Redis クイックリファレンス

文字列、リスト、セット、ハッシュ、Pub/Sub、永続化

接続

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
```

ドライバー接続（Python）

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

サーバー情報

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

文字列

基本操作

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

数値操作

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

文字列コマンド

SET key val	文字列値を設定
GET key	文字列値を取得
SETNX key val	キーが存在しない場合のみ設定
SETEX key sec val	秒単位の有効期限付きで設定
APPEND key val	既存の値に追記
STRLEN key	文字列値の長さ

リスト

リスト操作

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

リストコマンド

LPUSH / RPUSH	リストの左/右に追加
LPOP / RPOP	左/右からポップ
LRANGE key start stop	要素の範囲を取得
LLEN key	リストの長さ
LINDEX key idx	インデックスの要素
LREM key count val	val の count 件を削除
BLPOP key timeout	ブロッキングポップ（キュー用）

セットとソート済みセット

セット操作

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

セット演算

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

ソート済みセット操作

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

ソート済みセットコマンド

ZADD key score member	スコア付きメンバーを追加
ZRANGE key start stop	ランク範囲（低→高）
ZREVRANGE key start stop	ランク範囲（高→低）
ZINCRBY key incr member	メンバーのスコアをインクリメント
ZRANGEBYSCORE key min max	スコア値による範囲
ZCARD key	メンバー数

ハッシュ

ハッシュ操作

```
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
```

ハッシュコマンド

HSET key field val	ハッシュフィールドを設定
HGET key field	ハッシュフィールドを取得
HGETALL key	全フィールドと値を取得
HDEL key field	ハッシュフィールドを削除
HEXISTS key field	フィールドの存在確認
HINCRBY key field n	フィールドの値をインクリメント
HKEYS key	全フィールド名
HLEN key	フィールド数

キーと有効期限

キーコマンド

KEYS pattern	パターンに一致するキーを検索（低速）
SCAN cursor MATCH pat	キーをインクリメンタルに反復（安全）
EXISTS key	キーが存在するか確認
DEL key	キーを削除
TYPE key	キーのデータ型を取得
RENAME key newkey	キーをリネーム

有効期限コマンド

```
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
```

キーパターン

```
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

基本的な Pub/Sub

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

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

パターンサブスクライブ

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

Pub/Sub コマンド

SUBSCRIBE channel	チャンネルのメッセージを受信
PUBLISH channel msg	チャンネルにメッセージを送信
PSUBSCRIBE pattern	パターンでサブスクライブ
UNSUBSCRIBE channel	受信を停止
PUBSUB CHANNELS	アクティブなチャンネルを一覧表示

トランザクション

MULTI / EXEC

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

楽観的ロック

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

トランザクションコマンド

MULTI	トランザクションブロックを開始
EXEC	キューに入ったコマンドを実行
DISCARD	キューに入ったコマンドを破棄
WATCH key	キーの変更を監視（楽観的ロック）
UNWATCH	全監視キーを解除

永続化

RDB スナップショット

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

AOF（追記専用ファイル）

appendonly yes	redis.conf で AOF を有効化
appendfsync always	毎書き込みで fsync（最も安全、最も低速）
appendfsync everysec	1 秒ごとに fsync（推奨）
appendfsync no	OS に任せる（最速、最もリスクあり）

Redis クイックリファレンス

永続化コマンド

```
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
```

よくあるパターン

分散ロック

```
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
```

レートリミッター

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

キャッシュパターン

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

セッションストレージ

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