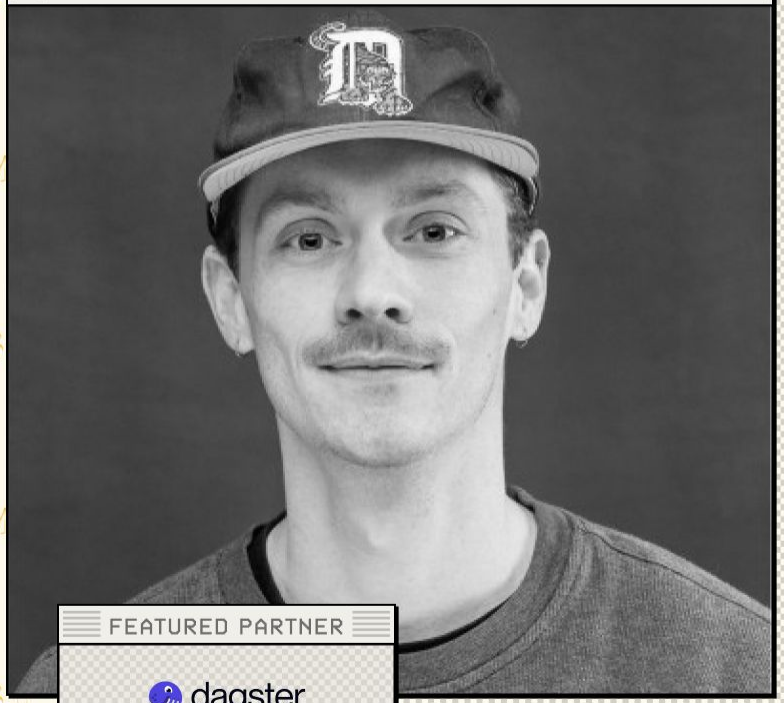
MDS Fest 3.0

MODERN DATA ARCHITECTURE AND ENGINEERING

Abstractions: Enabling data teams through reduced complexity

Colton Padden

Data Engineer & Developer Advocate,
Dagster Labs



FEATURED PARTNER

dagster

2,800

chick-fil-atech

Learn about how we are applying technology across the disciples of Software Engineering, Product Development, Analytics, and Enterprise Architecture to maximize our ability to serve Chick-fil-A Operators, Team Members, and Customers.

Follow publication

Observability at the Edge

How Chick-fil-A provides observability for 2,800+ K8s clusters

Brian Chambers Follow 8 min read · Aug 8, 2023

159 4

by Brian Chambers, Alex Crane

1,500,000,000

About Purpose News Investors Suppliers Careers Ask Walmart

Jan. 14, 2021
By Sanjay Radhakrishnan, Vice President, Global Tech, Walmart

As part of our digital transformation, we're using Internet of Things (IoT) at a scale unmatched across retail to improve food quality, lower energy consumption and keep costs low for our customers. Currently, Walmart manages more than 7 million unique IoT data points across our U.S. stores. Every day, this network of connected devices sends almost 1.5 billion messages regarding temperature, operating functions and energy use. To help manage this massive volume, the IoT team within Walmart Global Tech has built proprietary software that uses advanced algorithms to detect anomalous events in real time and take action to fix issues quickly.

IoT Explained

“We’re becoming more of a digital e-commerce operation, rather than a traditional quick-service restaurant business”

Written by

Bernard Marr

Bernard Marr is a world-renowned futurist, influencer and thought leader in the fields of business and technology, with a passion for using technology for the good of humanity. He is a best-selling author of over 20 books, writes a regular column for Forbes and advises and coaches many of the world’s best-known organisations. He has a combined following of 4

Dominos: Data-driven decision making, the world’s largest pizza delivery chain

23 July 2021

With over 10,000 outlets serving up millions of pizzas each year, Dominos is the largest pizza delivery chain in the world.

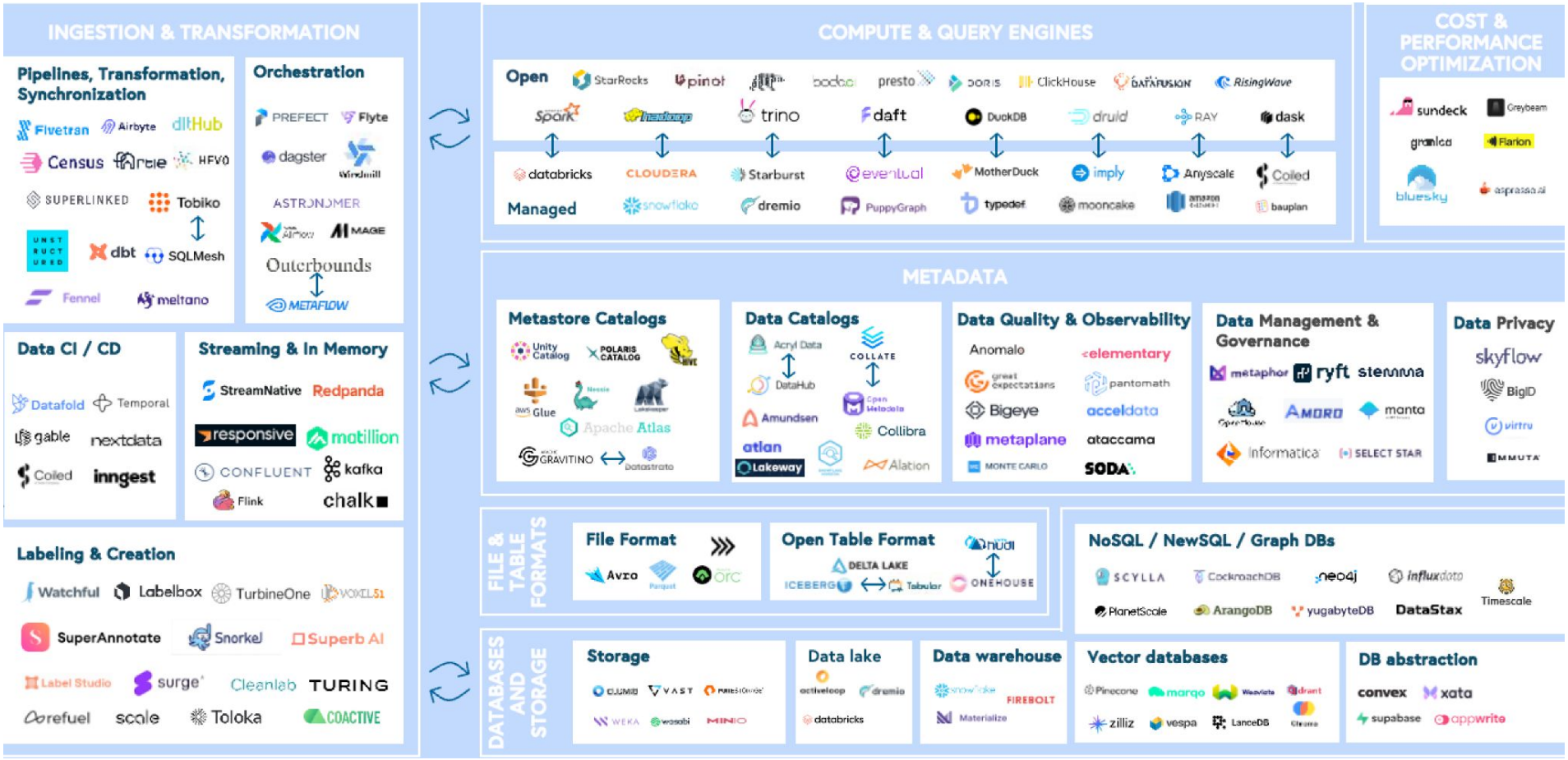
How Dominos uses Big Data in practice

The company has consistently pushed its brand onto new and developing tech, and it’s now possible to order pizzas on Twitter, smart watches and TVs, in-car entertainment systems such as Ford’s Synch, and social media platforms like Facebook. This drive to keep a Dominos order button at customers’ fingertips at all times is referred to as Dominos AnyWare.

It's estimated that 463 exabytes of data will be created each day globally.

Forbes

Data tooling has exploded in growth.

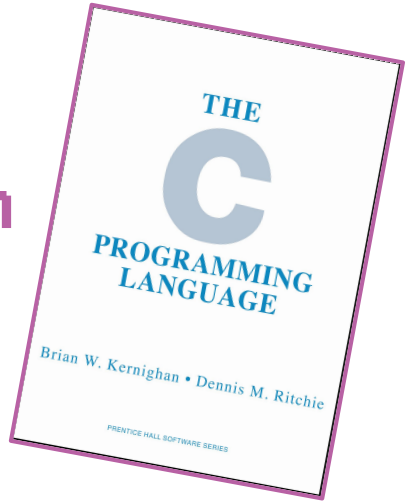


Source: Bessemer Venture Partners

A necessary complexity...

“Controlling complexity is the essence of computer programming.”

Brian Kernighan



The objective remains the same

To ~~increase shareholders value~~

To better understand the world around us!

- Evidence-based decision making
- Understand patterns in our society
- Advance scientific understanding

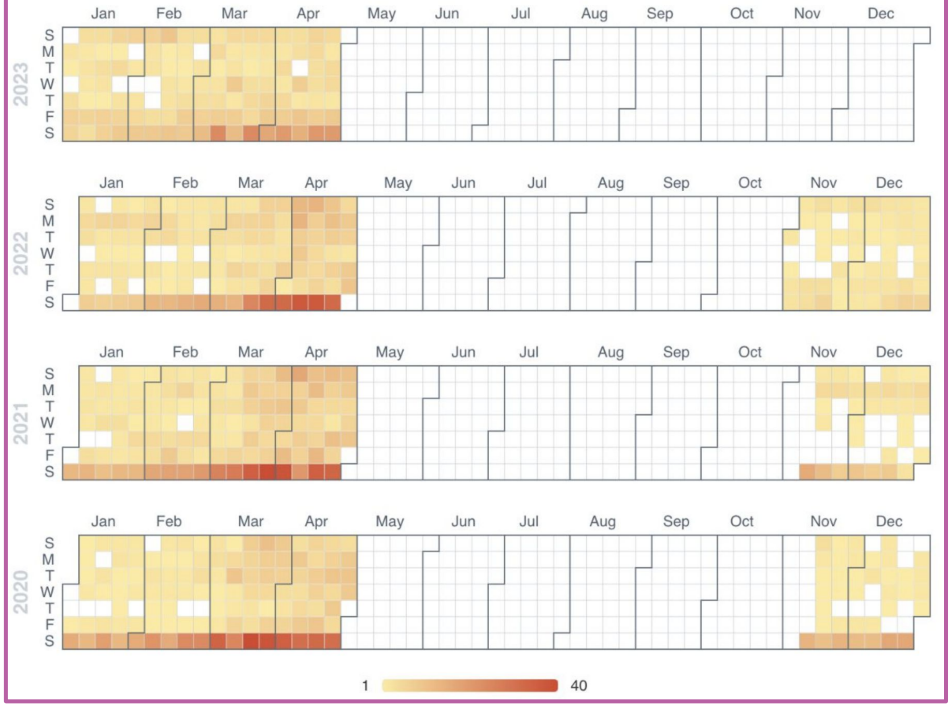
Duck Observations

There were **15** checklists, **20** observations, and **49** total species of Ducks.

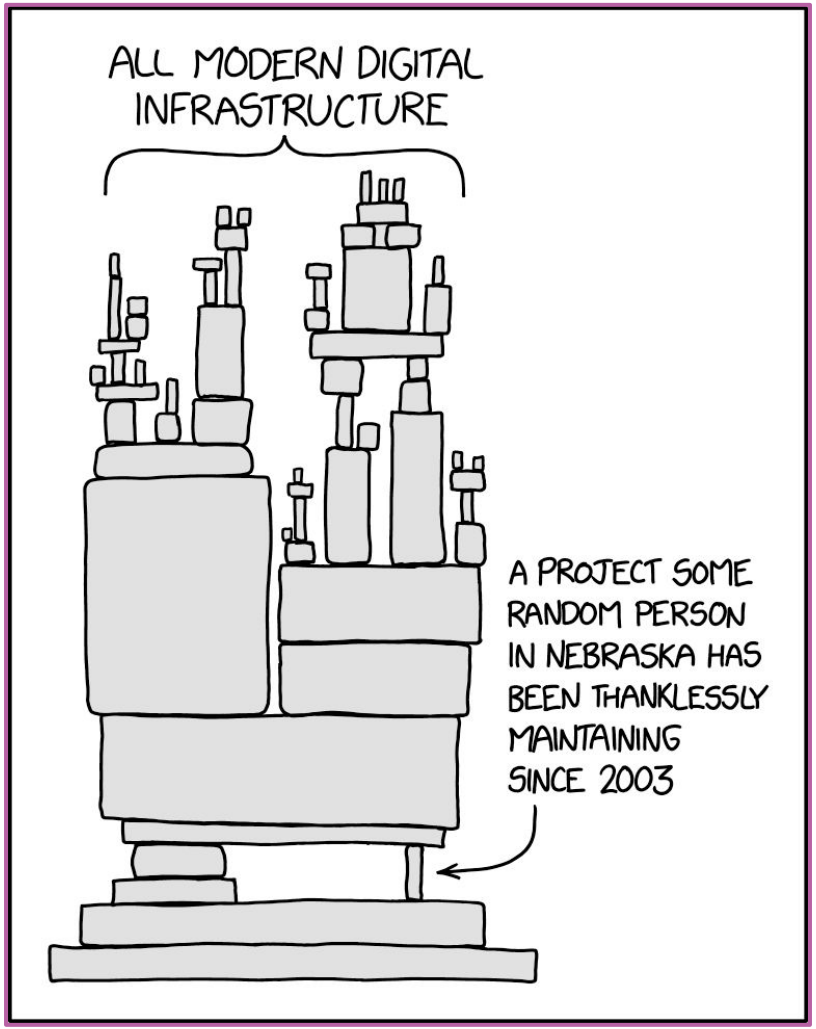
Most observations occur on Saturdays – the FeederWatch dataset is seasonal, operating from November to April of each year.

Calendar Heatmap

Daily Sales



Abstractions



xkcd.com/2347/

Types of abstractions in data engineering (1/4)

Code-level abstractions

- Utilities (CLIs) and libraries
- Design patterns (eg. factory methods)
- Domain-specific languages (DSLs)
  - YAML front-end
  - Configuration layer
- Web portals

Types of abstractions in data engineering (2/4)

Transformation Frameworks

- dbt
- SQLMesh

Abstracts

- Connection management
- Materialization strategies
- Lineage
- Query optimization

Types of abstractions in data engineering (3/4)

Table formats

- Apache Iceberg
- Delta Lake
- Hudi

Abstracts

- File management
- Compute engine
- Schema and partition management

Types of abstractions in data engineering (4/4)

Workflow Orchestrators

- Airflow
- Dagster
- Prefect

Abstracts

Scaling, resiliency, alerting, lineage, and more

The benefits: productivity

Reduced cognitive load: focus on the business logic

Reduced learning curve: no need to understand underlying systems

Standardization: consistent patterns

Faster development: make use of reusable components

The benefits: organizational

Cross-team collaboration: common abstractions can be shared, breaking down silos

Governance and compliance: through policy checks and managed processes

Knowledge encapsulation: specialized knowledge becomes accessible

Some common pitfalls

Premature abstraction:

Building generic solutions before fully understand the use case

Over-abstraction:

Not providing enough configuration, limiting functionality *

Leakiness:

Failing to hide implementation details of underlying technologies

Poor documentation:

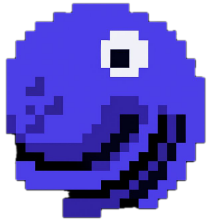
Lack of information required to use the implementation

* I've witnessed this one the most!

It's our nature to build layers of abstraction.

Being aware common pitfalls and design patterns is important, allowing us to be thoughtful in our implementation.

Example walkthrough with Dagster



But this can apply to any tool!

Dagster

Overview Runs Catalog Jobs Automation Insights Deployment

P prod C

Catalog

Reload definitions

Good evening, Colton

View all assets | View asset lineage

Search the Catalog

Filters: Asset Group | Code location | Column | Kind | Owner | Tag

Private selections

Favorites
7 assets

Saved selections

- build sheriff
- Dagster Open Platform — 34
- Maps Layer
- Marts Layer — 32
- Metrics Layer
- Model Layer — 4

Dagster

Overview Runs Catalog Jobs Automation Insights Deployment

prod

Global Asset Lineage

Reload definitions

Jump to...

Search and filter assets

Materialize all (988)...

- mart_finance
- mart_gtm
- mart_marketing
- mart_product
- mart_sales
- metrics_cloud_product
- metrics_pypi
- metrics_segment
- metrics_statsig
- metrics_telemetry
- model_bing
- model_cloud_product
- model_code_location
- model_common_room
- model_education
- model_gong
- model_google_ads

stg_thinkific__enrollments
dbt model stg_thinkific__enrollments
Healthy
Checks 4
Snowflake dbt

stg_thinkific__courses
dbt model stg_thinkific__courses
Healthy
Checks 3
Snowflake dbt

stg_thinkific__course_reviews
dbt model stg_thinkific__course_reviews

education_enrollments
Enrollments for users in courses
Healthy
Snowflake dbt

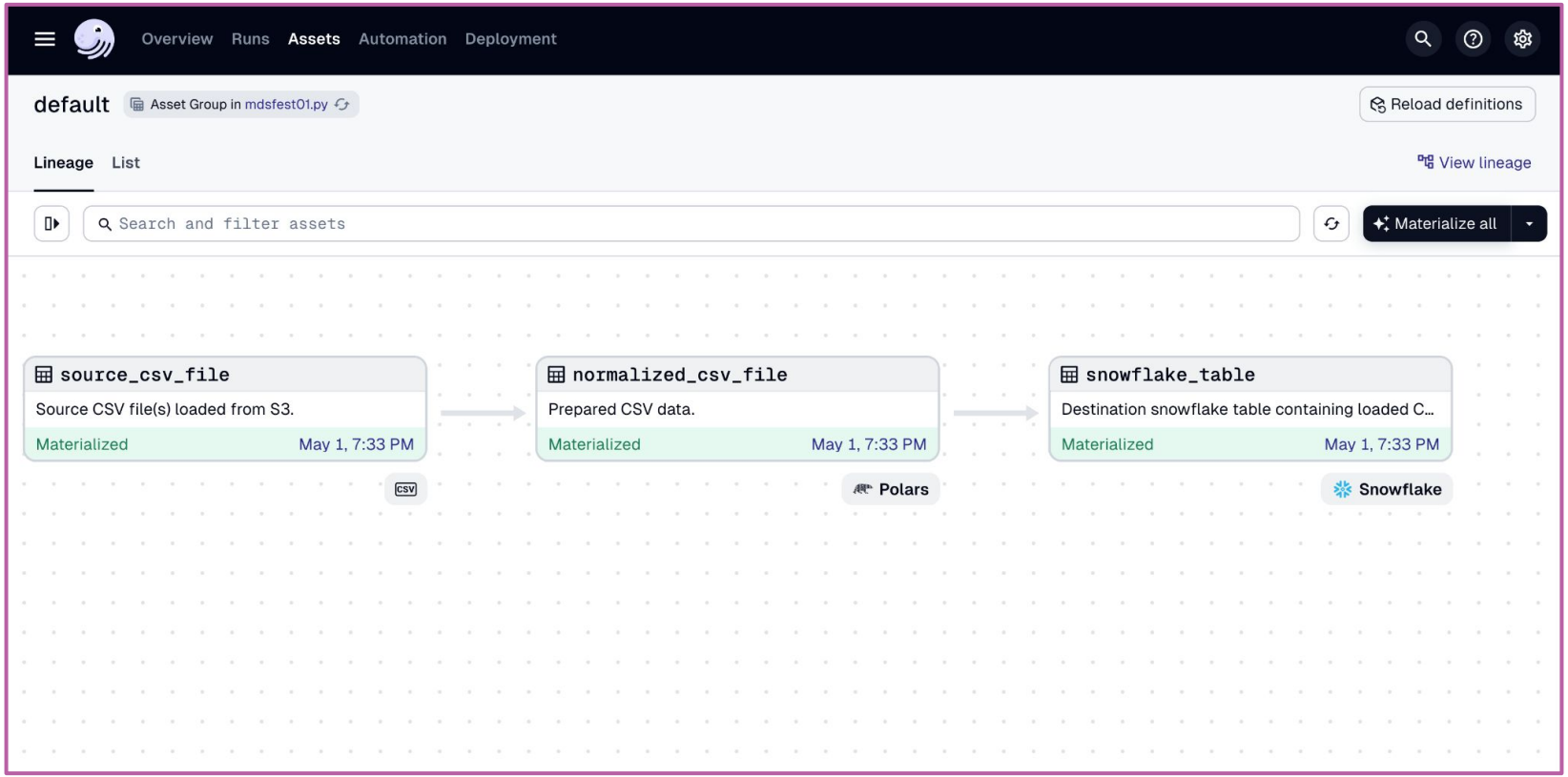
education_courses
Course information
Healthy
Snowflake dbt

education_course_reviews
All course reviews for all courses

Simplified code example

```python
import dagster as dg

TARGET_CSV_FILES = "s3://example-bucket/*.csv"
SNOWFLAKE_DESTINATION_TABLE = "example.schema.table"

@dg.asset(
    kinds={"csv"},
    description="Source CSV file(s) loaded from S3.",
)
def source_csv_file(): ...


@dg.asset(
    kinds={"polars"},
    description="Prepared CSV data.",
)
def normalized_csv_file(source_csv_file): ...


@dg.asset(
    kinds={"snowflake"},
    description="Destination snowflake table containing loaded CSV data.",
)
def snowflake_table(normalized_csv_file): ...
```

Dagster Assets



Requires understanding of...

Python concepts: Decorators, functions, sets, modules, etc

Transformation frameworks: In this case *Polars*

AWS: Authentication, object storage, boto3 bindings

Snowflake: Python SDK, connection details, etc

The goal : load CSV files to Snowflake

Using Dagster components

```yaml
type: mdsfest.lib.CSVSnowflakeComponent

attributes:
  source: s3://example-bucket/*.csv
  destination: example.schema.table
```

Enabled by the data platform owner...

```yaml
type: mdsfest.lib.CSVSnowflakeComponent

attributes:
  source: s3://example-bucket/*.csv
  destination: example.schema.table
  schedule: @daily
  owner: colton@dagsterlabs.com
  alerts:
   - on_null_values
   - on_anomoly_detection
```

The data platform owner can

Handle underlying implementation details

- Define level of abstraction
- Swap out underlying technologies
- Include data quality and other checks implicitly

Enforce standards (standard interfaces)

Expose common utilities cross-teams

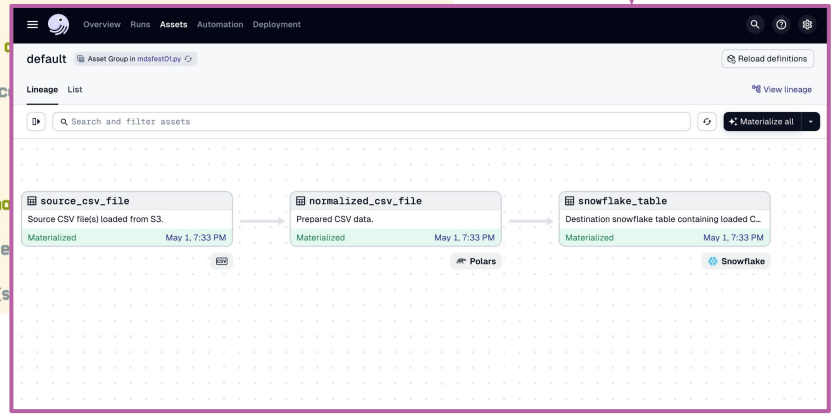
Complexity is managed by Component authors

```python
@dataclass
class CSVSnowflakeComponent(Component, Resolvable):

    source: str
    destination: str

    def build_defs(self, context: ComponentLoadContext) -> dg.Definitions:
        @dg.asset(
            kinds={"csv"},
            description="Source CSV file(s) loaded from S3.",
        )
        def source_csv_file(): ...

        @dg.asset(
            kinds={"polars"},
            description="Prepared CSV data.",
        )
        def normalized_csv_file(source_csv_file): ...


        @dg.asset(
            kinds={"snowflake"},
            description="Destination snowflake table containing loaded CSV data.",
        )
        def snowflake_table(normalized_csv_file): ...

        return dg.Definitions(assets=[source_csv_file, normalized_csv_file, snowflake_table])
```

Complexity is managed by Component authors

```yaml
type: mdsfest.lib.CSVSnowflakeComponent

attributes:
  source: s3://example-bucket/*.csv
  destination: example.schema.table
```

```python
@dataclass
class CSVSnowflakeComponent(Component, Resolvable):

    source: str
    destination: str

  def build_defs(self, context: ComponentLoadContext) -> dg.Definitions:
      @dg.asset(
          kinds={"csv"},
          description="Source CSV file(s) loaded from S3.",
      )
      def source_csv_file(): ...

      @dg.asset(
          kinds={"polars"},
          description="Prepared CSV d
      )
      def normalized_csv_file(sourc


      @dg.asset(
          kinds={"snowflake"},
          description="Destination sno
      )
      def snowflake_table(normalize

      return dg.Definitions(assets=[s
```



A marketplace of Components (eg. dbt)

Before

```python
from pathlib import Path

from dagster_dbt import (
    DbtCliResource,
    DbtProject,
    build_schedule_from_dbt_selection,
    dbt_assets,
)

import dagster as dg

RELATIVE_PATH_TO_MY_DBT_PROJECT = "./my_dbt_project"

my_project = DbtProject(
    project_dir=Path(__file__)
    .joinpath("..", RELATIVE_PATH_TO_MY_DBT_PROJECT)
    .resolve(),
)
my_project.prepare_if_dev()


@dbt_assets(manifest=my_project.manifest_path)
def my_dbt_assets(context: dg.AssetExecutionContext, dbt: DbtCliResource):
    yield from dbt.cli(["build"], context=context).stream()
```

After

```yaml
type: dagster_dbt.DbtProjectComponent

attributes:
  project: ../my_dbt_project
```

Automatic documentation

Overview Runs Assets Automation Deployment

Code locations / mdsfest

Reload

Overview Docs Beta Definitions

Jump to...

- mdsfest
  - mdsfest.lib.CSVSnowflakeComponent
- dagster

mdsfest.lib.CSVSnowflakeComponent

CSVSnowflakeComponent(target_csv_files: str, snowflake_destination_table: str)

1. Scaffolding
2. Schema
3. Example `component.yaml`

Scaffolding

Use the scaffolding command to generate the necessary files and configuration for your component.

```
dg scaffold mdsfest.lib.CSVSnowflakeComponent {component_name}
```

Schema

CSVSnowflakeComponentModel

snowflake_destination_table string

target_csv_files string

A quick tangent

(maybe?)

AI is the next layer of abstraction, and the vibe coders are coming...

Abstractions in the world of AI

1. Well **documented** interfaces and abstractions are critical
2. Thoughtful **information architecture** improves efficiency and accuracy of LLMs
3. Components and **self-contained modules** help fight sprawl
4. Components help enable “safe spaces” for **vibe coding**

Summary

Summary

- The world (and data engineering) is built on **abstractions**
- Abstractions **enable domain experts** without being bogged down
- **Thoughtful design** is crucial to prevent leaky or limiting abstractions
- Dagster provides **the Components framework** to enable data platform owners
- Strong information architecture prepares us for the next layer… **AI**

- dagster.io/slack
- docs.dagster.io

Come chat with me about abstractions & Dagster!

- bsky.app/profile/colton.boo
- linkedin.com/in/colton-padden

MDS FEST