OpenResty a fast web app server by extending nginx

UsingLuaRocks

YichunZhang, 25 January 2014 (created 6 August 2011)

permalink

no tags

This sample demonstrates usage of LuaRocks with **OpenResty**. It's been tested on Linux and Mac OS X, with the standard Lua interpreter or with **LuaJIT**.

LuaRocks is a deployment and management system for Lua modules. LuaRocks allows one to install Lua modules as self-contained packages called "rocks", which also contain version dependency information.

We assume that you have installed **OpenResty** into the default location, i.e., `/usr/local/openresty`. You can adjust the paths in this sample according to the actual installation prefix of your **OpenResty** installation. If you haven't installed OpenResty yet, check out the **Download** and **Installation** pages.

Install LuaRocks

First of all, let's install LuaRocks:

Download the LuaRocks tarball from *http://www.luarocks.org/en/Download*. As of this writing, the latest version is `2.1.2`, but we'll use `2.0.13` for compatibility throughout this sample.

```
wget http://luarocks.org/releases/luarocks-2.0.13.tar
tar -xzvf luarocks-2.0.13.tar.gz
cd luarocks-2.0.13/
./configure --prefix=/usr/local/openresty/luajit \
    --with-lua=/usr/local/openresty/luajit/ \
    --lua-suffix=jit-2.1.0-alpha \
    --with-lua-include=/usr/local/openresty/luajit/in
make
sudo make install
```

Install the Lua MD5 library with LuaRocks

In this sample, we'll use the Lua MD5 library to serve as an example, so let's install it with LuaRocks:

```
sudo /usr/local/openresty/luajit/luarocks install md5
```

Configuring our OpenResty application

Let's change the current directory to `/usr/local/openresty/nginx/`:

```
cd /usr/local/openresty/nginx/
```

Next, edit the `conf/nginx.conf` file to the following contents with your favorite text editor (like vim or emacs):

```
worker_processes  1;   # we could enlarge this settin
error_log  logs/error.log warn;

events {
    worker_connections  1024;
}
```

```
http {
    lua_package_path 'conf/?.lua;;';

    server {
        listen       80;
        server_name  localhost;

        location = /luarocks {
            content_by_lua '
                local foo = require("foo")
                foo.say("hello, luarocks!")
            ';
        }
    }
}
```

Finally, create the following two Lua module files `conf/foo.lua`

```
-- conf/foo.lua

module("foo", package.seeall)

local bar = require "bar"

ngx.say("bar loaded")

function say (var)
    bar.say(var)
end
```

and `conf/bar.lua`

```
-- conf/bar.lua
```

```
module("bar", package.seeall)

local rocks = require "luarocks.loader"
local md5 = require "md5"

ngx.say("rocks and md5 loaded")

function say (a)
    ngx.say(md5.sumhexa(a))
end
```

Start the Nginx server

Now we start the Nginx server with our app:

```
ulimit -n1024   # increase the maximal fd count limit
./sbin/nginx
```

If you have already started the Nginx server, then stop it before starting it:

```
./sbin/nginx -s stop
```

Test our app

Now we can test our app via the `curl` utility or any HTTP compliant clients like a web browser:

```
curl http://localhost/luarocks
```

we could get the following outputs at the first run:

```
rocks and md5 loaded
```

```
bar loaded
85e73df5c41378f830c031b81e4453d2
```

then at the second run:

```
85e73df5c41378f830c031b81e4453d2
```

The output changed because **LuaNginxModule** by default caches already loaded Lua modules and those outputing code run at Lua module loading time will no longer be run.

Now let's do some benchmark:

```
ab -c10 -n50000 http://127.0.0.1/luarocks
```

On my Thinkpad T400 laptop (Core2Duo T9600 CPU), it yields

```
Server Software:        ngx_openresty/1.0.4.2rc10
Server Hostname:        localhost
Server Port:            80

Document Path:          /luarocks
Document Length:        33 bytes

Concurrency Level:      10
Time taken for tests:   3.052 seconds
Complete requests:      50000
Failed requests:        0
Write errors:           0
Total transferred:      9400188 bytes
HTML transferred:       1650033 bytes
Requests per second:    16380.48 [#/sec] (mean)
Time per request:       0.610 [ms] (mean)
Time per request:       0.061 [ms] (mean, across all
Transfer rate:          3007.41 [Kbytes/sec] received
```

Note that the throughput is achieved by a single nginx worker process. While doing such benchmark on your own, just be careful about error log level settings in your nginx.conf and not to run out of dynamic port range on your local machine, or it'll be significantly slow after a short of period of time.

Known issues

Pior to **OpenResty** 1.0.4.2rc10, it's known that turning `lua_code_cache` on will cause LuaRocks atop **LuaNginxModule** to throw out the following exception in `error.log`:

```
lua handler aborted: runtime error: stack overflow
```

If you're using any version of **OpenResty** before 1.0.4.2rc10, please consider upgrading.