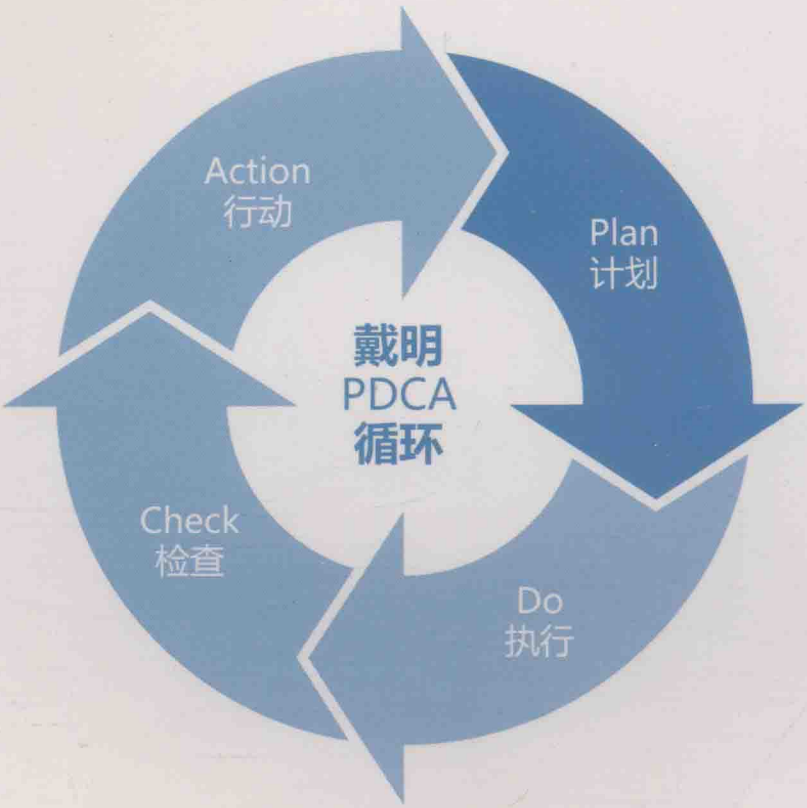
掌握Python自动化运维的热门技术与主流开源工具

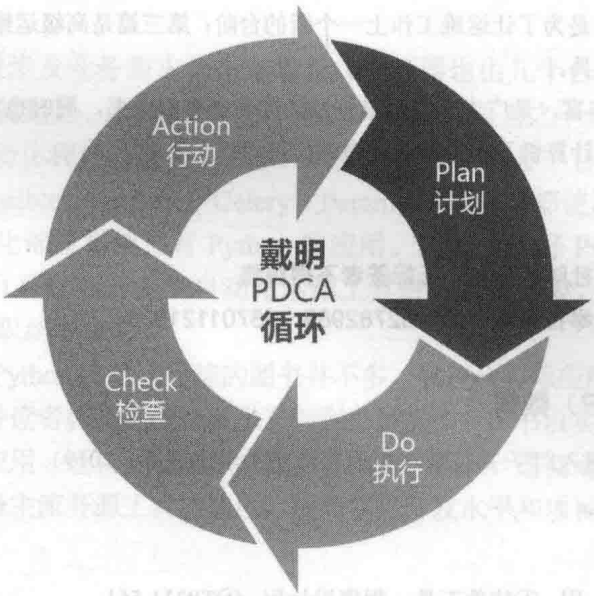
- 以实战为主旨，提供Python运维开发中常见的近百个应用场景
- 详解Python自动化运维量化背景、基础与体系
- 涵盖Ansible、APScheduler、Paramiko、Celery、Airflow等主流运维工具



Python自动化运维快速入门

郑征 著

清華大學出版社



Python自动化运维快速入门

郑征 著

清華大學出版社
北京

内 容 简 介

本书是一本从零开始、手把手教你运维的书籍，通过上百个实际运维场景案例，帮助读者理解并掌握自动化运维。

本书分为三篇共 11 章，第一篇是基础运维，介绍自动化运维、Python 基础、文本处理、日志、FTP 服务器、使用 Python 发邮件、微信等。通过本篇的学习，可以达到编写 Python 程序来解决基础运维问题的水平；第二篇是中级运维，介绍自动化运维工具（Ansible）、作业调度工具（APScheduler、Airflow）、分布式任务队列（Celery），目的是为了让运维工作上一个新的台阶；第三篇是高级运维，介绍 Docker 容器技术，现现已成运维人员必备的工具。

本书内容详尽、示例丰富，是广大从事运维开发的读者必备参考书，同时也非常适合学习 Python 的读者阅读，也可作为高等院校计算机及相关专业作为教材使用。

图书在版编目（CIP）数据

Python 自动化运维快速入门 / 郑征著.—北京：清华大学出版社，2019
ISBN 978-7-302-52580-6

Ⅰ. ①P… Ⅱ. ①郑… Ⅲ. ①软件工具－程序设计Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2019）第 043811 号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张**：17.5　　**字　　数**：448 千字
版　　次：2019 年 4 月第 1 版　　**印　　次**：2019 年 4 月第 1 次印刷
定　　价：59.00 元

产品编号：081950-01

前 言

随着 IT 技术的进步及业务需求的快速增长，服务器也由几十台上升到成百上千台，IT 运维自动化是一个必然的趋势。Python 是当今最流行的编程语言之一，由于 Python 语言本身的优势，因此在编写自动化程序时简单、高效，实用效果立竿见影。目前开源软件社区优秀的自动化运维软件，如 Ansible、Airflow、Celery、Paramiko 等框架都使用 Python 语言开发，甚至一些大型商用的自动化部署系统都有 Python 的应用。因此，学好 Python，不仅可以自己编写自动化运维程序，而且可以对开源的自动化运维工具进行二次开发，这样才能在就业严峻的市场环境中具备较强的职场竞争力。

目前市场上介绍 Python 自动化运维的图书并不多，真正从实际应用出发，通过各种典型应用场景和项目案例来指导读者提高运维开发水平的图书就更少。本书以实战为主旨，通过 Python 运维开发中常见的典型应用（近百个场景），让读者全面、深入、透彻地学习 Python 在自动化运维领域的各种热门技术及主流开源工具的使用，提高实际开发水平和项目实战能力。

本书特色

1. 从基础讲起，适合零基础学习 Python 运维的读者

为了便于读者理解本书内容，从基础知识开始讲述，并结合实际应用，激发学习兴趣，提高学习效率。

2. 涵盖自动化运维的主流开源工具

本书涵盖 Ansible、APScheduler、Paramiko、Celery、Airflow、Docker 等主流运维工具的架构、原理及详细使用方法。

3. 项目案例典型，实战性强，有较高的应用价值

本书每一篇都提供了大量的实战案例，这些案例来源于作者开发的实际项目，具有很高的应用价值和参考性，而且分别使用不同的框架组合实现。这些案例稍加修改，便可用于实际项目开发中。

本书内容

第 1 章　自动化运维与 Python

本章介绍了自动化运维的背景知识、相关的开源工具及如何构造成熟的自动化运维体系。

第 2 章　基础运维

本章介绍如何使用 Python 处理文件、监控系统信息、监控文件系统、调用外部命令、日

志记录、搭建 FTP 服务器、发送邮件报警等实用基础运维技能。

第 3~5 章 多进程、多线程、协程

第 3~5 章对多进程和多线程中的创建方法、锁、信号量、事件、队列、进程池、线程池、协程的定义和使用、适用场景等进行了详细介绍，并配有示例用于练习和实际使用。

第 7~10 章 开源工具的使用方法

第 7~10 章主要介绍开源工具的使用方法，包括自动化运维工具 Ansible、定时任务框架 APScheduler、执行远程命令架构 Paramiko、分布式任务队列 Celery 及任务调度平台 Airflow。

第 11 章 Docker 容器技术

本章介绍高级运维工具 Docker，包括 Docker 的框架、原理、所能解决的问题、安装部署、使用方法等，同时也对 Docker 中的卷、卷的共享、如何自制镜像、Docker 网络配置等做了详细介绍。

示例源代码

本书示例源代码下载地址请扫描右边的二维码获取。如果下载有问题，请联系 booksaga@163.com，邮件主题为“Python 自动化运维快速入门”。

本书读者

- 需要做运维自动化开发的技术人员；
- 从零开始学 Python 的运维人员；
- 运维工程师、运维经理和网络管理员。

本书由郑征主笔，其他参与创作的还有吴贵文、董山海，在此表示感谢。

著者

2019 年 2 月

目 录

第一篇 Python 与基础运维

第二篇 中级运维

第三篇 高级运维

第一篇

Python与基础运维

第一篇

Python与量化选股

第 1 章
自动化运维与Python

随着云计算、自动化及人工智能时代的来临，Python 语言也成为当下最热门的语言之一。本章首先从自动化运维展开，介绍自动化运维的趋势、成熟的自动化运维体系构成、自动化运维相关的优秀开源工具；其次介绍了为什么选择 Python 语言作为自动化运维的必备工具；最后重点讲述 Python 的安装、开发工具、基础语法及相应的实例。

一方面，随着 IT 技术的飞速发展，软/硬件的设施日益复杂，企业的运维压力随之上升，自动化运维相关的人才供不应求；另一方面，国内 Python 方面的人才也非常短缺，学习 Python 及自动化运维，前景自然非常光明。除此之外，学习 Python 不仅可以做自动化工具，还可以做服务器后台、开发网络爬虫、Web 网站等，因此本章关于 Python 的基础知识对 Python 初学者也非常有帮助。

1.1 自动化运维概述

在运维技术还不成熟的早期，都是通过手工执行命令管理硬件、软件资源，但是随着技术的成熟及软/硬件资源的增多，运维人员需要执行大量的重复性命令来完成日常的运维工作。而自动化运维就是将这些原本大量重复性的日常工作自动化，让工具或系统代替人工来自动完成具体的运维工作，解放生产力，提高效率，降低运维成本。可以说自动化运维是当下 IT 运维工作的必经之路。

1.1.1 自动化运维势在必行

自动化运维之所以势在必行，原因有以下几点：

（1）手工运维缺点多。传统的手工执行命令管理软/硬件资源易发生操作风险，只要是手工操作，难免会有失误，一旦执行错误的命令，后果可能是灾难性的。当软/硬件资源增多时，手工配置效率低，增加运维人员的数量也会导致人力成本变高。

（2）传统人工运维难以管理大量的软/硬件资源。试想当机器数目增长到 1000 台以上时，仅靠人力来维护几乎是非常困难的事情。

（3）业务需求的频繁变更。现在的市场瞬息万变，业务唯有快速响应市场的需求才能可持续发展，对工具的需求和变更更是会越来越多，频率也越来越快，程序升级、上线、变更都是需要运维条线来支撑的。同样的，只有借力自动化运维，使用工具才能满足频繁变更的业务需求。

（4）自动化运维的技术已经成熟。自动化运维被广泛关注的一个重要原因就是自动化运维的技术已经非常成熟，技术的成熟为自动化运维提供了智力支持。云计算、大数据一方面刺激着自动化运维的需求，另一方面也助力自动化运维。微服务的软件架构、容器等技术都在推动自动化运维。

（5）工具已经到位。关于自动化运维的工具，无论是开源的工具还是企业级的产品，都是应有尽有，实现自动化运维已经势不可挡。

1.1.2 什么是成熟的自动化运维平台

现在成熟的自动化运维平台都具备哪些要素呢？一般来说，有以下几点：

（1）需要有支持混合云的配置管理数据库（CMDB）。CMDB 存储与管理企业 IT 架构中设备的各种配置信息，它与所有服务支持和服务交付流程都紧密相连，支持这些流程的运转、发挥配置信息的价值，同时依赖于相关流程保证数据的准确性。现在更多的企业选择将服务器资源放在云上，无论是公有云还是私有云都提供资源管理接口，利用这些接口构建一个自动化的 CMDB，同时增加日志审计功能，通过接口对资源的操作都应该记录，供后续审计。

（2）有完备的监控和应用性能分析系统。运维离不开监控和性能分析。资源监控（如服务器、磁盘、网络）和性能监控（如中间件、数据库）都是较为基础的监控，开源工具有 Zabbix、Nagios、OpenFalcon（国产）。应用性能分析，如某些 Web 请求的响应速度、SQL 语句执行的快慢等对于问题的定位是非常有帮助的，开源工具有 pinpoint、zipkin、cat；商业工具有 New Reclic、Dynatrace。

（3）需要具备批量运维工具。如何有效降低运维的成本呢，肯定是更少的人干更多的活。批量运维工具可有效节省大量人力，使用少量的人管理大量的服务器软/硬件资源成为可能。开源的批量运维工具有 ansible、saltstack、puppet、chef，其中 ansible 和 saltstack 纯由 Python 编写，代码质量和社区活跃程度都很高，推荐使用。

（4）需要有日志分析工具。随着服务器的增多，日志的采集和分析成了运维中的难点，试想如何快速地从成百上千台服务中采集日志并分析出问题所在呢？日志采集方面工具有 Sentry，也是纯由 Python 打造，日志分析有 ELK，两者都是开源的。

（5）需要有持续集成和版本控制工具。持续集成是一种软件实践，团队成员经常集成他们的工作，每次集成都通过自动化的构建来验证，从而尽早发现集成错误。持续集成的工具有 Hudson、CruiseControl、Continuum、Jenkins 等。版本控制是软件开发中常用的工具，比较著名的是 svn、git。

（6）还要有漏洞扫描工具。借助商业的漏洞扫描工具扫描漏洞，保护服务器资源不受外界的攻击。

1.1.3 为什么选择 Python 进行运维

为什么选择 Python 作为运维方面的编程语言呢？网络上不乏已经开发好的运维软件，但是运维工作复杂多变，已有的运维软件不可能穷尽所有的运维需求，总有一些运维需求需要运

维人员自己去编写程序解决，这样做运维很有必要学会一门编程语言来解决实际问题，让程序代替人力去自动运维，减轻重复工作，提高效率。接下来，选择哪一门语言合适呢？当然是选一门学习成本低、应用效果高的，这方面 Python 的性价比最高，原因是：一方面，大部分的开源运维工具都是由纯 Python 编写的，如 Celery、ansible、Paramiko、airflow 等，学习 Python 后可以更加顺畅地使用这些开源工具提供的 API，可以阅读这些开源工具的源代码，甚至可以修改源代码以满足个性化的运维需求；另一方面，Python 与其他语言相比，有着以下优势。

- 简单、易学。阅读 Python 程序类似读英文，编码上避免了其他语言的烦琐。
- 更接近自然的思维方法，使你能够专注于解决问题而不是语法细节。
- 规范的代码，Python 采用强制缩进的方式使得代码具有较好的可读性。
- Python 拥有一个强大的标准库和丰富的第三方库，拿来即用，无须重复造轮子。
- 可移植性高，Linux、UNIX、Windows、Android、Mac OS 等一次编写，处处运行。
- 实用效果好，学习一个知识点，能够直接实战——用在工作上，立竿见影。
- 潜移默化，学习 Python 能够顺利理解并学习其他语言。

Python 也是最具潜力的编程语言，在 2018 年 IEEE 发布的顶级编程语言排行榜中，Python 排名第一，如图 1.1 所示。而图 1.2 表明，Python 现在已成为美国名校中最流行的编程入门语言。ANSI / ISO C + +标准委员会的创始成员 Bruce Eckel 曾说过：“life is short，You need Python。”一度成为 Python 的宣传语，这正是说明 Python 有着简单、开发速度快、节省时间和精力的特点。另外，Python 是开放的，也是开源的，有很多善良可爱的开发者在第三方库贡献了自己的源代码，许多功能都可以直接拿来使用，无须重新开发，这也是 Python 的强大之处。

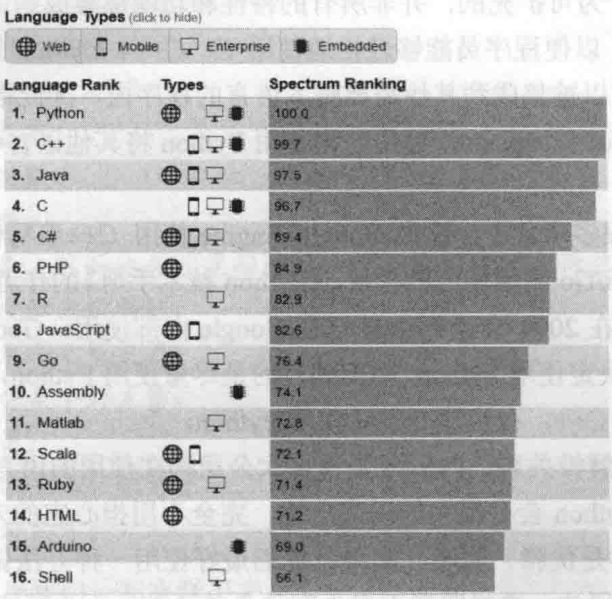


图 1.1　IEEE Spectrum 给出的编程语言排行榜

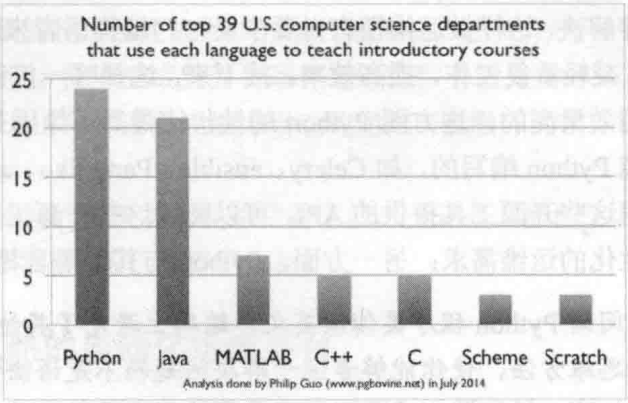


图 1.2　在美国名校编程语言的流行情况

下面摘抄一段 Python 在维基百科中的介绍。

Python 是完全面向对象的语言。函数、模块、数字、字符串都是对象，并且完全支持继承、重载、派生、多继承，有益于增强源代码的复用性。由于 Python 支持重载运算符，因此 Python 也支持泛型设计。相对于 Lisp 这种传统的函数式编程语言，Python 对函数式设计只提供了有限的支持。有两个标准库（functools, itertools）提供了 Haskell 和 Standard ML 中久经考验的函数式程序设计工具。

虽然 Python 被粗略地分类为“脚本语言”（Script Language），但实际上一些大规模软件开发项目，如 Zope、Mnet 及 BitTorrent 及 Google 也广泛地使用它。Python 的支持者喜欢称它为一种高级动态编程语言，原因是“脚本语言”泛指仅作简单程序设计任务的语言，如 shell script、VBScript 等只能处理简单任务的编程语言，并不能与 Python 相提并论。

Python 本身被设计为可扩充的，并非所有的特性和功能都集成到语言核心。Python 提供了丰富的 API 和工具，以便程序员能够轻松地使用 C、C++、Cython 来编写扩充模块。由于 Python 编译器本身也可以被集成到其他需要脚本语言的程序内，因此很多人还把 Python 作为一种“胶水语言”（Glue Language）使用，即使用 Python 将其他语言编写的程序进行集成和封装。

在 Google 内部的很多项目中，比如 Google Engine 使用 C++编写性能要求极高的部分，然后使用 Python 或 Java/Go 调用相应的模块。《Python 技术手册》的作者马特利（Alex Martelli）说：“这很难讲，不过在 2004 年，Python 已在 Google 内部使用，Google 招募了许多 Python 高手，但在这之前就已决定使用 Python。他们的目的是尽量使用 Python，在不得已时改用 C++；在操控硬件的场合使用 C++，在快速开发时使用 Python。”

一些技术术语不理解没关系，Python 是许多大公司都在使用的语言，如 Google、NASA、知乎、豆瓣等，学习 Python 会有很大的用武之地，完全不用担心它的未来。

Python 的设计哲学是优雅、明确、简单。提倡最好使用一种方法做一件事，Python 的开发者一般会拒绝花哨的语法，选择明确而很少有歧义的语法。下面再摘一段 Python 格言：

Beautiful is better than ugly.
Explicit is better than implicit.

Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
*Although never is often better than *right* now.*
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

上面的格言来自 Python 官方，也有中文版本，如下：

优美胜于丑陋，明晰胜于隐晦，
简单胜于复杂，复杂胜于繁芜，
扁平胜于嵌套，稀疏胜于密集，
可读性很重要。
虽然实用性比纯粹性更重要，
但特例并不足以把规则破坏掉。
错误状态永远不要忽略，
除非你明确地保持沉默，
直面多义，永不臆断。
最佳的途径只有一条，然而他并非显而易见———谁叫你不是荷兰人？
置之不理或许会比慌忙应对要好，
然而现在动手远比束手无策更好。
难以解读的实现不会是个好主意，
容易解读的或许才是。
名字空间就是个“顶呱呱”的好主意。
让我们想出更多的好主意！

Python 如此优秀，让我们一起来学习吧。