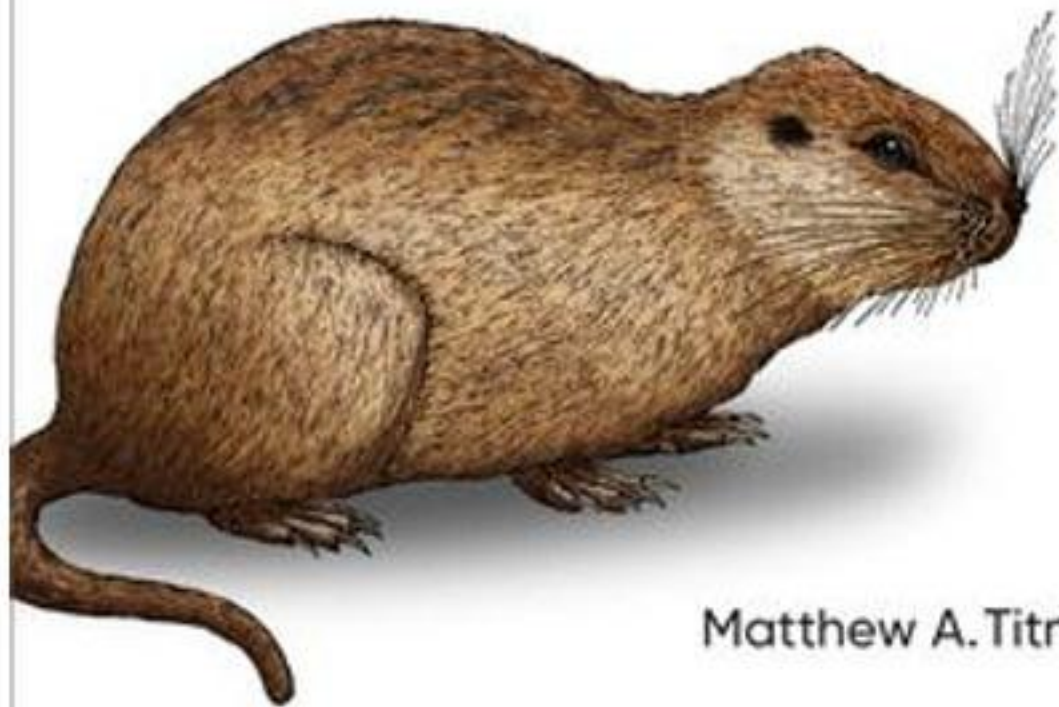


O'REILLY®

# Cloud Native Go

Building Reliable Services in  
Unreliable Environments



Matthew A. Titmus

## T

tag component, [Building your container image](#)

testability, [Verification and testing-Verification and testing](#), [VII. Data Isolation](#)

Three Pillars of Observability, [The “Three Pillars of Observability”-The “Three Pillars of Observability”](#)

throttle pattern, [Applicability](#), [Throttle-Sample code](#), [Circuit Breaking](#)

throttling, [Preventing Overload-Throttling](#)

Tickers, [Forever ticking tickers](#)

tight coupling

- communication, [Communications Between Services](#)

- definition of, [Tight Coupling](#)

- forms of, [Tight Coupling Takes Many Forms-Fixed addresses](#)

time dimension, [Forever ticking tickers](#)

time series, [Metrics](#)

timely function, [Forever ticking tickers](#)

timeout, [Timeout-Sample code](#)

timeouts, [Issuing HTTP Requests with net/http](#), [Timeouts](#), [Timing out gRPC client calls-Timing out gRPC client calls](#)



[calls](#)

[Timers](#), [Forever ticking tickers](#)

[timestamp](#), [Structure events for parsing](#)

[TLS](#), [Generation 3: Implementing Transport Layer Security-Transport Layer Summary](#)

[token bucket](#), [Implementation-Sample code](#), [Throttling](#)

[TOML](#), [Viper: The Swiss Army Knife of Configuration Packages](#)

[trace](#), [The “Three Pillars of Observability”](#)

[Tracer](#), [Obtaining a tracer](#)

[tracer provider](#), [Creating a tracer provider-Obtaining a tracer](#)

(see also [global tracer provider](#))

[traces](#), [Tracing Concepts](#)

[tracing](#), [The “Three Pillars of Observability”](#), [Tracing-Viewing your results in Jaeger](#)

(see also [distributed tracing](#))

[tracing exporters](#), [Creating the tracing exporters](#)

(see also [Console Exporter](#), [Jaeger Exporter](#))

[transaction log](#), [Generation 2: Persisting Resource State-Future improvements](#), [Our first plug](#)

[transitive downstream dependencies](#), [The Story So Far](#)

[transitive upstream dependencies](#), [The Story So Far](#)

transitive upstream dependencies, [The Story So Far](#)

transport layer security (see TLS)

Twelve-Factor App, [The Continuing Relevance of the Twelve-Factor App-XII. Administrative Processes, Configuring Your Application](#)

type assertion, [Type assertions](#)

type embedding, [Composition with Type Embedding-Struct embedding](#)  
(see also interface embedding, struct embedding)

types

arrays, [Container Types: Arrays, Slices, and Maps-Arrays](#)

Booleans, [Basic Data Types](#)

byte, [Simple Numbers, Strings as slices-Strings as slices](#)

containers, [Container Types: Arrays, Slices, and Maps-Map membership testing](#)

embedded pointers, [Struct embedding](#)

errors, [Error Handling](#)

extending standard composite types, [Methods](#)

floating point, [Simple Numbers](#)

interfaces, [Interfaces-Interfaces](#)



[interfaces](#), [interfaces-interfaces](#)

[maps](#), [Container Types: Arrays, Slices, and Maps](#), [Maps-Map membership testing](#)

[rune](#), [Simple Numbers](#), [Strings as slices-Strings as slices](#)

[signed integer](#), [Simple Numbers](#)

[slices](#), [Container Types: Arrays, Slices, and Maps](#), [Slices-Strings as slices](#)

[strings](#), [Basic Data Types, Strings](#), [Strings as slices-Strings as slices](#)

[structs](#), [Composition and Structural Typing](#), [Structs-Structs](#)

[unsigned integer](#), [Simple Numbers](#)

## U

[Uber Technologies](#), [The Jaeger Exporter](#)

[unary RPC](#), [Defining our service methods](#)

[underscore operator](#), [The Blank Identifier](#)

(see also [blank identifier](#))

[unsigned integer](#), [Simple Numbers](#)

[upstream dependencies](#), [The Story So Far](#)

(see also [transitive upstream dependencies](#))

## V

## V

value, [Your transaction log format](#)

variable declaration, [Variables-Short Variable Declarations](#)

variables, [Variables](#), [Pointers-Pointers](#)

variadic functions, [Working with slices](#), [Putting the Fun in Functions: Variadics and Closures-Passing slices as variadic values](#)

variadic operator, [Variadic Functions](#)

version control, [Configuration Good Practice](#)

vertical scaling, [Scalability](#), [VIII. Scalability](#), [Different Forms of Scaling](#), [The Four Common Bottlenecks](#)

vertical sharding, [Applicability-Implementation](#), [Minimizing locking with sharding](#)

Viper, [III. Configuration](#), [Viper: The Swiss Army Knife of Configuration Packages-Setting defaults in Viper](#)

## W

watchConfig function, [Making your configuration reloadable](#)

Windows Exporter, [Exposing the metrics endpoint](#)

write locks, [Making Your Data Structure Concurrency-Safe](#)

WriteDelete method, [Prototyping your transaction logger-Implementing your FileTransactionLogger](#),  
[Implementing your PostgresTransactionLogger](#)

WritePut method, [Prototyping your transaction logger-Implementing your FileTransactionLogger](#), [Implementing](#)



watchConfig function, [Making your configuration reloadable](#)

Windows Exporter, [Exposing the metrics endpoint](#)

write locks, [Making Your Data Structure Concurrency-Safe](#)

WriteDelete method, [Prototyping your transaction logger-Implementing your FileTransactionLogger](#),  
[Implementing your PostgresTransactionLogger](#)

WritePut method, [Prototyping your transaction logger-Implementing your FileTransactionLogger](#), [Implementing your PostgresTransactionLogger](#), [Reduce blocking with buffered channels](#)

## Y

YAML, [Working with JSON](#), [Working with YAML-Making your configuration reloadable](#), [Viper: The Swiss Army Knife of Configuration Packages](#)

yaml.Marshal function, [Encoding YAML-Struct field tags for YAML](#)

yaml.Unmarshal function, [Decoding YAML](#)

## Z

Zap logging, [The Zap Logging Package-Using dynamic sampling in Zap](#)

zero values, [Zero Values](#)



## About the Author

**Matthew A. Titmus** is a veteran of the software development industry. Since teaching himself to build virtual worlds in LPC, he's earned a surprisingly relevant degree in molecular biology, written tools to analyze terabyte-sized datasets at a high energy physics laboratory, developed an early web development framework from scratch, wielded distributed computing techniques to analyze cancer genomes, and pioneered machine learning techniques for linked data.

He was an early adopter and advocate of both cloud native technologies in general and the Go language in particular. For the past four years he has specialized in helping companies migrate monolithic applications into a containerized, cloud native world, allowing them to transform the way their services are developed, deployed, and managed. He is passionate about what it takes to make a system production quality, and has spent a lot of time thinking about and implementing strategies for observing and orchestrating distributed systems.

Matthew lives on Long Island with the world's most patient woman, to whom he is lucky to be married, and the world's most adorable boy, by whom he is lucky to be called "Dad."



## Colophon

The animal on the cover of *Cloud Native Go* is a member of the tuco-tuco family (*Ctenomyidae*). These neotropical rodents can be found living in excavated burrows across the southern half of South America.

The name “tuco-tuco” refers to a wide range of species. In general, these rodents have heavily built bodies with powerful short legs and well-developed claws. They have large heads but small ears, and though they spend up to 90% of their time underground, their eyes are relatively large compared to other burrowing rodents. The color and texture of the tuco-tucos’ fur varies depending on the species, but in general, their fur is fairly thick. Their tails are short and not particularly furry.

Tuco-tucos live in tunnel systems—which are often extensive and complicated—that they dig into sandy and/or loamy soil. These networks often include separate chambers for nesting and food storage. They have undergone a variety of morphological adaptations that help them create and thrive in these underground environments, including an improved sense of smell, which helps them orient themselves in the tunnels. They employ both scratch-digging and skull-tooth excavation when creating their burrows.

The diet of the tuco-tucos consists primarily of roots, stems, and grasses. Today, tuco-tucos are viewed as agricultural pests, but in pre-European South America they were an important foodsource for indigenous peoples, particularly in Tierra del Fuego. Today, their conservation status is contingent upon species and geographic location. Many species fall into the “Least Concern” category, while others are considered “Endangered.” Many of the animals on O’Reilly covers are endangered; all of them are important to the world.

The cover illustration is by Karen Montgomery, based on a black and white engraving from *English Cyclopedia*