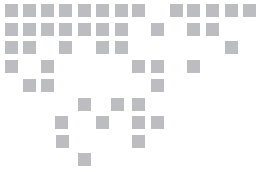
第一部分 Part 1

基 础 篇



Chapter 1 第 1 章

运维工程师的利器——自动化运维工具

随着网络云时代和大数据时代的到来，运维工程师负责管理的服务器数量也成倍地增长。如何管理好这些服务器为云时代和大数据时代保驾护航，是摆在运维工程师面前的一道难题。而解决这道难题就需要运维工程师对自动化运维工具的掌握达到一定的程度。笔者希望通过本章抛砖引玉，结合自己的经验介绍多年来使用自动化运维工具的心得和体会。本章首先介绍互联网运维工程师的职责、优秀运维工程师和普通运维工程师的区别；然后简要介绍常见的自动化运维工具；最后比较当前常见的自动化运维工具的优势，以此说明为什么要选择 Puppet 做我们的自动化运维工具。

1.1 浅谈运维工程师

想必大家都看过《好的程序员是普通程序员效率的数十倍》这篇文章，这句话是比尔·盖茨说的，被很多文章引用和转载。笔者读后感同身受，觉得这篇文章讲的并不夸张。程序员如此，运维工程师也是如此，一个优秀运维工程师的效率确实是普通运维工程师的数十倍。本节笔者将带领大家了解一下优秀运维工程师和普通运维工程师之间的不同之处。我们从运维工程师的定位和职责开始介绍，继而详细分析普通运维工程师和优秀运维工程师的差别，最后落脚到自动化运维工具。

1.1.1 运维工程师定位和职责

要想了解普通运维工程师和优秀运维工程师之间到底有什么不同，首先要了解运维工程师的定位，也就是运维工程师处于工作链的哪一环节；然后要了解运维工程师的职责是

什么。下面笔者将分别进行介绍。

1. 定位

首先我们来看运维工程师处于工作链的哪一环节。我们都知道，公司的老板需要根据市场提出指导思想；项目经理需要对市场状况进行调研分析；产品经理需要根据老板的指导思想，结合项目经理的调研分析设计出产品雏形并提交给开发工程师。在此期间，系统工程师和网络工程师需要根据产品选择 IDC、搭建网络环境、分配网络相关资源；开发工程师编写代码，并将开发后的产品提交给测试工程师；测试工程师对产品做系统和功能测试，将稳定的代码提交 SVN，这时轮到运维工程师上场了——运维工程师将代码从 SVN Check Out 出，推到线上系统供用户访问，并进行后续的升级维护等工作。从图 1-1 不难看出，运维工程师处于最后一个环节，也是最重要的一个环节，运维工程师负责产品后期的维护、升级、监控和服务保障等。如果运维工程师在日常运维中工作做得不到位就直接影响线上用户的访问和体验，后果是比较严重的。

图 1-1 产品研发流程图

需要注意的是，不同行业的公司对运维工程师的定义也不同。笔者在这里是以互联网公司为背景介绍的，当然即使同样处于互联网行业中的公司也会存在一些差别。规模小一点的公司，一个运维工程师可能需要完成包括系统工程师、网络工程师、测试工程师及部分开发工程师的工作内容；规模比较大的互联网公司，运维工程师的工作职责可能会比笔者介绍的更细致些。

2. 职责

有关运维工程师的职责，我们在这里同样以互联网公司为例。笔者将整个运维工作的链条细化为以下 3 个部分。

- 基础运维：硬件选型、IDC 部署、架构规划、容量规划、资源成本管理、预算控制、容灾、高可用设计和架构可扩展性等。
- 业务运维：发布变更、监控告警、故障处理、软件安装升级、系统调优、运营质量报告、线上环境备份、运营大数据分析、恢复和线上程序发布维护等。
- 数据运维：存储架构设计、提升数据的均衡负载能力、数据的备份与恢复、容灾的建设、数据的冷热分离、监控告警、DB 调优与 SQL 优化等。

1.1.2 优秀运维工程师 vs 普通运维工程师

优秀运维工程师和普通运维工程师确实存在着一些差别，特别是在一些大的互联网公司内，一个优秀的运维工程师在带来稳定服务的同时还可以为公司节约可观的成本。本小

节笔者结合自己的工作经历和体会，来分析一下优秀工程师和普通运维工程师究竟存在哪些差别。笔者希望通过本小节的介绍能使所有的运维工程师找到一点儿成为优秀运维工程师的灵感。当然很多运维工程师已经很优秀了，但是没有最好只有更好，我们可以综合其他优秀运维工程师的优点，让自己变得更优秀。

在《好的程序员是普通程序员效率的数十倍》中有这样一段话："很多程序员每天编码超过 2000 行，对于一个程序员来说这很正常，但我们可以说这是一个好的程序员吗？我们再来看一下，网上报道著名的软件公司——微软公司，他们的程序员每天只编写 50 行代码，差距这么大，难道微软公司的员工就不是好的程序员？"从这段话来看，程序员并不是每天写的代码越多就越优秀。优秀的程序员能够写出正常运行的程序是基本要求；但还需要通过算法提高程序的效率，降低程序使用的系统资源，同时提高程序的复用性和易用性。运维工程师也是一样，并不是每天的工作量大就能成为一个优秀的运维工程师。笔者自己做了比较，刚入行的普通运维工程师和工作多年的优秀运维工程师有着明显的差别。差别主要在以下两方面。

1）**对运维工具集的使用**。刚入行的运维工程师习惯自己写脚本来完成运维工具建设和日常的运维工作。笔者当年也一样，每次接到新任务都会重新写一套脚本来完成新项目。笔者在工作多年后，也时常看到很多刚入行的运维工程师采用和笔者当年一样的工作方式，这样无疑是比较耗费时间的。单就这方面来说，优秀运维工程师和普通运维工程师相比，主要有以下特点。

- ❑ 撰写通用的脚本。优秀运维工程师会找到每个项目的共同点写出通用的脚本，尽量降低重复劳动，提升工作效率，同时将通用的脚本共享。这样不仅降低了其他人的工作成本，还提高了整个组或部门的工作效率。
- ❑ 尽量使用开源工具。开源工具本身就节省了开发的成本。如果开源工具不能满足项目需求，优秀运维工程师会对开源工具做二次开发。这样一方面节约了大量的成本，另一方面他们也可以借鉴开源工具的一些思路和想法来拓展自己的视野。

2）**时间管理**。在互联网行业中，运维工程师和其他工种有很大的差别。运维工程师的时间比较零散，比如他们可能前一秒钟还在升级线上程序，后一秒钟由于程序 BUG 导致线上服务不可用需要他们立即停止正在进行的工作，准备迁移线上流量。运维工程师好比一个消防员，每天处理的都是十万火急的问题，一点都疏忽大意不得。而这样忙碌地工作一天后，总结起来可能实质性的工作并没有很多。其实这是很多运维工程师的弊病。所以想要成为一个优秀运维工程师，就要合理地管理和利用时间。好比一个球队在踢球前会预先设置好阵形，然后在比赛的过程中通过变换阵形来打赢比赛。运维工程师的工作也是如此，需要合理分配时间，平日可以用 70% 的时间来处理日常的运维工作，20% 的时间通过程序或工具将现有流程实现自动化，用 10% 的时间来提前准备明天的工作。只有这样合理分配时间，才能让我们的日常运维工作事半功倍、井井有条。

除了上面提到的对运维工具集的使用和时间管理外，成为一名优秀运维工程师还需要

很多其他方面的能力，比如：

- 强烈的学习欲望和钻研精神。
- 良好的沟通和团队协作能力。
- 技术分享、沉淀、总结、传承能力。
- 行业洞察力。
- 好的身体素质和吃苦耐劳精神。
- 良好的编程能力。
- 胆大心细，敢于承担责任。
- 规划能力。
- 好的心态和情商。

1.1.3 自动化运维工具

上面我们对比了一下优秀运维工程师和普通运维工程师的区别。读到这里想必大家心中也有了基本的了解。伴随着互联网的云时代和大数据时代到来，早年的运维工程师只需要管理几十台、几百台服务器，而到目前为止很多运维工程师需要管理几万台服务器。如果没有一套完善的运维工具，这样的工作量和维护成本是不可想象的。所以想要作为一个优秀的运维工程师，就要借助运维工具来让我们更好地完成日常运维工作。在这里笔者重点向大家推荐 3 款功能强大的自动化运维工具——Cfengine、Chef 和 Puppet，它们也被誉为自动化运维工具中的“三把斧”。下一节我们将向大家简单介绍一下这 3 款工具。

1.2 自动化运维工具箱

1.2.1 Cfengine

Cfengine 是一个借助 C 语言开发的、功能强大的自动化 UNIX 管理工具，最早出现于 1993 年。通过 Cfengine 可以轻而易举地管理客户端上的设备。Cfengine 不仅运行成本低、效率高、功能强大，而且使用范围广。Cfengine 可以管理各种环境下的设备，从一台到上千台服务器的集群均适用。如果运维工程师想同时修改 2000 台服务器的 root 密码，通过 Cfengine 可以轻松地在几分钟内实现。Cfengine 还包含以下主要的功能：

- 检查和配置网络接口。
- 编辑系统和用户的文本文件。
- 维护符号链接。
- 检查和设置文件的权限。
- 删除垃圾文件。

- 检查重要文件和文件系统的存在。
- 控制用户脚本和 shell 命令的执行。
- 基于类的判定结构。
- 程序和系统进程管理。

关于 Cfengine 更多信息，请大家参考 Cfengine 官方网站 http://cfengine.com/。

1. Cfengine 生命周期

Cfengine 工作时遵循的是系统生命周期管理的 Build-Deploy-Manage-Audit（BDMA）模式（如图 1-2 所示）。BDMA 包含系统生命周期的 4 个阶段：构建（Build）、部署（Deploy）、管理（Manage）和审计（Audit）。

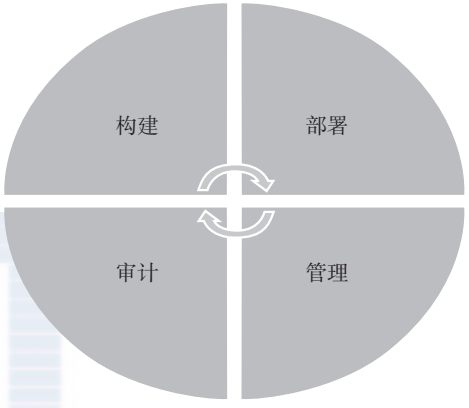


图 1-2 Cfengine 系统生命周期

下面分别介绍一下 BDMA 系统生命周期的 4 个阶段。

- 构建阶段：需要计划策略更改、规划想要的状态承诺（promise）以及构建所建议承诺的模板，这样如果所有机器均能做出并兑现这些承诺，系统便可无缝地运行。
- 部署阶段：需要向所有自主客户端（autonomous clients）发布策略，并且每个客户机都要运行一个代理，无需协助即可实现并维护这些策略。
- 管理阶段：这个自主代理负责管理该系统，运维工程师只需处理不能被自动处理的极少事件。
- 审计阶段：对更改进行本地审计和维护。决策结果由 Cfengine 内的设计确保且可自动维护。

2. Cfengine 如何工作

Cfengine 的工作过程如下。

1）管理员登录主服务器更新配置文件（SVN），通过运行 Cfrun 命令通知客户端进行更新。Cfrun 在 Cfrun.hosts 文件中查找客户端的列表。

2）Cfrun 与每个客户端上的 Cfservd 进行通信，然后 Cfservd 运行 Cfagent。

3）Cfagent 连接主服务器，首先检查 Update.conf 是否有新版本，如果有更新，则将它传输到客户端。

4）Cfagent 先评估 Update.conf 的内容，并获取策略文件（Cfagent.conf 和相关文件）的最新版本。随后评估 Cfagent.conf 以确定客户端是否处于所需状态。如果有偏差，Cfagent 将执行已定义的操作来更正客户端配置。

1.2.2 Chef

1. 什么是 Chef

自动化管理工具 Chef[⊖]由 Ruby 语言开发，是一种可以将框架转换为代码的自动化工具平台。人们不必关心设备是虚拟机还是物理机，也不必关心是几百台还是几千台服务器的集群，Chef 都可以方便自如地管理资源、进程、系统等信息。目前很多互联网公司也在应用 Chef，如 Facebook、亚马逊等。

Chef 服务器能够存储用户的配置数据和 Recipes，其中配置数据用于描述基础设施的所有组成部分，而 Recipes 将这些组成部分集合为一个完整的运行系统的指南。无论是在现场的实体和虚拟服务器上，还是在云端的实体和虚拟服务器上，Chef 用户都可以在系统的节点使用 Recipes。随着基础设施的变化和发展，用户可以使用工作区域随时更新 Chef 服务器。通过使用版本控制可以获取所有的更改。

2. Chef 如何工作

如果忽略所有的细节，Chef 是这样工作的：在工作站（Workstation）上定义各个客户端（Client）应该如何配置，然后将这些信息上传到中心服务器，每个客户端连到中心服务器工作站查看如何配置，然后进行自我配置。因此，在 Chef 的环境搭建完成以后，绝大部分工作是在工作站上进行的，客户端要获取工作站配置时，会主动连接并按照工作站的配置应用到客户端上，客户端并没有额外的工作。Chef 主要有以下 3 种运行模式。

- Chef-Solo：由一台普通计算机控制所有的服务器，不需要专设一台 Chef-Server。
- Client-Server ：所有的服务器作为 Chef-Client，统一由 Chef-Server 进行管理，包括安装、配置等工作。Chef-Server 可以自建，但安装的东西较多，由于使用 Solr 作为全文搜索引擎，因此还需要安装 Java。
- Opscode Platform ：类似于 Client-Server，只是 Server 端不需要自建，而是采用官方网站提供的 Chef-Server 服务。

上面 3 种管理模式中，无疑 Client-Server 模式是最好的，但是同时也是最复杂的。因为通过这一模式可以在本地环境中搭建一个私有的 Chef 集中管理环境，而无需依赖任何第三方平台。

1.2.3 Puppet

1. 什么是 Puppet

Puppet[⊖]是一款使用 GPLV2X 协议授权的开源管理配置工具，用 Ruby 语言开发。其既

⊖ Chef 官方网站 http://www.getchef.com/。

⊖ Puppet 官方网站 http://puppetlabs.com/。

可以通过客户端 – 服务器的方式运行，也可以独立运行。Puppet 可以为系统管理员提供方便、快捷的系统自动化管理。对于系统管理员来说通过 Puppet 配置管理系统，底层的操作系统的发行版本是透明的，Puppet 通过（Provider 又称提供者）属性来完成软件的配置与安装，管理员不必关心操作系统的种类与发行版本，如图 1-3 所示。Puppet 还可以提供一个强大的框架来完成系统管理功能，在框架的基础上系统管理员可以通过 Puppet 语言来描述系统的一些事务，如安装软件、初始化系统、启动、删除服务、推送配置文件和差异化配置管理服务器等。同时系统管理员和系统管理员之间可以分享用 Puppet 语言描述好的事务，从而减少重复劳动，提高工作效率。

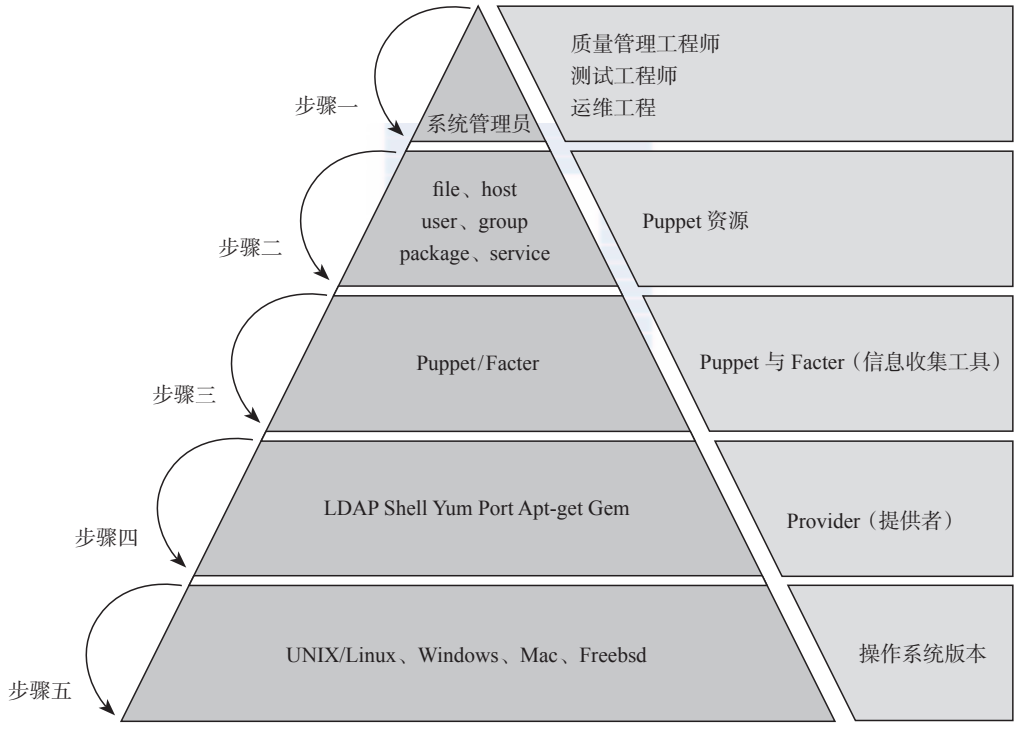


图 1-3 Puppet 管理操作系统流程图

Puppet 主要由 Luke Kanies 和他的公司 Puppet Labs 开发和维护。Kanies 从 1997 年开始从事 UNIX 的系统管理，并于 2005 年创立了一家专注于自动化工具的开源软件公司 Puppet Labs。不久之后，Puppet Labs 发布了他们的旗舰产品 Puppet。

2. Puppet 工作模型

Puppet 可以用来管理 UNIX/Linux 平台，同时也添加了对微软 Windows 的支持（要注意的是目前 Puppet 只对微软的 Windows 做客户端支持，并不能将 Windows 用作 Master 来管理其他的服务器）。

那么 Puppet 是如何工作的呢？目前 Puppet 有一个简单易懂的工作模型，如图 1-4 所示。其主要分为 3 层，分别是部署和调度层、配置语言和资源抽象层、事务层。

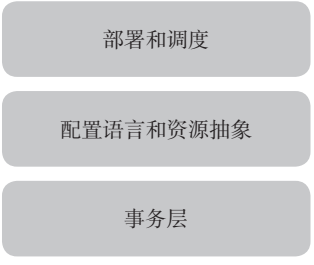


图 1-4 Puppet 工作模型

（1）部署和调度层

Puppet Master（Puppet 服务器，下称 Master）在一台机器上以守护进程的方式运行，同时还包含各客户端节点的配置信息。Puppet Agent（客户端，下称 Agent）在与 Master 通信的过程中，通过标准的 SSL 协议进行加密和验证，验证通过后，Agent 从 Master 上读取相应的节点配置信息。

需要注意的是，并不是每次连接 Agent 都会从 Master 上读取信息，只有该节点在 Master 上配置信息发生变化时才会被读取。

默认情况下 Agent 每 30 分钟连接一次 Master。但是这种方式在很多场景下不是很符合系统管理员的要求，所以很多系统管理员也会将 Agent 通过 Crontab（UNIX 定时任务计划）来管理，这样会更加灵活一些。

（2）配置语言及资源调度层

Puppet 使用描述性语言来定义配置项，在 Puppet 中将配置项被称为 Resource（资源）。这种描述性语言使得 Puppet 与其他配置工具截然不同。描述性语言在 Puppet 中还可以声明配置的状态，例如一个软件安装、配置、启动的各环节、上下游依赖关系等。

让我们来看这样一个例子。系统管理员需要在 CentOS、Ubuntu 的主机环境安装 Nginx 服务，如果不借助工具我们需要通过脚本来完成以下步骤。

步骤 1 连接到目的主机，输入用户密码或密钥。

步骤 2 检查是否安装相应 Nginx 服务。

步骤 3 如果没有安装，根据系统发行版本在系统上执行不同的命令。CentOS 可以执行 yum 命令，Ubuntu 可以使用 atp-get 命令，安装后启动 Nginx。

步骤 4 将安装后的信息返回给服务器。

而通过使用 Puppet，我们只需要在 Master 服务器的相应配置文件中通过配置语言定义一个 Package 资源。Package 资源的定义格式如下：

```
资源名 { '标题':
   属性 => 值
}
```

例如：

```
package { 'nginx':
  ensure => present,
}
```

在资源名里填写相应的标题，如 nginx，并将资源的属性 ensure 赋值为 present。这里

的属性 ensure 表示软件包的安装状态，present 表示希望安装这个 Nginx 软件包，而 absent 则表示希望卸载 Nginx 软件包。这样当 Agent 来连接 Master 时就会自动安装 Nginx 服务，并将安装好的消息以报告的形式上报 Master。

当 Agent 连接 Master 时，Master 并不知道 Agent 的操作系统型号和版本。Agent 通过 Facter 工具收集系统相关信息，并通过 SSL 协议将 Agent 的信息传递给 Master。Master 根据 Agent 收集到的相关信息，通过资源的提供者来为 Agent 服务。比如 Package 资源收到 Agent 的信息后，会识别 Agent 的系统型号版本，并通过资源提供者（如 yum aptitude pkgadd apt-get 等）匹配，为 Agent 服务。

（3）事务层

Puppet 事务层其实就是它的解析引擎。Puppet 事务层配置每一台主机的过程包括以下 4 步。

步骤 1　解析和配置编译。

步骤 2　将编译好的配置同步到 Agent。

步骤 3　在 Agent 上应用配置。

步骤 4　向 Master 报告运行结果。

首先 Puppet 会创建一个图表来表示所有资源的关系和上下游执行顺序，以及和 Agent 的关系。然后 Puppet 将按照资源之间的关系和上下游顺序依次执行。

接着 Puppet 为每一个 Agent 获取相应的资源，并把它们编译成“目录”，然后将目录依次分发到各主机，并通过 Agent 来应用它们，最后应用结果以报告形式反馈给 Master。

Puppet 并不是完全的事务，因为事务会记录日志，而 Puppet 并没有记录日志，也无法像数据库那样进行回滚。

1.3　自动化运维工具对比

在 1.2 节中我们介绍了现在比较常见的自动化配置工具 Cfengine、Chef 和 Puppet，下面再来看一下这 3 款自动化运维工具的区别，如表 1-1 所示。

表 1-1　自动化运维工具区别

	Cfengine	Chef	Puppet
开发语言	C 语言开发	Ruby 语言开发	Ruby 语言开发
语法	难理解	难理解	容易理解
使用人群	用户较多	用户较少	用户较多
学习门槛	门槛高	门槛高	门槛低
安装	复杂	复杂	简单
安装配置文件	较多	较多	较少

讲到这里，我们已经基本了解了 Cfengine、Chef 和 Puppet 这 3 款自动化运维工具。通

过表 1-1 可知，Puppet 的优势还是比较明显的。若是我们去 Puppet 的官方网站[⊖]上看一看，会发现很多使用 Puppet 作为公司自动化运维工具的例子，目前超过 18000 家公司在使用 Puppet 软件，其中包括 Twitter、Zynga、美国银行、纽约证券交易所、迪斯尼、Citrix、Oracle、Google、Adobe 和 Evernote 等。另外大型招聘信息搜索引擎 Indeed.com 的数据对各大公司招聘职位描述统计发现，很多公司增加了对 Puppet 技能的要求，如图 1-5 所示。

图 1-5 Puppet 技能职位工作趋势

若是大家一直在关注 Puppet 的动态，一定会发现这则令人振奋的消息：“2014 年 6 月 Puppet Labs 宣布获得 E 轮融资 4000 万美元”，投资者包括思科、Google Ventures、KPCB 和 Vmware 等，这无疑为 Puppet 的未来发展注入了更多的活力。既然 Puppet 具有如此多的优点和明朗的发展前景，就让笔者通过本书带领大家一起走进 Puppet 的自动化运维的世界吧。

⊖ http://puppet labs.com/about/customers。