

PROGRAMMING IN HASKELL

0



Chapter 6.2 - The Stack Tool

What is Stack?

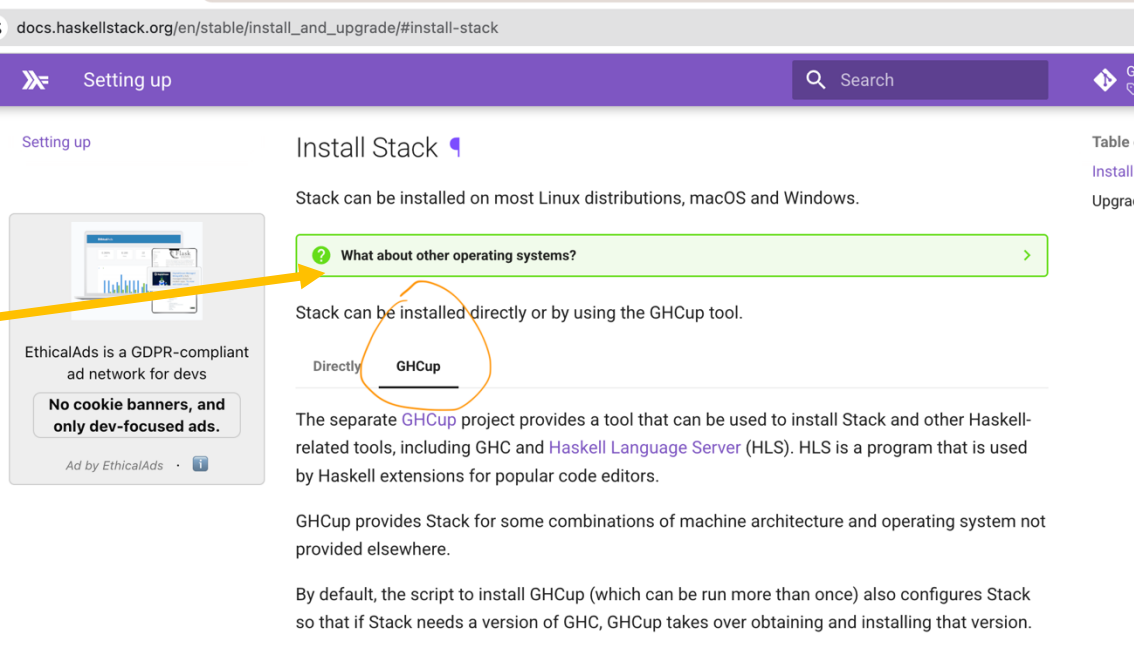
1

- ❑ Its main feature is that it 'sandboxes' the full installation of ghc, dependencies and code in the one isolated location.
- ❑ This means that you can safely run different versions of dependencies in different sandboxes.
- ❑ You can export the project as a fully standalone artefact that does not depend on any local versions of software.

What is Stack? - installation

- You will need to install stack on your machine.
- Clear instructions are available from here and more details are in this topics' labs.

The link is here: [here](#)



The screenshot shows the Haskell Stack installation page at `docs.haskellstack.org/en/stable/install_and_upgrade/#install-stack`. The page has a purple header with a search bar and a 'Setting up' tab. The main content area is titled 'Install Stack' and includes a sub-header 'Stack can be installed on most Linux distributions, macOS and Windows.' Below this, there is a green callout box with a question mark icon and the text 'What about other operating systems?'. A yellow arrow points from the text 'more details are in this topics' labs.' in the list to this callout box. Under the heading 'Stack can be installed directly or by using the GHCup tool.', there are two options: 'Directly' and 'GHCup'. The 'GHCup' option is circled in orange. Below this, the text explains that the separate GHCup project provides a tool to install Stack and other Haskell-related tools, including GHC and Haskell Language Server (HLS). It also mentions that GHCup provides Stack for some combinations of machine architecture and operating system not provided elsewhere, and that by default, the script to install GHCup also configures Stack.

docs.haskellstack.org/en/stable/install_and_upgrade/#install-stack

Setting up

Install Stack

Stack can be installed on most Linux distributions, macOS and Windows.

What about other operating systems?

Stack can be installed directly or by using the GHCup tool.

Directly GHCup

The separate [GHCup](#) project provides a tool that can be used to install Stack and other Haskell-related tools, including GHC and [Haskell Language Server](#) (HLS). HLS is a program that is used by Haskell extensions for popular code editors.

GHCup provides Stack for some combinations of machine architecture and operating system not provided elsewhere.

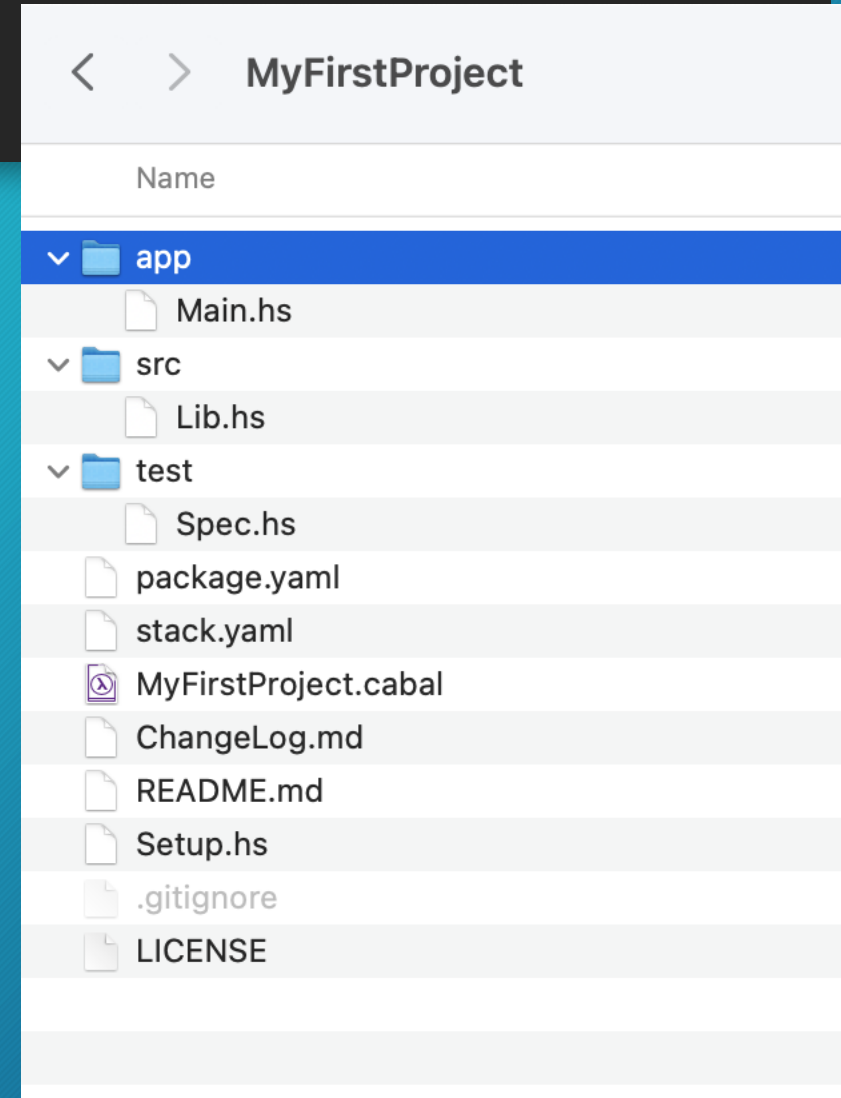
By default, the script to install GHCup (which can be run more than once) also configures Stack so that if Stack needs a version of GHC, GHCup takes over obtaining and installing that version.

What is Stack? - structure

3

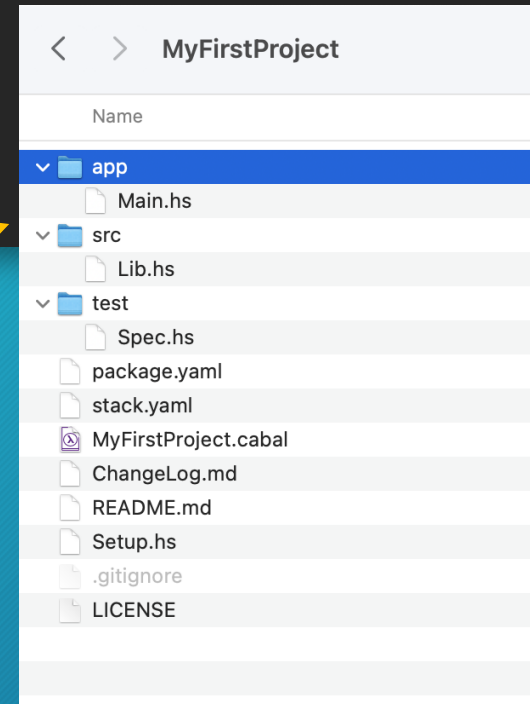
We will use the standard project structure in our projects as provided in the default installation, i.e.

‘stack new MyFirstProject’



Using Stack 1/3

```
$ stack new MyFirstProject
```



Make a small change to
app/Main.hs

```
dev > stack > MyFirstProject > app > Main.hs
1  module Main where
2
3  import Lib
4
5  main :: IO ()
6  main = putStrLn("hello world")
7
```

Using Stack 2/3


5

\$ stack build – builds/rebuilds project with updated code

\$ stack install – copies executable into common location

\$ MyFirstProject-exe – runs executable

```
dev > stack > MyFirstProject > app >  Main.hs
1  module Main where
2
3  import Lib
4
5  main :: IO ()
6  main = putStrLn("hello world")
7
```



```
(base) $MyFirstProject-exe
hello world
(base) $
```

Using Stack 3/3

6

Updating project

```
dev > stack > MyFirstProject > app > ⌘ Main.hs
1  module Main where
2
3  import Lib
4
5  main :: IO ()
6  main = putStrLn("hello world again")
7
```

You may need to add this common location to your system path

\$ stack build –rebuilds project with updated code

\$ stack install – copies updated executable into common location

\$ MyFirstProject-exe – runs updated executable

```
[(base) $MyFirstProject-exe
hello world again
(base) $
```


Where is everything in Stack structure?

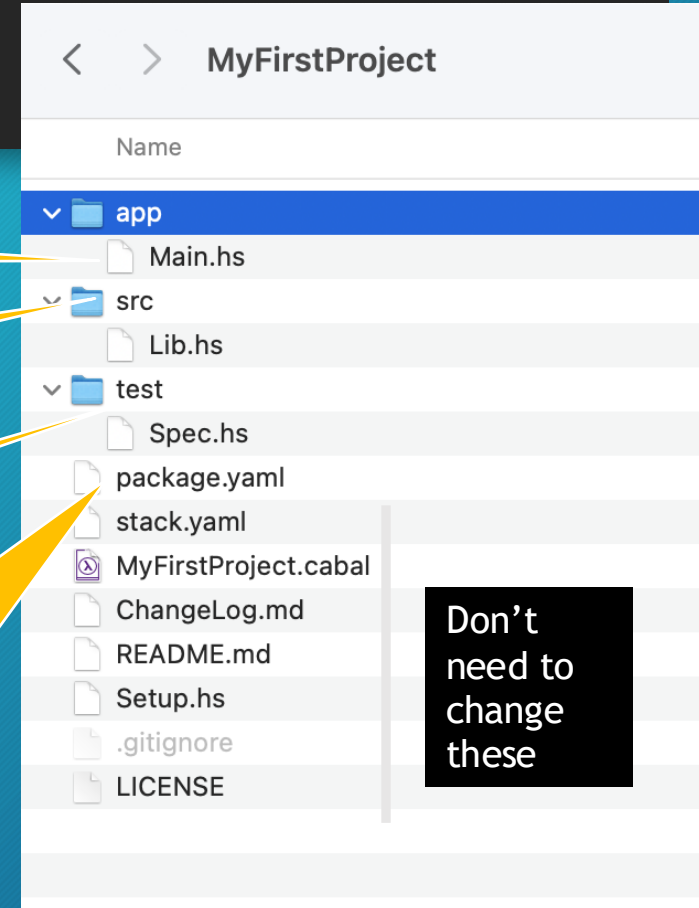
7

app - e.g Driver code - we talk to world here

src - e.g. Library code
- functions here
(‘heavy lifting’)

Testing section (we don’t use)

Where we define dependencies - allows us to sandbox



properties.yaml - contents

8

```
1  name:      MyFirstProject
2  version:   0.1.0.0
3  github:    "githubuser/MyFirstProject"
4  license:   BSD3
5  author:    "Author name here"
6  maintainer: "example@example.com"
7  copyright: "2020 Author name here"
8
9  extra-source-files:
10 - README.md
11 - ChangeLog.md
12
13 description: Please see the README at github.com/githubuser/MyFirstProject#readme
14
15 dependencies:
16 - base >= 4.7 && < 5
17
18 library:
19   source-dirs: src
20
21 executables:
22   MyFirstProject-exe:
23     main:      Main.hs
24     source-dirs: app
25     ghc-options:
26       - -threaded
27       - -rtsopts
28       - -with-rtsopts=-N
29     dependencies:
30       - MyFirstProject
31
32 tests:
33   MyFirstProject-test:
34     main:      Spec.hs
35     source-dirs: test
36     ghc-options:
37       - -threaded
38       - -rtsopts
39       - -with-rtsopts=-N
40     dependencies:
41       - MyFirstProject
42
```

the versions of
ghci base that are
allowed

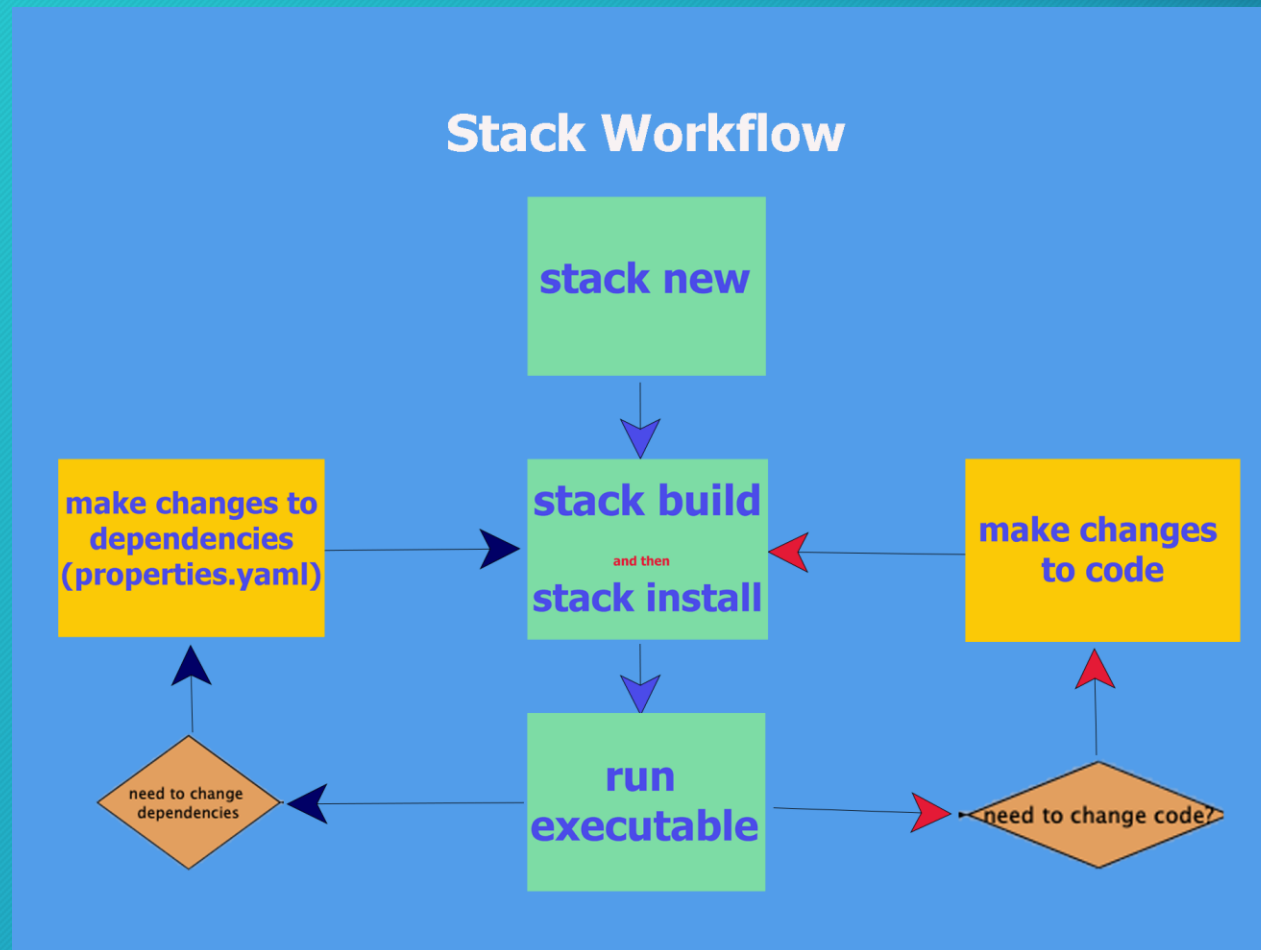
where the library
code is kept

this is where the
Driver code is kept.
This is written in
"main.hs" for now

this is where the
tests are placed.

Workflow of Stack

9



Updating package.yaml 1/2

10

We need to update *package.yaml* for two reasons:

1. When we are structuring our code as seen in the default stack structure, we may write many functions, then curate these related functions into (files in) folders of related functions,

For example, if we had a lot of sorting functions, we might put all such code in (files in) a *Sort* folder.

To use this code we need to tell package where code is situated.

2. Writing Haskell programs will involve using standard Haskell packages. We update package.yaml to include any packages we need (these are the dependencies)

Stack helps us to manage these dependencies by:

- Downloading a particular version (as defined in package.yaml) of a package
- Once the dependency is mentioned in package.yaml, Stack takes care of the rest. (downloads, installs etc)
- This project will use this snapshot of the package from now on so once the program works once (with this version of the package), it will not need to be updated because of those packages being changed.

- During the labs, you will see how to use an outside package ('split') and create our own library code.
- We will see how to make 'split' available to our code
- We will see how to structure our code to be used in the 'driver'/'app' code
- We will rewrite the app code (from the default code) to use our library code

