**nginx负载均衡以及静态资源的反向代理实施方案**

作者：袁旭云

目录

[nginx介绍 2](#_Toc460494290)

[本文说明 3](#_Toc460494291)

[背景说明 4](#_Toc460494292)

[初级负载均衡 4](#_Toc460494293)

[nginx相关配置 4](#_Toc460494294)

[1、下载地址 4](#_Toc460494295)

[2、目录结构 5](#_Toc460494296)

[3、nginx.conf配置 6](#_Toc460494297)

[4、Tomcat配置 7](#_Toc460494298)

[5、验证配置与测试负载均衡 7](#_Toc460494299)

[6、总结 9](#_Toc460494300)

[中级负载均衡 9](#_Toc460494301)

[目录重定向 10](#_Toc460494302)

[总结 12](#_Toc460494303)

[高级负载均衡 12](#_Toc460494304)

[1、Windows下安装并设置Redis 13](#_Toc460494305)

[2、tomcat的配置 13](#_Toc460494306)

[3、总结 18](#_Toc460494307)

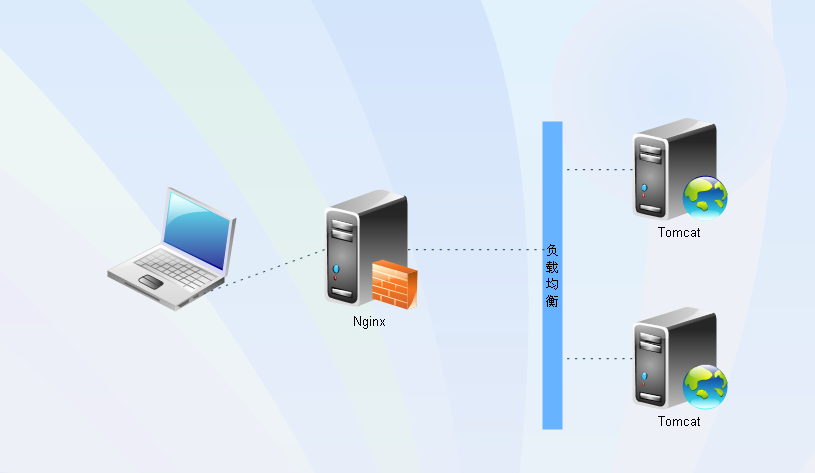
[运行环境展示 18](#_Toc460494308)

## nginx介绍

   nginx （发音同 engine x）是一款轻量级的Web 服务器／反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个BSD-like 协议下发行。  其特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页伺服器中表现较好.目前中国大陆使用nginx网站用户有：新浪、网易、 腾讯,另外知名的微网志Plurk也使用nginx。

## 本文说明

本文所有的配置都基于win7 64位操作系统进行相关配置，不适用与其他类型的操作系统。nginx下配置两台tomcat，结构如下图：



所用到的资源清单如下

nginx/Windows-1.10.1 Windows系统稳定版

Tomcat7 （后面高级部分有session同步机制 所以tomcat需要使用7 ）

jdk7 64位

(不需要session同步时不需要)

Redis-x64-3.2.100

commons-pool2-2.0.jar

jedis-2.5.2.jar

tomcat-redis-session-manager1.2.jar

## 背景说明

无论是高并发还是一般大应用程序，通过nginx来处理静态页面要比通过Tomcat处理在性能方面好很多，所以用nginx处理静态资源理论上能够提升系统性能的。单机部署项目虽然能够节省硬件成本，但是在产品更新迭代的时候由于系统不得不关闭，或者是由于硬件故障造成服务器宕机，多多少少会给用户的使用带来一定的影响。高并发下单机部署服务器压力会更大，这个时候负载均衡显得尤其重要。通过nginx可以实现单服务器的静态资源本地化，也可以实现tomcat集群的负载均衡，配合其他资源还能实现session的共享等等。基于上面这些特性，可以轻松的实现负载均衡，更新迭代能零宕机，做到用户体验和性能的双提升。

## 初级负载均衡

无session同步，无静态文件本地化，只做负载均衡。tomcat X 2 + nginx，jdk、tomcat相关配置及下载自行查阅相关资料，本文不做介绍

### nginx相关配置

### 1、下载地址

http://nginx.org/en/download.html ，这里我们推荐下载稳定版（stable versions）

### 2、目录结构

nginx-

|\_ conf 配置目录

|\_ contrib

|\_ docs 文档目录

|\_ logs 日志目录

|\_ temp 临时文件目录

|\_ html 静态页面目录

|\_ nginx.exe 主程序

window下安装Nginx极其简单，解压缩到一个无空格的英文目录即可（个人习惯，担心中文出问题），双击nginx启动，这里我解压缩到到：C:\service目录。



DOS环境启动

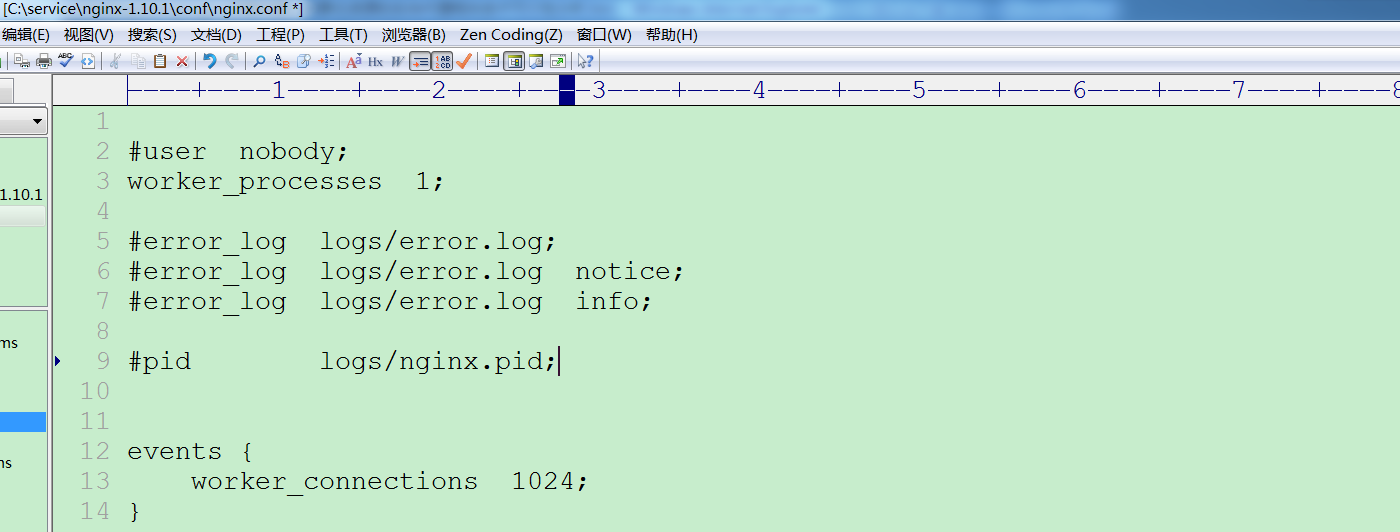


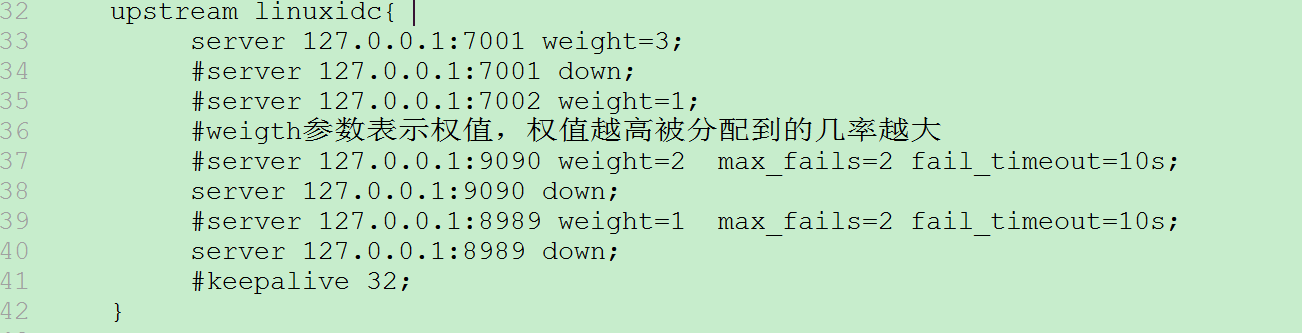
如果想停止nginx，dos环境运行命令：nginx -s stop

重新加载配置 nginx -s reload

### 3、nginx.conf配置

nginx配置文件默认在conf目录，主要配置文件为nginx.conf，下面是nginx作为前端反向代理服务器的配置。





多的就不再截图，文件在这里

### 4、Tomcat配置

对于tomcat大家都很熟悉，只需要修改server.xml配置文件即可，这里我们以apache-tomcat-7.0.70为例，注意：一台服务器开多tomcat需要修改，如果是不在一台服务器，则不用。

第一处端口修改：

<!--  修改port端口：18006 俩个tomcat不能重复，端口随意，别太小-->

<Server port="18006" shutdown="SHUTDOWN">

第二处端口修改：

<!-- port="18081" tomcat监听端口，随意设置，别太小 -->

<Connector port="18081" protocol="HTTP/1.1"

               connectionTimeout="20000"

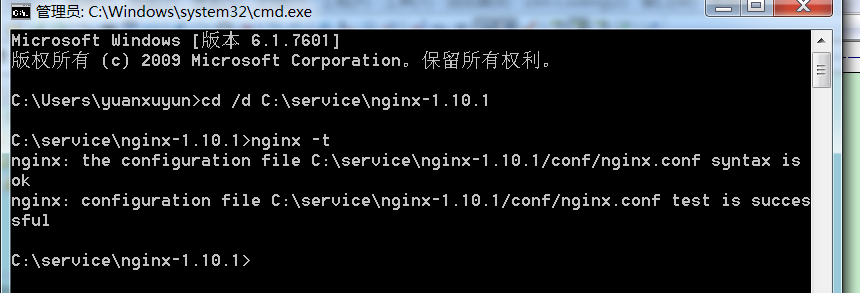
               redirectPort="8443" />

第三处端口修改：

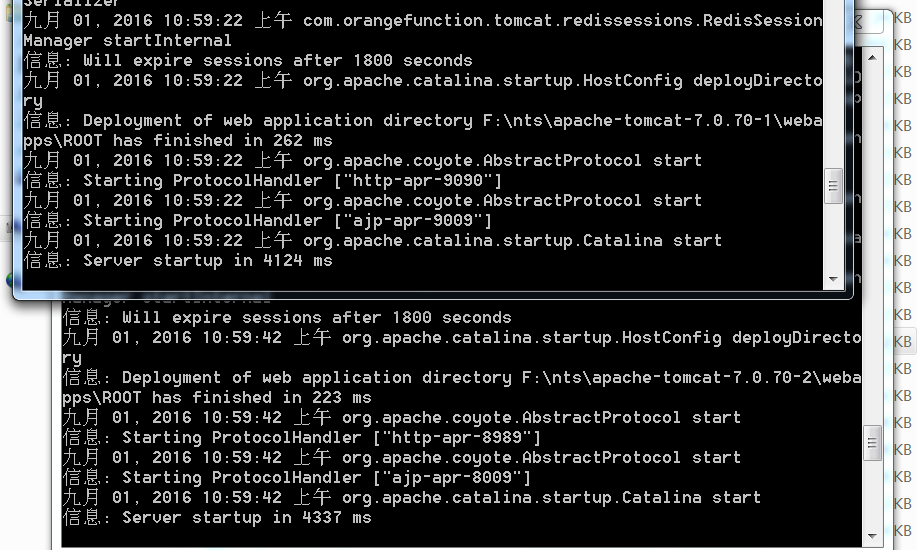
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

### 5、验证配置与测试负载均衡

首先测试nginx配置是否正确，测试命令：nginx -t (默认验证:conf\nginx.conf),也可以指定配置文件路径。



其次验证tomcat，启动两个tomcat，不出现端口冲突即为成功（tomcat依赖的java等搞“挨踢”的就废话不说了）



最后验证配置负载均衡设置，http://localhost/ 或http://localhost/index.jsp 。我修改了index.jsp页面，增加日志输出信息，便于观察。注意：左上角小猫头上的：access tomcat2、access tomcat1。说明访问了不同的tomcat。



至此window下nginx+tomcat负载均衡配置结束，关于tomcat Session的问题通常是采用memcached，或者采用nginx\_upstream\_jvm\_route ，他是一个 Nginx 的扩展模块，用来实现基于 Cookie 的 Session Sticky 的功能。如果tomcat过多不建议session同步，server间相互同步session很耗资源，高并发环境容易引起Session风暴。请根据自己应用情况合理采纳session解决方案。

### 6、总结

采用这种方式，可以随机的访问到某一台服务器，减轻服务器压力，同时在进行程序更新的时候（server 127.0.0.1:7001 down;），也能保证有一台服务器在处于工作状态，对新登录系统的用户不受影响。

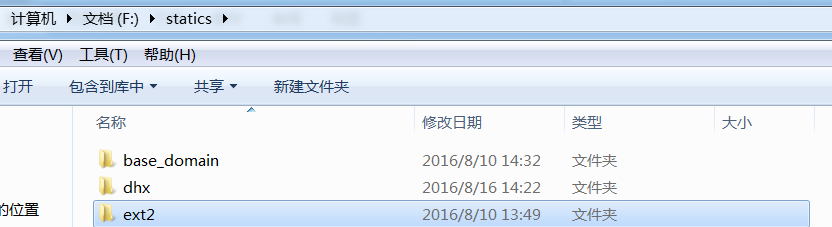
## 中级负载均衡

无session同步，静态文件本地化+负载均衡，Tomcat在高并发环境下处理动态请求时性能很低，而在处理静态页面更加脆弱。虽然Tomcat的最新版本支持epoll，但是通过Nginx来处理静态页面要比通过Tomcat处理在性能方面好很多。

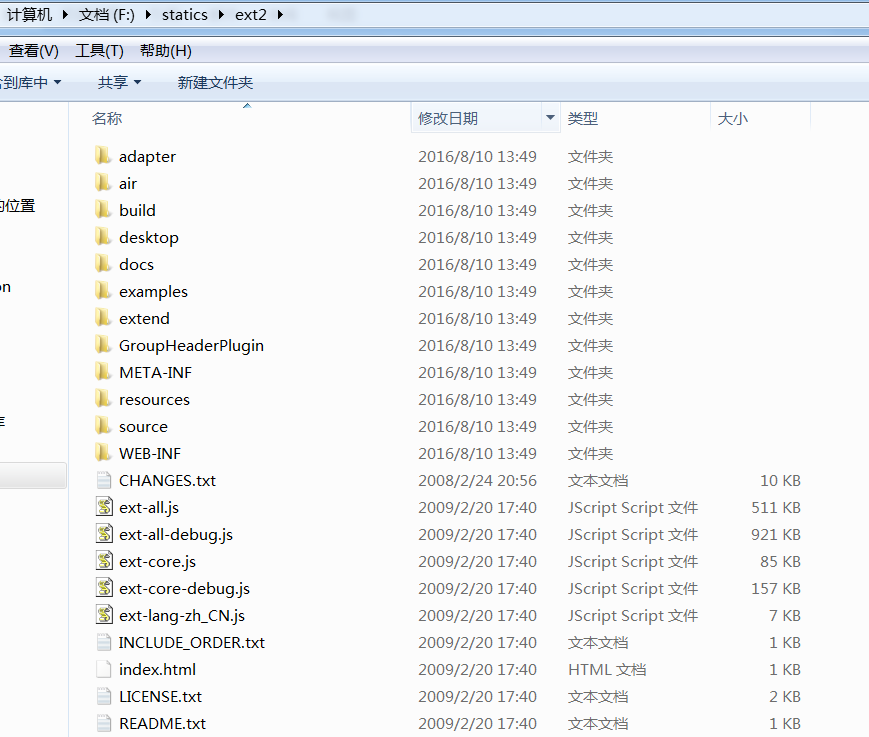
### 目录重定向

假设分析发现系统将ext2部署在同级目录中，ext2文件夹下全部为静态文件；

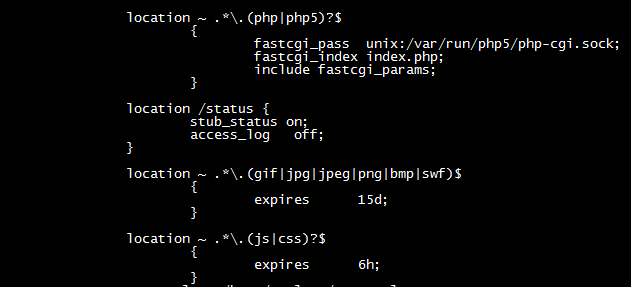
首先拷贝ext2文件夹到磁盘，如图



展开看看 如图



那么如何将以/ext2请求的文件拦截下来呢？这时候就要说一说nginx的url拦截配置了。如图



详细的解释官方文档也有说明，这里就不多说了。

[参考1](http://www.nginx.cn/115.html) [参考2](http://lobert.iteye.com/blog/1933417)

本文配置如下

location ^~ /ext2/ {

root F:/statics;

expires 30d;

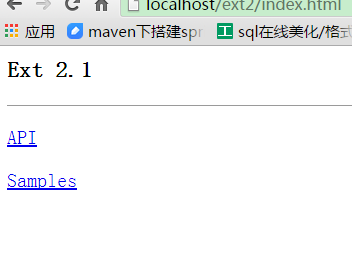
}

这里特别要注意，ext2这个文件夹是一定要存在的 并且要在 statics文件夹里面，这也就是root F:/statics; 的原因了，最终当带有ext2的url就会指向 F:/statics /ext2下面了，如果root F:/statics/ext2，这是不对了，这样会404，这里相信有不少人遇到，重点说下。

在cmd窗口执行（关闭tomcat 试试）

nginx -s reload

访问xxx/ ext2/index.html,看到了吧 成功出现下面的页面如图



看到了没，静态文件成功的指向了本地磁盘；

### 总结

采用这种方式，可以随机的访问到某一台服务器，减轻服务器压力，同时在进行程序更新的时候（server 127.0.0.1:7001 down;），也能保证有一台服务器在处于工作状态，对新登录系统的用户不受影响。同时通过Nginx来处理静态页面要比通过Tomcat处理在性能方面好很多。

## 高级负载均衡

session共享+静态文件本地化+负载均衡，memcached（尝试失败），或者采用nginx\_upstream\_jvm\_route（没有windows版本的），本文采用基于nginx + tomcat redis分布式web应用的session共享配置，这里有记点先说明下，首先我们是要在windows系统下进行这些工作的，前面文档开头就有说的，这里在再说下，win7 64 位，非64位 下面的步骤就不用尝试了，本文中tomcat-redis-session-manager1.2.jar 只能在tomcat7下使用，这个是自己编译好的；

### 1、Windows下安装并设置Redis

Redis对于Linux是官方支持的,安装和使用没有什么好说的,普通使用按照官方指导，5分钟以内就能搞定。详情请参考:<http://redis.io/download> ，但有时候又想在windows下折腾下Redis,可以从redis下载页面看到如下提示(在页面中搜索 "windows")：

[plain] [view plain](http://blog.csdn.net/renfufei/article/details/38474435) [copy](http://blog.csdn.net/renfufei/article/details/38474435)

Win64   Unofficial  The Redis project does not directly support Windows,

    however the Microsoft Open Tech group develops and maintains

    an Windows port targeting Win64.

大意就是 Redis官方是不支持windows的，只是 Microsoft Open Tech group 在 GitHub上开发了一个Win64的版本,项目地址是：

<https://github.com/MSOpenTech/redis>

打开以后，可以直接使用浏览器下载，或者git克隆。

下载地址: <https://github.com/MSOpenTech/redis/releases>

网上参考了一些资料,发觉可以使用,也就没有深究,直接拿来主义: [阅读原文](http://blog.csdn.net/renfufei/article/details/38474435)

### 2、tomcat的配置

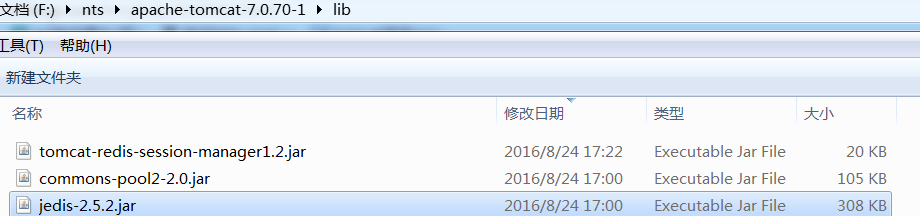
要求把新开发的代码推送到到生产系统中部署，生产系统要能够零宕机、对使用用户零影响。我的设想是使用集群来搞定，通过通知负载均衡Nginx，取下集群中的Tomcat节点，然后对Tomcat上的应用进行升级，再通知负载均衡Nginx，把Tomcat节点重新加载上去。依次这么做，把集群中的所有Tomcat都替换一次即可。那么问题来了，在取下Tomcat节点和加载新Tomcat节点时如何做到对用户无影响呢？方法很简单，共享Session。我们要使用tomcat-redis-session-manager这样的第三方库，主页见：

https://github.com/jcoleman/tomcat-redis-session-manager

要注意此库并非开箱即用的，使用时需要做一些调整你需要下载源码，并在更新了依赖库的版本后重建项目。构建过程这里就不多写 。

我也是在网上看到相关文章的，移步 [参考1](http://www.cnblogs.com/lengfo/p/4260363.html) [参考2](http://blog.csdn.net/chszs/article/details/42610365)

正式开始配置tomcat，先拷贝jar到lib目录哪几个jar 见下图



开始配置tomcat 打开 context.xml 编辑

加入下面代码 （注意了 tomcat-redis-session-manager 你如果直接用官方的话， 这里启动tomcat之后就会保错的 这个要你自己编译的 ）

<Valve className="com.orangefunction.tomcat.redissessions.RedisSessionHandlerValve" />

<Manager className="com.orangefunction.tomcat.redissessions.RedisSessionManager"

host="127.0.0.1"

port="6379"

database="0"

maxInactiveInterval="60" />

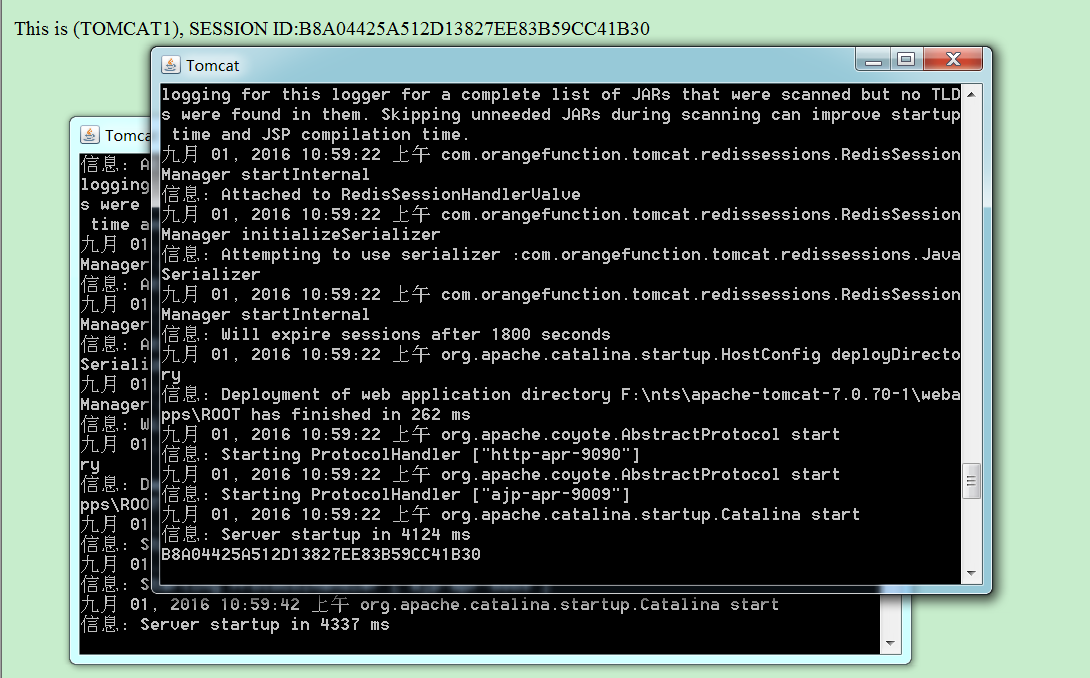
开始测试了

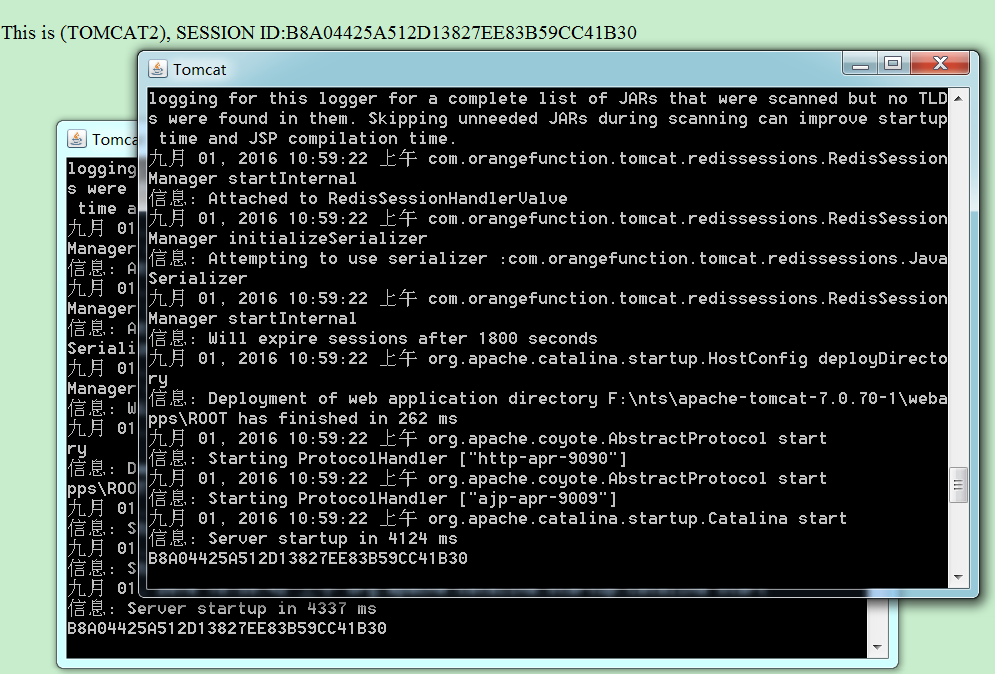
启动 nginx 、redis、tomcat1、 tomcat2

访问

xxx/s.jsp(s.jsp 自己写的一个 显示session的)

运行截图如下





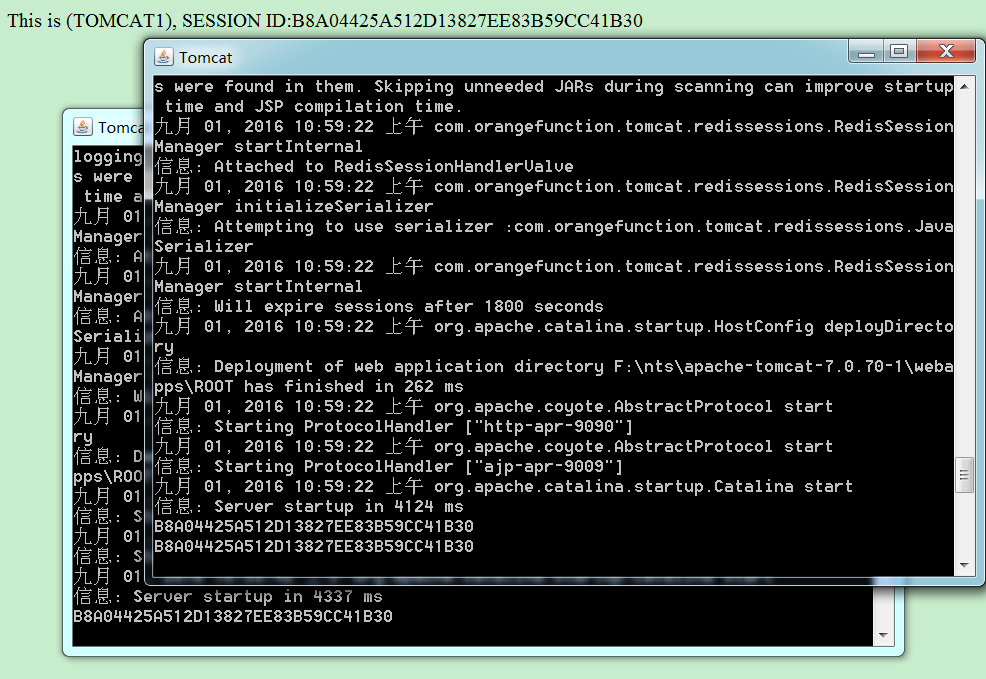
session id一样了，这就说明两台服务器session实现了共享，取下一台服务器，我们看看能不能正常切过来，取下一个节点，例如

#server 127.0.0.1:9090 weight=2 max\_fails=2 fail\_timeout=10s;

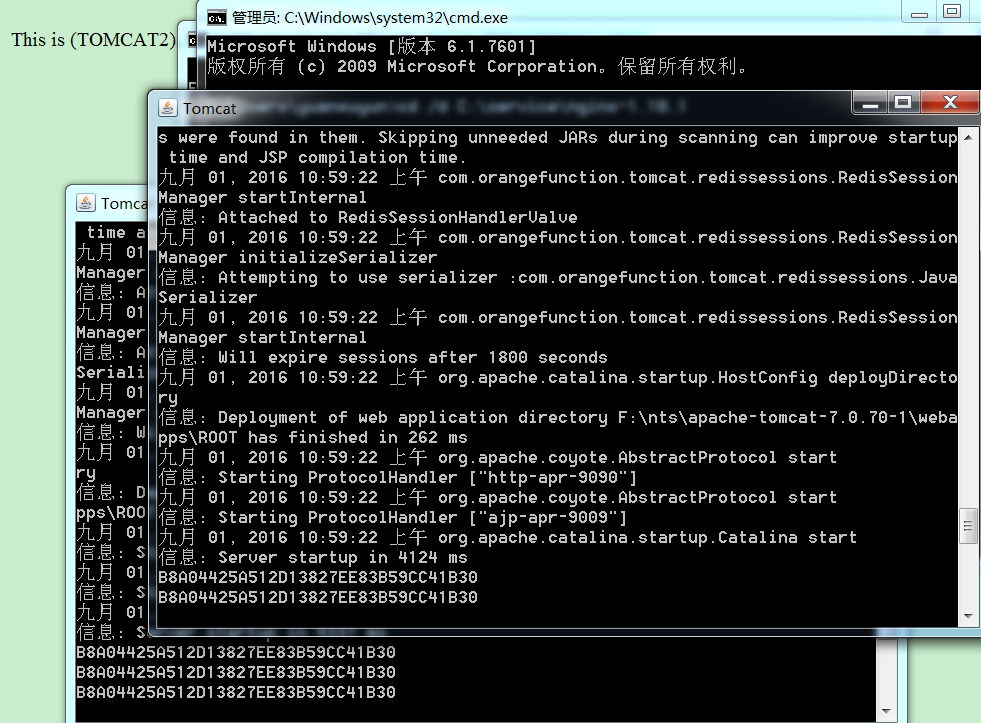
server 127.0.0.1:9090 down;

再访问看看有会不会有错误呢

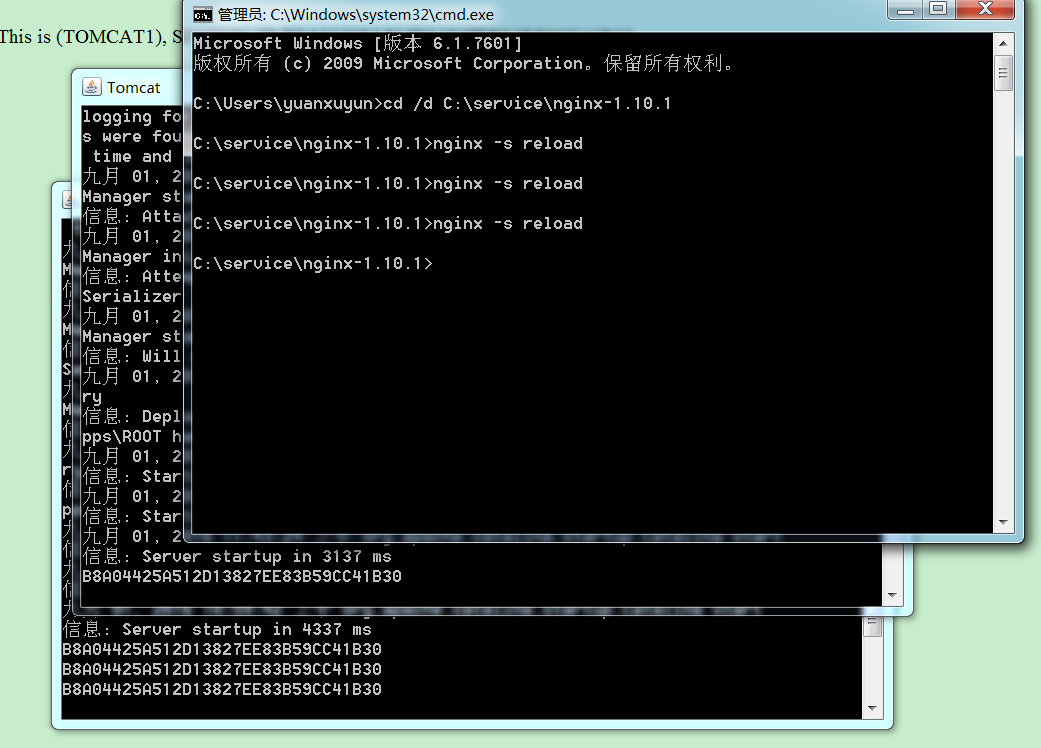
取下前



取下后



看到了吧 session不变，都转发到tomcat2了，重启tomcat1 之后再看看，如图



### 3、总结

采用这种方式，可以随机的访问到某一台服务器，减轻服务器压力，同时在进行程序更新的时候（server 127.0.0.1:7001 down;），也能保证有一台服务器在处于工作状态，能够零宕机、对使用用户零影响。同时通过Nginx来处理静态页面要比通过Tomcat处理在性能方面好很多。

## 运行环境展示

