配置Nginx的深入指南

精通 Nginx（第2版）

Mastering NGINX Second Edition

[瑞士] Dimitri Aivaliotis 著

李红军 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

精通 Nginx（第2版）

[瑞士] Dimitri Aivaliotis 著

李红军 译

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

精通Nginx : 第2版 / (瑞士) 艾维利(Dimitri Aivaliotis) 著 ; 李红军译. -- 北京 : 人民邮电出版社, 2017.8
ISBN 978-7-115-45996-1

Ⅰ. ①精… Ⅱ. ①艾… ②李… Ⅲ. ①互联网络－网络服务器 Ⅳ. ①TP368.5

中国版本图书馆CIP数据核字(2017)第143486号

版权声明

◆ 著 [瑞士] Dimitri Aivaliotis
译 李红军
责任编辑 陈冀康
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京市艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：16.5
字数：320 千字 2017 年 8 月第 1 版
印数：1－2 400 册 2017 年 8 月北京第 1 次印刷

著作权合同登记号 图字：01-2016-7603 号

定价：59.00 元

读者服务热线：(010)81055410 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

内容提要

Nginx 是一个高性能的轻量级 Web 服务器，本书从配置文件的角度出发，介绍了多种关于 Nginx 配置的技巧。

本书以模块化风格写成，几乎每一章都是一个独立的模块，读者将能够自由地在各个模块间切换阅读。全书分两部分，第一部分用 9 章内容介绍了安装 Nginx 及第三方模块、配置向导、使用 mail 模块、Nginx 作为反向代理、Nginx Http 服务器、Nginx 的开发、在 Nginx 中集成 Lua 以及故障排除技巧；第二部分用 4 个附录的形式介绍了指令参考、Rewrite 规则指南、Nginx 社区以及 Solaris 系统下的网络调优。

本书适合在安装和配置服务器方面有经验的系统管理员或系统工程师，阅读本书不需要任何 Nginx 使用经验，相信这本书会帮助读者更好地完成任务。

作者简介

Dimitri Aivaliotis 在硅谷担任产品工程师（production engineer）。他的职业生涯从为学校构建基于 Linux 的计算机网络到为银行构建多数据中心的高可用性基础设施和流行的门户网站。他在解决客户问题上已经花费了 10 多年时间，并且在这条路上发现了 Nginx。

Dimitri 以最优异的成绩获得了伦斯勒理工学院的理科学士，并且获得了佛罗里达州立大学管理信息系统的理科硕士。

人们会认为第 2 版应该很容易撰写，纠正第 1 版的错误并更新其内容。一方面，这比从头开始撰写要少写很多，但另一方面，一切都必须重新评估。它不像起初看起来那么容易。

我要感谢所有审稿人，让我诚实，并指出阐述不清楚的内容。当然，剩下的任何错误是我自己的。

感谢 Packt 再次给我撰写这本书的机会。

感谢 Nginx 公司创建一个如此灵活和高效的产品，它在当今仍被广泛使用。

审稿人简介

Markus Jelsma 是 Openindex 私人有限公司的 CTO 与共同所有人，Openindex 私人有限公司是一家专门从事开源搜索和爬虫解决方案的荷兰公司。作为 Apache Nutch 的提交者和 PMC 成员，他是搜索引擎技术和爬虫解决方案方面的专家。

译者序

Nginx 是俄罗斯软件工程师 Igor Sysoev 开发的免费开源 Web 服务器软件。Nginx 于 2004 年发布，聚焦于轻量级、高并发、高性能、高度模块化的设计与低内存消耗问题。它具有多种 Web 服务器功能特性：负载均衡、缓存、访问控制、带宽控制以及高效整合各种应用的能力，这些特性使得 Nginx 很适合于现代网站架构。目前，Nginx 是互联网上仅次于 Apache 的第二流行的开源 Web 服务器软件。

《精通 Nginx（第 2 版）》一书是艾维利的代表作，该书第 1 版于 2013 年 3 月出版，时隔 3 年，本书第 2 版于 2016 年 7 月出版。在《精通 Nginx（第 2 版）》中，艾维利从配置文件的角度出发，介绍了多种关于 Nginx 的配置技巧。本书以模块化风格写成，几乎每一章都是一个独立的模块，读者将能够自由地在各个模块间切换阅读。全书分两部分，第一部分用 9 章内容介绍了安装 Nginx 及第三方模块、配置指南、使用 mail 模块、Nginx 作为反向代理、反向代理高级话题、Nginx HTTP 服务器、Nginx 的开发、在 Nginx 中集成 Lua 以及故障排除技巧；第二部分用 4 个附录的形式介绍了指令参考、Rewrite 规则指南、Nginx 社区以及 Solaries 系统下的网络调优。了解 Nginx 新版本的新内容、熟练掌握 Nginx 的开发并将其灵活运用在工作环境中，这些都是 Nginx 用户和 Web 开发人员的迫切需求。

感谢原作者艾维利，感谢他为全球的 Nginx 开发者带来了这本充满智慧的 Nginx 图书。感谢本书第 1 版译者陶利军老师所做的工作。感谢人民邮电出版社信息技术分社引进如此高品质的图书，感谢人民邮电出版社的陈冀康老师对我的信任，把这本书交给了我翻译。他在本书编辑过程中所表现出来的热心、耐心和敬业精神令我十分感动，与陈冀康老师合作，让人非常愉快！人民邮电出版社的编辑们为本书的出版在幕后做了大量艰苦细致的工作，才让本书得以面世，译者谨向他们表示衷心的感谢。希望所有使用 Nginx 以及对 Nginx 开

发感兴趣的读者都能从中获益。

译者也是抱着一颗炽热的学习之心在阅读和翻译本书的，并且希望把这本书推荐给身边更多的朋友。由于译者自身对本书内容的理解深度可能有所欠缺，在翻译的过程中难免出现一些不够准确或者不清楚的描述。读者若发现翻译处理不当之处，欢迎批评指正。

最后，需要深深感谢在翻译过程中给予我理解、支持、帮助的家人和朋友们。

李红军

2016 年 12 月 19 日

hjliemail@yahoo.com

译者简介

李红军，W3China 论坛 Java/Eclipse 版版主。沈阳工业大学计算机应用技术专业硕士，2006 年加入 EasyJF 开源团队，并担任核心开发者，开发出了国内第一款最完备的 J2EE MVC 框架 EasyJWeb。2008 年与徐涵等共同翻译出版 REST 专著《RESTful Web Services 中文版》，2009 年翻译出版《MySQL 核心技术手册（第 2 版）》、《SQL 核心技术手册（第 3 版）》，2013 年翻译出版《云计算：原理与范式》。现为上海立信会计金融学院图书馆技术部工程师。研究兴趣包括网络存储、数据库、虚拟化与云计算以及 Linux、Apache 等开源技术。

前言

Nginx 是一台高性能 Web 服务器，它使用了非常少的系统资源。在互联网上出现很多 how-to 和示例配置文件，这会澄清 Nginx 配置的浑水。这样做，你将会学到在各种环境中如何调整 Nginx 以及一些模糊的配置选项，以便设计一个符合你需求的配置文件。

在已经理解了如何根据自己的需求来构建一个配置文件后，你就不再需要复制、粘贴、配置片段了。这是一个过程，而且会有曲折。不过本书有关技巧的解释，会使你觉得手写配置文件是一件很舒服的事情。万一事情不像期望中的那样工作，你将能够独立地调试该问题，或者至少能够寻求帮助，而不会觉得自己好像都没有尝试过。

这是一本以现代风格所写的书，这种设计有助于你尽可能快地获取信息。几乎每一章都是一个独立模块，你可以根据需要自由地跳到任何地方来获取更深入的特定主题。如果觉得错过了某些主要的内容，那么你可以返回去读取前面的章节。它们在这种方式下构建，以帮助你的配置文件一步步成长。

本书涵盖的内容

第 1 章，安装 Nginx 及第三方模块。教你如何在选择的操作系统上安装 Nginx 以及在你的安装中如何包含第三方模块。

第 2 章，配置指南，讲解 Nginx 的配置文件格式，你将学到每一个不同区段的配置，如何配置全局参数以及 location 的用处。

第 3 章，使用 mail 模块，探索 Nginx 的邮件代理模块，详细介绍配置的方方面面。在本章中，还有一个认证服务的代码示例。

第 4 章，Nginx 作为反向代理，介绍反向代理的概念，并且描述 Nginx 如何充当该角色。

第 5 章，反向代理高级话题，深入研究使用 Nginx 作为反向代理解决可伸缩及性能的问题。

第 6 章，Nginx HTTP 服务器，描述如何使用各种模块，包括通过 Nginx 解决常见的 Web 服务问题。

第 7 章，Nginx 的开发，展示 Nginx 如何与你的应用程序集成，以便更快速地把内容交付给你的用户。

第 8 章，在 Nginx 中集成 Lua，对如何使用嵌入式脚本语言 Lua 扩展 Nginx 功能提供一个概览。

第 9 章，故障排除技巧，研究一些常见的配置文件，一旦出现问题如何调试，以及调优性能的一些建议。

附录 A，指令参考，提供一个方便的配置指令参考，这些指令贯穿全书，也有一些以前未覆盖到的其他指令。

附录 B，Rewrite 规则指南，描述如何使用 Nginx 的 Rewrite 规则模块，并描述将 Apache 格式的 rewrite 规则转换为 Nginx 可处理的 rewrite 规则的一些步骤。

附录 C，Nginx 社区，介绍可以搜寻到的更多的线上资源。

附录 D，Solaris 系统下的网络调优，详述 Solaris 10 及以上版本系统下的网络调优的必要性。

使用这本书你需要做的

任何现代 Linux PC 都能充分运行本书中的示例代码。示例代码在每一章都给定了安装操作指南。基本上可以归纳如下。

- 构建环境：编译器、头文件，等等。
- Nginx：最好是最新版本。
- Ruby：最好从 https://rvm.io 安装。
- Perl：默认版本较好。

谁需要这本书

这本书适用有经验的系统管理员或者系统工程师，熟悉服务器的安装和配置以满足特定的需求。你不需要有使用 Nginx 的经验。

本书约定

本书采用了一些不同风格的文本样式，便于区分不同的信息。这里有一些格式的示例，并有解释说明。

文本格式的代码字符、数据库表名、文件夹名、文件名、文件扩展名、路径名、虚拟 URL、用户输入和 Twitter 处理如下列格式显示：“This section will be placed at the top of the `nginx.conf` configuration file.”

代码块设置如下：

```
http {
    include       /opt/local/etc/nginx/mime.types;
    default_type  application/octet-stream;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout  65;
    server_names_hash_max_size 1024;
}
```

任何命令行输入或者输出书写如下：

```
$ mkdir $HOME/build
$ cd $HOME/build && tar xzf nginx-<version-number>.tar.gz
```

新术语与**重要的字**以粗体显示。你在屏幕上看到的字，例如，在菜单或者对话框中，会出现类似这样的版本：“Clicking the **Next** button moves you to the next screen.”

警告或者重要的事项出现在这样的框内。

提示和技巧以这种方式显示。

读者反馈

从读者获悉的反馈总是受欢迎的，它让我们知道你对本书的想法——什么是你喜欢的，什么是你不喜欢的。对于我们而言，读者反馈很重要，它使我们的课题发挥更大作用。

向我们发送一般的反馈，只需简单地发送电子邮件至 feedback@packtpub.com，并且在邮件的主题部分提及本书名称就可以了。

如果有你擅长的主题，并且你对写作或者投稿感兴趣，那么你可以看看我们的作者向导 www.packtpub.com/authors。

客户支持

现在你自豪地成为 Packt 书的读者了。我们可以为你提供若干售后服务。

下载示例代码

你可以从 http://www.packtpub.com 上用你的账号下载本书的示例代码，如果你从别的地方购买了这本书，你可以访问 http://www.packtpub.com/support 并注册，那么代码会以邮件的形式直接发送给你。

你可以通过如下步骤来下载代码文件。

1. 使用你的电子邮件地址和密码登录或注册我们的网站。
2. 鼠标指针悬浮在顶部的 **SUPPORT** 标签。
3. 单击 **Code Downloads & Errata**。
4. 在 Search 框中，输入书名。
5. 选择你要下载代码文件的图书。
6. 从下拉菜单中选择你购买的图书。
7. 单击 **Code Download**。

下载完代码文件后，请确保你使用了如下最新版本的软件来解压或提取文件夹。

- Windows 系统下的 WinRAR、7-Zip。

- Mac 系统下的 Zipeg、iZip、UnRarX。
- Linux 系统下的 7-Zip、PeaZip。

下载本书的彩图

我们还为你提供了 PDF 文件，它载有本书使用到的截图。这些彩图可以更好地帮助你理解输出中的变化。你可以从 http://www.packtpub.com/sites/default/files/downloads/Bookname_ColorImages.pdf 下载该文件。

勘误表

尽管我们非常小心地确保图书内容的准确性，但是错误还是会发生。如果你在我们的任何一本图书中发现了错误——可能是文本错误，也可能是代码错误，并把它向我们报告，我们将非常感谢。通过这样做，你可以使其他读者免于阅读错误，并且帮助我们在本书的后续版本中更正。如果你想查找任何勘误表，请通过 http://www.packtpub. com/submit-errata 访问它们，选择你要找的图书，单击 **Errata Submission Form** 链接，就会进入本书的详细勘误表。一旦你的勘误表被校验，那么你的提交将会被接受，并且在我们网站上的勘误表将会被更新，或者添加到任何位于标题部分下的现有勘误表中。

进入 https://www.packtpub.com/books/content/support，通过在搜索框输入图书名称，你都可以查看之前的勘误表。需要的信息会出现在 **Errata** 部分。

版权保护

在互联网上，侵犯受版权保护资料的盗版行为，一直是一个持续的跨越全媒体的问题。在 Packt，我们非常重视版权和许可。如果你在互联网上偶然发现以任何形式非法复制我们文字的地方，请立即提供给我们地址或者网站的名字，以便使我们能够追究补救办法。

请联系我们 copyright@packtpub.com，并且附上涉嫌盗版的材料链接。

我们非常感谢你帮助保护我们的作者，并且我们有能力给读者带来有价值的内容。

问题

如果有任何关于本书的问题，你可以联系我们 questions@packtpub.com，我们将会尽量解决。

目录

第 1 章
安装 Nginx 及第三方模块

Nginx 最初的设计，是成为一个 HTTP 服务器，一个能解决 C10K 问题的 HTTP 服务器。关于 C10K 这个问题，Daniel Kegel 在 http://www.kegel.com/c10k.html 页面有具体描述，它旨在设计一个同时连接处理 10000 连接数的 Web 服务器。为了实现这个目标，Nginx 通过基于事件的连接—处理机制，并且操作系统也要使用相应的事件机制，便可以解决 C10K 问题。

在我们开始探索如何配置 Nginx 之前，首先我们要安装它。这一章将详细讲述如何安装 Nginx，以及如何获取正确的模块并安装与配置它们。Nginx 是模块化设计的，并且有非常丰富的第三方模块开发者社区。它们的设计者通过创建这些模块为核心 Nginx 服务器增添了功能，我们可以在编译安装 Nginx 时将它们添加到 Nginx 服务器。

在这一章中，本书涉及如下内容。

- 使用包管理器安装 Nginx。
- 通过源代码安装 Nginx。
- 为 Web 或者 Mail 服务器配置 Nginx。
- 配置 SSL 支持。
- 使用各种模块。
- 查找并安装第三方模块。
- 添加对 Lua 的支持。
- 组合在一起。

1.1 使用包管理器安装 Nginx

使用包管理器安装 Nginx 的机会，是你所使用的操作系统已经提供了 nginx 的安装包。使用包管理器安装 Nginx 的方法很简单，只需要使用包管理器安装命令就可以了：

- Linux（基于 deb）

```
sudo apt-get install nginx
```

- Linux（基于 rpm）

```
sudo yum install nginx
```

- FreeBSD

```
sudo pkg_install -r nginx
```

命令 sudo 表示的是通过操作系统中的超级用户（root）权限执行的命令。如果操作系统支持 RBAC（role-based access control），那么可以用一个不同的命令，例如“pfexec”，来达到同样的目的。

通过上述命令，Nginx 将会安装到操作系统的标准位置下。如果使用操作系统的安装包安装 Nginx，那么通过上面的命令来安装是最佳方式。

Nginx 核心团队也提供了稳定的二进制版本，可以从 http://nginx.org/en/download.html 页面下载可用的版本。未发布 nginx 安装包的系统用户（例如，CentOS），可以使用下面的指导来安装预测试、预编译二进制版本。

1.1.1 在 Centos 上安装 Nginx

通过创建下面的文件，在系统中添加 Nginx 仓库的 yum 配置：

```
sudo vi /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=0
enabled=1
```

然后，通过执行如下命令来安装 nginx：

```
sudo yum install nginx
```

也可以按照前面介绍的 URL 下载 nginx 发行版安装。

1.1.2　在 Debian 上安装 Nginx

使用如下步骤在 Debian 上安装 Nginx。

1．通过从 http://nginx.org/keys/nginx_signing.key 下载并安装 Nginx 签名 key，将该签名 key 添加到系统的 apt 密钥中：

```
sudo apt-key add nginx_signing.key
```

2．将 nginx.org 仓库追加到/etc/apt/sources.list 文件的末尾：

```
vi /etc/apt/sources.list
deb http://nginx.org/packages/debian/jessie nginx
deb-src http://nginx.org/packages/debian/jessie nginx
```

3．然后，通过执行如下命令来安装 nginx：

```
sudo apt-get update
sudo apt-get install nginx
```

如果所使用的操作系统在它可用的安装包中未包含 nginx，或是所包含的版本太老不能满足需要，或是 nginx.org 并未提供所需要的安装包，或是你想使用“development”版本的 Nginx，或者是你想启用或禁用特定的模块，那么从源代码编译的方法来安装 Nginx 是唯一可用的另外一个方法。

1.2　从源代码安装 Nginx

Nginx 代码提供了两种独立的下载分支——开发版与稳定版。开发分支是一个正处于积极开发状态的版本。在这个版本中，会有一些新功能被集成到其中，在稳定版中是找不到这些功能的。当发布一个“开发”版时，它会经历同样的 QA 和作为稳定版的一组类似功能测试。因此，无论哪一个分支都可以用于生产环境中。两者主要的不同，在于对第三方模块的支持。在开发版中，内部的 API 可能会发生改变，而稳定版则保持不变。因此，为了与第三方模块向下兼容，在稳定版中第三方模块都可以有效使用。

1.2.1　准备编译环境

为了从源代码编译 Nginx，系统需要满足某些必要条件。除了编译器之外，如果想

分别启用 SSL 支持和使用 rewrite 模块，那么还需要提供相应的 OpenSSL 与 PCRE（Perl Compatible Regular Expressions）库及开发头文件。rewrite 模块是默认安装的。如果你没有 PCRE 库与开发头文件，你需要在配置阶段禁用 rewrite 模块。这依赖于系统，也有可能在系统中已经默认安装了这些必要条件。如果没有安装，则需要从其安装包安装或者从源码下载并解压安装，并在 Nginx 的配置脚本文件中指定它们在系统中的安装位置。

如果在配置文件中使用了--with-<library>=<path>选项，那么 Nginx 会试图建立一个静态依赖库。如果你想让 Nginx 不依赖于系统任何其他部分，或是想多获得些 nginx 的二进制额外性能，那么你可能会使用构建静态库的做法。如果你使用的外部库功能只能从某一个版本起有效（例如，NPN[Next Protocol Negotiation] TLS 扩展从 OpenSSL 1.0.1 版有效），那么你就不得不将其指定到特定版本解压后的源代码路径中。

根据自己的喜好，你可能会提供其他的、可选安装包。你可以为这些安装包提供支持。它们包括 MD5 和 SHA-1 以支持散列算法、zip 压缩库、libatomic 库。在 Nginx 中，很多地方会用到散列算法，例如，为了计算 URI 散列进而计算缓存 key。

zlib 压缩库被用来投递 gzip 压缩内容。如果 atomic_ops 库有效，那么 Nginx 会用它来实现自动内存更新操作，以便实现高性能的内存锁定代码。

1.2.2 从源代码编译

读者可以从 http://nginx.org/en/download.html 下载 Nginx，在该页面找到.tar.gz 或者.zip 格式的源代码分支，按照如下步骤将下载的安装包解压到一个临时目录中：

```
$ mkdir $HOME/build
$ cd $HOME/build && tar xzf nginx-<version-number>.tar.gz
```

使用下面的命令配置 Nginx：

```
$ cd $HOME/build/nginx-<version-number> && ./configure
```

然后，使用下面的命令进行编译安装：

```
$ make && sudo make install
```

在编译自己的二进制 nginx 时，你会有很大的灵活性来包含你仅使用的功能。你已经指定使用哪个用户运行 Nginx 了吗？你要使用默认的 logfile 位置，以便不用在 Nginx 的配置文件明确地说明它们吗？表 1-1 所示是配置选项列表，通过它来帮助你设计出自己的 nginx 命令。这些选项对 Nginx 都是有效的，模块可以被独立激活。

表 1-1　　通用配置选项

选项	解释
--prefix=<path>	Nginx 安装的根路径，所有其他的安装路径都要依赖于该选项
--sbin- path=<path>	指定 nginx 二进制文件的路径。如果没有指定，那么这个路径会依赖于——prefix 选项
--conf- path=<path>	如果在命令行没有指定配置文件，那么将会通过这里指定的路径，nginx 将会去那里查找它的配置文件
--error-log- path=<path>	指定错误文件的路径，nginx 将会往其中写入错误日志文件，除非有其他的配置
--pid- path=<path>	指定的文件将会写入 nginx master 进程的 pid，通常在/var/run 下
--lock- path=<path>	共享存储器互斥锁文件的路径
--user=<user>	worker 进程运行的用户
--group=<group>	worker 进程运行的组
--with-file-aio	为 FreeBSD 4.3 +和 Linux 2.6.22 +系统启用异步 I/O
--with-debug	这个选项用于启用调试日志。在生产环境的系统中不推荐使用该选项

如表 1-2 所示，你可以使用优化编译，但你可能无法在包管理器安装中获得优化。这正是表 1-2 中选项的用武之地。

表 1-2　　配置优化选项

选项	说明
--with- cc=<path>	如果想设置一个不在默认 PATH 下的 C 编译器
--with- cpp=<path>	设置 C 预处理器的相应路径
--with-cc- opt=<options>	指定必要的 include 文件路径，可能(-I<path>) 指出，也可能是优化（-O4）并指定一个 64 位构建
--with-ld- opt=<options>	包含连接器库的路径（-L<path>）和运行路径（-R<path>）
--with-cpu- opt=<cpu>	通过该选项为特定的 CPU 构建 Nginx

1.2.3　为 Web 或者 Mail 服务器配置 Nginx

Nginx 是一个独一无二的高性能 Web 服务器，它也被设计成为一个邮件代理服务器。根据你构建 Nginx 的目标，可将其配置成一个 Web 加速器、Web 服务器、邮件代理，或者是集三者为一体。你可以将任何服务安装在一个二进制文件中，这样做的好处是可以通过配置文件来设置 Nginx 服务器的角色，或者根据需要在高性能的环境中安装一个精简的二进制 Nginx 文件。

1.2.4 邮件代理的配置选项

如表 1-3 所示，是邮件模块独有的配置选项。

表 1-3 mail 配置选项

选项	说明
--with-mail	该选项用于启用 mail 模块，该模块默认没有被激活
--with-mail_ssl_module	为了代理任何一种类型的使用 SSL/TLS 的 mail，需要激活该模块
--without-mail_pop3_module	在启用 mail 模块后，单独地禁用 POP3 模块
--without-mail_imap_module	在启用 mail 模块后，单独地禁用 IMAP 模块
--without-mail_smtp_module	在启用 mail 模块后，单独地禁用 SMTP 模块
--without-http	该选项将会完全禁用 http 模块，如果你只想支持 mail，那么可以使用它

对于典型的 mail 代理，我推荐将 Nginx 配置为：

```
$ ./configure --with-mail --with-mail_ssl_module --with-openssl=${BUILD_DIR}/openssl-1.0.1p
```

对于邮件服务器来说，现在几乎每一个邮件服务器的安装都需要安装 SSL/TLS，并且没有一个邮件代理启用了预期功能的劫持用户。我推荐静态编译 OpenSSL，以便对操作系统中的 OpenSSL 库没有依赖性。不过，这确实意味着你必须保持警惕，确保你的静态编译的 OpenSSL 保持最新，并在必要时重建你的二进制文件。在前面命令中使用的变量 BUILD_DIR 需要提前设置。

1.2.5 指定路径的配置选项

表 1-4 显示了 http 模块有效的配置选项，从激活 Perl 模块到指定临时目录的位置。

表 1-4 http 配置选项

选项	说明
--without-http-cache	在使用 upstream 模块时，Nginx 能够配置本地缓存内容。这个选项能够禁用缓存
--with-http_perl_module	Nginx 配置能够扩展使用 Perl 代码。这个选项启用这个模块（然而，I/O 受阻塞时，使用这个模块会降低性能。）
--with-perl_modules_path= <path>	对于额外嵌入的 Perl 模块，使用该选项指定该 Perl 解析器的路径。也可以通过配置选项来指定 Perl 模块解析器的位置

续表

选项	说明
--with-perl=<path>	如果在默认的路径中没有找到 Perl，那么指定 Perl（5.6.1 版本以上）的路径
--http-log-path=<path>	http 访问日志的默认路径
--http-client-body-temp- path= <path>	从客户端收到请求后，该选项设置的目录用于作为请求体临时存放的目录。如果 WebDAV 模块启用，那么推荐设置该路径为同一文件系统上的目录作为最终的目的地
--http-proxy-temp-path= <path>	在使用代理后，通过该选项设置存放临时文件路径
--http-fastcgi-temp-path= <path>	设置 FastCGI 临时文件的目录
--http-uwsgi-temp-path= <path>	设置 uWSGI 临时文件的目录
--http-scgi-temp-path=<path>	设置 SCGI 临时文件的目录

1.3　配置 SSL 支持

对于 TLS/SSL 协议，Nginx 使用 OpenSSL 项目。有关此开源工具包的更多信息，请访问 https://www.openssl.org。你可以从操作系统或者直接从工具包的单独副本来获取对 SSL 的支持。如果使用不带--with-ssl 选项的--with-http_ssl_module 或者--with-mail_ssl_module，你正在使用执行了 configure 命令的、安装在计算机上的 OpenSSL 库。如果你想要针对特定版本的 OpenSSL 进行编译，请下载该分发包，将其解压缩到一个目录中，然后将该目录的路径指定为--with-openssl 的参数。使用--with-openssl-opt 选项为 OpenSSL 本身指定额外的构建选项。

例如，为了使用具有优化椭圆曲线的 OpenSSL 来构建 Nginx，您将使用如下的命令：

```
$ ./configure --with-http_ssl_module --with-openssl=${BUILD_DIR}/openssl-1.0.1p --with-openssl-opt=enable-ec_nistp_64_gcc_128
```

1.4　使用各种模块

在 Nginx 发布的版本中，除了 http 和 mail 模块之外，还有其他一些模块。这些模块并没有在默认安装中激活，但是可以在编译安装时适当地配置选项--with-<module-name>_module 来启用相应的选项，如表 1-5 所示。

表 1-5 http 模块配置选项

选项	说明
--with-http_ssl_module	如果需要对流量进行加密，那么可以使用到这个选项，在 URL 中开始部分将会是 https（需要 OpenSSL 库）
--with-http_realip_module	如果你的 Nginx 在七层负载均衡器或者是其他设备之后，它们将 http 头中的客户端 IP 地址传递，那么你将会需要启用这个模块。在多个客户处于一个 IP 地址的情况下使用
--with-http_addition_module	这个模块作为一个输出过滤器，使你能够在请求经过一个 location 前或者后时在该 location 本身添加内容
--with-http_xslt_module	该模块用于处理 XML 响应转换，基于一个或者多个 XSLT 格式（需要 libxml2 和 libxslt 库）
--with-http_image_filter_module	该模块被作为图像过滤器使用，在将图像投递到客户之前进行处理（需要 libgd 库）
--with-http_geoip_module	使用该模块，能够设置各种变量以便在配置文件中的区段使用，基于地理位置查找客户端 IP 地址（需要 MaxMind GeoIP 库和相应的预编译数据库文件）
--with-http_sub_module	该模块实现了替代过滤，在响应中用一个字符串替代另一个字符串，提醒一句：使用该模块隐式禁用标头缓存
--with-http_dav_module	启用这个模块将激活使用 WebDAV 的配置指令。请注意，这个模块也只在有需要使用的基础上启用，如果配置不正确，它可能带来安全问题
--with-http_flv_module	如果需要提供 Flash 流媒体视频文件，那么该模块将会提供伪流媒体
--with-http_mp4_module	这个模块支持 H.264/AAC 文件伪流媒体
--with-http_gzip_static_module	当被调用的资源没有 .gz 结尾格式的文件时，如果想支持发送预压缩版本的静态文件，那么使用该模块
--with-http_gunzip_module	对于不支持 gzip 编码的客户，该模块用于为客户解压缩预压缩内容
--with-http_random_index_module	如果你想提供从一个目录中随机选择文件的索引文件，那么这个模块需要被激活
--with-http_secure_link_module	该模块提供了一种机制，它会将一个散列值链接到一个 URL 中，因此，只有那些使用正确的密码能够计算链接
--with-http_stub_status_module	启用这个模块后会收集 Nginx 自身的状态信息。输出的状态信息可以使用 RRDtool 或类似的内容来绘制成图

正如你所看到的，所有这些模块都是建立在 http 模块的基础之上，它们提供了额外的功能。在编译时启用这些模块根本不会影响到运行性能，以后在配置使用这些模块时性能会产生影响。

因此，对于网络加速器/代理，就配置选项来说，我想提出以下建议。

```
$ ./configure --with-http_ssl_module --with-http_realip_module --with- http_
geoip_module  --with-http_stub_status_module  --with-openssl=${BUILD_DIR}/
openssl-1.0.1p
```

下面是 Web 服务器的建议：

```
$ ./configure --with-http_stub_status_module
```

不同之处在于 Nginx 面对的客户，处于 Web 加速角色时，会考虑到 SSL 请求的终结，也包括处理代理客户和基于客户来源决策；处于 Web 服务角色时，则仅需要提供默认文件访问能力。

我总是推荐启用 stub_status 模块，这是因为它提供了收集 Nginx 如何执行、如何对其度量的一个方法。

禁用不再使用的模块

有些 http 模块通常情况下是激活的，但是可以通过设置适当的--without-<module-name>_module 选项禁用它们。如果在配置中不使用这些模块，如表 1-6 所示，那么你可以禁用它们。

表 1-6　　禁用的配置选项

选项	说明
--without-http_charset_module	该字符集模块负责设置 Content-Type 响应头，以及从一个字符集转换到另一个字符集
--without-http_gzip_module	gzip 模块作为一个输出过滤器，在将内容投递到客户时对内容进行压缩
--without-http_ssi_module	该模块是一个过滤器，用于处理 SSI 包含。如果启用了 Perl 模块，那么额外的 SSI 指令（perl）可用
--without-http_userid_module	userid 模块能够使得 Nginx 设置 cookies，用于客户标识。变量$uid_set 和$uid_got 可以记录用户跟踪
--without-http_access_module	access 模块基于 IP 地址控制访问 location
--without-http_auth_basic_module	该模块通过 HTTP 基本身份验证限制访问
--without-http_autoindex_module	如果一个目录中没有 index 文件，那么 autoindex 模块能够收集这个目录列出文件
--without-http_geo_module	该模块能够让你基于客户端 IP 地址设置配置变量，然后根据这些变量的值采取行动

续表

选项	说明
--without-http_map_module	map 模块能够让你映射一个变量到另一个变量
--without-http_split_clients_module	该模块创建用于 A/B 测试的变量
--without-http_referer_module	该模块能够让 Nginx 阻止基于 Referer HTTP 头的请求
--without-http_rewrite_module	通过 rewrite 模块能够让你基于变量条件改变 URI
--without-http_proxy_module	使用 proxy 模块允许 Nginx 将请求传递到其他服务器或者服务器组
--without-http_fastcgi_module	FastCGI 模块能够让Nginx 将请求传递到FastCGI 服务器
--without-http_uwsgi_module	该模块能够让 Nginx 将请求传递到 uWSGI 服务器
--without-http_scgi_module	SCGI 模块能够让 Nginx 将请求传递到 SCGI 服务器
--without-http_memcached_module	该模块能够使得 Nginx 与一个 memcached 服务器进行交互，将响应放置到变量查询中
--without-http_limit_conn_module	该模块能够使得 Nginx 基于某些键，通常是 IP 地址，设置连接限制
--without-http_limit_req_module	通过该模块，Nginx 能够限制每个用户的请求率
--without-http_empty_gif_module	在内存中空的 GIF 模块产生一个 1 像素×1 像素的透明 GIF 图像
--without-http_browser_module	browser 模块允许基于 User-Agent HTTP 请求头配置，变量的设置基于在该头中发现的版本
--without-http_upstream_ip_hash_module	该模块定义了一组可以与不同的代理模块结合使用的服务器

1.5 查找并安装第三方模块

由于有多个开源项目，所以在 Nginx 周围就会有一个活跃的开发社区。由于 Nginx 的模块化特性，这个社区能够开发和发布模块，从而为 Nginx 提供额外的功能。它们涵盖了广泛的应用，所以着手开发自己的模块之前应该看看有什么可用模块。

安装第三方模块的过程相当简单，步骤如下。

1. 定位你想要使用的模块（在 https://github.com 或者是 http://wiki.nginx.org/3rdPartyModules 查找）。

2. 下载该模块。

3．解压缩源代码安装包。

4．如果有 README 文件，那么阅读 README 文件，查看在安装中是否有依赖安装。

5．通过./configure-add-module=<path>选项配置使用该模块。

这个过程会给你的 nginx 二进制文件与模块附加这个功能。

需要注意的是，很多第三方模块是实验性质的。因此，在将这些模块用于生产系统之前，首先要测试使用这些模块。另外请记住，Nginx 的开发版本中可能会有 API 的变化，会导致第三方模块出现问题。

1.6　添加对 Lua 的支持

特别应该提到的是 ngx_lua 这个第三方模块，ngx_lua 模块提供了启用 Lua 的功能，而不是像 Perl 一样在配置时嵌入式脚本语言。该模块对于 perl 模块来说最大的优点就是它的无阻塞性，并与其他第三方模块紧密集成。对于它的安装说明的完整描述详见：https://github.com/openresty/lua-nginx-module#installation。我们将以这个模块为例，在下一节中介绍如何安装第三方模块。

1.7　组合在一起

现在你已经大概了解了各种配置选项，接下来你可以根据自己的需要设计一个二进制文件。下面的例子中，指定了 prefix、user、group，某些路径禁用了某些模块，启用了一些其他模块，并包括一些第三方模块。

```
$ export BUILD_DIR='pwd'
$ export NGINX_INSTALLDIR=/opt/nginx
$ export VAR_DIR=/home/www/tmp
$ export LUAJIT_LIB=/opt/luajit/lib
$ export LUAJIT_INC=/opt/luajit/include/luajit-2.0


$ ./configure \
        --prefix=${NGINX_INSTALLDIR} \
        --user=www \
        --group=www \
        --http-client-body-temp-path=${VAR_DIR}/client_body_temp \
```

```
            --http-proxy-temp-path=${VAR_DIR}/proxy_temp \
            --http-fastcgi-temp-path=${VAR_DIR}/fastcgi_temp \
            --without-http_uwsgi_module \
            --without-http_scgi_module \
            --without-http_browser_module \
            --with-openssl=${BUILD_DIR}/../openssl-1.0.1p \
            --with-pcre=${BUILD_DIR}/../pcre-8.32 \
            --with-http_ssl_module \
            --with-http_realip_module \
            --with-http_sub_module \
            --with-http_flv_module \
            --with-http_gzip_static_module \
            --with-http_gunzip_module \
            --with-http_secure_link_module \
            --with-http_stub_status_module \
            --add-module=${BUILD_DIR}/ngx_devel_kit-0.2.17 \
            --add-module=${BUILD_DIR}/ngx_lua-0.7.9
```

接下来，跟随的大量输出显示了在你的系统上能找到什么样的配置，概要打印出来，配置如下所示。

```
    Configuration summary
      + using PCRE library: /home/builder/build/pcre-8.32
      + using OpenSSL library: /home/builder/build/openssl-1.0.1p
      + md5: using OpenSSL library
      + sha1: using OpenSSL library
      + using system zlib library

      nginx path prefix: "/opt/nginx"
      nginx binary file: "/opt/nginx/sbin/nginx"
      nginx configuration prefix: "/ opt/nginx/conf"
      nginx configuration file: "/opt/nginx/conf/nginx.conf"
      nginx pid file: "/ opt/nginx/logs/nginx.pid"
      nginx error log file: "/opt/nginx/logs/error.log"
      nginx http access log file: "/opt/nginx/logs/access.log"
      nginx http client request body temporary files: "/home/www/tmp/ client_
body_temp"
      nginx http proxy temporary files: "/home/www/tmp/proxy_temp"
      nginx http fastcgi temporary files: "/home/www/tmp/fastcgi_temp"
```

如上所示，configure 找到了所有我们要查找的条目，并且按照我们的喜好设置了路径。现在，你可以构建你的 nginx 并安装它，正如本章一开始提到的。

1.8　小结

本章介绍了各种 Nginx 的有效模块，通过编译你自己的二进制文件，你可以定制 Nginx 能够为你提供哪些功能。对于你来说，构建和安装软件应该不会陌生。所以，创造一个构建环境或者确保所有依赖关系都存在，这并不会花费你很多的时间。一个 Nginx 的安装应该是按照你的需要，能随时启用或禁用模块，正如你看到的，启用或者是禁用一个模块应该感到很容易。

接下来，我们将介绍基本的 Nginx 配置概述，以便感受一下在通常情况下 Nginx 是如何配置的。

第 2 章
配置指南

Nginx 配置文件的格式非常合乎逻辑。学习这种格式以及如何使用每个部分是基础，这将有助于你手工创建一个配置文件。构造配置涉及为每个单独的段指定全局参数和指令。这些指令以及它们如何适应整个配置文件是本章的主要内容。目标是了解如何创建正确的配置文件以满足你的需求。

通过这一章的讨论话题，帮助你达到如下目标。

- 基本配置格式。
- Nginx 全局配置参数。
- 使用 include 文件。
- HTTP 的 server 部分。
- 虚拟服务器部分。
- location——where, when, how。
- mail 的 server 部分。
- 完整的示例配置文件。

2.1 基本配置格式

基本的 Nginx 配置文件由若干个部分组成，每一个部分都是通过下列方法定义的。

```
<section> {

    <directive> <parameters>;
```

```
}
```

需要注意的是，每一个指令行都由分号结束（;），这标记着一行的结束。大括号（{}）实际上表示一个新配置的上下文（context），但是在大多数情况下，我们将它们作为“节、部分（section）”来读。

2.2　Nginx 全局配置参数

全局配置部分被用于配置对整个 server 都有效的参数和前一个章节中的例外格式。全局部分可能包含配置指令，例如，user 和 worker_processes，也包括“节、部分（section）”。例如，events，这里没有大括号（{}）包围全局部分。

在全局部分中，最重要的配置指令都在表 2-1 中，这些配置指令将会是你处理的最重要部分。

表 2-1　　全局配置指令

全局配置指令	说明
user	使用这个参数来配置 worker 进程的用户和组。如果忽略 group，那么 group 的名字等于该参数指定用户的用户组
worker_processes	指定 worker 进程启动的数量。这些进程用于处理客户的所有连接。选择一个正确的数量取决于服务器环境、磁盘子系统和网络基础设施。一个好的经验法则是设置该参数的值与 CPU 绑定的负载处理器核心的数量相同，并用 1.5～2 之间的数乘以这个数作为 I/O 密集型负载
error_log	error_log 是所有错误写入的文件。如果在其他区段中没有设置其他的 error_log，那么这个日志文件将会记录所有的错误。该指令的第二个参数指定了被记录错误的级别（debug、info、notice、warn、error、crit、alert、emerg）。注意，debug 级别的错误只有在编译时配置了 --with-debug 选项才可以使用
pid	设置记录主进程 ID 的文件，这个配置将会覆盖编译时的默认配置
use	该指令用于指示使用什么样的连接方法。这个配置将会覆盖编译时的默认配置，如果配置该指令，那么需要一个 events 区段。通常不需要覆盖，除非是当编译时的默认值随着时间的推移产生错误时才需要被覆盖设置
worker_connections	该指令配置一个工作进程能够接受并发连接的最大数。这个连接包括客户连接和向上游服务器的连接，但并不限于此。这对于反向代理服务器尤为重要，为了达到这个并发性连接数量，需要在操作系统层面进行一些额外调整

下面是一个使用这些指令的简短示例：

```
# we want nginx to run as user 'www'
user www;

# the load is CPU-bound and we have 12 cores
worker_processes  12;

# explicitly specifying the path to the mandatory error log
error_log   /var/log/nginx/error.log;

# also explicitly specifying the path to the pid file
pid         /var/run/nginx.pid;

# sets up a new configuration context for the 'events' module
events {

    # we're on a Solaris-based system and have determined that
      nginx
    # will stop responding to new requests over time with the
      default
    # connection-processing mechanism, so we switch to the
      second-best
    use /dev/poll;
    # the product of this number and the number of
      worker_processes
    # indicates how many simultaneous connections per IP:port pair
      are
    # accepted
    worker_connections   2048;
}
```

这一部分应该放置在 nginx.conf 配置文件的顶部。

2.3 使用 include 文件

在 Nginx 的配置文件中，include 文件可以在任何地方，以便增强配置文件的可读性，并且能够使得部分配置文件重新使用。使用 include 文件，要确保被包含的文件自身有正确的 Nginx 语法，即配置指令和块（blocks），然后指定这些文件的路径。

```
include /opt/local/etc/nginx/mime.types;
```

在路径中出现通配符，表示可以配置多个文件。

```
include /opt/local/etc/nginx/vhost/*.conf;
```

如果没有给定全路径，那么 Nginx 将会依据它的主配置文件路径进行搜索。Nginx 测试配置文件很容易，通过下面的命令来完成。

```
nginx -t -c <path-to-nginx.conf>
```

该命令将测试 Nginx 的配置文件，包括 include 文件，但是它只检查语法错误。

2.4　HTTP 的 server 部分

在 HTTP 中，server 部分或者 HTTP 配置 context 是可用的，除非在编译安装 Nginx 时没有包含 HTTP 模块（也就是使用了--without-http）。这部分控制了 HTTP 模块的方方面面，是使用最多的一个部分。

本部分的配置指令用于处理 HTTP 连接，因此，该模块提供了相当数量的指令。为了更容易理解这些指令，我们将它们划分为不同的类型来讲述。

2.4.1　客户端指令

如表 2-2 所示，这一组指令用于处理客户端连接本身的各个方面以及不同类型的客户端。

表 2-2　　http 客户端指令

http 客户端指令	说明
chunked_transfer_encoding	在发送给客户端的响应中，该指令允许禁用 http/1.1 标准的块传输编码
client_body_buffer_size	为了阻止临时文件写到磁盘，可以通过该指令为客户端请求体设置缓存大小，默认的缓存大小为两个内存页面
client_body_in_file_only	用于调试或者是进一步处理客户端请求体。该指令设置为“on”能够将客户端请求体强制写入到磁盘文件
client_body_in_single_buffer	为了减少复制的操作，使用该指令强制 Nginx 将整个客户端请求体保存在单个缓存中
client_body_temp_path	该指令定义一个命令路径用于保存客户端请求体
client_body_timeout	该指令指定客户体成功读取的两个操作之间的时间间隔
client_header_buffer_size	该指令为客户端请求头指定一个缓存大小，当请求头大于 1KB 时会用到这个设置
client_header_timeout	该超时是读取整个客户端头的时间长度

续表

http 客户端指令	说明
client_max_body_size	该指令定义允许最大的客户端请求头，如果大于这个设置，那么客户端将会是 413（Request Entity Too Large）错误
keepalive_disable	该指令对某些类型的客户端禁用 keep-alive 请求功能
keepalive_requests	该指令定义在一个 keep-alive 关闭之前可以接受多少个请求
keepalive_timeout	该指令指定 keep-alive 连接持续多久。第二个参数也可以设置，用于在响应头中设置“keepalive”头
large_client_header_buffers	该指令定义最大数量和最大客户端请求头的大小
msie_padding	为了填充响应的大小至 512 字节，对于 MSIE 客户端，大于 400 的状态代码会被添加注释以便满足 512 字节，通过启用该命令可以阻止这种行为
msie_refresh	对于 MSIE 客户端，该指令可启用发送一个 refresh 头，而不是 redirect

2.4.2 文件 I/O 指令

这些指令用于控制 Nginx 如何投递静态文件以及如何管理文件描述符，如表 2-3 所示。

表 2-3 http 文件 I/O 指令

http 文件 I/O 指令	说明
aio	该指令启用异步文件 I/O。该指令对于现代版本的 FreeBSD 和所有 Linux 发行版都有效。在 FreeBSD 系统下，aio 可能被用于 sendfile 预加载数据。在 Linux 下，则需要 directio 指令，自动禁用 sendfile
directio	该指令用于启用操作系统特定的标志或者功能提供大于给定参数的文件。在 Linux 系统下，使用 aio 时需要使用该指令
directio_alignment	该指令设置 directio 的算法。默认值为 512，通常足够了，但是在 Linux 的 XFS 下推荐增加为 4KB
open_file_cache	该指令配置一个缓存用于存储打开的文件描述符、目录查询和文件查询错误
open_file_cache_errors	该指令按照 open_file_cache，启用文件查询错误缓存
open_file_cache_min_uses	open_file_cache 缓存的文件描述符保留在缓存中，使用该指令配置最少使用文件描述符的次数
open_file_cache_valid	该指令指定对 open_file_cache 缓存有效性检查的时间间隔
postpone_output	该指令指定 Nginx 发送给客户端最小的数值，如果可能的话，没有数据会发送，直到达到此值