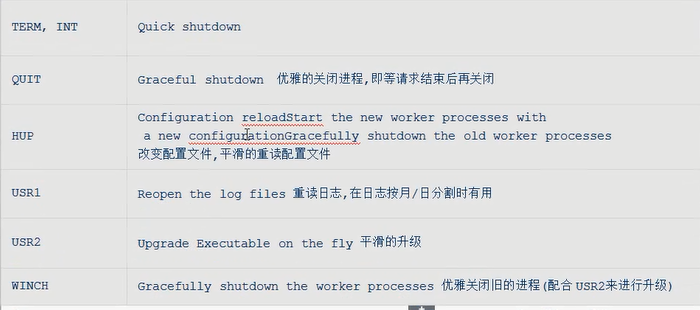
# Nginx安装

# Nginx信号控制



具体使用：Kill –信号选项 nginx端口号

Kill –INT port:迅速关闭nginx服务（不建议在生产环境建议使用）。

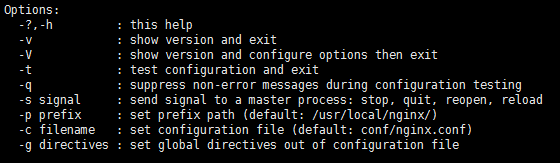
Kill –QUIT port:在请求结束后，关闭nginx服务。

Kill –HUP port：在改变配置文件后(nginx.conf),在不停止nginx服务下重新加载配置文件。

Kill –USR1 port :重新阅读日志文件(/usr/local/nginx/logs/access.log。

Kill –USR2 port:平滑的升级nginx

Kill –WINCH post: 优雅的关闭旧的进程。



# Nginx配置文件

Nginx的配置文件为nginx.conf文件，如果想使用新的配置文件启动nginx可以使用命令：/usr/local/nginx/sbin/nginx –c newConfigurationFileName

worker\_processes nums:全局配置，指有nums个工作的子进程，可以修改，但是太大没有意义，一般设置CPU个数\*内核数。

Log\_format:设置日志格式

events { worker\_connections 1024; }：一般配置nginx连接属性，这里值一个子进程最多允许的连接，可以修改。

http{server{}}:配置http服务器的主要段，server指的是虚拟的主机，可以配置多个

1. server{

listen post –监听的端口号

server\_name serverName –serverName可以为域名或IP

location{

root path –可以是相对路径或绝对路径，指定目录是上级目录，并且该上级目录含有location指定名称的同名目录。

alias –指定的目录是准确的，给location指定一个目录

index index.html index.htm --首页

access\_log logs/host.access.log main;--虚拟主机的日志文件路径和使用格式

proxy\_pass –代

}

}

注意事项：

1.使用alias时，目录名后面一定要加”/“。

2.使用alias标签的目录块中不能使用rewrite的break。

3.alias在使用正则匹配时，必须捕捉要匹配的内容并在指定的内容处使用。

4.alias只能位于location块中

# 日志

## 日志格式

1. main格式

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

Mian格式使我们定义好的日志格式，并起了一个名字方便我们调用。

2.自定义格式

$remote\_user用于记录远程客户端用户名称；

$time\_local用于记录访问时间与时区；

$request用于记录请求URL与HTTP协议；

$status用于记录请求状态，例如成功时状态为200，页面找不到时状态为404；

$body\_bytes\_sent用于记录发送客户端的文件主体内容大小；

$http\_referer用于记录是从哪个页面链接访问过来的；

$http\_user\_agent用于记录客户浏览器的相关信息。

$http\_x\_forwarded\_for  当前端有代理服务器时，设置web节点记录客户端地址的配置，此参数生效的前提是代理服务器也要进行相关的x\_forwarded\_for设置

$remote\_addr             客户端地址

$remote\_user             客户端用户名称                                --

$time\_local              访问时间和时区

$request                 请求的URI和HTTP协议

$http\_host               请求地址，即浏览器中你输入的地址（IP或域名）

$status                  HTTP请求状态

$upstream\_status         upstream状态

$body\_bytes\_sent         发送给客户端文件内容大小

$http\_referer            url跳转来源

$http\_user\_agent         用户终端浏览器等信息

$ssl\_protocol            SSL协议版本

$ssl\_cipher              交换数据中的算法

$upstream\_addr           后台upstream的地址，即真正提供服务的主机地址

$request\_time            整个请求的总时间

$upstream\_response\_time  请求过程中，upstream响应时间

# Location

Location有”定位“的意思，在nginx配置中比不可少，location可以把网站的不同部分，定位到不同的处理方法上。

语法：location[=|~|~\*|^~] patt{},中括号可以不写任何参数，作为一般匹配。

大致可以分为三类：

1. location=patt{ }[精准匹配]
2. location patt {}[一般匹配]
3. location ~patt{}[正则匹配]

## location匹配顺序

1."="前缀指令匹配，如果匹配成功，则停止其他匹配。

2.普通字符串指令匹配，顺序是从长到短，匹配成功的location如果使用^~，则停止其他匹配（正则匹配）

3.正则表达式指令匹配，按照配置文件里的顺序，成功就停止其他匹配

4.如果第三步中有匹配成功，则使用该结果，否则使用第二步结果

Ps: （1）匹配的顺序是先匹配普通字符串，然后再匹配正则表达式。另外普通字符串匹配顺序是根据配置中字符长度从长到短，也就是说使用普通字符串配置的location顺序是无关紧要的，反正最后nginx会根据配置的长短来进行匹配，但是需要注意的是正则表达式按照配置文件里的顺序测试。找到第一个比配的正则表达式将停止搜索。

1. 一般情况下，匹配成功了普通字符串location后还会进行正则表达location匹配。有两种方法改变这种行为，其一就是使用“=”前缀，这时执行的是严格匹配，并且匹配成功后立即停止其他匹配，同时处理这个请求；另外一种就是使用“^~”前缀，如果把这个前缀用于一个常规字符串那么告诉nginx 如果路径匹配那么不测试正则表达式。

# Nginx重写（rewrite）

Nginx中的rewrite的主要功能是实现URL地址的重定向，

重写中使用的指令：（1）if 空格(条件) {}：设置条件在进行重写。

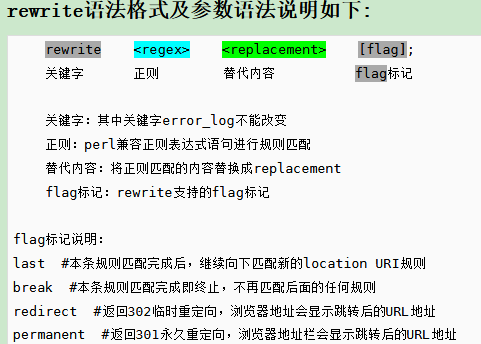
条件的三种写法：1.“=”来判断是否相等，区分大小写。

2.“~”使用正则匹配，区分大小写，“~\*”不区分大小写的正则。

3.-f -d -e判断是否为文件，为目录，是否存在。

（2）return:返回状态码

1. set:设置变量。
2. break:跳出rewrite
3. rewrite:重写。



# Nginx Module

## ngx\_http\_gzip\_module

gzip常用配置参数：

gzip on|off：是否开启gzip

gzip\_buffers 32 4K| 16 8K ：缓冲(压缩在内存中缓冲几块? 每块多大?)

gzip\_comp\_level [1-9] ：推荐6 压缩级别(级别越高,压的越小,越浪费CPU计算资源)

gzip\_disable ：正则匹配UA 什么样的Uri不进行gzip

gzip\_min\_length 200 ：开始压缩的最小长度(再小就不要压缩了,意义不在)

gzip\_http\_version 1.0|1.1 ：开始压缩的http协议版本(可以不设置,目前几乎全是1.1协议)

gzip\_proxied：设置请求者代理服务器,该如何缓存内容

gzip\_types text/plain application/xml ：对哪些类型的文件用压缩 如txt,xml,html ,css

gzip\_vary on|off ： 是否传输gzip压缩标志

## ngx\_http\_upstream\_module

### 调度算法

#### rr轮询（默认）

按客户端请求顺序把客户端的请求逐一的分配到不同的后端的服务器，如果后端服务器宕机，宕机服务器会自动剔除，是用户访问不受影响，将请求分配给正常的服务器。，

#### weight（权重）

在轮询算法的基础上加上权重（默认为rr+weight），权重轮询和访问成，权重越大，转发的请求越多，可以根据服务器的配置和性能分配权重大小，可以有效的解决新旧服务器性能不均进行请求分配问题。

#### ip\_hash

每一个请求按访问的ip的hash结果进行分配，当新的请求到达时，先将其客户端的ip通过哈希算法哈希出一个值，在随后请求客户端，ip的哈希值只要相同，就会分配到同一台机器，该调度算法可以有效的解决动态网页Session共享问题，但有时会导致请求分配不均，即无法保证1：1的负载均衡

#### fair（第三方，NO）动态算法

按照后端服务器RS的响应时间来分配请求，响应时间短有限分配，此种算法可以依据网页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本身不支持fair的，如果需要使用这种调度算法，必须瞎子啊Nginx的upstram\_fair模板

#### url\_hash（第三方，NO）

按访问url的hash结果来分配请求，让每个url定向同一个服务器，后端服务器为缓存服务器时效果明显，在upstream加入hash语句，server语句不能写入weight等其他参数，hash\_method是使用的hash算法。

url\_hash按照访问的url的hash结果分配请求，使每一个url定向到同一台后端服务器，可以进一步提高后端缓存服务器的效率命中率，Nginx本身不支持url\_hash,如果需要使用该调度算法，必须下载Nginx的hash软件包。

url\_hash(Web缓存节点)和ip\_hash(会话保持)类似。

#### least\_conn

最少连接数，哪一个机器的连接数少就分配。

#### 一致性hash

http://tengine.taobao.org/

## ngx\_http\_proxy\_module

#### http proxy模块参数

Nginx代理功能是使用http proxy模块实现的，默认在安装nginx时已经安装了http proxy模块，因此可以直接使用。

Proxy\_set\_header:设置后端的服务器获取用户的主机名或者真是ip值，以及代理 者的真是IP地址。

Client\_body\_buffer\_size:用于指定客户端请求主体缓冲区大小，可以理解为先保存到本地再传给用户。

Proxy\_connect\_timeout:表示与后端服务器连接超时的时间，即发起握手等待响应的超时时间。

Proxy\_send\_timeout:表示后端服务器的数据回传时间，即在规定的时间内，后端服务器必须传完所需的数据，否则，Nginx将断开这个连接。

Proxy\_read\_timeout:设置Nginx从代理后端服务器获取信息的时间，表示连接建立成功后，Nginx等待后端服务器响应的时间。