link: null
title: 珠峰架构师成长计划
description: nginx是一个开源且高性能、可靠的HTTP中间件和代理服务器
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=165 sentences=213, words=1348

1. nginx

nginx是一个开源且高性能、可靠的HTTP中间件和代理服务器

2. 学习环境

2.1 操作系统

CENTOS>=7.0,位数 X64 CENTOS 7.2

2.2 环境确认

2.2.1 关闭 iptables

iptables命令是Linux上常用的防火墙软件

功能 命令 停止防火墙 systemctl stop firewalld.service 永久关闭防火墙 systemctl disable firewalld.service

2.2.2 确认停用 selinux

- 安全增强型 Linux（Security-Enhanced Linux）简称 SELinux，它是一个 Linux 内核模块，也是 Linux 的一个安全子系统。
- SELinux 主要作用就是最大限度地减小系统中服务进程可访问的资源（最小权限原则）。

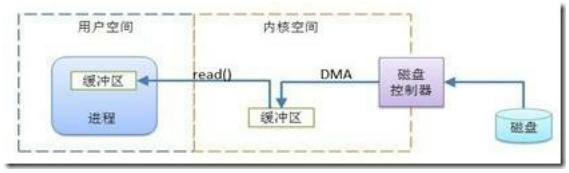
功能 命令 检查状态 getenforce 检查状态 /usr/sbin/sestatus -v 临时关闭 setenforce 0 永久关闭 /etc/selinux/config SELINUX=enforcing改为SELINUX=disabled

2.2.3 安装依赖模块

```
yum  -y install gcc gcc-c++ autoconf pcre pcre-devel make automake
yum  -y install wget httpd-tools vim
```

3. nginx的优势

- IO多路复用 多个描述符的IO操作都能在一个线程里并发交替顺序完成，复用线程
    - select 线性遍历文件描述符列表 1. 效率低下 2.最多只能有1024
    - epoll 每当fd就绪，采用系统回调函数将fd放入 1.效率高 2.没有1024限制
- CPU亲和 一种把CPU核心和Nginx工作进程绑定方式，把每个worker进程固定在一个CPU上执行，减少切换CPU和提交缓存命中率,获得更好的性能。
- sendfile 零拷贝传输模式 ![usercore]



4. nginx安装

4.1 版本分类

- Mainline version 开发版
- Stable version 稳定版
- Legacy versions 历史版本

4.2 下载地址

- nginx (http://nginx.org/en/download.html)
- linux_packages (http://nginx.org/en/linux_packages.html#stable)

4.3 CentOS下YUM安装

/etc/yum.repos.d/nginx.repo

```
[nginx]
name=nginx repo
baseurl=http:
gpgcheck=0
enabled=1
```

```
yum install nginx -y
nginx -v
nginx -V
```

5. 目录

5.1 安装目录

查看配置文件和目录

```
rpm -ql nginx
```

5.2 配置文件

类型 路径 用途 配置文件 /etc/logrotate.d/nginx 用于logrotate服务的日志切割 配置文件 /etc/nginx /etc/nginx/nginx.conf /etc/nginx/conf.d /etc/nginx/conf.d/default.conf 主配置文件 配置文件 /etc/nginx/fastcgi_params /etc/nginx/scgi_params /etc/nginx/uwsgi_params cgi配置,fastcgi配置 配置文件 /etc/nginx/koi-utf /etc/nginx/koi-win /etc/nginx/win-utf 编码转换映射转化文件 配置文件 /etc/nginx/mime.types 设置http协议的Content-Type与扩展名对应关系 配置文件 /usr/lib/systemd/system/nginx-debug.service /usr/lib/systemd/system/nginx.service /etc/sysconfig/nginx /etc/sysconfig/nginx-debug 用于配置系统守护进程管理器管理方式 配置文件 /etc/nginx/modules /usr/lib64/nginx/modules nginx模块目录 命令 /usr/share/doc/nginx-1.14.0 /usr/share/doc/nginx-1.14.0/COPYRIGHT nginx的手册和帮助文件 目录 /var/cache/nginx nginx的缓存目录 目录 /var/log/nginx nginx的日志目录

5.3 编译参数

5.3.1 安装目录和路径

```
--prefix=/etc/nginx
--sbin-path=/usr/sbin/nginx
--modules-path=/usr/lib64/nginx/modules
--conf-path=/etc/nginx/nginx.conf
--error-log-path=/var/log/nginx/error.log
--http-log-path=/var/log/nginx/access.log
--pid-path=/var/run/nginx.pid
--lock-path=/var/run/nginx.lock
```

5.3.2 执行对应模块时，nginx所保留的临时性文件

```
--http-client-body-temp-path=/var/cache/nginx/client_temp
--http-proxy-temp-path=/var/cache/nginx/proxy_temp
--http-fastcgi-temp-path=/var/cache/nginx/fastcgi_temp
--http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp
--http-scgi-temp-path=/var/cache/nginx/scgi_temp
```

5.3.3 设置nginx进程启动的用户和用户组

```
--user=nginx
--group=nginx
```

5.3.4 设置额外的参数将被添加到CFLAGS变量

```
--with-cc-opt='-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong
```

5.3.5 设置附加的参数，链接系统库

```
--with-ld-opt='-Wl,-z,relro -Wl,-z,now -pie'
```

5.3.6 其它参数

```
--with-compat
--with-file-aio
--with-threads
--with-http_addition_module
--with-http_auth_request_module
--with-http_dav_module
--with-http_flv_module
--with-http_gunzip_module
--with-http_gzip_static_module
--with-http_mp4_module
--with-http_random_index_module
--with-http_realip_module
--with-http_secure_link_module
--with-http_slice_module
--with-http_ssl_module
--with-http_stub_status_module
--with-http_sub_module
--with-http_v2_module
--with-mail
--with-mail_ssl_module
--with-stream
--with-stream_realip_module
--with-stream_ssl_module
--with-stream_ssl_preread_module
--param=ssp-buffer-size=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC'
```

5. 配置文件

- /etc/nginx/nginx.conf
- /etc/nginx/conf.d/*.conf /etc/nginx/conf.d/default.conf

5.1 全局和服务配置

分类 配置项 作用 全局 user 设置nginx服务的系统使用用户 全局 worker_processes 工作进程数,一般和CPU数量相同 全局 error_log nginx的错误日志 全局 pid nginx服务启动时的pid

5.2 全局和服务配置

分类 配置项 作用 events worker_connections 每个进程允许的最大连接数 10000 events use 指定使用哪种模型（select/poll/epoll）,建议让nginx自动选择,linux内核2.6以上一般能使用epoll，提高性能。

/etc/nginx/nginx.conf

```
user  nginx;    设置nginx服务的系统使用用户
worker_processes  1;   工作进程数,一般和CPU数量相同

error_log  /var/log/nginx/error.log warn;    nginx的错误日志
pid        /var/run/nginx.pid;    nginx服务启动时的pid

events {
    worker_connections  1024;每个进程允许的最大连接数 10000
}

http {
    include       /etc/nginx/mime.types;//文件后缀和类型类型的对应关系
    default_type  application/octet-stream;//默认content-type

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';  //日志记录格式

    access_log  /var/log/nginx/access.log  main;//默认访问日志

    sendfile        on;//启用sendfile
    #tcp_nopush     on;//懒发送

    keepalive_timeout  65;//超时时间是65秒

    #gzip  on;gzip压缩

    include /etc/nginx/conf.d/*.conf;//包含的子配置文件
}
```

default.conf

```
server {
    listen       80;  //监听的端口号
    server_name  localhost;  //用域名方式访问的地址

    #charset koi8-r; //编码
    #access_log  /var/log/nginx/host.access.log  main;  //访问日志文件和名称

    location / {
        root   /usr/share/nginx/html;  //静态文件根目录
        index  index.html index.htm;  //首页的索引文件
    }

    #error_page  404              /404.html;  //指定错误页面

    # redirect server error pages to the static page /50x.html
    # 把后台错误重定向到静态的50x.html页面
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    # 代理PHP脚本到80端口上的apache服务器
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    # 把PHP脚本9000端口上监听的FastCGI服务
    #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
    # 不允许访问.htaccess文件
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

6. 启动和重新加载

```
systemctl restart nginx.service
systemctl reload nginx.service
nginx -s reload
```

7. 日志类型

7.1 日志类型

- access_.log 访问日志
- error.log 错误日志

7.2 log_format

类型 用法 语法 log_format name [escape=default|json] string] 默认 log_format combined ... Context http

案例

```
log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';

log_format  zfpx  '$arg_name $http_referer sent_http_date"';
access_log  /var/log/nginx/access.log  main;

221.216.143.110 - - [09/Jun/2018:22:41:18 +0800] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/62.0.3202.94 Safari/537.36" "-"
```

7.3 HTTP请求变量

名称 含义 例子 arg_PARAMETER 请求参数 $arg_name http_HEADER 请求头 $http_referer sent_http_HEADER 响应头 sent_http_cookie

7.4 内置变量

ngx_http_log_module (http://nginx.org/en/docs/http/ngx_http_log_module.html) log_format (http://nginx.org/en/docs/http/ngx_http_log_module.html#log_format)

名称 含义 $remote_addr 客户端地址 $remote_user 客户端用户名称 $time_local 访问时间和时区 $request 请求的URI和HTTP协议 $http_host 请求地址，即浏览器中你输入的地址（IP或域名） $status HTTP请求状态 $body_bytes_sent 发送给客户端文件内容大小

8. nginx实战

8.1 静态资源Web服务

- 静态资源：一般客户端发送请求到web服务器，web服务器从内存在取到相应的文件，返回给客户端，客户端解析并渲染显示出来。
- 动态资源：一般客户端请求的动态资源，先将请求交于web容器，web容器连接数据库，数据库处理数据之后，将内容交给web服务器，web服务器返回给客户端解析渲染处理。

类型 种类 浏览器渲染 HTML、CSS、JS 图片 JPEG、GIF、PNG 视频 FLV、MPEG 下载文件 Word、Excel

8.2 CDN

- CDN的全称是Content Delivery Network，即内容分发网络。
- CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容，解决 Internet网络拥挤的状况，提高用户访问网站的响应速度。

cdn (http://img.zhufengpeixun.cn/cdn.jpg)

8.2.1 配置语法

8.2.1.1 sendfile

不经过用户内核发送文件

类型 种类 语法 sendfile on / off 默认 sendfile off; 上下文 http,server,location,if in location

8.2.1.2 tcp_nopush

在sendfile开启的情况 下，提高网络包的传输效率

类型 种类 语法 tcp_nopush on / off 默认 tcp_nopush off; 上下文 http,server,location

8.2.1.2 tcp_nodelay

在keepalive连接下，提高网络包的传输实时性

类型 种类 语法 tcp_nodelay on / off 默认 tcp_nodelay on; 上下文 http,server,location

8.2.1.3 gzip

压缩文件可以节约带宽和提高网络传输效率

类型 种类 语法 gzip on / off 默认 gzip off; 上下文 http,server,location

8.2.1.4 gzip_comp_level

压缩比率越高，文件被压缩的体积越小

类型 种类 语法 gzip_comp_level level 默认 gzip_comp_level 1; 上下文 http,server,location

8.2.1.5 gzip_http_version

压缩HTTP版本

类型 种类 语法 gzip_http_version 1.0/1.1 默认 gzip_http_version 1.1; 上下文 http,server,location

8.2.1.6 http_gzip-static_module

先找磁盘上找同名的 .gz这个文件是否存在,节约CPU的压缩时间和性能损耗

类型 种类 语法 gzip_static on/off 默认 gzip_static off; 上下文 http,server,location

```
location ~ .*\.(jpg|png|gif)$ {
       gzip off;
       gzip_http_version 1.1;
      gzip_comp_level 3;
      gzip_types image/jpeg image/png image/gif;
       root /data/images;
   }

   location ~ .*\.(html|js|css)$ {
       gzip on;
       gzip_min_length 1k;
       gzip_http_version 1.1;
       gzip_comp_level 9;
       gzip_types  text/css application/javascript;
       root /data/html;
   }

   location ~ ^/download {
       gzip_static on;
       tcp_nopush on;
       root /data/download;
   }
```

8.3 浏览器缓存

校验本地缓存是否过期

类型 种类 检验是否过期 Expires、Cache-Control(max-age) Etag Etag Last-Modified Last-Modified

8.3.1 expires

添加Cache-Control、Expires头

类型 种类 语法 expires time 默认 expires off; 上下文 http,server,location

```
location ~ .*\.(jpg|png|gif)$ {
        expires 24h;
}
```

8.4 跨域

类型 种类 语法 add_header name value 默认 add_header --; 上下文 http,server,location

```
location ~ .*\.json$ {
        add_header Access-Control-Allow-Origin http:
        add_header Access-Control-Allow-Methods GET,POST,PUT,DELETE,OPTIONS;
        root /data/json;
    }
```

```
let xhr = new XMLHttpRequest();
        xhr.open('GET', 'http://47.104.184.134/users.json', true);
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                console.log(xhr.responseText);
            }
        }
xhr.send();
```

8.5 防盗链

- 防止网站资源被盗用
- 保证信息安全
- 防止流量过量
- 区别哪些请求是非正常的用户请求
- 使用http_refer防盗链

类型 种类 语法 valid_referers none block server_names string... 默认 - 上下文 server,location

```
location ~ .*\.(jpg|png|gif)$ {
        expires 1h;
        gzip off;
        gzip_http_version 1.1;
        gzip_comp_level 3;
        gzip_types image/jpeg image/png image/gif;
+        valid_referers none blocked 47.104.184.134;
+        if ($invalid_referer) {
+           return 403;
+        }
        root /data/images;
    }
```

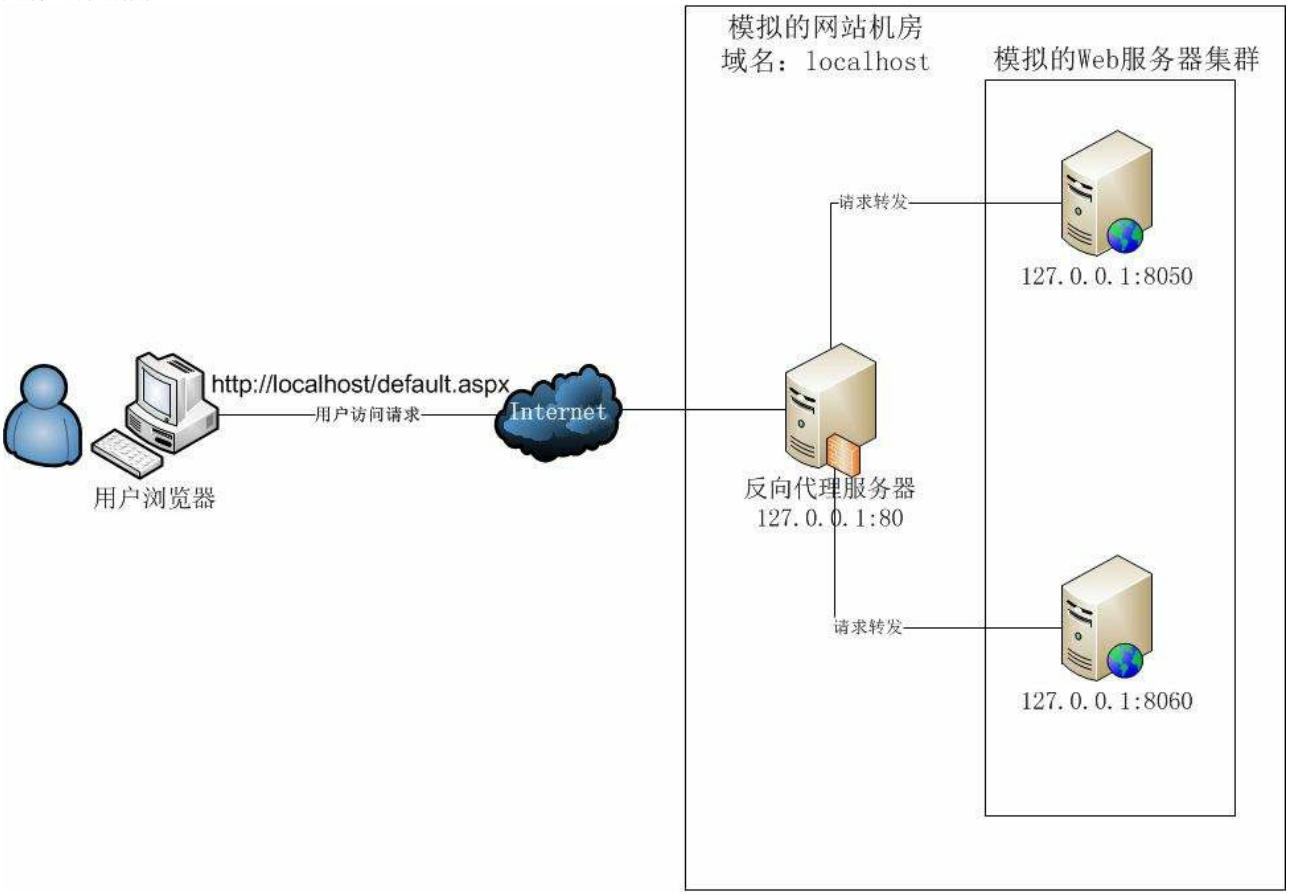
8.6 代理服务

8.6.1 配置

类型 种类 语法 proxy_pass URL 默认 - 上下文 server,location

8.6.2 反向代理

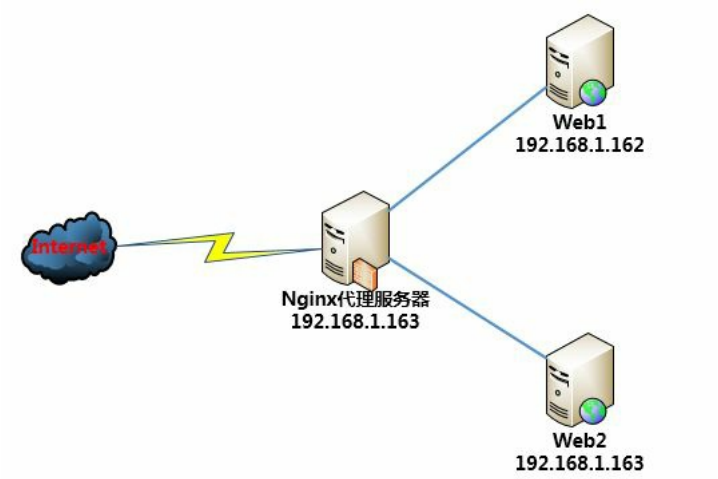
反向代理的对象的服务端




```
resolver 8.8.8.8;
location ~ ^/api {
      proxy_pass http://127.0.0.1:3000;
}
```

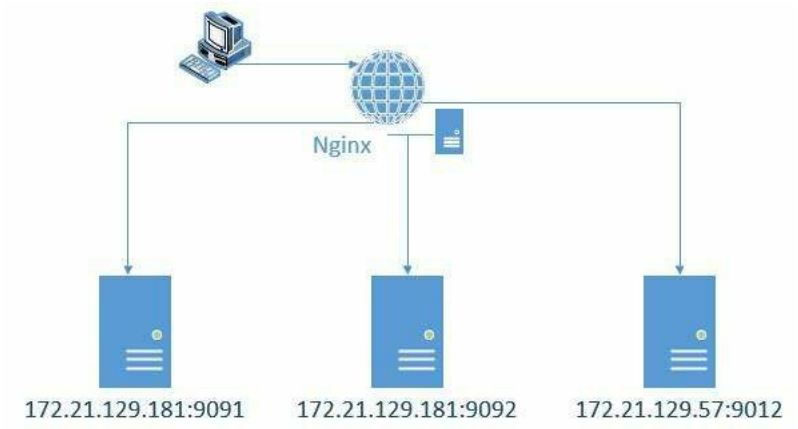
8.6.3 正向代理 #

正向代理的对象是客户端




```
location / {
       proxy_pass http:
   }
```

8.7 负载均衡 #



- 使用集群是网站解决高并发、海量数据问题的常用手段。
- 当一台服务器的处理能力、存储空间不足时，不要企图去换更强大的服务器，对大型网站而言，不管多么强大的服务器，都满足不了网站持续增长的业务需求。
- 这种情况下，更恰当的做法是增加一台服务器分担原有服务器的访问及存储压力。通过负载均衡调度服务器，将来自浏览器的访问请求分发到应用服务器集群中的任何一台服务器上，如果有更多的用户，就在集群中加入更多的应用服务器，使应用服务器的负载压力不再成为整个网站的瓶颈。

8.7.1 upstream

类型 种类 语法 upstream name {} 默认 - 上下文 http

```
upstream zfpx {
  ip_hash;
  server localhost:3000;
  server localhost:4000;
  server localhost:5000;
}

server {
    location / {
        proxy_pass http:
    }
```

8.7.2 后端服务器调试状态

状态 描述 down 不参与负载均衡 backup 备份的服务器 max_fails 允许请求失败的次数 fail_timeout 经过max_fails失败后，服务暂停的时间 max_conns 限制最大的接收的连接数

```
upstream zfpx {
  server localhost:3000 down;
  server localhost:4000 backup;
  server localhost:5000 max_fails=1 fail_timeout=10s;
}
```

8.7.3 分配方式

类型 种类 轮询（默认） 每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。 weight(加权轮询) 指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。 ip_hash 每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。 url_hash（第三方） 按访问的URL地址来分配 请求，每个URL都定向到同一个后端 服务器上(缓存) fair（第三方） 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。 least_conn 最小连接数，哪个连接少就分给谁 自定义hash hash自定义key

8.8 缓存

nginx代理缓存

proxy_cache (https://blog.csdn.net/dengjiexian123/article/details/53386586)

```
http{
    proxy_cache_path /data/nginx/tmp-test levels=1:2 keys_zone=tmp-test:100m inactive=7d max_size=1000g;
}
```

- proxy_cache_path 缓存文件路径
- levels 设置缓存文件目录层次；levels=1:2 表示两级目录
- keys_zone 设置缓存名字和共享内存大小
- inactive 在指定时间内没人访问则被删除
- max_size 最大缓存空间，如果缓存空间满，默认覆盖掉缓存时间最长的资源。

```
location /tmp-test/ {
  proxy_cache tmp-test;
  proxy_cache_valid  200 206 304 301 302 10d;
  proxy_cache_key $uri;
  proxy_set_header Host $host:$server_port;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_pass http:
}
```

- proxy_cache tmp-test 使用名为tmp-test的对应缓存配置
- proxy_cache_valid 200 206 304 301 302 10d; 对httpcode为200...的缓存10天
- proxy_cache_key $uri 定义缓存唯一key,通过唯一key来进行hash存取
- proxy_set_header 自定义http header头，用于发送给后端真实服务器。
- proxy_pass 指代理后转发的路径，注意是否需要最后的/