一、nginx正则及location匹配

1.rpm格式安装nginx

```
[root@localhost ~]# rpm -ivh
http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-
0.el7.ngx.noarch.rpm
获取http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-
0.el7.ngx.noarch.rpm
警告：/var/tmp/rpm-tmp.qoXCAi: 头V4 RSA/SHA1 Signature, 密钥 ID 7bd9bf62: NOKEY
准备中...                          ################################# [100%]
正在升级/安装...
   1:nginx-release-centos-7-0.el7.ngx ################################# [100%]
[root@localhost ~]# ls /etc/yum.repos.d/
backup  Centos-7.repo  nginx.repo
[root@gcc ~]# yum -y install nginx
[root@gcc ~]# systemctl start nginx
[root@gcc ~]# netstat -lnpt | grep :80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
63294/nginx: master
```

```
[root@gcc ~]# systemctl enable nginx
```

注意：在生产环境中，一定要敲这条命令，不然下次重启会有麻烦

```
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service
to /usr/lib/systemd/system/nginx.s
[root@gcc ~]# rpm -q nginx
nginx-1.16.1-1.el7.ngx.x86_64
[root@gcc ~]# rpm -ql nginx
/etc/nginx/nginx.conf    #主配置文件
/usr/share/nginx/html    #网页文件
```

2.Nginx location规则匹配

^~　　（强制）标识符匹配后面跟一个字符串,匹配字符串后将停止对后续的正则表达式进行匹配，

　　如location ^~ /images/，在匹配了/images/这个字符串后就停止对后续的正则匹配

=　　精准匹配，如location=/，只会匹配url为/的请求　http://www.a.com/后面啥都不能添加（优先级最高）

~　　区分大小写的匹配　　location ~ \.jsp$　　(\转义)

~*　不区分大小写的匹配　location ~* \.jsp$

!~　　对区分大小写的匹配取非

~*　　对不区分大小写的匹配取非

/　　通用匹配，如果没有其它匹配,任何请求都会被匹配到（优先级最低）

3.正则表达式

*　　重复前面的字符0次或多次

?　　重复前面的字符0次或1次

+　　重复前面的字符1次或多次

.　　匹配除换行符以外的任意1个字符

(a | b)　匹配a或b　　location ~* \.(png|jgp|bmp)$ { }

^　　以...开头

$　　以...结尾

{n}　　重复前面的字符n次

{n,}　　重复前面的字符n次或更多次

{n,m}　　重复前面的字符n到m次

4.正则表达式补充

[a]　　匹配单个字符a

[a-z]　　匹配a-z小写字母的任意一个

[^a]　　匹配除了a以外的任意字符

[^abc]　匹配除了abc这几个字母以外的任意字符

5.Nginx location应用规则

location [= | ~ | ~* | ^~ | !~ | !~* | /] /url/ {...}

默认值: no

使用字段: server

location参数根据URL的不同需求来进行配置，可以使用字符串与正则表达式匹配

```
    location ~* .*\.jsp$ {
            proxy_pass http://tomcat_ server;
    }
```

http://www. a. com/

1.优先级例子：（越严谨优先级越高）

```
location = / {
#精确匹配/，主机名后面不能带任何字符串
[ configuration A]
}

location / {
#因为所有的地址都以/开头，所以这条规则将匹配到所有请求
#但是正则和最长字符串会优先匹配（不一定会用它）
[ configuration B ]
}

location /documents/ {
#匹配任何以/documents/ 开头的地址，匹配符合以后，还要继续往下搜索
#只有后面的正则表达式没有匹配到时，这一条才会采用这一条
[ configuration C]
}

location ~ /documents/abc/ {
#匹配任何 以/documents/abc 开头的地址， 匹配符合以后，还要继续往下搜索
#只有后面的正则表达式没有匹配到时，这一条才会采用这一条
```

```
[ configuration CC ]
}

location ^~ /images/ {
#匹配任何以/images/ 开头的地址，匹配符合以后，停止往下搜索正则，采用这一条
[ configuration D ]
}

location ~* \.(gif | jpg | jpeg)$ {
#匹配所有以gifipg或jpeg结尾的请求
#然而，所有请求/images/ 下的图片会被configD处理，因为^~到达不了这一条正则
#那什么时候会生效呢，当你的图片不在/images/ 的目录下时，会用这条匹配
[ configuration E ]
}

location /images / {
#字符匹配到/images/, 继续往下，会发现^~存在
[ configuration F ]
}

location /images/abc {
#最长字符匹配到/images/abc, 继续往下，会发现^~存在
#F与G的放置顺序是没有关系的
[ configuration G ]
}

location ~ /images/abc/ {
#只有去掉configD才有效；先最长匹配configG开头的地址，继续往下搜索，匹配到这一条正则，采用
[ configuration H ]
}
```

ps: ~ ~* 谁的优先级高呢？　　答：~的优先级较高，它是要区分大小写的

2.匹配顺序优先级

(location =) > (location 完整路径) > (location *~ 路径) > (location ~, ~*正则顺序) > (location 部分起始路径) > (/)

按照上面的location匹配，分析以下案例

/ ->config A
精确完全匹配，即使/index. html也匹配不了

/ downloads/ download. html ->config B
匹配B以后，往下没有任何匹配，采用B

/ images/1.gif -> configuration D
匹配到B，往下匹配到D，停止往下

/ images/abc/def ->config D
最长匹配到G，往下匹配D，停止往下
你可以看到任何以/ images/开头的都会匹配到D并停止，FG写在这里是没有任何意义的
H是永远轮不到的，这里只是为了说明匹配顺序

/documents/ document. html ->config C
匹配到C，往下没有任何匹配，采用C

/documents/1. jpg -> configuration B
匹配到C，往下正则匹配到E

/ documents/abc ->config CC
最长匹配到C，往下正则顺序匹配到CC (完整路径)

6.实际使用建议

所以实际使用中，个人觉得至少有三个匹配规则定义，如下:
直接匹配网站根
通过域名访问网站首页比较频繁，
使用这个会加速处理
这里是直接转发给后端应用服务器了，也可以是一个静态首页。

1.第一个必选规则

```
location = / {
    proxy_pass http://tomcat:8080;
}
```

2.第二个必选规则是处理静态文件请求，这是nginx作为http服务器的强项

有两种配置模式，目录匹配或后缀匹配,任选其一或搭配使用

```
location ^~ /static/ {
    root /usr/local/nginx/html/static/;
}
location ~* \. (gif | jpg | jpeg | png | css | js | ico)$ {
    root /webroot/res/;
}
```

3.第三个规则就是通用规则

用来转发动态请求到后端应用服务器

非静态文件请求就默认是动态请求，自己根据实际把握

#毕竟目前的一些框架的流行，带. php,jsp.后缀的情况很少了

```
location / {
    proxy_pass http://tomcat:8080;
}
```

7.nginx.conf 主配置文件

```
[root@gcc ~]# vim /etc/nginx/nginx.conf
注意这里~  include /etc/nginx/conf.d/*.conf;
```

这个主配置文件里的内容很短，用了一个像超链接的方式，放在主配置文件中引用

可以在 /etc/nginx/conf.d/的目录下，创建很多.conf文件,名字随便起，但结尾必须得是.conf

```
[root@gcc ~]# vim /etc/nginx/conf.d/default.conf
[root@gcc ~]# cd /usr/share/nginx/html/
[root@gcc html]# ls
50x.html  index.html
[root@gcc html]# mkdir download
[root@gcc html]# touch download/{1..7}.{1..9}
[root@gcc html]# ls download/
```

ps:特殊案例（悬而未决）

```
    location = /status {
        stub_status off;
    }

    location /status {
        stub_status on;
    }
```

二、nginx rewrite 规则

1.URL和URI的区别

URI = Universal Resource Identifier

统一资源标志符，用来标识抽象或物理资源的一个紧凑字符串

URL = Universal Resource Locator

统一资源定位符，一种定位资源的主要访问机制的字符串，

一个标准的URL必须包括：protocol、host、port、path、parameter、anchor

如: http://www. crushlinux. com: 80/123/welcome. html

URN = Universal Resource Name 统一资源名称，

通过特定命名空间中的唯一名称或ID来标识资源

2.rewrite功能

1.使用nginx提供的全局变量或自己设置的变量

nginx有自己的变量　例如：

```
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                          '$status $body_bytes_sent "$http_referer" '
                        '"$http_user_agent" "$http_x_forwarded_for"';
```

2.结合正则表达式和标志位实现url重写以及重定向

- rewrite只能放在server { }, location　{ }, if { }中 -----UA分离实验
- **只能对域名后边的除去传递的参数外的字符串起作用**

  主要帮我们来改变资源路径的

  例如:

http://www. crushlinux. com/a/ we/ index. php?id=1&u=admin

只对URL中的/a/we/index. php等字符串起作用

3.Nginx的Rewrite规则

采用PCRE (Perl compatible Regular Expressions)

Perl 兼容正则表达式的语法进行规则匹配

---这也是为什么再装nginx之前要将pcre搞定的原因

如果需要Nginx的Rewrite功能，在编译安装Nginx之前，必须安装PCRE库

4.语法

rewrite 正则表达式 更换目标 [标志位];

rewrite和location功能有点像，都能实现跳转

主要区别:

rewrite 是在同一域名内

更改获取资源的路径

例如：http://www.a.com/xx.html ----> /xx/yy.html

一般都是在一个域名下实现的，域名不变--机器不变

location是对路径做控制访问或反向代理

可以使用proxy.pass到其他机器（跨机器处理）

location主要做匹配，匹配完里面的动作是什么，可以有很多种（限速、缓存、防盗链...）

很多情况下rewrite也会写在location 里

它们的执行顺序是:

1.执行 server块的rewrite指令

2.执行location匹配

3.执行选定的location中的rewrite指令

注意:

如果其中某步URI被重写，

则重新循环执行1-3，直到找到真实存在的文件;

循环超过10次，则返回500 Internal Server Error错误

5.flag标志位

1. last: 相当于Apache的[L]标记，表示完成rewrite 完了循环，**很少用**

一般写在 server 和 if 中

last不终止重写后的url匹配，即新的url会再从server走一遍匹配流程

2. break:　　本条规则匹配完成后，终止匹配，不再匹配后面的规则　当前动作以处理完成　用的比较多

break一般使用在location中

break终止重写后的匹配

break和last都能组织继续执行后面的rewrite指令

3. redirect:　　返回302**临时**重定向，浏览器地址栏会显示跳转后的URL地址

4. 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

这两个也是常用的

redirect和permanent用来实现URL跳转

浏览器地址栏会显示跳转后的URL地址

6. if指令与全局变量

if判断指令(类似于shell)

语法为 if (cnditiion){...}

对给定的条件condition进行判断

如果为真，大括号内的rewrite指令将被执行，

if 条件(conditon)可以是如下任何内容:

当表达式只是一个变量时，如果其值为空或任何以0开头的字符串时都会当作条件为false

直接比较变量和内容时，使用 = 或 !=

-f 和 !-f　　用来判断是否存在文件

-d 和 !-d　　用来判断是否存在目录

-e 和 !-e　　用来判断是否存在文件或目录

-x 和 !-x　　用来判断文件是否可执行

rewrite 常用语法:

例如:这种方式最常见

http://www.abc.com/**/a/b/test.html**

http://www.abc.com/msie**/a/b/test.html**

```
if ($http_user_agent ~ MSIE) {      # MSIE      IE浏览器的意思
rewrite ^(. *)$ /msie/$1 break;
```

```
}           //如果UA包含"MSIE", rewrite 请求到/msid/目录下

if ($request method = POST) {
return 405;
}           //如果提交方法为POST,则返回状态405 (Method not allowed)。return不能返回301,302
```

注意:

因为返回301和302不能只返回状态码，还必须有重定向的URL，

所以return指令无法返301，302

```
if (S$slow) {
limit rate 10k;
}               // 限速，$slow 可以通过set指令设置
```

ps : 限速的目的：防止单个用户占用访问流量，从而导致整个访问速度减慢的问题，限制每个用户的访问速度，让大家都能访问

```
if ($args ~ post=140) {
rewrite ^ http://example.com/  permanent;
    // 如果^是在开头的话，是连域名都会换的意思
    这里的意思是，不管你原来的域名或资源路径是啥，都会给你换1了
}       //如果query string中包含"post=140",永久重定向到example.com

location ~* \.(gif | jpg | png | swf | flv)$  {
valid_referers none blocked www.jefflei.com www.leizhenfang.com;
if ($invalid_referer)  {
return 404;
```

上面这里也可以写：rewrite ^(.*)$ /error/404.html break;

```
} //防盗链
```

7.Nginx变量

1.下面是可以用作 if 判断的变量

$args: 记录请求行中的参数，同$query_string

例如：http://www. crushlinux. com/a/ we/ index. php?id=1&u=admin

这里就是$args参数的位置

$content length:　　记录 请求头中的Content-length字段

$content type:　　记录请求头中的Content-Type 字段。

$document root: 记录当前请求在root指令中指定的值，

网页的存放位置(比如：/usr/local/nginx/html)

$host : 记录请求主机头字段，否则为服务器名称

$http_user_agent:　　记录用于记录客户端浏览器的相关信息

$http_cookie:　　记录客户端cookie信息

$limit_rate:　　记录可以限制连接速率。

$request_method:　　记录客户端请求的动作，通常为GET或POST。

$request_filename:　　记录当前请求的文件路径，由root或alias指令与URI请求生成。

$scheme :　　记录HTTP方法(如http，https)

$server_protocol;　　记录请求使用的协议，通常是HTTP/1.0或HTTP/1.1

$server_addr:　　记录服务器地址，在完成一次系统调用后可以确定这个值。

$server_name:　　记录服务器名称,域名

$server_port:　　记录请求到达服务器的端口号。

$request_url:　　记录包含请求参数的原始URI,不包含主机名，

如:”/foo/bar.php?arg=baz”

$uri　　记录不带请求参数的当前URI, $uri 不包含主机名，

例如”http://www.a.com/foo/bar.php”

$document uri:　与$uri 相同

$http_x_forwarded for　　记录远程客户端的ip地址

$remote_addr:　　记录远程客户端的IP地址

$remote_port:　　记录远程客户端的端口

$remote_user　　记录远程客户端用户名称

$time_local　　记录访问时间及时区

$request　　记录请求的URL与HTTP协议

$status　　记录请求的状态，例如成功时为200，页面找不到时为404

$body_byte_sent 记录发送给客户端的文件主体内容大小,请求主体

$http_referer　　记录是从哪个页面链接访问过来的，页面来自哪个浏览器

例如：[root@gcc ~]# vim /etc/nginx/nginx.conf

log_format　main

```
  '$remote_addr - $remote_user [$time_local] "$request" '
      远程客户端地址    用户     出现的时间    请求 状态
                          '$status $body_bytes_sent "$http_referer" '
请求主体字节          来源地址
                          '"$http_user_agent" "$http_x_forwarded_for"';
客户端浏览器相关信息          客户端地址（一般无效）
[root@gcc ~]# tail -f /var/log/nginx/access.log
192.168.200.128 - - [13/Apr/2020:11:38:42 +0800] "GET / HTTP/1.1" 304 0 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/80.0.3987.122 Safari/537.36" "-"
```

可以自己定义日志的输出格式，可以自己加一些变量

示例，如果访问的URL以"* .sh" "*.bash"结尾，则返回状态码403

```
location ~ .*\.(sh | bash)?$
{
      return 403;
}
```

将原来要访问/data目录重写为/bbs

```
rewrite ^/data/?$ /bbs/ permanent;
```

防止盗链

```
localtion ~* \. (gif | jpg | png | swf | flv)$ {
    valid_referers none blocked www.test.com * .test.com;
if($incalid_referer) {
    rewrite ^/(.*) http://www.test.com/block.html break;
}
```

2.实现域名跳转

所有对www.360buy.com的访问，rewrite 到www.jd.com

```
server {
listen  80;
server name www.jd.com;
charset utf-8;
root    html;
```

```
index index.html index.htm;
if ($host = "www.360buy.com") {
rewrite ^(.*)$ http://www.jd.com/$1 permanent;    #目的是将资源全部移动，所以用
^(.*)$
}
}
```

3.实验：实现域名重定向

```
[root@gcc ~]# vim /etc/hosts
192.168.200.104 www.360buy.com
192.168.200.104 www.jd.com
[root@gcc ~]# ping www.360buy.com
[root@gcc ~]# ping www.jd.com
[root@gcc ~]# vim /etc/nginx/conf.d/default.conf
    server_name  www.jd.com;
    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        if ($host = "www.360buy.com") {
        rewrite ^(.*)$ http://www.jd.com/$1 permanent;
    }
    }
[root@gcc ~]# systemctl restart nginx
[root@gcc ~]# elinks --dump http://www.360buy.com
```

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [1]nginx.org.
Commercial support is available at [2]nginx.com.

Thank you for using nginx.

References

Visible links

1. http://nginx.org/
2. http://nginx.com/

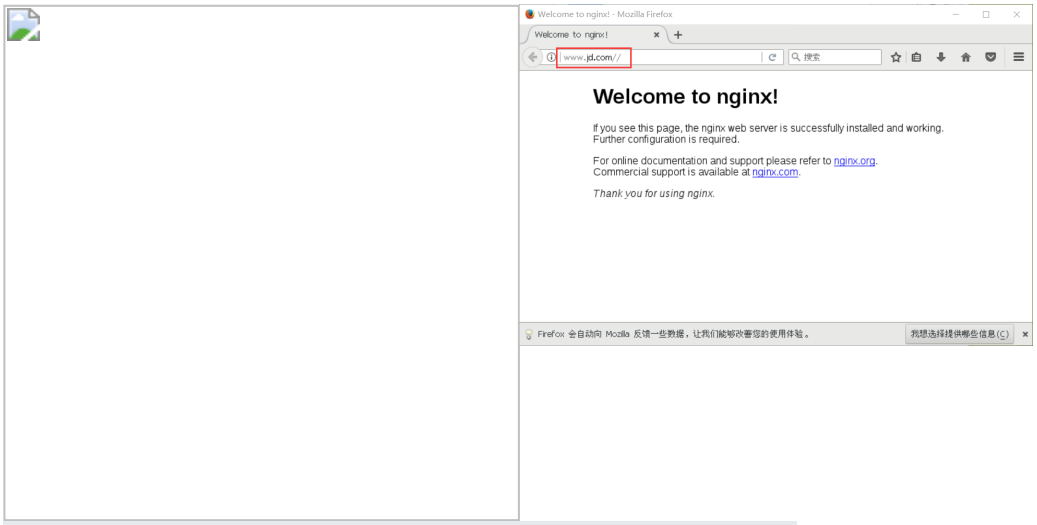
[root@gcc ~]# firefox &

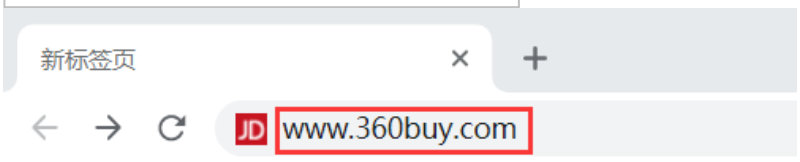
需要用到

Xmanager - Passive

在linux中打开火狐浏览器，因为只在linux中设置了hosts文件，没在windows中设置

“&”表示后台运行





面试题：

将访问www.baidu.com下的所有请求转发到http://www.baidu.com@123/

```
if ($host = "www.baidu.com") {
rewrite ^.*)$ http://www.baidu.com@123/$1 permanent;
}
```

location和rewrite这块，

建议从网上找些资料了解熟悉，并在实验里面应用一下，

自己多写，多练，要不然第一次写还是很慌的呀

这是对nginx熟练的证明

三、编译nginx平滑添加stream模块

1、操作背景

一般的，nginx+Tomcat用的是7层代理，比如：

```
location ~* .*\.jsp$ {          #这里匹配的是URL地址，而URL是http协议里才有的
    proxy_pass http://tomcat;
}
```

nginx从1.9.0版本开始

新增了ngx_stream_core_module模块，使nginx支持四层负载均衡

默认编译的时候该模块并未编译进去

需要编译的时候添加--with-stream，使其支持stream代理

那就看看现在在使用的额nginx是否包含这个新增的模块，有才能继续往下做

查关键词的方法：

```
[root@gcc ~]# vim 123    #先建一个文件 ，将要过滤的内容放入
[root@gcc ~]# grep "stream" 123
```

经过查看，是包括这个模块的

如果没有，进行nginx编译添加stream模块

只要1.9.0以上的版本的nginx就有这个功能，不需要安装包，

只需要在编译安装的时候特殊指定一下

2、nginx stream模块配置简析（企业中用的比较多　方便 简单）

注意：stream段的配置要与http段在同级目录

千万不要在这个文件里面修改，[root@gcc conf.d]# cat default.conf　因为server归属于http

```
[root@gcc ~]# cd /etc/nginx/conf.d/
[root@gcc conf.d]# ls
default.conf
[root@gcc conf.d]# cat default.conf        # 不要在这修改
[root@gcc conf.d]# vim ../nginx.conf
stream {
    server {
        listen 2222;
        proxy_connect_timeout 10s;
```

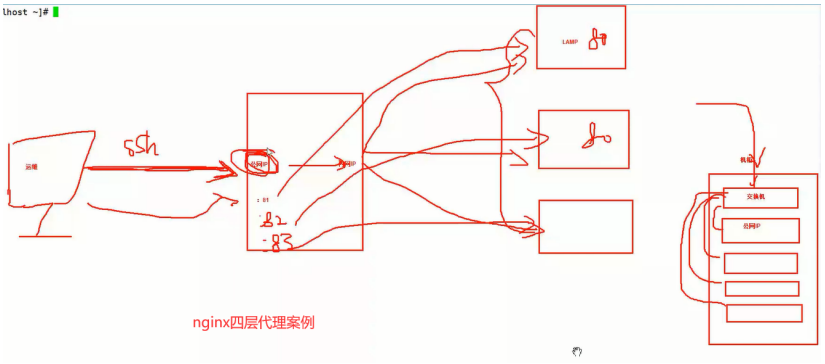
```
        proxy_timeout 10s;
        proxy_pass 192.168.200.104:22;
    }
}
[root@gcc conf.d]# nginx -t
[root@gcc conf.d]# systemctl restart nginx
[root@gcc ~]# netstat -lnpt | grep nginx
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN
67795/nginx: master
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
67795/nginx: master
```

公网IP需要花钱，这里的公网IP只有一个

机架式服务器

其他的主机通过公网IP所在机器，通过交换机内网进行通信

第一种方法：四层转发



---

client 可以连接公网IP所在机器，这台机器可以和内网所有机器进行通信，可以和其他机器进行连接　　-----这就是ssh

端口除了做远程管理以外，它还有很多端口，不见得所有端口都是用http协议，有时候为了转发TCP端口也会用到它

但是如果不是ssh,它们（内网机器）身上是跑项目的，外界怎么访问呢

这种情况的话，可以让nginx做4层转发

在nginx上开其他端口，与其他机器的80端口连接

这样可以让所有机器提供的服务对外来暴露出来

```
[root@gcc ~]# iptables -L     # 查看防火墙状态
```

新建ssh 连接

```
[root@gcc ~]# ifconfig
```

```
ens32: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.200.104  netmask 255.255.255.0  broadcast 192.168.200.255
```

例子：利用stream模块代理 zk服务的2181端口

```
stream {
        upstream zk_server {
                server 172.16.3.8:2181 weight=5;
        }
        server {
                listen 2181 tcp;
                proxy_responses 1;
                proxy_timeout 20s;
                proxy_pass zk_server;
        }
}
```

四层七层都有用，但是七层得保证访问协议是七层协议

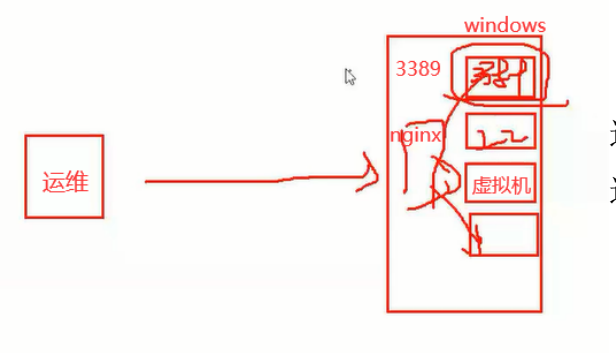
方法二：防火墙实来实现　--后续会讲

但是用防火墙比这个麻烦多了

因为需要很多规则

公网与内网的地址转换用防火墙比较好（snat dnat　源地址　目标地址）

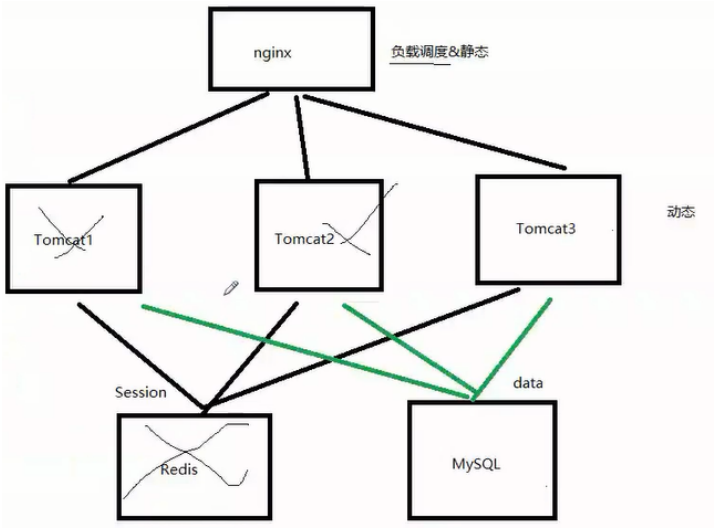
效率上四层更好，七层需要匹配URL协议啥的



这种情况只能是四层转发

这与http啥的没关系

客户反馈：网站访问突然变慢了（可以访问）



公司：集群架构

排查思路：nginx ---> tomcat -----> mysql

定位问题：

一：用户角度

一个用户（网络，DNS，CDN（缓存））

所有用户

在群里互换一下大家去访问你们公司网站

二：服务器角度

静态服务响应

/usr/local/nginx/html/a.html

负载（服务器离线，Tomcat掉了几台机器）

动态服务响应

/usr/local/nginx/html/b.jsp

top(系统负载 CPU 内存)

数据库

top(系统负载 CPU 内存)

mysql进程占用资源过多

mysql-> show proccesslist

浏览器 F12

总结：开发提交的代码中，有部分代码有逻辑问题，导致mysql产生了慢查询