Red Hat Developer Tools 1

Using Go 1.14.7 Toolset

Installing and using Go 1.14.7 Toolset

Last Updated: 2020-12-01

Red Hat Developer Tools 1 Using Go 1.14.7 Toolset

Installing and using Go 1.14.7 Toolset

Eva-Lotte Gebhardt
egebhard@redhat.com

Zuzana Zoubkova
zzoubkov@redhat.com

Olga Tikhomirova
otikhomi@redhat.com

Peter Macko

Kevin Owen

Vladimir Slavik

Legal Notice

Abstract

Go is a Red Hat offering for developers on the Red Hat Enterprise Linux platform. The Using Go provides an overview of this product, explains how to invoke and use the Go versions of the tools, and links to resources with more in-depth information.

Table of Contents

CHAPTER 1. GO

1.1. ABOUT GO TOOLSET

Go Toolset is a Red Hat offering for developers on the Red Hat Enterprise Linux platform. It provides the Go programming language tools and libraries. Go is alternatively known as **golang**.

Go Toolset is distributed as a part of Red Hat Developer Tools for Red Hat Enterprise Linux 7. Go Toolset is available as a module for Red Hat Enterprise Linux 8.

The following components are available as a part of Go Toolset:

Table 1.1. Go Toolset components

Name	Version	Description
golang	RHEL 7 – 1.14.9 RHEL 8 – 1.14.7	A Go compiler.
delve	1.4.1	A Go debugger.

1.2. COMPATIBILITY

Go Toolset is available for Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 8 on the following architectures:

- The 64-bit Intel and AMD architectures
- The IBM Power Systems architecture [1]
- The 64-bit ARM architecture [2]
- The little-endian variant of IBM Power Systems architecture
- The IBM Z Systems architecture

1.3. GETTING ACCESS TO GO TOOLSET ON RED HAT ENTERPRISE LINUX 7

This chapter lists the steps to perform before installing Go Toolset on a Red Hat Enterprise Linux 7 system. Complete the following steps to attach a subscription that provides access to the repository for Red Hat Developer Tools, and then enable the Red Hat Developer Tools and Red Hat Software Collections repositories.

Prerequisites

- Verify that **wget** is installed on your system. The tool is available from the default Red Hat Enterprise Linux repositories. To install it, run the following command as root:

```
# yum install wget
```

Procedure

1. Get the latest subscription data from the server:

```
# subscription-manager refresh
```

2. Use the following command to register the system:

```
# subscription-manager register
```

You can also register the system by following the appropriate steps in Registering and Unregistering a System in the Red Hat Subscription Management document.

3. Display a list of all subscriptions that are available for your system and identify the pool ID for the subscription:

```
# subscription-manager list --available
```

This command displays the subscription name, unique identifier, expiration date, and other details related to it. The pool ID is listed on a line beginning with **Pool ID**.

4. Attach the subscription that provides access to the **Red Hat Developer Tools** repository. Use the pool ID you identified in the previous step.

```
# subscription-manager attach --pool=<appropriate pool ID from the subscription>
```

5. Verify the list of subscriptions attached to your system:

```
# sudo subscription-manager list --consumed
```

6. Enable the **rhel-7-*variant*-devtools-rpms** repository:

```
# subscription-manager repos --enable rhel-7-variant-devtools-rpms
```

Replace *variant* with the Red Hat Enterprise Linux system variant (**server** or **workstation**).

Consider using Red Hat Enterprise Linux Server to access the widest range of the development tools.

7. Enable the **rhel-*variant*-rhscl-7-rpms** repository:

```
# subscription-manager repos --enable rhel-variant-rhscl-7-rpms
```

Replace *variant* with the Red Hat Enterprise Linux system variant (**server** or **workstation**).

8. Add the Red Hat Developer Tools key to your system:

```
# cd /etc/pki/rpm-gpg
# wget -O RPM-GPG-KEY-redhat-devel https://www.redhat.com/security/data/a5787476.txt
# rpm --import RPM-GPG-KEY-redhat-devel
```

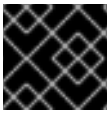
Once the subscription is attached to the system and the repositories are enabled, install Go Toolset as described in Section 1.4, “Installing Go Toolset”.

Additional Resources

- For more information on how to register your system using Red Hat Subscription Management and associate it with subscriptions, see the Red Hat Subscription Management collection of guides.

1.4. INSTALLING GO TOOLSET

Go Toolset is distributed as a collection of RPM packages that can be installed, updated, uninstalled, and inspected by using the standard package management tools that are included in Red Hat Enterprise Linux. Note that a valid subscription that provides access to the Red Hat Developer Tools content set is required to install Go Toolset on a Red Hat Enterprise Linux 7 system. For detailed instructions on how to associate your Red Hat Enterprise Linux 7 system with an appropriate subscription and get access to Go Toolset, see Section 1.3, "Getting access to Go Toolset on Red Hat Enterprise Linux 7".



IMPORTANT

Before installing Go Toolset, install all available Red Hat Enterprise Linux updates.

1. Install all the components included in Go Toolset for your operating system:
   - On Red Hat Enterprise Linux 7, install the **go-toolset-1.14.9** collection:

     ```
     # yum install go-toolset-1.14
     ```

   - On Red Hat Enterprise Linux 8, install the **go-toolset** module:

     ```
     # yum module install go-toolset
     ```

     This installs all development and debugging tools, and other dependent packages to the system.

2. Choose a Go language workspace directory and save its location as an environment variable to the **$HOME/.bashrc** file:

   ```
   $ mkdir -p workspace_dir
   $ echo 'export GOPATH=workspace_dir' >> $HOME/.bashrc
   $ source $HOME/.bashrc
   ```

   Select an appropriate value for the ***workspace_dir*** directory. A common choice is **$HOME/go**.

   Use the **-p** flag with the **mkdir** command to create the ***workspace_dir*** directory along with the directories that lead to it.

   If the **GOPATH** variable is not set, the **go** compiler uses the **~/go** directory.

Additional resources

- Workspaces – Description of the Go language workspace organization. Official documentation for the Go programming language.

Installable documentation

To install the Go Toolset installable documentation, complete the following steps.

- On Red Hat Enterprise Linux 7, install the **go-toolset-1.14-golang-docs** package:

```
# yum install go-toolset-1.14-golang-docs
```

The installable documentation will be installed to **/opt/rh/go-toolset-1.14/root/usr/lib/go-toolset-1.14-golang/doc/docs.html**.

- On Red Hat Enterprise Linux 8, install the **golang-docs** package:

```
# yum install golang-docs
```

The installable documentation will be installed to **/usr/lib/golang/doc/effective_go.html**.

1.5. ADDITIONAL RESOURCES

A detailed description of Go Toolset and all its features is beyond the scope of this book. For more information, see the resources listed below.

Online documentation

- The Go programming language Documentation – Official documentation for the Go programming language, tools, and libraries.

[1] Only available on RHEL 7.

[2] Only available on RHEL 8.

CHAPTER 2. GO

go is a build tool and dependency manager for the Go programming language.

2.1. WRITING GO PROGRAMS

When creating a Go program, developers must follow the rules for Go workspace layout. The **.go** source files must be placed in the subdirectory of **$GOPATH/src**.

Example 2.1. Creating a Go program

Consider a program named **hello** consisting of a single source file named **hello.go**:

```
$ mkdir -p $GOPATH/src/hello
$ cd $GOPATH/src/hello
$ touch hello.go
```

Edit the file **hello.go**, in a text editor of your choice, to add the following text:

```
package main

import (
    "fmt"
    "net/http"
)

func Welcome(w http.ResponseWriter, req *http.Request) {

    fmt.Fprintf(w, "<h1>Welcome to Go Toolset.</h1>")

}

func main() {

    fmt.Println("Hello.")
    fmt.Println("Starting http server.")
    // Register handler function
    http.HandleFunc("/welcome", Welcome)
    fmt.Println("Go to localhost:8080/welcome")
    fmt.Println("To terminate press CTRL+C.")
    // Start server
    http.ListenAndServe(":8080", nil)

}
```

Additional resources

- Workspaces – Description of the Go language workspace organization. Official documentation for the Go programming language.

2.2. USING THE GO COMPILER

To build a Go program using the command line, change to the project directory and run the **go** compiler as follows:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go build -o output_file go_main_package'
```

- For Red Hat Enterprise Linux 8:

```
$ go build -o output_file go_main_package
```

This creates a binary file named ***output_file*** in the current working directory. If the **-o** option is omitted, the compiler creates a file named after the *go_main_package*, ***go_main_package***.

If *go_main_package* is not a main package or if multiple projects or ***.go** files are specified, the resulting binaries are discarded. In that case, the **go build** command is used to verify that the supplied projects or files can be built.

NOTE

You can execute any command using the **scl** utility on Red Hat Enterprise Linux 7, causing it to be run with the Go binaries available. To use Go Toolset on Red Hat Enterprise Linux 7 without a need to use **scl enable** with every command, run a shell session with:

```
$ scl enable go-toolset-1.14 'bash'
```

Example 2.2. Compiling a Go program using the command line

After you have successfully created the program **hello** as shown in Example 2.1, "Creating a Go program", compile the program:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go build hello.go'
```

- For Red Hat Enterprise Linux 8:

```
$ go build hello.go
```

This creates a new binary file called **hello** in the current working directory.

2.3. RUNNING A GO PROGRAM

When **go** compiles a program, it creates an executable binary file. To run this program on the command line, change to the directory with the executable file and run the program:

```
$ ./file_name
```

Example 2.3. Running a Go program on the command line

After you have successfully compiled the **hello** binary file as shown in Example 2.2, "Compiling a Go program using the command line", run it by typing the following at a shell prompt:

```
$ ./hello
Hello.
Starting http server.
Go to localhost:8080/welcome
To terminate press CTRL+C.
```

2.4. INSTALLING COMPILED GO PROJECTS

Installing a Go project means that its executable files and libraries are compiled and copied to appropriate directories in the Go workspace. The **go** tool can then use the executable files and libraries in further projects. Dependencies of the installed project are installed, too.

To install a Go project, run the **go** tool:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go install go_project'
```

- For Red Hat Enterprise Linux 8:

```
$ go install go_project
```

The **install** command accepts the same options as the **build** command.

2.5. DOWNLOADING GO PROJECTS

To download a 3rd party Go project from an online source and install it, run the **go** tool:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go get 3rd_party_go_project'
```

- For Red Hat Enterprise Linux 8:

```
$ go get 3rd_party_go_project
```

For more details about the possible values of the *3rd_party_go_project* option, run the following command:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go help importpath'
```

- For Red Hat Enterprise Linux 8:

```
$ go help importpath
```

2.6. ADDITIONAL RESOURCES

A detailed description of the **go** compiler and its features is beyond the scope of this book. For more information, see the resources listed below.

Installed documentation

- The Go compiler **help** command provides information on its usage. To show the help index:
  - For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go help'
```

  - For Red Hat Enterprise Linux 8:

```
$ go help
```

- The Go compiler **doc** command shows documentation for Go packages. To show documentation for package *package_name*:
  - For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go doc package_name'
```

    To learn more about the **doc** command:

```
$ scl enable go-toolset-1.14 'go help doc'
```

  - For Red Hat Enterprise Linux 8:

```
$ go doc package_name
```

    To learn more about the **doc** command:

```
$ go help doc
```

Online documentation

- Command go – Official documentation of the **go** compiler.

See also

- Chapter 1, *Go* – An overview of Go and more information on how to install it on your system.

CHAPTER 3. GOFMT

gofmt is a code formatting tool for the Go programming language, packaged together with the **go** compiler.

3.1. FORMATTING CODE

To format all code in the *code_path* path, run the **gofmt** tool as follows:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'gofmt -w code_path'
```

- For Red Hat Enterprise Linux 8:

```
$ gofmt -w code_path
```

This command directly changes the code in the ***code_path*** path. When ***code_path*** is a single file, the changes apply only to the file. When ***code_path*** is a directory, all **.go** files in the directory are processed.

When the *code_path* is omitted, **gofmt** reads standard input instead.

To print the formatted code to standard output instead of writing to the original file, omit the **-w** option.

3.2. PREVIEWING CHANGES TO CODE

To preview changes done by formatting code in a given path ***code_path***, run the **gofmt** tool with the **-d** option as follows:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'gofmt -d code_path'
```

- For Red Hat Enterprise Linux 8:

```
$ gofmt -d code_path
```

The output in unified diff format is printed to standard output.

It is possible to combine both the **-d** and **-w** options.

3.3. SIMPLIFYING CODE

To simplify code in a given path ***code_path***, run the **gofmt** tool with the **-s** option as follows:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'gofmt -s code_path'
```

- For Red Hat Enterprise Linux 8:

```
$ gofmt -s code_path
```

The code under ***code_path*** is simplified. Use the **-d** option to show the differences, and use the **-w** option to apply the changes to the code.

3.4. REFACTORING CODE

The **gofmt** tool can be used to refactor code by applying arbitrary substitutions. To refactor code in a given path ***code_path*** according to a rule *rewrite_rule*, run the **gofmt** tool with the **-r** option as follows:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'gofmt -r rewrite_rule code_path'
```

- For Red Hat Enterprise Linux 8:

```
$ gofmt -r rewrite_rule code_path
```

The code under ***code_path*** is refactored according to the rule *rewrite_rule*. Use the **-d** option to show the differences, and use the **-w** option to apply the changes to the code. The additional options must be placed after the rule *rewrite_rule*:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'gofmt -r rewrite_rule -d code_path'
```

- For Red Hat Enterprise Linux 8:

```
$ gofmt -r rewrite_rule -d code_path
```

Detailed description of the rewrite rules is beyond the scope of this book. For more information, see the resources listed in Section 3.5, “Additional resources”.

3.5. ADDITIONAL RESOURCES

A detailed description of the **gofmt** tool and its features is beyond the scope of this book. For more information, see the resources listed below.

Online documentation

- Command gofmt – Official documentation of the **gofmt** tool.

See also

- Chapter 1, Go – An overview of Go Toolset and more information on how to install it on your system.

CHAPTER 4. GO RACE DETECTOR

Go Toolset includes the Go race detector, which is a feature of the Go standard library.

4.1. INSTALLING THE RACE DETECTOR

On Red Hat Enterprise Linux 7, the race detector is provided by the **go-toolset-1.14.9-golang-race** package:

```
# yum install go-toolset-1.14-golang-race
```

On Red Hat Enterprise Linux 8, the race detector is provided by the **golang-race** package:

```
# yum install golang-race
```

This command installs a variant of the Go standard library that contains runtime race detection.

4.2. USING THE RACE DETECTOR

To use the runtime race detector in a Go project, add the **-race** option to the **go** tool commands used when manipulating the project.

For a minimal approach to using the race detector, build the project with the **-race** option:

- For Red Hat Enterprise Linux 7:

```
$ scl enable go-toolset-1.14 'go build -race -o output_file go_main_package'
```

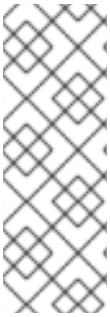
- For Red Hat Enterprise Linux 8:

```
$ go build -race -o output_file go_main_package
```

Run the resulting executable binary file, and the race detector prints warnings to the standard output when a race is detected.

IMPORTANT

The race detector has a significant runtime resource overhead.



NOTE

You can execute any command using the **scl** utility on Red Hat Enterprise Linux 7, causing it to be run with the Go binaries available. To use Go Toolset on Red Hat Enterprise Linux 7 without a need to use **scl enable** with every command, run a shell session with:

```
$ scl enable go-toolset-1.14 'bash'
```

4.3. ADDITIONAL RESOURCES

A detailed description of the Go race detector and its features is beyond the scope of this book. For more information, see the resources listed below.

Online documentation

- Data Race Detector – Official documentation of the Go race detector.

See also

- Chapter 1, Go – An overview of Go and more information on how to install it on your system.

CHAPTER 5. CONTAINER IMAGES WITH GO TOOLSET

Go Toolset is available as container images for Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 8. Container images can be downloaded from the Red Hat Container Registry.

5.1. IMAGES CONTENTS

The Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 8 container images provide content corresponding to the following packages:

Component	Version	Package
Go	1.14	go-toolset-1.14.z

5.2. ACCESSING THE IMAGES

To pull the required image, run the following command as **root**:

- For the Red Hat Enterprise Linux 7 container images:

```
# podman pull registry.redhat.io/devtools/go-toolset-rhel7
```

```
# podman pull registry.redhat.io/ubi7/go-toolset
```

NOTE

The two Red Hat Enterprise Linux 7 container images provide the same content.

- For the Red Hat Enterprise Linux 8 container images:

```
# podman pull registry.redhat.io/rhel8/go-toolset
```

```
# podman pull registry.redhat.io/ubi8/go-toolset
```

NOTE

The two Red Hat Enterprise Linux 8 container images provide the same content.

5.3. USING AS BUILDER IMAGES WITH SOURCE-TO-IMAGE

The Go Toolset container image is prepared for use as a Source-to-Image (S2I) builder image in Red Hat Enterprise Linux 7. Source-to-Image is not supported on Red Hat Enterprise Linux 8.

To do so, set the following build environment variables:

IMPORT_URL

Set this variable to a URL specifying the location of the code. The rules for the **go get** command option apply.

INSTALL_URL

Set this variable to a URL specifying the location of the package that will provide the application's main executable file when built. The rules for the **go install** command option apply.
This variable can be omitted if the main package location is identical with the location specified by the **IMPORT_URL** variable.

Example 5.1. Building a Go application image using Source-to-Image

To build the **md2man** package from its GitHub repository:

```
$ s2i build -e IMPORT_URL='github.com/cpuguy83/go-md2man' -e
INSTALL_URL='github.com/cpuguy83/go-md2man' git://github.com/cpuguy83/go-md2man
registry.access.redhat.com/devtools/go-toolset-rhel7 md2man-app
```

A locally available application image **md2man-app** is built from the repository on GitHub using the **Go** Toolset container image.

To fully leverage the Go as a S2I builder image, build custom images based on it, with modified S2I assemble scripts and further modifications to accomodate the particular application being built.

A detailed description of the Go usage with Source-to-Image is beyond the scope of this document. For more information about Source-to-Image, see:

- *OpenShift Container Platform 4.5 Image Creation Guide* , OpenShift Container Platform-specific guidelines
- *Using Red Hat Software Collections Container Images* , Chapter 2. Using Source-to-Image (S2I).

5.4. ADDITIONAL RESOURCES

- Go Toolset Container Images - entries in the Red Hat Container Registry
- Using Red Hat Software Collections Container Images

CHAPTER 6. CHANGES IN GO TOOLSET

Go Toolset has been updated from version **1.13** to **1.14.9**. Notable changes include:

- Go module system is now fully supported.
- SSL version 3.0 (SSLv3) is no longer supported. Notable Delve debugger enhancements include:
- The new command **examinemem** (or **x**) for examining raw memory
- The new command **display** for printing values of an expression during each stop of the program
- The new **--tty** flag for supplying a Teletypewriter (TTY) for the debugged program
- New coredump support for Arm64
- The new ability to print goroutine labels
- The release of the Debug Adapter Protocol (DAP) server
- Improved output from **dlv trace** and **trace** REPL (read-eval-print-loop) commands

For more information on Go Toolset, see the upstream Go 1.14.9 Release Notes.

For more information on Delve, see the upstream Delve documentation.