DevOps

[ElastiCache 2](#_Toc193689057)

[Overview 2](#_Toc193689058)

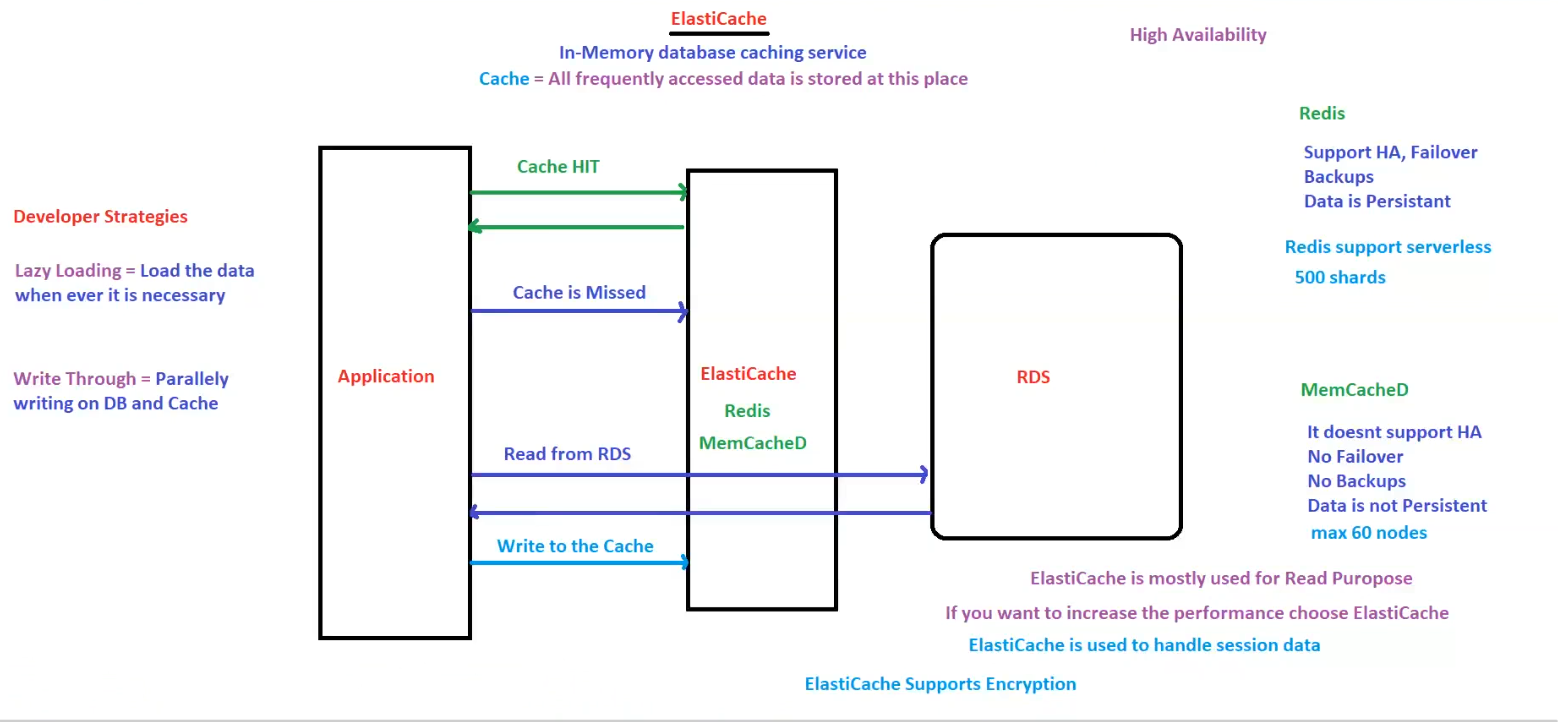
[AWS ElastiCache - Redis Cluster Mode 3](#_Toc193689059)

[Redis Cluster Setup with EC2 5](#_Toc193689060)

[AWS RedShift 7](#_Toc193689061)

# ElastiCache

## Overview



**AWS ElastiCache for Performance Optimization**

**ElastiCache** is an **in-memory database caching service** that stores **frequently accessed data** to **reduce latency and improve application performance**.

**1. Cache Workflow in the Application**

1. **Cache HIT**:
   * If the requested data is in **ElastiCache**, it is **returned immediately** to the application.
2. **Cache MISS**:
   * If the data is **not found** in ElastiCache:
     + The application **fetches the data from RDS**.
     + The data is **written back to the cache** for future requests.

**2. Caching Strategies**

**A. Lazy Loading**

* Data is **loaded into the cache only when requested**.
* Reduces **unnecessary data caching**, saving memory.
* **Downside**: **First request** results in a **cache miss**, causing a slight delay.

**B. Write-Through**

* Data is **simultaneously written to both the cache and the database**.
* Ensures that **cache always has updated data**.
* **Downside**: Higher **write latency** due to extra caching step.

**3. ElastiCache Options**

**A. Redis**

✅ Supports **High Availability (HA), Failover, and Backups**.  
✅ Data is **persistent**.  
✅ Supports **serverless mode** with up to **500 shards**.

**B. Memcached**

❌ **No High Availability**.  
❌ **No Failover or Backups**.  
❌ **Data is not persistent**.  
✅ **Supports up to 60 nodes**.

**4. Key Features of ElastiCache**

* **Used mainly for Read Optimization**.
* **Boosts performance** by reducing direct database access.
* **Supports Encryption** for secure data storage.
* **Handles session data efficiently**.

**5. When to Use ElastiCache?**

✅ If **application performance** needs improvement.  
✅ If **frequent database reads** slow down operations.  
✅ If **session data** needs to be cached for quick access.

**Conclusion**

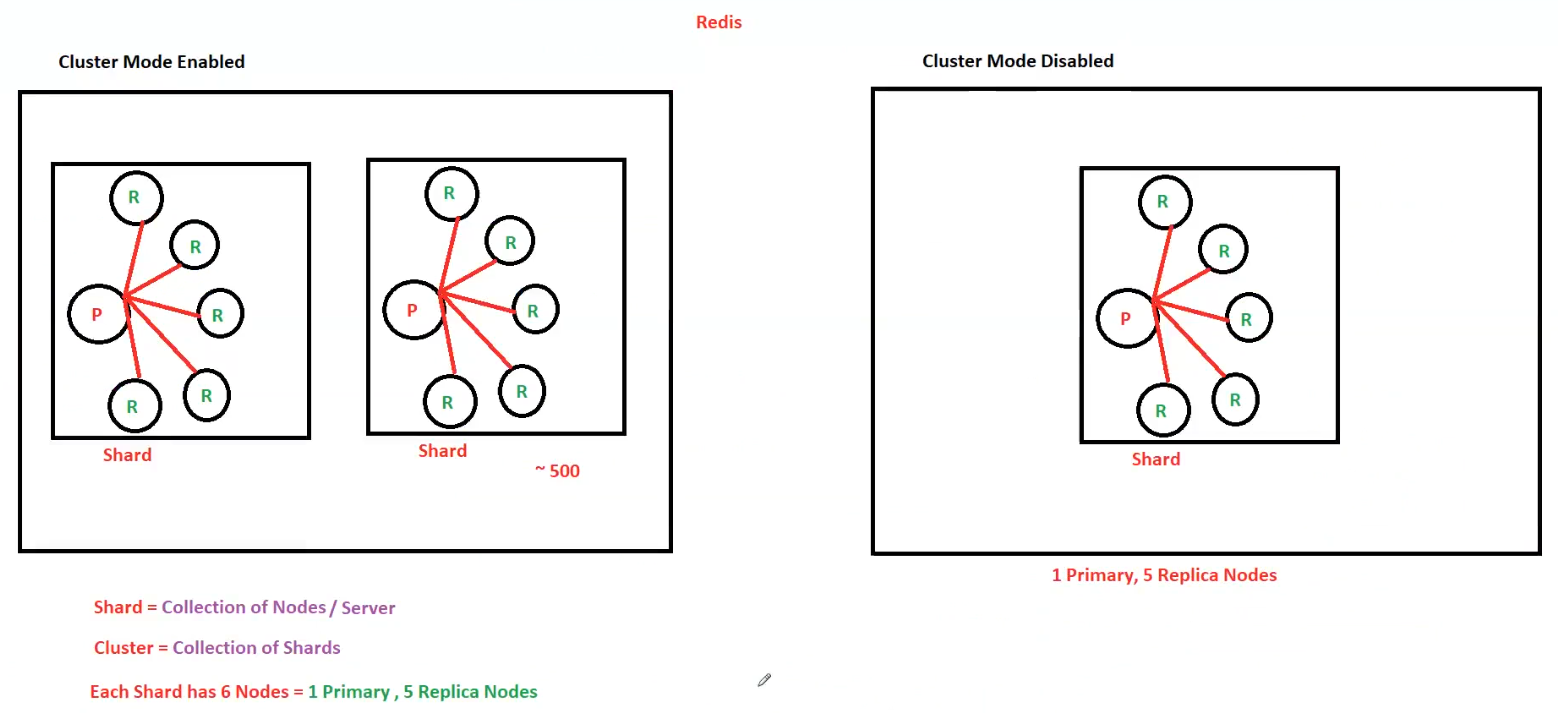
AWS ElastiCache (Redis or Memcached) is **a powerful caching solution** that reduces **database load**, improves **application speed**, and ensures **scalability**.

## AWS ElastiCache - Redis Cluster Mode

**AWS ElastiCache - Redis Cluster Mode Enabled vs Disabled**

Amazon **ElastiCache for Redis** supports two deployment modes:

1. **Cluster Mode Enabled** (Sharded Architecture)
2. **Cluster Mode Disabled** (Single Shard Architecture)



**1. Key Concepts**

* **Shard** = A collection of **nodes/servers** (data partitioning unit).
* **Cluster** = A collection of **multiple shards**.
* **Each Shard Contains**:
  + **1 Primary Node (P)**.
  + **5 Replica Nodes (R)** (for redundancy and high availability).

**2. Cluster Mode Enabled (Sharded Architecture)**

✅ **Supports horizontal scaling** (up to **500 shards**).  
✅ **Data is partitioned** across multiple shards.  
✅ **Read and write operations are distributed** among the shards.  
✅ **Better performance and high availability**.

**Best for:** Large-scale applications that require **high scalability and fault tolerance**.

**3. Cluster Mode Disabled (Single Shard Architecture)**

❌ **No horizontal scaling** (single shard only).  
❌ **All data is stored in a single shard**.  
✅ **Still supports replication** (1 Primary, 5 Replicas).  
✅ **Easier to set up and manage**.

**Best for:** Small to medium applications that **don’t require large-scale partitioning**.

**4. When to Use Each Mode?**

| **Feature** | **Cluster Mode Enabled** | **Cluster Mode Disabled** |
| --- | --- | --- |
| **Scalability** | ✅ Supports up to 500 shards | ❌ Limited to a single shard |
| **Data Partitioning** | ✅ Distributed across shards | ❌ All data in one shard |
| **High Availability** | ✅ Multiple replicas per shard | ✅ Multiple replicas but single shard |
| **Performance** | ✅ Better for high throughput applications | ❌ Limited performance |
| **Use Case** | Large-scale, high-demand applications | Small to medium applications |

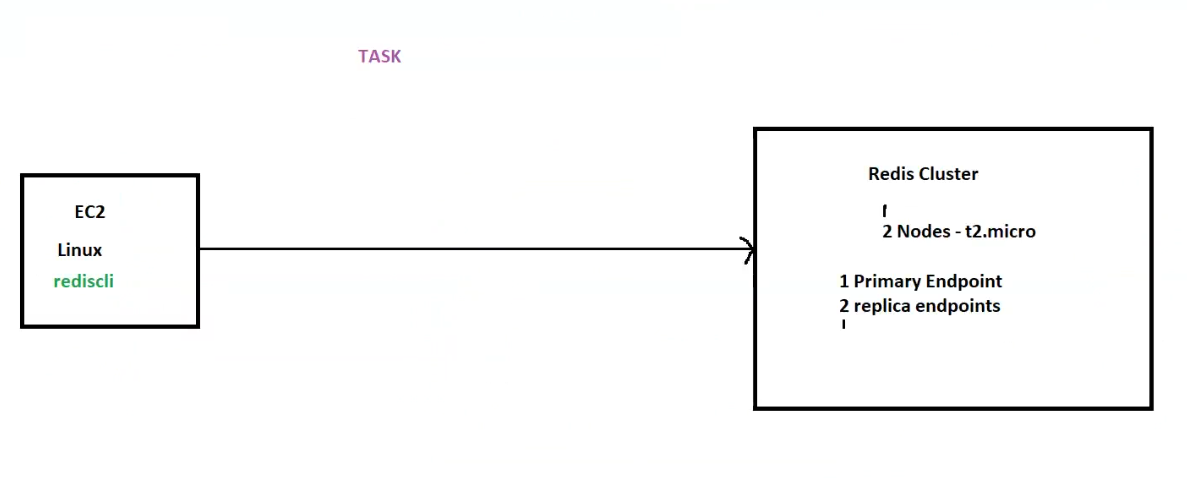
**5. Summary & Key Takeaways**

✅ **Cluster Mode Enabled** = Better **scalability, performance, and high availability**.  
✅ **Cluster Mode Disabled** = Easier setup but **limited to a single shard**.  
✅ **Each shard has 1 Primary + 5 Replicas** for **fault tolerance**.

**Conclusion**

Choosing between **Cluster Mode Enabled vs Disabled** depends on your **application’s scalability needs**.  
For **high-performance, large-scale applications**, **Cluster Mode Enabled is recommended**.

## Redis Cluster Setup with EC2



Here are the structured notes based on the provided diagram:

**AWS Redis Cluster Setup with EC2**

This architecture illustrates an **EC2 instance connecting to an AWS ElastiCache Redis Cluster** using redis-cli.

**1. Components**

**A. EC2 Instance**

* **OS**: Linux
* **Installed Tool**: redis-cli (Redis command-line interface)
* **Purpose**: Used to connect to the **Redis Cluster**.

**B. Redis Cluster (ElastiCache)**

* **Number of Nodes**: **2 Nodes** (t2.micro)
* **Endpoints**:
  + **1 Primary Endpoint** (For writes & reads)
  + **2 Replica Endpoints** (For read operations)

**2. Connection Process**

**A. Install redis-cli on EC2**

If redis-cli is not installed, install it using:

sudo yum install -y redis

or for Ubuntu:

sudo apt install -y redis-tools

**B. Connect to Redis Cluster**

redis-cli -h <Primary\_Endpoint> -p 6379

* Replace <Primary\_Endpoint> with your **Redis primary endpoint**.

**C. Verify Connection**

Run:

ping

If the response is:

PONG

It confirms that the **Redis cluster is reachable**.

**D. Store and Retrieve Data**

set test\_key "Hello Redis"

get test\_key

Expected output:

"Hello Redis"

**3. Key Features of this Redis Cluster**

✅ **Primary-Replica Architecture**: 1 Primary + 2 Read Replicas.  
✅ **High Availability**: Read replicas ensure fault tolerance.  
✅ **Performance Optimization**: Read operations are distributed across replicas.

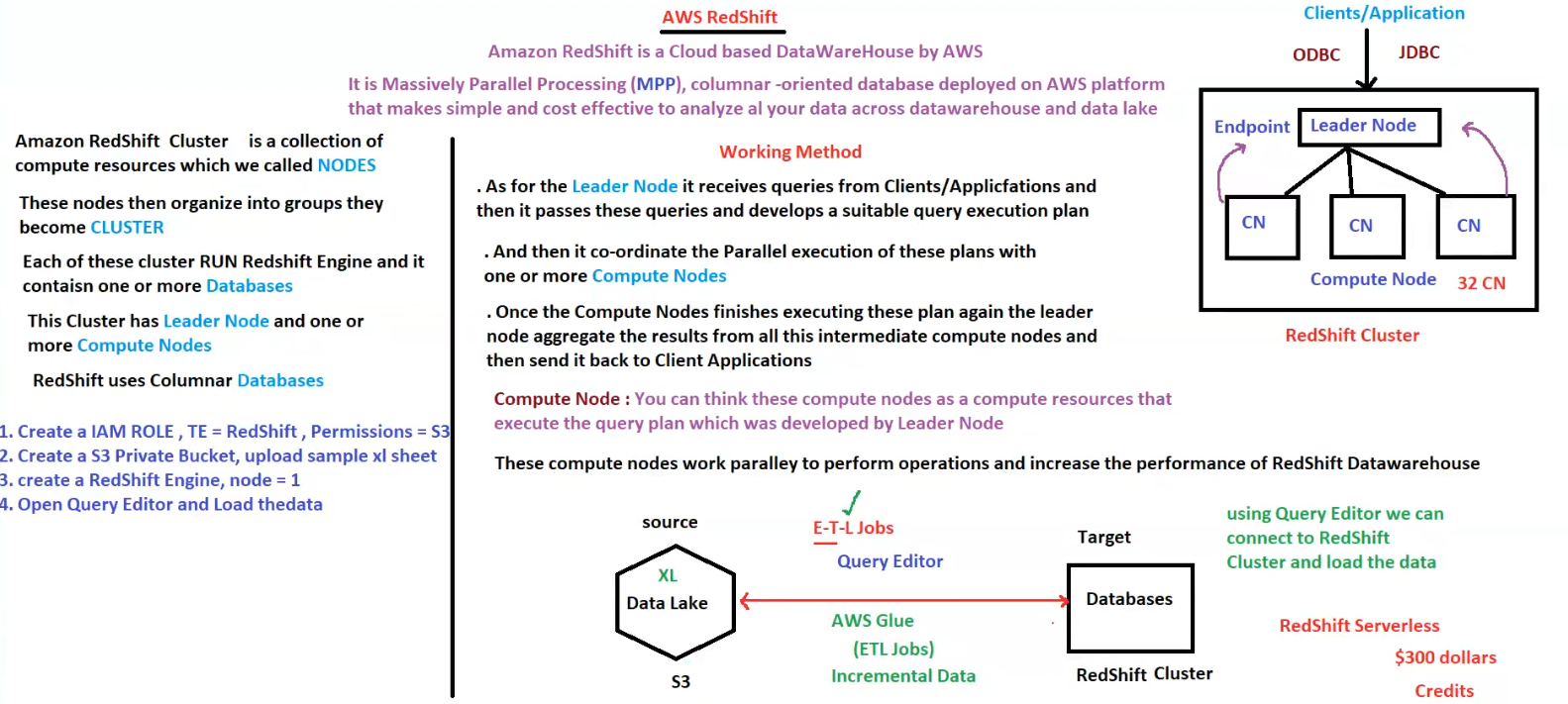
**4. Summary & Takeaways**

* The **EC2 instance** uses redis-cli to interact with **Redis ElastiCache**.
* The **primary endpoint** handles **both reads and writes**.
* The **replica endpoints** handle **only read operations** for scalability.
* **ElastiCache improves performance** by reducing direct database queries.

**Conclusion**

This setup ensures **scalability, high availability, and performance optimization** for applications that require **low-latency caching**.

# AWS RedShift



**AWS RedShift - Cloud Data Warehouse**

**Amazon RedShift** is a **cloud-based, massively parallel processing (MPP), columnar-oriented data warehouse** that enables **efficient data analysis across data warehouses and data lakes**.

**1. RedShift Architecture**

**A. RedShift Cluster**

* A **RedShift Cluster** consists of multiple **compute resources called NODES**.
* Nodes are **grouped together** to form a **CLUSTER**.
* Each cluster **runs the RedShift Engine** and contains **one or more databases**.
* **Cluster Components**:
  + **Leader Node**:
    - Receives queries from clients/applications.
    - Creates a query execution plan and coordinates with **Compute Nodes**.
  + **Compute Nodes (CN)**:
    - Execute the query plan in **parallel** to improve performance.
    - Results are sent back to the **Leader Node** for aggregation.
* **Columnar Database**:
  + RedShift uses **columnar storage**, making it **highly efficient for analytical queries**.

**2. Working Method**

1. **Clients/Applications** connect to the **Leader Node** using **ODBC/JDBC**.
2. **Leader Node** processes the query and sends execution instructions to **Compute Nodes**.
3. **Compute Nodes** process the data in **parallel**, improving performance.
4. **Results are aggregated and sent back to the Client/Application**.

**Key Feature:**  
✅ **Massively Parallel Processing (MPP)** enables RedShift to handle **large-scale data analysis** efficiently.

**3. ETL (Extract, Transform, Load) Process**

* **Source Data**:
  + Data is stored in **S3 Data Lake** in **Excel or other formats**.
* **ETL Jobs**:
  + **AWS Glue** is used for **Extract, Transform, Load (ETL) operations**.
  + **Incremental Data** can be processed into RedShift.
* **Query Editor**:
  + Used to **connect to RedShift Cluster** and load/query data.

**4. Steps to Set Up RedShift**

1. **Create an IAM Role**:
   * Assign **RedShift permissions** for **S3 access**.
2. **Create an S3 Bucket**:
   * Upload sample **Excel files or datasets**.
3. **Launch a RedShift Cluster**:
   * **Choose the number of nodes** (e.g., node = 1 for a small cluster).
4. **Use Query Editor**:
   * **Load the data** and start executing queries.

**5. RedShift Deployment Models**

**A. RedShift Cluster**

* Traditional **provisioned model** with fixed **compute resources**.
* Requires **manual scaling** based on workload.

**B. RedShift Serverless**

* **Fully managed** (No need to provision nodes).
* **Auto-scales** based on demand.
* **Cost-effective** with a **pay-as-you-go** pricing model.
* **$300 Free Credits** available for new users.

**6. Key Features & Benefits**

✅ **Scalable**: Supports **up to 32 Compute Nodes**.  
✅ **Fast Query Performance**: Uses **MPP & Columnar Storage**.  
✅ **Seamless Integration**: Works with **AWS Glue, S3, and ETL tools**.  
✅ **Supports SQL Queries**: Compatible with **BI tools**.

**Conclusion**

AWS RedShift is a **powerful, scalable, and cost-effective data warehouse** designed for **big data analytics**. It provides **fast query performance** using **MPP and columnar storage**.