**Web服务器之Nginx详解（操作部分）**

大纲

一、前言

二、Nginx 安装与配置

三、Nginx 配置文件详解

四、Nginx 命令参数

五、配置Nginx提供Web服务

六、配置Nginx的虚拟主机

七、配置Nginx的用户认证

八、配置Nginx提供状态页面

九、配置Nginx的错误页面

十、配置Nginx打开目录浏览功能

十一、配置Nginx基于ssl提供https服务

注，测试环境 CentOS 6.4 x86\_64 , Nginx 1.4.2 (Nginx 最新版)

一、前言

在上一篇博文中我们讲解了Web服务器的工作原理与Nginx的基本特性，不清楚的博友可以参考一下这篇博文<http://freeloda.blog.51cto.com/2033581/1285332>，我们知道Nginx有两个基本功能，一个是作为Web服务器（在这篇博文中重点讲解），另一个是作为反向代理（在下面的博文中详细讲解）。好了，下面我们就来演示具体配置。首先我们要做的就是配置安装Nginx服务器。

二、Nginx 安装与配置

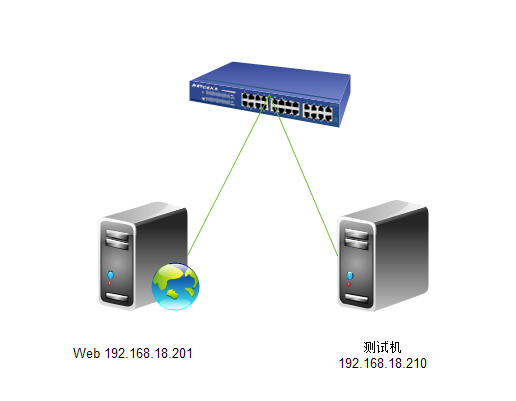
1.操作系统

* CenteOS 6.4 x86\_64

2.软件版本

* Nginx 1.4.2 （最新版）

3.实验拓扑

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846962eKm2.png)

注，实验拓扑很简单，一台Web服务器，一台Windows 7 测试机。

4.时间同步一下

|  |  |
| --- | --- |
| 1 | [root@web ~]# ntpdate 202.120.2.101 |

5.关闭防火墙与SELinux

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [root@web ~]# service iptables stop  [root@web ~]# chkconfig iptables off  [root@web ~]# chkconfig iptables --list  iptables           0:关闭    1:关闭    2:关闭    3:关闭    4:关闭    5:关闭    6:关闭  [root@web ~]# getenforce  Disabled |

6.安装yum源

|  |  |
| --- | --- |
| 1 | [root@web nginx]# rpm -ivh http://download.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm |

7.源码安装Nginx

(1).解压源码包

|  |  |
| --- | --- |
| 1  2 | [root@web src]# tar xf nginx-1.4.2.tar.gz  [root@web src]# cd nginx-1.4.2 |

(2).创建软链接

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [root@web local]# ln -sv nginx-1.4.2 nginx  "nginx" -> "nginx-1.4.2"  [root@web local]# cd nginx  [root@web nginx]# ll  总用量 588  drwxr-xr-x 6 1001 1001   4096 8月  29 17:32 auto  -rw-r--r-- 1 1001 1001 222366 7月  17 20:51 CHANGES  -rw-r--r-- 1 1001 1001 338680 7月  17 20:51 CHANGES.ru  drwxr-xr-x 2 1001 1001   4096 8月  29 17:32 conf  -rwxr-xr-x 1 1001 1001   2369 7月  17 20:51 configure  drwxr-xr-x 3 1001 1001   4096 8月  29 17:32 contrib  drwxr-xr-x 2 1001 1001   4096 8月  29 17:32 html  -rw-r--r-- 1 1001 1001   1397 7月  17 20:51 LICENSE  drwxr-xr-x 2 1001 1001   4096 8月  29 17:32 man  -rw-r--r-- 1 1001 1001     49 7月  17 20:51 README  drwxr-xr-x 8 1001 1001   4096 8月  29 17:32 src |

(3).新建nginx用户

|  |  |
| --- | --- |
| 1  2  3  4 | [root@web nginx]# groupadd -g 108  -r nginx  [root@web nginx]# useradd -u 108 -r -g 108 nginx  [root@web nginx]# id nginx  uid=108(nginx) gid=108(nginx) 组=108(nginx) |

(4).修改权限

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [root@web nginx]# chown -R root:nginx /usr/local/nginx/\*  [root@web nginx]# ll  总用量 588  drwxr-xr-x 6 root nginx   4096 8月  29 17:32 auto  -rw-r--r-- 1 root nginx 222366 7月  17 20:51 CHANGES  -rw-r--r-- 1 root nginx 338680 7月  17 20:51 CHANGES.ru  drwxr-xr-x 2 root nginx   4096 8月  29 17:32 conf  -rwxr-xr-x 1 root nginx   2369 7月  17 20:51 configure  drwxr-xr-x 3 root nginx   4096 8月  29 17:32 contrib  drwxr-xr-x 2 root nginx   4096 8月  29 17:32 html  -rw-r--r-- 1 root nginx   1397 7月  17 20:51 LICENSE  drwxr-xr-x 2 root nginx   4096 8月  29 17:32 man  -rw-r--r-- 1 root nginx     49 7月  17 20:51 README  drwxr-xr-x 8 root nginx   4096 8月  29 17:32 src |

(5).编译nginx

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | [root@web nginx]# yum -y install pcre-devel  [root@web nginx-1.4.2]# ./configure \    --prefix=/usr \     --sbin-path=/usr/sbin/nginx \     --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/nginx.pid  \     --lock-path=/var/lock/nginx.lock \     --user=nginx \     --group=nginx \     --with-http\_ssl\_module \     --with-http\_flv\_module \     --with-http\_stub\_status\_module \     --with-http\_gzip\_static\_module \     --http-client-body-temp-path=/var/tmp/nginx/client/ \     --http-proxy-temp-path=/var/tmp/nginx/proxy/ \     --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/ \     --http-uwsgi-temp-path=/var/tmp/nginx/uwsgi \     --with-pcre |

注，编译过程中会出错，下面是错误信息。

错误1：

|  |  |
| --- | --- |
| 1  2  3  4 | ./configure: error: SSL modules require the OpenSSL library.  You can either do not enable the modules, or install the OpenSSL library  into the system, or build the OpenSSL library statically from the source  with nginx by using --with-openssl=<path> option. |

解决1：

|  |  |
| --- | --- |
| 1 | [root@web nginx-1.4.2]# yum install -y openssl-devel |

再来编译一下，

|  |  |
| --- | --- |
| 1 | [root@web nginx-1.4.2]# ./configure   --prefix=/usr   --sbin-path=/usr/sbin/nginx   --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/nginx.pid    --lock-path=/var/lock/nginx.lock   --user=nginx   --group=nginx   --with-http\_ssl\_module   --with-http\_flv\_module   --with-http\_stub\_status\_module   --with-http\_gzip\_static\_module   --http-client-body-temp-path=/var/tmp/nginx/client/   --http-proxy-temp-path=/var/tmp/nginx/proxy/   --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/   --http-uwsgi-temp-path=/var/tmp/nginx/uwsgi   --http-scgi-temp-path=/var/tmp/nginx/scgi   --with-pcre |

出现在下面的选项说明编译成功，

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | Configuration summary    + using system PCRE library    + using system OpenSSL library    + md5: using OpenSSL library    + sha1: using OpenSSL library    + using system zlib library    nginx path prefix: "/usr"    nginx binary file: "/usr/sbin/nginx"    nginx configuration prefix: "/etc/nginx"    nginx configuration file: "/etc/nginx/nginx.conf"    nginx pid file: "/var/run/nginx/nginx.pid"    nginx error log file: "/var/log/nginx/error.log"    nginx http access log file: "/var/log/nginx/access.log"    nginx http client request body temporary files: "/var/tmp/nginx/client/"    nginx http proxy temporary files: "/var/tmp/nginx/proxy/"    nginx http fastcgi temporary files: "/var/tmp/nginx/fcgi/"    nginx http uwsgi temporary files: "/var/tmp/nginx/uwsgi"    nginx http scgi temporary files: "/var/tmp/nginx/scgi"  [root@web nginx-1.4.2]# make && make install |

8.为nginx提供SysV init脚本

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110 | [root@web nginx-1.4.2]# vim  /etc/init.d/nginx  #!/bin/sh  #  # nginx - this script starts and stops the nginx daemon  #  # chkconfig:   - 85 15  # description:  Nginx is an HTTP(S) server, HTTP(S) reverse \  #               proxy and IMAP/POP3 proxy server  # processname: nginx  # config:      /etc/nginx/nginx.conf  # config:      /etc/sysconfig/nginx  # pidfile:     /var/run/nginx.pid  # Source function library.  . /etc/rc.d/init.d/functions  # Source networking configuration.  . /etc/sysconfig/network  # Check that networking is up.  [ "$NETWORKING" = "no" ] && exit 0  nginx="/usr/sbin/nginx"  prog=$(basename $nginx)  NGINX\_CONF\_FILE="/etc/nginx/nginx.conf"  [ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx  lockfile=/var/lock/subsys/nginx  make\_dirs() {     # make required directories     user=`nginx -V 2>&1 | grep "configure arguments:" | sed 's/[^\*]\*--user=\([^ ]\*\).\*/\1/g' -`     options=`$nginx -V 2>&1 | grep 'configure arguments:'`     for opt in $options; do         if [ `echo $opt | grep '.\*-temp-path'` ]; then             value=`echo $opt | cut -d "=" -f 2`             if [ ! -d "$value" ]; then                 # echo "creating" $value                 mkdir -p $value && chown -R $user $value             fi         fi     done  }  start() {      [ -x $nginx ] || exit 5      [ -f $NGINX\_CONF\_FILE ] || exit 6      make\_dirs      echo -n $"Starting $prog: "      daemon $nginx -c $NGINX\_CONF\_FILE      retval=$?      echo      [ $retval -eq 0 ] && touch $lockfile      return $retval  }  stop() {      echo -n $"Stopping $prog: "      killproc $prog -QUIT      retval=$?      echo      [ $retval -eq 0 ] && rm -f $lockfile      return $retval  }  restart() {      configtest || return $?      stop      sleep 1      start  }  reload() {      configtest || return $?      echo -n $"Reloading $prog: "      killproc $nginx -HUP      RETVAL=$?      echo  }  force\_reload() {      restart  }  configtest() {    $nginx -t -c $NGINX\_CONF\_FILE  }  rh\_status() {      status $prog  }  rh\_status\_q() {      rh\_status >/dev/null 2>&1  }  case "$1" in      start)          rh\_status\_q && exit 0          $1          ;;      stop)          rh\_status\_q || exit 0          $1          ;;      restart|configtest)          $1          ;;      reload)          rh\_status\_q || exit 7          $1          ;;      force-reload)          force\_reload          ;;      status)          rh\_status          ;;      condrestart|try-restart)          rh\_status\_q || exit 0              ;;      \*)          echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload|configtest}"          exit 2  esac |

9.为此脚本赋予执行权限

|  |  |
| --- | --- |
| 1 | [root@web ~]# chmod +x /etc/init.d/nginx |

10.添加至服务管理列表，并让其开机自动启动

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@web ~]# chmod +x /etc/init.d/nginx  [root@web ~]# chkconfig --add nginx  [root@web ~]# chkconfig nginx on  [root@web ~]# chkconfig --list nginx  nginx              0:关闭    1:关闭    2:启用    3:启用    4:启用    5:启用    6:关闭 |

11.启动nginx

|  |  |
| --- | --- |
| 1  2 | [root@web ~]# service nginx start  正在启动 nginx：                                           [确定] |

12.查看一下端口号

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | [root@web ~]# netstat -ntulp  Active Internet connections (only servers)  Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name  tcp        0      0 0.0.0.0:80                  0.0.0.0:\*                   LISTEN      4801/nginx  tcp        0      0 0.0.0.0:22                  0.0.0.0:\*                   LISTEN      1033/sshd  tcp        0      0 127.0.0.1:25                0.0.0.0:\*                   LISTEN      1110/master  tcp        0      0 127.0.0.1:6010              0.0.0.0:\*                   LISTEN      1144/sshd  tcp        0      0 127.0.0.1:6011              0.0.0.0:\*                   LISTEN      1203/sshd  tcp        0      0 :::22                       :::\*                        LISTEN      1033/sshd  tcp        0      0 ::1:25                      :::\*                        LISTEN      1110/master  tcp        0      0 ::1:6010                    :::\*                        LISTEN      1144/sshd  tcp        0      0 ::1:6011                    :::\*                        LISTEN      1203/sshd |

13. 测试访问一下

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846963c7mk.png)

好了，到这里nginx的安装全部完成，下面我们来看一下nginx的配置文件。

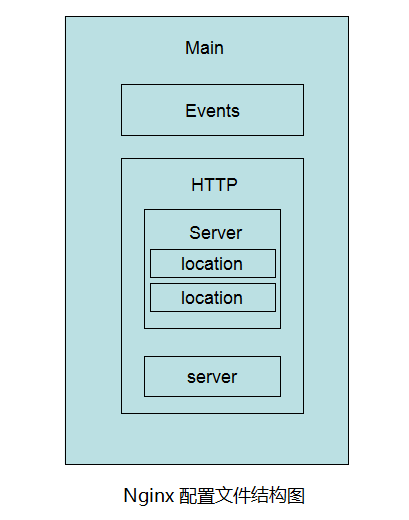
三、Nginx 配置文件说明

1.查看一下配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88 | [root@web ~]# cat /etc/nginx/nginx.conf  #user  nobody;  worker\_processes  1;  #error\_log  logs/error.log;  #error\_log  logs/error.log  notice;  #error\_log  logs/error.log  info;  #pid        logs/nginx.pid;  events {      worker\_connections  1024;  }  http {      include       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  logs/access.log  main;      sendfile        on;      #tcp\_nopush     on;      #keepalive\_timeout  0;      keepalive\_timeout  65;      #gzip  on;      server {          listen       80;          server\_name  localhost;          #charset koi8-r;          #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;          location = /50x.html {              root   html;          }          # proxy the PHP scripts to Apache listening on 127.0.0.1:80          #          #location ~ \.php$ {          #    proxy\_pass   http://127.0.0.1;          #}          # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000          #          #location ~ \.php$ {          #    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 {          #    deny  all;          #}      }      # another virtual host using mix of IP-, name-, and port-based configuration      #      #server {      #    listen       8000;      #    listen       somename:8080;      #    server\_name  somename  alias  another.alias;      #    location / {      #        root   html;      #        index  index.html index.htm;      #    }      #}      # HTTPS server      #      #server {      #    listen       443;      #    server\_name  localhost;      #    ssl                  on;      #    ssl\_certificate      cert.pem;      #    ssl\_certificate\_key  cert.key;      #    ssl\_session\_timeout  5m;      #    ssl\_protocols  SSLv2 SSLv3 TLSv1;      #    ssl\_ciphers  HIGH:!aNULL:!MD5;      #    ssl\_prefer\_server\_ciphers   on;      #    location / {      #        root   html;      #        index  index.html index.htm;      #    }      #}  } |

2.nginx 配置文件结构

Nginx配置文件主要分为4部分：main（全局设置）、server（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部分设置的指令将影响其他所有设置；server部分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部分用于匹配网页位置。这四者之间的关系如下：server继承main，location继承server，upstream既不会继承其他设置也不会被继承。如下图，

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846963yLhn.png)

在这4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、Http Gzip Static模块和Http Addition模块等。

下面通过一个Nginx配置实例，详细介绍nginx.conf每个指令的含义。为了能更清楚地了解Nginx的结构和每个配置选项的含义，这里按照功能点将Nginx配置文件分为7个部分依次讲解。下面就围绕这7个部分进行介绍。

3.配置文件详解

(1).Nginx 的全局配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #user  nobody;  worker\_processes  1;  #error\_log  logs/error.log;  #error\_log  logs/error.log  notice;  #error\_log  logs/error.log  info;  #pid        logs/nginx.pid;  worker\_rlimit\_nofile 65535;  events {      use epoll;      worker\_connections  1024;  } |

上面这段代码中每个配置选项的含义解释如下：

* user是个主模块指令，指定Nginx Worker进程运行用户以及用户组，默认由nobody账号运行。
* worker\_processes是个主模块指令，指定了Nginx要开启的进程数。每个Nginx进程平均耗费10MB～12MB内存。根据经验，一般指定一个进程足够了，如果是多核CPU，建议指定和CPU的数量一样多的进程数即可。（注，如果负载以CPU密集型应用为主，如SSL或压缩应用，则worker数应与CPU数相同；如果负载以IO密集型为主，如响应大量内容给客户端，则worker数应该为CPU个数的1.5或2倍。）
* error\_log是个主模块指令，用来定义全局错误日志文件。日志输出级别有debug、info、notice、warn、error、crit可供选择，其中，debug输出日志最为最详细，而crit输出日志最少。
* pid是个主模块指令，用来指定进程id的存储文件位置。
* worker\_rlimit\_nofile 用于绑定worker进程和CPU， Linux内核2.4以上可用。
* events指令用来设定Nginx的工作模式及连接数上限。
* use是个事件模块指令，用来指定Nginx的工作模式。Nginx支持的工作模式有select、poll、kqueue、epoll、rtsig和/dev/poll。其中select和poll都是标准的工作模式，kqueue和epoll是高效的工作模式，不同的是epoll用在Linux平台上，而kqueue用在BSD系统中。对于Linux系统，epoll工作模式是首选。
* worker\_connections也是个事件模块指令，用于定义Nginx每个进程的最大连接数，默认是1024。最大客户端连接数由worker\_processes和worker\_connections决定，即max\_client=worker\_processes\*worker\_connections，在作为反向代理时变为：max\_clients = worker\_processes \* worker\_connections/4。(注，进程的最大连接数受Linux系统进程的最大打开文件数限制，在执行操作系统命令“ulimit -n 65536”后worker\_connections的设置才能生效。)

(2).HTTP服务器配置

注，接下来开始对HTTP服务器进行配置。下面这段内容是Nginx对HTTP服务器相关属性的配置，代码如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | http {      include       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  logs/access.log  main;      sendfile        on;      #tcp\_nopush     on;      #keepalive\_timeout  0;      keepalive\_timeout  65;      #gzip  on; |

下面详细介绍这段代码中每个配置选项的含义。

* 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。
* log\_format是Nginx的HttpLog模块指令，用于指定Nginx日志的输出格式。main为此日志输出格式的名称，可以在下面的access\_log指令中引用。
* client\_max\_body\_size用来设置允许客户端请求的最大的单个文件字节数。
* client\_header\_buffer\_size用于指定来自客户端请求头的headerbuffer大小。对于大多数请求，1KB的缓冲区大小已经足够，如果自定义了消息头或有更大的cookie，可以增加缓冲区大小。这里设置为32KB。
* large\_client\_header\_buffers用来指定客户端请求中较大的消息头的缓存最大数量和大小， “4”为个数，“128K”为大小，最大缓存为4个128KB。
* sendfile参数用于开启高效文件传输模式。将tcp\_nopush和tcp\_nodely两个指令设置为on，用于防止网络阻塞。
* keepalive\_timeout用于设置客户端连接保持活动的超时时间。在超过这个时间之后，服务器会关闭该连接。
* client\_header\_timeout用于设置客户端请求头读取超时时间。如果超过这个时间，客户端还没有发送任何数据，Nginx将返回“Request time out（408）”错误。
* client\_body\_timeout用于设置客户端请求主体读取超时时间，默认值为60。如果超过这个时间，客户端还没有发送任何数据，Nginx将返回“Request time out（408）”错误。
* send\_timeout用于指定响应客户端的超时时间。这个超时仅限于两个连接活动之间的时间，如果超过这个时间，客户端没有任何活动，Nginx将会关闭连接。
* gzip用于设置开启或者关闭gzip模块，“gzip on”表示开启gzip压缩，实时压缩输出数据流。

(3).HttpGzip模块配置

下面配置Nginx的HttpGzip模块。这个模块支持在线实时压缩输出数据流。要查看是否安装了此模块，需要使用下面的命令：

|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@web nginx-1.4.2]# nginx -V  nginx version: nginx/1.4.2  built by gcc 4.4.7 20120313 (Red Hat 4.4.7-3) (GCC)  TLS SNI support enabled  configure arguments: --prefix=/usr --sbin-path=/usr/sbin/nginx --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/nginx.pid --lock-path=/var/lock/nginx.lock --user=nginx --group=nginx --with-http\_ssl\_module --with-http\_flv\_module --with-http\_stub\_status\_module --with-http\_gzip\_static\_module --http-client-body-temp-path=/var/tmp/nginx/client/ --http-proxy-temp-path=/var/tmp/nginx/proxy/ --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/ --http-uwsgi-temp-path=/var/tmp/nginx/uwsgi --http-scgi-temp-path=/var/tmp/nginx/scgi --with-pcre |

通过nginx -V 命令可以查看安装Nginx时的编译选项。由输出可知，我们已经安装了HttpGzip模块。下面是HttpGzip模块在Nginx配置中的相关属性设置：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | #gzip  on;  #gzip\_min\_length  1k;  #gzip\_buffers     4  16k;  #gzip\_http\_version  1.1;  #gzip\_comp\_level  2;  #gzip\_types  text/plain application/x-javascript text/css application/xml;  #gzip\_vary  on; |

* gzip用于设置开启或者关闭gzip模块，“gzip on”表示开启gzip压缩，实时压缩输出数据流。
* gzip\_min\_length用于设置允许压缩的页面最小字节数，页面字节数从header头的Content-Length中获取。默认值是0，不管页面多大都进行压缩。建议设置成大于1K的字节数，小于1K可能会越压越大。
* gzip\_buffers表示申请4个单位为16K的内存作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果。
* gzip\_buffers表示申请4个单位为16K的内存作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果。
* gzip\_comp\_level用来指定gzip压缩比，1 压缩比最小，处理速度最快；9 压缩比最大，传输速度快，但处理最慢，也比较消耗CPU资源。
* gzip\_types用来指定压缩的类型，无论是否指定，“text/html”类型总是会被压缩的。
* gzip\_vary选项可以让前端的缓存服务器缓存经过gzip压缩的页面，例如，用Squid缓存经过Nginx压缩的数据。

(4).负载均衡配置

下面设定负载均衡的服务器列表。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | upstream test.net{  ip\_hash;  server 192.168.10.13:80;  server 192.168.10.14:80  down;  server 192.168.10.15:8009  max\_fails=3  fail\_timeout=20s;  server 192.168.10.16:8080;  } |

* upstream是Nginx的HTTP Upstream模块，这个模块通过一个简单的调度算法来实现客户端IP到后端服务器的负载均衡。在上面的设定中，通过upstream指令指定了一个负载均衡器的名称test.net。这个名称可以任意指定，在后面需要用到的地方直接调用即可。
* Nginx的负载均衡模块目前支持4种调度算法，下面进行分别介绍，其中后两项属于第三方调度算法。
* 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，如果后端某台服务器宕机，故障系统被自动剔除，使用户访问不受影响。
* Weight。指定轮询权值，Weight值越大，分配到的访问机率越高，主要用于后端每个服务器性能不均的情况下。
* ip\_hash。每个请求按访问IP的hash结果分配，这样来自同一个IP的访客固定访问一个后端服务器，有效解决了动态网页存在的session共享问题。
* fair。这是比上面两个更加智能的负载均衡算法。此种算法可以依据页面大小和加载时间长短智能地进行负载均衡，也就是根据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。Nginx本身是不支持fair的，如果需要使用这种调度算法，必须下载Nginx的upstream\_fair模块。
* url\_hash。此方法按访问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，可以进一步提高后端缓存服务器的效率。Nginx本身是不支持url\_hash的，如果需要使用这种调度算法，必须安装Nginx 的hash软件包。
* 在HTTP Upstream模块中，可以通过server指令指定后端服务器的IP地址和端口，同时还可以设定每个后端服务器在负载均衡调度中的状态。常用的状态有：
* down，表示当前的server暂时不参与负载均衡。
* backup，预留的备份机器。当其他所有的非backup机器出现故障或者忙的时候，才会请求backup机器，因此这台机器的压力最轻。
* 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，在经历了max\_fails次失败后，暂停服务的时间。max\_fails可以和fail\_timeout一起使用。

注意，当负载调度算法为ip\_hash时，后端服务器在负载均衡调度中的状态不能是weight和backup。

(5).server虚拟主机配置

下面介绍对虚拟主机的配置。建议将对虚拟主机进行配置的内容写进另外一个文件，然后通过include指令包含进来，这样更便于维护和管理。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | server{  listen         80;  server\_name    192.168.12.188  www.test.net;  index index.html index.htm index.php;  root  /web/www/www.test.net  charset gb2312;  access\_log  logs/www.test.net.access.log  main; |

* server标志定义虚拟主机开始；
* listen用于指定虚拟主机的服务器端口；
* server\_name用来指定IP地址或者域名，多个域名之间用空格分开；
* index用于设定访问的默认首页地址；
* root指令用于指定虚拟主机的网页根目录，这个目录可以是相对路径，也可以是绝对路径；
* charset用于设置网页的默认编码格式。
* access\_log用来指定此虚拟主机的访问日志存放路径。最后的main用于指定访问日志的输出格式。

(6).URL匹配配置

URL地址匹配是Nginx配置中最灵活的部分。 location支持正则表达式匹配，也支持条件判断匹配，用户可以通过location指令实现Nginx对动、静态网页的过滤处理。

格式：location [ = | ~ | ~\* | ^~ ] uri { ... }

* location URI {}：对当前路径及子路径下的所有对象都生效；
* location = URI {}：精确匹配指定的路径，不包括子路径，因此，只对当前资源生效；
* location ~ URI {}，location ~\* URI {}：模式匹配URI，此处的URI可使用正则表达式，~区分字符大小写，~\*不区分字符大小写；
* location ^~ URI {}：不使用正则表达式

案例1：

|  |  |
| --- | --- |
| 1  2  3  4 | location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf)$  {                      root    /web/www/www.test.net;                      expires 30d;         } |

说明：上面这段设置是通过location指令来对网页URL进行分析处理，所有扩展名为.gif、.jpg、.jpeg、.png、.bmp、.swf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。

案例2：

|  |  |
| --- | --- |
| 1  2  3  4 | location ~ ^/(upload|html)/  {                  root    /web/www/www.test.net;                  expires 30d;  } |

说明：上面这段设置是将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/web/www/www.test.net目录中。

案例3：

|  |  |
| --- | --- |
| 1  2  3  4 | location ~ .\*.jsp$ {              index index.jsp;              proxy\_pass http://localhost:8080;            } |

说明：在最后这段设置中，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.jsp为后缀的文件都交给本机的8080端口处理。

location [ = | ~ | ~\* | ^~ ] 优先级

* location = URI {}：精确匹配指定的路径，不包括子路径，因此，只对当前资源生效；(优先级最高）
* location ^~ URI {}：不使用正则表达式；（优先级次之）
* location ~ URI {}，location ~\* URI {}：模式匹配URI，此处的URI可使用正则表达式，~区分字符大小写，~\*不区分字符大小写；（优先级次之）
* location URI {}：对当前路径及子路径下的所有对象都生效；（优先级最低）

案例：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | location = / {      [ configuration A ]  }  location / {      [ configuration B ]  }  location /documents/ {      [ configuration C ]  }  location ^~ /images/  {      [ configuration D ]  }  location ~\* \.(gif|jpg|jpeg)$ {      [ configuration E ]  } |

说明：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.

(7).StubStatus模块配置

StubStatus模块能够获取Nginx自上次启动以来的工作状态，此模块非核心模块，需要在Nginx编译安装时手工指定才能使用。以下指令指定启用获取Nginx工作状态的功能。

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | location /NginxStatus {      stub\_status     on;      access\_log             logs/NginxStatus.log;      auth\_basic             "NginxStatus";      auth\_basic\_user\_file    ../htpasswd;         } |

* stub\_status为“on”表示启用StubStatus的工作状态统计功能；
* access\_log 用来指定StubStatus模块的访问日志文件；
* auth\_basic是Nginx的一种认证机制；
* auth\_basic\_user\_file用来指定认证的密码文件。

由于Nginx的auth\_basic认证采用的是与Apache兼容的密码文件，因此需要用Apache的htpasswd命令来生成密码文件。例如要添加一个webadmin用户，可以使用下面的方式生成密码文件：

|  |  |
| --- | --- |
| 1 | /usr/local/apache/bin/htpasswd -c  /opt/nginx/conf/htpasswd webadmin |

要查看Nginx的运行状态，可以输入<http://ip/> NginxStatus，然后输入刚刚创建的用户名和密码就可以看到如下信息：

|  |  |
| --- | --- |
| 1  2  3  4 | Active connections: 1  server accepts handled requests  393411 393411 393799  Reading: 0 Writing: 1 Waiting: 0 |

* Active connections表示当前活跃的连接数。
* 第三行的3个数字表示 Nginx当前总共处理了393411个连接， 成功创建了393 411次握手，总共处理了393 799个请求。
* 最后一行的Reading表示Nginx读取到客户端Header信息数； Writing表示Nginx返回给客户端的Header信息数；Waiting表示Nginx已经处理完、正在等候下一次请求指令时的驻留连接数。

补充说明：

|  |  |
| --- | --- |
| 1  2  3  4  5 | error\_page  404              /404.html;  error\_page   500 502 503 504  /50x.html;  location = /50x.html {             root   html;         } |

在最后这段设置中，设置了虚拟主机的错误信息返回页面，通过error\_page指令可以定制各种错误信息的返回页面。在默认情况下，Nginx会在主目录的html目录中查找指定的返回页面。特别需要注意的是，这些错误信息的返回页面大小一定要超过512KB，否则会被IE浏览器替换为IE默认的错误页面。好了，到这里nginx的配置文件讲解全部完成。下面我们来说一说nginx命令参数。

四、Nginx 命令参数

不像许多其他软件系统，Nginx 仅有数个命令行参数，完全通过配置文件来配置（想象一下）。

* [#options 选项]
* [#example 示例]
* [#lncus 使用信号加载新的配置]
* [#utnbotf 平滑升级到新的二进制代码]

**选项**

* **-c </path/to/config>**为 Nginx 指定一个配置文件，来代替缺省的。
* **-t** 不运行，而仅仅测试配置文件。nginx 将检查配置文件的语法的正确性，并尝试打开配置文件中所引用到的文件。
* **-v** 显示 nginx 的版本。
* **-V** 显示 nginx 的版本，编译器版本和配置参数。

更多

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [root@web ~]# nginx -h  nginx version: nginx/1.4.2  Usage: nginx [-?hvVtq] [-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    -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/)    -c filename   : set configuration file (default: /etc/nginx/nginx.conf)    -g directives : set global directives out of configuration file |

五、配置Nginx提供Web服务

1.提供测页面

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [root@web nginx]# mkdir -pv /data/www  mkdir: 已创建目录 "/data/www"  [root@web nginx]# cd /data/www  [root@web www]# ll  总用量 0  [root@web www]# cat index.html  <h1>www.nginx.org</h1>  [root@web www]# chown -R nginx.nginx /data/www/\* |

2.备份配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [root@web ~]# cd /etc/nginx/  [root@web nginx]# cp nginx.conf nginx.conf.bak  [root@web nginx]# ls  fastcgi.conf          fastcgi\_params.default  mime.types          nginx.conf.bak      scgi\_params.default   win-utf  fastcgi.conf.default  koi-utf                 mime.types.default  nginx.conf.default  uwsgi\_params  fastcgi\_params        koi-win                 nginx.conf          scgi\_params         uwsgi\_params.default |

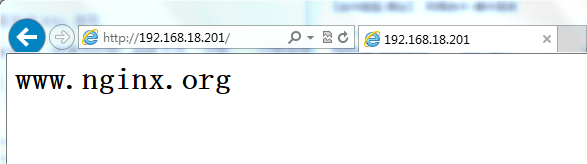
3.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@web www]# vim /etc/nginx/nginx.conf  server {        listen       80;        server\_name  localhost;        #charset koi8-r;        #access\_log  logs/host.access.log  main;        location / {            root   /data/www;            index  index.html index.htm;        } |

4.重新加载nginx配置

|  |  |
| --- | --- |
| 1  2  3  4 | [root@web run]# service nginx reload  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  重新载入 nginx：                                           [确定] |

5.测试一下

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846964h5Fv.png)

好了，一个基本的web服务器已配置完成，简单吧。

六、配置Nginx的虚拟主机

1.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | [root@web www]# vim /etc/nginx/nginx.conf  server {          listen       80;          server\_name  www.nginx.com;          location / {              root   /data/www;              index  index.html index.htm;          }      }      server {          listen       80;          server\_name  www.test.com;          location / {              root   /data/test;              index  index.html index.htm;          }      } |

2.提供测试页面

|  |  |
| --- | --- |
| 1  2  3  4 | [root@web data]# mkdir test  [root@web data]# cd test/  [root@web test]# cat index.html  <h1>www.test.com</h1> |

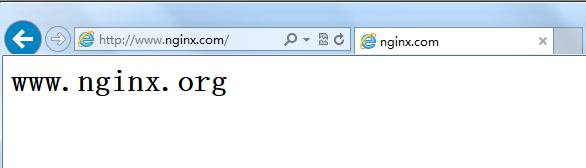
3.重新加载nginx配置

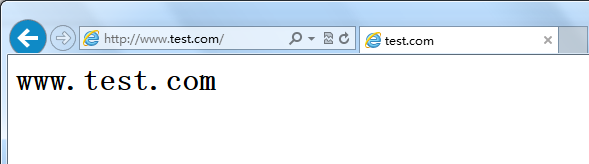
|  |  |
| --- | --- |
| 1  2  3  4 | [root@web run]# service nginx reload  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  重新载入 nginx：                                           [确定] |

4.修改测试机的hosts文件

|  |  |
| --- | --- |
| 1  2  3  4 | Windows 7下路径：C:\Windows\System32\drivers\etc\hosts  增加两行：  192.168.18.201    www.nginx.com  192.168.18.201    www.test.com |

5.测试一下

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846964jqsG.png)

[](http://img1.51cto.com/attachment/201308/30/2033581_13778469658Ukw.png)

好了，到这里基于域名的虚拟主机配置完成。

七、配置Nginx的访问控制

基于用户的访问控制，

1.提供测试文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [root@web run]# cd /data/www/  [root@web www]# ll  总用量 4  -rw-r--r-- 1 nginx nginx 23 8月  29 20:04 index.html  [root@web www]# mkdir bbs  [root@web www]# cd bbs/  [root@web bbs]# vim index.html  [root@web bbs]# cat index.html  <h1>Auth Page</h1> |

2.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | location /data {             root /www/bbs;             index index.html             auth\_basic             "Auth Page";             auth\_basic\_user\_file  /etc/nginx/.user;         } |

3.安装httpd

|  |  |
| --- | --- |
| 1 | [root@web bbs]# yum install -y httpd |

4.生成认证文件

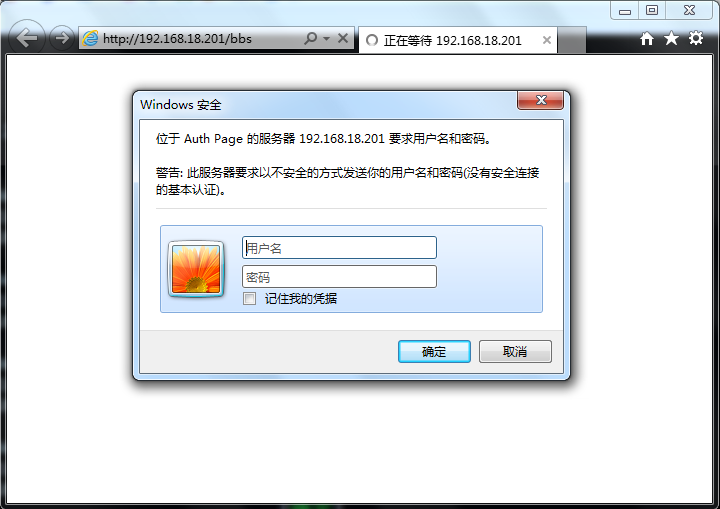
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | [root@web bbs]# htpasswd -c -m /etc/nginx/.user nginx  New password:  Re-type new password:  Adding password for user nginx  [root@web bbs]# ls -a /etc/nginx/  .                     fastcgi\_params          mime.types          nginx.conf.default   .user  ..                    fastcgi\_params.default  mime.types.default  .nginx.conf.swp      uwsgi\_params  fastcgi.conf          koi-utf                 nginx.conf          scgi\_params          uwsgi\_params.default  fastcgi.conf.default  koi-win                 nginx.conf.bak      scgi\_params.default  win-utf |

5.重新加载一下nginx配置文件

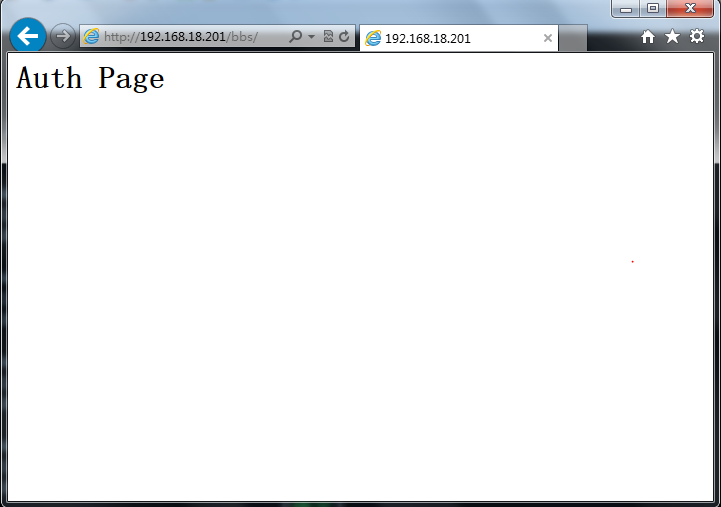
|  |  |
| --- | --- |
| 1  2  3  4 | [root@web bbs]# service nginx reload  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  重新载入 nginx：                                           [确定] |

6.测试一下

(1).输入用户名nginx，密码123456

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846965Tp1D.png)

(2).显示页面

[](http://img1.51cto.com/attachment/201308/30/2033581_13778469667neQ.png)

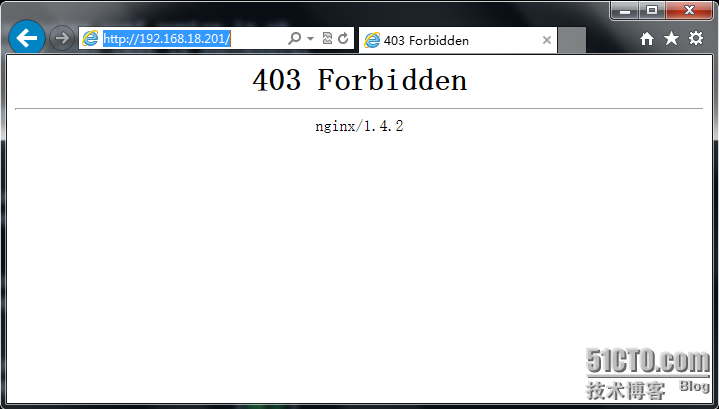
基于IP的访问控制，

1.控制指令

* allow 定义允许访问的规则
* deny 定义拒绝访问的规则
* allow all或deny all 定义默认规则

2.案例

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@web test]# vim /etc/nginx/nginx.conf  location / {           root   /data/www;           index  index.html index.htm;           #auth\_basic "Auth Page";           #auth\_basic\_user\_file /etc/nginx/.user;           deny 192.168.18.138;           allow 192.168.18.0/24;           deny  all;       } |

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846967BypM.png)

注，大家可以看到不允许访问。allow与deny指令使用很简单，唯一与httpd不同的是nginx没有定义默认规则，所以默认规则得自己定义。我这里定义是dell all；默认拒绝所有。

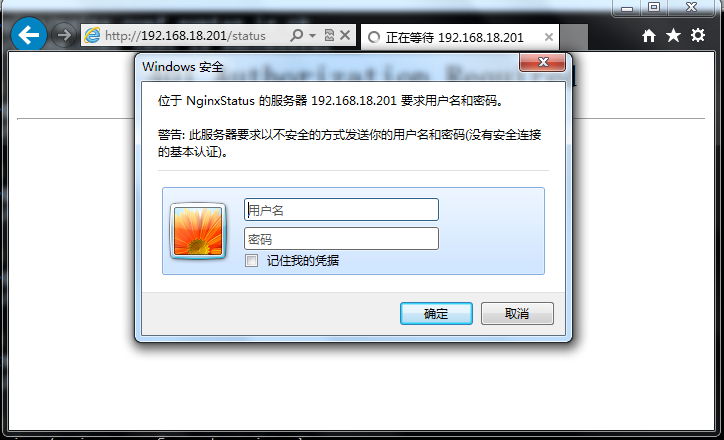
八、配置Nginx提供状态页面

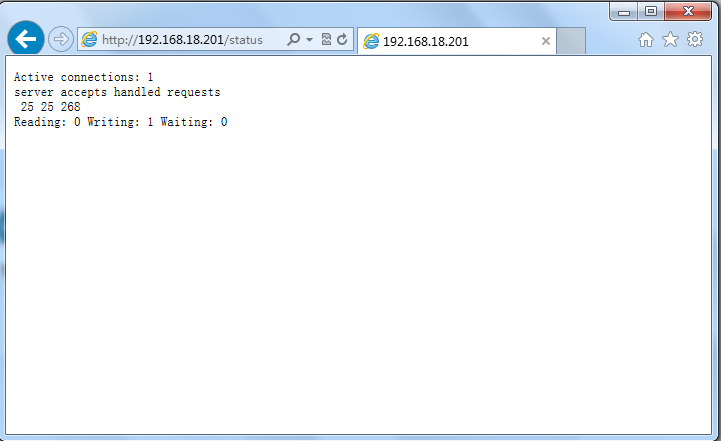
1.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@web test]# vim /etc/nginx/nginx.conf  location /status {       root /;       stub\_status on;       auth\_basic "NginxStatus";       auth\_basic\_user\_file /etc/nginx/.user;       } |

2.重新加载一下配置文件

3.测试

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846968pnM6.png)

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846969vYY4.png)

九、配置Nginx的错误页面

1.提供404错误页面

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [root@web www]# ll  总用量 8  drwxr-xr-x 2 root  root  4096 8月  29 20:36 bbs  -rw-r--r-- 1 nginx nginx   23 8月  29 20:04 index.html  [root@web www]# vim 404.html  [root@web www]# cat 404.html  <h1>404 error</h1>  <h1>404 error</h1>  <h1>404 error</h1>  <h1>404 error</h1>  …… |

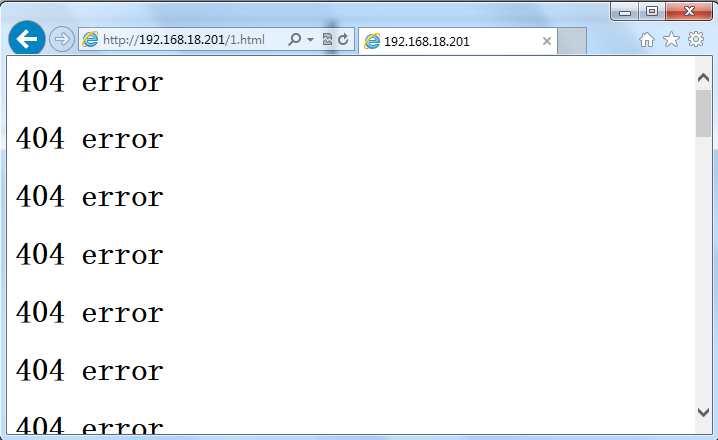
2.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4 | [root@web test]# vim /etc/nginx/nginx.conf  server {  error\_page  404                 /404.html;  } |

3.重新加载一下nginx配置文件

|  |  |
| --- | --- |
| 1  2  3  4 | [root@web www]# service nginx reload  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  重新载入 nginx：                                           [确定] |

4.我们访问一下不存在的页面

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846969niLC.png)

十、配置Nginx打开目录浏览功能

1.指令说明

Nginx默认是不允许列出整个目录的。如需此功能，打开nginx.conf文件，在location server 或 http段中加入autoindex on；另外两个参数最好也加上去，

* autoindex\_exact\_size off；默认为on，显示出文件的确切大小，单位是bytes。改为off后，显示出文件的大概大小，单位是kB或者MB或者GB。
* autoindex\_localtime on；默认为off，显示的文件时间为GMT时间。改为on后，显示的文件时间为文件的服务器时间。

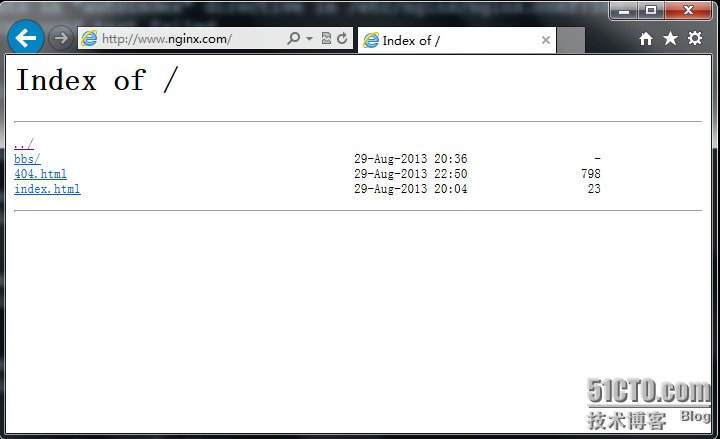
2.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | server {      listen       80;      server\_name  www.nginx.com;      location / {      autoindex on;      autoindex\_exact\_size on;      autoindex\_localtime on;      root   /data/www;      index  123.html;      }  } |

3.重新加载配置文件

|  |  |
| --- | --- |
| 1  2  3  4 | [root@web www]# service nginx reload  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  重新载入 nginx：                                           [确定] |

4.测试一下

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846970LXDI.png)

十一、配置Nginx基于ssl提供https服务

1.创建CA自签证书

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | [root@web ~]# cd /etc/pki/CA/  [root@web CA]# ls  certs  crl  newcerts  private  [root@web CA]# cd private/  [root@web private]# ls  [root@web private]# (umask 077; openssl genrsa 2048 > cakey.pem) #生成私钥  Generating RSA private key, 2048 bit long modulus  ...............................+++  .............+++  e is 65537 (0x10001)  [root@web CA]# openssl req -new -x509 -key ./private/cakey.pem -out cacert.pem #生成自签证书  You are about to be asked to enter information that will be incorporated  into your certificate request.  What you are about to enter is what is called a Distinguished Name or a DN.  There are quite a few fields but you can leave some blank  For some fields there will be a default value,  If you enter '.', the field will be left blank.  -----  Country Name (2 letter code) [XX]:CN  State or Province Name (full name) []:SH  Locality Name (eg, city) [Default City]:XH  Organization Name (eg, company) [Default Company Ltd]:JJHH  Organizational Unit Name (eg, section) []:Tech  Common Name (eg, your name or your server's hostname) []:ca.test.com  Email Address []:caadmin@test.com  [root@web private]# ll  总用量 8  -rw------- 1 root root 1679 8月  29 23:31 cakey.pem  [root@web CA]# touch serial  [root@web CA]# echo 01 > serial  [root@web CA]# touch index.txt  [root@web CA]# ll  总用量 24  -rw-r--r--  1 root root 1375 8月  29 23:34 cacert.pem  drwxr-xr-x. 2 root root 4096 3月   5 06:22 certs  drwxr-xr-x. 2 root root 4096 3月   5 06:22 crl  -rw-r--r--  1 root root    0 8月  29 23:35 index.txt  drwxr-xr-x. 2 root root 4096 3月   5 06:22 newcerts  drwx------. 2 root root 4096 8月  29 23:49 private  -rw-r--r--  1 root root    3 8月  29 23:35 serial |

2.生成证书申请

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | [root@web ~]# mkdir /etc/nginx/ssl  [root@web CA]# cd /etc/nginx/ssl/  [root@web ssl]# (umask 077; openssl genrsa 1024 > nginx.key) #生成私钥  Generating RSA private key, 1024 bit long modulus  .........................................++++++  ..................................++++++  e is 65537 (0x10001)  [root@web ssl]# openssl req -new -key nginx.key -out nginx.csr  You are about to be asked to enter information that will be incorporated  into your certificate request.  What you are about to enter is what is called a Distinguished Name or a DN.  There are quite a few fields but you can leave some blank  For some fields there will be a default value,  If you enter '.', the field will be left blank.  -----  Country Name (2 letter code) [XX]:CN  State or Province Name (full name) []:SH  Locality Name (eg, city) [Default City]:XH  Organization Name (eg, company) [Default Company Ltd]:JJHH  Organizational Unit Name (eg, section) []:Tech  Common Name (eg, your name or your server's hostname) []:www.test.com  Email Address []:  Please enter the following 'extra' attributes  to be sent with your certificate request  A challenge password []:  An optional company name []: |

3. 让CA签名并颁发证书

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | [root@web ssl]# openssl ca -in nginx.csr -out nginx.crt -days 3650  Using configuration from /etc/pki/tls/openssl.cnf  Check that the request matches the signature  Signature ok  Certificate Details:          Serial Number: 1 (0x1)          Validity              Not Before: Aug 29 15:51:53 2013 GMT              Not After : Aug 27 15:51:53 2023 GMT          Subject:              countryName               = CN              stateOrProvinceName       = SH              organizationName          = JJHH              organizationalUnitName    = Tech              commonName                = www.test.com          X509v3 extensions:              X509v3 Basic Constraints:                  CA:FALSE              Netscape Comment:                  OpenSSL Generated Certificate              X509v3 Subject Key Identifier:                  60:87:97:14:D5:A2:23:B9:C5:13:97:5D:0D:B9:D7:C3:C2:66:F0:4B              X509v3 Authority Key Identifier:                  keyid:9E:3E:5B:84:06:BE:68:01:C9:16:7C:08:5F:C5:54:0D:7B:FC:FA:87  Certificate is to be certified until Aug 27 15:51:53 2023 GMT (3650 days)  Sign the certificate? [y/n]:y  1 out of 1 certificate requests certified, commit? [y/n]y  Write out database with 1 new entries  Data Base Updated |

4.修改配置文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | server {        listen       443;        server\_name  localhost;        ssl                  on;        ssl\_certificate      /etc/nginx/ssl/nginx.crt;        ssl\_certificate\_key  /etc/nginx/ssl/nginx.key;        ssl\_session\_timeout  5m;        ssl\_protocols  SSLv2 SSLv3 TLSv1;        ssl\_ciphers  HIGH:!aNULL:!MD5;        ssl\_prefer\_server\_ciphers   on;        location / {            root   html;            index  index.html index.htm;        }    } |

5.重新启动一下nginx服务器

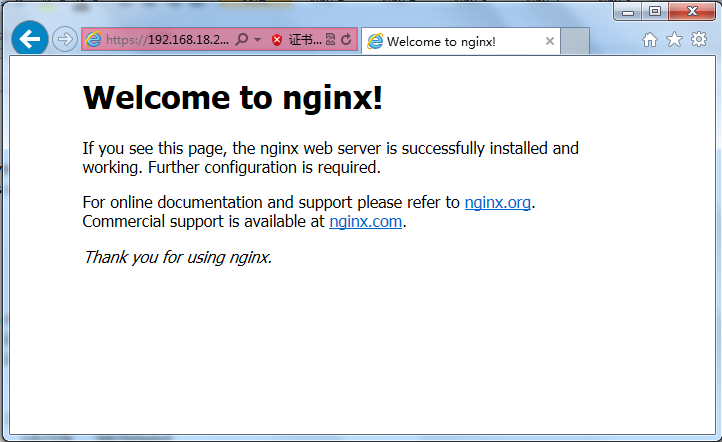
|  |  |
| --- | --- |
| 1  2  3  4  5 | [root@web ssl]# service nginx restart  nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  nginx: configuration file /etc/nginx/nginx.conf test is successful  停止 nginx：                                               [确定]  正在启动 nginx：                                           [确定] |

6.查看一下端口

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [root@web ssl]# netstat -ntlp  Active Internet connections (only servers)  Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name  tcp        0      0 0.0.0.0:80                  0.0.0.0:\*                   LISTEN      10661/nginx  tcp        0      0 0.0.0.0:22                  0.0.0.0:\*                   LISTEN      1033/sshd  tcp        0      0 127.0.0.1:25                0.0.0.0:\*                   LISTEN      1110/master  tcp        0      0 127.0.0.1:6010              0.0.0.0:\*                   LISTEN      9599/sshd  tcp        0      0 0.0.0.0:443                 0.0.0.0:\*                   LISTEN      10661/nginx  tcp        0      0 127.0.0.1:6012              0.0.0.0:\*                   LISTEN      9470/sshd  tcp        0      0 :::22                       :::\*                        LISTEN      1033/sshd  tcp        0      0 ::1:25                      :::\*                        LISTEN      1110/master  tcp        0      0 ::1:6010                    :::\*                        LISTEN      9599/sshd  tcp        0      0 ::1:6012                    :::\*                        LISTEN      9470/sshd |

7.测试一下

[](http://img1.51cto.com/attachment/201308/30/2033581_13778469719KxP.png)

[](http://img1.51cto.com/attachment/201308/30/2033581_1377846972T0TU.png)