



# Amazon DocumentDB

Deep Dive

Ryan Thurston & Karthik Vijayraghavan



# Table of contents

- What is Amazon DocumentDB?
- Use cases & case studies
- What is unique about the modern, cloud-native architecture of Amazon DocumentDB?
- What's new?
- What's next?

# Comprehensive set of services across Databases & Analytics



Aurora

RDS

DynamoDB

DocumentDB

ElastiCache

Keyspaces

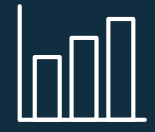
Redshift

EMR

Elasticsearch Service

Managed Streaming for Apache Kafka

QuickSight



Relational databases

Non-relational databases

Data Warehouses

Hadoop and Spark

Operational analytics

Real-time analytics

Business Intelligence

ORACLE

MySQL

Microsoft SQL Server

PostgreSQL

MariaDB

mongoDB

Apache CASSANDRA

redis

M

teradata

ORACLE

Microsoft SQL Server

cloudera

hadoop

APACHE SPARK

elasticsearch

logstash

kibana

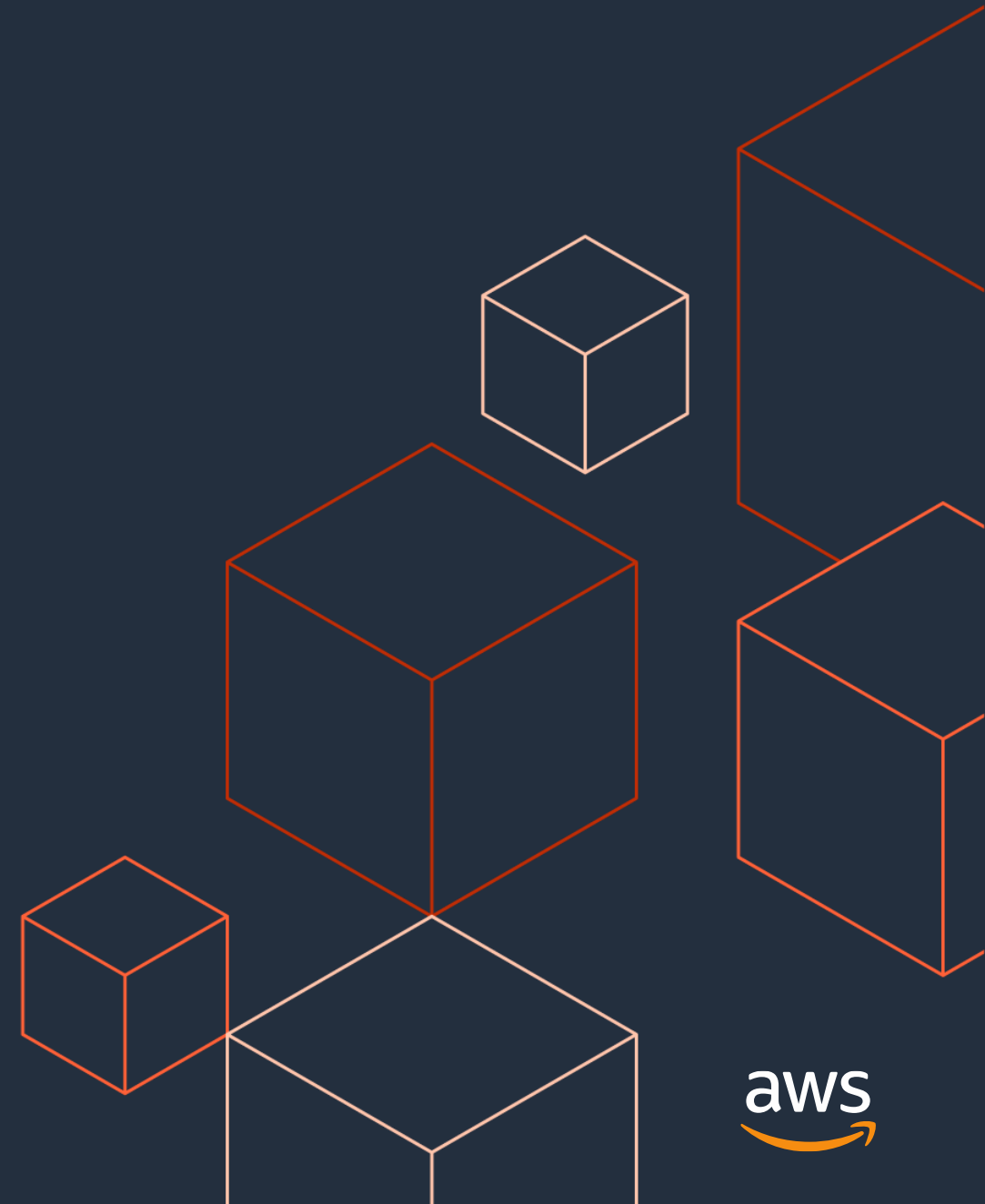
kafka

SAP

MicroStrategy

tableau

# What is Amazon DocumentDB?



# Amazon DocumentDB (with MongoDB compatibility)



Fully  
managed



Scalable



MongoDB API  
compatible

Fully managed and scalable document database service that supports MongoDB workloads

# Amazon DocumentDB (with MongoDB compatibility)



Fully  
managed

Built-in high availability

---

Backups enabled by default

---

Durable by default

---

Security best practices by default

---

Automatic patching

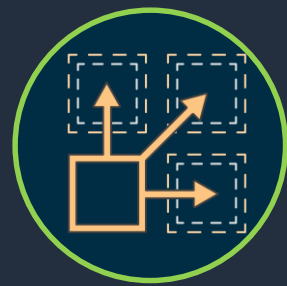
---

Monitoring and alerting



"Because DocumentDB is a fully managed service, our databases are scalable, highly available, backed up, and encrypted without any overhead from our engineering teams"

# Amazon DocumentDB (with MongoDB compatibility)



Scalable

Scale compute in minutes

---

Storage and IO autoscaling

---

Storage scales to 64TB

---

Scale out to 15 replicas for millions of reads



“With Amazon DocumentDB, we can add or scale instances in minutes, regardless of data size.”

# Amazon DocumentDB (with MongoDB compatibility)

Applications, drivers, and tools can be used with Amazon DocumentDB with little or no change

---

Supports hundreds of APIs, operators, and stages

---

Continually working backward from customers to deliver the capabilities they need



MongoDB API compatible



“We love that it’s compatible with MongoDB, so our applications didn’t require code changes, and we could easily spin up a DocumentDB cluster to test the MongoDB capabilities that we relied on.”



# When should you use a document database?



JSON data



Flexible schema for fast iteration



Ad hoc query capabilities



Flexible indexing



Operational and analytics workloads

Amazon DocumentDB makes it easy to store, query, and index JSON data

# Customers



Web publishing



Product catalog



IoT



Regulatory documents

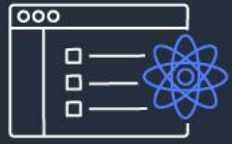


Gaming



Content management

# Use Cases



## Content Management

- News articles
- Blogs
- Recipes
- Patient records



## Mobile

- Native JSON
- Easy to store data you're collecting back and forth between devices



## Personalization

- Customize offers based on transaction history or financial data such as credit scores



## Catalogs

- Outputs of ML experiments
- Inventory descriptions
- Pharmaceutical trials



## Retail & marketing

- Track customers who purchase similar items
- Custom marketing campaigns



## User profiles

- Collecting game data
- Game management
- Any online profile

*Complex documents that are dynamic/changing and may require ad hoc querying, indexing, and aggregations*

# Case study: Rappi – LATAM unicorn startup

## Problem Statement

Need database platform to address:

1. Frequent outages
2. Rapid pace of innovation
3. Slow Performance

## Solution advantages

- No. of Outages : **Zero**
- Latency : 500 ms to 80 ms ↓
- Operational overhead: 50% ↓
- 100+ migrations in three months ↑

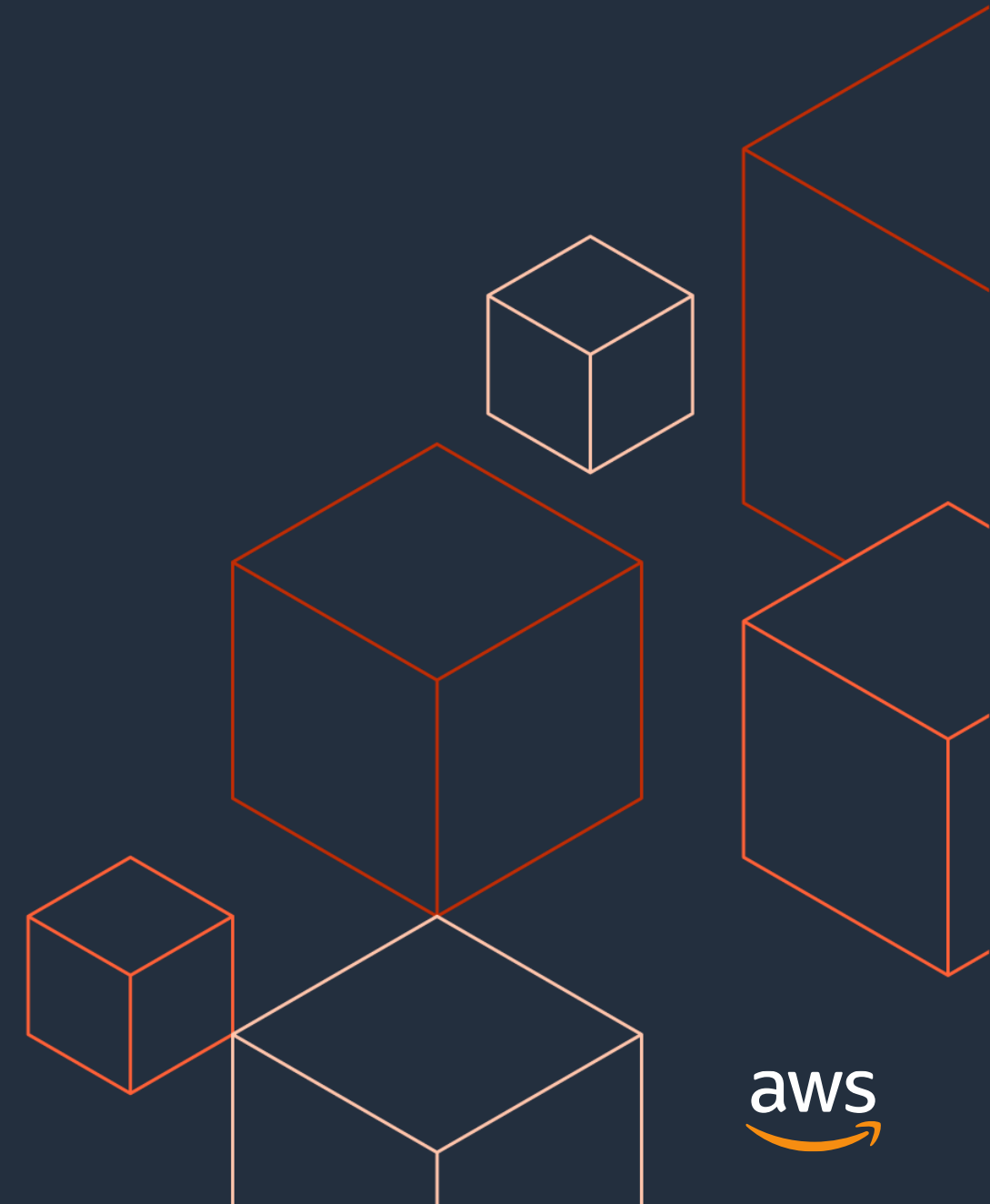
## Requirements

- Compatibility with MongoDB
- Independently scale microservices.
- Improve performance of complex ranking queries
- Seamlessly migrate to fully managed database service.

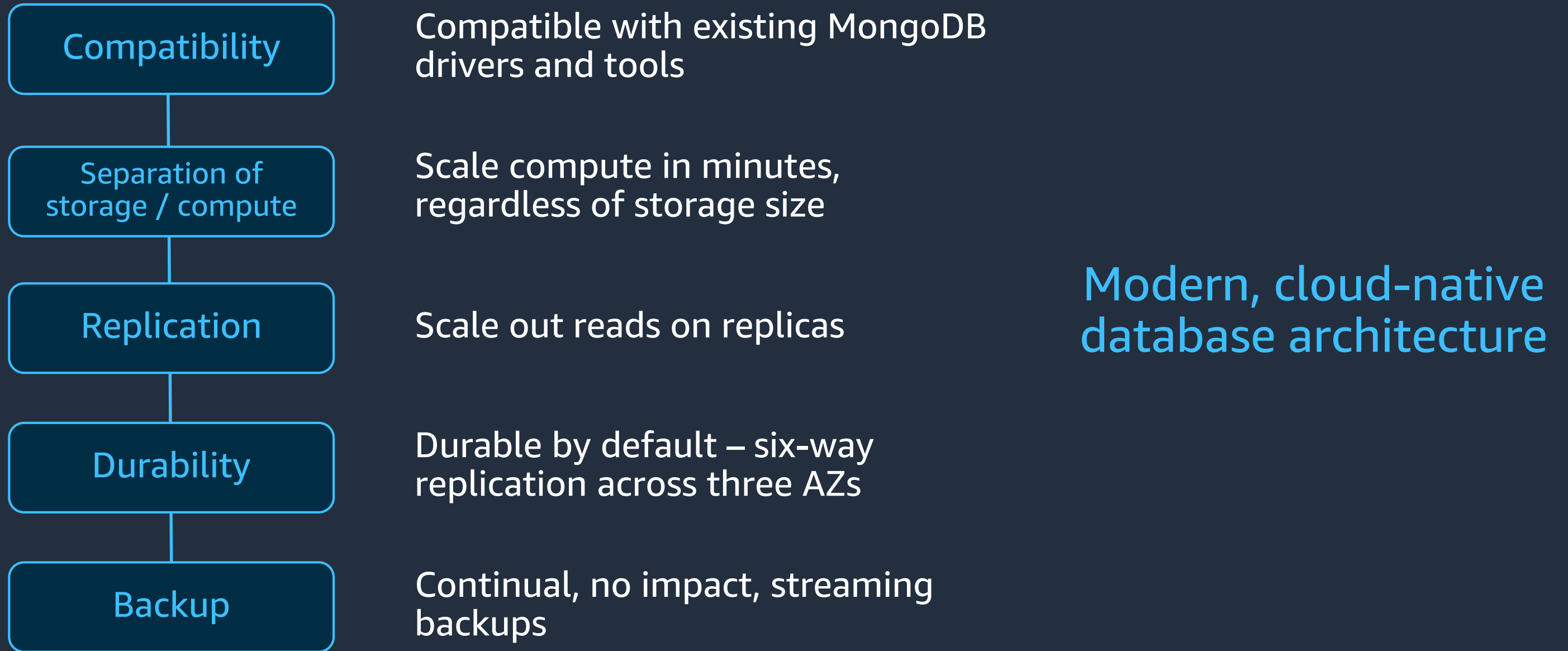
## Solution

- One cluster per LOB to scale microservices with 55% fewer instances.
- Read replicas to reduce latency for ranking queries by 16 times.
- MongoDB compatibility for minimal code-change migration, resulting in 60% FTE cost savings.

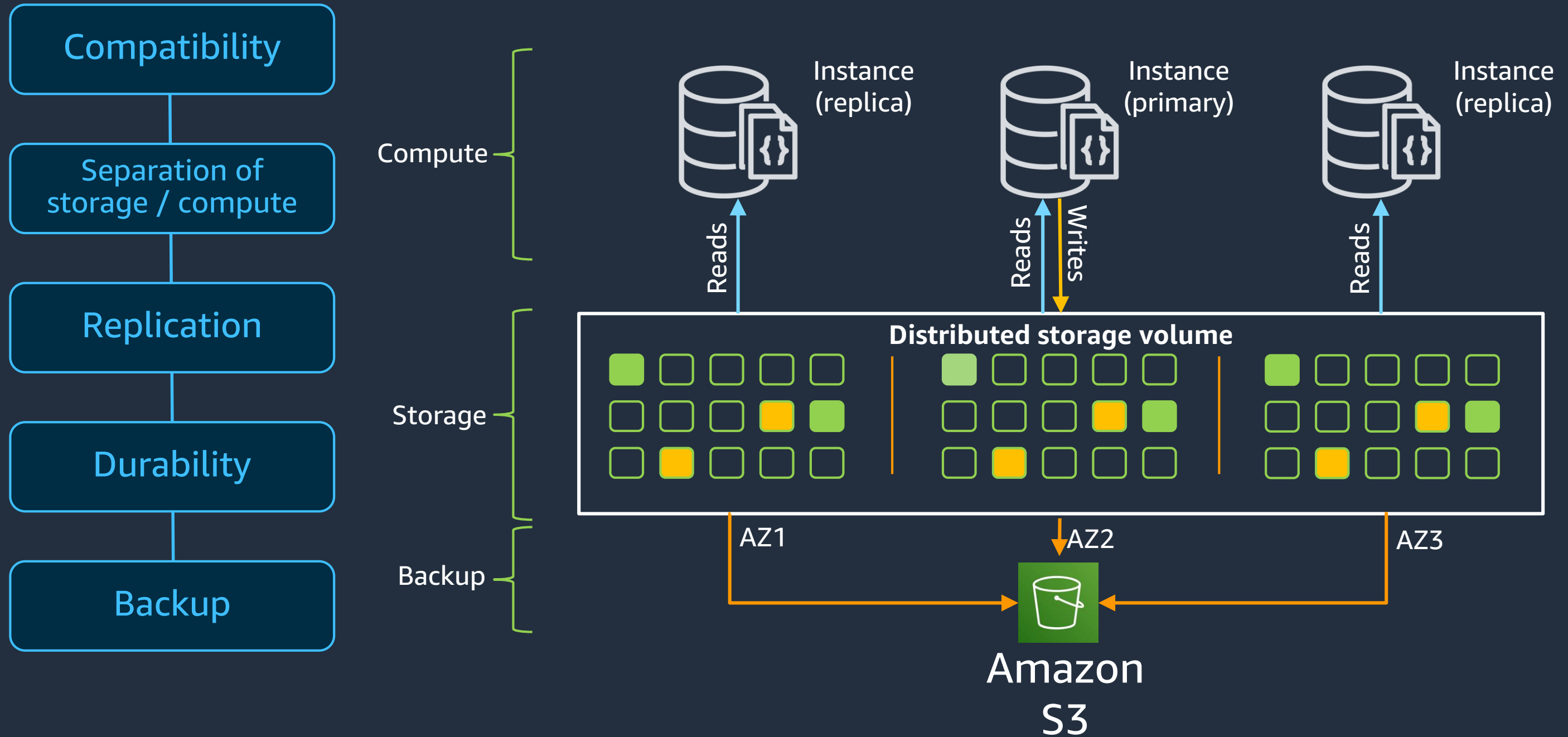
# Amazon DocumentDB Architecture



# Amazon DocumentDB Architecture



# Amazon DocumentDB Architecture



# Demo: Console overview





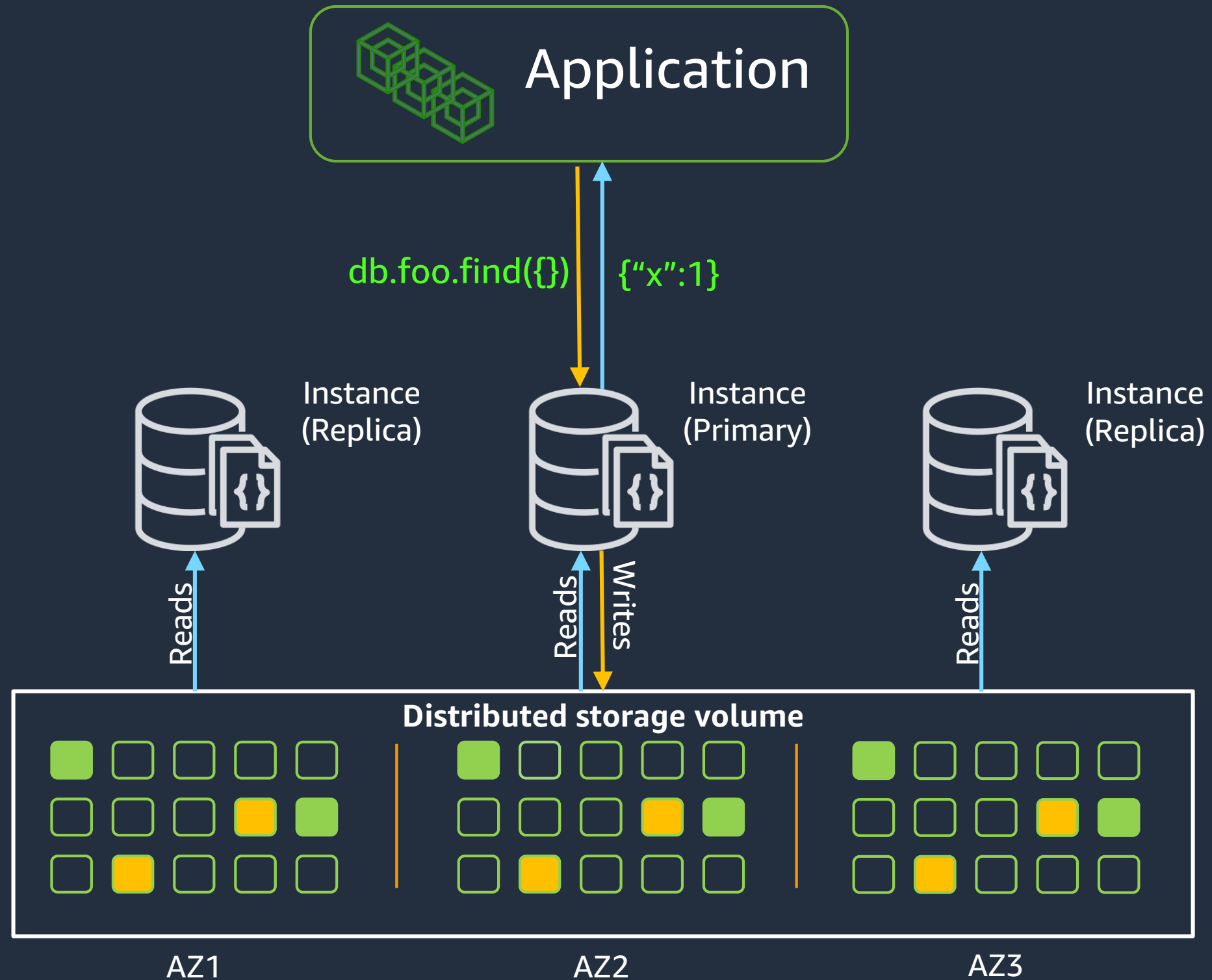
# Compatibility



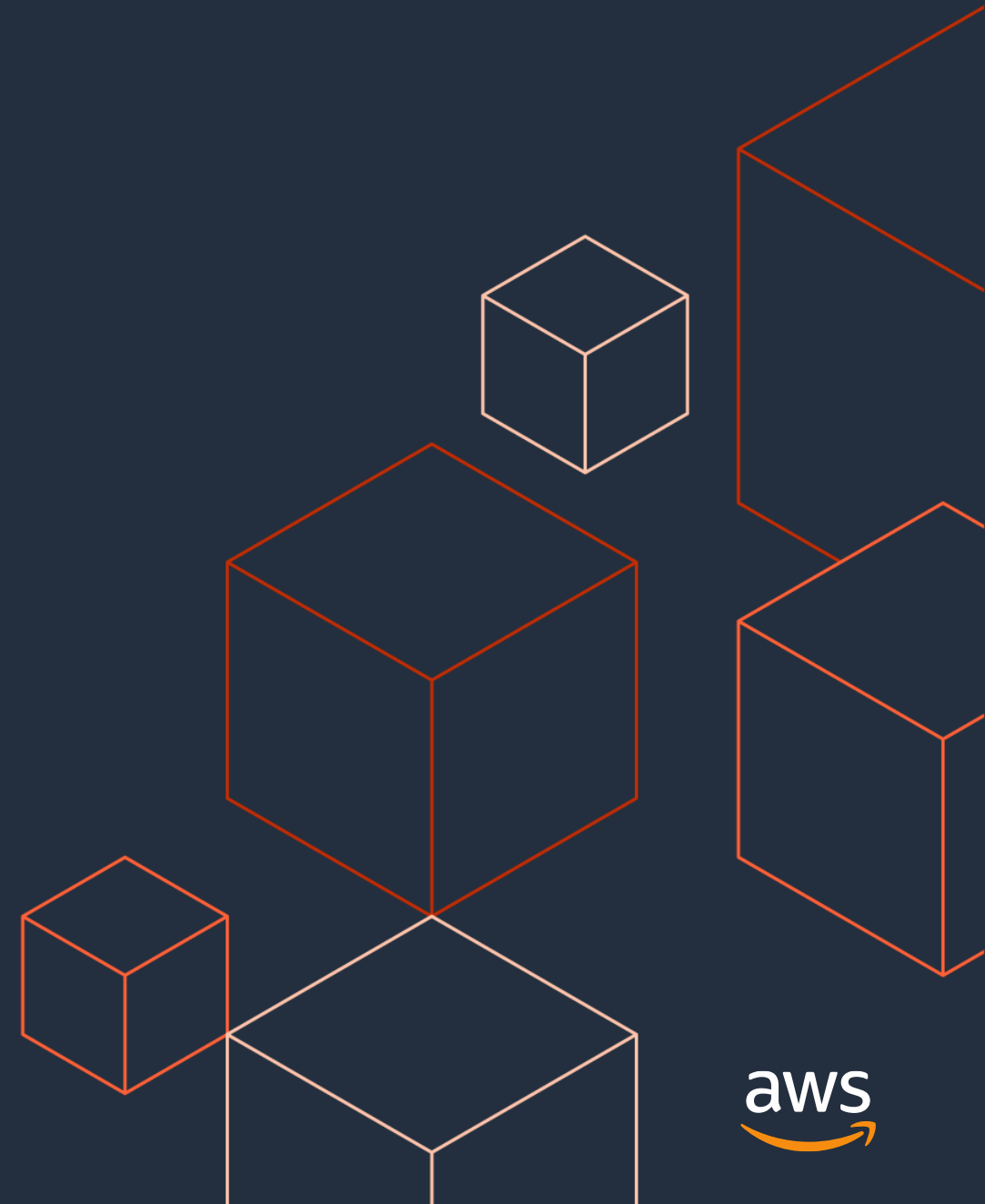
# Architecture

## Compatibility

Amazon DocumentDB emulates the MongoDB API



# Demo: Connecting to DocumentDB



# Demo code

```
1 import pymongo, boto3
2 from secretsManager import get_secret
3
4 secret = get_secret() ## Method to get secrets from Secrets Manager
5
6 ## Create mongo client with username and password from AWS Secrets Manager
7 client = pymongo.MongoClient(
8     secret['host'],
9     secret['port'],
10    username=secret['username'],
11    password=secret['password'],
12    ssl='true', ## TLS Enabled by default
13    ssl_ca_certs='rds-combined-ca-bundle.pem',
14    retryWrites='false',
15    replicaSet='rs0', ## Connect as a replica set
16    readPreference='secondaryPreferred' ## Reads are sent to replicas
17    ## DocumentDB implments the best practice of highly durable writes (write quorum of 4)
18    ## w='majority',
19    ## j = true
20 )
21
22 db = client.test ##Get the test database
23 db.col.insert_one({'x':'AmazonDocumentDB'}) ## Insert a doc(request routed to Primary)
24 x = db.col.find_one() ## Find a document (request routed to replica)
25 print(x) ## Print to screen
26 client.close() ## Close Client
```

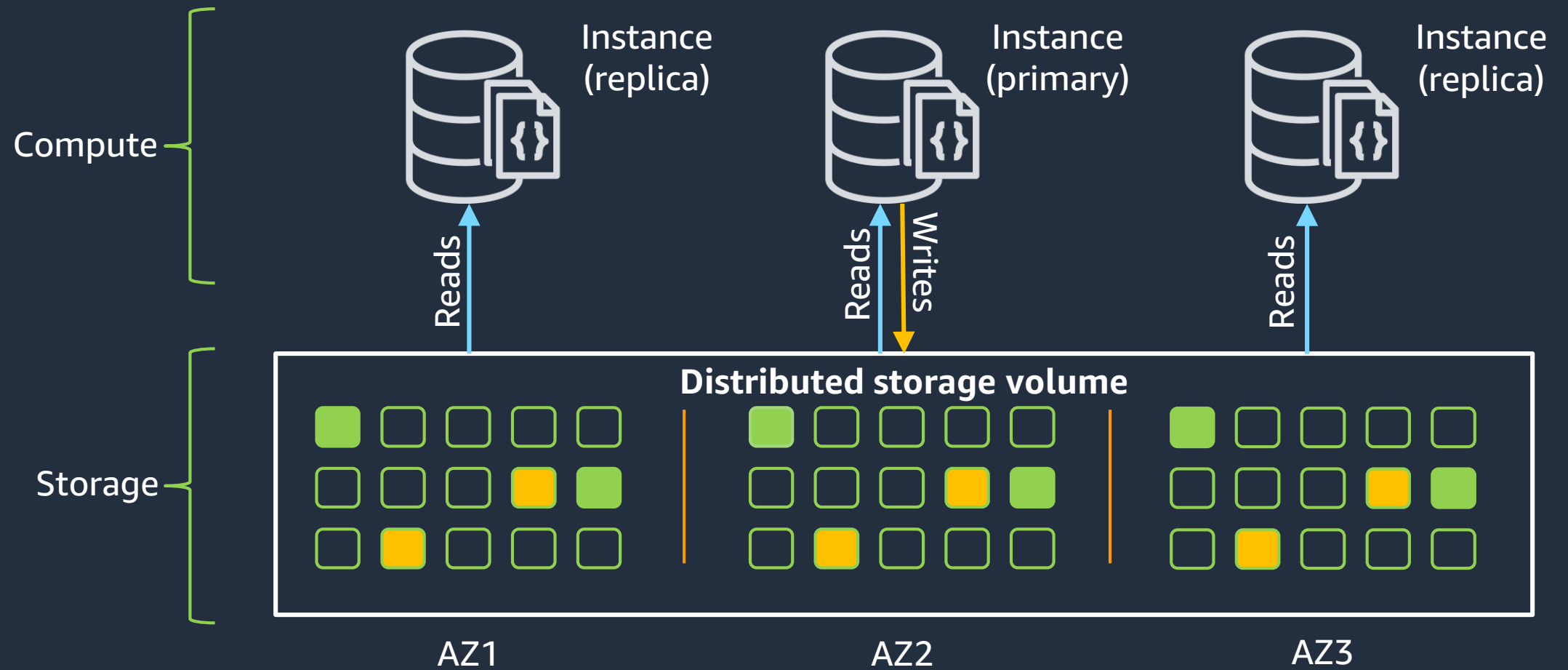
# Separation of storage and compute



# Architecture

Separation of storage / compute

How would you build a cloud-native database architecture?

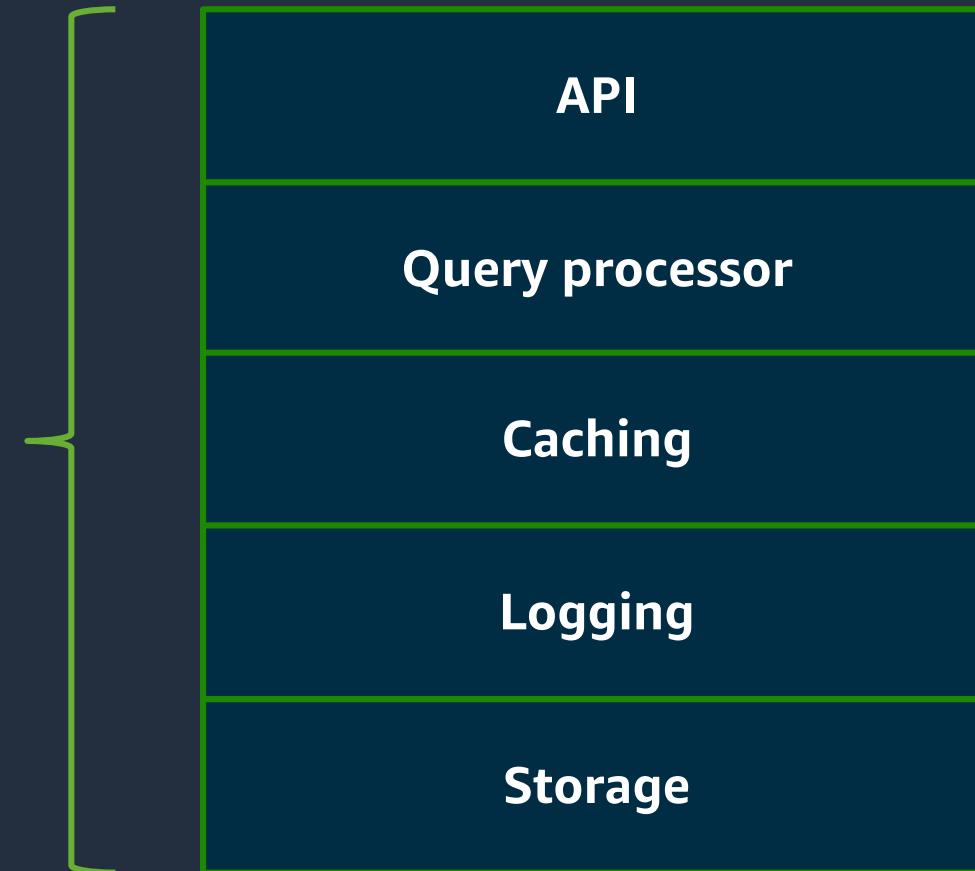


# Architecture

Separation of  
storage / compute

Traditional database  
architecture

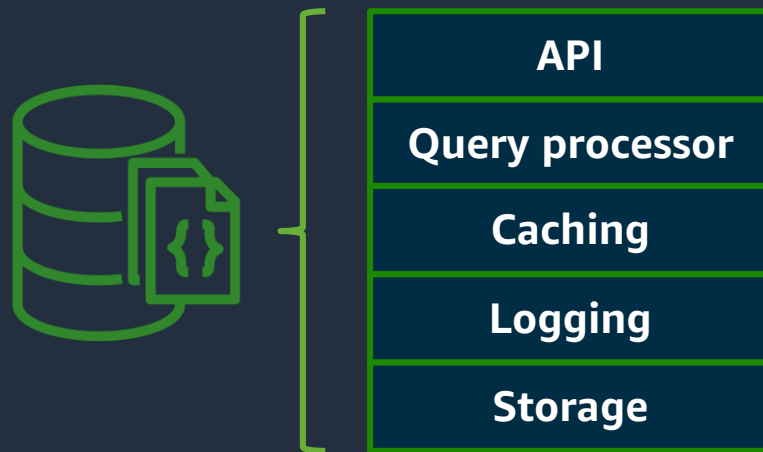
Monolithic,  
shared disk  
architecture



# Architecture

Separation of  
storage / compute

Scaling requires  
copying the  
whole stack

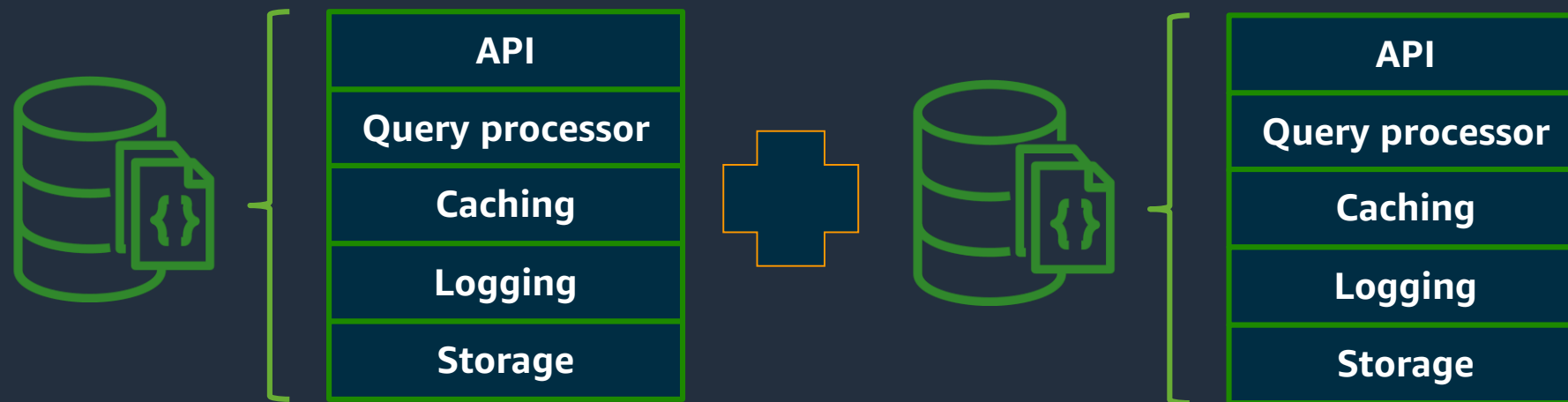




# Architecture

Separation of  
storage / compute

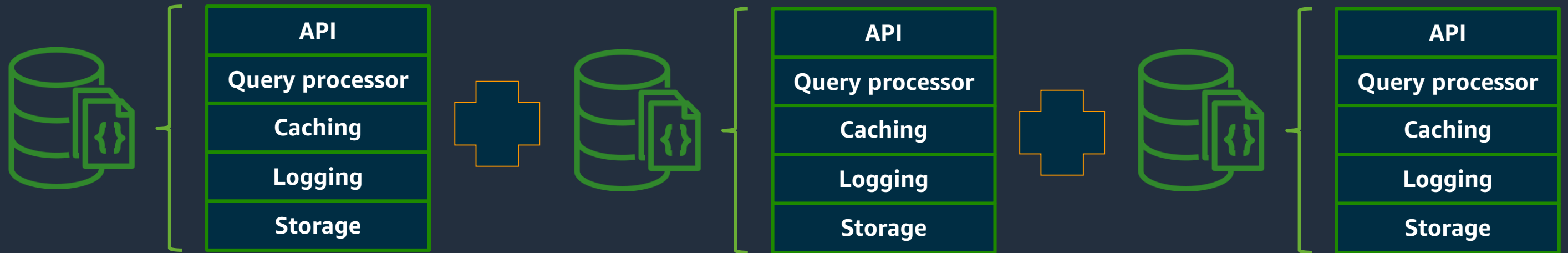
Scaling requires  
copying the  
whole stack



# Architecture

Separation of  
storage / compute

Scaling requires  
copying the  
whole stack

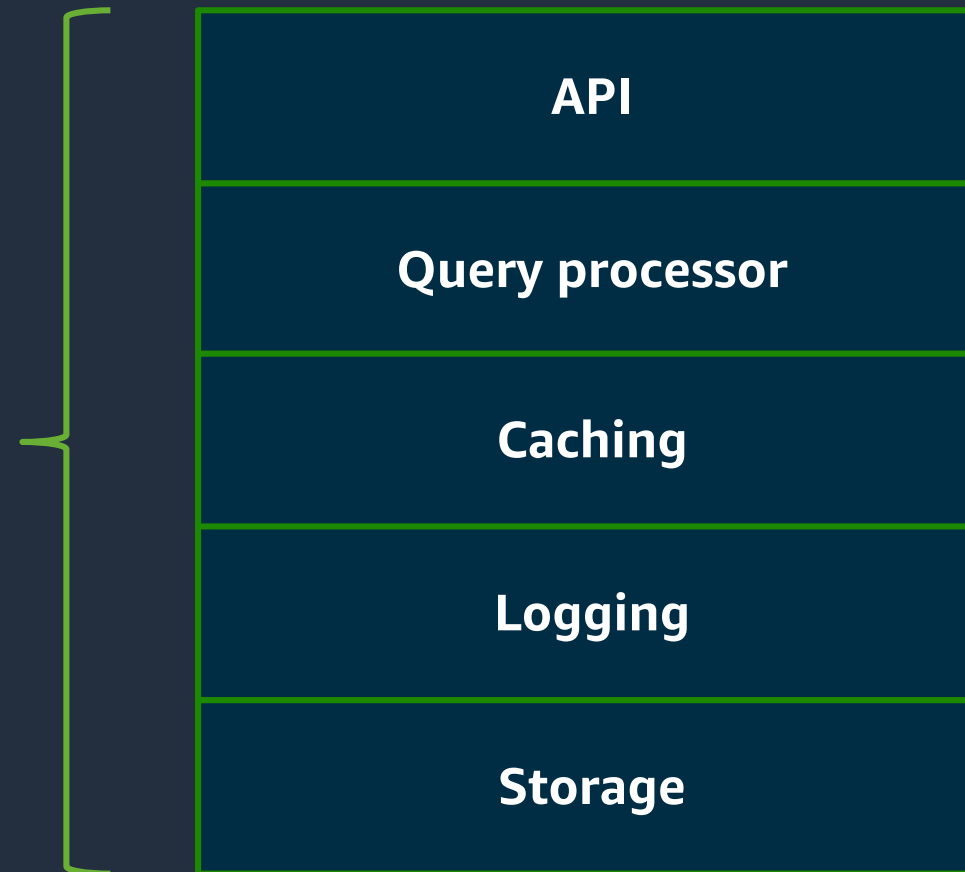


# Architecture

Separation of  
storage / compute

Traditional database  
architecture

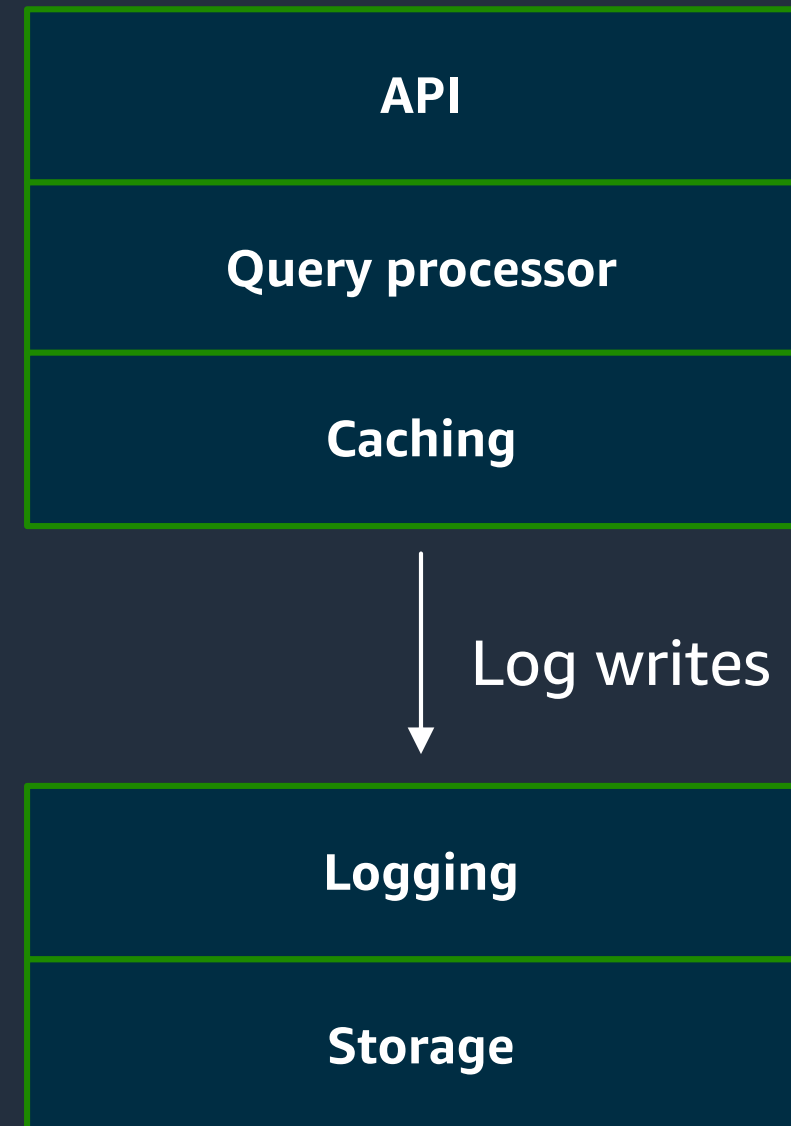
Monolithic,  
shared disk  
architecture



# Architecture

Separation of  
storage / compute

Separation of storage  
and compute



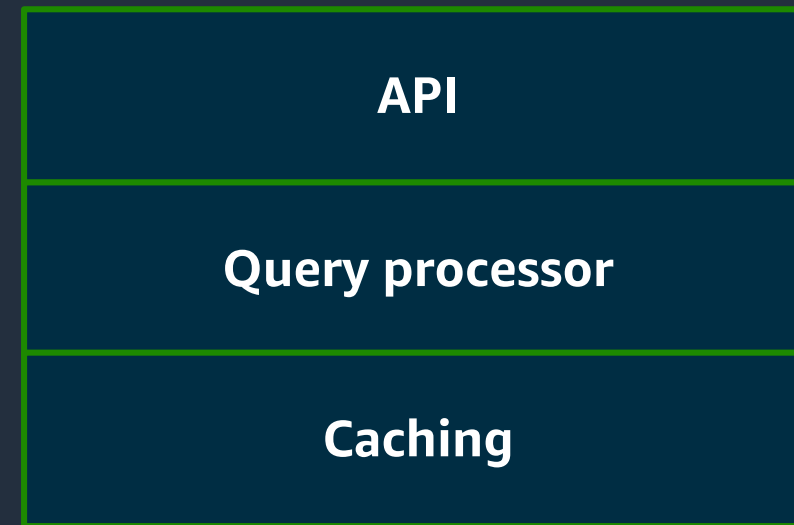
# Architecture

Separation of storage / compute

Separation of storage and compute



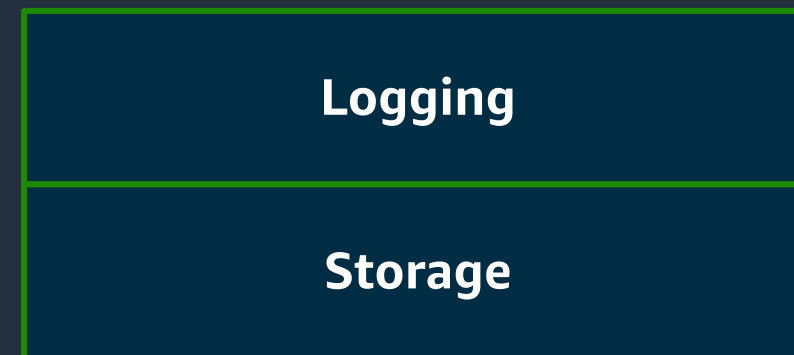
Compute layer



Log writes



Storage layer



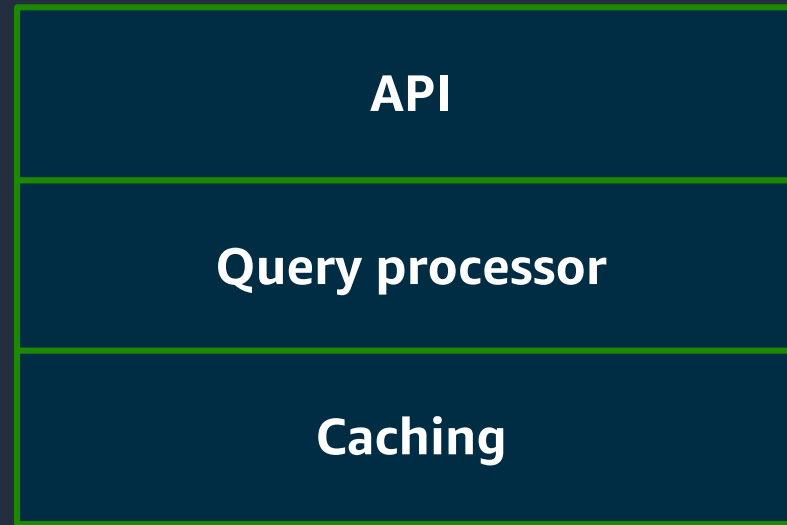
# Architecture

Separation of storage / compute

Separation of storage and compute



Compute layer

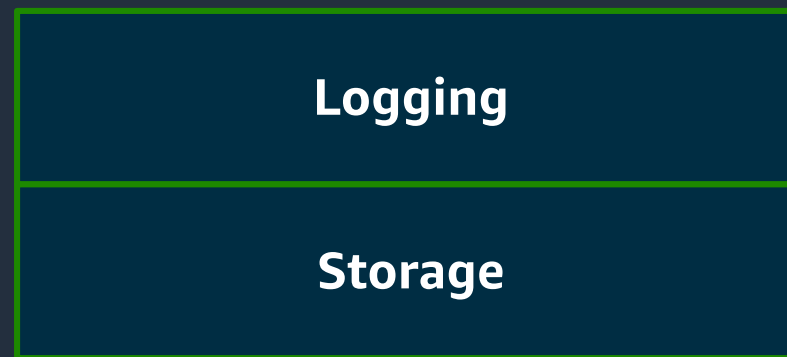


Scale compute →

Decouple compute and storage

Log writes

Storage layer



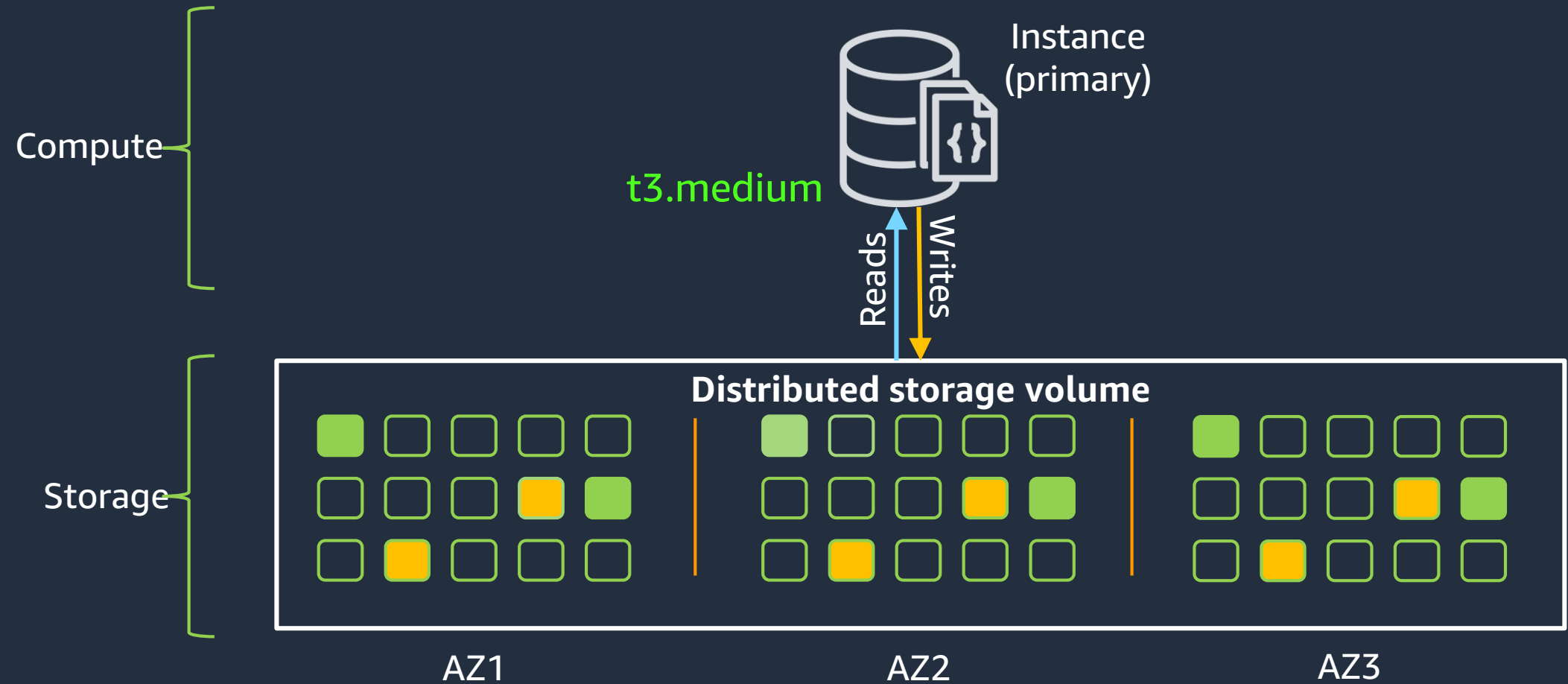
Scale storage →



# Architecture

## Separation of storage / compute

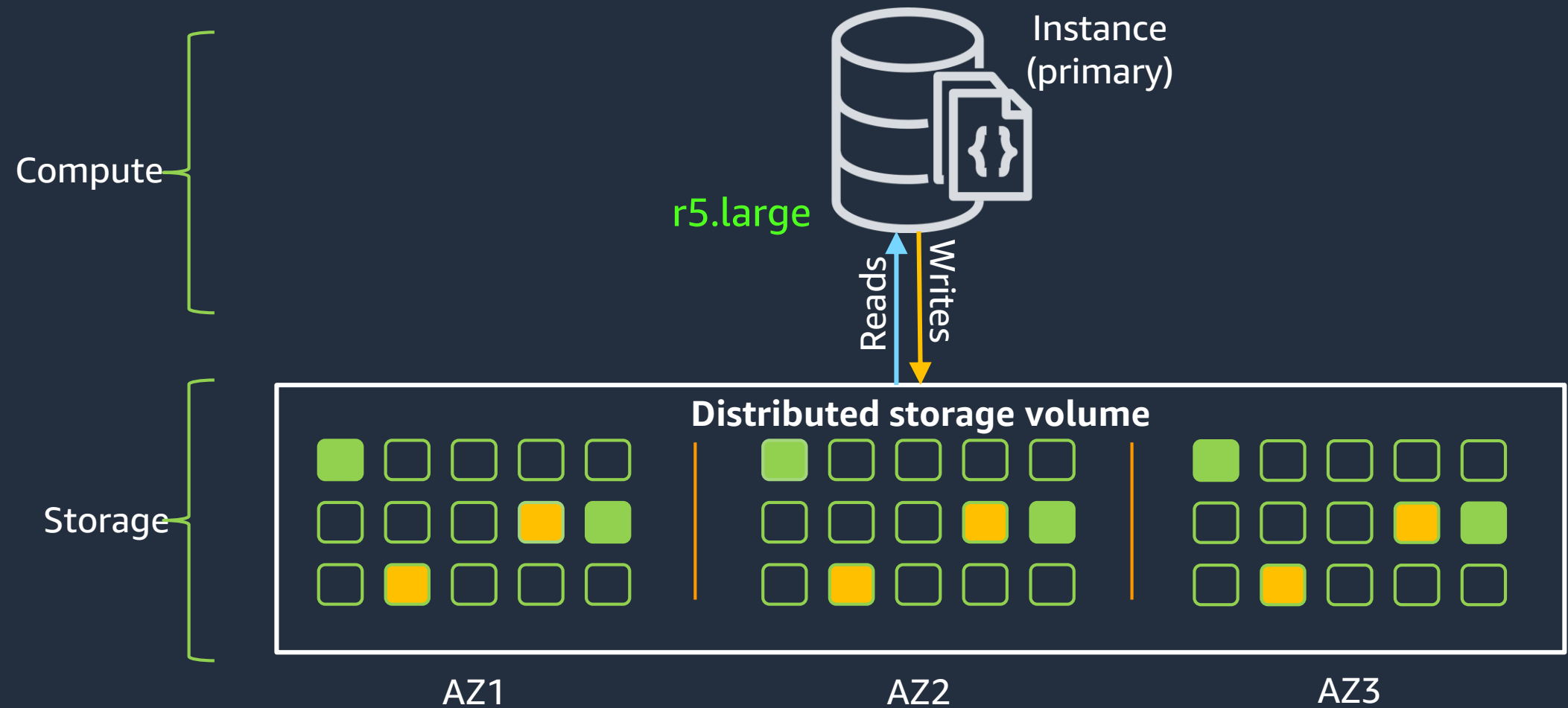
t3.medium:  
Dedicated single-  
instance cluster  
for dev / test  
(\$0.078 / hour)



# Architecture

Separation of storage / compute

Scale-up your existing instance in minutes, regardless of storage size

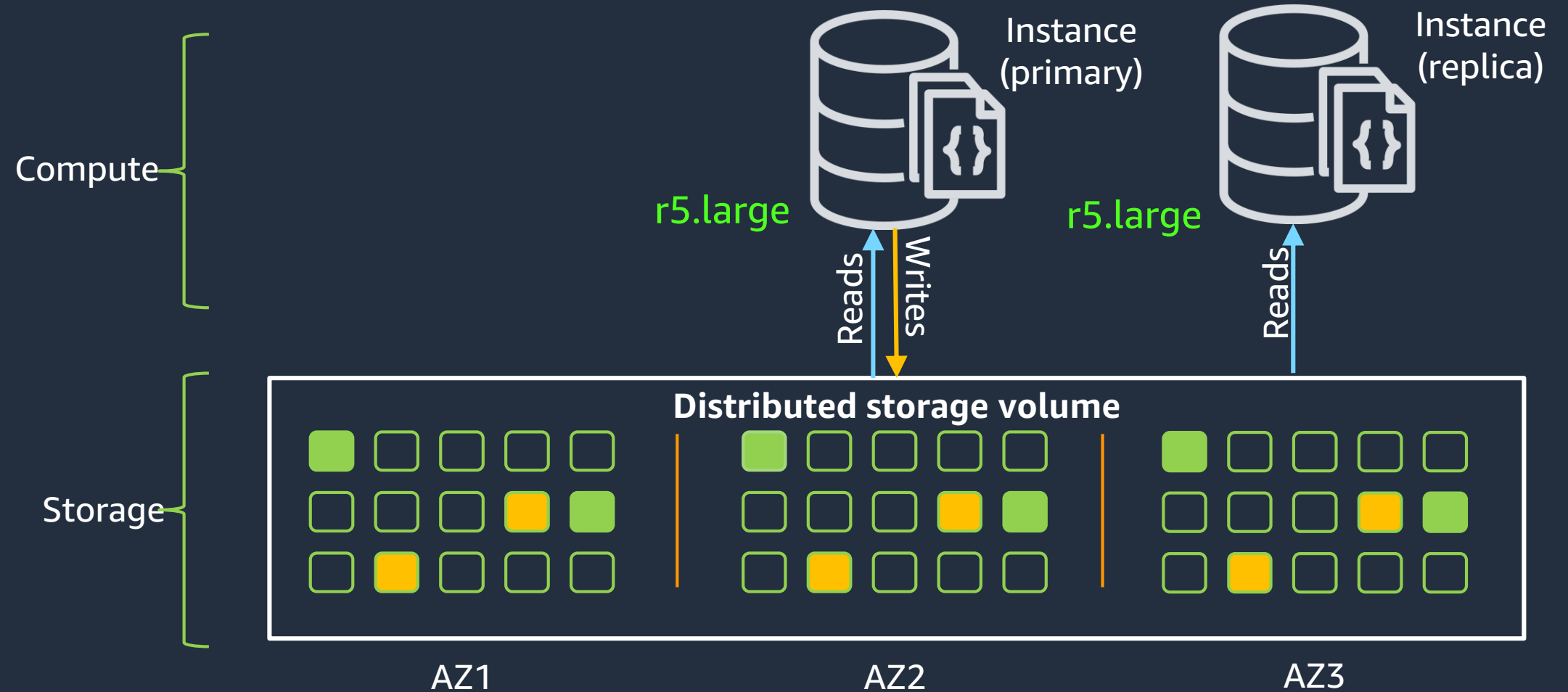




# Architecture

## Separation of storage / compute

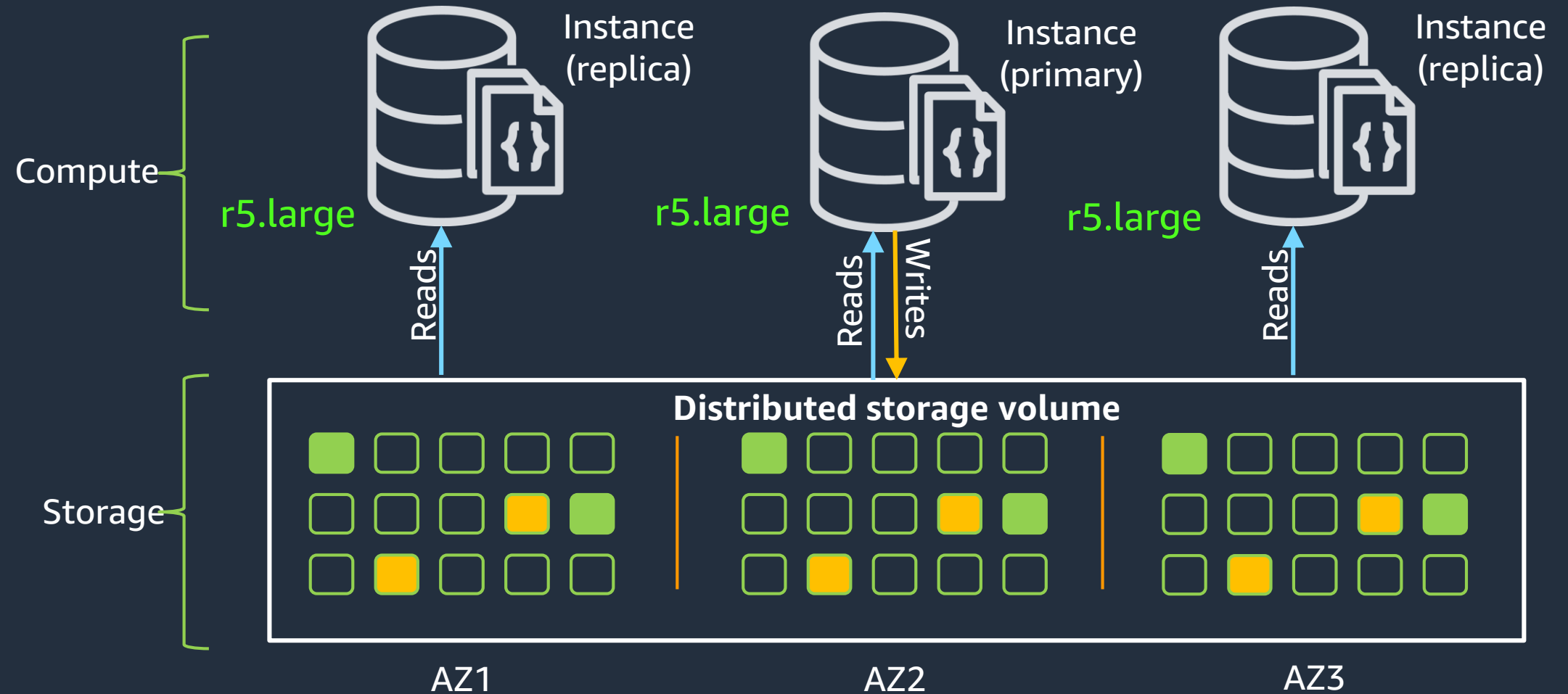
Add new instances in minutes, regardless of storage size



# Architecture

## Separation of storage / compute

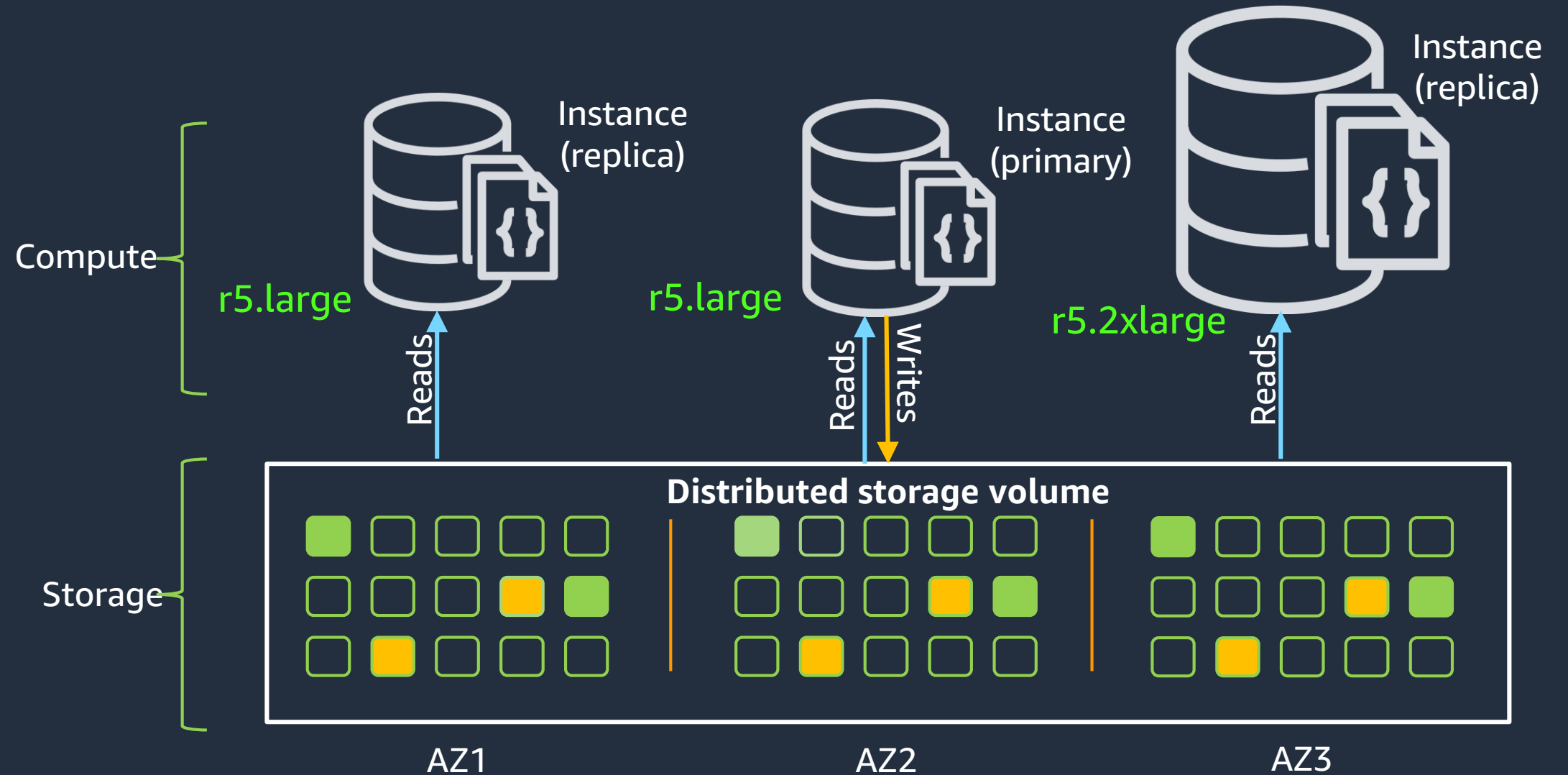
Add new instances in minutes, regardless of storage size



# Architecture

## Separation of storage / compute

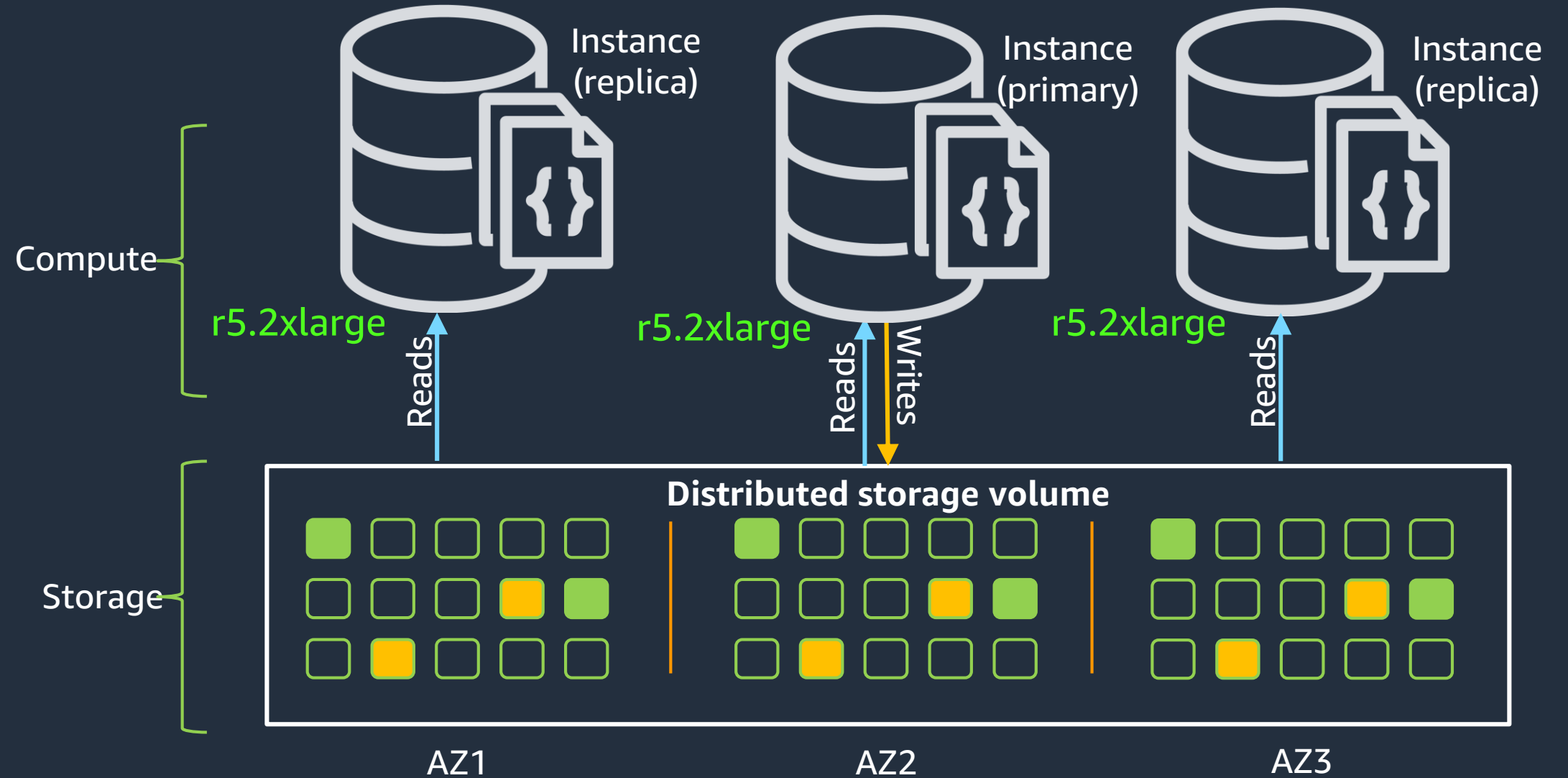
Scale-up your existing instance in minutes, regardless of storage size



# Architecture

## Separation of storage / compute

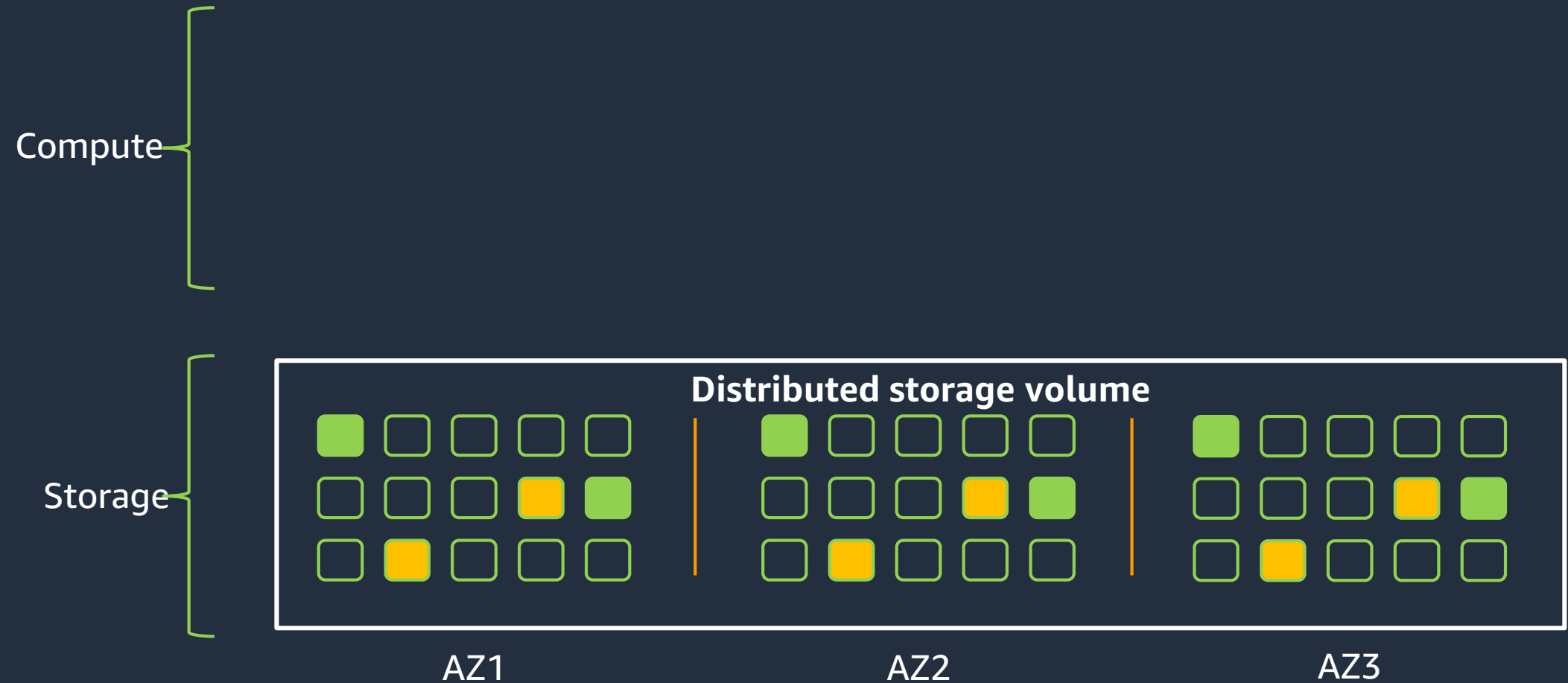
Scale-up your existing instance in minutes, regardless of storage size



# Architecture

## Separation of storage / compute

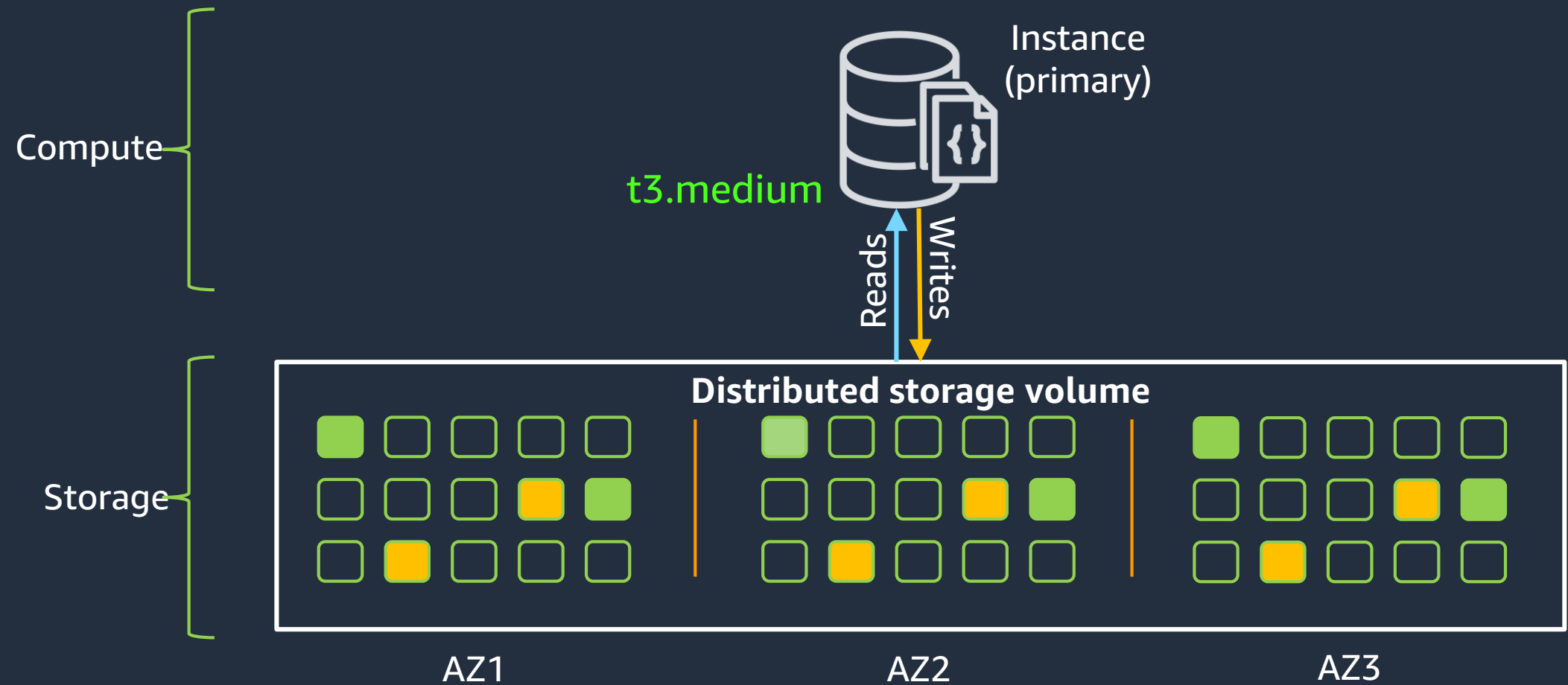
Compute and storage are separated to the degree that you can delete all compute and your data is still highly durable



# Architecture

Separation of storage / compute

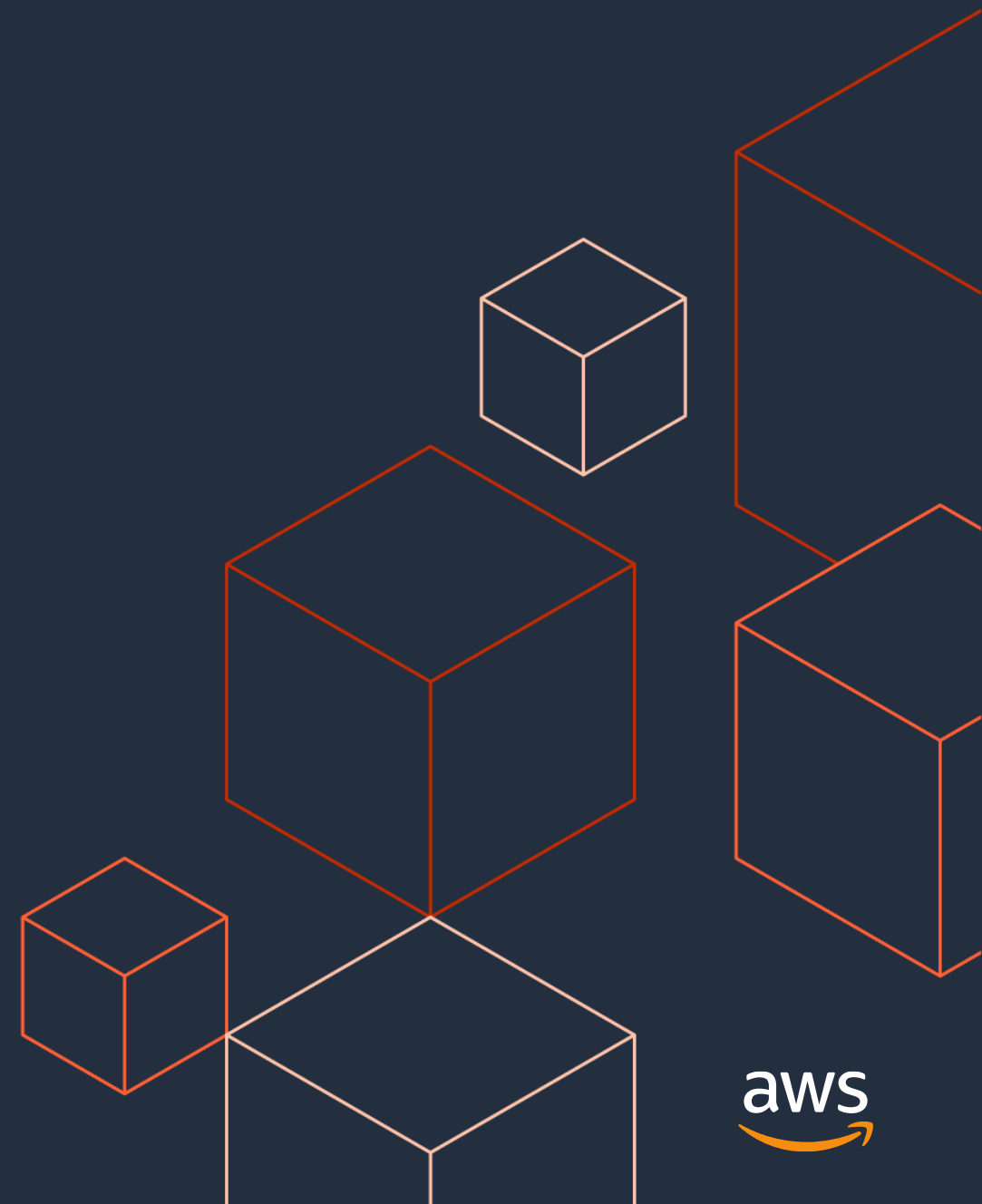
Single node cluster is still highly durable



# Demo: Adding a new instance to a 12 TB cluster + backup

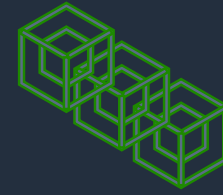


# Replication





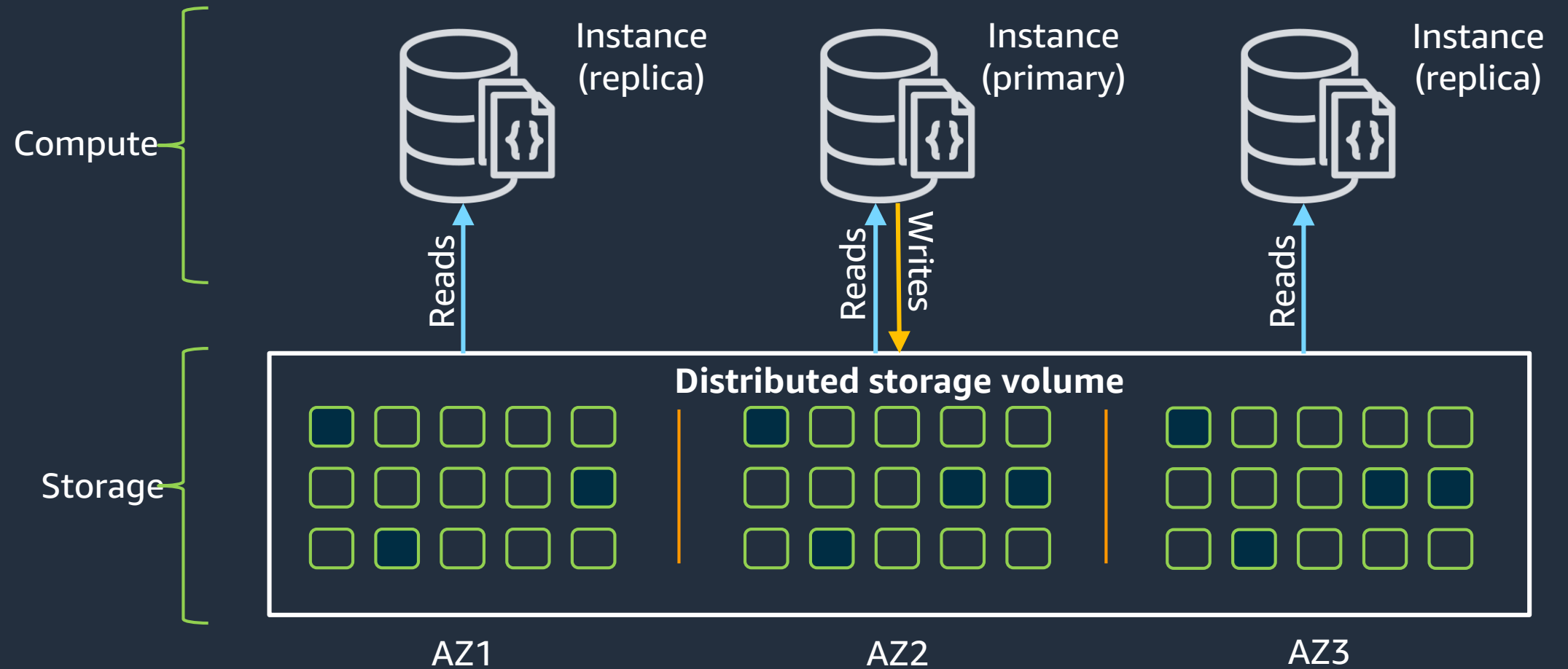
# Architecture



Application

Replication

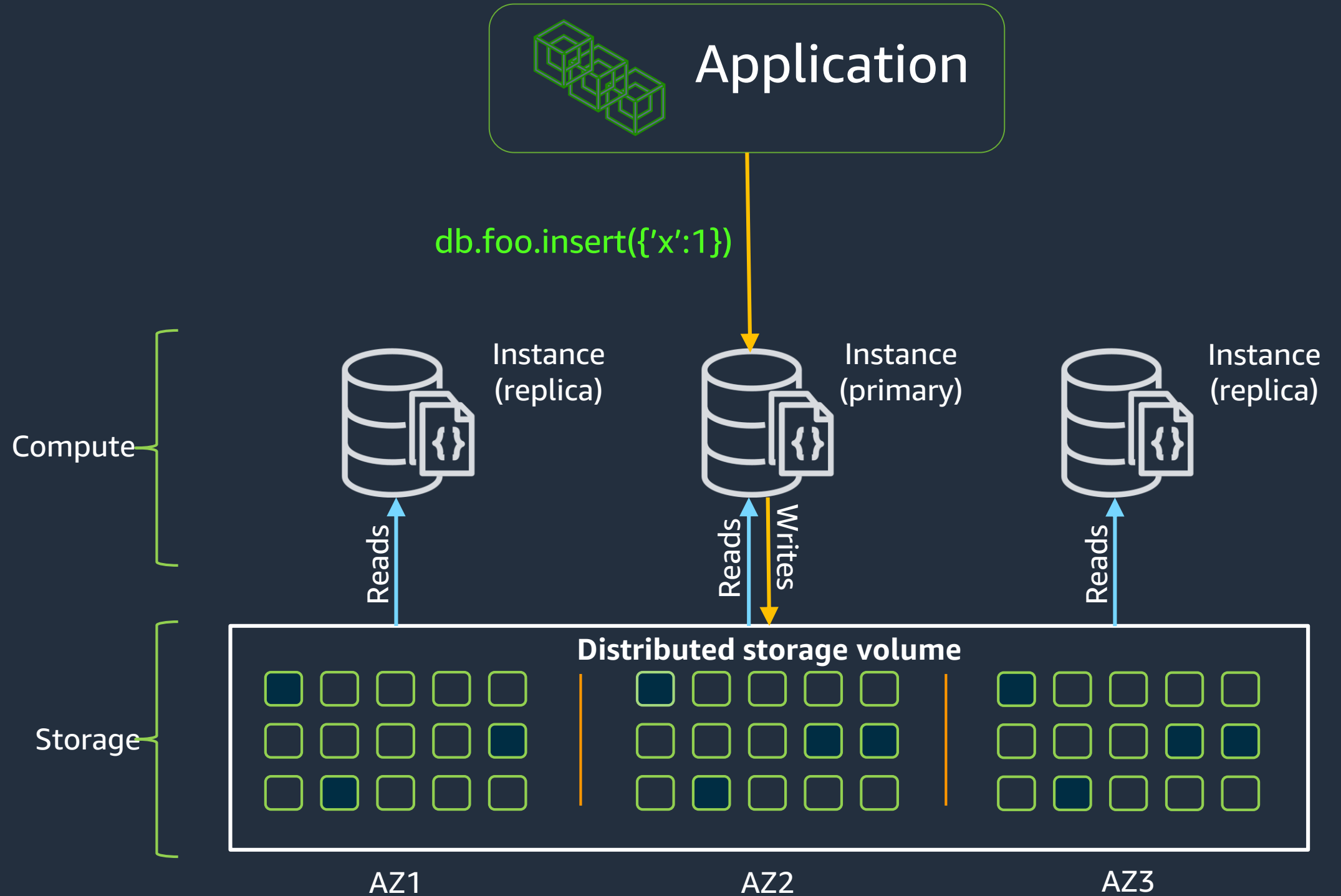
Default write concern is w:4 and j:true



# Architecture

## Replication

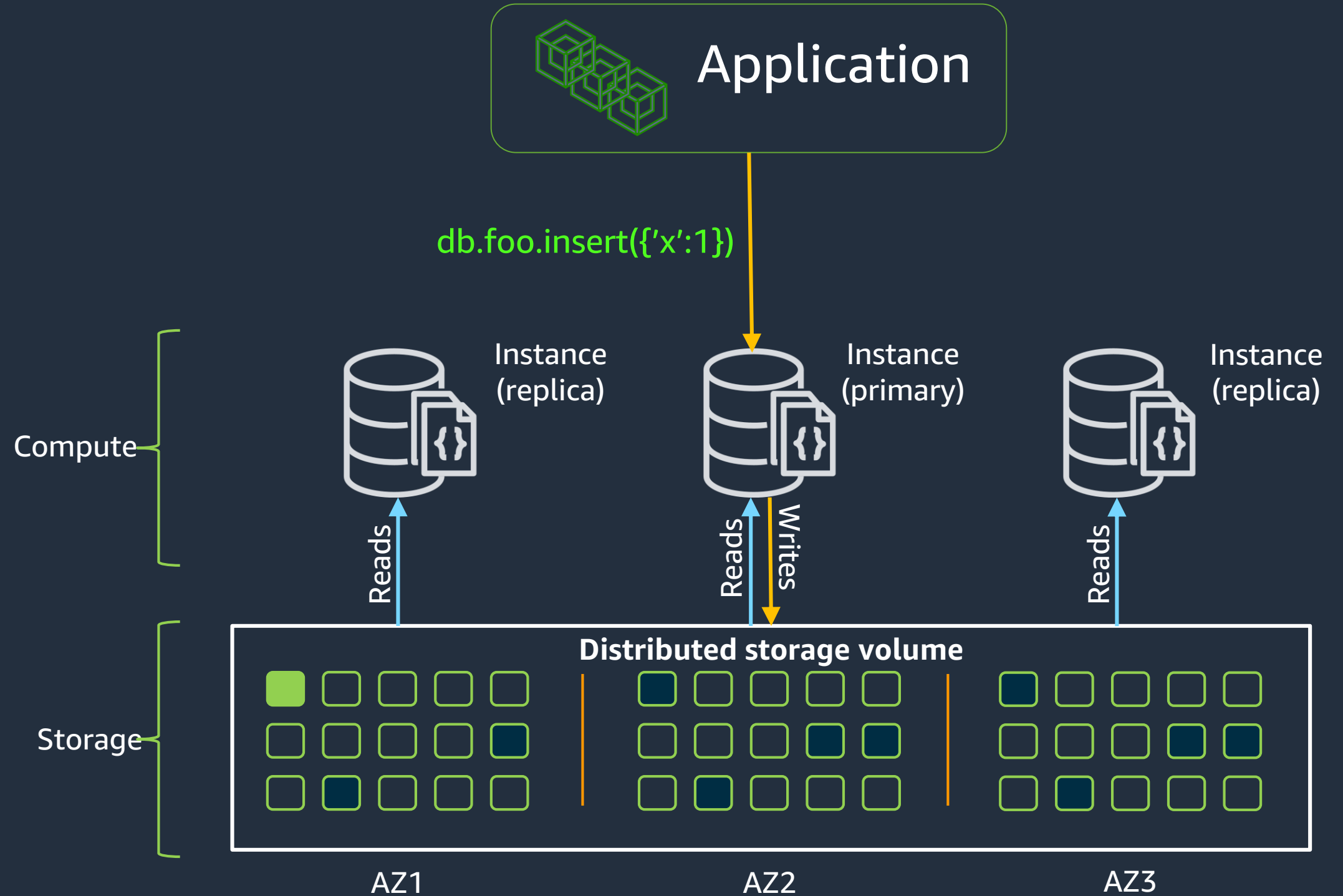
Writes are acknowledged after a quorum if ( $V_w=4$ )



# Architecture

## Replication

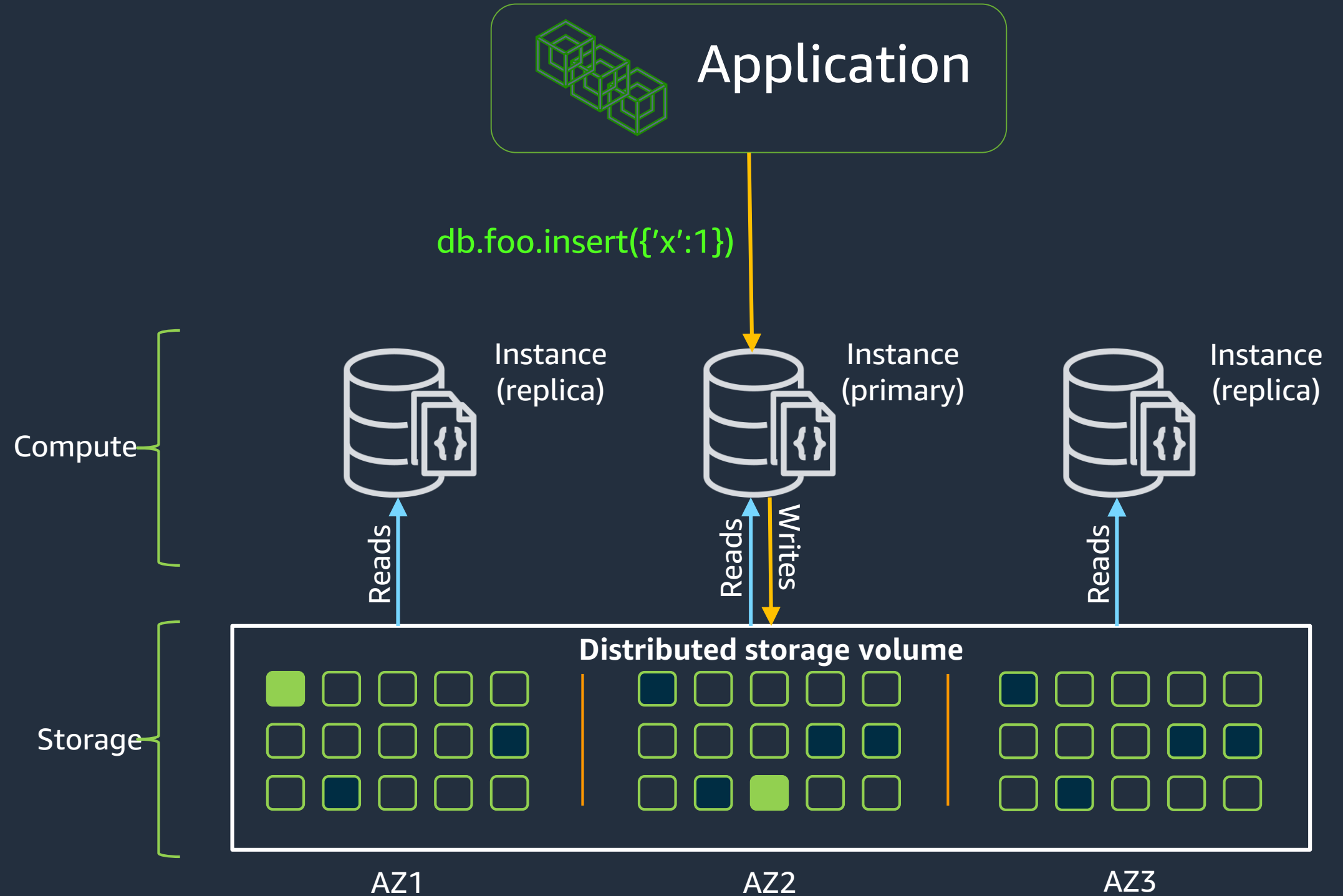
Writes are acknowledged after a quorum if ( $V_w=4$ )



# Architecture

## Replication

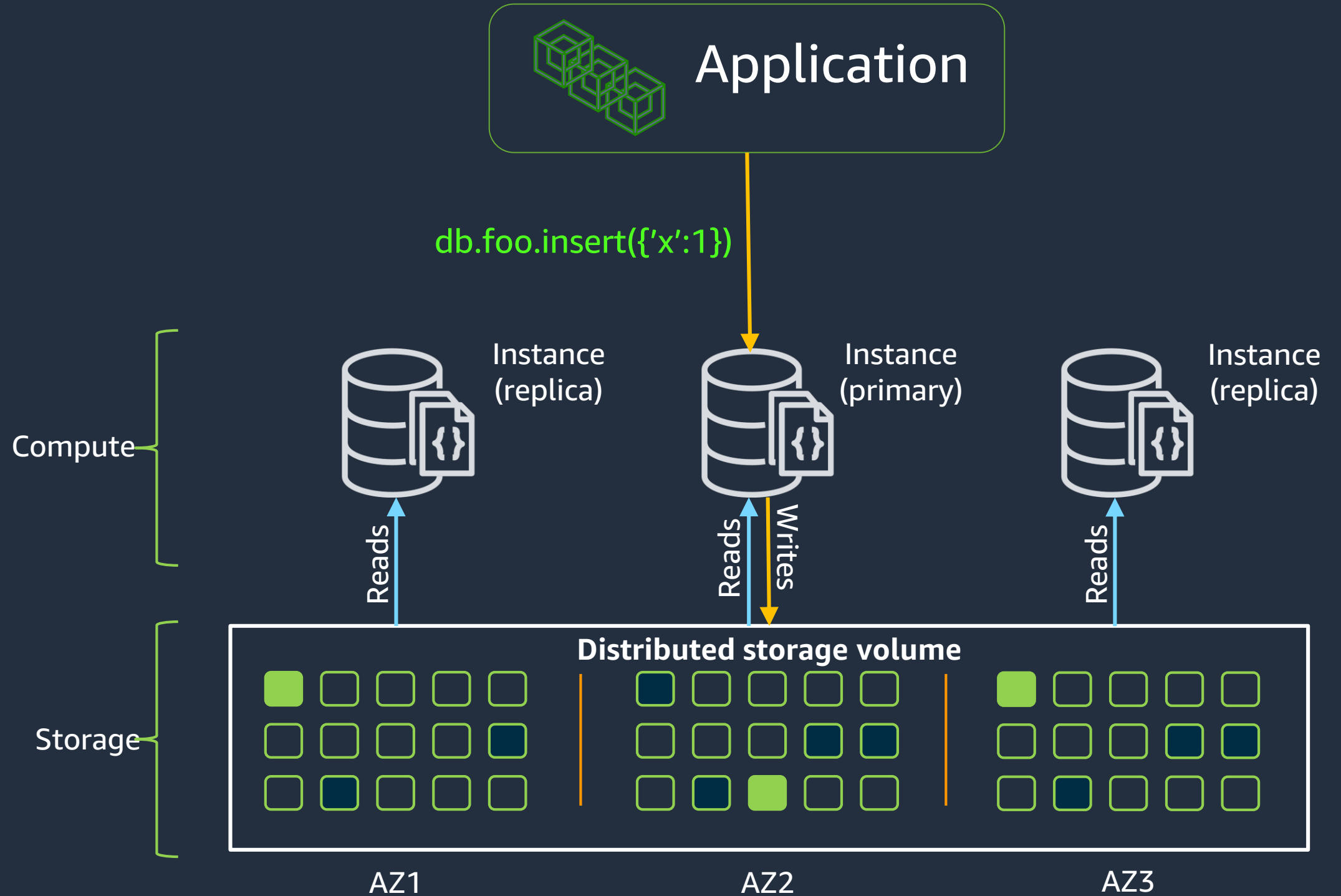
Writes are acknowledged after a quorum if ( $V_w=4$ )



# Architecture

## Replication

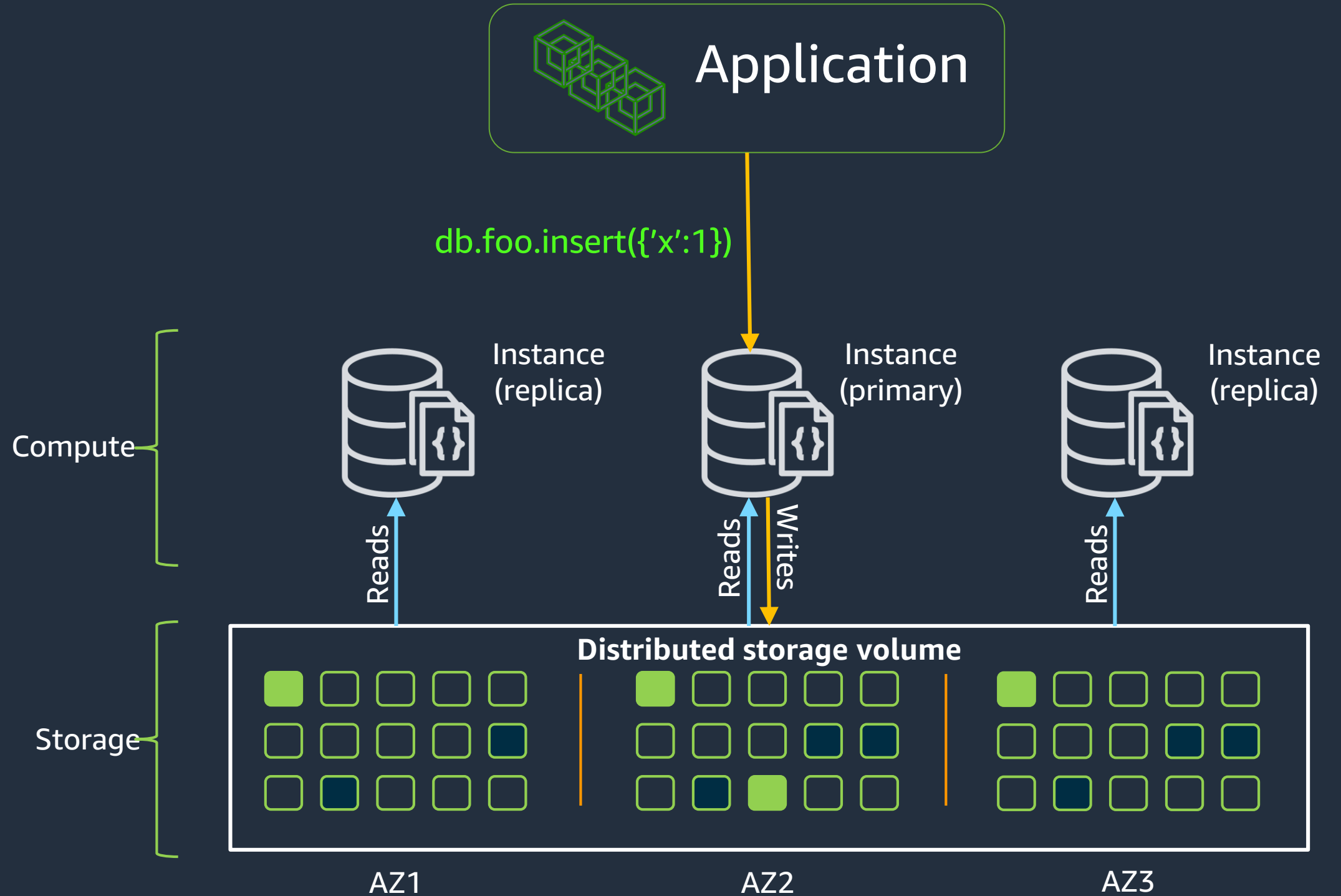
Writes are acknowledged after a quorum if ( $V_w=4$ )



# Architecture

## Replication

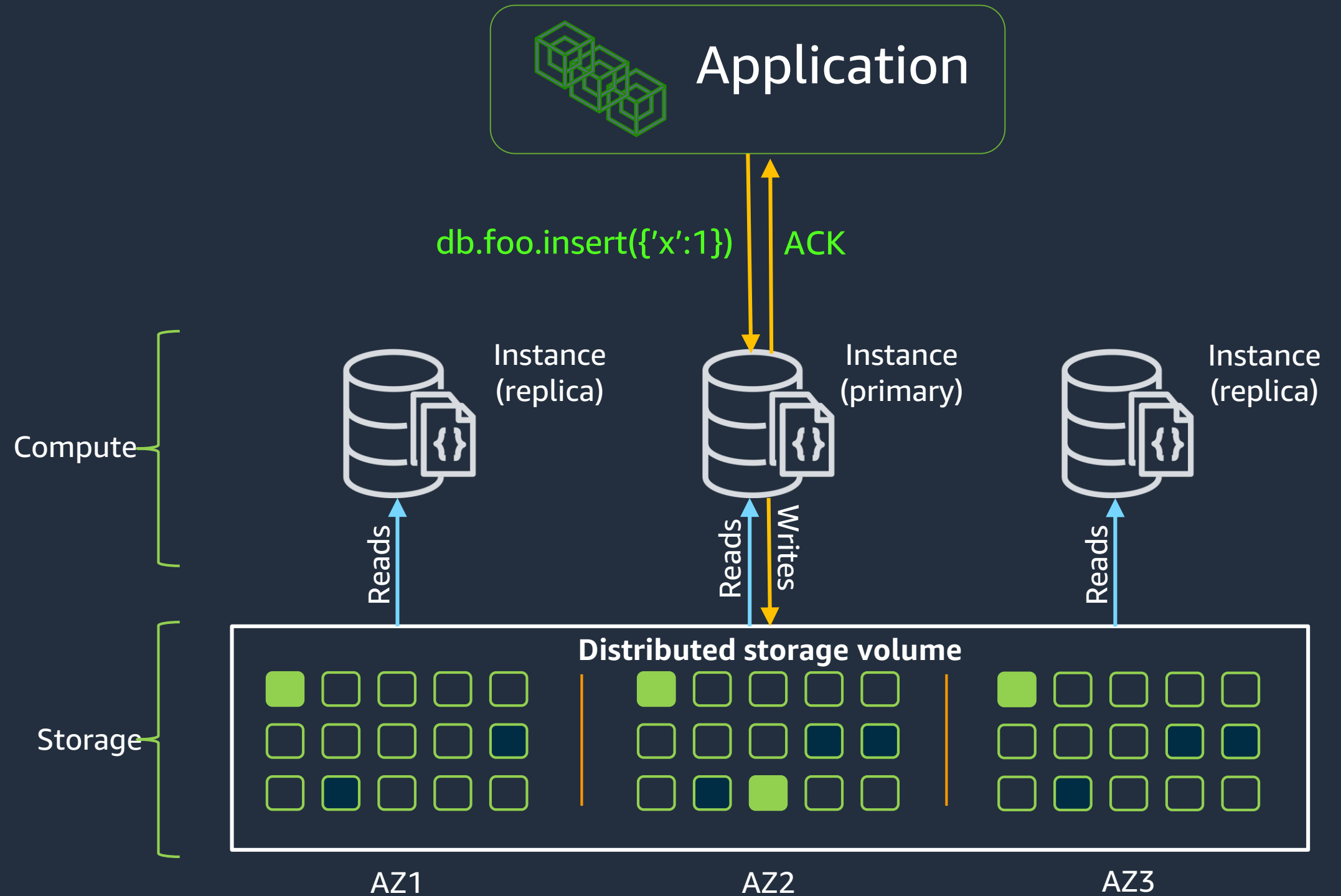
Writes are acknowledged after a quorum if ( $V_w=4$ )



# Architecture

## Replication

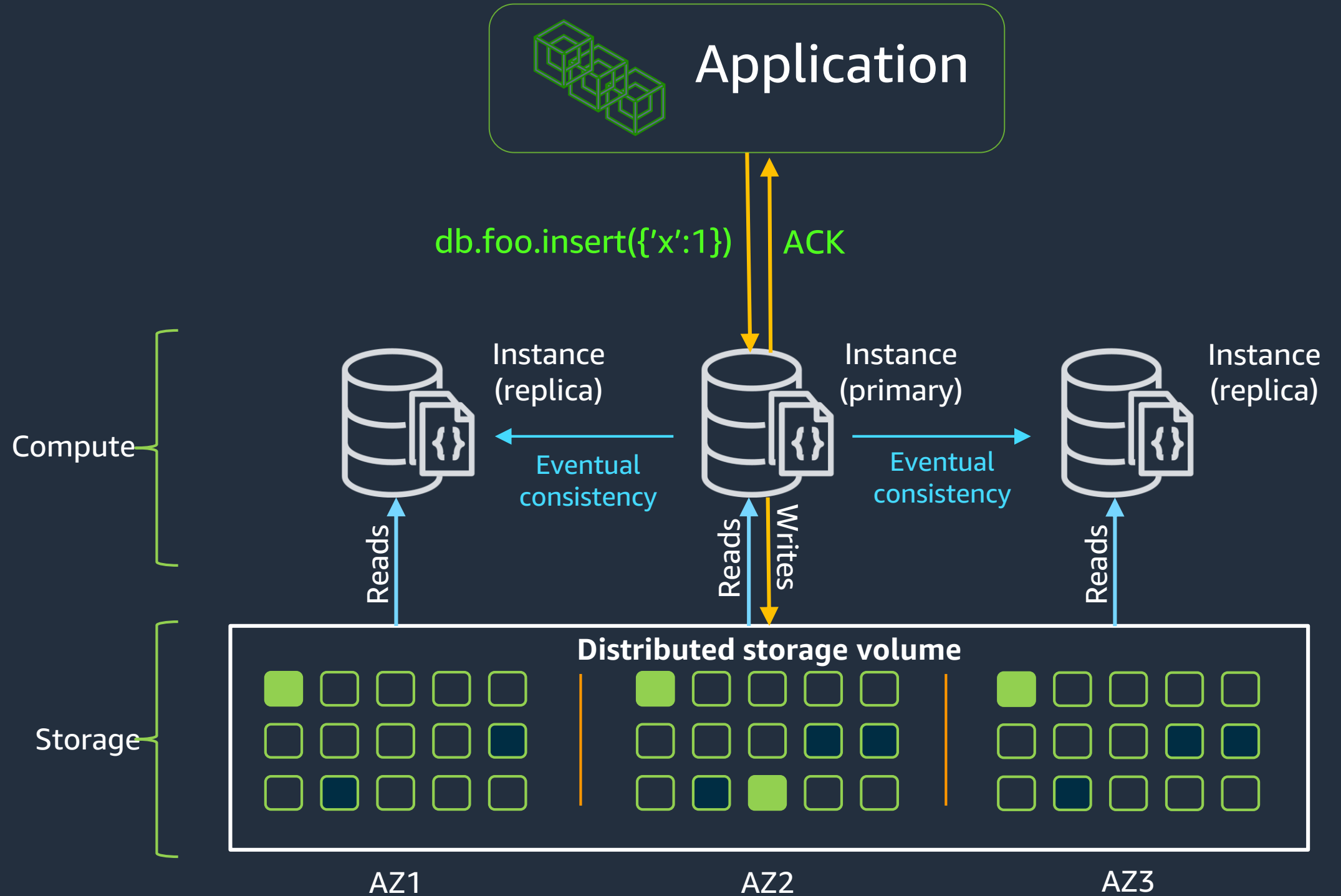
Writes are acknowledged after a quorum if ( $V_w=4$ )



# Architecture

## Replication

Data is further replicated to achieve 6 copies across three AZs (V=6)

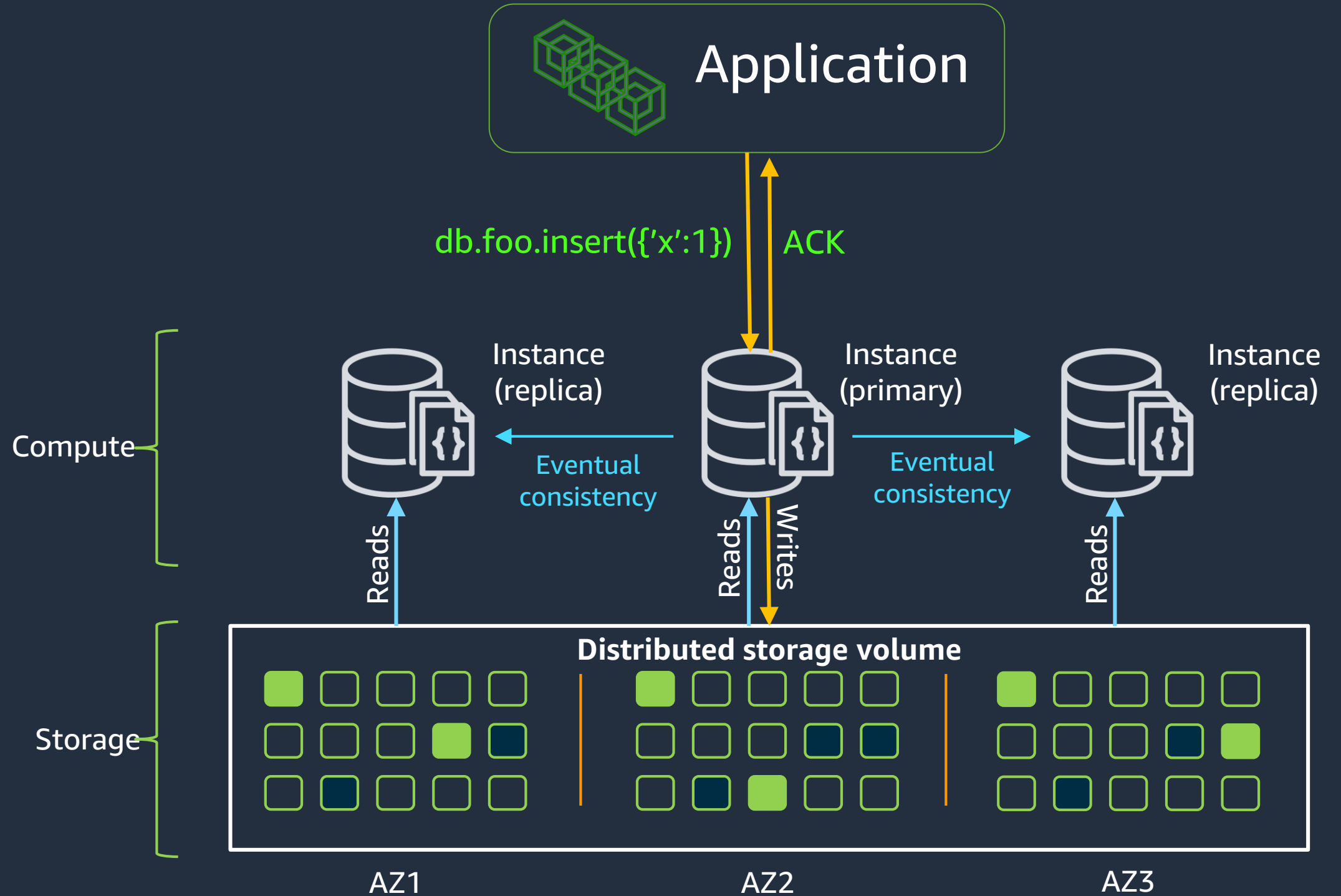




# Architecture

## Replication

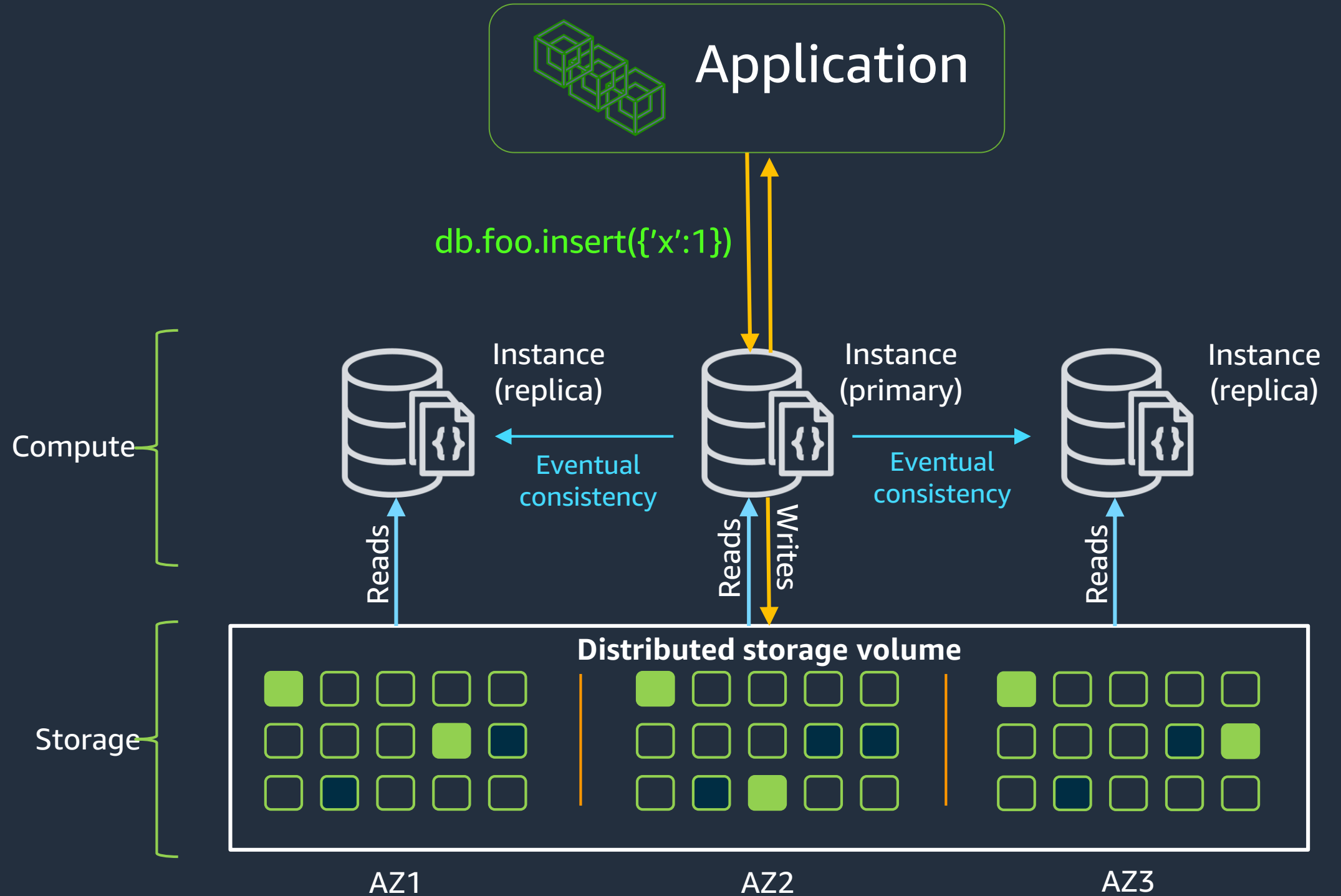
Replicas do not participate in data replication, freeing them up for reads



# Architecture

## Replication

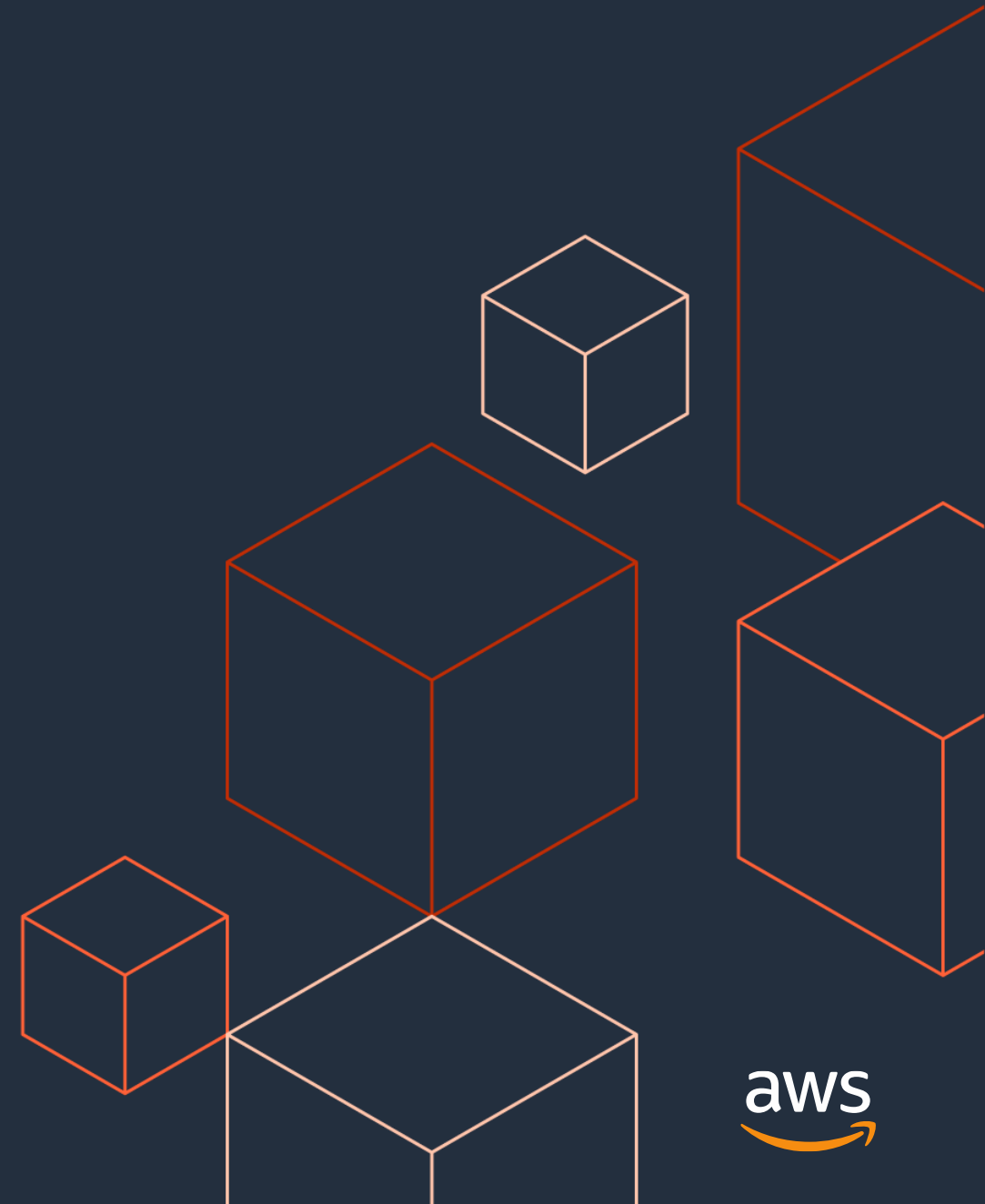
As a best practice, we recommend connecting as a replica set (with read preference, secondaryPreferred)



# Demo: Connecting as a replica set to scale reads



# Durability



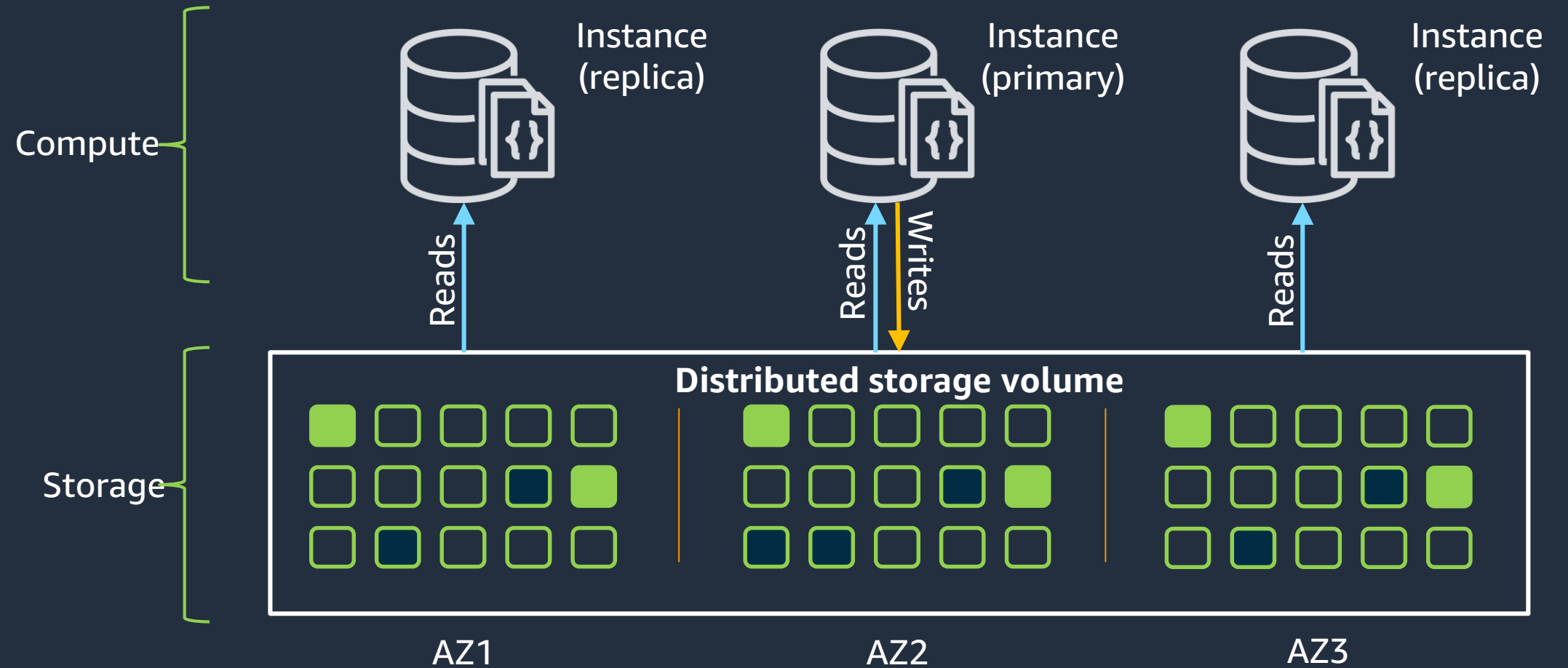
# Architecture

Highly durable by default

AZ + 1 resilient

## Durability

Storage is replicated six-ways across 3 AZs (10 GB partitions)



# Architecture

Highly durable by default

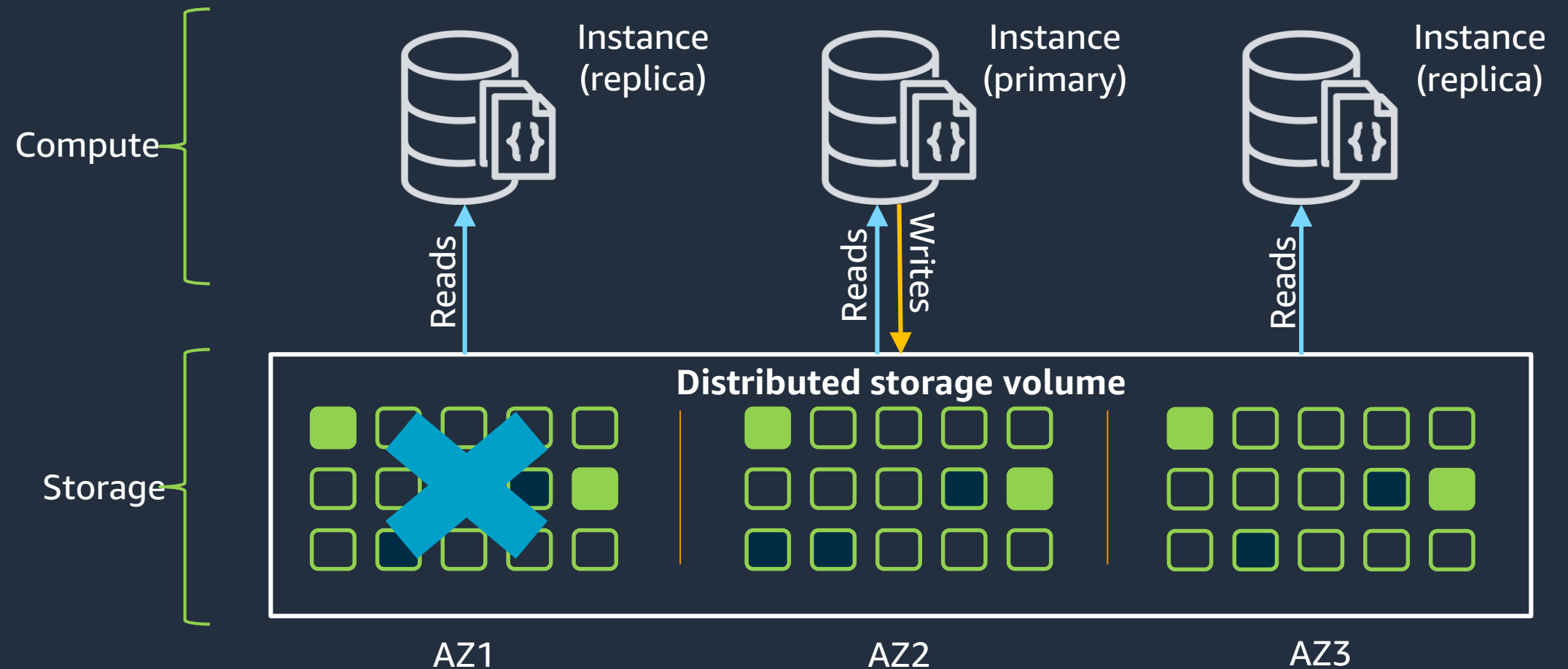
AZ + 1 resilient

Durability

**Loss of an AZ:**  
Write quorum is still achieved with the loss of an AZ

$$V_w=4$$

$$V_r=3$$



# Architecture

Highly durable by default

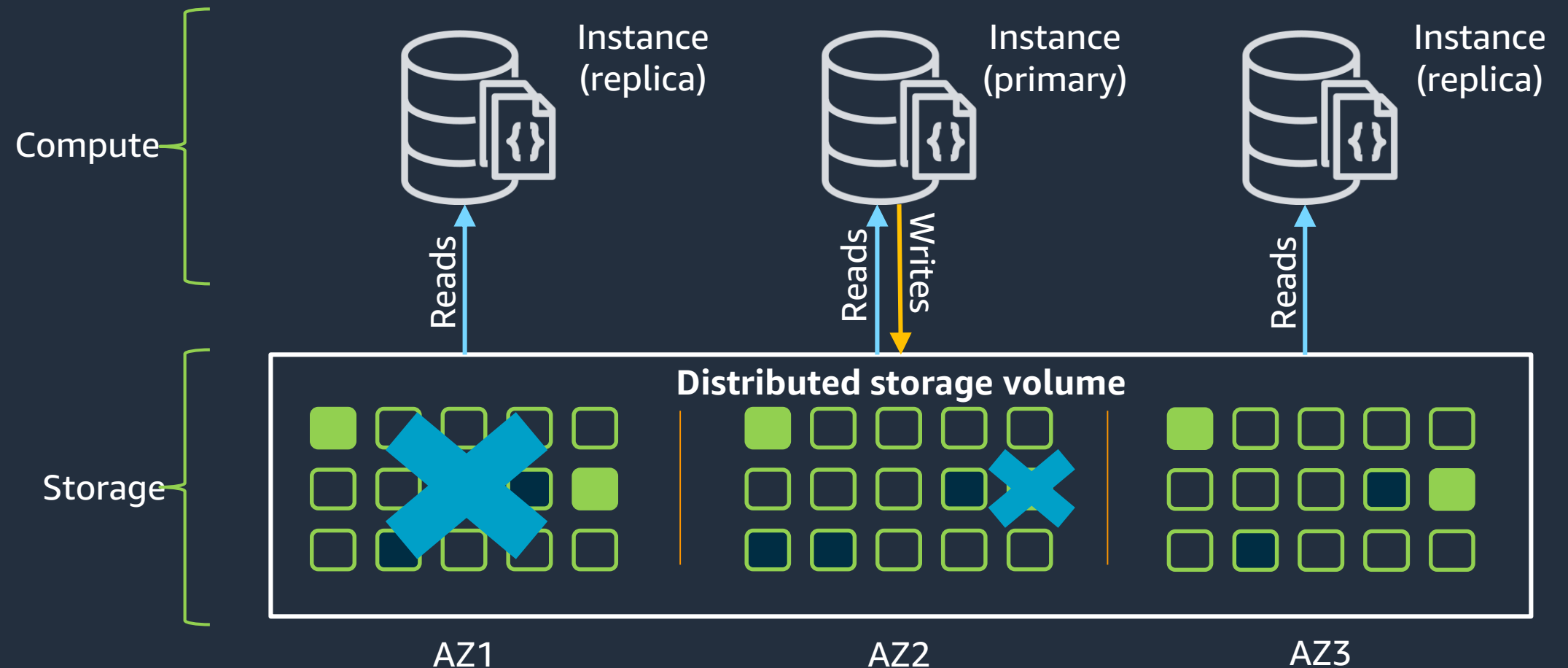
AZ + 1 resilient

Durability

Loss of an AZ +1:  
Read quorum and  
durability are still  
achieved

$$V_w=4$$

$$V_r=3$$



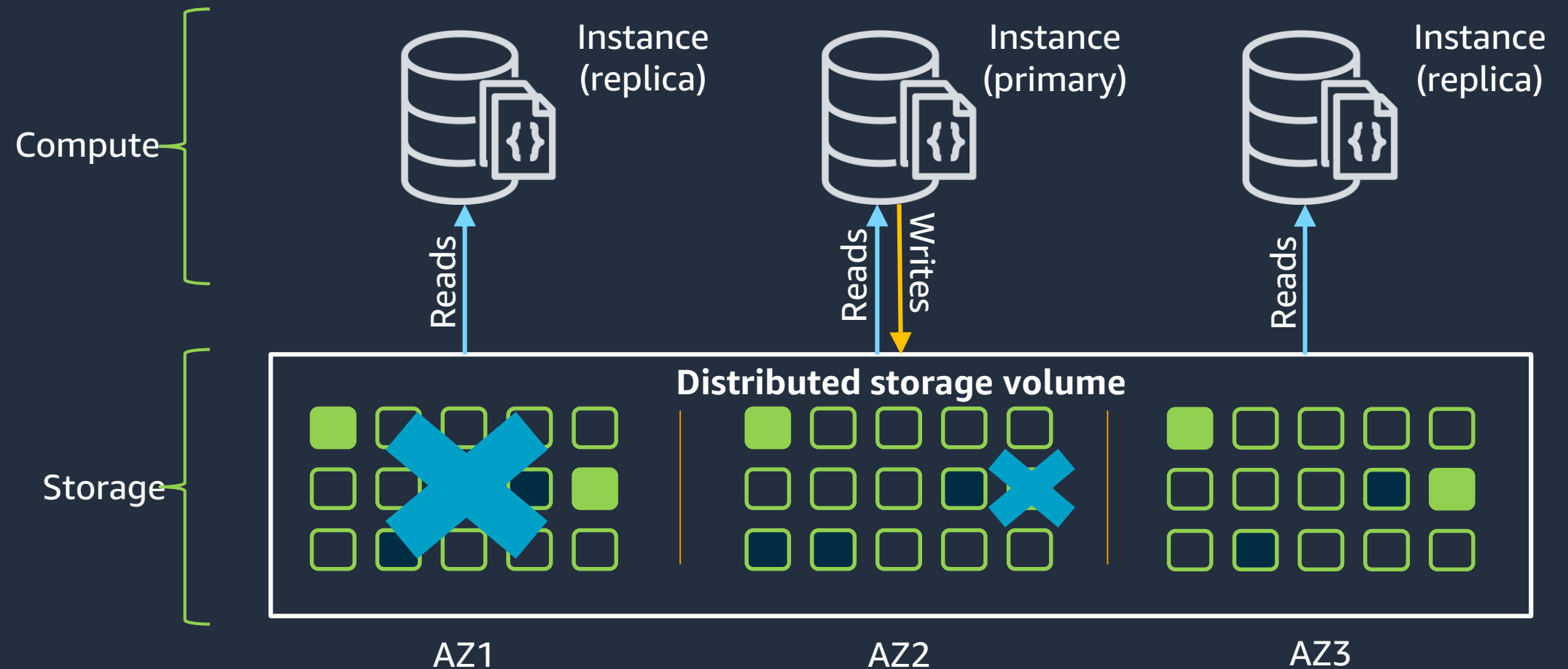
# Architecture

Highly durable by default

## Durability

Mean time-to-recovery is a function of replicating 10 GB

AZ + 1 resilient





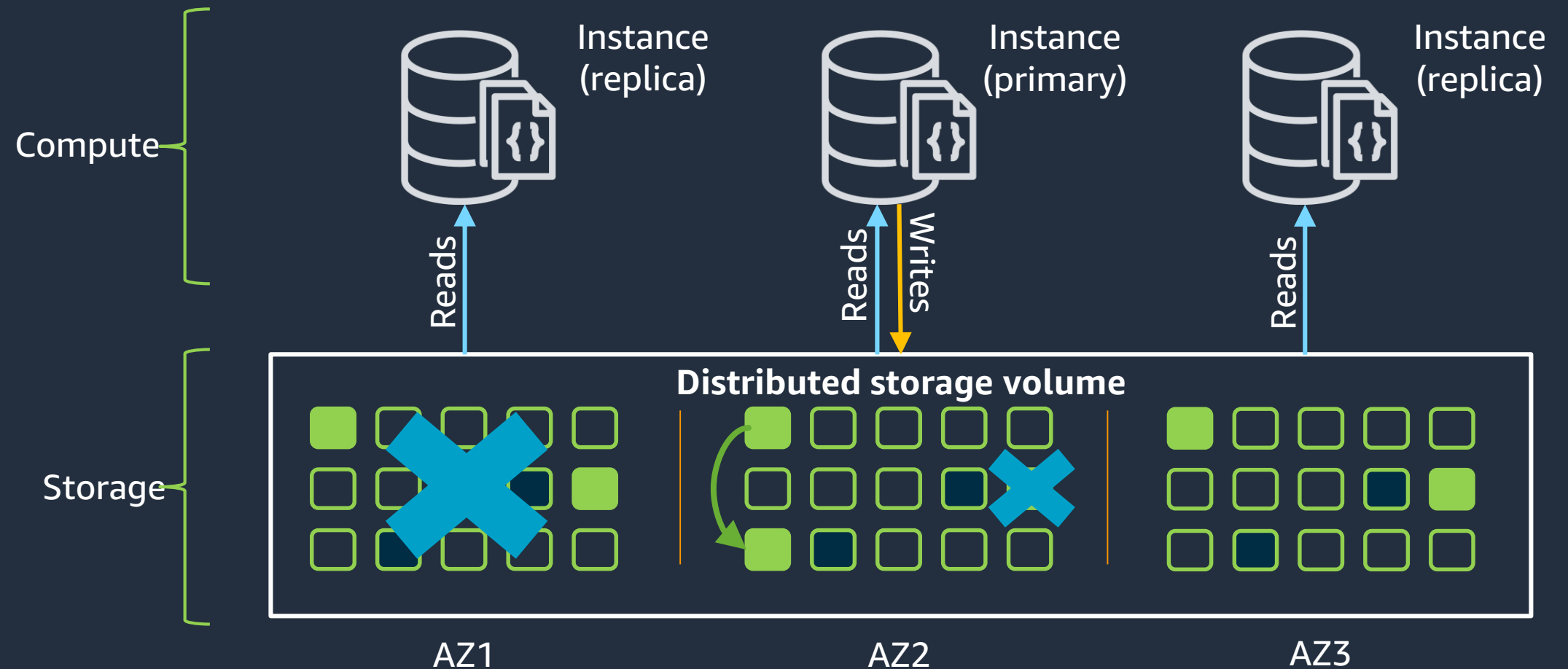
# Architecture

Highly durable by default

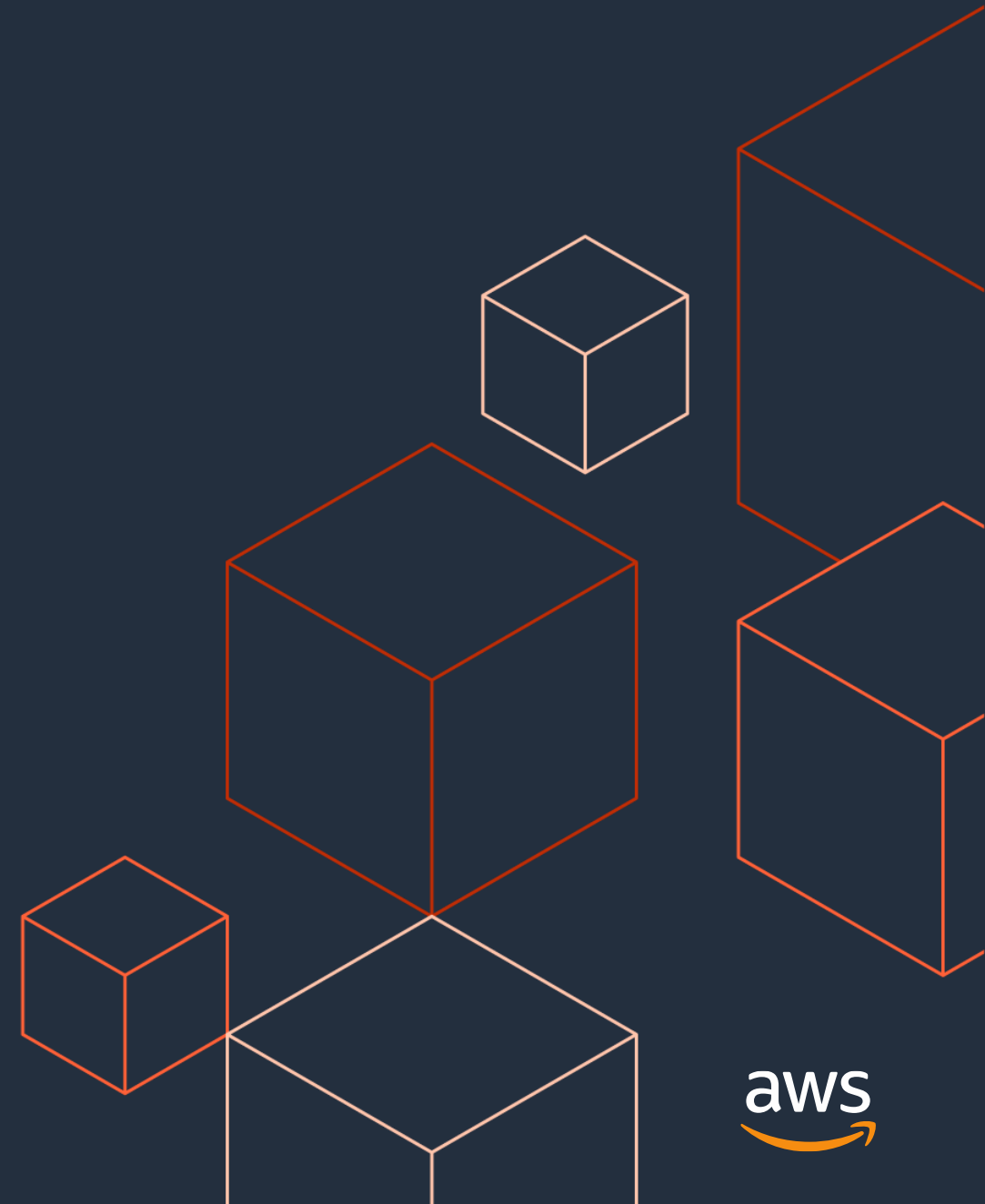
## Durability

Mean time-to-recovery is a function of replicating 10 GB

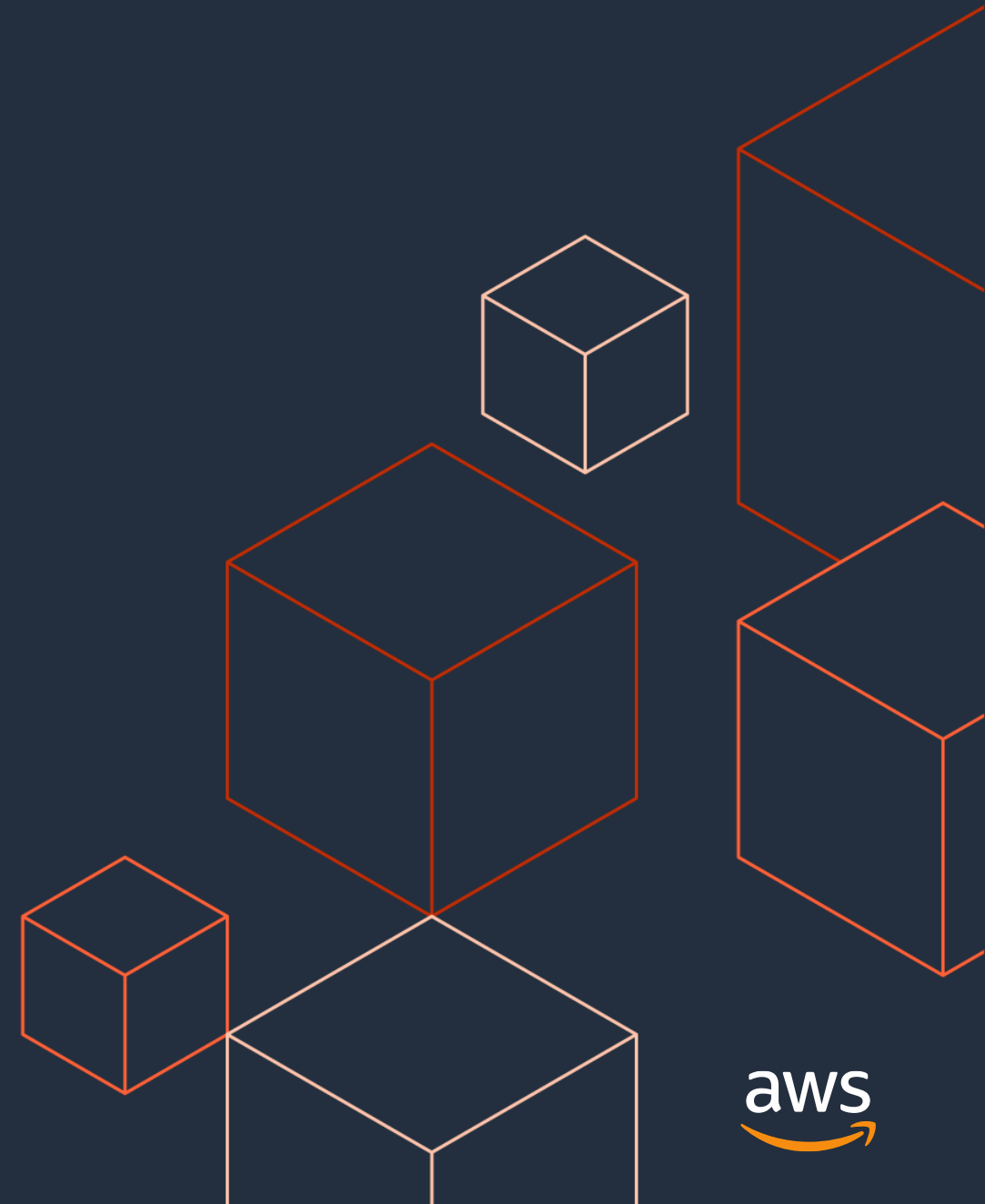
AZ + 1 resilient



# Demo: Write concern



# Backup



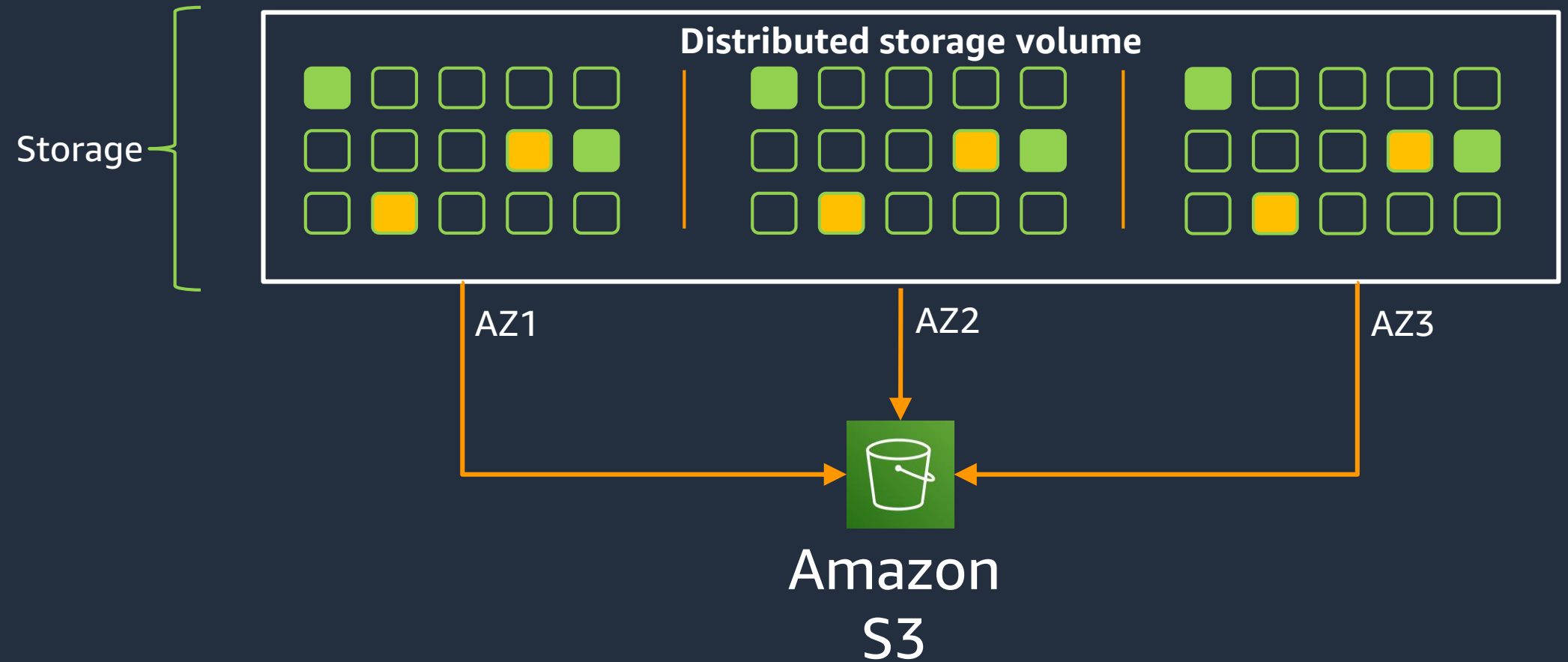
# Architecture

No impact backups

Backup

Backup does not use your compute instances

Point-in-time restore  
(35 days, second granularity)



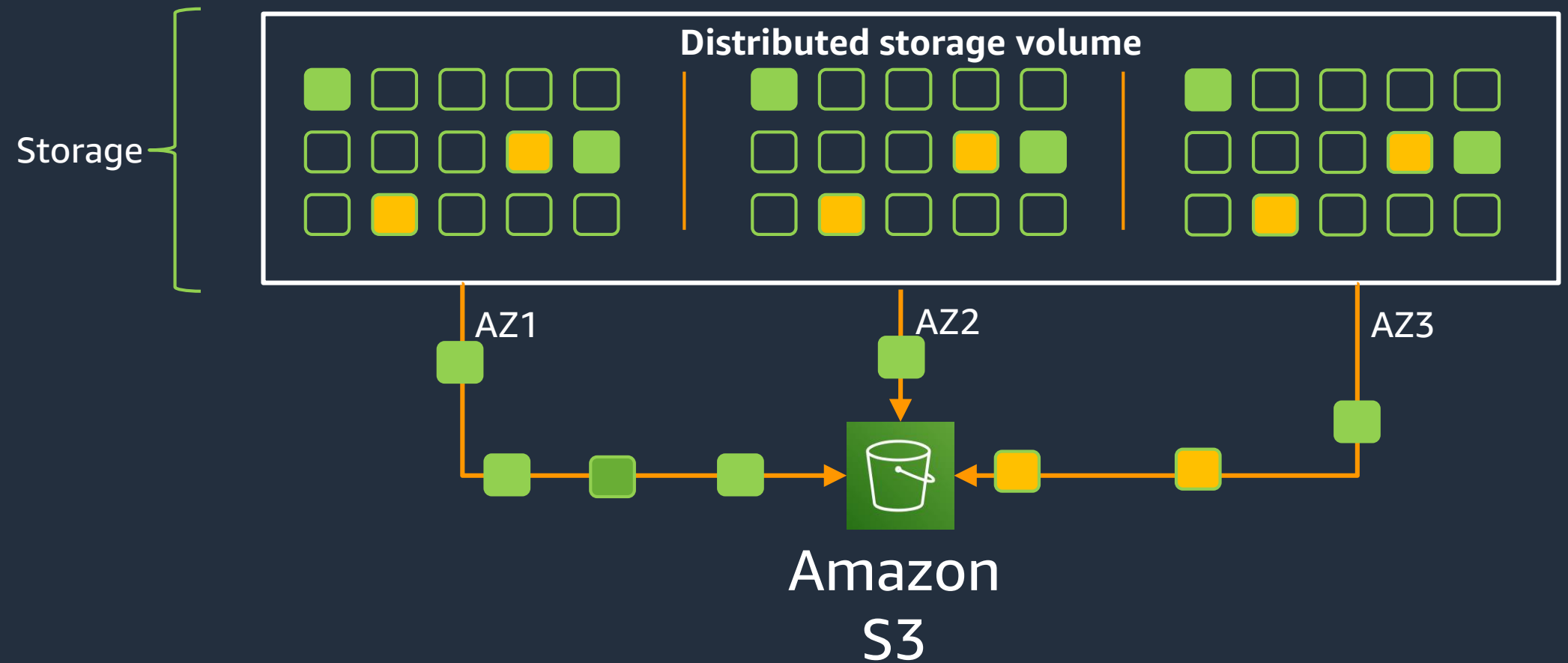
# Architecture

No impact backups

Backup

Continuous stream  
to Amazon S3

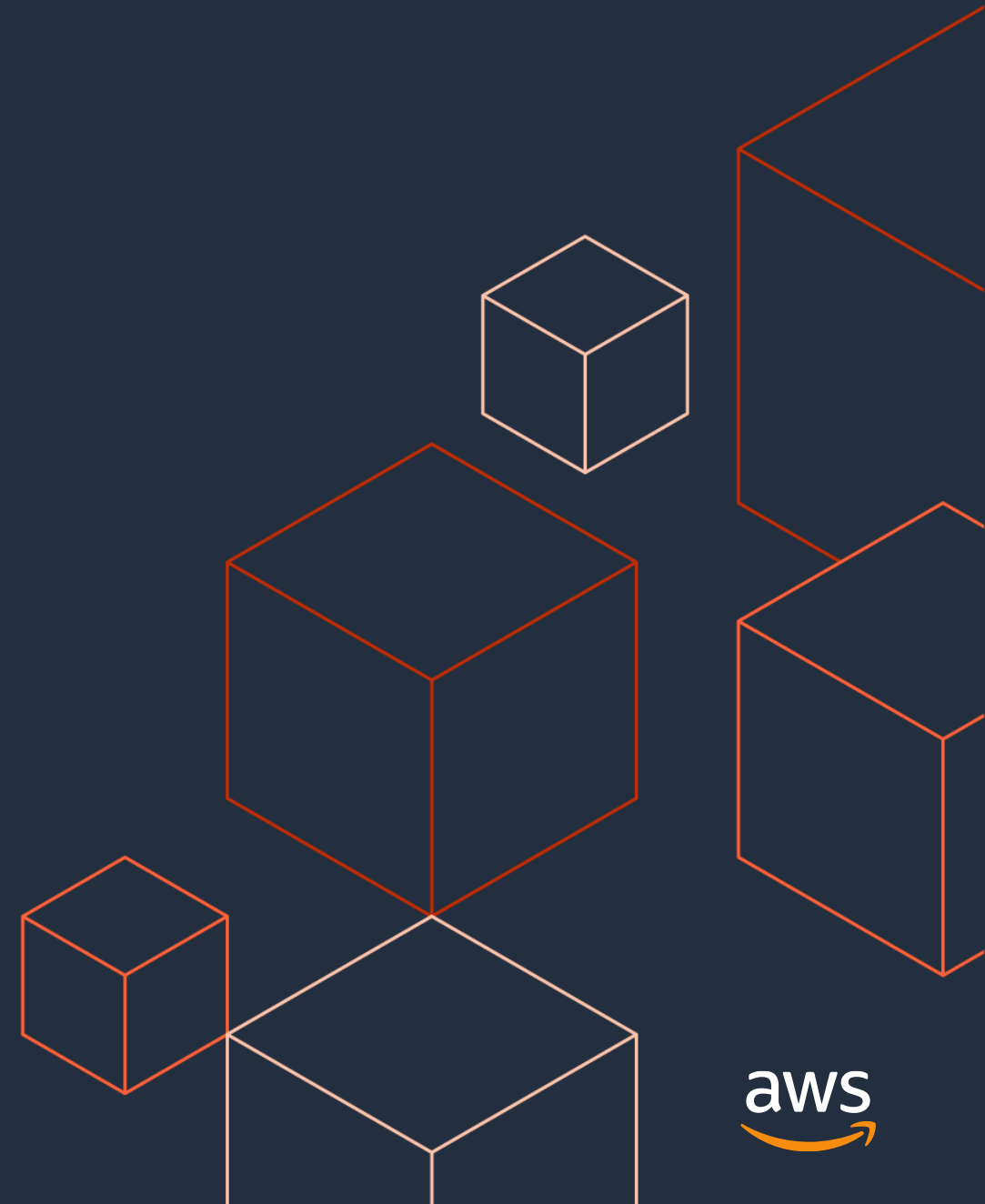
Point-in-time restore  
(35 days, second granularity)



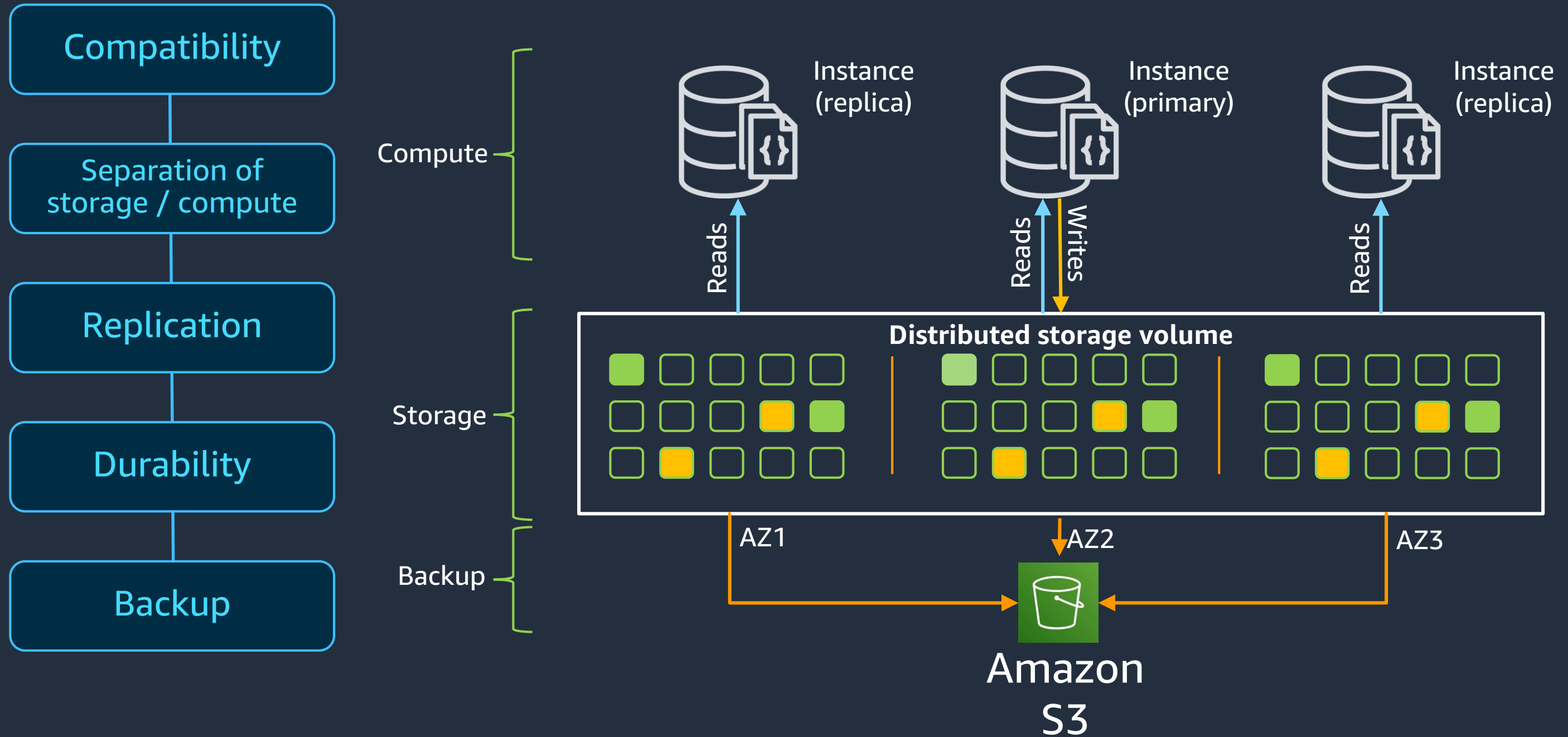
# Demo: Adding a new instance to a 12 TB cluster + backup



# Summary

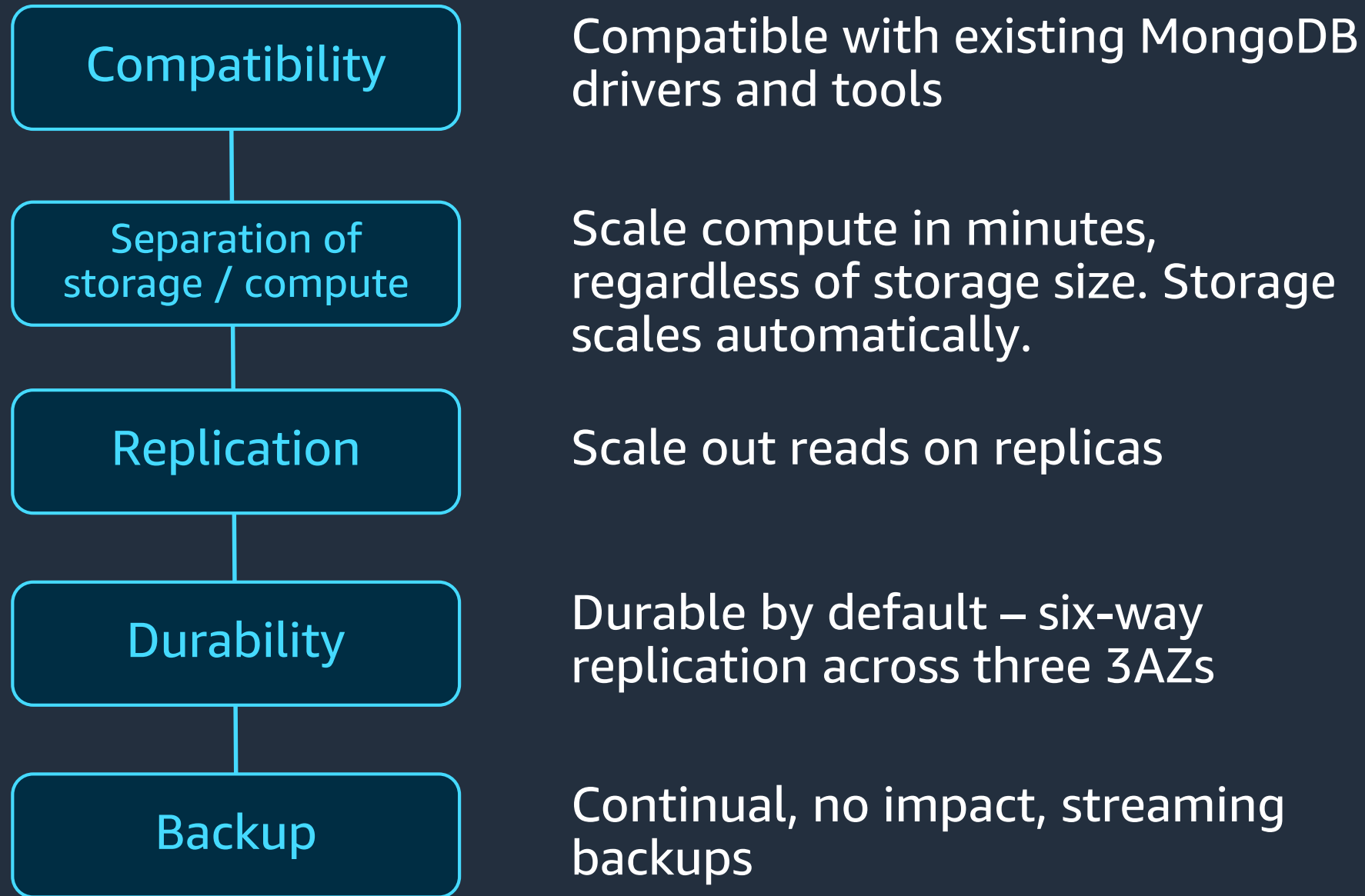


# Amazon DocumentDB Architecture



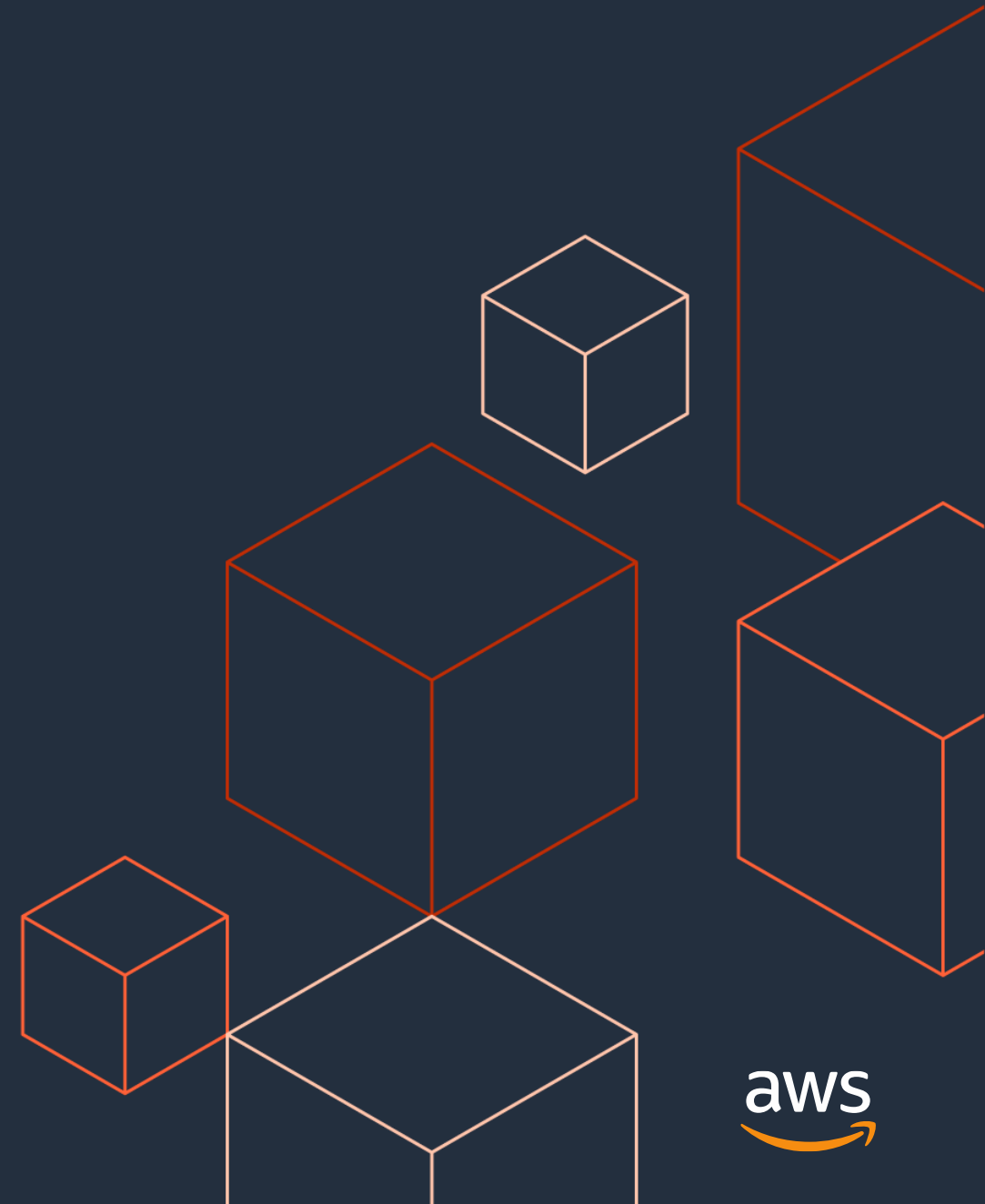


# Amazon DocumentDB Architecture



Modern, cloud-native database architecture

# Pricing



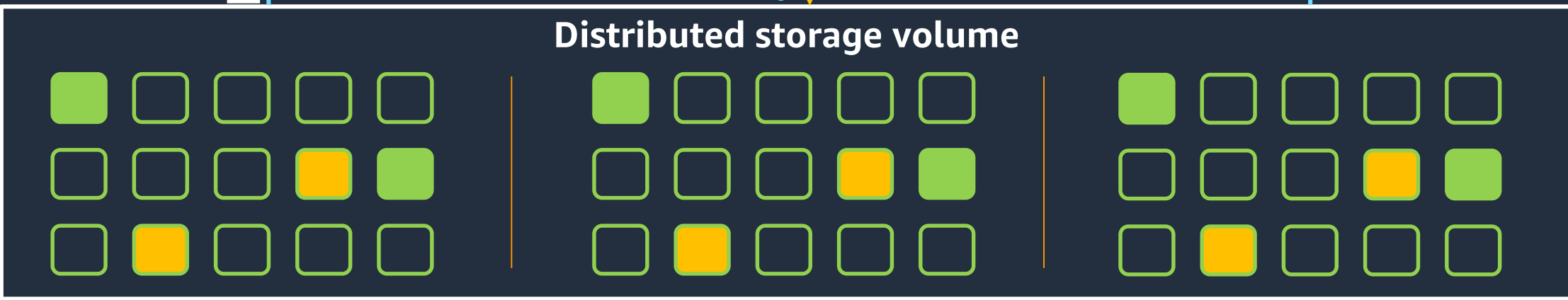
# Pricing (us-east-1)

1 Instances:  $\text{Size/hr} * \text{count}$  (db.t3.medium \$0.078/hr)

Compute  
Storage



2 I/O: Count (\$0.20/million)



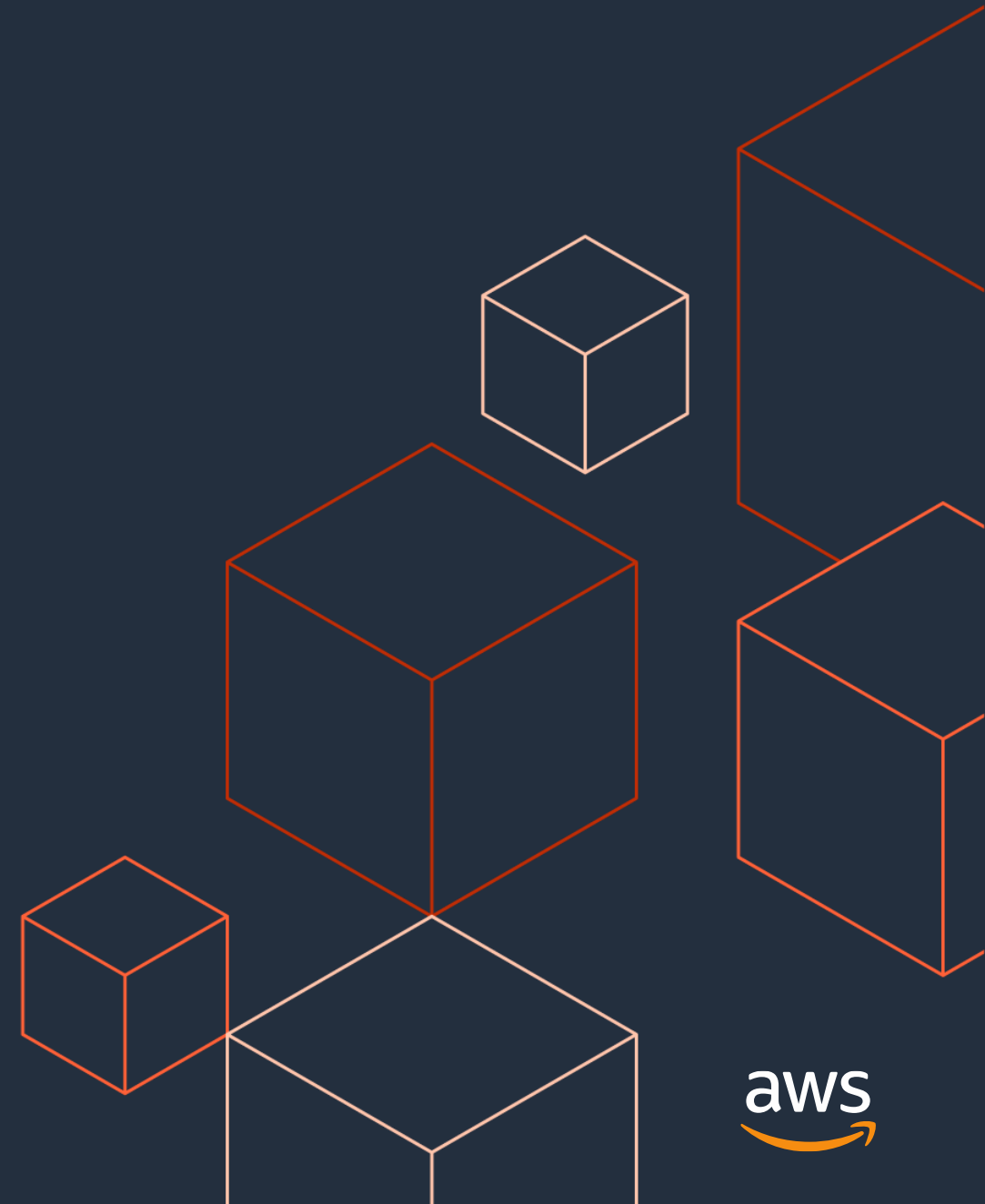
3 Storage: GB/mo (\$0.10/GB)



Amazon S3

4 Backup: GB/mo (100% Free! then \$0.021/GB)

# What is new?



# A few highlights from recent releases



MongoDB 4.0 + transactions



t3.medium instances



Role-based access control (built-in roles)



Cross-Region snapshot copy



Aggregation operators + stages + indexing improvements

And so  
much more!

# What's next?

“Amazon DocumentDB resources”

<https://aws.amazon.com/documentdb/resources/>

“Amazon DocumentDB immersion day workshop”

<https://documentdb-immersionday.workshop.aws/>

# Q&A

Ryan Thurston & Karthik Vijayraghavan



**Thank you!**

