Get Set Go

Go programming fundamentals, environment setup, and core concepts

Amrit Pal Singh

www.bpbonline.com

First Edition 2025

ISBN: 978-93-65898-866

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My wife, ***Dilraj****, and my daughter,* ***Jesmyra***

About the Author

Amrit Pal Singh is currently serving as Sr. Director of Cloud Software at Jubilee TV, based in Bengaluru, India. With a career spanning over 20 years, he has extensive experience in various domains. These include high-performance web backend platforms, cloud services deployment, media middleware, and firmware development.

Amrit holds a master's degree in software systems from Birla Institute of Technology and Science, Pilani. He has authored patents in the field of media content management.

In addition to his technical roles, Amrit is an active content creator on YouTube, where he shares insights on technology and software programming. His areas of interest include product development, system software and firmware, web-scale cloud computing system architectures, and ML and AI.

About the Reviewers

- **Shrinath Thube** is a seasoned software developer at IBM, USA, and a technology leader with extensive experience in cloud, cybersecurity, observability, and AI-driven automation. He specializes in building scalable and secure solutions, driving innovations in IBM Cloud, **application performance monitoring** (**APM**), and enterprise security. His expertise spans microservices architecture, governance and compliance, container security, and AI-powered automation.

  With a strong track record of leading mission-critical projects, Shrinath has enhanced system reliability, integrated security frameworks, and optimized cloud-based solutions for improved performance and visibility. He has worked extensively with modern cloud technologies, including Kubernetes, OpenShift, and security, ensuring seamless and secure deployments.

  Beyond technical contributions, Shrinath is passionate about mentoring, cross-functional collaboration, and driving innovation in emerging technologies. As a Technology Advisory Board Member at Avotrix, he provides strategic insights on AI adoption, cloud security, and observability, helping organizations navigate technological advancements and strengthen their security posture. He also serves as an Industry Expert on the Board of Studies for multiple academic institutions, shaping curriculum development and bridging the gap between education and industry needs.

  Shaped by a strong academic foundation, Shrinath holds a master's degree in electrical engineering from San Jose State University, USA. He is an active Senior IEEE Member and IEEE Computer Society Member, contributing to the global technology community through thought leadership, technical

publications, and industry collaborations. Committed to bridging research and real-world applications, he continues to leverage technology to build secure, scalable, and intelligent solutions that drive business success.

- **Vishwa** is passionate about writing clean, efficient, and maintainable code and is a strong advocate for best practices in software engineering, including **test-driven development** (**TDD**), code reviews, and continuous improvement.

  Currently, Vishwa is working at Publicis Sapient , where he is leading the development of cutting-edge microservices solutions that drive business value and innovation.

  Vishwa, thrives on solving complex technical challenges and enjoys mentoring teams to elevate their skills in Go and distributed systems. He is committed to helping organizations harness the full potential of Golang to build scalable, reliable, and high-performance applications.

Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to this book. A special thanks to my family and friends for their unwavering support and encouragement. Your love and motivation have been invaluable.

I am very grateful to BPB Publications for their guidance and expertise in bringing this book to life. Their support was crucial in navigating the publishing process. Thank you to the reviewers, technical experts, and editors for your valuable feedback. Your insights have greatly improved the quality of the book.

Finally, I want to thank the readers for your interest and support. Thank you to everyone who helped make this book a reality.

Preface

Go (Golang) is vital in today's software development and is a modern and efficient language. The book covers essential concepts for Go programming.

This book has thirteen focused chapters. It covers topics for understanding Go. We start with an introduction to Go and its setup. This guides you through setting up your environment. We then cover data types and control structures. Functions and error handling are also explained. Working with strings and slices builds a solid base. You will learn Go's core syntax and features.

Chapters 5 through 7 explore advanced topics. These include Go reflections and concurrency. Structs, methods, and interfaces are also covered. These showcase Go's unique capabilities. You will learn to build concurrent applications. Working with JSON and HTTP is discussed. Logging and testing are also covered. The usage of Go for web development and distributed systems is explained. These show how Go creates real-world solutions. Finally, we cover generics. An overview of Go for security and cryptography is also included. You will learn about advancements and secure coding.

This book is for anyone learning Go. It is for beginners and experienced developers. It is a portable reference and a guide to Go's key concepts for students and professionals.

Get Set Go aims to equip readers with knowledge. You will learn to write clean, efficient and maintainable Go code. You can build web servers, command-line tools, and distributed systems with this knowledge. I hope this book is a valuable resource. It will help you explore Go programming.

Chapter 1: Introduction to Go - This chapter dives into the fundamentals of Go and explore the historical context behind its creation. It starts by uncovering the reasons that led to the development of the language and examines how its design addresses the challenges of large-scale software engineering. Key

milestones in Go's evolution are highlighted, showcasing why it has become a go-to choice for modern software solutions. Along the way, we will examine the features that set Go apart, such as simplicity, efficiency, and effectiveness in building scalable and high-performance applications. Additionally, the chapter introduces the essential fmt package, demonstrating how it can be used to format and print output effectively.

Chapter 2: Data Types and Control Structures - In this chapter, we will explore the core building blocks of Go, data types and control structures. We will cover Go's basic types—integers, floats, strings, and booleans—and the concept of zero values. Zero values are default values assigned to uninitialized variables.

We will also dive into variable declaration methods using var, type inference, shorthand declarations, and variable scope. Composite types like arrays, slices, maps, and structs are explained, along with constants and enumerations using iota.

You will learn to manage program flow with conditional statements and looping constructs. Finally, we will address type conversions in Go's statically typed system. By the end, you will be well-versed in handling data and controlling program execution in Go.

Chapter 3: Functions and Error Handling - In Go, functions are a fundamental component of building any application. They enable code reusability, modularity, and organization. Understanding how to declare and use functions is key to writing clean and maintainable code. This chapter covers the syntax for declaring functions, different types of parameters, and return values. It also touches upon advanced concepts like variadic functions, anonymous functions, closures, and error handling. By mastering these topics, you will be equipped to write more flexible and reliable code in Go.

We will also dive into error handling topics like throwing errors, defining, and handling custom errors.

Chapter 4: Strings and Files - In this chapter, we will delve into two fundamental aspects of programming in Go: string manipulation and file handling. This chapter explores Go's built-

in packages, the strings and os packages. We will discuss essential string manipulation functions, UTF-8 encoding, and immutable string behavior. Additionally, we will dive into file and directory handling, which is essential for managing data storage, creating files and directories, and handling input and output operations. Command line arguments will also be covered. It gives us the flexibility to interact with programs directly from the terminal. Together, these topics provide a strong foundation for building powerful applications in Go.

Chapter 5: Go Reflection - Reflection in Go is a powerful feature that allows developers to inspect and manipulate types and values at runtime. This capability can make your code more dynamic and flexible. It enables you to write programs that can adapt to different types and structures without knowing them at compile time. In this chapter, we will explore the fundamentals of Go's reflect package, which provides the tools necessary for reflection. We will cover how to extract type information, work with values, and even modify them programmatically.

Chapter 6: Concurrency - Concurrency is one of Go's most powerful features that lets us write programs that are efficient at performing multiple tasks simultaneously. Concurrency in Go is driven by the lightweight and highly efficient nature of goroutines. In this chapter, we will delve into Go's concurrency model, exploring goroutines, channels, and synchronization mechanisms. Additionally, we will also explore the context package and understand how this package helps manage goroutines.

Chapter 7: Structs, Methods, and Interfaces - Structs are a foundational feature in Go that allow developers to group related data into cohesive units. Unlike arrays and slices, which store elements of a single type, structs can hold fields of differing types. In this chapter, we will explore how to define, instantiate, and use structs in Go. We will delve into advanced features like struct embedding for creating composite structures, overriding embedded fields, and defining methods to attach behavior to structs, enabling more organized and maintainable code. We will

also cover interfaces, their implementation, type assertion and type switch statement.

Chapter 8: Working with JSON and HTTP - In modern web development, handling data efficiently and building robust HTTP servers are fundamental skills. Go's standard library provides powerful tools to achieve this. The net/http package creates web servers and the encoding/json package seamlessly works with JSON data. This chapter dives into these essential components, offering a comprehensive guide to building web applications in Go. From encoding and decoding JSON to implementing middleware and context handling, you will learn how to create efficient, and maintainable web servers that cater to dynamic client requirements.

Chapter 9: Logging and Testing - In the world of software development, logging and testing are indispensable tools that ensure the reliability and maintainability of applications. Logging provides insights into the behavior of a program, helping developers diagnose issues and understand application flow. Testing, on the other hand, verifies that code behaves as expected, preventing bugs and regressions.

In this chapter, we will explore the logging capabilities provided by Go, including the use of the log package and the newer slog package for structured logging. We will also delve into Go's testing framework, covering how to write unit tests, run benchmarks, and effectively debug your code.

Chapter 10: Go in Web Development - Web development with Go has gained traction due to its simplicity, performance, and concurrency support. Unlike many other languages, Go offers a robust standard library with built-in packages like net/http for web servers and html/template for rendering dynamic content. This reduces the need for external dependencies. Go's strong typing and compile-time checks help catch errors early. The lightweight goroutines enable efficient handling of thousands of concurrent requests. Additionally, Go’s cross-compilation support ensures seamless deployment across platforms.

In this chapter, we will explore how to build web applications using Go, covering essential concepts such as building a simple web server, templating, handling forms and user input, integrating with popular web frameworks, and implementing basic authentication and session management.

Chapter 11: Go in Distributed Systems - In modern software development, distributed systems have become a cornerstone for building scalable and resilient applications. Go, with its simplicity and efficiency, is well-suited for developing such systems. This chapter delves into the use of Go in distributed systems, focusing on microservices, the context packages, and integrating with Apache Kafka using the Sarama library. We will also explore implementing distributed locks using Redis.

Chapter 12: Generics - In this chapter, we will explore one of Go's most anticipated features—generics. Generics were introduced in Go 1.18. Generics enable us to write reusable and flexible code by making functions and data structures to work with any data type while maintaining type safety. This chapter provides a guide to the syntax and structure of generics, from simple type parameters to advanced constraints. You will learn how to create generic functions and types, and use constraints to restrict type parameters.

Chapter 13: Go for Security and Cryptography - Security is a crucial aspect of software development, and Go provides robust tools to help developers secure their applications. In this chapter, we will explore how to secure your Go applications using the crypto package and best practices for writing secure Go code. We will cover the fundamentals of cryptography in Go, delve into hashing and encryption, learn how to generate secure random numbers, and understand how to handle user authentication and authorization with JWT.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/zt1vbph

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Get-Set-Go**.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Introduction to Go

Introduction

This chapter dives into the fundamentals of Go and explores the historical context behind its creation. It starts by uncovering the reasons that led to the development of the language and examines how its design addresses the challenges of large-scale software engineering. Key milestones in Go's evolution are highlighted, showcasing why it has become a go-to choice for modern software solutions. Along the way, we will examine the features that set Go apart, such as simplicity, efficiency, and effectiveness in building scalable and high-performance applications. Additionally, the chapter introduces the essential **fmt** package, demonstrating how it can be used to format and print output effectively.

Structure

This chapter covers the following topics:

- Introduction to Go and its history
- Setting up the Go environment
- Introduction to Go modules
- Writing and running your first Go program
- Format output using fmt package

Objectives

By the end of this chapter, you will understand the core principles behind Go's creation and the historical context that shaped its development. You will learn why Go was designed for simplicity, scalability, and performance, and how its features address the needs of modern software engineering. You will also explore Go's unique advantages in areas like concurrency, memory management, and cross-platform development. Additionally, you will be able to set up a Go development environment and gain insight into Go's key milestones and growing ecosystem.

Introduction to Go and its history

Go, often referred to as Golang, is a statically typed, compiled programming language. It is designed for simplicity, efficiency, and reliability. It was created by *Robert Griesemer*, *Rob Pike*, and *Ken Thompson* at *Google* and was first announced to the public in November 2009. The language was born out of a need to address the challenges of software development at scale, particularly within Google's vast infrastructure. Golang merges the performance of C with the simplicity and productivity of high-level languages like Python and Ruby.

Genesis of Go

The inception of Go can be traced back to late 2007 when the creators, frustrated with the complexity and inefficiency of existing languages, decided to design a new language. They wanted to create a language that could better meet the demands of modern software development. They aimed to create a language that combined the performance and safety of statically typed languages with the speed of dynamic languages.

Here are the key milestones of Go development:

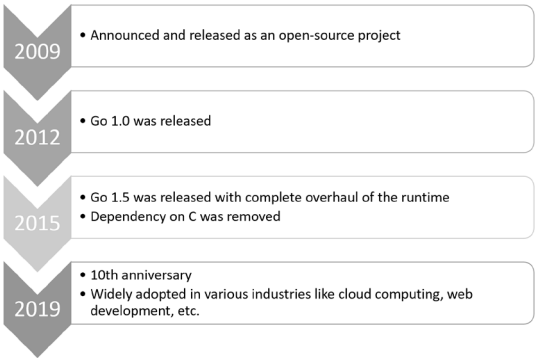

***Figure 1.1**: Key milestones of Go development*

- **2009**: Go was officially announced and released as an open-source project. This release included a compiler, tools, and a standard library, setting the stage for community involvement and rapid evolution.
- **2012**: Go 1.0, the language's first stable version was released in 2012. It established a stable foundation for Go's syntax and semantics. It also ensured backward compatibility for future versions.
- **2015**: The release of Go 1.5 was significant as it included a complete overhaul of the runtime. It also removed the dependency on C and made Go a self-hosting language.
- **2019**: Go celebrated its 10th anniversary, with a thriving community. Go received widespread adoption in various industries, from cloud computing to web development and beyond. According to Go developer survey 2024 H2 results (**https://go.dev/blog/survey/2024-h2-results**), Go continues to enjoy strong developer satisfaction (93%), with ease of development and user-friendly APIs being key strength on major cloud platforms.

Why Go?

Go shines in scenarios where scalability, simplicity, and performance are key. It has become a popular choice for building modern web services, microservices, DevOps tools, and cloud-native applications. Go's strong support for concurrency makes it

an excellent choice for applications requiring parallel processing. Its efficient memory usage helps in building high-performance servers and networked systems.

Go's impact and adoption

With a growing ecosystem and strong community support, Go has carved out a niche for itself as a language. It is particularly favoured in the following areas:

- **Cloud services**: Companies like Google, Docker, and Kubernetes use Go to build scalable and efficient cloud services.
- **Web development**: Go's performance and simplicity make it an excellent choice for web servers and APIs.
- **DevOps**: Tools like Terraform and Prometheus, which are essential in the DevOps ecosystem, are written in Go.

Key features of Go

Go has several features that make it an attractive choice for modern software development. We will take a look at them in this section.

Simplicity and clean syntax

Go is simple and has concise syntax. Go was designed to be easy to learn and read. The language avoids complex abstractions like inheritance, operator overloading, and implicit type conversion, promoting clarity in code.

Statically typed with type inference

Go is a statically typed language. This means variable types are checked at compile time. This provides more safety and preventing many types of bugs.

However, Go also supports type inference, allowing the compiler to automatically infer the type of variables based on the values assigned to them.

Let us look at this example:

```
var a int = 10
b := 20 // type inferred as int
```

Here, **a** is explicitly defined as an integer, while **b** is automatically inferred as an integer.

Efficient concurrency model

Go provides built-in support for concurrent programming through goroutines and channels. **Goroutines** are lightweight threads, managed by the Go runtime, and consume very little memory. This allows us to run thousands of concurrent tasks in parallel. Goroutines work in tandem with **channels**, which enable safe communication between them. This concurrency model simplifies the process of writing concurrent programs. It also avoids common issues like deadlocks and race conditions. This makes it easier to run multiple tasks simultaneously and leverage multi-core processors.

Goroutines and channels enable efficient concurrency, as evidenced by benchmarks (Golang vs. other languages, **https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html**).

Garbage collection

Go does automatic garbage collection. This means the Go runtime takes care of memory allocation and deallocation. Unlike C and C++, this relieves developers of the need to manually manage memory. Go's garbage collector is highly efficient and does not impact performance significantly. The garbage collection is optimized for latency and scalability. It is crucial for building web servers, microservices, and large-scale systems.

Fast compilation and execution

Unlike interpreted languages like Python and Ruby, Go is compiled. It converts code directly into machine code before execution. This results in faster execution and better performance. Go's compiler compiles large codebases quickly, even in projects with thousands of source files. The fast compilation and execution make Go ideal for scenarios where performance is critical.

Robust standard library

Go comes with an extensive standard library that simplifies common tasks, such as working with files, handling network