Please visit website: http://cxyroad.com

Nginx服务器超详细入门教程

一、Nginx介绍

Nginx是一款轻量级的Web服务器、反向代理服务器，是由俄罗斯的程序设计师Igor Sysoev所开发，使用C语言开发，由于它的内存占用少，启动速度极快，具有高并发的能力，在互联网项目中被广泛地应用。
它的功能丰富，可作为HTTP服务器，静态资源服务器，也可作为反向代理服务器，邮件服务器等。支持FastCGI、SSL、Virtual Host、URL Rewrite、Gzip等功能。并且支持很多第三方的模块扩展。Nginx在全球网站中的市场份额为33.5%，位居第二，仅次于Apache。

二、Nginx下载安装

这里不做详细描述，需要的参考文章：[惜缘梦博客](http://cxyroad.com/”http://ddosi.top/post/13”) [ddosi.top/post/13](http://cxyroad.com/”http://ddosi.top/post/13”)

三、Nginx核心功能

1、反向代理

1）什么是代理呢?

说到代理，首先我们要明确一个概念，所谓代理就是一个代表、一个渠道；此时就涉及到两个角色，一个是被代理角色，一个是目标角色，被代理角色通过这个代理访问目标角色完成一些任务的过程称为代理操作过程；如同生活中随处可见的中介，我们可以通过中介买到房子，中介就是代理，被代理的就是买房子的人，而我们就是目标角色。

2）正向代理

说到正向代理，应该很多人都接触过，也听过它的名字，那就是我们熟知的VPN：VPN通俗的讲就是一种中转服务，当我们电脑接入VPN 后，我们对外IP地址就会变成VPN服务器的公网IP，我们请求或接受任何数据都会通过这个VPN服务器后传入到我们本机，隐藏了本机的地址。那么这样做有什么好处呢？VPN利用低成本的公共网络作为企业骨干网，同时又克服了公共网络缺乏保密性的弱点，在VPN网络中，位于公共网络两端的网络在公共网络上传输信息时，其信息都是经过安全处理的，可以保证数据的完整性、真实性和私有性。

这里的 VPN 就是做正向代理的。正向代理服务器位于客户端和服务器之间，为了向服务器获取数据，客户端要向代理服务器发送一个请求，并指定目标服务器，代理服务器将目标服务器返回的数据转交给客户端。这里客户端是要进行一些正向代理的设置的。

3）反向代理

反向代理，对比正向代理来说其实就是反过来了，正向代理是隐藏用户的信息，而反向代理是隐藏真实的服务器信息。

反向代理和正向代理的区别就是：正向代理代理的是客户端，反向代理代理的是服务器。

2、负载均衡

利用反向代理可以作为内部负载均衡(load balance) 的手段.

举个例子来说, 比如我现在开发了一个 java web 的博客网站, 我把它直接部署它到 tomcat 服务器上, 让 tomcat 监听 80 端口, 直接对外服务. 一开始访问量也不大, 所以这样也是没有问题的，但是当用户增多，访问量上来的时候，一个tomcat进程处理不过来了，那现在怎么办呢？于是我想到了再起一个tomcat进程，但是这样又多了一个问题，我只有一个80端口，那么我只能起一个其他的端口，这样做的话显然是有问题的，用户在访问的时候不可能有的访问80端口，有的人在访问的时候用其他端口吧。

基于以上的问题，我们可以想到可以用Nginx的反向代理来实现两个tomcat进程的负载均衡，那么我们只需要代理两个tomcat进程，配置负载均衡策略，将入口交给Nginx来管理，那么用户在访问同一个地址的时候就可以将请求分发到不同的tomcat上以实现负载均衡。

一个简单的配置如下：

```
http {
    upstream server1{
        server 127.0.0.1:8080 weight=3;
        server 127.0.0.1:8081 weight=2;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://server1;
        }
    }
}
```

其中weight是权重，当机器的配置不一样的时候可以控制流向每个机器的流量占比。

3、限流

Nginx限流就是限制用户请求的速度，防止服务器过载的情况
限流一般有3种
正常限制访问频率（正常流量）
突发限制访问频率（突发流量）
限制并发连接数
Nginx的限流都是基于漏桶流算法，后面会说到

四、Nginx基础使用及配置文件介绍

1、Nginx基本命令

```
nginx –s stop ： 强制关闭Nginx，可能不保存相关信息，并迅速终止web服务。
pkill nginx ： 强制关闭nginx
nginx –s quit ： 平稳关闭Nginx，保存相关信息，有安排的结束web服务。
nginx –s reload ： 重新加载配置而不用重启服务。
nginx –s reopen ： 重新打开日志文件。
nginx –c filename ： 为 Nginx 指定一个配置文件，来代替缺省的。
nginx –t ： 不运行，而仅仅测试配置文件。nginx 将检查配置文件的语法的正确性，并尝试打开配置文件中所引用到的文件。
nginx –v：显示 nginx 的版本。
nginx –V：显示 nginx 的版本，编译器版本和配置参数。
```

2、Nginx配置文件结构

```
main                                # 全局配置
events {                            # nginx工作模式配置

}
http {                              # http设置
....
server {                          # 服务器主机配置
    ....
    location {                    # 路由配置
        ....
    }
    location path {
        ....
    }
    location otherpath {
        ....
    }
}
server {
    ....

    location {
        ....
    }
}
```

```
upstream name {                    # 负载均衡配置
    ....
}
}
```

如上述配置文件所示，主要由6个部分组成：

1、main：用于进行nginx全局信息的配置

2、events：用于nginx工作模式的配置

3、http：用于进行http协议信息的一些配置

4、server：用于进行服务器访问信息的配置

5、location：用于进行访问路由的配置

6、upstream：用于进行负载均衡的配置

main模块

```
# user nobody nobody;
worker_processes 2;
# error_log logs/error.log
# error_log logs/error.log notice
# error_log logs/error.log info
# pid logs/nginx.pid
```

ser用来指定nginx worker进程运行用户以及用户组，默认nobody账号运行

worker_processes指定nginx要开启的子进程数量，运行过程中监控每个进程

消耗内存(一般几M~几十M不等)根据实际情况进行调整，通常数量是CPU内核数量的整数倍

error_log定义错误日志文件的位置及输出级别debug / info / notice / warn / error / crit

pid用来指定进程id的存储文件的位置

worker_rlimit_nofile用于指定一个进程可以打开最多文件数量的描述

event 模块

```
event {
worker_connections 1024;
multi_accept on;
use epoll;
}
```

worker_connections 指定最大可以同时接收的连接数量，这里一定要注意，最大连接数量是和worker processes共同决定的。

multi_accept 配置指定nginx在收到一个新连接通知后尽可能多的接受更多的连接

use epoll 配置指定了线程轮询的方法，如果是linux2.6+，使用epoll，如果是BSD如Mac请使Kqueue

http模块

作为web服务器，http模块是nginx最核心的一个模块，配置项也是比较多的，项目中会设置到很多的实际业务场景，需要根据硬件信息进行适当的配置，常规情况下，使用默认配置即可！这里不做介绍。

server模块

srever模块配置是http模块中的一个子模块，用来定义一个虚拟访问主机，也就是一个虚拟服务器的配置信息。

```
server {
listen          80;
server_name localhost     192.168.1.100;
root          /nginx/www;
index          index.php index.html index.html;
charset          utf-8;
access_log     logs/access.log;
error_log     logs/error.log;
.....
}
```

server：一个虚拟主机的配置，一个http中可以配置多个server

server_name：用来指定ip地址或者域名，多个配置之间用空格分隔

root：表示整个server虚拟主机内的根目录，所有当前主机中web项目的根目录，可以用来做静态服务器

index：用户访问web网站时的全局首页

charset：用于设置www/路径中配置的网页的默认编码格式

access_log：用于指定该虚拟主机服务器中的访问记录日志存放路径

error_log：用于指定该虚拟主机服务器中访问错误日志的存放路径

location模块

location模块是nginx配置中出现最多的一个配置，主要用于配置路由访问信息

在路由访问信息配置中关联到反向代理、负载均衡等等各项功能。

```
location / {
root    /nginx/www;
index    index.php index.html index.htm;
    }
```

location /：表示匹配访问根目录

root：用于指定访问根目录时，访问虚拟主机的web目录

index：在不指定访问具体资源时，默认展示的资源文件列表

反向代理配置方式

通过反向代理代理服务器访问模式，通过proxy_set配置让客户端访问透明化

```
http {
    upstream server1{
        server 127.0.0.1:8080 weight=3;
        server 127.0.0.1:8081 weight=2;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://server1;
    index    index.php index.html index.htm;
        }
    }
}
```

五、Nginx具体使用

1、静态资源服务器

Nginx可以作为静态web服务器来部署静态资源。静态资源指在服务端真实存在并且能够直接展示的一些文件，比如常见的html页面、css文件、 js文件、图片、视频等资源。相对于Tomcat, Nginx处理静态资源的能力更加高效，所以在生产环境下，一般都会 将静态资源部署到Nginx中。将静态资源部署到Nginx非常简单，只需要将文件复制到Nginx安装目录下的html目录中，然后访问相应的地址即可下载对应的静态资源。

例如：

```
server {
listen 80; #监听端口
server_name localhost; #服务器名称
location / {  #匹配客户端请求url
root html; #指定静态资源根目录
index index.html; #指默认首页
}
}
```

通过以上配置文件启动nginx，那么我们只需要将静态文件例如图片，文件放到html目录下面，然后访问ip地址加80端口加对应的路径就可以访问到服务器中的文件。

2、反向代理

反向代理就是隐藏真实的服务器地址，保护真实的服务器，用户在访问时访问的是代理服务器。

nginx 中常见的反向代理指令有两个：proxy_pass 和 fastcgi_pass，前者使

用标准的 HTTP 协议转发，后者使用 FastCGI 协议转发。这里我们以 proxy_pass 为例来做介绍。

proxy_pass 有两种配置写法：
1、直接指向要代理的地址，可以是一台具体的主机（ip），也可以是一个具体的网址
2、可以搭配负载均衡指向一组服务器（参考负载均衡）

如下面的配置所示最简单的一个反向代理配置：

```
server {
listen 80;
server_name localhost;
location / {
proxy_.pass http://ddosi.top; #反向代理配置,将请求转发到指定服务
}
}
```

3、负载均衡

早期的网站流量和业务功能都比较简单，单台服务器就可以满足基本需求，但是随着互联网的发展,业务流量越来越大并且业务逻辑也越来越复杂，单台服务器的性能及单点故障问题就凸显出来了，因此需要多台服务器组成应用集群,进行性能的水平扩展以及避免单点故障出现。那么这时候Nginx就可以作为一个负载均衡服务器。

负载均衡基础配置：

```
upstream loadServer{ #upstream指令可以定义一组服务器
server 192.168.137.101:8080 weight=10; #可以设置权重，默认都是1，数值越大，分配的流量越多
server 192.168.137.101:8081 weight=5;
}
```

```
server {
listen 80;
server_name localhost;
location / {
proxy_pass http://loadServer;
}
}
```

负载均衡算法

1、轮询（默认方法）

每个请求按时间顺序逐一分配到不同的后端服务器。

```
upstream backserver {
 server http://ddosi.top/;
 server 192.168.0.100;
}
```

2、加权

weight的值越大分配，到的访问概率越高，只要用于服务器配置不一样，承受的流量不一样的情况。

```
upstream backserver {
 server http://ddosi.top/ weight=2;
 server 192.168.0.100 weight=3;
}
```

3、ip_hash

每个请求按访问IP的哈希结果分配，使来自同一个IP的访客固定访问一台后端服务器，并且可以有效解决动态网页存在的session共享问题

```
upstream backserver {
 ip_hash;
 server http://ddosi.top/ weight=2;
 server 192.168.0.100 weight=3;
}
```

4、热备

如果你有2台服务器，当一台服务器发生事故时，才启用第二台服务器给提供服务

```
upstream backserver {

 server http://ddosi.top/;
 server 192.168.0.100 backup;
}
```

5、fair

第三方插件，必须安装upstream_fair模块。

```
upstream backserver {
 fair;
 server http://ddosi.top/;
```

```
 server 192.168.0.100;
}
```

4、限流

Nginx提供了限制请求频率的模块，可以通过ngx_http_limit_req_module来实现。以下是一个简单的配置示例，该配置将请求限制在每秒1个请求，超过限制的请求将被拒绝。

```
http {
    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=1r/s;

    server {
        location / {
            limit_req zone=mylimit burst=5;
            proxy_pass http://my_upstream;
        }
    }
}
```

limit_req_zone: 指令定义了一个速率限制区域，其中$binary_remote_addr用于识别每个请求的IP地址，mylimit:10m是这个区的名字和内存大小，rate=1r/s设置了允许的平均请求速率。

limit_req: 指令应用了速率限制区域到具体的location，burst=5定义了允许的突发请求数，即当请求超过限制时，允许的额外请求数。。

这个配置将导致对于每个IP地址，Nginx只处理每秒1个请求，超过这个速率的请求将会收到503错误。

请注意，这只是一个基本示例，实际应用中可能需要根据实际需求调整限制的策略。例如，可以根据其他变量（如请求的大小、请求的方法等）来限流，或者使用更复杂的限流策略，如带权重的速率限制。

原文链接: https://juejin.cn/post/7380296247720837155