Tomcat集群

# 负载均衡

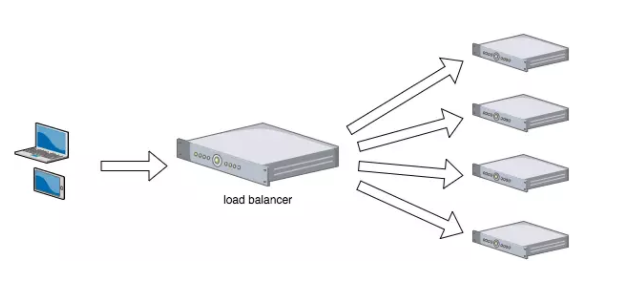
负载均衡（Load Balance），意思是将负载（工作任务，访问请求）进行平衡、分摊到多个操作单元（服务器，组件）上进行执行。是解决高性能，单点故障（高可用），扩展性（水平伸缩）的终极解决方案，常见的负载均衡方式主要分为两种：

一种是通过硬件来进行进行，常见的硬件有比较昂贵的NetScaler、F5、Radware和Array等商用的负载均衡器，也有类似于LVS、Nginx、HAproxy的基于Linux的开源的负载均衡策略,商用负载均衡里面NetScaler从效果上比F5的效率上更高。对于负载均衡器来说，不过商用负载均衡由于可以建立在四-七层协议之上，因此适用面更广所以有其不可替代性， 他的优点就是有专业的维护团队来对这些服务进行维护、缺点就是花销太大， 所以对于规模较小的网络服务来说暂时还没有需要使用。

另一种负载均衡的方式是通过软件：比较常见的有Apache，LVS、Nginx、HAproxy等，其中LVS是建立在四层协议上面的，而另外Nginx和HAproxy是建立在七层协议之上的。软负载的实现方式比较灵活，组件较多。可以针对自己的业务选择适合自己的组件及方案。

# 2.负载均衡原理

系统的扩展可分为纵向（垂直）扩展和横向（水平）扩展。纵向扩展，是从单机的角度通过增加硬件处理能力，比如CPU处理能力，内存容量，磁盘等方面，实现服务器处理能力的提升，不能满足大型分布式系统（网站），大流量，高并发，海量数据的问题。因此需要采用横向扩展的方式，通过添加机器来满足大型网站服务的处理能力。比如：一台机器不能满足，则增加两台或者多台机器，共同承担访问压力。这就是典型的集群和负载均衡架构：如下图：



应用集群：将同一应用部署到多台机器上，组成处理集群，接收负载均衡设备分发的请求，进行处理，并返回相应数据。

负载均衡设备：将用户访问的请求，根据负载均衡算法，分发到集群中的一台处理服务器。（一种把网络请求分散到一个服务器集群中的可用服务器上去的设备）。

负载均衡的作用（解决的问题）：

1.解决并发压力，提高应用处理性能（增加吞吐量，加强网络处理能力）

2.提供故障转移，实现高可用

3.通过添加或减少服务器数量，提供网站伸缩性（扩展性）

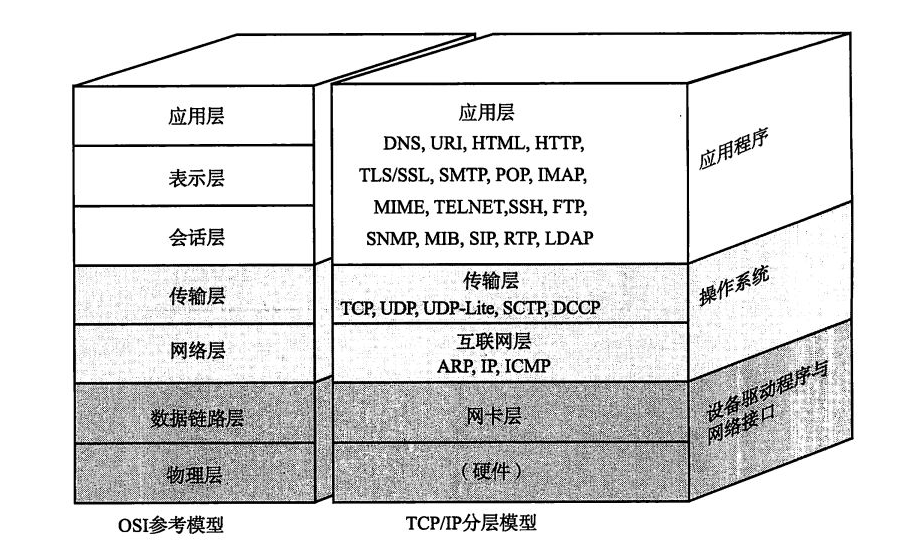
4.安全防护（负载均衡设备上做一些过滤，黑白名单等处理）

# 3.四层和七层

负载均衡又分为四层负载均衡和七层负载均衡。

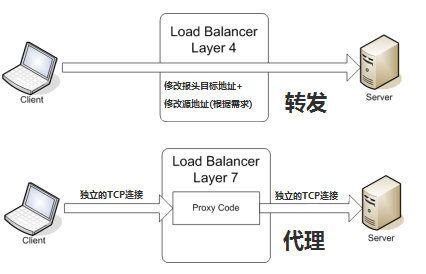
四层负载均衡工作在OSI模型的**传输层**，主要工作是转发，它在接收到客户端的流量以后通过修改数据包的地址信息将流量转发到应用服务器。

七层负载均衡工作在OSI参考模型的**应用层**，因为它需要解析应用层流量，所以七层负载均衡在接到客户端的流量以后，还需要一个完整的TCP/IP协议栈。七层负载均衡会与客户端建立一条完整的连接并将应用层的请求流量解析出来，再按照调度算法选择一个应用服务器，并与应用服务器建立另外一条连接将请求发送过去，**因此七层负载均衡的主要工作就是代理。**



## （1）技术原理上的区别

　所谓四层负载均衡，也就是主要通过报文中的目标地址和端口，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。以常见的TCP为例，负载均衡设备在接收到第一个来自客户端的SYN 请求时，即通过上述方式选择一个最佳的服务器，并对报文中目标IP地址进行修改(改为后端服务器IP），直接转发给该服务器。TCP的连接建立，即三次握手是客户端和服务器直接建立的，负载均衡设备只是起到一个类似路由器的转发动作。在某些部署情况下，为保证服务器回包可以正确返回给负载均衡设备，在转发报文的同时可能还会对报文原来的源地址进行修改。



　所谓七层负载均衡，也称为“内容交换”，也就是主要通过**报文中的真正有意义的应用层内容**，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。以常见的TCP为例，负载均衡设备如果要根据真正的应用层内容再选择服务器，只能先代理最终的服务器和客户端建立连接(三次握手)后，才可能接受到客户端发送的真正应用层内容的报文，然后再根据该报文中的特定字段，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。负载均衡设备在这种情况下，更类似于一个代理服务器。负载均衡和前端的客户端以及后端的服务器会分别建立TCP连接。所以从这个技术原理上来看，七层负载均衡明显的对负载均衡设备的要求更高，处理七层的能力也必然会低于四层模式的部署方式。那么，为什么还需要七层负载均衡呢？

## （2）应用场景的需求

　七层应用负载的好处，是使得整个网络更**"智能化"**, 参考我们之前的[另外一篇专门针对HTTP应用的优化的介绍](http://virtualadc.blog.51cto.com/3027116/580832)，就可以基本上了解这种方式的优势所在。例如访问一个网站的用户流量，可以通过七层的方式，将对图片类的请求转发到特定的图片服务器并可以使用缓存技术；将对文字类的请求可以转发到特定的文字服务器并可以使用压缩技术。当然这只是七层应用的一个小案例，从技术原理上，**这种方式可以对客户端的请求和服务器的响应进行任意意义上的修改，极大的提升了应用系统在网络层的灵活性**。很多在后台，(例如Nginx或者Apache)上部署的功能可以前移到负载均衡设备上，例如客户请求中的Header重写，服务器响应中的关键字过滤或者内容插入等功能。

　另外一个常常被提到功能就是**安全性**。网络中最常见的SYN Flood攻击，即黑客控制众多源客户端，使用虚假IP地址对同一目标发送SYN攻击，通常这种攻击会大量发送SYN报文，耗尽服务器上的相关资源，以达到*Denial of Service*(*DoS*)的目的。从技术原理上也可以看出，四层模式下这些SYN攻击都会被转发到后端的服务器上；**而七层模式下这些SYN攻击自然在负载均衡设备上就截止，不会影响后台服务器的正常运营**。另外负载均衡设备可以在七层层面设定多种策略，过滤特定报文，例如*SQL* Injection等应用层面的特定攻击手段，从应用层面进一步提高系统整体安全。现在的**7层负载均衡，主要还是着重于应用广泛的HTTP协议，所以其应用范围主要是众多的网站或者内部信息平台等基于B/S开发的系统**。 4层负载均衡则对应其他TCP应用，例如基于C/S开发的ERP等系统。

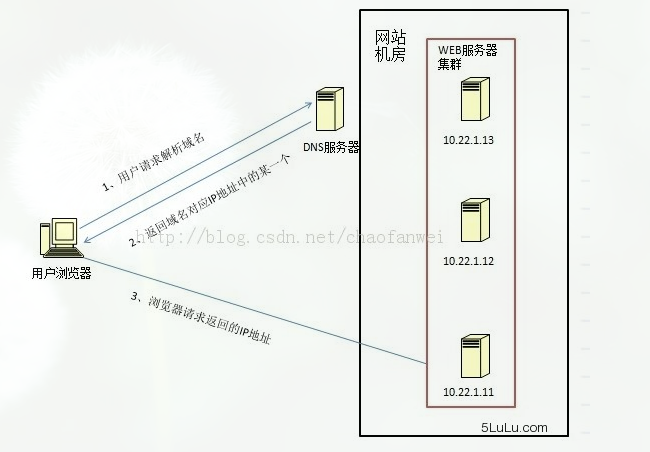
## （3）七层应用需要考虑的问题

1. 是否真的必要，七层应用的确可以提高流量智能化，*同时必不可免的带来设备配置复杂，负载均衡压力增高以及故障排查上的复杂性等问题*。在设计系统时需要考虑四层七层同时应用的混杂情况。
2. 是否真的可以提高安全性。例如SYN Flood攻击，七层模式的确将这些流量从服务器屏蔽，但*负载均衡设备本身要有强大的抗DDoS能力*，否则即使服务器正常而作为中枢调度的负载均衡设备故障也会导致整个应用的崩溃。
3. 是否有足够的灵活度。七层应用的优势是可以让整个应用的流量智能化，但是负载均衡设备需要提供完善的七层功能，满足客户根据不同情况的基于应用的调度。*最简单的一个考核就是能否取代后台Nginx或者Apache等服务器上的调度功能*。能够提供一个七层应用开发接口的负载均衡设备，可以让客户根据需求任意设定功能，才真正有可能提供强大的灵活性和智能性。

# 4.负载均衡分类

## （1）DNS负载均衡

最早的负载均衡技术，利用域名解析实现负载均衡，在DNS服务器，配置多个A记录，这些A记录对应的服务器构成集群。大型网站总是部分使用DNS解析，作为第一级负载均衡



　利用DNS处理域名解析请求的同时进行负载均衡是另一种常用的方案。在DNS服务器中配置多个A记录，如：www.mysite.com IN A 114.100.80.1、www.mysite.com IN A 114.100.80.2、www.mysite.com IN A 114.100.80.3.  
　每次域名解析请求都会根据负载均衡算法计算一个不同的IP地址返回，这样A记录中配置的多个服务器就构成一个集群，并可以实现负载均衡。

优点：

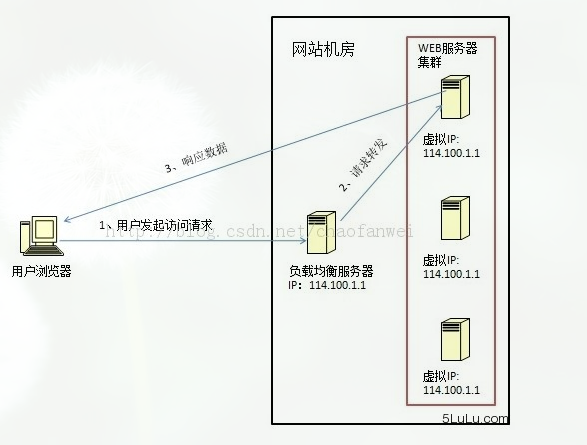
1. 使用简单：负载均衡工作，交给DNS服务器处理，省掉了负载均衡服务器维护的麻烦
2. 提高性能：可以支持基于地址的域名解析，解析成距离用户最近的服务器地址，可以加快访问速度，改善性能；

缺点：

1. 可用性差：DNS解析是多级解析，新增/修改DNS后，解析时间较长；解析过程中，用户访问网站将失败；
2. 扩展性低：DNS负载均衡的控制权在域名商那里，无法对其做更多的改善和扩展；
3. 维护性差：也不能反映服务器的当前运行状态；支持的算法少；不能区分服务器的差异（不能根据系统与服务的状态来判断负载）

## （2）数据链路层负载均衡(LVS)

数据链路层负载均衡是指在通信协议的数据链路层修改mac地址进行负载均衡



这种数据传输方式又称作三角传输模式，负载均衡数据分发过程中不修改IP地址，**只修改目的的mac地址**，通过配置真实物理服务器集群所有机器虚拟IP和负载均衡服务器IP地址一样，从而达到负载均衡，这种负载均衡方式又称为直接路由方式（DR）在上图中，用户请求到达负载均衡服务器后，负载均衡服务器将请求数据的目的mac地址修改为真是WEB服务器的mac地址，并不修改数据包目标IP地址，因此数据可以正常到达目标WEB服务器，该服务器在处理完数据后可以经过网管服务器而不是负载均衡服务器直接到达用户浏览器。使用三角传输模式的链路层负载均衡是目前大型网站所使用的最广的一种负载均衡手段。在linux平台上最好的链路层负载均衡开源产品是LVS(linux virtual server)。

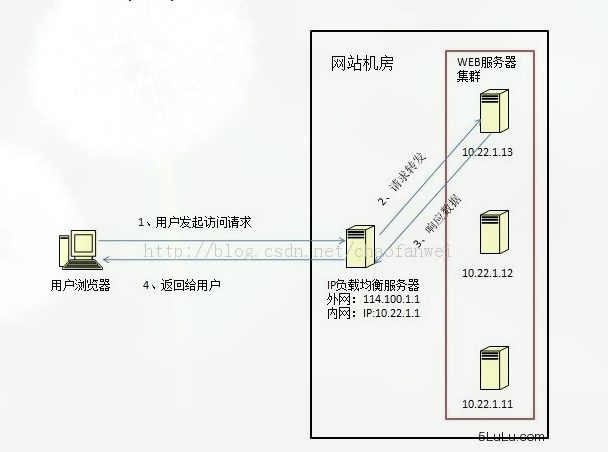
优点：性能好；

缺点：配置复杂；

实践建议：DR模式是目前使用最广泛的一种负载均衡方式。

## （3）IP负载均衡（SNAT）

IP负载均衡即在网络层通过修改请求目标地址进行负载均衡



用户请求数据包到达负载均衡服务器后，负载均衡服务器在操作系统内核进行获取网络数据包，根据负载均衡算法计算得到一台真实的WEB服务器地址，然后将数据包的IP地址修改为真实的WEB服务器地址，不需要通过用户进程处理。真实的WEB服务器处理完毕后，相应数据包回到负载均衡服务器，负载均衡服务器再将数据包源地址修改为自身的IP地址发送给用户浏览器。这里的关键在于真实WEB服务器相应数据包如何返回给负载均衡服务器。

存在两种方式：

（1）负载均衡服务器在修改目的ip地址的同时修改源地址。将数据包源地址设为自身盘，即源地址转换（snat）。

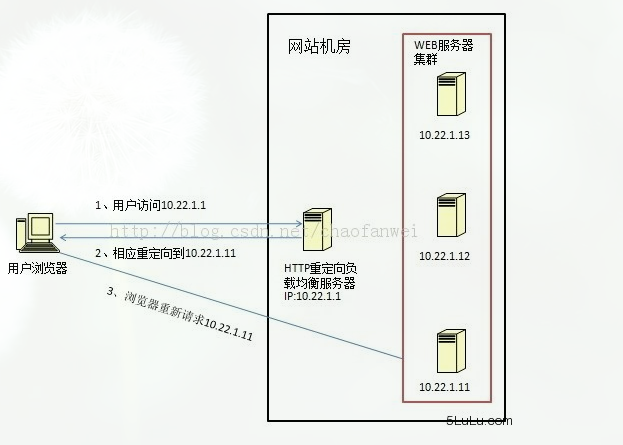
（2）将负载均衡服务器同时作为真实物理服务器集群的网关服务器。

 优点：在内核进程完成数据分发，比在应用层分发性能更好；

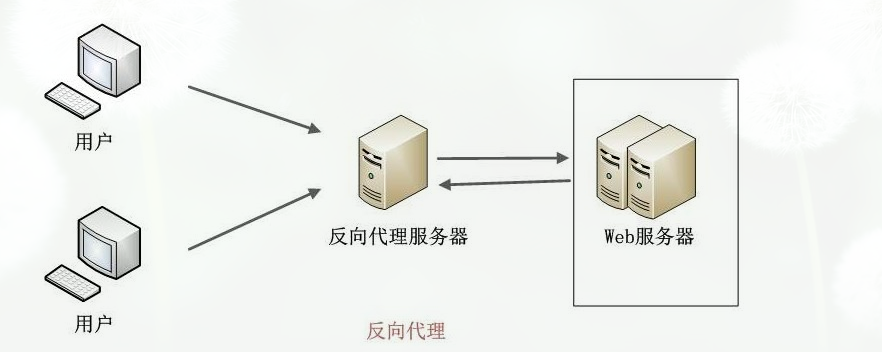
缺点：所有请求响应都需要经过负载均衡服务器，集群最大吞吐量受限于负载均衡服务器网卡带宽；

## （4）HTTP重定向负载均衡(少见)

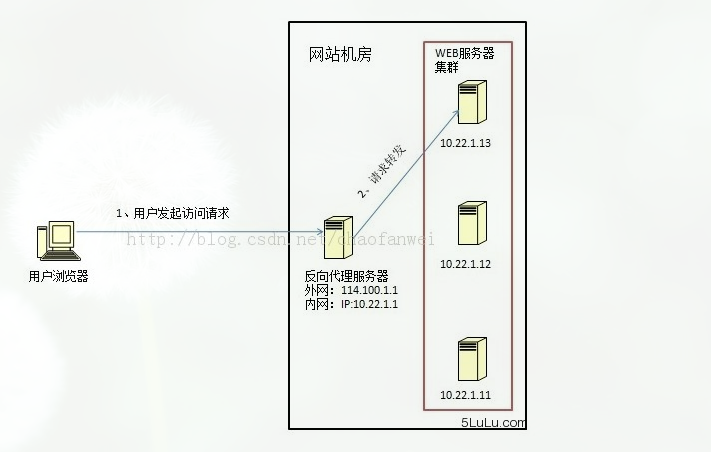
　HTTP重定向服务器是一台普通的应用服务器，其唯一的功能就是根据用户的HTTP请求计算一台真实的服务器地址，并将真实的服务器地址写入HTTP重定向响应中（响应状态吗302）返回给浏览器，然后浏览器再自动请求真实的服务器。这种负载均衡方案的优点是比较简单，缺点是浏览器需要每次请求两次服务器才能拿完成一次访问，性能较差；使用HTTP302响应码重定向，可能是搜索引擎判断为SEO作弊，降低搜索排名。重定向服务器自身的处理能力有可能成为瓶颈。因此这种方案在实际使用中并不见多



## （5）反向代理负载均衡



传统代理服务器位于浏览器一端，代理浏览器将HTTP请求发送到互联网上。而反向代理服务器则位于网站机房一侧，代理网站web服务器接收http请求。反向代理的作用是保护网站安全，所有互联网的请求都必须经过代理服务器，相当于在web服务器和可能的网络攻击之间建立了一个屏障。除此之外，代理服务器也可以配置缓存加速web请求。当用户第一次访问静态内容的时候，静态内存就被缓存在反向代理服务器上，这样当其他用户访问该静态内容时，就可以直接从反向代理服务器返回，加速web请求响应速度，减轻web服务器负载压力。另外，反向代理服务器也可以实现负载均衡的功能。



# 5.负载均衡组件

## Apache

它是Apache软件基金会的一个开放源代码的跨平台的网页服务器，属于老牌的web服务器了，支持基于Ip或者域名的虚拟主机，支持代理服务器，可正向代理也可以反向代理，支持安全Socket层(SSL)等等，结合tomcat等servlet容器处理jsp，Tomcat也是Apache基金会的，他们天然的结合得很好。官网：http://httpd.apache.org

## Ngnix

俄罗斯人开发的一个高性能的 HTTP和反向代理服务器。由于Nginx 超越 Apache 的高性能和稳定性，使得国内使用 Nginx 作为 Web 服务器的网站也越来越多，其中包括新浪博客、新浪播客、网易新闻、腾讯网、搜狐博客等门户网站频道等，在3w以上的高并发环境下，ngnix处理能力相当于apache的10倍。

Nginx的特点是：

1.工作在网络的7层之上，可以针对http应用做一些分流的策略，比如针对域名、目录结构；

2.Nginx对网络的依赖比较小；

3.Nginx安装和配置比较简单，测试起来比较方便；

4.也可以承担高的负载压力且稳定，一般能支撑超过1万次的并发；

5、Nginx可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持url来检测；

6、Nginx对请求的异步处理可以帮助节点服务器减轻负载；

7、Nginx能支持http和Email，这样就在适用范围上面小很多；

8、不支持Session的保持、对Big request header的支持不是很好，另外默认的只有Round-robin和IP-hash两种负载均衡算法。

## lvs

   Linux Virtual Server的简写，意即Linux虚拟服务器，是一个虚拟的服务器集群系统。由毕业于国防科技大学的章文嵩博士于1998年5月创立，可以实现LINUX平台下的简单负载均衡。了解更多，访问官网：<http://zh.linuxvirtualserver.org/>。

LVS：使用集群技术和Linux操作系统实现一个高性能、高可用的服务器，它具有很好的可伸缩性（Scalability）、可靠性（Reliability）和可管理性（Manageability）。

LVS的特点是：

1、抗负载能力强、是工作在网络4层之上仅作分发之用，没有流量的产生；

2、配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西，

所以并不需要太多接触，大大减少了人为出错的几率；

3、工作稳定，自身有完整的双机热备方案；

4、无流量，保证了均衡器IO的性能不会收到大流量的影响；

5、应用范围比较广，可以对所有应用做负载均衡；

6、LVS需要向IDC多申请一个IP来做Visual IP，因此需要一定的网络知识，所以对操作人的要求比较高。

## keepalived

这里说的keepalived不是apache或者tomcat等某个组件上的属性字段，它也是一个组件，可以实现web服务器的高可用(HA high availably)。它可以检测web服务器的工作状态，如果该服务器出现故障被检测到，将其剔除服务器群中，直至正常工作后，keepalive会自动检测到并加入到服务器群里面。实现主备服务器发生故障时ip瞬时无缝交接。它是LVS集群节点健康检测的一个用户空间守护进程，也是LVS的引导故障转移模块（director failover）。Keepalived守护进程可以检查LVS池的状态。如果LVS服务器池当中的某一个服务器宕机了。keepalived会通过一 个setsockopt呼叫通知内核将这个节点从LVS拓扑图中移除。

## HAProxy

  HAProxy提供高可用性、负载均衡以及基于TCP和HTTP应用的代理，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。HAProxy特别适用于那些负载特大的web站点， 这些站点通常又需要会话保持或七层处理。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中， 同时可以保护你的web服务器不被暴露到网络上.

HAProxy的特点是：

1、HAProxy是工作在网络7层之上。

2、能够补充Nginx的一些缺点比如Session的保持，Cookie的引导等工作

3、支持url检测后端的服务器出问题的检测会有很好的帮助。

4、更多的负载均衡策略比如：动态加权轮循(Dynamic Round Robin)，

加权源地址哈希(Weighted Source Hash)，

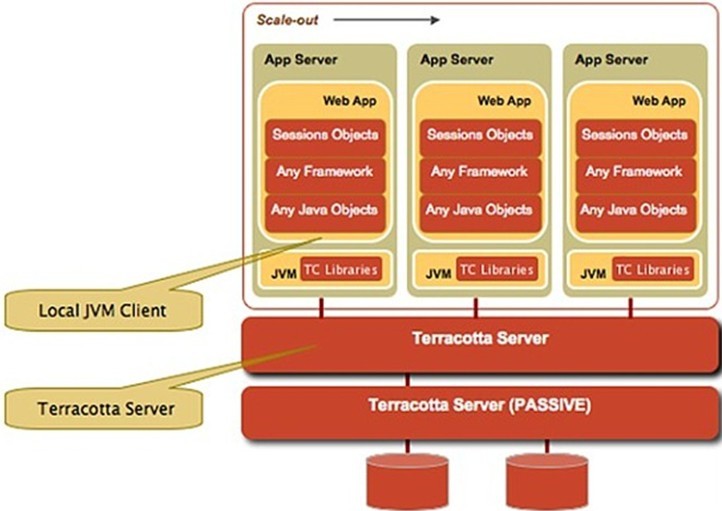
加权URL哈希和加权参数哈希(Weighted Parameter Hash)已经实现

5、单纯从效率上来讲HAProxy更会比Nginx有更出色的负载均衡速度。

6、HAProxy可以对Mysql进行负载均衡，对后端的DB节点进行检测和负载均衡。

## Terracotta

  Terracotta是一种分布式java集群技术，它巧妙得隐藏了多个分布式JVM带来的复杂性，使得java对象能够透明得在多个JVM集群中进行分享和同步，并能够进行持久化。从某种意义上讲它类似于hadoop中的zookeeper，可以作为zookeeper之外的另外一种选择。Terracotta采用的是一种被称之为中心辐射的架构。在这种架构里运行着分布式应用程序的JVM们在启动时都会与一台中心Terracotta服务器相连。Terracotta服务器负责存储DSO对象数据，协调JVM之间的并发线程。Terracotta库位于应用程序JVM中，在类加载过程中，它们用来对类的字节码进行增强，处理同步块内的lock和unlock请求，处理应用JVM之间的wait（），notify（）请求，处理运行时和Terracotta服务器的联系等等



是一个JVM级的开源群集框架，它最重要的一个功能就是DSO(Distributed Shared Object)，通过DSO我们可以把那些被频繁访问的，重要的数据缓存在TC Server上，然后供集群里的不同JVM共享，这样减轻了数据库的负载，它同时还提供：HTTP Session复制，spring security整合、与Hibernate整合，分布式缓存(收购了java开源缓存项目Ehcache以及Java任务调度项目Quartz,并对其做了深度整合)，POJO群集（比如spring框架中引入terracotta，那么spring中的bean就可以分布式集群了。bean被分布式了，那么我们可以无缝得扩展我们的web系统，我们的web系统也天然得有了故障转移机制），跨越群集的JVM来实现分布式应用程序协调(**采用代码注入的方式，所以你不需要修改任何**)，同时TC Server本身也可以配置成集群的形式，宕机的TC Server中的活动自动而无缝地转移到集群里的及其他TC Server上去

Terracotta是一个可以在不改变任何现有系统代码的基础上提供HA和HP的一个优秀解决方案，与其他集群相比：

1、由于大多数Web应用服务器都采用Java序列化和数据广播方式实现session数据的共享，导致任何一个节点对任何session数据进行修改，都造成大量的内存、CPU以及网络带宽的消耗。这种消耗随着应用服务器节点的增加而成级数级增长。当节点数目超过4个以上的时候，经常由于session复制导致的消耗过高，使得整个集群的吞吐量反而开始下降。由于普通session复制机制性能和可用性的缺陷，很多web开发人员不得不通过数据保存和共享session相关的数据，因此又增加了对数据库的压力，形成新的性能瓶颈，Terracotta集群实现sessino数据的共享。不使用广播机制，避免了Java序列化，只把被修改的字段的数据传递给服务器和使用节点，大大减少CPU和内存消耗

2、利用服务器实现网络扩展内存，使得有限内存的客户端节点可以访问远大于其内存容量的数据结构，而不必担心发生内存溢出的异常

3、数据保存在服务器端，因此客户端JVM宕机不会造成数据丢失

4、增量数据传递，智能数据推送，最大限度减少对网络的负担，使得客户端JVM可以横向扩展

5、服务器分片，实现服务器数据存储及数据吞吐量横向扩展

6、通过服务器实现共享数据持久化，通过服务器集群实现容错等

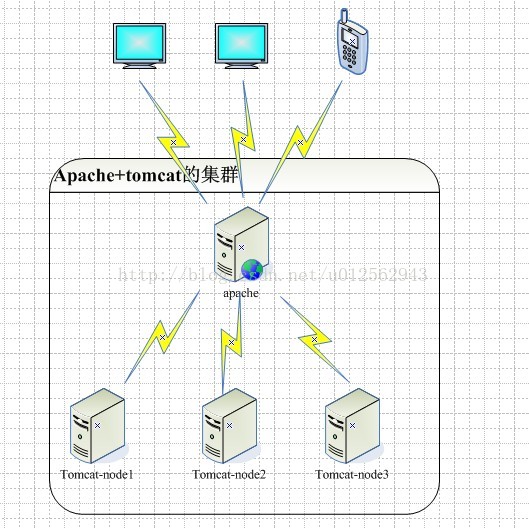
7、无需学习新的API，大大降低开发成本

8、广泛支持各种应用服务器：WebLogic, WebSphere, Tomcat, JBoss, Jetty, Geronimo等等，自动session数据迁移，集群范围内数据可视以及强大的管理和监控功能界面，大大方便对整合集群的共享数据、性能数据、软硬件指标等的实施监控、调试、优化(通过JMX开放服务器监控信息)

9、企业版的Terracotta服务器还提供了数据分片功能，使得集群吞吐量随着服务器数量的增加达到线性增长

# 6.案例

Apache2.4 + Tomcat7 组合



Apache HTTP [Server](https://baike.baidu.com/item/Server)（简称[Apache](https://baike.baidu.com/item/Apache/6265)）是[Apache软件基金会](https://baike.baidu.com/item/Apache%E8%BD%AF%E4%BB%B6%E5%9F%BA%E9%87%91%E4%BC%9A)的一个开放源码的网页服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是最流行的Web服务器端软件之一。它快速、可靠并且可通过简单的API扩展，将Perl/Python等解释器编译到服务器中。Apache HTTP服务器是一个模块化的服务器，源于NCSAhttpd服务器，经过多次修改，成为世界使用排名第一的[Web服务器](https://baike.baidu.com/item/Web%E6%9C%8D%E5%8A%A1%E5%99%A8)软件。它可以运行在几乎所有广泛使用的[计算机平台](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E5%B9%B3%E5%8F%B0)上。

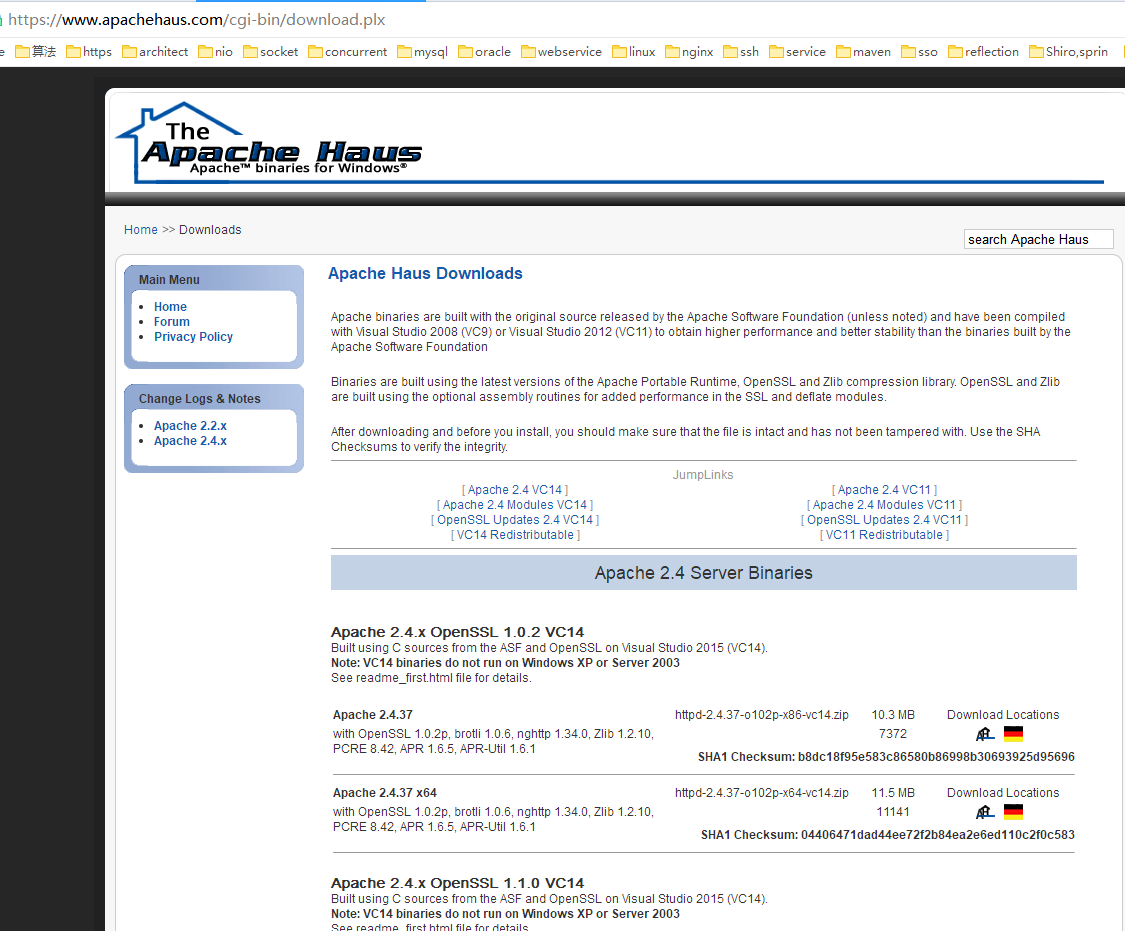
Apache源于NCSAhttpd服务器，经过多次修改，成为世界上最流行的[Web服务器](https://baike.baidu.com/item/Web%E6%9C%8D%E5%8A%A1%E5%99%A8)软件之一,Apache的特点是简单、速度快、性能稳定，并可做[代理服务器](https://baike.baidu.com/item/%E4%BB%A3%E7%90%86%E6%9C%8D%E5%8A%A1%E5%99%A8)来使用。

Apache和TomCat集群通常有3中模式：

1. jk\_model apache2.x支持
2. http\_proxy
3. ajp\_proxy （Apache推荐）

## （1）Apache安装启动

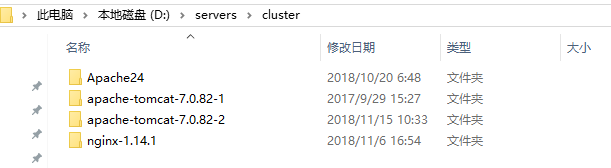
官网下载：https://www.apachehaus.com/cgi-bin/download.plx



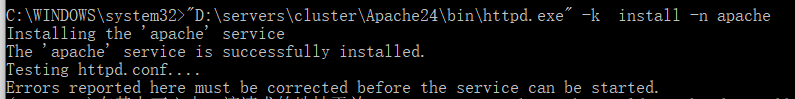
选择VC11（VC2012）或者 VC14（VC2015） 需要安装对应版本的C++运行库



解压到指定文件夹



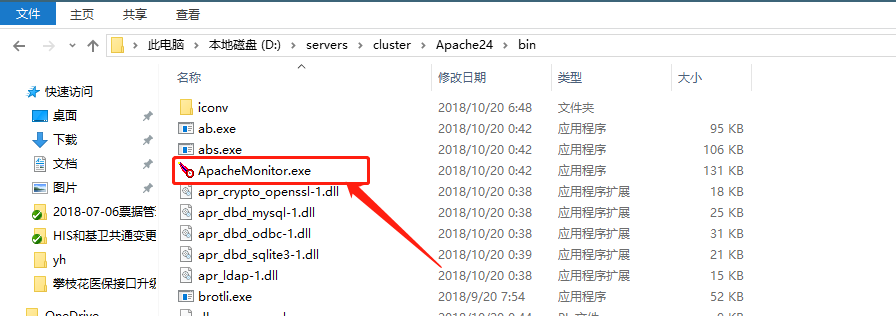
以管理员身份运行 "D:\servers\cluster\Apache24\bin\httpd.exe" –k install -n apache



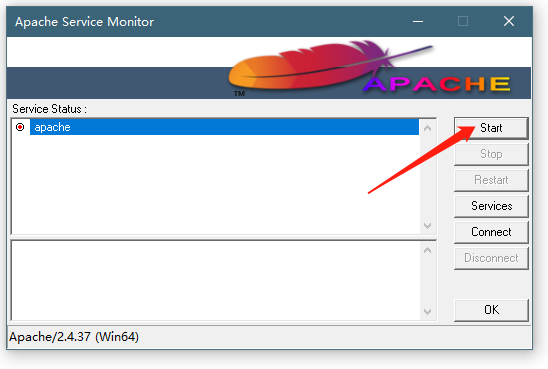
安装成功，如果要删除



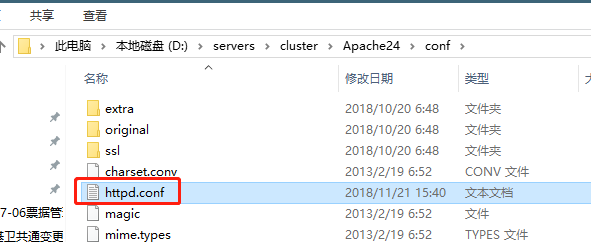
打开Apache监听器



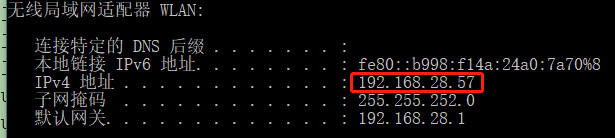
启动

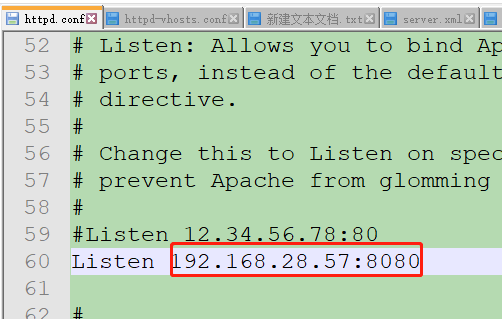


修改Apache24/conf/httpd.conf文件

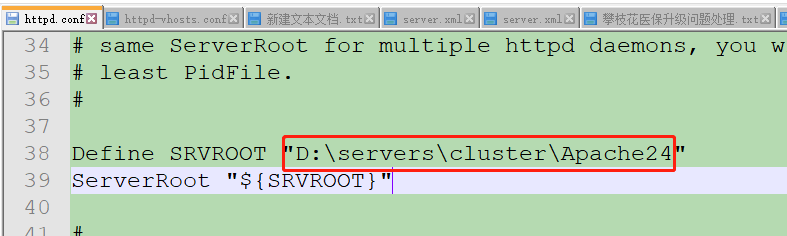


默认监听80端口，这里自行修改成本机IP，监听在8080端口

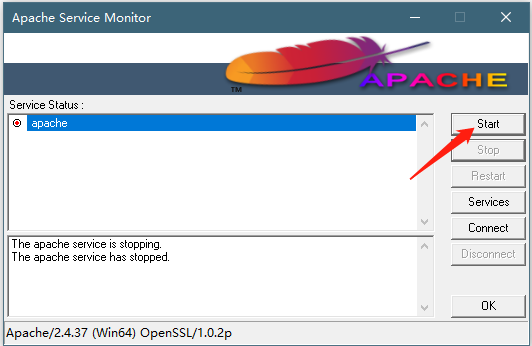




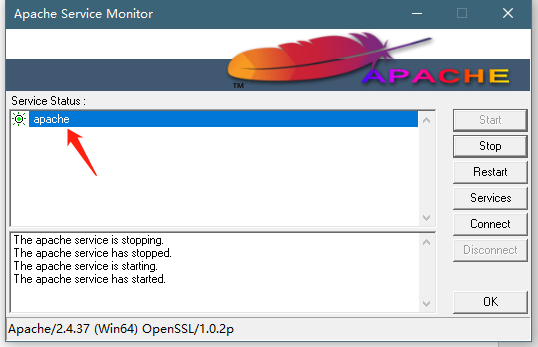
配置Apache服务根路径



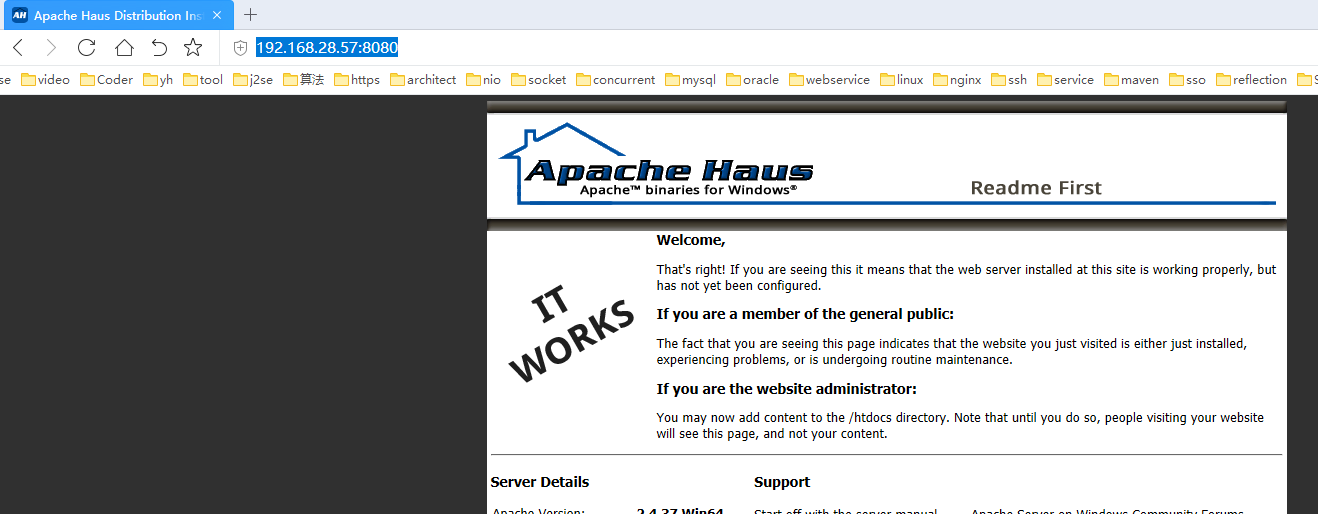
点启动

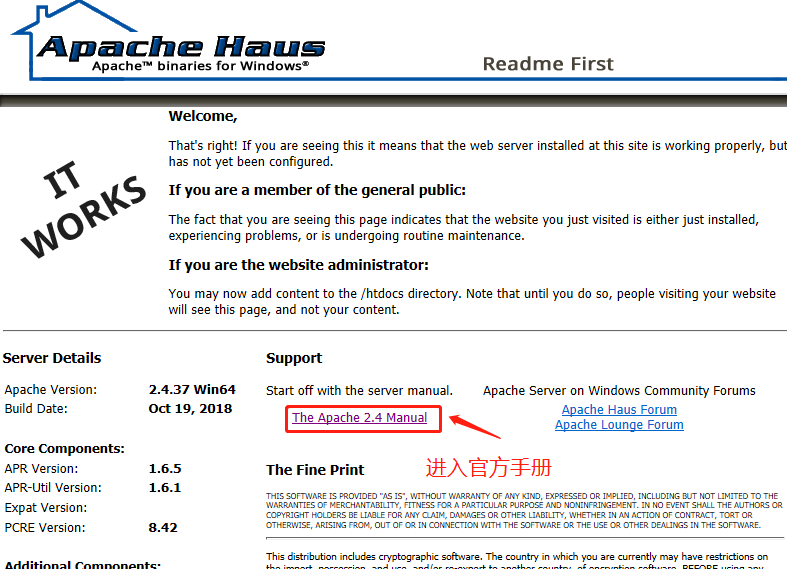


已启动状态：

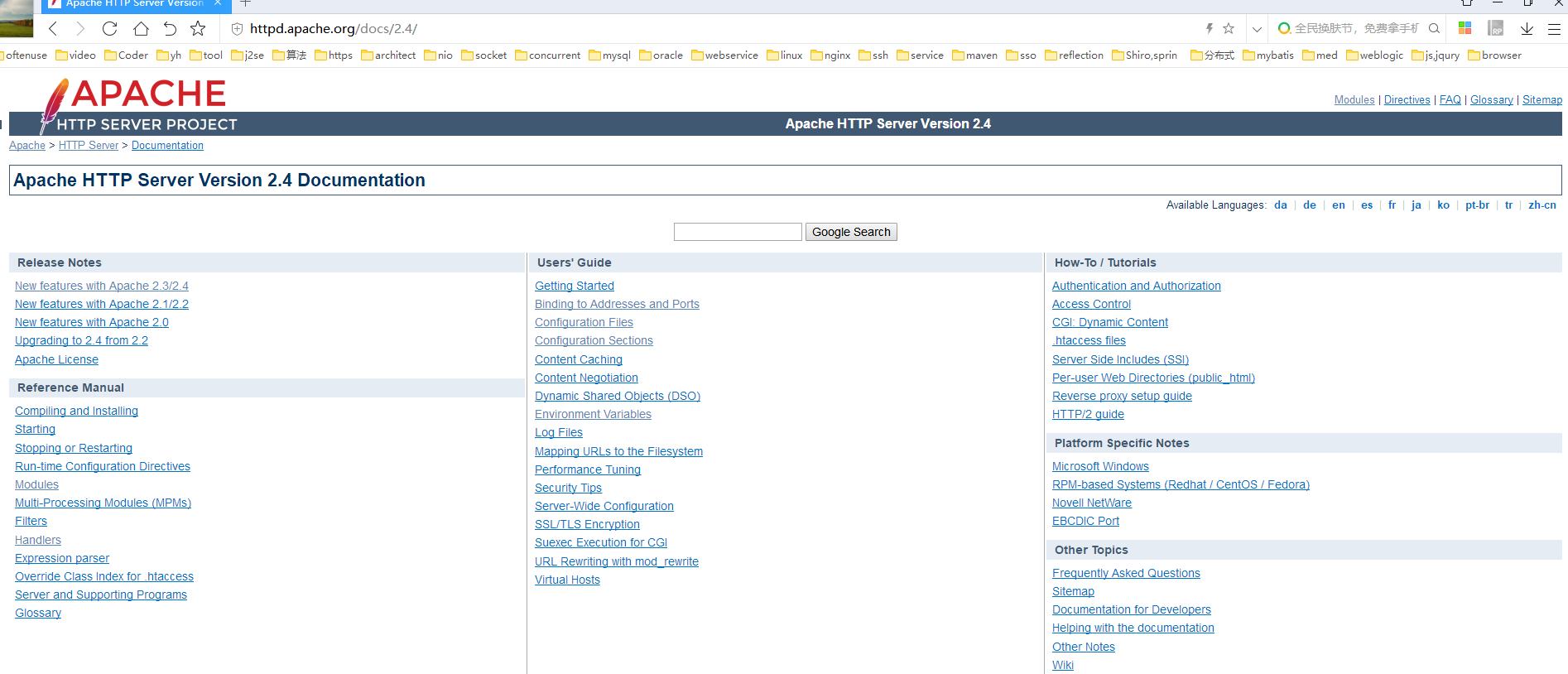


浏览器访问：<http://192.168.28.57:8080>进入首页



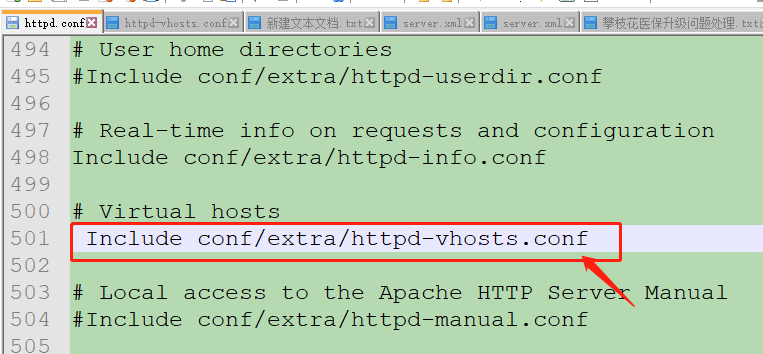


配置参数，参数选项，参数说明可以参考官方手册

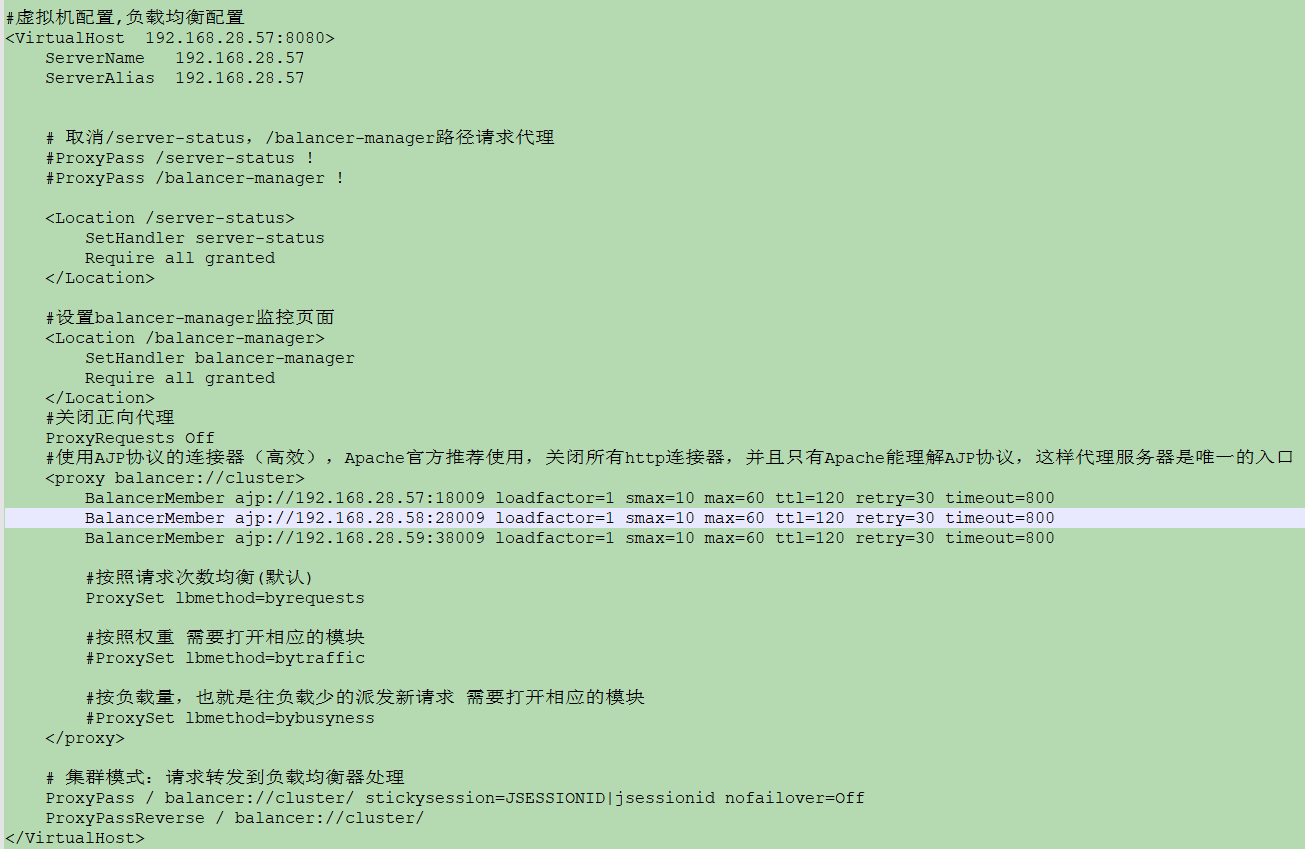


## （2）虚拟主机配置

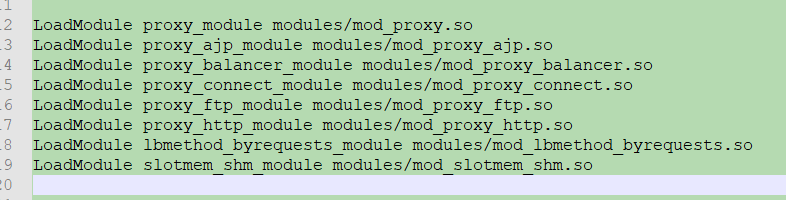
Httpd.conf中去掉”#”,将虚拟主机的配置包含进来。

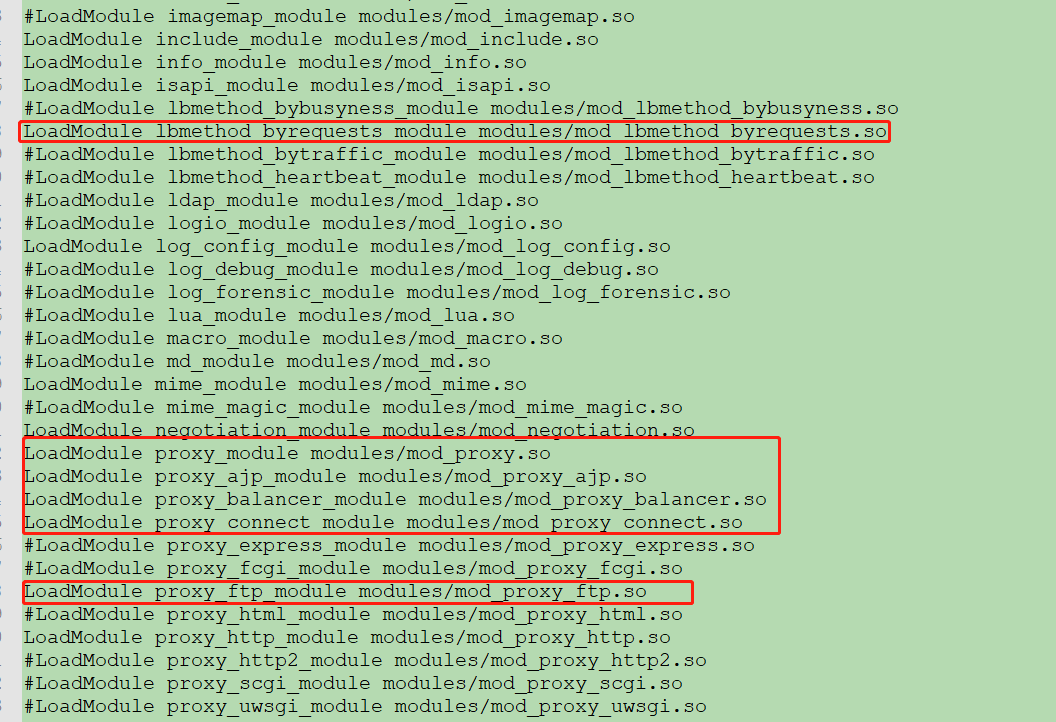


将VirtualHost节点全部注释掉，添加自己的配置

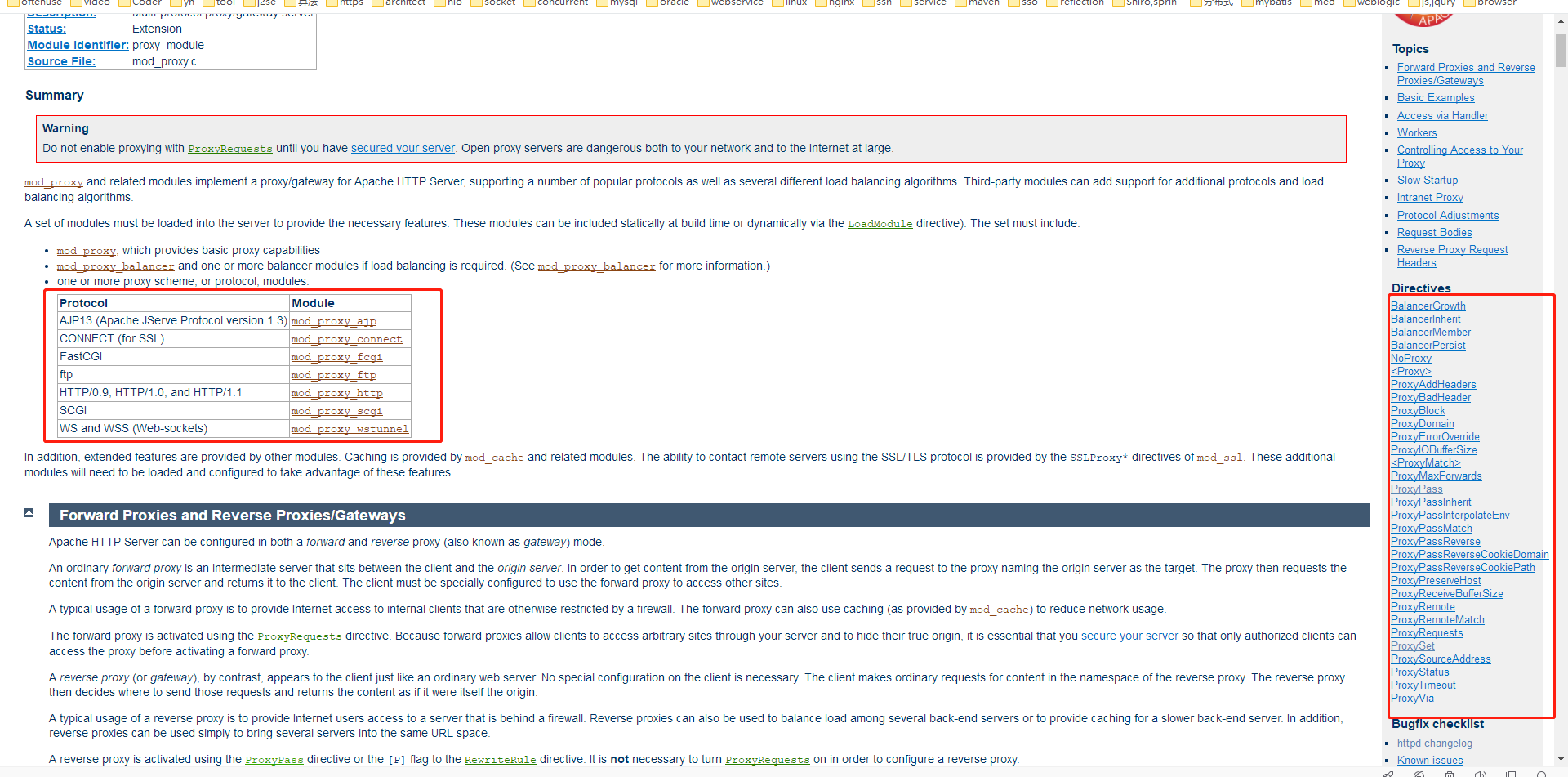


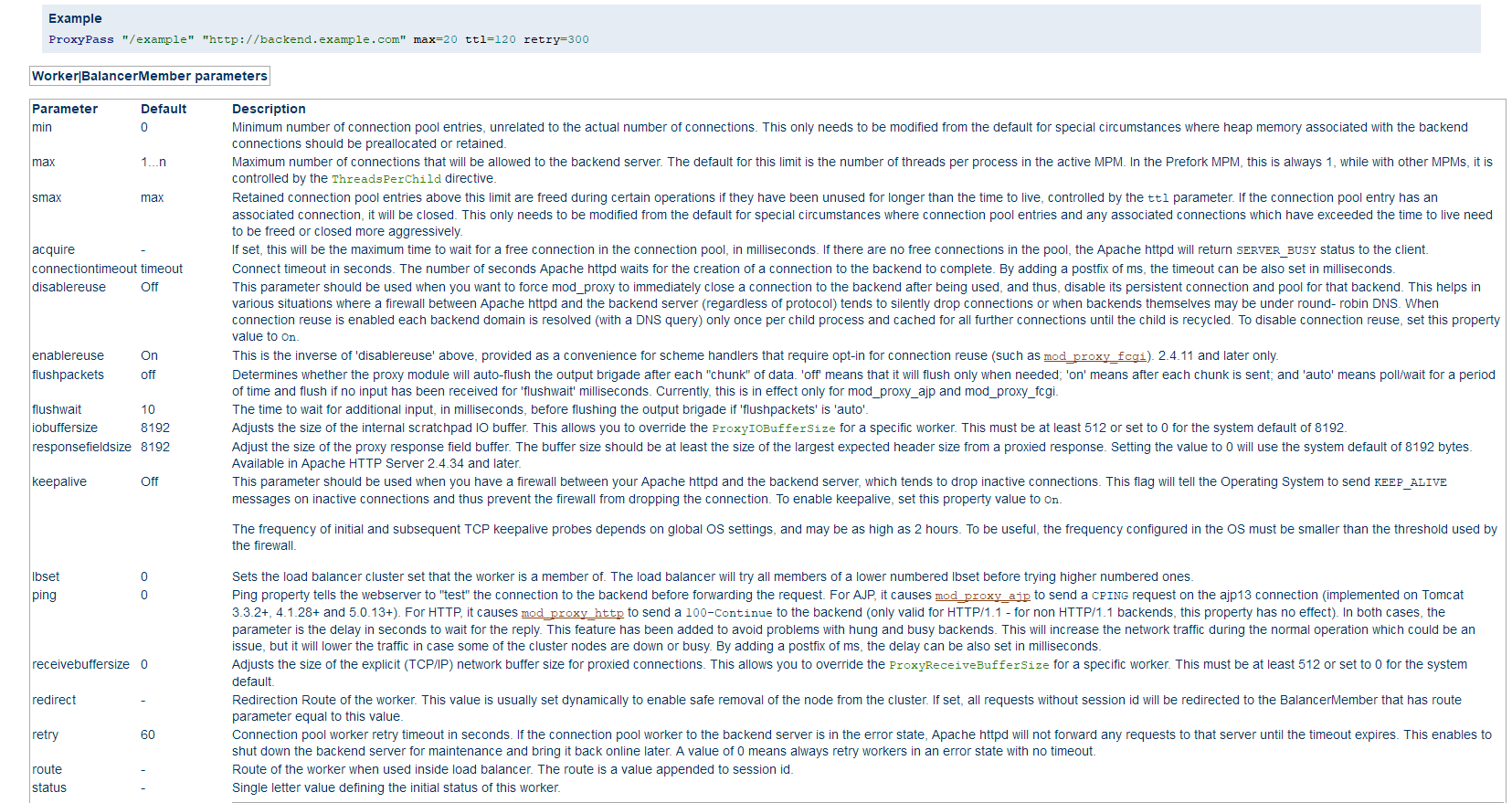
并且在httpd.conf中打开如下模块





关于参数的配置请参考 <http://httpd.apache.org/docs/2.4/mod/mod_proxy.html#proxypass>



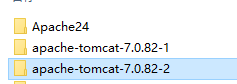


修改Apache的配置后，需要重启Apache服务器，配置才能生效

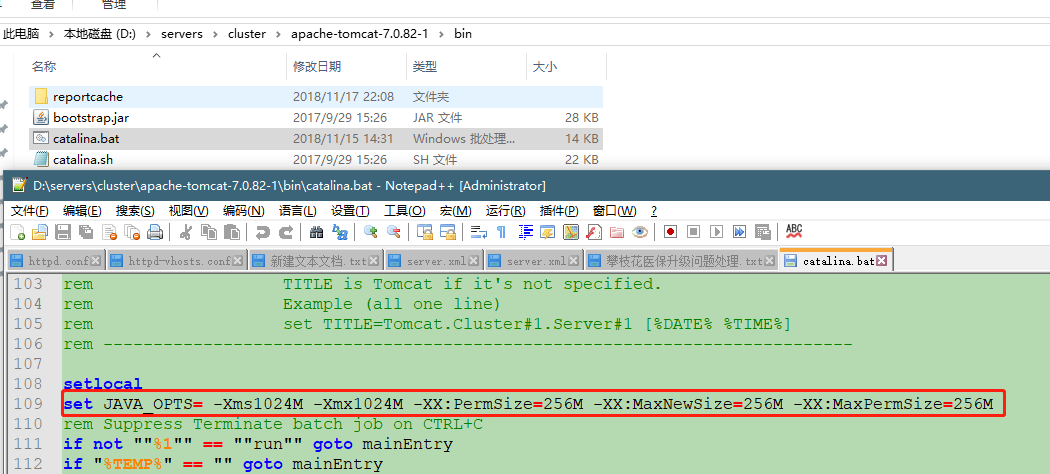


## （3）tomcat配置

Tomcat版本为7.0.82免安装版，复制两份

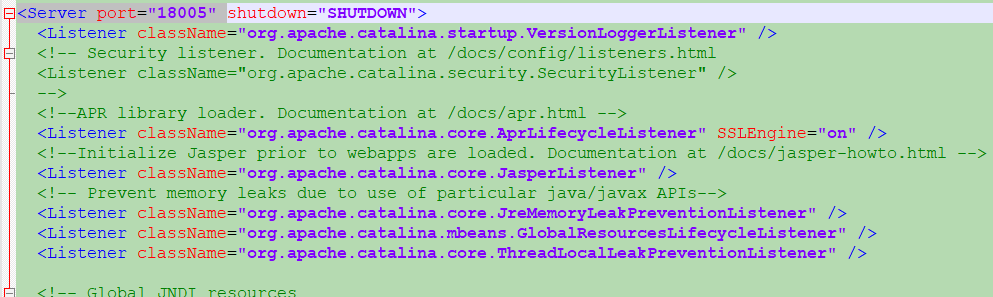


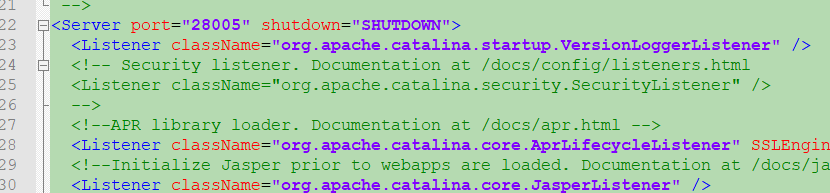
设置tomcat堆区大小，永久代大小。



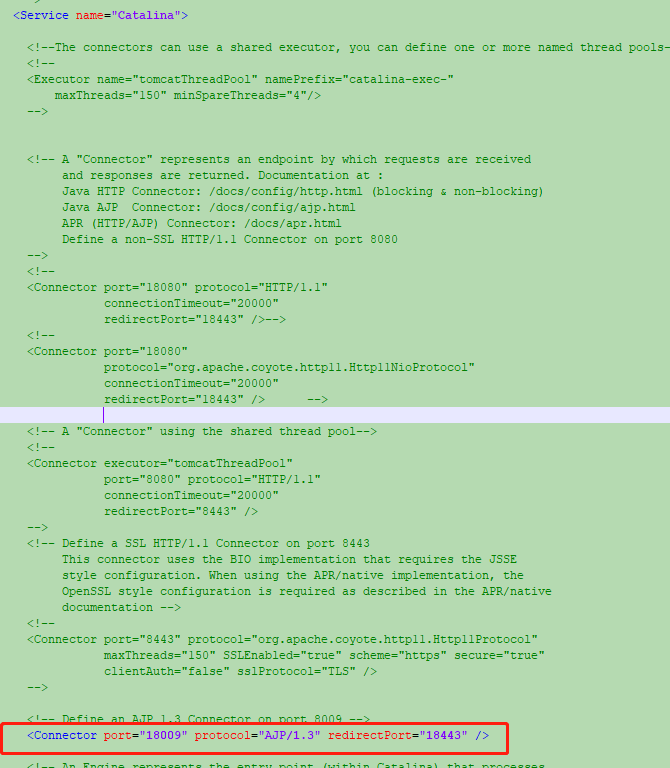
Server.xml配置

Server 端口不能重复

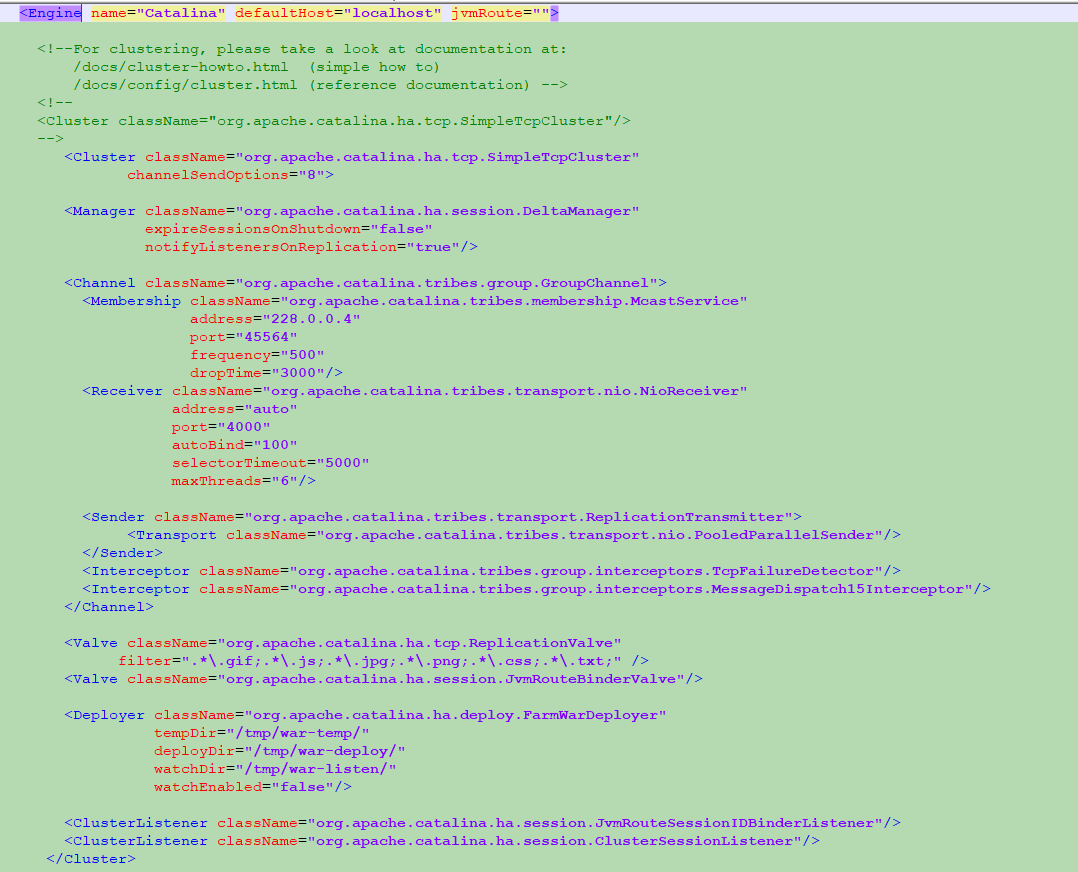




关闭所有HTTP连接器，只保留AJP连接器，并且端口不能重复



每个Tomcat的Engine节点下配置集群节点



<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"

channelSendOptions="8">

<Manager className="org.apache.catalina.ha.session.DeltaManager"

expireSessionsOnShutdown="false"

notifyListenersOnReplication="true"/>

<Channel className="org.apache.catalina.tribes.group.GroupChannel">

<MembershipclassName="org.apache.catalina.tribes.membership.McastService"

address="228.0.0.4"

port="45564"

frequency="500"

dropTime="3000"/>

<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"

address="auto"

port="4000"

autoBind="100"

selectorTimeout="5000"

maxThreads="6"/>

<SenderclassName="org.apache.catalina.tribes.transport.ReplicationTransmitter">

<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>

</Sender>

<InterceptorclassName="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>

<InterceptorclassName="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>

</Channel>

<Valve className="org.apache.catalina.ha.tcp.ReplicationValve"

filter=".\*\.gif;.\*\.js;.\*\.jpg;.\*\.png;.\*\.css;.\*\.txt;" />

<Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>

<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"

tempDir="/tmp/war-temp/"

deployDir="/tmp/war-deploy/"

watchDir="/tmp/war-listen/"

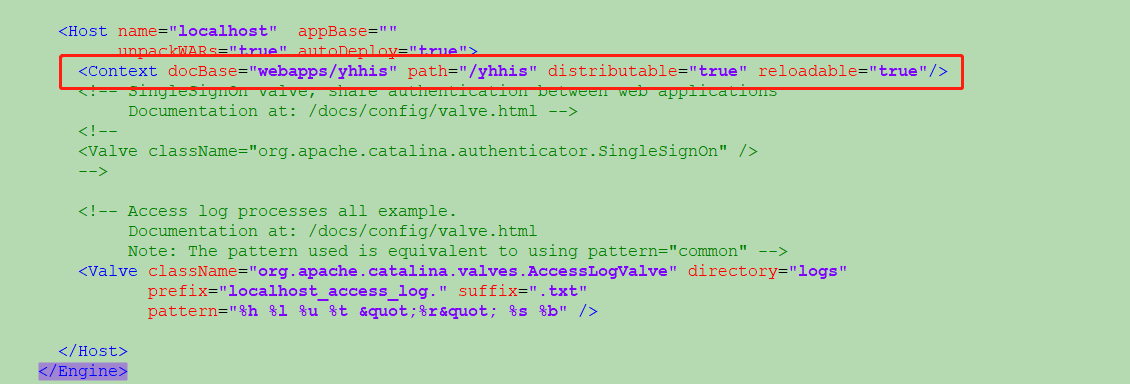
watchEnabled="false"/>

<ClusterListener className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener"/>

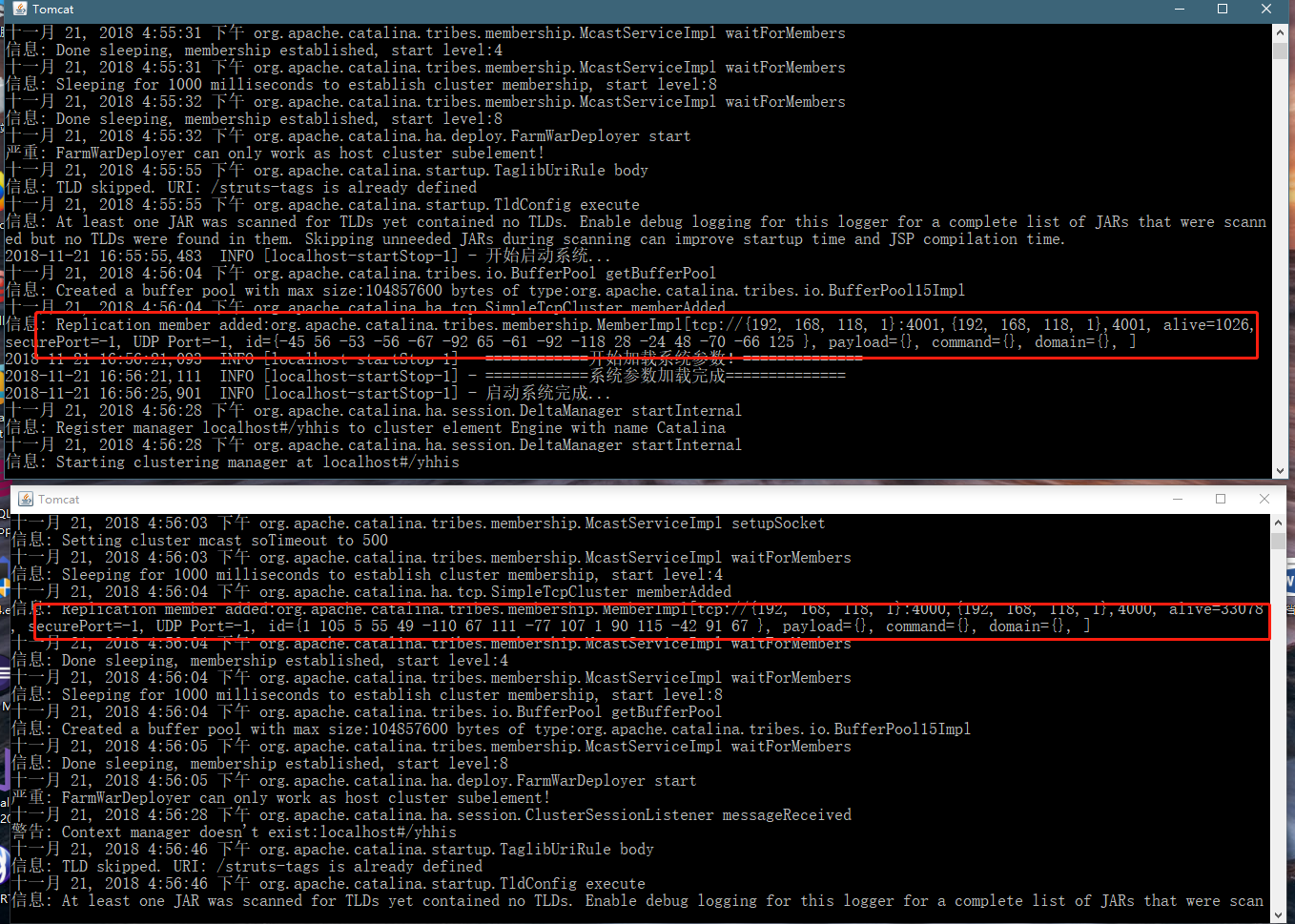
<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>

</Cluster>

Host节点下配置上下文



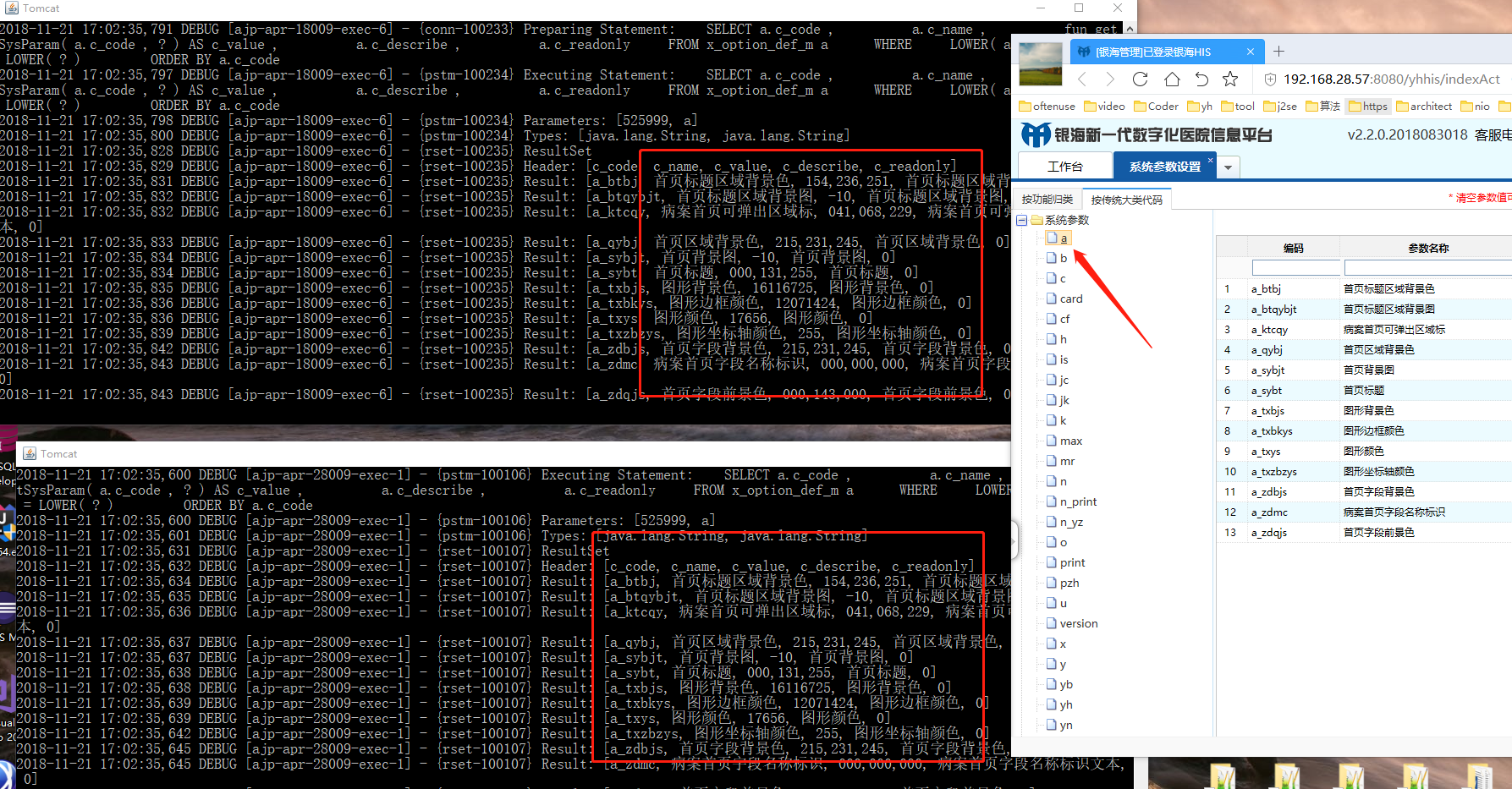
启动两个Tomcat



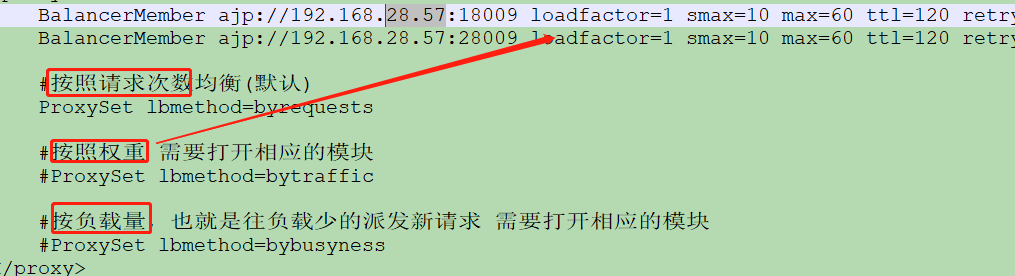
说明通过集群配，他们之间已经通过心跳检测发现对方

进入主页 http://192.168.28.57:8080/yhhis

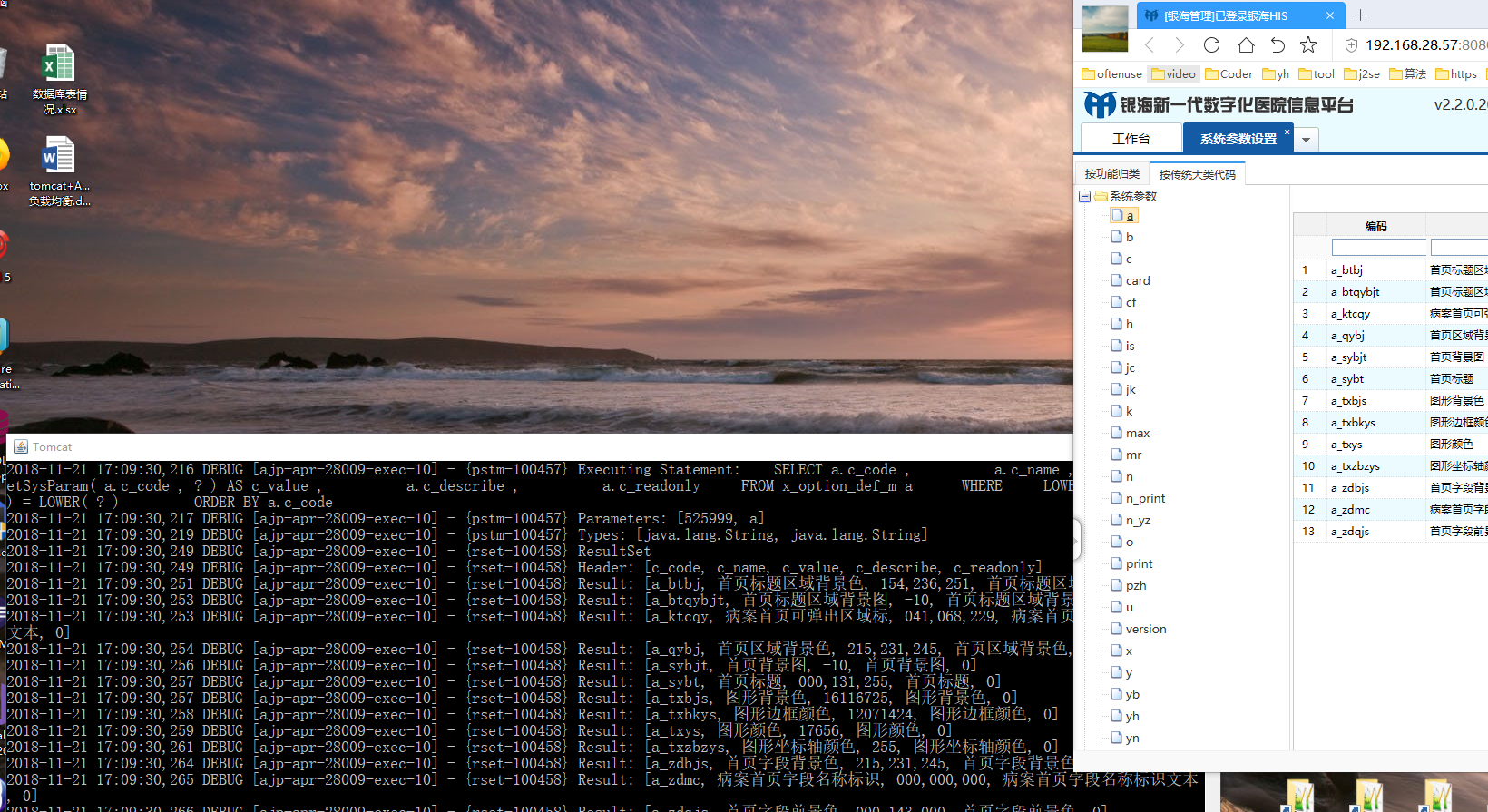




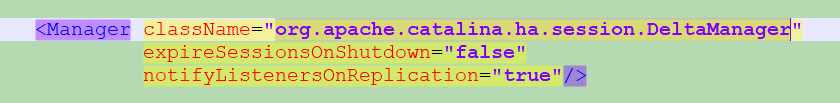
登录请求后，不断请求tomcat会按照配置的调度算法对用户的请求进行分发到相应的后台服务器。达到负责均衡的高性能效果。（三种方案可配置）

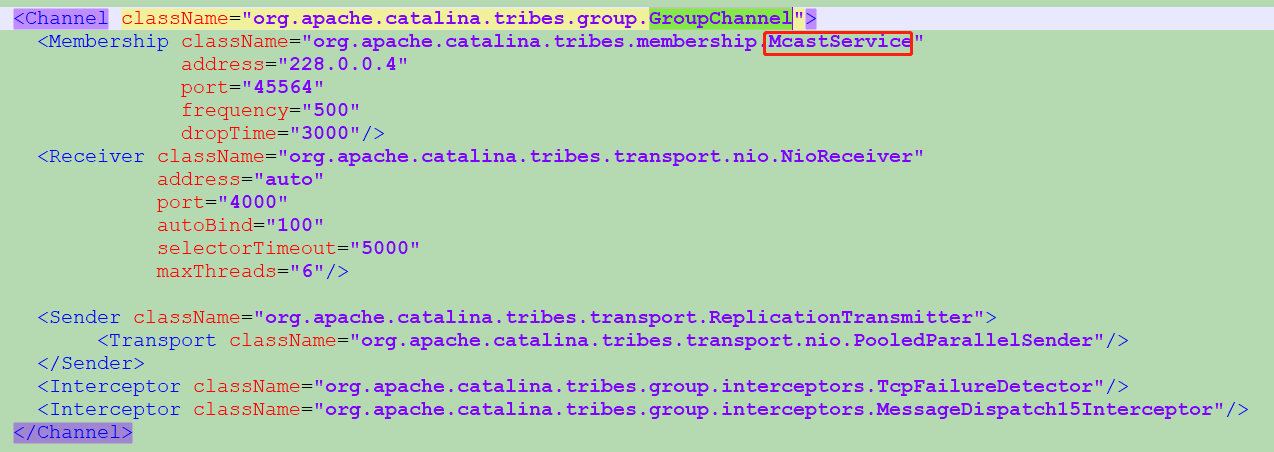


关掉一个tomcat，session不会丢失，用户还是能够进行业务处理，具有高可用。



以后就可以不用全部切断服务器的服务，进行平滑升级。

这里Session保持是通过Tomcat的配置，org.apache.catalina.ha.session.DeltaManager 这个类来管理。

通过多个tomcat之间GroupChannel,McastService

多播，心跳检测来进行通信。建立会话，复制session。所以这种方式Tomcat节点最好不超过4个，否则，用户量上来会多个节点之间多播，心跳检测 ，session复制带来的额外开销，吞吐量不上升。好处： 轻量级，不用考虑Session问题。

关于集群方式多种多样。待续………