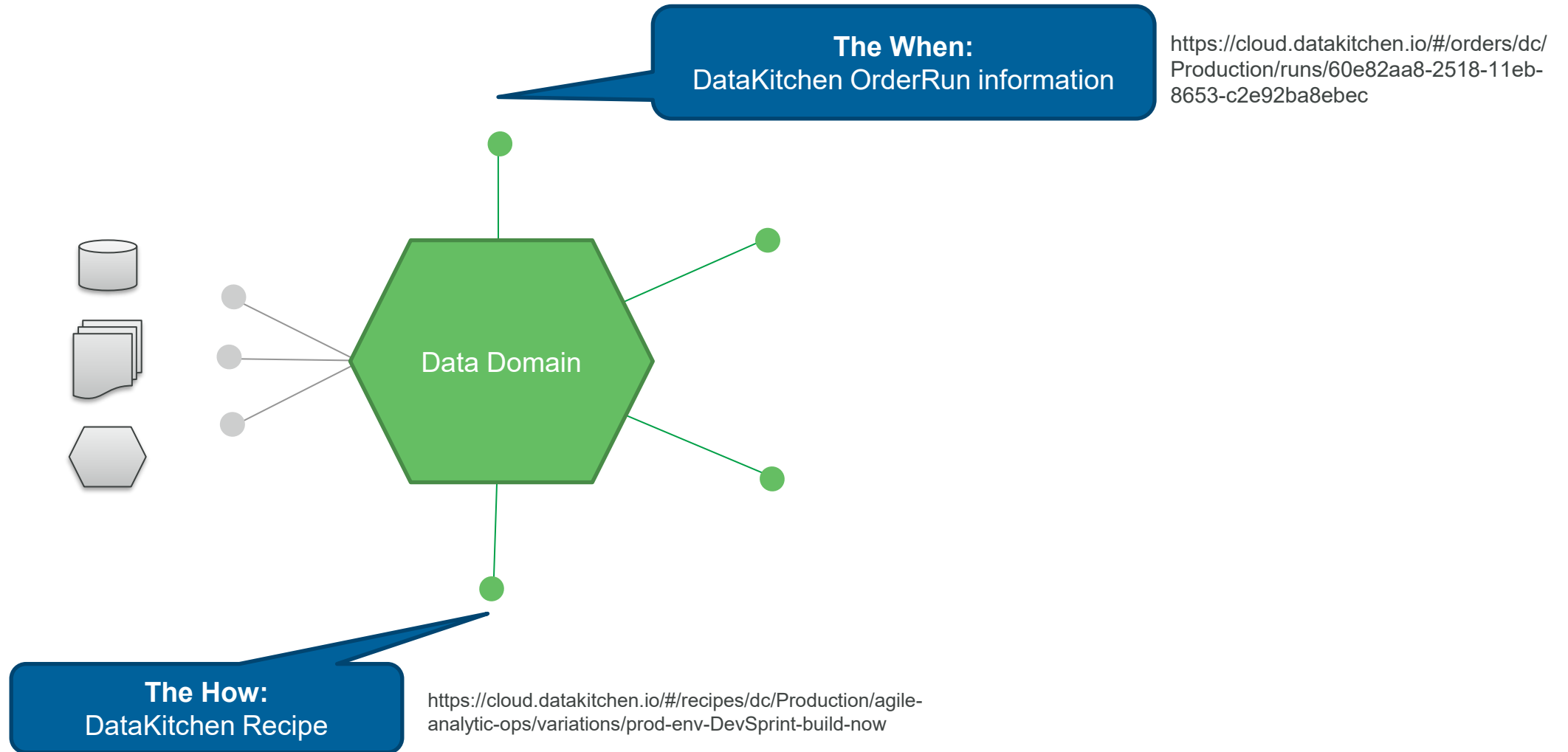
DataKitchen DataOps Capability

Intelligent, test-informed, system-wide production orchestration (meta-orchestration)

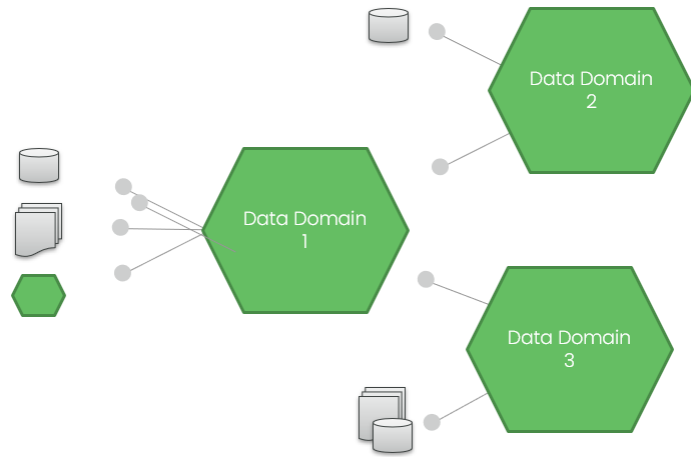
What workflow tools like Airflow, Control-M, or Azure Data Factory do not have

- Integrated Production Testing & Monitoring
- A set of connectors to the complex chain of data engineering, science, analytics, self-service, governance & database tools.
- **DataKitchen Recipes** Meta-Orchestration or a 'DAG of DAGs'

DataKitchen Domain Interfaces As URLs



DataKitchen Ingredients Allow Composition



- **DataKitchen Ingredients** allow reusable components that can be incorporated into other processing
- Each **domain can change independently**, with a centralized process to make sure the entire system is correct
- While **DataKitchen Kitchens** lets people work independently, **Ingredients** let people work dependently:
 - **Recipes** can reuse the data or artifacts that other **Recipe Variations** produce
 - **Recipes** need to incorporate other **Recipes Variations** when they run

Conclusion

Data Fabric, Data Mesh, and Functional Data engineering are exciting new paradigms

However, the DataOps part of is of paramount importance!

- The lineages & composition between domains are important
- Managing central process control & governance with local domain independence is very important

DataKitchen Features (e.g., Recipes, Tests, Kitchens & Ingredients) **can handle all the needs of the DataOps part of the mesh**

Accelerate These Patterns With DataKitchen Software

DataOps

Data Fabric

Data Mesh

Functional
Data
Engineering



DataKitchen DataOps Software Platform that delivers new business insights by enabling the development and deployment of innovative, high quality data analytic pipelines. Rapidly

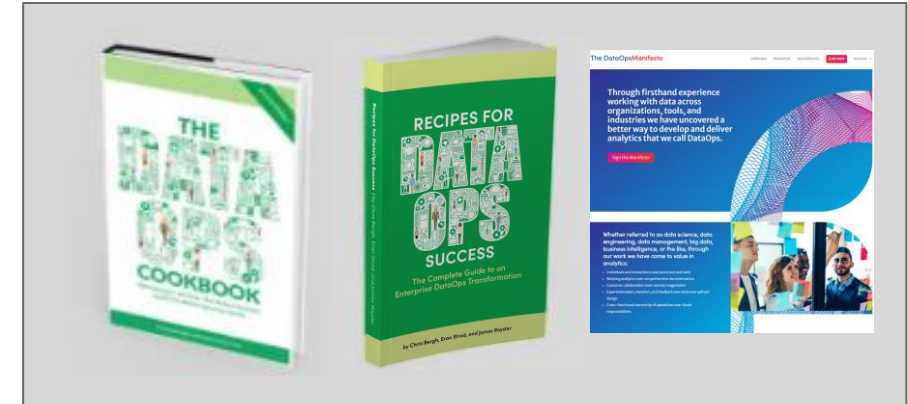
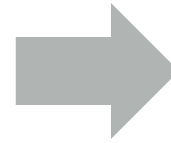
Learn More !

DataOps

Data Fabric

Data Mesh

Functional
Data
Engineering



Sign The DataOps Manifesto:

<http://dataopsmanifesto.org>

Free DataOps Cookbook:

<https://datakitchen.io/the-dataops-cookbook/>

Free DataOps Transformation Book

<https://datakitchen.io/recipes-for-dataops-success-guide-to-dataops-transformation/>