Referência Rápida de Redis

Strings, listas, conjuntos, hashes, pub/sub, persistência

Conexão

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
```

Conexão por Driver (Python)

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

Informações do Servidor

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

Strings

Operações Básicas

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

Operações Numéricas

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

Comandos de String

`SET key val`	Definir valor de string
`GET key`	Obter valor de string
`SETNX key val`	Definir apenas se a chave não existe
`SETEX key sec val`	Definir com expiração em segundos
`APPEND key val`	Acrescentar ao valor existente
`STRLEN key`	Comprimento do valor de string

Listas

Operações de Lista

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

Comandos de Lista

`LPUSH / RPUSH`	Empurrar para esquerda / direita da lista
`LPOP / RPOP`	Retirar da esquerda / direita
`LRANGE key start stop`	Obter intervalo de elementos
`LLEN key`	Comprimento da lista
`LINDEX key idx`	Elemento no índice
`LREM key count val`	Remover count ocorrências de val
`BLPOP key timeout`	Pop bloqueante (para filas)

Conjuntos e Conjuntos Ordenados

Operações de Conjunto

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

Operações Matemáticas de Conjunto

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

Operações de Conjunto Ordenado

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

Comandos de Conjunto Ordenado

`ZADD key score member`	Adicionar membro com pontuação
`ZRANGE key start stop`	Intervalo por rank (menor para maior)
`ZREVRANGE key start stop`	Intervalo por rank (maior para menor)
`ZINCRBY key incr member`	Incrementar pontuação do membro
`ZRANGEBYSCORE key min max`	Intervalo por valor de pontuação
`ZCARD key`	Número de membros

Hashes

Operações de Hash

```
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
```

Comandos de Hash

`HSET key field val`	Definir campo do hash
`HGET key field`	Obter campo do hash
`HGETALL key`	Obter todos os campos e valores
`HDEL key field`	Excluir campo do hash
`HEXISTS key field`	Verificar existência do campo
`HINCRBY key field n`	Incrementar valor do campo
`HKEYS key`	Todos os nomes de campos
`HLEN key`	Número de campos

Chaves e Expiração

Comandos de Chave

`KEYS pattern`	Encontrar chaves por padrão (lento)
`SCAN cursor MATCH pat`	Iterar chaves incrementalmente (seguro)
`EXISTS key`	Verificar se chave existe
`DEL key`	Excluir chave
`TYPE key`	Obter tipo de dado da chave
`RENAME key newkey`	Renomear uma chave

Comandos de Expiração

```
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
```

Padrões de Chave

```
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

Pub/Sub Básico

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

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

Assinatura por Padrão

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

Comandos Pub/Sub

`SUBSCRIBE channel`	Ouvir mensagens no canal
`PUBLISH channel msg`	Enviar mensagem ao canal
`PSUBSCRIBE pattern`	Assinar por padrão
`UNSUBSCRIBE channel`	Parar de ouvir
`PUBSUB CHANNELS`	Listar canais ativos

Transações

MULTI / EXEC

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

Bloqueio Otimista

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

Comandos de Transação

`MULTI`	Iniciar bloco de transação
`EXEC`	Executar comandos na fila
`DISCARD`	Descartar comandos na fila
`WATCH key`	Observar chave para mudanças (bloqueio otimista)
`UNWATCH`	Esquecer todas as chaves observadas

Persistência

Snapshots RDB

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

AOF (Append Only File)

`appendonly yes`	Habilitar AOF no redis.conf
`appendfsync always`	Fsync a cada escrita (mais seguro, mais lento)
`appendfsync everysec`	Fsync uma vez por segundo (recomendado)
`appendfsync no`	Deixar o SO decidir (mais rápido, mais arriscado)

Referência Rápida de Redis

Comandos de Persistência

```
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
```

Padrões Comuns

Lock Distribuído

```
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
```

Limitador de Taxa

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

Padrão de Cache

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

Armazenamento de Sessão

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