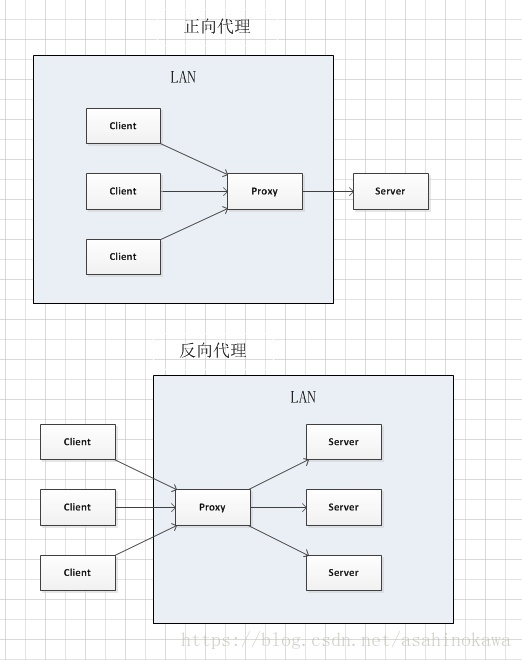
## 反向代理



正向代理是你（客户端）通过“中介”去找别人（服务端）；反向代理是别人（服务端）通过“中介”来找你（客户端）。但前一个中介的任务是牵线搭桥，他起的作用是决定性的，没他就不行；后一个中介的任务是分类、整理、派发，没他并不是不行，而是太乱了，眼花缭乱无所适从。这时这个中介就像太监帮皇上挑妃子，先整理了一大堆牌子，然后让你选择今晚翻谁牌子。

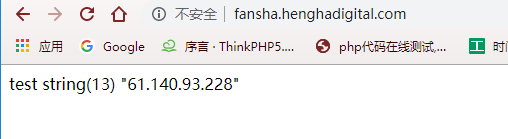
其实我觉得反向代理这个名字起的不太好，至少在中文里，使人感觉反向代理和正向代理是同类事物，其实它们是两回事。正向代理是一个跳板，帮你上网的。是指A无法直接达到C，所以A先联系B，由B去联系C，这样，就实现了A和C的联通。B就是代理了。

但反向代理只是服务请求的转发而已，它应该和DDNS归为同类事物。只不过DDNS需要加端口号。反向代理不需要端口号，而只需要变换三级域名。比如你打开搜狐，搜狐有首页，有邮箱，有新闻……不同的服务，只需要变换http://XXX.sohu.com中的XXX部分而已，邮箱就是mail、新闻就是news。而DDNS是改端口号，http://www.sohu.com：XXXX，不同的服务不同的端口号。

**实操一下：**

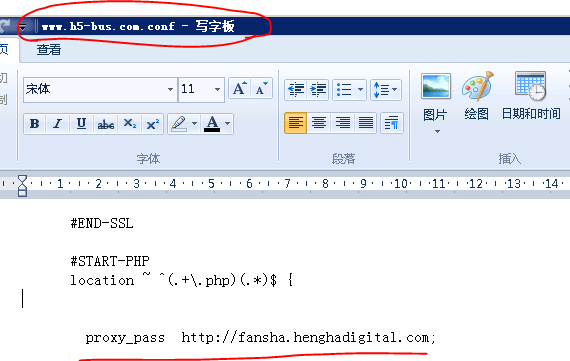
[www.h5-bus.com](http://www.h5-bus.com) = A网站

fansha.henghadigital.com = B网站

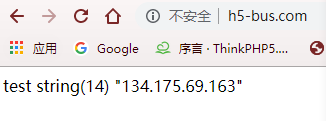


直接访问这个域名，打印出ip。

然后配置A网站 反向代理，



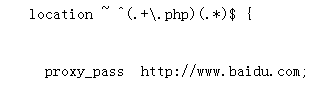
然后访问A网站

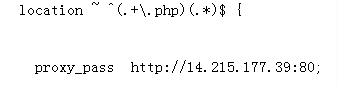


可以看到A网站 把请求转发到了 B网站，然后 B网站又把结果返回给 A网站 ， A网站再把结果返回给客户端。

但是可以看到，**并不是原封不动的转发，**客户端ip变成了代理服务器的Ip。

所以说其实反向代理就是一个转发





这样配置后，我访问原来自己的url就直接跳到百度了（上面的ip是百度的ip）。端口可填可不填。但是后面不要加”/”。

## 负载均衡

**疑问：**集群下的session，cookie，redis，mysql如何共享

## 动静分离

**描述：**

Nginx配置文件组成架构，主要由三部分组成，分别是全局块、event块、http块

## 热部署

**描述：**直白点的意思就是不重启nginx来重新加载配置。就是改了nginx配置的时候用的。

**实操：**

1. 进入安装目录
2. 

## 配置

**描述：**

Nginx配置文件组成架构，主要由三部分组成，分别是全局块、event块、http块

**全局块：**

主要负责会影响nginx整体的配置。如：nginx的用户和用户组、worker process（允许生成的进程数）、进程pid存放路径、日志类型及存放路径、配置文件的引入等。

**event块：**

主要负责服务端的nginx整体和客户端网络连接方面的配置。如：最大连接数等。

（注释：全局块和event块都是针对nginx整体配置的，http块是针对项目配置的）

**http块（重点部分！！！）：**

1. 最主要最频繁的配置部分。代理、缓存、日志定义等绝大多数功能和第三方模块的配置都在这里。
2. Http块也可以分成http全局块、server块。

**http全局块：**

主要负责文件引入、MIME-TYPE定义、链接超时时间、单链接请求数上限等以及http协议有关的设置（如允许添加的请求头，允许连接的ip域名等）。

**server块：**

这里是设置虚拟主机的，我喜欢叫虚拟主机块，对应虚拟主机的配置都在块下。

每个http块可以包括多个server块，每个server块相当于一个虚拟主机。而决定使用哪一个server块是通过域名和端口决定的。

而每个server块下也可分为server全局块，和多个location块。

**server全局块：**

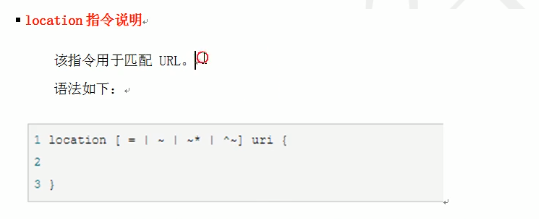
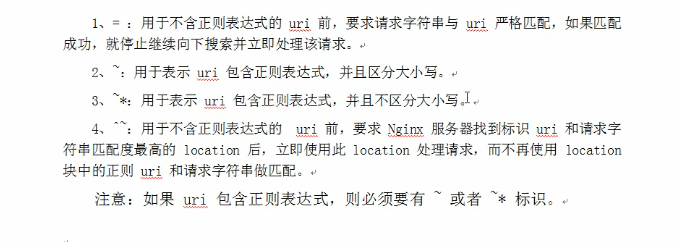
也就是针对这个域名和端口的虚拟主机的配置。常见的是本虚拟主机的名称、IP、监听配置等。

**location块：**

这一块是针对域名后面的部分做判断的。

如[www.xxx.com/index/index/index.php中的index/index/index.php](http://www.xxx.com/index/index/index.php中的index/index/index.php)部分，

对命中判断的请求进行处理。重定向、数据缓存、应答控制，以及第三方模块的配置也在这里。

## 高可用

**描述：**

如果是应用服务器挂了，那nginx能检测到，分发到其他应用服务器便是。

那如果分发服务器挂了，咋整？

弄个备份分发服务器，

用一个keepalived的软件和整一个虚拟ip出来绑定主备两台分发服务器，

**疑问：**

1、什么是高可用？

答：就是容灾性要强，如宕机等情况依然能正常访问。

2、虚拟ip绑定到两台服务器咋整?

答：

1. 服务器都挂了，这个keepalived是怎么起作用的？

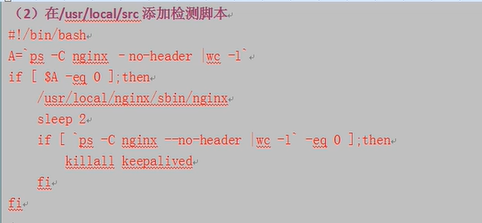
答：这个keepalived只能防止nginx服务挂了，不是服务器挂了，服务器直接挂了这个Keepalived就没办法了。而至于有没有应对这个服务器挂的方法应该是有的，只是这个keepalived不行。

4、keepalived配置使用

答：



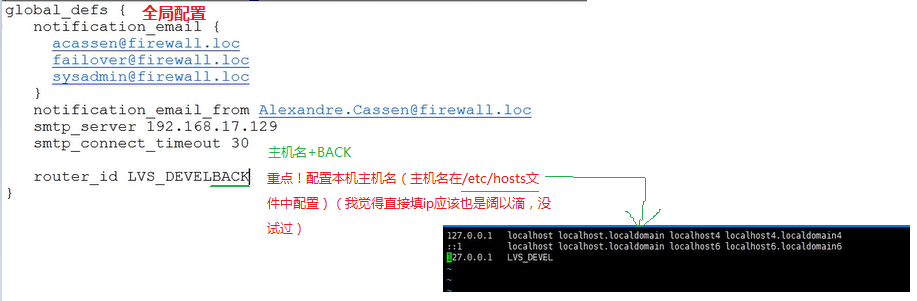




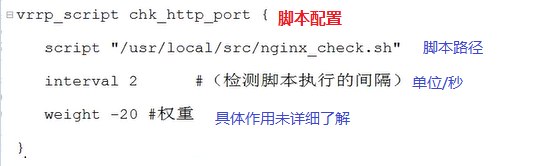
**Keepalived配置：**

配置主要分成三大部分，

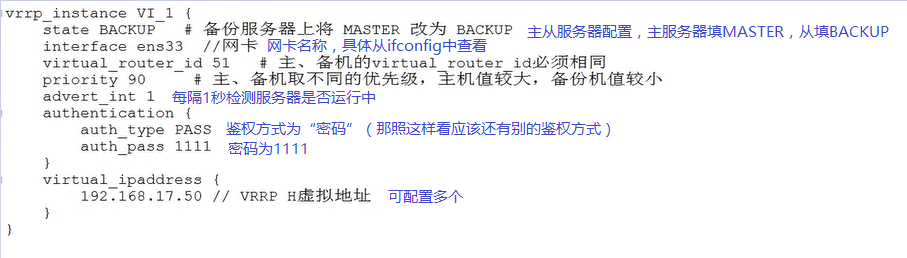
全局部分：



脚本部分：



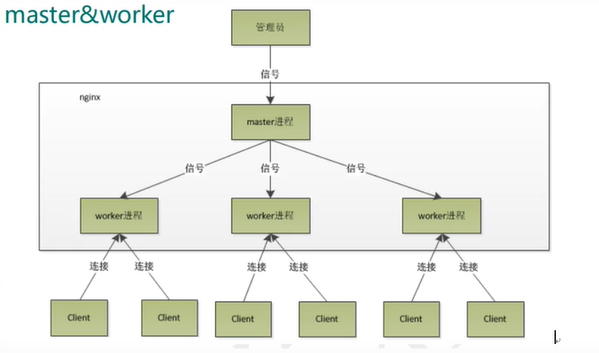
虚拟ip配置部分：



## 基本运行原理

**描述：**

先看看大致原理如下图，

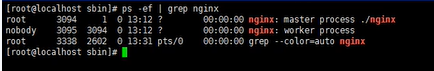


运行nginx后，会产生两个或多个进程，分别是master进程和worker进程

具体产生多少个worker进程，由nginx配置中全局块的worker process决定。

（每个worker进程可以把一个cpu的性能发挥到极致，座椅worker数设置为和服务器的cpu数相等是最为适宜的。设少了浪费cpu，设多了会造成cpu频繁上下文切换带来的损耗）

master进程负责分配接收到的请求，worker进程负责处理。



**疑问：**

1. 连接数worker\_connections的配置？发送一个请求占用了多少个worker的连接数？

答：2个或者4个。

如果访问的是静态资源，那直接由nginx转发到对应静态文件即可。牵涉者只有nginx和客户端。输入输出各占1条连接，所以是占用2个连接数。

如果访问的是动态文件，如php、java等后端脚本文件。则nginx需要连接相应的语言的解析器（php-cgi，tomcat等）去解析，所以牵涉者就有nginx、客户端、解析器这3者。同样输入输出各占1条连接，所以是4个连接数。

所以

动态资源请求并发数 = worker process \* worker\_connections/4

静态资源请求并发数 = worker process \* worker\_connections/2