关于本文档

文档名称：《高性能WEB服务NGINX》

使用协议：《知识共享公共许可协议(CCPL)》

贡献者

贡献者名称	贡献度	文档变更记录	个人主页
马哥（马永亮）	主编		http://github.com/iKubernetes/
杰哥（张士杰）	创始作者	97页	http://blogs.studylinux.net/
王晓春	版本升级	247页,版本迭代升级,修订和内容丰富	http://www.wangxiaochun.com

文档协议

致谢

本文档中，部分素材参考了相关项目的文档，以及通过搜索引擎获得的内容，这里先一并向相关的贡献者表示感谢。

高性能WEB服务NGINX

内容概述

- **Web 服务基础**
- **HTTP 协议**
- **I/O 模型**
- **nginx 介绍和架构**
- **nginx 安装**
- **nginx 各种模块实现web服务**
- **nginx 实现http负载均衡**
- **nginx 实现tcp反向代理**
- **nginx 实现fastcgi反向代理**
- **nginx 二次开发版**
- **nginx 高并发Linux内核优化**

1 WEB 服务和HTTP协议

1.1 Internet 因特网

因特网是"Internet”的中文译名，它起源于美国的五角大楼，它的前身是美国国防部高级研究计划局（ARPA）主持研制的ARPAnet。20世纪50年代末，正处于冷战时期。当时美国军方为了自己的计算机网络在受到袭击时,即使部分网络被摧毁，其余部分仍能保持通信联系，便由美国国防部的高级研究计划局（ARPA）建设了一个军用网，叫做"阿帕网”（ARPAnet）。

1969年正式启用阿帕网，当时仅连接了4台计算机，供科学家们进行计算机联网实验用，这就是因特网的前身。
到70年代，ARPAnet已经有了好几十个计算机网络，但是每个网络只能在网络内部的计算机之间互联通信，不同计算机网络之间仍然不能互通。为此， ARPA又设立了新的研究项目，支持学术界和工业界进行有关的研究，研究的主要内容就是想用一种新的方法将不同的计算机局域网互联，形成"互联网”。研究人员称之为"internetwork”，简称"Internet”。在研究实现互联的过程中，计算机软件起了主要的作用。

1974年，出现了连接分组网络的协议，其中就包括了TCP/IP协议。TCP/IP有一个非常重要的特点，就是开放性，即TCP/IP的规范和Internet的技术都是公开的。目的就是使任何厂家生产的计算机都能相互通信，使Internet成为一个开放的系统，这正是后来Internet得到飞速发展的重要原因。ARPA在1982年接受了TCP/IP，选定Internet为主要的计算机通信系统，并把其它的军用计算机网络都转换到TCP/IP。
1983年，ARPAnet分成两部分：一部分军用，称为MILNET；另一部分仍称ARPAnet，供民用。

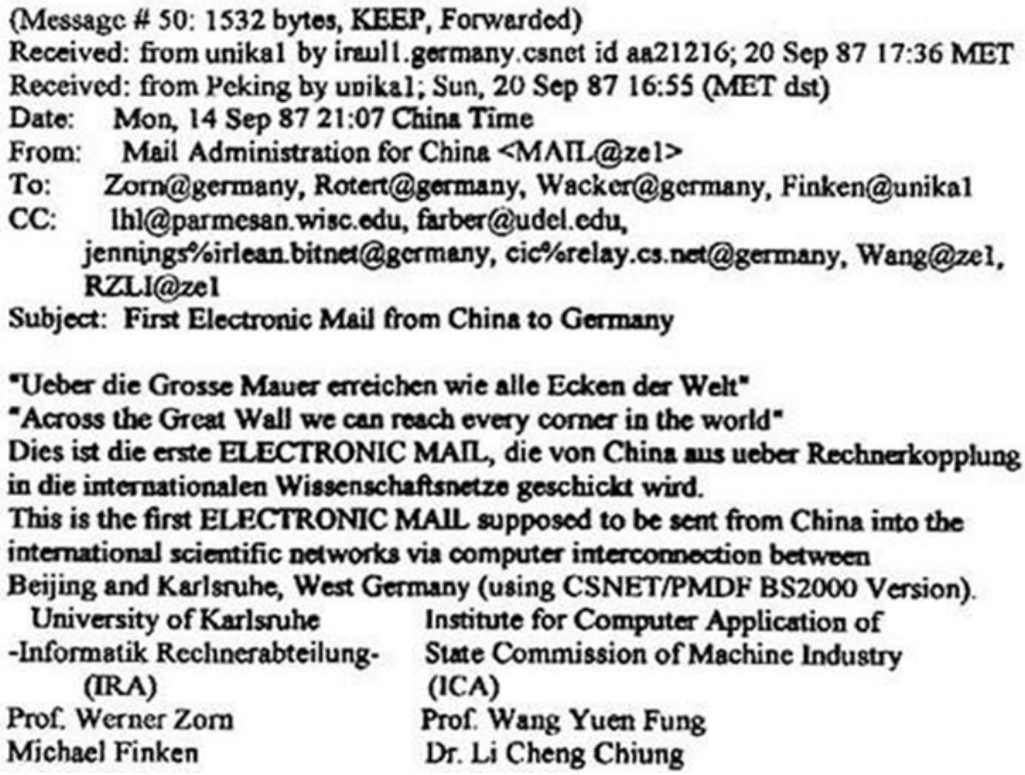
1986年，美国国家科学基金组织（NSF）将分布在美国各地的5个为科研教育服务的超级计算机中心互联，并支持地区网络，形成SNSFnet。1988 年，SNSFnet替代ARPAnet成为Internet的主干网。NSFnet主干网利用了在ARPAnet中已证明是非常成功的TCP/IP技术，准许各大学、政府或私人科研机构的网络加入。1989年，ARPAnet解散，Internet从军用转向民用。

Internet的发展引起了商家的极大兴趣。1992年，美国IBM、MCI、MERIT三家公司联合组建了一个高级网络服务公司（SNS），建立了一个新的网络，叫做SNSnet，成为Internet的另一个主干网。它与SNSFnet不同，NSFnet是由国家出资建立的，而SNSnet则是SNS 公司所有，从而使Internet开始走向商业化。
1995年4月30日，SNSFnet正式宣布停止运作。而此时Internet的骨干网已经覆盖了全球91个国家，主机已超过400万台。而在当前，因特网仍以惊人的速度向前发展

在90年代，超文本标识语言（HTML），即一个可以获得因特网的图像信息的超文本因特网协议被采用，使每一个人可以产生自己的图像页面（网址），然后成为一个巨大的虚拟超文本网络的组成部分。这个增强型的因特网又被非正式地称为万维网，与此同时产生了数量庞大的新用户群。于是，许多人用"因特网" 一词指这个网络的物理结构，包括连接所有事物的客户机、服务器和网络；而用"万维网"一词指利用这个网络可以访问的所有网站和信息。

1.2 Internet 和中国

北京时间1987年9月14日，物理研究员钱天白建立起一个网络节点，通过电话拨号连接到国际互联网，向他的德国朋友发出来自中国的第一封电子邮件：Across the Great Wall we can reach every corner in the world，自此，中国与国际计算机网络开始连接在一起



(Message # 50: 1532 bytes, KEEP, Forwarded)
Received: from unika1 by irau11.germany.csnet id aa21216; 20 Sep 87 17:36 MET
Received: from Peking by unika1; Sun, 20 Sep 87 16:55 (MET dst)
Date: Mon, 14 Sep 87 21:07 China Time
From: Mail Administration for China <MAIL@ze1>
To: Zorn@germany, Rotert@germany, Wacker@germany, Finken@unika1
CC: lhl@parmesan.wisc.edu, farber@udel.edu,
jennings%irlean.bitnet@germany, cic%relay.cs.net@germany, Wang@ze1,
RZLI@ze1
Subject: First Electronic Mail from China to Germany

"Ueber die Grosse Mauer erreichen wie alle Ecken der Welt"
"Across the Great Wall we can reach every corner in the world"
Dies ist die erste ELECTRONIC MAIL, die von China aus ueber Rechnerkopplung in die internationalen Wissenschaftsnetze geschickt wird.
This is the first ELECTRONIC MAIL supposed to be sent from China into the international scientific networks via computer interconnection between Beijing and Karlsruhe, West Germany (using CSNET/PMDF BS2000 Version).

University of Karlsruhe	Institute for Computer Application of
-Informatik Rechnerabteilung-	State Commission of Machine Industry
(IRA)	(ICA)
Prof. Werner Zorn	Prof. Wang Yuen Fung
Michael Finken	Dr. Li Cheng Chiung

1990年10月，钱天白教授代表中国正式在国际互联网络信息中心的前身DDN-NIC注册登记了我国的顶级域名CN，并且从此开通了使用中国顶级域名CN的国际电子邮件服务。由于当时中国尚未正式连入Internet，所以委托德国卡尔斯鲁厄大学运行CN域名服务器

1993年3月2日， 中国科学院高能物理研究所租用AT&T公司的国际卫星信道接入美国斯坦福线性加速器中心（SLAC）的64K专线正式开通,专线开通后，美国政府以Internet上有许多科技信息和其它各种资源，不能让社会主义国家接入为由，只允许这条专线进入美国能源网而不能连接到其它地方。尽管如此，这条专线仍是我国部分连入Internet的第一根专线

1994年4月20日，中国通过一条64k的国际专线全功能接入国际互联网，成为国际互联网大家庭中的第77个成员，正式开启了互联网时代。随后，中科院高能物理研究所推出第一个WWW网站和第一套网页

1994年5月21日， 在钱天白教授和德国卡尔斯鲁厄大学的协助下，中国科学院计算机网络信息中心完成了中国国家顶级域名(CN)服务器的设置，改变了中国CN顶级域名服务器一直放在国外的历史

1995年5月17日， 第27个世界电信日，邮电部正式宣布，向国内社会开放计算机互联网接入服务

1995年5月，北京的中关村南大街上出现了一块巨大的广告牌，"中国离信息高速公路还有多远？向北1500米。”那个位置就是一家叫"瀛海威”的网络科教馆，瀛海威正是information highway的音译，作为中国第一个互联网接入服务商，瀛海威几乎就是当时互联网的代名词

1996年1月，中国互联网全国骨干网建成并正式开通，开始提供服务

1995年4月，马云凑了两万块钱，成立杭州海博网络公司，专门给企业做主页

1997年5月，丁磊创立网易

1998年2月， 张朝阳创立搜狐

1998年6月18日，刘强东在中关村创办京东公司，代理销售光磁产品

1998年11月，马化腾和张志东成立深圳市腾讯计算机系统有限公司，OICQ开通

1998年12月， 新浪网成立，关键人物：王志东

1999年5月18日，中国第一家电子商务企业8848.com成立，创始人王峻涛也曾被誉为"中国电子商务教父”。2000年底，调查显示接近70%的人说上网买东西首选 8848

2000年1月，李彦宏创建了百度

2003年5月，阿里巴巴集团在创立淘宝网

2003年10月，淘宝网首次推出支付宝服务

2009年11月, 阿里开始举办双十一购物狂欢节

2012年1月11日淘宝商城正式更名为“天猫”

2014年9月19日阿里巴巴集团于纽约证券交易所正式挂牌上市

2004年1月，京东多媒体网正式开通，启用域名www.jdlaser.com

2010年4月，雷军创办小米

2011年1月21日，腾讯公司推出微信 (WeChat)

2012年3月，今日头条由张一鸣于创建

2012年7月10日，北京小桔科技有限公司成立，滴滴司机端3个月后北京上线

2016年9月20日, 字节跳动上线抖音

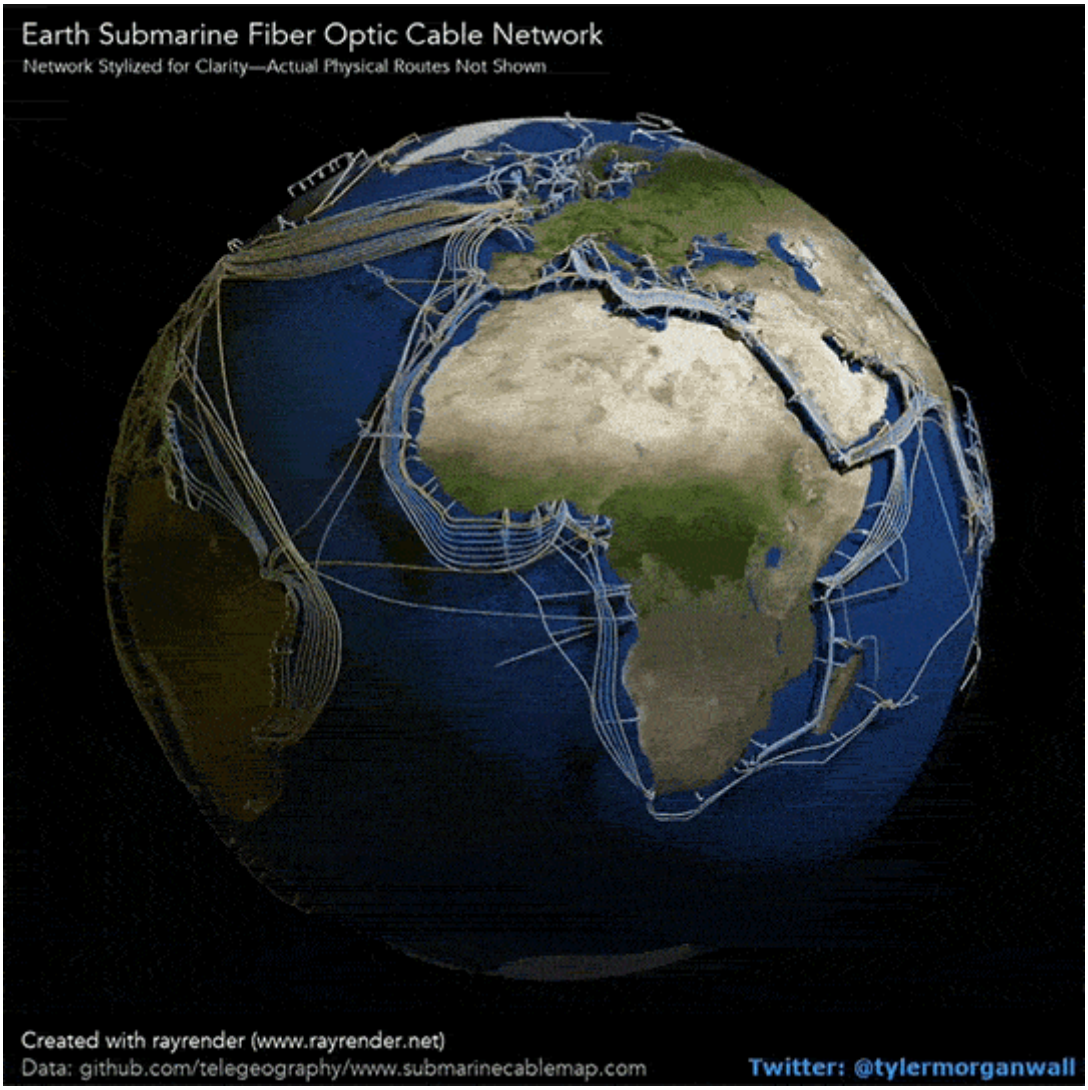
中国互联网连接世界

1885年台湾建省，首任巡抚刘铭传派人与福州船政联系，使用船政电报学堂毕业生为技术人员，于1887年铺设成功台湾淡水至福州川石海底电缆，全长117海里。这是我国自行设计安装的第一条海底电缆。此电缆毁于第二次世界大战我国于1989年开始投入到全球海底光缆的投资与建设中来，并于1993年实现了首条国际海底光缆的登陆（中日之间C-J海底光缆系统）；随后在1997年，我国参与建设的全球海底光缆系统（FLAG）建成并投入运营，这也是第一条在我国登陆的洲际海底光缆中国连接世界目前共有8条光缆，四个登陆站允许入境，目前我国的登陆站设立在三个城市的四个地区，分别是山东青岛登陆站（隶属中国联通）、上海崇明登陆站（隶属中国电信）、上海南汇登陆站（隶属中国联通）和广东汕头登陆站（隶属中国电信）

1993年3月2日，中国科学院高能物理研究所租用AT&T公司的国际卫星信道建立的接入美国SLAC国家实验室的64K专线正式开通，成为我国连入Internet的第一根专线。

参考链接: http://www.ihep.cas.cn/kxcb/kpcg/jsywl/201407/t20140714_4156699.html

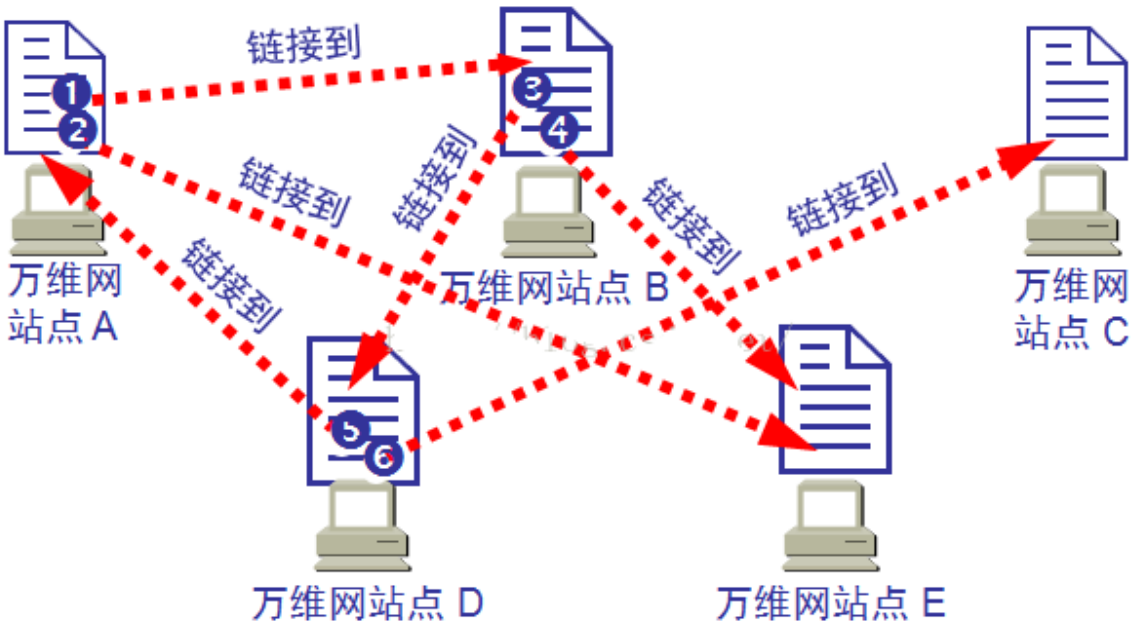
下图是海底光缆



1.3 HTTP 超文本传输协议

1.3.1 HTTP 相关概念

互联网：是网络的网络，是所有类型网络的母集。
因特网：世界上最大的互联网网络。即因特网概念从属于互联网概念。习惯上，大家把连接在因特网上的计算机都成为主机。
万维网：WWW（world wide web）万维网并非某种特殊的计算机网络，是一个大规模的、联机式的信息贮藏库，使用链接的方法能非常方便地从因特网上的一个站点访问另一个站点（超链技术），具有提供分布式服务的特点。万维网是一个分布式的超媒体系统，是超文本系统的扩充，基于B/S架构实现

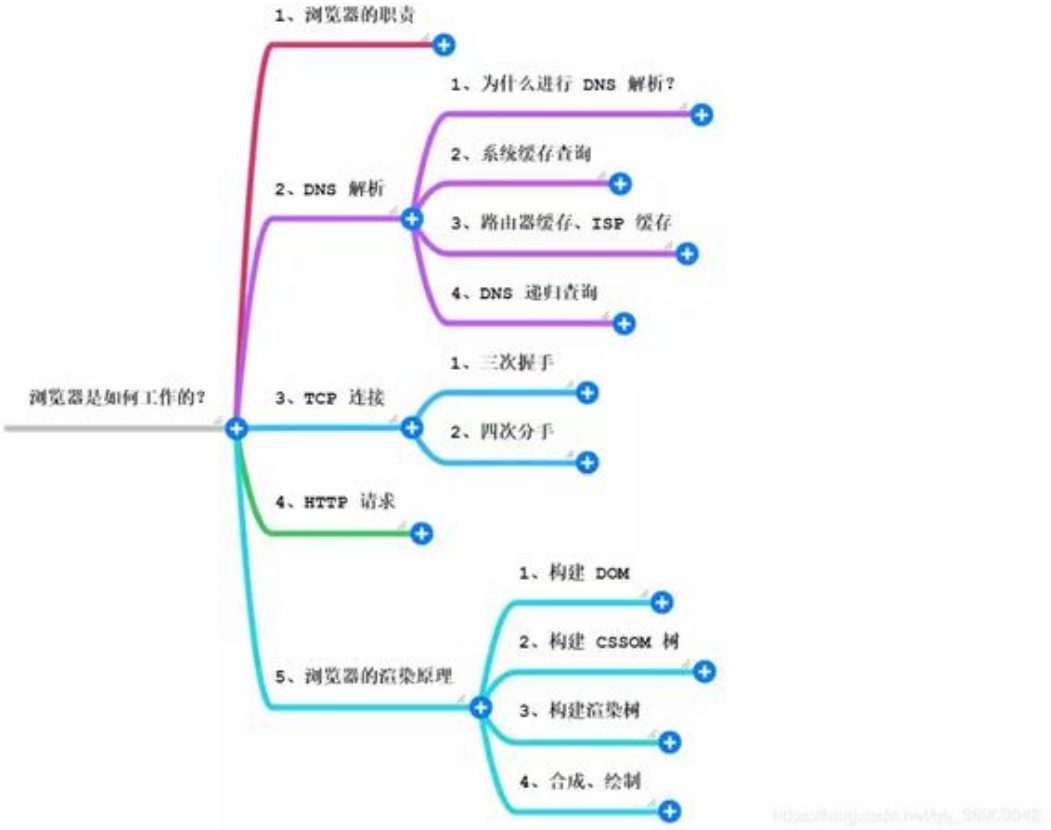


URL：万维网使用统一资源定位符（Uniform Resource Locator）来标志万维网上的各种文档，并使每个文档在整个因特网的范围内具有唯一的标识符URL。

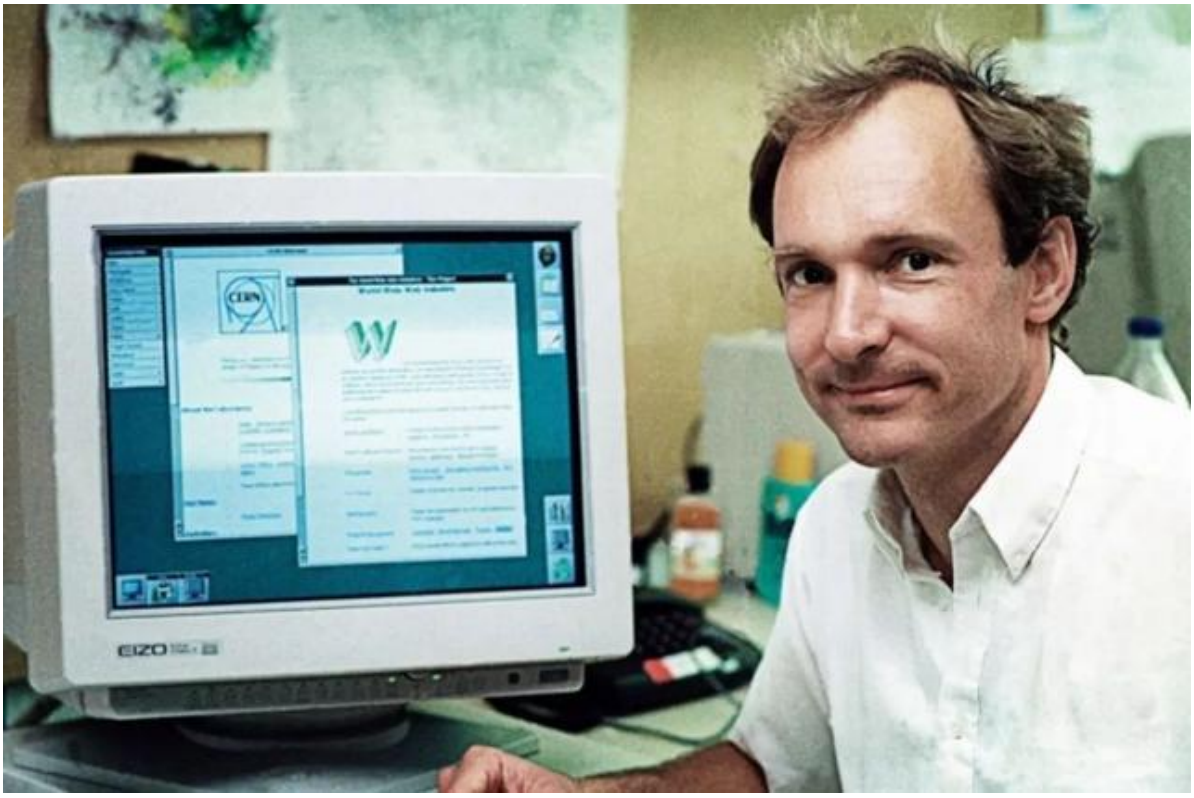
HTTP：为解决"用什么样的网络协议来实现整个因特网上的万维网文档"这一难题，就要使万维网客户程序（以浏览器为主，但不限于浏览器）与万维网服务器程序之间的交互遵守严格的协议，即超文本传送协议（HyperText Transfer Protocol）。HTTP是处于应用层的协议，使用TCP传输层协议进行可靠的传送。因此，需要特别提醒的是，万维网是基于因特网的一种广泛因特网应用系统，且万维网采用的是HTTP（80/TCP）和 HTTPS（443/TCP）的传输协议，但因特网还有其他的网络应用系统（如：FTP、SMTP等等）。

HTML：为了解决"怎样使不同作者创作的不同风格的万维网文档，都能在因特网上的各种主机上显示出来，同时使用户清楚地知道在什么地方存在着链接"这一问题，万维网使用超文本标记语言（HyperText Markup Language），使得万维网页面的设计者可以很方便地用链接从页面的某处链接到因特网的任何一个万维网页面，并且能够在自己的主机品目上将这些页面显示出来。HTML与txt一样，仅仅是是一种文档，不同之处在于，这种文档专供于浏览器上为浏览器用户提供统一的界面呈现的统一规约。且具备结构化的特征，这是txt所不具备的强制规定。

1.3.2 浏览器访问网站的过程



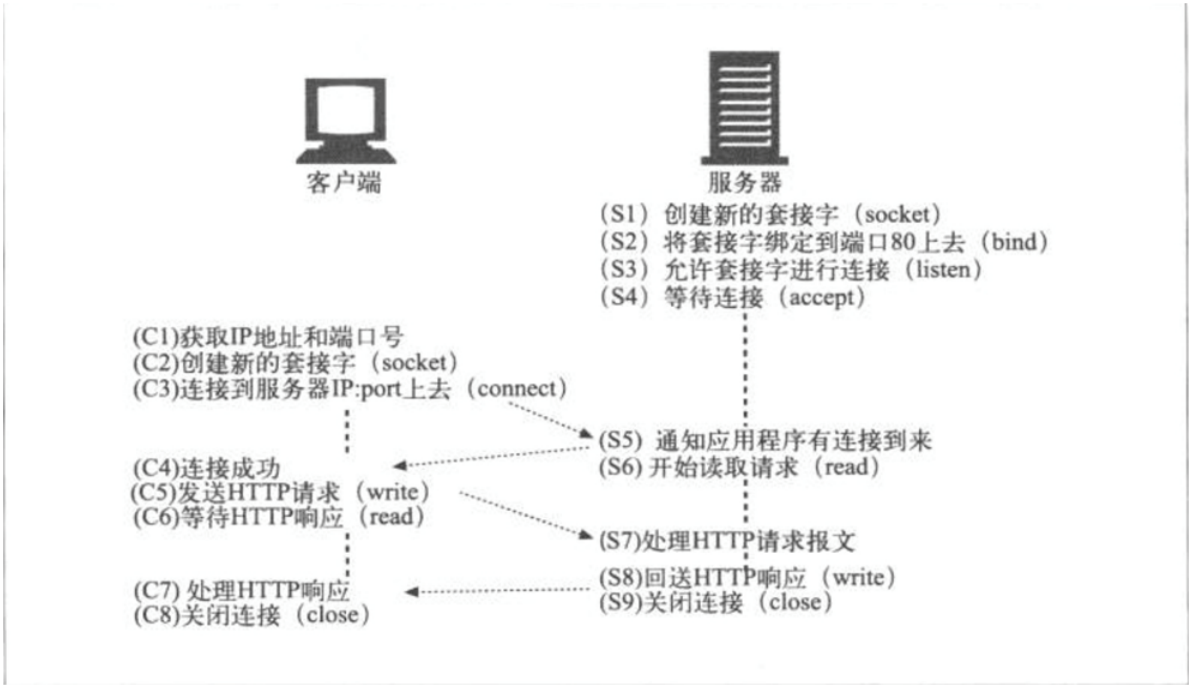
1.3.3 HTTP 协议通信过程



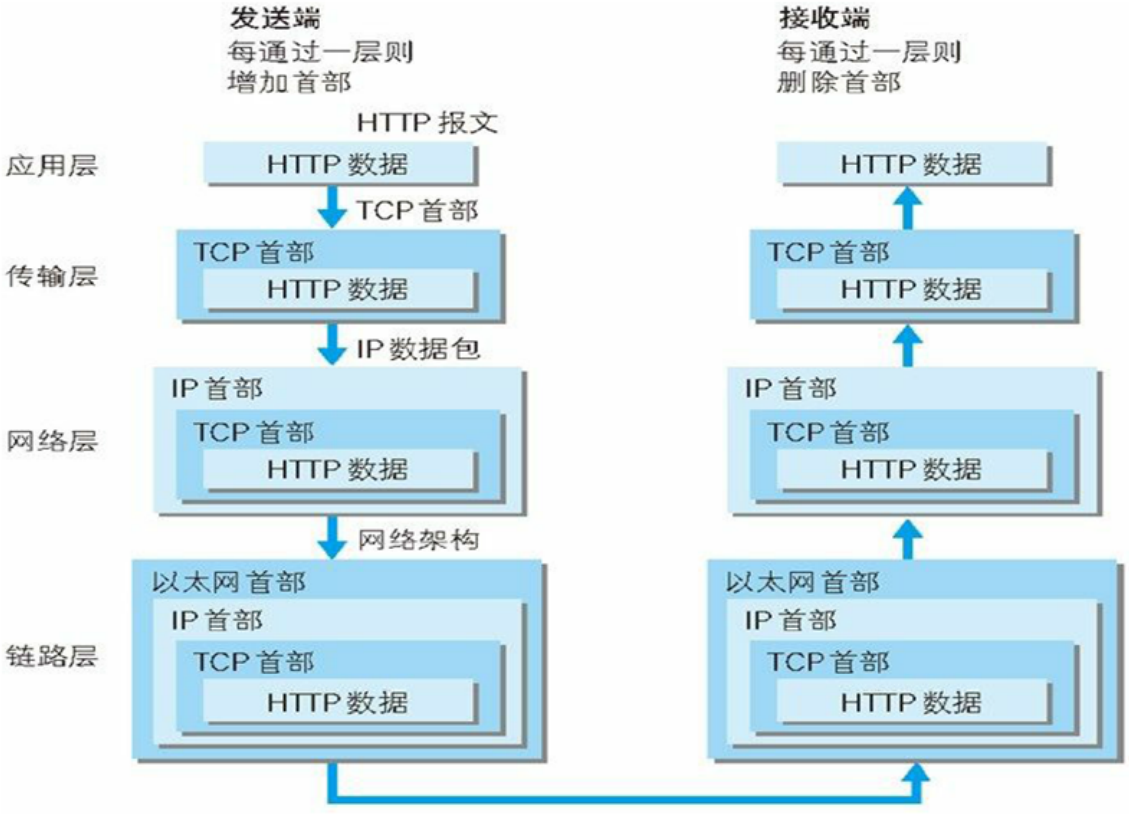
HTTP（HyperText Transfer Protocol，超文本传输协议）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP是万维网的数据通信的基础设计HTTP最初的目的是为了提供一种远距离共享知识的方式，借助多文档进行关联实现超文本，连成相互参阅的WWW（world wide web,万维网）

HTTP的发展是由蒂姆·伯纳斯-李（Tim Berners-Lee）于1989年在欧洲核子研究组织（CERN）所发起。HTTP的标准制定由万维网协会（World Wide Web Consortium，W3C）和互联网工程任务组（Internet Engineering Task Force，IETF）进行协调，最终发布了一系列的RFC，其中最著名的是1999年6月公布的 RFC 2616，定义了HTTP协议中现今广泛使用的一个版本——HTTP 1.1版

HTTP服务通信过程



HTTP协议分层



1.3.4 HTTP 相关技术

1.3.4.1 WEB 开发语言

http：Hyper Text Transfer Protocol 应用层协议，默认端口： 80/tcp

WEB前端开发语言：

- html
- css
- javascript

html

Hyper Text Markup Language 超文本标记语言，编程语言，主要负责实现页面的结构

范例：html 语言

```
<html>
<head>
<meta http-equiv=Content-Type content="text/html;charset=utf-8">
<title>HTML语言</title>
</head>
<body>
<img src="http://www.magedu.com/wp-content/uploads/2017/09/logo.png" >
<h1 style="color:red">欢迎</h1>
<p><a href=http://www.magedu.com>马哥教育</a>欢迎你</p>
</body>
</html>

#注意:html文件编码为utf-8编码
```

CSS

Cascading Style Sheet 层叠样式表， 定义了如何显示（装扮） HTML 元素，比如：字体大小和颜色属性等。样式通常保存在外部的 .css 文件中,用于存放一些HTML文件的公共属性,从而通过仅编辑一个简单的 CSS 文档，可以同时改变站点中所有页面的布局和外观。

范例： CSS

```
#test.html 建议用Vscode创建文件,用记事本可能会出现乱码
<html>
<head>
<meta http-equiv=Content-Type content="text/html;charset=utf-8">
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
<body>
<h1>这是 heading 1</h1>
<p>这是一段普通的段落。请注意，该段落的文本是红色的。在 body 选择器中定义了本页面中的默认文本颜色。</p>
<p class="ex">该段落定义了 class="ex"。该段落中的文本是蓝色的。</p>
</body>
</html>


#mystyle.css
body {color:red}
h1 {color:#00ff00}
p.ex {color:rgb(0,0,255)}
```

Js

javascript，实现网页的动画效果，但实属于静态资源

,Java和javascript的关系: 周杰和周杰伦的关系(N50牛康康语录)

范例：javascript

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv=Content-Type content="text/html;charset=utf-8">
</head>
<body>
<h2>我的第一个 JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
点击这里来显示日期和时间
</button>

<p id="demo"></p>
</body>
</html>
```

1.3.4.2 MIME

MIME : Multipurpose Internet Mail Extensions 多用途互联网邮件扩展

文件 `/etc/mime.types`,来自于mailcap包

MIME格式：type/subtype txt html jpg bmp

参考链接：

https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_Types

http://www.w3school.com.cn/media/media_mimeref.asp

范例：

```
text/plain    txt asc text pm el c h cc hh cxx hxx f90 conf log
text/html     html htm
text/css
image/jpeg    jpg jpeg
image/png
video/mp4
application/javascript
```

1.3.4.3 URI和URL

URI： Uniform Resource Identifier 统一资源标识，分为URL 和 URN

URN：Uniform Resource Naming，统一资源命名

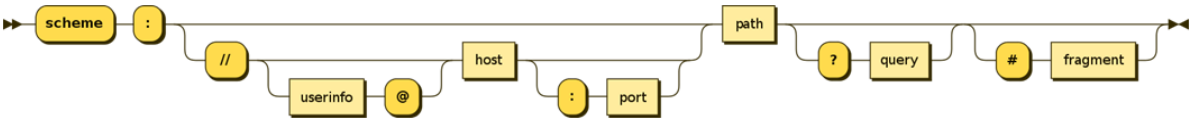
示例： P2P下载使用的磁力链接是URN的一种实现

magnet:?xt=urn:btih:660557A6890EF888666

URL：Uniform Resorce Locator，统一资源定位符，用于描述某服务器某特定资源位置

两者区别：URN如同一个人的名称，而URL代表一个人的住址。换言之，URN定义某事物的身份，而URL提供查找该事物的方法。URN仅用于命名，而不指定地址

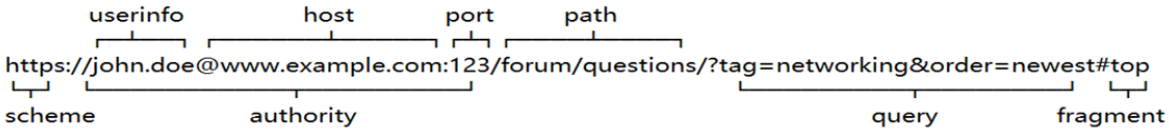
URL组成




```
<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag>
```

```
scheme:方案，访问服务器以获取资源时要使用哪种协议
user:用户，某些方案访问资源时需要的用户名
password:密码，用户对应的密码，中间用：分隔
Host:主机，资源宿主服务器的主机名或IP地址
port:端口,资源宿主服务器正在监听的端口号，很多方案有默认端口号
path:路径,服务器资源的本地名，由一个/将其与前面的URL组件分隔
params:参数，指定输入的参数，参数为名/值对，多个参数，用;分隔
query:查询，传递参数给程序，如数据库，用？分隔,多个查询用&分隔
frag:片段,一小片或一部分资源的名字，此组件在客户端使用，用#分隔
```

URL示例



```
http://www.magedu.com:8080/images/logo.jpg
ftp://mage:password@172.16.0.1/pub/linux.ppt
rtsp://videoserver/video_demo/   #Real Time Streaming Protocol
gcomm://10.0.0.8,10.0.0.18,10.0.0.28
http://www.magedu.com/bbs/hello;gender=f/send;type=title
https://list.jd.com/list.html?
cat=670,671,672&ev=14_2&sort=sort_totalsales15_desc&trans=1
http://apache.org/index.html#projects-list
```

1.3.4.4 网站访问量

网站访问量统计的重要指标

- IP(独立IP)：即Internet Protocol,指独立IP数。一天内来自相同客户机IP 地址只计算一次，记录远程客户机IP地址的计算机访问网站的次数，是衡量网站流量的重要指标
- PV(访问量)： 即Page View, 页面浏览量或点击量，用户每次刷新即被计算一次，PV反映的是浏览某网站的页面数，PV与来访者的数量成正比，PV并不是页面的来访者数量，而是网站被访问的页面数量
- UV(独立访客)：即Unique Visitor,访问网站的一台电脑为一个访客。一天内相同的客户端只被计算一次。可以理解成访问某网站的电脑的数量。网站判断来访电脑的身份是通过cookies实现的。如果更换了IP后但不清除cookies，再访问相同网站，该网站的统计中UV数是不变的

网站统计：http://www.alexa.cn/rank/

范例：网站访问统计

```
1. 甲乙丙三人在同一台通过 ADSL 上网的电脑上（中间没有断网），分别访问 www.magedu.com 网站，并且每人共用一个浏览器，各个浏览了2个页面，那么网站的流量统计是：
IP: 1   PV:6    UV: 1

2. 若三人都是ADSL重新拨号后,各个使用不同的浏览器，分别浏览了2个页面，则
IP: 3   PV:6    UV: 3
```

网站访问量

QPS：request per second，每秒请求数

PV，QPS和并发连接数换算公式

- QPS= PV * 页面衍生连接次数/ 统计时间（86400）
- 并发连接数 =QPS * http平均响应时间

峰值时间：每天80%的访问集中在20%的时间里，这20%时间为峰值时间

峰值时间每秒请求数(QPS)=(总PV数 *页面衍生连接次数）*80%) / (每天秒数 * 20%)

1.3.5 HTTP工作机制

一次http事务包括：

- http请求：http request
- http响应：http response

Web资源：web resource， 一个网页由多个资源（文件）构成，打开一个页面，通常会有多个资源展示出来，但是每个资源都要单独请求。因此，一个"Web 页面"通常并不是单个资源，而是一组资源的集合

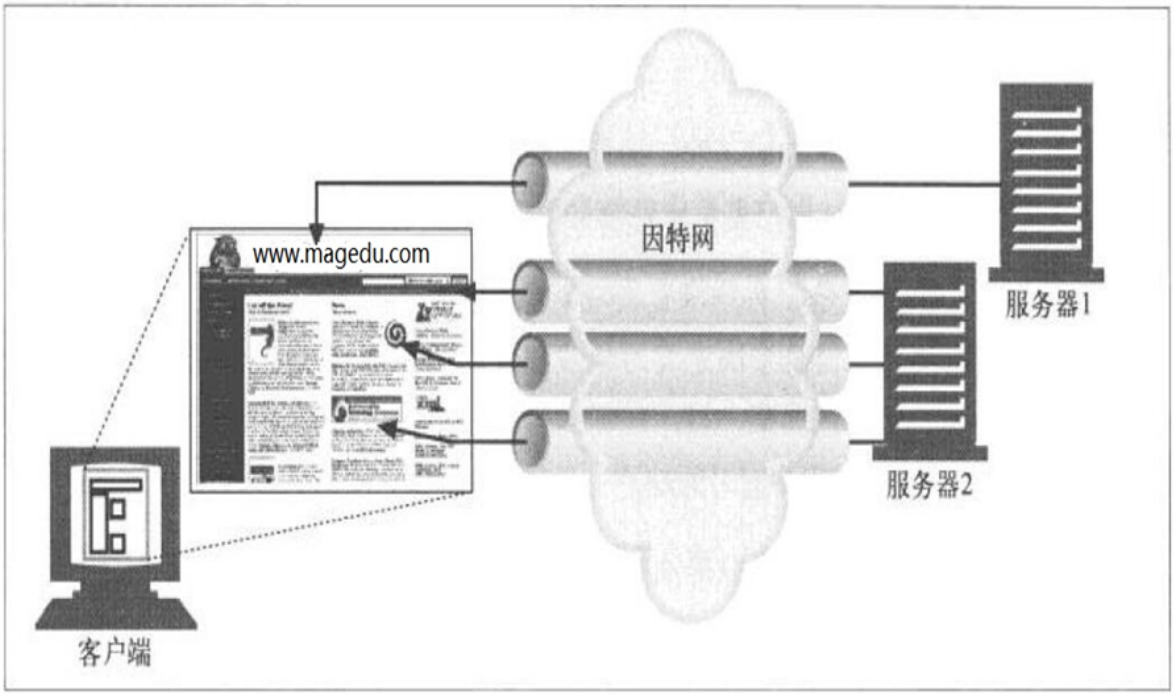
资源类型：

- 静态文件：无需服务端做出额外处理,服务器端和客户端的文件内容相同

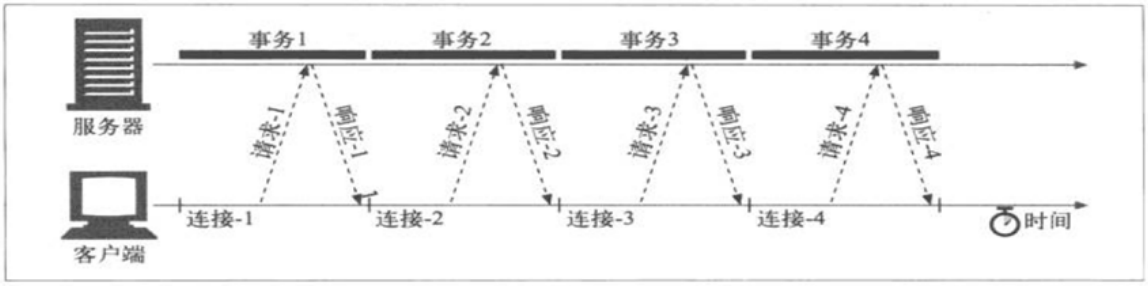
  常见文件后缀：.html, .txt, .jpg, .js, .css, .mp3, .avi
- 动态文件：服务端执行程序，返回执行的结果,服务器端和客户端的文件内容不相同

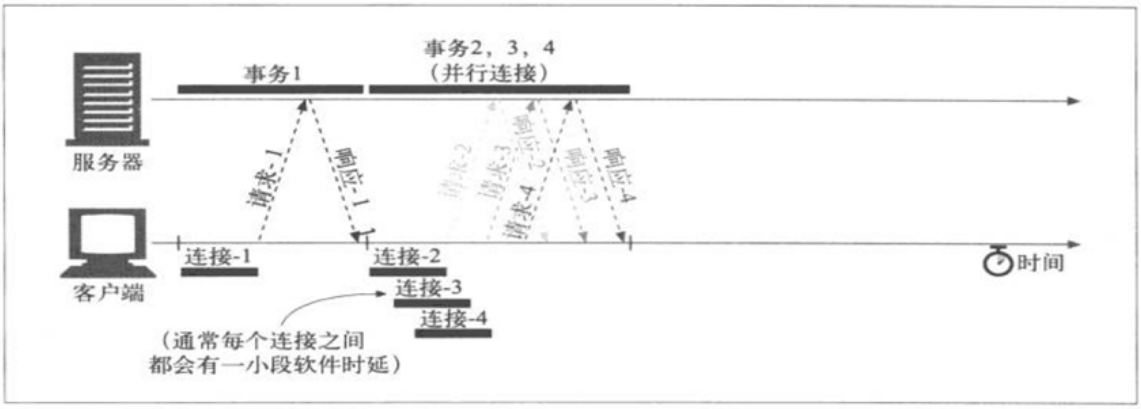
  常见文件后缀：.php, .jsp ,.asp

HTTP连接请求

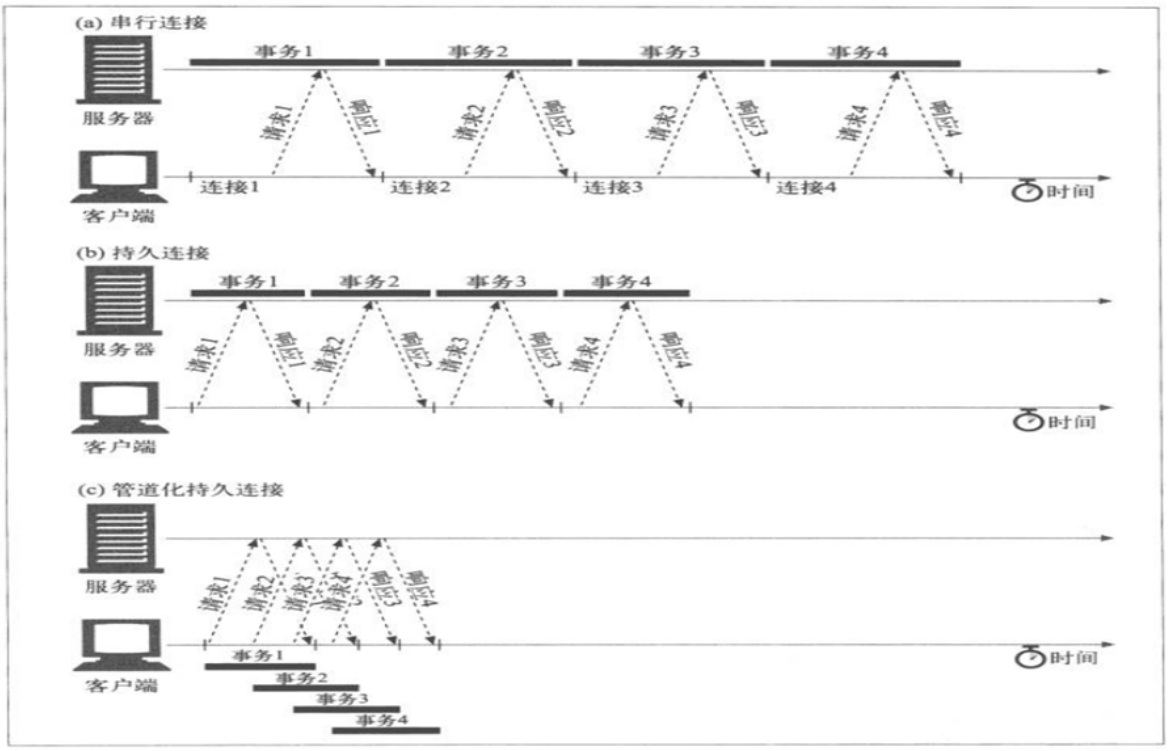


串行和并行连接





串行,持久连接和管道



提高HTTP连接性能

- 并行连接：通过多条TCP连接发起并发的HTTP请求
- 持久连接：keep-alive，重用TCP连接，以消除连接和关闭的时延,以事务个数和时间来决定是否关闭连接
- 管道化连接：通过共享TCP连接，发起并发的HTTP请求
- 复用的连接：交替传送请求和响应报文（实验阶段）

1.3.6 HTTP 协议版本

RFC Hypertext Transfer Protocol -- HTTP/1.1

```
https://tools.ietf.org/html/rfc2616
```

http/0.9:

1991，原型版本，功能简陋，只有一个命令GET。GET /index.html ,服务器只能回应HTML格式字符串，不能回应别的格式

http/1.0

1996年5月,支持cache, MIME, method

每个TCP连接只能发送一个请求，发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接引入了POST命令和HEAD命令头信息是 ASCII 码，后面数据可为任何格式。服务器回应时会告诉客户端，数据是什么格式，即Content-Type字段的作用。这些数据类型总称为MIME 多用途互联网邮件扩展，每个值包括一级类型和二级类型，预定义的类型，也可自定义类型, 常见Content-Type值：text/xml image/jpeg audio/mp3

http/1.1

1997年1月，引入了持久连接（persistent connection），即TCP连接默认不关闭，可以被多个请求复用，不用声明Connection: keep-alive。对于同一个域名，大多数浏览器允许同时建立6个持久连接引入了管道机制，即在同一个TCP连接里，客户端可以同时发送多个请求，进一步改进了HTTP协议的效率

新增方法：PUT、PATCH、OPTIONS、DELETE

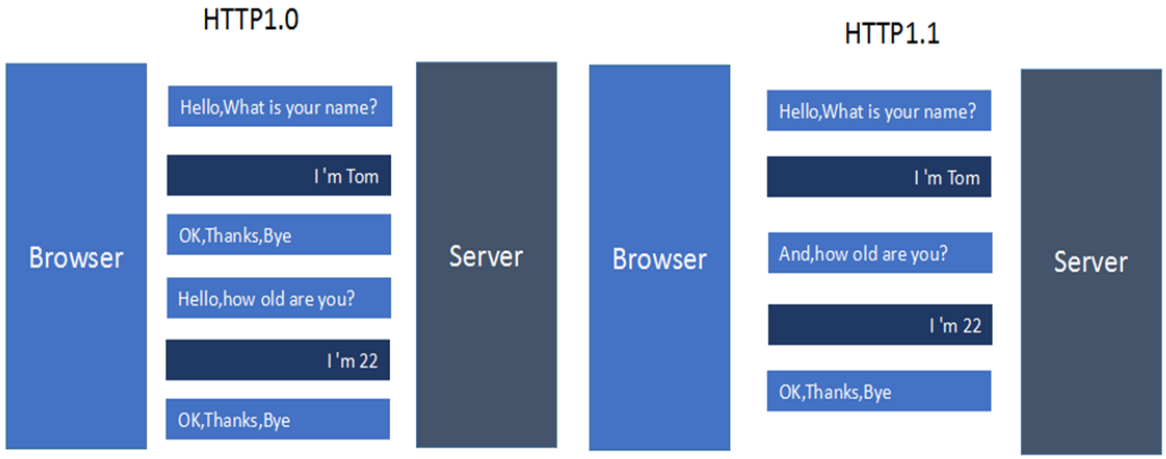
同一个TCP连接里，所有的数据通信是按次序进行的。服务器只能顺序处理回应，前面的回应慢，会有许多请求排队，造成"队头堵塞"（Head-of-line blocking）

为避免上述问题，两种方法：一是减少请求数，二是同时多开持久连接。

网页优化技巧，如合并脚本和样式表、将图片嵌入CSS代码、域名分片（domain sharding）等

HTTP 协议不带有状态，每次请求都必须附上所有信息。请求的很多字段都是重复的，浪费带宽，影响速度

HTTP1.0和HTTP1.1的区别



- 缓存处理，在HTTP1.0中主要使用header里的If-Modified-Since,Expires来做为缓存判断的标准，HTTP1.1则引入了更多的缓存控制策略例如Entity tag，If-Unmodified-Since, If-Match, If-None-Match等更多可供选择的缓存头来控制缓存策略
- 带宽优化及网络连接的使用，HTTP1.0中，存在一些浪费带宽的现象，例如：客户端只是需要某个对象的一部分，而服务器却将整个对象送过来了，并且不支持断点续传功能，HTTP1.1则在请求头引入了range头域，它允许只请求资源的某个部分，即返回码是206（Partial Content），方便了开发者自由的选择以便于充分利用带宽和连接
- 错误通知的管理，在HTTP1.1中新增24个状态响应码，如409（Conflict）表示请求的资源与资源当前状态冲突；410（Gone）表示服务器上的某个资源被永久性的删除
- Host 头处理，在HTTP1.0中认为每台服务器都绑定一个唯一的IP地址，因此，请求消息中的URL并没有传递主机名（hostname）。但随着虚拟主机技术的发展，在一台物理服务器上可以存在多个虚拟主机（Multi-homed Web Servers），并且它们共享一个IP地址。HTTP1.1的请求消息和响应消息都应支持Host头域，且请求消息中如果没有Host头域会报告一个错误（400 Bad Request）
- 长连接，HTTP 1.1支持持久连接（PersistentConnection）和请求的流水线（Pipelining）处理，在一个TCP连接上可以传送多个HTTP请求和响应，减少了建立和关闭连接的消耗和延迟，在HTTP1.1中默认开启Connection： keep-alive，弥补了HTTP1.0每次请求都要创建连接的缺点

HTTP1.0和1.1的问题

- HTTP1.x在传输数据时，每次都需要重新建立连接，无疑增加了大量的延迟时间，特别是在移动端更为突出
- HTTP1.x在传输数据时，所有传输的内容都是明文，客户端和服务器端都无法验证对方的身份，无法保证数据的安全性

- HTTP1.x在使用时，header里携带的内容过大，增加了传输的成本，并且每次请求header基本不怎么变化，尤其在移动端增加用户流量
- 虽然HTTP1.x支持了keep-alive，来弥补多次创建连接产生的延迟，但是keep-alive使用多了同样会给服务端带来大量的性能压力，并且对于单个文件被不断请求的服务(例如图片存放网站)，keep-alive可能会极大的影响性能，因为它在文件被请求之后还保持了不必要的连接很长时间

HTTPS协议：

为解决安全问题，网景在1994年创建了HTTPS，并应用在网景导航者浏览器中。 最初，HTTP是与SSL一起使用的；在SSL逐渐演变到TLS时（其实两个是一个东西，只是名字不同而已），最新的HTTPS也由在2000年五月公布的RFC 2818正式确定下来。HTTPS就是安全版的HTTP，目前大型网站基本实现全站HTTPS

HTTPS特点

- HTTPS协议需要到CA申请证书，一般免费证书很少，需要交费
- HTTP协议运行在TCP之上，所有传输的内容都是明文，HTTPS运行在SSL/TLS之上，SSL/TLS运行在TCP之上，所有传输的内容都经过加密的
- HTTP和HTTPS使用的是不同的连接方式，端口不同，前者是80，后者是443
- HTTPS可以有效的防止运营商劫持，解决了防劫持的一个大问题
- HTTPS 实现过程降低用户访问速度，但经过合理优化和部署，HTTPS 对速度的影响还是可以接受的

SPDY协议

SPDY：2009年谷歌研发，综合HTTPS和HTTP两者有点于一体的传输协议，主要特点：

- 降低延迟，针对HTTP高延迟的问题，SPDY优雅的采取了多路复用（multiplexing）。多路复用通过多个请求stream共享一个tcp连接的方式，解决了HOL blocking的问题，降低了延迟同时提高了带宽的利用率
- 请求优先级（request prioritization）。多路复用带来一个新的问题是，在连接共享的基础之上有可能会导致关键请求被阻塞。SPDY允许给每个request设置优先级，重要的请求就会优先得到响应。比如浏览器加载首页，首页的html内容应该优先展示，之后才是各种静态资源文件，脚本文件等加载，可以保证用户能第一时间看到网页内容
- header压缩。HTTP1.x的header很多时候都是重复多余的。选择合适的压缩算法可以减小包的大小和数量
- 基于HTTPS的加密协议传输，大大提高了传输数据的可靠性
- 服务端推送（server push），采用了SPDY的网页，例如网页有一个sytle.css的请求，在客户端收到sytle.css数据的同时，服务端会将sytle.js的文件推送给客户端，当客户端再次尝试获取sytle.js时就可以直接从缓存中获取到，不用再发请求了

HTTP2协议

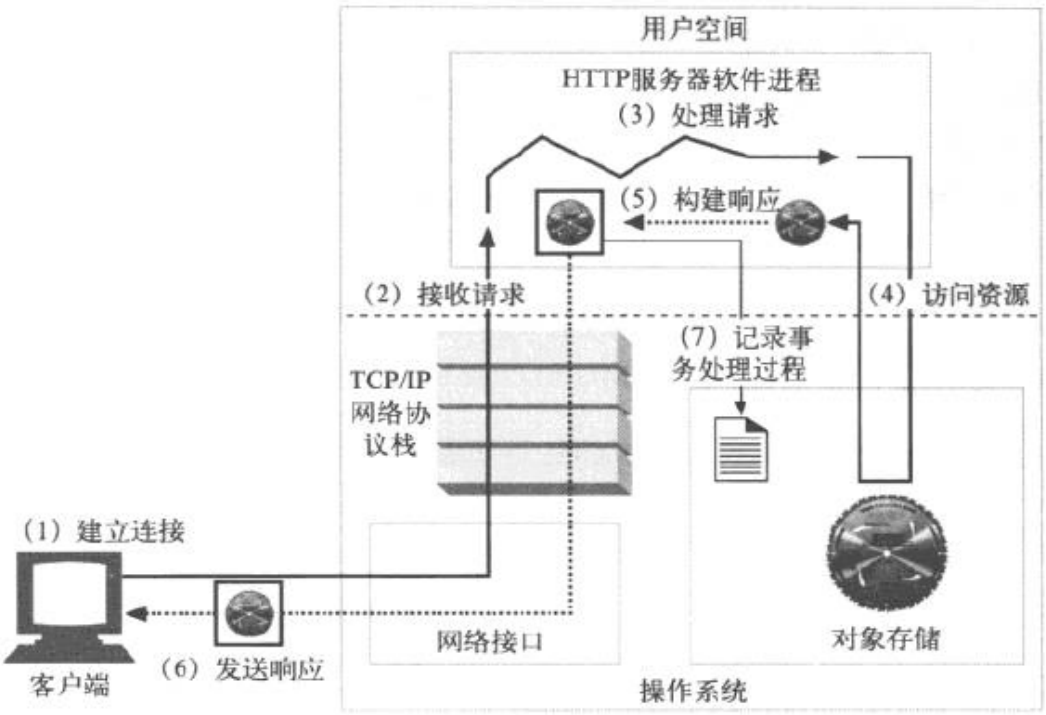
http/2.0：2015年，HTTP2.0是SPDY的升级版

- 头信息和数据体都是二进制，称为头信息帧和数据帧
- 复用TCP连接，在一个连接里，客户端和浏览器都可以同时发送多个请求或回应，且不用按顺序一一对应，避免了"队头堵塞",此双向的实时通信称为多工（Multiplexing）
- 引入头信息压缩机制（header compression）,头信息使用gzip或compress压缩后再发送；客户端和服务器同时维护一张头信息表，所有字段都会存入这个表，生成一个索引号，不发送同样字段，只发送索引号，提高速度
- HTTP/2 允许服务器未经请求，主动向客户端发送资源，即服务器推送（server push）

HTTP2.0和SPDY区别：

- HTTP2.0 支持明文 HTTP 传输，而 SPDY 强制使用 HTTPS
- HTTP2.0 消息头的压缩算法采用 HPACK，而非 SPDY 采用的 DEFLATE

1.3.7 HTTP 请求访问的完整过程

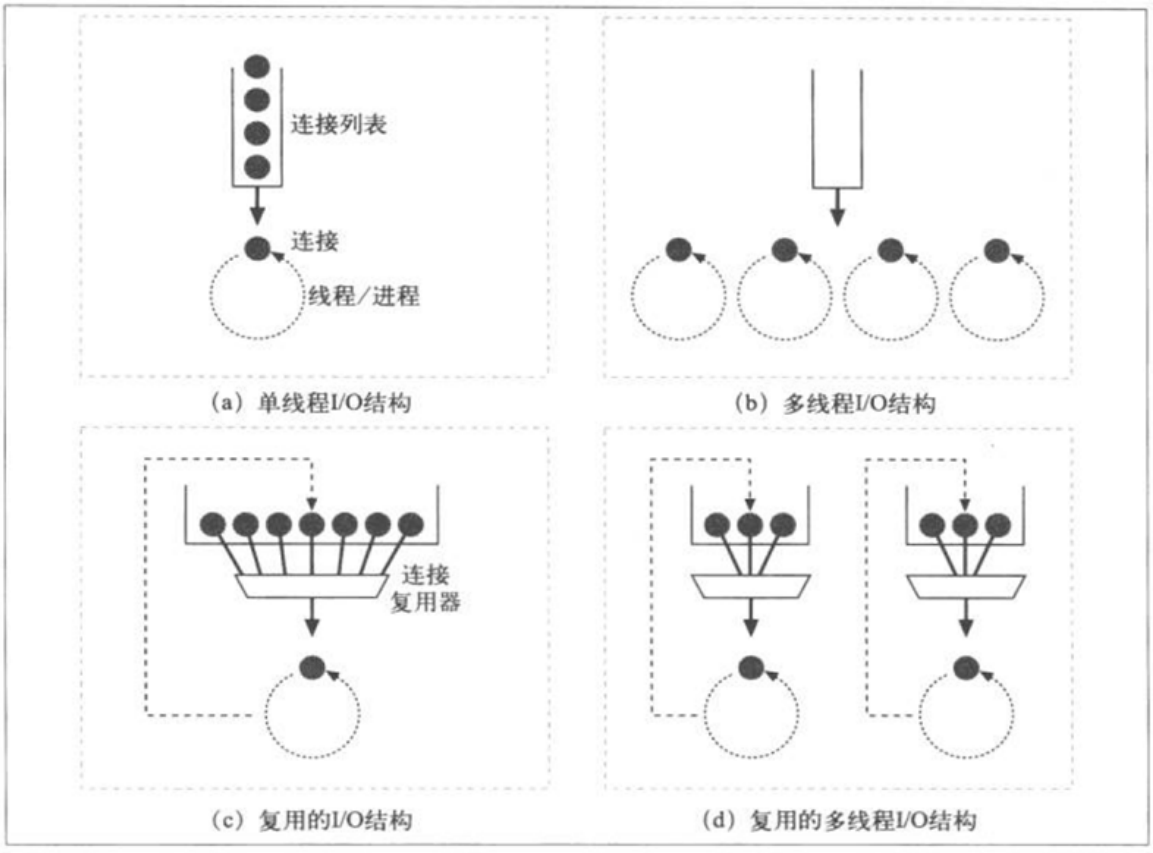


一次完整的http请求处理过程

1、建立连接：接收或拒绝连接请求

2、接收请求：接收客户端请求报文中对某资源的一次请求的过程

Web访问响应模型（Web I/O）



- 单进程I/O模型：启动一个进程处理用户请求，而且一次只处理一个，多个请求被串行响应
- 多进程I/O模型：并行启动多个进程,每个进程响应一个连接请求

- 复用I/O结构：启动一个进程，同时响应N个连接请求
- 复用的多进程I/O模型：启动M个进程，每个进程响应N个连接请求，同时接收M*N个请求

3、处理请求：服务器对请求报文进行解析，并获取请求的资源及请求方法等相关信息，根据方法，资源，首部和可选的主体部分对请求进行处理

常用请求Method：GET、POST、HEAD、PUT、DELETE、TRACE、OPTIONS

4、访问资源：

服务器获取请求报文中请求的资源web服务器，即存放了web资源的服务器，负责向请求者提供对方请求的静态资源，或动态运行后生成的资源

5、构建响应报文：

一旦Web服务器识别除了资源，就执行请求方法中描述的动作，并返回响应报文。响应报文中 包含有响应状态码、响应首部，如果生成了响应主体的话，还包括响应主体

1）响应实体：如果事务处理产生了响应主体，就将内容放在响应报文中回送过去。响应报文中通常包括：

描述了响应主体MIME类型的Content-Type首部
描述了响应主体长度的Content-Length
实际报文的主体内容

2）URL重定向：web服务构建的响应并非客户端请求的资源，而是资源另外一个访问路径

3）MIME类型： Web服务器要负责确定响应主体的MIME类型。多种配置服务器的方法可将MIME类型与资源管理起来

- 魔法分类：Apache web服务器可以扫描每个资源的内容，并将其与一个已知模式表(被称为魔法文件)进行匹配，以决定每个文件的MIME类型。这样做可能比较慢，但很方便，尤其是文件没有标准扩展名时
- 显式分类：可以对Web服务器进行配置，使其不考虑文件的扩展名或内容，强制特定文件或目录内容拥有某个MIME类型
- 类型协商： 有些Web服务器经过配置，可以以多种文档格式来存储资源。在这种情况下，可以配置Web服务器，使其可以通过与用户的协商来决定使用哪种格式(及相关的MIME类型)"最好"

6、发送响应报文

Web服务器通过连接发送数据时也会面临与接收数据一样的问题。服务器可能有很多条到各个客户端的连接，有些是空闲的，有些在向服务器发送数据，还有一些在向客户端回送响应数据。服务器要记录连接的状态，还要特别注意对持久连接的处理。对非持久连接而言，服务器应该在发送了整条报文之后，关闭自己这一端的连接。对持久连接来说，连接可能仍保持打开状态，在这种情况下，服务器要正确地计算Content-Length首部，不然客户端就无法知道响应什么时候结束

7、记录日志

最后，当事务结束时，Web服务器会在日志文件中添加一个条目，来描述已执行的事务

1.4 HTTP 协议报文头部结构

http协议：http/0.9, http/1.0, http/1.1, http/2.0，http/3.0

http协议：stateless 无状态, 服务器无法持续追踪访问者来源

解决http协议无状态方法

- cookie 客户端存放
- session 服务端存放

http事务：一次访问的过程

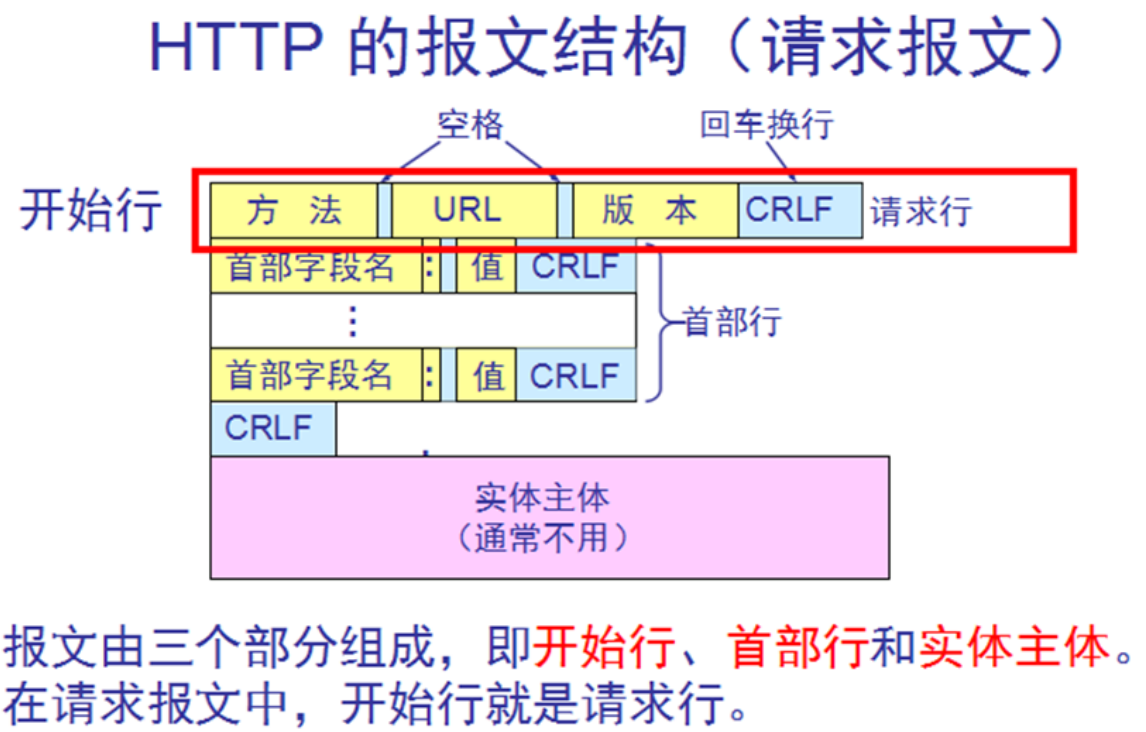
- 请求：request
- 响应：response

HTTP报文结构

协议查看或分析的工具：tcpdump, wireshark,tshark

参考资料：https://developer.mozilla.org/zh-CN/docs/Web/HTTP

1.4.1 HTTP请求报文



request报文格式

```
<method> <request-URL> <version>
<headers>
<entity-body>
```

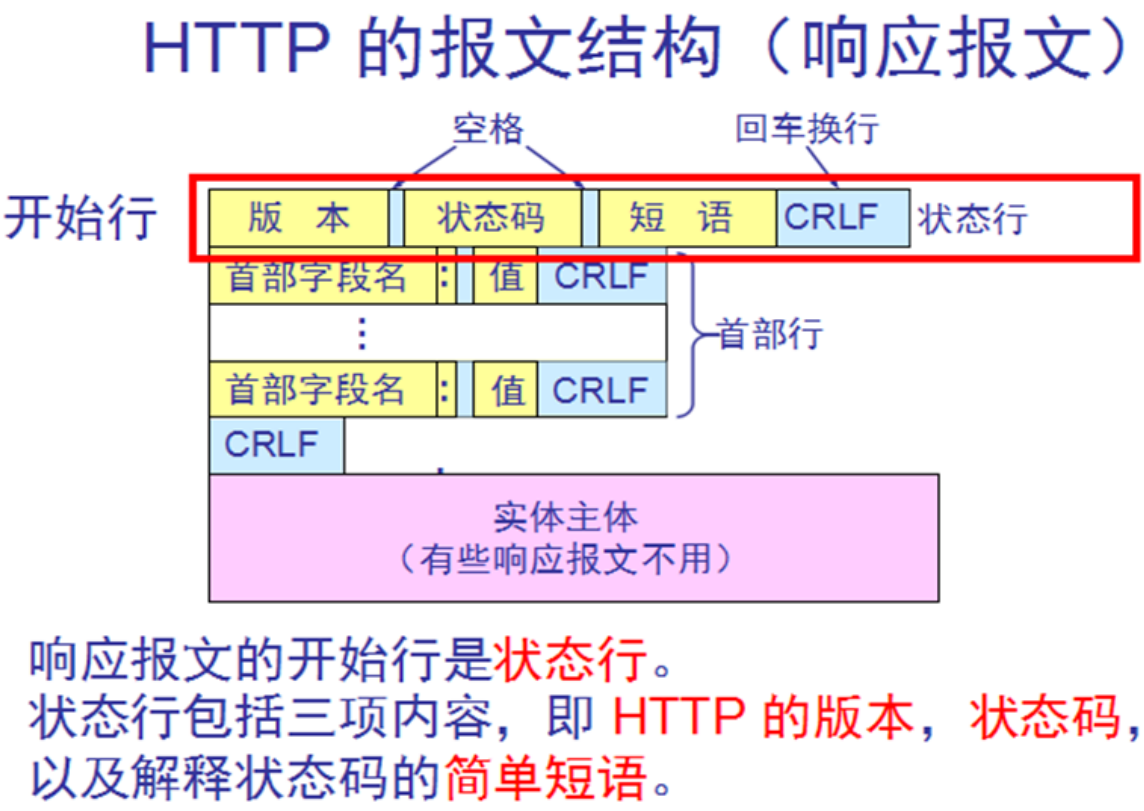
范例：

```
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: www.magedu.com
User-Agent: HTTPie/0.9.4
```

范例：

```
#post.html
<form action="index.html" method="POST">
username:<br>
<input type="text" name="username" >
<br>
password:<br>
<input type="text" name="password" >
<br><br>
<input type="submit" value="Submit">
</form>
```

1.4.2 HTTP响应报文



response报文格式

```
<version> <status> <reason-phrase>
<headers>
<entity-body>
```

范例：

```
HTTP/1.1 200 OK
Cache-Control: max-age=3, must-revalidate
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Thu, 07 Nov 2019 03:44:14 GMT
Server: Tengine
Transfer-Encoding: chunked
Vary: Accept-Encoding
Vary: Accept-Encoding, Cookie
```

1.4.3 HTTP报文格式详解

1.4.3.1 Method 方法

请求方法，标明客户端希望服务器对资源执行的动作，包括以下：

- GET： 从服务器获取一个资源
- HEAD： 只从服务器获取文档的响应首部
- POST： 向服务器输入数据，通常会再由网关程序继续处理
- PUT： 将请求的主体部分存储在服务器中，如上传文件
- DELETE： 请求删除服务器上指定的文档
- TRACE：追踪请求到达服务器中间经过的代理服务器
- OPTIONS：请求服务器返回对指定资源支持使用的请求方法
- CONNECT：建立一个到由目标资源标识的服务器的隧道
- PATCH：用于对资源应用部分修改

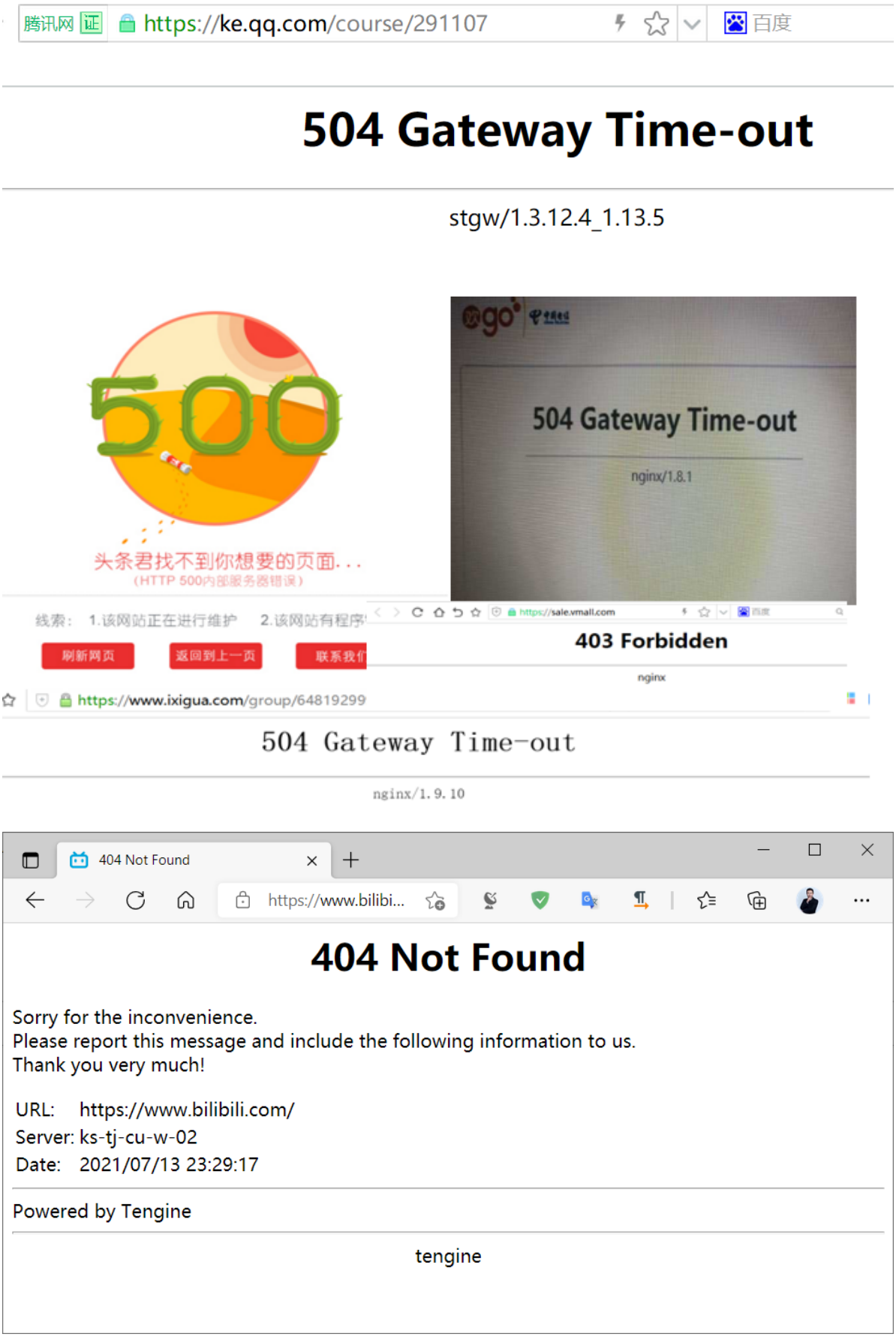
1.4.3.2 version版本

```
HTTP/<major>.<minor>
```

范例：

```
HTTP/1.1
```

1,4.3.3 status 状态码



三位数字，标记请求处理过程中发生的情况

参考资料：https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status

http协议状态码分类

```
1xx：100-101 信息提示
2xx：200-206 成功
3xx：300-307 重定向
4xx：400-415 错误类信息，客户端错误
5xx：500-505 错误类信息，服务器端错误
```

http协议常用的状态码

```
200： 成功，请求数据通过响应报文的entity-body部分发送;OK
301： 请求的URL指向的资源已经被删除；但在响应报文中通过首部Location指明了资源现在所处的新位置；Moved Permanently
302： 响应报文Location指明资源临时新位置 Moved Temporarily
304： 客户端发出了条件式请求，但服务器上的资源未曾发生改变，则通过响应此响应状态码通知客户端；Not Modified
307： 浏览器内部重定向
401： 需要输入账号和密码认证方能访问资源；Unauthorized
403： 请求被禁止；Forbidden
404： 服务器无法找到客户端请求的资源；Not Found
499： 客户端读超时关闭连接的错误码 499是客户端读超时关闭连接造成的，推荐从超时时间或者优化响应速度入手,web服务器发现客户端主动关闭连接后，记录到access日志中的。可能是客户端接收响应超时了,可以先在客户端统计下是不是这个原因，再调查为什么会导致超时
500： 服务器内部错误；Internal Server Error,比如:cgi程序没有执行权限
502： 代理服务器从后端服务器收到了一条伪响应，如无法连接到网关；Bad Gateway,比如后端服务端口没有打开
503： 服务不可用，临时服务器维护或过载，服务器无法处理请求,比如:php服务停止,无法处理php程序
504： 网关超时,或者后端服务器无回应报文
```

1.4.3.4 reason-phrase原因短语

状态码所标记的状态的简要描述

1.4.3.5 headers首部字段头

首部字段包含的信息最为丰富。首部字段同时存在于请求和响应报文内，并涵盖 HTTP 报文相关的内容信息。使用首部字段是为了给客服端和服务器端提供报文主体大小、所使用的语言、认证信息等内容

首部字段是由首部字段名和字段值构成的，中间用冒号":"分隔字段值对应，即key/value 键/值对

单个 HTTP 首部字段可以有多个值

参考资料：https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers

1.4.3.6 entity-body实体

请求时附加的数据或响应时附加的数据，例如：登录网站时的用户名和密码，博客的上传文章，论坛上的发言等。

1.4.4 Cookie 和 session

Cookies vs Sessions

无状态协议是指协议对事物处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它应答就很快。

HTTP是超本文传输协议，顾名思义，这个协议支持超文本的传输。什么是超文本？说白了就是使用HTML编写的页面。通常，我们使用客户端浏览器访问服务器的资源，最常见的URL也是以html为后缀的文件，因此可以说超文本是网络上最主要的资源。

既然HTTP协议的目的是在于支持超文本的传输，也就是资源的传输，那么客户端浏览器向HTTP服务器发送请求，继而HTTP服务器将相信资源发回给客户端这样一个过程中，无论对于客户端还是服务器，都没有必要记录这个过程，因为每一次请求和响应都是相对独立的，一般而言，一个URL对应着一个唯一的超文本，正是因为这样的唯一性，使得记录用户的行为状态变得毫无意义，所以，HTTP协议被设计为无状态的连接协议符合它本身的需求。

HTTP协议这种特性有优点也有缺点，优点在于解放了服务器，每一次请求"点到为止"，不会造成不必要的连接占用，缺点在于如果为了保留状态，每次请求都会传输大量的重复信息内容。

可是随着 Web 的不断发展，很多业务都需要对通信状态进行保存.

如果是一次性会话的过程： 打开浏览器 -> 访问一些服务器内容 -> 关闭浏览器

但目前有很多WEB访问场景，并不是一次性会话，而是多次相关的会话，比如：

登录场景：

打开浏览器 -> 浏览到登陆页面 -> 输入用户名和密码 -> 访问到用户主页(显示用户名) -> 修改密码（输入原密码） -> 修改收货地址.....

问题：在此处登录会话过程中产生的数据（用户会话数据）如何保存下来呢?

购物场景：

打开浏览器 -> 浏览商品列表 -> 加入购物车(把商品信息保存下来) -> 关闭浏览器
打开浏览器-> 直接进入购物车 -> 查看到上次加入购物车的商品 -> 下订单 -> 支付

问题： 在购物会话过程中，如何保存商品信息?

以上场景都需要保留会话数据，需要会话管理机制。

会话管理： 管理浏览器客户端和服务器端之间会话过程中产生的会话数据。

为了会话管理，HTTP就需要传输大量重复信息内容的问题，造成大量的网络带宽消耗。于是 Cookie 和 Session 技术闪亮登场了，它们可以为用户进行会话管理，实现保存状态。

1.4.4.1 Cookie

Cookie 又称为"小甜饼”。类型为"小型文本文件”，指某些网站为了辨别用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。由网景公司的前雇员卢·蒙特利在1993年3月发明

因为HTTP协议是无状态的，即服务器不知道用户上一次做了什么，这严重阻碍了交互式Web应用程序的实现。在典型的网上购物场景中，用户浏览了几个页面，买了一盒饼干和两瓶饮料。最后结帐时，由于HTTP的无状态性，不通过额外的手段，服务器并不知道用户到底买了什么，所以Cookie就是用来绕开HTTP的无状态性的"额外手段"之一。服务器可以设置或读取Cookies中包含信息，借此维护用户跟服务器会话中的状态。

在上面的购物场景中，当用户选购了第一项商品，服务器在向用户发送网页的同时，还发送了一段Cookie，记录着那项商品的信息。当用户访问另一个页面，浏览器会把Cookie发送给服务器，于是服务器知道他之前选购了什么。用户继续选购饮料，服务器就在原来那段Cookie里追加新的商品信息。结帐时，服务器读取发送来的Cookie就行了。

Cookie基于HTTP协议，也叫Web Cookie或浏览器Cookie，是服务器发送到用户浏览器并保存在客户端本地的一小块数据，它会在浏览器下次向同一服务器再发起请求时被携带并发送到服务器上。通常，它用于告知服务端两个请求是否来自同一浏览器，如保持用户的登录状态。Cookie使基于无状态的HTTP协议记录稳定的状态信息成为了可能。

cookie 的获取过程


```
#第一次请求过程
浏览器第一次发送请求时,不会携带任何cookie信息
服务器接收到请求之后，发现请求中没有任何cookie信息
服务器生成和设置一个cookie.并将此cookie设置通过set_cookie的首部字段保存在响应报文中返回给浏览器
浏览器接收到这个响应报文之后,发现里面有cookie信息,浏览器会将cookie信息保存起来
#第二次及其之后的过程
当浏览器第二次及其之后的请求报文中自动 cookie的首部字段携带第一次响应报文中获取的cookie信息
服务器再次接收到请求之后,会发现请求中携带的cookie信息,这样的话就认识是谁发的请求了
之后的响应报文中不会再添加set_cookie首部字段
```

Cookie主要用于以下三个方面：

- 会话状态管理（如用户登录状态、购物车、游戏分数或其它需要记录的信息）
- 个性化设置（如用户自定义设置、主题等）
- 浏览器行为跟踪（如跟踪分析用户行为等）

使用 Cookie 的状态管理

Cookie 技术通过在请求和响应报文中写入 Cookie 信息来控制客户端的状态。当服务器收到HTTP请求时，服务器可以在响应头里面添加一个Set-Cookie选项。浏览器收到响应后通常会保存下Cookie，之后对该服务器每一次请求中都通过Cookie请求头部将Cookie信息发送给服务器。服务器端发现客户端发送过来的 Cookie 后，会去检查究竟是从哪一个客户端发来的连接请求，然后对比服务器上的记录，最后得到之前的状态信息.另外，Cookie的过期时间、域、路径、有效期、适用站点都可以根据需要来指定。

Set-Cookie首部字段

- NAME=VALUE　赋予 Cookie 的名称和其值,此为必需项
- expires=DATE　Cookie 的有效期，若不明确指定则默认为浏览器关闭前为止

  会话期Cookie

  基于内存保存，会话期Cookie是最简单的Cookie：浏览器关闭之后它会被自动删除，也就是说它仅在会话期内有效。会话期Cookie不需要指定过期时间（Expires）或者有效期（Max-Age）。需要注意的是，有些浏览器提供了会话恢复功能，这种情况下即使关闭了浏览器，会话期Cookie也会被保留下来，就好像浏览器从来没有关闭一样。

  持久性Cookie

  基于硬盘保存，和关闭浏览器便失效的会话期Cookie不同，持久性Cookie可以指定一个特定的过期时间（Expires）或有效期（Max-Age）。

  ```
  Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT;
  ```

  提示：当Cookie的过期时间被设定时，设定的日期和时间只与客户端相关，而不是服务端。

- path=PATH　指定了主机下的哪些路径可以接受Cookie（该URL路径必须存在于请求URL中）。若不指定则默认为文档所在的文件目录,以字符 `%x2F` ("/") 作为路径分隔符，子路径也会被匹配。

  例如，设置 `Path=/docs`，则以下地址都会匹配：

  - `/docs`
  - `/docs/web/`
  - `/docs/web/HTTP`
- domain=域名　指定了哪些主机可以接受Cookie。如果不指定，默认为当前文档的主机（不包含子域名）。如果指定了Domain，则一般包含子域名。

  例如，如果设置 Domain=magedu.com，则Cookie也包含子域名（如：study.magedu.com）
- Secure　标记为 Secure 的Cookie只应通过被HTTPS协议加密过的请求发送给服务端。但即便设置了 Secure 标记，敏感信息也不应该通过Cookie传输，因为Cookie有其固有的不安全性，Secure 标记也无法提供确实的安全保障。从 Chrome 52 和 Firefox 52 开始，不安全的站点（http:）无法使用Cookie的 Secure 标记。
- HttpOnly　　加以限制使 Cookie 不能被 JavaScript 脚本访问,为避免跨域脚本 (XSS) 攻击，通过 JavaScript的 Document.cookie API无法访问带有 HttpOnly 标记的Cookie，它们只应该发送给服务端。如果包含服务端 Session 信息的 Cookie 不想被客户端 JavaScript 脚本调用，那么就应该为其设置 HttpOnly 标记

浏览器对cookie的限制：

Cookie 存储的限制是不一样的。例如：单个域名可存储的 Cookie 数量、Cookie 大小等。

	IE6.0	IE7.0/8.0	Opera	FF	Safari	Chrome
cookie个数	每个域为20个	每个域为50个	每个域为30个	每个域为50个	没有个数限制	每个域为53个
cookie大小	4095个字节	4095个字节	4096个字节	4097个字节	4097个字节	4097个字节

在进行页面 Cookie 操作的时候，应该尽量保证 Cookie 的个数小于 20 个，总大小小于 4KB，这是一个安全且保险的范围。

范例：响应报文中的set-cookie首部

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: yummy_cookie=choco
Set-Cookie: tasty_cookie=strawberry
```

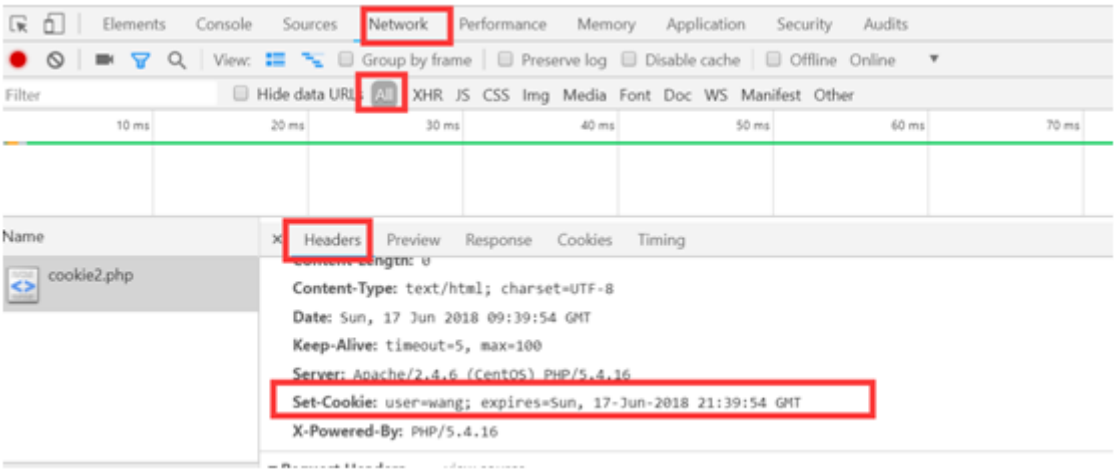
范例：请求报文中的cookie首部字段

```
GET /sample_page.html HTTP/1.1
Host: www.example.org
Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

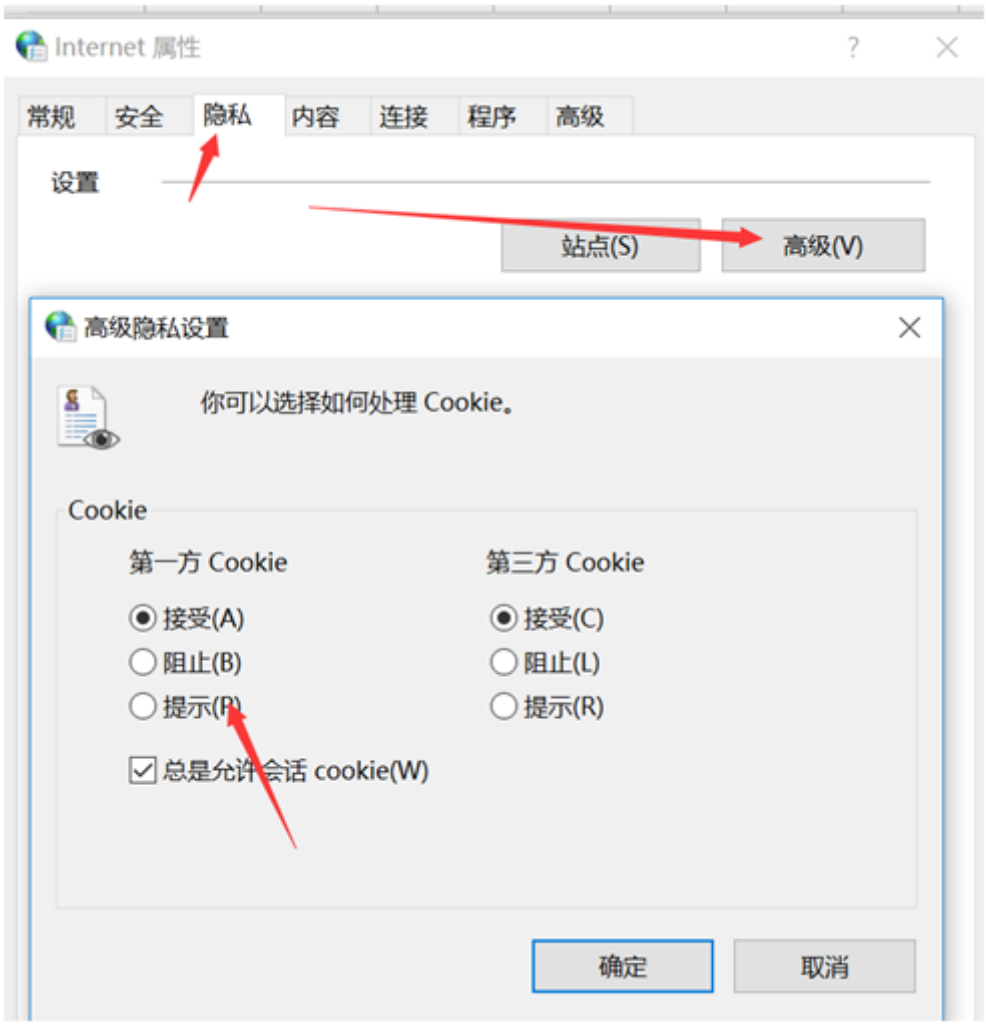
范例：响应报文set-cookie中的Secure 和 HttpOnly

```
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT; Secure; HttpOnly
```

范例：浏览器查看cookie



范例：chrome 浏览器禁止cookie

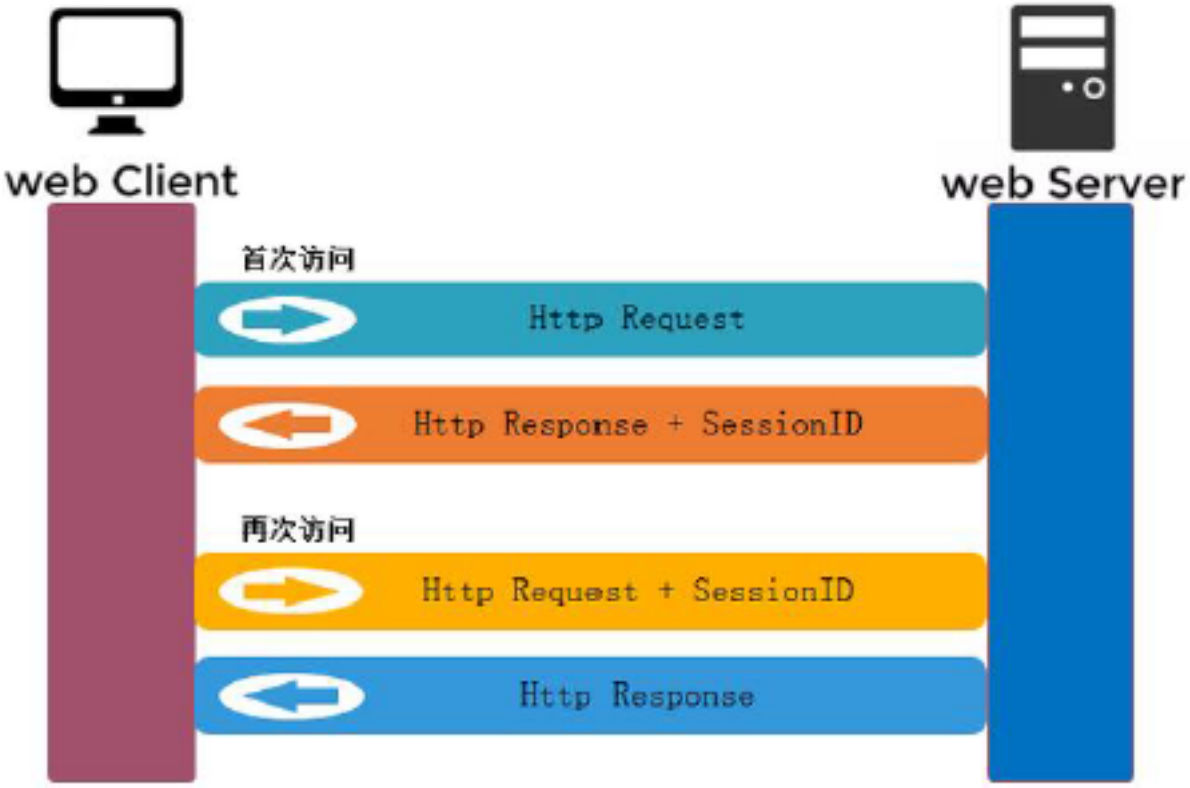

Internet 属性
常规
安全
隐私
内容
连接
程序
高级
设置
站点(S)
高级(V)
高级隐私设置
你可以选择如何处理 Cookie。
Cookie
第一方 Cookie
第三方 Cookie
接受(A)
阻止(B)
提示(P)
接受(C)
阻止(L)
提示(R)
总是允许会话 cookie(W)
确定
取消

范例：php语言实现cookie的管理

```
#设置cookie
#cat setcookie.php
<?php
setcookie('title','cto');     #有效期为会话级
setcookie('user','wang',time()+3600*12);  #有效期为12小时
echo "<h1>test setcookie </h1>"
?>
#说明：setcookie设置的cookie，只有下一次http请求才能生效

#显示cookie
cat showcookies.php
<?php
echo "<h1>test showcookie </h1>";
echo $_COOKIE["user"];     #显示user的这一个cookie
echo "<br />";
var_dump($_COOKIE);       #显示所有cookie
//print_r($_COOKIE);      #不如上面方式详细
?>

#删除cookie，通过设置过期时间实现
#vim delcookie.php
<?php
```

```
setcookie('user','wang',time()-3600*12);
echo "<h1>cookie:user is deleted </h1>";
?>
```

1.4.4.2 Session



session

session是相对于cookie的另外一个状态保持的解决方案，它是通过服务器来保持状态的。session指的是服务器上为每个客户端所开辟的独立存储空间，在其中保存的信息就是用于保存状态的。

Session是服务器端程序运行的过程中创建的，不同语言实现的应用程序有不同创建session的方法。在创建了session的同时，服务器会为该session生成唯一的sessionId，而这个sessionId被创建了之后，就可以调用session相关的方法往session中增加内容了，而这些内容只会保存在服务器中，每个sessionid就像数据库中主键，可以根据SessionId 关联每个session的相关信息，比如：购物车里的商器，登录用户等。但发送给客户端浏览器的只有sessionId。当客户端浏览器再次发送http请求时，会自动地将这个sessionId 附加在请求报文中，服务器收到请求之后就会根据sessionId找到对应的session，从而再次使用，使得用户的状态得以保持。

每个session都有一个sessionId，这个ID存放有两种方式：

1、通过URL存取，比如：Java程序中，URL会带上一个jsessionId=xxxxxx等，这样每次重新请求的时候都传了sessionId给服务器,但此方式不安全,所以很少使用,所以一般session是依赖于cookie的.即如果浏览器禁用了cookie,则session无法实现

2、通过cookie存取（Tomcat默认如此），这种cookie是session cookie，区别于persistent cookies也就是我们常说的cookie，session cookie要注意的是存储在浏览器内存中，而不是写到硬盘上。程序一开始执行，服务器就生成一个sessionId并通过cookie携带客户端浏览器的缓存中，当下一次访问的时候，服务器先检测一下是否有这个cookie，如果有就取它的ID，如果没有就再生成一个。这就是为什么关闭浏览器之后，再进去session已经没有了，其实在服务器端session并没有清空，而是sessionId变了。

当将浏览器关闭，服务器保存的session数据不是立即释放的，此时数据还会存在一段时间（可以在程序中加以设置，Tomcat默认15分钟），只要我们知道那个sessionId，就可以继续通过请求获得此session的信息。session里面的数据都放在服务器端，通过sessionId保证不会访问错误，服务端自动对session进行管理，如果在规定的时间内没有访问，则释放掉这个session。

最后提两点：

1、sessionId通常在浏览器地址中是看不到的，但是当我们把浏览器的cookie禁止之后，Web服务器会采用URL重写的方式传递sessionId，这样就可以在地址栏看到sessionId了

2、session cookie不可以跨窗口使用，但可以跨同一个窗口的多个标签页。

session 的工作流程

第一次请求:

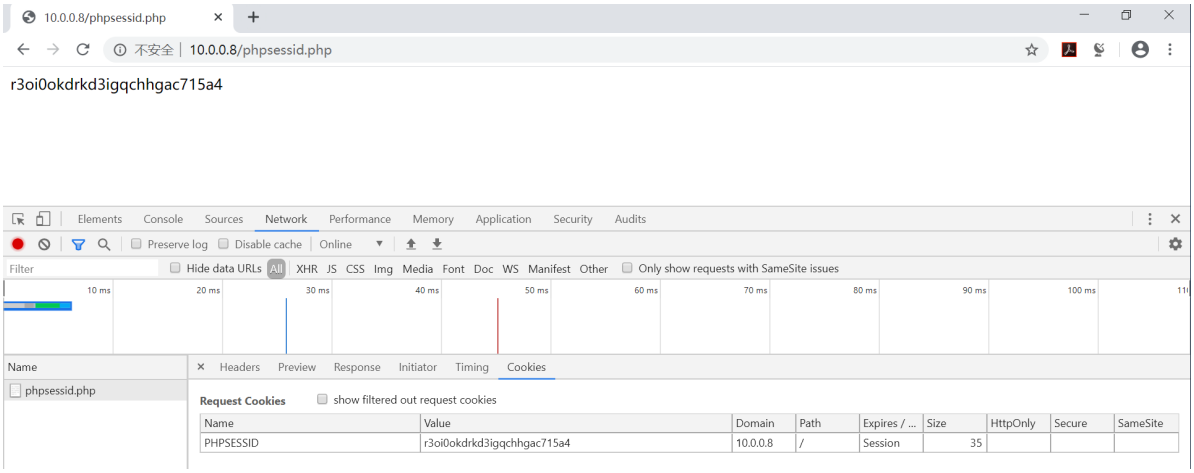
- 浏览器发起第一次请求的时候可以携带一些信息(比如: 用户名/密码) cookie中没有任何信息
- 当服务器接收到这个请求之后,进行用户名和密码的验证,验证成功后则可以设置session信息
- 在设置session信息的同时(session信息保存在服务器端) .服务器会在响应头中设置一个随机的session id的cookie信息
- 客户端(浏览器)在接收到响应之后,会将cookie信息保存起来(保存session id的信息)

第二次及其之后的请求:

- 第二次及其之后的请求都会携带session id信息
- 当服务器接收到这个请求之后,会获取到session id信息,然后进行验证
- 验证成功,则可以获取session信息(session信息保存在服务器端)

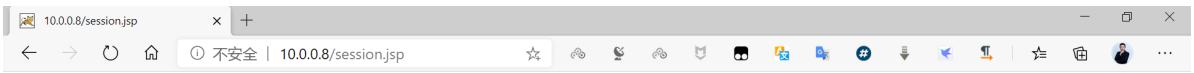
范例：PHP的PHPSESSID

```
<?php
session_start();
echo session_id();
?>
#执行结果如下图
```

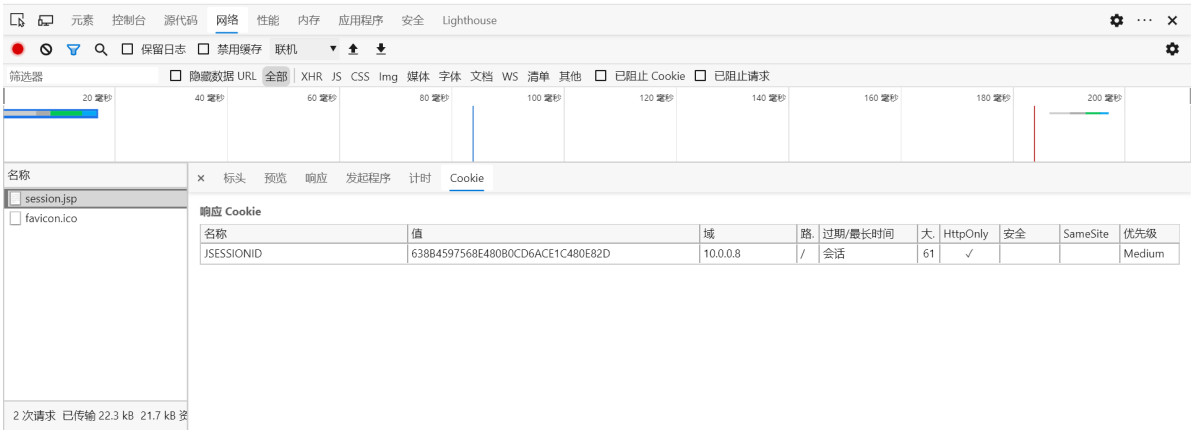


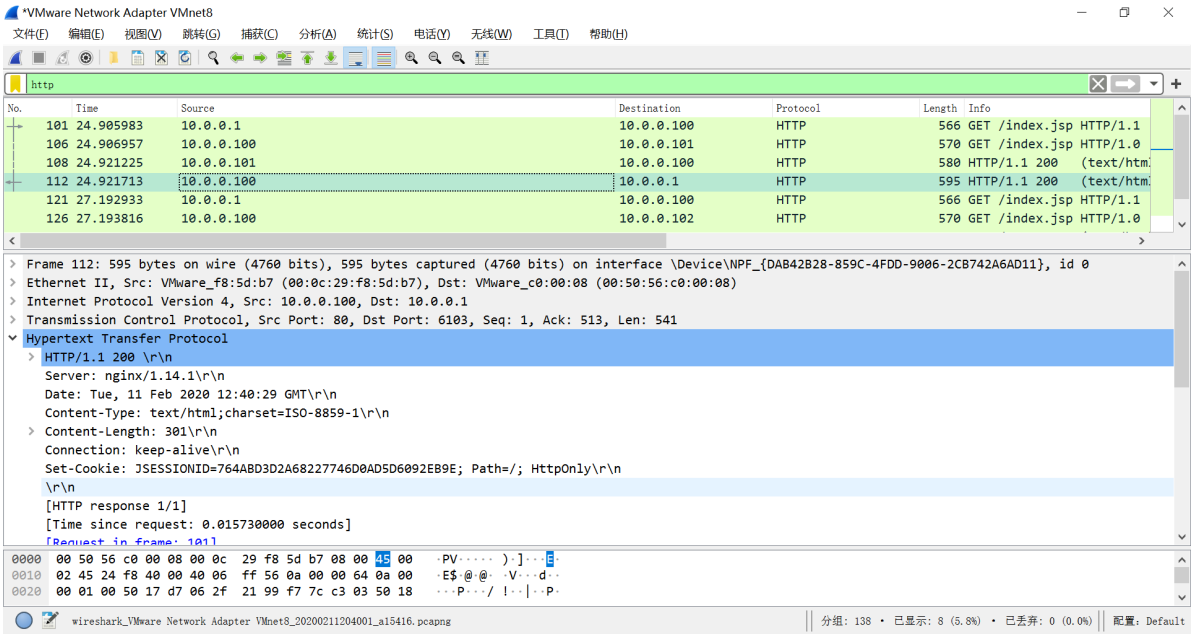
范例：JAVA的JSESSIONID

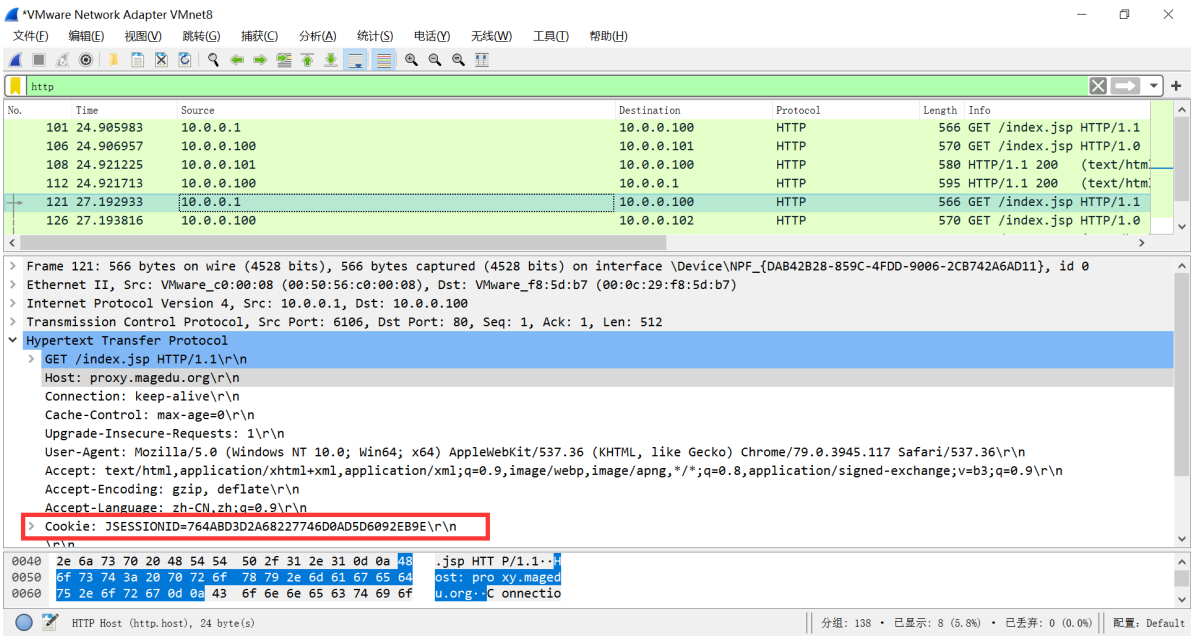
```
[root@centos8 ~]#cat /usr/local/tomcat/webapps/ROOT/session.jsp
<h1>
sessionId: <%=request.getSession().getId()%>
</h1>
```



sessionId: 638B4597568E480B0CD6ACE1C480E82D







1.4.4.3 cookie和session比较

cookie和session的相同和不同：

- cookie通常是在服务器生成,但也可以在客户端生成,session是在服务器端生成的
- session 将数据信息保存在服务器端，可以是内存，文件，数据库等多种形式,cookie 将数据保存在客户端的内存或文件中
- 单个cookie保存的数据不能超过4K，每个站点cookie个数有限制，比如IE8为50个、Firefox为50个、Opera为30个；session存储在服务器，没有容量限制
- cookie存放在用户本地，可以被轻松访问和修改，安全性不高；session存储于服务器，比较安全
- cookie有**会话cookie**和**持久cookie**，生命周期为浏览器会话期的会话cookie保存在缓存，关闭浏览器窗口就消失，持久cookie被保存在硬盘，知道超过设定的过期时间；随着服务端session存储压力增大，会根据需要定期清理session数据
- session中有众多数据，只将sessionID这一项可以通过cookie发送至客户端进行保留，客户端下次访问时，在请求报文中的cookie会自动携带sessionID，从而和服务器上的的session进行关联

cookie缺点：

1、使用cookie来传递信息，随着cookie个数的增多和访问量的增加，它占用的网络带宽也很大，试想假如cookie占用200字节，如果一天的PV有几个亿，那么它要占用多少带宽?

2、cookie并不安全，因为cookie是存放在客户端的，所以这些cookie可以被访问到，设置可以通过插件添加、修改cookie。所以从这个角度来说，我们要使用sesssion，session是将数据保存在服务端的，只是通过cookie传递一个sessionId而已，所以session更适合存储用户隐私和重要的数据

session 缺点：

1、不容易在多台服务器之间共享，可以使用session绑定，session复制，session共享解决

2、session存放在服务器中，所以session如果太多会非常消耗服务器的性能

cookie和session各有优缺点，在大型互联网系统中，单独使用cookie和session都是不可行的

1.5 Web相关工具

1.5.1 links

格式：

```
links  [OPTION]... [URL]...
```

常用选项：

- -dump　非交互式模式，显示输出结果
- -source　打印源码

1.5.2 wget

格式：

```
wget [OPTION]... [URL]...
```

常用选项：

```
#启动
```

```
-V, -version 显示wget的版本后退出
-h, -help 打印语法帮助
-b, -background 启动后转入后台执行
-e, -execute=COMMAND 执行`.wgetrc'格式的命令，wgetrc格式参见/etc/wgetrc或~/.wgetrc

#记录和输入文件
-o, -output-file=FILE 把记录写到FILE文件中
-a, -append-output=FILE 把记录追加到FILE文件中
-d, -debug 打印调试输出
-q, -quiet 安静模式(没有输出)
-v, -verbose 冗长模式(这是缺省设置)
-nv, -non-verbose 关掉冗长模式，但不是安静模式
-i, -input-file=FILE 下载在FILE文件中出现的URLs
-F, -force-html 把输入文件当作HTML格式文件对待
-B, -base=URL 将URL作为在-F -i参数指定的文件中出现的相对链接的前缀
-sslcertfile=FILE 可选客户端证书
-sslcertkey=KEYFILE 可选客户端证书的KEYFILE
-egd-file=FILE 指定EGD socket的文件名

#下载
-bind-address=ADDRESS
指定本地使用地址(主机名或IP，当本地有多个IP或名字时使用)
-t, -tries=NUMBER 设定最大尝试链接次数(0 表示无限制).
-O -output-document=FILE 把文档写到FILE文件中
-nc, -no-clobber 不要覆盖存在的文件或使用.#前缀
-c, -continue 接着下载没下载完的文件
-progress=TYPE 设定进程条标记
-N, -timestamping 不要重新下载文件除非比本地文件新
-S, -server-response 打印服务器的回应
-spider 不下载任何东西
-T, -timeout=SECONDS 设定响应超时的秒数
-w, -wait=SECONDS 两次尝试之间间隔SECONDS秒
-waitretry=SECONDS 在重新链接之间等待1…SECONDS秒
-random-wait 在下载之间等待0…2*WAIT秒
-Y, -proxy=on/off 打开或关闭代理
-Q, -quota=NUMBER 设置下载的容量限制
-limit-rate=RATE 限定下载输率

#目录
-nd -no-directories 不创建目录
-x, -force-directories 强制创建目录
-nH, -no-host-directories 不创建主机目录
-P, -directory-prefix=PREFIX 将文件保存到目录 PREFIX/…
-cut-dirs=NUMBER 忽略 NUMBER层远程目录

#HTTP 选项
-http-user=USER 设定HTTP用户名为 USER.
-http-passwd=PASS 设定http密码为 PASS.
-C, -cache=on/off 允许/不允许服务器端的数据缓存 (一般情况下允许).
-E, -html-extension 将所有text/html文档以.html扩展名保存
-ignore-length 忽略 `Content-Length'头域
-header=STRING 在headers中插入字符串 STRING
-proxy-user=USER 设定代理的用户名为 USER
-proxy-passwd=PASS 设定代理的密码为 PASS
-referer=URL 在HTTP请求中包含 `Referer: URL'头
-s, -save-headers 保存HTTP头到文件
-U, -user-agent=AGENT 设定代理的名称为 AGENT而不是 Wget/VERSION.
-no-http-keep-alive 关闭 HTTP活动链接 (永远链接).
```

```
-cookies=off 不使用 cookies.
-load-cookies=FILE 在开始会话前从文件 FILE中加载cookie
-save-cookies=FILE 在会话结束后将 cookies保存到 FILE文件中

#FTP 选项
-nr, -dont-remove-listing 不移走 `.listing'文件
-g, -glob=on/off 打开或关闭文件名的 globbing机制
-passive-ftp 使用被动传输模式 (缺省值).
-active-ftp 使用主动传输模式
-retr-symlinks 在递归的时候，将链接指向文件(而不是目录)

#递归下载
-r, -recursive 递归下载--慎用!
-l, -level=NUMBER 最大递归深度 (inf 或 0 代表无穷).
-delete-after 在现在完毕后局部删除文件
-k, -convert-links 转换非相对链接为相对链接
-K, -backup-converted 在转换文件X之前，将之备份为 X.orig
-m, -mirror 等价于 -r -N -l inf -nr.
-p, -page-requisites 下载显示HTML文件的所有图片

#递归下载中的包含和不包含(accept/reject)
-A, -accept=LIST 分号分隔的被接受扩展名的列表
-R, -reject=LIST 分号分隔的不被接受的扩展名的列表
-D, -domains=LIST 分号分隔的被接受域的列表
-exclude-domains=LIST 分号分隔的不被接受的域的列表
-follow-ftp 跟踪HTML文档中的FTP链接
-follow-tags=LIST 分号分隔的被跟踪的HTML标签的列表
-G, -ignore-tags=LIST 分号分隔的被忽略的HTML标签的列表
-H, -span-hosts 当递归时转到外部主机
-L, -relative 仅仅跟踪相对链接
-I, -include-directories=LIST 允许目录的列表
-X, -exclude-directories=LIST 不被包含目录的列表
-np, -no-parent 不要追溯到父目录
```

常用选项:

```
-q                  静默模式
-c                  断点续传
-P /path            保存在指定目录
-O  filename        保存为指定文件名，filename 为 - 时，发送至标准输出
--limit-rate=       指定传输速率，单位K，M等
```

范例:

```
[root@centos8 ~]#wget --limit-rate 1M -P /data
https://mirrors.aliyun.com/centos/8/isos/x86_64/CentOS-8-x86_64-1905-dvd1.iso
--2019-12-12 13:02:18--  https://mirrors.aliyun.com/centos/8/isos/x86_64/CentOS-
8-x86_64-1905-dvd1.iso
Resolving mirrors.aliyun.com (mirrors.aliyun.com)... 27.221.92.112,
119.167.168.225, 61.240.128.248, ...
Connecting to mirrors.aliyun.com (mirrors.aliyun.com)|27.221.92.112|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 7135559680 (6.6G) [application/octet-stream]
Saving to: ‘/data/CentOS-8-x86_64-1905-dvd1.iso’
```

```
CentOS-8-x86_64-1905-dvd1.iso             100%
[================================================================================
=========>]   6.65G  1.04MB/s    in 1h 53m

2019-12-12 14:55:45 (1024 KB/s) - ‘/data/CentOS-8-x86_64-1905-dvd1.iso’ saved
[7135559680/7135559680]

[root@centos8 ~]#ls /data
CentOS-8-x86_64-1905-dvd1.iso
```

范例：实现浏览器功能

```
[root@centos8 ~]#wget -qO - http://10.0.0.6/
<h1>welcome to magedu</1>
```

范例：下载指定目录

```
[root@centos8 ~]#wget -r -np -nH -R index.html http://www.example.com/dir/
[root@centos8 ~]#wget -c -r -np -k -L -p http://www.example.com/dir/

-r : 遍历所有子目录
-np : 不到上一层子目录去
-nH : 不要将文件保存到主机名文件夹
-R index.html : 不下载 index.html 文件
```

1.5.3 curl

curl是基于URL语法在命令行方式下工作的文件传输工具，它支持FTP, FTPS, HTTP, HTTPS, GOPHER, TELNET, DICT, FILE及LDAP等协议。curl支持HTTPS认证，并且支持HTTP的POST、PUT等方法， FTP上传， kerberos认证，HTTP上传，代理服务器，cookies，用户名/密码认证， 下载文件断点续传，上载文件断点续传, http代理服务器管道（ proxy tunneling），还支持IPv6，socks5代理服务器，通过http代理服务器上传文件到FTP服务器等，功能十分强大

格式：

```
curl  [options]  [URL...]
```

常见选项：

```
-A/--user-agent <string> 设置用户代理发送给服务器
-e/--referer <URL> 来源网址
--cacert <file> CA证书 (SSL)
-k/--insecure   允许忽略证书进行 SSL 连接
--compressed 要求返回是压缩的格式
-H/--header "key:value"  自定义首部字段传递给服务器
-i  显示页面内容，包括报文首部信息
-I/--head 只显示响应报文首部信息
-D/--dump-header <file>将url的header信息存放在指定文件中
--basic 使用HTTP基本认证
-u/--user <user[:password]>设置服务器的用户和密码
-L   如果有3xx响应码，重新发请求到新位置
-O  使用URL中默认的文件名保存文件到本地
-o <file>  将网络文件保存为指定的文件中
--limit-rate <rate> 设置传输速度
```

```
-0/--http1.0  数字0，使用HTTP 1.0
-v/--verbose  更详细
-C  选项可对文件使用断点续传功能
-c/--cookie-jar <file name> 将url中cookie存放在指定文件中
-x/--proxy <proxyhost[:port]> 指定代理服务器地址
-X/--request <command> 向服务器发送指定请求方法
-U/--proxy-user <user:password> 代理服务器用户和密码
-T  选项可将指定的本地文件上传到FTP服务器上
--data/-d 方式指定使用POST方式传递数据
-s --silent    Silent mode
-b name=data 从服务器响应set-cookie得到值，返回给服务器
-w <format>  显示相应的指定的报文信息，如：%{http_code}，%{remote_ip}等
-m, --max-time <time> 允许最大传输时间
```

范例:

```
[root@centos8 ~]#curl -I http://www.163.com
HTTP/1.1 403 Forbidden
Date: Thu, 12 Dec 2019 01:18:11 GMT
Content-Type: text/html
Content-Length: 234
Connection: keep-alive
Server: web cache
Expires: Thu, 12 Dec 2019 01:18:11 GMT
X-Ser: BC14_lt-tianjin-tianjin-3-cache-3
Cache-Control: no-cache,no-store,private
cdn-user-ip: 123.118.223.243
cdn-ip: 125.39.21.14
X-Cache-Remote: HIT
cdn-source: baishan

[root@centos8 ~]#curl -I -A ie10  http://www.163.com
HTTP/1.1 200 OK
Date: Thu, 12 Dec 2019 01:19:30 GMT
Content-Type: text/html; charset=GBK
Connection: keep-alive
Expires: Thu, 12 Dec 2019 01:20:45 GMT
Server: nginx
Cache-Control: no-cache,no-store,private
Age: 5
Vary: Accept-Encoding
X-Ser: BC20_dx-lt-yd-fujian-xiamen-8-cache-2, BC57_dx-lt-yd-fujian-xiamen-8-
cache-2, BC5_lt-tianjin-tianjin-3-cache-3, BC13_lt-tianjin-tianjin-3-cache-3
cdn-user-ip: 123.118.223.243
cdn-ip: 125.39.21.13
X-Cache-Remote: HIT
cdn-source: baishan

[root@centos6 ~]#curl -H "user-agent: firefox"  192.168.100.8
```

范例: 判断网站正常

```
[root@centos8 ~]#if [ "$(curl -sL -w '%{http_code}' http://www.wangxiaochun.com -o /dev/null)" = "200" ]; then
    echo "Success"
else
    echo "Fail"
fi

[root@centos8 ~]#if curl -sL --fail http://www.wangxiaochun.com -o /dev/null; then
    echo "Success"
else
    echo "Fail"
fi
```

范例：利用curl 获取响应码和远程主机IP

```
[root@ubuntu ~]#curl -s -I -m10 -o /dev/null   -w %{http_code}
http://www.baidu.com/
200
[root@ubuntu ~]#curl -s -I -m10 -o /dev/null   -w %{remote_ip}
http://wangxiaochun.com/
58.87.87.99
[root@centos8 ~]#curl -s -I -m10 -o /dev/null  -w %{local_ip}
http://wangxiaochun.com/
10.0.0.8
[root@centos8 ~]#curl -s -I -m10 -o /dev/null  -w %{local_port}
http://wangxiaochun.com/
45304
[root@centos8 ~]#curl -s -I -m10 -o /dev/null  -w %{remote_port}
http://wangxiaochun.com/
80
```

1.5.4 httpie

HTTPie 工具是功能丰富的 HTTP 命令行客户端，它能通过命令行界面与 Web 服务进行交互。它提供一个简单的 http 命令，允许使用简单而自然的语法发送任意的 HTTP 请求，并会显示彩色的输出

HTTPie 能用于测试、调试及与 HTTP 服务器交互。

主要特点：

- 具表达力的和直观语法
- 格式化的及彩色化的终端输出
- 内置 JSON 支持
- 表单和文件上传
- HTTPS、代理和认证
  任意请求数据
- 自定义头部
- 持久化会话
- 类似 wget 的下载
- 支持 Python 2.7 和 3.x

官方网站：https://httpie.org

安装：基于EPEL

```
[root@centos8 ~]#yum install httpie -y
```

范例：查看帮助

```
[root@centos8 ~]#http --help
usage: http [--json] [--form] [--pretty {all,colors,format,none}]
            [--style STYLE] [--print WHAT] [--headers] [--body] [--verbose]
            [--all] [--history-print WHAT] [--stream] [--output FILE]
            [--download] [--continue]
            [--session SESSION_NAME_OR_PATH | --session-read-only
SESSION_NAME_OR_PATH]
            [--auth USER[:PASS]] [--auth-type {basic,digest}]
            [--proxy PROTOCOL:PROXY_URL] [--follow]
            [--max-redirects MAX_REDIRECTS] [--timeout SECONDS]
            [--check-status] [--verify VERIFY]
            [--ssl {ssl2.3,ssl3,tls1,tls1.1,tls1.2}] [--cert CERT]
            [--cert-key CERT_KEY] [--ignore-stdin] [--help] [--version]
            [--traceback] [--debug]
            [METHOD] URL [REQUEST_ITEM [REQUEST_ITEM ...]]
```

范例：

```
# 显示信息（包含响应头200）
http www.magedu.com

# 显示详细的请求（包含请求和返回头200）
http -v www.magedu.com

# 只显示Header
http -h www.magedu.com
http --head www.magedu.com
http --header www.magedu.com
http --headers www.magedu.com

# 只显示Body
http -b www.magedu.com
```

```
http --body magedu.com

# 下载文件
http -d www.magedu.com

# 模拟提交表单
http -f POST www.magedu.com username='wang'

# 请求删除的方法
http DELETE www.magedu.com

# 传递JSON数据请求(默认就是JSON数据请求)
http PUT www.magedu.com username='wang' password='magedu'

# 如果JSON数据存在不是字符串则用:=分隔，例如
http PUT www.magedu.com username='wang' password='magedu' age:=30 a:=true
streets:='["a", "b"]'

# 模拟Form的Post请求, Content-Type: application/x-www-form-urlencoded;
charset=utf-8
http --form POST www.magedu.com username='wang'

# 模拟Form的上传, Content-Type: multipart/form-data
http -f POST www.magedu.com/jobs username='wang' file@~/test.pdf

# 修改请求头, 使用:分隔
http www.magedu.com  User-Agent:magedu-agent/1.0  'Cookie:a=b;b=c'
Referer:http://www.google.com/

# 认证
http -a username:password www.magedu.com
http -A basic -a username:password www.magedu.com

# 使用http代理
http --proxy=http:http://172.16.0.100:8081 proxy.magedu.com
http --proxy=http:http://user:pass@172.16.0.100:8081 proxy.magedu.com
http --proxy=https:http://172.16.0.100:8118 proxy.magedu.com
http --proxy=https:http://user:pass@172.16.0.100:8118 proxy.magedu.com
```

范例：查看信息及响应头

```
[root@centos8 ~]#http 192.168.8.8/test.html
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Length: 30
Content-Type: text/html; charset=UTF-8
Date: Thu, 07 Nov 2019 09:09:34 GMT
ETag: "1e-596be05bc9a34"
Keep-Alive: timeout=5, max=100
Last-Modified: Thu, 07 Nov 2019 09:09:27 GMT
Server: Apache/2.4.37 (centos)

<strong>马哥教育</strong>
```

范例：查看请示和响应头部及信息

```
[root@centos8 ~]#http -v 192.168.8.8/test.html
GET /test.html HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: 192.168.8.8
User-Agent: HTTPie/0.9.4



HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Length: 30
Content-Type: text/html; charset=UTF-8
Date: Thu, 07 Nov 2019 09:09:39 GMT
ETag: "1e-596be05bc9a34"
Keep-Alive: timeout=5, max=100
Last-Modified: Thu, 07 Nov 2019 09:09:27 GMT
Server: Apache/2.4.37 (centos)

<strong>马哥教育</strong>
```

范例：查看响应报文头部

```
[root@centos8 ~]#http HEAD  http://www.magedu.com
HTTP/1.1 200 OK
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Thu, 07 Nov 2019 08:09:49 GMT
Link: <http://www.magedu.com/wp-json/>; rel="https://api.w.org/"
Link: <http://www.magedu.com/>; rel=shortlink
Server: Tengine
Vary: Accept-Encoding
Vary: Accept-Encoding, Cookie
```

范例： 查看请求和响应报文头部

```
[root@centos8 ~]#http -p Hh  http://www.magedu.com
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: www.magedu.com
User-Agent: HTTPie/0.9.4

HTTP/1.1 200 OK
Cache-Control: max-age=3, must-revalidate
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Thu, 07 Nov 2019 03:44:14 GMT
Server: Tengine
```

```
Transfer-Encoding: chunked
Vary: Accept-Encoding
Vary: Accept-Encoding, Cookie
```

范例：指定请求头部的首部字段

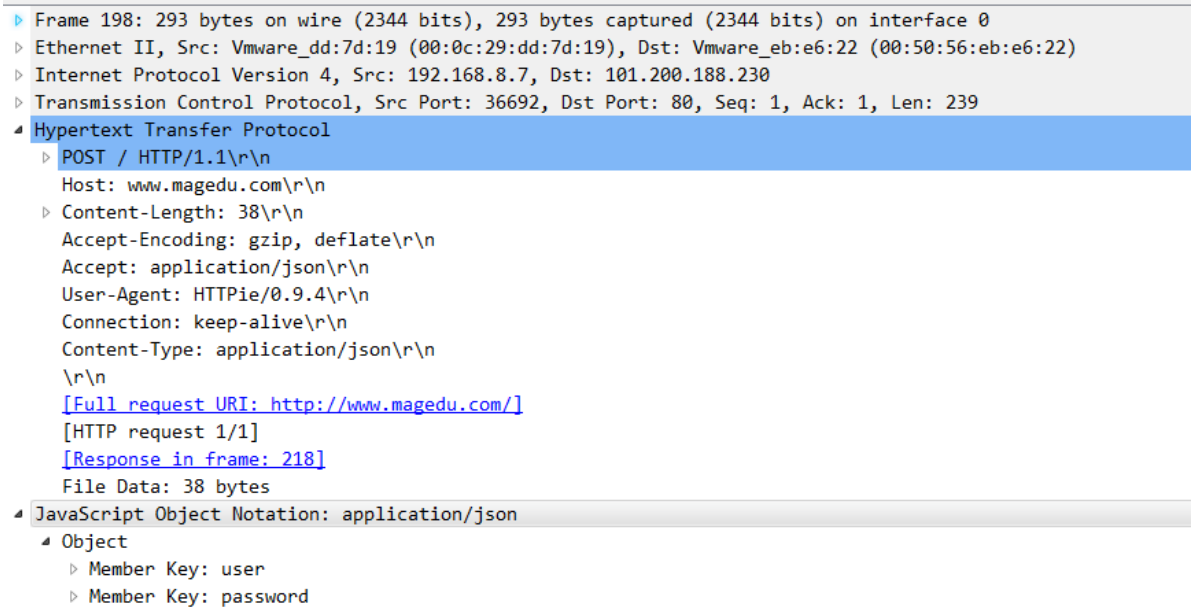
```
[root@centos8 ~]#http -p H  http://www.magedu.com  User-Agent:wangtest
Referer:http://www.baidu.com
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: www.magedu.com
Referer: http://www.baidu.com
User-Agent: wangtest
```

范例：下载资源

```
#方法1
[root@centos8 ~]#http http://www.magedu.com/wp-
content/uploads/2018/12/2018122312035677.png > logo.png
#方法2
[root@centos8 ~]#file logo.png
logo.png: PNG image data, 411 x 127, 8-bit/color RGBA, non-interlaced
[root@centos7 ~]#http --download   http://www.magedu.com/wp-
content/uploads/2018/12/2018122312035677.png
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Connection: keep-alive
Content-Length: 8983
Content-Type: image/png
Date: Thu, 07 Nov 2019 08:20:44 GMT
ETag: "5c1f79ac-2317"
Expires: Thu, 14 Nov 2019 08:20:44 GMT
Last-Modified: Sun, 23 Dec 2018 12:03:56 GMT
Server: Tengine
Vary: Accept-Encoding

Downloading 8.77 kB to "2018122312035677.png"
Done. 8.77 kB in 0.00091s (9.46 MB/s)
```

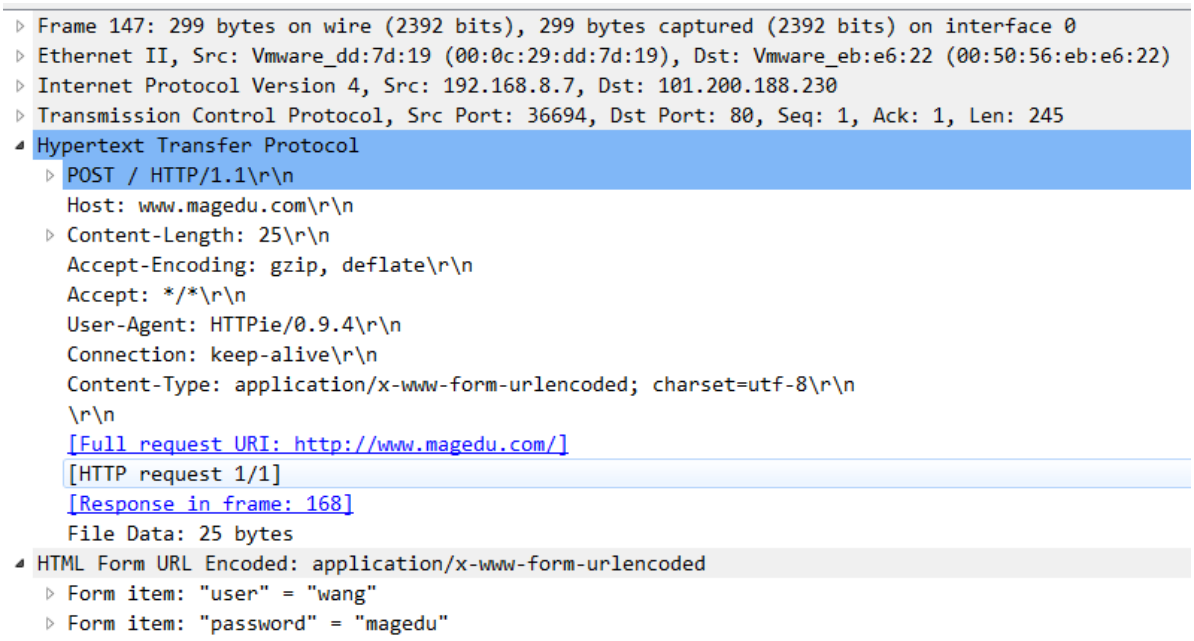
范例：用POST方法提交json格式的数据

```
[root@centos8 ~]#http http://www.magedu.com user=wang password=magedu
```



范例：用POST方法，指交表单数据

```
[root@centos8 ~]#http -f POST  http://www.magedu.com user=wang password=magedu
```



1.5.5 压力测试工具

httpd的压力测试工具：

- ab, webbench, http_load, seige
- Jmeter 开源
- Loadrunner 商业，有相关认证
- tcpcopy：网易，复制生产环境中的真实请求，并将之保存

ab 来自httpd-tools包

命令格式

```
ab  [OPTIONS]  URL
```

常见选项：

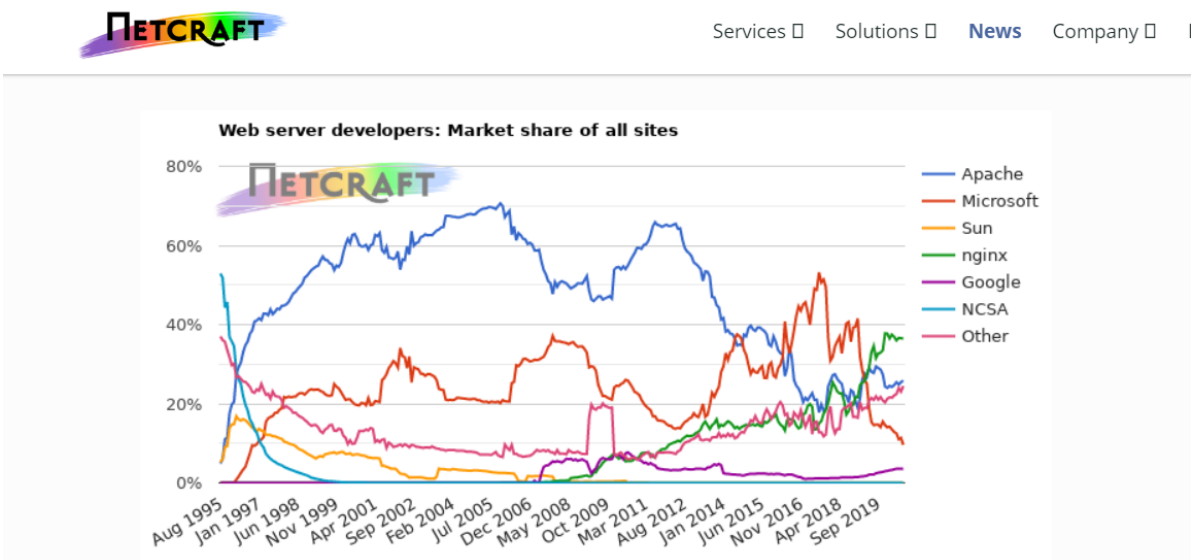
```
-n：总请求数
-c：模拟的并发数
-k：以持久连接模式测试
```

说明：并发数高于1024时，需要用 ulimit -n # 调整能打开的文件数

1.6 Web 服务介绍

Netcraft公司于1994年底在英国成立，多年来一直致力于互联网市场以及在线安全方面的咨询服务，其中在国际上最具影响力的当属其针对网站服务器，域名解析/主机提供商，以及SSL市场所做的客观严谨的分析研究。

https://news.netcraft.com/



Developer	July 2020	Percent	August 2020	Percent	Change
nginx	451,156,878	36.55%	448,602,806	36.45%	-0.10
Apache	314,054,523	25.45%	318,307,245	25.87%	0.42
Microsoft	140,264,332	11.36%	118,126,662	9.60%	-1.77
Google	44,290,430	3.59%	44,326,227	3.60%	0.01

1.6.1 Apache 经典的 Web 服务端

Apache起初由美国的伊利诺伊大学香槟分校的国家超级计算机应用中心开发，目前经历了两大版本分别是1.X和2.X，其可以通过编译安装实现特定的功能

官方网站：http://www.apache.org

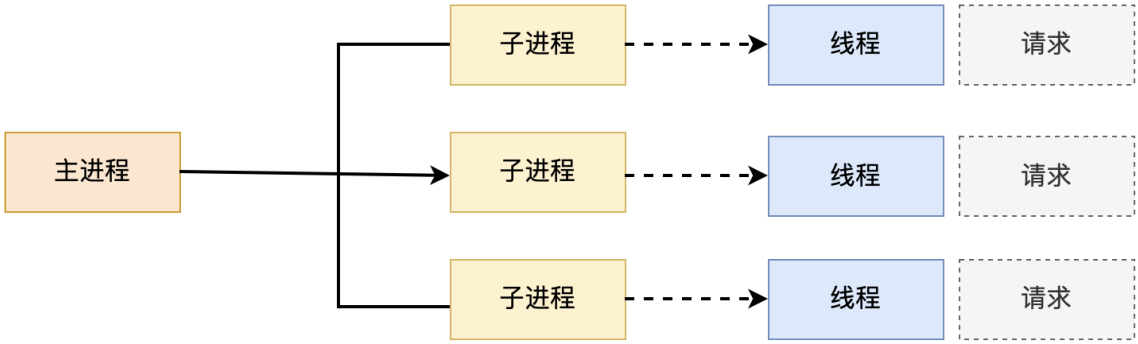
MPM multi-processing module 有三种工作模式

1.6.1.1 Apache prefork 模型

预派生模式，有一个主控制进程，然后生成多个子进程，每个子进程有一个独立的线程响应用户请求，相对比较占用内存，但是比较稳定，可以设置最大和最小进程数，是最古老的一种模式，也是最稳定的模式，适用于访问量不是很大的场景。

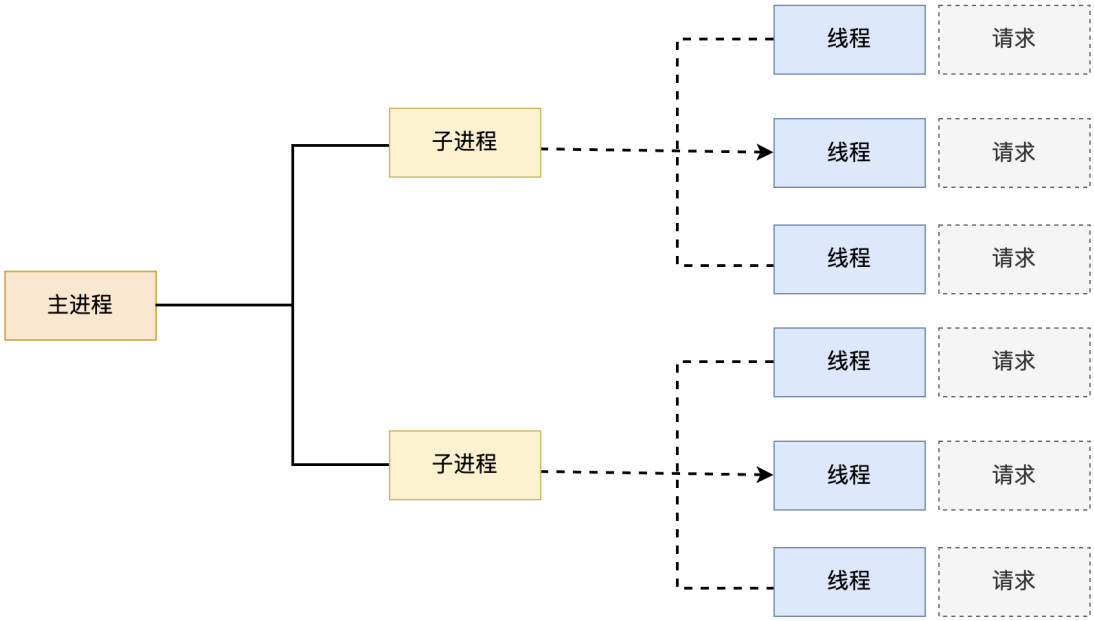
优点：稳定

缺点：每个用户请求需要对应开启一个进程,占用资源较多，并发性差,不适用于高并发场景



1.6.1.2 Apache worker 模型

一种多进程和多线程混合的模型，有一个控制进程，启动多个子进程，每个子进程里面包含固定的线程，使用线程程来处理请求，当线程不够使用的时候会再启动一个新的子进程，然后在进程里面再启动线程处理请求，由于其使用了线程处理请求，因此可以承受更高的并发。
优点：相比prefork 占用的内存较少，可以同时处理更多的请求
缺点：使用keepalive的长连接方式，某个线程会一直被占据，即使没有传输数据，也需要一直等待到超时才会被释放。如果过多的线程，被这样占据，也会导致在高并发场景下的无服务线程可用。（该问题在prefork模式下，同样会发生）

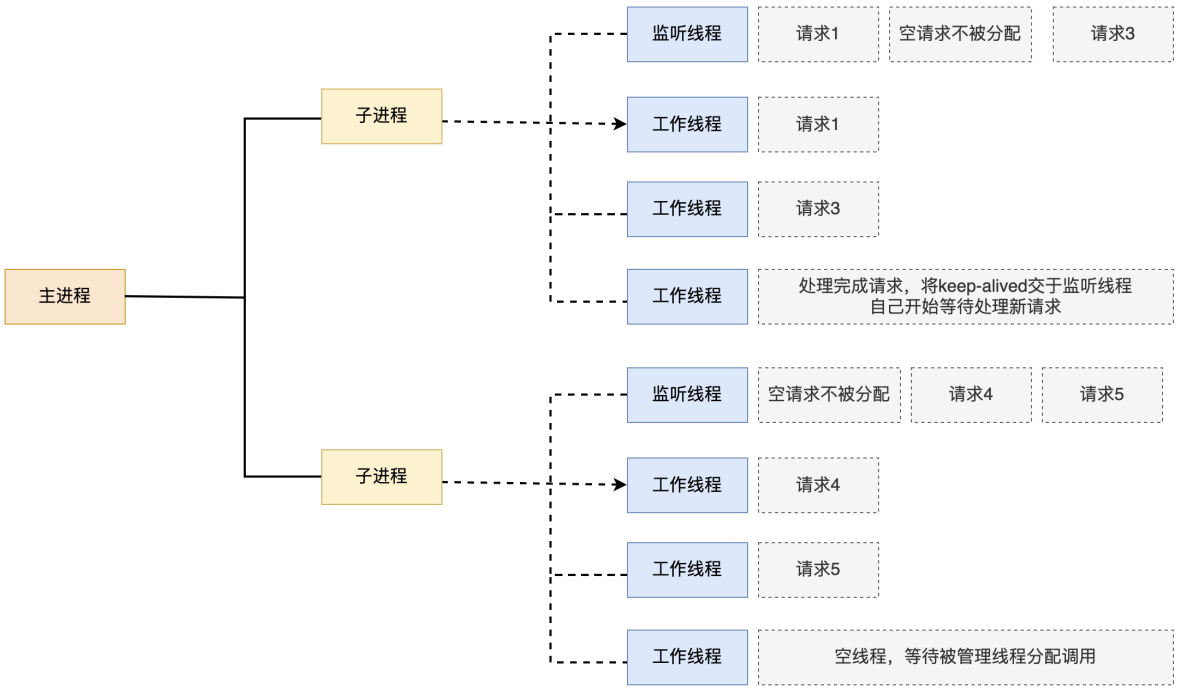


1.6.1.3 Apache event模型

Apache中最新的模式，2012年发布的apache 2.4.X系列正式支持event 模型，属于事件驱动模型(epoll)，每个进程响应多个请求，在现在版本里的已经是稳定可用的模式。它和worker模式很像，最大的区别在于，它解决了keepalive场景下，长期被占用的线程的资源浪费问题（某些线程因为被keepalive，空挂在哪里等待，中间几乎没有请求过来，甚至等到超时）。event MPM中，会有一个专门的线程来管理这些keepalive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放。这样增强了高并发场景下的请求处理能力。

优点：单线程响应多请求，占据更少的内存，高并发下表现更优秀，会有一个专门的线程来管理keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放

缺点：没有线程安全控制



1.6.2 Nginx-高性能的 Web 服务端

Nginx是由俄罗斯国立莫斯科鲍曼科技大学在1994年毕业的学生为俄罗斯rambler.ru公司开发的，开发工作最早从2002年开始，第一次公开发布时间是2004年10月4日，版本号是0.1.0

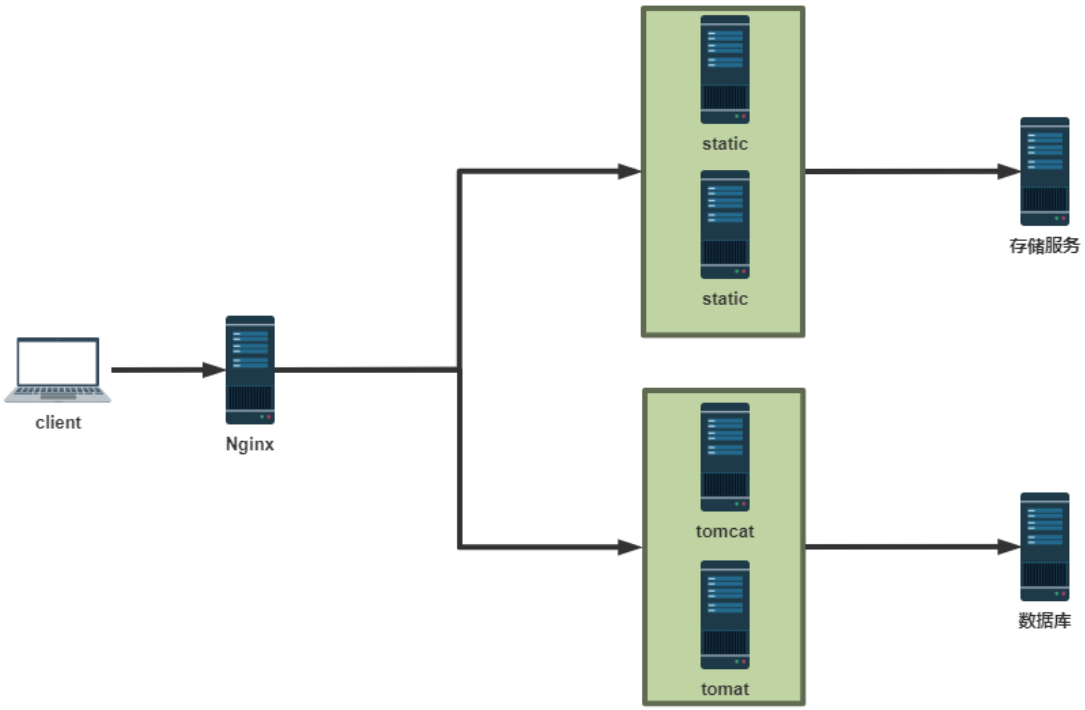
2019年3月11日F5 与 NGINX达成协议,F5 将收购 NGINX 的所有已发行股票，总价值约为 6.7 亿美元。6.7亿美金约合44.97亿人民币,nginx核心模块代码长度198430（包括空格、注释），所以一行代码约为2.2万人民币

官网地址 www.nginx.org

Nginx历经十几年的迭代更新（https://nginx.org/en/CHANGES）， 目前功能已经非常完善且运行稳定，分为社区版本和商业版，另外Nginx的社区版本分为开发版（奇数）、最新稳定版（偶数）和过期版，nginx以功能丰富著称，它即可以作为http服务器，也可以作为反向代理服务器或者邮件服务器，能够快速的响应静态网页的请求，支持FastCGI/SSL/Virtual Host/URL Rwrite/Gzip/HTTP Basic Auth/http或者TCP的负载均衡(1.9版本以上且开启stream模块)等功能，并且支持第三方的功能扩展。

天猫 淘宝 京东 小米 163 新浪等一线互联网公司都在用Nginx或者进行二次开发

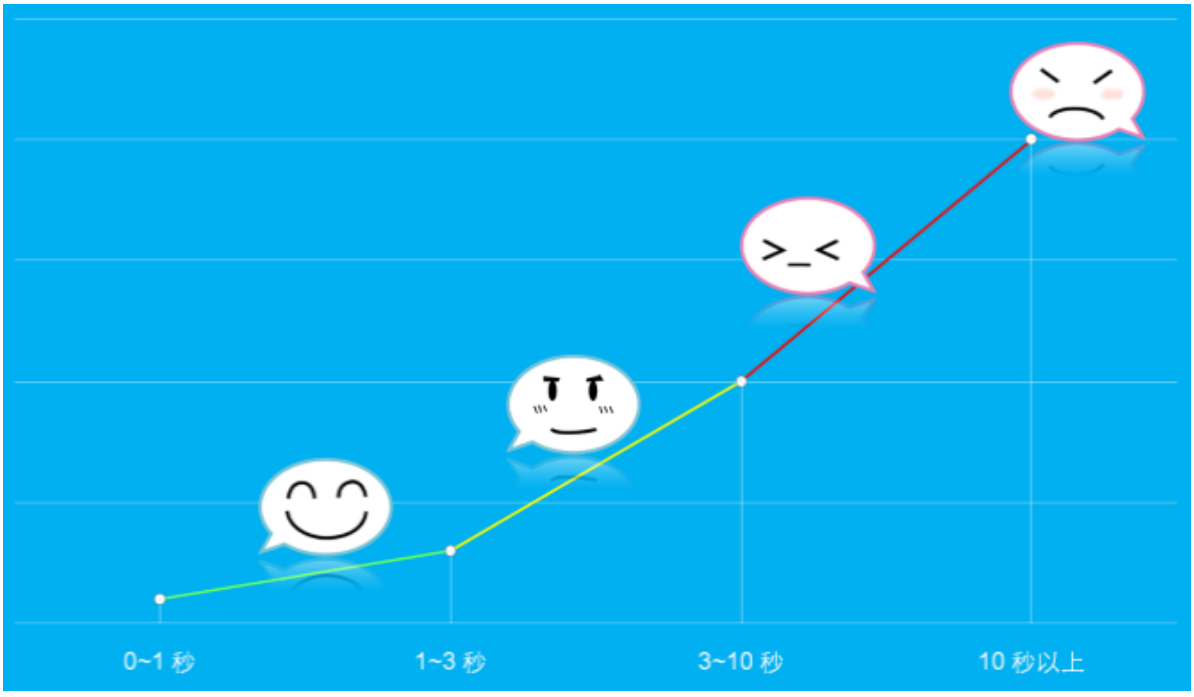
基于Nginx的工作场景：



1.6.3 用户访问体验和性能

1.6.3.1 用户访问体验统计

互联网存在用户速度体验的1-3-10原则，即1秒最优，1-3秒较优，3~10秒比较慢，10秒以上用户无法接受。用户放弃一个产品的代价很低，只是换一个URL而已。



全球最大搜索引擎 Google：慢500ms = 20% 将放弃访问。

全球最大的电商零售网站亚马逊：慢100ms = 1% 将放弃交易

有很多研究都表明，性能对用户的行为有很大的影响：

79%的用户表示不太可能再次打开一个缓慢的网站

47%的用户期望网页能在2秒钟以内加载

40%的用户表示如果加载时间超过三秒钟，就会放弃这个网站

页面加载时间延迟一秒可能导致转换损失7%，页面浏览量减少11%

8秒定律：用户访问一个网站时，如果等待网页打开的时间超过8秒，会有超过30%的用户放弃等待

请珍惜每一毫秒的时间！

1.6.3.2 影响用户体验的因素

据说马云在刚开始创业在给客户演示时，打开一个网站花了不到四个小时。

```
https://www.shuimiao.net/NjHaO/
```

影响用户体验的因素

```
#客户端
客户端硬件配置
客户端网络速率
客户端与服务端距离
#服务器
服务端网络速率
服务端硬件配置
服务端架构设计
服务端应用程序工作模式
服务端并发数量
服务端响应文件大小及数量 buffer cache
服务端I/O压力
```

1.6.4 服务端 I/O 流程

I/O在计算机中指Input/Output， IOPS (Input/Output Per Second)即每秒的输入输出量(或读写次数)，是衡量磁盘性能的主要指标之一。IOPS是指单位时间内系统能处理的I/O请求数量，一般以每秒处理的I/O请求数量为单位，I/O请求通常为读或写数据操作请求。

一次完整的I/O是用户空间的进程数据与内核空间的内核数据的报文的完整交换，但是由于内核空间与用户空间是严格隔离的，所以其数据交换过程中不能由用户空间的进程直接调用内核空间的内存数据，而是需要经历一次从内核空间中的内存数据copy到用户空间的进程内存当中，所以简单说I/O就是把数据从内核空间中的内存数据复制到用户空间中进程的内存当中。

Linux 的 I/O

- 磁盘I/O
- 网络I/O：一切皆文件,本质为对socket文件的读写

1.6.4.1 磁盘 I/O

磁盘I/O是进程向内核发起系统调用，请求磁盘上的某个资源比如是html 文件或者图片，然后内核通过相应的驱动程序将目标文件加载到内核的内存空间，加载完成之后把数据从内核内存再复制给进程内存，如果是比较大的数据也需要等待时间

```
机械磁盘的寻道时间、旋转延迟和数据传输时间：
寻道时间：是指磁头移动到正确的磁道上所花费的时间，寻道时间越短则I/O处理就越快，目前磁盘的寻道时间一般在3-15毫秒左右。

旋转延迟：是指将磁盘片旋转到数据所在的扇区到磁头下面所花费的时间，旋转延迟取决于磁盘的转速，通常使用磁盘旋转一周所需要时间的1/2之一表示，比如7200转的磁盘平均训传延迟大约为60*1000/7200/2=4.17毫秒，公式的意思为 （每分钟60秒*1000毫秒每秒/7200转每分/2），如果是15000转的则为60*1000/15000/2=2毫秒。
```

数据传输时间：指的是读取到数据后传输数据的时间，主要取决于传输速率，这个值等于数据大小除以传输速率，目前的磁盘接口每秒的传输速度可以达到600MB，因此可以忽略不计。

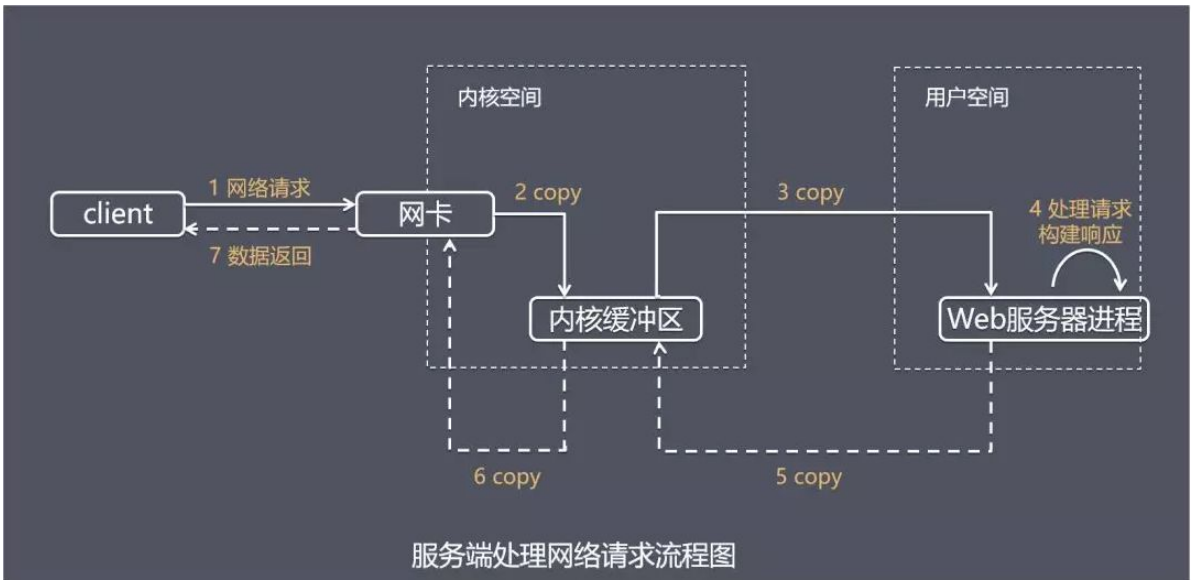
常见的机械磁盘平均寻道时间值：
7200转/分的磁盘平均物理寻道时间：9毫秒
10000转/分的磁盘平均物理寻道时间：6毫秒
15000转/分的磁盘平均物理寻道时间：4毫秒

常见磁盘的平均延迟时间：
7200转的机械盘平均延迟：60*1000/7200/2 = 4.17ms
10000转的机械盘平均延迟：60*1000/10000/2 = 3ms
15000转的机械盘平均延迟：60*1000/15000/2 = 2ms

每秒最大IOPS(每秒读写次数Input/OutputOperations Per Second)的计算方法：
7200转的磁盘IOPS计算方式：1000毫秒/(9毫秒的寻道时间+4.17毫秒的平均旋转延迟时间)=1000/13.13=75.9 IOPS
10000转的磁盘的IOPS计算方式：1000毫秒/(6毫秒的寻道时间+3毫秒的平均旋转延迟时间)=1000/9=111 IOPS
15000转的磁盘的IOPS计算方式：15000毫秒/(4毫秒的寻道时间+2毫秒的平均旋转延迟时间)=1000/6=166.6 IOPS

1.6.4.2 网络 I/O

网络通信就是网络协议栈到用户空间进程的IO就是网络IO



网络I/O 处理过程

获取请求数据，客户端与服务器建立连接发出请求，服务器接受请求（1-3）
构建响应，当服务器接收完请求，并在用户空间处理客户端的请求，直到构建响应完成（4）
返回数据，服务器将已构建好的响应再通过内核空间的网络 I/O 发还给客户端（5-7）

不论磁盘和网络I/O

每次I/O，都要经由两个阶段：
第一步：将数据从文件先加载至内核内存空间（缓冲区），等待数据准备完成，时间较长
第二步：将数据从内核缓冲区复制到用户空间的进程的内存中，时间较短

1.7 I/O 模型

1.7.1 I/O 模型相关概念

同步/异步：关注的是消息通信机制，即调用者在等待一件事情的处理结果时，被调用者是否提供完成状态的通知。

- 同步：synchronous，被调用者并不提供事件的处理结果相关的通知消息，需要调用者主动询问事情是否处理完成
- 异步：asynchronous，被调用者通过状态、通知或回调机制主动通知调用者被调用者的运行状态

阻塞/非阻塞：关注调用者在等待结果返回之前所处的状态

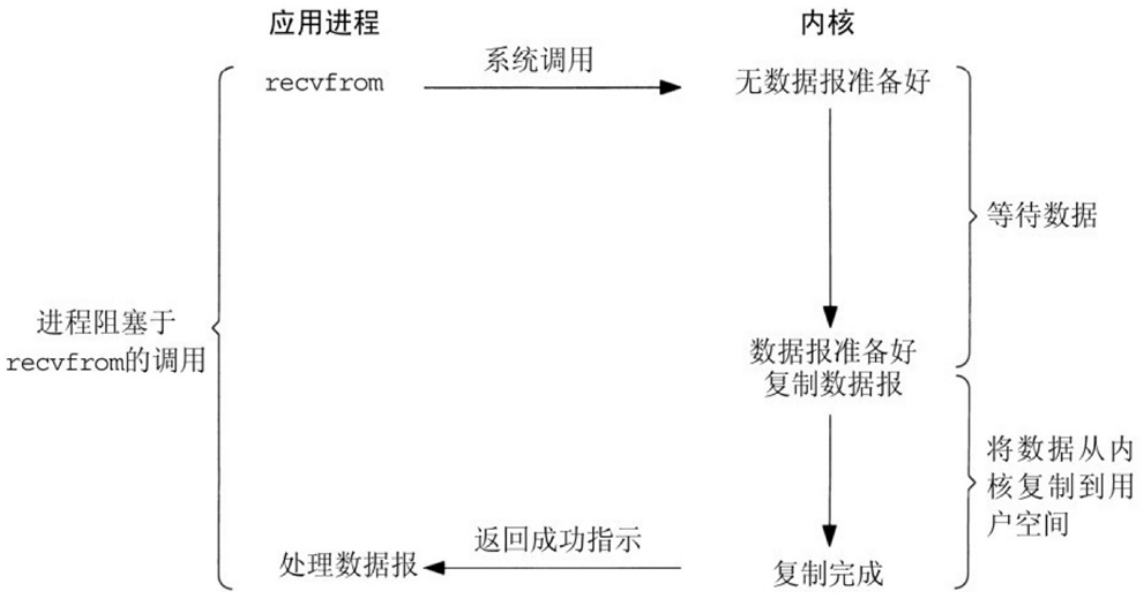
- 阻塞：blocking，指IO操作需要彻底完成后才返回到用户空间，调用结果返回之前，调用者被挂起，干不了别的事情。
- 非阻塞：nonblocking，指IO操作被调用后立即返回给用户一个状态值，而无需等到IO操作彻底完成，在最终的调用结果返回之前，调用者不会被挂起，可以去做别的事情。

1.7.2 网络 I/O 模型

阻塞型、非阻塞型、复用型、信号驱动型、异步

参看:《UNIX网络编程 卷1：套接字联网API 》(美)W. Richard Stevens 著

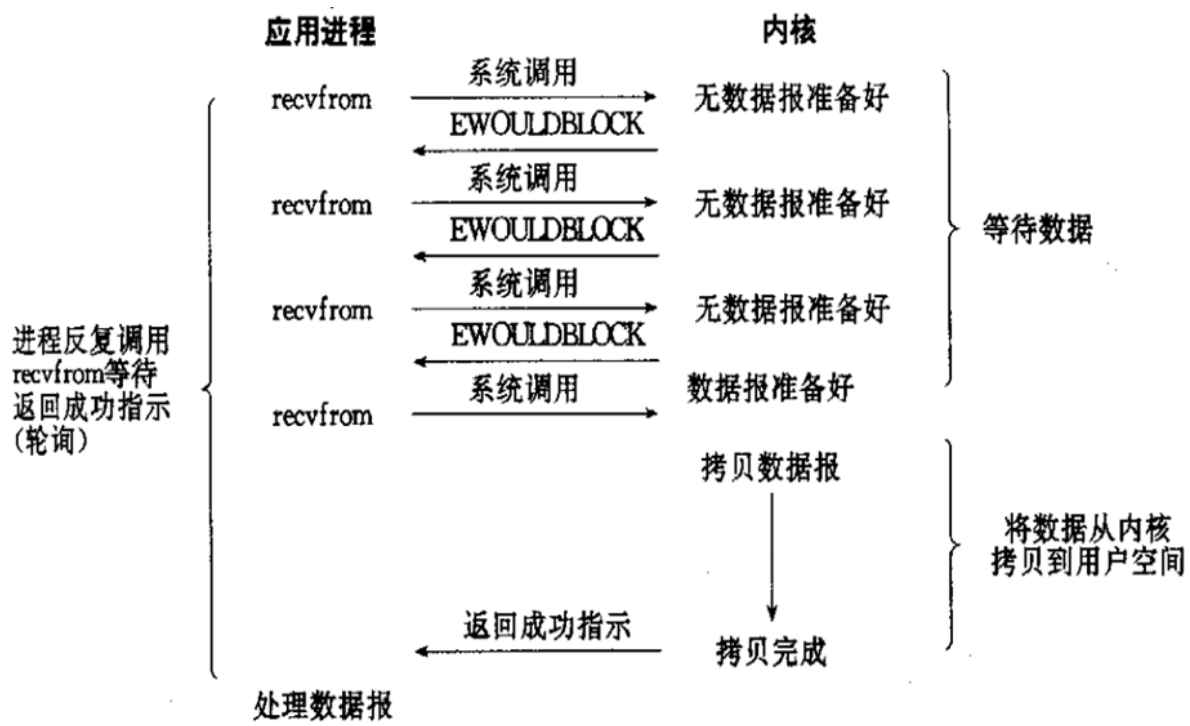
1.7.2.1 阻塞型 I/O 模型（blocking IO）



阻塞IO模型是最简单的I/O模型，用户线程在内核进行IO操作时被阻塞
用户线程通过系统调用read发起I/O读操作，由用户空间转到内核空间。内核等到数据包到达后，然后将接收的数据拷贝到用户空间，完成read操作
用户需要等待read将数据读取到buffer后，才继续处理接收的数据。整个I/O请求的过程中，用户线程是被阻塞的，这导致用户在发起IO请求时，不能做任何事情，对CPU的资源利用率不够
优点：程序简单，在阻塞等待数据期间进程/线程挂起，基本不会占用 CPU 资源
缺点：每个连接需要独立的进程/线程单独处理，当并发请求量大时为了维护程序，内存、线程切换开销较大，apache 的prefork使用的是这种模式。

> 同步阻塞：程序向内核发送I/O请求后一直等待内核响应，如果内核处理请求的IO操作不能立即返回,则进程将一直等待并不再接受新的请求，并由进程轮询查看I/O是否完成，完成后进程将I/O结果返回给Client，在IO没有返回期间进程不能接受其他客户的请求，而且是有进程自己去查看I/O是否完成，这种方式简单，但是比较慢，用的比较少。

1.7.2.2 非阻塞型 I/O 模型 (nonblocking IO)



用户线程发起IO请求时立即返回。但并未读取到任何数据，用户线程需要不断地发起IO请求，直到数据到达后，才真正读取到数据，继续执行。即 “轮询”机制存在两个问题：如果有大量文件描述符都要等，那么就得一个一个的read。这会带来大量的Context Switch（read是系统调用，每调用一次就得在用户态和核心态切换一次）。轮询的时间不好把握。这里是要猜多久之后数据才能到。等待时间设的太长，程序响应延迟就过大;设的太短，就会造成过于频繁的重试，干耗CPU而已，是比较浪费CPU的方式，一般很少直接使用这种模型，而是在其他IO模型中使用非阻塞IO这一特性。

> 非阻塞：程序向内核发送请I/O求后一直等待内核响应，如果内核处理请求的IO操作不能立即返回IO结果，进程将不再等待，而且继续处理其他请求，但是仍然需要进程隔一段时间就要查看内核I/O是否完成。

查看上图可知，在设置连接为非阻塞时，当应用进程系统调用 `recvfrom` 没有数据返回时，内核会立即返回一个 `EWOULDBLOCK` 错误，而不会一直阻塞到数据准备好。如上图在第四次调用时有一个数据报准备好了，所以这时数据会被复制到 `应用进程缓冲区` ，于是 `recvfrom` 成功返回数据

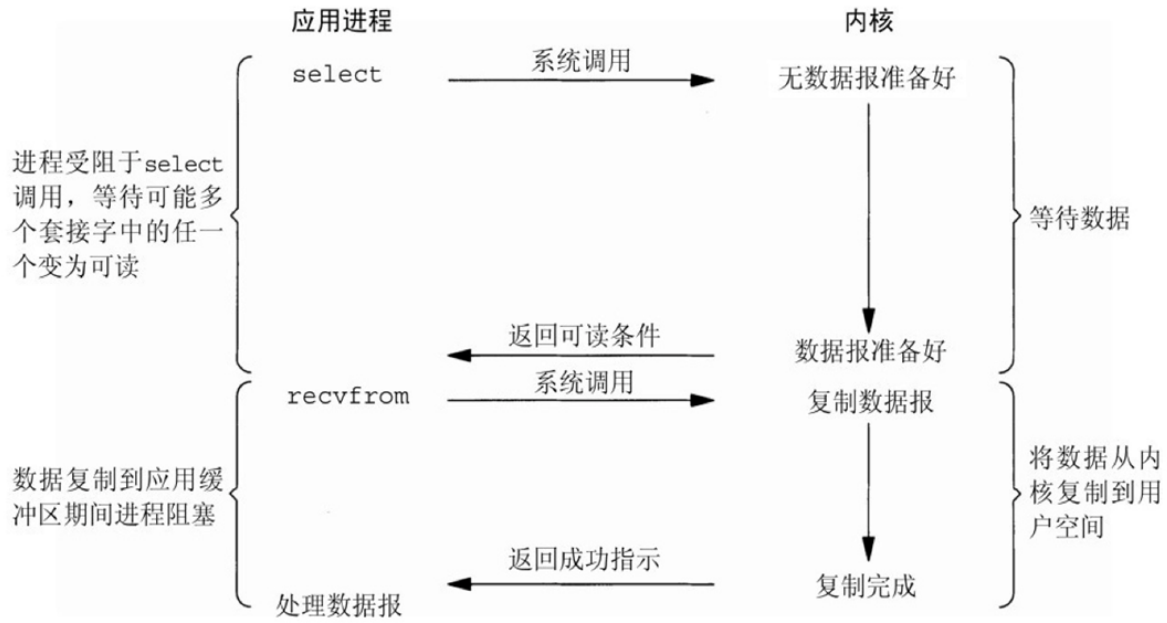
当一个应用进程这样循环调用 `recvfrom` 时，称之为轮询 `polling` 。这么做往往会耗费大量CPU时间，实际使用很少

1.7.2.3 I/O 多路复用型 (I/O multiplexing)

上面的模型中,每一个文件描述符对应的IO是由一个线程监控和处理

多路复用IO指一个线程可以同时（实际是交替实现，即并发完成）监控和处理多个文件描述符对应各自的IO，即复用同一个线程

一个线程之所以能实现同时处理多个IO,是因为这个线程调用了内核中的SELECT,POLL或EPOLL等系统调用，从而实现多路复用IO



I/O multiplexing 主要包括:select，poll，epoll三种系统调用，select/poll/epoll的好处就在于单个process就可以同时处理多个网络连接的IO。

它的基本原理就是select/poll/epoll这个function会不断的轮询所负责的所有socket，当某个socket有数据到达了，就通知用户进程。

当用户进程调用了select，那么整个进程会被block，而同时，kernel会“监视”所有select负责的socket，当任何一个socket中的数据准备好了，select就会返回。这个时候用户进程再调用read操作，将数据从kernel拷贝到用户进程。

Apache prefork是此模式的select，worker是poll模式。

> IO多路复用（IO Multiplexing）：是一种机制，程序注册一组socket文件描述符给操作系统，表示“我要监视这些fd是否有IO事件发生，有了就告诉程序处理”
> IO多路复用一般和NIO一起使用的。NIO和IO多路复用是相对独立的。NIO仅仅是指IO API总是能立刻返回，不会被Blocking;而IO多路复用仅仅是操作系统提供的一种便利的通知机制。操作系统并不会强制这俩必须得一起用，可以只用IO多路复用 + BIO，这时还是当前线程被卡住。IO多路复用和NIO是要配合一起使用才有实际意义
> IO多路复用是指内核一旦发现进程指定的一个或者多个IO条件准备读取，就通知该进程
> 多个连接共用一个等待机制，本模型会阻塞进程，但是进程是阻塞在select或者poll这两个系统调用上，而不是阻塞在真正的IO操作上
> 用户首先将需要进行IO操作添加到select中，同时等待select系统调用返回。当数据到达时，IO被激活，select函数返回。用户线程正式发起read请求，读取数据并继续执行
> 从流程上来看，使用select函数进行IO请求和同步阻塞模型没有太大的区别，甚至还多了添加监视IO，以及调用select函数的额外操作，效率更差。并且阻塞了两次，但是第一次阻塞在select上时，select可以监控多个IO上是否已有IO操作准备就绪，即可达到在同一个线程内同时处理多个IO请求的目的。而不像阻塞IO那种，一次只能监控一个IO
> 虽然上述方式允许单线程内处理多个IO请求，但是每个IO请求的过程还是阻塞的（在select函数上阻塞），平均时间甚至比同步阻塞IO模型还要长。如果用户线程只是注册自己需要的IO请求，然后去做自己的事情，等到数据到来时再进行处理，则可以提高CPU的利用率
> IO多路复用是最常使用的IO模型，但是其异步程度还不够“彻底”，因它使用了会阻塞线程的select系统调用。因此IO多路复用只能称为异步阻塞IO模型，而非真正的异步IO

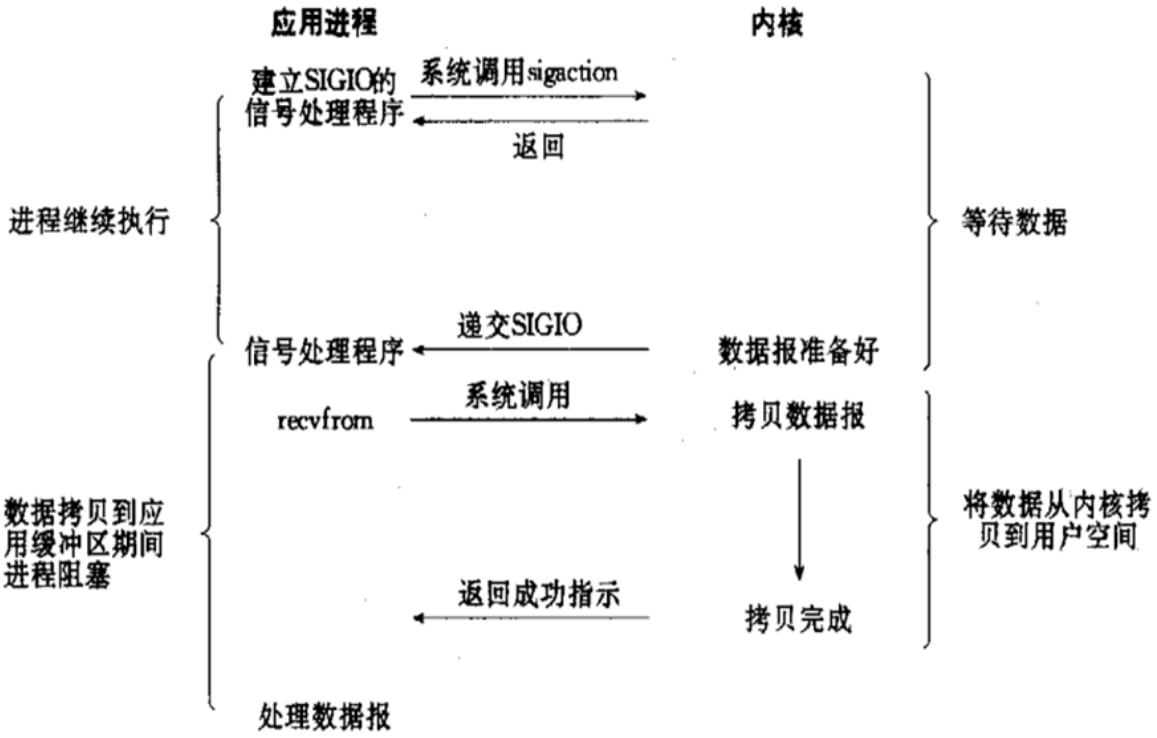
优缺点

- 优点：可以基于一个阻塞对象，同时在多个描述符上等待就绪，而不是使用多个线程(每个文件描述符一个线程)，这样可以大大节省系统资源
- 缺点：当连接数较少时效率相比多线程+阻塞 I/O 模型效率较低，可能延迟更大，因为单个连接处理需要 2 次系统调用，占用时间会有增加

IO多路复用适用如下场合：

- 当客户端处理多个描述符时（一般是交互式输入和网络套接口），必须使用I/O复用
- 当一个客户端同时处理多个套接字时，此情况可能的但很少出现
- 当一个服务器既要处理监听套接字，又要处理已连接套接字，一般也要用到I/O复用
- 当一个服务器即要处理TCP，又要处理UDP，一般要使用I/O复用
- 当一个服务器要处理多个服务或多个协议，一般要使用I/O复用

1.7.2.4 信号驱动式 I/O 模型 (signal-driven IO)



信号驱动I/O的意思就是进程现在不用傻等着，也不用去轮询。而是让内核在数据就绪时，发送信号通知进程。

调用的步骤是，通过系统调用`sigaction`，并注册一个信号处理的回调函数，该调用会立即返回，然后主程序可以继续向下执行，当有I/O操作准备就绪,即内核数据就绪时，内核会为该进程产生一个`SIGIO`信号，并回调注册的信号回调函数，这样就可以在信号回调函数中系统调用`recvfrom`获取数据,将用户进程所需要的数据从内核空间拷贝到用户空间

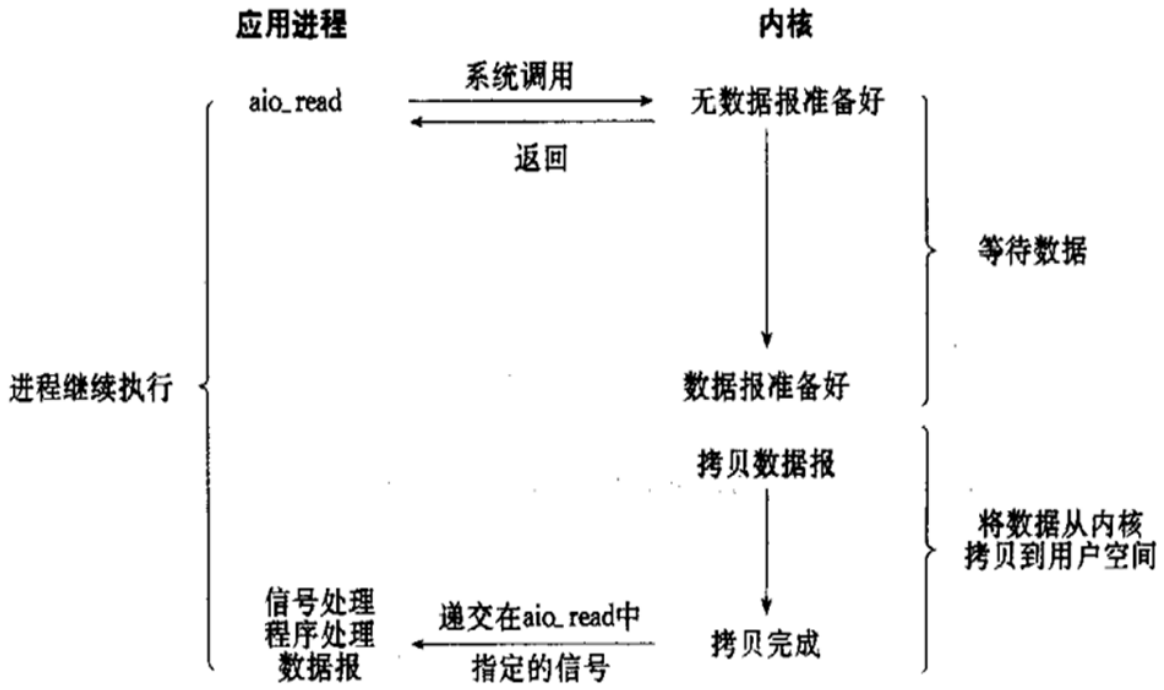
此模型的优势在于等待数据报到达期间进程不被阻塞。用户主程序可以继续执行，只要等待来自信号处理函数的通知。

在信号驱动式 I/O 模型中，应用程序使用套接口进行信号驱动 I/O，并安装一个信号处理函数，进程继续运行并不阻塞

当数据准备好时，进程会收到一个 SIGIO 信号，可以在信号处理函数中调用 I/O 操作函数处理数据。优点：线程并没有在等待数据时被阻塞，内核直接返回调用接收信号，不影响进程继续处理其他请求因此可以提高资源的利用率
缺点：信号 I/O 在大量 IO 操作时可能会因为信号队列溢出导致没法通知

> 异步阻塞：程序进程向内核发送IO调用后，不用等待内核响应，可以继续接受其他请求，内核收到进程请求后进行的IO如果不能立即返回，就由内核等待结果，直到IO完成后内核再通知进程。

1.7.2.5 异步 I/O 模型 (asynchronous IO)



异步I/O 与 信号驱动I/O最大区别在于，信号驱动是内核通知用户进程何时开始一个I/O操作，而异步I/O是由内核通知用户进程I/O操作何时完成，两者有本质区别,相当于不用去饭店场吃饭，直接点个外卖，把等待上菜的时间也给省了

相对于同步I/O，异步I/O不是顺序执行。用户进程进行aio_read系统调用之后，无论内核数据是否准备好，都会直接返回给用户进程，然后用户态进程可以去做别的事情。等到socket数据准备好了，内核直接复制数据给进程，然后从内核向进程发送通知。IO两个阶段，进程都是非阻塞的。

信号驱动IO当内核通知触发信号处理程序时，信号处理程序还需要阻塞在从内核空间缓冲区拷贝数据到用户空间缓冲区这个阶段，而异步IO直接是在第二个阶段完成后，内核直接通知用户线程可以进行后续操作了

优点：异步 I/O 能够充分利用 DMA 特性，让 I/O 操作与计算重叠

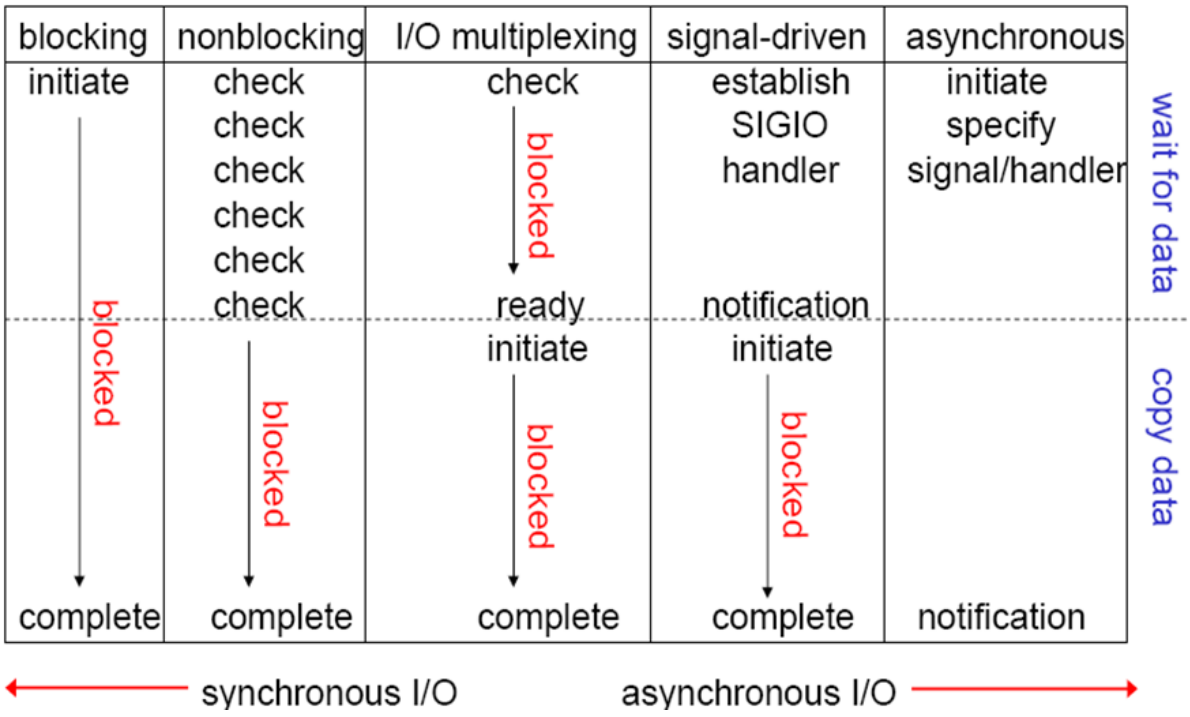
缺点：要实现真正的异步 I/O，操作系统需要做大量的工作。目前 Windows 下通过 IOCP 实现了真正的异步 I/O，在 Linux 系统下，Linux 2.6才引入，目前 AIO 并不完善，因此在 Linux 下实现高并发网络编程时以 IO 复用模型模式+多线程任务的架构基本可以满足需求

Linux提供了AIO库函数实现异步，但是用的很少。目前有很多开源的异步IO库，例如libevent、libev、libuv。

> 异步非阻塞：程序进程向内核发送IO调用后，不用等待内核响应，可以继续接受其他请求，内核调用的IO如果不能立即返回，内核会继续处理其他事物，直到IO完成后将结果通知给内核，内核在将IO完成的结果返回给进程，期间进程可以接受新的请求，内核也可以处理新的事物，因此相互不影响，可以实现较大的同时并实现较高的IO复用，因此异步非阻塞使用最多的一种通信方式。

1.7.3 五种 IO 对比

这五种 I/O 模型中，越往后，阻塞越少，理论上效率也是最优前四种属于同步 I/O，因为其中真正的 I/O 操作(recvfrom)将阻塞进程/线程，只有异步 I/O 模型才与 POSIX 定义的异步 I/O 相匹配



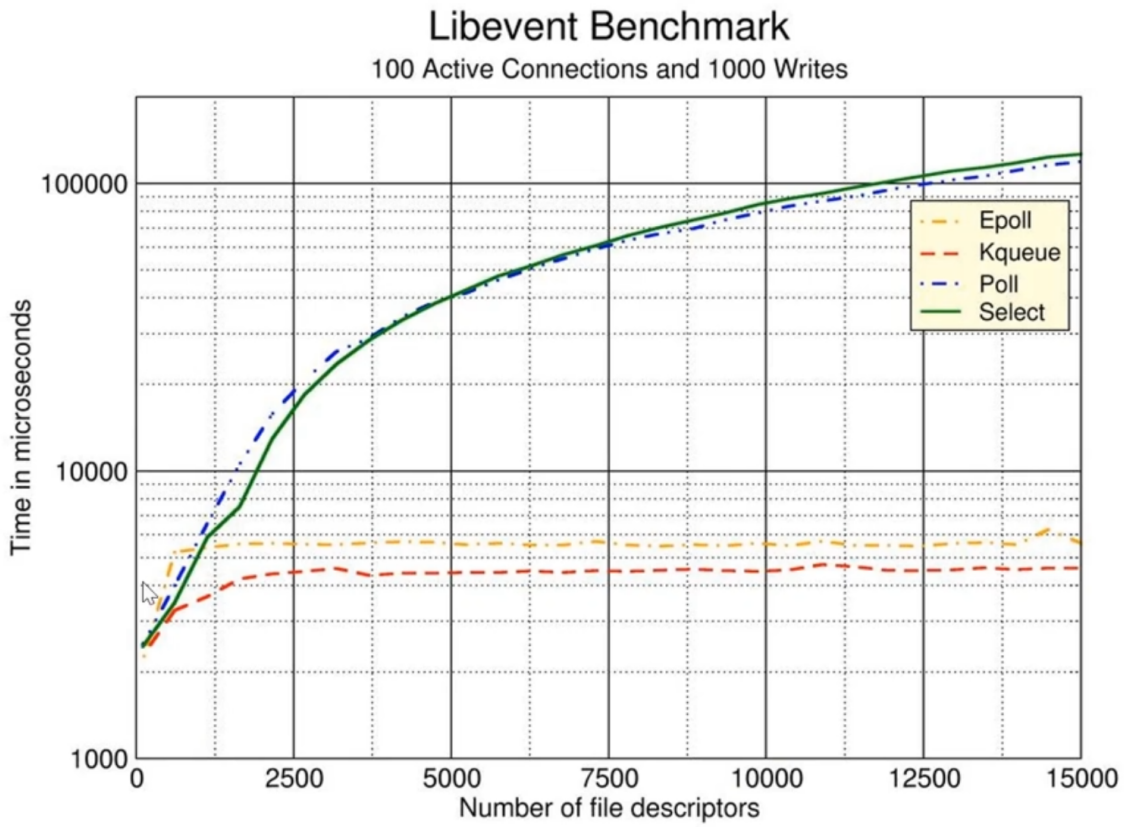
1.7.4 I/O 的具体实现方式

1.7.4.1 I/O常见实现

Nginx支持在多种不同的操作系统实现不同的事件驱动模型，但是其在不同的操作系统甚至是不同的系统版本上面的实现方式不尽相同，主要有以下实现方式：

1、select：
select库是在linux和windows平台都基本支持的 事件驱动模型库，并且在接口的定义也基本相同，只是部分参数的含义略有差异，最大并发限制1024，是最早期的事件驱动模型。
2、poll：
在Linux 的基本驱动模型，windows不支持此驱动模型，是select的升级版，取消了最大的并发限制，在编译nginx的时候可以使用--with-poll_module和--without-poll_module这两个指定是否编译select库。
3、epoll：
epoll是库是Nginx服务器支持的最高性能的事件驱动库之一，是公认的非常优秀的事件驱动模型，它和select和poll有很大的区别，epoll是poll的升级版，但是与poll有很大的区别.
epoll的处理方式是创建一个待处理的事件列表，然后把这个列表发给内核，返回的时候在去轮询检查这个表，以判断事件是否发生，epoll支持一个进程打开的最大事件描述符的上限是系统可以打开的文件的最大数，同时epoll库的I/O效率不随描述符数目增加而线性下降，因为它只会对内核上报的“活跃”的描述符进行操作。
4、kqueue：
用于支持BSD系列平台的高校事件驱动模型，主要用在FreeBSD 4.1及以上版本、OpenBSD 2.0级以上版本，NetBSD级以上版本及Mac OS X 平台上，该模型也是poll库的变种，因此和epoll没有本质上的区别，效效率和epoll相似
5、IOCP：
Windows系统上的实现方式，对应第5种（异步I/O）模型。
6、/dev/poll:
用于支持unix衍生平台的高效事件驱动模型，主要在Solaris 平台、HP/UX，该模型是sun公司在开发Solaris系列平台的时候提出的用于完成事件驱动机制的方案，它使用了虚拟的/dev/poll设备，开发人员将要见识的文件描述符加入这个设备，然后通过ioctl()调用来获取事件通知，因此运行在以上系列平台的时候请使用/dev/poll事件驱动机制。效率和epoll相似
7、rtsig：
不是一个常用事件驱动，最大队列1024，不是很常用
8、eventport：
该方案也是sun公司在开发Solaris的时候提出的事件驱动库，只是Solaris 10以上的版本，该驱动库可防止内核崩溃等情况的发生。

1.7.4.2 常用I/O模型比较



	select	poll	epoll
操作方式	遍历	遍历	回调
底层实现	数组	链表	哈希表
IO效率	每次调用都进行线性遍历，时间复杂度为O(n)	同左	事件通知方式，每当fd就绪，系统注册的回调函数就会被调用，将就绪的fd放到rdllist里，时间复杂度O(1)
最大连接数	1024(x86) 2048(x64)	无上限	无上限
fd拷贝	每次调用select都需要把fd集合从用户拷贝到内核态	每次调用poll，都需要把fd集合从用户态拷贝到内核态	调用epoll_ctl时拷贝进内核并保存，之后每次epoll_wait不拷贝

Select：
POSIX所规定，目前几乎在所有的平台上支持，其良好跨平台支持也是它的一个优点，本质上是通过设置或者检查存放fd标志位的数据结构来进行下一步处理
缺点
单个进程能够监视的文件描述符的数量存在最大限制，在Linux上一般为1024，可以通过修改宏定义FD_SETSIZE，再重新编译内核实现，但是这样也会造成效率的降低
单个进程可监视的fd数量被限制，默认是1024，修改此值需要重新编译内核
对socket是线性扫描，即采用轮询的方法，效率较低
select 采取了内存拷贝方法来实现内核将 FD 消息通知给用户空间，这样一个用来存放大量fd的数据结构，这样会使得用户空间和内核空间在传递该结构时复制开销大

poll：
本质上和select没有区别，它将用户传入的数组拷贝到内核空间，然后查询每个fd对应的设备状态
其没有最大连接数的限制，原因是它是基于链表来存储的
大量的fd的数组被整体复制于用户态和内核地址空间之间，而不管这样的复制是不是有意义
poll特点是“水平触发”，如果报告了fd后，没有被处理，那么下次poll时会再次报告该fd
select是边缘触发即只通知一次

epoll：
在Linux 2.6内核中提出的select和poll的增强版本
支持水平触发LT和边缘触发ET，最大的特点在于边缘触发，它只告诉进程哪些fd刚刚变为就绪态，并且只会通知一次
使用“事件”的就绪通知方式，通过epoll_ctl注册fd，一旦该fd就绪，内核就会采用类似callback的回调机制来激活该fd，epoll_wait便可以收到通知
优点：
没有最大并发连接的限制：能打开的FD的上限远大于1024(1G的内存能监听约10万个端口)，具体查看/proc/sys/fs/file-max，此值和系统内存大小相关
效率提升：非轮询的方式，不会随着FD数目的增加而效率下降;只有活跃可用的FD才会调用callback函数，即epoll最大的优点就在于它只管理“活跃”的连接，而跟连接总数无关
内存拷贝，利用mmap(Memory Mapping)加速与内核空间的消息传递;即epoll使用mmap减少复制开销

总结:

1、epoll只是一组API，比起select这种扫描全部的文件描述符，epoll只读取就绪的文件描述符
2、epoll比select等多路复用方式来说，减少了遍历循环及内存拷贝的工作量，因为活跃连接只占总并发连接的很小一部分。
3、epoll基于事件的就绪通知机制，使用回调机制,所以性能比较好
4、基于epoll的IO多路复用减少了进程间切换的次数，使得操作系统少做了相对于用户任务来说的无用功。

1.7.4.3 开发的选择

- 完全跨平台，可以使用select、poll。但是性能较差
- 针对不同操作系统自行选择支持的最优技术，比如: Linux选用epoll,Mac选用kqueue,Windows 选用IOCP,提高IO处理的性能

范例: 最大并发连接数和内存有直接关系

```
#内存1G
[root@centos8 ~]#free -h
               total        used        free      shared  buff/cache   available
Mem:           952Mi       168Mi       605Mi        12Mi       178Mi       629Mi
Swap:          2.0Gi          0B       2.0Gi

[root@centos8 ~]#cat /proc/sys/fs/file-max
92953

#内存2G
[root@centos8 ~]#free -h
               total        used        free      shared  buff/cache   available
Mem:           1.9Gi       258Mi       1.3Gi        12Mi       341Mi       1.6Gi
Swap:          2.0Gi          0B       2.0Gi
[root@centos8 ~]#cat /proc/sys/fs/file-max
195920
```

范例: 内核限制

```
[root@centos8 ~]#grep -R FD_SETSIZE linux-5.8/*
linux-5.8/Documentation/userspace-api/media/v4l/func-select.rst:
 ``FD_SETSIZE``.
linux-5.8/include/uapi/linux/posix_types.h:#undef __FD_SETSIZE
linux-5.8/include/uapi/linux/posix_types.h:#define __FD_SETSIZE 1024  #单个进程能够
监视的文件描述符的文件最大数量
linux-5.8/include/uapi/linux/posix_types.h: unsigned long fds_bits[__FD_SETSIZE
/ (8 * sizeof(long))];
linux-5.8/tools/include/nolibc/nolibc.h:#define FD_SETSIZE 256
linux-5.8/tools/include/nolibc/nolibc.h:typedef struct { uint32_t
fd32[FD_SETSIZE/32]; } fd_set;
linux-5.8/tools/include/nolibc/nolibc.h:    if (fd < 0 || fd >= FD_SETSIZE)
linux-5.8/tools/testing/selftests/net/nettest.c:    rc = select(FD_SETSIZE,
NULL, &wfd, NULL, tv);
```

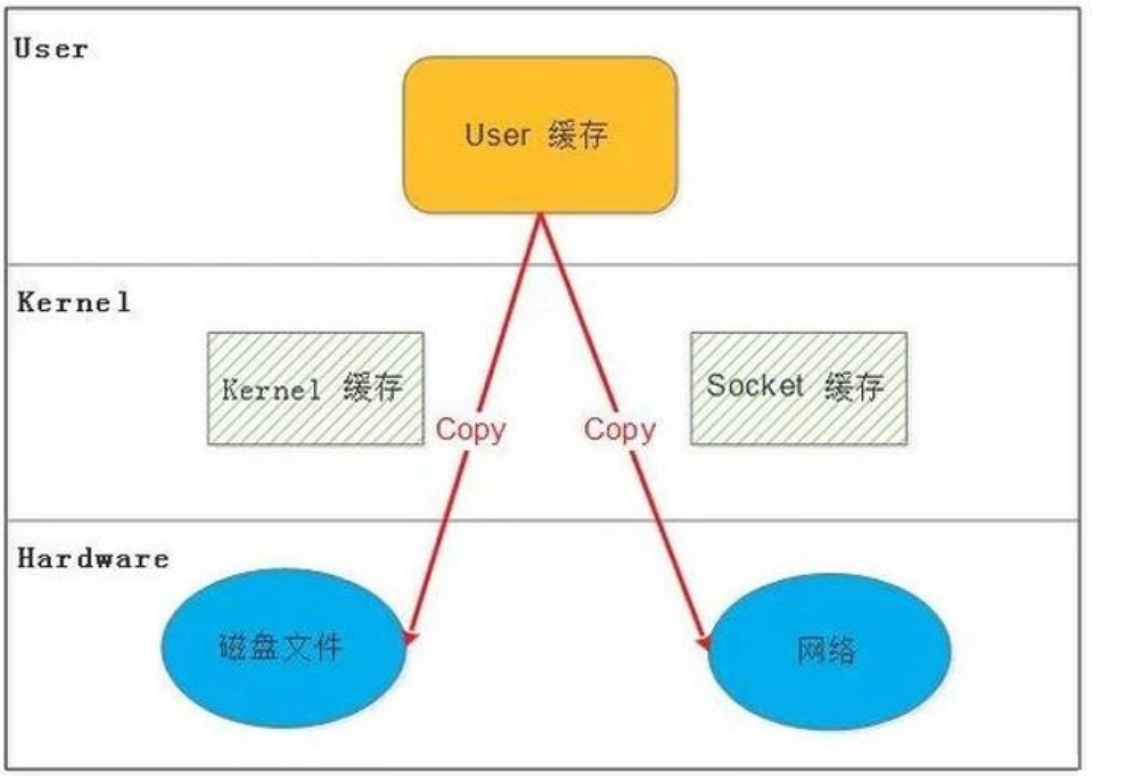
范例: select 和epoll 帮助

```
[root@centos8 ~]#whatis epoll
epoll (7)            - I/O event notification facility
[root@centos8 ~]#whatis select
select (2)           - synchronous I/O multiplexing
select (3)           - synchronous I/O multiplexing
select (3p)          - synchronous I/O multiplexing
[root@centos8 ~]#whatis poll
poll (2)             - wait for some event on a file descriptor
poll (3p)            - input/output multiplexing


[root@centos8 ~]#man 2  select
SELECT(2)                                                  Linux Programmer's
Manual                                                  SELECT(2)

NAME
       select, pselect, FD_CLR, FD_ISSET, FD_SET, FD_ZERO - synchronous I/O
multiplexing

[root@centos8 ~]#man 2  poll
POLL(2)                                                    Linux Programmer's
Manual                                                   POLL(2)

NAME
       poll, ppoll - wait for some event on a file descriptor
```

1.8 零拷贝

1.8.1 零拷贝介绍

1.8.1.1 传统 Linux中 I/O 的问题



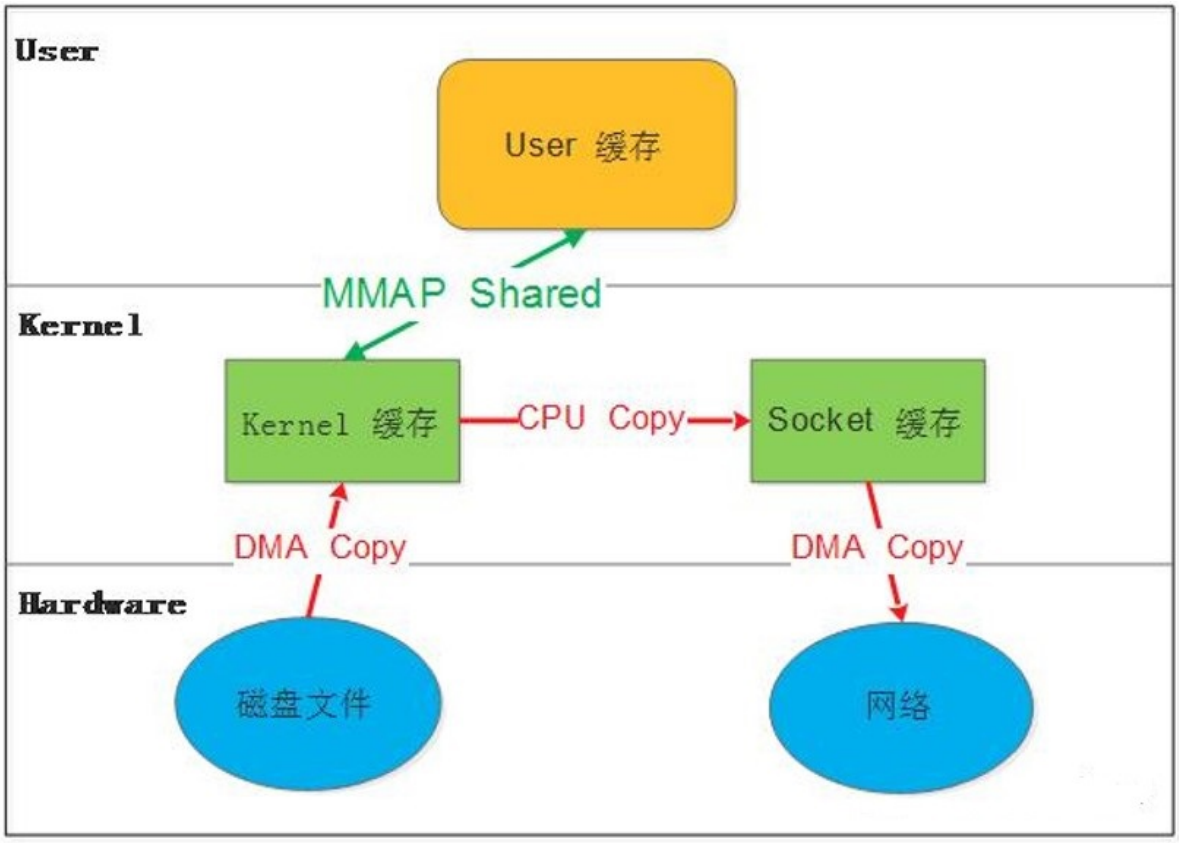
传统的 Linux 系统的标准 I/O 接口（read、write）是基于数据拷贝的，也就是数据都是 copy_to_user 或者 copy_from_user，这样做的好处是，通过中间缓存的机制，减少磁盘 I/O 的操作，但是坏处也很明显，大量数据的拷贝，用户态和内核态的频繁切换，会消耗大量的 CPU 资源，严重影响数据传输的性能，统计表明，在Linux协议栈中，数据包在内核态和用户态之间的拷贝所用的时间甚至占到了数据包整个处理流程时间的57.1%

1.8.1.2 什么是零拷贝

零拷贝就是上述问题的一个解决方案，通过尽量避免拷贝操作来缓解 CPU 的压力。零拷贝并没有真正做到“0”拷贝，它更多是一种思想，很多的零拷贝技术都是基于这个思想去做的优化

1.8.2 零拷页相关技术

1.8.2.1 MMAP (Memory Mapping)

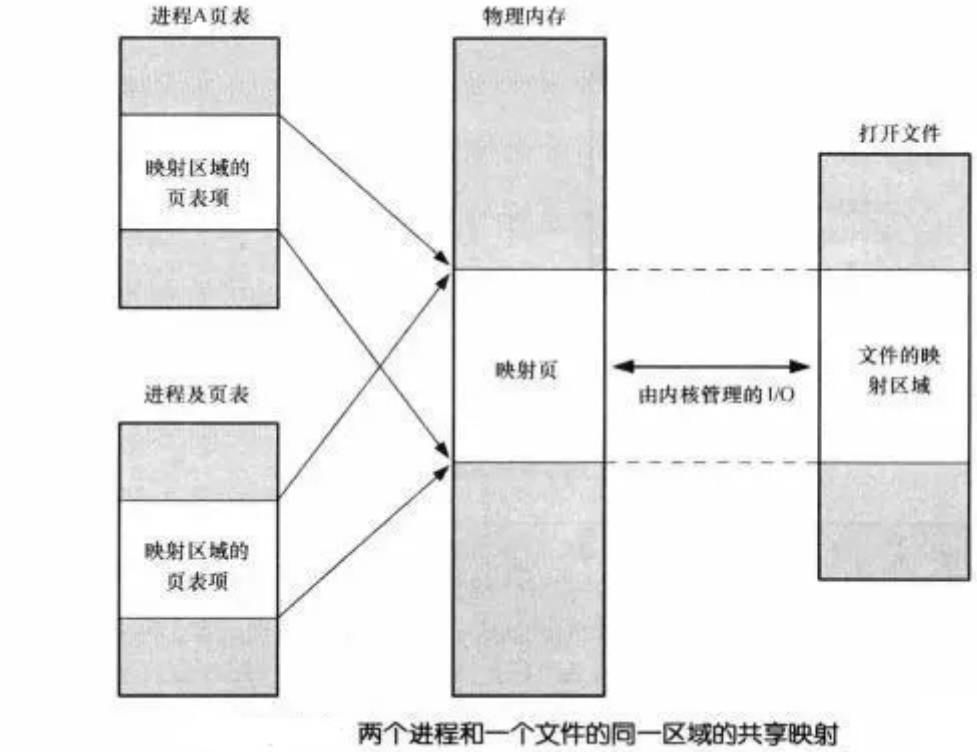


mmap()系统调用使得进程之间通过映射同一个普通文件实现共享内存。普通文件被映射到进程地址空间后，进程可以向访问普通内存一样对文件进行访问。

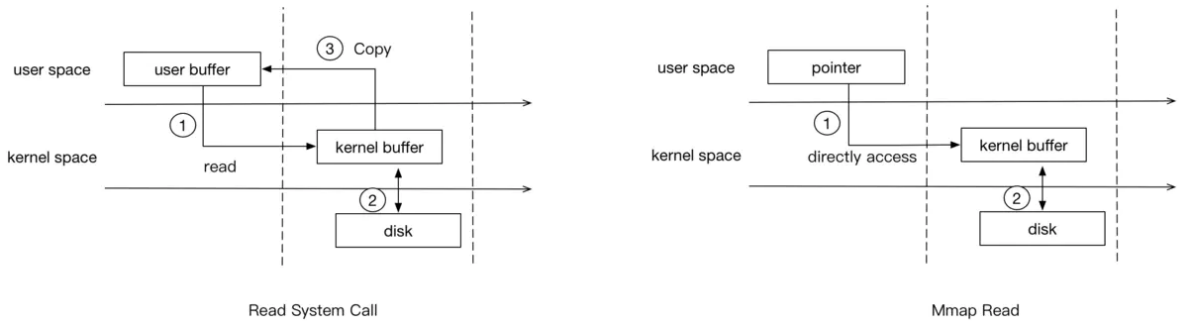
mmap是一种内存映射文件的方法，即将一个文件或者其它对象映射到进程的地址空间，实现文件磁盘地址和进程虚拟地址空间中一段虚拟地址的一一对映关系。

实现这样的映射关系后，进程就可以采用指针的方式读写操作这一段内存，而系统会自动回写脏页面到对应的文件磁盘上，即完成了对文件的操作而不必再调用read,write等系统调用函数。相反，内核空间对这段区域的修改也直接反映用户空间，从而可以实现不同进程间的文件共享。

内存映射减少数据在用户空间和内核空间之间的拷贝操作,适合大量数据传输

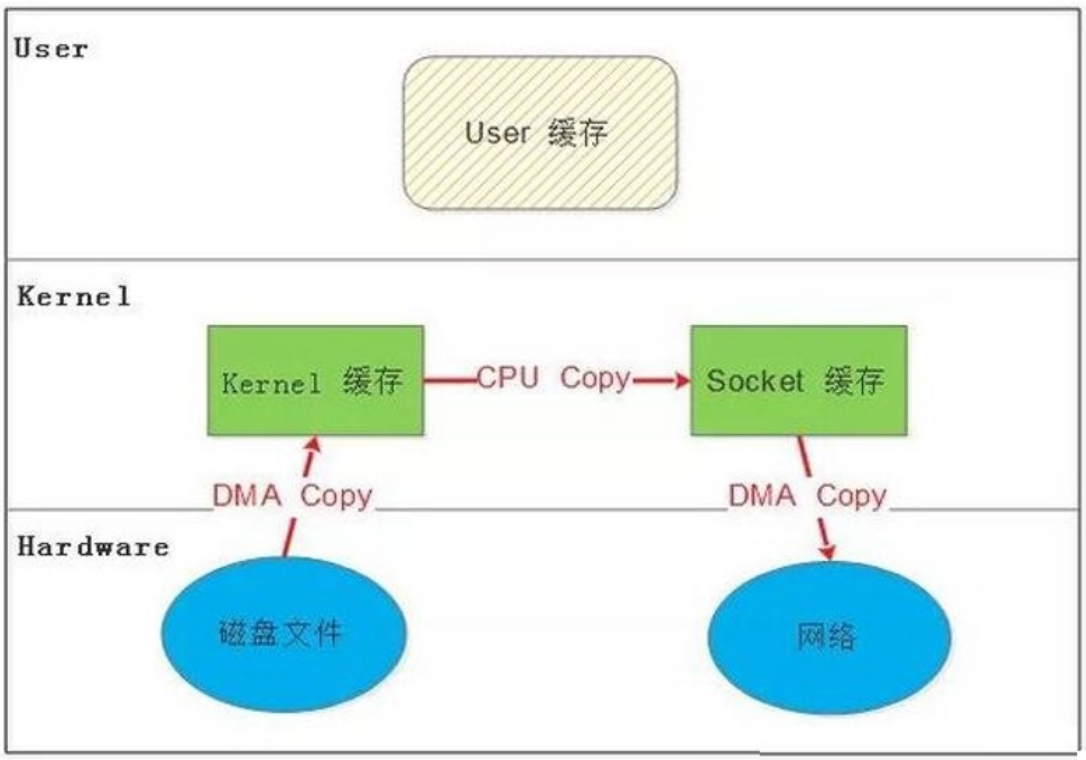


两个进程和一个文件的同一区域的共享映射

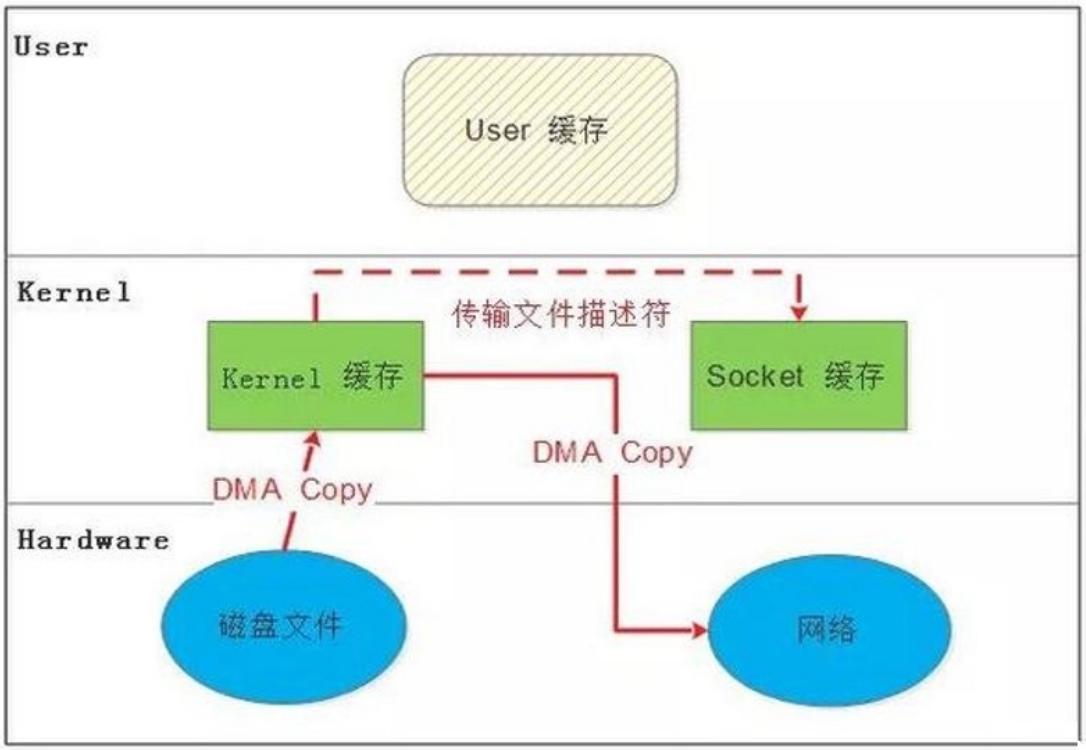


上面左图为传统读写,右图为MMAP.两者相比mmap要比普通的read系统调用少了一次copy的过程。因为read调用，进程是无法直接访问kernel space的，所以在read系统调用返回前，内核需要将数据从内核复制到进程指定的buffer。但mmap之后，进程可以直接访问mmap的数据(page cache)。

1.8.2.2 SENDFILE



1.8.2.3 DMA 辅助的 SENDFILE



2 Nginx 架构和安装

2.1 Nginx 概述

2.1.1 Nginx 介绍

Nginx：engine X，2002年开发，分为社区版和商业版(nginx plus)

2019年3月11日 F5 Networks 6.7亿美元的价格收购

Nginx是免费的、开源的、高性能的HTTP和反向代理服务器、邮件代理服务器、以及TCP/UDP代理服务器

解决C10K问题（10K Connections）

解决C1000K问题参考链接: http://www.ideawu.net/blog/archives/740.html

Nginx官网：http://nginx.org

nginx的其它的二次发行版：

- Tengine：由淘宝网发起的Web服务器项目。它在Nginx的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。Tengine的性能和稳定性已经在大型的网站如淘宝网，天猫商城等得到了很好的检验。它的最终目标是打造一个高效、稳定、安全、易用的Web平台。从2011年12月开始，Tengine成为一个开源项目，官网: http://tengine.taobao.org/
- OpenResty：基于 Nginx 与 Lua 语言的高性能 Web 平台， 章亦春团队开发，官网：http://openresty.org/cn/

2.1.2 Nginx 功能介绍

- 静态的web资源服务器html，图片，js，css，txt等静态资源
- http/https协议的反向代理
- 结合FastCGI/uWSGI/SCGI等协议反向代理动态资源请求
- tcp/udp协议的请求转发（反向代理）
- imap4/pop3协议的反向代理

2.2.3 基础特性

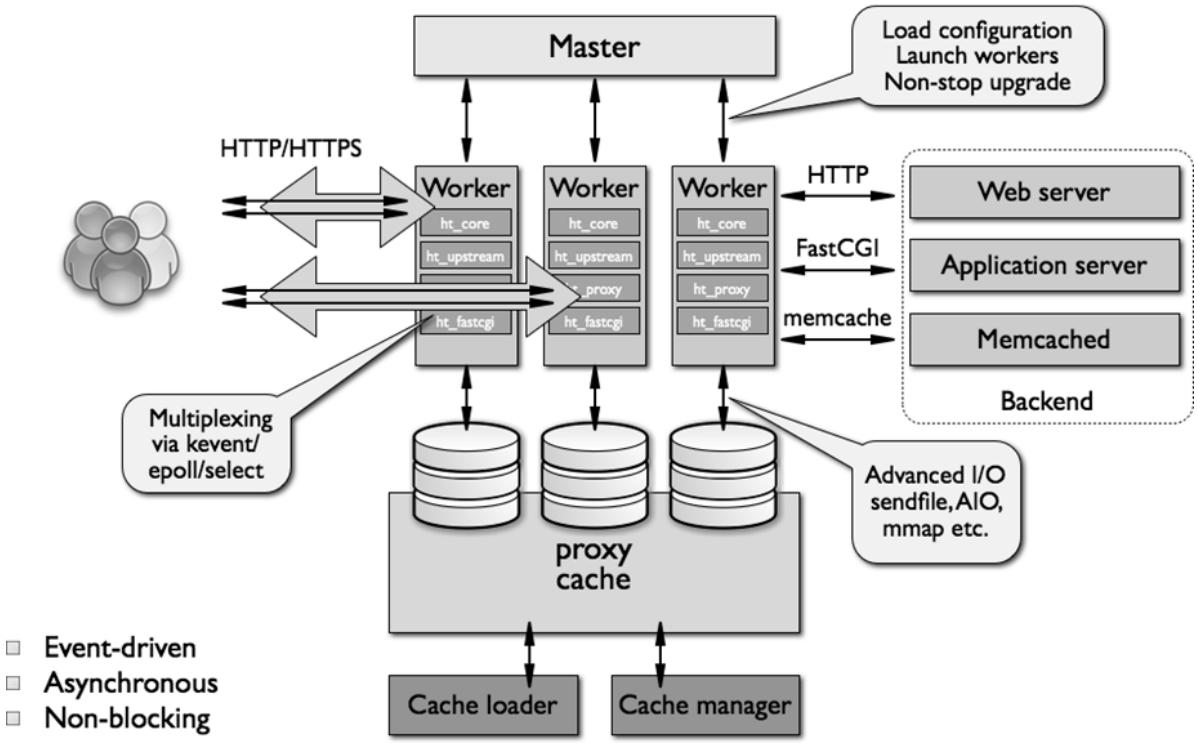
- 模块化设计，较好的扩展性
- 高可靠性
- 支持热部署：不停机更新配置文件，升级版本，更换日志文件
- 低内存消耗：10000个keep-alive连接模式下的非活动连接，仅需2.5M内存
- event-driven,aio,mmap，sendfile

2.2.4 Web 服务相关的功能

- 虚拟主机（server）
- 支持 keep-alive 和管道连接(利用一个连接做多次请求)
- 访问日志（支持基于日志缓冲提高其性能）
- url rewirte
- 路径别名
- 基于IP及用户的访问控制
- 支持速率限制及并发数限制
- 重新配置和在线升级而无须中断客户的工作进程

2.2 Nginx 架构和进程

2.2.1 Nginx 架构

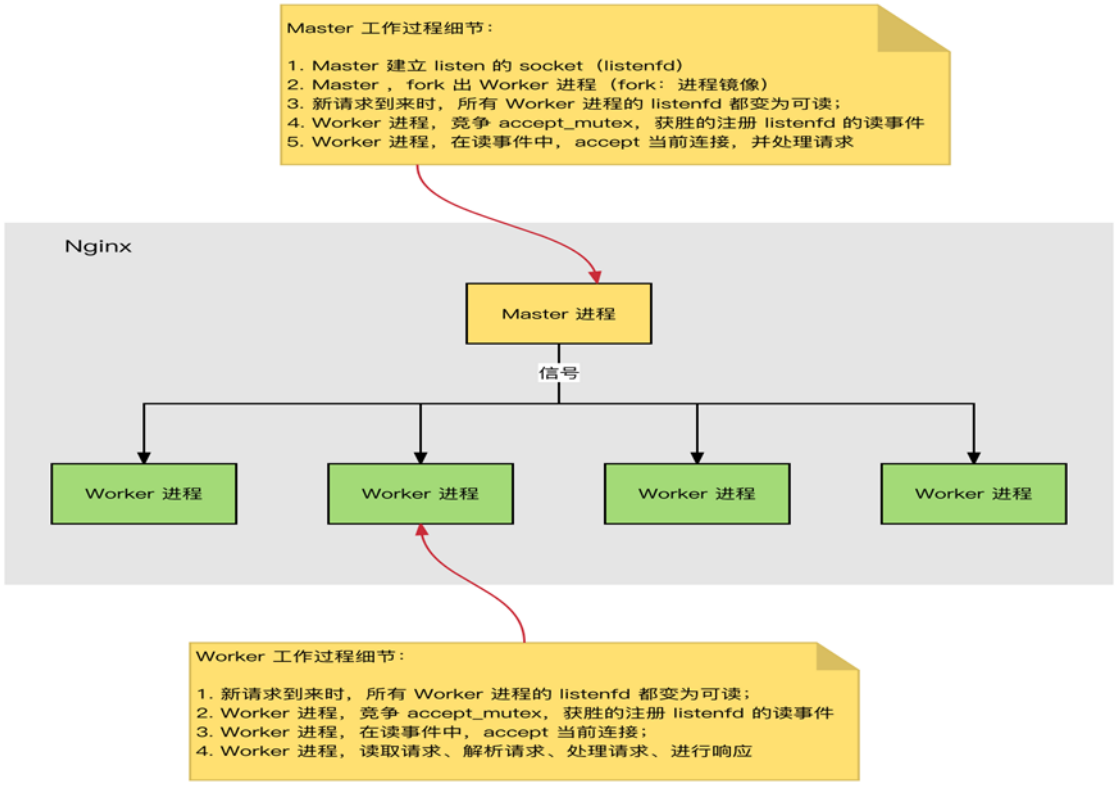


2.2.2 Nginx 进程结构

web请求处理机制

- 多进程方式：服务器每接收到一个客户端请求就有服务器的主进程生成一个子进程响应客户端，直到用户关闭连接，这样的优势是处理速度快，子进程之间相互独立，但是如果访问过大会导致服务器资源耗尽而无法提供请求。
- 多线程方式：与多进程方式类似，但是每收到一个客户端请求会有服务进程派生出一个线程和此客户端进行交互，一个线程的开销远远小于一个进程，因此多线程方式在很大程度减轻了web服务器对系统资源的要求，但是多线程也有自己的缺点，即当多个线程位于同一个进程内工作的时候，可以相互访问同样的内存地址空间，所以他们相互影响，一旦主进程挂掉则所有子线程都不能工作了，IIS服务器使用了多线程的方式，需要间隔一段时间就重启一次才能稳定。

Nginx是多进程组织模型，而且是一个由Master主进程和Worker工作进程组成。



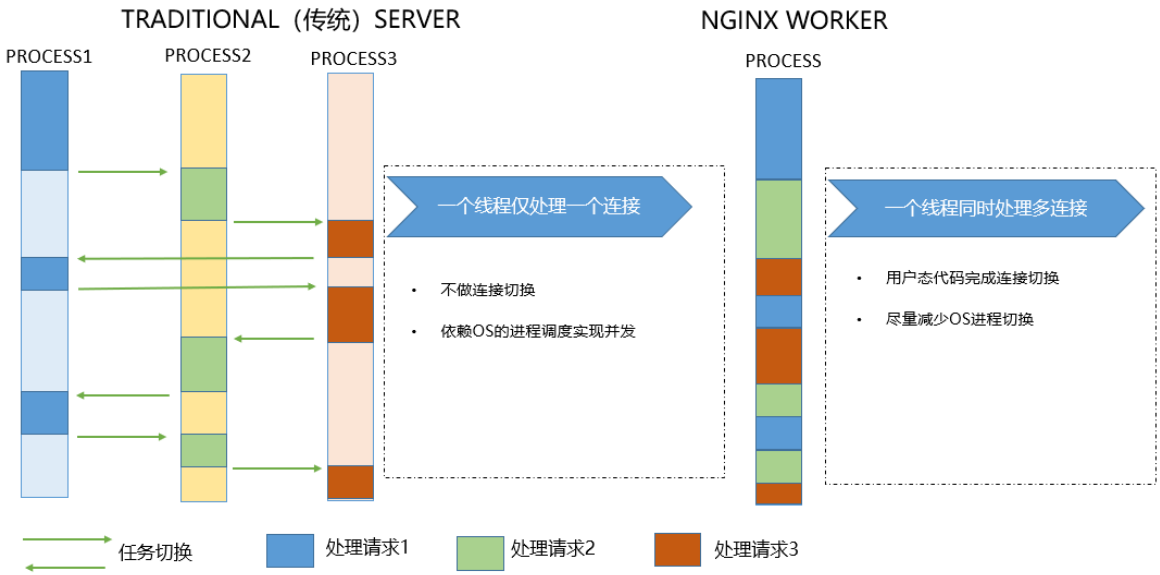
主进程(master process)的功能：

```
对外接口：接收外部的操作（信号）
对内转发：根据外部的操作的不同，通过信号管理 worker
监控：监控 worker 进程的运行状态，worker 进程异常终止后，自动重启 worker 进程
读取Nginx 配置文件并验证其有效性和正确性
建立、绑定和关闭socket连接
按照配置生成、管理和结束工作进程
接受外界指令，比如重启、升级及退出服务器等指令
不中断服务，实现平滑升级，重启服务并应用新的配置
开启日志文件，获取文件描述符
不中断服务，实现平滑升级，升级失败进行回滚处理
编译和处理perl脚本
```

工作进程（worker process）的功能：

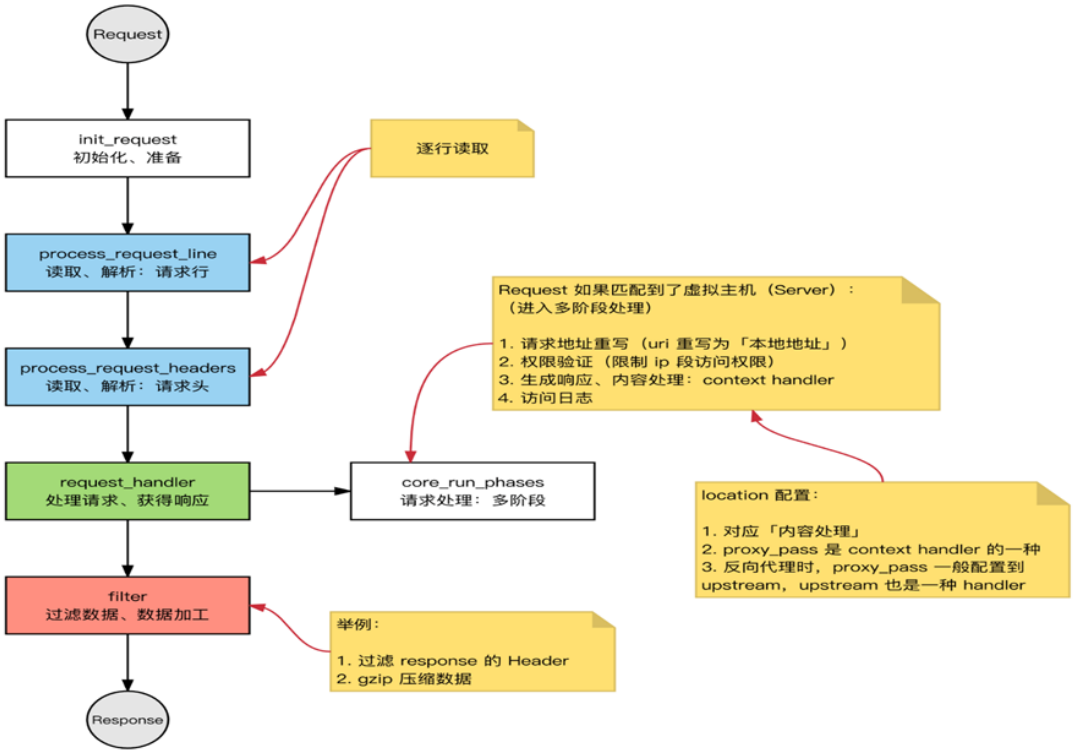
所有 worker 进程都是平等的
实际处理：网络请求，由 worker 进程处理
Worker进程数量：一般设置为核心数，充分利用CPU资源，同时避免进程数量过多，导致进程竞争CPU资源，增加上下文切换的损耗
接受处理客户的请求
将请求依次送入各个功能模块进行处理
I/O调用，获取响应数据
与后端服务器通信，接收后端服务器的处理结果
缓存数据，访问缓存索引，查询和调用缓存数据
发送请求结果，响应客户的请求
接收主程序指令，比如重启、升级和退出等

2.2.3 Nginx 启动和 HTTP 连接建立



- Nginx 启动时，Master 进程，加载配置文件
- Master 进程，初始化监听的 socket
- Master 进程，fork 出多个 Worker 进程
- Worker 进程，竞争新的连接，获胜方通过三次握手，建立 Socket 连接，并处理请求

2.2.4 HTTP 处理过程



2.3 Nginx 模块介绍

nginx 有多种模块

- 核心模块：是 Nginx 服务器正常运行必不可少的模块，提供错误日志记录 、配置文件解析 、事件驱动机制 、进程管理等核心功能
- 标准HTTP模块：提供 HTTP 协议解析相关的功能，比如： 端口配置 、 网页编码设置 、 HTTP响应头设置 等等
- 可选HTTP模块：主要用于扩展标准的 HTTP 功能，让 Nginx 能处理一些特殊的服务，比如： Flash 多媒体传输 、解析 GeoIP 请求、 网络传输压缩 、 安全协议 SSL 支持等
- 邮件服务模块：主要用于支持 Nginx 的 邮件服务 ， 包括对 POP3 协议、 IMAP 协议和 SMTP协议的支持
- Stream服务模块: 实现反向代理功能,包括TCP协议代理
- 第三方模块：是为了扩展 Nginx 服务器应用，完成开发者自定义功能，比如： Json 支持、 Lua 支持等

nginx高度模块化，但其模块早期不支持DSO机制;1.9.11 版本支持动态装载和卸载

模块分类：

```
核心模块：core module
标准模块：
    HTTP 模块：  ngx_http_*
            HTTP Core modules       #默认功能
            HTTP Optional modules   #需编译时指定
    Mail 模块：  ngx_mail_*
    Stream 模块 ngx_stream_*
第三方模块
```

Nginx 模块图

2.4 Nginx 安装

2.4.1 Nginx版本和安装方式

Nginx版本

- Mainline version 主要开发版本,一般为奇数版本号,比如1.19
- Stable version 当前最新稳定版,一般为偶数版本,如:1.20
- Legacy versions 旧的稳定版,一般为偶数版本,如:1.18

Nginx安装可以使用yum或源码安装，但是推荐使用源码编译安装

- yum的版本比较旧
- 编译安装可以更方便自定义相关路径
- 使用源码编译可以自定义相关功能，更方便业务的上的使用

2.4.2 基于 yum 安装 Nginx

2.4.2.1 查看当前系统中的 nginx 版本

范例: 查看系统和EPEL的nginx版本

```
[root@centos8 ~]#dnf info nginx
Last metadata expiration check: 0:53:10 ago on Tue 22 Sep 2020 11:01:33 AM CST.
Available Packages
Name         : nginx
Epoch        : 1
Version      : 1.14.1
Release      : 9.module_el8.0.0+184+e34fea82
Architecture : x86_64
Size         : 570 k
Source       : nginx-1.14.1-9.module_el8.0.0+184+e34fea82.src.rpm
```

```
Repository   : AppStream
Summary      : A high performance web server and reverse proxy server
URL          : http://nginx.org/
License      : BSD
Description  : Nginx is a web server and a reverse proxy server for HTTP, SMTP,
POP3 and
             : IMAP protocols, with a strong focus on high concurrency,
performance and low
             : memory usage.

#CentOS 7 需要提前配置好epel源
[root@centos7 ~]#yum info nginx
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base:
Available Packages
Name        : nginx
Arch        : x86_64
Epoch       : 1
Version     : 1.16.1
Release     : 1.el7
Size        : 562 k
Repo        : epel/7/x86_64
Summary     : A high performance web server and reverse proxy server
URL         : http://nginx.org/
License     : BSD
Description : Nginx is a web server and a reverse proxy server for HTTP, SMTP,
POP3 and
            : IMAP protocols, with a strong focus on high concurrency,
performance and low
            : memory usage.
```

2.4.2.2 官方包源安装最新版本 nginx

系统和EPEL源的中 nignx版本较旧,可以安装官方源的最新版本

官方包链接:

```
http://nginx.org/en/linux_packages.html
```

官方 yum 源链接

```
http://nginx.org/en/linux_packages.html#RHEL-CentOS
```

范例: 通过官方 yum 源安装nginx

```
[root@centos8 ~]#cat /etc/yum.repos.d/nginx.repo
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true

[nginx-mainline]
```

```
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true

#列出所有版本
[root@centos8 ~]#yum list  nginx --showduplicates
Last metadata expiration check: 0:01:08 ago on Tue 22 Sep 2020 12:01:59 PM CST.
Available Packages
nginx.x86_64    1.16.0-1.el8.ngx                                nginx-stable
nginx.x86_64    1:1.14.1-9.module_el8.0.0+184+e34fea82          AppStream
nginx.x86_64    1:1.16.1-1.el8.ngx                              nginx-stable
nginx.x86_64    1:1.18.0-1.el8.ngx                              nginx-stable

#查看版本信息
[root@centos8 ~]#dnf info  nginx
Last metadata expiration check: 0:02:50 ago on Tue 22 Sep 2020 12:01:59 PM CST.
Available Packages
Name         : nginx
Epoch        : 1
Version      : 1.18.0
Release      : 1.el8.ngx
Architecture : x86_64
Size         : 806 k
Source       : nginx-1.18.0-1.el8.ngx.src.rpm
Repository   : nginx-stable
Summary      : High performance web server
URL          : http://nginx.org/
License      : 2-clause BSD-like license
Description  : nginx [engine x] is an HTTP and reverse proxy server, as well as
             : a mail proxy server.

#安装指定版本
[root@centos8 ~]#yum -y install nginx-1.18.0
```

2.4.2.3 检查安装

查看nginx安装包信息

```
[root@centos8 ~]#rpm -q nginx
nginx-1.18.0-1.el8.ngx.x86_64

[root@centos8 ~]#rpm -qi  nginx
Name        : nginx
Epoch       : 1
Version     : 1.18.0
Release     : 1.el8.ngx
Architecture: x86_64
Install Date: Tue 22 Sep 2020 12:05:14 PM CST
Group       : System Environment/Daemons
Size        : 3815084
License     : 2-clause BSD-like license
Signature   : RSA/SHA1, Tue 21 Apr 2020 11:19:19 PM CST, Key ID abf5bd827bd9bf62
Source RPM  : nginx-1.18.0-1.el8.ngx.src.rpm
Build Date  : Tue 21 Apr 2020 11:07:57 PM CST
```

```
Build Host  : ip-10-1-17-47.eu-central-1.compute.internal
Relocations : (not relocatable)
Vendor      : Nginx, Inc.
URL         : http://nginx.org/
Summary     : High performance web server
Description :
nginx [engine x] is an HTTP and reverse proxy server, as well as
a mail proxy server.


[root@centos8 ~]#rpm -ql nginx
/etc/logrotate.d/nginx
/etc/nginx
/etc/nginx/conf.d
/etc/nginx/conf.d/default.conf
/etc/nginx/fastcgi_params
/etc/nginx/koi-utf
/etc/nginx/koi-win
/etc/nginx/mime.types
/etc/nginx/modules
/etc/nginx/nginx.conf
/etc/nginx/scgi_params
/etc/nginx/uwsgi_params
/etc/nginx/win-utf
/etc/sysconfig/nginx
/etc/sysconfig/nginx-debug
/usr/lib/.build-id
/usr/lib/.build-id/72
/usr/lib/.build-id/72/32b3d274d95e3739c1690a6344d3aac4c6272b
/usr/lib/.build-id/7f
/usr/lib/.build-id/7f/0259e101e7ab850dee7e6ac2907ac62f6f5917
/usr/lib/systemd/system/nginx-debug.service
/usr/lib/systemd/system/nginx.service
/usr/lib64/nginx
/usr/lib64/nginx/modules
/usr/libexec/initscripts/legacy-actions/nginx
/usr/libexec/initscripts/legacy-actions/nginx/check-reload
/usr/libexec/initscripts/legacy-actions/nginx/upgrade
/usr/sbin/nginx
/usr/sbin/nginx-debug
/usr/share/doc/nginx-1.18.0
/usr/share/doc/nginx-1.18.0/COPYRIGHT
/usr/share/man/man8/nginx.8.gz
/usr/share/nginx
/usr/share/nginx/html
/usr/share/nginx/html/50x.html
/usr/share/nginx/html/index.html
/var/cache/nginx
/var/log/nginx

#带有自动日志切割功能
[root@centos8 ~]#cat /etc/logrotate.d/nginx
/var/log/nginx/*log {
    create 0664 nginx root
    daily
    rotate 10
    missingok
    notifempty
```

```
    compress
    sharedscripts
    postrotate
        /bin/kill -USR1 `cat /run/nginx.pid 2>/dev/null` 2>/dev/null || true
    endscript
}
```

2.4.2.4 nginx 程序用法帮助

使用安装完成的二进制文件nginx

```
[root@centos8 ~]#nginx -h
nginx version: nginx/1.18.0
Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit  #显示版本和编译参数
  -t            : test configuration and exit                   #测试配置文件是否异
常
  -T            : test configuration, dump it and exit          #测试并打印
  -q            : suppress non-error messages during configuration testing #静默模
式
  -s signal     : send signal to a master process: stop, quit, reopen, reload #发
送信号,reload信号 会生成新的worker,但master不会重新生成
  -p prefix     : set prefix path (default: /etc/nginx/)
#指定Nginx 目录
  -c filename   : set configuration file (default: /etc/nginx/nginx.conf)
#配置文件路径
  -g directives : set global directives out of configuration file#设置全局指令,注意
和配置文件不要同时配置,否则冲突
```

范例: nginx 命令使用

```
[root@centos8 ~]#nginx  -g "worker_processes 6;"
nginx: [emerg] "worker_processes" directive is duplicate in
/apps/nginx/conf/nginx.conf:3

[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf

#worker_processes  1;

[root@centos8 ~]#nginx  -g "worker_processes 6;"
[root@centos8 ~]#ps aux|grep nginx
root        8843  0.0  0.0  41048   844 ?        Ss   18:53   0:00 nginx: master
process nginx -g worker_processes 6;
nginx       8844  0.0  0.4  74572  4832 ?        S    18:53   0:00 nginx: worker
process
nginx       8845  0.0  0.4  74572  4832 ?        S    18:53   0:00 nginx: worker
process
nginx       8846  0.0  0.4  74572  4832 ?        S    18:53   0:00 nginx: worker
process
nginx       8847  0.0  0.4  74572  4832 ?        S    18:53   0:00 nginx: worker
process
```

```
nginx       8848  0.0  0.4  74572  4832 ?        S    18:53   0:00 nginx: worker
process
nginx       8849  0.0  0.4  74572  4832 ?        S    18:53   0:00 nginx: worker
process
root        8851  0.0  0.1  12108  1076 pts/1    S+   18:53   0:00 grep --
color=auto nginx

[root@centos8 ~]#nginx  -s quit
[root@centos8 ~]#ps aux|grep nginx
root        8858  0.0  0.1  12108  1100 pts/1    S+   18:54   0:00 grep --
color=auto nginx

#前台运行
[root@centos8 ~]#nginx  -g 'daemon off;'
^C[root@centos8 ~]#
```

2.4.2.5 验证 Nginx

```
[root@centos8 ~]#nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

[root@centos8 ~]#nginx -V
nginx version: nginx/1.18.0
built by gcc 8.3.1 20190507 (Red Hat 8.3.1-4) (GCC)
built with OpenSSL 1.1.1c FIPS  28 May 2019
TLS SNI support enabled
configure arguments: --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --modules-
path=/usr/lib64/nginx/modules --conf-path=/etc/nginx/nginx.conf --error-log-
path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-
path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --http-client-body-temp-
path=/var/cache/nginx/client_temp --http-proxy-temp-
path=/var/cache/nginx/proxy_temp --http-fastcgi-temp-
path=/var/cache/nginx/fastcgi_temp --http-uwsgi-temp-
path=/var/cache/nginx/uwsgi_temp --http-scgi-temp-
path=/var/cache/nginx/scgi_temp --user=nginx --group=nginx --with-compat --with-
file-aio --with-threads --with-http_addition_module --with-
http_auth_request_module --with-http_dav_module --with-http_flv_module --with-
http_gunzip_module --with-http_gzip_static_module --with-http_mp4_module --with-
http_random_index_module --with-http_realip_module --with-
http_secure_link_module --with-http_slice_module --with-http_ssl_module --with-
http_stub_status_module --with-http_sub_module --with-http_v2_module --with-mail
--with-mail_ssl_module --with-stream --with-stream_realip_module --with-
stream_ssl_module --with-stream_ssl_preread_module --with-cc-opt='-O2 -g -pipe -
Wall -Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -
fexceptions -fstack-protector-strong -grecord-gcc-switches -
specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -specs=/usr/lib/rpm/redhat/redhat-
annobin-cc1 -m64 -mtune=generic -fasynchronous-unwind-tables -fstack-clash-
protection -fcf-protection -fPIC' --with-ld-opt='-Wl,-z,relro -Wl,-z,now -pie'
```

2.4.2.6 Nginx 启动文件

```
[root@centos8 ~]# cat /usr/lib/systemd/system/nginx.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
LimitNOFILE=100000

[Install]
WantedBy=multi-user.target
```

2.4.2.7 Nginx 配置文件

范例: 查看配置文件列表

```
[root@centos8 ~]#rpm -qc nginx
/etc/logrotate.d/nginx
/etc/nginx/conf.d/default.conf
/etc/nginx/fastcgi_params
/etc/nginx/koi-utf
/etc/nginx/koi-win
/etc/nginx/mime.types
/etc/nginx/nginx.conf
/etc/nginx/scgi_params
/etc/nginx/uwsgi_params
/etc/nginx/win-utf
/etc/sysconfig/nginx
/etc/sysconfig/nginx-debug


[root@centos8 ~]#cat /etc/sysconfig/nginx
# Configuration file for the nginx service.

NGINX=/usr/sbin/nginx
CONFFILE=/etc/nginx/nginx.conf

[root@centos8 ~]#tree /etc/nginx/
/etc/nginx/
├── conf.d
│   └── default.conf
├── fastcgi_params
├── koi-utf
├── koi-win
├── mime.types
├── modules -> ../../usr/lib64/nginx/modules
├── nginx.conf
├── scgi_params
```

```
├── uwsgi_params
└── win-utf

2 directories, 9 files
```

范例: 配置文件/etc/nginx/nginx.conf 默认配置

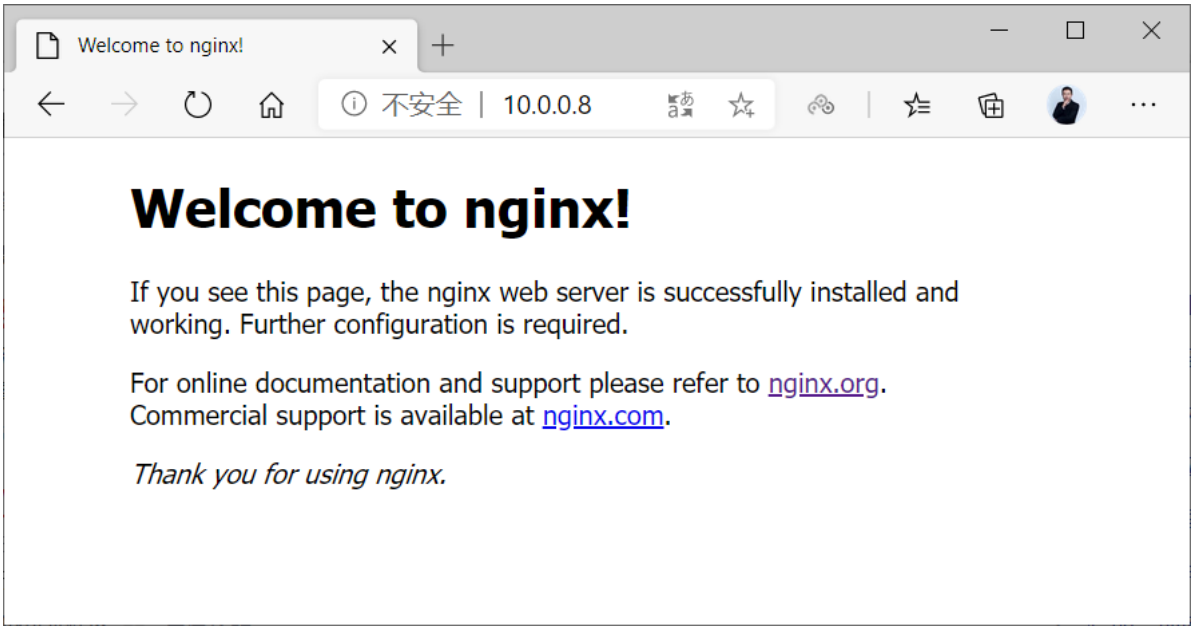
```
[root@centos8 ~]#grep -Ev "^ *#|^$" /etc/nginx/nginx.conf
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
    sendfile        on;
    keepalive_timeout  65;
    include /etc/nginx/conf.d/*.conf;
}
```

2.4.2.8 启动 Nginx

```
[root@centos8 ~]#systemctl enable --now  nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service →
/usr/lib/systemd/system/nginx.service.
[root@centos8 ~]#systemctl status  nginx
● nginx.service - nginx - high performance web server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset:
disabled)
   Active: active (running) since Tue 2020-09-22 12:24:19 CST; 4s ago
     Docs: http://nginx.org/en/docs/
  Process: 24584 ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf
(code=exited, status=0/SUCCESS)
 Main PID: 24585 (nginx)
    Tasks: 2 (limit: 5882)
   Memory: 2.4M
   CGroup: /system.slice/nginx.service
           ├─24585 nginx: master process /usr/sbin/nginx -c
/etc/nginx/nginx.conf
           └─24586 nginx: worker process

Sep 22 12:24:19 centos8.wangxiaochun.com systemd[1]: Starting nginx - high
performance web server...
Sep 22 12:24:19 centos8.wangxiaochun.com systemd[1]: Started nginx - high
performance web server.
[root@centos8 ~]#ps aux|grep nginx
root       24585  0.0  0.0  41312   860 ?        Ss   12:24   0:00 nginx: master
process /usr/sbin/nginx -c /etc/nginx/nginx.conf
```

```
nginx      24586  0.0  0.5  74896  4884 ?        S    12:24   0:00 nginx: worker
process
root       24590  0.0  0.1  12108  1088 pts/0    S+   12:24   0:00 grep --
color=auto nginx
[root@centos8 ~]#pstree -p |grep nginx
           |-nginx(24585)---nginx(24586)
```

2.4.2.9 访问 Nginx



2.4.3 Nginx 编译安装

编译器介绍

源码安装需要提前准备标准的编译器，GCC的全称是（GNU Compiler collection），其有GNU开发，并以GPL即LGPL许可，是自由的类UNIX即苹果电脑Mac OS X操作系统的标准编译器，因为GCC原本只能处理C语言，所以原名为GNU C语言编译器，后来得到快速发展，可以处理C++,Fortran，pascal，objective-C，java以及Ada等其他语言，此外还需要Automake工具，以完成自动创建Makefile的工作，Nginx的一些模块需要依赖第三方库，比如：pcre（支持rewrite），zlib（支持gzip模块）和openssl（支持ssl模块）等。

2.4.3.1 编译安装 Nginx

官方源码包下载地址：

```
https://nginx.org/en/download.html
```

范例: 编译安装

```
[root@centos8 ~]#yum -y install gcc pcre-devel openssl-devel zlib-devel
[root@centos8 ~]#useradd  -s  /sbin/nologin nginx

[root@centos8 ~]#cd /usr/local/src/
[root@centos8 src]#wget http://nginx.org/download/nginx-1.18.0.tar.gz
[root@centos8 src]#tar xf nginx-1.18.0.tar.gz
[root@centos8 src]#cd nginx-1.18.0/

[root@centos8 nginx-1.18.0]#./configure --prefix=/apps/nginx \
--user=nginx  \
```

```
--group=nginx \
--with-http_ssl_module \
--with-http_v2_module \
--with-http_realip_module \
--with-http_stub_status_module  \
--with-http_gzip_static_module \
--with-pcre \
--with-stream \
--with-stream_ssl_module \
--with-stream_realip_module

[root@centos8 nginx-1.18.0]#make && make install

#修改权限
[root@centos8 nginx-1.18.0]#chown -R nginx.nginx /apps/nginx
```

nginx完成安装以后，有四个主要的目录

```
#生成目录
[root@centos8 nginx-1.18.0]#ll /apps/nginx/
total 0
drwxr-xr-x 2 root root 333 Sep 22 12:49 conf
drwxr-xr-x 2 root root  40 Sep 22 12:49 html
drwxr-xr-x 2 root root   6 Sep 22 12:49 logs
drwxr-xr-x 2 root root  19 Sep 22 12:49 sbin
```

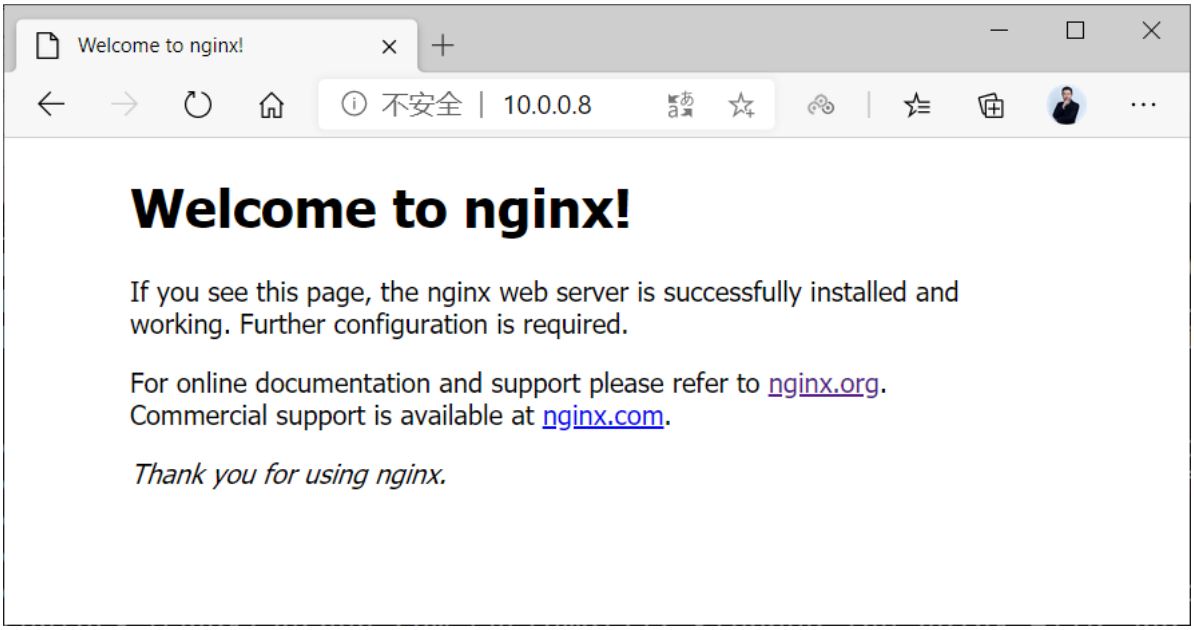
conf：保存nginx所有的配置文件，其中nginx.conf是nginx服务器的最核心最主要的配置文件，其他的.conf则是用来配置nginx相关的功能的，例如fastcgi功能使用的是fastcgi.conf和fastcgi_params两个文件，配置文件一般都有一个样板配置文件，是以.default为后缀，使用时可将其复制并将default后缀去掉即可。
html目录中保存了nginx服务器的web文件，但是可以更改为其他目录保存web文件,另外还有一个50x的web文件是默认的错误页面提示页面。
logs：用来保存nginx服务器的访问日志错误日志等日志，logs目录可以放在其他路径，比如/var/logs/nginx里面。
sbin：保存nginx二进制启动脚本，可以接受不同的参数以实现不同的功能。

2.4.3.2 验证版本及编译参数

```
[root@centos8 nginx-1.18.0]#ls /apps/nginx/sbin/
nginx
[root@centos8 nginx-1.18.0]#ln -s /apps/nginx/sbin/nginx /usr/sbin/

#查看版本
[root@centos8 ~]#nginx -v
nginx version: nginx/1.18.0

#查看编译参数
[root@centos8 ~]#nginx -V
nginx version: nginx/1.18.0
built by gcc 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)
built with OpenSSL 1.1.1c FIPS  28 May 2019
TLS SNI support enabled
configure arguments: --prefix=/apps/nginx --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream
--with-stream_ssl_module --with-stream_realip_module
```

2.4.3.3 启动和停止 nginx 测试访问 web 界面

```
#启动nginx
[root@centos8 ~]#nginx

#浏览器可以访问看到下面图示

[root@centos8 ~]#ss -ntl
State             Recv-Q            Send-Q       Local Address:Port    Peer
Address:Port
LISTEN            0                 100              127.0.0.1:25
 0.0.0.0:*
LISTEN            0                 128                0.0.0.0:80
 0.0.0.0:*
LISTEN            0                 128                0.0.0.0:22
 0.0.0.0:*
LISTEN            0                 100                  [::1]:25
[::]:*
LISTEN            0                 128                  [::]:22
[::]:*


#关闭nginx
[root@centos8 ~]#nginx  -s stop
[root@centos8 ~]#ss -ntl
State             Recv-Q            Send-Q     Local Address:Port    Peer
Address:Port
LISTEN            0                 100            127.0.0.1:25
 0.0.0.0:*
LISTEN            0                 128              0.0.0.0:22
 0.0.0.0:*
LISTEN            0                 100                [::1]:25
[::]:*
LISTEN            0                 128                [::]:22
[::]:*
```



2.4.3.4 创建 Nginx 自启动文件

```
#复制同一版本的nginx的yum安装生成的service文件
[root@centos8 ~]#vim  /usr/lib/systemd/system/nginx.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/apps/nginx/run/nginx.pid #指定pid文件的目录,默认在logs目录下,可选配置
ExecStart=/apps/nginx/sbin/nginx -c /apps/nginx/conf/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
LimitNOFILE=100000

[Install]
WantedBy=multi-user.target

#创建pid文件存放的目录
[root@centos8 ~]#mkdir /apps/nginx/run/

#修改配置文件
[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf
pid   /apps/nginx/run/nginx.pid;
```

2.4.3.5 验证 Nginx 自启动文件

```
[root@centos8 ~]#systemctl daemon-reload
[root@centos8 ~]#systemctl enable --now nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service →


[root@centos8 ~]#ll /apps/nginx/run/
total 4
-rw-r--r-- 1 root root 5 Sep 22 13:01 nginx.pid

[root@centos8 ~]#ss -ntl
State             Recv-Q            Send-Q  Local Address:Port   Peer
Address:Port
LISTEN            0                 100         127.0.0.1:25          0.0.0.0:*

LISTEN            0                 128           0.0.0.0:80          0.0.0.0:*

LISTEN            0                 128           0.0.0.0:22          0.0.0.0:*

LISTEN            0                 100             [::1]:25             [::]:*

LISTEN            0                 128              [::]:22             [::]:*


[root@centos8 ~]#systemctl stop nginx
[root@centos8 ~]#systemctl status  nginx
● nginx.service - The nginx HTTP and reverse proxy server
```

```
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset:
disabled)
   Active: inactive (dead) since Tue 2020-09-22 13:01:49 CST; 1s ago
  Process: 8113 ExecStart=/apps/nginx/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 8112 ExecStartPre=/apps/nginx/sbin/nginx -t (code=exited,
status=0/SUCCESS)
  Process: 8110 ExecStartPre=/usr/bin/rm -f /apps/nginx/logs/nginx.pid
(code=exited, status=0/SUCCESS)
 Main PID: 8115 (code=exited, status=0/SUCCESS)

Sep 22 12:58:53 centos8.wangxiaochun.com systemd[1]: Starting The nginx HTTP and
reverse proxy server...
Sep 22 12:58:53 centos8.wangxiaochun.com nginx[8112]: nginx: the configuration
file /apps/nginx/conf/nginx.conf syntax is ok
Sep 22 12:58:53 centos8.wangxiaochun.com nginx[8112]: nginx: configuration file
/apps/nginx/conf/nginx.conf test is successful
Sep 22 12:58:53 centos8.wangxiaochun.com systemd[1]: Started The nginx HTTP and
reverse proxy server.
Sep 22 13:01:49 centos8.wangxiaochun.com systemd[1]: Stopping The nginx HTTP and
reverse proxy server...
Sep 22 13:01:49 centos8.wangxiaochun.com systemd[1]: Stopped The nginx HTTP and
reverse proxy server.

[root@centos8 ~]#ss -ntl
State             Recv-Q            Send-Q   Local Address:Port    Peer
Address:Port
LISTEN            0                 100         127.0.0.1:25           0.0.0.0:*

LISTEN            0                 128           0.0.0.0:22           0.0.0.0:*

LISTEN            0                 100             [::1]:25              [::]:*

LISTEN            0                 128              [::]:22              [::]:*
```

2.4.4 自动化部署 nginx

2.4.4.1 实战案例: 一键安装 nginx 脚本

```
#!/bin/bash
#
#**************************************************************************
#Author:            wangxiaochun
#QQ:                29308620
#Date:              2020-12-01
#FileName:          install_nginx.sh
#URL:               http://www.wangxiaochun.com
#Description:       The test script
#Copyright (C):     2021 All rights reserved
#**************************************************************************
SRC_DIR=/usr/local/src
NGINX_URL=http://nginx.org/download/
NGINX_FILE=nginx-1.18.0
TAR=.tar.gz
NGINX_INSTALL_DIR=/apps/nginx
CPUS=`lscpu |awk '/^CPU\(s\)/{print $2}'`
```

```
color () {
    RES_COL=60
    MOVE_TO_COL="echo -en \\033[${RES_COL}G"
    SETCOLOR_SUCCESS="echo -en \\033[1;32m"
    SETCOLOR_FAILURE="echo -en \\033[1;31m"
    SETCOLOR_WARNING="echo -en \\033[1;33m"
    SETCOLOR_NORMAL="echo -en \E[0m"
    echo -n "$1" && $MOVE_TO_COL
    echo -n "["
    if [ $2 = "success" -o $2 = "0" ] ;then
        ${SETCOLOR_SUCCESS}
        echo -n $"  OK  "
    elif [ $2 = "failure" -o $2 = "1"  ] ;then
        ${SETCOLOR_FAILURE}
        echo -n $"FAILED"
    else
        ${SETCOLOR_WARNING}
        echo -n $"WARNING"
    fi
    ${SETCOLOR_NORMAL}
    echo -n "]"
    echo
}

os_type () {
   awk -F'[ "]' '/^NAME/{print $2}' /etc/os-release
}

os_version () {
   awk -F'"' '/^VERSION_ID/{print $2}' /etc/os-release
}

check () {
    [ -e ${NGINX_INSTALL_DIR} ] && { color "nginx 已安装,请卸载后再安装" 1; exit; }
    cd  ${SRC_DIR}
    if [  -e ${NGINX_FILE}${TAR} ];then
        color "相关文件已准备好" 0
    else
        color '开始下载 nginx 源码包' 0
        wget ${NGINX_URL}${NGINX_FILE}${TAR}
        [ $? -ne 0 ] && { color "下载 ${NGINX_FILE}${TAR}文件失败" 1; exit; }
    fi
}

install () {
    color "开始安装 nginx" 0
    if id nginx  &> /dev/null;then
        color "nginx 用户已存在" 1
    else
        useradd -s /sbin/nologin -r  nginx
        color "创建 nginx 用户" 0
    fi
    color "开始安装 nginx 依赖包" 0
    if [ `os_type` == "CentOS" -a `os_version` == '8' ] ;then
        yum -y -q install make gcc-c++ libtool pcre pcre-devel zlib zlib-devel
openssl openssl-devel perl-ExtUtils-Embed
    elif [ `os_type` == "CentOS" -a `os_version` == '7' ];then
```

```
        yum -y -q  install make gcc pcre-devel openssl-devel zlib-devel perl-
ExtUtils-Embed
    else
        apt update &> /dev/null
        apt -y install make gcc libpcre3 libpcre3-dev openssl libssl-dev zlib1g-
dev &> /dev/null
    fi
    cd $SRC_DIR
    tar xf ${NGINX_FILE}${TAR}
    NGINX_DIR=`echo ${NGINX_FILE}${TAR}| sed -nr 's/^(.*[0-9]).*/\1/p'`
    cd ${NGINX_DIR}
    ./configure --prefix=${NGINX_INSTALL_DIR} --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream
--with-stream_ssl_module --with-stream_realip_module
    make -j $CPUS && make install
    [ $? -eq 0 ] && color "nginx 编译安装成功" 0 ||  { color "nginx 编译安装失败,退
出!" 1 ;exit; }
    echo "PATH=${NGINX_INSTALL_DIR}/sbin:${PATH}" > /etc/profile.d/nginx.sh
    cat > /lib/systemd/system/nginx.service <<EOF
[Unit]
Description=The nginx HTTP and reverse proxy server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=${NGINX_INSTALL_DIR}/logs/nginx.pid
ExecStartPre=/bin/rm -f ${NGINX_INSTALL_DIR}/logs/nginx.pid
ExecStartPre=${NGINX_INSTALL_DIR}/sbin/nginx -t
ExecStart=${NGINX_INSTALL_DIR}/sbin/nginx
ExecReload=/bin/kill -s HUP \$MAINPID
KillSignal=SIGQUIT
LimitNOFILE=100000
TimeoutStopSec=5
KillMode=process
PrivateTmp=true

[Install]
WantedBy=multi-user.target
EOF
    systemctl daemon-reload
    systemctl enable --now nginx &> /dev/null
    systemctl is-active nginx &> /dev/null ||  { color "nginx 启动失败,退出!" 1 ;
exit; }
    color "nginx 安装完成" 0
}

check
install
```

2.4.4.2 利用 ansible 的playbook 实现 nginx 编译安装

```
[root@centos8 ~]#cat install_nginx.yml
---
- hosts: webservers
  remote_user: root
  gather_facts: yes
```

```
vars:
  nginx_verison: nginx-1.18.0
  suffix: .tar.gz
  down_dir: /usr/local/src/
  ins_dir: /apps/nginx/
  user: nginx
  group: nginx

tasks:
  - name: install packages on centos8
    yum:
      name:
        - gcc
        - make
        - pcre-devel
        - openssl-devel
        - zlib-devel
      state: present
    when:
      - ansible_facts['distribution'] == "CentOS"
      - ansible_facts['distribution_major_version'] == "8"
  - name: install packages on centos7
    yum:
      name:
        - gcc
        - make
        - pcre-devel
        - openssl-devel
        - zlib-devel
        - perl-ExtUtils-Embed
      state: present
    when:
      - ansible_facts['distribution'] == "CentOS"
      - ansible_facts['distribution_major_version'] == "7"

  - name: install packages on ubuntu18.04
    apt:
      name:
        - gcc
        - make
        - libpcre3
        - libpcre3-dev
        - openssl
        - libssl-dev
        - zlib1g-dev
      state: present
    when:
      - ansible_facts['distribution'] == "Ubuntu"
      - ansible_facts['distribution_major_version'] == "18"
  - name: group "{{ group }}"
    group: name={{ group }} state=present system=yes
  - name: user "{{ user }}"
    user:
      name: "{{ user }}"
      shell: /sbin/nologin
      system: yes
      group: "{{ group }}"
      home: "/home/{{ user }}"
```

```
        create_home: no
    - name: unarchive
      unarchive:
        src: "http://nginx.org/download/{{ nginx_verison }}{{ suffix }}"
        dest: "{{ down_dir }}"
        remote_src: yes
    - name: configure
      shell: ./configure \
                --prefix={{ ins_dir }} \
                --user={{ user }} --group={{ group }} \
                --with-http_ssl_module \
                --with-http_realip_module \
                --with-http_v2_module \
                --with-http_stub_status_module \
                --with-http_gzip_static_module \
                --with-pcre \
                --with-stream \
                --with-stream_ssl_module \
                --with-stream_realip_module
      args:
        chdir: "{{ down_dir }}{{ nginx_verison }}/"
    - name: make
      shell: make -j "{{ ansible_processor_vcpus }}" && make install
      args:
        chdir: "{{ down_dir }}{{ nginx_verison }}/"
    - name: link
      file:
        src: "{{ ins_dir }}sbin/nginx"
        dest: /usr/sbin/nginx
        state: link
    - name: service file
      copy: content='[Unit]\nDescription=nginx - high performance web
server\nDocumentation=http://nginx.org/en/docs/\nAfter=network-online.target
remote-fs.target nss-lookup.target\nWants=network-
online.target\n\n[Service]\nType=forking\nPIDFile={{ ins_dir
}}run/nginx.pid\nExecStart=/usr/sbin/nginx -c {{ ins_dir
}}conf//nginx.conf\nExecReload=/bin/sh -c "/bin/kill -s HUP $(/bin/cat {{ ins_dir
}}run/nginx.pid)"\nExecStop=/bin/sh -c "/bin/kill -s TERM $(/bin/cat {{ ins_dir
}}run/nginx.pid)"\nLimitNOFILE=100000\n[Install]\nWantedBy=multi-user.target'
dest=/lib/systemd/system/nginx.service
    - name: change configure pid location and change worker_processes
      lineinfile:
        path: "{{ item.path }}"
        regexp: "{{ item.regexp }}"
        line: "{{ item.line }}"
      with_items:
        - { path: "{{ ins_dir }}conf/nginx.conf",regexp:
'^worker_processes',line: "worker_processes  {{ ansible_processor_vcpus }};" }
        - { path: "{{ ins_dir }}conf/nginx.conf",regexp: '^#pid',line: 'pid
   run/nginx.pid;' }
    - name: create dir run and modify directory permission
      file:
        path: "{{ ins_dir }}run"
        owner: "{{ user }}"
        group: "{{ group }}"
        state: directory
    - name: start serivce
      service:
```

```
        name: nginx
        state: started
        enabled: yes
```

2.4.4.3 利用 ansible 的role 实现 nginx 编译安装

```
[root@centos8 nginx_ansible]#tree
.
├── ansible.cfg
├── hosts
├── install_nginx.yml
└── roles
    └── nginx
        ├── handlers
        |   └── main.yml
        ├── tasks
        |   ├── config.yml
        |   ├── group.yml
        |   ├── install.yml
        |   ├── main.yml
        |   ├── package.yml
        |   ├── service.yml
        |   ├── unarchive.yml
        |   ├── unit.yml
        |   └── user.yml
        ├── templates
        |   ├── nginx.conf.j2
        |   └── nginx.service.j2
        └── vars
            └── main.yml

6 directories, 16 files

[root@centos8 nginx_ansible]#cat hosts
[local]
10.0.0.8 ansible_connection=local

[dnsservers]
10.0.0.10

[webservers]
10.0.0.7
10.0.0.18
10.0.0.100
10.0.0.152
[web8servers]
10.0.0.18
10.0.0.28
10.0.0.38
10.0.0.48
10.0.0.58

[web7servers]
10.0.0.7
10.0.0.17
10.0.0.27
10.0.0.37
```

```
[root@centos8 nginx_ansible]#cat roles/nginx/vars/main.yml
nginx_version: 1.18.0
nginx_url: http://nginx.org/download
nginx_full_name: nginx-{{ nginx_version }}.tar.gz
nginx_src: /usr/local/src
nginx_install_dir: /usr/local/nginx

user_name: nginx
group_name: nginx

centos7_packages: [ make,gcc,pcre-devel,openssl-devel,zlib-devel,perl-ExtUtils-
Embed ]
centos8_packages: [ make,gcc-c++,libtool,pcre,pcre-devel,zlib,zlib-
devel,openssl,openssl-devel,perl-ExtUtils-Embed ]
ubuntu_packages: [ make,gcc,libpcre3,libpcre3-dev,openssl,libssl-dev,zlib1g-dev
]


[root@centos8 nginx_ansible]#grep -Ev "#|^$" roles/nginx/templates/nginx.conf.j2
user  {{ user_name }} {{ group_name }};
worker_processes  {{ ansible_processor_vcpus }};
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    server {
        listen       80;
        server_name  localhost;
        location / {
            root   html;
            index  index.html index.htm;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
[root@centos8 nginx_ansible]#grep -Ev "#|^$"
roles/nginx/templates/nginx.service.j2
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target
[Service]
Type=forking
PIDFile={{ nginx_install_dir }}/logs/nginx.pid
ExecStart={{ nginx_install_dir }}/sbin/nginx -c {{ nginx_install_dir
}}/conf/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
LimitNOFILE=100000
```

```
[Install]
WantedBy=multi-user.target

[root@centos8 nginx_ansible]#grep -Ev "#|^$" roles/nginx/handlers/main.yml
- name: restart nginx
  service: name=nginx state=restarted

[root@centos8 nginx_ansible]#cat roles/nginx/tasks/main.yml
- include: group.yml
- include: user.yml
- include: package.yml
- include: unarchive.yml
- include: install.yml
- include: unit.yml
- include: config.yml
- include: service.yml

[root@centos8 nginx_ansible]#cat roles/nginx/tasks/group.yml
- name: create {{ group_name }} group
  group: name={{ group_name }} system=yes
[root@centos8 nginx_ansible]#cat roles/nginx/tasks/user.yml
- name: create {{ user_name }} user
  user: name={{ user_name }} system=yes group={{ group_name }}
shell=/sbin/nologin
[root@centos8 nginx_ansible]#cat roles/nginx/tasks/package.yml
- name: yum nginx dependence packages to centos7
  yum: name={{ centos7_packages }}
  when: ansible_distribution_major_version == "7"
- name: yum nginx dependence packages to centos8
  yum: name={{ centos8_packages }}
  when: ansible_distribution_major_version == "8"
- name: apt nginx dependence packages to ubuntu
  apt: name={{ ubuntu_packages }}
  when: ansible_os_family == "Debian"
- name: download {{ nginx_full_name }}
  get_url: url={{ nginx_url }}/{{ nginx_full_name }} dest={{ nginx_src }}
[root@centos8 nginx_ansible]#cat roles/nginx/tasks/unarchive.yml
- name: unarchive {{ nginx_full_name }}
  unarchive: src={{ nginx_src }}/{{ nginx_full_name }} dest={{ nginx_src }}
 copy=no
[root@centos8 nginx_ansible]#cat roles/nginx/tasks/install.yml
- name: configure
  shell:
    ./configure \
    --prefix={{ nginx_install_dir }} \
    --user=nginx --group=nginx \
    --with-http_ssl_module \
    --with-http_v2_module \
    --with-http_realip_module \
    --with-http_stub_status_module \
    --with-http_gzip_static_module \
    --with-pcre \
    --with-stream \
    --with-stream_ssl_module \
    --with-stream_realip_module
  args:
    chdir: "{{ nginx_src }}/nginx-{{ nginx_version }}"
- name: make && make install
```

```
  shell: make -j {{ ansible_processor_vcpus }} && make install
  args:
    chdir: "{{ nginx_src }}/nginx-{{ nginx_version }}"
- name: add nginx to profile
  copy: content=PATH={{ nginx_install_dir }}/sbin:{{ ansible_env.PATH }}
dest=/etc/profile.d/nginx.sh

[root@centos8 nginx_ansible]#cat roles/nginx/tasks/unit.yml
- name: copy nginx.service to centos
  template: src=nginx.service.j2 dest=/usr/lib/systemd/system/nginx.service
  when: ansible_os_family == "RedHat"
- name: copy nginx.service to ubuntu
  template: src=nginx.service.j2 dest=/lib/systemd/system/nginx.service
  when: ansible_os_family == "Debian"
- name: reload systemd manager configuration
  shell: systemctl daemon-reload
[root@centos8 nginx_ansible]#cat roles/nginx/tasks/config.yml
- name: template config to remote hosts
  template: src=nginx.conf.j2 dest={{ nginx_install_dir }}/conf/nginx.conf
  notify:
    - restart nginx
[root@centos8 nginx_ansible]#cat roles/nginx/tasks/service.yml
- name: start nginx.service
  service: name=nginx state=started enabled=yes
```

2.5 nginx 命令和信号

2.5.1 nginx命令

nginx 命令支持向其发送信号,实现不同功能

nginx 格式

```
nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]
```

选项说明

```
帮助: -? -h
使用指定的配置文件: -c
指定配置指令:-g
指定运行目录:-p
测试配置文件是否有语法错误:-t -T
打印nginx的版本信息、编译信息等:-v -V
发送信号: -s  示例: nginx -s reload
```

信号说明:

```
立刻停止服务:stop,相当于信号SIGTERM,SIGINT
优雅的停止服务:quit,相当于信号SIGQUIT
平滑重启，重新加载配置文件: reload,相当于信号SIGHUP
重新开始记录日志文件:reopen,相当于信号SIGUSR1,在切割日志时用途较大
平滑升级可执行程序:发送信号SIGUSR2,在升级版本时使用
优雅的停止工作进程:发送信号SIGWINCH,在升级版本时使用
```

范例: 查看nginx帮助

```
[root@centos8 ~]#nginx -h
nginx version: nginx/1.14.1
Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit
  -t            : test configuration and exit
  -T            : test configuration, dump it and exit
  -q            : suppress non-error messages during configuration testing
  -s signal     : send signal to a master process: stop, quit, reopen, reload
  -p prefix     : set prefix path (default: /usr/share/nginx/)
  -c filename   : set configuration file (default: /etc/nginx/nginx.conf)
  -g directives : set global directives out of configuration file
```

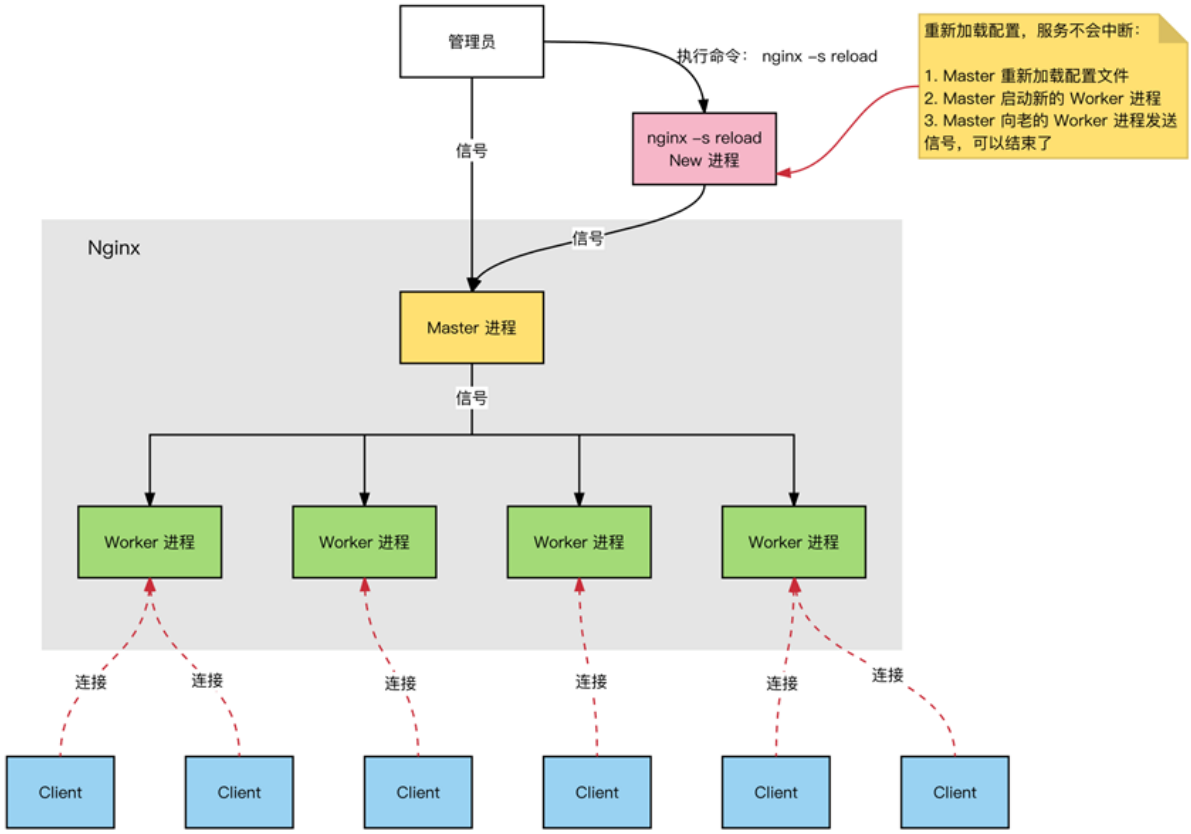
2.5.2 quit 实现worker进程优雅关闭

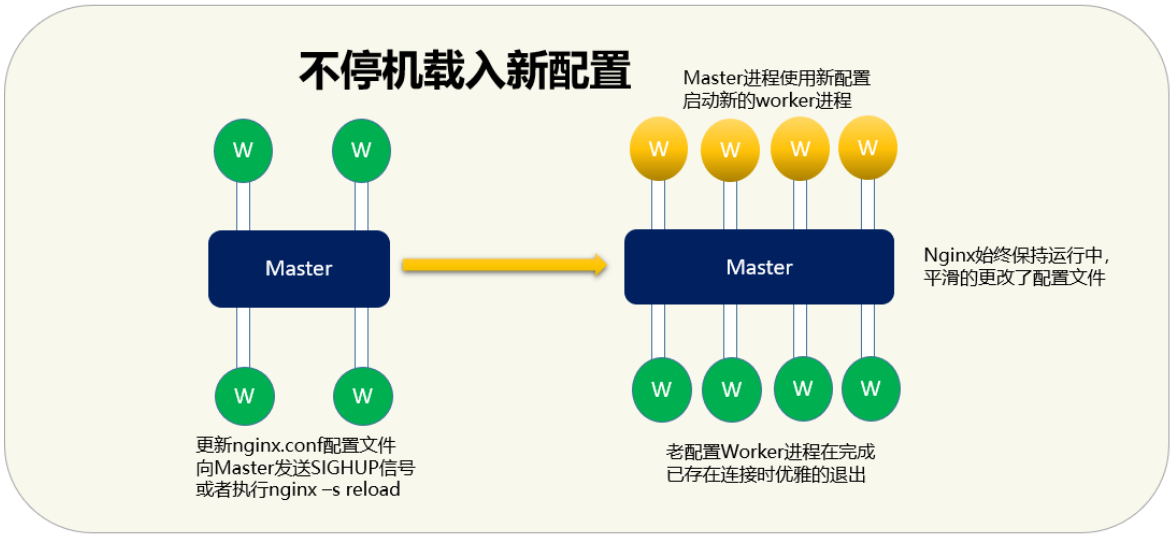
- 设置定时器: worker_shutdown_timeout

  ```
  http://nginx.org/en/docs/ngx_core_module.html#worker_shutdown_timeout

  Syntax: worker_shutdown_timeout time;
  Default:    —
  Context:    main
  This directive appeared in version 1.11.11.

  Configures a timeout for a graceful shutdown of worker processes. When the
  time expires, nginx will try to close all the connections currently open to
  facilitate shutdown.
  ```

- 关闭监听句柄
- 关闭空闲连接
- 在循环中等待全部连接关闭
- 退出nginx所有进程

2.5.3 reload 流程





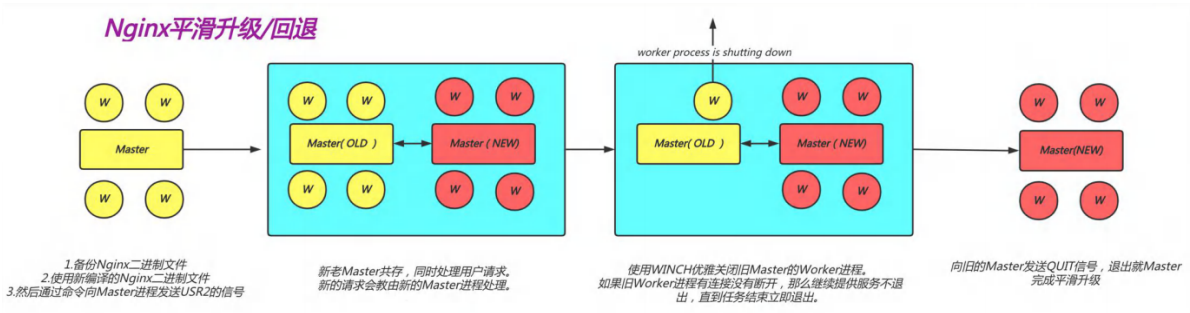
利用 reload 可以实现平滑修改配置并生效

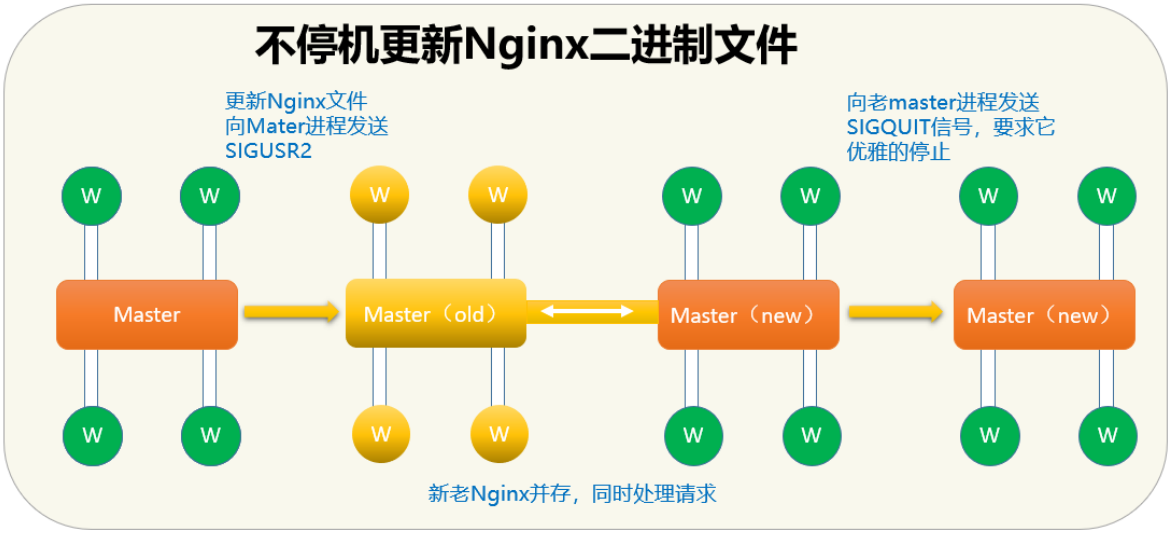
- 向master进程发送HUP信号(reload命令)
- master进程校验配置语法是否正确
- master进程打开新的监听端口
- master进程用新配置启动新的worker子进程
- master进程向老worker子进程发送QUIT信号，老的worker对已建立连接继续处理，处理完才会优雅退出.未关闭的worker旧进程不会处理新来的请求
- 老worker进程关闭监听句柄，处理完当前连接后结束进程

2.6 平滑升级和回滚

有时候我们需要对Nginx版本进行升级以满足对其功能的需求，例如添加新模块，需要新功能，而此时Nginx又在跑着业务无法停掉，这时我们就可能选择平滑升级

2.6.1 平滑升级流程





- 将旧Nginx二进制文件换成新Nginx程序文件（注意先备份）
- 向master进程发送USR2信号
- master进程修改pid文件名加上后缀.oldbin,成为nginx.pid.oldbin
- master进程用新Nginx文件启动新master进程成为旧master的子进程,系统中将有新旧两个Nginx主进程共同提供Web服务,当前新的请求仍然由旧Nginx的worker进程进行处理,将新生成的master进程的PID存放至新生成的pid文件nginx.pid
- 向旧的Nginx服务进程发送WINCH信号，使旧的Nginx worker进程平滑停止
- 向旧master进程发送QUIT信号，关闭老master，并删除Nginx.pid.oldbin文件
- 如果发现升级有问题,可以回滚：向老master发送HUP，向新master发送QUIT

2.6.2 平滑升级和回滚案例

```
#下载最新稳定版
[root@centos8 ~]#wget http://nginx.org/download/nginx-1.20.1.tar.gz
[root@centos8 ~]#tar xvf nginx-1.20.1.tar.gz
[root@centos8 ~]#cd nginx-1.20.1

#查看当前使用的版本及编译选项。结果如下：
[root@centos8 nginx-1.20.1]#/apps/nginx/sbin/nginx -V
nginx version: nginx/1.18.0
built by gcc 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)
built with OpenSSL 1.1.1g FIPS  21 Apr 2020
TLS SNI support enabled
configure arguments: --prefix=/apps/nginx --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream
--with-stream_ssl_module --with-stream_realip_module
```

```
#configure arguments后面是以前编译时的参数。现在编译使用一样的参数

#开始编译新版本
[root@centos8 nginx-1.20.1]#./configure  --prefix=/apps/nginx --user=nginx --
group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre --with-stream --with-stream_ssl_module --with-stream_realip_module

#只要make无需要make install
[root@centos8 nginx-1.20.1]#make
[root@centos8 nginx-1.20.1]#objs/nginx -v
nginx version: nginx/1.20.1

#查看两个版本
[root@centos8 nginx-1.20.1]#ll objs/nginx /apps/nginx/sbin/nginx
-rwxr-xr-x 1 nginx nginx 7591096 Jun  7 16:28 /apps/nginx/sbin/nginx
-rwxr-xr-x 1 root  root  7723272 Jun  7 17:27 objs/nginx

#把之前的旧版的nginx命令备份
[root@centos8 nginx-1.20.1]#cp /apps/nginx/sbin/nginx
/usr/local/nginx/sbin/nginx.old

#把新版本的nginx命令复制过去,注意:在ubuntu18.04上需要加 -f 选项强制覆盖,否则会提示busy
[root@centos8 nginx-1.20.1]#cp -f ./objs/nginx /apps/nginx/sbin/

#检测一下有没有问题
[root@centos8 nginx-1.20.1]#/apps/nginx/sbin/nginx -t

#发送信号USR2 平滑升级可执行程序,将存储有旧版本主进程PID的文件重命名为nginx.pid.oldbin，并启
动新的nginx
#此时两个master的进程都在运行,只是旧的master不在监听,由新的master监听80
#此时Nginx开启一个新的master进程，这个master进程会生成新的worker进程，这就是升级后的Nginx进
程，此时老的进程不会自动退出，但是当接收到新的请求不作处理而是交给新的进程处理。

[root@centos8 nginx-1.20.1]#kill -USR2 `cat /apps/nginx/logs/nginx.pid`

#可以看到两个master,新的master是旧版master的子进程,并生成新版的worker进程
[root@centos8 nginx-1.20.1]#ps auxf|grep nginx
root       12018  0.0  0.0  12112  1092 pts/0    S+   17:32   0:00  |
\_ grep --color=auto nginx
root        8814  0.0  0.2  42460  3760 ?        Ss   16:58   0:00 nginx: master
process /apps/nginx/sbin/nginx -c /apps/nginx/conf/nginx.conf
nginx       8957  0.0  0.2  77172  4724 ?        S    17:23   0:00  \_ nginx:
worker process
nginx       8958  0.0  0.2  77172  4724 ?        S    17:23   0:00  \_ nginx:
worker process
root       12014  0.0  0.3  42448  5512 ?        S    17:32   0:00  \_ nginx:
master process /apps/nginx/sbin/nginx -c /apps/nginx/conf/nginx.conf
nginx      12015  0.0  0.2  77192  4904 ?        S    17:32   0:00      \_
nginx: worker process
nginx      12016  0.0  0.2  77192  4908 ?        S    17:32   0:00      \_
nginx: worker process

[root@centos8 nginx-1.20.1]#lsof -i :80
COMMAND   PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nginx    8814  root    8u  IPv4  40843      0t0  TCP *:http (LISTEN)
nginx    8957 nginx    8u  IPv4  40843      0t0  TCP *:http (LISTEN)
nginx    8958 nginx    8u  IPv4  40843      0t0  TCP *:http (LISTEN)
```

```
nginx   12014  root    8u  IPv4  40843      0t0  TCP *:http (LISTEN)
nginx   12015 nginx    8u  IPv4  40843      0t0  TCP *:http (LISTEN)
nginx   12016 nginx    8u  IPv4  40843      0t0  TCP *:http (LISTEN)


#先关闭旧nginx的worker进程,而不关闭nginx主进程方便回滚
#向原Nginx主进程发送WINCH信号，它会逐步关闭旗下的工作进程（主进程不退出），这时所有请求都会由新版Nginx处理
[root@centos8 nginx-1.20.1]#kill -WINCH `cat /apps/nginx/logs/nginx.pid.oldbin`

#如果旧版worker进程有用户的请求,会一直等待处理完后才会关闭
[root@centos8 nginx-1.20.1]#ps auxf|grep nginx
root       12066  0.0  0.0  12112  1112 pts/0    S+   17:38   0:00  |
\_ grep --color=auto nginx
root        8814  0.0  0.1  42460  2656 ?        Ss   16:58   0:00 nginx: master
process /apps/nginx/sbin/nginx -c /apps/nginx/conf/nginx.conf
nginx       8957  0.0  0.1  77172  3484 ?        S    17:23   0:00  \_ nginx:
worker process is shutting down
root       12014  0.0  0.2  42448  3664 ?        S    17:32   0:00  \_ nginx:
master process /apps/nginx/sbin/nginx -c /apps/nginx/conf/nginx.conf
nginx      12015  0.0  0.1  77192  3284 ?        S    17:32   0:00      \_
nginx: worker process
nginx      12016  0.0  0.1  77192  3280 ?        S    17:32   0:00      \_
nginx: worker process

[root@centos8 nginx-1.20.1]#pstree -p|grep nginx
           |-nginx(8814)---nginx(12014)-+-nginx(12015)
           |                            `-nginx(12016)


#经过一段时间测试，新版本服务没问题，最后发送QUIT信号,退出老的master
[root@centos8 nginx-1.20.1]#kill -QUIT `cat /apps/nginx/logs/nginx.pid.oldbin`

#查看版本是不是已经是新版了
[root@centos8 nginx-1.20.1]#nginx -v
nginx version: nginx/1.20.1
[root@centos8 nginx-1.20.1]#curl -I 127.0.0.1
HTTP/1.1 200 OK
Server: nginx/1.20.1
Date: Mon, 07 Jun 2021 09:48:48 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Mon, 07 Jun 2021 08:28:12 GMT
Connection: keep-alive
ETag: "60bdd89c-264"
Accept-Ranges: bytes

#回滚
#如果升级的版本发现问题需要回滚,可以发送HUP信号,重新拉起旧版本的worker
[root@centos8 nginx-1.20.1]#kill -HUP `cat /apps/nginx/logs/nginx.pid.oldbin`
[root@centos8 nginx-1.20.1]#pstree -p |grep nginx
           |-nginx(8814)-+-nginx(12014)-+-nginx(12015)
           |             |              `-nginx(12016)
           |             |-nginx(12090)
           |             `-nginx(12091)

#最后关闭新版的master
[root@centos8 nginx-1.20.1]#kill -QUIT `cat /apps/nginx/logs/nginx.pid`
```

3 Nginx 核心配置详解

3.1 配置文件说明

nginx 官方帮助文档

```
http://nginx.org/en/docs/
```

tengine 帮助文档

```
http://tengine.taobao.org/nginx_docs/cn/docs/
```

Nginx的配置文件的组成部分：

- 主配置文件：nginx.conf
- 子配置文件：include conf.d/*.conf
- fastcgi， uwsgi，scgi 等协议相关的配置文件
- mime.types：支持的mime类型，MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型，MIME消息能包含文本、图像、音频、视频以及其他应用程序专用的数据，是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名，以及一些媒体文件打开方式。MIME参考文档：https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_Types

nginx 配置文件格式说明

```
配置文件由指令与指令块构成
每条指令以;分号结尾，指令与值之间以空格符号分隔
可以将多条指令放在同一行,用分号分隔即可,但可读性差,不推荐
指令块以{ }大括号将多条指令组织在一起,且可以嵌套指令块
include语句允许组合多个配置文件以提升可维护性
使用#符号添加注释，提高可读性
使用$符号使用变量
部分指令的参数支持正则表达式
```

Nginx 主配置文件的配置指令方式：

```
directive value [value2 ...];

注意
(1) 指令必须以分号结尾
(2) 支持使用配置变量
    内建变量：由Nginx模块引入，可直接引用
    自定义变量：由用户使用set命令定义,格式: set variable_name value;
    引用变量：$variable_name
```

主配置文件结构：四部分

```
main block：主配置段，即全局配置段，对http,mail都有效
```

```
#事件驱动相关的配置
event {
            ...
}

#http/https 协议相关配置段
http {
        ...
}

#默认配置文件不包括下面两个块
#mail 协议相关配置段
mail {
        ...
}

#stream 服务器相关配置段
stream {
        ...
}
```

默认的nginx.conf 配置文件格式说明

```
#全局配置端，对全局生效，主要设置nginx的启动用户/组，启动的工作进程数量，工作模式，Nginx的PID路径，日志路径等。
user  nginx nginx;
worker_processes  1;    #启动工作进程数数量
events { #events设置快，主要影响nginx服务器与用户的网络连接，比如是否允许同时接受多个网络连接，使用哪种事件驱动模型处理请求，每个工作进程可以同时支持的最大连接数，是否开启对多工作进程下的网络连接进行序列化等。
     worker_connections  1024;    #设置单个nginx工作进程可以接受的最大并发，作为web服务器的时候最大并发数为worker_connections * worker_processes，作为反向代理的时候为(worker_connections * worker_processes)/2
}
http { #http块是Nginx服务器配置中的重要部分，缓存、代理和日志格式定义等绝大多数功能和第三方模块都可以在这设置，http块可以包含多个server块，而一个server块中又可以包含多个location块，server块可以配置文件引入、MIME-Type定义、日志自定义、是否启用sendfile、连接超时时间和单个链接的请求上限等。
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on; #作为web服务器的时候打开sendfile加快静态文件传输，指定是否使用sendfile系统调用来传输文件,sendfile系统调用在两个文件描述符之间直接传递数据(完全在内核中操作)，从而避免了数据在内核缓冲区和用户缓冲区之间的拷贝，操作效率很高，被称之为零拷贝，硬盘 >> kernel buffer (快速拷贝到kernelsocket buffer) >>协议栈。
    keepalive_timeout  65;  #长连接超时时间，单位是秒
    server { #设置一个虚拟机主机，可以包含自己的全局快，同时也可以包含多个location模块。比如本虚拟机监听的端口、本虚拟机的名称和IP配置，多个server 可以使用一个端口，比如都使用80端口提供web服务、
        listen       80;  #配置server监听的端口
        server_name  localhost; #本server的名称，当访问此名称的时候nginx会调用当前serevr内部的配置进程匹配。
        location / { #location其实是server的一个指令，为nginx服务器提供比较多而且灵活的指令，都是在location中体现的，主要是基于nginx接受到的请求字符串，对用户请求的UIL进行匹配，并对特定的指令进行处理，包括地址重定向、数据缓存和应答控制等功能都是在这部分实现，另外很多第三方模块的配置也是在location模块中配置。
            root   html; #相当于默认页面的目录名称，默认是安装目录的相对路径，可以使用绝对路径配置。
```

```
            index  index.html index.htm; #默认的页面文件名称
        }
        error_page   500 502 503 504  /50x.html; #错误页面的文件名称
        location = /50x.html { #location处理对应的不同错误码的页面定义到/50x.html，这个跟对应其server中定义的目录下。
            root   html;  #定义默认页面所在的目录
        }
    }

#和邮件相关的配置
#mail {
#                ...
#        }          mail 协议相关配置段

#tcp代理配置，1.9版本以上支持
#stream {
#                ...
#        }       stream 服务器相关配置段

#导入其他路径的配置文件
#include /apps/nginx/conf.d/*.conf
}
```

3.2 全局配置

Main 全局配置段常见的配置指令分类

- 正常运行必备的配置
- 优化性能相关的配置
- 用于调试及定位问题相关的配置
- 事件驱动相关的配置

全局配置说明:

```
user  nginx nginx; #启动Nginx工作进程的用户和组
worker_processes  [number | auto]; #启动Nginx工作进程的数量,一般设为和CPU核心数相同
worker_cpu_affinity 00000001 00000010 00000100 00001000 | auto ; #将Nginx工作进程绑定到指定的CPU核心，默认Nginx是不进行进程绑定的，绑定并不是意味着当前nginx进程独占以一核心CPU，但是可以保证此进程不会运行在其他核心上，这就极大减少了nginx的工作进程在不同的cpu核心上的来回跳转，减少了CPU对进程的资源分配与回收以及内存管理等，因此可以有效的提升nginx服务器的性能。
CPU MASK: 00000001：0号CPU
          00000010：1号CPU
          10000000：7号CPU
#示例:
worker_cpu_affinity 0001 0010 0100 1000;第0号---第3号CPU
worker_cpu_affinity 0101 1010;

#示例
worker_processes  4;
worker_cpu_affinity 00000010 00001000 00100000 10000000;
[root@centos8 ~]# ps axo pid,cmd,psr | grep nginx
 31093 nginx: master process /apps   1
 34474 nginx: worker process         1
 34475 nginx: worker process         3
 34476 nginx: worker process         5
 34477 nginx: worker process         7
 35751 grep nginx
```

```
#auto 绑定CPU
#The special value auto (1.9.10) allows binding worker processes automatically to
available CPUs:
worker_processes auto;
worker_cpu_affinity auto;
#The optional mask parameter can be used to limit the CPUs available for
automatic binding:
worker_cpu_affinity auto 01010101;

#错误日志记录配置，语法：error_log file  [debug | info | notice | warn | error | crit
| alert | emerg]
#error_log  logs/error.log;
#error_log  logs/error.log  notice;
error_log  /apps/nginx/logs/error.log error;

#pid文件保存路径
pid        /apps/nginx/logs/nginx.pid;

worker_priority 0; #工作进程优先级，-20~20(19)
worker_rlimit_nofile 65536; #所有worker进程能打开的文件数量上限,包括:Nginx的所有连接（例
如与代理服务器的连接等），而不仅仅是与客户端的连接,另一个考虑因素是实际的并发连接数不能超过系统级
别的最大打开文件数的限制.最好与ulimit -n 或者limits.conf的值保持一致,

daemon off;  #前台运行Nginx服务用于测试、docker等环境。
master_process off|on; #是否开启Nginx的master-worker工作模式，仅用于开发调试场景,默认为
on

events {
    worker_connections  65536;  #设置单个工作进程的最大并发连接数
    use epoll; #使用epoll事件驱动，Nginx支持众多的事件驱动，比如:select、poll、epoll，只
能设置在events模块中设置。
    accept_mutex on; #on为同一时刻一个请求轮流由work进程处理,而防止被同时唤醒所有worker,避
免多个睡眠进程被唤醒的设置，默认为off，新请求会唤醒所有worker进程,此过程也称为"惊群"，因此
nginx刚安装完以后要进行适当的优化。建议设置为on
    multi_accept on; #on时Nginx服务器的每个工作进程可以同时接受多个新的网络连接，此指令默认
为off，即默认为一个工作进程只能一次接受一个新的网络连接，打开后几个同时接受多个。建议设置为on
}
```

范例: 实现 nginx 的高并发配置

```
[root@centos7 ~]#ulimit  -n 102400
[root@centos7 ~]#while true;do  ab -c 5000 -n 10000 http://10.0.0.8/;sleep
0.5;done

#默认配置不支持高并发,会出现以下错误日志
[root@centos8 conf]#tail /apps/nginx/logs/error.log
2020/09/24 21:19:33 [crit] 41006#0: *1105860 open() "/apps/nginx/html/50x.html"
failed (24: Too many open files), client: 10.0.0.7, server: localhost, request:
"GET / HTTP/1.0", host: "10.0.0.8"
2020/09/24 21:19:33 [crit] 41006#0: accept4() failed (24: Too many open files)
2020/09/24 21:19:33 [crit] 41006#0: *1114177 open()
"/apps/nginx/html/index.html" failed (24: Too many open files), client: 10.0.0.7,
server: localhost, request: "GET / HTTP/1.0", host: "10.0.0.8"


#如果systemd启动,则需要修改nginx.service文件中加LimitNOFILE=100000,才能有效
```

```
#如果非systemd启动,可以修改下面pam限制
[root@centos8 ~]#vim /etc/security/limits.conf
*                soft    nofile          1000000
*                hard    nofile          1000000


[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf
worker_rlimit_nofile 100000;
[root@centos8 ~]#systemctl restart nginx

[root@centos8 ~]# watch -n1  'ps -axo pid,cmd,nice | grep nginx #验证进程优先级
```

3.3 http 配置块

http 协议相关的配置结构

```
http {
    ...
    ...  #各server的公共配置
    server {    #每个server用于定义一个虚拟主机,第一个server为默认虚拟服务器
        ...
    }
    server {
        ...
        server_name   #虚拟主机名
        root     #主目录
        alias     #路径别名
        location [OPERATOR] URL {     #指定URL的特性
            ...
            if CONDITION {
                ...
            }
        }
    }
}
```

http 协议配置说明

```
http {
    include       mime.types; #导入支持的文件类型,是相对于/apps/nginx/conf的目录
    default_type  application/octet-stream; #除mime.types中文件类型外,设置其它文件默认
类型，访问其它类型时会提示下载不匹配的类型文件

#日志配置部分
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';
    #access_log  logs/access.log  main;

#自定义优化参数
    sendfile        on;
    #tcp_nopush     on; #在开启了sendfile的情况下，合并请求后统一发送给客户端,必须开启
sendfile
```

```
    #tcp_nodelay    off; #在开启了keepalived模式下的连接是否启用TCP_NODELAY选项，当为off时，延迟0.2s发送，默认On时，不延迟发送，立即发送用户响应报文。
    #keepalive_timeout  0;
    keepalive_timeout  65 65; #设置会话保持时间,第二个值为响应首部:keep-Alived:timeout=65,可以和第一个值不同
    #gzip  on; #开启文件压缩

    server {
        listen       80; #设置监听地址和端口
        server_name  localhost; #设置server name，可以以空格隔开写多个并支持正则表达式，如:*.magedu.com    www.magedu.*  ~^www\d+\.magedu\.com$ default_server
        #charset koi8-r; #设置编码格式，默认是俄语格式，建议改为utf-8
        #access_log  logs/host.access.log  main;
        location / {
            root   html;
            index  index.html index.htm;
        }

        #error_page  404              /404.html;
        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html; #定义错误页面
        location = /50x.html {
            root   html;
        }

        # proxy the PHP scripts to Apache listening on 127.0.0.1:80
        #
        #location ~ \.php$ { #以http的方式转发php请求到指定web服务器
        #    proxy_pass   http://127.0.0.1;
        #}

        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
        #
        #location ~ \.php$ { #以fastcgi的方式转发php请求到php处理
        #    root           html;
        #    fastcgi_pass   127.0.0.1:9000;
        #    fastcgi_index  index.php;
        #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
        #    include        fastcgi_params;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht { #拒绝web形式访问指定文件，如很多的网站都是通过.htaccess文件来改变自己的重定向等功能。
        #    deny  all;
        #}
        location ~ /passwd.html {
            deny  all;
        }

    }

    # another virtual host using mix of IP-, name-, and port-based configuration
    #
    #server { #自定义虚拟server
```

```
    #    listen       8000;
    #    listen       somename:8080;
    #    server_name  somename  alias  another.alias;

    #    location / {
    #        root   html;
    #        index  index.html index.htm; #指定默认网页文件，此指令由
ngx_http_index_module模块提供

    #    }
    #}

    # HTTPS server
    #
    #server { #https服务器配置
    #    listen       443 ssl;
    #    server_name  localhost;

    #    ssl_certificate      cert.pem;
    #    ssl_certificate_key  cert.key;

    #    ssl_session_cache    shared:SSL:1m;
    #    ssl_session_timeout  5m;

    #    ssl_ciphers  HIGH:!aNULL:!MD5;
    #    ssl_prefer_server_ciphers  on;

    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #}
```

3.3.1 MIME

```
#在响应报文中将指定的文件扩展名映射至MIME对应的类型
include           /etc/nginx/mime.types;
default_type      application/octet-stream;#除mime.types中的类型外，指定其它文件的默认
MIME类型，浏览器一般会提示下载
types {
    text/html  html;
    image/gif  gif;
    image/jpeg jpg;
}

#MIME参考文档：
https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_Types
```

范例: 识别php文件为text/html

```
[root@centos8 ~]#cat /apps/nginx/html/test.php
<?php
phpinfo();
?>

[root@centos7 ~]#curl 10.0.0.8/test.php -I
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Thu, 24 Sep 2020 13:54:39 GMT
Content-Type: application/octet-stream
Content-Length: 20
Last-Modified: Thu, 24 Sep 2020 13:53:26 GMT
Connection: keep-alive
ETag: "5f6ca4d6-14"
Accept-Ranges: bytes


[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf
http {
    include       mime.types;
    default_type  text/html;
......
[root@centos8 ~]#nginx -s reload

[root@centos7 ~]#curl 10.0.0.8/index.html -I
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Thu, 24 Sep 2020 13:56:26 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Thu, 24 Sep 2020 12:13:03 GMT
Connection: keep-alive
ETag: "5f6c8d4f-264"
Accept-Ranges: bytes
```

3.3.2 指定响应报文server首部

```
#是否在响应报文中的Content-Type显示指定的字符集，默认off不显示
charset charset | off;

#示例
charset utf-8;

#是否在响应报文的Server首部显示nginx版本
server_tokens on | off | build | string;
```

范例: 修改server字段

```
如果想自定义响应报文的nginx版本信息，需要修改源码文件，重新编译
如果server_tokens on，修改 src/core/nginx.h  修改第13-14行，如下示例
#define NGINX_VERSION      "1.68.9"
#define NGINX_VER          "wanginx/" NGINX_VERSION

如果server_tokens off，修改 src/http/ngx_http_header_filter_module.c
第49行，如下示例：
static char ngx_http_server_string[] = "Server: nginx" CRLF;
把其中的nginx改为自己想要的文字即可,如：wanginx
```

范例: 修改 Server 头部信息

```
[root@centos8 ~]#vim /usr/local/src/nginx-1.18.0/src/core/nginx.h
#define NGINX_VERSION      "1.68.9"
#define NGINX_VER          "wanginx/" NGINX_VERSION

[root@centos8 ~]#vim nginx-1.18.0/src/http/ngx_http_header_filter_module.c
static u_char ngx_http_server_string[] = "Server: magedu-nginx" CRLF;

[root@centos7 ~]#curl -I www.magedu.org
HTTP/1.1 200 OK
Server: wanginx/1.68.9
Date: Thu, 24 Sep 2020 07:44:05 GMT
Content-Type: text/html
Content-Length: 8
Last-Modified: Wed, 23 Sep 2020 14:39:21 GMT
Connection: keep-alive
ETag: "5f6b5e19-8"
Accept-Ranges: bytes


[root@centos8 ~]#vim /apps/nginx/conf/conf.d/pc.conf
server {
     ......
     server_tokens off;
     ......

[root@centos7 ~]#curl -I www.magedu.org
HTTP/1.1 200 OK
Server: magedu-nginx
Date: Thu, 24 Sep 2020 07:44:59 GMT
Content-Type: text/html
Content-Length: 8
Last-Modified: Wed, 23 Sep 2020 14:39:21 GMT
Connection: keep-alive
ETag: "5f6b5e19-8"
Accept-Ranges: bytes
```

3.4 核心配置示例

基于不同的IP、不同的端口以及不用得域名实现不同的虚拟主机，依赖于核心模块ngx_http_core_module实现。

3.4.1 新建一个 PC web 站点

```
#定义子配置文件路径
[root@centos8 ~]# mkdir  /apps/nginx/conf/conf.d
[root@centos8 ~]# vim /apps/nginx/conf/nginx.conf
http {
    ......
    include /apps/nginx/conf/conf.d/*.conf; #在配置文件的最后面添加此行,注意不要放在最前面,会导致前面的命令无法生效
}

#创建PC网站配置
[root@centos8 ~]# cat /apps/nginx/conf/conf.d/pc.conf
```

```
server {
  listen 80;
  server_name www.magedu.org;
  location / {
    root /data/nginx/html/pc;
  }
}

[root@centos8 ~]# mkdir -p /data/nginx/html/pc
[root@centos8 ~]# echo "pc web" > /data/nginx/html/pc/index.html


[root@centos8 ~]# systemctl   reload nginx
#访问测试
```

3.4.2 新建一个 Mobile web 站点


```
[root@centos8 ~]# cat  /apps/nginx/conf/conf.d/mobile.conf
server {
  listen 80;
  server_name m.magedu.org;  #指定第二个站点名称
  location / {
    root /data/nginx/html/mobile;
  }
}
[root@centos8 ~]# mkdir -p /data/nginx/html/mobile
[root@centos8 ~]# echo "mobile web" >> /data/nginx/html/mobile/index.html

[root@centos8 ~]# systemctl  reload nginx
```

3.4.3 root 与 alias

root：指定web的家目录，在定义location的时候，文件的绝对路径等于 root+location

范例:

```
server {
  listen 80;
  server_name www.magedu.org;
  location / {
    root /data/nginx/html/pc;
  }

  location /about {
    root /opt/html; #必须要在html目录中创建一个名为about的目录才可以访问，否则报错。
  }
}

[root@centos8 ~]# mkdir  -p /opt/html/about
[root@centos8 ~]# echo about > /opt/html/about/index.html

#重启Nginx并访问测试
```

alias：定义路径别名，会把访问的路径重新定义到其指定的路径,文档映射的另一种机制;仅能用于location上下文,此指令使用较少

范例:

```
server {
  listen 80;
  server_name www.magedu.org;
  location / {
    root /data/nginx/html/pc;
  }

  location /about { #注意about后不要加/ ，使用alias的时候uri后面如果加了斜杠,则下面的路径配置必须加斜杠，否则403
    alias /opt/html/about; #当访问about的时候，会显示alias定义的/opt/html/about里面的内容。
  }
}

#重启Nginx并访问测试
```

注意：location中使用root指令和alias指令的意义不同

```
root 给定的路径对应于location中的/uri 左侧的/
alias 给定的路径对应于location中的/uri 的完整路径
```

3.4.4 location 的详细使用

在一个server中location配置段可存在多个，用于实现从uri到文件系统的路径映射；ngnix会根据用户请求的URI来检查定义的所有location，按一定的优先级找出一个最佳匹配，而后应用其配置

在没有使用正则表达式的时候，nginx会先在server中的多个location选取匹配度最高的一个uri，uri是用户请求的字符串，即域名后面的web文件路径，然后使用该location模块中的正则url和字符串，如果匹配成功就结束搜索，并使用此location处理此请求。

location 官方帮助:

```
http://nginx.org/en/docs/http/ngx_http_core_module.html#location
```

```
#语法规则：
location [ = | ~ | ~* | ^~ ] uri { ... }

=    #用于标准uri前，需要请求字串与uri精确匹配，大小敏感,如果匹配成功就停止向下匹配并立即处理请求
^~   #用于标准uri前，表示包含正则表达式,并且匹配以指定的正则表达式开头,对uri的最左边部分做匹配检查，不区分字符大小写
~    #用于标准uri前，表示包含正则表达式,并且区分大小写
~*   #用于标准uri前，表示包含正则表达式,并且不区分大写
不带符号     #匹配起始于此uri的所有的uri
\    #用于标准uri前，表示包含正则表达式并且转义字符。可以将 . * ?等转义为普通符号

#匹配优先级从高到低：
=, ^~, ~/~*, 不带符号
```

官方范例

```
location = / {
    [ configuration A ]
}

location / {
    [ configuration B ]
}

location /documents/ {
    [ configuration C ]
}

location ^~ /images/ {
    [ configuration D ]
}

location ~* \.(gif|jpg|jpeg)$ {
    [ configuration E ]
}
The “/” request will match configuration A(?), the “/index.html” request will
match configuration B,
the “/documents/document.html” request will match configuration C, the
“/images/1.gif” request will match configuration D, and the “/documents/1.jpg”
request will match configuration E.
```

3.4.4.1 匹配案例-精确匹配

在server部分使用location配置一个web界面，例如：当访问nginx 服务器的/logo.jpg的时候要显示指定html文件的内容

精确匹配一般用于匹配组织的logo等相对固定的URL,匹配优先级最高

范例: 精确匹配 logo

```
[root@centos8 ~]# cat /apps/nginx/conf/conf.d/pc.conf
server {
  listen 80;
  server_name www.magedu.org;
  location / {
    root /data/nginx/html/pc;
  }
  location = /logo.jpg {
    root /data/nginx/images;
    index index.html;
  }
}

#上传logo.jpg图片到/data/nginx/images，重启Nginx并访问测试
#访问测试：http://www.magedu.org/logo.jpg
```

3.4.4.2 匹配案例-区分大小写

~ 实现区分大小写的模糊匹配. 以下范例中, 如果访问uri中包含大写字母的JPG，则以下location匹配Ax.jpg条件不成功，因为 ~ 区分大小写，当用户的请求被执行匹配时发现location中定义的是小写的jpg，本次访问的uri匹配失败，后续要么继续往下匹配其他的location（如果有），要么报错给客户端

```
location ~ /A.?\.jpg { #匹配字母A开头的jpg图片，后面?表示A后面零次或一个字符
    index index.html;
    root /data/nginx/html/image;
}

#重启Nginx并访问测试
#将只能访问以小写字符的jpg图片，不能识别大写的JPG结尾的图片
```

3.4.4.3 匹配案例-不区分大小写

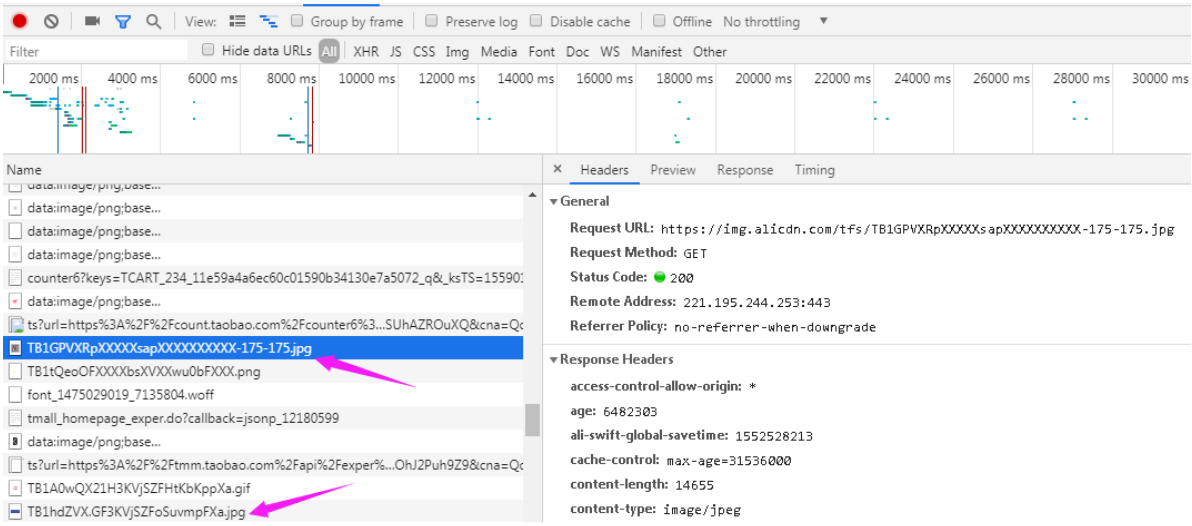
~* 用来对用户请求的uri做模糊匹配，uri中无论都是大写、都是小写或者大小写混合，此模式也都会匹配，通常使用此模式匹配用户request中的静态资源并继续做下一步操作,此方式使用较多

注意: 此方式中,对于Linux文件系统上的文件仍然是区分大小写的,如果磁盘文件不存在,仍会提示404

```
#正则表达式匹配:
  location ~* /A.?\.jpg {
    index index.html;
    root /opt/nginx/html/image;
  }
```

#重启Nginx并访问测试

对于不区分大小写的location，则可以访问任意大小写结尾的图片文件,如区分大小写则只能访问Aa.jpg此类文件，不区分大小写则可以访问除了aa.jpg以外,还有其它的资源比如Aa.JPG、aA.jPG这样的混合名称文件，但是还同时也要求nginx服务器的资源目录有相应的文件，比如:必须有Aa.JPG,aA.jPG这样文件存在。



3.4.4.4 匹配案例-URI开始

```
location ^~ /images {
    root /data/nginx/;
    index index.html;
  }

location  /api {
    alias /data/nginx/api;
    index index.html;
}

#重启Nginx并访问测试，实现效果是访问/images和/app返回不同的结果
[root@centos8 ~]# curl  http://www.magedu.org/images/
```

```
images
[root@centos8 ~]# curl  http://www.magedu.org/api/
api web
```

3.4.4.5 匹配案例-文件名后缀

```
[root@centos8 ~]# mkdir /data/nginx/static
#上传一个图片到/data/nginx/static

  location ~* \.(gif|jpg|jpeg|bmp|png|tiff|tif|ico|wmf|js|css)$ {
    root /data/nginx/static;
    index index.html;
  }

#重启Nginx并访问测试
```

3.4.4.6 匹配案例-优先级

```
location = /1.jpg {
    root /data/nginx/static1;
    index index.html;
  }

  location  /1.jpg {
   root /data/nginx/static2;
   index index.html;
}

  location ~* \.(gif|jpg|jpeg|bmp|png|tiff|tif|ico|wmf|js)$ {
    root /data/nginx/static3;
    index index.html;
 }

[root@centos8 ~]# mkdir -p /data/nginx/static{1,2,3}
#上传图片到 /data/nginx/static{1,2,3} 并重启nginx访问测试

#匹配优先级: =, ^~, ~/~*, /
location优先级: (location =) > (location ^~ 路径) > (location ~,~* 正则顺序) >
(location 完整路径) > (location 部分起始路径) > (/)
```

3.4.4.7 生产使用案例

```
#直接匹配网站根会加速Nginx访问处理
location = /index.html {
   ......;
}

location / {
   ......;
}

#静态资源配置方法1
location ^~ /static/ {
    ......;
}
```

```
#静态资源配置方法2,应用较多
location ~* \.(gif|jpg|jpeg|png|css|js|ico)$ {
    ......;
}

#多应用配置
location ~* /app1  {
      ......;
}
location ~* /app2  {
      ......;
}
```

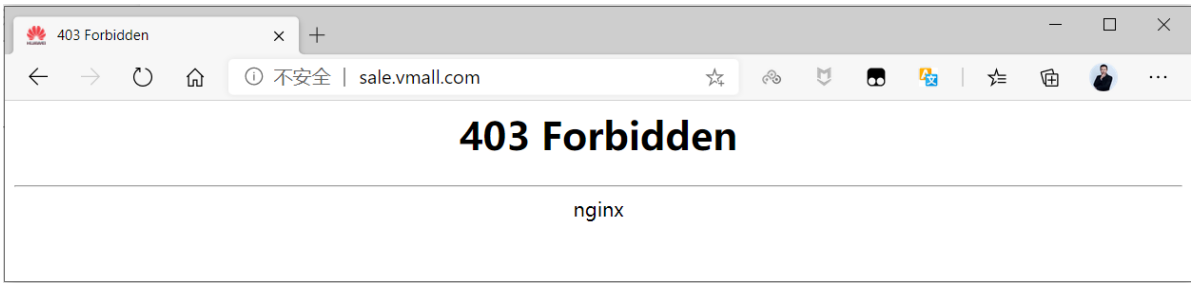
3.4.5 Nginx 四层访问控制

访问控制基于模块ngx_http_access_module实现，可以通过匹配客户端源IP地址进行限制

注意: 如果能在防火墙设备控制,最好就不要在nginx上配置,可以更好的节约资源

官方帮助:

```
http://nginx.org/en/docs/http/ngx_http_access_module.html
```



范例:

```
location = /login/ {
    root /data/nginx/html/pc;
    allow 10.0.0.0/24;
    deny all;
  }

location /about {
    alias /data/nginx/html/pc;
    index index.html;
    deny  192.168.1.1;
    allow 192.168.1.0/24;
    allow 10.1.1.0/16;
    allow 2001:0db8::/32;
    deny  all;  #按先小范围到大范围排序
  }
```

3.4.6 Nginx 账户认证功能

由 ngx_http_auth_basic_module 模块提供此功能

官方帮助:

```
http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html
```

范例:

```
#CentOS安装包
[root@centos8 ~]#yum -y install httpd-tools
#Ubuntu安装包
[root@Ubuntu ~]#apt -y install apache2-utils

#创建用户
#-b 非交互式方式提交密码
[root@centos8 ~]# htpasswd  -cb /apps/nginx/conf/.htpasswd user1 123456
Adding password for user user1
[root@centos8 ~]# htpasswd  -b /apps/nginx/conf/.htpasswd user2 123456
Adding password for user user2
[root@centos8 ~]# tail /apps/nginx/conf/.htpasswd
user1:$apr1$RjmOu2Kr$VHvkAIc5OYg.3ZoaGwaGq/
user2:$apr1$nIqnxoJB$LR9W1DTJT.viDJhXa6wHv.

#安全加固
[root@centos8 ~]# chown nginx.nginx /apps/nginx/conf/.htpasswd
[root@centos8 ~]# chmod 600 /apps/nginx/conf/.htpasswd



[root@centos8 ~]# vim  /apps/nginx/conf/conf.d/pc.conf
  location = /login/ {
    root /data/nginx/html/pc;
    index index.html;
    auth_basic           "login password";
    auth_basic_user_file  /apps/nginx/conf/.htpasswd;
  }

#重启Nginx并访问测试

[root@centos6 ~]#curl http://user1:123456@www.magedu.org/login/
login page
[root@centos6 ~]#curl -u user2:123456  www.magedu.org/login/
login page
```

3.4.7 自定义错误页面

自 定义错误页，同时也可以用指定的响应状态码进行响应, 可用位置：http, server, location, if in location

```
error_page code ... [=[response]] uri;
```

范例:

```
listen 80;
server_name www.magedu.org;
error_page  500 502 503 504 /error.html;
location = /error.html {
   root /data/nginx/html;
}

#重启nginx并访问不存在的页面进行测试
```

范例: 自定义错误页面

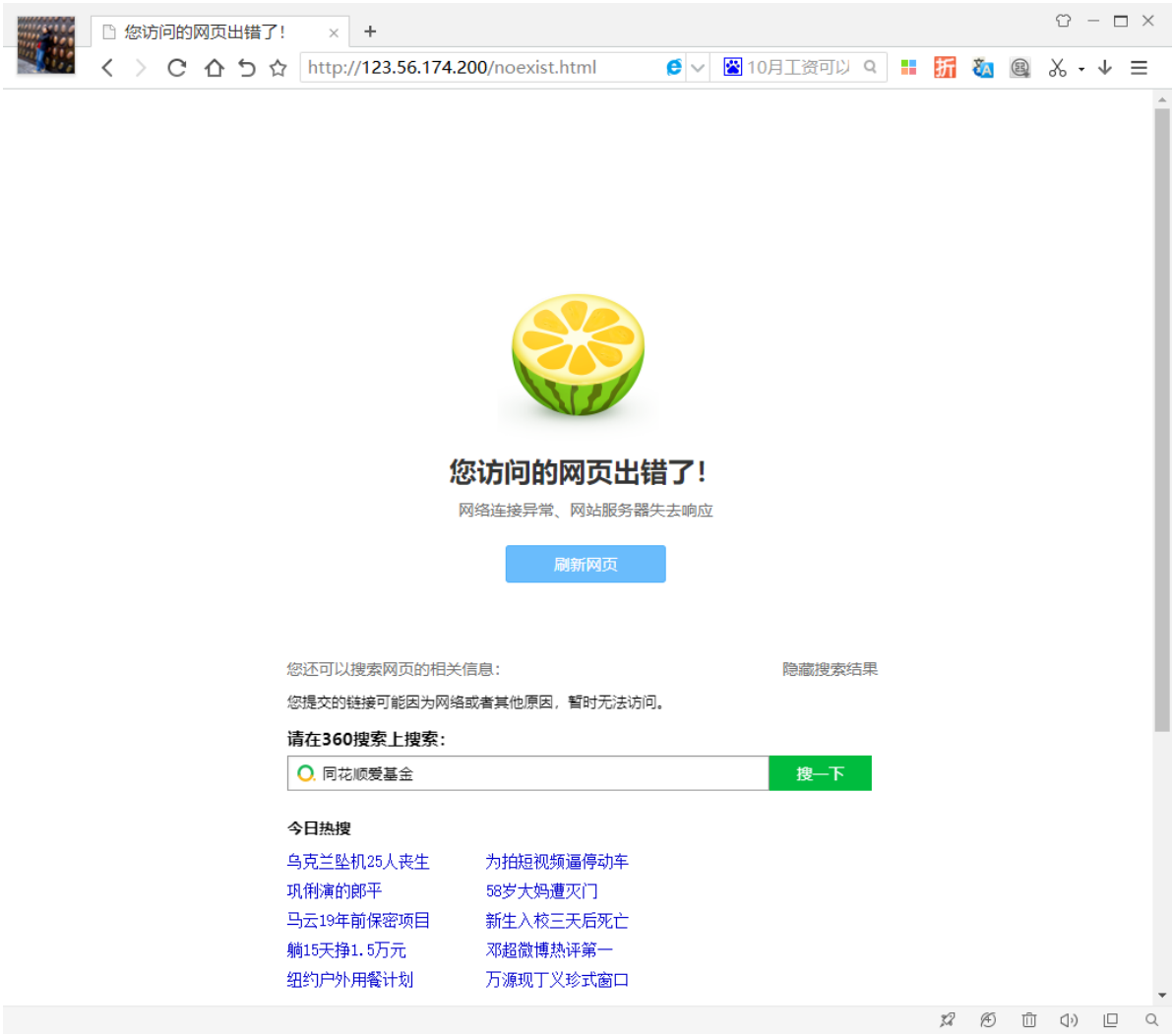
```
error_page 404 /40x.html;
location = /40x.html {
    root /data/html/ ;
}
```

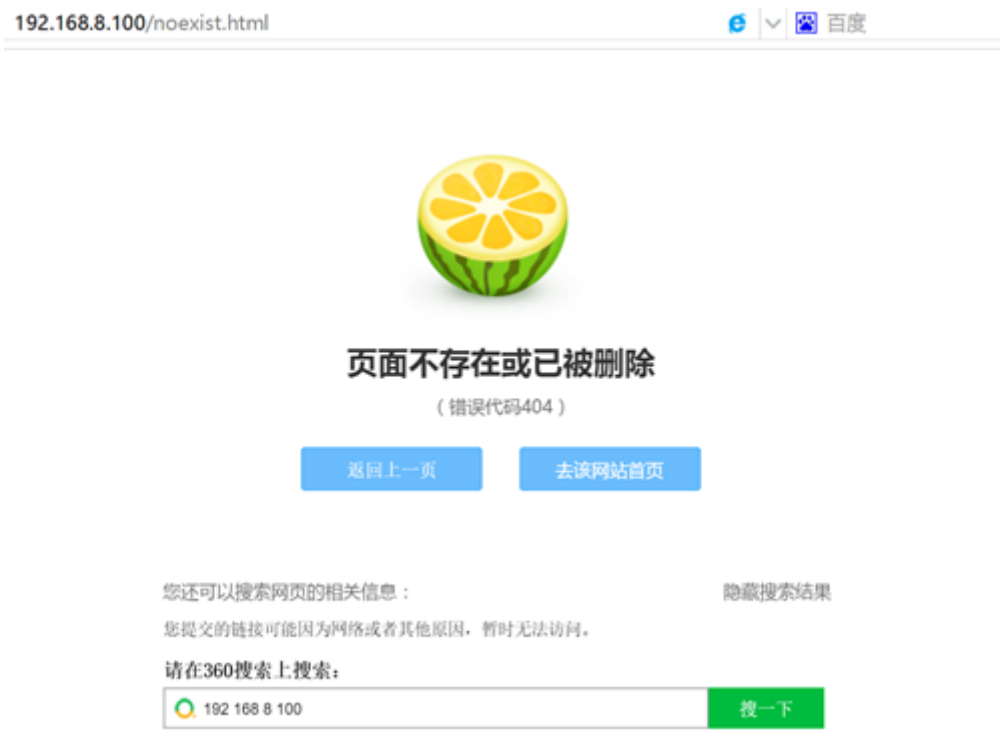
范例: 如果404,就转到主页

```
#404转为302
#error_page 404 /index.html;
error_page  404  =302  /index.html;
error_page 500 502 503 504 /50x.html;
      location = /50x.html {
}
```

范例: 浏览器"截胡"

使用360浏览器访问基于IP的web主机上面的不存在的页面时,会自动用广告页面代替错误页面


您访问的网页出错了！
http://123.56.174.200/noexist.html
10月工资可以
您访问的网页出错了！
网络连接异常、网站服务器失去响应
刷新网页
您还可以搜索网页的相关信息：
隐藏搜索结果
您提交的链接可能因为网络或者其他原因，暂时无法访问。
请在360搜索上搜索：
同花顺爱基金
搜一下
今日热搜
乌克兰坠机25人丧生
为拍短视频逼停动车
巩俐演的郎平
58岁大妈遭灭门
马云19年前保密项目
新生入校三天后死亡
躺15天挣1.5万元
邓超微博热评第一
纽约户外用餐计划
万源现丁义珍式窗口


192.168.8.100/noexist.html
百度
页面不存在或已被删除
（错误代码404）
返回上一页
去该网站首页
您还可以搜索网页的相关信息：
隐藏搜索结果
您提交的链接可能因为网络或者其他原因，暂时无法访问。
请在360搜索上搜索：
192 168 8 100
搜一下

3.4.8 自定义错误日志

可以自定义错误日志

```
Syntax: error_log file [level];
Default:
error_log logs/error.log error;
Context: main, http, mail, stream, server, location
level: debug, info, notice, warn, error, crit, alert, emerg
```

范例:

```
[root@centos8 ~]# mkdir /data/nginx/logs
  listen 80;
  server_name www.magedu.org;
  error_page  500 502 503 504 404 /error.html;
  access_log /apps/nginx/logs/magedu-org_access.log;
  error_log /apps/nginx/logs/magedu-org_error.log; #定义错误日志
  location = /error.html {
    root   html;
  }
#重启nginx并访问不存在的页面进行测试并验证是在指定目录生成新的日志文件
```

3.4.9 检测文件是否存在

try_files会按顺序检查文件是否存在，返回第一个找到的文件或文件夹（结尾加斜线表示为文件夹），如果所有文件或文件夹都找不到，会进行一个内部重定向到最后一个参数。只有最后一个参数可以引起一个内部重定向，之前的参数只设置内部URI的指向。最后一个参数是回退URI且必须存在，否则会出现内部500错误。

语法格式

```
Syntax: try_files file ... uri;
try_files file ... =code;
Default:    —
Context:    server, location
```

范例: 如果不存在页面, 就转到default.html页面

```
location  / {
    root /data/nginx/html/pc;
    index index.html;
    try_files $uri  $uri.html $uri/index.html /about/default.html;
    #try_files $uri $uri/index.html $uri.html =489;
}
[root@centos8 ~]# echo "default page" >> /data/nginx/html/pc/about/default.html

#重启nginx并测试，当访问到http://www.magedu.org/about/xx.html等不存在的uri会显示
default.html，如果是自定义的状态码则会显示在返回数据的状态码中

#注释default.html行,启用上面489响应码的那一行,在其生效后再观察结果
location  / {
    root /data/nginx/html/pc;
    index index.html;
    #try_files $uri  $uri.html $uri/index.html /about/default.html;
```

```
    try_files $uri $uri/index.html $uri.html =489;
}

[root@centos8 ~]# curl -I  http://www.magedu.org/about/xx.html
HTTP/1.1 489 #489就是自定义的状态返回码
Server: nginx
Date: Thu, 21 Feb 2019 00:11:40 GMT
Content-Length: 0
Connection: keep-alive
Keep-Alive: timeout=65
```

3.4.10 长连接配置

```
keepalive_timeout timeout [header_timeout];  #设定保持连接超时时长，0表示禁止长连接，默认为75s，通常配置在http字段作为站点全局配置
keepalive_requests number;  #在一次长连接上所允许请求的资源的最大数量，默认为100次,建议适当调大,比如:500
```

范例:

```
keepalive_requests 3;
keepalive_timeout  65 60;
#开启长连接后，返回客户端的会话保持时间为60s，单次长连接累计请求达到指定次数请求或65秒就会被断开，第二个数字60为发送给客户端应答报文头部中显示的超时时间设置为60s；如不设置客户端将不显示超时时间。
Keep-Alive:timeout=60  #浏览器收到的服务器返回的报文

#如果设置为0表示关闭会话保持功能，将如下显示:
Connection:close  #浏览器收到的服务器返回的报文

#使用命令测试:
[root@centos8 ~]# telnet  www.magedu.org 80
Trying 10.0.0.8...
Connected to www.magedu.org.
Escape character is '^]'.
GET / HTTP/1.1
HOST: www.magedu.org

#Response Headers(响应头信息):
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Thu, 24 Sep 2020 04:35:35 GMT
Content-Type: text/html
Content-Length: 7
Last-Modified: Wed, 23 Sep 2020 14:39:21 GMT
Connection: keep-alive
Keep-Alive: timeout=60
ETag: "5c8a6b3a-7"
Accept-Ranges: bytes

#页面内容
pc web
```

3.4.11 作为下载服务器配置

ngx_http_autoindex_module 模块处理以斜杠字符 "/" 结尾的请求，并生成目录列表,可以做为下载服务配置使用

官方文档:

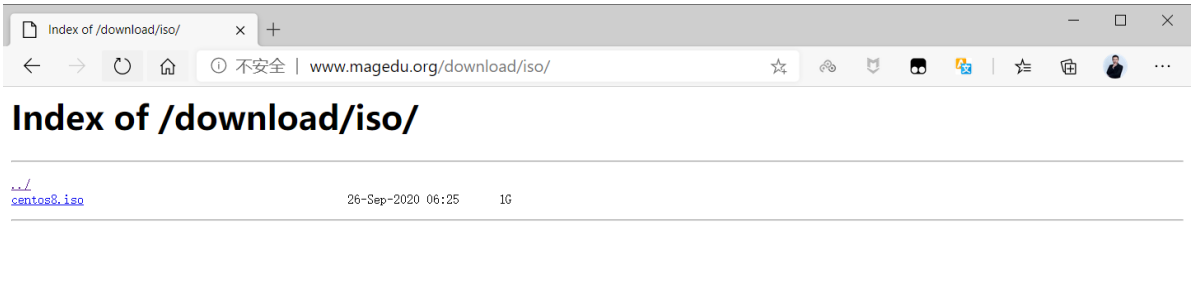
```
http://nginx.org/en/docs/http/ngx_http_autoindex_module.html
```

相关指令:

```
autoindex on | off;#自动文件索引功能，默为off
autoindex_exact_size on | off;  #计算文件确切大小（单位bytes），off 显示大概大小（单位K、M），默认on
autoindex_localtime on | off ; #显示本机时间而非GMT(格林威治)时间，默认off
autoindex_format html | xml | json | jsonp; #显示索引的页面文件风格，默认html
limit_rate rate; #限制响应客户端传输速率(除GET和HEAD以外的所有方法)，单位B/s,即bytes/second，默认值0,表示无限制,此指令由ngx_http_core_module提供
set $limit_rate 4k; #也可以通变量限速,单位B/s,同时设置,此项优级高.Rate limit can also be set in the $limit_rate variable, however, since version 1.17.0, this method is not recommended:
```

范例: 实现下载站点

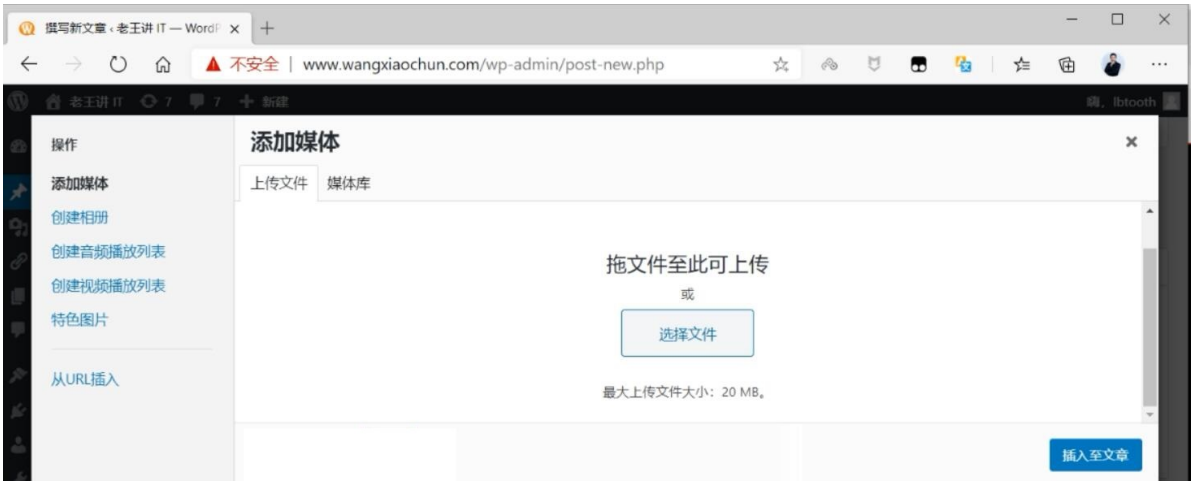
```
#注意:download不需要index.html文件
[root@centos8 ~]# mkdir -p /data/nginx/html/pc/download

[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf
  location /download  {
    autoindex on;    #自动索引功能
    autoindex_exact_size   on; #计算文件确切大小（单位bytes），此为默认值,off只显示大概大小（单位kb、mb、gb）
    autoindex_localtime    on; #on表示显示本机时间而非GMT(格林威治)时间,默为为off显示GMT时间
    limit_rate 1024k;          #限速,默认不限速
    root /data/nginx/html/pc;
  }

[root@centos8 ~]# cp /root/anaconda-ks.cfg  /data/nginx/html/pc/download/
#重启Nginx并访问测试下载页面
```



3.4.12 作为上传服务器

以下指令控制上传数据

```
client_max_body_size 1m; #设置允许客户端上传单个文件的最大值，默认值为1m,上传文件超过此值会出413错误
client_body_buffer_size size;   #用于接收每个客户端请求报文的body部分的缓冲区大小;默认16k;超出此大小时，其将被暂存到磁盘上的由client_body_temp_path指令所定义的位置
client_body_temp_path path [level1 [level2 [level3]]];
#设定存储客户端请求报文的body部分的临时存储路径及子目录结构和数量，目录名为16进制的数字，使用hash之后的值从后往前截取1位、2位、2位作为目录名

[root@centos8 ~]# md5sum  /data/nginx/html/pc/index.html
95f6f65f498c74938064851b1bb 96 3d 4  /data/nginx/html/pc/index.html

1级目录占1位16进制，即2^4=16个目录  0-f
2级目录占2位16进制，即2^8=256个目录 00-ff
3级目录占2位16进制，即2^8=256个目录 00-ff

#配置示例：
client_max_body_size 100m; #如果太大，上传时会出现下图的413错误,注意:如果php上传,还需要修改php.ini的相关配置
client_body_buffer_size 1024k;
client_body_temp_path   /apps/nginx/client_body_temp/ 1 2 2; #上传时,Nginx会自动创建相关目录
```




```
#上传超过nginx指定client_max_body_size值会出413错误
[root@wang-liyun-pc ~]# tail /apps/nginx/logs/nginx.access.log
125.41.184.117 - - [27/Sep/2020:00:09:00 +0800] "POST /wp-admin/async-upload.php
HTTP/1.1" 413 578 "http://www.wangxiaochun.com/wp-admin/post-new.php"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/85.0.4183.121 Safari/537.36 Edg/85.0.564.63" "-"
```

413 Request Entity Too Large

nginx

```
#上传文件后,会自动生成相关目录
[root@wang-liyun-pc ~]# tree /apps/nginx/client_body_temp/
/apps/nginx/client_body_temp/
├── 5
│   └── 00
│       └── 00
└── 6
    └── 00
        └── 00
```

3.4.13 其他配置

```
keepalive_disable none | browser ...;
#对哪种浏览器禁用长连接
```

```
limit_except method ... { ... }，仅用于location
#禁止客户端使用除了指定的请求方法之外的其它方法,如果使用会出现403错误
method:GET, HEAD, POST, PUT, DELETE，MKCOL, COPY, MOVE, OPTIONS, PROPFIND,
PROPPATCH, LOCK, UNLOCK, PATCH
limit_except GET {
    allow 192.168.0.0/24;
    allow 10.0.0.1;
    deny  all;
}
```

```
#除了GET和HEAD 之外其它方法仅允许192.168.1.0/24网段主机使用
[root@centos8 ~]# mkdir /data/nginx/html/pc/upload
[root@centos8 ~]# echo "upload" > /data/nginx/html/pc/upload/index.html
[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf
location  /upload  {
        root /data/nginx/html/pc;
        index index.html;
        limit_except GET {
            allow  10.0.0.6;
            deny all;
        }
}


#重启Nginx并进行测试上传文件
[root@centos8 ~]# systemctl  restart  nginx
[root@centos8 ~]#
[root@centos6 ~]# curl  -XPUT  /etc/issue http://www.magedu.org/about
curl: (3) <url> malformed
<html>
<head><title>405 Not Allowed</title></head> #Nginx已经允许,但是程序未支持上传功能
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx</center>
</body>
</html>
[root@centos8 ~]# curl  -XPUT  /etc/issue http://www.magedu.org/upload
curl: (3) <url> malformed
<html>
<head><title>403 Forbidden</title></head> #Nginx拒绝上传
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

```
aio on | off  #是否启用asynchronous file I/O(AIO)功能，需要编译开启 --with-file-aio

#linux 2.6以上内核提供以下几个系统调用来支持aio：
1、SYS_io_setup：建立aio 的context
2、SYS_io_submit：提交I/O操作请求
3、SYS_io_getevents：获取已完成的I/O事件
4、SYS_io_cancel：取消I/O操作请求
5、SYS_io_destroy：毁销aio的context
```

```
directio size | off; #操作完全和aio相反，aio是读取文件而directio是写文件到磁盘，启用直接I/O，默认为关闭，当文件大于等于给定大小时，例如:directio 4m，同步（直接）写磁盘，而非写缓存。
```

```
open_file_cache off;  #是否缓存打开过的文件信息
open_file_cache max=N [inactive=time];

#nginx可以缓存以下三种信息：
(1) 文件元数据：文件的描述符、文件大小和最近一次的修改时间
(2) 打开的目录结构
(3) 没有找到的或者没有权限访问的文件的相关信息
max=N：#可缓存的缓存项上限数量;达到上限后会使用LRU(Least recently used，最近最少使用)算法实现管理
inactive=time：#缓存项的非活动时长，在此处指定的时长内未被命中的或命中的次数少于open_file_cache_min_uses指令所指定的次数的缓存项即为非活动项，将被删除

open_file_cache_valid time; #缓存项有效性的检查验证频率，默认值为60s
open_file_cache_errors on | off; #是否缓存查找时发生错误的文件一类的信息,默认值为off
open_file_cache_min_uses number; #open_file_cache指令的inactive参数指定的时长内，至少被命中此处指定的次数方可被归类为活动项,默认值为1
```

范例:

```
open_file_cache max=10000 inactive=60s; #最大缓存10000个文件，非活动数据超时时长60s
open_file_cache_valid    60s;  #每间隔60s检查一下缓存数据有效性
open_file_cache_min_uses 5; #60秒内至少被命中访问5次才被标记为活动数据
open_file_cache_errors   on; #缓存错误信息
```

4 Nginx 高级配置

4.1 Nginx 状态页

基于nginx 模块 ngx_http_stub_status_module 实现，在编译安装nginx的时候需要添加编译参数 --with-http_stub_status_module，否则配置完成之后监测会是提示语法错误

注意: 状态页显示的是整个服务器的状态,而非虚拟主机的状态

```
#配置示例：
location  /nginx_status {
    stub_status;
    auth_basic            "auth login";
    auth_basic_user_file  /apps/nginx/conf/.htpasswd;
    allow 192.168.0.0/16;
    allow 127.0.0.1;
    deny all;
  }

#状态页用于输出nginx的基本状态信息：
#输出信息示例：
Active connections: 291
server accepts handled requests
    16630948 16630948 31070465
    上面三个数字分别对应accepts,handled,requests三个值
Reading: 6 Writing: 179 Waiting: 106

Active connections：  #当前处于活动状态的客户端连接数，包括连接等待空闲连接数
=reading+writing+waiting
accepts：#统计总值，Nginx自启动后已经接受的客户端请求连接的总数。
```

```
handled：#统计总值，Nginx自启动后已经处理完成的客户端请求连接总数，通常等于accepts，除非有因worker_connections限制等被拒绝的连接
requests：#统计总值，Nginx自启动后客户端发来的总的请求数。
Reading：#当前状态，正在读取客户端请求报文首部的连接的连接数,数值越大,说明排队现象严重,性能不足
Writing：#当前状态，正在向客户端发送响应报文过程中的连接数,数值越大,说明访问量很大
Waiting：#当前状态，正在等待客户端发出请求的空闲连接数，开启 keep-alive的情况下,这个值等于active - (reading+writing)
```

范例:分析网站当前访问量

```
[root@centos7 ~]#curl http://wang:123456@www.wangxiaochun.com/nginx_status 2>/dev/null |awk  '/Reading/{print $2,$4,$6}'
0 1 15
```

4.2 Nginx 第三方模块

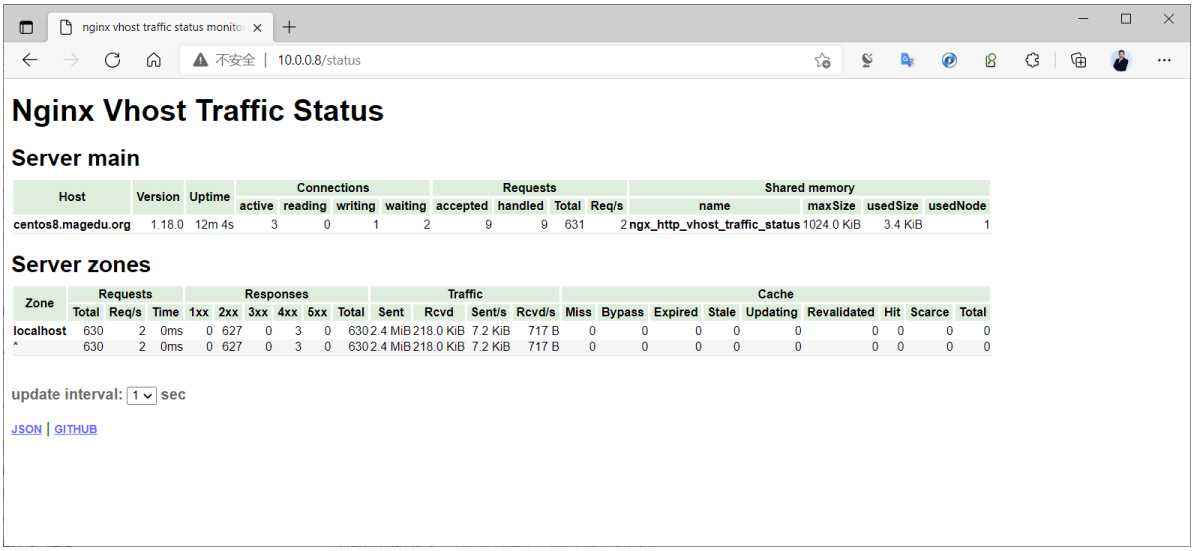
第三模块是对nginx 的功能扩展，第三方模块需要在编译安装Nginx 的时候使用参数--add-module=PATH指定路径添加，有的模块是由公司的开发人员针对业务需求定制开发的，有的模块是开源爱好者开发好之后上传到github进行开源的模块，nginx的第三方模块需要从源码重新编译进行支持

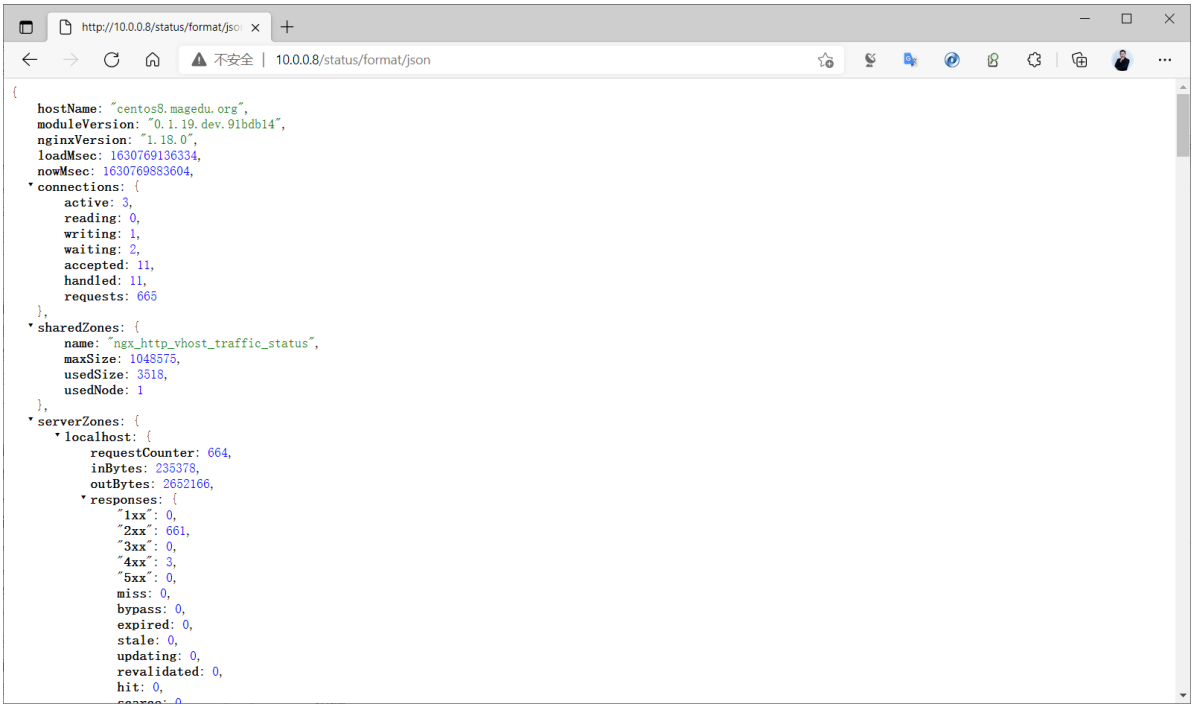
4.2.1 nginx-module-vts 模块实现流量监控

```
https://github.com/vozlt/nginx-module-vts
```

范例:

```
[root@centos8 ~]#cd /usr/local/src
[root@centos8 src]#git clone git://github.com/vozlt/nginx-module-vts.git
[root@centos8 src]#cd nginx-1.18.0/
[root@centos8 nginx-1.18.0]#./configure --prefix=/apps/nginx --add-module=/usr/local/src/nginx-module-vts
[root@centos8 nginx-1.18.0]#make && make install
[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf
http {
    ......
    vhost_traffic_status_zone;
    ......
    server {
        ......
        location /status {
            vhost_traffic_status_display;
            vhost_traffic_status_display_format html;
        }
        ......
    }
}
[root@centos8 ~]#systemctl restart nginx
#浏览器访问:http://<nginx_ip>/status 可以看到下面显示
```





4.2.2 echo 模块实现信息显示

开源的echo模块可以用来打印信息,变量等

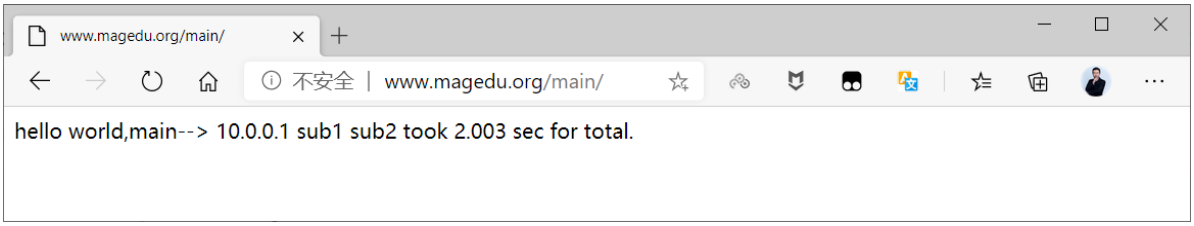
```
https://github.com/openresty/echo-nginx-module
```

范例:

```
[root@centos8 ~]# systemctl  stop nginx
[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf
 location /main {
     index index.html;
     default_type text/html;
     echo "hello world,main-->";
     echo $remote_addr ;
     echo_reset_timer;   #将计时器开始时间重置为当前时间
     echo_location /sub1;
     echo_location /sub2;
     echo "took $echo_timer_elapsed sec for total.";
 }
```

```
 location /sub1 {
     echo_sleep 1;
     echo sub1;
 }
 location /sub2 {
     echo_sleep 1;
     echo sub2;
 }

[root@centos8 ~]# /apps/nginx/sbin/nginx -t
nginx: [emerg] unknown directive "echo_reset_timer" in
/apps/nginx/conf/conf.d/pc.conf:86
nginx: configuration file /apps/nginx/conf/nginx.conf test failed

#解决以上报错问题
[root@centos8 ~]# cd /usr/local/src
[root@centos8 src]# yum install git -y

#github网站国内访问不稳定,可能无法下载
[root@centos8 src]# git clone https://github.com/openresty/echo-nginx-module.git
#如果上面链接无法下载,可以用下面链接
[root@centos8 src]# git clone https://github.com.cnpmjs.org/openresty/echo-
nginx-module.git

[root@centos8 src]# cd nginx-1.18.0/
[root@centos8 src]# ./configure  \
--prefix=/apps/nginx \
--user=nginx --group=nginx \
--with-http_ssl_module \
--with-http_v2_module \
--with-http_realip_module \
--with-http_stub_status_module \
--with-http_gzip_static_module \
--with-pcre \
--with-stream \
--with-stream_ssl_module \
--with-stream_realip_module \
--with-http_perl_module \
--add-module=/usr/local/src/echo-nginx-module  #指定模块源代码路径

[root@centos8 src]# make && make install

#确认语法检测通过
[root@centos8 ~]# /apps/nginx/sbin/nginx -t
nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /apps/nginx/conf/nginx.conf test is successful

#重启nginx访问测试
[root@centos8 ~]# systemctl  restart nginx
[root@centos8 ~]#nginx  -V
nginx version: wanginx/1.68.9
built by gcc 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)
built with OpenSSL 1.1.1c FIPS  28 May 2019
TLS SNI support enabled
```

```
configure arguments: --prefix=/apps/nginx --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream
--with-stream_ssl_module --with-stream_realip_module --add-
module=/usr/local/src/echo-nginx-module

#测试查看结果
[root@centos7 ~]#curl  http://www.magedu.org/main
hello world,main-->
10.0.0.7
sub1
sub2
took 2.003 sec for total.
.
```



4.3 Nginx 变量使用

nginx的变量可以在配置文件中引用，作为功能判断或者日志等场景使用

变量可以分为内置变量和自定义变量

内置变量是由nginx模块自带，通过变量可以获取到众多的与客户端访问相关的值。

4.3.1 内置变量

官方文档

```
http://nginx.org/en/docs/varindex.html
```

常用内置变量

```
$remote_addr;
#存放了客户端的地址，注意是客户端的公网IP

$proxy_add_x_forwarded_for
#此变量表示将客户端IP追加请求报文中X-Forwarded-For首部字段,多个IP之间用逗号分隔,如果请求中没有X-Forwarded-For,就使用$remote_addr
the “X-Forwarded-For” client request header field with the $remote_addr variable appended to it, separated by a comma. If the “X-Forwarded-For” field is not present in the client request header, the $proxy_add_x_forwarded_for variable is equal to the $remote_addr variable.

$args;
#变量中存放了URL中的所有参数，例如:http://www.magedu.org/main/index.do?id=20190221&partner=search
#返回结果为: id=20190221&partner=search

$is_args
```

```
#如果有参数为? 否则为空
“?” if a request line has arguments, or an empty string otherwise

$document_root;
#保存了针对当前资源的请求的系统根目录,例如:/apps/nginx/html。

$document_uri;
#保存了当前请求中不包含参数的URI，注意是不包含请求的指令，比
如:http://www.magedu.org/main/index.do?id=20190221&partner=search会被定义
为/main/index.do
#返回结果为:/main/index.do

$host;
#存放了请求的host名称


limit_rate 10240;
echo $limit_rate;
#如果nginx服务器使用limit_rate配置了显示网络速率，则会显示，如果没有设置， 则显示0

$remote_port;
#客户端请求Nginx服务器时随机打开的端口，这是每个客户端自己的端口

$remote_user;
#已经经过Auth Basic Module验证的用户名

$request_body_file;
#做反向代理时发给后端服务器的本地资源的名称

$request_method;
#请求资源的方式，GET/PUT/DELETE等

$request_filename;
#当前请求的资源文件的磁盘路径，由root或alias指令与URI请求生成的文件绝对路径，
如:/apps/nginx/html/main/index.html

$request_uri;
#包含请求参数的原始URI，不包含主机名，相当于:$document_uri?$args,例如：/main/index.do?
id=20190221&partner=search

$scheme;
#请求的协议，例如:http，https,ftp等

$server_protocol;
#保存了客户端请求资源使用的协议的版本，例如:HTTP/1.0，HTTP/1.1，HTTP/2.0等

$server_addr;
#保存了服务器的IP地址

$server_name;
#请求的服务器的主机名

$server_port;
#请求的服务器的端口号

$http_user_agent;
#客户端浏览器的详细信息
```

```
$http_cookie;
#客户端的所有cookie信息

$cookie_<name>
#name为任意请求报文首部字部cookie的key名

$http_<name>
#name为任意请求报文首部字段,表示记录请求报文的首部字段，ame的对应的首部字段名需要为小写，如果有横线需要替换为下划线
arbitrary request header field; the last part of a variable name is the field name converted to lower case with dashes replaced by underscores #用下划线代替横线

#示例:
echo $http_user_agent;
echo $http_host;

$sent_http_<name>
#name为响应报文的首部字段，name的对应的首部字段名需要为小写，如果有横线需要替换为下划线,此变量有问题
echo $sent_http_server;

$arg_<name>
#此变量存放了URL中的指定参数，name为请求url中指定的参数
echo $arg_id;
```

范例:

```
[root@centos8 ~]#vi /apps/nginx/conf/conf.d/pc.conf
  location /main {
     index index.html;
     default_type text/html;
     echo "hello world,main-->";
     echo $remote_addr ;
     echo $args ;
     echo $document_root;
     echo $document_uri;
     echo $host;
     echo $http_user_agent;
     echo $http_cookie;
     echo $request_filename;
     echo $scheme;
     echo $scheme://$host$document_uri?$args;
  }

[root@centos6 ~]#curl -b title=ceo 'http://www.magedu.org/main/index.do?id=20190221&partner=search'
hello world,main-->
10.0.0.6
id=20190221&partner=search
/apps/nginx/html
/main/index.do
www.magedu.org
curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.27.1 zlib/1.2.3 libidn/1.18 libssh2/1.4.2
title=ceo
/apps/nginx/html/main/index.do
http
```

```
http://www.magedu.org/main/index.do?id=20190221&partner=search
```

范例:

```
[root@centos8 conf.d]#vim www.conf
.....
    location /echo {
        echo $request;
        echo $proxy_add_x_forwarded_for;
        echo $args;
        echo $document_uri;
        echo $request_uri;
        echo $document_root;
        echo $host;
        echo $request_method;
        echo $request_filename;
        echo $scheme;
        set  $test $http_host;
        echo $test;
        echo $http_User_Agent;
        echo $http_cookie;
        echo $cookie_key1;
    }
}

[root@centos7 ~]#curl -b 'key1=v1;key2=v2'
"http://www.magedu.org/echo/index.html?id=123456&partner=search"
GET /echo/index.html?id=123456&partner=search HTTP/1.1
10.0.0.7
id=123456&partner=search
/echo/index.html
/echo/index.html?id=123456&partner=search
/data/nginx/html/pc
www.magedu.org
GET
/data/nginx/html/pc/echo/index.html
http
www.magedu.org
curl/7.29.0
key1=v1;key2=v2
v1
```

4.3.2 自定义变量

假如需要自定义变量名称和值，使用指令set $variable value;

语法格式:

```
Syntax: set $variable value;
Default:    —
Context:    server, location, if
```

范例:

```
set $name magedu;
echo $name;
set $my_port $server_port;
echo $my_port;
echo "$server_name:$server_port";

#输出信息如下
[root@centos6 ~]#curl www.magedu.org/main
magedu
80
www.magedu.org:80
```

4.4 Nginx 自定义访问日志

访问日志是记录客户端即用户的具体请求内容信息，而在全局配置模块中的error_log是记录nginx服务器运行时的日志保存路径和记录日志的level，因此两者是不同的，而且Nginx的错误日志一般只有一个，但是访问日志可以在不同server中定义多个，定义一个日志需要使用access_log指定日志的保存路径，使用log_format指定日志的格式，格式中定义要保存的具体日志内容。

访问日志由 ngx_http_log_module 模块实现

官方帮助文档:

```
http://nginx.org/en/docs/http/ngx_http_log_module.html
```

语法格式

```
Syntax: access_log path [format [buffer=size] [gzip[=level]] [flush=time]
[if=condition]];
access_log off; #关闭访问日志
Default:
access_log logs/access.log combined;
Context:    http, server, location, if in location, limit_except
```

4.4.1 自定义默认格式日志

如果是要保留日志的源格式，只是添加相应的日志内容，则配置如下:

```
#注意:此指令只支持http块,不支持server块
log_format  nginx_format1  '$remote_addr - $remote_user [$time_local] "$request"
'
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"'
                    '$server_name:$server_port';

#注意:此指令一定要在放在log_format命令后
access_log  logs/access.log  nginx_format1;

#重启nginx并访问测试日志格式
==> /apps/nginx/logs/access.log <==
10.0.0.1 - - [22/Feb/2019:08:44:14 +0800] "GET /favicon.ico HTTP/1.1" 404 162 "-
" "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:65.0) Gecko/2
0100101 Firefox/65.0" "-"www.magedu.org:80
```

4.4.2 自定义json格式日志

Nginx 的默认访问日志记录内容相对比较单一，默认的格式也不方便后期做日志统计分析，生产环境中通常将nginx日志转换为json日志，然后配合使用ELK做日志收集,统计和分析。

```
    log_format access_json '{"@timestamp":"$time_iso8601",'
        '"host":"$server_addr",'
        '"clientip":"$remote_addr",'
        '"size":$body_bytes_sent,'
        '"responsetime":$request_time,'      #总的处理时间
        '"upstreamtime":"$upstream_response_time",'
        '"upstreamhost":"$upstream_addr",'    #后端应用服务器处理时间
        '"http_host":"$host",'
        '"uri":"$uri",'
        '"xff":"$http_x_forwarded_for",'
        '"referer":"$http_referer",'
        '"tcp_xff":"$proxy_protocol_addr",'
        '"http_user_agent":"$http_user_agent",'
        '"status":"$status"}';
     access_log  /apps/nginx/logs/access_json.log  access_json;

#重启Nginx并访问测试日志格式,参考链接:http://json.cn/
{"@timestamp":"2019-02-
22T08:55:32+08:00","host":"10.0.0.8","clientip":"10.0.0.1","size":162,"responset
ime":0.000,"upstreamtime":"-","upstreamhost":"-
","http_host":"www.magedu.org","uri":"/favicon.ico","xff":"-","referer":"-
","tcp_xff":"","http_user_agent":"Mozilla/5.0 (Windows NT 6.1; Win64; x64;
rv:65.0) Gecko/20100101 Firefox/65.0","status":"404"}
```

4.4.3 json 格式的日志访问统计

```
#python3
[root@centos8 ~]#dnf -y install python3
[root@centos8 ~]#cat log.py
#!/usr/bin/env python3
#coding:utf-8

status_200= []
status_404= []
with open("access_json.log") as f:
    for line in f.readlines():
        line = eval(line)
        if line.get("status") == "200":
            status_200.append(line.get)
        elif line.get("status") == "404":
            status_404.append(line.get)
        else:
            print("状态码 ERROR")
        print((line.get("clientip")))
f.close()

print("状态码200的有--:",len(status_200))
print("状态码404的有--:",len(status_404))

#python2
[root@centos8 ~]#dnf -y install python2
```

```
[root@centos8 ~]#cat log.py
#!/usr/bin/env python
#coding:utf-8

status_200= []
status_404= []
with open("access_json.log") as f:
    for line in f.readlines():
        line = eval(line)
        if line.get("status") == "200":
            status_200.append(line.get)
        elif line.get("status") == "404":
            status_404.append(line.get)
        else:
            print("状态码 ERROR")
        print(line.get("clientip"))
f.close()

print "状态码200的有--:",len(status_200)
print "状态码404的有--:",len(status_404)

#保存日志文件到指定路径并进测试:
[root@centos7 ~]# python nginx_json.py
状态码200的有--: 1910
状态码404的有--: 13


#转换python2语法到python3
[root@centos8 ~]#pip3 install 2to3
[root@centos8 ~]#2to3  -w  log.py
```

4.5 Nginx 压缩功能

Nginx支持对指定类型的文件进行压缩然后再传输给客户端，而且压缩还可以设置压缩比例，压缩后的文件大小将比源文件显著变小，这样有助于降低出口带宽的利用率，降低企业的IT支出，不过会占用相应的CPU资源。

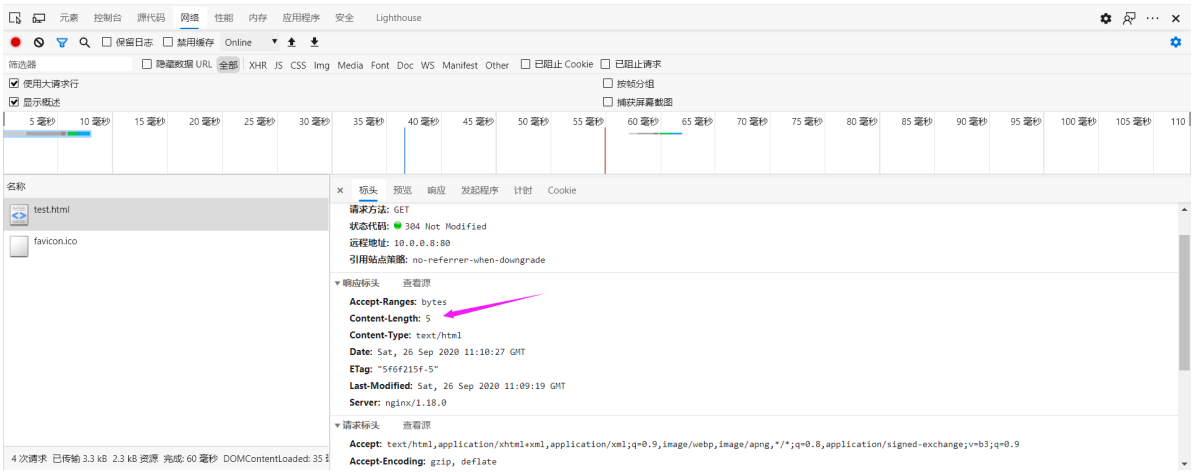
Nginx对文件的压缩功能是依赖于模块 ngx_http_gzip_module,默认是内置模块

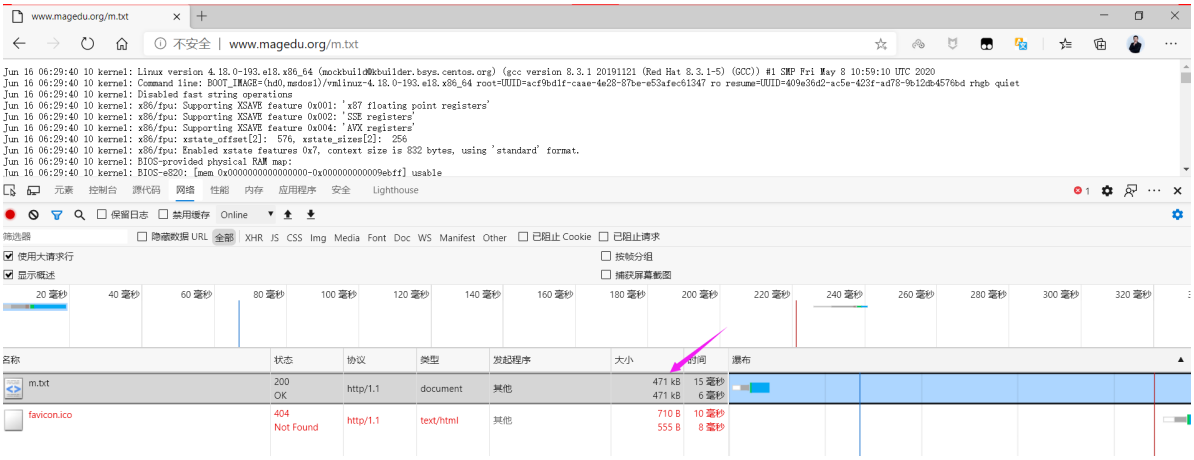
官方文档： https://nginx.org/en/docs/http/ngx_http_gzip_module.html

配置指令如下：

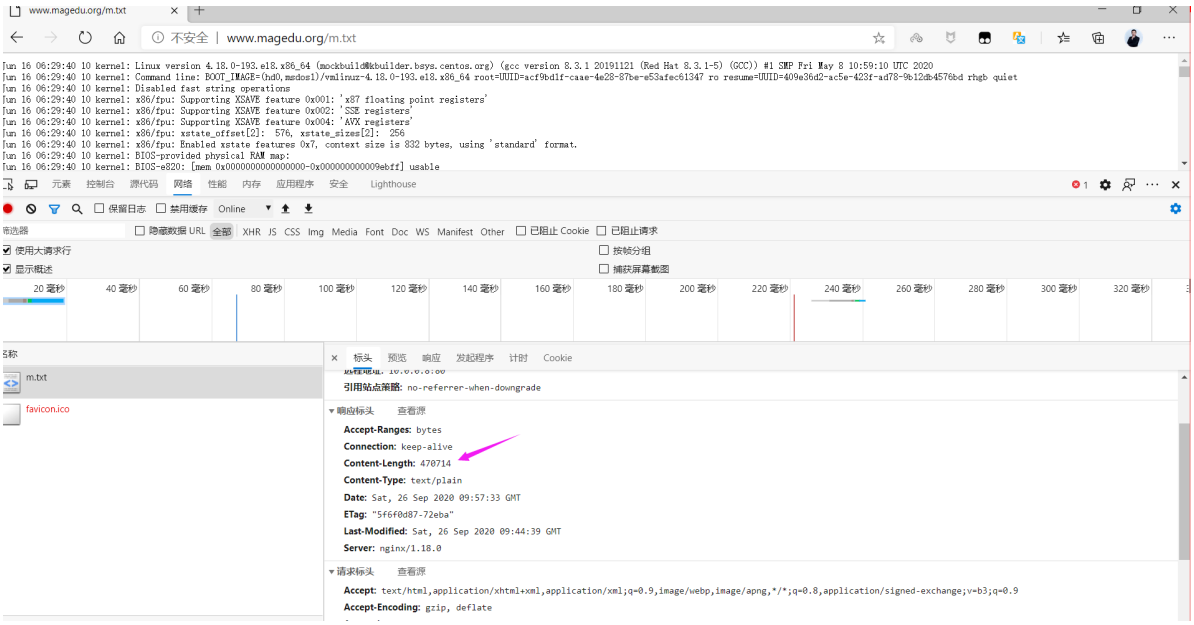
```
#启用或禁用gzip压缩，默认关闭
gzip on | off;

#压缩比由低到高从1到9，默认为1
gzip_comp_level level;

#禁用IE6 gzip功能
gzip_disable "MSIE [1-6]\.";

#gzip压缩的最小文件，小于设置值的文件将不会压缩
gzip_min_length 1k;

#启用压缩功能时，协议的最小版本，默认HTTP/1.1
gzip_http_version 1.0 | 1.1;
```

```
#指定Nginx服务需要向服务器申请的缓存空间的个数和大小,平台不同,默认:32 4k或者16 8k;
gzip_buffers number size;

#指明仅对哪些类型的资源执行压缩操作;默认为gzip_types text/html，不用显示指定，否则出错
gzip_types mime-type ...;

#如果启用压缩，是否在响应报文首部插入“Vary: Accept-Encoding”,一般建议打开
gzip_vary on | off;

#预压缩，即直接从磁盘找到对应文件的gz后缀的式的压缩文件返回给用户，无需消耗服务器CPU
#注意：来自于ngx_http_gzip_static_module模块
gzip_static on | off;

#重启nginx并进行访问测试压缩功能
[root@centos8 ~]# cp /apps/nginx/logs/access.log /data/nginx/html/pc/m.txt
[root@centos8 ~]# echo "test" > /data/nginx/html/pc/test.html #小于1k的文件测试是否
会压缩
[root@centos8 ~]# vim /apps/nginx/conf/nginx.conf
gzip  on;
gzip_comp_level 5;
gzip_min_length 1k;
gzip_types text/plain application/javascript application/x-javascript text/css
application/xml text/javascript application/x-httpd-php  image/gif image/png;

gzip_vary on;

#重启Nginx并访问测试：
[root@centos8 ~]# curl  --head  --compressed  http://www.magedu.org/test.html
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 22 Feb 2019 01:52:23 GMT
Content-Type: text/html
Last-Modified: Thu, 21 Feb 2019 10:31:18 GMT
Connection: keep-alive
Keep-Alive: timeout=65
Vary: Accept-Encoding
ETag: W/"5c6e7df6-171109"
Content-Encoding: gzip #压缩传输

#验证不压缩访问的文件大小：
```

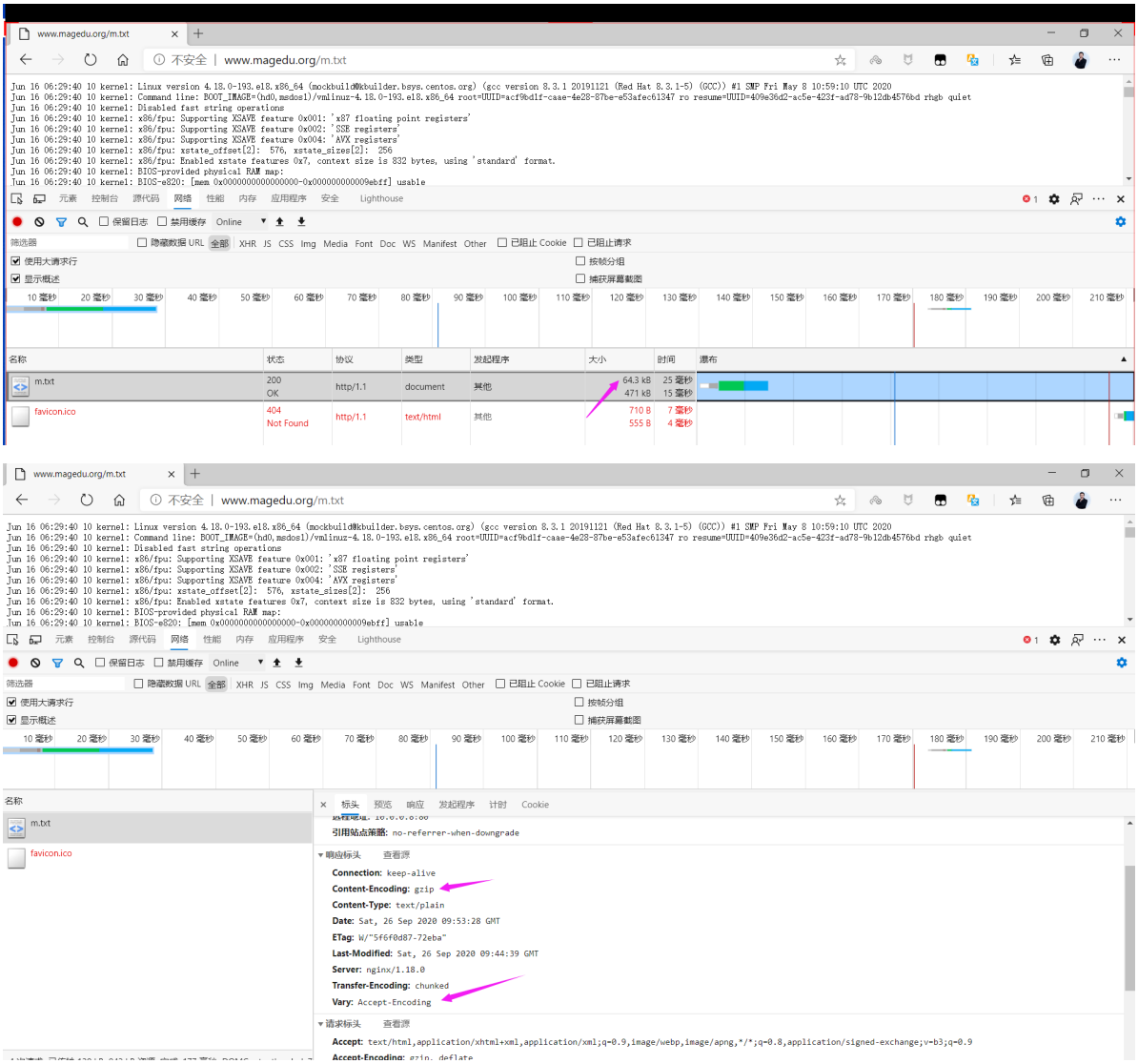




验证不压缩后访问的文件大小：

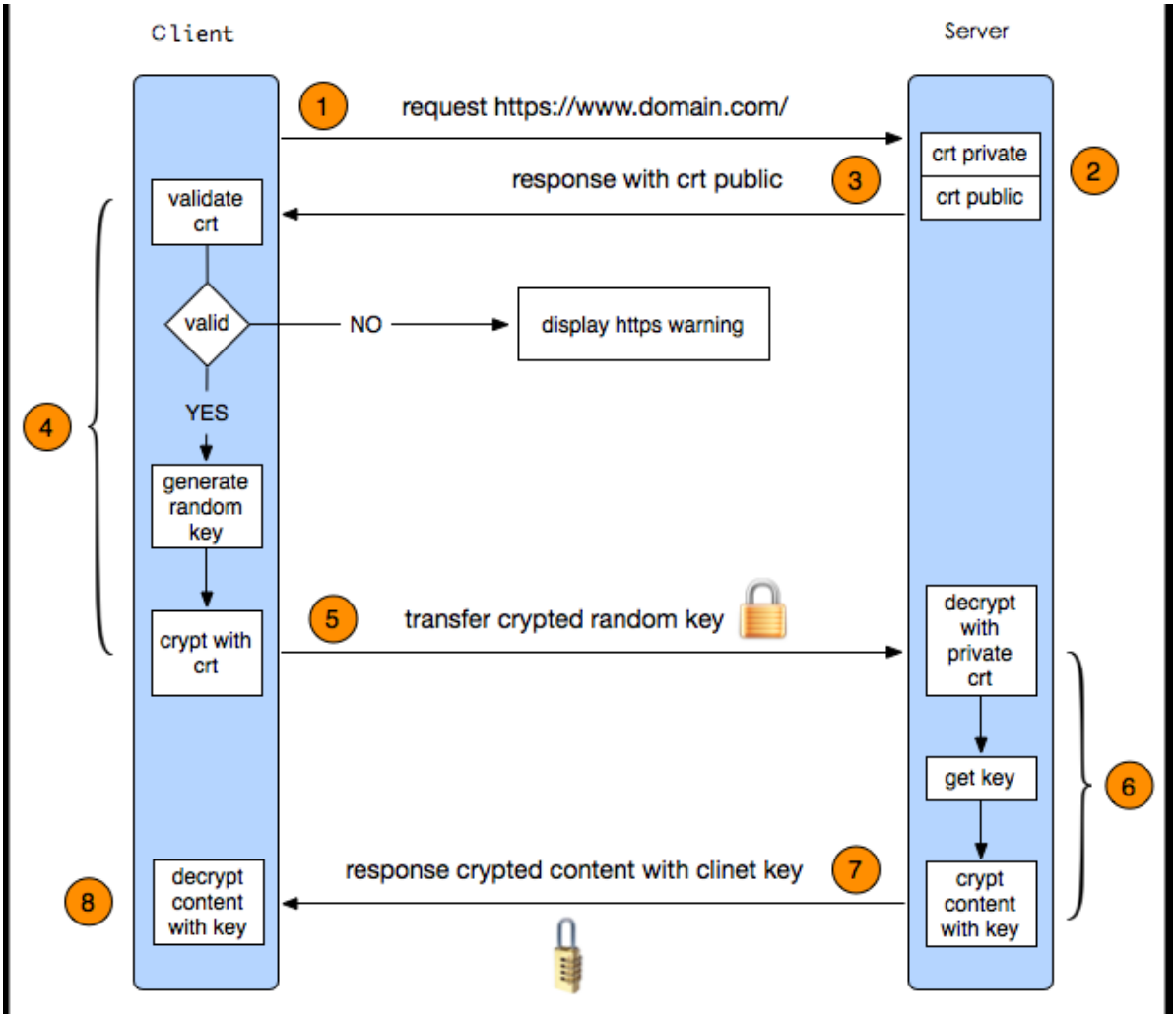


验证压缩后访问的文件大小：



4.6 https 功能

Web网站的登录页面通常都会使用https加密传输的，加密数据以保障数据的安全，HTTPS能够加密信息，以免敏感信息被第三方获取，所以很多银行网站或电子邮箱等等安全级别较高的服务都会采用HTTPS协议，HTTPS其实是有两部分组成：HTTP + SSL / TLS，也就是在HTTP上又加了一层处理加密信息的模块。服务端和客户端的信息传输都会通过TLS进行加密，所以传输的数据都是加密后的数据。



https 实现过程如下：

1.客户端发起HTTPS请求：

客户端访问某个web端的https地址，一般都是443端口

2.服务端的配置：

采用https协议的服务器必须要有一套证书，可以通过一些组织申请，也可以自己制作，目前国内很多网站都自己做的，当你访问一个网站的时候提示证书不可信任就表示证书是自己做的，证书就是一个公钥和私钥匙，就像一把锁和钥匙，正常情况下只有你的钥匙可以打开你的锁，你可以把这个送给别人让他锁住一个箱子，里面放满了钱或秘密，别人不知道里面放了什么而且别人也打不开，只有你的钥匙是可以打开的。

3.传送证书：

服务端给客户端传递证书，其实就是公钥，里面包含了很多信息，例如证书得到颁发机构、过期时间等等。

4.客户端解析证书：

这部分工作是有客户端完成的，首先回验证公钥的有效性，比如颁发机构、过期时间等等，如果发现异常则会弹出一个警告框提示证书可能存在问题，如果证书没有问题就生成一个随机值，然后用证书对该随机值进行加密，就像2步骤所说把随机值锁起来，不让别人看到。

5.传送4步骤的加密数据：

就是将用证书加密后的随机值传递给服务器，目的就是为了让服务器得到这个随机值，以后客户端和服务端的通信就可以通过这个随机值进行加密解密了。

6.服务端解密信息：

服务端用私钥解密5步骤加密后的随机值之后，得到了客户端传过来的随机值(私钥)，然后把内容通过该值进行对称加密，对称加密就是将信息和私钥通过算法混合在一起，这样除非你知道私钥，不然是无法获取其内部的内容，而正好客户端和服务端都知道这个私钥，所以只要机密算法够复杂就可以保证数据的安全性。

7.传输加密后的信息：

服务端将用私钥加密后的数据传递给客户端，在客户端可以被还原出原数据内容。

8.客户端解密信息：

客户端用之前生成的私钥获解密服务端传递过来的数据，由于数据一直是加密的，因此即使第三方获取到数据也无法知道其详细内容。

4.6.1 https 配置参数

nginx 的https 功能基于模块ngx_http_ssl_module实现，因此如果是编译安装的nginx要使用参数ngx_http_ssl_module开启ssl功能，但是作为nginx的核心功能，yum安装的nginx默认就是开启的，编译安装的nginx需要指定编译参数--with-http_ssl_module开启

官方文档：

```
https://nginx.org/en/docs/http/ngx_http_ssl_module.html
```

配置参数如下：

```
ssl on | off;
#为指定的虚拟主机配置是否启用ssl功能，此功能在1.15.0废弃，使用listen [ssl]替代
listen 443 ssl;

ssl_certificate /path/to/file;
#指向包含当前虚拟主机和CA的两个证书信息的文件，一般是crt文件

ssl_certificate_key /path/to/file;
#当前虚拟主机使用的私钥文件，一般是key文件

ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2];
#支持ssl协议版本，早期为ssl现在是TLS，默认为后三个

ssl_session_cache off | none | [builtin[:size]] [shared:name:size];
#配置ssl缓存
    off:  #关闭缓存
    none:  #通知客户端支持ssl session cache，但实际不支持
    builtin[:size]:#使用OpenSSL内建缓存，为每worker进程私有
    [shared:name:size]:#在各worker之间使用一个共享的缓存，需要定义一个缓存名称和缓存空间大小，一兆可以存储4000个会话信息，多个虚拟主机可以使用相同的缓存名称

ssl_session_timeout time;
#客户端连接可以复用ssl session cache中缓存的有效时长，默认5m
```

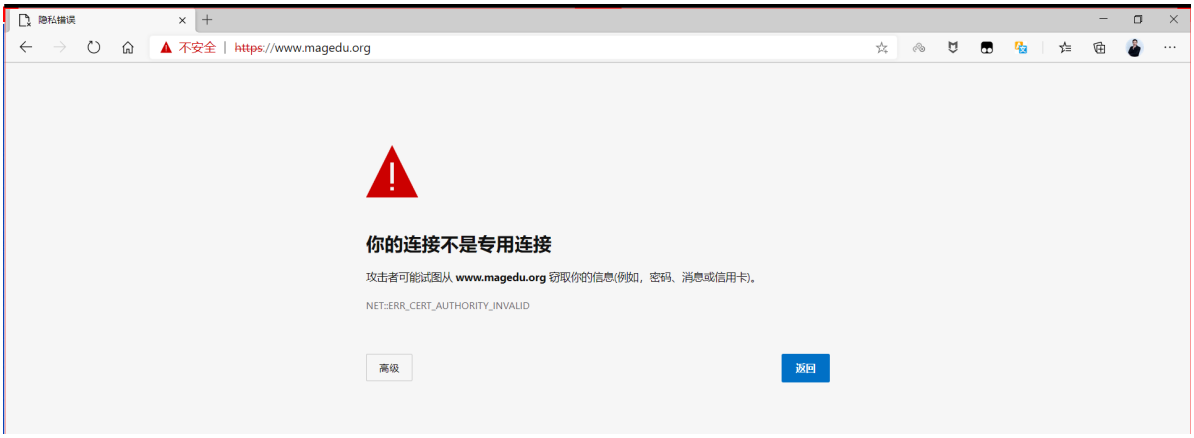
4.6.2 自签名证书

```
#自签名CA证书
[root@centos8 ~]# cd /apps/nginx/
[root@centos8 nginx]# mkdir  certs
[root@centos8 nginx]# cd certs/
[root@centos8 nginx]#  openssl req  -newkey rsa:4096 -nodes -sha256 -keyout ca.key  -x509 -days 3650 -out ca.crt  #自签名CA证书
Generating a 4096 bit RSA private key
..................++
.....
Country Name (2 letter code) [XX]:CN #国家代码
State or Province Name (full name) []:BeiJing   #省份
Locality Name (eg, city) [Default City]:Beijing #城市名称
Organization Name (eg, company) [Default Company Ltd]:magedu.Ltd #公司名称
Organizational Unit Name (eg, section) []:magedu #部门
Common Name (eg, your name or your server's hostname) []:ca.magedu.org #通用名称
Email Address []: #邮箱
[root@centos8 certs]# ll ca.crt
```

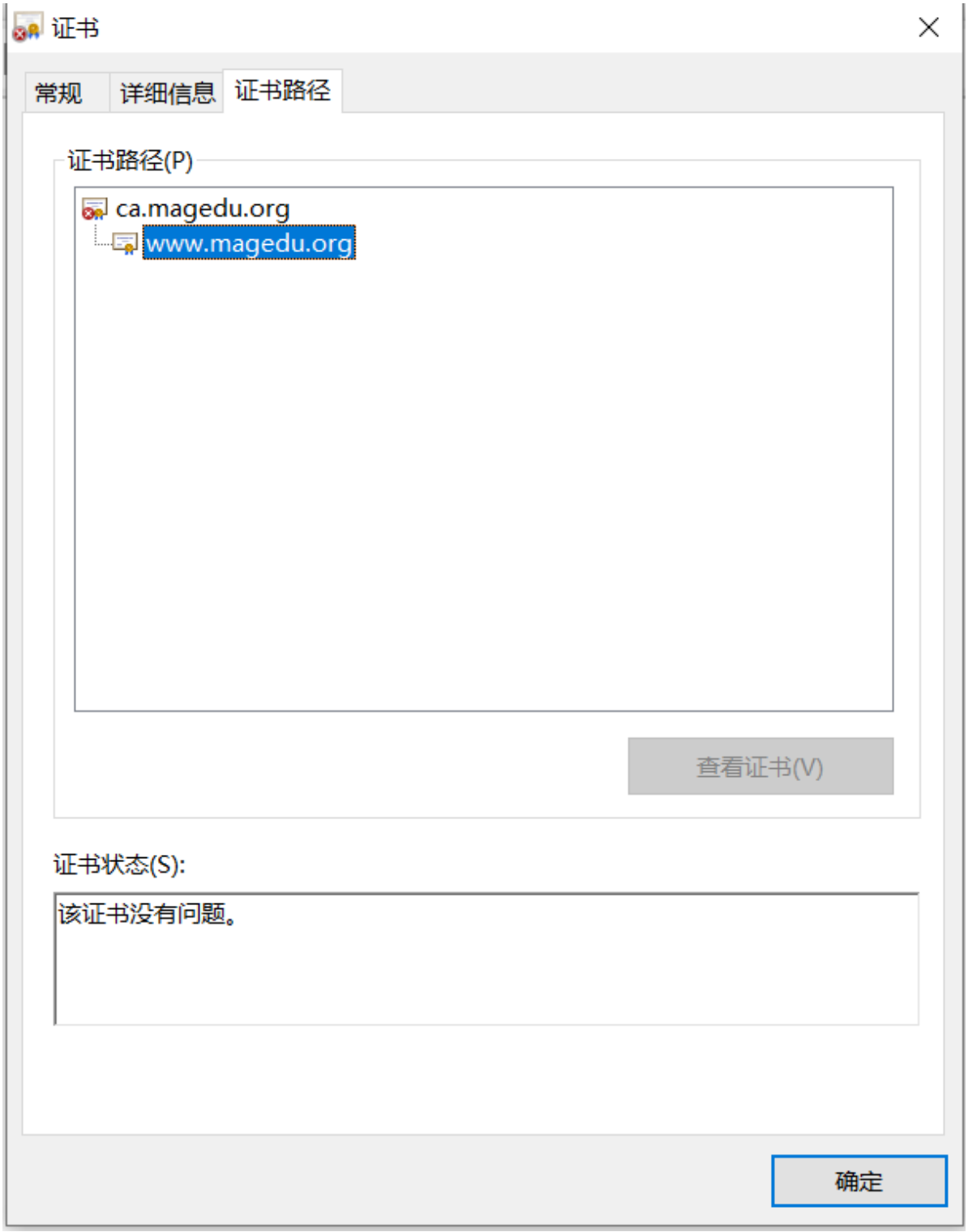
```
-rw-r--r-- 1 root root 2118 Feb 22 12:10 ca.crt

#自制key和csr文件
[root@centos8 certs]# openssl req  -newkey rsa:4096 -nodes -sha256 -keyout
www.magedu.org.key     -out  www.magedu.org.csr
Generating a 4096 bit RSA private key
.....................................................................++
......
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:BeiJing
Locality Name (eg, city) [Default City]:BeiJing
Organization Name (eg, company) [Default Company Ltd]:magedu.org
Organizational Unit Name (eg, section) []:magedu.org
Common Name (eg, your name or your server's hostname) []:www.magedu.org
Email Address []:2973707860@qq.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@centos8 certs]# ll
total 16
-rw-r--r-- 1 root root 2118 Feb 22 12:10 ca.crt
-rw-r--r-- 1 root root 3272 Feb 22 12:10 ca.key
-rw-r--r-- 1 root root 1760 Feb 22 12:18 www.magedu.org.csr
-rw-r--r-- 1 root root 3272 Feb 22 12:18 www.magedu.org.key

#签发证书
[root@centos8 certs]# openssl x509 -req -days 3650 -in  www.magedu.org.csr -CA
ca.crt -CAkey ca.key  -CAcreateserial  -out www.magedu.org.crt

#验证证书内容
[root@centos8 certs]# openssl x509 -in www.magedu.org.crt  -noout -text
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            bb:76:ea:fe:f4:04:ac:06
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CN, ST=BeiJing, L=Beijing, O=magedu.Ltd, OU=magedu,
CN=magedu.ca/emailAddress=2973707860@qq.com
        Validity
            Not Before: Feb 22 06:14:03 2019 GMT
            Not After : Feb 22 06:14:03 2020 GMT
        Subject: C=CN, ST=BeiJing, L=BeiJing, O=magedu.org, OU=magedu.org,
CN=www.magedu.org/emailAddress=2973707860@qq.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)

#合并CA和服务器证书成一个文件,注意服务器证书在前
[root@centos8 certs]#cat  www.magedu.org.crt ca.crt > www.magedu.org.pem
```

4.6.3 https 配置

```
server {
    listen 80;
    listen 443 ssl;
    ssl_certificate /apps/nginx/certs/www.magedu.org.pem;
    ssl_certificate_key /apps/nginx/certs/www.magedu.org.key;
    ssl_session_cache shared:sslcache:20m;
    ssl_session_timeout 10m;
    root /data/nginx/html;
}

#重启Nginx并访问验证
```




证书
常规
详细信息
证书路径
证书信息
无法将这个证书验证到一个受信任的证书颁发机构。
颁发给: www.magedu.org
颁发者: ca.magedu.org
有效期从 2020/9/26 到 2021/9/26
颁发者说明(S)
确定



4.6.4 实现多域名 https

Nginx 支持基于单个IP实现多域名的功能，并且还支持单IP多域名的基础之上实现HTTPS，其实是基于Nginx的 SNI（Server Name Indication）功能实现，SNI是为了解决一个Nginx服务器内使用一个IP绑定多个域名和证书的功能，其具体功能是客户端在连接到服务器建立SSL链接之前先发送要访问站点的域名（Hostname），这样服务器再根据这个域名返回给客户端一个合适的证书。

范例: SNI功能

```
[root@centos8 ~]#nginx -V
nginx version: nginx/1.18.0
built by gcc 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)
built with OpenSSL 1.1.1c FIPS  28 May 2019
TLS SNI support enabled
configure arguments: --prefix=/apps/nginx --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream
--with-stream_ssl_module --with-stream_realip_module --add-module=../echo-nginx-
module/
```

范例:

```
#制作key和csr文件
[root@centos8 certs]# openssl req  -newkey rsa:4096 -nodes -sha256 -keyout
m.magedu.org.key -out  m.magedu.org.csr

Generating a 4096 bit RSA private key
..........
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:BeiJing
Locality Name (eg, city) [Default City]:BeiJing
Organization Name (eg, company) [Default Company Ltd]:magedu
Organizational Unit Name (eg, section) []:magedu
Common Name (eg, your name or your server's hostname) []:m.magedu.org
Email Address []:2973707860@qq.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

#签名证书
[root@centos8 certs]#  openssl x509 -req -days 3650  -in  m.magedu.org.csr -CA
ca.crt -CAkey ca.key  -CAcreateserial  -out m.magedu.org.crt


#验证证书内容
[root@centos8 certs]# openssl x509 -in m.magedu.org.crt  -noout -text
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            bb:76:ea:fe:f4:04:ac:07
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CN, ST=BeiJing, L=Beijing, O=magedu.Ltd, OU=magedu,
CN=magedu.ca
        Validity
            Not Before: Feb 22 13:50:43 2019 GMT
            Not After : Feb 19 13:50:43 2029 GMT
        Subject: C=CN, ST=BeiJing, L=BeiJing, O=magedu, OU=magedu,
CN=m.magedu.org/emailAddress=2973707860@qq.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
..............
```

```
#合并证书文件
[root@centos8 certs]#cat m.magedu.org.crt ca.crt > m.magedu.org.pem

#Nginx 配置
[root@centos8 certs]# cat /apps/nginx/conf/conf.d/mobile.conf
server {
  listen 80 default_server;
  server_name  m.magedu.org;
  rewrite ^(.*)$ https://$server_name$1 permanent;
}

server {
  listen 443 ssl;
  server_name m.magedu.org;
  ssl_certificate /apps/nginx/certs/m.magedu.org.pem;
  ssl_certificate_key /apps/nginx/certs/m.magedu.org.key;
  ssl_session_cache shared:sslcache:20m;
  ssl_session_timeout 10m;
  location / {
    root "/data/nginx/html/mobile";
  }
location  /mobile_status {
    stub_status;
  }
}
```

4.6.5 实现 HSTS

官方文档:

```
https://www.nginx.com/blog/http-strict-transport-security-hsts-and-nginx/
```

注意: 配置rewrite才能实现http跳转到https

范例:

```
server {
    listen 443 ssl;
    server_name www.magedu.org;
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
always;
    location / {
       if ( $scheme = http ) {
          rewrite ^/(.*)$ https://www.magedu.org/$1 redirect;

        }
        .....
    }
}
```

范例:

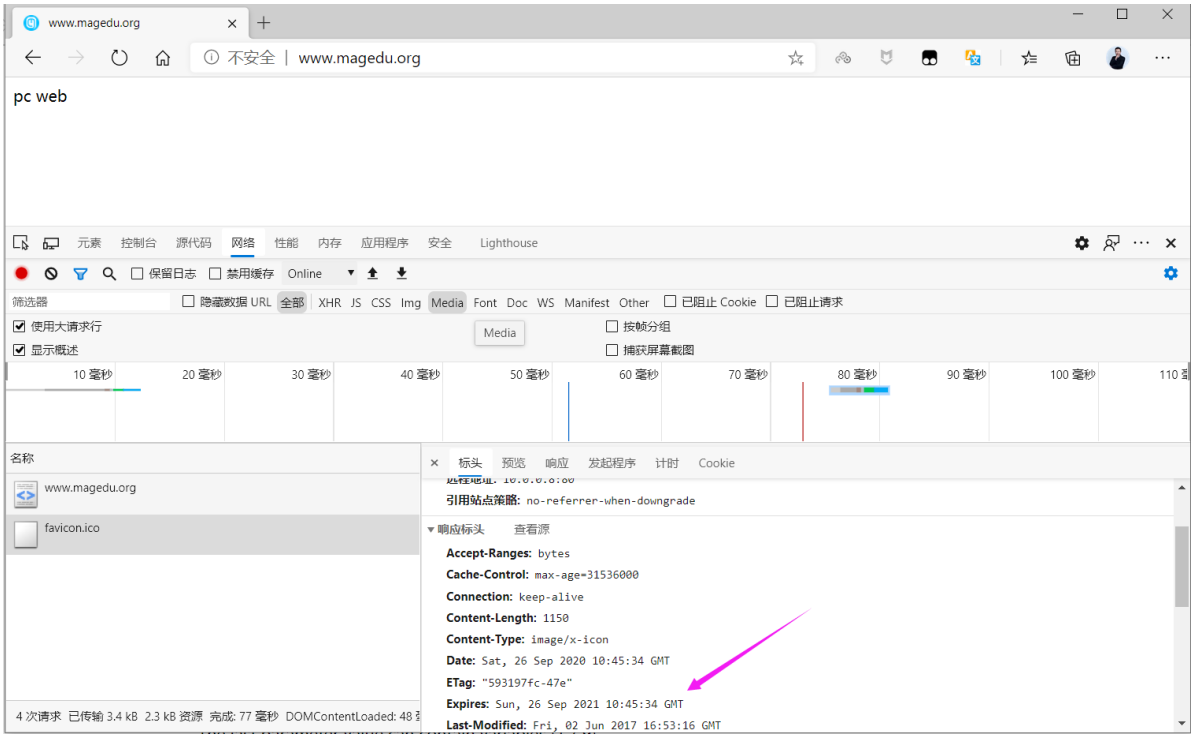
```
[root@centos8 ~]#vim /apps/nginx/conf/conf.d/pc.conf
server {
  listen 80;
  listen 443 ssl;
  ssl_certificate /apps/nginx/conf/conf.d/www.magedu.org.crt;
  ssl_certificate_key /apps/nginx/conf/conf.d/www.magedu.org.key;
  ssl_session_cache shared:sslcache:20m;
  ssl_session_timeout 10m;
  server_name www.magedu.org;
  error_log /apps/nginx/logs/magedu.org_error.log notice;
  access_log  /apps/nginx/logs/magedu.org_access.log  main;
  add_header Strict-Transport-Security "max-age=31536000; includeSubDomains"
always;
  location / {
    root /data/nginx/html/pc;
    if ( $scheme = http ) {
          rewrite ^/(.*)$ https://www.magedu.org/$1 redirect;

    }
  }
[root@centos8 ~]#systemctl restart nginx


[root@centos7 ~]#curl -ikL https://www.magedu.org
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 15:29:56 GMT
Content-Type: text/html
Content-Length: 7
Last-Modified: Sat, 26 Sep 2020 01:18:32 GMT
Connection: keep-alive
ETag: "5f6e96e8-7"
Strict-Transport-Security: max-age=31536000; includeSubDomains
Accept-Ranges: bytes

pc web
```

4.7 关于 favicon.ico

favicon.ico 文件是浏览器收藏网址时显示的图标，当客户端使用浏览器问页面时，浏览器会自己主动发起请求获取页面的favicon.ico文件，但是当浏览器请求的favicon.ico文件不存在时，服务器会记录404日志，而且浏览器也会显示404报错

解决办法：

```
#方法一：服务器不记录访问日志：
location = /favicon.ico {
    log_not_found off;
    access_log off;
}

#方法二：将图标保存到指定目录访问：
#location ~ ^/favicon\.ico$ {
location = /favicon.ico {
     root    /data/nginx/html/pc/images;
     expires 365d;  #设置文件过期时间
}
```



4.8 升级 OpenSSL 版本

OpenSSL程序库当前广泛用于实现互联网的传输层安全（TLS）协议。心脏出血（Heartbleed），也简称为心血漏洞，是一个出现在加密程序库OpenSSL的安全漏洞，此漏洞于2012年被引入了软件中，2014年4月首次向公众披露。只要使用的是存在缺陷的OpenSSL实例，无论是服务器还是客户端，都可能因此而受到攻击。此问题的原因是在实现TLS的心跳扩展时没有对输入进行适当验证（缺少边界检查），因此漏洞的名称来源于“心跳”（heartbeat）。该程序错误属于缓冲区过读，即可以读取的数据比应该允许读取的还多。

范例: 升级OpenSSL解决安全漏洞

```
#准备OpenSSL源码包：
[root@centos8 ~]#cd /usr/local/src
[root@centos8 src]#wget https://www.openssl.org/source/openssl-1.1.1h.tar.gz
[root@centos8 src]#tar xvf openssl-1.1.1h.tar.gz
```

```
#编译安装Nginx并制定新版本OpenSSL路径：
[root@centos8 ~]#cd /usr/local/src/nginx-1.18.0/
[root@centos8 nginx-1.18.0]#./configure --prefix=/apps/nginx --user=nginx --
group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre --with-stream --with-stream_ssl_module --with-stream_realip_module --
add-module=/usr/local/src/echo-nginx-module --with-
openssl=/usr/local/src/openssl-1.1.1h

[root@centos8 nginx-1.18.0]#make && make install

#验证并启动Nginx：
[root@centos8 ~]#nginx  -t
nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
[root@centos8 ~]#systemctl restart nginx

[root@centos8 ~]#nginx  -V
nginx version: wanginx/1.68.9
built by gcc 8.3.1 20191121 (Red Hat 8.3.1-5) (GCC)
built with OpenSSL 1.1.1h  22 Sep 2020
TLS SNI support enabled
configure arguments: --prefix=/apps/nginx --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream
--with-stream_ssl_module --with-stream_realip_module --add-
module=/usr/local/src/echo-nginx-module --with-openssl=/usr/local/src/openssl-
1.1.1h
```

4.9 Nginx Rewrite 相关功能

Nginx服务器利用 ngx_http_rewrite_module 模块解析和处理rewrite请求，此功能依靠 PCRE(perl compatible regular expression)，因此编译之前要安装PCRE库，rewrite是nginx服务器的重要功能之一，用于实现URL的重写，URL的重写是非常有用的功能，比如它可以在我们改变网站结构之后，不需要客户端修改原来的书签，也无需其他网站修改我们的链接，就可以设置为访问，另外还可以在一定程度上提高网站的安全性。

4.9.1 ngx_http_rewrite_module 模块指令

官方文档: https://nginx.org/en/docs/http/ngx_http_rewrite_module.html

4.9.1.1 if 指令

官方文档:

```
https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#if
```

用于条件匹配判断，并根据条件判断结果选择不同的Nginx配置，可以配置在server或location块中进行配置，Nginx的if语法仅能使用if做单次判断，不支持使用if else或者if elif这样的多重判断，用法如下：

```
if （条件匹配） {
   action
}
```

使用正则表达式对变量进行匹配，匹配成功时if指令认为条件为true，否则认为false，变量与表达式之间使用以下符号链接：

```
=   #比较变量和字符串是否相等，相等时if指令认为该条件为true，反之为false
!=  #比较变量和字符串是否不相等，不相等时if指令认为条件为true，反之为false
~   #区分大小写字符，可以通过正则表达式匹配，满足匹配条件为真，不满足匹配条件为假
!~  #区分大小写字符,判断是否匹配，不满足匹配条件为真，满足匹配条件为假

~*  #不区分大小写字符，可以通过正则表达式匹配，满足匹配条件为真，不满足匹配条件为假
!~* #不区分大小字符,判断是否匹配，满足匹配条件为假，不满足匹配条件为真

-f 和 !-f     #判断请求的文件是否存在和是否不存在
-d 和 !-d     #判断请求的目录是否存在和是否不存在
-x 和 !-x     #判断文件是否可执行和是否不可执行
-e 和 !-e     #判断请求的文件或目录是否存在和是否不存在(包括文件，目录，软链接)

#注意：
#如果$变量的值为空字符串或0，则if指令认为该条件为false，其他条件为true。
#nginx 1.0.1之前$变量的值如果以0开头的任意字符串会返回false

#示例：
location /main {
     index index.html;
     default_type text/html;
     if ( $scheme = http ){
       echo "if-----> $scheme";
     }
     if ( $scheme = https ){
      echo "if ----> $scheme";
    }

     #if (-f $request_filename) {
     #   echo "$request_filename  is exist";
     #}
     if (!-e $request_filename) {
        echo "$request_filename  is not exist";
        #return 409;
    }
 }
```

4.9.1.2 set 指令

指定key并给其定义一个变量，变量可以调用Nginx内置变量赋值给key，另外set定义格式为set $key value，value可以是text, variables和两者的组合。

```
location /main {
    root /data/nginx/html/pc;
    index index.html;
    default_type text/html;
    set $name magedu;
    echo $name;
    set $my_port $server_port;
    echo $my_port;
}
```

4.9.1.3 break 指令

用于中断当前相同作用域(location)中的其他Nginx配置，与该指令处于同一作用域的Nginx配置中，位于它前面的配置生效，位于后面的 `ngx_http_rewrite_module` 模块中指令就不再执行，Nginx服务器在根据配置处理请求的过程中遇到该指令的时候，回到上一层作用域继续向下读取配置，该指令可以在server块和locationif块中使用

注意: 如果break指令在location块中后续指令还会继续执行,只是不执行 `ngx_http_rewrite_module` 模块的指令,其它指令还会执行

使用语法如下:

```
if ($slow) {
    limit_rate 10k;
    break;
}

location /main {
    root /data/nginx/html/pc;
    index index.html;
    default_type text/html;
    set $name magedu;
    echo $name;
    break;  #location块中break后面指令还会执行
    set $my_port $server_port;
    echo $my_port;
  }
```

4.9.1.4 return 指令

return用于完成对请求的处理，并直接向客户端返回响应状态码，比如:可以指定重定向URL(对于特殊重定向状态码，301/302等) 或者是指定提示文本内容(对于特殊状态码403/500等)，处于此指令后的所有配置都将不被执行，return可以在server、if 和 location块进行配置

语法格式:

```
return code; #返回给客户端指定的HTTP状态码
return code [text]; #返回给客户端的状态码及响应报文的实体内容，可以调用变量,其中text如果有空格,需要用单或双引号
return code URL; #返回给客户端的URL地址
```

范例:

```
location / {
    root /data/nginx/html/pc;
    default_type text/html;
    index index.html;
      if ( $scheme = http ){
        #return 666;
        #return 666 "not allow http";
        #return 301 http://www.baidu.com;
        return 500 "service error";
        echo "if-----> $scheme"; #return后面的将不再执行
     }
     if ( $scheme = https ){
      echo "if ----> $scheme";
```

```
    }
}
```

范例: return

```
location /test {
    default_type application/json;
    return 200 '{"status:"success"}';
}
```

范例: http跳转到https

```
server {
    listen 80;
    server_name www.magedu.org;
    return 302 https://$server_name$request_uri;
}
server {
    listen 443 ssl;
    server_name www.magedu.org;
    ssl_certificate     /etc/nginx/ssl/www.magedu.org.crt;
    ssl_certificate_key /etc/nginx/ssl/www.magedu.org.key;
    location / {
        root  /data/www/html;
    }
}
```

4.9.1.5 rewrite_log 指令

设置是否开启记录ngx_http_rewrite_module 模块日志记录到 error_log日志文件当中，可以配置在 http、server、location 或 if 中

注意: 需要日志级别为 notice

```
 location /main {
     index index.html;
     default_type text/html;
     set $name magedu;
     echo $name;
     rewrite_log on;
     break;
     set $my_port $server_port;
     echo $my_port;
 }

#重启nginx，访问并验证error_log：
[root@centos8 ~]# tail -f /apps/nginx/logs/error.log
2020/02/27 15:10:02 [warn] 5815#0: *3 using uninitialized "my_port" variable,
client: 10.0.0.1, server: www.magedu.org, request: "GET /main HTTP/1.1", host:
"www.magedu.org"
```

4.9.2 rewrite 指令

通过正则表达式的匹配来改变URI，可以同时存在一个或多个指令，按照顺序依次对URI进行匹配，rewrite主要是针对用户请求的URL或者是URI做具体处理

官方文档:

```
https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#rewrite
```

rewrite可以配置在 server、location、if

语法格式：

```
rewrite regex replacement [flag];
```

rewrite将用户请求的URI基于regex所描述的模式进行检查，匹配到时将其替换为表达式指定的新的URI

注意：如果在同一级配置块中存在多个rewrite规则，那么会自下而下逐个检查;被某条件规则替换完成后，会重新一轮的替换检查，隐含有循环机制,但不超过10次;如果超过，提示500响应码，[flag]所表示的标志位用于控制此循环机制

如果替换后的URL是以http://或https://开头，则替换结果会直接以重定向返回给客户端， 即永久重定向301

正则表达式格式

```
.   #匹配除换行符以外的任意字符
\w  #匹配字母或数字或下划线或汉字
\s  #匹配任意的空白符
\d  #匹配数字
\b  #匹配单词的开始或结束
^   #匹配字付串的开始
$   #匹配字符串的结束
*   #匹配重复零次或更多次
+   #匹配重复一次或更多次
?   #匹配重复零次或一次
(n) #匹配重复n次
{n,} #匹配重复n次或更多次
{n,m} #匹配重复n到m次
*?  #匹配重复任意次，但尽可能少重复
+?  #匹配重复1次或更多次，但尽可能少重复
??  #匹配重复0次或1次，但尽可能少重复
{n,m}? #匹配重复n到m次，但尽可能少重复
{n,}? #匹配重复n次以上，但尽可能少重复
\W  #匹配任意不是字母，数字，下划线，汉字的字符
\S  #匹配任意不是空白符的字符
\D  #匹配任意非数字的字符
\B  #匹配不是单词开头或结束的位置
[^x] #匹配除了x以外的任意字符
[^magedu] #匹配除了magedu 这几个字母以外的任意字符
```

4.9.2.1 rewrite flag 使用介绍

利用nginx的rewrite的指令，可以实现url的重新跳转，rewrite有四种不同的flag，分别是redirect(临时重定向302)、permanent(永久重定向301)、break和last。其中前两种是跳转型的flag，后两种是代理型

- 跳转型指由客户端浏览器重新对新地址进行请求,客户端的浏览器地址信息会发生变化,如:301,302
- 代理型是在WEB服务器内部实现跳转,客户端的浏览器地址信息不会发生变化,如: last,break

rewrite 格式

```
Syntax: rewrite regex replacement [flag]; #通过正则表达式处理用户请求并返回替换后的数据包。
Default:    –
Context:    server, location, if
```

flag 说明

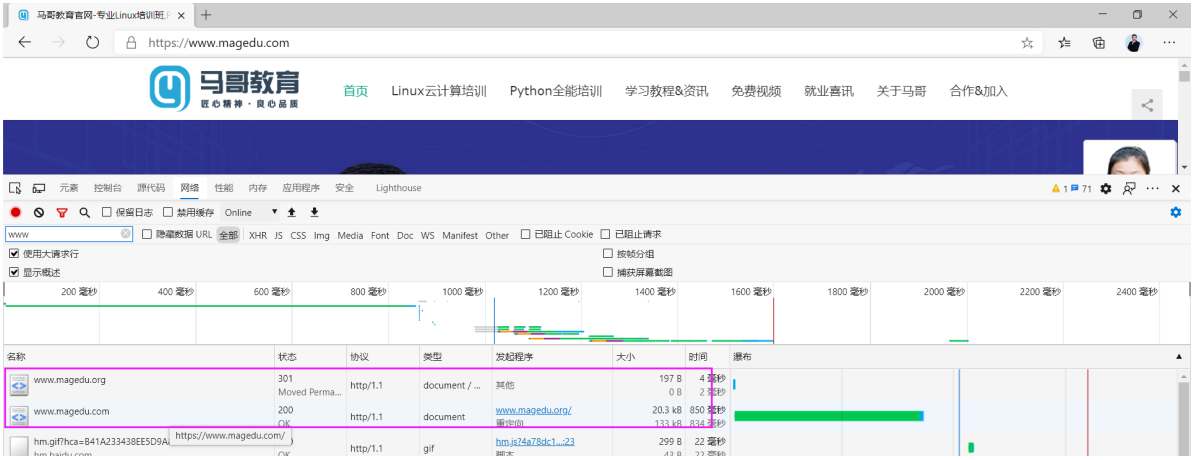
```
redirect;
#临时重定向，重写完成后以临时重定向方式直接返回重写后生成的新URL给客户端，由客户端重新发起请求;使用相对路径,或者http://或https://开头，状态码：302

permanent;
#重写完成后以永久重定向方式直接返回重写后生成的新URL给客户端，由客户端重新发起请求，状态码：301

break;
#重写完成后,停止对当前URL在当前location中后续的其它重写操作，而后直接跳转至重写规则配置块之后的其它配置;结束循环，建议在location中使用
#适用于一个URL一次重写

last;
#重写完成后,停止对当前URI在当前location中后续的其它重写操作，而后对新的URL启动新一轮重写检查，不建议在location中使用
#适用于一个URL多次重写，要注意避免出现超过十次以及URL重写后返回错误的给用户
```

4.9.2.2 rewrite案例: 域名永久与临时重定向

域名的临时的调整，后期可能会变，之前的域名或者URL可能还用、或者跳转的目的域名和URL还会跳转，这种情况浏览器不
会缓存跳转,临时重定向不会缓存域名解析记录(A记录)，但是永久重定向会缓存。

示例: 因业务需要，将访问源域名 www.magedu.org 的请求永久重定向到 www.magedu.com

```
  location  / {
    root /data/nginx/html/pc;
    index index.html;
    rewrite / http://www.magedu.com permanent;
    #rewrite / http://www.magedu.com redirect;
  }
#重启Nginx并访问域名 http://www.magedu.org 进行测试
```

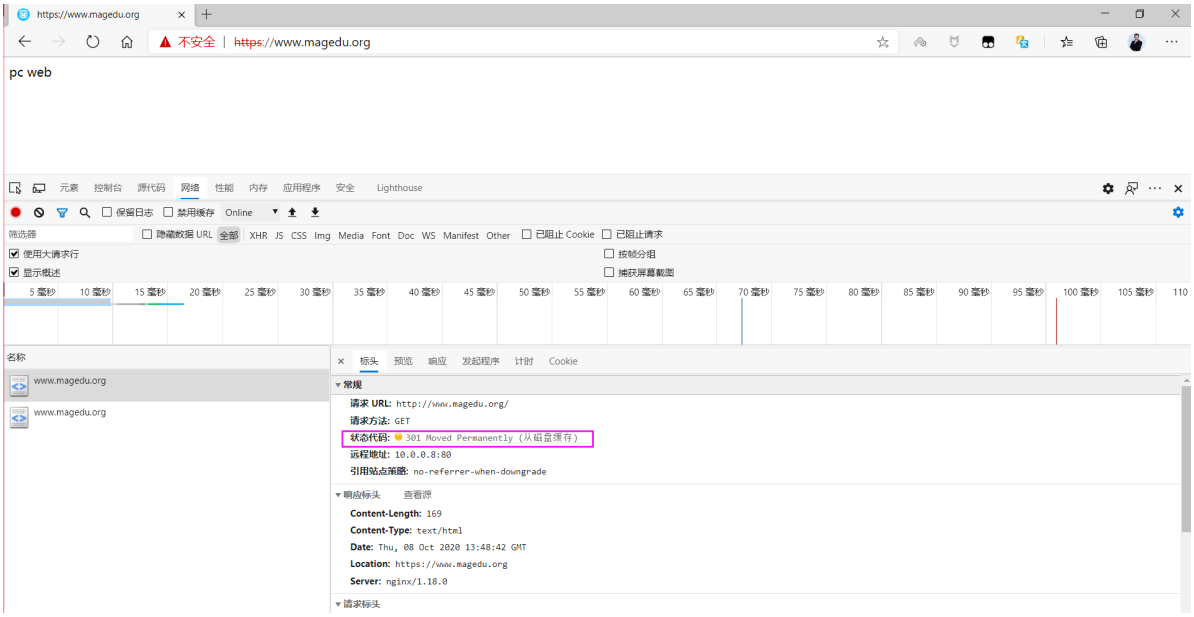


4.9.2.2.1 永久重定向301

域名永久型调整，即域名永远跳转至另外一个新的域名，之前的域名再也不使用，跳转记录可以缓存到客户端浏览器

永久重定向会缓存DNS解析记录，浏览器中有 `from disk cache` 信息,即使nginx服务器无法访问,浏览器也会利用缓存进行重定向

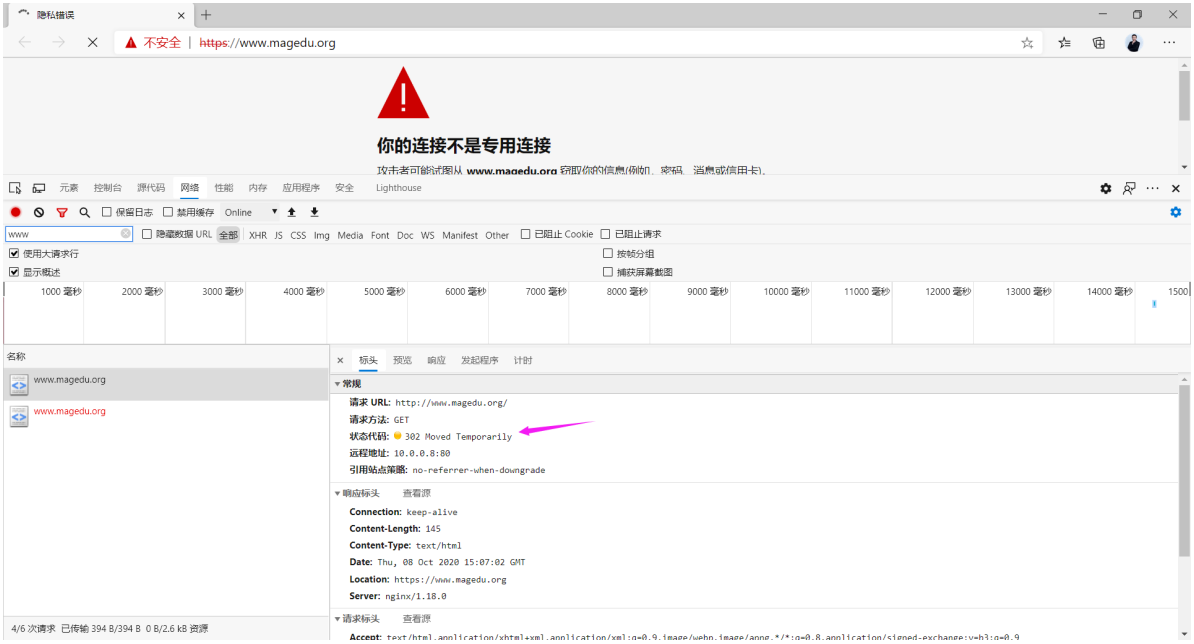
比如: 京东早期的域名 www.360buy.com 由于与360公司类似，于是后期永久重定向到了 www.jd.com



4.9.2.2.2 临时重定向302

域名临时重定向，告诉浏览器域名不是固定重定向到当前目标域名，后期可能随时会更改，因此浏览器不会缓存当前域名的解析记录，而浏览器会缓存永久重定向的DNS解析记录，这也是临时重定向与永久重定向最大的本质区别。

即当nginx服务器无法访问时,浏览器不能利用缓存,而导致重定向失败



4.9.2.3 rewrite 案例: break 与 last

4.9.2.3.1 break和last区别案例

```
[root@centos8 ~]#cat /apps/nginx/conf.d/pc.conf
server {
    listen 80;
    server_name www.magedu.org;
    root /data/nginx/html/pc;
    location /break {
        rewrite ^/break/(.*) /test/$1 break;
    }
    location /last {
        rewrite ^/last/(.*) /test/$1 last;
    }
    location   /test {
      default_type text/plain;
      return 999 "new test";
    }
}
[root@centos8 ~]#cat /data/nginx/html/pc/test/index.html
/data/nginx/html/pc/test/index.html


#分别访问/break/index.html和/last/index.html,观察结果
[root@ubuntu1804 ~]#curl www.magedu.org/break/index.html
/data/nginx/html/pc/test/index.html

[root@ubuntu1804 ~]#curl www.magedu.org/last/index.html
new test
```

范例: last 配置不当容易造成500的死循环

```
[root@centos8 ~]#vim /apps/nginx/conf.d/mobile.conf
server {
    listen 80;
    server_name  m.magedu.org;
    root  /apps/nginx/html/mobile;
```

```
    access_log  logs/mobile-access.log  main;
    location /break {
        rewrite ^/break/(.*) /test/$1 break;
    }
    location /last {
        rewrite ^/last/(.*) /test/$1 last;
    }
    location   /test {
      #default_type text/plain;
      #return 999 "new test";
      rewrite ^/test/(.*) /last/$1 last;
    }
}

[root@ubuntu1804 ~]#curl m.magedu.org/last/index.html
<html>
<head><title>500 Internal Server Error</title></head>
<body>
<center><h1>500 Internal Server Error</h1></center>
<hr><center>nginx/1.18.0</center>
</body>
</html>


[root@centos8 ~]#tail /apps/nginx/logs/error.log
2021/07/20 21:40:56 [error] 54763#0: *1 rewrite or internal redirection cycle
while processing "/test/index.html", client: 10.0.0.100, server: m.magedu.org,
request: "GET /last/index.html HTTP/1.1", host: "m.magedu.org"
```

4.9.2.3.2 break 案例

```
#break测试案例：当客户端访问break的时候，测试通过rewrite将URL重写为test1，然后再通过
rewrite将test1重写为test2测试两条write规则最终哪一条生效，并且测试重写后的URL会不会到其他
location重新匹配
[root@centos8 ~]#vim /apps/nginx/conf.d/pc.conf
   location /break {
      #return 666 "break";
      root /data/nginx;
      index index.html;
      rewrite ^/break/(.*) /test1/$1 break;  #break匹配成功后不再向下匹配，也不会跳转到
其他的location，即直接结束匹配并给客户端返回结果数据。
      rewrite ^/test1/(.*) /test2/$1 break;  #break不会匹配后面的rewrite规则也不匹配
location
    }

    location   /test1 {
      default_type text/plain;
      echo "new test1";
      return 999 "new test1";
    }

     location   /test2 {
      default_type text/plain;
      echo "new test2";
      return 666 "new test2";
```

```
    }

#创建资源路径：
[root@centos8 ~]# mkdir /data/nginx/break
[root@centos8 ~]# mkdir /data/nginx/test1
[root@centos8 ~]# mkdir /data/nginx/test2

[root@centos8 ~]# echo break >  /data/nginx/break/index.html
[root@centos8 ~]# echo test1 > /data/nginx/test1/index.html
[root@centos8 ~]# echo test2 > /data/nginx/test2/index.html


#break访问测试：注意下面的index.html必须加
[root@centos7 ~]#curl -i  www.magedu.org/break/index.html
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 14:16:22 GMT
Content-Type: text/html
Content-Length: 6
Last-Modified: Thu, 08 Oct 2020 14:08:47 GMT
Connection: keep-alive
ETag: "5f7f1d6f-6"
Accept-Ranges: bytes

test1

#最终的结果不会超出break的所在的location而且不会继续匹配当前location后续的write规则，而且直接返回数据给客户端。
```

break适用于不改变客户端访问方式，但是要将访问的目的URL做单次重写的场景，比如:有V1/V2两个版本的网站前端页面并存，旧版本的网站数据在statics,当前还不能丢失，但是要将访问新版本的请求重写到新的静态资源路径static

```
location  /statics {  #旧路径重写至新路径,再响应
     root /data/nginx;
     index index.html;
     rewrite ^/statics/(.*) /static/$1 break;
}
location /static {  #新路径也可以直接响应请的请求
    root /data/nginx;
    index index.html;
}
```

4.9.2.3.3 last 案例

last：对某个location的URL匹配成功后,会停止当前location的后续rewrite规则，并结束当前location，然后将匹配生成的新URL跳转至其他location继续匹配，直到没有location可匹配后, 将最后一次location的数据返回给客户端。

last 适用于要不改变客户端访问方式但是需做多次目的URL重写的场景，使用场景不是很多。

```
[root@centos8 ~]#vim /apps/nginx/conf.d/pc.conf
     location  /test2 {
       default_type text/plain;
       return 666 "new test2";
```

```
        #echo "new test2";
      }
      location  /test1 {
       default_type text/plain;
       #return 999 "new test1";
       #echo "new test1";
       rewrite ^/test1/(.*) /test2/$1 last;
     }
     location /last {
       root /data/nginx;
       index index.html;
       rewrite ^/last/(.*)  /test1/$1 last;
       rewrite ^/test1/(.*) /test2/$1 last; #如果第一条rewrite规则匹配成功则不执行本条，否则执行本条rewrite规则。
     }

#last访问测试：
[root@centos8 ~]#curl   -L   http://www.magedu.org/last/index.html
new test2 #会匹配多个location，直到最终全部匹配完成，返回最后一个location的匹配结果给客户端。
```

4.9.2.4 rewrite案例: 自动跳转 https

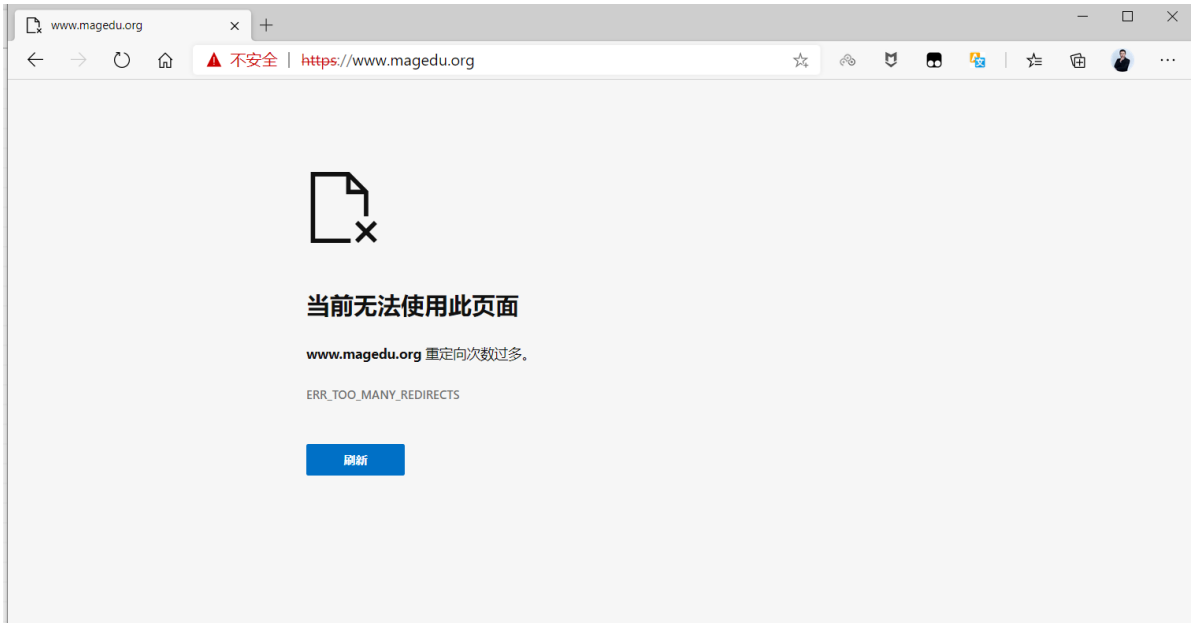
案例：基于通信安全考虑公司网站要求全站 https，因此要求将在不影响用户请求的情况下将http请求全部自动跳转至 https，另外也可以实现部分 location 跳转

```
[root@centos8 ~]#vim /apps/nginx/conf.d/pc.conf
server {
  listen 443 ssl;
  listen 80;
  ssl_certificate /apps/nginx/certs/www.magedu.org.crt;
  ssl_certificate_key /apps/nginx/certs/www.magedu.org.key;
  ssl_session_cache shared:sslcache:20m;
  ssl_session_timeout 10m;
  server_name www.magedu.org;
  location  / {    #针对全站跳转
    root /data/nginx/html/pc;
    index index.html;
    if ($scheme = http ){  #如果没有加条件判断，会导致死循环
        rewrite / https://$host redirect;
   }
  }
  location /login {     #针对特定的URL进行跳转https
    if ($scheme = http ){  #如果没有加条件判断，会导致死循环
        rewrite / https://$host/login redirect;
    }
  }
}
#重启Nginx并访问测试
[root@centos7 ~]#curl -ikL  www.magedu.org
HTTP/1.1 302 Moved Temporarily
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 15:23:48 GMT
Content-Type: text/html
Content-Length: 145
Connection: keep-alive
Location: https://www.magedu.org
```

```
HTTP/1.1 200 OK
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 15:23:48 GMT
Content-Type: text/html
Content-Length: 7
Last-Modified: Sat, 26 Sep 2020 01:18:32 GMT
Connection: keep-alive
ETag: "5f6e96e8-7"
Accept-Ranges: bytes

pc web
```

范例: 如果是因为规则匹配问题导致的陷入死循环

```
[root@centos7 ~]#curl -IkL  http://www.magedu.org
HTTP/1.1 302 Moved Temporarily
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 13:07:19 GMT
Content-Type: text/html
Content-Length: 145
Connection: keep-alive
Location: https://www.magedu.org/

HTTP/1.1 302 Moved Temporarily
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 13:07:19 GMT
Content-Type: text/html
Content-Length: 145
Connection: keep-alive
Location: https://www.magedu.org/

HTTP/1.1 302 Moved Temporarily
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 13:07:19 GMT
Content-Type: text/html
Content-Length: 145
Connection: keep-alive
Location: https://www.magedu.org/

HTTP/1.1 302 Moved Temporarily
Server: nginx/1.18.0
Date: Thu, 08 Oct 2020 13:39:59 GMT
Content-Type: text/html
Content-Length: 145
Connection: keep-alive
Location: https://www.magedu.org

curl: (47) Maximum (50) redirects followed
```



4.9.2.5 rewrite 案例: 判断文件是否存在

案例：当用户访问到公司网站的时输入了一个错误的URL，可以将用户重定向至官网首页

```
[root@centos8 ~]#vim /apps/nginx/conf.d/pc.conf
location  / {
    root /data/nginx/html/pc;
    index index.html;
    if (!-e $request_filename) {
        rewrite .*  http://www.magedu.org/index.html; #实现客户端浏览器的302跳转
        #rewrite .*  /index.html; #web服务器内部跳转
    }
}
#重启Nginx并访问测试
```

4.9.2.6 其它案例

```
#案例1:如果客户端浏览器包含MSIE，则rewrite客户端请求到/msie目录下
if ( $http_user_agent ~ MSIE){
    rewrite ^(.*)$  /msie/$1 break;
}

#案例2: 更换目录访问方式,目录转换为对象存储形式
#要求:
#/20200106/static ->/static?id=20200106
#/20200123/image ->/image?id=20200123
rewrite ^/(\d+)/(.+)/    /$2?id=$1 last;

#案例3:多目录转换访问方式
#要求: www.magedu.com/images/20200106/1.jpg => www.magedu.com/index.do?
name=images&dir=20200106=&file=1.jpg
#规则配置:
if ($host ~* (.*)\.magedu\.com) {
    rewrite ^/(.*)/(\d+)/(.*)$   /index.do?name=$1&dir=$2&file=$3 last;
}
```

4.9.3 Nginx 防盗链

防盗链基于客户端携带的referer实现，referer是记录打开一个页面之前记录是从哪个页面跳转过来的标记信息，如果别人只链接了自己网站图片或某个单独的资源，而不是打开了网站的整个页面，这就是盗链，referer就是之前的那个网站域名，正常的referer信息有以下几种：

```
none：#请求报文首部没有referer首部，比如用户直接在浏览器输入域名访问web网站，就没有referer信息。
blocked：#请求报文有referer首部，但无有效值，比如为空。
server_names：#referer首部中包含本主机名及即nginx 监听的server_name。
arbitrary_string：#自定义指定字符串，但可使用*作通配符。示例：*.magedu.org www.magedu.*
regular expression：#被指定的正则表达式模式匹配到的字符串,要使用~开头，例如：
~.*\.magedu\.com
```

正常通过搜索引擎搜索web 网站并访问该网站的referer信息如下：

```
[root@centos8 ~]#tail -f /apps/nginx/logs/magedu.org_access.log
10.0.0.1 - - [11/Oct/2020:09:28:10 +0800] "GET /images/logo.png HTTP/1.1" 302
145 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/86.0.4240.75 Safari/537.36 Edg/86.0.622.38" "-"
10.0.0.1 - - [11/Oct/2020:09:30:39 +0800] "GET /images/logo.png HTTP/1.1" 200
5934 "http://10.0.0.18/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
Edg/86.0.622.38" "-"
10.0.0.1 - - [11/Oct/2020:09:32:20 +0800] "GET /images/logo.png HTTP/1.1" 200
5934 "http://www.mageedu.org/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
Edg/86.0.622.38" "-"
10.0.0.1 - - [11/Oct/2020:09:35:07 +0800] "GET /test1.html HTTP/1.1" 200 283 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.75 Safari/537.36 Edg/86.0.622.38" "-"
10.0.0.1 - - [11/Oct/2020:09:35:48 +0800] "GET / HTTP/1.1" 200 7
"http://www.magedu.org/test1.html" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
Edg/86.0.622.38" "-"
10.0.0.1 - - [11/Oct/2020:09:37:39 +0800] "GET /images/logo.png HTTP/1.1" 200
5934 "http://www.baidu.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36" "-"
10.0.0.1 - - [11/Oct/2020:09:38:17 +0800] "GET /images/logo.png HTTP/1.1" 200
5934 "http://www.baidu.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36" "-"
10.0.0.1 - - [11/Oct/2020:09:38:27 +0800] "GET / HTTP/1.1" 200 7
"http://www.baidu.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36" "-"
10.0.0.1 - - [11/Oct/2020:09:38:27 +0800] "GET /favicon.ico HTTP/1.1" 200 1150
"http://www.magedu.org/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36" "-"
10.0.0.1 - - [11/Oct/2020:09:41:35 +0800] "GET / HTTP/1.1" 304 0
"http://www.baidu.com/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36" "-"
10.0.0.1 - - [11/Oct/2020:09:41:35 +0800] "GET /favicon.ico HTTP/1.1" 200 1150
"http://www.magedu.org/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36" "-"
10.0.0.1 - - [11/Oct/2020:09:42:41 +0800] "GET /test.html HTTP/1.1" 200 5 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.75 Safari/537.36 Edg/86.0.622.38" "-"
```

```
10.0.0.1 - - [11/Oct/2020:09:42:45 +0800] "GET /test1.html HTTP/1.1" 304 0 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.75 Safari/537.36 Edg/86.0.622.38" "-"
```

json格式日志

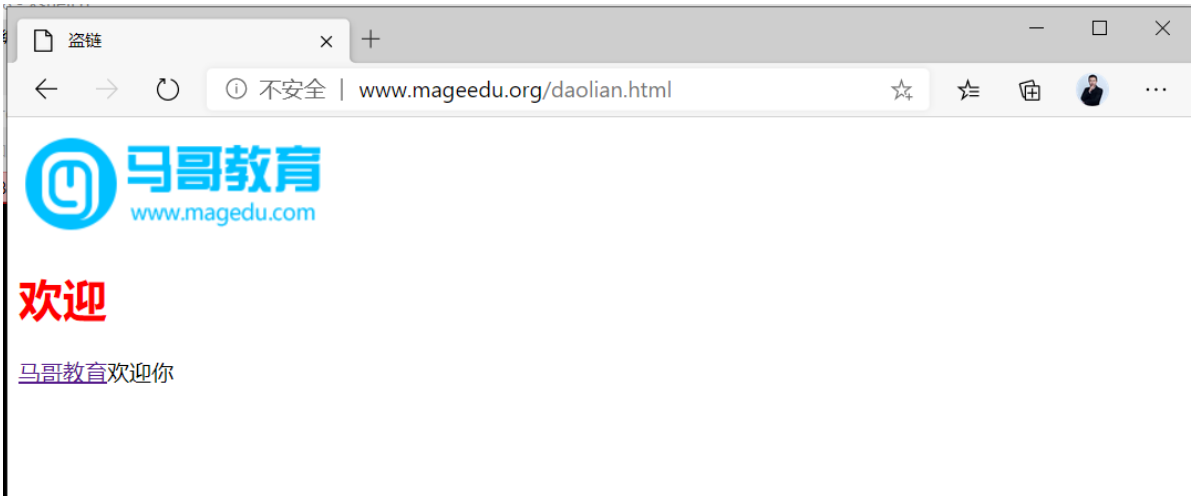
```
#通过搜索引擎访问web网站的referer信息：
==> /apps/nginx/logs/access_json.log <==
{"@timestamp":"2019-02-
28T13:58:46+08:00","host":"10.0.0.100","clientip":"10.0.0.1","siz
e":0,"responsetime":0.000,"upstreamtime":"-","upstreamhost":"-
","http_host":"www.magedu.org","uri":"/index.html","domain":"www.magedu.org","xf
f":"-","referer":"https://www.baidu.com/s?ie=utf-
8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=www.magedu.org&oq=www.mageedu.net&rsv_pq=d6
3060680002eb69&rsv_t=de01TWnmyTdcJqph7SfI1hXgXLJxSSfUPcQ3QkWdJk%2FLNrN95ih3XOhbR
s4&rqlang=cn&rsv_enter=1&inputT=321&rsv_sug3=41&rsv_sug2=0&rsv_sug4=1626","tcp_x
ff":"","http_user_agent":"Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119
Safari/537.36","status":"304"}
```

4.9.3.1 实现盗链

在一个web 站点盗链另一个站点的资源信息，比如:图片、视频等

```
#新建一个主机www.mageedu.org,盗取另一台主机www.magedu.org的图片
[root@centos8 conf.d]# pwd
/apps/nginx/conf/conf.d
[root@centos8 conf.d]# cat  mageedu.org.conf
server {
  listen 80;
  server_name  www.mageedu.org;

  location / {
    index index.html;
    root "/data/nginx/html/magedu";
    access_log /apps/nginx/logs/magedu.org_access.log main;
  }
}

#准备盗链web页面：
[root@centos8 conf.d]# mkdir /data/nginx/html/magedu
[root@centos8 conf.d]# cat /data/nginx/html/magedu/daolian.html
<html>
<head>
<meta http-equiv=Content-Type content="text/html;charset=utf-8">
<title>盗链</title>
</head>
<body>
<img src="http://www.magedu.org/images/logo.png" >
<h1 style="color:red">欢迎大家</h1>
<p><a href=http://www.magedu.org>马哥教育</a>欢迎你</p>
</body>
</html>
```

```
#重启Nginx并访问http://www.mageedu.org/daolian.html 测试
#验证两个域名的日志，是否会在被盗连的web站点的日志中出现以下盗链日志信息：
[root@centos8 ~]#tail /apps/nginx/logs/magedu.org_access.log
10.0.0.1 - - [11/Oct/2020:09:50:07 +0800] "GET /images/logo.png HTTP/1.1" 200
5934 "http://www.mageedu.org/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
Edg/86.0.622.38" "-"
```



4.9.3.2 实现防盗链

基于访问安全考虑，nginx支持通过ngx_http_referer_module模块,检查访问请求的referer信息是否有效实现防盗链功能

官方文档:

```
https://nginx.org/en/docs/http/ngx_http_referer_module.html
```

语法格式:

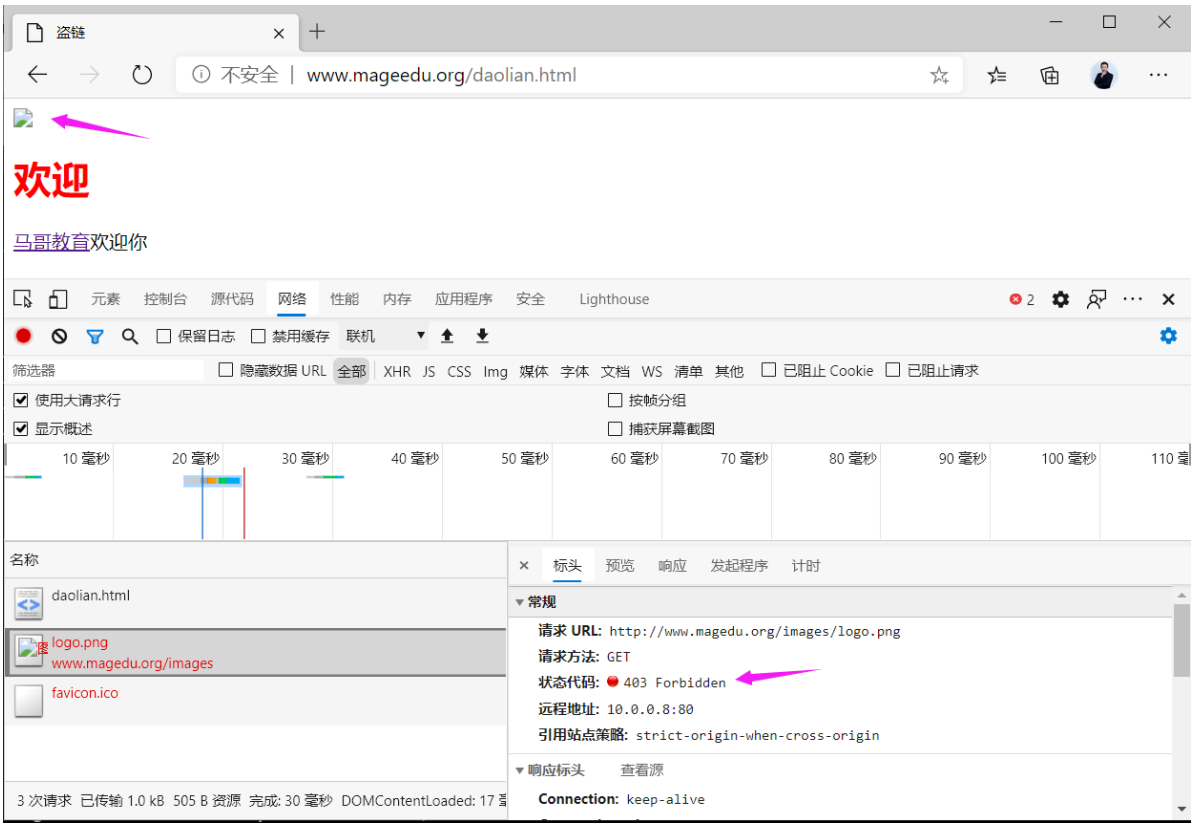
```
location   /images {
    root /data/nginx/html/pc;
    index index.html;
    valid_referers none blocked server_names
          *.example.com example.* www.example.org/galleries/
          ~\.google\.;

    if ($invalid_referer) {
      return 403;
    }
```

范例: 定义防盗链:

```
[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf
server {
    index index.html;
    valid_referers none blocked server_names  *.magedu.com *.magedu.org
~\.google\.  ~\.baidu\.  ~\.bing\. ~\.so\.  ~\.dogedoge\. ;       #定义有效的
referer
    if ($invalid_referer) {     #假如是使用其他的无效的referer访问
      return 403 "Forbidden Access"; #返回状态码403
    }
......
```

```
}
#重启Nginx并访问测试

#指定referer为http://www.baidu.com进行访问
[root@centos7 ~]# curl -e 'http://www.baidu.com' www.magedu.org

#指定referer为http://www.xxx.com进行访问,被拒绝
[root@centos7 ~]# curl -e 'http://www.xxx.com' www.magedu.org
#不加http的referer不会拒绝
[root@centos7 ~]# curl -e 'www.xxx.com' www.magedu.org
```

使用浏览器访问盗链网站 http://www.mageedu.org/daolian.html， 验证是否提前状态码403：



在被盗链的nginx服务器查看日志

```
[root@centos8 ~]#tail /apps/nginx/logs/magedu.org_access.log
10.0.0.1 - - [11/Oct/2020:09:55:26 +0800] "GET /images/logo.png HTTP/1.1" 403 16
"http://www.mageedu.org/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
Edg/86.0.622.38" "-"
```

4.9.4 其它相关高级功能

第三方模块

```
https://github.com/agile6v/awesome-nginx/
```

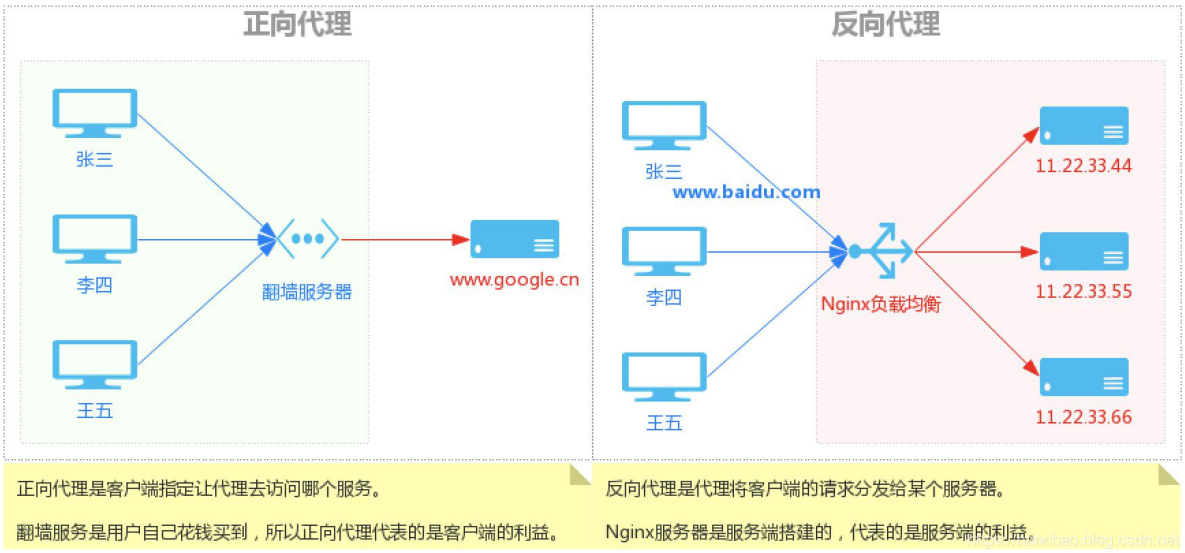
Lua 参考网站

```
https://www.runoob.com/lua/lua-tutorial.html
```

自动生成 nginx 配置文件

```
#需要科学上网
https://www.digitalocean.com/community/tools/nginx
```

5 Nginx 反向代理功能

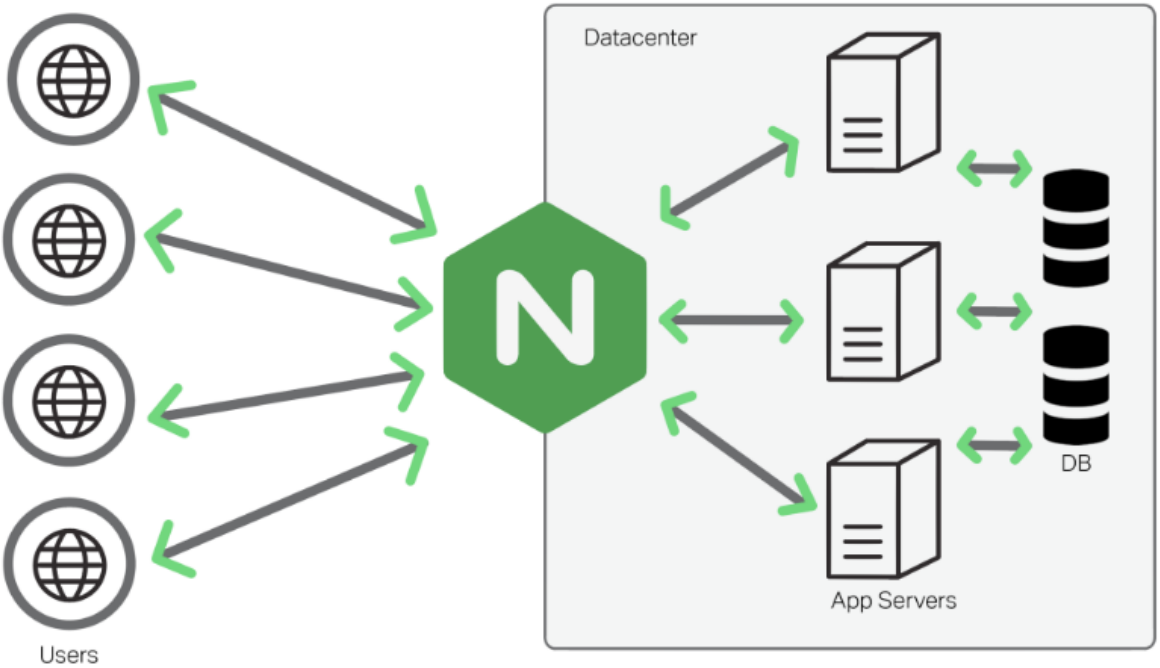


反向代理：reverse proxy，指的是代理外网用户的请求到内部的指定的服务器，并将数据返回给用户的一种方式，这是用的比较多的一种方式。

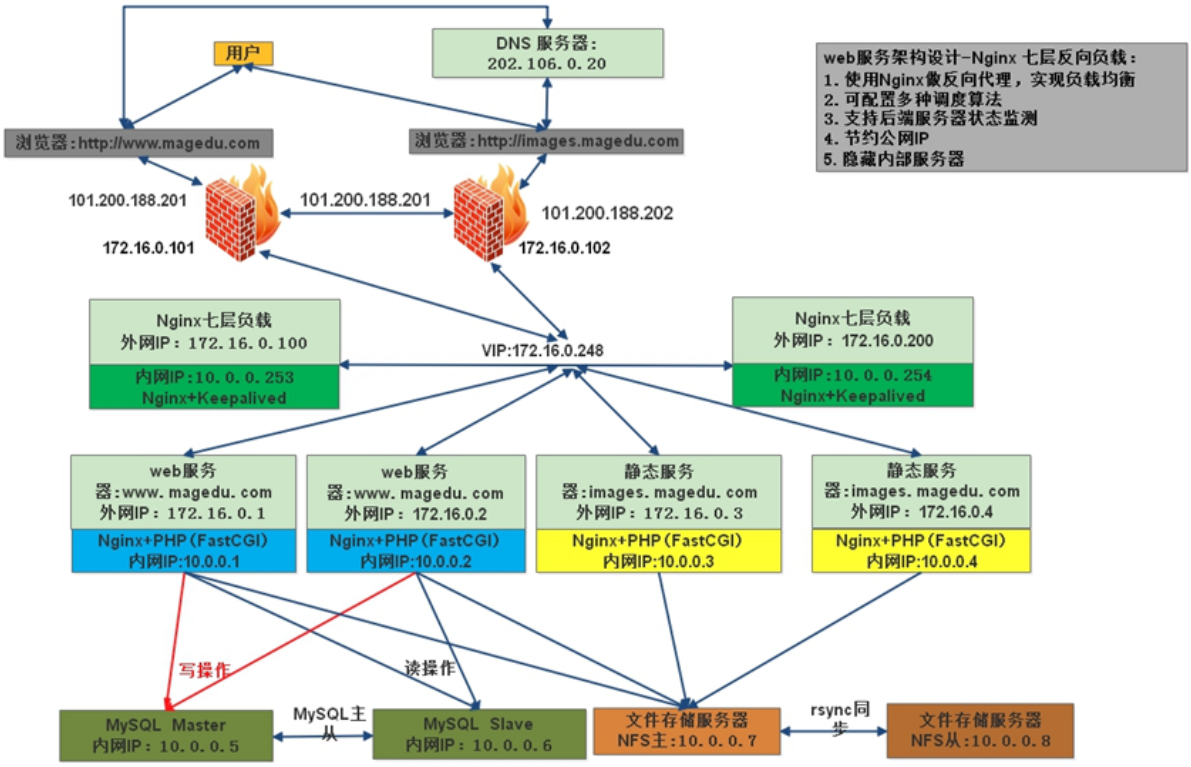
Nginx 除了可以在企业提供高性能的web服务之外，另外还可以将 nginx 本身不具备的请求通过某种预定义的协议转发至其它服务器处理，不同的协议就是Nginx服务器与其他服务器进行通信的一种规范，主要在不同的场景使用以下模块实现不同的功能

```
ngx_http_proxy_module:    #将客户端的请求以http协议转发至指定服务器进行处理
ngx_http_upstream_module  #用于定义为proxy_pass,fastcgi_pass,uwsgi_pass等指令引用的后端服务器分组
ngx_stream_proxy_module:  #将客户端的请求以tcp协议转发至指定服务器处理
ngx_http_fastcgi_module:  #将客户端对php的请求以fastcgi协议转发至指定服务器助理
ngx_http_uwsgi_module:    #将客户端对Python的请求以uwsgi协议转发至指定服务器处理
```

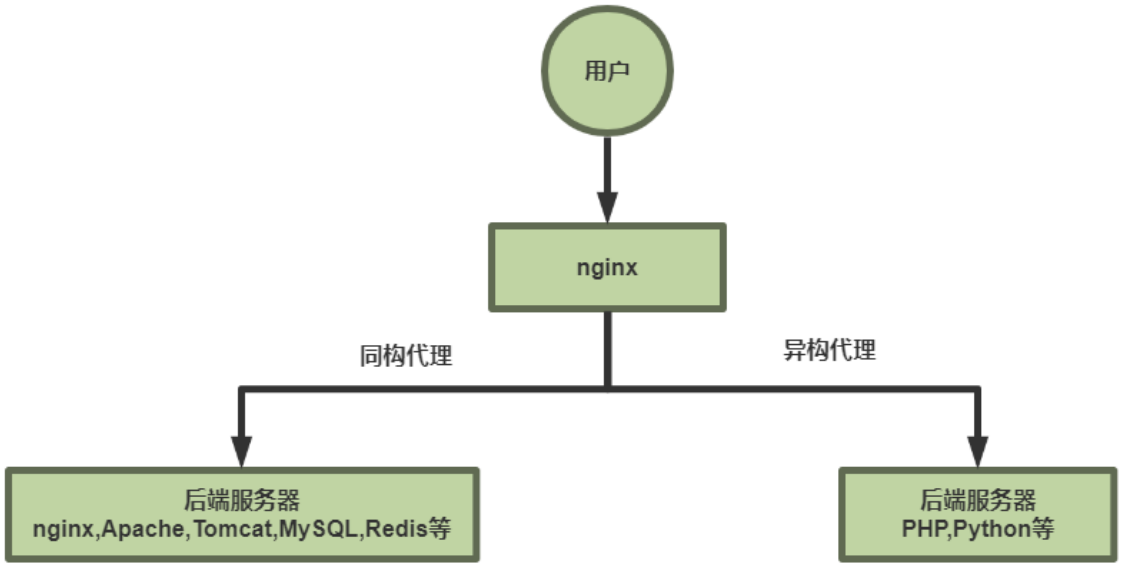
逻辑调用关系：



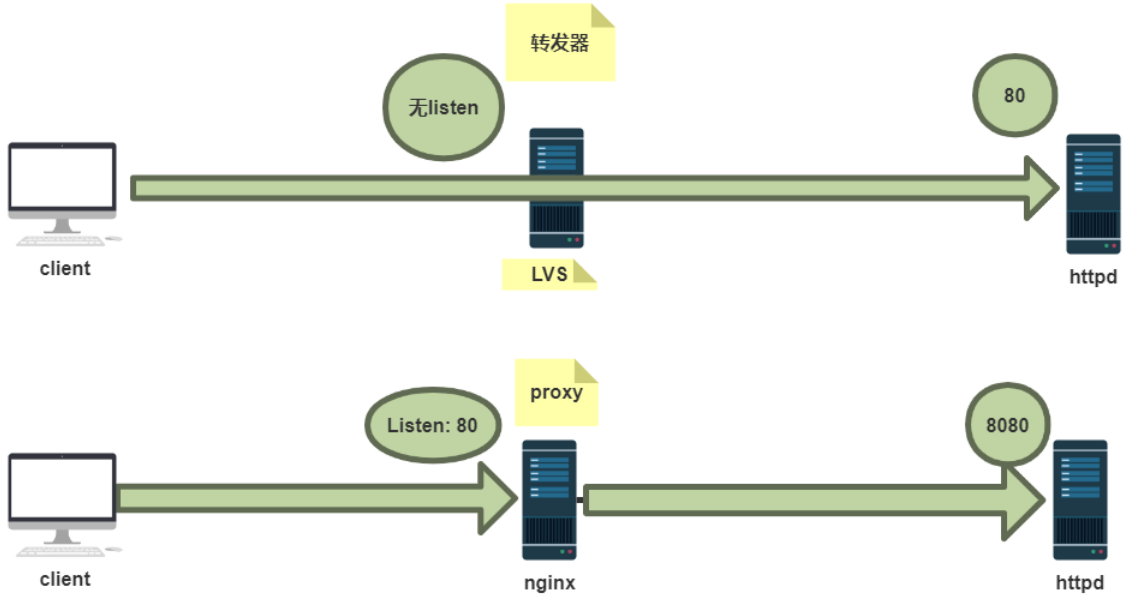
生产环境部署结构：



访问逻辑图：



LVS和nginx的区别



5.1 实现 http 反向代理

官方文档： https://nginx.org/en/docs/http/ngx_http_proxy_module.html,

5.1.1 http 协议反向代理

5.1.1.1 反向代理配置参数

```
#官方文档：https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass
proxy_pass;
#用来设置将客户端请求转发给的后端服务器的主机，可以是主机名(将转发至后端服务做为主机头首部)、IP地址：端口的方式
#也可以代理到预先设置的主机群组，需要模块ngx_http_upstream_module支持

#示例：
 location  /web {
    index index.html;
    proxy_pass http://10.0.0.18:8080; #8080后面无uri,即无 / 符号,需要将location后面url 附加到proxy_pass指定的url后面,此行为类似于root
    #proxy_pass指定的uri不带斜线将访问的/web,等于访问后端服务器http://10.0.0.18:8080/web/index.html，即后端服务器配置的站点根目录要有web目录才可以被访问
    # http://nginx/web/index.html ==> http://10.0.0.18:8080/web/index.html

    proxy_pass http://10.0.0.18:8080/;   #8080后面有uri,即有 / 符号,相当于置换,即访问/web时实际返回proxy_pass后面uri内容.此行为类似于alias
    #proxy_pass指定的uri带斜线，等于访问后端服务器的http://10.0.0.18:8080/index.html 内容返回给客户端
 }  # http://nginx/web/index.html ==> http://10.0.0.18:8080

 #重启Nginx测试访问效果：
 #curl -L  http://www.magedu.org/web



#如果location定义其uri时使用了正则表达式模式(包括~,~*,但不包括^~)，则proxy_pass之后必须不能使用uri; 即不能有/ ,用户请求时传递的uri将直接附加至后端服务器之后
server {
```

```
    ...
    server_name HOSTNAME;
    location ~|~* /uri/ {
        proxy_pass http://host:port; #proxy_pass后面的url 不能加/
    }
    ...
 }
 http://HOSTNAME/uri/ --> http://host/uri/
```

```
proxy_hide_header field;
#用于nginx作为反向代理的时候，在返回给客户端http响应时，隐藏后端服务器相应头部的信息，可以设置在http,server或location块

#示例: 隐藏后端服务器ETag首部字段
 location /web {
    index index.html;
    proxy_pass http://10.0.0.18:8080/;
    proxy_hide_header ETag;
 }
```

```
proxy_pass_header field;
#默认nginx在响应报文中不传递后端服务器的首部字段Date, Server, X-Pad,  X-Accel等参数，如果要传递的话则要使用 proxy_pass_header field声明将后端服务器返回的值传递给客户端
#field 首部字段大小不敏感

#示例:透传后端服务器的Server和Date首部给客户端,同时不再响应报中显示前端服务器的Server字段
proxy_pass_header Server;
proxy_pass_header Date;
```

```
proxy_pass_request_body on | off;
#是否向后端服务器发送HTTP实体部分,可以设置在http,server或location块，默认即为开启
```

```
proxy_pass_request_headers on | off;
#是否将客户端的请求头部转发给后端服务器，可以设置在http,server或location块，默认即为开启
```

```
proxy_set_header;
#可更改或添加客户端的请求头部信息内容并转发至后端服务器，比如在后端服务器想要获取客户端的真实IP的时候，就要更改每一个报文的头部

#示例:
#proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
$proxy_add_x_forwarded_for
the “X-Forwarded-For” client request header field with the $remote_addr variable
appended to it, separated by a comma. If the “X-Forwarded-For” field is not
present in the client request header, the $proxy_add_x_forwarded_for variable is
equal to the $remote_addr variable.

proxy_set_header X-Real-IP  $remote_addr;
#添加HOST到报文头部，如果客户端为NAT上网那么其值为客户端的共用的公网IP地址，常用于在日之中记录客户端的真实IP地址。
#在后端httpd服务器修改配置,添加日志记录X-Forwarded-For字段
 LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" \"%{X-Real-IP}i\"" combined
```

```
#在后端服务器查看日志
[root@centos8 ~]#tail /var/log/httpd/access_log   -f
10.0.0.8 - - [09/Oct/2020:21:50:57 +0800] "HEAD /static/index.html HTTP/1.0" 200
- "-" "curl/7.29.0" "10.0.0.7"
```

```
proxy_connect_timeout time;
#配置nginx服务器与后端服务器尝试建立连接的超时时间，默认为60秒，用法如下：
proxy_connect_timeout 6s;
#60s为自定义nginx与后端服务器建立连接的超时时间,超时会返回客户端504响应码

proxy_read_timeout time;
#配置nginx服务器向后端服务器或服务器组发起read请求后，等待的超时时间，默认60s
proxy_send_timeout time;
#配置nginx项后端服务器或服务器组发起write请求后，等待的超时  时间，默认60s
```

```
proxy_http_version 1.0;
#用于设置nginx提供代理服务的HTTP协议的版本，默认http 1.0
```

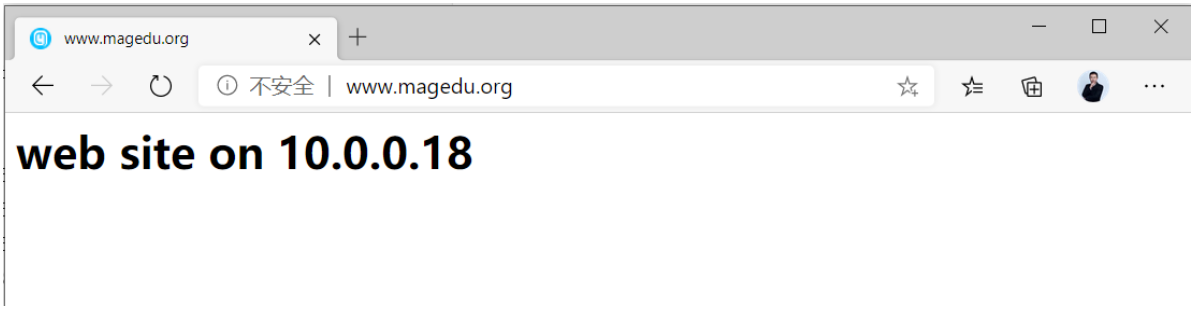
```
proxy_ignore_client_abort off;
#当客户端网络中断请求时，nginx服务器中断其对后端服务器的请求。即如果此项设置为on开启，则服务器会忽略客户端中断并一直等着代理服务执行返回，如果设置为off，则客户端中断后Nginx也会中断客户端请求并立即记录499日志，默认为off。
```

```
proxy_headers_hash_bucket_size 128;
#当配置了 proxy_hide_header和proxy_set_header的时候，用于设置nginx保存HTTP报文头的hash表的上限

proxy_headers_hash_max_size 512;
#设置proxy_headers_hash_bucket_size的最大可用空间

server_namse_hash_bucket_size 512;
#server_name hash表申请空间大小

server_names_hash_max_size  512;
#设置服务器名称hash表的上限大小

proxy_next_upstream error | timeout | invalid_header | http_500 | http_502 |
http_503 | http_504;
#当一台后端朋务器出错,超时,无效首部,500等时,切换至下一个后端服务器提供服务
```

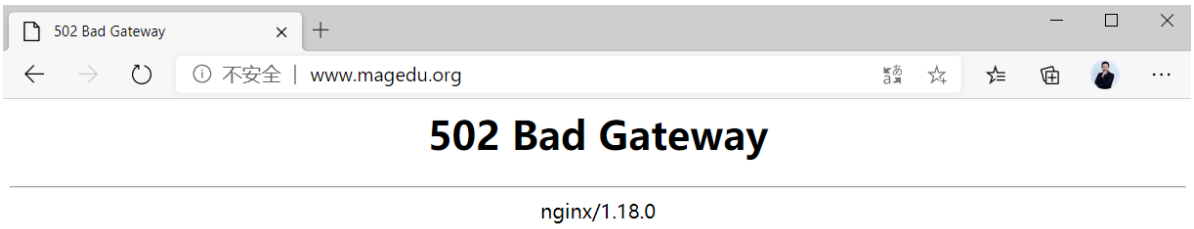
5.1.1.2 实战案例: 反向代理单台 web 服务器

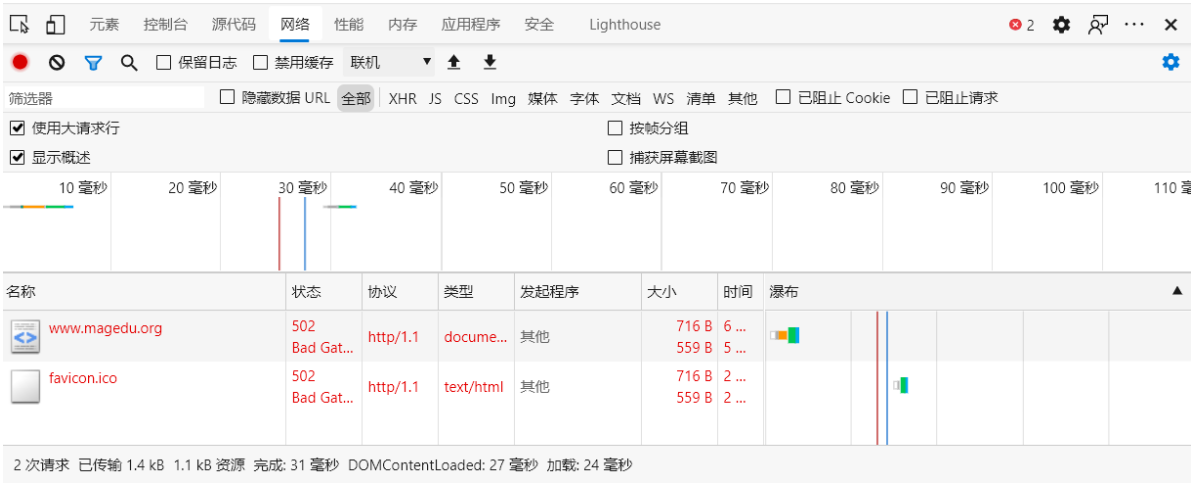
要求：将用户对域 www.magedu.org 的请求转发给后端服务器处理

```
[root@centos8 ~]# cat /apps/nginx/conf/conf.d/pc.conf
server {
  listen 80;
  server_name  www.magedu.org;
  location / {
      proxy_pass http://10.0.0.18/; # http://10.0.0.18/ 最后的 / 可加或不加
  }
}
#重启Nginx 并访问测试
```

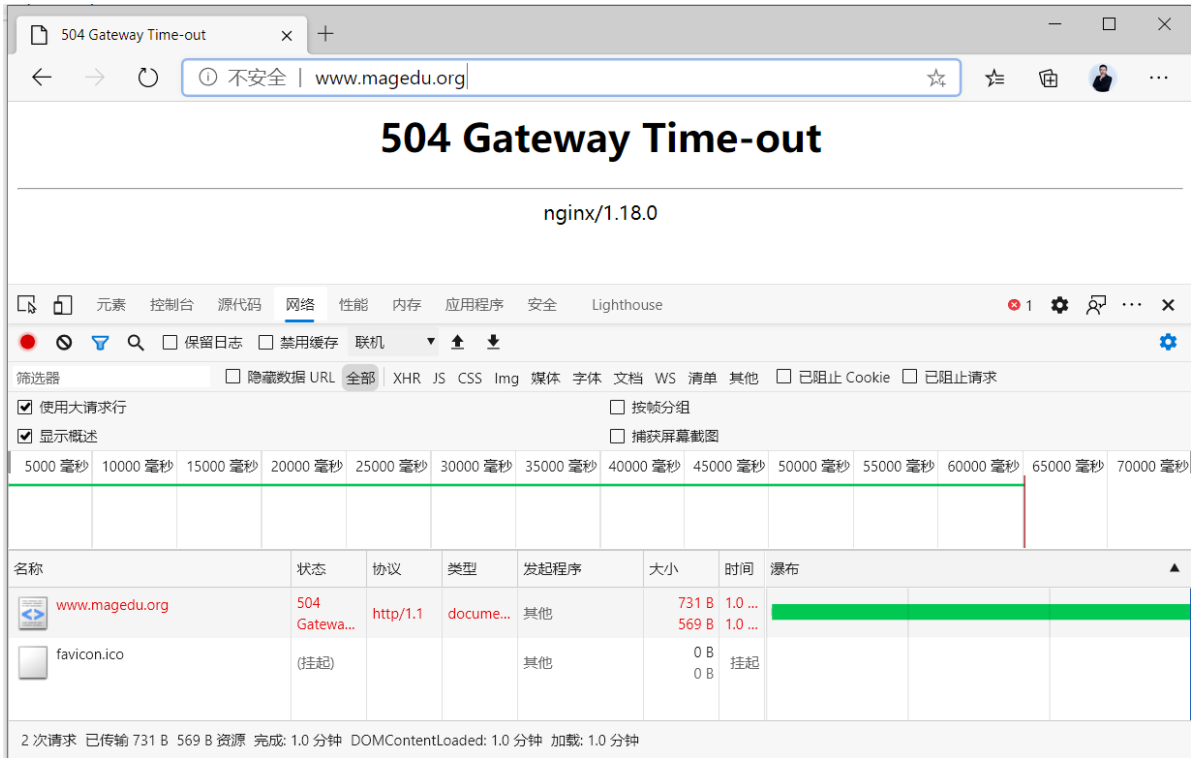


如果后端服务器无法连接((比如:iptables -AINPUT -s nginx_ip -j REJECT或者systemctl stop httpd)), 会出现下面提示





默认在1分钟内后端服务器无法响应(比如:iptables -AINPUT -s nginx_ip -j DROP),会显示下面的504超时提示

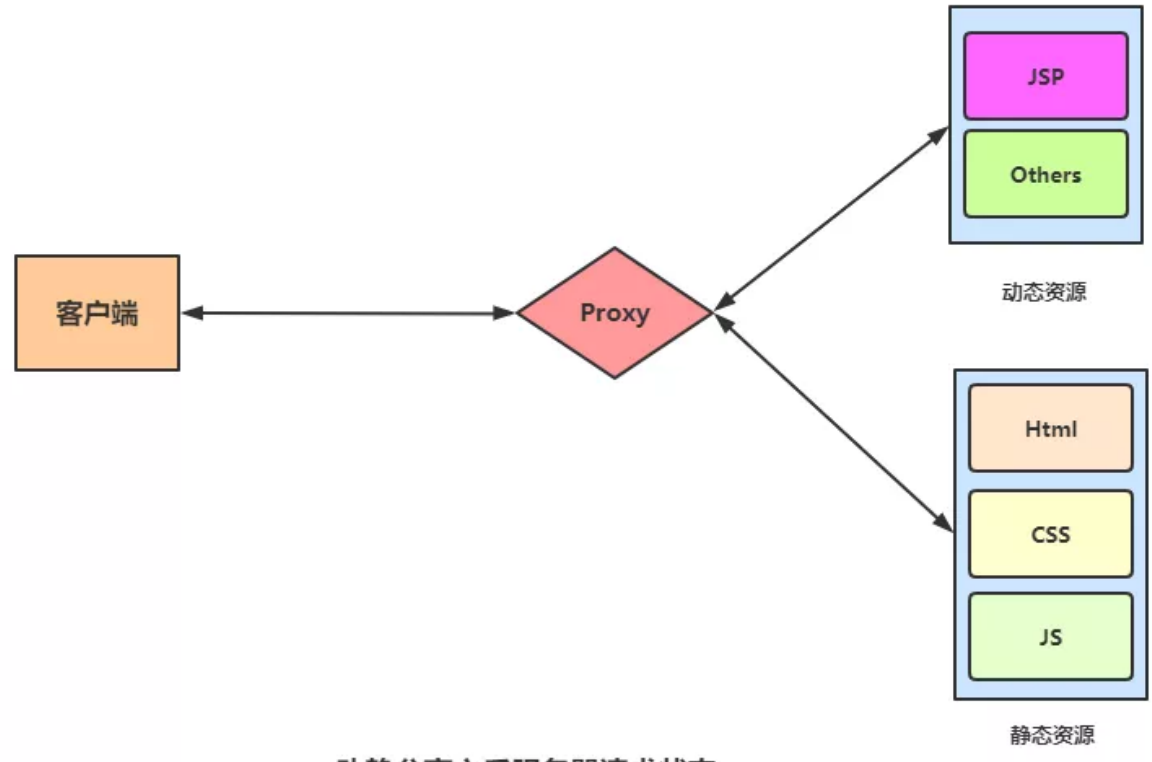


5.1.1.3 实战案例: 指定 location 实现反向代理

5.1.1.3.1 针对指定的 location

```
server {
  listen 80;
  server_name  www.magedu.org;
  location / {
    index index.html index.php;
    root /data/nginx/html/pc;
   }

 location /static {
   #proxy_pass http://10.0.0.18:80/; #注意有后面的/,  表示置换
   proxy_pass http://10.0.0.18:80;  #后面没有 / , 表示附加
 }
}

#后端web服务器必须要有相对于的访问URL
[root@centos8 ~]# mkdir /var/www/html/static
[root@centos8 ~]# echo "web1 page for apache" > /var/www/html/index.html
[root@centos8 ~]# echo "web2 page for apache" > /var/www/html/static/index.html

#重启Nginx并访问测试：
[root@centos8 ~]# curl -L  http://www.magedu.org/
pc web
[root@centos8 ~]# curl -L  http://www.magedu.org/static
web2 page for apache

#Apache的访问日志：
[root@centos8 ~]# tail -f /var/log/httpd/access_log
10.0.0.8 - - [04/Mar/2019:18:52:00 +0800] "GET /static/ HTTP/1.1" 200 21 "-"
"curl/7.29.0"
```

5.1.1.3.2 针对特定的资源实现代理



动静分离之后服务器请求状态

```
[root@centos8 ~]#vim /apps/nginx/conf/conf.d/pc.conf
server {
......
  location ~ \.(jpe?g|png|bmp|gif)$ {
    proxy_pass http://10.0.0.18;      #如果加/ 语法出错

  }
}

#如果 http://10.0.0.18/ 有 /  语法出错
[root@centos8 ~]#nginx -t
nginx: [emerg] "proxy_pass" cannot have URI part in location given by regular
expression, or inside named location, or inside "if" statement, or inside
"limit_except" block in /apps/nginx/conf/conf.d/pc.conf:19
nginx: configuration file /apps/nginx/conf/nginx.conf test failed
```

5.1.1.4 反向代理示例: 缓存功能

缓存功能默认关闭状态,需要先动配置才能启用

```
proxy_cache zone_name | off;  默认off
#指明调用的缓存，或关闭缓存机制;Context:http, server, location
#zone_name 表示缓存的名称.需要由proxy_cache_path事先定义
```

```
proxy_cache_key string;
#缓存中用于“键”的内容，默认值：proxy_cache_key $scheme$proxy_host$request_uri;
```

```
proxy_cache_valid [code ...] time;
#定义对特定响应码的响应内容的缓存时长，定义在http{...}中
    示例:
    proxy_cache_valid 200 302 10m;
    proxy_cache_valid 404 1m;
```

```
proxy_cache_path;
#定义可用于proxy功能的缓存;Context:http
proxy_cache_path path [levels=levels] [use_temp_path=on|off]
keys_zone=zone_name:size [inactive=time] [max_size=size] [manager_files=number]
[manager_sleep=time] [manager_threshold=time] [loader_files=number]
[loader_sleep=time] [loader_threshold=time] [purger=on|off]
[purger_files=number] [purger_sleep=time] [purger_threshold=time];

#示例：在http配置定义缓存信息
proxy_cache_path /var/cache/nginx/proxy_cache #定义缓存保存路径，proxy_cache会自动创
建
   levels=1:2:2 #定义缓存目录结构层次，1:2:2可以生成2^4x2^8x2^8=2^20=1048576个目录
   keys_zone=proxycache:20m #指内存中缓存的大小，主要用于存放key和metadata（如：使用次
数）,一般1M可存放8000个左右的key
   inactive=120s  #缓存有效时间
   max_size=10g; #最大磁盘占用空间，磁盘存入文件内容的缓存空间最大值
```

```
#调用缓存功能，需要定义在相应的配置段，如server{...};或者location等
proxy_cache proxycache;
proxy_cache_key $request_uri; #对指定的数据进行MD5的运算做为缓存的key
proxy_cache_valid 200 302 301 10m; #指定的状态码返回的数据缓存多长时间
proxy_cache_valid any 1m;   #除指定的状态码返回的数据以外的缓存多长时间,必须设置,否则不会缓存
```

```
proxy_cache_use_stale error | timeout | invalid_header | updating | http_500 | http_502 | http_503 | http_504 | http_403 | http_404 | off ; #默认是off
#在被代理的后端服务器出现哪种情况下，可直接使用过期的缓存响应客户端

#示例
proxy_cache_use_stale error http_502 http_503;
```

```
proxy_cache_methods GET | HEAD | POST ...;
#对哪些客户端请求方法对应的响应进行缓存，GET和HEAD方法总是被缓存
```

扩展知识: 清理缓存

```
方法1: rm -rf 缓存目录
方法2: 第三方扩展模块ngx_cache_purge
```

5.1.1.4.1 非缓存场景压测

```
#准备后端httpd服务器
[root@centos8 app1]# pwd
/var/www/html/static
[root@centos8 static]# cat /var/log/messages  > ./log.html #准备测试页面
[root@centos8 ~]# ab -n 2000 -c200  http://www.magedu.org/static/log.html
Concurrency Level:      200
Time taken for tests:   15.363 seconds
Complete requests:      2000
Failed requests:        0
Write errors:           0
Total transferred:      1486898000 bytes
HTML transferred:       1486316000 bytes
Requests per second:    130.18 [#/sec] (mean)
Time per request:       1536.342 [ms] (mean)
Time per request:       7.682 [ms] (mean, across all concurrent requests)
Transfer rate:          94513.36 [Kbytes/sec] received
```

5.1.1.4.2 准备缓存配置

```
[root@centos8 ~]# vim /apps/nginx/conf/nginx.conf
proxy_cache_path /data/nginx/proxycache   levels=1:1:1  keys_zone=proxycache:20m inactive=120s  max_size=1g; #配置在nginx.conf http配置段

[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf
 location /static {  #要缓存的URL 或者放在server配置项对所有URL都进行缓存
    proxy_pass http://10.0.0.18:80;
    proxy_cache proxycache;
    proxy_cache_key $request_uri;
    #proxy_cache_key $host$uri$is_args$args;
    proxy_cache_valid 200 302 301 10m;
```

```
    proxy_cache_valid any 5m;  #必须指定哪些响应码的缓存
    #proxy_set_header clientip $remote_addr;
 }

[root@centos8 ~]# systemctl  restart nginx

#/data/nginx/proxycache/ 目录会自动生成
[root@centos8 ~]#ll /data/nginx/proxycache/ -d
drwx------ 2 nginx root 6 Oct 10 11:30 /data/nginx/proxycache/
[root@centos8 ~]#tree /data/nginx/proxycache/
/data/nginx/proxycache/

0 directories, 0 files
```

5.1.1.4.3 访问并验证缓存文件

```
#访问web并验证缓存目录
[root@centos8 ~]# curl  http://www.magedu.org/static/log.html
[root@centos8 ~]# ab -n 2000 -c200  http://www.magedu.org/static/log.html
Concurrency Level:      200
Time taken for tests:   8.165 seconds
Complete requests:      2000
Failed requests:        0
Write errors:           0
Total transferred:      1486898000 bytes
HTML transferred:       1486316000 bytes
Requests per second:    244.96 [#/sec] (mean)  #性能提高近一倍
Time per request:       816.476 [ms] (mean)
Time per request:       4.082 [ms] (mean, across all concurrent requests)
Transfer rate:          177843.44 [Kbytes/sec] received


#验证缓存目录结构及文件大小
[root@centos8 ~]#tree /data/nginx/proxycache/
/data/nginx/proxycache/
└── d
    └── b
        └── e
            └── a971fce2cfaae636d54b5121d7a74ebd

3 directories, 1 file

[root@centos8 ~]#ll -h
/data/nginx/proxycache/d/b/e/a971fce2cfaae636d54b5121d7a74ebd
-rw------- 1 nginx nginx 727K Oct 10 11:33
/data/nginx/proxycache/d/b/e/a971fce2cfaae636d54b5121d7a74ebd

#验证文件内容：
[root@centos8 ~]#file
/data/nginx/proxycache/d/b/e/a971fce2cfaae636d54b5121d7a74ebd
/data/nginx/proxycache/d/b/e/a971fce2cfaae636d54b5121d7a74ebd: Hitachi SH big-
endian COFF object file, not stripped, 0 section, symbol offset=0x813a815f

[root@centos8 ~]#head -n20
/data/nginx/proxycache/d/b/e/a971fce2cfaae636d54b5121d7a74ebd
#会在文件首部添加相应码
:_
```

```
  )_q,_梳gs"b56f6-5b14894970c60"
KEY: /static/log.html
HTTP/1.1 200 OK
Date: Sat, 10 Oct 2020 03:37:21 GMT
Server: Apache/2.4.37 (centos)
Last-Modified: Sat, 10 Oct 2020 03:22:52 GMT
ETag: "b56f6-5b14894970c60"
Accept-Ranges: bytes
Content-Length: 743158
Connection: close
Content-Type: text/html; charset=UTF-8

Jun 16 06:29:40 10 kernel: Linux version 4.18.0-193.el8.x86_64
(mockbuild@kbuilder.bsys.centos.org) (gcc version 8.3.1 20191121 (Red Hat 8.3.1-
5) (GCC)) #1 SMP Fri May 8 10:59:10 UTC 2020
Jun 16 06:29:40 10 kernel: Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-
193.el8.x86_64 root=UUID=acf9bd1f-caae-4e28-87be-e53afec61347 ro
resume=UUID=409e36d2-ac5e-423f-ad78-9b12db4576bd rhgb quiet
Jun 16 06:29:40 10 kernel: Disabled fast string operations
Jun 16 06:29:40 10 kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87
floating point registers'
Jun 16 06:29:40 10 kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE
registers'
Jun 16 06:29:40 10 kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX
registers'
Jun 16 06:29:40 10 kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:
 256
Jun 16 06:29:40 10 kernel: x86/fpu: Enabled xstate features 0x7, context size is
832 bytes, using 'standard' format.
```

5.1.1.5 添加响应报文的头部信息

nginx基于模块ngx_http_headers_module可以实现对后端服务器响应给客户端的报文中添加指定的响应首部字段

参考链接: https://nginx.org/en/docs/http/ngx_http_headers_module.html

```
Syntax:  add_header name value [always];
Default:  —
Context: http, server, location, if in location

#添加响应报文的自定义首部:
add_header name value  [always];

#示例:
add_header X-Via   $server_addr; #当前nginx主机的IP
add_header X-Cache $upstream_cache_status; #是否缓存命中
add_header X-Accel $server_name; #客户访问的FQDN


#添加自定义响应信息的尾部，使用较少,1.13.2版后支持
add_trailer name value [always];
```

5.1.1.5.1 Nginx 配置

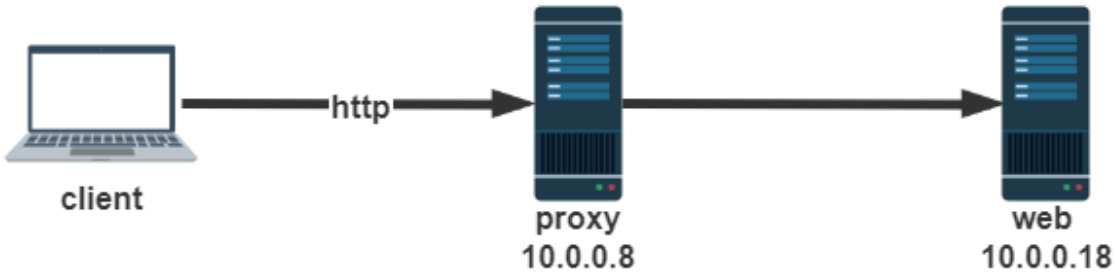
```
location /static {
   proxy_pass http://10.0.0.101:80/;
   proxy_cache proxycache;
   proxy_cache_key $request_uri;
   proxy_cache_valid 200 302 301 1h;
   proxy_cache_valid any 1m;

   #proxy_set_header clientip $remote_addr;
   add_header X-Via  $server_addr;
   add_header X-Cache $upstream_cache_status;
   add_header X-Accel $server_name;
}
```

5.1.1.5.2 验证头部信息

```
[root@centos7 ~]#curl http://www.magedu.org/static/log.html -I
HTTP/1.1 200 OK
Date: Sat, 10 Oct 2020 03:42:31 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 743158
Connection: keep-alive
Vary: Accept-Encoding
Server: Apache/2.4.37 (centos)
Last-Modified: Sat, 10 Oct 2020 03:22:52 GMT
ETag: "b56f6-5b14894970c60"
X-Via: 10.0.0.8
X-Cache: MISS                #第一次无缓存
X-Accel: www.magedu.org
Accept-Ranges: bytes

[root@centos7 ~]#curl http://www.magedu.org/static/log.html -I
HTTP/1.1 200 OK
Date: Sat, 10 Oct 2020 03:42:38 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 743158
Connection: keep-alive
Vary: Accept-Encoding
Server: Apache/2.4.37 (centos)
Last-Modified: Sat, 10 Oct 2020 03:22:52 GMT
ETag: "b56f6-5b14894970c60"
X-Via: 10.0.0.8
X-Cache: HIT                 #第二次命令缓存
X-Accel: www.magedu.org
Accept-Ranges: bytes
```

5.1.1.6 实现反向代理客户端 IP 透传

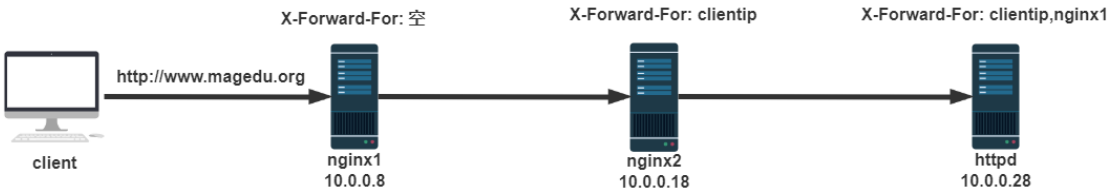
5.1.1.6.1 一级代理实现客户端IP透传




```
[root@centos8 ~]# cat /apps/nginx/conf/conf.d/pc.conf
server {
  listen 80;
  server_name  www.magedu.org;
  location / {
    index index.html index.php;
    root /data/nginx/html/pc;
    proxy_pass http://10.0.0.18;
    #proxy_set_header X-Real-IP  $remote_addr;           #只添加客户端IP到请求报文头部,转发至后端服务器
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; #添加客户端IP和反向代理服务器IP到请求报文头部
 }
}
#说明$proxy_add_x_forwarded_for
#此变量表示将客户端IP追加请求报文中X-Forwarded-For首部字段,多个IP之间用逗号分隔,如果请求中没有X-Forwarded-For,就使用$remote_addr

#重启nginx
[root@centos8 ~]#systemctl restart nginx

#后端Apache配置:
[root@centos8 ~]#vim /etc/httpd/conf/httpd.conf
 LogFormat "%{X-Forwarded-For}i %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
 #重启apache访问web界面并验证apache日志
 [root@centos8 ~]#cat /var/log/httpd/access_log
10.0.0.1 10.0.0.8 - - [05/Mar/2019:00:40:46 +0800] "GET / HTTP/1.0" 200 19 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36"

#Nginx配置:
[root@centos8 conf.d]# cat /apps/nginx/conf/nginx.conf
"$http_x_forwarded_for"' #默认日志格式就有此配置

#重启nginx访问web界面并验证日志格式:
10.0.0.8 - - [04/Mar/2019:16:40:51 +0800] "GET / HTTP/1.0" 200 24 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36" "10.0.0.1"
```

5.1.1.6.2 多级代理实现客户端 IP 透传

范例: 多级代理实现IP透传




```
#第一个代理服务器
[root@nginx1 ~]#vim /apps/nginx/conf/nginx.conf
#开启日志格式,记录x_forwarded_for
http {
    include       mime.types;
    default_type  application/octet-stream;
    proxy_cache_path /data/nginx/proxycache   levels=1:1:1
 keys_zone=proxycache:20m inactive=120s  max_size=1g;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;

#定义反向代理
[root@nginx1 ~]#vim /apps/nginx/conf/conf.d/pc.conf
server {
   location / {
   proxy_pass http://10.0.0.18;
   proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                                }
.......
}

#第二个代理服务器
[root@nginx2 ~]#vim /apps/nginx/conf/nginx.conf
#开启日志格式,记录x_forwarded_for
http {
    include       mime.types;
    default_type  application/octet-stream;
    proxy_cache_path /data/nginx/proxycache   levels=1:1:1
 keys_zone=proxycache:20m inactive=120s  max_size=1g;
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;

#定义反向代理
[root@nginx2 ~]#vim /etc/nginx/nginx.conf
  server {
        location / {
          proxy_pass http://10.0.0.28;
          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        }
```

```
        .....
}

#在第一个proxy上面查看日志
[root@nginx1 ~]#tail /apps/nginx/logs/access.log -f
10.0.0.7 - - [11/Oct/2020:14:37:00 +0800] "GET /index.html HTTP/1.1" 200 11 "-"
"curl/7.58.0" "-"

#在第二个proxy上面查看日志
[root@nginx2 ~]#tail /apps/nginx/logs/access.log -f
10.0.0.8 - - [11/Oct/2020:14:37:00 +0800] "GET /index.html HTTP/1.0" 200 11 "-"
"curl/7.58.0" "10.0.0.7"

#后端服务器配置日志格式
[root@centos8 ~]#vim /etc/httpd/conf/httpd.conf
LogFormat "\"%{x-Forwarded-For}i\"  %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%
{User-Agent}i\"" testlog
CustomLog "logs/access_log" testlog


#测试访问
[root@centos7 ~]#curl www.magedu.org/index.html
<h1> web site on 10.0.0.28 </h1>

#后端服务器查看日志
[root@centos8 ~]#tail -f /var/log/httpd/access_log
"10.0.0.7, 10.0.0.8"  10.0.0.18 - - [11/Oct/2020:14:37:00 +0800] "GET
/index.html HTTP/1.0" 200 34 "-" "curl/7.29.0"
```

5.1.2 http 反向代理负载均衡

在上一个节中Nginx可以将客户端的请求转发至单台后端服务器但是无法转发至特定的一组的服务器，而且不能对后端服务器提供相应的服务器状态监测，Nginx 可以基于ngx_http_upstream_module模块提供服务器分组转发、权重分配、状态监测、调度算法等高级功能

官方文档：https://nginx.org/en/docs/http/ngx_http_upstream_module.html

5.1.2.1 http upstream配置参数

```
#自定义一组服务器，配置在http块内
upstream name {
    server .....
    ......
}

#示例
upstream backend {
    server backend1.example.com weight=5;
    server 127.0.0.1:8080  max_fails=3 fail_timeout=30s;
    server unix:/tmp/backend3;

    server backup1.example.com  backup;
}
```

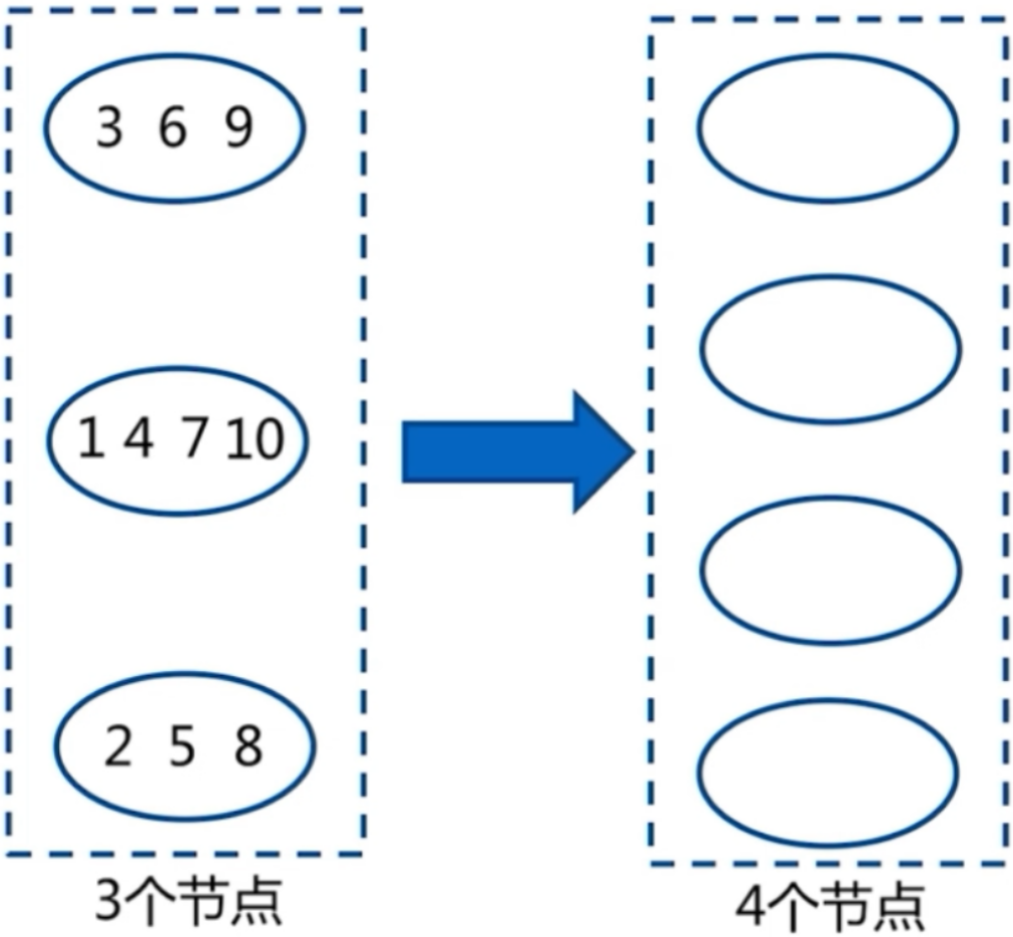
```
server address [parameters];
#配置一个后端web服务器，配置在upstream内，至少要有一个server服务器配置。

#server支持的parameters如下：
weight=number #设置权重，默认为1,实现类似于LVS中的WRR,WLC等
max_conns=number  #给当前后端server设置最大活动链接数，默认为0表示没有限制
max_fails=number  #后端服务器的下线条件,当客户端访问时,对本次调度选中的后端服务器连续进行检测多少次,如果都失败就标记为不可用,默认为1次,当客户端访问时,才会利用TCP触发对探测后端服务器健康性检查,而非周期性的探测
fail_timeout=time #后端服务器的上线条件,对已经检测到处于不可用的后端服务器,每隔此时间间隔再次进行检测是否恢复可用，如果发现可用,则将后端服务器参与调度,默认为10秒
backup  #设置为备份服务器，当所有后端服务器不可用时,才会启用此备用服务器
down    #标记为down状态,可以平滑下线后端服务器,新用户不再调度到此主机,旧用户不受影响
```

```
hash KEY [consistent];
#基于指定请求报文中首部字段或者URI等key做hash计算，使用consistent参数，将使用ketama一致性hash算法，适用于后端是Cache服务器（如varnish）时使用，consistent定义使用一致性hash运算，一致性hash基于取模运算

#示例
hash $request_uri consistent; #基于用户请求的uri做hash
hash $cookie_sessionid  #基于cookie中的sessionid这个key进行hash调度,实现会话绑定
```

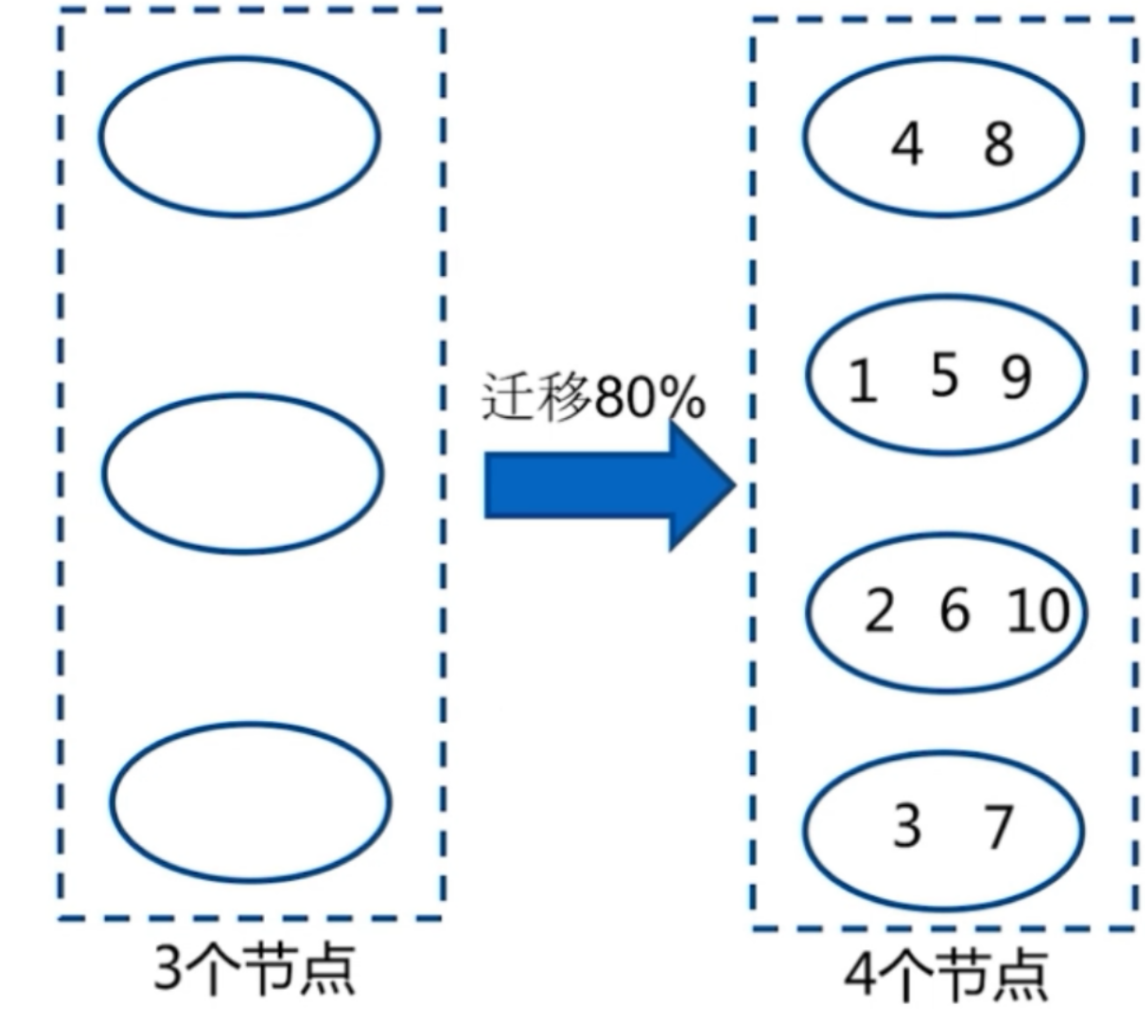
```
ip_hash;
#源地址hash调度方法，基于的客户端的remote_addr(源地址IPv4的前24位或整个IPv6地址)做hash计算，以实现会话保持
#hash $remote_addr 则是对全部32bit的IPV4进行hash计算
```

```
least_conn;
#最少连接调度算法，优先将客户端请求调度到当前连接最少的后端服务器,相当于LVS中的WLC
```

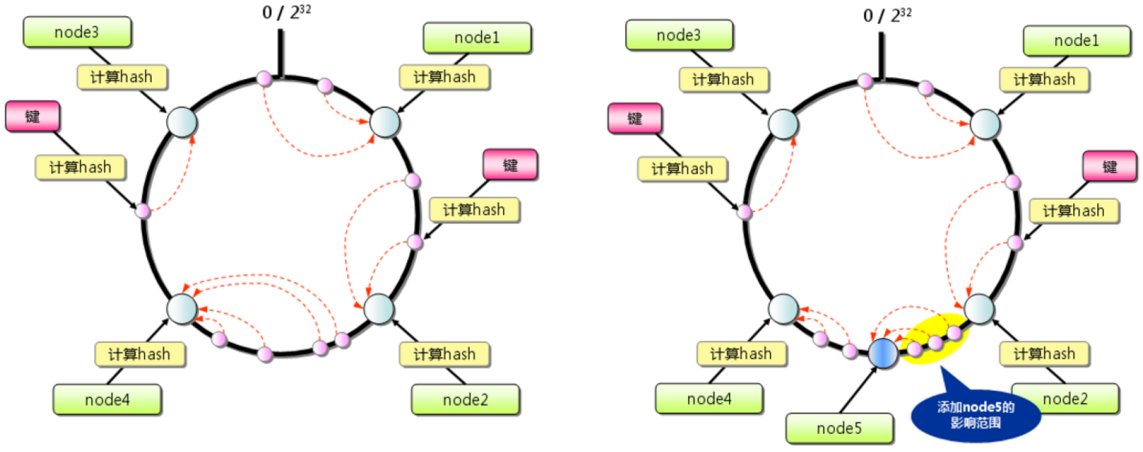
基于 hash 的调度算法



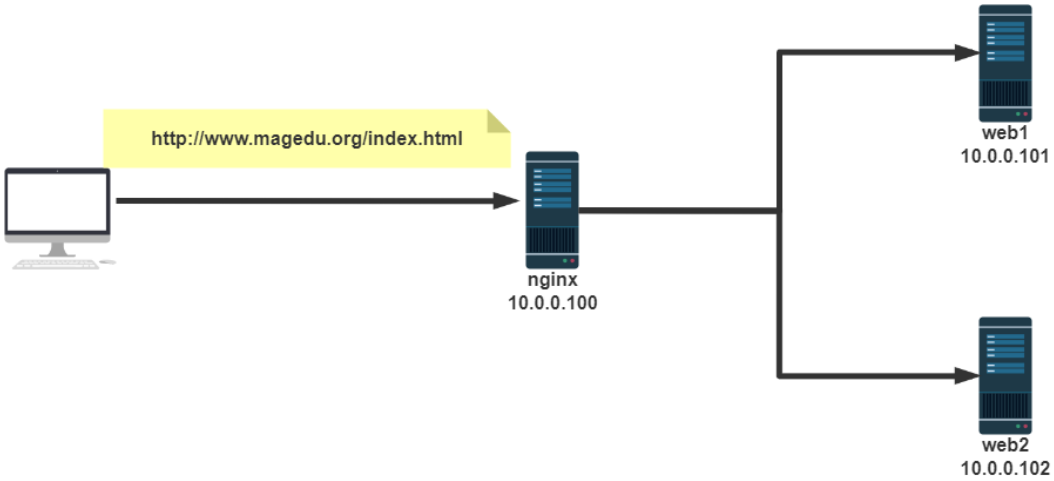
添加节点后,会导致很多的调度的缓存信息失效



一致性hash 算法



5.1.2.2 反向代理示例: 后端多台 web服务器



环境说明：

```
10.0.0.100  #Nginx 代理服务器
10.0.0.101  #后端web A，Apache部署
10.0.0.102  #后端web B，Apache部署
```

5.1.2.2.1 部署后端 Apache服务器

```
[root@centos8 ~]# yum install httpd -y
[root@centos8 ~]# echo "web1 10.0.0.101" > /var/www/html/index.html
[root@centos8 ~]# systemctl enable --now httpd

[root@centos8 ~]# yum install httpd -y
[root@centos8 ~]# echo "web2 10.0.0.102" >> /var/www/html/index.html
[root@centos8 ~]# systemctl enable --now httpd

#访问测试
[root@centos8 ~]# curl http://10.0.0.101
web1 10.0.0.101
[root@centos8 ~]# curl http://10.0.0.102
web2 10.0.0.102
```

5.1.2.2.2 配置 nginx 反向代理

注意: 本节实验过程中先关闭缓存

```
[root@centos8 ~]# cat /apps/nginx/conf/conf.d/pc.conf
http {
 upstream  webserver {
  #hash $request_uri consistent;
  #hash $cookie_sessionid
  #ip_hash;
  #least_conn;
  server 10.0.0.101:80 weight=1 fail_timeout=5s max_fails=3; #后端服务器状态监测
  server 10.0.0.102:80 weight=1 fail_timeout=5s max_fails=3;
  #server 127.0.0.1:80 weight=1 fail_timeout=5s max_fails=3 backup;
 }

server {
  listen 80;
  server_name  www.magedu.org;
  location / {
```

```
    index index.html index.php;
    root /data/nginx/html/pc;
   }

 location /web {
    index index.html;
    proxy_pass http://webserver/;
    proxy_next_upstream error | timeout | invalid_header | http_500 | http_502 |
http_503 | http_504;
 }
}

#重启Nginx 并访问测试
[root@centos8 ~]# curl   http://www.magedu.org/web
web1 10.0.0.101
[root@centos8 ~]# curl  http://www.magedu.org/web
web2 10.0.0.102

#关闭10.0.0.101和10.0.0.102，测试nginx backup服务器可用性：
[root@centos8 ~]# while  true;do curl http://www.magedu.org/web;sleep 1;done
```

5.1.2.4 实战案例: 基于Cookie 实现会话绑定

```
[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf
http {
    upstream websrvs {
        hash $cookie_hello;  #hello是cookie的key的名称
        server 10.0.0.101:80 weight=2;
        server 10.0.0.102:80 weight-1;

    }
}
[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf
server {
    location /{
    #proxy_cache mycache;
    proxy_pass http://websrvs;
    }
}

#测试
[root@centos8 ~]#curl  -b  hello=wang  http://www.magedu.org/
```

5.2 实现 Nginx 四层负载均衡

Nginx在1.9.0版本开始支持tcp模式的负载均衡，在1.9.13版本开始支持udp协议的负载，udp主要用于DNS的域名解析，其配置方式和指令和http 代理类似，其基于ngx_stream_proxy_module模块实现tcp负载，另外基于模块ngx_stream_upstream_module实现后端服务器分组转发、权重分配、状态监测、调度算法等高级功能。

如果编译安装,需要指定 --with-stream 选项才能支持ngx_stream_proxy_module模块

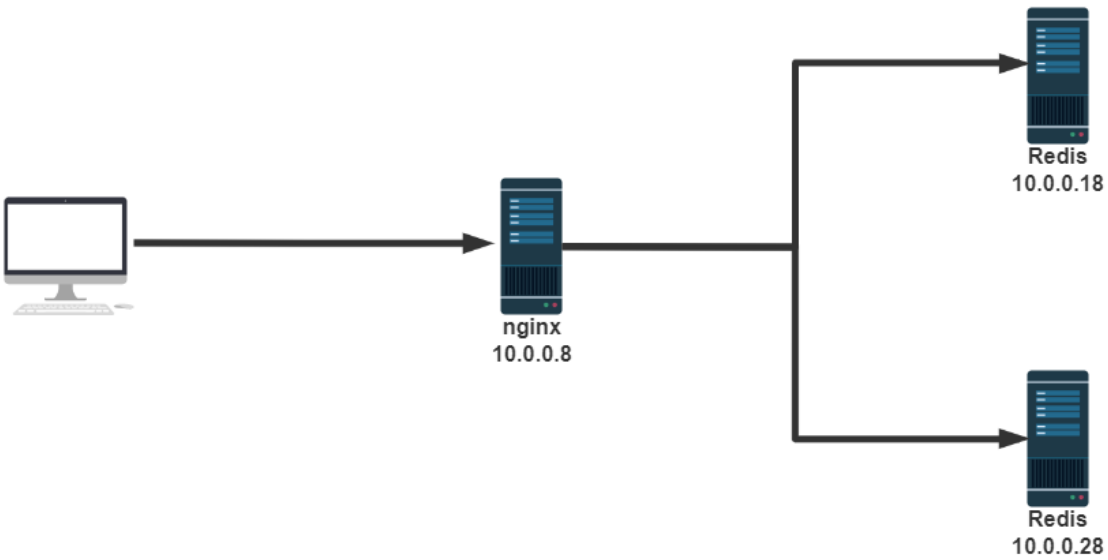
官方文档:

```
https://nginx.org/en/docs/stream/ngx_stream_proxy_module.html
http://nginx.org/en/docs/stream/ngx_stream_upstream_module.html
```

5.2.1 tcp负载均衡配置参数

```
stream { #定义stream相关的服务，Context:main
    upstream backend { #定义后端服务器
        hash $remote_addr consistent; #定义调度算法

        server backend1.example.com:12345 weight=5; #定义具体server
        server 127.0.0.1:12345      max_fails=3 fail_timeout=30s;
        server unix:/tmp/backend3;
    }

    upstream dns {  #定义后端服务器
       server 10.0.0.1:53535;  #定义具体server
       server dns.example.com:53;
    }

    server { #定义server
        listen 12345; #监听IP:PORT
        proxy_connect_timeout 1s; #连接超时时间
        proxy_timeout 3s; #转发超时时间
        proxy_pass backend; #转发到具体服务器组
    }

    server {
        listen 127.0.0.1:53 udp reuseport;
        proxy_timeout 20s;
        proxy_pass dns;
    }

    server {
        listen [::1]:12345;
        proxy_pass unix:/tmp/stream.socket;
    }
}
```

5.2.2 负载均衡实例：Redis



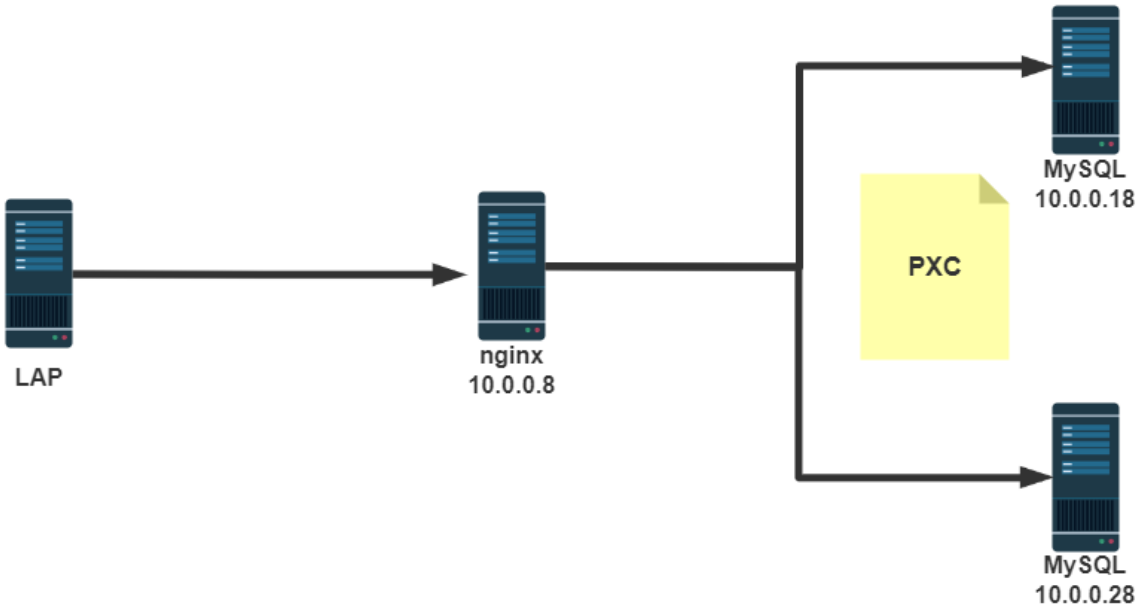
5.2.2.1 后端服务器安装 redis

```
#安装两台redis服务器
[root@centos8 ~]# yum -y install redis
[root@centos8 ~]# sed -i '/^bind /c bind 0.0.0.0' /etc/redis.conf

[root@centos8 ~]# systemctl  enable --now redis
[root@centos8 ~]# ss -tnl | grep 6379
LISTEN     0      128          *:6379                     *:*
```

5.2.2.2 nginx 配置

```
[root@centos8 ~]# vim /apps/nginx/conf/nginx.conf
include /apps/nginx/conf/tcp/tcp.conf; #注意此处的include与http模块平级

[root@centos8 ~]# mkdir /apps/nginx/conf/tcp
[root@centos8 ~]# cat /apps/nginx/conf/tcp/tcp.conf
stream {
  upstream redis_server {
    #hash $remote_addr consistent;
    server 10.0.0.18:6379 max_fails=3 fail_timeout=30s;
    server 10.0.0.28:6379 max_fails=3 fail_timeout=30s;
  }

  server {
    listen 10.0.0.8:6379;
    proxy_connect_timeout 3s;
    proxy_timeout 3s;
    proxy_pass redis_server;
  }
}

#重启nginx并访问测试
[root@centos8 ~]# systemctl  restart nginx
[root@centos8 ~]# ss -tnl | grep 6379
LISTEN     0      128    10.0.0.8:6379                    *:*


#测试通过nginx 负载连接redis：
[root@centos8 ~]#redis-cli  -h 10.0.0.8 set name wang
OK
[root@centos8 ~]#redis-cli  -h 10.0.0.8 get name
(nil)
[root@centos8 ~]#redis-cli  -h 10.0.0.8 get name
"wang"
```

5.2.3 负载均衡实例: MySQL



5.2.3.1 后端服务器安装 MySQL

```
#先修改后端两个服务器的主机名,方便测试
[root@centos8 ~]#hostnamectl set-hostname mysql-server1.magedu.org
[root@centos8 ~]#hostnamectl set-hostname mysql-server2.magedu.org

#安装MySQL服务
[root@centos8 ~]# yum  -y install mariadb-server
[root@centos8 ~]# systemctl enable --now mariadb
MariaDB [(none)]> create user wang@'10.0.0.%' identified by 'magedu';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> exit
```

5.2.3.2 nginx配置

```
[root@centos8 ~]# cat /apps/nginx/conf/tcp/tcp.conf
stream {
    upstream redis_server {
      server 10.0.0.18:6379 max_fails=3 fail_timeout=30s;
      server 10.0.0.28:6379 max_fails=3 fail_timeout=30s;
      #server 10.0.0.28:6379 max_fails=3 fail_timeout=30s backup;
    }

    upstream mysql_server {
     least_conn;
     server 10.0.0.18:3306 max_fails=3 fail_timeout=30s;
    }
########################################################################
  server {
    listen 10.0.0.8:3306;
    proxy_connect_timeout 6s;
    proxy_timeout 15s;
    proxy_pass mysql_server;
    }

  server {
    listen 10.0.0.8:6379;
    proxy_connect_timeout 3s;
```

```
    proxy_timeout 3s;
    proxy_pass redis_server;
    }
}

#重启nginx并访问测试：
[root@centos8 ~]# systemctl  restart nginx

#测试通过nginx负载连接MySQL：
[root@centos8 ~]#mysql -uwang -pmagedu -h10.0.0.8 -e 'show variables like
"hostname"'
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| hostname      | mysql-server1.magedu.org |
+---------------+-------------------------+
[root@centos8 ~]#mysql -uwang -pmagedu -h10.0.0.8 -e 'show variables like
"hostname"'
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| hostname      | mysql-server2.magedu.org |
+---------------+-------------------------+

#在10.0.0.28停止MySQL服务
[root@centos8 ~]#systemctl stop mysqld

#再次测试访问,只会看到mysql-server1.magedu.org进行响应
[root@centos8 ~]#mysql -uwang -pmagedu -h10.0.0.8 -e 'show variables like
"hostname"'
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| hostname      | mysql-server1.magedu.org |
+---------------+-------------------------+
```

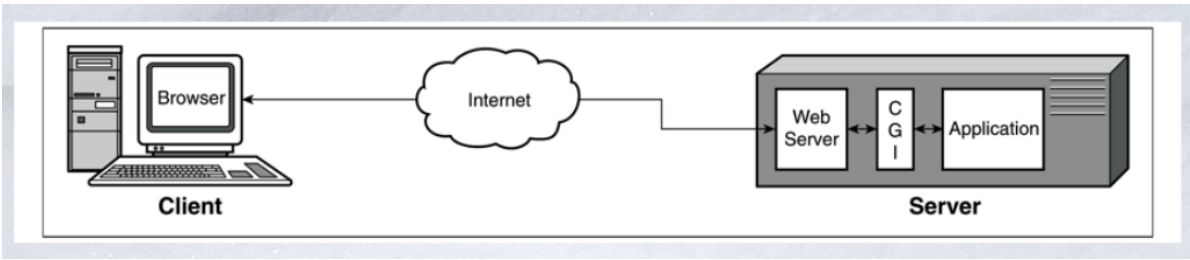
5.2.4 udp 负载均衡实例: DNS

```
stream {
    upstream dns_servers {
        server 10.0.0.7:53;
        server 10.0.0.17:53;
   }
   server {
        listen 53 udp;
        proxy_pass dns_servers;
    }
}
```

5.3 实现 FastCGI

CGI的由来：

最早的Web服务器只能简单地响应浏览器发来的HTTP请求，并将存储在服务器上的HTML文件返回给浏览器，也就是静态html文件，但是后期随着网站功能增多网站开发也越来越复杂，以至于出现动态技术，比如像php(1995年)、java(1995)、python(1991)语言开发的网站，但是nginx/apache服务器并不能直接运行 php、java这样的文件，apache实现的方式是打补丁，但是nginx缺通过与第三方基于协议实现，即通过某种特定协议将客户端请求转发给第三方服务处理，**第三方服务器会新建新的进程处理用户的请求，处理完成后返回数据给Nginx并回收进程**，最后nginx在返回给客户端，那这个约定就是通用网关接口(common gateway interface，简称CGI)，CGI（协议） 是web服务器和外部应用程序之间的接口标准，是cgi程序和web服务器之间传递信息的标准化接口。



为什么会有FastCGI?

CGI协议虽然解决了语言解析器和 Web Server 之间通讯的问题，但是它的效率很低，因为 Web Server 每收到一个请求都会创建一个CGI进程，PHP解析器都会解析php.ini文件，初始化环境，请求结束的时候再关闭进程，对于每一个创建的CGI进程都会执行这些操作，所以效率很低，而FastCGI是用来提高CGI性能的，FastCGI每次处理完请求之后不会关闭掉进程，而是保留这个进程，使这个进程可以处理多个请求。这样的话每个请求都不用再重新创建一个进程了，大大提升了处理效率。

什么是PHP-FPM?

PHP-FPM(FastCGI Process Manager：FastCGI进程管理器)是一个实现了Fastcgi的程序，并且提供进程管理的功能。进程包括master进程和worker进程。master进程只有一个，负责监听端口，接受来自web server的请求。worker进程一般会有多个，每个进程中会嵌入一个PHP解析器，进行PHP代码的处理。

5.3.1 FastCGI配置指令

Nginx基于模块ngx_http_fastcgi_module实现通过fastcgi协议将指定的客户端请求转发至php-fpm处理，其配置指令如下：

```
fastcgi_pass address;
#转发请求到后端服务器，address为后端的fastcgi server的地址，可用位置：location, if in
location
#示例
fastcgi_pass localhost:9000;
fastcgi_pass unix:/tmp/fastcgi.socket;

fastcgi_index name;
#fastcgi默认的主页资源，示例：fastcgi_index index.php;

fastcgi_param parameter value [if_not_empty];
#设置传递给FastCGI服务器的参数值，可以是文本，变量或组合，可用于将Nginx的内置变量赋值给自定义
key
fastcgi_param  REMOTE_ADDR        $remote_addr; #客户端源IP
fastcgi_param  REMOTE_PORT        $remote_port; #客户端源端口
fastcgi_param  SERVER_ADDR        $server_addr; #请求的服务器IP地址
fastcgi_param  SERVER_PORT        $server_port; #请求的服务器端口
fastcgi_param  SERVER_NAME        $server_name; #请求的server name

Nginx默认配置示例：
    location ~ \.php$ {
      root           /scripts;
      fastcgi_pass   127.0.0.1:9000;
      fastcgi_index  index.php;
      fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name; #默认脚
本路径
      #fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;  #此行写法不再
需要上面的 root 指令
      include        fastcgi_params;    #此文件默认系统已提供,存放的相对路径为
prefix/conf
    }
```

fastcgi 缓存定义指令：注意使用fastcgi缓存, 可能会导致源代码更新失败,生产慎用

```
fastcgi_cache_path path [levels=levels] [use_temp_path=on|off]
keys_zone=name:size [inactive=time] [max_size=size] [manager_files=number]
[manager_sleep=time] [manager_threshold=time] [loader_files=number]
[loader_sleep=time] [loader_threshold=time] [purger=on|off]
[purger_files=number] [purger_sleep=time] [purger_threshold=time];
#定义fastcgi的缓存;
    path   #缓存位置为磁盘上的文件系统路径
    max_size=size #磁盘path路径中用于缓存数据的缓存空间上限
    levels=levels：缓存目录的层级数量，以及每一级的目录数量，levels=ONE:TWO:THREE，示例：
leves=1:2:2
    keys_zone=name:size #设置缓存名称及k/v映射的内存空间的名称及大小
    inactive=time #缓存有效时间，默认10分钟，需要在指定时间满足fastcgi_cache_min_uses 次
数被视为活动缓存。
```

缓存调用指令：

```
fastcgi_cache zone | off;
```

```
#调用指定的缓存空间来缓存数据，可用位置：http, server, location

fastcgi_cache_key string;
#定义用作缓存项的key的字符串，示例：fastcgi_cache_key $request_uri;

fastcgi_cache_methods GET | HEAD | POST ...;
#为哪些请求方法使用缓存

fastcgi_cache_min_uses number;
#缓存空间中的缓存项在inactive定义的非活动时间内至少要被访问到此处所指定的次数方可被认作活动项

fastcgi_keep_conn on | off;
#收到后端服务器响应后，fastcgi服务器是否关闭连接，建议启用长连接

fastcgi_cache_valid [code ...] time;
#不同的响应码各自的缓存时长

fastcgi_hide_header field; #隐藏响应头指定信息
fastcgi_pass_header field; #返回响应头指定信息，默认不会将Status、X-Accel-...返回
```

5.3.2 FastCGI实战案例：Nginx与php-fpm在同一服务器

php安装可以通过yum或者编译安装，使用yum安装相对比较简单，编译安装更方便自定义参数或选项。

5.3.2.1 php 环境准备

使用base源自带的php版本

```
#yum安装默认版本php和相关APP依赖的包
[root@centos8 ~]# yum -y install php-fpm  php-mysqlnd php-json #默认版本
#或者安装清华的php源
[root@centos7 ~]# yum -y install
https://mirrors.tuna.tsinghua.edu.cn/remi/enterprise/remi-release-7.rpm

[root@centos8 ~]# systemctl  enable --now php-fpm
[root@centos8 ~]# ps -ef | grep php-fpm
root       4925      1  0 17:13 ?        00:00:00 php-fpm: master process
(/etc/php-fpm.conf)
apache     4927   4925  0 17:13 ?        00:00:00 php-fpm: pool www
apache     4928   4925  0 17:13 ?        00:00:00 php-fpm: pool www
apache     4929   4925  0 17:13 ?        00:00:00 php-fpm: pool www
apache     4930   4925  0 17:13 ?        00:00:00 php-fpm: pool www
apache     4931   4925  0 17:13 ?        00:00:00 php-fpm: pool www
root       4933   3235  0 17:13 pts/0    00:00:00 grep --color=auto php-fpm
```

5.3.2.2 php相关配置优化

```
[root@centos8 ~]# grep "^[a-Z]" /etc/php-fpm.conf
include=/etc/php-fpm.d/*.conf
pid = /run/php-fpm/php-fpm.pid
error_log = /var/log/php-fpm/error.log
daemonize = yes #是否后台启动

[root@centos8 ~]#grep -Ev '^;.*$|^ *$' /etc/php-fpm.d/www.conf
[www]
user = nginx
group = nginx
listen = /run/php-fpm/www.sock    #指定使用UDS,或者使用下面形式
;listen = 127.0.0.1:9000  #监听地址及IP
listen.acl_users = apache,nginx
listen.allowed_clients = 127.0.0.1
pm = dynamic
pm.max_children = 50
pm.start_servers = 5
pm.min_spare_servers = 5
pm.max_spare_servers = 35
pm.status_path = /pm_status  #修改此行
ping.path = /ping            #修改此行
ping.response = ping-pong    #修改此行
slowlog = /var/log/php-fpm/www-slow.log  #慢日志路径
php_admin_value[error_log] = /var/log/php-fpm/www-error.log #错误日志
php_admin_flag[log_errors] = on
php_value[session.save_handler] = files  #phpsession保存方式及路径
php_value[session.save_path]    = /var/lib/php/session #当时使用file保存session的文
件路径
php_value[soap.wsdl_cache_dir]  = /var/lib/php/wsdlcache
php_value[upload_max_filesize] = 20m
php_value[post_max_size]  = 20m
php_value[date.timezone]  = Asia/Shanghai


#修改配置文件后记得重启php-fpm
[root@centos8 ~]# systemctl  restart php-fpm
```

5.3.2.3 准备php测试页面

```
[root@centos8 ~]# mkdir -p  /data/php
[root@centos8 ~]# cat /data/php/index.php #php测试页面
<?php
phpinfo();
?>
```

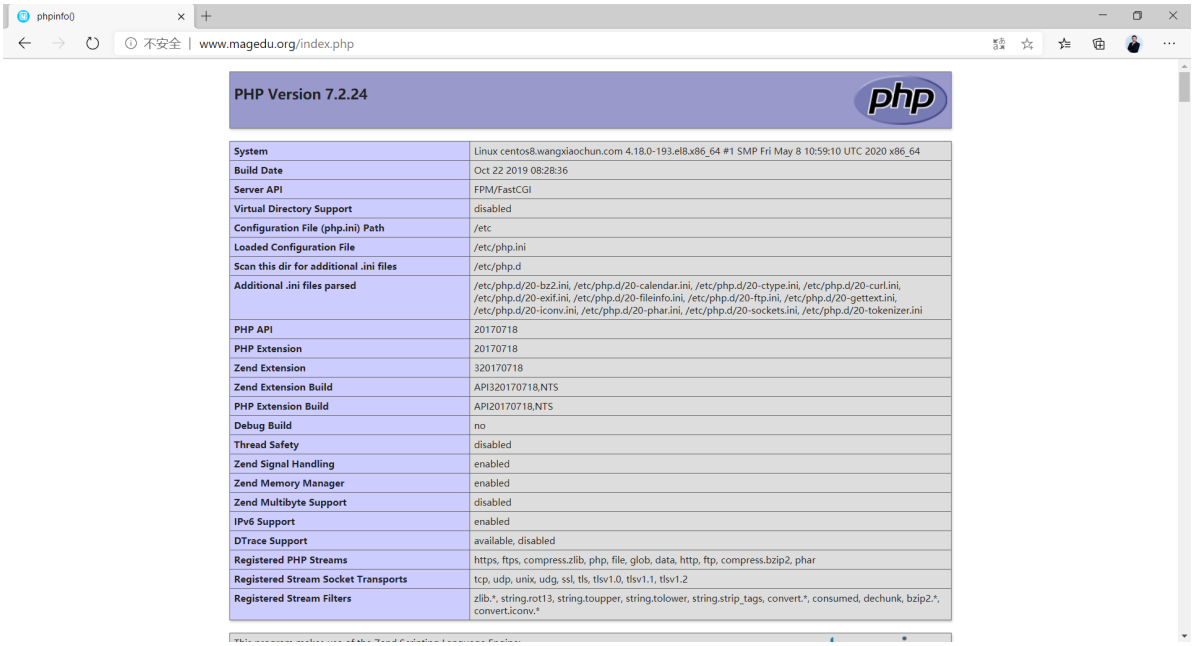
5.3.2.4 Nginx配置转发

Nginx安装完成之后默认生成了与fastcgi的相关配置文件，一般保存在nginx的安装路径的conf目录当中，比如/apps/nginx/conf/fastcgi.conf、/apps/nginx/conf/fastcgi_params。

```
[root@centos8 ~]# vim /apps/nginx/conf/conf.d/pc.conf #在指定文件配置fastcgi
server {
  listen 80;
  server_name www.magedu.org;
```

```
    index index.php index.html;
    location ~ \.php$|pm_status|ping {
        root           /data/php; #下面的$document_root调用此行的root指令指定的目录
        fastcgi_pass   unix:/run/php-fpm/www.sock;
        #fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        #fastcgi_param SCRIPT_FILENAME  /data/php$fastcgi_script_name;
        #如果SCRIPT_FILENAME是上面的绝对路径则可以省略root /data/php;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        include        fastcgi_params;
        fastcgi_param HTTPS $https if_not_empty;     #有些应用支持https需要此项
    }
}
#重启Nginx并访问web测试
[root@centos8 ~]# systemctl  restart nginx
```

5.3.2.5 访问验证php测试页面



System	Linux centos8.wangxiaochun.com 4.18.0-193.el8.x86_64 #1 SMP Fri May 8 10:59:10 UTC 2020 x86_64
Build Date	Oct 22 2019 08:28:36
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

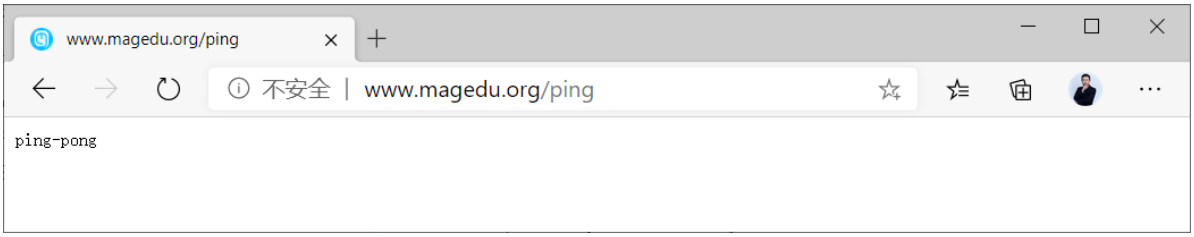
```
#常见的错误：
File not found.  #路径不对
502  #php-fpm处理超时、服务停止运行等原因导致的无法连接或请求超时
```

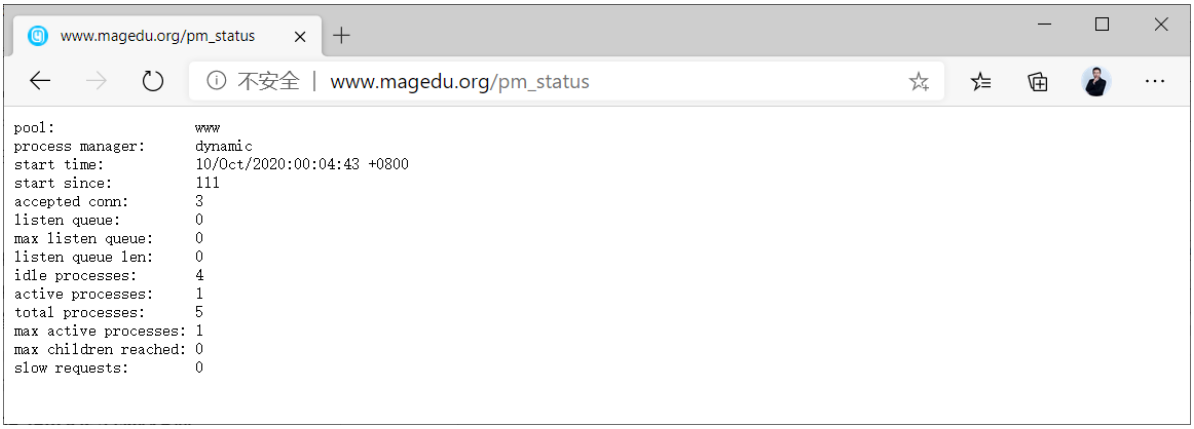
5.3.2.6 php-fpm 的运行状态页面

访问配置文件里面指定的路径，会返回php-fpm的当前运行状态。

Nginx配置：

```
location ~ ^/(ping|pm_status)$ {
    fastcgi_pass unix:/run/php-fpm/www.sock;
    #fastcgi_pass 127.0.0.1:9000;
    fastcgi_param PATH_TRANSLATED $document_root$fastcgi_script_name;
    include fastcgi_params;
}

#重启Nginx并测试：
```

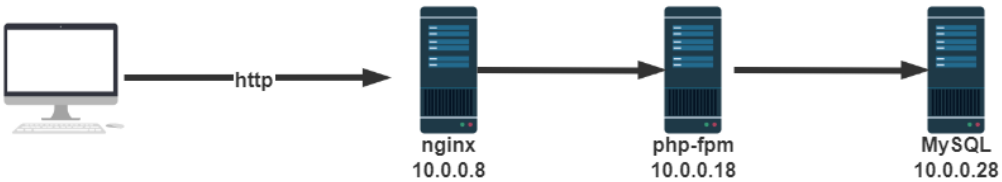





```
[root@centos8 ~]#curl  http://www.magedu.org/ping
ping-pong
[root@centos8 ~]#curl  http://www.magedu.org/pm_status
pool:                 www
process manager:      dynamic
start time:           10/Oct/2020:00:04:43 +0800
start since:          155
accepted conn:        5
listen queue:         0
max listen queue:     0
listen queue len:     0
idle processes:       4
active processes:     1
total processes:      5
max active processes: 1
max children reached: 0
slow requests:        0

[root@centos8 ~]# curl  http://www.magedu.org/pm_status?full
[root@centos8 ~]# curl  http://www.magedu.org/pm_status?html
[root@centos8 ~]# curl  http://www.magedu.org/pm_status?json
```

5.3.3 FastCGI实战案例：Nginx与php不在同一个服务器

nginx会处理静态请求，但是会转发动态请求到后端指定的php-fpm服务器，因此php代码需要放在后端的php-fpm服务器，即静态页面放在Nginx服务器上,而动态页面放在后端php-fpm服务器，通常情况下，一般都是采用在同一个服务器



5.3.3.1 yum安装较新版本php-fpm

```
[root@centos8 ~]#dnf -y install php-fpm php-mysqlnd php-json

#验证安装路径:
[root@centos8 ~]#rpm -ql  php-fpm
/etc/httpd/conf.d/php.conf
/etc/logrotate.d/php-fpm
/etc/nginx/conf.d/php-fpm.conf
/etc/nginx/default.d/php.conf
/etc/php-fpm.conf
/etc/php-fpm.d
/etc/php-fpm.d/www.conf
/etc/systemd/system/php-fpm.service.d
/run/php-fpm
/usr/lib/.build-id
/usr/lib/.build-id/bf
/usr/lib/.build-id/bf/8d6b2bca109fb20f8037d72220670f2d6cc954
/usr/lib/systemd/system/httpd.service.d/php-fpm.conf
/usr/lib/systemd/system/nginx.service.d/php-fpm.conf
/usr/lib/systemd/system/php-fpm.service
/usr/sbin/php-fpm
/usr/share/doc/php-fpm
/usr/share/doc/php-fpm/php-fpm.conf.default
/usr/share/doc/php-fpm/www.conf.default
/usr/share/fpm
/usr/share/fpm/status.html
/usr/share/licenses/php-fpm
/usr/share/licenses/php-fpm/fpm_LICENSE
/usr/share/man/man8/php-fpm.8.gz
/var/lib/php/opcache
/var/lib/php/session
/var/lib/php/wsdlcache
/var/log/php-fpm
```

5.3.3.2 修改php-fpm监听配置

php-fpm默认监听在127.0.0.1的9000端口，也就是无法远程连接，因此要做相应的修改。

```
#修改监听配置
[root@centos8 ~]#vim /etc/php-fpm.d/www.conf
;listen = /run/php-fpm/www.sock #注释此行
listen = 9000  #修改此行,指定监听端口
;listen.allowed_clients = 127.0.0.1  #注释此行
```

5.3.3.3 准备php测试页面

```
#准备php数据目录
[root@centos8 ~]# mkdir -p /data/php
[root@centos8 ~]# vim /data/php/index.php
<?php
  phpinfo();
?>
```

5.3.3.4 启动并验证php-fpm

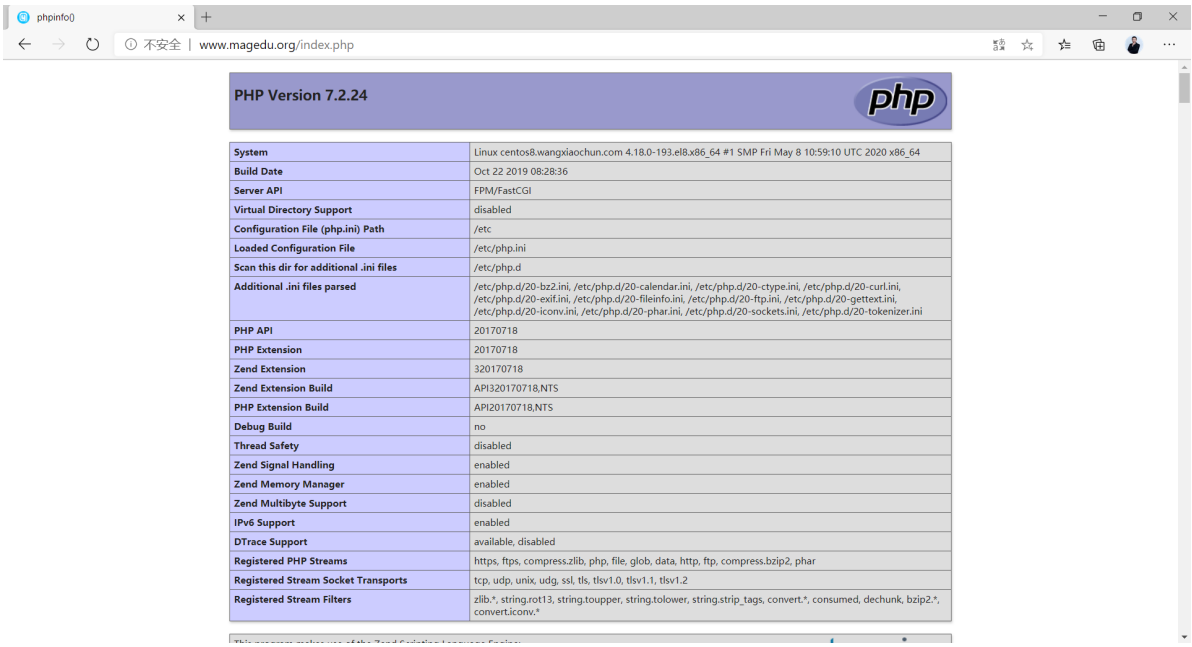
```
#启动php-fpm
[root@centos8 ~]#systemctl enable --now php-fpm.service

#验证php-fpm进程及端口：
[root@centos8 ~]#ps -ef | grep php-fpm
root        1506       1  0 23:24 ?        00:00:00 php-fpm: master process
(/etc/php-fpm.conf)
apache      1507    1506  0 23:24 ?        00:00:00 php-fpm: pool www
apache      1508    1506  0 23:24 ?        00:00:00 php-fpm: pool www
apache      1509    1506  0 23:24 ?        00:00:00 php-fpm: pool www
apache      1510    1506  0 23:24 ?        00:00:00 php-fpm: pool www
apache      1511    1506  0 23:24 ?        00:00:00 php-fpm: pool www
root        1517    1102  0 23:25 pts/0    00:00:00 grep --color=auto php-fpm

[root@centos8 ~]#ss -tnl
State             Recv-Q            Send-Q    Local Address:Port  Peer
Address:Port
LISTEN            0                 100           127.0.0.1:25         0.0.0.0:*

LISTEN            0                 128             0.0.0.0:22         0.0.0.0:*

LISTEN            0                 100               [::1]:25            [::]:*

LISTEN            0                 128                   *:9000             *:*

LISTEN            0                 128                [::]:22            [::]:*


[root@centos8 ~]#lsof -i :9000
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
php-fpm 1506   root    9u  IPv6  33898      0t0  TCP *:cslistener (LISTEN)
php-fpm 1507 apache   11u  IPv6  33898      0t0  TCP *:cslistener (LISTEN)
php-fpm 1508 apache   11u  IPv6  33898      0t0  TCP *:cslistener (LISTEN)
php-fpm 1509 apache   11u  IPv6  33898      0t0  TCP *:cslistener (LISTEN)
php-fpm 1510 apache   11u  IPv6  33898      0t0  TCP *:cslistener (LISTEN)
php-fpm 1511 apache   11u  IPv6  33898      0t0  TCP *:cslistener (LISTEN)
```

5.3.3.5 Nginx配置转发

```
[root@centos8 ~]#vi /apps/nginx/conf/conf.d/pc.conf
 location ~ \.php$ {
    root           /data/php;
    fastcgi_pass   10.0.0.18:9000;
    fastcgi_index  index.php;
    #fastcgi_param SCRIPT_FILENAME  /data/php$fastcgi_script_name;
    fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    include        fastcgi_params;
  }
  location ~ ^/(ping|pm_status)$ {
    fastcgi_pass 10.0.0.18:9000;
    fastcgi_param PATH_TRANSLATED $document_root$fastcgi_script_name;
    include fastcgi_params;
  }
```

```
#重启nginx
[root@centos8 ~]# systemctl  restart nginx
```

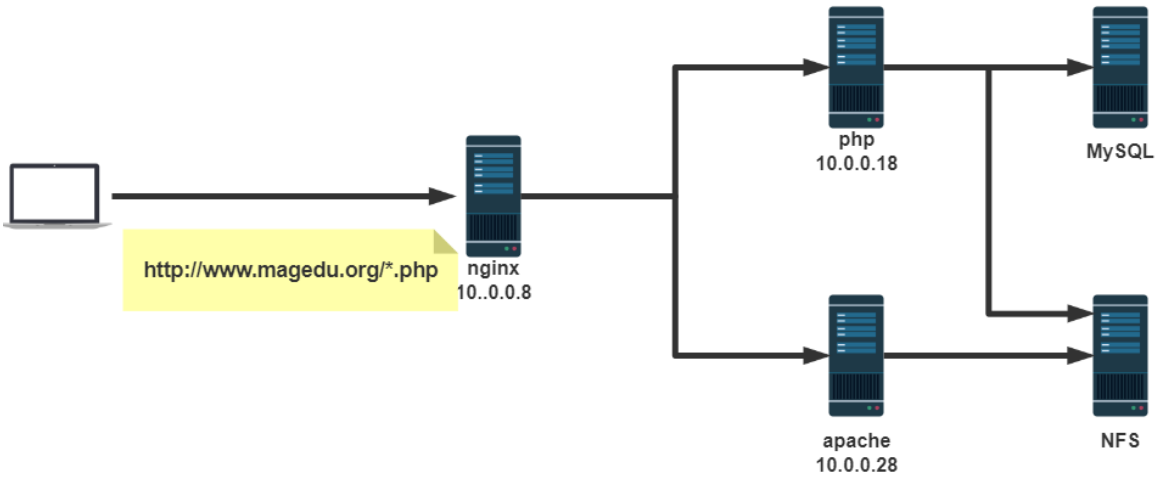
5.3.3.6 访问验证php测试页面



System	Linux centos8.wangxiaochun.com 4.18.0-193.el8.x86_64 #1 SMP Fri May 8 10:59:10 UTC 2020 x86_64
Build Date	Oct 22 2019 08:28:36
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

5.3.4 实现动静分离

要求：将客户端对除php以外的资源的访问转发至后端服务器 10.0.0.28上

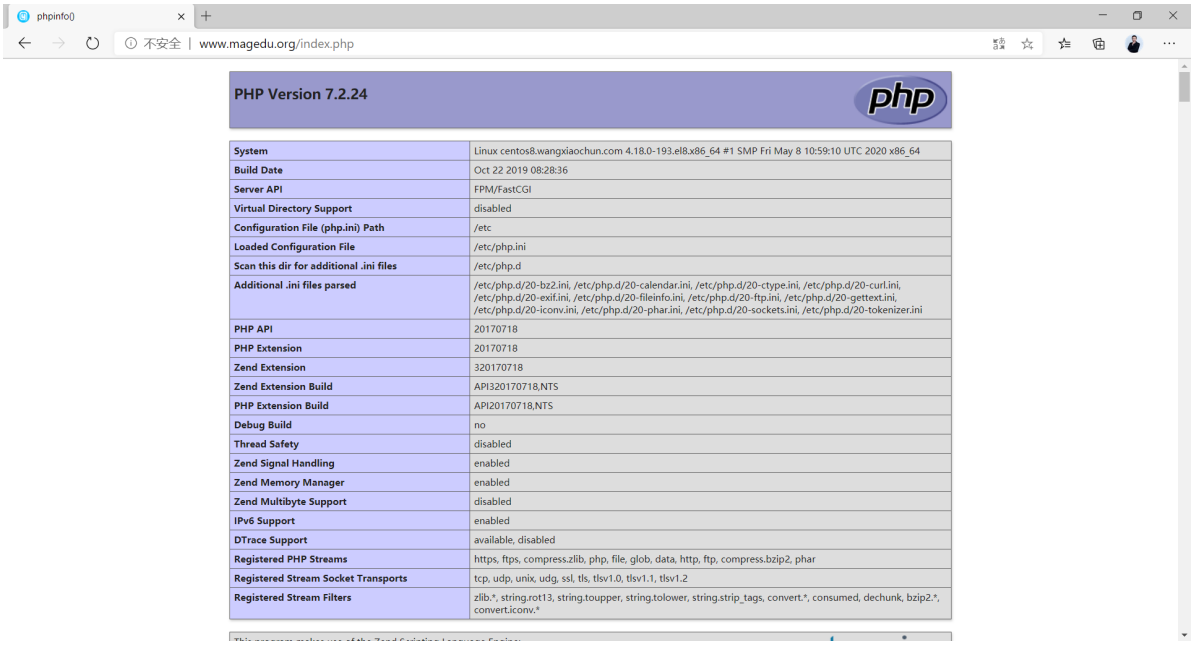


5.3.4.1 配置 nginx 实现反向代理的动静分离

```
[root@centos8 ~]#vi /apps/nginx/conf/conf.d/pc.conf
  location / {
    proxy_pass http://10.0.0.28;
    index index.html;
  }
  location ~ \.php$ {
    root /data/php;
    fastcgi_pass   10.0.0.18:9000;
    fastcgi_index  index.php;
    #fastcgi_param SCRIPT_FILENAME  /data/php$fastcgi_script_name;
    fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
    include        fastcgi_params;
  }
```

5.3.4.2 准备后端 httpd 服务器

```
#在后端服务器10.0.0.28上安装httpd服务
[root@centos8 ~]#dnf -y install httpd
[root@centos8 ~]#systemctl enable --now httpd
[root@centos8 ~]#mkdir /var/www/html/images
[root@centos8 ~]#wget -O /var/www/html/images/magedu.jpg
http://www.magedu.com/wp-content/uploads/2019/05/2019052306372726.jpg
```

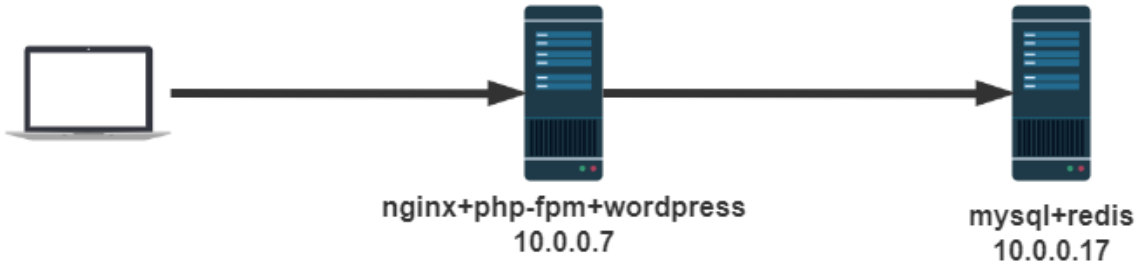


System	Linux centos8.wangxiaochun.com 4.18.0-193.el8.x86_64 #1 SMP Fri May 8 10:59:10 UTC 2020 x86_64
Build Date	Oct 22 2019 08:28:36
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

5.3.5 项目实战：利用 LNMP 实现WordPress站点搭建

LNMP项目实战环境说明

```
L：Linux（CentOS7）https://mirrors.aliyun.com/centos/7/isos/x86_64/
N：Nginx（1.18.0） https://nginx.org/en/download.html
M：MySQL（8.0.19） https://dev.mysql.com/downloads/mysql/
P：PHP（7.4.10） http://php.net/downloads.php
Wordpress（5.4.2）：https://cn.wordpress.org/download/
#部署规划：
10.0.0.7：Nginx  php-fpm 运行web服务
10.0.0.17：运行MySQL数据库,Redis服务
```



5.3.5.1 部署数据库

在10.0.0.17主机部署MySQL服务

5.3.5.1.1 二进制部署MySQL数据库

```
[root@centos7 ~]#ll
total 473728
-rw-r--r--  1 root root      2433 Sep  8 23:09
install_mysql5.7or8.0_for_centos.sh
-rw-r--r--  1 root root 485074552 Aug 22 23:40 mysql-8.0.19-linux-glibc2.12-
x86_64.tar.xz

#一键安装脚本
[root@centos7 ~]#cat install_mysql5.7or8.0_for_centos.sh
#!/bin/bash
```

```
#
#**********************************************************************
#Author:        wangxiaochun
#QQ:            29308620
#Date:          2020-02-12
#FileName:       install_mysql5.7_for_centos.sh
#URL:           http://www.magedu.com
#Description:        The test script
#Copyright (C):     2020 All rights reserved
#**********************************************************************
#MySQL Download URL: https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-5.7.29-linux-glibc2.12-x86_64.tar.gz

. /etc/init.d/functions
SRC_DIR=`pwd`
MYSQL='mysql-8.0.19-linux-glibc2.12-x86_64.tar.xz'
COLOR='echo -e \E[01;31m'
END='\E[0m'
MYSQL_ROOT_PASSWORD=magedu


check (){

if [ $UID -ne 0 ]; then
  action "当前用户不是root,安装失败" false
  exit 1
fi

cd  $SRC_DIR
if [ !  -e $MYSQL ];then
        $COLOR"缺少${MYSQL}文件"$END
        $COLOR"请将相关软件放在${SRC_DIR}目录下"$END
        exit
elif [ -e /usr/local/mysql ];then
        action "数据库已存在，安装失败" false
        exit
else
    return
fi
}

install_mysql(){
    $COLOR"开始安装MySQL数据库..."$END
    yum  -y -q install libaio numactl-libs   libaio &> /dev/null
    cd $SRC_DIR
    tar xf $MYSQL -C /usr/local/
    MYSQL_DIR=`echo $MYSQL| sed -nr 's/^(.*[0-9]).*/\1/p'`
    ln -s  /usr/local/$MYSQL_DIR /usr/local/mysql
    chown -R  root.root /usr/local/mysql/
    id mysql &> /dev/null || { useradd -s /sbin/nologin -r  mysql ; action "创建mysql用户"; }

    echo 'PATH=/usr/local/mysql/bin/:$PATH' > /etc/profile.d/mysql.sh
    .  /etc/profile.d/mysql.sh
    ln -s /usr/local/mysql/bin/* /usr/bin/
    cat > /etc/my.cnf <<-EOF
[mysqld]
server-id=`hostname -I|cut -d. -f4`
```

```
log-bin
datadir=/data/mysql
socket=/data/mysql/mysql.sock

log-error=/data/mysql/mysql.log
pid-file=/data/mysql/mysql.pid
[client]
socket=/data/mysql/mysql.sock
EOF
    mysqld --initialize --user=mysql --datadir=/data/mysql
    cp /usr/local/mysql/support-files/mysql.server  /etc/init.d/mysqld
    chkconfig --add mysqld
    chkconfig mysqld on
    service mysqld start
    [ $? -ne 0 ] && { $COLOR"数据库启动失败，退出!"$END;exit; }
    MYSQL_OLDPASSWORD=`awk '/A temporary password/{print $NF}'
/data/mysql/mysql.log`
    mysqladmin  -uroot -p$MYSQL_OLDPASSWORD password $MYSQL_ROOT_PASSWORD
&>/dev/null
    action "数据库安装完成"
}


check

install_mysql

#运行脚本安装数据库
```

```
[root@centos7 ~]#bash install_mysql5.7or8.0_for_centos.sh
开始安装MySQL数据库...
创建mysql用户                                              [  OK  ]
Starting MySQL..                                           [  OK  ]
数据库安装完成                                             [  OK  ]
```

5.3.5.1.2 创建wordpress数据库和用户并授权

```
[root@centos7 ~]#mysql -uroot -pmagedu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database wordpress;
Query OK, 1 row affected (0.01 sec)

mysql> create user wordpress@'10.0.0.%' identified by '123456';
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> grant all on wordpress.* to wordpress@'10.0.0.%';
Query OK, 0 rows affected (0.01 sec)
```

5.3.5.1.3 验证MySQL账户权限

在WordPress服务器使用授权的MySQL账户远程登录测试权限

```
[root@centos7 ~]#mysql -uwordpress -p123456 -h10.0.0.17
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| wordpress          |
+--------------------+
2 rows in set (0.00 sec)
```

5.3.5.2 部署PHP

在10.0.0.7主机部署php-fpm服务

5.3.5.2.1 编译安装 php

```
[root@centos7 ~]#yum -y install gcc openssl-devel libxml2-devel bzip2-devel
libmcrypt-devel sqlite-devel oniguruma-devel
[root@centos7 ~]#cd /usr/local/src
[root@centos7 src]#wget https://www.php.net/distributions/php-7.4.11.tar.xz
[root@centos7 src]#ll -h  php-7.4.11.tar.xz
-rw-r--r-- 1 root root 9.9M Sep 29 18:40 php-7.4.11.tar.xz


[root@centos7 php-7.4.11]#./configure --prefix=/apps/php74 --enable-mysqlnd --
with-mysqli=mysqlnd --with-pdo-mysql=mysqlnd --with-openssl    --with-zlib  --
with-config-file-path=/etc --with-config-file-scan-dir=/etc/php.d --enable-
mbstring --enable-xml --enable-sockets --enable-fpm --enable-maintainer-zts --
disable-fileinfo
......
config.status: creating main/php_config.h
config.status: executing default commands

+--------------------------------------------------------------------+
| License:                                                           |
| This software is subject to the PHP License, available in this     |
```

```
| distribution in the file LICENSE. By continuing this installation  |
| process, you are bound by the terms of this license agreement.     |
| If you do not agree with the terms of this license, you must abort |
| the installation process at this point.                            |
+--------------------------------------------------------------------+

Thank you for using PHP.

[root@centos7 php-7.4.11]#make -j 8 && make install
```

5.3.5.2.2 准备 php 配置文件

```
#生成配置文件
[root@centos7 php-7.4.11]#cp /usr/local/src/php-7.4.11/php.ini-production
/etc/php.ini

[root@centos7 php-7.4.11]#cd /apps/php74/etc
[root@centos7 etc]#cp php-fpm.conf.default php-fpm.conf
[root@centos7 php-7.4.11]#cd php-fpm.d/
[root@centos7 php-fpm.d]#cp www.conf.default www.conf

[root@centos7 php-fpm.d]#vim www.conf
[root@centos7 php-fpm.d]#grep '^[^;]' www.conf
[www]
user = www
group = www
listen = 127.0.0.1:9000
pm = dynamic
pm.max_children = 5
pm.start_servers = 2
pm.min_spare_servers = 1
pm.max_spare_servers = 3
pm.status_path = /pm_status
ping.path = /ping
access.log = log/$pool.access.log
slowlog = log/$pool.log.slow

#创建用户
[root@centos7 php-fpm.d]#useradd -r -s /sbin/nologin www

#创建访问日志文件路径
[root@centos7 php-fpm.d]#mkdir /apps/php74/log
```

5.3.5.2.3 启动并验证 php-fpm 服务

```
[root@centos7 ~]#/apps/php74/sbin/php-fpm -t
[22-Oct-2020 10:55:19] NOTICE: configuration file /apps/php74/etc/php-fpm.conf
test is successful

[root@centos7 ~]#cp /usr/local/src/php-7.4.11/sapi/fpm/php-fpm.service
/usr/lib/systemd/system/

[root@centos7 ~]#systemctl daemon-reload
[root@centos7 ~]#systemctl enable --now php-fpm
Created symlink from /etc/systemd/system/multi-user.target.wants/php-fpm.service
to /usr/lib/systemd/system/php-fpm.service.
[root@centos7 php-7.4.11]#ss -ntl
```

```
State      Recv-Q Send-Q                     Local Address:Port
                  Peer Address:Port
LISTEN     0      128                              *:22
                             *:*
LISTEN     0      100                          127.0.0.1:25
                             *:*
LISTEN     0      128                          127.0.0.1:9000
                             *:*
LISTEN     0      128                               [::]:22
                           [::]:*
LISTEN     0      100                              [::1]:25
                           [::]:*

[root@centos7 ~]#pstree -p |grep php
           |-php-fpm(16331)-+-php-fpm(16332)
           |                 `-php-fpm(16333)
[root@centos7 ~]#ps -ef |grep php
root      16331      1  0 12:46 ?        00:00:00 php-fpm: master process
(/apps/php74/etc/php-fpm.conf)
www       16332  16331  0 12:46 ?        00:00:00 php-fpm: pool www
www       16333  16331  0 12:46 ?        00:00:00 php-fpm: pool www
root      16403   1095  0 13:12 pts/0    00:00:00 grep --color=auto php
```

5.3.5.3 部署 Nginx

在10.0.0.7主机部署nginx服务

5.3.5.3.1 编译安装 nginx

```
[root@centos7 ~]#yum -y install gcc pcre-devel openssl-devel zlib-devel
[root@centos7 ~]#cd /usr/local/src/
[root@centos7 src]#wget http://nginx.org/download/nginx-1.18.0.tar.gz
[root@centos7 src]#tar xf nginx-1.18.0.tar.gz
[root@centos7 src]#cd nginx-1.18.0/
[root@centos7 nginx-1.18.0]#./configure --prefix=/apps/nginx \
--user=www  \
--group=www \
--with-http_ssl_module \
--with-http_v2_module \
--with-http_realip_module \
--with-http_stub_status_module  \
--with-http_gzip_static_module \
--with-pcre \
--with-stream \
--with-stream_ssl_module \
--with-stream_realip_module

[root@centos7 nginx-1.18.0]#make && make install
```

5.3.5.3.2 准备服务文件并启动 nginx

```
[root@centos8 ~]#vim  /usr/lib/systemd/system/nginx.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
```

```
Wants=network-online.target

[Service]
Type=forking
PIDFile=/apps/nginx/run/nginx.pid
ExecStart=/apps/nginx/sbin/nginx -c /apps/nginx/conf/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=multi-user.target

#创建目录
[root@centos8 ~]#mkdir /apps/nginx/run/

#修改配置文件
[root@centos8 ~]#vim /apps/nginx/conf/nginx.conf
pid   /apps/nginx/run/nginx.pid;

[root@centos7 ~]#systemctl daemon-reload
[root@centos7 ~]#systemctl enable --now  nginx
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to
/usr/lib/systemd/system/nginx.service.

[root@centos7 ~]#ss -ntl
State      Recv-Q Send-Q       Local Address:Port    Peer Address:Port

LISTEN     0      128                      *:22                 *:*
LISTEN     0      100              127.0.0.1:25                 *:*
LISTEN     0      128              127.0.0.1:9000               *:*
LISTEN     0      128                      *:80                 *:*
LISTEN     0      128                   [::]:22              [::]:*
LISTEN     0      100                  [::1]:25              [::]:*
```

5.3.5.3.3 配置 Nginx 支持 fastcgi

```
[root@centos7 ~]#vim /apps/nginx/conf/nginx.conf
[root@centos7 ~]#grep -Ev '#|^$' /apps/nginx/conf/nginx.conf
worker_processes  1;
pid        /apps/nginx/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    server {
        listen       80;
        server_name  www.magedu.org;  #指定主机名
        location / {
            root   /data/nginx/wordpress;  #指定数据目录
            index  index.php index.html index.htm;          #指定默认主页
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
```

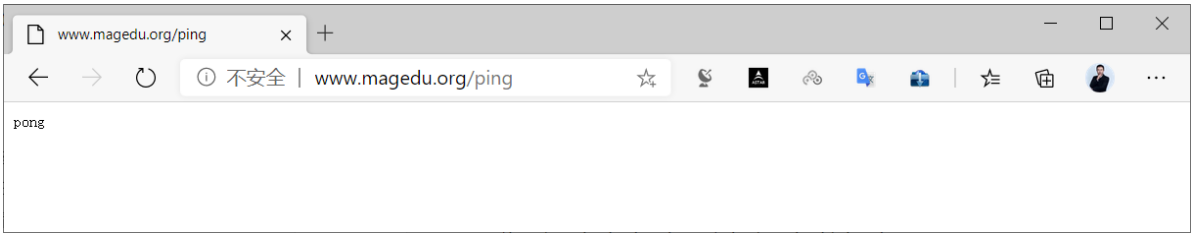
```
            root   html;
        }
        location ~ \.php$ {                          #实现php-fpm
            root           /data/nginx/wordpress;
            fastcgi_pass   127.0.0.1:9000;
            fastcgi_index  index.php;
            fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
            include        fastcgi_params;
        }
        location ~ ^/(ping|pm_status)$ {             #实现状态页
            include fastcgi_params;
            fastcgi_pass 127.0.0.1:9000;
            fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name;
            #fastcgi_param PATH_TRANSLATED $document_root$fastcgi_script_name; 此
配置也可以
        }
    }
}
[root@centos7 ~]# /apps/nginx/sbin/nginx -t
nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
[root@centos7 ~]#systemctl reload nginx
```

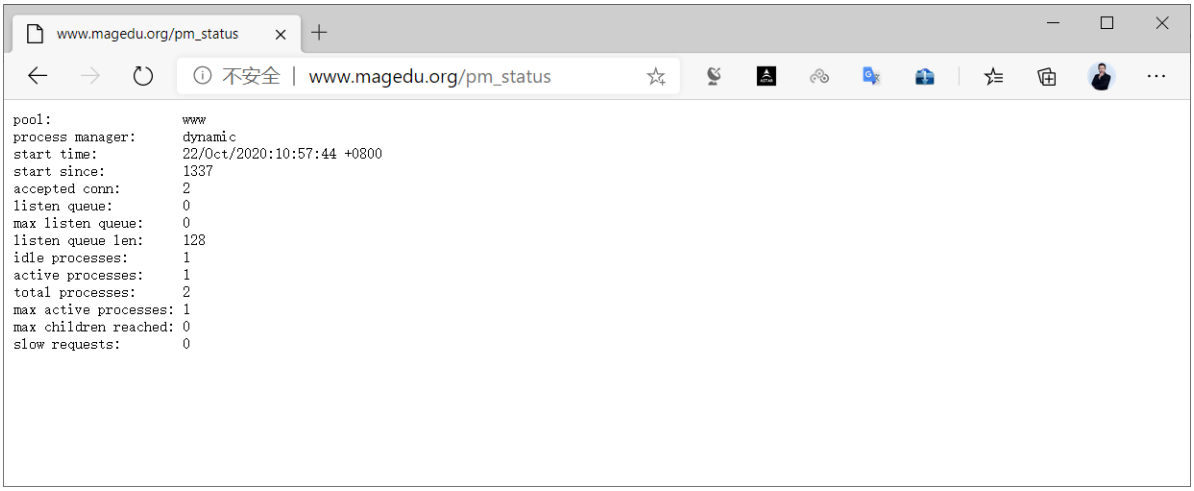
5.3.5.3.4 准备 php 测试页

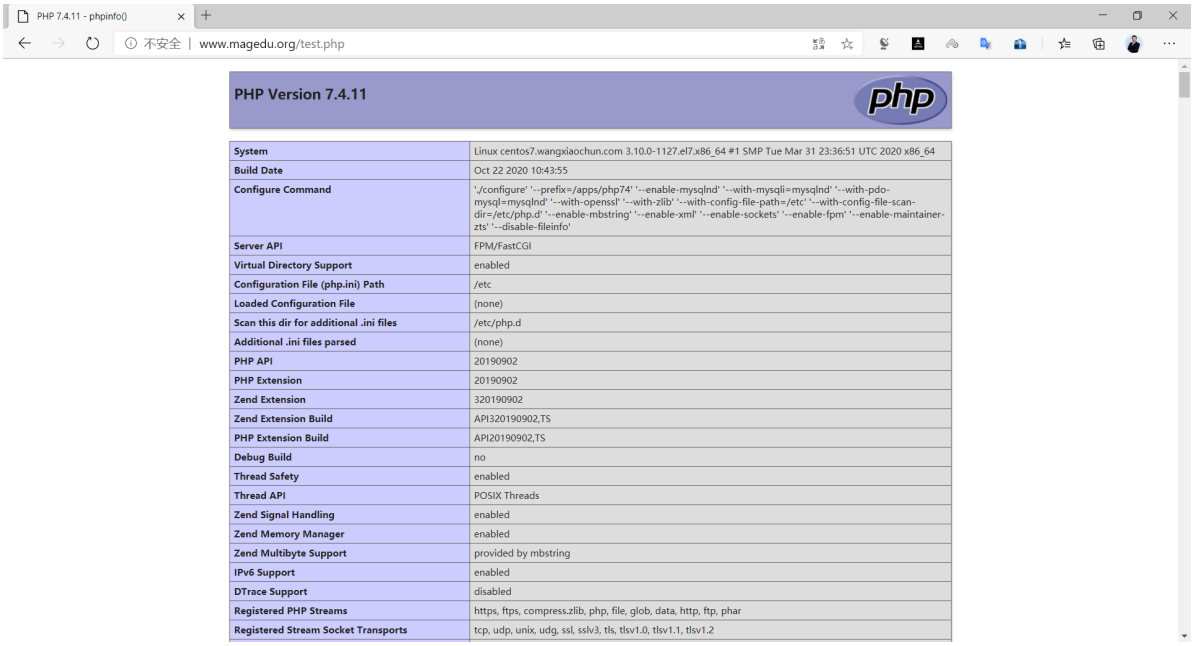
准备测试页

```
[root@centos7 ~]#mkdir -p /data/nginx/wordpress
[root@centos7 ~]#vim /data/nginx/wordpress/test.php
[root@centos7 ~]#cat /data/nginx/wordpress/test.php
<?php
phpinfo();
?>
```

5.3.5.3.5 验证 php 测试页







System	Linux centos7.wangxiaochun.com 3.10.0-1127.el7.x86_64 #1 SMP Tue Mar 31 23:36:51 UTC 2020 x86_64
Build Date	Oct 22 2020 10:43:55
Configure Command	'./configure' '--prefix=/apps/php74' '--enable-mysqlnd' '--with-mysqli=mysqlnd' '--with-pdo-mysql=mysqlnd' '--with-openssl' '--with-zlib' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--enable-mbstring' '--enable-xml' '--enable-sockets' '--enable-fpm' '--enable-maintainer-zts' '--disable-fileinfo'
Server API	FPM/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,TS
PHP Extension Build	API20190902,TS
Debug Build	no
Thread Safety	enabled
Thread API	POSIX Threads
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2

5.3.5.4 部署 WordPress

在10.0.0.7主机部署 wordpress

5.3.5.4.1 准备 WordPress 文件

```
[root@centos7 ~]#wget https://cn.wordpress.org/latest-zh_CN.tar.gz
#[root@centos7 ~]#tar xf wordpress-5.4.2-zh_CN.tar.gz
[root@centos7 ~]#tar xf latest-zh_CN.tar.gz
[root@centos7 ~]#mv  wordpress/* /data/nginx/wordpress
[root@centos7 ~]#chown -R www.www /data/nginx/wordpress/
```

5.3.5.4.2 初始化web页面

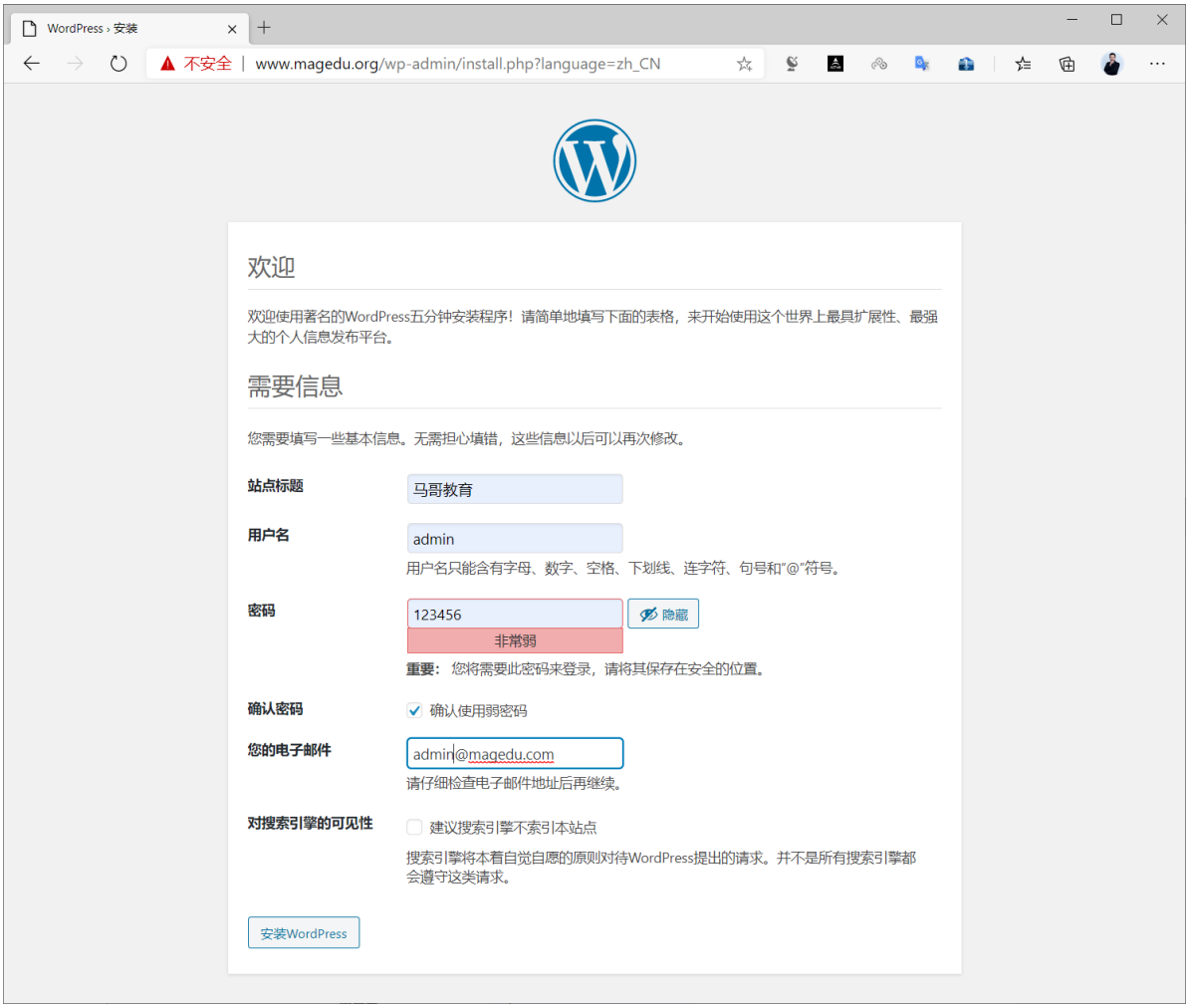
打开浏览器访问下面链接

```
http://www.magedu.org/
```

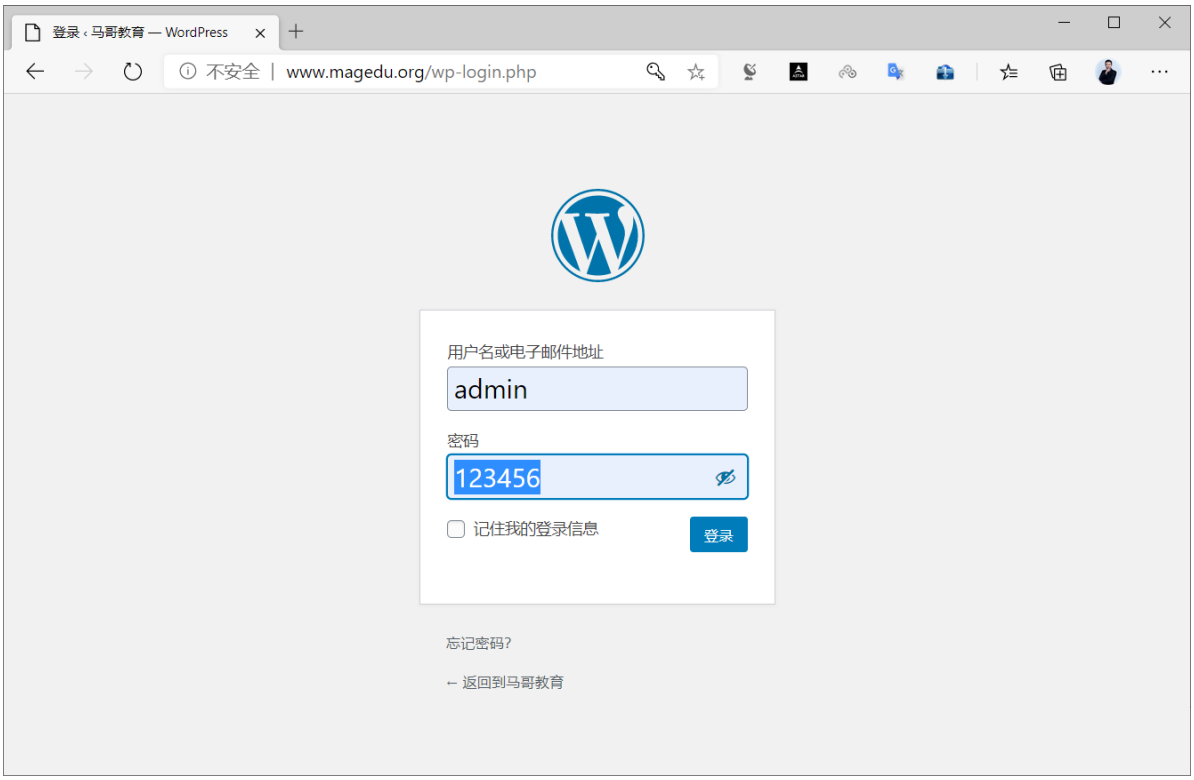

WordPress › 调整配置文件
不安全 | www.magedu.org/wp-admin/setup-config.php
欢迎使用WordPress。在开始前，我们需要您数据库的一些信息。请准备好如下信息。
1. 数据库名
2. 数据库用户名
3. 数据库密码
4. 数据库主机
5. 数据表前缀（table prefix，特别是当您要在一个数据库中安装多个WordPress时）
我们会使用这些信息来创建一个wp-config.php文件。 如果自动创建未能成功，不用担心，您要做的只是将数据库信息填入配置文件。您也可以在文本编辑器中打开wp-config-sample.php，填入您的信息，并将其另存为wp-config.php。 需要更多帮助？看这里。
绝大多数时候，您的网站服务提供商会给您这些信息。如果您没有这些信息，在继续之前您将需要联系他们。如果您准备好了...
现在就开始!

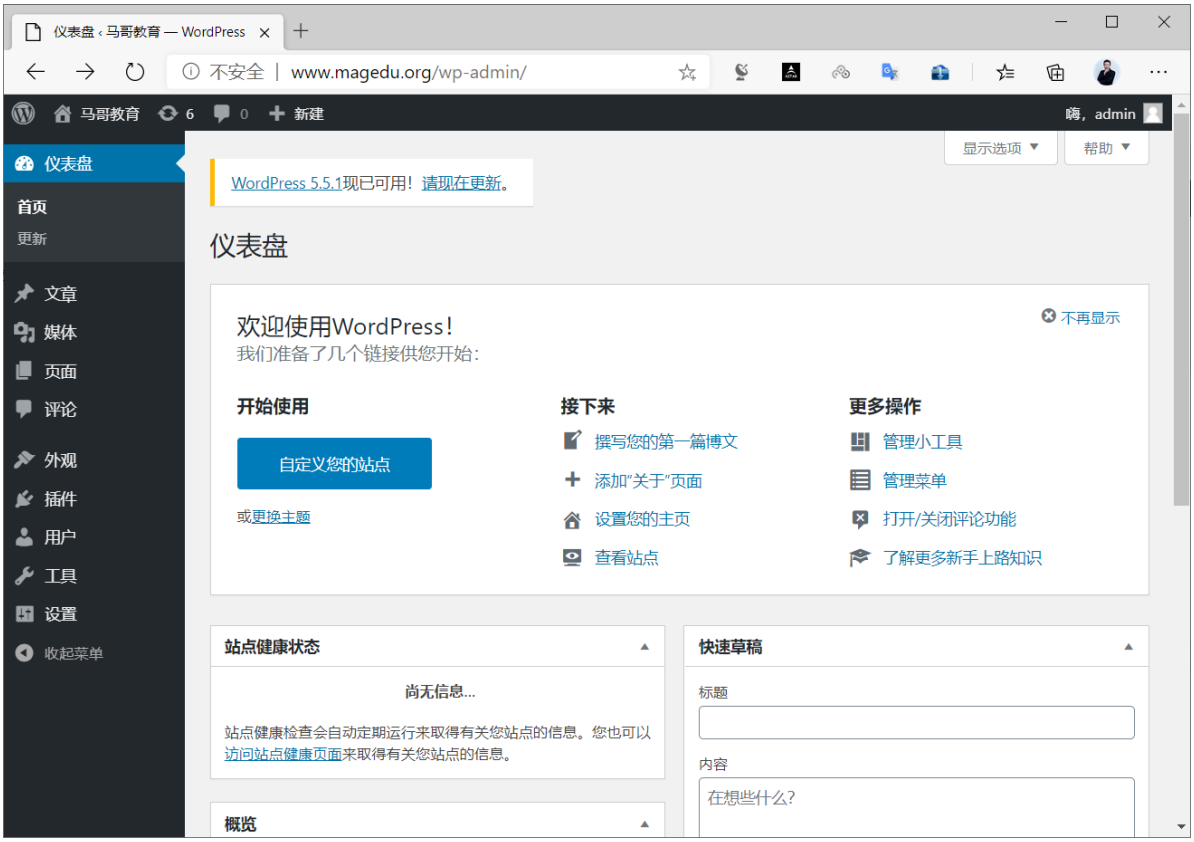

WordPress › 调整配置文件
不安全 | www.magedu.org/wp-admin/setup-config.php?step=1&language=zh_CN
请在下方填写您的数据库连接信息。如果您不确定，请联系您的服务提供商。
数据库名
wordpress
希望将WordPress安装到的数据库名称。
用户名
wordpress
您的数据库用户名。
密码
123456
您的数据库密码。
数据库主机
10.0.0.17
如果localhost不能用，您通常可以从网站服务提供商处得到正确的信息。
表前缀
wp_
如果您希望在同一个数据库安装多个WordPress，请修改前缀。
提交


WordPress › 调整配置文件
不安全 | www.magedu.org/wp-admin/setup-config.php?step=2
不错。您完成了安装过程中重要的一步，WordPress现在已经可以连接数据库了。如果您准备好了的话，现在就...
现在安装



5.3.5.4.3 登录后台管理界面并发表文章





5.3.5.4.4 验证发表的文章

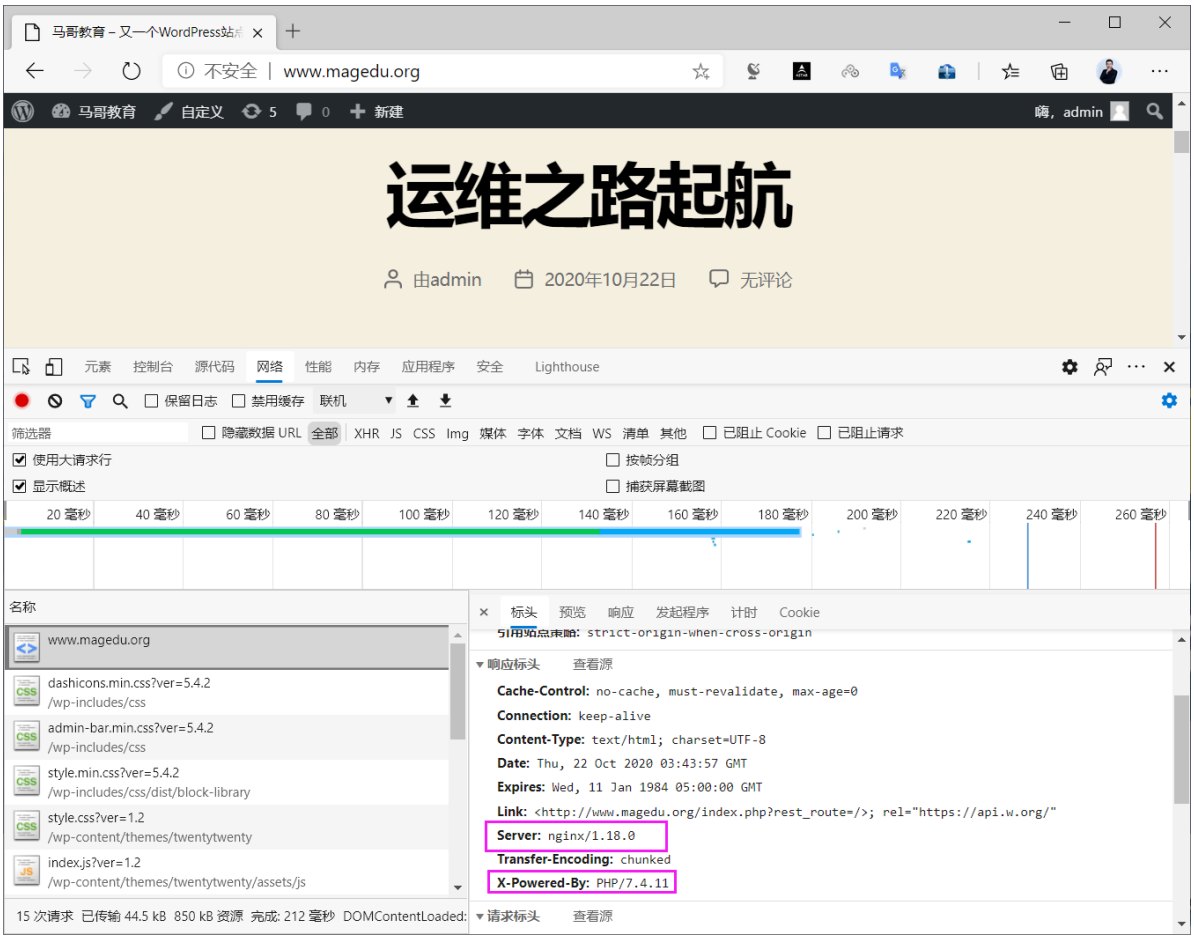

```
#可以看到上传的图片
[root@centos7 ~]#tree /data/nginx/wordpress/wp-content/uploads/
/data/nginx/wordpress/wp-content/uploads/
└── 2020
    └── 10
        └── magedu.jpg

2 directories, 1 file
```

5.3.5.4.5 配置允许上传大文件

```
#注意:默认只支持1M以下文件上传,要利用php程序上传大图片,还需要修改下面三项配置,最大上传由三项值的最小值决定
#直接上传大于1M文件,会出现下面413错误
[root@centos7 ~]#tail -f /apps/nginx/logs/access.log
10.0.0.1 - - [27/Nov/2020:12:21:16 +0800] "POST /wp-admin/async-upload.php HTTP/1.1" 413 585 "http://10.0.0.7/wp-admin/upload.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.67 Safari/537.36 Edg/87.0.664.47"

#nginx上传文件大小限制
[root@centos7 ~]#vim /apps/nginx/conf/nginx.conf
server {
        client_max_body_size 10m; #默认值为1M
.....

#php上传文件大小限制
[root@centos7 ~]#vim /etc/php.ini
post_max_size = 30M   #默认值为8M
upload_max_filesize = 20M  #默认值为2M
```

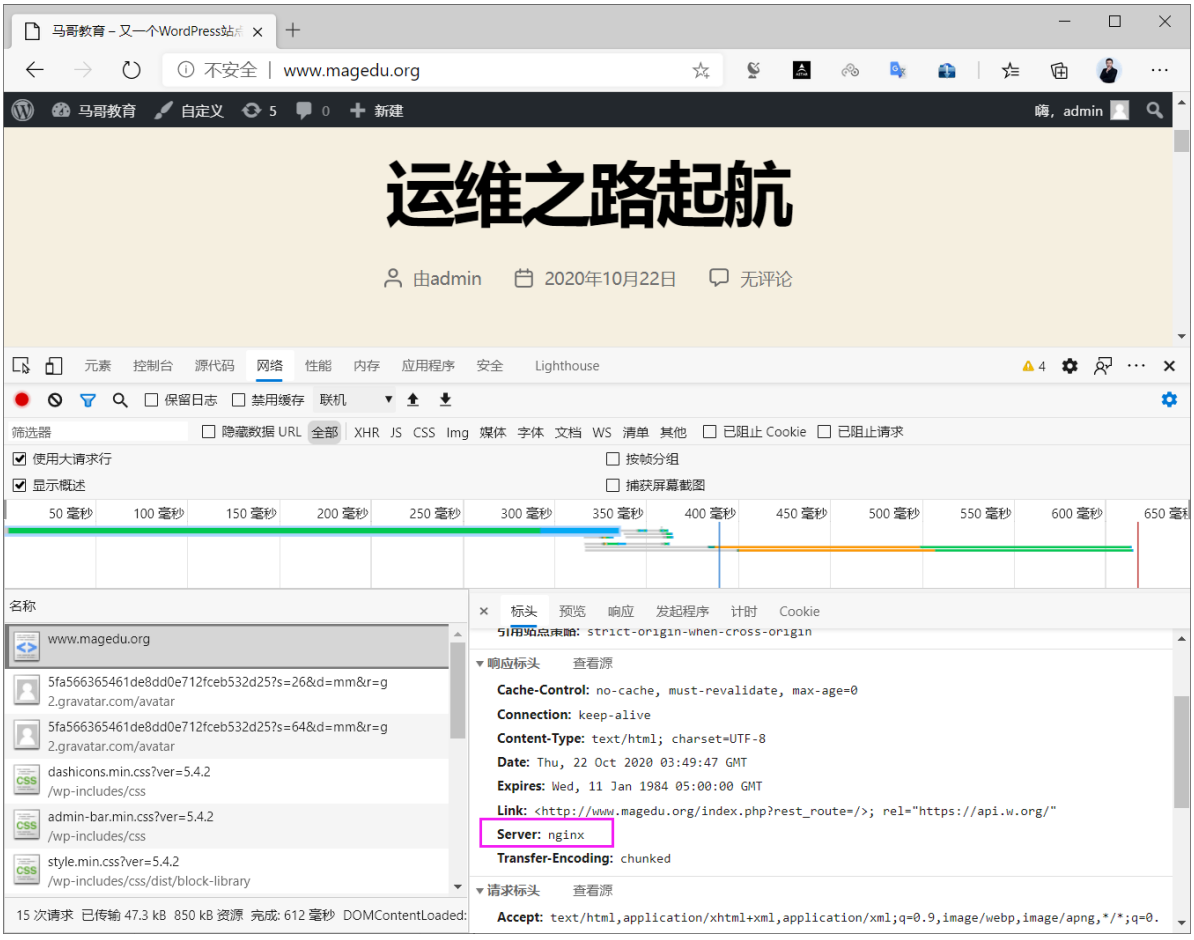
```
[root@centos7 ~]#systemctl restart nginx php-fpm
```

5.3.5.4.6 安全加固




```
[root@centos7 ~]#vim /apps/nginx/conf/nginx.conf
[root@centos7 ~]#grep -Ev '#|^$' /apps/nginx/conf/nginx.conf
worker_processes  1;
pid        /apps/nginx/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;
    server {
        listen       80;
        server_name  www.magedu.org;
        server_tokens off; #添加此行
        location / {
            root   /data/nginx/wordpress;
            index  index.php index.html index.htm;
        }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
```

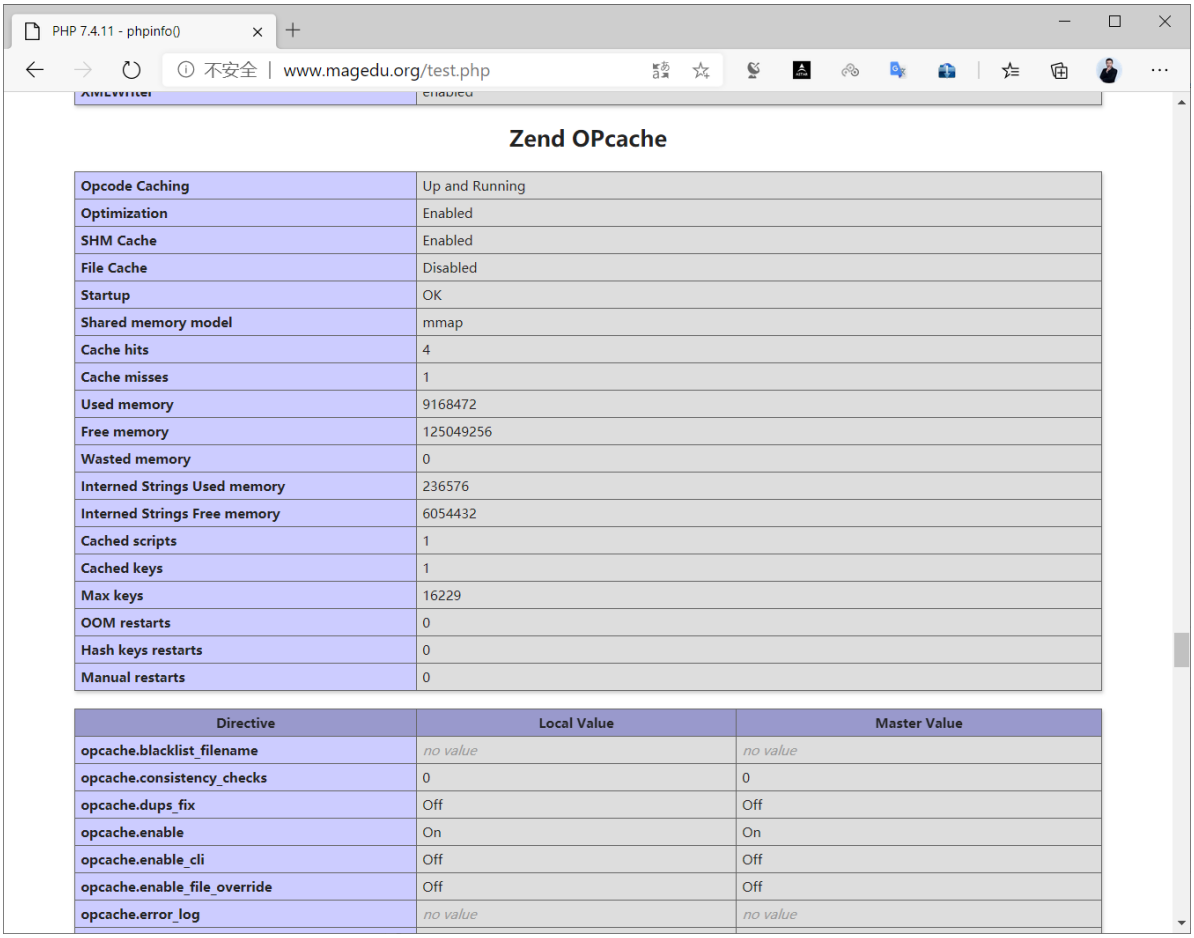
```
            root   html;
        }
        location ~ \.php$ {
            root           /data/nginx/wordpress;
            fastcgi_pass   127.0.0.1:9000;
            fastcgi_index  index.php;
            fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
            include        fastcgi_params;
            fastcgi_hide_header X-Powered-By;  #添加此行
        }
        location ~ ^/(ping|pm_status)$ {
            include fastcgi_params;
            fastcgi_pass 127.0.0.1:9000;
            fastcgi_param PATH_TRANSLATED $document_root$fastcgi_script_name;
        }
    }
}
[root@centos7 ~]#systemctl reload nginx
```



5.3.5.4.7 配置 php 开启 opcache 加速

在10.0.0.7主机进行以下修改配置

```
#编辑php.ini配置文件
[root@centos7 ~]#vim /etc/php.ini
[opcache]
; Determines if Zend OPCache is enabled
zend_extension=opcache.so
                              opcache.enable=1
.....
[root@centos7 ~]#systemctl restart php-fpm

#访问测试页确认开启opcache加速
```



Zend OPcache

Opcode Caching	Up and Running
Optimization	Enabled
SHM Cache	Enabled
File Cache	Disabled
Startup	OK
Shared memory model	mmap
Cache hits	4
Cache misses	1
Used memory	9168472
Free memory	125049256
Wasted memory	0
Interned Strings Used memory	236576
Interned Strings Free memory	6054432
Cached scripts	1
Cached keys	1
Max keys	16229
OOM restarts	0
Hash keys restarts	0
Manual restarts	0

Directive	Local Value	Master Value
opcache.blacklist_filename	*no value*	*no value*
opcache.consistency_checks	0	0
opcache.dups_fix	Off	Off
opcache.enable	On	On
opcache.enable_cli	Off	Off
opcache.enable_file_override	Off	Off
opcache.error_log	*no value*	*no value*

5.3.5.5 PHP 扩展session模块支持redis

PECL是 PHP 扩展的存储库，提供用于下载和开发 PHP 扩展的所有已知扩展和托管功能的目录

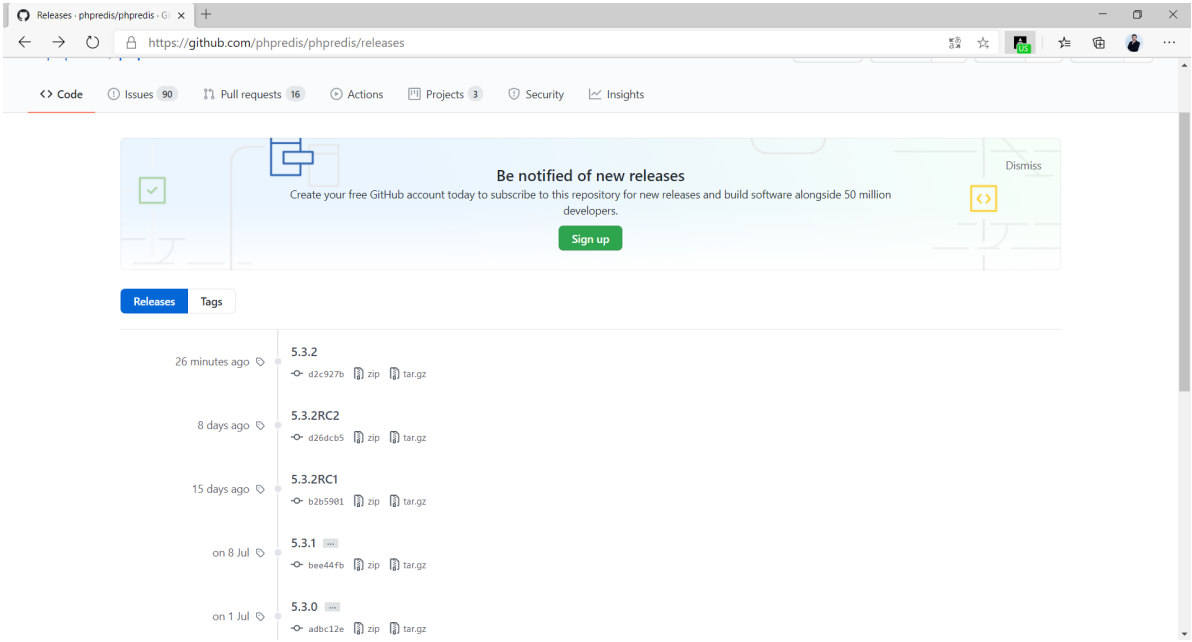
官方链接: http://pecl.php.net/package-stats.php

github: https://github.com/phpredis/phpredis

github安装文档: https://github.com/phpredis/phpredis/blob/develop/INSTALL.markdown

开始在 PHP 中使用 Redis 前， 需要确保已经安装了 redis 服务及 PHP redis 驱动，

PHP redis 驱动下载地址为:**https://github.com/phpredis/phpredis/releases**



5.3.5.5.1 编译安装phpredis模块

在10.0.0.7主机进行以下编译安装phpredis模块

```
[root@centos7 ~]#cd /usr/local/src
[root@centos7 src]#ls
[root@centos7 src]#wget http://pecl.php.net/get/redis-5.3.1.tgz
[root@centos7 src]#tar xf redis-5.3.1.tgz
[root@centos7 src]#cd redis-5.3.1/
[root@centos7 redis-5.3.1]#ls
arrays.markdown    config.m4    INSTALL.markdown  README.markdown     redis.c
      redis_sentinel.c    sentinel_library.h
cluster_library.c  config.w32  liblzf            redis_array.c
redis_cluster.c    redis_sentinel.h    sentinel.markdown
cluster_library.h  COPYING     library.c         redis_array.h
redis_cluster.h    redis_session.c     tests
cluster.markdown   crc16.h     library.h         redis_array_impl.c
redis_commands.c   redis_session.h
common.h           CREDITS     php_redis.h       redis_array_impl.h
redis_commands.h   sentinel_library.c

#如果是yum安装php,需要执行yum -y install php-cli php-devel
#以下为编译安装php的对应方式
[root@centos7 redis-5.3.1]#/apps/php74/bin/phpize
Configuring for:
PHP Api Version:         20190902
Zend Module Api No:      20190902
Zend Extension Api No:   320190902
Cannot find autoconf. Please check your autoconf installation and the #报错提示
$PHP_AUTOCONF environment variable. Then, rerun this script.

[root@centos7 redis-5.3.1]#yum -y install autoconf
#重新执行成功
[root@centos7 redis-5.3.1]#/apps/php74/bin/phpize
Configuring for:
PHP Api Version:         20190902
Zend Module Api No:      20190902
Zend Extension Api No:   320190902
```

```
#查看生成configure脚本
[root@centos7 redis-5.3.1]#ls
arrays.markdown    common.h      COPYING           library.h
redis_array_impl.h  redis_sentinel.c    sentinel_library.h
autom4te.cache     config.h.in   crc16.h           php_redis.h         redis.c
          redis_sentinel.h    sentinel.markdown
build              config.m4     CREDITS           README.markdown
redis_cluster.c     redis_session.c     tests
cluster_library.c  configure     INSTALL.markdown  redis_array.c
redis_cluster.h     redis_session.h
cluster_library.h  configure.ac  liblzf            redis_array.h
redis_commands.c    run-tests.php
cluster.markdown   config.w32    library.c         redis_array_impl.c
redis_commands.h    sentinel_library.c

#如果是基于yum安装php,无需指定--with-php-config
[root@centos7 redis-5.3.1]#./configure  --with-php-config=/apps/php74/bin/php-
config
[root@centos7 redis-5.3.1]#make -j 8 && make install
.....
See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------

Build complete.
Don't forget to run 'make test'.

Installing shared extensions:     /apps/php74/lib/php/extensions/no-debug-zts-
20190902/

#验证redis模块
#yum安装php,模块文件默认存放在 /usr/lib64/php/modules/redis.so
[root@centos7 redis-5.3.1]#ll  /apps/php74/lib/php/extensions/no-debug-zts-
20190902/
total 9584
-rwxr-xr-x 1 root root 4647668 Oct 22 10:44 opcache.a
-rwxr-xr-x 1 root root 2509416 Oct 22 10:44 opcache.so
-rwxr-xr-x 1 root root 2651320 Oct 22 12:10 redis.so
```

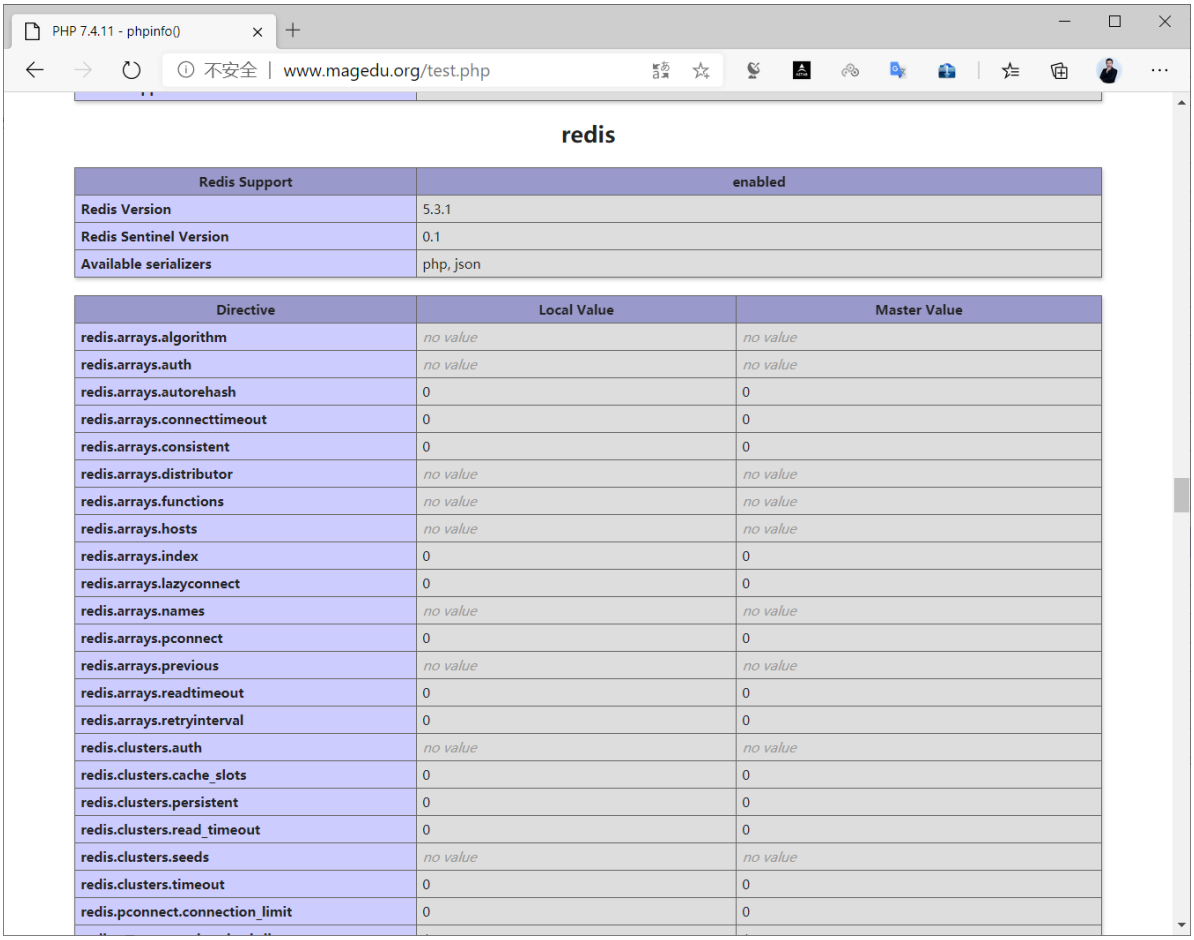
5.3.5.5.2 编辑php配置文件支持redis

在10.0.0.7主机进行以下修改配置

```
#编辑php.ini配置文件，扩展redis.so模块
[root@centos7 ~]#vim /etc/php.ini
.....
#extension=/apps/php74/lib/php/extensions/no-debug-zts-20190902/redis.so
extension=redis.so    #文件最后一行添加此行,路径可省略

[root@centos7 ~]#systemctl restart php-fpm
```

5.3.5.5.3 验证加载 redis 模块

访问测试页确认redis模块开启



5.3.5.5.4 安装和配置 redis 服务

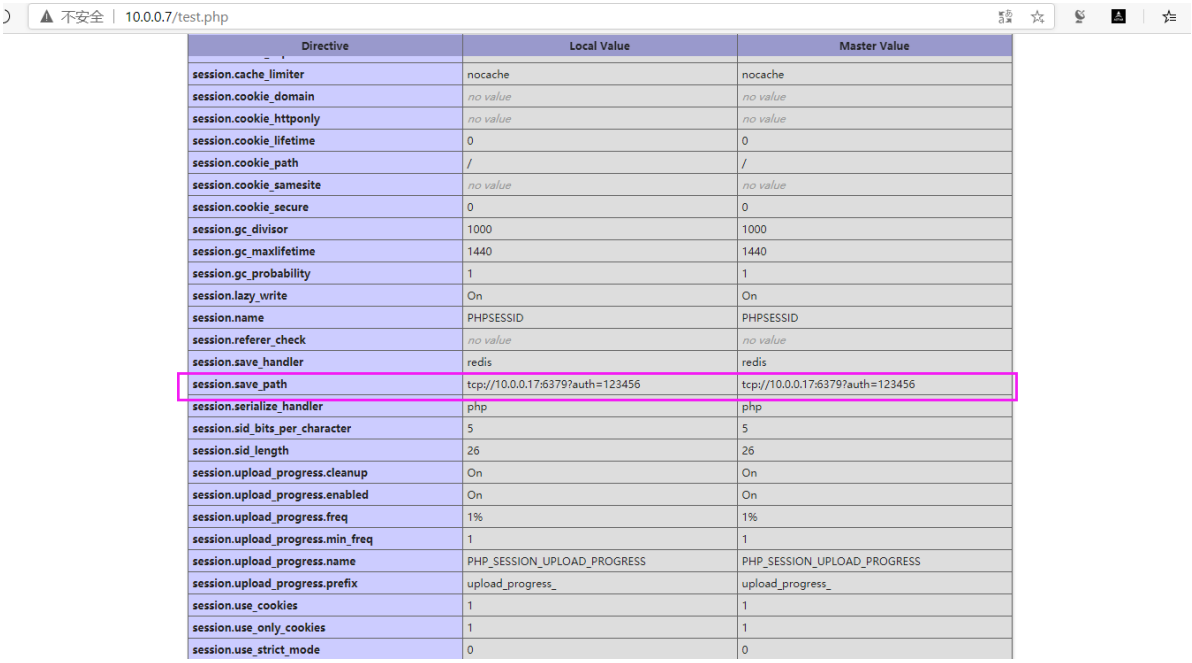
在10.0.0.17主机进行安装redis 服务

```
#在10.0.0.17主机安装redis服务
[root@centos7 ~]#yum -y install redis
[root@centos7 ~]#vim /etc/redis.conf
bind 0.0.0.0
requirepass 123456
[root@centos7 ~]#systemctl enable --now redis
[root@centos7 ~]#ss -tnl
State      Recv-Q Send-Q                     Local Address:Port
                    Peer Address:Port
LISTEN     0      128                              *:22
                                *:*
LISTEN     0      100                          127.0.0.1:25
                                *:*
LISTEN     0      128                              *:6379
                                *:*
LISTEN     0      128                           [::]:22
                               [::]:*
LISTEN     0      100                          [::1]:25
                               [::]:*
LISTEN     0      70                            [::]:33060
                                [::]:*
LISTEN     0      128                           [::]:3306
                               [::]:*
```

5.3.5.5.5 配置 php 支持 redis 保存 session

在10.0.0.7 主机进行以下配置

```
#在10.0.0.7主机配置php的session保存在redis服务
[root@centos7 ~]#vim /etc/php.ini
[Session]
; Handler used to store/retrieve data.
; http://php.net/session.save-handler
session.save_handler = redis
session.save_path = "tcp://10.0.0.17:6379?auth=123456"

[root@centos7 ~]#systemctl restart php-fpm
```



Directive	Local Value	Master Value
session.cache_limiter	nocache	nocache
session.cookie_domain	*no value*	*no value*
session.cookie_httponly	*no value*	*no value*
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_samesite	*no value*	*no value*
session.cookie_secure	0	0
session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.lazy_write	On	On
session.name	PHPSESSID	PHPSESSID
session.referer_check	*no value*	*no value*
session.save_handler	redis	redis
session.save_path	tcp://10.0.0.17:6379?auth=123456	tcp://10.0.0.17:6379?auth=123456
session.serialize_handler	php	php
session.sid_bits_per_character	5	5
session.sid_length	26	26
session.upload_progress.cleanup	On	On
session.upload_progress.enabled	On	On
session.upload_progress.freq	1%	1%
session.upload_progress.min_freq	1	1
session.upload_progress.name	PHP_SESSION_UPLOAD_PROGRESS	PHP_SESSION_UPLOAD_PROGRESS
session.upload_progress.prefix	upload_progress_	upload_progress_
session.use_cookies	1	1
session.use_only_cookies	1	1
session.use_strict_mode	0	0

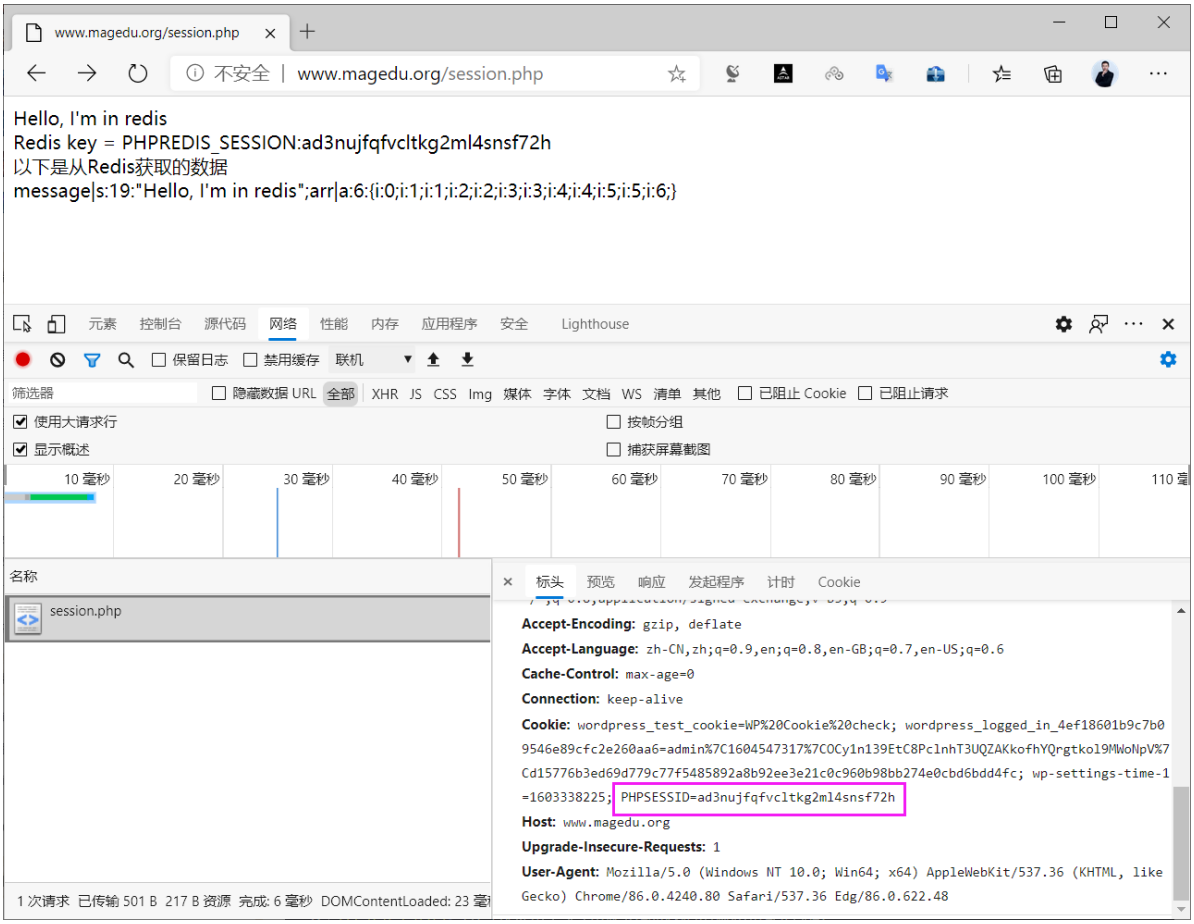
5.3.5.5.6 准备 php实现 session 的测试页面

在10.0.0.7主机进行准备相关文件

```
[root@centos7 ~]#cat /data/nginx/wordpress/session.php
<?php
session_start();
//redis用session_id作为key 并且是以string的形式存储
$redisKey = 'PHPREDIS_SESSION:' . session_id();

// SESSION 赋值测试
$_SESSION['message'] = "Hello, I'm in redis";
$_SESSION['arr'] = [1, 2, 3, 4, 5, 6];

echo $_SESSION["message"] , "<br/>";
echo "Redis key =    " . $redisKey . "<br/>";

echo "以下是从Redis获取的数据", "<br/>";
// 取数据'
$redis = new Redis();
$redis->connect('10.0.0.17', 6379);
$redis->auth('123456');

echo $redis->get($redisKey);
```

```
?>
```

5.3.5.5.7 访问 web 页面测试实现session保存在redis服务



5.3.5.5.8 redis 主机验证 session 数据

在10.0.0.17主机进行验证

```
[root@centos7 ~]#redis-cli  -h 10.0.0.17 -a 123456
10.0.0.17:6379> keys *
1) "PHPREDIS_SESSION:ad3nujfqfvcltkg2ml4snsf72h"
10.0.0.17:6379> get PHPREDIS_SESSION:ad3nujfqfvcltkg2ml4snsf72h
"message|s:19:\"Hello, I'm in redis\";arr|a:6:
{i:0;i:1;i:1;i:2;i:2;i:3;i:3;i:4;i:4;i:5;i:5;i:6;}"
```

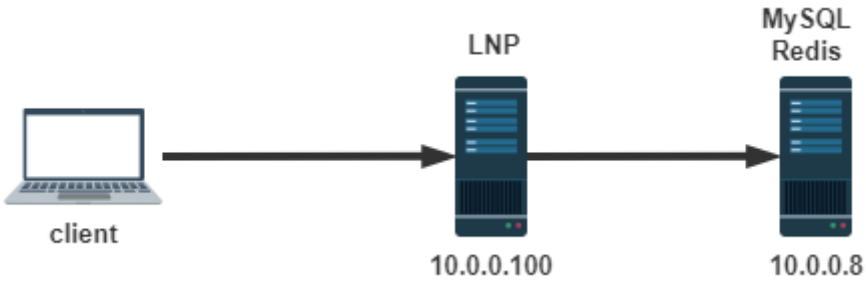
5.3.6 项目实战：利用LNMP实现可道云私有云

可道云，原名芒果云，是基于Web技术的私有云在线文档管理解决方案。Kod，读音通code，意为“代码，编码”，中文名为“可道”。

官网： http://kodcloud.com/

5.3.6.1 环境说明

```
#部署规划：
10.0.0.100：CentOS8,Nginx,php-fpm,kodbox
10.0.0.8：CentOS8,MySQL8.0,Redis5.0
```



5.3.6.2 准备 MySQL 数据库

```
[root@centos8 ~]#yum -y install mysql-server
[root@centos8 ~]#systemctl enable --now mysqld
[root@centos8 ~]#mysql
mysql> create database kodbox;
Query OK, 1 row affected (0.00 sec)

mysql> create user kodbox@'10.0.0.%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on kodbox.* to  kodbox@'10.0.0.%';
Query OK, 0 rows affected (0.00 sec)
```

5.3.6.3 准备 Redis 服务

```
[root@centos8 ~]#yum -y install redis
[root@centos8 ~]#vim /etc/redis.conf
bind 0.0.0.0 #修改此行
[root@centos8 ~]#systemctl enable --now redis
```

5.3.6.4 准备 Nginx 服务

```
[root@centos7 ~]#yum -y install nginx
[root@centos7 ~]#mkdir -pv /data/html
#创建配置文件
[root@centos7 ~]#vim /etc/nginx/conf.d/kod.conf
server {
    listen 80;
    server_name cloud.magedu.org;
    root /data/html;
    location / {
        index index.php index.html;
    }
    location ~ \.php$ {
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        include        fastcgi_params;
    }
}
[root@centos7 ~]#systemctl --now enabled nginx
```

5.3.6.5 安装和配置 php 支持 redis

5.3.6.5 安装和配置 php 支持 redis

5.3.6.5.1 CentOS8 php 支持 redis

```
#安阗php相关包
[root@rocky8 ~]#yum -y install php-fpm php-mbstring php-json php-xml php-gd php-
mysqlnd

#修改fpm配置文件
[root@rocky8 ~]#vim /etc/php-fpm.d/www.conf
user = nginx
group = nginx
listen = 127.0.0.1:9000

#安装编译相关包
[root@rocky8 ~]#yum -y install php-cli php-devel

#编译redis模块
[root@rocky8 ~]#wget https://pecl.php.net/get/redis-5.3.7.tgz
[root@rocky8 ~]#tar xf redis-5.3.7.tgz
[root@rocky8 ~]#cd redis-5.3.7/
[root@rocky8 redis-5.3.7]#cat INSTALL
[root@rocky8 redis-5.3.7]#phpize
[root@rocky8 redis-5.3.7]#./configure
[root@rocky8 redis-5.3.7]#make -j 2 && make install
......
----------------------------------------------------------------------

Build complete.
Don't forget to run 'make test'.

Installing shared extensions:     /usr/lib64/php/modules/


[root@rocky8 ~]#ll /usr/lib64/php/modules/redis.so
-rwxr-xr-x 1 root root 3530824 Feb 26 10:21 /usr/lib64/php/modules/redis.so

[root@rocky8 ~]#vim /etc/php.d/31-redis.ini
extension=redis

[root@rocky8 ~]#systemctl enable --now php-fpm

#测试连接
[root@rocky8 ~]#vim /data/html/redis.php
<?php
    //连接本地的 Redis 服务
   $redis = new Redis();
   $redis->connect('127.0.0.1', 6379);
   //$redis->auth('123456');
   echo "Connection to Redis sucessfully! ";
   //查看服务是否运行
   echo "Server is running: " . $redis->ping();

[root@rocky8 ~]#curl http://10.0.0.8/redis.php
Connection to server sucessfully! Server is running: 1
```

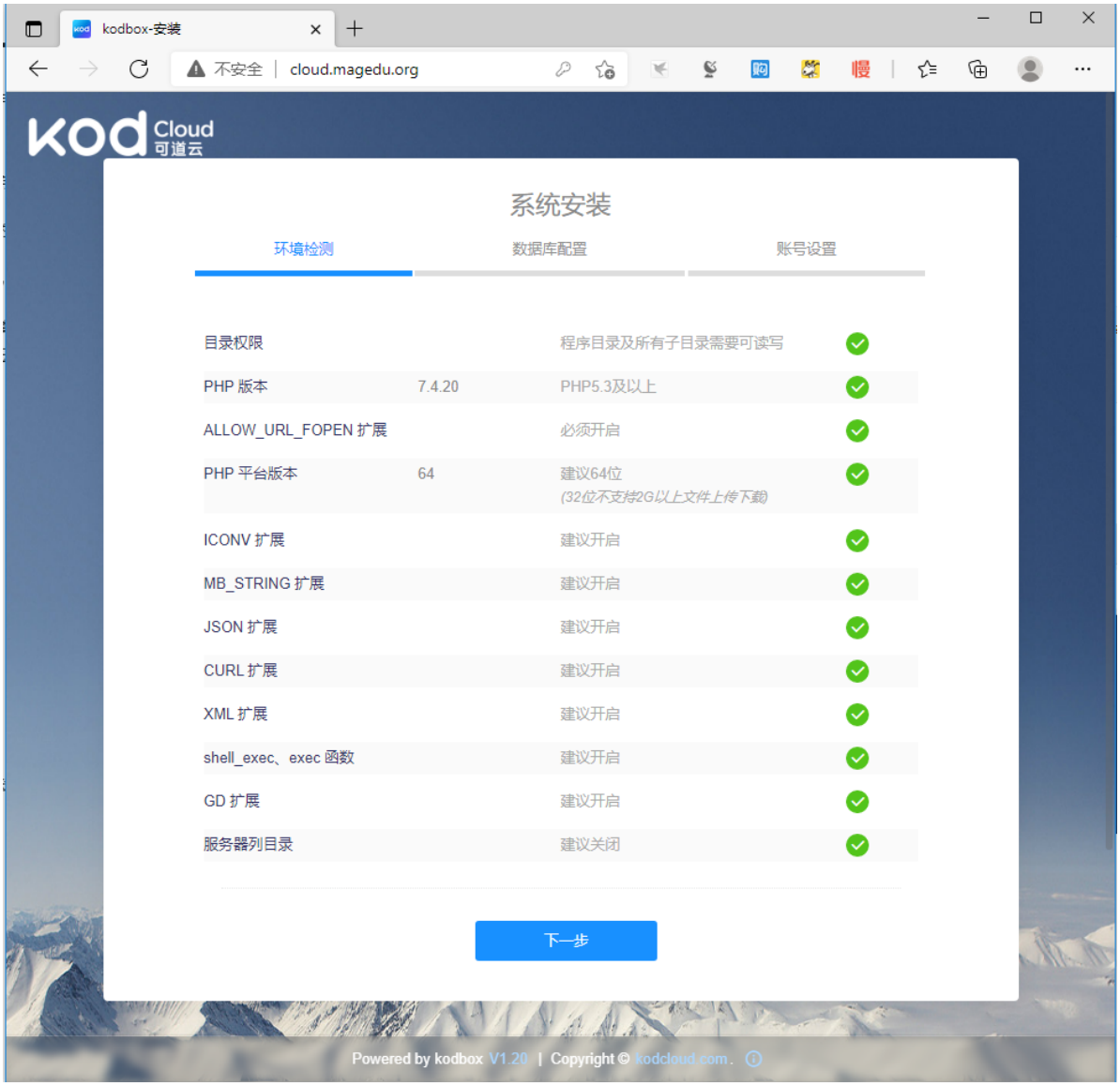
5.3.6.5.2 CentOS7 php 支持 redis

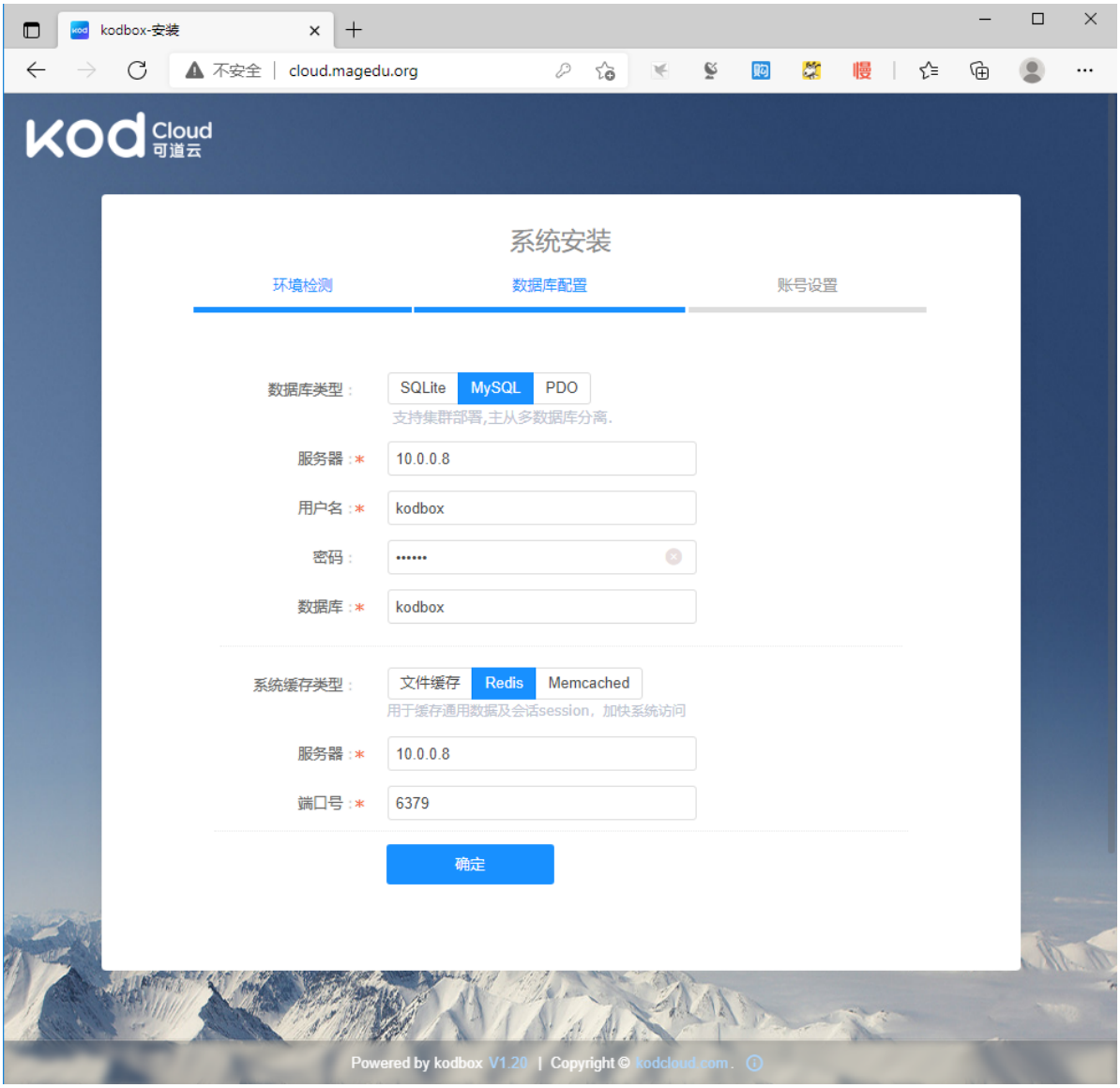
```
[root@centos7 ~]#yum install
https://mirror.tuna.tsinghua.edu.cn/remi/enterprise/remi-release-7.rpm

#安装必要的包
[root@centos7 ~]#yum -y install php74-php-fpm  php74-php-mysqlnd php74-php-pecl-
redis5 php74-php-mbstring  php74-php-xml php74-php-gd

[root@centos7 ~]#vim /etc/opt/remi/php74/php-fpm.d/www.conf
user = nginx
group = nginx
listen = 127.0.0.1:9000

#文件最后修改下面两行
php_value[session.save_handler] = redis
php_value[session.save_path]    = "tcp://10.0.0.8:6379"
```

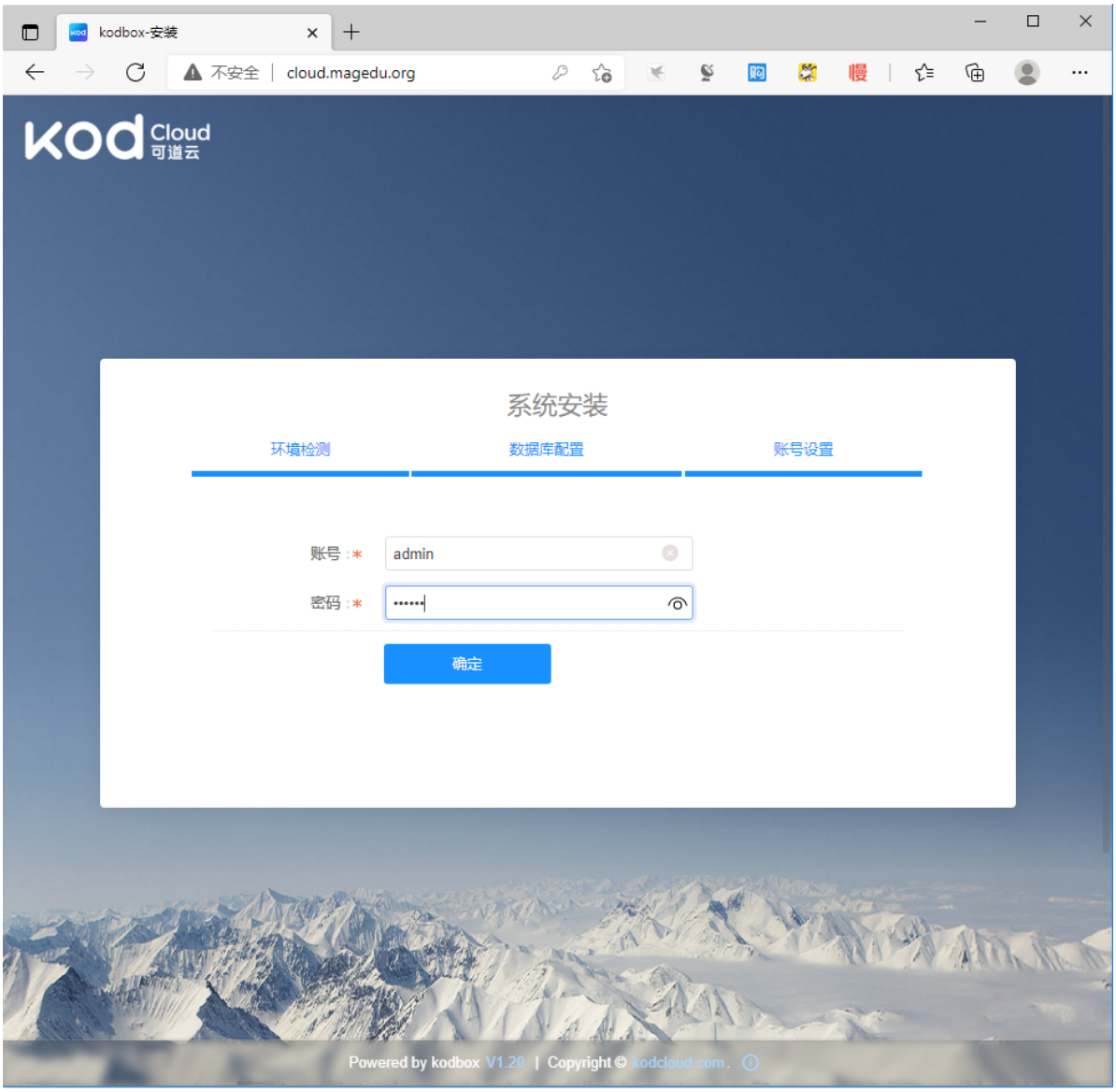
5.3.6.6 准备可道云程序

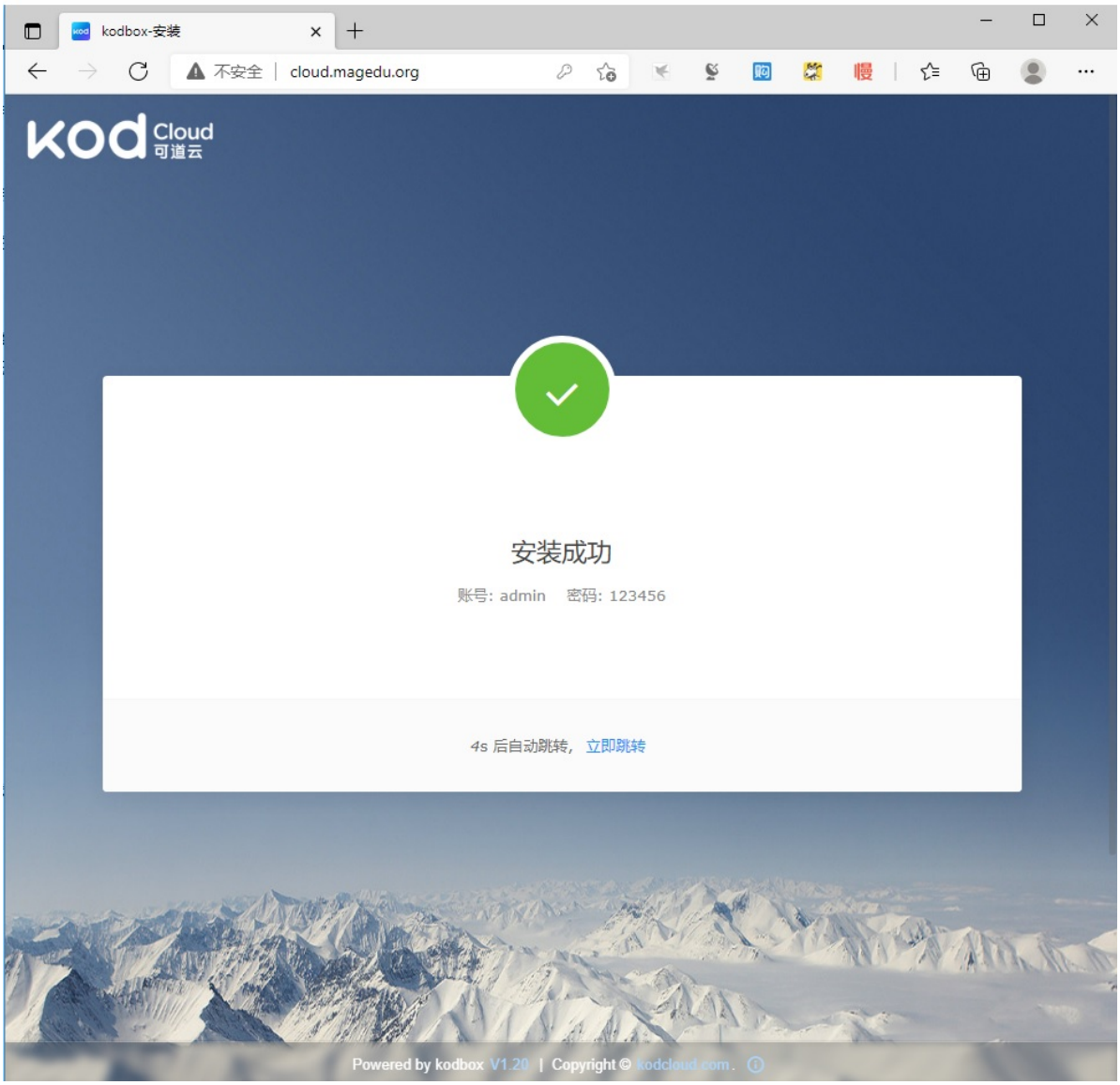
```
[root@centos7 ~]#wget
https://static.kodcloud.com/update/download/kodbox.1.20.zip
[root@centos7 ~]#unzip kodbox.1.20.zip  -d /data/html
[root@centos7 ~]#chown -R nginx.nginx /data/html
```

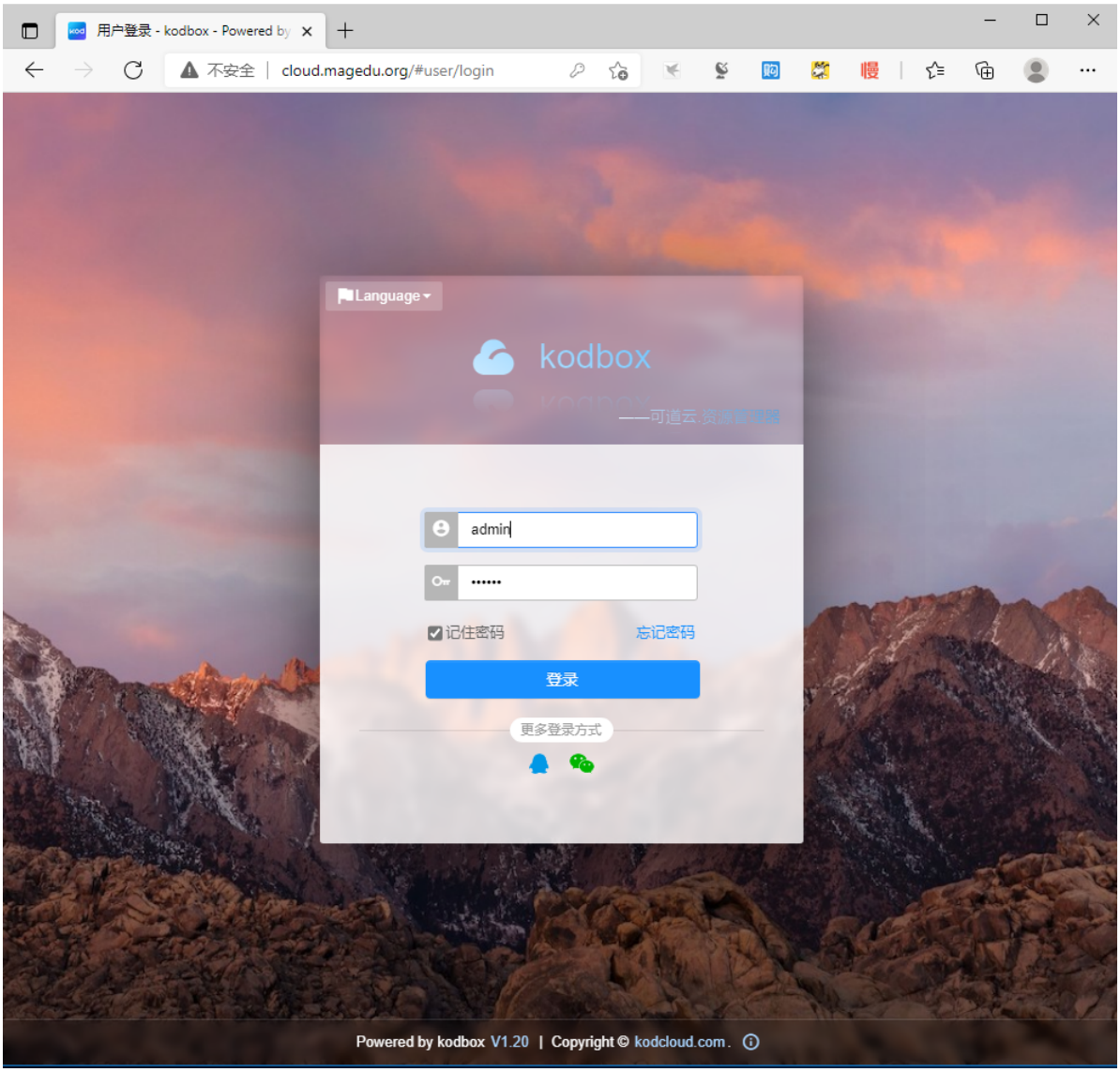
5.3.6.7 初始化和登录可道云

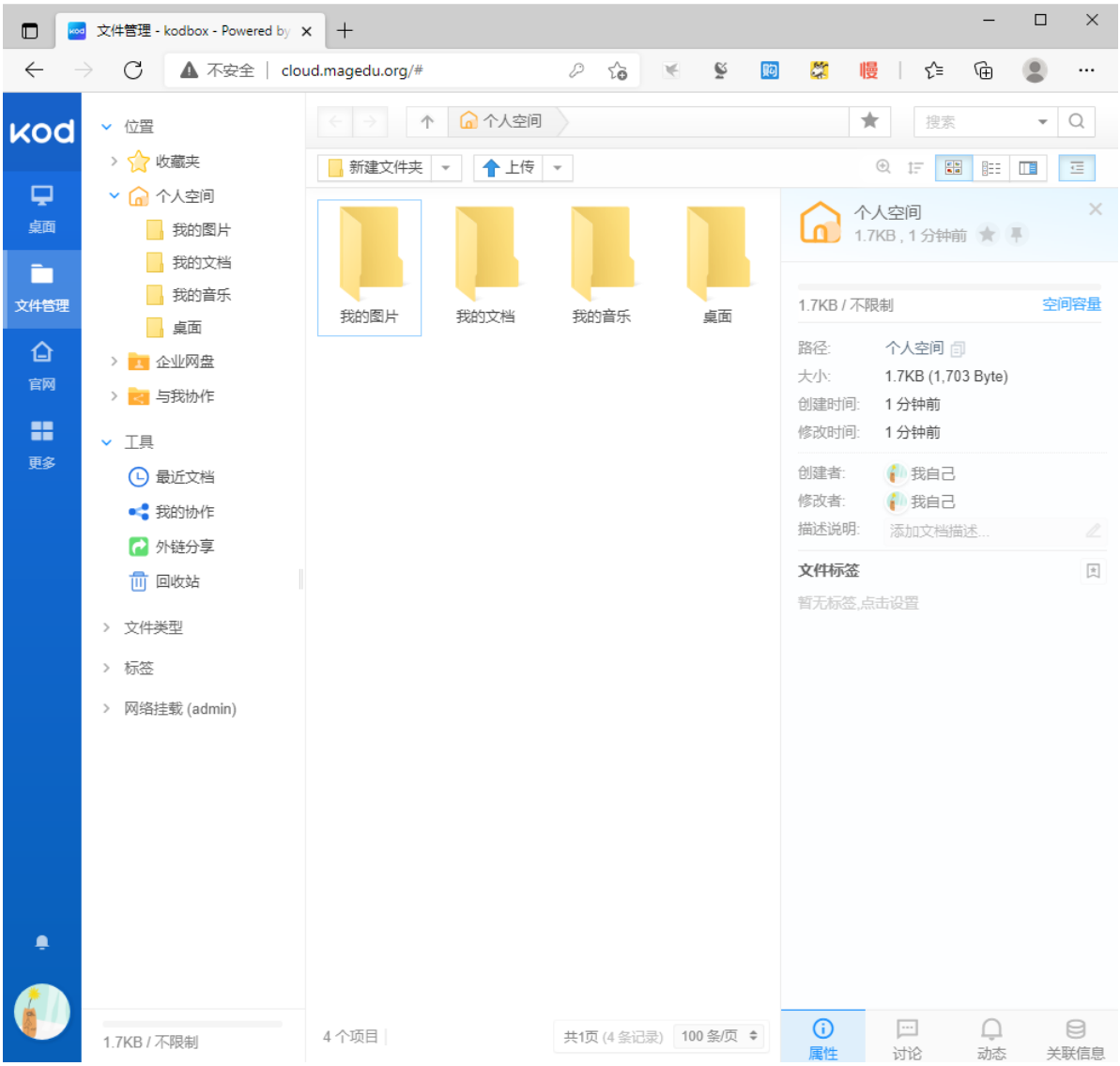

kodbox-安装
不安全 | cloud.magedu.org
kod Cloud 可道云
系统安装
环境检测
数据库配置
账号设置
目录权限
程序目录及所有子目录需要可读写
PHP 版本
7.4.20
PHP5.3及以上
ALLOW_URL_FOPEN 扩展
必须开启
PHP 平台版本
64
建议64位
(32位不支持2G以上文件上传下载)
ICONV 扩展
建议开启
MB_STRING 扩展
建议开启
JSON 扩展
建议开启
CURL 扩展
建议开启
XML 扩展
建议开启
shell_exec、exec 函数
建议开启
GD 扩展
建议开启
服务器列目录
建议关闭
下一步
Powered by kodbox V1.20 | Copyright © kodcloud.com.

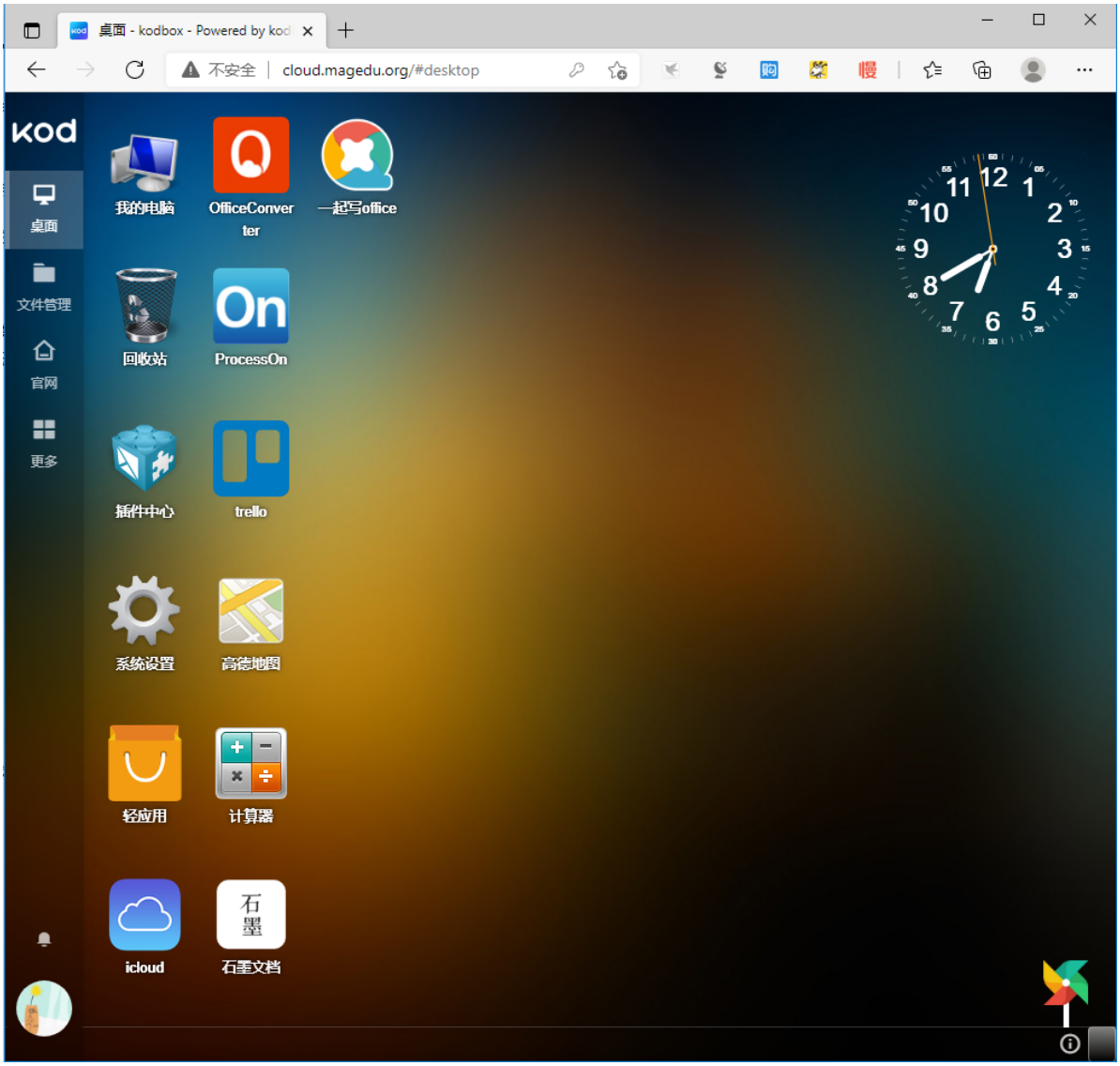

kodbox-安装
不安全 | cloud.magedu.org
kod Cloud 可道云
系统安装
环境检测
数据库配置
账号设置
数据库类型：
SQLite
MySQL
PDO
支持集群部署,主从多数据库分离.
服务器：*
10.0.0.8
用户名：*
kodbox
密码：
••••••
数据库：*
kodbox
系统缓存类型：
文件缓存
Redis
Memcached
用于缓存通用数据及会话session，加快系统访问
服务器：*
10.0.0.8
端口号：*
6379
确定
Powered by kodbox V1.20 | Copyright © kodcloud.com.

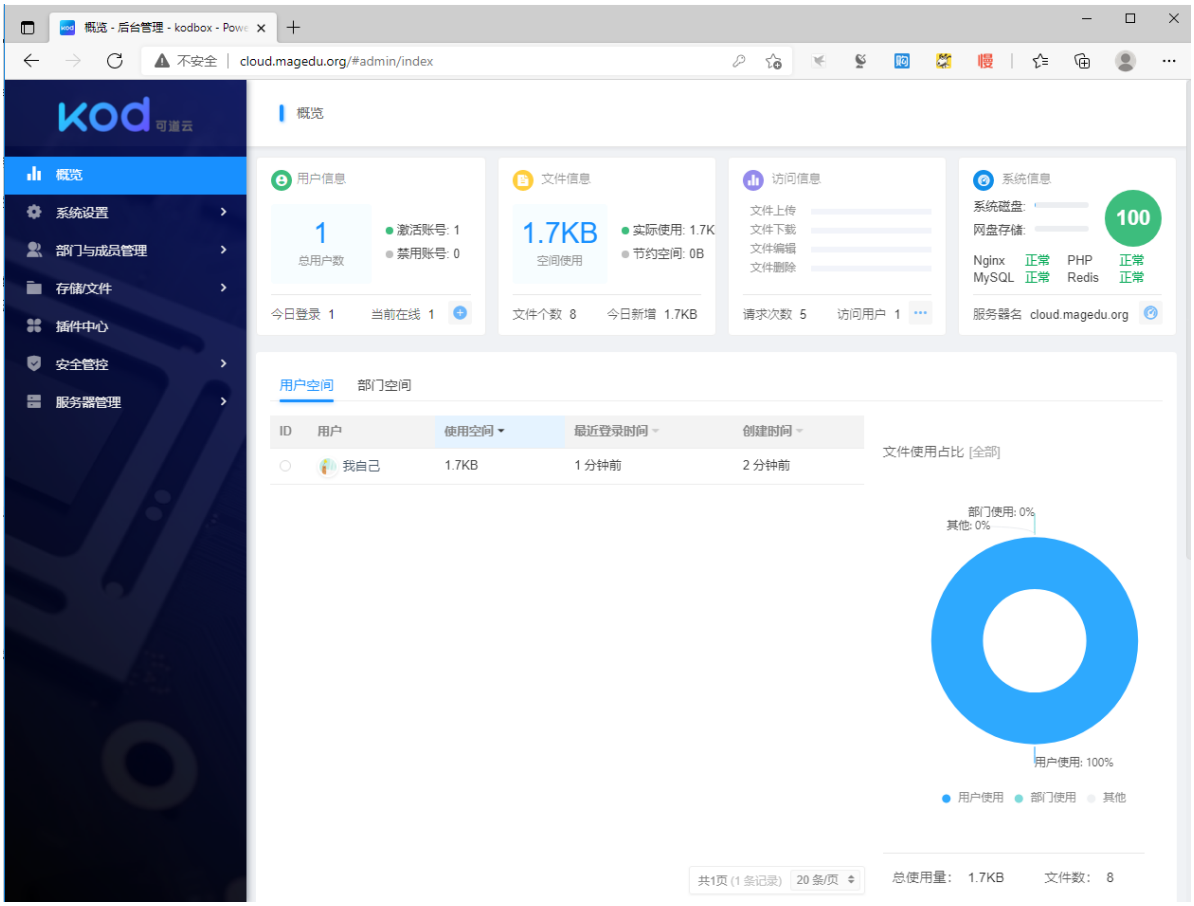

kodbox-安装
不安全 | cloud.magedu.org
kod Cloud 可道云
系统安装
环境检测
数据库配置
账号设置
账号：*
admin
密码：*
确定
Powered by kodbox V1.20 | Copyright © kodcloud.com.


kodbox-安装
不安全 | cloud.magedu.org
kod Cloud 可道云
安装成功
账号: admin 密码: 123456
4s 后自动跳转, 立即跳转
Powered by kodbox V1.20 | Copyright © kodcloud.com.


用户登录 - kodbox - Powered by
不安全 | cloud.magedu.org/#user/login
Language
kodbox
——可道云.资源管理器
admin
记住密码
忘记密码
登录
更多登录方式
Powered by kodbox V1.20 | Copyright © kodcloud.com.


文件管理 - kodbox - Powered by
不安全 | cloud.magedu.org/#
kod
桌面
文件管理
官网
更多
位置
收藏夹
个人空间
我的图片
我的文档
我的音乐
桌面
企业网盘
与我协作
工具
最近文档
我的协作
外链分享
回收站
文件类型
标签
网络挂载 (admin)
1.7KB / 不限制
个人空间
搜索
新建文件夹
上传
我的图片
我的文档
我的音乐
桌面
4 个项目
共1页 (4 条记录)
100 条/页
个人空间
1.7KB , 1 分钟前
1.7KB / 不限制
空间容量
路径: 个人空间
大小: 1.7KB (1,703 Byte)
创建时间: 1 分钟前
修改时间: 1 分钟前
创建者: 我自己
修改者: 我自己
描述说明: 添加文档描述...
文件标签
暂无标签,点击设置
属性
讨论
动态
关联信息


桌面 - kodbox - Powered by kod
不安全 | cloud.magedu.org/#desktop
kod
桌面
文件管理
官网
更多
我的电脑
OfficeConverter
一起写office
回收站
ProcessOn
插件中心
trello
系统设置
高德地图
轻应用
计算器
icloud
石墨文档


概览 - 后台管理 - kodbox - Powe
不安全 | cloud.magedu.org/#admin/index
kod 可道云
概览
系统设置
部门与成员管理
存储/文件
插件中心
安全管控
服务器管理
用户信息
1
总用户数
激活账号: 1
禁用账号: 0
今日登录 1
当前在线 1
文件信息
1.7KB
空间使用
实际使用: 1.7K
节约空间: 0B
文件个数 8
今日新增 1.7KB
访问信息
文件上传
文件下载
文件编辑
文件删除
请求次数 5
访问用户 1
系统信息
系统磁盘:
网盘存储:
100
Nginx 正常
PHP 正常
MySQL 正常
Redis 正常
服务器名 cloud.magedu.org
用户空间
部门空间
ID
用户
使用空间
最近登录时间
创建时间
我自己
1.7KB
1 分钟前
2 分钟前
文件使用占比 [全部]
部门使用: 0%
其他: 0%
用户使用: 100%
用户使用
部门使用
其他
共1页 (1 条记录)
20 条/页
总使用量: 1.7KB
文件数: 8

5.3.6.8 验证数据库和 session 信息

```
[root@centos8 ~]#redis-cli
127.0.0.1:6379> keys *
 1) "32b4b_UserModel_getInfoFull_ID_1"
 2) "32b4b_SystemOption_System.taskList"
 3) "32b4b_SystemOption_System.autoTask"
 4) "32b4b_UserOption_User.noticeList_1"
 5) "32b4b_UserFavModel_listData_ID_1"
 6) "32b4b_UserOption_User.tagList_1"
 7) "32b4b_SystemOption_"
 8) "32b4b_UserOption__0"
 9) "32b4b_%7Bsource%3A19%7D%2FattachmentTemp%2F"
10) "32b4b_UserTagSourceModel_listData_ID_1"
11) "32b4b_UserModel_getInfoSimple_ID_1"
12) "32b4b_SystemOption_System.roleList"
13) "32b4b_UserOption__1"
14) "32b4b_SystemOption_System.noticeList"
15) "b3ccde624f1bc5f45f3eec27812d90e5"
16) "32b4b_UserOption_editor_1"
17) "32b4b_SystemOption_System.sourceAuthList"
18) "32b4b_IO_hashMd5_shell"
19) "32b4b_IO_check_2e12e08959028ba362957fe10e2fea43"
20) "32b4b_SystemOption_System.pluginList"
21) "32b4b_UserModel_getInfo_ID_1"
22) "32b4b_GroupModel_getInfoSimple_ID_1"
23) "32b4b_GroupModel_getInfo_ID_1"
24) "32b4b_SystemOption_System.storageList"
25) "32b4b_ShareModel_listSimple_ID_1"
26) "32b4b_SystemOption_System.LightApp"
127.0.0.1:6379>


mysql> use kodbox
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------+
| Tables_in_kodbox  |
+-------------------+
| comment           |
| comment_meta      |
| comment_praise    |
| group             |
| group_meta        |
| io_file           |
| io_file_contents  |
| io_file_meta      |
| io_source         |
| io_source_auth    |
| io_source_event   |
| io_source_history |
| io_source_meta    |
| io_source_recycle |
| share             |
```

```
| share_report      |
| share_to          |
| system_log        |
| system_option     |
| system_session    |
| user              |
| user_fav          |
| user_group        |
| user_meta         |
| user_option       |
+-------------------+
25 rows in set (0.00 sec)
```

6 Nginx 二次开发版本

6.1 Tengine

6.1.1 Tengine 介绍

Tengine是由淘宝网发起的Web服务器项目。它在Nginx的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。Tengine的性能和稳定性已经在大型的网站如淘宝网，天猫商城等得到了很好的检验。它的最终目标是打造一个高效、稳定、安全、易用的Web平台。

官网：

http://tengine.taobao.org

官方文档：

http://tengine.taobao.org/documentation_cn.html

```
继承Nginx-1.16.0的所有特性，兼容nginx的配置
支持HTTP的CONNECT方法，可用于正向代理场景
支持异步OpenSSL，可使用硬件如:QAT进行HTTPS的加速与卸载
增强相关运维、监控能力,比如异步打印日志及回滚,本地DNS缓存,内存监控等
Stream模块支持server_name指令
更加强大的负载均衡能力，包括一致性hash模块、会话保持模块，还可以对后端的服务器进行主动健康检查，
根据服务器状态自动上线下线，以及动态解析upstream中出现的域名
输入过滤器机制支持。通过使用这种机制Web应用防火墙的编写更为方便
支持设置proxy、memcached、fastcgi、scgi、uwsgi在后端失败时的重试次数
动态脚本语言Lua支持。扩展功能非常高效简单
支持按指定关键字(域名，url等)收集Tengine运行状态
组合多个CSS、JavaScript文件的访问请求变成一个请求
自动去除空白字符和注释从而减小页面的体积
自动根据CPU数目设置进程个数和绑定CPU亲缘性;
监控系统的负载和资源占用从而对系统进行保护
显示对运维人员更友好的出错信息，便于定位出错机器
更强大的防攻击（访问速度限制）模块
更方便的命令行参数，如列出编译的模块列表、支持的指令等
可以根据访问文件类型设置过期时间
```

6.1.2 动态模块

http://tengine.taobao.org/document_cn/dso_cn.html

```
这个模块主要是用来运行时动态加载模块，而不用每次都要重新编译Tengine.
如果想要编译官方模块为动态模块，需要在configure的时候加上类似这样的指令(--with-
http_xxx_module=shared),./configure --help可以看到更多的细节.
如果只想要安装官方模块为动态模块(不安装Nginx)，那么就只需要configure之后，执行 make
dso_install命令.
动态加载模块的个数限制为128个.
如果已经加载的动态模块有修改，那么必须重启Tengine才会生效.
只支持HTTP模块.
模块 在Linux/FreeBSD/MacOS下测试成功.

例子:
worker_processes  1;

dso {
    load ngx_http_lua_module.so;
    load ngx_http_memcached_module.so;
}

events {
    worker_connections  1024;
}

#注意
在Tengine-2.3.0版本后废弃Tengine的dso_tool工具以及dso配置指令，若之前有使用Tengine的dso功
能、则可以切换到Nginx官方的load_module指令,详细文档
```

6.1.2.1 编译安装 tengine-2.1.2

注意: 不支持CentOS8

```
[root@centos7 ~]#yum -y install gcc pcre-devel openssl-devel
[root@centos7 ~]#useradd -r -s /sbin/nologin nginx

[root@centos7 ~]#cd /usr/local/src
[root@centos7 src]#wget http://tengine.taobao.org/download/tengine-2.1.2.tar.gz
[root@centos7 src]#tar xf tengine-2.1.2.tar.gz
[root@centos7 src]#cd tengine-2.1.2/
[root@centos7 tengine-2.1.2]#./configure --prefix=/apps/tengine-2.1.2 --
user=nginx --group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre
[root@centos7 tengine-2.1.2]#make && make install


[root@centos7 tengine-2.1.2]#tree /apps/tengine-2.1.2/
/apps/tengine-2.1.2/
├── conf
|   ├── browsers
|   ├── fastcgi.conf
|   ├── fastcgi.conf.default
|   ├── fastcgi_params
```

```
|   ├── fastcgi_params.default
|   ├── koi-utf
|   ├── koi-win
|   ├── mime.types
|   ├── mime.types.default
|   ├── module_stubs
|   ├── nginx.conf
|   ├── nginx.conf.default
|   ├── scgi_params
|   ├── scgi_params.default
|   ├── uwsgi_params
|   ├── uwsgi_params.default
|   └── win-utf
├── html
|   ├── 50x.html
|   └── index.html
├── include
|   ├── nginx.h
|   ├── ngx_alloc.h
|   ├── ngx_array.h
|   ├── ngx_atomic.h
|   ├── ngx_auto_config.h
|   ├── ngx_auto_headers.h
|   ├── ngx_buf.h
|   ├── ngx_channel.h
|   ├── ngx_conf_file.h
|   ├── ngx_config.h
|   ├── ngx_connection.h
|   ├── ngx_core.h
|   ├── ngx_crc32.h
|   ├── ngx_crc.h
|   ├── ngx_crypt.h
|   ├── ngx_cycle.h
|   ├── ngx_errno.h
|   ├── ngx_event_busy_lock.h
|   ├── ngx_event_connect.h
|   ├── ngx_event.h
|   ├── ngx_event_openssl.h
|   ├── ngx_event_pipe.h
|   ├── ngx_event_posted.h
|   ├── ngx_event_timer.h
|   ├── ngx_file.h
|   ├── ngx_files.h
|   ├── ngx_gcc_atomic_x86.h
|   ├── ngx_hash.h
|   ├── ngx_http_busy_lock.h
|   ├── ngx_http_cache.h
|   ├── ngx_http_config.h
|   ├── ngx_http_core_module.h
|   ├── ngx_http.h
|   ├── ngx_http_reqstat.h
|   ├── ngx_http_request.h
|   ├── ngx_http_script.h
|   ├── ngx_http_ssi_filter_module.h
|   ├── ngx_http_ssl_module.h
|   ├── ngx_http_upstream.h
|   ├── ngx_http_upstream_round_robin.h
|   ├── ngx_http_v2.h
```

```
|   ├── ngx_http_v2_module.h
|   ├── ngx_http_variables.h
|   ├── ngx_inet.h
|   ├── ngx_linux_config.h
|   ├── ngx_linux.h
|   ├── ngx_list.h
|   ├── ngx_log.h
|   ├── ngx_md5.h
|   ├── ngx_murmurhash.h
|   ├── ngx_open_file_cache.h
|   ├── ngx_os.h
|   ├── ngx_palloc.h
|   ├── ngx_parse.h
|   ├── ngx_pipe.h
|   ├── ngx_process_cycle.h
|   ├── ngx_process.h
|   ├── ngx_proc.h
|   ├── ngx_proxy_protocol.h
|   ├── ngx_queue.h
|   ├── ngx_radix_tree.h
|   ├── ngx_rbtree.h
|   ├── ngx_regex.h
|   ├── ngx_resolver.h
|   ├── ngx_segment_tree.h
|   ├── ngx_setaffinity.h
|   ├── ngx_setproctitle.h
|   ├── ngx_sha1.h
|   ├── ngx_shmem.h
|   ├── ngx_shmtx.h
|   ├── ngx_slab.h
|   ├── ngx_socket.h
|   ├── ngx_string.h
|   ├── ngx_sysinfo.h
|   ├── ngx_syslog.h
|   ├── ngx_thread.h
|   ├── ngx_time.h
|   ├── ngx_times.h
|   ├── ngx_trie.h
|   └── ngx_user.h
├── logs
├── modules
└── sbin
    ├── dso_tool
    └── nginx

6 directories, 101 files


[root@centos7 tengine-2.1.2]#ln -s /apps/tengine-2.1.2/sbin/* /usr/sbin/
[root@centos7 tengine-2.1.2]#nginx -v
Tengine version: Tengine/2.1.2 (nginx/1.6.2)

[root@centos7 tengine-2.1.2]#nginx

[root@centos7 ~]#ss -tnl
State      Recv-Q Send-Q   Local Address:Port  Peer Address:Port
LISTEN     0      128                  *:80               *:*
LISTEN     0      128                  *:22               *:*
```

```
LISTEN     0      100          127.0.0.1:25               *:*
LISTEN     0      128               [::]:22            [::]:*
LISTEN     0      100              [::1]:25            [::]:*

[root@centos7 ~]#curl 10.0.0.7
<!DOCTYPE html>
<html>
<head>
<title>Welcome to tengine!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to tengine!</h1>
<p>If you see this page, the tengine web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://tengine.taobao.org/">tengine.taobao.org</a>.</p>

<p><em>Thank you for using tengine.</em></p>
</body>
</html>
```

6.1.2.2 在tengine-2.1.2中添加 lua 动态模块

```
#基于上一小节继续以下操作
[root@centos7 ~]#yum -y install lua-devel
[root@centos7 ~]#cd /usr/local/src/tengine-2.1.2

[root@centos7 tengine-2.1.2]#./configure --help |grep http_lua
--with-http_lua_module         enable ngx_http_lua_module (will also enable --
with-md5 and --with-sha1)
--with-http_lua_module=shared  enable ngx_http_lua_module (shared) (will also
enable --with-md5 and --with-sha1)

[root@centos7 tengine-2.1.2]#./configure --prefix=/apps/tengine-2.1.2 --
user=nginx --group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre --with-http_lua_module=shared

[root@centos7 tengine-2.1.2]#make dso_install
[root@centos7 tengine-2.1.2]#tree /apps/tengine-2.1.2/modules/
/apps/tengine-2.1.2/modules/
└── ngx_http_lua_module.so

0 directories, 1 file

[root@centos7 tengine-2.1.2]#vim /apps/tengine-2.1.2/conf/nginx.conf
......
events {
    worker_connections  1024;
```

```
}

# load modules compiled as Dynamic Shared Object (DSO)
#
#dso {
#    load ngx_http_fastcgi_module.so;
#    load ngx_http_rewrite_module.so;
#}
dso {

    load ngx_http_lua_module.so;
}
http {
......
[root@centos7 tengine-2.1.2]#nginx -t
the configuration file /apps/tengine-2.1.2/conf/nginx.conf syntax is ok
configuration file /apps/tengine-2.1.2/conf/nginx.conf test is successful

[root@centos7 tengine-2.1.2]#nginx -V |& grep share
    ngx_http_lua_module (shared, 3.1)
```

6.1.2.3 编译安装 tengine-2.3.2

```
[root@centos8 ~]#yum -y install gcc pcre-devel openssl-devel
[root@centos8 ~]#useradd -r -s /sbin/nologin nginx

[root@centos8 ~]#cd /usr/local/src
[root@centos8 src]#wget http://tengine.taobao.org/download/tengine-2.3.2.tar.gz
[root@centos8 src]#tar xvf tengine-2.3.2.tar.gz
[root@centos8 src]#cd tengine-2.3.2/

[root@centos8 tengine-2.3.2]#./configure --prefix=/apps/tengine-2.3.2 --
user=nginx --group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre --with-stream --with-stream_ssl_module --with-stream_realip_module
[root@centos8 tengine-2.3.2]#make && make install

[root@centos8 tengine-2.3.2]#tree /apps/tengine-2.3.2/
/apps/tengine-2.3.2/
├── conf
|   ├── fastcgi.conf
|   ├── fastcgi.conf.default
|   ├── fastcgi_params
|   ├── fastcgi_params.default
|   ├── koi-utf
|   ├── koi-win
|   ├── mime.types
|   ├── mime.types.default
|   ├── nginx.conf
|   ├── nginx.conf.default
|   ├── scgi_params
|   ├── scgi_params.default
|   ├── uwsgi_params
|   ├── uwsgi_params.default
|   └── win-utf
├── html
|   ├── 50x.html
```

```
|   └── index.html
├── logs
└── sbin
    └── nginx

4 directories, 18 files
[root@centos8 tengine-2.3.2]#ln -s /apps/tengine-2.3.2/sbin/* /usr/sbin/
[root@centos8 tengine-2.3.2]#nginx -v
Tengine version: Tengine/2.3.2
nginx version: nginx/1.17.3

[root@centos8 tengine-2.3.2]#nginx
[root@centos8 tengine-2.3.2]#ps aux|grep nginx
root       31008  0.0  0.0  41188   864 ?        Ss   13:10   0:00 nginx: master
process nginx
nginx      31009  0.0  0.5  74732  5260 ?        S    13:10   0:00 nginx: worker
process
root       31011  0.0  0.1  12108  1064 pts/0    S+   13:10   0:00 grep --
color=auto nginx
[root@centos8 tengine-2.3.2]#ss -ntl
State             Recv-Q            Send-Q   Local Address:Port  Peer
Address:Port
LISTEN            0                 128            0.0.0.0:80         0.0.0.0:*

LISTEN            0                 128            0.0.0.0:22         0.0.0.0:*

LISTEN            0                 100          127.0.0.1:25         0.0.0.0:*

LISTEN            0                 128               [::]:22            [::]:*

LISTEN            0                 100             [::1]:25            [::]:*



[root@centos8 ~]#curl 10.0.0.8
<!DOCTYPE html>
<html>
<head>
<title>Welcome to tengine!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to tengine!</h1>
<p>If you see this page, the tengine web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://tengine.taobao.org/">tengine.taobao.org</a>.</p>

<p><em>Thank you for using tengine.</em></p>
</body>
</html>
```

6.1.3 concat 模块使用

6.1.3.1 concat模块说明

网站中的css、js等文件都是小文件，单个文件大小几k甚至几十个字节，而小而多的文件会造成网站加载时http请求较多，且网络传输时间比较短，甚至有时候请求时间比传输时间还长.当网站的这类小文件很多时，大量的http请求就会造成传输效率低，影响网站的访问速度和客户端体验，这时合并http请求就非常有必要了，concat模块就提供了合并文件http请求的功能，这个模块由淘宝开发，功能和apache的mod_concat模块类似

模块说明

```
http://tengine.taobao.org/document_cn/http_concat_cn.html
```

访问方式

```
请求参数需要用两个问号（'??'）例如：
http://example.com/??style1.css,style2.css,foo/style3.css

参数中某位置只包含一个‘?’，则'?'后表示文件的版本，例如：
http://example.com/??style1.css,style2.css,foo/style3.css?v=102234
```

6.1.3.2 编译安装 concat 模块

```
#基于上一小节继续以下操作
[root@centos7 ~]#cd /usr/local/src/tengine-2.1.2

#tengine-2.1.2有此模块
[root@centos7 tengine-2.1.2]#./configure --help |grep http_concat
  --with-http_concat_module          enable ngx_http_concat_module
  --with-http_concat_module=shared   enable ngx_http_concat_module (shared)

#tengine-2.3.2无此模块
[root@centos8 tengine-2.3.2]#./configure --help |grep http_concat
[root@centos8 tengine-2.3.2]#


[root@centos7 tengine-2.1.2]#./configure --prefix=/apps/tengine-2.1.2 --
user=nginx --group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre --with-http_lua_module=shared --with-http_concat_module=shared

[root@centos7 tengine-2.1.2]#make dso_install

[root@centos7 tengine-2.1.2]#tree /apps/tengine-2.1.2/modules/
/apps/tengine-2.1.2/modules/
├── ngx_http_concat_module.so
└── ngx_http_lua_module.so

0 directories, 2 files

#通过动态模块实现concat功能：
[root@centos7 ~]#vim /apps/tengine-2.1.2/conf/nginx.conf
dso {
```

```
    load ngx_http_lua_module.so;
    load ngx_http_concat_module.so;
}

[root@centos7 ~]#nginx -t
the configuration file /apps/tengine-2.1.2/conf/nginx.conf syntax is ok
configuration file /apps/tengine-2.1.2/conf/nginx.conf test is successful
[root@centos7 ~]#nginx -s reload

[root@centos7 tengine-2.1.2]#nginx -V 2>&1| grep share
    ngx_http_concat_module (shared, 3.1)
    ngx_http_lua_module (shared, 3.1)

#通过重新编译tengine实现concat功能：
./configure --with-http_concat_module
```

6.1.4 tengine配置文件

tengine兼容Nginx指定版本的配置参数

```
#tengine配置文件：
[root@centos8 ~]# cat /apps/tenginx/conf/conf.d/pc.conf
server {
    listen 80;
    server_name www.magedu.org *.www.magedu.org pc.magedu.org;
    location / {
        concat on;
        root html;
        index index.html index.htm;
    }
    location ~ \.php$ {
        root /data/php;
        fastcgi_pass 10.0.0.18:9000;
        fastcgi_index index.php;
        #fastcgi_param SCRIPT_FILENAME /data/nginx/php$fastcgi_script_name;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

6.2 openresty

6.2.1 openresty 介绍

Nginx 是俄罗斯人发明的， Lua 是巴西几个教授发明的，中国人章亦春把 LuaJIT VM 嵌入到 Nginx 中，实现了 OpenResty 这个高性能服务端解决方案

OpenResty® 是一个基于 Nginx 与 Lua 的高性能 Web 平台，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。用于方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。

OpenResty® 通过汇聚各种设计精良的 Nginx 模块（主要由 OpenResty 团队自主开发），从而将 Nginx 有效地变成一个强大的通用 Web 应用平台。这样，Web 开发人员和系统工程师可以使用 Lua 脚本语言调动 Nginx 支持的各种 C 以及 Lua 模块，快速构造出足以胜任 10K 乃至 1000K 以上单机并发连接的高性能 Web 应用系统。

OpenResty® 的目标是让你的Web服务直接跑在 Nginx 服务内部，充分利用 Nginx 的非阻塞 I/O 模型，不仅仅对HTTP 客户端请求,甚至于对远程后端诸如 MySQL、PostgreSQL、Memcached 以及 Redis 等都进行一致的高性能响应。

OpenResty 由于有功能强大且方便的的API,可扩展性更强,如果需要实现定制功能,OpenResty是个不错的选择

官网: http://openresty.org/cn/

6.2.2 编译安装 openresty

```
[root@centos8 ~]#dnf -yq install gcc pcre-devel openssl-devel perl
[root@centos8 ~]#useradd -r -s /sbin/nologin nginx

[root@centos8 ~]#cd /usr/local/src
[root@centos8 src]#wget https://openresty.org/download/openresty-1.17.8.2.tar.gz
[root@centos8 src]#tar xf openresty-1.17.8.2.tar.gz

[root@centos8 src]#cd openresty-1.17.8.2/
[root@centos8 openresty-1.17.8.2]#./configure --prefix=/apps/openresty --
user=nginx --group=nginx --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_stub_status_module --with-http_gzip_static_module
--with-pcre --with-stream --with-stream_ssl_module --with-stream_realip_module
[root@centos8 openresty-1.17.8.2]#make && make install

[root@centos8 openresty-1.17.8.2]#tree /apps/openresty/
./apps/openresty/
├── bin
│   ├── md2pod.pl
│   ├── nginx-xml2pod
│   ├── openresty -> /apps/openresty/nginx/sbin/nginx
│   ├── opm
│   ├── resty
│   ├── restydoc
│   └── restydoc-index
├── COPYRIGHT
├── luajit
│   ├── bin
│   │   ├── luajit -> luajit-2.1.0-beta3
│   │   └── luajit-2.1.0-beta3
│   ├── include
│   │   └── luajit-2.1
│   │       ├── lauxlib.h
│   │       ├── luaconf.h
│   │       ├── lua.h
│   │       ├── lua.hpp
│   │       ├── luajit.h
│   │       └── lualib.h
│   ├── lib
│   │   ├── libluajit-5.1.a
│   │   ├── libluajit-5.1.so -> libluajit-5.1.so.2.1.0
│   │   ├── libluajit-5.1.so.2 -> libluajit-5.1.so.2.1.0
```

```
|   |   ├── libluajit-5.1.so.2.1.0
|   |   ├── lua
|   |   |   └── 5.1
|   |   └── pkgconfig
|   |       └── luajit.pc
|   └── share
|       ├── lua
|       |   └── 5.1
|       ├── luajit-2.1.0-beta3
|       |   └── jit
|       |       ├── bc.lua
|       |       ├── bcsave.lua
|       |       ├── dis_arm64be.lua
|       |       ├── dis_arm64.lua
|       |       ├── dis_arm.lua
|       |       ├── dis_mips64el.lua
|       |       ├── dis_mips64.lua
|       |       ├── dis_mipsel.lua
|       |       ├── dis_mips.lua
|       |       ├── dis_ppc.lua
|       |       ├── dis_x64.lua
|       |       ├── dis_x86.lua
|       |       ├── dump.lua
|       |       ├── p.lua
|       |       ├── v.lua
|       |       ├── vmdef.lua
|       |       └── zone.lua
|       └── man
|           └── man1
|               └── luajit.1
├── lualib
|   ├── cjson.so
|   ├── librestysignal.so
|   ├── ngx
|   |   ├── balancer.lua
|   |   ├── base64.lua
|   |   ├── errlog.lua
|   |   ├── ocsp.lua
|   |   ├── pipe.lua
|   |   ├── process.lua
|   |   ├── re.lua
|   |   ├── req.lua
|   |   ├── resp.lua
|   |   ├── semaphore.lua
|   |   ├── ssl
|   |   |   └── session.lua
|   |   └── ssl.lua
|   ├── rds
|   |   └── parser.so
|   ├── redis
|   |   └── parser.so
|   ├── resty
|   |   ├── aes.lua
|   |   ├── core
|   |   |   ├── base64.lua
|   |   |   ├── base.lua
|   |   |   ├── ctx.lua
|   |   |   ├── exit.lua
```

```
|   |   |   ├── hash.lua
|   |   |   ├── misc.lua
|   |   |   ├── ndk.lua
|   |   |   ├── phase.lua
|   |   |   ├── regex.lua
|   |   |   ├── request.lua
|   |   |   ├── response.lua
|   |   |   ├── shdict.lua
|   |   |   ├── time.lua
|   |   |   ├── uri.lua
|   |   |   ├── utils.lua
|   |   |   ├── var.lua
|   |   |   └── worker.lua
|   |   ├── core.lua
|   |   ├── dns
|   |   |   └── resolver.lua
|   |   ├── limit
|   |   |   ├── conn.lua
|   |   |   ├── count.lua
|   |   |   ├── req.lua
|   |   |   └── traffic.lua
|   |   ├── lock.lua
|   |   ├── lrucache
|   |   |   └── pureffi.lua
|   |   ├── lrucache.lua
|   |   ├── md5.lua
|   |   ├── memcached.lua
|   |   ├── mysql.lua
|   |   ├── random.lua
|   |   ├── redis.lua
|   |   ├── sha1.lua
|   |   ├── sha224.lua
|   |   ├── sha256.lua
|   |   ├── sha384.lua
|   |   ├── sha512.lua
|   |   ├── sha.lua
|   |   ├── shell.lua
|   |   ├── signal.lua
|   |   ├── string.lua
|   |   ├── upload.lua
|   |   ├── upstream
|   |   |   └── healthcheck.lua
|   |   └── websocket
|   |       ├── client.lua
|   |       ├── protocol.lua
|   |       └── server.lua
|   └── tablepool.lua
├── nginx
|   ├── conf
|   |   ├── fastcgi.conf
|   |   ├── fastcgi.conf.default
|   |   ├── fastcgi_params
|   |   ├── fastcgi_params.default
|   |   ├── koi-utf
|   |   ├── koi-win
|   |   ├── mime.types
|   |   ├── mime.types.default
|   |   ├── nginx.conf
```

```
|   |   ├── nginx.conf.default
|   |   ├── scgi_params
|   |   ├── scgi_params.default
|   |   ├── uwsgi_params
|   |   ├── uwsgi_params.default
|   |   └── win-utf
|   ├── html
|   |   ├── 50x.html
|   |   └── index.html
|   ├── logs
|   └── sbin
|       └── nginx
├── pod
|   ├── array-var-nginx-module-0.05
|   |   └── array-var-nginx-module-0.05.pod
|   ├── drizzle-nginx-module-0.1.11
|   |   └── drizzle-nginx-module-0.1.11.pod
|   ├── echo-nginx-module-0.62
|   |   └── echo-nginx-module-0.62.pod
|   ├── encrypted-session-nginx-module-0.08
|   |   └── encrypted-session-nginx-module-0.08.pod
|   ├── form-input-nginx-module-0.12
|   |   └── form-input-nginx-module-0.12.pod
|   ├── headers-more-nginx-module-0.33
|   |   └── headers-more-nginx-module-0.33.pod
|   ├── iconv-nginx-module-0.14
|   |   └── iconv-nginx-module-0.14.pod
|   ├── lua-5.1.5
|   |   └── lua-5.1.5.pod
|   ├── lua-cjson-2.1.0.8
|   |   └── lua-cjson-2.1.0.8.pod
|   ├── luajit-2.1
|   |   ├── changes.pod
|   |   ├── contact.pod
|   |   ├── ext_c_api.pod
|   |   ├── extensions.pod
|   |   ├── ext_ffi_api.pod
|   |   ├── ext_ffi.pod
|   |   ├── ext_ffi_semantics.pod
|   |   ├── ext_ffi_tutorial.pod
|   |   ├── ext_jit.pod
|   |   ├── ext_profiler.pod
|   |   ├── faq.pod
|   |   ├── install.pod
|   |   ├── luajit-2.1.pod
|   |   ├── running.pod
|   |   └── status.pod
|   ├── luajit-2.1-20200102
|   |   └── luajit-2.1-20200102.pod
|   ├── lua-rds-parser-0.06
|   ├── lua-redis-parser-0.13
|   |   └── lua-redis-parser-0.13.pod
|   ├── lua-resty-core-0.1.19
|   |   ├── lua-resty-core-0.1.19.pod
|   |   ├── ngx.balancer.pod
|   |   ├── ngx.base64.pod
|   |   ├── ngx.errlog.pod
|   |   ├── ngx.ocsp.pod
```

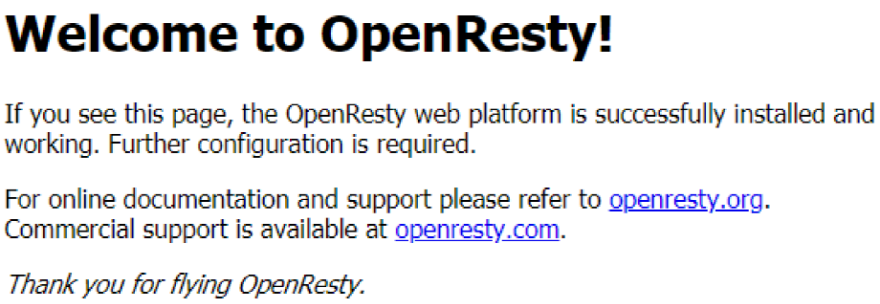
```
|   |   ├── ngx.pipe.pod
|   |   ├── ngx.process.pod
|   |   ├── ngx.re.pod
|   |   ├── ngx.req.pod
|   |   ├── ngx.resp.pod
|   |   ├── ngx.semaphore.pod
|   |   ├── ngx.ssl.pod
|   |   └── ngx.ssl.session.pod
|   ├── lua-resty-dns-0.21
|   |   └── lua-resty-dns-0.21.pod
|   ├── lua-resty-limit-traffic-0.07
|   |   ├── lua-resty-limit-traffic-0.07.pod
|   |   ├── resty.limit.conn.pod
|   |   ├── resty.limit.count.pod
|   |   ├── resty.limit.req.pod
|   |   └── resty.limit.traffic.pod
|   ├── lua-resty-lock-0.08
|   |   └── lua-resty-lock-0.08.pod
|   ├── lua-resty-lrucache-0.10
|   |   └── lua-resty-lrucache-0.10.pod
|   ├── lua-resty-memcached-0.15
|   |   └── lua-resty-memcached-0.15.pod
|   ├── lua-resty-mysql-0.21
|   |   └── lua-resty-mysql-0.21.pod
|   ├── lua-resty-redis-0.28
|   |   └── lua-resty-redis-0.28.pod
|   ├── lua-resty-shell-0.03
|   |   └── lua-resty-shell-0.03.pod
|   ├── lua-resty-signal-0.02
|   |   └── lua-resty-signal-0.02.pod
|   ├── lua-resty-string-0.12
|   |   └── lua-resty-string-0.12.pod
|   ├── lua-resty-upload-0.10
|   |   └── lua-resty-upload-0.10.pod
|   ├── lua-resty-upstream-healthcheck-0.06
|   |   └── lua-resty-upstream-healthcheck-0.06.pod
|   ├── lua-resty-websocket-0.07
|   |   └── lua-resty-websocket-0.07.pod
|   ├── lua-tablepool-0.01
|   |   └── lua-tablepool-0.01.pod
|   ├── memc-nginx-module-0.19
|   |   └── memc-nginx-module-0.19.pod
|   ├── nginx
|   |   ├── accept_failed.pod
|   |   ├── beginners_guide.pod
|   |   ├── chunked_encoding_from_backend.pod
|   |   ├── configure.pod
|   |   ├── configuring_https_servers.pod
|   |   ├── contributing_changes.pod
|   |   ├── control.pod
|   |   ├── converting_rewrite_rules.pod
|   |   ├── daemon_master_process_off.pod
|   |   ├── debugging_log.pod
|   |   ├── development_guide.pod
|   |   ├── events.pod
|   |   ├── example.pod
|   |   ├── faq.pod
|   |   ├── freebsd_tuning.pod
```

```
|   |   ├── hash.pod
|   |   ├── howto_build_on_win32.pod
|   |   ├── install.pod
|   |   ├── license_copyright.pod
|   |   ├── load_balancing.pod
|   |   ├── nginx_dtrace_pid_provider.pod
|   |   ├── nginx.pod
|   |   ├── ngx_core_module.pod
|   |   ├── ngx_google_perftools_module.pod
|   |   ├── ngx_http_access_module.pod
|   |   ├── ngx_http_addition_module.pod
|   |   ├── ngx_http_api_module_head.pod
|   |   ├── ngx_http_auth_basic_module.pod
|   |   ├── ngx_http_auth_jwt_module.pod
|   |   ├── ngx_http_auth_request_module.pod
|   |   ├── ngx_http_autoindex_module.pod
|   |   ├── ngx_http_browser_module.pod
|   |   ├── ngx_http_charset_module.pod
|   |   ├── ngx_http_core_module.pod
|   |   ├── ngx_http_dav_module.pod
|   |   ├── ngx_http_empty_gif_module.pod
|   |   ├── ngx_http_f4f_module.pod
|   |   ├── ngx_http_fastcgi_module.pod
|   |   ├── ngx_http_flv_module.pod
|   |   ├── ngx_http_geoip_module.pod
|   |   ├── ngx_http_geo_module.pod
|   |   ├── ngx_http_grpc_module.pod
|   |   ├── ngx_http_gunzip_module.pod
|   |   ├── ngx_http_gzip_module.pod
|   |   ├── ngx_http_gzip_static_module.pod
|   |   ├── ngx_http_headers_module.pod
|   |   ├── ngx_http_hls_module.pod
|   |   ├── ngx_http_image_filter_module.pod
|   |   ├── ngx_http_index_module.pod
|   |   ├── ngx_http_js_module.pod
|   |   ├── ngx_http_keyval_module.pod
|   |   ├── ngx_http_limit_conn_module.pod
|   |   ├── ngx_http_limit_req_module.pod
|   |   ├── ngx_http_log_module.pod
|   |   ├── ngx_http_map_module.pod
|   |   ├── ngx_http_memcached_module.pod
|   |   ├── ngx_http_mirror_module.pod
|   |   ├── ngx_http_mp4_module.pod
|   |   ├── ngx_http_perl_module.pod
|   |   ├── ngx_http_proxy_module.pod
|   |   ├── ngx_http_random_index_module.pod
|   |   ├── ngx_http_realip_module.pod
|   |   ├── ngx_http_referer_module.pod
|   |   ├── ngx_http_rewrite_module.pod
|   |   ├── ngx_http_scgi_module.pod
|   |   ├── ngx_http_secure_link_module.pod
|   |   ├── ngx_http_session_log_module.pod
|   |   ├── ngx_http_slice_module.pod
|   |   ├── ngx_http_spdy_module.pod
|   |   ├── ngx_http_split_clients_module.pod
|   |   ├── ngx_http_ssi_module.pod
|   |   ├── ngx_http_ssl_module.pod
|   |   ├── ngx_http_status_module.pod
```

```
|   |   ├── ngx_http_stub_status_module.pod
|   |   ├── ngx_http_sub_module.pod
|   |   ├── ngx_http_upstream_conf_module.pod
|   |   ├── ngx_http_upstream_hc_module.pod
|   |   ├── ngx_http_upstream_module.pod
|   |   ├── ngx_http_userid_module.pod
|   |   ├── ngx_http_uwsgi_module.pod
|   |   ├── ngx_http_v2_module.pod
|   |   ├── ngx_http_xslt_module.pod
|   |   ├── ngx_mail_auth_http_module.pod
|   |   ├── ngx_mail_core_module.pod
|   |   ├── ngx_mail_imap_module.pod
|   |   ├── ngx_mail_pop3_module.pod
|   |   ├── ngx_mail_proxy_module.pod
|   |   ├── ngx_mail_smtp_module.pod
|   |   ├── ngx_mail_ssl_module.pod
|   |   ├── ngx_stream_access_module.pod
|   |   ├── ngx_stream_core_module.pod
|   |   ├── ngx_stream_geoip_module.pod
|   |   ├── ngx_stream_geo_module.pod
|   |   ├── ngx_stream_js_module.pod
|   |   ├── ngx_stream_keyval_module.pod
|   |   ├── ngx_stream_limit_conn_module.pod
|   |   ├── ngx_stream_log_module.pod
|   |   ├── ngx_stream_map_module.pod
|   |   ├── ngx_stream_proxy_module.pod
|   |   ├── ngx_stream_realip_module.pod
|   |   ├── ngx_stream_return_module.pod
|   |   ├── ngx_stream_split_clients_module.pod
|   |   ├── ngx_stream_ssl_module.pod
|   |   ├── ngx_stream_ssl_preread_module.pod
|   |   ├── ngx_stream_upstream_hc_module.pod
|   |   ├── ngx_stream_upstream_module.pod
|   |   ├── ngx_stream_zone_sync_module.pod
|   |   ├── request_processing.pod
|   |   ├── server_names.pod
|   |   ├── stream_processing.pod
|   |   ├── switches.pod
|   |   ├── syntax.pod
|   |   ├── sys_errlist.pod
|   |   ├── syslog.pod
|   |   ├── variables_in_config.pod
|   |   ├── websocket.pod
|   |   ├── welcome_nginx_facebook.pod
|   |   └── windows.pod
|   ├── ngx_coolkit-0.2
|   ├── ngx_devel_kit-0.3.1
|   |   ├── ngx_devel_kit-0.3.1.pod
|   |   └── readme_auto_lib.pod
|   ├── ngx_lua-0.10.17
|   |   └── ngx_lua-0.10.17.pod
|   ├── ngx_lua_upstream-0.07
|   |   └── ngx_lua_upstream-0.07.pod
|   ├── ngx_postgres-1.0
|   |   ├── ngx_postgres-1.0.pod
|   |   └── todo.pod
|   ├── ngx_stream_lua-0.0.8
|   |   ├── dev_notes.pod
```

```
|   |   └── ngx_stream_lua-0.0.8.pod
|   ├── opm-0.0.5
|   |   └── opm-0.0.5.pod
|   ├── rds-csv-nginx-module-0.09
|   |   └── rds-csv-nginx-module-0.09.pod
|   ├── rds-json-nginx-module-0.15
|   |   └── rds-json-nginx-module-0.15.pod
|   ├── redis2-nginx-module-0.15
|   |   └── redis2-nginx-module-0.15.pod
|   ├── redis-nginx-module-0.3.7
|   ├── resty-cli-0.25
|   |   └── resty-cli-0.25.pod
|   ├── set-misc-nginx-module-0.32
|   |   └── set-misc-nginx-module-0.32.pod
|   ├── srcache-nginx-module-0.32
|   |   └── srcache-nginx-module-0.32.pod
|   └── xss-nginx-module-0.06
|       └── xss-nginx-module-0.06.pod
├── resty.index
└── site
    ├── lualib
    ├── manifest
    └── pod

83 directories, 313 files
[root@centos8 openresty-1.17.8.2]#

[root@centos8 openresty-1.17.8.2]#ln -s  /apps/openresty/bin/* /usr/bin/
[root@centos8 openresty-1.17.8.2]#openresty  -v
nginx version: openresty/1.17.8.2

[root@centos8 openresty-1.17.8.2]#openresty
[root@centos8 openresty-1.17.8.2]#ss -ntl
State             Recv-Q            Send-Q        Local Address:Port   Peer
Address:Port
LISTEN            0                 128                 0.0.0.0:80
 0.0.0.0:*
LISTEN            0                 128                 0.0.0.0:22
 0.0.0.0:*
LISTEN            0                 100               127.0.0.1:25
 0.0.0.0:*
LISTEN            0                 128                    [::]:22
[::]:*
LISTEN            0                 100                   [::1]:25
[::]:*

[root@centos8 openresty-1.17.8.2]#ps -ef |grep nginx
root       16682       1  0 13:50 ?        00:00:00 nginx: master process
openresty
nginx      16683   16682  0 13:50 ?        00:00:00 nginx: worker process
root       16692    1195  0 13:51 pts/1    00:00:00 grep --color=auto nginx

[root@centos8 ~]#curl 10.0.0.18
<!DOCTYPE html>
<html>
<head>
<title>Welcome to OpenResty!</title>
<style>
```

```
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to OpenResty!</h1>
<p>If you see this page, the OpenResty web platform is successfully installed
and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="https://openresty.org/">openresty.org</a>.<br/>
Commercial support is available at
<a href="https://openresty.com/">openresty.com</a>.</p>

<p><em>Thank you for flying OpenResty.</em></p>
</body>
</html>
```



7 系统参数优化

默认的Linux内核参数考虑的是最通用场景，不符合用于支持高并发访问的Web服务器的定义，根据业务特点来进行调整，当Nginx作为静态web内容服务器、反向代理或者提供压缩服务器的服务器时，内核参数的调整都是不同的，此处针对最通用的、使Nginx支持更多并发请求的TCP网络参数做简单的配置

7.1 优化内核参数

修改/etc/sysctl.conf

```
fs.file-max = 1000000
#表示单个进程较大可以打开的句柄数

net.ipv4.tcp_tw_reuse = 1
#参数设置为 1 ，表示允许将TIME_WAIT状态的socket重新用于新的TCP链接，这对于服务器来说意义重大，因为总有大量TIME_WAIT状态的链接存在

net.ipv4.tcp_keepalive_time = 600
#当keepalive启动时，TCP发送keepalive消息的频度;默认是2小时，将其设置为10分钟，可更快的清理无效链接

net.ipv4.tcp_fin_timeout = 30
#当服务器主动关闭链接时，socket保持在FIN_WAIT_2状态的较大时间
```

```
net.ipv4.tcp_max_tw_buckets = 5000
#表示操作系统允许TIME_WAIT套接字数量的较大值，如超过此值，TIME_WAIT套接字将立刻被清除并打印警告信息,默认为8000，过多的TIME_WAIT套接字会使Web服务器变慢

net.ipv4.ip_local_port_range = 1024 65000
#定义UDP和TCP链接的本地端口的取值范围

net.ipv4.tcp_rmem = 10240 87380 12582912
#定义了TCP接受缓存的最小值、默认值、较大值

net.ipv4.tcp_wmem = 10240 87380 12582912
#定义TCP发送缓存的最小值、默认值、较大值

net.core.netdev_max_backlog = 8096
#当网卡接收数据包的速度大于内核处理速度时，会有一个列队保存这些数据包。这个参数表示该列队的较大值

net.core.rmem_default = 6291456
#表示内核套接字接受缓存区默认大小

net.core.wmem_default = 6291456
#表示内核套接字发送缓存区默认大小

net.core.rmem_max = 12582912
#表示内核套接字接受缓存区较大大小

net.core.wmem_max = 12582912
#表示内核套接字发送缓存区较大大小
注意：以上的四个参数，需要根据业务逻辑和实际的硬件成本来综合考虑

net.ipv4.tcp_syncookies = 1
#与性能无关。用于解决TCP的SYN攻击

net.ipv4.tcp_max_syn_backlog = 8192
#这个参数表示TCP三次握手建立阶段接受SYN请求列队的较大长度，默认1024，将其设置的大一些可使出现Nginx繁忙来不及accept新连接时，Linux不至于丢失客户端发起的链接请求

net.ipv4.tcp_tw_recycle = 1
#这个参数用于设置启用timewait快速回收

net.core.somaxconn=262114
#选项默认值是128，这个参数用于调节系统同时发起的TCP连接数，在高并发的请求中，默认的值可能会导致链接超时或者重传，因此需要结合高并发请求数来调节此值。

net.ipv4.tcp_max_orphans=262114
#选项用于设定系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤立链接将立即被复位并输出警告信息。这个限制指示为了防止简单的DOS攻击，不用过分依靠这个限制甚至认为的减小这个值，更多的情况是增加这个值
```

7.2 PAM 资源限制优化

在/etc/security/limits.conf 最后增加：

```
* soft nofile 65535
* hard nofile 65535
* soft nproc 65535
* hard nproc 65535
```

8 常见面试题

- http 协议的各个版本和区别
- 比较nginx 和 apache 的特性
- nginx 的功能
- 常见的nginx 性能优化方法
- location的优先级
- rewrite 中 redirect,permanent, break和last区别
- 如何获取客户端真实IP
- nginx的常见模块
- nginx的反向代理的调度算法
- 比较nginx，haprorxy和LVS的区别
- http的常见的响应码
- 访问web页面时出现502和504错误，请简述一下排查思路
- 有同事反应，网站访问速度很慢，有时候会出现打不开网站的情况，刷新等待好长时间后又正常打开，请分析并说一说故障排查思路?
- 上家公司的服务各做了哪些优化，NGINX做了哪些优化