

18 反向代理的配置和实例

更新时间：2020-02-06 18:42:01



“

构成我们学习最大障碍的是已知的东西，而不是未知的东西。—— 贝尔纳

”

前言

我们在上一篇文章中介绍了 [代理](#) 的概念，包括什么是 [正向代理](#)，什么是 [反向代理](#)。本文我们就在前面的基础之上学习一下如何配置反向代理。

其实 [反向代理](#) 一般情况和 [负载均衡](#) 配合使用，我们会在后面介绍 [负载均衡](#)。

反向代理模块介绍

[nginx](#) 的 [反向代理](#) 功能主要是由 [ngx_http_proxy_module](#) 模块实现的。这个模块有很多指令，详细的指令可以参考[Nginx文档](#)。

[Example Configuration](#)

[Directives](#)

[proxy_bind](#)

[proxy_buffer_size](#)

[proxy_buffering](#)

[proxy_buffers](#)

[proxy_busy_buffers_size](#)

[proxy_cache_path](#)

[proxy_cacne](#)
[proxy_cache background update](#)
[proxy_cache bypass](#)
[proxy_cache convert head](#)
[proxy_cache key](#)
[proxy_cache lock](#)
[proxy_cache lock age](#)
[proxy_cache lock timeout](#)
[proxy_cache max range offset](#)
[proxy_cache methods](#)
[proxy_cache min uses](#)
[proxy_cache path](#)
[proxy_cache purge](#)
[proxy_cache revalidate](#)
[proxy_cache use stale](#)
[proxy_cache valid](#)
[proxy_connect timeout](#)
[proxy_cookie domain](#)
[proxy_cookie path](#)
[proxy_force_ranges](#)
[proxy_headers hash bucket size](#)
[proxy_headers hash max size](#)
[proxy_hide_header](#)
[proxy_http version](#)

有很多指令

工作中我们经常用到的只有几个，我们熟悉这些就行了，下面就是一些线上服务器的配置：

```
proxy_connect_timeout 15s;  
proxy_read_timeout 24s;  
proxy_send_timeout 10s;  
proxy_buffer_size 64k;  
proxy_buffers 8 64k;  
proxy_busy_buffers_size 128k;  
proxy_temp_file_write_size 128k;
```

`proxy_pass` 指令

其实 `proxy_pass` 的用法很多，这个指令可以把特定的请求 [反向代理](#) 到一个 [服务器组](#) (这里牵涉到负载均衡，我们在后面的文章中会介绍)，也可以代理到一个 [IP](#)，一个 [URL](#) 等。

这个指令是 `ngx_http_proxy_module` 模块的核心指令，它实现了反向代理的功能。

我们这里只介绍这一个指令，通过这个指令，配合例子，我们就基本可以抓住 反向代理 的核心了。

实例

我们会使用一个非常非常简单的例子，这个例子主要是为了帮助大家进一步理解 反向代理 的含义，加深这个概念。

反向代理是 Nginx 非常重要的功能，我们无论怎么强调都不为过。

为了实现反向代理，我们需要在机器上面启动两个 nginx 进程，一个作为 前端机 (也即是接收客户端请求的服务器)。一个作为 反向代理 机，也即是实际完成工作的服务器。这也是我们实际工作中经常用到的模式。

首先，我们创建一个 nginx_proxy.conf 文件，作为 反向代理 机的配置文件。

```
[root@3a878d4dc393 html]# cat -n /usr/local/nginx/conf/nginx_proxy.conf
 1
 2 worker_processes 1;
 3
 4
 5 events {
 6     worker_connections 1024;
 7 }
 8
 9
10 http {
11     include mime.types;
12     default_type application/octet-stream;
13
14     server {
15         listen 8088;
16         server_name www.proxy.com;
17         access_log /usr/local/nginx/logs/proxy_access.log;
18
19         location / {
20             root proxy;
21             index index.html index.htm;
22         }
23     }
24 }
25
26 }
27
28
29
```

反向代理配置

然后配置一个 前端机：

```
[root@3a878d4dc393 html]# cat -n /usr/local/nginx/conf/nginx.conf
```

```
1
2 worker_processes 1;
3
4
5 events {
6     worker_connections 1024;
7 }
8
9
10 http {
11     include mime.types;
12     default_type application/octet-stream;
13
14     server {
15         listen 80;
16         server_name localhost;
17         access_log /usr/local/nginx/logs/access.log;
18         error_log /usr/local/nginx/logs/error.log;
19
20         location / {
21             proxy_pass http://www.proxy.com:8088/;
22         }
23     }
24 }
25
26 }
27
28
```

将所有的请求都代理到代理机器上面

然后启动两个 nginx 进程：

```
/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx_proxy.conf
/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
```

我们查看 nginx 进程：

```
[root@3a878d4dc393 html]# ps -ef | grep nginx
root      238      0  0 10:53 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.c
onf
nobody    244    238  0 10:53 ?        00:00:00 nginx: worker process
root      248      0  0 10:54 ?        00:00:00 nginx: master process /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx_p
roxy.conf
nobody    249    248  0 10:54 ?        00:00:00 nginx: worker process
```

我们分别打开 **前端机** 和 **反向代理机** 的访问日志，然后请求

`curl http://localhost/hello.html`，如下：

```
[root@3a878d4dc393 html]# curl -i http://localhost/hello.html
HTTP/1.1 200 OK
Server: nginx/1.16.1
Date: Tue, 04 Feb 2020 11:06:20 GMT
Content-Type: text/html
Content-Length: 354
Connection: keep-alive
Last-Modified: Tue, 04 Feb 2020 10:31:33 GMT
ETag: "5e394805-162"
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Title Page</title>
  </head>
  <body>
    <h1 class="text-center">Hello World From Proxy</h1>
  </body>
</html>
```

反向代理机的结果

我们可以从日志中看到这个过程：

前端机的日志	反向代理机的日志
<pre>[root@3a878d4dc393 logs]# [root@3a878d4dc393 logs]# tail -f proxy_access.log 127.0.0.1 - - [04/Feb/2020:11:06:20 +0000] "GET /hello.html HTTP /1.0" 200 354 "-" "curl/7.29.0"</pre>	<pre>[root@3a878d4dc393 logs]# [root@3a878d4dc393 logs]# tail -f access.log 127.0.0.1 - - [04/Feb/2020:11:06:20 +0000] "GET /hello.html HTTP/ 1.1" 200 354 "-" "curl/7.29.0"</pre>

其实从日志中我们可以看到，我们请求了 **前端机**，但是 **前端机** 把请求转发给了 **反向代理机**，而后者才是真正处理请求的机器。

总结

我们在本文中通过大量的图片和一个实例，配和上一篇文章，让大家加深对 **反向代理** 的理解。

其实 **反向代理** 和 **负载均衡** 是非常亲密的关系，大家在后面的学习中会深刻的理解到这一点。

}