Week06｜课时 1｜为什么脚本跑通不等于数据产品可运营

Table of contents

跑通脚本只是开始，能运营资产才是系统能力

这一讲先立住 Week06 的问题意识：

AI 数据工程的失败，很多时候不是脚本没跑，而是团队不知道哪些数据资产处于什么状态。

进入课时 2 返回 Week06 总览

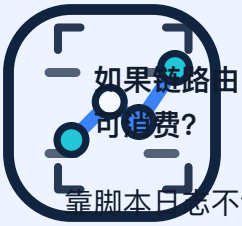
下载讲义

提供适合离线阅读的 PDF 版和适合批注整理的 Word 版。

PDF 版 · 打印 / 离线阅读 Word 版 · 批注 / 二次整理

这节课解决什么问题

Week03 已经让我们有了 ingest、checkpoint、replay、backfill 的脚本思维。问题是：



如果链路由多个脚本、表、指标和未来索引组成，团队靠什么判断今天的数据产品是否可消费？

靠脚本日志不够，靠口头顺序不够，靠“我刚才跑过了”更不够。

参考学习时间

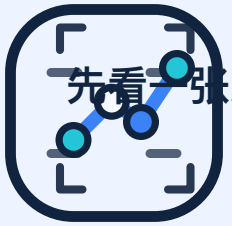
45–55 分钟

学完这一讲，你应该能做到什么

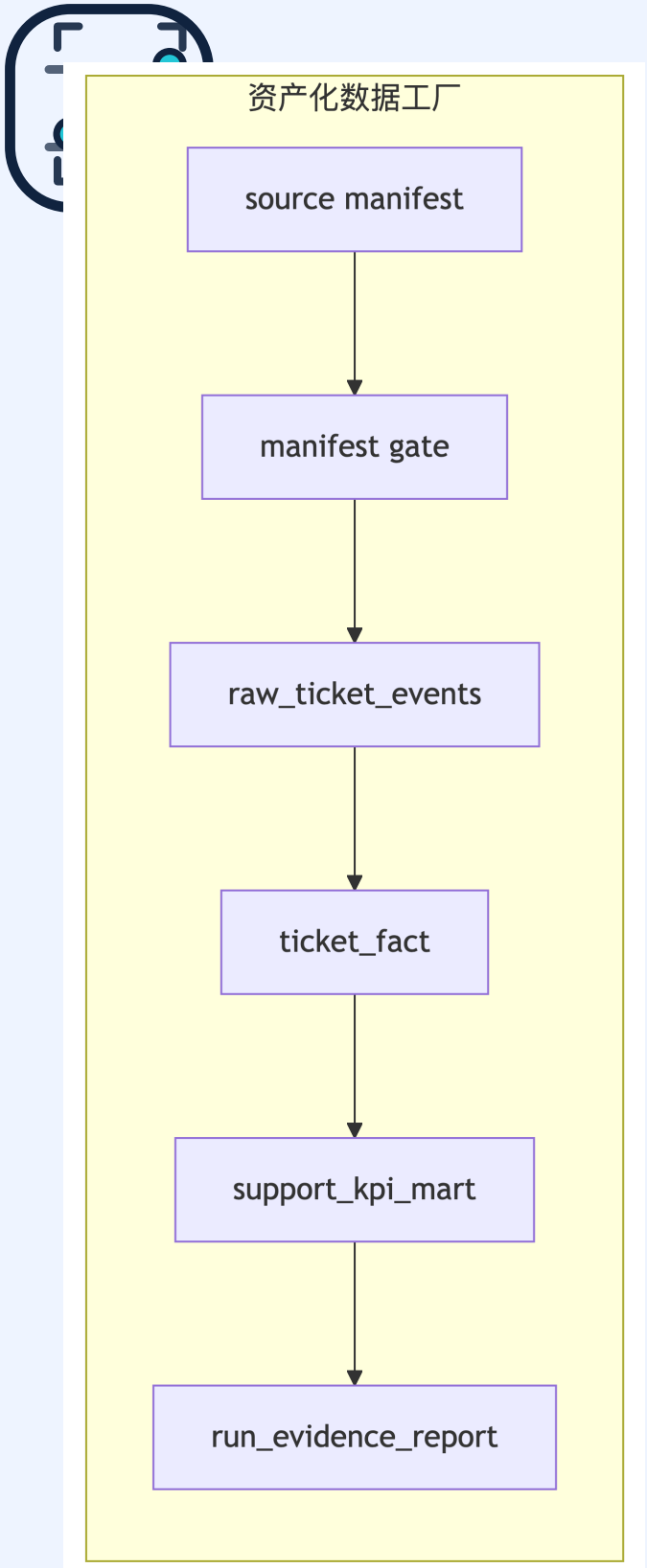
1. 区分“任务流”和“资产流”。
2. 解释为什么 Week03 的 replay/backfill 需要升级成 asset-level recovery。
3. 说清 AI 数据链路为什么必须有资产状态、分区状态和运行证据。
4. 识别脚本堆叠带来的交接、回填和下游阻断风险。
5. 说明 Week06 为什么不是 Dagster UI 教程。

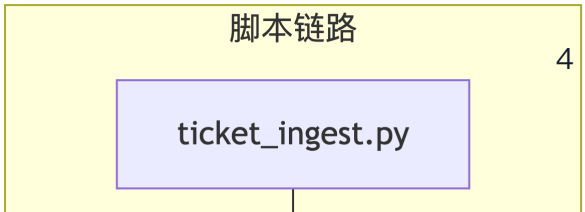
本课产出

- docs/blueprints/week06/week06-data-factory-blueprint.md
- docs/blueprints/week06/week06-asset-graph.md



先看一张总图


资产化数据工厂
source manifest
manifest gate
raw_ticket_events
ticket_fact
support_kpi_mart
run_evidence_report


脚本链路
4
ticket_ingest.py

这张图的差别不在“换了工具”，而在：

- 左边关心脚本是否跑过
- 右边关心资产是否可消费
- 左边依赖人工记忆
- 右边记录依赖、分区、检查和证据

1. 脚本跑通为什么仍然不可运营

先看一个课堂里很常见的 bad case：

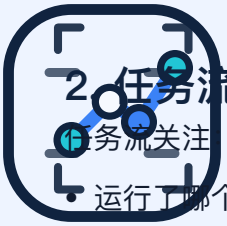
> ⚠ 事故小剧场
>
> A 同学在本机跑完 ticket_ingest.py，日志里出现 success。B 同学第二天接手时只知道“昨天跑过”，但不知道 ticket_fact 哪个分区完成、Week05 KPI mart 是否同步、run evidence 是否生成。下游如果继续做 Week08 index，很可能把一个缺证据的状态当成正式 baseline。

脚本式 pipeline 最容易留下四类盲区：

盲区	表现	后果
顺序盲区	只知道先跑哪个脚本	换人后无法复现
状态盲区	不知道哪个分区成功	补数靠猜
质量盲区	跑完没有 checks	坏数据进入下游
证据盲区	没有 run evidence	评测、追踪、治理断链

团队真正会问的不是“脚本是不是跑过”，而是：

1. 今天哪些 asset 可以被下游消费?
2. 哪个 partition 是最新可信状态?
3. 上游 manifest 是哪一版?
4. 输入和输出行数是否对得上?
5. 有没有重复写入或漏数?
6. 如果失败，应该 retry、replay 还是 backfill?
7. Week04 / Week05 依赖没启用时，是 blocked、skipped 还是 not_available?
8. 证据 JSON 在哪里，schema 是否校验过?
9. 谁能批准下游 unblock?
10. 另一个人能不能照 runbook 复现恢复路径?



2. 任务流和资产流的区别

任务流关注：

- 运行了哪个脚本
- 脚本 exit code 是什么
- log 里有没有报错

资产流关注：

- 哪个 asset 被 materialize
- 上游依赖是否满足
- 哪个 partition 成功或失败
- 哪些 checks 通过
- 证据 report 在哪里
- 下游是否应该放行

> ! 核心判断
>
> **任务流回答“我跑了什么”；资产流回答“系统现在能相信什么”。**

任务日志 / Pipeline Run / Asset State / Run Evidence

对象	它能说明什么	它不能替代什么
脚本日志	某段代码运行时发生了什么	不能证明下游可消费
pipeline run	某次编排是否结束	不能表达具体 asset 的业务状态
asset state	某个资产和分区的 materialization / check 状态	不能替代结构化证据归档
run evidence	运行事实、reason code、report path、downstream decision	不能替代代码测试和人工 runbook

3. 为什么 AI 数据工程更需要资产流

AI 数据系统的下游不只是看板，还会包括：

- Week07 文档解析与 chunk
- Week08 检索和证据服务

- Week10 工具调用
- Week11 评测
- Week14 治理和发布追踪

这些系统不会关心你昨晚跑了哪个脚本，它们关心：

- 输入来自哪个 manifest
- 哪个 partition 已完成
- 数据是否重复或漏数
- run evidence 是否完整
- release / trace / git sha 是否能追溯

4. Week06 的最小升级路径

Week06 不是重写 Week03–Week05，而是做薄包装：

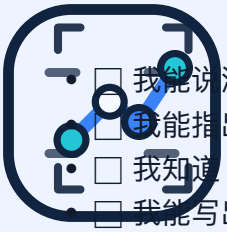
```
existing ingest scripts
  -> Dagster assets
  -> daily partitions
  -> asset checks
  -> run evidence
  -> data factory runbook
```

常见误解

误解	纠偏
Dagster 的价值是 UI 好看	UI 只是表象，核心是 asset state model
脚本能重跑就等于可恢复	没有分区边界和幂等证据，重跑仍然危险
有日志就有证据	日志是过程材料，run evidence 才是可校验交付物
有 Dagster UI 就有数据工厂	没有 asset key、checks、evidence、runbook，UI 只是展示层
资产图越复杂越专业	Student Core 先要少量关键资产和清晰命名
runbook 是最后补文档	runbook 是系统设计的一部分

自检清单

- [ ] 我能解释任务流和资产流的差别



- [ ] 我能说清为什么“跑过”不等于“可消费”
- [ ] 我能指出 Week03 replay/backfill 的资产化升级方向
- [ ] 我知道 Week06 不能抢跑 Week07 / Week08 / Week14
- [ ] 我能写出一条最小资产化升级路径
- [ ] 我能说明失败后应该补哪个 partition，而不是只说“重跑一下”
- [ ] 我能把运行证据交给另一个人复核

课后最小行动

在项目笔记中写下：

```
## Week06 operational asset risks

- 哪些脚本目前只靠人工顺序运行
- 哪些数据资产缺少分区状态
- 哪些结果缺少 checks
- 哪些 report 可以作为 run evidence 起点
```