Nginx

> *Nginx* (engine x) 是一个高性能的HTTP和反向代理服务器，也是一个IMAP/POP3/SMTP服务器
>
> 反向代理（Reverse Proxy）
>
> 以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器
>
> 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。
>
> 负载均衡：
>
> 负载均衡服务器（load-balancing server）是进行负载分配的服务器。通过负载均衡服务器，将服务请求均衡分配到实际执行的服务中，从而保证整个系统的响应速度。
>
> http://nginx.org/en/docs/
>
> https://www.nginx.cn/doc/

安装

> 虚拟机环境测试：
>
> ssh localhost
>
> ------ connect to host localhost port 22 : connection refused
>
> sudo apt-get install openssh-server

Windows

> 下载包，解压缩，双击exe运行
>
> http://localhost/ 访问

```
# 启动nginx服务
start nginx # ok
# 关闭nginx服务器
# 必须在nginx.exe目录 nginx: [alert] could not open error log file: CreateFile()
# 必须在运行状态 nginx: [error] CreateFile()
# stop是快速停止nginx，可能并不保存相关信息；quit是完整有序的停止nginx，并保存相关信息。
nginx -s stop/quit
# 重新载入Nginx / 重启  必须在启动状态下
# 当配置信息修改，需要重新载入这些配置时使用此命令。
nginx -s reload
# 重新打开日志文件
nginx -s reopen

# 查看Nginx版本
```

```
nginx -v/V
# nginx 测试配置文件
nginx -t
# 指定配置文件
nginx -c filename
```

目录结构：

```
nginx
|---conf
|---contrib    发行库
|---docs       文档（假文档）
|---html       默认静态页 index  500
|---logs       日志文件  access.log  error.log
|---temp       临时文件
|---nginx.exe  执行文件
```

Linux

安装

[nginx 官网http://nginx.org/](http://nginx.org/)

documention----installing nginx ----packages

```
#查看操作系统版本信息
cat /etc/*release*

# sudo apt-key add nginx_signing.key

vi /etc/apt/sources.list
#在末尾追加
deb http://nginx.org/packages/ubuntu/ codename nginx
deb-src http://nginx.org/packages/ubuntu/ codename nginx
sudo apt-get update
sudo apt-get install nginx
nginx -V 看安装路径
# 测试
# 浏览器地址栏输入ip
# 配置文件  nginx -t
```

```
1.Nginx日志轮转，用于logrotate服务的日志切割 | 配置文件
/etc/logrotate.d/nginx
1
2.Nginx住配置文件 | 目录、配置文件
/etc/nginx
/etc/nginx/nginx.conf
```

```
/etc/nginx/conf.d
/etc/nginx/conf.d/default.conf
1
2
3
4
3.cgi配置相关，fastcgi | 配置文件
/etc/nginx/fastcgi_params
/etc/nginx/scgi_params
/etc/nginx/uwsgi_params
1
2
3
4.编码转换映射转化文件 | 配置文件
/etc/nginx/koi-utf
/etc/nginx/koi-win
/etc/nginx/win-utf
1
2
3
5.设置http协议的Content-Type与扩展名对应关系：返回数据的类型 | 配置文件
/etc/nginx/mime.types
1
6.用于配置出系统守护进程管理器管理方式 ： centos7.2 | 配置文件
/usr/lib/systemd/system/nginx-dubug.service
/usr/lib/systemd/system/nginx.service
/etc/sysconfig/nginx
/etc/sysconfig/nginx-dubug
1
2
3
4
7.Nginx模块目录 | 目录
/usr/lib64/nginx/modules
/etc/nginx/modules
1
2
8.Nginx服务的启动管理的终端命令 | 命令
/usr/sbin/nginx
/usr/sbin/nginx-debug
1
2
9.Nginx的手册和帮助文件 | 文件、目录
/usr/share/doc/nginx-1.10.2
/usr/share/doc/nginx-1.10.2/COPYRIGHT
/usr/share/man/man8/nginx.8.gz
1
2
3
10.Nginx的缓存目录 | 目录
/var/cache/nginx
1

11.Nginx的日志目录 | 目录
```

```
/var/log/nginx
```

```
# 启动
# --> start nginx ?
nginx # 不可重复启动
# 停止 不可重复停止
# nginx 启动时, 会创建一个nginx.pid的文件  保存了nginx 的进程信息
# 停止nginx 时, 会删除pid文件
# stop 强行停止/快速停止/
# quit 平滑的停止
nginx -s stop/quit
# 重新加载nginx 配置
nginx -s reload
# 测试配置文件是否有错误
nginx -t
#--------------------------------------
#重新打开日志文件
nginx -s reopen
#查看nginx 版本信息
nginx -v/V

nginx -c path #指定配置文件启动

杀进程-----------------------------------
1、ps 命令用于查看当前正在运行的进程。
grep 是搜索
例如： ps -ef | grep java
表示查看所有进程里 CMD 是 java 的进程信息
2、ps -aux | grep java
-aux 显示所有状态
ps
3. kill 命令用于终止进程
例如： kill -9 [PID]
-9 表示强迫进程立即停止
通常用 ps 查看进程 PID ，用 kill 命令终止进程

----------------------

本文来自 yfgcq 的CSDN 博客 ，全文地址请点击：
https://blog.csdn.net/yfgcq/article/details/51733118?utm_source=copy
```

配置文件

```
# Nginx用户及组：用户 组。window下不指定
#user  nobody;
#启动进程,通常设置成和cpu的数量相等
worker_processes  1;

#全局错误日志
```

```
#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#PID文件
#pid        logs/nginx.pid;

#工作模式及连接数上限
events {
    #epoll是多路复用IO(I/O Multiplexing)中的一种方式,
    #仅用于linux2.6以上内核,可以大大提高nginx的性能
    use   epoll;
    #单个后台worker process进程的最大并发链接数
    worker_connections  1024;

  # 并发总数是 worker_processes 和 worker_connections 的乘积
    # 即 max_clients = worker_processes * worker_connections
    # 在设置了反向代理的情况下，max_clients = worker_processes * worker_connections / 4  为什么
    # 为什么上面反向代理要除以4，应该说是一个经验值
    # 根据以上条件，正常情况下的Nginx Server可以应付的最大连接数为：4 * 8000 = 32000
    # worker_connections 值的设置跟物理内存大小有关
    # 因为并发受IO约束，max_clients的值须小于系统可以打开的最大文件数
    # 而系统可以打开的最大文件数和内存大小成正比，一般1GB内存的机器上可以打开的文件数大约是10万左右
    # 我们来看看360M内存的VPS可以打开的文件句柄数是多少：
    # $ cat /proc/sys/fs/file-max
    # 输出 34336
    # 32000 < 34336，即并发连接总数小于系统可以打开的文件句柄总数，这样就在操作系统可以承受的范围之内
    # 所以，worker_connections 的值需根据 worker_processes 进程数目和系统可以打开的最大文件总数进行适当地进行设置
    # 使得并发总数小于操作系统可以打开的最大文件数目
    # 其实质也就是根据主机的物理CPU和内存进行配置
    # 当然，理论上的并发总数可能会和实际有所偏差，因为主机还有其他的工作进程需要消耗系统资源。
}


http {
    #设定mime类型,类型由mime.type文件定义
    include       mime.types;
    default_type  application/octet-stream;

    #设定日志格式
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;

    #sendfile 指令指定 nginx 是否调用 sendfile 函数（zero copy 方式）来输出文件，
    #对于普通应用，必须设为 on,
    #如果用来进行下载等应用磁盘IO重负载应用，可设置为 off，
    #以平衡磁盘与网络I/O处理速度，降低系统的uptime.

    sendfile        on;
```

```
    #tcp_nopush     on;

    #连接超时时间
    #keepalive_timeout  0;
    keepalive_timeout  65;

    #开启gzip压缩
    #gzip  on;

    #设定虚拟主机配置
    server {
        #侦听80端口，此端口为nginx端口。。。。
        listen       80;
        ##定义使用 localhost/ip/127.0.0.1 访问
        # 此处可以使用域名，但是域名必须与IP做过映射
        server_name  localhost;
        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            index  index.html index.htm;
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }

        # proxy the PHP scripts to Apache listening on 127.0.0.1:80
        #
        #location ~ \.php$ {
        #    proxy_pass   http://127.0.0.1;
        #}

        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
        #
        #location ~ \.php$ {
        #    root           html;
        #    fastcgi_pass   127.0.0.1:9000;
        #    fastcgi_index  index.php;
        #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
        #    include        fastcgi_params;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one

        #
```

```
        #location ~ /\.ht {
        #    deny  all;
        #}
    }

    # 可以配置多个server
    # tomcat



    # another virtual host using mix of IP-, name-, and port-based configuration
    #
    #server {
    #    listen       8000;
    #    listen       somename:8080;
    #    server_name  somename  alias  another.alias;

    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #}


    # HTTPS server
    #
    #server {
    #    listen       443 ssl;
    #    server_name  localhost;

    #    ssl_certificate      cert.pem;
    #    ssl_certificate_key  cert.key;

    #    ssl_session_cache    shared:SSL:1m;
    #    ssl_session_timeout  5m;

    #    ssl_ciphers  HIGH:!aNULL:!MD5;
    #    ssl_prefer_server_ciphers  on;

    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #}

}
```

JDK & TOMCAT

```
启用root
sudo passwd root
设置密码
su root
```

```
# 下载jdk 安装包
sudo wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" http://download.oracle.com/otn-pub/java/jdk/8u161-b12/2f38c3b165be4555a1fa6e98c45e0808/jdk-8u161-linux-i586.tar.gz
# 配置环境变量
vi /ext/profile
export JAVA_HOME=/usr/local/jdk1.8.0_171
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$JAVA_HOME/bin:$PATH

出现如下信息：
  程序 ‘java’ 已包含在下列软件包中：
    * default-jre
    * gcj-5-jre-headless
    * openjdk-8-jre-headless
    * gcj-4.8-jre-headless
    * gcj-4.9-jre-headless
    * openjdk-9-jre-headless
  请尝试：sudo apt install <选定的软件包>

  解决办法：需要执行如下2条命令进行修复（路径修改成自己的路径）
  sudo update-alternatives –install /usr/bin/java java /usr/lib/jdk/jdk1.8.0_151/bin/java 300
    sudo update-alternatives –install /usr/bin/javac javac /usr/lib/jdk/jdk1.8.0_151/bin/javac 300

  java -version
```

tomcat

```
cp -r递归复制文件夹复制
mkdir -p 递归创建文件夹
cp test.war到tomcat8中修改server.xml端口号
请求路径/test/first

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
的8009改为其它。

<Server port="8005" shutdown="SHUTDOWN">改为其它。
```

编写测试代码：

```
mkdir test
cd test
vi Hello.java
```

```
import java.io.IOException;
import java.util.Properties;

import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet(urlPatterns="/test")
public class Test extends HttpServlet{

    protected void service(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        Properties p = new Properties();
        p.load(Test.class.getClassLoader().getResourceAsStream("p.properties"));
        System.out.println(p.getProperty("key"));
        req.getRequestDispatcher("/index.jsp").forward(req, resp);
    }

}
```

```
# 使用tomcat访问项目
```

反向代理，代理tomcat

tomcat 启动速度慢

```
百度搜索
bea weblogic sip
https://docs.oracle.com/cd/E13209_01/wlcp/wlss30/index.html
securerandom.source=file:/dev/./urandom
```

1.怎么搭建文件服务器?

http服务器

```
# 1.nginx默认启动的时候使用的为conf/nginx.conf
```

```
http{
  server {
      listen       80;       # 监听的端口
      server_name  localhost;   # 服务器的Ip

      location / {
          root   html;          # 访问的根目录
          index  index.html index.htm;  # 默认访问页面
      }
      # 浏览器发送localhost/www ==> /app/www/index.html
      # 浏览器发送localhost/www/aaa.html ==> /app/www/aaa.html
      # 浏览器发送localhost/url(必须以www开头) ==> /app/url
      location /www {
        root /app;
        index index.html
      }
  }
}
```

```
<html>

<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

<p>请仅仅把标题标签用于标题文本。不要仅仅为了产生粗体文本而使用它们。请使用其它标签或 CSS 代替。</p>

</body>
</html>
```

2.反向代理

使用nginx代理Tomcat（使用nginx代理其他服务器）

```
http{

   #  后台服务器列表
   upstream tomcat-server{
       server 127.0.0.1:8080;
   }
```

```
    server {
        listen       80;       # 监听的端口
        server_name  localhost;   # 服务器的Ip

        location / {
            proxy_pass http://tomcat-server;
            proxy_pass http://localhost:8080;
        }


        location /test {
            proxy_pass http://tomcat_server;
            proxy_pass http://localhost:8080/test;
        }
    }
}
```

3.虚拟主机

在一台主机上搭建多个网站，使用80端口

```
http{

    #  后台服务器列表
    upstream sina_server{
        server 127.0.0.1:8080;
    }
    upstream sohu_server{
        server 127.0.0.1:8081;
    }

    server {
        listen       80;       # 监听的端口
        server_name  www.sina.com;   # 服务器的Ip

        location / {
            proxy_pass http://sina_server;
        }
    }

     server {
        listen       80;       # 监听的端口
        server_name  www.sohu.com;   # 服务器的Ip

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

4.负载均衡

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
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。
3、ip_hash
每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。
4、fair（第三方）
按后端服务器的响应时间来分配请求，响应时间短的优先分配。
5、url_hash（第三方）
按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，后端服务器为缓存时比较有效。

每个样式的含义如下：
$server_name：虚拟主机名称。
$remote_addr：远程客户端的IP地址。
-：空白，用一个“-”占位符替代，历史原因导致还存在。
$remote_user：远程客户端用户名称，用于记录浏览者进行身份验证时提供的名字，如登录百度的用户名scq2099yt，如果没有登录就是空白。
[$time_local]：访问的时间与时区，比如18/Jul/2012:17:00:01 +0800，时间信息最后的"+0800"表示服务器所处时区位于UTC之后的8小时。
$request：请求的URI和HTTP协议，这是整个PV日志记录中最有用的信息，记录服务器收到一个什么样的请求
$status：记录请求返回的http状态码，比如成功是200。
$uptream_status：upstream状态，比如成功是200.

$upstream_addr:后端服务器的IP地址

```
user www-data;
worker_processes 4;
pid /run/nginx.pid;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

   log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                     '$upstream_addr ';
     access_log /var/log/nginx/access.log main;

    upstream www.test.com{
        server 127.0.0.1:8080;
        server 127.0.0.1:8081;
   }

   server{
      listen 80;
      server_name www.test.com;
```

```
       location / {
          proxy_pass http://www.test.com;
       }
    }


}
```

5.动静分离

```
     server {
        listen       80;       # 监听的端口
        server_name  www.sohu.com;   # 服务器的Ip


        # 对非静态资源的处理
        location / {
            proxy_pass http://sohu_server;
        }

        # 对静态资源处理 app-->www-->index.html
        location /www{
            root /app;
            index index.html
        }
     }
```