是在磁盘上的。即使 JVM 或机器重新启动，File Channel 也不丢失数据，只要磁盘上存储的数据仍然是起作用的和可访问的。机器和 Agent 一旦开始运行，任何存储在 File Channel 中的数据将最终被访问。

Channel 本质上是事务性的。此处的事务不同于数据库事务。每个 Flume 事务代表一批自动写入到 Channel 或从 Channel 删除的事件。无论是当 Source 将事件写入 Channel 时，或 Sink 从 Channel 读取事件时，它必须在事务的范围之内进行操作。

Flume 保证事件至少一次被送到它们的目的地。Flume 只有一次倾力写数据，且不存在任何类型的故障事件只被写一次。但是像网络超时或部分写入存储系统的错误，可能导致事件不止被写一次，因为 Flume 将重试写操作直到它们完全成功。网络超时可能表示写操作的失败，或者只是机器运行缓慢。如果是机器运行缓慢，当 Flume 重试这将导致重复。因此，确保每个事件都有某种形式的唯一标识符通常是一个好主意，如果需要，最终可以用来删除事件数据。

Flume Channel 中的事务

事务性语义是由 Flume 保证的“没有数据丢失”的关键。当每个 Source（或 Sink）写入到 Channel 或读取来自一个 Channel 的数据，会利用 Channel 启动一个事务。对于所有 Flume 自带的 Channel，每个事务是本地线程的。出于这个原因，不同类型 Source 和 Sink 的事务处理略有不同，但基本原理是一样的：每个线程应该运行它自己的事务。对于所有 Pollable Source 和所有 Sink——如前所述，是由运行线程驱动的——每个过程调用只应该启动一个事务，并且如果事务回滚则抛出一个异常，以告知运行线程回退。即使 Source 或 Sink 生成多个 I/O 的新线程，也最好遵循这个协议，如果 `process` 方法失败引发了众多事务之一的事务，以避免歧义。

当 Source 写事件到 Channel 时，事务由 Channel 处理器所处理，所以 Source 无须处理自己的事务。只有当事件被成功写入 Channel，Channel 处理器才提交事务；否则，Channel 处理器将回滚该事务并关闭它。因为每个 Source 可以写入到多个 Channel，所以 Channel 处理器可以为 Source 依次写事件并提交它们。因此，数据被写出并被提交到一些 Channel 而不是其他 Channel 是可能的。在这种情况下，Flume 不能回滚已经被提交的事务，但能确保数据成功写出到所有 Channel，Flume 将重试写数据到所有 Channel，包括之前成功写入的那些；这可能会导致重复。

24

对于终端 Sink，只有当数据被安全写出到存储系统，事务才应该被提交。一旦数据在最终目的地是安全的，那么可以提交事务，且 Channel 可以删除该事务中的事件。如果写操作失败，Flume 必须回滚事务，以确保事件不会丢失。所有 Flume 自带的 Sink 都以这种方式工作，并在事务提交前确保数据已经在 HDFS、HBase、Solr、Elastic Search 等。