OpenResty安装使用

在Nginx新版本中没有直接提供Lua模块的集成，Lua脚本模块已经作为第三方集成模块中去了，查看Nginx博客地址可知，集成到OpenResty中，OpenResty由主要国人章亦春维护的项目。

OpenResty是一个基于 Nginx 与 Lua 的高性能 Web 平台，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。用于方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。

充分利用 Nginx的非阻塞 I/O 模型，不仅仅对 HTTP 客户端请求,甚至于对远程后端诸如 MySQL、PostgreSQL、Memcached 以及 Redis 等都进行一致的高性能响应。

安装OpenResty

有二进制安装包、源码构建两种方式来安装OpenResty

安装OpenResty

官网提供二进制安装包，以CentOS为例，通过yum命令即可安装。其他系统参考官网。

```
# 添加 openresty 仓库
sudo yum install yum-utils
sudo yum-config-manager --add-repo
https://openresty.org/package/centos/openresty.repo
# 安装OpenResty
sudo yum install openresty
# 安装命令行工具 resty
sudo yum install openresty-resty
```

构建OpenResty

下载解压

```
# 安装依赖工具
yum install -y pcre-devel openssl-devel gcc curl
# 下载源码包
wget https://openresty.org/download/openresty-1.15.8.1.tar.gz
# 解压构建并安装
tar -xzvf openresty-1.15.8.1.tar.gz
sudo mv openresty-1.15.8.1 openresty
sudo mv openresty /usr/local
cd openresty/
```

编译选项

可以通过./configure --help来了解，编译的各种选项，例如

```
......
```

```
--without-http_srcache_module      disable ngx_http_srcache_module
--without-http_lua_module          disable ngx_http_lua_module
--without-http_lua_upstream_module disable ngx_http_lua_upstream_module
--without-http_headers_more_module disable ngx_http_headers_more_module
--without-http_array_var_module    disable ngx_http_array_var_module
--without-http_memc_module         disable ngx_http_memc_module
--without-http_redis2_module       disable ngx_http_redis2_module
--without-http_redis_module        disable ngx_http_redis_module
--without-http_rds_json_module     disable ngx_http_rds_json_module
--without-http_rds_csv_module      disable ngx_http_rds_csv_module
--without-stream_lua_module        disable ngx_stream_lua_module
--without-ngx_devel_kit_module     disable ngx_devel_kit_module
--without-http_ssl_module          disable ngx_http_ssl_module
--without-stream_ssl_module        disable ngx_stream_ssl_module
......
```

你可以这样来添加和去除你想要选择的模块。

```
./configure --prefix=/opt/openresty \ #指定编译目录，默认为/usr/local/openresty
            --with-luajit \
            --without-http_redis2_module \
            --with-http_iconv_module \
            --with-http_postgres_module
```

你也可以使用默认配置

```
sudo ./configure
```

编译安装

```
sudo make
sudo make install
```

启动OpenResty

实际上openresty使用的是nginx，openresty文件通过链接的形式，连到/usr/local/openresty/nginx/sbin/nginx文件，所以使用openresty与使用nginx并无太大区别。

启动

```
sudo bin/openresty
```

关闭

```
sudo bin/openresty -s stop
```

测试Lua脚本

编写Lua脚本

```
cd /usr/local/openresty/
sudo mkdir lua_scripts
cd lua_scripts
sudo vim mydata.lua
```

往mydata.lua中输入内容

```
local _M = {}

local data = {
   dog = 5,
   cat = 3,
   pig = 1,
}

function _M.get_age(name)
   return data[name]
end
return _M
```

调整配置

```
sudo vim /usr/local/openresty/nginx/conf/nginx_lua.conf
```

配置内容

```
worker_processes  1;
events {
    worker_connections 1024;
}

http {
    lua_package_path "/usr/local/openresty/lua_scripts/?.lua;;";
    server {
        listen 8080;
        location /lua {
            default_type text/html;
            content_by_lua_block {
                local mydata = require "mydata"
               ngx.say(mydata.get_age("dog"))
            }
        }
    }
}
```

启动测试

```
sudo bin/openresty -c conf/nginx_lua.conf
```

有同学可能在奇怪，为什么配置文件不是nginx/conf/nginx_lua.conf。那是因为使用nginx/conf/nginx_lua.conf路径，发现报错，错误内容告诉我们找不到配置文件，错误信息里面的路径上多了一个nginx目录。

输出内容

```
curl http://localhost:8080/lua
5
```

看到输出内容，表示已经安装成功，并且成功的访问了，实现的一个小小的lua脚本。

使用Redis

参考《redis_demo.lua》、《nginx_lua_redis.conf》文件

使用MySQL

参考《redis_demo.lua》、《nginx_lua_redis.conf》文件

使用HTTP客户端

OpenResty默认没有提供Http客户端，需要使用第三方提供，从github上搜索相应的客户端，比如lua-resty-http

下载插件

下载lua-resty-http客户端到lualib

```
cd /usr/local/openresty/lualib/resty/
sudo wget https://raw.githubusercontent.com/pintsized/lua-resty-
http/master/lib/resty/http_headers.lua
sudo wget https://raw.githubusercontent.com/pintsized/lua-resty-
http/master/lib/resty/http.lua
```

lua脚本

```
cd /usr/local/openresty/lua_scripts
sudo vim http_demo.lua
```

输入内容，参考《http_demo.lua》

nginx配置

参考《nginx_lua_redis.conf》文件，在http部分添加谷歌dns解析器，修改对应的lua脚本文件。

```
resolver 8.8.8.8;
```

Lua模板渲染器

动态web网页开发是Web开发中一个常见的场景，Lua中也有许多模板引擎，我们通过lua-resty-template来完成动态页面的渲染。

下载lua-resty-template

```
cd /usr/local/openresty/lualib/resty/
wget https://raw.githubusercontent.com/bungle/lua-resty-
template/master/lib/resty/template.lua
mkdir /usr/local/openresty/lualib/resty/html
cd /usr/local/openresty/lualib/resty/html
wget https://raw.githubusercontent.com/bungle/lua-resty-
template/master/lib/resty/template/html.lua
```

使用

nginx配置文件，添加模板解析路径

```
#首先匹配nginx下面的模板目录
set $template_location "/templates";
#然后匹配指定的模板根目录，我们这里是/usr/local/openresty/lua_scripts
set $template_root "/usr/local/openresty/lua_scripts";
#或者通过root指令来切换路径
root /usr/local/openresty/lua_scripts;
```

lua内容，html_template.lua

```
local template = require("resty.template")

local context = {
    title = "lua模板渲染",
    name = "hash同学",
    description = "<script>alert(1);</script>",
    age = 20,
    hobby = {"电影", "音乐", "阅读"},
    score = {语文 = 90, 数学 = 80, 英语 = 70},
    score2 = {
        {name = "语文", score = 90},
        {name = "数学", score = 80},
        {name = "英语", score = 70},
    }
}
template.render("resty_template.html", context)
```

resty_template.html文件的html内容，将文件放置在/usr/local/openresty/lua_scripts目录或者nginx的templates目录

```
<html>
    <head>
        <title>{* titile *}</title>
    </head>
```

```
    <body>
        {# 不转义变量输出 #}
        姓名：{* string.upper(name) *}<br/>
        {# 转义变量输出 #}
        简介：{{description}}<br/>
        {# 可以做一些运算 #}
        年龄: {* age + 1 *}<br/>
        {# 循环输出 #}
        爱好：
        {% for i, v in ipairs(hobby) do %}
           {% if i > 1 then %}，{% end %}
           {* v *}
        {% end %}<br/>

        成绩：
        {% local i = 1; %}
        {% for k, v in pairs(score) do %}
           {% if i > 1 then %}，{% end %}
           {* k *} = {* v *}
           {% i = i + 1 %}
        {% end %}<br/>
        成绩2：
        {% for i = 1, #score2 do local t = score2[i] %}
           {% if i > 1 then %}，{% end %}
            {* t.name *} = {* t.score *}
        {% end %}<br/>
        {# 中间内容不解析 #}
        {-raw-}{(file)}{-raw-}
    </body>
</html>
```

{* var *}：变量输出；

{{ var }}：变量转义输出；

{% code %}：代码片段；

{# comment #}：注释；

{-raw-}：中间的内容不会解析，作为纯文本输出；

{(include_file)}：包含另一个模板文件