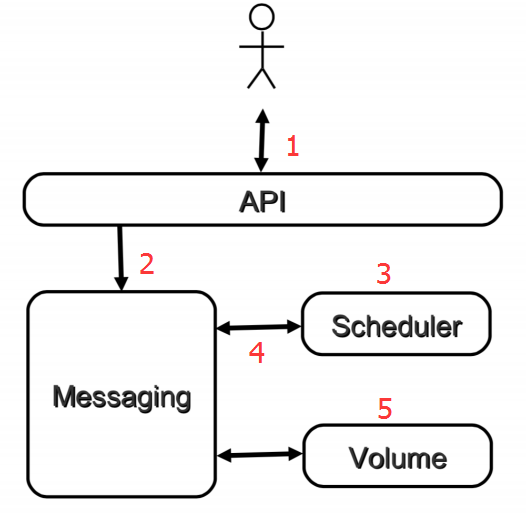


上一节介绍了 Cinder 的架构，这节讨论 Cinder 个组件如何协同工作及其设计思想。

**从 volume 创建流程看 cinder-\* 子服务如何协同工作**

对于 Cinder 学习来说，Volume 创建是一个非常好的场景，涉及各个 cinder-\* 子服务，下面是流程图。  


1. 客户（可以是 OpenStack 最终用户，也可以是其他程序）向 API（cinder-api）发送请求：“帮我创建一个 volume”
2. API 对请求做一些必要处理后，向 Messaging（RabbitMQ）发送了一条消息：“让 Scheduler 创建一个 volume”
3. Scheduler（cinder-scheduler）从 Messaging 获取到 API 发给它的消息，然后执行调度算法，从若干计存储点中选出节点 A
4. Scheduler 向 Messaging 发送了一条消息：“让存储节点 A 创建这个 volume”
5. 存储节点 A 的 Volume（cinder-volume）从 Messaging 中获取到 Scheduler 发给它的消息，然后通过 driver 在 volume provider 上创建 volume。

上面是创建虚机最核心的几个步骤，当然省略了很多细节，我们会在后面的章节详细讨论。

**Cinder 的设计思想**

Cinder 延续了 Nova 的以及其他组件的设计思想。

**API 前端服务**

cinder-api 作为 Cinder 组件对外的唯一窗口，向客户暴露 Cinder 能够提供的功能，当客户需要执行 volume 相关的操作，能且只能向 cinder-api 发送 REST 请求。这里的客户包括终端用户、命令行和 OpenStack 其他组件。

设计 API 前端服务的好处在于：

1. 对外提供统一接口，隐藏实现细节
2. API 提供 REST 标准调用服务，便于与第三方系统集成
3. 可以通过运行多个 API 服务实例轻松实现 API 的高可用，比如运行多个 cinder-api 进程

**Scheduler 调度服务**

Cinder 可以有多个存储节点，当需要创建 volume 时，cinder-scheduler 会根据存储节点的属性和资源使用情况选择一个最合适的节点来创建 volume。

调度服务就好比是一个开发团队中的项目经理，当接到新的开发任务时，项目经理会根据任务的难度，每个团队成员目前的工作负荷和技能水平，将任务分配给最合适的开发人员。

**Worker 工作服务**

调度服务只管分配任务，真正执行任务的是 Worker 工作服务。

在 Cinder 中，这个 Worker 就是 cinder-volume 了。这种 Scheduler 和 Worker 之间职能上的划分使得 OpenStack 非常容易扩展：

当存储资源不够时可以增加存储节点（增加 Worker）。

当客户的请求量太大调度不过来时，可以增加 Scheduler。

**Driver 框架**

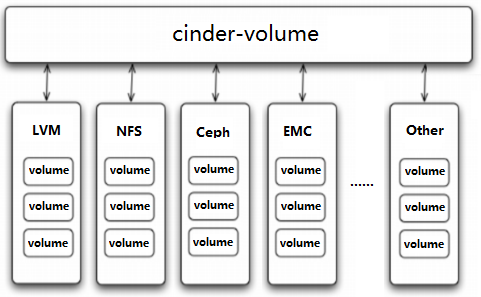
OpenStack 作为开放的 Infrastracture as a Service 云操作系统，支持业界各种优秀的技术，这些技术可能是开源免费的，也可能是商业收费的。

这种开放的架构使得 OpenStack 保持技术上的先进性，具有很强的竞争力，同时又不会造成厂商锁定（Lock-in）。

那 OpenStack 的这种开放性体现在哪里呢？一个重要的方面就是采用基于 Driver 的框架。

以 Cinder 为例，存储节点支持多种 volume provider，包括 LVM, NFS, Ceph, GlusterFS，以及 EMC, IBM 等商业存储系统。

cinder-volume 为这些 volume provider 定义了统一的 driver 接口，volume provider 只需要实现这些接口，就可以 driver 的形式即插即用到 OpenStack 中。下面是 cinder driver 的架构示意图：



在 cinder-volume 的配置文件 /etc/cinder/cinder.conf 中 volume\_driver 配置项设置该存储节点使用哪种 volume provider 的 driver，下面的示例表示使用的是 LVM。

http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160612-1465733848406013133.jpg

下一节我们将详细讨论 Cinder 的每一个组件。