Extracted from:

Distributed Services with Go

Your Guide to Reliable, Scalable, and Maintainable Systems

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Distributed Services with Go

Your Guide to Reliable, Scalable, and Maintainable Systems

Travis Jeffery

edited by Dawn Schanafelt and Katharine Dvorak

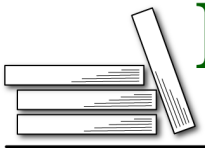
Distributed Services with Go

Your Guide to Reliable, Scalable, and Maintainable Systems

Travis Jeffery

The Pragmatic Bookshelf

Raleigh, North Carolina



For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Dawn Schanafelt and Katharine Dvorak
Copy Editor: L. Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-760-7
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—March 2021

Introduction

Go has become the most popular language for building distributed services, as shown by projects like Docker, Etcd, Vault, CockroachDB, Prometheus, and Kubernetes. Despite the number of prominent projects such as these, however, there's no resource that teaches you why or how you can extend these projects or build your own.

Where do you begin if you want to build a distributed service?

When I began learning how to build distributed services, I found the existing resources to be of two extremes:

- Concrete code—distributed services are large, complex projects, and the prominent ones have had teams working on them for years. The layout of these projects, their technical debt, and their spaghetti code bury the ideas you're interested in, which means you have to dig them out. At best, learning from code is inefficient. Plus there's the risk that you may uncover outdated and irrelevant techniques that you're better off avoiding in your own projects.
- Abstract papers and books—papers and books like *Designing Data-Intensive Applications* by Martin Kleppmann[1] describe how the data structures and algorithms behind distributed services work but cover them as discrete ideas, which means you're left on your own to connect them before you can apply them in a project.

These two extremes leave a chasm for you to cross. I wanted a resource that held my hand and taught me how to build a distributed service—a resource that explained the big ideas behind distributed services and then showed me how to make something of them.

I wrote this book to be that resource. Read this book, and you'll be able to build your own distributed services and contribute to existing ones.

1. https://www.oreilly.com/library/view/designing-data-intensive-applications/9781491903063

Who This Book Is For

This book is for intermediate to advanced developers who want to learn how to build distributed services. I've geared the book toward Go programmers, and prior Go experience will help, but you don't have to be an expert. This book shows you how to build distributed services, and the concepts are the same regardless of what language you use. So if you're writing distributed services in Go, you can take full advantage of this book; if not, you can apply the ideas I present here in any language.

This book's code is compatible with Go 1.13+.

What's in This Book

We will design, develop, and deploy a distributed service to explore what Go can do. We'll develop and deploy the service in layers: from the bare essentials of storage handling, to the networking of a client and server, to distributing server instances, deployment, and testing. I divided this book into four parts that parallel those layers. (Don't worry if you aren't familiar with the technologies I mention next—I explain them in the relevant chapters.)

Part I — Get Started

We'll begin with the basic elements: building our project's storage layer and defining its data structures.

In Chapter 1, Let's Go, on page ?, we'll kick off our project by building a simple JSON over HTTP commit log service.

In Chapter 2, Structure Data with Protocol Buffers, on page ?, we'll set up our protobufs, generate our data structures, and set up automation to quickly generate our code as we make changes.

In Chapter 3, Write a Log Package, on page ?, we'll build a commit log library that'll serve as the heart of our service, storing and looking up data.

Part II — Network

This part is where we'll make our service work over a network.

In Chapter 4, Serve Requests with gRPC, on page ?, we'll set up gRPC, define our server and client APIs in protobuf, and build our client and server.

In Chapter 5, Secure Your Services, on page ?, we'll make our connections secure by authenticating our server with SSL/TLS to encrypt data exchanged between client and server and by authenticating requests with access tokens.

In Chapter 6, Observe Your Systems, on page ?, we'll make our service observable by adding logs, metrics, and tracing.

Part III — Distribute

In this part we'll make our service distributed—highly available, resilient, and scalable.

In Chapter 7, Server-to-Server Service Discovery, on page ?, we'll build discovery into our service to make server instances aware of each other.

In Chapter 8, Coordinate Your Services with Consensus, on page ?, we'll add consensus to coordinate the efforts of our servers and turn them into a cluster.

In Chapter 9, Discover Servers and Load Balance from the Client, on page ?, we'll code discovery in our gRPC clients so they discover and connect to the servers with client-side load balancing.

Part IV — Deploy

Here's where we'll deploy our service and make it live.

In Chapter 10, Deploy Applications with Kubernetes Locally, on page ?, we'll set up Kubernetes locally and run a cluster on your local machine. And we'll prepare to deploy to the cloud.

In Chapter 11, Deploy Applications with Kubernetes to the Cloud, on page ?, we'll create a Kubernetes cluster on Google Cloud's Kubernetes Engine and deploy our service to the cloud so that people on the Internet can use it.

If you plan on building the project as you read (which is a great idea), read the parts in order so that your code works. It's also fine to skip around in the book as well; the ideas we'll explore in each chapter have value on their own.

Online Resources

The code we'll develop is available on the Pragmatic Bookshelf website: https://pragprog.com/book/tjgo. You'll also find an errata-submission form there for you to ask questions, report any problems with the text, or make suggestions for future versions of this book.

Let's get *Go*ing!