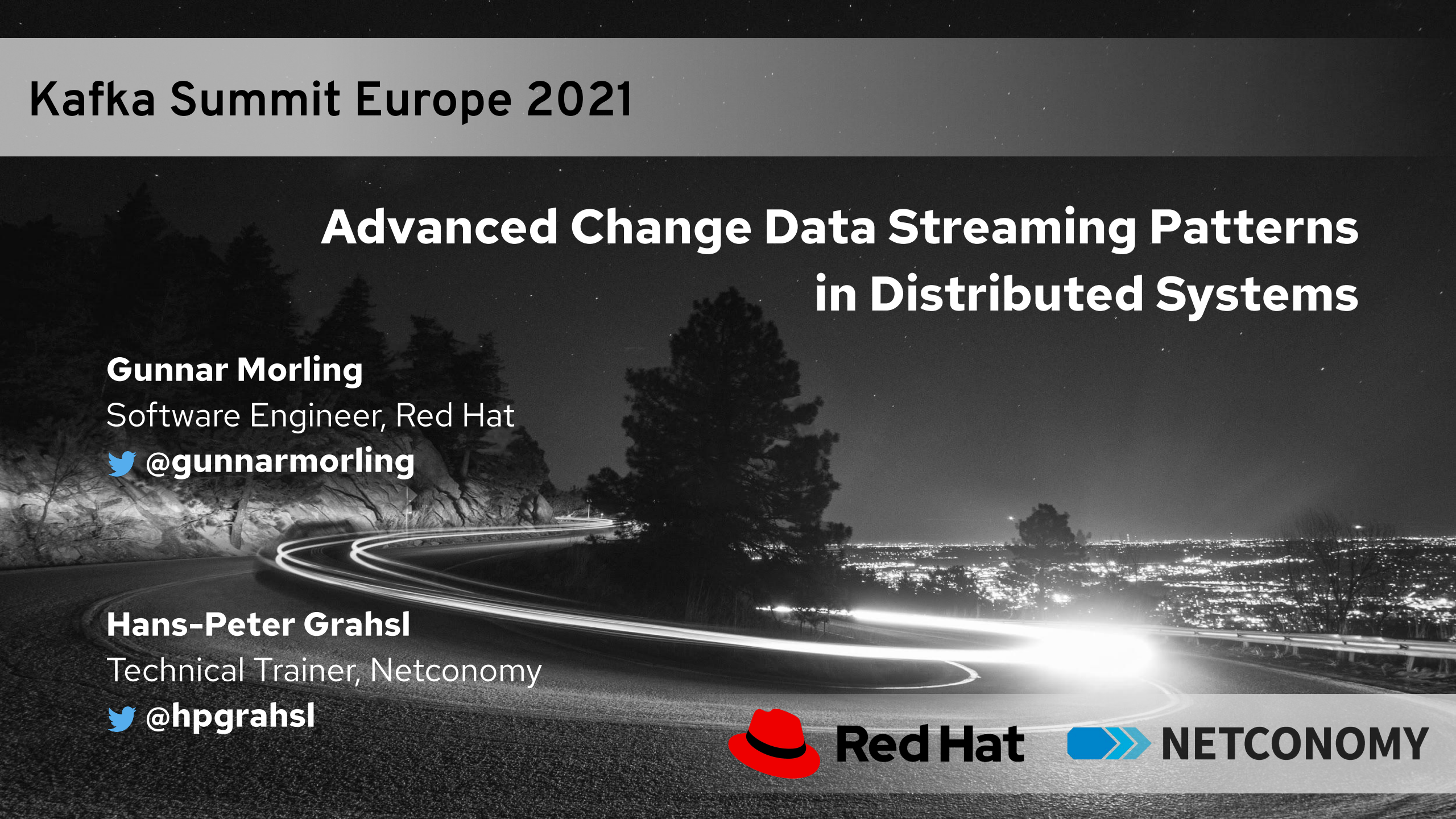
Kafka Summit Europe 2021

Advanced Change Data Streaming Patterns in Distributed Systems

Gunnar Morling
Software Engineer, Red Hat
@gunnarmorling

Hans-Peter Grahsl
Technical Trainer, Netconomy
@hpgrahsl



Gunnar Morling

- Open source software engineer at Red Hat
  - **Debezium**
  - Quarkus
- **Spec Lead** for Bean Validation 2.0
- Java Champion

@gunnarmorling

Hans-Peter Grahsl

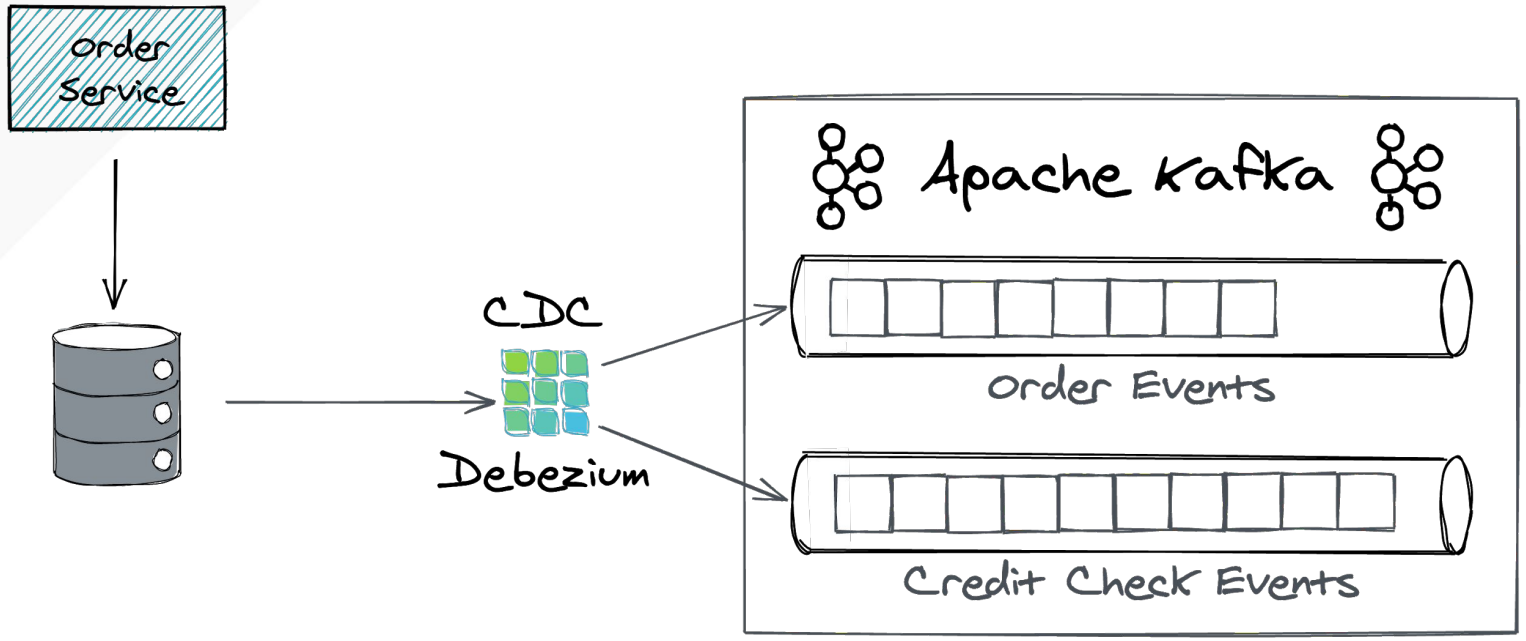
- Technical Trainer at NETCONOMY
- Independent Engineer & Consultant
- Confluent Community Catalyst
- MongoDB Champion

@hpgrahsl

Today's Objective: Learn About...

... implemented using **Change Data Capture**

Debezium — Log-based Change Data Capture

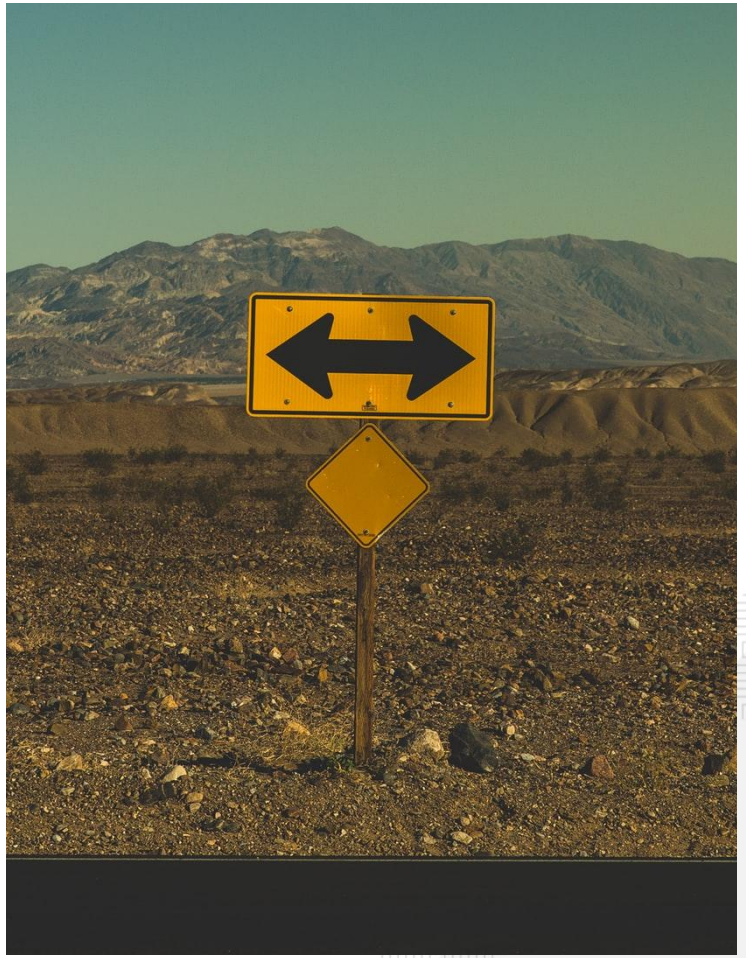


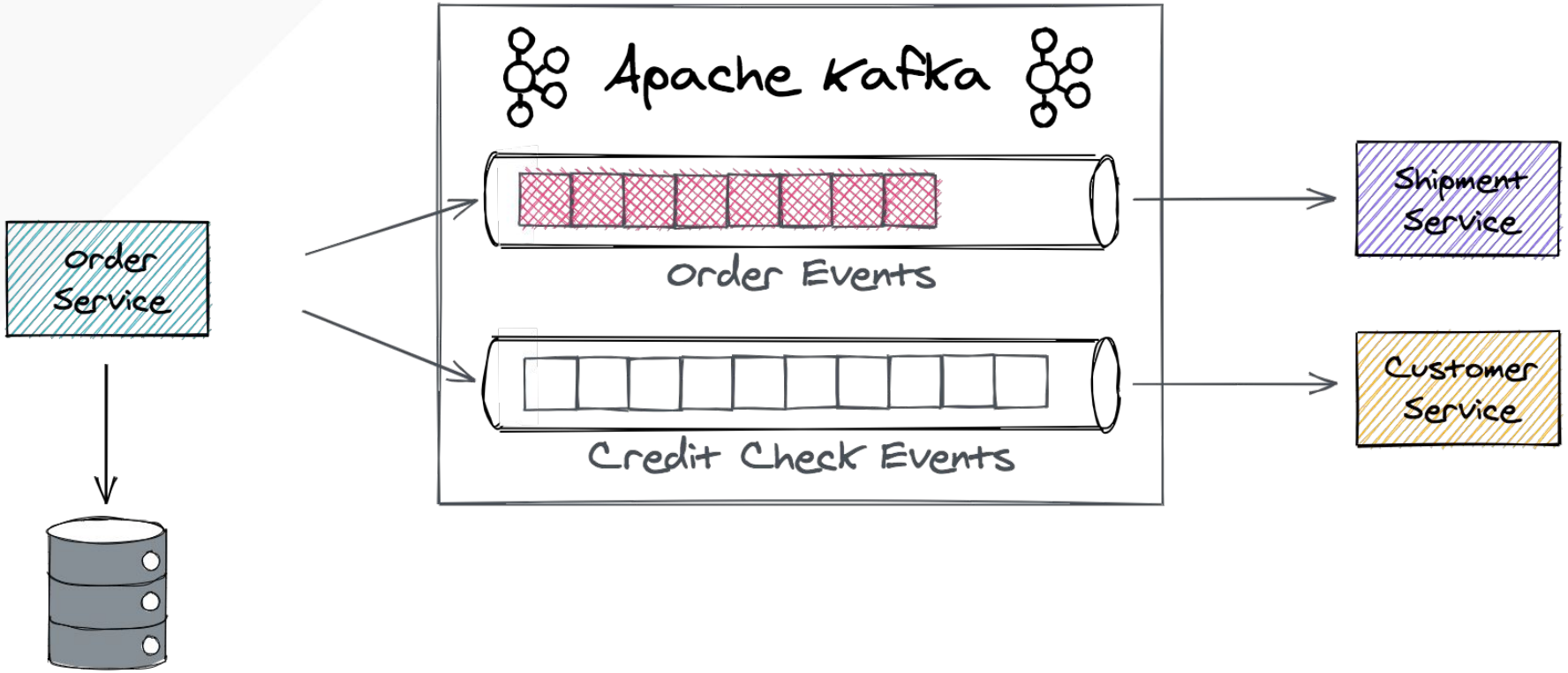
- **Taps into TX log** to capture INSERT/UPDATE/DELETE events
- Propagated to consumers via **Apache Kafka** and Kafka Connect

Outbox Pattern

The Problem: Microservices Data Exchange

- Services need to update their **database,**
- send messages to **other services,**
- and that **consistently**!

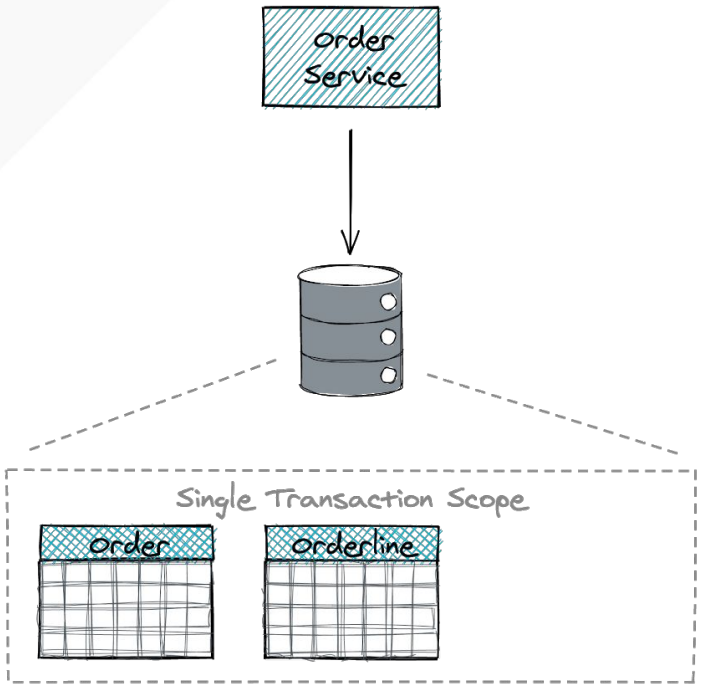


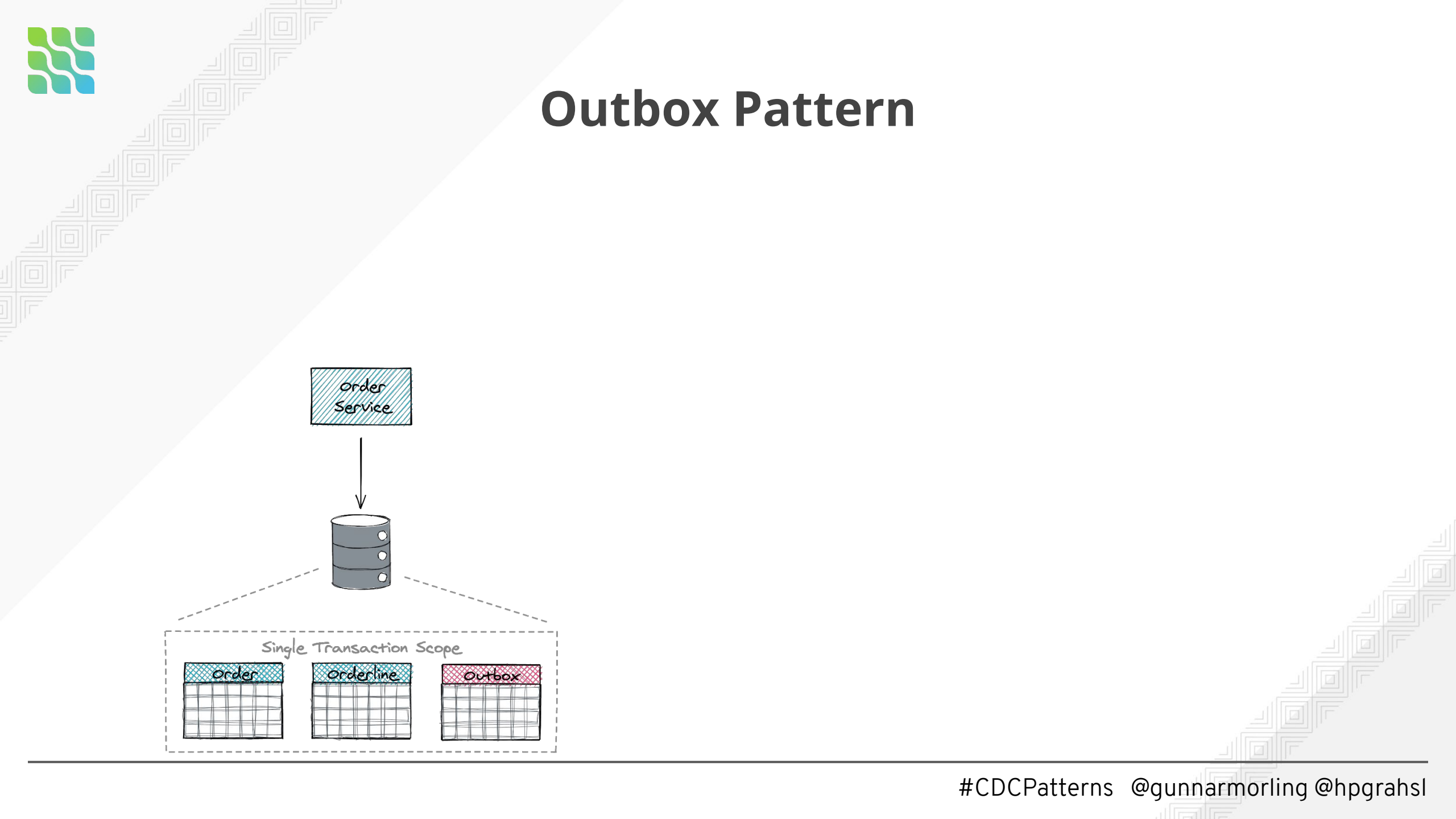


Outbox Pattern

"Dual writes" are **prone to inconsistencies**!

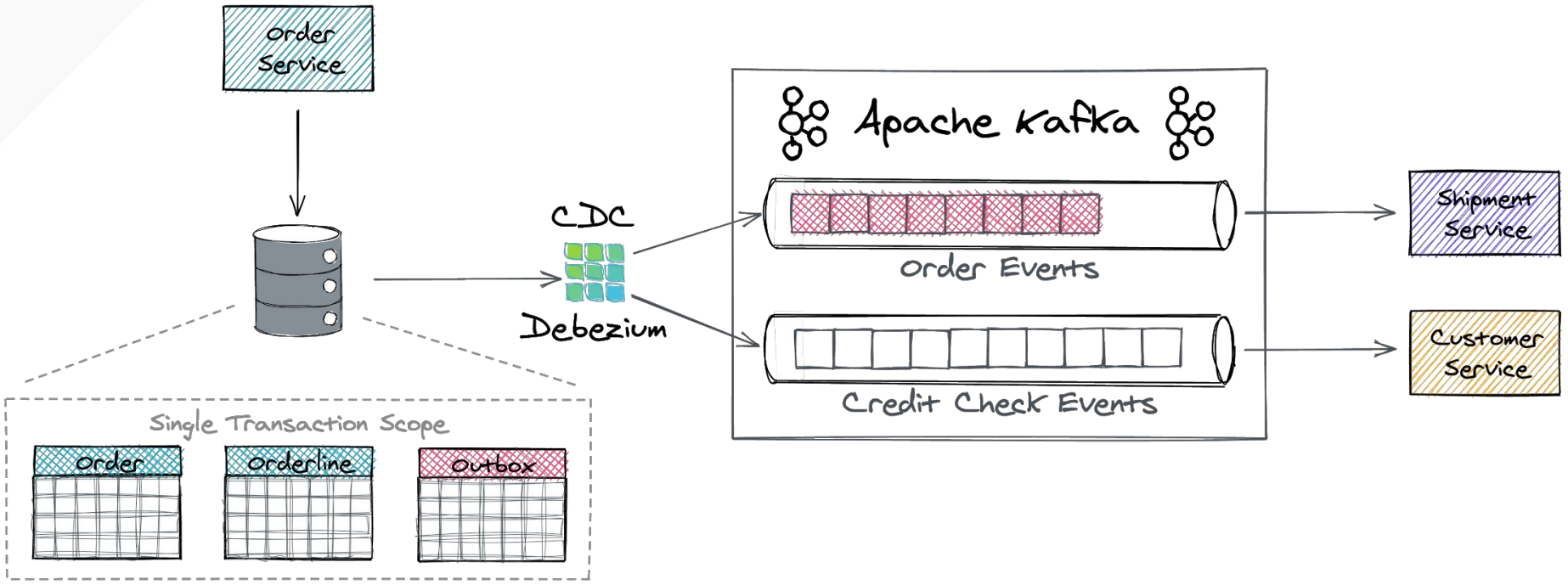
Outbox Pattern





Outbox Pattern

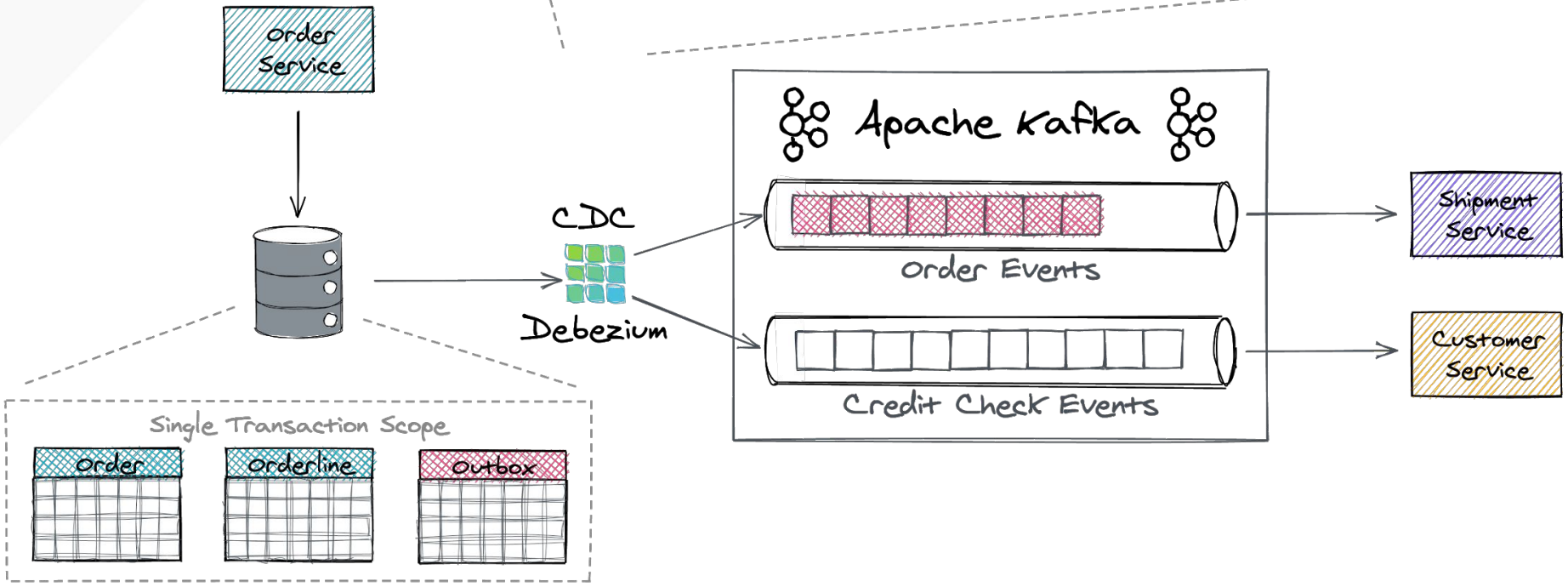
Outbox Pattern



Outbox Pattern

Outbox Table

Id	AggregateType	AggregateId	EventType	Payload
ec6e	order	123	orderCreated	{"id": 123,...}
8af8	order	456	orderLineChanged	{"id": 456,...}
...				

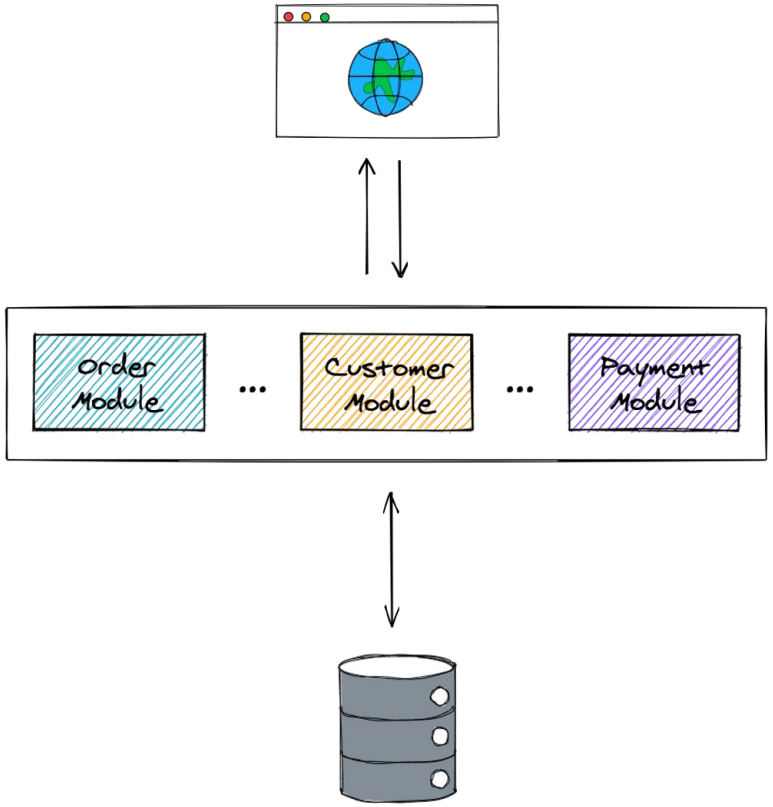


Strangler Fig Pattern

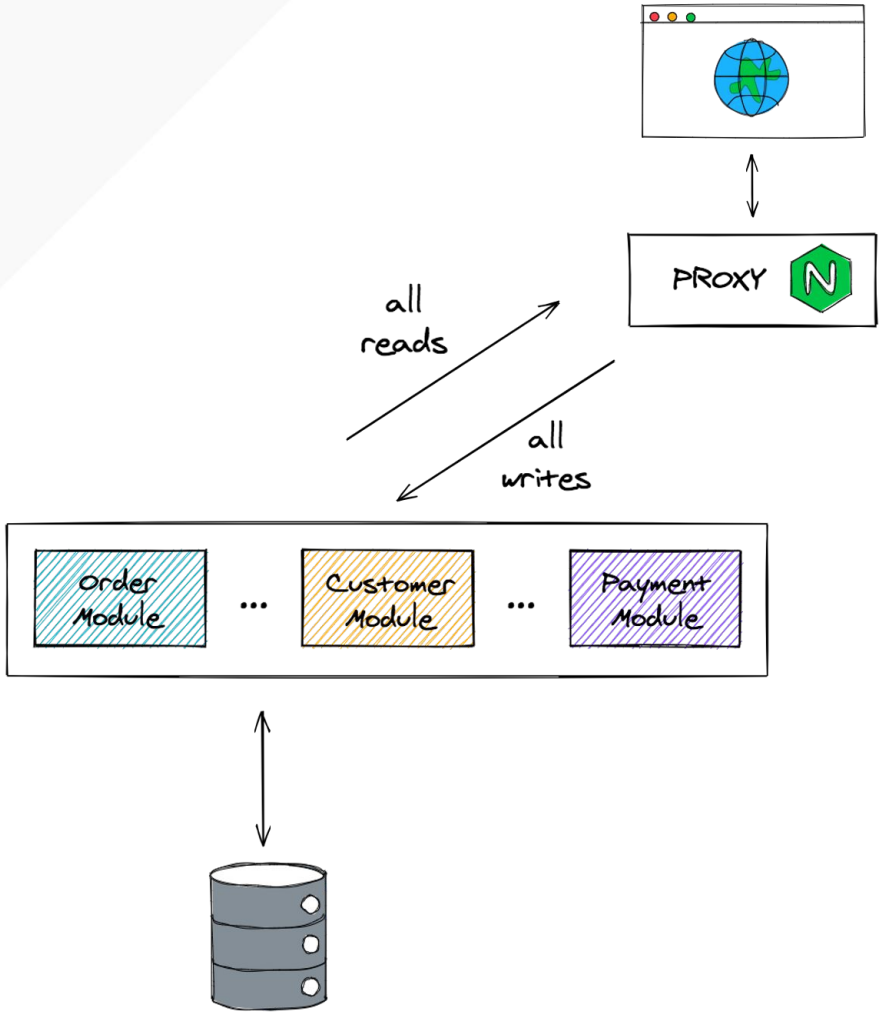
The Problem: Migrating Systems

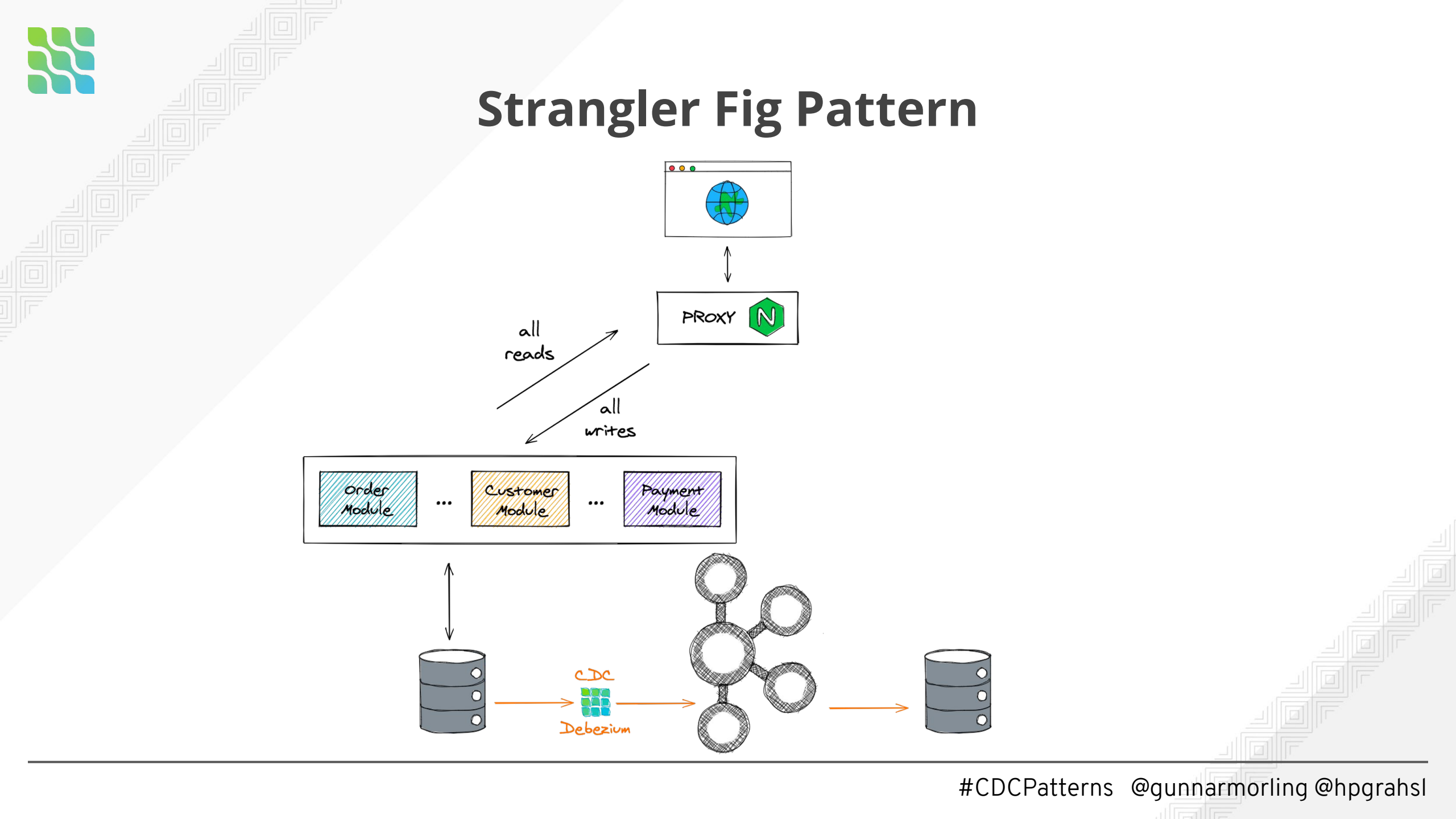
- **Gradually evolve** from old into new
- Support temporary **coexistence**
- **Avoid big bang** cut-over

Strangler Fig Pattern

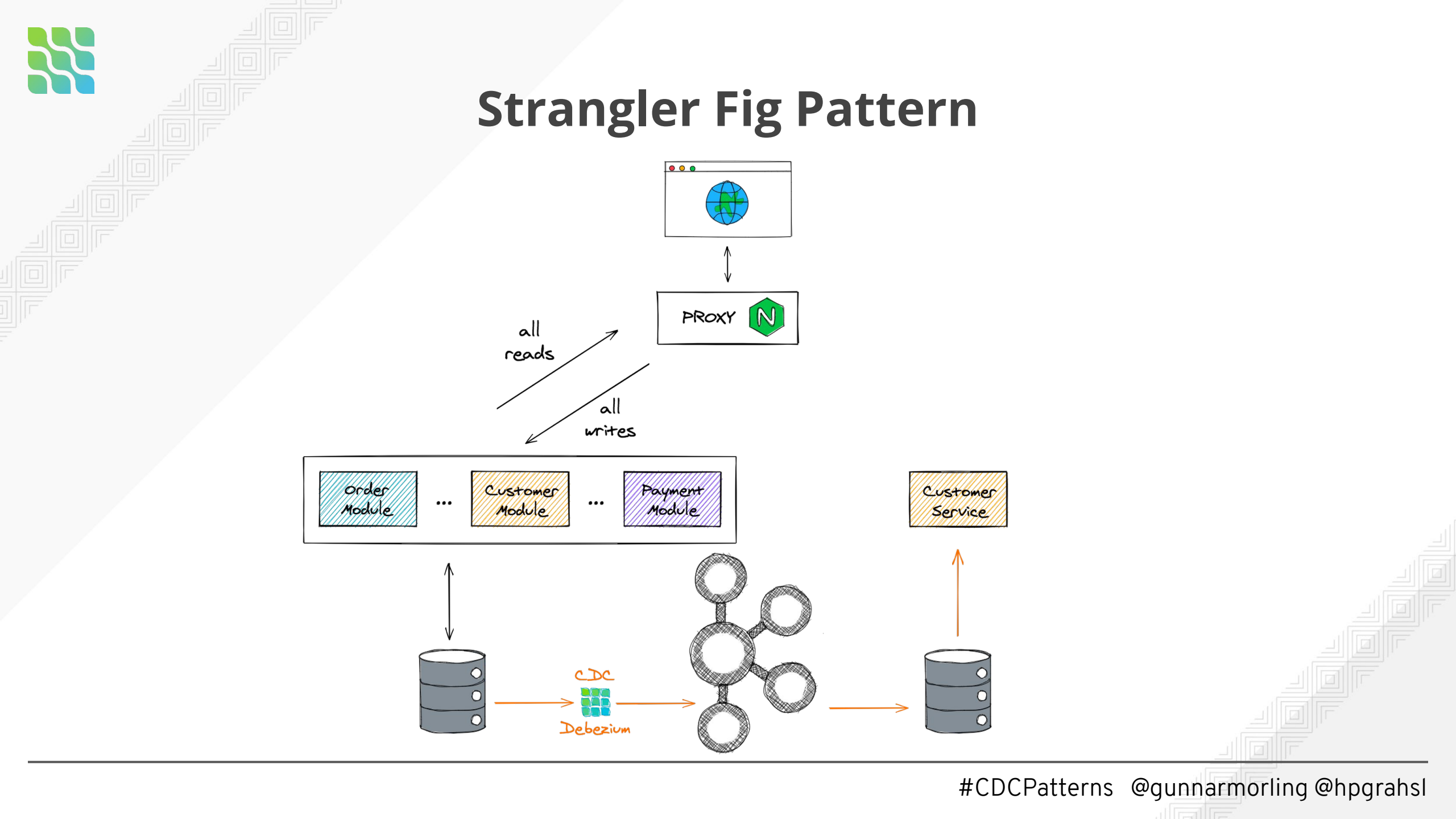


Strangler Fig Pattern



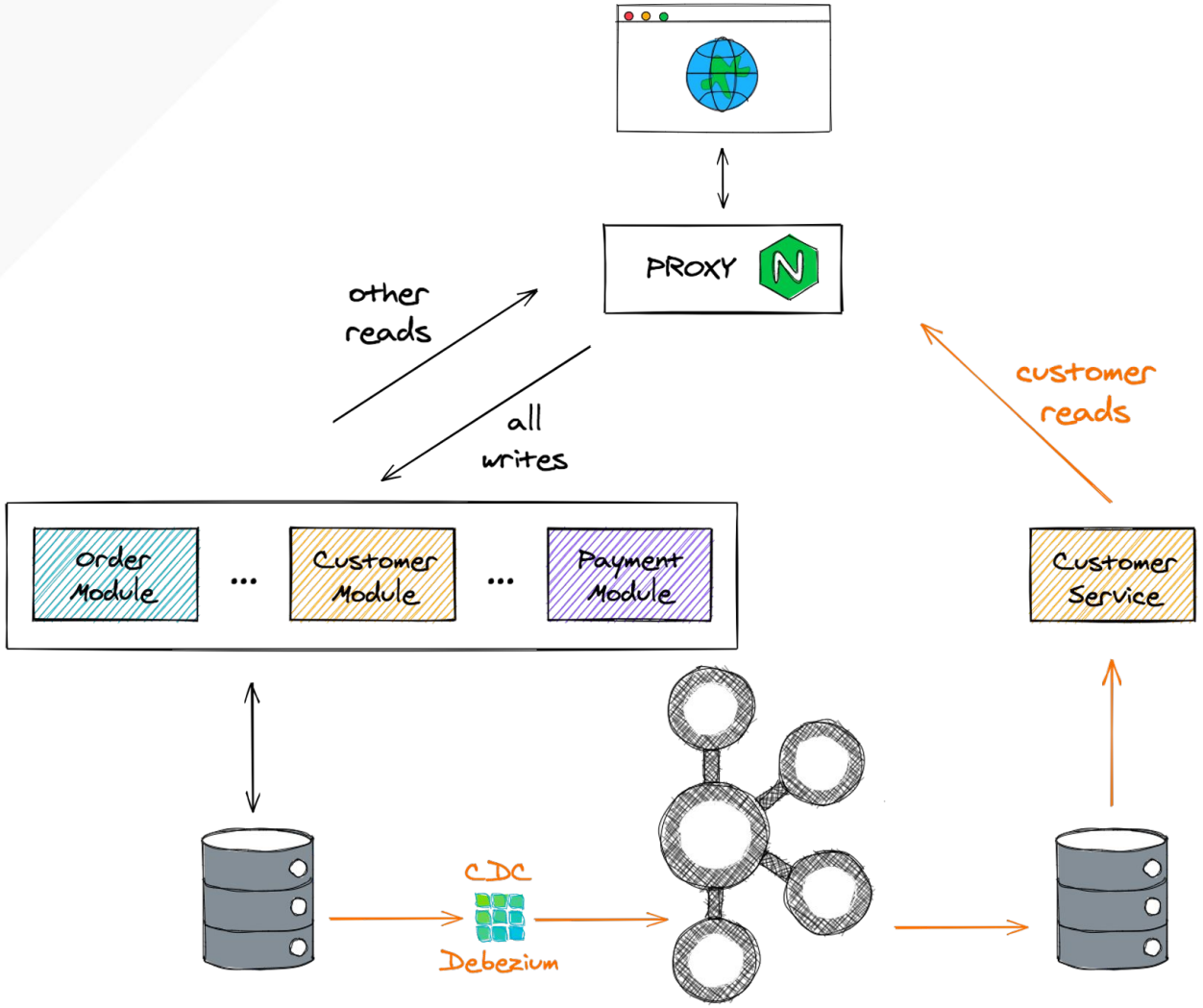


Strangler Fig Pattern

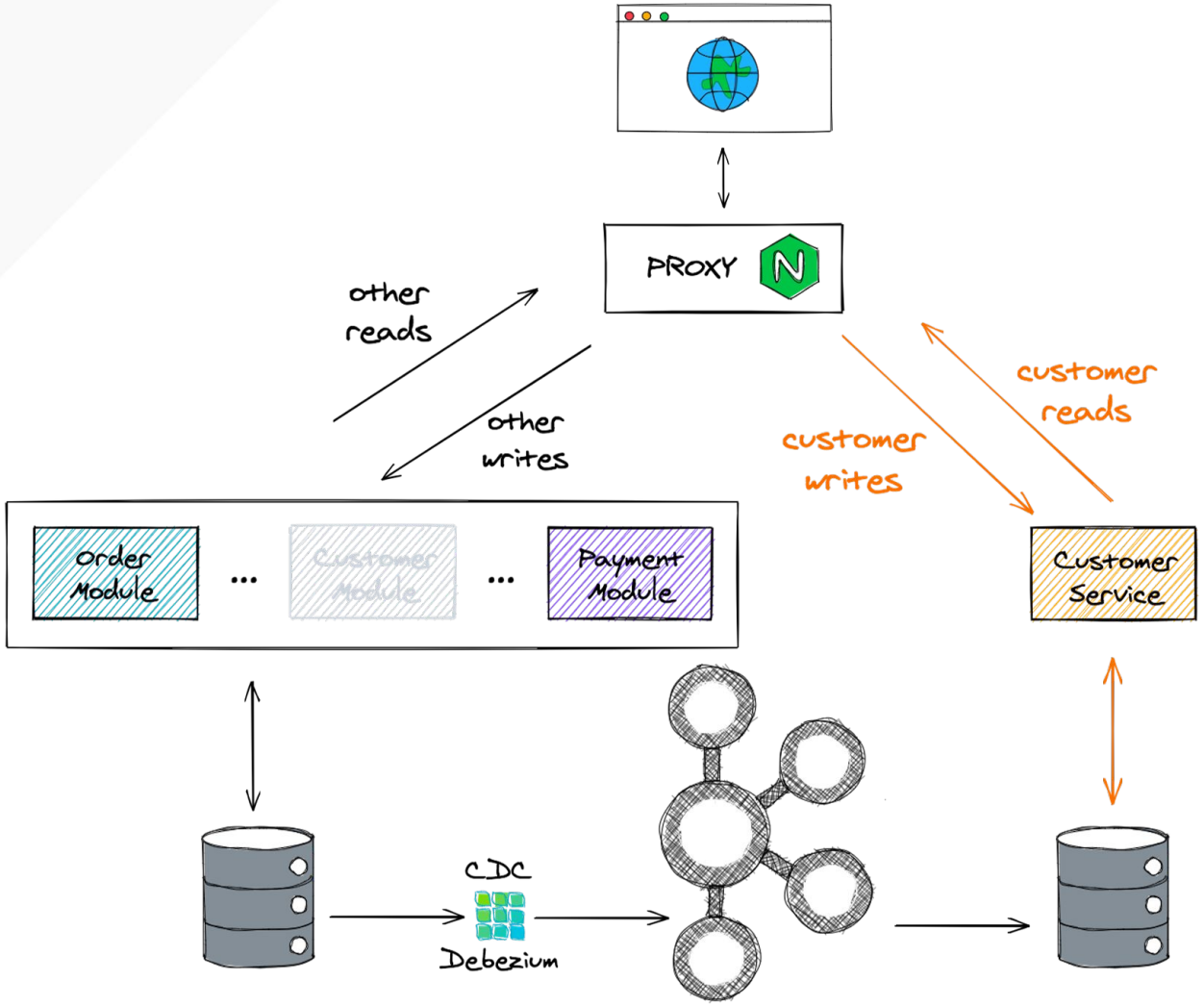


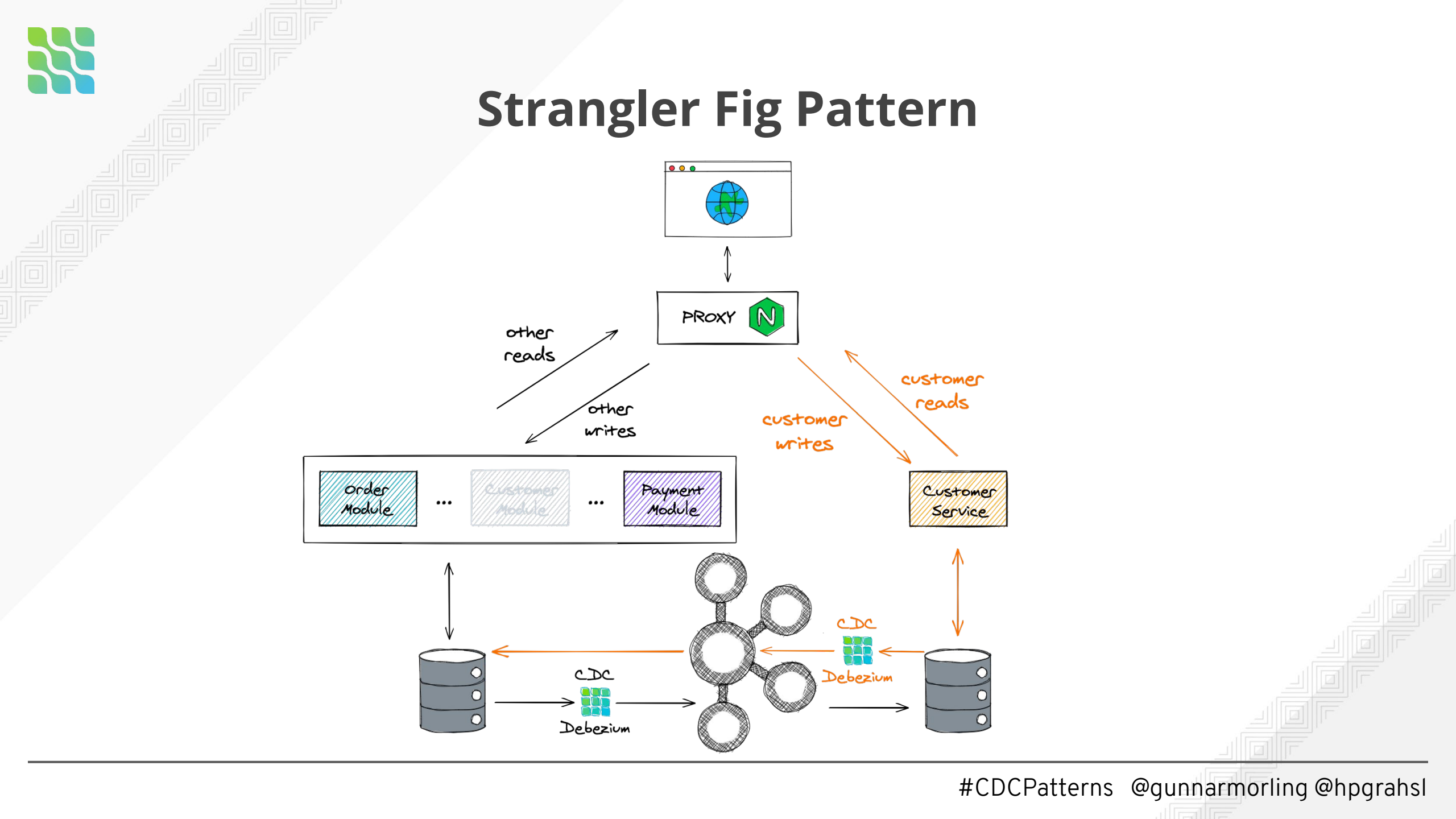
Strangler Fig Pattern

Strangler Fig Pattern



Strangler Fig Pattern





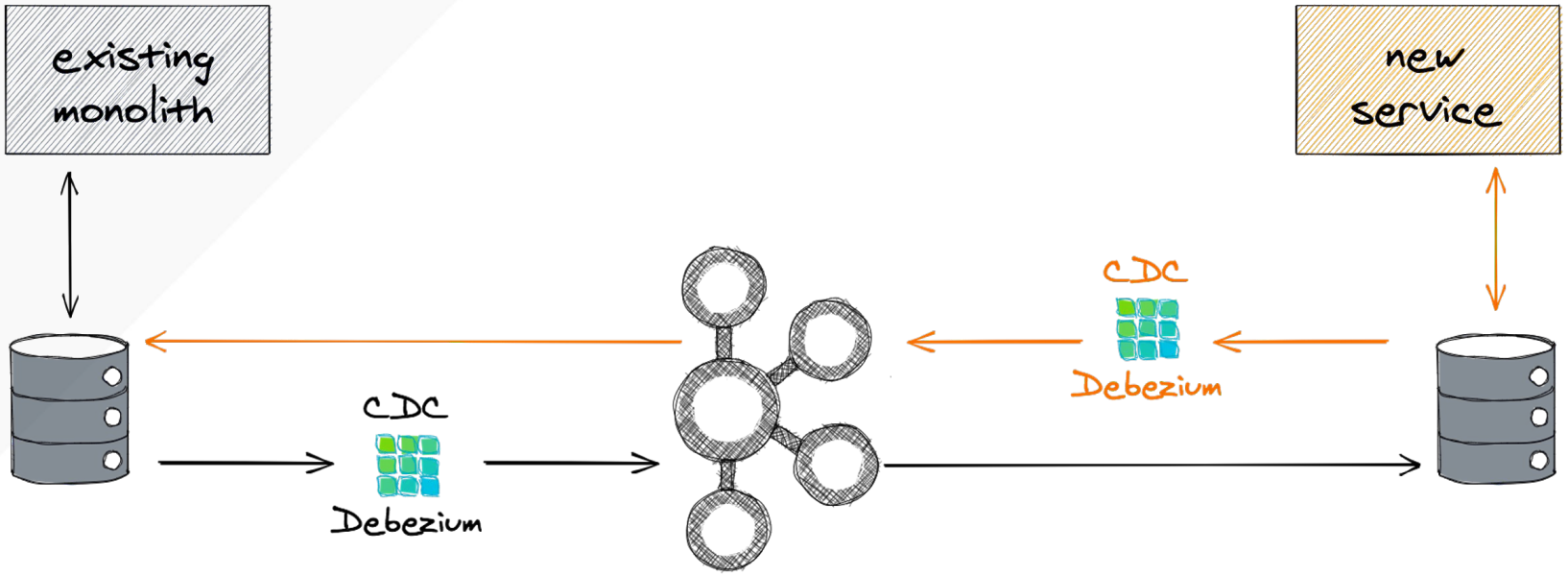
Strangler Fig Pattern

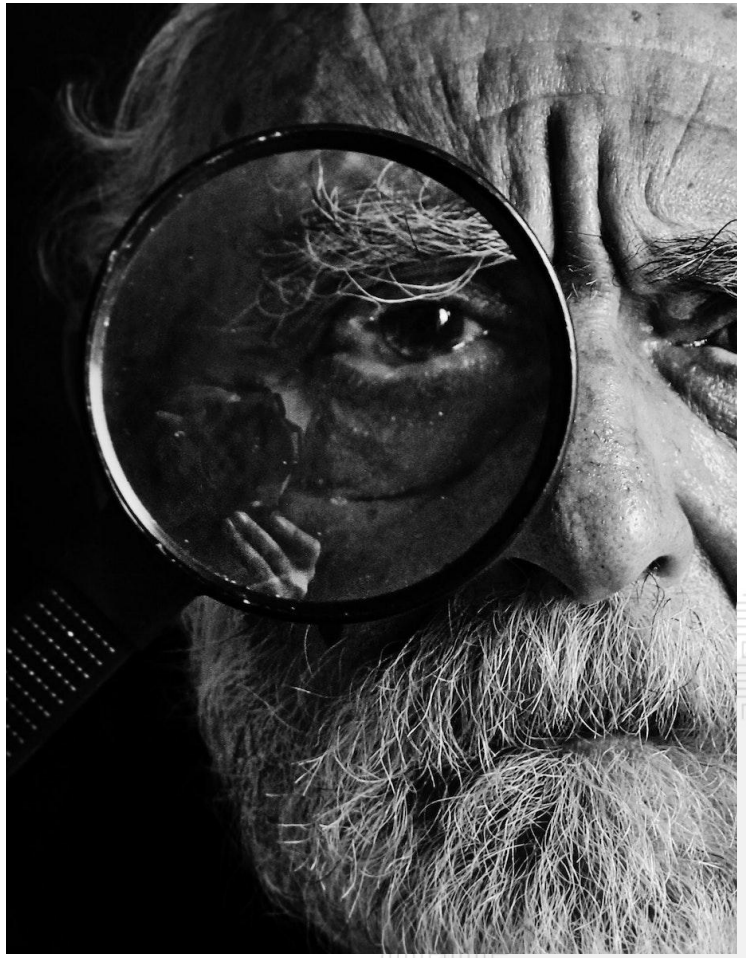
Benefits

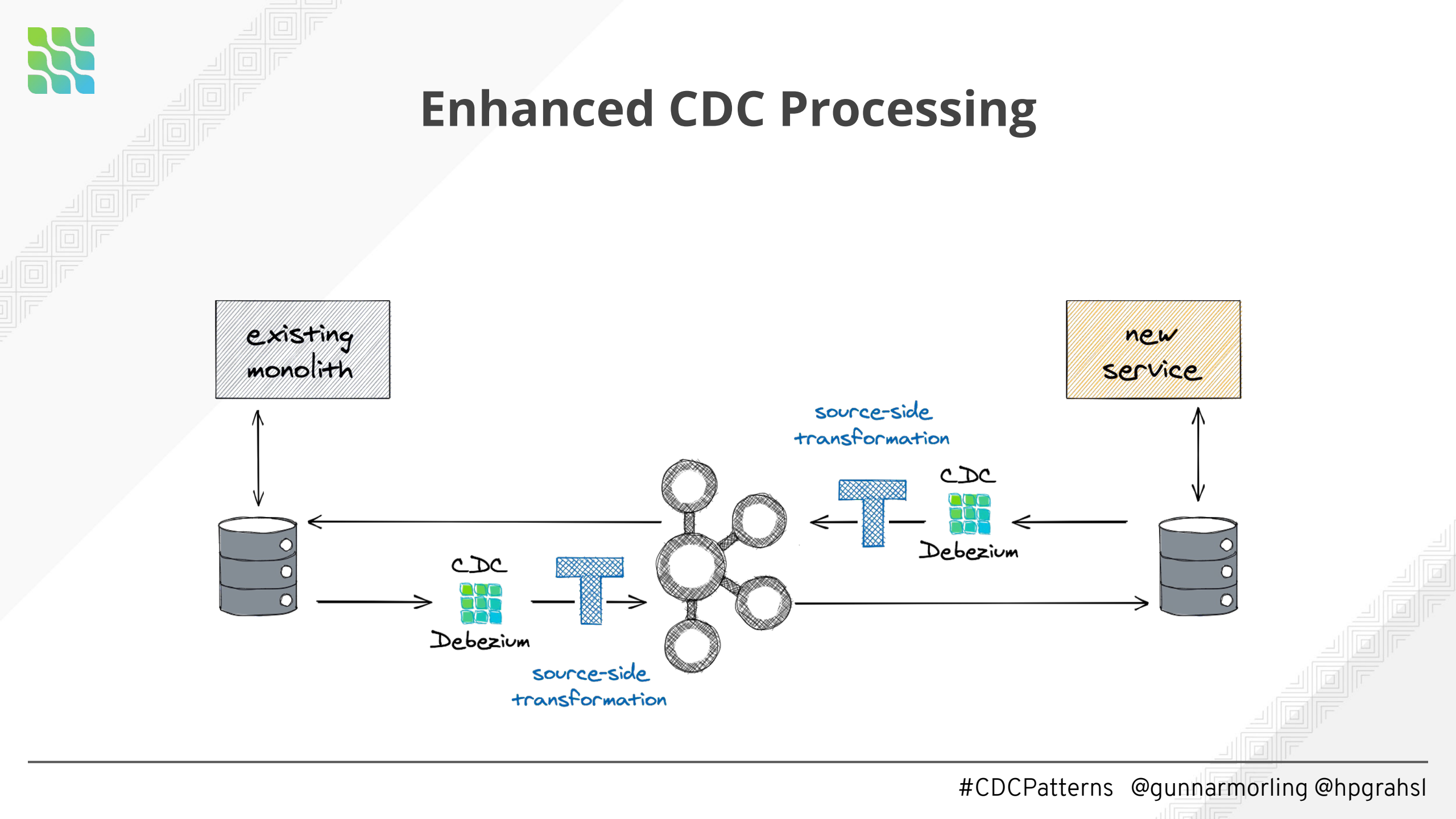
- Incremental migration → **“baby steps”**
- **Pause or stop** migration
  without losing spent efforts
- Migration steps ideally **reversible**

Rationale: ⚠️ **minimize risk** ⚠️

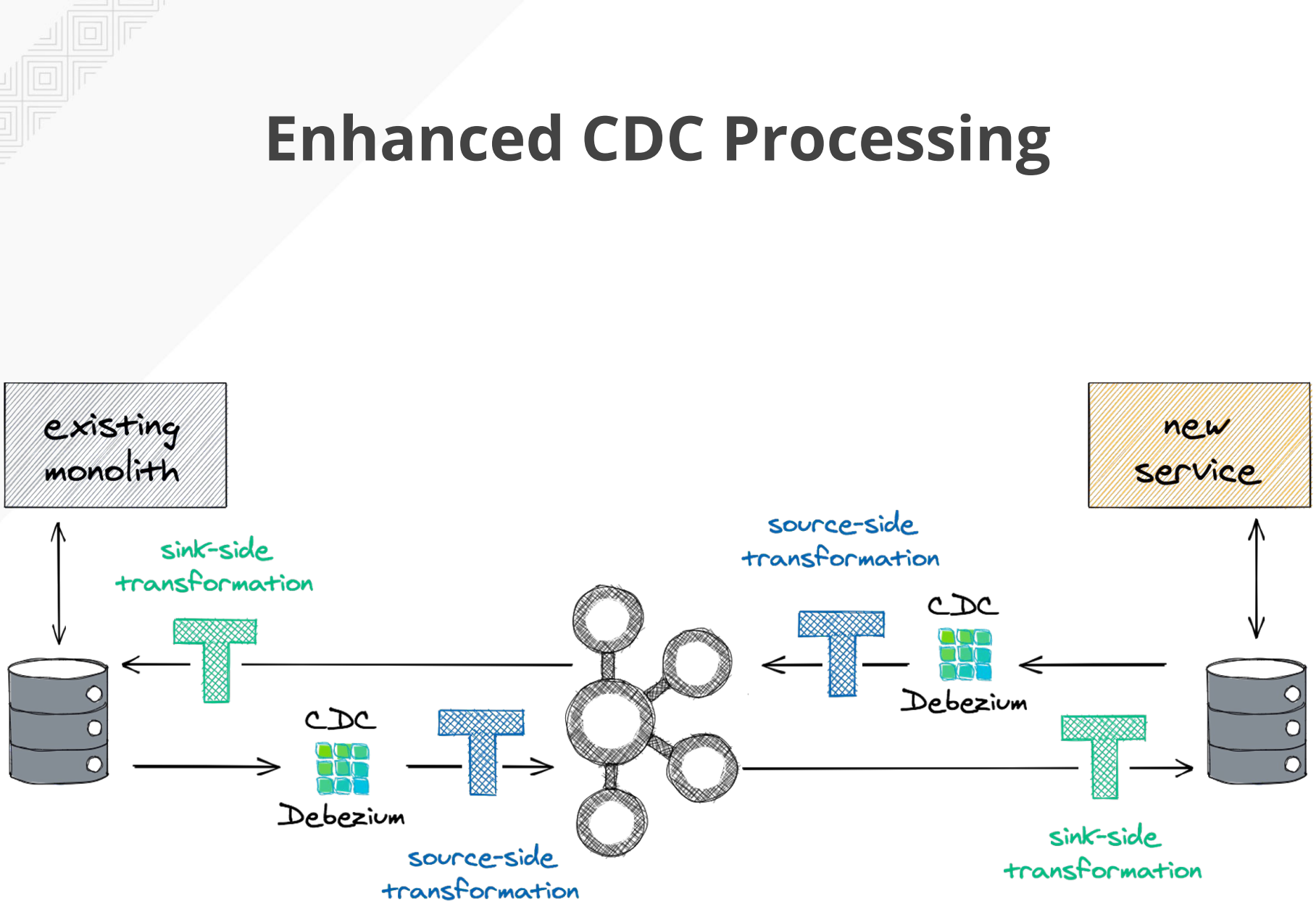
CDC Close-Up

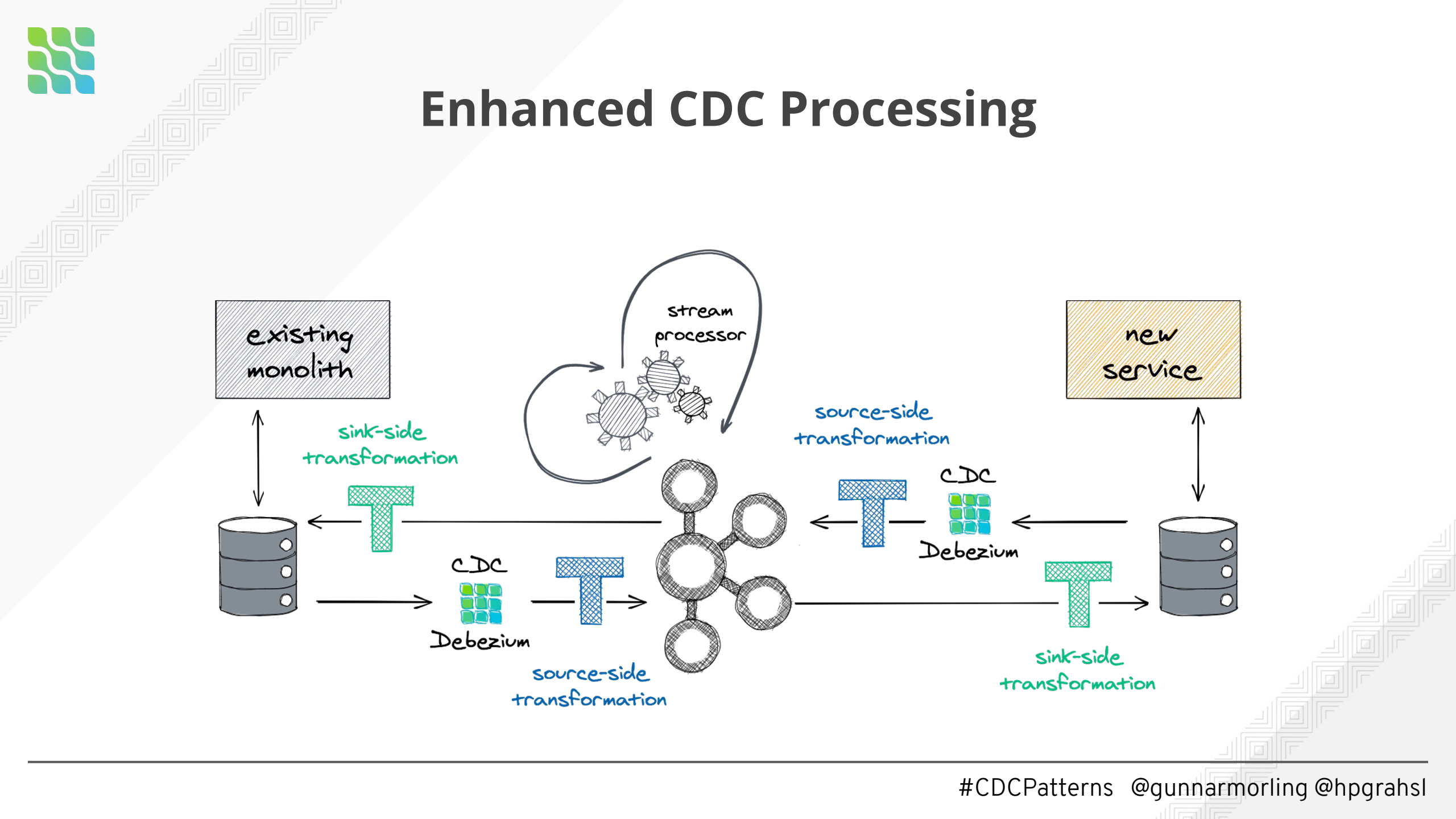






Enhanced CDC Processing


Enhanced CDC Processing
existing monolith
new service
sink-side transformation
source-side transformation
CDC
Debezium
CDC
Debezium
source-side transformation
sink-side transformation



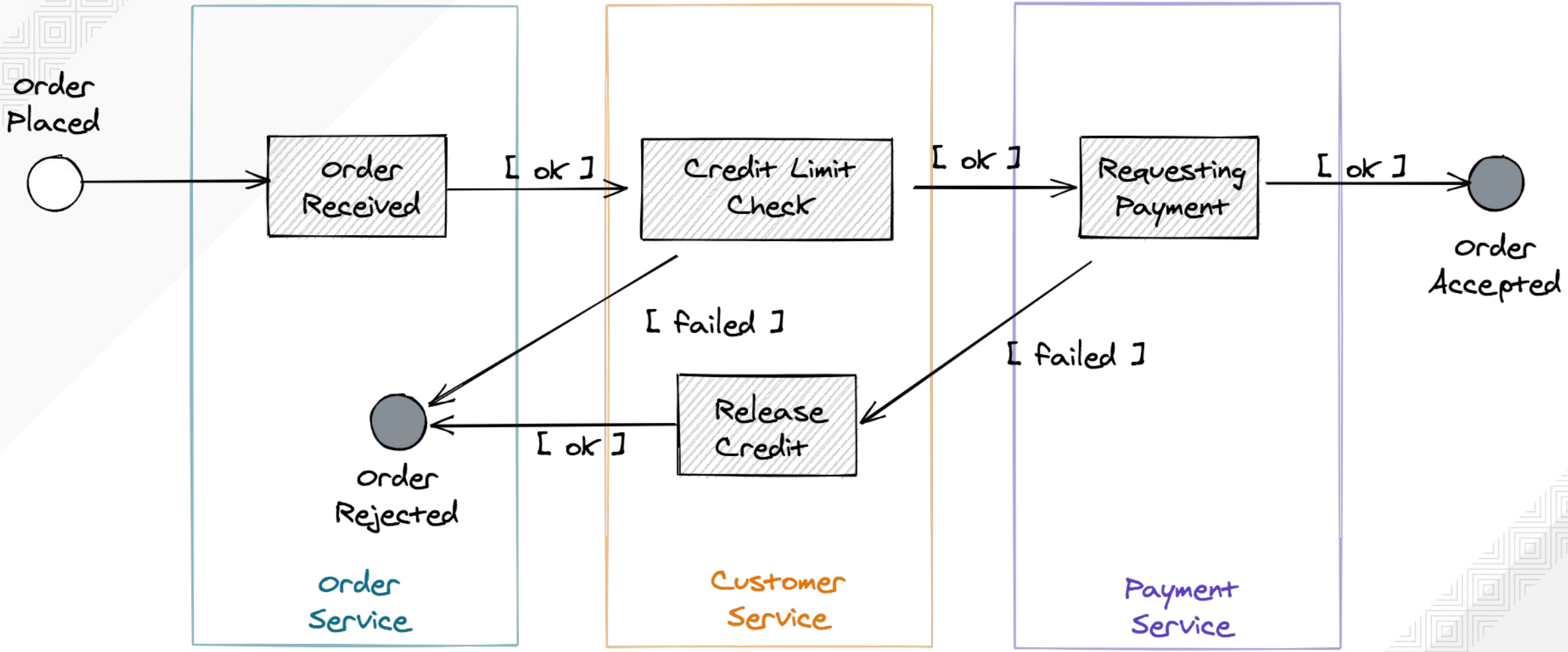
Enhanced CDC Processing

Saga Pattern

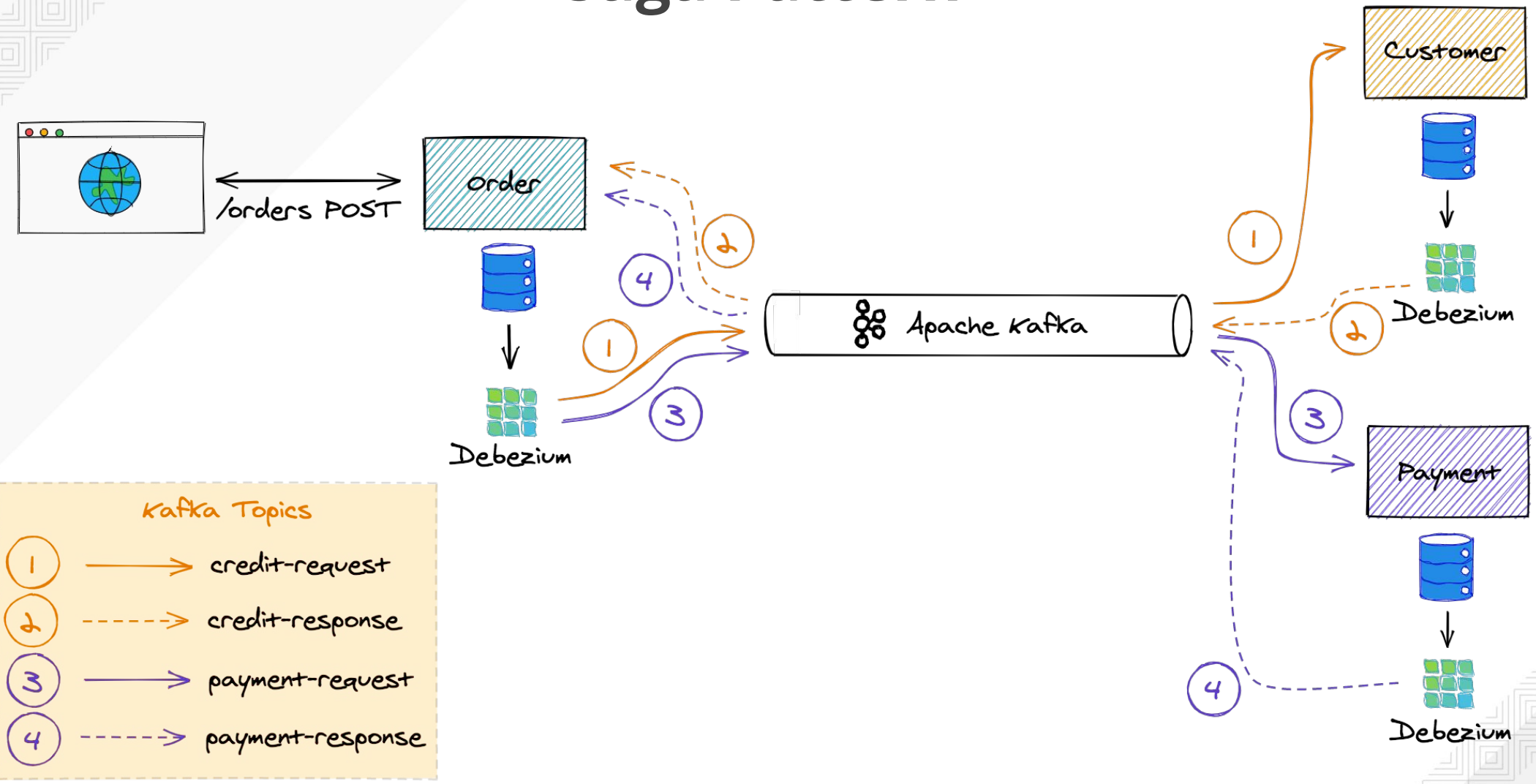
The Problem: Long-running Business Transactions

- Multiple services need to **act collaboratively** to achieve a consistent outcome
- **Without 2-phase commit** protocols
- Ensure **correctness** in case of failures

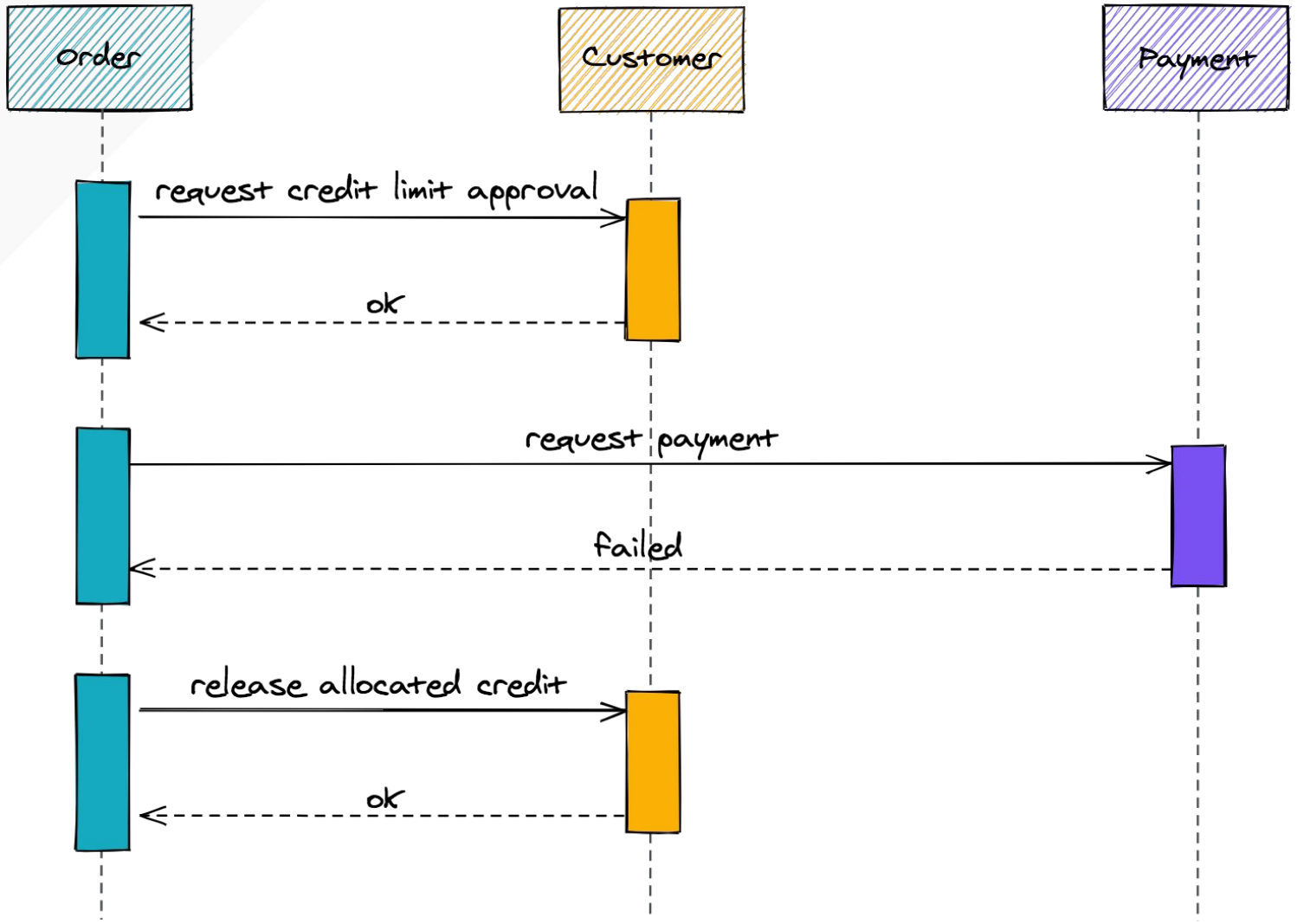
Saga Pattern



Saga Pattern



Saga Pattern

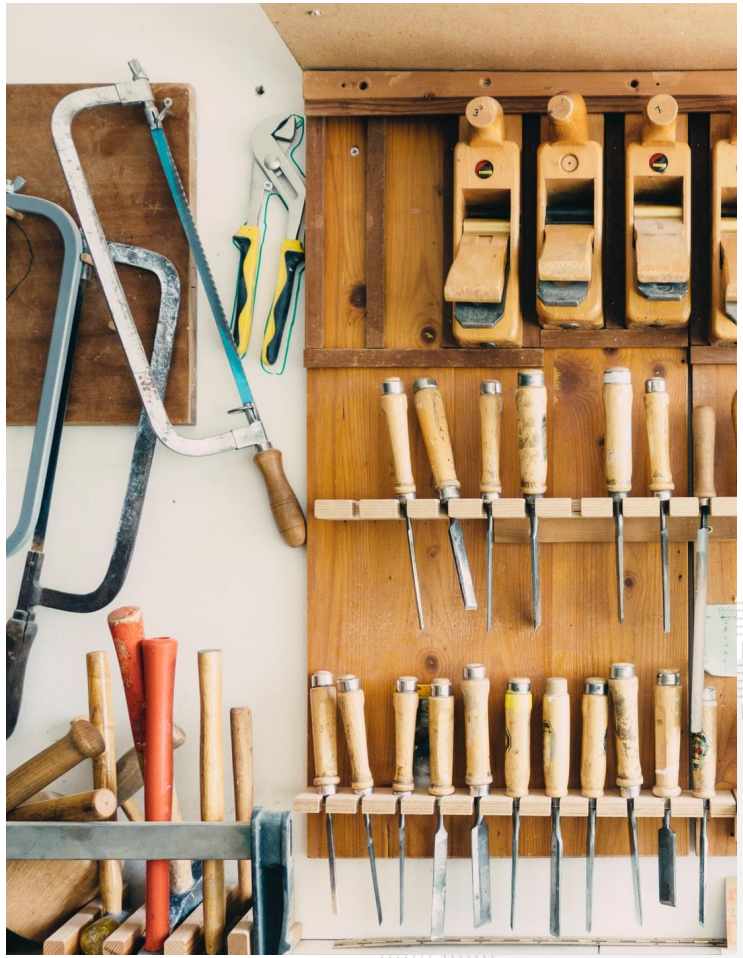


Demo

Wrap-Up

Takeaways

- **CDC:** a powerful tool in the box for event-driven architectures
- **Debezium:** open-source CDC for a variety of databases
- **Call to Action:**
  Would you like **built-in Saga support**?



Resources

- **Outbox** implementation
  https://debezium.io/blog/2019/02/19/reliable-microservices-data-exchange-with-the-outbox-pattern/
- **Strangler fig** pattern
  https://martinfowler.com/bliki/StranglerFigApplication.html
- **Saga** implementation
  https://www.infoq.com/articles/saga-orchestration-outbox/

Q & A

gunnar@hibernate.org

@gunnarmorling

grahslhp@gmail.com

@hpgrahsl

Image Credits

In Order of Appearance

Unsplash https://unsplash.com/license

© Shiro hatori https://unsplash.com/photos/WR-ifjFy4CI
© Pablo García Saldaña https://unsplash.com/photos/lPQIndZz8Mo
© David Clode https://unsplash.com/photos/T49WTav4LgU
© Aaron Burden https://unsplash.com/photos/GFpxQ2ZyNc0
© Nathan Dumlao https://unsplash.com/photos/wQDysNUCKfw
© mari lezhava https://unsplash.com/photos/q65bNe9fW-w
© Michał Parzuchowski https://unsplash.com/photos/Bt0PM7cNJFQ
© Charles Forerunner https://unsplash.com/photos/3fPXt37X6UQ

Flickr

Attribution 2.0 Generic https://creativecommons.org/licenses/by/2.0/
© Thomas Kamann https://flic.kr/p/coa2c

CC0 1.0 Universal Public Domain Dedication https://creativecommons.org/publicdomain/zero/1.0/
© Wall Boat https://flic.kr/p/Y6zkmX

Attribution-ShareAlike 2.0 Generic https://creativecommons.org/licenses/by-sa/2.0/
© Maria Eklind https://flic.kr/p/251HmVw