为流程固定镜像版本并挂载输入输出目录

完整操作教程 · 参考固定 · 往返恢复验收

固定Docker摘要或Apptainer SIF校验值，用只读输入和独立输出验证一个小任务，并记录宿主条件。

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。

容器能封装一部分软件依赖，但可复现还需要固定输入、参考、参数和运行方式。只保存一个latest镜像名，下次拉取可能已经不同；把结果留在临时容器内部，退出后也可能找不到。先用小任务验证固定镜像与挂载，再搬完整流程。

先确认运行时和镜像来源

本文以Linux上的Docker或Apptainer为可选路线，不要求两者同时安装。使用平台已经允许的运行时与可信镜像，核对CPU架构、工具版本和使用条件。Docker访问权限由管理员配置，不为了跑一条命令随意把成员加入高权限组。

Docker用包含sha256摘要的镜像引用固定内容；Apptainer使用已取得的SIF文件并保存其校验值。镜像内必须确有本文调用的samtools。容器共享宿主内核，GPU还需匹配驱动与运行时配置，并非任何镜像在哪台机器都能用。

检查项	记录	容易误判
镜像身份	完整摘要或SIF校验值、来源	只留可变tag
输入	同一小BAM、参考及参数	同名文件当同一内容
路径	宿主位置与容器内路径对应	只在宿主创建目录却未挂载
输出	独立可写目录与文件所有者	程序成功但结果未落盘

Docker挂载只读输入与新输出

准备INPUT_DIR，里面有已验证的reads.bam；OUTPUT_DIR为不存在的绝对路径，IMAGE为本机已有、固定摘要且含samtools的镜像。下例以本人UID/GID写输出，使用单个排序工作线程及演示内存值，不是正式项目资源推荐。

```
set -eu
: "${INPUT_DIR:?■■■reads.bam■■■■■}"
: "${OUTPUT_DIR:?■■■■■■■■■■■}"
: "${IMAGE:?■■■■■■■■■■■■■■■}"
test -r "$INPUT_DIR/reads.bam" && test ! -e "$OUTPUT_DIR" || exit 2
mkdir "$OUTPUT_DIR"
docker image inspect "$IMAGE" > "$OUTPUT_DIR/image.json"
docker run --rm --pull=never --user "$(id -u):$(id -g)" \
  --mount "type=bind,src=$INPUT_DIR,dst=/input,readonly" \
  --mount "type=bind,src=$OUTPUT_DIR,dst=/output" \
  --entrypoint samtools "$IMAGE" sort -@ 1 -m 256M \
  -o /output/sorted.bam /input/reads.bam \
  > "$OUTPUT_DIR/stdout.log" 2> "$OUTPUT_DIR/stderr.log"
test -s "$OUTPUT_DIR/sorted.bam"
```

镜像原有ENTRYPOINT可能不是所需工具，所以这里显式指定。若镜像不允许这种UID运行或需要额外初始化，按该镜像文档改路线，不能直接加privileged解决。绑定的输入即使只读，其他默认可见目录和运行时权限也仍需按平台管理。

Apptainer使用同样的路径合同

下面为另一轮独立试验，SIF为固定本地文件，APPT_OUT必须是新目录。挂载路径示例不含逗号和冒号；输入仍是同一小BAM。普通执行通常使用调用者权限，能否使用用户挂载取决于平台配置。

```
set -eu
: "${SIF:?■■■■■■■■■■}"
: "${INPUT_DIR:?■■■■■■■■■}"
: "${APPT_OUT:?■■■■■■■■■■}"
test -r "$SIF" && test -r "$INPUT_DIR/reads.bam" && test ! -e "$APPT_OUT" || exit 2
mkdir "$APPT_OUT"
```

```
sha256sum "$SIF" > "$APPT_OUT/image.sha256"
apptainer exec --cleanenv --bind "$INPUT_DIR:/input:ro" \
  --bind "$APPT_OUT:/output:rw" "$SIF" samtools sort \
  -@ 1 -m 256M -o /output/sorted.bam /input/reads.bam \
  > "$APPT_OUT/stdout.log" 2> "$APPT_OUT/stderr.log"
test -s "$APPT_OUT/sorted.bam"
```

输出写权限要同时看宿主目录的所有者、组和挂载设置。Docker指定UID/GID后不一定自动获得宿主的全部附加组；镜像内也可能缺少对应用户名。先用本人新目录验证，必要组权限由平台按范围配置，不把所有项目目录改成人人可写来绕过。

验收容器与结果两层

确认退出成功、镜像与工具版本正确、宿主能读到新BAM且文件权限合理，再用兼容samtools检查、读取记录并核对排序与数量。文件非空只是必要条件，不证明全部数据或科学质量。失败时按镜像缺工具、架构不匹配、挂载路径和写权限分别排查。

把镜像身份、启动命令、挂载、输入校验和输出验收一起保存。更换镜像或宿主条件后重跑相同样例；容器结果能复读，再验证正式输入与资源，不把一次小任务完成当整条流程已完全复现。

先验证UID、权限和输出持久性

容器进程写出的文件最终由宿主文件系统接收，属主和组取决于运行UID/GID、目录权限、ACL和附加组。先以本人身份在新的测试输出目录创建一个小文件，再检查宿主的属主、权限和能否被同课题组成员按计划读取。若Docker用户映射与宿主不匹配，使用平台批准的组策略或Apptainer路线；不要直接chmod 777，也不要让容器以root加特权运行来绕过问题。

镜像可复现还依赖宿主和参考数据

镜像摘要固定了镜像层内容，但没有固定宿主内核、CPU指令集、GPU驱动、挂载数据、环境变量、网络服务和许可证。把这些外部依赖写进运行记录；参考库建议单独版本化并只读挂载，避免每次在容器内临时下载不同版本。涉及网络的工具应记录是否连接外部服务及其版本，若需离线运行，提前在允许的环境准备并校验所有依赖。

检查结果而不止检查退出码

samtools sort示例完成后，可核对输出记录数与输入一致、BAM头中排序顺序符合预期，并用quickcheck检查结构；实际项目还要按参考和分析目的做索引、比对摘要或下游小样本分析。退出码0和非空文件是必要条件但不能证明生物学有效。将同一输入在容器外的可信基线结果对照，说明允许的差异和排序随机性。

保存足以重建运行的清单

每次运行归档镜像摘要或SIF散列、来源、启动参数、挂载路径、用户身份、环境变量白名单、宿主操作系统与运行时版本、输入校验值、日志、退出码和输出验收摘要。镜像更新时用新摘要和独立目录再跑小样例，不覆盖旧结果。保留原镜像引用和数据库发布号，让课题组成员可判断差异来自代码、数据还是宿主环境。

小样本前先做镜像验收

启动真实数据前，可以在输入目录放一个允许共享的微型BAM或测试数据，运行samtools --version、检查镜像内工具位置、测试只读输入和新目录写入。保存docker image inspect或SIF散列、stdout/stderr、退出码和产物校验值。通过后再换代表性项目输入，不要把测试数据与真实样本混在同一输出目录。若镜像无法在无网络环境启动，先核对镜像是否完整拉取或SIF是否完整，而非将数据路径错误当成镜像错误。

避免命令行参数被路径或变量误解析

绑定目录尽量采用绝对路径并确认没有逗号等影响bind语法的字符。镜像、输入和输出变量要先打印到记录文件并检查输入只读挂载，避免相对路径随运行目录变化。命令使用引号保护空格和特殊字符。Docker在远程daemon场景中，绑定源路径可能指daemon主机而非当前客户端；必须确认实际运行主机和存储位置。Apptainer则检查默认绑定和站点配置，不能假定容器内路径与宿主路径一致。

安全与数据管理边界

只读输入挂载可以减少误写，但不等于敏感数据已得到完整隔离。涉及人类遗传数据时遵守项目授权、访问控制、加密和日志政策；使用获批存储与账号，不把数据复制到公开镜像仓库或个人临时目录。镜像构建也不要把凭据、私钥或样本数据写进镜像层。需要共享时只分享许可证允许的代码、合成示例和不含敏感信息的运行说明。

常见故障与下一步判断

容器提示找不到文件时，先在宿主检查绝对路径，再在容器内查看挂载目标，不要重拉镜像。输出权限拒绝先查宿主目录属主、ACL与容器UID。exec format error查CPU架构或镜像构建平台。工具版本不符核对实际摘要和镜像内命令路径。结果文件存在但不可读，则查宿主文件系统、损坏和未完成写入。每次只针对一类证据调整，避免同时换镜像、改挂载和换参数后无法判断修复原因。

网络和依赖应在运行前可见

若镜像入口脚本会自动检查更新、下载参考库或查询外部服务，运行行为就不再完全由固定镜像决定。通过官方文档确认运行参数是否允许离线或禁止隐式拉取；已缓存镜像时设置避免自动拉取的选项，并把参考数据独立挂载。网络不可用时，先识别失败发生在镜像层、参考库层还是分析阶段。需要联网的行为应符合课题组数据管理规则，并记录服务端资源版本。

读者可以按此清单验收迁移

运行容器前核对镜像摘要、工具版本和输入校验；启动时核对只读输入挂载、独立输出路径和实际UID；运行后确认退出码、日志、文件可读、属主合理以及结果内容检查通过。将清单与命令一同放进项目README，让下一位成员能在干净目录重复。出现失败时只改一个变量并保留前后记录。容器带来的可复现性来自这些约束的组合，而不是“代码运行在Docker里”这一事实。

当Docker与Apptainer结果不同

先对照镜像内容或SIF散列、samtools版本、UID/GID、工作目录、环境变量、挂载读写属性和输入散列。检查两边实际执行的命令参数与标准错误，不先假设容器格式改变了算法。若BAM header或记录排序结果不同，确认输出是否被旧文件覆盖、输入是否同一份及程序是否完整退出。容器间对照应基于可读取的结果和确定的差异范围，图像不同或文件散列不同不能单独说明分析结果错误。

将容器参数纳入项目README

README应说明需要的运行时及最低版本、镜像来源和摘要、目标架构、所需挂载、输入输出目录合同、用户权限、环境变量、示例命令、验收方式和已知限制。若Docker与Apptainer任选一种，分别写明二者的命令差异，不要给读者一条无法判断运行时的混合命令。镜像重建时生成新摘要并保留旧版本记录，注明变化的软件包和验证样本。

容器化中不应忽略临时目录

容器内/tmp可能映射到宿主默认临时空间，也可能受容器限制；不同运行时和站点配置并不完全一样。大型排序任务应显式指定临时目录，并确认这个目录在容器内存在、宿主有空间、作业额度覆盖实际内存使用。临时文件写满时区分宿主盘满、tmpfs满、用户配额和容器限制。失败日志与挂载检查一起留存，避免容器退出后临时目录消失而失去诊断证据。

从小样本到正式批量的扩展规则

容器小样本验收后，先用中等规模代表输入复查运行时间、CPU/RAM、输出目录增长和临时文件峰值，再按计划并发逐级增加。容器本身不会自动把磁盘或内存需求变小；并发任务数、线程数和临时目录应一起估算。每次扩展都检查产物和退出码，不因小样本通过就立即启动整批数据。扩展失败时回到上一稳定规模并定位哪项资源或输入改变。

保存可复用的容器验收记录

在项目归档中保留两份材料：简短README供成员运行，完整记录供日后审计。完整记录包含镜像/SIF散列、运行时版本、宿主系统、架构、GPU驱动（如适用）、挂载、UID、输入输出清单、命令、日志、资源高水位和结果核验。下一次服务器迁移时使用同一微型输入重跑，比较工具版本和输出检查结果。能在新环境解释并重建差异，才算容器迁移验收通过。

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

Docker官方：镜像、挂载、用户与入口 · 来源资料

Apptainer：绑定路径与默认挂载 · 来源资料

samtools sort接口 · 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。