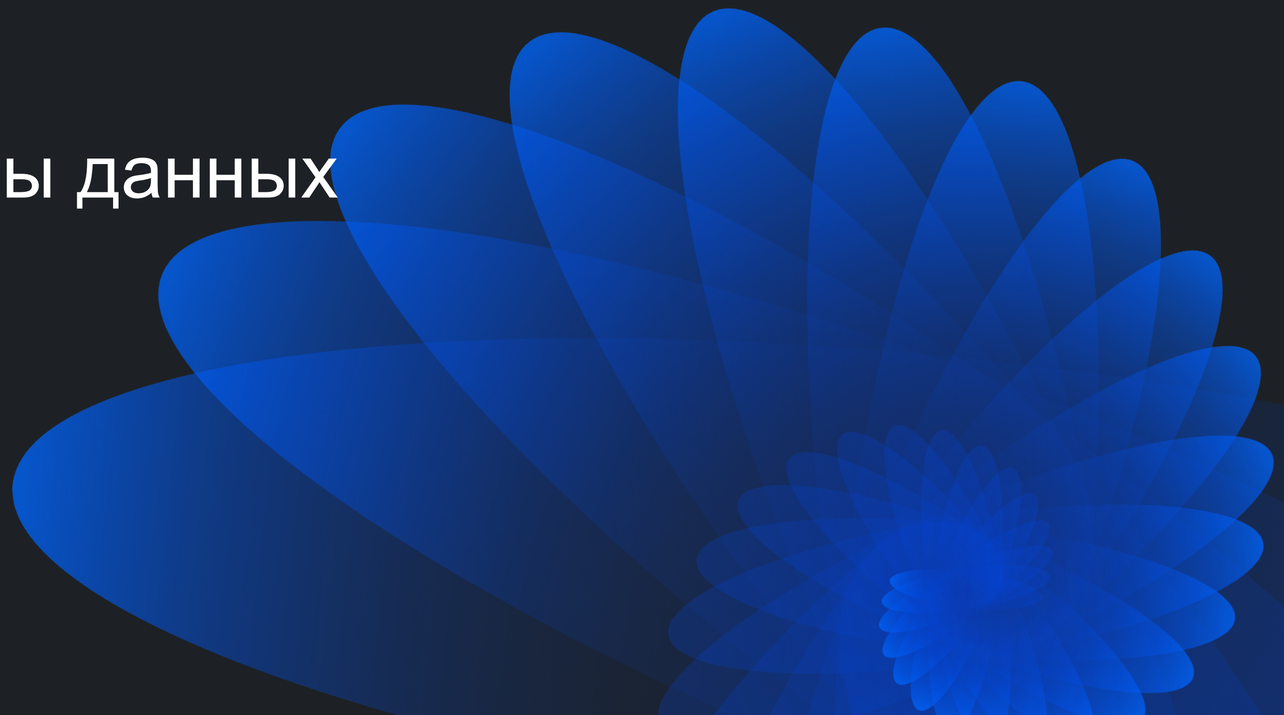
Toloka

dbt

Ядро современной платформы данных



Ермаков Евгений,
Head of Data Platform

Выпускник МГТУ им Н.Э.Баумана
к.т.н. в области СУБД

Более 10 лет опыта работы с системами обработки данных в различных сферах: финансы, e-com, IT

Евгений Ермаков

Head of Data Platform at Toloka

jkermakov@toloka.ai

Toloka

Три кита и четыре столпа дата платформы

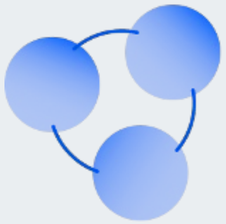
Что?

Архитектура
=
Data Lakehouse

Почему?

Процессы
=
Data mesh

Как?

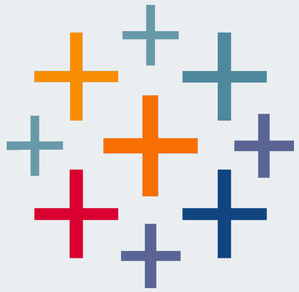


Технологии
=
Modern Data Stack

databricks

dbt

airflow



tableau

Три кита и четыре столпа дата платформы

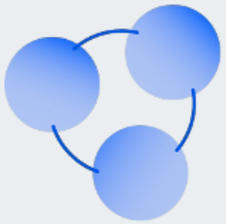
Что?

Архитектура
=
Data Lakehouse

Почему?

Процессы
=
Data mesh

Как?

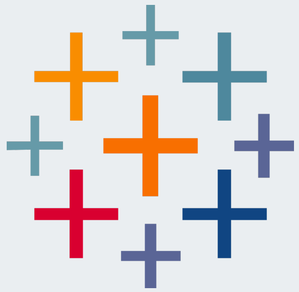


Технологии
=
Modern Data Stack

databricks

dbt

airflow



tableau

План доклада

0 — **Интро**

- Что такое toloka.ai?

1 — **Платформа данных toloka.ai**

- Что такое платформа данных toloka?
- Из каких систем она состоит?
- Какое место занимает dbt?

2 — **Что такое dbt?**

- Какие возможности дает инструмент?
- dbt 101

3 — **Как мы применили dbt?**

- Как мы организовали работу с dbt?
- Какие костыли мы собрали?
- Где подстелили соломку?

4 — **Выводы**

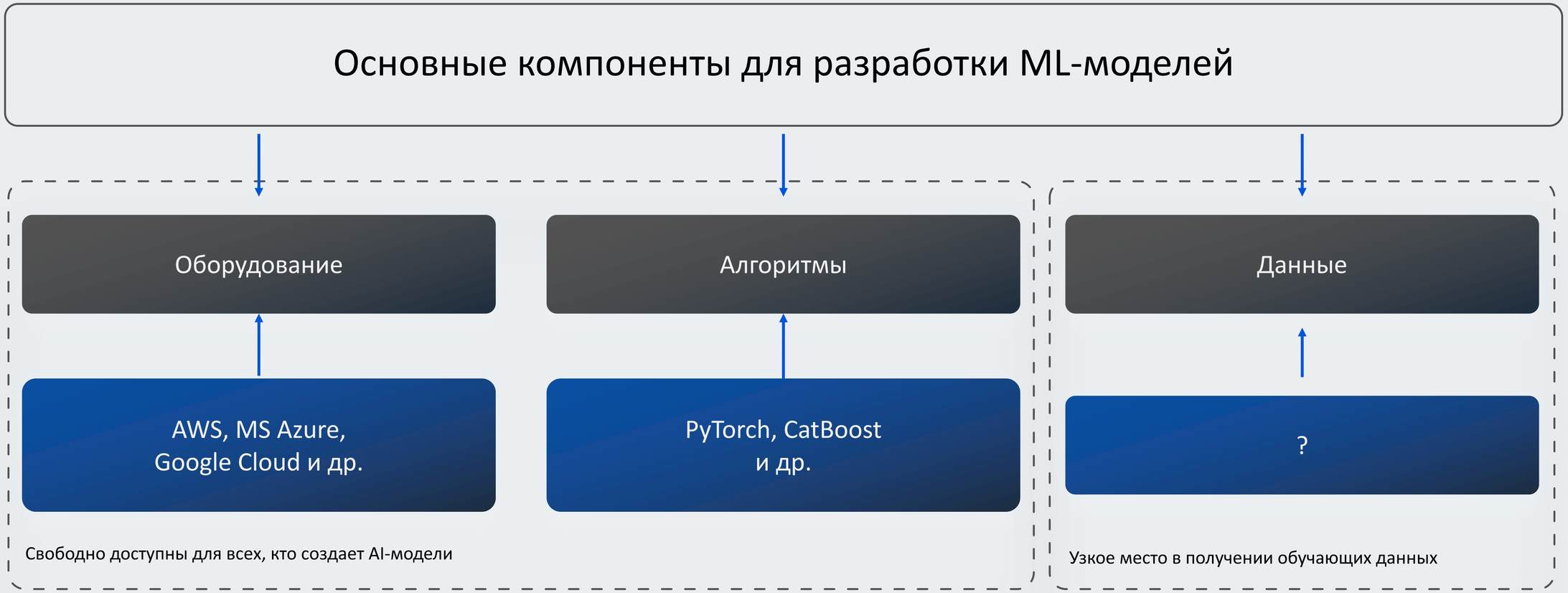
- Как это все работает прямо сейчас?

0. Интро

Что такое toloka.ai?

Датацентричный подход к ML

Каждый имеет доступ к оборудованию и алгоритмам.
Данные — ключ к получению конкурентного преимущества.



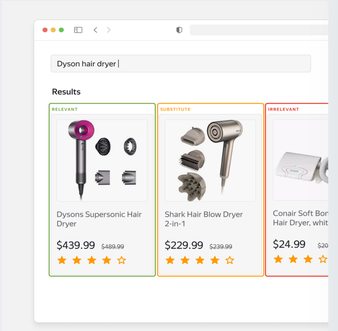
Обучающие данные – ядро ML

В среднем 80% времени*, затрачиваемого на AI-разработку, уходит на обработку обучающих данных

Точность моделей зависит от качества данных

Беспилотные автомобили

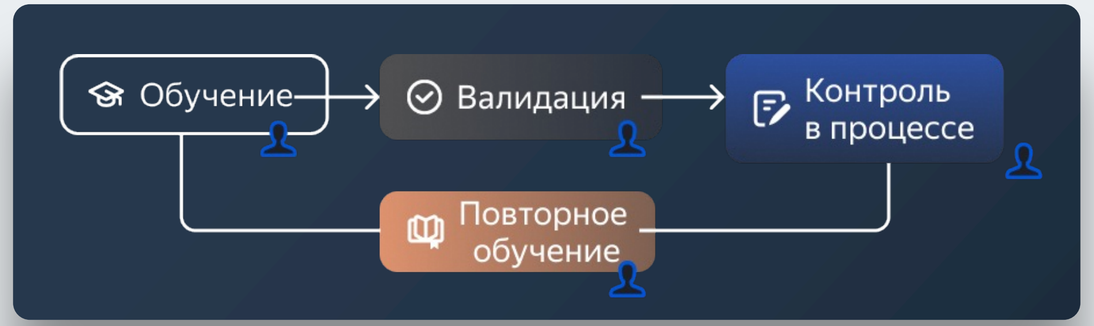
Речевые технологии



Технологии поиска

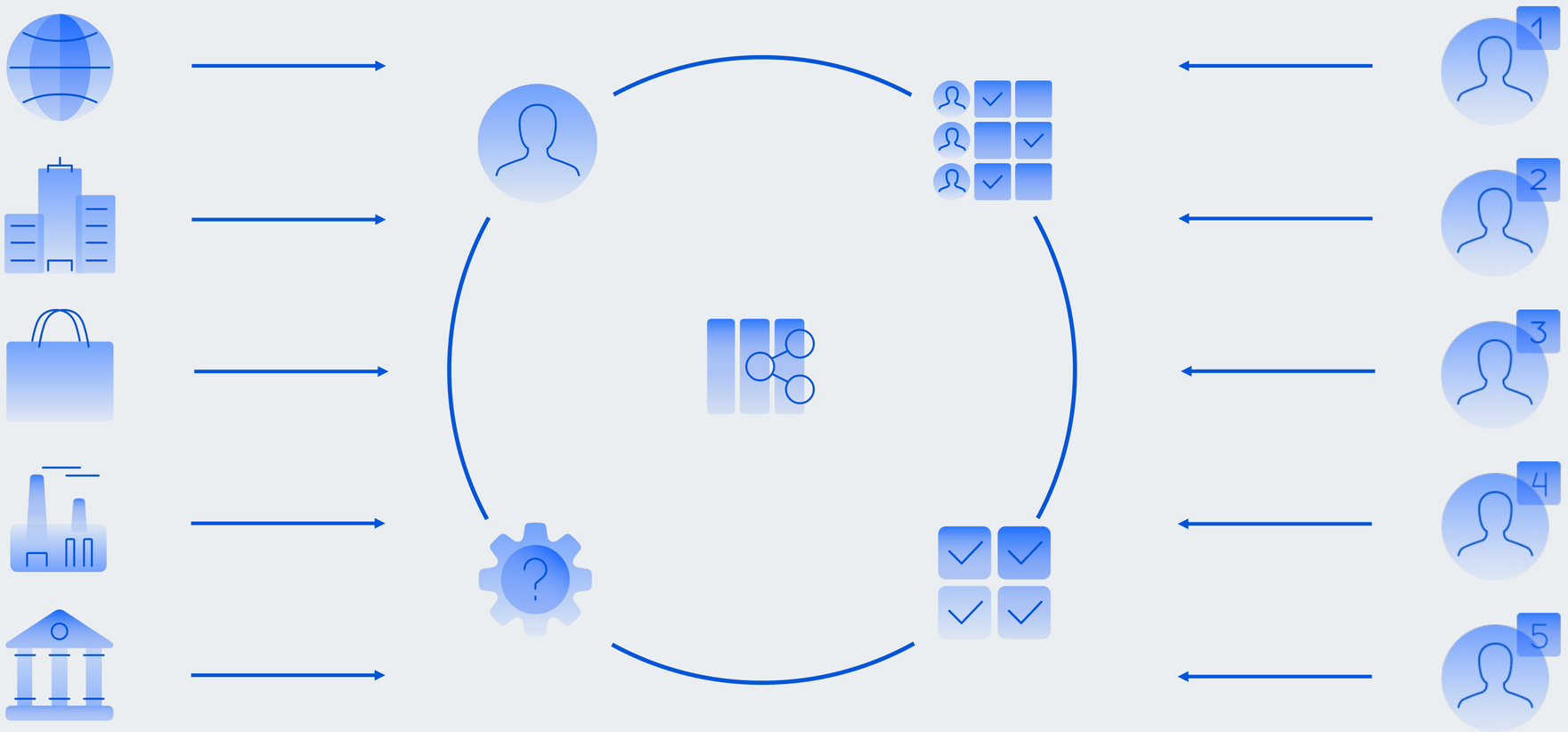
Машинный перевод

Потребность обработанных вручную больших массивов данных возникает на каждом этапе жизненного цикла ML-разработки и внедрения



https://www.cognilytica.com/document/report-data-engineering-preparation-and-labeling-for-ai-2019/

Toloka platform



ML-команды размещают задания и выставляют за них цену

Интеллектуальная платформа и технологическая инфраструктура

Толокеры выполняют задания и получают оплату

1. Платформа данных

Из чего состоит toloka data platform?

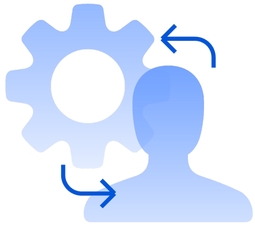
Характеристики платформы

Data / Daily delta

500 Tb / 270 Gb

Data models

over 350

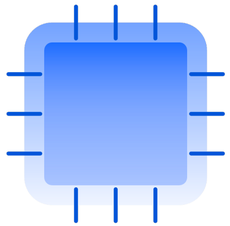


Internal Users

>100

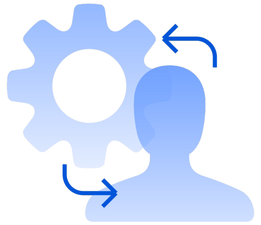
Airflow dags

over 100



Tasks per day

over 5000



Data Domains

42

Good ol' Yandex stack

Ingestion

Event Streaming

Logbroker

Connectors / EL

Logfeller

Transformation

Streaming Processing

Hurma

Batch Processing

YT

Orchestration

Nirvana

Reactor

Storage

YT

MDS

Consumption

Quering

YQL

ClickHouse over YT

Business Intelligence

DataLens

Analytical tools

YQL

jupyter

Data Governance

Data Quality

Data Catalog \ Data Observability

Monitoring

Juggler

Solomon

Три кита дата платформы

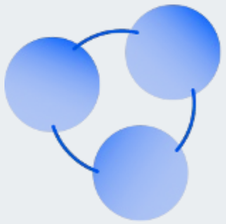
Что?

Архитектура
=
Data Lakehouse

Почему?

Процессы
=
Data mesh

Как?



Технологии
=
Modern Data Stack

Четыре столпа дата платформы

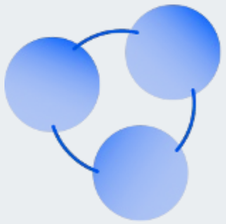
Что?

Архитектура
=
Data Lakehouse

Почему?

Процессы
=
Data mesh

Как?

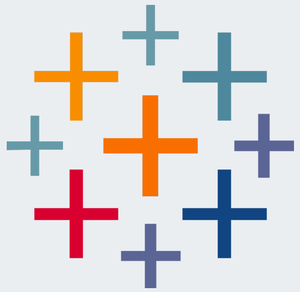


Технологии
=
Modern Data Stack

databricks

dbt

airflow



tableau

Toloka Data Platform

Ingestion

Event Streaming

kafka

Connectors / EL

Apache Airflow

Transformation

Streaming Processing

Spark

Batch Processing

dbt Spark

Orchestration

Apache Airflow dbt

Storage

Consumption

Quering

Business Intelligence

Analytical tools

Data Governance

Data Quality

dbt

Data Catalog \ Data Observability

dbt

Monitoring

Grafana

Dbt at Toloka Data Platform

Ingestion

Event Streaming

kafka

Connectors / EL

Apache Airflow

Transformation

Streaming Processing

Spark

Batch Processing

dbt Spark

Orchestration

Apache Airflow dbt

Storage

Consumption

Quering

Business Intelligence

Analytical tools

Data Governance

Data Quality

dbt

Data Catalog \ Data Observability

dbt

Monitoring

Grafana

Transformation

Удобный механизм работы для разных специалистов

- Аналитики: SQL-only или low-code
- Разработчики: императивный код
- DE: поддержка миграций и data lineage

- Python\Scala Spark
- Все возможности императивного кода

Developers | Data engineers

- SQL + pyspark
- Поддержка миграций
- Сокрытие сложности вставки данных

Data engineers | Analysts

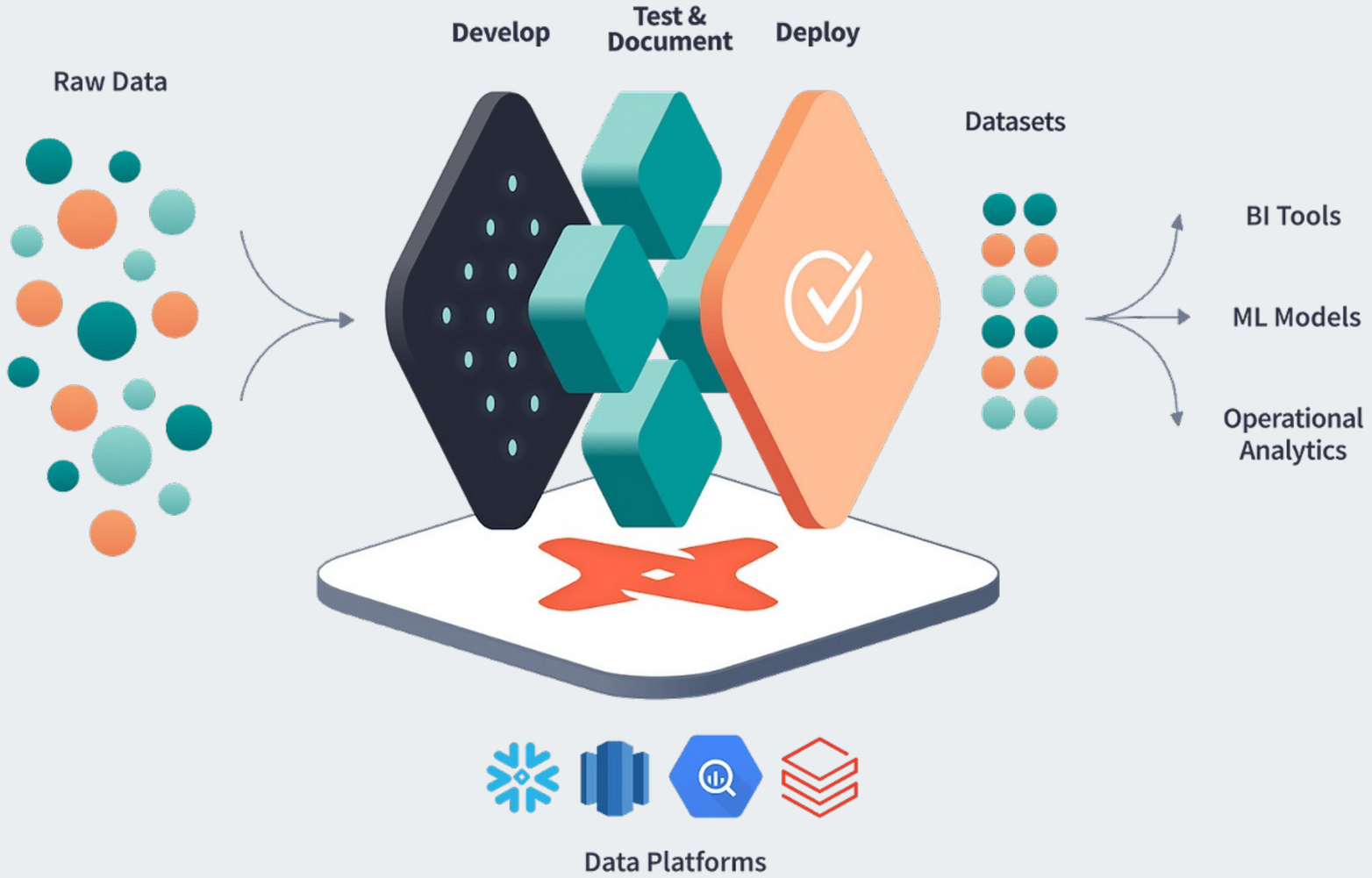
2. dbt

Что такое dbt?

Data Transformation Tool

DBT (Data Building Tool) — это инструмент командной строки, который позволяет аналитикам и инженерам данных преобразовывать данные в своих хранилищах, используя операторы выбора.

DBT отвечает за T (трансформацию) в ETL-процессе, но не выполняет операции извлечения (E) и загрузки (L).

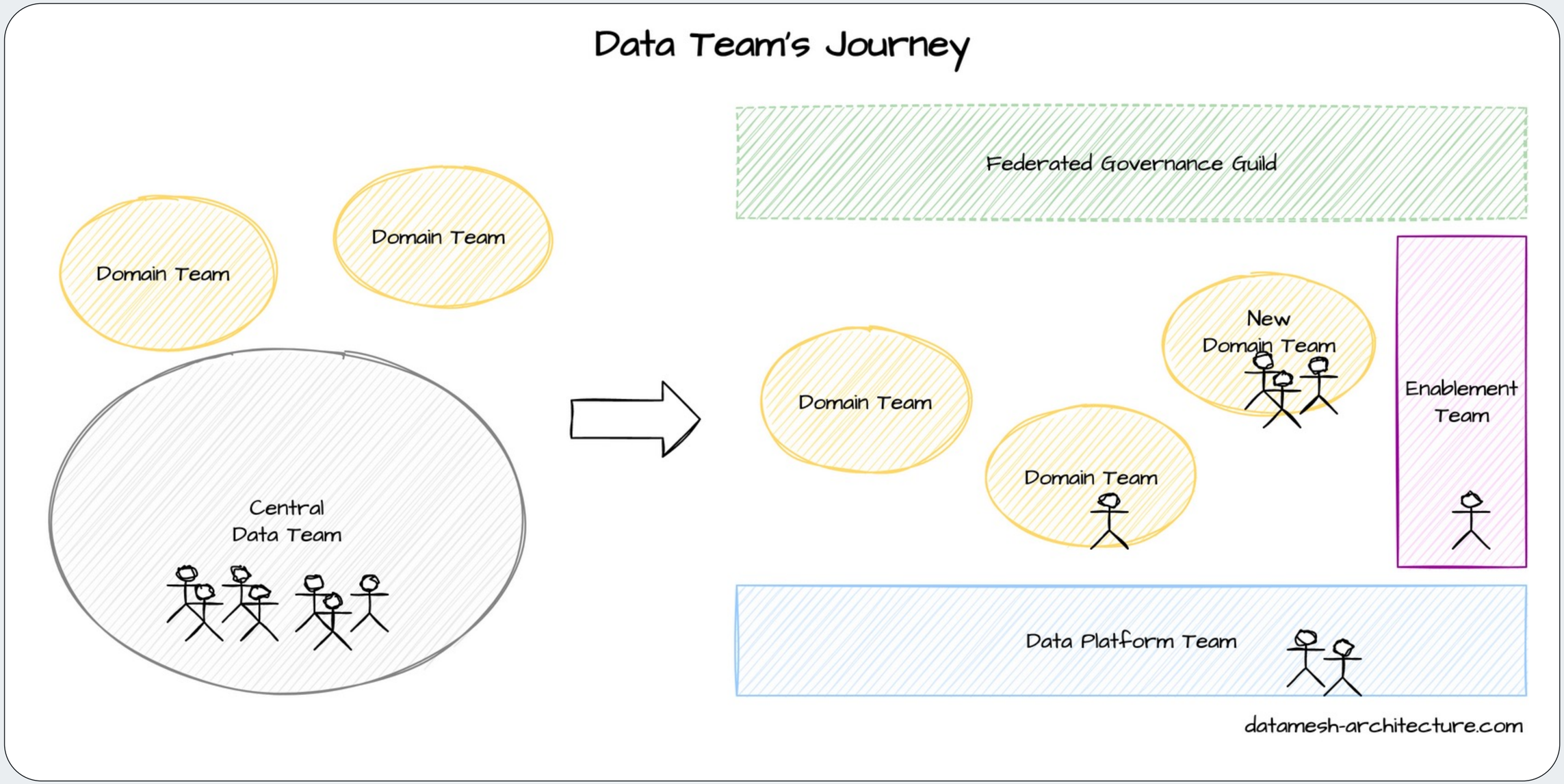


Analytics Engineer

Data Engineer	Analytics Engineer	Data Analyst
• Build custom data integrations • Manage overall pipeline orchestration • Develop & deploy machine learning endpoints • Build and maintain the data platform • Data warehouse performance optimizations	• Provide clean, transformed data ready for analysis • Apply software engineering best practices to analytics code (ex: version control, testing, continuous integration • Maintain data documentation & definitions • Train business users on how to use data visualization tools	• Deep insights work (ex: why did churn spike last month? what are the best acquisition channels?) • Work with business users to understand data requirements • Build critical dashboards • Forecasting

https://www.getdbt.com/what-is-analytics-engineering/

Data mesh


Data Team's Journey
Domain Team
Domain Team
Central
Data Team
Federated Governance Guild
Domain Team
Domain Team
New
Domain Team
Enablement
Team
Data Platform Team
datamesh-architecture.com

Из чего состоит dbt?

dbt model

Основной “кирпичик” dbt-проекта.

Состоит из двух файлов: .sql\.py с описанием логики и .yml с описанием конфигурации.

Особенности:

- select запрос на формирование объекта
- jinja шаблонизация
- другие модели используются через макрос ref() или source()

```
with customers as (
    select *
    from {{ ref('stg_eltool__customers') }}
),
state as (
    select *
    from {{ ref('stg_eltool__state') }}
)
select c.customer_id,
    c.zipcode,
    c.city,
    c.state_code,
    s.state_name,
    c.datetime_created,
    c.datetime_updated,
    c.dbt_valid_from::TIMESTAMP as valid_from,
    CASE
        WHEN c.dbt_valid_to IS NULL THEN '9999-12-31'::TIMESTAMP
        ELSE c.dbt_valid_to::TIMESTAMP
    END as valid_to
from customers c
    join state s on c.state_code = s.state_code
```

dbt model

Основной “кирпичик” dbt-проекта.

Состоит из двух файлов: .sql\.py с описанием логики и .yml с описанием конфигурации.

Особенности:

- Возвращаем dataframe с рассчитанными данными
- Все возможности pyspark
- другие модели используются через специальные команды ref() или source()

```python
import pyspark.sql.functions as F

def model(dbt, session):
    dbt.config(materialized = "incremental")
    df = dbt.ref("upstream_table")

    if dbt.is_incremental:

        # only new rows compared to max in current table
        max_from_this = f"select max(updated_at) from {dbt.this}"
        df = df.filter(df.updated_at >= session.sql(max_from_this).collect()[0][0])

        # or only rows from the past 3 days
        df = df.filter(df.updated_at >= F.date_add(F.current_timestamp(), F.lit(-3)))

    ...

    return df
```

dbt materialization

Материализация – стратегия, по которой dbt выполнит sql\py запрос в СУБД.

Виды материализаций:

- table (sql\py)
- view (sql)
- incremental (sql\py)
  - append
  - merge
  - insert_overwrite
  - delete+insert
- ephemeral (sql)
- materialized view (sql)

models/events/stg_event_log.sql

```
{{ config(materialized='table', sort='timestamp', dist='user_id') }}

select *
from ...
```

models/my_model.sql

```
{{
  config(
    materialized='incremental',
    unique_key='date_day',
    incremental_strategy='delete+insert',
    ...
  )
}}

select ...
```

Настройки проекта

dbt_project.yml – центральный файл с настройками проекта

- пути к моделям
- общие настройки моделей
- профиль по-умолчанию

dbt_project.yml

```yaml
# Example dbt_project.yml file
name: 'jaffle_shop'
profile: 'jaffle_shop'
...
```

profiles.yml – файл с настройками подключения

- Может быть неограниченное количество
- Обычно зависит от окружения (prod\test\dev) и от инстанса СУБД

~/.dbt/profiles.yml

```yaml
# example profiles.yml file
jaffle_shop:
  target: dev
  outputs:
    dev:
      type: postgres
      host: localhost
      user: alice
      password: <password>
      port: 5432
      dbname: jaffle_shop
      schema: dbt_alice
      threads: 4
```

Поддерживаемые платформы

AlloyDB
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.1

BigQuery
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.4

Databricks
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.2

Dremio*
- ✘ Install using the CLI

pypi package 1.5.0

Postgres
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.1

Redshift
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.1

Snowflake
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.2

Spark
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.0

Starburst/Trino
- ✘ Set up in dbt Cloud
- ✘ Install using the CLI

pypi package 1.6.1

Fabric Synapse*
- ✘ Install using the CLI

pypi package 1.4.0rc3

🚧 Verification in progress

Azure Synapse*
- ✘ Install using the CLI

pypi package 1.3.2

🚧 Verification in progress

Teradata*
- ✘ Install using the CLI

pypi package 1.5.4

Как запустить dbt?

Команды dbt

› dbt build

Собирает все зависимости между моделями и компилирует sql\py запросы. Результат пишется в manifest.json

```
$ dbt build
Running with dbt=0.21.0-b2
Found 1 model, 4 tests, 1 snapshot, 1 analysis, 341 macros, 0 operations, 1 seed file, 2 sources, 2 exposures

18:49:43 | Concurrency: 1 threads (target='dev')
18:49:43 |
18:49:43 | 1 of 7 START seed file dbt_jcohen.my_seed..... [RUN]
18:49:43 | 1 of 7 OK loaded seed file dbt_jcohen.my_seed..... [INSERT 2 in 0.09s]
18:49:43 | 2 of 7 START view model dbt_jcohen.my_model..... [RUN]
18:49:43 | 2 of 7 OK created view model dbt_jcohen.my_model..... [CREATE VIEW in 0.12s]
18:49:43 | 3 of 7 START test not_null_my_seed_id..... [RUN]
18:49:43 | 3 of 7 PASS not_null_my_seed_id..... [PASS in 0.05s]
18:49:43 | 4 of 7 START test unique_my_seed_id..... [RUN]
18:49:43 | 4 of 7 PASS unique_my_seed_id..... [PASS in 0.03s]
18:49:43 | 5 of 7 START snapshot snapshots.my_snapshot..... [RUN]
18:49:43 | 5 of 7 OK snapshotted snapshots.my_snapshot..... [INSERT 0 5 in 0.27s]
18:49:43 | 6 of 7 START test not_null_my_model_id..... [RUN]
18:49:43 | 6 of 7 PASS not_null_my_model_id..... [PASS in 0.03s]
18:49:43 | 7 of 7 START test unique_my_model_id..... [RUN]
18:49:43 | 7 of 7 PASS unique_my_model_id..... [PASS in 0.02s]
18:49:43 |
18:49:43 | Finished running 1 seed, 1 view model, 4 tests, 1 snapshot in 1.01s.

Completed successfully

Done. PASS=7 WARN=0 ERROR=0 SKIP=0 TOTAL=7
```

Команды dbt

› dbt run

Запускает скомпилированный sql\py запрос в зависимости от переданных параметров.

Запуск модели по имени\тегу

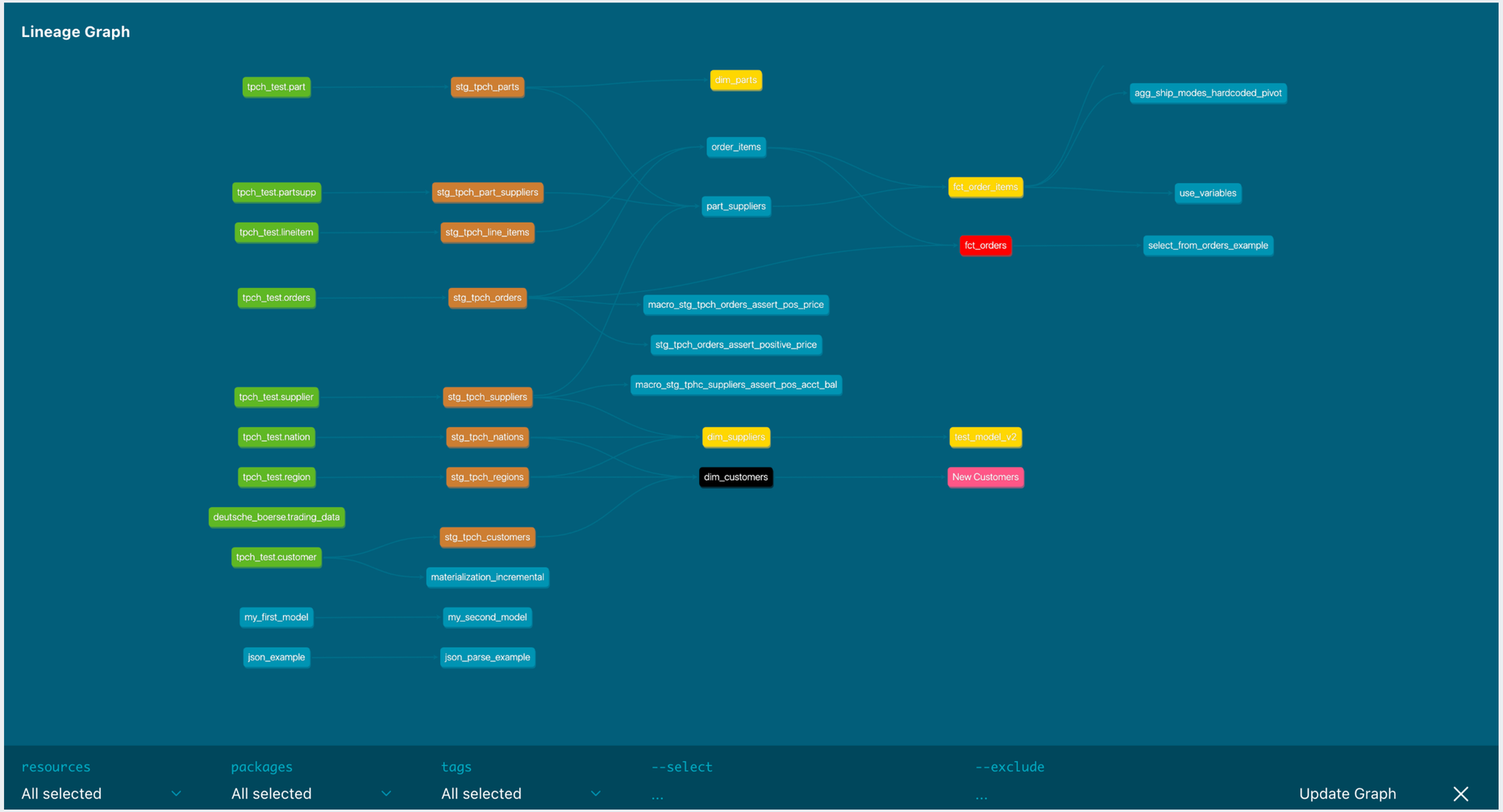
```
$ dbt run --select my_dbt_project_name   # runs all models in your project
$ dbt run --select my_dbt_model          # runs a specific model
$ dbt run --select path.to.my.models     # runs all models in a specific directory
$ dbt run --select my_package.some_model # run a specific model in a specific package
$ dbt run --select tag:nightly           # run models with the "nightly" tag
$ dbt run --select path/to/models        # run models contained in path/to/models
$ dbt run --select path/to/my_model.sql  # run a specific model by its path
```

Запуск модели по графу

```
$ dbt run --select my_model+          # select my_model and all children
$ dbt run --select +my_model          # select my_model and all parents
$ dbt run --select +my_model+         # select my_model, and all of its parents and children
```

Документация dbt

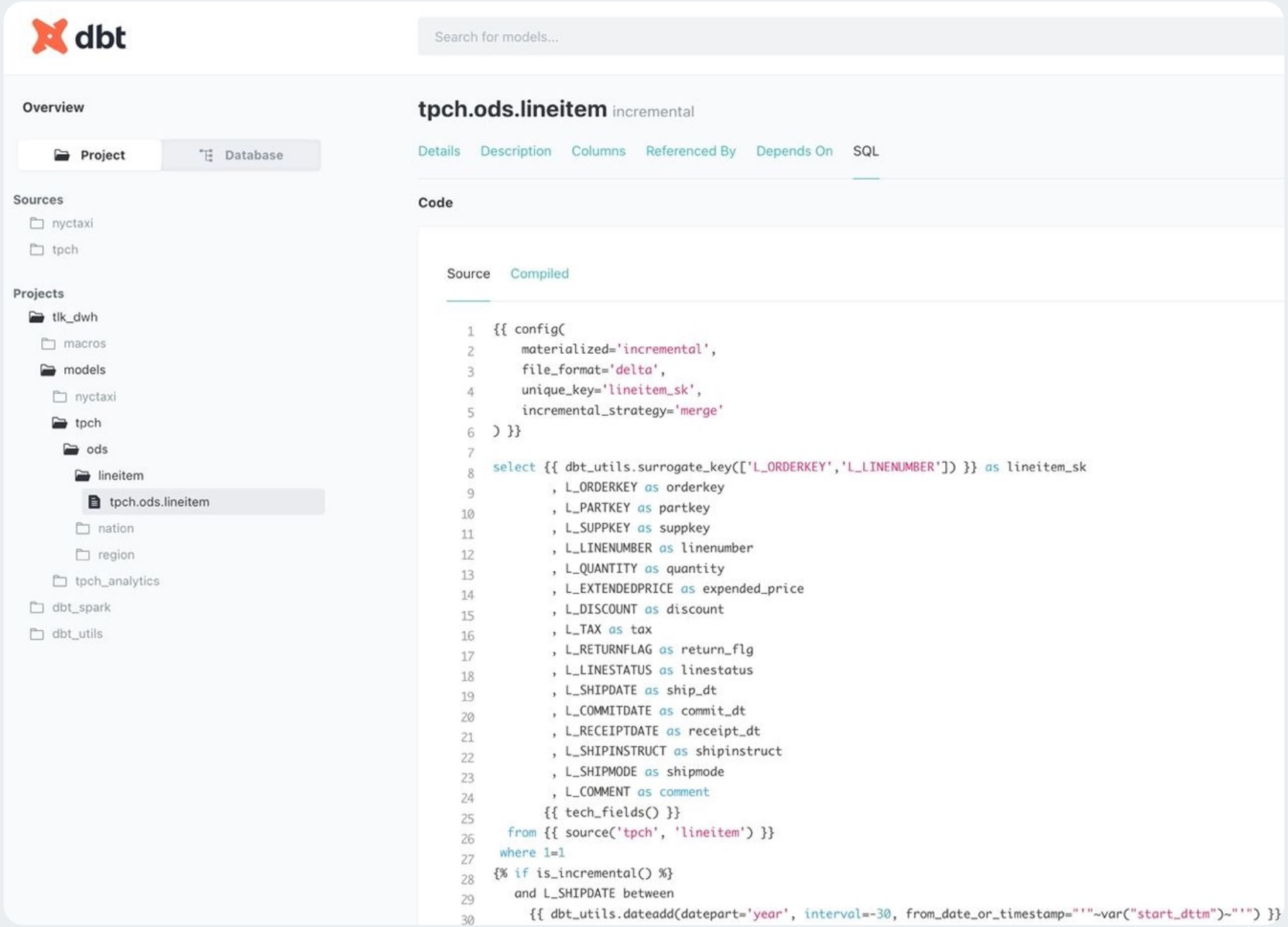
- dbt docs generate
- dbt docs serve



Toloka https://docs.getdbt.com/reference/resource-configs/docs

Документация dbt

- dbt docs generate
- dbt docs serve



dbt test

- Описываются в yml файле модели
- Применяются как на поле, так и на всю таблицу
- Запускаются командой dbt test

```yaml
version: 2

models:
  - name: fact_interaction
    description: '{{ doc("fact_interaction_description") }}'

    columns:
        - name: sk_interaction
          description: Surrogate key unique identifier for the case
          tests:
              - unique
              - not_null
        - name: bk_source_driver
          description: Business key from the source
              - unique
              - not_null
        - name: count_interactions
          description: Number of interactions with the customer
              - not_null
        - name: fk_interaction_detail
          description: Foreign key pointing to interaction detail dimension
          tests:
              - not_null
              - unique
              - relationships:
                  to: source('sid', 'dim_interaction_detail')
                  field: sk_interaction_detail
        - name: fk_interaction_text
          description: Foreign key pointing to interaction text dimension
          tests:
              - not_null
              - unique
              - relationships: # Referential integrity test
                  to: source('sid', 'dim_interaction_text')
                  field: sk_interaction_detail
        - name: performance_band
          description: Performance classified as Ungraded or Very Good or Good or OK or Bad or Very Bad
          tests:
            - accepted_values:
                values: ['Ungraded', 'Very Good', 'Good', 'OK', 'Bad', 'Very Bad']
```

Сущности dbt

snapshots	A way to capture the state of your mutable tables so you can refer to it later.
seeds	CSV files with static data that you can load into your data platform with dbt.
tests	SQL queries that you can write to test the models and resources in your project.
macros	Blocks of code that you can reuse multiple times.
docs	Docs for your project that you can build.
sources	A way to name and describe the data loaded into your warehouse by your Extract and Load tools.
exposures	A way to define and describe a downstream use of your project.
metrics	A way for you to define metrics for your project.
groups	Groups enable collaborative node organization in restricted collections.
analysis	A way to organize analytical SQL queries in your project such as the general ledger from your QuickBooks.

https://docs.getdbt.com/docs/build/projects

dbt

Сильные стороны

- Низкий порог входа
- Сокрытие сложности вставки и миграции данных
- Документация и lineage «из коробки»
- DQ\тестирование данных в виде ожиданий от данных

Слабые стороны

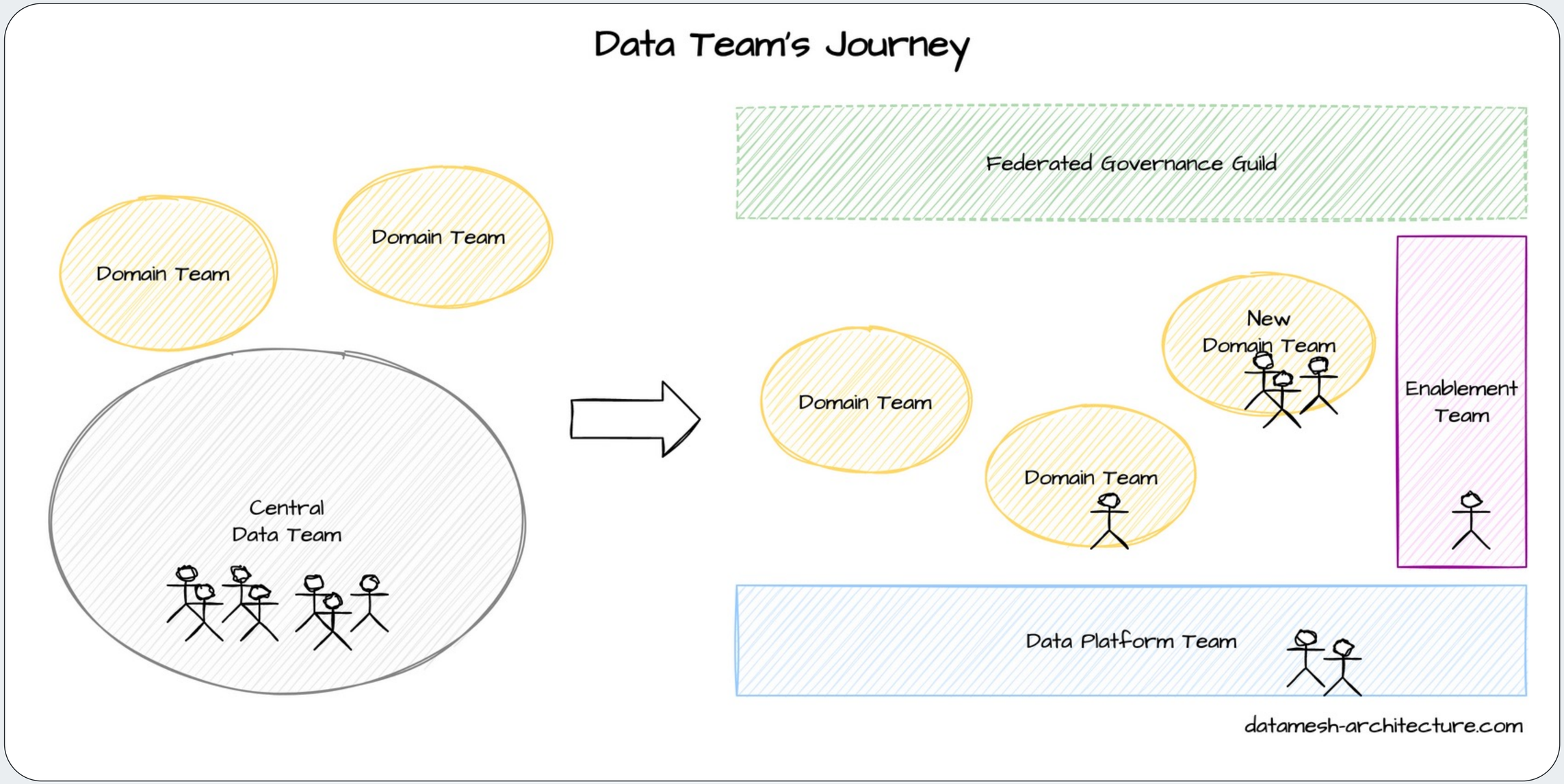
- Jinja макросы ☹
- Местами сложная кастомизируемость (заданная структура проекта, одно имя модели на все проекты и т.п.)
- Слабая поддержка py-моделей (появилась в версии 1.4)

3. Внедрение dbt

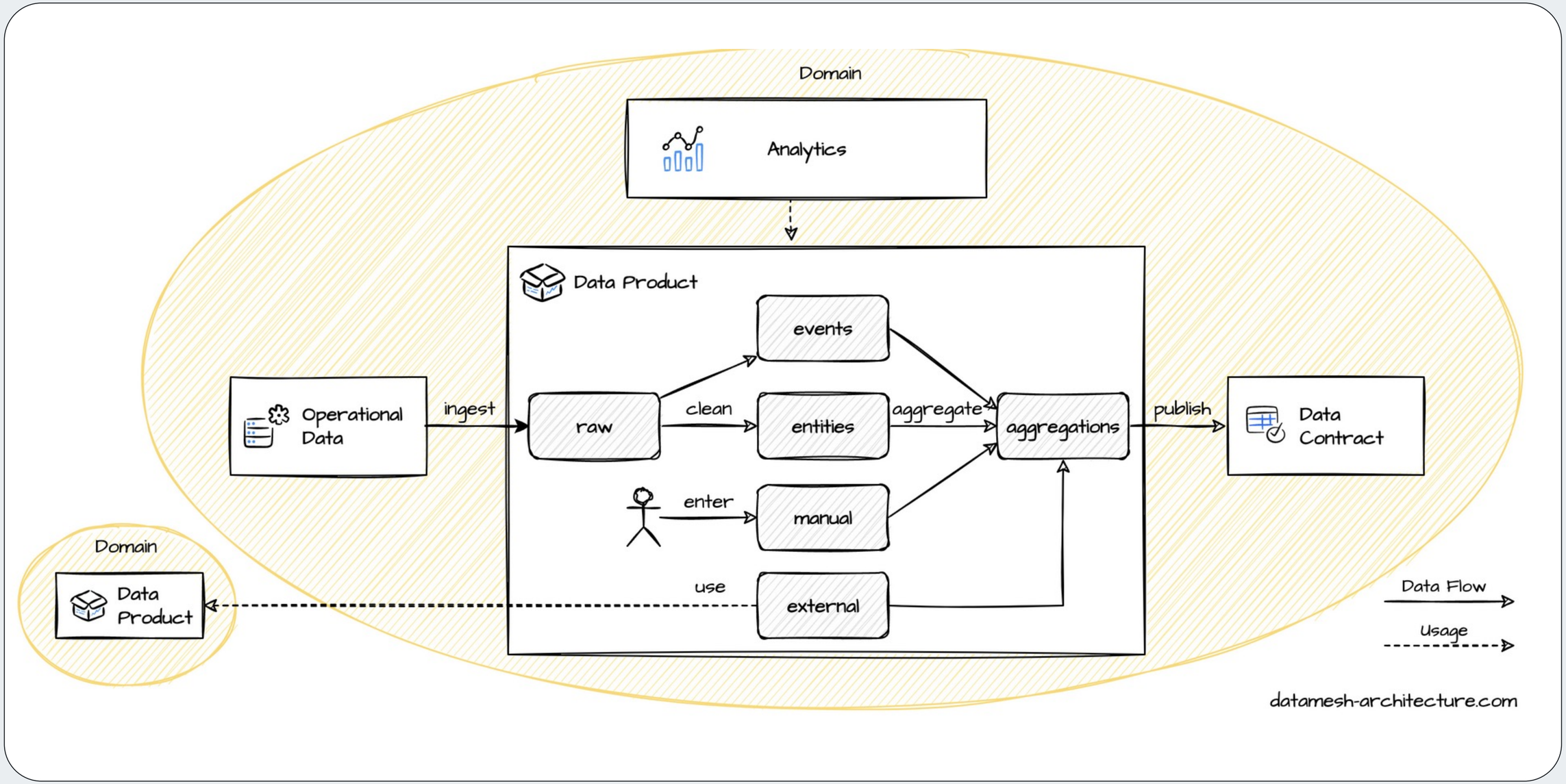
Как мы использовали dbt?

Вопрос: как организовать проект (с учетом Data Mesh) ?

Data mesh

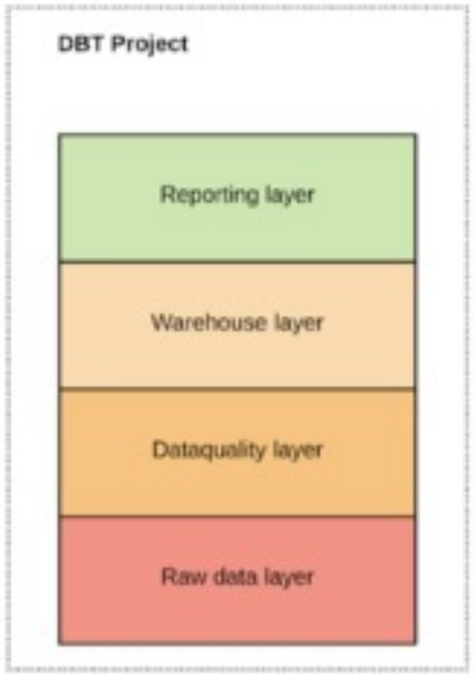

Data Team's Journey
Domain Team
Domain Team
Central
Data Team
Federated Governance Guild
Domain Team
Domain Team
New
Domain Team
Enablement
Team
Data Platform Team
datamesh-architecture.com

dbt+airflow

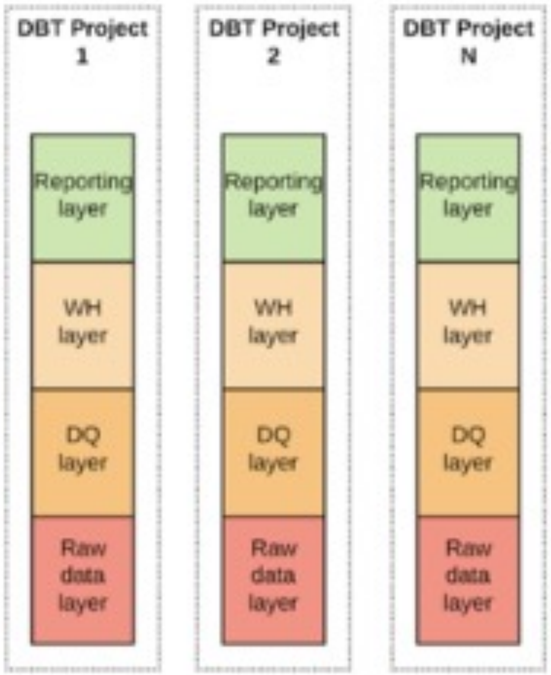

Domain
Analytics
Data Product
events
Operational Data
ingest
raw
clean
entities
aggregate
aggregations
publish
Data Contract
enter
manual
Domain
Data Product
use
external
Data Flow
Usage
datamesh-architecture.com

Четыре варианта организации проекта

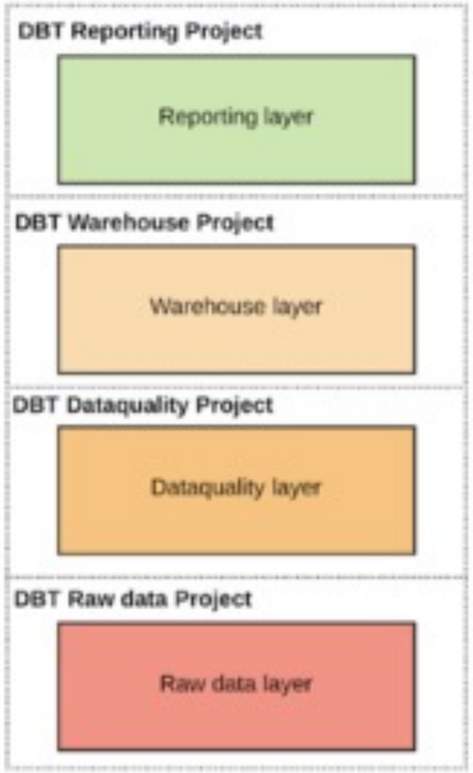
Монолитная



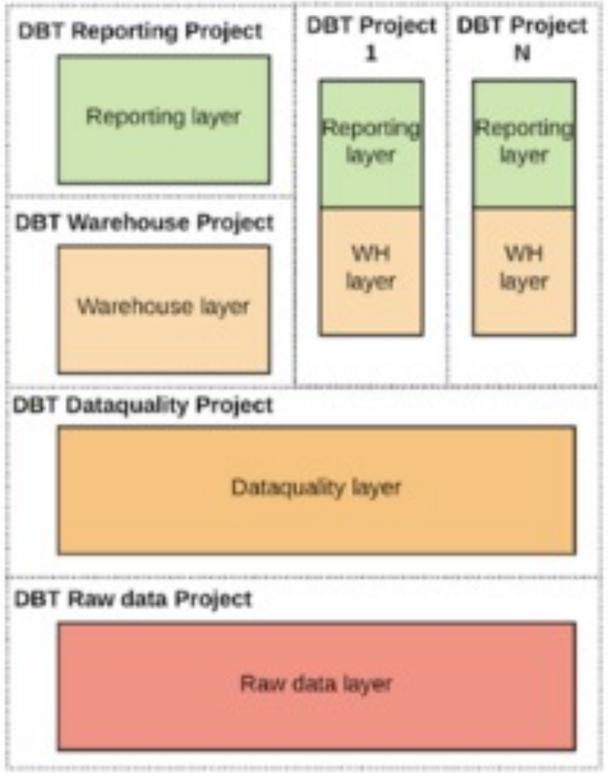
Микросервисная



Послойная



Смешанная

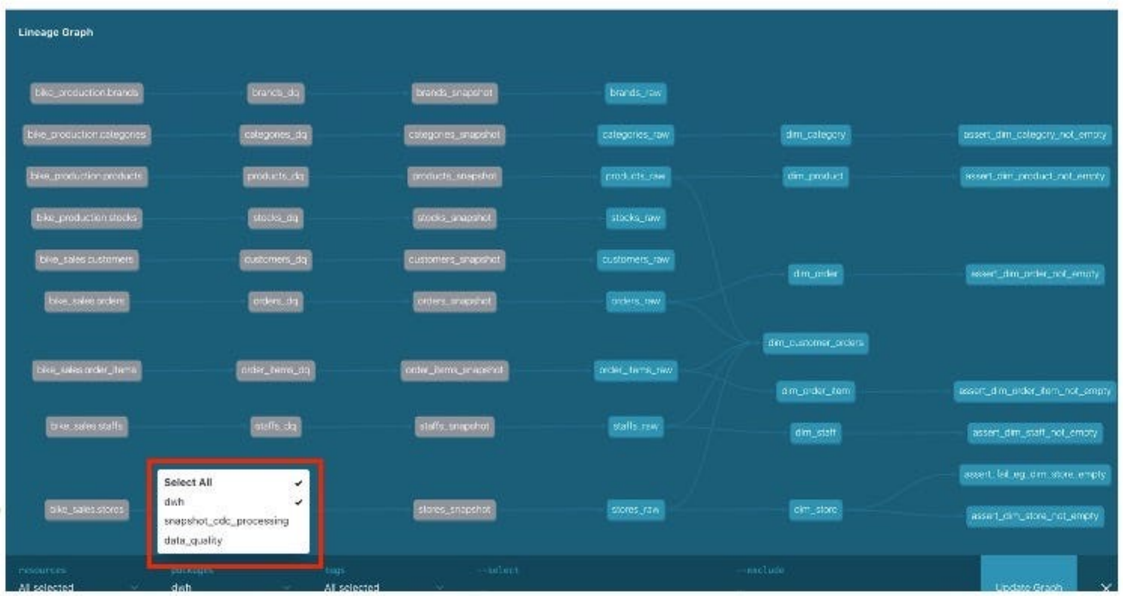


Toloka https://medium.com/geekculture/how-to-structure-your-dbt-project-c62103deceb4

Data Mesh – микросервисная архитектура DWH

Микросервисная организация проекта

«Фейковый» проект для общей документации

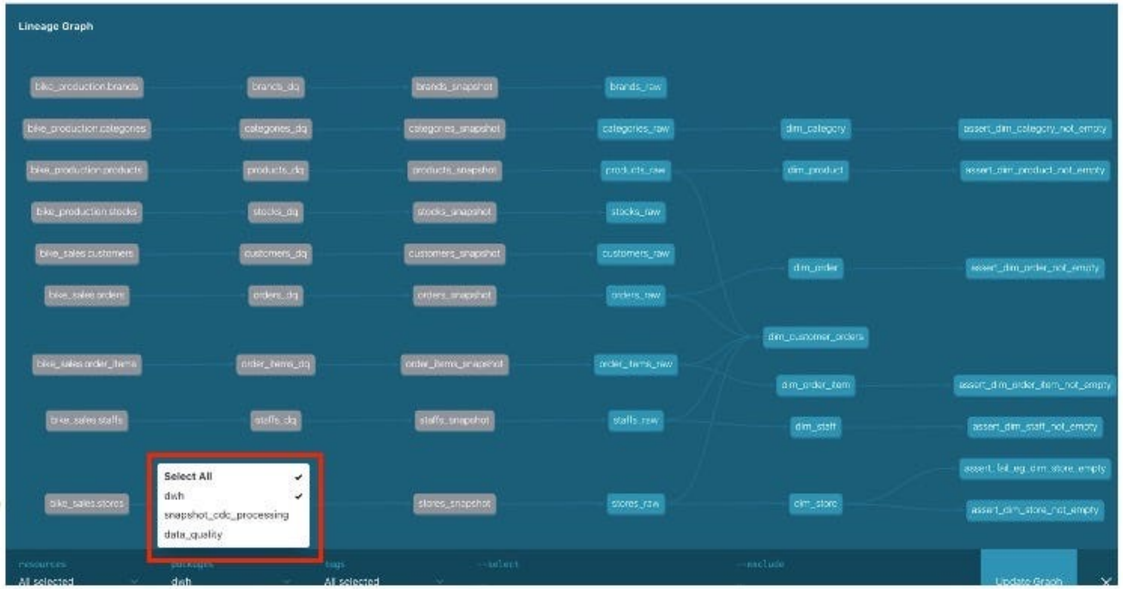


https://medium.com/geekculture/how-to-structure-your-dbt-project-c62103deceb4#ed44

Data Mesh – микросервисная архитектура DWH

Микросервисная организация проекта

«Фейковый» проект для общей документации



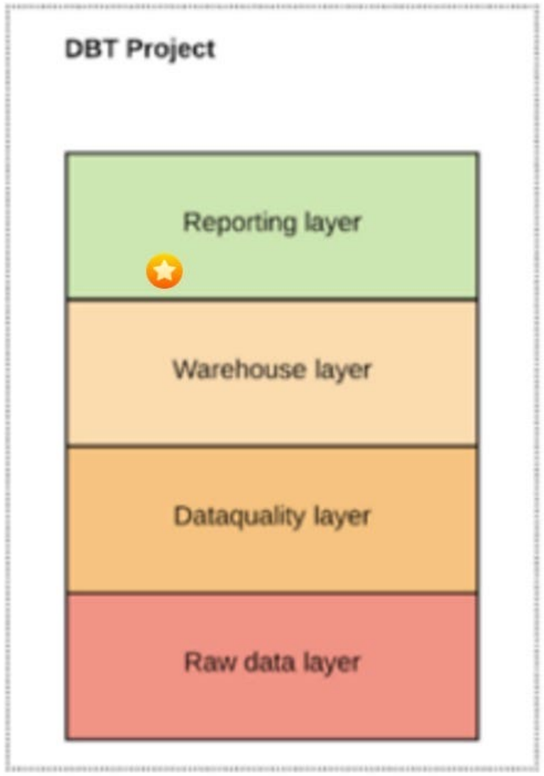
Да, но нет.

› Мы не можем называть одинаково модели в разных проектах (починили с версии 1.6)
https://docs.getdbt.com/faqs/models/unique-model-names

› Проекты\домены не могут импортировать друг друга, проблема циклических зависимостей

Подоменная организация

Монолитная



File/folder structure of the (default)
'Monolithic' dbt project structure

```
<dbt project name>
    └── analysis
    └── data
    └── macros
    └── models
            └── raw_data_layer
            └── data_quality_layer
            └── warehouse_layer
            └── reporting_layer
    └── profiles
    └── snapshots
    └── tests
    dbt_project.yml
```

Домен данных на пути к модели

```
<dbt project name>
├── dbt_project.yml
├── data
├── macros
└── models
    ├── data_domain
    └── data_domain
        └── data_layer
            ├──model.yml
            └──model.sql
        └── data_layer
├── snapshots
└── tests
```

Плюсы\минусы решения

Сильные стороны

- Общая документация и переиспользование данных
- Единые контролируемые стандарты
- Единый граф расчета всей организации
- Codeowners доменов средствами GH

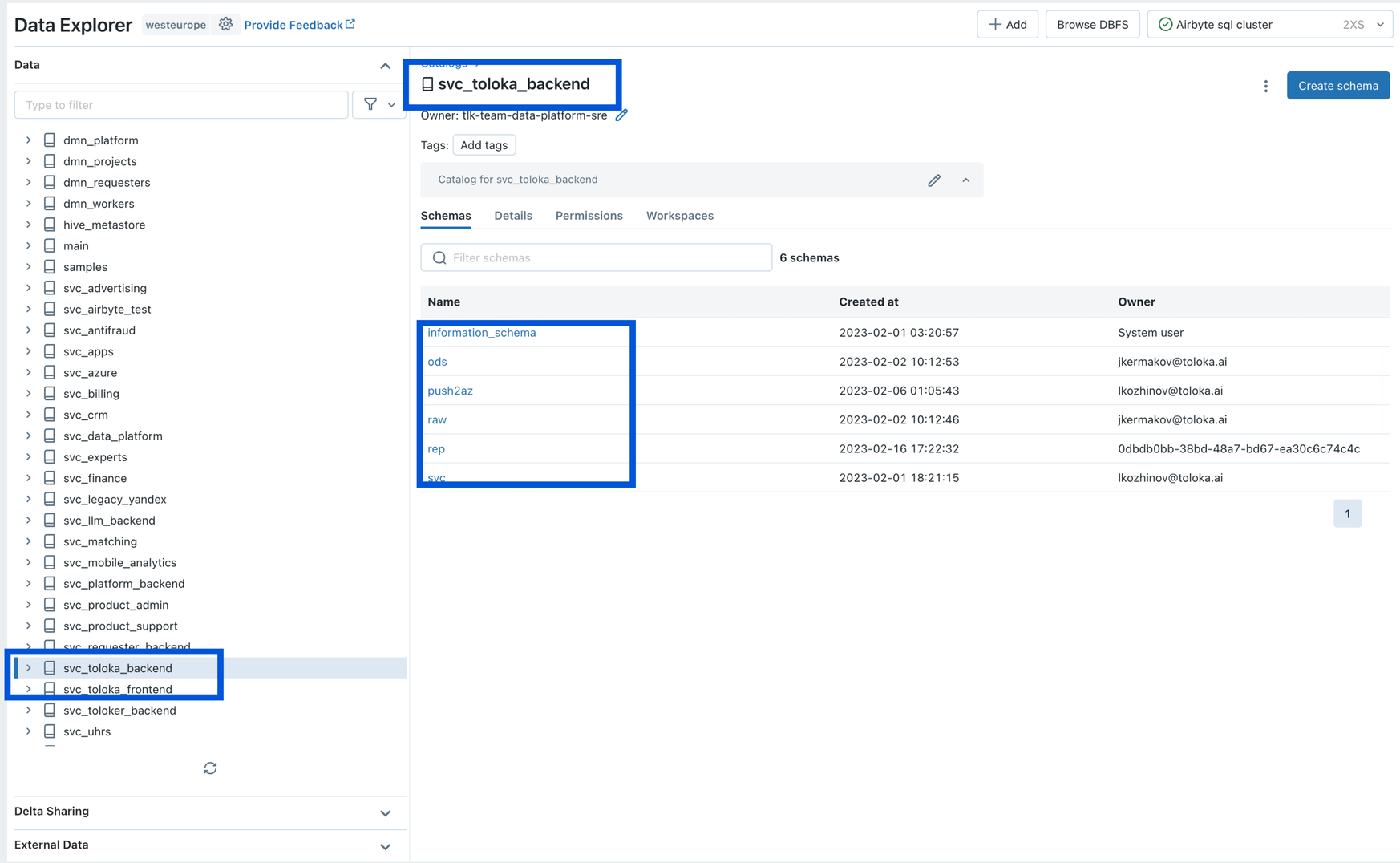
Слабые стороны

- Конфликты в общем dbt_projects.yml
- Невозможность закрыть часть кода (например, антифрод)
- Единовременный релизный процесс на все домены

Вопрос: как разделить prod\dev enviroment ?

Production environment

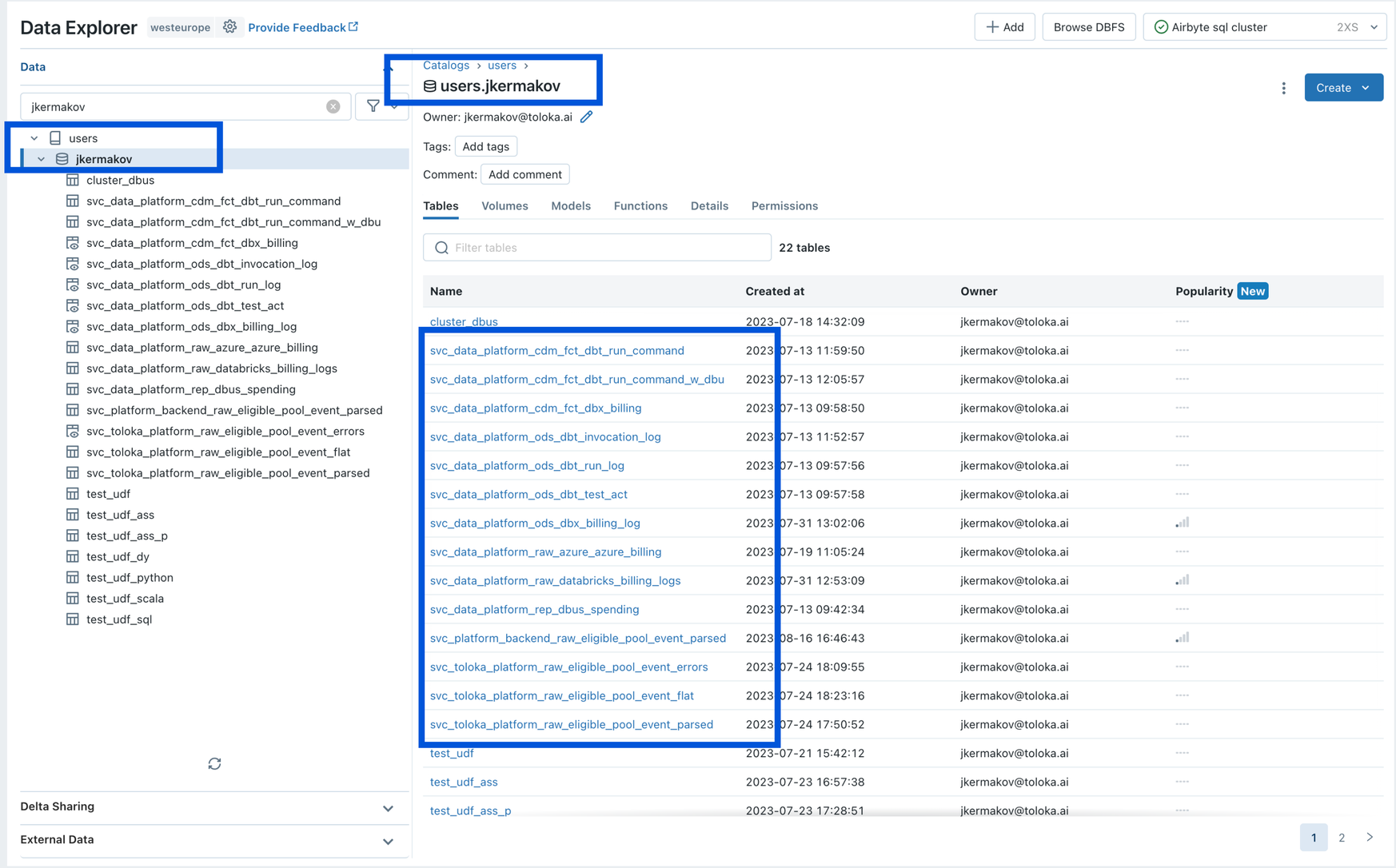
Объект формируется по пути <domain_name>.<layer_name>.<model_name>



Toloka

Development emviroment

Объект формируется по пути **users.<user_name>.<domain_name>_<layer_name>_<model_name>**



Toloka

dbt custom names

В dbt есть возможность переопределить макросы, отвечающие за имена

- generate_schema_name
- generate_database_name
- generate_alias_name

Простая логика:

- Если target==prod, то генерируем обычное имя объекта
- Если target==dev, то генерируем имя в user-область

get_custom_alias.sql

```
{% macro generate_alias_name(custom_alias_name=none, node=none) -%}

    {%- if custom_alias_name -%}

        {{ custom_alias_name | trim }}

    {%- elif node.version -%}

        {{ return(node.name ~ "_v" ~ (node.version | replace(".", "_"))) }}

    {%- else -%}

        {{ node.name }}

    {%- endif -%}

{%- endmacro %}
```

Плюсы\минусы решения

Сильные стороны

- Удобно и понятно пользователям
- Возможность протестировать модель на prod-мощностях
- Не нужно генерировать отдельные тестовые срезы

Слабые стороны

- Сложно тестировать изменения в макросах

Вопрос: как запускать модели по расписанию?

Три кита и четыре столпа дата платформы

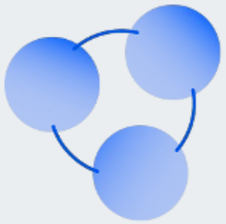
Что?

Архитектура
=
Data Lakehouse

Почему?

Процессы
=
Data mesh

Как?



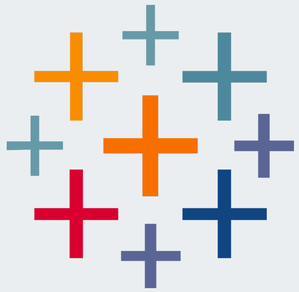
Технологии
=
Modern Data Stack

databricks

dbt

?

airflow



tableau

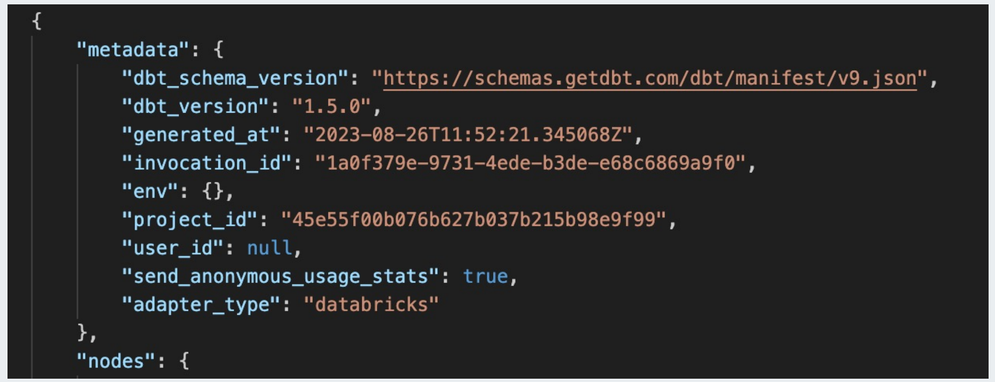
airflow+dbt

Основная идея: у нас уже есть граф в dbt, на базе него можно создать граф в airflow

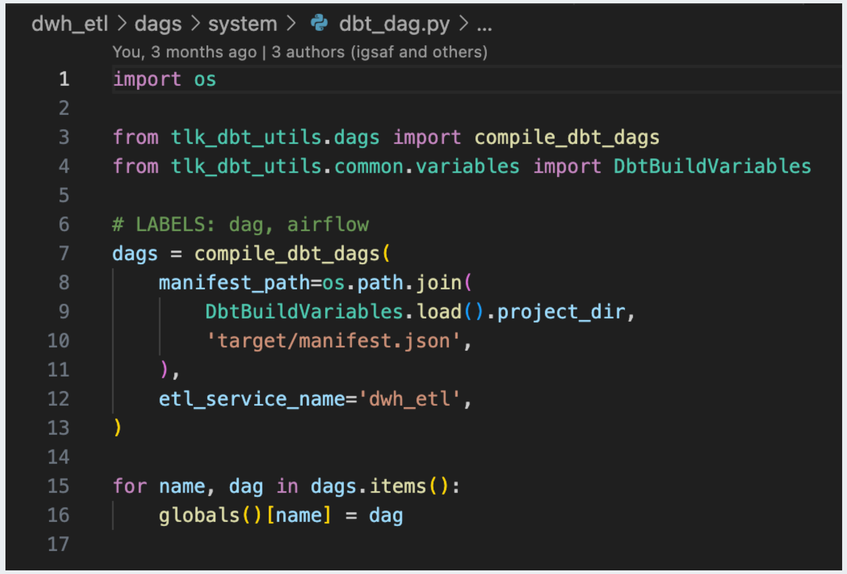
1. Собираем dbt

› dbt build

2. Получаем все зависимости в manifest.json



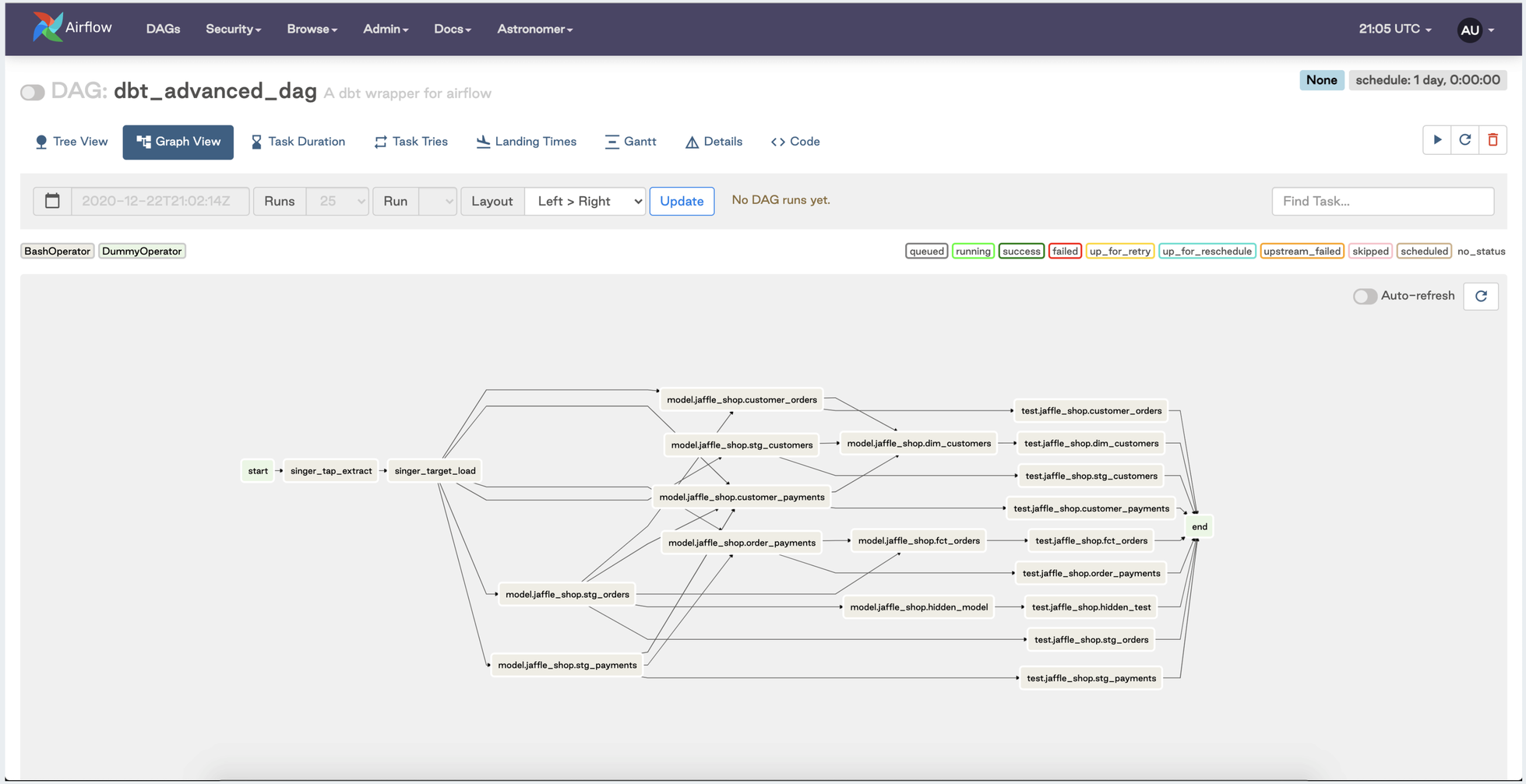
3. На базе зависимостей строим Dag



Toloka https://www.astronomer.io/blog/airflow-dbt-2/

airflow+dbt

Основная идея: у нас уже есть граф в dbt, на базе него можно создать граф в airflow



https://www.astronomer.io/blog/airflow-dbt-2/

model task

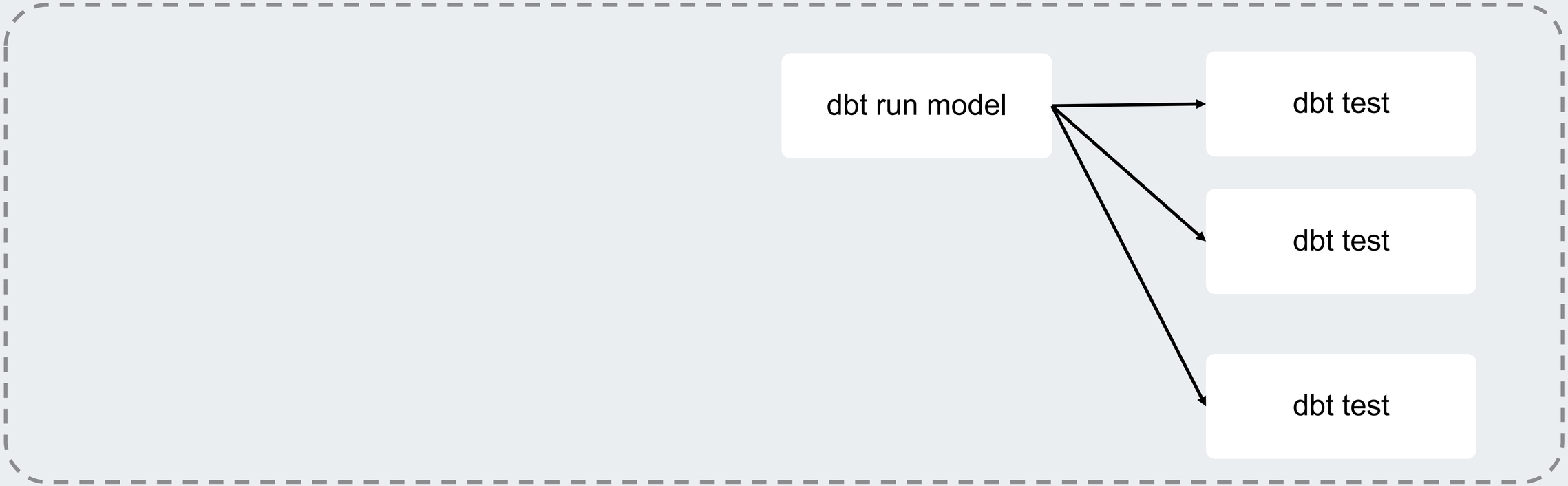
- Основной оператор
- Отвечает за построение модели

```
dbt run model
```

model task

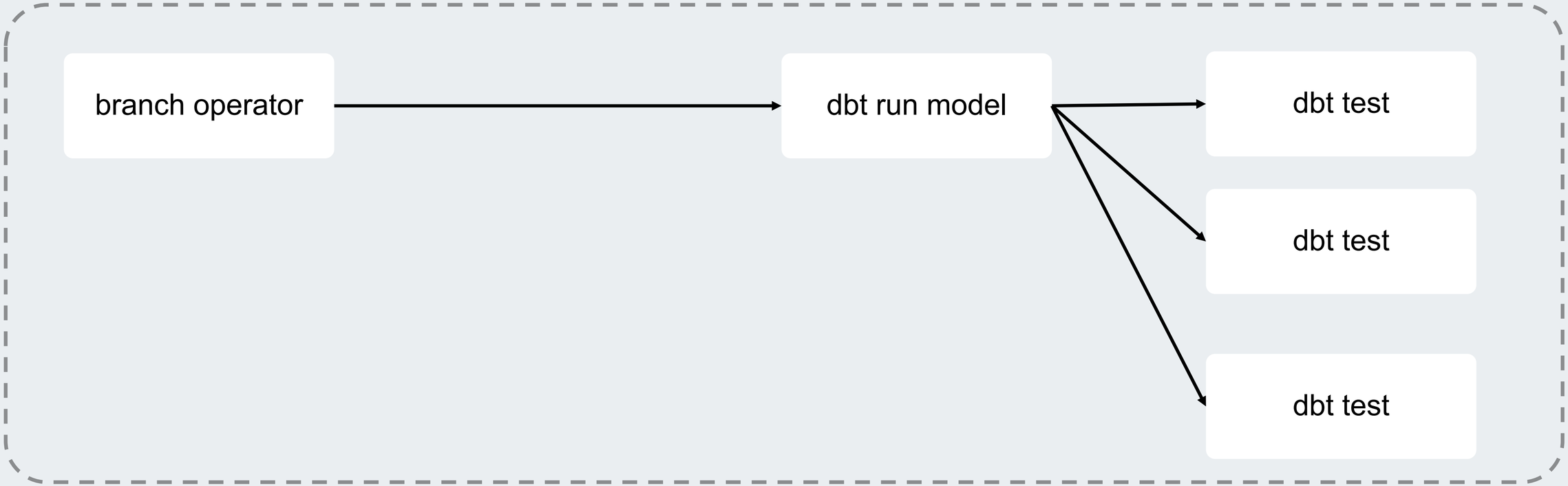
- Основной оператор
- Отвечает за построение модели

- Тестирование модели
- В зависимости от тега стартуют
  - в конце taskgroup
  - в конце dag
  - раз в неделю



model task

- Проверяет, включена ли модель
- Основной оператор
- Отвечает за построение модели
- Тестирование модели
- В зависимости от тега стартуют
  - в конце taskgroup
  - в конце dag
  - раз в неделю



Airflow Dag


dmn_platfrom__rep__order_worker_metrics_agg_daily
dmn_platform__rep__order_stat_agg_overall
dmn_platform__rep__order_stat_agg_daily
dmn_platform__cdm__fct_worker_bonus
dmn_platform__cdm__fct_assignment_issuing
dmn_platform__cdm__fct_pool_activity_daily
dmn_platform__cdm__dim_pool_act
dmn_platform__rep__position_count_daily
dmn_platform__rep__ranking_attention_agg_overall
dmn_platform__cdm__fct_ad_buying
dmn_platform__cdm__fct_task
dmn_platform__cdm__dim_pool_hist
dmn_platform__cdm__fct_order_activity_daily
dmn_platform__cdm__dim_order_act
dmn_platform__rep__task_data_daily
dmn_platform__rep__task_data_line_points

Плюсы\минусы решения

Сильные стороны

- Единый и понятный интерфейс для запуска задач - airflow
- Возможность перезапускать модели и делать backfill
- Интеграция со всеми остальными инструментами

Слабые стороны

- Дополнительная самописная интегарция
- Запуск через BashOperator

Вопрос: как соединить с обычными dags ?

Airflow dataset

С версии airflow 2.4 доступен Data-aware scheduling

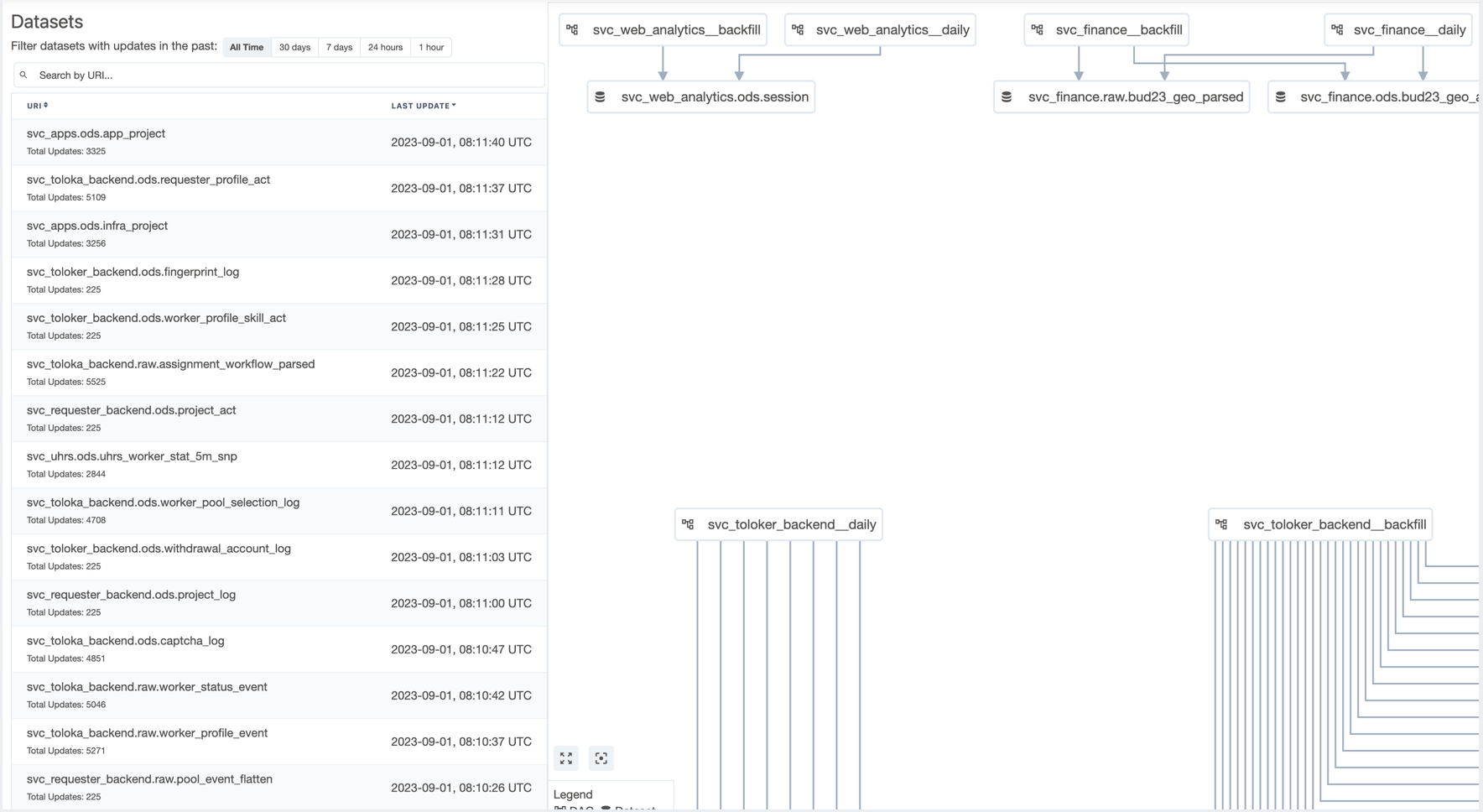
```
from airflow.datasets import Dataset

with DAG(...):
    MyOperator(
        # this task updates example.csv
        outlets=[Dataset("s3://dataset-bucket/example.csv")],
        ...,
    )


with DAG(
    # this DAG should be run when example.csv is updated (by dag1)
    schedule=[Dataset("s3://dataset-bucket/example.csv")],
    ...,
):
    ...
```

Airflow dataset

- Зарегистрировали все dbt модели как dataset в airflow
- Любой даг можно поставить в зависимость от dbt модели



Плюсы\минусы решения

Сильные стороны

- Простая возможность интегрировать dbt и custom dags
- Нативно для airflow

Слабые стороны

- Сложный дебаг в случае ошибок

Вопрос: как уменьшить сложность графа?

Упрощение dag

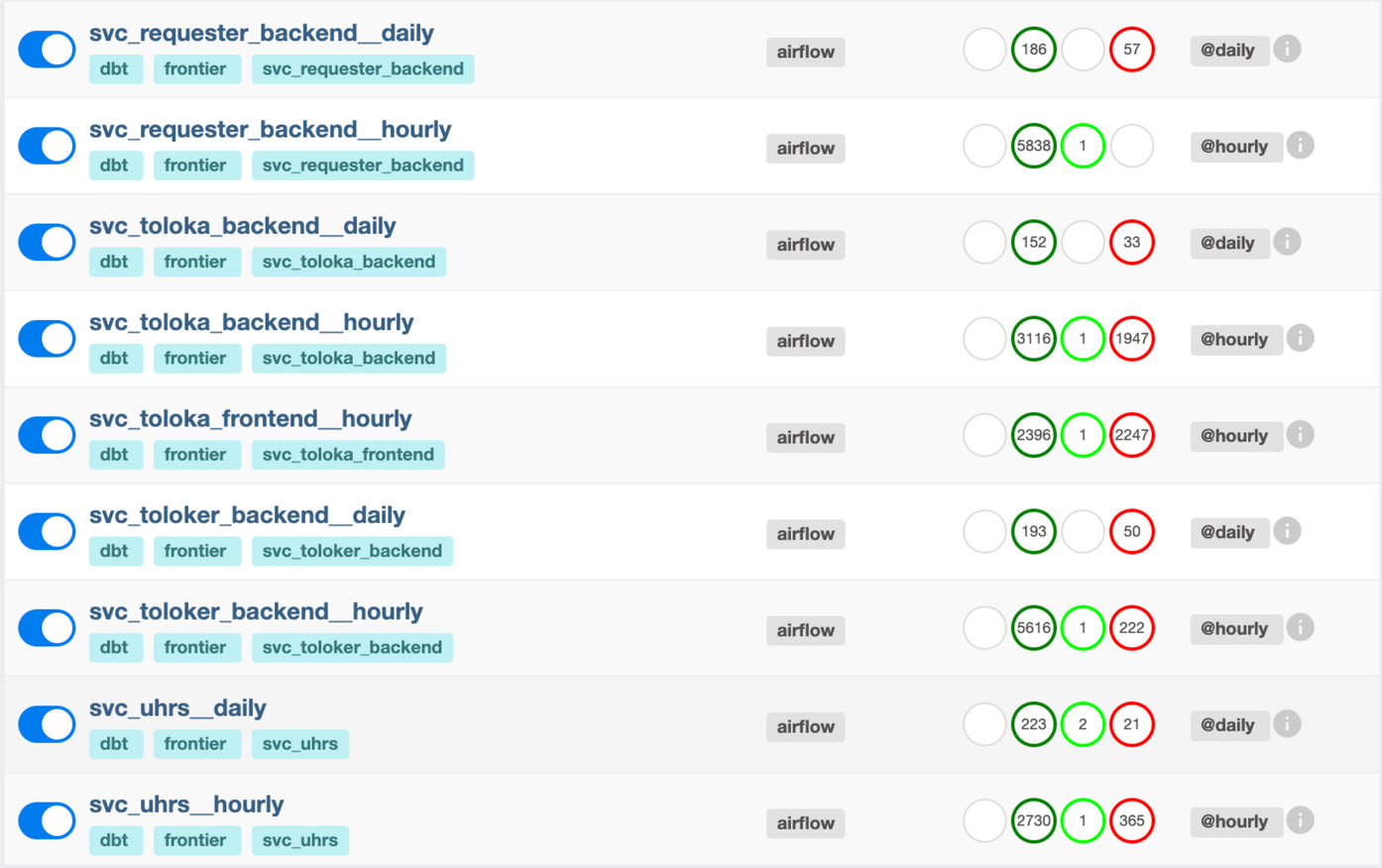
- Доменная организация – каждая модель принадлежит домену
- Модели могут считаться с разной частотой

Dag = Domains X shedule

Упрощение dag

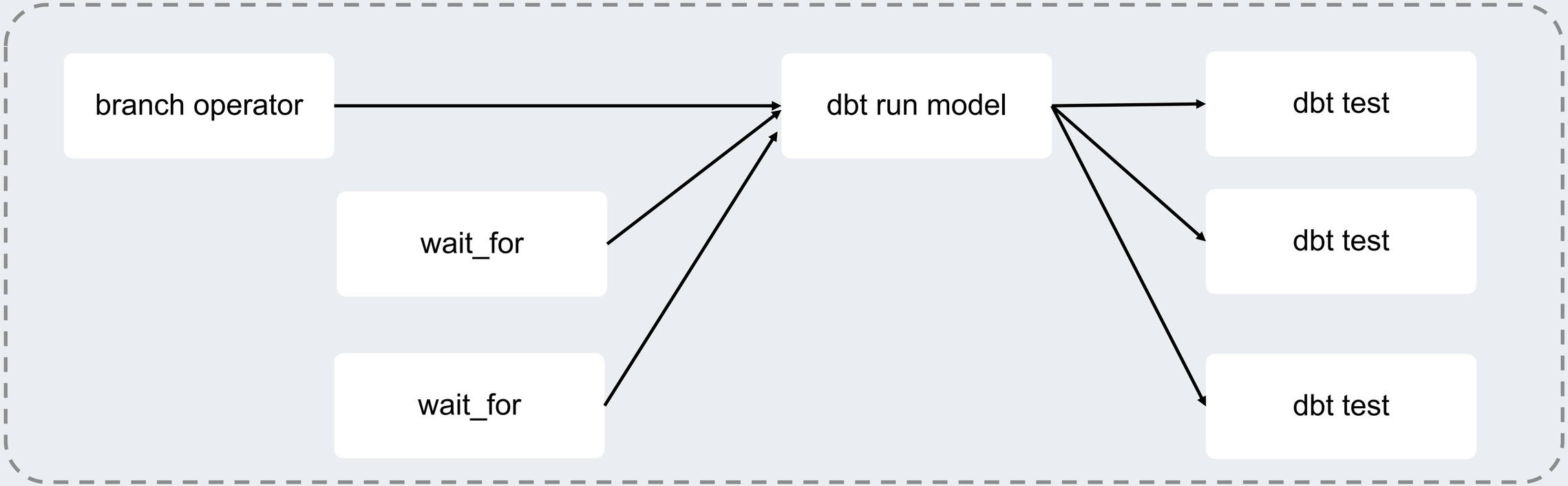
- Доменная организация – каждая модель принадлежит домену
- Модели могут считаться с разной частотой

Dag = Domains X shedule



model task

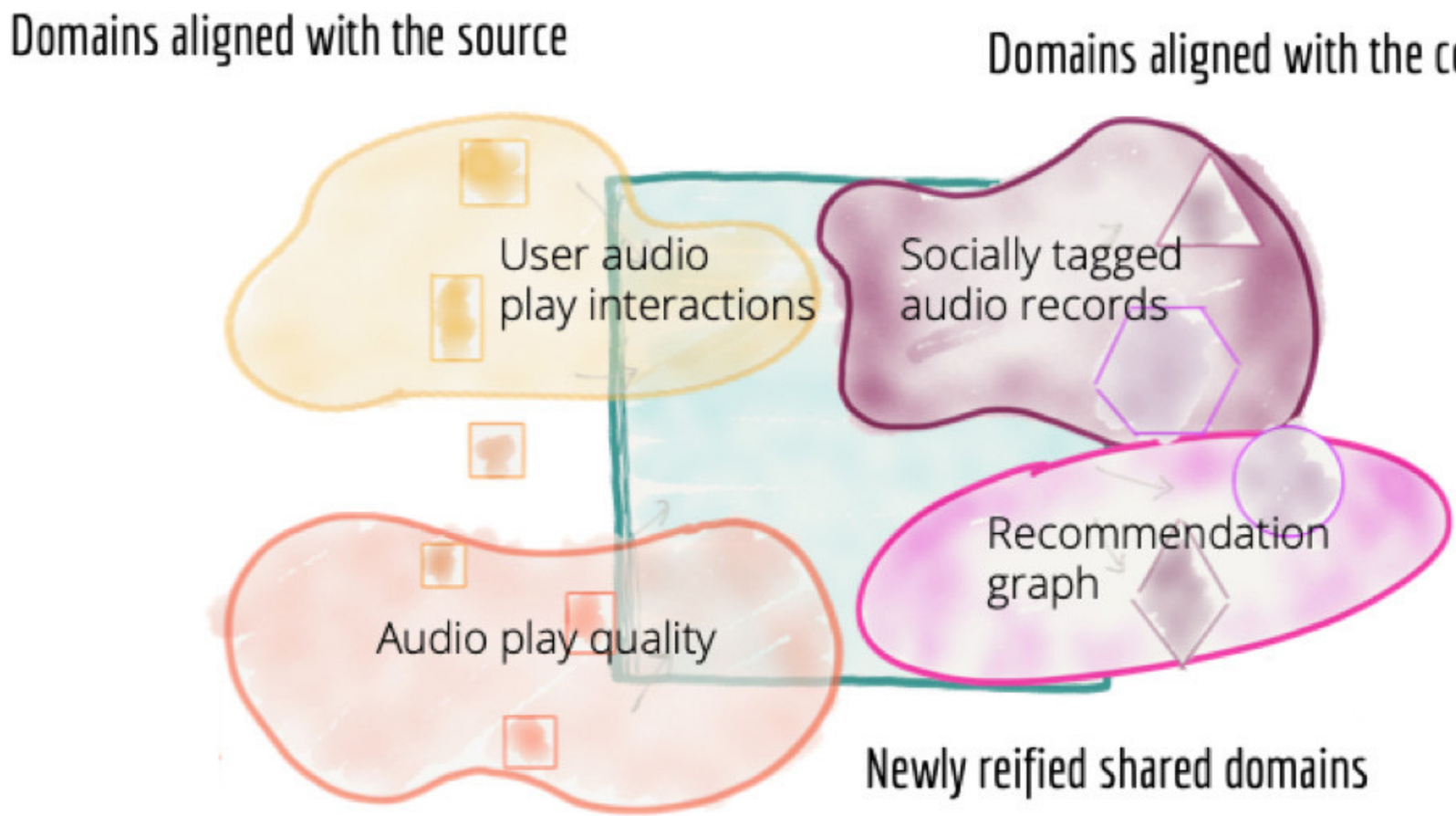
- Проверяет, включена ли модель
- external_task
- Ожидает построения связанной модели
  - из другого домена
  - с другой частотой
- Основной оператор
- Отвечает за построение модели
- Тестирование модели
- В зависимости от тега стартуют
  - в конце taskgroup
  - в конце dag
  - раз в неделю



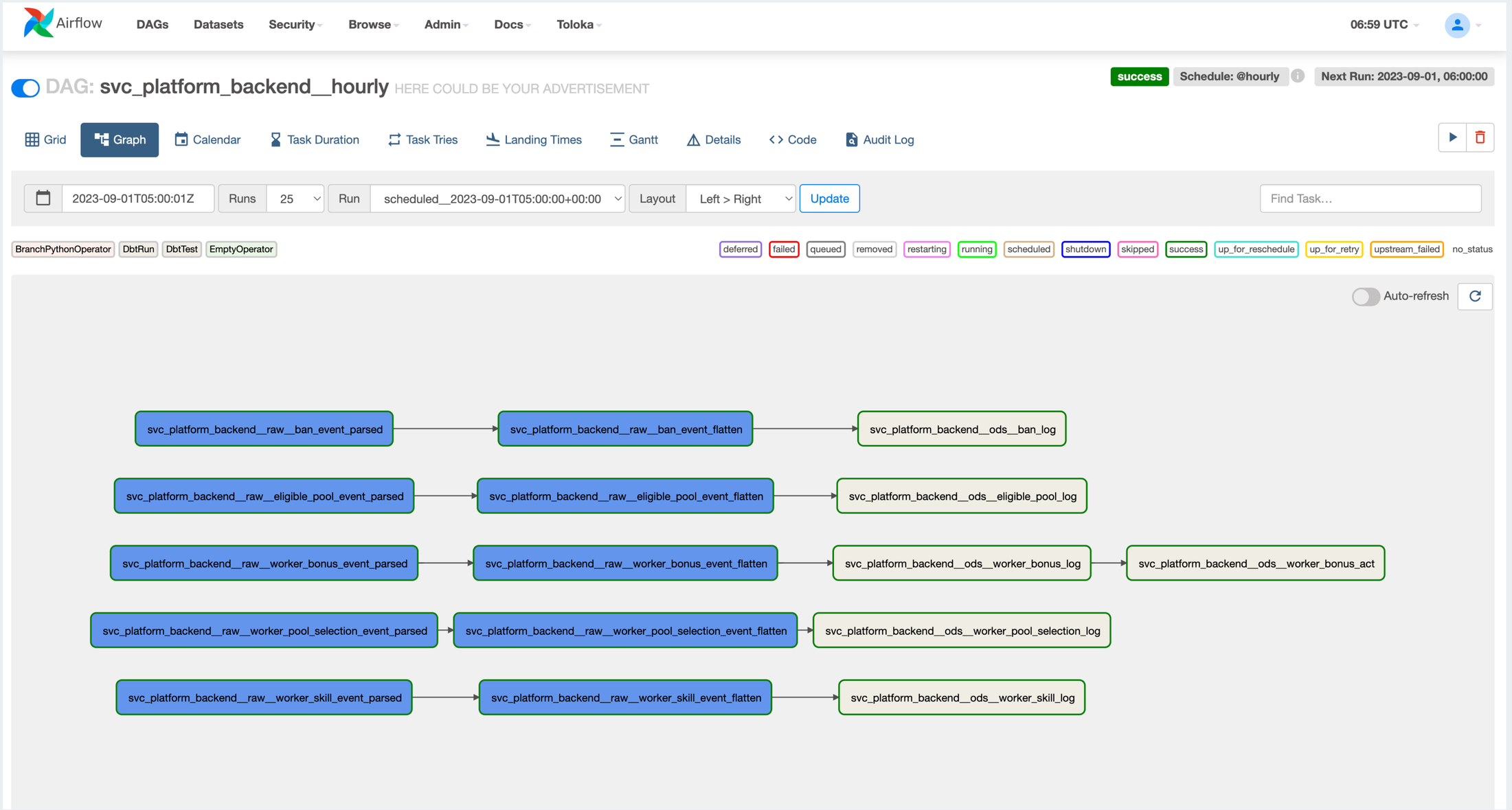
model task

- Проверяет, включена ли модель
- external_task
- Ожидает построения связанной модели
  - из другого домена
  - с другой частотой
- Основной оператор
- Отвечает за построение модели
- Тестирование модели
- В зависимости от тега стартуют
  - в конце taskgroup
  - в конце dag
  - раз в неделю

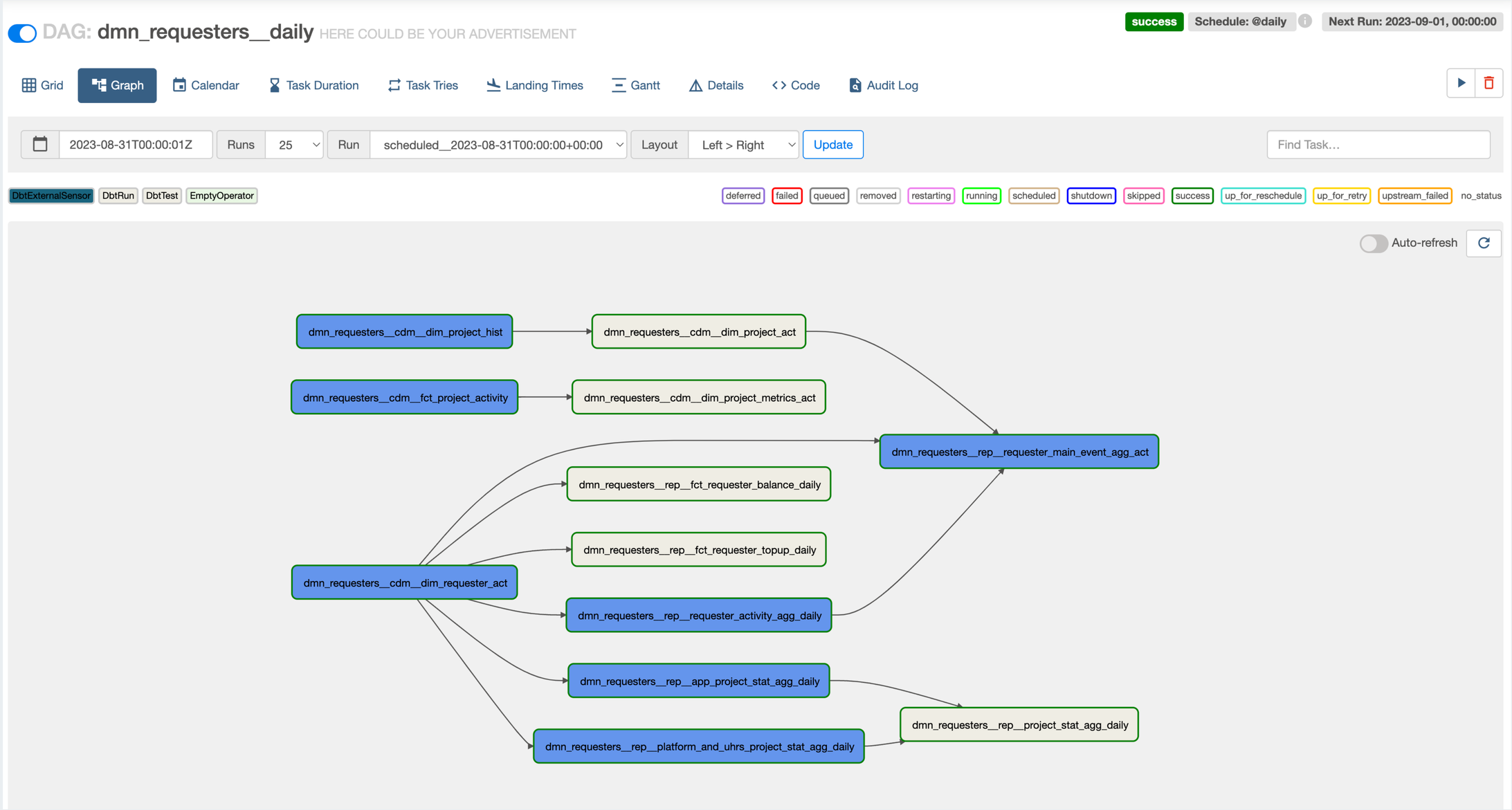
Data Mesh domain types



Classic service (source) domain



Classic analytics (consumption) domain



Плюсы\минусы решения

Сильные стороны

- У каждой доменной команды свой набор графов и контроль за ними
- Ответственность за dag у каждой отдельной доменной команды
- Легко определять зависимости

Слабые стороны

- Крайне сложно проводить backfill в случае ошибки
- Запуск dag по времени может приводить к излишним ожиданиям

4. Выводы

Что у нас получилось?

Dbt at Toloka Data Platform

Ingestion

Event Streaming

kafka

Connectors / EL

Apache Airflow

Transformation

Streaming Processing

Spark

Batch Processing

dbt Spark

Orchestration

Apache Airflow dbt

Storage

Consumption

Quering

Business Intelligence

Analytical tools

Data Governance

Data Quality

dbt

Data Catalog \ Data Observability

dbt

Monitoring

Grafana

Три кита и четыре столпа дата платформы

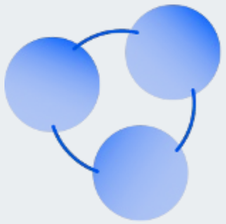
Что?

Архитектура
=
Data Lakehouse

Почему?

Процессы
=
Data mesh

Как?

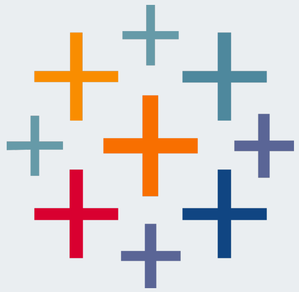


Технологии
=
Modern Data Stack

databricks

dbt

airflow



tableau

Итого

Сильные стороны

- dbt – простой инструмент, скрывающий сложности работы с данными
- dbt+airflow позволяет использовать преимущества обоих продуктов
- dbt позволяет реализовать data mesh через ввод доменов и analytics engineers
- Возможность удобной работы как разработчику, так и аналитику

Слабые стороны

- Слабости монорепы
- Дополнительная самописная библиотека для интеграции
- Сложность во взаимодействии между доменами в случае ошибки

Thank you!

jkermakov@toloka.ai

www.toloka.ai

iJKos.com