$time\_iso8601

# nginx日志配置 - 运维生存时间

http://www.ttlsa.com/linux/the-nginx-log-configuration/

日志对于统计排错来说非常有利的。本文总结了[nginx](http://www.ttlsa.com/nginx/" \t "_blank" \o "nginx)日志相关的配置如access\_log、log\_format、open\_log\_file\_cache、log\_not\_found、log\_subrequest、rewrite\_log、error\_log。  
nginx有一个非常灵活的日志记录模式。每个级别的配置可以有各自独立的访问日志。日志格式通过log\_format命令来定义。ngx\_http\_log\_module是用来定义请求日志格式的。

**1.** access\_log指令

语法: access\_log path [format [buffer=size [flush=time]]];  
access\_log path format gzip[=level] [buffer=size] [flush=time];  
access\_log syslog:server=address[,parameter=value] [format];  
access\_log off;  
默认值: access\_log logs/access.log combined;  
配置段: http, server, location, if in location, limit\_except  
gzip压缩等级。  
buffer设置内存缓存区大小。  
flush保存在缓存区中的最长时间。  
不记录日志：access\_log off;  
使用默认combined格式记录日志：access\_log logs/access.log 或 access\_log logs/access.log combined;

**2.** log\_format指令

语法: log\_format name string …;  
默认值: log\_format combined “…”;  
配置段: http

name表示格式名称，string表示等义的格式。log\_format有一个默认的无需设置的combined日志格式，相当于apache的combined日志格式，如下所示：

|  |  |
| --- | --- |
| 1  2  3 | log\_format  combined  '$remote\_addr - $remote\_user  [$time\_local]  '                                     ' "$request"  $status  $body\_bytes\_sent  '                                     ' "$http\_referer"  "$http\_user\_agent" '; |

如果nginx位于负载均衡器，squid，nginx反向代理之后，web服务器无法直接获取到客户端真实的IP地址了。 $remote\_addr获取反向代理的IP地址。反向代理服务器在转发请求的http头信息中，可以增加X-Forwarded-For信息，用来记录 客户端IP地址和客户端请求的服务器地址。PS: [获取用户真实IP](http://www.ttlsa.com/html/2235.html) 参见<http://www.ttlsa.com/html/2235.html>如下所示：

|  |  |
| --- | --- |
| 1  2  3 | log\_format  porxy  '$http\_x\_forwarded\_for - $remote\_user  [$time\_local]  '                               ' "$request"  $status $body\_bytes\_sent '                               ' "$http\_referer"  "$http\_user\_agent" '; |

日志格式允许包含的变量注释如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | $remote\_addr, $http\_x\_forwarded\_for 记录客户端IP地址  $remote\_user 记录客户端用户名称  $request 记录请求的URL和HTTP协议  $status 记录请求状态  $body\_bytes\_sent 发送给客户端的字节数，不包括响应头的大小； 该变量与Apache模块mod\_log\_config里的“%B”参数兼容。  $bytes\_sent 发送给客户端的总字节数。  $connection 连接的序列号。  $connection\_requests 当前通过一个连接获得的请求数量。  $msec 日志写入时间。单位为秒，精度是毫秒。  $pipe 如果请求是通过HTTP流水线(pipelined)发送，pipe值为“p”，否则为“.”。  $http\_referer 记录从哪个页面链接访问过来的  $http\_user\_agent 记录客户端浏览器相关信息  $request\_length 请求的长度（包括请求行，请求头和请求正文）。  $request\_time 请求处理时间，单位为秒，精度毫秒； 从读入客户端的第一个字节开始，直到把最后一个字符发送给客户端后进行日志写入为止。  $time\_iso8601 ISO8601标准格式下的本地时间。  $time\_local 通用日志格式下的本地时间。 |

[warning]发送给客户端的响应头拥有“sent\_http\_”前缀。 比如$sent\_http\_content\_range。[/warning]

实例如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | http {  log\_format  main  '$remote\_addr - $remote\_user [$time\_local] "$request" '                                          '"$status" $body\_bytes\_sent "$http\_referer" '                                          '"$http\_user\_agent" "$http\_x\_forwarded\_for" '                                          '"$gzip\_ratio" $request\_time $bytes\_sent $request\_length';    log\_format srcache\_log '$remote\_addr - $remote\_user [$time\_local] "$request" '                                  '"$status" $body\_bytes\_sent $request\_time $bytes\_sent $request\_length '                                  '[$upstream\_response\_time] [$srcache\_fetch\_status] [$srcache\_store\_status] [$srcache\_expire]';    open\_log\_file\_cache max=1000 inactive=60s;    server {  server\_name ~^(www\.)?(.+)$;  access\_log logs/$2-access.log main;  error\_log logs/$2-error.log;    location /srcache {  access\_log logs/access-srcache.log srcache\_log;  }  }  } |

**3. open\_log\_file\_cache指令**

语法: open\_log\_file\_cache max=N [inactive=time] [min\_uses=N] [valid=time];  
open\_log\_file\_cache off;  
默认值: open\_log\_file\_cache off;  
配置段: http, server, location

对于每一条日志记录，都将是先打开文件，再写入日志，然后关闭。可以使用open\_log\_file\_cache来设置日志文件缓存(默认是off)，格式如下：  
参数注释如下：  
max:设置缓存中的最大文件描述符数量，如果缓存被占满，采用LRU算法将描述符关闭。  
inactive:设置存活时间，默认是10s  
min\_uses:设置在inactive时间段内，日志文件最少使用多少次后，该日志文件描述符记入缓存中，默认是1次  
valid:设置检查频率，默认60s  
off：禁用缓存  
实例如下：

|  |  |
| --- | --- |
| 1 | open\_log\_file\_cache max=1000 inactive=20s valid=1m min\_uses=2; |

**4.** log\_not\_found指令

语法: log\_not\_found on | off;  
默认值: log\_not\_found on;  
配置段: http, server, location  
是否在error\_log中记录不存在的错误。默认是。

**5.** log\_subrequest指令

语法: log\_subrequest on | off;  
默认值: log\_subrequest off;  
配置段: http, server, location  
是否在access\_log中记录子请求的访问日志。默认不记录。

**6. rewrite\_log**指**令**

由ngx\_http\_rewrite\_module模块提供的。用来记录重写日志的。对于调试重写规则建议开启。 [Nginx重写规则指南](http://www.ttlsa.com/html/2689.html)  
语法: rewrite\_log on | off;  
默认值: rewrite\_log off;  
配置段: http, server, location, if  
启用时将在error log中记录notice级别的重写日志。

**7.** error\_log指令

语法: error\_log file | stderr | syslog:server=address[,parameter=value] [debug | info | notice | warn | error | crit | alert | emerg];  
默认值: error\_log logs/error.log error;  
配置段: main, http, server, location  
配置错误日志。

# 日志服务 - [ OpenResty最佳实践-最新版 ]

https://www.kancloud.cn/allanyu/openresty-best-practices/82610

日志服务

Nginx日志主要有两种：access\_log(访问日志)、error\_log(错误日志)。

**access\_log(访问日志)**

access\_log主要记录客户端访问Nginx的每一个请求，格式可以自定义。通过access\_log，你可以得到用户地域来源、跳转来源、使用终端、某个URL访问量等相关信息。

log\_format指令用于定义日志的格式，语法: log\_format name string; 其中name表示格式名称，string表示定义的格式字符串。log\_format有一个默认的无需设置的combined日志格式。

默认的无需设置的combined日志格式

log\_format combined '$remote\_addr - $remote\_user [$time\_local] '

' "$request" $status $body\_bytes\_sent '

' "$http\_referer" "$http\_user\_agent" ';

access\_log指令用来指定访问日志文件的存放路径（包含日志文件名）、格式和缓存大小，语法：access\_log path [format\_name [buffer=size | off]]; 其中path表示访问日志存放路径，format\_name表示访问日志格式名称，buffer表示缓存大小，off表示关闭访问日志。

log\_format使用事例：在access.log中记录客户端IP地址、请求状态和请求时间

log\_format myformat '$remote\_addr $status $time\_local';

access\_log logs/access.log myformat;

需要注意的是：log\_format配置必须放在http内，否则会出现警告。Nginx进程设置的用户和组必须对日志路径有创建文件的权限，否则，会报错。

定义日志使用的字段及其作用：

| **字段** | **作用** |
| --- | --- |
| $remote\_addr与$http\_x\_forwarded\_for | 记录客户端IP地址 |
| $remote\_user | 记录客户端用户名称 |
| $request | 记录请求的URI和HTTP协议 |
| $status | 记录请求状态 |
| $body\_bytes\_sent | 发送给客户端的字节数，不包括响应头的大小 |
| $bytes\_sent | 发送给客户端的总字节数 |
| $connection | 连接的序列号 |
| $connection\_requests | 当前通过一个连接获得的请求数量 |
| $msec | 日志写入时间。单位为秒，精度是毫秒 |
| $pipe | 如果请求是通过HTTP流水线(pipelined)发送，pipe值为“p”，否则为“.” |
| $http\_referer | 记录从哪个页面链接访问过来的 |
| $http\_user\_agent | 记录客户端浏览器相关信息 |
| $request\_length | 请求的长度（包括请求行，请求头和请求正文） |
| $request\_time | 请求处理时间，单位为秒，精度毫秒 |
| $time\_iso8601 | ISO8601标准格式下的本地时间 |
| $time\_local | 记录访问时间与时区 |

**error\_log(错误日志)**

error\_log主要记录客户端访问Nginx出错时的日志，格式不支持自定义。通过查看错误日志，你可以得到系统某个服务或server的性能瓶颈等。因此，将日志好好利用，你可以得到很多有价值的信息。

error\_log指令用来指定错误日志，语法: error\_log path(存放路径) level(日志等级); 其中path表示错误日志存放路径，level表示错误日志等级，日志等级包括debug、info、notice、warn、error、crit，从左至右，日志详细程度逐级递减，即debug最详细，crit最少，默认为crit。

注意：error\_log off并不能关闭错误日志记录，此时日志信息会被写入到文件名为off的文件当中。如果要关闭错误日志记录，可以使用如下配置：

Linux系统把存储位置设置为空设备

error\_log /dev/null;

Windows系统把存储位置设置为空设备

error\_log nul;

另外Linux系统可以使用tail命令方便的查阅正在改变的文件,tail -f filename会把filename里最尾部的内容显示在屏幕上,并且不断刷新,使你看到最新的文件内容。Windows系统没有这个命令，你可以在网上找到动态查看文件的工具。

# Log rotation directly within Nginx configuration file - Cambus.net

https://www.cambus.net/log-rotation-directly-within-nginx-configuration-file/

Log rotation directly within Nginx configuration file

2014-05-04 18:17 · [Nginx](https://www.cambus.net/categories/nginx)

Ever since I discovered PostgreSQL allowed to embed variables in log\_filename allowing to split logs without using logrotate or cronolog, I've been wanting to do the same with Nginx.

从那时起我发现PostgreSQL在log\_filename可以配置嵌入的参数，允许切分日志无须借助工具logrotate和cronolog。我一直想要在Nginx做同样的事情

As it turns out, it's not only possible but also pretty easy to achieve, since **Nginx** introduced the possibility to use **variables in access\_log** directives (added in 0.7.4), and the **$time\_iso8601** variable providing time in ISO 8601 format (added in 0.9.6).

后来人们发现，不仅可能,而且很容易实现。在0.7.4中引入了access\_log，在0.9.6中添加了$time\_iso8601变量

Please note, that it's not possible to embed **variables in error\_log** directives, as any potential error would not be able to be logged if the file cannot be written.

Here is an example showing $time\_iso8601 format :

2014-05-04T18:12:02+02:00

All there is to do is use an "if" block to split required data using a regular expression and set time variables.

For splitting logs by day, one would use the following snippet in the server block :

if ($time\_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})") {

set $year $1;

set $month $2;

set $day $3;

}

access\_log /var/log/nginx/$year-$month-$day-access.log;

Alternatively, we can also use the Perl compatible syntax for named regular expression capture :

if ($time\_iso8601 ~ "^(?&lt;year&gt;\d{4})-(?&lt;month&gt;\d{2})-(?&lt;day&gt;\d{2})") {}

access\_log /var/log/nginx/$year-$month-$day-access.log;

To also create variables for hours, minutes and seconds, we can use the following snippet :

if ($time\_iso8601 ~ "^(\d{4})-(\d{2})-(\d{2})T(\d{2}):(\d{2}):(\d{2})") {

set $year $1;

set $month $2;

set $day $3;

set $hour $4;

set $minutes $5;

set $seconds $6;

}

Lastly, and I cannot stress it enough, please make sure the **Nginx workers** have proper permissions to access and write to the log directory.

This method is currently used in production on a busy Nginx server and has been running flawlessly for almost a month now. The only drawback is that buffered writes do not work when the access\_log file path contain variables, so it isn't possible to gzip logs on the fly.