链滴

Nginx 配置说明

作者：QForever

原文链接：https://ld246.com/article/1586002423167

来源网站：链滴

```
# 表示工作进程的数量
worker_processes auto;
# 进程文件位置
pid /run/nginx.pid;
# worker进程最大打开文件数
worker_rlimit_nofile 261120;
# 设置多少秒后关闭连接
worker_shutdown_timeout 10s ;
# 如果后端是一个动态程序，那么 upstream 的 fail_timeout 应该设置为 0 将 fail_timeout 设置为 0，那么无论后端失败了多少次， NGINX 会继续把请求分发到这个后端服务器地址。
max_fails=0 fail_timeout=0
# 如果并发请求超过了 keepalive 指定的最大连接数，Nginx 会启动新的连接 来转发请求，新连接请求完毕后关闭，而且新建立的连接是长连接，这可能会造成额外的问题，
#  keepalive 指定的 数值 是 Nginx 每个 worker  连接后端的最大长连接数，而不是整个 Nginx 的。而且这里的后端指的是「所有的后端」，而不是每一个后端(已验证)。
keepalive 32;
# events模块
events {
# 设置网路连接序列化，防止惊群现象发生，默认为on
accept_mutex on;
# 设置一个进程是否同时接受多个网络连接，默认为off
multi_accept on;
# 事件驱动模型，使用epoll的I/O 模型(值得注意的是如果你不知道Nginx该使用哪种轮询方法的话，会选择一个最适合你操作系统的)
use epoll;
# 最大连接数
worker_connections 1024;
# 客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个请求头的大小不超过1k，不过由于一般系统分页都要大于1k，所以这里设置为系统分页大小。查看系统分页可以使用etconf PAGESIZE命令
client_header_buffer_size 4k;
# 为打开文件指定缓存，默认是没有启用的，max指定缓存最大数量，建议和打开文件数一致，inactive是指经过多长时间文件没被请求后删除缓存 打开文件最大数量为我们再main配置的worker_rlimit_nfile参数
open_file_cache max=2000 inactive=60s;
# 这个是指多长时间检查一次缓存的有效信息。如果有一个文件在inactive时间内一次没被使用，它被移除
open_file_cache_valid 60s;
# open_file_cache指令中的inactive参数时间内文件的最少使用次数，如果超过这个数字，文件描述一直是在缓存中打开的，如果有一个文件在inactive时间内一次没被使用，它将被移除。
open_file_cache_min_uses 1
}
# nginx的headers_more模块的使用
more_set_headers 用于 添加、修改、清除 响应头
more_clear_headers 用于 清除 响应头
more_set_input_headers 用于 添加、修改、清除 请求头
more_clear_input_headers 用于 清除 请求头
# 通过指定port_in_redirect off;告知nginx在redirect的时候不要带上port，如果没有配置，默认该为true
port_in_redirect off;
```

补充

```
# nginx针对不同的操作系统，有不同的事件驱动模型
```

```
# A、标准事件模型
    Select、poll属于标准事件模型，如果当前系统不存在更有效的方法，nginx会选择select或poll
# B、高效事件模型
    Kqueue：使用于FreeBSD 4.1+, OpenBSD 2.9+, NetBSD 2.0 和 MacOS X.使用双处理器的Mac
S X系统使用kqueue可能会造成内核崩溃。
    Epoll:使用于Linux内核2.6版本及以后的系统。
    /dev/poll：使用于Solaris 7 11/99+, HP/UX 11.22+ (eventport), IRIX 6.5.15+ 和 Tru64 UNIX 5
1A+。
    Eventport：使用于Solaris 10. 为了防止出现内核崩溃的问题， 有必要安装安全补丁
```

map指令

```
# 匹配请求 url 的参数，如果参数是 debug 则设置 $foo = 1 ，默认设置 $foo = 0
map $args $foo {
    default 0;
    debug   1;
}
# $args 是nginx内置变量，就是获取的请求 url 的参数。 如果 $args 匹配到 debug 那么 $foo 的值
被设为 1 ，如果 $args 一个都匹配不到 $foo 就是default 定义的值，在这里就是 0
```

map语法

```
map $var1 $var2 {...}
```

map 指令的三个参数

```
1、default ： 指定源变量匹配不到任何表达式时将使用的默认值。当没有设置 default，将会用一个
的字符串作为默认的结果。
2、hostnames ： 允许用前缀或者后缀掩码指定域名作为源变量值。这个参数必须写在值映射列表
最前面。
3、include ： 包含一个或多个含有映射值的文件。
```

注意：在 Nginx 配置文件中的作用段： http{} ，注意 map 不能写在 server{} 否则会报错

```
map 的 $var1 为源变量，通常可以是 nginx 的内置变量，$var2 是自定义变量。 $var2 的值取决于
var1 在对应表达式的匹配情况。 如果一个都匹配不到则 $var2 就是 default 对应的值。
```

一个正则表达式如果以 “~” 开头，表示这个正则表达式对大小写敏感。以 “~*” 开头，表示这个则表达式对大小写不敏感。

```
map $http_user_agent $agent {
    default "";
    ~curl curl;
    ~*apachebench" ab;
}
```

正则表达式里可以包含命名捕获和位置捕获，这些变量可以跟结果变量一起被其它指令使用。

```
map $uri $value {
    /abc                    /index.php;
    ~^/teacher/(?<suffix>.*)$  /boy/;
    ~/fz(/.*)                /index.php?fz=1;
}
```

==注意：不能在map块里面引用命名捕获或位置捕获变量。如~^/qupeicom/(.*) /peiyin/$1; 这样

报错nginx: [emerg] unknown variable==

==注意二：如果源变量值包含特殊字符如 ‘~’ ，则要以 ‘\’ 来转义。==

```
map $http_referer $value {
    Mozilla    111;
    \~Mozilla  222;
}
```

源变量匹配表达式对应的结果值可以是一个字符串也可以是另外一个变量。

```
map $http_referer $value {
    Mozilla    'chrom';
    \~safity    $http_user_agent;
```

使用 map 来实现允许多个域名跨域访问的问题

如果是允许单域名跨域访问直接配置就行了，如下：

```
# 这些配置可以写在 http{} 或者 server{} 都是支持的。
add_header Access-Control-Allow-Origin "http://www.tutu.com";
add_header Access-Control-Allow-Methods "POST, GET, PUT, OPTIONS, DELETE";
add_header Access-Control-Max-Age "3600";
add_header Access-Control-Allow-Headers "Origin, X-Requested-With, Content-Type, Accept;

#!/bin/bash
# 上面的配置只允许 http://www.tutu.com 跨域访问，如果要支持所有域名都可以跨域调用该站。
上面一行改成这样，不过不推荐这样做，因为不安全
add_header Access-Control-Allow-Origin "*";
```

如果不想允许所有，但是又需要允许多个域名，那么就需要用到 map

```
map $http_origin $corsHost {
    default 0;
    "~http://www.haibakeji.com" http://www.haibakeji.com;
    "~http://m.haibakeji.com" http://m.haibakeji.com;
    "~http://wap.haibakeji.com" http://wap.haibakeji.com;
}
server
{
    listen 80;
    server_name www.haibakeji.com;
    root /nginx;
    location /
    {
        add_header Access-Control-Allow-Origin $corsHost;
    }
}
```

使用源变量（通常是 nginx 内置变量）匹配一些规则，创建自定义变量，然后在页面输出. 这通常在试的时候非常有用

```
http {
map $uri $match {
    ~^/hello/(.*) http://www.hello.com/;
```

```
}
server {
    listen       8080;
    server_name  test.hello.com;

    location /hello {
        default_type text/plain;
        echo uri: $uri;
        echo match: $match;
        echo capture: $1;
        echo new: $match$1;
    }
```

map 涉及的性能问题

大家可能会有一个问题，map 既然只能用在 http 段，这是全局的啊。 这个设置会让访问所有虚拟主的请求都要匹配并设置一遍变量的值，然而事实并非如此，对于没有用到相关变量的请求来说，并不执行 map 操作。 就没有性能上的损失。

匹配优先级问题

如果匹配到多个特定的变量，如掩码和正则同时匹配，那么会按照下面的顺序进行选择：

1. 没有掩码的字符串
2. 最长的带前缀的字符串，例如: “*.example.com”
3. 最长的带后缀的字符串，例如： “mail.*”
4. 按顺序第一个先匹配的正则表达式 （在配置文件中体现的顺序）
5. 默认值

map_hash_bucket_size

- 语法: map_hash_bucket_size size;
- 默认值: map_hash_bucket_size 32|64|128;
- 配置段: http
- 指定一个映射表中的变量在哈希表中的最大值，这个值取决于处理器的缓存。
- map_hash_max_size
- 语法: map_hash_max_size size;
- 默认值: map_hash_max_size 2048;
- 配置段: http
- 设置映射表对应的哈希表的最大值。