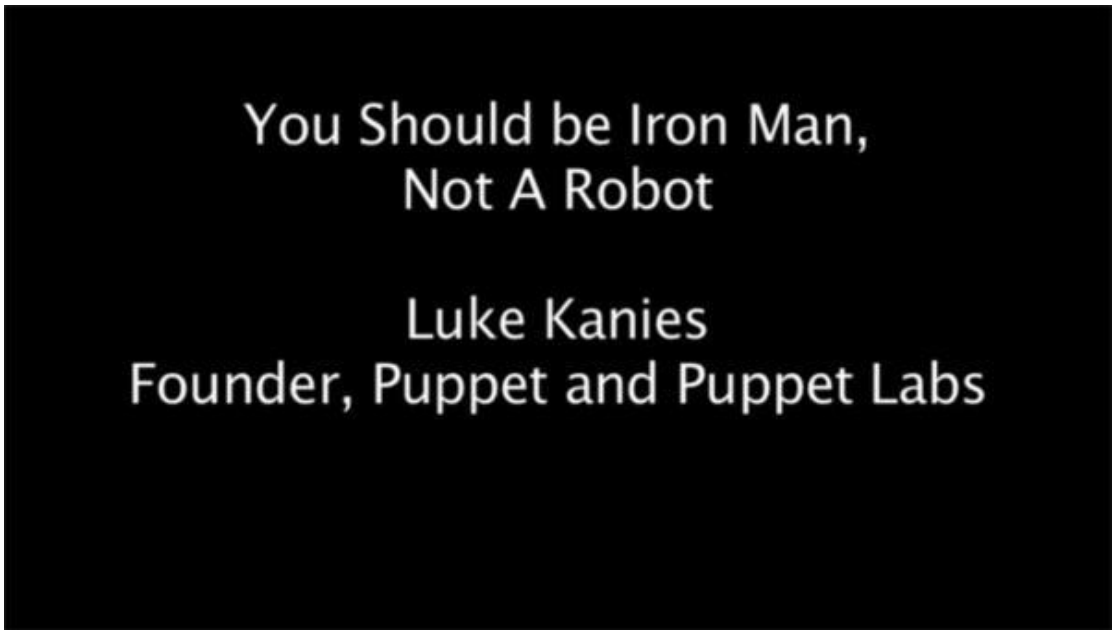
Puppet——Luke Kanies 的钢铁侠



注：本文来自互联网转载 http://www.ituring.com.cn/article/53193，感觉总结的不错，给大家分享一下：

Puppet 对于做 DevOps 的同学来说，是个非常熟悉的名字，但仍有人还不了解它。那么我先来简单介绍一下：Puppet 是由 Puppetlabs 公司开发的系统管理框架和工具集，被用于 IT 服务的自动化管理。由于良好的声明式语言和易于扩展的框架设计以及可重用可共享的模块，使得 Google、Cisco、Twitter、RedHat、New York Stock Exchange 等众多公司和机构在其数据中心的自动化管理中用到了 puppet。半年一度的 PuppetConf 大会也跻身于重要技术会议之列。AWS 的 CloudFormation 文档中有一段关于 Puppet 的介绍，其开头是这么说的: Puppet has become the de facto industry standard for IT automation。

同时，puppet 在 Openstack 中也发挥着重要的作用：Openstack-intra 社区将其用于 Openstack wiki 系统,持续集成系统等等的运维管理；此外社区的 puppet-openstack 项目用于完成 Openstack 服务的自动化部署和管理，目前已经在 stackforge 中托管并通过 Openstack 的 Gerrit 系统来管理代码提交；此外，Cisco,RedHat,Miriantis 等多家公司的 Openstack 发行版或部署工具中均使用到了 puppet-openstack。目前，Puppet 在 UnitedStack 的日常运维管理和产品的自动化部署中也起到了重要作用。

好了，刚刚还不了解 Puppet 的读者们现在已经知道 Puppet 是一个牛逼哄哄的自动化运维管理工具。可能有人已经下完了软件包跃跃欲试了，先别急，有关 puppet 的使用资料在网上可以搜到一大堆，官方的文档也详细到了“令人发指”。关于 puppet 的使用经验分享和各种特性的深入探讨以及如何使用 Puppet 管理 Openstack 部署的方案分析，哦，都不在本文的讨论范围之内。 本文的重点是八卦一下 Puppet 为什么可以这么成功。同时，为避免引起非 Puppet 程序员感官上的不适，我屏蔽了各种代码级别的展示和对细节的探讨。

故事要从 Luke Kanies，这位 twitter 昵称与 puppet 的服务器端进程名同名的哥们说起，在很久很久以前...

成长：苦逼的学生时代

在 1992 年的时候，Luke 进入诺思兰学院成为了一名化学专业的学生，这是一所位于威斯康辛州的小学校，全美排名在 178 左右徘徊。

小伙子很争气只呆了一年就跑到了大名鼎鼎的里德学院，成为了乔布斯的校友。不过倒霉的是里德学院被评为全美十大苦逼学校之一，6 年毕业率仅为 75%，也就是有 1/4 的同学拿不到毕业证。有别于很多文理学院自由选课的模式，里德的大一学生必须完成规定的人文必修课程，学习希腊及罗马的古典文化。这门课程已经有超过 50 年的历史，里德动用了学校最为强大的师资力量来为学生奠定文化基础。这还没完呢，之后学生还必须在四个拓展领域选课：文学、哲学、宗教、艺术方面;历史、社科、心理学方面;自然科学方面;还有数学、逻辑、语言学或外语。大三学生必须通过专业测试，大四学生则必须完成专题毕业论文才能毕业。毕业论文并不可怕，可怕的是里德学院的毕业论文的学时是一年，所以有想出国留学的同学，请密切关注另外九所大学的名字…

该来的还是会来的。1996 年，Luke 大四了。要想顺利毕业，那么他必须得修完这长达一年的论文项目，这意味着在这一年内他必须要动手设计和实现，并用实验数据证明，最终组织成论文来完成课题。Luke 的论文题目是 Site-directed Mutagenesis in Soy Cytosolic Ascorbate Peroxidase，我推敲了半天，中文翻译大致是：大豆抗坏血酸盐过氧化物酶胞质的定点诱变。

打工：漂泊和积累

万幸的是，我们不用去研究一口气都念不完名字的论文。扯远了，Luke 毕业后没去找一份和化学相关的工作而是去了 Cypersite 当起了 Mac 系统管理员。在 Cypersite 的日子里，

Luke 使用 AppleScript 干着行 MacOS 的管理工作，不过干了不到一年还没转正的时候，他便跑路了。

因为在 97 年的 12 月份，他在 Metro One Telecommunications 找到了一份系统管理员的活儿。Metro One Telecommunications 当年可是纳斯达克上市公司，主要业务是提供电话号码查询服务。在其巅峰时，公司在全美拥有 7000 名雇员，然而在 2009 年初，在售完最后一部分的经营业务后，该公司还剩余 3 人。瞄了一眼 Metro One 今天的股价：0.01 美元。通信行业早已是昨日黄花了，吴军博士已经在《浪潮之巅》中将这些历史描述得尽致淋漓。

又扯远了，在 Metro One 的 1 年零 9 个月里，Luke 的主要工作是管理分散在全美的 30 个呼叫中心，包括了呼叫中心计算设备的部署和设置，外加从总部对其进行不间断的维护。看到这里，我突然想到某家 startup 的创始人之前在电信部门做过相同的工作，后来他去做了一个很炫的可视化部署工具，我猜测通信行业的部署工作充满了重复的机械劳动，有一种自动化的强烈需求。这里提一下，provision 曾是电信行业中的一个术语，专指安装通信设备前的准备工作，而在 devops 中常说提起的 bare-metal provision 是指在计算机上安装操作系统或者 hypervisor 的过程。

在 1999 年，Luke 离开了这家电话公司，在 BlueStar 担任系统工程师，BlueStar 是一家卖解决方案的经销商，主营业务有：RFID, Auto ID, POS, Mobility products。Luke 主要负责一家 DSL ISP 服务器端基础架构的设计和实现。他构建了一套自动化且可集中管理的系统，并负责基础架构项目的持续开发以及实施和维护。

Luke 在“蓝翔”干了还没到两年，又跑去了卡特彼勒融资服务公司担任顾问，提供系统自动化管理相关的咨询。我查了下，卡特彼勒公司位列世界 500 强，成立于 1925 年，是世界上最大的工程机械和矿山设备生产厂家、燃气发动机和工业用燃气轮机生产厂家之一，也是世界上最大的柴油机厂家之一。我在网上没有查到 Luke 在这段时间具体干了些什么，只能感叹 Luke 作为一个系统管理员是怎么混进一个高帅富的融资公司担任顾问的。

创业伊始：对轮子的修修补补

在卡特彼勒待了两年后，Luke 开始单飞，找人合伙创建了 Reductive Consulting（2010 年改名为 Puppetlabs），头衔是独立顾问，专门从事 Unix 基础设施自动化相关的咨询。他着手研究当前开源的系统管理和监控工具如 CFEngine,ISconf,Nagios 等等，并且通过二次开发把这些工具联结成一套满足客户需求的解决方案。这是一个重要的阶段，Luke 开始将积累多年的经验和思考转变为对工具的改写，这期间他的主要工作包括重写了 CFEngine 的解析器和开发了 ISConf3，这对后来的 Puppet 开发工作产生了重要影响。

CFEngine 简介

首先，来说一下大名鼎鼎的 CFEngine，这是一款出生于 1993 年的老牌系统配置管理工具，CFEngine 的作者 Mark Burgess 希望可以使简单的管理任务自动化，使困难的任务变得较容易。它的核心理念是使系统从任何状态都能收敛到一种理想状态。 这样的工具对当

时还在使用零零散散的脚本来管理机器的系统管理员来说简直就是冬天的棉袄，夏天的雪糕，黑暗中的灯泡，饥饿中的面包。之后，CFEngine 自然而然地成为了系统配置管理工具中的标杆。 CFEngine 有两种工作模式：既可以使用 standalone 模式即通过 cfagent 来完成单台服务器的配置管理工作，也可以通过 C/S 架构（cfservd 和 cfagent）来管理整个集群的配置管理的分发工作。 CFEngine 的工作方式是基于脚本分发：在配置文件中有一个参数称为 shellcommands，用于配置要执行的命令或脚本，actionsequence 则用于设置 shellcommands 的执行顺序。 再来看看 CFEngine 的版本更新：1993 年，CFEngine1 发布；1998 年，CFEngine2 发布，而十年后，直到 2008 年，CFEngine3 姗姗来迟，第三版发生了巨大改变，可以通过使用 DSL 来定义系统状态，以至于其不能再兼容旧版本 CFEngine2 的配置语言。本文谈到 CFEnigne 时，是指 CFEngine2。

ISconf 简介

ISconf 则是另外一款配置管理工具，它的核心理念是系统的最终状态是一致的，即使被管理的机器是关机状态，当它们完成启动之后，相关命令就会被执行，到达一致的状态。同时整个系统无需中心节点，命令可以在任何一台节点上执行并复制到所有节点上。ISconf 总共经历了 4 代的演化。ISconf1 和 2 是由 shell 脚本编写，Luke 在 2002 年的时候开发和设计了 ISconf3，并使用 Perl 在 2 的基础上进行了重写。

ISconf3 有三个核心特性：

- 确定性的执行顺序
- 执行中有失败时立即退出
- 状态维护

从上面的特性来看，是一个挺酷的产品。Luke 在 02 年的时候完成了开发，并在 03 年的 LISA（Large Installation Systems Administration）会议上发表一篇名为'ISconf: Theory, Practice, and Beyond'的 paper，谈到了 ISconf 的特性，开发 ISconf 中获得的经验和教训，以及和 CFengine 的集成，分析 ISconf 3 的适用场景等等。不过 ISconf 社区在介绍 ISconf3 的历史时对 Luke 的论文颇有微词，你可以在 ISconf 的官网读到这篇文章。不过，ISconf 止步于第四版，最后的版本发布时间停留在 06 年 8 月 13 日，原因未知。

萌芽：自造轮子

随着 Luke 的梦想越来越大，这些工具集也变得越来越庞大。而此时对这些现有的开源工具的修修补补已经不能满足他的需求了。最终，他意识到只有他自己，才能去创造自己想要的工具。作为一个多年的运维人员，他深深感受到了苦逼的系统管理员们需要有一个崭新的工具来使得他们的工作更高效，更便捷。因此，他开始考虑去开发一个新工具，首先这个工具是为系统管理员而设计的。那么怎么才能称为得上是为系统管理员设计的呢？

先来简单思考一下系统管理员对于配置管理工具的主要需求：学习成本低，开发高效，跨平台，代码可复用，安全，可扩展性高等等。

最好的情况就是系统管理员只需关心某个服务或者软件包的状态，例如我希望部署一个 apache web 服务器，我只关心 apache 的包是已安装的状态，服务是运行的状态，而不要让我去操心这个 apache 包是装在 Ubuntu 下的还是 Redhat 下，然后到底是要执行 yum install 还是 apt-get install，然后还要操心这个 apache 进程到底是用 init 还是 upstart 管理的。

此外，还有如何对依赖关系进行处理。先前的配置管理工具都是关注在如何去完成每一项互相独立的工作，例如配置 apache 服务就是一段 shell 脚本而已，而没有去考虑它们之间是存在关联的，例如当 apache 的配置文件发生变更时，是应该重启 apache 服务来使之生效，而重启 apache 服务的前提是系统已经安装了 apache。

这对于当时占据主流的以分发 shell/perl 脚本的配置管理软件来说，简直是天方夜谭。不过 Luke 就是这么设想的，他在构思 Puppet 的设计时，就希望将系统抽象成资源：采用面向对象的概念，将每个资源类型组成为一组属性的集合，每个属性都有相应的行为，并将资源类型和属性构造成类，最终使用这些属性来使得资源到达期望的状态，而不是用资源本身来完成这些工作。同时，将所有资源的依赖关系构建成一张有向图，通过这种依赖描述，系统管理员们可以实现复杂的业务逻辑的管理。

越想越兴奋，于是 Luke 开始动手 coding 了。Puppet 的第一个原型写于 04 年夏天，但他并没有把此事放在最重要的位置上。于是在 9 月份的时候，Luke 居然跑去了 BladeLogic 担任产品设计，这是一家专门做商业配置管理软件的公司，它的产品在当时已经取得了成功，然而 Luke 发现 BladeLogic 的品味仅仅是想卖软件给大公司而不是立志为系统管理员设计伟大的工具。在 05 年的 2 月份，Luke 下决心继续搞 Puppet 的开发工作。于是，在 BladeLogic 呆够了 7 个月后，Luke 选择了离开。BladeLogic 也终于如愿以偿：被 BMC 收购了，当然这是后话。

艰难抉择：开发语言和设计哲学

随后 Luke 回到了 Reductive Consulting，也就是 Puppetlabs 的前身，开始了 Puppet 的全职开发。不过他遇到了一个棘手的技术问题：身为一个 Perl 程序员，Luke 痛苦地发现，Perl 居然无法处理那些 puppet 类之间的关系。那时 Python 被认为是系统开发的最佳选择，但是 Luke 同学接受不了的不是 Python 的缩进问题，而是 print 是一个语句而不是函数，len 是一个函数而不是方法的事实，让他感觉“眼睛在流血”。这时候，有个朋友告诉他 Ruby 很酷，于是他在尝试了 4 个小时后，就写出了一个功能原型，从此不可自拔。不过他也有点担心，在当时 Ruby 还属于非主流，因为 ROR（Ruby on Rails）都还没有出生，不过鉴于他的体验，觉得使用 Ruby 的开发效率非常高，因此他决定冒一次险。

在这个问题解决之后，Luke 又面临另外一个难题：虽然在此之前他已经写了不少的小工具，积攒了丰富的经验，不过这些工具没有一个的代码量是超过 1 万行的。这也意味着在设计 Puppet 的架构的过程中，肯定得摔不少的坑。因此，Luke 在开发 Puppet 的过程中始终要求自己和开发团队遵守两个指导思想：首先设计尽可能地做到简洁，程序的可用性总是高于新特性；此外 Puppet 首先是个框架其次才是一个应用。在解决了语言选型和设计哲学的问题后，Luke 开始潜心于 Puppet 的开发。

然后呢

然后就没有然后了，随着 Puppet 的发布和版本的快速迭代，以及社区的火热发展，Puppet 自然而然地就发展成了今天的模样。本来是要开始深入技术细节展开介绍，但是应广大群众要求保持纯八卦风格的呼声，下半部分在内部审批时被惨无人道地砍掉。对于 Puppet 和 Luke 的介绍到此告一段落，咱们再来八一下群众们喜闻乐见的热点事件。

投资风云

事情源于两年前，VMWare 作为 Puppetlabs 的 6 家投资公司之一，向 Puppetlabs 投了 850 万美刀。在今年的 2 月份，VMWare 宣布提高对 PuppetLabs 的投资：3000 万。此外，VMware 负责云架构和管理的执行副总裁 Raghu Raghuram 将进入 Puppet Labs 董事会。

这从资本市场的角度说明了 Luke 和 Puppet 的成功，叫好又叫座。但是也引发了人们的担忧：一些人认为 Puppet 的独立性是非常重要的，理由是 DevOps 的成功非常依赖于这个组织选择什么样的工具用于企业基础设施管理，Puppetlabs 作为 IT 自动化管理领域里的领头羊，然而 VMware 对 Puppet 存在强大的话语权，目前 Puppetlabs 花了很大的精力在 Puppet 企业版本的开发上(Puppet PE)，用于和 VMWare vCloud Automation Center 等产品的集成。这并不是一件好事，可能会使得社区版本和商业版本产生巨大的差距，甚至可能会发生 Oracle 和 Mysql 那样的故事。

再看看 VMWare 的老对手 Amazon 的反应，即使 Amazon 在开头提及的文档里承认 Puppet 是事实上的行业标准，但考虑到它的老朋友 VMWare 手握对 Puppetlabs 2/3 的投资，在其新推出的 OpsWork 服务中清一色地使用了 Chef 用于系统的配置管理，这是诞生于 Puppet 之后的另外一款开源的配置管理工具。

再来听听其他人的声音。Matt Asay 与 Luke 是多年的好友，在 Lukes 与 Andrew Shafer 创立 Puppet Labs 后，Matt 曾经提出了许多商业化建议，包括进行融资，但 Luke 更希望公司保持独立。Matt 淡定地回答记者：如果你了解 Luke Kanies 这个家伙，你就知道保持独立对于他而言是无比重要的事。他在田纳西州长大，不是那种出卖灵魂的人。

Luke 在 IM 上和 Matt 表达了自己的看法："向云端转换的趋势将给下一代系统平台管理工具带来无限的机会，对 VMware 如此，对 PuppetLabs 亦是如此。即便 Puppetlabs 被 VMware 掌控，但这些系统仍构建在开源的基础上，并赋予它们一定的自治权。Puppet 拥有活跃的开源生态圈，就像 OpenStack 一样，是对投资很好的补充。"

最后谈谈我的看法，首先 Puppet 天生就是开源血统，其社区非常活跃，而且从 2.7 开始变为更为友好的 Apache 许可证（之前是严格的 GPL）。因此，即使发生像 MySQL 那样的事情，照样会有另一家 MariaDB 出现，更何况还有后起之秀 Chef 在一旁虎视眈眈，VMWare 理应不会下这步昏棋。同时，对于指责 Puppet 只关注于与 VMware 产品的集成是有失偏颇的，虽然最近 PE 的动静很大，但是 Puppetlabs 也花费了很大的精力投入到 Puppet-Openstack 社区中，在之后的 Puppet 系列博文中会对此进行阐述。所以，我并不担心 Puppet 是否会成为下一个 MySQL。

我们学到了什么？

在前面洋洋洒洒的一堆八卦中，我简单地谈了一些关于 Luke 和 Puppet 的故事。每个人可能都会从中看到不同的东西。对我而言，Puppet 之所以能取得成功，我认为有：

做自己最擅长的事：Luke 拥有丰富的运维经验和配置管理工具的开发经验，在此基础上去开发 Puppet 是获得成功的基础之一； 取其精华：在调研同类工具和产品时，Luke 不只看到了其中的不足，还吸收了它们的优点； 志存高远：Luke 开发 puppet 时候的动机是为系统管理员提供最棒的 CMS 工具，而不仅为了挣钱，被收购，上市。动机越单纯，成功的概率越大； 敢想敢做：现在去挑选手机，我们几乎不会去考虑手机是不是触摸屏的而是去看屏幕有多大，PPI 多高，因为手机是触摸屏这是理所当然的，然而在 iPhone 之前，触摸屏手机还是很稀有的东西，大家都在往功能更多的方向越走越远；同样，现在在使用 puppet 时，会觉得使用声明式配置语言是理所当然的事情，殊不知在 puppet 诞生的同时代中，都是清一色的命令式或者过程式的配置语言。 免责声明

本文的所有情节来自本人花费一周的夜生活时间，阅读了各式各样的博客，新闻，评论，代码，勉强拼凑而成，若有与实际不符之处，请一笑而过。

为了能够和大家更好的交流和学习 Puppet，本人 2014 年又新开辟了微信公众号进行交流学习，目前已经有 300 多人同时收听，喜欢 Puppet 的大神们可自行加入哦。

如果你有好的有关 Puppet 的咨询也可以给我投稿，投稿地址：admin@kisspuppet.com

微信公众号：“puppet2014”，可搜索加入，也可以扫描以下二维码