

An abstract graphic on the right side of the slide, consisting of numerous thin, curved red lines that create a sense of depth and movement, resembling a stylized wave or a complex network.

HIGH-AVAILABILITY REŠENJA KOD REDIS BAZE PODATAKA

Student: Uroš Pešić 1465
Profesor: Dr Aleksandar Stanimirović

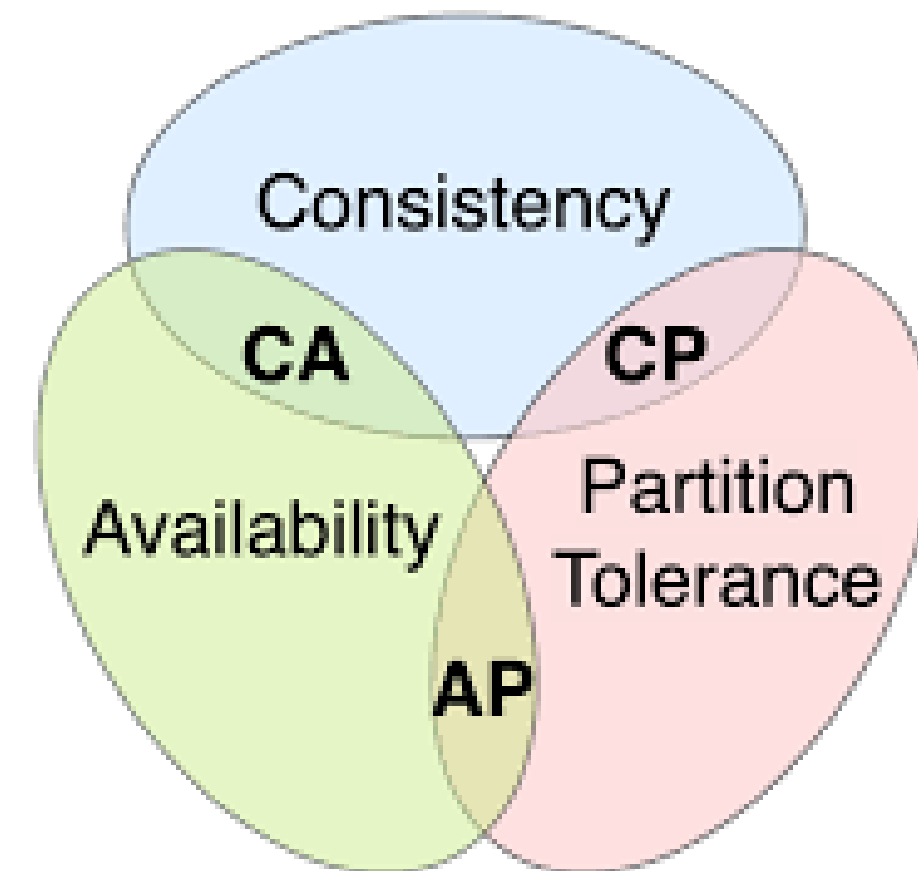
Elektronski fakultet Niš

CAP TEOREMA

U idealnim uslovima, želeli bismo da svaka distribuirana NoSQL baza podataka ima sledeća 3 svojstva:

- **Konzistentnost** – Odnosi se na konzistentnost između replika, tj. na situaciju kada sve kopije sve vreme imaju identične podatke.
- **Dostupnost** – Sistem uvek može da prihvati operaciju čitanja ili upisa.
- **Tolerantnost na otkaze čvorova** (eng. Partition Tolerance) – Ukoliko čvor u sistemu otkaze, sistem ne bi trebalo da nastavi da funkcioniše neometano.

Prema **CAP teoremi**, distribuirana u jednom trenutku je moguće ispuniti najviše dve od tri navedene osobine. U praksi se najčešće preferira dostupnost i tolerantnost na otkaze, po ceni smanjene konzistentnosti.



VISOKA DOSTUPNOST

Visoka dostupnost predstavlja kontinuirani rad sistema, uz minimalno vreme zastoja, čak i u slučajevima hardverskih ili softverskih otkaza ili bilo kakvih drugih nepredviđenih situacija.

Neke karakteristike visoko dostupnih sistema:

1 KONTINUIRANI RAD

I u slučaju grešaka ili otkaza delova sistema korisnici mogu obavljati operacije upisa i čitanja podataka. Čak i u slučajevima kada je potrebno vršiti održavanje, u visoko dostupnim sistemima, ovaj proces za korisnika mora biti potpuno transparentan.

3 MOGUĆNOST OPORAVKA

Ključno je unapred detektovati koje su moguće greške koje se mogu javiti i definisati najefikasniji načini oporavka od tih grešaka, kako bi se zadovoljili kriterijumi visoke dostupnosti.

2 BLAGOVREMENA DETEKCIJA GREŠAKA

Ukoliko dođe do greške, u visoko dostupnom sistemu je potrebno otkriti njen uzrok u nekom doglednom vremenskom intervalu. Zbog toga monitoring svih komponenti u sistemu predstavlja jedan od ključnih procesa u visoko dostupnim rešenjima.

4 POUZDANOST

Visoko dostupan sistem mora da bude sačinjen od pouzdanih hardverskih i softverskih komponenti.

UZROCI NARUŠAVANJA DOSTUPNOSTI

NEPLANIRANI

- Kvar u datacentru
- Otkaz klastera
- Otkaz čvora u klasteru
- Mrežni otkazi
- Otkaz skladišta podataka
- Oštećeni podaci
- Zalutali i izgubljeni upisi
- Ljudski faktor
- Kašnjenje sistema usled nedostatka resursa

PLANIRANI

- Promene softvera
- Promene na nivou sistema
- Promene podataka
- Promena u aplikaciji

POSTIZANJE VISOKE DOSTUPNOSTI

Da bi se postigla visoka dostupnost sistema posebno je obezbediti:

1 REDUNDANTNOST

Kritične komponente, čiji bi otkaz izazvao zastoj čitavog sistema je potrebno multiplicirati, kako bi u slučaju otkaza neke od njih, njena druga instanca preuzela njene odgovornosti. Na taj način se eliminišu jedinstvene tačke otkaza (**Single Point of Failure**).

3 AUTOMATSKI FAILOVER

Mehanizam koji bi u slučaju otkaza neke komponente, automatski prebacio njenu odgovornost na rezervnu repliku te komponente. Na taj način se obezbeđuje minimalno vreme zastoja u slučaju otkaza u sistemu.

2 DETEKCIJA GREŠAKA

Visoko dostupni sistemi moraju imati podsistem zadužen za monitoring svih ostalih komponenti, kako bi se otkaz, ukoliko do njega dođe, detektovao što ranije i obezbedio automatski failover.

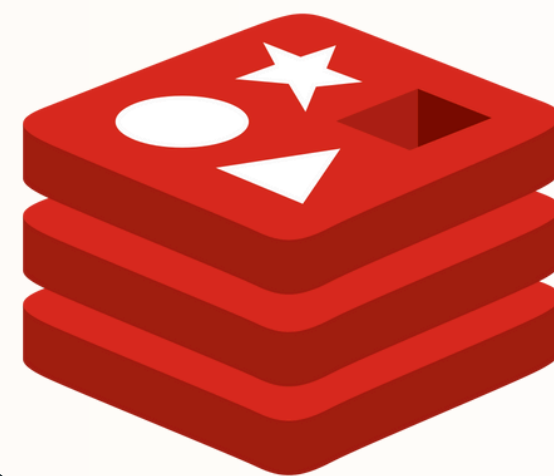
4 BALANSIRANJE OPTEREĆENJA*

Replikacijom se, osim redundantnosti, postiže i mogućnost boljeg balansiranja opterećenja čitavog sistema. U slučaju da imamo više replika baze podataka, moguće je koristiti bilo koju od njih za usluživanje operacije čitanja podataka. Posebna komponenta sistema – load balancer tada može prosleđivati zahteve ka odgovarajućoj replici, shodno njihovim trenutnim opterećenjima.

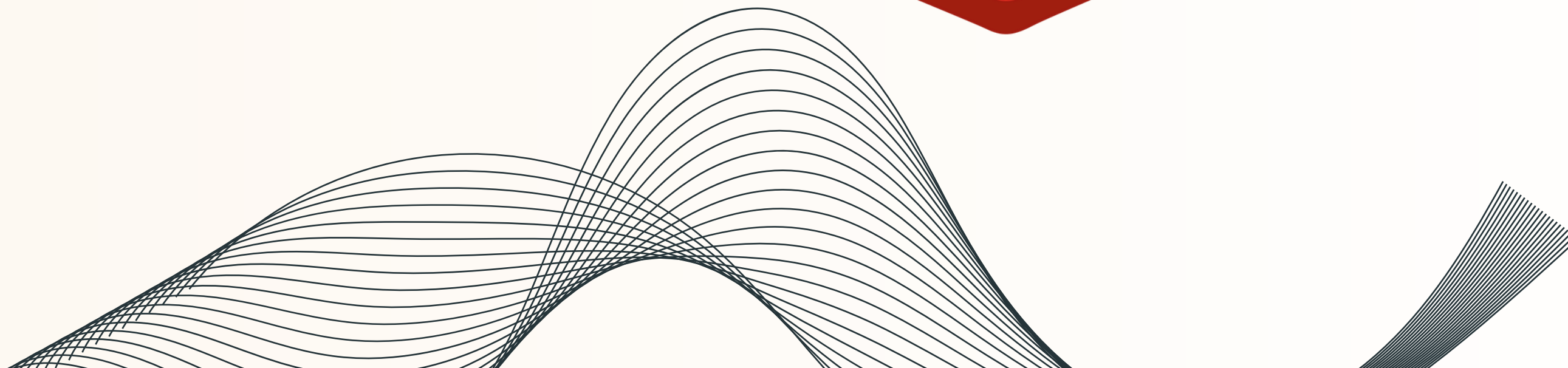
REDIS HIGH-AVAILABILITY

Redis (Remote Dictionary Server) je distribuirana key-value NoSQL baza podataka. Specifična je po tome što podatke čuva u glavnoj memoriji (in-memory store) i zahvaljujući toj činjenici ima jako dobre performanse upisa i čitanja. Opciono je moguće omogućiti i perzistenciju na disk, kako ne bi došlo do gubitka podataka u slučaju otkaza, ili restartovanja čvora na kome se izvršava. Zbog ovih svojih osobina se često koristi za potrebe keširanja podataka, implementaciju real-time rang listi, redova poruka, a zahvaljujući svom publish/subscribe mehanizmu i za real-time razmenu poruka.

Za postizanje visoke dostupnosti, Redis koristi tehnike replikacije (kako Active/Passive) i automatski failover korišćenjem Redis Sentinela.

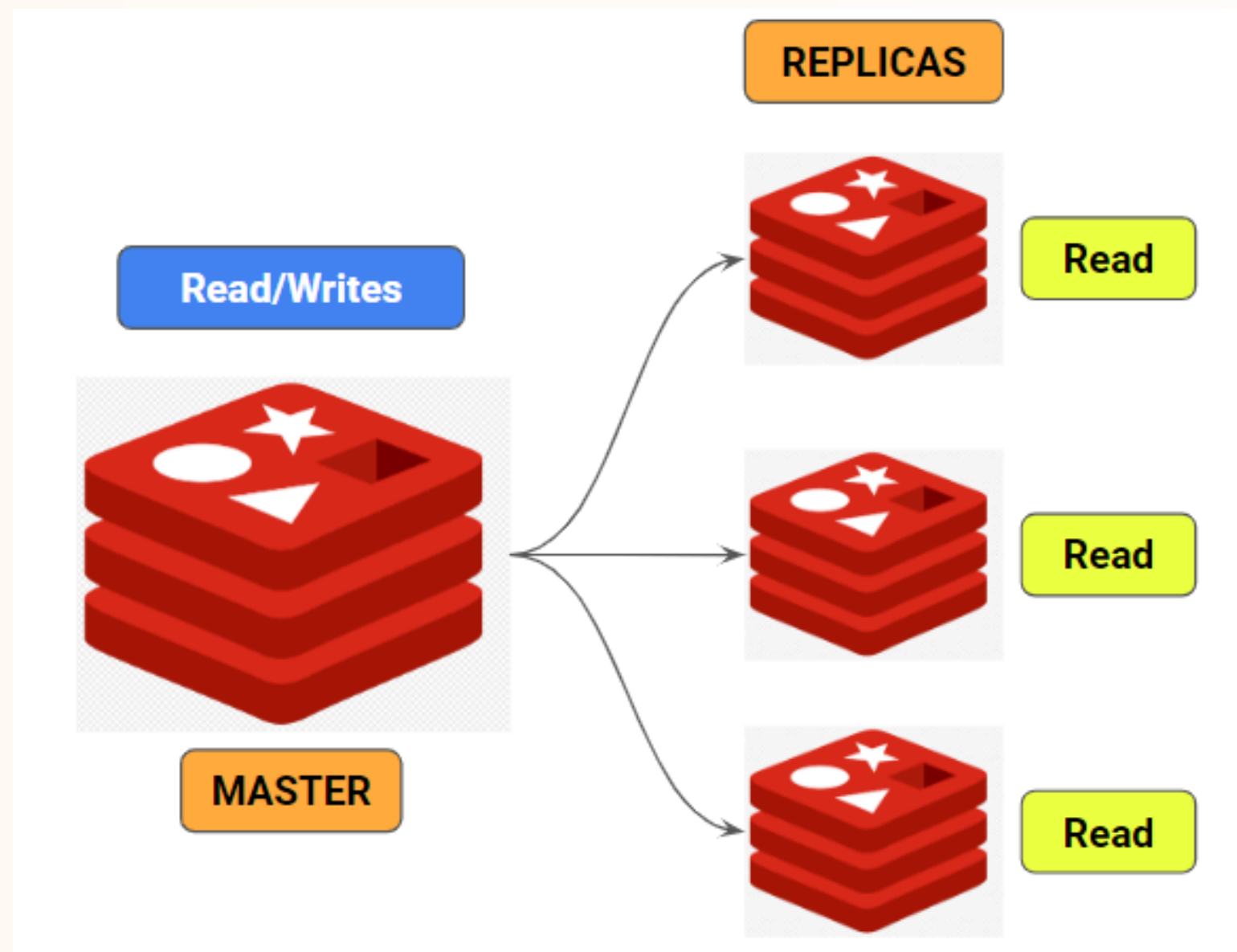


redis



REDIS REPLIKACIJA

Redis podržava Leader-Follower (Master-Slave) replikaciju koja je jednostavna za konfiguraciju. Svaka replika predstavlja identičnu kopiju master instance. Ukoliko dođe do prekida veze između mastera i replika, replike će automatski probati da se povežu nazad na master instancu i, ukoliko to uspešno učini, dobiće instrukcije kako bi prihvatile nove podatke koji su u međuvremenu bili upisani na masteru.



POTPUNA RESIHNRONIZACIJA

```
5014:C 08 Jul 2024 20:07:59.035 * Fork CoW for RDB: current 0 MB, peak 0 MB, average 0 MB
4693:M 08 Jul 2024 20:07:59.132 * Background saving terminated with success
4693:M 08 Jul 2024 20:10:44.062 * Replica 127.0.0.1:6380 asks for synchronization
4693:M 08 Jul 2024 20:10:44.062 * Full resync requested by replica 127.0.0.1:6380
4693:M 08 Jul 2024 20:10:44.062 * Replication backlog created, my new replication IDs are 'b5b17283881a4cae7bb7ceb4513d0a9dcdc1b2' and '0000000000000000000000000000000000000000000000000000000000000000'
4693:M 08 Jul 2024 20:10:44.062 * Delay next BGSAVE for diskless SYNC
4693:M 08 Jul 2024 20:10:49.696 * Starting BGSAVE for SYNC with target: replicas sockets
4693:M 08 Jul 2024 20:10:49.697 * Background RDB transfer started by pid 5255
5255:C 08 Jul 2024 20:10:49.701 * Fork CoW for RDB: current 0 MB, peak 0 MB, average 0 MB
4693:M 08 Jul 2024 20:10:49.701 * Diskless rdb transfer, done reading from pipe, 1 replicas still up.
4693:M 08 Jul 2024 20:10:49.710 * Background RDB transfer terminated with success
4693:M 08 Jul 2024 20:10:49.710 * Streamed RDB transfer with replica 127.0.0.1:6380 succeeded (socket). Waiting for REPLCONF ACK from replica to enable streaming
4693:M 08 Jul 2024 20:10:49.710 * Synchronization with replica 127.0.0.1:6380 succeeded
```

MASTER



Redis 7.2.5 (00000000/0) 64 bit

Running in standalone mode
Port: 6380
PID: 5250

<https://redis.io>

```
5250:S 08 Jul 2024 20:10:44.058 * Server initialized
5250:S 08 Jul 2024 20:10:44.058 * Loading RDB produced by version 7.2.5
5250:S 08 Jul 2024 20:10:44.061 * RDB age 165 seconds
5250:S 08 Jul 2024 20:10:44.061 * RDB memory usage when created 1.13 Mb
5250:S 08 Jul 2024 20:10:44.061 * Done loading RDB, keys loaded: 1, keys expired: 0.
5250:S 08 Jul 2024 20:10:44.061 * DB loaded from disk: 0.003 seconds
5250:S 08 Jul 2024 20:10:44.061 * Ready to accept connections tcp
5250:S 08 Jul 2024 20:10:44.061 * Connecting to MASTER 127.0.0.1:6379
5250:S 08 Jul 2024 20:10:44.061 * MASTER <-> REPLICAsync started
5250:S 08 Jul 2024 20:10:44.061 * Non blocking connect for SYNC fired the event.
5250:S 08 Jul 2024 20:10:44.062 * Master replied to PING, replication can continue...
5250:S 08 Jul 2024 20:10:44.062 * Partial resynchronization not possible (no cached master)
5250:S 08 Jul 2024 20:10:49.696 * Full resync from master: b5b17283881a4cae7bb7ceb4513d0a9dcdc1b2:17
5250:S 08 Jul 2024 20:10:49.699 * MASTER <-> REPLICAsync: receiving streamed RDB from master with EOF to disk
5250:S 08 Jul 2024 20:10:49.699 * MASTER <-> REPLICAsync: Flushing old data
5250:S 08 Jul 2024 20:10:49.699 * MASTER <-> REPLICAsync: Loading DB in memory
5250:S 08 Jul 2024 20:10:49.709 * Loading RDB produced by version 7.2.5
5250:S 08 Jul 2024 20:10:49.709 * RDB age 0 seconds
5250:S 08 Jul 2024 20:10:49.710 * RDB memory usage when created 1.28 Mb
5250:S 08 Jul 2024 20:10:49.710 * Done loading RDB, keys loaded: 3, keys expired: 0.
5250:S 08 Jul 2024 20:10:49.710 * MASTER <-> REPLICAsync: Finished with success
```

REPLICA

PARCIJALNA RESIHNRONIZACIJA

```
4693:M 08 Jul 2024 20:39:31.644 * Connection with replica 127.0.0.1:6380 lost.
4693:M 08 Jul 2024 20:39:39.675 * Replica 127.0.0.1:6380 asks for synchronization
4693:M 08 Jul 2024 20:39:39.675 * Partial resynchronization request from 127.0.0.1:6380 accepted. Sending 80 bytes of backlog starting from offset 2412.
```

MASTER



Redis 7.2.5 (00000000/0) 64 bit

Running in standalone mode
Port: 6380
PID: 5907

<https://redis.io>

REPLICA

```
5907:S 08 Jul 2024 20:39:39.666 * Server initialized
5907:S 08 Jul 2024 20:39:39.674 * Loading RDB produced by version 7.2.5
5907:S 08 Jul 2024 20:39:39.674 * RDB age 8 seconds
5907:S 08 Jul 2024 20:39:39.674 * RDB memory usage when created 1.28 Mb
5907:S 08 Jul 2024 20:39:39.674 * Done loading RDB, keys loaded: 3, keys expired: 0.
5907:S 08 Jul 2024 20:39:39.674 * DB loaded from disk: 0.000 seconds
5907:S 08 Jul 2024 20:39:39.674 * Before turning into a replica, using my own master parameters to synthesize a cached master: I may be able to synchronize with the new master with just a partial transfer.
5907:S 08 Jul 2024 20:39:39.674 * Ready to accept connections tcp
5907:S 08 Jul 2024 20:39:39.674 * Connecting to MASTER 127.0.0.1:6379
5907:S 08 Jul 2024 20:39:39.674 * MASTER <-> REPLICA sync started
5907:S 08 Jul 2024 20:39:39.674 * Non blocking connect for SYNC fired the event.
5907:S 08 Jul 2024 20:39:39.675 * Master replied to PING, replication can continue...
5907:S 08 Jul 2024 20:39:39.675 * Trying a partial resynchronization (request b5b17283881a4caebc7bb7ceb4513d0a9dcdc1b2:2412).
5907:S 08 Jul 2024 20:39:39.675 * Successful partial resynchronization with master.
5907:S 08 Jul 2024 20:39:39.675 * MASTER <-> REPLICA sync: Master accepted a Partial Resynchronization.
```

| WAIT KOMANDA – BOLJA KONZISTENTNOST

Redis podrazumevano koristi asinhronu replikaciju – Nakon što se operacija uspešno izvrši na master instanci i bude prosleđena svim replikama, master se ne blokira čekajući da je i replike izvrše, već nastavlja da prima dalje zahteve. Ovakvom implementacijom se održavaju visoke performanse i minimalno kašnjenje sistema. Ukoliko je potrebno, moguće je predefinisati ovakvo ponašanje korišćenjem WAIT komande nakon prilikom zadavanja operacija. WAIT komanda ima sledeći oblik:

WAIT broj_replika timeout;

WAIT funkcioniše tako što će da blokira sistem dok broj replika (definisan prvim parametrom) nije potvrdio uspešno izvršenje emitovane operacije, ili dok ne istekne *timeout* milisekundi. Ovakva sinhrona logika potencira bolju konzistentnost po ceni manje dostupnosti i generalno lošijih performansi sistema, pa se izbegava u visoko dostupnim sistemima, osim ukoliko nije apsolutno neophodna.

READ-ONLY REPLIKE

Za postizanje bolje dostupnosti i boljih performansi sistema u kojima su operacije čitanja značajno učestalije od operacija upisa, moguće je omogućiti čitanje iz replika navođenjem parametra `replica-read-only yes` u konfiguracionom fajlu. U slučaju read-only replika je moguće balansirati opterećenje prosleđivanjem zahteva za čitanje različitim instancama, dok će operacije čitanja biti odbijene.

```
>_ CLI
Connecting...

Pinging Redis server on 127.0.0.1:6380
Connected.
Ready to execute commands.

> SET 6 "can i do this?"
"READONLY You can't write against a read only replica."

> GET 5
"new data while replica is down"
```


KONFIGURACIJA REPLIKACIJE

Prilikom startovanja Redis instance, potrebno je navesti putanju do konfiguracionog fajla koji sadrži potrebne parametre (Redis prilikom instalacije dolazi uz podrazumevani redis.conf fajl). Za kreiranje replike, potrebno je u konfiguracionom fajlu nove instance navesti parametre:

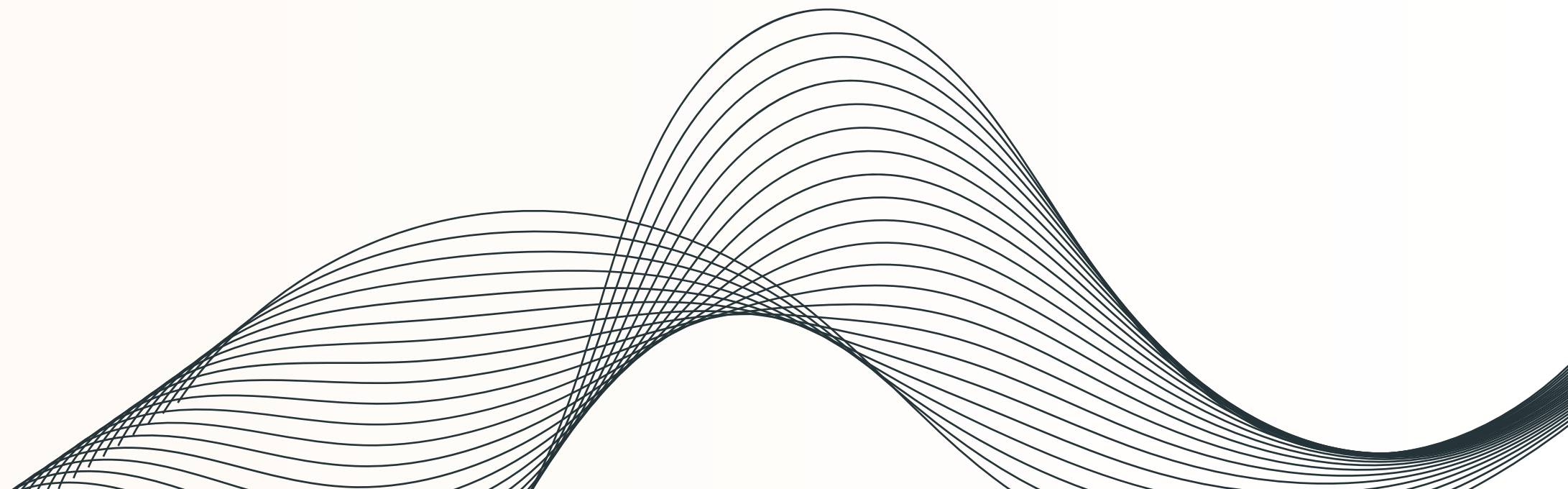
port „broj porta“, gde se broj porta naravno mora razlikovati od broja porta master instance
replicaof „IP adresa mastera“ „port mastera“

```
1 # replication|
2 port 6380
3 replicaof 127.0.0.1 6379
4
5 # read-only, allow stale data
6
7 replica-read-only yes
8 replica-serve-stale-data yes
9
10 # auth to master, require auth for replica
11
12 masterauth uros2307
13 requirepass uros2307
```

JOŠ KONFIGURACIJE

Dodatno, moguće je specificirati još neke parametre:

- **replica-read-only yes** (yes je podrazumevana vrednost ovog parametra)
- **replica-serve-stale-data** – mogućnost čitanja i potencijalno zastarelih podataka iz replike. Narušava konzistentnost po ceni boljih performansi čitanja (jer je proces sinhronizacije asinhroni)
- **masterauth „master lozinka“** – ukoliko je master instanca zaštićena lozinkom, nju je potrebno navesti prilikom startovanja replike
- **requirepass „lozinka“** – ukoliko želimo da i replika bude zaštićena lozinkom, to možemo uraditi ovim parametrom
- **min-replicas-to-write „broj replika“**
min-replicas-max-lag „broj sekundi“



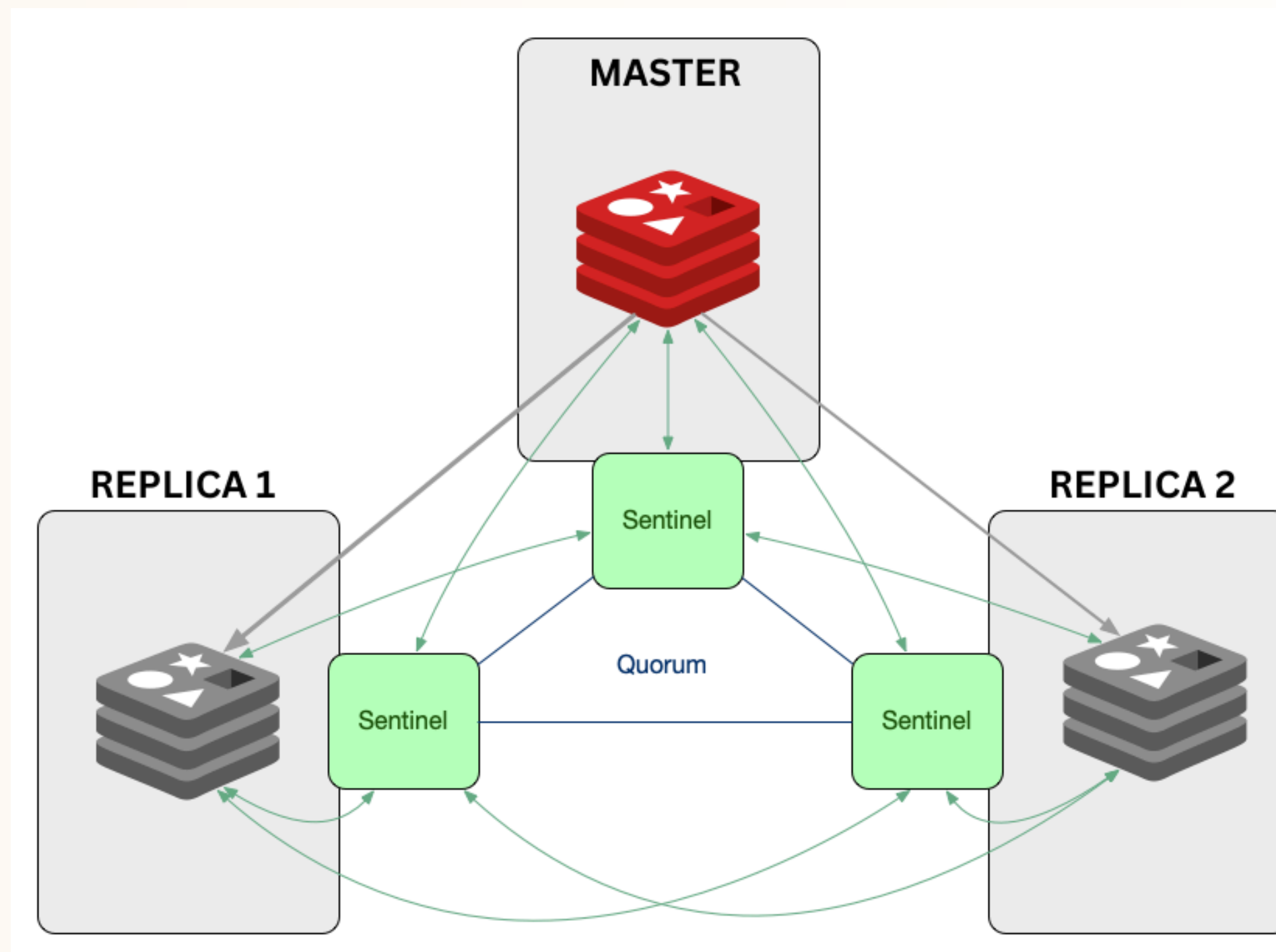
AUTOMATSKI FAILOVER – REDIS SENTINEL

Redis Sentinel predstavlja rešenje koje obezbeđuje automatski failover za postizanje visoke dostupnosti u Redis non-cluster sistemima. Osim ove glavne funkcionalnosti, Redis Sentinel obavlja i druge zadatke, ključne za visoko dostupne sisteme:

- **Monitoring** – Redis Sentinel konstantno nadgleda status master instance i replika kojima je dodeljen.
- **Obaveštavanje** – Posедуje mogućnost da notifikuje administratora, ili druge programe, kada dođe do zastoja neke od instanci koje nadgleda.
- **Provajder konfiguracije** – Klijentske aplikacije mogu da koriste Redis Sentinel za otkrivanje servisa – klijenti se prvo povezuju na instancu Sentinel-a kako bi dobili IP adresu master instance. Ukoliko usled otkaza mastera nova instanca preuzme ovu ulogu, Sentinel će o ovome obavestiti klijente.

AUTOMATSKI FAILOVER - REDIS SENTINEL

Redis Sentinel predstavlja distribuirani sistem koji se sastoji iz više Sentinel procesa koji međusobno komuniciraju i sarađuju kako bi detektovali otkaze i obezbedili automatski failover.



REDIS SENTINEL – KVORUM

Kvorum predstavlja jedan od ključnih parametara Redis Sentinela – njime se definiše minimalan broj Sentinel instanci koje moraju da se slože da je došlo do otkaza master instance da bi se pokrenuo proces automatskog failover-a. *Kvorum se koristi samo za potrebe detekcije otkaza!* U slučaju da se Sentineli slože da je master instanca otkazala, jedan od Sentinela će pokušati da startuje proces failover-a. Proces će se uspešno startovati samo ukoliko je **većina Sentinel instanci** potvrdila ovaj proces.

REDIS SENTINEL – KVORUM PRIMERI

Primer 1 - quorum nije postignut:

```
3242:X 09 Jul 2024 18:37:28.663 * Sentinel new configuration saved on disk
3242:X 09 Jul 2024 18:37:28.664 * Sentinel ID is 0d6cf3b5595d3c0d7a3f7b5115f6d04748e711d3
3242:X 09 Jul 2024 18:37:28.664 # +monitor master myprimary 127.0.0.1 6379 quorum 2
3242:X 09 Jul 2024 18:37:28.665 * +slave slave 127.0.0.1:6380 127.0.0.1 6380 @ myprimary 127.0.0.1 6379
3242:X 09 Jul 2024 18:37:28.674 * Sentinel new configuration saved on disk
3242:X 09 Jul 2024 18:38:33.181 * +sentinel sentinel e513daec9c10a4060c93ec859c9f37ba873ed09e 127.0.0.1 5001 @ myprimary 127.0.0.1 6379
3242:X 09 Jul 2024 18:38:33.192 * Sentinel new configuration saved on disk
3242:X 09 Jul 2024 18:39:24.330 # +sdown sentinel e513daec9c10a4060c93ec859c9f37ba873ed09e 127.0.0.1 5001 @ myprimary 127.0.0.1 6379
3242:X 09 Jul 2024 18:41:15.318 # +sdown master myprimary 127.0.0.1 6379
3242:X 09 Jul 2024 18:41:39.796 # -sdown master myprimary 127.0.0.1 6379
```

Primer 2 - quorum postignut, nema failovera:

```
3791:X 09 Jul 2024 18:46:35.505 # +sdown sentinel 26b2122d901c0a983aad451ea06cd9e104dfbed 127.0.0.1 5001 @ myprimary 127.0.0.1 6379
3791:X 09 Jul 2024 18:46:53.129 # +sdown master myprimary 127.0.0.1 6379
3791:X 09 Jul 2024 18:46:53.129 # +odown master myprimary 127.0.0.1 6379 #quorum 1/1
3791:X 09 Jul 2024 18:46:53.129 # +new-epoch 1
3791:X 09 Jul 2024 18:46:53.129 # +try-failover master myprimary 127.0.0.1 6379
3791:X 09 Jul 2024 18:46:53.137 * Sentinel new configuration saved on disk
3791:X 09 Jul 2024 18:46:53.137 # +vote-for-leader 23ca5e21dd041404ea24e1704d4f9d85a8f0 1
3791:X 09 Jul 2024 18:47:00.068 # -failover-abort-not-elected master myprimary 127.0.0.1 6379
```


REDIS SENTINEL – KVORUM PRIMERI

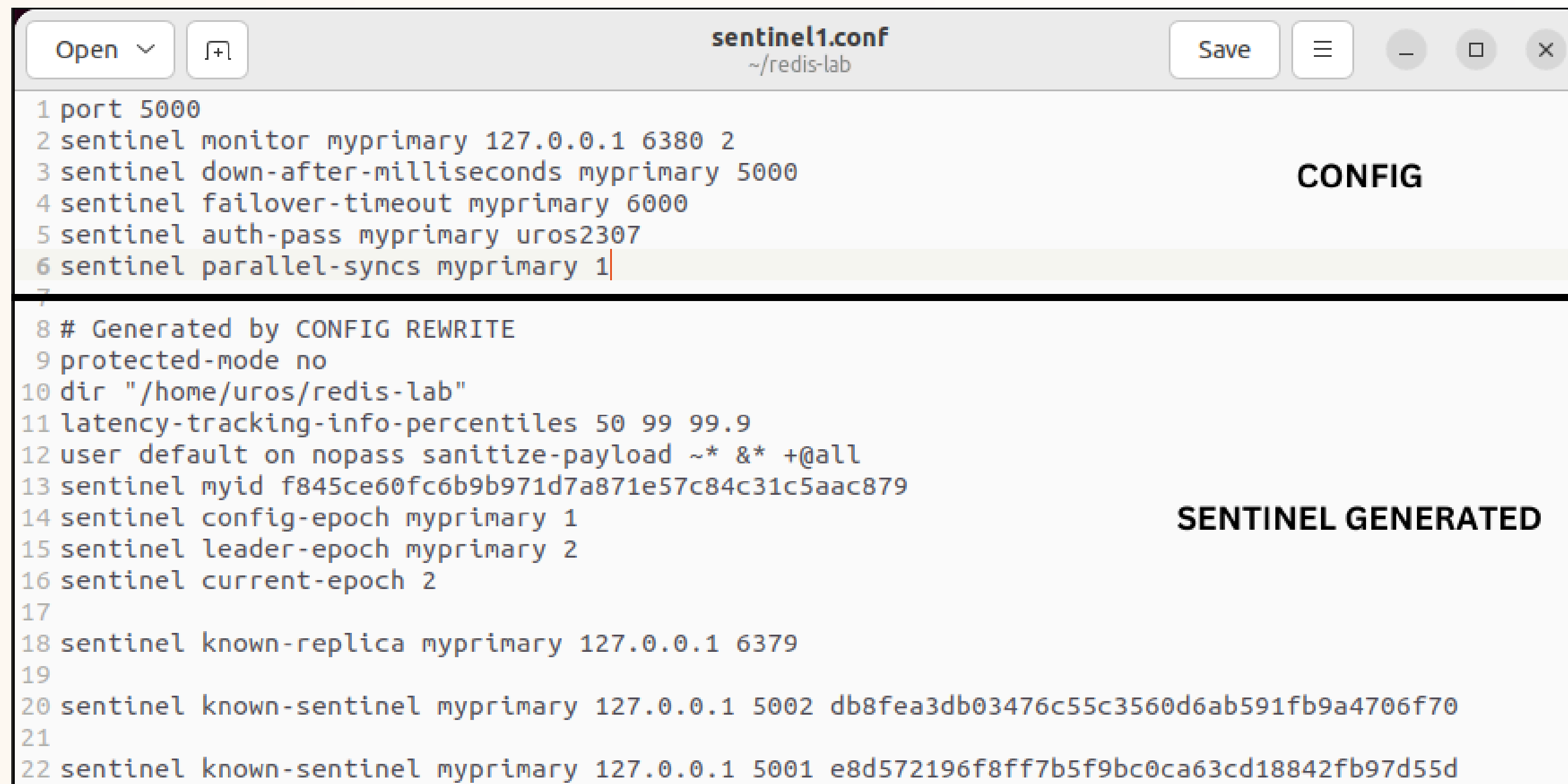
Primer 3 - uspešan failover:

```
4175:X 09 Jul 2024 18:54:36.987 # +sdown sentinel db8fea3db03476c55c3560d6ab591fb9a4706f70 127.0.0.1 5002 @ myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:34.839 # +sdown master myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:34.930 # +odown master myprimary 127.0.0.1 6379 #quorum 2/2
4175:X 09 Jul 2024 18:55:34.930 # +new-epoch 1
4175:X 09 Jul 2024 18:55:34.930 # +try-failover master myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:34.939 * Sentinel new configuration saved on disk
4175:X 09 Jul 2024 18:55:34.940 # +vote-for-leader f845ce60fc6b9b971d7a871e57c84c31c5aac879 1
4175:X 09 Jul 2024 18:55:34.949 * e8d572196f8ff7b5f9bc0ca63cd18842fb97d55d voted for f845ce60fc6b9b971d7a871e57c84c31c5aac879 1
4175:X 09 Jul 2024 18:55:34.993 # +elected-leader master myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:34.993 # +failover-state-select-slave master myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:35.070 # +selected-slave slave 127.0.0.1:6380 127.0.0.1 6380 @ myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:35.070 * +failover-state-send-slaveof-noone slave 127.0.0.1:6380 127.0.0.1 6380 @ myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:35.126 * +failover-state-wait-promotion slave 127.0.0.1:6380 127.0.0.1 6380 @ myprimary 127.0.0.1 6379
4175:X 09 Jul 2024 18:55:35.986 * Sentinel new configuration saved on disk
```

Navedeni primeri takođe naglašavaju i tipične probleme koji se mogu javiti ukoliko koristimo samo dve Sentinel instance. Otkazom bilo koje od njih, gubi se mogućnost automatskog failover-a, čime čitav sistem postaje nedostupan. Ovo je loša praksa koju treba izbegavati. Generalno, potrebne su **bar 3 instance Sentinela za robusnu arhitekturu**.

KONFIGURACIJA SENTINELA

Jedna Redis Sentinel instanca se pokreće komandom ***redis-server sentinel.conf -sentinel***, gde **sentinel.conf** predstavlja konfiguracioni fajl koji je neophodno navesti. U konfiguracionom fajlu se navode svi potrebni konfiguracioni parametri.



```
1 port 5000
2 sentinel monitor myprimary 127.0.0.1 6380 2
3 sentinel down-after-milliseconds myprimary 5000
4 sentinel failover-timeout myprimary 6000
5 sentinel auth-pass myprimary uros2307
6 sentinel parallel-syncs myprimary 1
7
8 # Generated by CONFIG REWRITE
9 protected-mode no
10 dir "/home/uros/redis-lab"
11 latency-tracking-info-percentiles 50 99 99.9
12 user default on nopass sanitize-payload ~* &* +@all
13 sentinel myid f845ce60fc6b9b971d7a871e57c84c31c5aac879
14 sentinel config-epoch myprimary 1
15 sentinel leader-epoch myprimary 2
16 sentinel current-epoch 2
17
18 sentinel known-replica myprimary 127.0.0.1 6379
19
20 sentinel known-sentinel myprimary 127.0.0.1 5002 db8fea3db03476c55c3560d6ab591fb9a4706f70
21
22 sentinel known-sentinel myprimary 127.0.0.1 5001 e8d572196f8ff7b5f9bc0ca63cd18842fb97d55d
```

KONFIGURACIJA SENTINELA

- **sentinel monitor <ime> <IP adresa mastera> <port mastera> <kvorum>** - Sentinel u se dodeljuje master koji treba da se nadgleda navođenjem njegove IP adrese i porta. Nije potrebno navoditi adrese replika, Sentinel na osnovu mastera može sam da otkrije adrese svih aktivnih replika.
- **sentinel down-after-milliseconds <ime> <broj_milisekundi>** - Definiše se minimalni vremenski interval tokom kojeg neka instanca mora biti neaktivna, da bi se proglasio njen otkaz
- **sentinel failover-timeout <ime> <broj_milisekundi>** - Vremenski interval nakon kojeg će failover ponovo moći da se izvrši, nakon što je već bio izvršen.
- **sentinel auth-pass <ime> <lozinka>** - Lozinka za pristup master instanci.
- **sentinel parallel-syncs <ime> <broj>** - Definiše broj replika koje će moći istovremeno da se sinhronizuju sa novim masterom nakon failover-a.

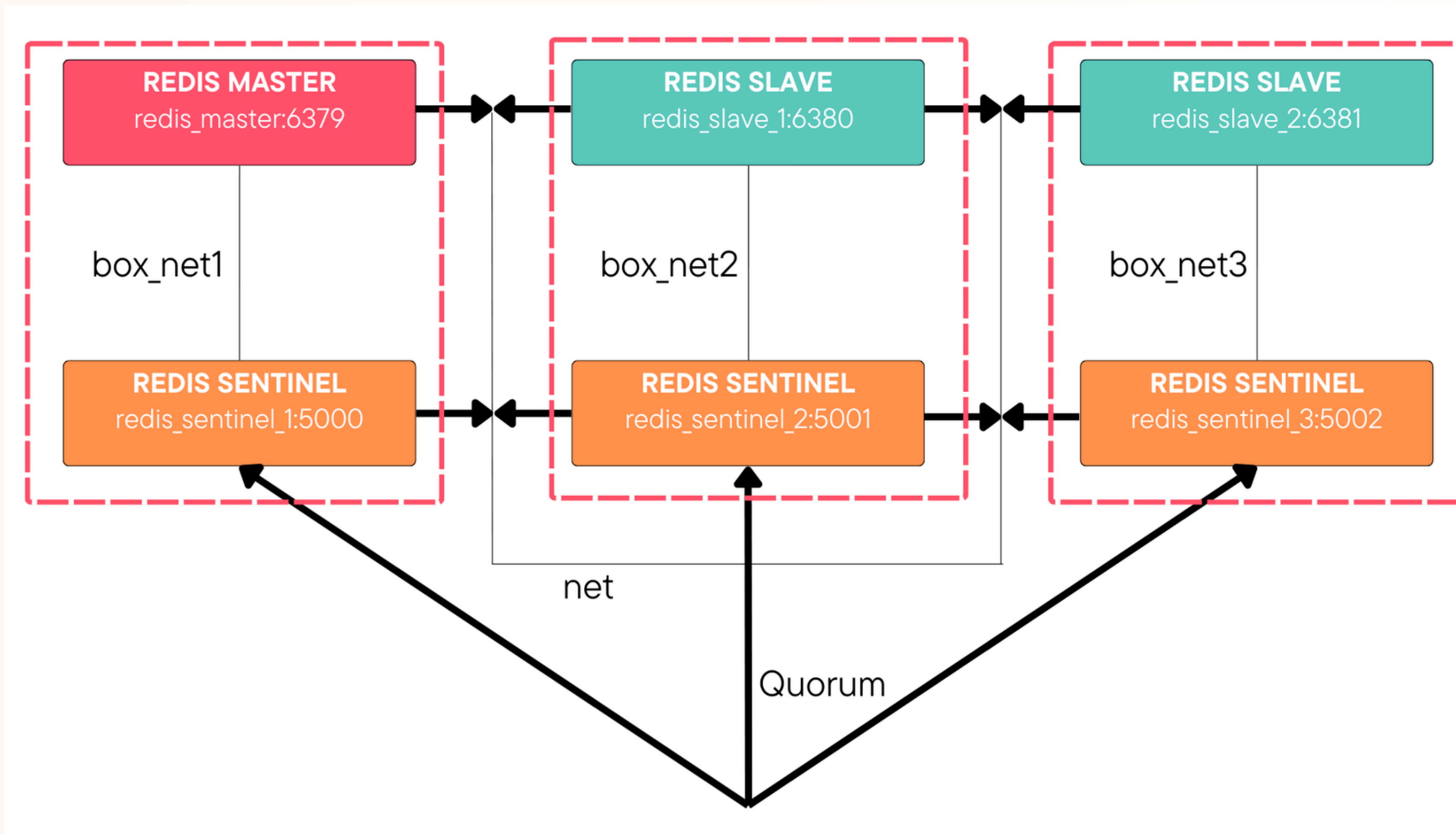
KONFIGURACIJA SENTINELA

Nakon što su Sentinel instance pokrenute, različitim komandama je moguće videti informacije o masteru, replikama, ili drugim Sentinelima, korišćenjem komande ***sentinel <komanda> <ime>***:

```
uros@uros-VirtualBox:~/redis-lab$ redis-cli -p 5000
127.0.0.1:5000> sentinel master myprimary
1) "name"
2) "myprimary"
3) "ip"
4) "127.0.0.1"
5) "port"
6) "6379"
7) "runid"
8) "10bc980372abcb3f11832e20d9cfbb76958e222a"
9) "flags"
10) "master"
11) "link-pending-commands"
12) "0"
13) "link-refcount"
14) "1"
15) "last-ping-sent"
16) "0"
17) "last-ok-ping-reply"
18) "896"
19) "last-ping-reply"
20) "896"
21) "down-after-milliseconds"
22) "5000"
23) "info-refresh"
24) "3563"
25) "role-reported"
26) "master"
27) "role-reported-time"
28) "23697"
29) "config-epoch"
30) "0"
31) "num-slaves"
32) "1"
33) "num-other-sentinels"
34) "2"

uros@uros-VirtualBox:~$ redis-cli -p 5000
127.0.0.1:5000> sentinel replicas myprimary
1) 1) "name"
2) "127.0.0.1:6380"
3) "ip"
4) "127.0.0.1"
5) "port"
6) "6380"
7) "runid"
8) "4ef9ca2a54f62b828d9ad255c0dadb81b934dd0d"
9) "flags"
10) "slave"
11) "link-pending-commands"
12) "0"
13) "link-refcount"
14) "1"
15) "last-ping-sent"
16) "0"
17) "last-ok-ping-reply"
18) "1040"
19) "last-ping-reply"
20) "1040"
21) "down-after-milliseconds"
22) "5000"
23) "info-refresh"
24) "117"
25) "role-reported"
26) "slave"
27) "role-reported-time"
28) "90540"
29) "master-link-down-time"
30) "0"
31) "master-link-status"
32) "ok"
33) "master-host"
34) "127.0.0.1"
35) "master-port"
36) "6379"
```

PRIMER ROBUSNE ARHITEKTURE



PRIMER ROBUSNE ARHITEKTURE - REPLIKACIJA

```
uros@uros-VirtualBox: ~/redis-lab
uros@uros-VirtualBox:~/redis-lab$ docker ps
CONTAINER ID   IMAGE                      COMMAND                  CREATED        STATUS        PORTS                                                                 NAMES
6dc6b68997f1   bitnami/redis-sentinel:latest "/opt/bitnami/script..." 12 seconds ago Up 12 seconds 0.0.0.0:5002->5002/tcp, :::5002->5002/tcp, 26379/tcp redis_sentinel_3
bd0cff9a1555   bitnami/redis-sentinel:latest "/opt/bitnami/script..." 15 seconds ago Up 14 seconds 0.0.0.0:5001->5001/tcp, :::5001->5001/tcp, 26379/tcp redis_sentinel_2
67afba231961   bitnami/redis-sentinel:latest "/opt/bitnami/script..." 17 seconds ago Up 17 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp, 26379/tcp redis_sentinel_1
bd4bb412c31a   redis                    "docker-entrypoint.s..." 20 seconds ago Up 19 seconds 6379/tcp, 0.0.0.0:6381->6381/tcp, :::6381->6381/tcp redis_slave_2
55d25a524fd2   redis                    "docker-entrypoint.s..." 22 seconds ago Up 22 seconds 6379/tcp, 0.0.0.0:6380->6380/tcp, :::6380->6380/tcp redis_slave_1
e6e6fdd5b085   redis                    "docker-entrypoint.s..." 25 seconds ago Up 24 seconds 0.0.0.0:6379->6379/tcp, :::6379->6379/tcp redis_master

uros@uros-VirtualBox:~/redis-lab$
```

```
uros@uros-VirtualBox:~$ redis-cli -p 6379
127.0.0.1:6379> GET 1
(nil)
127.0.0.1:6379> SET 1 "first"
OK
127.0.0.1:6379> GET 1
"first"
127.0.0.1:6379>
```

MASTER

```
uros@uros-VirtualBox:~$ redis-cli -p 6380
127.0.0.1:6380> GET 1
(nil)
127.0.0.1:6380> GET 1
"first"
127.0.0.1:6380> SET 2 "second"
(error) READONLY You can't write against a read only replica.
127.0.0.1:6380>
```

REPLICA 1

```
uros@uros-VirtualBox:~$ redis-cli -p 6381
127.0.0.1:6381> GET 1
(nil)
127.0.0.1:6381> GET 1
"first"
127.0.0.1:6381> SET 2 "second"
(error) READONLY You can't write against a read only replica.
127.0.0.1:6381>
```

REPLICA 2

```
18) "641"
19) "last-ping-reply"
20) "641"
21) "down-after-milliseconds"
22) "5000"
23) "info-refresh"
24) "3292"
25) "role-reported"
26) "master"
27) "role-reported-time"
28) "555550"
29) "config-epoch"
30) "0"
31) "num-slaves"
32) "2"
33) "num-other-sentinels"
34) "2"
35) "quorum"
36) "2"
37) "failover-timeout"
38) "6000"
39) "parallel-syncs"
40) "1"
127.0.0.1:5000>
```

SENTINEL 1

PRIMER ROBUSNE ARHITEKTURE - AUTOMATSKI FAILOVER

```
uros@uros-VirtualBox: ~/redis-lab
1:X 09 Jul 2024 22:02:04.804 # +config-update-from sentinel 9b1d0db534f59896a0ee0dfa2197422b85d17dcd 172.18.0.7 5002 @ mymaster 172.18.0.2 6379
1:X 09 Jul 2024 22:02:04.804 # +switch-master mymaster 172.18.0.2 6379 172.18.0.4 6381
1:X 09 Jul 2024 22:02:04.805 * +slave slave 172.18.0.3:6380 172.18.0.3 6380 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:02:04.807 * +slave slave 172.18.0.2:6379 172.18.0.2 6379 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:02:04.813 * Sentinel new configuration saved on disk
1:X 09 Jul 2024 22:02:09.812 # +sdown slave 172.18.0.2:6379 172.18.0.2 6379 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:02:28.183 # -sdown slave 172.18.0.2:6379 172.18.0.2 6379 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:02:38.151 * +convert-to-slave slave 172.18.0.2:6379 172.18.0.2 6379 @ mymaster 172.18.0.4 6381

```

```
uros@uros-VirtualBox: ~$ redis-cli -p 6379
127.0.0.1:6379> SET 2 "second"
(error) READONLY You can't write against a read only replica.
127.0.0.1:6379> GET 2
"SECOND"
127.0.0.1:6379>

REPLICA
```

```
uros@uros-VirtualBox: ~$ redis-cli -p 6380
127.0.0.1:6380> SET 2 "SECOND"
(error) READONLY You can't write against a read only replica.
127.0.0.1:6380> GET 2
"SECOND"
127.0.0.1:6380>

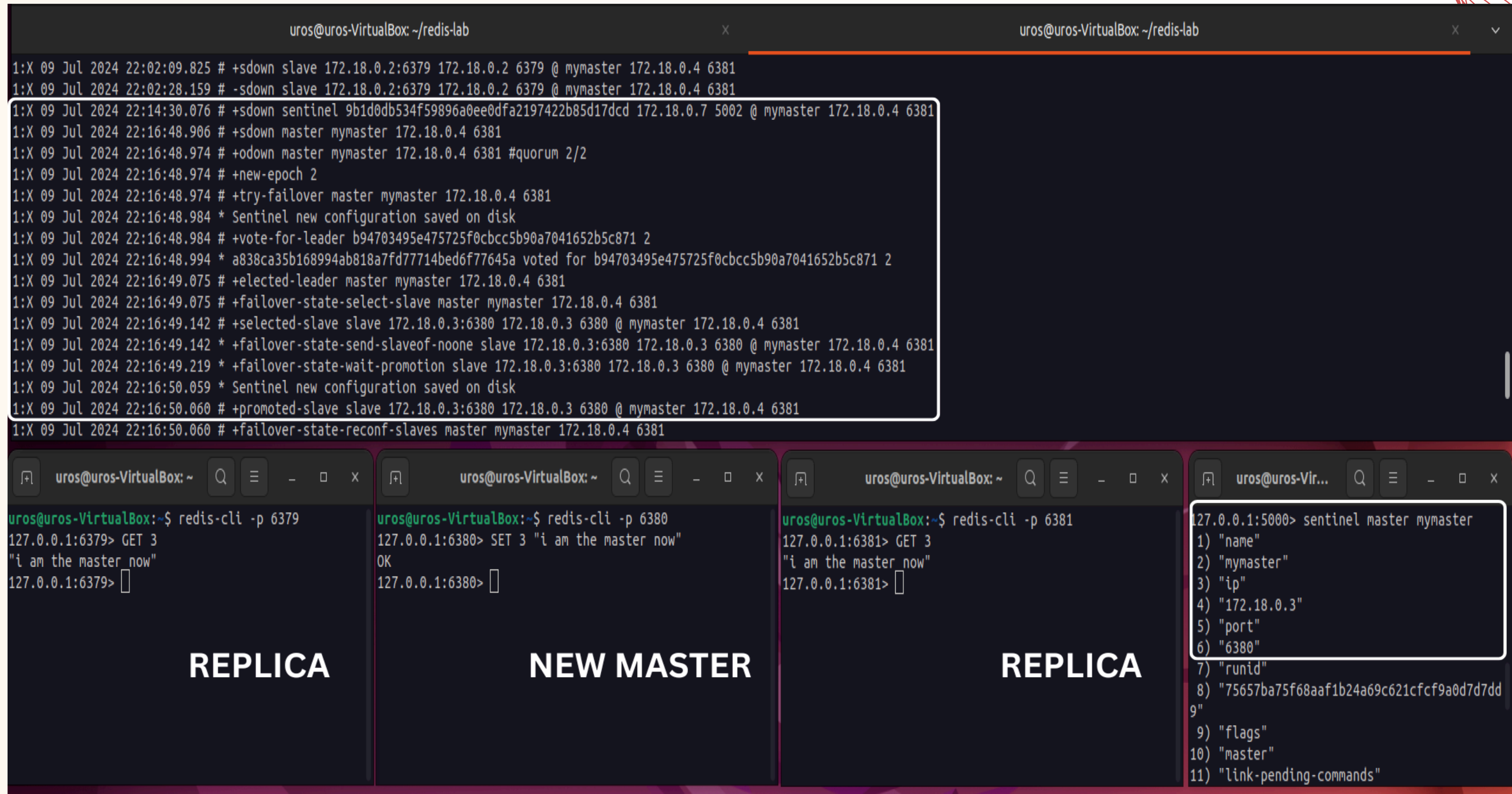
REPLICA
```

```
uros@uros-VirtualBox: ~$ redis-cli -p 6381
127.0.0.1:6381> SET 2 "SECOND"
OK
127.0.0.1:6381> GET 2
"SECOND"
127.0.0.1:6381>

NEW MASTER
```

```
uros@uros-VirtualBox: ~$ redis-cli -p 5000
127.0.0.1:5000> sentinel master mymaster
1) "name"
2) "mymaster"
3) "ip"
4) "172.18.0.4"
5) "port"
6) "6381"
7) "runid"
8) "2465f7a1eb33b25aa4e065fb2a13fb71d455571b"
9) "flags"
10) "master"
11) "link-pending-commands"
12) "0"
13) "link-refcount"
14) "1"
15) "last-ping-sent"
16) "0"
17) "last-ok-ping-reply"
18) "337"
19) "last-ping-reply"
20) "337"
```

PRIMER ROBUSNE ARHITEKTURE - OTKAZ MASTERA I SENTINELA



The image shows a terminal window titled "uros@uros-VirtualBox: ~/redis-lab" displaying the Redis Sentinel log. The log records a failover event where the master "mymaster" (172.18.0.4:6381) is down, and a new master is elected from the slave "127.0.0.1:6380". The log includes timestamps, log levels, and specific Sentinel commands like "+sdown", "+odown", "+new-epoch", "+try-failover", "+vote-for-leader", "+elected-leader", "+failover-state-select-slave", "+selected-slave", "+failover-state-send-slaveof-noone", "+failover-state-wait-promotion", "+promoted-slave", and "+failover-state-reconf-slaves".

```
uros@uros-VirtualBox: ~/redis-lab
1:X 09 Jul 2024 22:02:09.825 # +sdown slave 172.18.0.2:6379 172.18.0.2 6379 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:02:28.159 # -sdown slave 172.18.0.2:6379 172.18.0.2 6379 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:14:30.076 # +sdown sentinel 9b1d0db534f59896a0ee0dfa2197422b85d17dcd 172.18.0.7 5002 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:48.906 # +sdown master mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:48.974 # +odown master mymaster 172.18.0.4 6381 #quorum 2/2
1:X 09 Jul 2024 22:16:48.974 # +new-epoch 2
1:X 09 Jul 2024 22:16:48.974 # +try-failover master mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:48.984 * Sentinel new configuration saved on disk
1:X 09 Jul 2024 22:16:48.984 # +vote-for-leader b94703495e475725f0cbcc5b90a7041652b5c871 2
1:X 09 Jul 2024 22:16:48.994 * a838ca35b168994ab818a7fd77714bed6f77645a voted for b94703495e475725f0cbcc5b90a7041652b5c871 2
1:X 09 Jul 2024 22:16:49.075 # +elected-leader master mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:49.075 # +failover-state-select-slave master mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:49.142 # +selected-slave slave 172.18.0.3:6380 172.18.0.3 6380 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:49.142 * +failover-state-send-slaveof-noone slave 172.18.0.3:6380 172.18.0.3 6380 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:49.219 * +failover-state-wait-promotion slave 172.18.0.3:6380 172.18.0.3 6380 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:50.059 * Sentinel new configuration saved on disk
1:X 09 Jul 2024 22:16:50.060 # +promoted-slave slave 172.18.0.3:6380 172.18.0.3 6380 @ mymaster 172.18.0.4 6381
1:X 09 Jul 2024 22:16:50.060 # +failover-state-reconf-slaves master mymaster 172.18.0.4 6381
```

Below the log, three terminal windows show the Redis CLI interaction:

- REPLICA** (127.0.0.1:6379):

```
uros@uros-VirtualBox: ~$ redis-cli -p 6379
127.0.0.1:6379> GET 3
"i am the master now"
127.0.0.1:6379>
```
- NEW MASTER** (127.0.0.1:6380):

```
uros@uros-VirtualBox: ~$ redis-cli -p 6380
127.0.0.1:6380> SET 3 "i am the master now"
OK
127.0.0.1:6380>
```
- REPLICA** (127.0.0.1:6381):

```
uros@uros-VirtualBox: ~$ redis-cli -p 6381
127.0.0.1:6381> GET 3
"i am the master now"
127.0.0.1:6381>
```

On the far right, a terminal window shows the output of the `sentinel master mymaster` command:

```
127.0.0.1:5000> sentinel master mymaster
1) "name"
2) "mymaster"
3) "ip"
4) "172.18.0.3"
5) "port"
6) "6380"
7) "runid"
8) "75657ba75f68aaf1b24a69c621cfcf9a0d7d7dd9"
9) "flags"
10) "master"
11) "link-pending-commands"
```


| SENTINEL – OBAVEŠTENJA

Jedna od mnogobrojnih mogućnosti Redis Sentinela jeste i obaveštavanje administratora sistema, ili drugih podstistema o događajima koji se dešavaju u klasteru. U te svrhe Sentinel koristi Redis PUB/SUB mehanizam i objavljuje (eng. Publish) sve događaje koji se dešavaju u sistemu na odgovarajućim kanalima. Korišćenjem Redis klijenta, iz drugih aplikacija je moguća pretplata (eng. Subscribe) na ove događaje, ukoliko je potrebno da se na njih reaguje. Neki od događaja koje Redis Sentinel prati su:

- **+slave** – nova replika je dodata u sistem
- **+sentinel** – novi Sentinel je dodat u sistem
- **+failover-end** – proces failover-a je završen uspešnost
- **+sdown** – neka instanca (Redis, Sentinel) je otkazala
- **-sdown** – instanca je ponovo dostupna
- **+try-failover** – pokušaj failover-a je iniciran od strane Sentinela
- **+elected-leader** – nova master replika je izglasana

SENTINEL - OBAVEŠTENJA PRIMER

```
uros@uros-VirtualBox:... x uros@uros-VirtualBox:... x uros@uros-VirtualBox:...  
127.0.0.1:5000> SUBSCRIBE +sdown  
1) "subscribe"  
2) "+sdown"  
3) (integer) 1  
1) "message"  
2) "+sdown"  
3) "master myprimary 127.0.0.1 6379"  
1) "message"  
2) "+sdown"  
3) "slave 127.0.0.1:6379 127.0.0.1 6379 @ myprimary 127.0.0.1 6380"  
Reading messages... (press Ctrl-C to quit or any key to type command)
```

| SENTINEL - TILT MOD

Redis Sentinel zavisi od sistemskog časovnika. Da bi utvrdio koje instance nisu više dostupne, Sentinel upoređuje vremena poslednjeg odgovora instanci na PING komandu i upoređuje je sa trenutnim vremenom. Ukoliko se vreme iznenada promeni iz bilo kog razloga, Sentinel može da počne čudno da se ponaša.

Zbog toga Sentinel sadrži i specijalni TILT mod, u koji prelazi ukoliko dođe do opisane situacije. U ovom modu, Sentinel nastavlja da nadgleda sistem, međutim prestaje da preduzima bilo kakve akcije (pokretanje failover-a, glasanje, detektovanje otkaza) i na komandu `SENTINEL is-master-down-by-addr` uvek odgovara odrično – jer se Sentinelu u TILT modu ne može verovati da pouzdano detektuje otkaze. Ukoliko nakon 30 sekundi vreme prestane drastično da se menja, Sentinel se vraća na normalni mod rada.

SENTINEL - TILT MOD PRIMER

```
# Sentinel
sentinel_masters:1
sentinel_tilt:0
sentinel_tilt_since_seconds:-1
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=myprimary,status=ok,address=127.0.0.1:6380,slaves=1,sentinels=2
127.0.0.1:5000>
```

Normalan rad

```
35887:X 10 Jul 2024 01:20:56.166 * Sentinel ID is f32f5e8476a48829b5e7cf1de7c12902dd9e0c95
35887:X 10 Jul 2024 01:20:56.166 # +monitor master myprimary 127.0.0.1 6380 quorum 2
35887:X 10 Jul 2024 01:20:30.056 # +tilt #tilt mode entered
35887:X 10 Jul 2024 01:18:30.014 # +tilt #tilt mode entered
```

Logovi sentinela - Promena
časovnika

```
# Sentinel
sentinel_masters:1
sentinel_tilt:1
sentinel_tilt_since_seconds:14
sentinel_running_scripts:0
sentinel_scripts_queue_length:0
sentinel_simulate_failure_flags:0
master0:name=myprimary,status=ok,address=127.0.0.1:6380,slaves=1,sentinels=2
127.0.0.1:5000>
```

TILT

ZAKLJUČAK

Visoka dostupnost je ključni aspekt savremenih baza podataka, posebno u okruženjima gde je kontinuirani pristup podacima od suštinskog značaja. Redis, kao visoko performantna baza podataka, nudi robusna rešenja za postizanje visoke dostupnosti koristeći mehanizme replikacije i automatskog failover-a korišćenjem Redis Sentinela. Pored gotovih rešenja, potrebno je primenjivati i tehnike kao što su inženjering haosa i redovno testiranje sistema u produkciji, koje pomažu u identifikaciji slabosti i potencijalnih problema pre nego što se oni ozbiljnije manifestuju.

HVALA NA PAŽNJI!