Linux服务器构建与运维管理

第04章：代理服务器

阮晓龙

13938213680 / ruanxiaolong@hactcm.edu.cn

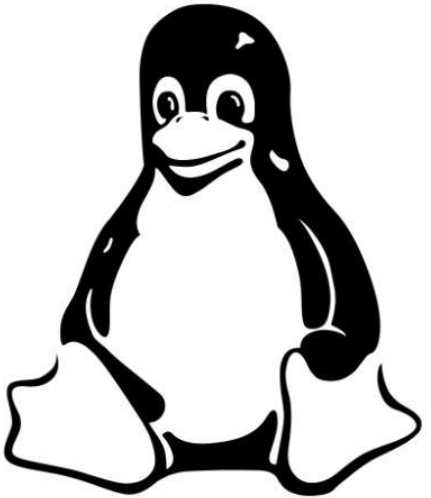
http://linux.xg.hactcm.edu.cn
http://www.51xueweb.cn

河南中医药大学信息管理与信息系统教研室
信息技术学院网络与信息系统科研工作室
河南中医药大学医疗健康信息工程技术研究所

2022.9

提纲

- 代理服务
- 使用Nginx实现反向代理
  - Nginx Open Source 与 NGINX Plus
  - 安装部署Nginx服务
  - 使用Nginx实现反向代理
  - Nginx配置文件与日志
- 使用Nginx实现负载均衡
  - 使用Nginx实现网站负载均衡
  - 提升Nginx的安全性
- 使用Apache实现Web负载均衡
  - 认识Apache Proxy
  - 使用Apache Proxy实现网站负载均衡

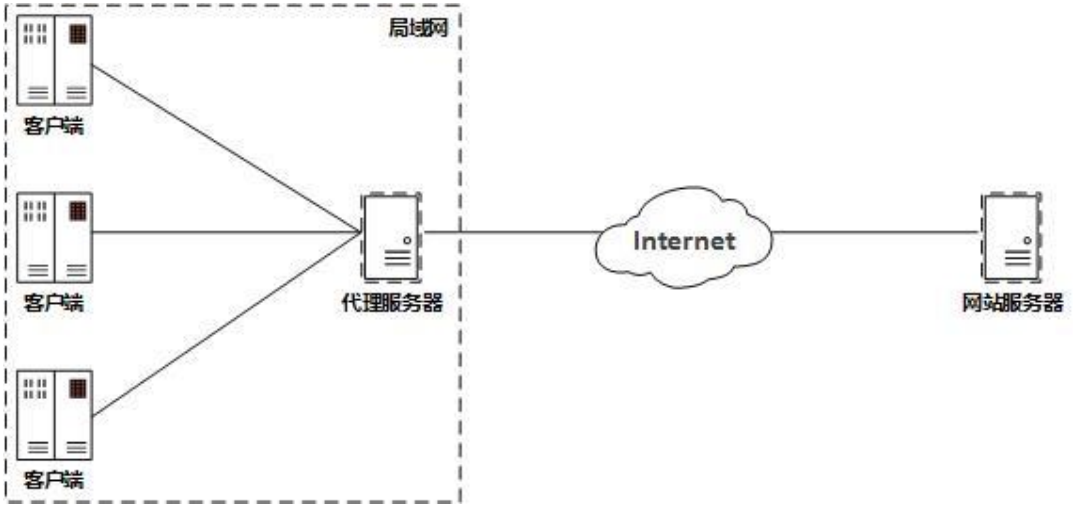


1.代理服务

- 代理服务是连接互联网与局域网的重要安全措施。
- 代理服务可以实现互联网与局域网之间通信，分为正向代理和反向代理两种。
- 正向代理（Forward Proxy，转发代理）
  - 当客户端无法访问外部资源时，可以通过正向代理间接访问。
  - 正向代理服务器是位于客户端与互联网上的网站服务器之间的服务器。
  - 为了从互联网上的网站服务器获取内容，客户端发送请求到正向代理服务器，然后正向代理服务器从互联网上的网站服务器中获取内容并返回给客户端。
  - 客户端必须专门配置正向代理服务器，如在浏览器中配置代理服务器等。

1.代理服务

- 代理服务可以实现互联网与局域网之间通信，分为正向代理和反向代理两种。
- 正向代理（Forward Proxy，转发代理）



1.代理服务

- 正向代理服务器的工作步骤
  - 客户端计算机向代理服务器发出访问互联网的请求。
  - 代理服务器接收客户端请求后，会检查请求的来源地址和目的地址，如果两者都能满足访问规则要求，那么代理服务器将继续进行下一步的处理，否则将拒绝客户端的请求。
  - 代理服务器会先查找本地缓存，如果缓存中存在客户端请求的数据，则把数据直接返回给客户端并结束此次处理；否则将进行下一步。
  - 如果代理服务器在缓存中没有找到客户端所请求的数据，那么代理服务器会代替客户端向互联网上的相应服务器发出请求。
  - 互联网上的服务器返回代理服务器所请求的数据，在接收到返回的数据后，代理服务器会把数据复制一份到缓存中。
  - 代理服务器把数据返回给客户端，并结束本次处理。

1.代理服务

- 正向代理服务器的用途
  - 提高速度
    - 代理服务器接收远程服务器提供的数据并进行缓存，如果有许多用户同时使用相同代理服务器，他们对Internet站点所有的请求都会经由这台代理服务器，当有人访问过某一站点后，所访问的内容会被保存在代理服务器上，如果下次再有人访问，这些内容便可直接从代理服务中获取，而不必再次连接远程服务器。
    - 可以节约带宽、提高访问速度。
  - 节省IP
    - 使用代理服务器可以减少对IP地址的需求，对使用局域网方式接入Internet的方式，如果为局域网（LAN）内的每一个用户都申请一个IP地址，将造成极大的资源浪费。
    - 使用代理服务器后，只需代理服务器有一个IP地址，局域网内其他用户可以使用私有IP地址，可以节约大量的IP地址，降低网络的维护成本。

1.代理服务

- 正向代理服务器的用途
  - 防止攻击
    - 代理服务器可保护局域网的安全，起到防火墙的作用。对使用代理服务器的局域网来说，在外部看来只有代理服务器是可见的，其他局域网内的用户对外是不可见的，代理服务器为局域网的安全起到了屏障作用。
    - 通过代理服务器，用户可设置IP地址过滤，限制内部网对外部的访问权限。代理服务器也可通过限制封锁IP地址的方式，禁止用户对某些网页的访问。
  - 用户管理
    - 通过代理服务器，可设置用户验证功能，对用户行为进行记录。没有经过授权认证的用户无权通过代理服务器访问互联网。
    - 所有用户对互联网的访问均通过代理服务器进行，可记录局域网用户对互联网访问的时间、内容、流量等信息，基于此信息可进行一定的数据分析统计。

1.代理服务

- 正向代理服务器的用途
  - 突破限制
    - 代理服务器可突破网络限制，比如局域网对上网用户的访问端口、目的网站、协议等的限制，都可以突破。
  - 隐藏身份
    - 使用代理服务器会隐藏黑客的身份，黑客进行攻击时常常采用代理服务器的方式，并以此为跳板，对局域网内机器进行渗透。

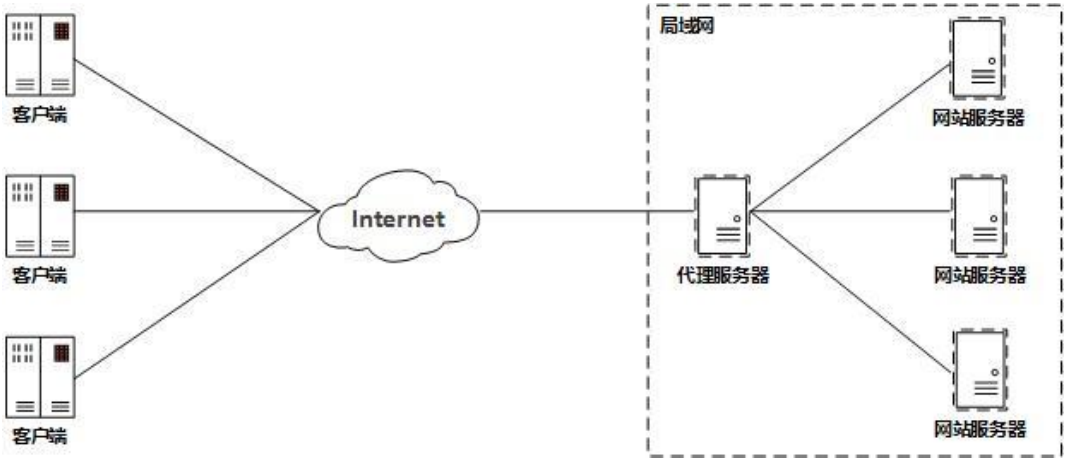
正向代理服务器与NAT功能相似，但两者有本质的区别。

1.代理服务

- 代理服务可以实现互联网与局域网之间通信，分为正向代理和反向代理两种。
- 反向代理（reverse proxy）
  - 反向代理与正向代理相反，在客户端看来它就像是一个普通的网站服务器，客户端不需做任何配置。
  - 客户端发送请求到代理服务器，代理服务器决定将这些请求发往何处。

1.代理服务

- 代理服务可以实现互联网与局域网之间通信，分为正向代理和反向代理两种。
- 反向代理（reverse proxy）

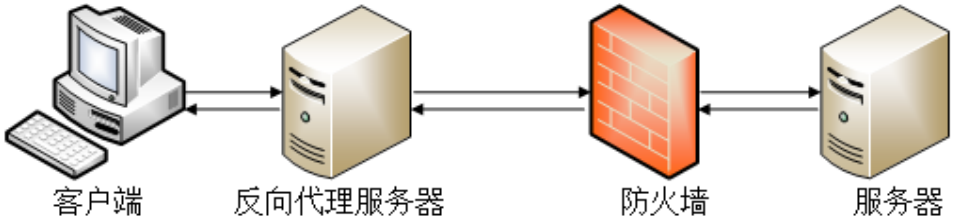


1.代理服务

- 转发代理与反向代理的对比
  - 相同点：
    - 所处位置都是在客户端与网站服务器之间。
    - 都是用户和服务器之间的中介，完成用户请求和结果的转发。
  - 不同点：
    - 正向代理是客户端的代理，反向代理是服务器的代理。
    - 正向代理一般是为客户端架设的，反向代理一般是为服务器架设的。
    - 正向代理中网站服务器无法获知客户端的真正地址，反向代理中客户端无法获知网站服务器的真正地址。
    - 正向代理主要是解决内部访问外部网络受到限制的问题，反向代理主要是提供更为安全稳定的网站服务。

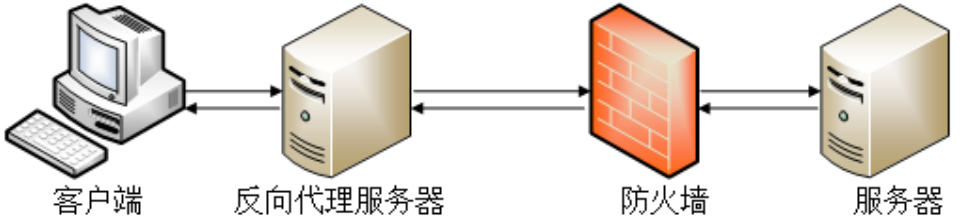
1.代理服务

- 反向代理服务器（reverse proxy）
  - 在客户端来看，反向代理服务器就像一个普通的Web服务器，客户端不要做任何特殊的配置即可使用。
  - 客户端发送普通的请求来获取反向代理所属空间的内容；反向代理决定将这些请求发往何处，然后就好像它本身就是原始服务器一样将请求内容返回。



1.代理服务

- 反向代理服务器（reverse proxy）
  - 反向代理服务器的一个典型应用就是代理处于防火墙后的服务器所支撑的服务，使外部用户可访问。
  - 反向代理能够用于为多个后端服务器提供负载均衡，或者为相对较慢的后端服务器提供缓存。
  - 反向代理还能够简单地将多个服务器映射到同一个URL上。



1.代理服务

- 反向代理服务器的作用
  - 隐藏服务器真实IP，客户端只能看到代理服务器地址。
  - 实现业务负载均衡，代理服务器可根据网站服务器的负载情况，将客户端请求分发到不同网站服务器。
  - 提高业务访问速度，代理服务器提供缓存服务，提高网站等业务的访问速度。
  - 提供安全保障，代理服务器可作为应用层防火墙，为网站提供防护。

1.代理服务

- 反向代理服务器可以提高Web服务器的交付安全，采用反向代理的方式可将真实的数据服务器隐藏起来的。
  - 当采用反向代理的模式进行应用部署时，真实的数据服务器与用户之间由反向代理服务器分割，将真实的数据服务器隐藏。
  - 当用户访问具体的业务时，需要先通过反向代理服务器，同样的当业务遭受攻击时，也会通过反向代理服务器。
  - 反向代理服务器可通过地址保护、访问范围限制等方式进行攻击拦截。

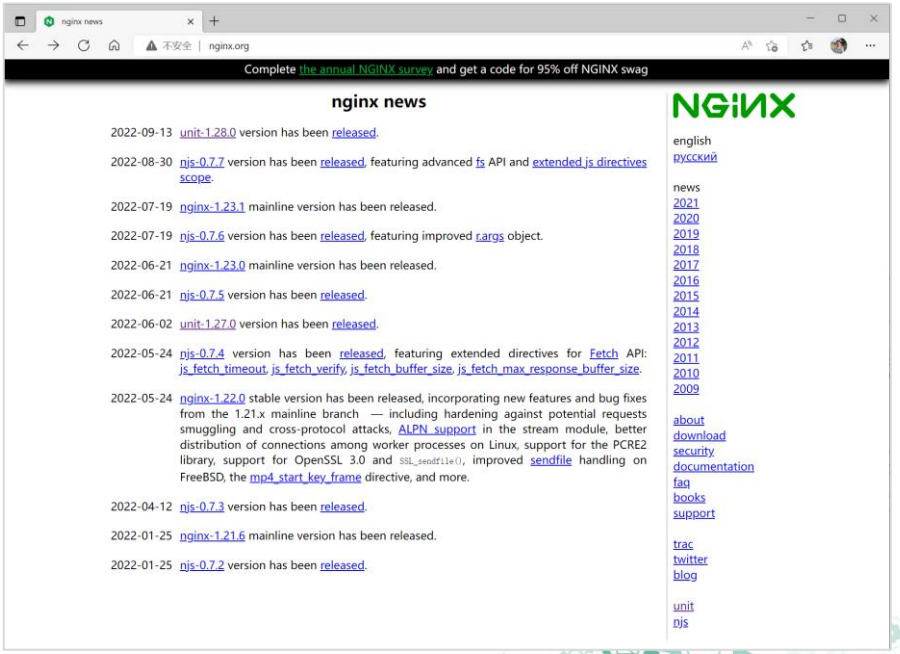
1.代理服务

- 反向代理服务器可以实现负载均衡。
  - 反向代理服务器可以将请求转发给内部的Web服务器，基于此原理可以让代理服务器将请求均匀的转发给多台的Web服务器上，达到负载均衡。
  - 使用反向代理进行负载均衡的实现时还可以将负载均衡与代理服务器的高速缓存技术结合起来，提高性能和安全性。

- 反向代理服务器可以提升信息安全。
  - 反向代理服务器在访问业务时进行了请求转发，在请求转发的过程中，可以配置过滤规则，利用反向代理服务器有效提升信息安全。
  - 通过设置规则实现防范DDOS攻击、资源盗链、SQL注入等威胁。

2.使用Nginx实现反向代理

- Nginx是开源的轻量级网站服务器软件，是高性能的HTTP和反向代理服务器软件，同时也是IMAP/POP3/SMTP协议的代理服务器软件。
  - Nginx官网地址为：http://nginx.org
  - 使用版本为NGINX Open Source 1.28.0



2.使用Nginx实现反向代理

- Nginx的主要特性
  - 基于模块化的结构
  - 基于EPOLL事件驱动模型
  - 提供反向代理服务，可使用缓存加速反向代理，支持简单负载均衡和容错
  - 支持基于文件的配置
  - 支持基于IP和域名的虚拟网站配置
  - 支持SSL和TLS SNI
  - 支持视频流式服务
  - 支持嵌入Perl语言
  - 支持FastCGI、Uwsgi、SCGI
  - 支持IMAP、POP3、SMTP代理

2.使用Nginx实现反向代理

- Nginx是由内核和模块组成的，内核主要通过查找配置文件将客户端请求映射到location block，然后通过location block配置的指令启动不同的模块完成相应的工作。
- Nginx的模块从结构上分为以下三种。
  - 核心模块：
    - HTTP模块、EVENT模块、MAIL模块 … …
  - 基础模块：
    - HTTP Access模块、HTTP FastCGI模块、HTTP Proxy模块
    - HTTP Rewrite模块 … …
  - 第三方模块：
    - HTTP Upstream Request Hash模块、Notice模块、HTTP Access Key模块
    - … …

2.使用Nginx实现反向代理

- Nginx的模块从功能上分为以下四种。
  - Core（核心模块）：构建Nginx基础服务，管理其它模块。
  - Handlers（处理器模块）：此类模块直接处理请求，进行输出内容和修改headers信息等操作。
  - Filters（过滤器模块）：此类模块主要对其它处理器模块输出的内容进行修改操作，最后由Nginx输出。
  - Proxies（代理类模块）：此类模块是Nginx的HTTP Upstream之类的模块，这些模块主要与后端一些服务（比如FastCGI等）进行交互，实现服务代理和负载均衡等功能。

- Nginx的核心模块主要负责建立Nginx服务模型、管理网络层和应用层协议以及启动针对特定应用的一系列模块。
- 其他模块负责网站服务器的实际工作，当Nginx发送文件或转发请求到其它服务器时，由Handlers、Proxies模块提供服务，当需要Nginx把输出压缩或者增加一些数据时，由Filters模块提供服务。

2.使用Nginx实现反向代理

- Nginx进程模型
  - Nginx默认采用多进程工作方式。
  - Nginx启动后会运行一个主进程和多个子进程。
    - 主进程充当整个进程组与用户的交互接口，对进程进行监护，管理子进程来实现服务重启、平滑升级、配置文件实时生效等功能。
    - 子进程用来处理来自客户端的请求。

2.使用Nginx实现反向代理

任务1：安装Nginx

任务2：使用Nginx实现反向代理

2.使用Nginx实现反向代理

任务1：安装Nginx

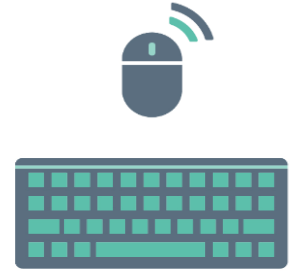
- 步骤1：创建虚拟机并完成CentOS的安装
- 步骤2：完成虚拟机的主机配置、网络配置及通信测试
- 步骤3：通过在线方式安装Nginx
- 步骤4：启动Nginx服务
- 步骤5：查看Nginx运行信息
- 步骤6：配置nginx服务为开机自启动

表 4-1-1 虚拟机与操作系统配置

<table>
<tr><th>虚拟机配置</th><th>操作系统配置</th></tr>
<tr><td rowspan="3">虚拟机名称：
VM-Project-04-Task-01-10.10.2.106
内存：1024MB
CPU：1 颗 1 核心
虚拟硬盘：10GB
网卡：2 块
网卡 1 桥接，网卡 2 内部网络</td><td>主机名：Project-04-Task-01</td></tr>
<tr><td>网络接口：ens192
IP 地址：10.10.2.106
子网掩码：255.255.255.0
网关：10.10.2.1
DNS：8.8.8.8</td></tr>
<tr><td>网络接口：ens224
IP 地址：172.16.0.254
子网掩码：255.255.255.0
网关：不配置
DNS：不配置</td></tr>
</table>

操作视频 / 现场演示

- 任务1：安装Nginx
  - 任务目标
    - 完成Nginx安装
    - 本地主机能够访问Nginx



命令指南 / 操作引导

```
#使用yum工具安装Nginx
[root@Project-04-Task-01 ~]# yum install -y nginx

#使用systemctl start命令启动nginx服务
[root@Project-04-Task-01 ~]# systemctl start nginx

#使用systemctl status查看Nginx服务运行状态
[root@Project-04-Task-01 ~]# systemctl status nginx

#使用systemctl enable命令可设置nginx服务为开机自启动
[root@Project-04-Task-01 ~]# systemctl enable nginx

#使用systemctl list-unit-files命令确认nginx服务是否已配置为开机自启动
[root@Project-04-Task-01 ~]# systemctl list-unit-files | grep nginx.service
```

2.使用Nginx实现反向代理

任务2：使用Nginx实现反向代理

2.使用Nginx实现反向代理

任务2：使用Nginx实现反向代理

表 4-2-1 服务器规划表

虚拟机名称	业务名称	作用
VM-Project-04-Task-01-10.10.2.106	代理服务器-Nginx	实现网站代理
VM-Project-04-Task-02-172.16.0.1	网站服务器-内部-1	发布内部网站业务

表 4-2-2 网站服务器-内部-网站规划表

网站名称	服务器	网站目录	访问地址	网站首页内容
Site-Clone-1	网站服务器-内部-1	/var/www/html	http://172.16.0.1	Site-Clone-1：http://172.16.0.1

2.使用Nginx实现反向代理

任务2：使用Nginx实现反向代理

- 步骤1：发布网站Site-Clone-1
- 步骤2：配置Nginx实现反向代理
- 步骤3：重新载入Nginx的配置文件
- 步骤4：验证反向代理服务

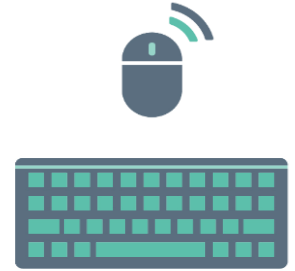
操作视频 / 现场演示

- 任务2：使用Nginx实现反向代理
  - 任务目标：
    - 本地主机访问Nginx服务器地址，访问到内部网站

命令指南 / 操作引导

```
#在网站服务器-内部-1上操作
#安装Apache并配置
[root@Project-04-Task-02 ~]# yum install –y httpd
[root@Project-04-Task-02 ~]# systemctl start httpd
[root@Project-04-Task-02 ~]# systemctl enable httpd

#创建网站并发布
[root@Project-04-Task-02 ~]# echo "<h1>Site-Clone-1:  http://172.16.0.1</h1>" > /var/www/html/index.html

#配置安全措施
[root@Project-04-Task-02 ~]# systemctl stop firewalld
[root@Project-04-Task-02 ~]# setenforce 0
```

```
#在Nginx服务器上操作
#使用vi工具修改Nginx配置文件
# [root@Project-04-Task-01 ~]# vi /etc/nginx/nginx.conf
#配置文件信息后附

#使用systemctl reload命令重新载入Nginx配置文件
[root@Project-04-Task-01 ~]# systemctl reload nginx
#使用systemctl stop命令关闭防火墙
[root@Project-04-Task-01 ~]# systemctl stop firewalld
#使用setenforce命令将SELinux设置为permissive模式
[root@Project-04-Task-01 ~]# setenforce 0
```



命令指南 / 操作引导

1. 配置文件：/etc/nginx/nginx.conf

```
#nginx.conf配置文件内容较多，本部分仅显示与反向代理配置有关的内容
server {
    #侦听端口为80
    listen       80 default_server;
    listen       [::]:80 default_server;
    #下述server_name未配置，Nginx默认定义请求识别路径为"_"
    server_name  _;
    #默认网站根路径为/usr/share/nginx/html
    root         /usr/share/nginx/html;
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
    #根路径请求设置
    location / {
        #将所有请求转发到http://172.16.0.1:80
        proxy_pass http://172.16.0.1:80;
    }
    #定义404错误提示页面
    error_page 404 /404.html;
        location = /40x.html {
    }
    #定义500、502、503、504错误提示页面
    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

2.使用Nginx实现反向代理

- Nginx配置文件
  - 在CentOS中，Nginx的配置文件存放位置是/etc/nginx目录。
  - 主要目录结构如下：
    - /etc/nginx/nginx.conf是主配置文件。
    - /etc/nginx/conf.d、/etc/nginx/default.d是扩展配置文件目录，如果不想频繁修改主配置文件nginx.conf，可以在此将配置独立出来。
    - /etc/nginx/fastcgi_params、/etc/nginx/scgi_params、/etc/nginx/uwsgi_params分别是fastcgi、scgi、uwsgi的配置文件目录。
    - /etc/nginx/mime.types是定义HTTP协议的Content-Type类型值。
    - /var/log/nginx用于存放日志文件。
    - /usr/share/nginx/html是默认网站的存放目录。

2.使用Nginx实现反向代理

- Nginx日志
  - Nginx日志存放在/var/log/nginx目录下，包含访问日志access.log和错误日志error.log。
  - nginx日志通过/etc/nginx/nginx.conf主配置文件进行设置，查看nginx.conf配置文件可获知日志的默认设置信息。

配置文件：/etc/nginx/nginx.conf

```
1.  #nginx.conf 配置文件内容较多，本部分仅显示与日志配置有关的内容
2.  #错误日志存放位置、记录等级等信息，默认为记录 error 等级日志
3.  error_log /var/log/nginx/error.log;
4.  #访问日志设置信息
5.  http {
6.      #定义了名为“main”的日志格式
7.      log_format   main   '$remote_addr - $remote_user [$time_local] "$request" '
8.                          '$status $body_bytes_sent "$http_referer" '
9.                          '"$http_user_agent" "$http_x_forwarded_for"';
10.     #访问日志文件存储路径为/var/log/nginx/access.log，日志记录格式为 main 定义的格式
11.     access_log   /var/log/nginx/access.log   main;
12. }
```

表 4-2-5 日志格式中常用字符串含义

变量	含义
$bytes_sent	发送给客户端的总字节数
$connection	连接的序列号
$connection_requests	当前通过一个连接获得的请求数量
$msec	日志写入时间，单位为秒，精度是毫秒
$pipe	如果请求是通过 HTTP 流水线（pipelined）发送的，值为“p”，否则为“.”
$request_length	请求长度，包括请求头和请求正文
$request_time	请求处理时间（从读入客户端的第一个字节开始，到最后一个字符发送给客户端后进行日志写入为止）单位为秒，精度是毫秒
$status	响应状态
$time_iso8601	IOS8601 标准下的服务器本地时间
$time_local	通用日志格式下的服务器本地时间

错误日志中如不设置记录级别则默认为error，不同级别日志记录详细程度不同，比如说当指定级别为error时，crit、alert、emerg信息也会被记录。

表 4-2-6 错误日志记录等级

等级	说明
emerg	紧急，系统无法使用
alert	必须立即采取措施
crit	由于配置不当导致的关键错误，危险情况的警告
error	一般错误
warn	警告信息，不算是错误信息，主要记录服务器出现的某种信息
notice	需要引起注意的情况
info	值得报告的一般消息，比如服务器重启
debug	由运行 debug 模式的程序所产生的消息

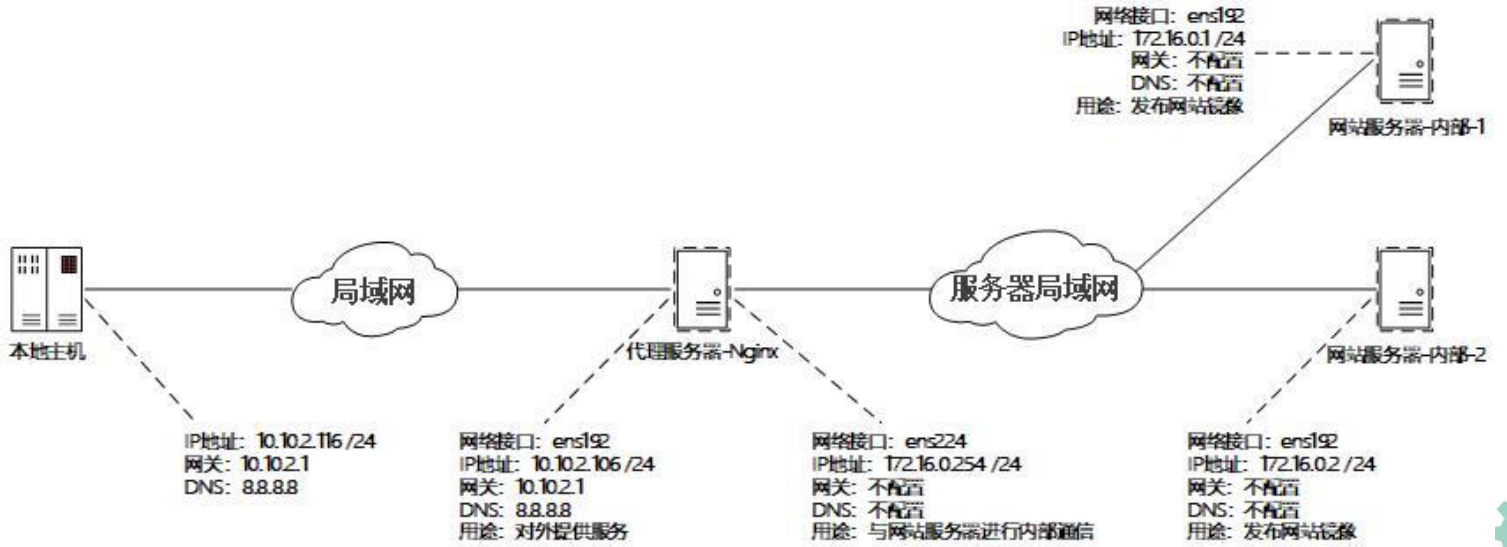
3.使用Nginx实现负载均衡

任务3：使用Nginx实现网站负载均衡

任务4：提升Nginx的安全性

3.使用Nginx实现负载均衡

任务3：使用Nginx实现网站负载均衡



3.使用Nginx实现负载均衡

任务3：使用Nginx实现网站负载均衡

表 4-3-1 服务器规划表

虚拟机名称	业务名称	作用
VM-Project-04-Task-01-10.10.2.106	代理服务器-Nginx	实现网站负载均衡
VM-Project-04-Task-02-172.16.0.1	网站服务器-内部-1	发布内部网站业务
VM-Project-04-Task-03-172.16.0.2	网站服务器-内部-2	发布内部网站业务

表 4-3-2 网站服务器-内部-网站规划表

网站名称	服务器	网站目录	访问地址	网站首页内容
Site-Clone-1	网站服务器-内部-1	/var/www/html	http://172.16.0.1	Site-Clone-1：http://172.16.0.1
Site-Clone-2	网站服务器-内部-2	/var/www/html	http://172.16.0.2	Site-Clone-2：http://172.16.0.2

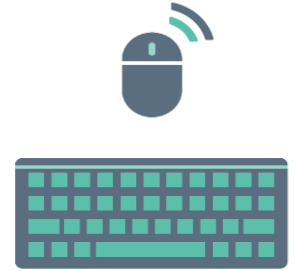
3.使用Nginx实现负载均衡

任务3：使用Nginx实现网站负载均衡

- 步骤1：发布网站Site-Clone-1
- 步骤2：发布网站Site-Clone-2
- 步骤3：配置Nginx实现负载均衡
- 步骤4：配置Nginx访问日志格式
- 步骤5：验证负载均衡服务
- 步骤6：验证负载均衡对内部业务的容灾性

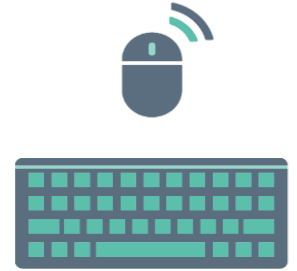
操作视频 / 现场演示

- 任务3：使用Nginx实现网站负载均衡
  - 任务目标：
    - 完成内部网站服务的部署与发布
    - 通过Nginx实现网站负载均衡
    - 完成负载均衡服务的测试



命令指南 / 操作引导

```
#在网站服务器-内部-2上操作
#安装Apache并配置
[root@Project-04-Task-03 ~]# yum install –y httpd
[root@Project-04-Task-03 ~]# systemctl start httpd
[root@Project-04-Task-03 ~]# systemctl enable httpd

#创建网站并发布
[root@Project-04-Task-03 ~]# echo "<h1>Site-Clone-2：http://172.16.0.2</h1>" > /var/www/html/index.html

#配置安全措施
[root@Project-04-Task-03 ~]# systemctl stop firewalld
[root@Project-04-Task-03 ~]# setenforce 0


#在Nginx服务器上操作
#使用vi工具修改Nginx配置文件实现负载均衡
# [root@Project-04-Task-01 ~]# vi /etc/nginx/nginx.conf
#配置文件修改内容后附

#使用vi工具修改Nginx配置文件进行访问日志格式的配置
# [root@Project-04-Task-01 ~]# vi /etc/nginx/nginx.conf
#配置文件修改内容后附
```



命令指南 / 操作引导

1. **配置文件**：/etc/nginx/nginx.conf

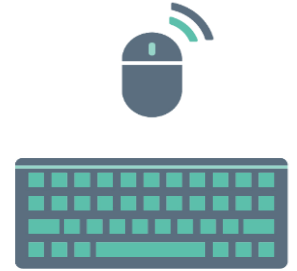
```
#nginx.conf配置文件内容较多，本部分仅显示与负载均衡配置有关的内容
#定义网站服务器组名称、网站服务器地址信息、权重信息
#权重表示将接收的请求以什么比例转发给内部服务器，下述配置表示server 172.16.0.1承担1/4请求
upstream load1{
    server 172.16.0.1:80 weight=1;
    server 172.16.0.2:80 weight=3;
}
server {
    #侦听端口为80
    listen       80 default_server;
    listen       [::]:80 default_server;
    #下述server_name未配置，Nginx默认定义请求识别路径为"_"
    server_name  _;
    #默认网站根路径为/usr/share/nginx/html
    root         /usr/share/nginx/html;
    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
    #根路径请求设置
    location / {
        #将所有请求转发到定义的网站服务器组load1中
        proxy_pass http://load1;
    }
    #定义404错误提示页面
    error_page 404 /404.html;
        location = /40x.html { }
    #定义500、502、503、504错误提示页面
    error_page 500 502 503 504 /50x.html;
        location = /50x.html { }
}
```

```
upstream load1{
    server 172.16.0.1:80 weight=1;
    server 172.16.0.2:80 weight=3;
}
```

在进行网站服务器组的server项配置时，可以使用以下参数。

- max_conns，限制到代理服务器的最大并发连接数，默认值为0表示不限制
- max_fails，与网站服务器通信尝试的失败次数，如果失败次数达到该值，则在fail_timeout时间段内，不再向其发送请求，默认为1，设为0则表示一直可用
- fail_timeout，网站服务器被设置为不可用的时间段，默认为10秒
- backup，标记为备用服务器。当主服务器不可用以后，请求会被传给备用服务器，该值不能在hash、ip_hash、random负载均衡模式下使用
- down，标记服务器永久不可用

- Nginx Plus版本支持resolve、server、route等参数配置。



命令指南 / 操作引导

1. **配置文件**：/etc/nginx/nginx.conf
2. #nginx.conf配置文件内容较多，本部分仅显示与日志配置有关的内容
3. http {
4. #定义名称为main1的日志格式
5. log_format main1
6. #服务器本地时间、客户端来源IP、请求信息
7. '[$time_local] $remote_addr $request '
8. #负载均衡转发地址信息
9. '$upstream_addr '
10. #网站服务器节点返回总耗时
11. 'ups_resp_time: $upstream_response_time '
12. #请求总耗时
13. 'request_time: $request_time';
14. #访问日志存放在/var/log/nginx/access.log目录下，日志记录格式为main1定义的格式
15. access_log /var/log/nginx/access.log main1;
16. }

验证
负载均衡服务
正确性
可用性

测试
负载均衡实现
内部业
务容灾

验证
负载均衡服务
正确性
可用性

通过两个方法验证负载均衡服务：

- 通过本地主机浏览器访问以人工验证
- 通过阅读Nginx访问日志以确认验证

测试
负载均衡实现
内部业务容灾

将虚拟机VM-Project-04-Task-03-172.16.0.2关闭

内部网站业务仅保留网站Site-Clone-1正常提供服务。

- 在本地主机打开浏览器，输入代理服务器的默认网站访问地址，多次刷新页面将只能看到网站Site-Clone-1的内容。
- 查看Nginx访问日志文件，可以看到网站Site-Clone-2关闭后，所有的请求都转发到了网站Site-Clone-1。

3.使用Nginx实现负载均衡

- Nginx负载均衡模式
  - Nginx主要通过ngx_http_upstream_module、ngx_http_proxy_module模块实现网站的负载均衡，支持轮询（round-robin）、最少连接优先（least-connected）、持续会话（ip-hash）、权重负载均衡（Weighted load balancing）等负载均衡方式。
    - 轮询（round-robin）：Nginx将客户端请求循环发送给各网站服务器节点，各网站服务器节点接收到的请求数量基本是一样的。Nginx默认为轮询模式。
    - 最少连接优先（least-connected）：Nginx将避免把请求发送到繁忙的网站服务器节点，而是将请求发送给不太繁忙的网站服务器节点。
    - 持续会话（ip-hash）：Nginx将客户端的会话一直保持在同一台网站服务器节点，直到该网站服务器节点不可用，一般用于需要维持session会话的网站业务。
    - 权重负载均衡（Weighted load balancing）：Nginx根据设置的网站服务器权重信息，将客户端请求按照权重进行分发，权重值与访问比率成正比，一般用于服务器性能不均的情况。
  - 除了上述负载均衡模式之外，Nginx还支持keepalive、least_time、random等方式。

3.使用Nginx实现负载均衡

- Nginx负载均衡健康检查
  - Nginx通过主动、被动两种方式进行参与负载均衡的各网站服务器节点的健康度检查。
  - 被动模式：
    - 当接收到一个客户端请求时，Nginx会根据设置的负载均衡方式去请求相应的网站服务器节点，如果该节点连续失败多次（由max_fails设置值决定失败次数），则Nginx将其标记为失败状态，且在一段时间（由fail_timeout设置值决定时间段）内不再向其发送请求，继续请求下一个网站服务器节点。
    - 如果定义的所有网站服务器节点都请求失败，则返回给客户端Nginx定义的错误信息。
    - 设置时间过去后，Nginx会根据客户端请求探测该网站服务器节点，如果探测成功则将其标记为存活状态。
  - 主动模式：
    - Nginx会周期性的探测各网站服务器节点，同时对探测结果进行标记。
    - 主动模式是NGINX Plus版本的独有功能，NGINX Open Source版本仅支持被动检查模式。

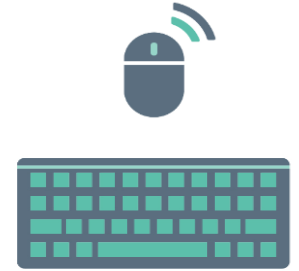
3.使用Nginx实现负载均衡

任务4：提升Nginx的安全性

- 步骤1：开启Nginx基本状态监控
- 步骤2：设置访问范围限制
- 步骤3：防DDOS攻击
- 步骤4：防SQL注入
- 步骤5：使用Linux安全措施提升代理服务安全性

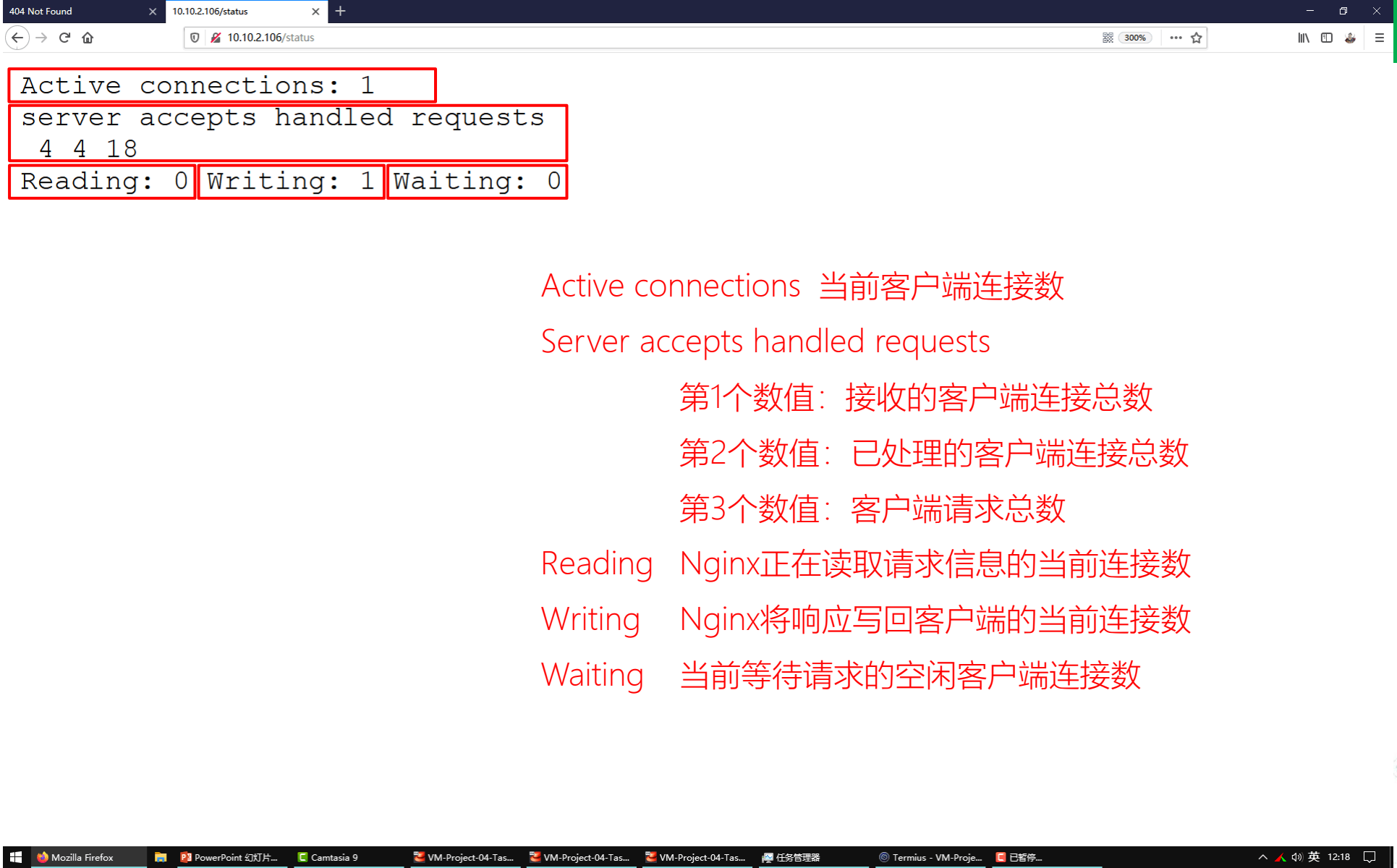
操作视频 / 现场演示

- 任务4：提升Nginx的安全性
  - 任务目标：
    - 实现Nginx的安全性配置
    - 完成安全性配置效果的测试



命令指南 / 操作引导

```
#实现Nginx的状态监控功能需要修改配置文件
#使用vi工具编辑nginx.conf配置文件。

配置文件：/etc/nginx/nginx.conf
#nginx.conf配置文件内容较多，本部分仅显示与状态页配置有关的内容
server {
    #侦听端口为80
    listen       80 default_server;
    listen       [::]:80 default_server;
    #为了排版方便此处删除了部分提示信息

    #基本状态页发布设置，/status表示访问路径，可自行定义。
    location /status {
        #展示Nginx基本状态信息
        stub_status;
    }
    #为了排版方便此处删除了部分提示信息
}

#配置完成后，重新载入配置文件使其生效。
#使用systemctl reload命令重新载入Nginx配置文件
[root@Project-04-Task-01 ~]# systemctl reload nginx
```



Active connections 当前客户端连接数

Server accepts handled requests

第1个数值：接收的客户端连接总数

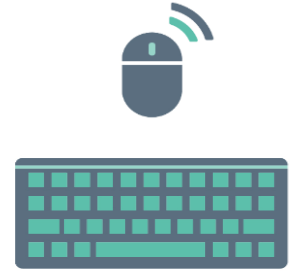
第2个数值：已处理的客户端连接总数

第3个数值：客户端请求总数

Reading Nginx正在读取请求信息的当前连接数

Writing Nginx将响应写回客户端的当前连接数

Waiting 当前等待请求的空闲客户端连接数

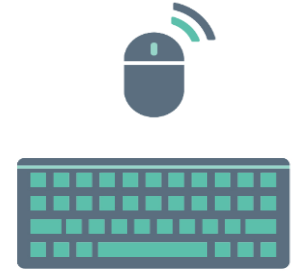


命令指南 / 操作引导

```
#通过ngx_http_access_module模块实现代理服务器的访问范围限制。
#代理服务器的访问范围设置为两条规则，具体如下。
#禁止所有地址访问
#允许IP地址在10.10.2.1/24范围内的主机访问
#使用vi工具编辑nginx.conf文件实现访问范围限制。

配置文件：/etc/nginx/nginx.conf
#nginx.conf配置文件内容较多，本部分仅显示与网站访问范围限制有关的内容
server {
    #侦听端口为80
    listen       80 default_server;
    listen       [::]:80 default_server;
    #为了排版方便此处删除了部分提示信息

    #根路径请求设置
    location / {
        #将所有请求转发到定义的网站服务器组load1中
        proxy_pass http://load1;
        #按照设计的规则设置访问范围
        allow 10.10.2.1/24;
        deny all;
    }
    #为了排版方便此处删除了部分提示信息
}

#配置完成后，重新载入配置文件使其生效。
#使用systemctl reload命令重新载入Nginx配置文件
[root@Project-04-Task-01 ~]# systemctl reload nginx
```

命令指南 / 操作引导

```
#通过ngx_http_limit_req_module、ngx_http_limit_conn_module模块
#分别限制每秒请求数、单个IP的并发请求数实现防DDOS攻击。
#限制每秒请求数设置四条规则：
#（1）限制每秒请求数为4个（2）分配10MB内存存储会话
#（3）允许超过频率限制的请求数不多于2个（4）超出的请求不做延迟处理
#限制单个IP的并发请求数为5个
使用vi工具编辑nginx.conf配置文件实现防DDOS攻击。

配置文件：/etc/nginx/nginx.conf
#nginx.conf配置文件内容较多，本部分仅显示与防DDOS攻击配置有关的内容
#定义区域名称为one，请求限制为每秒4个请求，并分配10M内存
limit_req_zone $binary_remote_addr zone=one:10m rate=4r/s;
#定义区域名称为addr，并分配10M内存
limit_conn_zone $binary_remote_addr zone=addr:10m;
server {
    #侦听端口为80
    listen       80 default_server;
    listen       [::]:80 default_server;
    #限制同一来源IP，并发请求不得超过5个
    limit_conn addr 5;
    #根路径请求设置
    location / {
        #将所有请求转发到定义的网站服务器组load1中
        proxy_pass http://load1;
        #执行one设置的限制，并设置超过频率限制的请求数不多于2个，超出的请求不延迟处理
        limit_req zone=one burst=2 nodelay;
        #按照设计的规则设置访问范围
        deny 10.10.2.116;
        allow all; }
}
#配置完成后，重新载入配置文件使其生效。
```



命令指南 / 操作引导

```
#筛选客户端敏感请求将其重定向至404页面，从而实现防SQL注入。
#使用vi工具编辑nginx.conf配置文件，编辑后的文件信息如下所示。

配置文件：/etc/nginx/nginx.conf
#nginx.conf配置文件内容较多，本部分仅显示与防SQL注入有关的内容
server {
    #侦听端口为80
    listen       80 default_server;
    listen       [::]:80 default_server;

    #如果请求连接中含有SQL特殊字符，则返回404页面
    if ($query_string ~* ".*('|--|union|insert|drop|truncate|update|from|grant|exec|where|select|and|or|count|chr|mid|like|iframe|script|alert|webscan|dbappsecurity|style|confirm|innerhtml|innertext|class).*")
    { return 404; }

    #为了排版方便此处删除了部分提示信息
}

#配置完成后，重新载入配置文件使其生效。
#使用systemctl reload命令重新载入Nginx配置文件
[root@Project-04-Task-01 ~]# systemctl reload nginx
```

命令指南 / 操作引导

```
#使用setenforce命令更改SELinux模式为enforcing
[root@Project-04-Task-01 ~]# setenforce 1
#使用sestatus查看SELinux状态信息
[root@Project-04-Task-01 ~]# sestatus
#使用setsebool -P httpd_can_network_connect命令开启httpd网络连接
[root@Project-04-Task-01 ~]# setsebool -P httpd_can_network_connect 1


#开启防火墙并验证防火墙状态
[root@Project-04-Task-01 ~]# systemctl start firewalld
[root@Project-04-Task-01 ~]# systemctl status firewalld

#在防火墙上开放TCP80端口，并重载防火墙配置使其生效
[root@Project-04-Task-01 ~]# firewall-cmd --zone=public --add-port=80/tcp –permanent
[root@Project-04-Task-01 ~]# firewall-cmd –reload
```

3.使用Nginx实现负载均衡

- NGINX Plus
  - NGINX Plus是基于NGINX Open Source构建的负载均衡、网站服务器、内容缓存软件。
  - 除了NGINX Open Source版本提供的功能外，还具备独有的企业级功能，如持久会话、通过API进行配置和运行状态健康检查等。

3.使用Nginx实现负载均衡

- NGINX Plus功能模块
  - 高性能负载均衡。
    - NGINX Open Source、NGINX Plus都可以实现基于HTTP、TCP和UDP协议的负载均衡，NGINX Plus通过企业级负载均衡扩展了NGINX Open Source，包括持久会话、活动运行状态检查、无需重启服务即可动态配置负载均衡服务器组等。
  - 大规模可扩展的内容缓存。
    - NGINX最典型的应用便是作为内容缓存，既可以加速本地资源访问速度，也可作为CDN的一部分提供服务。
    - NGINX Plus扩展了NGINX Open Source的功能，增加了清除缓存的功能，并实现了更为丰富的缓存状态可视化信息。
  - 网站服务器。
    - NGINX Plus具备同NGINX Open Source相同的网站发布模式、支持反向代理多种协议、支持HTTP流式视频服务、支持HTTP/2协议等。
  - 安全控制。
    - 支持请求和连接的限制、支持TLS 1.3协议、支持动态证书加载、支持ACL访问范围限制、支持基于API和OpenID Connect的JWT身份验证、支持NGINX WAF动态模块。

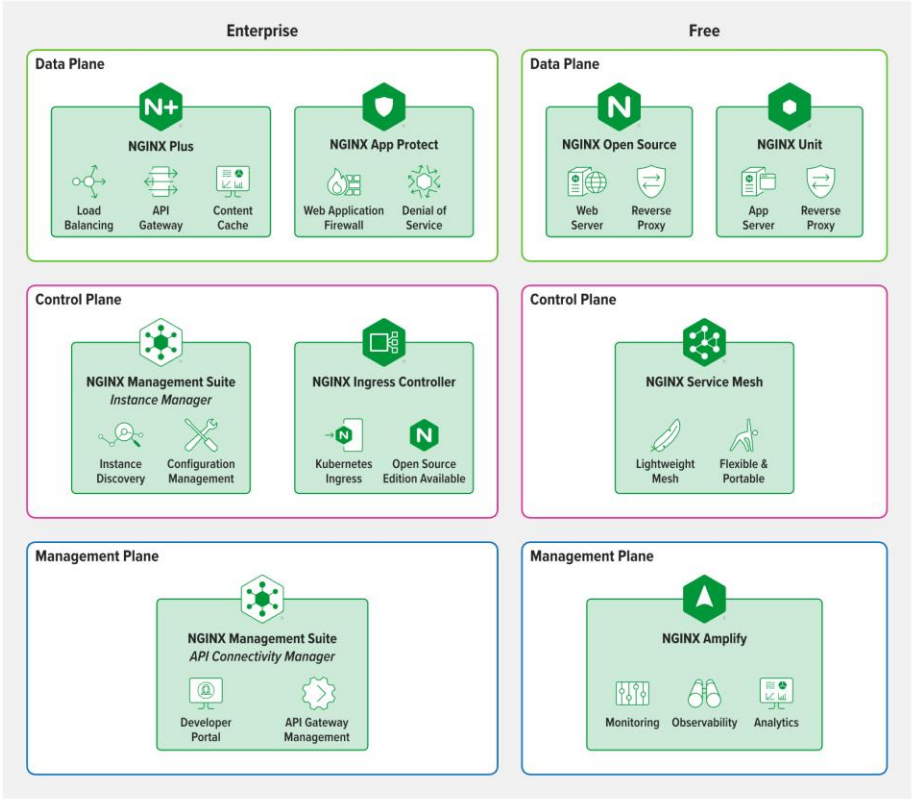
3.使用Nginx实现负载均衡

- NGINX Plus功能模块
  - 实时监控。
    - NGINX Plus包含一个实时的活动监控接口，提供关键模块的负载和性能指标，通过简单的配置即可查看到统计监控信息。
  - 流媒体。
    - NGINX Open Source被广泛的用于通过HTTP流式获取MP4和FLV视频内容，通过流式服务客户端不需下载整个资源即可得到所需内容。
    - NGINX Plus扩展了这一功能，通过Apple HLS和Adobe HDS支持视频点播应用（VOD），通过RTMP实现基于Flash的服务，并且自适应流允许视频播放器实时选择合适的比特率。

- Compare Versions
  - https://www.nginx.com/products

An Architecture For Modern Applications

F5 NGINX provides a suite of products that together form the core of what organizations need to create apps and APIs with performance, reliability, security, and scale.



NGINX > Products > NGINX Plus

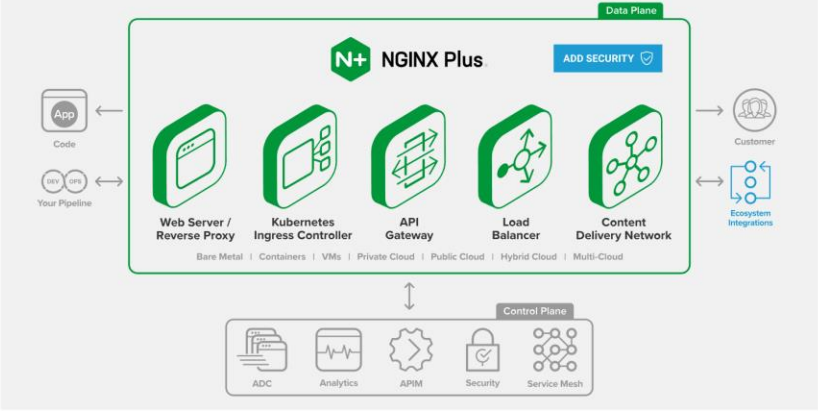
F5 NGINX Plus

The software load balancer, reverse proxy, web server, & content cache with the enterprise features and support you expect.

Try NGINX Plus

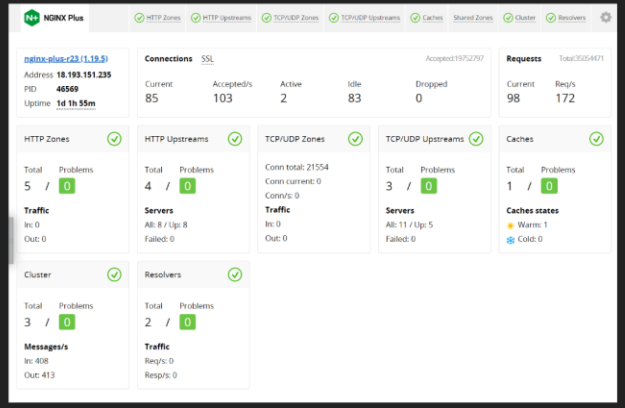
If You Like NGINX, You'll Love NGINX Plus

Modern app infrastructure and dev teams love NGINX Plus. More than just the fastest web server around, NGINX Plus brings you everything you love about NGINX Open Source, adding enterprise-grade features like high availability, active health checks, DNS system discovery, session persistence, and a RESTful API. NGINX Plus is a cloud-native, easy-to-use reverse proxy, load balancer, and API gateway. Whether you need to integrate advanced monitoring, strengthen security controls, or orchestrate Kubernetes containers, NGINX Plus delivers with the five-star support you expect from NGINX.



How to Use NGINX Plus

- Enterprise-Grade Features
- Advanced Security
- Best-of-Breed Support
- Load Balancer
- API Gateway
- Reverse Proxy
- F5 Device ID+



NGINX Plus provides scalable and reliable high availability along with monitoring to support debugging and diagnosing complex application architectures. Active health checks proactively poll upstream server status to get ahead of issues, and the integrated live activity monitoring dashboard provides a single-pane view of your app environment. The NGINX Plus API enables integration with your existing tools, optimizing resources and reducing tool sprawl.

Resources

- Product Page: High Availability →
- Product Page: Live Activity Monitoring →
- Product Page: Dynamic Modules →
- Solution Brief: Sizing Guide for Deploying NGINX Plus on Bare Metal Servers →

2022年应用策略现状报告 - 数字化转型浪潮给予了我们什么启示?

随着数字化转型规模不断扩大，爆炸式增长的应用、集成和平台数量正在创造越来越难以管理的复杂性。像您这样的企业/机构是如何实现现代化？以及在过程中如何避免掉入陷阱?

获取报告

探索应对主要挑战的解决方案

概览　按行业查看　按用例查看

安全性 ›

了解应用容易受到攻击的原因及其受到攻击的方式，以便于制定解决方案来降低风险。

性能 ›

改善负载均衡、安全性、性能和管理，以提供快速、不间断的应用访问。

自动化 ›

使用自动化工具链的组件进行高效预配、配置和管理用于为应用提供支持的服务。

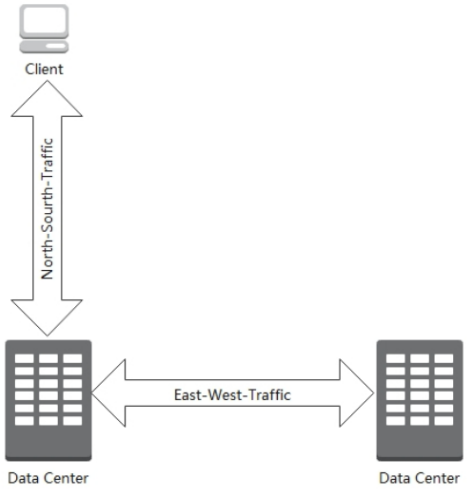
可见性与洞察 ›

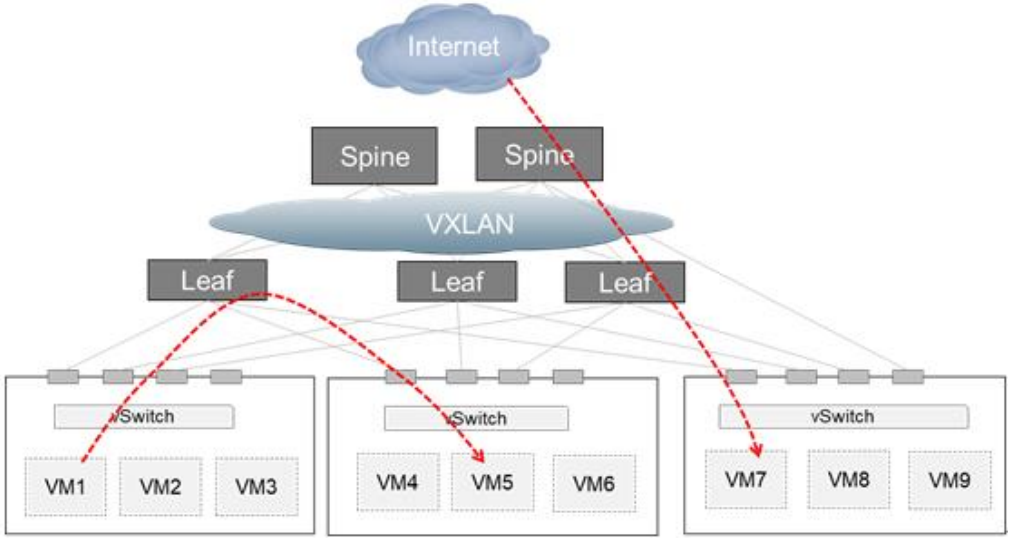
确保对您的应用组合的端到端可见性，以便您可以在问题影响客户体验*前*发现并加以解决。

免受潜伏的危害， 保护您的应用和 API

介绍基于 SaaS 的 F5 基于分布式云的 Web 应用和 API 防护解决方案（WAAP），这是保护部署在多云和分布式环境中的 Web 应用和 API 最全面、有效、易于实施的方法。

南北向流量 ⟶ 南北向流量





推荐阅读：http://blog.nsfocus.net/east-west-flow-sum/

4.使用Apache实现Web负载均衡

- Apache除了可以实现网站服务器，还可以通过mod_proxy模块实现反向代理服务。
- Apache通过mod_proxy实现反向代理后，具备安全性、高可用性、负载均衡等功能，并可实现集中式身份验证授权。

4.使用Apache实现Web负载均衡

- Apache实现反向代理和负载均衡所需的主要模块如下。
  - mod_proxy：支持多种协议的代理，支持的协议如表4-0-1所示。

表 4-0-1 mod_proxy 模块支持协议表

协议	对应模块
AJP13 (Apache JServe Protocol version 1.3)	mod_proxy_ajp
CONNECT（用于 SSL）	mod_proxy_connect
FastCGI	mod_proxy_fcgi
ftp	mod_proxy_ftp
HTTP/0.9, HTTP/1.0, and HTTP/1.1	mod_proxy_http
SCGI	mod_proxy_scgi
WS and WSS (Web-sockets)	mod_proxy_wstunnel

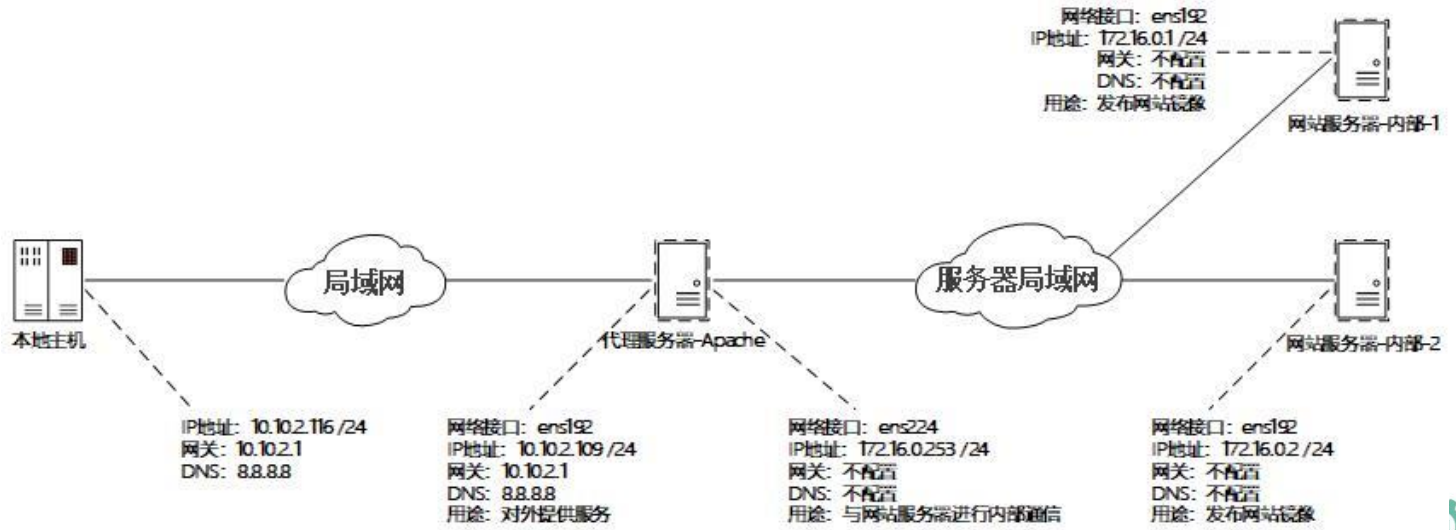
4.使用Apache实现Web负载均衡

- Apache实现反向代理和负载均衡所需的主要模块如下。
  - mod_proxy：支持多种协议的代理。
  - mod_proxy_balancer：提供负载均衡，支持的协议主要有：HTTP、FTP、AJP13、WebSocket等。
  - mod_proxy_balancer本身不提供负载均衡算法模型，而是依托模块mod_lbmethod_byrequests、mod_lbmethod_bytraffic、mod_lbmethod_bybusyness、mod_lbmethod_heartbeat提供算法。
  - mod_proxy_hcheck：提供负载均衡节点的健康检查，此模块需要依赖mod_watchdog模块提供的服务。

  - Apache实现反向代理仅使用mod_proxy即可。
  - Apache实现负载均衡必须同时使用mod_proxy、mod_proxy_balancer以及至少一个负载均衡算法模块。

4.使用Apache实现Web负载均衡

任务5：使用Apache Proxy实现负载均衡



4.使用Apache实现Web负载均衡

任务5：使用Apache Proxy实现负载均衡

表 4-5-1 服务器规划表

虚拟机名称	业务名称	作用
VM-Project-04-Task-04-10.10.2.109	代理服务器-Apache	实现网站反向代理与负载均衡
VM-Project-04-Task-02-172.16.0.1	网站服务器-内部-1	发布内部网站业务
VM-Project-04-Task-03-172.16.0.2	网站服务器-内部-2	发布内部网站业务

表 4-5-2 网站服务器-内部-网站规划表

网站名称	服务器	网站目录	访问地址	网站首页内容
Site-Clone-1	网站服务器-内部-1	/var/www/html	http://172.16.0.1	Site-Clone-1：http://172.16.0.1
Site-Clone-2	网站服务器-内部-2	/var/www/html	http://172.16.0.2	Site-Clone-2：http://172.16.0.2

4.使用Apache实现Web负载均衡

任务5：使用Apache Proxy实现负载均衡

- 步骤1：发布网站Site-Clone-1
- 步骤2：发布网站Site-Clone-2
- 步骤3：创建虚拟机并完成CentOS的安装（用于安装Apache Proxy）
- 步骤4：完成虚拟机的主机配置、网络配置及通信测试
- 步骤5：安装并配置Apache

4.使用Apache实现Web负载均衡

任务5：使用Apache Proxy实现负载均衡

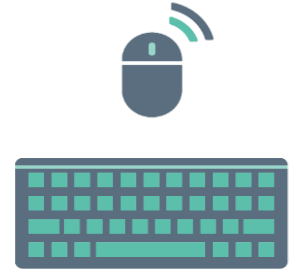
- 步骤6：配置Apache实现负载均衡
- 步骤7：重新载入Apache的配置文件
- 步骤8：负载均衡业务服务的实时监控
- 步骤9：验证负载均衡服务
- 步骤10：验证负载均衡对内部业务的容灾性

表 4-5-3 虚拟机与操作系统配置

虚拟机配置	操作系统配置
虚拟机名称： VM-Project-04-Task-04-10.10.2.109 内存：1024MB CPU：1 颗 1 核心 虚拟硬盘：10GB 网卡：2 块 网卡 1 桥接，网卡 2 内部网络	主机名：Project-04-Task-04
	网络接口：ens192 IP 地址：10.10.2.109 子网掩码：255.255.255.0 网关：10.10.2.1 DNS：8.8.8.8
	网络接口：ens224 IP 地址：172.16.0.253 子网掩码：255.255.255.0 网关：不配置 DNS：不配置

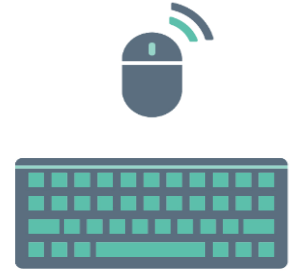
操作视频 / 现场演示

- 任务5：使用Apache Proxy实现负载均衡
  - 任务目标：
    - 实现Apache反向代理与负载均衡发布网站服务；
    - 实现Apache负载均衡的测试；
    - 实现Apache负载均衡的运行监控。



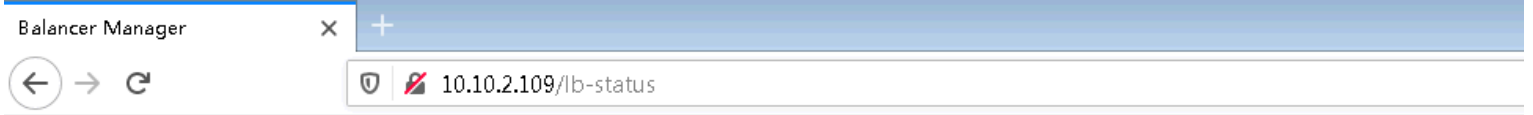
命令指南 / 操作引导

```
#在Apache Proxy服务器上操作
#通过Apache的mod_proxy、mod_proxy_balancer、mod_lbmethod_byrequests模块实现负载均衡。
#使用vi工具编辑Apache的httpd.conf配置文件。

配置文件：/etc/httpd/conf/httpd.conf
#httpd.conf配置文件内容较多，本部分仅显示与负载均衡配置有关的内容
#配置Apache mod_proxy禁止使用反向代理，以负载均衡方式发布网站服务。
ProxyRequests Off
#定义负载均衡网站服务器组名称、网站服务器地址信息、权重信息
<Proxy balancer://load2>
    BalancerMember http://172.16.0.1:80 loadfactor=1
    BalancerMember http://172.16.0.2:80 loadfactor=3
</Proxy>
#定义代理转发请求到负载均衡网站服务器组load2
ProxyPass / balancer://load2

#配置完成后，重新载入配置文件使其生效。
 [root@Project-04-Task-04 ~]# systemctl reload httpd
#关闭防火墙等安全措施
#使用systemctl stop命令关闭防火墙
[root@Project-04-Task-04 ~]# systemctl stop firewalld
#使用setenforce命令将SELinux设置为permissive模式
[root@Project-04-Task-04 ~]# setenforce 0
```



命令指南 / 操作引导

```
#在Apache Proxy服务器上操作
#Apache通过mod_status、mod_proxy_balancer模块实现负载均衡运行状态和性能的实时监控。
#使用vi工具编辑Apache的httpd.conf配置文件。

配置文件：/etc/httpd/conf/httpd.conf
#httpd.conf配置文件内容较多，本部分仅显示与开启负载均衡运行监控有关的内容
#配置网站负载均衡实时监控访问路径
<Location "/lb-status">
    #设置不进行转发
    proxypass !
    #设置监控负载均衡性能
    SetHandler balancer-manager
</Location>

#配置完成后，重新载入配置文件使其生效。
 [root@Project-04-Task-04 ~]# systemctl reload httpd
```



Load Balancer Manager for 10.10.2.109

Server Version: Apache/2.4.37 (centos)
Server Built: Dec 23 2019 20:45:34
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

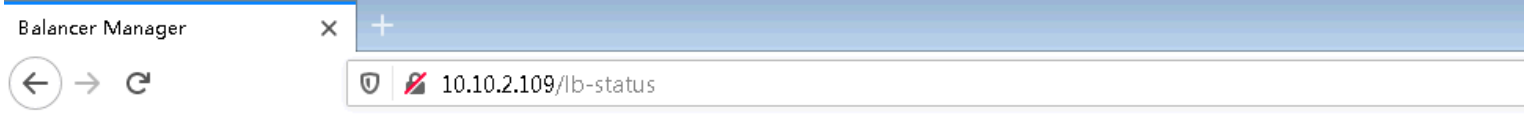
LoadBalancer Status for balancer://load2 [p85af8949_load2]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Me
2 [2 Used]	(None)	Off	0	1	byred

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To
http://172.16.0.1			1.00	0	Init Ok	251	0	-200	46K
http://172.16.0.2			3.00	0	Init Ok	750	0	200	137K

表 4-5-4 监控信息说明表

字段	字段名	功能
1	Server Version	网站服务器当前版本信息
2	Server Built	服务器创建时间
3	MaxMembers	负载均衡节点最大数量及使用数量
4	StickySession	粘性会话设置信息
5	DisableFailover	禁用故障转移
6	Timeout	超时时间
7	FailoverAttempts	故障转移尝试次数
8	Method	负载均衡模式
9	Path	负载均衡配置路径
10	Active	是否活跃
11	Worker URL	负载均衡节点地址
12	Route	设置路由信息
13	Route Redirect	路由重定向地址
14	Factor	节点权重信息



Load Balancer Manager for 10.10.2.109

Server Version: Apache/2.4.37 (centos)
Server Built: Dec 23 2019 20:45:34
Balancer changes will NOT be persisted on restart.
Balancers are inherited from main server.
ProxyPass settings are inherited from main server.

LoadBalancer Status for balancer://load2 [p85af8949_load2]

MaxMembers	StickySession	DisableFailover	Timeout	FailoverAttempts	Me
2 [2 Used]	(None)	Off	0	1	byrec

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	Busy	Load	To
http://172.16.0.1			1.00	0	Init Ok	251	0	-200	46K
http://172.16.0.2			3.00	0	Init Ok	750	0	200	137K

表 4-5-4 监控信息说明表

字段	字段名	功能
15	Set	设置负载均衡节点编号
16	Status	该负载均衡节点状态
17	Elected	向该节点转发的请求数
18	Busy	该节点处于繁忙状态的请求数
19	Load	该节点负载值
20	To	该节点响应的总字节数
21	From	该节点接收的总字节数
22	HC Method	健康状态检查模式
23	HC Interval	健康检查时间间隔
24	Passes	成功的运行状况健康检查次数，默认值为 1
25	Fails	失败的运行状况健康检查次数，默认值为 1
26	HC uri	设置的信息将附加到 Worker URL 以进行健康检查
27	HC Expr	表达式名称，用于检查响应头健康状态

4.使用Apache实现Web负载均衡

- Apache实现负载均衡，主要有以下四种算法模型。
  - byrequests：
    - 按照请求次数进行负载均衡，未设置负载均衡模式时，默认使用该项
  - bytraffic：
    - 按照请求字节数进行负载均衡
  - bybusyness：
    - 按照繁忙程度进行负载均衡，总是分配给活跃请求最少的节点
  - heartbeat：
    - 按照心跳流量进行负载均衡，Apache 2.3及之后版本才支持的负载均衡模式，属于实验性质的模块