Ansible FAQ

Jun 11th, 2013 9:18 pm

本文是从原Ansible官网的FAQ页面翻译而来，网站改版后该页面已无法访问，但可以从Github历史提交中获得。翻译这篇原始FAQ文档是因为它陈述了Ansible这款工具诞生的原因，设计思路和特性，以及与Puppet、Fabric等同类软件的比较，可以让我们对Ansible有一个整体的了解，所以值得使用者一读。

目录

为什么命名为“Ansible”？

我最喜爱的书籍之一是奥森·斯科特·卡特的《安德的游戏》。在这本书中，“Ansible”是一种能够跨越时空的即时通讯工具。强烈推荐这本书！

Ansible受到了谁的启发？

我在Red Hat任职期间主要开发Cobbler，很快我和几个同事就发现在部署工具（Cobbler）和配置管理工具（cfengine、Puppet等）之间有一个空缺，即如何更高效地执行临时性的任务。虽然当时有一些并行调用SSH脚本的方案，但并没有形成统一的API。所以我们（Adrian Likins、Seth Vidal、我）就开发了一个SSH分布式脚本框架——Func。

我一直想在Func的基础上开发一个配置管理工具，但因为忙于Cobbler和其他项目的开发，一直没有动手。在此期间，John Eckersberg开发了名为Taboot的自动化部署工具，它基于Func，采用YAML描述，和目前Ansible中的Playbooks很像。

近期我在一家新公司尝试引入Func，但遇到一些SSL和DNS方面的问题，所以想要开发一个更为简单的工具，吸收Func中优秀的理念，并与我在Puppet Labs的工作经验相结合。我希望这一工具能够易于学习，且不需要进行任何安装步骤。使用它不需要引入一整套新的理论，像Puppet和Chef那样，从而降低被某些运维团队排挤的可能。

我也曾参与过一些大型网站的应用部署，发觉现有的配置管理工具都太过复杂了，超过了这些公司的需求。程序发布的过程很繁复，需要一个简单的工具来帮助开发和运维人员。我不想教授他们Puppet或Chef，而且他们也不愿学习这些工具。

于是我便思考，应用程序的部署就应该那么复杂吗？答案是否定的。

我是否能开发一款工具，让运维人员能够在15分钟内学会使用，并用自己熟悉的语言来扩展它？这就是Ansible的由来。运维人员对自己的服务器设施最清楚，Ansible深知这一点，并将同类工具中最核心的功能提取出来，供我们使用。

Ansible不仅易于学习和扩展，它更是集配置管理、应用部署、临时任务等功能于一身。它非常强大，甚至前所未有。

我很想知道你对Ansible的看法，到邮件列表里发表一下意见吧。

与同类软件比较

Func？

Ansible默认使用SSH，而非SSL和守护进程，无需在远程服务器上安装任何软件。你可以使用任何语言编写插件，只要它能够返回JSON格式即可。Ansible的API深受Func的影响，但它和Func相较提供了配置管理和多节点统一化部署（Playbooks）等功能。

Puppet？

首先我要强调的是，如果没有Puppet，就不会有Ansible。Puppet从cfengine中吸收了配置管理的概念，并更合理地加以实现。但是，我依旧认为它可以再简单一些。

Ansible的playbook是一套完整的配置管理系统。和Puppet不同，playbook在编写时就隐含了执行顺序（和Chef类似），但同时也提供了事件机制（和Puppet类似），可以说是结合了两者的优点。

Ansible没有中心节点的概念，从而避免了惊群效应。它一开始就是为多节点部署设计的，这点Puppet很难做到，因为它是一种“拉取”的架构。Ansible以“推送”为基础，从而能够定义执行顺序，同时只操作一部分服务器，无需关注它们的依赖关系。又因为Ansible可以用任何语言进行扩展，因此并不是只有专业的程序员才能为其开发插件。

Ansible中资源的概念深受Puppet的启发，甚至“state”这一关键字直接来自Puppet的“ensure”一词。和Puppet不同的是，Ansbile可以用任何语言进行扩展，甚至是Bash，只需返回JSON格式的输出即可。你不需要懂得Ruby。

和Puppet不同，Ansible若在配置某台服务器时发生错误，它会立即终止这台服务器的配置过程。它提倡的是“提前崩溃”，修正错误，而非最大化应用。这一点在我们需要配置包含依赖关系的服务器架构时尤为重要。

Ansible的学习曲线非常平滑，你不需要掌握编程技能，更不需要学习新的语言。Ansible内置的功能应该能够满足超过80%的用户需求，而且它不会遇到扩展性方面的瓶颈（因为没有中心节点）。

如果系统中安装了factor，Ansible同样支持从中获取系统信息。Ansible使用jinja2作为模板语言，类似于Puppet使用erb文件作为模板。Ansible可以使用自己的信息收集工具，因此factor并不是必需的。

Chef？

Ansible与Chef的区别和Puppet类似。Chef的配置非常困难，而且需要你掌握Ruby语言。也因为如此，Chef在Rails使用者中很流行。

Ansible是按照编写顺序来执行任务的，而不是显示地定义依赖关系，这点和Chef相似。但

Ansible更进一步，它支持事件触发，比如修改了Apache的配置文件，Apache就会被重启。

和Chef不同的是，Ansible的playbook不是一门编程语言，而是一种可以存储的数据结构。这就意味着你的运维工作不是一项开发型的任务，测试起来也相对简单。

无论你有怎样的语言背景，都可以使用Ansible。Chef和Puppet有超过六万行的代码，而Ansible则是一段小巧简单的程序。我相信这一点会使得Ansible更加健壮和可靠，并汇聚一批活跃的社区贡献者——因为任何人都可以提交补丁或是模块。

Ansible同样支持从ohai中获取系统信息，当然这同样不是必需的。

Capistrano/Fabric？

这些工具并不适合用作服务器配置工具，它们主要用于应用程序的部署。

而Ansible则提供了完整的配置管理，以及在扩展性方面提供了一些高级特性。

Ansible playbook的语法简介只占一个HTML页面，有着非常平缓的学习曲线。由于Ansible使用了“推送”的设计，因此对系统管理员（不仅仅是开发者）同样适用，并能用它处理各种临时性的任务。

其它问题

Ansible的安全性如何？

Ansible没有守护进程，主要使用OpenSSH进行通信，这是一款已被反复检验并广泛使用的软件。其它工具都会在远程服务器上以root用户运行守护进程，因此相较于这些工具，Ansible会更为安全，且无需担心网络方面的问题。

如果你的中心节点遭到入侵（或是被恶意员工登录），只要你是使用SSH-agent、或是经过加密的密码，那你的密钥仍然是被锁定的，别人无法操控你的节点。而对于Chef、Puppet等工具来说，一旦配置文件遭到篡改，那危及的将是整个网络。

此外，由于Ansible没有守护进程，可以节省下一部分内存和计算资源，这对需要最大化性能的用户来说也是一个优点。

Ansible如何扩展？

无论是在单次执行模式还是playbook模式下，Ansible都可以并行执行任务，这要感谢Python提供的多进程处理模块。

你可以自行决定要一次性配置5台还是50台服务器，这取决于服务器的计算能力，以及你想要多快完成任务。

由于没有守护进程，所以平时不会占用任何资源，而且你不用担心一次性有太多节点一起从控制节点上获取信息。

对于SSH，Ansible默认使用paramiko库，当然也能使用原始的openssh。Ansible可以利用SSH的ControlMaster特性来重用网络连接。

当要维护上万个节点时，单个Ansible playbook可能不太合理，这时你就能使用Ansible的“拉取”模式。这种模式下需要配合git和cron，可以扩展到任意多台服务器。“拉取”模式可以使用本地连接，或是SSH。关于这个模式的详细说明可以在帮助文档的“Advanced Playbooks”一节查阅。即使在“拉取”模式下，你同样能够享受到Ansible的种种便利。

如果你想进一步探讨扩展性，可以加入到邮件列表中。

是否支持SSH以外的协议？

目前Ansible支持SSH和本地连接，但它的接口实际上是非常易于扩展的，因此你可以编写补丁来使Ansible运行于消息系统或XMPP协议之上。

如果你有任何建议，可以加入到邮件列表中一起探讨。Ansible中对于连接的管理都已单独抽象出来，有很强的可扩性。

Ansible的适用场景有哪些？

最适场景？使用playbook进行多节点云主机部署；从一个初始的操作系统开始部署应用，或是配置一个现有的系统。

Ansible同样适用于执行临时性的任务，能够用于各类 Unix-like 系统，因为它使用的就是系统本身自带的工具，无需安装额外软件。

你还可以用Ansible来编写各类脚本，用于收集信息、执行各种任务，对QA、运维等团队均适用。

Posted by Ji ZHANG Jun 11th, 2013 9:18 pm translation