After the replica receives a command from the master, it processes it and apply it to its own state. This will work exactly like a regular command sent by a client, except that the replica doesn't send a response back to the master.

For example, if the command `SET foo 1` is propagated to the replica by a master, the replica must update its database to set the value of `foo` to `1`. Unlike commands from a regular client though, it must not reply with `+OK\r\n`.

Tests

The tester will spawn a Redis master, and it'll then execute your program as a replica like this:

```
./spawn_redis_server.sh --port <PORT> --replicaof "<MASTER_HOST> <MASTER_PORT>"
```

Just like in the previous stages, your replica should complete the handshake with the master and receive an empty RDB file.

Once the RDB file is received, the master will propagate a series of write commands to your program.

```
SET foo 1 # propagated from master to replica
SET bar 2 # propagated from master to replica
SET baz 3 # propagated from master to replica
```

The tester will then issue `GET` commands to your program to check if the commands were processed correctly.

```
$ redis-cli GET foo # expecting `1` back
$ redis-cli GET bar # expecting `2` back
# ... and so on
```

Notes