[集中化运维管理——Puppet](http://www.programmer.com.cn/13525/" \o "Permanent Link to 集中化运维管理——Puppet管理之路)

大数据时代高伸缩性、容错性的特点给运维提出了更高的要求。系统管理不再是疲于安装操作系统、对系统参数进行逐一配置与优化、打补丁、安装软件、配置软件、添加某个服务的时代。为了提高效率、避免重复劳动、减少错误、积累知识，系统管理员都已开始做一些局部的自动化工作。但这些还远不够， 为了满足运维需求，需要更彻底地应用自动化运维工具。

**运维工作流程**

常见的运维工作流程包括：安装系统→优化系统与配置→安装软件→配置软件→添加监控→检查。后续可能还会有添加服务→配置变更→打补丁修复漏洞等。是不是觉得很繁琐？尤其当你负责大量设备，无法凭借一己之力完成时，便需要一些工具来帮忙。

运维工作面临的各种不确定性更让人头疼。在10台机器上变更应用还是很简单的事，但如果上升到千台、万台就会变得非常复杂。重复性的劳动还会让人觉得疲惫和乏味，久而久之可能还会产生厌倦工作的情绪。使用Puppet则能将这些问题迎刃而解。

**自己实现自动化**

为了提升工作效率，减少出错机率。很多公司都在逐步采用自动化来实现上述工作。有的公司选择自己开发一套工具，因为可以按照需求任意定制，但这真的有必要吗？我们来看看这样做的缺点。

1. 从头造轮子：构建脚本工作的挑战与繁琐。

2. 程序的可维护性无法保障（语言）。

3. 支撑不同的平台。

4. 系统重装后的考虑。

统筹与规划整个系统需要花费很长时间，伴随着人员的流动，技能水平不一，还会带来新的问题。而且单独开发的系统不可能只是支撑一个平台——跨平台开发则意味着更多不确定性。

**自动化配置工具Puppet的功能**



**剖析Puppet**

在使用任何软件前我们都需要了解其工作原理，否则会给后续使用带来诸多不便。Puppet采用了非常简单的C/S架构，所有数据的交互都通过SSL进行，以保证安全。它的工作流程如图1所示。

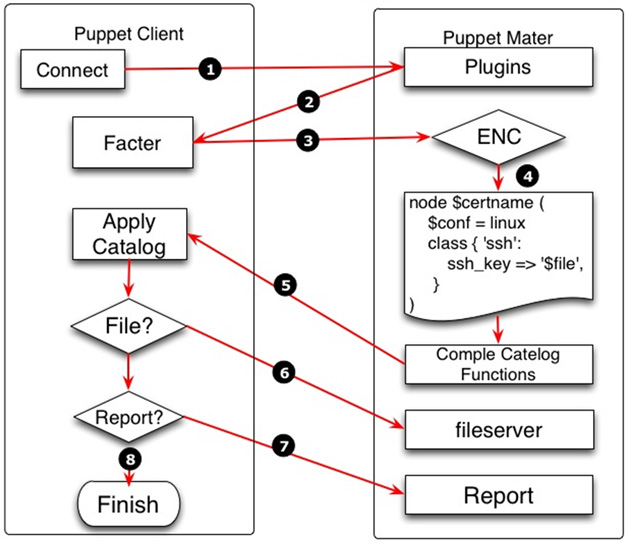


图1 Puppet工作流程

1. 客户端Puppetd向Master发起认证请求，或使用带签名的证书。

2. Master告诉Client你是合法的。

3. 客户端Puppetd调用Facter，Facter探测出主机的一些变量，例如主机名、内存大小、IP地址等。Puppetd将这些信息通过SSL连接发送到服务器端。

4. 服务器端的Puppet Master检测客户端的主机名，然后找到manifest对应的node配置，并对该部分内容进行解析。Facter送过来的信息可以作为变量处 理，node牵涉到的代码才解析，其他没牵涉的代码不解析。解析分为几个阶段，首先是语法检查，如果语法错误就报错；如果语法没错，就继续解析，解析的结 果生成一个中间的“伪代码”（catelog），然后把伪代码发给客户端。

5. 客户端接收到“伪代码”，并且执行。

6. 客户端在执行时判断有没有File文件，如果有，则向fileserver发起请求。

7. 客户端判断有没有配置Report，如果已配置，则把执行结果发送给服务器。

8. 服务器端把客户端的执行结果写入日志，并发送给报告系统。

**当服务器超过千台**

当你的服务器越来越多时，你可能会发现Puppet执行效率开始下降，服务器已无法满足你的需求。下面介绍几种方案来解决这类问题。

**LoadBlancer**

这是通过非常简单的扩容Master方案，来提升Master计算“伪代码”的能力。通常这种架构能支撑1000台左右的服务器。当然，这也依赖于你的系统是否够“复杂”。

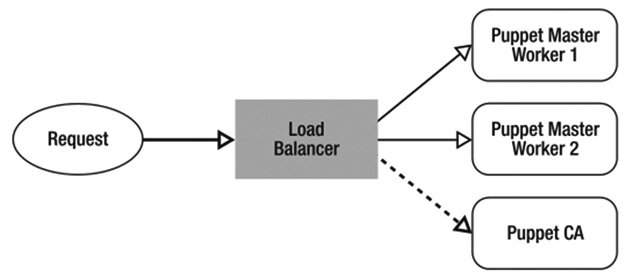


图2 LoadBlancer方案

这种架构有两种常用的实现方式：Apache+Passenger，

以及Nginx+Mongrel。本文将以后者为例简单介绍其工作方式。

1. Puppet Master运行多个进程：

Puppet Master+Mongrel，port 18140

Puppet Master+Mongrel，port 18141

Puppet Master+Mongrel，port 18142

Puppet Master+Mongrel，port 18143

2. Nginx通过Upstream的方式实现对Puppet Master的负载均衡。Nginx监听port 8140将除文件下发之外的请求，代理转发给上面的4个Puppet Master实例之一，Nginx会对客户端证书进行验证，但需要配置CA签发的证书才允许请求，我们也可以再配置8141提供证书签发。

3. 如果使用了fileserver，Nginx也可以直接处理。

**Puppet认证负载均衡**

仅有多个Master是否够用？一台机器还是有风险，为此我们需要有容错，将Master分布在不同机器上，而CA认证也是非常重点的一部分，我们可以采用以下架构做一个热备。如图3所示。

这 架构还可以进行扩展。我们再次回顾Puppet工作原理；Puppet客户端跟Nginx之间是HTTPS连接，Nginx跟各个Mongrel之间用的 是HTTP连接。对客户端证书的验证由Nginx负责，而Nginx只需要有CA的公钥就可以做验证工作。这样做的好处是多台管理机之间不需要同步客户端 的证书等设置，只需要有CA的公钥，公钥复制就能使用。但这样有一个缺点：不太方便删除客户端证书。不过可以采用一个主管理机的方式，其他管理机从这台管 理机上实时同步证书。

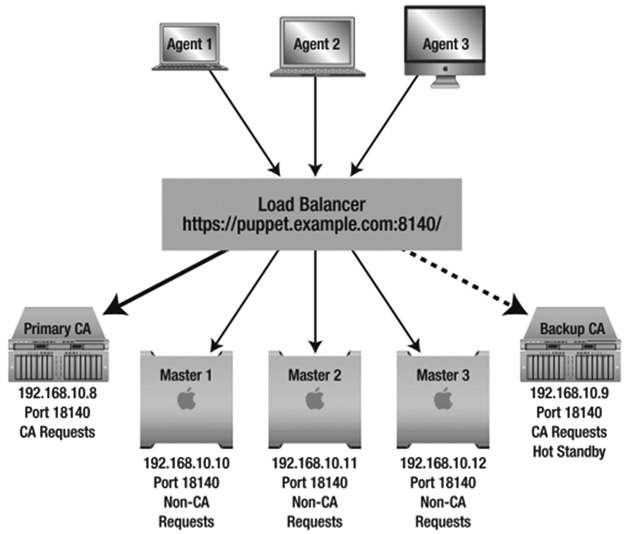


图3 Puppet认证负载均衡方案

Puppet管理机集群思路如下：

1. 将CA配置同步到每台机器上，包括公私钥；

2. 用CA给每台管理机器签发一个证书；

3. 每台管理机都配成LoadBalancer的方式，8140提供配置管理，8141提供证书签发；

4. 各管理机之间可以使用Keeplived实现高可用性及故障切换，包括HA等，架构可随意扩展；

5. 每台管理机配置分Production和Development两种，简单地通过Git中发布到管理机上；

6. 测试时只修改Development部分，在个别客户端指定用它，成功后再推到Production下；

7. 配置一台主CA管理机，解决删除认证的问题。

**合理规划**

所有的一切事后挽救方案都不如使用前合理规划，你需要非常清楚当前业务的状态、运维的现状。了解你需要解决什么问题，然后将它分解，再逐步击破。

推荐采用Git管理Puppet；

规范HostName，采用DNS管理；

fileServer独立，将不经常变化的放在fileserver里，经常变化的放在模板中；

沟通自定义OS。

可能很多人不太明白为什么要自定义OS，它最大的优点就是可以在系统初始化安装时帮你做一些Puppet所需要的软件包，通过采购设备时拿到的SN号，在 WebUI系统里注册这台机器的信息，机器在启动后即可完成所有的配置。如果你的WebUI做得更好，可以调用监控系统的API完成监控。这样是不是很完 美？