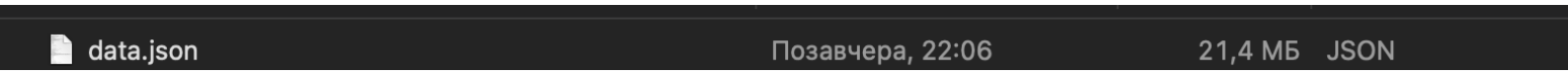
1) Скачаем сгенерированный data.json размером около 20 МБ.



2) Скачем образ redis и запустим контейнер, прокинув порт 6379.

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker images | grep redis
redis        latest    bf3306b23f41   2 months ago   158MB
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker run -d -p 6379:6379 --name redis_hw2 --rm redis:latest
28da5a8554cc3ecd2f9d461e5e088254678379b811bc1820280f72319e01e458
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
28da5a8554cc   redis:latest   "docker-entrypoint.s…"   38 seconds ago   Up 37 seconds   0.0.0.0:6379->6379/tcp   redis_hw2
(base) moisha@MacBook-Pro-Mihail-2 hw_2 %
```

3) Зайдем в контейнер и проверим, работает ли он.

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker exec -it redis_hw2 redis-cli
127.0.0.1:6379> PING
PONG
127.0.0.1:6379>
```

4) Используя библиотеку redis и питон, будем переводить наш JSON в различные форматы и закидывать их в контейнер. (main.py)

5) Проверим, что у нас в контейнере.

```
127.0.0.1:6379> KEYS *
1) "data_zset"
2) "data_list"
3) "data_hset"
4) "data_string"
127.0.0.1:6379> TYPE data_zset
zset
127.0.0.1:6379> TYPE data_list
list
127.0.0.1:6379> TYPE data_hset
hash
127.0.0.1:6379> TYPE data_string
string
127.0.0.1:6379>
```

СОЗДАНИЕ КЛАССТЕРА

7) Для мастер создадим redis_master.config и Dockerfile.master

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % cat redis_master.conf
# Порт для соединений Redis
port 6379

# Включить поддержку кластеризации
cluster-enabled yes

# Файл конфигурации кластера
cluster-config-file nodes.conf

# Время ожидания между узлами кластера
cluster-node-timeout 3000

# Включить журналирование операций
appendonly yes
```

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % cat Dockerfile.master
FROM redis:latest

RUN mkdir /usr/local/etc/redis/
COPY redis_master.conf /usr/local/etc/redis/redis.conf

CMD [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
(base) moisha@MacBook-Pro-Mihail-2 hw_2 %
```

8) Для реплик аналогично

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % cat redis_replica.conf
port 6379
replicaof 127.0.0.1 6379
```

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % cat Dockerfile.replica
FROM redis:latest

RUN mkdir /usr/local/etc/redis/

COPY redis_replica.conf /usr/local/etc/redis/redis.conf

CMD [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
```

9) Соберем эти образы.

```
REPOSITORY   TAG       IMAGE ID       CREATED                  SIZE
redis        replica   aeca7b8ed85a   59 minutes ago           158MB
redis        master    65ce9c19baa5   About an hour ago        158MB
```

10) Сделаем сеть, чтобы наши контейнеры могли общаться между собой

```
4bb39441a52c   redis-network   bridge    local
```

11) Запустим контейнеры

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker run -d --name master-redis --net redis-network redis:master
```

```
% docker run -d --name replica1-redis --net redis-network redis:replica --slaveof master-redis 6379
```

12) В итоге:

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED             STATUS             PORTS      NAMES
ef6473a72e7a   redis:replica   "docker-entrypoint.s…"   About an hour ago   Up About an hour   6379/tcp   replica2-redis
7c810b8a66b0   redis:replica   "docker-entrypoint.s…"   About an hour ago   Up About an hour   6379/tcp   replica1-redis
96a5f622d860   redis:master    "docker-entrypoint.s…"   About an hour ago   Up About an hour   6379/tcp   master-redis
```

13) Запустим класстер

```
(base) moisha@MacBook-Pro-Mihail-2 hw_2 % docker exec -it 96 redis-cli --cluster create 127.0.0.1:6379 127.0.0.1:6379 127.0.0.1:6379
>>> Performing hash slots allocation on 3 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
>>> Trying to optimize slaves allocation for anti-affinity
[WARNING] Some slaves are in the same host as their master
M: 01f731ccf8a0dbd57de793539db9457faa594f44 127.0.0.1:6379
   slots:[0-5460] (5461 slots) master
M: 01f731ccf8a0dbd57de793539db9457faa594f44 127.0.0.1:6379
   slots:[5461-10922] (5462 slots) master
M: 01f731ccf8a0dbd57de793539db9457faa594f44 127.0.0.1:6379
   slots:[10923-16383] (5461 slots) master
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
>>> Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join

>>> Performing Cluster Check (using node 127.0.0.1:6379)
M: 01f731ccf8a0dbd57de793539db9457faa594f44 127.0.0.1:6379
   slots:[0-16383] (16384 slots) master
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
```

14) Зайдем в контейнер с мастер нодой и попросим redis вывести роль

```
root@96a5f622d860:/data# redis-cli
127.0.0.1:6379> role
1) "master"
2) (integer) 5533
3) 1) 1) "172.22.0.4"
      2) "6379"
      3) "5533"
   2) 1) "172.22.0.3"
      2) "6379"
      3) "5533"
```

15) Аналогично для реплики

```
root@ef6473a72e7a:/data# redis-cli
127.0.0.1:6379> role
1) "slave"
2) "master-redis"
3) (integer) 6379
4) "connected"
5) (integer) 5463
```

16) Добавим пару ключей в мастер ноу и посмотрим, появится ли они в реплике

```
(integer) 1
127.0.0.1:6379> SET test HI_FROM_MASTER
OK
127.0.0.1:6379> keys *
1) "test"
127.0.0.1:6379>
```

-master_node

```
127.0.0.1:6379> keys *
1) "test"
127.0.0.1:6379> get test
"HI_FROM_MASTER"
127.0.0.1:6379>
```

-Реплика

17) Заметим, что в реплику добавить ничего нельзя

```
127.0.0.1:6379> set test_1 МЫШКА
(error) READONLY You can't write against a read only replica.
127.0.0.1:6379>
```