New features in LuaRocks 3

Hisham Muhammad
h@hisham.hm

Lua Workshop 2018
Kaunas, Lithuania
About me

- Lead developer of LuaRocks
- Other Lua-related projects
  - luacov, lua-compat-5.x, Kepler, Titan
- Author of htop, a process viewer for Unix
- Co-author of GoboLinux, an alternative Linux distro
- Core developer at Kong, an open source API gateway (we’re hiring!)
What is LuaRocks

Package manager for Lua modules

- written in Lua (.lua files)
- or binary modules (.so/.dll files)

Compiles, installs, downloads, uploads

- specification format (.rockspec)
- distribution format (.rock)
- optional built-in build tool
luarocks.org: steady growth
In this talk

luarocks test: integrated test support

New rockspec format

luarocks init: project based workflows

Distribution model
luarocks test
Making it easier to run tests of Lua modules

Motivated by:
- people adding dependencies for testing purposes
- the difficulty of running LuaRocks’ own test suite!
luarocks test: how it works

- Like **luarocks build**, it is extensible, supporting multiple backends:
- It ships with two backends by default:
  - `luarocks.test.busted`
  - `luarocks.test.command`
- **test** block in the rockspec:

  ```lua
  test = {
    type = "busted",
    flags = { "-o", "gtest" },
  }
  ```
Using luarocks test

- If the rockspec doesn’t have a `test` section, it tries to autodetect:
  - if it has a `.busted` file, use `busted`
  - if it has a `test.lua` file, use `command`
- You can pass arbitrary arguments:
  `luarocks test -- -o gtest -v --exclude-tags=postgres`
luarocks test is extensible

- If you use another Lua testing tool, you can create your own test backend
  - `test.type = "foo"` will load `luarocks.test.foo`
- You can load your test module (or any other test-only dependencies) using the new `test_dependencies` block
  ```
  test_dependencies = {
    "luacov > 0.1",
    "my_custom_testing_tool",
  }
  ```
New rockspec format
First, a word about compatibility

- Format 1.0 frozen since 1.0 (2008)
- LR3 assumes 1.0 by default
- `rockspec_format = "3.0"
  - unlocks new features!
  - produces a nice error message in LR2
Improvements in builtin build

- Less boilerplate to use the `builtin` build type
  - ~80% of rockspecs in luarocks.org use `builtin`!

- `build.type = "builtin"` by default
- `build.modules` are autodetected if not specified
  - `*.lua`, `*.c`, `src/*/*.lua`...
- same heuristics as `luarocks write_rockspec`
New dependency types

- **build_dependencies**
  - only used if building from source
  - `luarocks build`
  - `luarocks install` if only a rockspec or src.rock file is available

- **test_dependencies**
  - only used when running `luarocks test`
A minimal valid rockspec

rockspec_format = "3.0"
package = "foobar"
version = "1.0-1"
source = {
    url = "https://example.com/foobar-1.0.tar.gz",
}
A real example: LR3 rockspec

```plaintext
rockspec_format = "3.0"
package = "luarocks"
version = "dev-1"
source = {
    url = "git+https://github.com/luarocks/luarocks"
}
description = {
    summary = "A package manager for Lua modules."
    detailed = "[[LuaRocks is a package manager...]]",
    homepage = "http://www.luarocks.org",
    issues_url = "https://github.com/luarocks/luarocks/issues",
    maintainer = "Hisham Muhammad",
    license = "MIT",
}
test_dependencies = { "luacov" }
test = {
    type = "busted",
    platforms = {
        windows = { flags = { "--exclude-tags=ssh,git,unix" } },
        unix = { flags = { "--exclude-tags=ssh,git" } }
    }
}
```
Format 3.0: miscellanea

- build.macosx_deployment_target
- LuaJIT can be specified in dependencies:
  - "luajit >= 2.1"
- source.dir auto-detection is improved
  - A single dir inside a tarball is auto-detected
- New (optional) description fields **labels** and **issues_url**
- git mode fetches submodules by default
- patches can delete files
luarocks init
A new way of working with LR

- Project-based workflow
  - “Everything in the current directory”

- Motivation
  - Lua developers often have multiple environments in their machines
    - Lua versions, library dependencies, etc.
  - Getting in line with modern package managers
Run `luarocks init` to start a project.

- It creates:
  - `.luarocks/config-5.x.lua`
  - `lua_modules`
  - `./lua`
  - `./luarocks`
  - `your_folder_name-dev-1.rockspec`
It’s a regular rocks tree

- Dependencies are shared as usual, no npm-style replication

- lua_modules/
  - share/
    - lua/5.3/
      - foo.lua
  - lib/
    - lua/5.3/
      - bar.so
  - luarocks/rocks-5.3/
    - foo/1.0-1/
      - foo-1.0-1.rockspec
      - rock_manifest
Picking a Lua

- New flags you can use in any `luarocks` command
  - `--lua-version=5.x`
  - `--lua-dir=/path/to/lua/prefix`
- When you use it with `luarocks init`, the `.lua` and `.luarocks` wrappers remember this setting
Distribution model
Distribution model changes

- Windows single binary
  - luarocks.exe
    - Statically built binary, has everything it needs
    - Supports both MSVC (autodetects) and MinGW
  - Simplified configure script for Unix
    - No more luarocks/core/site_config_5.x.lua
More detection at runtime

- Windows architecture (x86 or x86-64)
- Some helper binaries (wget, etc.)
- Lua include dir and library dir
  - Tries all possible names
    - lua51, lua-51, lua5.1, lua-5.1, lua/5.1, ...
  - Based on path of the interpreter
  - Unix build also checks at configure time for better error messages
LuaRocks 3.0.2 is out today!

luarocks init improvements

improved module autodetection

bugfixes in luarocks show

fix upgrade/downgrade when a single rock has clashing module filenames (ssl.lua vs ssl.so)

bugfix for autodetected external dependencies with nonalphabetic characters (libstdc++.so)
Ačiū!

https://luarocks.org
github.com/luarocks/luarocks
gitter.im/luarocks/luarocks
@luarocksorg

https://hisham.hm
h@hisham.hm
@hisham_hm
@hisham_hm@mastodon.social
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/