本篇文章来探讨下nginx核心配置文件的常用配置项意义解读。先按照如下图的最小化配置来说明其含义，日后再慢慢拓展其他的配置含义。希望通过这篇文章迅速摸清楚nginx核心配置文件里面都配置了啥，方便后续的学习。

```
#user  nobody;
worker_processes  auto;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  /var/log/nginx/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  oursnail.cn;

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

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

一、nobody用户

```
#user  nobody;
worker_processes  auto;
```

`nginx.conf` 的前两行是如上这个配置。第二个之前已经解释过了，决定了工作进程的数量，这里用 `auto` 表示按照实际的CPU核心数去自动生成对应的工作进程的数量；第一个指定了工作进程的用户是 `nobody` 用户。

我们可以查看下进程情况：

```
[root@VM-0-13-centos conf]# ps -ef | grep nginx
root     12124     1  0  2020 ?        00:00:00 nginx: master
process ./nginx
nobody   25677 12124  0  2020 ?        00:00:04 nginx: worker
process
```

这个用户显然是启动 `nginx` 后，系统自动赋予的用户，这个 `nobody` 是个啥呢?

> 1、Windows系统在安装后会自动建立一些用户帐户，在Linux系统中同样有一些用户帐户是在系统安装后就有的，就像Windows系统中的内置帐户一样。
> 2、它们是用来完成特定任务的，比如nobody和ftp等，我们访问LinuxSir.Org的网页程序时，官网的服务器就是让客户以'nobody'身份登录的(相当于Windows系统中的匿名帐户);我们匿名访问ftp时，会用到用户ftp或nobody。
> 3、nobody就是一个普通账户，因为默认登录shell是'/sbin/nologin'，所以这个用户是无法直接登录系统的，也就是黑客很难通过漏洞连接到你的服务器来做破坏。此外这个用户的权限也给配置的很低。因此有比较高的安全性。一切都只给最低权限。这就是nobody存在的意义。

当然了，这里可以设置 `user` 为 `root` 用户。

二、错误日志

下面的三行紧接着就是:

```
#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info

#pid logs/nginx.pid
```

显然，这里设置的是错误日志路径，不过为什么有三行呢? 下面的 `notice` 和 `info` 是什么意思呢? 这里显然是日志级别， `nginx` 的日志级别为:

> debug | info | notice | warn | error | crit | alert | emerg

错误级别从左到右越来越大。级别越高记录的信息越少，如果不定义，默认级别为error。还记得我们安装nginx的时候，指定的error日志默认位置是 `/var/log/nginx/error.log` 。

假设我们错误地访问不存在的路径: `http://111.231.119.253/hello` ，实时打印 `error.log` 日志，可以看到会有错误日志打印输出:

下面一行是pid，即进程号，这个也是我们在安装的时候指定了位置：

```
--pid-path=/var/run/nginx/nginx.pid
```

编译安装之后就会自动生成此文件，我们看下这个文件：

```
[root@VM-0-13-centos sbin]# cat /var/run/nginx/nginx.pid
12124
[root@VM-0-13-centos sbin]# ps -ef | grep nginx
nobody    4087 12124  0 23:19 ?        00:00:00 nginx: worker
process
root      4551 25026  0 23:22 pts/0    00:00:00 grep --color=auto
nginx
root     12124     1  0  2020 ?        00:00:00 nginx: master
process ./nginx
```

可以看到，记录的就是 `master` 进程的进程号。当我们关闭 `nginx` 并重启后，进程号会发生变化，这个文件里面记录的进程号也会根据实际的情况改变。

```
[root@VM-0-13-centos sbin]# ./nginx -s stop
[root@VM-0-13-centos sbin]#
[root@VM-0-13-centos sbin]# ps -ef | grep nginx
root      5119 25026  0 23:25 pts/0    00:00:00 grep --color=auto
nginx
[root@VM-0-13-centos sbin]# ./nginx
[root@VM-0-13-centos sbin]# ps -ef | grep nginx
root      5186     1  0 23:25 ?        00:00:00 nginx: master
process ./nginx
nobody    5187  5186  0 23:25 ?        00:00:00 nginx: worker
process
root      5196 25026  0 23:25 pts/0    00:00:00 grep --color=auto
nginx
[root@VM-0-13-centos sbin]# cat /var/run/nginx/nginx.pid
5186
```

三、events

下一个就是来到了：

```
events {
    worker_connections  1024;
}
```

实际上可以改写为：

```
events {
    #epoll是多路复用IO(I/O Multiplexing)中的一种方式,
    #仅用于linux2.6以上内核,可以大大提高nginx的性能
    use epoll;
    # 每个worker允许连接的客户端最大连接数
    worker_connections  1024;
}
```

在前面的学习中我们对 `epoll` 有了一定的认识，这里就不再赘述了。

四、http模块

下面就是最重要的一大块了。

```
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  oursnail.cn;

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

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
}
```

4.1 mime.types

其中第一项是：

```
include       mime.types;
```

显然是一个导入，那么导入的这个 `mime.types` 是啥呢？在 `nginx.conf` 的同级目录下就有这个文件。

```
types {
    text/html                                        html htm shtml;
    text/css                                         css;
    text/xml                                         xml;
    image/gif                                        gif;
    image/jpeg                                       jpeg jpg;
    application/javascript                           js;
    application/atom+xml                             atom;
    application/rss+xml                              rss;

    text/mathml                                      mml;
    text/plain                                       txt;
    text/vnd.sun.j2me.app-descriptor                 jad;
    text/vnd.wap.wml                                 wml;
    text/x-component                                 htc;

    image/png                                        png;
    image/svg+xml                                    svg svgz;
    image/tiff                                       tif tiff;
    image/vnd.wap.wbmp                               wbmp;
    image/webp                                       webp;
    image/x-icon                                     ico;
    image/x-jng                                      jng;
    image/x-ms-bmp                                   bmp;

    font/woff                                        woff;
    font/woff2                                       woff2;

    application/java-archive                         jar war ear;
    application/json                                 json;
    application/mac-binhex40                         hqx;
    application/msword                               doc;
    application/pdf                                  pdf;
    application/postscript                           ps eps ai;
    application/rtf                                  rtf;
    application/vnd.apple.mpegurl                    m3u8;
    application/vnd.google-earth.kml+xml             kml;
    application/vnd.google-earth.kmz                 kmz;
    application/vnd.ms-excel                         xls;
    application/vnd.ms-fontobject                    eot;
    application/vnd.ms-powerpoint                    ppt;
    application/vnd.oasis.opendocument.graphics      odg;
    application/vnd.oasis.opendocument.presentation  odp;
    application/vnd.oasis.opendocument.spreadsheet   ods;
    application/vnd.oasis.opendocument.text          odt;
    application/vnd.openxmlformats-officedocument.presentationml.presentation
                                                     pptx;
    application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
                                                     xlsx;
    application/vnd.openxmlformats-officedocument.wordprocessingml.document
                                                     docx;
    application/vnd.wap.wmlc                         wmlc;
"mime.types" 97L, 5231C
```

`mime.types` 意思是媒体类型，前面学习过HTTP协议就非常容易理解了，`mime.types` 作用就是：

> 当web服务器收到静态的资源文件请求时，依据请求文件的后缀名在服务器的MIME配置文件中找到对应的MIME Type，再根据MIME Type设置HTTP Response的Content-Type，然后浏览器根据Content-Type的值处理文件。
> 在浏览器中显示的内容有 HTML、有 XML、有 GIF、还有 Flash ...
> 那么，浏览器是如何区分它们，绝对什么内容用什么形式来显示呢？答案是 MIME Type，也就是该资源的媒体类型。
> 媒体类型通常是通过 HTTP 协议，由 Web 服务器告知浏览器的，更准确地说，是通过 Content-Type 来表示的，例如：
> Content-Type: text/HTML
> 表示内容是 text/HTML 类型，也就是超文本文件。

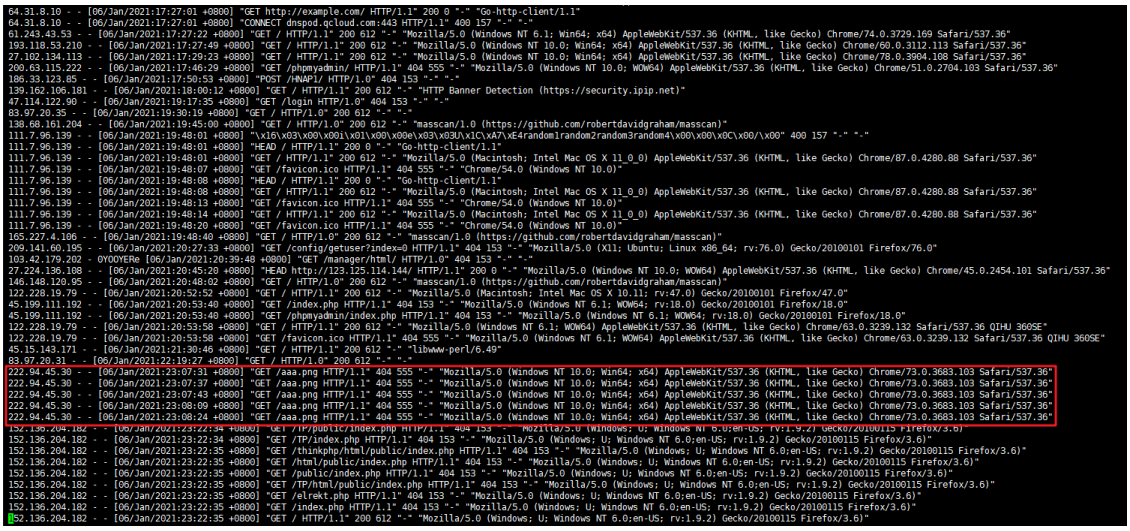
第二项配置是：

```
default_type  application/octet-stream;
```

我们上面说了，`nginx`依据请求文件的后缀名在服务器的MIME配置文件中找到对应的MIME Type，以Content-Type告诉浏览器这个文件类型。那如果nginx没有此对应的MIME Type或者nginx无法判断类型的话，则采用默认的`application/octet-stream`，这个默认表示响应是普通的文件流，让客户端下载该文件。

4.2 log_format和access_log

这个就比较简单了，第一个是指定了日志的格式，第二个是http访问的记录，只要有一次http的请求，这里都会进行记录。



可以看到我自己刚才访问了一个不存在的图片，IP、请求时间、请求url等信息都记录了下来。

4.3 sendfile和tcp_nopush和keepalive_timeout和gzip

```
#开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。注意：如果图片显示不正常把这个改成off。
#sendfile指令指定 nginx 是否调用sendfile 函数（zero copy 方式）来输出文件，对于普通应用，必须设为on。如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络IO处理速度，降低系统uptime。
sendfile        on;

#此选项允许或禁止使用socke的TCP_CORK的选项，此选项仅在使用sendfile的时候使用
#作用是数据包累计到一定大小的时候再发送，需要跟sendfile=on配合使用
#tcp_nopush     on;
```

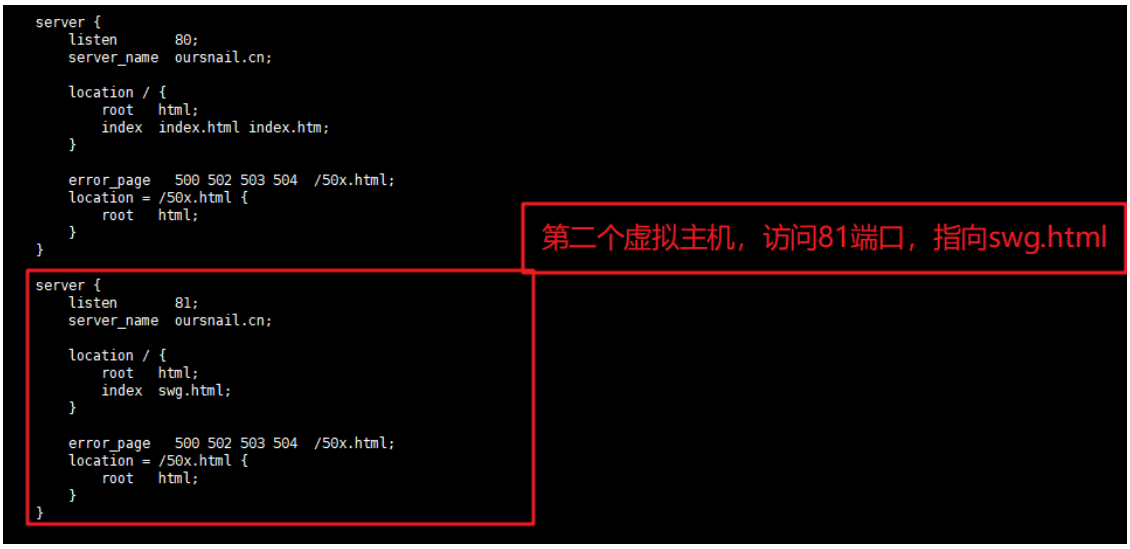
```
#客户端连接服务器的长连接超时时间，单位是秒
#HTTP本身是无状态协议，客户端发起请求后，服务端响应成功后一般会断开本次连接，不过如果客户端发送多个请求都是独立建立和销毁连接，效率比较低。如果在一个请求结束后，服务端暂时先不关闭连接，保持这个连接一段时间，那么服务端就可以利用这个连接继续处理下一个来自这个客户端的请求。
#keepalive_timeout  0;
keepalive_timeout  65;

#开启gzip压缩输出
#比如html、js会被压缩，提高http传输效率
#gzip  on;
```

4.4 server

来到了虚拟主机的配置，这也是最关键的配置。

```
server {
    #监听端口
    listen       80;
    #域名可以有多个，用空格隔开
    server_name  oursnail.cn;
    #上面匹配上了，则来到下面，默认配置的是/，即访问的是nginx根路径
    location / {
        # 默认是找到与conf与文件夹同级目录下的html目录
        root   html;
        # 默认展示html目录下的index页面
        index  index.html index.htm;
    }
    # nginx异常显示的目录
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   html;
    }
}
```

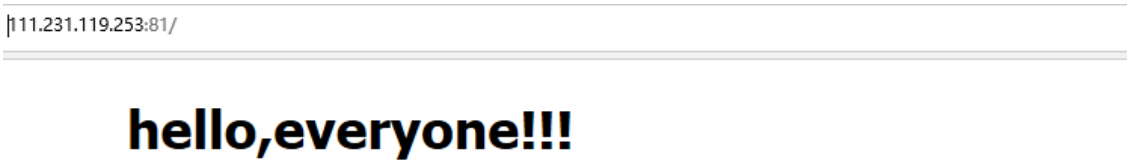
显然，我们可以配置多个虚拟主机。



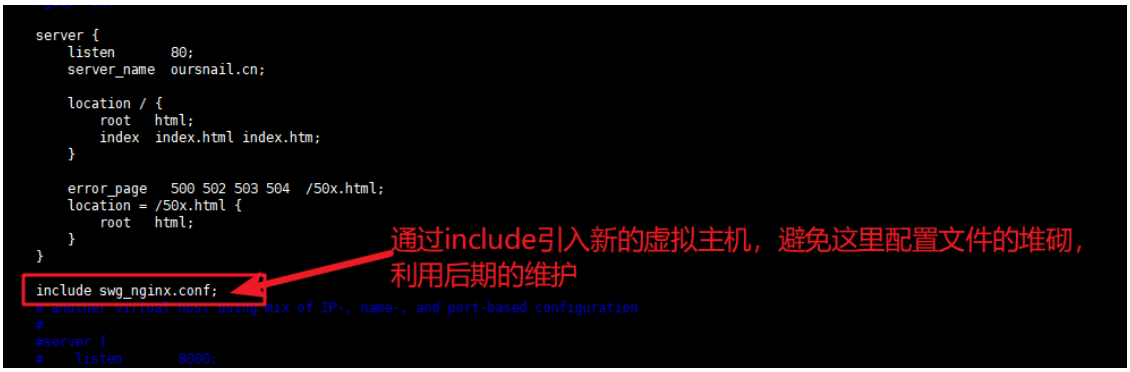
在 `/usr/local/nginx/html` 目录下我新建了 `swg.html`：

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>hello,everyone!!!<h1>
</body>
</html>
```

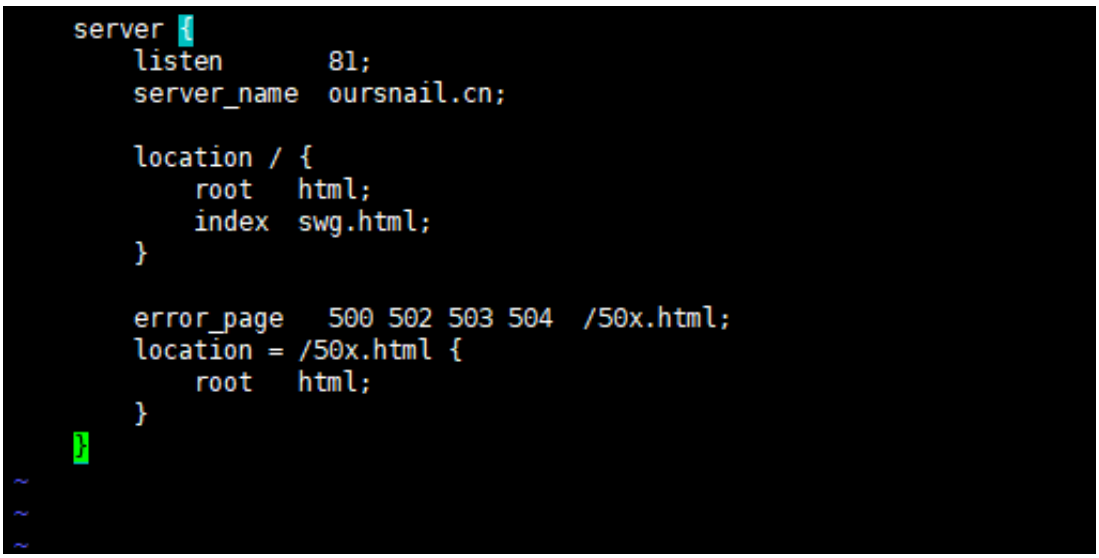
`./nginx -t` 确定配置文件无误和 `./nginx -s reload` 重启后，访问 `http://111.231.119.253:81/`：



精益求精，我们改为 `include` 的方式引入配置文件，而不是直接在 `nginx.conf` 中堆砌。



而独立的 `swg_nginx.conf` 文件内容跟前面一致：



最终效果一样，这样的话会让主配置文件看起来比较清爽，在业务模块比较庞杂时，是十分有益的。注意，我这里使用的是云服务器，记得放开81端口这个安全组限制。好了，关于配置文件，暂时就先介绍到这里。

五、nginx常用命令介绍

最后补充下nginx的最常用的命令。

> ./nginx -s stop：暴力关闭，不管现在有没有客户端连接，一般是碰到了非法用户可紧急关闭。
> ./nginx -s quit：优雅停止，如果当前没有用户请求，则关闭；如果还有用户连接，等用户连接断开再关闭。
> ./nginx -t：检测配置文件是否正确
> ./nginx -v：简单展示当前版本号
> ./nginx -V：比较详细的环境信息，包括版本号
> ./nginx：启动nginx
> ./nginx -s reload：重启