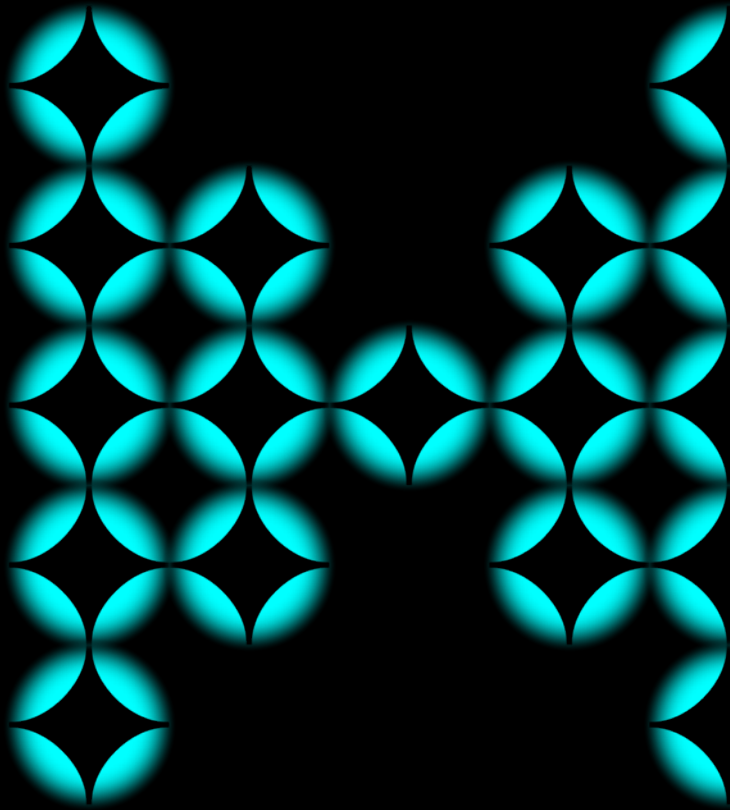
HAZELCAST



Zero downtime on Kubernetes

Nicolas Fränkel

@nicolas_frankel

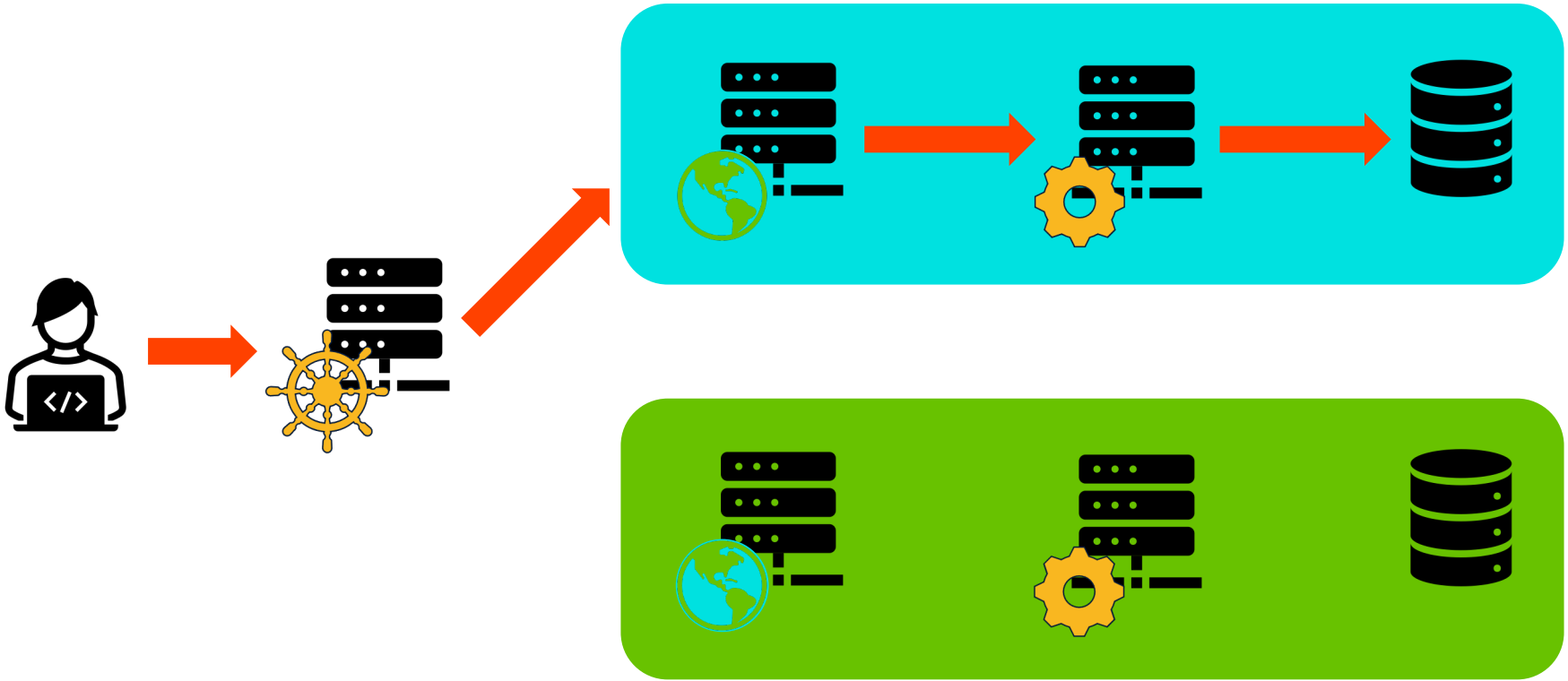
Me, myself and I

- Developer
- Developer Advocate
- We live in interesting times and I’m curious

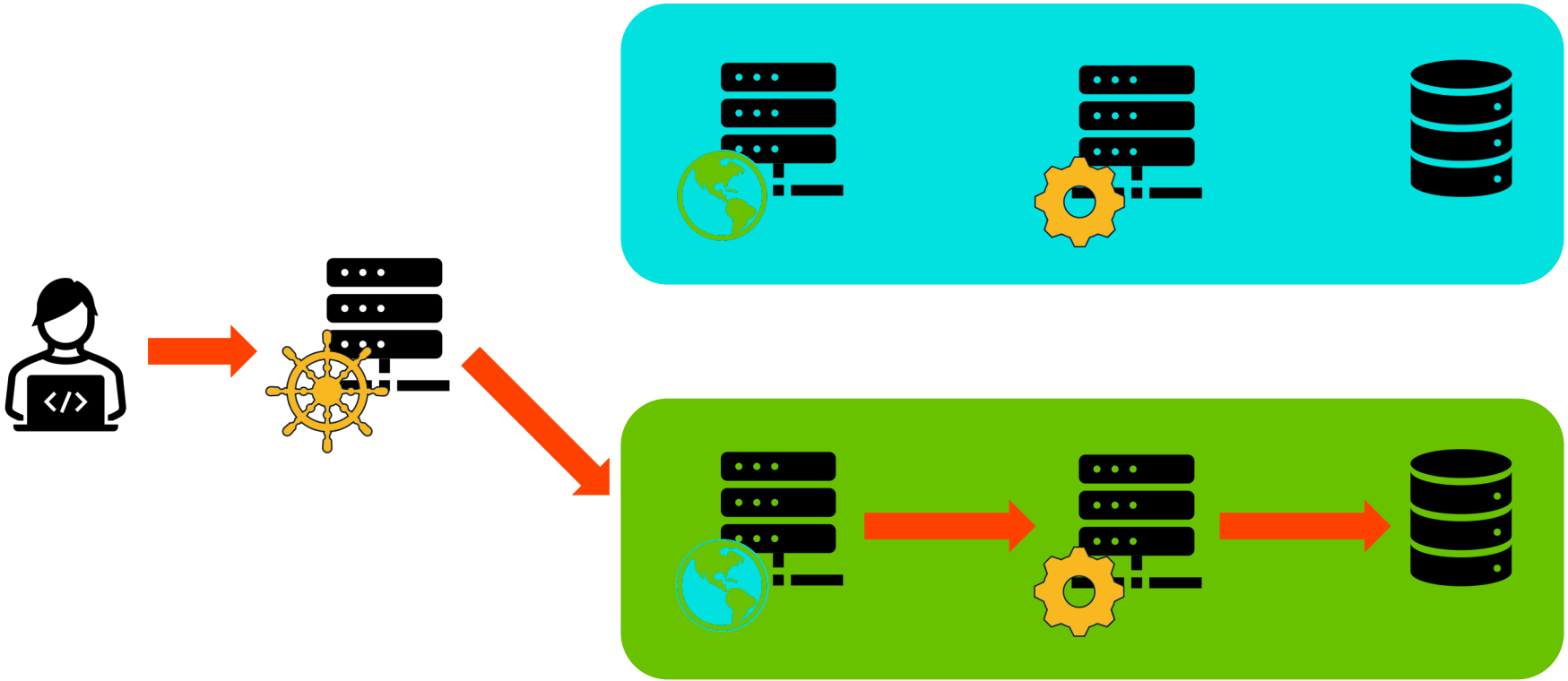
Why zero downtime?

1. Business wants it
   - Downtime has a cost
2. Users expect it
   - When was the last time you saw Google Search display “Please come back later”?

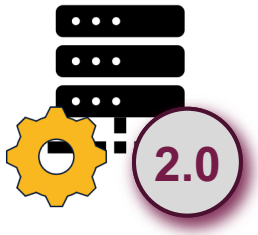
Blue-Green deployment

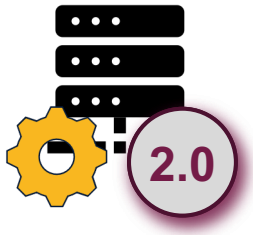


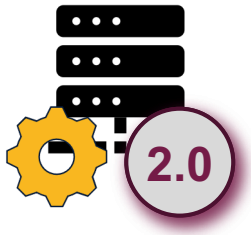
Blue-Green deployment



Kubernetes rolling updates principle

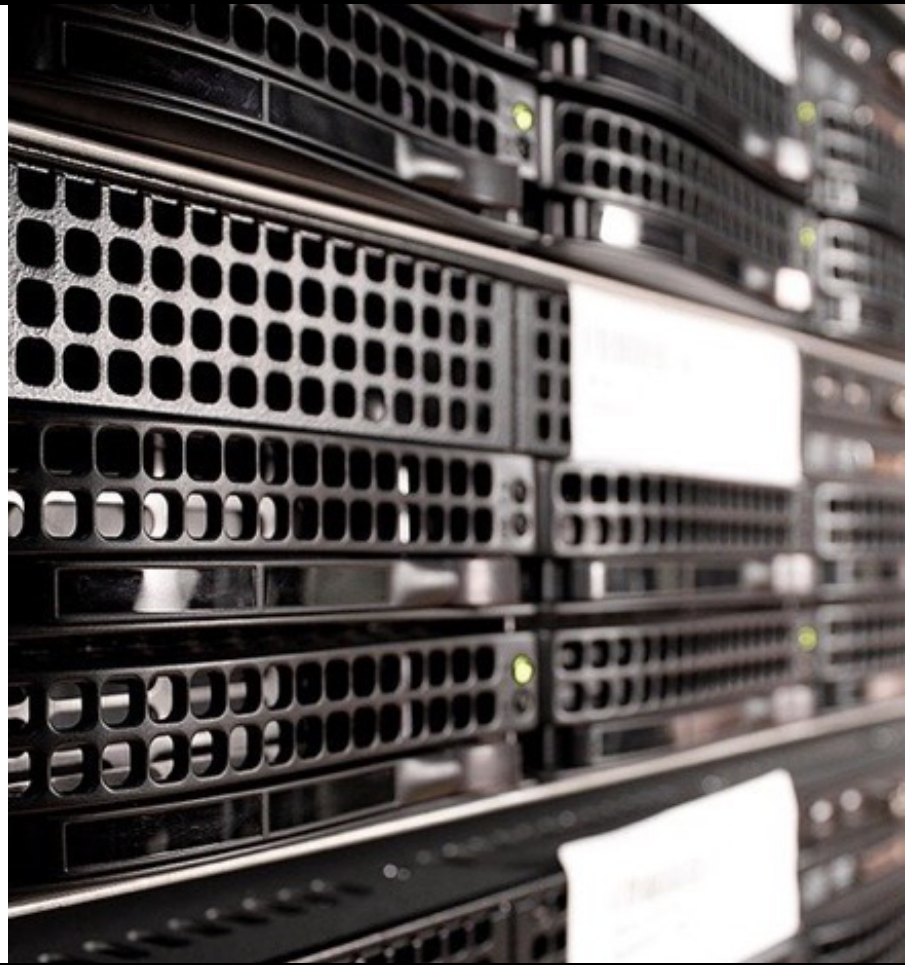




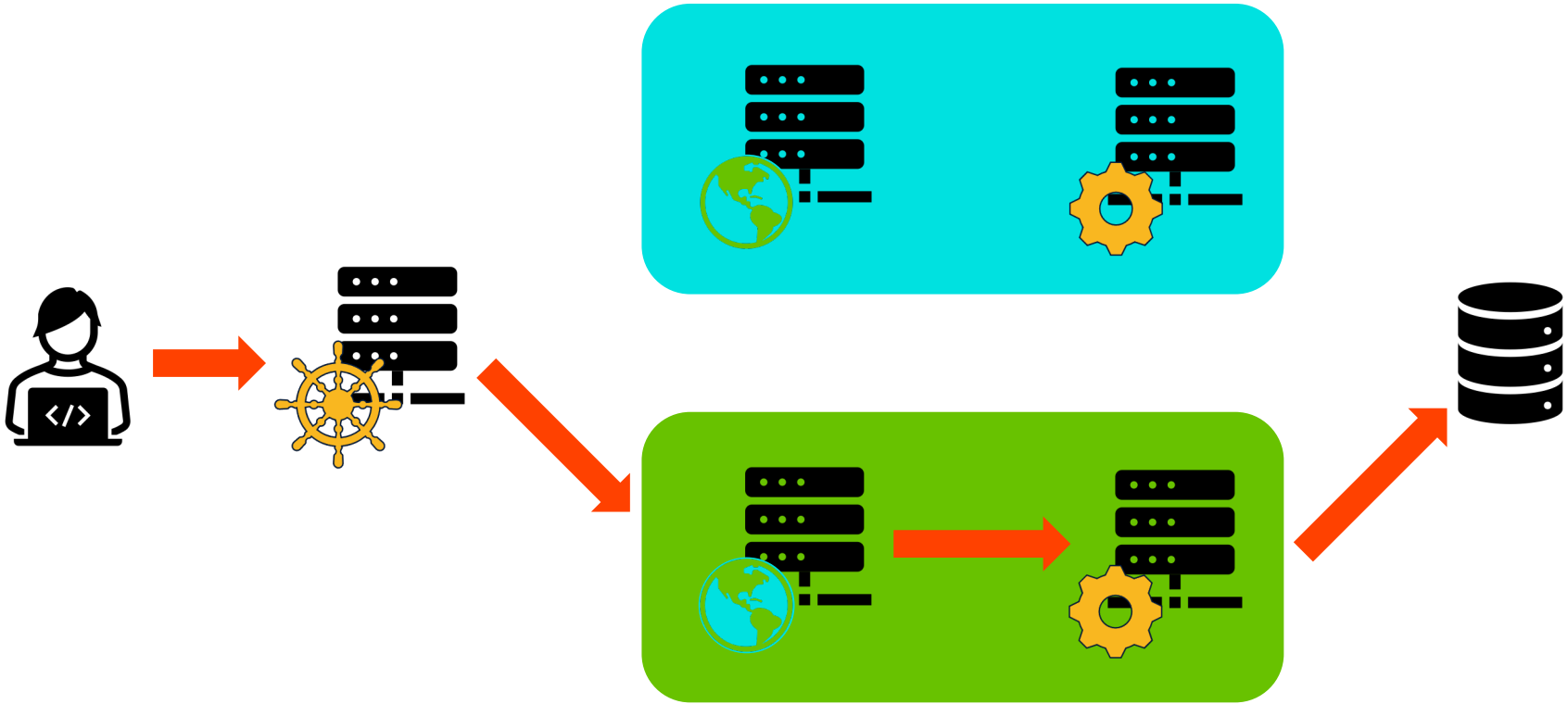


Zero downtime's issues relate to state

- State **in memory**
  - User sessions
  - → Session replication
- State **in the database**
  - → That's the hard spot!



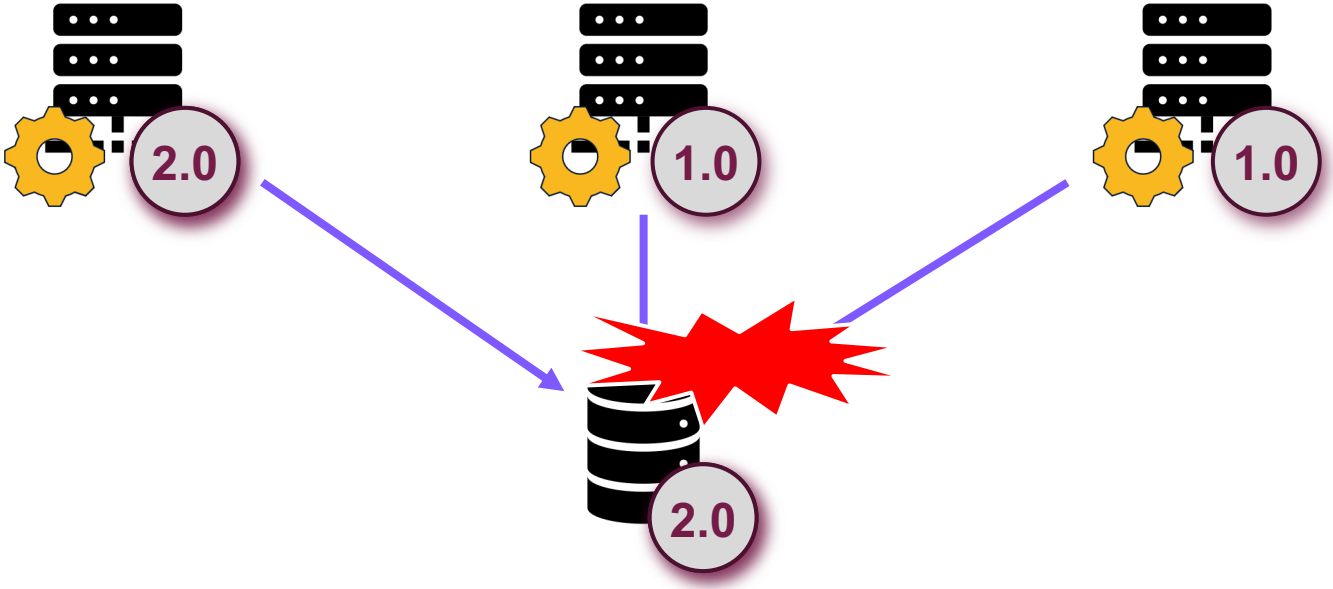
Blue-Green deployment variant



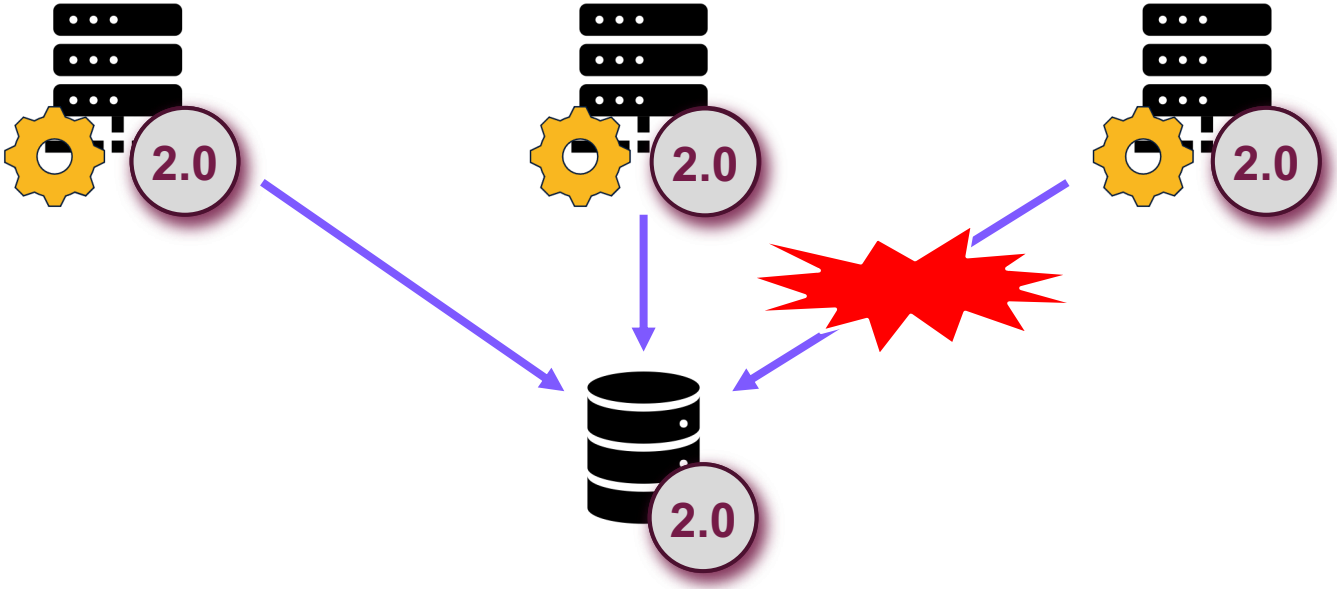
Option 1: Keep the same database

- The application needs to cope with two versions of the schema

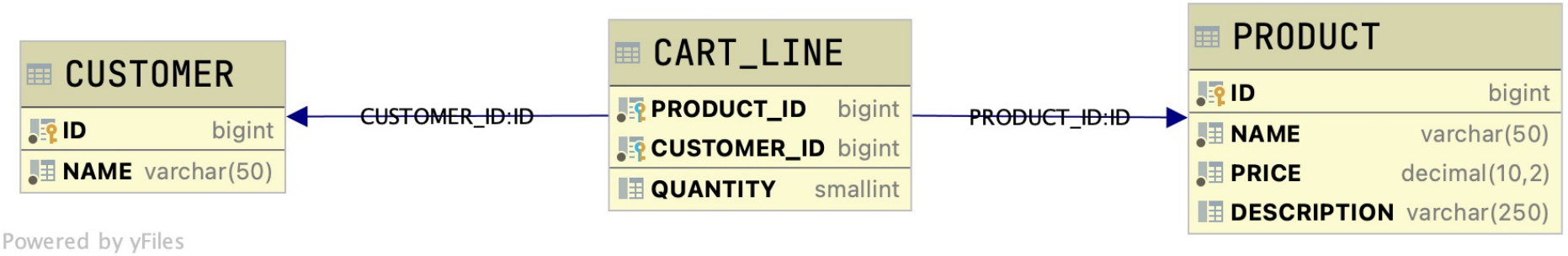
Rolling upgrade issue with a database



More issues with rollback



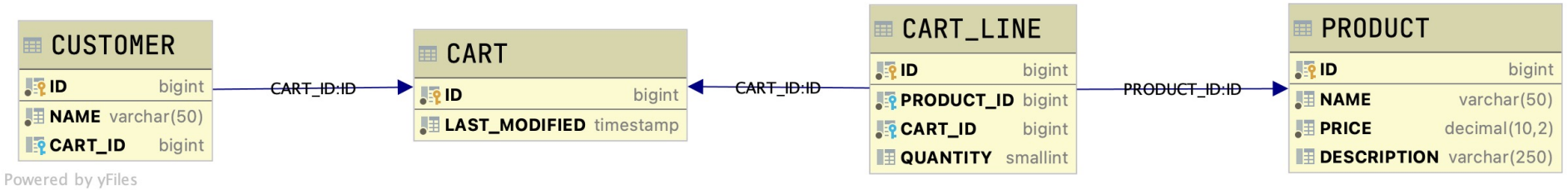
An e-commerce use-case



New business requirement comes in!

- Keep track of when a cart was last updated
  - To send a reminder email after some time has passed

Target schema



Handling schema breaking changes

- Split the breaking change into a series of changes compatible side-by-side
- Plan for rollback (it happens!)

Steps' decomposition

1. Create CART table
   - App uses "old" data model
   - Trigger inserts CART when the first CART_LINE is inserted
2. CART becomes the "source of truth"
   - App uses the CART table
   - Trigger updates CART_LINE with CUSTOMER_ID every time it's inserted
3. **Migration of untouched data**
4. Cleanup

Issues of keeping the same database

- Requires steps' decomposition
- Rollback a single step only
- Needs planning across the organization (devs, DBAs, Ops)
- You will **need to migrate data anyway**

Option 2: Embrace data migration

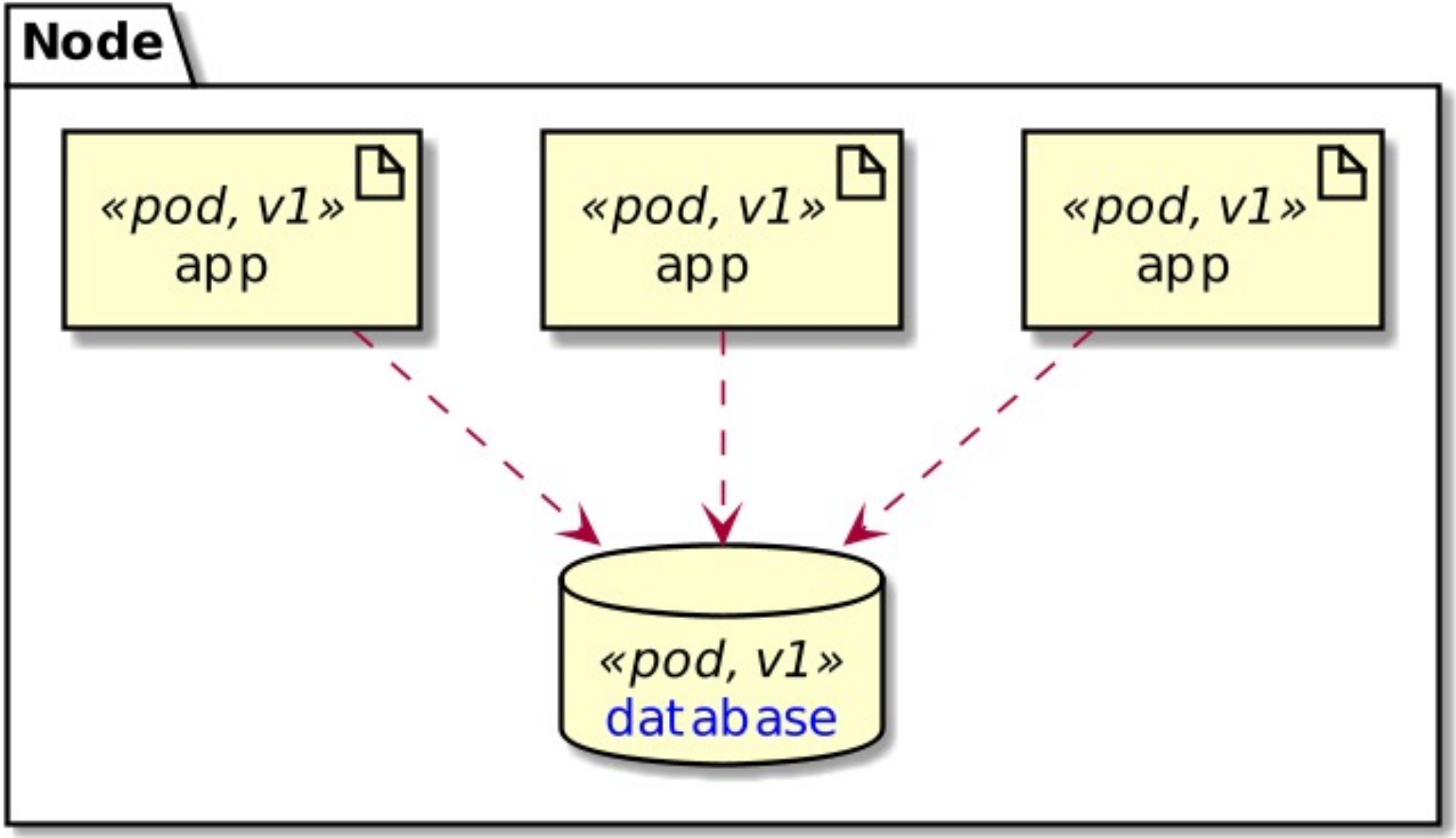
- Have two different databases
- Migration implemented by:
  - Change-Data-Capture
  - **Data streaming**
- Developers are not impacted by Ops' concerns
- It works with any deployment option *e.g.* canary release

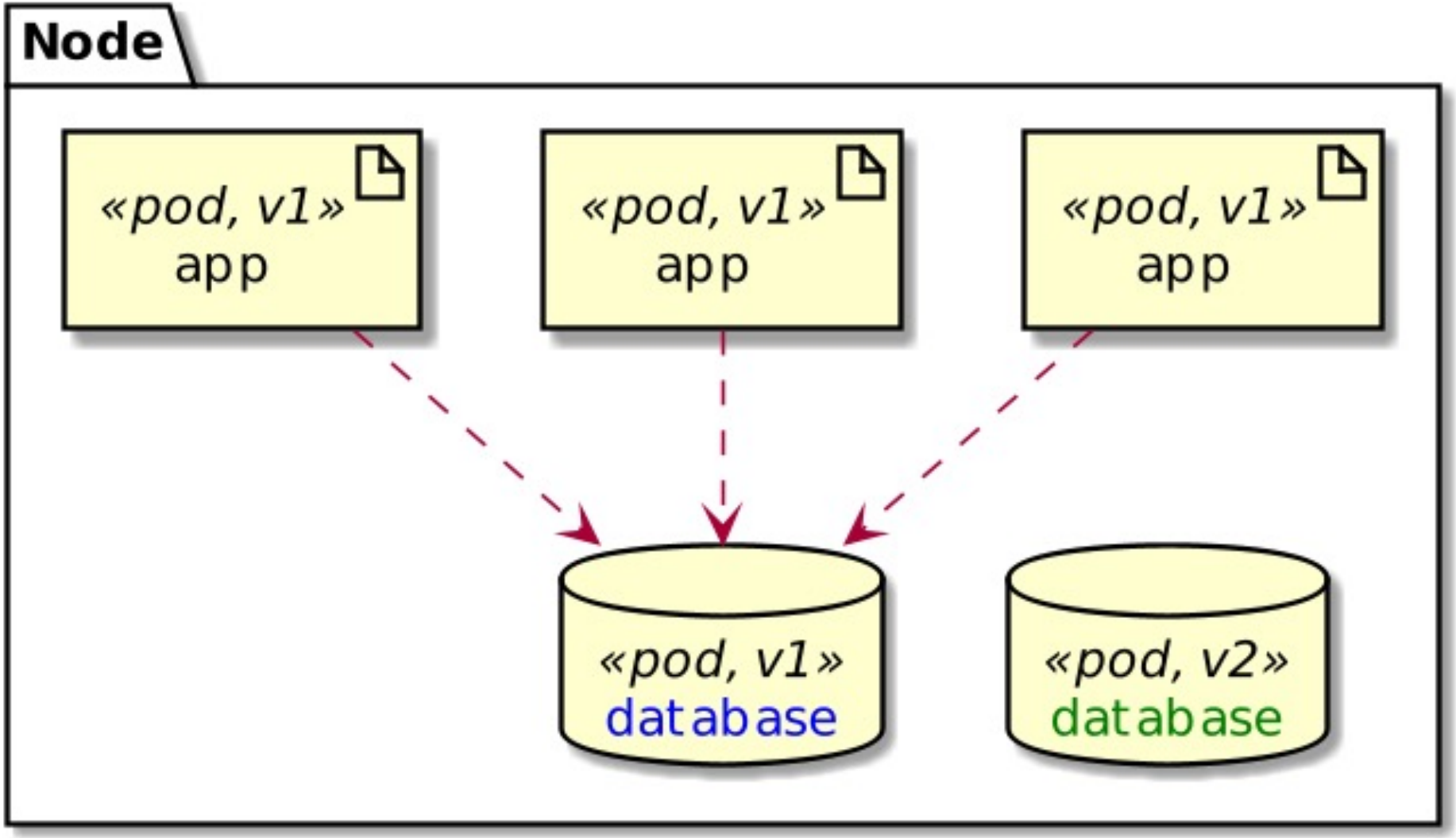
Change-Data-Capture

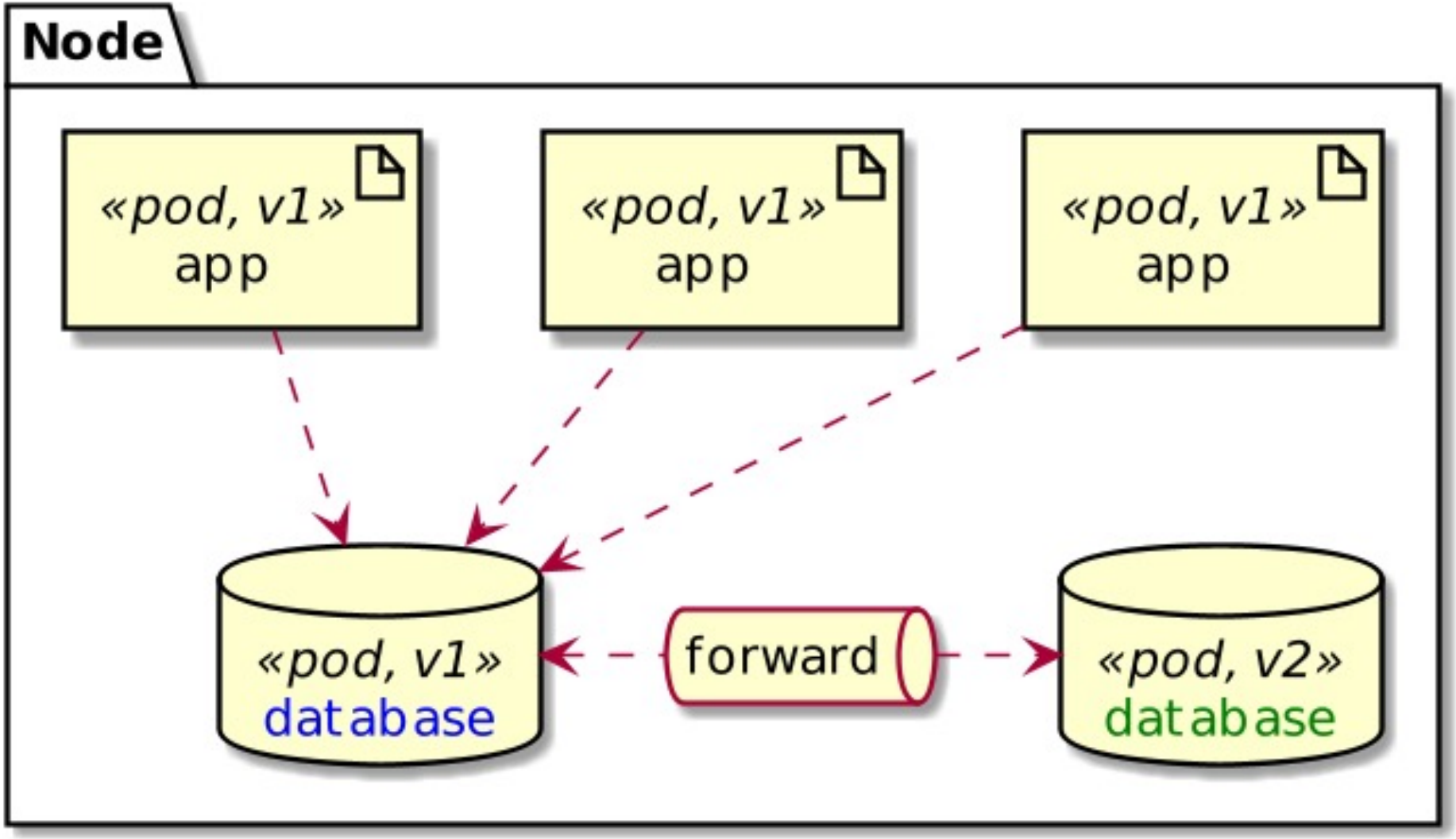
“In databases, Change Data Capture is a set of software design patterns used to **determine and track the data that has changed** so that action can be taken using the changed data.

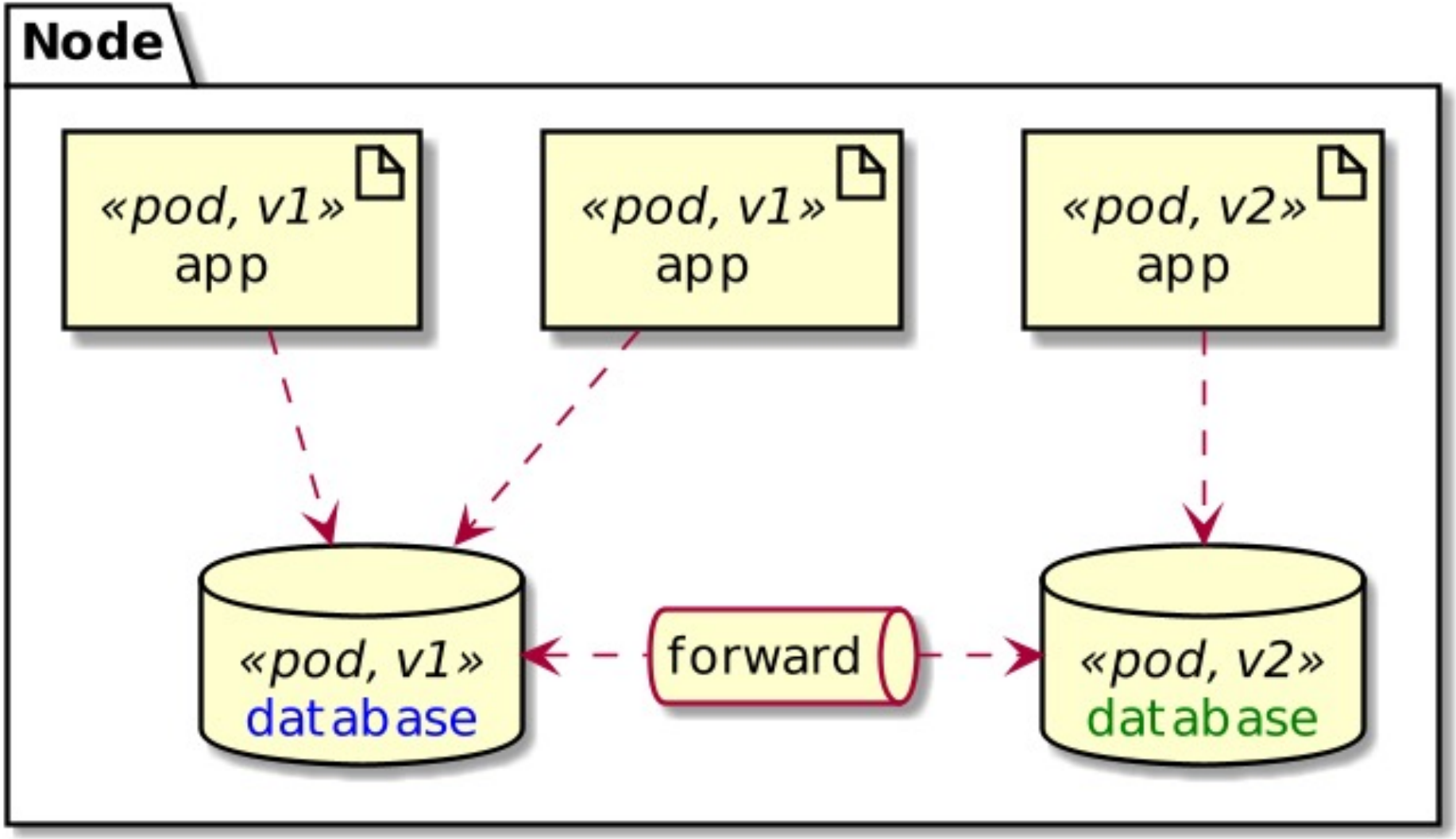
CDC is an approach to data integration that is based on the **identification, capture and delivery of the changes made to enterprise data sources**.”

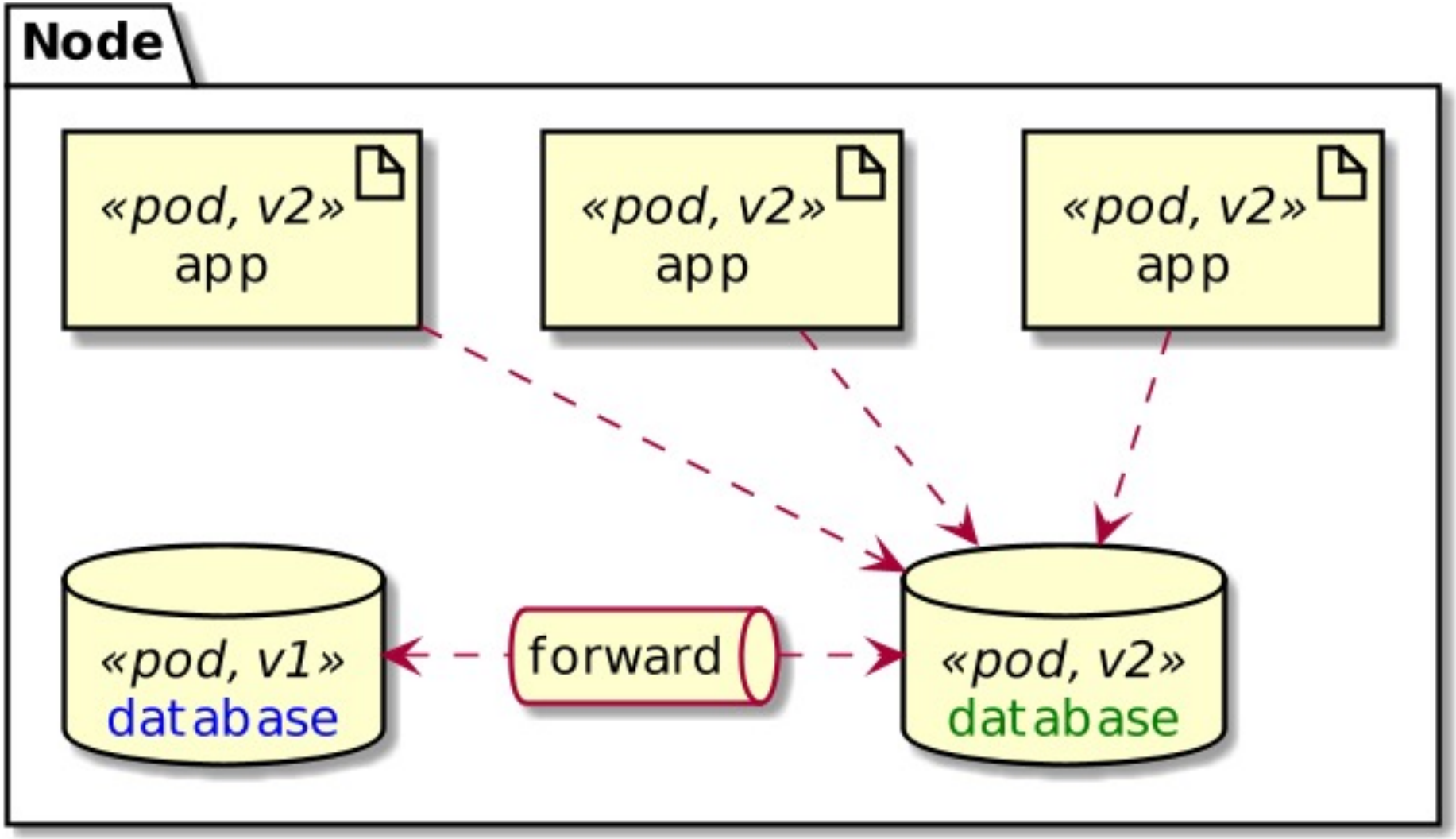
-- https://en.wikipedia.org/wiki/Change_data_capture

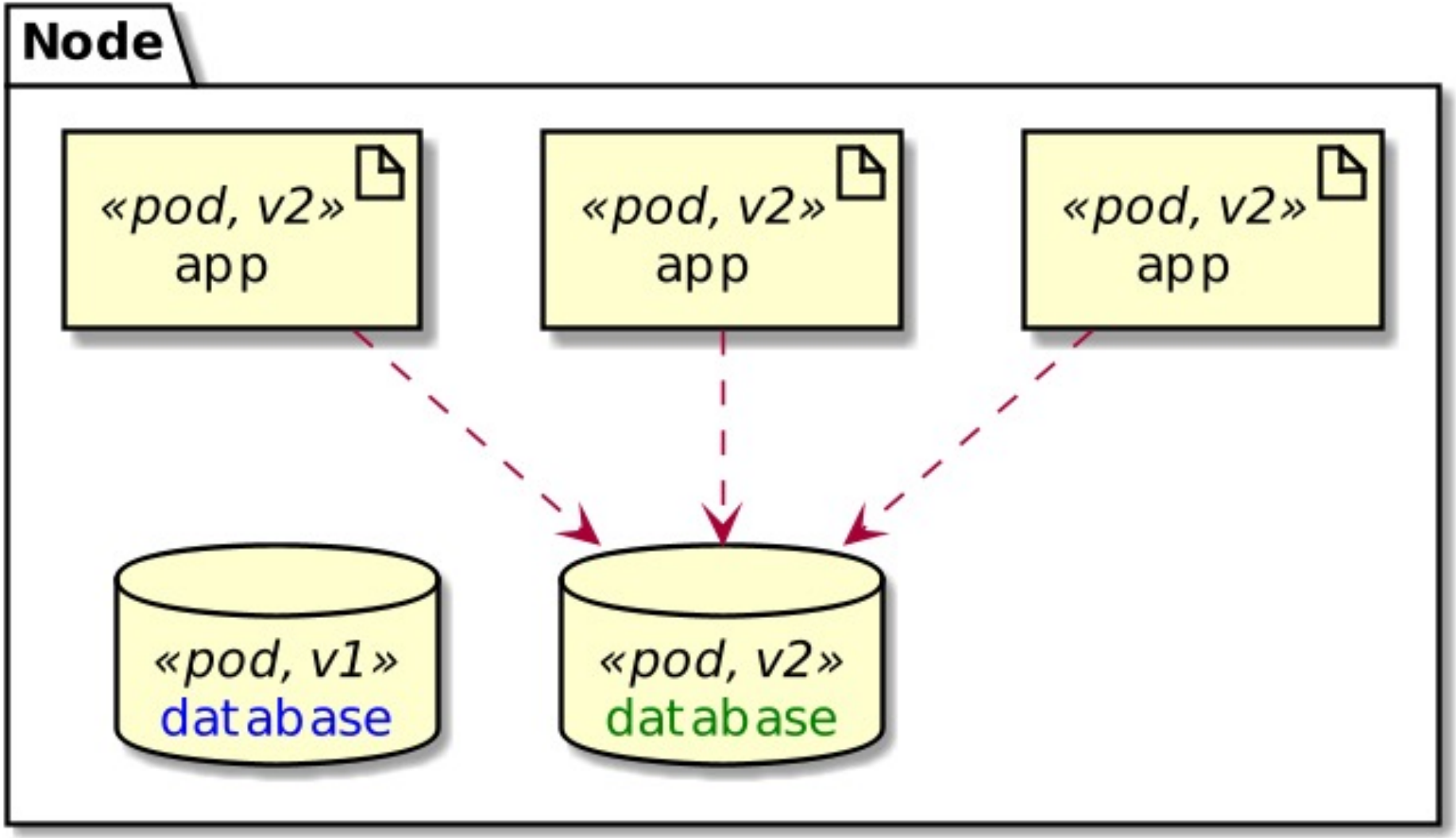

Node
«pod, v1»
app
«pod, v1»
app
«pod, v1»
app
«pod, v1»
database

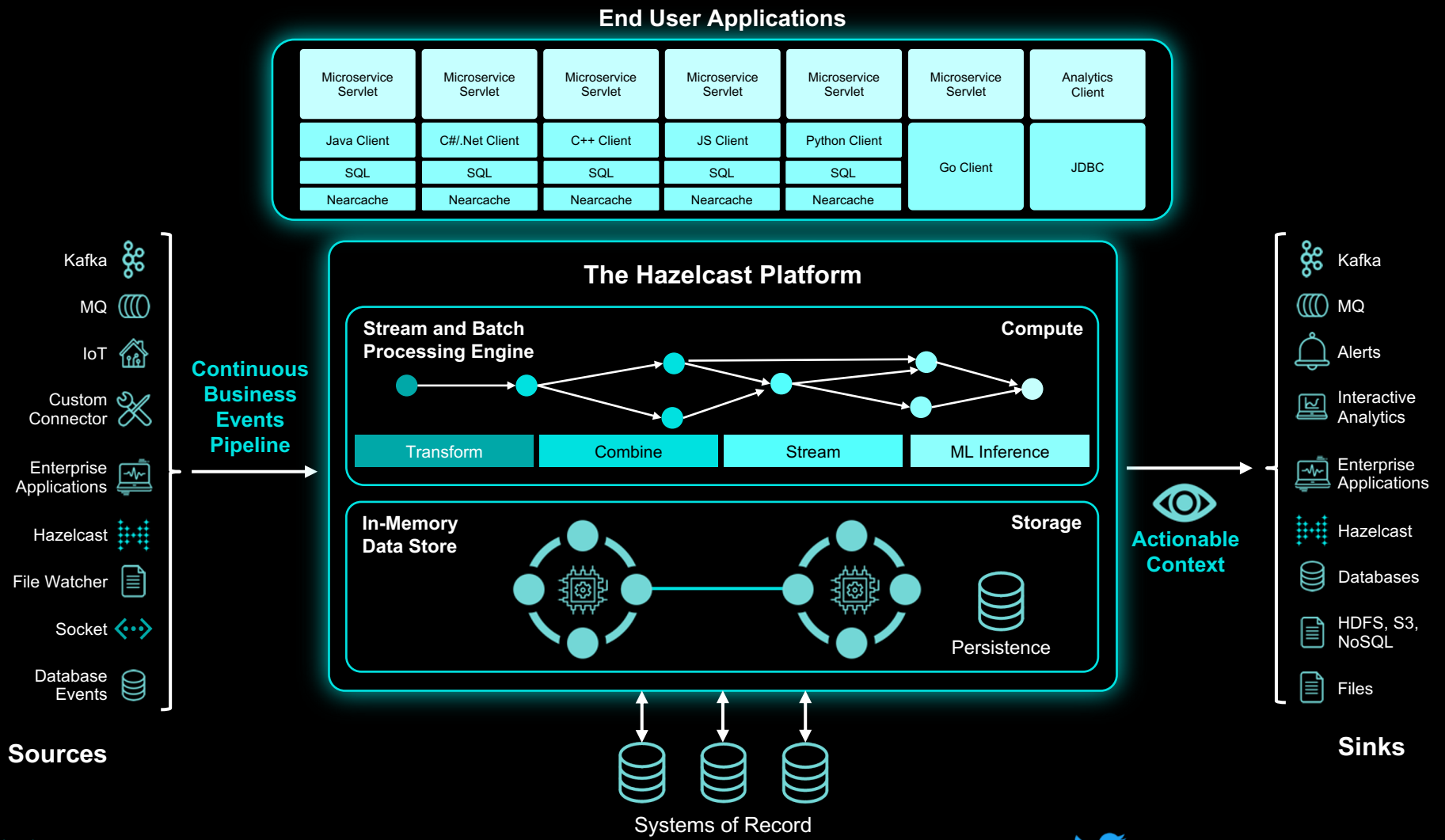

Node
«pod, v1»
app
«pod, v1»
app
«pod, v1»
app
«pod, v1»
database
«pod, v2»
database


Node
«pod, v1»
app
«pod, v1»
app
«pod, v1»
app
«pod, v1»
database
forward
«pod, v2»
database


Node
«pod, v1»
app
«pod, v1»
app
«pod, v2»
app
«pod, v1»
database
forward
«pod, v2»
database

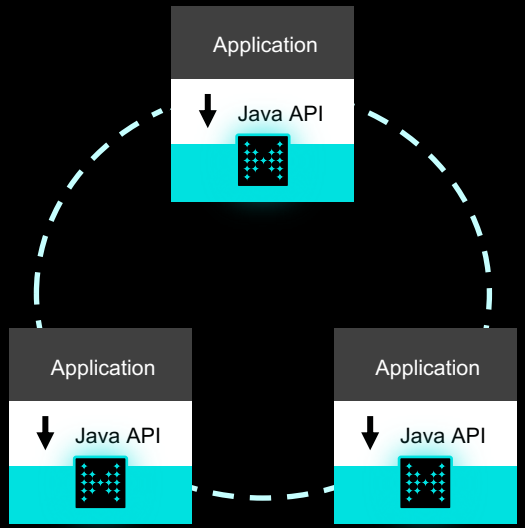

Node
«pod, v2»
app
«pod, v2»
app
«pod, v2»
app
«pod, v1»
database
forward
«pod, v2»
database


Node
«pod, v2»
app
«pod, v2»
app
«pod, v2»
app
«pod, v1»
database
«pod, v2»
database


End User Applications
Microservice Servlet
Microservice Servlet
Microservice Servlet
Microservice Servlet
Microservice Servlet
Microservice Servlet
Analytics Client
Java Client
C#/.Net Client
C++ Client
JS Client
Python Client
Go Client
JDBC
SQL
Nearcache
Kafka
MQ
IoT
Custom Connector
Enterprise Applications
Hazelcast
File Watcher
Socket
Database Events
Sources
Continuous Business Events Pipeline
The Hazelcast Platform
Stream and Batch Processing Engine
Compute
Transform
Combine
Stream
ML Inference
In-Memory Data Store
Storage
Persistence
Systems of Record
Actionable Context
Kafka
MQ
Alerts
Interactive Analytics
Enterprise Applications
Hazelcast
Databases
HDFS, S3, NoSQL
Files
Sinks
HAZELCAST
@nicolas_frankel

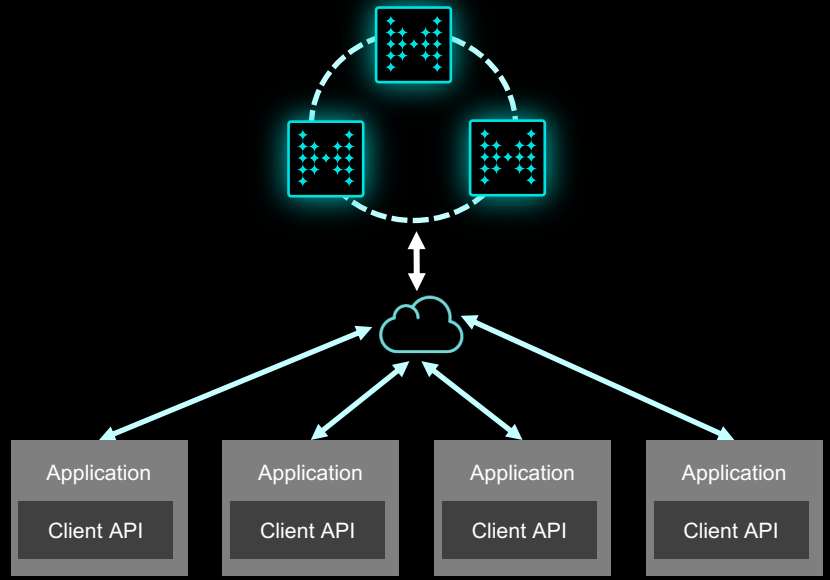
Hazelcast Deployment Options

Embedded Mode



Great for microservices,
OEM and ops simplification

Client-Server Mode

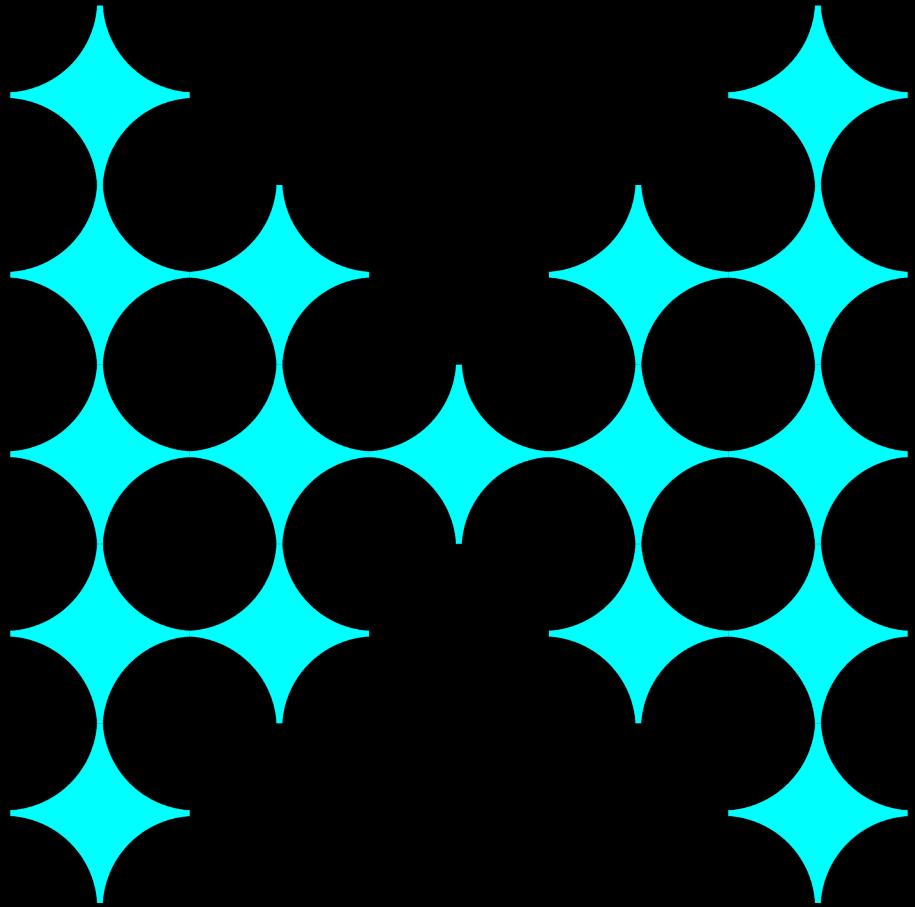


Great for scale-up or scale-out deployments with cluster lifecycle decoupled from app servers

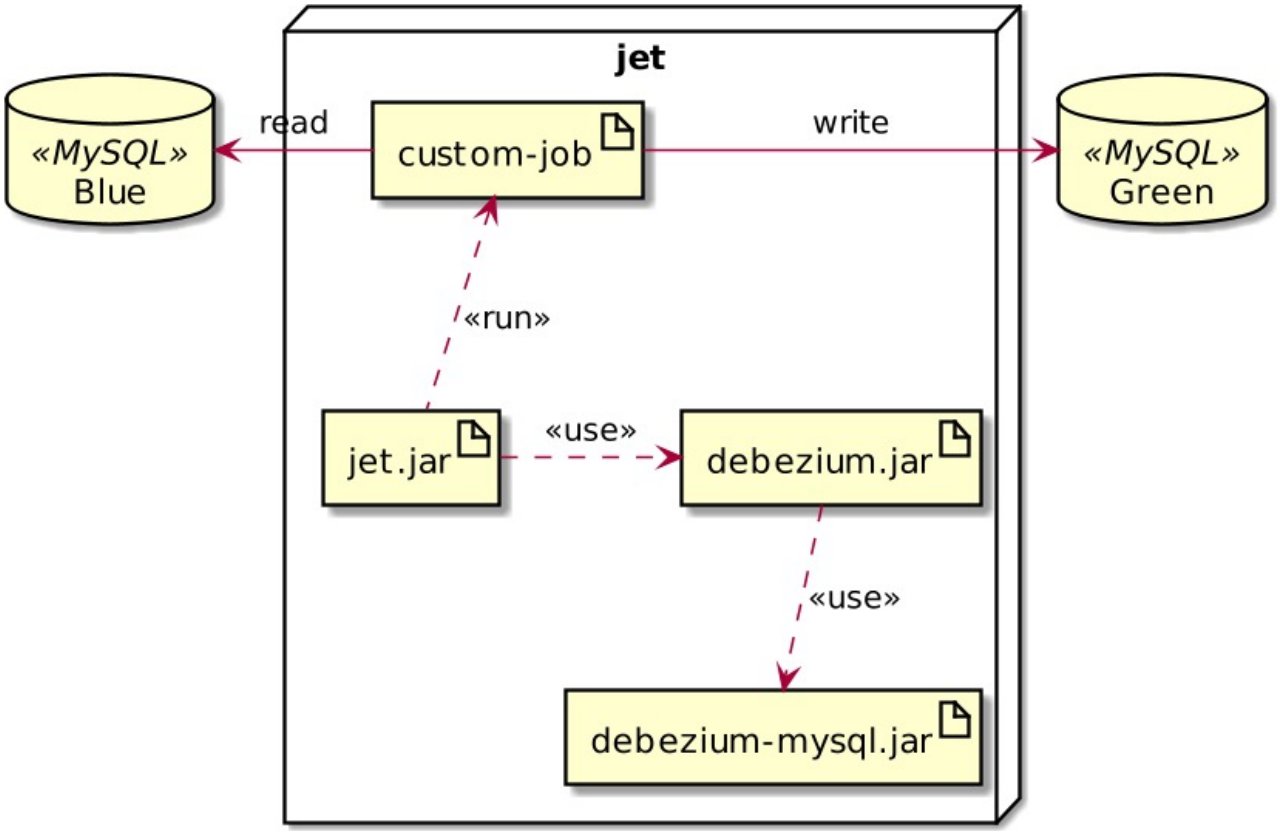
Clients available in Java, Node.js, C#, C++, Python, and Golang

Implementation details

- Hazelcast for Session Replication
  - Via Spring Session
- Hazelcast for CDC
  - With Debezium



Hazelcast Jet & Debezium



Talk is cheap, show me the code!

Takeaways

1. Zero-downtime is within your reach
2. Session replication
3. Change-Data-Capture + Data Streaming for the database

Thanks for your attention!

- https://blog.frankel.ch/
- @nicolas_frankel
- https://bit.ly/zero-downtime
- https://slack.hazelcast.com/
- https://training.hazelcast.com/