Nginx使用Lua脚本连接Redis验证身份并下载文件

目录

安装Nginx

下载

```
# 进入下载目录
```

```
cd /root/software

# 下载
wget http://nginx.org/download/nginx-1.18.0.tar.gz
```

解压安装包

```
# 进入安装目录
cd /root/program/

# 创建目录
mkdir nginx
cd nginx/

# 复制文件
cp /root/software/nginx-1.18.0.tar.gz .

# 解压文件
tar -zxvf nginx-1.18.0.tar.gz
```

安装依赖

```
# 安装gcc，nginx底层采用c++编写，因此需要gcc环境进行编译
yum install gcc-c++

# 安装pcre，一个Perl库，包括perl兼容的正则表达式，nginx的http模块使用pcre来解析正则表达式
yum install pcre pcre-devel

# 安装zlib，zlib库提供了多种压缩和解压缩方式，nginx使用zlib对http包的内容进行gzip
yum install zlib zlib-devel

# 安装openssl，openssl是一个强大的安全套接字层密码库，囊括了主要的密码算法、常用的秘钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用
yum install openssl openssl-devel
```

安装

```
# 进入安装目录
cd /root/program/nginx

# 创建安装目录
mkdir nginx

# 指定安装目录编译
cd /root/program/nginx/nginx-1.18.0
./configure --prefix=/root/program/nginx/nginx

# 编译
cd /root/program/nginx/nginx-1.18.0
make

# 安装
```

```
cd /root/program/nginx/nginx-1.18.0
make install

# 确认安装后文件，只是生成了启动文件，并没有启动
cd /root/program/nginx/nginx
ll
```

启动

```
# 进入目录
cd /root/program/nginx/nginx/sbin

# 创建软连接
ln -s /root/program/nginx/nginx/sbin/nginx /usr/bin/nginx

# 启动
nginx
```

修改后配置文件如下：

```
# 启动用户
user  root;
worker_processes  1;

#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;
}

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;

    #gzip  on;

    server {
        listen       80;
        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;
        }

    }

}
```

测试访问

```
# 地址
http://192.168.110.129/
```

安装LuaJIT

LuaJIT，即Lua及时编译器。

```
# 进入软件下载目录
cd /root/software

# 下载
wget https://luajit.org/download/LuaJIT-2.0.5.tar.gz

# 创建安装目录
cd /root/program/
mkdir LuaJIT
cd LuaJIT

# 解压
cp /root/software/LuaJIT-2.0.5.tar.gz .
tar xf LuaJIT-2.0.5.tar.gz
cd LuaJIT-2.0.5

# 编译并安装
make && make install

# 建立软连接，如果不建立软链接，则会出现share object错误
ln -s /usr/local/lib/libluajit-5.1.so.2 /lib64/libluajit-5.1.so.2
```

```
# 验证软连接
ll libluajit-5.1.so.2

# 加载lua库，加入到ld.so.conf文件
echo "/usr/local/LuaJIT/lib" >> /etc/ld.so.conf
```

安装ngx_devel_kit

```
# 进入软件下载目录
cd /root/software
mkdir ngx_devel_kit
cd ngx_devel_kit

# 下载
wget https://github.com/simpl/ngx_devel_kit/archive/v0.2.19.tar.gz

# 创建安装目录
cd /root/program/
mkdir ngx_devel_kit
cd ngx_devel_kit

# 解压
cp /root/software/ngx_devel_kit/v0.2.19.tar.gz .
tar xf v0.2.19.tar.gz
```

安装lua-nginx-module

```
# 进入软件下载目录
cd /root/software
mkdir lua-nginx-module
cd lua-nginx-module

# 下载
wget https://github.com/openresty/lua-nginx-module/archive/v0.10.13.tar.gz

# 创建安装目录
cd /root/program/
mkdir lua-nginx-module
cd lua-nginx-module

# 解压
cp /root/software/lua-nginx-module/v0.10.13.tar.gz .
tar xf v0.10.13.tar.gz
```

在已安装的Nginx中添加Lua模块

```
# 进入源码目录
cd /root/program/nginx/nginx-1.18.0

# 查看编译参数
nginx -V
```

```
# 构造已运行的Nginx的模块模块
./configure --prefix=/root/program/nginx/nginx

# 构造新的Nginx模块参数，注意根据实际情况修改目录地址
./configure --prefix=/root/program/nginx/nginx **新增模块配置，运行时删除** --add-
module=/root/program/ngx_devel_kit/ngx_devel_kit-0.2.19/ --add-module=/root/program/lua-
nginx-module/lua-nginx-module-0.10.13/

# 重新编译Nginx，make完成后不要继续输入“make install”，以免现在的nginx出现问题
make

# 以上完成后，会在objs目录下生成一个nginx文件
cd objs
./nginx -V

# 替换Nginx文件
mv /root/program/nginx/nginx/sbin/nginx /root/program/nginx/nginx/sbin/nginx.bak
cp nginx /root/program/nginx/nginx/sbin/

# 重新启动，重要，一定要停机后重新启动
nginx -s quit
ps -ef | grep nginx
nginx
ps -ef | grep nginx
```

Lua脚本测试

编写lua脚本

```
# 进入配置文件目录
cd /root/program/nginx/nginx/conf

# 创建脚本文件
vim test.lua

# 脚本内容
ngx.say("hello world.");
```

修改nginx.conf

```
# 在Server节点下添加下面的内置内容：
# 中文
charset utf-8;

location /lua {
    default_type 'text/html';
    lua_code_cache off;
    content_by_lua_file conf/test.lua;
}

# 文件下载服务
location /file_server {
```

```
    # 内部请求（即一次请求的Nginx内部请求），禁止外部访问，重要。
    internal;
    # 文件路径
    alias /root/data/files/;
    limit_rate 200k;
    # 浏览器访问返回200，然后转由后台处理
    #error_page 404 =200;
}

########################################################################

# 验证配置文件
nginx -t

# 重新加载配置文件
nginx -s reload

# 测试访问
http://192.168.110.129/lua
```

https://blog.csdn.net/yinjinshui/article/details/109738580

连接单例Redis

下载第三方依赖库

```
# Git地址
https://github.com/openresty/lua-resty-redis
```

复制第三方依赖库

```
# 进入系统第三方包目录
cd /usr/local/lib

# 将解压后的文件上传至该目录
# 目录名称：lua-resty-redis-master

# 包文件路径
cd /usr/local/lib/lua-resty-redis-master/lib/resty
```

在nginx配置文件中添加依赖

```
# 进入目录
cd /root/program/nginx/nginx/conf/

# 修改配置文件
vim nginx.conf

# 在http节点下添加下面配置，具体目录地址视具体服务器调整
# you do not need the following line if you are using
# the OpenResty bundle:
```

```
lua_package_path "/usr/local/lib/lua-resty-redis-master/lib/resty/?.lua;;";
```

编写脚本文件

```
# 进入脚本文件目录
cd /root/program/nginx/nginx/conf

# 修改脚本
vim test.lua

# 脚本内容如下所示：
```

```
local redis = require "resty.redis"
--local cjson = require "cjson"

local red = redis:new()

-- 从header中取值
--local token = ngx.req.get_headers()[“token”];
local args = ngx.req.get_uri_args();

-- 获取参数
local userid = args.userid;
local showlength = args.showlength;

red:set_timeouts(1000, 1000, 1000) -- 1 sec

local ok, err = red:connect("10.120.160.110", 6379)
if not ok then
    ngx.say("failed to connect: ", err)
    return
end

local redis_key = "Navigation:"..userid;
local res, err = red:get(redis_key)

if not res then
    ngx.say("failed to get: ", err)
    return
end

if res == ngx.null then
    ngx.say("<b>not found.</b>")
    return
end

--ngx.say("<b>[Navigation:"..userid.."]: </b>", string.sub(res,1,showlength), "<br/>")

-- 请求文件
res = ngx.location.capture("/file_server/1.zip");

if res.status ~= 200 then
    ngx.print("Not found file.");
    return;
end
```

```
-- 下载文件
ngx.header["Content-Disposition"] = "attachment; filename=1.zip";
ngx.header["Content-Type"] ="application/x-zip-compressed";
ngx.print(res.body);
```

执行

```
# 检查nginx配置
nginx -t

# 重新加载配置
nginx -s reload

# 浏览器中输入测试地址
http://192.168.110.129/lua?userid=10076&showlength=500

# 请求后可以直接下载文件。
```

连接Redis集群

下载第三方依赖库

```
# Redis集群连接库依赖连接Redis单实例连接库[resty.redis]，因此需要提前安装此第三方依赖库

# resty-redis-cluster
https://github.com/steve0511/resty-redis-cluster

# lua-resty-lock
https://gitee.com/mirrors_bungle/lua-resty-lock/tree/master
```

复制第三方依赖库

```
# 进入系统第三方包目录
cd /usr/local/lib

# 将解压后的文件上传至该目录
# 目录名称：resty-redis-cluster-master

# 包文件路径
cd /usr/local/lib/resty-redis-cluster-master/lib/resty
cd /usr/local/lib/lua-resty-lock/lib/resty
```

在nginx配置文件中添加依赖

```
# 进入目录
cd /root/program/nginx/nginx/conf/
```

```
# 修改配置文件
vim nginx.conf

# 修改内容如下，具体视需求而定
```

```
# 在http节点下添加下面配置，具体目录地址视具体服务器调整
lua_package_path "/usr/local/lib/lua-resty-redis-master/lib/?.lua;;;/usr/local/lib/lua-resty-lock/lib/resty/?.lua;;;/usr/local/lib/resty-redis-cluster-master/lib/?.lua;;;/usr/local/lib/resty-redis-cluster-master/lib/resty/?.lua;;";

# 添加缓存配置
lua_shared_dict redis_cluster_slot_locks 100k;

# 在Server节点下添加下面的内置内容：
# 中文
charset utf-8;

location /lua/cluster/ {
    default_type 'text/html';
    lua_code_cache off;
    content_by_lua_file conf/test_cluster.lua;
}

# 文件下载服务
location /file_server {
    # 内部请求（即一次请求的Nginx内部请求），禁止外部访问，重要。
    internal;
    # 文件路径
    alias /root/data/files/;
    limit_rate 200k;
    # 浏览器访问返回200，然后转由后台处理
    #error_page 404 =200;
}
```

```
# 验证配置文件
nginx -t

# 重新生效配置文件
nginx -s reload
```

编写脚本文件

```
# 进入脚本文件目录
cd /root/program/nginx/nginx/conf

# 修改脚本
vim test_cluster.lua

# 脚本内容如下所示：
```

```
-- 获取请求信息
local request_uri = ngx.var.request_uri;
local args = ngx.req.get_uri_args();
```

```
-- 获取参数
local key = args.key;
local internal_file_name = args.filename;
local file_type = request_uri:match(".+%.(%w+)");

local config = {
    name = "testCluster",                    --rediscluster name
    serv_list = {                            --redis cluster node list(host and port),
        { ip = "10.120.160.110", port = 7000 },
        { ip = "10.120.160.114", port = 7000 },
        { ip = "10.120.68.96", port = 7000 },
        { ip = "10.120.160.110", port = 7001 },
        { ip = "10.120.160.114", port = 7001 },
        { ip = "10.120.68.96", port = 7001 }
    },
    keepalive_timeout = 60000,               --redis connection pool idle timeout
    keepalive_cons = 1000,                   --redis connection pool size
    connect_timeout = 1000,                --timeout while connecting
    read_timeout = 1000,                     --timeout while reading
    send_timeout = 1000,                     --timeout while sending
    max_redirection = 5,                     --maximum retry attempts for redirection
    max_connection_attempts = 1,             --maximum retry attempts for connection
    auth = "***************"                --set password while setting auth
}

local redis_cluster = require "rediscluster"
local red_c = redis_cluster:new(config)

local redis_key = key;
local res, err = red_c:get(redis_key)
if err or (not res) or (res == ngx.null) then
    ngx.exit(401);
    return;
end

return ngx.exec("/file_server/"..internal_file_name.."."..file_type);
```

执行

```
# 检查nginx配置
nginx -t

# 重新加载配置
nginx -s reload

# 浏览器中输入测试地址
http://192.168.110.129/lua/cluster/张三.zip?key=name&filename=1
http://192.168.110.129/lua/cluster/test01.docx?key=name&filename=test

# 请求后可以直接下载文件。
```

附录-nginx.conf配置文件

```
user  root;
worker_processes  1;

#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;
}

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;

    #gzip  on;

    # you do not need the following line if you are using
    # # the OpenResty bundle:
    lua_package_path "/usr/local/lib/lua-resty-redis-master/lib/?.lua;;;/usr/local
/lib/lua-resty-lock/lib/?.lua;;;/usr/local/lib/resty-redis-cluster-master/lib/?.lua;;;
/usr/local/lib/resty-redis-cluster-master/lib/resty/?.lua;;";

    # 添加缓存配置
    lua_shared_dict redis_cluster_slot_locks 100k;

    server {
        listen       80;
        server_name  localhost;

        # 中文
        charset utf-8;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

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

```
    location /lua {
        default_type 'text/html';
        lua_code_cache off;
        content_by_lua_file conf/test.lua;
    }

    location /lua/cluster {
        default_type 'text/html';
        lua_code_cache off;
        content_by_lua_file conf/test_cluster.lua;
    }

    # 文件下载服务
    location /file_server {
        # 内部请求（即一次请求的Nginx内部请求），禁止外部访问，重要。
        internal;
        lua_code_cache off;
        alias /root/data/files/;
        limit_rate 1024k;
        add_header Cache-Control no-store; # 禁止浏览器缓存文件
        # 浏览器访问返回200，然后转由后台处理
        #error_page 404 =200;
    }

    # 文件下载服务
    location /test_file_server {
        # 内部请求（即一次请求的Nginx内部请求），禁止外部访问，重要。
        #internal;
        lua_code_cache off;
        set $filepath "";
        # 文件路径
        content_by_lua_file  conf/test_file_server.lua;
        #return 200 $filepath;
        #try_files $filepath $filepath/;
    }

    # 文件下载服务
    location /test_find_file {
        # 内部请求（即一次请求的Nginx内部请求），禁止外部访问，重要。
        internal;
        lua_code_cache off;
        alias /root/data/files/;
        limit_rate 200k;
        # 浏览器访问返回200，然后转由后台处理
        #error_page 404 =200;
    }

    #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;
    }

}
```

```
}
```