Chaos Engineering with Go

Building resilient systems through controlled chaos

Kapil Kumar Khandelwal

Mohit Garg

www.bpbonline.com

First Edition 2025

ISBN: 978-93-55519-641

To View Complete BPB Publications Catalogue Scan the QR Code:

www.bpbonline.com

Dedicated to

To my parents
Sh. Mahesh Chand** and **Smt. Rukmani Devi
whose unwavering support and encouragement have been my foundation

*To my wife, **Ankita***
whose love and understanding have been my strength.

*And to my son, **Divit,** who inspires me every day*

With all my love and gratitude
– Kapil Kr. Khandelwal

To my parents
Sh. Dev Raj Garg** and **Smt. Late Nisha Garg
for their unwavering belief in me.

*To my wonderful wife, **Vatika Garg***
whose love and patience have made this journey possible.

And to my children
Avika Garg** and **Yash Garg
who are my greatest source of joy and motivation
– Mohit Garg

About the Authors

- **Kapil Kumar Khandelwal** is a passionate tech and product enthusiast with over a decade of experience. He launched his illustrious career after achieving his B.Tech degree and later earned an executive MBA in Leadership and Finance. He also holds multiple certifications in quality, product, and cloud.

  Kapil has worked in leading service and product firms, contributing to mission-critical products and unlocking the potential of Chaos Engineering in real-world scenarios. His unquenchable curiosity in Chaos Engineering has propelled him to become a highly sought-after authority in the field.

  When he is not immersed in tech challenges, you can find Kapil exploring hiking trails, learning new tools, and building connections within the tech and product community through insightful networking events.

- **Mohit Garg** completed his Bachelor of Technology in Information Technology in 2010, followed by an MBA in Leadership and Finance. With over 13 years of experience in the IT industry, he has contributed to and engaged with leading product and service-based companies. His vast experience has been instrumental in writing this book, providing practical insights and deep understanding of Chaos Engineering.

  Throughout his career, Mohit has remained active in learning new tools and technologies, constantly expanding his skill set and building new connections. This dedication to continuous improvement and networking has kept him at the forefront of industry trends and best practices.

  Outside of his professional life, Mohit enjoys spending quality time with his family. He loves playing with his children, Avika and Yash, exploring music, watching movies, and taking long drives. These hobbies provide him with a refreshing break from work and keep him motivated.

  Mohit's dedication to his profession, combined with his personal interests and family life, has contributed to a well-rounded perspective, making this book a valuable resource for anyone looking to explore Chaos Engineering.

About the Reviewers

- **Daniel Moreira Cardoso** is a seasoned Senior Software Engineer with several years of experience in software development. He has proficiency in a variety of programming languages and technologies, including Typescript, Kotlin, Golang, PostgreSQL, Apache Kafka, Kubernetes, Google Cloud Platform, AWS, Datadog, Next.js, and React. His expertise lies in developing solutions for financial domains, payments, municipal public sectors, and sales platforms, significantly improving the performance, availability, reliability, and resilience of backend systems. In addition to his technical abilities, Daniel has experience leading technical teams and contributing to major projects that enhanced business efficiency and customer satisfaction. He is known for his proactive problem-solving approach and dedication to automating processes and integrating innovative solutions to meet dynamic business and client needs. Notably, Daniel has made significant contributions to optimizing tooling for load tests and implementing Chaos Engineering practices, which have improved the resilience and reliability of the systems he has worked with.

- **Syed** is a pioneer and subject matter expert (SME) in application modernization and cloud computing. He is a holder of multiple industry certifications in Azure, AWS, GCP, Oracle, PMP, and CSM. With more than a decade of expertise in product development, architecture, scalability, and modernization, Syed has held positions with Oracle, the Big 4 technology advisory companies like PwC and Ernst & Young, as well as other contemporary application enterprises. Syed has also published articles in the leading Journal of Cloud Computing. Syed is a skilled and devoted technology management executive who employs more than 15 years of expertise in the design and development of high-demand On Premise / Commercial of the Shelf Software / SaaS solutions. He possesses specialized knowledge in designing and constructing highly scalable and highly available systems. In order to assist them in establishing and managing their businesses by taking control of their frequently cumbersome applications, Syed has provided consulting services to some of the world's top corporations. Syed assists businesses in modernizing their applications by emphasizing scale and availability, aiding cloud migration, DevOps transformations, and risk-based problem identification. Syed offers expert guidance, strategy, and thought leadership to assist in solving application modernization issues. Syed promotes change by collaborating with enterprises of all sizes and executive teams.

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and inspired me throughout the journey of writing this book.

To my family, your unwavering love and encouragement have been my foundation. A special thanks to my parents, whose blessings and support have shaped my path, and to my beloved wife, Ankita, and son, Divit, for their endless patience and understanding.

To my extended family and especially my sisters, your constant support and belief in me have been a source of great motivation.

I am immensely grateful to BPB Publications, whose expertise and dedication have been instrumental in bringing this book to life. Your professional guidance and support have been invaluable.

I would also like to acknowledge the valuable contributions of my colleagues and co-workers during my years in the tech industry. Your teachings and feedback have been crucial to my growth.

A special thank you to my close friend and co-author, Mohit Garg. Your collaboration, insights, and hard work have been pivotal in shaping this book. It has been an incredible journey working alongside you.

Finally, to everyone who has contributed in any way to the creation of this book, thank you. Your support, encouragement, and contributions have made this possible.

- Kapil Kr. Khandelwal

Creating this book has been a remarkable journey, and I am incredibly thankful for the support of many individuals. First, I would like to thank my parents, Sh. Dev Raj Garg and Smt. Late Nisha Garg. Your unwavering love and support have been my guiding light.

To my wonderful wife, Vatika Garg, your endless encouragement and understanding have been vital to this project. To my children, Avika Garg and Yash Garg, your energy and enthusiasm have been a constant source of inspiration.

To my extended family, especially my sister, thank you for your support and encouragement. To my friends, your belief in me has been a driving force throughout this journey

I extend my heartfelt thanks to BPB Publications for providing me with this incredible opportunity. Their professional guidance and trust have been paramount in bringing this book to life. Their support has been instrumental in every step of this journey.

I am also deeply grateful to my close friend and co-author, Kapil Kr. Khandelwal. Your collaboration, hard work, and insights have been essential to this project. Working with you has been an enriching experience, and I value your friendship and partnership immensely.

Lastly, I am incredibly thankful to all those who have supported me along the way.

- Mohit Garg

Preface

Chaos Engineering with Go is a comprehensive guide crafted for both beginners and experienced professionals aiming to enhance system reliability and minimize downtime. This book serves a diverse audience, including **Site Reliability Engineers** (**SREs**), DevOps engineers, tech professionals, and students, offering valuable insights into building resilient systems through controlled chaos.

Readers should have a basic understanding of programming fundamentals, the Go programming language, the **Software Development Life Cycle** (**SDLC**), cloud-native concepts, distributed systems, and DevOps principles. For those with prior experience in Chaos Engineering or Go programming, this book provides advanced topics and practical applications to deepen their knowledge.

Discover innovative techniques for chaos fault injection, explore the latest chaos testing tools, and design effective chaos experiments to strengthen your infrastructure against unexpected failures. Learn about resilience patterns, fault tolerance strategies, and security Chaos Engineering. Real-world case studies demonstrate the practical application of these concepts in various scenarios.

With a focus on best practices and lessons learned, this book equips you with the knowledge and tools needed to embrace chaos, ensuring robust and reliable systems in an ever-evolving digital landscape. Whether you are a seasoned engineer or a curious beginner, Chaos Engineering with Go is your indispensable companion on the journey to engineering resilience.

Chapter 1: Exploring the Essence of Chaos Engineering - This chapter is an essential part of the book Chaos Engineering with Go as it lays the groundwork for those who are unfamiliar with the topic. Using relevant examples, it offers a crucial comprehension of the underlying concepts, evolution, and advantages. It explores the mentality and cultural elements required for firms to successfully adopt Chaos Engineering.

Chapter 2: Chaos Engineering Concepts -This chapter discusses topics such as principles of Chaos Engineering, and how it differs from functional and non-functional testing. Understanding these principles allows readers to execute Chaos Engineering activities in a methodical and structured manner.

Chapter 3: Revision with Go - This chapter introduces the Go programming language, which is essential for implementing Chaos Engineering with Go. It covers Go's syntax, principles, and key ideas to ensure readers have a solid understanding before exploring specific Chaos Engineering topics.

Managing packages and dependencies in Go is crucial for using existing Chaos Engineering tools and creating innovative solutions. Common tasks in Chaos Engineering include simulating concurrent events, injecting errors, and monitoring system reactions. Understanding concurrency and goroutines allows readers to leverage Go's robust concurrency features for effective chaos experiments. Additionally, mastering Go's error management techniques is vital for properly handling errors in Chaos Engineering experiments.

Chapter 4: Fault Tolerance and Resilience Patterns - In this chapter, you will understand fault tolerance in microservices architecture and its crucial role in maintaining system reliability. This chapter outlines the fundamental principles for implementing fault tolerance and laying the groundwork for resilient systems. Explore key resilience patterns such as timeouts, retries, and circuit breakers, examining how they mitigate failures. Learn effective strategies for testing fault tolerance to ensure robustness in distributed systems. Through a compelling case study of a distributed system built in Go, discover third-party libraries for fault injection and gain valuable lessons and best practices for constructing resilient distributed systems in Go.

Chapter 5: Chaos Fault Injection Techniques - This chapter focuses on the practical application of Chaos Engineering in real-world scenarios. It provides detailed guidance on using Go to introduce faults and simulate various stress conditions. Topics include a range of failure scenarios such as process crashes, disk space issues, high CPU usage, out-of-memory situations, component failures, and load balancer changes. Readers will learn how to replicate these failures and monitor system responses effectively.

The chapter equips readers with the knowledge and techniques to create automated chaos experiments using Go, covering topics like error automation and dependency testing. It also serves as a guide to leveraging Go's libraries, tools, and frameworks for successful fault injection and testing system resilience.

Chapter 6: Chaos Testing Tools - This chapter offers an in-depth exploration of essential chaos testing tools, including Chaos Monkey, Toxiproxy, Chaos Toolkit, and Chaos Mesh. Readers gain insights into their functionalities and applications in fault injection testing. Step-by-step instructions for installing Chaos Mesh on Ubuntu facilitate practical implementation. Explore the specific faults that can be injected via Chaos Mesh, such as

Pod and DNS failures, along with stress experiments. Learn about Chaos Mesh's limitations and emerging industry trends in chaos engineering tools like Gremlin and Litmus Chaos, helping readers select the right tools for their chaos engineering endeavors.

Chapter 7: Chaos Experiment Design - This chapter stresses how crucial it is to approach the planning and execution of experiments in chaos engineering in a logical and systematic manner. In order to successfully prepare, carry out, and analyze chaos experiments, it offers readers a structured framework. It aids readers in understanding the general objectives of each chaotic experiment as well as the particular system characteristics that researchers want to test, failure scenarios that they hope to model, and specific failure scenarios that they want to simulate. To evaluate the system's behavior during chaos experiments, this chapter assists readers in defining relevant metrics and observability measurements.

Chapter 8: Chaos with Emerging Tech Stack - Explore cutting-edge chaos engineering techniques in this chapter, featuring the powerful Azure Chaos Studio—an indispensable tool within the Azure ecosystem. Uncover the intricacies of high availability testing through chaos engineering methodologies, ensuring your systems are robust and resilient. Dive into the realm of Artificial Intelligence, discovering how AI elevates chaos engineering to new heights, introducing automated scenario generation and adaptive testing. This chapter provides a succinct guide to leveraging Azure Chaos Studio and AWS fault simulator, mastering high availability testing, and seamlessly integrating AI into chaos engineering for unparalleled system resilience and innovation.

Chapter 9: Essence of Observability in Distributed System - In this chapter, we will explore the essence of observability and its crucial role in understanding distributed systems. It is a foundational guide for engineers, covering key components and implications in modern infrastructures. We will also discuss application monitoring approaches and tools like Dynatrace, emphasizing the importance of logging with the ELK Stack and best practices for Go. Tracing is highlighted, focusing on its relevance and a step-by-step guide to implementing OpenTelemetry. By the end, readers will grasp observability principles and practical tools essential for building resilient, observable distributed systems.

Chapter 10: Observability in Chaos Engineering - This chapter emphasizes the vital role of observability in chaos engineering. It explains how observability provides real-time visibility into system performance during chaos experiments, identifying vulnerabilities before they escalate. Readers will explore various observability tools tailored for chaos engineering and learn to select the most suitable ones for their needs. Key metrics are categorized into resilience, infrastructure, and application metrics, highlighting their importance in evaluating system resilience. By effectively logging and tracing chaos experiments, engineers can reconstruct system behavior and derive actionable insights, equipping them to build resilient systems through controlled chaos.

Chapter 11: Security Chaos Engineering Overview - Explore Security Chaos Engineering (SCE) essentials, from foundational principles to practical application. Learn how to apply chaos engineering techniques to fortify security resilience, utilizing specialized tools tailored for SCE. Join us as we navigate the intersection of Chaos Engineering and cybersecurity, forging stronger defenses through deliberate disruption.

Chapter 12: Case Studies: Chaos Engineering in Action - This chapter presents real-world case studies demonstrating how Chaos Engineering is applied across various sectors to enhance system resilience and reliability. These examples serve to inspire readers to implement Chaos Engineering techniques within their organizations, highlighting successful implementations in specific domains.

Chapter 13: Best Practices and Lessons Learned - This chapter consolidates insights from real-world chaos engineering experiences, offering valuable information for readers. It analyzes common pitfalls and mistakes to help avoid them on the path to resilient systems. The chapter discusses effectively integrating Chaos Engineering into the Software Development Life Cycle (SDLC) as a preventive strategy.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/oxg6aji

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Chaos-Engineering-with-Go.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

CHAPTER 1
Exploring the Essence of Chaos Engineering

Introduction

The first chapter of this book will cover groundwork for those who are unfamiliar with the topic of Chaos Engineering. Using relevant examples, it offers a crucial comprehension of the underlying concepts, evolution, and advantages.

This chapter outlines the fundamental concepts underlying Chaos Engineering, such as definition, aims, and benefits, to ensure that the readers comprehend its purpose. It explores the mentality and cultural elements required for an organization to successfully adopt Chaos Engineering.

Structure

In this chapter, we will discuss the following topics:

- Overview of Chaos Engineering
- Benefits of Chaos Engineering
- Challenges and considerations
- History of Chaos Engineering
- Importance of resilience in distributed system

- Elevating system reliability
- Chaos Engineering examples

Objectives

The objective of the chapter titled *Exploring the Essence of Chaos Engineering* is to provide readers with an in-depth understanding of the fundamental principles, concepts, and practices that constitute Chaos Engineering. It aims to introduce readers to the core elements involved in Chaos Engineering and shed light on its significance in improving system resilience and reliability. By the end of the chapter, readers should have a clear grasp of what Chaos Engineering entails and how it can be applied to enhance system robustness and user experience.

In the industrial revolution of digitization, the world has undergone a rapid transformation, heavily reliant on technology in almost every aspect of our lives. With the advent of the internet, our global community has become more interconnected than ever before, driving industries towards the online platform. However, this technological shift is still relatively new, and both emerging and developed economies are continuously striving to optimize the infrastructure and ecosystem required to support these online businesses. As a consequence, uncertainties arise, leading to a higher prevalence of failures.

We frequently encounter headlines reporting challenges in accessing mobile and online banking services, instances of bank websites being unavailable or not functioning correctly, and occurrences of *Service Unavailable* messages. Such unpredictability has become a regular occurrence in today's fast-paced digital landscape.

These outages and failures predominantly occur in complex and distributed systems, where multiple components may fail simultaneously, increasing the impact of the problem. Identifying and rectifying these bugs can take anywhere from minutes to hours, depending on the complexities of the system architecture. Consequently, these incidents not only result in revenue losses for the affected companies, but also lose the customer trust as well.

Although these systems are designed to handle individual failures, the complexities of large and chaotic systems pose the risk of cascading failures, leading to severe and extended outages. The challenge lies in enhancing the resilience and reliability of these complex systems to withstand the ever-increasing demands of the digital era.

Standard application testing is not capable of identifying errors in a distributed system. Companies must find more inventive ways to regularly test micro services. Chaos Engineering is a tactic that is gaining notice.

Overview of Chaos Engineering

Chaos Engineering is a specialized field within software engineering that strives to achieve a specific goal to improve system resilience and reliability by intentionally inducing controlled failures in a system. It emerged as a response to the growing complexity of modern distributed systems, where even minor disruptions can have catastrophic consequences.

Understanding the concepts behind Chaos Engineering allows readers to develop a systematic approach to designing and conducting experiments that simulate real-world failures and stress scenarios. This proactive approach is used to identify bottlenecks, performance issues, and scalability limitations; enabling reader to fine-tune systems for optimal performance. It builds confidence in their system's ability to handle unexpected events, and ultimately create more robust and dependable applications.

Chaos Engineering is not only a technical practice; it requires a mind-set shift and a culture of embracing failure as a means to improve system resilience. Chaos Engineering extends its influence across diverse roles and domains, encompassing SRE teams, DevOps engineers, system administrators, and enterprise architects. It involves cross-functional collaboration and effective communication between teams. This promotes collaboration, facilitates knowledge sharing, and aligns the team towards a common goal of building resilient systems.

Chaos Engineering has become an essential practice in modern software engineering to ensure system reliability and resilience. By embracing controlled chaos and learning from failures, organizations can build robust, fault-tolerant systems that provide a more sensible experience for users and withstand the challenges of today's complex computing landscape.

Benefits of Chaos Engineering

Chaos Engineering offers a range of benefits that contribute to the overall reliability, resilience, and customer experience of software systems. Here are some key advantages of adopting Chaos Engineering practices:

- **Early fault detection**: Chaos Engineering enables early detection of potential weaknesses and vulnerabilities within a system, reducing the likelihood of major outages or service disruptions in production.
- **Enhanced resilience**: By subjecting a system to controlled failures, Chaos Engineering helps developers build a more resilient architecture, capable of self-healing from unexpected events.
- **Improved incident response**: Chaos experiments serve as intense training for our incident response team, refining troubleshooting, communication, and recovery skills in a controlled setting. This results in faster, more efficient reactions to

real-world problems, refining important skills like as root cause analysis and collaboration. Measuring response metrics allows you to identify areas for improvement and continuously optimizes your incident response plan to adapt to changing systems.

- **Enhancing trust and better customer service**: Chaos Engineering, by intentionally inducing failures in systems, helps build resilience and trust in services. Chaos experiments expose weaknesses before they become critical issues. This translates to fewer unexpected outages and smoother customer experiences, fostering trust and loyalty. When outages occur, practiced incident response teams can quickly diagnose and resolve issues, minimizing downtime and impact on customers. Proactively communicating potential weaknesses and planned chaos experiments demonstrates transparency and commitment to reliability. This builds trust with customers and fosters a sense of partnership. By minimizing outages, resolving issues quickly, and communicating openly, Chaos Engineering ultimately leads to increased customer satisfaction and loyalty. For instance, Spotify, a music streaming service, uses Chaos Engineering to ensure high availability. This translates to reliable music access for their users, contributing to their high customer satisfaction scores.
- **Cost savings**: Chaos Engineering enables companies to uncover weaknesses in infrastructure, optimize cloud costs, reduce mean time to recovery, prevent data loss, and streamline development processes. For instance, by simulating failures, organizations can fine-tune systems, avoiding costly downtime and ensuring efficient resource allocation, leading to significant financial savings.

By prioritizing the above benefits and leveraging them effectively, companies can build stronger relationships with customers, protect their brand reputation, mitigate risks, drive efficiency and innovation, and attract key talent and partners, all of which contribute to long-term success and profitability.

Challenges and considerations

Implementing Chaos Engineering, while highly beneficial, comes with its own set of challenges and considerations. It is important to be aware of these factors to ensure successful integration and effective practice within an organization. Here are some challenges and considerations to keep in mind:

- **Safety first**: In Chaos Engineering, challenges are intentionally introduced during testing, and occasionally, these challenges might be unnecessary. While Chaos Engineering is beneficial, it should be carried out with caution. Safety measures must be in place to prevent experiments from causing severe disruptions to critical systems.
- **Start small**: Organizations that are new to Chaos Engineering should begin with simple experiments and gradually increase complexity as their understanding and expertise grow.

- **Communication and collaboration**: Chaos Engineering requires close collaboration between developers, operations, and other stakeholders to ensure that all parties are aware of the experiments and potential risks involved.
- **Observability**: One of the most common issues engineers run across while implementing chaotic engineering is their inability to keep track of the observation. When there are gaps in monitoring, the team finds it difficult to understand the entire effects of the issue on the system. The teams are unable to locate the problem's root cause, which does not solve the issue and makes it complicated.
- **Impact analysis**: Experimenters should carefully assess the potential impact of each experiment to avoid unintended consequences and maintain system stability.

History of Chaos Engineering

Beginning in the early 2000s, Chaos Engineering has a long and interconnected history that is closely associated with the emergence of complex distributed systems and the need for improved reliability in technology infrastructure. The concept was pioneered by engineers at Netflix, who faced the challenges of running a vast and rapidly growing online streaming service that required high availability and fault tolerance.

The following key milestones highlight the evolution of Chaos Engineering:

- **Netflix's start**: In the mid-2000s, Netflix started transitioning from a DVD rental service to an online streaming platform. As the scale of their service expanded, they encountered various technical challenges, including outages and system failures. The traditional approaches of testing and monitoring were not enough to ensure the robustness of their distributed systems.

  In 2011, Netflix introduced the Simian Army, a suite of tools designed to create controlled disruptions within their production environment. One of the most notable components was Chaos Monkey, which randomly terminated virtual machine instances. This forced engineering teams to build applications and infrastructure that could withstand such failures.

  The rationale behind *Chaos Monkey*, according to former VP of Product Engineering at Netflix *John Ciancutti*, is that,

If we aren't constantly testing our ability to succeed despite failure, then it isn't likely to work when it matters most–in the event of an unexpected outage.

- **Wider recognition**: Chaos Engineering gained wider recognition in the software engineering community as Netflix shared its experience and success stories at technology conferences and in blog posts. Other companies became interested in applying similar principles to their own systems.