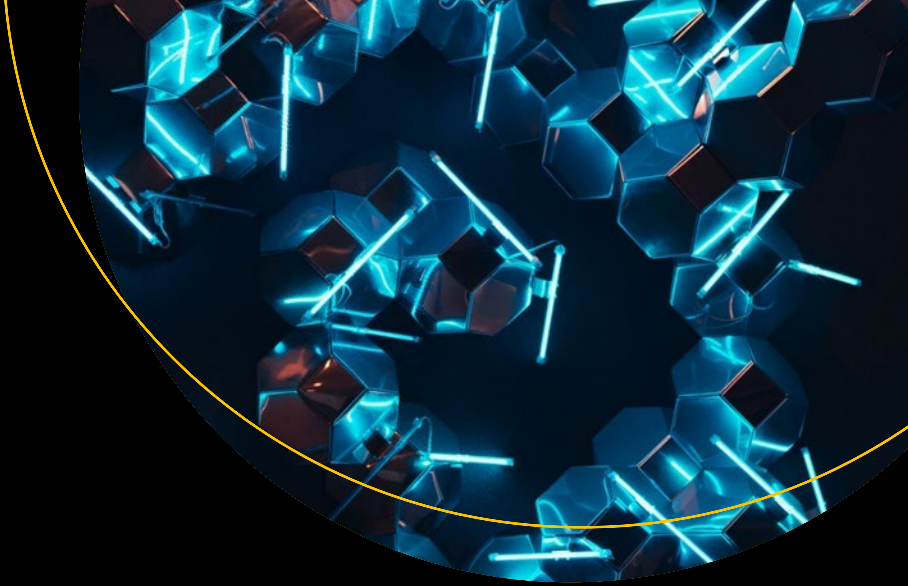

Practical CockroachDB

Building Fault-Tolerant Distributed SQL Databases

—

Rob Reid

Apress®

Practical CockroachDB

Building Fault-Tolerant Distributed SQL Databases

Rob Reid

Apress®

Practical CockroachDB: Building Fault-Tolerant Distributed SQL Databases

Rob Reid
Liss, Hampshire, UK

ISBN-13 (pbk): 978-1-4842-8223-6 ISBN-13 (electronic): 978-1-4842-8224-3
https://doi.org/10.1007/978-1-4842-8224-3

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover image photo by Maximalfocus on Unsplash

For Emily, Ruby, and our little one on the way.
Thank you for giving me the space and patience
I needed to put this wonderful project together.

Table of Contents

About the Author

Rob Reid is a software developer from London, England. In his career, he has written back-end, front-end, and messaging software for the police, travel, finance, commodity, sports betting, telecom, retail, and aerospace industries. He is an avid user of CockroachDB and has worked with the Cockroach Labs team in recent years to promote the database and embed it into development teams in the United States and the UK.

About the Technical Reviewer

Fernando Ipar has been working on and with open source databases since 2000, focusing on performance, scaling, and high availability. He currently works as a Database Reliability Engineer at Life360. Before that, he has worked at Perceptyx, Pythian, and Percona, among other places. When not working, Fernando enjoys going to plant nurseries with his wife and playing music with their children while being a good service employee for the family's cat.

Acknowledgments

I'm incredibly grateful to the following people. Their contributions to this book have been invaluable to me.

Kai Niemi (Solutions Engineer (EMEA) at Cockroach Labs) – I met Kai when he was a customer of Cockroach Labs and have witnessed him transition from being a CockroachDB expert at one company to an expert the global CockroachDB community can be grateful to have.

Daniel Holt (Director, Sales Engineering, International (EMEA and APAC), at Cockroach Labs) – I worked very closely with Daniel from the moment he joined Cockroach Labs and have often marvelled at his comprehensive knowledge of the database.

Katarina Vetrakova (Privacy Programme Manager at GoCardless) – Katarina is quite possibly the most enthusiastic data privacy specialist you could ever hope to meet. She's completely dedicated to the art, and since working with her at Lush, her passion and knowledge have been inspiring to me.

Jonathan Gennick (Assistant Editorial Director of Databases at Apress) – I'd like to thank Jonathan Gennick for approaching me to write this book. Without him, this amazing (and terrifying) opportunity wouldn't have found me. He has been amazing throughout the process of writing this book, and his patient knowledge sharing allowed this first-time author to really find his feet and enjoyment in writing.

The Cockroach Labs team – The Cockroach Labs team is among the smartest people I've ever met. They're incredibly dedicated to their database and its customers and are a big reason for my affection toward CockroachDB. I'd like to thank the following people from Cockroach Labs (past and present) for their help, inspiration, hospitality, and friendship: Jim Walker, Jeff Miller, Carolyn Parrish, Jordan Lewis, Bram Gruneir, Kai Niemi, Daniel Holt, Glenn Fawcett, Tim Veil, Jessica Edwards, Dan Kelly, Lakshmi Kannan, Spencer Kimball, Peter Mattis, Ben Darnell, Nate Stewart, Jesse Seldess, Andy Woods, Meagan Goldman, Megan Mueller, Andrew Deally, Isaac Wong, Vincent Giacomazza, Maria Toft, Tom Hannon, Mikael Austin, Eric Goldstein, Amruta Ranade, Armen Kopoyan, Robert Lee, Charles Sutton, Kevin Maro, James Weitzman, and anyone I've failed to mention.

Introduction

Every so often, the technology community is blessed with truly disruptive technology. We've seen the likes of Kubernetes for orchestration, Kafka for streaming, gRPC for Remote Procedure Call, and Terraform for infrastructure. CockroachDB does what these technologies have done for their respective use cases; it's a game changer for data.

I first discovered CockroachDB in 2016, where I used it to create rapid prototypes during company Hackathons at my then employer. It immediately felt familiar and as if it had been designed for a developer to build reliable and scalable software without an army of database specialists to help them.

In this book, I'll share my excitement for this database and the experience I've gained from using it for many different use cases.

Who Should Read This Book

This book is for developers, database specialists, and enterprise owners. So whether you're in a position of convincing people to use CockroachDB or you're looking for a tool to complement your enterprise tech stack, there's something in this book for you.

You don't need to have existing knowledge of CockroachDB, as we'll start with the basics and quickly ramp up to real-world examples. Any experience with relational databases (especially Postgres) will be beneficial but is not required.

Navigating This Book

This book is both an introduction and a reference to CockroachDB. It starts with the "why" of CockroachDB – why it was created and what problems it solves. It then dives into the "what" – the database's data types, constructs, and fundamentals. Finally, it covers the "how" – how you can use what you've learned to solve real-world scaling, safety, and performance challenges.

The book aims to remain practical, so it hovers above the database's internal details. To continue your journey, I recommend reading the excellent documentation and blog posts available on the Cockroach Labs website: www.cockroachlabs.com.

Using Code Examples

Code examples can be found in GitHub and are arranged in chapters to help you find what you're looking for as you make your way through the book.

https://github.com/codingconcepts/practical-cockroachdb

Code is self-contained to a chapter, meaning you won't have to read the whole book to get something working. In some cases, code examples are split across adjoining code blocks, but this will be highlighted.

I execute all of the code examples against version v21.1.7 of CockroachDB, which is the current stable version of the database at the time of writing.

Contacts

CockroachDB
info@cockroachlabs.com
53 W 23rd Street
8th Floor
New York, NY
10010
www.cockroachlabs.com
https://forum.cockroachlabs.com

Rob Reid
hello@robreid.io
https://robreid.io
https://twitter.com/robreid_io
https://github.com/codingconcepts
www.linkedin.com/in/rob-reid

CHAPTER 1

The Reason for CockroachDB

Databases are a critical part of almost every stateful system. However, correctly running them can be challenging, especially where price, availability, and shifting international data privacy regulations are concerned. CockroachDB makes navigating this tricky landscape not only easier but enjoyable.

In this chapter, we'll explore the *why* of CockroachDB – why it came to be, and why you might want to consider using it for your data.

What Is CockroachDB?

CockroachDB is a cloud-native Relational Database Management System (RDBMS). It falls into the class of "NewSQL" or "DistSQL" (Distributed SQL) databases, which aim to provide the scalability of NoSQL databases while giving users the experience and features of a traditional SQL database. It is wire-compatible with Postgres, which means you can connect to a CockroachDB cluster with most Postgres tools and drivers. To find out which drivers are available, visit `www.cockroachlabs.com/docs/stable/install-client-drivers.html`.

On paper, CockroachDB is "CP," which, in terms of the CAP Theorem, means it favors data **C**onsistency over **A**vailability in the face of network **P**artitions. In reality, however, CockroachDB – as its name suggests – has been built with availability in mind as well. In practice, if you were to lose the majority of replicas in a CockroachDB cluster, rather than compromising the integrity of your data, requests will block until the nodes have recovered. Sizing your cluster to ensure CockroachDB can continue to function in the event of lost nodes is, therefore, an essential constituent to availability.

R. Reid, *Practical CockroachDB*, https://doi.org/10.1007/978-1-4842-8224-3_1

Cockroach Labs – the company behind CockroachDB – was founded in 2015 by Spencer Kimball, Peter Mattis, and Ben Darnell. Inspired by Google's Spanner database, the team sought to bring Spanner's level of scalability to the masses, removing Spanner's dependency on atomic clocks and writing in the performant, modern, and increasingly ubiquitous programming language, Go.

It gives you the freedom to create multiregion databases without the need for a database department. Importantly, it's also a safety net, making it harder to create that multiregion database incorrectly.

CockroachDB's Architecture

CockroachDB uses the Raft Consensus algorithm to give users a multiactive system, which means every node is the same. There are no special "active," "leader," or "write" nodes. Every node can receive a portion of read and write requests for different shards (or "ranges") of data, helping CockroachDB to scale horizontally.

Under the hood, CockroachDB is a distributed key/value store. But thanks to its layered architecture of SQL > Transactions > Distribution > Replication > Storage, you can interact with data as richly as you would in any other RDBMS.

What Does CockroachDB Solve?

Creating a new database is relatively easy, regardless of the technology. You download a binary, pull a Docker image, or start a subscription, and away you go. On the other hand, you typically install a traditional database on a single machine that will need to be vertically scaled, sharded, or have read replicas added when outgrown. All of these needs pose complexity, availability, and consistency challenges.

With CockroachDB, the challenges of complexity and reliability shift to the database. It takes care of distribution, scaling, and replication, leaving you to focus on your data. Generally speaking, if a cluster requires more capacity, you simply add a node to that cluster. CockroachDB will automatically rebalance the data evenly across all nodes.

Figure 1-1 depicts a cluster of three nodes, each with a capacity of 100GB and each with a 40GB share of 120GB total data.

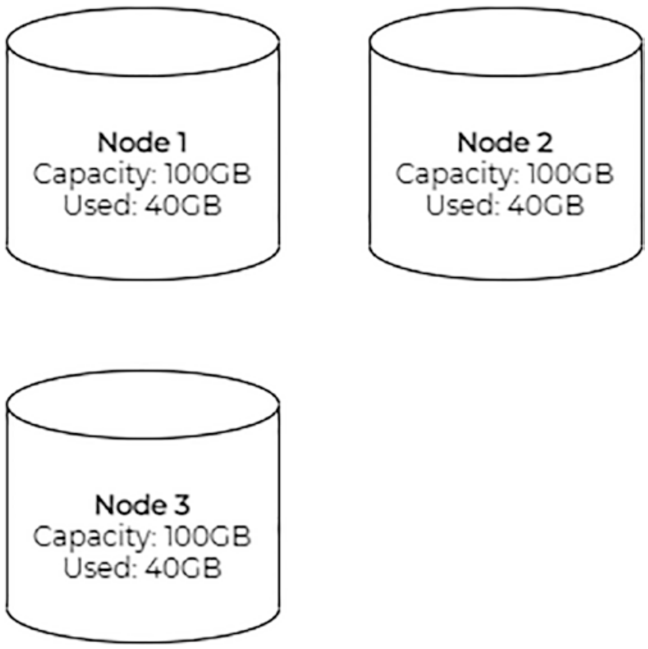

Figure 1-1. *A three-node cluster*

In Figure 1-2, you'll see that when we add a node, data in the cluster will be rebalanced across all four nodes, resulting in shares of 30GB per node of the 120GB total.

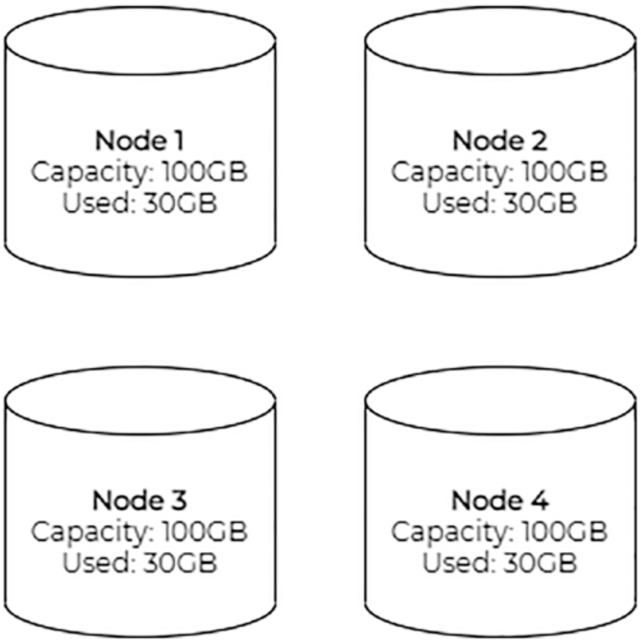

Figure 1-2. *The same cluster, rebalanced across four nodes*

Who Is CockroachDB For?

CockroachDB is an excellent choice for anyone needing a scalable, reliable, and distributed Online Transaction Processing (OLTP) database. Built with modern, cloud-based infrastructure and workloads in mind, it will be at home in the cloud, on-premises, or a hybrid of the two.

It's also a great choice if you're unsure of your database storage requirements. You can start small, expanding your cluster node by node as your data grows. CockroachDB will ensure that your data is distributed evenly and safely across the growing cluster.

If you are familiar with RDBMSs, CockroachDB will feel familiar to you. For Postgres users, you may even find that CockroachDB can be a drop-in replacement for your database.

As you'll see shortly, CockroachDB can be installed with minimal fuss whether you want to create a local development database or a distributed production database.

CockroachDB is trusted to run critical, globally distributed workflows by some of the world's largest tech, finance, travel, retail, telecoms, video streaming, and gaming companies. And its position on `https://db-engines.com/en/ranking` continues to improve. It is now more popular than Google's Spanner, the database that inspired it.

CHAPTER 2

Installing CockroachDB

One of CockroachDB's many strengths is the ease with which you can install it. In this chapter, we'll explore CockroachDB's licensing model and the various ways you can install it.

Licensing

We'll start with a tour of CockroachDB's licensing options to familiarize ourselves with what each option provides. There are both free and paid-for models, and your choice of model will depend on your requirements and the features available in each.

Free

There are two free options available. One is an on-premises installation of the CockroachDB Core functionality, and the other is a cloud offering:

- **CockroachDB Core** – Build and elastically scale a distributed database with many of the most critical features as standard; a good option if you don't need some of CockroachDB's more advanced features. You would need to install this option onto either cloud-based or on-premises hardware.
- **CockroachDB Serverless (Free Tier)** – Sign up and create a free CockroachDB cluster in the cloud. At the time of writing, this cloud option provides similar but slightly more limited functionality to CockroachDB Core, limiting users to a multitenant, single-node machine with 1 vCPU and 5GB of storage.

© Rob Reid 2022
R. Reid, *Practical CockroachDB*, https://doi.org/10.1007/978-1-4842-8224-3_2

Paid For

To unlock everything that CockroachDB has to offer (and to support Cockroach Labs in continuing to build the database into the future), the following paid-for models are available:

- **CockroachDB Self-Hosted** – The complete CockroachDB offering for cloud-based or on-premises installations. Sporting advanced features such as geo and archival partitioning, incremental backups, encryption at rest, change data capture, and direct support from Cockroach Labs.
- **CockroachDB Dedicated** – The complete CockroachDB offering in the cloud. In addition to CockroachDB Enterprise, it provides SLAs and cloud-specific options like VPC peering.

A decision to use any of the aforementioned licenses will depend on your requirements. In each of the following installation options, I'll provide a recap of the licensing model to help you decide which best matches your needs.

CockroachDB Core

Beginning in version 19.2, CockroachDB Core is subject to multiple licenses. Some functionality retains the original Apache 2.0 license, meaning it is fully Open Source. Other features are subject to a Cockroach Community License, which protects Cockroach Labs from companies using their code to build products that do not benefit Cockroach Labs.

In short, if you plan on using CockroachDB Core's free features to power your own databases and do not intend to sell CockroachDB as a service to others, CockroachDB Core is a good choice for you.

Local Installation

In this section, I'll show you how to install CockroachDB on your local machine. For each installation method, we'll run CockroachDB with insecure configuration for brevity for each installation method, which means **no authentication or encryption**. This is acceptable for a local development database but not for anything else. Once we've installed and tested local insecure deployments, we'll move on to some real-world secure implementations.

Binary Install

You can install the cockroach binary in several ways, depending on your operating system.

For Linux users, run the following commands to get started:

```
$ curl https://binaries.cockroachdb.com/cockroach-v21.1.7.linux-amd64.tgz
| tar -xz
```

```
$ sudo cp cockroach-v21.1.7.linux-amd64/cockroach /usr/local/bin/
```

For Mac users, run the following command to get started:

```
$ brew install cockroachdb/tap/cockroach
```

For Windows 10 users, run the following commands to get started:

```
$ curl -o cockroach-v21.1.7.windows-6.2-amd64.zip
https://binaries.cockroachdb.com/cockroach-v21.1.7.windows-6.2-amd64.zip
```

```
$ powershell.exe -NoP -NonI -Command "Expand-Archive '.\cockroach-
v21.1.7.windows-6.2-amd64.zip' '.'"
```

Either change directory (cd) into the cockroach-v21.1.7.windows-6.2-amd64 directory and invoke the cockroach executable when you need it, or add its containing directory to your PATH and invoke the cockroach executable from any directory.

The cockroach executable is now available, and you can start a local instance of CockroachDB using the following command:

```
$ cockroach start-single-node --insecure --listen-addr=localhost
```

We've just started a single node, insecure cluster, listening on localhost with the default 26527 port for SQL connections and the 8080 port for HTTP connections.

To test that CockroachDB is up and running, use the cockroach sql command to enter the CockroachDB SQL shell. Note that the --host argument can be omitted, as the default value assumes a locally running node using the default port:

```
$ cockroach sql --insecure
#
# Welcome to the CockroachDB SQL shell.
# All statements must be terminated by a semicolon.
# To exit, type: \q.
#
# Client version: CockroachDB CCL v21.1.5 (x86_64-w64-mingw32, built
2021/07/02 04:03:50, go1.15.11)
# Server version: CockroachDB CCL v21.1.7 (x86_64-w64-mingw32, built
2021/08/09 18:01:51, go1.15.14)
# Cluster ID: 66424e6d-680e-4b26-8d8f-e029967b00c4
#
# Enter \? for a brief introduction.
#
root@:26257/defaultdb>
```

Docker Install

To install CockroachDB with Docker, first, make sure you have Docker installed on your machine. If you don't have Docker installed, visit the Docker installation website https://docs.docker.com/get-docker for instructions.

You can start an instance of CockroachDB in Docker with the following command:

```
$ docker run \
    --rm -it \
    --name=cockroach \
    -p 26257:26257 \
    -p 8080:8080 \
    cockroachdb/cockroach:v21.1.7 start-single-node \
        --insecure
```

This command will pull the CockroachDB Docker image and run it with port 26257 for client connections and port 8080 for HTTP connections exposed. The command will block until the process terminates, at which point, the CockroachDB Docker container will be stopped and removed.

To test that CockroachDB is up and running, use the `cockroach sql` command to enter the CockroachDB SQL shell:

```
$ cockroach sql --insecure
#
# Welcome to the CockroachDB SQL shell.
# All statements must be terminated by a semicolon.
# To exit, type: \q.
#
# Client version: CockroachDB CCL v21.1.5 (x86_64-w64-mingw32, built
2021/07/02 04:03:50, go1.15.11)
# Server version: CockroachDB CCL v21.1.7 (x86_64-unknown-linux-gnu, built
2021/08/09 17:55:28, go1.15.14)
# Cluster ID: 0b3ff09c-7bcc-4631-8121-335cfd83b04c
#
# Enter \? for a brief introduction.
#
root@:26257/defaultdb>
```

Kubernetes Install

To install CockroachDB with Kubernetes, make sure you have the following prerequisites installed:

- **A local installation of Kubernetes** – There are plenty of ways you can install Kubernetes on your machine. Here are some of the options available:
    - kind (https://kind.sigs.k8s.io/docs/user/quick-start)
    - minikube (https://minikube.sigs.k8s.io/docs/start)
    - k3s (https://rancher.com/docs/k3s/latest/en/installation)

I will be using kind to create a local Kubernetes cluster, and at the time of writing, this installs Kubernetes version 1.21.

- **kubectl** – The Kubernetes CLI; visit https://kubernetes.io/docs/tasks/tools for installation instructions.

With these prerequisites installed, we're ready to get started.

Being a database, CockroachDB is considered a stateful resource in Kubernetes, so you'll need to give it a Persistent Volume Claim (PVC). PVCs live at the node level, not at the pod level, so your data will be safe if any of your CockroachDB pods get deleted or restarted.

To install a simple CockroachDB cluster in Kubernetes, create the files described in each of the following subsections. If you'd rather avoid handcrafting the following files, you can apply a ready-made Kubernetes manifest from Cockroach Labs here:

```
https://github.com/cockroachdb/cockroach/blob/master/cloud/kubernetes/
cockroachdb-statefulset.yaml
```

1_pod-disruption-budget.yaml

This file creates the PodDisruptionBudget resource for our StatefulSet. A Pod Disruption Budget provides Kubernetes with a tolerance for pod failures against a given application (identified by the selector "cockroachdb"). It ensures that there are never more than `maxUnavailable` pods unavailable in the cluster at any one time. By setting this to 1, we'll prevent Kubernetes from removing more than 1 CockroachDB node during operations like rolling updates, etc.

If you're configuring a cluster of 5 nodes in Kubernetes, consider setting this to 2, as you'll still have a 3-node cluster if 2 of your nodes are temporarily unavailable.

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: cockroachdb-budget
  labels:
    app: cockroachdb
spec:
  selector:
    matchLabels:
      app: cockroachdb
  maxUnavailable: 1
```

Run the following command to create the pod disruption budget:

```
$ kubectl apply -f 1_pod-disruption-budget.yaml
```

2_stateful-set.yaml

Now we're getting into the guts of our Kubernetes configuration. It's time to create our StatefulSet. A StatefulSet is like a deployment that provides additional guarantees around pod scheduling to ensure that pods have a persistent disk.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cockroachdb
spec:
  serviceName: "cockroachdb"
  replicas: 3
  selector:
    matchLabels:
      app: cockroachdb
  template:
    metadata:
      labels:
        app: cockroachdb
    spec:
      affinity:
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: app
                  operator: In
                  values:
                  - cockroachdb
              topologyKey: kubernetes.io/hostname
      containers:
      - name: cockroachdb
        image: cockroachdb/cockroach:v21.1.7
        imagePullPolicy: IfNotPresent
        resources:
          requests:
            cpu: "1"
            memory: "1Gi"
          limits:
            cpu: "1"
            memory: "1Gi"
        ports:
        - containerPort: 26257
          name: grpc
        - containerPort: 8080
          name: http
        readinessProbe:
          httpGet:
            path: "/health?ready=1"
            port: http
          initialDelaySeconds: 10
          periodSeconds: 5
          failureThreshold: 2
        volumeMounts:
        - name: datadir
          mountPath: /cockroach/cockroach-data
        env:
        - name: COCKROACH_CHANNEL
          value: kubernetes-insecure
        - name: GOMAXPROCS
          valueFrom:
            resourceFieldRef:
              resource: limits.cpu
              divisor: "1"
        command:
          - "/bin/bash"
          - "-ecx"
          - exec
```