Univerzitet u Nišu,

Elektronski fakultet Niš

Logo

Description automatically generated

*Seminarski rad*

Sistemi za upravljanje bazama podataka

**Replikacija podataka kod MongoDB baze podataka**

Mentor: Student:

Aleksandar Stanimirović Ivan Milojković 1687

Niš, sepembar 2024. godine

Sadržaj

[Uvod 3](#_Toc138800029)

[Replikacija 4](#_Toc138800030)

[Skup replika 6](#_Toc138800031)

[Primarni čvor 7](#_Toc138800032)

[Sekundarni čvorovi 10](#_Toc138800033)

[Arbitražni čvor 16](#_Toc138800034)

[Asinhrone replikacije 17](#_Toc138800035)

[Rollback tokom failovera 19](#_Toc138800036)

[Sinhronizacija 19](#_Toc138800037)

[Read preference 21](#_Toc138800038)

[Praktični primer 23](#_Toc138800039)

[Zaključak 29](#_Toc138800047)  
Literatura ………………………………………………………………………………………………………………………………………………30

# **Uvod**

Svi smo svedoci da su pouzdanost i dostupnost podataka od suštinskog značaja za poslovni uspeh, organizacija svih veličina, od velikih korporacija do malih preduzeća, i sve te orgnaizacije se suočavaju sa ogromnim posledicama u slučaju gubitka ili nedostupnosti podataka. Upravo zbog toga, baze podataka razvijaju raznovrsne mehanizme replikacije, a među njima je i MongoDB sa svojim skalabilnim i pouzdanim replikacionim sistemima. Korišćenjem replikacije, podaci se mogu kopirati i distribuirati na više čvorova unutar sistema, čime se postiže visoka dostupnost i otpornost na kvarove.

Replikacija u MongoDB-u nudi ključne prednosti, kao što su visoka dostupnost podataka i mogućnost horizontalne skalabilnosti. U slučaju pada jednog čvora, sistem može automatski preći na drugi, sekundarni čvor, osiguravajući kontinuitet rada i minimizirajući potencijalne gubitke podataka. Ovaj princip posebno je značajan za kritične poslovne aplikacije koje zahtevaju 24/7 dostupnost, kao što su e-trgovine, društvene mreže i sistemi za upravljanje poslovanjem.

Uz tehnike replikacije koje MongoDB koristi, organizacije mogu postati otpornije na tehničke kvarove, dok istovremeno obezbeđuju visoke performanse i kontinualnu dostupnost podataka. Kroz ovaj rad biće analizirane prednosti implementacije replikacije u MongoDB-u, sa posebnim osvrtom na načine na koje ova tehnika doprinosi poboljšanju pouzdanosti i efikasnosti u upravljanju velikim količinama podataka.

# **Replikacija**

Replikacija baze podataka je proces kreiranja i održavanja više kopija iste baze podataka na različitim lokacijama ili čvorovima, kako bi se osigurala dostupnost, pouzdanost i otpornost podataka. Time se omogućava pristup korisnicima sa različitih geografskih lokacija da bez kašnjenja, a u slučaju kvara ili prekida rada jednog čvora, druge kopije podataka preuzimaju ulogu, čime se smanjuje rizik od gubitka podataka i downtime-a. Replikacija je ključna za poslovni kontinuitet jer pruža rezervne kopije koje omogućavaju nesmetan rad sistema.  
  
 Replikacija baze podataka takođe poboljšava skalabilnost sistema, jer omogućava raspodelu opterećenja između različitih replika. Na primer, različiti serveri mogu rukovati upitima za čitanje podataka, smanjujući pritisak na primarni server i poboljšavajući ukupne performanse sistema. Osim toga, replikacija olakšava geografski raspoređene operacije, jer korisnici mogu pristupati najbližoj replici, čime se smanjuje latencija i poboljšava korisničko iskustvo. U slučaju katastrofalnih događaja, poput gubitka podataka ili prekida mreže, replikacija osigurava kontinuitet poslovanja i minimalizuje posledice na operacije.

Ova tehnika je posebno značajna u organizacijama sa velikim obimom podataka i potrebom za visokom dostupnošću, gde su brzina pristupa i pouzdanost ključni faktori. Replikacija ne samo da osigurava integritet podataka, već omogućava i napredne strategije poput disaster recovery planova i failover sistema, čime se osigurava robustan i stabilan IT ekosistem.

Replikacija podataka može se klasifikovati u različite tipove na osnovu metode, svrhe i karakteristika procesa replikacije. Tri glavna tipa replikacije podataka su transakcijska replikacija (eng. Transaction replication), snapshot replikacija (eng. Snapshot replication) i replikacija spajanja (eng. Merge replication).

Transakcijska replikacija uključuje kopiranje baza podataka u celosti sa primarnog servera i slanje na sekundarne servere. Sve promene u podacima se dosledno i kontinuirano ažuriraju. Budući da se podaci replikovani u realnom vremenu šalju sa primarne baze podataka na sekundarne servere u redosledu njihovog nastanka, obezbeđuje se doslednost transakcija. Ovaj tip replikacije baza podataka se često koristi u okruženjima server-na-server (eng. Server-to-server).

Kod snapshot replikacija, snimak baze podataka se distribuira sa primarnog servera na sekundarne servere. Umesto kontinuiranih ažuriranja, podaci se šalju onakvi kakvi su bili u trenutku snimanja. Ovaj tip replikacije baze podataka je preporučljiv kada nema mnogo promena u podacima ili kada se inicijalno usklađivanje između primarnih servera i sekundarnih servera tek započinje. Iako nije koristan za rezervne kopije podataka jer ne prati promene u podacima, replikacija snimaka može pomoći u oporavku u slučaju slučajnog brisanja.

Replikacija spajanja sastoji se u kombinovanju dve baze podataka u jednu. Kao rezultat, sve promene u podacima mogu se ažurirati sa publisher-a na subscriber-e. Ovaj tip replikacije je složen jer obe strane (primarni server i sekundarni serveri) mogu vršiti promene u podacima. Ovaj tip replikacije se preporučuje samo za upotrebu u okruženju server-na-klijent.

Šeme replikacije su operacije i zadaci potrebni za izvođenje replikacije podataka. Tri glavne šeme replikacije podataka su potpuna replikacija, delimična replikacija i bez replikacije (eng. No replication).

Sa potpunom replikacijom, primarna baza podataka se u potpunosti kopira na svako mesto u distribuiranom sistemu. Ovaj globalni distribucioni sistem pruža visoku redundantnost baze podataka, smanjenu latenciju i ubrzano izvršavanje upita. Nedostaci potpune replikacije su to što je teško postići konkurentnost i što su procesi ažuriranja spori.

U šemi delimične replikacije, neki delovi baze podataka su replikovani na nekim ili svim mestima, i to obično podaci koji su nedavno ažurirani. Delimična replikacija omogućava davanje prioriteta podacima koji su važni i treba ih replikovati.

Bez replikacije (eng . No replication) je šema u kojoj se svi podaci čuvaju samo na jednom mestu. Ovo omogućava lako obnavljanje podataka i postizanje konkurentnosti. Nedostaci su što negativno utiče na dostupnost i usporava izvršavanje upita.

Tehnike replikacije podataka odnose se na metode i mehanizme koji se koriste za repliciranje podataka sa primarnog izvora na jedan ili više ciljnih sistema ili lokacija. Najčešće korišćene tehnike replikacije podataka su replikacija celih tabela, replikacija zasnovana na ključevima i replikacija zasnovana na logovima.

Sa potpunom replikacijom tabela, svi podaci se kopiraju sa izvora podataka na odredište, uključujući sve nove i postojeće podatke. Ova tehnika se preporučuje ako se zapisi redovno brišu ili ako druge tehnike nisu tehnički moguće. Zbog veličine skupova podataka, potpuna replikacija tabela zahteva više procesnih i mrežnih resursa, kao i veće troškove.

Kod inkrementalnih replikacija zasnovanih na ključevima, replikovani su samo novi podaci koji su dodati od prethodnog ažuriranja. Ova tehnika je efikasnija jer se kopira manji broj redova. Jedan nedostatak inkrementalnih replikacija zasnovanih na ključevima je što ne omogućava replikaciju podataka iz prethodnog ažuriranja koji su trajno obrisani.

Replikacija zasnovana na logovima beleži promene napravljene na podacima na izvoru podataka praćenjem logova baze podataka (log fajl ili ChangeLog). Ove promene se zatim replikuju na ciljne sisteme i primenjuju se samo na podržane izvore baza podataka. Replikacija zasnovana na logovima se preporučuje kada je izvorna struktura baze podataka statična, jer bi u suprotnom mogla postati veoma zahtevna u pogledu resursa.

## 

## **Skup replika**

Skup replika u MongoDB-u je grupa mongod procesa koji održavaju isti skup podataka. Replica setovi obezbeđuju redundansu i visoku dostupnost, i predstavljaju osnovu za sve produkcione implementacije.

Kao što je rečeno, replikacija obezbeđuje redundantnost i povećava dostupnost podataka. Sa više kopija podataka na različitim serverskim bazama podataka, replikacija pruža određeni nivo tolerancije na greške u slučaju gubitka jednog serverskog sistema baze podataka.

U nekim slučajevima, replikacija može omogućiti povećan kapacitet za čitanje, jer klijenti mogu slati operacije čitanja na različite servere. Održavanje kopija podataka u različitim data centrima može povećati lokalnost podataka i dostupnost za distribuirane aplikacije. Takođe, može održavati dodatne kopije za posebne namene, kao što su oporavak od katastrofa ili backup.

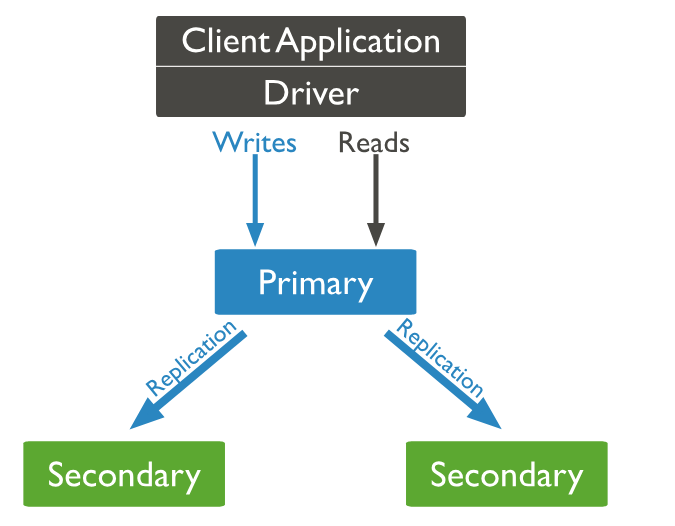


Slika 1. Dodavanje člana skupu replika

### **Primarni čvor**

Primarni čvor u MongoDB skupu replika je jedini član koji prima pisane operacije. Sve promene koje su izvršene na primarnom čvoru beleže se u Oplog (operativni log), a zatim se kopiraju na sekundarne čvorove u asinhronom procesu. Skupovi replika obezbeđuju visoku dostupnost tako što omogućavaju preuzimanje uloge primarnog čvora od strane sekundarnih čvorova u slučaju pada trenutnog primarnog.

U sledećem tročlanom replika setu, primarni čvor prihvata sve operacije pisanja. Zatim sekundarni čvorovi repliciraju Oplog kako bi primenili operacije na svoje skupove podataka.



Slika 2. – Primarni i sekundarni čvorovi

### 

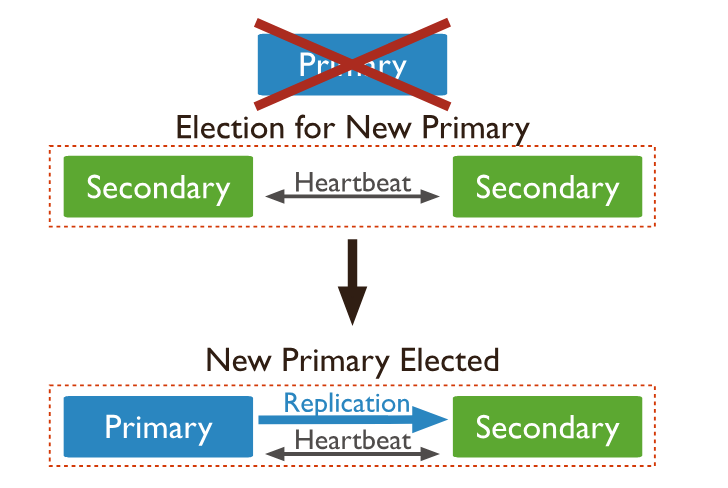
### Svi članovi replike seta mogu prihvatiti operacije čitanja. Međutim, po podrazumevanim postavkama, aplikacija usmerava operacije čitanja ka primarnom čvoru.

### Skupovi replika mogu imati najviše jednog primarnog člana. Ako trenutni primarni čvor postane nedostupan, izbori određuju novog primarnog člana.

### Skupovi replika koriste izbore kako bi odredili koji član tog skupa će postati primarni. Skupovi replika mogu pokrenuti izbore kao odgovor na različite događaje, kao što su:

* Dodavanje novog čvora u skupu replika,
* Inicijalizacija skupa replike,
* Izvođenje održavanja skupa replika korišćenjem metoda kao što su rs.stepDown() ili rs.reconfig(), i
* Sekundarni članovi gube konekciju sa primarnim na duže od podešenog timeout-a (podrazumevano 10 sekundi).

U sledećem dijagramu, primarni čvor je bio nedostupan duže od podešenog timeout-a, što je pokrenulo automatski failover proces. Jedan od preostalih sekundarnih čvorova poziva izbore za odabir novog primarnog i automatski nastavlja normalne operacije.



Slika 3.Biranje novog primarnog čvora

Replika set ne može obrađivati operacije pisanja dok izbor novog primarnog čvora ne bude uspešno završen. Replika set može nastaviti da prihvata upite za čitanje ako su upiti konfigurisani da se izvode na sekundarnim čvorovima.

Prosečno vreme potrebno za izbor novog primarnog čvora u klasteru obično ne bi trebalo da prelazi 12 sekundi, pod pretpostavkom da su korišćene podrazumevane konfiguracije replika seta. Ovo uključuje vreme potrebno za označavanje primarnog čvora kao nedostupnog i za pozivanje i završavanje izbora. Ovo vreme možete prilagoditi menjajući opciju `settings.electionTimeoutMillis` u konfiguraciji replikacije. Faktori poput mrežne latencije mogu produžiti vreme potrebno za završavanje izbora u replika setu, što može uticati na vreme tokom kojeg klaster može raditi bez primarnog čvora. Ovi faktori zavise od specifične arhitekture klastera.  
  
Faktori koji utiču na izbore su:

1. Protokol izbora replikacije

Protocol version:1 replikacije: smanjuje vreme failovera u replika setu i ubrzava otkrivanje više istovremenih primarnih čvorova.

1. Heartbeats

Članovi replika seta šalju međusobne pingove svake dve sekunde. Ako odgovor na heartbeat ne stigne u roku od 10 sekundi, ostali članovi označavaju člana koji nije odgovorio kao nedostupnim.

1. Prioritet Članova

Nakon što replika set dobije stabilnog primarnog člana, algoritam izbora će nastojati da sekundarni član sa najvišim prioritetom inicira izbore. Prioritet člana utiče i na vreme i na ishod izbora: sekundarni članovi sa višim prioritetom pokreću izbore relativno ranije nego članovi sa nižim prioritetom i imaju veće šanse da pobede. Međutim, član sa nižim prioritetom može biti izabran za primarnog na kratak period, čak i ako je član sa višim prioritetom dostupan. Članovi replika seta nastavljaju da pokreću izbore dok član sa najvišim prioritetom ne postane primarni.

1. Mirrored reads

MongoDB pruža mirrored reads kako bi unapred pripremio keš na izborne sekundarne članove sa najnovije pristupanim podacima. Kroz mirrored reads, primarni član može da preslika deo operacija koje primi i pošalje ih na odabrani skup izbornih sekundarnih članova. Ova priprema keša na sekundarnim članovima može pomoći u bržem vraćanju performansi nakon izbora.

5) Gubitak Data Centra

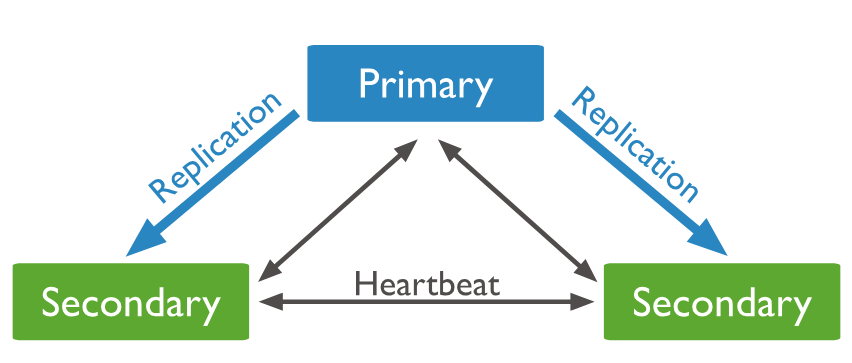
Kod distribuiranog replika seta, gubitak jednog data centra može uticati na sposobnost preostalih članova u drugim data centrima da izaberu novog primarnog člana.

Ako je moguće, raspored članova replica seta bi trebala biti preko više data centara kako bi se povećala verovatnoću da, čak i u slučaju gubitka jednog data centra, neki od preostalih članova replica seta može postati novi primarni član.

### **Sekundarni čvor**

Sekundarni član održava kopije podataka sa primarnog člana. Da bi replikovao podatke, sekundarni član primenjuje operacije iz oplog-a primarnog člana na sopstveni skup podataka u asinhronom procesu. Replika set može imati jedan ili više sekundarnih članova.

U prikazanom tročlanom replica setu, postoje dva sekundarna člana. Sekundarni članovi repliciraju oplog primarnog člana i primenjuju operacije na svoje skupove podataka.



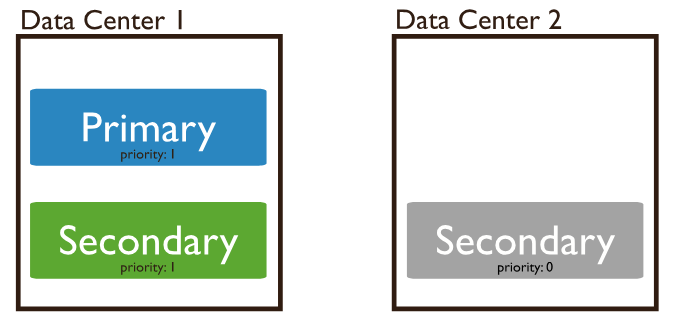
Slika 4. – Skup replika sa procesom otkucaja srca

Iako klijenti ne mogu da upisuju podatke na sekundarne članove, klijenti mogu da čitaju podatke sa sekundarnih članova.  
Sekundarni član može postati primarni. Ako trenutni primarni postane nedostupan, replikacioni skup organizuje izbore da bi odabrao koji od sekundarnih članova postaje novi primarni. O tome je bilo reči u prethodnim poglavljima.  
  
 Moguće je konfigurirati sekundarni član za specifičnu svrhu, odnosno da:  
 1) Se spreči da postane primarni tokom izbora, što omogućava da se nalazi u sekundarnom data centru ili da služi kao hladna rezervna kopija.  
Član sa prioritetom 0 je član koji ne može postati primarni i ne može pokrenuti izbore. Članovi sa prioritetom 0 mogu potvrditi pisanje koje je izdano sa write concern vrednošću w : <broj>. Za "majority" write concern, član sa prioritetom 0 takođe mora biti član sa pravom glasa (tj. members[n].votes je veće od 0) da bi potvrdio pisanje. Članovi replikacionog skupa koji nemaju pravo glasa (tj. members[n].votes je 0) ne mogu doprineti potvrđivanju operacija pisanja sa "majority" write concern.

Osim pomenutih ograničenja, sekundarni članovi sa prioritetom 0 funkcionišu kao normalni sekundarni: održavaju kopiju skupa podataka, prihvataju operacije čitanja i glasaju na izborima.

Konfigurisanje člana replikacionog skupa sa prioritetom 0 može biti poželjno ako je taj član raspoređen u data centru koji je udaljen od glavnog rasporeda i stoga ima veću latenciju. Može dobro služiti lokalnim zahtevima za čitanje, ali možda nije idealan kandidat za obavljanje dužnosti primarnog zbog svoje latencije.

U ovom slučaju, sledeći dijagram prikazuje data centar sa leve strane koji je domaćin primarnom i sekundarnom član, i data centar sa desne strane koji je domaćin sekundarnom članu koji je konfigurisan da ima prioritet 0 kako bi se sprečilo da postane primarni. Zbog ove postavke, samo članovi u data centru sa leve strane su podobni da postanu primarni u izborima.



Slika 5. Data centri sa čvorovima i prioritetima

Write concern je opcija u MongoDB-u koja određuje nivo potvrde i sigurnosti prilikom izvršavanja operacija pisanja. To je mehanizam koji omogućava kontrolu nad time koliko će instanci u replikacionom skupu potvrditi da je operacija pisanja uspešno obavljena pre nego što se vraća odgovor aplikaciji. Write concern pomaže u balansiranju između brzine izvršavanja i pouzdanosti podataka.

Vrste write concern opcija uključuju:

1) w: 0 - Ova opcija označava da se operacija pisanja izvršava bez čekanja na potvrdu od servera. Nema garancije da će operacija pisanja biti sačuvana u bazi podataka.

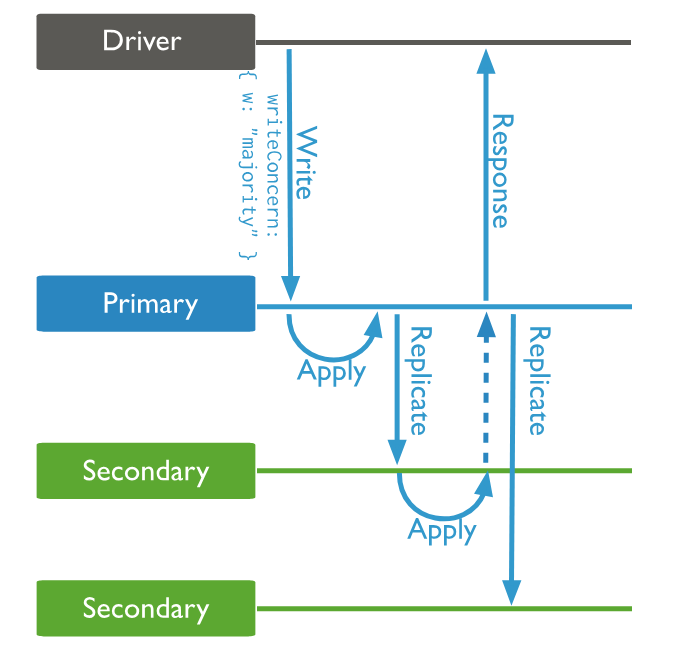
2) w: 1 - Ova opcija znači da će pisanje biti potvrđeno kada ga primarni čvor potvrdi. Ovo je osnovna opcija koja obezbeđuje da su podaci zapisani na primarni čvor.

3) w: <broj> - Ova opcija omogućava specifikovanje broja članova replikacionog skupa koji moraju potvrditi operaciju pisanja pre nego što se operacija smatra uspešnom. Na primer, w: 2 znači da barem dva čvora (uključujući primarni) moraju potvrditi pisanje.

4) w: "majority"- Ova opcija obezbeđuje da operacija pisanja bude potvrđena od strane većine članova replikacionog skupa. Ovo je najviši nivo sigurnosti jer osigurava da su podaci pisani u većini članova replikacionog skupa, čime se smanjuje rizik od gubitka podataka u slučaju kvara.

5) j: true - Ova opcija obezbeđuje da pisanje bude potvrđeno samo nakon što je zapisano u "journal" (dnevnik) na disku. Ovo povećava sigurnost podataka u slučaju kvara sistema.

6) wtimeout: <vreme> - Ova opcija postavlja vremensko ograničenje za čekanje potvrde od članova replikacionog skupa. Ako se potvrda ne primi u navedenom vremenskom periodu, operacija pisanja će biti prekinuta.

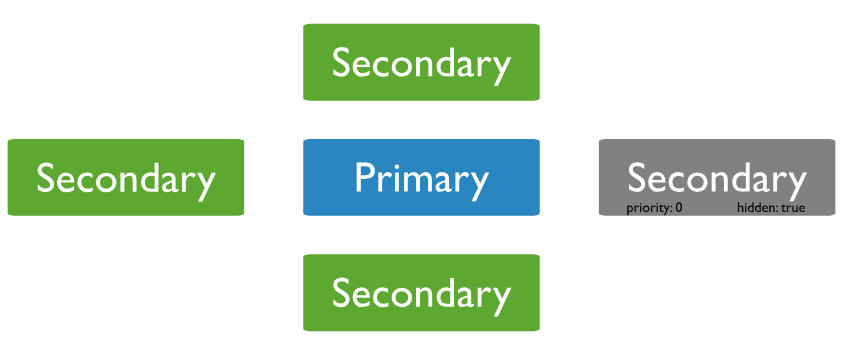
Kombinovanjem ovih opcija, write concern se može koristiti kako bi se došlo do optimizacije brzine pisanja i pouzdanosti podataka u skladu sa zahtevima aplikacije.  
  
 Slika 6. Write concern

Aplikacija koja izvršava operaciju pisanja koja zahteva potvrdu write concern-a čeka dok primarni čvor ne primi potvrdu od potrebnog broja članova za specificirani write concern. Kada je write concern postavljen na `w` veće od 1 ili `w: "majority"`, primarni čvor čeka dok svi potrebni sekundarni čvorovi ne potvrde pisanje pre nego što vrati potvrdu o write concern-u. Za write concern `w: 1`, primarni čvor može odmah da vrati potvrdu o write concern-u čim lokalno primeni pisanje, budući da je on sam dovoljno za zadovoljenje zahteva za write concern.

Što više članova potvrdi pisanje, manja je verovatnoća da će napisani podaci biti vraćeni u slučaju kvara primarnog čvora. Međutim, postavljanje visokog write concern-a može povećati latenciju, jer klijent mora da čeka dok ne dobije traženi nivo potvrde write concern-a.

Idealna vrednost za write concern zavisi od ciljeva i performansi aplikacija i zahteva. Aplikacija čeka dok primarni čvor ne vrati potvrdu o write concern-u, što znači da je izračunata većina članova sa pravom glasa koji sadrže podatke potvrdila operaciju pisanja. Na primer, u replikacionom skupu sa 3 člana (P-S-S), operacija bi zahtevala potvrdu od 2 od 3 člana. Ako se replikacioni skup kasnije proširi dodavanjem dva dodatna članova sa pravom glasa, ista operacija bi zahtevala potvrdu od 3 od 5 članova skupa. Ako primarni čvor ne vrati potvrdu o write concern-u u okviru zadanog `wtimeout` limita, operacija pisanja će neuspešno završiti sa greškom u vezi sa write concern-om.

Ako operacija pisanja istekne čekajući na zadati write concern, to samo ukazuje na to da traženi broj članova replikacionog skupa nije potvrdio operaciju pisanja unutar `wtimeout` vremenskog perioda. To ne mora nužno značiti da primarni čvor nije primenio pisanje. Podaci mogu biti prisutni na delu članova replikacionog skupa u trenutku greške u vezi sa write concern-om i mogu nastaviti da se repliciraju dok svi čvorovi u klasteru ne dobiju te podatke. Aplikacije treba da uzmu u obzir potencijalnu dostupnost pisanih podataka bez obzira na stanje potvrde o write concern-u.

2) Se spreče aplikacije da čitaju iz njega, što omogućava da se koristi za aplikacije koje zahtevaju odvajanje od normalnog saobraćaja.  
 Skriveni član održava kopiju skupa podataka primarnog čvora, ali je nevidljiv za klijentske aplikacije. Skriveni članovi su dobri za radne opterećenja sa različitim obrascima korišćenja u poređenju sa drugim članovima u replikacionom skupu. Skriveni članovi moraju uvek biti članovi sa prioritetom 0 i stoga ne mogu postati primarni čvor. Oni mogu da učestvuju u izborima.  


Slika 7. Skriveni član

3) Održava stalni "istorijski" snapshot za upotrebu u oporavku od određenih grešaka, kao što su nenamerno obrisane baze podataka.

Članovi sa zakašnjenjem (delayed members) sadrže kopije skupa podataka replike. Međutim, podaci na članu sa zakašnjenjem odražavaju ranije, ili kašnjenje stanje skupa. Na primer, ako je trenutno vreme 09:52, a član ima kašnjenje od jednog sata, član sa zakašnjenjem neće imati operacije novije od 08:52.

Budući da članovi sa zakašnjenjem funkcionišu kao "prolazna rezervna kopija" ili stalni "istorijski" snapshot skupa podataka, mogu vam pomoći da se oporavite od različitih vrsta ljudskih grešaka. Na primer, član sa zakašnjenjem može omogućiti oporavak od neuspešnih nadogradnji aplikacija i grešaka operatera, uključujući obrisane baze podataka i kolekcije.

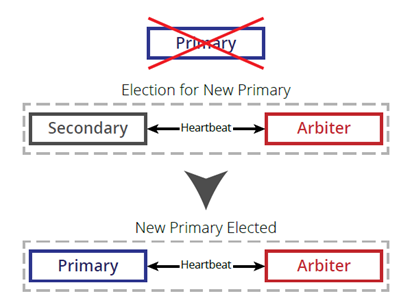
Replikaciona baza podataka i rezervne kopije baza podataka (mirroring) često se mešaju, ali nisu isti proces. Mirroring je oblik replikacije podataka gde se održava kompletna rezervna kopija baze podataka kao sigurnosna mera u slučaju kvara primarne baze podataka. Kao što je gore opisano, replikacija se odnosi na baze podataka i njen glavni cilj obično je operativna efikasnost i veća dostupnost podataka.

**Arbitražni čvor**

U nekim okolnostima (kao što je kada imamo primarni i sekundarni čvor, ali troškovna ograničenja onemogućavaju dodavanje još jednog sekundarnog čvora), možemo odlučiti da dodamo mongod instancu u replika skup kao arbiter. Arbiter učestvuje u izborima, ali ne drži podatke i ne može postati primarni.

Kako bi se izbegli problemi sa konzistentnošću podataka, trebalo bi da postoji samo jedan arbitražni čvor. Da bi se osiguralo da će upis trajati nakon kvara primarnog čvora, većina čvorova treba da potvrdi operaciju pisanja. Arbitri ne čuvaju nikakve podatke, ali doprinose broju čvorova u skupu replika. Ako skup replika ima više arbitražnih čvorova, manja je verovatnoća da će većina čvora biti dostupna nakon kvara. Pomoću njega se štede resursi jer ne zahteva dodatni prostor na disku potreban za replikaciju podataka jer ne sadrži kopije podataka, te postojanje arbitarskog čvora ne povećava opterećenje sistema.

Uloga ovog čvora ne povećava otpornost na kvarove niti povećava dostupnost podataka jer se ne radi o kopiji, već samo pomaže u izboru novog primarnog čvora. Jedina komunikacija između arbitražnog čvora i ostalih članova skupa su glasovi tokom izbora, proces otkucaja srca i podaci o konfiguraciji



Slika 8. Arbitar

**Asinhrone replikacije**

Sekundarni članovi replike repliciraju primarni oplog i primenjuju operacije na svoje skupove podataka asinhrono. Time što sekondarni članovi održavaju podatke u skladu sa primarnim, replikacioni skup može nastaviti da funkcioniše i pored kvara jednog ili više članova.

Oplog (log operacija) je specijalna kolekcija sa ograničenim kapacitetom koja čuva stalni zapis svih operacija koje menjaju podatke u vašim bazama podataka. Ako operacije pisanja ne menjaju podatke ili ne uspeju, ne kreiraju zapise u oplogu.

Za razliku od drugih kolekcija sa ograničenim kapacitetom, oplog može da preraste svoj konfigurisan veličinski limit kako bi izbegao brisanje većine tačke potvrde.

MongoDB primenjuje operacije nad bazom podataka na primarnom članu, a zatim beleži te operacije u primarnom oplogu. Sekundarni članovi zatim kopiraju i primenjuju ove operacije u asinhronom procesu. Svi članovi replikacionog skupa sadrže kopiju oploga u kolekciji local.oplog.rs, što im omogućava da održe trenutni stanje baze podataka.

Kako bi se olakšala replikacija, svi članovi replikacionog skupa šalju heartbeat-ove (pingove) svim drugim članovima. Svaki sekundarni član može da uveze zapise oploga od bilo kog drugog člana.

Svaka operacija u oplogu je idempotentna. To znači da operacije oploga daju iste rezultate, bez obzira na to da li su primenjene jednom ili više puta na ciljnom skupu podataka.  
  
 Kada prvi put pokrenemo član replikacionog skupa, MongoDB kreira oplog sa podrazumevanom veličinom ako ne navedemo veličinu oploga. U većini slučajeva, podrazumevana veličina oploga je dovoljna. Na primer, ako oplog zauzima 5% slobodnog prostora na disku i napuni se u toku 24 sata operacija, sekundarni članovi mogu prestati da kopiraju zapise iz oploga do 24 sata bez da postanu previše zastare za nastavak replikacije. Međutim, većina replikacionih skupova ima mnogo manji obim operacija, a njihovi oplogovi mogu da drže mnogo veći broj operacija.

Pre nego što mongod kreira oplog, može se navesti njegova veličina koristeći opciju oplogSizeMB. Nakon što ste prvi put pokrene član replikacionog skupa, korišćenjem administrativne komande replSetResizeOplog da se promeni veličina oploga. replSetResizeOplog omogućava da se dinamički promeni veličina oploga bez ponovnog pokretanja mongod procesa.  
  
 Minimalni broj sati tokom kojih će operacije u oplog-u biti očuvane mogu se konfigurirati kako bi se obezbedilo da podaci ostanu dostupni duže vreme. Kada se postavi minimalni period očuvanja, operacije će biti uklonjene iz oplog-a samo ako oplog dostigne svoju maksimalnu konfiguriranu veličinu i ako je operacija starija od ovog minimalnog perioda očuvanja.

Podrazumevano, MongoDB automatski briše najstarije operacije iz oplog-a kako bi održao njegovu veličinu unutar konfigurisanih granica. Da bi se postavio minimalni period očuvanja tokom inicijalnog pokretanja mongod-a, može se:

- Dodati opcija `storage.oplogMinRetentionHours` u konfiguracioni fajl mongod-a

- Koristiti opcija `--oplogMinRetentionHours` u komandnoj liniji prilikom pokretanja mongod-a

Ako je potrebno da dodje do izmene minimalnog perioda očuvanja oplog-a nakon početnog pokretanja, može se koristiti komanda `replSetResizeOplog`. Ova komanda omogućava dinamično prilagođavanje veličine oplog-a i, ako se postavi minimalni period očuvanja, on će prevazići sve prethodno postavljene vrednosti

Ako postoji tročlani skup replika, članove je moguće distribuirati na sledeće načine:

* Dva centra podataka – dva člana bi se nalazila u centru podataka 1, a jedan član u centru podataka 2. Ukoliko je jedan od članova arbitražni čvor bilo bi dobro da se on nalazi u centru podataka 1 zajedno sa primarnim čvorom koji nosi podatke. Ako centar podataka 1 prestane sa radom, skup replika počinje da služi samo za operacije čitanja, jer su operacije pisanja moguće samo na primarnom čvoru. Ako prestane sa radom centar podataka 2, skup replika i dalje može da prima operacije čitanja i pisanja.
* Tri centra podataka – sva tri člana skupa replika idu u poseban centar podataka. Ako bilo koji centar podaka postane nedostupan, skup replika i dalje zadržava mogućnost pisanja jer preostali članovi mogu održati izbore.

Dobra praksa je da se članovi skupa replika distribuiraju u bar dva centra podataka, jer u slučaju da se jedan pokvari podaci su i dalje dostupni za čitanje. Ako postane nedostupan onaj centar podataka koji ima manje članova, skup replika i dalje može služiti za pisanje i za čitanje, međutim ako postane nedostupan onaj centar sa više članova pisanje postaje nemoguće.

Ako postoji petočlani skup replika, članove je moguće distribuirati na sledeće načine

Mogući načini distribucije:

* Dva centra podataka – tri člana idu u centar podataka 1, a dva člana u centar podataka 2. Ako centar podataka 1 prestane sa radom, skup replika će primati samo operacije čitanja, jer se u njemu nalazi većina, a ako centar podataka 2 prestane, skup replika će primati i pisanje.
* Tri centra podataka – po dva člana u centre podataka 1 i 2 i jedan član u centar podataka 3. Ako se neki od centara pokvari, skup replika ostaje i za operacije pisanja.

**Povratak (Rollback) tokom failover-a u replikacionom skupu**

Povratak (rollback) vraća zapisane operacije na prethodnom primarnom čvoru kada se član ponovo pridruži svom replikacionom skupu nakon failover-a. Povratak je potreban samo ako je primarni čvor prihvatio zapisane operacije koje sekundarni čvorovi nisu uspeli da repliciraju pre nego što je primarni čvor odstupio. Kada se primarni čvor ponovo pridruži skupu kao sekundarni čvor, vraća, ili "vrati", svoje zapisane operacije kako bi očuvala doslednost baze podataka sa drugim članovima.

MongoDB pokušava da izbegne povratke, koji bi trebali biti retki. Kada se povratak dogodi, to je često rezultat mrežnog particionisanja. Sekundarni čvorovi koji ne mogu da prate obim operacija na prethodnom primarnom čvoru povećavaju veličinu i uticaj povratka.

Povratak se ne dešava ako se zapisane operacije repliciraju na drugog člana replikacionog skupa pre nego što primarni čvor odstupi i ako taj član ostane dostupan i pristupačan većini članova replikacionog skupa.

**Sinhronizacija**

Da bi se očuvale ažurne kopije podataka u deljenom skupu, sekundarni članovi replikacionog skupa u MongoDB-u prolaze kroz sinhronizaciju. U MongoDB-u postoje dva glavna tipa sinhronizacije podataka: inicijalna sinhronizacija i replikacija tekućih promena.

Inicijalna sinhronizacija prenosi sve podatke sa jednog člana replikacionog skupa na drugi. Moguće je odabrati početni izvor za sinhronizaciju koristeći parametar `initialSyncSourceReadPreference`, koji se može postaviti samo prilikom pokretanja `mongod` procesa. Inicijalna sinhronizacija može se obavljati na dva načina: Logičkom sinhronizacijom ili kopiranjem datoteka.

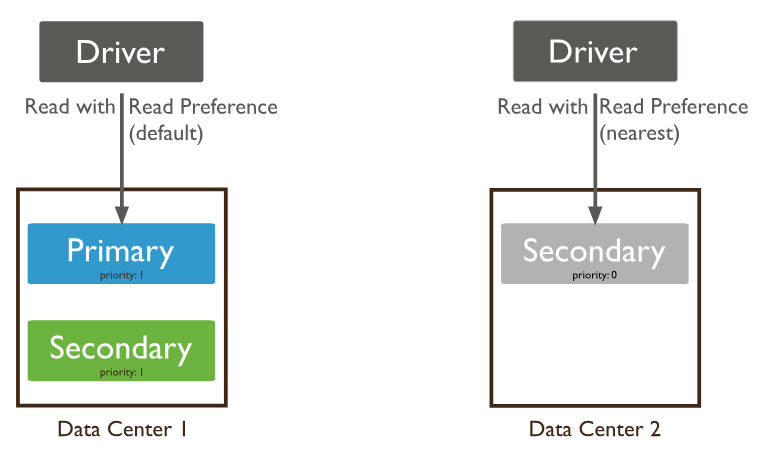
U logičkoj sinhronizaciji, svi podaci osim lokalne baze se kloniraju. MongoDB skenira svaku kolekciju u izvornoj bazi podataka i upisuje podatke u svoje lokalne kopije tih kolekcija. Takođe se prave indeksi za svaku kolekciju, a tokom kloniranja se povlače novododati oplog zapisi. Bitno je da ciljna instanca ima dovoljno slobodnog prostora na disku da privremeno skladišti oploge dok se vrši kopiranje. MongoDB koristi oplog iz izvornog čvora kako bi ažurirao svoje podatke i očuvala trenutnu konzistentnost.

Kod metode inicijalne sinhronizacije koja je zasnovana na kopiranju datoteka sinhronizacije započinje proces kopiranja i premještanja datoteka u sistem datoteka, što može biti brže od logičke inicijalne sinhronizacije. Da bi se omogućila ova vrsta sinhronizacije, potrebno je postaviti parametar `initialSyncMethod` na `fileCopyBased` na odredišnom čvoru, što se može učiniti samo prilikom pokretanja. Ova metoda menja lokalnu bazu podataka ciljnog člana sa bazom podataka iz izvornog čvora. Međutim, tokom ovog procesa, rezervne kopije sa izvornog čvora ne mogu se praviti.

Ako sekundarna replika koja vrši sihronizaciju naiđe na grešku tokom procesa sinhronizacije, on ponovo pokreće sinhronizaciju od početka. Sekundarna inicijalna sinhronizacija može pokušati da nastavi proces sinhronizacije ako je prekinuta greškom ili pod nekim drugim uslovom. Sekundarna replika pokušava da nastavi inicijalnu sinhronizaciju tokom 24h. Dodaje se parametar *initialSyncTransientErrorRetryPeriodSeconds* za kontrolu količine vremena tokom kojem sekundarna replika pokušava da nastavi inicijalnu sinhronizaciju. U slučaju da nije moguće da uspešno nastavi proces sinhronizacije tokom ovog vremena, bira se novi izvor iz skupa replika i ponovo pokreće proces sinhronizacije.

Sekundarni članovi kontinuirano kopiraju podatke nakon inicijalne sinhronizacije. Oni kopiraju oplog primarnog i te operacije primenjuju u asinhronom procesu. MongoDB primenjuje operacije pisanja u serijama koristeći više niti. Pravi grupe prema ID-u dokumenta i istovremeno primenjuje grupe operacija koristeći drugu nit. Operacije pisanja se uvek primenjuju u onom redosledu kako su u originalu primenjene. Čitanje iz snimaka obezbeđuje konzistentan prikaz podataka i omogućava da se čitanje odvija istovremeno sa tekućom replikacijom bez potrebe za zaključavanjem. Administratori mogu ograničiti brzinu kojom primarni čvor primenjuje operacije sa ciljem da ciljem da se zadrži urezano kašnjeje ispod konfigurabilne maksimalne vrednosti *flowControlTargetLagSeconds*.

## **Read preference**

Read preference opisuje kako klijenti MongoDB-a usmeravaju operacije čitanja na članove replikacionog skupa.  
  


Slika 9. Read preference

Podrazumevano, aplikacija usmerava svoje operacije čitanja na primarni član u replikacionom skupu (tj. režim read preference "primary"). Međutim, klijenti mogu da navedu read preference kako bi usmerili operacije čitanja na sekundarne članove.  
Svi režimi read preference osim "primary" mogu vratiti zastarele podatke, jer sekundarni članovi repliciraju operacije sa primarnog čvora u asinhronom procesu.

Read preference ne utiče na vidljivost podataka; tj. klijenti mogu videti rezultate pisanja pre nego što budu potvrđeni ili se propagiraju većini članova replikacionog skupa.

Read preference ne utiče na uzročnu doslednost. Garancije uzročne doslednosti koje pružaju uzročno dosledne sesije za operacije čitanja sa "majority" write concern-om i operacije pisanja sa "majority" write concern-om važe za sve članove MongoDB implementacije.

Preferencija za čitanje se sastoji od režima preferencije čitanja i liste skupova oznaka, opciono.

Postoji nekoliko modova:

1) primary – sve operacije čitanja koriste samo trenutni primarni skup replika i ovo je podrazumevani režim čitanja. U slučaju da primarni čvor nije dostupan, dolazi do greške prilikom operacije čitanja ili do izuzetka.

2) primaryPreferred – u većini slučajeva, operacije čitanja se obavlja iz primarnog čvora. Međutim, ako je primarni čvor nedostupan, operacija čitanja se obavlja sa sekundarnih čvorova. Kada preferencija čitanja uključuje listu oznaka i ne postoji primarni čvor, klijent pokušava da pronađe sekundarne članove sa odgovarajućim oznakama. Ako se pronađu podudarni sekundari, klijent nasumično bira sekundar iz najbliže grupe. Ako nijedan sekundar nema odgovarajuću oznaku, dolazi do greške. Čitanje pomoću ovog režima može da vrati zastarele podatke.

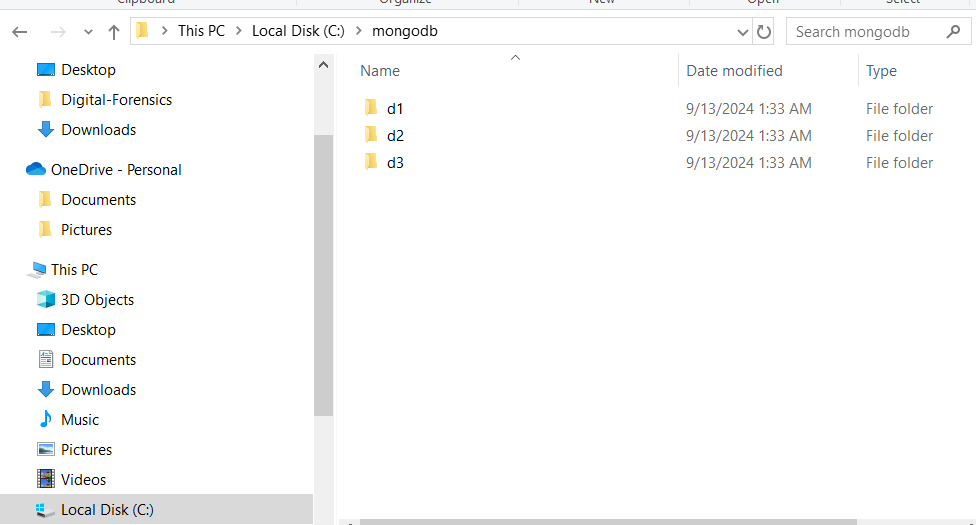
3) secondary – operacije čitanja se obavljaju samo na sekundarnim čvorovima i ukoliko oni nisu dostupni dolazi do greške prilikom ove operacije. Kada uključuje listu oznaka, klijent pokušava da pronađe sekundarne članove sa odgovarajućim. Ako se pronađu podudarni sekundari, klijent bira nasumični sekundar iz najbliže grupe odgovarajućih sekundarnih podataka.

4) secondaryPreferred – u većini slučajeva čitanje se obavlja iz sekundarnih replika, ali u ako u skupu replika postoji samo primarni član onda će se čitanje obaviti iz primarnog.

5) nearest – čitanje se obavlja sa člana čija je mrežna latencija najprihvatljivija, bez obzira na to da li je to primarni ili sekundarni čvor, u ovom režimu su oni jednaki. Pomoću ovog režima minimizira se kašnjenje.

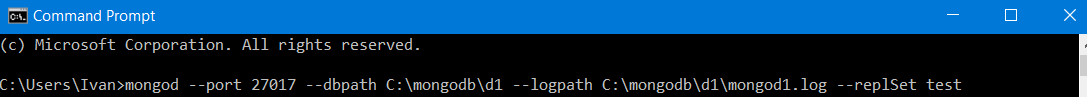
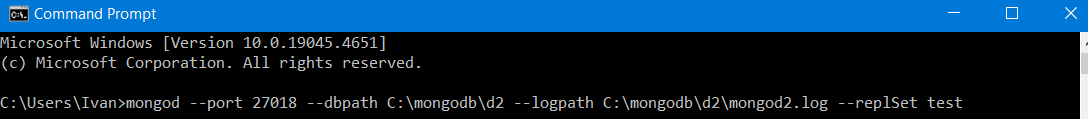
**Praktični primer**

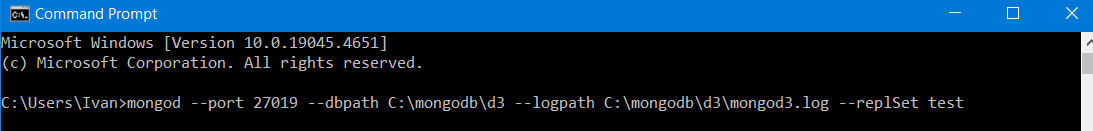
U ovom poglavlju biće naveden praktični primer nad replika skupom. Da bismo to uradili, prvo je potrebno kreirati tri direktorijuma, bilo gde na računaru. To se može uraditi ručno ili komandama: mkdir d1, mkdir d2, mkdir d3, respektivno, pozicionirajući se prvo u željeni direktorijum.



Slika 10. Direktorijumi za praktičan primer

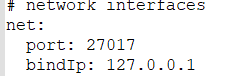
Dalje, potrebno je otvoriti tri cmd prompta i u svakom od njih ukucati sledeće komande.

  
 Slika 11. Komanda za pokretanje MongoDB instance   
 Slika 12. Komanda za pokretanje MongoDB instance



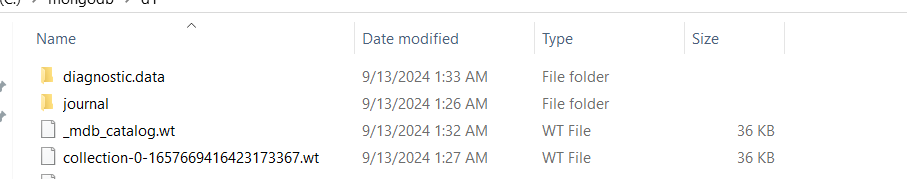
Slika 13. Komanda za pokretanje MongoDB instance

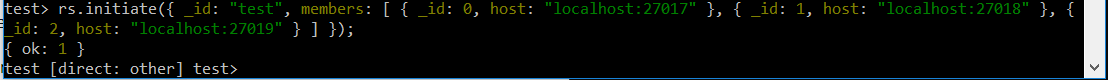
Razlog zbog čega su odabrani portovi 27017, 27018 i 27019, je zbog toga što je u mongod.exe fajlu port postavljen na 27017.



Slika 14. Port number kod mongod.exe fajla

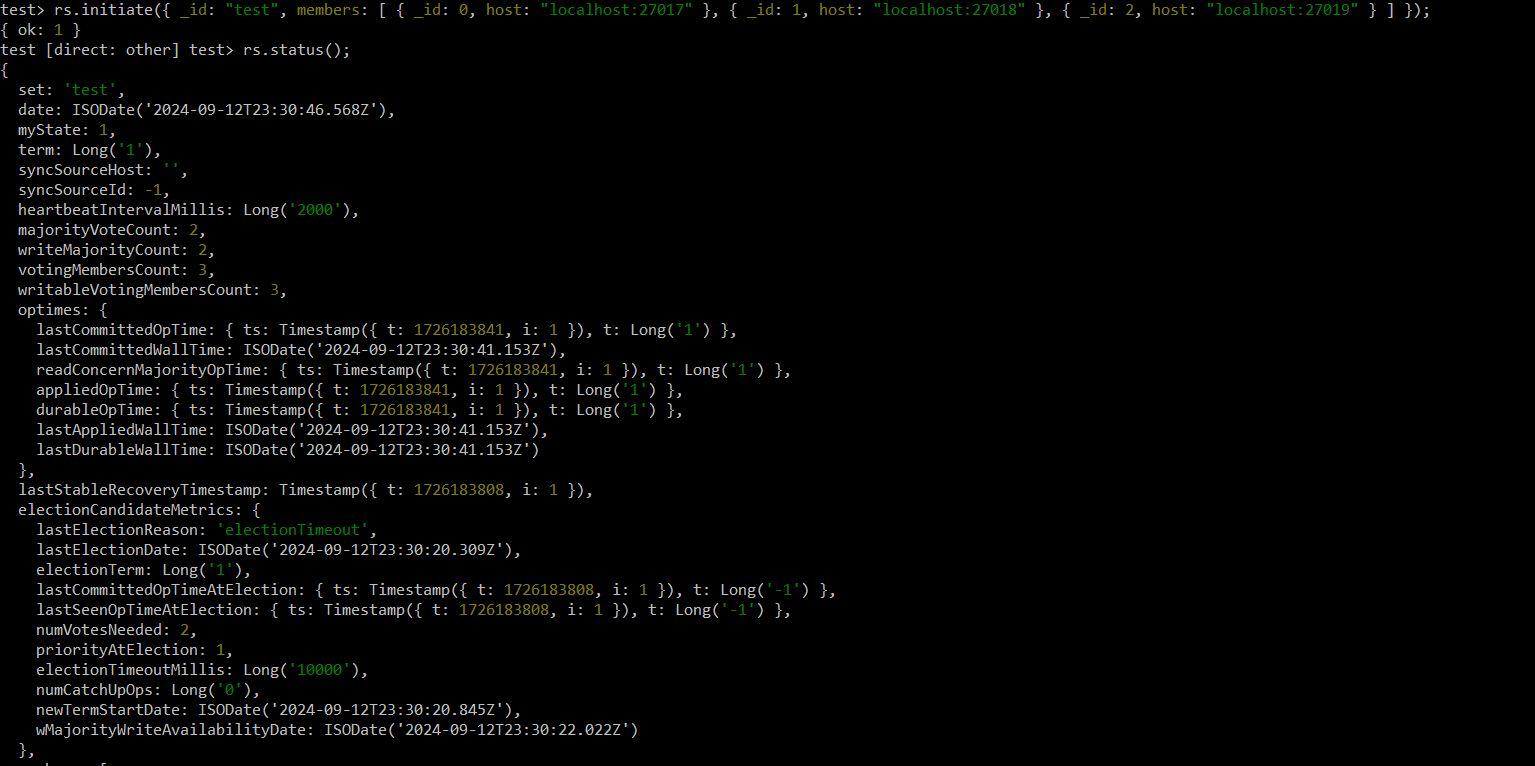
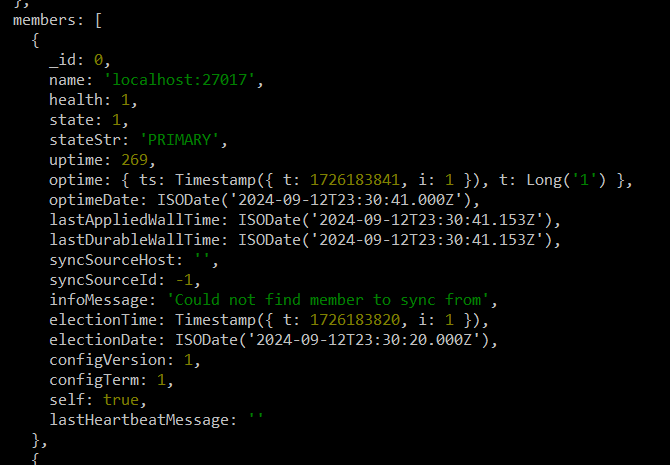
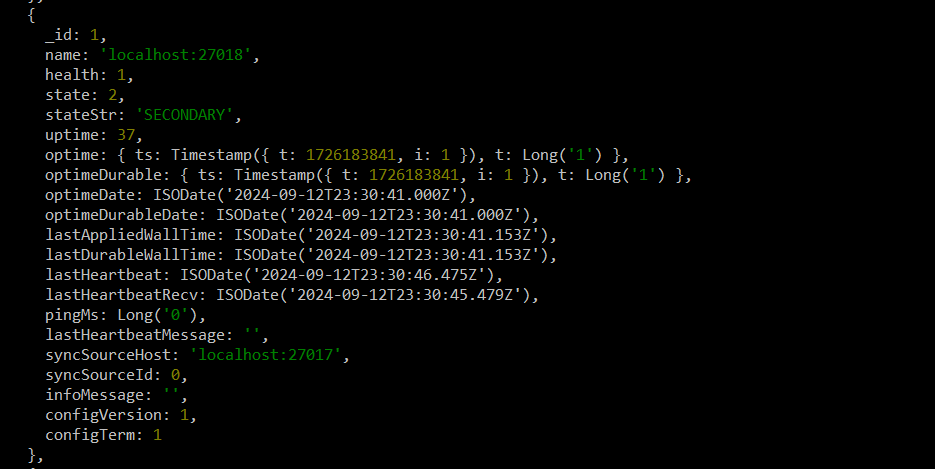
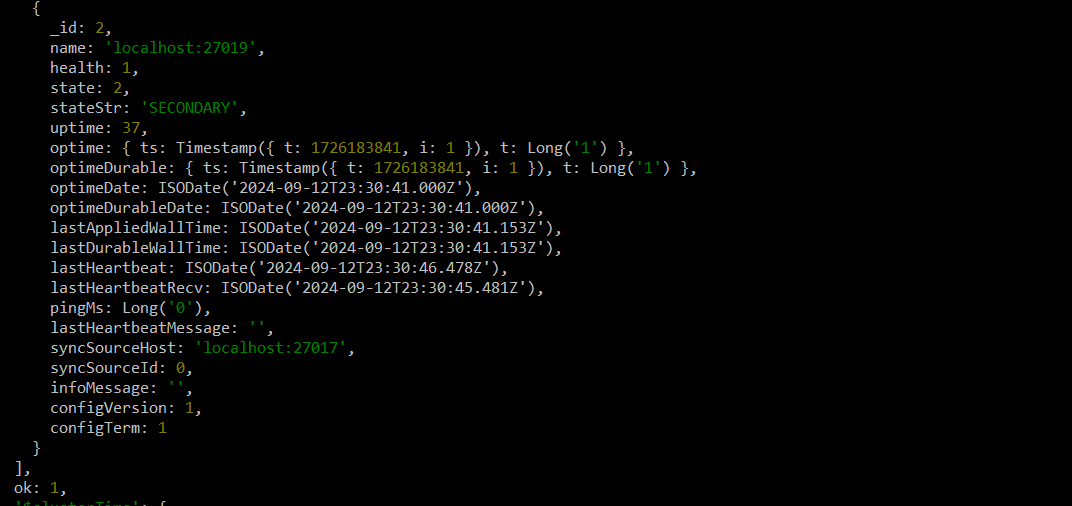
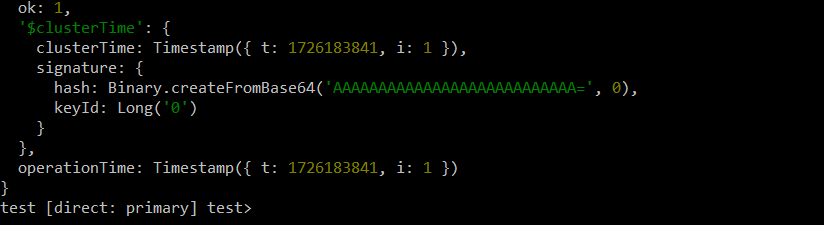
Izgled datoteka nakon izvršene komande je:

  
 Slika 15. Izgled direktorijuma nakon izvršene komande iz cmd-a

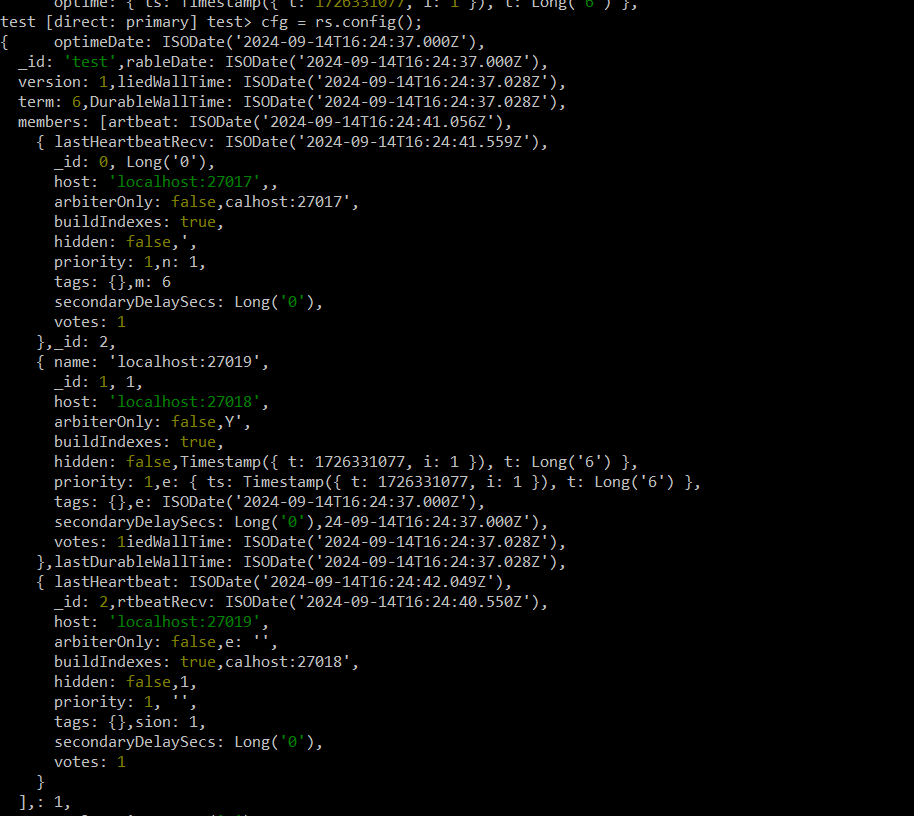
U sledećem koraku inicijalizujemo skup replika. U mongosh terminalu, i kucamo sledeću komandu:   
  
 Slika 16. Inicijalizacija skup replika  
  
 Ova komanda inicijalizuje replikacioni skup sa tri člana. \_id je identifikator replikacionog skupa, a members sadrži listu članova sa njihovim jedinstvenim \_id i host.

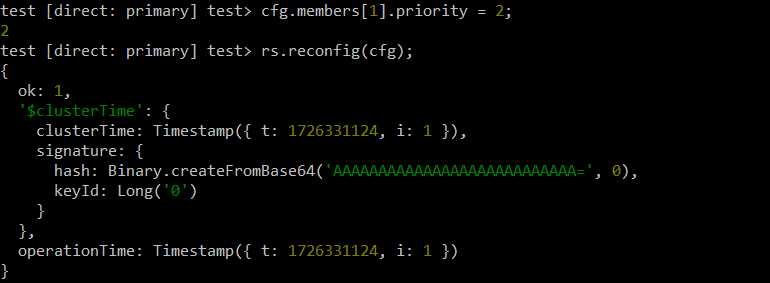
Kao odgovor dobijamo ok : 1, što znači da je izvršenje ove komande uspešno.

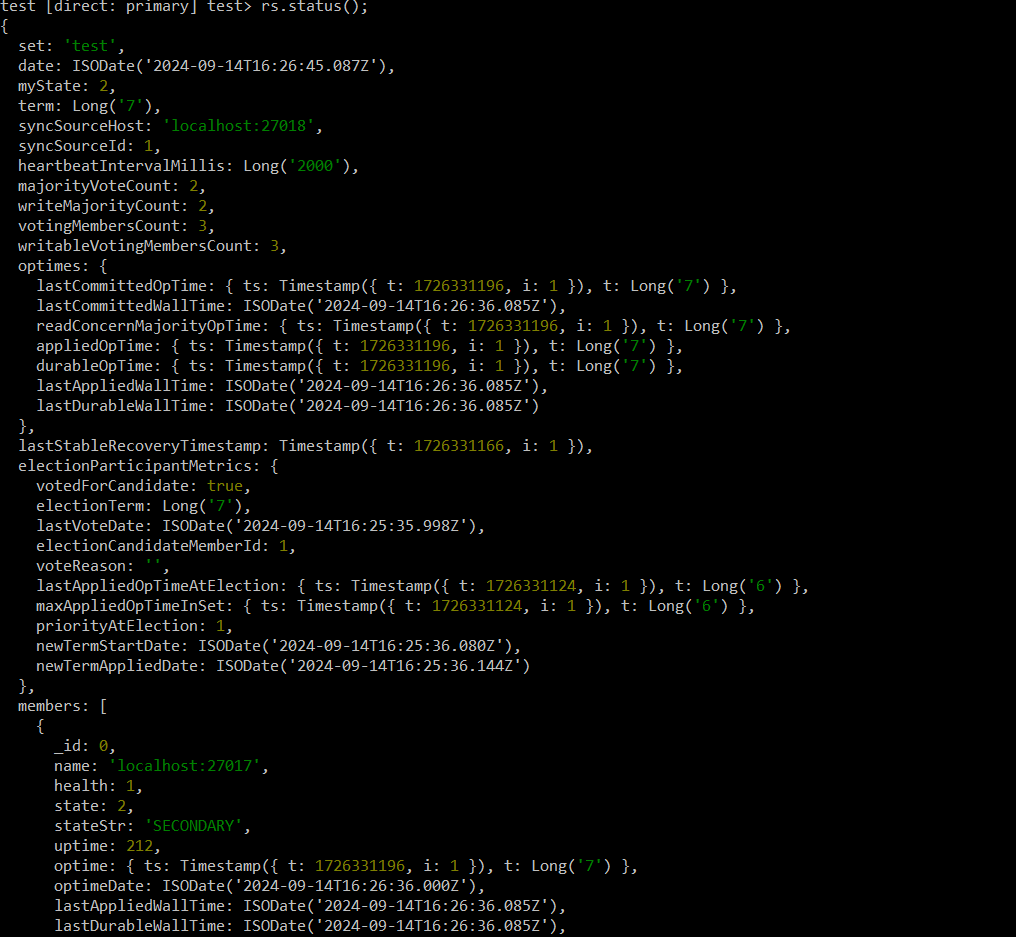
Nakon inicijalizacije, proveravamo status skupa replike da bismo videli da li su članovi pravilno dodati komandom rs.status(): I kao rezultat dobijamo podatke sa slika ispod.

  
 Slika 17. Detalji trenutnog statusa skupa replika  
  
Slika 18. Detalji trenutnog statusa skupa replika  
  
Slika 19. Detalji trenutnog statusa skupa replika  
  
Slika 20. Detalji trenutnog statusa skupa replika  
  
Slika 21. Detalji trenutnog statusa skupa replika

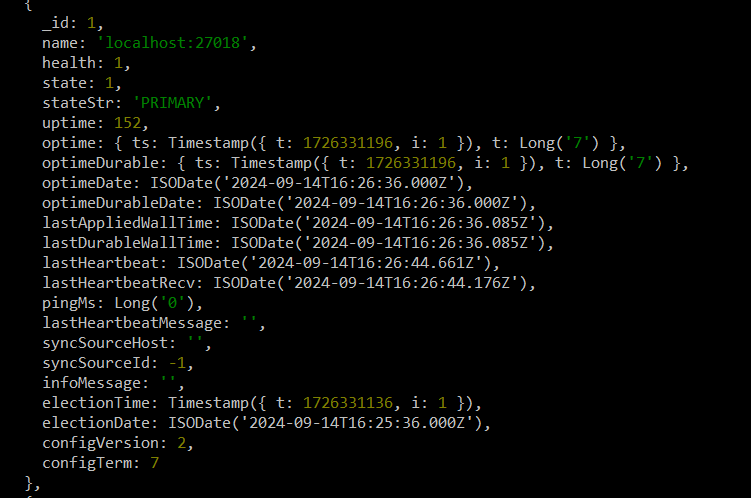
Dalje, izazvaćemo automatski failover, iz gornjih slika možemo videti da je član na localost:27017 primarni, dok su ostala dva sekundarni. Ukoliko promenimo prioritet nekog od ta dva sekundarna člana, kada dodje do neke greške sa primarnim čvorom (u našem slučaju ubijanja), primarni čvor će postati onaj koji ia najveži prioritet. Na sledećim slikama su komande i koraci do postizanja toga.

  
Slika 22. Uzimanje i prikazivanje trenutne konfiguracije

Slika 23. Promena prioriteta drugog člana



Slika 24. Promena primarnog čvora

  
Slika 25. Promena primarnog čvora

# **Zaključak**

Replikacija podataka u MongoDB-u predstavlja ključni mehanizam za očuvanje visoke dostupnosti i pouzdanosti sistema, omogućavajući organizacijama da održe kontinuitet poslovanja i minimiziraju rizike povezane sa gubicima podataka. Kroz implementaciju replika skupova, MongoDB pruža fleksibilne i efikasne načine za zaštitu podataka, obezbeđujući da svi kritični podaci budu dostupni čak i u slučaju kvara ili otkaza jednog ili više čvorova.

Pored osnovnih funkcionalnosti replikacije, kao što su sinhronizacija podataka između primarnog i sekundarnih čvorova, MongoDB nudi napredne opcije za upravljanje replikacijom, uključujući prilagodbu prioriteta članova, konfiguraciju opcija za očuvanje podataka i optimizaciju performansi. Ove opcije omogućavaju administratorima da precizno prilagode replikaciju prema specifičnim potrebama i zahtevima svojih sistema.

Na taj način, MongoDB ne samo da omogućava visoku dostupnost i otpornost na greške, već takođe poboljšava ukupnu efikasnost sistema i korisničko iskustvo. Kao rezultat toga, replikacija u MongoDB-u je ključna komponenta u dizajnu savremenih baza podataka, pružajući robusnu osnovu za upravljanje podacima u dinamičnim i izazovnim okruženjima.

**Literatura**<https://www.ibm.com/topics/data-replication>  
<https://www.qlik.com/us/data-replication/database-replication>  
<https://shantanubansal.medium.com/demystifying-mongodb-oplog-a-comprehensive-guide-with-oplog-entry-examples-3bd716789f78>  
<https://www.mongodb.com/docs/manual/replication/>   
<https://www.mongodb.com/docs/manual/core/replica-set-primary/>   
<https://www.mongodb.com/docs/manual/core/replica-set-oplog/>   
<https://www.mongodb.com/docs/manual/core/replica-set-elections/>   
<https://www.mongodb.com/docs/manual/core/replica-set-secondary/>   
<https://www.mongodb.com/docs/manual/core/replica-set-arbiter>/  
<https://www.mongodb.com/docs/manual/core/replica-set-rollbacks/>   
<https://www.mongodb.com/docs/manual/core/replica-set-write-concern/>   
<https://www.mongodb.com/docs/manual/core/read-preference/>   
<https://www.mongodb.com/docs/manual/tutorial/deploy-replica-set/>