Scaling data pipelines @Telekom

Aleks & Georg

Agenda

01 Scaling data usage within enterprise
Principles of scaling and historical problems

02 Technical foundation for scaling
Development process, Modeling, Governance

03 Dagster@Telekom
Advantages of asset-based data orchestration

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

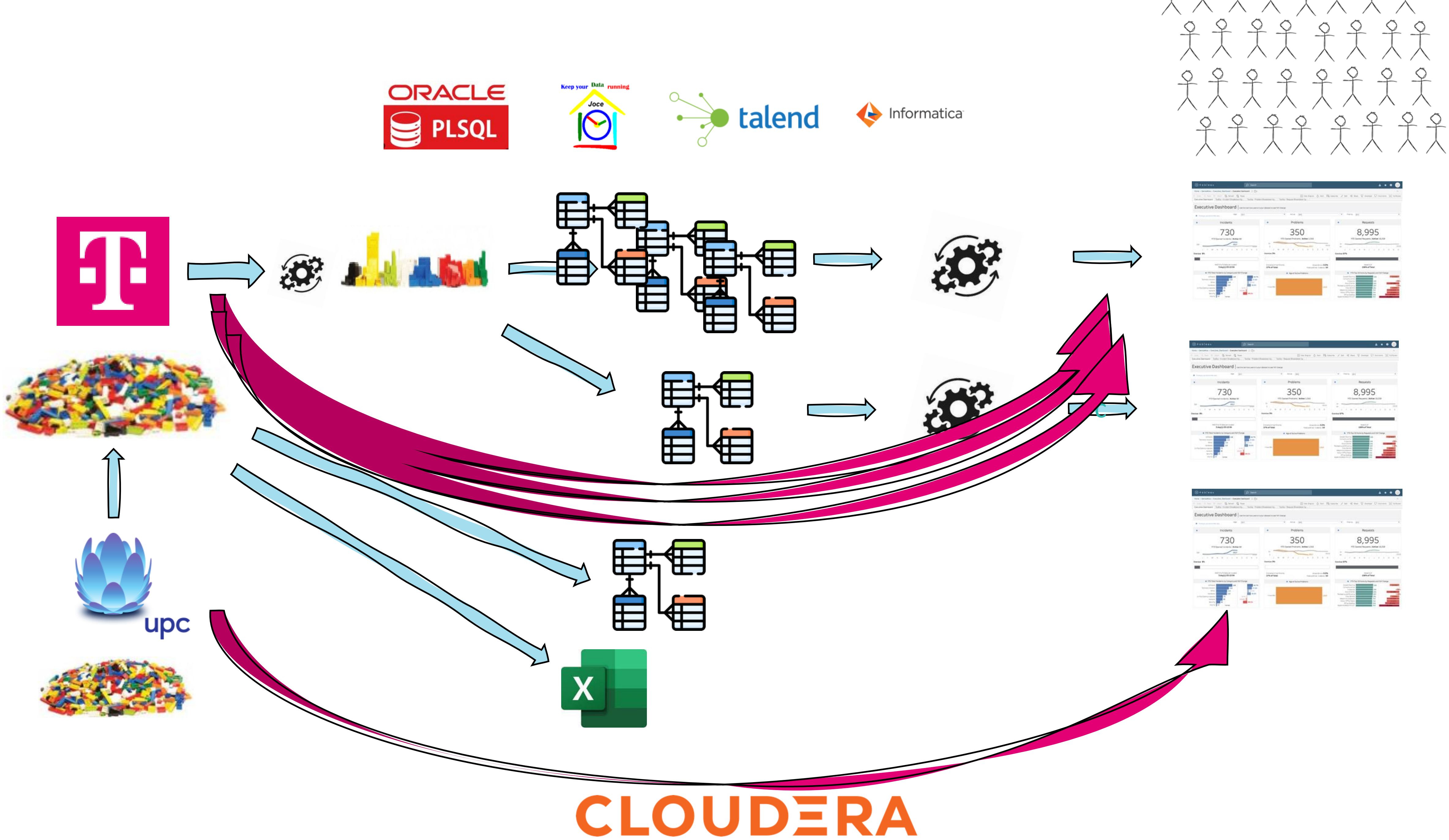
milicevica23

milicevica23

@milicevica23.bsky.social

[public live stream about the basics: georgheiler.com/event/magenta-pixi-25](https://yam-united.telekom.com/pages/data-magenta/apps/blog/updates/view/1496666c-c479-4e09-8ef5-0ebfe709e179)
https://yam-united.telekom.com/pages/data-magenta/apps/blog/updates/view/1496666c-c479-4e09-8ef5-0ebfe709e179

There is a chaos out there


ORACLE
PLSQL
Keep your Data running
Joce
talend
Informatica
Executive Dashboard
730
350
8,995
upc
CLOUDERA

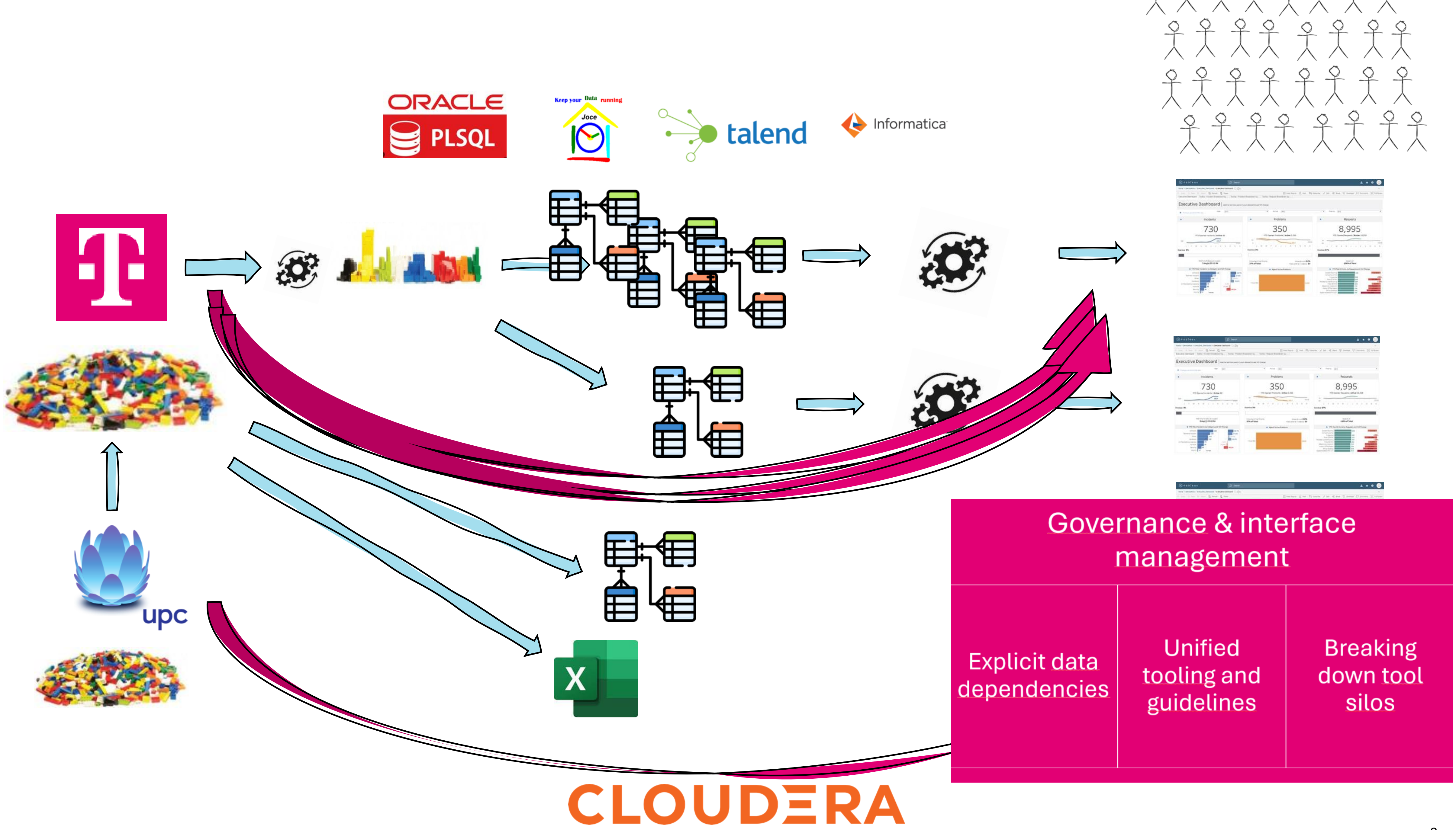
How did we end here? Time!

business grows (merger)
demand for data grows
methodology and tooling changes

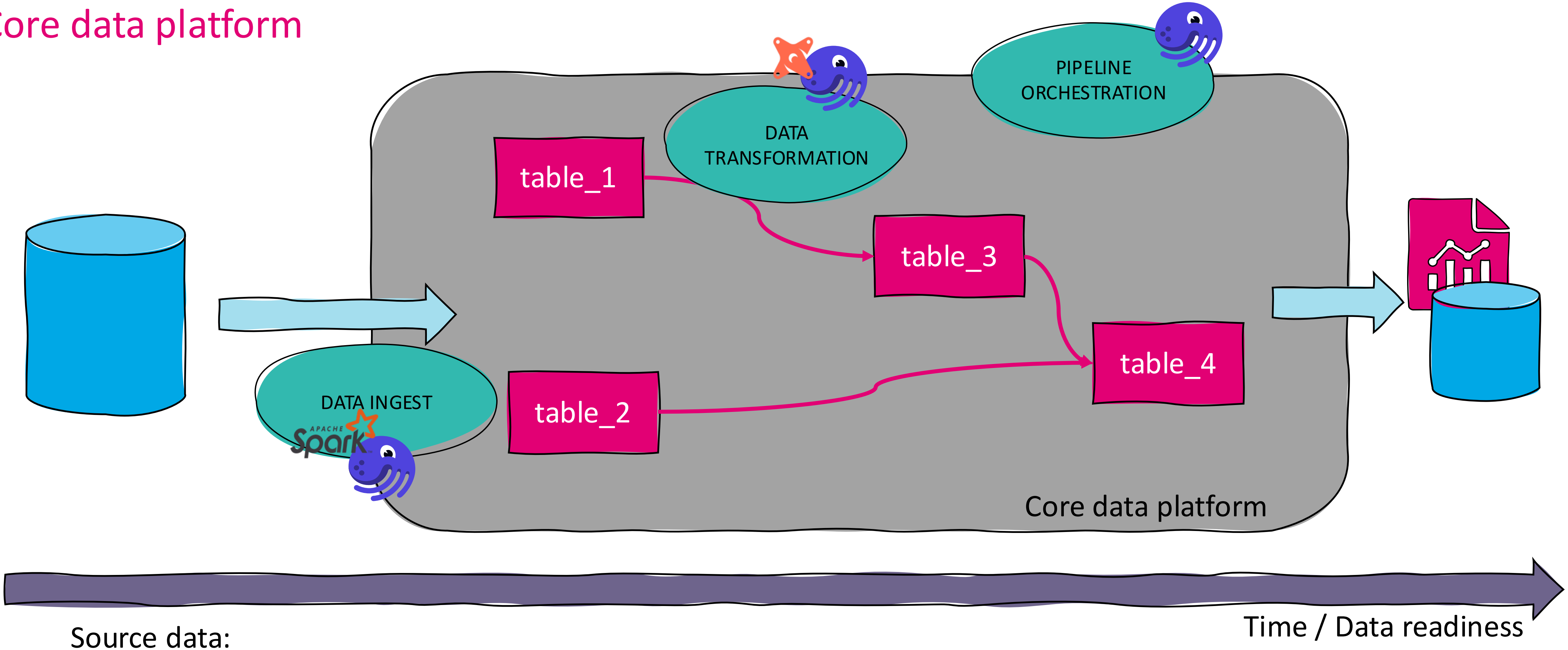
- Missing lineage
- Missing semantics
- Missing collaboration
- High lead times
- Limited quality

Governance & interface management

Explicit data dependencies	Unified tooling and guidelines	Breaking down tool silos


ORACLE
PLSQL
Keep your Data running
Joce
talend
Informatica
Executive Dashboard
730
350
8,995
upc
Governance & interface management
Explicit data dependencies
Unified tooling and guidelines
Breaking down tool silos
CLOUDERA

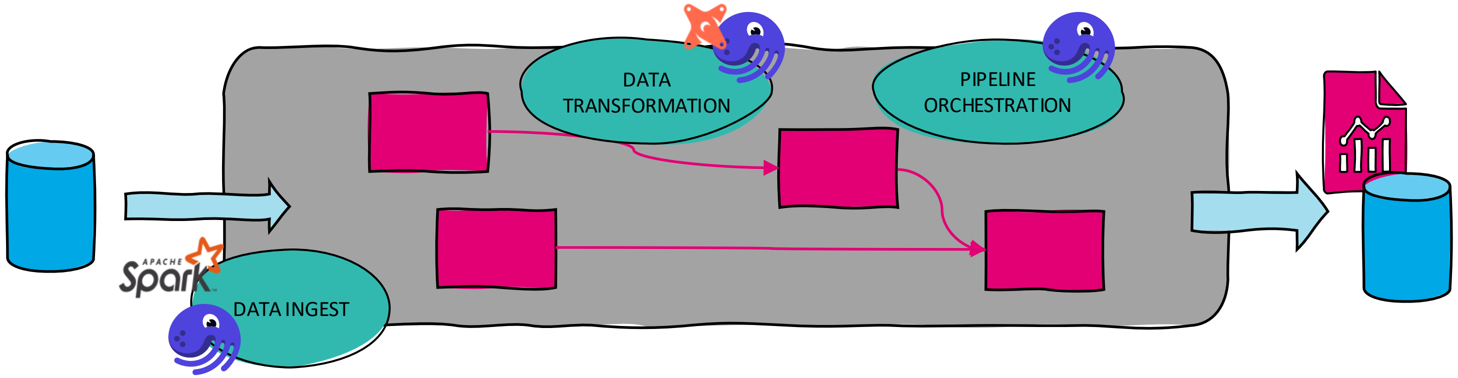
Core data platform



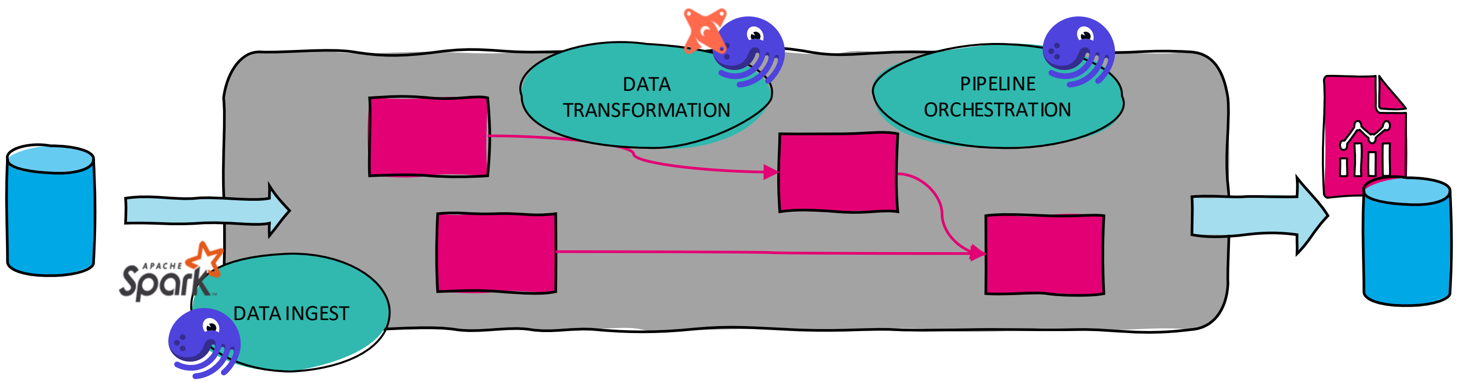
Source data:

- Kafka
- Files
- Database systems

I fear that what we build is very hard to push into the business units



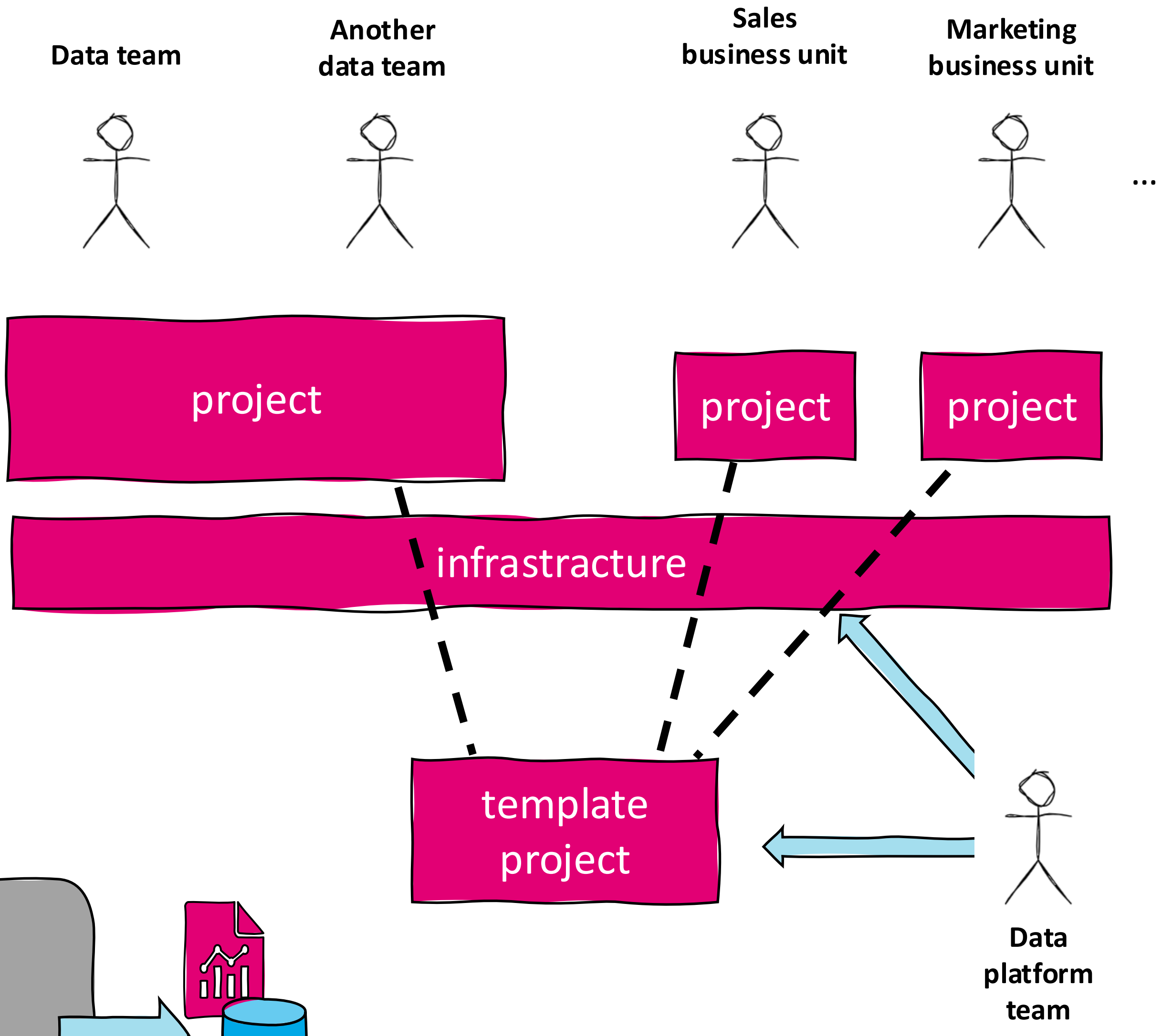
I ~~fear~~ know that what we build is very ~~hard~~ easy to push into the business units

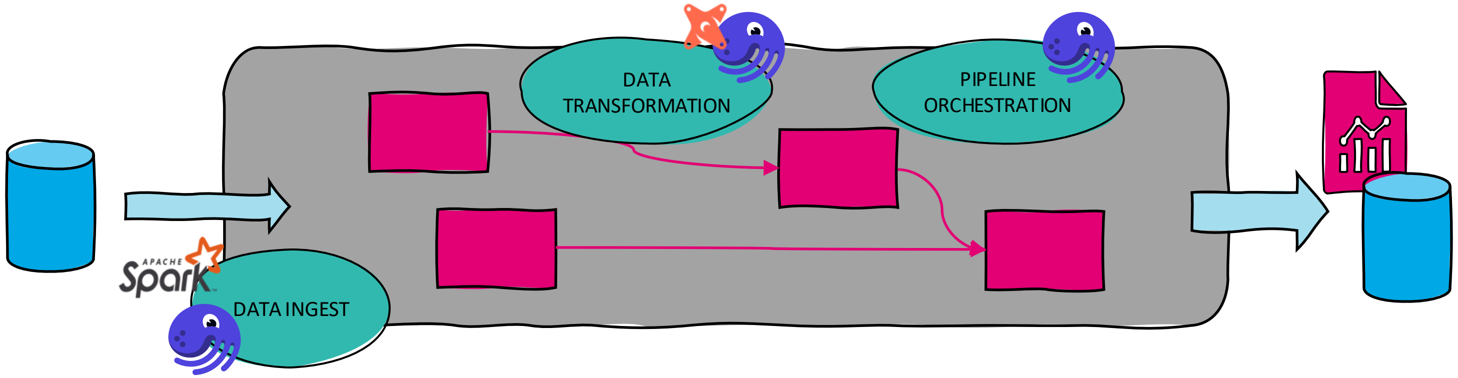


Observation

- Process is straight forward: ingest, transform, use
- Everything we do - we do for business to provide better service
- Hard to scale across company

- Dividing people into **develop framework** and **use framework** groups
- Thinking in a **building block** structure
- Introduce modern tooling supporting software engineering practices: **dbt, dagster, pixi, docker**
- Introduce **new processes**, **modeling** and **metadata tooling** for better governance



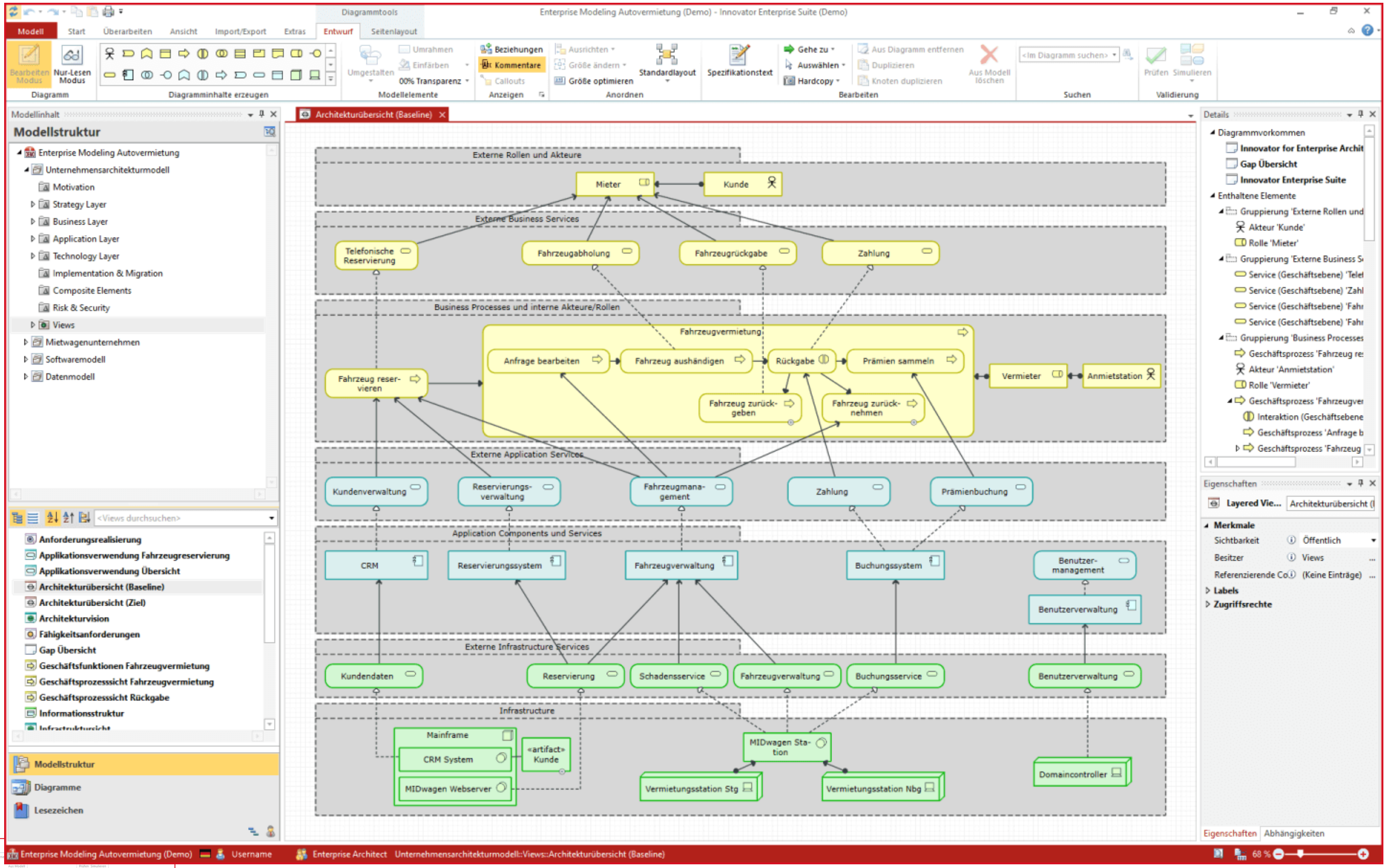


Having a data platform team doesn’t mean that your platform scales

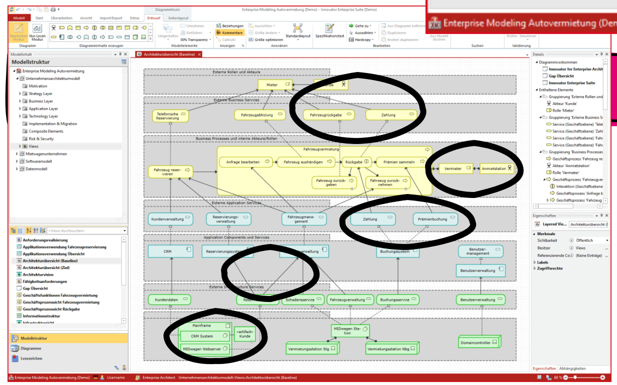
1. Building block with governance, modeling and software-engineering principles
2. Understanding data platform vendor war
3. Understanding tool silos

Break the silos with the building block

Data team

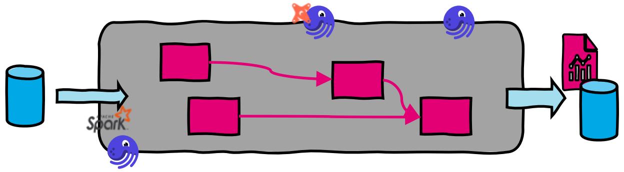


Sales business unit



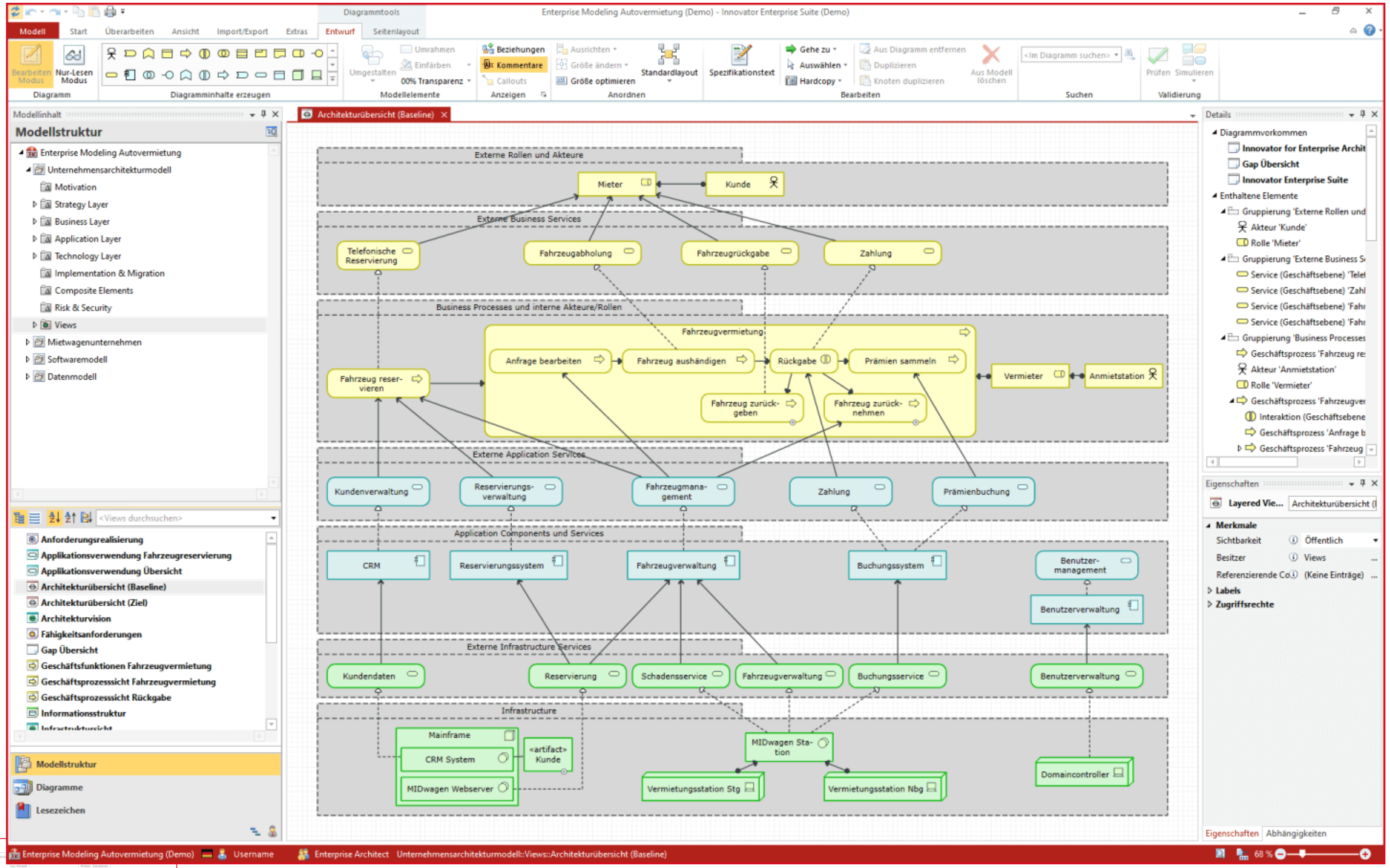
infrastructure

template project

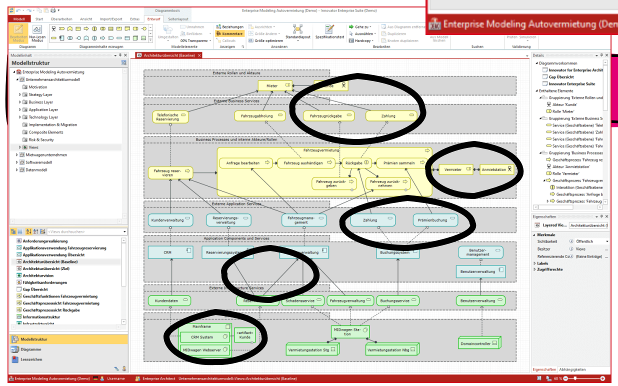


Break the silos with the building block

Data team

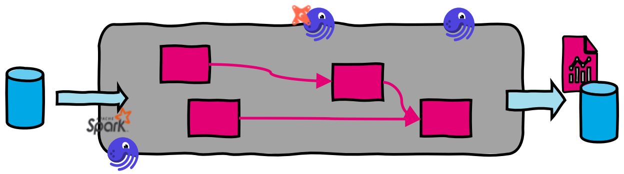


Sales business unit

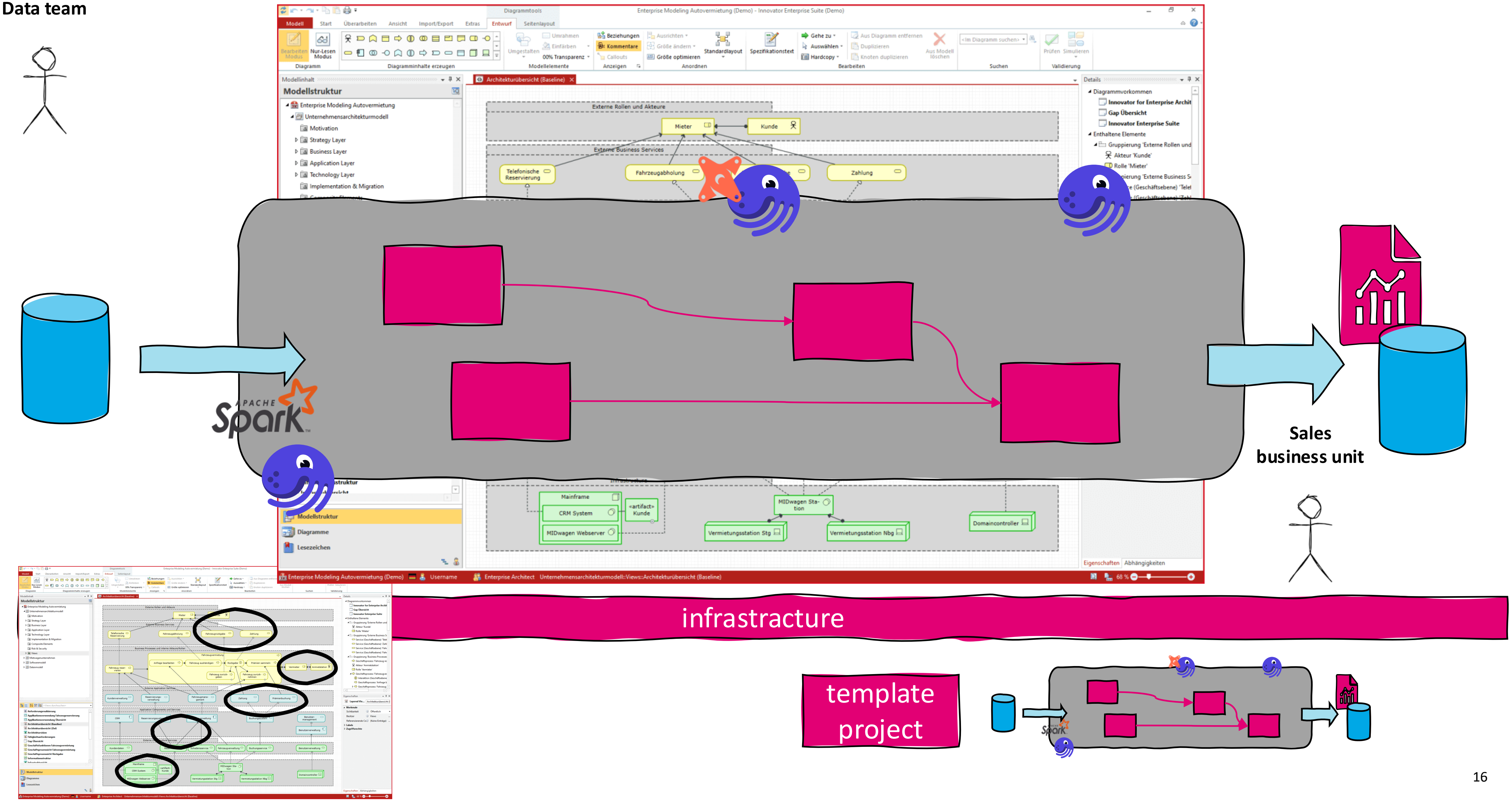


infrastructure

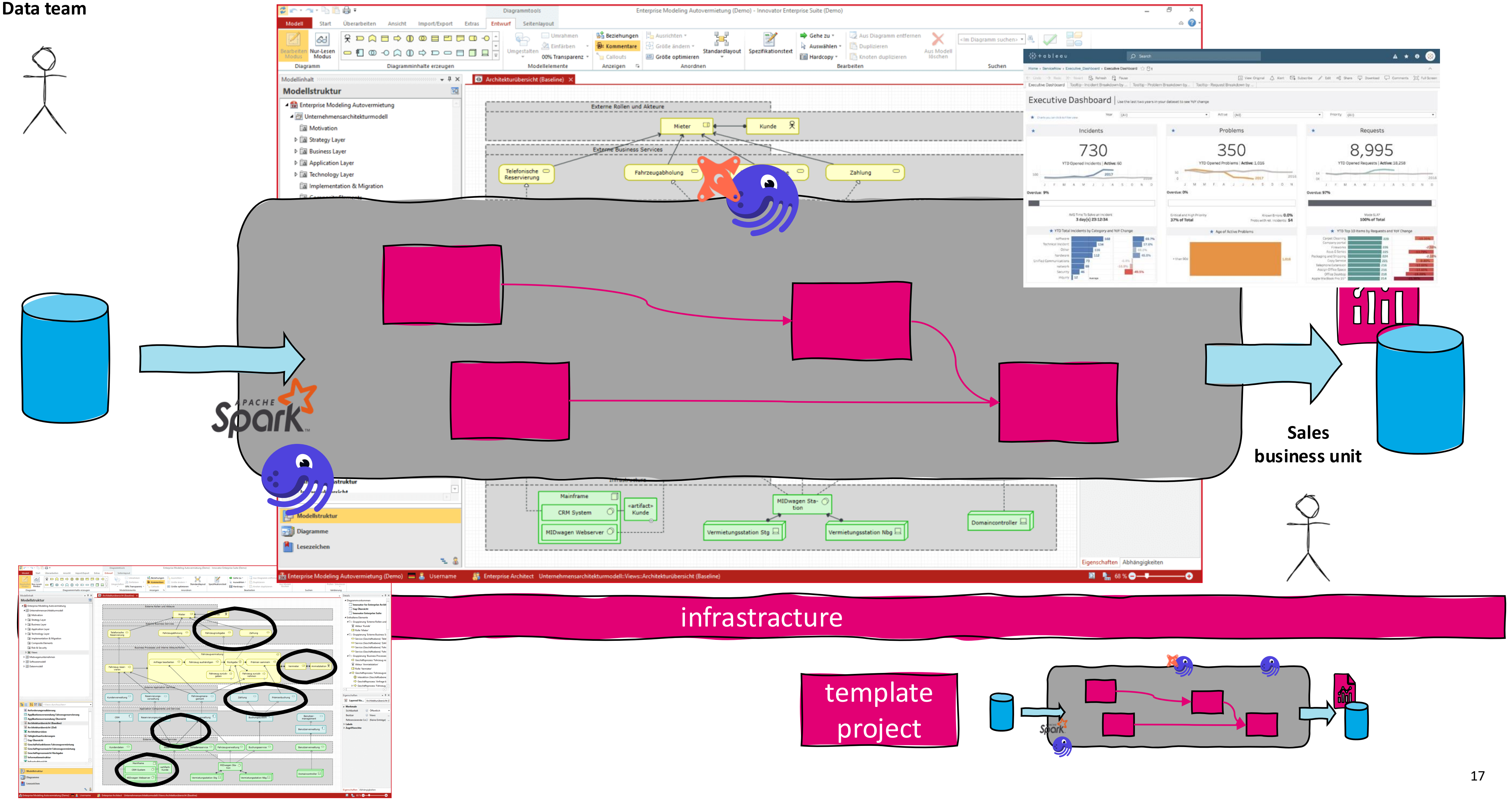
template project



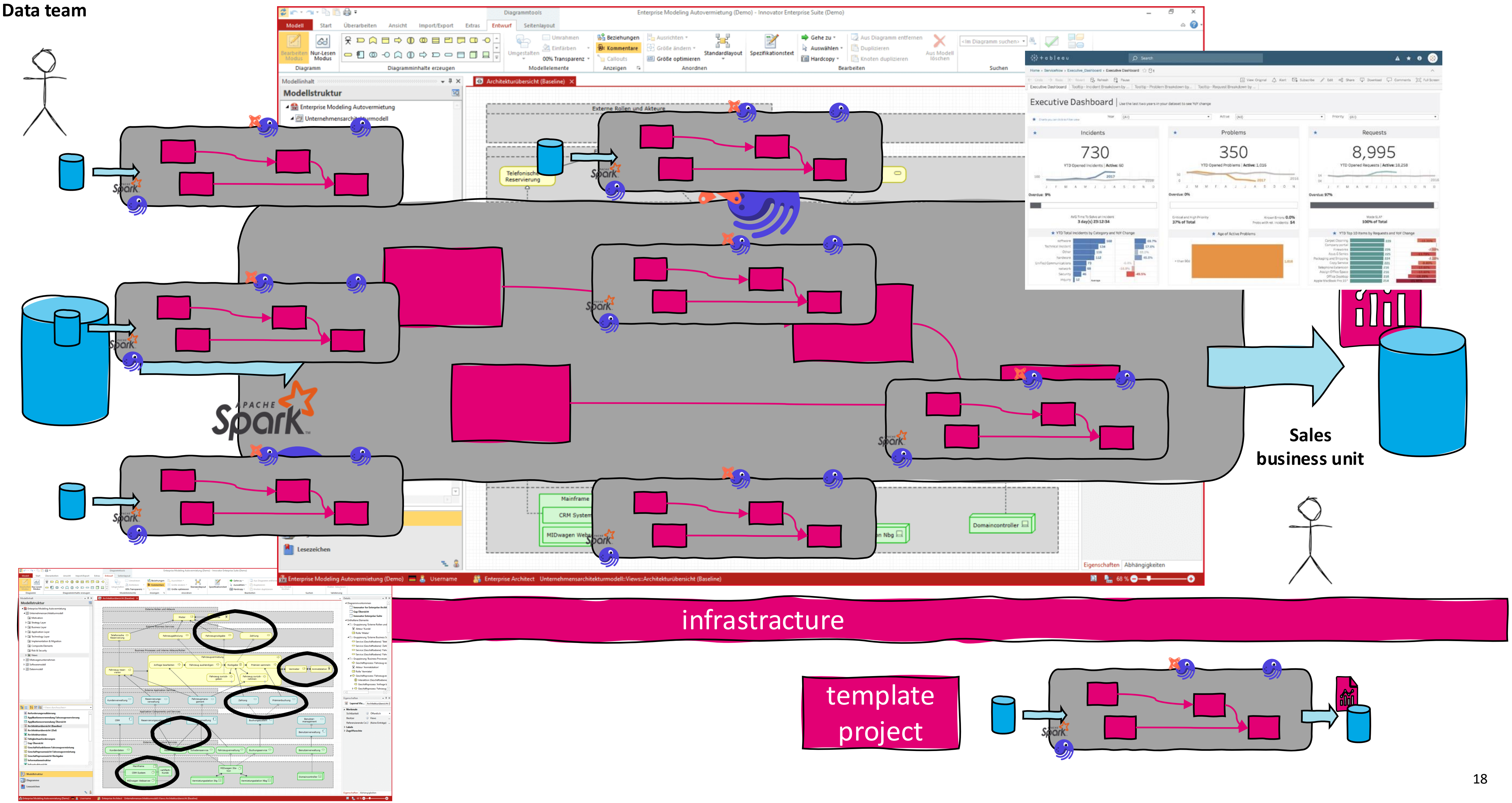
Break the silos with the building block



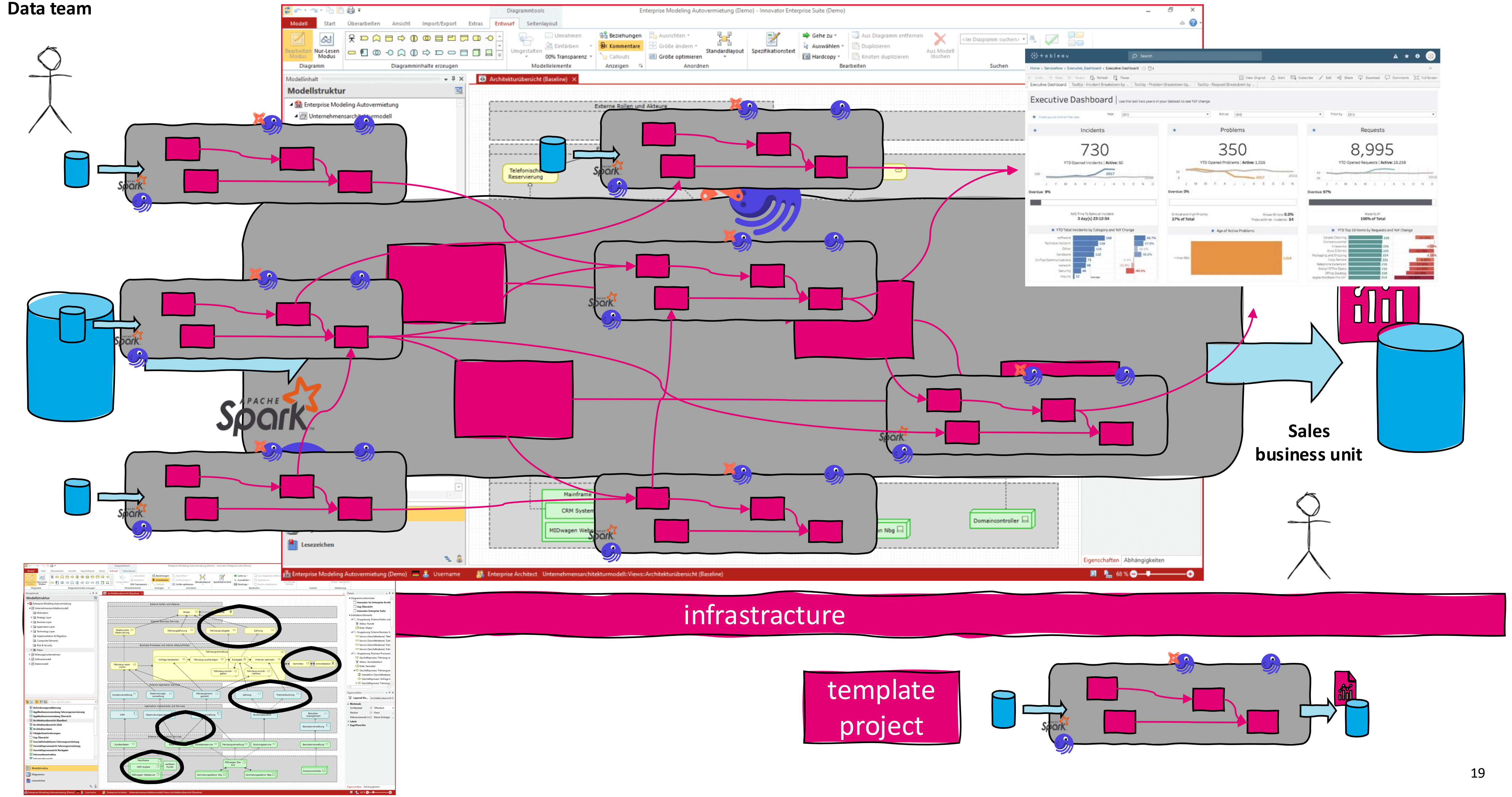
Break the silos with the building block



Break the silos with the building block



Break the silos with the building block

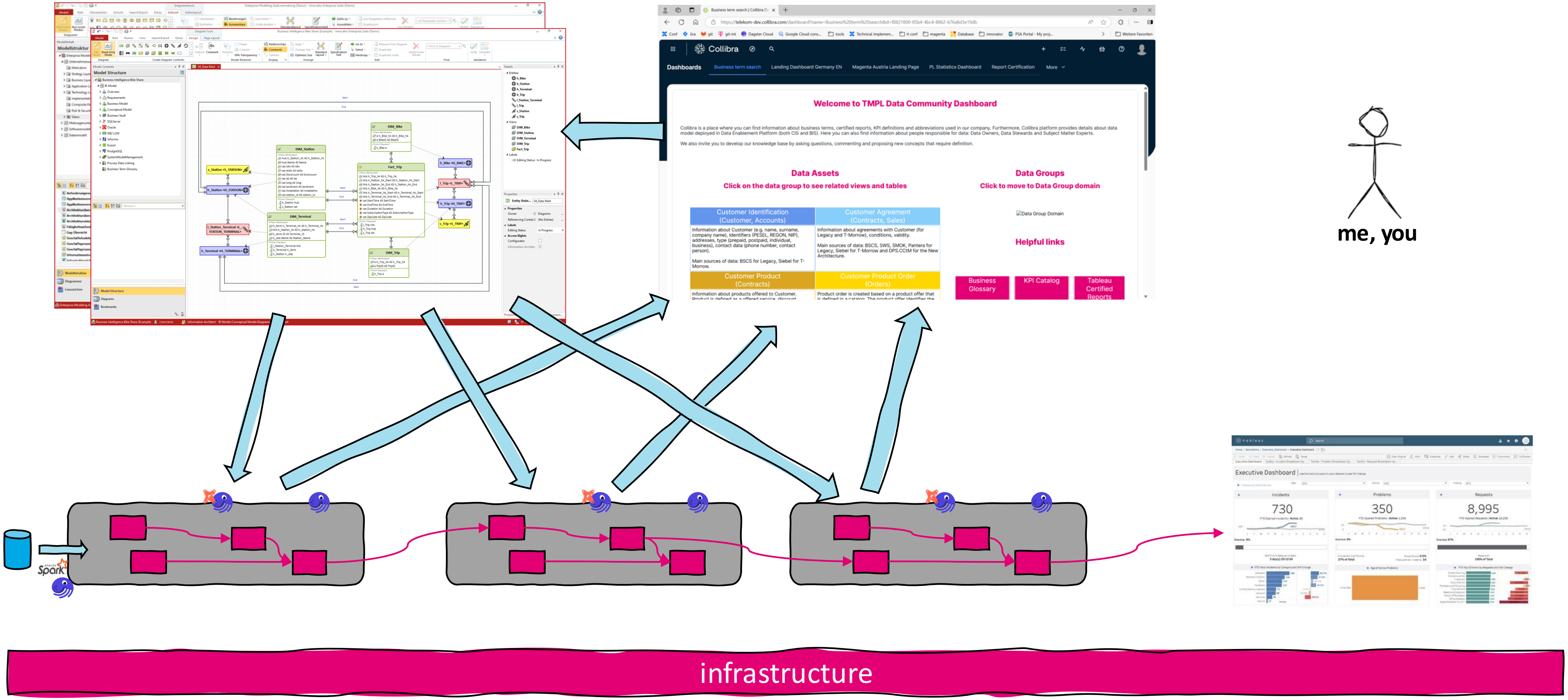


Multi-project setup is a challenge!

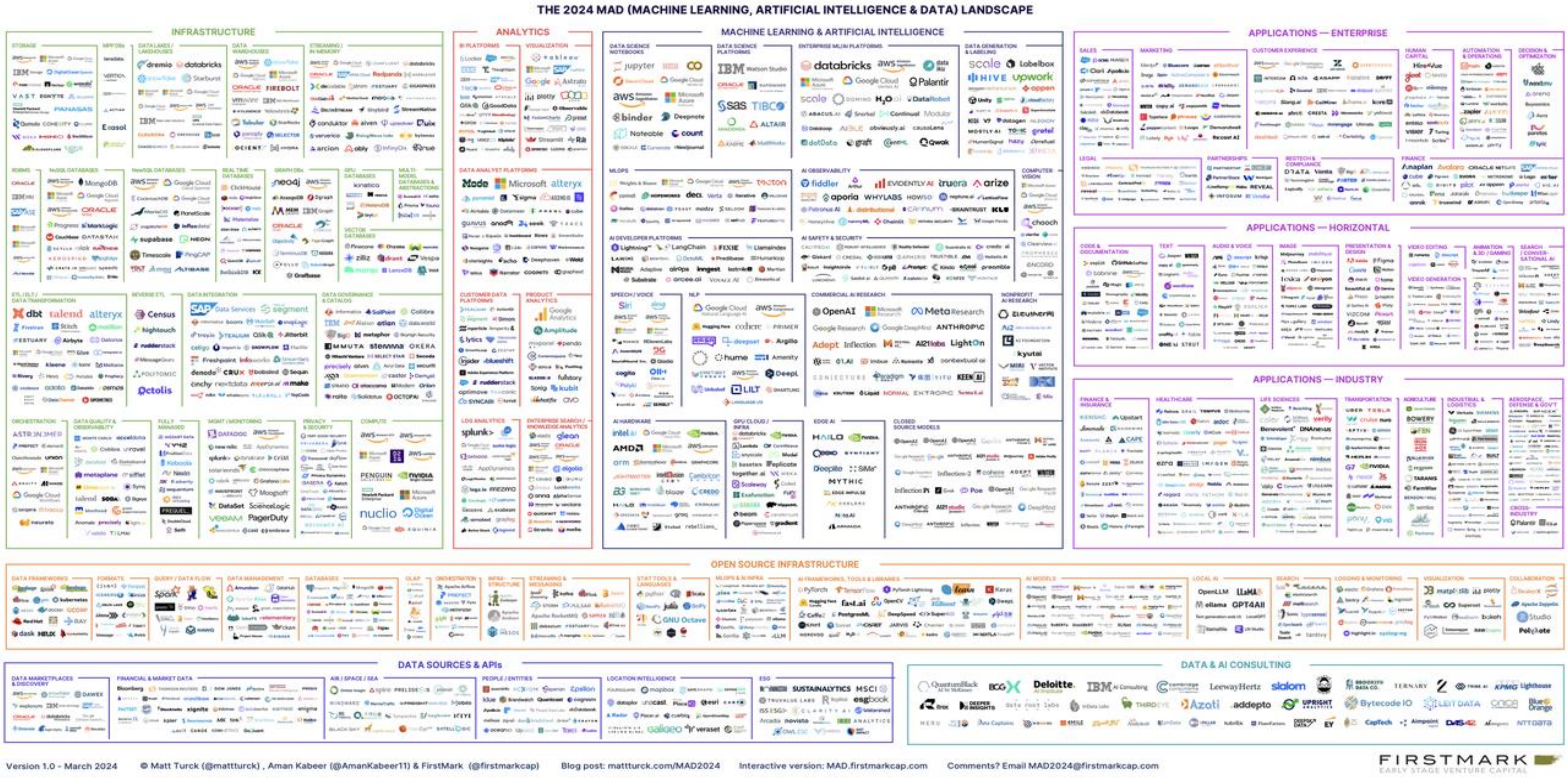
1. Dedicated team maintaning infrastructure and template project
2. Governance and modeling tools

Modeling and governance are keys for success

Development phase **Exploration phase**



Understanding data platform vendor war



execution engine

Cloud:

Engine:

Google BigQuery

DevEnv:

jupyter jupyter jupyter

Orc:

- Full stack offered by one vendor
- E2E integration lock in
- Deployment is always a workaround
- No SWE, no local development
- Orchestrator is second class citizen and always task based

+ Frontier in the lakehouse approach
+ Notebooks environment is very convenient
+ Everything on one place

sql transformation framework

dbt SQLMesh

- Orchestration is just for **DWH/SQL** part of the platform

+ Frontier in the SQL development
+ SWE for DWH development

orchestration engine

dbt

MAGE

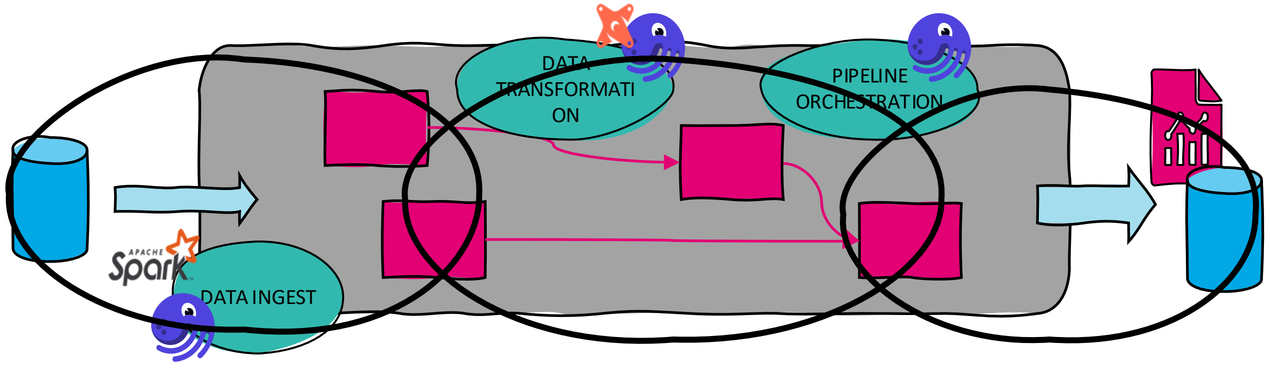
- More technical knowledge needed to setup and use correctly

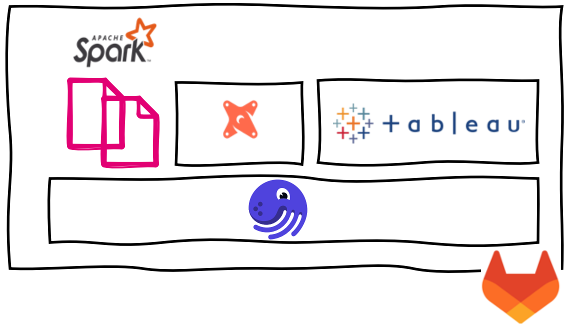
+ Integration is important
+ full SWE and local development

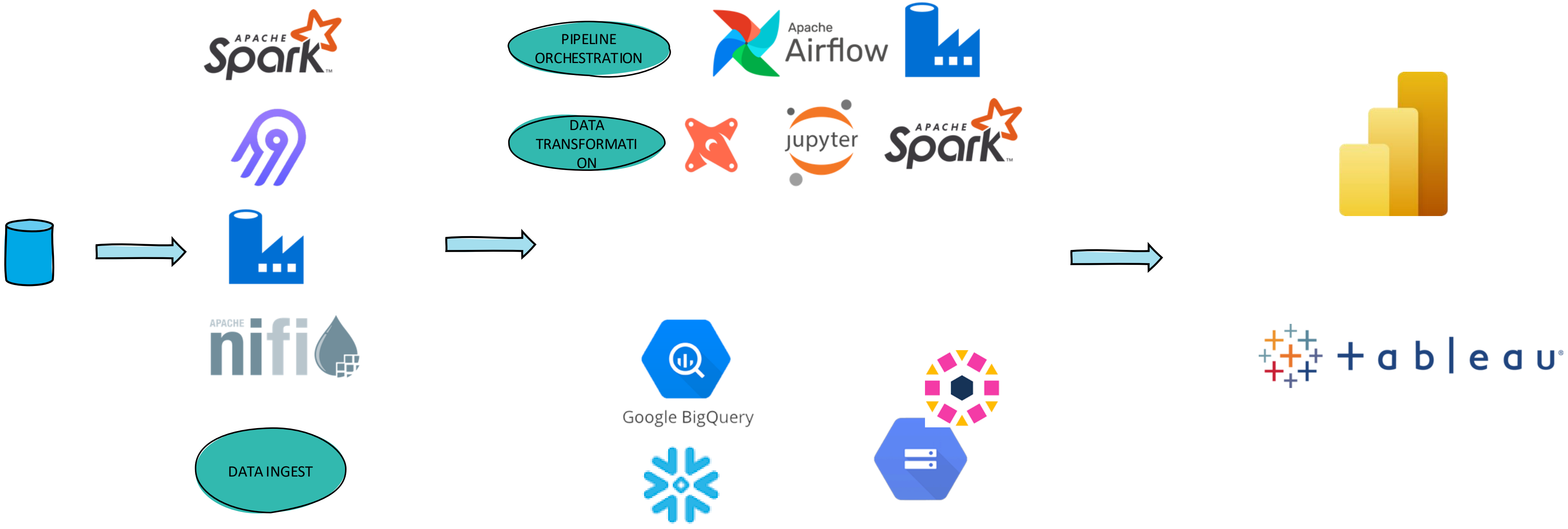
Understanding tool silos

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

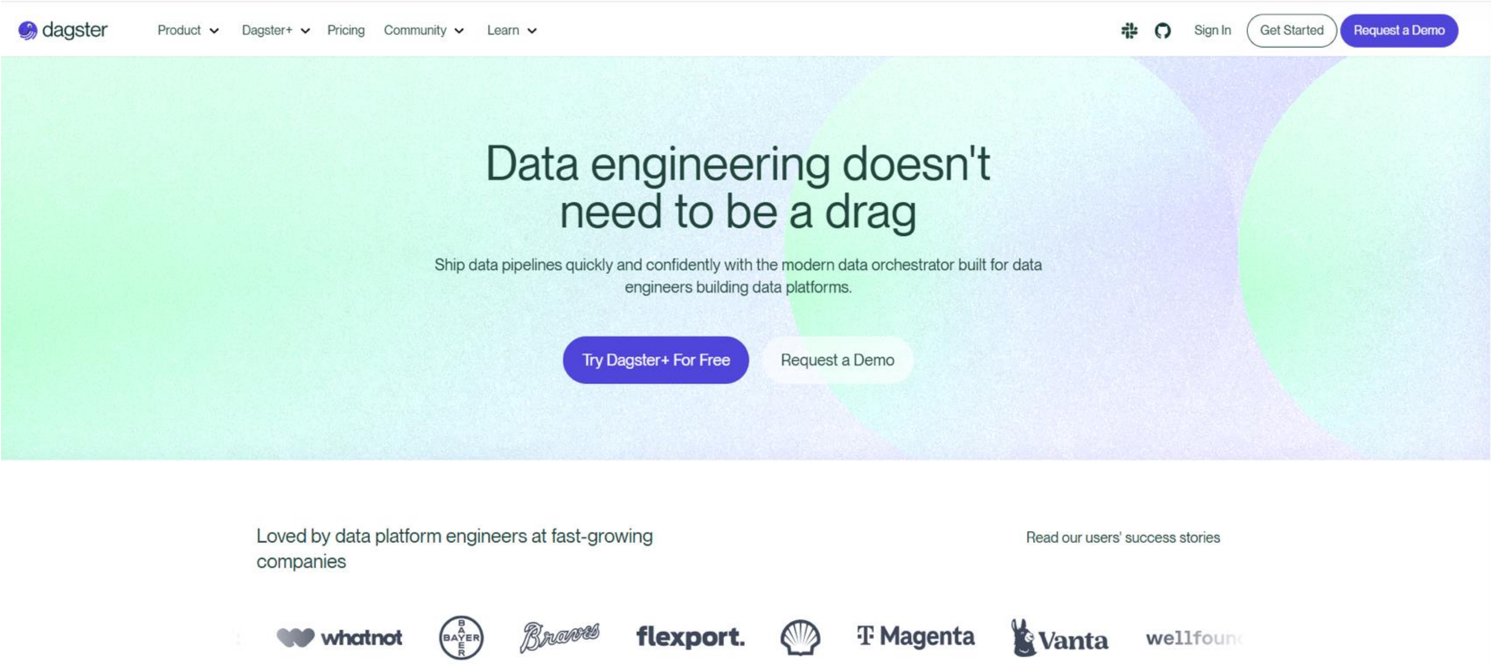
Developer







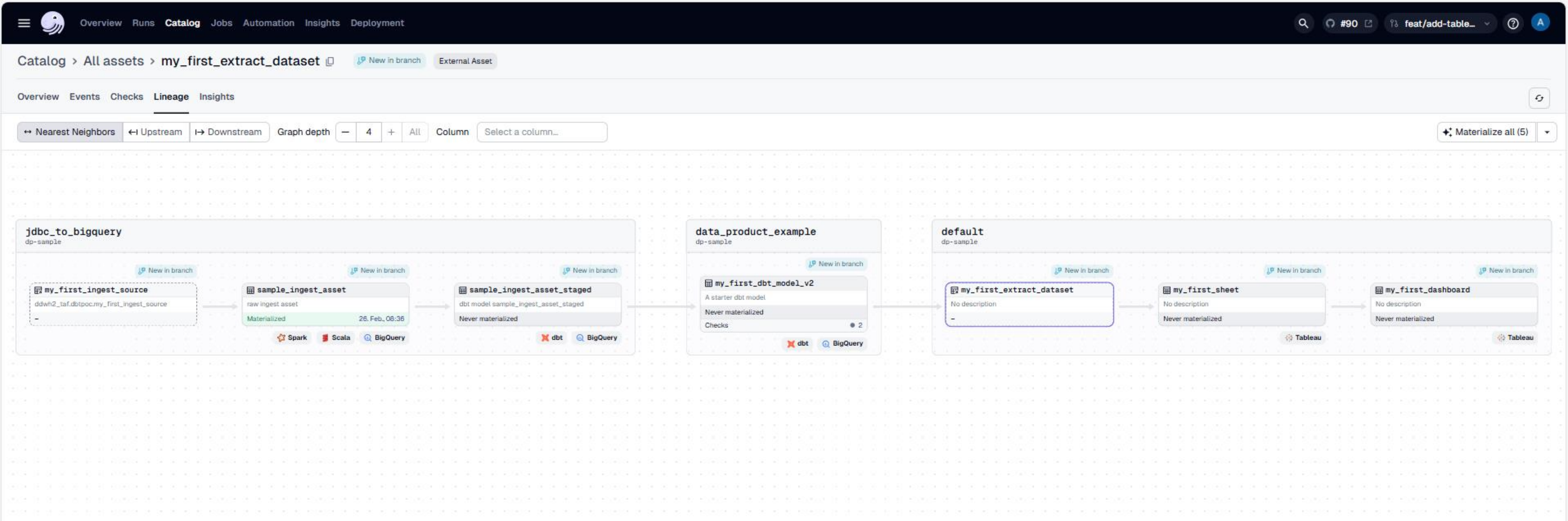
Dagster as the core of the platform



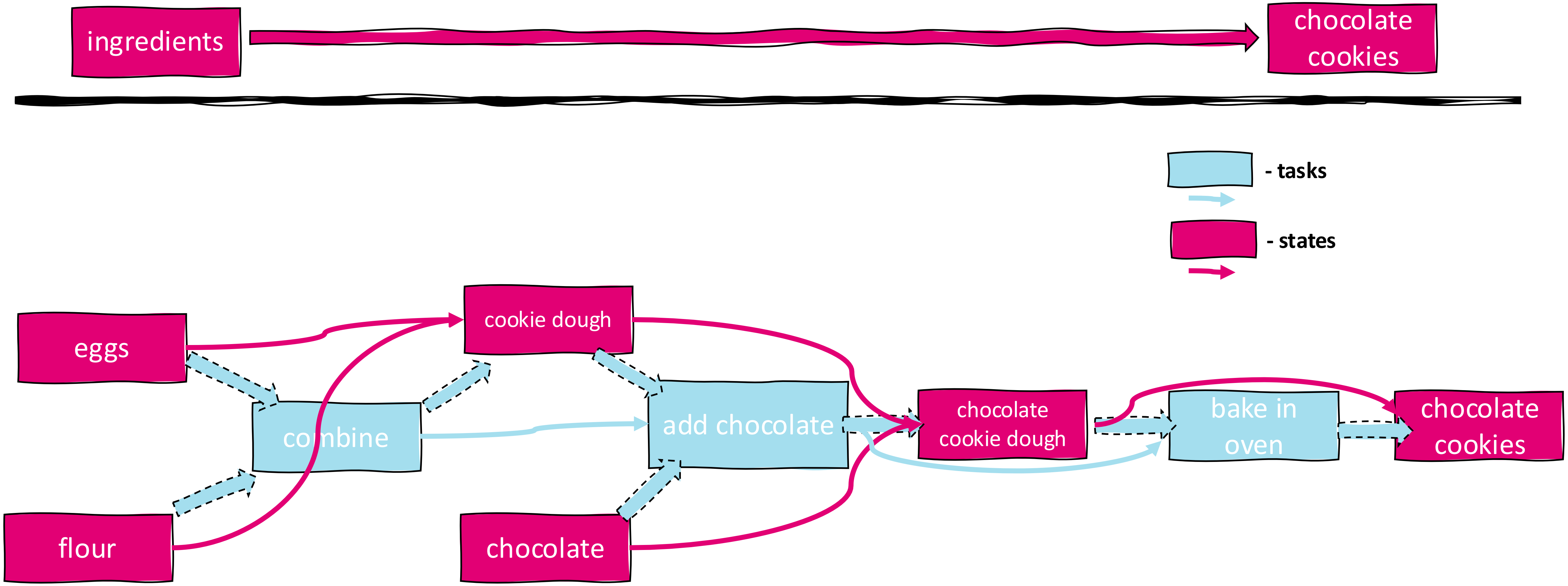
- at Magenta we decided to build around the orchestrator and not around a execution engine
  - hybrid deployment – controlplane SaaS – runtime in our k8s
  - software engineering best practices for project development and deployment
  - asset-based mindset for data flows (graph like a calculator for data dependencies)
- new concepts in orchestration

New enabled concepts

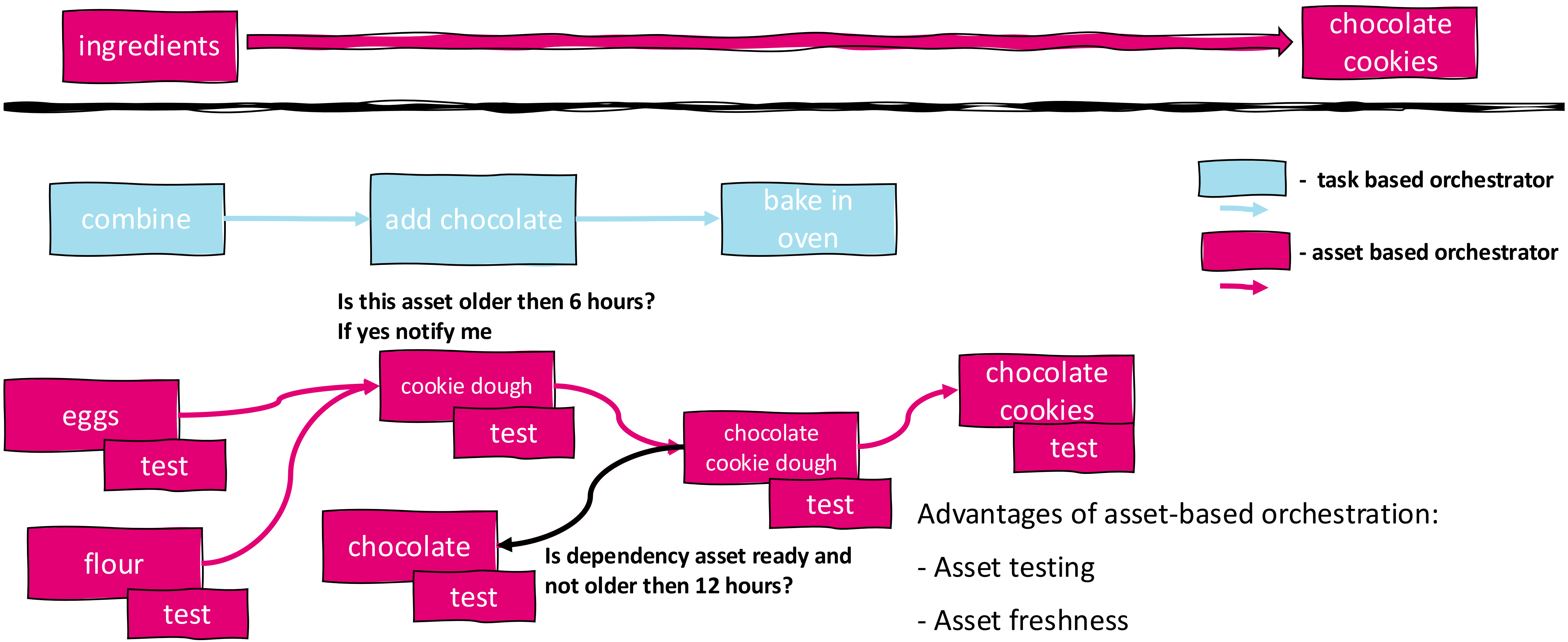
- Asset based graph
- Metadata driven pipeline creation
- Reusable Components
- ....



Asset and Task based orchestration: Chocolate cookie example



Asset based orchestration

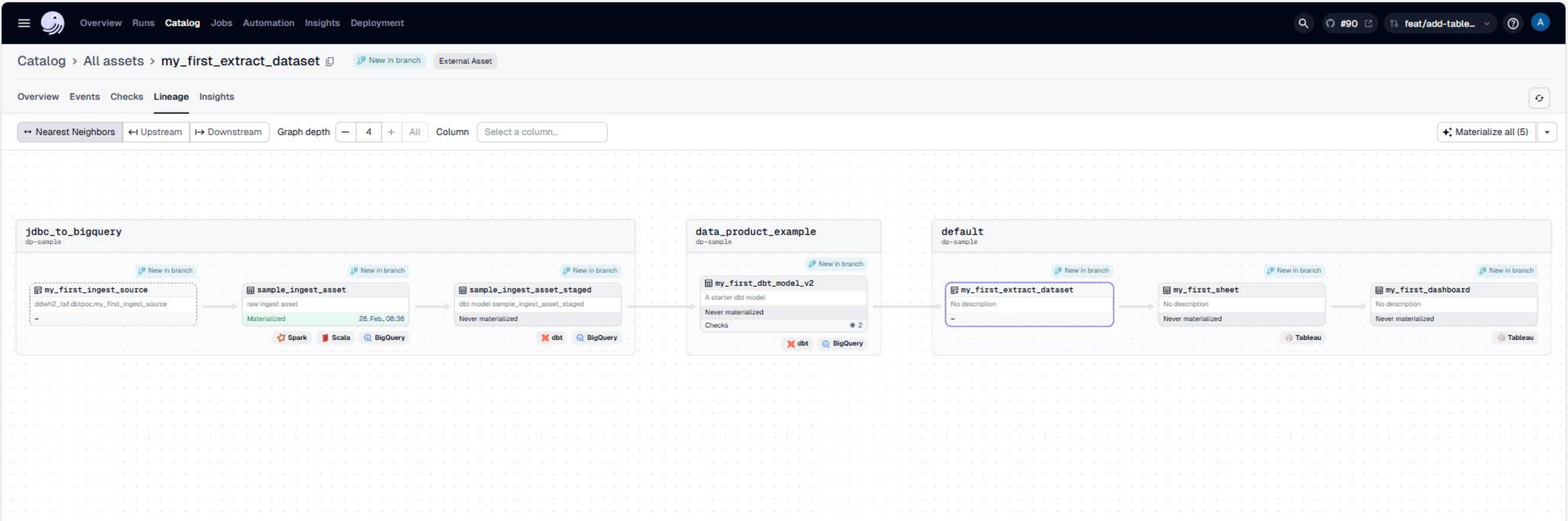


Advantages of asset-based orchestration:

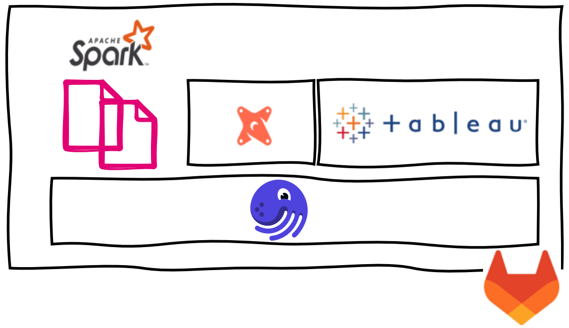
- Asset testing
- Asset freshness
- Asset dependecy graph with granular declarative scheduling approach

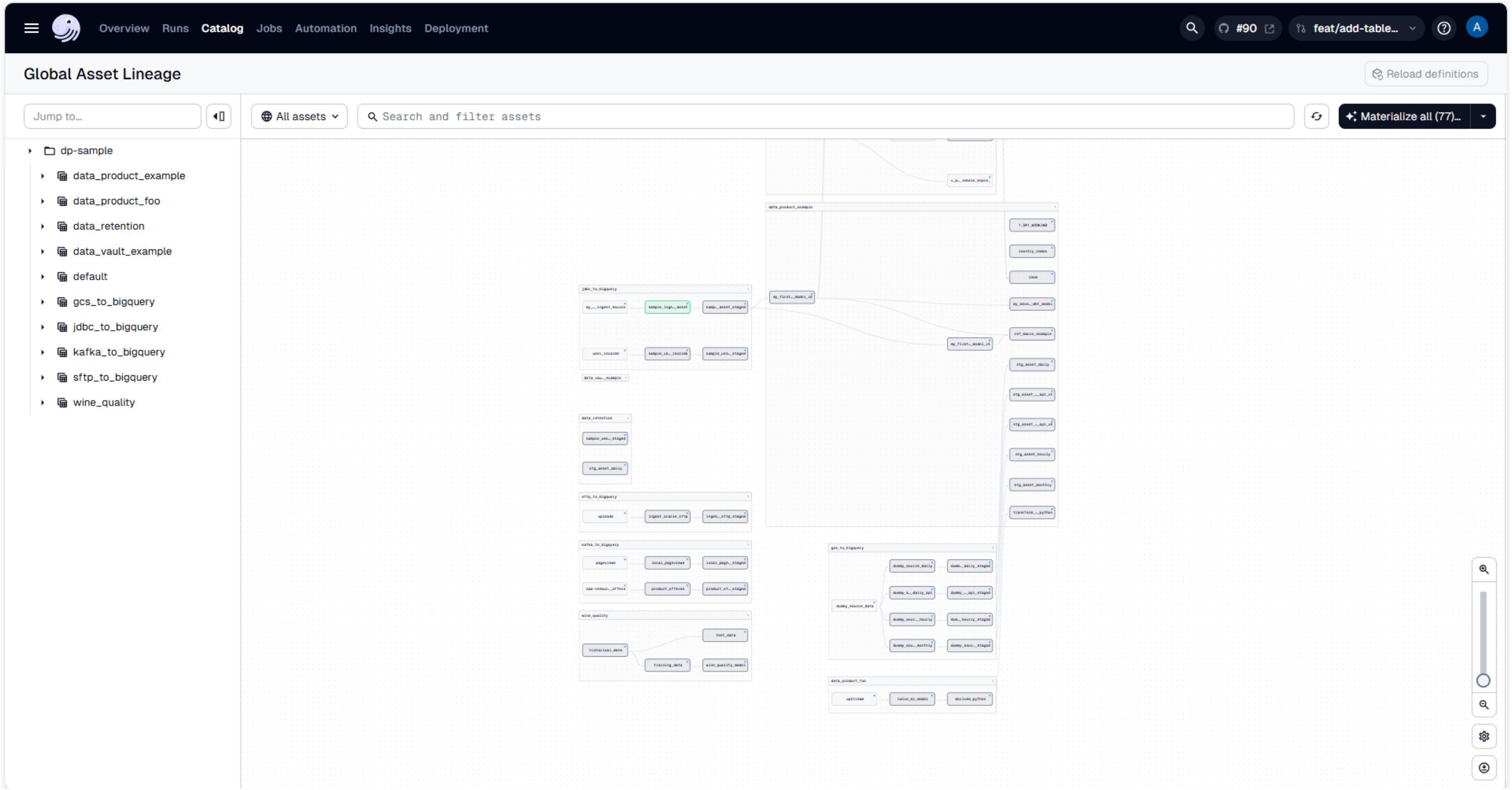
New enabled concepts

- Asset based graph
- Metadata driven pipeline creation
- Reusable Components
- ....

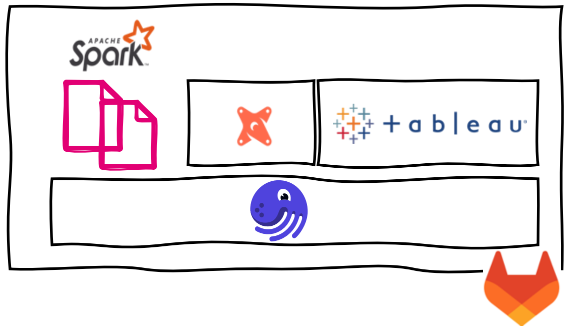


Machine-readable metadata pipeline generation

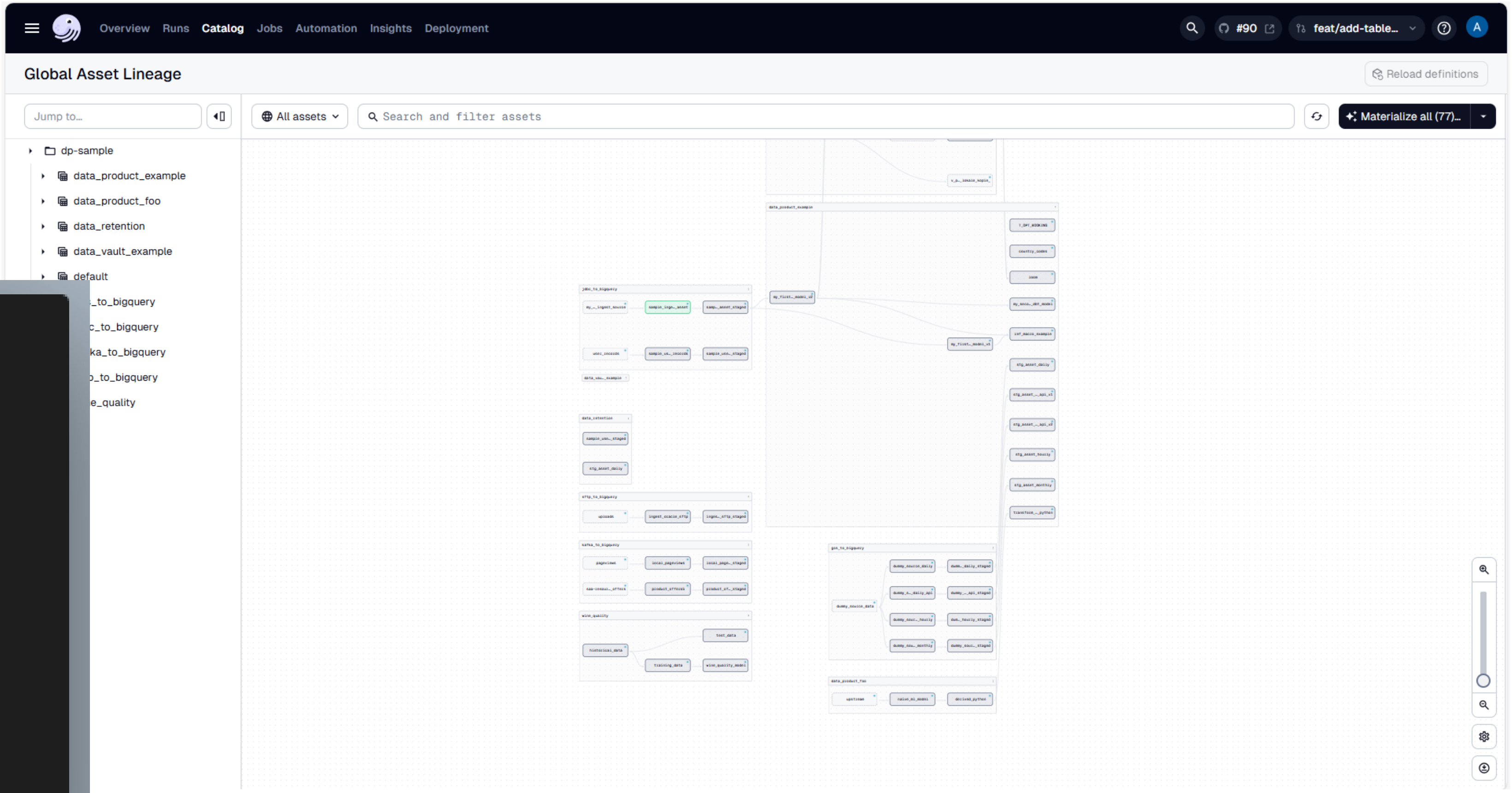




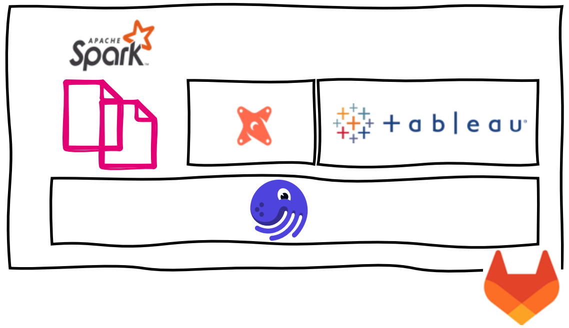
Machine-readable metadata pipeline generation



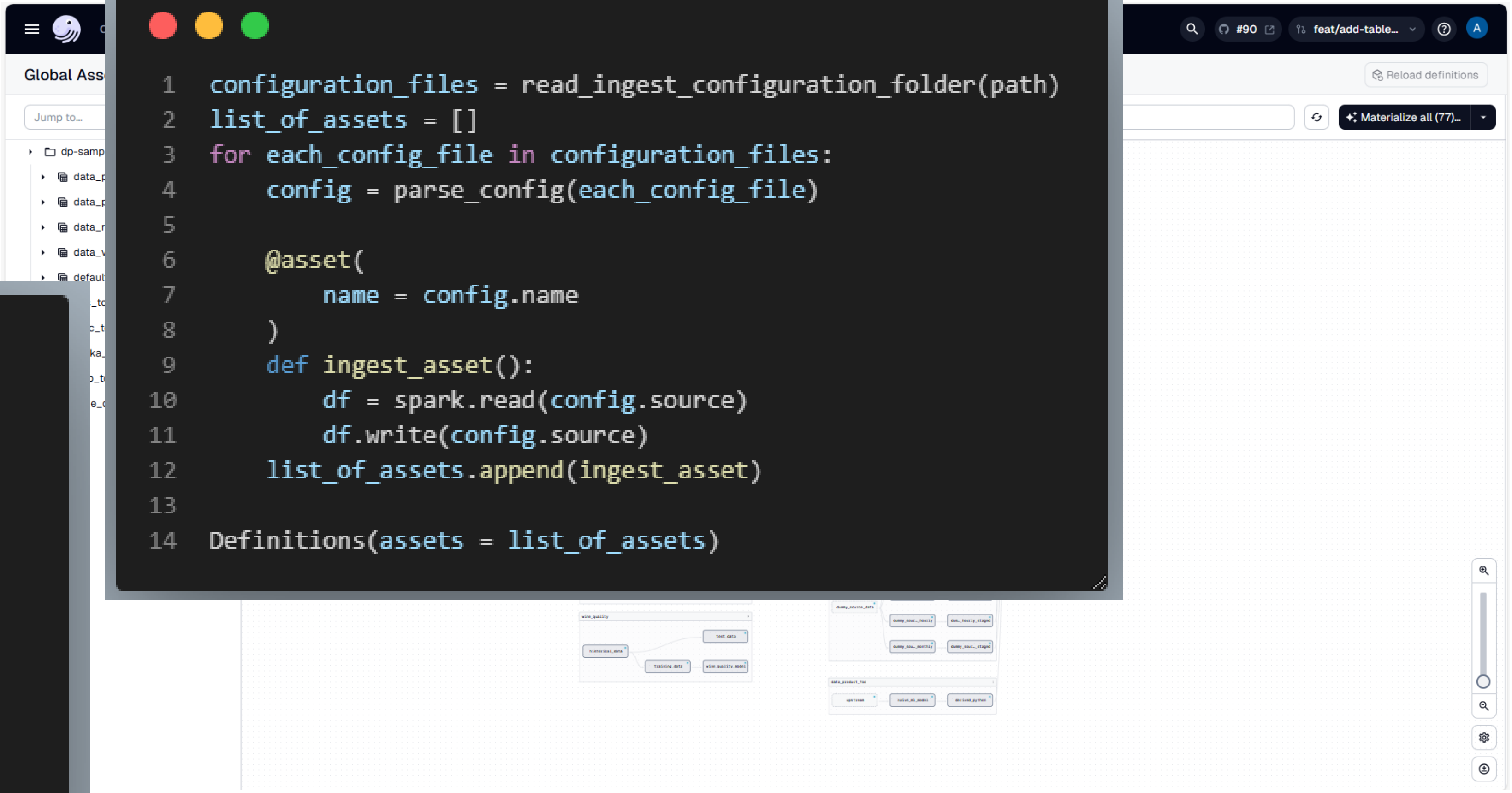

```python
metadata = collect_metadata()

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)
```



Machine-readable metadata pipeline generation

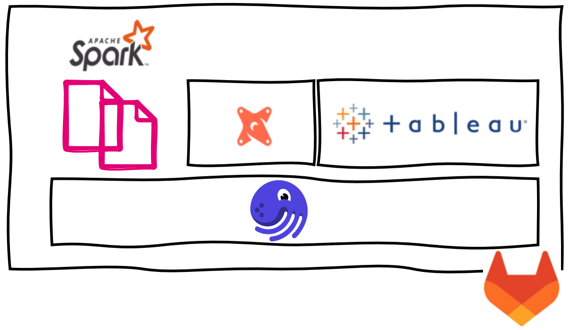



```python
metadata = collect_metadata()

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
configuration_files = read_ingest_configuration_folder(path)
list_of_assets = []
for each_config_file in configuration_files:
    config = parse_config(each_config_file)

    @asset(
        name = config.name
    )
    def ingest_asset():
        df = spark.read(config.source)
        df.write(config.source)
    list_of_assets.append(ingest_asset)

Definitions(assets = list_of_assets)
```

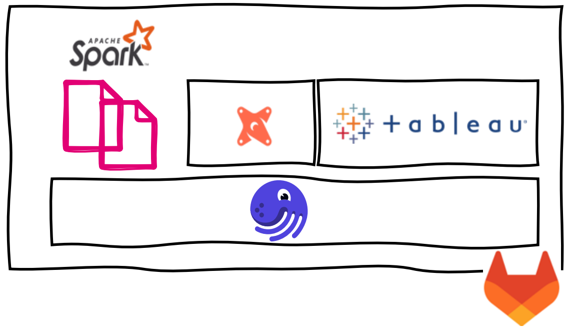
Machine-readable metadata pipeline generation




```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
manifest = read_dbt_manifest(path)
list_of_assets = []
for each_node in manifest:
    model_name = each_node.name
    model_deps = each_node.deps
    @asset(
        name = model_name,
        deps = model_deps
    )
    def run_dbt_asset(dbt: DbtClient)
        dbt.run(f"--select {model_name}")

    list_of_assets.append(run_dbt_asset)

Definitions(assets = list_of_assets)
```

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

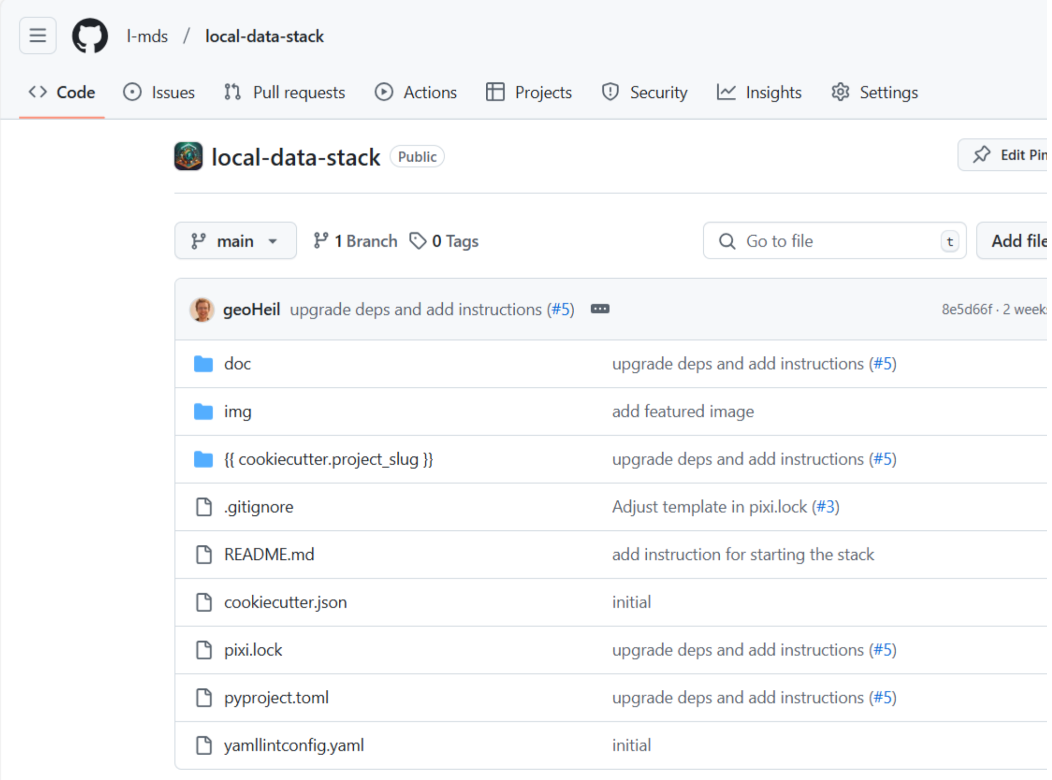
- Define once, test & reuse
- Resources → Encapsulate complex logic to interact with external systems
- IO manager → Make complex IO interactions substitutable & testable
- Benefits
  - Dependency injection
  - Day 1 productivity: Scale the data pipeline down to a single laptop
  - Increase self-service: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

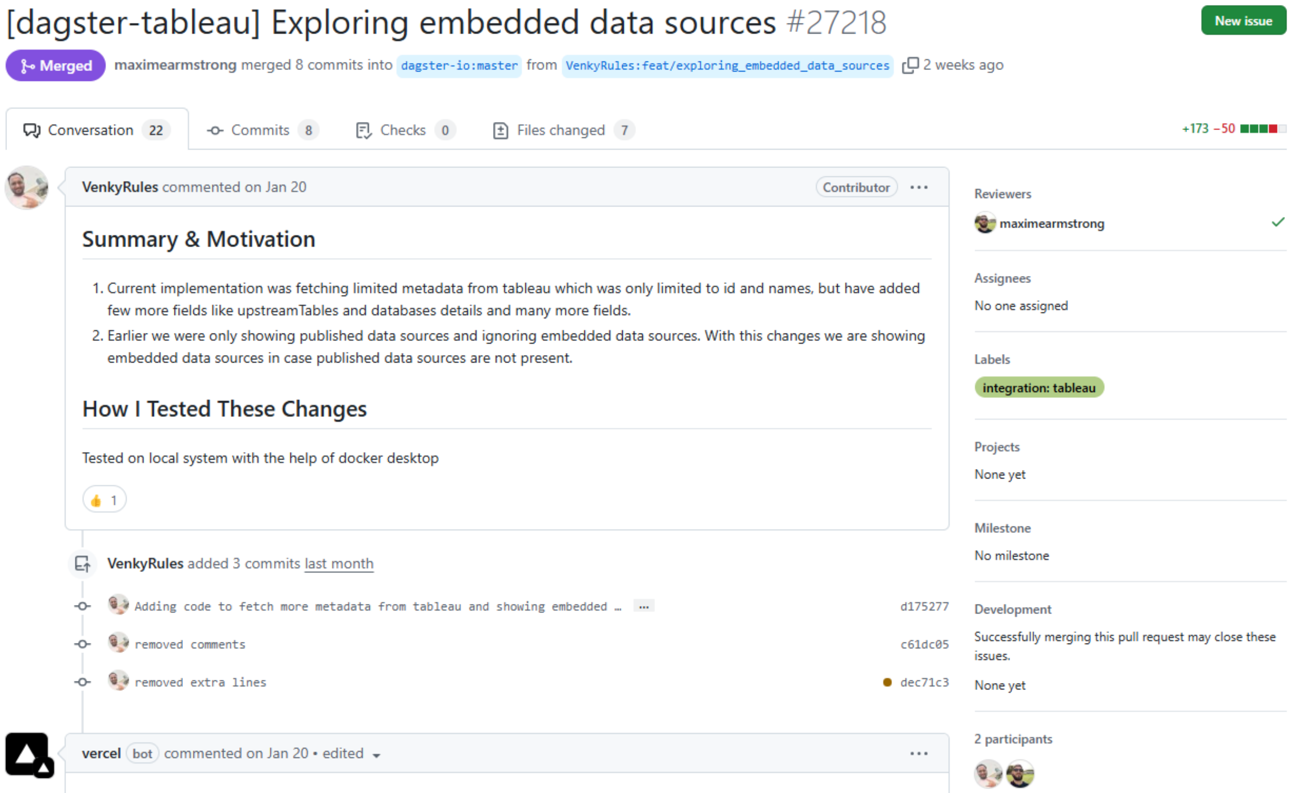
```python
@asset(
    io_manager_key="bigquery_io_manager",
)
def awesome_ml_model(context, reference_addresses: pd.DataFrame, bigquery: BigQueryResource) -> pd.DataFrame:
    # simple normal python code here
    # IO is abstracted
    context.log.info(f"from source: \n{reference_addresses.head()}")
    # 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]})
```

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)

Last things





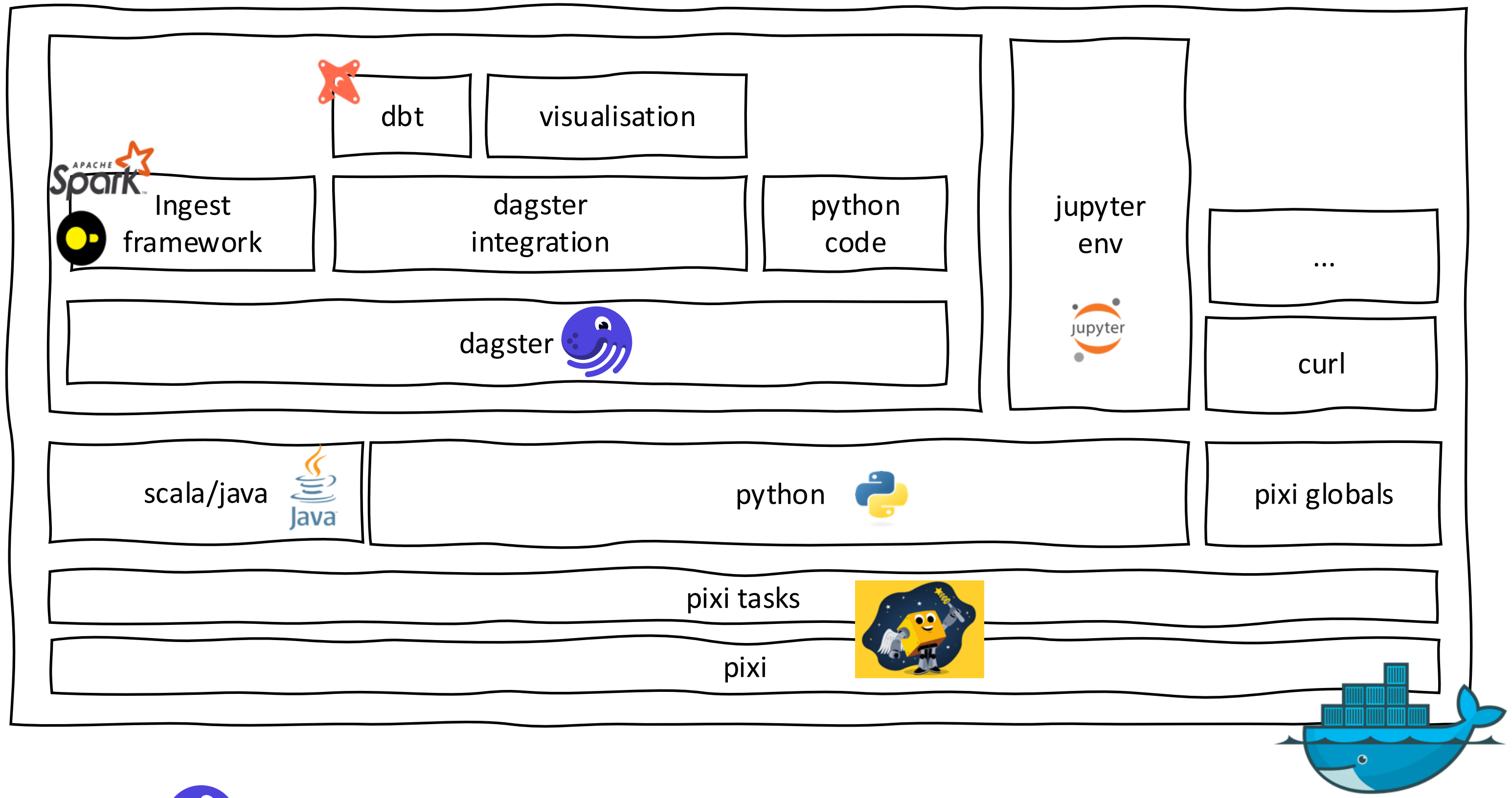
Data platform is team work and we are very proud and excited about the jurney ahead

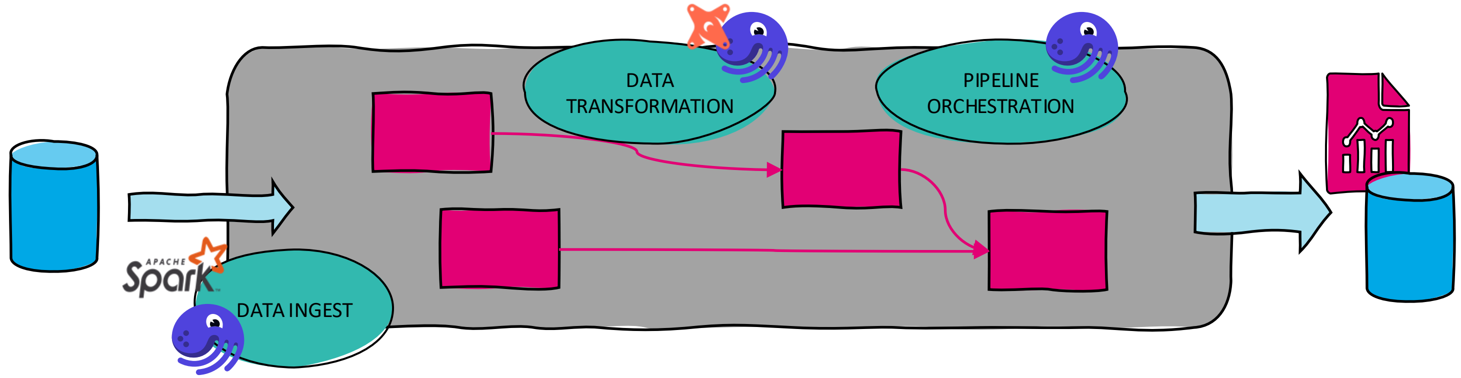
Scaling data pipelines @Telekom

Aleks & Georg

Building block

- Dagster is the core
- Need for java because of custom spark code
- Development tooling for testing and code quality
- Pixi is holding everything together as the environment manager





What should I do now?