Exercise 1 (Install Redis)

a)
check files
RedisSetup
SshCommands
gradle.build

if its executed, private ip4 and dns are printed to console

b)
An instance type from the general purpose family may not be the best choice when running a Redis server. Which family type of EC2 instances do you think has an advantage when running a Redis server? Why do you think the instance types from your chosen EC2 family are better suited for that task?

Some dynamic and optimized storage one would be better fittin.

Exercise 2 (Redis Benchmark)

a)
Which commands are executed by redis-benchmark ?

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-67-92 ~]$ redis-benchmark -q -n 500000 -a PeterPan
PING_INLINE: 88683.93 requests per second
PING_BULK: 93040.56 requests per second
SET: 92558.31 requests per second
GET: 91058.09 requests per second
INCR: 88074.69 requests per second
LPUSH: 97809.08 requests per second
RPUSH: 93040.56 requests per second
LPOP: 95147.48 requests per second
RPOP: 95347.06 requests per second
SADD: 91642.23 requests per second
HSET: 96116.88 requests per second
SPOP: 93668.05 requests per second
ZADD: 94286.25 requests per second
ZPOPMIN: 85631.10 requests per second
LPUSH (needed to benchmark LRANGE): 97012.03 requests per second
LRANGE_100 (first 100 elements): 54680.66 requests per second
LRANGE_300 (first 300 elements): 21265.74 requests per second
LRANGE_500 (first 450 elements): 16792.61 requests per second
LRANGE_600 (first 600 elements): 13569.63 requests per second
MSET (10 keys): 97904.84 requests per second
```

a)Briefly describe the commands used by the benchmark.

https://redis.io/commands/

PING_INLINE

Old pre-RESP-Format way of pinging

PING_BULK
Binary safe way of pinging

SET
how many elements can get inserted into the dbms per second
SET key value [EX seconds|PX milliseconds|KEEPTTL] [NX|XX] [GET] - Set key to hold the string value. If key already holds a value, it is overwritten, regardless of its type. Any previous time to live associated with the key is discarded on successful SET operation.

GET
Get the value of key. If the key does not exist the special value nil is returned. An error is returned if the value stored at key is not a string, because GET only handles string values.
how many elements can get retrieved by the dbms per second

INCR
how many keys can get incremented per sec.
NCR key - Increments the number stored at key by one. If the key does not exist, it is set to 0 before performing the operation. An error is returned if the key contains a value of the wrong type or contains a string that can not be represented as integer. This operation is limited to 64 bit signed integers.

LPUSH
LPUSH key element [element ...] - Insert all the specified values at the head of the list stored at key. If key does not exist, it is created as empty list before performing the push operations. When key holds a value that is not a list, an error is returned.

RPUSH
RPUSH key element [element ...] - Insert all the specified values at the tail of the list stored at key. If key does not exist, it is created as empty list before performing the push operation. When key holds a value that is not a list, an error is returned.

LPOP
LPOP key [count] - Removes and returns the first elements of the list stored at key. By default, the command pops a single element from the beginning of the list. When provided with the optional count argument, the reply will consist of up to count elements, depending on the list's length.

RPOP
RPOP key [count] - Removes and returns the last elements of the list stored at key. By default, the command pops a single element from the end of the list. When provided with the optional count argument, the reply will consist of up to count elements, depending on the list's length.

SADD
Add the specified members to the set stored at key. Specified members that are already a member of this set are ignored. If key does not exist, a new set is created before adding the specified members. An error is returned when the value stored at key is not a set.

HSET
Sets field in the hash stored at key to value. If key does not exist, a new key holding a hash is created. If field already exists in the hash, it is overwritten

SPOP
Removes and returns one or more random members from the set value store at key.
ZADD
Adds all the specified members with the specified scores to the sorted set stored at key. It is possible to specify multiple score / member pairs. If a specified member is already a member of the sorted set, the score is updated and the element reinserted at the right position to ensure the correct ordering.

ZPOPMIN
ZPOPMIN key [count] Removes and returns up to count members with the lowest scores in the sorted set stored at key.

LPUSH
LPUSH key element [element ...] - Insert all the specified values at the head of the list stored at key. If key does not exist, it is created as empty list before performing the push operations. When key holds a value that is not a list, an error is returned.

LRANGE_100
LRANGE key start stop - Returns the specified elements of the list stored at key. The offsets start and stop are zero-based indexes, with 0 being the first element of the list (the head of the list), 1 being the next element and so on.

LRANGE_300

LRANGE_500

LRANGE_600

MSET
MSET key value [key value ...] - Sets the given keys to their respective values. MSET replaces existing values with new values, just as regular SET.

• What can you infer from the values displayed?
How many requests per second can be executed

b) To test your answer from Exercise 1b, start up another instance with the code of Exercise 1 and use a different instance type this time. Choose an instance type which belongs to the EC2 family you gave as answer in the previous exercise. Arrange within your project team that each member uses a different instance type.

Also check the screenshot from above → C4.Xlarge

Benchmark Output (T2Medium):

[ec2-user@ip-172-31-43-108 ~]$ redis-benchmark -q -n 500000 -a PeterPan
PING_INLINE: 81859.86 requests per second
PING_BULK: 80360.01 requests per second
SET: 81819.67 requests per second
GET: 80853.81 requests per second
INCR: 81314.03 requests per second

LPUSH: 81512.88 requests per second
RPUSH: 81699.35 requests per second
LPOP: 81129.32 requests per second
RPOP: 81037.27 requests per second
SADD: 80231.07 requests per second
HSET: 81499.59 requests per second
SPOP: 81129.32 requests per second
ZADD: 81459.76 requests per second
ZPOPMIN: 80958.55 requests per second
LPUSH (needed to benchmark LRANGE): 81499.59 requests per second
LRANGE_100 (first 100 elements): 48206.71 requests per second
LRANGE_300 (first 300 elements): 19803.55 requests per second
LRANGE_500 (first 450 elements): 14780.66 requests per second
LRANGE_600 (first 600 elements): 11846.94 requests per second
MSET (10 keys): 82644.62 requests per second

Benchmark Output (T2Large):

[ec2-user@ip-172-31-64-46 ~]$ redis-benchmark -q -n 500000 -a "PeterPan"
PING_INLINE: 78517.59 requests per second
PING_BULK: 77627.70 requests per second
SET: 77639.75 requests per second
GET: 78149.42 requests per second
INCR: 77748.41 requests per second
LPUSH: 78566.94 requests per second
RPUSH: 78480.62 requests per second
LPOP: 78382.19 requests per second
RPOP: 78419.07 requests per second
SADD: 77978.79 requests per second
HSET: 78653.45 requests per second
SPOP: 77748.41 requests per second
ZADD: 78517.59 requests per second
ZPOPMIN: 77833.12 requests per second
LPUSH (needed to benchmark LRANGE): 78247.27 requests per second
LRANGE_100 (first 100 elements): 47691.72 requests per second
LRANGE_300 (first 300 elements): 18688.79 requests per second
LRANGE_500 (first 450 elements): 14244.61 requests per second
LRANGE_600 (first 600 elements): 11350.22 requests per second
MSET (10 keys): 79390.28 requests per second

C4.Xlarge is the fastest

Exercise 3:
a) dbsize = (integer) 100000
b) get "Michael Hauser" = (nil), so no
c) set "Michael Hauser" 48
get "Michael Hauser"
>"48"
d) KEYS *Michael*
1) "Markuss Modley"
1) "Mattheus Michaelsen"

2) "Michaelah Celli"
3) "Michaeljoseph Harer"
4) "Michaelee Blattner"
5) "Michaeleen Scouller"
6) "Michaelangelo Gertken"
7) "Michaeljames Holzwarth"
8) "Michaeljay Cosson"
9) "Michaelann Voth"
10) "Michaelah Linzie"
11) "Karlianne Michael"
12) "Michaelyn Taul"
13) "Michaelray Alvis"
14) "Michaelandrew Rippel"
15) "Michael-junior Stanbery"
16) "Michaelthomas Chindlund"
17) "Michaelallen Denio"
18) "Michael Hauser"
19) "Michael-paul Neusch"
20) "Michaelene Uran"
21) "Michaelyn Anchia"
22) "Juryj Michael"
23) "Michael-jude Fishburne"
24) "Michaell Rhule"
25) "Michaelina Gieber"
26) "Michaelin Tyberg"
27) "Michaelina Gailun"
28) "Michael-jay Czelusniak"
29) "Michaelia Baruffi"
30) "Michaeljoseph Klawinski"
31) "Jelessa Michaelsen"
32) "Michaelangelo Glen"
33) "Michael-tyler Lennard"
34) "Michael-jay Theimer"
35) "Michaeljames Uchida"
36) "Michaelena Rattay"
37) "Michaeldavid Gleich"
38) "Drumwright Michaelsen"
39) "Afat Michaelsen"
40) "Nahili Michaelson"
41) "Michael-paul Scalise"
42) "Michaella Stolfi"
43) "Michaelthomas Pechart"

e) No this is not easy, as the value is not the key, so you cant search for a specific key pattern. You cant directly search for a specific value. It is much easier to do it in sql