Nginx

1. 基本介绍

Nginx是高性能的http/https和反向代理服务器，特点是占用内存少，处理高并发能力强大。有报告表明Nginx能够支持50000个并发连接数。

1. 安装

**安装环境：CentOS7**

1. 安装gcc-c++

yum install gcc-c++

1. 安装pcre

wget [http://downloads.sourceforge.net/project/pcre/pcre/8.44/pcre-8.44.tar.gz](http://downloads.sourceforge.net/project/pcre/pcre/8.37/pcre-8.37.tar.gz)

tar -xvf pcre-8.44.tar.gz

./configigure

make

make install

1. 安装openssl、zlib、gcc

yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel

1. 安装nginx

官网下载：http://nginx.org/

tar -xvf nginx-1.16.1.tar.gz

./configigure

make

make install

1. 命令

nginx安装目录：/usr/local/nginx

命令行操作nginx前，先cd到/usr/local/nginx/sbin

查看版本号 ./nginx -v

启动nginx ./nginx

停止nginx ./nginx -s stop

重加载nginx ./nginx -s reload

1. 访问nginx主页

centos防火墙默认关闭80端口，外网需要访问到该服务，需要开放对应端口号

查看防火墙开放全部端口号

firewall-cmd --list-all

添加端口号

firewall-cmd --add-service=http --permanent

firewall-cmd --add-port=80/tcp --permanent

重启防火墙

Firewall-cmd --reload

1. nginx配置文件

#user nobody;

# 全局块

worker\_processes 1; # 工作进程数

# 事件块

events {

worker\_connections 1024; # 单个工作进程连接数

}

# http块

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

#tcp\_nopush on;

keepalive\_timeout 65; # 保活超时检测时间

#gzip on;

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root html;

index index.html index.htm;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000;

# fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

# include fastcgi\_params;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080;

# server\_name somename alias another.alias;

# location / {

# root html;

# index index.html index.htm;

# }

#}

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# ssl\_certificate cert.pem;

# ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

# index index.html index.htm;

# }

#}

}

1. 配置实例-反向代理
2. 正向代理

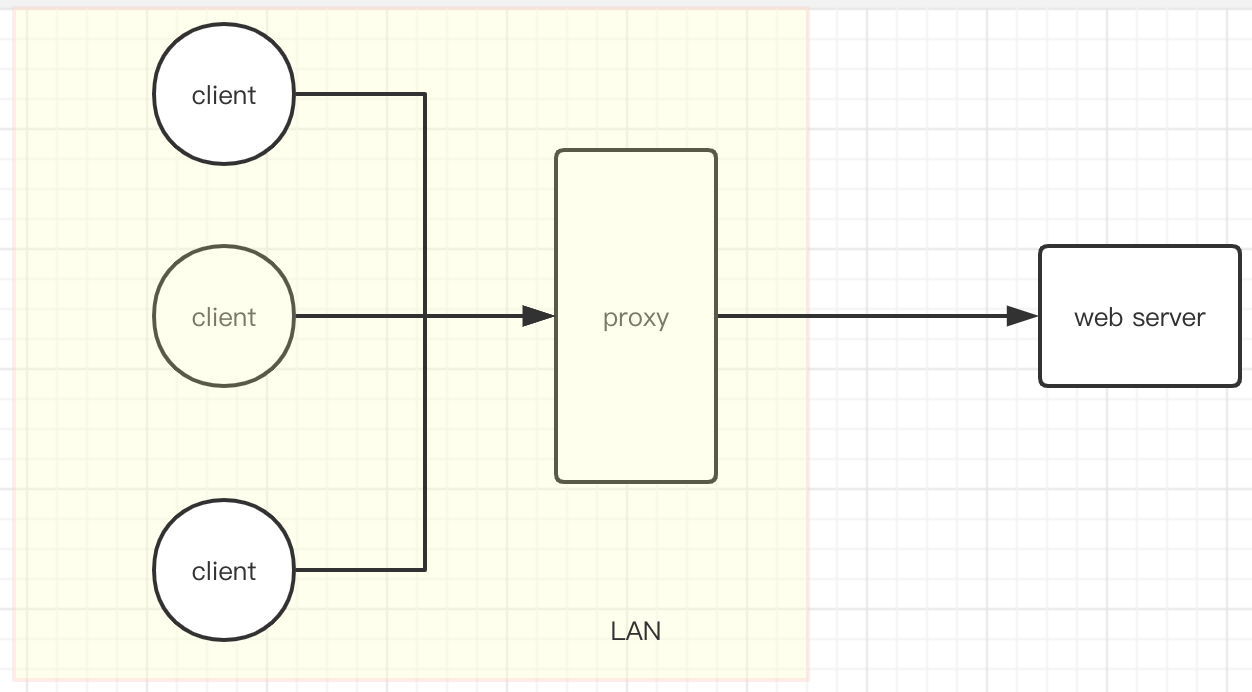


图1 正向代理示意图

客户端和代理服务器处于同一局域网，客户端通过代理服务器向web服务器请求数据，web服务器通过代理服务器返回数据给客户端。

1. 反向代理

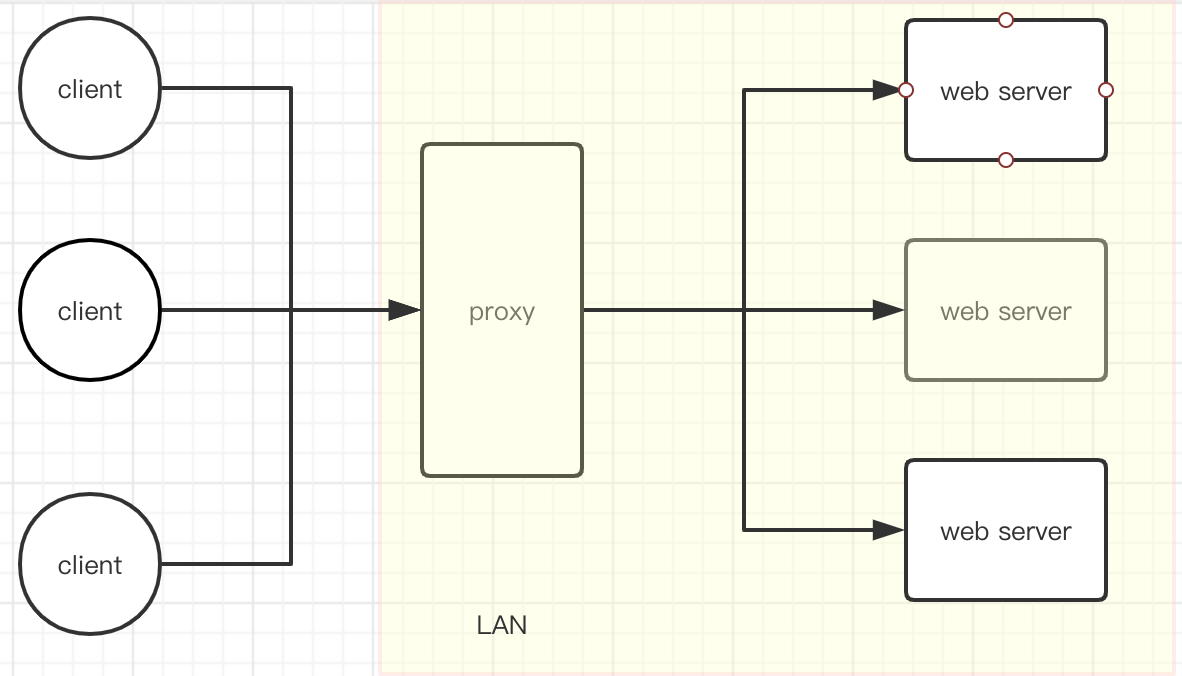


图2 反向代理示意图

反向代理和正向代理刚好相反，代理服务器和web服务器处于同一局域网，客户端对web服务器是无感知的。

1. nginx实现反向代理



图3 反向代理nginx配置文件

使用tomcat作为web服务器，nginx作为代理服务器。ngixn将客户端请求转发到tomcat。

1. 配置实例-负载均衡
2. 负载均衡概念

负载均衡指的是有多台服务器接受客户端的请求，按照平均分发或者权重比例的方式，将客户端的请求(负载)分发不同的服务器，使之达到负载均衡的效果。

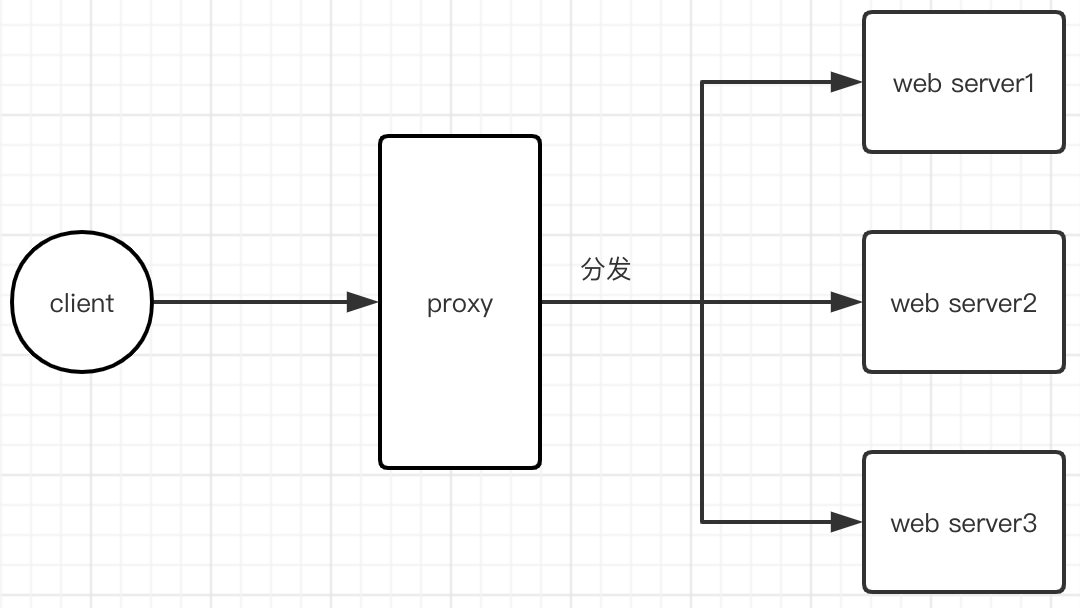


图4 负载均衡示意图

1. nginx实现负载均衡

* 我们需要准备两个tomcat服务器，端口号分别为8080、8081
* 在tomcat/webapps目录下新增test目录，test目录下新增test.html文件
* 修改nginx配置文件，如下

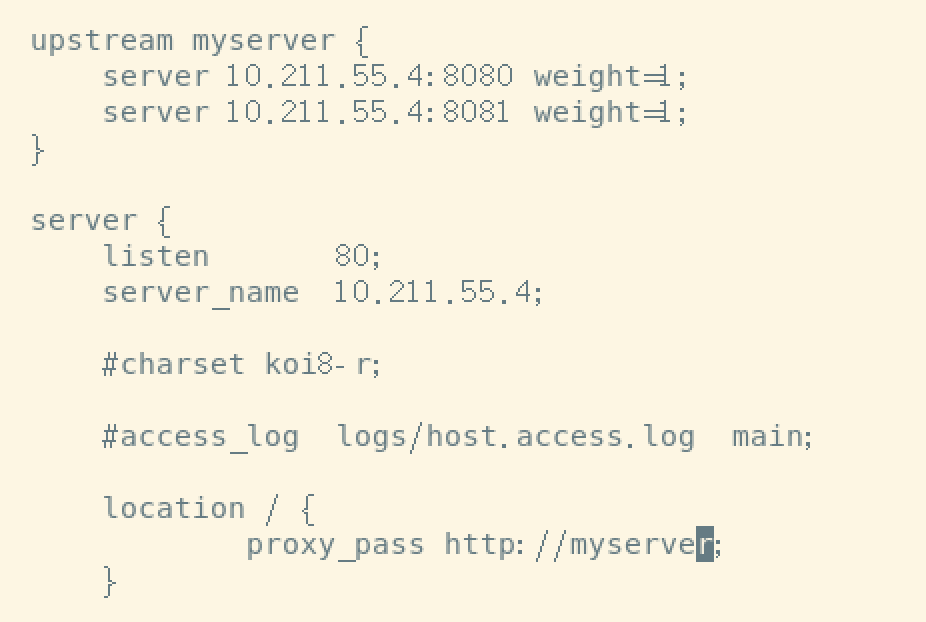


图5 负载均衡配置文件

* 分别启动两台tomcat和nginx
* 浏览器URL栏输入<http://10.211.55.4/test/test.html>验证

1. nginx分配服务策略
2. 轮询(默认)

每个请求按时间顺序逐一分配。如果后面的服务宕机，能自动剔除。

1. 权重

weight代表权重，默认值为1.权重越高分配的客户端请求越多。weight和访问比例成正比，用于服务器性能不均的情况。如下

*upstream server\_pool{*

*server 10.211.55.4:8080 weight=10;*

*server 10.211.55.4:8081 weight=10;*

*}*

1. ip\_hash

按ip的hash值来分配请求，可以保证每个访客固定请求一个服务器，解决session问题。

*upstream server\_pool{*

*ip\_hash;*

*server 10.211.55.4:8080;*

*server 10.211.55.4:8081;*

*}*

1. fair

按后端服务器的响应时间来分配请求，响应时间短的优先分配。

upstream server\_pool{

server *10.211.55.4:8080;*

server *10.211.55.4:8081;*

fair;

}

1. 配置实例-动静分离
2. 动静分离的概念

通过 location 指定不同的后缀名实现不同的请求转发。通过 expires 参数设置，可以使浏览器缓存过期时间，减少与服务器之前的请求和流量。具体 Expires 定义：是给一个资源设定一个过期时间，也就是说无需去服务端验证，直接通过浏览器自身确认是否过期即可，所以不会产生额外的流量。此种方法非常适合不经常变动的资源。（如果经常更新的文件，不建议使用 Expires 来缓存），我这里设置 3d，表示在这 3 天之内访问这个 URL，发送一个请求，比对服务器该文件最后更新时间没有变化，则不会从服务器抓取，返回状态码 304，如果有修改，则直接从服务器重新下载，返回状态码 200。

动静分离从目前实现角度来讲大致分为两种，一种是纯粹把静态文件独立成单独的域名，放在独立的服务器上，也是目前主流推崇的方案；另外一种方法就是动态跟静态文件混合在一起发布，通过 nginx来分开。

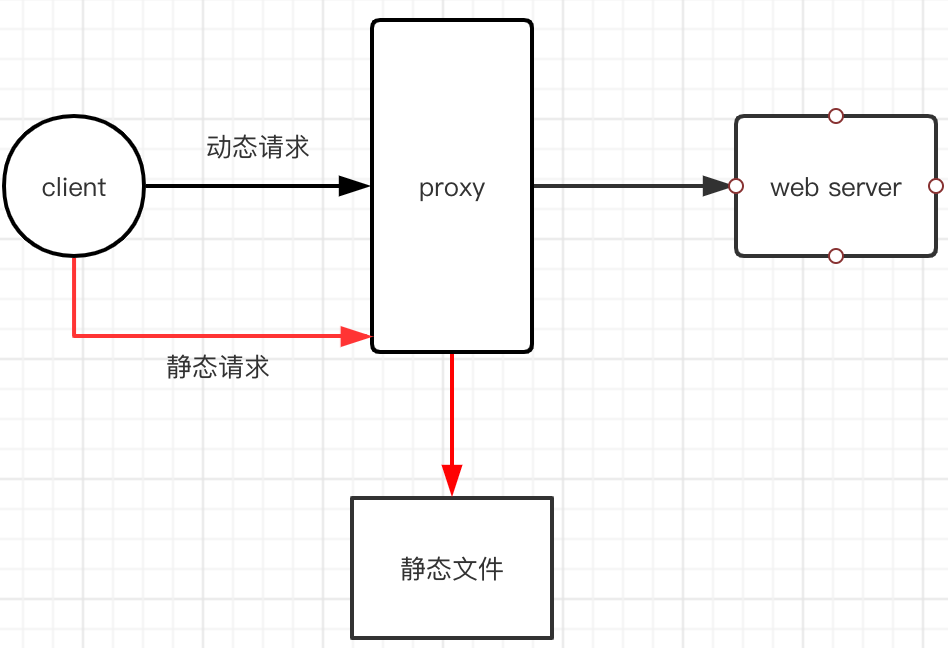


图6 动静分离示意图

1. nginx实现动静分离

* 在nginx所在服务器根目录创建data目录，data目录下创建www、books目录，目录下分别放置一些文件
* 修改nginx配置文件，如下：

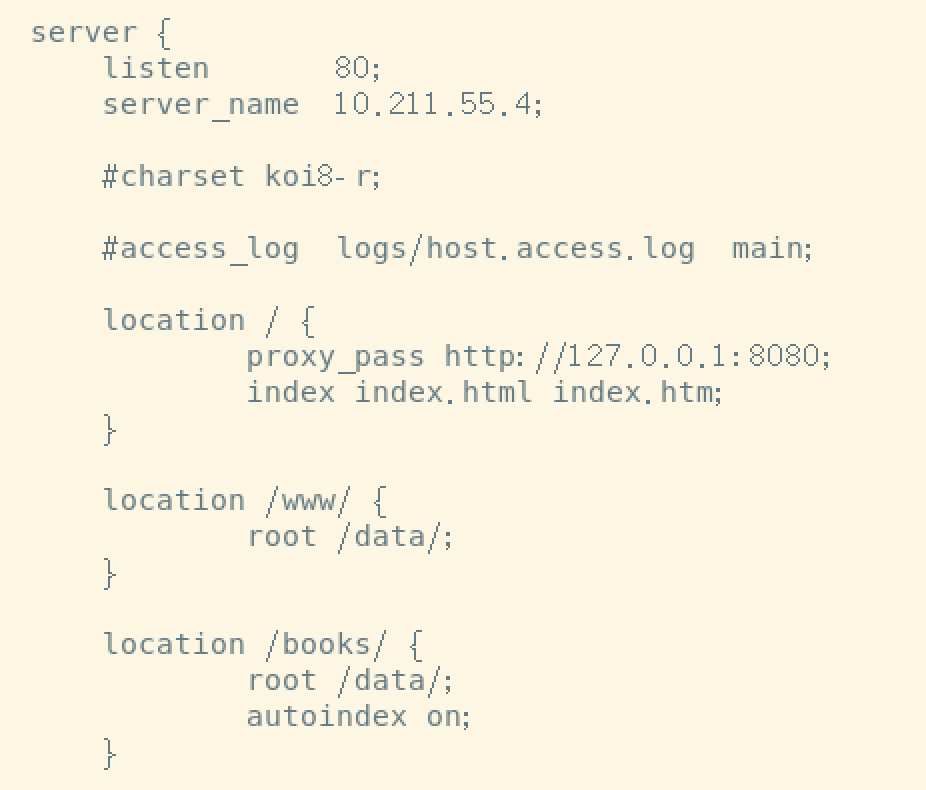


图7 nginx动静分离配置文件

* 分别启动tomcat、nginx

1. 测试动静分离

* 访问静态文件：浏览器中输入10.211.55.4/www/test.html



图8 测试静态资源(一)

* 浏览器中输入<http://10.211.55.4/books/UML2.pdf>

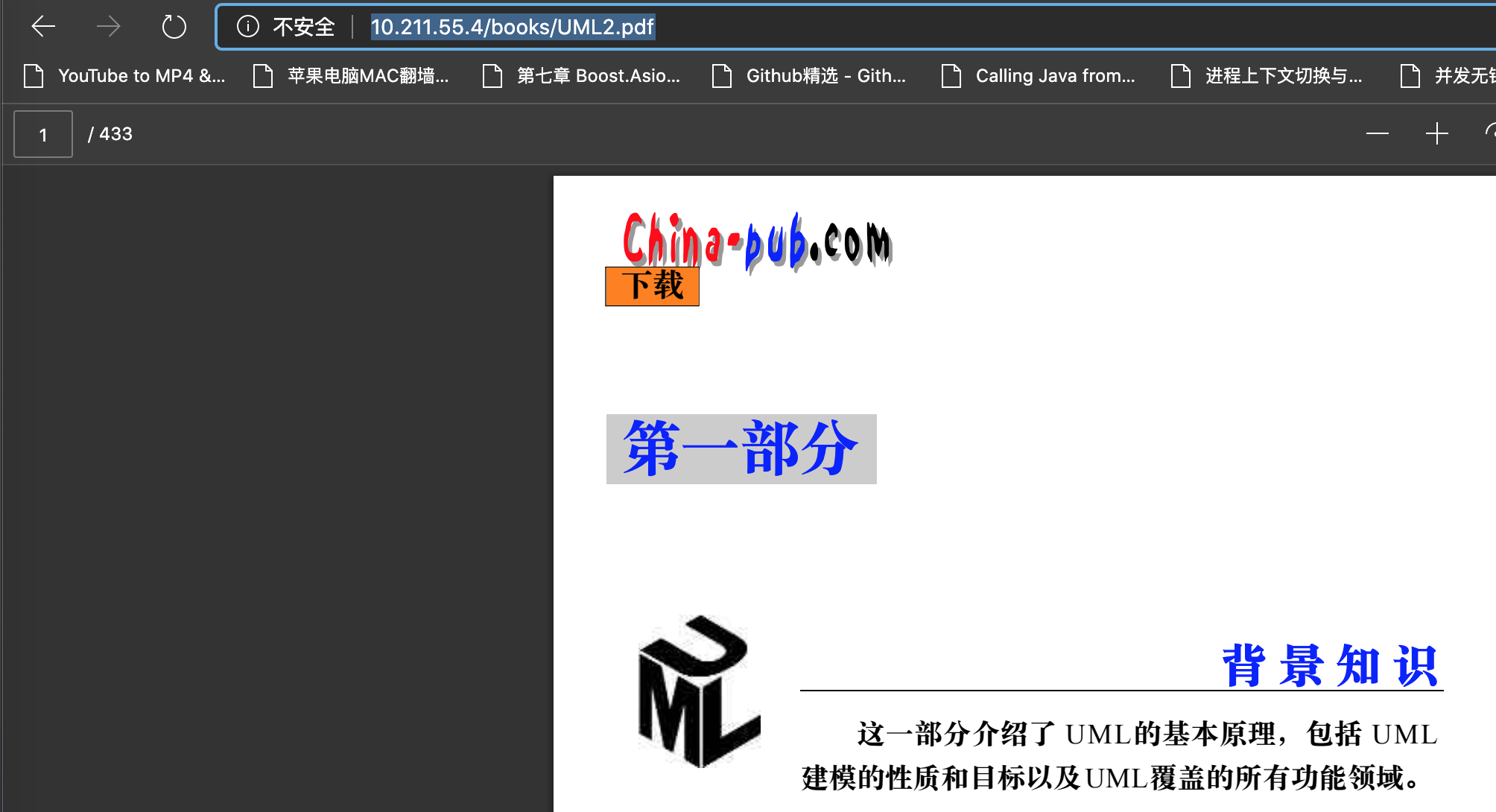


图9 测试静态资源(二)

**autoindex属性**

* 当这个属性值设置为on时，可以访问当前目录下当文件列表，如下



图10 autoindex属性为on的效果

* 访问动态页面：浏览器输入<http://10.211.55.4/>

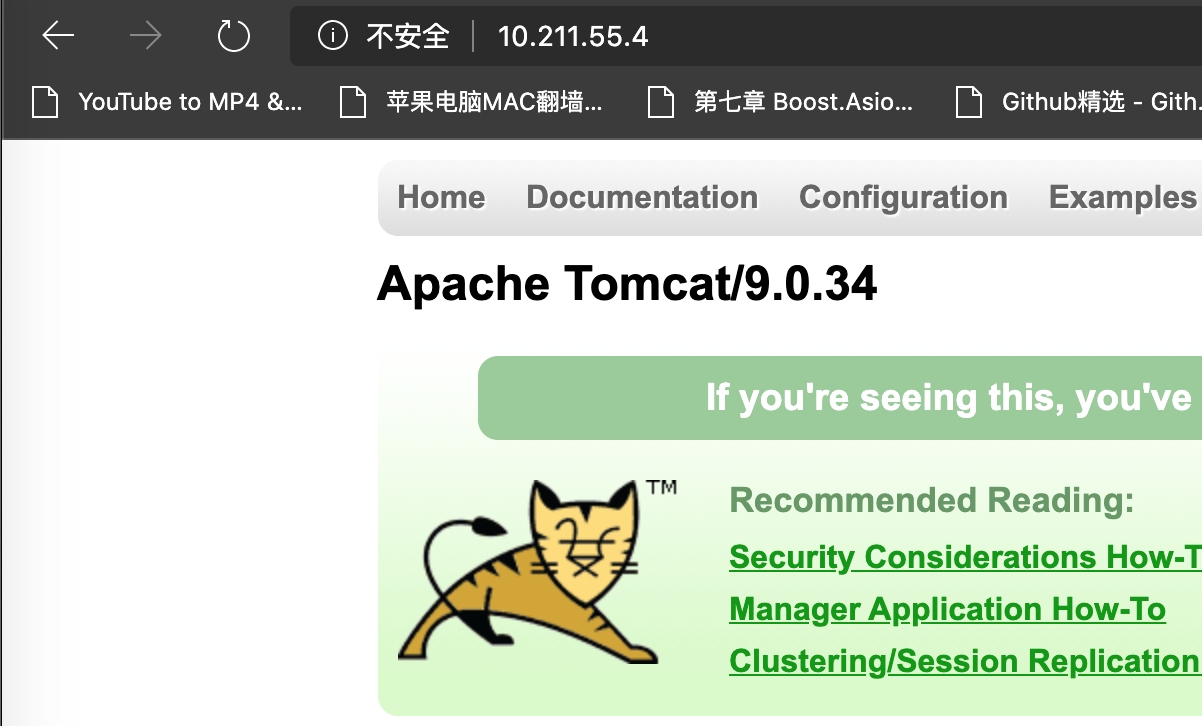


图11 测试访问动态页面

1. 高可用集群(主从模式)
2. 高可用概念

在两台服务器设置两个服务(主/备)，通过虚拟IP对外提供唯一当访问IP。正常情况下访问主服务；当主服务宕机后，自动切换到备份服务上，继续对外提供访问服务。

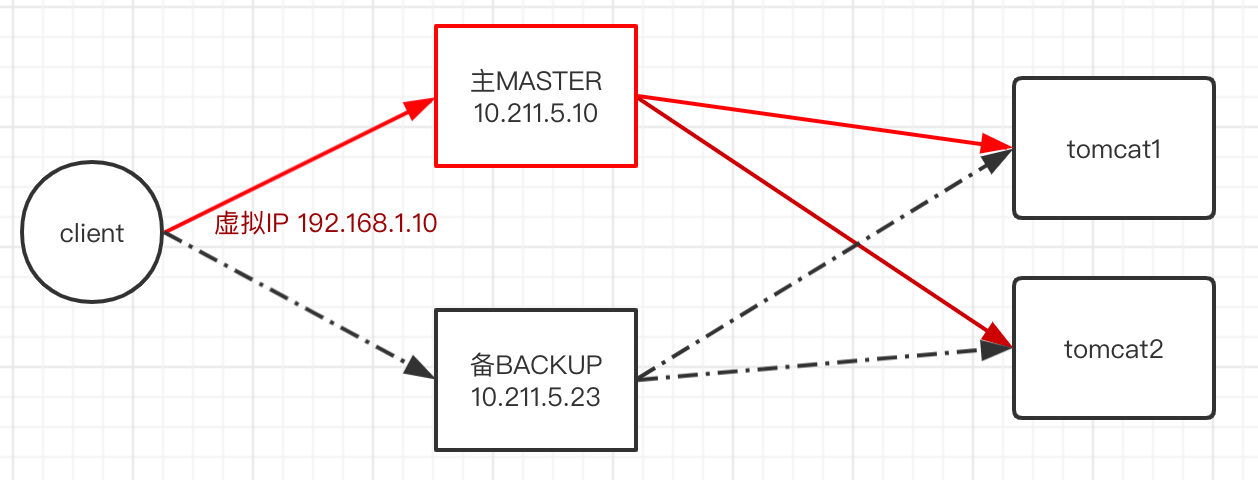


图12 高可用示意图

1. 安装keepalived

两台主机上都要安装nginx和keepalivied。centos环境下使用命令来安装keepalived：**yum install keepalived -y**

安装完成后，生成/etc/keepalived/keepalived.conf配置文件

1. 高可用配置

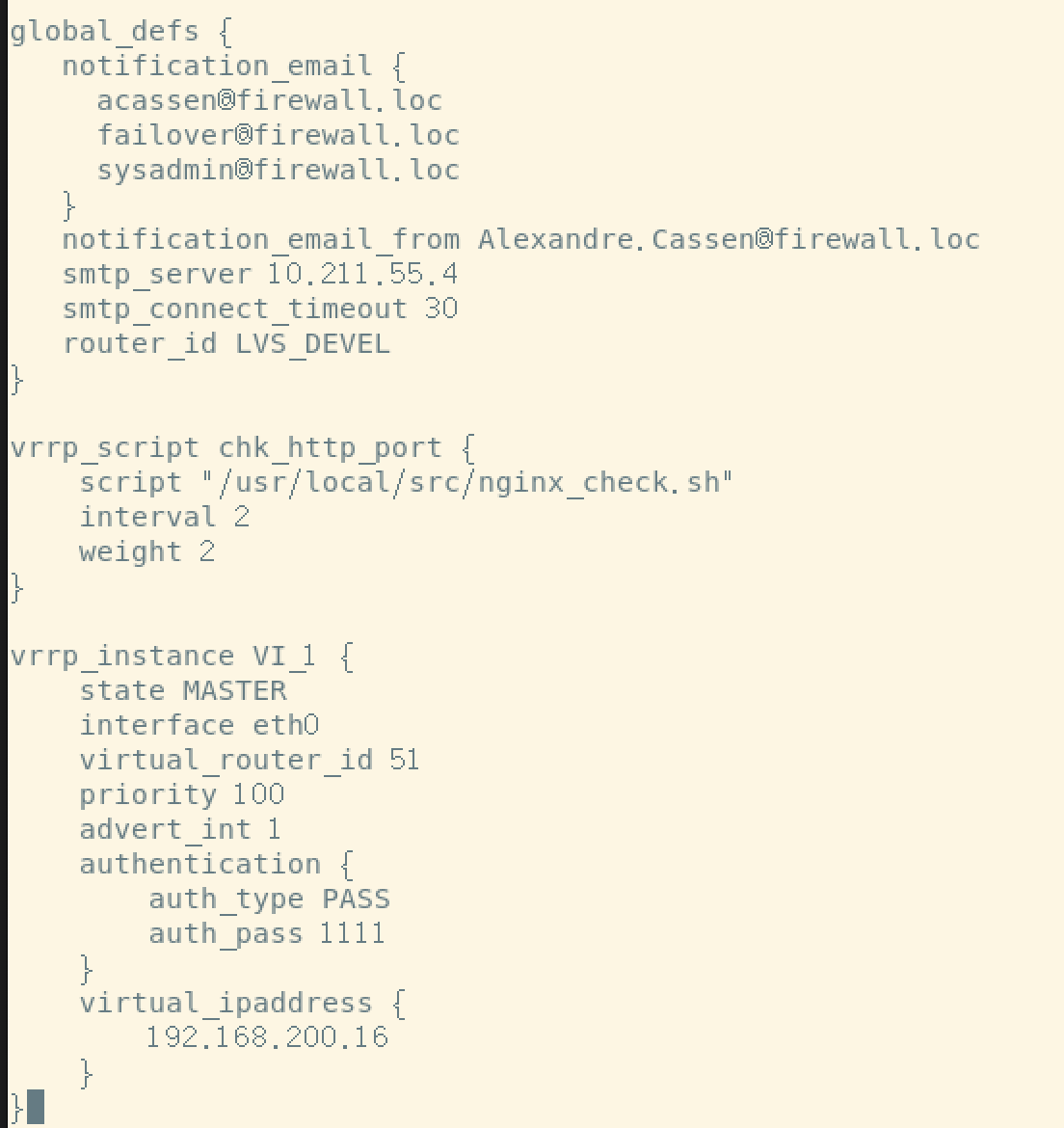


图13 主服务器配置文件

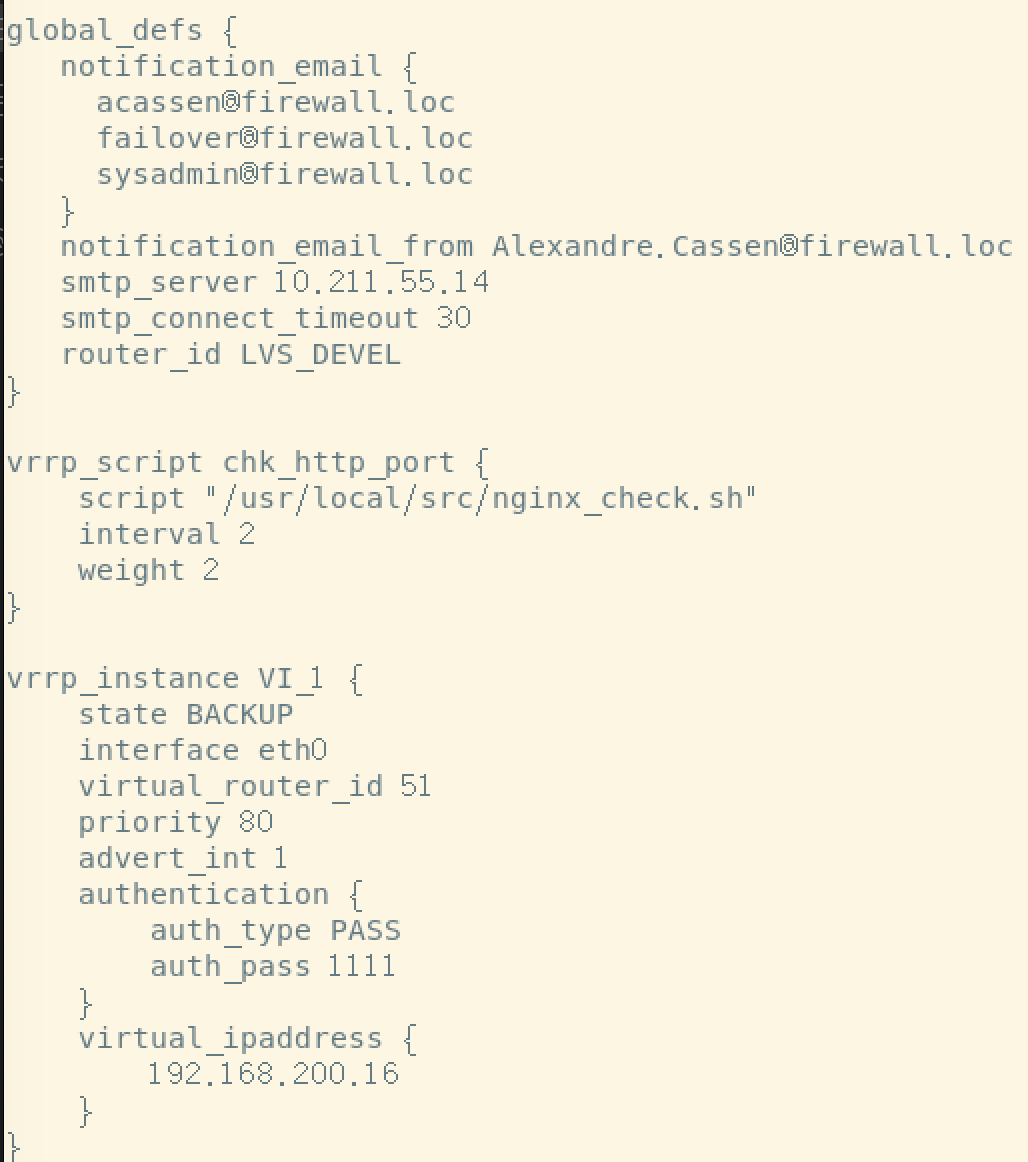


图14 备份服务器配置文件

1. 添加检查脚本

/usr/local/src/nginx\_check.sh

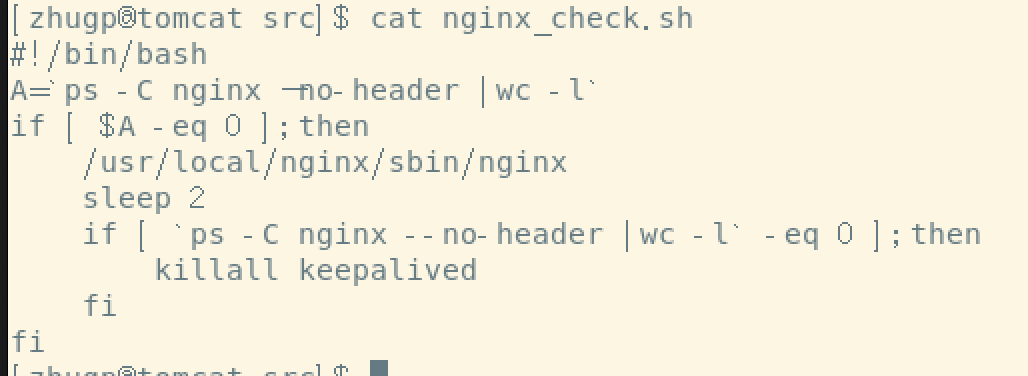


图15 nginx检测脚本

1. 把两台服务器上 nginx 和 keepalived 启动

启动 nginx：./nginx

启动 keepalived：systemctl start keepalived.service

1. 测试

浏览器输入虚拟IP：192.168.200.16

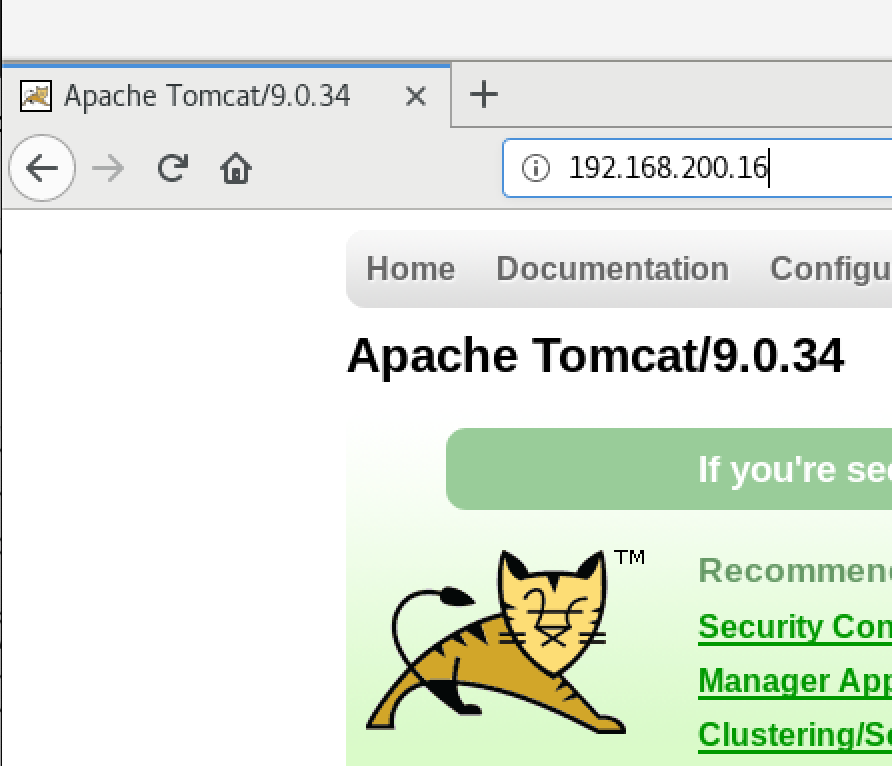


图16 访问主服务器

第二步，停掉主服务器，按照第一步操作访问：192.168.200.16

1. nginx原理

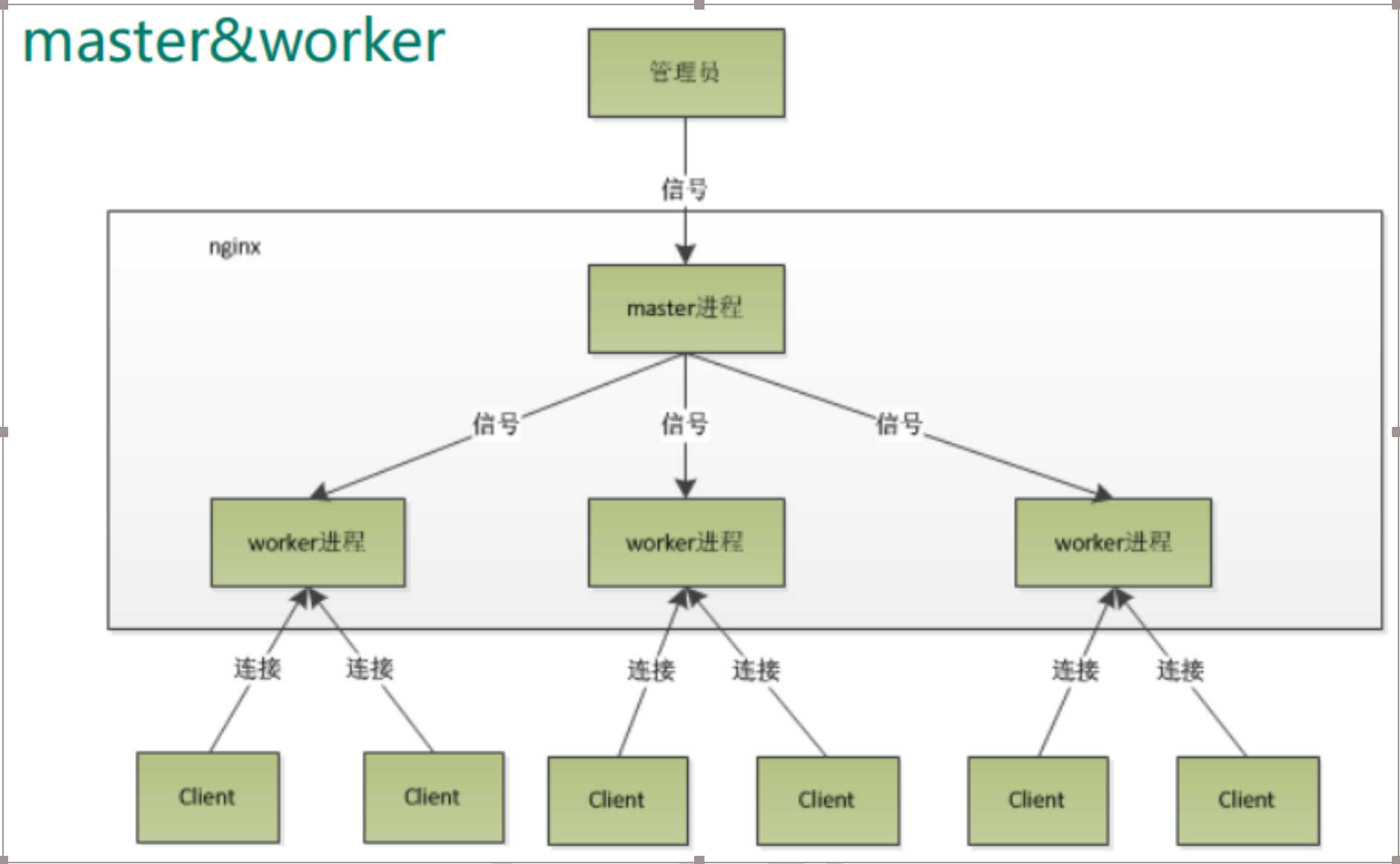


图17 nginx原理图

1. master-workers 的机制的好处

稳定性：多进程架构，任一进程挂掉不会影响其他线程执行

不需要加锁：每个worker是一个独立的进程，无需加锁，避免锁带来的开销

1. 需要设置多少个 worker

Nginx同redis 类似都采用了 io 多路复用机制，每个 worker 都是一个独立的进程，但每个进程里只有一个主线程，通过异步非阻塞的方式来处理请求， 即使是千上万个请求也不在话下。每个 worker 的线程可以把一个 cpu 的性能发挥到极致。所以 worker 数和服务器的 cpu数相等是最为适宜的。设少了会浪费 cpu，设多了会造成 cpu 频繁切换上下文带来的损耗。

1. 连接数 worker\_connection

这个值是表示每个 worker 进程所能建立连接的最大值，所以，一个 nginx 能建立的最大连接数，应该是worker\_connections \* worker\_processes。当然，这里说的是最大连接数，对于HTTP请求本地资源来说，能够支持的最大并发数量是worker\_connections \* worker\_processes，如果是支持 http1.1 的浏览器每次访问要占两个连接，所以普通的静态访问最大并发数是：worker\_connections \* worker\_processes /2，而如果是 HTTP作为反向代理来说，最大并发数量应该是worker\_connections \* worker\_processes/4。因为作为反向代理服务器，每个并发会建立与客户端的连接和与后端服务的连接，会占用两个连接。