Learning Go with Networking

Automating network programming, operations, and security

Yoganath T.V.

Balachandar A.

www.bpbonline.com

First Edition 2025

Copyright © BPB Publications, India

ISBN: 978-93-65899-597

Dedicated to

Family and friends

About the Authors

- **Yogananth T.V.,** was born in the temple city of South India, Madurai, and brought up in Chennai for schooling and college. He has secured M.S. in software engineering from BITS PILANI - 2004 & PC-SCM from IIT Roorkee - 2023. At the time of writing this book, he works in Gorilla Technology as solutions director and is pursuing a Ph.D at Shiv Nadar University with blockchain as core domain. Previously, he worked with the Reliance Jio Platform (supply chain), and HCL Technologies for Cisco Systems Inc.

  He has overall 24+ years of IT experience in various domains, with core competence in Cisco networking, unified communication, and DataCenter. Developing products and services for I18N, GST, Azure/IoT, and test automation frameworks, he has shown extensive experience in Java, Python, and Golang. His career started as a fresher at HCL Technologies-Cisco Systems in 2000, when everyone was fascinated by the Y2K problem. During the early stages in his career, he was involved in the development of SIP protocol stack in C/C++. He was certified as CCIE-VOICE, CCNP-R&S, ONAP-Network Automation, CCDH (Hadoop), VCP-4.0.

  Apart from domain certifications, he has also earned JLPT N4, Japanese Language Proficiency. He is highly adaptive, a multitasking, systematic personality, and known for his problem-solving skills, mentoring abilities, and a pronounced teacher on various technologies like networking, virtualization, OpenStack, and Kubernetes. He also performed the role of guest faculty at BITS Pilani, conducting a course titled, Design and Operations of Datacenter.

- **Balachandar A.** (born on 11 November 1991), holds a master's degree MCA - Computer from RVS College of Engineering and Tech – Dindigul, Tamil Nadu, India. He is a Senior software engineer with hands-on experience in Golang, Python, REST API's, CLI based applications, scripting, microservices, SAAS applications, AWS, CI/CD automation, Docker Containerization, Linux, application solution designing, large scale applications development, all levels of testing, including performance, functional, integration, system, regression and user acceptance testing and cloud. He is a supportive and enthusiastic team player dedicated to streamlining, processing, and efficiently resolving project issues. His contributions to this field have been rewarded by Telstra, where he has successfully led and delivered multiple projects in enterprise applications and infrastructure cloud-based solutions. He is also a passionate traveler.

About the Reviewer

Shrinath Thube is a seasoned software developer at IBM, USA, and a technology leader with expertise in cloud security, 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.

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 in electrical engineering from San Jose State University. An active Senior IEEE Member, he contributes to the tech community through thought leadership, technical publications, and industry collaborations.

Acknowledgements

- I want to express my sincere and deepest gratitude to Mr. Babu Jayaraj (VP- supply chain platform for e-commerce and new commerce (Ajio/Digital/JioMart), one who introduced Golang to me, around the year 2015. It was an excellent opportunity to work and learn Golang from him. Another member who inspires me is Dr. Pethuraj (Chief architect and vice president, edge AI division, Reliance Jio Platforms Ltd), who can talk about any technology that would interest you and is also the author of a good number of books.

  I am also grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. It was a long journey of revising this book, with valuable participation and collaboration of reviewers, technical experts, and editors.

  The list would not be complete if I did not mention the 3 Gorillas. Yes, you read it right – Gorillas! Mr. Mohan Rajkumar (CDO, Gorilla Technology Group) with whom I had worked over a decade and who introduced me to the other two Gorillas: Mr. Jay Chandan (CEO/Chairman, Gorilla Technology Group) and Dr. Rajesh Natarajan (CTO, Gorilla Technology Group). I extend my sincere thanks to them, as they are going to be early readers of my book.

  And of course, thanks to my family members for their support, especially my kids, who are keen to hear about the release of this book.

  Finally, I would like to thank all the readers who have taken an interest in my book and for their support in making it a reality. Your encouragement has been invaluable.

  Yogananth T.V.

- I would like to express my heartfelt gratitude to Chandra Narayanasamy, co-founder of Nomiso, who first introduced me to Golang back in 2015. That introduction marked a pivotal moment in my professional journey, sparking a deep interest in the language that ultimately led to this book.

  My sincere thanks also go to Mr. Babu Jayaraj, vice president – supply chain platform for e-commerce and new commerce (Ajio/Digital/JioMart), whose guidance in solutioning and designing, played a crucial role in shaping my understanding and practical application of Golang. Working with him has been a highly enriching experience.

I would also like to express my sincere gratitude to Yoganath. T.V, solutions director for R&D, whose expertise in areas such as system design, cloud, virtualization, IoT, and more has been a significant source of learning and inspiration for me. His guidance in solutioning and deep technical knowledge have played a key role in broadening my understanding of the evolving tech landscape.

I am also deeply thankful to Ashish Jadhav, senior vice president of technology and director of architecture at Reliance Industries Limited, Digital Business Group (Jio Platforms Limited). His leadership, vision, and deep architectural insights have consistently inspired me to think bigger and build better.

A big thank you to BPB Publications for their unwavering support and professional guidance in bringing this book to publication. The long process of revisions and fine-tuning was made possible through the valuable collaboration of dedicated reviewers, technical experts, and editors.

A very special thanks to my wife, Manimekalai, whose patience, understanding, and unwavering support have sustained me throughout this journey. Her encouragement and excitement about the release of this book have been my true driving force. I am equally grateful to my parents, whose blessings, values, and constant encouragement have shaped who I am and supported me every step of the way.

Finally, I extend my heartfelt thanks to all the readers. Your interest and support have made this endeavor truly meaningful. I hope this book proves valuable as you embark on or continue your journey with Golang.

Balachandar A.

Preface

This book is the result of that curiosity and the hours we spent experimenting, building, breaking, and learning Golang in building scalable systems to solve real-world problems.

This is not a book full of complicated theory. It is a hands-on guide written by learners, for learners. We begin with the basics—introducing Go, its speed, simplicity, and why it is a great language for building modern systems. Then, we move into networking, showing how machines connect using TCP, UDP, and sockets.

From there, we explore how to build real web services, work with data formats like JSON and XML, and use tools like Keycloak for identity management. You will also learn about network packages and how Go makes high-performance server programming easier. Performance matters, and so we look at ways to make programs faster using profiling and optimization. We dive into automation too—how Go can help manage devices, networks, and cloud services.

Security is also a big focus here. We covered authentication, authorization, cryptography, and OWASP vulnerabilities. Moreover, we also explore ethical hacking with Go in safe, controlled environments.

We ended with a glossary to make sure you do not get stuck on tricky terms. This book may not be perfect—but it is honest, practical, and made with care. We are excited to share what we have learned.

This book is all about using the Go programming language (also called Golang) to build real-world systems. It teaches you not just how to write code, but how to make things work—like building fast and safe websites, connecting to other computers, securing information, and even testing networks in a safe way. Each chapter has hands-on examples to help you understand how these things work in real life.

The chapters are arranged so that you start with the basics and then move to more advanced stuff step by step. Here is a quick look at what each chapter covers:

Chapter 1: Introduction to Go Language - Learn where Go comes from, why it is popular, and how it helps developers build strong, fast programs without too much complexity.

Chapter 2: Networking Essentials - Understand how computers talk to each other over a network. This chapter explains important concepts like TCP, UDP, and how to make simple connections in Go.

Chapter 3: Application Essentials - Build your own web services. You will learn how to send and receive data, connect with databases, and make applications that people can use over the internet.

Chapter 4: Data Essentials - Learn how to work with different kinds of data, like JSON and XML. You will also learn how computers call each other's functions remotely using RPC.

Chapter 5: Network Packages Unleashed - Use Go's powerful libraries to build tools that help you scan networks, analyze data, and create efficient server systems.

Chapter 6: Introduction to Performance Essentials - Speed matters. This chapter shows how to make your Go programs run faster and use less memory, with tools that help you measure performance.

Chapter 7: Automation Essentials - Learn how to write Go programs that automate tasks—like managing servers or setting up devices—making your life (and job) easier.

Chapter 8: Authentication, Authorization, and Cryptography - Security is super important. This chapter covers how to verify users, control access to systems, and protect data using cryptography.

Chapter 9: OWASP with Golang - Find out about the most common security problems in websites and how to fix them using Go. You will also try out tools that help catch these issues early.

Chapter 10: Hacking the Network - This one is exciting to every programmer or coder! It is about about ethical hacking, how to test networks safely using Go, like simulating attacks in a lab to learn how to defend against them.

APPENDIX: Technical Essentials - This final chapter explains important terms and concepts in a simple way, so you can quickly look things up while learning or building your own projects.

Code Bundle and Coloured Images

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

https://rebrand.ly/22b503j

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Learning-Go-with-Networking.
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.

Table of Contents

Chapter 1
Introduction to Go Language

Introduction

By 2007, Google was developing Go (popularly known as Golang) in C, an open-source programming language launched in 2009 with BSD licensing for Linux and Mac Platforms. GO 1.0 is the first production-ready version which was released in March 2012. We are currently at Golang version 1.23 (which we will use in this entire chapter).

Many other languages like C, C++, Java, and C# inspired the development of this language. For instance, the idea of package definitions in this programming language draws inspiration from Java and C#, while the concept of interfaces is influenced by Java's inheritance mechanism. Moreover, the syntax and structure of this language bear a strong resemblance to C.

While pronounced programming language exists widely, you might be thinking, why is there yet another programming language? Well, all these languages suffer from the great compilation overhead with an increased footprint of artifacts getting generated.

For example, the #include header files, during the build, are included as many times as referenced in the overall code, resulting in duplicate import and increased artifact size. Golang avoids this duplication, resulting in fabulous compilation and building speed through its inbuilt dependency management. The compilation aspect of Golang code is almost negligible compared to other languages. It is also highly independent of the underlying platform architecture. For example, no JVM requirement, which has its limitation in terms of maximum memory, could be allocated to a JVM. It can be cross compiled by setting the GOARCH environment variable.

Structure

In this chapter, we will cover the following topics:

- Choosing Golang
- Setting up the environment
- Environment variables
- Folder structures
- Welcome to Go
- Packages
- Modules
- Import
- Export
- Data types

Objectives

This chapter contains an introduction to Golang with basic elements and features like yield or generator, concurrency, oops, generics, fuzzing, and any other core components of Golang.

Before delving into the details of setting up the environment, understanding environment variables, folder structure, the significance of **`main`** and **`init`** functions, and different data types, it is essential to lay a solid foundation in programming. This introductory passage will provide you with an overview of these topics and their importance in the development process.

Choosing Golang

Golang (Go) is widely regarded as an efficient and high-performance language that is particularly suitable for systems programming, backend services, and cloud-native applications. This section explores Go's key advantages, benchmark comparisons with other languages, and its suitability for various use cases.

Performance and efficiency

Go is known for its excellent performance, as it compiles directly into machine code, eliminating the need for an interpreter or **virtual machine** (**VM**). This results in faster execution times and lower memory consumption.

Benchmark comparisons

The following table provides a performance comparison of Golang with other popular programming languages:

Feature	Golang 1.23	C++	Java	Python
Compilation Time (ms)	150	450	600	2000
Execution Speed (Ops/sec)	120,000	140,000	100,000	20,000
Memory Usage (MB)	25	20	50	90
Concurrency Support	High	Medium	Medium	Low
Binary Size (MB)	2.5	1.8	15	N/A

***Table 1.1**: Performance comparison of Golang with other popular programming languages. Data sourced from various benchmarking sites and tested on an Intel i7 processor with 16GB RAM.*

Concurrency and scalability

One of Go's biggest strengths is its lightweight concurrency model, which makes it ideal for scalable applications. Instead of OS threads, Go uses Goroutines, which are much lighter and consume fewer resources. Let us go over the following:

- **Goroutines versus threads**: Unlike traditional threading models in Java and C++, Go's Goroutines require minimal memory and can efficiently handle thousands of concurrent operations.
- **Channels for safe communication**: Go provides channels for synchronized communication between Goroutines, reducing race conditions and complexity.

Simplicity and readability

Go is designed to be simple, with a clean syntax that makes it easy to learn and maintain:

- **Minimalist syntax**: Unlike C++ or Java, Go avoids complex features like manual memory management, class inheritance, and templates.
- **Garbage collection**: Go includes an optimized garbage collector that ensures efficient memory management.
- **Standard library**: Go has a robust standard library with built-in support for networking, file I/O, and encryption.

Cross-platform and portability

Go offers native support for cross-compilation, enabling developers to compile binaries for different platforms without additional dependencies. Its features are:

- **No JVM required**: Unlike Java, Go does not require a runtime environment, making it ideal for lightweight deployments.
- **Cross-compilation**: Setting the GOARCH environment variable allows easy compilation for different system architectures.

Adoption in DevOps and cloud-native applications

Go has become the preferred choice for DevOps tools and cloud-native applications due to its efficiency and scalability. Its features are:

- **Used by Kubernetes, Docker, and Terraform**: Major cloud-native platforms are written in Go.
- **Microservices-friendly**: Go's lightweight nature makes it a great choice for building microservices.
- **Fast deployment**: The small binary size and fast execution speed enhance deployment efficiency in containerized environments.

Setting up the environment

One of the initial steps in programming is setting up the environment. This involves installing the necessary software tools and configuring them to create an environment suitable for coding. By properly setting up the environment, you ensure that you have the required compilers, libraries, and dependencies to write and execute your code efficiently:

- **Environment variables**: Environment variables play a crucial role in the programming ecosystem. They are dynamic values that can affect the behavior of your software. Understanding how environment variables work and how to manipulate them allows you to control various aspects of your application, such as configuring paths, defining constants, or adjusting runtime behavior.
- **Folder structure**: Organizing your project files and directories in a logical and consistent manner is essential for maintaining a structured and manageable codebase. A well-defined folder structure helps in locating files, separating concerns, and collaborating with other developers. We will explore best practices for designing a folder structure that promotes modularity, scalability, and ease of maintenance.

Binary distributions can be found at: **https://go.dev/dl/**

The source code can be downloaded from: **https://github.com/golang/go/tree/release-branch.go1.20**

Refer to **https://go.dev/doc/install** for installation instructions for Linux, Mac, or Windows.

The following is the text given in the installation instructions, which is tested and verified for functionality.

Installation of Go in Linux

To install Go on Linux and verify its installation, follow these steps:

1. Downloading Go:
    a. Open a web browser and visit the official Go website: **https://golang.org/dl/**.
    b. On the Downloads page, find the Linux distribution that matches your system architecture (For example, Linux x86-64).
    c. Click the download link to save the Go installation archive (tarball) to your computer.
2. Installing Go:
    a. Open a terminal window.
    b. Navigate to the directory where you downloaded the Go installation archive.
    c. Extract the tarball using the following command:

    ```
    tar -C /usr/local -xzf go<version>.linux-amd64.tar.gz
    ```

    Replace `<version>` with the version number you downloaded. This will extract the Go binaries to the `/usr/local/go` directory.
3. Configuring Go environment variables:
    a. Open your shell's configuration file (For example, `.bashrc`, `.bash_profile`, or `.zshrc`) using a text editor:

    ```
    vim ~/.bashrc
    ```

    b. Add the following lines at the end of the file to set Go-related environment variables:

    ```
    export PATH=$PATH:/usr/local/go/bin
    export GOPATH=$HOME/go
    export PATH=$PATH:$GOPATH/bin
    ```

    Save the file and exit the text editor.
    c. Reload the shell configuration by running the following command:

    ```
    source ~/.bashrc
    ```

    This will update the environment variables for the current shell session.