JANUARY 2026 SERIES

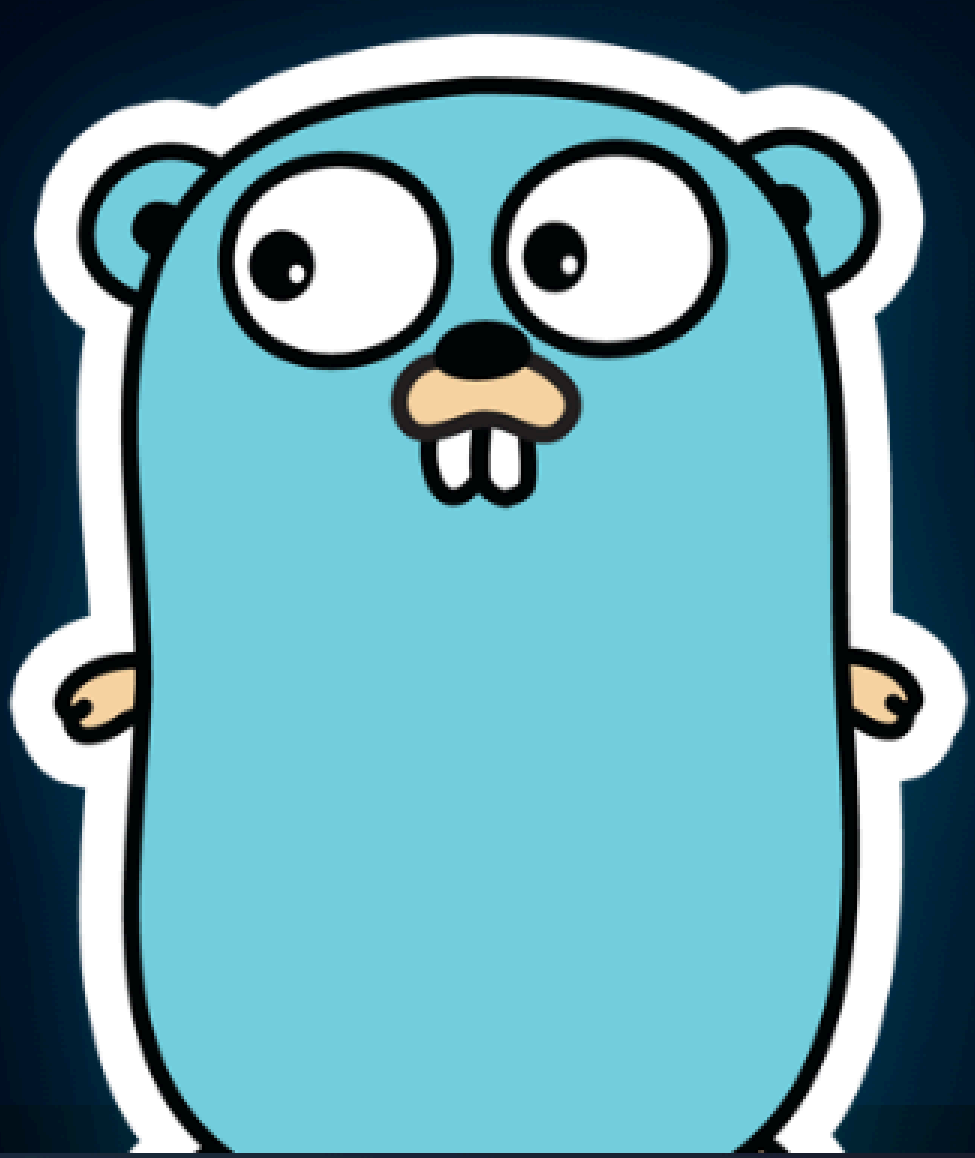
FROM GO BUILD TO GO RUN

GOLANG 2026 - NIV RAVE

#01

WHY GO?

PHILOSOPHY & STRENGTHS



Who And Why?

I am a Software (Mostly Backend) Engineer.
Across all the languages I've worked with (C++, C#, Python, JS, etc.), Go is the one that consistently surprised me with how simple and predictable backend development becomes.

At the same time, I've been looking for an extra challenge – something to "break the daily loop", something that lets me contribute back to others while doing something I love.

So I'm starting a 31 – day Go series.
Two posts a day.
Short, practical, incremental lessons.
Real code. Real explanations. No fluff.

This series is designed to provide incremental lessons in Go programming, delivering two posts daily that focus on practical skills . Each lesson aims to build on the previous one effectively.

Why Go stands out to me

My top 5:

1. Zero Magic
2. Honest Concurrency
3. Deployment Heaven
4. Go forces consistency
5. Coding in the age of AI

#1 - Zero Magic

There are no inheritance hierarchies, no magic and hidden frameworks, and almost zero implicit behaviour.

You focus on the problem, not the language, keeping things simple and explicit.

#2 - Honest Concurrency

Goroutines and channels are minimal but incredibly powerful primitives.

They don’t hide complexity – they expose it cleanly so you can actually reason about it.

#3 - Deployment Heaven

One binary. No environment hell.
The number of deployment issues this alone solves is ridiculous.
You build once → ship a single file → run it anywhere.

#4 - Go forces consistency

Code written by different developers tends to look the same.
Short, explicit, consistent.
This makes onboarding, debugging, and reading others' code far easier.
This part is so underrated!

#5 - Coding in the age of AI

Combining reasons #2 and #4, we get two huge advantages of using Go right now:

- AI workloads are inherently parallel. Running them in a language whose main strength is lightweight concurrency is a natural fit.

- Having an entire codebase that basically “looks the same” as training data for LLM models makes the outcome more robust – from my experience, using LLMs to write Go code will provide you almost the same code you would have written if you are experienced working with Go.

+

GOLANG /// BA

Let’s build something!

Join me for the next 31 days. Two posts daily.
Short, clear and practical.

Which of these 5 reasons is the biggest "win" for your workflow? Let’s chat below!