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下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_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
```

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：


```
[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"
```

前端机的日志

```
[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"
```

反向代理机的日志

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。

}

17 什么是正向代理？什么又是反向代理？

19 何为负载均衡：人多力量大