**Nginx静态服务**

03.Nginx静态服务

1.静态资源类型

2.静态资源场景

3.静态资源配置语法

4.静态资源文件压缩

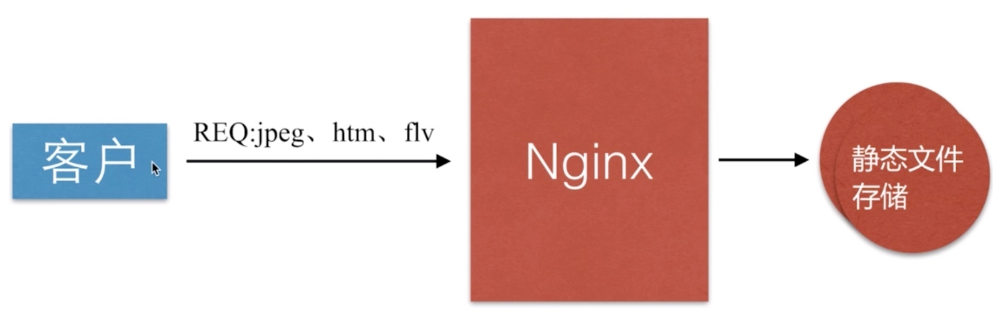
5.静态资源浏览器缓存

6.静态资源跨域访问

7.静态资源防盗链

**1.静态资源类型**

Nginx作为静态资源web服务器部署配置，传输非常的高效，常常用于静态资源处理，请求，动静分离

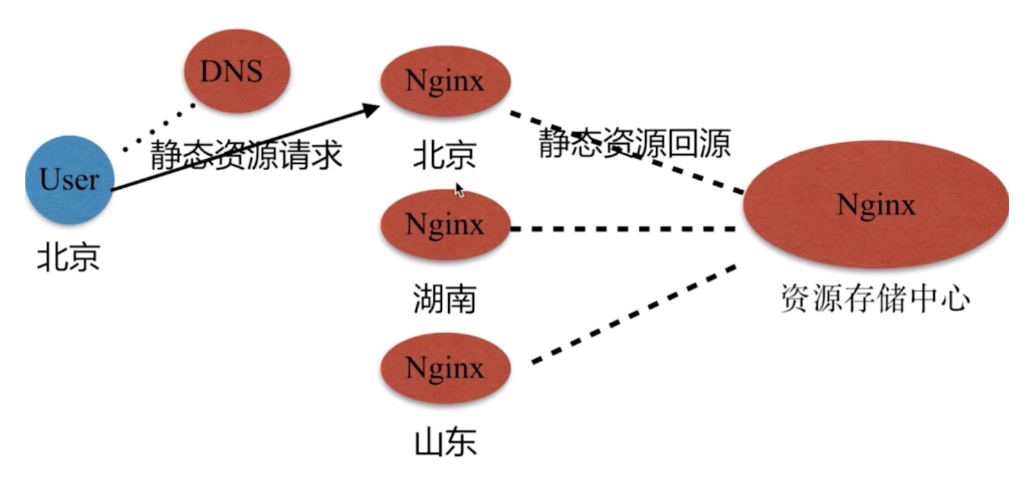


非服务器动态生成的文件属于静态资源

|  |  |
| --- | --- |
| **类型** | **种类** |
| 浏览器端渲染 | HTML、CSS、JS |
| 图片 | JPEG、GIF、PNG |
| 视频 | FLV、Mp4 |
| 文件 | TXT、任意下载文件 |

**2.静态资源场景**

静态资源传输延迟最小化



**3.静态资源配置语法**

1.文件读取高效sendfile

Syntax: sendfile on | off;

Default: sendfile off;

Context: http, server, location, if in location

2.提高网络传输效率nopush

Syntax: tcp\_nopush on | off;

Default: tcp\_nopush off;

Context: http, server, location,

作用： sendfile开启情况下，提高网络包的‘传输效率’

2.与tcp\_nopush之对应的配置 tcp\_nodelay

Syntax: tcp\_ nodelay on | off;

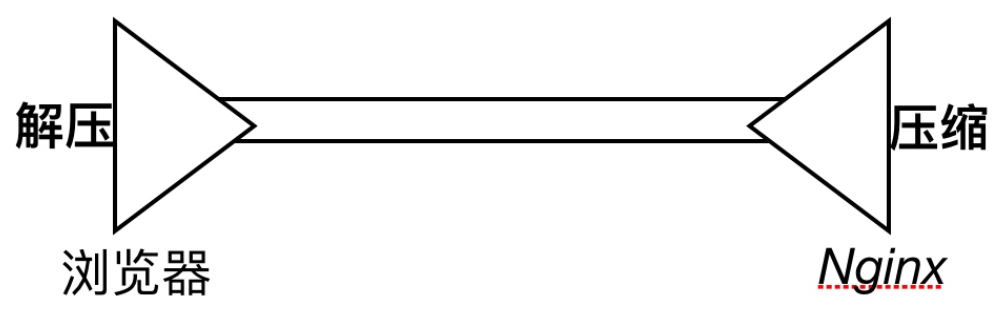
Default: tcp\_ nodelay off;

Context: http, server, location,

作用：在keepalive连接下，提高网络的传输‘实时性’

**4.静态资源文件压缩**

Nginx将响应报文发送至客户端之前可以启用压缩功能，这能够有效地节约宽带，并提高响应至客户端的速度。



1. gzip压缩配置语法

Syntax: gzip on | off;

Default: gzip off;

Context: http, server, location, if in location

作用： 传输压缩

1. gzip压缩比率配置语法

Syntax: gzip\_comp\_level level;

Default: gzip\_comp\_level 1;

Context: http, server, location

作用：压缩本身比较耗费服务端性能

1. gzip压缩协议版本

Syntax: gzip\_http\_version 1.0 | 1.1;

Default: gzip\_http\_version 1.1;

Context: http, server, location

作用： 压缩使用在http那个协议，主流版本1.1

1. 扩展压缩模块

Syntax: gzip\_static on | off | always;

Default: gzip\_static off;

Context: http, server, location

作用：预读gzip功能

1. 图片压缩案例

[root@Nginx conf.d]# mkdir -p /soft/code/images

[root@Nginx conf.d]# cat static\_server.conf

server {

listen 80;

server\_name 192.168.56.11;

sendfile on;

access\_log /var/log/nginx/static\_access.log main;

location ~ .\*\.(jpg|gif|png)$ {

**gzip** on;

**gzip\_http\_Version** 1.1;

**gzip\_comp\_level** 2;

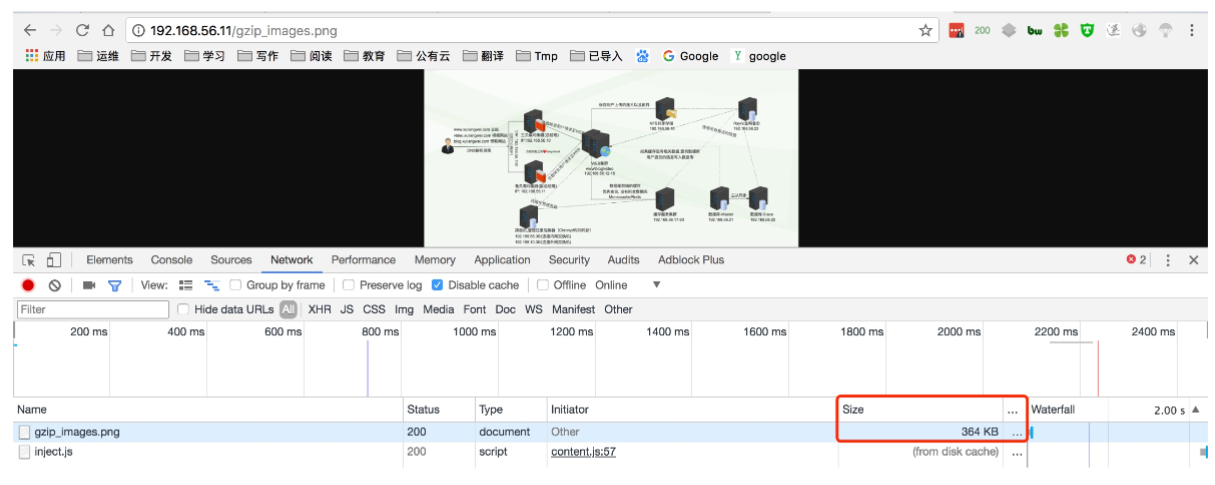
**gzip\_types text** /plain application/json application/x-javascript application/css application/xml application/xml+rss text/javascript application/x-http d-php image/jpeg image/gif image/png

root /soft/code/images;

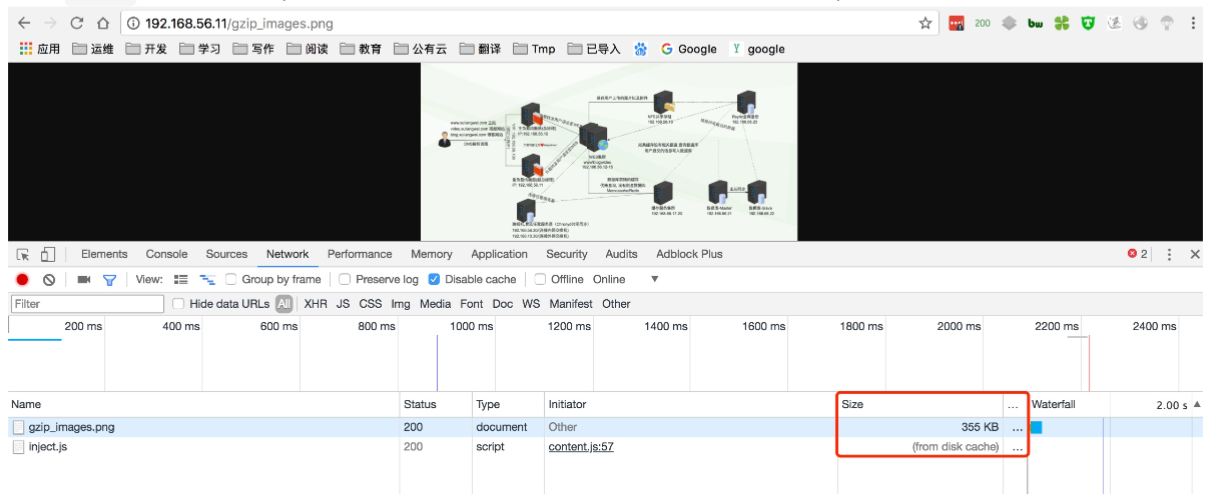
}

}

没有开启gzip图⽚片压缩



启用gzip压缩图片后（由于图片之前压缩过，所以压缩比率不太明显）



1. 文件压缩案例

发

[root@Nginx conf.d]# mkdir -p /soft/code/doc

[root@Nginx conf.d]# cat static\_server.conf

server {

listen 80;

server\_name 192.168.56.11;

sendfile on;

access\_log /var/log/nginx/static\_access.log main;

location ~ .\*\.(txt | xml )$ {

gzip on;

gzip\_http\_version 1.1;

gzip\_comp\_level 1;

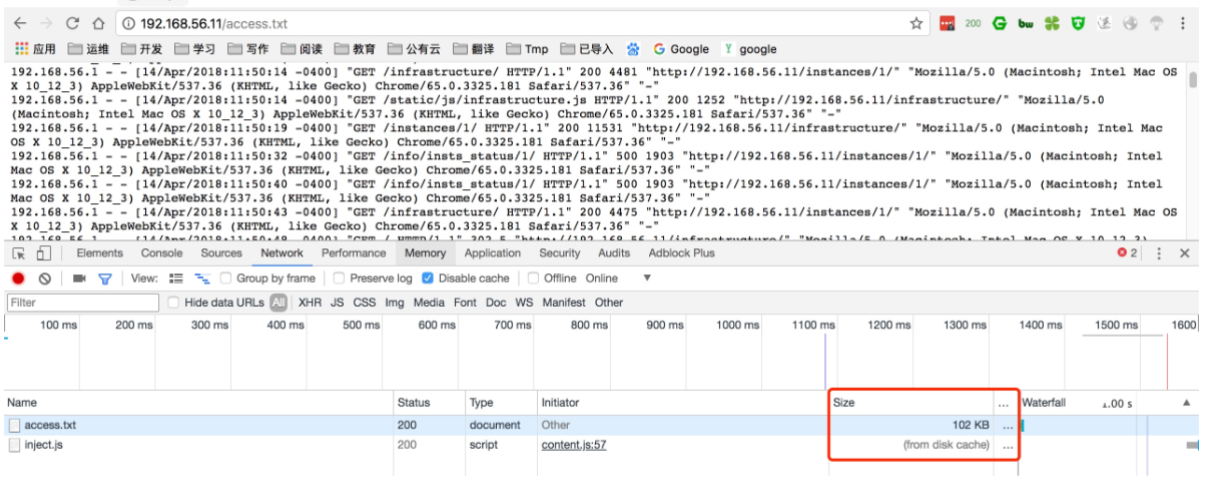
gzip\_types text/plain application/json application/x-javascript application/css application/xml application/xml+rss text/javascript application/x-httpd-php image/jpeg image/gif image/png;

root /soft/code/doc;

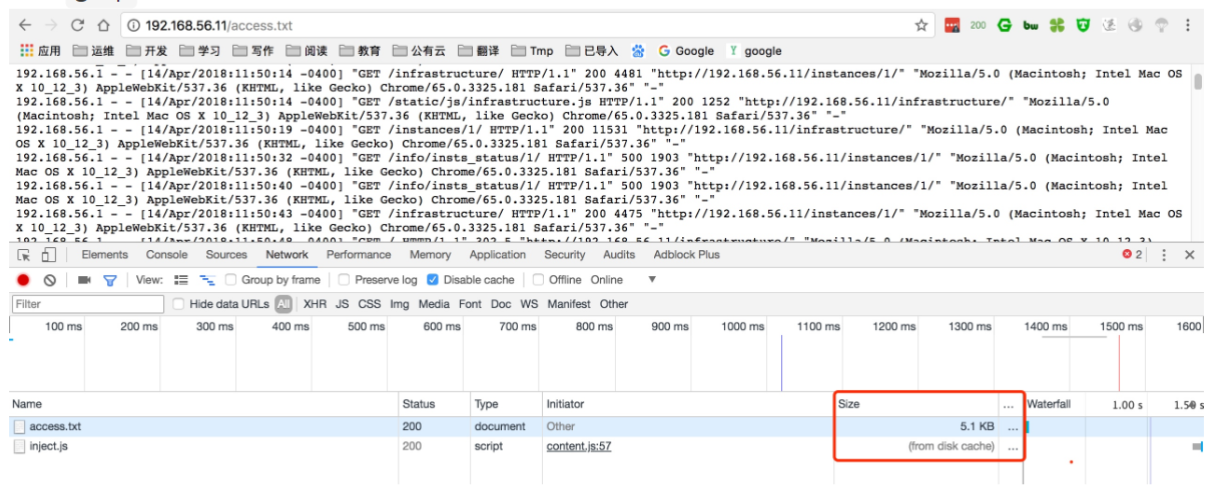
}

}

没有启用gzip文件压缩



启用gzip压缩文件



**5.静态资源浏览器缓存**

HTTP协议定义的缓存机制（如:Expires; Cache-control等）

1. 浏览器无缓存

浏览器请求 ->无缓冲 -> 请求WEB服务器 -> 请求响应 -> 呈现

1. 浏览器有缓存

浏览器请求 -> 有缓存 -> 校验过期 -> 是否有更新 ->呈现

校验是否过期 Expires HTTP1.0, Cache-Control(max-age) HTTP1.1

协议中Etag头信息校验Etag()

Last-Modified 头信息校验Last-Modified(具体时间)

1. 缓存配置语法expires

Syntax: expires [modified] time;

expires epoch | max | off;

Default: expires off;

Context: http, server, location, If in location

作⽤用: 添加Cache-Control Expires头

2.配置静态资源缓存

Location ~ .\*\.(js|css|html)$ {

root /soft/code/js;

expires 1h;

}

location ~ .\*\.(jpg|gif|png)$ {

root /soft/code/images;

expires 7d;

}

3.开发代码没有正式上线时, 希望静态⽂文件不不被缓存

//取消js css html等静态⽂文件缓存

location ~ .\*\.(css| js | swf | json | mp4 | htm | html)$ {

add\_header Cache-Control no-store;

add\_header Pragma no-cache;

}

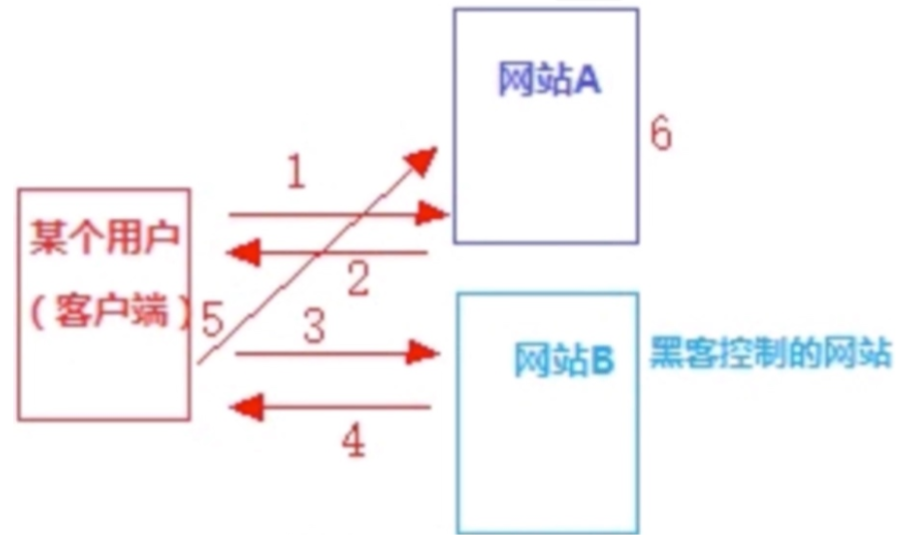
阿⾥里里云缓存策略略帮助⼿手册

Nginx静态资源缓存

1. **静态资源跨域访问**



浏览器禁止跨域访问，主要不安全，容易出现CSRE攻击



Nginx跨域访问配置

Syntax: add\_header name value [always];

Default: —

Context: http, server, location, if in location

Access-Control-Allow-Origin

1.准备html⽂文件

//在www.xuliangwei.com⽹网站添加跨越访问⽂文件

[root@Nginx ~]# cat /soft/code/http\_origin.html

<html lang="en">

<head>

<meta charset="UTF-8" />  
<title>测试ajax和跨域访问</title>

<script src="http://libs.baidu.com/jquery/2.1.4/jquery.min.js"></script>

</head>

<script type="text/javascript">

$(document).ready(function(){

$.ajax({

type: "GET",

url: "http://kt.xuliangwei.com/index.html",

success: function(data) {

alert("sucess!!!");

},

error: function() {

alert("fail!!,请刷新再试!");

}

});

});

</script>

<body>

<h1>测试跨域访问</h1>

</body>

</html>

2.配置Nginx跨域访问

//运⾏行行www.xuliangwei.com域名跨域访问

[root@Nginx conf.d]# cat origin.conf

server {

listen 80;

server\_name kt.xuliangwei.com;

sendfile on;

access\_log /var/log/nginx/kuayue.log main;

location ~ .\*\.(html|htm)$ {

add\_header Access-Control-Allow-Origin http://www.xuliangwei.com;

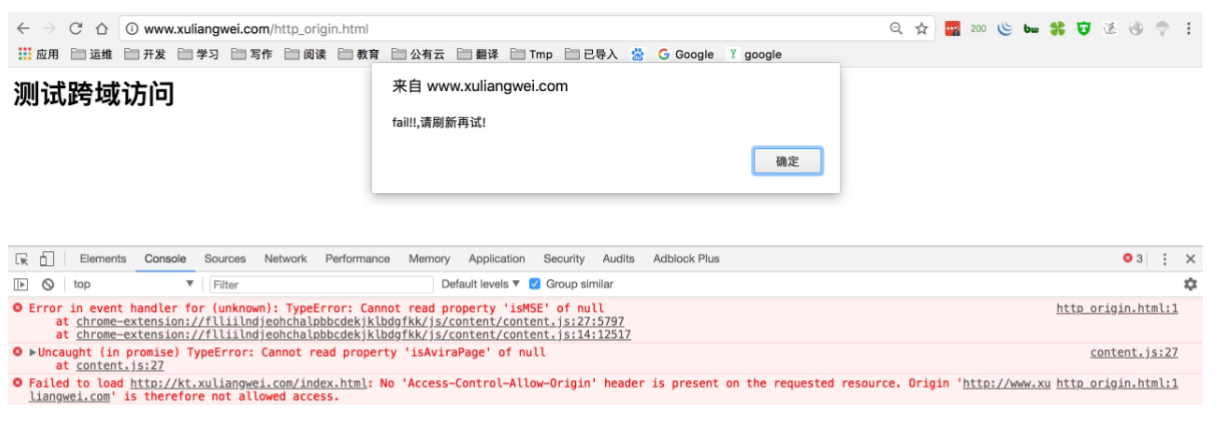
add\_header Access-Control-Allow-methodsGET,POST,PUT,DELETE,OPTIONS;

root /soft/code;

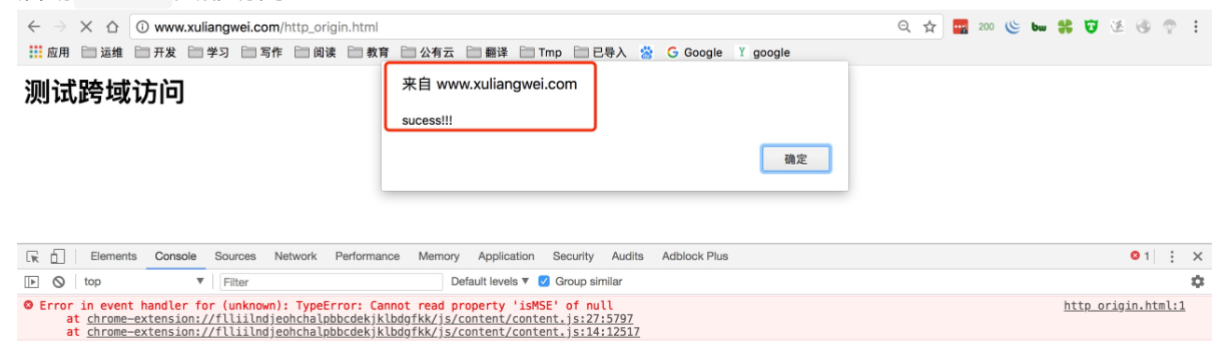
}

}

没启动header头部访问



启动header头部访问



**7.静态资源防盗链**

盗链指的是在⾃自⼰己的界⾯面展示不不在⾃自⼰己服务器器上的内容，通过技术⼿手段获得他⼈人服务器器的资源地址，绕过别⼈人资源展示⻚页面，在⾃自⼰己⻚页⾯面向⽤用户提供此内容，从⽽而减轻⾃自⼰己服务器器的负担，因为真实的空间和流量量来⾃别⼈人服务器器

防盗链设置思路路: 区别哪些请求是⾮非正常⽤用户请求

基于http\_refer防盗链配置模块

Syntax: valid\_referers none | blocked | server\_names | string ...;

Default: —

Context: server, location

**1.准备html⽂文件**

<html>

<head>

<meta charset="utf-8">

<title>pachong<title>

</head>

<body style="background-color:red;">

<img src="http://192.168.69.113/test.jpg">

</body>

</html>

**2.启动防盗链**

//⽀支持IP、域名、正则⽅方式

location ~ .\*\.(jpg|gif|png)$ {

valid\_referers none blocked www.xuliangwei.com;

if ($invalid\_referer) {

return 403;

}

root /soft/code/images;

}

**3.验证**

//伪造协议头访问

[root@C-Server ~]# curl -e "http://www.baidu.com" -I http://192.168.69.113/test.jpg

HTTP/1.1403 Forbidden

Server: nginx/1.12.2

Date: Tue, 17 Apr 2018 04:55:18 GMT

Content-Type: text/html

Content-Length: 169

Connection: keep-alive

//伪造协议头访问

[root@C-Server ~]# curl -e "http://www.xuliangwei.com" -I http://192.168.69.113/test.jpg

HTTP/1.1 200 OK

Server: nginx/1.12.2

Date: Tue, 17 Apr 2018 04:55:27 GMT

Content-Type: image/jpeg

Content-Length: 174315

Last-Modified: Wed, 29 Nov 2017 03:16:08 GMT

Connection: keep-alive

ETag: "5a1e2678-2a8eb"

Expires: Tue, 17 Apr 2018 16:55:27 GMT

Cache-Control: max-age=43200

Accept-Ranges: bytes