1. Nginx简介
   1. 什么是nginx

Nginx是俄罗斯人编写的十分轻量级的HTTP服务器,Nginx，它的发音为“engine X”， 是一个高性能的HTTP和反向代理服务器，同时也是一个IMAP/POP3/SMTP 代理服务器。Nginx是由俄罗斯人 Igor Sysoev为俄罗斯访问量第二的 Rambler.ru站点开发的，它已经在该站点运行超过两年半了。Igor Sysoev在建立的项目时，使用基于BSD许可。自Nginx 发布来，Nginx 已经因为它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。

* 1. Nginx的优点
     + 1. **高并发连接：**

官方测试能够支撑5万并发连接，在实际生产环境中跑到2～3万并发连接数。

* + - 1. **内存消耗少：**

在3万并发连接下，开启的10个Nginx 进程才消耗150M内存（15M\*10=150M）。

* + - 1. **配置文件非常简单：**

风格跟程序一样通俗易懂。

* + - 1. **成本低廉：**

Nginx为开源软件，可以免费使用。而购买F5 BIG-IP、NetScaler等硬件负载均衡交换机则需要十多万至几十万人民币。

* + - 1. **支持Rewrite重写规则**：

能够根据域名、URL的不同，将 HTTP 请求分到不同的后端服务器群组。

* + - 1. **内置的健康检查功能：**

如果 Nginx Proxy 后端的某台 Web 服务器宕机了，不会影响前端访问。

* + - 1. **节省带宽：**

支持 GZIP 压缩，可以添加浏览器本地缓存的 Header 头。

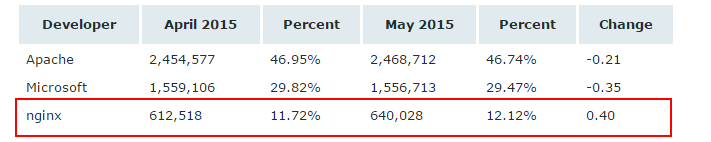
* + - 1. **稳定性高：**

用于反向代理，宕机的概率微乎其微。

Nginx专为性能优化而开发，性能是其最重要的考量,实现上非常注重效率 。它支持内 核Poll模型，能经受高负载的考验,有报告表明能支持高达 50,000个并发连接数。 Nginx具有很高的稳定性，其它HTTP服务器当遇到访问的峰值，或者有人恶意发起慢速连接时，也很可能会导致服务器物理内存耗尽频繁交换，失去响应只能重启服务器。例如当前Apache一旦上到200个以上进程，web响应速度就明显非常缓慢了。而Nginx采取了分阶段资源分配技术，使得它的CPU与内存占用率非常低。Nginx官方表示保持 10,000个没有活动的连接，它只占2.5M内存，所以类似DOS这样的攻击对Nginx来说基本上是毫无用处的。就稳定性而言，nginx比lighttpd更胜一筹。 Nginx支持热部署，它的启动特别容易, 并且几乎可以做到7\*24不间断运行，即使运 行数个月也不需要重新启动。你还能够在不间断服务的情况下，对软件版本进行进行升级。 Nginx采用master-slave模型，能够充分利用SMP的优势，且能够减少工作进程在磁 盘I/O的阻塞延迟。 Nginx代码质量非常高，代码很规范，手法成熟， 模块扩展也很容易。 Nginx采用了一些os提供的最新特性如对sendfile (Linux2.2+)，accept-filter(FreeBSD4.1+)，TCP\_DEFER\_ACCEPT (Linux 2.4+)的支持，从而大大提高了性 能。

* 1. 哪里使用到nginx

**Nginx所占市场份额不断提升**：



**国内有企业有哪些在使用nginx服务器：**

国内使用Nginx的网站：新浪、网易、腾讯、CSDN、酷六、水木社区、豆瓣、六房间、小米等。

* 1. Nginx和Apache的区别

Nginx和Apache一样，都是一个HTTP服务器软件，功能实现上都采用模块化结构设计，都支持通用的语言接口，如PHP、Perl、Python等，同时还支持正、反向代理，虚拟主机，URL重写，压缩传输，SSL加密传输等。它们之间最大的差别是Apache处理速度很慢，且占用很多内存资源，而Nginx却恰恰相反；在功能实现上，Apache的所有模块都支持动、静态编译，而Nginx模块都是静态编译的，同时，Apache对Fcgi支持不好，而Nginx对Fcgi的支持非常的好；最重要的是，在处理连接方式上，Nginx支持epoll，而Apache却不支持；在大小上，Nginx安装包仅仅有几百K，和Nginx比起来Apache绝对是庞然大物。在了解了Nginx和Apache之间的异同点后基本知道了Nginx作为HTTP服务器的优势所在。

1. 安装Nginx服务器
   1. 在windows上安装
      * 1. 下载nginx的windows版本

<http://nginx.org/download/nginx-1.8.0.zip>

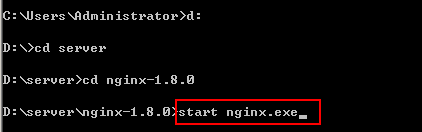
* + - 1. 将压缩文件解压到非中文的目录下（绿色软件只需要压缩即可）



* + - 1. Nginx的启动和停止（前提：**确保apache关闭**）

1. **启动**

进入解压的目录下执行start nginx.exe 命令



启动后在浏览器中访问到以下界面说明安装启动成功



1. **停止**

nginx -s stop 快速关闭

nginx -s quit 安全关闭

nginx -s reload 重载配置文件

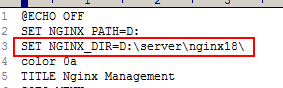
nginx -s reopen 重新打开日志文件



1. **配置重启和关闭的控制器台**
   * + 1. 拷贝nginx.bat文件到nginx文件夹下



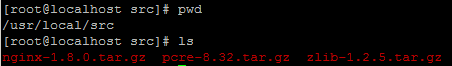
* + - 1. 修改nginx.bat中配置nginx的安装路径



* + - 1. 使用nginx.bat控制nginx



* 1. 在Linux上安装
     1. 写在前面
        1. 软件的源文件都放在/usr/local/src

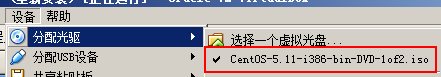


* + - 1. 软件安装在/usr/local/
    1. 准备使用yum安装nginx的运行环境
       1. **将光驱作为yum源**

yum安装时首先到网络下载相关软件.如果将光驱作为yum源的话就不需要从网络上下载而是从光驱中找到相关软件.

* 在虚拟机中选中光盘  
  选择centos的第一张光盘所在位置.



选择后的效果.  


* 挂载光驱到/mnt/cdrom文件下





* 创建配置文件并且写入以下配置

vi /etc/yum.repos.d/cdrom.repo

配置:

[cdrom]

name=CentOS5 - cdrom

baseurl=file:///mnt/cdrom/

enable=1

gpgcheck=0

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5

* 查看被用到的yum源

yum repolis

* + - 1. **使用yum安装编译环境**

yum -y install automake autoconf libtool make

yum -y install gcc gcc-c++

* + 1. 安装pcre

cd /usr/local/src

tar -zxvf pcre-8.32.tar.gz

cd pcre-8.32

./configure --prefix=/usr/local/pcre

make && make install

* + 1. 安装zlib库

cd /usr/local/src

tar -zxvf zlib-1.2.5.tar.gz

cd zlib-1.2.5

./configure --prefix=/usr/local/zlib

make && make install

* + 1. 安装nginx

cd /usr/local/src

tar -zxvf nginx-1.8.0.tar.gz

cd nginx-1.8.0

./configure --prefix=/usr/local/nginx \

--sbin-path=/usr/local/nginx/nginx \

--conf-path=/usr/local/nginx/nginx.conf \

--pid-path=/usr/local/nginx/nginx.pid \

--with-pcre=/usr/local/src/pcre-8.32 \

--with-zlib=/usr/local/src/zlib-1.2.5

make && make install

通过浏览器访问虚拟机的ip地址得到nginx页面说明安装成功!



* + 1. 控制nginx

到安装目录下:

cd /usr/local/nginx

检查配置文件是否正确

./nginx -t

启动操作

./nginx -c /usr/local/nginx/nginx.conf

-c参数指定了要加载的nginx配置文件路径. 默认为安装配置时指定的配置文件路径.

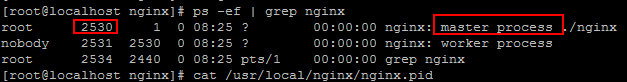
停止操作

停止操作是通过向nginx进程发送信号来进行的

步骤1：查询nginx主进程号

ps -ef | grep nginx

在进程列表里 面找master进程，它的编号就是主进程号了。



步骤2：发送信号

从容停止Nginx：

kill -QUIT 主进程号

快速停止Nginx：

kill -TERM 主进程号

强制停止Nginx：

pkill -9 主进程号

另外， 若在nginx.conf配置了pid文件存放路径则该文件存放的就是Nginx主进程号，如果没指定则放在nginx的logs目录下。有了pid文 件，我们就不用先查询Nginx的主进程号，而直接向Nginx发送信号了，命令如下：

kill -信号类型(QUIT| TERM) '/usr/local/nginx/nginx.pid'

平滑重启

如果更改了配置就要重启Nginx，要先关闭Nginx再打开？不是的，可以向Nginx 发送信号，平滑重启。

平滑重启命令：

kill -HUP 住进称号或进程号文件路径

或者使用

./nginx -s reload

* + 1. nginx安装服务

如果想通过以下命令控制nginx:

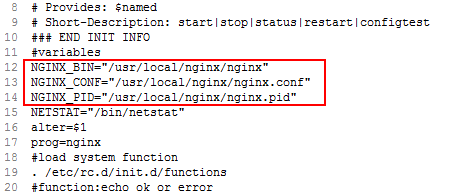
service nginx start

service nginx stop

需要在linux中安装服务脚本:



* + - 1. 编辑并且正确配置该脚本中的变量.如下:



* + - 1. 做以下的操作:

#拷脚本到/etc/init.d/

cp nginx /etc/init.d/

#修改服务脚本的执行权限

chmod 755 /etc/init.d/nginx

#nginx加入服务

chkconfig --add nginx

#nginx 设置为启动

/sbin/chkconfig nginx on

* + - 1. 服务器脚本支持以下功能



#启动

service nginx start

#关闭

service nginx stop

#重启

service nginx restart

#状态

service nginx status

#检查配置文件

service nginx configtest

1. Nginx的配置文件详解
   1. Nginx的主配置文件概述
      1. 认识配置文件

Linux 下基本上每个服务都会有它的主配置文件，该文件会定义服务应该如果去运行，

使用些什么参数，启用些什么功能，相关会涉及到的一些操作文件在哪，所以主配置文件对 服务是至关重要的。下面我们来分析一下 Nginx 的主配置文件。

Nginx的主配置文件默认情况下位于/usr/local/nginx/nginx.conf 以下为Nginx

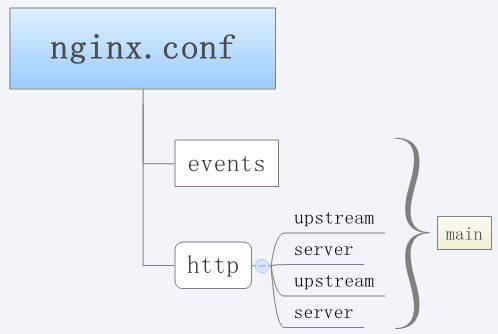
配置文件一些参数的注释。

请先参考该配置文件:



* + 1. ****nginx的配置文件结构****

nginx.conf由多个块组成，最外面的块是main，main包含events和http,http包含多个upstream和多个server，server又包含多个location：



main（全局设置）、server（虚拟主机设置）、upstream（负载均衡服务器设置）和 location（URL匹配特定位置的设置）。

* main块设置的指令将影响其他所有设置；
* server块的指令主要用于指定主机和端口；
* upstream指令主要用于负载均衡，设置一系列的后端服务器；
* location块用于匹配网页位置。

这四者之间的关系式：server继承main，location继承server，upstream既不会继承其他设置也不会被继承。

在这四个部分当中，每个部分都包含若干指令，这些指令主要包含Nginx的主模块指令、事件模块指令、HTTP核心模块指令，同时每个部分还可以使用其他HTTP模块指令，例如Http SSL模块、HttpGzip Static模块和Http Addition模块等。

* + 1. ****nginx的全局配置****

#指定使用的用户和组

#user nginx nginx;

#启动进程,通常设置成和cpu的数量相等

worker\_processes 1;

#全局错误日志

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#PID文件--存放进程号的文件

#pid logs/nginx.pid;

创建全局配置中nginx需要的运行用户

groupadd nginx

useradd -g nginx nginx

* 1. events配置

#工作模式及连接数上限

events {

#单个后台worker process进程的最大并发链接数

worker\_connections 1024;

#并发总数是 worker\_processes 和 worker\_connections 的乘积

#Nginx支持如下处理连接的方法（I/O复用方法），这些方法可以通过 use指令指定。

#use [ kqueue | rtsig | epoll | /dev/poll | select | poll];

use epoll; #使用 epoll（linux2.6的 性能方式 ）

}

服务器为linux时, 请讲use epoll打开. 利用linux的内核提供性能优化方案.

* 1. http的配置

http {

#Nginx对HTTP服务器相关属性的配置

include mime.types;

default\_type application/octet-stream;

#设定虚拟主机配置

server {

#侦听80端口

listen 80;

#定义使用 www.itsource.cn访问

server\_name [www.itsource.cn](http://www.itsource.cn);

location {

}

location {

}

…..

}

server {

#侦听80端口

listen 80;

#定义使用 www.example.cn访问

server\_name [www.example.cn](http://www.example.cn);

location {

}

location {

}

…..

}

}

include是个主模块指令，实现对配置文件所包含的文件的设定，可以减少主配置文件的复杂度。类似于Apache中的include方法。

default\_type属于HTTP核心模块指令，这里设定默认类型为二进制流，也就是当文件类型未定义时使用这种方式，例如在没有配置PHP环境时，Nginx是不予解析的，此时，用浏览器访问PHP文件就会出现下载窗口。

server块的指令主要用于指定主机和端口(虚拟主机)；

location部分主要用于匹配网页位置,设置不同的功能特征. 比如:缓存,重定向等..

* 1. [nginx重要指令之location](http://www.nginx.cn/115.html)

location部分主要用于匹配网页位置,设置不同的功能特征. 比如:缓存,重定向等..

**实例:**

有扩展名以.gif、.jpg、.jpeg、.png、.bmp、.swf结尾的静态文件都交给nginx处理，而expires用来指定静态文件的过期时间，这里是30天

location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf)$ {

root /wwwroot/www.itsource.com;

expires 30d;

}

upload和html下的所有文件都交给nginx来处理，当然，upload和html目录包含在/web/wwwroot/www.itsource.cn目录中

location ~ ^/(upload|html)/ {

root /web/wwwroot/www.itsource.com;

expires 30d;

}

location是对此虚拟主机下动态网页的过滤处理，也就是将所有以.php为后缀的文件都交给本机的8080端口处理

location ~ .\*.php$ {

index index.php;

proxy\_pass http://localhost:8080;

}

**location的语法:**

* ~      #波浪线表示执行一个正则匹配，区分大小写
* ~\*    #表示执行一个正则匹配，不区分大小写
* ^~    #^~表示普通字符匹配，如果该选项匹配，只匹配该选项，不匹配别的选项，一般用来匹配目录
* =      #进行普通字符精确匹配

location中可以接受一下的语法规则

location = / {

# 只匹配"/".

[ configuration A ]

}

location / {

# 匹配任何请求，因为所有请求都是以"/"开始

# 但是更长字符匹配或者正则表达式匹配会优先匹配

[ configuration B ]

}

location ^~ /images/ {

# 匹配任何以 /images/ 开始的请求，并停止匹配 其它location

[ configuration C ]

}

location ~\* \.(gif|jpg|jpeg)$ {

# 匹配以 gif, jpg, or jpeg结尾的请求.

# 但是所有 /images/ 目录的请求将由 [Configuration C]处理.

[ configuration D ]

}

**优先级:**

* + - 1. =前缀的指令严格匹配这个查询。如果找到，停止搜索。
      2. 所有剩下的常规字符串，最长的匹配。如果这个匹配使用^〜前缀，搜索停止。
      3. 正则表达式，在配置文件中定义的顺序。
      4. 如果第3条规则产生匹配的话，结果被使用。否则，如同从第2条规则被使用。

请求URI例子:

/ -> 符合configuration A

/documents/document.html -> 符合configuration B

/images/1.gif -> 符合configuration C

/documents/1.jpg ->符合 configuration D

1. nginx中的rewrite
   1. 什么是rewrite

Rewrite对称URL Rewrite，即URL重写，就是把传入Web的请求重定向到其他

URL的过程。URL Rewrite最常见的应用是URL伪静态化，是将动态页面显示为静态页面方式的一种技术。比如http://www.123.com/news/index.asp?id=123 使用URLRewrite 转换后可以显示为 <http://www.123.com/news/123.html>

对于追求完美主义的网站设计师，就算是网页的地址也希望看起来尽量简洁明快。形如

http://www.123.com/news/index.asp?id=123的网页地址，自然是毫无美感可言，而用UrlRewrite技术，你可以轻松把它显示为 http://www.123.com/news/123.html。

理论上，搜索引擎更喜欢静态页面形式的网页，搜索引擎对静态页面的评分一般要高于

动态页面。所以，UrlRewrite可以让我们网站的网页更容易被搜索引擎所收录。

从安全角度上讲，如果在url中暴露太多的参数，无疑会造成一定量的信息泄漏，可能会被一些黑客利用，对你的系统造成一定的破坏，所以静态化的url地址可以给我们带来更高的安全性。

* 1. rewrite的命令的作用域和优先级

作用域：server,location.

rewrite指令可以写在server和location中.

nginx rewrite指令执行顺序：

* + - 1. 执行server块的rewrite指令(这里的块指的是server关键字后{}包围的区域，其它xx块类似)
      2. 执行location匹配
      3. 执行选定的location中的rewrite指令  
         如果其中某步URI被重写，则重新循环执行1-3，直到找到真实存在的文件

例如:



* 1. if指令
     1. if指令的语法

语法：if(condition){...}  
默认值：无  
作用域：server,location

* + 1. if指令中使用的逻辑运算符

对给定的条件condition进行判断。如果为真，大括号内的rewrite指令将被执行。  
if条件(conditon)可以是如下任何内容:

* + - 1. 一个变量名；false如果这个变量是空字符串或者以0开始的字符串；
      2. 使用= ,!= 比较的一个变量和字符串
      3. 是用~， ~\*与正则表达式匹配的变量，如果这个正则表达式中包含}，;则整个表达式需要用" 或' 包围
      4. 使用-f ，!-f 检查一个文件是否存在
      5. 使用-d, !-d 检查一个目录是否存在
      6. 使用-e ，!-e 检查一个文件、目录、符号链接是否存在
      7. 使用-x ， !-x 检查一个文件是否可执行
    1. If指令中可以使用的变量

在匹配过程中可以引用一些Nginx的全局变量，更多的变量请参考

http://wiki.nginx.org/NginxHttpCoreModule 的 Variables 部分

* $args, 请求中的参数;
* $document\_root, 针对当前请求的根路径设置值;
* $host, 请求信息中的"Host"，如果请求中没有Host行，则等于设置的服务器名;
* $limit\_rate, 对连接速率的限制;
* $request\_method, 请求的方法，比如"GET"、"POST"等;
* $remote\_addr, 客户端地址;
* $remote\_port, 客户端端口号;
* $remote\_user, 客户端用户名，认证用;
* $request\_filename, 当前请求的文件路径名
* $query\_string, 与$args相同;
* $scheme, 所用的协议，比如http或者是https
* $server\_protocol, 请求的协议版本，"HTTP/1.0"或"HTTP/1.1";
* $server\_addr, 服务器地址，如果没有用listen指明服务器地址，使用这个变量将发起一次系统调用
* 以取得地址(造成资源浪费);
* $server\_name, 请求到达的服务器名;
* $document\_uri 与$uri一样，URI地址;
* $server\_port, 请求到达的服务器端口号;
  + 1. if指令实例
       1. 匹配访问的url地址是否是个目录

if (-d $request\_filename) {

…;

}

* + - 1. 匹配访问的地址是否以www开头

if ($hosts ~\* ^www) {

…;

}

* 1. rewrite指令
     1. rewrite指令语法

语法：rewrite regex replacement [flag];

默认值：无

作用域：server,location,if

regex: 请求地址,该地址可以是一个正则表达式

replacement: 重写地址.

flag: rewrite标记(last,break,redirect,permanent)

* + 1. flag标记

rewrite 指令根据表达式来重定向URI，或者修改字符串。可以应用于server,

location, if环境下 每行rewrite指令最后应该根一个flag标记，支持的flag标记有:

last 相当于Apache里的[L]标记，表示完成rewrite

break 本条规则匹配完成后，终止匹配，不再匹配后面的规则

redirect 返回302临时重定向，浏览器地址会显示跳转后的URL地址

permanent 返回301永久重定向，浏览器地址会显示跳转后URL地址

* + - * last和break标记的区别在于，last标记在本条rewrite规则执行完后，会对其所在的
      * server { … } 标签重新发起请求，而break标记则在本条规则匹配完成后，停止匹配，不
      * 再做后续的匹配。另有些时候必须使用last，比如在使用alias指令时，而使用proxy\_pass
      * 指令时则必须使用break。

location /abc/ {

rewrite “^/abc/(.\*)\.html$” /abc/index.html last;

}

我们应该将上面的last改成break以避免死循环。

**redirect和 permanent区别:**

redirect和 permanent区别则是返回的不同方式的重定向，对于客户端来说一般状态下是没有区别的。而对于搜索引擎，相对来说301的重定向更加友好，如果我们把一个地址采用301跳转方式跳转的话，搜索引擎会把老地址的相关信息带到新地址，同时在搜索引擎索引库中彻底废弃掉原先的老地址。

使用302重定向时，搜索引擎(特别是google)有时会查看跳转前后哪个网址更直观，

然后决定显示哪个，如果它觉的跳转前的URL更好的话，也许地址栏不会更改，那么很有

可能出现URL劫持的现像。

我们在做URI重写时，有时会发现URI中含有相关参数，如果需要将这些参数保存下来，并且在重写过程中重新引用，我们可以用到 () 和 $N 的方式来解决。

* + 1. set指令

set 指令是用于定义一个变量，并且赋值。应用于server,location,if环境。

语法格式为: set　$变量名　变量值

* + 1. return指令

return 指令用于返回状态码给客户端, 应用于server，location，if环境。

语法:

return xxx;

* + 1. rewrite实例
       1. 匹配访问的url地址是否是个目录，如果是则自动加个 /

if (-d $request\_filename) {

rewrite ^/(.\*)([^/])$ http://$host/$1$2/ permanent;

}

* + - 1. 用户访问的网址为www.test.com/abc.html 重写后真实地址是

www.test.com/login.php?user=abc

location ~\* /php/.\*\.html$ {

rewrite /php/(.\*)\.html /login.php?user=$1 last;

}

* + - 1. 用户访问地址为/uplook/11-22-33.html重写后真实地址为 /uplook/11/22/33.html

location /uplook/ {

rewrite /uplook/([0-9]+)-([0-9]+)-([0-9]+).html /uplook/$1/$2/$3.html last;

}

* + - 1. 当访问任意目录下的whoami.html都重定向到 /who.html

location ~\* .\*/whoami\.html$ {

set $who 'who.html';

rewrite .\* /$who break;

}

* + - 1. 如果访问的　.sh　结尾的文件则返回403操作拒绝错误

location ~\* .\*\.sh$ {

return 403;

}

1. nginx的虚拟主机
   1. 什么是nginx的虚拟主机

虚拟主机是用来映射网站目录和网站代码文件夹的关系.它可以通过server配置, 每个server表示一个虚拟机主机.

* 1. 标准的虚拟主机配置

http {

#默认的虚拟机主机

 server {

        listen          80 default;

        server\_name     \_ \*;

        access\_log      logs/default.access.log main;

        location / {

            index index.html;

            root  /var/www/default/htdocs;

        }

}

#www.domain1.com对应的虚拟主机

    server {

        listen          80;

        server\_name     www.domain1.com;

        access\_log      logs/domain1.access.log main;

        location / {

            index index.html;

            root  /var/www/domain1.com/htdocs;

        }

}

#www.domain2.com对应的虚拟主机

    server {

        listen          80;

        server\_name     www.domain2.com;

        access\_log      logs/domain2.access.log main;

        location / {

            index index.html;

            root  /var/www/domain2.com/htdocs;

        }

    }

}

* 1. 规划虚拟主机的配置文件

以上全部配置在主配置文件中可能不小心修改掉配置文件导致nginx.conf文件出错. 可以使用include指令将每个虚拟主机放在一个单独的配置文件中.

将nginx服务器目录结构设置为:

nginx

nginx.conf

vhost

-- www.domain1.com.conf

-- www.domain2.com.conf

每个.conf结尾的文件中保存一个虚拟主机(server)的配置信息.在nginx.conf中通过include指令包含这些配置文件.

**在nginx/config/nginx.conf中的配置:**

http {

#默认的虚拟机主机

 server {

        listen          80 default;

        server\_name     \_ \*;

        access\_log      logs/default.access.log main;

        location / {

            index index.html;

            root  /var/www/default/htdocs;

        }

}

include vhost/\*.conf;

}

**在nginx/config/vhost/www.domain1.com.conf中的配置:**

 server {

        listen          80;

        server\_name     www.domain1.com;

        access\_log      logs/domain1.access.log main;

        location / {

            index index.html;

            root  /var/www/domain1.com/htdocs;

        }

}

**在nginx/config/vhost/www.domain2.com.conf中的配置:**

 server {

        listen          80;

        server\_name     www.domain2.com;

        access\_log      logs/domain2.access.log main;

        location / {

            index index.html;

            root  /var/www/domain2.com/htdocs;

        }

}

1. 动静分离

当用户请求js,css等静态资源时有当前nginx服务器处理, 请求php动态脚本代码时由其他的服务器处理(反向代理).

 #配置Nginx动静分离，定义的静态页面直接从Nginx发布目录读取。

    location ~ .\*\.(gif|jpg|jpeg|bmp|png|ico|txt|js|css)$

    {

    root /data/www/wugk;

#expires定义用户浏览器缓存的时间为3天，如果静态页面不常更新，可以设置更长，这样可以节省带宽和缓解服务器的压力

    expires      3d;

}

  #动态页面交给其他服务器处理

    location ~ .\*\.(php|jsp|cgi)?$

    {

       #发送给其他服务器处理

    }

容量符号缩写

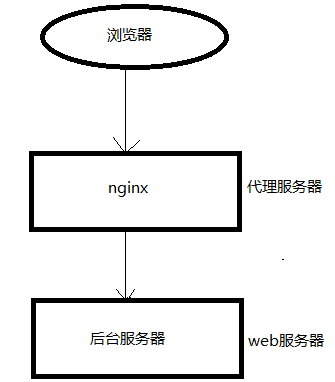
|  |  |
| --- | --- |
| k,K | 千字节 |
| m,M | 兆字节 |

时间符号缩写

|  |  |
| --- | --- |
| ms | 毫秒 |
| s | 秒 |
| m | 分钟 |
| h | 小时 |
| d | 日 |
| w | 周 |
| M | 一个月, 30天 |
| y | 年, 365 天 |

1. nginx的反向代理
   1. 什么是反向代理

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连的 客户端，此时代理服务器对外就表现为一个服务器。



* 1. 明确两个概念

web前端服务器: 代理服务器可以作为前端服务器处理静态资源

web后端服务器: web前端服务器无法处理的动态请求发送给后台web服务器处理.

* 1. 特点

反向代理又称为Web服务器加速，是针对Web服务器提供加速功能的。它作为代理

Cache，但并不针对浏览器用户，而针对一台或多台特定Web服务器（这也是反向代理名

称的由来）。代理服务器可以缓存一些web的页面，降低了web服务器的访问量，所以可

以降低web服务器的负载。web服务器同时处理的请求数少了，响应时间自然就快了。同

时代理服务器也存了一些页面，可以直接返回给客户端，加速客户端浏览。

* 1. 反向代理的配置

当用户放到php文件时,将该请求交给apache服务器处理php文件

server {

#侦听192.168.8.x的80端口

listen 80;

server\_name www.itsource.cn;

#对php后缀的进行请求

location ~ .\*\.php$ {

#定义服务器的默认网站根目录位置

root /root;

#定义首页索引文件的名称

index index.php index.html index.htm;

#请求转向apache服务器,apache服务器运行在81端口上

proxy\_pass http://localhost:81;

#以下是一些反向代理的配置可删除.

proxy\_redirect off;

#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

#允许客户端请求的最大单文件字节数

client\_max\_body\_size 10m;

#缓冲区代理缓冲用户端请求的最大字节数，

client\_body\_buffer\_size 128k;

#nginx跟后端服务器连接超时时间(代理连接超时)

proxy\_connect\_timeout 90;

#连接成功后，后端服务器响应时间(代理接收超时)

proxy\_read\_timeout 90;

#设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy\_buffer\_size 4k;

#proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置

proxy\_buffers 4 32k;

#高负荷下缓冲大小（proxy\_buffers\*2）

proxy\_busy\_buffers\_size 64k;

#设定缓存文件夹大小，大于这个值，将从upstream服务器传

proxy\_temp\_file\_write\_size 64k;

}

}

* 1. 可以将代理配置单独放在一个配置文件中

存放代理配置的nginx/conf/proxy.conf: 其他的server虚拟主机都可以使用该配置.

#以下是一些反向代理的配置可删除.

proxy\_redirect off;

#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

#允许客户端请求的最大单文件字节数

client\_max\_body\_size 10m;

#缓冲区代理缓冲用户端请求的最大字节数，

client\_body\_buffer\_size 128k;

#nginx跟后端服务器连接超时时间(代理连接超时)

proxy\_connect\_timeout 90;

#连接成功后，后端服务器响应时间(代理接收超时)

proxy\_read\_timeout 90;

#设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy\_buffer\_size 4k;

#proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置

proxy\_buffers 4 32k;

#高负荷下缓冲大小（proxy\_buffers\*2）

proxy\_busy\_buffers\_size 64k;

#设定缓存文件夹大小，大于这个值，将从upstream服务器传

proxy\_temp\_file\_write\_size 64k;

使用代理配置文件:

server {

listen 80;

server\_name www.itsource.cn;

#对php后缀的进行请求

location ~ .\*\.php$ {

#定义服务器的默认网站根目录位置

root /root;

#定义首页索引文件的名称

index index.php index.html index.htm;

#请求转向apache服务器,apache服务器运行在81端口上

proxy\_pass http://localhost:81;

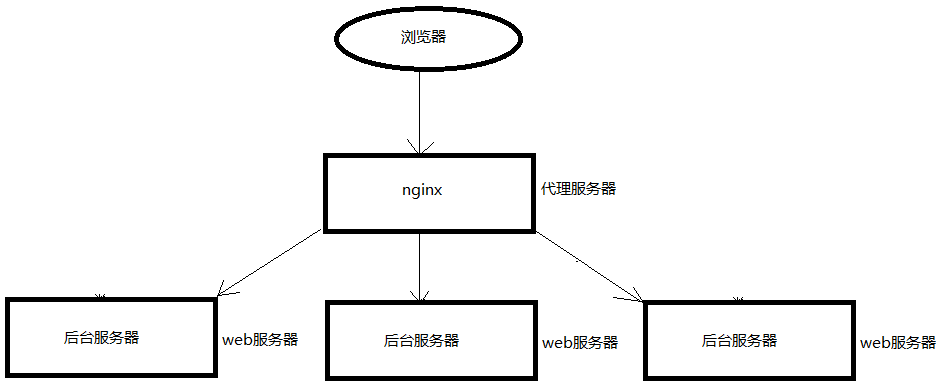
#以下是一些反向代理的配置可删除.

include proxy.conf;

}

}

1. nginx的负载均衡(自学)
   1. 什么是负载均衡



后台服务器组成了一个服务器集群(多台服务器). 有中间服务器(nginx)接受到请求分发给不同的服务器后台. 该nginx就是一个负载均衡服务器.

* 1. 负载均衡的优点
* 分散后台服务器的压力
* 自动去掉挂掉的后台服务器
* 缓存后台服务器响应内容
  1. 负载均衡的分配策略

nginx 的 upstream目前支持 4 种方式的分配   
**1)、轮询（默认）**  
      每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。   
**2)、weight**  
      指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。   
**2)、ip\_hash**  
      每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。    
**3)、fair（第三方）**   
      按后端服务器的响应时间来分配请求，响应时间短的优先分配。    
**4)、url\_hash（第三方）**

按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为缓存时比较有效。

* 1. 负载均衡配置

在http节点里添加:

#定义负载均衡设备的 ip及设备状态   
upstream myServer {

server 127.0.0.1:9090 down;   
    server 127.0.0.1:8080 weight=2;   
    server 127.0.0.1:6060;   
    server 127.0.0.1:7070 backup;

[**ip\_hash | fair | url\_hash**]  
}

在需要使用负载的server节点下添加

proxy\_pass http://myServer;

upstream 每个设备的状态:

down 表示单前的server暂时不参与负载   
weight  默认为1.weight越大，负载的权重就越大。   
max\_fails ：允许请求失败的次数默认为1.当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误   
fail\_timeout:max\_fails 次失败后，暂停的时间。   
backup： 其它所有的非backup机器down或者忙的时候，请求backup机器。所以这台机器压力会最轻。

1. 安装PHP
2. PHP-FPM
   1. 什么是PHP-FPM

PHP-FPM是一个PHPFastCGI管理器软件，是只用于PHP的。PHPFastCGI可以解析PHP脚本.在PHP5.3以及以上的版本中集成了PHP-FPM软件.

* 1. 为什么要是使用PHP-FPM

nginx本身不能处理PHP，它只是个web服务器，当接收到请求后，如果是php请求，则发给php解释器处理，并把结果返回给客户端。

* 1. 安装并且启动PHP-FPM
     1. 安装

PHP中已经集成了php-fpm,所以安装好PHP之后在PHP的安装目录:

/usr/local/php/sbin/php-fpm

就是php-fpm的可执行程序.

* + 1. fpm的配置

#准备配置文件php-fpm.conf

cp /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf

#修改配置文件

vi /usr/local/lnmp/php/etc/php-fpm.conf

去掉以下配置前的注释符号#

user = www

group = www

pid = run/php-fpm.pid

pm.start\_servers = 20

pm.min\_spare\_servers = 5

pm.max\_spare\_servers = 35

如果www用户和组不存在,请创建:

groupadd www   
useradd -g www www

* + 1. 启动和停止

/usr/local/php/sbin/php-fpm start

/usr/local/php/sbin/php-fpm stop

* + 1. 自启动php-fpm

#从php源代码文件中拷贝

cp /usr/loca/src/php-5.3.6/sapi/fpm/init.d.php-fpm /etc/init.d/php-fpm

#添加执行权限

chmod +x /etc/inti.d/php-fpm

# php-fpm加入服务

chkconfig --add php-fpm

# 设置自启动

chkconfig php-fpm on

* + 1. 检查php-fpm是否启动

php-fmp默认情况下运行在9000端口上.查看是否存在9000的端口,如果存在说明已经启动.

netstat –apn | grep 9000

* 1. nginx使用php-fpm处理php

在一个虚拟主机中加入一下配置即可:

当请求php文件时交给

location ~ \.php$ {  
root html;  
fastcgi\_pass 127.0.0.1:9000;  
fastcgi\_index index.php;  
fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  
include fastcgi\_params;  
}

在/usr/local/nginx/html下创建index.php文件，输入如下内容

<?php

echo phpinfo();

?>

直接访问index.php是否能够访问到. 如果能够访问到说明配置成功.