# Let Us Go!

A Complete Beginner's Guide to Golang Programming and Development

Rahul Sid Patil

Apress®

# Let Us Go!

## A Complete Beginner's Guide to Golang Programming and Development

**Rahul Sid Patil**

Apress®

## *Let Us Go!: A Complete Beginner's Guide to Golang Programming and Development*

Rahul Sid Patil
Pune, Maharashtra, India

ISBN-13 (pbk): 979-8-8688-1441-9 ISBN-13 (electronic): 979-8-8688-1442-6
https://doi.org/10.1007/979-8-8688-1442-6



Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Anandadeep Roy
Editorial Assistant: Jessica Vakili

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny @springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a Delaware LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

*I dedicate this book to all the budding Gophers and cloud-native developers who are shaping the future of modern, intelligent, and cloud-native enterprise software. Your passion and innovation are building the foundation for a smarter and more connected world, driving technology to make life better for humankind. May this book serve as a guide and inspiration on your journey.*

# Table of Contents

# About the Author

**Rahul Sid Patil** is a seasoned software engineer and thought leader with more than a decade of experience in Golang, cloud-native development, and distributed systems. As the Head of the Golang Community at EPAM Systems, India, he plays a pivotal role in fostering collaboration and innovation within the developer ecosystem.

A dynamic public speaker, Rahul has shared his expertise at prestigious international tech events, including the Great Indian Developer Summit (GIDS). He has also served as a mentor and jury member at the Smart India Hackathon, one of the world's largest hackathons, inspiring and guiding the next generation of tech talent.

Rahul is a popular author on Medium, where he writes on software development and modern technology practices, and an open source contributor known for the "crongen" library. He is the founder of the Cloud Native Developer's Forum (CNDF), a YouTube channel dedicated to educating developers on cloud-native technologies.

# About the Technical Reviewer

**Pranav Manole** is a seasoned web developer with experience in working with various web technologies. He's been working in the IT industry for more than 7.5 years. Currently, he is working as a Senior Software Engineer at Victoria's Secret & Co., Bangalore. For the past four years, he has been actively working in Golang and exploring various aspects of it. He feels Golang, one of the most widely accepted languages, is very easy to learn. Due to its key features like concurrency, memory management, error handling, and most importantly simplicity, it is getting popular in the IT industry. He feels this book is written in such a way that the reader will find it engaging due to the question-and-answer structure. The what, why, where, and how of Golang concepts are explained in a systematic way.

# Acknowledgments

I would like to extend my heartfelt gratitude to the following people who have played a crucial role in making this book a reality:

- **Anandadeep Roy**, editor (web development and open source), for giving me the opportunity to write this book and for his invaluable guidance throughout the process
- **Deepa Shirley Tryphosa**, project coordinator for this book, for her patient follow-ups and insightful inputs that helped shape this work
- **Pranav Manole**, Senior Software Engineer at Victoria's Secret & Co., for his thorough and on-point technical review, ensuring the quality and accuracy of the content
- **My family**—my wife Bhagyashree, my daughter Urvi, and my mother Nandini—for their unwavering support and understanding of my lack of availability for the family during this project

Without their encouragement and dedication, this book would not have been possible.

# Introduction

Welcome to *Let Us Go!*, your practical guide to mastering Go programming! This book is designed to help you navigate the fundamentals of Go, a modern programming language that's transforming the way developers build scalable, efficient, and high-performance applications.

Whether you are

> **A beginner programmer** with little to no prior experience in Go but eager to learn with clear, step-by-step guidance
>
> **A student** seeking a comprehensive resource to supplement your coursework and apply programming concepts through hands-on projects
>
> **A self-taught developer** transitioning into professional Go development, looking for a structured approach and practical examples
>
> **A hobbyist or enthusiast** keen on exploring a new language and building projects you can showcase
>
> **A professional developer** aiming to upskill and leverage Go's capabilities for enterprise-level software

this book is tailored to make your learning journey engaging, interactive, and rewarding.

# Why Golang Is the Future of Enterprise Software Development

In the fast-paced world of enterprise software development, efficiency, scalability, and maintainability are paramount. With modern businesses demanding robust, high-performance applications that can scale seamlessly in cloud environments, development teams are reevaluating their language choices. Enter Golang—a language that's not just a tool but a game-changer. Here's why enterprise software development companies are increasingly turning to Golang to build the future.

# The Rise of Golang: An Overview

Developed by Google in 2007 and released to the public in 2009, Golang (or simply Go) was designed to address the challenges of modern software development. With a syntax reminiscent of C, combined with modern programming constructs, Golang was created to balance simplicity and power. Today, it's the language behind some of the most critical systems, including Kubernetes, Docker, and Prometheus.

# Why Enterprises Love Golang

## 1. Blazing Performance Without Complexity

**Compiled Language**: Golang compiles directly to machine code, eliminating the overhead of runtime interpretation. This translates to faster execution times and better resource utilization.

**Optimized for Concurrency**: In Go, a **goroutine** is a lightweight way to perform multiple tasks at the same time. Imagine cooking dinner while the washing machine is running—**goroutines** allow your program to

handle such multitasking effortlessly. You'll get hands-on experience with **goroutines** when we cover concurrency in later chapters.

**Garbage Collection**: This is an automatic process that cleans up unused memory while your program is running. Think of it as a janitor that ensures your application runs smoothly without you having to manually manage memory. Don't worry—we'll dive deeper into how Go handles memory in later chapters.

---

**Note** If you're new to terms like "garbage collection" or "goroutines," don't worry! These concepts will be explained step by step with practical examples in later chapters.

---

## 2. Perfect for Cloud-Native Environments

**Built for Scalability**: In the age of distributed systems, Golang's concurrency model shines. Tools like Kubernetes, Docker, and Istio, all written in Go, leverage its capabilities to scale effortlessly in cloud environments.

**Cross-Platform Compatibility**: Go makes it easy to build applications that work across various platforms. Its ability to produce static binaries simplifies deployment, especially in containerized environments.

**Standard Library for Networking**: Go's robust standard library has built-in support for networking and HTTP servers, reducing the need for external dependencies and speeding up development.

## 3. Simplicity Equals Productivity

**Clean and Readable Syntax**: Go was designed to reduce cognitive load. With a focus on simplicity, it eliminates features like inheritance and complex metaprogramming, making the code easier to read, write, and maintain.

**Short Learning Curve**: For enterprises, onboarding new developers can be a challenge. Go's simplicity enables developers, even those new to the language, to become productive quickly.

**Minimal Magic**: Unlike languages that rely heavily on frameworks and hidden abstractions, Go's philosophy is "what you see is what you get." This predictability makes debugging and optimization straightforward.

## 4. Cost Efficiency

**Fewer Resources, Greater Output**: Thanks to Go's efficient use of resources, enterprises can achieve more with fewer servers and reduced operational costs.

**Developer Productivity**: Go's emphasis on simplicity and tooling ensures that development teams spend less time on debugging and refactoring and more on delivering value.

**Open Source Ecosystem**: Go's vibrant community and ecosystem mean enterprises can leverage a plethora of high-quality libraries and tools without additional costs.

## 5. Reliability and Maintainability

**Strong Typing**: Go uses a system called static typing, which means you must define the type of each variable (like whether it holds a number or text) before you use it. This helps catch errors early and makes your code more reliable. We'll explore Go's type system in later chapters.

**Built-In Testing**: Go includes a testing framework as part of its standard library, encouraging a test-driven development approach that improves software quality.

**Backward Compatibility**: Go has committed to backward compatibility in its updates, ensuring that existing codebases remain functional and future-proof.

# Use Cases Driving Golang Adoption

## 1. Microservices Architecture

Golang's lightweight binaries, rapid startup times, and support for gRPC make it an ideal choice for building scalable microservices architectures.

## 2. DevOps and Infrastructure Tools

The very tools that power DevOps—like Kubernetes, Docker, and Terraform—are written in Go. Enterprises building custom DevOps solutions are naturally drawn to the language.

## 3. High-Performance APIs

With its low latency and high concurrency capabilities, Go is widely used to build RESTful APIs and high-performance backend services for enterprises.

## 4. Real-Time Applications

From chat systems to gaming platforms, Golang's concurrency model is well suited for applications requiring real-time processing.

## 5. Big Data and Stream Processing

For tasks like log aggregation, real-time data processing, and ETL pipelines, Go's speed and scalability make it a strong contender.

# Case Studies: Enterprises Betting on Golang

## 1. Google

As the creator of Go, Google uses it extensively for internal tools and external projects like Kubernetes and gVisor.

## 2. Uber

Uber migrated many of its core services to Golang, citing its performance and ease of deployment as key advantages.

## 3. Netflix

Known for their real-time data processing needs, Netflix uses Go for various critical backend services, leveraging its efficiency and concurrency.

## 4. Dropbox

Dropbox transitioned from Python to Go for performance-critical components, reducing latency and improving scalability.

## 5. Monzo

This UK-based digital bank built its core banking platform using Go, emphasizing its role in creating scalable and reliable systems.

# Your Career with Golang: A World of Opportunities

As enterprises continue to embrace Golang for its performance and simplicity, the demand for skilled Go developers is skyrocketing. Whether you're an aspiring developer, a seasoned professional, or someone looking to pivot to a high-demand field, Golang opens doors to exciting and rewarding career opportunities. Let's explore why building a career in Golang could be more rewarding than pursuing other popular technologies like Python or Java.

## 1. Golang Career Paths

Golang offers versatile career opportunities across industries and roles. Here are some of the most prominent paths you can pursue:

**Backend Developer**: Design and build high-performance APIs, microservices, and distributed systems. Contribute to cloud-native applications that are scalable, secure, and efficient.

**DevOps Engineer**: Work on critical infrastructure tools like Kubernetes, Docker, and Terraform—written in Go. Build CI/CD pipelines, automate deployments, and manage scalable cloud environments.

**Systems Engineer**: Develop networking tools, proxies, and monitoring systems that handle large-scale traffic and data. Build custom operating systems and kernel modules optimized for performance.

**Data Engineer**: Create efficient ETL pipelines, data processors, and real-time stream processing applications. Leverage Go's concurrency features to process massive datasets with speed and accuracy.

**Open Source Contributor**: Join or contribute to major open source projects like Kubernetes, Istio, or Prometheus. Establish yourself as a leader in the thriving Go community.

**Freelance Developer**: Capitalize on the growing demand for Go developers in startups and enterprises seeking microservices, cloud-native apps, and DevOps solutions.

**Emerging Fields**

**Blockchain**: Develop fast and secure blockchain systems, leveraging Go's efficiency.

**AI/ML Tooling**: Although Go isn't primarily an AI/ML language, its integration capabilities make it a valuable tool for building AI pipelines.

## 2. Why Golang Is More Rewarding Than Other Technologies

**Enterprise-First Design**

While Python and Java excel in their respective domains, Golang is purpose-built for modern enterprise needs:

> **Python**: Known for data science and scripting, it lacks the performance and concurrency features required for enterprise-grade backend systems.
>
> **Java**: While powerful, its verbosity, steep learning curve, and reliance on frameworks make it less productive for quick iterations.

**High Demand, Low Supply**

As more companies adopt Golang, the talent pool remains limited compared to Python and Java developers. This gap translates to higher salaries and better job security for Go developers. Go's niche appeal means that mastering it can set you apart in the competitive tech job market.

**Faster Development Cycles**

Golang's simplicity and tooling (e.g., built-in testing, formatting, and linting) lead to faster development cycles, making developers highly

productive. This is a critical advantage in enterprise environments where time to market matters.

**Cross-Disciplinary Opportunities**
Go's ecosystem spans diverse domains, from cloud infrastructure to web services and networking. This breadth allows you to explore and switch domains without changing languages.

**Community and Open Source**
Go's vibrant and welcoming community offers mentorship, networking, and collaborative opportunities that can accelerate your career growth.

Contributing to high-impact projects like Kubernetes can boost your professional visibility and credibility.

**Future-Proofing Your Career**
With enterprises moving toward cloud-native, distributed architectures, the demand for Go is projected to grow exponentially.

Unlike older technologies that face gradual decline, Go's growth trajectory is upward, ensuring long-term career relevance.

## Conclusion: Your Golang Journey Awaits

Go is more than just a programming language—it's a gateway to building scalable, high-performance applications and an exciting career in modern software development.

Congratulations on taking the first step toward learning Go programming! This book is designed to make your journey enjoyable and rewarding, providing you with the skills to build practical Go applications, including command-line tools and web services.

With Go, you'll discover how simplicity and power come together to create efficient and reliable applications. By the end of this book, you'll have the knowledge and confidence to start building real-world Go projects on your own.

Let's not stop here—turn to *Chapter 1* and start coding your first Go program using Go Playground.

Together, we'll explore the foundations of Go and set the stage for creating useful, impactful applications.

**Let Us Go!**

CHAPTER 1

# Let Us Go on the Playground

## 1.1 Introduction

Welcome to your journey into Go programming! In this chapter, we will briefly explore the essential features of Go that form the foundation of the language. Through a series of sample programs that you can run on the Go Playground, you will gain a hands-on understanding of Go's syntax and semantics. By the end of this chapter, you will be equipped to write basic Go programs, laying a solid groundwork for more advanced and deep dive topics in the subsequent chapters.

## 1.2 The Go Playground

The Go Playground is an online tool that allows developers to write, run, and share Go programs in a web browser without needing to set up a local development environment. It provides a safe, sandboxed environment to execute code and supports most of the Go standard library. This tool is invaluable for beginners and professionals alike, offering a quick way to experiment with Go code, debug small snippets, or demonstrate functionality. However, it has limitations, such as no access to the file system or external network connections.


R. S. Patil, *Let Us Go!*, https://doi.org/10.1007/979-8-8688-1442-6_1

The Go Playground is an excellent tool for experimenting with Go code and sharing examples with others.

Try out the Go Playground at https://go.dev/play/.

---

**Note** All programs in this chapter can be run on the Go Playground. Just paste the code, click Run, and observe the output.

---

# 1.3 Writing and Running Basic Programs

Let's start with a simple "Hello World" program to get a feel for the Go Playground.

## 1.3.1 Hello World

The "Hello World" program is the starting point for learning any programming language. In Go, it demonstrates the simplicity and structure of the language.

**Sample Program:**

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, World!")
}
```

**Explanation:**

- **`package main`**: Defines the package name. The *`main`* package is mandatory for building an executable Go program.

- **`import "fmt"`**: Imports the *`fmt`* package for formatted I/O operations.
- **`func main()`**: Declares the *`main`* function, the entry point of the program.
- **`fmt.Println("Hello, World!")`**: Calls the *`Println`* function from the *`fmt`* package to print the text to the console, followed by a newline.

**Output Console:**

```
Hello, World!
```

Try this on Go Playground: https://go.dev/play/.

## 1.3.2 Values

Values represent immutable data such as *`numbers`*, *`strings`*, and *`booleans`*. They are used to define fixed data in a program and can be used in expressions.

**Sample Program:**

```
package main

import "fmt"

func main() {
    fmt.Println("Integer:", 42)
    fmt.Println("Float:", 3.14)
    fmt.Println("String:", "Go is fun!")
    fmt.Println("Boolean:", true)
}
```

**Explanation:**

- **`fmt.Println("Integer:", 42)`**: Prints an integer value
- **`fmt.Println("Float:", 3.14)`**: Prints a floating-point value
- **`fmt.Println("String:", "Go is fun!")`**: Prints a string value
- **`fmt.Println("Boolean:", true)`**: Prints a boolean value

**Output Console:**

```
Integer: 42
Float: 3.14
String: Go is fun!
Boolean: true
```

Try this on Go Playground: https://go.dev/play/.

## 1.3.3 Variables

Variables in Go store data that can be manipulated during program execution. They are declared using the `var` keyword or shorthand syntax (`:=`). Variables can be of primitive types like `int`, `float64`, `string`, or complex types like `structs`.

### a) Short Variable Declaration

The shorthand declaration uses `:=` and is often used inside functions.

**Sample Program:**

```
package main

import "fmt"

func main() {
    name := "Alice" // Shorthand declaration
    age := 30       // Type inferred as int
    fmt.Println("Name:", name)
    fmt.Println("Age:", age)
}
```

**Explanation:**

- **`name := "Alice"`**: Declares and initializes `name` with the value `"Alice"`.
- **`age := 30`**: Declares and initializes `age` with the value `30`. The type is inferred as `int`.
- **`fmt.Println(...)`**: Prints the values of `name` and `age`.

**Output Console:**

```
Name: Alice
Age: 30
```

Try this on Go Playground: https://go.dev/play/.

## b) Normal Variable Declaration

The `var` keyword is used for explicit variable declarations.