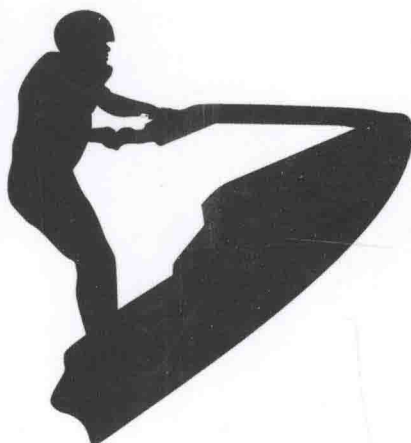


资深运维专家撰写，InfoQ和51CTO等技术社区，以及童剑、田逸、南非蜘蛛、程辉等业界资深专家联袂推荐

详细讲解Puppet的功能和使用方法，深入剖析Puppet的工作原理，系统总结Puppet的使用技巧，包含大量来自一线的战案例和最佳实践



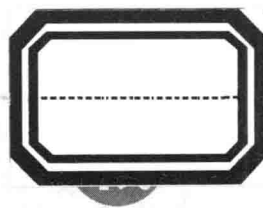
刘宇 著

Puppet in Action

Puppet实战



机械工业出版社
China Machine Press



Puppet in Action

Puppet实战

刘宇 著



机械工业出版社
China Machine Press

图书在版编目(CIP)数据

Puppet 实战 / 刘宇著. —北京: 机械工业出版社, 2013.12

ISBN 978-7-111-44518-0

I. P… II. 刘… III. 程序开发工具 IV. TP311.52

中国版本图书馆 CIP 数据核字 (2013) 第 251103 号

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问 北京市展达律师事务所

Puppet 领域的经典之作, 资深运维专家多年一线经验结晶, 51CTO 技术社区强烈推荐, 新浪研发中心平台架构部高级总监童剑、资深运维专家田逸、中国最大开源社区 ChinaUnix 创始人之一南非蜘蛛、OpenStack 基金会董事程辉等业界资深专家联袂推荐。本书详细讲解了 Puppet 的功能和使用方法, 深入剖析了 Puppet 的工作原理, 系统总结了 Puppet 的使用技巧, 包含大量来自一线实战案例和最佳实践。

全书一共 20 章, 共分为四部分: 准备篇 (第 1~4 章) 介绍了 Puppet 用途、组织结构、工作原理、核心配置文件、各种平台下的安装与配置, 以及它的运行环境, 是使用 Puppet 前必须做好的准备工作; 基础篇 (第 5~10 章) 详细讲解了 Puppet 的理论知识 and 功能使用, Puppet 的语法与命令、资源、模块、类、模板、节点管理, 以及 Facter、数组、函数、变量和标签; 实战篇 (第 11~13 章) 通过几个经典案例, 包括大规模 Nginx 集群的部署方案、分布式监控系统部署方案、OpenStack 快速部署方案等, 使读者能快速将 Puppet 运用到实践中; 进阶篇 (第 14~20 章) 综合讲解了 Puppet 的扩展模式、版本控制、报告系统、控制台、扩展工具及 MCollective, 让读者了解一个完整的 Puppet 生产流程。



机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑: 姜 影

北京市荣盛彩色印刷有限公司印刷

2014 年 1 月第 1 版第 1 次印刷

186mm × 240mm · 23.5 印张

标准书号: ISBN 978-7-111-44518-0

定 价: 69.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzsj@hzbook.com

前言

为什么要写这本书

子曰：“工欲善其事，必先利其器。”作为系统管理员，最应该具备的一种技能就是利用各种优秀开源软件快速完成自己的工作，让自己变得更轻松。但并不是所有的软件都适合你，你需要根据自己的环境及需求进行选择。

Puppet 就是系统管理员的一把利器。几年前我开始学习 Puppet 的时候，不少朋友都问我：为什么选择 Puppet？它有什么优势？当时 pssh、CFEngine 等都已经很成熟，并且也能帮助系统管理员有效地解决繁重的工作。为什么还要选择 Puppet？“需求决定产品”。Luke Kanies (Puppet 作者，以下简称 Luke) 曾经全面参与了重写 CFEngine 的解析器和开发 ISConf3 的工作，但是他还是觉得现有的工具已经无法满足他的工作需求，需要自己创造一个全新的工具，才能彻底使他的工作更加高效、便捷。这便是 Puppet 诞生的背景和原因。

Puppet 注重设计简洁、先框架后应用两个中心思想。这也造就了 Puppet 今日的成功，它的“三板斧”——资源、类、模板，可以轻松地帮我们完成复杂的业务逻辑关系管理。同时，Puppet 并不具备执行功能，因此在某些程度上存在短板，（比如 `exec` 是为解决特定系统管理员整脚的执行命令需求而开发的，但 Luke 一再强调不建议使用。）

或许更多的系统管理员抱怨 Puppet 没有命令执行功能。Luke 早就考虑到了这一点，并提前收购了 MCollective，采用消息型总线的中间件来实现命令执行、系统管理、Puppet 客户端管理等，以弥补了 Puppet 在这方面的不足，可见 Puppetlabs 非常有远见。伴随着 Puppet 各子功能及扩展工具的遍地开花，Puppet 的商业化及各种开源社区的支持，Puppet 可谓是蒸蒸日上。

在新兴的同类工具中，我的另一个关注点是 SaltStack，它天生具备命令执行和配置管理两大核心功能，相比 Puppet 来说有一定优势，但 SaltStack 的成长还需要我们耐心等待。Puppet 打败 1993 年“出生”的老牌系统配置管理工具 CFEngine 就花费了近 7 年时间。而 Puppet 与 Salt Stack 真正鹿死谁手，我们拭目以待。相信以后的市场竞争会越来越激烈。

为了让更多的系统管理员了解并深入学习 Puppet，可以利用 Puppet 的集成方案解决系统管理复杂而繁重的任务，而不是盲目地寻找问题，我觉得有必要结合自己在学习 Puppet 过程中走过的弯路将工作中的经验和总结以实战形式呈现给大家。我也希望更多的人能加入开源社区，拥抱开源，拥抱变化，通过学习 Puppet，体会到与开源爱好者交流的乐趣，而不是为了工具选型而犹豫甚至争斗。这也是我写作本书的初衷。

本书技术深入而阅读简单，涉及系统管理员所需的很多方面的基础知识，同时通过穿插各种实例及代码详解以便使读者能够快速掌握 Puppet，并迅速将其运用到工作环境当中。通过这样一本以系统管理员为核心的书，希望能给读者带来的不只是技术能力提升，更多的是开源与奉献精神。也希望更多的系统管理员通过多阅读、多交流，建立起享受分享的技术氛围。

读者对象

根据本书内容定位，适合阅读本书的读者有：

- ❑ Puppet 用户和爱好者
- ❑ UNIX 与 Linux 系统管理员
- ❑ 运维工程师

如何阅读本书

本书分为四大部分：

第一部分为准备篇（第 1 ~ 4 章），简单地介绍了 Puppet 的发展历程和相关理论，帮助读者了解一些基础背景知识，并快速搭建测试环境。

第二部分为基础篇（第 5 ~ 10 章），着重讲解 Puppet 的基础理论知识，包括语法、资源、类、模板、模块、节点、**Fact**er、数组、函数、变量。结合不同实例让读者感觉理论知识不再那么枯燥。

第三部分为实战篇（第 11 ~ 13 章），从实战角度进行讲解，结合流行监控系统 Nagios 和 Zabbix，包括最为热门的云计算 OpenStack 的部署，使读者能快速掌握 Puppet 并运用到实践中。

第四部分为进阶篇（第 14 ~ 20 章），通过对 Puppet 扩展模式、版本控制、报告系统、控制台、扩展工具及 MCollective 的综合讲解，让读者了解一个完整的 Puppet 生产流程。

其中第三部分以实战来讲解 Puppet 应用，相比于前两部分更加复杂。如果你是一名经验丰富的资深用户，能够理解 Puppet 的相关基础知识和工作原理，那么你可以直接阅读这部分内容。但是如果你是一名初学者，请一定从第 1 章的基础理论知识开始学习。

勘误和支持

由于作者的水平有限，加之编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为此，我特意在博客（<http://liuyu.github.io>）的基础上开辟了在线支持与应急方案的站点 <http://puppet.bubbyroom.com>。你可以将书中的错误以及遇到的有关 Puppet 的任何问题，通过访问该站点的 Q&A 页面提交给我。书中的全部源文件除可以从华章网站^①下载外，还可以从这个站点下载。同时我也会将相应的功能更新在这个站点上及时发布出来。如果你有更多的宝贵意见，也欢迎发送邮件至邮箱 liuyu105@gmail.com (`sed 's/#/@/g'`)，标题请注明《Puppet 实战》，期待能够得到你们的真挚反馈。同样你也可以关注我为本书建立的微信公共平台（[puppetchina](#)），我会在此公共平台定期发布 Puppet 相关信息。

致谢

首先要感谢伟大的 Puppet 作者 Luke Kanies，他开发了一款影响我整个人生的软件。

特别感谢刘长元、王广胜、吴问志、刘继伟、王哲，感谢你们对本书提出的宝贵修改意见。

感谢为本书撰写的童剑先生、窦哲先生、田逸先生、程辉先生，感谢你们在繁忙的工作中抽出时间，阅读了本书的样稿并写下推荐和评论。

感谢机械工业出版社华章公司的编辑杨福川，是你的引导才促使我完成这本书。特别感谢姜影编辑，在这一年多的时间中始终帮我校对并支持我的写作，你的鼓励和帮助引导我顺利完成全部书稿。

最后感谢我的笨笨，一如既往地支持我写作，给予我无尽的支持与灵感，并时时刻刻为我灌输着信心和力量！

谨以此书献给我即将出生的 Baby，以及众多热爱 Puppet 的朋友们！

刘宇（守住每一天）

于中国北京

① 参见华章网站 www.hzbook.com。——编辑注

目 录

前 言

第一部分 准备篇

第 1 章 认识 Puppet.....2

1.1 Puppet 的起源与发展现状.....2

1.1.1 什么是 Puppet.....2

1.1.2 Puppet 起源与发展.....2

1.1.3 版本语言特征.....3

1.1.4 命令差异.....4

1.1.5 Puppet 3.0 新特性.....4

1.2 为什么要使用 Puppet.....5

1.2.1 都有谁在使用 Puppet.....5

1.2.2 常见集中化管理工具对比.....5

1.2.3 推荐 Puppet 的理由.....6

1.3 Puppet 的作用和特色.....6

1.3.1 为什么要有自己的语言.....6

1.3.2 为什么是 Ruby.....6

1.3.3 管理任何机器.....6

1.4 Puppet 组织结构.....7

1.5 Puppet 工作原理.....7

1.5.1 Puppet 基本结构.....7

1.5.2 Puppet 是如何工作的.....8

1.5.3 Puppet 数据流.....8

1.5.4 文件结合.....9

1.5.5 详细交互过程.....9

1.5.6 安全与认证.....10

1.6 Puppet 核心配置文件详解.....11

1.6.1 主配置文件 puppet.conf.....11

1.6.2 主机配置文件 site.pp.....13

1.6.3 认证与安全配置文件.....14

1.6.4 客户端自动认证配置.....16

1.6.5 报告系统配置.....16

1.6.6 文件系统配置文件.....16

1.7 本章小结.....17

第 2 章 Puppet 安装与配置.....18

2.1 Puppet 对各系统平台的支持.....18

2.2 Puppet 对 Ruby 的支持.....19

2.3 Puppet 的安装步骤.....19

2.4 在 Linux 下安装.....20

2.4.1 包管理器方式安装.....21

2.4.2 从源代码进行安装.....23

2.4.3 从 Git 版本库进行安装.....24

2.4.4 通过 Gems 进行安装.....25

2.5 在 Mac OS X 下安装.....25

2.5.1 通过二进制发布包进行

安装.....25

2.5.2 从 Git 版本库进行安装	26	5.1.3 类	52
2.5.3 从 Ports 仓库进行安装	27	5.1.4 模块	52
2.6 配置 Puppet	28	5.1.5 节点	52
2.7 在 Windows 下安装与使用	30	5.2 主机、模块和类的命名	53
2.7.1 包管理器方式安装	31	5.2.1 主机的命名	53
2.7.2 在 Windows 下使用 Puppet	33	5.2.2 模块的命名	54
2.7.3 Puppet 在 Windows 下的 功能	35	5.2.3 类的命名	54
2.8 如何升级	35	5.3 资源、变量、参数和标签的命名	54
2.9 本章小结	36	5.3.1 资源的命名	54
		5.3.2 变量的命名	55
		5.3.3 参数的命名	55
		5.3.4 标签的命名	55
第 3 章 创建你的第一个 Puppet 配置	37	5.4 Puppet 语法风格	55
3.1 配置一个测试节点	37	5.4.1 间距、缩进和空白字符	56
3.2 检测你的配置文件	38	5.4.2 注释	56
3.3 客户端运行配置	39	5.4.3 变量的引用	56
3.4 查看运行结果	40	5.4.4 资源	57
3.5 增加 httpd 模块	40	5.4.5 条件语句	60
3.6 本章小结	42	5.4.6 类	61
第 4 章 Puppet 运行环境	43	5.5 检查命令的用法	64
4.1 服务器端配置	43	5.5.1 语法检查	64
4.2 客户端配置	44	5.5.2 代码调试	65
4.3 如何运用环境配置	45	5.6 Puppet 命令详解	66
4.4 本章小结	48	5.6.1 Puppet 常用命令	68
		5.6.2 帮助命令详解	73
		5.6.3 模块和不常用命令	75
		5.7 本章小结	76
第二部分 基础篇			
第 5 章 Puppet 语法与命令详解	50	第 6 章 Puppet 资源详解	77
5.1 Puppet 的命名规范	51	6.1 什么是资源	78
5.1.1 资源	51	6.1.1 图解核心资源	81
5.1.2 属性	51	6.1.2 什么是 manifests	81

6.1.3 资源的依赖	82	6.8.4 资源引用	120
6.2 虚拟资源	85	6.8.5 数字	120
6.2.1 虚拟资源的定义	85	6.8.6 哈希类型	121
6.2.2 虚拟资源的用法	86	6.8.7 正则表达式	121
6.3 常用资源的用法	87	6.8.8 数组	122
6.3.1 用户资源	88	6.9 标签	123
6.3.2 用户组资源	90	6.10 stage 运行阶段	123
6.3.3 软件安装	91	6.11 本章小结	124
6.3.4 文件管理	94		
6.3.5 服务管理	97	第 7 章 Puppet 模块、类、模板	125
6.3.6 定时脚本	99	7.1 图解模块结构	125
6.3.7 执行命令	101	7.2 模块管理	126
6.3.8 调试与输出	103	7.2.1 实例：创建一个模块	127
6.4 Puppet 作用域与变量	104	7.2.2 模块布局	129
6.4.1 作用域	104	7.3 类管理	130
6.4.2 变量	108	7.3.1 类的定义	131
6.5 条件语句	110	7.3.2 类的继承	131
6.5.1 if 语句	110	7.3.3 参数化类	132
6.5.2 case 语句	112	7.4 模板管理	135
6.5.3 selector 选择器	112	7.4.1 定义与声明	135
6.6 表达式	113	7.4.2 ERB 模板语法	136
6.6.1 什么是表达式	113	7.5 融合	139
6.6.2 运用位置	114	7.6 从 Puppet Forge 获取模块	141
6.6.3 操作顺序	114	7.7 从 Example42 获取模块	142
6.6.4 比较运算符	114	7.8 本章小结	143
6.6.5 布尔运算符	115		
6.6.6 算术运算符	116	第 8 章 节点管理	144
6.7 函数	116	8.1 什么是节点	144
6.8 数据类型	118	8.2 主机名命名规范	145
6.8.1 布尔类型	118	8.3 节点继承	146
6.8.2 未定义	119	8.3.1 节点继承关系	146
6.8.3 字符串	119	8.3.2 继承变量覆盖	147

8.3.3 默认类与默认节点	147	10.2.3 tag 函数	170
8.3.4 节点继承的判断	148	10.2.4 tagged 函数	171
8.4 节点管理方法	149	10.2.5 识别标签	171
8.4.1 每个主机名独立	149	10.3 指定标签运行特定配置	172
8.4.2 采用正则匹配	150	10.3.1 在命令行中指定特定 标签	172
8.4.3 使用外部节点分类器	150	10.3.2 在配置文件中指定	174
8.4.4 利用 WEB-UI 管理	153	10.3.3 在 Node 节点配置中 指定	174
8.5 如何选择合适的管理方式	154	10.4 标签的更多用法	175
8.6 本章小结	154	10.4.1 在收集资源中使用	175
第 9 章 认识 Facter	155	10.4.2 实例化资源	175
9.1 什么是 Facter	155	10.4.3 创建资源集合	176
9.2 Facter 的作用与特点	156	10.5 本章小结	176
9.3 Facter 的常用变量	157		
9.3.1 操作系统名	158		
9.3.2 操作系统相关	159		
9.3.3 主机名	159		
9.3.4 IP 地址	160		
9.3.5 内存管理	160		
9.3.6 系统状态信息	161		
9.3.7 版本管理	161		
9.4 如何自定义 fact	162		
9.5 案例一：条件语句	164		
9.6 案例二：匹配不同硬件配置	165		
9.7 本章小结	167		
第 10 章 小标签大用途	168		
10.1 Puppet 标签的定义	168		
10.2 Puppet 标签的说明	168		
10.2.1 自动分配标签	169		
10.2.2 tag 元参数	169		
		第三部分 实战篇	
		第 11 章 大规模 Nginx 集群部署 方案	178
		11.1 应用场景	178
		11.2 场景需求分析	178
		11.2.1 日常变更分析	178
		11.2.2 网络及架构分析	179
		11.2.3 软件安装分析	180
		11.2.4 软件配置分析	180
		11.2.5 节点管理分析	181
		11.3 合理规划	181
		11.3.1 系统安装	181
		11.3.2 系统初始化	182
		11.3.3 部署规划	182
		11.3.4 关注点	183

11.4 实施步骤.....183	13.2.1 环境准备.....214
11.4.1 前期准备：创建软件 仓库.....183	13.2.2 安装软件及 Puppet 模块.....216
11.4.2 Puppet 配置文件管理.....185	13.2.3 部署 controller.....218
11.4.3 初始化操作系统.....187	13.2.4 部署 compute.....220
11.4.4 编写 nginx 模块.....187	13.2.5 验证 OpenStack 部署.....221
11.4.5 采用 Forge 的 nginx 模块.....192	13.3 本章小结.....222
11.5 本章小结.....194	
第 12 章 分布式监控系统部署方案195	第四部分 进阶篇
12.1 利用 Puppet 部署 Zabbix.....196	第 14 章 Puppet 版本控制224
12.1.1 Zabbix 简介.....196	14.1 Puppet 版本控制方法.....224
12.1.2 Zabbix 架构.....197	14.1.1 为什么要使用版本控制.....224
12.1.3 利用 Puppet 部署 Zabbix.....198	14.1.2 版本控制的架构与原理.....225
12.1.4 Zabbix 自定义监控.....201	14.1.3 Git 与 SVN 的区别.....226
12.2 利用 Puppet 部署 Nagios.....202	14.1.4 为什么采用 Git.....226
12.2.1 Nagios 简介.....202	14.2 使用 Git 实现 Puppet.....226
12.2.2 Nagios 架构.....203	14.2.1 安装与配置 Git.....227
12.2.3 Nagios 服务端安装.....204	14.2.2 将 Puppet 加入 Git.....228
12.2.4 Nagios 模块应用.....206	14.2.3 使用 Rake 自动更新副本.....229
12.2.5 创建 Nagios 客户端监控.....208	14.2.4 使用 hook 实现自动 语法检查.....231
12.3 本章小结.....210	14.3 本章小结.....232
第 13 章 OpenStack 快速部署 方案211	第 15 章 Puppet 架构扩展与分 布式233
13.1 OpenStack 简介.....211	15.1 Puppet 瓶颈分析.....233
13.1.1 什么是 OpenStack.....211	15.1.1 单台 Puppet Master 瓶颈.....233
13.1.2 OpenStack 的组件、服务 及逻辑架构.....212	15.1.2 认证的瓶颈.....234
13.1.3 OpenStack 版本说明.....213	15.1.3 文件的瓶颈.....234
13.2 部署 OpenStack.....214	15.1.4 网路的瓶颈.....234
	15.2 架构扩展之单台 Puppet Master.....234
	15.2.1 Nginx+Mongrel 模式.....235

15.2.2	Apache+Passenger 模式	238	16.7	File 资源的缺点	276
15.2.3	Nginx+Passenger 模式	242	16.8	本章小结	276
15.3	架构扩展之多台 Puppet Master	244	第 17 章	强大的报告系统	277
15.3.1	配置前的准备	248	17.1	report 介绍	277
15.3.2	Puppet CA 认证服务器 部署	250	17.2	Puppet 信息记录方式	278
15.3.3	Puppet LB 负载均衡器 部署	251	17.3	tagmail 发送邮件报告	279
15.3.4	Puppet Master 服务器 部署	252	17.4	rrdgraph 图形化报告	280
15.3.5	Puppet 客户端配置	254	17.5	自定义报告处理器	282
15.3.6	验证架构	254	17.6	本章小结	284
15.4	架构扩展之利用 Git 构建 分布式的 Puppet	254	第 18 章	必须了解的控制台	285
15.4.1	实现原理	255	18.1	Puppet DashBoard	285
15.4.2	安装与部署	256	18.1.1	简介	285
15.5	本章小结	259	18.1.2	DashBoard 安装	285
第 16 章	File 资源管理优化	260	18.1.3	配置 DashBoard	287
16.1	深入理解 File 资源	260	18.1.4	集成 DashBoard	292
16.2	操作实践	262	18.2	Foreman	297
16.3	File 资源配置方法	269	18.2.1	Foreman 简介	297
16.3.1	模块文件目录配置	269	18.2.2	安装 Foreman	298
16.3.2	统一文件目录配置	270	18.2.3	配置 Foreman	303
16.3.3	content 属性	271	18.2.4	使用 Foreman 管理 Puppet	311
16.4	File 资源的优化	271	18.2.5	从 Foreman 显示报告	313
16.4.1	配置 Nginx 代理	272	18.2.6	Foreman 其他功能	314
16.4.2	选择 File 资源还是 ERB	272	18.3	本章小结	314
16.4.3	大文件下发方法	272	第 19 章	Puppet 扩展工具	315
16.5	从 filebucket 检索文件	272	19.1	生成 HTML 文档	315
16.6	备份与恢复文件	275	19.1.1	利用 puppet doc 生成 HTML	316

19.1.2 puppet doc 的其他使用方法.....	318	20.1.4 MCollective 给 Puppet 带来的改变.....	338
19.2 生成依赖关系图.....	319	20.2 消息中间件.....	339
19.2.1 什么是关系图.....	319	20.2.1 Stomp.....	339
19.2.2 配置方法.....	319	20.2.2 ActiveMQ.....	339
19.2.3 关系图说明.....	322	20.2.3 RabbitMQ.....	339
19.3 PuppetDB.....	322	20.3 标准化部署 MCollective.....	340
19.3.1 PuppetDB 功能与特性.....	322	20.3.1 体系结构与配置.....	340
19.3.2 安装 PuppetDB.....	324	20.3.2 安全模型.....	340
19.3.3 PuppetDB 配置文件详解.....	326	20.3.3 未来扩展.....	341
19.3.4 配置与使用 PuppetDB.....	329	20.4 部署 MCollective 步骤.....	341
19.3.5 PuppetDB 瓶颈.....	332	20.4.1 创建和收集证书.....	342
19.4 Hiera.....	335	20.4.2 部署和配置中间件.....	344
19.4.1 Hiera 的特点.....	335	20.4.3 MCollective 安装与配置.....	348
19.4.2 Hiera 的使用.....	335	20.5 如何使用 MCollective.....	355
19.5 本章小结.....	335	20.5.1 mco 基本命令的用法.....	355
第 20 章 MCollective 结合.....	336	20.5.2 执行 RPC 请求.....	356
20.1 MCollective 简介.....	336	20.5.3 过滤器的使用.....	358
20.1.1 什么是 MCollective.....	336	20.6 MCollective 使用 Shell Commands.....	359
20.1.2 MCollective 角色互换.....	337	20.7 MCollective 控制 Puppet.....	360
20.1.3 MCollective 的特点.....	338	20.8 本章小结.....	362

第一部分

准备篇

Puppet 可谓是配置管理工具中的旗舰产品，有别于 chef 等其他配置管理工具，它可以使系统管理员的工作变得更轻松，从而提升工作效率，并且只需要维护简单的关系就能掌握一切。使用 Puppet，我们可以从系统安装、配置管理、系统监控来实现运维体系的一体化、流水线化、产品化。通过简单的后台系统我们就可以完成所有的动作，并且能查看机器的进度与当前状态。是不是觉得这项工作非常让人振奋呢？那就让我们一起探究 Puppet 的奥秘吧！

本篇主要是为了更好地学习和使用 Puppet 而做的准备。我们通过第 1 章来了解 Puppet，看看它的发展历程，重点掌握它的工作原理；通过第 2 章对安装、配置 Puppet 的讲解进一步加深对 Puppet 工作原理的理解；通过第 3 章的实例，创建一个属于自己的配置，进而掌握基本的配置方法；最后通过第 4 章学习 Puppet 的几种工作环境，我们以后能够更好地利用 Puppet 实现不同环境的配置管理。

第 1 章

认识 Puppet

本章首先介绍什么是 Puppet，为什么它具有这么优秀的特征，以及它的发展历程。随后将 Puppet 和当前主流开源配置管理工具进行对比，阐述为什么要使用 Puppet。了解 Puppet 的组织结构并掌握 Puppet 的工作原理，这是本章的重点，希望读者多次阅读这部分内容，以便完全理解。Puppet 最新版本为 3.0（截至 2013 年 9 月），不同版本之间新的特性及是否可以混合使用，以及配置文件的相关知识，这些都是本章将要介绍的内容。

1.1 Puppet 的起源与发展现状

1.1.1 什么是 Puppet

官方的定义是这样的：Puppet 是一个开源的新一代的集中化配置管理工具，它由自己所声明的语言表达系统配置，通过客户端与服务端之间的连接，维护着关系库。Puppet 的设计目标是让 Puppet 成为一个由富有表现力的语言支撑的足够强大的库。这样只需要编写短短的几行代码的自动化应用程序即可实现设计目标。同时 Puppet 是开放的，允许添加任何新的功能。

通常这样定义：Puppet 是一个跨平台的集中化配置管理系统，它使用自有的描述语言，可管理配置文件、用户、Cron、软件包，系统服务等，Puppet 把这些统称为“资源”。Puppet 的设计目标就是简化对这些资源的管理以及妥善处理资源之间的依赖关系。

Puppet 是基于 Ruby 语言（<http://www.ruby-lang.org/>）并使用 Apache 协议授权的开源软件（在 Puppet 2.7.0 与 Facter 1.6.0 之前是基于 GPLv2 协议授权的），它既能以客户端 - 服务端（C/S）的方式运行，也能独立运行。客户端默认每 30 分钟会与服务端确认一次更新，以确保配置的一致性。

1.1.2 Puppet 起源与发展

Puppet 主要由 Luke Kanies 和他的公司 Puppet Labs 开发，于 2005 年正式面世。Luke

Kanies 于 1997 年就涉足 UNIX 和系统管理, 由于平常需要进行大量的开发工作, 在试用当时的配置管理软件不满意后, 他便想到自己开发一套工具来管理资源。他认为这套工具实现的关键是如何使每个资源成为一个合集, 每个资源又有自己的行为, 并且产生相应的行为动作。在 Luke Kanies 的努力下终于有了今天这款出色的 Puppet 产品。当然也感谢 Ruby 语言。

Puppet 于 2005 年面世, 发行版本也由 0.2 发展到了 3.0。Puppet Labs 的目标不只是把它当做配置工具, 还把它当做“拯救世界”的工具。特别是在近几年, 其发展势头迅猛。截至 2011 年 11 月 Puppet Labs 共融资 1600 万美元, 加上商业软件 Puppet Enterprise 的发布及对 Mcollective 软件的收购, 版本更新也更加迅速, 但 Puppet Labs 的目标不只是让版本发行更迅速, 还要极大地提升性能。我们可以看到 2.7 较之前版本在编译性能上提升了 60% 以上, 3.0 版本对 Ruby1.9 提供了完美支持、Master 在 CPU 消耗上提升了 6 倍的性能、提供了更多发行版操作系统的支持。特别是在 2012 年 5 月, Puppet Labs 率先与 OpenStack 整合, 这非常振奋人心。由此可见, Puppet Labs 的前景不可估量。

Puppet 从 0.25.x 直接跳到大版本 2.6 且逐渐不再维护 0.25.x, 在 2010 年时通过 Yum 安装的 Puppet 还是 0.25.x 版本。有人会问 2.6 就比 0.25.x 性能好吗? 其实不然, 2.6 主要是在功能上的改进, 为此 Puppet Labs 改变了版本命令方式, 增加了新的特性并取消了 XML-RPC 传输层, 除此之外没有其他大的改动。因此我们可以理解为, 只是命令方式的变更。

虽然目前 Puppet 在短短 2 年内由 2.6 发展至 3.0, 从 3.0.0 版本开始, Puppet 使用的是严格的 3 段版本号, 最左边为大版本号 (功能向后兼容), 中间为新功能变化, 最右为修复 bug。伴随着版本的更新官方文档也在快速更新当中, 这为我们学习提供了很大的便捷, 也增强了我们掌握 Puppet 的信心。

由于 Puppet 0.2x 支持的特性非常少, 不建议再安装使用它。对于正在使用此版本的用户, 建议升级成最新版。对于新用户, 建议直接安装 Puppet 3.0 版本。对于正在使用 Puppet 2.6 或 2.7 的用户, 建设逐步升级, 且先升级 Master, 然后再升级 Agent, 平滑过渡, 以避免遇到不可预知的问题。

提示

更多 Puppet 版本信息可参考: http://projects.puppetlabs.com/projects/1/wiki/Release_Notes。

1.1.3 版本语言特征

Puppet 各版本之间的语言也存在着差异, 当然, 版本越高, 支持的特征越多, 具体如下表 1-1 所示。

表 1-1 Puppet 各版本语言特征差异对比

语法	0.24.x	0.25.x	2.6.x	2.7.x	3.x
运算符	支持	支持	支持	支持	支持

(续)

语法	0.24.x	0.25.x	2.6.x	2.7.x	3.x
多资源的关系	支持	支持	支持	支持	支持
类的继承	支持	支持	支持	支持	支持
追加变量	支持	支持	支持	支持	支持
类命名	支持	支持	支持	支持	支持
C 注释风格	支持	支持	支持	支持	支持
节点正则	不支持	支持	支持	支持	支持
变量表达式	不支持	支持	支持	支持	支持
正则表达式	不支持	支持	支持	支持	支持
条件表达式	不支持	不支持	支持	支持	支持
链接资源	不支持	不支持	支持	支持	支持
哈希	不支持	不支持	支持	支持	支持
类的参数化	不支持	不支持	支持	支持	支持
运行阶段	不支持	不支持	支持	支持	支持
In 语法	不支持	不支持	支持	支持	支持
Unless 语法	不支持	不支持	不支持	不支持	支持

1.1.4 命令差异

Puppet 命令在 2.6 版本发布时进行了变更,且在发布 3.0 版本时将 2.6 版本之前的旧命令完全丢弃。具体差异对比如表 1-2 所示。

表 1-2 不同版本 Puppet 的命令差异对比

2.6 版本之前	2.6 版本之后
Puppetmasterd	Puppet master
Puppetd	Puppet apply
Puppetca	Puppet cert
Ralsh	Puppet resource
Puppetrun	Puppet kick
Puppetqd	Puppet queue
Filebucket	Puppet filebucket
Puppetdoc	Puppet doc
Pi	Puppet describe

1.1.5 Puppet 3.0 新特性

目前 Puppet 最新版为 3.0,其中增添了许多新特性。

□ 性能提升: 目录编译采用 JSON 作为目录缓存。

□ 增强操作系统与平台支持: 对 Ruby1.9 的完美支持,对操作系统 Windows、Solaris 包和服务的更多支持, Yumrepo 对 SSL 的支持。