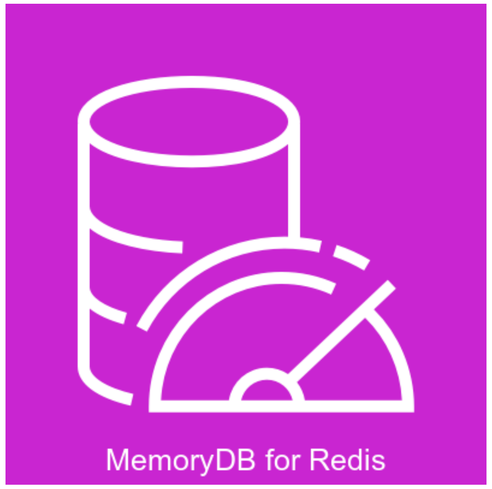
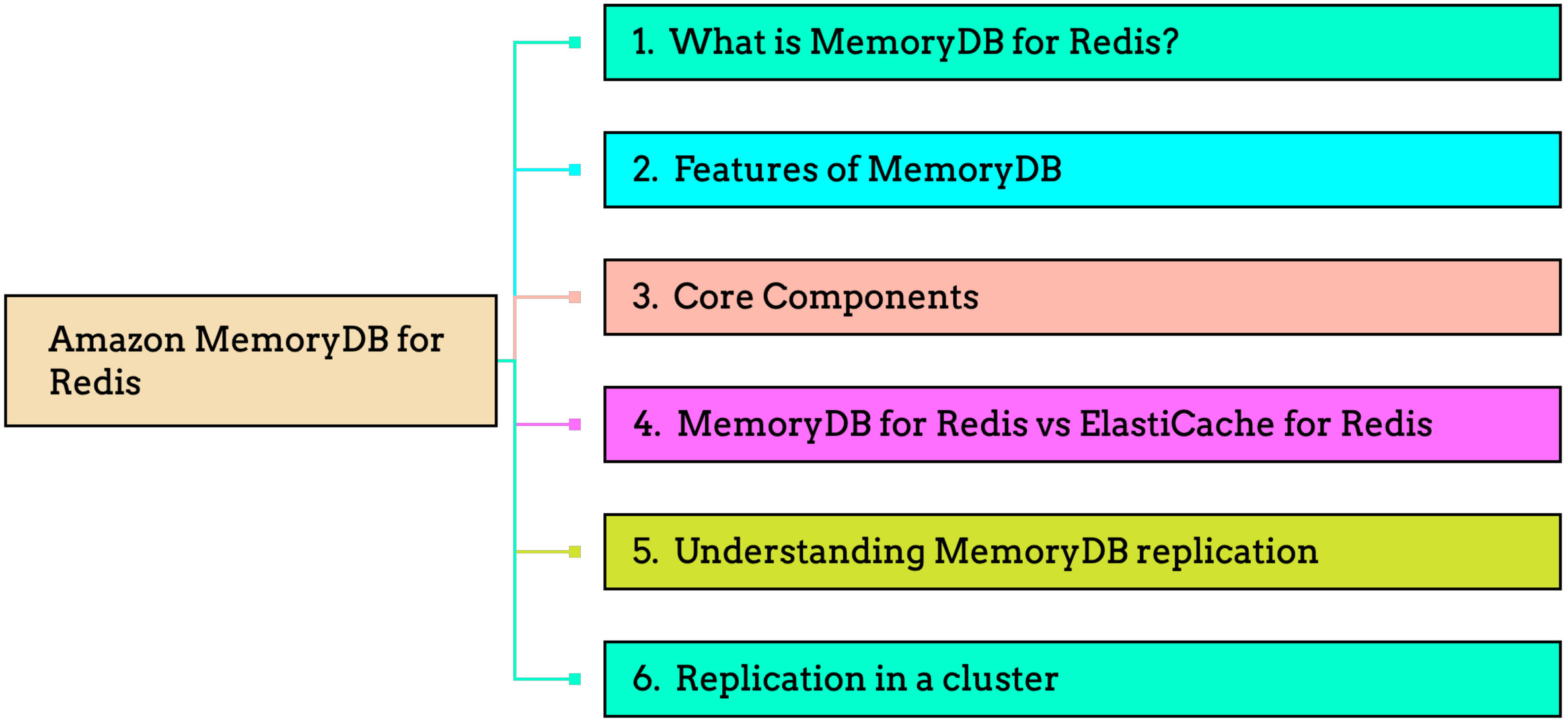
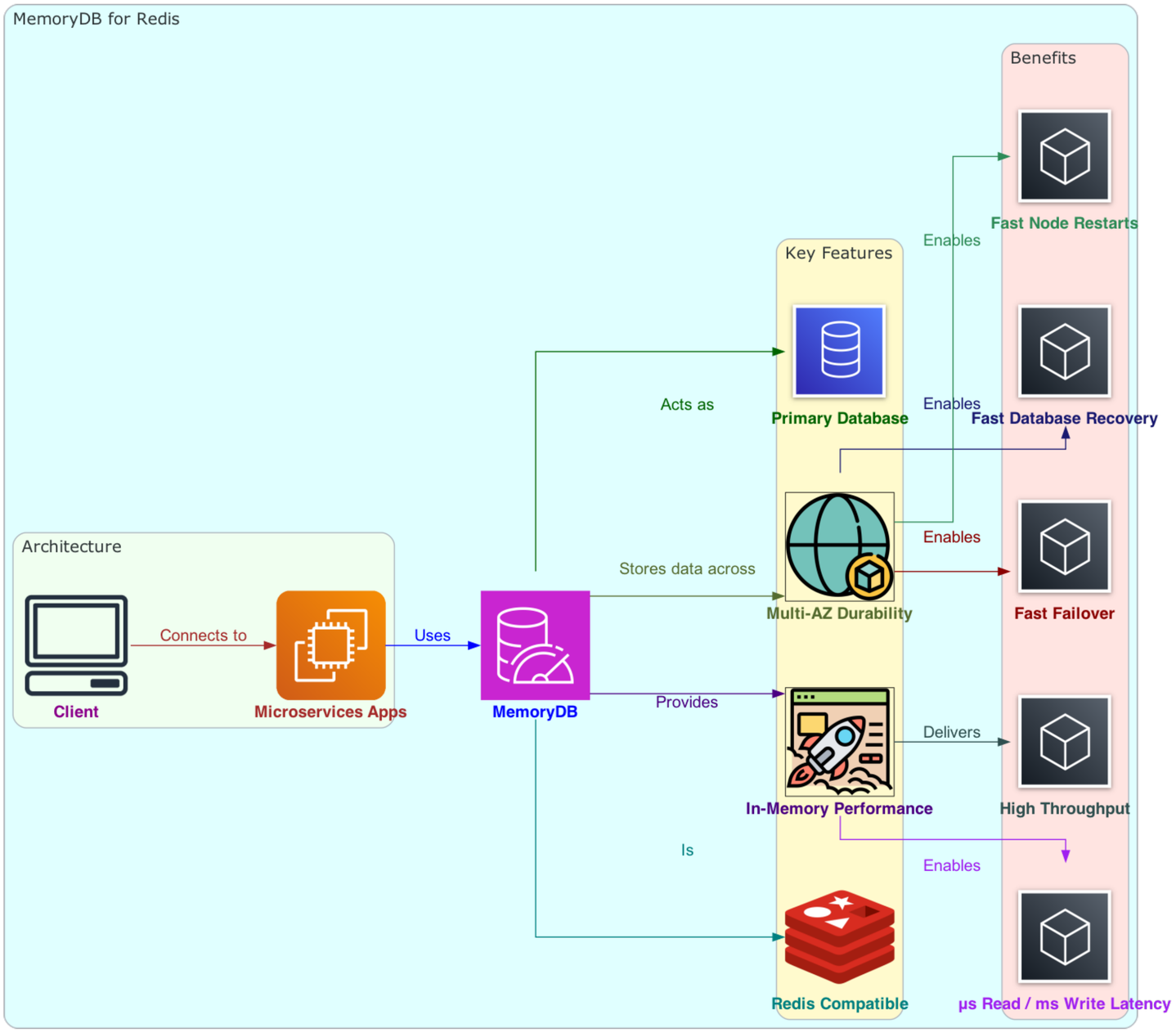


Amazon MemoryDB for Redis

Table of Contents



What is MemoryDB for Redis?



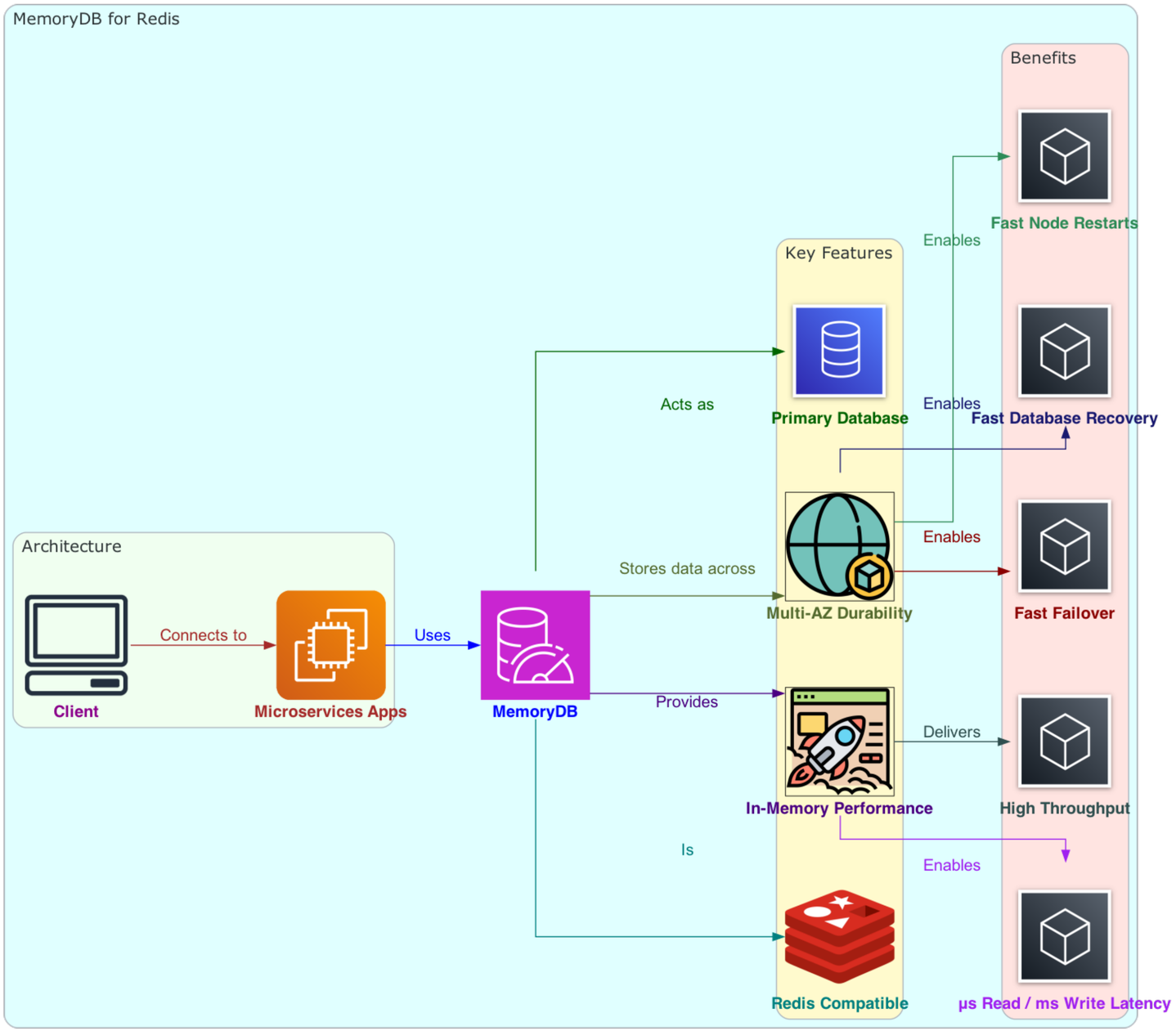
1. 🌐 Durable, in-memory database service
   - 📦 Database service
   - 🗃️ Provides durability
   - 🧠 Stores data in memory
   - 🏎️ Fast access

2. ⚡ Ultra-fast performance
   - 💾 In-memory data storage
   - ⏱️ µs read, ms write latency
   - 🌊 High throughput

3. 🎯 Purpose-built for microservices architectures
   - 🏗️ Designed for modern apps
   - 🔍 Caters to microservices needs

4. 🔄 Redis-compatible
   - 🤝 Compatible with Redis
   - 🔧 Familiar data structures, APIs, commands

What is MemoryDB for Redis?



5. 💾 In-memory data storage
📩 All data stored in memory
⚡ Enables lightning-fast performance
⏬ Low latency

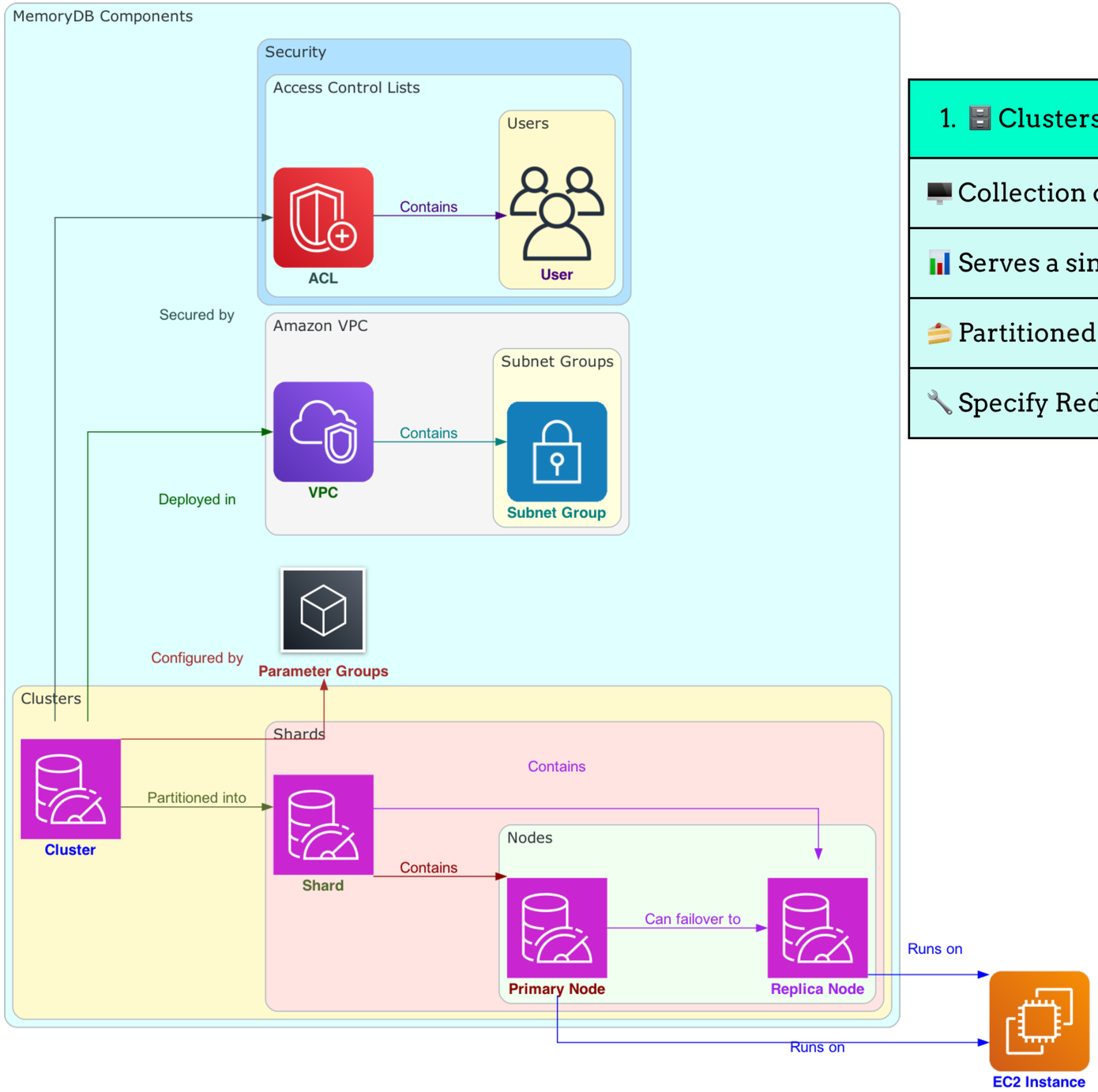
6. 🌍 Multi-AZ durability
🌎 Stores data across multiple AZs
📒 Uses Multi-AZ transactional log
🔀 Enables fast failover
🔄 Database recovery
🔁 Node restarts

7. 🐿️ High-performance primary database
⚡ In-memory performance
🌍 Multi-AZ durability
🗑️ Eliminates separate cache, durable DB management

Features of MemoryDB

1. 🔒 **Strong consistency**: 🎯 Primary nodes, ⏳ Eventual consistency for replicas
2. ⚡ **High performance**: ⏱️ µs read, ms write latencies, 🚀 Up to 160M TPS per cluster
3. 🔧 **Redis compatibility**: 🧩 Flexible data structures, APIs, 🚚 Easy migration
4. 🗃️ **Data durability**: 🌍 Multi-AZ transactional log, 🔄 Fast recovery, restart
5. 🎭 **Multi-AZ availability**: 🔀 Automatic failover, 🩹 Node failure detection, recovery
6. 📏 **Scalability**: ➕ Horizontal scaling with nodes, ⬆️ Vertical scaling with node types, 🔪 Scale write throughput with shards, 📖 Scale read throughput with replicas
7. 📖 **Read-after-write consistency**: 🎯 Primary nodes, ⏳ Eventual consistency for replicas
8. 🔐 **Security**: 🔒 Encryption in transit (TLS), 🗝️ Encryption at rest (AWS KMS), 🛡️ User authentication, authorization (Redis ACLs), 🔑 IAM integration, tag-based access control
9. 📸 **Automatic snapshots**: 💾 Stored in Amazon S3, 📅 Up to 35 days retention
10. 📊 **Cluster capacity**: 🖥️ Up to 500 nodes, 💿 Over 100 TB storage per cluster, 🎭 1 replica per shard
11. 🤖 **AWS Graviton2 support**
12. 🤝 **AWS integration**: 📈 CloudWatch monitoring, 🌐 Amazon VPC networking, 🔍 CloudTrail logging, 📪 Amazon SNS notifications
13. 👷 **Fully managed**: 🩹 Automated patching, upgrades

Core Components

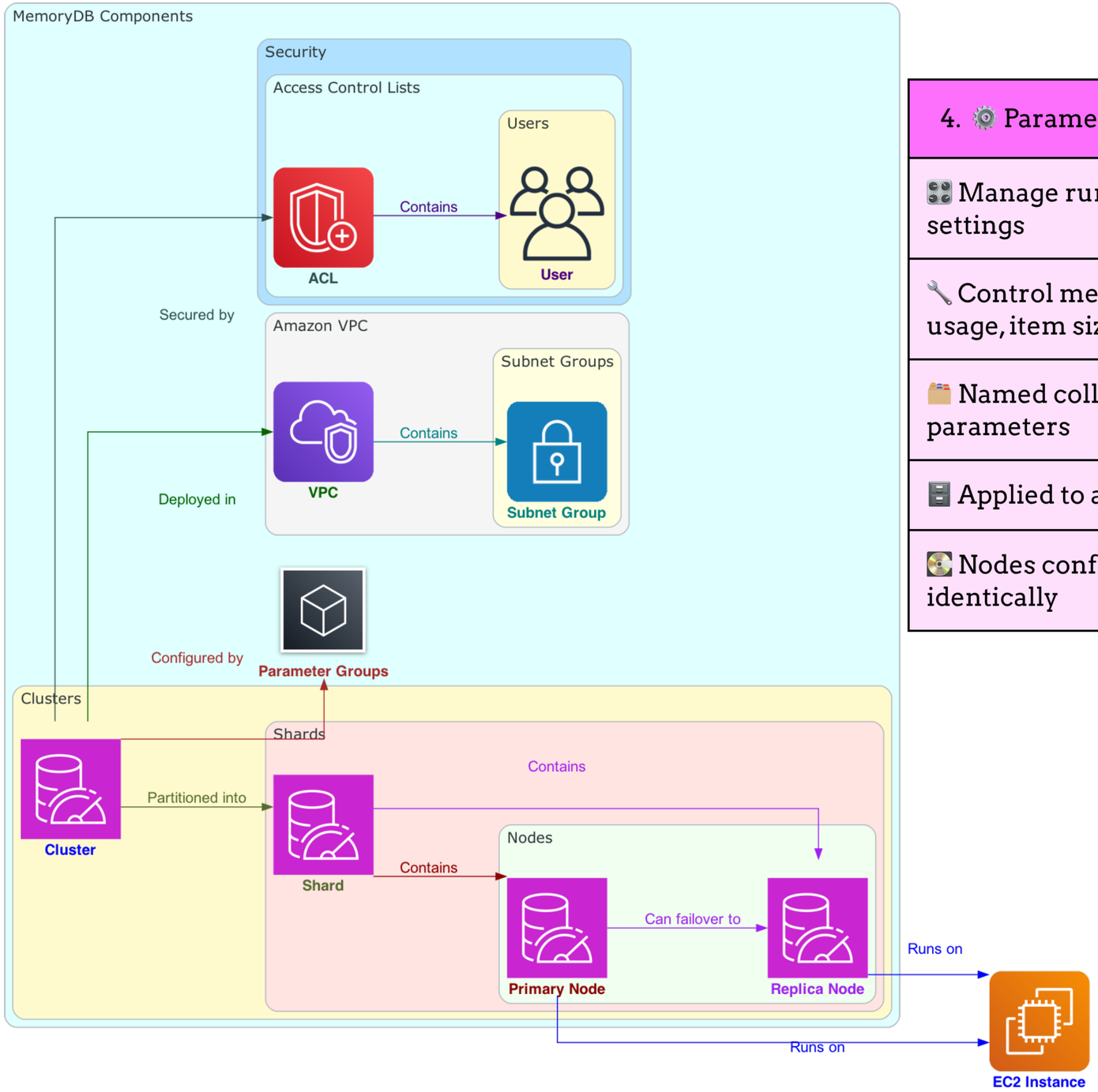


1. 🗄️ Clusters
🖥️ Collection of nodes
📊 Serves a single dataset
🍰 Partitioned into shards
🔧 Specify Redis version

2. 💿 Nodes
🧱 Smallest building block
🖥️ Runs on EC2 instance
🔧 Runs chosen Redis version
🗄️ Belongs to a shard, cluster
⚙️ Scalable to different types
💾 Multiple types, varying memory

3. 🍰 Shards
🗃️ Grouping of 1-6 nodes
✍️ 1 primary write node
📖 Up to 5 read replicas
🗄️ Cluster has at least 1 shard
📊 Up to 500 shards per cluster
🍰 Data partitioned across shards

Core Components



4. ⚙️ Parameter groups

- 🎛️ Manage runtime settings
- 🔧 Control memory usage, item sizes
- 🗂️ Named collection of parameters
- 🗄️ Applied to a cluster
- 💽 Nodes configured identically

5. 🌐 Subnet Groups

- 🗃️ Collection of subnets
- 🔒 Typically private
- ☁️ For clusters in Amazon VPC
- 🗄️ Specify or use default
- 🖥️ Associates IPs with nodes

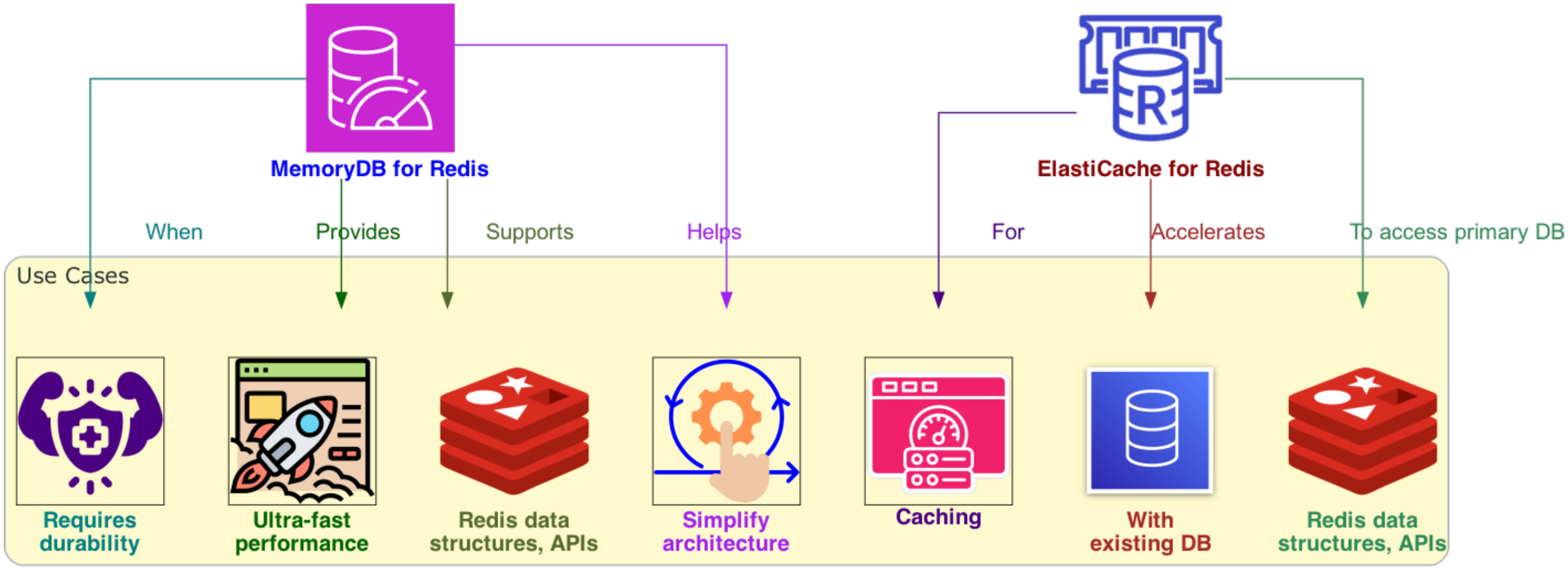
6. 🔐 Access Control Lists

- 👥 Collection of users
- 🔑 Follows Redis ACL rules
- 🗝️ Authorizes user access

7. 👤 Users

- 🔑 Username and password
- 📊 Accesses data, issues commands
- 👥 Member of an ACL
- 🗝️ ACL determines permissions

VS MemoryDB for Redis vs ElastiCache for Redis



1. 🥞 MemoryDB: Durable, in-memory primary database
🗃️ Durable, in-memory database
⚡ Ultra-fast workloads

2. ⚡ Ultra-fast performance: µs read, ms write latency
⏱️ µs read, ms write latency
🏎️ High-speed data access

3. 🧩 Redis data structures & APIs with durability
🔧 Familiar Redis data structures, APIs
🗃️ Ensures data durability
🎯 Primary, durable database

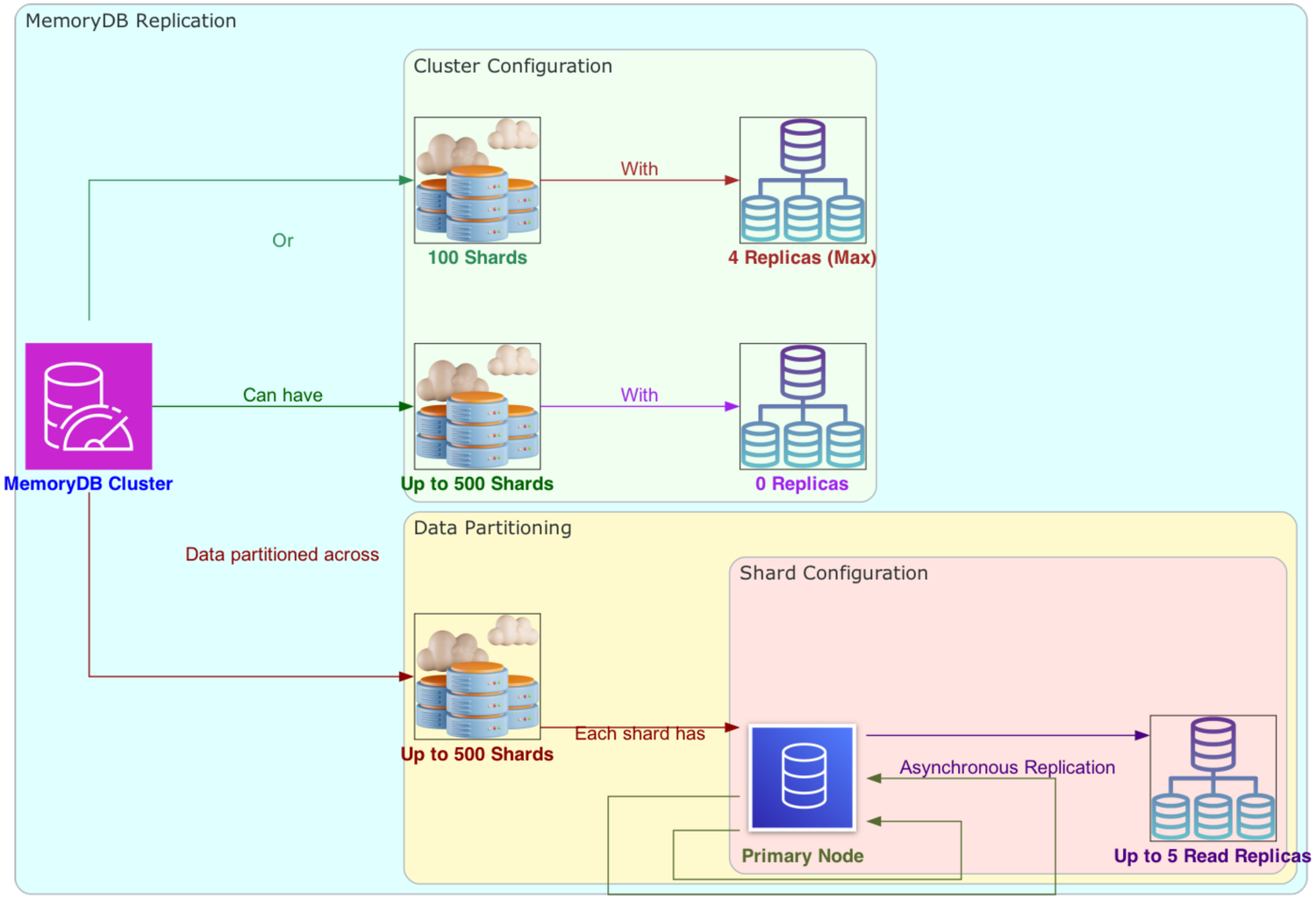
4. 🎯 Simplify architecture, lower costs
🏗️ Simplify application architecture
💸 Reduce costs
🗑️ Replace database with cache

5. 🌐 ElastiCache: Caching service for existing databases
🗃️ Caching service
🏎️ Accelerates data access

6. 🏎️ Accelerate data access: µs read/write
⏱️ µs read/write performance
🚀 Speed up data access
🗃️ Existing primary database or data store

7. 🔧 Use Redis to access primary database data
🧩 Leverage Redis data structures, APIs
📊 Access data in primary database or data store

Understanding MemoryDB replication



1. 🍰 Data partitioning

📊 Across up to 500 shards

2. 🗃️ Shard configuration

✍️ Single read/write primary node

📖 Up to 5 read-only replicas

3. 📈 Primary node performance

🚀 Sustains up to 100 MB/s

4. 🖥️ Cluster configuration

⚖️ Up to 500 nodes per cluster

🔢 Higher shards, lower replicas

5. 🔪 Configuration range

🗄️ 500 shards, 0 replicas

💯 100 shards, 4 replicas (max)

Replication in a cluster

1. 📖 Read replicas

🗃️ Maintains data copy from primary

2. 🔄 Asynchronous replication

📒 Uses transaction logs

🔄 Keeps replicas synchronized with primary

3. 📖 Application read access

📖 Read from any node in cluster

4. ✍️ Application write access

✍️ Write only to primary nodes

5. 🔪 Read scalability

📖 Enhanced by read replicas

6. 🗃️ Data durability

📒 Stored in durable transaction logs

🔒 No risk of data loss

7. 🍰 Data partitioning

🗃️ Across shards in cluster

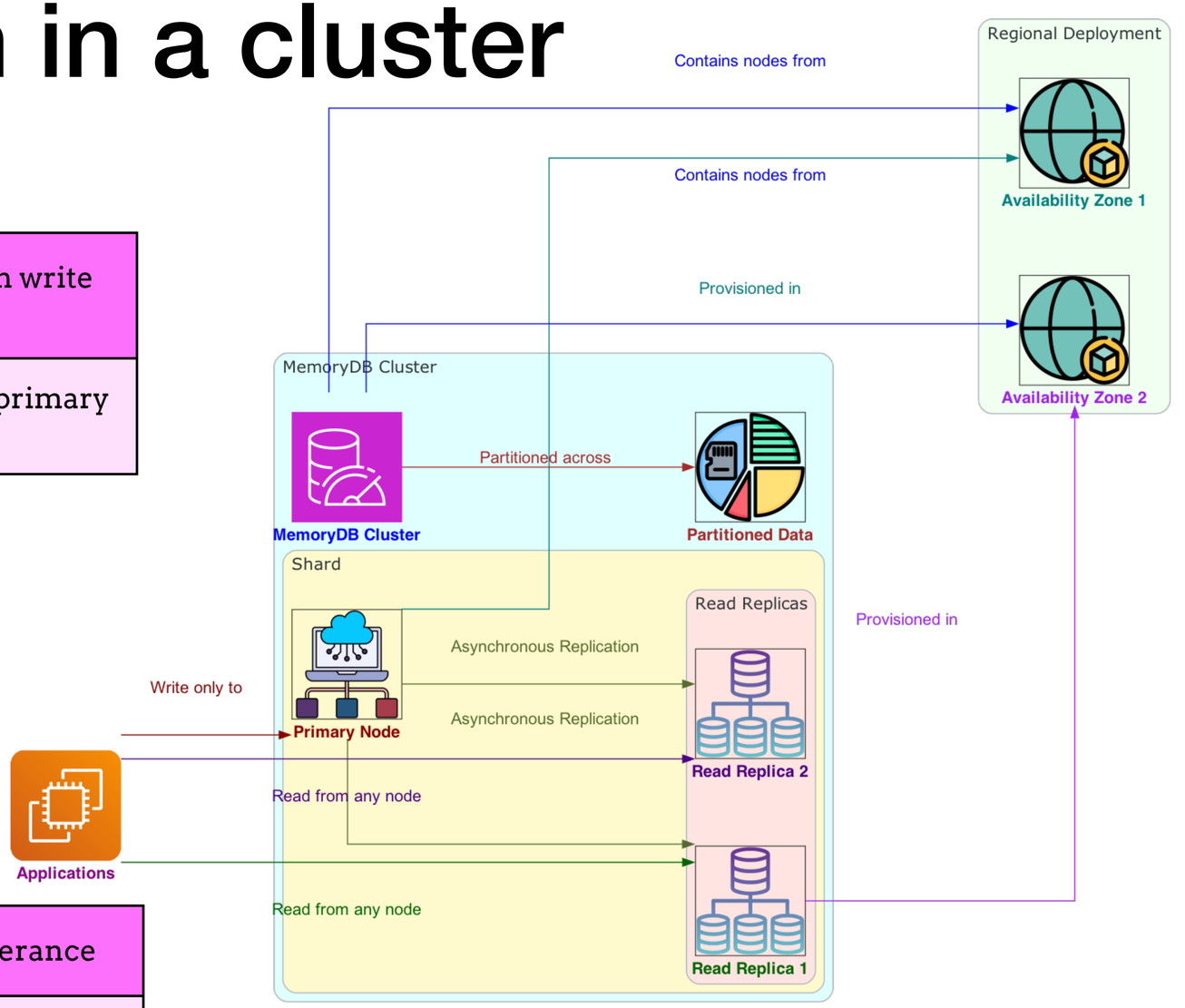
8. 🌐 Regional deployment

🗺️ Clusters contain nodes from one region

9. 🤘 Fault tolerance

🖥️ Provision primaries, replicas across AZs

🌍 Within the same region



Thanks for Watching