组织依赖、日志、缓存与失败恢复的样例工作流

完整操作教程 · 参考固定 · 往返恢复验收

用Nextflow DSL2小流程展示输入、输出、任务日志与resume条件，再替换为经过验收的分析阶段。

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

Nextflow与Snakemake都能组织依赖和批量任务，但它们不会自动修正分析方法。本篇只演示Nextflow的DSL2路线：用两个小文本检查输入、独立任务目录、输出发布和失败恢复。先理解这些机制，再将已验证的生信命令放入流程。

准备能判断对错的小输入

前提是已安装兼容的Nextflow及其要求的Java，记录版本并按该安装版文档运行。创建一个全新项目目录，里面准备data/A.txt和data/B.txt，分别写三行与两行非空文本，每行以换行结束。这个计数例子只验证调度链路，不做测序或科学质量判断。

```
nextflow -version
java -version
```

保存下面为main.nf，使用当前仍支持的传统DSL2进程语法。样本ID固定为安全的A和B，输出名因此不会冲突；正式流程改为样本表时要额外检查ID唯一、路径和方法参数，不把任意文本直接拼入shell。

```
nextflow.enable.dsl=2

process COUNT_LINES {
    cpus 1
    errorStrategy 'terminate'
    publishDir 'results', mode: 'copy'

    input:
    tuple val(id), path(input_file)

    output:
    tuple val(id), path("${id}.count.txt")

    script:
    """
    test -s "${input_file}"
    wc -l < "${input_file}" > "${id}.count.txt"
    """
}

workflow {
    inputs = Channel.of(
        tuple('A', file('data/A.txt', checkIfExists: true)),
        tuple('B', file('data/B.txt', checkIfExists: true))
    )
    COUNT_LINES(inputs)
    COUNT_LINES.out.view()
}
```

先观察一次正常运行

在项目根目录执行下例，保存流程总日志、任务trace与报告。每个任务在work下独立执行，声明的path输入被暂存到任务环境。publishDir复制结果是交付出口，后续分析依赖应通过流程输出通道连接，不从发布目录猜测下游文件何时到达。

```
set -eu
nextflow run main.nf -with-trace trace-first.tsv -with-report report-first.html
cat results/A.count.txt results/B.count.txt
```

预期输出分别为3和2，并在trace看到两个任务成功。检查A和B都有结果，不用最后一个任务成功代表全批完成。发布文件可能异步复制，应等工作流正常完成，再做交付验收。正式流程还需记录软件环境、参考版本与输入校验。

材料	作用	恢复时需要
main.nf与配置	任务逻辑、依赖和参数	保留实际版本与修改记录
.nextflow与运行记录	任务缓存元数据	匹配原会话或明确恢复目标
work任务目录	实际输入、脚本及输出	缓存命中仍需相关结果存在
trace及任务日志	状态和失败位置	找到具体失败任务的工作目录

用失败样例理解resume

在独立测试副本中把B输入改为空，验证test使任务非零失败，再按任务对应的work目录查看.command.sh、.command.out、.command.err与退出状态。不要删除整个work来“修复”一个样本，也不要忽略错误把缺结果流程算成功。

```
nextflow run main.nf -resume -with-trace trace-resume.tsv
```

修复输入后重跑时，只有缓存键匹配且work中必要输出仍在的任务才可复用；并非看到results已有文件就自动跳过。修改输入、脚本、容器或相关设置可能使任务重跑，缓存也不是数据备份。记录恢复目标，避免在多个项目中对错会话操作。

把真实分析阶段逐个接入

先为每阶段声明输入、输出和资源，再固定所需包或容器版本。线程参数显式使用分配值，命令失败与输出缺失都应让阶段失败；通用引擎检查不能替代样本、格式和科学质量验收。

最后用正常、缺输入和任务中途失败三类小样例复跑，确认已完成阶段可按规则复用，失败阶段有证据可定位。得到可解释的任务图与验收记录后，再扩大样本量和并发。

Nextflow缓存和resume不是通用断点续算

Nextflow通过任务输入、脚本、容器/环境和缓存元数据判断能否复用。使用-resume前，保留对应工作目录、.nextflow缓存元数据、流程版本和配置；删除work目录或换了项目目录后，缓存可能无法恢复。缓存命中表示任务输入与流程标识符合复用条件，不代表最终结果已由人工验收。

如果修改main.nf、参数、输入或软件环境，先确认哪些任务应重新运行。调试时使用独立测试副本和新的输出目录，保留首次失败目录；不要手动复制一个旧结果到work目录让流程认为任务完成。发布目录是交付副本，不应被当成流程依赖的可靠输入通道。

输入通道和样本级清单

正式生信流程应从受控样本表建立channel，显式传递sample_id、文件路径、对照、参考和参数。启动前检查样本ID唯一、路径存在且可读、成对文件匹配、文件散列可记录。避免把未校验的用户文本直接插入shell脚本；使用Nextflow输入声明和安全参数传递。

小文本计数示例只验证任务调度、输出发布和失败处理，不代表测序分析验证。替换成FASTQ/BAM命令后，先以一个小样本和已知预期结果验证输入与方法，再扩到全批。每个任务的结果应能回溯到样本清单行和环境版本。

资源声明与并发

在process中声明cpus、memory和time并不保证所有执行器都会按相同方式强制限制；还需配置本地/集群executor或profile。记录实际调度器资源请求、容器限制和进程峰值。多个任务并发时，总内存与临时盘需求可能叠加，不能只看单个process配置。

当任务失败或长时间排队时查看trace中的状态、退出码、CPU、内存和运行时，并结合调度器日志。CPU不足、OOM、磁盘满、容器拉取失败和输入错误应分开诊断。只有同类真实样本测得的资源才能用于大队列估算。

错误策略与重试

errorStrategy决定失败后是否终止、重试或忽略。对非零退出码任务，不应设置ignore后让不完整结果进入下游；重试仅适合可恢复的临时故障，并应设置有限次数和清晰日志。输入格式或方法错误重复重试不会修复。

resume前检查失败任务对应的work目录、日志和输入是否仍在。任务失败后修复环境或资源配置，保存新配置版本，再以相同输入重跑并比较。清理缓存前确认其关联的最终结果和复现需求，避免破坏唯一恢复证据。

流程验收和交付

一个批次通过需要所有必需任务状态成功、输出文件齐全、样本集合覆盖完整、关键结果结构检查通过。最后一个任务成功不代表全批成功。保存trace、report、timeline、流程提交、配置、环境锁文件和输入/输出清单。

正式流程还需由研究者审核方法、参考和统计设计。工作流调度正确只说明计算步骤按依赖运行，不证明分析定义正确。分享流程时移除真实路径、样本名和密钥，使用占位或合成数据。

容器和软件环境需要锁定

Nextflow可通过conda或容器为任务提供环境，但流程脚本本身不会自动固定镜像内容。记录容器digest或环境锁文件、Nextflow/Java版本、外部工具版本和参考数据库。不同节点的容器运行时、挂载路径和权限也可能不同，先用小任务验证可读写目录。

缓存复用条件包括任务脚本和输入等元数据；外部数据库若未声明为输入，内容变化可能不触发预期重跑。将数据库版本、参数和软件环境显式纳入流程配置或输入清单，更新时做新的run ID并检查任务状态。

发布目录与流程内部依赖

publishDir用于整理输出，不保证文件复制时序适合作为下游依赖。依赖关系应通过channel和process输出声明，避免下游脚本直接扫描results目录。否则并发时可能读到部分输出或旧文件。

发布模式copy、symlink或move有不同生命周期影响。共享结果目录应避免不同任务使用同名输出覆盖；在样本ID中加入安全唯一标识，并使用配置定义目标路径。归档前校验文件与任务work路径一致。

参数化样本清单与方法设计

从CSV/TSV读取样本时验证表头、路径存在、ID唯一、输入配对和参考版本。将样本级参数放到明确列，不通过拼接不受信任字符串构建任意shell。修改参数后，trace和运行报告应能显示本次每个任务使用的真实值。

流程调度成功只证明程序按依赖运行。正式分析方法、样本分组、过滤和统计仍需单独设计、验证。将计算工程测试和生物学验证分开写在README，防止计数示例通过被误当作生信流程验收。

中断后如何恢复

进程被终止后确认Nextflow主进程和容器子进程已停止，保存最后日志和失败任务work路径。若原缓存、流程版本和输入保持不变，可在文档支持条件下-resume；若目录搬迁或清除了work，恢复能力可能受限。

修复资源配置后保留旧trace，启动新的尝试并比较失败任务状态。不要把完整结果目录重命名或覆盖到流程缓存中。最终用样本表逐一核对必需输出、退出码和发布文件，失败任务留在汇总中。

流程测试从单任务到全批

在方法命令接入前，先验证任务成功、失败、发布和恢复四种状态。成功用预期值核对，失败用刻意缺失输入触发，恢复在测试副本验证。再用一个真实但小型的生信样本检查软件、参考和资源，不能把纯文本计数示例当作生物信息学功能测试。

扩展到多样本前确认ID唯一、输入覆盖全、队列结束状态正确。汇总中列出成功、失败、跳过和待处理样本。流程发布目录中残留的旧文件不能当作新一轮成功输出。

参数配置和报告

将方法参数放到版本控制的配置文件，记录哪些参数是样本级、哪些是全局。参数改变后流程缓存可能使部分任务复用；检查trace中的实际任务和命令。正式运行保存run name、提交、配置、输入清单和报告路径。

资源报告显示的峰值以执行器和采样方式为准。容器内进程内存、宿主机内存和调度器申请额度可能不同，咨询容量时给出这些口径。工作流工具只管理执行，不替研究者决定如何处理生物学重复或统计阈值。

当任务结果依赖外部数据库或API时，将查询日期、数据库版本及缓存策略写入流程输入。若数据会变化但没有版本锁定，Nextflow缓存可能复用与当前资源不匹配的旧结果。

维护者交接时提供如何启动、如何检查trace、失败如何定位、缓存何时可清理和结果如何归档的README。新成员先跑小型测试副本再接入正式数据。

任务完成后按流程输出清单核对发布文件，确认预期样本和结果都在。清理缓存前保存trace和失败恢复所需材料。

若流程同时使用本地conda和容器，明确每个process实际使用哪种环境，不要仅记录全局环境文件。不同任务可能有不同依赖，部署报告应按process列出版本和容器摘要。

对每次run生成唯一run name并保存trace路径，避免同名报告覆盖。

缓存目录保留策略应写入项目说明，明确何时可清理以及恢复前要保留哪些work路径。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

Nextflow：当前进程与DSL语法 · 来源资料

Nextflow：任务缓存及恢复条件 · 来源资料

Nextflow：配置来源与版本记录 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。