Redis 快速参考

字符串、列表、集合、Hash、Pub/Sub 与持久化

连接

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
```

驱动连接（Python）

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

服务器信息

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

字符串

基本操作

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

数值操作

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

字符串命令

SET key val	设置字符串值
GET key	获取字符串值
SETNX key val	仅当 key 不存在时设置
SETEX key sec val	设置值并指定过期秒数
APPEND key val	追加到已有值
STRLEN key	字符串值的长度

列表

列表操作

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

列表命令

LPUSH / RPUSH	从左 / 右插入元素
LPOP / RPOP	从左 / 右弹出元素
LRANGE key start stop	获取指定范围的元素
LLEN key	列表长度
LINDEX key idx	获取指定索引的元素
LREM key count val	删除 count 个值为 val 的元素
BLPOP key timeout	阻塞弹出（适用于队列）

集合与有序集合

集合操作

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

集合运算

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

有序集合操作

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

有序集合命令

ZADD key score member	添加带分数的成员
ZRANGE key start stop	按排名从低到高获取范围
ZREVRANGE key start stop	按排名从高到低获取范围
ZINCRBY key incr member	增加成员分数
ZRANGEBYSCORE key min max	按分数范围获取成员
ZCARD key	成员数量

Hash

Hash 操作

```
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
```

Hash 命令

HSET key field val	设置 Hash 字段
HGET key field	获取 Hash 字段
HGETALL key	获取所有字段和值
HDEL key field	删除 Hash 字段
HEXISTS key field	检查字段是否存在
HINCRBY key field n	对字段值加 n
HKEYS key	所有字段名
HLEN key	字段数量

Key 与过期

Key 命令

KEYS pattern	按模式查找 key（速度慢）
SCAN cursor MATCH pat	增量迭代 key（安全）
EXISTS key	检查 key 是否存在
DEL key	删除 key
TYPE key	获取 key 的数据类型
RENAME key newkey	重命名 key

过期命令

```
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
```

Key 模式

```
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

基本 Pub/Sub

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

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

模式订阅

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

Pub/Sub 命令

SUBSCRIBE channel	监听频道消息
PUBLISH channel msg	向频道发布消息
PSUBSCRIBE pattern	按模式订阅
UNSUBSCRIBE channel	取消订阅
PUBSUB CHANNELS	列出活跃频道

事务

MULTI / EXEC

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

乐观锁

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

事务命令

MULTI	开启事务块
EXEC	执行队列中的命令
DISCARD	丢弃队列中的命令
WATCH key	监视 key 的变化（乐观锁）
UNWATCH	取消所有 WATCH

持久化

RDB 快照

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

AOF（追加只写文件）

appendonly yes	在 redis.conf 中启用 AOF
appendfsync always	每次写入都同步（最安全，最慢）
appendfsync everysec	每秒同步一次（推荐）
appendfsync no	由 OS 决定（最快，风险最高）

Redis 快速参考

持久化命令

```
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
```

常用模式

分布式锁

```
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
```

限流器

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

缓存模式

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

Session 存储

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