Broadview
www.broadview.com.cn

cnode 社区原创

Kenneth Kan
Pixbi 联合创始人、CTO
何翊宇（dead_horse）
天猫前端技术专家

倾情力荐

Node.js 实战（第2季）

吴中骅 雷宗民 赵坤 刘亚中 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Node.js实战（第2季）

吴中骅 雷宗民 赵坤 刘亚中 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内容简介

本书通过7个实例分别讲解了Node.js在实战开发中的应用，这些章节既涉及Docker、Koa 等最新技术，也涉及OAuth2、命令行工具、消息队列、单元测试、编写C/C++模块等实战中经常会遇到的问题和应用场景。本书章节大体按照从简单到复杂的难度编排，每一章都通过一个实例指引读者从头开发一个Node.js应用，让读者循序渐进地学习Node.js，以及在实战开发中的编程技巧。本书不但着重讲解了每个实战案例所涉及的基础知识、思路和方法，也详细解释了源码的关键部分，希望有利于读者的学习和理解。

本书适合有一定Node.js基础及服务器端开发基础的读者阅读，也适合想了解Node.js可以做什么、想迅速上手实践的读者阅读。

图书在版编目（CIP）数据

Node.js实战. 第2季 / 吴中骅等著. —北京：电子工业出版社，2015.10
ISBN 978-7-121-27139-7

Ⅰ. ①N… Ⅱ. ①吴… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2015）第216018号

策划编辑：张国霞
责任编辑：徐津平
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　　邮编：100036
开　　本：720×1000　1/16　印张：19.25　字数：340千字
版　　次：2014年5月第1版
　　2015年10月第2版
印　　次：2015年10月第1次印刷
印　　数：3000册　　定价：59.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

推荐序

写一本与Node.js实战相关的书是一项非常有挑战性的任务。众所周知，Node.js 自身现在正处于一个飞速发展的阶段，它的第三方库同样正处于一个爆发增长期，各种新兴技术和框架层出不穷，在短短几年里，它的第三方库的数量就已经超越了Java。因此，要想完成这个任务，除了需要扎实的技术功底，还需要常人难及的耐心和表达能力，才能够跟进社区的快速发展，并将知识同步给读者。

本书的几位作者都是Node.js 领域的牛人，不论是在社区、个人博客还是在GitHub 上，都一直在孜孜不倦地分享与Node.js 实战相关的经验和文章。他们不仅有扎实的技术功底，对于如何分享自己所掌握的知识也非常有经验。

本书在内容上同大部分技术入门书籍相比有所不同，它并没有花费太多篇幅来讲解具体的语言、框架等基础内容，而是通过7个实战项目来介绍Node.js，能够让读者直接感知到Node.js到底能做什么，以及怎样才能写出可靠的Node.js代码。这些项目既包含了Docker、Koa等最新技术，也包含了OAuth2、单元测试、消息队列等在实战中经常会遇到的问题和应用场景，每个项目都有详尽的源码，因此在读本书时可边读边实践，通过实际的项目来学习Node.js。如果你是一名Node.js新手，那么通过本书能够快速地掌握如何使用Node.js来搭建应用；如果你具备一定的Node.js开发经验，那么也可以从中学习到许多优秀的开发技巧。

翻开本书，打开编辑器，跟着这7个小项目，一步一步地开始深入了解Node.js的世界吧！

何翊宇（dead_horse），天猫前端技术专家

前　言

自本书第1季《Node.js实战（双色）》出版以来，JavaScript界又发生了许多重大事件：React.js和AngularJS 2.0相继出现；ES6于2015年年中正式定稿；io.js从Node.js社区中分裂出来，后又与其合并。截至本书出版时，npm上有接近18万的模块，是去年同期的3倍，周下载量接近5.6亿次，是2014年同期的10倍。越来越多的创业公司和大公司都不同程度地使用了Node.js， Node.js已经成为一门成熟、稳定且具有独特魅力的技术。

延续《Node.js实战（双色）》的写作思路，本书不会从头讲解Node.js是什么，而是面向有一定Node.js基础的读者，建议读者把本书当作入门与进阶之间的过渡书籍来阅读。本书通过7个实例来讲解Node.js在不同场景下的应用，通过阅读本书，读者可以快速熟悉并使用Node.js进行开发。本书由4位作者共同编写，其中吴中骅完成了第1、3章的创作，雷宗民完成了第2、4章的创作，赵坤完成了第5章的创作，刘亚中完成了第6、7章的创作。

第1章主要介绍了如何使用Docker快速发布一个Nginx+Express+Redis项目，然后使用Jenkins进行简单的持续集成发布工作，其中介绍了Docker的基础概念、用法和Jenkins的安装配置方法。

第2章介绍了当前比较流行的OAuth2认证。OAuth2认证是当API服务器对外提供服务时，验证API使用者权限的有效的认证方式。本章主要介绍了使用Node.js搭建一个基本的API服务器所涉及的组件、方法和技术细节。

第3章主要介绍了如何在Node.js中使用消息队列软件RabbitMQ来解决Web服务器或应用服务器间的通信问题。对于服务器间的跨语言通信，以前

一般采用XMLRPC方式，而现在比较流行采用HTTP的RESTful方式，使用RabbitMQ能够很灵活地处理这些事情。

第4章以一个静态博客系统构建工具作为实例，介绍如何使用Node.js的commander模块来编写一个命令行工具。

第5章介绍了ES6中生成器、yield，以及Node.js下一代Web开发框架Koa及其中间件的用法，最后通过搭建一个简单的论坛系统，让读者从实践中学习如何基于Koa快速开发Web应用。

第6章分享了作者在一家时尚杂志相关的互联网创业团队工作的部分经历：为该团队搭建一套较基础的Node.js测试服务，测试范围涵盖服务器、浏览器、Mag+、Adobe InDesign等平台。希望读者通过这个分享能够找到一种更为全面的方式去保证项目代码的质量。

第7章分享了作者与51Degrees团队远程工作的细节，可以让读者大致了解如何将一个已有的C/C++代码库拓展到Node.js平台上进行使用。

感谢电子工业出版社博文视点张国霞编辑的热心帮助和指导；感谢神猪、河蟹、AKI、何翊宇（dead_horse）、Kenneth Kan、Baptiste Gaillard、Jorn Zaefferer和Fish对本书的校验；再次感谢何翊宇为本书慷慨作序。

如果您对本书有任何评论或建议，可加入本书官方QQ群（156627943）进行讨论，或者到读者反馈网站http://nodejs.ucdok.com进行反馈，真诚欢迎您的意见与反馈。

目　录

第1章
通过Docker快速发布Node.js应用

本章将主要介绍如何利用Docker快速发布一个Nginx+Express+Redis项目，然后使用Jenkins进行简单的持续集成发布工作，其中将介绍Docker的基础概念、用法和Jenkins的安装配置方法。Node.js配合Docker和Jenkins可以更加方便地管理我们的应用。

在学习本章之前，读者需要对Linux基本命令行操作、Nginx简单配置、Express框架、Redis有所了解。

1.1 什么是Docker

Docker在2013年首次进入业界的视线，其受到广泛关注则是在2014年的下半年。Docker 1.0自2014年6月公布后的短短几个月内，人气一路飙升。红帽公司在新的RHEL 7版本中增添了支持Docker的功能，IBM公开拥抱Docker和容器，亚马逊推出了EC2容器服务，就连公认的竞争对手VMware也宣布支持Docker。

2014年8月，Linux.com和The New Stack在于美国芝加哥举办的CloudOpen大会上公布了一项由550名从业者参与的调查结果。在最受欢迎的开源云项目

评选中，OpenStack位列第一，Docker位列第二。

业界人士认为，Docker技术在2015年将不会停留于“热度”层面，而会深入地走向部署和应用，因此也将会进一步激发不同开源技术与平台之间的碰撞和整合，最终推动开源及容器技术的向前发展。在中国，将会有更多的云厂商宣布支持Docker。

那么Docker到底是什么？我们该如何定义这个开源界的新宠呢？

Docker官网对Docker的定义如下：

“Docker是一个为开发者和运维管理员搭建的开放平台软件，可以在这个平台上创建、管理和运行生产应用。Docker Hub是一个云端服务，可以用它共享应用或自动化工作流。Docker能够从组件快速开发应用，并且可以轻松地创建开发环境、测试环境和生产环境。”

通俗地说，Docker是一个开源的应用容器引擎，可以让开发者打包自己的应用及依赖包到一个可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。Docker容器完全使用沙箱机制，独立于硬件、语言、框架、打包系统，相互之间不会有任何接口，几乎没有任何性能开销，便可以很容易地在机器和数据中心中运行。最重要的是，它不依赖于任何语言、框架或系统。

比如，作为一名开发者，在自己的电脑上开发应用程序时，一切都运行正常，但如果将其部署到其他环境中就可能不能正常工作了。由于开发者使用了自己喜欢的栈、开发语言和版本，所以把它们部署到新的环境如测试环境或生产环境时，就会出现问题。这时，运维工程师和开发者需要花费大量的时间、精力和财力，通过进行大量的沟通才能达成一致。但如果使用Docker进行开发，就可以将这一切封装到一个或者几个可相互通信的容器中，而这个容器自身就可以完成所有工作。之后，开发者只需将该容器部署到其他环境中即可。

其次，相对于虚拟机，由于Docker容器不必运行操作系统，所以其体积更小。底层的Linux容器已经被包含在内核当中，这意味着镜像体积非常小、

非常快。虚拟机的体积以GB为单位，需要一到两分钟的启动时间，而容器只以MB为单位，并且可以在几毫秒内启动。这可以加速开发进度，允许开发者轻松地移动容器。

此外，由于容器体积小，可以快速部署，所以有助于开发者进行超大规模的部署。相对于虚拟机，开发者可以使用更少的存储空间、内存和CPU，因为其在性能方面基本上不需要系统开销。

1.2　Nginx作为Node.js前端Web Server的作用

在开始Docker之旅前，我想先说明将Nginx放置在Node.js前端的作用。

对于Nginx想必大家都不会陌生，不过在这里我还是要不厌其烦地再介绍一下。

Nginx（发音同engine x）是一款由俄罗斯程序员Igor Sysoev开发的轻量级网页服务器、反向代理服务器及电子邮件（IMAP/POP3）代理服务器。起初是供俄国大型门户网站及搜索引擎Rambler（俄语为Рамблер）使用的，并因其稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。此软件在BSD-like协议下发行，可以在UNIX、GNU/Linux、BSD、Mac OS X、Solaris及Windows等操作系统中运行。

这里正是看重了Nginx出色的HTTP反向代理能力，才把它放置在Node.js前端，用来处理我们的各种需求。可能有读者不理解“反向代理”这个名词，笔者在这里稍作解释。

有反向代理就肯定有正向代理，对于正向代理我们接触得比较多，比如我们想访问一些国外的网站，可是由于某些原因无法正常打开该网站或者打开缓慢，这时我们通过香港的HTTP代理就可以正常访问一些国外网站了，在此，香港的这个HTTP代理就是正向代理。反向代理的情况正好相反，比如我们有一个对外的API服务api.nodeInAction.com，初期我们启动一台服务器、

一个Node.js进程就可以完成负载，但是随着后期访问量的加大，一个进程、一台服务器已经不能满足我们的需要了，这时Nginx就可以发挥自己反向代理的能力。我们可以在Nginx后端添加多个服务器或启动多个进程来分担访问压力。在这里，Nginx的作用就是反向代理。

理解了Nginx的反向代理原理后，现在说说为什么要把它放在Node.js的应用之前。其实，这样做大致有如下好处。

1. 静态文件性能

Node.js的静态文件处理性能受制于它的单线程异步I/O模型，注定了静态文件性能不会很好（所以在某些情况下，单线程异步I/O并不是性能的代名词）。在一台普通的4CPU服务器上，使用Nginx处理静态文件的性能基本上是纯Node.js的2倍以上，所以我们应该避免在生产环境下直接使用Node.js来处理静态文件。关于Node.js处理静态文件的更多内容，在《Node.js实战（双色）》中有详细的对比和介绍，欢迎读者阅读。

2. 反向代理规则

有时会存在反向代理服务器配置多样化的情况，有时我们希望配置较好的机器能够分担更多的压力，有时又因为session的关系，我们需要将同一来源IP的客户端全转发到同一个进程上，对于诸如此类的规则，使用Nginx的配置文件就可以简单实现。

3. 扩展性

Nginx可以加入许多扩展来帮助我们处理业务，最典型的就是加入Lua语言的扩展。胶水语言Lua赋予了Nginx复杂逻辑判断的能力，并且保持了一贯的高效性。例如我们有一个API服务，对访问会进行MD5签名或对同一客户端来源有访问频率限制，这部分代码是后端业务处理前必须通过的验证，具有卡口的作用。利用Lua扩展，我们就可以高效、简单地完成这个卡口。

4. 稳定性和转发性能

对Nginx的稳定性大家有目共睹，Nginx在同样的负载下，相比Node.js占

用的CPU和内存资源更少。同时，高效地转发性能、便捷地转发配置也是我们选择它作为反向代理的原因之一，比如我们可以根据不同的URL请求路径转发到不同的后端机器上，也可以设定超时时间等，方便管理。

5. 安全性

Nginx已经被各大互联网公司广泛应用，经过一些配置可以有效抵挡类似slowloris等的DoS攻击，而Node.js在这方面做得还不够，关于Node.js开发安全方面的更多内容，可以参考《Node.js实战（双色）》一书，其中的第8章专门讨论了如何更安全地开发Node.js的Web应用。

6. 运维管理

如果我们目前只有一台Web服务器，同时有多个站点需要占用80端口，这时我们只需让Node.js服务监听本地的特殊端口如3000，通过Nginx的反向代理配置，就可以将多个站点域名指向一台机器了。当然，如果公司配置了专门的运维部门来管理服务器，相信他们对Nginx的熟悉程度一定远远大于Node.js，他们自己就可以轻松地修改一些配置，而不用来麻烦我们了。

所以，一个好习惯就是，在生产环境中，永远把Nginx放置在Node.js的前端，对性能、安全性和将来的扩展都有益处。

1.3　安装Docker和下载Images镜像

在介绍完Nginx之后，我们继续Docker之旅，在Docker官网有详细的各个系统安装流程，这里介绍CentOS下的安装流程，其他系统的安装地址为https://docs.docker.com/。

对于使用CentOS 7的用户，直接运行如下命令，就可以安装最新版本的Docker。

```
$ sudo yum install docker
```

使用CentOS 6.5的用户需要先获取epel源并导入。

```
$ wget -c http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
$ rpm -ivh epel-release-6-8.noarch.rpm
$ rpm -import /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```

接着通过yum安装Docker。

```
$ yum install docker-io --enablerepo=epel
```

启动Docker服务，并且把Docker服务注册为开机启动。

```
$ sudo service docker start
$ sudo chkconfig docker on
```

我们可以输入如下命令，检查Docker进程是否已经启动。

```
$ ps -ef|grep docker
```

如果发现Docker进程未成功启动，就需要进入/var/log/目录下查看Docker日志文件的信息了，应用CentOS 6.5的用户可能会发现系统报出如下错误。

```
/usr/bin/docker: relocation error: /usr/bin/docker: symbol dm_task_get_info_with_deferred_remove, version Base not defined in file libdevmapper.so.1.02 with link time reference
```

执行如下命令可以修复，然后重新启动Docker服务。

```
$ yum-config-manager --enable public_ol6_latest
$ yum install -y device-mapper-event-libs
$ yum update -y device-mapper-libs
```

现在，Docker服务已经安装并启动了，我们需要下载Image镜像，镜像就是我们应用运行的环境，比如我们可以自己安装好Node.js和npm，然后发布到Docker Hub上供自己或者别人下载，我们也可以下载安装一些官方镜像，把它作为自己镜像的基础。下面我们先下载CentOS镜像。

```
$ sudo docker pull centos:7
```