51 | 谈谈Kubernetes开源社区和未来走向

51 | 谈谈Kubernetes开源社区和未来走向

张磊 2018-12-19

09:28

讲述：张磊 大小：8.68M

你好，我是张磊。今天我和你分享的主题是：谈谈 Kubernetes 开源社区和未来走向。

在前面的文章中，我已经为你详细讲解了容器与 Kubernetes 项目的所有核心技术点。在今天这最后一篇文章里，我就跟你谈一谈 Kubernetes 开源社区以及 CNCF 相关的一些话题。

我们知道 Kubernetes 这个项目是托管在 CNCF 基金会下面的。但是，我在专栏最前面讲解容器与 Kubernetes 的发展历史的时候就已经提到过，CNCF 跟 Kubernetes 的关系，并不是传统意义上的基金会与托管项目的关系，CNCF 实际上扮演的，是 Kubernetes 项目的 Marketing 的角色。

这就好比，本来 Kubernetes 项目应该是由 Google 公司一家维护、运营和推广的。但是为了表示中立，并且吸引更多的贡献者加入，Kubernetes 项目从一开始就选择了由基金会托管的模式。而这里的关键在于，这个基金会本身，就是 Kubernetes 背后的“大佬们”一手创建出来的，然后以中立的方式，对 Kubernetes 项目进行运营和 Marketing。

通过这种方式，Kubernetes 项目既避免了因为 Google 公司在开源社区里的“不良作风”和非中立角色被竞争对手口诛笔伐，又可以站在开源基金会的制高点上团结社区里所有跟容器相关的力量。而随后 CNCF 基金会的迅速发展和壮大，也印证了这个思路其实是非常正确和有先见之明的。

不过，在 Kubernetes 和 Prometheus 这两个 CNCF 的一号和二号项目相继毕业之后，现在 CNCF 社区的更多职能，就是扮演一个传统的开源基金会的角色，吸纳会员，帮助项目孵化和运转。

只不过，由于 Kubernetes 项目的巨大成功，CNCF 在云计算领域已经取得了极高的声誉和认可度，也填补了以往 Linux 基金会在这一领域的空白。所以说，你可以认为现在的 CNCF，就是云计算领域里的 Apache ，而它的作用跟当年大数据领域里 Apache 基金会的作用是一样的。

不过，需要指出的是，**对于开源项目和开源社区的运作来说，第三方基金会从来就不是一个必要条件。**事实上，这个世界上绝大多数成功的开源项目和社区，都来自于一个聪明的想法或者一帮杰出的黑客。在这些项目的发展过程中，一个独立的、第三方基金会的作用，更多是在该项目发展到一定程度后主动进行商业运作的一部分。开源项目与基金会间的这一层关系，希望你不要本末倒置了。

另外，需要指出的是，CNCF 基金会仅仅负责成员项目的 Marketing， 而绝不会、也没有能力直接影响具体项目的发展历程。无论是任何一家成员公司或者是 CNCF 的 TOC（Technical Oversight Committee，技术监督委员会），都没有对 Kubernetes 项目“指手画脚”的权利，除非这位 TOC 本人就是 Kubernetes 项目里的关键人物。

所以说，真正能够影响 Kubernetes 项目发展的，当然还是 Kubernetes 社区本身。可能你会好奇，**Kubernetes 社区本身的运作方式，又是怎样的呢?**

通常情况下，一个基金会下面托管的项目，都需要遵循基金会本身的管理机制，比如统一的 CI 系统、Code Review 流程、管理方式等等。

但是，在我们这个社区的实际情况，是先有的 Kubernetes，然后才有的 CNCF，并且 CNCF 基金会还是 Kubernetes “一手带大”的。所以，在项目治理这个事情上，Kubernetes 项目早就自成体系，并且发展的非常完善了。而基金会里的其他项目一般各自为阵，CNCF 不会对项目本身的治理方法提出过多的要求。

而说到 **Kubernetes 项目的治理方式，其实还是比较贴近 Google 风格的，即：重视代码，重视社区的民主性。**

首先，Kubernetes 项目是一个没有“Maintainer”的项目。这一点非常有意思，Kubernetes 项目里曾经短时间内存在过 Maintainer 这个角色，但是很快就被废弃了。取而代之的，则是 approver+reviewer 机制。这里具体的原理，是在 Kubernetes 的每一个目录下，你都可以添加一个 OWNERS 文件，然后在文件里写入这样的字段：

```
approvers:
- caesarxuchao
reviewers:
- lavalamp
labels:
- sig/api-machinery
- area/apiserver
```

复制代码

比如，上面这个例子里，approver 的 GitHub ID 就是 caesarxuchao （Xu Chao），reviewer 就是 lavalamp。这就意味着，任何人提交的 Pull Request（PR，代码修改请求），只要修改了这个目录下的文件，那么就必须要经过 lavalamp 的 Code Review，然后再经过 caesarxuchao 的 Approve 才可以被合并。当然，在这个文件里，caesarxuchao 的权力是最大的，它可以既做 Code Review，也做最后的 Approve。但， lavalamp 是不能进行 Approve 的。

当然，无论是 Code Review 通过，还是 Approve，这些维护者只需要在 PR 下面 Comment /lgtm 和 /approve，Kubernetes 项目的机器人（k8s-ci-robot）就会自动给该 PR 加上 lgtm 和 approve 标签，然后进入 Kubernetes 项目 CI 系统的合并队列，最后被合并。此外，如果你要对这个项目加标签，或者把它 Assign 给其他人，也都可以通过 Comment 的方式来进行。

可以看到，在上述整个过程中，代码维护者不需要对 Kubernetes 项目拥有写权限，就可以完成代码审核、合并等所有流程。这当然得益于 Kubernetes 社区完善的机器人机制，这也是 GitHub 最吸引人的特性之一。

顺便说一句，很多人问我，**GitHub 比 GitLab 或者其他代码托管平台强在哪里？**实际上， GitHub 庞大的 API 和插件生态，才是这个产品最具吸引力的地方。

当然，当你想要将你的想法以代码的形式提交给 Kubernetes 项目时，除非你的改动是 bugfix 或者很简单的改动，否则，你直接提交一个 PR 上去，是大概率不会被 Approve 的。**这里的流程，**一定要按照我下面的讲解来进行：

1. 在 Kubernetes 主库里创建 Issue，详细地描述你希望解决的问题、方案，以及开发计划。而如果社区里已经有相关的 Issue 存在，那你就必须要在这里把它们引用过来。而如果社区里已经存在相同的 Issue 了，你就需要确认一下，是不是应该直接转到原有 issue 上进行讨论。
2. 给 Issue 加上与它相关的 SIG 的标签。比如，你可以直接 Comment /sig node，那么这个 Issue 就会被加上 sig-node 的标签，这样 SIG-Node 的成员就会特别留意这个 Issue。
3. 收集社区对这个 Issue 的信息，回复 Comment，与 SIG 成员达成一致。必要的时候，你还需要参加 SIG 的周会，更好地阐述你的想法和计划。
4. 在与 SIG 的大多数成员达成一致后，你就可以开始进行详细的设计了。
5. 如果设计比较复杂的话，你还需要在 Kubernetes 的设计提议目录（在 Kubernetes Community 库里）下提交一个 PR，把你的设计文档加进去。这时候，所有关心这个设计的社区成员，都会来对你的设计进行讨论。不过最后，在整个 Kubernetes 社区只有很少一部分成员才有权限来 Review 和 Approve 你的设计文档。他们当然也被定义在了这个目录下面的 OWNERS 文件里，如下所示：

   reviewers:

   - brendandburns

   - dchen1107

   - jbeda

   - lavalamp

   - smarterclayton

   - thockin

   - wojtek-t

   - bgrant0607

   approvers:

   - brendandburns

   - dchen1107

   - jbeda

   - lavalamp

   - smarterclayton

   - thockin

   - wojtek-t

   - bgrant0607

   labels:

   - kind/design

□复制代码
这几位成员，就可以称为社区里的“大佬”了。不过我在这里要提醒你的是，“大佬”并不一定代表水平高，所以你还是要擦亮眼睛。此外，Kubernetes 项目的几位创始成员，被称作 Elders（元老），分别是 jbeda、bgrant0607、brendandburns、dchen1107 和 thockin。你可以查看一下这个列表与上述“大佬”名单有什么不同。

1. 上述 Design Proposal 被合并后，你就可以开始按照设计文档的内容编写代码了。这个流程，才是正常大家所熟知的编写代码、提交 PR、通过 CI 测试、进行 Code Review，然后等待合并的流程。
2. 如果你的 feature 是需要要在 Kubernetes 的正式 Release 里发布上线的，那么你还需要在Kubernetes Enhancements这个库里面提交一个 KEP（即 Kubernetes Enhancement Proposal）。这个 KEP 的主要内容，是详细地描述你的编码计划、测试计划、发布计划，以及向后兼容计划等软件工程相关的信息，供全社区进行监督和指导。

以上内容，就是 Kubernetes 社区运作的主要方式了。

总结

在本篇文章里，我为你详细讲述了 CNCF 和 Kubernetes 社区的关系，以及 Kubernetes 社区的运作方式，希望能够帮助你更好地理解这个社区的特点和它的先进之处。

除此之外，你可能还听说过 Kubernetes 社区里有一个叫作 Kubernetes Steering Committee 的组织。这个组织，其实也是属于Kubernetes Community 库的一部分。这个组织成员的主要职能，是对 Kubernetes 项目治理的流程进行约束和规范，但通常并不会直接干涉 Kubernetes 具体的设计和代码实现。

其实，到目前为止，Kubernetes 社区最大的一个优点，就是把“搞政治”的人和“搞技术”的人分得比较清楚。相信你也不难理解，这两种角色在一个活跃的开源社区里其实都是需要的，但是，如果这两部分人发生了大量的重合，那对于一个开源社区来说，恐怕就是个灾难了。

思考题

你能说出 Kubernetes 社区同 OpenStack 社区相比的不同点吗？你觉得这两个社区各有哪些优缺点呢？