# 一、Nginx配置WS

|  |
| --- |
| map $http\_upgrade $connection\_upgrade {  default upgrade;  '' close;  }  upstream wsbackend{  server ip1:port1;  server ip2:port2;    keepalive 1000;  }  server {  listen 20038;  location /{  proxy\_http\_version 1.1;  proxy\_pass http://wsbackend;  proxy\_redirect off;  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_read\_timeout 3600s;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection $connection\_upgrade;  }  } |

**分别分析上述配置的具体含义**

|  |
| --- |
| map $http\_upgrade $connection\_upgrade {  default upgrade;  '' close;  } |

表示的是：

* 如果$http\_upgrade 不为 '' (空)， 则$connection\_upgrade 为 upgrade 。
* 如果$http\_upgrade 为 '' (空)，  则 $connection\_upgrade 为 close。

表示的是监听的服务器的配置

listen 20038 表示 nginx 监听的端口

locations / 表示监听的路径(/表示所有路径，通用匹配，相当于default)

proxt\_http\_version 1.1 表示反向代理发送的HTTP协议的版本是1.1，HTTP1.1支持长连接

proxy\_pass http://wsbackend; 表示反向代理的uri，这里可以使用负载均衡变量

proxy\_redirect off; 表示不要替换路径，其实这里如果是/则有没有都没关系，因为default也是将路径替换到proxy\_pass的后边

proxy\_set\_header Host $host; 表示传递时请求头不变， $host是nginx内置变量，表示的是当前的请求头，proxy\_set\_header表示设置请求头

proxy\_set\_header X-Real-IP $remote\_addr; 表示传递时来源的ip还是现在的客户端的ip

proxy\_read\_timeout 3600s；表的两次请求之间的间隔超过 3600s 后才关闭这个连接，默认的60s，自动关闭的元凶

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; 表示X-Forwarded-For头不发生改变

proxy\_set\_header Upgrade $http\_upgrade; 表示设置Upgrade不变

proxy\_set\_header Connection $connection\_upgrade; 表示如果 $http\_upgrade为upgrade，则请求为upgrade(websocket)，如果不是，就关闭连接

此时，访问 ws://localhost:20038 就会被转发到 ip1:port1 和 ip2:port2 上。

<https://www.teqng.com/2021/03/15/%E4%B8%80%E6%96%87%E5%B8%A6%E4%BD%A0%E6%90%9E%E6%87%82nginx%E5%A6%82%E4%BD%95%E9%85%8D%E7%BD%AEhttp%E3%80%81https%E3%80%81ws%E3%80%81wss%EF%BC%81/>

<https://www.cnblogs.com/faberbeta/p/nginx007.html>

# 二、Nginx 反向代理重写 URL

**需求**

nginx 服务器代理前端页面，并且反代后端服务器。开发时使用没有什么问题，部署后存在同样请求根地址的情况（当然可以针对一级后端地址写多个配置，但是我懒……）。想起来之前查过这个重写的方案。

**一般反向代理**

一般会定义一个统一前缀，比如：api，则配置如下

|  |
| --- |
| server {  listen 80;  server\_name default;    location /api/ {  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-NginX-Proxy true;  proxy\_pass http://example.com;  }  } |

则请求到 /api/exampleapi 时，会转发到 http://example.com/api/exampleapi。

设置proxy\_pass即可。请求只会替换域名。

但我的情况是本来打算前后端放在一起，所以 server 那边并没有加 api 这个前缀。所以要将 /api/exampleapi 转发到 http://example.com/exampleapi，则可按照如下两种配置。

方案一 在 proxy\_pass 后增加 / 则 nginx 会将 /api 之后的内容拼接到 proxy\_pass 之后。

|  |
| --- |
| server {  listen 80;  server\_name default;    location /api/ {  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-NginX-Proxy true;  proxy\_pass http://example.com/;  }  } |

方案二 使用 rewrite，注意到 proxy\_pas s结尾没有 /， rewrite 重写了 url。

|  |
| --- |
| server {  listen 80;  server\_name default;    location /api/ {  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  proxy\_set\_header X-NginX-Proxy true;  rewrite ^/api/(.\*)$ /$1 break;  proxy\_pass http://example.com;  }  } |

关于 rewrite

|  |
| --- |
| syntax: rewrite regex replacement [flag]  Default: —  Context: server, location, if |

<http://zhhb.studio/posts/Nginx-proxy_pass/>

<https://www.cnblogs.com/woshimrf/p/nginx-proxy-rewrite-url.html>

# 三、nginx实现CORS跨域配置

<https://www.imqianduan.com/nginx/cors.html>

|  |
| --- |
| **跨域处理方式** 为什么要讲简单请求和非简单请求呢？因为浏览器对对这两种请求的处理，是不一样的。  ****1.简单请求****  对于简单请求，浏览器直接发出CORS请求。具体来说，就是在头信息之中，增加一个Origin字段。  处理如下：  [复制](javascript:;)   1. add\_header 'Access-Control-Allow-Origin' '\*'; 2. add\_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE'; 3. add\_header 'Access-Control-Allow-Headers' 'Content-Type';   在nginx上经以上配置，一般能够解决简单请求的跨域处理。  CORS请求默认不发送Cookie和HTTP认证信息。但是如果要把Cookie发到服务器，要服务器同意，指定Access-Control-Allow-Credentials字段。  [复制](javascript:;)   1. add\_header 'Access-Control-Allow-Credentials' 'true';   需要注意的是，如果要发送Cookie，Access-Control-Allow-Origin就不能设为星号，必须指定明确的、与请求网页一致的域名。  ****2.非简单请求****  (1)非简单请求的CORS请求，会在正式通信之前，增加一次HTTP查询请求，称为”预检”请求（preflight）， ****预捡的请求方法是OPTIONS****。浏览器先询问服务器，当前网页所在的域名是否在服务器的许可名单之中，以及可以使用哪些HTTP动词和头信息字段。  因此我们可以在浏览器的开发者工具中查看头信息，若头信息中有OPTIONS方法，说明此次CORS请求是非简单请求，需要在nginx中添加头:   1. add\_header 'Access-Control-Allow-Origin' '\*'; 2. add\_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE'; 3. add\_header 'Access-Control-Allow-Headers' 'Content-Type';   此时虽然我们已经添加了所有的Methods方法，但是预检请求仍不会通过，因此此时nginx对OPTIONS方法返回”405 Method Not Allowed”或者403。我们需要在nginx对OPTIONS方法进行处理，如下：  [复制](javascript:;)   1. if ($request\_method = 'OPTIONS') { 2. add\_header 'Access-Control-Allow-Origin' '\*'; 3. add\_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE'; 4. add\_header 'Access-Control-Allow-Headers' 'Content-Type'; 5. return 200; 6. } 7. 或 8. add\_header 'Access-Control-Allow-Origin' '\*'; 9. add\_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS, PUT, DELETE'; 10. add\_header 'Access-Control-Allow-Headers' 'Content-Type'; 11. #在location处添加以下内容 12. if ($request\_method = 'OPTIONS') { 13. return 200; 14. }   我们在nginx上将OPTIONS方法返回200，而不是405或403。  (2)服务器经过以上设置通过了”预检”请求，以后每次浏览器正常的CORS请求，就都跟简单请求一样，会有一个Origin头信息字段。 |

**四、在原有源码安装NGINX上再yum 安装NGINX**

4.1先停掉源码安装NGINX

|  |
| --- |
| *#service nginx stop*  *#/usr/local/nginx/sbin/nginx -s stop* |

4.2部署yum nginx

|  |
| --- |
| *#rpm -Uvh <http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm>*  *#yum install -y nginx*  *#vim /etc/nginx/conf.d/default.conf (将端口修改为非80端口，或者把default.conf移到其他地方)*  *#service nginx start* |

4.3 源码安装只能以下方式启停

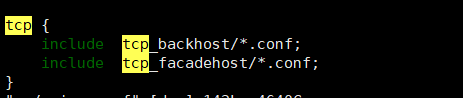
|  |
| --- |
| *#/usr/local/nginx/sbin/nginx*  *#/usr/local/nginx/sbin/nginx -s stop*  *#/usr/local/nginx/sbin/nginx -t*  *#/usr/local/nginx/sbin/nginx -s reload* |

1. **配置TCP代理**

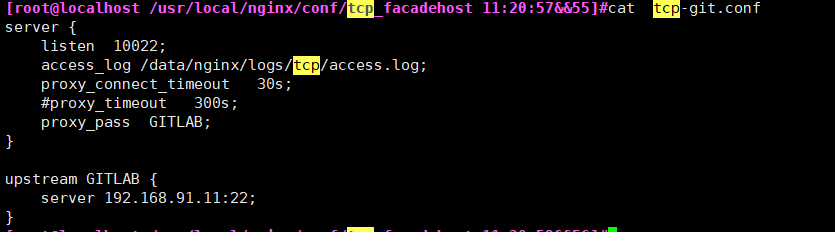
首先要确定是否加载了nginx\_tcp\_proxy\_module模块。

由于在主配置文件nginx.conf指定了tcp配置文件目录。所以直接在/usr/local/nginx/conf/tcp\_facadehost、/usr/local/nginx/conf/tcp\_backhost/配置相关信息。

nginx.conf配置信息



/usr/local/nginx/conf/tcp\_facadehost 配置信息



|  |
| --- |
| server {  listen 10022;  access\_log /data/nginx/logs/tcp/access.log;  proxy\_connect\_timeout 30s;  #proxy\_timeout 300s;  proxy\_pass GITLAB;  }  upstream GITLAB {  server 192.168.91.11:22;  } |

1. **限流**

*<https://chenyongjun.vip/articles/81>*

*<https://segmentfault.com/a/1190000023150705>*

*<https://zhaox.github.io/nginx/2019/09/05/explain-nginx-rate-limit>*

*<https://www.yinxiang.com/everhub/note/8b1b9b26-6ba7-4b2c-be59-045442b27eba>*

1. **配置https**

|  |
| --- |
| server {  listen 80;  server\_name www.itdance.cn;  rewrite ^(.\*)$ https://www.itdance.cn$1 permanent;  }  server {  listen 443 ssl; #必须加ssl，否则访问不了。  server\_name www.itdance.cn; # 域名  access\_log /data/nginx/logs/itdance/access.log main;  charset utf-8;  #ssl on; # 启用ssl功能 ，新版已经不用，可以去掉  ssl\_certificate ssl/8052219\_itdance.cn.pem;  ssl\_certificate\_key ssl/8052219\_itdance.cn.key;  ssl\_session\_timeout 5m; # 客户端可以重用会话参数的时间  ssl\_protocols TLSv1 TLSv1.1 TLSv1.2; # 使用的协议  ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE; # 配置加密套件  ssl\_prefer\_server\_ciphers on;  set $flag 0;  if ($host ~\* "\d+\.\d+\.\d+\.\d+"){  set $flag "${flag}1";  }    if ($request\_uri = "/tclearcache"){  set $flag "${flag}2";  }    if ($flag = "01"){  return 444;  }  #if ($host !~ "(pg|yldc)\.\*"){  # return 444;  #}  root html;  index index.htm index.html index.jsp;  trim on;  trim\_js on;  trim\_css on;  location / {  if ( $http\_destination ~\* ^https(.\*)$ ) {  set $fixed\_destination http$1;  }  proxy\_next\_upstream http\_502 http\_504 http\_503 http\_500 error timeout invalid\_header;  session\_sticky\_hide\_cookie upstream=itdance;  proxy\_pass\_header Set-Cookie;  proxy\_pass http://itdance;  }  location ~ .\*\.(htm|html|js|css|gif|jpeg|bmp|ico|swf|flv)$ {  proxy\_next\_upstream http\_500 http\_502 http\_503 http\_504 error timeout invalid\_header;  session\_sticky\_hide\_cookie upstream=itdance;  proxy\_pass http://itdance;  expires 30m;  }  } |

*<https://www.php.cn/nginx/421959.html>*

*<http://www.zuidaima.com/blog/4337284528704512.htm>*