* Cache란 무엇이며 왜 써야 하는가?
 - Cache는 CPU 칩 안에 들어가 있는 작은 메모리(물리적 실체).
 - 프로세서가 필요한 데이터가 있을 때마다 메인 메모리에 일일이 접근하여 속도가 지연되는 것을 막기 위해 자주 사용하는 데이터를 담아두는 곳
 - 즉 처리 속도 향상을 위해 존재하는 작은 칩이자 메모리
 - L1,L2,L3로 나뉘며 숫자가 적을 수록 도달하는 속도가 빠름
 - Cache는 CPU와 메모리 사이 뿐만 아니라, 메모리와 디스크 사이에서도 발생함
 - 후술할 In Memory Cache는 메모리와 디스크 사이의 Caching을 의미

* In Memory Cache(In Memory DataBase)
 - 데이터 처리 속도를 향상시키기 위한 메모리 기반의 DBMS
 - 메모리 위에 모든 데이터를 올려두고 사용하는 데이터베이스의 일종(ElastiCache가 AWS 카테고리에서 DB 부분에 있는 이유)
 - 디스크에 최적화된 Database(RDS 등)에서 저장된 쿼리 결과나 자주 사용하는 데이터를 메모리에 적재하여 사용하는 것은 비효율적
 - 즉 모든 데이터를 메모리 위에 올려두어 굳이 디스크 기반의 데이터베이스에까지 이동하여 데이터를 가져와 속도가 저하되는 것을 막음
 - 또한 데이터베이스의 데이터뿐만 아니라, 디스크, 세션, 기타 동적으로 생성된 데이터를 저장할 수 있음
 - 다만 메모리 기반의 데이터베이스이기 때문에, 휘발성 메모리라는 단점이 존재하며 전원 공급 차단시 모든 데이터가 유실되고 할당된 메모리에 한해 저장 가능

* ElastiCache
 - AWS의 In Memory Cache Service
 - Memcached와 Redis로 나뉨
 - Memached, Redis 모두 비관계데이터베이스형(NosQL) 서비스이며, Key-Value 기반임
 - Memached, Redis 모두 이미 존재하는 서비스이며 AWS에서 사용가능하도록 구현한 것
 - ElastiCache는 Node로 구성되어 서비스를 제공하며, Node는 EC2처럼 다양한 Type을 가지고 유형에 따라 다양한 메모리 크기를 가짐
 - 다양한 Type을 갖는 이유는 적은 양의 메모리가 필요할 경우, 작은 Type의 Node를 사용하여 비용을 적게 들게 하기 위함
 - 유형이 결정된 Node들은 '고정된' 메모리 크기를 가지며, 각자의 DNS로 이루어진 엔드포인트를 보유함

* Memcached의 특징
 - Cluster로 구성되어 있으며, Cluster 내에는 Node들이 존재하여 인 메모리 캐시로서의 역할을 담당함
 - 각 Node는 Type별로 메모리를 보유하며 서비스를 제공하며, 필요시 Node를 늘려 서비스 용량을 향상시킬 수 있음
 - 각 Node별로 AZ를 따로 둘 수 있지만, 장애 조치(Failover)가 불가능하고 복제본을 둘 수 없음

* Redis의 특징
 - 기본적으로 Cluster로 구성되지는 않지만, Cluster로 구성이 가능하며 샤드와 Node를 가지고 있음
 - 샤드는 여러 Node로 구성되며, 하나의 Node가 읽기/쓰기를 담당하고 나머지 Node는 복제본 역할을 함
 - Cluster로 구성되지 않은 Redis는 하나의 샤드만을 가지지만, Cluster로 구성될 경우 다수의 샤드를 갖게 됨
 - 복제본을 가지므로, 장애조치(복제본을 기본노드로 승격)가 가능하며 Multi-AZ 기능을 지원함

**참고링크
- https://docs.aws.amazon.com/ko_kr/AmazonElastiCache/latest/mem-ug/WhatIs.html
- https://docs.aws.amazon.com/ko_kr/AmazonElastiCache/latest/red-ug/WhatIs.html
- https://aws.amazon.com/ko/elasticache/faqs/
- http://pyrasis.com/book/TheArtOfAmazonWebServices/Chapter15
- https://engkimbs.tistory.com/869
- https://ko.wikipedia.org/wiki/Memcached
- https://ko.wikipedia.org/wiki/%EB%A0%88%EB%94%94%EC%8A%A4
- https://charsyam.wordpress.com/2016/07/27/%EC%9E%85-%EA%B0%9C%EB%B0%9C-%EC

%99%9C-cache%EB%A5%BC-%EC%82%AC%EC%9A%A9%ED%95%98%EB%8A%94%EA%B0%80/
 - https://parksb.github.io/article/29.html
 - https://ojava.tistory.com/70
 - http://blog.leekyoungil.com/?p=200