aws databases

Amazon ElastiCache Deep Dive

Powering modern applications with low latency and high throughput

Michael Labib
Sr. Manager, Non-Relational Databases

Agenda

- Introduction to Amazon ElastiCache
- Redis Topologies & Features
- ElastiCache Use Cases
- Monitoring, Sizing & Best Practices

Introduction to Amazon ElastiCache

aws databases

Purpose-built databases

Relational	Key-Value	Document	In-Memory	Graph	Time-series	Ledger	Wide Column
Amazon Aurora Amazon Relational Database Service (RDS)	Amazon DynamoDB	Amazon DocumentDB	Amazon ElastiCache	Amazon Neptune	Amazon Timestream	Amazon QLDB	Amazon Keyspaces (for Apache Cassandra)

Purpose-built databases

Relational	Key-Value	Document	In-Memory	Graph	Time-series	Ledger	Wide Column
Aurora RDS	DynamoDB	DocumentDB	ElastiCache	Neptune	Timestream	QLDB	Keyspaces (for Apache Cassandra)

Modern real-time applications require

Performance, Scale & Availability

- E-Commerce
- Media streaming
- Social media
- Online gaming
- Shared economy

Users	1M+
Data volume	Terabytes—petabytes
Locality	Global
Performance	Microsecond latency
Request rate	Millions per second
Access	Mobile, IoT, devices
Scale	Up-out-in
Economics	Pay-as-you-go
Developer access	Open API

aws databases

Amazon ElastiCache – Fully Managed Service

Redis & Memcached compatible

Fully compatible with open source Redis and Memcached

Extreme performance

In-memory data store and cache for microsecond response times

Secure and reliable

Network isolation, encryption at rest/transit, HIPAA, PCI, FedRAMP, multi AZ, and automatic failover

Easily scales to massive workloads

Scale writes and reads with sharding and replicas

What is Redis?

Initially released in 2009, Redis provides:

- Complex data structures: Strings, Lists, Sets, Sorted Sets, Hash Maps, HyperLogLog, Geospatial, and Streams
- High-availability through replication
- Scalability through online sharding
- Persistence via snapshot / restore
- Multi-key atomic operations
- LUA scripting
- Open Source

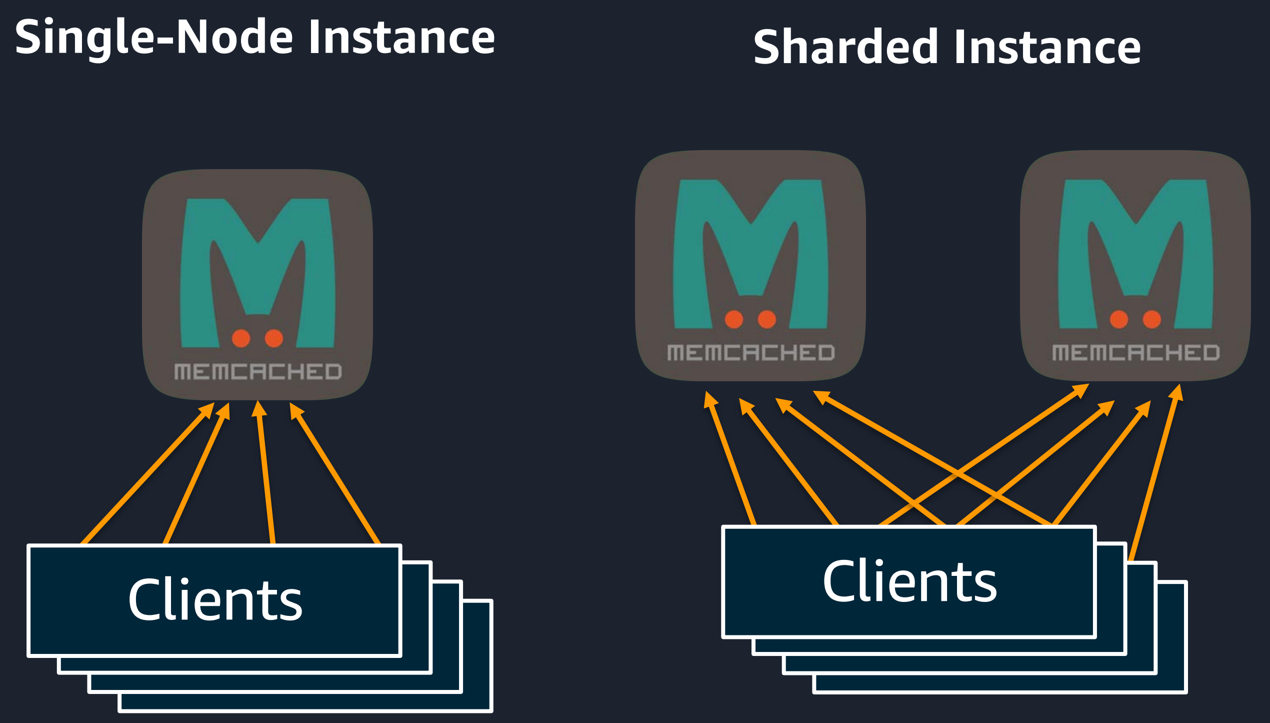
A high-speed, in-memory, non-Relational data store.

Customers love that Redis is easy to use.

What is Memcached?

Initially released in 2003, Memcached provides:

- Simple, in-memory, LRU cache
- Simple key-value (string-string) store
- Supports strings, objects
- Multi-threaded
- Sharding via client-side library
- Easy to Scale
- No persistence
- Open source



The need for speed…

ElastiCache + RDS

ElastiCache + Aurora

ElastiCache + Redshift

ElastiCache + DynamoDB

ElastiCache + DocumentDB

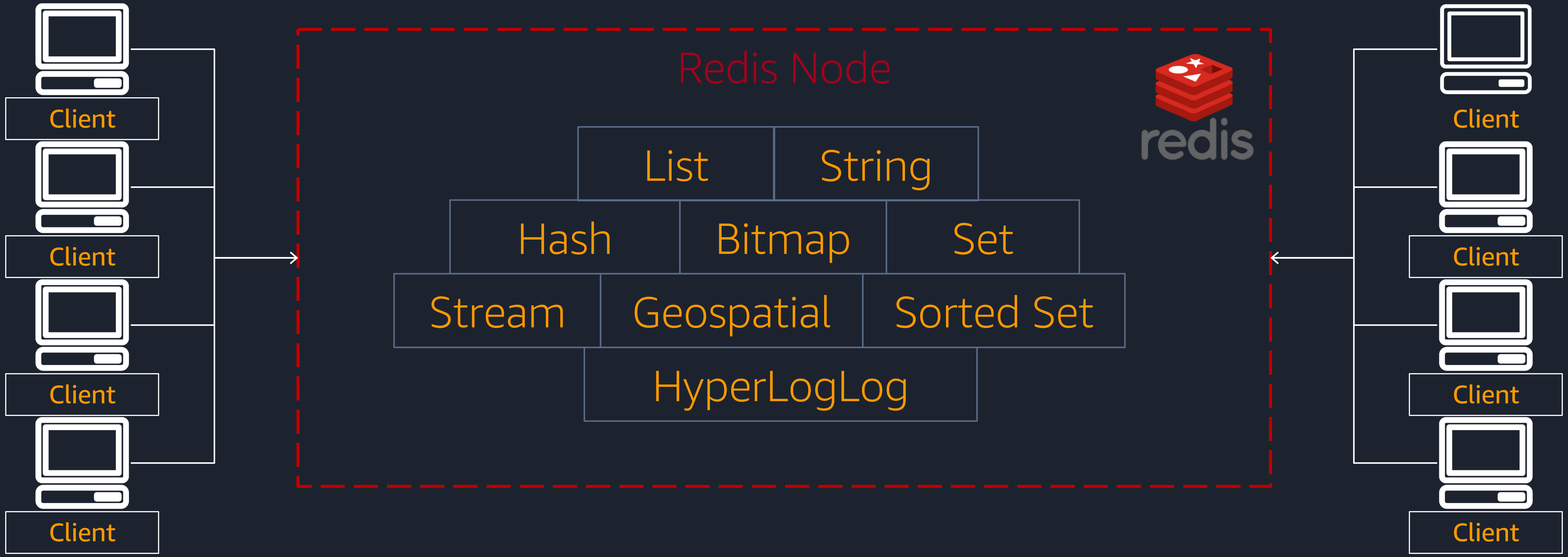
ElastiCache + ….

aws databases

Redis Topologies & Features

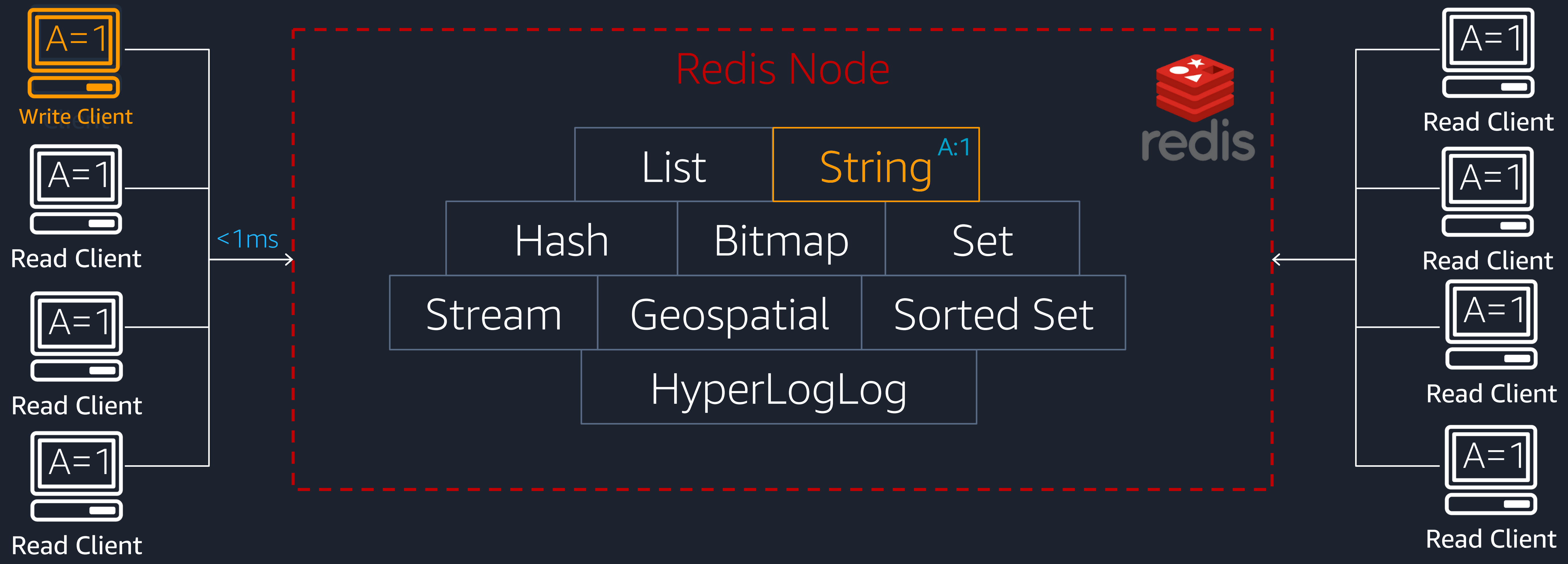
aws databases

ElastiCache Redis: Distributed In-Memory Data Store



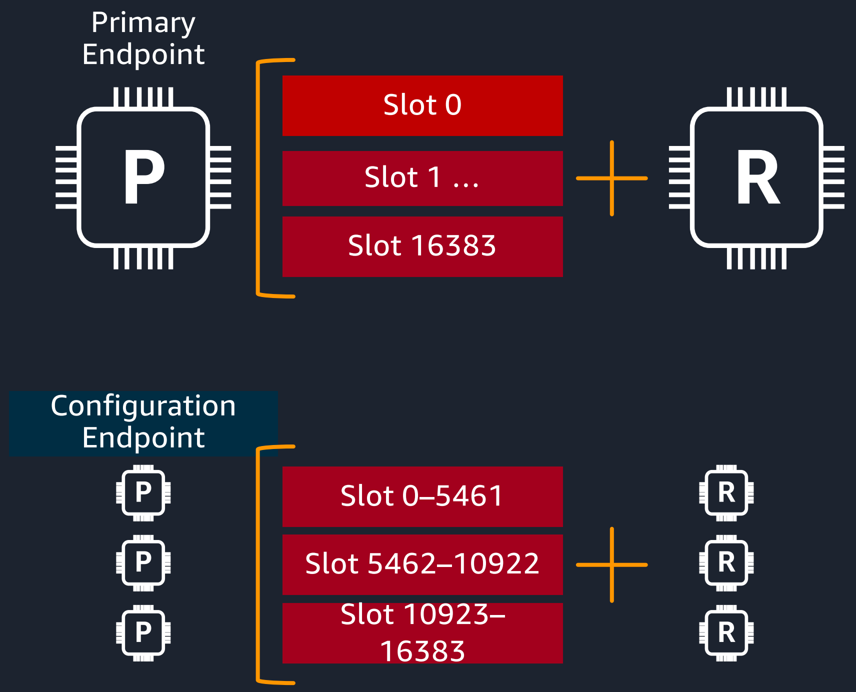
aws databases

ElastiCache Redis: Distributed In-Memory Data Store

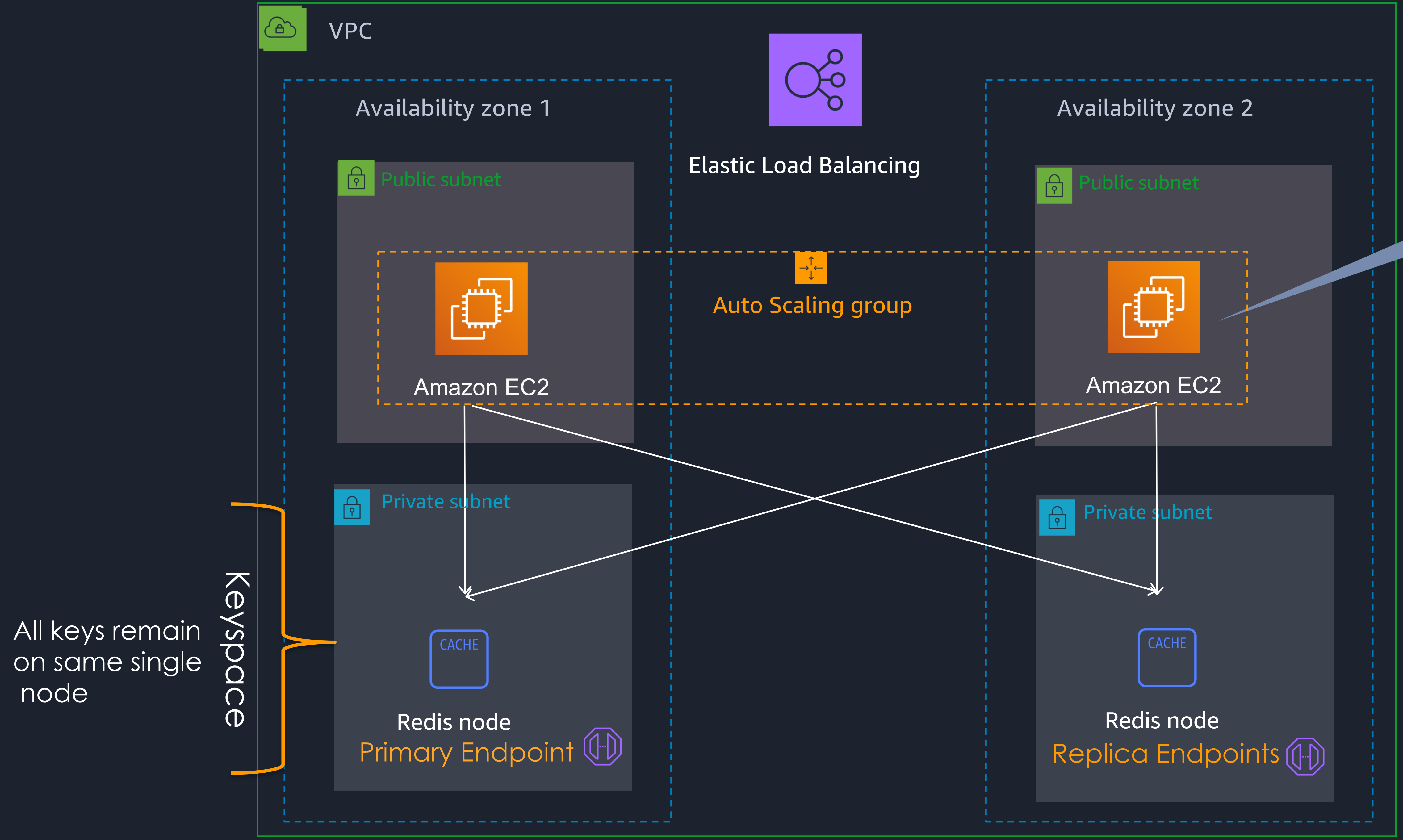


Redis Cluster Mode – Enabled vs. Disabled

Feature	Redis Cluster (enabled)	Redis Cluster (disabled)
Recovery Time	10-20 sec (non-DNS)	~30+ sec (DNS)
Failover Impact	Writes affected on failed shard. Reads available	Writes affected on entire data set. Reads available
Node Scale	Up to 250* nodes (90 = 15 shards + 5 replicas soft limit) 0–5 replicas per shard	1 primary 0-5 replicas (max. 6 nodes)
Storage	170 TB (635 GB x 250)	635 GB
Max Connections	16.25 million (65,000 x 250)	390,000 (65,000 x 6)
Online Scaling	Shards and read replicas	Read replicas only
Migration Path	Backup/Restore Snapshot	Online Migration Tool
Scalability and Performance	• Achieve greater throughput through horizontal scaling • Horizontal/Vertical scaling Supported	• Throughput limited by 1 primary, 5 replicas • Horizontal Scale for Reads (Replicas) supported • Vertical scaling for Replicas/Primary also supported
Scaling Operation	Cluster Resizing (zero-downtime) • Horizontal Scaling to add/remove shards • Read Scalability to add/remove replicas	Vertical Scaling • Writes/Reads continue during scale up operation



Redis Cluster-mode disabled (Scaled Vertically)



Connect to Primary for Read/Writes and Replica's for Reads

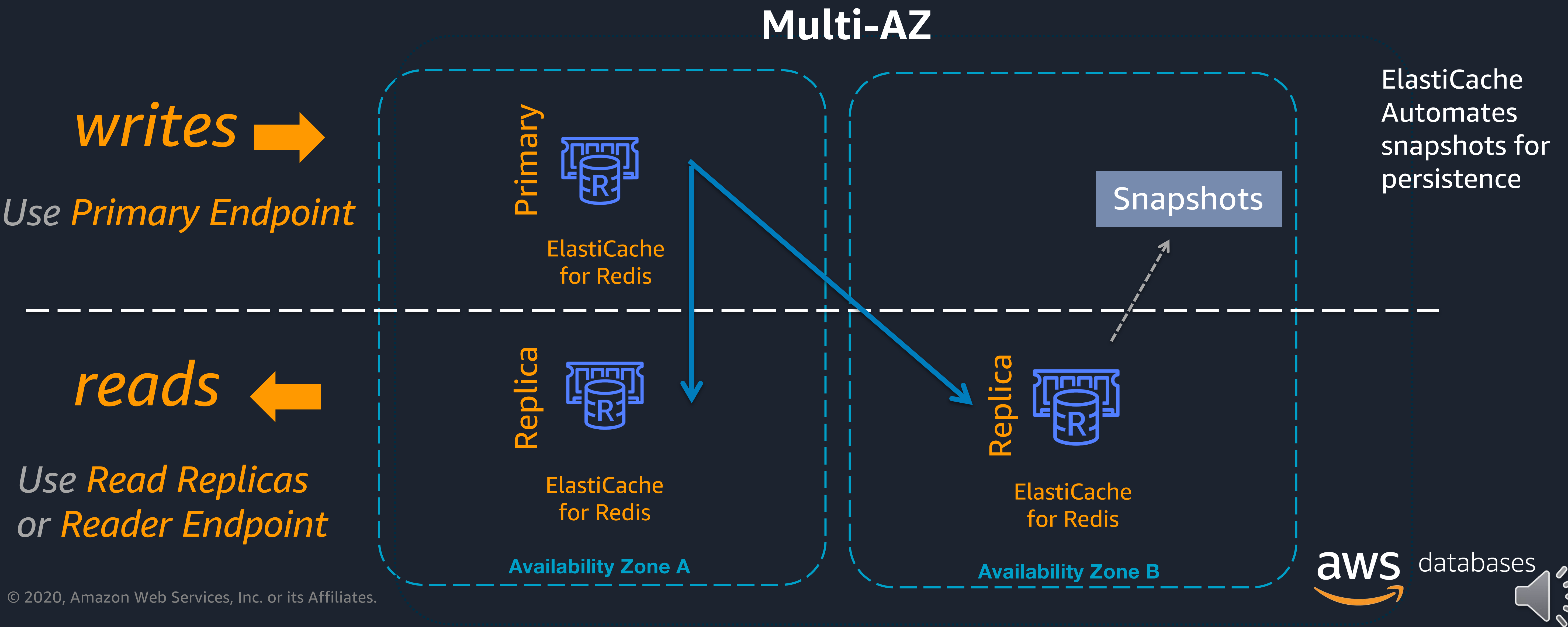
aws databases

ElastiCache for Redis Multi-AZ

Auto-Failover

- Chooses replica with lowest replication lag
- DNS endpoint is same

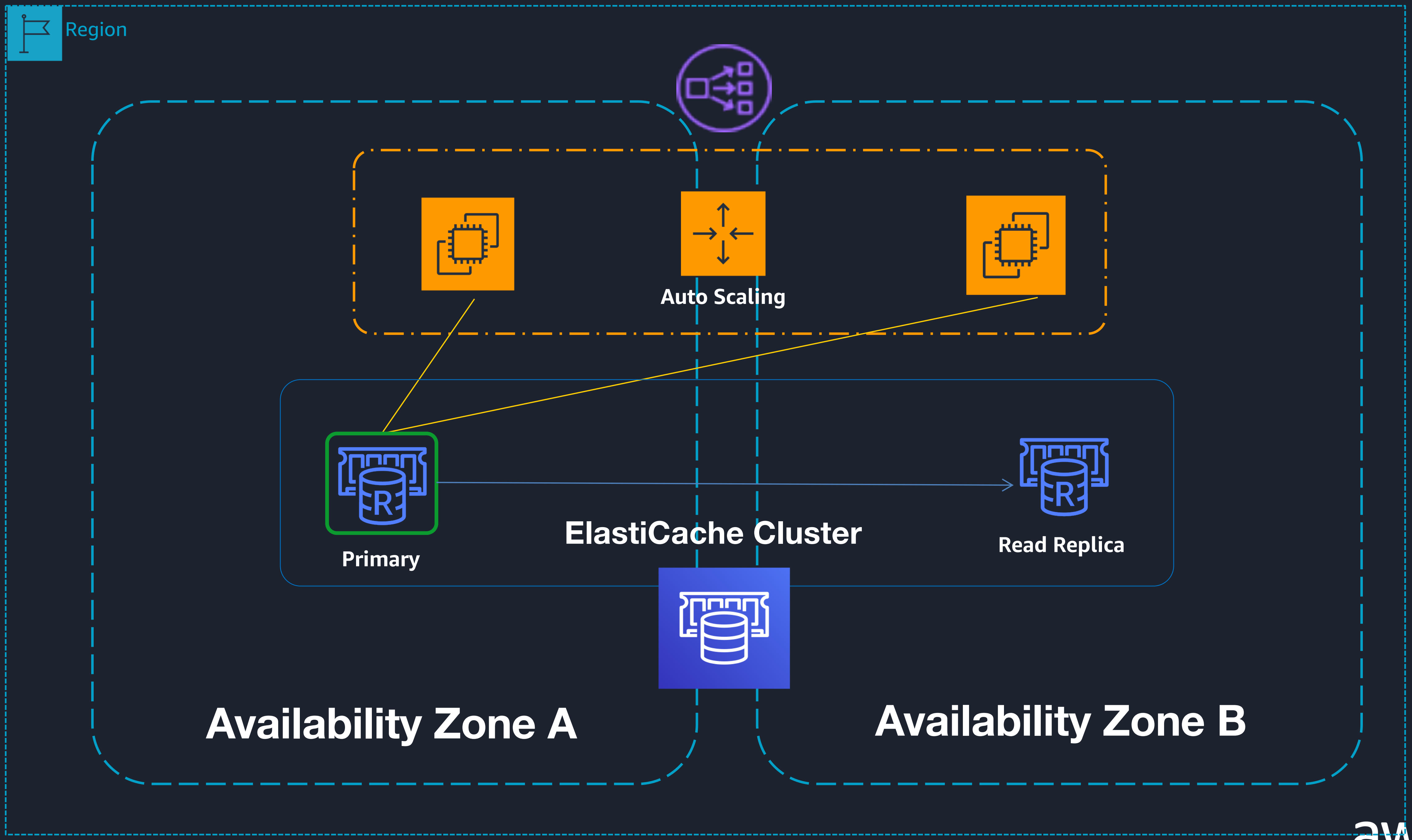
Automatic Failover to a read replica in case of primary node failure



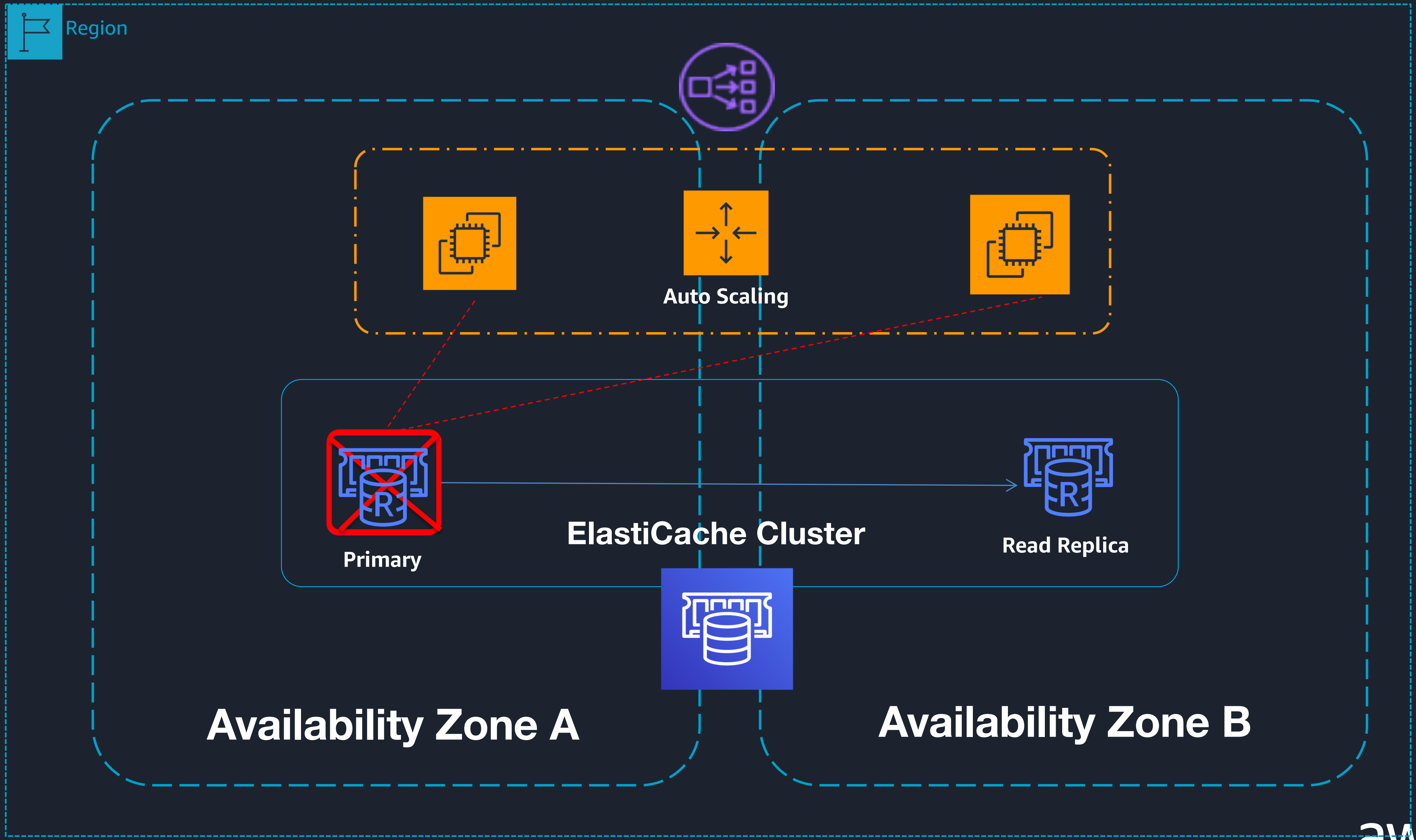
ElastiCache Automates snapshots for persistence

aws databases

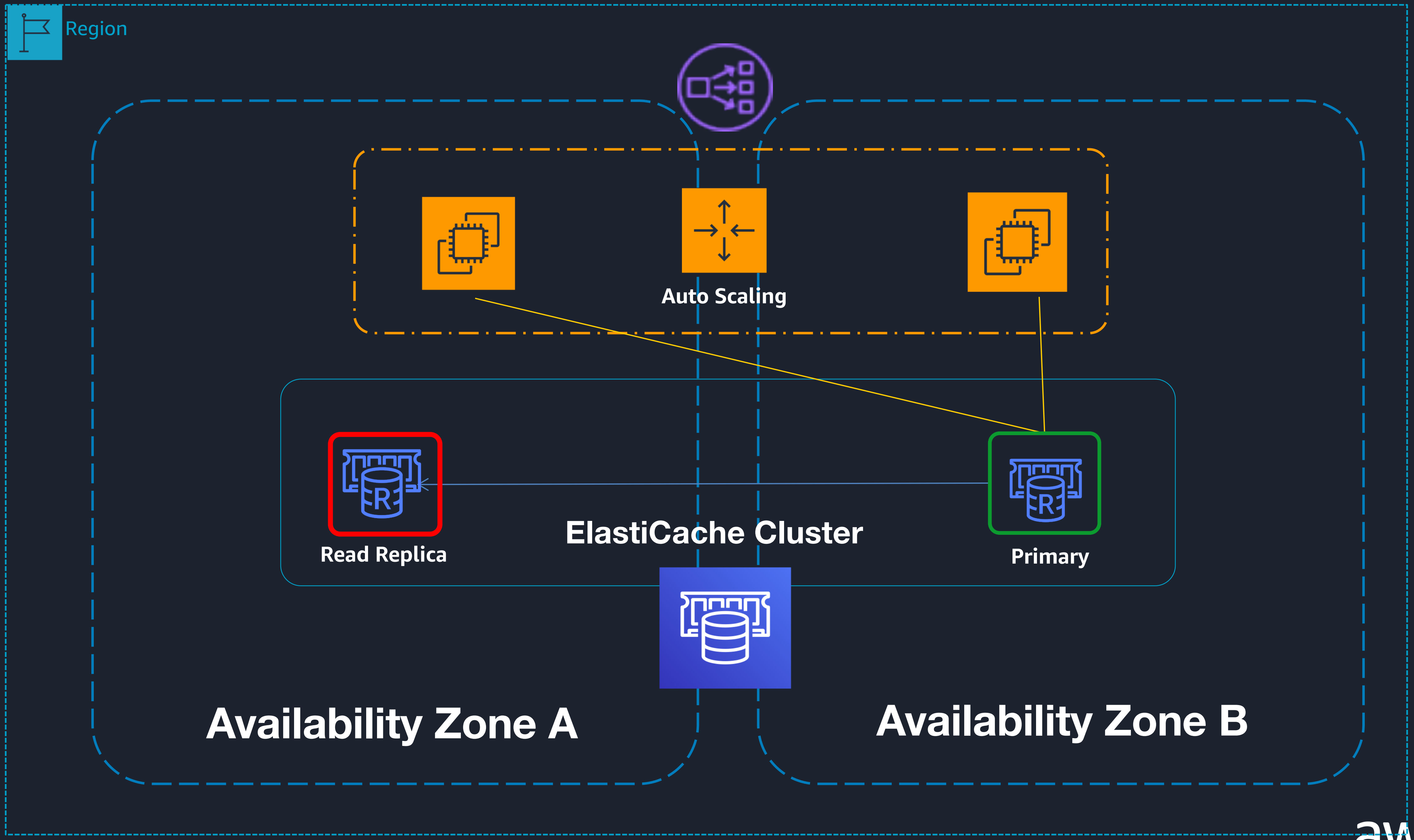
ElastiCache with Redis Multi-AZ



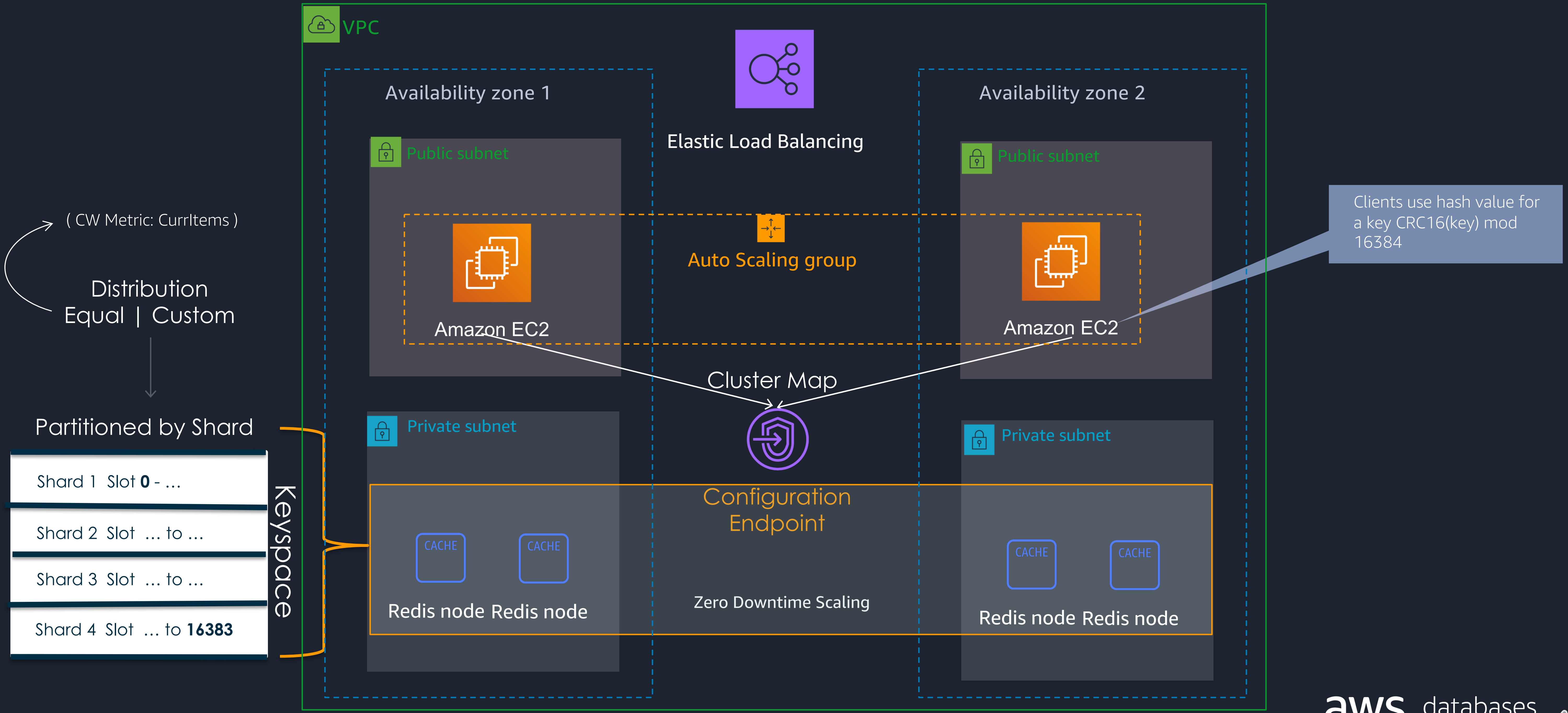
ElastiCache with Redis Multi-AZ



ElastiCache with Redis Multi-AZ



Redis Cluster-mode enabled (Scaled Horizontally)



(CW Metric: CurrItems)

Distribution
Equal | Custom

Partitioned by Shard

Shard 1 Slot **0** - …

Shard 2 Slot … to …

Shard 3 Slot … to …

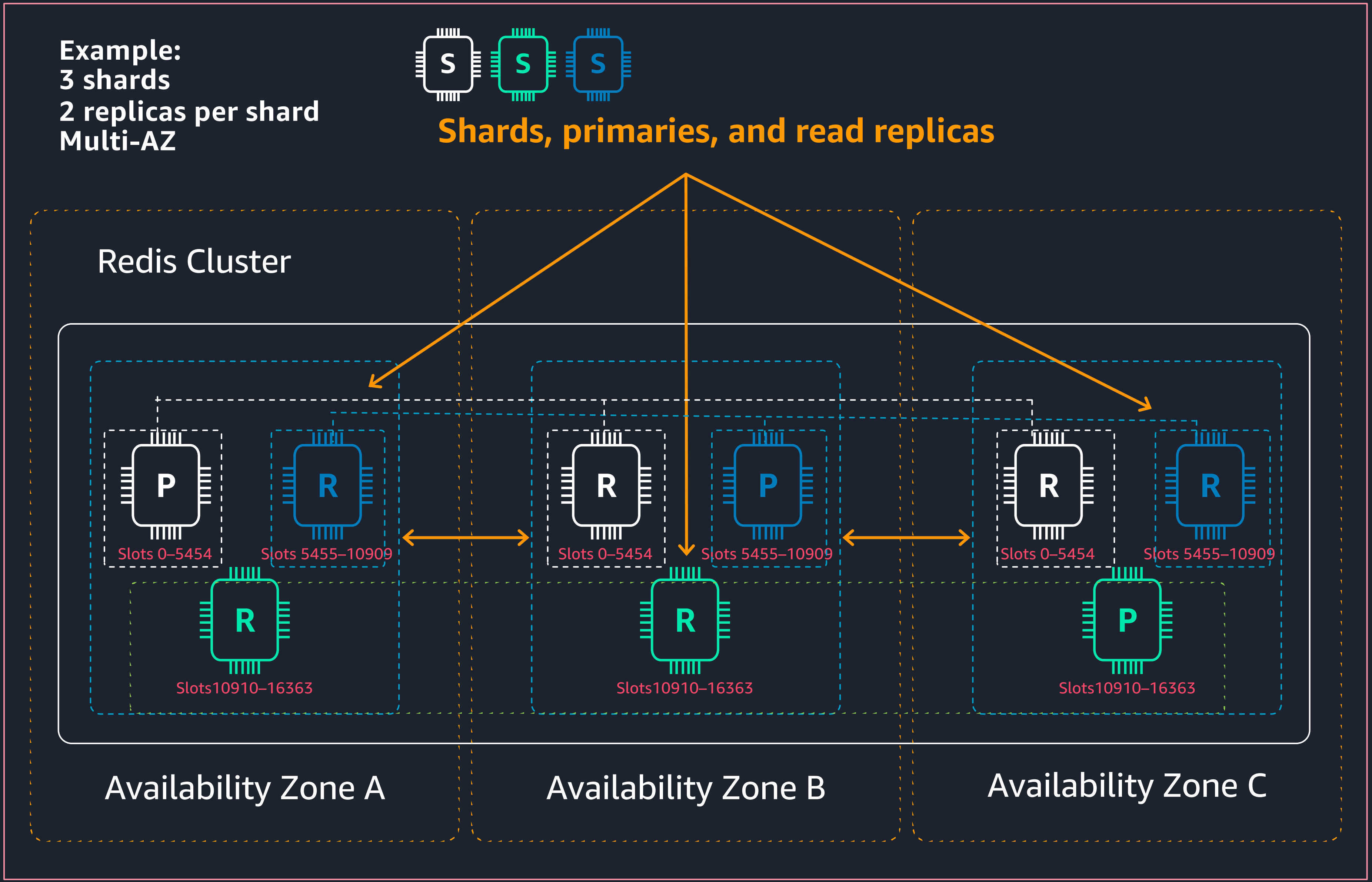
Shard 4 Slot … to **16383**

Keyspace

Clients use hash value for a key CRC16(key) mod 16384

aws databases

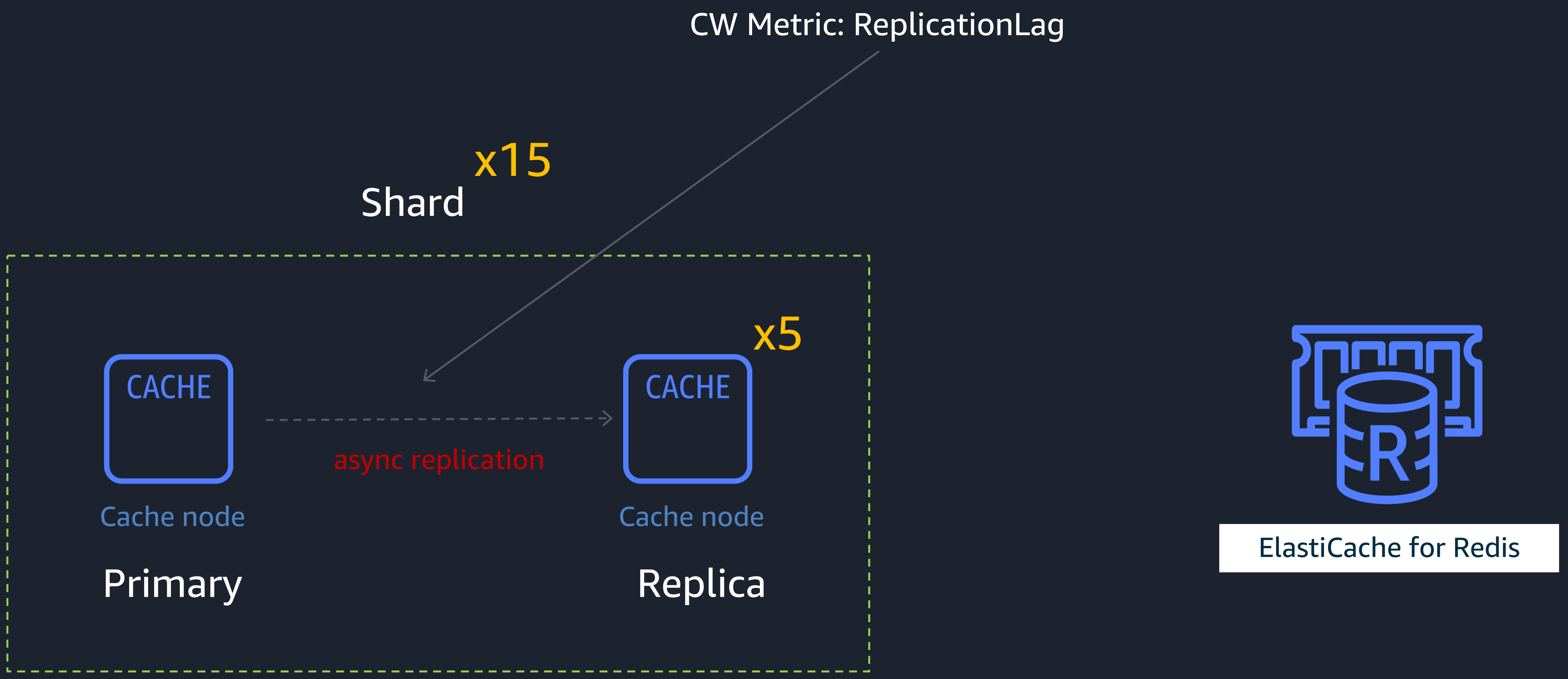
Topology - Redis Cluster Mode Enabled



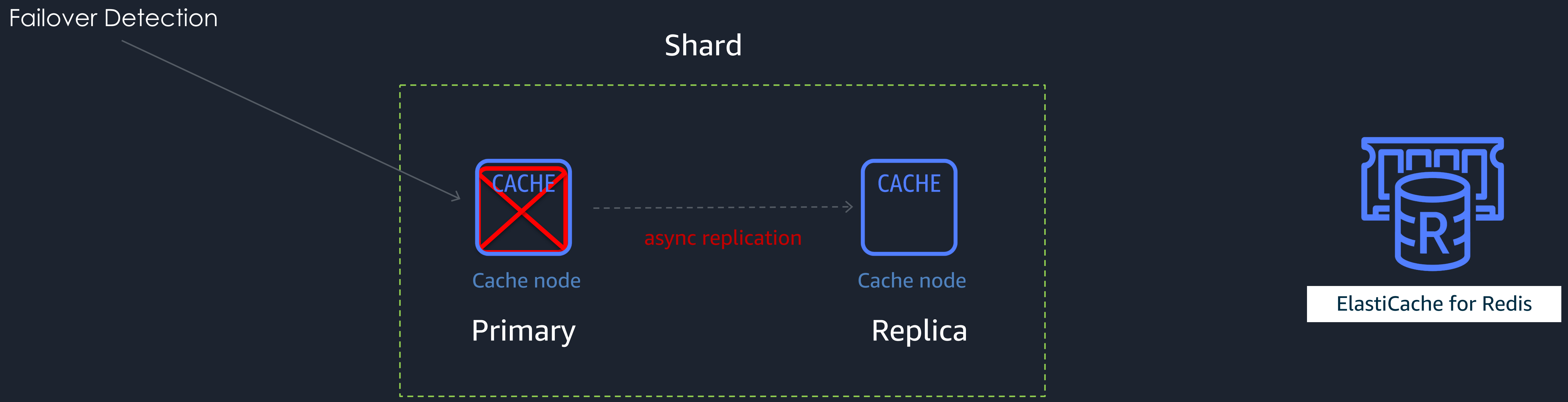
- Add shards to scale reads/writes, increase in-memory capacity
- Add replicas to scale reads, increase availability
- Able to specify availability zones. Multi-AZ default
- Able to customize slot distributions, equal distribution default

aws databases

Cluster mode-enabled Failover

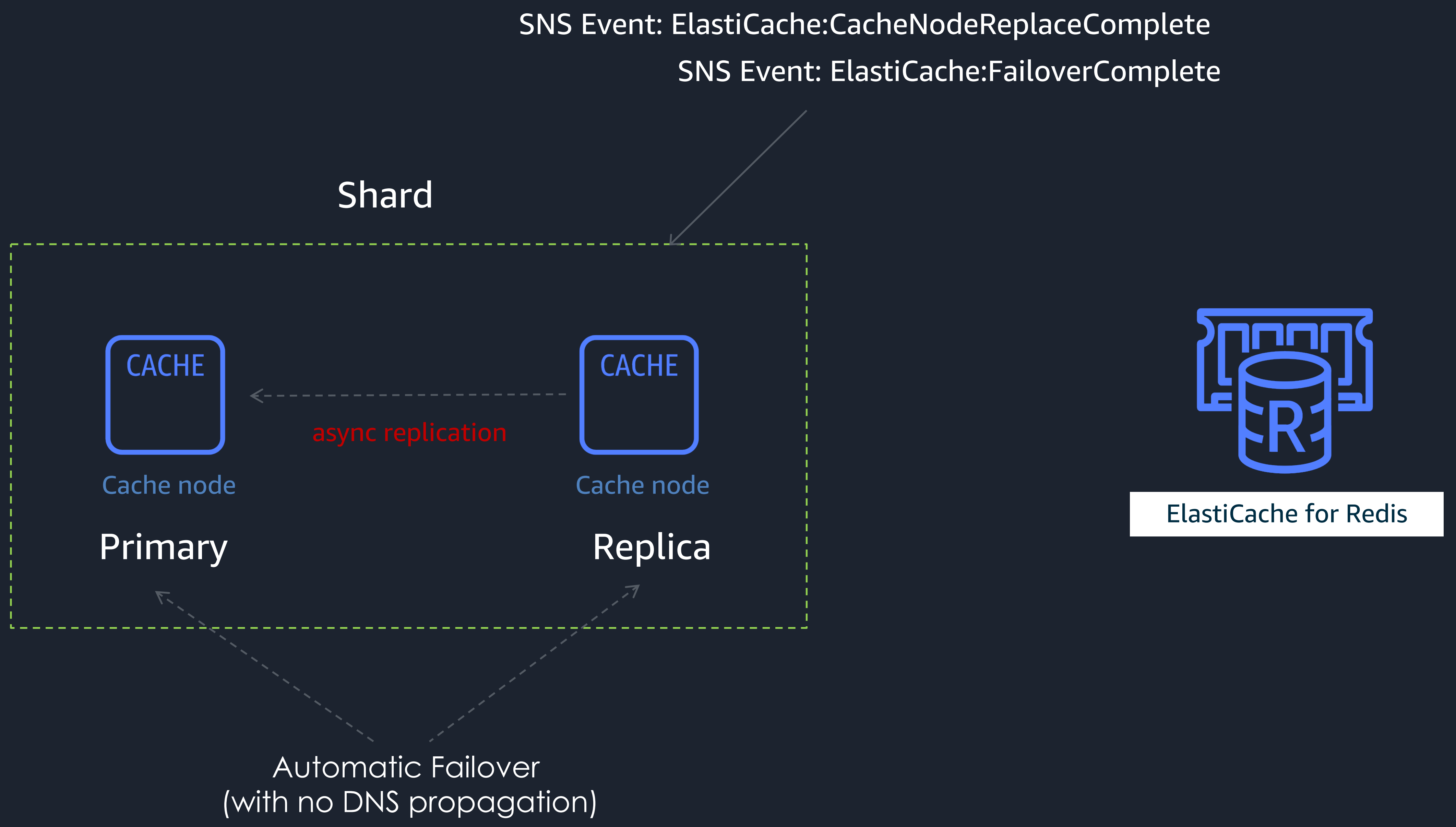


Cluster mode-enabled Failover



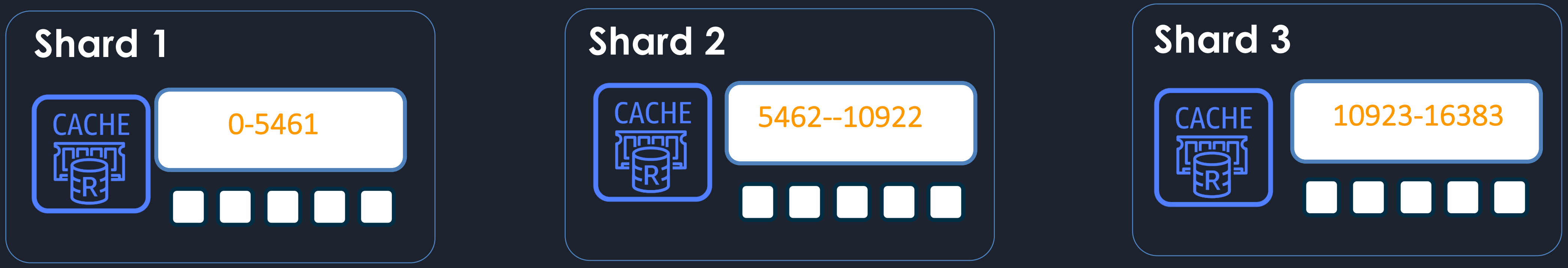
aws databases

Cluster mode-enabled Failover


SNS Event: ElastiCache:CacheNodeReplaceComplete
SNS Event: ElastiCache:FailoverComplete
Shard
CACHE
Cache node
Primary
async replication
CACHE
Cache node
Replica
Automatic Failover
(with no DNS propagation)
ElastiCache for Redis

Test with Failover API

Online Re-Sharding – Zero Downtime

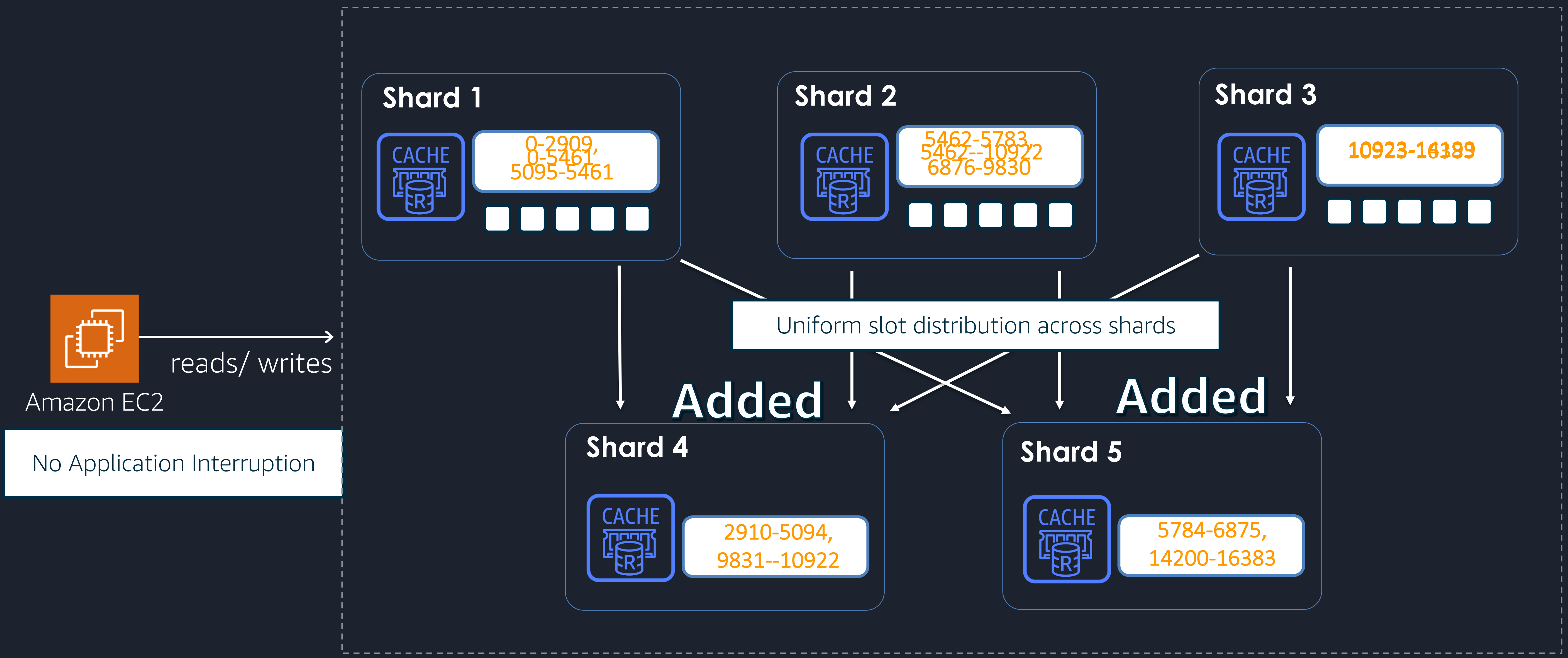


Simple API

```
aws elasticache modify-replication-group-shard-configuration    replication-group-id  rep-group-id
        --apply-immediately  --node-group-count 5
```

Scale In || Out

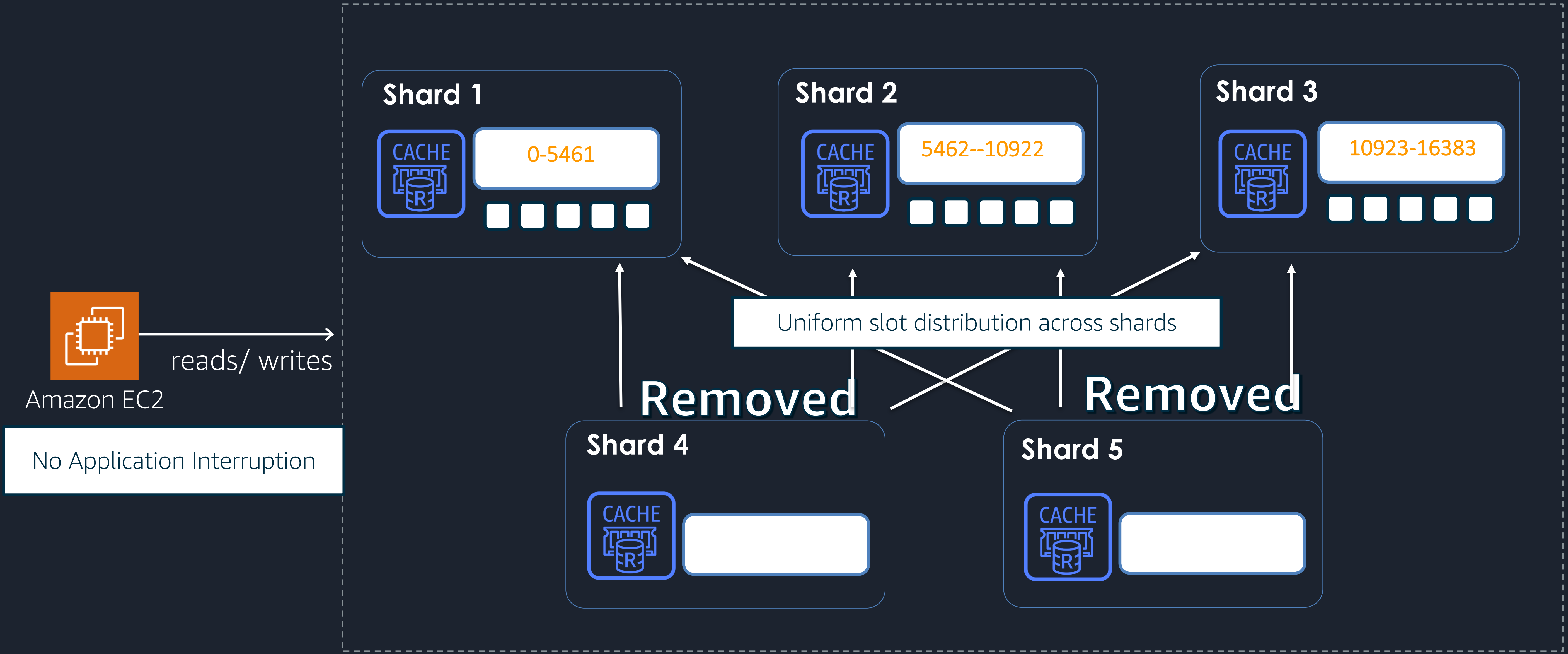
Zero downtime - Online re-sharding - scale out



Uniform slot distribution across shards

No Application Interruption

Zero downtime - Online re-sharding - scale in

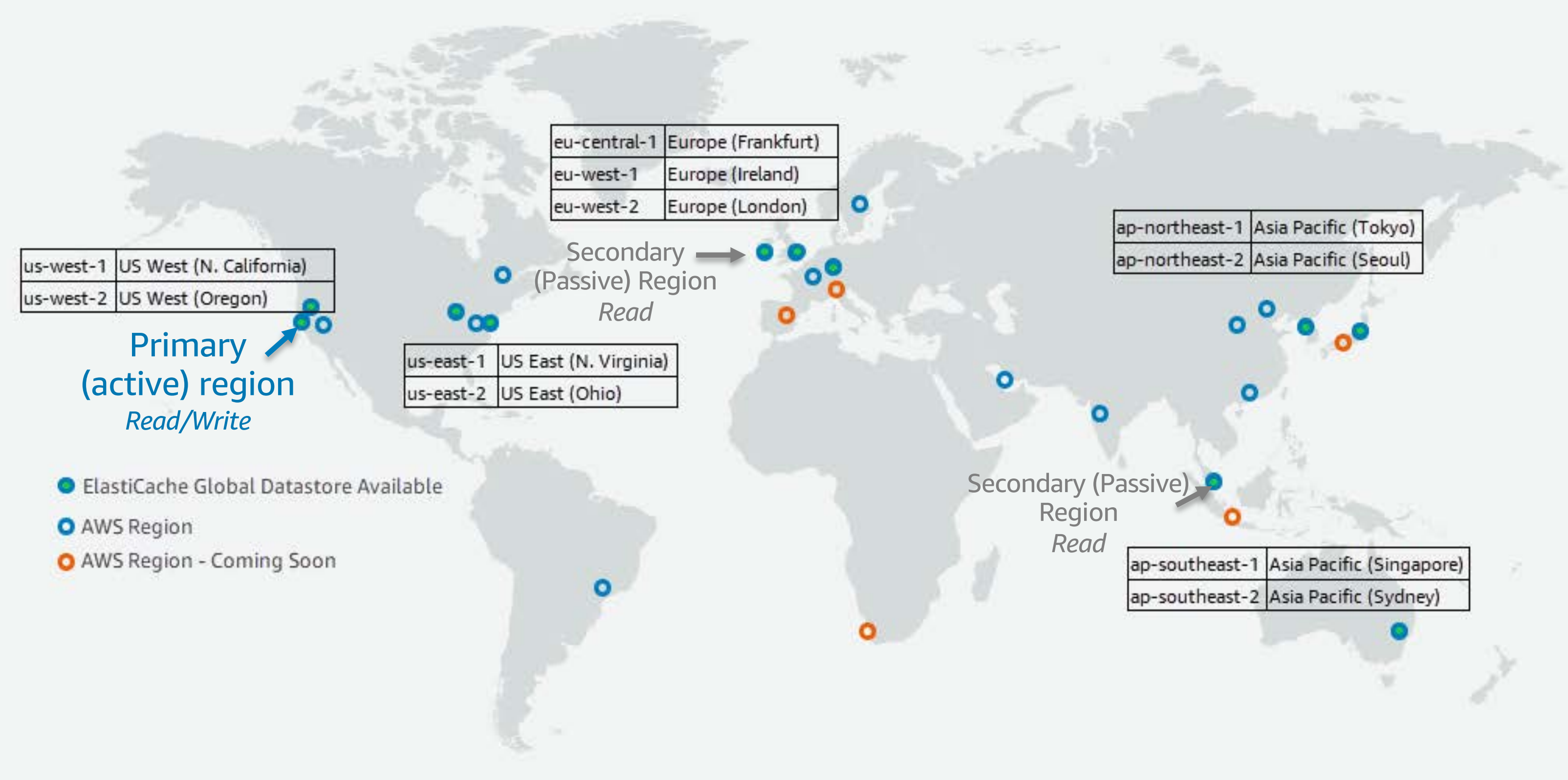


No Application Interruption

Global Datastore (Cross Region Replication)

- One-click setup for existing clusters
- Write locally, read globally
- Enable cross-region disaster recovery
- Leverage extreme performance with Redis' sub-millisecond latency
- Secure encryption in transit for cross-region traffic
- Use with AWS Management Console, or latest AWS SDK or CLI

Example for a worldwide application



us-west-1	US West (N. California)
us-west-2	US West (Oregon)

us-east-1	US East (N. Virginia)
us-east-2	US East (Ohio)

eu-central-1	Europe (Frankfurt)
eu-west-1	Europe (Ireland)
eu-west-2	Europe (London)

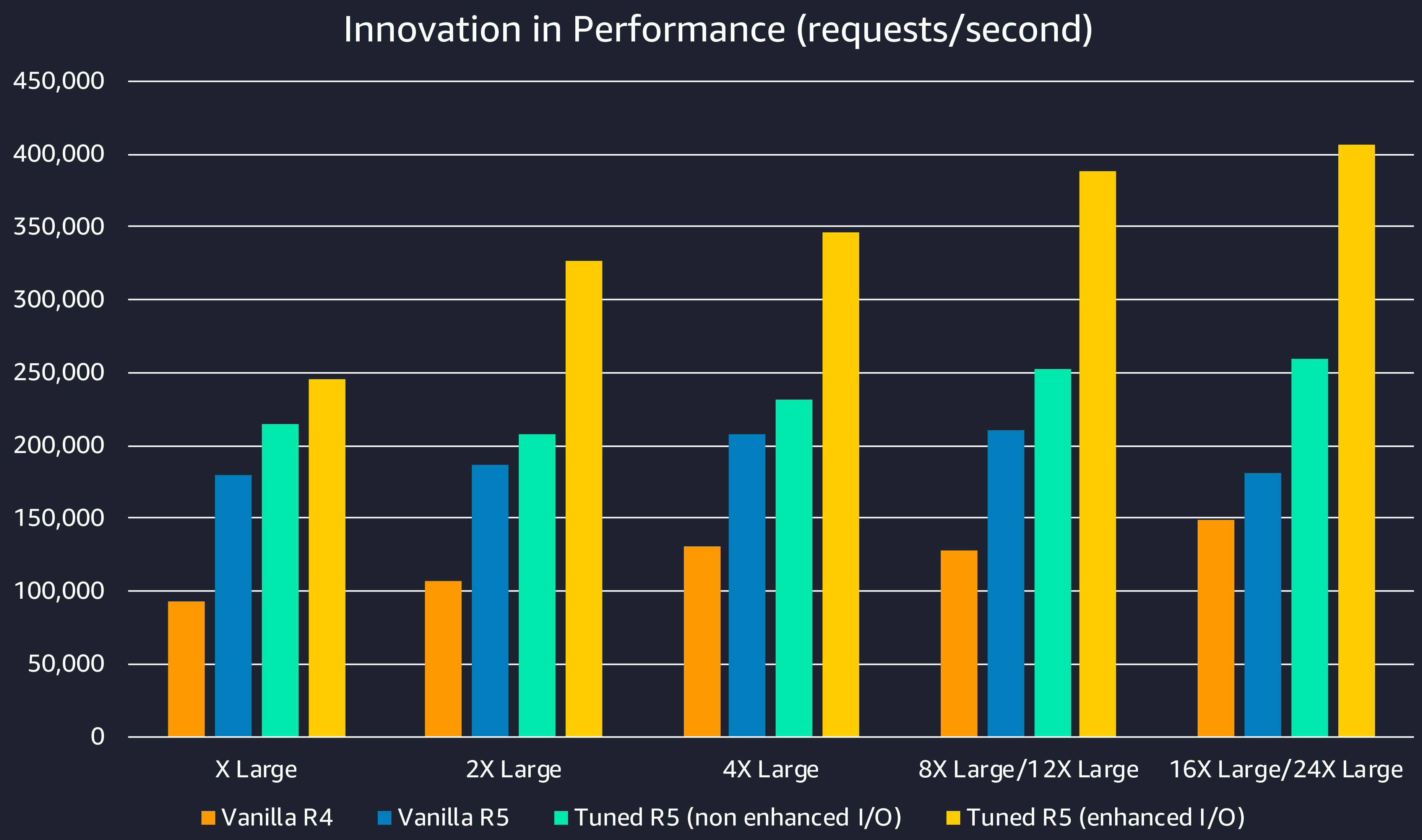
ap-northeast-1	Asia Pacific (Tokyo)
ap-northeast-2	Asia Pacific (Seoul)

ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)

Primary (active) region
Read/Write
Secondary (Passive) Region
Read
Secondary (Passive) Region
Read
● ElastiCache Global Datastore Available
○ AWS Region
○ AWS Region - Coming Soon

aws databases

Optimized M5 and R5 instances & Enhanced I/O



- Scale up to **170 TB** of **in-memory capacity**
- Delivers performance indistinguishable from **bare-metal**
- Dynamic network processing to enhance **I/O**

aws databases

Self-managing Redis is challenging

Difficult to manage

Manage server provisioning, software patching, setup, configuration, and backups

Hard to make highly available

Need to implement fast error detection and remediation

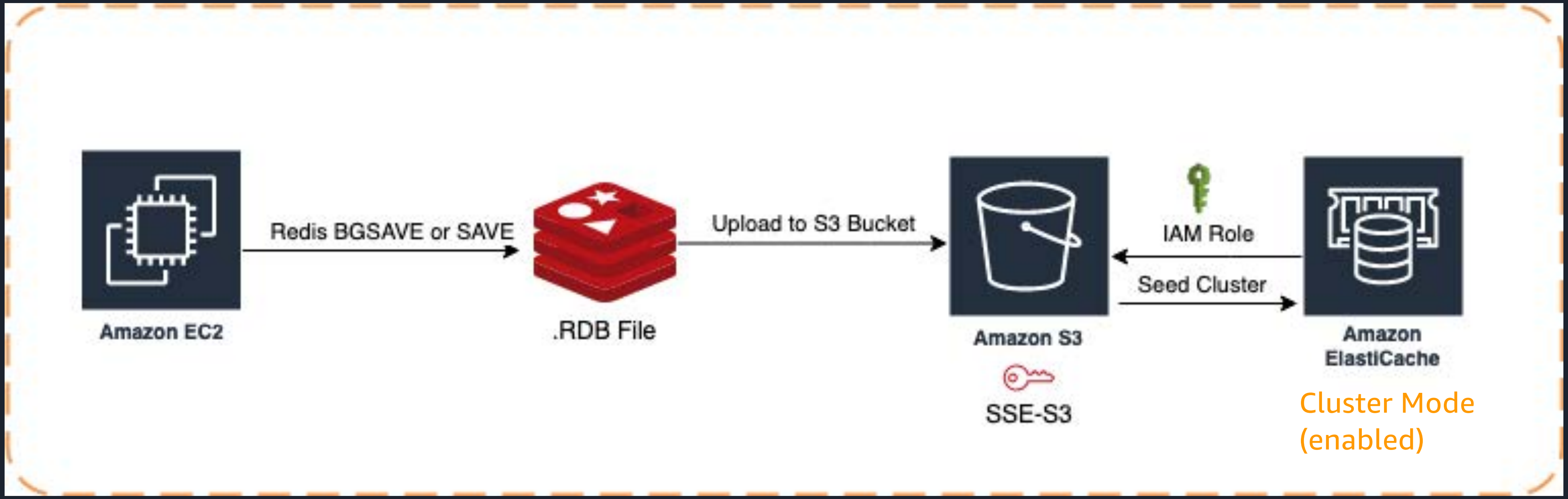
Difficult to scale

Online scaling can be error prone, replication performance needs to be monitored

Expensive

Invest in people, processes, hardware, and software

Migrate using Backup/Restore



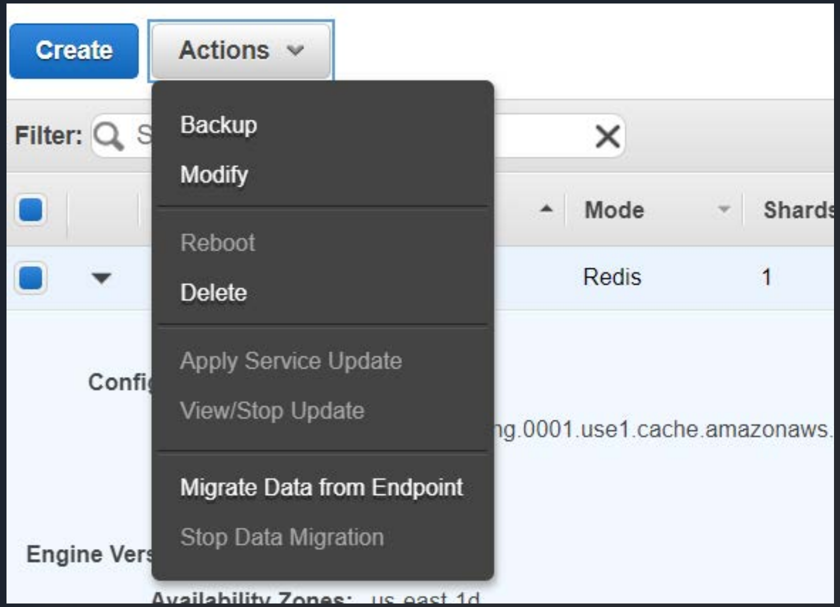
Cluster Mode (enabled)

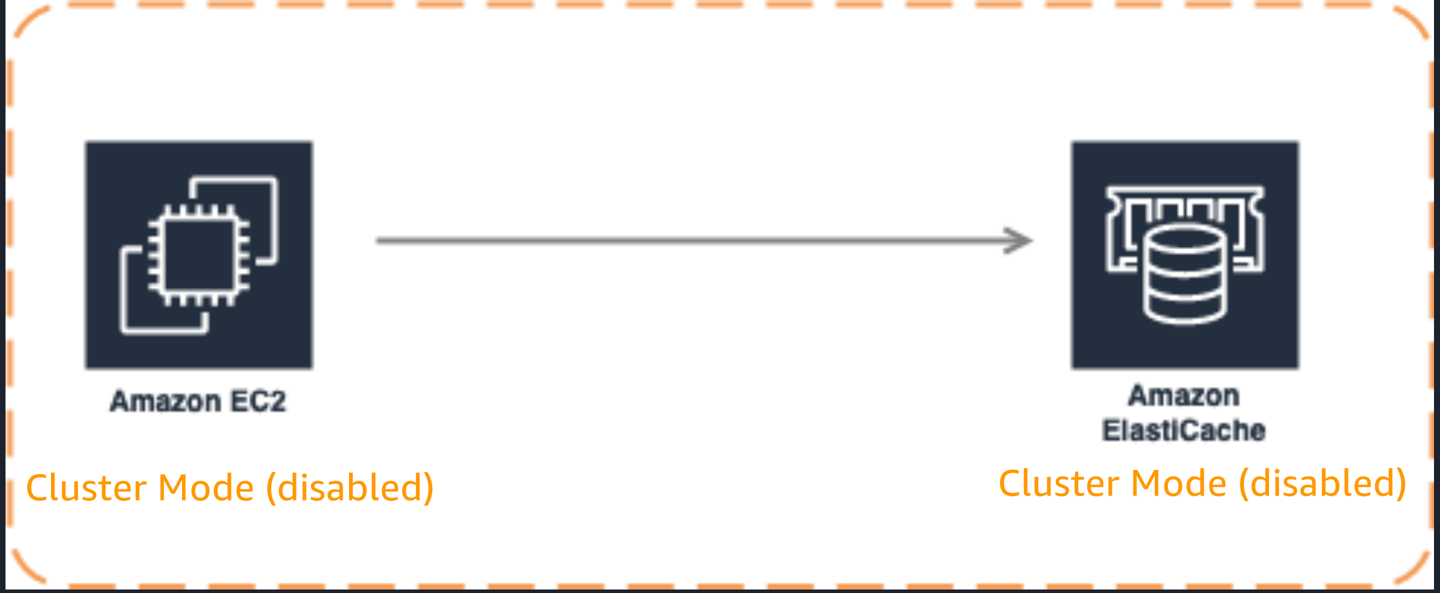
Recommendation: Leverage planned maintenance window

1. Create a Redis Backup
2. Create an Amazon S3 Bucket and Folder
3. Upload Your Backup to Amazon S3
4. Grant ElastiCache Read Access to the .RDB File
5. Seed the ElastiCache Cluster with the .RDB File Data

```
{
   "Version": "2012-10-17",
   "Id": "Policy15397346",
   "Statement": [
      {
         "Sid": "Stmt15399483",
         "Effect": "Allow",
         "Principal": {
            "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
         },
         "Action": [
            "s3:GetObject",
            "s3:ListBucket",
            "s3:GetBucketAcl"
         ],
         "Resource": [
            "arn:aws:s3:::example-bucket",
            "arn:aws:s3:::example-bucket/backup1.rdb",
            "arn:aws:s3:::example-bucket/backup2.rdb"
         ]
      }
   ]
}
```

Migrate using the Online Migration tool





Source **Target**

Overview:

- Replicates data in real-time
- Supported Instances include T3, M4, M5, R4 and R5
- Health monitoring during and after the migration
- Customer decides when to cutover to the migrated cluster

Security - Encryption

In-Transit Encryption

Encrypts application-to-node and node-to-node network communications

TLS 1.0 – 1.2 supported

Server verification / authentication

May impact performance

At-Rest Encryption

Used for snapshots and during replication

May impact performance

Authentication

Ability to set AUTH token

Compliance

- HIPAA Eligibility for ElastiCache for Redis
- Included in AWS Business Associate Addendum
- Redis 3.2.6

ElastiCache Use Cases

Lots of use cases for real-time apps

Caching

Real-time analytics

Gaming leaderboards

Geospatial

Media streaming

Session store

Chat apps

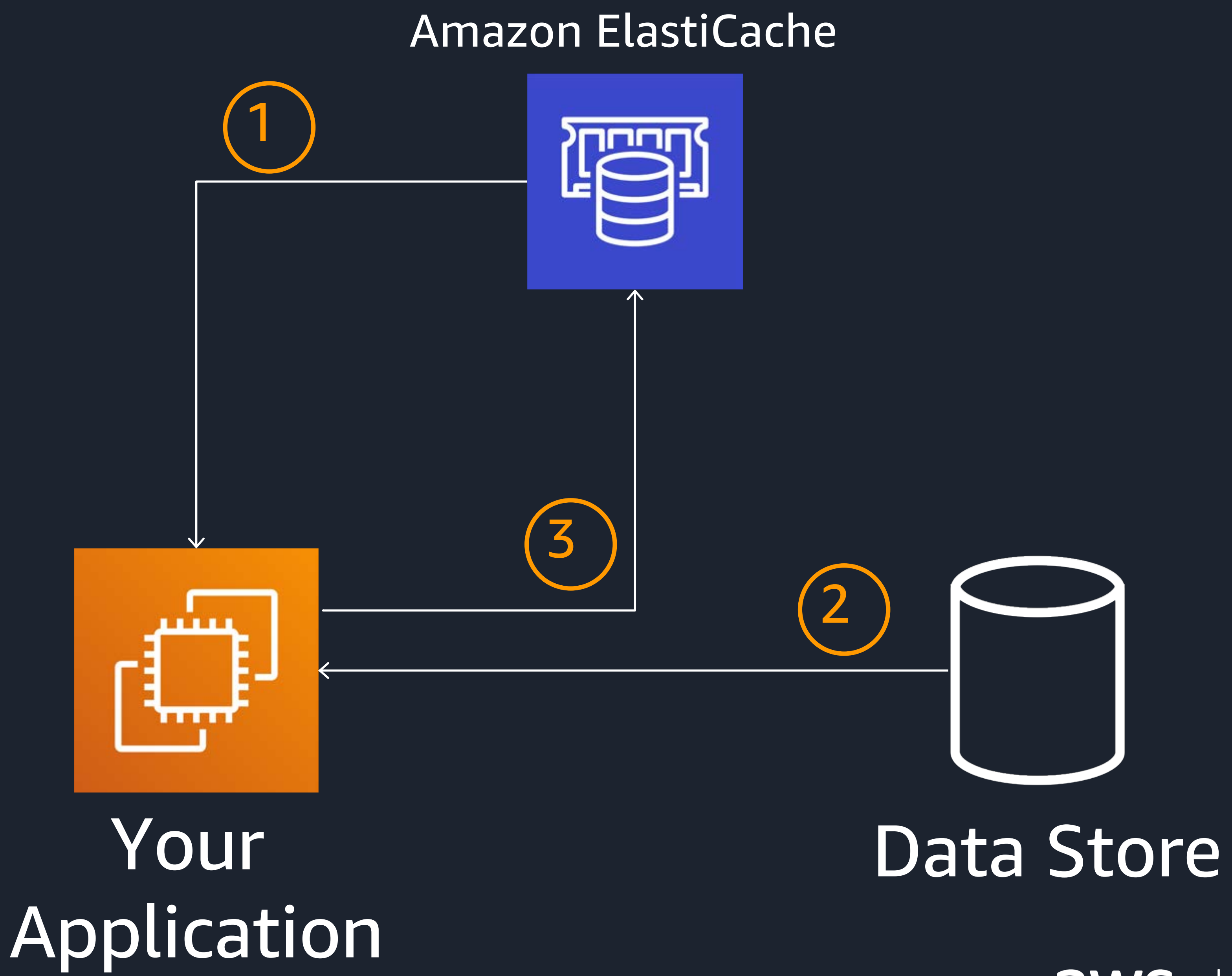
Message queues

Machine learning

Database Query Caching

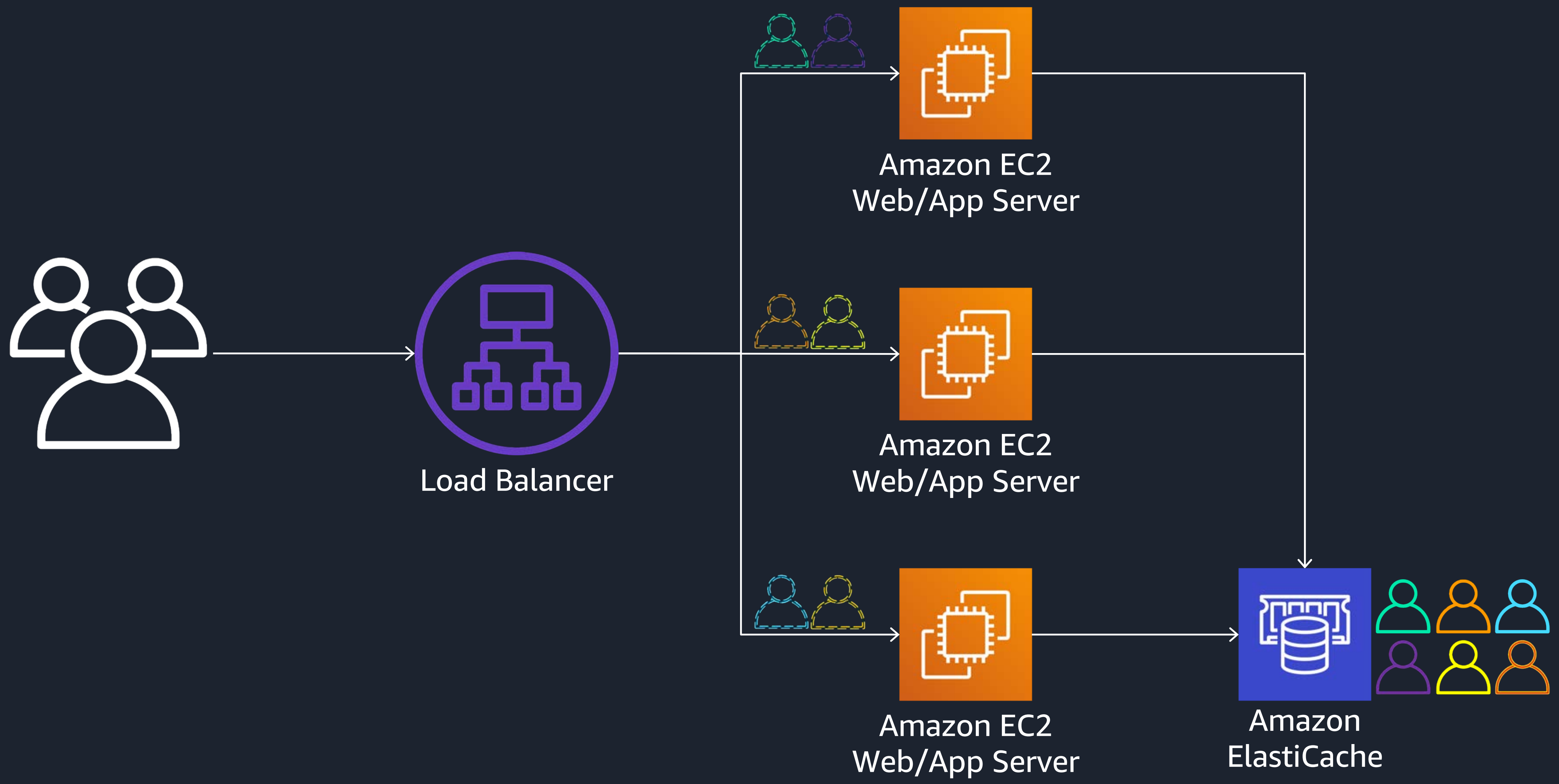
Lazy Loading

1. Cache hit: read from cache
2. Cache miss: read from database
3. Write data to cache (with TTL)



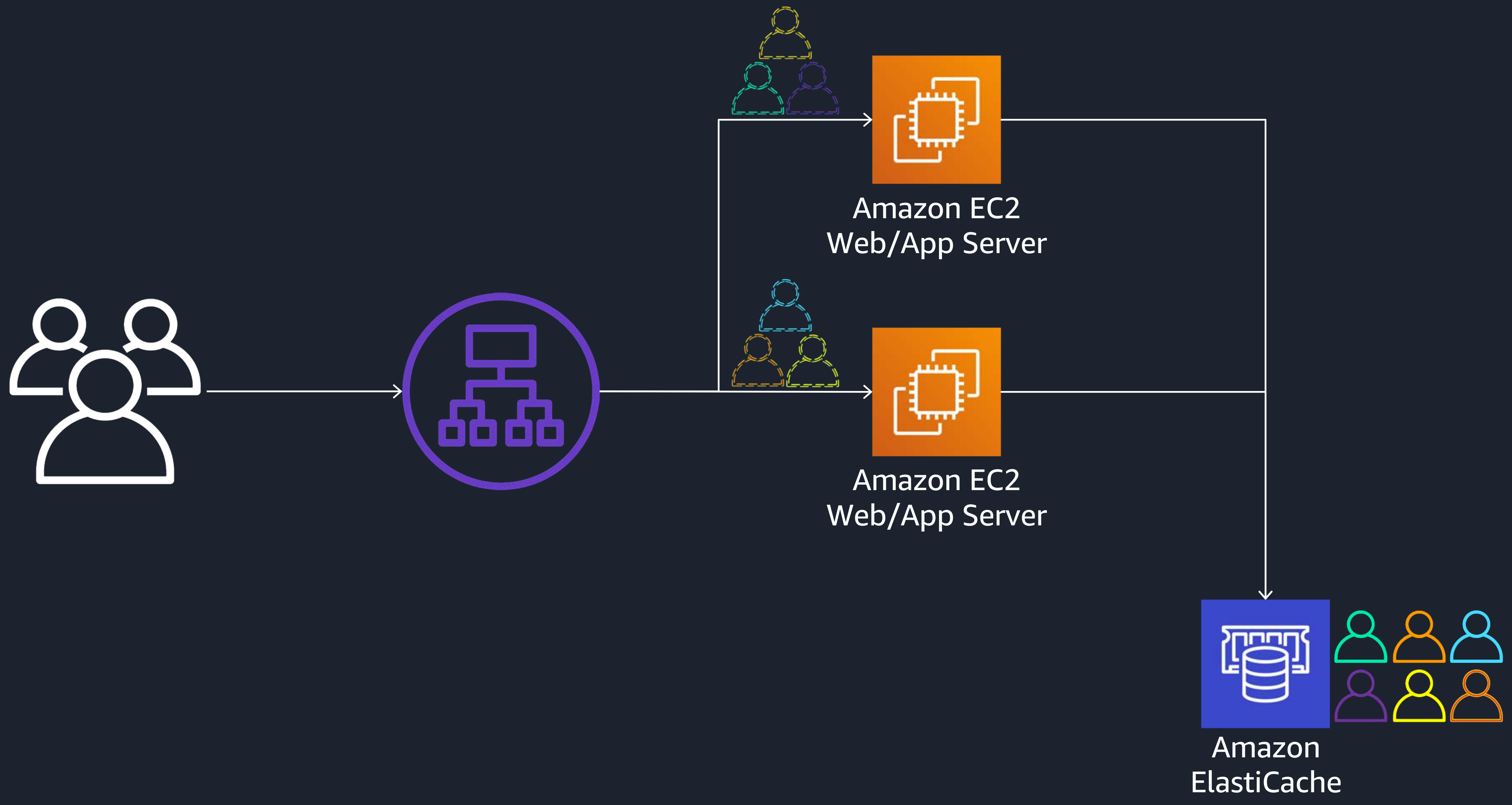
aws databases

Application Session Store



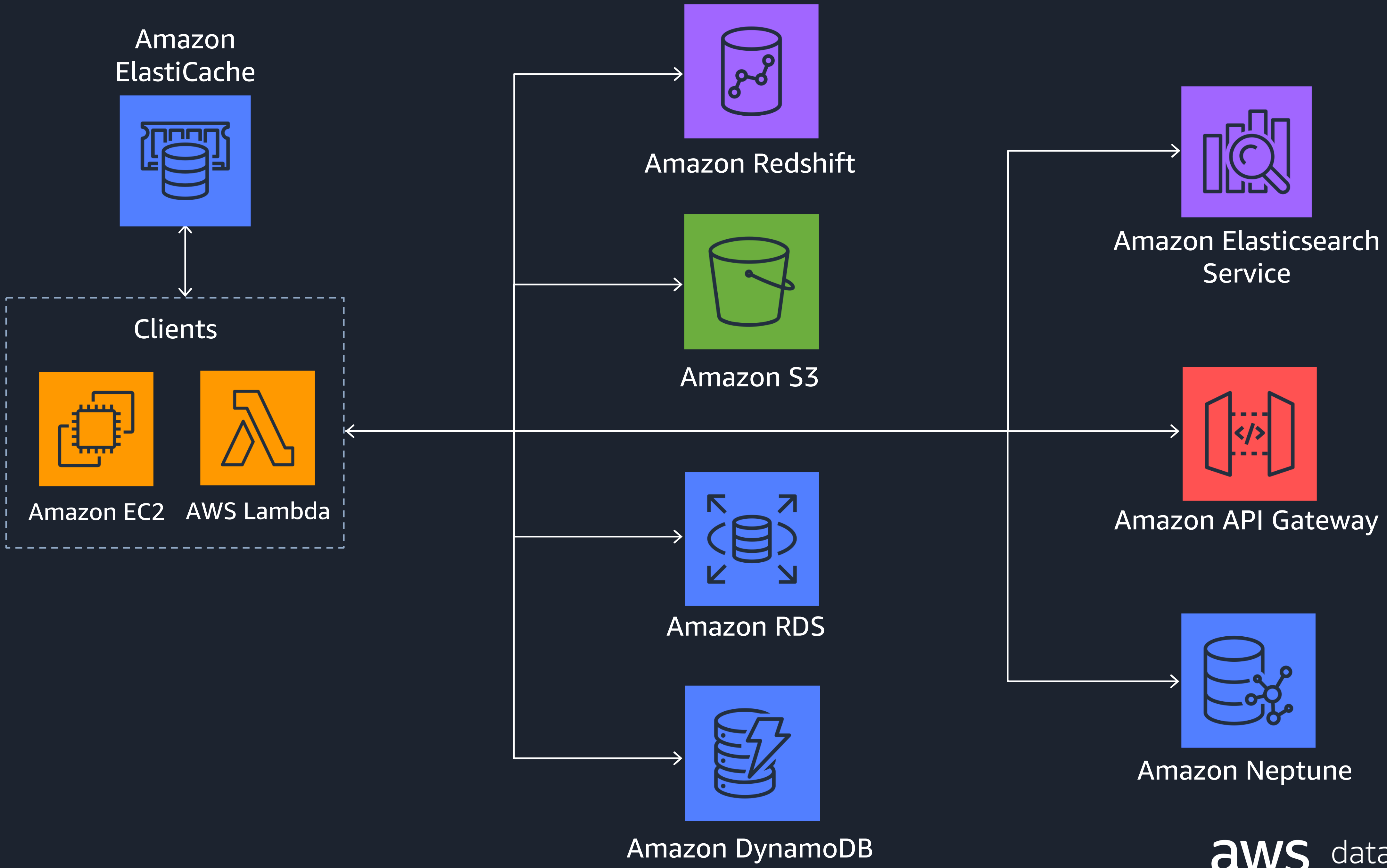
aws databases

Application Session Store



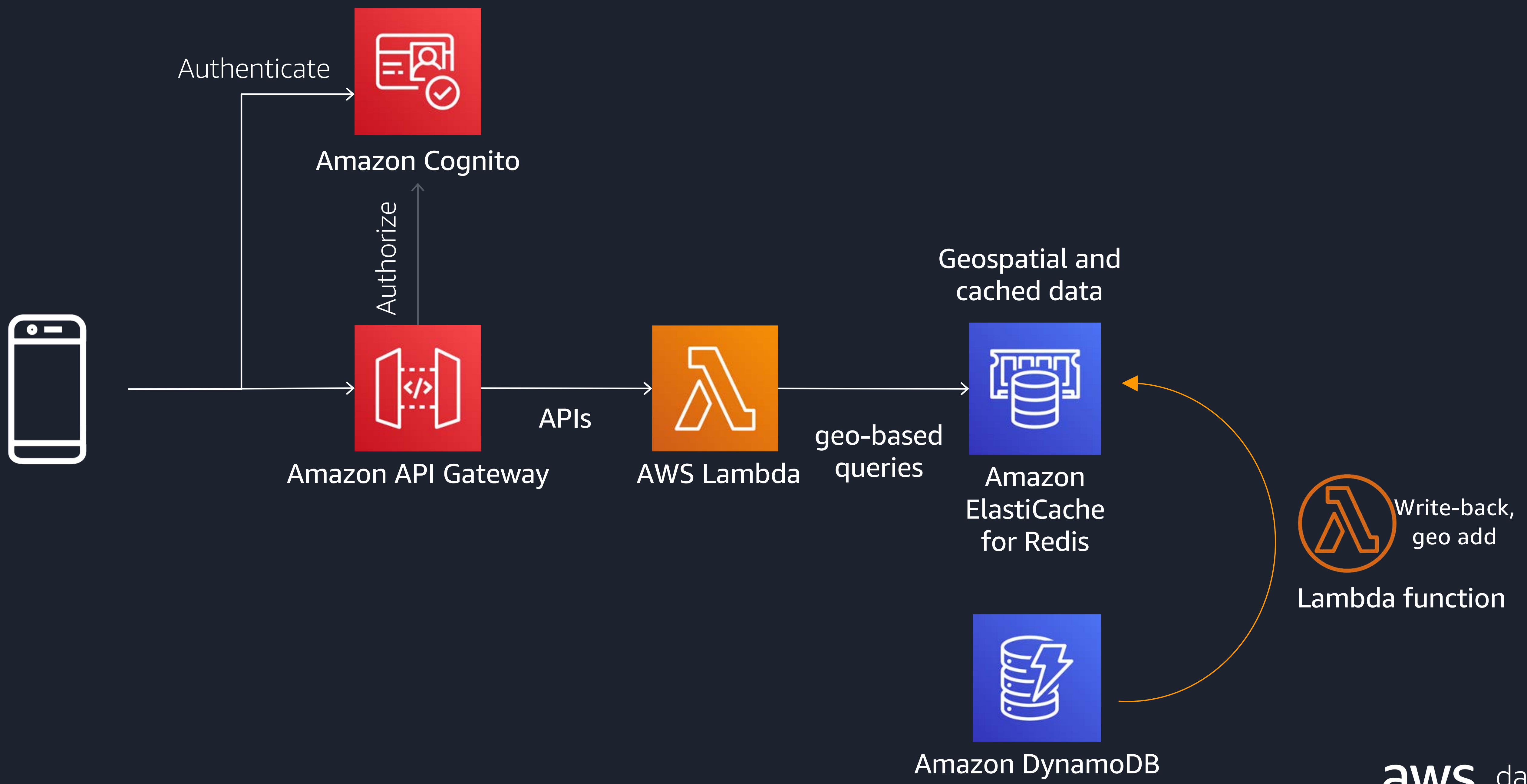
Cache Aside

In-memory data store and cache to decrease access latency, increase throughput, and ease the load off databases and applications



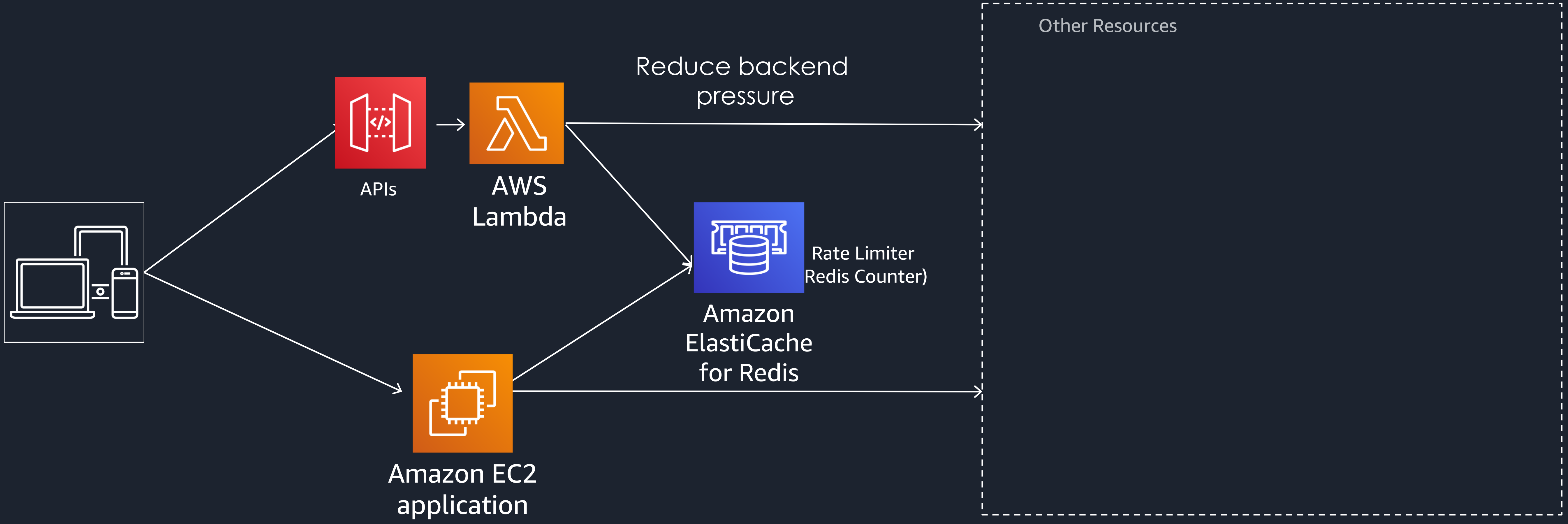
aws databases

Mobile

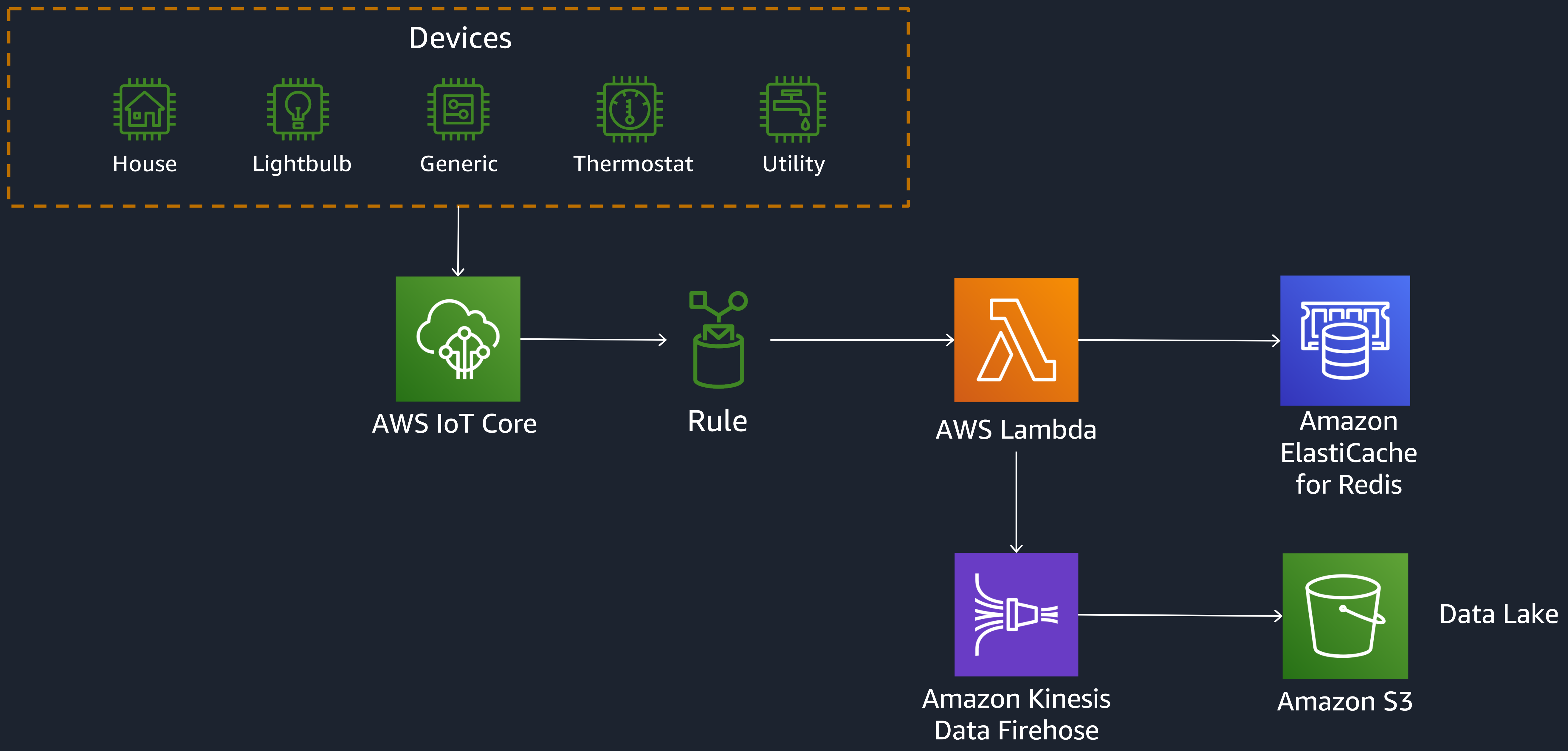


aws databases

Rate Limiting

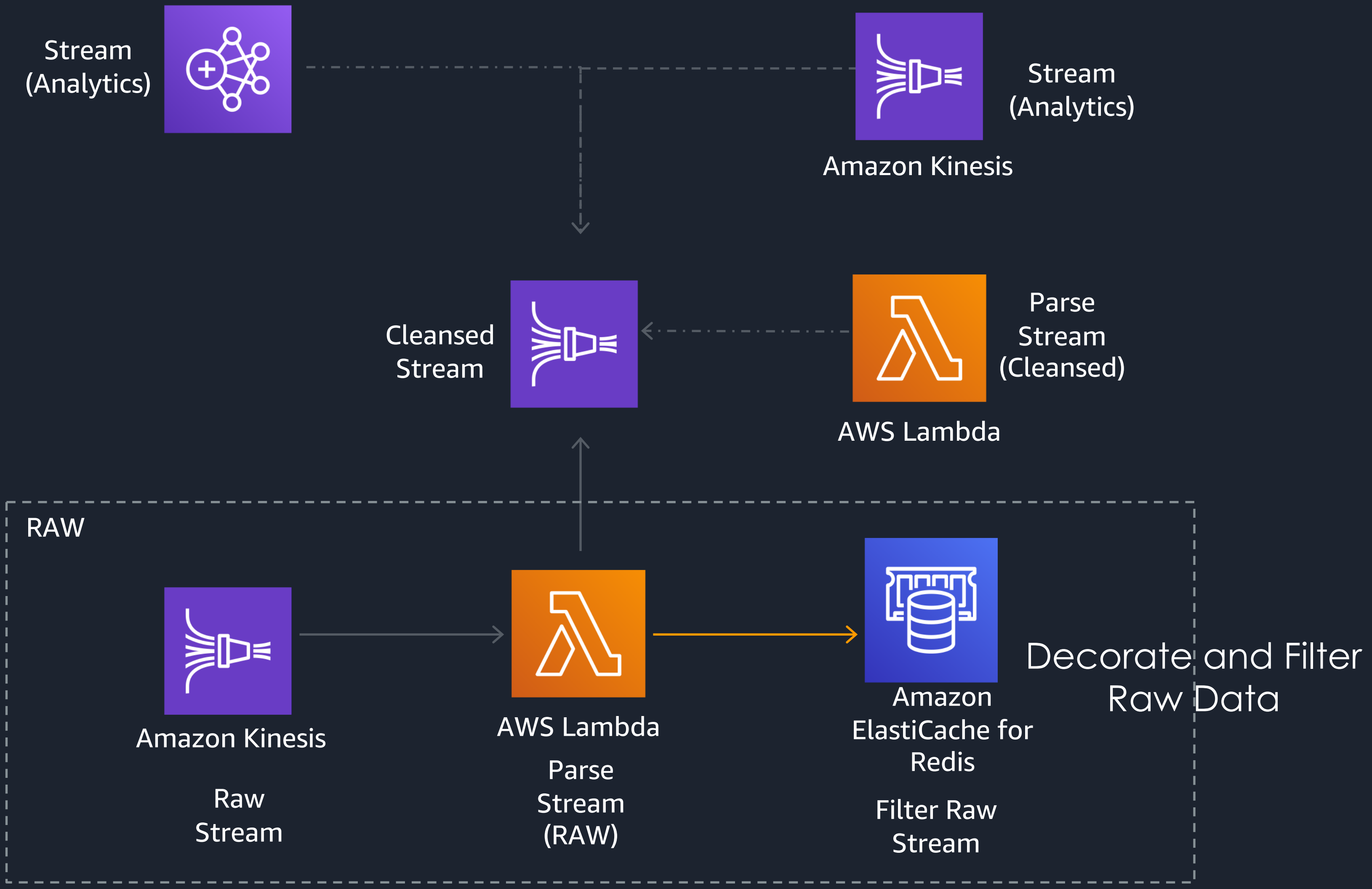


AWS IoT Core



aws databases

Real-time: Data Filtering



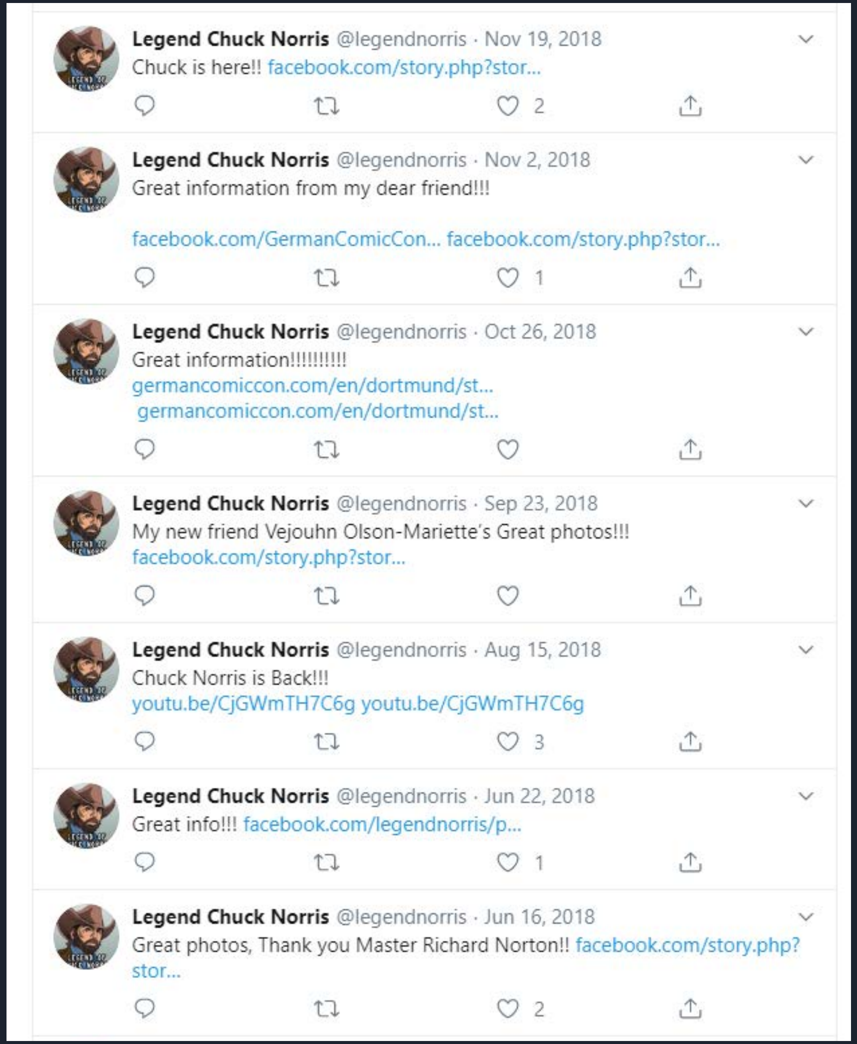
aws databases

Redis data types: STREAMS

Redis steams support a time sequenced series of records (like a log file).

Operations:

- Add records to the end of the stream
- Trim/Discard old entries from the stream
- Ranges of records can be retrieved and/or counted
- Multiple clients can independently process the same stream
- Consumer groups allow clients to split a stream across clients



Legend Chuck Norris @legendnorris · Nov 19, 2018
Chuck is here!! facebook.com/story.php?stor...

Legend Chuck Norris @legendnorris · Nov 2, 2018
Great information from my dear friend!!!

facebook.com/GermanComicCon... facebook.com/story.php?stor...

Legend Chuck Norris @legendnorris · Oct 26, 2018
Great information!!!!!!!!!!
germancomiccon.com/en/dortmund/st...
germancomiccon.com/en/dortmund/st...

Legend Chuck Norris @legendnorris · Sep 23, 2018
My new friend Vejouhn Olson-Mariette's Great photos!!!
facebook.com/story.php?stor...

Legend Chuck Norris @legendnorris · Aug 15, 2018
Chuck Norris is Back!!!
youtu.be/CjGWmTH7C6g youtu.be/CjGWmTH7C6g

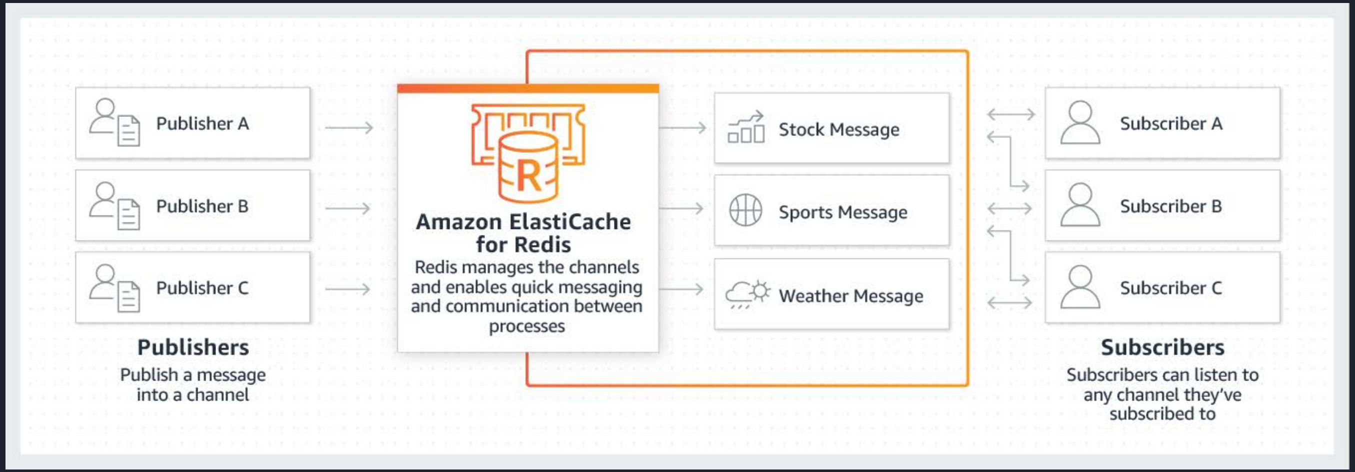
Legend Chuck Norris @legendnorris · Jun 22, 2018
Great info!!! facebook.com/legendnorris/p...

Legend Chuck Norris @legendnorris · Jun 16, 2018
Great photos, Thank you Master Richard Norton!! facebook.com/story.php?stor...

Redis Pub/Sub

- Messages are categorized into channels
- Subscribers can subscribe to multiple patterns or channels
- Publishers publish to a given channel
- Messages are not persisted
  - Clients must be connected to receive
- Two main commands: PUBLISH and SUBSCRIBE

```
> publish sports:patriots "Goooo team!"
(integer) 1
```


Publisher A
Publisher B
Publisher C
Publishers
Publish a message into a channel
Amazon ElastiCache for Redis
Redis manages the channels and enables quick messaging and communication between processes
Stock Message
Sports Message
Weather Message
Subscriber A
Subscriber B
Subscriber C
Subscribers
Subscribers can listen to any channel they've subscribed to

```
> psubscribe sports:*
Reading messages...
1) "psubscribe"
2) "sports:*"
3) (integer) 1

1) "pmessage"
2) "sports:*"
3) "sports:patriots"
4) "Goooo team!"
```

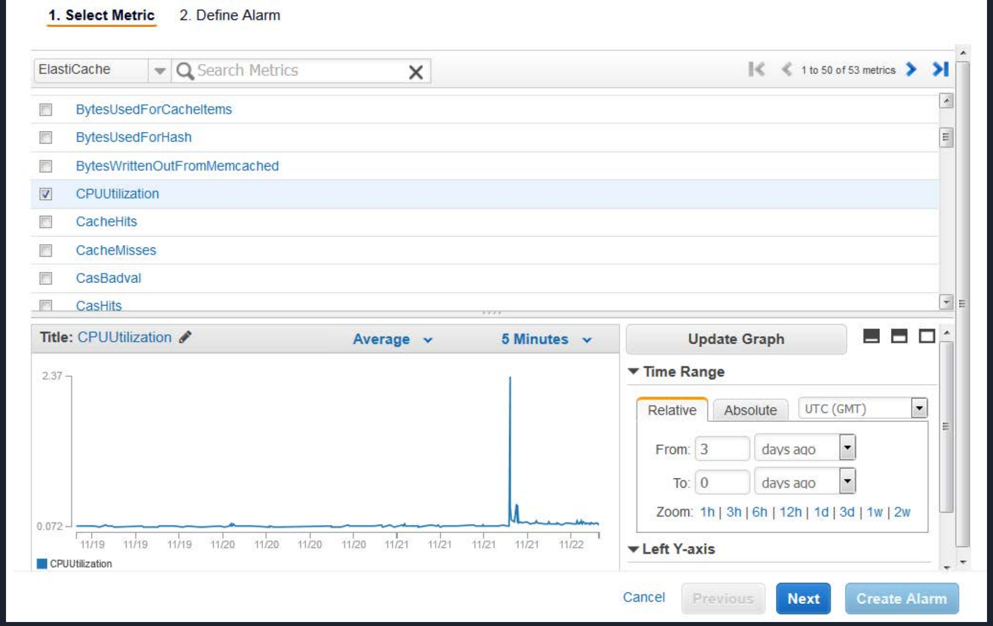
Monitoring, Sizing & Best Practices

aws databases

Key ElastiCache CloudWatch Metrics

- CPUUtilization
    - Memcached – up to 90% ok
- EngineCPUUtilization
    - Redis CPU [Up to 90% OK]
- SwapUsage low
- CacheMisses / CacheHits Ratio low
- Evictions near zero
    - Exception: Russian doll caching
- CurrConnections stable

- Setup alarms with CloudWatch Metrics



ElastiCache

□ BytesUsedForCacheItems
□ BytesUsedForHash
□ BytesWrittenOutFromMemcached
☑ CPUUtilization
□ CacheHits
□ CacheMisses
□ CasBadval
□ CasHits

Title: CPUUtilization   Average   5 Minutes

UTC (GMT)

From: 3 days ago

To: 0 days ago

Redis max-memory policies

Select a max-memory policy based on your workload needs

Eviction Policy Type	Subtype	Name	Description
LRU	** All Keys	`allkeys-lru`	Evicts the least recently used (LRU) regardless of TTL set
LRU	* Volatile	`volatile-lru`	Evicts the least recently used (LRU) from those that have a TTL set
LFU	** All Keys	`allkeys-lfu`	Evict any key using approximated least frequently used (LFU)
LFU	* Volatile	`volatile-lfu`	Evict using approximated LFU among the keys with a TTL set
TTL	* Volatile	`volatile-ttl`	Evicts the keys with shortest TTL set
Random	* Volatile	`volatile-random`	Randomly evicts keys with a TTL set
Random	** All Keys	`allkeys-random`	Randomly evicts keys regardless of TTL set
No Eviction	No Eviction	`no-eviction`	Doesn’t evict keys at all. This blocks future writes until memory frees up.

** Volatile policies only evicts keys with TTLs*

*** Highlighted policies are typically considered safest until key patterns are well understood*

aws databases

ElastiCache Scaling Considerations

- Cluster mode enabled Scale Out/in [add/remove shards]:
  - No downtime, cluster remains available for requests while slots are evenly distributed across Shards
  - If applicable, it is recommended to resize a cluster during off-peak hours to avoid a performance penalty

ElastiCache Scaling Considerations

- Cluster mode enabled Scale Out/in [add/remove shards]:
  - No downtime, cluster remains available for requests while slots are evenly distributed across Shards
  - If applicable, it is recommended to resize a cluster during off-peak hours to avoid a performance penalty
- Scaling Vertically:
  - A new cluster is initialized beside the existing, new node type is applied to all nodes.
  - Upon cluster synchronization, Redis 5.0.5 cutover is <1sec, older versions can take up to a minute.

ElastiCache Scaling Considerations

- Cluster mode enabled Scale Out/in [add/remove shards]:
  - No downtime, cluster remains available for requests while slots are evenly distributed across Shards
  - If applicable, it is recommended to resize a cluster during off-peak hours to avoid a performance penalty
- Scaling Vertically:
  - A new cluster is initialized beside the existing, new node type is applied to all nodes.
  - Upon cluster synchronization, Redis 5.0.5 cutover is <1sec, older versions can take up to a minute.
- Cluster mode disabled Read-only Scaling [add replicas] :
  - Add/Remove replicas incurs no downtime to application.
  - Reader endpoint stays up-to-date in real-time during replica addition/removal & distributes traffic evenly.

ElastiCache Scaling Considerations

- Cluster mode enabled Scale Out/in [add/remove shards]:
  - No downtime, cluster remains available for requests while slots are evenly distributed across Shards
  - If applicable, it is recommended to resize a cluster during off-peak hours to avoid a performance penalty
- Scaling Vertically:
  - A new cluster is initialized beside the existing, new node type is applied to all nodes.
  - Upon cluster synchronization, Redis 5.0.5 cutover is <1sec, older versions can take up to a minute.
- Cluster mode disabled Read-only Scaling [add replicas] :
  - Add/Remove replicas incurs no downtime to application.
  - Reader endpoint stays up-to-date in real-time during replica addition/removal & distributes traffic evenly.
- Compute Node Types:
  - R5 & M5 instance types leveraging AWS Nitro System optimizations and Enhanced IO improvements.
  - This provides significantly better price/throughput allowing your cluster to handle more traffic while keeping the cost low

ElastiCache Scaling Considerations

- Cluster mode enabled Scale Out/in [add/remove shards]:
  - No downtime, cluster remains available for requests while slots are evenly distributed across Shards
  - If applicable, it is recommended to resize a cluster during off-peak hours to avoid a performance penalty
- Scaling Vertically:
  - A new cluster is initialized beside the existing, new node type is applied to all nodes.
  - Upon cluster synchronization, Redis 5.0.5 cutover is <1sec, older versions can take up to a minute.
- Cluster mode disabled Read-only Scaling [add replicas] :
  - Add/Remove replicas incurs no downtime to application.
  - Reader endpoint stays up-to-date in real-time during replica addition/removal & distributes traffic evenly.
- Compute Node Types:
  - R5 & M5 instance types leveraging AWS Nitro System optimizations and Enhanced IO recommended.
  - In addition to significantly better price/throughput, improves seamless scaling and failover operations.
- Redis Engine Version In-Place Upgrade:
  - Upgrade of version with minimal downtime.
  - Cluster available for reads during engine upgrades, writes are interrupted only for <1sec with version 5.0.5
  - Upgrading versions earlier than 5.0.5 can incur <1minute interruption due to DNS propagation.

aws databases

Q&A

Thank you!