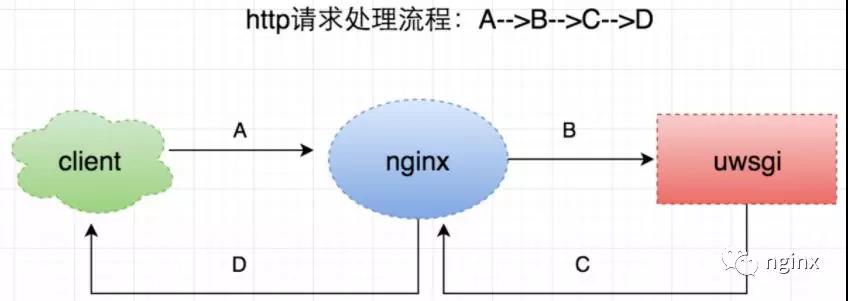
**[Nginx code 状态码说明](https://www.cnblogs.com/kerwing/p/9072268.html)**

最近了解下Nginx的Code状态码，在此简单总结下。

先来再回顾一下一个http请求处理流程：



一个普通的http请求处理流程，如上图所示： A -> client端发起请求给nginx B -> nginx处理后，将请求转发到uwsgi，并等待结果 C -> uwsgi处理完请求后，返回数据给nginx D -> nginx将处理结果返回给客户端 每个阶段都会有一个预设的超时时间，由于网络、机器负载、代码异常等等各种原因，如果某个阶段没有在预期的时间内正常返回，就会导致这次请求异常，进而产生不同的状态码。

1）504

504主要是针对B、C阶段。一般nginx配置中会有：

location / {

    ...

    uwsgi\_connect\_timeout 6s;

    uwsgi\_send\_timeout 6s;

    uwsgi\_read\_timeout 10s;

    uwsgi\_buffering on;

    uwsgi\_buffers 80 16k;

    ...

}

这个代表nginx与上游服务器（uwsgi）通信的超时时间，也就是说，如果在这个时间内，uwsgi没有响应，则认为这次请求超时，返回504状态码。

具体的日志如下

access\_log[16/May/2016:22:11:38 +0800] 10.4.31.56 201605162211280100040310561523 15231401463407888908 10.\*.\*.\* 127.0.0.1:8500 "GET /api/media\_article\_list/?count=10&source\_type=0&status=all&from\_time=0&item\_id=0&flag=2&\_=1463407896337 HTTP/1.1" 504 \*\*.\*\*\*.com \*\*.\*\*.\*\*.39, \*\*.\*\*.\*\*.60 10.000 10.000 "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36" ...error\_log2016/05/16 22:11:38 [error] 90674#0: \*947302032 upstream timed out (110: Connection timed out) while reading response header from upstream, client: 10.6.19.81, server: \*\*.\*\*\*.com, request: "GET /api/media\_article\_list/?count=10&source\_type=0&status=all&from\_time=0&item\_id=0&flag=2&\_=1463407896337 HTTP/1.1", upstream: "http://127.0.0.1:8500/\*\*/\*\*/api/media\_article\_list/?count=10&source\_type=0&status=all&from\_time=0&item\_id=0&flag=2&\_=1463407896337", host: "mp.toutiao.com", referrer: "https://\*\*.\*\*\*.com/articles/?source\_type=0"error\_log中upstream timed out (110: Connection timed out) while reading response header from upstream，

意思是说，在规定的时间内，没有从header中拿到数据，即uwsgi没有返回任何数据。

2）502

502主要针对B 、C阶段。产生502的时候，对应的error\_log中的内容会有好几种：

access\_log  
[16/May/2016:16:39:49 +0800] 10.4.31.56 201605161639490100040310562612 2612221463387989972 10.6.19.81 127.0.0.1:88 "GET /articles/?source\_type=0 HTTP/1.1" 503 \*\*.\*\*\*.com \*\*.\*\*.\*\*.4, \*\*.\*\*.\*\*.160 0.000 0.000 "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36" "uuid=\x22w:546d345b86ca443eb44bd9bb1120e821\x22; tt\_webid=15660522398; lasttag=news\_culture; sessionid=f172028cc8310ba7f503adb5957eb3ea; sid\_tt=f172028cc8310ba7f503adb5957eb3ea; \_ga=GA1.2.354066248.1463056713; \_gat=1"

error\_log

2016/05/16 16:39:49 [error] 90693#0: \*944980723 recv() failed (104: Connection reset by peer) while reading response header from upstream, client: 10.6.19.80, server: \*\*.\*\*\*.com, request: "GET /articles/ HTTP/1.1", upstream: "http://127.0.0.1:8500/\*\*/\*\*/articles/", host: "\*\*.\*\*\*.com", referrer: [http://\*\*.\*\*\*.com/new\_article/](http://**.***.com/new_article/)

列一下常见的几种502对应的 error\_log：

* recv() failed (104: Connection reset by peer) while reading response header from upstream
* upstream prematurely closed connection while reading response header from upstream
* connect() failed (111: Connection refused) while connecting to upstream
* ....

这些都代表，在nginx设置的超时时间内，上游uwsgi没有给正确的响应（但是是有响应的，不然如果一直没响应，就会变成504超时了），因此nginx这边的状态码为502。

如上，access\_log中出现503，为什么？

这个是因为nginx upstream的容灾机制。如果nginx有如下配置：

upstream app\_backup {

    server 127.0.0.1:8500 max\_fails=3 fail\_timeout=5s;

    server 127.0.0.1:88 backup;

}

* max\_fails=3 说明尝试3次后，会认为“ server 127.0.0.1:8500” 失效，于是进入 “server 127.0.0.1:88 backup”，即访问本机的88端口;

nginx upstream的容灾机制，默认情况下，Nginx 默认判断失败节点状态以connect refuse和time out状态为准，不过location里加了这个配置：

proxy\_next\_upstream error http\_502;

proxy\_connect\_timeout 1s;

proxy\_send\_timeout    6s;

proxy\_read\_timeout    10s;

proxy\_set\_header Host $host;

* 这个配置是说，对于http状态是502的情况，也会走upstream的容灾机制；
* 概括一下就是，如果连续有3次(max\_fails=3)状态为502的请求，则会任务这个后端server 127.0.0.1:8500 挂掉了，在接下来的5s(fail\_timeout=5s)内，就会访问backup，即server 127.0.0.1:88 ，看下88端口对应的是什么：

server {

 listen 88;

 access\_log /var/log/nginx/failover.log;

 expires 1m;

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

 location / {

     return 503;

 }

 location = /500.html {

     root /\*\*/\*\*/\*\*/nginx/5xx/;

 }

}

这个的意思就是，对于访问88端口的请求，nginx会返回503状态码，同时返回/opt/tiger/ss\_conf/nginx/5xx/这个路径下的500.html文件。 因此，access\_log中看到的是503

3）499

client发送请求后，如果在规定的时间内（假设超时时间为500ms）没有拿到nginx给的响应，则认为这次请求超时，会主动结束，这个时候nginx的access\_log就会打印499状态码。 A+B+C+D > 500ms 其实这个时候，server端有可能还在处理请求，只不过client断掉了连接，因此处理结果也无法返回给客户端。 499如果比较多的话，可能会引起服务雪崩。 比如说，client一直在发起请求，客户端因为某些原因处理慢了，没有在规定时间内返回数据，client认为请求失败，中断这次请求，然后再重新发起请求。这样不断的重复，服务端的请求越来越多，机器负载变大，请求处理越来越慢，没有办法响应任何请求

官网总结nginx返回499的情况，是由于：

client has closed connection    #客户端主动关闭了连接。

解决的话，可以添加

proxy\_ignore\_client\_abort    on;

还有一种原因，确实是客户端关闭了连接，或者连接超时。主要是因为PHP进程数太少，或php进程占用，资源不能很快释放，请求堆积。这种情况要解决的话，需要在程序上做优化。

4）500

服务器内部错误，也就是服务器遇到意外情况，而无法执行请求。发生错误，一般的几种情况：

* web脚本错误，如php语法错误，lua语法错误等。
* 访问量大的时候，由于系统资源限制，而不能打开过多的文件句柄

分析错误的原因

* 查看nginx，php的错误日志
* 如果是too many open files，修改nginx的worker\_rlimit\_nofile参数，使用ulimit查看系统打开文件限制，修改/etc/security/limits.conf
* 如果脚本存在问题，则需要修复脚本错误，并优化代码
* 各种优化都做好，还是出现too many open files，那就需要考虑做负载均衡，把流量分散到不同服务器上去

5）503

503是服务不可用的返回状态。 由于在nginx配置中，设置了limit\_req的流量限制，导致许多请求返回503错误代码，在限流的条件下，为提高用户体验，希望返回正常Code 200，且返回操作频繁的信息：

................................................Nginx Code Status...............................

200：服务器成功返回网页

403：服务器拒绝请求。404：请求的网页不存在

499：客户端主动断开了连接。500：服务器遇到错误，无法完成请求。502：服务器作为网关或代理，从上游服务器收到无效响应。503 - 服务不可用

504：服务器作为网关或代理，但是没有及时从上游服务器收到请求。

这些状态码被分为五大类：

100-199 用于指定客户端应相应的某些动作。

200-299 用于表示请求成功。

300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。

400-499 用于指出客户端的错误。 （自己电脑这边的问题） 自己电脑这边的问题）

500-599 用于支持服务器错误。 （对方的问题） 对方的问题）---------------------------------------------------------------------------------------------200 （成功） 服务器已成功处理了请求。 通常，这表示服务器提供了请求的网页。

201 （已创建） 请求成功并且服务器创建了新的资源。

202 （已接受） 服务器已接受请求，但尚未处理。

203 （非授权信息） 服务器已成功处理了请求，但返回的信息可能来自另一来源。

204 （无内容） 服务器成功处理了请求，但没有返回任何内容。

205 （重置内容） 服务器成功处理了请求，但没有返回任何内容。206 （部分内容） 服务器成功处理了部分 GET 请求。

---------------------------------------------------------------------------------------------300 （多种选择） 针对请求，服务器可执行多种操作。 服务器可根据请求者 (user agent) 选择一项操作，或提供操作列表供请求者选择。

301 （永久移动） 请求的网页已永久移动到新位置。 服务器返回此响应（对 GET 或 HEAD 请求的响应）时，会自动将请求者转到新位置。302 （临时移动） 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。303 （查看其他位置） 请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码。304 （未修改） 自从上次请求后，请求的网页未修改过。 服务器返回此响应时，不会返回网页内容。

305 （使用代理） 请求者只能使用代理访问请求的网页。 如果服务器返回此响应，还表示请求者应使用代理。

307 （临时重定向） 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

---------------------------------------------------------------------------------------------400 （错误请求） 服务器不理解请求的语法。

401 （未授权） 请求要求身份验证。 对于需要登录的网页，服务器可能返回此响应。

403 （禁止） 服务器拒绝请求。404 （未找到） 服务器找不到请求的网页。405 （方法禁用） 禁用请求中指定的方法。

406 （不接受） 无法使用请求的内容特性响应请求的网页。

407 （需要代理授权） 此状态代码与 401（未授权）类似，但指定请求者应当授权使用代理。408 （请求超时） 服务器等候请求时发生超时。

409 （冲突） 服务器在完成请求时发生冲突。 服务器必须在响应中包含有关冲突的信息。

410 （已删除） 如果请求的资源已永久删除，服务器就会返回此响应。

411 （需要有效长度） 服务器不接受不含有效内容长度标头字段的请求。

412 （未满足前提条件） 服务器未满足请求者在请求中设置的其中一个前提条件。

413 （请求实体过大） 服务器无法处理请求，因为请求实体过大，超出服务器的处理能力。

414 （请求的 URI 过长） 请求的 URI（通常为网址）过长，服务器无法处理。

415 （不支持的媒体类型） 请求的格式不受请求页面的支持。

416 （请求范围不符合要求） 如果页面无法提供请求的范围，则服务器会返回此状态代码。

417 （未满足期望值） 服务器未满足"期望"请求标头字段的要求。

---------------------------------------------------------------------------------------------500 （服务器内部错误） 服务器遇到错误，无法完成请求。

501 （尚未实施） 服务器不具备完成请求的功能。 例如，服务器无法识别请求方法时可能会返回此代码。

502 （错误网关） 服务器作为网关或代理，从上游服务器收到无效响应。

503 （服务不可用） 服务器目前无法使用（由于超载或停机维护）。 通常，这只是暂时状态。

504 （网关超时） 服务器作为网关或代理，但是没有及时从上游服务器收到请求。

505 （HTTP 版本不受支持） 服务器不支持请求中所用的 HTTP 协议版本。

proxy\_intercept\_errors 当上游服务器响应头回来后，可以根据响应状态码的值进行拦截错误处理，与error\_page 指令相互结合。用在访问上游服务器出现错误的情况下。

如下的一个配置实例：

[root@dev ~]# cat ssl-zp.wangshibo.conf

upstream mianshi1 {

    server 192.168.1.33:8080 max\_fails=3 fail\_timeout=10s;

    #server 192.168.1.32:8080 max\_fails=3 fail\_timeout=10s;

}

 server {

    listen 443;

    server\_name zp.wangshibo.com;ssl on;

    ### SSL log files ###

    access\_log logs/zrx\_access.log;

    error\_log logs/zrx\_error.log;

    ### SSL cert files ###

    ssl\_certificate ssl/wangshibo.cer;

    ssl\_certificate\_key ssl/wangshibo.key;ssl\_session\_timeout 5m;

    error\_page 404 301 https://zp.wangshibo.com/zrx-web/;

    location /zrx-web/ {

        proxy\_pass http://mianshi1;

        proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503;

        proxy\_set\_header Host $host;

        proxy\_set\_header X-Real-IP $remote\_addr;proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

        # proxy\_set\_header X-Forwarded-Proto https;

        #proxy\_set\_header X-Forwarded-Proto https;proxy\_redirect off;

        proxy\_intercept\_errors on;

    }

}

<https://www.cnblogs.com/kerwing/p/9072268.html>