Ansible/Puppet/saltstack比较

运维大湿兄 关注
0.143 2019.06.04 12:31:10 字数 1,184 阅读 9,280

Puppet、Chef、Ansible、Salt对比

工具	语言	架构	协议
Puppet	Ruby	C/S	HTTP
Chef	Ruby	C/S	HTTP
Ansible	Python	无Client	SSH
Saltstack	Python	C/S(可无Client)	SSH/ZMQ/RAET

Ansible或者Puppet

存在即是合理，起码是存在3年以上的；没有最好的，只有合适的，你说白菜和青菜哪个最好?

一般来说，有两种配置管理：

1. 推模式
2. 拉模式

两种模式有不同的擅长点，有不同的使用场景。

拉模式 (puppet)

这种模式主张去中心化的设计思路，典型代表 puppet。一般实现多为在每个节点上部署 agent，定时获取该节点的配置信息，根据配置信息配置本节点。如果一次配置失败了，那么下次继续尝试，直到地老天荒。这个节点完全不管其他节点的执行情况，一心只顾做好自己的事情。

所以它比较适合这种场景：
对配置何时生效不敏感，不关心的。你知道它总是会生效的，可能是下一分钟，也可能是下个小时，但是对你没什么影响。
节点和节点之间不需要协作的。比如这种 场景就不合适： A 先升级，然后 B 在升级。
即使某一次拉取信息失败了，下一次还能补上，所以比较适合跨地域的大规模部署。

推模式 (ansible)

推模式有一个中心节点，用于将最新的配置信息推到各个节点上，典型代表 ansible。很明显，推模式的瓶颈就在中心节点，如果同一时间有 10000 个节点需要更新配置，那么中心节点如何稳定的工作就比较有学问。

它比较适合这种场景：
对配置生效的时间敏感，十分关心。必须让他们即可生效，如果不生效，立马要采取行动让他们生效。
配置生效的顺序十分关心和敏感。比如需要这10个节点一起生效，或者按照依次生效。

SaltStack或Ansible

SaltStack与Ansible都是Python写的而且较新，网上评论也很好。

> 1、是否需要每台机器部署agent（客户端）
> 很多选用ansible的朋友，都是因为agentless这个原因，觉得要维护agent很麻烦。
> 而一些使用saltstack比较顺的朋友，觉得这个问题无所谓，agent出问题的概率有，但不高。

其实ansible也支持agent的方式，即所谓的“pull”的模式，就是通过一个客户端去拉取要执行的任务。

2、大规模并发的能力
这方面的对比已经比较多了，因为实现机制的差异，也导致saltstack在这方面是占优的。不过对于几十台-200台规模的兄弟来讲，ansible的性能也可接受。
注：前期调研的大多数都是中下企业，服务器规模一般不超过200**台，所以对这个问题不算太看重。如果一次操作的机器过千台，可能还是用**saltstack**效率更高一些。**
ansible的执行架构已经有所优化，采用基于MQ的agent机制，已支持比较大规模（1000-10000台）的服务器的批量自动化运维。这样，在这种存在大规模运维的需求的客户这里，也可以应用丰富的ansible的Playbook了。

3、二次开发扩展的能力
ansible和saltstack都是基于python的，而python在运维开发这个圈子里接受度还是非常高的，二次开发的人员相对也好招。
这也是这两个工具相对于puppet和chef更容易被接受度原因，这两个曾经的主流工具都是基于ruby，而现在ruby的活跃度越来越低了，要招人也不容易。
ansible和saltstack都具备很好的二次开发扩展能力，可以写YAML编排。

4、开源社区的对接
在github上，ansbile有18300多颗星，salt有6700多颗星。
直接按关键字搜索，ansible的相关项目也更多一些。
这些指标虽然不能直接说明什么，但很多技术人员会关注自己所使用的技术的活跃度。一般来说，越活跃的开源项目，得到的关注会更高些，功能完善和问题解决的效率也会更高。