

# Building the next generation Cloud Native Database

Benefits and what problems does it solve?

# Introduction

## Sunny Bains

Architect, PingCAP

- Working on database internals since 2001.
- Was MySQL/InnoDB team lead at Oracle.
- @sunbains (Twitter/X)



# Agenda

- Brief history
- Current Architecture
- Serverless Architecture

# Trusted by Global Innovation Leaders

**3,000** + global adopters use TiDB in production

- *Top Banks in China/Global*



- *Top Fintech in the World*



- *Global Top Lighthouses*

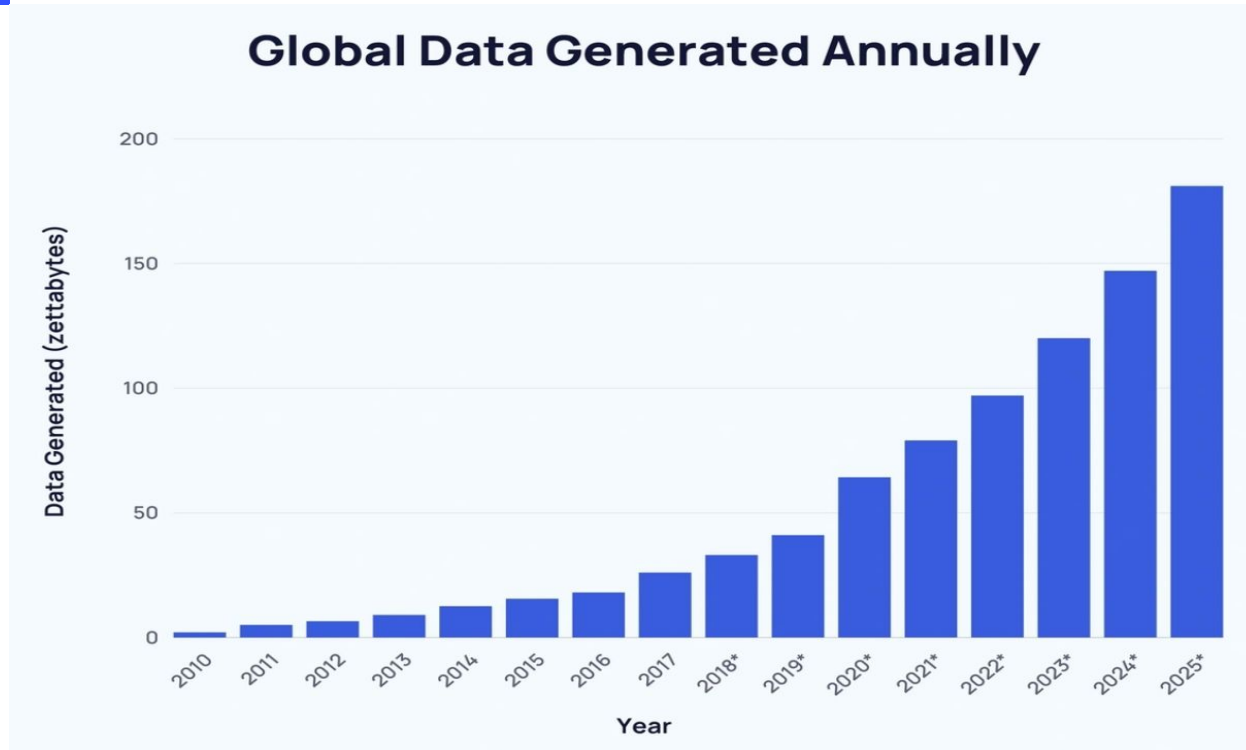


- *E-commerce & Logistics Lighthouses*



**And 4 out of top 5 Cryptocurrency Exchanges**

# Data generation trend



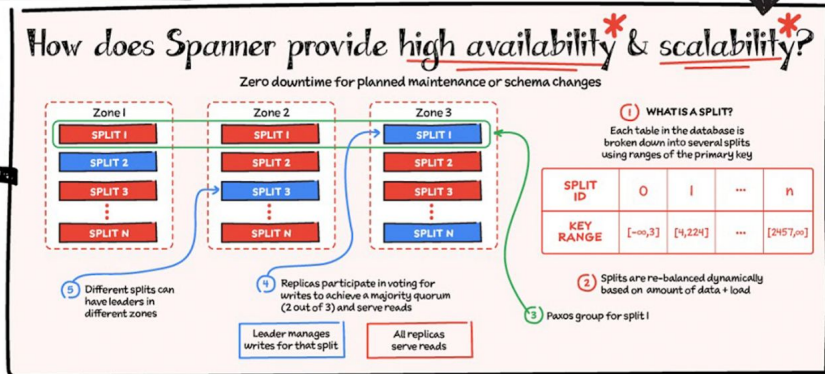
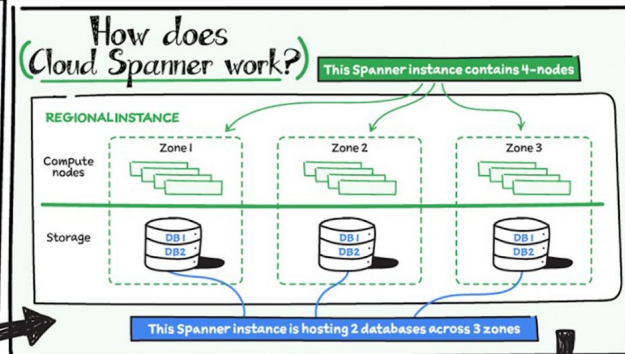
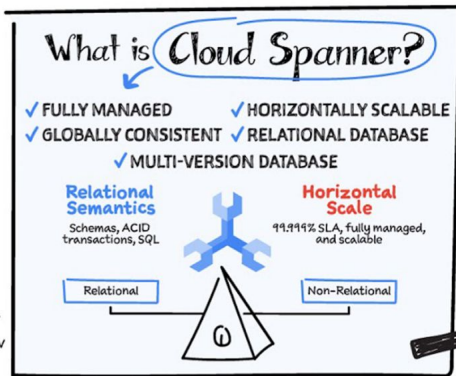
## A little bit of history [2014]

“It was the best of times, it was the worst of times ...”

- Rapid business and data growth
- Hundreds of TB
- Reshard / Rebalance (at midnight)
- Keep strong consistency when a MySQL node went down
- Explicit sharding (and resharding) was required to scale



# We found Spanner (Paper)



# But...



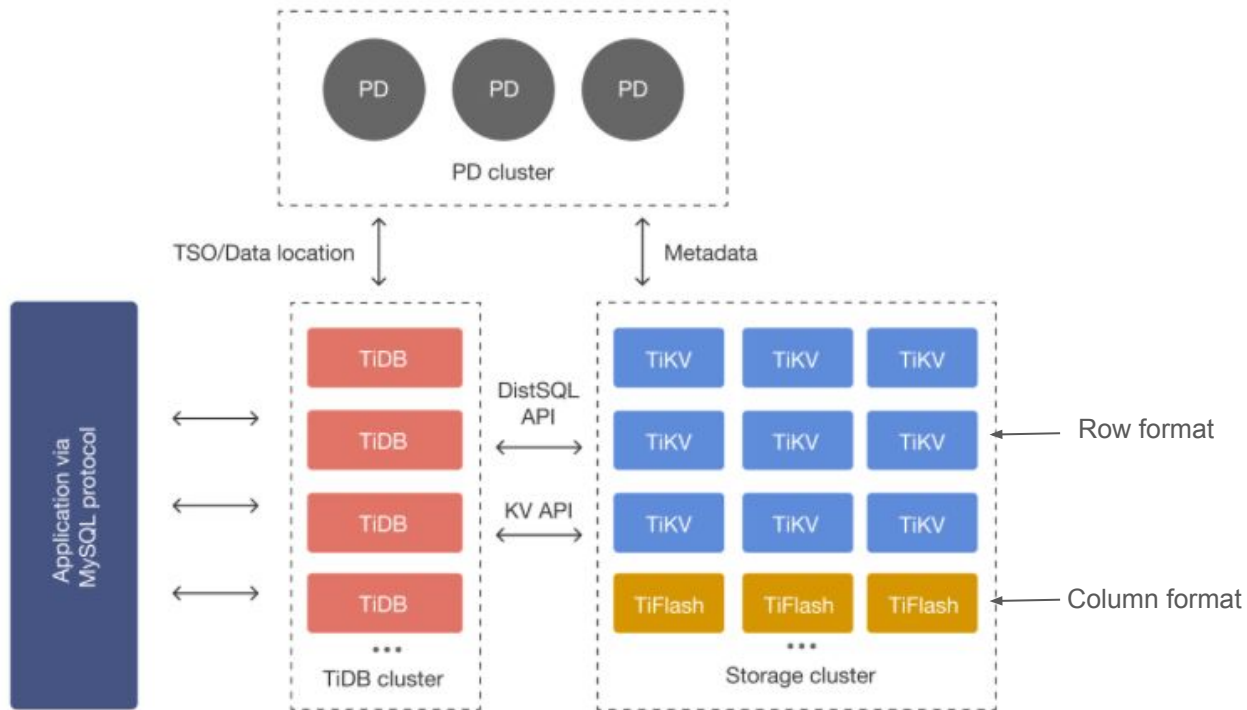
- Proprietary, **not open-source**
- Special hardware is required (TrueTime)
  - We wanted to run it on commodity hardware
- Specialized and proprietary APIs, not standard SQL and wired protocol



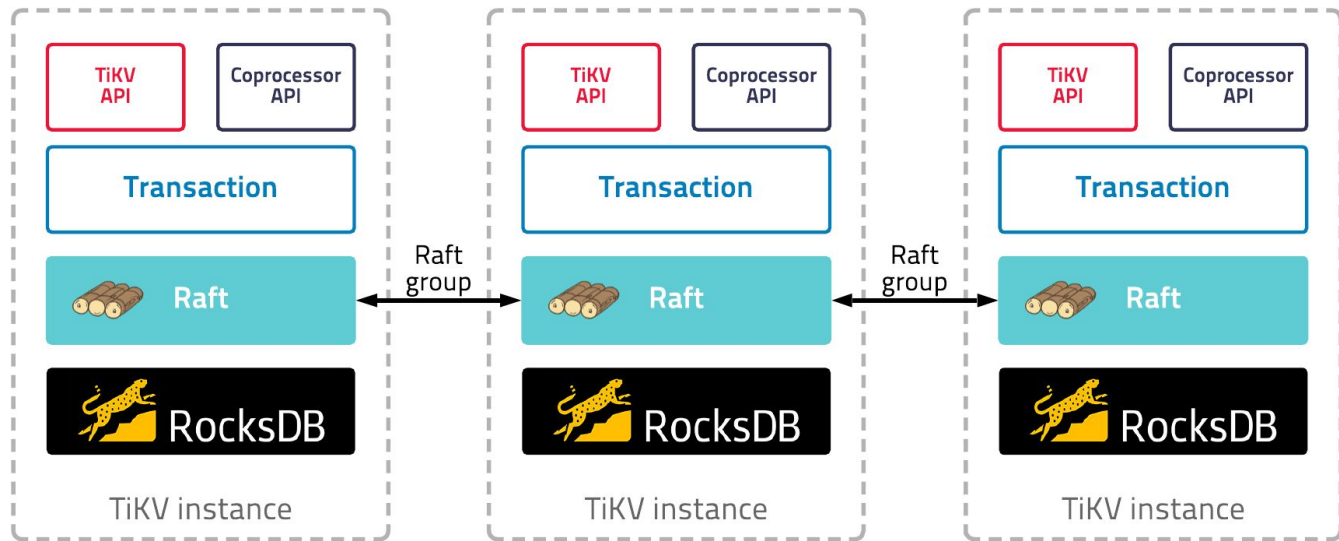
# TiDB Design Philosophy

- **Shared-nothing architecture**
  - Developers should not be concerned with shard details
  - Developers should have the flexibility to control data placement
- **Standard SQL**
  - Execution engine should determine the details of distributed execution
- **Distributed Transactions**
  - Strong transactional consistency guarantees out of the box
  - Flexible read consistency policies should be provided
- **Built-in Highly Available**
  - Not at the cost of **strong consistency**

# TiDB: Reference Architecture

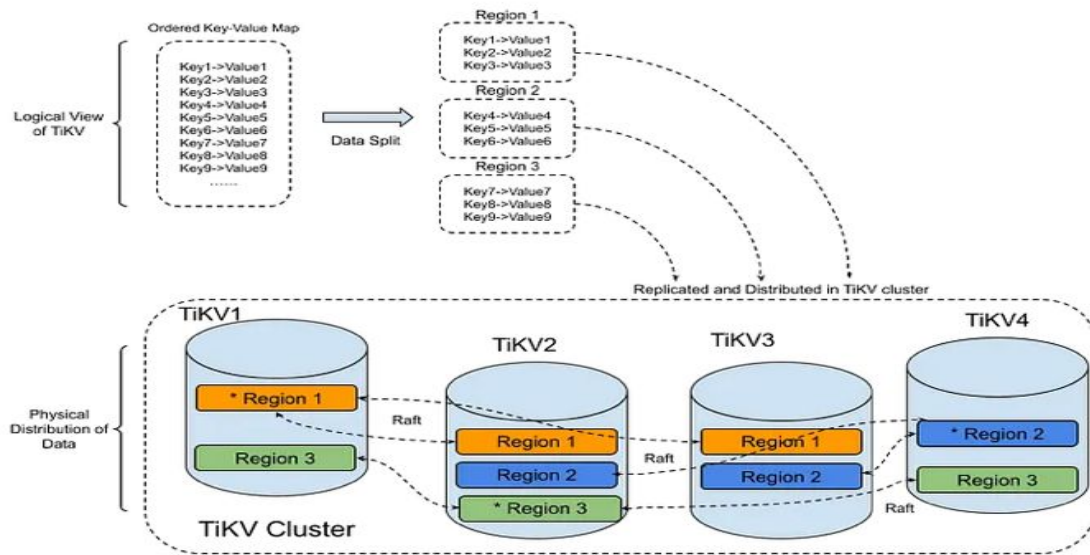


# TiKV: Distributed KV Storage [Built in Rust]



# TiKV - Data Storage Example

Example to illustrate how TiKV partitions and manages the data



# TiDB Region

A **Region** is TiKV's **logical** scale unit

- Operations such as load balance, scale-out, scale-in are all based on **region**
- **Regions** are replicated using the Raft consensus protocol
  - A replicated **region** is called a Raft group
  - **Regions** are spread across the nodes in the cluster
  - A single storage node contains many **Regions**
  - **Regions** are stored on RocksDB, there is one instance of RocksDB per storage node.
  - Rows in a **Region** are ordered

# Distributed Transactions in TiDB

- TiDB supports Read-Committed and Snapshot Isolation levels
  - The Snapshot Isolation is mapped to MySQL/InnoDB's Repeatable Read
- TiDB uses an optimized version of the Percolator algorithm for distributed transactions
- A transaction requires a start time stamp and a commit timestamp
  - PD is responsible for handing out these timestamps
  - These timestamps are used in TiDB's MVCC implementation
- Async commit in TiDB
  - The SQL nodes are the Txn Coordinators (TC)
  - The TiKV nodes are the participants
  - Works well when the transaction write set is small and Phase II time dominates
- Supports 1PC Commit Optimization
  - If transaction only updates a non-index column of a record
  - Or, Inserts a record without a secondary index,
  - Only involves a single Region

# Challenges Yet To Be Fully Addressed

- How to improve stability at scale?
  - Copying data, LSM compaction may slow down your workloads
  - Provisioning can take time
- How to take scalability to the next level?
  - Exploding data volume, customers asking for 200PB clusters
  - Multiple applications share a single cluster for greater efficiency and lower costs
- How to make it easier and more cost effective for users?
  - Maintenance burden of a distributed system, trade-offs become “knobs”
  - Scale down when the workload reduces



# Leverage the Cloud Infrastructure



- Allow a developer to start small and scale to any size
- Better and more varied capabilities e.g., disaggregated storage, use S3
- Elasticity and resilience
- Hide the complexity of a distributed system
- Integrate the database with the capabilities of cloud
- The only cost of a quiescent instance should be storage

# Leverage Cloud Storage

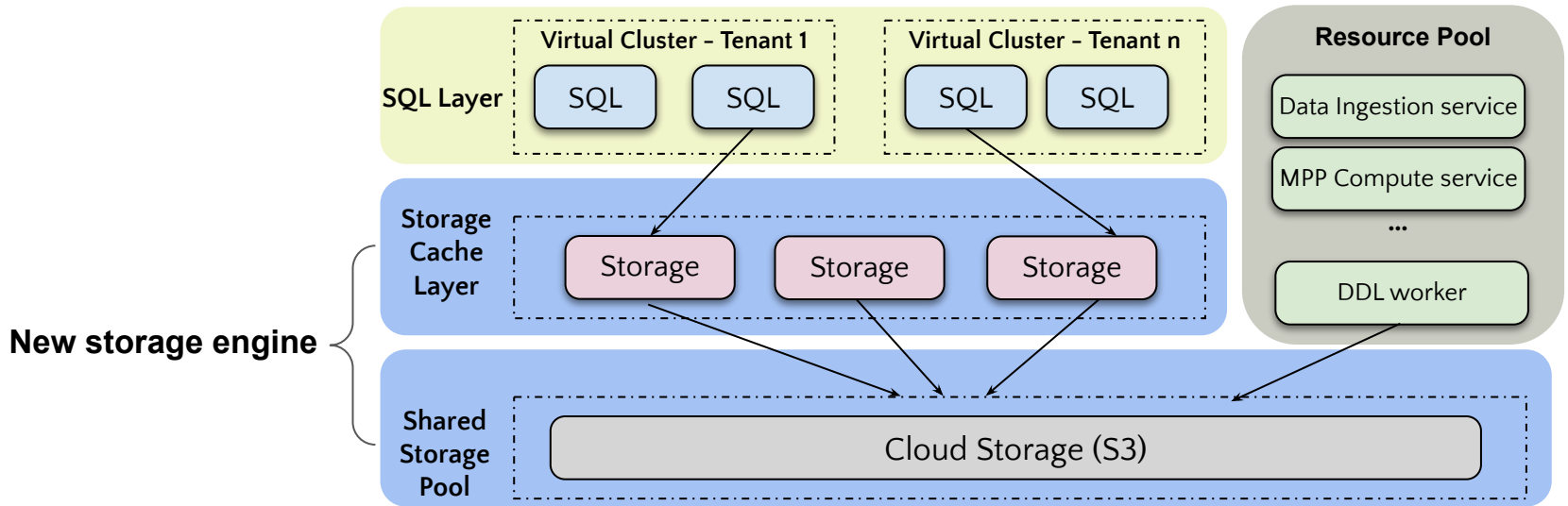
Storage Option	Storage Cost	Storage Limit	Durability	Latency	Request Cost
Object Storage (e.g., S3)	<b>Low</b>	<b>Unlimited</b>	<b>High</b>	High	High
Block Storage (e.g., EBS)	High	Limited	Low	<b>Low</b>	<b>None</b>

**Best of both worlds**

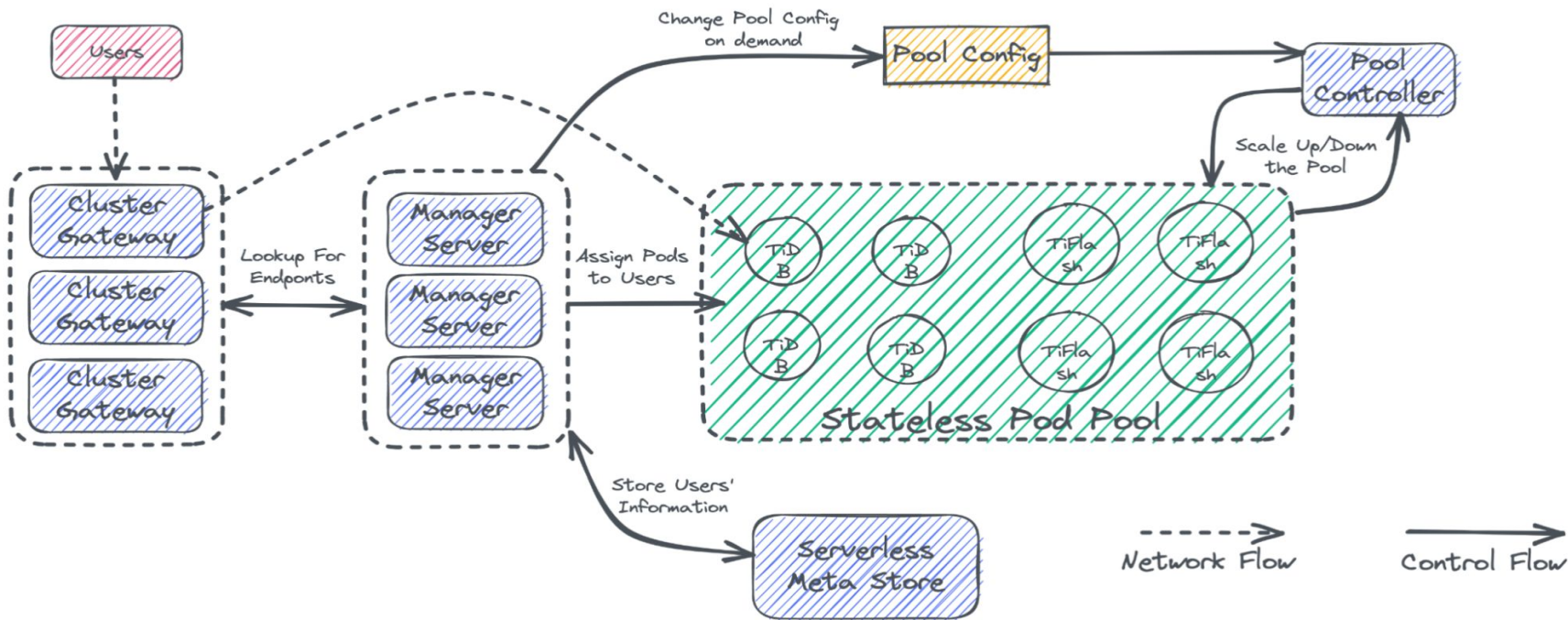
# TiDB Serverless Architecture

- Multi-tenant architecture
- Disaggregated storage (new LSM storage engine, replaces RocksDB)
  - Local disk or memory as write through cache
  - Cold data on S3
  - Data shared by row & column store
- Shared resource pool [ spot instances ]
  - For background tasks
  - Heavy compute load
- Remove LSM WAL, Raft log serves as the only log

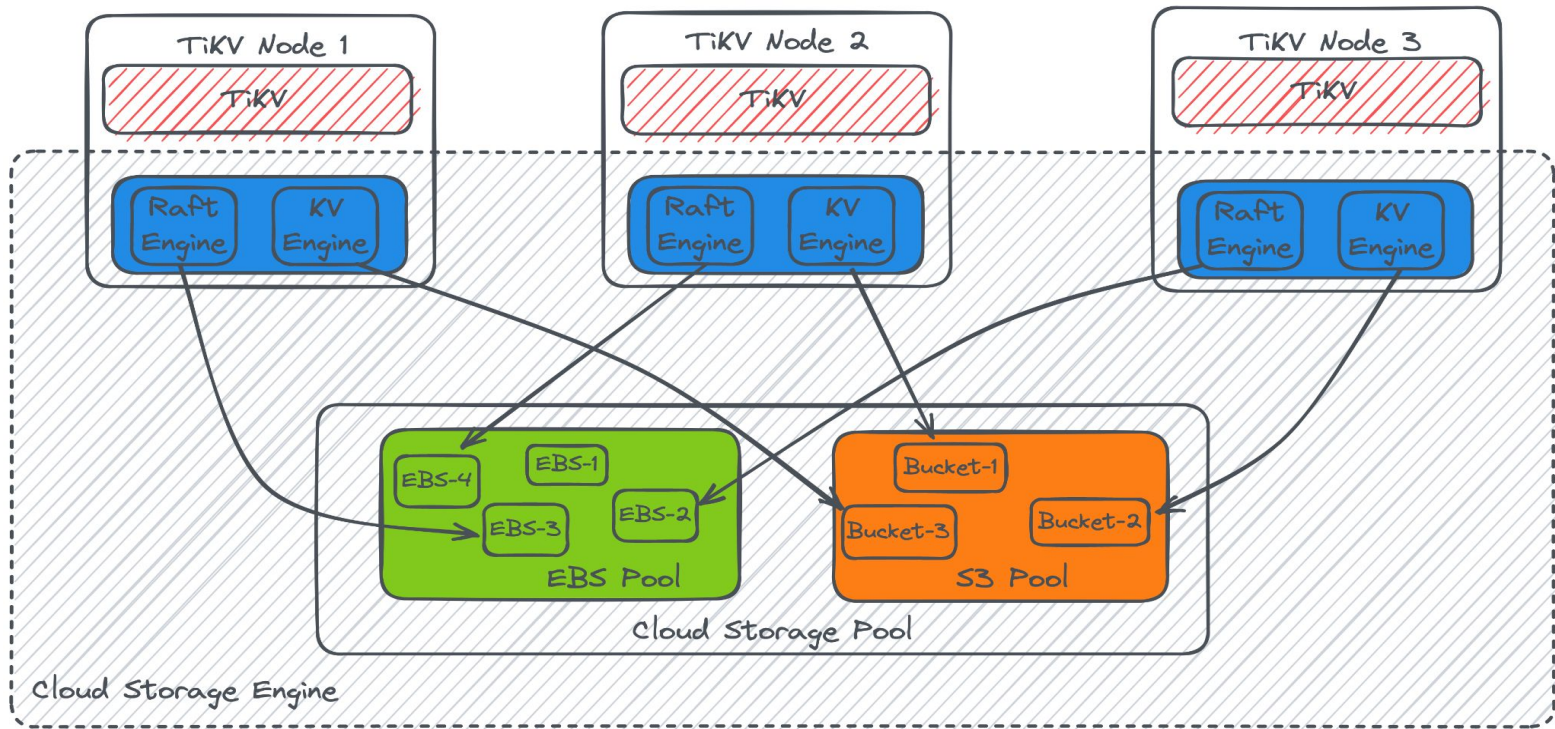
# TiDB Serverless Architecture



# Cloud Storage Engine Architecture



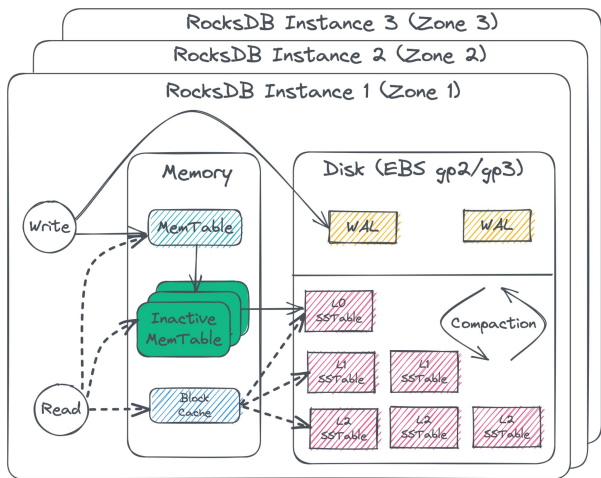
# Cloud Storage Engine Architecture





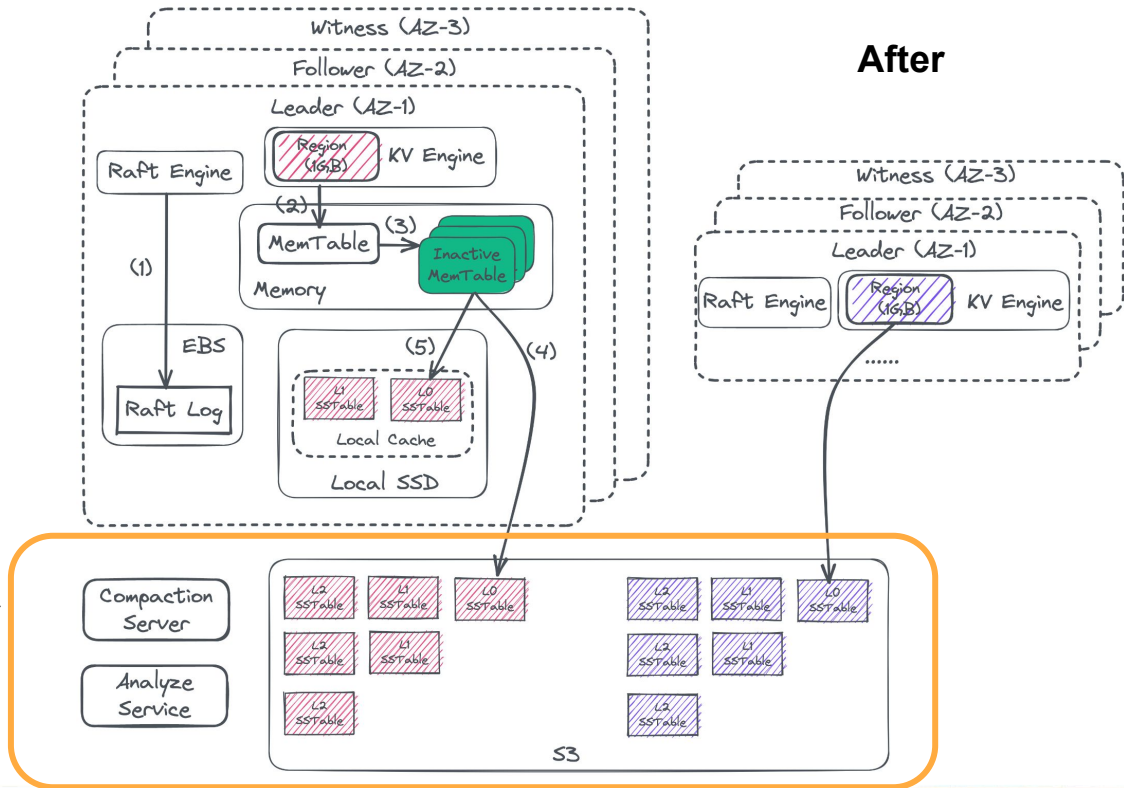
# The New Cloud-Based Storage Engine

**Before**



**Remote Storage and Services**

**After**





# THANK YOU.

