# Week03｜课时 4｜从任务流到资产流：用 Dagster 组织 ingest、分区与回放

## Table of contents


先把 ingest 组织成可理解的资产流，再谈更复杂的治理闭环 ..... 2
这节课解决什么问题 ..... 2
参考学习时间（50–60 分钟） ..... 3
学完这一讲，你应该能做到什么 ..... 3
本课产出 ..... 3
先看一张总图：为什么“任务流”不够 ..... 4
1. 为什么 Week03 一定要从任务流走向资产流 ..... 4
2. 先看当前 repo 里已经有什么 ..... 5
    2.1 项目已经不是“只有脚本，没有资产化入口” ..... 6
    2.2 这节课的正确姿势不是大重构，而是读懂并扩展 ..... 6
3. 资产流的 5 个关键概念 ..... 6
    3.1 Asset ..... 6
    3.2 Materialization ..... 7
    3.3 Partition ..... 7
    3.4 Backfill ..... 7
    3.5 Asset Check ..... 8
    把这 5 个概念放到一张表里 ..... 8
4. 再看一张图：Week03 里 asset flow 怎么接住 manifest 和 ingest ..... 9
5. 为什么 partition 和 backfill 现在就要讲，而不是等 Week06 再说 ..... 9
    现在就要建立的判断 ..... 10
6. 这节课的实践不会让你重写 Dagster，而是让你学会“读 + 记 + 计划” ..... 10
    Step 1：先确认基础服务已启动 ..... 10
    Step 2：先跑一次最小 ingest 基线 ..... 10
    Step 3：再看 Dagster 资产图 ..... 11
    Step 4：记录一次最小 materialization smoke report ..... 11
    Step 5：补一份资产流规划 ..... 11
7. 你现在要学会的不是“把 Dagster 用得很花”，而是 3 个工程判断 ..... 12
    判断 1：manifest 不是资产 ..... 12
    判断 2：job 成功不等于下游可以安全消费 ..... 12

判断 8：backfill 必须围绕 asset + partition 做，不要围绕“脚本名字”做 ..... 12
8. 这节课的产出为什么会直接影响 Week04 和 Week06 ..... 12
对 Week04 的影响 ..... 12
对 Week06 的影响 ..... 12
9. 本课最重要的 8 个判断 ..... 12
10. 自检清单 ..... 13
11. 课后最小行动 ..... 13
延伸阅读 ..... 14

## 先把 ingest 组织成可理解的资产流，再谈更复杂的治理闭环

这一讲不打算把 Dagster 讲成调度平台百科。

先解决更关键的问题：

**为什么“任务成功”不等于“资产就绪”，以及为什么 Week03 应该先把 ingest asset 讲清，而不是直接跳到更高层的湖仓资产。**

进入课时 5 回看课时 3 返回 Week03 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

当 ingest 开始进入稳定运行阶段，单纯依赖：

- cron
- shell script
- “今天跑完了就算成功”

很快就会失控。

这一讲要回答 4 个问题：

1. 为什么单纯 cron + script 很快会失控
2. 为什么“任务成功”不等于“资产就绪”
3. partition / batch window / backfill policy 有什么工程意义
4. 怎样把当前 repo 里的 ingest baseline 组织成可以被 Dagster 理解的资产流

## 参考学习时间（50–60 分钟）

如果你只阅读正文，大约需要 30–35 分钟；如果你跟着本课一起对照 assets.py、definitions.py 和 Dagster UI 补齐 asset_flow_plan_v1.md 与 partition_backfill_strategy_v1.md，建议预留 50–60 分钟。

## 学完这一讲，你应该能做到什么

完成这一讲后，你应该能够：

1. 解释为什么“任务成功”不等于“数据资产已经可被下游安全消费”。
2. 看懂 OmniSupport Copilot 当前 Dagster 资产入口是如何把 manifest -> raw asset 串起来的。
3. 理解 asset / partition / backfill / asset check 在采集链路里的真实作用。
4. 为 Week03 的双源 ingest baseline 写出一份最小资产流设计方案。
5. 知道 Week04 / Week06 会如何继续吃掉这节课定义的资产边界。

## 本课产出

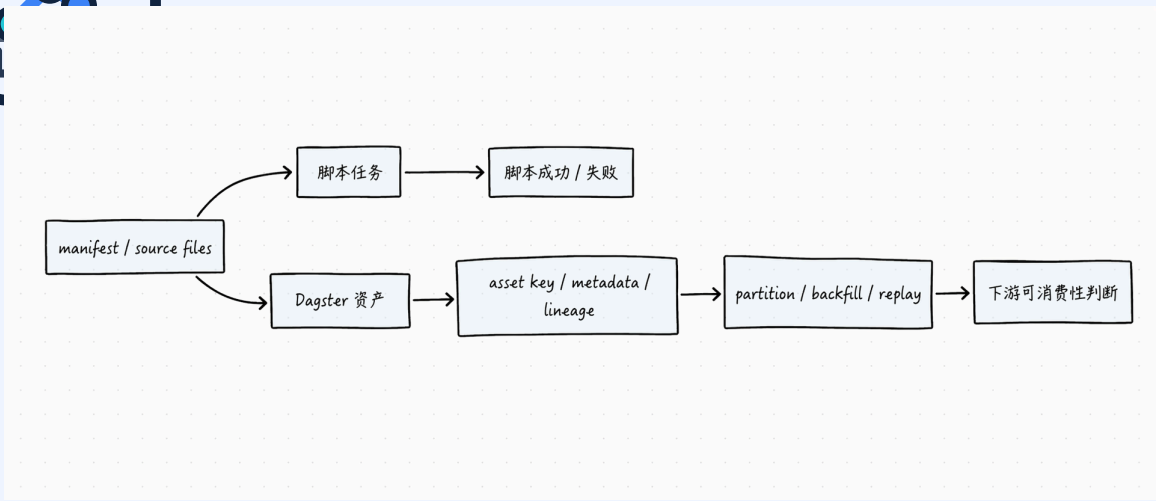
完成本课后，你至少应该产出下面 3 个文件：

- docs/blueprints/week03/asset_flow_plan_v1.md
- docs/blueprints/week03/partition_backfill_strategy_v1.md
- reports/week03/dagster_materialization_smoke_report.md

这 3 个文件的作用分别是：

| 文件 | 作用 |
| --- | --- |
| asset_flow_plan_v1.md | 把 Week03 采集链路从脚本步骤改写成资产视角 |
| partition_backfill_strategy_v1.md | 说明未来如何按时间 / 批次边界做回填与补数 |
| dagster_materialization_smoke_report.md | 记录一次最小 materialization 的观察结果 |

## 先看一张总图：为什么“任务流”不够




任务流最擅长回答的是：

- 哪个脚本跑了
- 哪个脚本失败了
- 哪个任务耗时长

但数据工程真正更关心的是：

- 哪个**资产**已经产生
- 哪个**资产分区**缺失
- 哪个**资产版本**不可信
- 哪个**下游资产**应该阻断

这也是为什么 Dagster 会把“资产”而不是“任务”放在一等公民位置。官方文档直接把 asset 定义成持久化存储中的对象，比如表、文件或模型；asset definition 则是“这个资产应该存在、以及如何生成它”的代码描述。[1] [2]

## 1. 为什么 Week03 一定要从任务流走向资产流

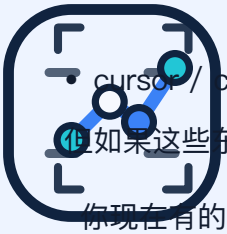
你在前 3 课已经逐步得到：

- contract
- manifest
- ingest baseline

---

[1] Dagster 官方指南把 asset 定义为持久化存储中的对象，asset definition 则是代码里对“这个资产应该存在以及如何生成它”的描述。Dagster Docs | Assets

[2] Dagster API 文档进一步把 software-defined asset 描述为“asset key + 计算函数 + 上游依赖”的组合。Dagster Docs | Assets API



- cursor / checkpoint / replay 思维

但如果这些东西还只是几个脚本之间的约定，真正出故障时，你还是会遇到下面的问题：

| 你现在有的东西 | 还是不够的地方 |
| --- | --- |
| manifest | 它声明了这次 ingest 想做什么，但没有天然表达“哪些资产已经 materialized” |
| seed_loader.py / ticket_ingest.py / doc_ingest.py | 它们能做事情，但还没自动告诉你“资产关系”和“分区边界” |
| batch_id / release_id | 它们能帮助追踪，但还需要一个资产图来解释“谁依赖谁” |
| backfill / replay 思维 | 你知道要补，但还需要明确“补哪一个资产、哪一个分区、哪一个窗口” |

所以课时 4 不是“把脚本替换成 Dagster”，而是：

**让 Week03 当前已经有的 ingest 能力，第一次有了一张资产图。**

## 2. 先看当前 repo 里已经有什么

这一节不要从“理想中的 Dagster 项目”开始，而要从当前 OmniSupport Copilot 的真实入口开始。

当前 repo 里已经有两个很关键的文件：

- pipelines/ingestion/assets.py
- pipelines/definitions.py

在 assets.py 里，当前已经定义了：

- seed_manifests
- raw_doc_assets
- raw_ticket_events
- ingest_all_job

而 definitions.py 则把 ingestion、parse_normalize、lakehouse 这些模块里的资产统一注册到 Dagster Definitions 中。[3][4]

[3] 当前 pipelines/ingestion/assets.py 已定义 seed_manifests、raw_doc_assets、raw_ticket_events 和 ingest_all_job，并明确注释“Week03 起接入真实采集器，打通 MinIO 落盘与 PostgreSQL 元数据写入”。assets.py

[4] 当前 pipelines/definitions.py 已把 ingestion、parse_normalize、lakehouse 三类 assets 统一注册到 Dagster Definitions。definitions.py

这意味着什么?

### 2.1 项目已经不是“只有脚本，没有资产化入口”

虽然 Week03 还没有把所有 ingest 逻辑 fully assetized，但当前 repo 已经把最重要的方向铺好了：

- `seed_manifests` 是资产流的入口
- `raw_doc_assets` / `raw_ticket_events` 是 Bronze 层的最小落点
- `ingest_all_job` 已经把 Week01 的“会跑”推进到 Week03 的“可以组织成一次资产 materialization”

### 2.2 这节课的正确姿势不是大重构，而是读懂并扩展

所以你在课时 4 最重要的动作不是：

- 重写 Dagster 全栈
- 自己造另一套 orchestration
- 立刻补全所有 sensors / schedules / partitions / auto-materialize

而是先学会：

1. 读懂现有资产图
2. 识别哪些资产已经存在
3. 识别哪些资产还需要补齐 metadata / partition / backfill policy
4. 为 Week04 / Week06 做好接力准备

## 3. 资产流的 5 个关键概念

Dagster 文档很多，但在当前课程阶段，你只需要先把 5 个概念连成一套。

### 3.1 Asset

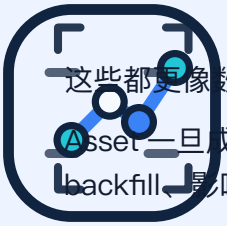
Asset 不是“函数”，而是“持久化结果”。

在当前项目里，你可以先把它理解成：

- 一份清单
- 一张表
- 一个原始落盘文件集合
- 一批经过规范化的记录

例如：

- `seed_manifests`
- `raw_ticket_events`
- `raw_doc_assets`



这些都更像数据资产，而不是任务步骤。

Asset 一旦成立，你就应该顺手关心它的 asset key。因为没有稳定的 key，后面的 lineage、backfill、影响分析就很容易混乱。

## 3.2 Materialization

Materialization 不是“任务跑了”，而是“这份资产被成功产出了一次，而且这次产出最好能带着 metadata”。

对 Week03 来说，materialization 最重要的价值是：

- 让每一次 ingest 不只是执行日志，而是资产级结果
- 把 manifest、batch_id、run_id、source coverage 这些运行证据挂到资产产出上
- 让 Week04 / Week06 后面要做的 lineage、回放、补数有稳定抓手

## 3.3 Partition

Partition 解决的是“这份资产是不是可以按子集管理”。

Dagster 官方文档明确把 partitioning 描述为管理大数据集、提升性能和实现 incremental processing 的强工具。[5]

对 Week03 来说，它不是抽象概念，而会直接落到这些场景：

- 按天 ingest 工单事件
- 按 batch / snapshot ingest 文档清单
- 未来按时间窗口 replay / backfill
- Week06 用分区进行精准补数

## 3.4 Backfill

Backfill 不是“再跑一遍”，而是对**一组缺失或需要重算的分区资产**进行有边界的补跑。

Dagster 官方文档对 backfill 的定义很清楚：它是对不存在或需要更新的分区资产执行补跑。[6]

这对当前课程的意义是：

- 你不应该把 backfill 理解成“遇到问题就整链路重跑”
- 真正的 backfill 一定要和 **asset + partition** 连起来看

---

[5] Dagster 官方文档把 partitions 描述为管理大型数据集、提升性能和实现 incremental processing 的关键技术。Dagster Docs | Partitioning Assets

[6] Dagster 官方文档把 backfill 定义为对不存在或需要更新的分区资产执行补跑。Dagster Docs | Backfilling Data

### 3.5 Asset Check

Asset check 不是另一个“测试框架”，而是附着在资产上的可执行性质检查，例如：

- 某个字段不能为空
- 某个时间窗口的记录数不能低于阈值
- 某个分区的 schema 或 quality 状态必须满足条件

Dagster 把 asset check 定义为验证资产某个属性的机制。[7]

对 Week03 来说，现在不要求你立刻把所有检查都写进 Dagster，但你必须先形成一个判断：

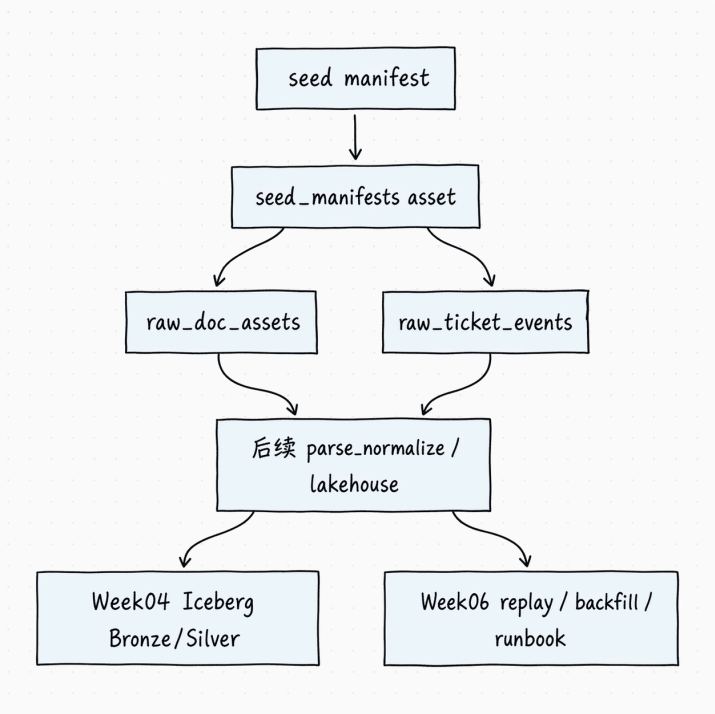
**后面很多 gate，并不是“脚本外置规则”，而是资产本身的健康约束。**

## 把这 5 个概念放到一张表里

| 概念 | 你在 Week03 里最该抓住什么 |
| --- | --- |
| asset | 采集链路真正要持续存在的结果对象 |
| materialization | 一次资产被成功产出的证据，以及它附带的运行 metadata |
| partition | 未来增量、回放、补数要围绕哪个窗口或子集管理 |
| backfill | 对缺失或需要重算的历史分区做有边界的补跑 |
| asset check | 对资产健康状态做可执行约束，而不是只看脚本是否退出 0 |

[7] Dagster 官方文档把 asset check 定义为验证数据资产某个属性的机制。Dagster Docs | Asset Checks

## 4. 再看一张图：Week03 里 asset flow 怎么接住 manifest 和 ingest




这张图对应的是当前项目里已经显式存在的设计方向：

- seed_manifests 先变成资产
- 结构化和文档源各自 materialize 到 raw 层
- 后面再继续进入 parse / normalize 和 lakehouse

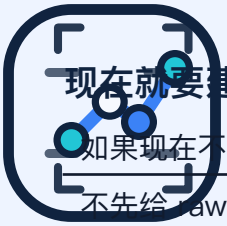
注意这里的关键不是“图画得对不对”，而是你要形成一个稳定判断：

> **manifest 是这次 ingest 的声明；asset 是这次 ingest 产生的持久化对象；job 只是触发 materialization 的方式。**

## 5. 为什么 partition 和 backfill 现在就要讲，而不是等 Week06 再说

这个问题很多学生会问。

答案是：因为 Week03 一旦不先把资产边界立住，Week06 再讲 backfill 时就只能变成“补数脚本技巧”，而不是系统性的恢复能力。



### 现在就要建立的判断

| 如果现在不做什么 | 后面会发生什么 |
| --- | --- |
| 不先给 raw ingest 资产化 | 你只能知道脚本跑过，不知道资产是否可消费 |
| 不先想 partition 维度 | 你后面很难精准 replay / backfill |
| 不先把 metadata 带进 asset materialization | Week04/06 之后的 lineage 和 runbook 会很虚 |
| 不先确定批次 / 时间窗口的边界 | 补数时很容易“多补 / 少补 / 重复补” |

所以这节课真正要做的是：

**把“未来要用到的能力边界”，提前在资产层立起来。**

## 6. 这节课的实践不会让你重写 Dagster，而是让你学会“读 + 记 + 计划”

### Step 1：先确认基础服务已启动

推荐继续使用 repo 的 Docker-first 命令：

```
cp infra/env/.env.example infra/env/.env.local

docker compose --env-file infra/env/.env.local \
  -f infra/docker-compose.yml up -d --build
```

如果服务正常，你应该能访问：

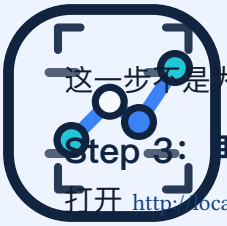
- http://localhost:3000 —— Dagster UI
- http://localhost:9001 —— MinIO Console
- http://localhost:6006 —— Phoenix

这些入口在当前 README 里已经写明。[8]

### Step 2：先跑一次最小 ingest 基线

```
docker compose --profile tools --env-file infra/env/.env.local \
  -f infra/docker-compose.yml run --rm devbox \
  python -m pipelines.ingestion.seed_loader --manifest-dir data/seed_manifests
```

[8] 当前 README 已明确给出 Docker-only / Docker-first 启动方式，以及 Dagster UI http://localhost:3000 的访问方式。OmniSupport Copilot README



这一步不是为了重复课时 1～2，而是为了让 Dagster 资产化观察有“现实基础”。

## Step 3：再看 Dagster 资产图

打开 http://localhost:3000，重点观察：

- 是否能看到 ingestion 相关 assets
- seed_manifests 是否处在入口位置
- raw_doc_assets 与 raw_ticket_events 是否作为下游资产存在
- 当前 job 是否是 ingest_all_job

## Step 4：记录一次最小 materialization smoke report

请把你观察到的内容写进：

```
reports/week03/dagster_materialization_smoke_report.md
```

建议记录：

- 日期 / 时间
- 使用的 manifest 或输入目录
- 看到的 assets
- materialization 是否成功
- 哪些元数据还不够
- 哪些 partition / backfill 还只是未来规划

## Step 5：补一份资产流规划

把下面两个模板填完：

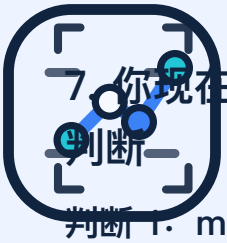
```
docs/blueprints/week03/asset_flow_plan_v1.md
docs/blueprints/week03/partition_backfill_strategy_v1.md
```

前者解决：

- 当前有哪些 ingest 资产
- 它们之间怎么依赖
- 下游将如何消费

后者解决：

- 将来按什么维度 partition
- 哪些源适合按时间窗口 backfill
- 哪些源更适合按 snapshot 维度重建



## 7. 你现在要学会的不是“把 Dagster 用得很花”，而是 3 个工程判断

### 判断 1：manifest 不是资产

manifest 描述这次 ingest 想做什么； 资产描述这次 ingest 最终产生了什么。

### 判断 2：job 成功不等于下游可以安全消费

任务跑通只是开始。 真正重要的是：

- 资产有没有 materialize
- 元数据有没有带齐
- 下游能不能基于它做 retrieval / lakehouse / replay

### 判断 3：backfill 必须围绕 asset + partition 做，不要围绕“脚本名字”做

如果你未来说的是：

- “重跑 `doc_ingest.py`”

这通常还只是任务视角。

如果你说的是：

- “回填 `raw_doc_assets` 在 2026-04-15 这个分区缺失的数据”

那才是资产视角。

## 8. 这节课的产出为什么会直接影响 Week04 和 Week06

### 对 Week04 的影响

Week04 进入 Iceberg Bronze / Silver 后，会更依赖稳定的 asset 边界：

- 哪份 raw 资产进入 Bronze
- 哪些 metadata 要一起保留
- 哪个 batch 对应哪个 snapshot

### 对 Week06 的影响

Week06 要讲回填、重试、Runbook。 如果这节课没先把 asset + partition 的想法立住，Week06 的 replay / backfill 就会很像“高级脚本技巧”，而不是系统恢复能力。

## 9. 本课最重要的 8 个判断

1. 任务成功，不等于数据资产已经可被下游安全消费。
2. Dagster 的价值不只是调度，而是让“应该存在的数据资产”成为一等公民。

3. manifest 声明这次 ingest 想做什么，asset 表达这次 ingest 实际产生了什么。
4. 当前 OmniSupport Copilot 已经有了最小资产化入口，不需要学生自己重造一套编排系统。[9][10]
5. Partition 的价值不在于“切块”，而在于让 incremental processing、replay、backfill 都有明确边界。[11]
6. Backfill 的本质是补跑分区资产，不是整条链路重启。[12]
7. Week03 课时 4 最重要的不是写很多 Dagster 代码，而是学会用资产流思维重新看 ingest。
8. 这节课定义的资产边界，会直接影响 Week04 的 lakehouse 设计和 Week06 的恢复能力。

## 10. 自检清单

学完这一讲后，你应该能勾掉下面这些项：

- ☐ 我能解释任务流和资产流的区别
- ☐ 我知道当前 repo 里哪些文件是 Dagster 入口
- ☐ 我能说清 seed_manifests、raw_doc_assets、raw_ticket_events 分别是什么资产
- ☐ 我能解释 manifest、asset、job 三者的不同职责
- ☐ 我能说清 partition 和 backfill 为什么现在就要提前考虑
- ☐ 我已经完成 asset_flow_plan_v1.md
- ☐ 我已经完成 partition_backfill_strategy_v1.md
- ☐ 我已经记录一次 dagster_materialization_smoke_report.md

## 11. 课后最小行动

在进入课时 5 之前，请你完成下面这组动作：

1. 跑通一次 Docker-first 启动
2. 跑一次 seed_loader
3. 打开 Dagster UI，确认当前 ingestion assets
4. 填完 asset_flow_plan_v1.md
5. 填完 partition_backfill_strategy_v1.md

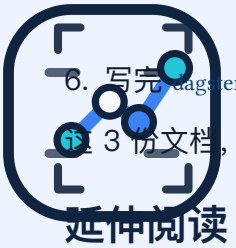
---

[9] 当前 pipelines/ingestion/assets.py 已定义 seed_manifests、raw_doc_assets、raw_ticket_events 和 ingest_all_job，并明确注释“Week03 起接入真实采集器，打通 MinIO 落盘与 PostgreSQL 元数据写入”。assets.py

[10] 当前 pipelines/definitions.py 已把 ingestion、parse_normalize、lakehouse 三类 assets 统一注册到 Dagster Definitions。definitions.py

[11] Dagster 官方文档把 partitions 描述为管理大型数据集、提升性能和实现 incremental processing 的关键技术。Dagster Docs | Partitioning Assets

[12] Dagster 官方文档把 backfill 定义为对不存在或需要更新的分区资产执行补跑。Dagster Docs | Backfilling Data



6. 写完 dagster_materialization_smoke_report.md

这 3 份文档，就是你进入课时 5“故障自愈与补数”之前最关键的准备。

## 延伸阅读

- Dagster / Defining assets
- Dagster / Partitions and backfills
- Dagster / Asset checks
- OmniSupport Copilot / README
- OmniSupport Copilot / pipelines/ingestion/assets.py
- OmniSupport Copilot / pipelines/definitions.py