




Lab 2 – Tópicos especiales en telemática

por Santiago Vanegas – svaneg11@eafit.edu.co

1. Instalar y configurar redis en un solo nodo.

Se crea el primer nodo, y se le asocia una IP elastica.

Instances (1) Info					
<input type="text" value="Search"/>					
<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	
<input type="checkbox"/>	Node-1	i-04dd2e9c76948ca36	Running  	t2.micro	

Se instala Redis y se procede a probar usando ping.

```
ec2-user@ip-172-31-86-16:~  
[ec2-user@ip-172-31-86-16 ~]$ redis-cli ping  
PONG  
[ec2-user@ip-172-31-86-16 ~]$ redis-cli -v  
redis-cli 6.2.6  
[ec2-user@ip-172-31-86-16 ~]$
```

1.1. Se configura la contraseña para redis.

```
[ec2-user@ip-172-31-86-16 etc]$ sudo vim redis/redis.conf
```

```
# IMPORTANT NOTE: starting with Redis 6 "requirepass" is just a compatibility  
# layer on top of the new ACL system. The option effect will be just setting  
# the password for the default user. Clients will still authenticate using  
# AUTH <password> as usually, or more explicitly with AUTH default <password>  
# if they follow the new protocol: both will work.  
#  
# The requirepass is not compatable with aclfile option and the ACL LOAD  
# command, these will cause requirepass to be ignored.  
#  
requirepass telematica
```

1.2. Configurando la persistencia en redis usando un archivo de log, para que guarde cada segundo los cambios en la base de datos.

```
appendonly yes

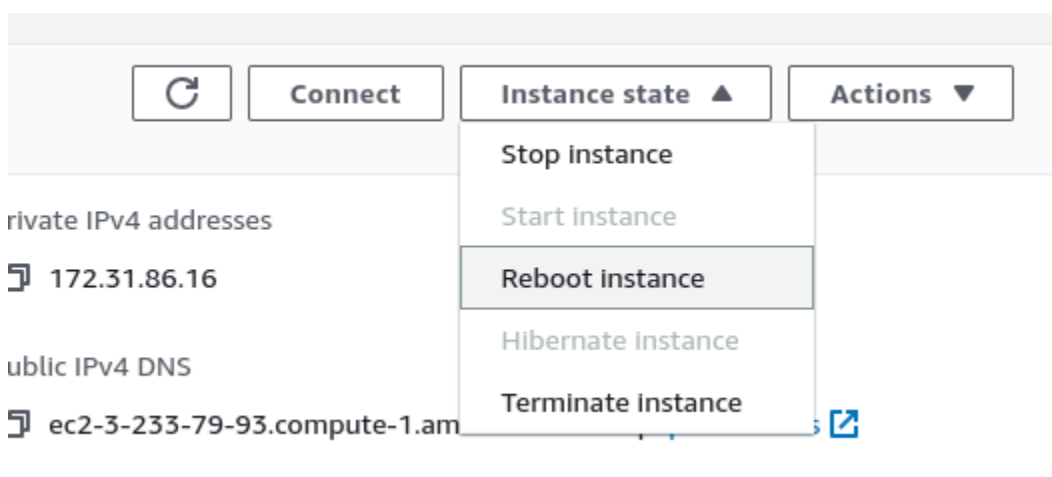
# The name of the append only file (default: "appendonly.aof")

appendfilename "appendonly.aof"

# The fsync() call tells the Operating System to actually write data on disk
# instead of waiting for more data in the output buffer. Some OS will really flush
# data on disk, some other OS will just try to do it ASAP.
#
# Redis supports three different modes:
#
# no: don't fsync, just let the OS flush the data when it wants. Faster.
# always: fsync after every write to the append only log. Slow, Safest.
# everysec: fsync only one time every second. Compromise.
#
# The default is "everysec", as that's usually the right compromise between
# speed and data safety. It's up to you to understand if you can relax this to
# "no" that will let the operating system flush the output buffer when
# it wants, for better performances (but if you can live with the idea of
# some data loss consider the default persistence mode that's snapshotting),
# or on the contrary, use "always" that's very slow but a bit safer than
# everysec.
#
# More details please check the following article:
# http://antirez.com/post/redis-persistence-demystified.html
#
# If unsure, use "everysec".

# appendfsync always
appendfsync everysec
# appendfsync no
```

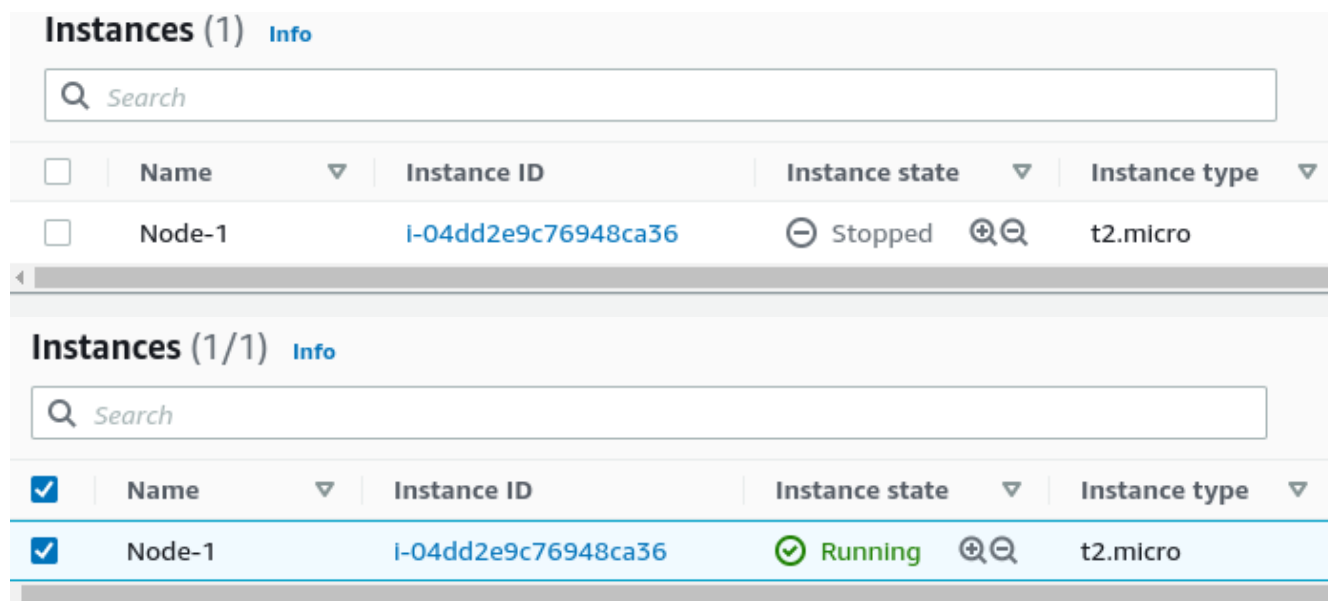
> Finalmente se hace reboot de la instancia para reiniciar redis-server y que tome los cambios del conf.



> Ahora el password es requerido.

```
[ec2-user@ip-172-31-86-16 ~]$ redis-cli
127.0.0.1:6379> set x z
(error) NOAUTH Authentication required.
127.0.0.1:6379> auth telematica
OK
127.0.0.1:6379> set x z
OK
127.0.0.1:6379> 
```

> Se detiene y corre de nuevo la instancia para probar la persistencia de redis.



> La persistencia funciona!

```
[santi@localhost Escritorio]$ ssh -i "redis-nodes.pem" ec2-user@ec2-3-233-79-93.compute-1.amazonaws.com
Last login: Sun Mar  6 19:04:39 2022 from 201.221.176.3

 _ _ | _ _ | _ )
 _ | ( _ _ /   Amazon Linux 2 AMI
 _ _ | \ _ _ | _ _ |

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-86-16 ~]$ redis-cli
127.0.0.1:6379> auth telematica
OK
127.0.0.1:6379> get x
"z"
127.0.0.1:6379> █
```

2. Operaciones CRUD en Redis (*Seven Databases in Seven Weeks, Chap 8*).

> SET y GET

```
OK
127.0.0.1:6379> SET 7wks http://www.sevenweeks.org/
OK
127.0.0.1:6379> GET 7wks
"http://www.sevenweeks.org/"
127.0.0.1:6379> █
```

> Eliminar con DEL

```
127.0.0.1:6379> SET user svaneg11
OK
127.0.0.1:6379> GET user
"svaneg11"
127.0.0.1:6379> DEL user
(integer) 1
127.0.0.1:6379> GET user
(nil)
```

> MSET y MGET (Multiple-SET y Multiple-GET)

```
127.0.0.1:6379> MSET gog http://www.google.com.ezproxy.eafit.edu.co yah http://www.yahoo.com
OK
127.0.0.1:6379> MGET gog yah
1) "http://www.google.com.ezproxy.eafit.edu.co"
2) "http://www.yahoo.com"
127.0.0.1:6379> █
```

> Incrementar y disminuir enteros.

```
127.0.0.1:6379> SET count 2
OK
127.0.0.1:6379> INCR count
(integer) 3
127.0.0.1:6379> DECR count
(integer) 2
```

3. Haciendo operaciones CRUD desde python con Redis

> Se habilita el puerto 6379 en el security group del Nodo-1 para que pueda recibir peticiones desde fuera, y no rechaza la conexión de python.

> Se elimina la linea bind 127.0.0.1 del archivo */etc/redis/redis.conf* para que redis escuche todas las interfaces de red (si no lanza un connection refused desde el cliente python)

Luego se procede a correr los programas de Python.

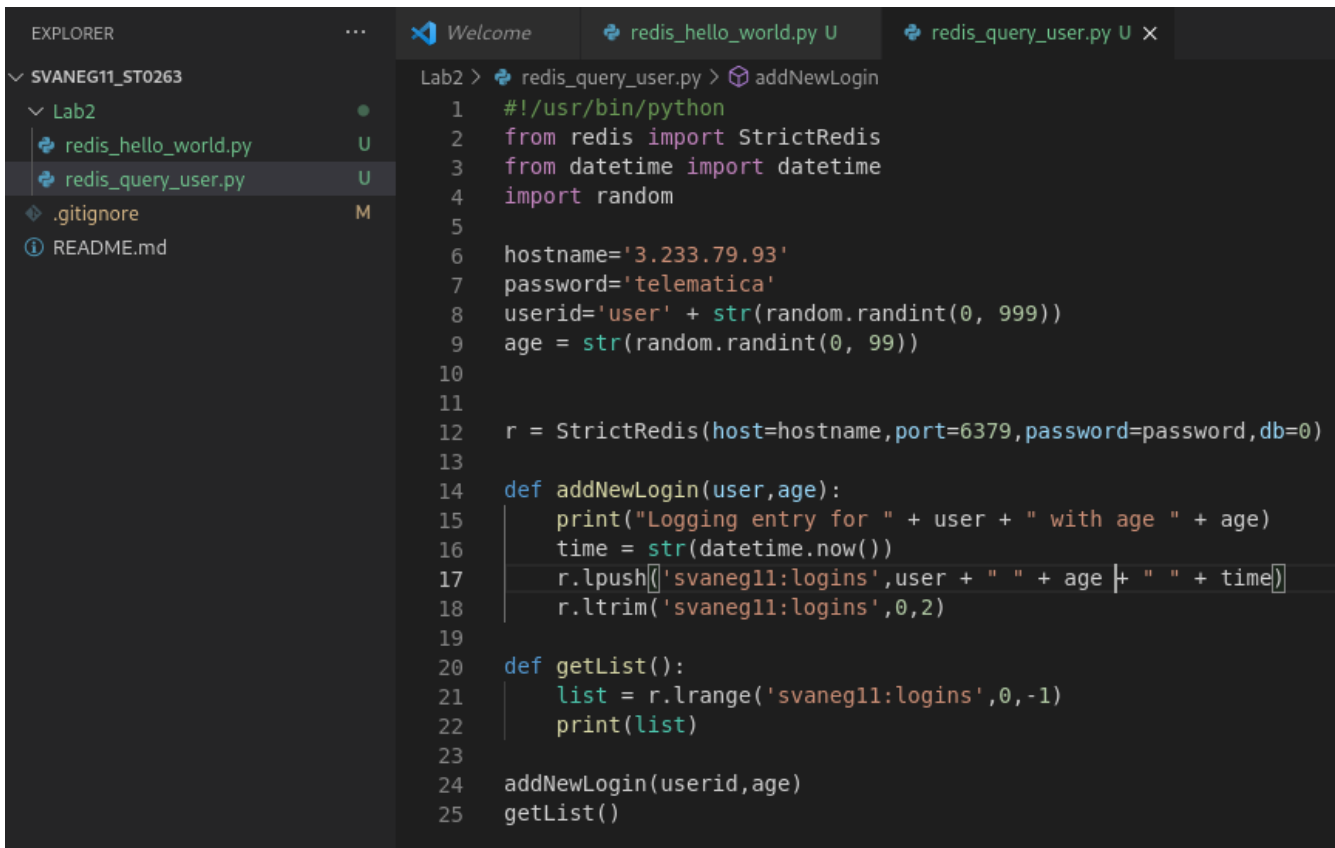
> **redis_hello_world.py**

```
SVANEG11_ST0263
└─ Lab2
   └─ redis_hello_world.py
   └─ .gitignore
   └─ README.md

Lab2 > redis_hello_world.py > ...
1  #!/usr/bin/python
2  from redis import StrictRedis
3  import sys
4
5  hostname='3.233.79.93'
6  password='telematica'
7
8  r = StrictRedis(host=hostname,port=6379,password=password,db=0)
9
10 def getPacktWelcome():
11     #GET value stored in packt:welcome
12     print("Displaying current welcome message...")
13     value = r.get('svaneg11:welcome')
14     print("message = " + str(value))
15
16 def setPacktWelcome():
17     #SET new value packt:welcome
18     print("Writing \"Hello world from Python!\" to Redis...")
19     r.set('svaneg11:welcome','Hello world from Python!')
20
21 getPacktWelcome()
22 setPacktWelcome()
23 getPacktWelcome()
```

```
[santi@fedora svaneg11_ST0263]$ /usr/bin/python /home/santi/Escritorio/svaneg11_ST0263/Lab2/redis_hello_world.py
Displaying current welcome message...
message = None
Writing "Hello world from Python!" to Redis...
Displaying current welcome message...
message = b'Hello world from Python!'
```

> redis_query_user.py



```
Lab2 > redis_query_user.py > addNewLogin
1  #!/usr/bin/python
2  from redis import StrictRedis
3  from datetime import datetime
4  import random
5
6  hostname='3.233.79.93'
7  password='telematica'
8  userid='user' + str(random.randint(0, 999))
9  age = str(random.randint(0, 99))
10
11
12  r = StrictRedis(host=hostname,port=6379,password=password,db=0)
13
14  def addNewLogin(user,age):
15      print("Logging entry for " + user + " with age " + age)
16      time = str(datetime.now())
17      r.lpush('svaneg11:logins',user + " " + age + " " + time)
18      r.ltrim('svaneg11:logins',0,2)
19
20  def getList():
21      list = r.lrange('svaneg11:logins',0,-1)
22      print(list)
23
24  addNewLogin(userid,age)
25  getList()
```

```
[santi@fedora svaneg11_ST0263]$ /usr/bin/python /home/santi/Escritorio/svaneg11_ST0263/Lab2/redis_query_user.py
Logging entry for user249 with age 32
[b'user249 32 2022-03-08 00:58:48.405039']
[santi@fedora svaneg11_ST0263]$ /usr/bin/python /home/santi/Escritorio/svaneg11_ST0263/Lab2/redis_query_user.py
Logging entry for user747 with age 83
[b'user747 83 2022-03-08 00:59:00.178936', b'user249 32 2022-03-08 00:58:48.405039']
[santi@fedora svaneg11_ST0263]$ /usr/bin/python /home/santi/Escritorio/svaneg11_ST0263/Lab2/redis_query_user.py
Logging entry for user164 with age 48
[b'user164 48 2022-03-08 00:59:02.488266', b'user747 83 2022-03-08 00:59:00.178936', b'user249 32 2022-03-08 00:58:48.405039']
[santi@fedora svaneg11_ST0263]$
```

> redis pub y sub

redis_pub.py

```
redis_pub.py U x
Lab2 > redis_pub.py > ...
1  #!/usr/bin/python
2  from redis import StrictRedis
3
4  hostname='3.233.79.93'
5  password='telematica'
6  channel='1'
7
8  r = StrictRedis(host=hostname,port=6379,password=password,db=0)
9  publisher = r.pubsub()
10
11 while True:
12     message=input("Describe play, or press [Enter] to quit: ")
13
14     if not message:
15         break
16     else:
17         r.publish(channel,message)
18         #print message
19
20 print ("Publish program ended.")
```

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE

```
[santi@fedora Lab2]$ python redis_pub.py
Describe play, or press [Enter] to quit: Hello World!!!
Describe play, or press [Enter] to quit: Topicos
Describe play, or press [Enter] to quit: especiales
Describe play, or press [Enter] to quit: en
Describe play, or press [Enter] to quit: telematica
Describe play, or press [Enter] to quit: 1234
Describe play, or press [Enter] to quit:
Publish program ended.
[santi@fedora Lab2]$
```

redis_sub.py

```
redis_sub.py U x
Lab2 > redis_sub.py > ...
1  #!/usr/bin/python
2  from redis import StrictRedis
3
4  hostname='3.233.79.93'
5  password='telematica'
6  channel='1'
7
8  r = StrictRedis(host=hostname,port=6379,password=password,db=0)
9  channels = r.pubsub()
10 channels.subscribe(channel)
11
12 for message in channels.listen():
13     if message['data']=='END':
14         break
15
16     if message['data']!=1:
17         print(message['data'])
18
19 channels.unsubscribe(channel)
20 print("Unsubscribed")
```

```
[santi@fedora Lab2]$ python redis_sub.py
b'Hello World!!!'
b'Topicos'
b'especiales'
b'en'
b'telematica'
b'1234'
□
```


4. Crear un clúster de redis.

> Creando cluster en local

Dentro de la carpeta cluster-test, se crea una carpeta en local por cada instancia de redis. El numero de la carpeta indica el puerto que usa cada server de redis. También se ubica el ejecutable de redis-server en la carpeta

```
[santi@localhost cluster-test]$ ls
7000 7001 7002 7003 7004 7005 redis-server
[santi@localhost cluster-test]$
```

Dentro de cada carpeta se ubica un archivo redis.conf que indica su respectivo puerto.

```
redis.conf 7000 x  redis.conf 7001  redis.conf 7002  redis.conf 7003  redis.conf 7004  redis.conf 7005
7000 > redis.conf
1  port 7000
2  cluster-enabled yes
3  cluster-config-file nodes.conf
4  cluster-node-timeout 5000
5  appendonly yes
6  daemonize yes
```

Luego se ubica dentro de cada carpeta y se corre cada instancia de redis-server usando su respectivo archivo de configuracion

```
[santi@localhost cluster-test]$ cd 7000
[santi@localhost 7000]$ ../redis-server ../redis.conf
```

```
[santi@localhost 7000]$ cd ../7001
[santi@localhost 7001]$ ../redis-server ../redis.conf
[santi@localhost 7001]$ cd ../7002
[santi@localhost 7002]$ ../redis-server ../redis.conf
[santi@localhost 7002]$ cd ../7003
[santi@localhost 7003]$ ../redis-server ../redis.conf
[santi@localhost 7003]$ ls
appendonly.aof nodes.conf redis.conf
[santi@localhost 7003]$ cd ../7004
[santi@localhost 7004]$ ../redis-server ../redis.conf
[santi@localhost 7004]$ cd ../7005
```

Finalmente se corre redis-cli con la opcion de `--cluster` y se indica la direccion de cada servidor, y se indica cuantas replicas por nodo maestro habrá con la opcion `--cluster replicas 1` (3 replicas por nodo maestro, para un total de 6)

```
[santi@localhost cluster-test]$ redis-cli --cluster create 127.0.0.1:7000 127.0.0.1:7001 \
> 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005 \
> --cluster-replicas 1
```

```
>>> Performing hash slots allocation on 6 nodes...
```

```
Master[0] -> Slots 0 - 5460
```

```
Master[1] -> Slots 5461 - 10922
```

```
Master[2] -> Slots 10923 - 16383
```

```
Adding replica 127.0.0.1:7004 to 127.0.0.1:7000
```

```
Adding replica 127.0.0.1:7005 to 127.0.0.1:7001
```

```
Adding replica 127.0.0.1:7003 to 127.0.0.1:7002
```

```
>>> Trying to optimize slaves allocation for anti-affinity
```

```
[WARNING] Some slaves are in the same host as their master
```

```
M: 9189d8a16853f49a5a27067428a5c1b29d7051c0 127.0.0.1:7000  
slots:[0-5460] (5461 slots) master
```

```
M: a72cb42640c5d49fef4486410933d78542f0b70a 127.0.0.1:7001  
slots:[5461-10922] (5462 slots) master
```

```
M: 2e6cb8ad92154c72cb2c27f0e31ea8eb7a9a63aa 127.0.0.1:7002  
slots:[10923-16383] (5461 slots) master
```

```
S: ca4752a967bc8c948a3138961d30be2e4c8d639b 127.0.0.1:7003  
replicates 2e6cb8ad92154c72cb2c27f0e31ea8eb7a9a63aa
```

```
S: 17e411302b32adb59ca942a15caefa46cf7c7492 127.0.0.1:7004  
replicates 9189d8a16853f49a5a27067428a5c1b29d7051c0
```

```
S: 7651c7e7987bd07d1c8c8a7a463173efc958ed01 127.0.0.1:7005  
replicates a72cb42640c5d49fef4486410933d78542f0b70a
```

```
Can I set the above configuration? (type 'yes' to accept): yes
```

Redis configura 3 nodos maestros y un nodo esclavo para cada maestro y asigna a cada maestro un rango de las particiones hash (slots) para cubrir todo el total de 16384 particiones que usa Redis.

```
>>> 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 127.0.0.1:7000)
```

```
M: 9189d8a16853f49a5a27067428a5c1b29d7051c0 127.0.0.1:7000  
slots:[0-5460] (5461 slots) master  
1 additional replica(s)
```

```
S: 7651c7e7987bd07d1c8c8a7a463173efc958ed01 127.0.0.1:7005  
slots: (0 slots) slave  
replicates a72cb42640c5d49fef4486410933d78542f0b70a
```

```
S: 17e411302b32adb59ca942a15caefa46cf7c7492 127.0.0.1:7004  
slots: (0 slots) slave  
replicates 9189d8a16853f49a5a27067428a5c1b29d7051c0
```

```
M: a72cb42640c5d49fef4486410933d78542f0b70a 127.0.0.1:7001  
slots:[5461-10922] (5462 slots) master  
1 additional replica(s)
```

```
S: ca4752a967bc8c948a3138961d30be2e4c8d639b 127.0.0.1:7003  
slots: (0 slots) slave  
replicates 2e6cb8ad92154c72cb2c27f0e31ea8eb7a9a63aa
```

```
M: 2e6cb8ad92154c72cb2c27f0e31ea8eb7a9a63aa 127.0.0.1:7002  
slots:[10923-16383] (5461 slots) master  
1 additional replica(s)
```

```
[OK] All nodes agree about slots configuration.
```

```
>>> Check for open slots...
```

```
>>> Check slots coverage...
```

```
[OK] All 16384 slots covered.
```

Ahora se puede escribir en un nodo de redis y los otros nodos reconoceran a cual nodo pertenece esa llave, indicando al cliente que nodo consultar.

```
[santi@localhost cluster-test]$ redis-cli -h 127.0.0.1 -p 7000
127.0.0.1:7000> SET materia "Topicos especiales en telematica"
OK
127.0.0.1:7000>
[santi@localhost cluster-test]$ redis-cli -h 127.0.0.1 -p 7005
127.0.0.1:7005> GET materia
(error) MOVED 2530 127.0.0.1:7000
```

> Creando el cluster en EC2

Se crea una instancia de EC2 y se aplican los mismos pasos que al crear el cluster local. Ademas se agrega la linea "requirepass telematica" en el redis.conf para que la autenticacion remota con contraseña funcione.

```
[ec2-user@ip-172-31-25-168 cluster-test]$ cd 7000 && ../redis-server ./redis.conf && cd ..
[ec2-user@ip-172-31-25-168 cluster-test]$ cd 7001 && ../redis-server ./redis.conf && cd ..
[ec2-user@ip-172-31-25-168 cluster-test]$ cd 7002 && ../redis-server ./redis.conf && cd ..
[ec2-user@ip-172-31-25-168 cluster-test]$ cd 7003 && ../redis-server ./redis.conf && cd ..
[ec2-user@ip-172-31-25-168 cluster-test]$ cd 7004 && ../redis-server ./redis.conf && cd ..
[ec2-user@ip-172-31-25-168 cluster-test]$ cd 7005 && ../redis-server ./redis.conf && cd ..
[ec2-user@ip-172-31-25-168 cluster-test]$ ls 7005
appendonly.aof nodes.conf redis.conf
[ec2-user@ip-172-31-25-168 cluster-test]$ redis-cli --cluster create 127.0.0.1:7000 127.0.0.1:7001 \
> 127.0.0.1:7002 127.0.0.1:7003 127.0.0.1:7004 127.0.0.1:7005 \
> --cluster-replicas 1
>>> Performing hash slots allocation on 6 nodes...
Master[0] -> Slots 0 - 5460
Master[1] -> Slots 5461 - 10922
Master[2] -> Slots 10923 - 16383
Adding replica 127.0.0.1:7004 to 127.0.0.1:7000
Adding replica 127.0.0.1:7005 to 127.0.0.1:7001
Adding replica 127.0.0.1:7003 to 127.0.0.1:7002
```

Nos conectamos a un nodo de redis por el puerto 7000, especificamos al cliente de redis que redirija las peticiones al nodo correcto usando el argumento **-c**

Hacemos SET mylab lab2 en el nodo 7000
Luego hacemos GET mylab en el nodo 7005.

El cluster funciona.

```
[ec2-user@ip-172-31-25-168 cluster-test]$ redis-cli -c -h 127.0.0.1 -p 7000 -a telematica
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
127.0.0.1:7000> SET mylab lab2
OK
127.0.0.1:7000>
[ec2-user@ip-172-31-25-168 cluster-test]$ redis-cli -c -h 127.0.0.1 -p 7005 -a telematica
Warning: Using a password with '-a' or '-u' option on the command line interface may not be safe.
127.0.0.1:7005> GET mylab
-> Redirected to slot [4945] located at 127.0.0.1:7000
"lab2"
```

5. Conclusiones

> **Especificaciones funcionales**

Para nuestro proyecto 1 implementaremos, los métodos SET, GET y DEL.

> **Especificaciones no funcionales**

Pariticionamiento:

La base de datos minimalista contará con particionamiento de tipo Key by Hash Range,

Replicación y Tolerancia a fallos:

Aunque la idea es no centrarnos tanto en la parte de replicación, si nos alcanza el tiempo pensariamos en aplicar estrategias, como las de reemplazar el nodo replica por el nodo maestro en caso de que el maestro pase un determinado tiempo sin responder a los otros nodos.

Disponibilidad:

Para el diseño de la base de datos preferimos la disponibilidad por encima de la consistencia por lo que la base de datos usará una estrategia de replicación de datos asincrona.

Transacciones:

No se implementará

6. Ejercicios adicionales con Redis.

> Transacciones con MULTI y EXEC en Redis.

```
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> SET prag http://pragprog.com
QUEUED
127.0.0.1:6379(TX)> INCR count
QUEUED
127.0.0.1:6379(TX)> EXEC
1) OK
2) (integer) 3
```

> Descartar transacciones con DISCARD

```
127.0.0.1:6379> multi
OK
127.0.0.1:6379(TX)> incr count
QUEUED
127.0.0.1:6379(TX)> incr count
QUEUED
127.0.0.1:6379(TX)> incr count
QUEUED
127.0.0.1:6379(TX)> discard
OK
```

> Objetos anidados con dos puntos

```
127.0.0.1:6379> MSET user:svaneg11:name "Santiago Vanegas" user:svaneg11:password pass1234
OK
127.0.0.1:6379> MGET user:svaneg11:name user:svaneg11:password
1) "Santiago Vanegas"
2) "pass1234"
127.0.0.1:6379> □
```

> Objetos anidados con Hashes

```
127.0.0.1:6379> HMSET user:svaneg11 name "Santiago Vanegas" password pass1234
OK
127.0.0.1:6379> HVALS user:svaneg11
1) "Santiago Vanegas"
2) "pass1234"
□
```

> Recuperar todas las llaves (Keys) de un Hash

```
127.0.0.1:6379> HKEYS user:svaneg11
1) "name"
2) "password"
```

> Hash Multiple GET (HMGET)

```
127.0.0.1:6379> HMGET user:svaneg11 name password
1) "Santiago Vanegas"
2) "pass1234"
```

> Listas

> Insertar al final de la lista con RPush y sacar el primer elemento con LPOP

```
127.0.0.1:6379> RPush svaneg11:wishlist 7wks gog prag
(integer) 3
127.0.0.1:6379> LPOP svaneg11:wishlist
"7wks"
```

> Listar los objetos restantes con LRange

```
(empty array)
127.0.0.1:6379> LRange svaneg11:wishlist 0 -1
1) "gog"
2) "prag"
```

> Remover todos los objetos de una lista que coincidan con un valor con LREM.

```
127.0.0.1:6379> LREM svaneg11:wishlist 0 gog
(integer) 1
127.0.0.1:6379> LRange svaneg11:wishlist 0 -1
1) "prag"
```

> Sets

- > SADD: Crear set
- > SMEMBERS: Mostrar los elementos de un set
- > SINTER: Interseccion de dos conjuntos
- > SDIFF: Elementos del set 1 que no estan en el set 2
- > SUNION: Union de dos conjuntos

```
127.0.0.1:6379> SADD news nytimes.com pragprog.com
(integer) 2
127.0.0.1:6379> SMEMBERS news
1) "pragprog.com"
2) "nytimes.com"
127.0.0.1:6379> SADD tech pragprog.com apple.com
(integer) 2
127.0.0.1:6379> SINTER news tech
1) "pragprog.com"
127.0.0.1:6379> SDIFF news tech
1) "nytimes.com"
127.0.0.1:6379> SUNION news tech
1) "apple.com"
2) "pragprog.com"
3) "nytimes.com"
```