	T0	T1	T2	T3	T4	Other
0	rm -rf data1 data2 data3					
1	mkdir data1 data2 data3					
2		mongod --port 27001 --replSet practReplica --dbpath ./data1 --oplogSize 50				
3			mongod --port 27002 --replSet practReplica --dbpath ./data2 --oplogSize 50			
4				mongod --port 27003 --replSet practReplica --dbpath ./data3 --oplogSize 50		
5	mongo --port 27001					
6	config = {_id: "practReplica", members: [{_id:1, host:hostname()+":27001" }, {_id:2, host:hostname()+":27002" }, {_id:3, host:hostname()+":27003" }]}					
7	rs.initiate(config)					
8				Ctr-C		
9	rs.status()					
10						lsof -iTCP -sTCP:LISTEN | grep mongo
11			Observar los mensajes dados.			
12	use pruebas					
13	db.tururu.insert({dato:"tuturu"}, { writeConcern: { w:3, wtimeout: 1500 }})					
14				mongod --port 27003 --replSet practReplica --dbpath ./data3 --oplogSize 50		
15	use pruebas					
16	db.tururu.drop()					

	T0	T1	T2	T3	T4	Other
17	for (i=0;i<1000;i++) {db.tururu.insert({dato:"tururu"+i});}					
18	db.tururu.find().count()					
19	db.tururu.find().pretty()					
20	Ctr-d					
21		Ctr-C				
22						Borramos data1
23	mongo --port 27002					
24	use pruebas					
25	db.tururu.find().count()					
26	Ctr-d					
27	mongo --port 27003					
28	use pruebas					
29	db.tururu.find().count()					
30	Ctr-d					
31	mongo --port 27002					
32		mkdir data1				
33		mongod --port 27001 --replSet practReplica --dbpath ./data1 --oplogSize 50				
34	Ctr-d					
35	mongo --port 27001					
36	use pruebas					

	T0	T1	T2	T3	T4	Other
37	db.tururu.find().readPref("primaryPreferred")					
38	rs.status()					
39	Ctr-d					
40	mongo --port 27002 (Primario)					
41	use pruebas					
42	db.tururu.find().readPref("secondaryPreferred")					
43					mkdir data4	
44					mongod --port 27004 --replSet practReplica --dbpath ./data4 --oplogSize 50	
45	rs.status()					
46	rs.add("MacBook-Pro-de-Miguel.local:27004")					
47	rs.status()					
48	Ctr-d					
49	mongo --port 27002 (Primario)					
50	rs.stepDown()					
51	rs.status()					
52	Ctr-d					
53	mongo --port 27001 (Primario)					
54	rs.remove("MacBook-Pro-de-Miguel.local:27002")					
55	rs.status()					

Respuestas para la pregunta 1:

Todos los comandos en la tabla superior.

Respuestas para la pregunta 2:

Desde T0:

```
rs.status()
> ……”members” : [
        {
            "_id" : 1,
            "name" : "MacBook-Pro-de-Miguel.local:27001",
            "ip" : "127.0.0.1",
            "health" : 1,
            "state" : 1,
……
        {
            "_id" : 3,
            "name" : "MacBook-Pro-de-Miguel.local:27003",
            "ip" : "127.0.0.1",
            "health" : 0,
            "state" : 8,
            "stateStr" : "(not reachable/healthy)",
            "uptime" : 0,
            "optime" : {
                "ts" : Timestamp(0, 0),
                "t" : NumberLong(-1)
            },
            "optimeDurable" : {
                "ts" : Timestamp(0, 0),
                "t" : NumberLong(-1)
            },
            "optimeDate" : ISODate("1970-01-01T00:00:00Z"),
            "optimeDurableDate" : ISODate("1970-01-01T00:00:00Z"),
            "lastHeartbeat" : ISODate("2020-03-15T11:35:57.499Z"),
            "lastHeartbeatRecv" : ISODate("2020-03-15T11:35:49.287Z"),
            "pingMs" : NumberLong(0),
```

```
            "lastHeartbeatMessage" : "Error connecting to MacBook-Pro-de-Miguel.local:27003 (127.0.0.1:27003)
:: caused by :: Connection refused",
            "syncingTo" : "",
            "syncSourceHost" : "",
            "syncSourceId" : -1,
            "infoMessage" : "",
            "configVersion" : -1
        }
```

Desde linux:

```
>lsof -iTCP -sTCP:LISTEN | grep mongo
mongod  1671 miguelangelarroyoclemente    9u  IPv4 0x32c684360d7fd0d7      0t0  TCP localhost:27001 (LISTEN)
mongod  1677 miguelangelarroyoclemente    9u  IPv4 0x32c684360caa69b7      0t0  TCP localhost:27002 (LISTEN)
```

Falta el servidor en 27003

Desde T2:

Aparecen mensajes del tipo:

```
I  CONNPOOL [ReplicaSetMonitor-TaskExecutor] Connecting to MacBook-Pro-de-Miguel.local:27003
2020-03-15T12:41:04.382+0100
I  REPL_HB  [replexec-6] Heartbeat to MacBook-Pro-de-Miguel.local:27003 failed after 2 retries, response
status: HostUnreachable: Error connecting to MacBook-Pro-de-Miguel.local:27003 (127.0.0.1:27003) :: caused
by :: Connection refused
```

Respuesta a la pregunta 3:

Con writeConcern: { w:3, wtimeout: 1500 } le decimos que asegure la escritura en 3 servidores (w: 3) y que lo haga en menos de 1500 milisegundos, cosa que no consigue porque servidor 3 está caído, así que da el error de escritura:

```
WriteResult({
    "nInserted" : 1,
    "writeConcernError" : {
        "code" : 64,
        "codeName" : "WriteConcernFailed",
        "errmsg" : "waiting for replication timed out",
        "errInfo" : {
            "wtimeout" : true
        }
```

```
    }
})
```

Respuesta a la pregunta 4.G

Se puede entrar a ambos, pero T2 ahora es primario y T3 sigue siendo secundario, así que desde T2 si podemos acceder a los datos, pero desde T3 no por ser secundario. Excepto que usemos .readPref(“primaryPreferred | secondary | secondaryPreferred”) para leer desde un secundario.

Respuesta a la pregunta 5:

No se puede ver, pero si utilizamos la opción .readPref(“primaryPreferred | secondary | secondaryPreferred”) si se puede acceder, siempre que haya pasado un tiempo para la copia de datos.

Respuesta a la pregunta 6:

Comando 42 de la tabla.

Respuesta a la pregunta 7:

Comando 46 de la tabla.

Respuesta a la pregunta 8:

Comandos desde el 47 al 55.
En primer lugar ejecutamos rs.status() para saber cuál es el primario.
Después nos conectamos a él y lo eliminamos como primario.
Volvemos a ejecutar rs.status() para saber el nombre que necesitaremos para eliminarlo.
Nos cambiamos al primario si no estamos en él.
Lo eliminamos con rs.remove(“MacBook-Pro-de-Miguel.local:27002”)
Por ultimo ejecutamos rs.status() para ver qué se ha borrado.