

Redis Cheatsheet

Essential commands for in-memory data structure operations

This cheatsheet provides a quick reference to fundamental Redis operations, syntax, and advanced features. Redis is an open-source, in-memory data structure store used as a database, cache, and message broker. It supports various data structures such as strings, hashes, lists, sets, and more. Ideal for both beginners and experienced developers for efficient data processing and caching.

String Operations Store and manipulate text values	Data Structures Work with lists, sets, and hashes	Key Management Manage keys and expiration
Database Operations Administer Redis instances		Performance Monitoring Track and optimize Redis performance

Redis Installation & Setup

Get Redis up and running on your system and connect to the server.

Docker: `docker run redis` Quickest way to get Redis running locally. <pre># Run Redis in Docker docker run --name my-redis -p 6379:6379 -d redis # Connect to Redis CLI docker exec -it my-redis redis-cli # Run with persistent storage docker run --name redis-persistent -p 6379:6379 -v redis-data:/data -d redis</pre>	Linux: `sudo apt install redis` Install Redis server on Ubuntu/Debian systems. <pre># Install Redis sudo apt update sudo apt install redis-server # Start Redis service sudo systemctl start redis-server # Enable auto-start on boot sudo systemctl enable redis-server # Check status sudo systemctl status redis</pre>	Connect & Test: `redis-cli` Connect to Redis server and verify installation. <pre># Connect to local Redis redis-cli # Test connection redis-cli PING # Connect to remote Redis redis-cli -h hostname -p 6379 -a password # Execute single command redis-cli SET mykey "Hello Redis"</pre>
---	--	---

Basic String Operations

Set & Get: `SET` / `GET`

Store simple values (text, numbers, JSON, etc.)

```
# Set a key-value pair
SET mykey "Hello World"
# Get value by key
GET mykey
# Set with expiration (in seconds)
SET session:123 "user_data" EX 3600
# Set only if key doesn't exist
SET mykey "new_value" NX
```

String Manipulation: `APPEND` / `STRLEN`

Modify and inspect string values.

```
# Append to existing string
APPEND mykey " - Welcome!"
# Get string length
STRLEN mykey
# Get substring
GETRANGE mykey 0 4
# Set substring
SETRANGE mykey 6 "Redis"
```

List Operations

Lists are ordered sequences of strings, useful as queues or stacks.

Add Elements: `LPUSH` / `RPUSH`

Add elements to the left (head) or right (tail) of a list.

```
# Add to head (left)
LPUSH mylist "first"
# Add to tail (right)
RPUSH mylist "last"
# Add multiple elements
LPUSH mylist "item1" "item2" "item3"
```

Remove Elements: `LPOP` / `RPOP`

Remove and return elements from list ends.

```
# Remove from head
LPOP mylist
# Remove from tail
RPOP mylist
# Blocking pop (wait for element)
BLPOP mylist 10
```

Set Operations

Sets are collections of unique, unordered string elements.

Basic Set Operations: `SADD` / `SMEMBERS`

Add unique elements to sets and retrieve all members.

```
# Add elements to set
SADD myset "apple" "banana" "cherry"
# Get all set members
SMEMBERS myset
# Check if element exists
SISMEMBER myset "apple"
# Get set size
SCARD myset
```

Set Modifications: `SREM` / `SPOP`

Remove elements from sets in different ways.

```
# Remove specific elements
SREM myset "banana"
# Remove and return random element
SPOP myset
# Get random element without removing
SRANDMEMBER myset
```

Hash Operations

Hashes store field-value pairs, like mini JSON objects or dictionaries.

Basic Hash Operations: `HSET` / `HGET`

Set and retrieve individual hash fields.

```
# Set hash field
HSET user:123 name "John Doe" age 30
# Get hash field
HGET user:123 name
# Set multiple fields
HMSET user:123 email "john@example.com" city "NYC"
# Get multiple fields
HMGET user:123 name age email
```

Hash Inspection: `HKEYS` / `HVALS`

Examine hash structure and contents.

```
# Get all field names
HKEYS user:123
# Get all values
HVALS user:123
# Get all fields and values
HGETALL user:123
# Get number of fields
HLEN user:123
```

Key Management

Essential commands for managing keys, expiration, and key patterns.

Key Inspection: `KEYS` / `EXISTS`

Find keys using patterns and check existence.

```
# Get all keys (use carefully in production)
KEYS *
# Keys with pattern
KEYS user:*
# Keys ending with pattern
KEYS *-profile
# Single character wildcard
KEYS order?
# Check if key exists
EXISTS mykey
```

Key Information: `TYPE` / `TTL`

Get key metadata and expiration information.

```
# Get key data type
TYPE mykey
# Get time to live (seconds)
TTL mykey
# Get TTL in milliseconds
PTTL mykey
# Remove expiration
PERSIST mykey
```

Sorted Set Operations

Sorted sets combine uniqueness of sets with ordering based on scores.

Basic Operations: `ZADD` / `ZRANGE`

Add scored members and retrieve ranges.

```
# Add members with scores
ZADD leaderboard 100 "player1" 200 "player2"
# Get members by rank (0-based)
ZRANGE leaderboard 0 -1
# Get with scores
ZRANGE leaderboard 0 -1 WITHSCORES
# Get by score range
ZRANGEBYSCORE leaderboard 100 200
```

Sorted Set Info: `ZCARD` / `ZSCORE`

Get information about sorted set members.

```
# Get set size
ZCARD leaderboard
# Get member score
ZSCORE leaderboard "player1"
# Get member rank
ZRANK leaderboard "player1"
# Count members in score range
ZCOUNT leaderboard 100 200
```

Database Management

Administrative commands for managing Redis databases and server operations.

Database Selection: `SELECT` / `FLUSHDB`

Manage multiple databases within Redis.

```
# Select database (0-15 by default)
SELECT 0
# Clear current database
FLUSHDB
# Clear all databases
FLUSHALL
# Get current database size
DBSIZE
```

Server Info: `INFO` / `PING`

Get server statistics and test connectivity.

```
# Test server connection
PING
# Get server information
INFO
# Get specific info section
INFO memory
INFO replication
# Get server time
TIME
```

Performance Monitoring

Monitor Redis performance, track slow commands, and analyze server metrics.

Real-time Monitoring: `MONITOR` / `SLOWLOG`

Track commands and identify performance bottlenecks.

```
# Monitor all commands in real-time
MONITOR
# Get slow query log
SLOWLOG GET 10
# Get slow log length
SLOWLOG LEN
# Reset slow log
SLOWLOG RESET
```

Memory Analysis: `MEMORY USAGE` / `MEMORY STATS`

Analyze memory consumption and optimization.

```
# Get key memory usage
MEMORY USAGE mykey
# Get memory statistics
MEMORY STATS
# Get memory doctor report
MEMORY DOCTOR
# Purge memory
MEMORY PURGE
```

Advanced Features

Explore Redis advanced capabilities including transactions, pub/sub, and scripting.

Transactions: `MULTI` / `EXEC`

Execute multiple commands atomically.

```
# Start transaction
MULTI
SET key1 "value1"
INCR counter
# Execute all commands
EXEC
# Discard transaction
DISCARD
# Watch keys for changes
WATCH mykey
```

Pub/Sub: `PUBLISH` / `SUBSCRIBE`

Implement message passing between clients.

```
# Subscribe to channel
SUBSCRIBE news sports
# Publish message
PUBLISH news "Breaking: Redis 7.0 released!"
# Pattern subscription
PSUBSCRIBE news:*
# Unsubscribe
UNSUBSCRIBE news
```

Data Types Overview

Quick reference for Redis data type capabilities and use cases.

01 Strings: Most versatile type Can store text, numbers, JSON, binary data. Max size: 512MB. Use for: caching, counters, flags. <pre>SET user:123:name "John" GET user:123:name INCR page:views</pre>	02 Lists: Ordered collections Linked lists of strings. Use for: queues, stacks, activity feeds, recent items. <pre>LPUSH queue:jobs "job1" RPOP queue:jobs LRANGE recent:posts 0 9</pre>	03 Sets: Unique collections Unordered collections of unique strings. Use for: tags, unique visitors, relationships. <pre>SADD post:123:tags "redis" "database" SISMEMBER post:123:tags "redis" SINTER user:123:friends user:456:friends</pre>
--	---	--

Redis Configuration Tips

Essential configuration settings for production Redis deployments.

Memory Management

Configure memory limits and eviction policies.

```
# Set memory limit
CONFIG SET maxmemory 2gb
# Set eviction policy
CONFIG SET maxmemory-policy allkeys-lru
# Check memory usage
INFO memory
```

Persistence Settings

Configure data durability options.

```
# Enable AOF
CONFIG SET appendonly yes
# Set save intervals
CONFIG SET save "900 1 300 10 60 10000"
# AOF rewrite settings
CONFIG SET auto-aof-rewrite-percentage 100
```

Security Settings

Basic security configurations for Redis.

```
# Set password
CONFIG SET requirepass mypassword
# Authenticate
AUTH mypassword
# Disable dangerous commands
CONFIG SET rename-command FLUSHALL ""
```

Performance Tuning

Optimize Redis for better performance.

```
# Set timeout
CONFIG SET timeout 300
# TCP keep alive
CONFIG SET tcp-keepalive 60
# Max clients
CONFIG SET maxclients 10000
```

Reference: This cheatsheet covers essential Redis commands and best practices for efficient in-memory data structure operations, caching, and real-time data processing in modern applications.