第 1 章 Chapter 1

Ansible 架构及特点

IT 行业的工作变得越来越有趣了，我们不再是把软件交付给客户，然后安装在单独的服务器上运行，我们都慢慢地变成了系统工程师。

我们现在部署应用软件的方式是通过服务串联起来，运行在一系列分布式的计算资源上并用各种不同的网络协议进行通信。常见的应用包括 Web 服务、应用服务、基于内存的缓存服务系统、任务队列、消息队列、SQL 数据库、NoSQL 数据存储、负载均衡等。

我们也需要确保采用合适的冗余，当故障发生时软件系统能够很好地处理、适应这些故障。另外有些辅助的服务需要部署、维护，例如日志管理、监控系统、分析系统，需要与第三方服务交互，如通过与 IaaS 接口交互来管理虚拟主机实例。

你可以用手动方式来搭建这些服务：安装服务器操作系统，SSH 登录每一台，安装软件包，编辑配置文件，等等。这种方式耗费大量时间还经常出错，特别是在做了 3 ～ 4 次之后，这枯燥重复的手工劳动是令人非常痛苦的。对于更复杂的任务，比如在你应用环境中搭建一个 OpenStack 云环境，由手工来操作会让人发疯。应有更好的方法。

如果你读到这里，你可能已经有了配置管理的思想，并考虑采用 Ansible 做为你的配置管理工具。无论你是一个开发人员想要把代码部署到生产环境，还是一个系统管理员寻找更好的自动化方法。我觉得 Ansible 对于这些问题都是很好的解决方案。

1.1 Ansible 软件及公司

IT 自动化配置管理最近 20 年获得了迅猛的发展，特别最近几年在移动互联、云计算、大数据、互联网 + 等大规模应用平台的需求推动下，涌现出一批成熟的大规模自动化运维工具。维基百科里列出了二十多个，其中 Puppet、Chef 和 Salt，以及 CFEngine、Vagrant 和 NixOS，大家都可能耳熟能详了。不过后起之秀 Ansible（http://www.ansible.com/）的人气更高，已经是当今最常用的管理基础架构的开源管理工具之一。

从开源仓库 GitHub 上受到使用者、开发者的关注度、加星、贡献、评论（见表 1-1）可以看出，Ansible 的受欢迎程度的数据已经远远超过 Puppet、Chef、CFEngine、SaltStack。

表 1-1 GitHub 上开源自动化工具受关注程度信息表（截至 2015 年 8 月 30 日）

开源自动化配置工具	关注（Watch）	加星（Star）	复制（Fork）	开始时间	评论数	贡献者
ansible/ansible	1 009	12 416	3 697	2012 年 2 月 5 日	15 821	1 146
puppetlabs/puppet	414	3 514	1 468	2005 年 4 月 10 日	20 618	394
saltstack/salt	451	5 573	2 370	2011 年 2 月 20 日	58 452	1 194
Chef/chef	338	3 794	1 554	2008 年 5 月 2 日	13 195	399
CFEngine/core	62	224	136	2007 年 12 月 30 日	12 544	73

Ansible 自从 2012 年 2 月发布以来，一直得到 Ansible 爱好者、用户、开发者的热情参与、持续贡献，如图 1-1 所示。

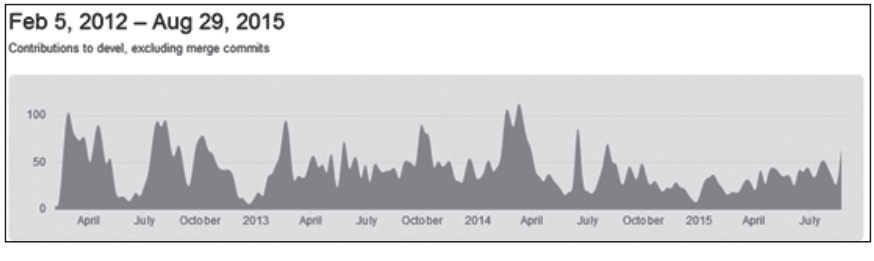

图 1-1 Ansible 受贡献者的支持趋势

Ansible 使用 Python 作为开发语言，巧妙地设计、实现了简单易用、功能强大的自动化管理工具。Ansible 由 Michael DeHaan 发起、开发、创建，他同时也是著名工具软件 Cobbler 与 Func 的开发者。Ansible 的第一个版本发布于 2012 年 2 月，目前下载量已经超过了 100 万。当前在 GitHub 上，它是排名前 10 位的 Python 项目，可以预见

Ansible 的发展不可限量。

Ansible 已经广泛应用于各种规模、各个领域的企业，包括 Rackspace、Twitter、Evernote、NASA、GoPro、Atlassian 等知名企业。

1.1.1　Ansible 应用领域

Ansible 的编排引擎可以出色地完成配置管理、流程控制、资源部署等多方面工作。与其他 IT 自动化产品相比较，Ansible 为你提供一种不需要安装客户端软件、管理简便、功能强大的基础架构配置、维护工具。

Ansible 基于 Python 语言实现，由 Paramiko 和 PyYAML 两个关键模块构建。Ansible 具有独特的设计理念：

- 安装部署过程特别简单，学习曲线很平坦。
- 管理主机便捷，支持多台主机并行管理。
- 避免在被管理主机上安装客户代理，打开额外端口，采用无代理方式，只是利用现有的 SSH 后台进程。
- 用于描述基础架构的语言无论对机器还是对人都是友好的。
- 关注安全，很容易对执行的内容进行审计、评估、重写。
- 能够立即管理远程被管理主机，不需要预先安装任何软件。
- 不仅仅支持 Python，可运行使用任何动态语言开发模块。
- 非 root 账户也可以使用。
- 成为最简单、易用的 IT 自动化系统。

在云计算时代的浪潮中，基础架构必须满足按需自动伸缩、按使用量计费的基本特性，IT 自动化运维软件就是最重要的必备工具之一。下面来看看几个关键的领域中取得的巨大的进展。

1. 配置管理

配置管理领域已经涌现出多种工具，配置管理的目标就是确保被管理的主机尽可能快速、按照正确方式达到配置文件中描述的状态，这对管理 IT 环境至关重要。例如，在网站高峰时候需要扩展新的 Web 服务器，这需要一台由配置管理控制的机器能够快速就位，这也就是通常所说的代码化基础架构（Infrastructure as code），因为构建基础架构所有必须的代码都存储在源码控制系统中。这也是逐步引入对代码化基础架

构按照软件开发生命周期（Software Development Lifecycle，SDLC）方式进行管理，这些包括辅助基础架构测试的工具有 Ansible、CFEngine、Chef、Puppet、Salt 等，基础架构测试工具有 Serverspec、Test kitchen 等。

2. 服务即时开通

这个领域的工具主要是在数据中心、虚拟化环境、云计算中快速开通新的主机。几乎所有云计算的服务提供商都有相应的 API 接口，这些自动化工具通过这些 API 接口能够快速地创建主机实例。对于基于 Linux 或最近快速发展的容器（如 Docker、LXC），越来越多的人开始采用自动化工具的方式来保证这些容器的开通。Ansible 在这些场景扮演了重要的角色。

3. 应用部署

这个领域的工具重点关注如何尽可能地零停机部署应用。许多单位已经采用滚动式部署（rolling deployments）或金丝雀部署（canary deployments），Ansible 对这两种方式都支持。流水线式部署也是很常见的，常见的工具包括 ThoughtWorks Go、Atlassian Bamboo、大量插件支持的 Jenkins 条，都是比较优秀的。

4. 流程编排

流程编排主要是进行部署时候如何保证基础架构中的各种组件协调一致。例如，在你对 Web 服务器部署新的软件版本时候，需要确保该 Web 服务器从负载均衡器上移出，这是很常见的场景。这类工具有 Ansible、Mcollective、Salt、Serf、Chef 等。

5. 监控告警

监控告警工具已经发展到能够适应快速处理大规模服务器的环境。以前有成熟的 Nagios、Ganglia、Zenoss、Zabbix，最新发展的有 Graphite、Sensu、Riemann 等，都是相对不错的工具。

6. 日志记录

集中日志数据确保能够正确地收集跨系统和应用的日志，同时能够按照规则进行智能过滤、根本原因分析、告警等。常见的工具有 Logstash-Kibana、SumoLogic、Rsyslog 等。

在上面关键的六个领域中，Ansible 能够非常完美地完成前面四个领域的工作。通

过使用 Ansible，无论是系统管理员、运维团队、基础架构管理员、开发者，或者其他任何需要基础架构自动化者都可以从中受益。本书目的就是介绍如何构建健壮的 IT 基础架构自动化运维系统。

Ansible 软件创始人：Michael DeHaan

2012 年 2 月，曾在 Red Hat 开发 Cobbler 和 Func、又在 Puppet 工作过的 Michael DeHaan 看到了 IT 自动化领域的机会：Linux 管理员不得不用好几类工具来应付不同的工作场景，如配置管理用 Puppet 或 Chef，部署时要用 Fabric 或 Capistrano，还要用 Func 或 mCollective 处理其他任务，总之，太复杂了。同时，多节点部署却没有处理得很好的工具，而在云计算和大规模互联网的基础设施里，这恰恰是最有意思的问题。

一天，DeHaan 在自己的沙发上开始用 Python 开发一个新工具，他的目标是：极为易用，连他自己都很想用；任何人可以在几分钟之内学会并使用。经过短短的 6 个月，第一个版本的工具诞生了，这就是 Ansible。

由于 DeHaan 在运维圈已经很有名气，Ansible 发布后很快流行起来。这期间，Fedora 的 Seth Vidal(yum 作者）采用并在 4 月份发表了 High Scalability，都非常关键。

这之后，DeHaan 还参与了 OpenStack 的开发，但在用 Puppet 自动化管理 OpenStack 的过程中不断撞墙。这时候，Ansible 在 GitHub 上火了起来。很快他决定成立公司——AnsibleWorks。2013 年 8 月公司获得 600 万投资，后来改名为 Ansible 公司。

Ansible 只依赖 SSH，无需在远程机器上安装代理，极为容易上手。Hacker News 上有人称之为 shell scripting++，很到位。

1.1.2　Ansible 软件发布

Ansible 公司负责 Ansible 开源软件的维护、管理，是 Ansible 软件发展的最大贡献者。Ansible 开发团队非常高效，软件发布周期大约是 2 个月发布一个新版本。由于发布周期如此之短，轻微的 bug 通常是在下一个版本中得到修补，而不是对稳定版本发布新补丁。重大的 bug 经评估后，如果确实需要将会发布对稳定版本的补丁，但这种情况很少出现。

Ansible 项目重要目标之一是向后兼容，因此这些实例不用修改也能在将来的版本中很好地运行。Ansible 每个主要版本代号都是 Van Halen 乐队的一首曲子。在编写本书时，最新稳定版是 2015 年 7 月 26 日发布的 1.9.2 版本，代号为“Dancing In the Street”。1.9.2 是对之前 1.9.0/1.9.1 的小版本升级，主要修复一些安全、功能方面的 bug。

编写本书时，2.0 版本已经在开发中，但还没最后确定内容和发布时间，主要将改进以下内容。

1. 变更功能

- 引入新的 block/rescue/always 指令符，允许执行任务块和异常处理的语义。
- 扩展新的 plugin 策略，可以在每个 play 中控制任务执行的过程，默认时与以前一样。
- 改进异常处理，现在你将可以得到更多的详细解析信息，将会更新一般的异常处理和显示。
- 任务中 include 将在执行者进行评判，最终的结果与之前一样，但现在可以包含动态的 include 和选项。
- 新版本中将可以使用更多动态的 include 的重要特性以“with_”开头的循环。
- Callback、connection、lookup 的插件 API 略有变化，新版中有的需要修改才能工作。
- Callbacks 现在与活动目录一起，不需要复制，只需要添加到 ansible.cfg 的白名单中。
- 许多 API 已经变更，虽然现在运行的不能直接在新版本中运行，但新的 API 更容易使用、测试。
- 设置将更有继承性，你在 play、block 或 role 中设置的参数将由容器自动继承，这样将可以在所有层级上设置，以前需要手工处理这些代码。
- 模板代码现在保持 bool 或数字类型，而不是转换成字符串，如果你要用以前的方式，只要把值用引号引起来，将会是按照字符串方式处理；如果是 null 的情况，输出的结果将会是一个空字符串。
- 增加了 refresh_inventory 元语句，强制在 play 中重新读取资源清单。
- vars 现在可以在 play、block、role 和 task 不同级别中设置。
- 在 yaml 中设置的空变量或 null 变量将不再转换成空字符串，它们将保留 None 的值；如果要保留以前的处理方式，只需要在配置文件中设置 null_representation 或环境变量 ANSIBLE_NULL_REPRESENTATION 中进行设置。

2. 改进模块

对少量模块进行更新，包括 ec2_ami_search (ec2_ami_find)、quantum_network (os_network)、glance_image、nova_compute (os_server)、quantum_floating_ip (os_floating_ip)，括号内是对应的新版本模块。

3. 新增

新增多达 79 个支持模块，主要是增强对云计算环境的支持，特别是对 Amazon、CloudStack、OpenStack、VmWare 等主流云平台的支持，还有些是增强对应用 WebFaction、Win_IIS、Zabbix 等的支持。

Ansible 名称的来历

最早是厄休拉·勒古恩（Ursula K. Le Guin）在 1966 年的小说《罗卡农的星球》（Rocannon's World）中创造了 Ansible 这个词，用以表示一种能在浩瀚宇宙中即时通信的装置。这在她后来的作品中也得到了沿用，并很快传播到了其他科幻作家的作品之中，Ansible 往往作为一种速度比光速还快的虚构通信工具。最出名的案例也许是奥森·斯科特·卡德（Orson Scott Card）的《安德的游戏》（Ender's Game），其中地球两次遭遇过虫族的进攻，国际舰队认为必须在世界各地寻找天资聪颖的孩童，将把他们塑造成舰队指挥官，使人类在与虫族的战斗中占领先机，并得到存活的希望。安德使用 Ansible 装置实时远程指挥前线的舰队，这是相距数光年作战的唯一可行方式。

Michael DeHaan 想用这个词来比喻控制远端大量的服务器。

那么，勒古恩又是怎么想到这个单词的呢？在 2001 年 Usenet 的一个帖子里，戴夫·古德曼宣称勒古恩曾经告诉他“Ansible”是从“answerable”（可以应答）演变来的，她后来发现把这个词的字母换一下顺序就变成了“lesbian”（女同性恋者），这一点也使她觉得相当有趣。

4. 扩展

扩展了新的资源清单，支持 CloudStack、Fleetctl、Openvz、Proxmox、Serf 等环境。

详见 Ansible 官方信息发布网站 https://github.com/ansible/ansible/blob/devel/

CHANGELOG.md 说明，或 http://www.ansible.cn/forum.php?mod=viewthread&tid=334（部分已翻译）。

1.1.3 Ansible 公司服务

Ansible 既可以是指软件开源的名称，也可以指运营开源项目的公司名称。Michael DeHaan 是 Ansible 软件的发起者、创始人，Ansible 公司前技术总监（CTO）。为了避免混淆，我们在后面涉及 Ansible 都指 Ansible 软件，指公司的时候都用 Ansible 公司表示。

Ansible 公司是负责 Ansible 软件开发、开源社区的管理，确保 Ansible 软件工具适应 IT 自动化的需求。同时 Ansible 公司在开源 Ansible 软件基础之上，开发了基于 Web 界面友好的 Ansible Tower 专有 IT 自动化管理工具。提供 Ansible 软件推广、咨询、培训服务，为支持 Ansible 发展的可持续提供保证。Ansible 的服务团队成员都是经过精心挑选、久经沙场、经验丰富的 IT 专业人士，包括系统管理员、软件开发者、咨询服务精英、自动化运维专家。无论是公共的还是专有的领域，他们在各种规模系统上都取得过成功。团队的所有成员都是高级的 Ansible 用户，都为开源产品贡献过代码，是社区的活跃成员。他们提供如下一些服务。

1. Ansible 健康检查服务

如果已经使用 Ansible 有段时间了，但你想要确保符合最佳实践标准；如果你的上线日期已经很近，想要在上线截至日期之前一切准备就绪，都可以寻求“健康检查服务”。服务团队将评估你 Ansible 和 Ansible Tower 管理的内容和配置，用 Ansible 最佳实践来验证你的系统，并为你将来的项目提供建议。有如下服务：

- 审查你当前的 Ansible 配置是否遵循最佳实践。
- 建议改进你 Ansible 的实现。
- 对在审查过程中发现的问题将按照目标进行培训。

2. 协助重大迁移服务

发现 Ansible 是很易用的产品，但现在已经部署了 Puppet、Chef、CFEngine 或其他定制化的解决方案。要从现在的配置管理、部署方案、创建 Ansible 与之对应，这需要做重大迁移服务工作，这项工作包括：

- 评估：与你原来使用工具的专家一起分析当前工具管理的内容，设计一个迁移计划。

- 实现：根据评估的结果，按照推荐的方式进行实施，用 Ansible 工具创建相应的管理功能。

根据客户的需求，Anisble 咨询团队将交付部分或全部迁移服务，使运行在 Ansible 维护工具中。

Ansible 团队还将提供一份文档化的解决方案，一种可行的迁移方式，探寻“ train the trainer”方式，让你的团队将来能够自己可以进行迁移。

3. 推进 IT 自动化项目

推进自动服务是一项能够加快建立起自动化很好方法，这些协议将提供：

- 为你的团队培训 Ansible、Ansible Tower 的基础使用和高级知识。
- 安装、配置 Ansible 和 Ansible Tower。
- Ansible 导师将在你的流程中部署关键组件。

4. 客户咨询服务

Ansible 公司可以提供个性化客户服务，包括从项目启动、健康检查、重要迁移到最后完成设计目标过程中大大小小所有事情。

可能交付的内容包括：定制化模块、Jinja2 模板、动态资产库脚本、角色和 playbook，能够帮助你解决挑战的问题，加速任何 Ansible 项目的推进。

5. Ansible 培训服务

Ansible 官方网站已经提供了快速入门的视频短篇教程，帮助你了解 Ansible。这个短篇介绍了 Ansible 作为一款强大的自动化 IT 运维、流程编排的工具，有哪些优势，这也是为你开始自动化项目提供必备的基础知识。

Ansible 提供老师讲解的课程，包括动手学习环境、实际问题处理、现场建设性问题解答。也致力于根据你的个性需求开发特定裁剪的课程，个性化的培训也可以根据要达到的目标签订专业的服务协议。

1.2　Ansible 架构模式

Ansible 维护模式通常由控制机和被管机组成。控制机是用来安装 Ansible 工具软

件、执行维护指令的服务器或工作站，是 Ansible 维护的核心。被管机是运行业务服务的服务器，由控制机通过 SSH 来进行管理。

1.2.1 Ansible 管理方式

Ansible 是一个模型驱动的配置管理器，支持多节点发布、远程任务执行。默认使用 SSH 进行远程连接。无需在被管节点上安装附加软件，可使用各种编程语言进行扩展。

Ansible 总体系统构成如图 1-2 所示。Ansible 管理系统由控制主机和一组被管节点组成。控制主机直接通过 SSH 控制被管节点，被管节点通过 Ansible 的资源清单（inventory）来进行分组管理。

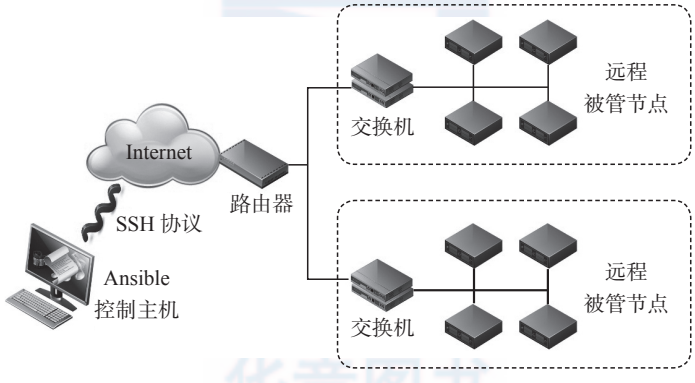

图 1-2 Ansible 总体系统构成

Ansible 如何进行配置管理呢？图 1-3 给出了实例，这是 Ansible 用剧本（playbook）方式对 3 台运行 Nginx 服务的 Ubuntu 服务器进行配置管理。她编写一个叫 webservers.yml 的 Ansible 脚本。在 Ansible 中，这个脚本称为 playbook。在 playbook 中描述了包含被管节点的 hosts 和对这些 hosts 按照顺序执行的任务列表（task）。

在此例子中，hosts 包括 web1、web2、web3，任务列表包括如下过程：

- 安装 Nginx（Install Nginx）。
- 创建 Nginx 配置文件（/etc/nginx/nginx.conf）。
- 基于安全证书 SSH 方式拷贝配置文件，重启 Nginx 服务。
- 确保 Nginx 服务处于启动状态。

然后在 Ansible 系统的控制主机上执行：

```
$ ansible-playbook webservers.yml
```

Ansible 将会通过 SSH 连接并行地在 web1、web2、web3 上面安装、配置、运行 Nginx 服务。

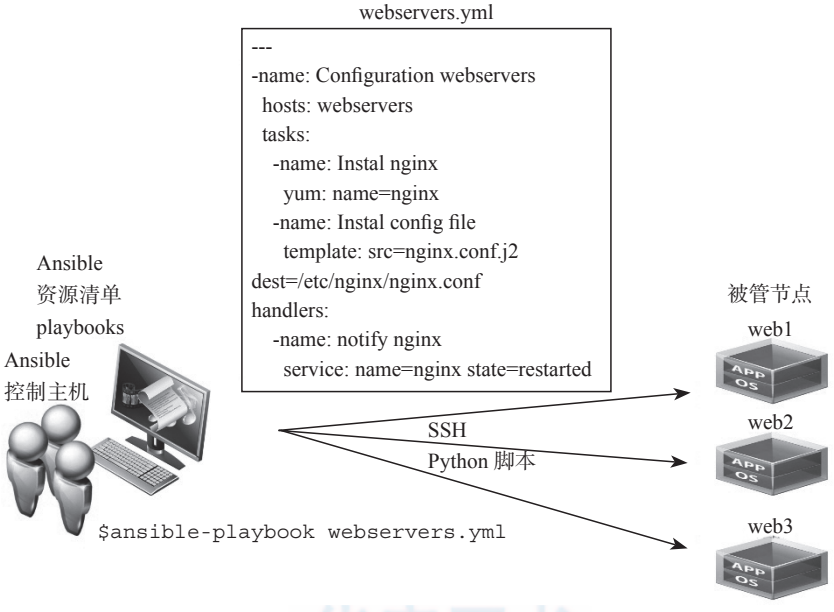

图 1-3　运行 Ansible 的 playbook 对 3 台 Web 服务器进行配置

本书后面的章节将会对这里涉及如何编写 playbook，以及 webservers.yml 中各个字段的含义、工作内容做深入的讲解。

1.2.2　Ansible 系统架构

Ansible 集合了众多优秀运维工具（Puppet、Cfengine、Chef、Func、Fabric）的优点，实现了批量系统配置、批量程序部署、批量运行命令等功能。Ansible 是基于模块工作的，本身没有批量部署的能力。真正具有批量部署的是 Ansible 所运行的模块，Ansible 只是提供一种框架。Ansible 的基本架构见图 1-4，用户通过 Ansible 编排引擎操作公有云 / 私有云或 CMDB（配置管理数据库）中的主机。

从图 1-4 可以看到，Ansible 由以下各部分组成：

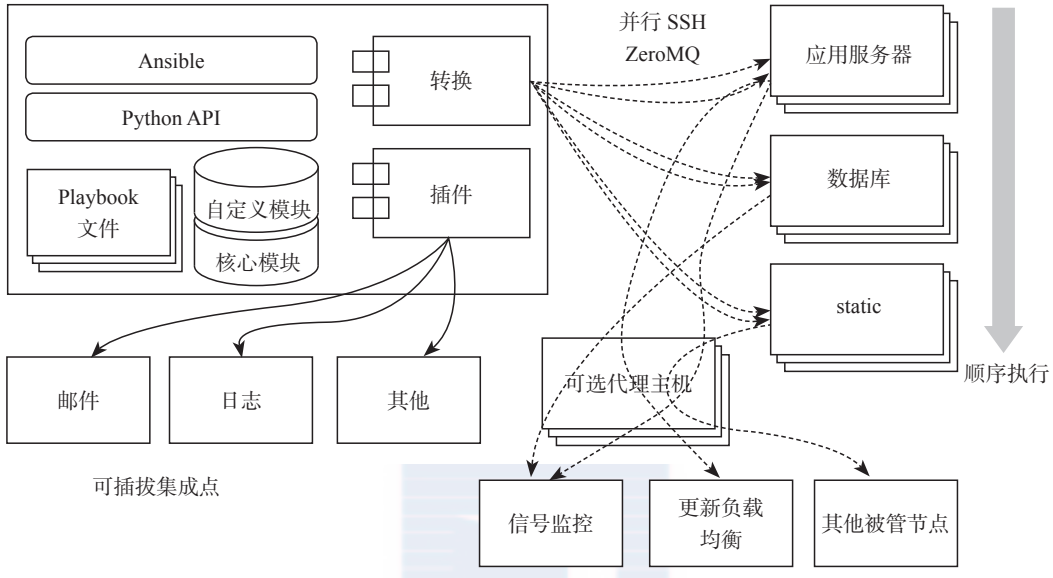

图 1-4 Ansible 基本架构

- 核心引擎：即 Ansible。
- 核心模块（core modules）：这些都是 Ansible 自带的模块，Ansible 模块资源分发到远程节点使其执行特定任务或匹配一个特定的状态。Ansible 遵循“batteries included“哲学，所以你可以有各种各样任务的核心模块。这意味着模块是最新的，你不需要寻找一个在你平台上工作的实现。你可能认为模块库的工具箱里充满了有用的系统管理工具，和 playbook 一起构建自动化运维系统的基础源材料。
- 自定义模块（custom modules）：如果核心模块不足以完成某种功能，可以添加自定义模块。
- 插件（plugins）：完成模块功能的补充，借助于插件完成记录日志、邮件等功能。
- 剧本（playbook）：定义 Ansible 任务的配置文件，可以将多个任务定义在一个剧本中，由 Ansible 自动执行，剧本执行支持多个任务，可以由控制主机运行多个任务，同时对多台远程主机进行管理。
- playbook 是 Ansible 的配置、部署和编排语言，可以描述一个你想要的远程系统执行策略，或一组步骤的一般过程。如果 Ansible 模块作为你的工作室工具，playbook 是你的设计方案。在基本层面上，剧本可用于管理配置和部署远程机器。在更高级的应用中，可以序列多层应用及滚动更新，并可以把动作委托给其他主机，与监控服务器和负载平衡器交互。

- 连接插件（connectior plugins）：Ansible 基于连接插件连接到各个主机上，负责和被管节点实现通信。虽然 Ansible 是使用 SSH 连接到各被管节点，但它还支持其他的连接方法，所以需要有连接插件。
- 主机清单（host inventory）：定义 Ansible 管理的主机策略，默认是在 Ansible 的 hosts 配置文件中定义被管节点，同时也支持自定义动态主机清单和指定配置文件的位置。

Ansible 采用 paramiko 协议库（Fabric 也使用这个），通过 SSH 或者 ZeroMQ 等连接主机。Ansible 在控制主机将 Ansible 模块通过 SSH 协议（或者 Kerberos、LDAP）推送到被管节点执行，执行完之后自动删除，可以使用 SVN 等来管理自定义模块及编排。从图 1-4 可以了解到，控制主机与被管节点之间支持 local、SSH、ZeroMQ 三种连接方式，默认使用基于 SSH 的连接。在规模较大的情况下使用 ZeroMQ 连接方式会明显改善执行速度。

Ansible-galaxy 分享平台

Ansible 提供了一个在线 playbook 分享平台，地址：https://galaxy.ansible.com/，该平台汇聚了各类常用功能的角色（Role），当前分为 web、system、packaging、networking、monitoring、development、database:sql、database:nosql、database、clustering、cloud:rax、cloud:gce、cloud:ec2、cloud 共 14 类，已经超过 4000 个功能角色。

根据需要找到适合你的角色后，只需要使用命令“ansible-galaxy install+ 作者 ID. 角色包名称”就可以安装到本地，比如想安装 debops 提供的 Nginx 安装与配置的角色，直接运行“ansible-galaxy install debops.nginx”即可安装到本地，该角色的详细地址为：https://galaxy.ansible.com/list#/roles/1580。

1.2.3　任务执行模式

Ansible 系统由控制主机对被管节点的操作方式可分为两类，即 ad-hoc 和 playbook：

- ad-hoc 模式使用单个模块，支持批量执行单条命令。
- playbook 模式是 Ansible 主要管理方式，也是 Ansible 功能强大的关键所在。playbook 通过多个 task 集合完成一类功能，如 Web 服务的安装部署、数据库服务器的批量备份等。可以简单地把 playbook 理解为通过组合多条 ad-hoc 操作的配置文件。

在 Ansible 系统内部又是如何执行的呢？ Ansible 执行过程如图 1-5 所示。

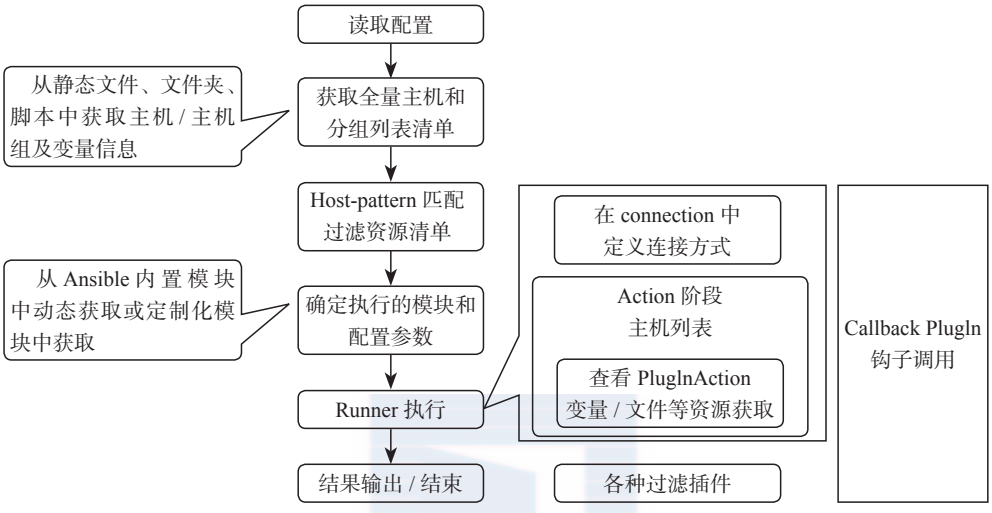

图 1-5 Ansible 执行过程

1.3 Ansible 特性

Ansible 是基于一致性、安全性、高可靠性设计的轻量级自动化工具，具有功能强大、部署便捷、描述清晰等特性。对于管理员、开发者、IT 经理等都容易上手，学习曲线较低，能够快速理解、掌握 Ansible 的自动化体系，满足不同技术级别的用户需求。

同时 Ansible 是一款满足当代大规模、复杂环境的 IT 基础架构自动化管理的工具。Ansible 相对与其他自动化解决方案，在核心能力上效率更高。也很好地解决了统一配置、统一部署、流程编排等复杂的 IT 自动化管理问题。下面就介绍其功能特性以及与其他工具的对比。

1.3.1 Ansible 功能特性

从功能上看，Ansible 可以实现以下目标：

- 应用代码自动化部署。
- 系统管理配置自动化。
- 支持持续交付自动化。
- 支持云计算、大数据平台（如 AWS、OpenStack、CloudStack、VMWare 等）环境。

- 轻量级，无需在客户端安装 agent，更新时只需在控制机上进行一次更新即可。
- 批量任务执行可以写成脚本，不用分发到远程就可以执行。
- 使用 Python 编写，维护更简单，Ruby 语法过于复杂。
- 支持非 root 用户管理操作，支持 sudo。

实现上述目标是很有挑战性的工作，需要满足健壮性、易于管理的架构，这在应用维度一直没能很好地解决，因为管理工具不能对被管环境施加额外的影响，而 Ansible 很好地解决了这些问题。Ansible 作为优秀管理工具具备以下一些特点。

1. 语法简单、易读

Ansible 的配置管理脚本 playbook 语法基于 YAML，它是一种可读性高、用来表达数据序列的格式标准。YAML 参考了 XML、C、Python、Perl 以及电子邮件格式 RFC2822 等其他多种语言，可读性好、交互性强，使用了实现语言的数据类型，有一个一致的信息模型，可以基于流方式来处理，具有表达能力强、扩展性好等特点。

可以把 playbook 称为“可执行的文档”，就像 README 文件一样描述你需要部署软件的指令，由于这些代码是可以直接执行的，这些指令永远不会过时。

2. 不需要在被管节点安装客户端软件

Ansible 项目管理系统的核心是通过 SSH 来连接被管节点，可以使用 paramiko（一个 Python 库）或使用原生的 OpenSSH（带参数 -c ssh）。当使用 SSH 客户端连接时候，默认的 OpenSSH 连接是可以重用的。无论哪种方式，SSH 都是作为一种传输方式，不是作为一种执行的 Shell。

模块是包含一些参数的 Ansible 小程序，通过 SCP 或 SFTP 传送到远端被管机器的临时目录中进行执行，然后再删除。这些模块通过标准的输出方式返回 JSON 格式的数据，这些返回的数据通过控制主机上的 Ansible 程序进行处理。

这种方式能管理非常多远程活动，并且只需要很小的交互流量。模块按照“幂等资源”的方式进行管理，也就是发起多次执行也不会给服务带来负面响应。它不是简单的命令或脚本，例如一个模块可以确定应该安装某个特定版本的软件包，但如果这个系统已经有合适的工作状态，就不会再执行任何命令。这种方式有如下优点：

- 提高网络安全：由于不需要在远端被管节点上运行 agent，因此 Ansible 遭受攻

击的机会很少。只需要运行 OpenSSH 一个后台进程，它在全球范围内是最严格审查的程序之一。Ansible 知道做好加密是一项极其艰难的事情，因此没有考虑使用自己的后台进程和认证方式，而是依赖可用的最安全的远程管理系统。OpenSSH 极其广泛地在各种系统中使用，有时非常轻量级。一旦 OpenSSH 出现安全问题，将会非常快速地推出补丁修复。当我们认为可能编写一个安全的 OpenSSL 实现时，我们也一直跟踪远程拓展在这一应用领域的类似工具，期望尽量避免出现类似问题。

- 可以使用非 root 用户访问（可用 sudo）：Ansible 的 playbook 能够用任何用户账号登录远程系统，可以用初始连接的用户来运行程序，也可以使用 sudo 切换到其他用户（包括 root）。如果需要可以直接用 root 登录，sudo 可以需要口令，也可以不需要口令，当管理的一些系统根本不能使用 root 登录的时候，或者不允许 root 登录但可以通过 sudo 切换到 root 的，这些方式是非常合适的。这里有个例子，一个没有超级权限的用户能够用 Ansible 管理自己 home 目录下的内容，即使在这个系统上没有 root 或 sudo 权限。甚至当使用 sudo 时，文件传输是没有限制的。Ansible 也包含一个兼容 sudo 的文件传输工具，内容是按照正常的 SFTP 方式传输的，Ansible 会把文件复制到授权的位置。通过比较源文件和目标文件的校验和方式，Ansible 能够智能地识别文件是否需要传输。
- 限制传输潜在敏感数据：Ansible 总是尽量把最少的数据传输到被管节点，由于控制服务器是逻辑控制中心，只有远端被管节点必要的变量才送给它。例如，有一个全局变量叫 foo，只有在远端被管节点中明确在资源或模板中使用它，否则它不会发送到该远端节点。这种 Ansible 的推送方式，只给远端节点需要看到的最少数据。类似地，Ansible 没有包含客户文件服务器的实现，只通过 SFTP、SCP、Rsync（基于 SSH）传输文件，并且只有 playbook 需要的文件才传输。结果是被管节点请求的文件或模板可以提供，但敏感数据不会提供给。也就是不能给一台远程主机查看将要提供给其他被管节点的数据。这使得 Ansible 适用于带有敏感数据的环境，包括社会科学工作、健康医疗、政府政务等应用。
- （Credential Segregation 凭证隔离）：Ansible 适用于不同用户有不同安全级别的环境。可以定义管理主机的通用变量，然后使用自己的凭证访问远端节点。例如，允许研发工程师管理开发的服务器，QA 工程师管理 QA 的服务器，系统管理员管理生产环境服务器，不存在研发工程师把代码推送到生产环境的风险。
- 去中心化：由于 Ansible 的 playbook 需要用户的凭证来执行，不存在中心点，只

是通过访问配置内容、管理软件链条，不需要访问 SSH 密钥，就可以接管建立“僵死网络”，避免使它成为攻击的目标。用户可以加密自己的密钥，不需与任何人使用相同密码。在可选的管理形式中，访问授权的软件资源都有可能导致部署系统自动化。要使用 Ansible，用户需要两个条件：能够访问到资源库，具有管理远端节点的凭证。当使用锁定的密钥（甚至是密码）时，不存在暴露安全漏洞中心被攻击点，也不允许从远端硬件来访问管理服务器。

- 没有“管理的管理程序”的问题：许多配置管理解决方案的主要问题之一是“管理的管理”，为了开始管理服务器必须先在远端节点上安装软件。当更新管理软件时候，通常各种 agent 需要先更新（许多系统是无法自己自动更新的）。有时会产生管理服务器与 agent 软件版本兼容性问题，或 agent 与运行语言版本问题，Ansible 通过基于 SSH 传输模块避免了这类问题。SSH 二进制代码已经是 OS 的一部分了，并且是每个主流操作系统的核心。不需要任何 agent，也就不存在 agent 以任何方式崩溃的问题，因此服务器存在的安全风险是非常小的。
- 管理服务器的可扩展性：由于 Ansible 使用推送方式来管理远端节点（当然，也很容易配置成远端节点查询更新的方式），因此 Ansible 对于“羊群效应”的管理问题是有免疫能力的。在其他的一些解决方案中，管理 agent 周期性检查会对管理服务造成破坏，经常会超过管理服务器的运行负荷，导致需要对管理服务器进行横向或纵向扩展。频繁管理服务器需要为远端节点消耗昂贵的计算资源。为了解决这个问题，Ansible 通过推送的方式，通过配置每次推送节点数量来做限制。均衡了远程节点需要的管理服务器的负载能力，可以让计算系统负载更加均衡，即使个人笔记本电脑也可以作为 Ansible 系统的控制服务器。
- 资源的利用：在 Ansible 不管理远端节点时，这些远端节点什么也不做，也就是没有后台进程在消耗 CPU、内存。曾经报道过，某些应用服务器在唤醒配置窗口时会产生明显的降低性能，某些 agent 可能会占用超过 400MB 的内存。在虚拟化环境中，这些资源的消耗可能会快速叠加，就会增加对硬件的支出。Ansible 能最大限度让你的关键性能负荷使用所有的计算资源，你也可以选用间隔方式运行管理程序，也不存在由于内存泄漏或 agent 软件崩溃导致无法管理远端节点的问题。
- 防火墙的友好性和确定性：不像某些基于消息总线的系统，Ansible 不需要长时间在控制节点与被管主机之间建立连接。在生产环境，这种长连接可能受限于防火墙，防火墙一般不喜欢长连接，Ansible 很好地规避了这个问题。当管理服务与应用之间的连接被断开又没法重置的时候，就只能等到重启 agent 端服务了，

Ansible 没有这个问题。没有严格意义上的无代理系统，Ansible 也避免在这些架构中的其他问题，获得确定性的响应。并不是只有出现响应节点才管理，对于你要管理但无法达到的就静默了。如果一个节点宕机，运行 Ansible 时你就会知道，将会返回一个失效信息，这个你可能忽略或修复。这对于你给所有节点或部分子集部署软件更新，是很重要的信息，知道哪些节点无法连接是非常重要的，而对“发布 – 订阅”架构通常是没有提供这些信息的。

3. 基于推送（Push）方式

有些配置管理系统（如 Chef、Puppet、Saltstack 等）是使用代理模式的，它们默认基于拉（pull）方式。被管节点上安装代理后，代理服务将周期性检查中心控制主机的服务，并从控制主机上取来配置信息。被管节点的变更过程大致如下：

1）修改配置管理脚本。

2）把修改好的管理配置脚本推送到中心控制主机上。

3）被管节点的代理服务周期性触发检查。

4）被管节点连接配置管理的控制主机。

5）被管节点下载新的配置管理脚本。

6）被管节点在本地执行配置管理脚本，被管节点状态相应地变化。

而 Ansible 默认基于推送（push）方式，管理过程如下：

1）增加或修改 playbook。

2）执行新的 playbook。

3）控制节点的 Ansible 连接被管节点并执行修改被管节点状态的模块。

一旦你执行 ansible-playbook 命令，Ansible 就会连接到远程的被管节点并按照脚本要求执行。

基于推送方式有很大的优势，你能控制什么时候让远程被管节点发生变更，不需要等到被管节点上周期性的时间。拉方式的支持者声称拉方式具有管理服务器规模数

量的优势，新被管节点随时可以在线添加。但是 Ansible 已经在管理超过上万台节点规模的系统，很方便地动态增减或删除被管节点。

当然，如果你一定要使用拉方式，Ansible 官方也已经支持这种拉方式，可以使用 ansible-pull 这个工具。

4. 方便管理小规模场景

Ansible 能够管理成千上万台主机，但如何管理缩小规模的场景呢？其实，用 Ansible 也很容易配置小规模集群甚至单台主机，你只要简单编写一个 playbook 即可。Ansible 遵循艾伦·凯（Alan Kay）的名言：“简单的东西应该简单，复杂的东西才有可能成功（Simple things should be simple，complex things should be possible）”。

5. 大量内置模块

使用 Ansible 能够在被管节点上执行任何 shell 命令，但是 Ansible 真正的威力在于内置大量的模块。使用这些模块可以完成如软件包安装、重启服务、拷贝配置文件等操作。Ansible 模块是声明式的（*declarative*），你只需使用这些模块描述被管节点期望达到的状态。例如，你可以用 user 模块对用户授权，确保系统有一个 deploy 用户，并且属于 web 组，代码如下：

```
user: name=deploy group=web
```

Ansible 内置模块都是等幂性的（idempotent）。这就是如果系统中没有用户 deploy，Ansible 将会创建一个 deploy 账号；如果这个账号已经存在，则 Ansible 什么也不做。等幂性对于自动化维护是非常重要的特性，这样在被管节点上多次执行 Ansible 的 playbook 也能达到同样效果。相对于直接执行操作系统脚本的方式是巨大的改进，操作系统脚本再执行一次可能就会产生不同的、非预期的结果。

什么是收敛性

配置管理的资料中经常提到“收敛”这个概念，在配置管理中收敛最切确的说法是在 Mark Burgess 以及他的 CFEngine 配置管理中所写的。

如果一个配置管理系统是收敛的，那么这个系统可以多次运行，都达到最终期望的状态，每次执行只是更加接近而已。

这个收敛的思想在 Ansible 中不完全适用，Ansible 没有在被管节点上多次运行的概念，Ansible 在模块中实现了这种方式，一次执行 playbook 就会确保每台被管节点都能达到期望的状态。

“等幂性”和“收敛性”是有差别的，而且有非常重大的差异。等幂性是指如果系统已经处于期望的状态，则对系统什么也不操作。而收敛性是指系统不需要变更或操作的特性。最终的结果可能一样，但 CFEngine 相对于等幂性更强调收敛性，并且在自动化系统中内嵌了大量的逻辑，避免执行不必要的操作。

6. 非常轻量级的抽象层

有些配置管理工具通过一个抽象层，可以把相同的脚本在不同的操作系统上管理。例如，安装软件包有 yum、apt 工具，对于配置管理工具就可能抽象成 package。

Ansible 不同于这种方式，你需要使用 apt 模块来安装基于 apt 软件管理的系统，用 yum 模块来安装基于 yum 软件管理的系统。

这听起来也许不太方便，在实践中我们发现这使得 Ansible 更方便使用。Ansible 不需要再学新知识来屏蔽不同操作系统的差异性、新的抽象环境。这相对减少了 Ansible 要求的知识领域。这样，你不需要了解太多就可以编写 playbook 了。

如果你确实需要，你也可以在 Ansible 的 playbook 中对不同被管节点的操作系统编写不同的动作脚本。但不太建议这么做，建议编写针对特定操作系统的 playbook，如 CentOS、Ubuntu。

Ansible 社区中最主要的重用是模块，模块都是比较小的，与特定操作系统相关，具有良好的定义，容易实现，方便共享。Ansible 项目是非常开放的，经常接纳社区贡献的模块代码。

Ansible 的 playbook 不是真正地在不同上下文之间能够重用，本书后面讲到的 roles 方式整合 playbook 是更好的重用方式，大量在线的 roles 资源库收集在 Ansible Galaxy 中。现在已经超过 4000 个。

工作实践中，每个单位配置的服务器有所差异，应最好为自己的单位编写 playbook，而不是尝试重用通用的 playbook。查看其他人的 playbook 主要是学习人家如何做到的。

目标是提供面对广泛的自动化挑战如何获得大型生产力的优势。当 Ansible 提供更强大的生产力逐步替代其他许多核心性能的自动化解决方案时，它也在寻求解决其他还没解决的 IT 挑战，如何将复杂多层级工作流程清晰化、清楚统一地配置 OS、在单一框架下进行应用软件的部署。

1.3.2　Ansible 与其他配置管理的对比

当前几款与 Ansible 功能类似的主流配置管理软件有 Chef、Puppet、SaltStack 等，这里只对各个软件的技术特性做个简单对比，其中不针对各个软件的性能作比较。具体内容见表 1-2。

表 1-2　Ansible、Puppet、SaltStack 技术特性比较

项目	Puppet	SaltStack	Ansible
开发语言	Ruby	Python	Python
是否有客户端	有	有	无
是否支持二次开发	不支持	支持	支持
服务器与远程机器是否相互验证	是	是	是
服务器与远程机器通信是否加密	是，标准 SSL 协议	是，使用 AES 加密	是，使用 OpenSSH
平台支持	支　持 AIX、BSD、HP-UX、Linux、Mac OS X、Solaris、Windows	支持 BSD、Linux、Mac OS X、Solaris、Windows	支持 AIX、BSD、HP-UX、Linux、Mac OS X、Solaris
是否提供 Web UI	提供	提供	提供，不过是商业版本
配置文件格式	Ruby 语法格式	YAML	YAML
命令行执行	不支持，但可通过配置模块实现	支持	支持

与其他的自动化工具相比较，Ansible 不需要安装管理客户端就能轻松地管理、配置你的基础架构。它们各自的优缺点见表 1-3。

表 1-3　Ansible、Puppet、Chef、SaltStack 的优缺点

产品	优势	劣势	成本
Ansible	● 模块可以用任何语言开发 ● 被管节点不需要安装代理软件 ● 有 Web 管理界面，可以配置用户、组、资源清单和执行 playbook ● 安装、运行极其简单	● 对被管节点是 Windows 有管理待加强 ● Web 管理界面是内置 Ansible 的一部分 ● 需要导入资源清单	● Ansible 开源版本是免费的 ● Ansible Tower 小于 10 台时被管节点是免费 ● 超过 10 台之后每年每台需要支付 $100 ～ $250 的支持服务费用

（续）

产品	优势	劣势	成本
Puppet	• 模块由 Ruby 或 Ruby 子集编写 • push 命令能够立即触发变更 • Web 界面生成处理报表、资源清单、实时节点管理 • 在代理运行端进行详细、深入的报告和对节点进行配置	• 需要学习 Puppet 的 DSL 或 Ruby • 安装过程缺少错误检查和产生错误报表	• 开源版本是免费的 • Puppet 企业版需要每年每台花费约 $100
SaltStack	• 状态文件可以用简单的 YAML 配置模板或复杂的 Python/PyDSL 脚本 • 与客户端通信可以基于 SSH 或在被管节点安装代理 • Web 界面可以看到运行的工作、minion 状态、事件日志，可以在客户端执行命令 • 扩展能力极强	• Web 界面相对于竞争产品还不太完整、稳定 • 缺乏生成深度报告的能力	• 开源软件是免费的 • SaltStack 企业版每年每个节点花费约 $150，随着数量增加将有优惠折扣
Chef	• 可充分使用 Ruby 的强大功能 • 集中基于 JSON 的 data bags 在运行时产生变量 • Web 界面可以搜索节点清单，查看活动节点，分配 Cookbooks、roles 和节点	• 需要 Ruby 的编程知识 • 当前缺少 push 功能的命令 • 文档有时不太清晰	• 开源版本是免费的 • 5 台企业版的 Chef 是免费的，20 台以内费用每月是 $120，50 台以内每月是 $300，100 台以内每月是 $600，等等

这些自动化管理软件都具备强大的功能、灵活的系统管理和状态配置，都提供丰富的模板及 API，对云计算平台、大数据都有很好的支持。

Puppet、Chef、SaltStack 设置比较复杂，有代理和服务，有远程被管节点需要长期运行管理进程，而且很多术语和概念都是自己一套。Ansible 相对于其他产品要简单得多，本质上混合了声明式和命令式。不仅适用于大规模的 IT 环境，而且对于一些规模较小的环境（20 ～ 50 台机器）Ansible 也是很实用的。

1.4 Ansible 与 DevOps

本节简要介绍下 DevOps(Development 和 Operations 的组合)。DevOps 是一组过程、方法与系统的统称，用于促进软件开发（应用程序 / 软件工程）、技术运营和质量保障（QA）部门之间的沟通、协作与整合，如图 1-6 所示。

DevOps 概念的引入能对产品交付、测试、功能开发和维护（包括常见的“热补丁”）起到意义深远的影响。在缺乏 DevOps 能力的组织中，开发与运营之间存在着信息“鸿沟”，例如运营人员要求更好的可靠性和安全性，开发人员则希望基础设施响应

更快，而业务人员的需求则是更快地将更多的特性发布给最终用户使用。这种信息鸿沟就是最常出问题的地方。

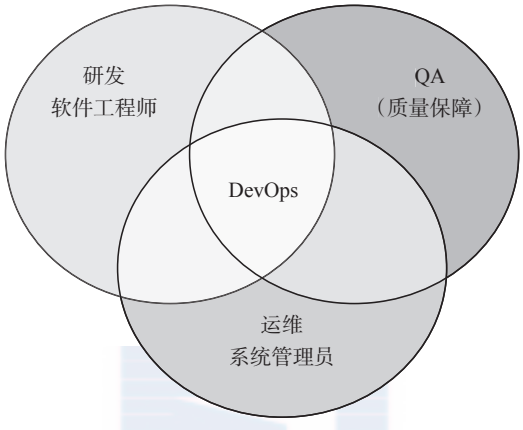

图 1-6　DevOps 的涉及范围

需要频繁交付的单位更需要具有 DevOps 能力，大量互联网在线移动应用随时根据用户的反馈进行迭代开发，持续改进用户体验，这需要能够支撑业务部门每天可能都有几次、几十次部署的需求。这种能力也称为持续部署，并且经常与精益创业方法联系起来。下面几个因素促使引入 DevOps：

- 使用敏捷或其他软件开发过程与方法。
- 业务负责人要求加快产品交付的速度。
- 虚拟化和云计算基础设施日益普遍。
- 数据中心自动化技术和配置管理工具的普及。

DevOps 将使开发团队与运营团队之间更具协作性、更高效的关系。由于团队间协作关系的改善，整个组织的效率因此得到提升，伴随频繁变化而来的生产环境的风险也能得到降低。团队之间相互融合，运维从开发流程的计划阶段就参与进来，运维团队就能了解将要开发的产品，同时可以在让产品开发设计初期就考虑后期运维的需求。

DevOps 成功的关键有如下因素：

- **文化建设**。首先要调动开发和运营部门之间的协作，鼓励运营人员采纳软件开发方法，利用云计算基础设施来完成测试和代码部署。其次在软件开发、测试、质量保障、集成、预生产和生产部署等方面必须打散小团队，更好地整合开发和运

营人员。例如，在讨论运营解决方案或扰乱事后评估报告时应该邀请开发人员加入。相反地，应该邀请运营人员列席开发人员规划会议。让交叉组合的工作模式成为制度，可以让团队之间合作融洽，消除沟通不畅导致的延误或疏忽，使 DevOps 的推进更加有效。有时可以运用岗位轮换或者知识共享的方法。

- **自动化工具支撑**。要超越文化的影响，组织还必须依靠各种 DevOps 工具。例如，开发人员编写代码需要工具，QA 测试人员需要用工具完成新版软件的部署，环境准备、将新代码在测试系统和生产系统之间迁移也必须用到云资源调度工具。

现在已经有了大量的自动化工具，但都是在某些领域或某个领域做得好。运营工具厂商有 BMC、CA 和 XebiaLabs 等公司，软件开发工具厂家有 IBM、Electric Cloud、Serena Software 等公司，工作流程、架构设计和软件发布工具的专业供应商包括 Atlassian、CollabNet、Rally Software、ThoughtWorks、OpenMake 等公司。评估供应商时，除了对基本功能考察之外，还需要考虑这些公司随时可能会被并购，其产品可用性和未来发展也会因此受影响。

尽管企业 IT 环境中的应用各种各样，但其操作系统、基础组件还是有很多共性的。手动安装操作系统的过程通常需要小时级才能交付。并且安装系统的可靠性完全依赖于管理人员的工作经验、技术能力、对复杂过程操作的精确程度。然后管理员需要对这些系统一遍又一遍地重复同样的过程，如图 1-7 所示。

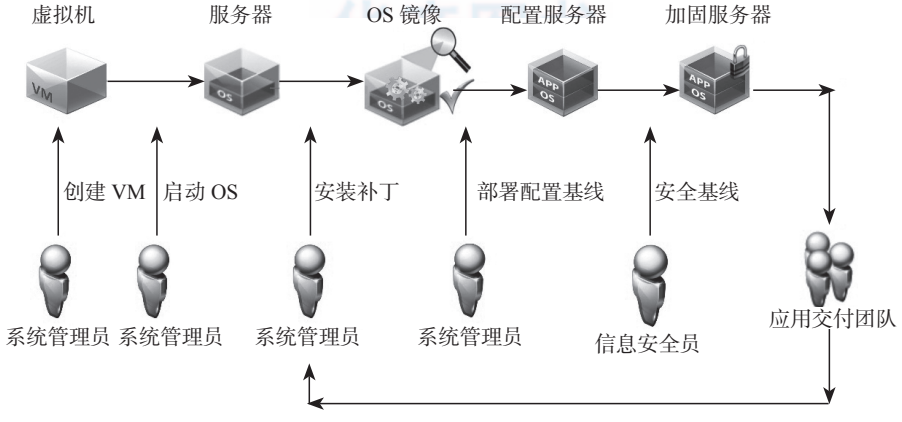

图 1-7　传统工厂运维方式

也可能让其他团队一起参与进来搭建系统，在应用交付团队能够工作之前，信息安全团队需要验证是否达到安全基线。需要接触系统的团队越多，实现的时间就越长，

也就更加可能延时和增加错误方式。

服务器和系统还比较好理解、容易自动化。尽管你当前 OS 搭建过程有很多需要与基础环境交互，自动化搭建和配置过程将按照需要能够重复部署 OS 镜像、合适的工具，管理这些建好的系统将与创建一个新的一样简单，这样这些系统就会总是看起来差不多。

一旦操作系统的搭建和管理自动化之后，把这些自动化推广到其他团队就相对容易了。通常客户搭建一个 OS，如研发、测试、QA 团队运行他们的应用时，能够确保总是运行在配置合适的 OS 之上。

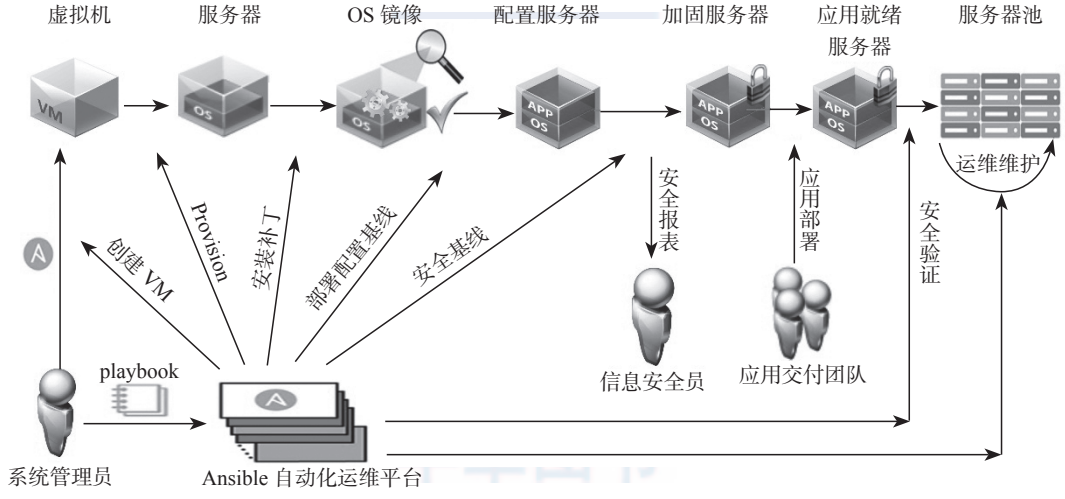

图 1-8　Ansible 自动化运维方式

但是，仅仅搭建系统，就忽略了更困难部分的问题了：如何保持它们整个生命周期中处于更新状态，如何保证当前的基本需求是正确的？这再一次需要采用自动化工具来管理它们之间的差异。

过去传统虚拟机（VM）生命周期管理方法是一个主要问题，需要独立的流程来维护存在的虚拟机，许多交付工具都对在线运行系统的更新、变更缺乏有效手段。

幸运的是，借助服务器自动化搭建和维护过程，可大大减少或完全消除服务器手工交付的过程，使用 Ansible 作为自动化工具，使用相同的 playbook 来搭建系统，就能够保证创建出来的系统是一样的。

甚至在基础架构层应用，也可以自动创建、交付、管理，将节省大量的时间，为单位的扩展团队提供额外的好处。

1.5 本章小结

Ansible 关键的想法是，开发一个好的自动化系统，能认识到计算机是一组而不只是一个个分开的机器，也就是所谓“多层编排”。建模过程与建模状态同样重要。不按传统配置管理依赖定制代理架构的思路，避免了证书交换，以及反向解析 DNS 和 NTP 的问题。默认可插拔，人人都可以很容易地贡献，因此 Ansible 获得了广泛的参与和采用。保持简单（用 YAML 等），制定计划并坚持，然后乐观其成。