Installing Go from Source

This lesson serves as a guide to show how the installation of Go from compilable source-code can be processed and verified.

WE'LL COVER THE FOLLOWING

It is instructive to download and compile the full source-code of Go yourself. Complete and up to date instructions for that process are available here. We'll provide an overview of the different steps, providing additional info and references as needed. Because the Go toolchain is written in Go, you need a (previous) Go compiler installed to build it, as explained in the last lesson. Let's get an overview of the basic steps.

Step 1: Install git if needed

To download the Go source code or to use the **go get** command, you need Git. Check by typing **git** on a command-line. If it is not installed, download git from this link.

Step 2: Install a compiler

It is only needed if your Go packages call C code. Use the standard installation methods for your system. Also, see this page for more details.

Step 3: Fetch the Go repository

Step 3: Fetch the Go repository

Go will install its source code in a folder named *go*. Change to the directory that will be its parent and make sure go directory does not exist. Then, check out the repository with the command:

```
$ git clone https://go.googlesource.com/go
$ cd go
$ git checkout go1.11.5
```

Change the version number in the last line to the version you want to install. To play with the latest changes, you want to use the master branch:

```
git checkout master
```

Step 4: Build the source code

To build the source code (in Linux/Mac OS) type the following command:

```
cd go/src
./all.bash
```

But, for Windows the last line is:

```
./all.bat
```

The building and testing takes some time (in the order of minutes), and when successful, the following text appears on Linux/Mac OS:

```
ALL TESTS PASSED
---
Installed Go for linux/amd64 in /home/user/go.
Installed commands in /home/user/go/bin.
*** You need to add /home/user/go/bin to your $PATH. ***
```

On Windows, the following text appears:

```
ALL TESTS PASSED
---
Installed Go for windows/amd64 in c:\Go
Installed commands in c:\go\bin
```

Step 5: Get additional Go tools

First, define the GOPATH variable, as explained in the next lesson. Then, issue the following command:

```
go get golang.org/x/tools/cmd/...
```

Step 6: Verifying the release of installation

Issue the following command:

```
go version
```

This will result (in Linux/Mac os) for example in the output:

```
go version go1.11.5 linux/amd64
```

Or, on Windows:

```
go version go1.11.5 windows/amd64
```

From within Go-code, the current version can be obtained with the function `Version` from the package `runtime`.

```
package main
import (
    "fmt"
    "runtime"
)
func main() {
    // Printing Version number of Go on your machine
    fmt.Printf("%s", runtime.Version())
}
```

Go Version

Our platform has version 1.6.2. Thus, the output for the above executable is **go1.6.2**.

Step 7: Update to a newer release

For example, if you want to update to *version 1.11.5*, following code will do the work:

```
cd go/src
git fetch
git checkout go1.11.5
```

Then, you have to repeat Step 4.

To release notes for different versions, view this page. The gofix tool can be used to update the Go source-code (written in an older version) to the latest release.

Now that you know how to install Go and check for newer releases let's see how to create a Go environment.