



Red Hat Developer Tools 1

Using Go 1.21.0 Toolset

Installing and using Go 1.21.0 Toolset

Red Hat Developer Tools 1 Using Go 1.21.0 Toolset

Installing and using Go 1.21.0 Toolset

Jacob Valdez

jvaldez@redhat.com

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Go Toolset is a Red Hat offering for developers on the Red Hat Enterprise Linux (RHEL) operating system. Use this guide for an overview of Go Toolset, to learn how to invoke and use different versions of Go tools, and to find resources with more in-depth information.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	3
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	4
CHAPTER 1. GO TOOLSET	5
1.1. GO TOOLSET COMPONENTS	5
1.2. GO TOOLSET COMPATIBILITY	5
1.3. INSTALLING GO TOOLSET	5
1.4. INSTALLING GO DOCUMENTATION	5
1.5. ADDITIONAL RESOURCES	6
CHAPTER 2. THE GO COMPILER	7
2.1. PREREQUISITES	7
2.2. SETTING UP A GO WORKSPACE	7
2.3. COMPILING A GO PROGRAM	7
2.4. RUNNING A GO PROGRAM	8
2.5. INSTALLING COMPILED GO PROJECTS	8
2.6. DOWNLOADING AND INSTALLING GO PROJECTS	8
2.7. ADDITIONAL RESOURCES	9
CHAPTER 3. THE GOFMT FORMATTING TOOL	11
3.1. PREREQUISITES	11
3.2. FORMATTING CODE	11
3.3. PREVIEWING CHANGES TO CODE	11
3.4. SIMPLIFYING CODE	12
3.5. REFACTORING CODE	12
3.6. ADDITIONAL RESOURCES	13
CHAPTER 4. THE GO RACE DETECTOR	14
4.1. PREREQUISITES	14
4.2. USING THE GO RACE DETECTOR	14
4.3. ADDITIONAL RESOURCES	14
CHAPTER 5. CONTAINER IMAGES WITH GO TOOLSET	15
5.1. RED HAT ENTERPRISE LINUX GO TOOLSET CONTAINER IMAGES CONTENTS	15
5.2. ACCESSING RED HAT ENTERPRISE LINUX CONTAINER IMAGES	15
5.3. ACCESSING THE UBI GO TOOLSET CONTAINER IMAGE ON RHEL 8	15
5.4. ACCESSING GO TOOLSET FROM THE BASE UBI CONTAINER IMAGE ON RHEL 8	16
5.5. ADDITIONAL RESOURCES	16
CHAPTER 6. CHANGES IN GO TOOLSET	17

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting feedback through Jira (account required)

1. Log in to the [Jira](#) website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. GO TOOLSET

Go Toolset is a Red Hat offering for developers on Red Hat Enterprise Linux (RHEL). It provides the Go programming language tools and libraries. Note that Go is alternatively known as golang.

Go Toolset is available as a module for Red Hat Enterprise Linux 8. Go Toolset is available as packages for Red Hat Enterprise Linux 9.

1.1. GO TOOLSET COMPONENTS

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

Name	Version	Description
golang	RHEL 8 – 1.21.0, RHEL 9 – 1.21.0	A Go compiler.
delve	RHEL 8 – 1.21.0, RHEL 9 – 1.21.0	A Go debugger.

1.2. GO TOOLSET COMPATIBILITY

Go Toolset is available for Red Hat Enterprise Linux 8 and Red Hat Enterprise Linux 9 on the following architectures:

- AMD and Intel 64-bit
- 64-bit ARM
- IBM Power Systems, Little Endian
- 64-bit IBM Z

1.3. INSTALLING GO TOOLSET

Complete the following steps to install Go Toolset, including all dependent packages.

Prerequisites

- All available Red Hat Enterprise Linux updates are installed.

Procedure

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

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

On Red Hat Enterprise Linux 9, install the **go-toolset** package by running:

```
# dnf install go-toolset
```

1.4. INSTALLING GO DOCUMENTATION

You can install documentation for the Go programming language on your local system.

Prerequisites

- Go Toolset is installed.
For more information, see [Installing Go Toolset](#).

Procedure

To install the **golang-docs** package, run the following command:

- On Red Hat Enterprise Linux 8:

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

You can find the documentation under the following path: **`/usr/lib/golang/doc/go_spec.html`**.

- On Red Hat Enterprise Linux 9:

```
# dnf install golang-docs
```

You can find the documentation under the following path: **`/usr/lib/golang/doc/go_spec.html`**.

1.5. ADDITIONAL RESOURCES

- For more information on the Go programming language, tools, and libraries, see the [official Go documentation](#).

CHAPTER 2. THE GO COMPILER

The Go compiler is a build tool and dependency manager for the Go programming language. It offers error checking and optimization of your code.

2.1. PREREQUISITES

- Go Toolset is installed.
For more information, see [Installing Go Toolset](#).

2.2. SETTING UP A GO WORKSPACE

To compile a Go program, you need to set up a Go workspace.

Procedure

1. Create a workspace directory as a subdirectory of **\$GOPATH/src**.
A common choice is **\$HOME/go**.
2. Place your source files into your workspace directory.
3. Set the location of your workspace directory as an environment variable to the **\$HOME/.bashrc** file by running:

```
$ echo 'export GOPATH=<workspace_dir>' >> $HOME/.bashrc
$ source $HOME/.bashrc
```

Replace `<workspace_dir>` with the name of your workspace directory.

Additional resources

- The [official Go workspaces documentation](#).

2.3. COMPILING A GO PROGRAM

You can compile your Go program using the Go compiler. The Go compiler creates an executable binary file as a result of compiling.

Prerequisites

- A set up Go workspace with configured modules.
For information on how to set up a workspace, see [Setting up a Go workspace](#).

Procedure

In your project directory, run:

- On Red Hat Enterprise Linux 8:

```
$ go build <output_file> <go_main_package>
```

- Replace `<output_file>` with the desired name of your output file and `<go_main_package>` with the name of your main package.

- On Red Hat Enterprise Linux 9:

```
$ go build <output_file> <go_main_package>
```

- Replace **<output_file>** with the desired name of your output file and **<go_main_package>** with the name of your main package.

2.4. RUNNING A GO PROGRAM

The Go compiler creates an executable binary file as a result of compiling. Complete the following steps to execute this file and run your program.

Prerequisites

- Your program is compiled.
For more information on how to compile your program, see [Compiling a Go program](#).

Procedure

To run your program, run in the directory containing the executable file:

```
$. /<file_name>
```

- Replace **<file_name>** with the name of your executable file.

2.5. INSTALLING COMPILED GO PROJECTS

You can install already compiled Go projects to use their executable files and libraries in further Go projects. After installation, the executable files and libraries of the project are copied to according directories in the Go workspace. Its dependencies are installed as well.

Prerequisites

- A Go workspace with configured modules.
For more information, see [Setting up a Go workspace](#).

Procedure

To install a Go project, run:

- On Red Hat Enterprise Linux 8:

```
$ go install <go_project>
```

- Replace **<go_project>** with the name of the Go project you want to install.

- On Red Hat Enterprise Linux 9:

```
$ go install <go_project>
```

- Replace **<go_project>** with the name of the Go project you want to install.

2.6. DOWNLOADING AND INSTALLING GO PROJECTS

You can download and install third-party Go projects from online resources to use their executable files and libraries in further Go projects. After installation, the executable files and libraries of the project are copied to according directories in the Go workspace. Its dependencies are installed as well.

Prerequisites

- A Go workspace.
For more information, see [Setting up a Go workspace](#).

Procedure

- To download and install a Go project, run:
 - On Red Hat Enterprise Linux 8:

```
$ go install <third_party_go_project>
```

 - Replace **<third_party_go_project>** with the name of the project you want to download.
 - On Red Hat Enterprise Linux 9:

```
$ go install <third_party_go_project>
```

 - Replace **<third_party_go_project>** with the name of the project you want to download.
- For information on possible values of third-party projects, run:
 - On Red Hat Enterprise Linux 8:

```
$ go help importpath
```
 - On Red Hat Enterprise Linux 9:

```
$ go help importpath
```

2.7. ADDITIONAL RESOURCES

- For more information on the Go compiler, see the [official Go documentation](#).
- To display the **help** index included in Go Toolset, run:
 - On Red Hat Enterprise Linux 8:

```
$ go help
```
 - On Red Hat Enterprise Linux 9:

```
$ go help
```
- To display documentation for specific Go packages, run:
 - On Red Hat Enterprise Linux 8:

```
$ go doc <package_name>
```

-
- o On Red Hat Enterprise Linux 9:

```
$ go doc <package_name>
```

See [Go packages](#) for an overview of Go packages.

CHAPTER 3. THE GOFMT FORMATTING TOOL

Instead of a style guide, the Go programming language uses the **gofmt** code formatting tool. **gofmt** automatically formats your code according to the Go layout rules.

3.1. PREREQUISITES

- Go Toolset is installed.
For more information, see [Installing Go Toolset](#).

3.2. FORMATTING CODE

You can use the **gofmt** formatting tool to format code in a given path. When the path leads to a single file, the changes apply only to the file. When the path leads to a directory, all **.go** files in the directory are processed.

Procedure

To format your code in a given path, run:

- On Red Hat Enterprise Linux 8:

```
$ gofmt -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to format.

- On Red Hat Enterprise Linux 9:

```
$ gofmt -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to format.



NOTE

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

3.3. PREVIEWING CHANGES TO CODE

You can use the **gofmt** formatting tool to preview changes done by formatting code in a given path. The output in unified diff format is printed to standard output.

Procedure

To show differences in your code in a given path, run:

- On Red Hat Enterprise Linux 8:

```
$ gofmt -d <code_path>
```

- Replace **<code_path>** with the path to the code you want to compare.

- On Red Hat Enterprise Linux 9:

```
$ gofmt -d <code_path>
```

- Replace **<code_path>** with the path to the code you want to compare.

3.4. SIMPLIFYING CODE

You can use the **gofmt** formatting tool to simplify your code.

Procedure

- To simplify code in a given path, run:

- On Red Hat Enterprise Linux 8:

```
$ gofmt -s -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to simplify.

- On Red Hat Enterprise Linux 9:

```
$ gofmt -s -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to simplify.

- To apply the changes, run:

- On Red Hat Enterprise Linux 8:

```
$ gofmt -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to format.

- On Red Hat Enterprise Linux 9:

```
$ gofmt -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to format.

3.5. REFACTORING CODE

You can use the **gofmt** formatting tool to refactor your code by applying arbitrary substitutions.

Procedure

- To refactor your code in a given path, run:

- On Red Hat Enterprise Linux 8:

```
$ gofmt -r -w <rewrite_rule> <code_path>
```

- Replace **<code_path>** with the path to the code you want to refactor and **<rewrite_rule>** with the rule you want it to be rewritten by.

- On Red Hat Enterprise Linux 9:


```
$ gofmt -r -w <rewrite_rule> <code_path>
```

- Replace **<code_path>** with the path to the code you want to refactor and **<rewrite_rule>** with the rule you want it to be rewritten by.

- To apply the changes, run:

- On Red Hat Enterprise Linux 8:

```
$ gofmt -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to format.

- On Red Hat Enterprise Linux 9:

```
$ gofmt -w <code_path>
```

- Replace **<code_path>** with the path to the code you want to format.

3.6. ADDITIONAL RESOURCES

- The [official gofmt documentation](#).

CHAPTER 4. THE GO RACE DETECTOR

Go Toolset includes the Go race detector, which is a tool of the Go standard library for finding race conditions.

Note that the race detector has a significant runtime resource overhead.

4.1. PREREQUISITES

- Go Toolset is installed.
For more information, see [Installing Go Toolset](#).

4.2. USING THE GO RACE DETECTOR

Use the Go race detector to check your code for race conditions.

Procedure

To use the race detector, run:

- On Red Hat Enterprise Linux 8:

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

- Replace **<output_file>** with the name of your executable file and **<go_main_package>** with the name of the package you want to test.

- On Red Hat Enterprise Linux 9:

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

- Replace **<output_file>** with the name of your executable file and **<go_main_package>** with the name of the package you want to test.

4.3. ADDITIONAL RESOURCES

- The [official Go race detector documentation](#).

CHAPTER 5. CONTAINER IMAGES WITH GO TOOLSET

You can build your own Go Toolset containers from either Red Hat Enterprise Linux container images or Red Hat Universal Base Images (UBI).

5.1. RED HAT ENTERPRISE LINUX GO TOOLSET CONTAINER IMAGES CONTENTS

The Red Hat Enterprise Linux 8 and Red Hat Enterprise Linux 9 container images of Go Toolset contain the following packages:

Component	Version	Package
Go	1.21	RHEL 8 – go-toolset-1.21.0 RHEL 9 – go-toolset-1.21.0

5.2. ACCESSING RED HAT ENTERPRISE LINUX CONTAINER IMAGES

Pull the container image from the Red Hat registry before running your container and performing actions.

Procedure

To pull the required image, run:

- For an image based on Red Hat Enterprise Linux 8:

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

- For an image based on Red Hat Enterprise Linux 9:

```
# podman pull registry.redhat.io/rhel9/go-toolset
```

5.3. ACCESSING THE UBI GO TOOLSET CONTAINER IMAGE ON RHEL 8

On RHEL 8, install the UBI Go Toolset container image to access Go Toolset. Alternatively, you can install Go Toolset to the RHEL 8 base UBI container image. For further information, see [Accessing Go Toolset from the base UBI container image on RHEL 8](#).

Procedure

To pull the UBI Go Toolset container image from the Red Hat registry, run:

- On Red Hat Enterprise Linux 8

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

- On Red Hat Enterprise Linux 9

```
# podman pull registry.access.redhat.com/ubi9/go-toolset
```

5.4. ACCESSING GO TOOLSET FROM THE BASE UBI CONTAINER IMAGE ON RHEL 8

On RHEL 8, Go Toolset packages are part of the Red Hat Universal Base Images (UBIs) repositories, which means you can install Go Toolset as an addition to the base UBI container image. To keep the container image size small, install only individual packages instead of the entire Go Toolset. Alternatively, you can install the UBI Go Toolset container image to access Go Toolset. For further information, see [Accessing the UBI Go Toolset container image on RHEL 8](#).

Prerequisites

- An existing Containerfile.
For information on creating Containerfiles, see the [Dockerfile reference](#) page.

Procedure

- To create a container image containing Go Toolset, add the following lines to your Containerfile:

```
FROM registry.access.redhat.com/ubi8/ubi:latest
```

```
RUN yum module install -y go-toolset
```

- To create a container image containing an individual package only, add the following lines to your Containerfile:

```
RUN yum install -y <package-name>
```

- Replace **<package-name>** with the name of the package you want to install.

5.5. ADDITIONAL RESOURCES

- [Go Toolset Container Images in the Red Hat Container Registry](#) .
- For more information on Red Hat UBI images, see [Working with Container Images](#).
- For more information on Red Hat UBI repositories, see [Universal Base Images \(UBI\): Images, repositories, packages, and source code](#).

CHAPTER 6. CHANGES IN GO TOOLSET

Go Toolset has been updated from version 1.20 to 1.21.0 on RHEL 8 and RHEL 9.

Notable changes include:

- **min**, **max**, and **clear** built-ins have been added.
- Package initialization order is more precisely defined.
- Type inferencing is improved.
- Changes to loop variable semantics.
- Backwards compatibility support is improved.

For detailed information regarding the updates, see the upstream [Go 1.21 Release Notes](#).