RÉFÉRENCE RAPIDE REDIS

Chaînes, listes, ensembles, hashes, pub/sub, persistance

Connexion

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
```

Connexion via driver (Python)

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

Informations serveur

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

Chaînes

Opérations de base

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

Opérations numériques

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

Commandes de chaîne

`SET key val`	Définir une valeur de chaîne
`GET key`	Obtenir une valeur de chaîne
`SETNX key val`	Définir uniquement si la clé n'existe pas
`SETEX key sec val`	Définir avec expiration en secondes
`APPEND key val`	Ajouter à la valeur existante
`STRLEN key`	Longueur de la valeur de chaîne

Listes

Opérations sur les listes

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

Commandes de liste

`LPUSH / RPUSH`	Pousser à gauche / droite de la liste
`LPOP / RPOP`	Dépiler à gauche / droite
`LRANGE key start stop`	Obtenir une plage d'éléments
`LLEN key`	Longueur de la liste
`LINDEX key idx`	Élément à l'index
`LREM key count val`	Supprimer count occurrences de val
`BLPOP key timeout`	Pop bloquant (pour les files)

Ensembles et ensembles triés

Opérations sur les ensembles

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

Opérations ensemblistes

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

Opérations sur les ensembles triés

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

Commandes d'ensemble trié

`ZADD key score member`	Ajouter un membre avec score
`ZRANGE key start stop`	Plage par rang (du plus bas au plus haut)
`ZREVRANGE key start stop`	Plage par rang (du plus haut au plus bas)
`ZINCRBY key incr member`	Incrémenter le score du membre
`ZRANGEBYSCORE key min max`	Plage par valeur de score
`ZCARD key`	Nombre de membres

Hashes

Opérations sur les hashes

```
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
```

Commandes de hash

`HSET key field val`	Définir un champ de hash
`HGET key field`	Obtenir un champ de hash
`HGETALL key`	Obtenir tous les champs et valeurs
`HDEL key field`	Supprimer un champ de hash
`HEXISTS key field`	Vérifier l'existence du champ
`HINCRBY key field n`	Incrémenter la valeur du champ
`HKEYS key`	Tous les noms de champs
`HLEN key`	Nombre de champs

Clés et expiration

Commandes de clé

`KEYS pattern`	Trouver les clés correspondant au motif (lent)
`SCAN cursor MATCH pat`	Itérer les clés de façon incrémentale (sûr)
`EXISTS key`	Vérifier si la clé existe
`DEL key`	Supprimer une clé
`TYPE key`	Obtenir le type de données de la clé
`RENAME key newkey`	Renommer une clé

Commandes d'expiration

```
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
```

Motifs de clés

```
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 de base

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

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

Abonnement par motif

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

Commandes Pub/Sub

`SUBSCRIBE channel`	Écouter les messages sur un canal
`PUBLISH channel msg`	Envoyer un message sur un canal
`PSUBSCRIBE pattern`	S'abonner à un motif
`UNSUBSCRIBE channel`	Arrêter d'écouter
`PUBSUB CHANNELS`	Lister les canaux actifs

Transactions

MULTI / EXEC

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

Verrouillage optimiste

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

Commandes de transaction

`MULTI`	Démarrer un bloc de transaction
`EXEC`	Exécuter les commandes en file
`DISCARD`	Annuler les commandes en file
`WATCH key`	Surveiller une clé pour les modifications (verrou optimiste)
`UNWATCH`	Oublier toutes les clés surveillées

Persistance

Instantanés RDB

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

AOF (Append Only File)

`appendonly yes`	Activer AOF dans redis.conf
`appendfsync always`	Fsync à chaque écriture (le plus sûr, le plus lent)
`appendfsync everysec`	Fsync une fois par seconde (recommandé)
`appendfsync no`	Laisser l'OS décider (le plus rapide, le plus risqué)

Commandes de persistance

```
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
```

Motifs courants

Verrou distribué

```
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
```

Limiteur de débit

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

Motif de cache

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

Stockage de session

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