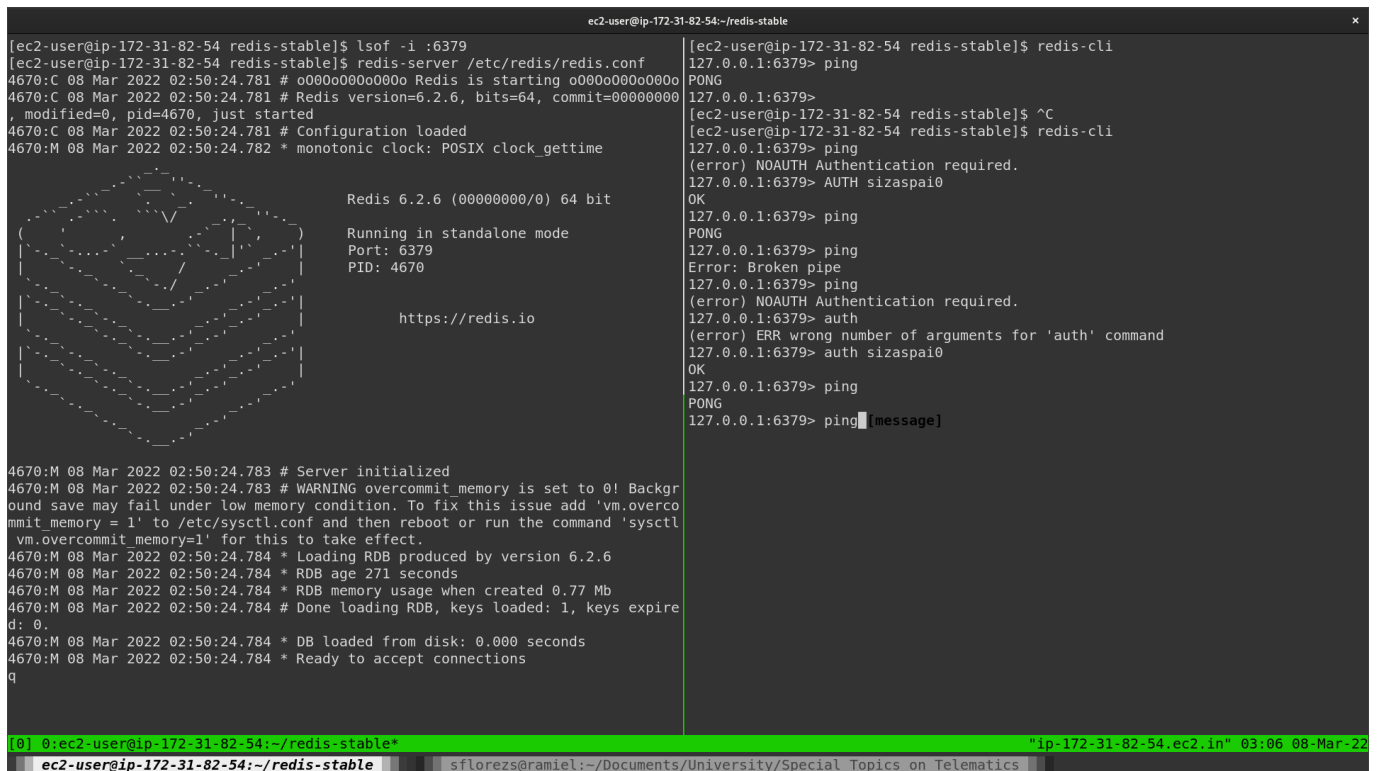


## Open a terminal

```
# install a c compiler
sudo yum install -y gcc
# Download and install redis
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make distclean
make
# install globally
sudo make install
# create a backup for the config file
cp redis.conf{,.bk}
# setup a password for auth
sed 's/# requirepass foobard/requirepass M1V3r1S7R0NGP455/g' redis.conf
# run the redis server with the config, make sure to open another terminal
since this command will lock the terminal until the server is terminated
redis-server redis.conf
```

## On other terminal

```
# test the connection
redis-cli
127.0.0.1:6379> ping
# (error) NOAUTH Authentication required.
127.0.0.1:6379> auth M1V3r1S7R0NGP455
# OK
127.0.0.1:6379> ping
# PONG
```



```

[ec2-user@ip-172-31-82-54 ~redis-stable]$ lsof -i :6379
[ec2-user@ip-172-31-82-54 redis-stable]$ redis-server /etc/redis/redis.conf
4670:C 08 Mar 2022 02:50:24.781 # oOoOoOoOoOoOo Redis is starting oOoOoOoOoOoOo
4670:C 08 Mar 2022 02:50:24.781 # Redis version=6.2.6, bits=64, commit=00000000
, modified=0, pid=4670, just started
4670:C 08 Mar 2022 02:50:24.781 # Configuration loaded
4670:M 08 Mar 2022 02:50:24.782 * monotonic clock: POSIX clock_gettime

Redis 6.2.6 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 4670

https://redis.io

4670:M 08 Mar 2022 02:50:24.783 # Server initialized
4670:M 08 Mar 2022 02:50:24.783 # WARNING overcommit_memory is set to 0! Backgr
ound save may fail under low memory condition. To fix this issue add 'vm.overco
mmit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl
vm.overcommit_memory=1' for this to take effect.
4670:M 08 Mar 2022 02:50:24.784 * Loading RDB produced by version 6.2.6
4670:M 08 Mar 2022 02:50:24.784 * RDB age 271 seconds
4670:M 08 Mar 2022 02:50:24.784 * RDB memory usage when created 0.77 Mb
4670:M 08 Mar 2022 02:50:24.784 # Done loading RDB, Keys Loaded: 1, keys expire
d: 0.
4670:M 08 Mar 2022 02:50:24.784 * DB loaded from disk: 0.000 seconds
4670:M 08 Mar 2022 02:50:24.784 * Ready to accept connections

[ec2-user@ip-172-31-82-54 redis-stable]$ redis-cli
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> ^C
[ec2-user@ip-172-31-82-54 redis-stable]$ redis-cli
127.0.0.1:6379> ping
(error) NOAUTH Authentication required.
127.0.0.1:6379> AUTH sizaspai0
OK
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> ping
Error: Broken pipe
127.0.0.1:6379> ping
(error) NOAUTH Authentication required.
127.0.0.1:6379> auth
(error) ERR wrong number of arguments for 'auth' command
127.0.0.1:6379> auth sizaspai0
OK
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> ping [message]

```

## Redis examples from Seven Databases in Seven Weeks

```

redis-cli
127.0.0.1:6379> AUTH M1V3r1S7R0NGP455
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> 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> SET count 2
OK
127.0.0.1:6379> INCR count
(integer) 3
127.0.0.1:6379> GET count"3"
127.0.0.1:6379> SET bad_count "a"
OK
127.0.0.1:6379> INCR bad_count(error) ERR value is not an integer or out of
range
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> SET prag http://pragprog.comQUEUED
127.0.0.1:6379(TX)> INCR count
QUEUED
127.0.0.1:6379(TX)> EXEC
1) OK
2) (integer) 4

```

```

ec2-user@ip-172-31-82-54:~/redis-stable
127.0.0.1:6379>
127.0.0.1:6379> auth sizaspaio
OK
127.0.0.1:6379>
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> GET 7wks
"http://www.sevenweeks.org/"
127.0.0.1:6379>
127.0.0.1:6379>
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> 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> SET count 2
OK
127.0.0.1:6379> INCR count
(integer) 3
127.0.0.1:6379> GET count
"3"
127.0.0.1:6379> SET bad_count "a"
OK
127.0.0.1:6379> INCR bad_count
(error) ERR value is not an integer or out of range
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) 4
127.0.0.1:6379>
[0] 0:ec2-user@ip-172-31-82-54:~/redis-stable*Z "ip-172-31-82-54.ec2.in" 03:25 08-Mar-22
ec2-user@ip-172-31-82-54:~/redis-stable sflores@ramiel:~/Documents/University/Special Topics on Telematics

```

## Unbind redis from loopback

By default redis is bound to the loopback interface, if we want to access redis from outside we need to comment the specific line in the config.

```
sed 's/bind 127.0.0.1 -:::1/# bind 127.0.0.1 -:::1/g' redis.conf
```

## Python examples

You can find some examples in this directory for a simple cli program implementing the CRUD operations.

The basic usage of the programs:

```
usage: redis_<operation> [-h] [-H HOST] [-p PORT] [-k KEY] [-v VALUE]
```

Example python application to **test** the create operation **in** redis.

options:

- h, --**help** show this **help** message and **exit**
- H HOST, --host HOST IP of the host to connect to.
- p PORT, --port PORT Port **in which** the redis server is listening **in** the host.
- k KEY, --key KEY Key of the value.
- v VALUE, --value VALUE Value.

```
# redis_create.py
> python redis_create.py -H 184.73.121.22 -p 6379 -k password -v
M1V3r1S7R0NGP455
> Password:
Setting "password" to "M1V3r1S7R0NGP455"
password was set successfully!
```

```
# redis_read.py
> python redis_read.py -H 184.73.121.22 -p 6379 -k password
> Password:
Searching for "password"
password = M1V3r1S7R0NGP455
```

```
# redis_update.py
> python redis_update.py -H 184.73.121.22 -p 6379 -k password -v >
AN07H3R_P455
Password:
Searching for "password"
key "password" was set to "M1V3r1S7R0NGP455"
key "password" is now set to "AN07H3R_P455"
```

```
# redis_delete.py
> python redis_delete.py -H 184.73.121.22 -p 6379 -k password
> Password:
Searching for "password"
key found, proceeding to delete!
key "password" was deleted!
```

## Cluster

Redis cluster uses the port 16379 for inter node bus communications. You need to enable access to that port in your firewall (or security group).

For the cluster we need more machines running the server. If you are running an AWS EC2 instance, you may create an AMI with a snapshot of your current instance.

Also, we need to enable the cluster related options in the `redis.conf` file.

You may run this command to do so automatically.

```
sed '/# cluster-/s/^# //' -i redis.conf
```

And then run `redis-server redis.conf` on all the nodes.

In my case, I am using 4 nodes, 3 of which were cloned from the one used in the past examples.

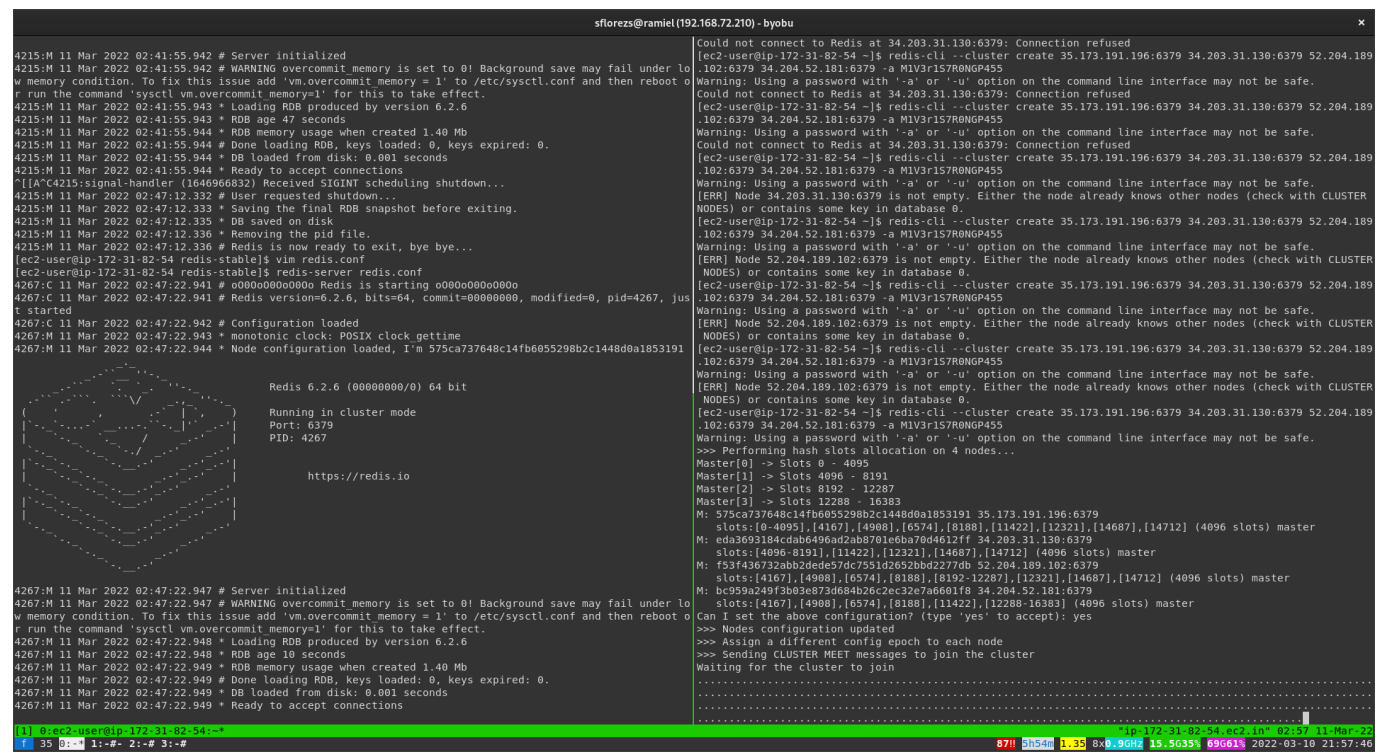
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IF
<input type="checkbox"/>	-	<a href="#">i-09247ec5634d3cb23</a>	<span>Running</span>	t2.micro	<span>Initializing</span>	No alarms +	us-east-1a	ec2-35-173-191-196.co...	35.173.191.196	-	-
<input type="checkbox"/>	-	<a href="#">i-022ee0154a6ea3dde</a>	<span>Running</span>	t2.micro	<span>Initializing</span>	No alarms +	us-east-1b	ec2-34-203-31-130.co...	34.203.31.130	-	-
<input type="checkbox"/>	-	<a href="#">i-0f8ad8efed21a1ac6</a>	<span>Running</span>	t2.micro	<span>Initializing</span>	No alarms +	us-east-1b	ec2-52-204-189-102.co...	52.204.189.102	-	-
<input type="checkbox"/>	-	<a href="#">i-0c372684dd713cb0c</a>	<span>Running</span>	t2.micro	-	No alarms +	us-east-1b	ec2-34-204-52-181.co...	34.204.52.181	-	-

Select a node randomly to create the cluster, and run:

```
redis-cli --cluster create node1:port node2:port node3:port node4:port -a M1V3r1S7R0NGP455
```

note: replace node1 with the public ip for node1 and port with the port in which de server is running.

Now it should ask for a configuration



Sadly I had a problem which I could not solve, the machines were able to identify each other, and some were registered on the others node list, however the command stuck on "waiting for "waiting for the cluster to join" which judging by the online discussions seems to be a common problem.

## Conclusions

For the first project we want to create an API (the spec can be found in the Github repository in the Project 1 folder) and a Python client similar to the one used with StrictRedis. We think that implementing just the basic CRUD operations should be relatively easy, except for delete depending on the disk save method we decide to use. One of the non-functional specifications for this project that we will most certainly use will be partitioning (albeit only statically) using a simple DHT algorithm. Replication will not be a priority, we understand how useful it is however we think we should focus our attention on other features. Our project

should be consistent since there will only be 3 data nodes and most likely they will not have replication. Fault tolerance will therefore not be guaranteed, however it will not be completely ignored. From this lab with Redis we had a good approach to the concept of both key-value databases and distributed databases, although they may seem relatively easier to use given their apparent simplicity, they should not be underestimated at all. They are certainly powerful tools for the functions they are intended to perform and looking at the different methods that Redis contains has given us a good idea of what kind of operations we should handle and how to present them.