黑客派

linux 下 nginx 安装与常用配置

作者：ldk

原文链接：https://hacpai.com/article/1515725585588

来源网站：黑客派

<h2 id="一-linux下nginx的安装">一、Linux 下 nginx 的安装</h2>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
(adsbygoogle = window.adsbygoogle || []).push({});
</script>
<h3 id="1-1-下载nginx及pcre安装包">1.1 下载 nginx 及 pcre 安装包</h3>
nginx 下载地址：<a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fnginx.
rg%2Fdownload%2Fnginx-1.13.8.tar.gz" target="_blank" rel="nofollow ugc">http://nginx.org
download/nginx-1.13.8.tar.gz
 pcre 下载地址：<a href="https://link.hacpai.com/for
ard?goto=https%3A%2F%2Fsourceforge.net%2Fprojects%2Fpcre%2Ffiles%2Fpcre%2F8.41%2
pcre-8.41.tar.gz%2Fdownload" target="_blank" rel="nofollow ugc">https://sourceforge.net/p
ojects/pcre/files/pcre/8.41/pcre-8.41.tar.gz/download
<h3 id="1-2-pcre安装">1.2 pcre 安装</h3>
新建安装目录并上传文件
计划将 nginx 安装在/usr/pcre 目录下，新建目录/usr/pcre
<pre><code class="highlight-chroma">cd /usr
mkdir pcre
cd pcre
</code></pre>
将下载下来的 Nginx 使用 fileZilla 上传到/usr/pcre 目录（即当前目录）下。
上传 pcre 安装包 pcre-8.41.tar.gz 到/usr/pcre 目录下
解压 pcre 安装包
<pre><code class="highlight-chroma">tar -zxvf pcre-8.41.tar.gz
</code></pre>
得到 pcre-8.41 文件夹
对当前文件夹授予全部读写权限：
<pre><code class="highlight-chroma">chmod -R 777 pcre-8.41
</code></pre>
初始化配置
切换到/usr/pcre/pcre-8.41 目录下，运行以下命令进行 pcre 初始化配置。
<pre><code class="highlight-chroma">./configure
</code></pre>
进行编译
<pre><code class="highlight-chroma">make install
</code></pre>
进行安装，至此 PCRE 安装完成。
<h3 id="1-3-nginx安装">1.3 nginx 安装</h3>
新建安装目录并上传文件
计划将 nginx 安装在/usr/nginx 目录下，新建目录/usr/nginx
<pre><code class="highlight-chroma">cd /usr
mkdir nginx
cd nginx
</code></pre>
将下载下来的 Nginx 使用 fileZilla 上传到/usr/nginx 目录（即当前目录）下。
解压 nginx 压缩包
<pre><code class="highlight-chroma">tar -zxvf nginx-1.12.2.tar.gz
</code></pre>
得到 nginx-1.12.2 文件夹
初始化配置

切换到/usr/nginx/nginx-1.12.2 目录下，运行 <code>./configure</code> 进行初始化配置。
/p>
<pre><code class="highlight-chroma">cd /usr/nginx/nginx-1.12.2
./configure
</code></pre>
如果显示缺少别的支持库。首先安装系统常用支持库。减少安装过程中很多错误的出现。
<code>yum install -y automake make gcc gdb strace gcc-c++ autoconf libjpeg libjpeg-d
vel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glib
-devel glib2 glib2-devel bzip2 bzip2-devel ncurses ncurses-devel curl curl-devel e2fsprogs pa
ch e2fsprogs-devel krb5-devel libidn libidn-devel openldap-devel nss_ldap openldap-clients
penldap-servers libevent-devel libevent uuid-devel uuid mysql-devel</code>

安装完成后重复运行 <code>./configure</code> 命令进行初始化。
如果显示没有安装 pcre，请先按 2.4.2 步骤安装 pcre。
安装完 pcre 之后，重复运行 <code>./configure</code> 命令，若提示。/configure: error: t
e HTTP gzip module requires the zlib library.
则需要安装“zlib-devel”即可。执行以下命令：
<pre><code class="highlight-chroma">yum install -y zlib-devel
</code></pre>
安装完成，再次运行 <code>./configure</code> 进行初始化即可。注意这里生成的配置文件
尤其箭头所指的方向，是启动 nginx 时的路径。
进行编译
运行以下命令进行编译：
<pre><code class="highlight-chroma">make install
</code></pre>
测试启动、停止、重启

 启动 <pre><code class="highlight-chroma"> /usr/local/nginx/sbin/nginx -c /u
r/nginx/nginx-1.12.2/conf/nginx.conf
</code></pre> 或 <pre><code class="highlight-chroma"> /usr/local/nginx/sbin/
ginx
</code></pre>
 停止 # 查询 nginx 主进程号 <pre><code class="highlight-chroma">
s -ef | grep nginx
</code></pre> # 停止进程 <pre><code class="highlight-chroma">kill -QUIT 主
程号
</code></pre> # 快速停止 <pre><code class="highlight-chroma">kill -TERM 主
程号
</code></pre> # 强制停止 <pre><code class="highlight-chroma">pkill -9 nginx
</code></pre>
 重启(首次启动需：<code>/usr/local/nginx/sbin/nginx -c /usr/nginx/nginx-1.12.2/con
/nginx.conf</code>) <pre><code class="highlight-chroma">/usr/local/nginx/sbin/ngi
x -s reload
</code></pre>
 测试 # 测试端口 netstat -na | grep 80 # 浏览器中测试</
> <a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fip%3A80" target="_b

ank" rel="nofollow ugc">http://ip:80 出现以下画面说明启动正常。 <i
g src="https://static.hacpai.com/images/img-loading.svg" alt="7f5dad7ed4ad4a32be0f6c4e0
c35612-image.png" data-src="https://ozeauwce0.bkt.clouddn.com//file/2018/1/7f5dad7ed4a
4a32be0f6c4e04c35612-image.png">

<h2 id="二-nginx配置">二、nginx 配置</h2>
为了避免其他人把未备案的域名解析到自己的服务器 IP，而导致服务器被断网，配置 nginx 将
求分发到内网，同时配置静态文件分离部署。需要对 nginx 进行相关配置。
为保证 nginx 可以使用 80 端口不被占用，外界必须通过 nginx 访问应用，可进行以下配置：<b
> 推荐方式一：将 Tomcat 部署在本机的非 80 端口下（假设为 8081），且防火墙配置为不开启应
服务器对应端口（如不开启 8081 端口）。
 推荐方式二：将 Tomcat 部署在内网其他机器上，n
inx 通过内网 ip 连接应用服务器。
 两种方式都可以支持一个 nginx 配置多个应用的代理。

nginx.conf 配置
在 nginx-1.12.2/conf 目录下，编辑 nginx.conf 文件
<pre><code class="highlight-chroma">cd /usr/nginx/nginx-1.12.2/conf
vi nginx.conf
</code></pre>
修改为如下配置：
<hr>
<pre><code class="highlight-chroma">#user nobody;
worker_processes 1; #工作进程的个数，可以配置多个，建议设置为等于CPU总核心数
#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;
#pid logs/nginx.pid;

events {

worker_connections 1024; #单个进程的最大连接数（该服务器的最大连接数=连接数*进程数）

}

http {

include mime.types;

default_type application/octet-stream;

#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 on;

#tcp_nopush on;

#keepalive_timeout 0;

keepalive_timeout 65;

```
#开启gzip压缩
#gzip on;

client_max_body_size 50m; #缓冲区代理缓冲用户端请求的最大字节数,可以理解为保存到本地再传用户
client_body_buffer_size 256k;
client_header_timeout 3m;
client_body_timeout 3m;
send_timeout 3m;
proxy_connect_timeout 300s; #nginx跟后端服务器连接超时时间(代理连接超时)
proxy_read_timeout 300s; #连接成功后，后端服务器响应时间(代理接收超时)
proxy_send_timeout 300s;
proxy_buffer_size 64k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小
proxy_buffers 4 32k; #proxy_buffers缓冲区，网页平均在32k以下的话，这样设置
proxy_busy_buffers_size 64k; #高负荷下缓冲大小（proxy_buffers*2）
proxy_temp_file_write_size 64k; #设定缓存文件夹大小，大于这个值，将从upstream服务器传递请，而不缓冲到磁盘
proxy_ignore_client_abort on; #不允许代理端主动关闭连接

#配置只能指定域名访问，ip不能访问，同时配置动静分离，静态文件缓存等。
include hosts/host1.conf;
}
</code></pre>

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>

<!-- 黑客派PC帖子内嵌-展示 -->

<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<hr>
<p>修改完文件后，按 <code>Esc</code> 键退出文本编辑状态，输入 <code>:wq</code> 确
编辑文本。</p>
<p><strong>新建 host1.conf 文件</strong></p>
<p>在当前目录下新建文件夹 hosts，并在 hosts 文件夹下新建文件 host1.conf</p>
<pre><code class="highlight-chroma">mkdir hosts
cd hosts
touch host1.conf
</code></pre>
<p><strong>配置 host1.conf</strong></p>
<pre><code class="highlight-chroma">vi host1.conf
```

</code></pre>
填写为以下内容，并将下文中对应部分替换为自己的信息：
<hr>
<pre><code class="highlight-chroma">server {
  listen 80; #表示当前的代理服务器监听的端口，默认的是监听80端口。
  server_name _; #没有匹配到的其他servername
  return 403; #过滤其他域名的请求，返回403状态码
}

#对应域名为www.aaa.com的服务

server {

listen 80; #表示当前的代理服务器监听的端口，默认的是监听80端口。

server_name www.aaa.com; #此处配置为允许访问的域名

#实际上我们的需求不会是直接匹配所有路径，我们需要分文件类型来进行过滤，

#html,js,css这些不需要处理的，直接给nginx进行缓存。

#进行一下配置，让JSP页面直接给tomcat，而html,png等一些图片和JS等直接给nginx进行缓存。

#配置静态资源路径。若不做动静分离，此项location配置可删除。

location ~ .(html|js|css|png|gif|jpg)$ {

root /usr/myProject/aaaa; #此处配置为静态资源所在路径

}

#location: 表示匹配的路径，这时配置了/表示所有请求都被匹配到这里

location / {

#root html; #里面配置了root这时表示当匹配这个请求的路径时，将会在这个文件夹内寻找相应的文，这里对我们之后的静态文件伺服很有用。

#index index.html index.htm; #当没有指定主页时，默认会选择这个指定的文件，它可以有多个，按顺序来加载，如果第一个不存在，则找第二个，依此类推。

proxy_pass http://127.0.0.1:8081; #proxy_pass，它表示代理路径，相当于转发，此处配置为内网应用服务器所在ip及端口。

}

}
</code></pre>

<hr>
修改完文件后，按 Esc 键退出文本编辑状态，输入：wq 确定编辑文本。
配置完成重启 nginx 即可使配置生效。