Redis Quick Reference

Strings, lists, sets, hashes, pub/sub, persistence

Connecting

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
```

Driver Connection (Python)

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

Server Info

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

Strings

Basic Operations

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

Numeric Operations

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

String Commands

`SET key val`	Set string value
`GET key`	Get string value
`SETNX key val`	Set only if key does not exist
`SETEX key sec val`	Set with expiry in seconds
`APPEND key val`	Append to existing value
`STRLEN key`	Length of string value

Lists

List Operations

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

List Commands

`LPUSH / RPUSH`	Push to left / right of list
`LPOP / RPOP`	Pop from left / right
`LRANGE key start stop`	Get range of elements
`LLEN key`	Length of list
`LINDEX key idx`	Element at index
`LREM key count val`	Remove count occurrences of val
`BLPOP key timeout`	Blocking pop (for queues)

Sets & Sorted Sets

Set Operations

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

Set Math

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

Sorted Set Operations

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

Sorted Set Commands

`ZADD key score member`	Add member with score
`ZRANGE key start stop`	Range by rank (low to high)
`ZREVRANGE key start stop`	Range by rank (high to low)
`ZINCRBY key incr member`	Increment member score
`ZRANGEBYSCORE key min max`	Range by score value
`ZCARD key`	Number of members

Hashes

Hash Operations

```
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 Commands

`HSET key field val`	Set hash field
`HGET key field`	Get hash field
`HGETALL key`	Get all fields and values
`HDEL key field`	Delete hash field
`HEXISTS key field`	Check field existence
`HINCRBY key field n`	Increment field value
`HKEYS key`	All field names
`HLEN key`	Number of fields

Keys & Expiry

Key Commands

`KEYS pattern`	Find keys matching pattern (slow)
`SCAN cursor MATCH pat`	Iterate keys incrementally (safe)
`EXISTS key`	Check if key exists
`DEL key`	Delete key
`TYPE key`	Get key's data type
`RENAME key newkey`	Rename a key

Expiry Commands

```
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 Patterns

```
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

Basic Pub/Sub

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

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

Pattern Subscribe

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

Pub/Sub Commands

`SUBSCRIBE channel`	Listen for messages on channel
`PUBLISH channel msg`	Send message to channel
`PSUBSCRIBE pattern`	Subscribe to pattern
`UNSUBSCRIBE channel`	Stop listening
`PUBSUB CHANNELS`	List active channels

Transactions

MULTI / EXEC

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

Optimistic Locking

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

Transaction Commands

`MULTI`	Start transaction block
`EXEC`	Execute queued commands
`DISCARD`	Discard queued commands
`WATCH key`	Watch key for changes (optimistic lock)
`UNWATCH`	Forget all watched keys

Persistence

RDB Snapshots

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

AOF (Append Only File)

`appendonly yes`	Enable AOF in redis.conf
`appendfsync always`	Fsync every write (safest, slowest)
`appendfsync everysec`	Fsync once per second (recommended)
`appendfsync no`	Let OS decide (fastest, riskiest)

Redis Quick Reference

Persistence Commands

```
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
```

Common Patterns

Distributed Lock

```
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
```

Rate Limiter

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

Caching Pattern

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

Session Storage

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