МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра инструментального и прикладного программного обеспечения

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1

по дисциплине

«Настройка и администрирование сервисного программного обеспечения»

Тема: «Установка приложения по работе с контейнерами Docker» по дисциплине «Настройка и администрирование сервисного программного обеспечения»

Выполнил студент группы ИКБО-32-21

Принял преподаватель

Практическая работа выполнена «__»_____202__ г. *(подпись студента)*

«Зачтено» «__»_____202__ г. *(подпись руководителя)*

Москва 2023

1. Цель работы

Получить навыки по развёртыванию приложения Docker.

2. Теоретическое введение

Технологии контейнеризации приложений нашли широкое применение в сферах разработки ПО и анализа данных. Эти технологии помогают сделать приложения более безопасными, облегчают их развёртывание и улучшают возможности по их масштабированию. Рост и развитие технологий контейнеризации можно считать одним из важнейших трендов современности.

Docker — это платформа, которая предназначена для разработки, развёртывания и запуска приложений в контейнерах.

Для тестирования возможностей Docker была выбрана программа Redis.

Redis — резидентная система управления базами данных класса NoSQL с открытым исходным кодом, работающая со структурами данных типа «ключ — значение».

Управление данными в Redis выполняется определенными командами, введенными в Redis CLI (Command-Line).

3. Выполнение работы

Задание:

В redis выполнить команды SET https://redis.io/commands/set и GET https://redis.io/commands/get Создать 5 ключей со значениями с помощью SET и прочитать ключи со значениями с помощью GET.

Решение:

Для создания пары ключ-значение воспользуемся командой SET.

Структура записи будет выглядеть так:

Листинг 1.

```
SET key “value”
```

Для получения значения по ключу воспользуемся командой GET.

Структура получения значения по ключу:

Листинг 2.

```
GET key
```

Результат:

Листинг 3.

```
# redis-cli
127.0.0.1:6379> set h1 Hi
OK
127.0.0.1:6379> set h2 Hii
OK
127.0.0.1:6379> set h3 Hiii
OK
127.0.0.1:6379> set h4 HA
OK
127.0.0.1:6379> set h5 HX
OK
127.0.0.1:6379> get h1
"Hi"
127.0.0.1:6379> get h2
"Hii"
127.0.0.1:6379> get h3
"Hiii"
127.0.0.1:6379> get h4
"HA"
127.0.0.1:6379> get h5
"HX"
```

Задание:

Получение значения по ключу и его замена на новое.

Решение:

Для замены значения по ключу используется команда GETSET:

Листинг 4.

```
GETSET key new_value
```

Результат:

Листинг 5.

```
# redis-cli
127.0.0.1:6379> set h "hi"
OK
127.0.0.1:6379> get h
"hi"
127.0.0.1:6379> getset h zxczxc
"hihihi"
127.0.0.1:6379> get h
"zxczxc"
```

Задание:

Добавление строки к существующему значению.

Решение:

Для добавления строки к существующему значению воспользуемся командой APPEND:

Листинг 6.

```
APPEND key value_to_add
```

Результат:

Листинг 7.

```
127.0.0.1:6379> set key Hello
OK
127.0.0.1:6379> append key World!
(integer) 11
127.0.0.1:6379> get key
"HelloWorld!"
```

Задание:

Добавление числа и изменение его значения.

Решение:

Для добавление числа воспользуемся стандартной командой SET, а для изменения числа INCRBY:

Листинг 8.

```
INCRBY key value_of_change
```

Результат:

Листинг 9.

```
127.0.0.1:6379> set zxc 123
OK
127.0.0.1:6379> INCRBY zxc 10
(integer) 133
127.0.0.1:6379> get zxc
"133"
```

Задание:

Создание ключа со значением типа хеш-таблица.

Решение:

Для создания и работы с хэш-таблицей используются команды: HSET, HGET, HGETALL.

Для создания хэш-таблицы используют команду HSET:

Листинг 10.

```
HSET key field value
```

Для получения значения из хэш-таблицы используют HGET:

Листинг 11.

```
HGET key field
```

Для получения всех элементов хэш-таблицы используют HGETALL:

Листинг 12.

```
HGETALL key
```

Результат:

Листинг 13.

```
# redis-cli
127.0.0.1:6379> HSET hash k1 H k2 HH k3 HHH
(integer) 3
127.0.0.1:6379> HGET hash k1
"H"
127.0.0.1:6379> hgetall hash
1) "k1"
2) "H"
3) "k2"
4) "HH"
```

```
5) "k3"
6) "HHH"
```

Задание:

Работа со множествами. Задействовать команды SADD, SDIFF, SMOVE, SPOP, SUNION, SREM.

Решение:

Для работы с множествами существует набор команд в redis:

SADD – добавляет новое значение в множество.

SDIFF – вычитает из одного множества другое.

SMOVE – перемещает значение из одного множества в другое.

SPOP – удаляет случайное значение из множества.

SUNION – объединяет два множества.

SREM – удаляет из множества определенное значение.

Результат:

Листинг 14.

```
# redis-cli
127.0.0.1:6379> sadd set 1 2 3 4
(integer) 4
127.0.0.1:6379> sadd set1 2 3
(integer) 2
127.0.0.1:6379> sdiff set set1
1) "1"
2) "4"
127.0.0.1:6379> smove set set1 4
(integer) 1
127.0.0.1:6379> smembers set1
```

```
1) "2"
2) "3"
3) "4"
127.0.0.1:6379> spop set
"2"
127.0.0.1:6379> smembers set
1) "1"
2) "3"
127.0.0.1:6379> sunion set set1
1) "1"
2) "2"
3) "3"
4) "4"
127.0.0.1:6379> smembers set1
1) "2"
2) "3"
3) "4"
127.0.0.1:6379> srem set1 2
(integer) 1
127.0.0.1:6379> smembers set1
1) "3"
2) "4"
```

Задание:

Работа с упорядоченными наборами. Задействовать команды ZADD, ZCOUNT, ZDIFF, ZPOPMAX, ZPOPMIN, ZUNION, ZMSCORE, ZLEXCOUNT

Решение:

Для работы с упорядоченными множествами существует набор команд в redis:

ZADD – добавляет элемент по ключу в упорядоченное множество.

ZDIFF – вычитает из одного упорядоченного множества другое.

ZCOUNT – считает количество элементов по условиям ключа

ZPOPMAX – случайно удаляет из упорядоченного множества элементы. Количество возможных удалений < [max].

ZPOPMIN – случайно удаляет элементы из упорядоченного множества. Количество возможных удалений > [min].

ZUNION – объединяет два упорядоченных множества.

ZMSCORE – вывод значения элементов в упорядоченном множестве.

ZLEXCOUNT – считает количество элементов, удовлетворяющих условию. Условие – имя элемента.

Результат:

Листинг 15.

```
127.0.0.1:6379> ZADD people 2015 sasha
(integer) 1
127.0.0.1:6379> ZADD people 2012 dima
(integer) 1
127.0.0.1:6379> ZADD people 1987 katya
(integer) 1
127.0.0.1:6379> ZADD people 1999 vanya
(integer) 1
127.0.0.1:6379> ZCOUNT people 2000 2014
```

```
(integer) 1
127.0.0.1:6379> ZCOUNT people 1900 2014
(integer) 3
127.0.0.1:6379> ZADD people2 1999 vanya 2012 dima
(integer) 2
127.0.0.1:6379> ZDIFF 2 people people2 WITHSCORES
1) "katya"
2) "1987"
3) "sasha"
4) "2015"
127.0.0.1:6379> ZDIFF 2 people people2
1) "katya"
2) "sasha"
127.0.0.1:6379> ZPOPMAX people 2
1) "sasha"
2) "2015"
3) "dima"
4) "2012"
127.0.0.1:6379> ZPOPMIN people 1
1) "katya"
2) "1987"
127.0.0.1:6379> ZUNION 2 people people2
1) "dima"
2) "vanya"
127.0.0.1:6379> ZMSCORE people vanya dima
1) "1999"
2) (nil)
127.0.0.1:6379> ZADD people 2001 dima 1893 opas 1233 oleg
```

```
(integer) 3
127.0.0.1:6379> ZRANGE people 0 3000 WITHSCORES
1) "oleg"
2) "1233"
3) "opas"
4) "1893"
5) "vanya"
6) "1999"
7) "dima"
8) "2001"
127.0.0.1:6379> ZLEXCOUNT people - +
(integer) 4
127.0.0.1:6379> ZLEXCOUNT people [a [b
(integer) 0
127.0.0.1:6379> ZLEXCOUNT people [a [s
(integer) 2
```

Задание:

Из документации выбрать любые не использовавшиеся ранее 5 команд и задействовать их в работе.

Решение:

Были выбраны команды: LPUSH, RPUSH, LLEN, LRANGE, LINDEX.

Данные команды позволяют работать со списком.

LPUSH – добавляет элемент в начало списка.

RPUSH – добавляет элемент в конец списка.

LLEN – выводит длину списка.

LRANGE – выводит элементы списка согласно условию.

LINDEX – выводит элемент списка по его порядковому номеру.

Результат:

```
# redis-cli
127.0.0.1:6379> lpush list a b c d e f
(integer) 6
127.0.0.1:6379> rpush list aa
(integer) 7
127.0.0.1:6379> lpush list bb
(integer) 8
127.0.0.1:6379> llen list
(integer) 8
127.0.0.1:6379> lindex list 3
"d"
127.0.0.1:6379> lrange list 0 2
1) "bb"
2) "f"
3) "e"
```

4. Выводы

В результате выполненной работы были получены знания и навыки о развёртывании приложения Docker и работе с СУБД Redis.