

Managing Projects with the Haskell Tool Stack

Max Levatich & Stephen A. Edwards

Columbia University

Fall 2025

The Haskell Stack: Cross-Platform Build Tool

You specify a GHC version and which packages (and versions) to use, then can build and test your project (executables and libraries).

<http://www.haskellstack.org>

```
$ stack new my-project
$ cd my-project
$ stack setup
$ stack build
$ stack exec my-project-exe
$ stack run
$ stack install
```

Files generated by *stack new*

my-project/	
.gitignore	Files for git to ignore
LICENSE	E.g., BSD3. Add your name
ChangeLog.md	If you like
README.md	E.g., for github
→ stack.yaml	GHC version, non-standard package details
→ package.yaml	Build instructions: packages, libraries, versions, etc.
my-project.cabal	Generated from package.yaml as necessary
Setup.hs	Part of Cabal build system; boilerplate
app/	Source files for executables
→ Main.hs	Main function for my-project-exe
src/	Source files for libraries
Lib.hs	Sample library file
test/	Unit test files
Spec.hs	Sample test file

YAML Ain't (a) Markup Language (but it's almost JSON)

Single-line comments

key1: value1

key2: # Keys in a group should be distinct

key1: value2 # Value here is a dictionary

key2: 34 # Space-only indentation for grouping

key3:

- list-element # List element here is a string

- list-element # List elements may repeat

key4: [el1, el2] # Alternative syntax for lists

key5:

- item: foo

price: 42

name: "The first name" # Double-quotes forces a string type

- item: bar

price: 17

stack.yaml: Global build configuration

Main thing here is the “resolver”: a combination of GHC version and versions for many (2500+) standard packages.

Use Long-Term Support packages from Stackage:

<https://www.stackage.org>

```
resolver: lts-22.33
```

This is GHC-9.6.6 plus containers-0.6.7, bytestring-0.11.5.3, etc.

See, e.g., <https://www.stackage.org/lts-22.33>

```
packages:
```

```
- .
```

Optional list of directories (this is the default value).

“There’s one package to be built in the current directory” (see *package.yaml*)

stack.yaml optional fields

```
extra-deps:      # Packages outside the resolver
- acme-missiles-0.3
- git: https://github.com/commercialhaskell/stack.git
  commit: e7b331f14bcfffb8367cd58fbfc8b40ec7642100a

require-stack-version: ">=2.5"

extra-include-dirs: # Searched during builds
- /opt/include
- baz/include

extra-lib-dirs:    # Searched during builds
- foo/baz/lib
```

package.yaml: Package-specific build rules

Translated into .cabal files by sparsely-documented *hpack*

<https://github.com/sol/hpack>

```
name:                peng                      # The main name
version:             0.1.0.0
github:             "sedwards-lab/peng"
license:            BSD3
author:             "Stephen A. Edwards"
maintainer:         "sedwards@cs.columbia.edu"
copyright:          "2020 Stephen A. Edwards"
extra-source-files:
- README.md
- ChangeLog.md
description:         Please see the README on GitHub
```

package.yaml: Common, optional directives

In executable, library, tests, or global

```
source-dirs: src    # Directory in which to look for .hs files
```

```
ghc-options:        # A list to pass to GHC while compiling
```

- -Wall
- -threaded

```
dependencies:        # On which libraries to depend
```

- base >= 4.7 && < 5 # In resolver
- acme-missiles # or extra-deps in stack.yaml

```
build-tools:
```

- alex # Scanner generator, for .x files
- happy # Parser generator, for .y files

package.yaml: the library directive

All but the smallest projects will include this

```
library:  
  source-dirs: src      # Consider all the .hs files here  
  
  ghc-options:          # Optional  
  - -Wall  
  
  build-tools:          # Optional  
  - happy
```

package.yaml: executables

executables:

```
my-exe:                # Generates a my-exe executable
  main: Main.hs        # Where to look for main
  source-dirs: app     # Consider all .hs files here
  dependencies:        # Optional
    - peng             # Name of the package (library)

another-exe:           # Optional
  main: Another.hs
  source-dirs: app2    # May want to make it distinct
```

package.yaml: tests

```
tests:
```

```
  basic-test:           # Name of the particular test/executable
    type: exitcode-stdio-1.0 # Interface to the test (default)
    main: test/Basic.hs   # Where to find the main function
    dependencies:        # We typically test the main library
      - peng
```

```
  another-test:
```

```
    type: detailed-1.0 # More complicated than exitcode-stdio-1.0
    main: test/Another.hs
    dependencies:
      - peng
```

```
$ stack test           # Runs all tests
```

```
$ stack test peng:basic-test # Run a single test
```

Approach

Mostly editing *package.yaml* and source files in *src/*

Have *app/Main.hs* include the *main* function, command-line stuff, and calls into the library. Don't put other *.hs* files in *app/*

Tests are set up for unit tests. See the documentation for *cabal* for more information about how to structure tests