What's new in LuaRocks

Hisham Muhammad
h@hisham.hm

Lua Workshop 2014
Moscow, Russia
What is LuaRocks

Package manager...
- like dpkg (apt-get), RPM, etc.
- like RubyGems, Python eggs, npm, CPAN, etc.

...for Lua modules
- written in Lua (.lua files)
- or binary modules (.so/.dll files)
What does it do

- The usual tasks of a language-oriented package manager
  - Install
    - ...and make sure that Lua will find the module
  - Remove
    - ...and make sure things don't blow up
  - Verify dependencies
    - ...when installing and removing
  - Compile
    - ...because Lua modules may be written in Lua or C (or any other language, but typically C)
How does it work

Command-line tools
- **luarocks** and **luarocks-admin**

Packaging rules specification format
- .rockspec files

Package format
- .rock files

Serving packaged modules
- rocks server
Up and running in one slide!

```bash
~$ wget http://lua.org/ftp/lua-5.2.3.tar.gz
~$ tar zxvpf lua-5.2.3.tar.gz
~$ cd lua-5.2.3
~$ make linux; sudo make install; cd ..
~$ wget http://luarocks.org/releases/luarocks-2.2.0.tar.gz
~$ tar zxvpf luarocks-2.2.0.tar.gz
~$ cd luarocks-2.2.0
~$ ./configure; sudo make bootstrap; cd ..
~$ sudo luarocks install luasocket
~$ lua
Lua 5.2.3 Copyright (C) 1994-2013 Lua.org, PUC-Rio
> require "socket"
```
The rokspec format

```plaintext
package = "midialsa";
version = "1.21-0"

source = {
    md5 = "072844348e66c04cee42a5b489784453"
}

description = {
    summary = "Provides access to the ALSA sequencer",
    license = "MIT/X11"
}

dependencies = { "lua >= 5.1" }

external_dependencies = {
    ALSA = { header = "alsa/asoundlib.h", library = "asound" }
}

build = {
    type = "builtin",
    modules = {
        ["midialsa"] = "midialsa.lua",
        ["C-midialsa"] = {
            sources = { "C-midialsa.c" },
            libraries = { "asound" },
            incdirs = { "$(ALSA_INCDIR)" },
            libdirs = { "$(ALSA_LIBDIR)" },
        },
    },
    copy_directories = [ "doc", "test" ]
}
```
A rock contains modules and the rockspec
- May contain binaries or source code

- `package-version-revision.type.rock`
- `luafilesystem-1.5.0-2.src.rock`
- `lpeg-0.10-2.win32-x86.rock`
- `lxsh-0.8.6-1.all.rock`

- It's just a zip file with standard contents:
  - Rockspec and `rock_manifest` at the root
  - Subdirectories: `lua/`, `lib/`, `bin/`...
Rocks server

- A location containing .rock/.rockspec files
  - Directory with rocks and a **manifest** index file
  - Generate: `luarocks-admin make-manifest dirname`
  - A set of rocks installed locally is called a "rocks tree"

```bash
~$ luarocks install luasocket \
   --from=http://example.com/my_repository/

~$ luarocks install luasocket --from=/usr/local/myrocks/

~$ luarocks install ./foo-1.0-1.rockspec

~$ luarocks install http://example.com/foo-1.0-1.rockspec
```
A quick look back at last year's talk

- "LuaRocks - past, present and future"
- Part III - The future
  - Future of the rocks server: curation and scalability
  - LuaDist and Lua for Windows
  - Improving the interplay with distros
  - Further development
    - More extensibility
    - LuaRocks as a library
What has changed this year

- **LuaRocks 2.1.2**
  - `luarocks doc foo`
  - improvements on Windows
  - `rocks_provided` so you can preload dependencies
- **LuaRocks 2.2.0**
  - preliminary support for Lua 5.3
  - `luarocks upload foo-1.0-1.rockspec`
  - new default rocks server!
Documentation for modules: it's important we all want it

Like in many aspects of the Lua world, there are no standards

Something is better than nothing, so I came up with some heuristics

- Is there a doc directory? docs?
- \{index| readme | manual\}{.htm | .html | .md | .txt | ...}
- Use the system browser or print it in stdout
- When in doubt, just list the available docs
rocks_provided

- A table in the configuration file with rocks that are considered "installed" even if they are not in the rocks tree
  - **bit32** in Lua 5.2, **luabitop** in LuaJIT, **utf8** in Lua 5.3...
  - Not sure how to specify **ffi** there
- This *could* evolve into something to be used by distros
  - Let them auto-register Lua modules installed outside of LuaRocks
The big change in the ecosystem

- MoonRocks is now the default rocks server
  - by Leaf Corcoran - http://rocks.moonscript.org
- Anyone can upload rocks
- And host their own server:
  - Rocks you own go into the root manifest, immediately available for everyone

```bash
~$ luarocks install luasocket \n   --from=http://rocks.moonscript.org/manifests/user
```
luarocks upload

- Uploads rocks to MoonRocks
  - Go to MoonRocks, create an account
  - Go to Settings, generate an API key

```
~$ luarocks upload ./foo-1.0-1.rockspec  
   --api-key=i5c02i3slkcrbd2if2sicd2rf289i23ndck2
```

- It packs a .src.rock file and uploads both the .rockspec and .src.rock to MoonRocks
- API key is saved in your home, no need to reenter it every time
Growth of the repository

![Graph showing growth of the repository from March 2013 to September 2014.]

- March 2013
- September 2013
- March 2014
- September 2014
Limitations of the rockspec format are well-known:

- "builtin" build mode compiles only C89, can't pass custom compiler flags
- Can't use platform-specific detection for dependencies (pkg-config, etc.)
- No separation between build/runtime dependencies
- etc.

Instead of a big redesign and another freeze, let's make it really extensible.
LuaRocks add-ons

- The plan:
  - New kind of dependency that loads LuaRocks add-ons
  - Add-ons may add entries to the rockspec typechecker (new tables, new fields)
  - Hooks in build/install steps for add-ons to run
    - Possibilities: tests, generate docs, etc.
  - Let users guide development
Rough proposal

- Namespace: `luarocks.addon.youraddon`
  - Using: `using={"foo"}` loads `luarocks.addon.foo`
  - Lots of details to decide, but now LuaRocks can upgrade itself so we can evolve the `rockspec_format`
  - The future could look like this:

```plaintext
rockspec_format = "3.0"
using = {
    "build_dependencies", -- adds support for build-only dependencies
    "busted", -- ensures Busted is installed, runs tests
    "ldoc", -- generates docs using LDoc
    "build.ext >= 2.0", -- example build type extending builtin
}
build_dependencies = { "bin2c >= 1.2" }
build = {
    type = "ext",
    modules = { ["foo"] = { language = "c99", sources = "foo.c" } },
}
doc = { --[[ ldoc specific flags ]] }
```
"LuaRocks as a library"

- Embedability
  - Make LuaRocks fully reentrant (remove all global state)
  - This will require a major refactoring
  - Typed Lua is coming for the rescue!
- Let's make LuaRocks extensible and embeddable, like Lua!
Thank you!

http://luarocks.org

Contact:
http://hisham.hm/
h@hisham.hm
@hisham_hm
About these slides

- Feel free to share this presentation and to use parts of it in your own material
- Licensed under the Creative Commons CC BY 4.0:
  - https://creativecommons.org/licenses/by/4.0/