**Nginx**

本笔记根据哔哩哔哩尚硅谷视频自学编写注意：本笔记中所记录或者截图中的ip地址均为自己本机或虚拟机的地址，如果需要练习，根据自己的实际ip练习。视频链接：https://www.bilibili.com/video/BV1zJ411w7SV?spm\_id\_from=pageDriver

# 1.Nginx基本概念

## 1.Nginx是什么，能做什么事情

### 什么是Nginx

Nginx（“engine x”）是一个高性能的HTTP和反向代理服务器，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服务器中表现比较好，Nginx转为性能优化而开发，性能是其最重要的考量，实现上非常注重效率，能经受高负载的考验，有报告表明能支持达50000个并发连接数。

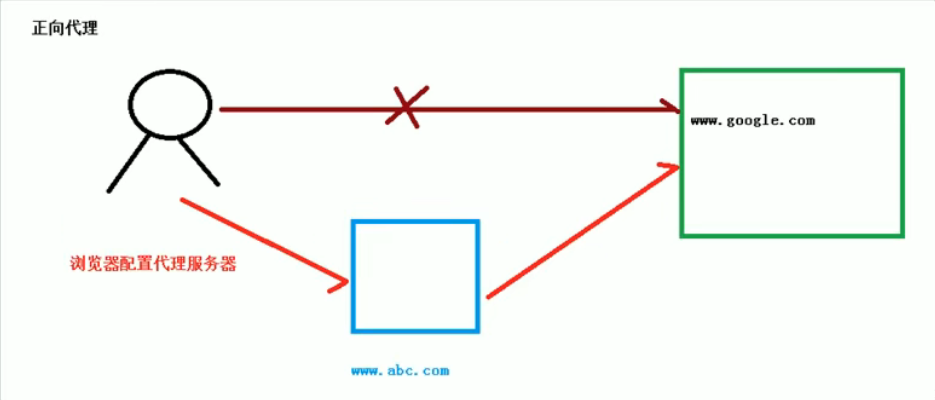
### Nginx作为web服务器

Nginx可以作为静态页面的web服务器，同时还支持CGI协议的动态语言，比如Perl、PHP等。但是不支持Java。Java程序只能通过与Tomcat配合完成。Nginx转为性能优化而开发，性能是其最重要的考量，实现上非常注重效率，能经受高负载的考验，有报告表明能支持达50000个并发连接数。https://lnmp.org/nginx.html

## 2.反向代理

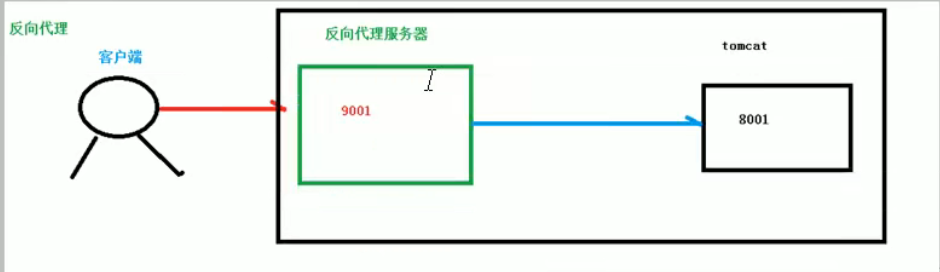
### 1.正向代理

Nginx不仅可以做反向代理，实现负载均衡。还能用作正向代理来进行上网等功能。正向代理：如果把局域网外的Internet想象成一个巨大的资源库，则局域网中的客户端要访问Internet，则需要通过代理服务器来访问，这种代理就成为正向代理。在客户端（浏览器）配置代理服务器，通过代理服务器进行互联网访问。



### 2.反向代理

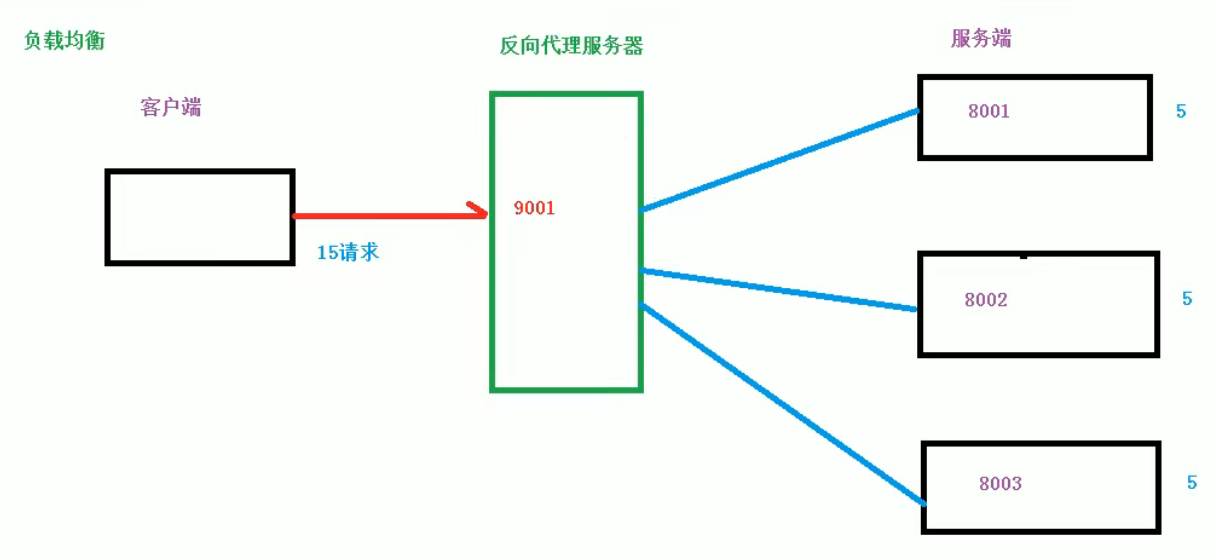
反向代理，其实客户端对代理是无感知的，因为客户端不需要任何配置就可以访问，我们只需要将请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据候，再返回给客户端，此时反向代理服务器和目标服务器对外就是一个服务器，暴露的是代理服务器地址，隐藏了真实服务器IP地址。



## 3.负载均衡

客户端发送多个请求到服务器，服务器处理请求，有一些可能要与数据库进行交互，服务器处理完毕后，再将结果返回给客户端。这种架构模式对于早期的系统相对单一，并发请求相对较少的情况下是比较适合的，成本也低。但是随着信息数量的不断增长，访问量和数据量的飞速增长，以及系统业务的复杂度增加，这种架构会造成服务器相应客户端的请求日益缓慢，并发量特别大的时候，还容易造成服务器直接崩溃。很明显这是由于服务器性能的瓶颈造成的问题，那么如何解决这种情况呢？我们首先想到的可能是升级服务器的配置，比如提高CPU执行频率，加大内存等提高机器的物理性能来解决次问题，但是我们知道摩尔定律的日益失效，硬件的性能提升已经不能满足日益提升的需求了。最明显的一个例子，天猫双十一当天，某个热销商品的瞬时访问量是极其庞大的，那么类似上面的系统架构，将机器都增加到现有的顶级物理配置，都是不能够满足需求的。那么怎么办呢？上面的分析我们去掉了增加服务器物理配置来解决问题的办法，也就是说纵向解决问题的办法行不通了，那么横向增加服务器的数量呢？这时候集群的概念产生了，单个服务器解决不了，我们增加服务器的数量，然后将请求分发到各个服务器上，将原先请求集中到单个服务器上的情况改为将请求分发到多个服务器上，将负载分发到不同的服务器，也就是我们所说的负载均衡。

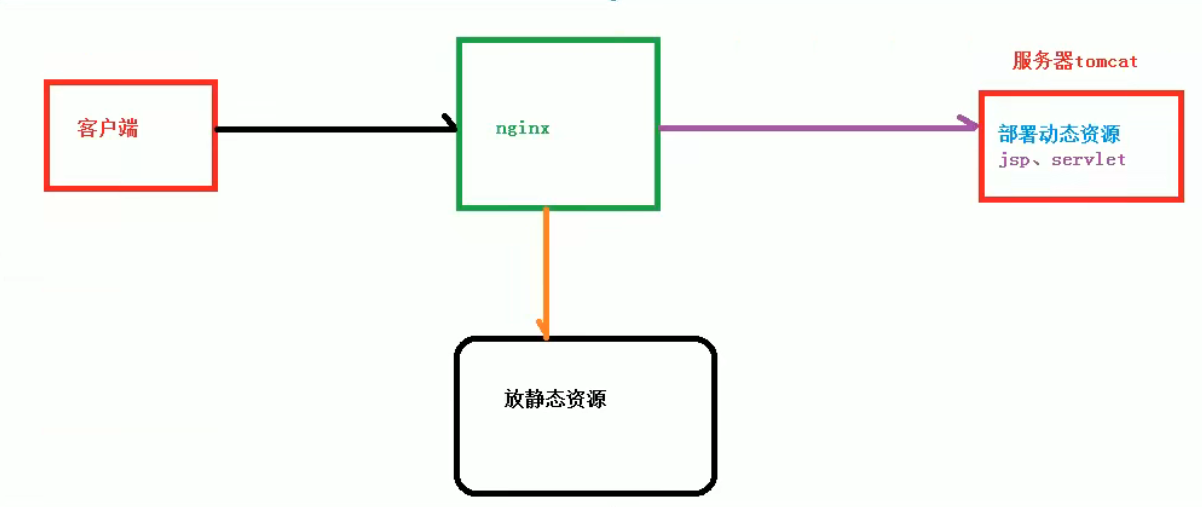
### 



## 4.动静分离

为了加快网站的解析速度，可以把动态页面和静态页面由不同的服务器来解析，加快解析速度。降低原来单个服务器的压力。

### 



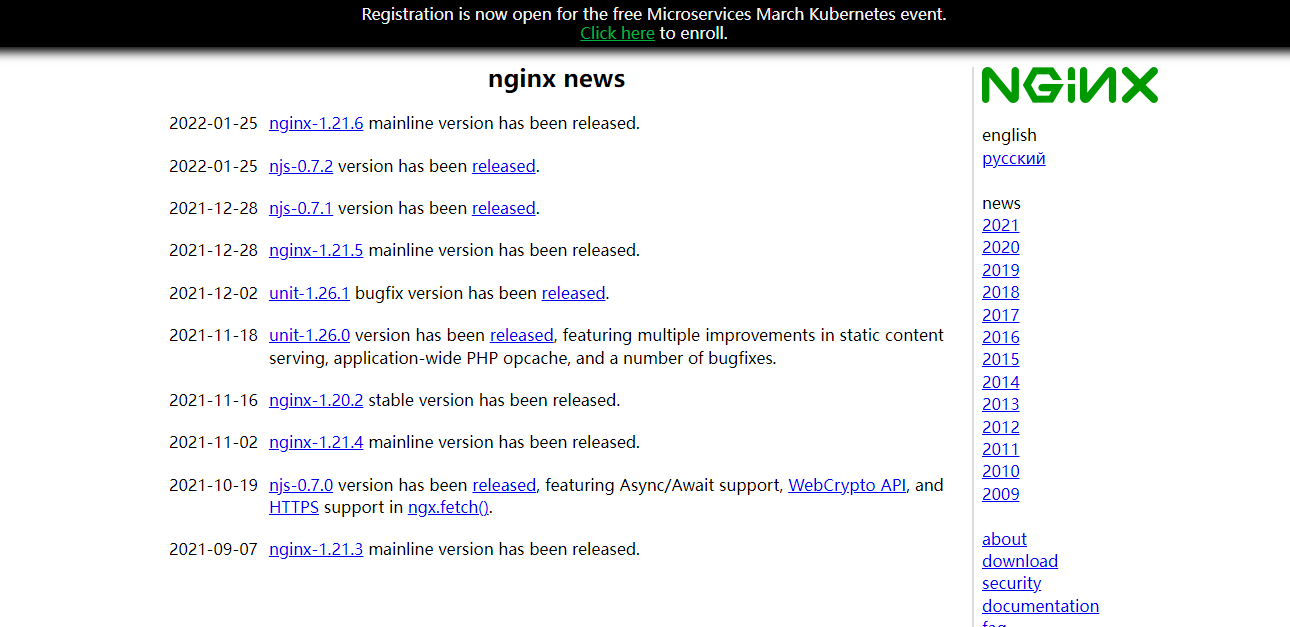
# 2.Nginx安装、常用命令和配置文件

## 1.在linux系统中安装Nginx

### 1.使用远程连接工具连接linux操作系统

### 2.进入官网下载

* http://nginx.org



### nginx相关素材（依赖）

* pcre
* openssl
* zlib
* nginx



* 注：版本不固定

### 安装步骤

* 1.安装pcre依赖

• 1.把安装压缩文件放到linux系统中

• 例如：进入到/usr/src目录下，使用Xshell工具将pcre压缩文件传入该目录下

• 2.解压压缩文件

• 例如：tar -xvf pcre2-10.39.tar.gz

• 3.进入解压之后的目录，执行命令

• ./configure

• 4.执行命令进行编译安装

• make && make install

• 安装完成后查看版本号命令

• pcre-config --version

• pcre2-config --version

* 2.安装其他依赖，采用yum命令安装方式

• yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel

* 3.安装Nginx

• 1.把Nginx安装文件放入到linux系统中

• 例如：进入到/usr/src目录下，使用Xshell工具将nginx压缩文件传入该目录下

• 2.解压压缩文件

• tar -xvf nginx-1.20.2.tar.gz

• 3.进入解压之后的目录，执行命令

• ./configure

• 4.执行命令进行编译安装

• make && make install

### 安装成功后，在/usr中多出一个文件夹/usr/local/nginx，在nginx文件夹中的sbin文件夹中有启动脚本

### 通过ifconfig命令查看地址，并输入到地址栏访问

* linux默认防火墙没有开放端口，需要将要访问的端口号添加到防火墙中开放
* 相关命令

• 查看开放的端口号

• firewall-cmd --list-all

• firewall-cmd --list-ports

• 设置开放的端口号

• firewall-cmd --zone=public --add-port=80/tcp --permanent

• firewall-cmd --zone=public --add-port=[端口号] --permanent

• firewall-cmd --add-port=80/tcp --permanent

• firewall-cmd --add-port=[端口号] --permanent

• 重启防火墙

• firewall-cmd --reload

## 2.Nginx操作的常用命令

### 1.使用nginx操作命令的前提条件：必须进入nginx的目录

* /usr/local/nginx/sbin/

### 2.查看nginx的版本号

* ./nginx -v

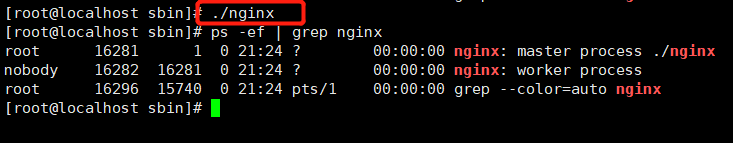
•

desc

### 3.启动nginx

* ./nginx

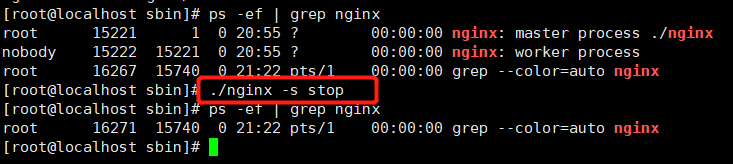
•



### 4.关闭nginx

* ./nginx -s stop

•



### 5.重新加载nginx

* ./nginx -s reload

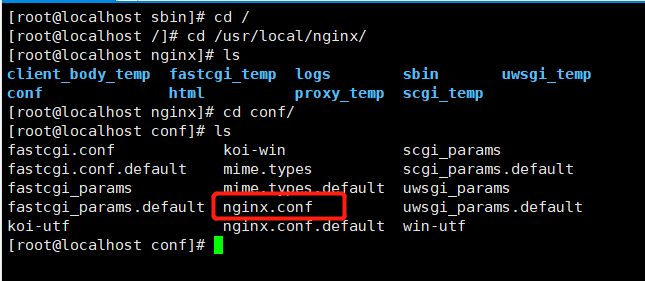
•

desc

## 3.Nginx配置文件

### 1.nginx配置文件在linux系统中的位置

* /usr/local/nginx/conf/nginx.conf



### 2.nginx配置文件组成

* 1.nginx的配置文件有三部分组成

nginx安装目录下，其默认的配置文件都放在这个目录的conf目录下，而主配置文件nginx.conf也在其中，后续对nginx的使用基本上都是对此配置文件进行相应的修改。配置文件中有很多#，开头的表示注释内容，我们去掉所有以#开头的段落，精简之后的内容如图

• 第一部分 全局块

• 从配置文件开始到events块之间的内容，主要会设置一些影响nginx服务器整体运行的配置指令，主要包括配置运行Nginx服务器的用户（组）、允许生成的worker\_processes数，进程PID存放路径、日志存放路径和类型以及配置文件的引入等。

• 比如上面第一行配置的

•

desc

• 这是Nginx服务器并发处理服务的关键配置，work\_processes值越大，可以支持的并发处理量也越多，但是会受到硬件、软件设备的制约

• 第二部分 events块

• events块设计的指令主要影响Nginx服务器与用户的网络连接，常用的设置包括是否开启对多worker\_connections下的网络连接进行序列化，是否允许同时接收多个网络连接，选取哪种事件驱动模型来处理连接请求，每个worker\_connections可以同时支持的最大连接数等。

• 比如

•

desc

• 上述例子就表示每个worker\_connections支持的最大连接数为1024

• 这部分的配置对Nginx的性能影响较大，在实际中应该灵活配置

• 第三部分 http块

•



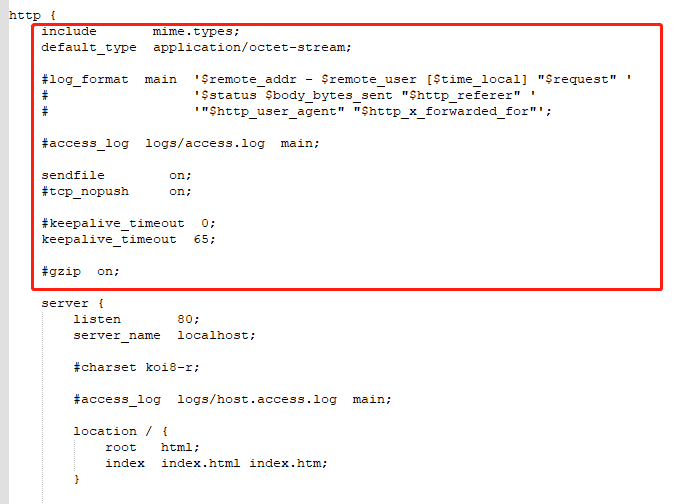
• 这算是Nginx服务器配置中最频繁的部分，代理、缓存和日志定义等绝大多数数功能和第三方模块的配置都在这里。

• 需要注意的是：http块也可以包括http全局快、server块

• http全局块

• http全局块配置的指令包括文件引入、MIME-TYPE定义、日志自定义、连接超时时间、单链接请求数上限等。

•



• server块

• 这块和虚拟主机有密切关系，虚拟主机从用户角度看，和一台独立的硬件主机是完全一样的，该技术的产生是为了节省互联网服务器硬件成本。每个http块可以包括多个server块，而每个server块就相当于一个虚拟主机。而每个server块也分为全局server块，以及可以同时包含多个location块。

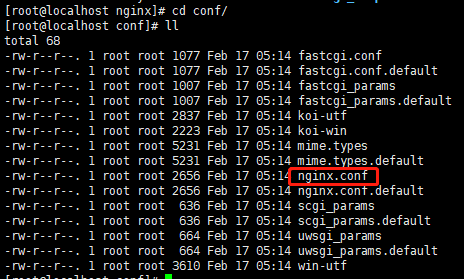
• 1.全局server块

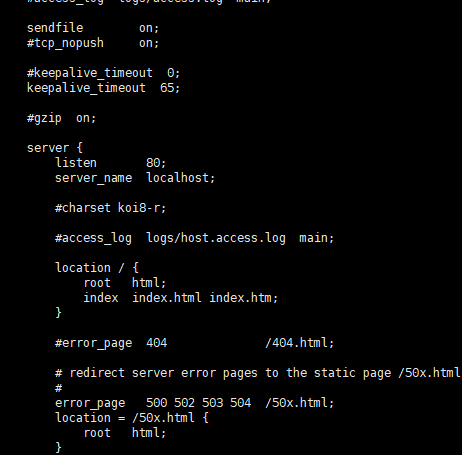
• 最常见的配置是本虚拟机主机的监听配置和本虚拟主机的名称或IP配置。

• 2.location块

• 一个server块可以配置多个location块。

• 这块的主要作用是基于Nginx服务器接收到的请求字符创（例如server\_name/uri-string），对虚拟主机名称（也可以是IP别名）之外的字符串（例如前面的/uri-string）进行匹配，对特定的请求进行处理。地址定向、数据缓存和应答控制等功能，还有许多第三方模块的配置也在这里进行。





# 3.Nginx配置实例

## 1.反向代理实例1

### 1.实现效果

* 打开浏览器，在浏览器的地址栏中输入一个地址，如：www.123.com，跳转到linux系统tomcat主页面中。

### 2.具体实现

* 准备工作

• 1.在linux系统安装tomcat，使用默认端口8080

• tomcat安装文件放入linux系统的/usr/src/目录中，解压

• 进入tomcat的bin目录中，./startup.sh启动tomcat服务器

• 2.对外开放访问的端口

• firewall-cmd --zone=public --add-port=8080/tcp --permanent

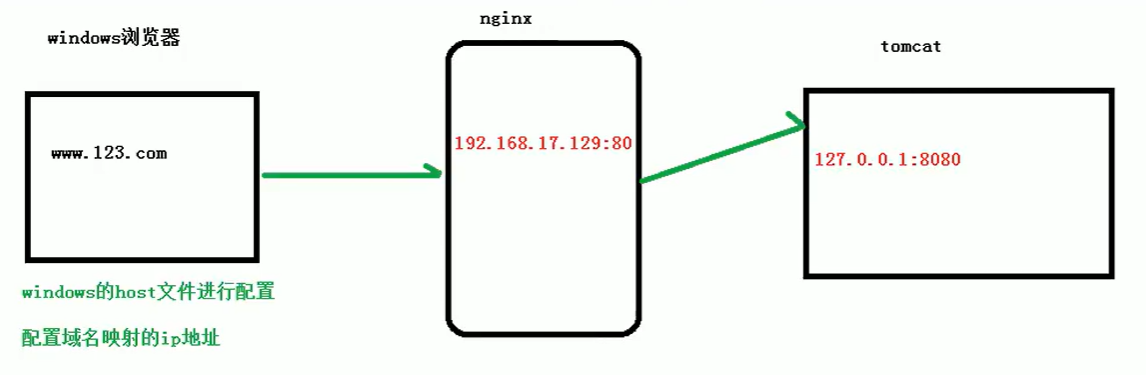
• firewall-cmd --add-port=8080/tcp --permanent

• 重启防火墙

• firewall-cmd --reload

• 3.在windows系统中通过浏览器访问tomcat服务器

### 3.访问过程的分析



* 注意：nginx中的ip地址为linux系统中ifconfig查到的ip地址（下述截图中的ip地址均如此）

### 4.具体配置

* 第一步 在windows系统的hosts文件中进行域名和ip对应关系的配置

• 添加内容在hosts文件中

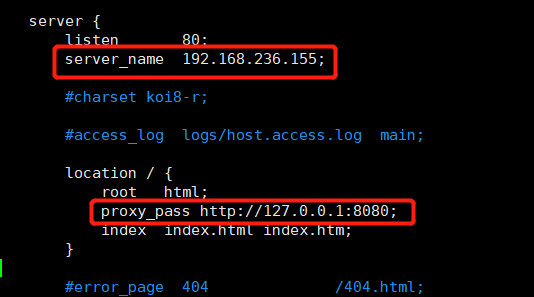
•

desc

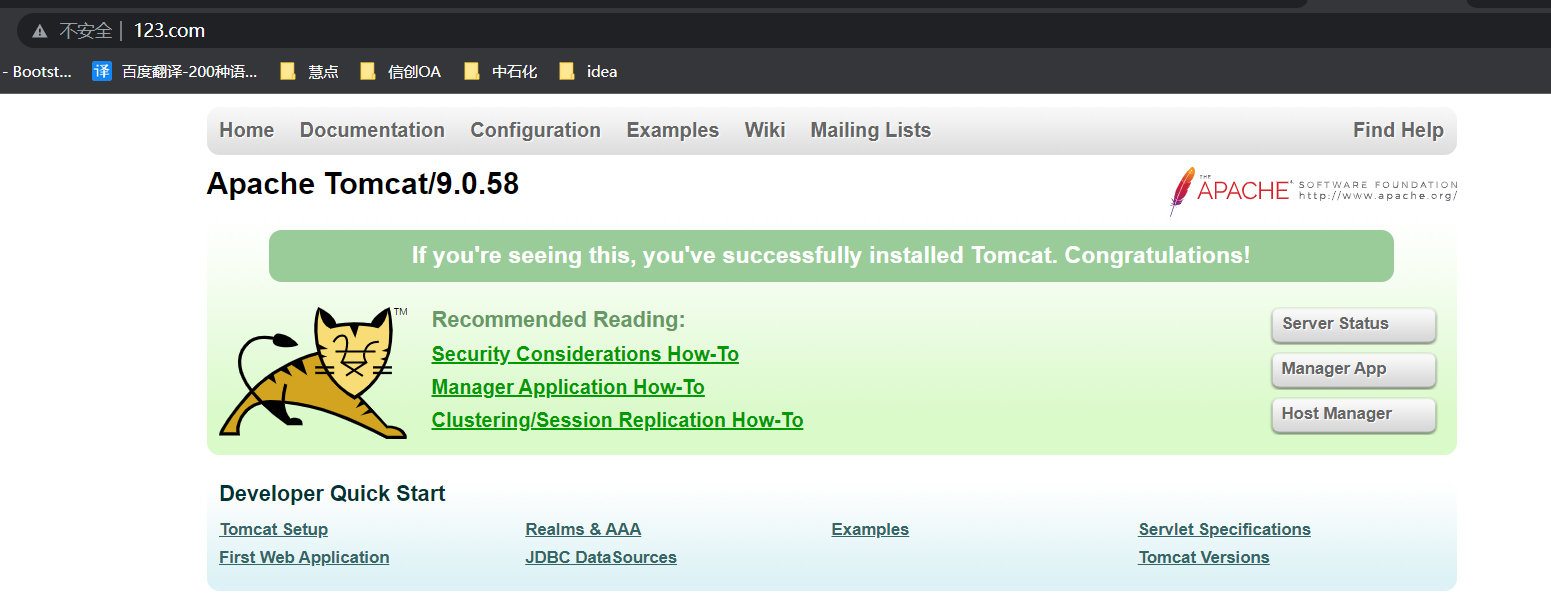
• 注意：nginx中的ip地址为linux系统中ifconfig查到的ip地址

* 第二步 在nginx中进行请求转发的配置（反向代理配置）

•



### 5.最终测试



## 2.反向代理实例2

### 1.实现效果

* 使用nginx反向代理，根据访问路径跳转到不同端口的服务中，nginx监听的端口为9001
* 访问http://192.168.232.155:9001/edu/ 直接跳转到127.0.0.1:8081
* 访问http://192.168.232.155:9001/vod/ 直接跳转到127.0.0.1:8082

### 2.准备工作

* 1.准备两个tomcat服务器，一个8080端口，一个8081端口（注意防火墙中开放的端口）
* 2.创建文件夹和测试页面

• 分别在两个tomcat的webapps下新建文件夹（文件夹名字随便起）/edu(放在8080下)和/vod(放在8081下)

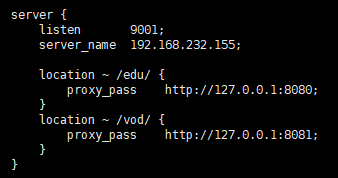
• 新建html文件分别存入上述两个文件夹

• 分别用浏览器访问两个地址：http://192.168.232.155:8080/edu/a.htmlhttp://192.168.232.155:8081/vod/a.html

* 3.具体配置

• 1.找到nginx的配置文件，进行反向代理的配置

•



• location指令说明

• 该指令用于匹配URL。

• 语法如下

• location [= | ~ | ~\* | ^~] uri {}

• 1.=:用于不含正则表达式的uri前，要请求字符串与uri严格匹配，如果匹配成功，就停止继续向下搜索并立即处理该请求。

• 2.~:用于表示uri包含正则表达式，并且区分大小写。

• 3.~\*:用于表示uri包含正则表达式，并且不区分大小写。

• 4.^~:用于不含正则表达式的uri前，要求nginx服务器找到标识uri和请求字符串匹配度最高的location后，立即使用此location处理请求，而不再使用location块中的正则uri和请求字符串做匹配。

• 注意：如果uri包含正则表达式，则必须要有~或者~\*标识。

• 2.开放对外访问的端口号，目前需要开放的端口号有：9001，8080,8081,80

* 4.最终测试

•



•



## 3.负载均衡

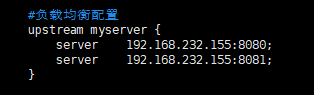
### 1.实现效果

* 1.通过浏览器的地址栏输入地址http://192.168.232.155/edu/a.html，负载均衡效果，把请求平均分配到8080和8081的端口

### 2.准备工作

* 1.准备两台tomcat服务器，一台8080，一台8081
* 2.在两台tomcat里面webapps目录中，分别创建文件夹/edu(放在8080和8081中)，分别在文件夹中创建a.html文件，用于测试。

### 3.在nginx的配置文件中进行负载均衡的配置





* 注意防火墙开放的端口号
* 随着互联网信息的爆炸性增长，负载均衡（load balance）已经不再是一个很陌生的话题，顾名思义，负载均衡即是将负载分摊到不同的服务单元，既保证服务的可用性，又保证响应足够快，给用户很好的体验。快速增长的访问量和数据流量催生了各式各样的负载均衡产品，很多专业的负载均衡硬件提供了很好的功能，但价格却不菲，这使得负载均衡软件大受欢迎，Nginx就是其中一个，在linux下有Nginx、LVS、Haproxy等等服务可以提供负载均衡服务，而且nginx提供了几种分配方式（策略）

• 1.轮询（默认）

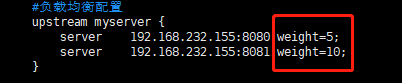
• 每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。

• 2.weight

• weight代表权重，默认为1，权重越高，被分配的客户端越多。

• 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。比如

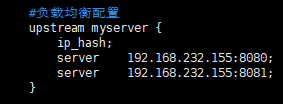
•



• 3.ip\_hash

• 每个请求按访问ip的hash结果分配，这样每个访客固定一个后端服务器，可以解决session共享的问题。

•

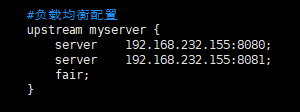


• 4.fair（第三方）

• 前提：需要安装fair模块

• 按后端服务器端的相应时间来分配请求，相应时间短的优先分配。

•



### 4.最终测试

* 在地址栏中输入http://192.168.236.155:9002/edu/a.html会平均将请求分配到两个tomcat中

•



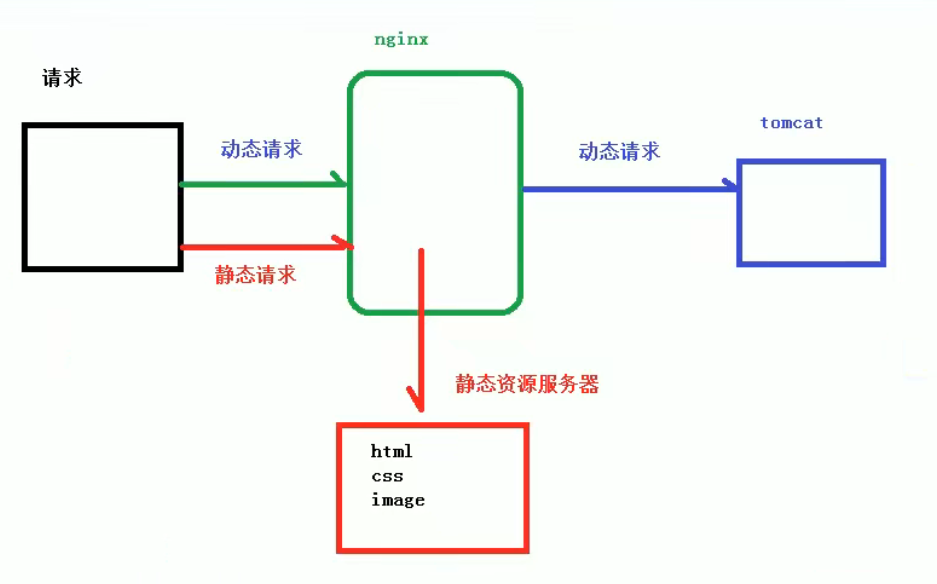
•



## 4.动静分离

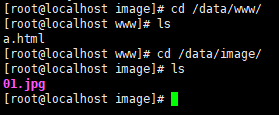
### 1.什么是动静分离

* Nginx动静分离简单来说就是把动态跟静态请求分开，不能理解成只是单纯的把动态页面和静态页面物理分离。严格意义上说应该是动态请求跟静态请求分开，可以理解成使用Nginx处理静态页面，Tomcat处理动态页面。动静分离从目前实现角度来讲大致分为两种，一种是纯粹把静态文件独立成单独的域名，放在独立的服务器上，也是目前主流推崇的方案；另一种方法就是动态跟静态文件混合在一起发布，通过nginx来分开。 通过location指定不同的后缀名实现不同的请求转发。通过expires参数设置，可以使浏览器缓存过期时间，减少与服务器之前的请求和流量。具体Expires定义：是给一个资源设定一个过期时间，也就是说无需去服务端验证，直接通过浏览器自身确认是否过期即可，所以不会产生额外的流量。此种方法非常适合不经常变动的资源。（如果经常更新的文件，不建议使用Expires来缓存），我这里设置3d，表示在这3天之内访问这个URL，发送一个请求，比对服务器该文件最后更新时间没有变化，则不会从服务器抓取，返回状态码304，如果有修改，则直接从服务器重新下载，返回状态码200。



### 2.准备工作

* 在linux系统中准备一些静态资源，用于进行访问



### 3.具体配置



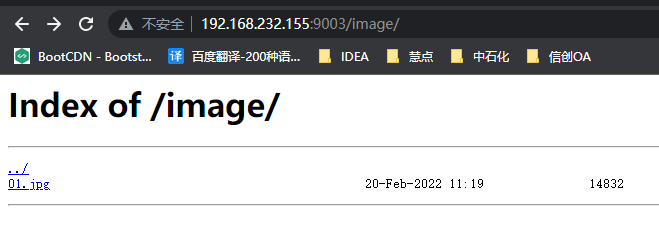
* 注意配置中端口是否为防火墙开放端口
* 添加监听端口、访问名字。重点是添加location，最后检查Nginx配置是否正确，然后测试动静分离是否成功，之后需要删除后端tomcat服务器上的某个静态资源，查看是否能访问，如果可以访问说明是静态资源nginx直接返回了，不走后端tomcat服务器。

### 4.最终测试

* 1.浏览器中输入地址

• http://192.168.232.155:9003/image/

• 因为在配置文件中配置过autoindex on;所以会将image里面的文件列出来。



* 2.在浏览器中输入地址

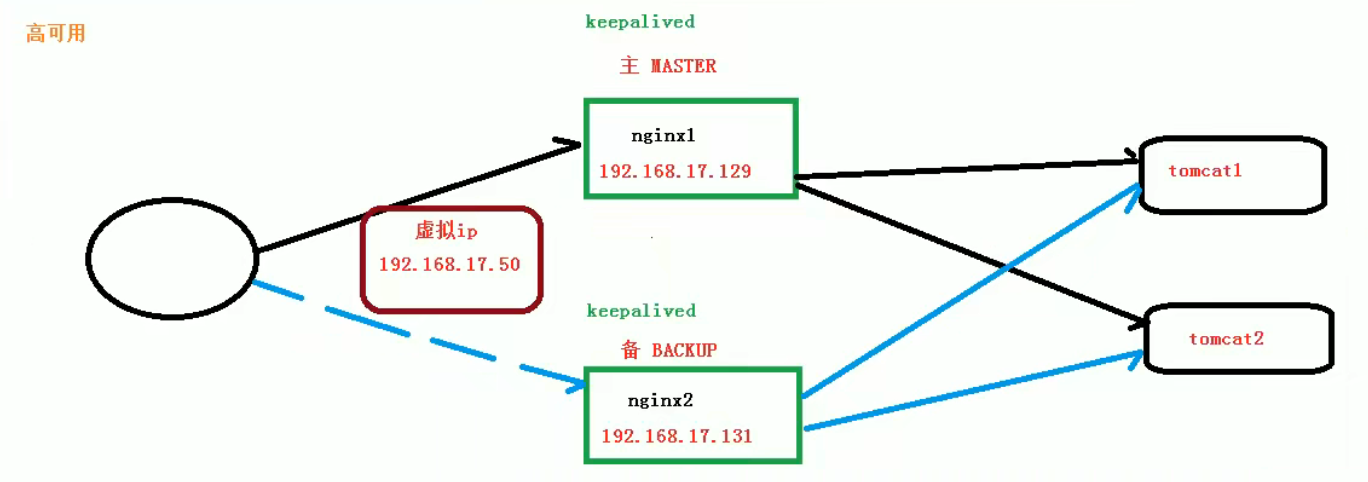
•



# 4.Nginx配置高可用集群

## 1.什么是nginx高可用

### 



### nginx高可用就是服务器挂掉了，系统依然可以正常完成请求的过程。

### 1.需要两台nginx服务器

### 2.需要Keepalived

### 3.需要虚拟ip

## 2.配置高可用的准备工作

### 1.需要两台服务器（两个虚拟机环境）

* 192.168.232.155
* 192.168.232.156

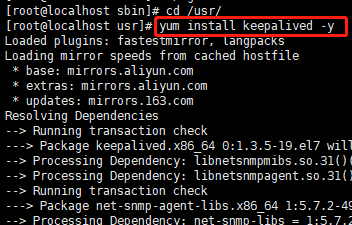
### 2.在两台服务器上安装nginx

### 3.在两台服务器上安装keepalived

## 3.在两台服务器上安装keepalived

### 1.使用yum命令安装或者提前下载好安装包进行解压安装

* yum install keepalived -y



### 2.安装之后，在/etc里面自动生成/keepalived，有文件keepalived.conf

## 4.完成高可用配置（主从配置）

### 1.修改/etc/keepalived/keepalived.con配置文件

* #全局配置global\_defs { notification\_email { acassen@firewall.loc failover@firewall.loc sysadmin@firewall.loc } notification\_email\_from Alexandre.Cassen@firewall.loc smtp\_server 192.168.232.155 smtp\_connect\_timeout 30 router\_id LVS\_DEVEL#访问到主机的名字，linux中在/etc/host中配置，如当前实例在linux中配置为：127.0.0.1 LVS\_DEVEL}#脚本相关配置vrrp\_script chk\_http\_port {script "/usr/local/src/nginx\_check.sh"interval 2# (检测脚本执行的间隔)，单位：秒weight 2#权重，如果值为-20，当脚本中的条件成立，那么把当前主机的权重降低20}#虚拟IP配置vrrp\_instance VI\_1 { state MASTER#设置服务器是主服务器还是备份服务器，主服务器值为 MASTER ， 备份服务器值为 BACKUP interface ens33//网卡名称，在linux中输入ifconfig，查看绑定的网卡名称 virtual\_router\_id 51#主、备机的virtual\_router\_id必须相同，相当于主备机的唯一标识id priority 100#主、备机取不同的优先级，主机值较大，备份机值较小 advert\_int 1#时间间隔，标识每隔多少秒检测服务器是否还活着，默认是1秒 authentication {#校验权限的方式 auth\_type PASS#采用密码形式 auth\_pass 1111#密码是1111 } virtual\_ipaddress { 192.168.232.50// VRRP H虚拟地址，前三个地址需要和主机在同一个网段 }}

### 2.在/usr/local/src添加脚本

* #！/bin/bashA=`ps -C nginx -no-header |wc =l`if [ $A -eq 0 ];then /usr/local/nginx/sbin/nginx#启动nginx脚本的位置 sleep 2 if [ `ps -C nginx -no-header |wc -l` -eq 0 ];then killall keepalived fifi

### 3.把两台服务器上的nginx和keepalived启动起来

* 启动nginx

• ./nginx

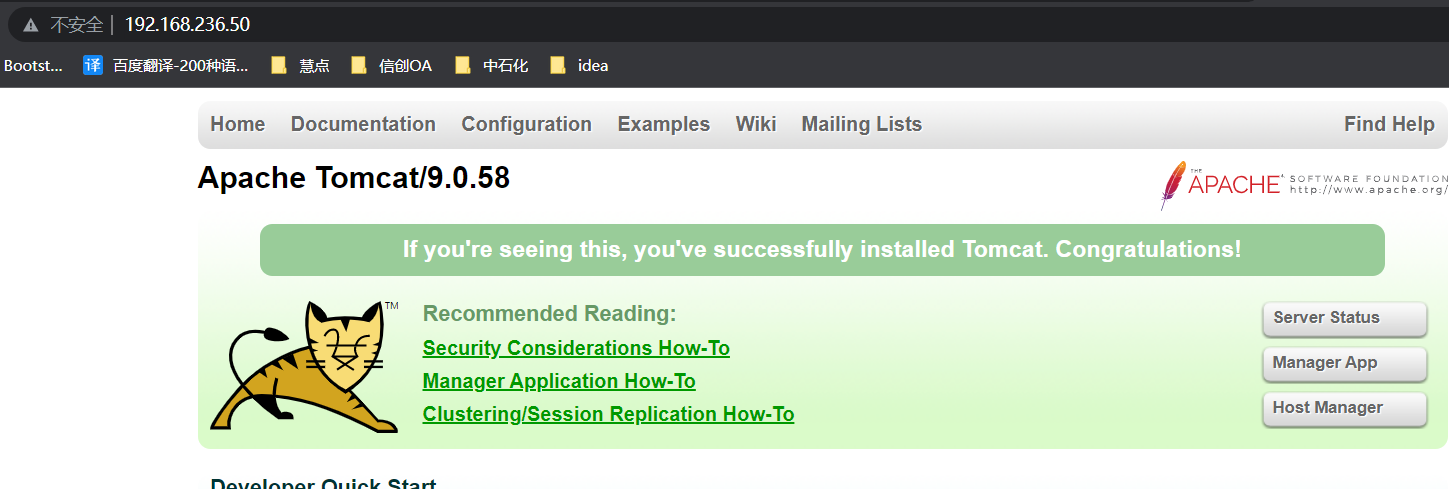
* 启动keepalived

• systemctl start keepalived.service

## 5.最终测试

### 1.在浏览器地址栏中输入虚拟ip地址

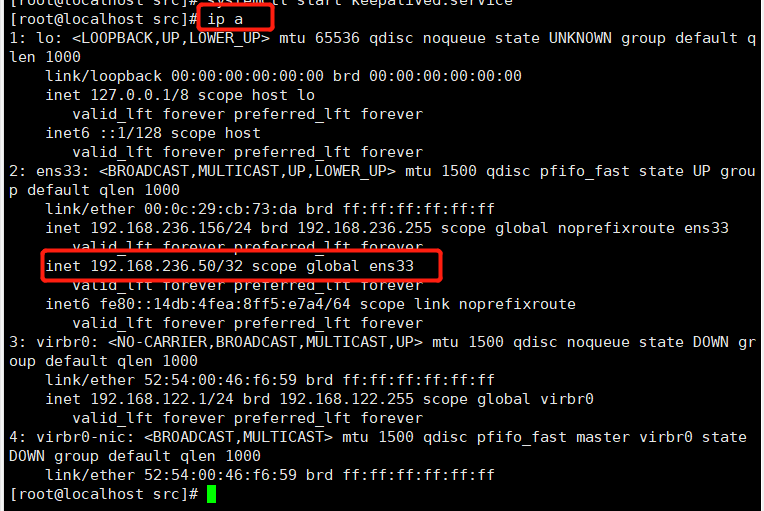
* 192.168.236.50

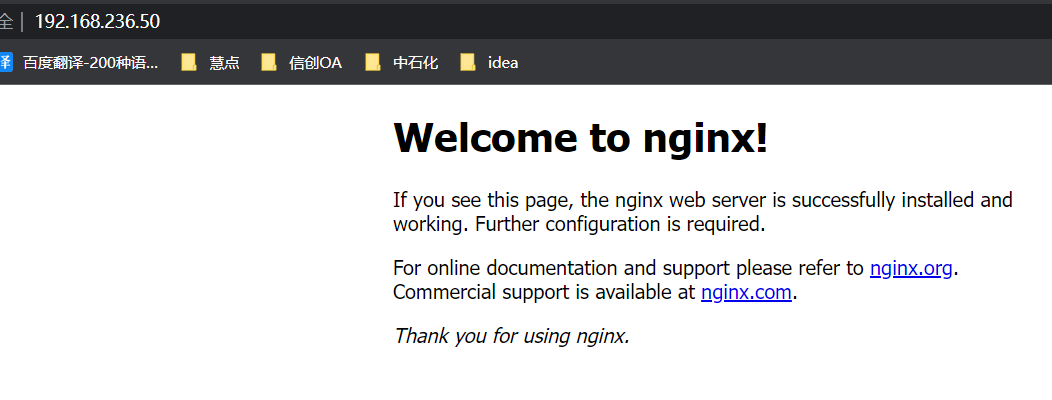


* 访问到的是tomcat页面原因是nginx.conf里面的配置

### 2.把主服务器(192.168.236.155)的nginx和keepalived停止，再在浏览器地址栏中输入虚拟ip地址

* 192.168.236.50



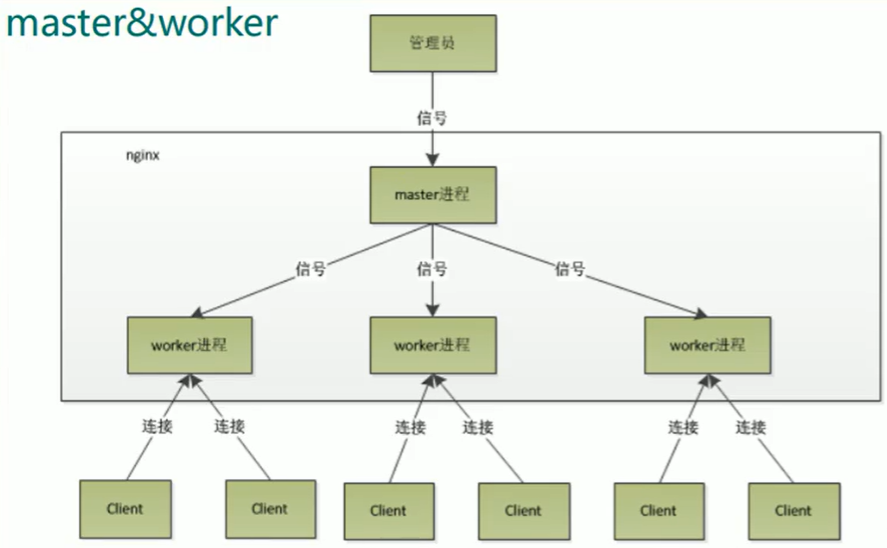


* 注意两台机器的防火墙开放端口问题

# 5.Nginx原理

## 1.master和worker

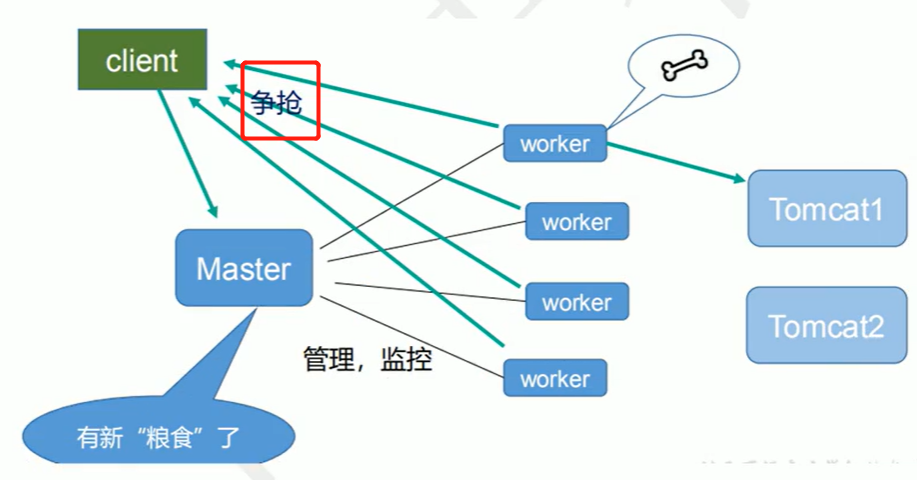
### 



### 启动nginx后有两个进程，master和worker，worker进程可以有多个

## 2.worker如何进行工作的

### 



## 3.一个master和多个worker的机制的好处

### 1.可以使用命令./nginx -s reload 热部署，利于nginx进行热部署操作

### 2.对于每个worker进程来说，独立的进程，不需要加锁，所以省掉了锁带来的开销，同时在编程以及问题查找时，也会方便很多。其次，采用独立的进程，可以让互相之间不会影响，一个进程退出后，其他进程还在工作，服务不会中断，master进程则很快启动新的worker进程。当然，worker进程的异常退出，肯定是程序有bug了，异常退出，会导致当前worker上的所有请求失败，不过不会影响到所有请求，所以降低了风险。

## 4.设置多少个worker最合适

### nginx和redis类似都采用了io多路复用机制，每隔worker都是一个独立的进程，但每个进程里只有一个主线程，通过异步非阻塞的方式来处理请求，即使是成千上万个请求也不在话下。每个worker的线程可以把一个cpu的性能发挥到极致。所以worker数和服务器的cpu数相等是最为适宜的。设少了会浪费cpu，设多了会造成cpu频繁切换上下文带来的损耗。

## 5.连接数worker\_connection

### 设置worker数量

* worker\_processes 4
* #

### work绑定cpu(4 worker 绑定 4cpu)

* worker\_cpu\_affinity 0001 0010 0100 1000

### work绑定cpu(4 worker 绑定 8cpu 中的4个)

* 00000001 00000010 00000100 00001000

### 连接数worker\_connection

* 这个值是表示每个worker进程所能建立连接的最大值，所以，一个nginx能建立的最大连接数，应该是worker\_connections \* worker\_processes。当然，这里所说的最大连接数，对于HTTP请求本地资源来说，能够支持的最大并发数量是worker\_connections \* worker\_processes，如果是支持http1.1的浏览器每次访问要占两个链接，所以普通的静态访问最大并发数量是：worker\_connections \* worker\_processes / 2，而如果是HTTP作为反向代理来说，最大并发数量应该是worker\_connections \* worker\_process / 4。因为作为反向代理服务器，每个并发会建立与客户端的连接和后端服务器的连接，会占用两个连接。

### 发送一个请求，占用了worker的几个连接数？答案：2个或4个。2个：客户端发请求---->worker---->客户端4个：客户端发请求---->worker----->tomcat---->worker---->客户端

### 问题二：nginx有一个master，有4个worker，每个worker支持最大的连接数是1024，那么worker支持的最大并发数是多少？普通的静态访问最大并发数量:worker\_connections \* worker\_processes / 24 \* 1024 / 2 = 2048而如果是HTTP作为反向代理来说，最大并发数量:worker\_connections \* worker\_process / 44 \* 1024 / 4 = 1024

# Xshell中文件传输命令

## 从linux系统将文件传到windows系统中

### sz 文件名

### 然后选择文件存放位置

## 从windows系统将文件传到linux系统中

### rz

### 在弹出框上选择要上传的文件