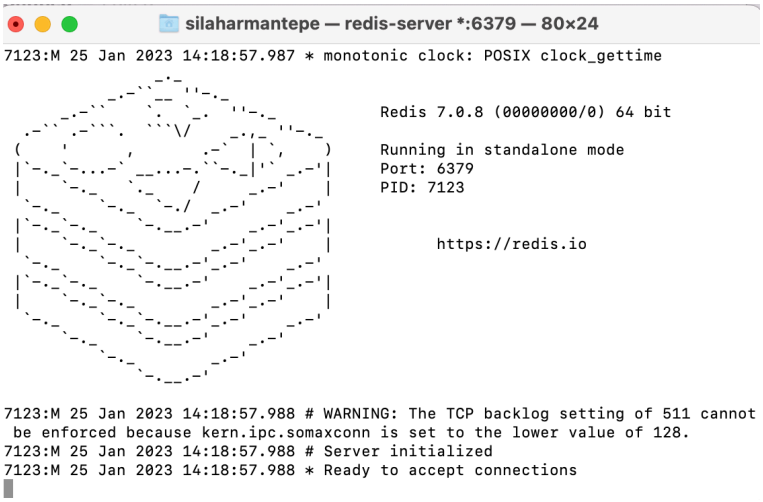
HARMANTEPE Melis

TP 4 - Redis

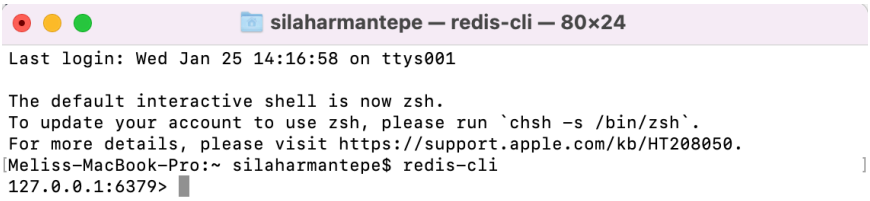
Redis Installation Tutorial on Mac :

From the terminal, run : **brew install redis**

To test your Redis installation, you can run the redis-server executable from the command line : **redis-server**

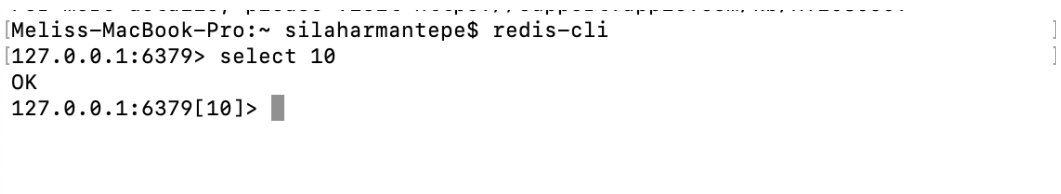


Once Redis is running, you can test it by running in another terminal window : **redis-cli**



Create a database in Redis:
In Redis, a database is essentially a separate space for storing key-value pairs. Each Redis instance can have multiple databases, numbered from 0 to 15 by default. By default, Redis uses database 0, but you can switch to a different database by using the SELECT command followed by the number of the database you want to use.

Let's switch to database number 10:



We can switch back to the default database 0 by running the **SELECT** command followed by 0:

Redis **Lists** are simply lists of strings, sorted by insertion order. You can add elements in Redis lists in the head or the tail of the list.

Here with the command **LPUSH** I create a list of strings names “friends” and I add one by one the strign values “melis”, “maroua” and “mai”.
Then we can display the contents of the list with the command **LRANGE** specifying the range between 0 and 10.

```
127.0.0.1:6379> LPUSH friends melis
(integer) 1
127.0.0.1:6379> LPUSH friends maroua
(integer) 2
127.0.0.1:6379> LPUSH friends mai
(integer) 3
127.0.0.1:6379> LRANGE friends 0 10
1) "mai"
2) "maroua"
3) "melis"
127.0.0.1:6379>
```

LPUSH can push multiple items at once:

```
127.0.0.1:6379> LPUSH shopping_list carrots milk bread
(integer) 3
127.0.0.1:6379> LRANGE shopping_list 0 -1
1) "bread"
2) "milk"
3) "carrots"
127.0.0.1:6379>
```

The HSET command in Redis is used to set the value of a field in a hash stored at a key in the Redis database. A Redis hash is a collection of key-value pairs, where each key is a field, and each value is the value for that field : **HSET key field value**

Where "key" is the name of the key to set, "field" is the name of the field within the hash, and "value" is the value to set for the field.

For example, to set the value "John Doe" for the field "name" in a hash stored at key "user:1", you would run the following command:

HSET user:2 name "John Doe”
HSET user:3 name “Melis"

```
127.0.0.1:6379> HSET user:3 name "Melis"
(integer) 1
```

You can also use the command to set multiple fields at once like this:

HSET user:1 name "sophie larrod" age 12 email "sophielarr@hotmail.com"
HSET user:4 name "Melissa Princess" age 30 email "melissa@example.com"

```
127.0.0.1:6379> HSET user:4 name "Melissa Princess" age 30 email "melissa@exampl
e.com"
(integer) 3
```

HGETALL returns all fields and values of the hash stored at key:

```
127.0.0.1:6379> HGETALL user:1
1) "name"
2) "sophie larrod"
3) "age"
4) "12"
5) "email"
6) "sophielarr@hotmail.com"
127.0.0.1:6379> HGETALL user:2
1) "name"
2) "John Doe"
127.0.0.1:6379> HGETALL user:3
1) "name"
2) "Melis"
127.0.0.1:6379> HGETALL user:4
1) "name"
2) "Melissa Princess"
3) "age"
4) "30"
5) "email"
6) "melissa@example.com"
127.0.0.1:6379>
```

Functionalities and limitations of Redis:

Functionalities:

- In-memory storage: Redis stores data in memory, which makes it extremely fast for read and write operations.
- Data structures: Redis supports a variety of data structures such as strings, hashes, lists, sets, and sorted sets.
- Pub/sub messaging: Redis supports a publish/subscribe messaging pattern, which allows clients to subscribe to channels and receive messages when they are published.
- Lua scripting: Redis supports Lua scripting, which allows you to execute custom scripts on the server to perform complex operations.
- Replication: Redis supports master-slave replication, which allows you to set up a replica of the master server to improve availability and scalability.
- Persistence: Redis supports persistence, which allows you to save the data to disk at certain intervals or under certain conditions.
- Caching: Redis is often used as a caching solution, which allows you to store frequently accessed data in memory for fast retrieval.

Limitations:

- Memory limitations: Redis stores data in memory, so the amount of data that can be stored is limited by the amount of available memory on the server.
- Single-threaded: Redis is single-threaded, which means that it can only process one command at a time. This can become a bottleneck when handling a high number of requests.
- Limited data types: Redis supports a variety of data types, but its functionality is more limited compared to a relational database.
- Limited querying capabilities: Redis doesn't support advanced querying capabilities like JOIN, GROUP BY, and subqueries.
- Limited scalability: Redis is generally used as a cache, it doesn't scale horizontally as well as a distributed database like Cassandra or MongoDB.

It's important to keep in mind that Redis is a NoSQL database, it doesn't have tables, relations, and it doesn't run SQL commands. It's a great solution for caching, messaging, and simple storage, but it's not a replacement for a relational database.