**AWS: ELASTICACHE**

**Amazon ElastiCache**

* All reads/write from this caching solution
* Fully managed **in-memory data store;** not on disk
* Scalable
* ***Engines***: Redis and Memcached

***Redis*** *can handle complex data types; replicate data; automatic failover; pub/sub*

***Memcached*** *simple data types; run large nodes with multiple threads*

**Use Cases:**

* Cache frequently accessed data --- user profile, preferences, item description, etc.

**ElastiCache Components: (ElCa)**

* **Node –** smallest building block of ElCa deployment; each node runs an instance; multiple nodes available

***Memcached cluster*** *logical grouping of one or more ElCa nodes; each node has its own endpoints*

***Redis Shard*** *grouping of one to six nodes---*

***Redis Cluster (enabled)*** *has one or multiple shard*

***Redis Cluster (disabled)*** *one shard*

**Replica** nodes are **read-only**

**Primary** node all **read** node will **write** to this

Redis **Replication** group two to six nodes; **asynchronous** application

In **single** node, there is **no replication**

**OPTIMIZING AMAZON ELASTICACHE**

How does data got written into AWS ElCa cluster: (strategies)

* **Lazy loading** caching strategy the loads into the cache when **necessary, such as:**

***If data doesn’t exist on the cache, we call it cache miss***

*-only requested data is cached*

*-node failures are not fatal to application*

*-but increase response time with cache miss “can cause delay getting data into app”*

* **Write-Through** straight forward; **add** data or **updates data** in the **cache** whenever data is **written on database**

**-***data in cache is never stale; always current*

**-***write penalty vs read penalty*

**-***no data gets added during the* ***read***

**TTL “Time to Live”** data expires from cache after specifying this

**Scaling Memcached clusters:**

**\* Horizontal Scaling** add/remove nodes to cluster; read from cache

**-** consistent hashing is calculating the hash of the object’s key and determine the specific node key is stored

**-** utilizes Auto Discovery

**-** 1 to 20 nodes

**-** In **Redis Cluster** (disabled mode) u **cannot partition** ur data across **multiple nodes**

**\* Vertical Scaling** changing node types

* **Scaling Redis Offline and Online**
* *Resharding as changing the number of shards*
* *Rebalancing the keys across those shards*

***Two ways:***

* **Offline** – unable to serve request; able to change node type, AZ, engine version, etc.
* **Online** – cluster will stay up and running; scale out, in, rebalance; cannot specify shard, etc.; ensure sufficient free memory availability

**Resharding –** compute-intensive operation; monitor CloudWatch metrics

**AVOID** expensive commands

**FOLLOW Lua** best practices

**Redis Reserved Memory** memory set aside for nondate use

* Control using **reserved-memory and reserved-memory-percent** parameter
* **Must** be set with parameter group
* **ELASTICACHE ENDPOINTS**

**Red-Clus (disabled) endpoints:**

* **Primary endpoint** is a DNS name that always resolves to the **primary node** in the cluster; immune to changes; **write/read**
* **Reader endpoint split incoming connections** to the endpoint between all read replicas; **read**
* **Node endpoint** resolves specifics endpoints; DNS of the load balancer

**Red-Clus (enabled) endpoints:**

* Has multiple shares means multiple primary node
* **Configuration endpoint** know all the primary and node endpts in the cluster; **read/write**
* **Node endpoint** for the primary node can be used for both **read and write**
* **Replicas endpoint** for **reading**

**Memcached endpoints**

* Read/write endpoints
* Must have **auto-discovery**
* **Does not** maintain node-to-key mapping
* **Redis Global Datastore**
* **Can help increase the responsiveness of ur app**
* **Asynchronous replication**
* **Disaster recovery**
* **Primary cluster** accepts **writes and read**; **writes** are replicated to all cluster within the **Global Datastore**; “**active**”
* **Secondary cluster** accepts **read requests**; replicates data updates from a primary cluster; **“passive cluster”**
* **Managing Costs with Tags**
* **ElastiCache** cost allocation tag is a **key-value** pair that u define and associate with ElastiCache Resource
* **Analyze** AWS bill based on tags
* Can be managed using **console, CLI, API**

**AUTOMATE ELASTICACHE DEPLOYMENTS WITH CLOUDFORMATION**

**Infra as a Code –** replicate ur infra **quickly**

* Control and track infra changes
* Critical component of DevOps

**AWS CloudFormation**

* Iaas to model and provision any AWS resources
* Can be coded using JSON/YAML or programming language (AWS CDK)
* **CLOUDFORMATION CHANGE SETS**
* Shows impact to running resources with the template change
* Displayed **BEFORE** the template is executed

**CloudFormation DeletionPolicy**

* Resources deleted when stack is deleted (default)
* **retain or snapshot**
* Retain elasticache
* **Only** available for redis cluster

**MIGRATE AND BACKUP/RESTORE OF ELASTICACHE**

* These are **not** supported for **memcached;** only for **redis**
* **Are kept** in S3 for durable storage
* Copy the entire redis cluster
* Backup from one of the primary node or one of the read replicas; IT IS PREFERRABLE TO USE **READ REPLICAS** FOR **BACKUP** IN HAVING **BETTER PERFORMANCE**
* Adjust the reserved-memory-percent
* Can enable **automatic backups**

**Backup window** specify backup period during each day

**Backup retention limit** number of days backup is retained in Amazon S3; 35 days

* **Migration** is also **not supported** for **memcached**; **redis** only
* **Constraints: clustermode disabled; encryption not enabled; multi-AZ with auto failover**
* **LOGGING AND MONITORING**

**Amazon CloudWatch** application logging and monitoring

**AWS CloudTrail** infra logging

**ElastiCache Monitoring**

* To maintain reliability, availability and performance
* **Cloudwatch** provides metrics out of the box
* Memcached and redis have different metrics:

***CPUUtilization*** *host level metric reported as percentage*

***Swapusage*** *host level metric reported in bytes*

***Evictions*** *not a host level; cache engine metric*

***CurrConnections*** *cache engine metric indicating application issue*

*UNDER CLOUDWATCH > METRIC > ELASTICACHE*

*ClusterID show u all*

*Aggregated both Memcached and redis*

***ElastiCache Error Types:***

* ***HTTP status code 5xx*** *SERVER ERROR*
* ***HTTP 500*** *INTERNAL SERVER ERROR*
* ***HTTP 503*** *SERVICE UNAVAILABLE*
* ***HTTPS status code 4xx*** *CLIENT ERROR*

***Common errors:***

***Max number of manual snapshots for this node***

***Reached ur quota***

**Infrastructure Logging**

**CloudTrail** record of actions – in JSON format; will go to Amazon S3

**Amazon SNS** notifies every logs

**SECURING AMAZON ELASTICACHE WITH ENCRYPTION AND IAM**

***ElastiCache Security***

* **Data protection at rest**
* KMS supported
* Encryption can have performance impact
* **Data protection in transit**
* Encrypt traffic using HTTPS (SSL/TLS)
* Internetwork traffic privacy

**Redis Auth**

Amazon E2, Token/Password

**Redis Role-based access control (RBAC)**

Create user with ACL rules

Create user group

Assign user group to replication group