Run the server

First of all, we need to get the server running. For it, in the folder server we execute

ruby server.rb {port we wish to use}

As we can see in the next image.

```
[sebapandolfi@192 server]$ ruby server.rb 10001
Started server.........
^C
Shutting down gracefully...
[sebapandolfi@192 server]$
```

To shutdown the server correctly we can do it with ctrl + c or ctrl + z.

Run the client

The client only works, if it is run in the same machine that the server. That is because it is configured to use the interface 127.0.0.1 that is local to the machine. We can simply solve this changing the next line.

```
socket = TCPSocket.open( "127.0.0.1", ARGV[0] )
```

Instead of using 127.0.0.1, we configure the IP address where the server is running.
To run the client, in the folder client we execute

ruby client.rb {same port we use previously}

Once it's running the first message must be of authentication. For this we send a fake set message and after that the user and password.
After this we can start sending requests. This must be done following the structure defined in the protocol's documentation.
The next image is a sample of this exchanging message.

```
[sebapandolfi@192 client]$ ruby client.rb 10001
Please enter your username to establish a connection...
set
seba pass
STORED
set 1 1 100 4
test
STORED
set 2 2 200 5
hello
STORED
get 1 2
VALUE 1 1 4
test
VALUE 2 2 5
hello
END
SERVER_ERROR, closing connection
[sebapandolfi@192 client]$
```

Running tests

The files for the unit test are in the folder spec. The file serverTest.rb is a copy of the original server.rb but without the last line.

```
Server.new( ARGV[0], "0.0.0.0" ) # listen in all interfaces
```

To run the unit tests in the folder cache we execute

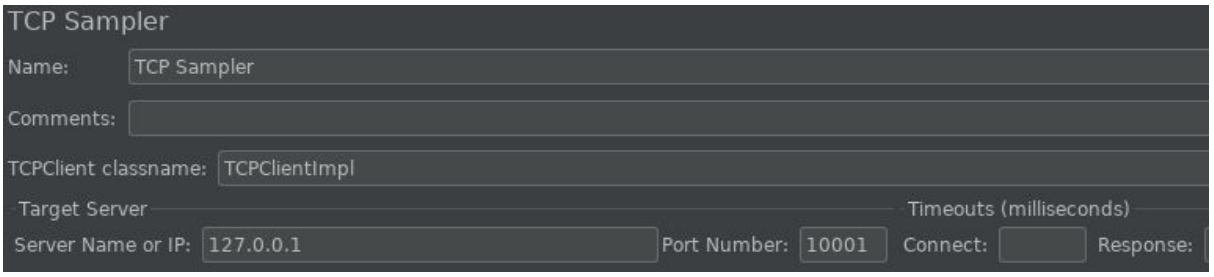
```
rspec
```

If everything is correct, after a short delay because of the expiration time tests, the test's finish with 36 correct examples.

```
[sebapandolfi@192 cache]$ rspec
Started server.........
....................................

Finished in 12.15 seconds (files took 0.10399 seconds to load)
36 examples, 0 failures
```

The file for the load test is Test Plan.jmx, in the folder apache-jmeter-5.3/bin. Remember that the server must be running for requests to be answered. And the IP address and port in the tcp sampler must be correctly configured. In this case, it is using the loopback address and the port 10001. The server must be running using the same port.



This file run's 1000 sample client's in 10 seconds, every client sends 1 message of authentication, 1 message of set and 1 message of get.
To run this file we have 2 options, in the graphical interface or from the terminal.
From the terminal, in the folder apache-jmeter-5.3/bin we execute

```
./jmeter -n -t 'Test Plan'.jmx -l results.jtl
```

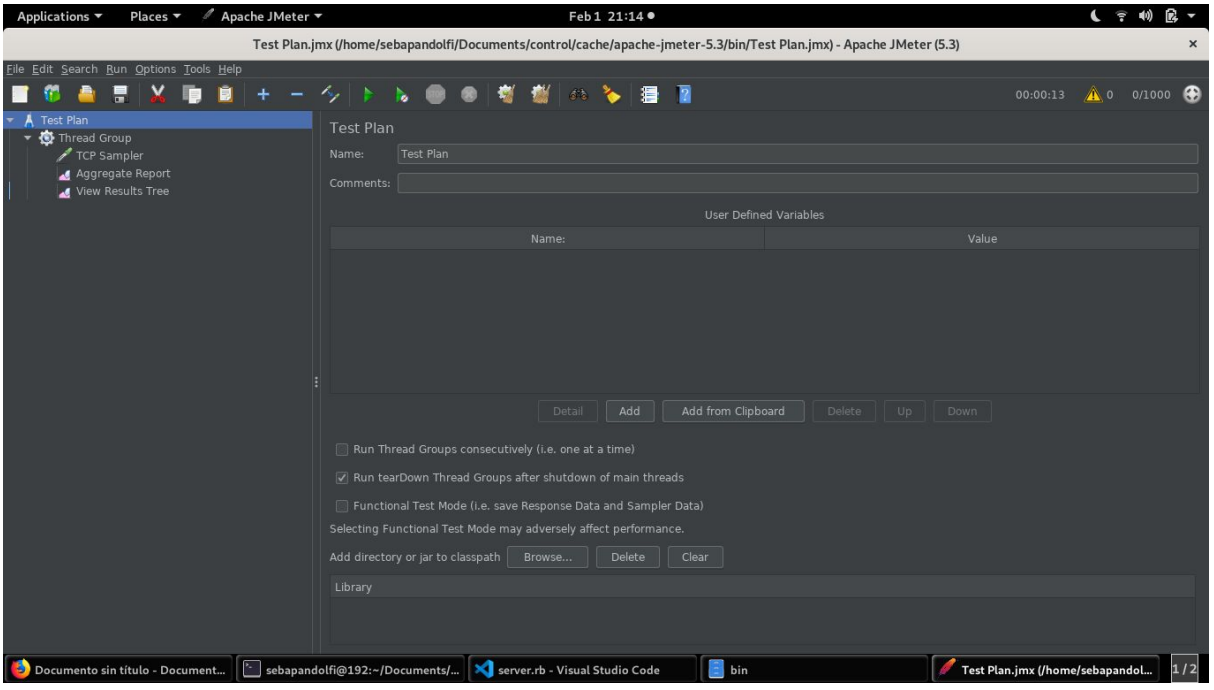
To end the test we need to kill the server with ctrl + c or ctrl + z. The results of this test are saved in the same folder under the name of results.jtl.

```
[sebapandolfi@192 bin]$ ./jmeter -n -t 'Test Plan.jmx' -l results.jtl
Creating summariser <summary>
Created the tree successfully using Test Plan.jmx
Starting standalone test @ Mon Feb 01 21:06:31 UYT 2021 (1612224391038)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary =   1000 in 00:00:36 =   27.7/s Avg: 30908 Min: 25933 Max: 35812 Err:     0 (0.00%)
Tidying up ...    @ Mon Feb 01 21:07:07 UYT 2021 (1612224427620)
... end of run
```

The graphical interface is open in the folder apache-jmeter-5.3/bin with the command

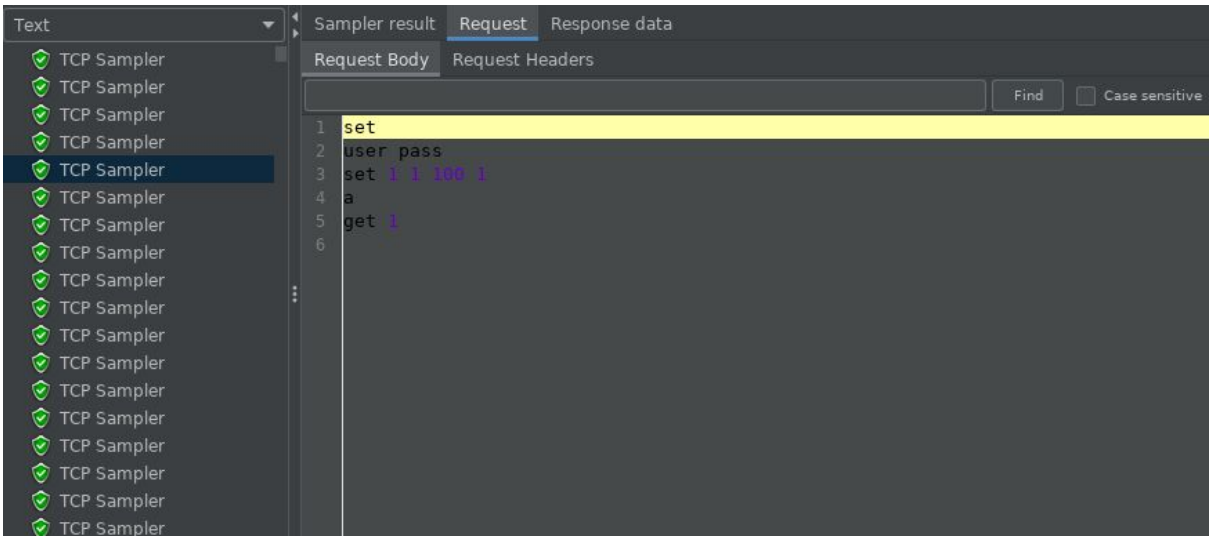
```
./jmeter
```

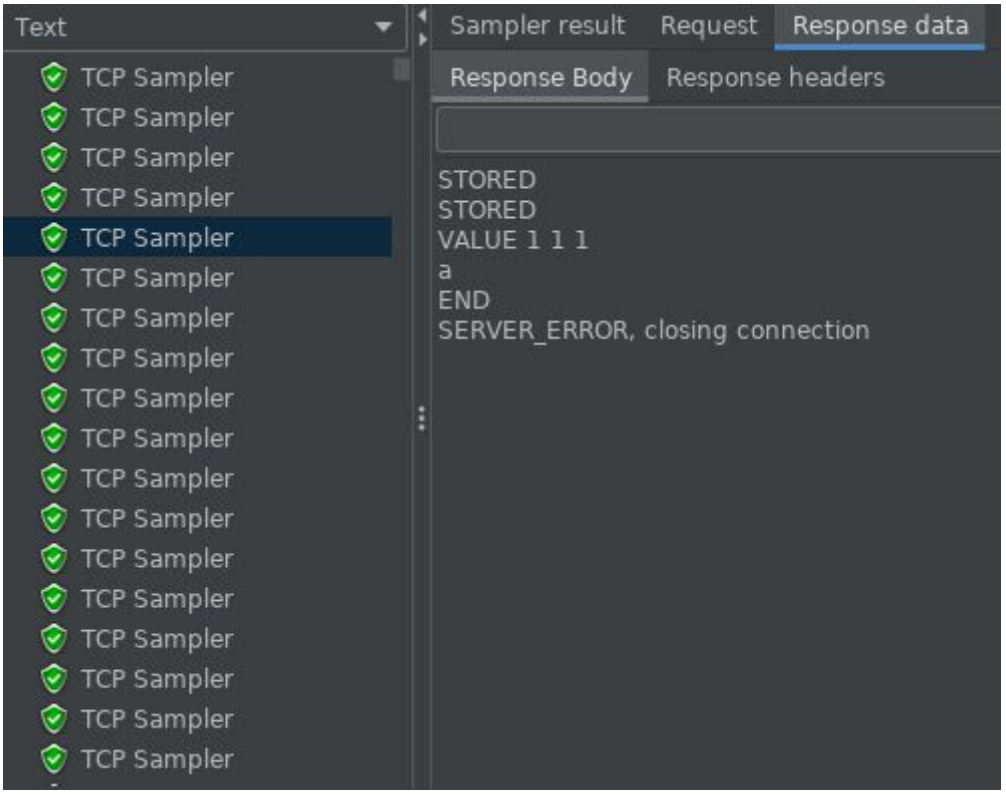
Once in there we open the file of Test Plan.jmx and we press play with the green arrow.

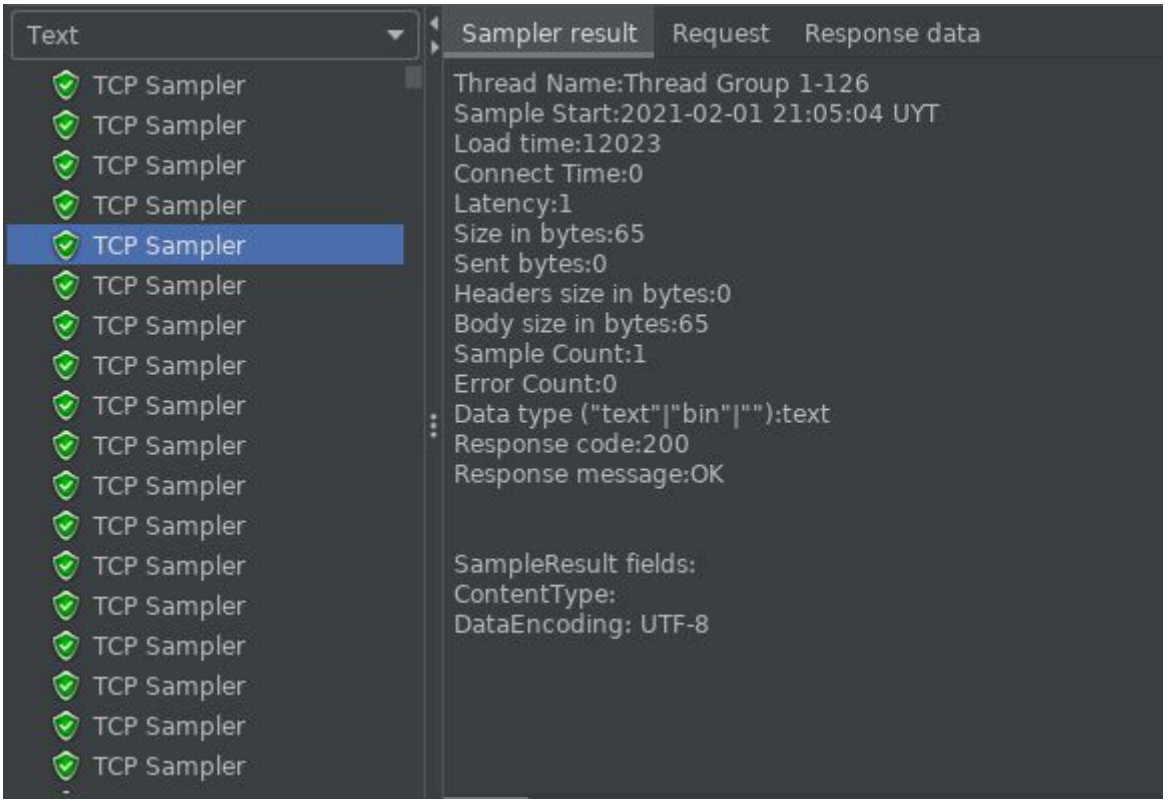


Then it starts running. In the upper right corner it shows the seconds elapsed since the start and the number of tcp sampler executed from the total. Once it reaches the total of 1000, we need to kill the server.
The listeners show the result.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Through...	Received...	Sent KB/...
TCP Sam...	1000	8321	8341	12274	12775	13181	3333	13265	0.00%	75.0/sec	4.76	0.00
TOTAL	1000	8321	8341	12274	12775	13181	3333	13265	0.00%	75.0/sec	4.76	0.00







As we can see all the clients receive the correct response, the time dont give much information because it depends on the moment we kill the server.
The important information is that it can answer 100 clients per second without problems.