很重要的是要理解 Sink 组中所有 Sink 是不会在同时被激活的；任何时候只有它们中的一个用来发送数据。因此，Sink 组不应该用来更快地清除 Channel，在这种情况下，多个 Sink 应该只是设置为自己操作自己，而没有 Sink 组，且它们应该配置从相同的 Channel 进行读取。

每个 Sink 组在活跃列表中被声明为一个组件，就像 Source、Sink 和 Channel 一样，使用 `sinkgroups` 关键字。每个 Sink 组是一个需要命名的组件，因为每个 Agent 可以有多个 Sink 组。Sink 组以下面的方式进行定义：

```
agent.sinkgroups = sg1 sg2
```

该配置展示了两个定义的 Sink 组：`sg1` 和 `sg2`。然后每个 Sink 组配置了一些 Sink 作为 <158> 组的一部分。在 Sink 活跃集合中的 Sink 列表，优先于指定为 Sink 组一部分的 Sink 列表。因此，所有作为 Sink 组的一部分的 Sink，也必须分别定义在 Sink 的活跃集中，以用来激活它们。下面展示用 Sink 集合配置 `sg1` 和 `sg2`:

```
agent.sinks = s1 s2 s3 s4
agent.sinkgroups.sg1.sinks = s1 s2
agent.sinkgroups.sg2.sinks = s3 s4
```

Sink 组中的每个 Sink 必须单独进行配置。这包括以下这些配置，Sink 从哪个 Channel 读取，它写数据到哪些主机或集群等。如果 Sink 组表示一组 RPC Sink，意味着传达到下一层，每个连接的主机必须配置一个 Sink，用来发送数据给它。据推测，它们都从同一 Channel 读取，因为这是层和层之间的通信。在理想情况下，如果 Sink 组中建立了几个 Sink，所有的 Sink 将从相同的 Channel 读取，这将有助于在当前层以合理的速度清除数据，确保将要被发送到多台机器的数据，以一种支持负载均衡和故障转移的方式进行发送。

清除 Channel 的速度比单个 Sink 组这样做的速度要快很重要，但也要求当建立的 Agent 发送数据到多个主机时，可以添加多个 Sink 组，每个组中的 Sink 有相似的配置。例如，前一个示例中的 `sg1` 和 `sg2` 分别有 Sink `s1`、`s2` 和 `s3`、`s4`。`s1` 和 `s3` 可以有相同的配置（从同一 Channel 将数据推送到相同的主机和端口），同时 `s2` 和 `s4` 可以有相似的配置。这将确保更多的对于目的地 Agent 的连接是开放每一个 Agent 的，同时如果需要，也允许数据被推到不止一个的 Agent。这使得 Channel 被清除得更快，同时确保负载均衡和故障转移自动发生。

目前为止，我们已经讨论了 Sink 组如何创建流去做负载均衡和故障转移，但是我们还没有讨论如何实际地告知 Sink 组它们应该进行负载均衡或故障转移。这是通过使用 Sink 处理器完成的。Sink 处理器是任何时候决定哪个 Sink 是活跃的组件。