

28 OpenResty 的作用及环境搭建

更新时间：2020-03-19 11:05:45



“当你做成功一件事，千万不要等待着享受荣誉，应该再做那些需要的事。—— 巴斯德”

前言

我们在前面的章节中介绍了 **Nginx** 的许多内容，从 **Nginx** 的安装到配置以及优化等等，大家可以从中学体会到 **Nginx** 的出色性能。基于优秀的事件通知机制，**Nginx** 服务可以稳定的提供高并发服务。

大多数时候通过合理的参数配置以及内核优化等方案就可以满足我们的业务场景，但是有些时候我们可能要扩展 **Nginx** 的某些功能，或者更最大限度的榨取 **Nginx** 的性能，这时候我们可以通过编写 **Nginx** 扩展 **Module** 来实现。**Nginx** 扩展可以充分的利用 **Nginx** 异步事件机制，拥有非常高的性能。但是要用 **C** 语言实现的，并且要透彻的理解 **Nginx** 的各个处理过程，难度比较高。

OpenResty 给我们提供了另外一种方式，我们可以通过 **Lua** 代码来实现我们的功能。

OR 简介

OpenResty 又称 **Ngx_Openresty**，简称 **OR**，由国人章亦春(江湖人称 **春哥**)开发和维护。这是一个基于 **Nginx** 和 **Lua** 的高性能 **web** 平台。它的内部集成了大量优秀的 **Lua** 库，第三方模块等。这样开发人员就可以使用 **Lua** 脚本调用 **Nginx** 支持的各种 **C** 以及 **Lua** 模块。快速构造出足以胜任 **C10K** 乃至 **C1000K** 以上单机并发连接的高性能 **Web** 应用系统。

OpenResty 的目标是让你的 Web 服务直接跑在 Nginx 服务内部，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 等进行一致的高性能响应。

— OR 官网

我们可以从官网上找到 OpenResty 支持的第三方模块：

- [NginxDevelKit](#)
- [LuaCjsonLibrary](#)
- [LuaNginxModule](#)
- [LuaRdsParserLibrary](#)
- [LuaRedisParserLibrary](#)
- [LuaRestyCoreLibrary](#)
- [LuaRestyDNSLibrary](#)
- [LuaRestyLockLibrary](#)
- [LuaRestyLrucacheLibrary](#)
- [LuaRestyMemcachedLibrary](#)
- [LuaRestyMySQLLibrary](#)
- [LuaRestyRedisLibrary](#)
- [LuaRestyStringLibrary](#)
- [LuaRestyUploadLibrary](#)
- [LuaRestyUpstreamHealthcheckLibrary](#)
- [LuaRestyWebSocketLibrary](#)
- [LuaRestyLimitTrafficLibrary](#)
- [LuaRestyShellLibrary](#)

这里面几乎包含了我们平时常用的所有模块，我们可以充分的利用这些模块完成我们的应用开发。

在实际的工作中，我接触的使用到 OpenResty 最多的场景就是网关的开发。Lua 的脚本特性，加上 Nginx 的高效性，二者有效的融合在了一起，极大的加速了网关的开发。

安装

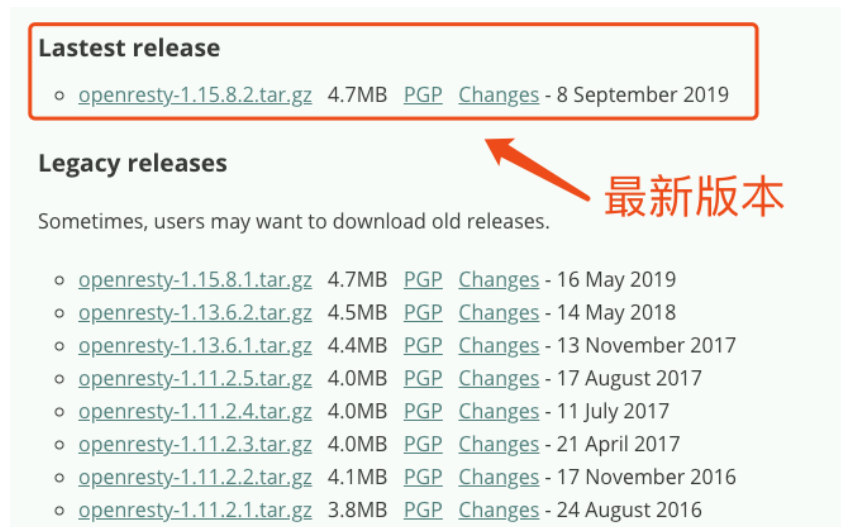
工欲善其事必先利其器

在深入了解 OpenResty 之前，我们必须先让安装框架，让它 run 起来才行。

打开 **OpenResty** 的安装页面，老规矩，作为一个程序员，我们推荐使用自己编译的方式，这样可以最大程度的个性化定制功能。

逐步编译

打开 **download** 页面，下载最新版本。



The screenshot shows the OpenResty download page. A red box highlights the 'Lastest release' section, which contains the link for 'openresty-1.15.8.2.tar.gz' (4.7MB, PGP, Changes - 8 September 2019). Below this is the 'Legacy releases' section, which lists several older versions. A red arrow points from the text '最新版本' (Latest Version) to the 'Lastest release' section.

Lastest release

- [openresty-1.15.8.2.tar.gz](#) 4.7MB [PGP](#) [Changes](#) - 8 September 2019

Legacy releases

Sometimes, users may want to download old releases.

- [openresty-1.15.8.1.tar.gz](#) 4.7MB [PGP](#) [Changes](#) - 16 May 2019
- [openresty-1.13.6.2.tar.gz](#) 4.5MB [PGP](#) [Changes](#) - 14 May 2018
- [openresty-1.13.6.1.tar.gz](#) 4.4MB [PGP](#) [Changes](#) - 13 November 2017
- [openresty-1.11.2.5.tar.gz](#) 4.0MB [PGP](#) [Changes](#) - 17 August 2017
- [openresty-1.11.2.4.tar.gz](#) 4.0MB [PGP](#) [Changes](#) - 11 July 2017
- [openresty-1.11.2.3.tar.gz](#) 4.0MB [PGP](#) [Changes](#) - 21 April 2017
- [openresty-1.11.2.2.tar.gz](#) 4.1MB [PGP](#) [Changes](#) - 17 November 2016
- [openresty-1.11.2.1.tar.gz](#) 3.8MB [PGP](#) [Changes](#) - 24 August 2016

我们下载的最新的是 **1.15.8.2** 版本，所以按步骤执行如下命令：

```
tar -xvf openresty-1.15.8.2.tar.gz
cd openresty-1.15.8.2
./configure -j2
make -j2
sudo make install
```

这样我们就完成了安装工作。

测试

是骡子是马，拉出来溜溜

安装成功之后，我们简单的体验与喜爱 **OpenResty** 的功能。

我们先创建几个目录，用于保存 **OpenResty** 的配置目录和日志目录：

```
mkdir ~/work
cd ~/work
mkdir logs/ conf/
```

配置文件

我们在上面的 **~/work/conf** 目录下面创建一个 **nginx.conf** 配置文件，内容如下：

```

worker_processes 1;
error_log logs/error.log;
events {
    worker_connections 1024;
}
http {
    server {
        listen 8081;
        location / {
            default_type text/html;
            content_by_lua_block {
                ngx.say("<p>hello, world</p>")
            }
        }
    }
}

```

大家可能注意到了，这里的配置文件和我们在前面学习的 **Nginx** 的配置文件非常的相似。因为 **OpenResty** 本质上是 **Nginx** 的一个扩展模块，在这个模块中增加了一些自定义的指令，比如上面的 **content_by_lua_block** 等，所以我们的配置文件，包括 **OpenResty** 的各种操作都和 **Nginx** 是相同的。大家一定要注意这一点。

启动服务器

我们启动 **OpenResty** 服务器。

```
/usr/local/openresty/bin/openresty -c ~/work/conf/nginx.conf
```

查看进程状况：

```

[root@53a367fac3e8 conf]# ps -ef | grep nginx
root      8817      1  0 15:04 ?        00:00:00 nginx: master process /usr/local/openresty/bin/openresty
nobody    8828    8817  0 15:13 ?        00:00:00 nginx: worker process
root      8830     17  0 15:16 pts/1    00:00:00 grep --color=auto nginx

```

请求服务

```

[root@53a367fac3e8 work]#
[root@53a367fac3e8 work]# curl http://localhost:8081/
<p>hello, world</p>
[root@53a367fac3e8 work]#

```

如上可以知道，**OpenResty** 已经可以成功的接收我们的请求了。

总结

我们在这篇文章中简单的介绍了 **OpenResty** 的概念以及用途，以及安装的内容。大家要仔细看看这个相应的文档，更多的了解 **OpenResty**，这是非常有必要的。

}