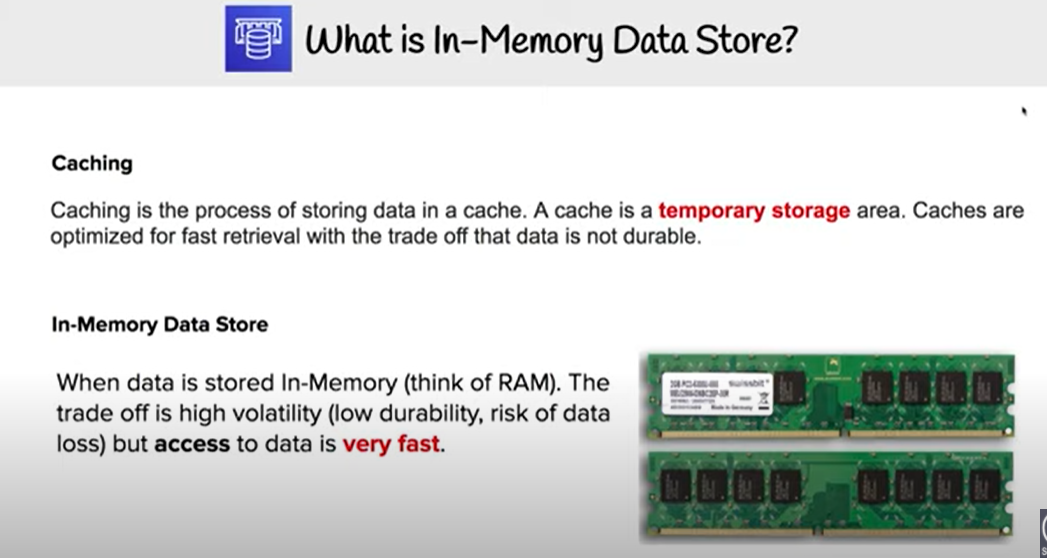
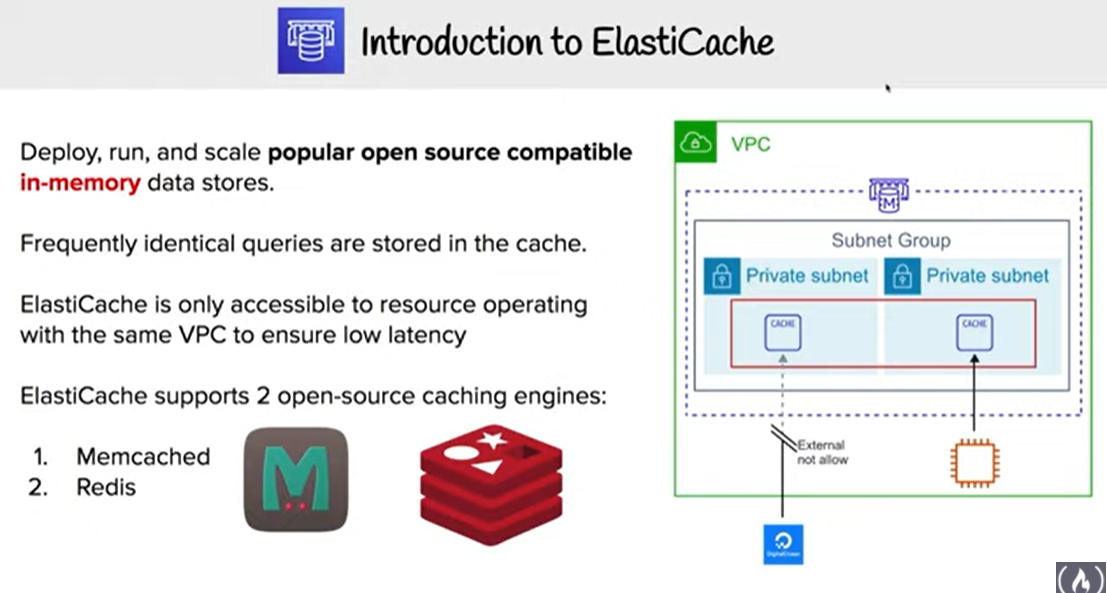
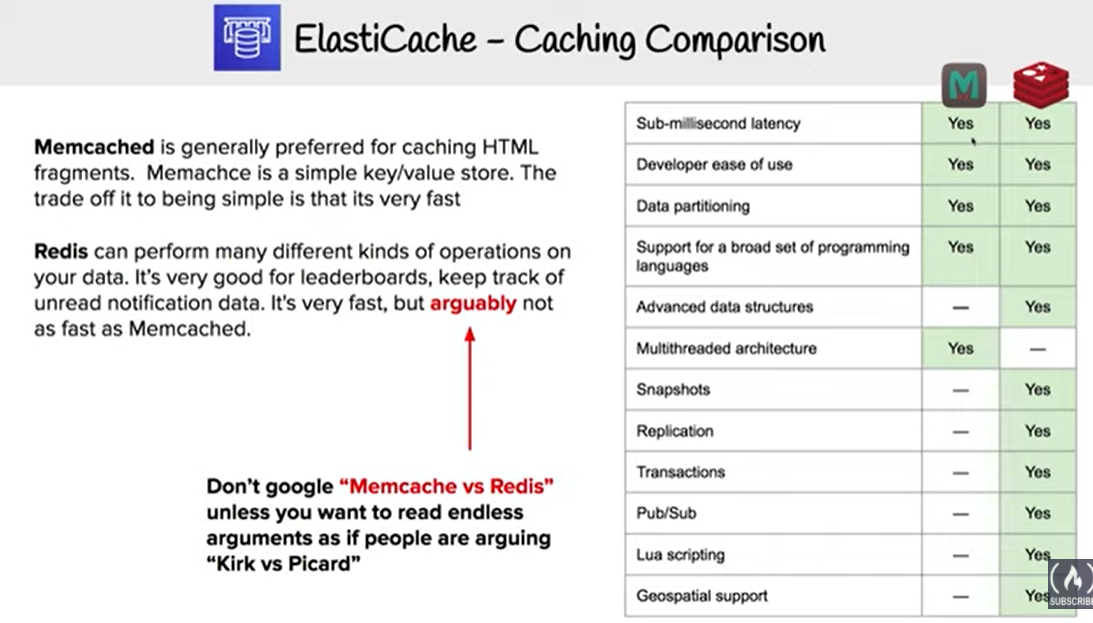
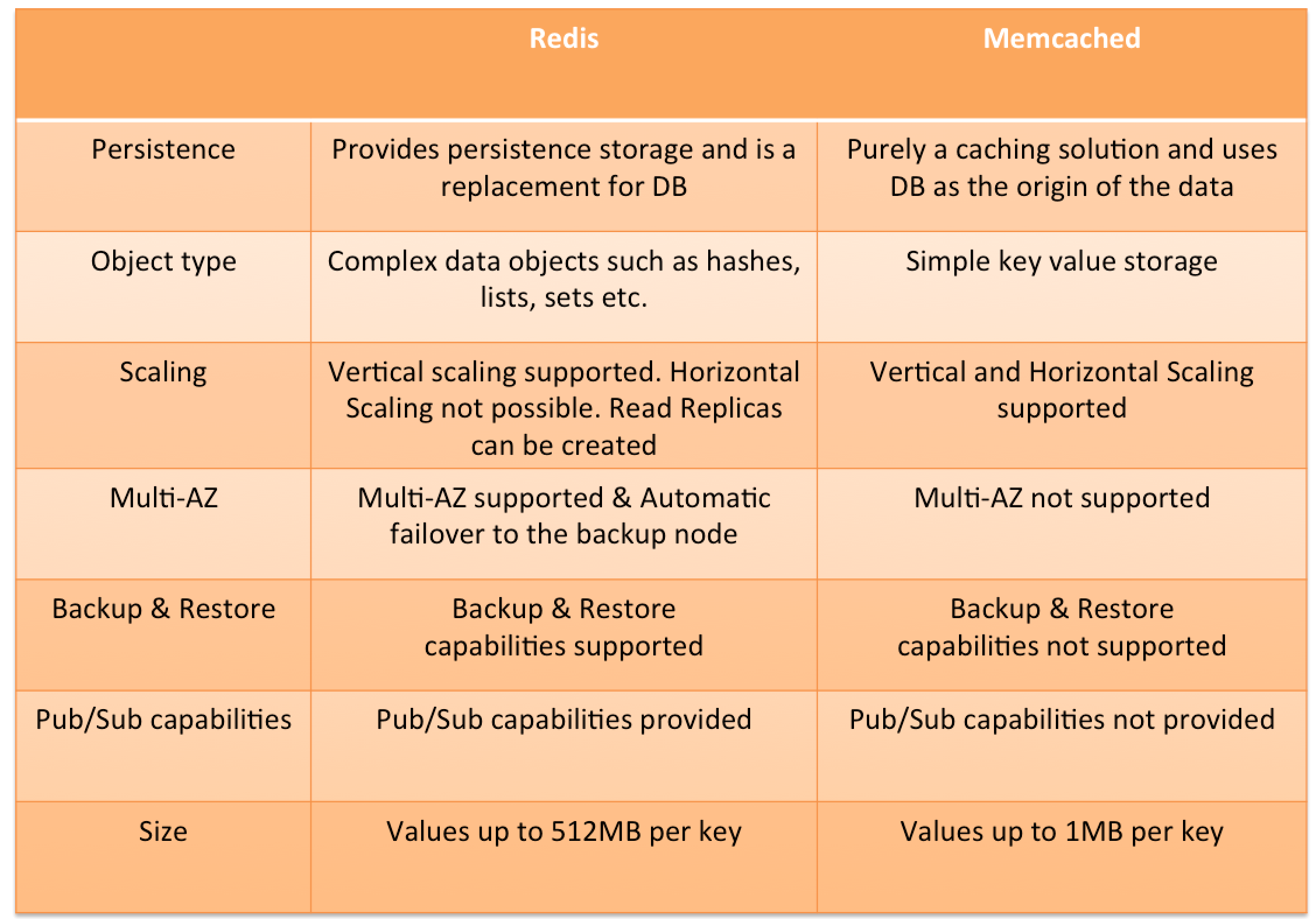
AWS ElastiCache

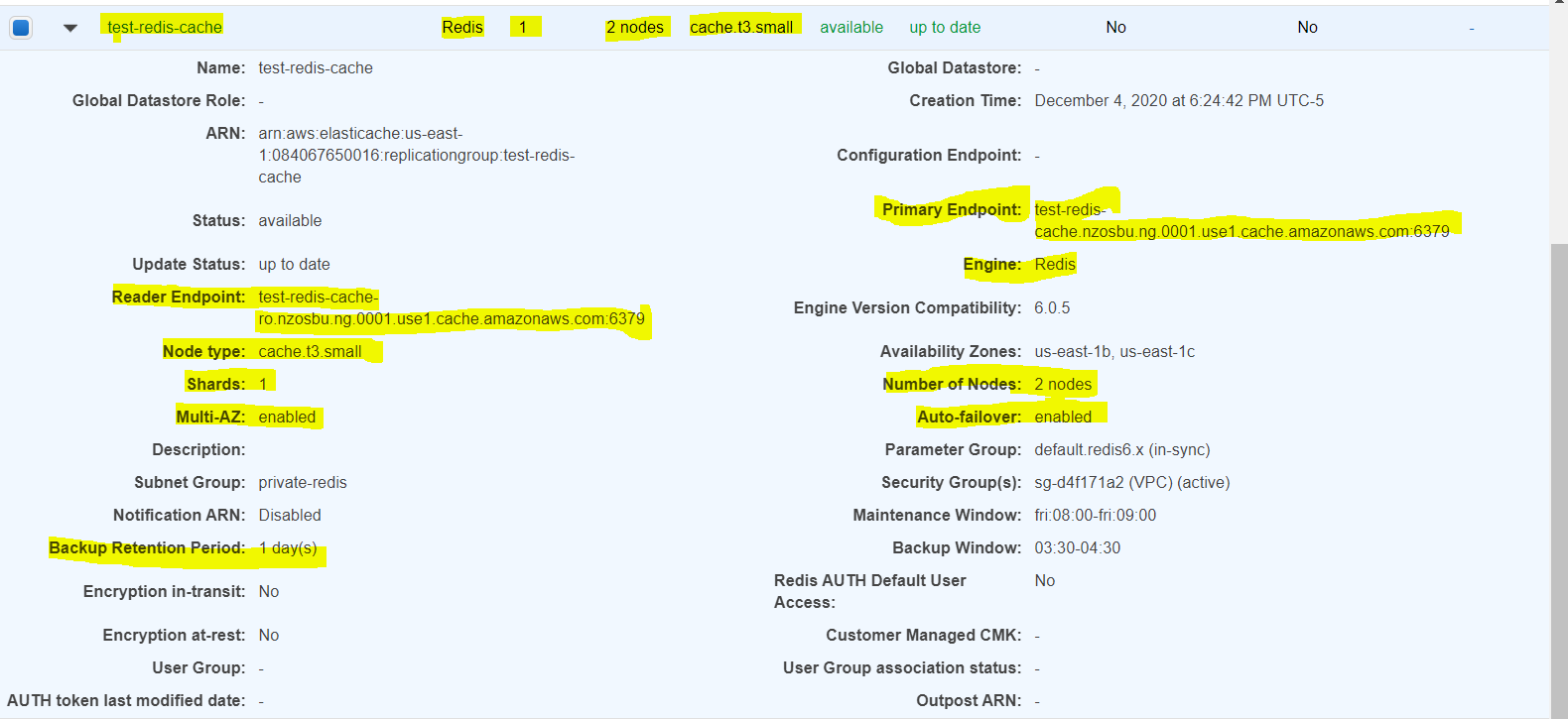




* AWS ElastiCache is a managed web service that helps deploy and run Memcached or Redis protocol-compliant cache clusters in the cloud easily
* ElastiCache is available in two flavours: **Memcached and Redis**
* ElastiCache helps
  + simplify and offload the management, monitoring, and operation of in-memory cache environments, enabling the engineering resources to focus on developing applications
  + automate common administrative tasks required to operate a distributed cache environment.
  + improves the performance of web applications by allowing retrieval of information from a fast, managed, in-memory caching system, instead of relying entirely on slower disk-based databases.
  + helps improve load & response times to user actions and queries, but also reduce the cost associated with scaling web applications
  + helps automatically detect and replace failed cache nodes, providing a resilient system that mitigates the risk of overloaded databases, which can slow website and application load times
  + provides enhanced visibility into key performance metrics associated with the cache nodes through integration with CloudWatch
  + code, applications, and popular tools already using Memcached or Redis environments work seamlessly, with being protocol- compliant with Memcached and Redis environments
* ElastiCache provides in-memory caching which can
  + significantly lower latency and improve throughput for many
    - read-heavy application workloads for e.g. social networking, gaming, media sharing and Q&A portals or
    - compute-intensive workloads such as a recommendation engine
  + improve application performance by storing critical pieces of data in memory for low-latency access.
  + be used to cache results of I/O-intensive database queries or the results of computationally-intensive calculations.
* ElastiCache currently allows access only from the EC2 network and cannot be accessed from outside networks like on-premises servers







## test-redis-cache.nzosbu.ng.0001.use1.cache.amazonaws.com => test-redis-cache-001.nzosbu.0001.use1.cache.amazonaws.com => 10.10.20.91

## test-redis-cache-ro.nzosbu.ng.0001.use1.cache.amazonaws.com => test-redis-cache-002.nzosbu.0001.use1.cache.amazonaws.com => 10.10.20.60

## 

## Redis

* [Redis](http://redis.io/) is an open source, BSD licensed, advanced key-value cache & store
* ElastiCache enables the management, monitoring and operation of a Redis node; creation, deletion and modification of the node
* ElastiCache for Redis can be used as a primary in-memory key-value data store, providing fast, sub millisecond data performance, high availability and scalability up to 16 nodes plus up to 5 read replicas, each of up to 3.55 TiB of in-memory data
* ElastiCache for Redis supports (similar to RDS features)
  + Redis Master/Slave replication.
  + Multi-AZ operation by creating read replicas in another AZ
  + Backup and Restore feature for persistence using snapshots
* ElastiCache for Redis can be vertically scaled upwards by selecting a larger node type, however it cannot be scaled down
* Parameter group can be specified for Redis during installation, which acts as a “container” for Redis configuration values that can be applied to one or more Redis primary clusters
* Append Only File (AOF)
  + provides persistence and can be enabled for recovery scenarios
  + if a node restarts or service crash, Redis will replay the updates from an AOF file, thereby recovering the data lost due to the restart or crash
  + **cannot protect against all failure scenarios, cause if the underlying hardware fails, a new server would be provisioned and the AOF file will no longer be available to recover the data**
  + **Enabling Redis Multi-AZ is a Better Approach to Fault Tolerance, as failing over to a read replica is much faster than rebuilding the primary from an AOF file**

### 

### Redis Read Replica

* Read Replicas help provide Read scaling and handling failures
* Read Replicas are kept in sync with the Primary node using Redis’s asynchronous replication technology
* Redis Read Replicas provides
  + Horizontal scaling beyond the compute or I/O capacity of a single primary node for read-heavy workloads.
  + Serving read traffic while the primary is unavailable either being down due to failure or maintenance
  + Data protection scenarios to promote a Read Replica as primary node, in case the primary node or the AZ of the primary node fails
* ElastiCache supports initiated or forced failover where it flips the DNS record for the primary node to point at the read replica, which is in turn promoted to become the new primary
* Read replica cannot span across regions and may only be provisioned in the same or different AZ of the same Region as the cache node primary

### Redis Multi-AZ

* **ElastiCache for Redis shard consists of a primary and up to 5 read replicas**
* Redis asynchronously replicates the data from the primary node to the read replicas
* ElastiCache for Redis Multi-AZ mode
  + provides enhanced availability and smaller need for administration as the node failover is automatic
  + impact on the ability to read/write to the primary is limited to the time it takes for automatic failover to complete
  + no longer needs monitoring of Redis nodes and manually initiating a recovery in the event of a primary node disruption
* **During certain types of planned maintenance, or in the unlikely event of ElastiCache node failure or AZ failure,**
  + **it automatically detects the failure,**
  + **selects a replica, depending upon the read replica with the smallest asynchronous replication lag to the primary, and promotes it to become the new primary node**
  + **it will also propagate the DNS changes so that the the primary endpoint remains the same**

### Redis Backup & Restore

* Backup and Restore allows users to create snapshots of the Redis clusters
* Snapshots can be used for recovery, restoration, archiving purpose or warm start an ElastiCache for Redis cluster with preloaded data
* Snapshots can created on a cluster basis and uses Redis’ native mechanism to create and store an RDB file as the snapshot
* Increased latencies for a brief period at the node might be encountered while taking a snapshot, and is recommended to be taken from a Read Replica minimizing performance impact
* Snapshots can be created either automatically (if configured) or manually
* ElastiCache for Redis cluster when deleted removes the automatic snapshots. However, manual snapshots are retained

## Memcached

* [Memcached](http://memcached.org/) is an in-memory key-value store for small chunks or arbitrary data
* ElastiCache for Memcached can be used to cache a variety of objects
  + from the content in persistent data stores such as RDS, DynamoDB, or self-managed databases hosted on EC2) to
  + dynamically generated web pages for e.g. with Nginx or
  + transient session data that may not require a persistent backing store
* ElastiCache for Memcached
  + can be scaled Vertically by increasing the node type size
  + can be scaled Horizontally by adding and removing nodes
  + does not support persistence of data
* ElastiCache for Memcached cluster can have
  + nodes can span across multiple AZs within the same region
  + maximum of 20 nodes per cluster with a maximum of 100 nodes per region (soft limit and can be extended)
* ElastiCache for Memcached supports auto discovery, which enables automatic discovery of cache nodes by clients when they are added to or removed from an ElastiCache cluster

## ElastiCache Mitigating Failures

* ElastiCache should be designed to plan so that failures have a minimal impact upon your application and data
* Mitigating Failures when Running Memcached
  + Mitigating Node Failures
    - spread the cached data over more nodes
    - as Memcached does not support replication, a node failure will always result in some data loss from the cluster
    - having more nodes will reduce the proportion of cache data lost
  + Mitigating Availability Zone Failures
    - locate the nodes in as many availability zones as possible, only the data cached in that AZ is lost, not the data cached in the other AZs
* Mitigating Failures when Running Redis
  + Mitigating Cluster Failures
    - Redis Append Only Files (AOF)
      * enable AOF so whenever data is written to the Redis cluster, a corresponding transaction record is written to a Redis AOF
      * when Redis process restarts, ElastiCache creates a replacement cluster and provisions it and repopulating it with data from AOF
      * It is time consuming
      * AOF can get big
      * Using AOF cannot protect you from all failure scenarios
    - Redis Replication Groups
      * A Redis replication group is comprised of a single primary cluster which your application can both read from and write to, and from 1 to 5 read-only replica clusters.
      * Data written to the primary cluster is also asynchronously updated on the read replica clusters
      * When a Read Replica fails, ElastiCache detects the failure, replaces the instance in the same AZ and synchronizes with the Primary Cluster
      * **Redis Multi-AZ with Automatic Failover, ElastiCache detects Primary cluster failure, promotes a read replica with least replication lag to primary**
      * **Multi-AZ with Auto Failover is disabled, ElastiCache detects Primary cluster failure, creates a new one and syncs the new Primary with one of the existing replicas**
  + Mitigating Availability Zone Failures
    - locate the clusters in as many availability zones as possible

