Redis 빠른 참조

문자열, 리스트, 집합, 해시, Pub/Sub, 영속성

연결

CLI

```
redis-cli
redis-cli -h 127.0.0.1 -p 6379
redis-cli -a password -n 2
redis-cli --tls -u rediss://user:pass@host:6380
```

드라이버 연결 (Python)

```
import redis
r = redis.Redis(host='localhost', port=6379, db=0)
r.set('key', 'value')
print(r.get('key'))
```

서버 정보

```
PING           -- returns PONG
INFO server    -- server details
INFO memory    -- memory usage
DBSIZE         -- number of keys in current db
```

문자열

기본 작업

```
SET name "Alice"
GET name
SET counter 100
MSET a 1 b 2 c 3
MGET a b c
```

숫자 연산

```
INCR counter        -- 101
INCRBY counter 10   -- 111
DECR counter        -- 110
DECRBY counter 5    -- 105
INCRBYFLOAT price 2.5
```

문자열 명령어

SET key val	문자열 값 설정
GET key	문자열 값 가져오기
SETNX key val	키가 없을 때만 설정
SETEX key sec val	만료 시간(초)과 함께 설정
APPEND key val	기존 값에 추가
STRLEN key	문자열 값 길이

리스트

리스트 작업

```
LPUSH queue "first"
RPUSH queue "last"
LRANGE queue 0 -1    -- all elements
LPOP queue
RPOP queue
```

리스트 명령어

LPUSH / RPUSH	리스트 왼쪽 / 오른쪽에 추가
LPOP / RPOP	왼쪽 / 오른쪽에서 꺼내기
LRANGE key start stop	원소 범위 가져오기
LLEN key	리스트 길이
LINDEX key idx	인덱스의 원소
LREM key count val	val의 count개 제거
BLPOP key timeout	블로킹 pop (큐용)

집합 및 정렬된 집합

집합 작업

```
SADD tags "python" "redis" "docker"
SMEMBERS tags
SISMEMBER tags "python"   -- 1 (true)
SREM tags "docker"
SCARD tags                -- count
```

집합 연산

```
SUNION set1 set2       -- union
SINTER set1 set2       -- intersection
SDIFF set1 set2        -- difference
```

정렬된 집합 작업

```
ZADD leaderboard 100 "Alice" 85 "Bob"
ZRANGE leaderboard 0 -1 WITHSCORES
ZREVRANGE leaderboard 0 2
ZSCORE leaderboard "Alice"
ZRANK leaderboard "Alice"    -- 0-based rank
```

정렬된 집합 명령어

ZADD key score member	점수와 함께 멤버 추가
ZRANGE key start stop	순위 범위 (낮음→높음)
ZREVRANGE key start stop	순위 범위 (높음→낮음)
ZINCRBY key incr member	멤버 점수 증가
ZRANGEBYSCORE key min max	점수 값으로 범위
ZCARD key	멤버 수

해시

해시 작업

```
HSET user:1 name "Alice" age 30
HGET user:1 name
HGETALL user:1
HMSET user:2 name "Bob" age 25
HMGET user:1 name age
```

해시 명령어

HSET key field val	해시 필드 설정
HGET key field	해시 필드 가져오기
HGETALL key	모든 필드와 값 가져오기
HDEL key field	해시 필드 삭제
HEXISTS key field	필드 존재 여부 확인
HINCRBY key field n	필드 값 증가
HKEYS key	모든 필드 이름
HLEN key	필드 수

키 및 만료

키 명령어

KEYS pattern	패턴에 맞는 키 찾기 (느림)
SCAN cursor MATCH pat	키를 점진적으로 순회 (안전)
EXISTS key	키 존재 여부 확인
DEL key	키 삭제
TYPE key	키의 데이터 타입 가져오기
RENAME key newkey	키 이름 변경

만료 명령어

```
EXPIRE key 3600       -- expire in 1 hour
PEXPIRE key 5000      -- expire in 5000 ms
TTL key               -- seconds until expiry
PTTL key              -- ms until expiry
PERSIST key           -- remove expiry
```

키 패턴

```
SET session:abc123 "data" EX 1800
-- EX = seconds, PX = milliseconds
-- NX = only if not exists
-- XX = only if exists
SET lock:order42 "owner" NX EX 10
```

Pub/Sub

기본 Pub/Sub

```
-- Subscriber (terminal 1)
SUBSCRIBE news alerts

-- Publisher (terminal 2)
PUBLISH news "Breaking: Redis 8 released"
```

패턴 구독

```
PSUBSCRIBE news.*
-- matches news.tech, news.sports, etc.
```

Pub/Sub 명령어

SUBSCRIBE channel	채널에서 메시지 수신 대기
PUBLISH channel msg	채널에 메시지 전송
PSUBSCRIBE pattern	패턴 구독
UNSUBSCRIBE channel	수신 중지
PUBSUB CHANNELS	활성 채널 나열

트랜잭션

MULTI / EXEC

```
MULTI
SET balance:1 900
SET balance:2 1100
EXEC        -- executes atomically
```

낙관적 잠금

```
WATCH balance:1
val = GET balance:1   -- read current
MULTI
SET balance:1 (val - 100)
EXEC
-- EXEC returns nil if balance:1 changed
```

트랜잭션 명령어

MULTI	트랜잭션 블록 시작
EXEC	대기 중인 명령 실행
DISCARD	대기 중인 명령 폐기
WATCH key	키 변경 감시 (낙관적 잠금)
UNWATCH	모든 감시 키 해제

영속성

RDB 스냅샷

```
SAVE             -- synchronous snapshot
BGSAVE           -- background snapshot
LASTSAVE         -- timestamp of last save
```

AOF (추가 전용 파일)

appendonly yes	redis.conf에서 AOF 활성화
appendfsync always	매 쓰기마다 Fsync (가장 안전, 가장 느림)
appendfsync everysec	초당 Fsync (권장)
appendfsync no	OS에게 맡김 (가장 빠름, 위험)

Redis 빠른 참조

영속성 명령어

```
CONFIG GET save
CONFIG SET save "900 1 300 10"
-- snapshot if 1 change in 900s or 10 in 300s
BGREWRITEAOF    -- rewrite AOF in background
```

일반 패턴

분산 잠금

```
SET lock:resource "owner-id" NX EX 30
-- NX = acquire only if not held
-- EX 30 = auto-release after 30s
DEL lock:resource  -- explicit release
```

속도 제한기

```
key = "rate:user:42"
INCR key
EXPIRE key 60   -- 60-second window
-- reject if GET key > max_requests
```

캐싱 패턴

```
val = GET "cache:user:1"
if val is nil:
  val = fetch_from_db(1)
  SET "cache:user:1" val EX 300
```

세션 저장

```
HSET sess:abc uid 42 role "admin"
EXPIRE sess:abc 1800   -- 30 min TTL
HGETALL sess:abc
```