Nginx|Tengine

加入尚学堂，一起进步！

Nginx配置解析

- nginx.conf配置文件
  - #定义Nginx运行的用户和用户组
  - user www www;
  - #nginx进程数，建议设置为等于CPU总核心数。
  - worker_processes 8;
  - #全局错误日志定义类型，[debug | info | notice | warn | error | crit]
  - error_log /var/log/nginx/error.log info;
  - #进程文件
  - pid /var/run/nginx.pid;
  - #一个nginx进程打开的最多文件描述符数目，理论值应该是最多打开文件数（系统的值ulimit -n）与nginx进程数相除，但是nginx分配请求并不均匀，所以建议与ulimit -n的值保持一致。
  - worker_rlimit_nofile 65535;

Nginx配置解析

- #工作模式与连接数上限
- events
- {
- #参考事件模型，use [kqueue | rtsig | epoll | /dev/poll | select | poll]; epoll模型是Linux 2.6以上版本内核中的高性能网络I/O模型，如果跑在FreeBSD上面，就用kqueue模型。
- use epoll;
- #单个进程最大连接数（最大连接数=连接数*进程数）
- worker_connections 65535;
- }

Nginx配置解析

- event下的一些配置及其意义
  - #单个后台worker process进程的最大并发链接数
  - worker_connections 1024;
  - # 并发总数是 worker_processes 和 worker_connections 的乘积
  - # 即 max_clients = worker_processes * worker_connections
  - # 在设置了反向代理的情况下，max_clients = worker_processes * worker_connections / 4 为什么
  - # 为什么上面反向代理要除以4，应该说是一个经验值
  - # 根据以上条件，正常情况下的Nginx Server可以应付的最大连接数为：4 * 8000 = 32000
  - # worker_connections 值的设置跟物理内存大小有关
  - # 因为并发受IO约束，max_clients的值须小于系统可以打开的最大文件数

Nginx配置解析

- event下的一些配置及其意义
  - # 而系统可以打开的最大文件数和内存大小成正比，一般1GB内存的机器上可以打开的文件数大约是10万左右
  - # 我们来看看360M内存的VPS可以打开的文件句柄数是多少：
  - # $ cat /proc/sys/fs/file-max
  - # 输出 34336
  - # 32000 < 34336，即并发连接总数小于系统可以打开的文件句柄总数，这样就在操作系统可以承受的范围之内
  - # 所以，worker_connections 的值需根据 worker_processes 进程数目和系统可以打开的最大文件总数进行适当地进行设置
  - # 使得并发总数小于操作系统可以打开的最大文件数目
  - # 其实质也就是根据主机的物理CPU和内存进行配置
  - # 当然，理论上的并发总数可能会和实际有所偏差，因为主机还有其他的工作进程需要消耗系统资源。
  - # ulimit -SHn 65535

Nginx配置解析

- http下的一些配置及其意义
  - #设定http服务器
  - http
  - {
  - include mime.types; #文件扩展名与文件类型映射表
  - default_type application/octet-stream; #默认文件类型
  - #charset utf-8; #默认编码
  - server_names_hash_bucket_size 128; #服务器名字的hash表大小
  - client_header_buffer_size 32k; #上传文件大小限制
  - large_client_header_buffers 4 64k; #设定请求缓
  - client_max_body_size 8m; #设定请求缓
  - sendfile on; #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。
  - autoindex on; #开启目录列表访问，合适下载服务器，默认关闭。
  - tcp_nopush on; #防止网络阻塞
  - tcp_nodelay on; #防止网络阻塞
  - keepalive_timeout 120; #长连接超时时间，单位是秒

Nginx配置解析

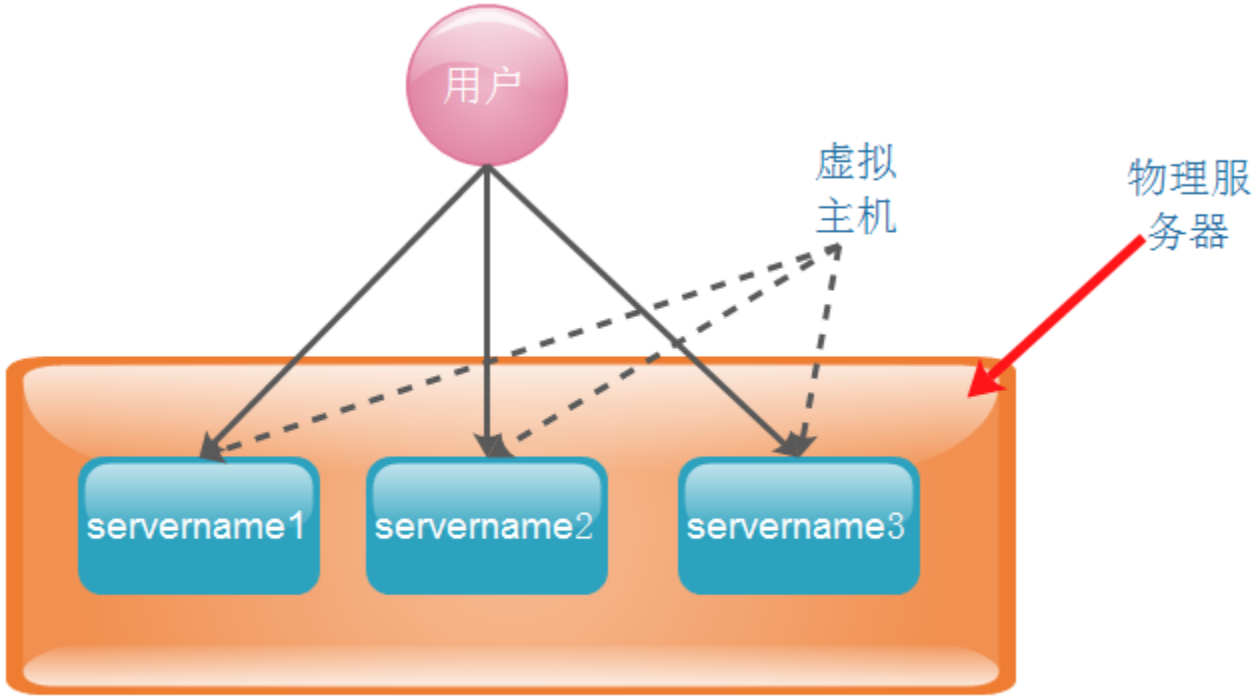
- gzip的一些配置及其意义
  - #gzip模块设置
    gzip on; #开启gzip压缩输出
    gzip_min_length 1k; #最小压缩文件大小
    gzip_buffers 4 16k; #压缩缓冲区
    gzip_http_version 1.0; #压缩版本（默认1.1，前端如果是squid2.5请使用1.0）
    gzip_comp_level 2; #压缩等级
    gzip_types text/plain application/x-javascript text/css application/xml;
    #压缩类型，默认就已经包含text/html，所以下面就不用再写了，写上去也不会有问题，但是会有一个warn。
    gzip_vary on;
    #limit_zone crawler $binary_remote_addr 10m; #开启限制IP连接数的时候需要使用

Nginx配置解析

- 虚拟主机一些配置及其意义
  - #虚拟主机的配置
    server
    {
    #监听端口
    listen 80;
    #域名可以有多个，用空格隔开
    server_name www.ha97.com ha97.com;
    index index.html index.htm index.jsp;
    root /data/www/ha97;
    location ~ .*\.(php|php5)?$
    {
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_index index.jsp;
    include fastcgi.conf;
    }

Nginx配置解析

- 什么是虚拟主机
  - 虚拟主机是一种特殊的软硬件技术，它可以将网络上的每一台计算机分成多个虚拟主机，每个虚拟主机可以独立对外提供www服务，这样就可以实现一台主机对外提供多个web服务，每个虚拟主机之间是独立的，互不影响的



Nginx配置解析

- 通过nginx可以实现虚拟主机的配置，nginx支持三种类型的虚拟主机配置，
  - 1、基于ip的虚拟主机，（一块主机绑定多个ip地址）
  - 2、基于域名的虚拟主机（servername）
  - 3、基于端口的虚拟主机（listen如果不写ip端口模式）
  - 示例基于虚拟机ip的配置，这里需要配置多个ip
  - server
  - {
  - listen 192.168.20.20:80;
  - server_name www.linuxidc.com;
  - root /data/www;
  - }
  - server
  - {
  - listen 192.168.20.21:80;
  - server_name www.linuxidc.com;
  - root /data/www;
  - }

Nginx配置解析

- nginx.conf下的配置
  - http{
  - server{
  - #表示一个虚拟主机
  - }
  - }

Nginx配置解析

- location 映射（ngx_http_core_module）
  - location [= | ~ | ~* | ^~] uri { ... }
  - location URI {}:
  - 对当前路径及子路径下的所有对象都生效；
  - location = URI {}: 注意URL最好为具体路径。
  - 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
  - 优先级：= > ^~ > ~|~* > /|/dir/

Nginx配置解析

- location配置规则
- Directives with the = prefix that match the query exactly. If found, searching stops.
- All remaining directives with conventional strings, longest match first. If this match used the ^~ prefix, searching stops.
- Regular expressions, in order of definition in the configuration file.
- If #3 yielded a match, that result is used. Else the match from #2 is used.
- =前缀的指令严格匹配这个查询。如果找到，停止搜索。
- 所有剩下的常规字符串，最长的匹配。如果这个匹配使用^~前缀，搜索停止。
- 正则表达式，在配置文件中定义的顺序。
- 如果第3条规则产生匹配的话，结果被使用。否则，如同从第2条规则被使用

Nginx配置解析

- location配置规则
- location 的执行逻辑跟 location 的编辑顺序无关。
  矫正：这句话不全对，“普通 location ” 的匹配规则是“最大前缀” ，因此“普通 location ” 的确与 location 编辑顺序无关；但是“正则 location ” 的匹配规则是“顺序匹配，且只要匹配到第一个就停止后面的匹配” ；“普通location ” 与“正则 location ” 之间的匹配顺序是？先匹配普通 location ，再“考虑”匹配正则 location 。注意这里的“考虑”是“可能”的意思，也就是说匹配完“普通 location ” 后，有的时候需要继续匹配“正则 location ” ，有的时候则不需要继续匹配“正则 location ” 。两种情况下，不需要继续匹配正则 location ：（ 1 ）当普通 location 前面指定了“ ^~ ” ，特别告诉 Nginx 本条普通 location 一旦匹配上，则不需要继续正则匹配；（ 2 ）当普通location 恰好严格匹配上，不是最大前缀匹配，则不再继续匹配正则