Caching, Performance and OS

Prepared By: Aatiz Ghimire, for Herald Center for AI.

Summer, 2025

Learning Objectives.

- Set up a Redis instance using Play with Docker.
- Interact with Redis using redis-cli.
- Explore Redis data types (Strings, Hashes, Lists, Sets, Sorted Sets).
- Perform basic Redis operations (CRUD, configuration, and more).

Introduction

This workshop introduces Redis, an open-source, in-memory key-value store known for its speed and versatility. We'll use Play with Docker to run Redis in a containerized environment, making it easy to experiment without local setup. By the end, you'll understand Redis's key features, data types, and basic operations.

Prerequisites

- A web browser with access to Play with Docker.
- Basic familiarity with terminal commands.
- No prior Redis or Docker experience required.

1 Setting Up Redis in Play with Docker

Access Play with Docker

1. Go to `labs.play-with-docker.com`.
2. Log in with your Docker Hub account or create a free account.
3. Click "Start" to create a new session, then click "Add New Instance" to get a terminal.

Run a Redis Container

In the Play with Docker terminal, run the following command to start a Redis container:

```
docker run -d --name redis-server -p 6379:6379 redis
```

- `-d`: Runs the container in detached mode.
- `--name redis-server`: Names the container.
- `-p 6379:6379`: Maps port 6379 (Redis default) to the host.
- `redis`: Uses the official Redis Docker image.

Verify Redis is Running

Connect to the Redis container's CLI:

```
docker exec -it redis-server redis-cli
```

At the `redis 127.0.0.1:6379>` prompt, type:

```
PING
```

Expected output: `PONG`

Exercise: Check Redis Connection

- Run the `PING` command and confirm you get `PONG`.
- Try the `INFO` command to view server details, such as `redis_version` and `used_memory`.

2 Exploring Redis Data Types

Redis supports rich data types like Strings, Hashes, Lists, Sets, and Sorted Sets. Let's explore each with hands-on exercises.

2.1 Strings

Redis strings are binary-safe and can store up to 512 MB.

```
SET tutorial "Redis Workshop"
GET tutorial
```

Expected Output: `"Redis Workshop"`

Exercise:

- Set a key `username` with your name as the value.
- Retrieve the value using `GET`.
- Use `STRLEN username` to get the length of the stored string.

2.2 Hashes

Hashes are maps of string fields and values, ideal for representing objects.

```
HMSET user:1 name "Alice" role "Admin" points 100
HGETALL user:1
```

Expected Output:

```
1) "name"
2) "Alice"
3) "role"
4) "Admin"
5) "points"
6) "100"
```

Exercise:

- Create a hash `user:2` with fields `name`, `role`, and `points`.
- Retrieve all fields using `HGETALL`.
- Increment the `points` field by 50 using `HINCRBY user:2 points 50`.

2.3 Lists

Lists are ordered collections of strings, supporting operations like push and pop.

```
LPUSH tutorials "Redis"
LPUSH tutorials "MongoDB"
LPUSH tutorials "MySQL"
LRANGE tutorials 0 -1
```

Expected Output:

```
1) "MySQL"
2) "MongoDB"
3) "Redis"
```

Exercise:

- Create a list `mylist` and add three items using `LPUSH`.
- Retrieve all items using `LRANGE mylist 0 -1`.
- Remove the first item using `LPOP mylist` and check the updated list.

2.4 Sets

Sets are unordered collections of unique strings.

```
SADD techstack "Redis"
SADD techstack "MongoDB"
SADD techstack "MongoDB"
SMEMBERS techstack
```

Expected Output:

```
1) "Redis"
2) "MongoDB"
```

Exercise:

- Create a set `skills` and add three unique skills.
- Try adding a duplicate skill and verify it's not added using `SMEMBERS`.
- Check the set size with `SCARD skills`.

2.5 Sorted Sets

Sorted Sets associate each member with a score for ordering.

```
ZADD leaderboard 100 "Alice"
ZADD leaderboard 90 "Bob"
ZADD leaderboard 95 "Charlie"
ZRANGE leaderboard 0 -1 WITHSCORES
```

Expected Output:

```
1) "Bob"
2) "90"
3) "Charlie"
4) "95"
5) "Alice"
6) "100"
```

Exercise:

- Create a sorted set `scores` with three members and scores.
- Retrieve members in order using `ZRANGE scores 0 -1 WITHSCORES`.

- Increment a member's score using `ZINCRBY scores 10 "member`$_{name}$".

3 Redis Configuration

Redis configuration can be managed via the `redis.conf` file or `CONFIG` commands.

View Configuration

```
CONFIG GET loglevel
```

Expected Output:

```
1) "loglevel"
2) "notice"
```

Set Configuration

```
CONFIG SET loglevel "warning"
CONFIG GET loglevel
```

Expected Output:

```
1) "loglevel"
2) "warning"
```

Exercise:

- Use `CONFIG GET *` to list all configuration settings.
- Change the `maxclients` setting to 500 using `CONFIG SET maxclients 500`.
- Verify the change with `CONFIG GET maxclients`.

4 Basic Redis Operations

4.1 Keys Management

Manage keys with commands like `DEL`, `EXISTS`, and `KEYS`.

```
SET mykey "Hello"
EXISTS mykey
DEL mykey
EXISTS mykey
```

Expected Output:

```
(integer) 1
(integer) 0
```

Exercise:

- Create a key `testkey` with a value.
- Check if it exists using `EXISTS`.
- Delete it using `DEL` and confirm it's gone.

4.2 Transactions

Redis transactions ensure atomic execution of multiple commands.

```
MULTI
SET transactionkey "Test"
INCR counter
EXEC
```

Expected Output:

```
1) OK
2) (integer) 1
```

Exercise:

- Start a transaction with `MULTI`.
- Queue a `SET` and an `INCR` command.
- Execute with `EXEC` and verify results.

4.3 Publish/Subscribe

Redis supports a pub/sub messaging system.

Example (In Two Terminals):

Terminal 1 (Subscriber):

```
SUBSCRIBE news
```

Terminal 2 (Publisher):

```
docker exec -it redis-server redis-cli
PUBLISH news "Breaking News: Redis Workshop!"
```

Exercise:

- Subscribe to a channel `updates` in one terminal.
- Publish two messages to `updates` from another terminal.
- Observe the messages in the subscriber terminal.

5 Redis Backup

Create a Backup

```
SAVE
```

Expected Output: `OK`

Background Save

```
BGSAVE
```

Expected Output: `Background saving started`

Exercise:

- Run `SAVE` to create a backup.
- Run `BGSAVE` and check the Redis directory for `dump.rdb`:

```
docker exec redis-server ls /data
```

6 Redis with Java (Optional)

For those interested in programmatic access, here's a simple Java example using Jedis to connect to the Redis container.

Create a Java File

In the Play with Docker terminal, create a file `RedisJava.java`:

```
nano RedisJava.java
```

Add the Following Code

```java
import redis.clients.jedis.Jedis;

public class RedisJava {
    public static void main(String[] args) {
        Jedis jedis = new Jedis("localhost", 6379);
        System.out.println("Connection to server successfully");
        System.out.println("Server is running: " + jedis.ping());
        jedis.set("workshop", "Redis with Docker");
        System.out.println("Stored string: " + jedis.get("workshop"));
    }
}
```

Run the Java Program

Install Java and Jedis in the container:

```
docker run -it --name java-redis --link redis-server:redis -v $(pwd):/app openjdk:11 bash
curl -L -o jedis.jar https://repo1.maven.org/maven2/redis/clients/jedis/4.4.6/jedis-4.4.6.jar
javac -cp jedis.jar RedisJava.java
java -cp .:jedis.jar RedisJava
```

Expected Output:

```
Connection to server successfully
Server is running: PONG
Stored string: Redis with Docker
```

Exercise:

- Modify the Java code to store and retrieve a list instead of a string.
- Compile and run the updated code.

7 Cleanup

Stop and Remove the Redis Container

```
docker stop redis-server
docker rm redis-server
```

Remove the Java Container (if used)

```
docker stop java-redis
docker rm java-redis
```

8 Additional Resources (Optional)

For further exploration of Redis Enterprise and Redis Stack, you can access a collection of Google Colab-based notebooks provided by Redis Labs. These notebooks highlight various aspects of Redis and are available at the following GitHub repository:

- **Redis-Workshops Repository**: `https://github.com/Redislabs-Solution-Architects/Redis-Workshops`
- **Description**: A collection of Google Colab-based notebooks that demonstrate different features of Redis Enterprise and Redis Stack.
- **License**: MIT

Conclusion

In this workshop, you:

- Set up Redis in a Docker container using Play with Docker.
- Explored Redis data types (Strings, Hashes, Lists, Sets, Sorted Sets).
- Performed configuration, key management, transactions, and pub/sub operations.
- Optionally interacted with Redis using Java.

For further learning, explore Redis documentation at `redis.io` or experiment with advanced features like HyperLogLog and scripting.

The - End