Nginx实用插件

安装Nginx

由于插件的兼容关系，请使用nginx-1.4.7这个版本

```
cd ~
# 下载nginx-1.4.7
wget https://nginx.org/download/nginx-1.4.7.tar.gz
tar -xvzf nginx-1.4.7.tar.gz
sudo mv nginx-1.4.7 /usr/local/
```

健康检查插件

主动检查内部上游的服健康状态插件，淘宝开源实现nginx_upstream_check_module。

安装

```
cd ~
wget https://github.com/yaoweibin/nginx_upstream_check_module/archive/master.zip
\
-O nginx_upstream_check_module-master.zip
unzip nginx_upstream_check_module-master.zip

cd /usr/local/nginx-1.4.7
# 补丁包
sudo patch -p1 < /home/wesley/nginx_upstream_check_module-
master/check_1.2.6+.patch

sudo ./configure \
    --prefix=/usr/local/nginx1.4 \
    --sbin-path=/usr/local/nginx1.4/nginx \
    --conf-path=/usr/local/nginx1.4/conf/nginx.conf \
    --error-log-path=/usr/local/nginx1.4/logs/error.log \
    --pid-path=/usr/local/nginx1.4/pid/nginx.pid \
    --add-module=/home/wesley/echo-nginx-module-0.61 \
    --with-http_stub_status_module \
    --add-module=/home/wesley/nginx_upstream_check_module-master

sudo make
sudo make install
```

配置

配置Nginx配置文件内容如下：

```
http {
    upstream cluster {
        # simple round-robin
```

```
        server 127.0.0.1:8080;
        server 127.0.0.1:8081;

        check interval=5000 rise=1 fall=3 timeout=4000;

        #check interval=3000 rise=2 fall=5 timeout=1000 type=ssl_hello;

        #check interval=3000 rise=2 fall=5 timeout=1000 type=http;
        #check_http_send "HEAD / HTTP/1.0\r\n\r\n";
        #check_http_expect_alive http_2xx http_3xx;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://cluster;
        }

        location /status {
            check_status;
            access_log   off;
            allow 192.168.120.0/255;
            deny all;
       }
    }

}
```

指令介绍

```
# 为上游服务添加健康检查，以及相关配置
check
    syntax: *check interval=milliseconds [fall=count] [rise=count]
    [timeout=milliseconds] [default_down=true|false]
    [type=tcp|http|ssl_hello|mysql|ajp|fastcgi]*
    default: *none, if parameters omitted, default parameters are
    interval=30000 fall=5 rise=2 timeout=1000 default_down=true type=tcp*

# 设置检查上游服务的访问URI
check_http_send
    syntax: *check_http_send http_packet*
    default: *"GET / HTTP/1.0\r\n\r\n"*

# 展示服务的健康检查状态
check_status
    syntax: *check_status [html|csv|json]*
```

更多的参考github的内容。

启动Nginx服务

测试验证

```
curl http://hostname/server/ip
```

关掉8080端口服务，验证是否还能将请求打到8080上面。

动态更新后端服务插件

安装

首先安装 nginx 动态 upstream 配置模块ngx_http_dyups_module。

```
cd ~
wget https://github.com/yzprofile/ngx_http_dyups_module/archive/v0.2.9.tar.gz \
-O ngx_http_dyups_module-0.2.9.tar.gz
tar -xvzf ngx_http_dyups_module-0.2.9.tar.gz

cd /usr/local/nginx-1.4.7
sudo ./configure \
    --prefix=/usr/local/nginx1.5 \
    --sbin-path=/usr/local/nginx1.5/nginx \
    --conf-path=/usr/local/nginx1.5/conf/nginx.conf \
    --error-log-path=/usr/local/nginx1.5/logs/error.log \
    --pid-path=/usr/local/nginx1.5/pid/nginx.pid \
    --add-module=/home/wesley/echo-nginx-module-0.61 \
    --with-http_stub_status_module \
    --add-module=/home/wesley/nginx_upstream_check_module-master \
    --add-module=/home/wesley/ngx_http_dyups_module-0.2.9

sudo make
sudo make install
```

配置

修改nginx的配置文件

```
# 简单指令以;结尾
worker_processes  4;
worker_rlimit_nofile 65535;
# 大括号属于块指令
events {
    worker_connections  1024;
}
http {
    include upstream.conf;
    # 默认主机
    server {
        listen 80;
        location / {
            proxy_pass http://$host;
        }
    }

    # 动态配置 upstream 的接口站点
    server {
        listen 81;
        location / {
            dyups_interface;     # 这个指令表示这边是接口站点
```

```
        }
    }

    # upstream 后面的 realserver,2 台 801, ,802
    server {
        listen 801;
        location / {
            return 200 "801";
        }
    }
    server {
        listen 802;
        location / {
            return 200 "802";
        }
    }
}
```

upstream.conf 配置

```
upstream server80 {
    server 127.0.0.1:801;
}
upstream server81 {
    server 127.0.0.1:802;
}
```

指令介绍

语法: dyups_interface 默认: none 配置段: location 启用配置 upstream 的接口

语法: dyups_read_msg_timeout time 默认: 1s 配置段: main 设置从共享内存中读取 commands 的超时时间，默认为 1 秒

语法: dyups_shm_zone_size size 默认: 2MB 配置段: main 设置存储 commands 的共享内存

语法: dyups_upstream_conf path 默认: none 配置段: main 这个指令用来指定 upstream 配置文件的路径,他会在启动的时候加载

语法: dyups_trylock on | off 默认: off 配置段: main 是否启用锁，如果启用了它，同一时刻有人在修改，那么将会返回 409

测试验证

增删改查后端服务

```
# 初次访问，$host变量设置为dyhost，原有的upstream中没有dyhost
# 所以访问获得错误信息
curl -H "host: dyhost" 127.0.0.1:80
<html>
<head><title>502 Bad Gateway</title></head>
<body bgcolor="white">
<center><h1>502 Bad Gateway</h1></center>
<hr><center>nginx/1.3.13</center>
</body>
</html>
```

```
# 添加/修改 一个upstream列表信息，名字为dyhost
curl -d "server 127.0.0.1:8080;server 127.0.0.1:8081;"
127.0.0.1:81/upstream/dyhost
success

# 检查dyhost列表服务的状态
curl -H "host: dyhost" 127.0.0.1:8080
Welcome to balancer!
curl -H "host: dyhost" 127.0.0.1:8081
Welcome to balancer!

# 通过ngx_http_dyups_module接口查看当前的upstream列表信息
curl 127.0.0.1:81/detail
host1
server 127.0.0.1:8088 weight=1 max_conns=0 max_fails=1 fail_timeout=10 backup=0
down=0

host2
server 127.0.0.1:8089 weight=1 max_conns=0 max_fails=1 fail_timeout=10 backup=0
down=0

dyhost
server 127.0.0.1:8089 weight=1 max_conns=0 max_fails=1 fail_timeout=10 backup=0
down=0
server 127.0.0.1:8088 weight=1 max_conns=0 max_fails=1 fail_timeout=10 backup=0
down=0

# 再次访问dyhost服务列表信息，成功返回信息
curl -H "host: dyhost" 127.0.0.1:80
Welcome to balancer!

# 删除一个upstream
curl -i -X DELETE 127.0.0.1:81/upstream/dyhost
success

# 再次查看当前后台服务信息
curl 127.0.0.1:81/detail
host1
server 127.0.0.1:8088 weight=1 max_conns=0 max_fails=1 fail_timeout=10 backup=0
down=0

host2
server 127.0.0.1:8089 weight=1 max_conns=0 max_fails=1 fail_timeout=10 backup=0
down=0
```

实际使用过程中，可以将$host固定下来，让后通过ngx_http_dyups_module的接口进行修改对应的upstream的列表信息，从而达到动态更新服务列表。

限流插件

Limit Upload Rate是淘宝开源的限流插件，针对客户端请求速率的限制。已经集成在tengine中。

安装

```
cd ~
wget https://github.com/cfsego/limit_upload_rate/archive/master.zip \
-O limit_upload_rate-master.zip
unzip limit_upload_rate-master.zip

cd /usr/local/nginx-1.4.7

# nginx中使用limit_upload_rate需要tengine的input body filter的支持
# 通过path来给nginx打补丁，for-nginx-1.4.4.patch基于nginx 1.4.4，可与nginx 1.5.x一起使用。
# 我们这里是nginx 1.4.7，所以选择for-nginx-1.4.4.patch补丁
sudo patch -p1 < /home/wesley/limit_upload_rate-master/for-nginx-1.4.4.patch
sudo ./configure \
    --prefix=/usr/local/nginx1.6 \
    --sbin-path=/usr/local/nginx1.6/nginx \
    --conf-path=/usr/local/nginx1.6/conf/nginx.conf \
    --error-log-path=/usr/local/nginx1.6/logs/error.log \
    --pid-path=/usr/local/nginx1.6/pid/nginx.pid \
    --add-module=/home/wesley/echo-nginx-module-0.61 \
    --with-http_stub_status_module \
    --add-module=/home/wesley/nginx_upstream_check_module-0.3.0 \
    --add-module=/home/wesley/ngx_http_dyups_module-0.2.9 \
    --add-module=/home/wesley/limit_upload_rate-master

sudo make
sudo make install
```

配置

```
# 由淘宝提供，需要limit_upload_rate的支持。
# 限制客户端上传速率，跟limit_rate用法类似
# 可以放在http, server, location, if in location指令块中
limit_upload_rate 5k;
limit_upload_rate_after 50k;
```

测验

通过上传页面检查

黑白名单插件

该插件很久没有维护了，而且不能正常使用，可以忽略该插件。

ngx_white_black_list 通过配置文件来指定黑白名单内容。

功能描述： 处在黑名单中的 ip 与网络，将无法访问 web 服务。 处在白名单中的 ip，访问 web 服务时，将不受 nginx 所有安全模块的限制。 支持动态黑名单（需要与 ngx_http_limit_req 配合），需要修改nginx代码这里就不提及了。如有需要，参考给到的ngx_white_black_list链接

安装

```
wget https://github.com/codehunte/ngx_white_black_list/archive/master.zip \
-O ngx_white_black_list-master.zip
unzip ngx_white_black_list-master.zip

cd /usr/local/nginx-1.4.7
sudo ./configure \
    --prefix=/usr/local/nginx1.7 \
    --sbin-path=/usr/local/nginx1.7/nginx \
    --conf-path=/usr/local/nginx1.7/conf/nginx.conf \
    --error-log-path=/usr/local/nginx1.7/logs/error.log \
    --pid-path=/usr/local/nginx1.7/pid/nginx.pid \
    --add-module=/home/wesley/echo-nginx-module-0.61 \
    --with-http_stub_status_module \
    --add-module=/home/wesley/nginx_upstream_check_module-0.3.0 \
    --add-module=/home/wesley/ngx_http_dyups_module-0.2.9 \
    --add-module=/home/wesley/limit_upload_rate-master \
    --add-module=/home/wesley/ngx_white_black_list-master

# make的时候ngx_white_black_list会有告警，导致编译不通过。修改Makefile文件，忽略告警。
sudo vim objs/Makefile
# 将CFLAGS =  -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Werror -
g换成下行
# 忽略Makefile中的警告
CFLAGS =  -pipe  -O -W -Wall -Wpointer-arith -Wno-unused-parameter -Wall -g

sudo make
sudo make install
```

配置

准备黑白名单列表文件，参考资料white.list、black.list、

nginx配置文件内容

```
# 简单指令以;结尾
worker_processes  4;
worker_rlimit_nofile 65535;
# 大括号属于块指令
events {
    worker_connections  1024;
}

http {
    # 定义 黑名单或白名单文件 空间key
    # 只能在http指令块中
    # white_black_list_conf可以配置多个 只需 zone=value 其中的value不同就可
    white_black_list_conf conf/white.list zone=white:2m;
    white_black_list_conf conf/black.list zone=black:4m;

    # 启用黑白名单，on/off，启用/关闭。
    # 在http、server、location下使用，功能默认是关闭
    # 配置在对应的指令块中，对应的指令块上下文生效
    white_list white on; #白名单  white 在整个http{}中都开启
    black_list black on; #黑名单  black 在整个http{}中都开启
```

```
    server {
        listen 80;
        root /data/www/;

        # 黑白名单配置调整接口，通过它可以调整配置内容
        location /sec_config {
            sec_config on;
        }
    }
}
```

测验

查看黑白名单定义内容

访问http://xxx/sec_config

查看zone_name 为white 的 list_path中的具体内容

http://xxx/sec_config?zone_name=white

向 zone_name 为white 中增加192.168.141.23

http://xxx/sec_config?zone_name=white&add_item=192.168.141.23

在 zone_name 为white 中删除192.168.141.23

http://xxx/sec_config?zone_name=white&delete_item=192.168.141.23