dbt Labs DataFun.

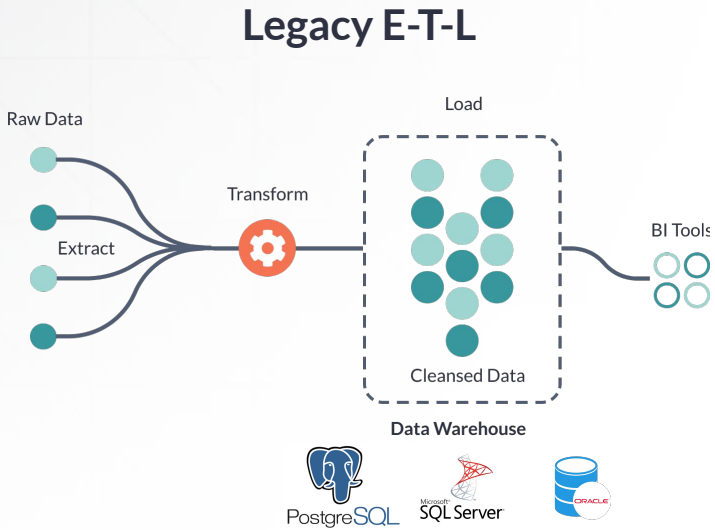
通过 dbt 把软件开发的最佳实践带到数据领域

Chenyu Li, Sr Software Engineer, dbt Labs

因为dbt还是一个比较新的产品，社区也主要集中在美国欧洲，很多材料并没有中文翻译，我会尽量用中文讲解，做的不好的地方还请大家见谅。

- 传统数据分析中的流程问题
- 云原生数仓带来的机会
- dbt 想要提供的解决方案

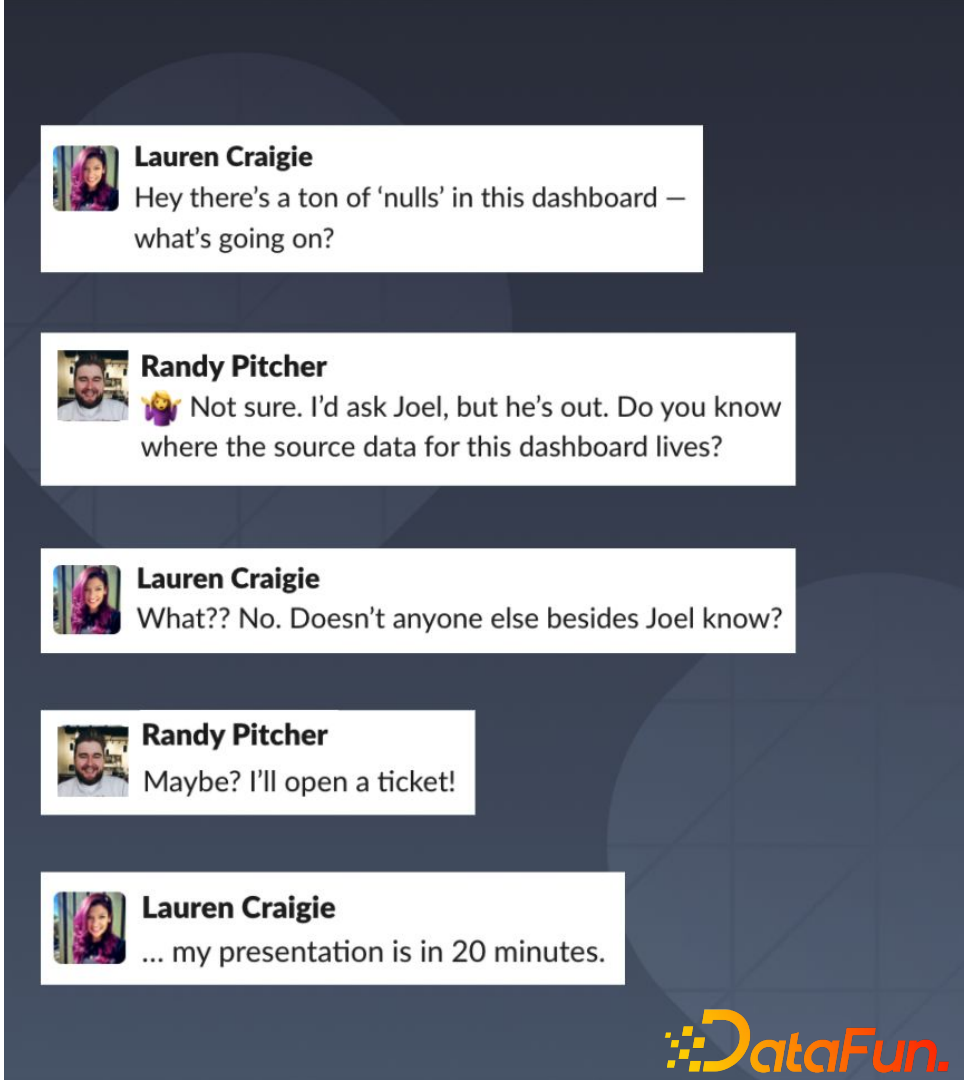
以前的数据仓库非常昂贵



- 数据转换发生在 **数据存储层之外**
- 基础设施的管理是一份**全职工作**
- **数据分析任务分散在** 工程师，数据分析师和 stakeholder中间

传统数据分析中的流程问题

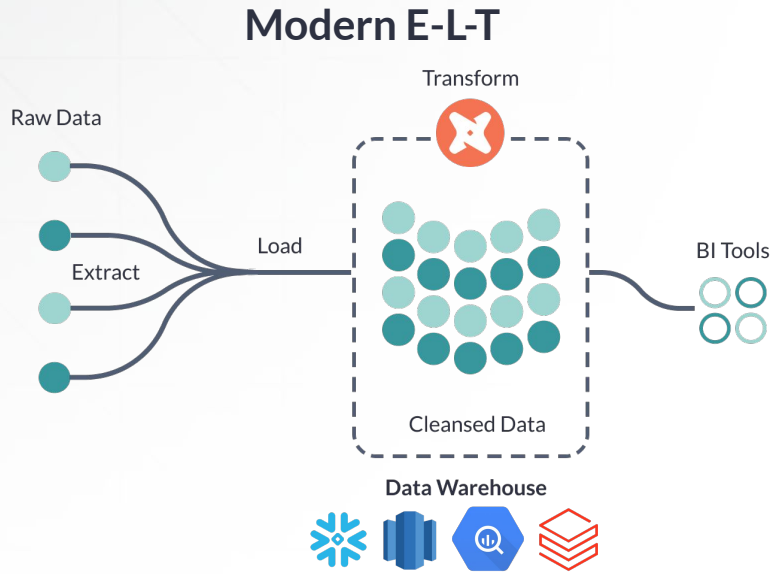
1. 工程师成为 **每一个更改** 的瓶颈
2. **从头创建** 比查找现有代码更简单
3. **不可追踪的变更流程** 破坏数据 pipeline，降低大家对数据的信任.



dbt Labs DataFun.

云原生数仓带来的机会

云原生数仓降低成本，并且更容易使用



- 弹性存储和计算使得 **数仓内转换变得可行.**
- 云原生架构 **减少了基础设施的管理**
- 工程师，分析师可以更加专注在 **高回报的任务上，比如优化和构建数据转换流程**
- **这些变化给数据工作流程创新提供了可能性，dbt在这样的契机下，提出了自己的一套解决方案**

dbt 想要提供的解决方案

The dbt viewpoint: **Build data like developers build applications**

- 模块化
- 可测试
- 持续集成
- 有文档

数据分析流程更快更稳定的更新

```
-- orders.sql

select *
from {{ ref('stg_orders') }}
where is_deleted = false
```

Runs in the warehouse

```
create table analytics.dev.orders as (

select *
from analytics.dev.stg_orders
where is_deleted = false

);
```

how dbt want data teams work together

1. Enable **anyone who knows SQL** to quickly build and test data
2. Use **version control** to **update once** and deploy everywhere
3. Provide **documentation tool and auto-refreshing** lineage

"This isn’t anything new, it’s how every high-quality software project is run. You expect there to be tests. You expect there to be documentation. You expect the PR process to be collaborative. You're building software together.
We're just applying this to analytics code as well."

A centralized environment for collaborative development

Develop

- IDE or CLI
- Modular SQL
- No DDL/DML
- Pre-built packages

Document

- Dependency management
- Auto-generate DAG
- Auto-updated docs

Test

- Schema tests
- Data value testing
- Pre-packaged tests for complex logic

Deploy

- Job scheduling
- CI/CD
- Version control
- Logging & alerting

A centralized environment for collaborative development

Develop

- IDE or CLI
- Modular SQL
- No DDL/DML
- Pre-built packages

Develop faster with SELECT statements(declarative)

- Express business logic in **SQL**
- Includes several **materializations**
  - Table
  - View
  - Incremental
  - snapshot

```
-- orders.sql


select *
from analytics.dev.stg_orders
where is_deleted = false
```

Runs in the warehouse

```
create table analytics.dev.orders as (

select *
from analytics.dev.stg_orders
where is_deleted = false

);
```

Develop faster without having to think about run order

- Run the same code in dev, test and prod– **the correct schema is resolved for you**
- **Dependencies built automatically** so you can focus on modeling, not run order

-- orders.sql

```
select *
from {{ ref('stg_orders') }}
where is_deleted = false
```

Runs in the warehouse

```
create table analytics.dev.orders as (

select *
from analytics.dev.stg_orders
where is_deleted = false

);
```

Develop faster without having to think about run order

- Run the same code in dev, test and prod– **the correct schema is resolved for you**
- **Dependencies built automatically** so you can focus on modeling, not run order

-- orders.sql

```
select *
from {{ ref('stg_orders') }}
where is_deleted = false
```

Runs in the warehouse

```
create table analytics.prod.orders as (

select *
from analytics.prod.stg_orders
where is_deleted = false

);
```

Macros

- A sandbox environment to execute user logic
- Abstract snippets of SQL into reusable macros — these are analogous to functions in most programming languages.
- Use control structures (e.g. if statements and for loops) in SQL
- Use environment variables in your dbt project for production deployments
- Operate on the results of one query to generate another query

Use packages to skip boilerplate code

- Apply industry standard code to your project
  - Check out the dbt Packages Hub
- Akin to python libraries
- Get to focusing on unique business logic rather than implementing something people have already solved for
- Types of packages:
  - Transforming data from a structured SaaS dataset
  - Writing dbt macros to answer “How do I do this in SQL?”(i.e. `Dbt_utils.equal_rowcount, date conversion`)
  - Auditing & Testing

A centralized environment for collaborative development

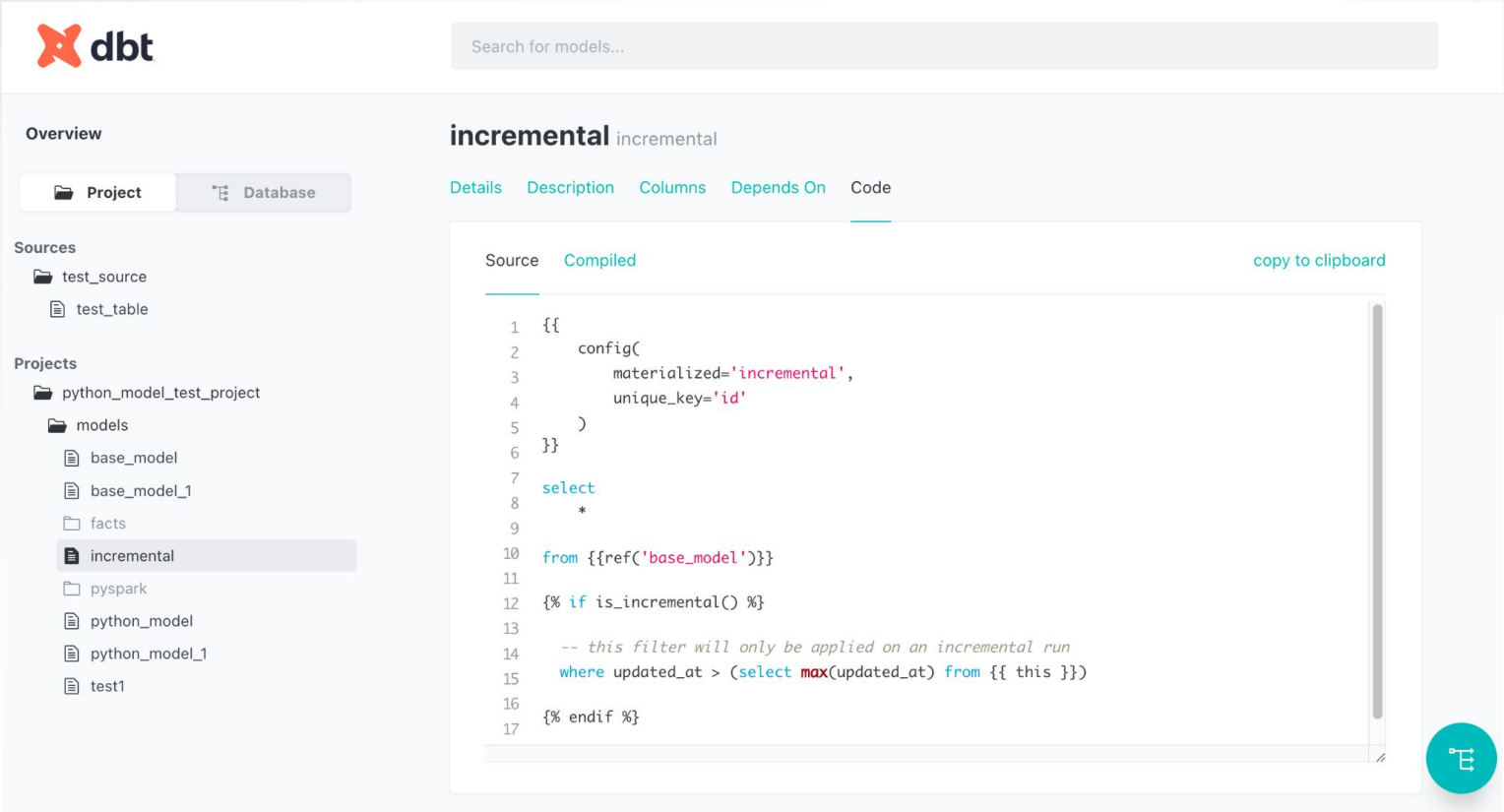
Develop

- IDE or CLI
- Modular SQL
- No DDL/DML
- Pre-built packages

Document

- Dependency management
- Auto-generate DAG
- Auto-updated docs

Maintain shared understanding with auto-updating lineage


dbt
Search for models...
Overview
Project
Database
Sources
test_source
test_table
Projects
python_model_test_project
models
base_model
base_model_1
facts
incremental
pyspark
python_model
python_model_1
test1
incremental incremental
Details
Description
Columns
Depends On
Code
Source
Compiled
copy to clipboard
{{
config(
materialized='incremental',
unique_key='id'
)
}}
select
*
from {{ref('base_model')}}
{% if is_incremental() %}
-- this filter will only be applied on an incremental run
where updated_at > (select max(updated_at) from {{ this }})
{% endif %}

A centralized environment for collaborative development

Develop

- IDE or CLI
- Modular SQL
- No DDL/DML
- Pre-built packages

Document

- Dependency management
- Auto-generate DAG
- Auto-updated docs

Test

- Schema tests
- Data value testing
- Pre-packaged tests for complex logic

Preserve quality by testing in-line

- Test **assumptions** about data, and the **validity** of transformations
- **Custom + out of the box** tests including:
  - Uniqueness
  - Null values
  - Certain values
  - Is a valid foreign key to another table

schema.yml

```yaml
columns:
  - name: order_id
    tests:
      - unique
      - not null
  - name: status
    tests:
    - accepted_values:
        values: ['placed', 'shipped', 'completed']
```

tests/assert_payment_amount_is_positive.sql

```sql
select
    order_id,
    sum(amount) as total_amount
from {{ ref('fct_payments' )}}
group by 1
having not(total_amount >= 0)
```

A centralized environment for collaborative development

Develop

- IDE or CLI
- Modular SQL
- No DDL/DML
- Pre-built packages

Document

- Dependency management
- Auto-generate DAG
- Auto-updated docs

Test

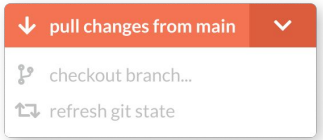
- Schema tests
- Data value testing
- Pre-packaged tests for complex logic

Deploy

- Job scheduling
- CI/CD
- Version control
- Logging & alerting

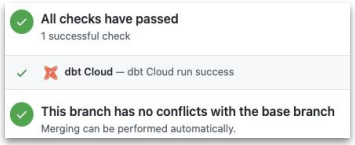
Deploy seamlessly with version control and CI/CD

Version Control



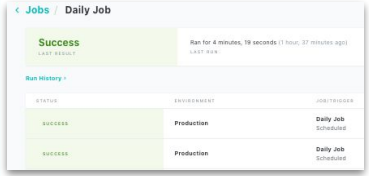
Integrate with git provider of choice

Continuous Integration



Minimize wasteful runs by testing only changes

Continuous Deployment

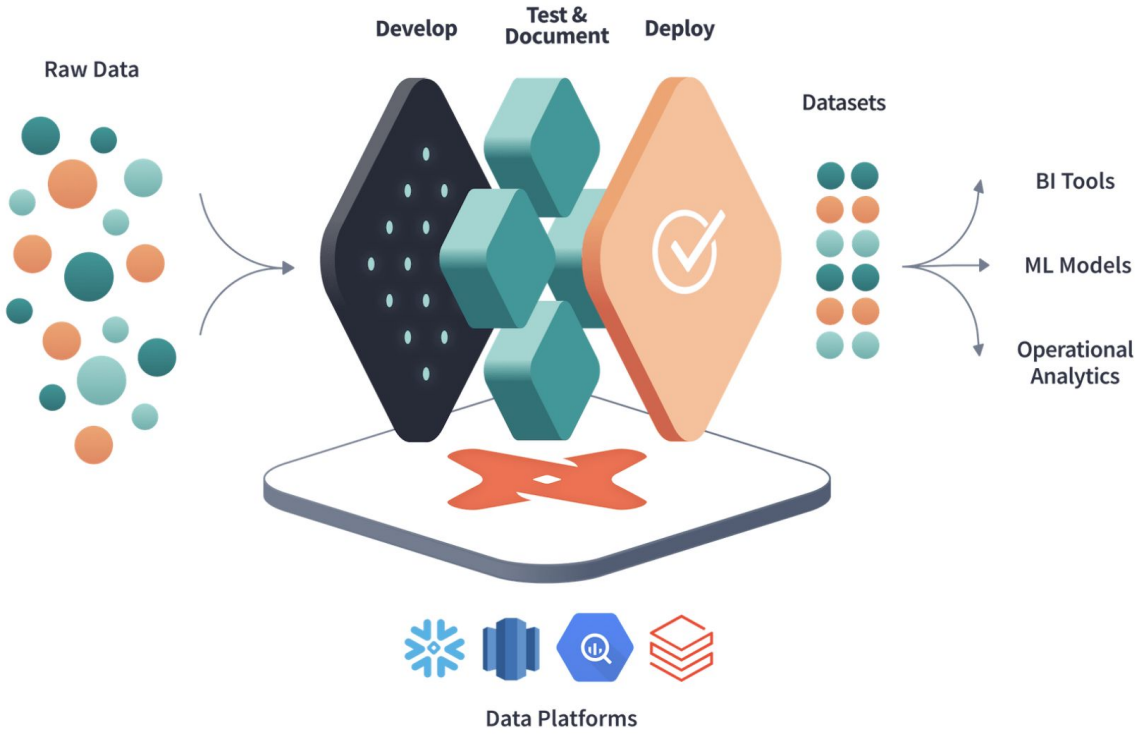


Job scheduling and alerting

Logging & Alerting

The analytics engineering workflow

With dbt, data teams work directly within the warehouse to produce trusted datasets for reporting, ML modeling, and operational workflows.



Thank you! Questions?