# Amazon Elastic File System
Building Blocks for a Cloud-Native File System

Jacob Strauss, Principal Engineer

Storage Developer Conference
September 2020

## Cloud-Native File System

# Cloud-native file system

## Elastic

- Grow & shrink on demand
- No need to provision and manage infrastructure & capacity
- Pay as you go, pay only for what you use
- Simple to use, create a file system in seconds

## Scalable

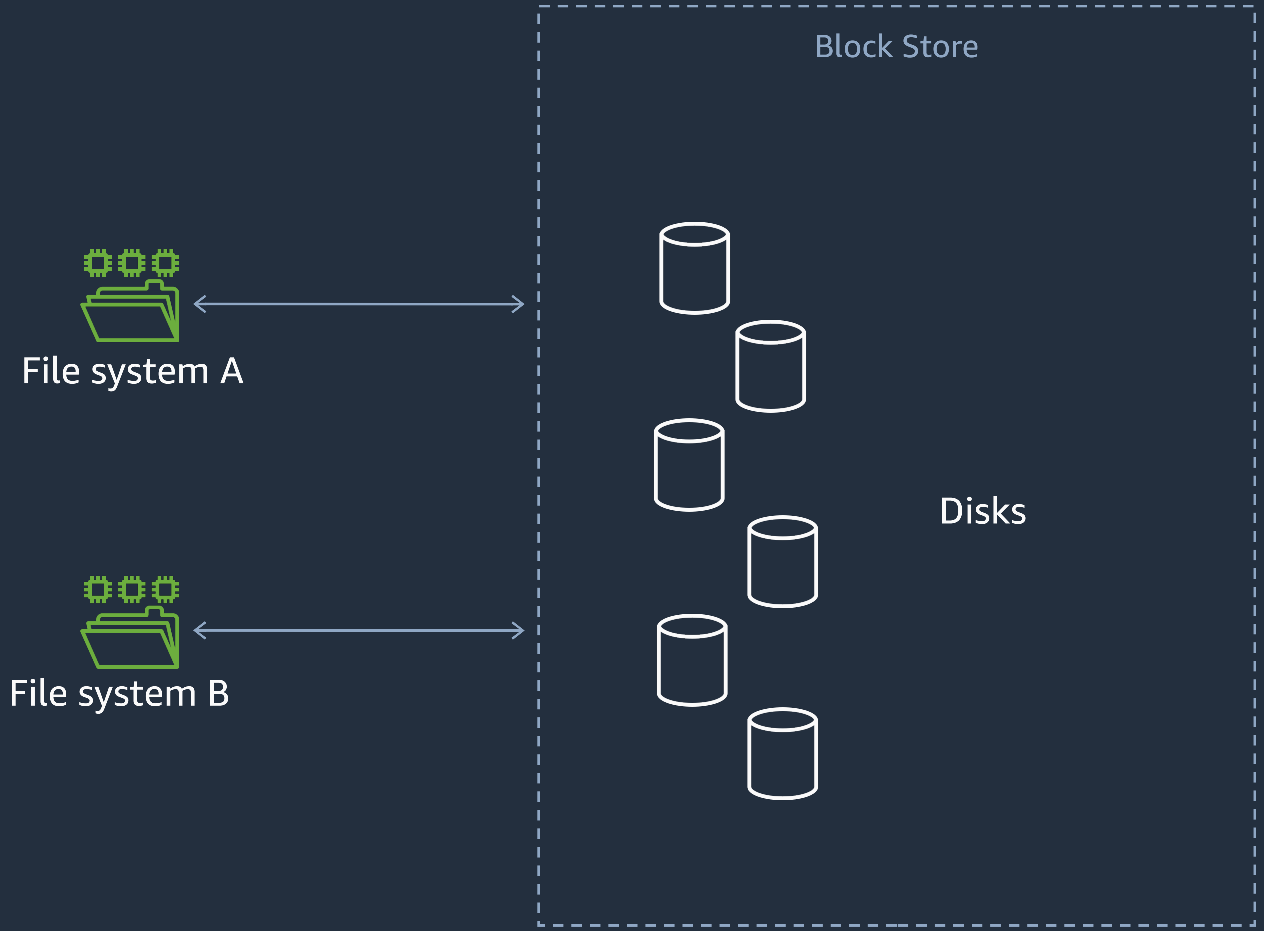
- Grow up to petabytes
- Performance modes for low latencies and maximum I/O
- Throughput that scales with storage
- Provisioned throughput available

## Integrated

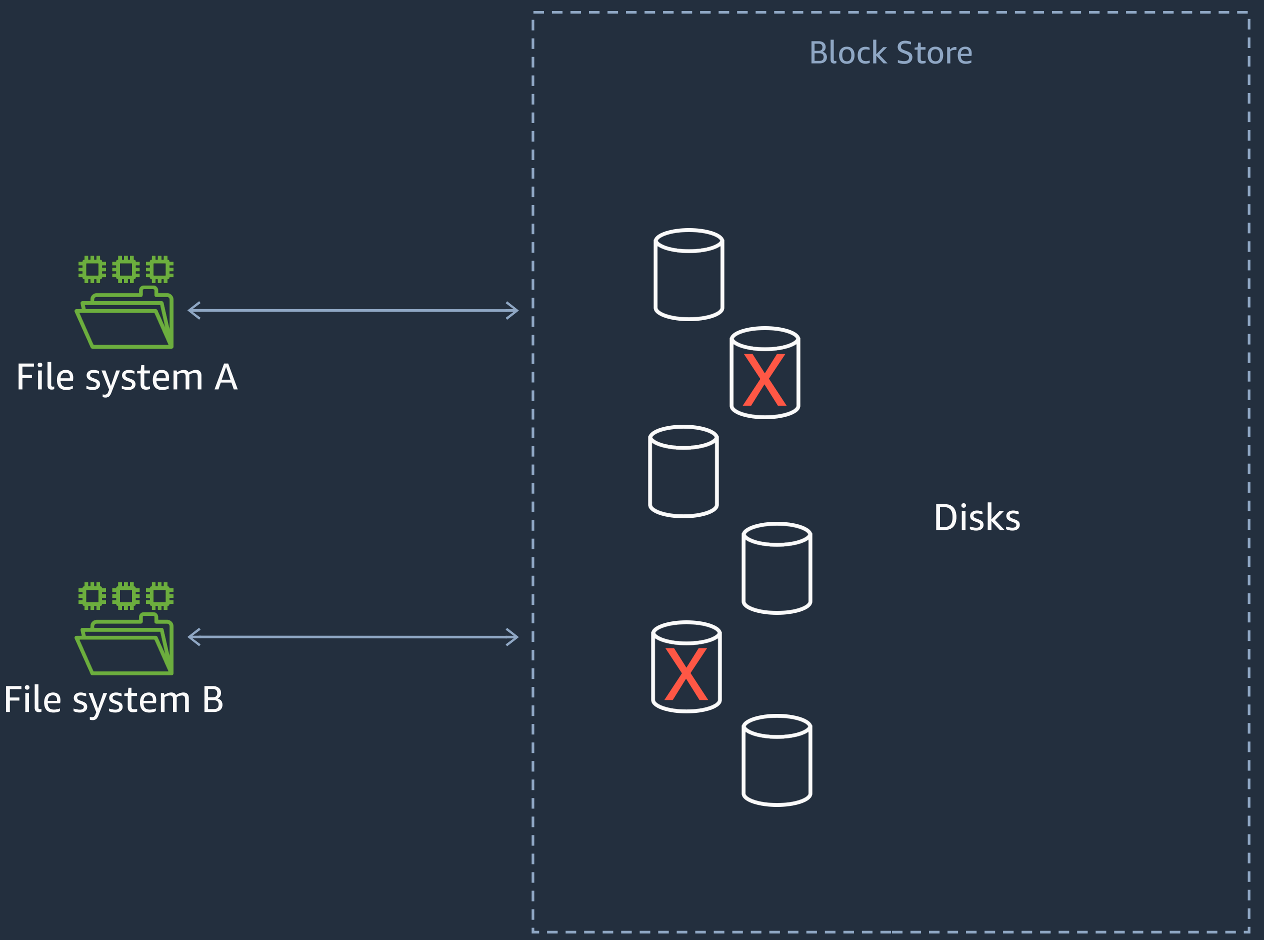
- Shared access from on-premises, inter region, and cloud-native applications
- Integrated with various AWS computing models
- Access concurrently from Amazon EC2, AWS Lambda, and Amazon ECS and EKS containers

## Deep Dive: Block Layer

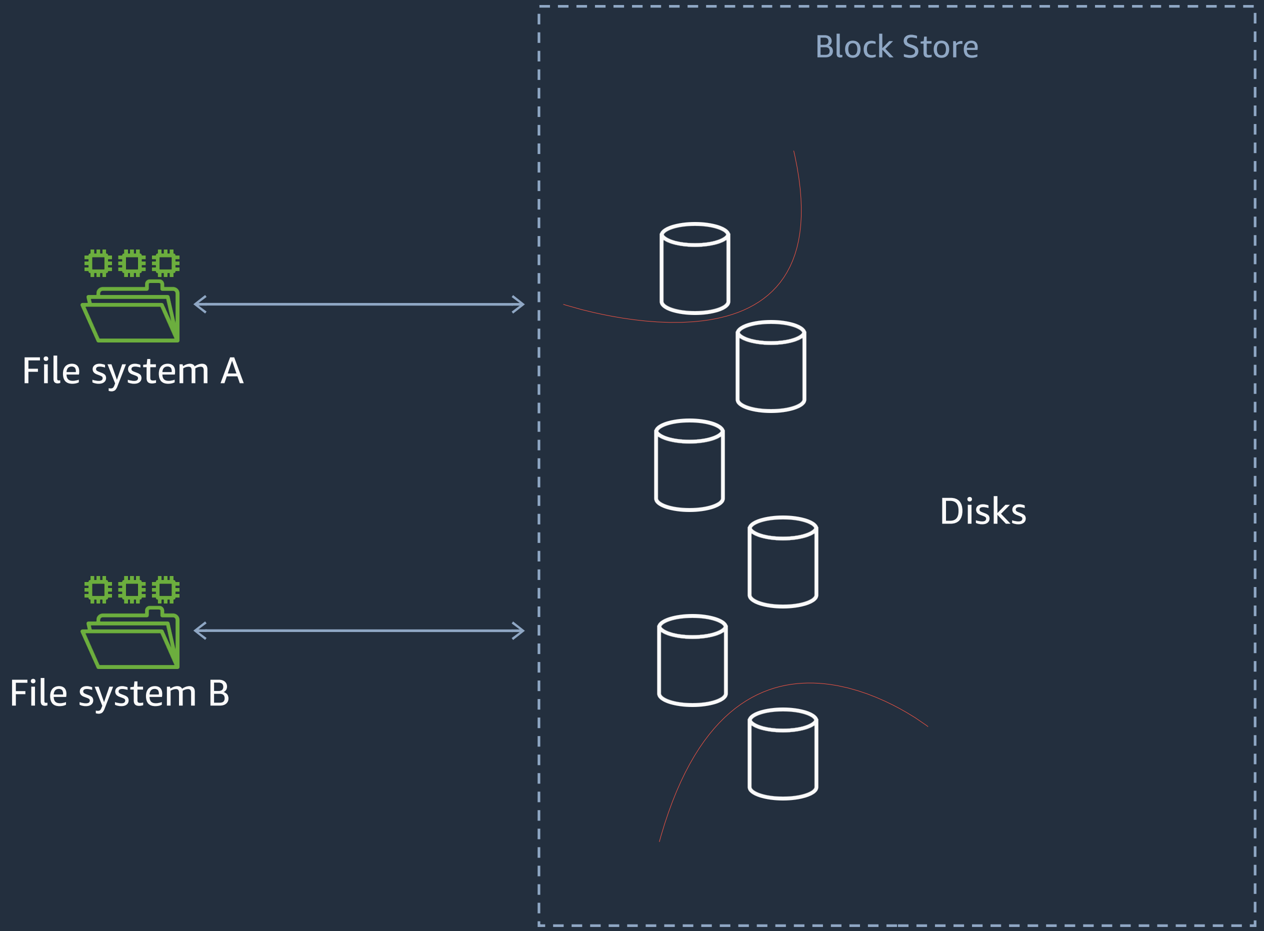
# Block responsibilities



## Block responsibilities: security & durability

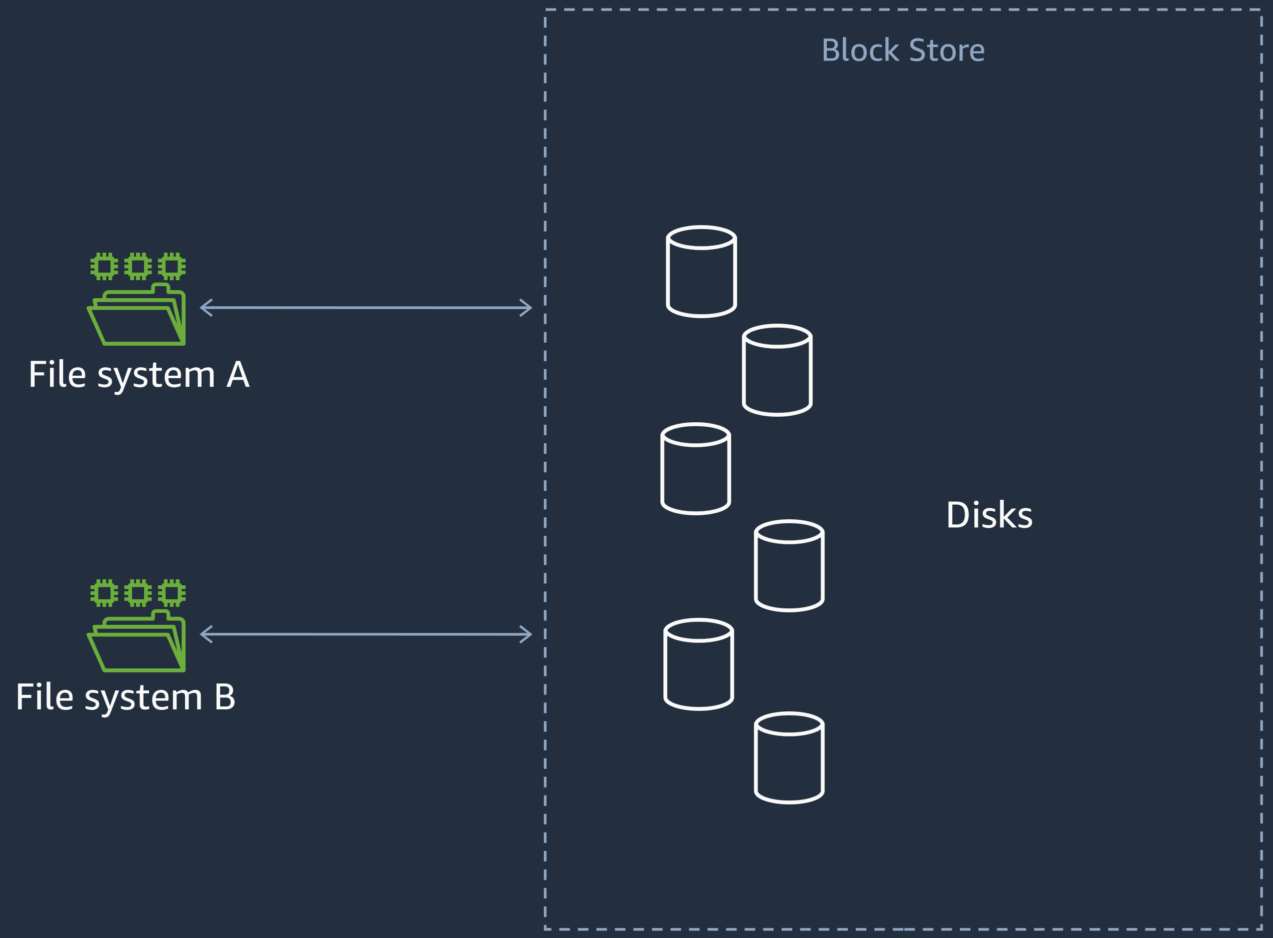


## Block responsibilities: availability

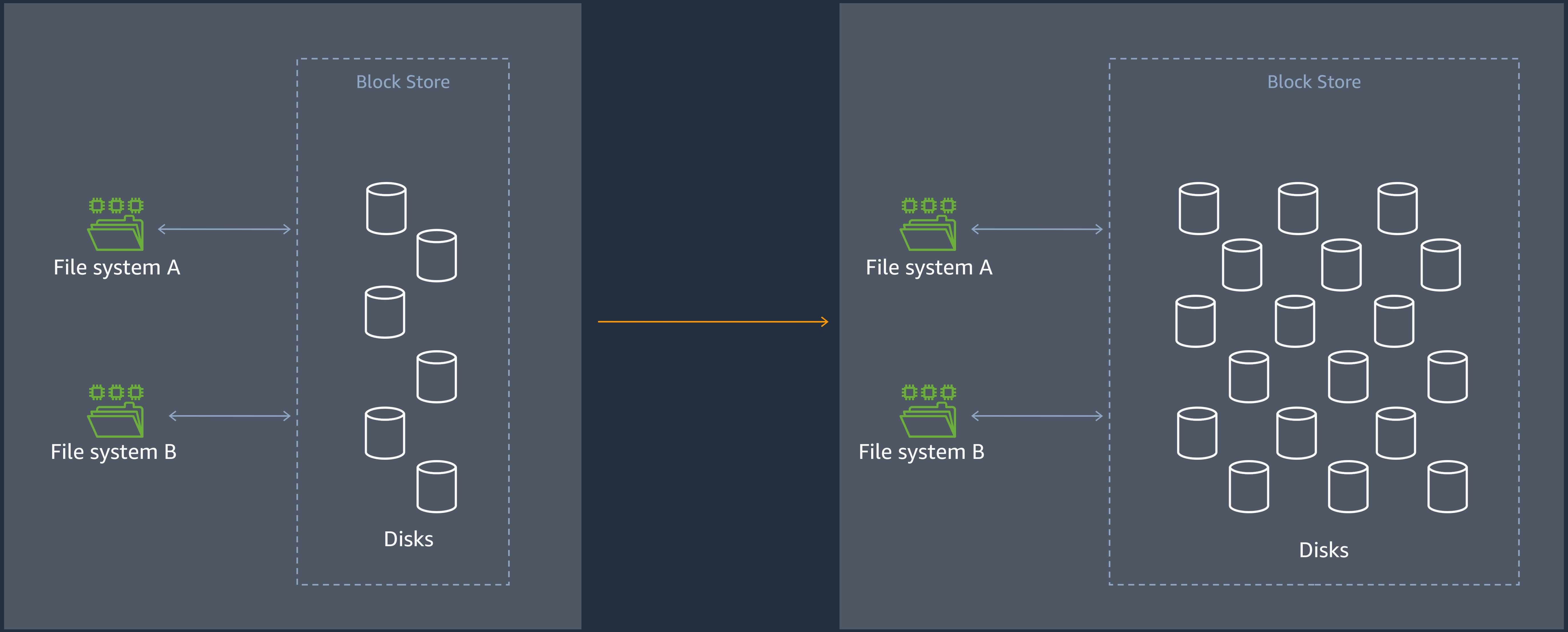


## Block responsibilities: naming, locating, interface

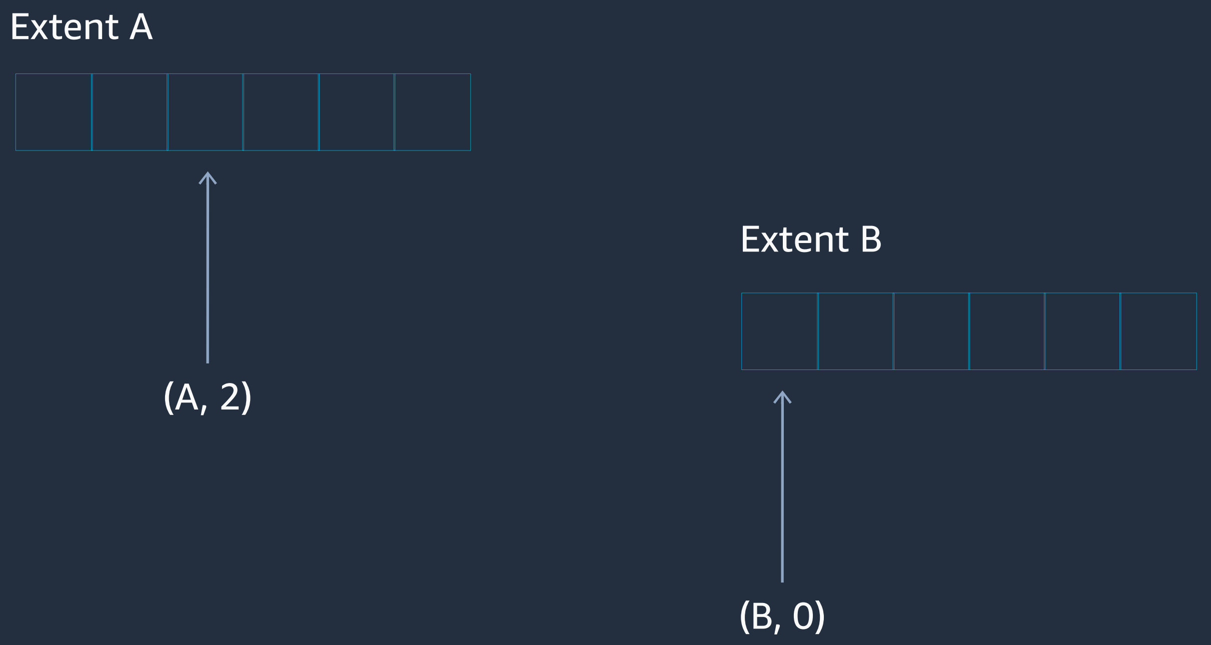
- Single address space
- Location independence
- Read, write, allocate, delete
- Ordering



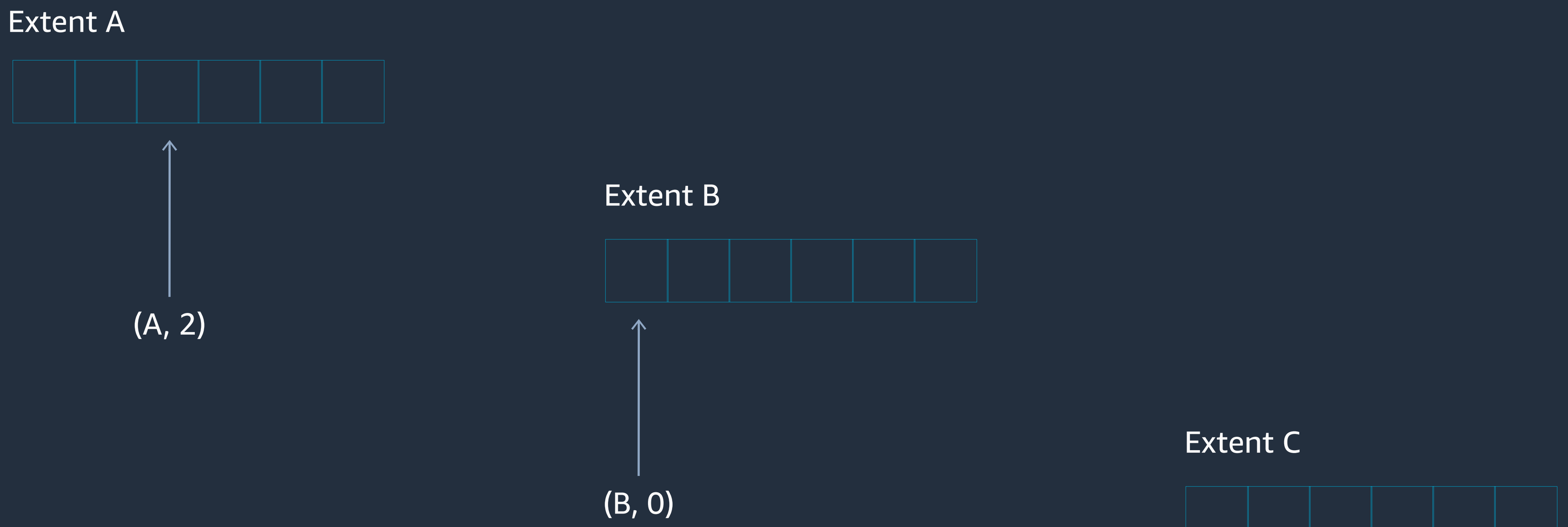
# Block responsibilities: elastic scaling



## Logical data structure: Extent

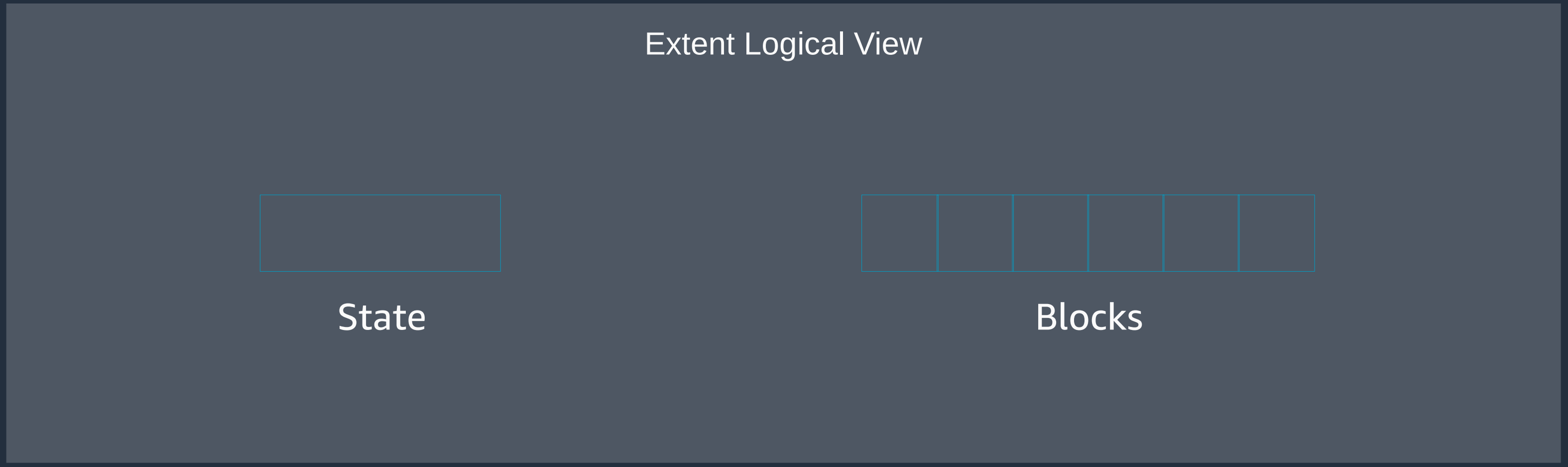


## Logical data structure: Extent

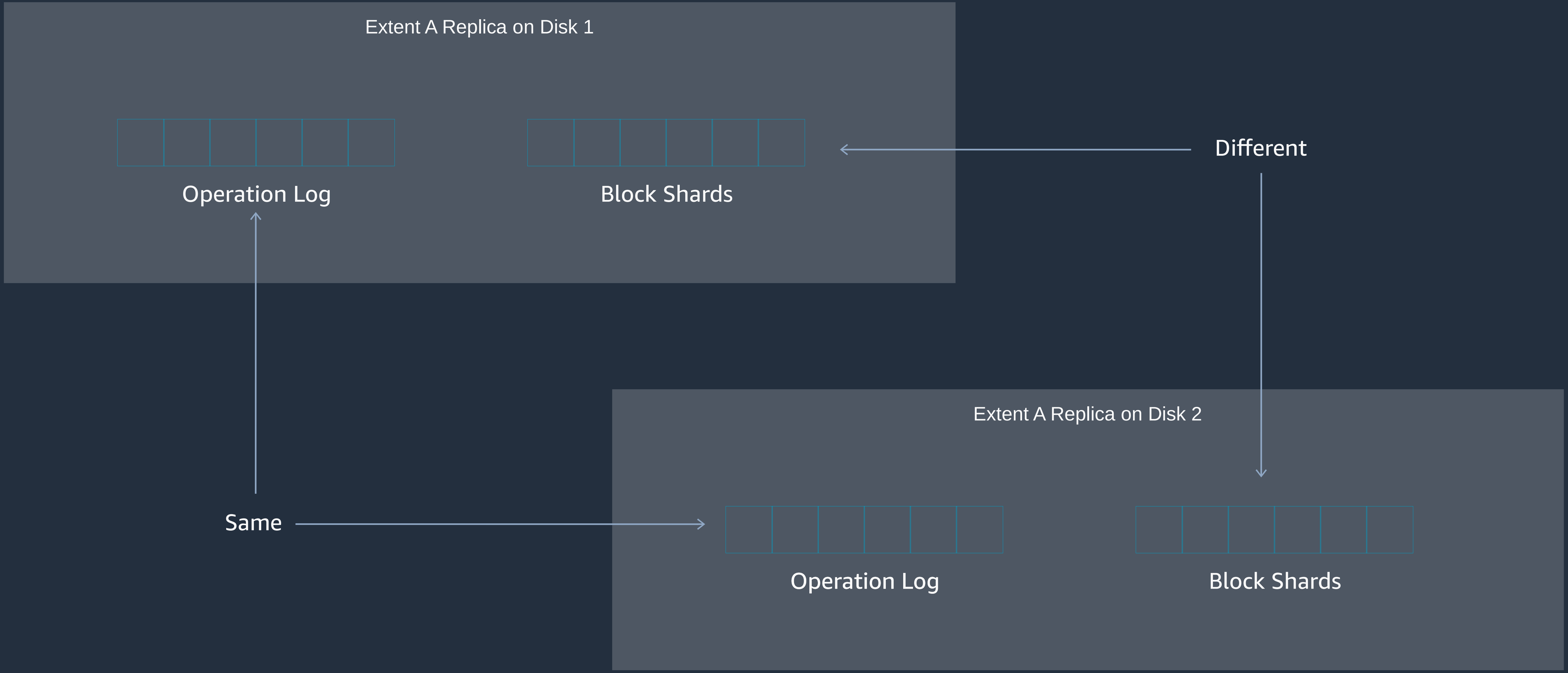


## What is an extent?

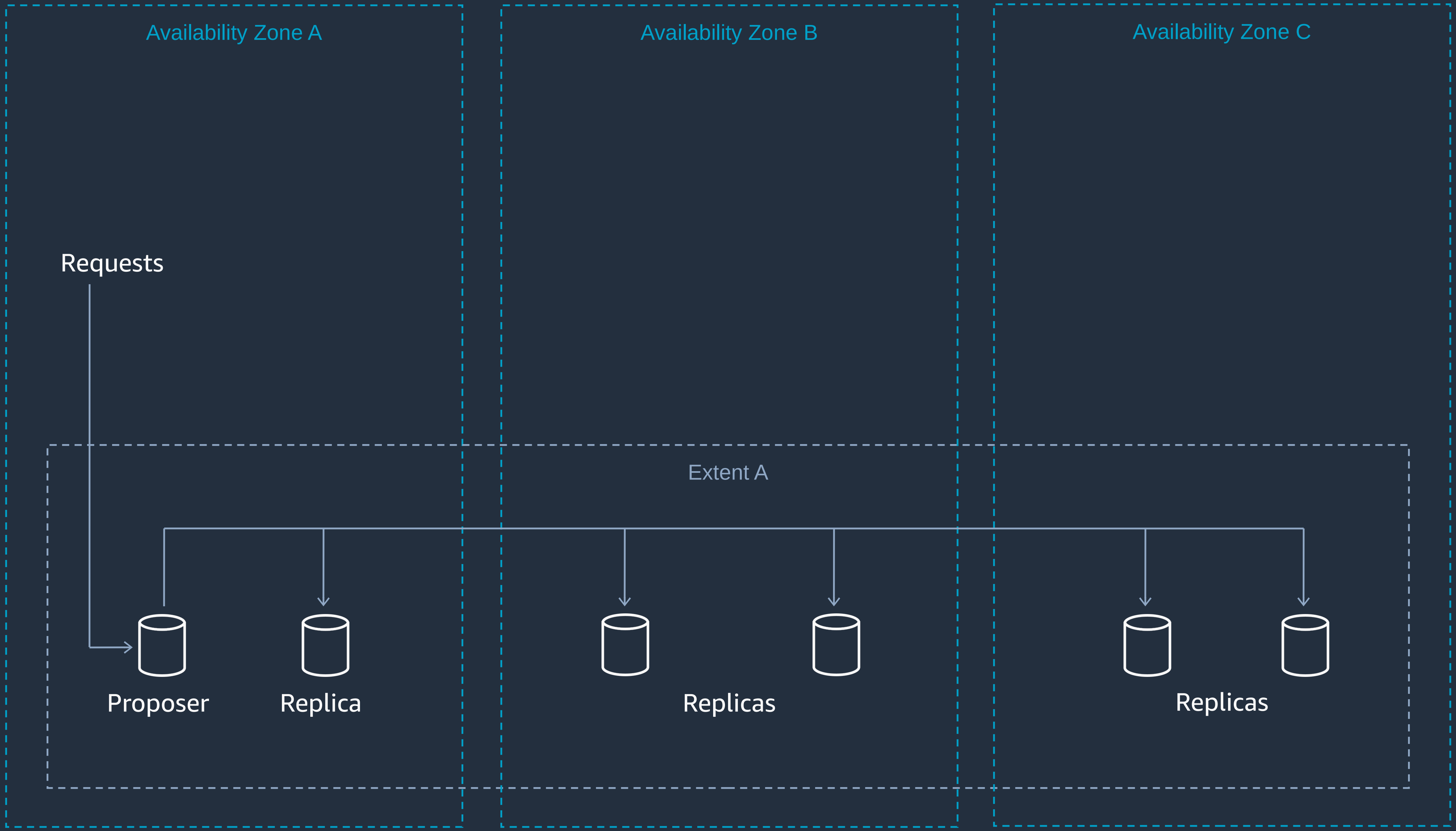
## Paxos Replicated State Machine



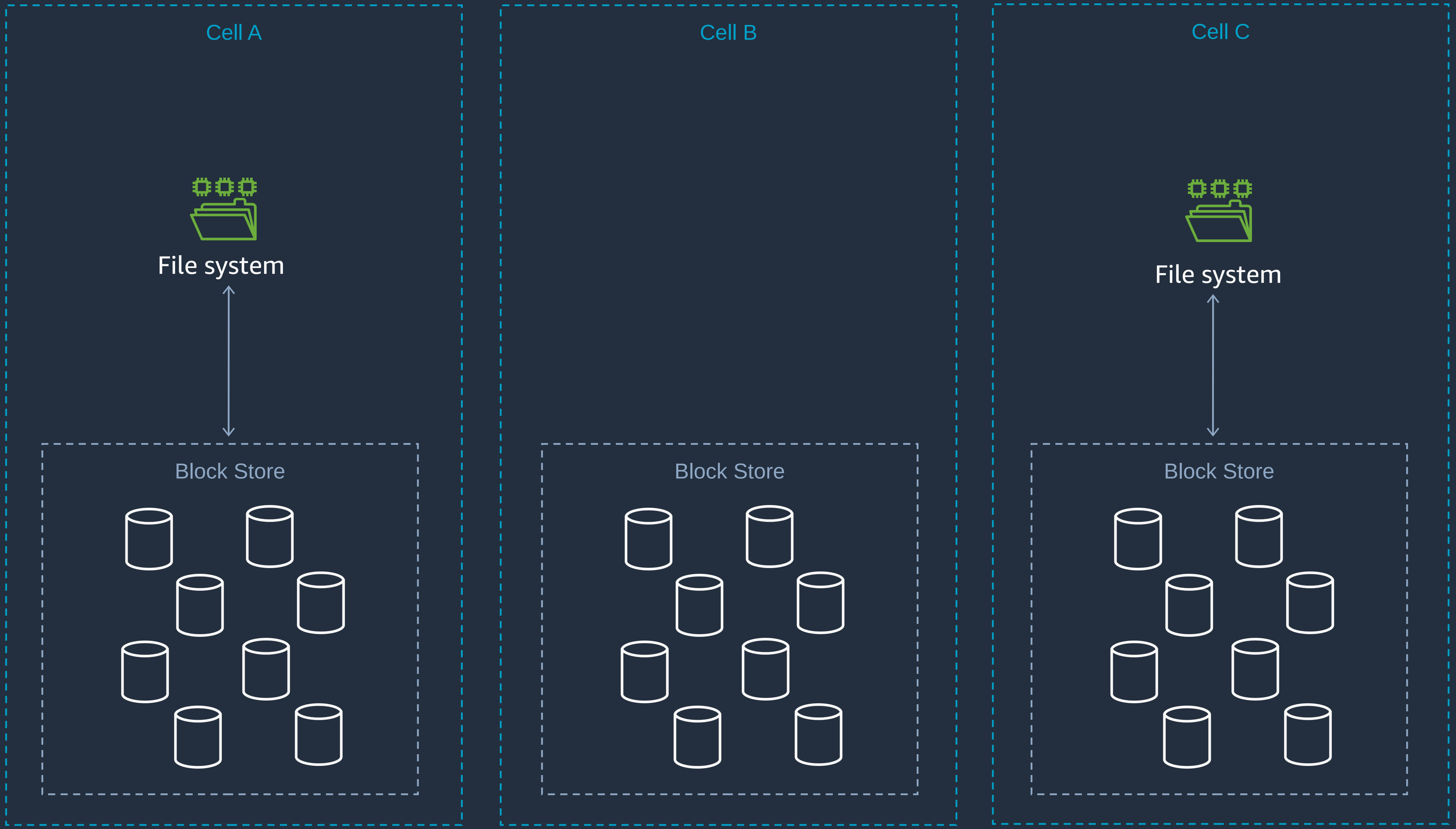
## What is an extent on disk?



# Extent replicas across availability zones



# Cellular architecture



## Operation ordering: conditional writes

Block Interface

```
 (data, version)  ← read(block)
         version ← write(block, data, version)
```

## Operation ordering: conditional writes

Block Interface

```
 (data, version) ← read(block)
         version ← write(block, data, version)

       version[] ← multiWrite(block[], data[], version[])
```

# Operation ordering: conditional writes

## Block Interface

```
(data, version) ← read(block)
        version ← write(block, data, version)

      version[] ← multiWrite(block[], data[], version[])
```

## File Operations

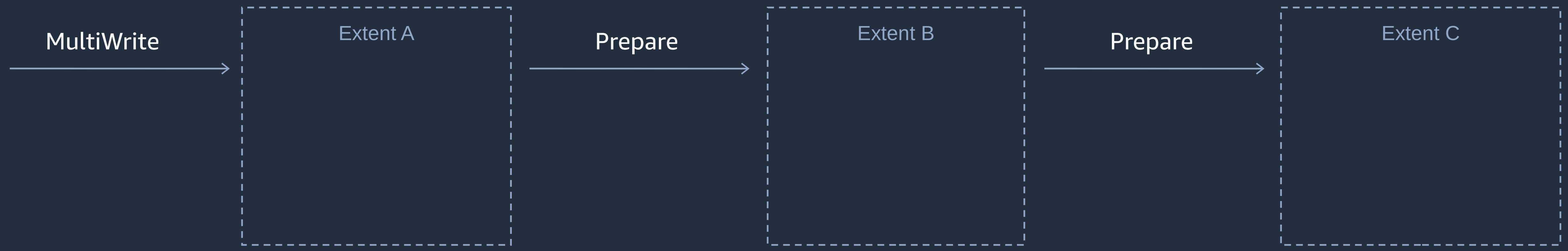
Begin transaction

    Read blocks

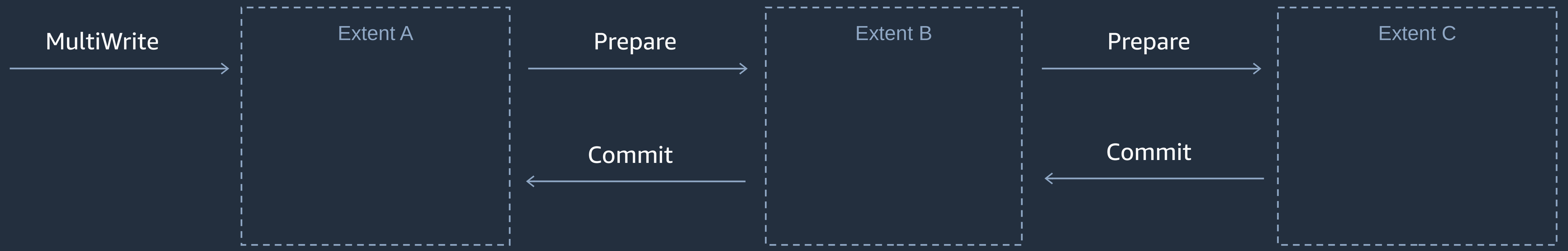
    Modify blocks

Commit or start over

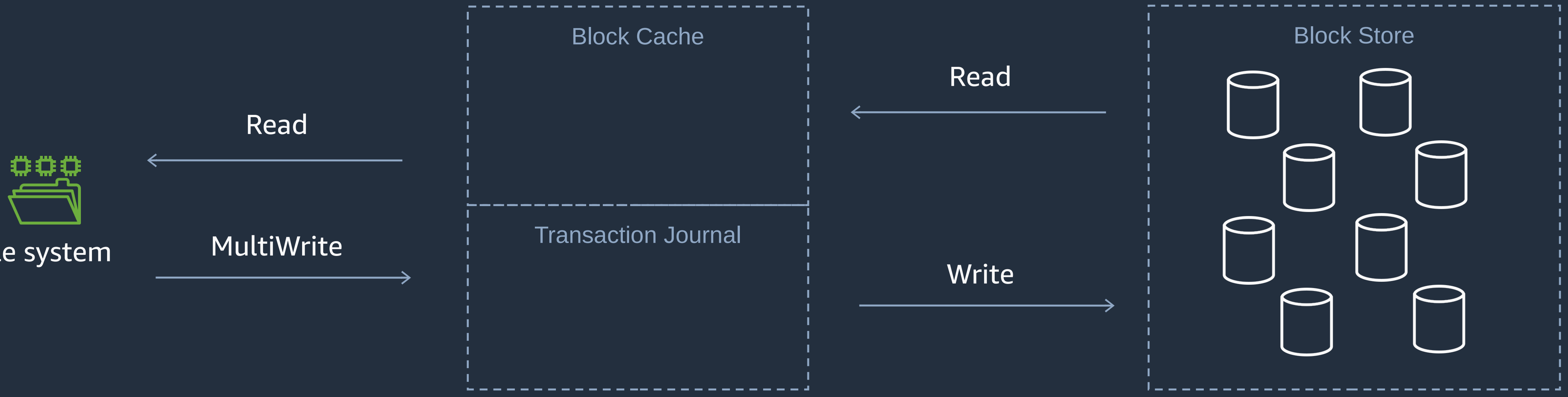
## Multi-block conditional write is a 2-phase commit



## Multi-block conditional write is a 2-phase commit



# Reduce transaction latency



## Cloud-native file system building blocks

### Properties

- Durable and secure
- Available
- Elastic scaling
- Simple interface
- Resilient

### Techniques

- Replicated state machine
- Multi-AZ consensus
- Cellular architecture
- Optimistic transactions
- Scale-out and lower latency

# Thank You!

Jacob Strauss
jsstraus@amazon.com