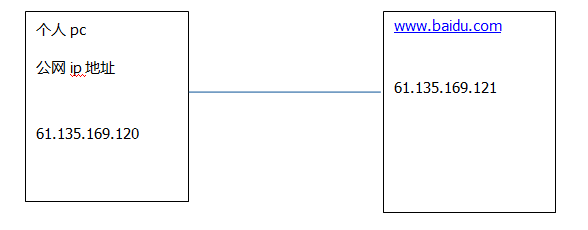
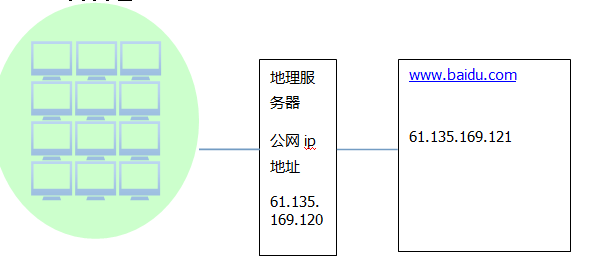
# Nginx的反向代理



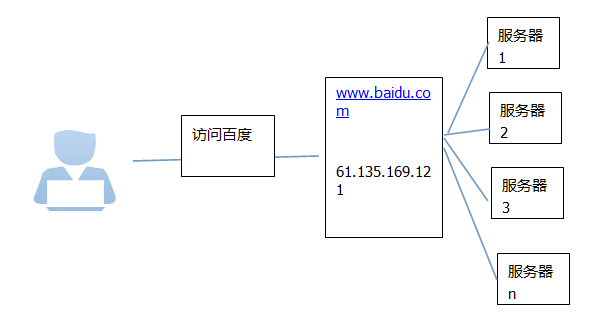
## 正向代理

正向代理，意思是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，**客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。**客户端才能使用正向代理。



## 反向代理

**反向代理（Reverse Proxy）方式是指以**[**代理服务器**](http://baike.baidu.com/view/751.htm)**来接受internet上的连接请求，**然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

**使用nginx实现反向代理**

Nginx只做请求的转发，后台有多个http服务器提供服务，nginx的功能就是把请求转发给后面的服务器，决定把请求转发给谁。

### 安装tomcat

在一个虚拟机上创建两个tomcat实例，模拟多个服务器。

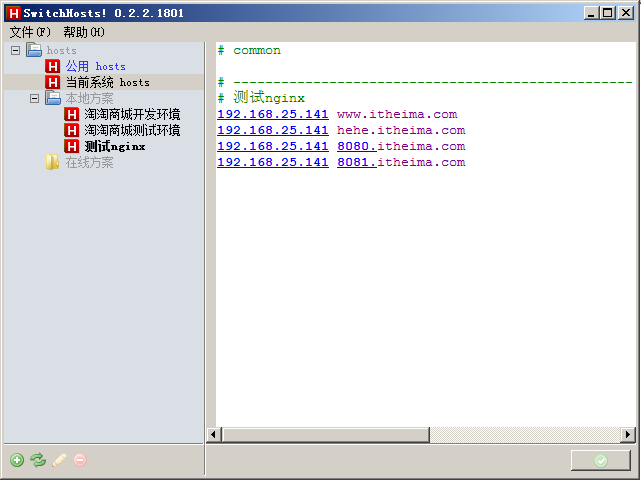
### 需求

通过访问不同的域名访问运行在不同端口的tomcat

8080.itheima.com 访问运行8080端口的tomcat

8081.itheima.com 访问运行8081端口的tomcat

### 域名需要配置host文件：



### Nginx的配置

|  |
| --- |
| **upstream tomcatserver1 {**  **server 192.168.25.141:8080;**  }  **upstream tomcatserver2 {**  **server 192.168.25.141:8081;**  }  server {  listen 80;  **server\_name 8080.itheima.com;**  #charset koi8-r;  #access\_log logs/host.access.log main;  **location / {**  **proxy\_pass http://tomcatserver1;**  **index index.html index.htm;**  }  }  server {  listen 80;  **server\_name 8081.itheima.com;**  #charset koi8-r;  #access\_log logs/host.access.log main;  **location / {**  **proxy\_pass http://tomcatserver2;**  **index index.html index.htm;**  }  } |

**如果在同一个域名下有多台服务器提供服务，此时需要nginx负载均衡。**

# 负载均衡

## 什么是负载均衡

**负载均衡 建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。**

负载均衡，英文名称为Load Balance，其意思就是分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。

## 需求

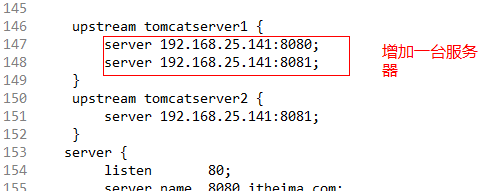
nginx作为负载均衡服务器，用户请求先到达nginx，再由nginx根据负载配置将请求转发至 tomcat服务器。

nginx负载均衡服务器：192.168.25.141

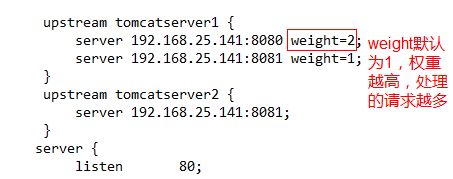
tomcat1服务器：192.168.25.141:8080

tomcat2服务器：192.168.25.141:8081

## 配置nginx的负载均衡



## 配置负载均衡的权重



|  |
| --- |
| **节点说明：**  在http节点里添加:  #定义负载均衡设备的 Ip及设备状态  upstream myServer {  server 127.0.0.1:9090 down;  server 127.0.0.1:8080 weight=2;  server 127.0.0.1:6060;  server 127.0.0.1:7070 backup;  }  在需要使用负载的Server节点下添加  proxy\_pass http://myServer;  upstream 每个设备的状态:  **down 表示单前的server暂时不参与负载**  weight 默认为1.weight越大，负载的权重就越大。  **max\_fails ：允许请求失败的次数默认为1.当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误**  fail\_timeout:max\_fails 次失败后，暂停的时间。  **backup： 其它所有的非backup机器down或者忙的时候，请求backup机器。所以这台机器压力会最轻。** |

# Nginx的高可用

解决高可用的方案就是添加冗余。