第1章 Nginx介绍

1.Nginx是什么

```
Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器。
Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的.
第一个公开版本0.1.0发布于2004年10月4日。

Nginx主要特点有
开源: 直接获取源代码
高性能: 支持海量并发
可靠: 服务稳定
```

2.为什么选择 Nginx 服务

```
互联网公司大都选择 Nginx
1.Nginx 技术成熟，具备的功能是企业最常使用而且最需要的
2.适合当前主流架构趋势，微服务、云架构、中间层
3.统一技术栈，降低维护成本，降低技术更新成本。
```

3.Nginx重要特性

```
1.开源,可以从官网直接获取源代码
2.高性能,Nginx性能非常残暴,支持海量并发
3.高可靠,服务稳定,占用内存底
4.模块化,Nginx具有丰富的模块可以按需使用，并且有开发能力的技术人员还可以二次开发
5.支持热更新配置文件，一般情况下修改配置文件可以平滑生效，不用重新启动服务
```

4.Nginx应用场景

```
提供静态网页服务
作为多个网站和域名的虚拟主机服务
反向代理负载均衡服务
提供简单的下载服务
```

第2章 Nginx架构

Nginx是多进程架构,当我们启动时会使用root用户创建一个Master进程，然后再由Master进程创建出多个Worker进程。

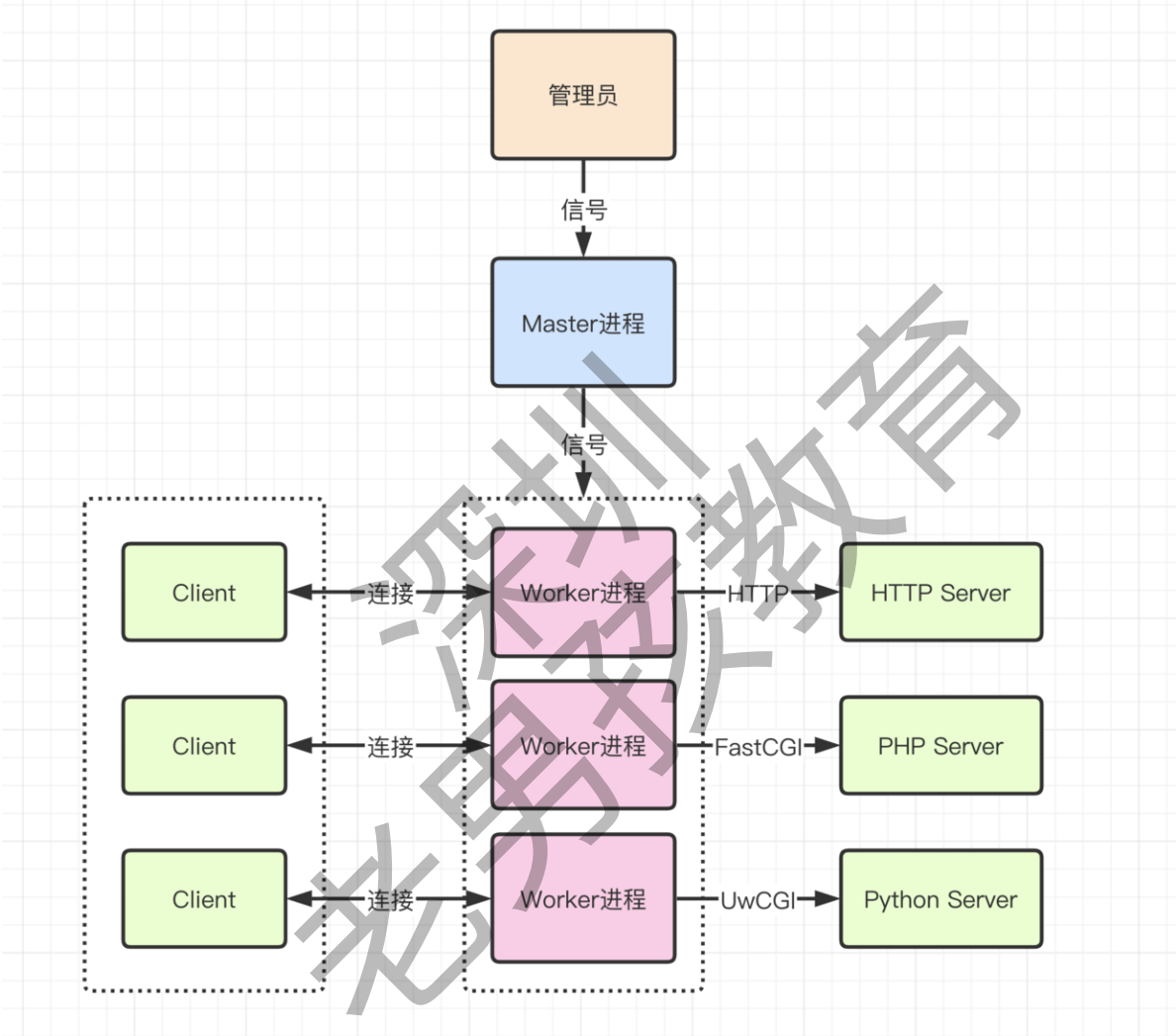
1.master 主进程功能

```
1.启动时读取并检查Nginx配置文件是否有语法或拼写错误
2.根据配置文件里的参数创建和监控worker进程状态
3.监听本地的socekt，接收用户发起的请求，然后worker进程竞争连接，获胜的处理并响应用户请求
4.接收管理员发送的管理Nginx操作信号并将接收的管理信号发送给worker进程
5.如果管理员发送了平滑重启的命令，则会读取配置文件并创建新的worker进程，然后结束旧的worker进程
```

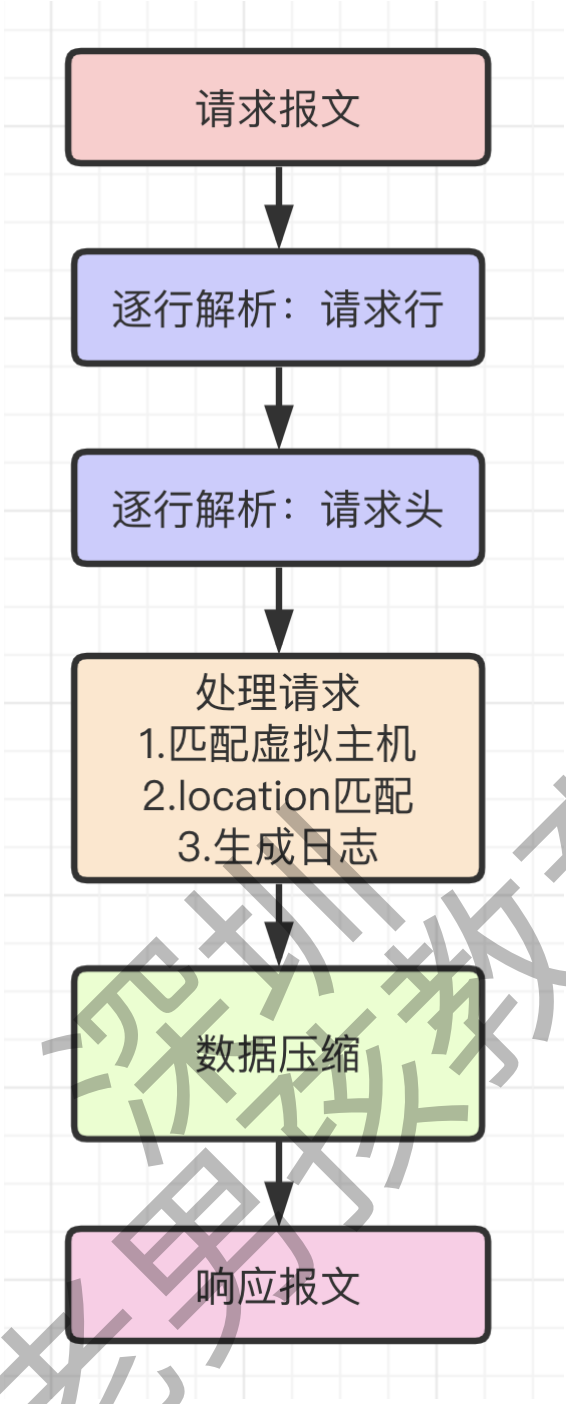
2.worker 工作进程功能

```
1.实际处理网络请求的进程是worker进程
2.master进程根据配置文件的参数决定创建多少个worker进程
3.当有用户请求的事件产生时，worker进程会向master进程竞争，获胜的工作进程和用户建立连接，并处理用户的请求
4.接收用户请求后，与后端服务器进行通信，后端处理完后接收处理结果
5.接收并处理master进程发送的信号，例如启动/重启/结束等信号
```

3.Nginx进程间架构图



4.Nginx处理HTTP请求



4.Nginx模块介绍

Nginx一个非常重要的特性就是拥有丰富的模块，有核心的模块，拓展的模块和第三方拓展模块。

Nginx模块主要可以分为以下几类：

```
核心模块：
1.HTTP 模块：用来发布http web服务网站的模块。
2.event模块：用来处理nginx 访问请求，并进行回复。

基本模块：
HTTP Access模块： 用来进行虚拟主机发布访问模块，起到记录访问日志。
HTTP FastCGI模块：用于和PHP程序进行交互的模块，负责将来访问nginx 的PHP请求转发到后端的PHP上。
HTTP Proxy模块：配置反向代理转发的模块，负责向后端传递参数。
HTTP Rewrite模块：支持Rewrite 规则重写，支持域名跳转。
```

第3章 Nginx安装部署

Nginx分为几种

```
1.源码编译(1.版本随意 2.安装复杂 3.升级繁琐)
2.epel仓库(1.版本较低 2.安装简单 3.配置不易读)
3.官方仓库(1.版本较新 2.安装简单 3.配置易读，推荐)
下面分别介绍编译安装和yum安装方法
```

1.编译安装方法

官方文档

```
http://nginx.org/en/docs/configure.html
```

创建www用户

```
groupadd www -g 666
useradd www -s /sbin/nologin -M -u 666 -g 666
id www
```

安装依赖包

```
yum install openssl-devel pcre-devel -y
```

下载解压软件包

```
mkdir /data/soft -p
cd /data/soft/
wget http://nginx.org/download/nginx-1.19.0.tar.gz
tar zxvf nginx-1.19.0.tar.gz
```

配置编译参数

```
cd /data/soft/nginx-1.19.0/
./configure --help
./configure --user=www --group=www --prefix=/opt/nginx-1.19.0 --with-
http_stub_status_module --with-http_ssl_module --with-pcre
```

编译安装

```
cd cd /data/soft/nginx-1.19.0/
make && make install
```

创建软链接

```
ln -s /opt/nginx-1.19.0/ /opt/nginx
ls -lh /opt/
```

检查语法

```
[root@web-7 /opt/nginx]# /opt/nginx/sbin/nginx -t
nginx: the configuration file /opt/nginx-1.19.0/conf/nginx.conf syntax is ok
nginx: configuration file /opt/nginx-1.19.0/conf/nginx.conf test is successful
```

启动nginx

```
/opt/nginx/sbin/nginx
```

检查测试

```
[root@web-7 /opt/nginx]# netstat -lntup|grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
12828/nginx: master
[root@web-7 /opt/nginx]# curl 10.0.0.7
```

2.YUM安装方法

安装依赖包

```
yum install openssl-devel pcre-devel -y
```

配置官方yum源

```
cat > /etc/yum.repos.d/nginx.repo << 'EOF'
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key

[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
EOF
```

安装nginx服务

```
yum install nginx -y
```

启动服务并配置开机自启动

```
[root@web-7 ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@web-7 ~]# systemctl start nginx
[root@web-7 ~]# systemctl enable nginx
```

测试访问

```
curl 10.0.0.7
```

Nginx启动方式说明

编译安装启动管理方式

```
nginx -t
nginx
nginx -s reload
nginx -s stop
```

yum安装启动管理方法

```
nginx -t
systemctl start nginx
systemctl reload nginx
systemctl restart nginx
systemctl stop  nginx
```

第4章 Nginx重要配置文件说明

1.查看重要文件

```
[root@web-7 ~]# rpm -ql nginx
.................................................
/etc/logrotate.d/nginx                    #nginx日志切割的配置文件
/etc/nginx/nginx.conf                     #nginx主配置文件
/etc/nginx/conf.d                         #子配置文件
/etc/nginx/conf.d/default.conf            #默认展示的页面一样
/etc/nginx/mime.types                     #媒体类型 （http协议中的文件类型）
/etc/sysconfig/nginx                      #systemctl 管理 nginx的使用的文件
/usr/lib/systemd/system/nginx.service     #systemctl 管理nginx（开 关 重启
reload)配置文件
/usr/sbin/nginx                           #nginx命令
/usr/share/nginx/html                     #站点目录 网站的根目录
/var/log/nginx                            #nginx日志 access.log 访问日志
.................................................
```

2.查看已经编译的模块

```
[root@web-7 ~]# nginx -V
```

3.配置文件注解

```
Nginx 主配置文件/etc/nginx/nginx.conf 是一个纯文本类型的文件，整个配置文件是以区块的形式组织的。
每个区块以一对大括号{}来表示开始与结束。
Nginx 主配置文件整体分为三块进行学习

CoreModule(核心模块)
EventModule(事件驱动模块)
HttpCoreModule(http 内核模块)
```

第一部分：配置文件主区域配置

```
user  nginx;                                #定义运行nginx进程的用户
worker_processes  auto;                     #Nginx运行的work进程数量(建议与CPU数量一致或auto)

error_log  /var/log/nginx/error.log warn;           #nginx错误日志
pid        /var/run/nginx.pid;                      #nginx运行pid
```

第二部分：配置文件事件区域

```
events {
    worker_connections  1024;  #每个 worker 进程支持的最大连接数
}
```

第三部分：配置http区域

```
http {
    include       /etc/nginx/mime.types;          #Nginx支持的媒体类型库文件
    default_type  application/octet-stream;       #默认的媒体类型

    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  /var/log/nginx/access.log  main;    #访问日志保存路径

    sendfile        on;                             #开启高效传输模式
    #tcp_nopush     on;                                 #必须配合tcp_nopush使用，当数据包累计到一定大小后就发送
    keepalive_timeout  65;                          #连接超时时间，单位是秒
    #gzip  on;                                      #开启文件压缩
    include /etc/nginx/conf.d/*.conf;               #包含子配置文件
}
```

第四部分：子配置文件内容

```
server {
    listen       80;              #指定监听端口
    server_name  localhost;       #指定监听的域名
    location / {
        root   /usr/share/nginx/html;      #定义站点的目录
        index  index.html index.htm;       #定义首页文件
    }
}
```

http server location 扩展了解项

```
http{}层下允许有多个 Server{}层，一个 Server{}层下又允许有多个 Location
http{} 标签主要用来解决用户的请求与响应。
server{} 标签主要用来响应具体的某一个网站。
location{} 标签主要用于匹配网站具体 URL 路径
```

第5章 Nginx虚拟主机配置实战

1.基于域名的虚拟主机

```
[root@web-7 ~]# cat /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
```

```
    server   {
        listen       80;
        server_name  www.oldboy.com;
        location / {
            root   /usr/share/nginx/html/www;
            index  index.html index.htm;
        }
    }
    server   {
        listen       80;
        server_name  blog.oldboy.com;
        location / {
            root   /usr/share/nginx/html/blog;
            index  index.html index.htm;
        }
    }
}
```

2.基于端口的虚拟主机

```
[root@web-7 ~]# cat /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
    server   {
        listen       81;
        server_name  www.oldboy.com;
        location / {
            root   /usr/share/nginx/html/www;
```

```
            index  index.html index.htm;
        }
    }
    server   {
        listen       82;
        server_name  blog.oldboy.com;
        location / {
            root   /usr/share/nginx/html/blog;
            index  index.html index.htm;
        }
    }
}
```

3.基于IP的虚拟主机

添加第二IP

```
ip addr add 10.0.0.11/24 dev eth0
```

配置文件

```
[root@web-7 ~]# cat /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    #include /etc/nginx/conf.d/*.conf;
    server   {
        listen       10.0.0.7:80;
        server_name  www.oldboy.com;
```

```
        location / {
            root   /usr/share/nginx/html/www;
            index  index.html index.htm;
        }
    }
    server   {
        listen       10.0.0.11:80;
        server_name  blog.oldboy.com;
        location / {
            root   /usr/share/nginx/html/blog;
            index  index.html index.htm;
        }
    }
}
```

第6章 Nginx虚拟主机配置优化

所有配置都写入一个配置文件维护起来比较麻烦，如果修改错了，影响所有的页面，所以我们应该拆分nginx的配置文件为各个子配置

1.Nginx主配置文件

```
[root@web-7 /etc/nginx/conf.d]# cat /etc/nginx/nginx.conf

user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/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  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}
```

2.子配置文件www

```
[root@web-7 /etc/nginx/conf.d]# cat /etc/nginx/conf.d/01-www.conf
server   {
    listen       80;
    server_name  www.oldboy.com;
    location / {
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
    }
}
```

3.子配置文件blog

```
[root@web-7 /etc/nginx/conf.d]# cat /etc/nginx/conf.d/02-blog.conf
server   {
    listen       80;
    server_name  blog.oldboy.com;
    location / {
        root   /usr/share/nginx/html/blog;
        index  index.html index.htm;
    }
}
```

4.创建代码目录及首页

```
mkdir /usr/share/nginx/html/{www,blog}
echo "www" > /usr/share/nginx/html/www/index.html
echo "blog" > /usr/share/nginx/html/blog/index.html
```

5.检查语法重启服务

```
[root@web-7 ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@web-7 ~]# systemctl restart nginx
```

6.访问测试

```
[root@web-7 ~]# tail -1 /etc/hosts
10.0.0.7 www.oldboy.com blog.oldboy.com
[root@web-7 ~]# curl www.oldboy.com
www
[root@web-7 ~]# curl blog.oldboy.com
blog
```

第7章 Nginx日志

1.Nginx日志说明

```
Nginx的日志分为访问日志和错误日志两种，其中访问日志的格式我们可以根据自己的需求定义成不同的格式，比如为了方便日后的日志分析，我们可以将Nginx日志设置为json格式。
```

2.Nginx日志字段解释

```
$remote_addr              #记录客户端 IP 地址
$remote_user              #记录客户端用户名
$time_local               #记录通用的本地时间
$time_iso8601       #记录 ISO8601 标准格式下的本地时间
$request                      #记录请求的方法以及请求的 http 协议
$status                       #记录请求状态码(用于定位错误信息)
$body_bytes_sent    #发送给客户端的资源字节数，不包括响应头的大小
$bytes_sent               #发送给客户端的总字节数
$msec                         #日志写入时间。单位为秒，精度是毫秒。
$http_referer       #记录从哪个页面链接访问过来的
$http_user_agent    #记录客户端浏览器相关信息
$http_x_forwarded_for   #记录客户端 IP 地址
$request_length     #请求的长度（包括请求行， 请求头和请求正文）。
$request_time       #请求花费的时间，单位为秒，精度毫秒
# 注:如果 Nginx 位于负载均衡器， nginx 反向代理之后， web 服务器无法直接获取到客 户端真实的 IP 地址。
# $remote_addr 获取的是反向代理的 IP 地址。 反向代理服务器在转发请求的 http 头信息中，
# 增加 X-Forwarded-For 信息，用来记录客户端 IP 地址和客户端请求的服务器地址。
```

3.自定义Nginx日志格式

转换为json格式日志:

```
log_format json '{ "time_local": "$time_local", '
                        '"remote_addr": "$remote_addr", '
                        '"referer": "$http_referer", '
                        '"request": "$request", '
                        '"status": $status, '
                        '"bytes": $body_bytes_sent, '
                        '"agent": "$http_user_agent", '
                        '"x_forwarded": "$http_x_forwarded_for", '
                        '"up_addr": "$upstream_addr",'
                        '"up_host": "$upstream_http_host",'
                        '"upstream_time": "$upstream_response_time",'
                        '"request_time": "$request_time"'
    ' }';
    access_log  /var/log/nginx/access.log  json;
```

4.Nginx日志切割方法

为什么需要日志切割?

```
nginx日志默认是不切割的，这样当我们运行时间久了之后自然而然的会产生大量的日志，对我们日后分析不是很友好，所以工作中一般都是按天切割日志。
```

第一种方法：自己写脚本切割

```
cd /var/log/nginx/
tar zcf $(date +%F)-nginx-log.tar.gz access.log
rm -rf access.log
systemctl reload nginx
```

第二种方法：使用logrotate工具切割日志

```
logrotate是一款自动切割日志的工具。
如果使用了rpm安装nginx，会自动生成logrotate的配置文件
```

查看nginx的logrotate配置文件

```
[root@web-7 ~]# rpm -qc nginx|grep logrotate
/etc/logrotate.d/nginx
```

nginx的logrotate配置解释

```
[root@web-7 ~]# cat /etc/logrotate.d/nginx
/var/log/nginx/*.log {
        daily                           #按日切割
        missingok                       #忽略错误
        rotate 52                       #最多保留多少个存档，超过数量之后删除最久的
        compress                        #切割完成后将已经切割好的日志打包压缩
        delaycompress                   #将上一个日志文件的压缩推迟到下一个循环周期。仅在与compress结合使用时才有效。
        notifempty                      #如果日志为空，则不切割
        create 640 nginx adm            #以指定的权限创建权限的日志文件，同时重命名原始日志
        sharedscripts           #共享脚本，此处为空
        postrotate                      #当其他命令完成后执行的命令，这是是重新加载nginx进程的命令
                if [ -f /var/run/nginx.pid ]; then
                        kill -USR1 `cat /var/run/nginx.pid`
                fi
        endscript                       #最后执行的命令，此处为空
}
```

logrotate切割nginx日志实战

```
#1.安装压测生成访问日志
[root@web-7 ~]# yum install httpd-tools -y
[root@web-7 ~]# ab -n 1000 -c 100 http://127.0.0.1/

#2.查看未切割之前的日志
[root@web-7 ~]# ll /var/log/nginx/
总用量 932
-rw-r----- 1 nginx adm 949400 5月   6 21:32 access.log
-rw-r----- 1 nginx adm    700 5月   6 21:35 error.log

#3.执行logrotate命令
[root@web-7 ~]# /usr/sbin/logrotate -f /etc/logrotate.d/nginx

#4.查看切割后的文件，会发现只是重命名了，但是没有压缩，只是因为我们设置了本次日志的压缩放在下一次循环执行
```

```
[root@web-7 ~]# ll /var/log/nginx/
总用量 932
-rw-r----- 1 nginx adm      0 5月   6 21:35 access.log
-rw-r----- 1 nginx adm 949400 5月   6 21:32 access.log.1
-rw-r----- 1 nginx adm      0 5月   6 21:35 error.log
-rw-r----- 1 nginx adm    700 5月   6 21:35 error.log.1

#5.手动设置时间到明天
[root@web-7 ~]# date -s 20210507

#6.再次执行压测命令
[root@web-7 ~]# ab -n 1000 -c 100 http://127.0.0.1/

#7.重新执行logrotate命令
[root@web-7 ~]# /usr/sbin/logrotate -f /etc/logrotate.d/nginx

#8.再次查看日志情况
[root@web-7 ~]# ll /var/log/nginx/
总用量 932
-rw-r----- 1 nginx adm      0 5月   6 21:37 access.log
-rw-r----- 1 nginx adm 940000 5月   6 21:37 access.log.1
-rw-r----- 1 nginx adm   3351 5月   6 21:32 access.log.2.gz
-rw-r----- 1 nginx adm    205 5月   6 21:37 error.log
-rw-r----- 1 nginx adm    700 5月   6 21:35 error.log.1

#9.将切割命令写入定时任务
[root@web-7 ~]# crontab -l
#update time
* * * * * /usr/sbin/ntpdate time1.aliyun.com > /dev/null 2>&1

#logrotate nginx log
01 00 * * * /usr/sbin/logrotate -f /etc/logrotate.d/nginx >> logrotate_nginx
2>&1x
```