# hi-nginx
# 多语言通用服务器
# 使用手册

hi-nginx@webcpp.net

2017 年 8 月 25 日

## 目录

**摘要**

hi-nginx 是一款基于 nginx 写成的通用服务器。它既是 web server，也是 application server；它不仅继承了 nginx 的全部功能，完全兼容 nginx，而且支持多种语言混合开发 web 应用。它性能强劲，易于开发，部署方便。目前，hi-nginx 支持混合使用 c++，python，lua 以及 java 同时进行 web 应用开发。用户应该根据应用场景的实际需要，细粒度地选择最为合适的开发语言，最大限度的发挥 hi-nginx 的潜能。

# 1 导言

hi-nginx 既是 web server，也是 application server。这是它区别于 nginx 的最主要的特点。作为前者，它跟 nginx 一样，可作静态资源服务器，可作反向代理服务器，还可作负载均衡服务器，一切都一如 nginx。作为后者，它让 c++ 程序员，python 程序员，以及 lua 程序员和 java 程序员写的 web application 完全运行在 nginx 服务器内部，从而可以轻松提供高性能的、支持大并发的 web application。

为什么 hi-nginx 要同时支持四种编程语言？其原因有三。第一，我最常用的编程语言是 c++，它必须被支持；而且它非常快，非常适合处理“热点”业务。第二，python 库资源极为丰富，非常适合处理常规业务，几乎没有它未曾涉猎的开发领域，因而它能够极大地加快开发速度。第三，lua 比 python 快，但是库资源不及后者，支持它是为了方便处理那些介于”热点”业务和“常规”业务之间的业务。第四，java 在 web 领域有很深的积淀，支持它虽非必要却不失为有趣的选择。

因此，使用 hi-nginx，让其发挥出最大潜能，需要开发者同时熟知 c++、python、lua、java 四种编程语言。这并不是非常高的要求。实际上，这四种语言都非常易学易用，尽管并不是所有人都认同这一点。当然，用户只熟知其中的某一种编程语言也无妨——即便是对 python 程序员而言，hi-nginx 也能提供非常高效的并发处理能力。

目前，把 c 或者 c++ 运用于 web 应用开发的最主要的方式是 script 加 c 或者 c++ extension。这样做的目的其实主要地还是为了解决 script 可能无法胜任“热点”业务的问题。首先是脚本，然后是 c 或者 c++ 扩展，最后再回到脚本。这条性能优化路线在 web 应用开发中极为常见。hi-nginx 不仅支持这条路线——用户照样可以为 python 和 lua 开发 c 或 c++ 扩展——而且还支持另一条路线，即直接用 c++ 写 web application。这条路线省去了“脱裤子”的麻烦；对于能写 c 或 c++ 扩展的程序员而言，这是极为便利的。当然，如果用户仅仅能写脚本，hi-nginx 也保证提供比已有的反向代理方案更强大的并发能力。

hi-nginx 致力于增强用户的工作，而不是改变用户的工作。当用户不满意它时，用户可以安全地“回滚”至之前的工作状态，而不会产生任何损失。

# 2 快速部署

hi-nginx 目前仅仅支持 Linux 系统。

## 2.1 下载源码包

hi-nginx 是一个开源在https://github.com/webcpp/hi-nginx上的一个开源项目。用户可以直接从该地址下载 hi-nginx 的源码包。建议下载最新的正式发布版，前往https://github.com/webcpp/hi-nginx/releases查看并点击下载最新版本即可。

不建议直接使用 git clone 命令下载未正式分布的源码包。

## 2.2 编译及安装

要编译 hi-nginx，需要安装一些依赖软件。包括：

- gcc
- gcc-c++
- make
- pcre-devel
- zlib-devel
- openssl-devel
- hiredis-devel
- python-devel
- boost-devel
- luajit-devel
- jdk

假设用户使用的 Linux 是 CentOS，那么很简单，执行：

```
sudo yum install gcc gcc-c++ make pcre-devel zlib-devel openssl-devel hiredis-devel python-devel boost-devel luajit-devel
```

即可。如果用户使用的是 Ubuntu，请执行：

```
sudo apt-get install build-essential libpcre3-dev zlib1g-dev libssl-dev libhiredis-dev python-dev libboost-all-dev
    libluajit-5.1-dev
```

关于 java 环境的安装，推荐采用 sdkman 来进行：

```
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk selfupdate force
sdk install java
```

安装以上依赖软件之后，解压缩 hi-nginx 源码包。进入源码目录后，会看到一个演示性的安装脚本 install-demo.sh。如果用户仅仅是尝试 hi-nginx，可以直接运行该该脚本。它的内容很简单：

```
#!/bin/bash
./configure  --with-http_ssl_module \
             --with-http_v2_module \
             --prefix=/home/centos7/nginx \
             --add-module=ngx_http_hi_module
```

这个脚本告诉用户，编译 hi-nginx 与编译 nginx 没有什么不同，只需使用配置选项--add-module 指定 ngx_http_hi_module，其他则一如后者。

执行完以上步骤之后，就只剩下 make && make install 了。安装目录是--prefix 选项指定的/home/centos7/nginx。

## 2.3 基本测试

现在，hi-nginx 已经安装到了/home/centos7/nginx 目录中。用户可以从该目录启动 hi-nginx。启动方法与 nginx 无异。

但是，既然已经安装好了 hi-nginx，在正式使用它之前，就应该测试一下它 application server 功能，确保安装的成功。为此，用户需要做两件事情。第一件事情是安装 redis 服务器，因为 hi-nginx 的会话功能需要它。安装的方法也很简单：sudo yum install redis。第二件事情是下载https://github.com/webcpp/hi_demo上的演示代码。git clone https://github.com/webcpp/hi_demo.git 即可。演示代码假定 hi-nginx 的安装目录为/home/centos7/nginx，执行 make && make install 即可。如果安装目录不同于此，需修改 hi_demo 目录下 Makefile 中 NGINX_INSTALL_DIR 变量的值。

hi_demo 目录下有两个特殊文件，一个是 demo.html，一个是 demo.conf。前者可带领用户测试 hi-nginx，后者则用于配置 hi-nginx 以正确加载 hi_demo。用户执行以下命令就可以正确安装这两个文件：

```
install demo.html /home/centos7/nginx/html
install demo.conf /home/centos7/nginx/conf
```

完成以上步骤之后，就可以正式测试 hi-nginx 了。

进入 hi-nginx 安装目录，执行 sbin/nginx -c conf/demo.conf，然后访问http://localhost:8765/demo.html，按链接指引点击即可。如无意外，hi-nginx 会通过所有测试。如果遇到问题，用户可查看/home/centos7/nginx/conf/demo.conf 文件：

```
    hi_need_cache on;       #缓存开关
    hi_cache_size 10;       #缓存容器大小
    hi_cache_expires 300s;  #缓存过期时间
    hi_need_headers off;    #http header 开关
    hi_need_cookies off;    #http cookie 开关
    hi_need_session off;    #http session 开关
    hi_session_expires 300s; #http session 过期时间
    hi_redis_host 127.0.0.1; #redis 主机
    hi_redis_port 6379;  #redis 端口
    hi_java_classpath
         "-Djava.class.path=.:/home/centos7/nginx/java:/home/centos7/nginx/java/hi-nginx-java.jar:/home/centos7/nginx/java/freemarker
         #java classpath
    hi_java_servlet_cache_expires 300s; #java servlet缓存过期时间
    hi_java_servlet_cache_size 10; #java servlet缓存容器大小
    hi_java_version 8; #java 版本

    expires 10s;
    location = /hello {
        hi_need_cache off;
        hi hi/hello.so;
    }



    location ^~ /form {
        rewrite ^/form/(\d+)$ /form/?item=$1 break;
        hi_need_cache off;
        hi_need_headers on;
        hi_need_cookies on;
        hi hi/form.so;
    }

    location = /error {
        hi hi/error.so;
    }

    location = /redirect {
        hi hi/redirect.so;
    }

    location = /empty {
        hi hi/empty.so;
    }

    location = /math {
        hi_cache_expires 5s;
        hi hi/math.so;
    }

```

```
48          location = /session {
49              hi_need_cache off;
50              hi_need_session on;
51              hi_session_expires 30s;
52              hi hi/session.so;
53          }
54
55      location = /pyecho {
56              hi_need_cache off;
57              hi_python_content "hi_res.status(200)\nhi_res.content('hello,world')";  #运行python内容块
58
59      }
60
61      location ~ \.py$ {
62              hi_need_cache off;
63              hi_need_headers on;
64              hi_need_session on;
65              hi_session_expires 30s;
66              hi_python_script python;                                    #运行python脚本
67      }
68
69      location ~ \.do$ {
70              rewrite ^/(.*)\.do$ /$1.py break;
71              hi_need_cache off;
72              hi_need_headers on;
73              hi_need_cookies on;
74              hi_python_script python;
75          }
76
77      location = /luaecho {
78              hi_need_cache off;
79              hi_lua_content "hi_res:status(200)\nhi_res:content('hello,world')";  #运行lua内容块
80
81      }
82
83      location ~ \.lua$ {
84              hi_need_cache off;
85              hi_need_headers on;
86              hi_need_session on;
87              hi_session_expires 30s;
88              hi_lua_script lua;                                          #运行lua脚本
89      }
90
91      location = /jhello {
92              hi_need_cache off;
93              hi_java_servlet_cache_expires 60s;
94              hi_java_servlet hi/jhello;                                  #调用hi/jhello类
95      }
96
97          location / {
98              root  html;
99              index index.html index.htm;
100         }
```

对照以上配置检查哪里出现不意状态。也可查看logs/error.log，看看有何种提示信息。

# 3 起步

## 3.1 hi-project

hi-project 是一个辅助脚本，安装在/home/centos7/nginx/hi 目录中。它的用途是为用户创建“起步”代码模板。运行它可以使用三个选项，依次是：

- 工程名，可选，默认 demo
- 工程类型，可选，支持 cpp，python，lua 和 java，默认 cpp
- hi-nginx 安装路径，可选，默认/home/centos7/nginx

用户可以通过-h 或者--help 参看使用说明。

为了利用 hi-project 提供的便利，最好是在一个新建的空目录中运行它——它会在运行目录下创建一个总的 Makefile，这个文件能够控制所有的子项目。

## 3.2 hello world

hello world 工程包含了 hi-nginx web application 开发的最基本要素。

### 3.2.1 cpp

使用 hi-project 脚本创建一个 cpp 工程，名为 hello：

```
/home/centos7/nginx/hi/hi-project hello cpp /home/centos7/nginx
```

后面两个参数是可省的。这时，hi-project 会创建一个名为hello 的目录，并在该目录中创建两个文件，一个是Makefile，一个是hello.cpp。前者帮助用户在执行 make && make install 时把 web application 编译、安装至正确位置；后者则帮助用户正确创建合乎 hi-nginx 要求的 class：

```
#include "servlet.hpp"


namespace hi {

    class hello : public servlet {
    public:

        void handler(request& req, response& res) {
            res.headers.find("Content-Type")->second = "text/plain;charset=UTF-8";
            res.content = "hello,world";
            res.status = 200;
        }

    };
}
```

```
17
18 extern "C" hi::servlet* create() {
19     return new hi::hello();
20 }
21
22 extern "C" void destroy(hi::servlet* p) {
23     delete p;
24 }
```

以上代码一目了然，无需过多解释，任何熟知 c++ 的程序员都能看懂。没错，hi-nginx 并不要求 cpp 程序员“精通”自己的工具，只需熟知即可。当然，熟知 http 协议是必要的，否则很难正确地使用 request 类和 response 类。如何使用这两个类，下文会详述。在此先按下不表。

用户 make && make install 之后，hi-project 后把编译生成的hello.so 安装在/home/centos7/nginx/hi 目录下。要启用这个 web application，只需在 hi-nginx 的配置文件中加入：

```
1 location = /hello {
2     hi hi/hello.so;
3 }
```

然后，hi-nginx 执行 reload 即可。

### 3.2.2 python

使用 hi-project 脚本创建一个 python 工程，名为 python：

```
1 /home/centos7/nginx/hi/hi-project python python /home/centos7/nginx
```

最后一个参数是可省的。hi-project 会创建一个名为python 的目录，并在该目录中创建两个文件，一个是Makefile，一个是python.py。前者的作用如上。后者很简单：

```
1 hi_res.status(200)
2 hi_res.content('hello,python')
```

用户可以把其他 python 类型的 web 应用全放在python 目录下，它们都能被正确安装。

### 3.2.3 lua

使用 hi-project 脚本创建一个 lua 工程，名为 lua：

```
1 /home/centos7/nginx/hi/hi-project luahello lua /home/centos7/nginx
```

最后一个参数是可省的。hi-project 会创建一个名为lua 的目录，并在该目录中创建两个文件，一个是Makefile，一个是lua.lua。前者的作用如上。后者很简单：

```
1 hi_res:status(200)
2 hi_res:content('hello,lua')
```

用户可以把其他 lua 类型的 web 应用全放在lua 目录下，它们都能被正确安装。

### 3.2.4 java

使用 hi-project 脚本创建一个 jhello 工程，名为 jhello：

```
/home/centos7/nginx/hi/hi-project jhello java /home/centos7/nginx
```

最后一个参数是可省的。hi-project 会创建一个名为jhello 的目录，并在该目录中创建一个是Makefile。还会创建一个名为hi 的子目录，该目录下会有一个名为jhello.java 的源文件。前者的作用如上。后者很简单：

```
package hi;

public class jhello implements hi.servlet {

    public jhello() {

    }

    public void handler(hi.request req, hi.response res) {
        res.status = 200;
        res.content ="hello,world"

    }
}
```

用户可以把其他 java 类型的 web 应用全放在hi 目录下，它们都能被正确安装。

## 3.3 请求类

hi-nginx 把请求分解为以下几个部分：

- 一般信息
  - client
  - user_agent
  - method
  - uri
  - param
- headers 头信息，需要开启 hi_need_headers
- form 表单信息
- cookies 信息，需要开启 hi_need_cookies
- session 会话信息，需要开启 hi_need_session 并配置 hi_redis_host 和 hi_redis_port

用户可以通过配合“开关指令”获取想要的信息。

### 3.3.1 python api

`python` 用户可以通过类实例 `hi_req` 结合以下方法获取想要的信息：

- client
- user_agent
- method
- uri
- param
- has_xxx
- get_xxx

其中 `xxx` 可取值 `header`，`form`，`session` 和 `cookie`。

### 3.3.2 lua api

`lua` 用户可以通过类实例 `hi_req` 结合以下方法获取想要的信息：

- client
- user_agent
- method
- uri
- param
- has_xxx
- get_xxx

其中 `xxx` 可取值 `header`，`form`，`session` 和 `cookie`。

## 3.4 响应类

`hi-nginx` 把响应分解为四个部分：

- status，状态码
- headers，响应头
- content，响应体
- session，会话信息

用户通过以上四个变量操纵 `hi-nginx` 应答请求。

#### 3.4.1 python api

`python` 用户可以通过类实例 `hi_res` 结合以下方法写入想要的响应：

- status
- header
- content
- session

#### 3.4.2 lua api

`lua` 用户可以通过类实例 `hi_res` 结合以下方法写入想要的响应：

- status
- header
- content
- session

### 3.5 指令详解

#### 3.5.1 hi

这个指令负责装载 `cpp application`。它接受一个参数，参数值为动态 `*.so` 的动态链接库。

#### 3.5.2 hi__need__cache

这个指令负责开关缓存管理器。`hi-nginx` 的缓存管理器采用的时间过期和最近最少使用算法相结合的缓存管理策略。用户可以结合以下命令控制缓存管理器

- `hi_cache_size`
- `hi_cache_expires`

#### 3.5.3 hi__need__headers

这个指令负责开关 http 头信息获取。

#### 3.5.4 hi__need__cookies

这个指令负责开关 http cookie 信息获取

### 3.5.5 hi_need_session

这个指令负责开关 http session 会话管理器。hi-nginx 通过 redis 服务器来管理会话数据，因此，用户应该在打开这个开关的同时配置以下两项：

- hi_redis_host
- hi_redis_port

此外，用户还可以通过 hi_session_expires 指令配置会话过期时间。

会话管理器与 hi_need_cookies 是同步打开的，hi-nginx 需要通过 cookie 设置客户端唯一识别 ID，而且限定识别 ID 必须具有 SESSIONID 名。用户可以使用 nginx 内置模块 ngx_http_userid_module 来配置识别 ID，例如：

```
1    userid on;
2    userid_name SESSIONID;
3    userid_domain localhost;
4    userid_path /;
```

### 3.5.6 hi_py_content

这个指令负责运行 python 内容块

### 3.5.7 hi_py_script

这个指令负责运行 python 脚本，它接受一个参数，可指定查找脚本的目录。

### 3.5.8 hi_lua_content

这个指令负责运行 lua 内容块

### 3.5.9 hi_lua_script

这个指令负责运行 lua 脚本，它接受一个参数，可指定查找脚本的目录。

### 3.5.10 hi_java_classpath

这个指令负责为虚拟机设置 CLASSPATH，一般的写法是：

```
1  hi_java_classpath "-Djava.class.path=.:/home/centos7/nginx/java:/home/centos7/nginx/java/hi-nginx-java.jar";
```

hi-nginx-java.jar 是必须添加的。其他 jar 文件可以自行添加。

### 3.5.11 hi_java_servlet_cache_expires

这个指令负责安排 servlet 缓存时间，默认 300 秒，使用它可以周期性的自动更新 servlet class。如果你修改了 servlet，更新安装后可以实现自动更新，而无需 reload。

#### 3.5.12 hi_java_servlet_cache_size

这个指令可指定 servlet 缓存容器大小。

#### 3.5.13 hi_java_version

这个指令可指定虚拟机版本，默认是 8，即 java8。可以指定 6, 即 java6。

#### 3.5.14 hi_java_servlet

这个指令负责调用 servlet 类。比如，

```
1 hi_java_servlet hi/jhello;
```

它表示调用的是 hi-nginx 安装目录下的java/hi/jhello.class。

## 4 第三方模块

hi-nginx 目前集成了一些常用的第三方模块。包括:

- array-var-nginx-module
- form-input-nginx-module
- headers-more-nginx-module
- iconv-nginx-module
- memc-nginx-module
- ngx_coolkit
- ngx_devel_kit
- rds-csv-nginx-module
- rds-json-nginx-module
- redis2-nginx-module
- set-misc-nginx-module
- srcache-nginx-module
- xss-nginx-module
- nginx-http-concat
- nginx-push-stream-module
- nginx-rtmp-module

- nchan
- echo-nginx-module
- nginx-upload-module

它们都放在3rd 目录中，需要时，只需--add-module=3rd/xxx 即可。例如：

```
#!/bin/bash
./configure   --with-http_ssl_module \
              --with-http_v2_module \
              --prefix=/home/centos7/nginx \
              --add-module=ngx_http_hi_module \
              --add-module=3rd/ngx_devel_kit-0.3.0 \
              --add-module=3rd/nginx-http-concat-1.2.2 \
              --add-module=3rd/echo-nginx-module-0.60 \
              --add-module=3rd/array-var-nginx-module-0.05 \
              --add-module=3rd/form-input-nginx-module-0.12 \
              --add-module=3rd/headers-more-nginx-module-0.32 \
              --add-module=3rd/iconv-nginx-module-0.14 \
              --add-module=3rd/memc-nginx-module-0.18 \
              --add-module=3rd/ngx_coolkit-0.2rc3 \
              --add-module=3rd/rds-csv-nginx-module-0.07 \
              --add-module=3rd/rds-json-nginx-module-0.14 \
              --add-module=3rd/redis2-nginx-module-0.14 \
              --add-module=3rd/set-misc-nginx-module-0.31 \
              --add-module=3rd/srcache-nginx-module-0.31 \
              --add-module=3rd/xss-nginx-module-0.05 \
              --add-module=3rd/nginx-push-stream-module-0.5.2 \
              --add-module=3rd/nchan-1.1.6 \
              --add-module=3rd/nginx-rtmp-module-1.1.7.10 \
              --add-module=3rd/nginx-upload-module-2.255
```

如果用户有需要集成的其他模块，最好也放在3rd 目录中，便于统一管理。然后，接着上面的配置脚本，--add-module=3rd/xxx，依样画葫芦即可。

# 5 模块开发

一般而言，nginx 模块开发以 c 语言为基础。但是，对 hi-nginx 而言，c 和 c++ 都是完全支持的。hi-nginx 本身即以 c++ 模块 ngx_http_hi_module 为基础，它可以作为用户使用 c++ 为 hi-nginx 开发自定义模块的范例演示。

实际上，把 ngx_http_hi_module 放在3rd 目录中也是完全可以的。

# 6 演示代码

hi-nginx 配有较为完整的演示代码。请参考https://github.com/webcpp/hi_demo

# 7 分支版本

hi-nginx 有两个分支版本:

- hi-tengine ，https://github.com/webcpp/hi-tengine
- hi-openresty，https://github.com/webcpp/hi-openresty

这两个版本分别建立在 tengine 和 openresty 之上，用法与 hi-nginx 完全相同。[1]

[1]这两个分支已经没有必要，请不要使用!