Introduction to using and scaling dagster

Aleksandar Milicevic
Georg Heiler

Agenda

1 **Welcome and introduction to**
- What is a data platform
- What is the role of dagster

2 **Crash course on Dagster concepts**
- Asset based
- Metadata-created pipelines
- Resources and IO managers

3 **Hands-on lab** (local or GitHub Codespace)
- Spin up the environment
- Tour Dagster UI and run first asset
- Understand, run and extend the above Dagster concepts

4. **Dagster @Magenta**
- Open source implementation with local-data-stack

QA & Wrap-up

About us

Data expert in academia and industry Magenta Telecom

- meetup organizer and conference speaker

- data architecture, multimodal and complex data challenges

geoheil

geoheil

@geoheil.com

Data engineer at Magenta Telecom

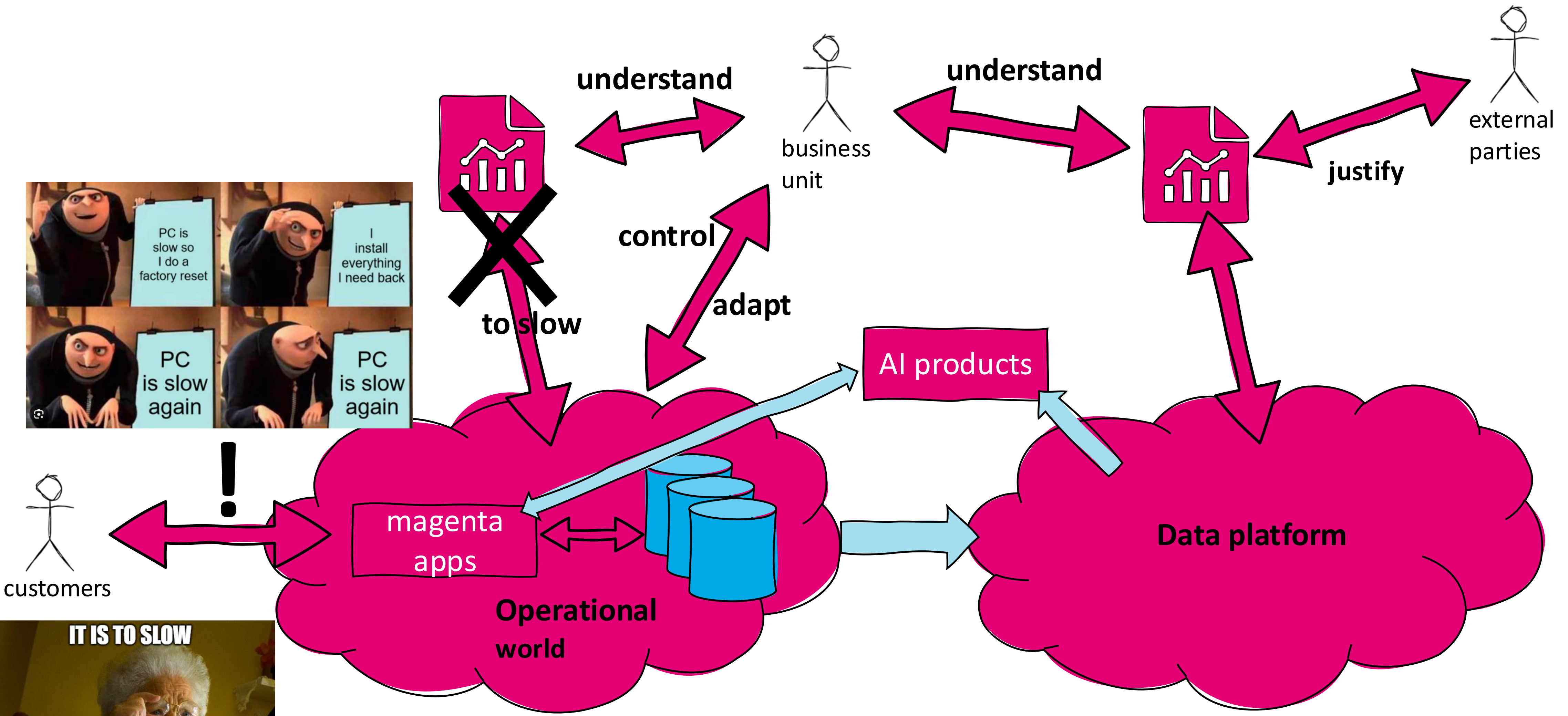
- database internals, data platform engineering

milicevica23

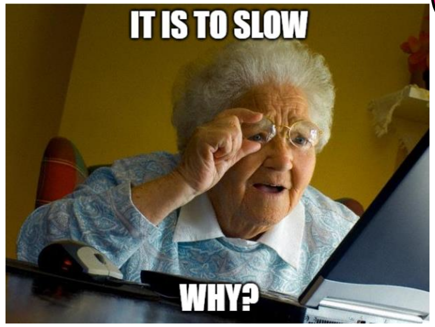
milicevica23

@milicevica23.bsky.social

Data platform is there to help understand business process improve service



PC is slow so I do a factory reset
I install everything I need back
PC is slow again
PC is slow again

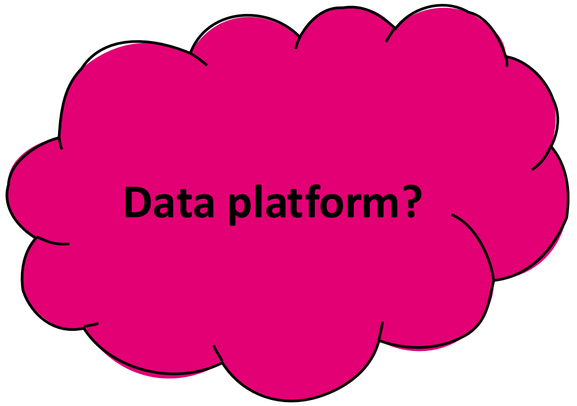


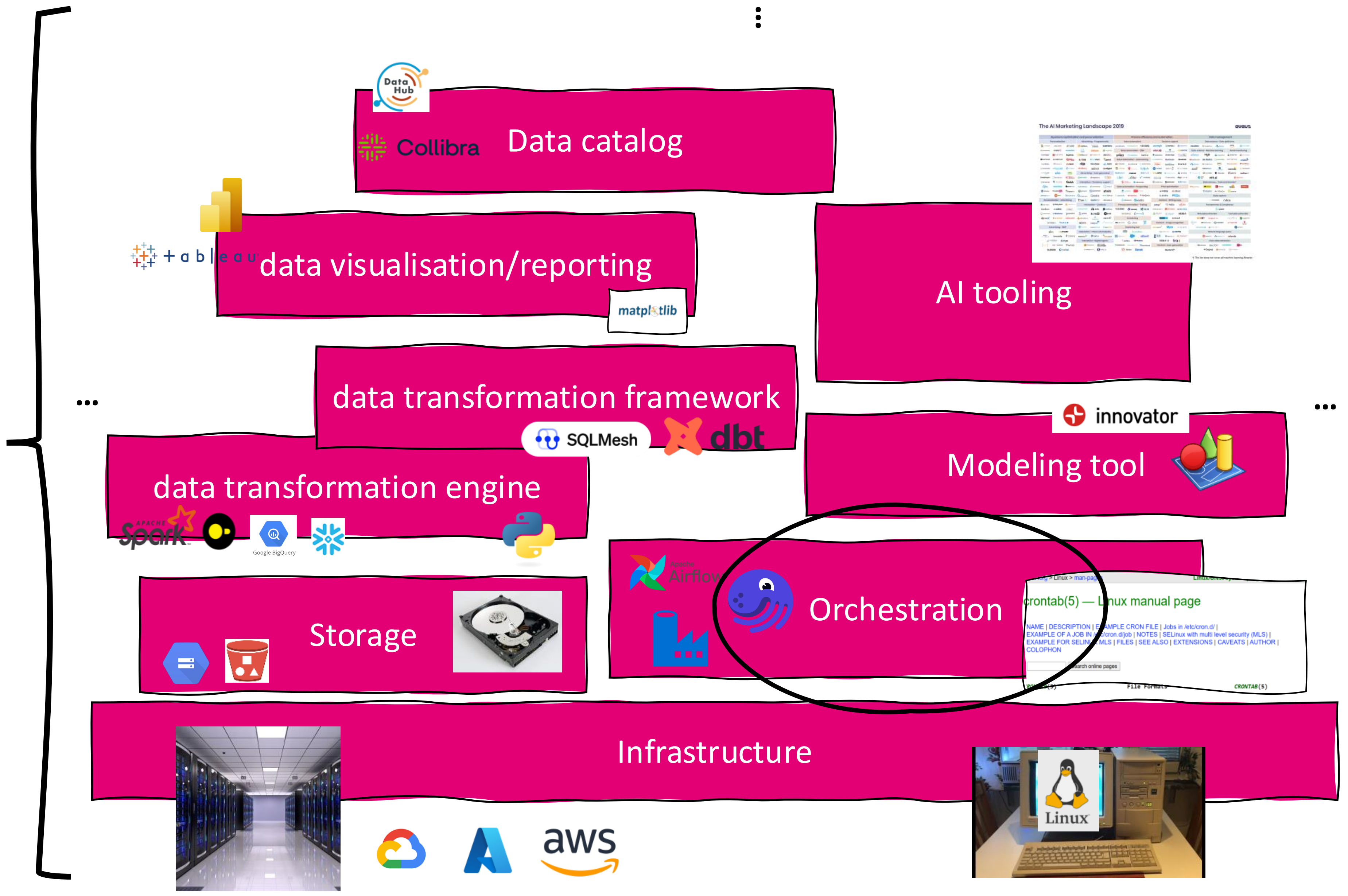
- customer facing performance is important – reporting is second priority (in a short run)
- different technology for different use case – oltp vs olap databases
- company data is an asset as car, phone or internet cable

Data platform components

What do I need for a bare minimum E2E reporting?

How to make it last?
Modern tooling, governance, development…

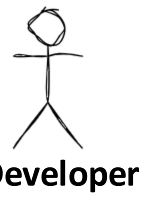


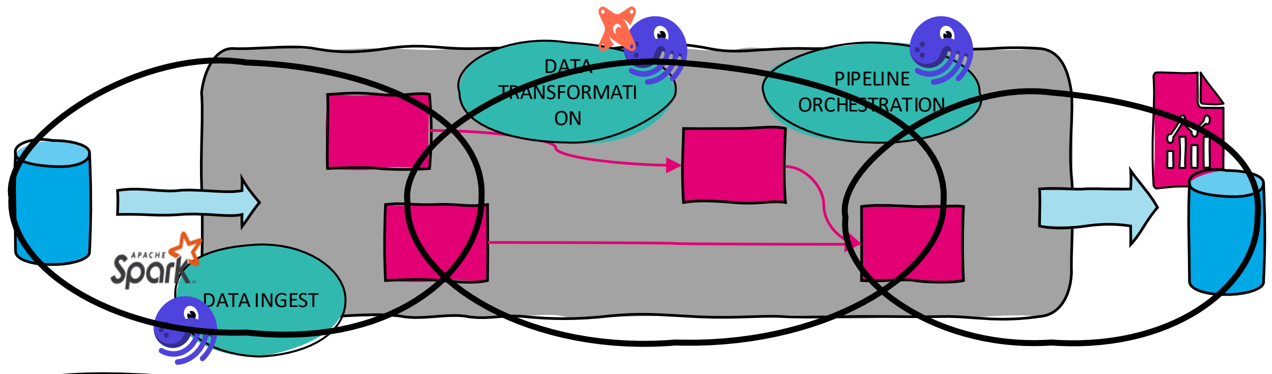


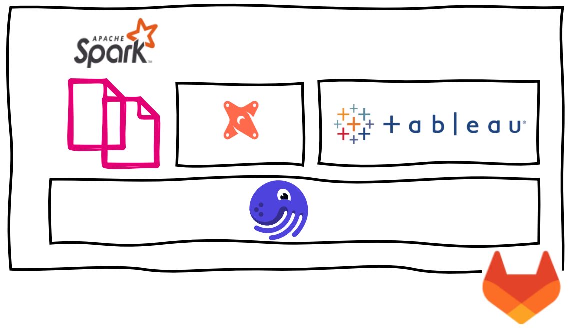
How to make it appealing?
AI…

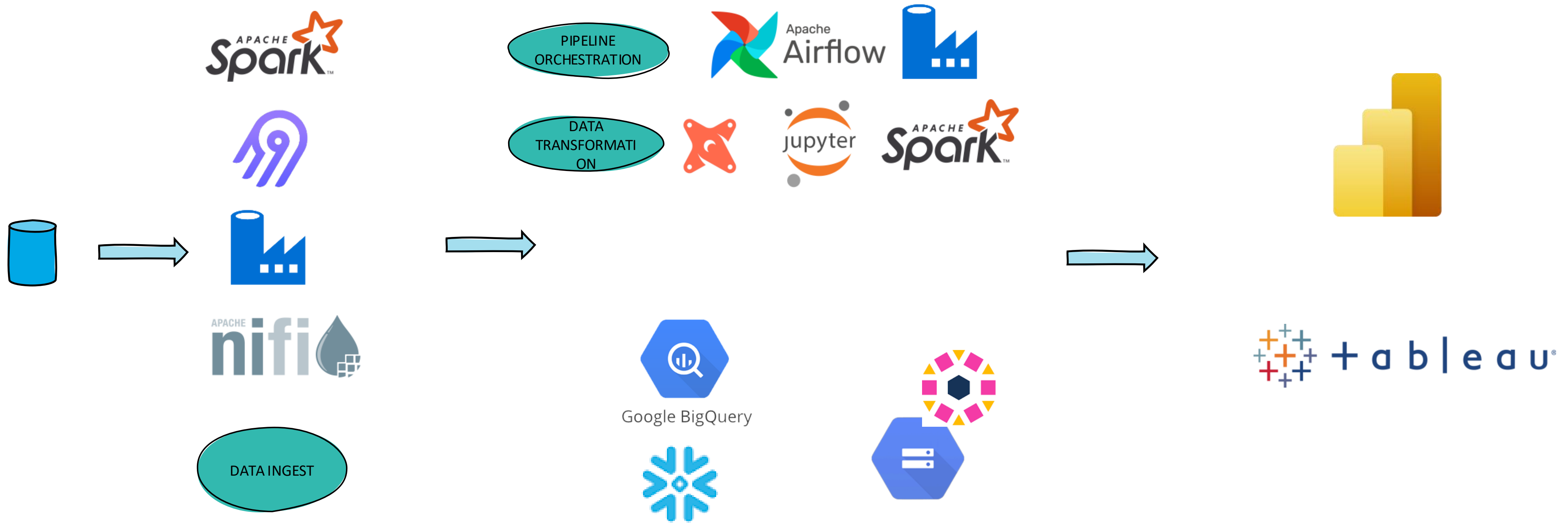
Understanding tool silos

What should i do to get E2E reporting use case done?

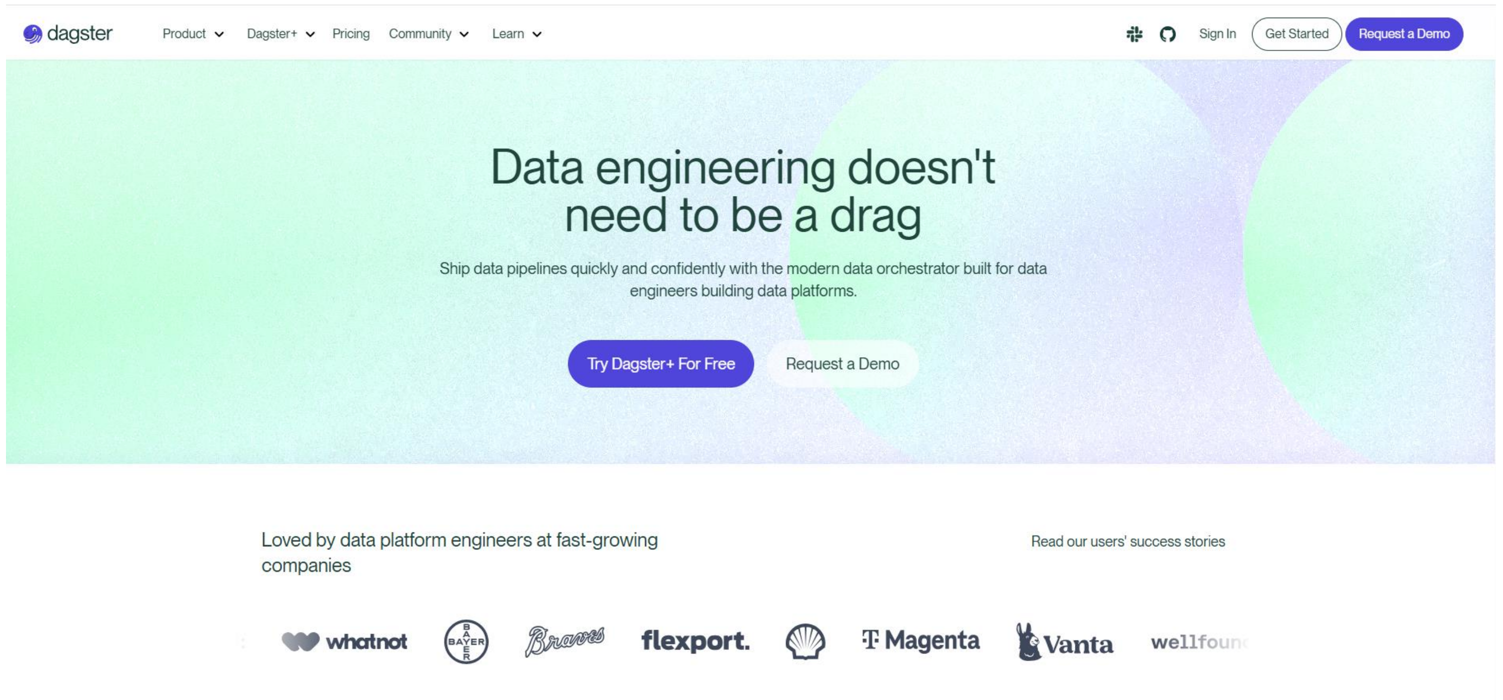








Dagster as the core of the platform



Data engineering doesn't need to be a drag

Ship data pipelines quickly and confidently with the modern data orchestrator built for data engineers building data platforms.

Loved by data platform engineers at fast-growing companies

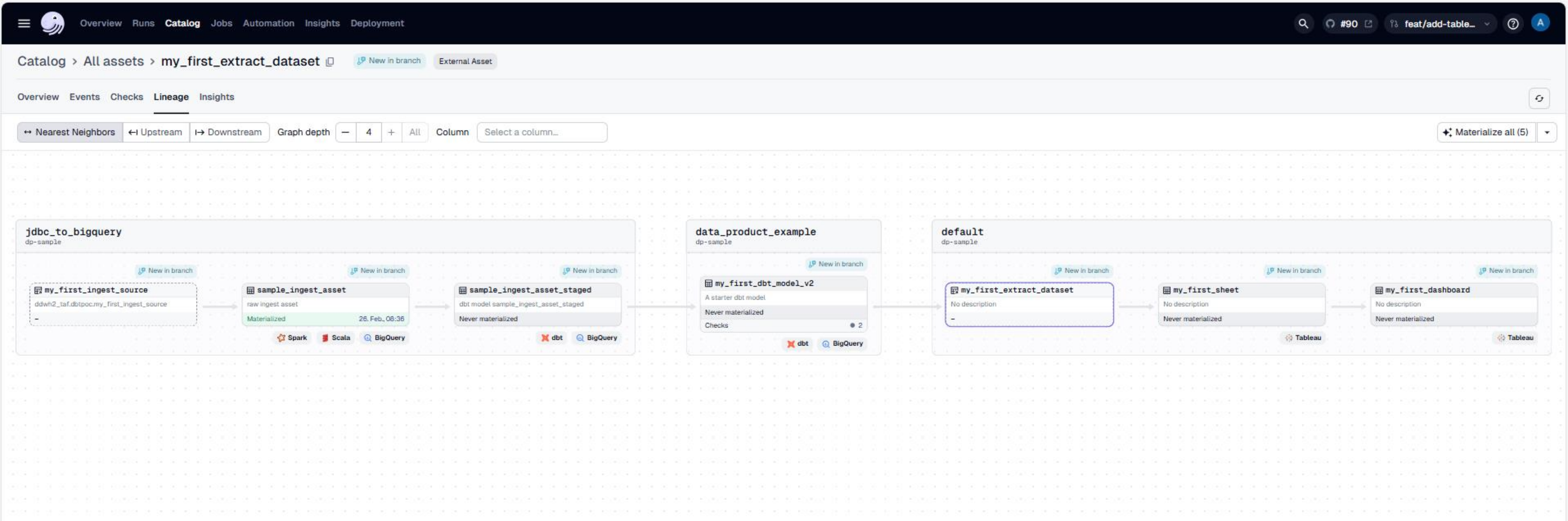
Read our users' success stories

- at Magenta we decided to build around the orchestrator
  - hybrid deployment – controlplane SaaS – runtime in our k8s
  - software engineering best practices for project development and deployment
  - asset-based mindset for data flows (graph like a calculator for data dependencies)
- new concepts in orchestration...

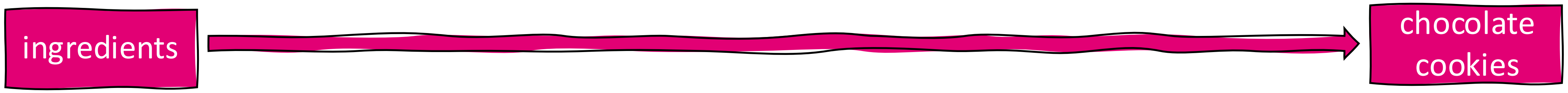
New enabled concepts

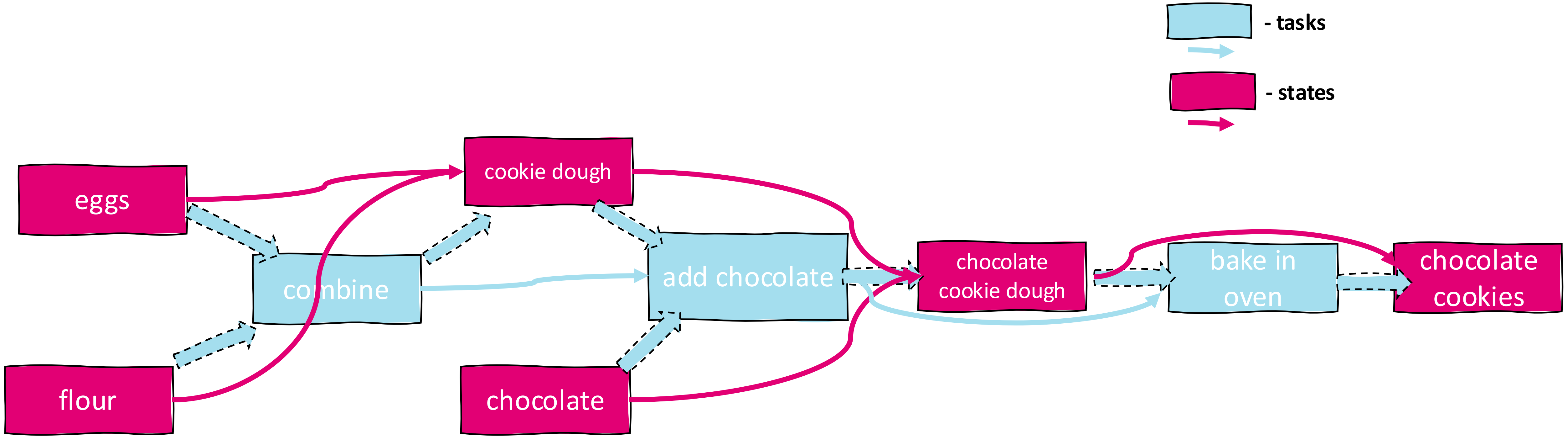
docs.dagster.io/getting-started/concepts

- Asset based graph
- Metadata driven pipeline creation
- Reusable objects (resource, io manager)

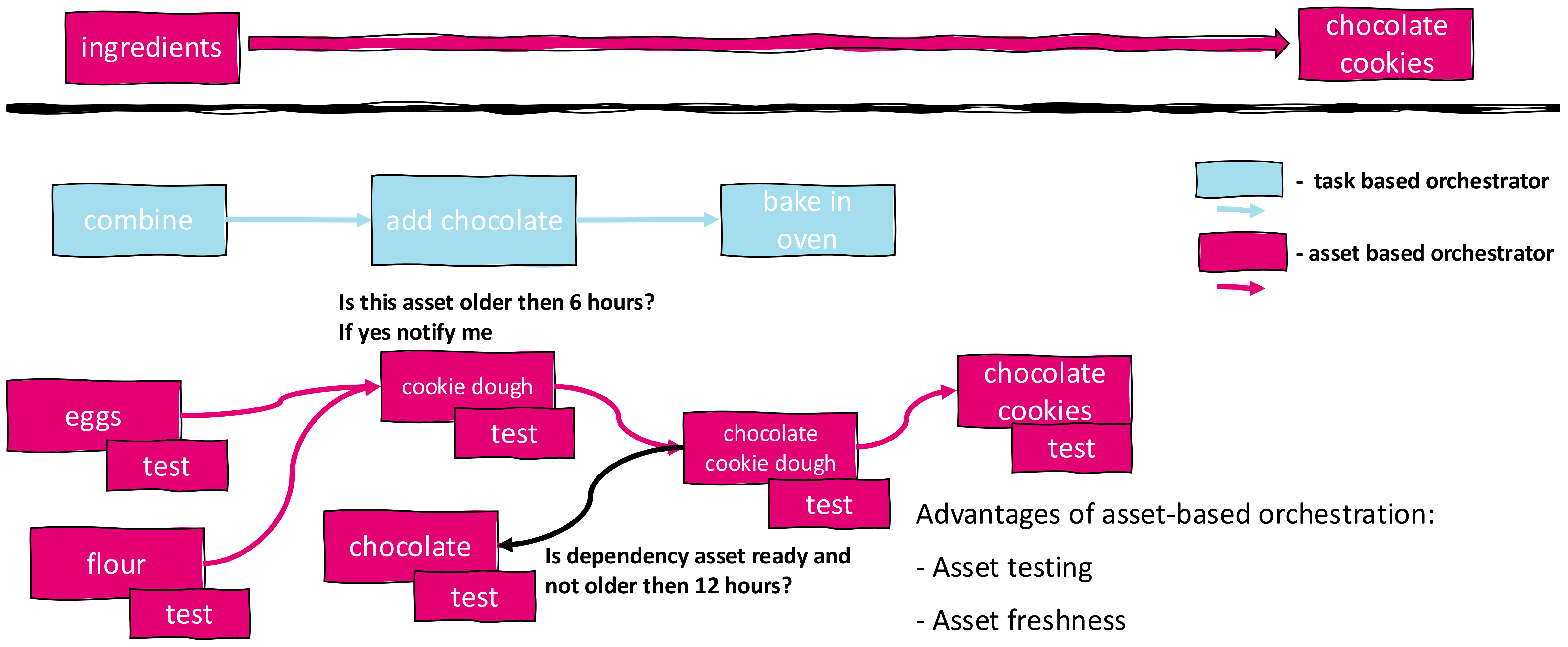


Asset and Task based orchestration: Chocolate cookie example





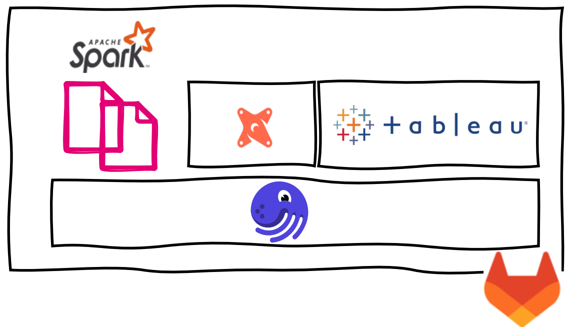
Asset based orchestration

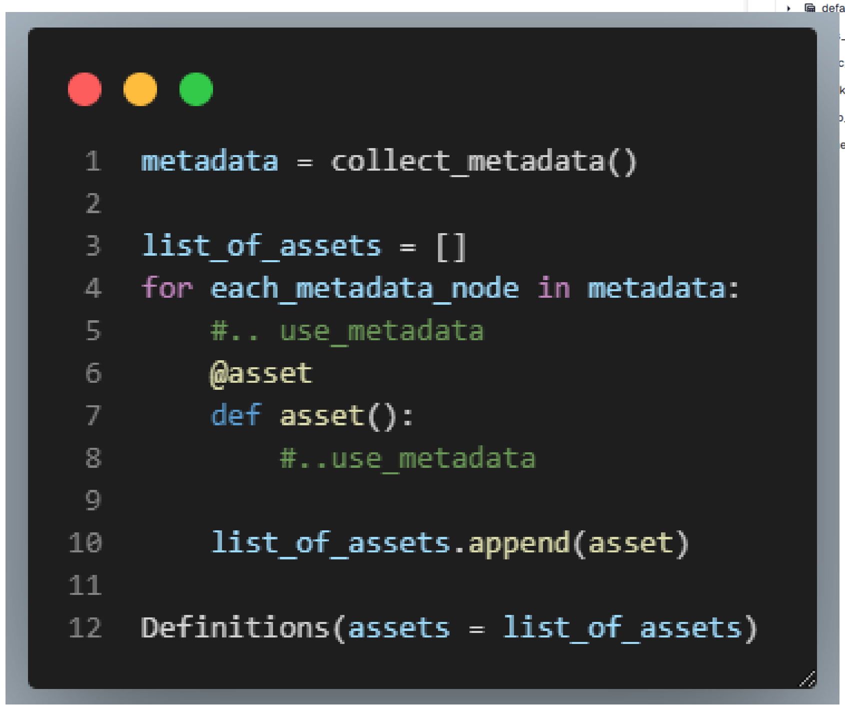


Advantages of asset-based orchestration:

- Asset testing
- Asset freshness
- Asset dependecy graph with granular declarative scheduling approach

Machine-readable metadata pipeline generation



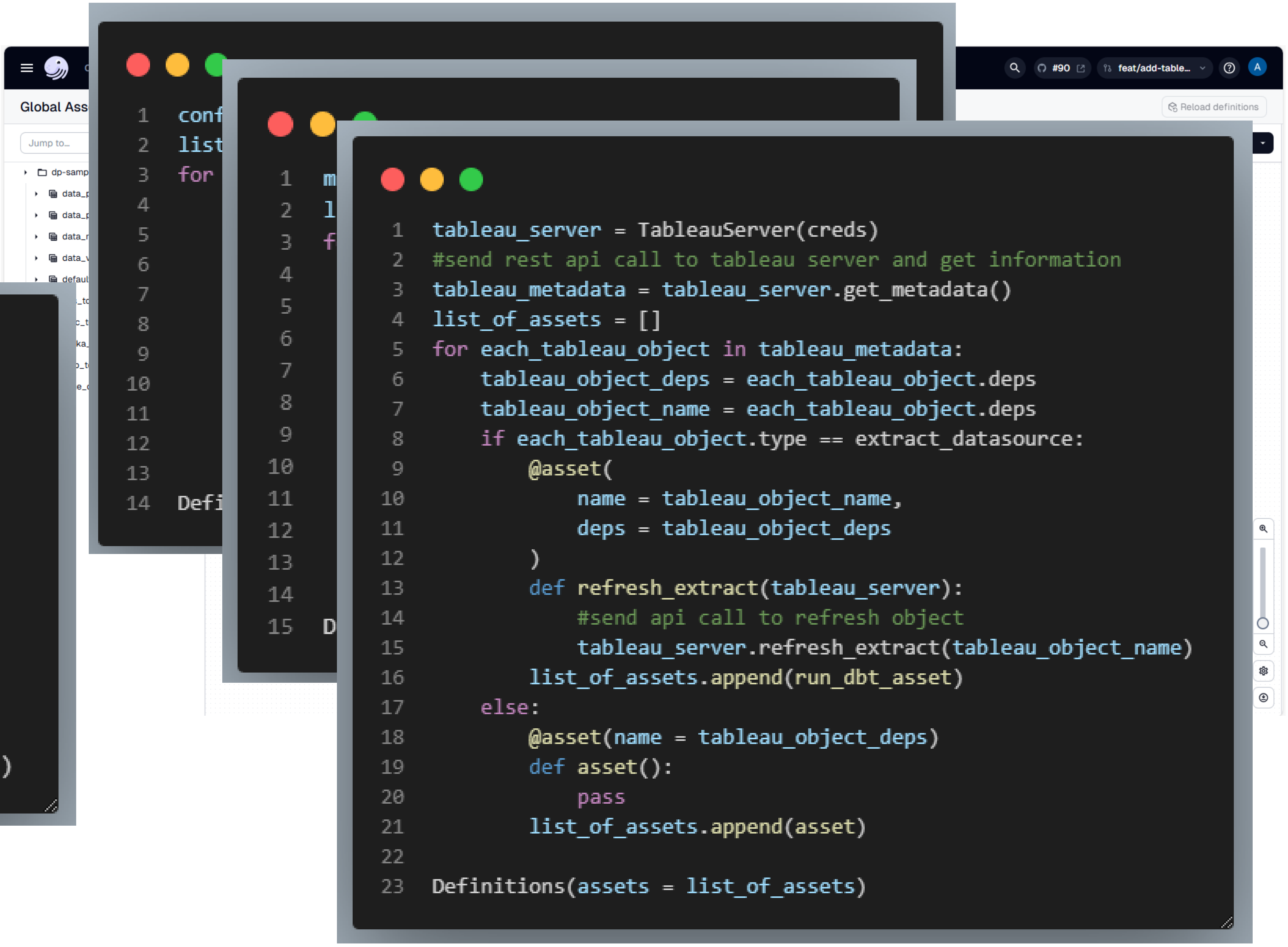


```python
metadata = collect_metadata()

list_of_assets = []
for each_metadata_node in metadata:
    #.. use_metadata
    @asset
    def asset():
        #..use_metadata

    list_of_assets.append(asset)

Definitions(assets = list_of_assets)
```

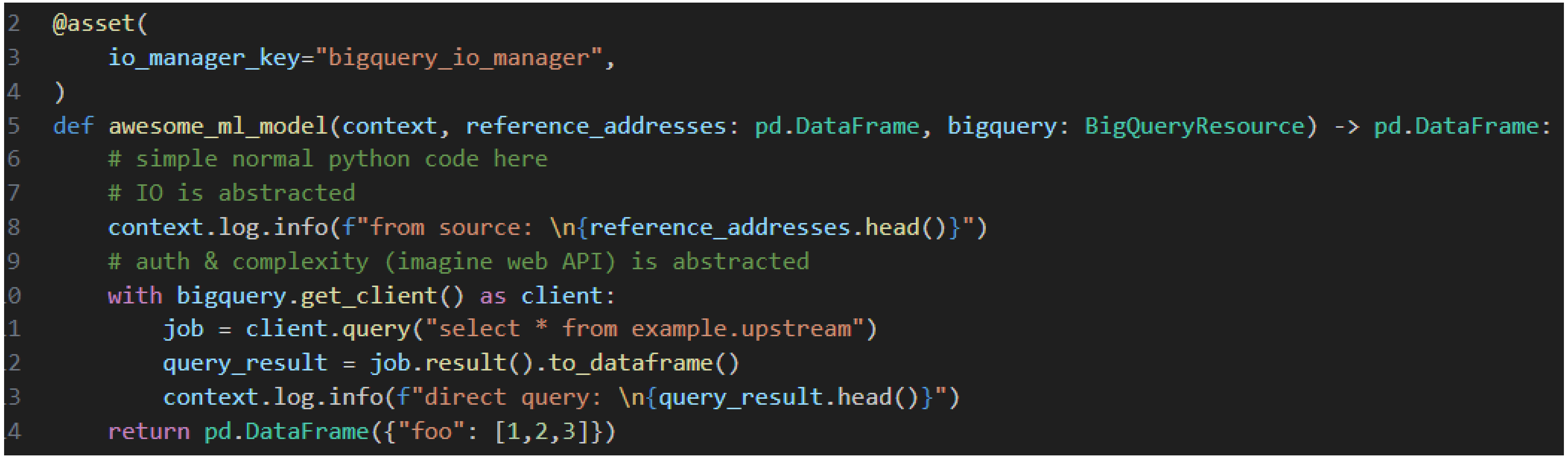


```python
tableau_server = TableauServer(creds)
#send rest api call to tableau server and get information
tableau_metadata = tableau_server.get_metadata()
list_of_assets = []
for each_tableau_object in tableau_metadata:
    tableau_object_deps = each_tableau_object.deps
    tableau_object_name = each_tableau_object.deps
    if each_tableau_object.type == extract_datasource:
        @asset(
            name = tableau_object_name,
            deps = tableau_object_deps
        )
        def refresh_extract(tableau_server):
            #send api call to refresh object
            tableau_server.refresh_extract(tableau_object_name)
        list_of_assets.append(run_dbt_asset)
    else:
        @asset(name = tableau_object_deps)
        def asset():
            pass
        list_of_assets.append(asset)

Definitions(assets = list_of_assets)
```

Reusable components

- Reusable objects
  - Resources → Encapsulate complex logic to interact with external systems
  - IO manager → Make complex IO interactions substitutable & testable
- Benefits
  - Dependency injection
  - Day 1 productivity: Scale the data pipeline down to a single laptop
  - Increase self-service: Business/DS focus not required to handle complex IO

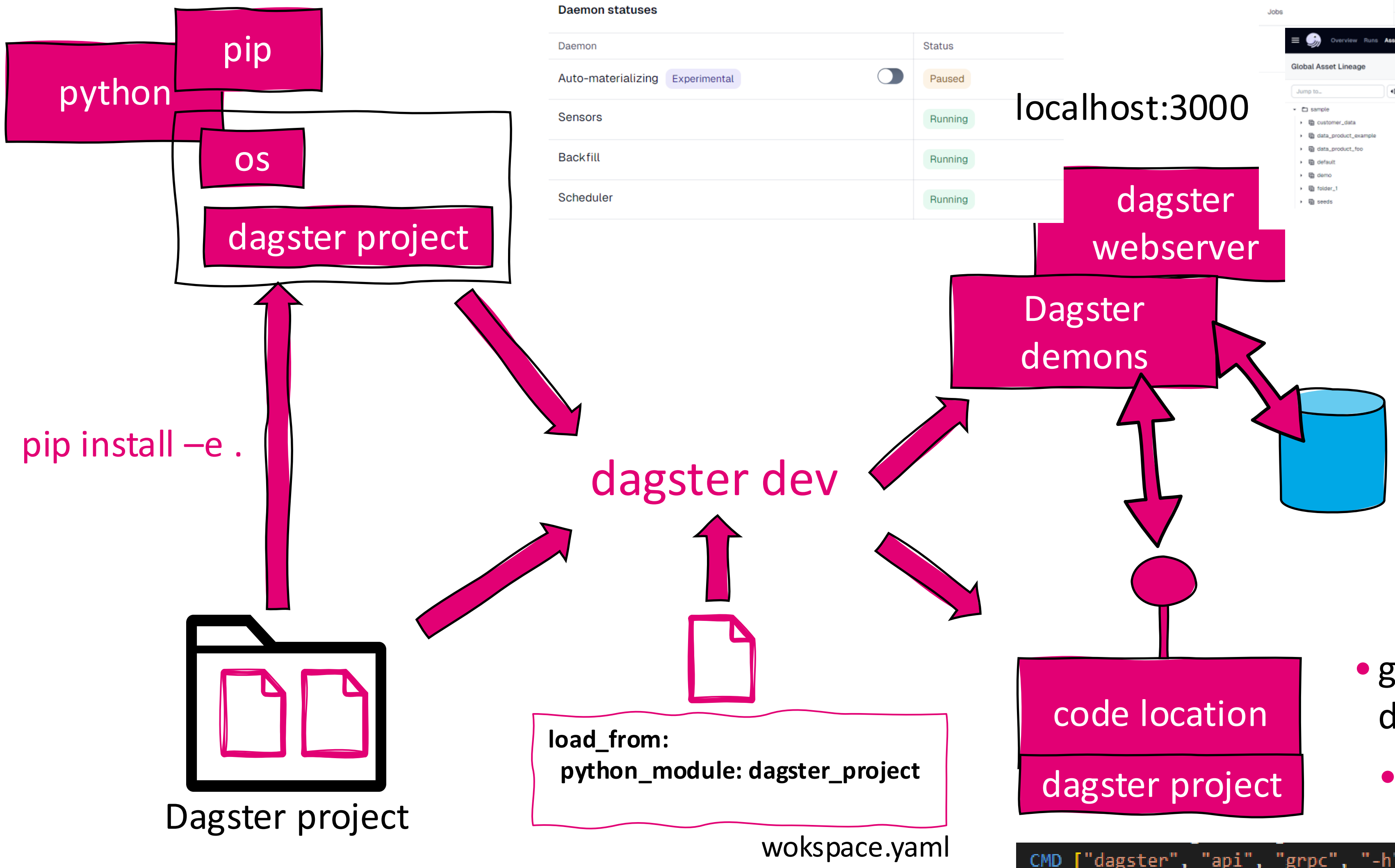


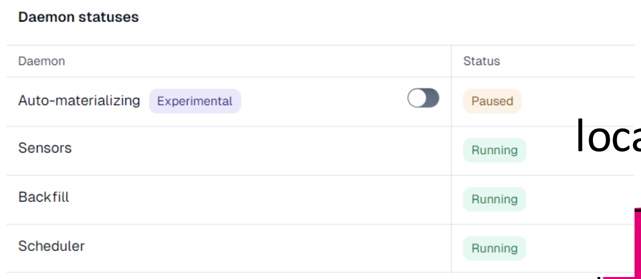
```python
@asset(
    io_manager_key="bigquery_io_manager",
)
def awesome_ml_model(context, reference_addresses: pd.DataFrame, bigquery: BigQueryResource) -> pd.DataFrame:
    # simple normal python code here
    # IO is abstracted
    context.log.info(f"from source: \n{reference_addresses.head()}")
    # auth & complexity (imagine web API) is abstracted
    with bigquery.get_client() as client:
        job = client.query("select * from example.upstream")
        query_result = job.result().to_dataframe()
        context.log.info(f"direct query: \n{query_result.head()}")
    return pd.DataFrame({"foo": [1,2,3]})
```

Lab - Demo

- basic asset and UI overview
- resources
- metadata pipeline creation
- components

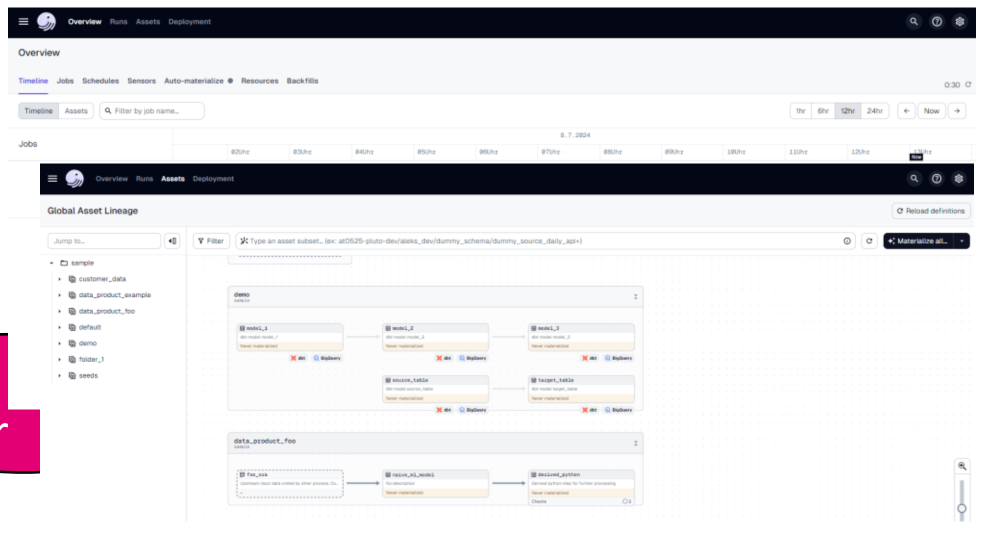
dagster - pip install dagster





Daemon statuses

Daemon	Status
Auto-materializing Experimental	Paused
Sensors	Running
Backfill	Running
Scheduler	Running


Overview
Timeline Jobs Schedules Sensors Auto-materialize Resources Backfills
Global Asset Lineage
Reload definitions
Materialize all

- storage for dagster runs and metadata
  - sqlite/postgres

- grpc server serving the assets to the dagster deamon
  - needs definition object from the module

```
CMD ["dagster", "api", "grpc", "-h", "0.0.0.0", "-p", "4000", "-f", "sample/__init__.py"]
```

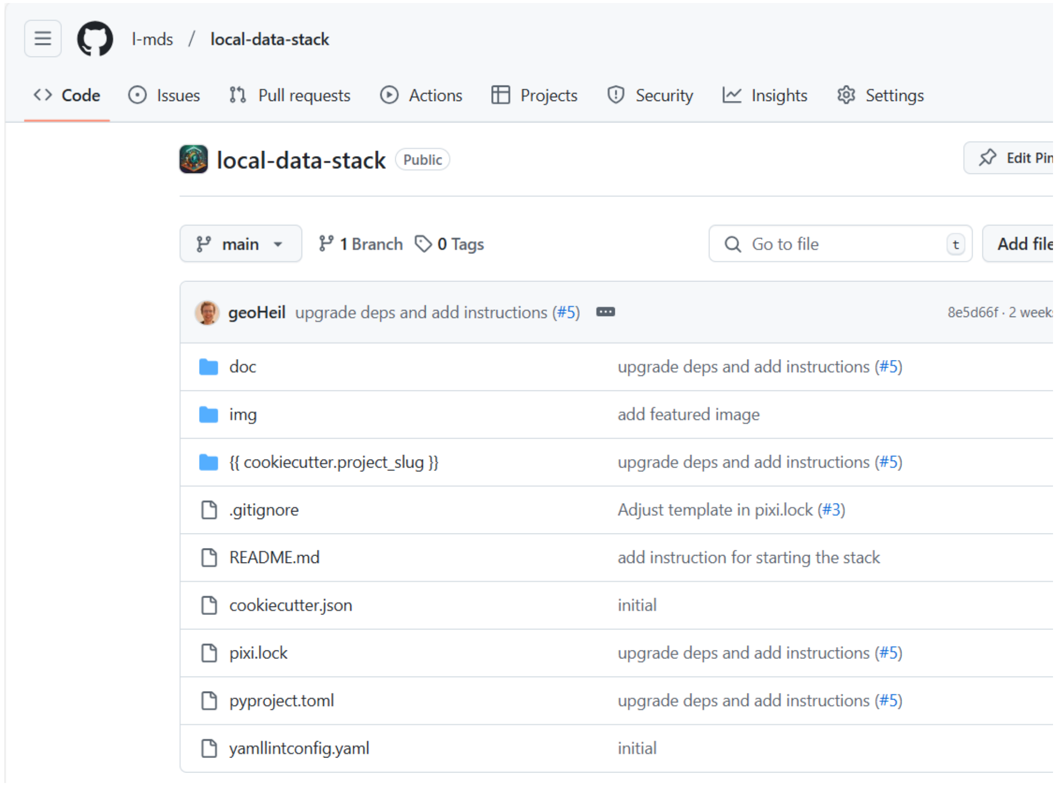
Takeaways

- Integrated asset-based graph is key (from ingest, transformation, reporting, tests – to AI)
  - Event driven connection
  - Better collaboration (scaling)
- Software engineering principles enable business self service
  - Blueprint
  - Automate all the things: CI/CD (stateful & stateless)
  - DRY: build tested foundation – dependency injection
  - Make business departments part of the key processes and pipelines
- Executable specification (metadata, contracts)
  - Interface Mangement
  - Preserve semantics
  - Preserve compliance (security classification, PII, retention)

Q&A

Dagster @Magenta
Local Modern Data Stack

Open-source contributions: L-MDS + Tableau integration + VertexAI integration



l-mds / local-data-stack

Code · Issues · Pull requests · Actions · Projects · Security · Insights · Settings

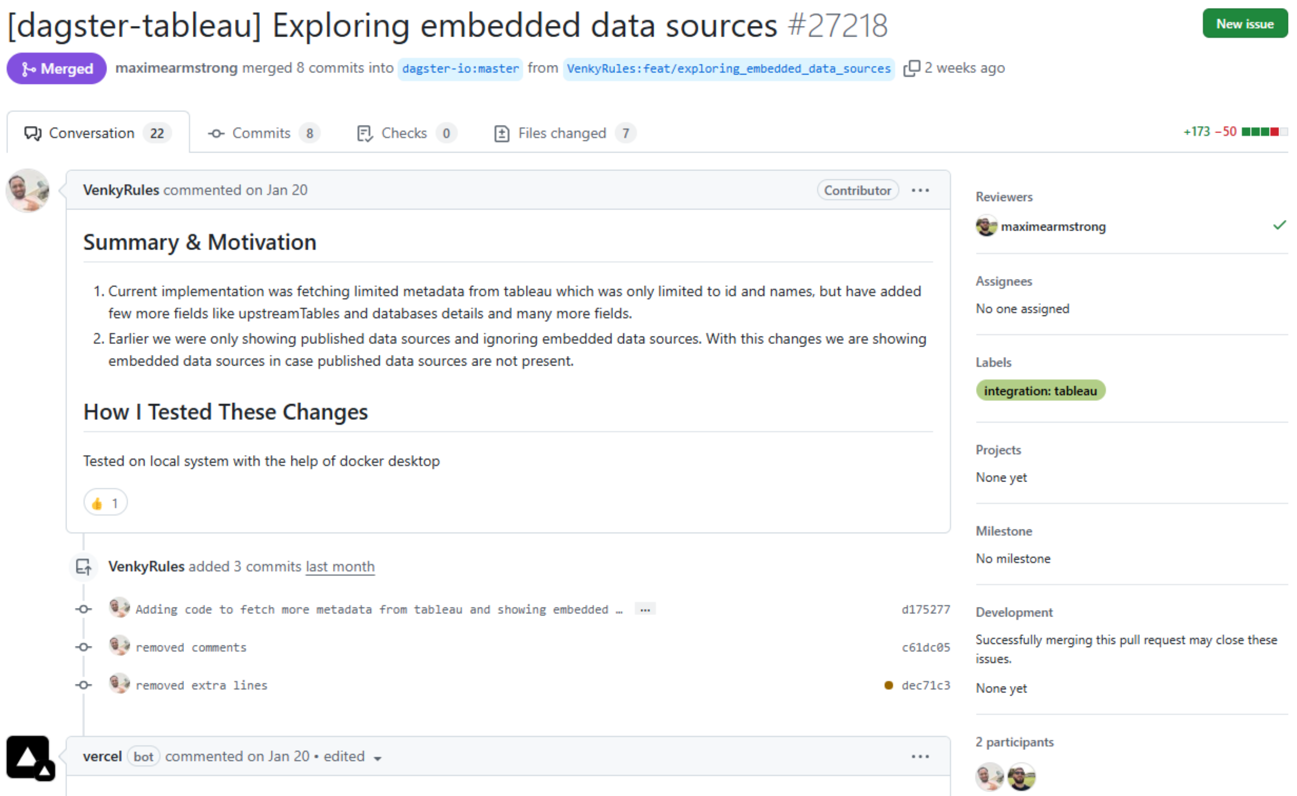
local-data-stack Public

Edit Pin

main · 1 Branch · 0 Tags · Go to file · Add file

geoHeil upgrade deps and add instructions (#5) · 8e5d66f · 2 week

doc	upgrade deps and add instructions (#5)
img	add featured image
{{ cookiecutter.project_slug }}	upgrade deps and add instructions (#5)
.gitignore	Adjust template in pixi.lock (#3)
README.md	add instruction for starting the stack
cookiecutter.json	initial
pixi.lock	upgrade deps and add instructions (#5)
pyproject.toml	upgrade deps and add instructions (#5)
yamllintconfig.yaml	initial



[dagster-tableau] Exploring embedded data sources #27218

New issue

Merged maximearmstrong merged 8 commits into `dagster-io:master` from `VenkyRules:feat/exploring_embedded_data_sources` 2 weeks ago

Conversation 22 · Commits 8 · Checks 0 · Files changed 7 · +173 −50

VenkyRules commented on Jan 20 · Contributor

Summary & Motivation

1. Current implementation was fetching limited metadata from tableau which was only limited to id and names, but have added few more fields like upstreamTables and databases details and many more fields.
2. Earlier we were only showing published data sources and ignoring embedded data sources. With this changes we are showing embedded data sources in case published data sources are not present.

How I Tested These Changes

Tested on local system with the help of docker desktop

👍 1

VenkyRules added 3 commits last month

```
Adding code to fetch more metadata from tableau and showing embedded …   d175277
removed comments                                                          c61dc05
removed extra lines                                                       dec71c3
```

vercel bot commented on Jan 20 • edited

Reviewers: maximearmstrong ✓

Assignees: No one assigned

Labels: integration: tableau

Projects: None yet

Milestone: No milestone

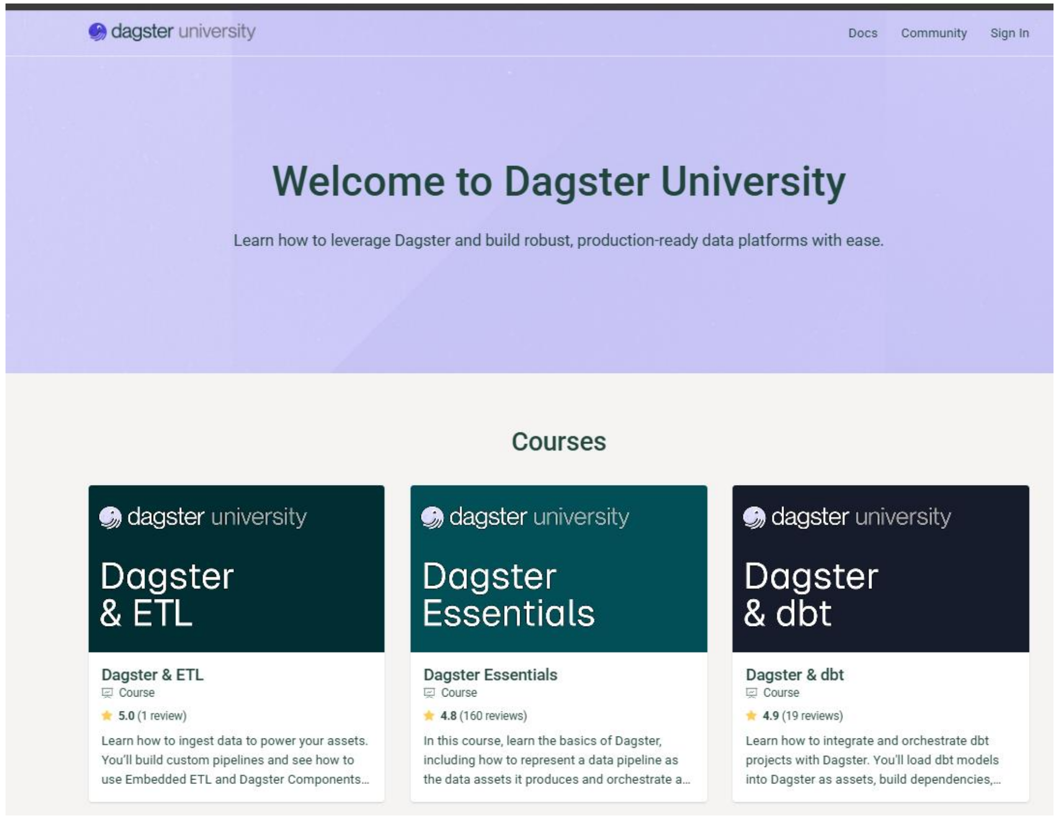
Development: Successfully merging this pull request may close these issues. None yet

2 participants

Building a data platform is team work

Further material

- public live stream about the basics: georgheiler.com/event/magenta-pixi-25
- n-depth Magenta implementation georgheiler.com/event/magenta-data-architecture-25
- https://courses.dagster.io/



Welcome to Dagster University

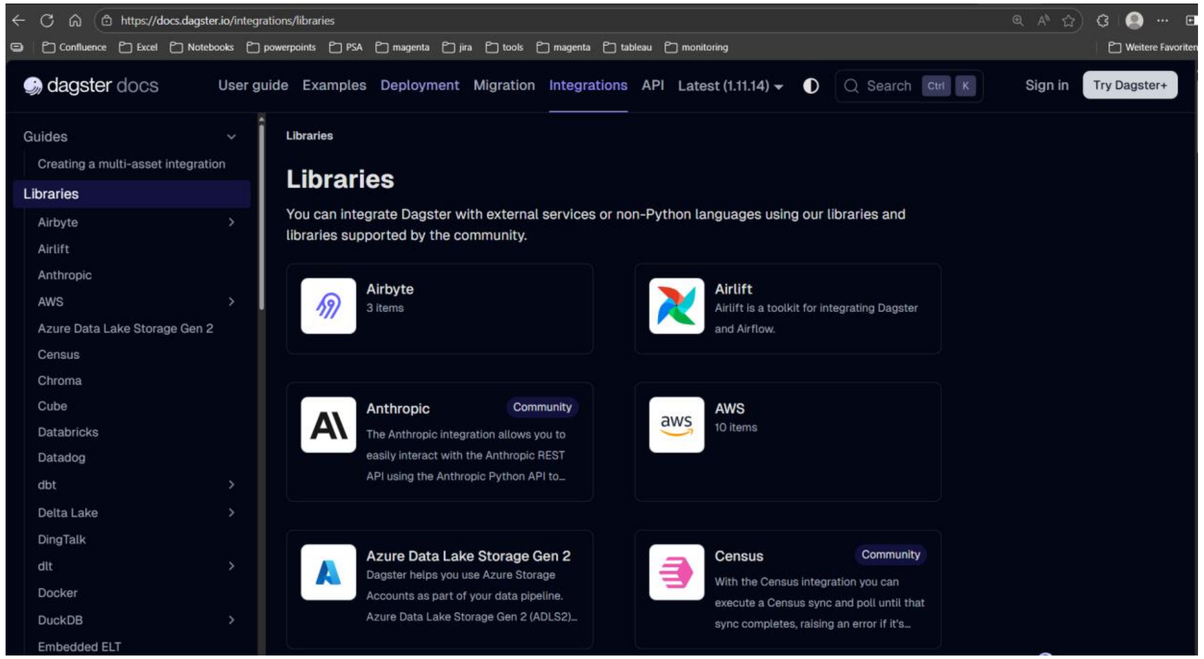
Learn how to leverage Dagster and build robust, production-ready data platforms with ease.

Courses

Dagster & ETL
Course
5.0 (1 review)
Learn how to ingest data to power your assets. You'll build custom pipelines and see how to use Embedded ETL and Dagster Components…

Dagster Essentials
Course
4.8 (160 reviews)
In this course, learn the basics of Dagster, including how to represent a data pipeline as the data assets it produces and orchestrate a…

Dagster & dbt
Course
4.9 (19 reviews)
Learn how to integrate and orchestrate dbt projects with Dagster. You'll load dbt models into Dagster as assets, build dependencies,…



Libraries

Libraries

You can integrate Dagster with external services or non-Python languages using our libraries and libraries supported by the community.

Airbyte
3 items

Airlift
Airlift is a toolkit for integrating Dagster and Airflow.

Anthropic Community
The Anthropic integration allows you to easily interact with the Anthropic REST API using the Anthropic Python API to…

AWS
10 items

Azure Data Lake Storage Gen 2
Dagster helps you use Azure Storage Accounts as part of your data pipeline. Azure Data Lake Storage Gen 2 (ADLS2)…

Census Community
With the Census integration you can execute a Census sync and poll until that sync completes, raising an error if it's…