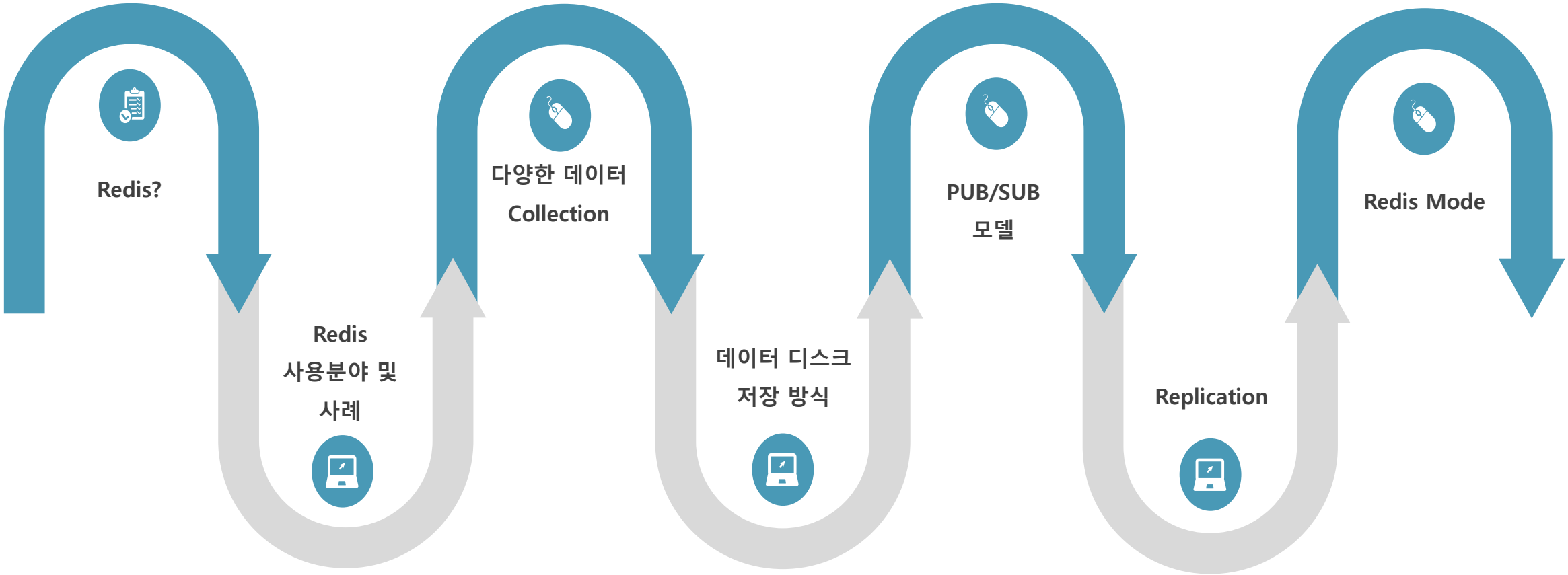
바지개발

Run Redis in Docker

Good Look in your developer life

Run Redis in Docker

목차



Remote Dictionary Server

- 데이터 저장소로 디스크가 아닌 메모리를 사용 → 초고속 데이터 저장소
- key-value 저장 방식 + collection 제공
- 영속적인 데이터 보존 기능 제공
- 메인 데이터베이스보다는 보조적인 수단인 Cache, Messgae broker용도로 많이 사용
- 전 세계 데이터베이스에서 6위

Run Redis in Docker – Redis 사용분야 및 사례

01 디지털 트윈

수많은 센서데이터를 실시간으로 받아 처리
시뮬레이션 결과를 현실 사물에 반영

02 채팅/메신저/챗봇

대화 내용을 임시로 레디스에 저장하고,
사용자 상태정보를 레디스 pub/sub으로 구현

03 사물인터넷

수많은 세션들로부터 데이터를 받아
처리하는데 레디스를 사용

04 세션 스토어

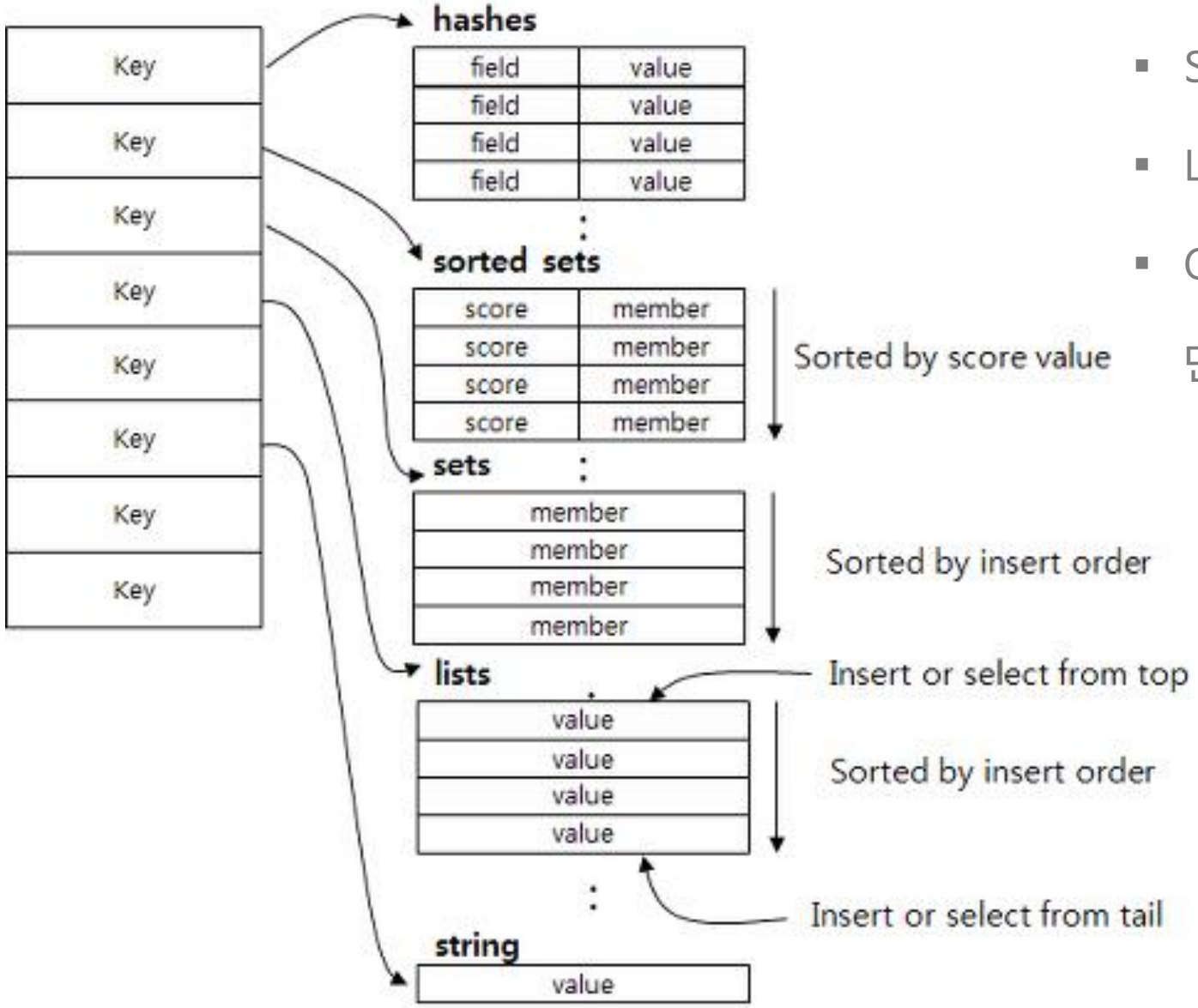
레디스가 가장 광범위하게 사용되는 곳은
데이터 저장 용도

기업 적용 사례

- 트위터 : 초당 30만 트윗을 처리할 수 있도록 구축, Timeline Cache, List를 수정/개선해서 사용
- 라인 : 앞 단 Queue 용도로 레디스를 사용
- 카카오톡 : cache 영역에서 사용

웨이보, 핀터레스트 등...

Run Redis in Docker – 다양한 데이터 Collection



- String : 일대일 관계
- Lists, Sets, Sorted Sets, Hashes : 일대다 관계
- Collection을 잘 이용함으로 여러가지 개발 시간을 단축시키고, 문제를 줄여줄 수 있음

Run Redis in Docker – 다양한 데이터 Collection

String : 문자열 <key, value>

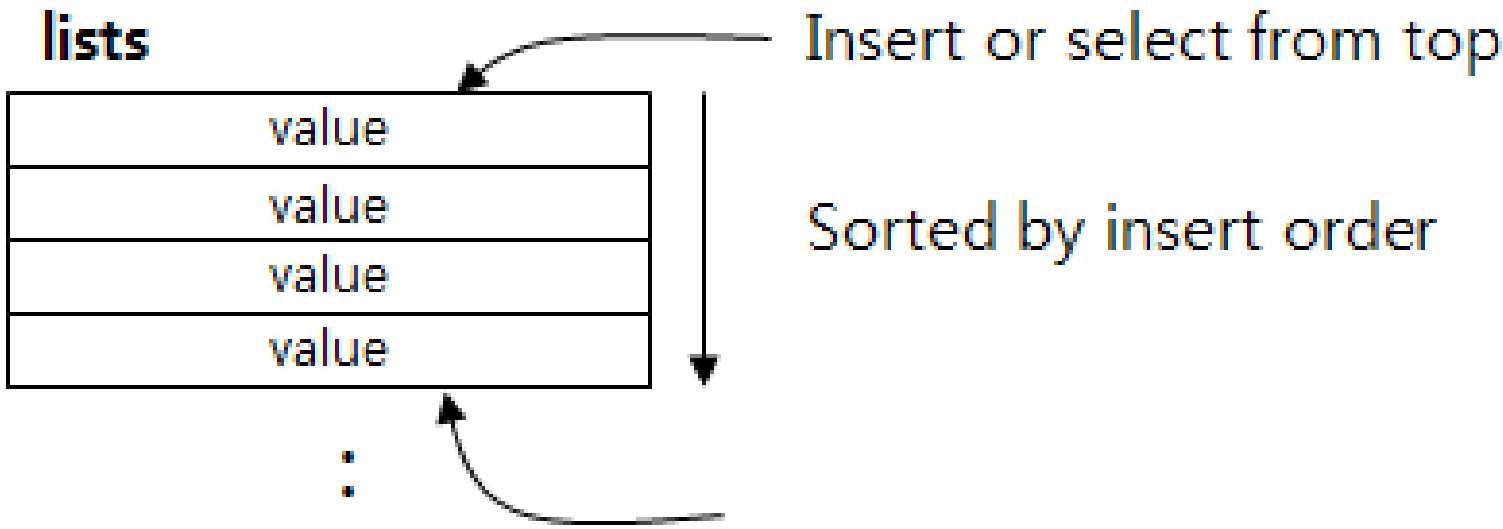
- key와 value는 일대일 관계, 모두 최대 길이는 512MBkey-value 저장 방식 + collection 제공

Key	Value
Key	Value
Key	Value
Key	Value
Key	Value

Run Redis in Docker – 다양한 데이터 Collection

Lists : 리스트 <key, value[]>

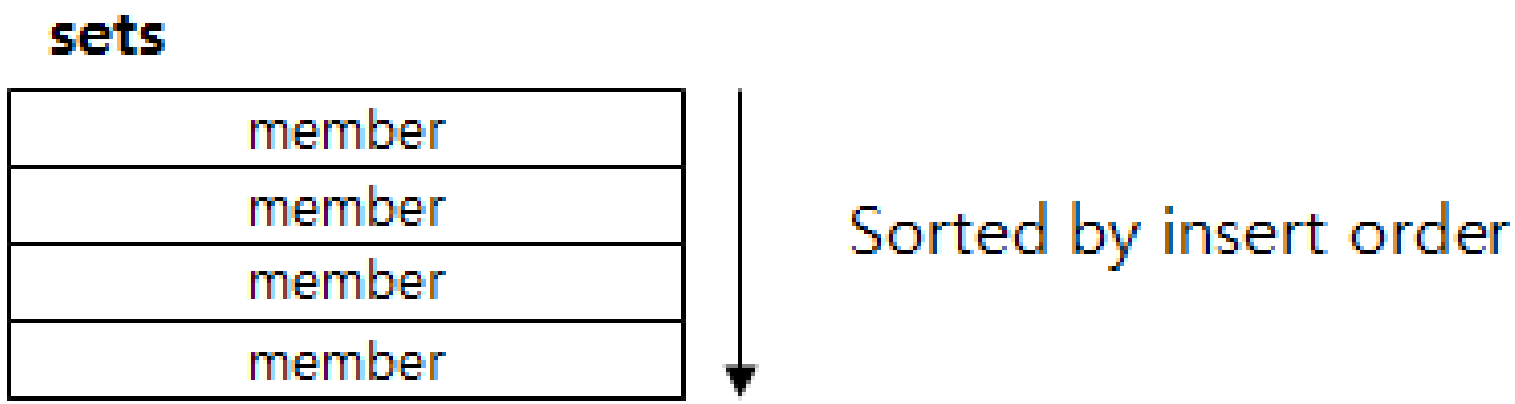
- 데이터를 순차적으로 저장/처리하는데 사용
- 데이터 값의 중복을 허용
- 큐, 스택으로 사용



Run Redis in Docker – 다양한 데이터 Collection

Sets : 집합 <key, Set<value>>

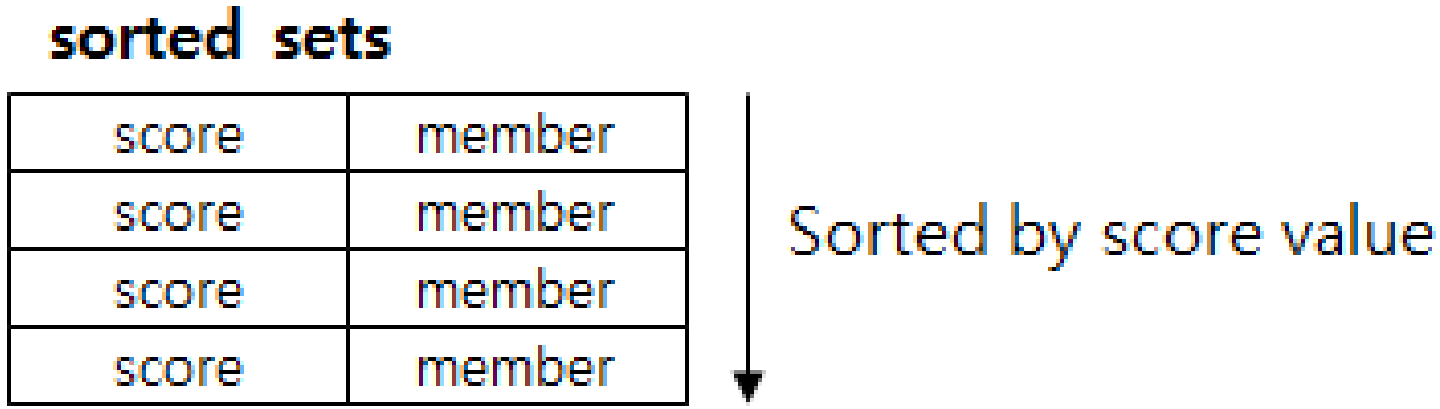
- 멤버(데이터) 중복을 허용하지 않음
- 집합의 성격을 갖는 데이터에 사용
- 집합연산(합집합, 교집합, 차집합)을 제공



Run Redis in Docker – 다양한 데이터 Collection

Sorted Sets : 정렬이 되는 집합 <key, Set<value(with score)>>

- key 하나에 여러개의 score와 value로 구성
- score로 member의 순서를 정렬, 정렬된 데이터가 필요한 경우 사용
- member의 중복을 허용하지 않음
- 합집합, 교집합 연산을 할 수 있고, score를 이용한 연산이 제공



Run Redis in Docker – 다양한 데이터 Collection

Hash : 해시 <key, <field, value>>

- key 하나에 여러개의 field와 value로 구성
- value의 이름으로 구분할 수 있도록 field name이 제공
- RDB테이블과 유사함. Hash Key는 table 의 PK, field는 column
- Hash의 field수는 40억개로 무제한

hashes

field	value
field	value
field	value
field	value

Run Redis in Docker – 데이터 디스크 저장 방식

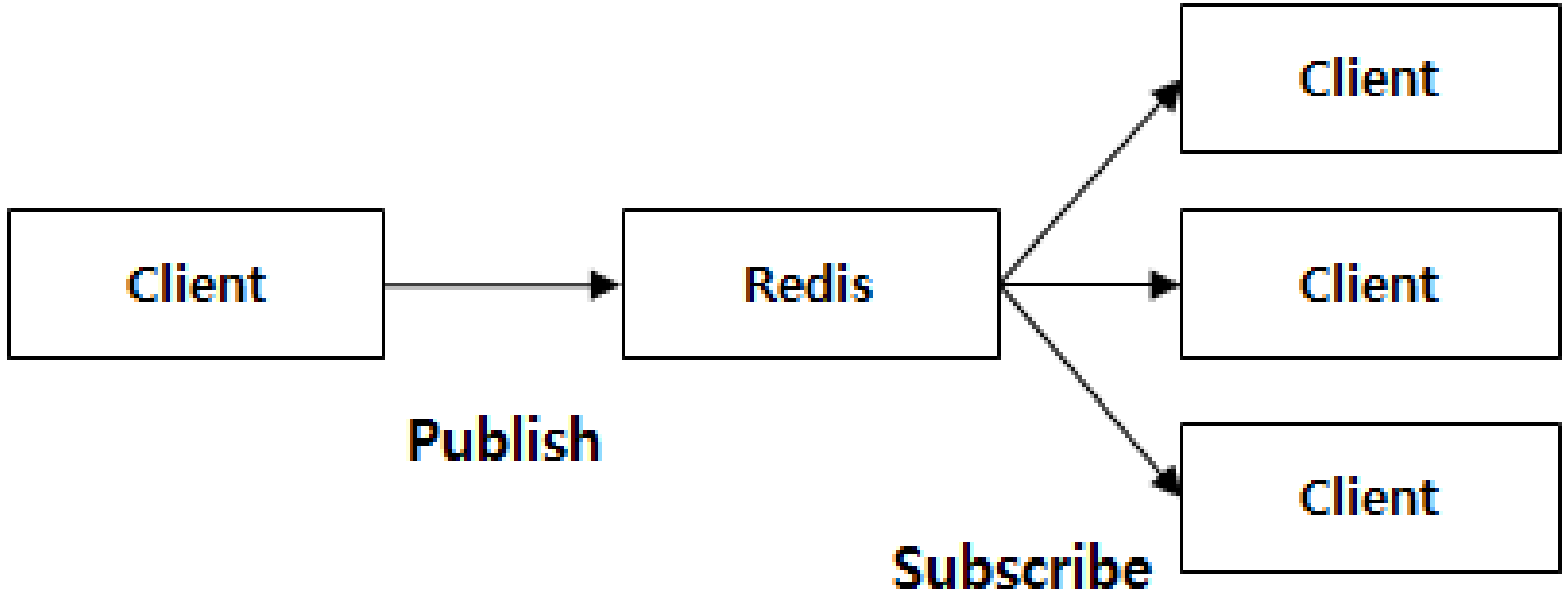
	RDB(snapshot, 백업)	AOF(Append Only File, 보관)
형태	• 순간적으로 메모리에 있는 내용을 DISK 전체에 옮겨 담는 방식 • SAVE와 BGSAVE 두 가지 방식 – SAVE는 blocking 방식으로 순간적으로 Redis의 모든 동작을 정지시키고, 그때의 snapshot을 disk에 저 – 장BGSAVE는 non-blocking 방식으로 별도의 process를 띄운 후, 명령어 수행 당시의 메모리 snapshot을 disk에 저장하며, 저장 순간에 Redis는 동작을 멈추지 않고 정상적으로 동작	• 모든 쓰기, 업데이트 연산 자체를 모두 로그 파일에 기록하는 형태 • 서버가 재시작 될 때 기록된 write, update 연산을 순차적으로 재실행하여 데이터를 복구 • 기본적으로 non-blocking call
장점	서버 restart시 snapshot만 load하면 되므로 restart 시간이 빠름	• 현재 시점까지의 로그를 기록 • Log file에 기록만 하기 때문에 log write 속도가 빠름
단점	• snapshot을 추출하는데 시간이 오래 걸림 • snapshot 추출된 후 서버가 down되면 snapshot 추출 이후 데이터는 유실	• 모든 쓰기, 업데이트 연산을 로그로 남기기 때문에 RDB방식보다 데이터 량이 큼 • 복구 시 모든 연산을 재실행하기 때문에 재기동 속도가 느림

Run Redis in Docker – 데이터를 디스크에 저장하는 방식

권장 저장 방식
RDB + AOF RDB와 AOF 장단점을 서로 상쇄하기 위해 두가지 방식을 같이 쓸 것을 권장함 1. 주기적으로 snapshot으로 백업 + 다음 snapshot까지 저장을 AOF로 수행 2. 서버 재기동시 백업된 snapshot을 reload하고 소량의 로그만 재실행 이 방식은 AOF 로그만 replay하면 되기 때문에, restart 시간을 절약하고 데이터의 유실을 방지함

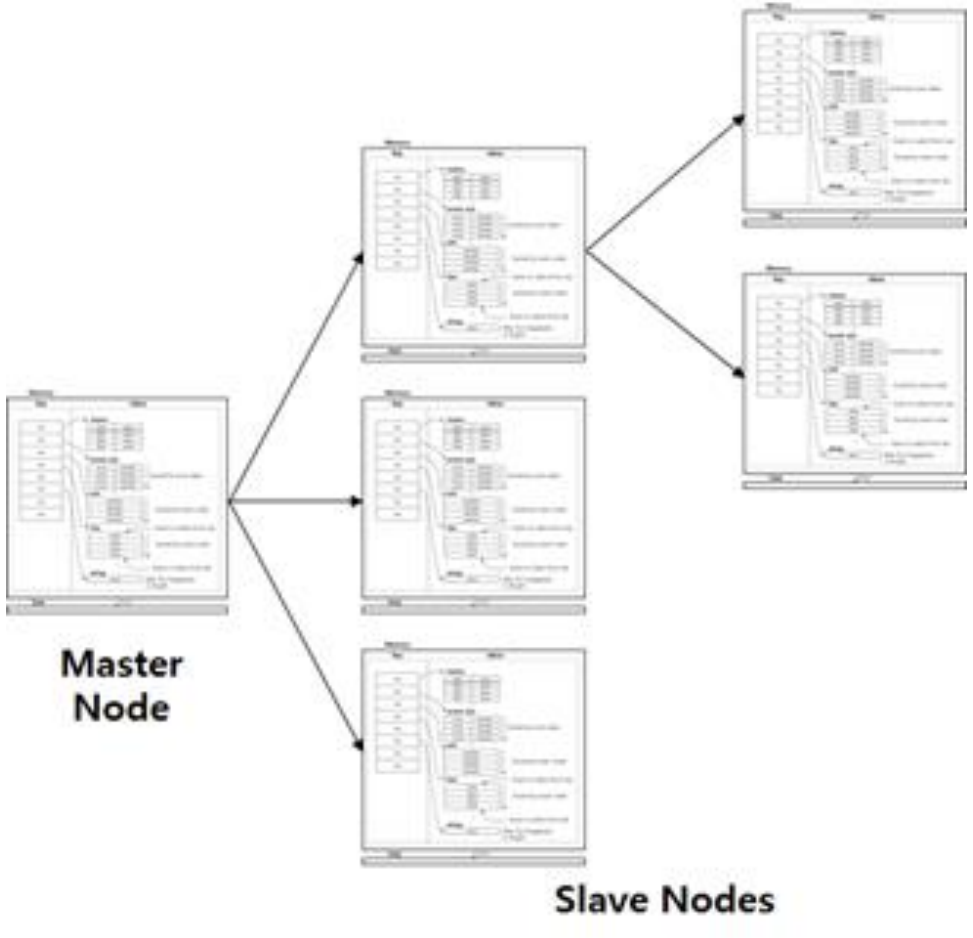
Run Redis in Docker – PUB/SUB 모델

- 하나의 Client가 메세지를 Publish하면, 이 Topic에 연결되어 있는 다수의 클라이언트가 메세지를 받을 수 있는 구조
- 1:1 형태의 Queue 뿐만 아니라 1:N 형태의 Publish/Subscribe 메세징도 지원.
- 일반적인 Pub/Sub 시스템의 경우 Subscribe 하는 하나의 Topic에서만 Subscribe하는데 반해서, Redis에서는 pattern matching을 통해서 다수의 Topic에서 message 를 subscribe할 수 있음



Run Redis in Docker – Replication

- Master/Slave 구조의 Replication(복제)를 지원. 즉 Master 노드에 write된 내용을 Slave 노드가 복제하는 Non-Blocking 구조
- 1개의 master node는 n개의 slave node를 가질 수 있으며, 각 slave node도 그에 대한 slave node를 또 가질 수 있음
- Master-Slave 구조에서 복제 연결이 되어있는 동안 Master 노드의 데이터는 실시간으로 Slave 노드에 복사
- 동시접속자수나 처리 속도를 증가시킬 수 있음



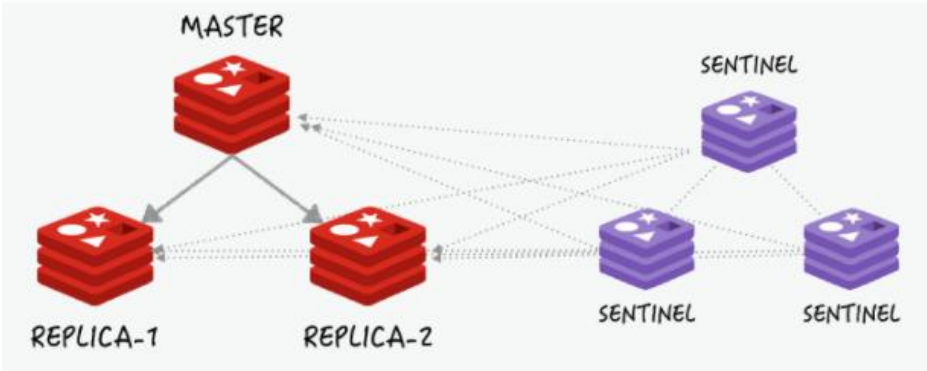
Run Redis in Docker – Redis Mode

Standalone Mode

- 하나의 Redis 인스턴스로 서비스를 하는 방식

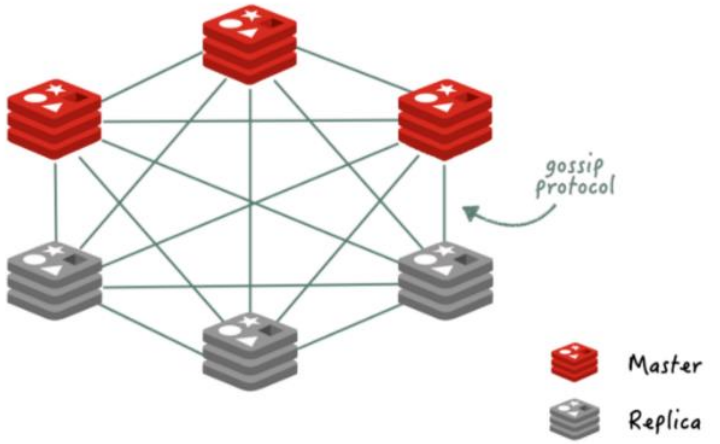
Sentinel Mode

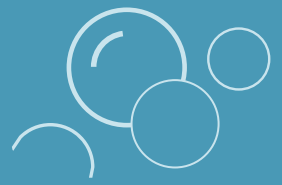
- Redis의 Master/Slave를 모니터링하는 서버
- Sentinel은 Master와 Slave 노드를 계속 모니터링하며 장애상황에는 Slave 노드를 Master 노드로 승격시키기 위해 자동 Failover를 진행
- 정상적인 기능을 위해서는 적어도 세 개 이상의 홀 수의 Sentinel 인스턴스가 필요하고, 세 대의 Sentinel 노드 중 과반수 이상이 동의해야만 Failover를 시작



Cluster Mode

- 데이터셋을 여러 노드에 자동으로 분산하는 확장성 및 고성능의 특징
- 일부 노드가 종료되어도 계속 사용 가능한 고가용성의 특징
- 클러스터를 사용하기 위해서는 최소 세 개의 Master 노드가 필요, 모든 master와 replica 노드는 서로 연결되어 있음
- Replica 노드는 Master 노드의 정확한 복제본을 가지고 있음





Thank you

이제 직접 설치 실습으로~

Good Look in your developer life