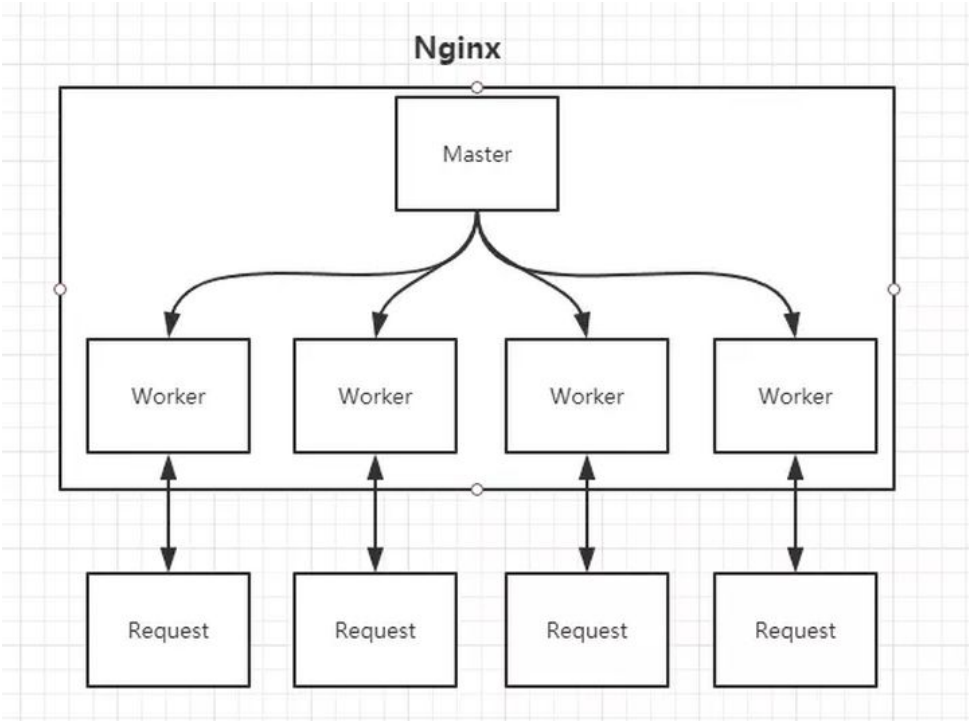
Nginx

> Nginx是一款轻量级的Web服务器、反向代理服务器，由于它的内存占用少，启动极快，高并发能力强，在互联网项目中广泛应用。
>
> 特点：反向代理 支持高并发
>
> 正向代理代理的是客户端，对服务端隐藏真正的访问者 反向代理代理的服务端，对客户端隐藏真正提供服务的目标

Nginx的Master-Worker模式



> 启动Nginx，默认使用80端口监听，启动之后涉及Master进程和Worker进程。
>
> Master进程的作用是?
>
> **读取并验证配置文件nginx.conf；管理worker进程；**
>
> Worker进程的作用是?
>
> **每一个Worker进程都维护一个线程（避免线程切换），处理连接和请求；注意Worker进程的个数由配置文件决定，一般和CPU个数相关（有利于进程切换），配置几个就有几个Worker进程。**

Nginx如何做到高并发下的高效处理?

> Nginx的worker进程个数与CPU绑定、worker进程内部包含一个线程高效回环处理请求，除此之外**Nginx采用了Linux的epoll模型，epoll模型基于事件驱动机制，它可以监控多个事件是否准备完毕，如果OK，那么放入epoll队列中，这个过程是异步的。worker只需要从epoll队列循环处理即可。**
>
> - Nginx能够提高速度的其中一个特性就是：动静分离，就是把静态资源放到Nginx上，由Nginx管理，动态请求转发给后端。
> - **我们可以在Nginx下把静态资源、日志文件归属到不同域名下（也即是目录），这样方便管理维护。**

- **Nginx可以进行IP访问控制，有些电商平台，就可以在Nginx这一层，做一下处理，内置一个黑名单模块，那么就不必等请求通过Nginx达到后端在进行拦截，而是直接在Nginx这一层就处理掉。**

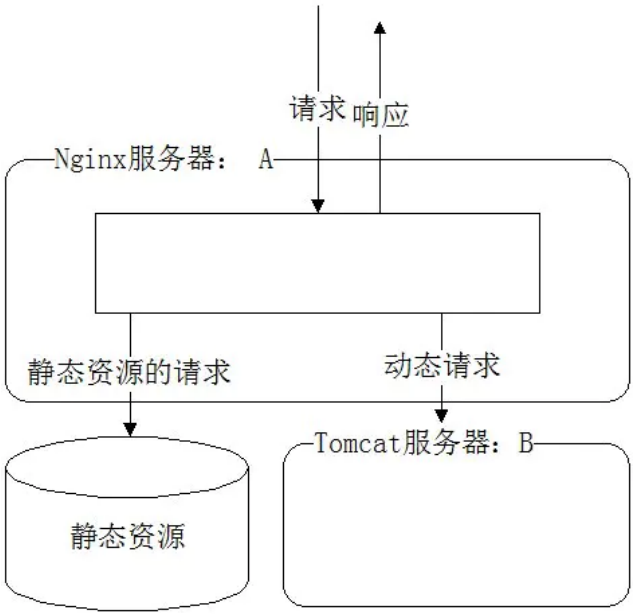
反向代理【proxy_pass】

所谓反向代理，很简单，其实就是在location这一段配置中的root替换成**proxy_pass**即可。root说明是静态资源，可以由Nginx进行返回；而proxy_pass说明是动态请求，需要进行转发，比如代理到Tomcat上。

反向代理，上面已经说了，过程是透明的，比如说request -> Nginx -> Tomcat，那么对于Tomcat而言，请求的IP地址就是Nginx的地址，而非真实的request地址，这一点需要注意。不过好在Nginx不仅仅可以反向代理请求，还可以由用户**自定义设置HTTP HEADER**。

Nginx有哪些应用?

1.动静分离



动静分离其实就是 Nginx 服务器将接收到的请求分为**动态请求**和**静态请求**。

静态请求直接从 nginx 服务器所设定的根目录路径去取对应的资源，动态请求转发给真实的后台（前面所说的应用服务器，如图中的Tomcat）去处理。

nginx动静分离的好处

api接口服务化：动静分离之后，后端应用更为服务化，只需要通过提供api接口即可，可以为多个功能模块甚至是多个平台的功能使用，可以有效的节省后端人力，更便于功能维护。

前后端开发并行：前后端只需要关心接口协议即可，各自的开发相互不干扰，并行开发，并行自测，可以有效的提高开发时间，也可以有些的减少联调时间

减轻后端服务器压力，提高静态资源访问速度：后端不用再将模板渲染为html返回给用户端，且静态服务器可以采用更为专业的技术提高静态资源的访问速度。

```nginx
server {
        listen       8080;
        server_name  localhost;

        location / {
            root   html; # Nginx默认值
            index  index.html index.htm;
        }

        # 静态化配置，所有静态请求都转发给 nginx 处理，存放目录为 my-project
        location ~ .*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|js|css)$ {
            root /usr/local/var/www/my-project; # 静态请求所代理到的根目录
        }

        # 动态请求匹配到path为'node'的就转发到8002端口处理
        location /node/ {
            proxy_pass http://localhost:8002; # 充当服务代理
        }
}
```

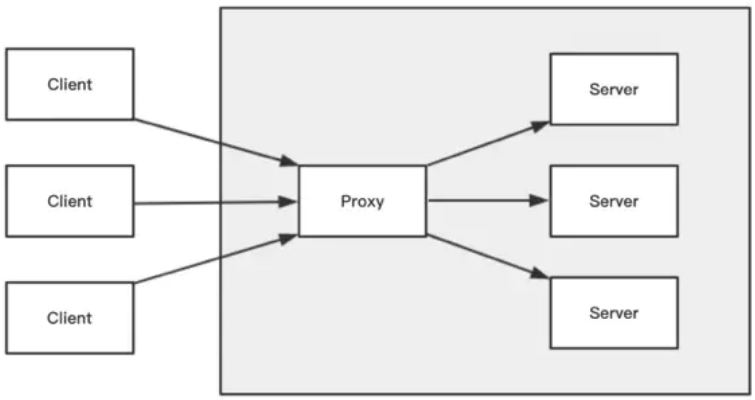
2.反向代理

1. 保障应用服务器的安全（增加一层代理，可以屏蔽危险攻击，更方便的控制权限）
2. 实现负载均衡（稍等~下面会讲）
3. 实现跨域（号称是最简单的跨域方式）

如上配置的 'proxy_pass http://localhost:8002' 就是反向代理，请求被转发至8002端口

现实中反向代理多用于负载均衡

反向代理



nginx 就是充当图中的 proxy。左边的3个 client 在请求时向 nginx 获取内容，是感受不到3台 server 存在的。此时，proxy就充当了3个 server 的反向代理。

反向代理应用十分广泛，CDN 服务就是反向代理经典的应用场景之一。除此之外，反向代理也是实现负载均衡的基础，很多大公司的架构都应用到了反向代理。

负载均衡

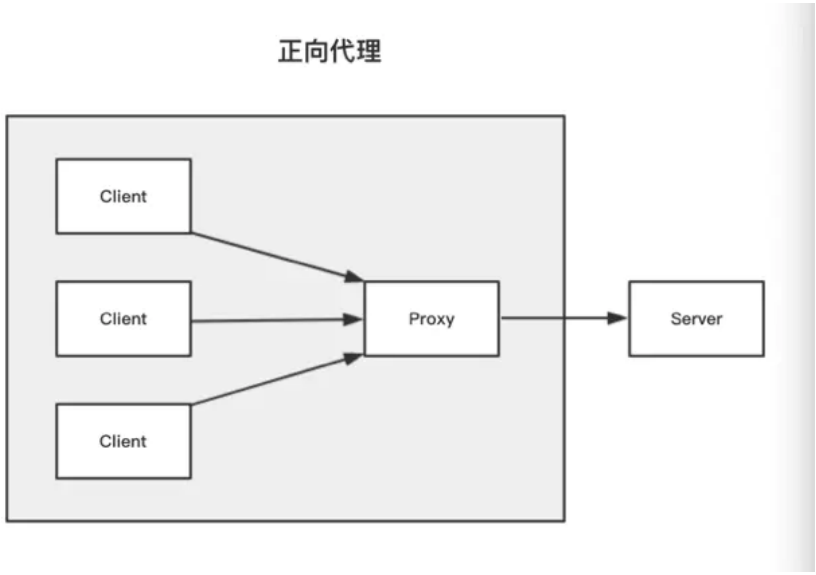
在服务器集群中，Nginx 可以将接收到的客户端请求按照某种策略分配到这个集群中的服务器上。这个就叫做**负载均衡**。

```nginx
# 负载均衡：设置domain
upstream domain {
    server localhost:8000;
    server localhost:8001;
}
server {
        listen       8080;
        server_name  localhost;

        location / {
            # root   html; # Nginx默认值
            # index  index.html index.htm;

            proxy_pass http://domain; # 负载均衡配置，请求会被平均分配到8000和8001端口
            proxy_set_header Host $host:$server_port;
        }
}
```

正向代理



nginx 就是充当图中的 proxy。左边的3个 client 在请求时向 nginx 获取内容，server 是感受不到3台 client 存在的。此时，proxy 就充当了3个 client 的正向代理。

tomcat 与 nginx，apache的区别是什么?

Apache Tomcat则是Apache基金会下的另外一个项目，与Apache HTTP Server相比，Tomcat能够**动态**的生成资源并返回到客户端。Apache HTTP Server和Nginx都能够将某一个文本文件的内容通过HTTP协议返回到客户端，但是这个文本文件的内容是固定的——也就是说无论何时、任何人访问它得到的内容都是完全相同的，这样的资源我们称之为**静态**资源。

> Apache HTTP Server和Nginx本身不支持生成动态页面，但它们可以通过其他模块来支持（例如通过Shell、PHP、Python脚本程序来动态生成内容）。
> 如果想要使用Java程序来动态生成资源内容，使用这一类HTTP服务器很难做到。Java Servlet技术以及衍生的Java Server Pages技术可以让Java程序也具有处理HTTP请求并且返回内容（由程序动态控制）的能力，Tomcat正是支持运行Servlet/JSP应用程序的容器（Container）
>
> Tomcat运行在JVM之上，它和HTTP服务器一样，绑定IP地址并监听TCP端口，同时还包含以下指责：
>
> - 管理Servlet程序的生命周期
> - 将URL映射到指定的Servlet进行处理
> - 与Servlet程序合作处理HTTP请求——根据HTTP请求生成HttpServletResponse对象并传递给Servlet进行处理，将Servlet中的HttpServletResponse对象生成的内容返回给浏览器