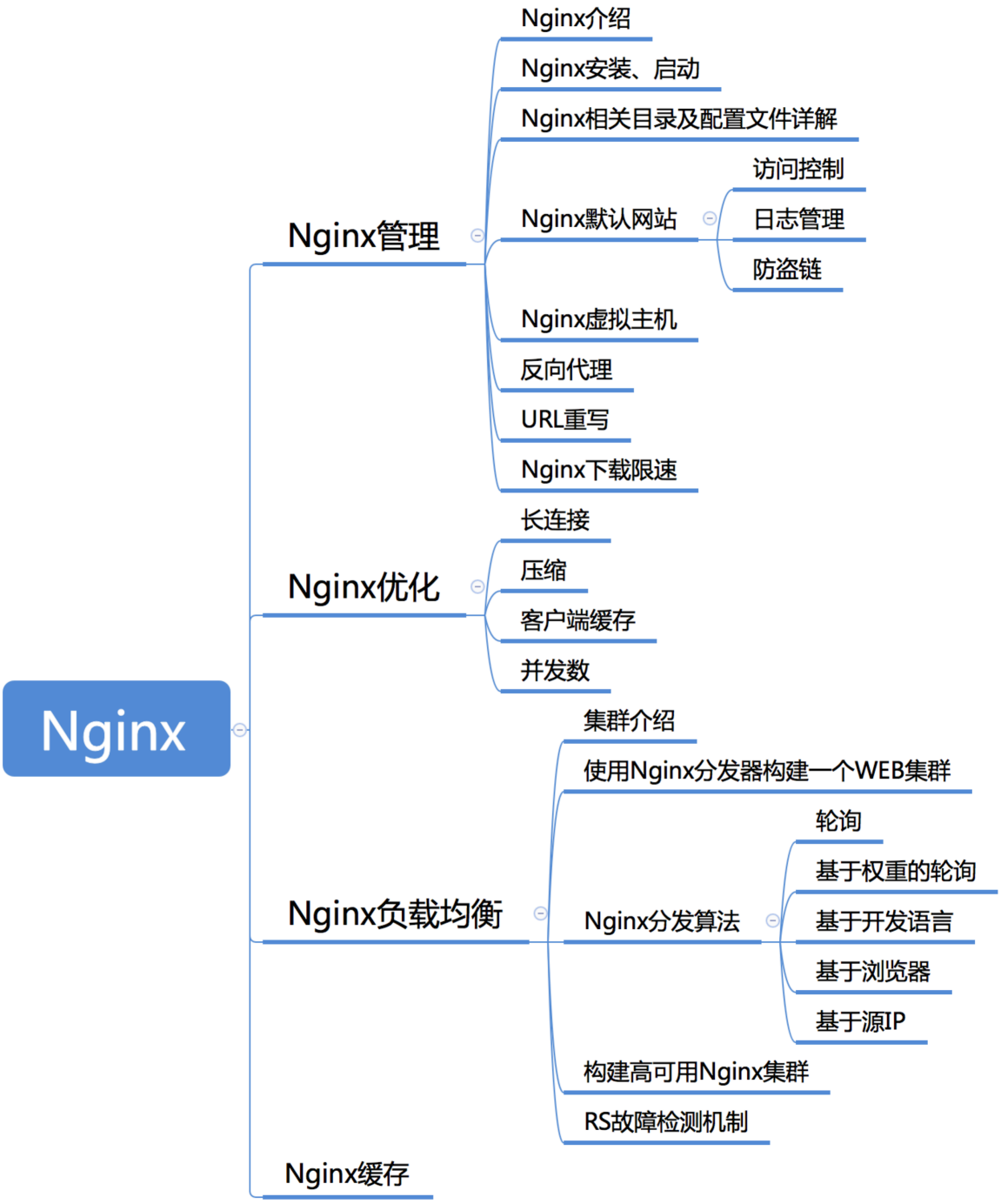
Nginx 讲义

一、课程大纲



二、nginx 安装

1、Nginx安装

1.1）获得软件

wget http://nginx.org/download/nginx-1.15.5.tar.gz -P /usr/src

1.2）安装前准备

```
cd /usr/src
tar xf nginx-1.15.5.tar.gz
cd nginx-1.15.5
yum -y install gcc pcre-devel zlib zlib-devel
```

1.3）配置

1）检查环境 是否 满足安装条件 依赖解决

2)指定安装方式 配置文件 命令文件 各种文件放哪里 开启模块功能【内置模块 三方模块】

3）指定软件安装在那里

```
./configure --prefix=/usr/local/nginx
```

1.4）编译 使用gcc将源码生成可执行程序

```
make -j4
```

1.5）安装

```
make install
```

2、相关目录

```
nginx path prefix: "/usr/local/nginx"
nginx binary file: "/usr/local/nginx/sbin/nginx"
nginx modules path: "/usr/local/nginx/modules"
nginx configuration prefix: "/usr/local/nginx/conf"
nginx configuration file: "/usr/local/nginx/conf/nginx.conf"
nginx pid file: "/usr/local/nginx/logs/nginx.pid"
nginx error log file: "/usr/local/nginx/logs/error.log"
nginx http access log file: "/usr/local/nginx/logs/access.log"
```

3、Nginx启动

/usr/local/nginx/sbin/nginx

4、验证

netstat –ntpl

lsof -i :80

5、浏览器测试

elinks　文本界面浏览器

三、Nginx配置文件详解

四、默认网站

1、默认网站

```
server {
        listen          80;
        server_name    localhost;
        location / {
            root     html;
            index    index.html index.htm;
        }
        error_page     500 502 503 504    /50x.html;
        location = /50x.html {
            root     html;
        }
    }
```

2、访问控制

```
location /a {
        allow 192.168.1.0/24;
        deny all;
        #return 404;
        return http://www.jd.com;
        }
```

3、登陆验证

```
location /c {
        auth_basic "登陆验证";
        auth_basic_user_file /etc/nginx/htpasswd;

}
```

4、日志管理

Nginx访问日志主要有两个参数控制

log_format　#用来定义记录日志的格式（可以定义多种日志格式，取不同名字即可）

access_log　#用来指定日至文件的路径及使用的何种日志格式记录日志

```
access_log    logs/access.log    main;
```

log_format格式变量：

$remote_addr　#记录访问网站的客户端地址

$remote_user　#远程客户端用户名

$time_local　#记录访问时间与时区

$request　#用户的http请求起始行信息

$status　#http状态码，记录请求返回的状态码，例如：200、301、404等

$body_bytes_sent #服务器发送给客户端的响应body字节数

$http_referer #记录此次请求是从哪个连接访问过来的，可以根据该参数进行防盗链设置。

$http_user_agent #记录客户端访问信息，例如：浏览器、手机客户端等

$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

案例

自定义一个json格式的访问日志

```
log_format main_json '{"@timestamp":"$time_local",'
'"client_ip": "$remote_addr",'
'"request": "$request",'
'"status": "$status",'
'"bytes": "$body_bytes_sent",'
'"x_forwarded": "$http_x_forwarded_for",'
'"referer": "$http_referer"'
'}';

access_log logs/access_json.log main_json;
```

5、防盗链

```
location /images/ {
alias /data/images/;
valid_referers none blocked *.ayitula.com;
if ($invalid_referer) {
rewrite ^/   http://www.ayitula.com/daolian.gif;
#return 403;
}
}
```

6、日志截断

```
mv access.log access.log.0
 killall -USR1 `cat master.nginx.pid`
 sleep 1
 gzip access.log.0
```

五、虚拟主机

就是把一台物理服务器划分成多个“虚拟”的服务器，每一个虚拟主机都可以有独立的域名和独立的目录

同时发布两个网站：

```
   DocumentRoot    /usr/local/nginx/html/web1
   DocumentRoot    /usr/local/nginx/html/web2
```

1、基于IP的虚拟主机

实现条件

1) 两个IP

2）DR 存在

3）索引页　index.html

```
#每个网站都需要一个IP
#缺点　需要多个IP　如果是公网IP　每个IP都需要付费
server {
      listen           192.168.10.42:80;
      location / {
            root     html/web1;
            index    index.html index.htm index.php;
      }
}
```

```
server {
    listen          192.168.10.52:80;
location / {
        root    html/web2;
        index   index.html index.htm;
    }
}
```

2、基于端口的虚拟主机

#只需要一个IP

#缺点　端口你是无法告诉公网用户　　无法适用于公网客户　　适合内部用户

基于端口

```
server {
    listen          80;
    #server_name    www.abc.com;
    location / {
        root    html/web1;
        index   index.html index.htm index.php;
    }
}
server {
    listen          8080;
    #server_name    www.abc.com;
    location / {
        root    html/web2;
        index   index.html index.htm;
    }
}
```

3、基于域名的虚拟主机

一个网站必然有一个域名

基于域名

```
server {
    listen          80;
    server_name    www.abc.com;

    location / {
        root     html/web1;
        index    index.html index.htm index.php;

    }
}


server {
    listen          80;
    server_name    www.cbd.com;

    location / {
        root     html/web2;
        index    index.html index.htm;
    }
}
```

六、反向代理

代理服务器，客户机在发送请求时，不会直接发送给目的主机，而是先发送给代理服务器，代理服务接受客户机请求之后，再向主机发出，并接收目的主机返回的数据，存放在代理服务器的硬盘中，再发送给客户机。

client mac http://192.168.10.42
反代 Nginx 42
业务机器 book.ayitula.com http://118.190.209.153:4000/

```
location / {
  index index.php index.html index.htm;     #定义首页索引文件的名称
proxy_pass    http://mysvr ;#请求转向mysvr  定义的服务器列表
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
client_max_body_size 10m;       #允许客户端请求的最大单文件字节数
client_body_buffer_size 128k;   #缓冲区代理缓冲用户端请求的最大字节数，
proxy_connect_timeout 90;   #nginx跟后端服务器连接超时时间(代理连接超时)
proxy_send_timeout 90;             #后端服务器数据回传时间(代理发送超时)
proxy_read_timeout 90;              #连接成功后，后端服务器响应时间(代理接收超时)
proxy_buffer_size 4k;                  #设置代理服务器（nginx）保存用户头信息的缓冲区大小
proxy_buffers 4 32k;                    #proxy_buffers缓冲区，网页平均在32k以下的话，这样设置
proxy_busy_buffers_size 64k;       #高负荷下缓冲大小（proxy_buffers*2）
proxy_temp_file_write_size 64k;   #设定缓存文件夹大小，大于这个值，将从upstream服务器传
}
```

七、限速

限流（rate limiting）是NGINX众多特性中最有用的，也是经常容易被误解和错误配置的，特性之一。该特性可以限制某个用户在一个给定时间段内能够产生的HTTP请求数。请求可以简单到就是一个对于主页的GET请求或者一个登陆表格的POST请求。

限流也可以用于安全目的上，比如减慢暴力密码破解攻击。通过限制进来的请求速率，并且（结合日志）标记出目标URLs来帮助防范DDoS攻击。一般地说，限流是用在保护上游应用服务器不被在同一时刻的大量用户请求湮没。

算法思想是：

水（请求）从上方倒入水桶，从水桶下方流出（被处理）；

来不及流出的水存在水桶中（缓冲），以固定速率流出；

水桶满后水溢出（丢弃）。

这个算法的核心是：缓存请求、匀速处理、多余的请求直接丢弃。

相比漏桶算法，令牌桶算法不同之处在于它不但有一只“桶”，还有个队列，这个桶是用来存放令牌的，队列才是用来存放请求的。

Nginx官方版本限制IP的连接和并发分别有两个模块：
limit_req_zone 用来限制单位时间内的请求数，即速率限制,采用的漏桶算法"leaky bucket"。
limit_req_conn 用来限制同一时间连接数，即并发限制。

limit_req_zone 参数配置
Syntax: limit_req zone=name [burst=number] [nodelay];
Default: —
Context: http, server, location
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

#基于IP对下载速率做限制 限制每秒处理1次请求，对突发超过5个以后的请求放入缓存区
http {
    limit_req_zone $binary_remote_addr zone=baism:10m rate=1r/s;
    server {
        location /abc {

```
            limit_req zone=baism burst=5 nodelay;
        }
}
limit_req_zone $binary_remote_addr zone=baism:10m rate=1r/s;
```

第一个参数：$binary_remote_addr 表示通过remote_addr这个标识来做限制，“binary_”的目的是缩写内存占用量，是限制同一客户端ip地址。

第二个参数 :zone=baism:10m表示生成一个大小为10M ,名字为one的内存区域，用来存储访问的频次信息。

第三个参数 :rate=1r/s表示允许相同标识的客户端的访问频次，这里限制的是每秒1次，还可以有比如30r/m的。

limit_req zone=baism burst=5 nodelay;

第一个参数：zone=baism 设置使用哪个配置区域来做限制，与上面limit_req_zone 里的name对应。

第二个参数：burst=5，重点说明一下这个配置，burst爆发的意思，这个配置的意思是设置一个大小为5的缓冲区当有大量请求（爆发）过来时，超过了访问频次限制的请求可以先放到这个缓冲区内。

第三个参数：nodelay，如果设置，超过访问频次而且缓冲区也满了的时候就会直接返回503，如果没有设置，则所有请求会等待排队。

```
#基于IP做连接限制  限制同一IP并发为1  下载速度为100K
limit_conn_zone $binary_remote_addr zone=addr:10m;

server {
        listen          80;
        server_name   localhost;

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

```
        location /abc {
            limit_conn addr 1;
            limit_rate 100k;
        }

    }

}

http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout  65;
#基于IP做连接限制  限制同一IP并发为1  下载速度为100K
limit_conn_zone $binary_remote_addr zone=addr:10m;
#基于IP对下载速率做限制  限制每秒处理1次请求，对突发超过5个以后的请求放入缓存区
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
    server {
        listen       80;
        server_name  localhost;

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

        location /abc {
```

```
            limit_req zone=one burst=5 nodelay;
            limit_conn addr 1;
            limit_rate 100k;
        }

    }

}
```

八、URL重定向

URL 模块语法

1) set 设置变量

2) if 负责语句中的判断

3) return 返回返回值或URL

4) break 终止后续的rewrite规则

5) rewrite 重定向URL

set指令 自定义变量

Syntax:

set $variable value;

Default:

—

Context:

server, location, if

将http://www.ayitula.com 重写为 http://www.ayitula.com/baism

```
location / {
                set $name baism;
                rewrite ^(.*)$ http://www.ayitula.com/$name;
        }
```

if 指令　负责判断
Syntax:
if (condition) { ... }
Default:
—
Context:
server, location

```
 location / {
                root html;
                index index.html index.htm;
                if ($http_user_agent ~* 'Chrome') {
                        break;
                        return 403;
                        #return http://www.jd.com;
                }
        }

#模糊匹配    ~匹配   !~不匹配      ~*  不区分大小写的匹配
#精确匹配   =   !=
```

return 指令　定义返回数据
Syntax:　return code [text];
return code URL;
return URL;
Default:　—

Context: server, location, if

```
location / {
                root html;
                index index.html index.htm;
                if ($http_user_agent ~* 'Chrome') {
                        return 403;
                        #return http://www.jd.com;
                }
        }
```

break 指令 停止执行当前虚拟主机的后续rewrite指令集
Syntax: break;
Default:—
Context:server, location, if

```
location / {
                root html;
                index index.html index.htm;
                if ($http_user_agent ~* 'Chrome') {
                        break;
                        return 403;
      }
        }
```

rewrite　　<regex>　　<replacement>　　[flag];
关键字　　正则　　替代内容　　flag标记

flag:
last　　#本条规则匹配完成后，继续向下匹配新的location URI规则
break　　#本条规则匹配完成即终止，不再匹配后面的任何规则
redirect　#返回302临时重定向，浏览器地址会显示跳转后的URL地址
permanent　#返回301永久重定向，浏览器地址栏会显示跳转后的URL地址

域名跳转
www.ayitula.com　　重写为　www.jd.com

```
server {
        listen          80;
        server_name www.ayitula.com;
        location / {
            rewrite ^/$ http://www.jd.com permanent ;

          }
}
```

注意:
重定向就是将网页自动转向重定向
301永久性重定向：新网址完全继承旧网址，旧网址的排名等完全清零

301重定向是网页更改地址后对搜索引擎友好的最好方法，只要不是暂时搬移的情况，都建议使用301来做转址。

302临时性重定向：对旧网址没有影响，但新网址不会有排名

搜索引擎会抓取新的内容而保留旧的网址

break　类似临时重定向

根据用户浏览器重写访问目录

如果是chrome浏览器　就将　http://192.168.10.42/$URI　　重写为 http://http://192.168.10.42/chrome/$URI

实现 步骤

1）URL重写

2）请求转给本机location

```
location / {
.....
if ($http_user_agent ~* 'chrome'){
                #^ 以什么开头 ^a
                #$ 以什么结尾 c$
                #. 除了回车以外的任意一个字符
                #* 前面的字符可以出现多次或者不出现
                #更多内容看正则表达式 re
                rewrite ^(.*)$ /chrome/$1 last;
                }


        location /chrome {
            root html ;
            index index.html;
```

```
        }
}
```

url重写后，马上发起一个新的请求，再次进入server块，重试location匹配，超过10次匹配不到报500错误，地址栏url不变
last 一般出现在server或if中

数据包走向 client-->nginx nginx告诉客户端让服务器的新地址(真实服务器)，客户端收到后再去找服务器 client--->server

作业

```
    # 访问 /baism00.html 的时候，页面内容重写到 /index.html 中
    rewrite /baism00.html /index.html last;

    # 访问 /baism01.html 的时候,页面内容重写到 /index.html 中,并停止后续的匹配
    rewrite /baism01.html /index.html break;

    # 访问 /baism02.html 的时候，页面直接302定向到 /index.html中
    rewrite /baism02.html /index.html redirect;

    # 访问 /baism03.html 的时候，页面直接301定向到 /index.html中
    rewrite /baism03.html /index.html permanent;

    # 把 /html/*.html => /post/*.html ，301定向
    rewrite ^/html/(.+?).html$ /post/$1.html permanent;
```

```
    # 把 /search/key => /search.html?keyword=key
    rewrite ^/search\/([^\/]+?)(\/|$) /search.html?keyword=$1 permanent;
```

九、优化

1、并发优化

```
worker_processes    4;
worker_cpu_affinity     0001 0010 0100 1000;

events {
    worker_connections    1024;
}
```

2、长连接

```
keepalive_timeout    0;    0代表关闭
#keepalive_timeout    100;
#keepalive_requests 8192;
```

3、压缩

```
gzip    on;
gzip_proxied any;
gzip_min_length 1k;
gzip_buffers 4 8k;
gzip_comp_level 6;
gzip_types text/plain text/css application/x-javascript application/javascript
application/xml;

 # 开启gzip
    gzip off;

    # 启用gzip压缩的最小文件，小于设置值的文件将不会压缩
```

```
    gzip_min_length 1k;

    # gzip 压缩级别，1-9，数字越大压缩的越好，也越占用CPU时间，后面会有详细说明
    gzip_comp_level 1;

    # 进行压缩的文件类型。javascript有多种形式。其中的值可以在 mime.types 文件中找到。
    gzip_types text/plain application/javascript application/x-javascript text/css application/xml text/javascript application/x-httpd-php image/jpeg image/gif image/png application/vnd.ms-fontobject font/ttf font/opentype font/x-woff image/svg+xml;

    # 是否在http header中添加Vary: Accept-Encoding，建议开启
    gzip_vary on;

    # 禁用IE 6 gzip
    gzip_disable "MSIE [1-6]\.";

    # 设置压缩所需要的缓冲区大小
    gzip_buffers 32 4k;

    # 设置gzip压缩针对的HTTP协议版本
    gzip_http_version 1.0;

#找大文件
 find /usr/share/    -type f -size +1M

4、客户端缓存
location ~*    \.(png|gif)$ {
                    expires 1h;
```

}

测试　浏览器刷新　　以Chrome为例
ctrl+f5　清空本地缓存从服务器拿数据
F5或者 点击 浏览器的刷新图标　　优先从本地找　然后 去找服务器核对信息是否一致　一致 返回304 从本地那数据
回车　从本地缓存拿数据
视频中刷新方式是点击图标 ,所以会拿不到数据 因为去服务器核实数据的时候已经没有数据了