**Univerzitet u Nišu**

**Elektronski fakultet**

**MongoDB klaster rešenja**

**Student:** Jovana Nikolić 1147

**Mentor:** Doc. dr Aleksandar Stanimirović

# Sadržaj

[Sadržaj 2](#_Toc75740651)

[Uvod 3](#_Toc75740652)

[Skupovi replika u MongoDB bazi podataka 4](#_Toc75740653)

[Klasteri u MongoDB-u 4](#_Toc75740654)

[Skup replika 4](#_Toc75740655)

[Podešavanje klastera lokalno 6](#_Toc75740656)

[Dokument konfiguracije replike 9](#_Toc75740657)

[Komande za replikaciju 10](#_Toc75740658)

[Oplog fajl 12](#_Toc75740659)

[Failover i izbori 13](#_Toc75740660)

[Write concern 14](#_Toc75740661)

[Read concern 14](#_Toc75740662)

[Read preference 15](#_Toc75740663)

[Shard-ovani klasteri 17](#_Toc75740664)

[Arhitektura shard-ovanog klastera 17](#_Toc75740665)

[Podešavanje shard-ovanog klastera 18](#_Toc75740666)

[Balansiranje 21](#_Toc75740667)

[Zaključak 23](#_Toc75740668)

[Literatura 24](#_Toc75740669)

# Uvod

MongoDB baza podataka je najpopularnija i najkorišćenija NoSQL baza podataka, spada u dokumentu orjentisane NoSQL baze podatala. Koristi JSON ili BSON dokumente za smeštanje podataka. MongoDB baza podataka je postalo open-source 2009. godine, pri čemu je jedna od njenih glavnih karakteristika performanse. MongoDB je nerelaciona baza koja je pravljena kao jako bliska relacionim bazama podataka, sa mogućnošću kreiranja sopstvene šeme, koja je jako fleksibilna [1]. Umesto korišćenja tabela za smeštanje podataka, podaci su smešteni u kolekcijama, dok umesto jasno definisanih i uređenih redova kod relacionih baza podataka, kod MongoDB baze podataka, podaci se smeštaju u dokumentima, koja se sastoje od key-value parova, ime key-value para predstavlja polje, koje predstavlja pandan kolonama kod relacionih baza podataka [2]. MongoDB baza podataka je od svih NoSQL baza podataka, baza koja najviše liči na relacione baze podataka, i koja kao cilj ima posedovanje mnogih svojstava relacionih baza podataka.

Klaster baze podataka, je klaster računara koji izvršava instancu DBMS-a na svakom čvoru. On je proces kombinovanja više servera ili instanci koje povezuju bazu podataka. Javljaju se kao rešenje u slučajevima kada samostojeći server ne može da se nosi sa planiranom količinom podataka ili upitima. Glavni razlog za klasterizaciju baza podataka jeste postizanje redundanse podataka, balansiranja, visoke dostupnosti, kao i praćenja i automatizacije. Kao jedna od baza podataka koja za svoju karakteristiku navodi rad sa velikom količinom podataka, MongoDB pruža puno mogućnosti kada je u pitanju klasterizacija podataka [3].

Za praćenje koraka pri podešavanju klastera je neophodno instalirati MongoDB bazu podataka sa verzijom koja je veća od 3.6. Kroz teoriju i primere će u radi biti prikazana klasterizacija podataka kod MongoDB baze podataka.

# Skupovi replika u MongoDB bazi podataka

Za razliku od jednoserverskih MongoDB baza podataka, MongoDB klaster omogućava MongoDB bazi podataka da se skalira ili horizontalno kroz više servera uz shardovanje, ili omogućava replikaciju podataka, time garantujući visoku dostupnost kroz MongoDB skupove replika, samim tim poboljšavajući sveukupne performanse i pouzdanost MongoDB klastera.

## Klasteri u MongoDB-u

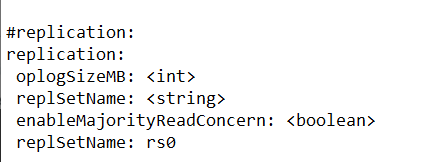
U kontekstu MongoDB-a, klaster je reč koja se koristi za skup replika ili za skup isparčanih klastera. MongoDB baza podataka koristi asinhrone statement based replikaciju jer je platformama nezavisna, i dozvoljava više fleksibilnosti u oviru samog seta replika. Replikacija je koncept čuvanja više kopija podataka. Replikacija je neophodna, jer idealan sistem ne postoji, te uvek postoji realna mogućnost otkaza servera, ili server nije dostupan zbog održavanja. Poenta replikacije jeste da podaci budu dostupni u svakom trenutku. To dovodi do koncepta dostupnosti. Baza podataka koja ne koristi replikaciju ima samo jedan server, i oni se zovu stamostojeći čvorovi. Ukoliko taj čvor prestane da radi, podaci se mogu izgubiti, i server više ne odgovara na upućene zahteve za čitanje ili upis. Kao rešenje tog problema se javila mogućnost replikacije, odnosno čuvanja podataka u više kopija na više servera. U okviru MongoDB baze podataka, skup računara koji imaju iste podatke se nazivaju skupom replika. U samom skupu, sve operacije se obavljaju nad jednim čvorom, a na ostalim čvorovima je da drže korak sa njim kroz asinhroni mehanizam. Čvor gde se šalju podaci se zove primar, a svi ostali čvorovi su sekundari. Cilj je da svi čvorovi budu konzistentni među sobom, te u slučaju da primar prestane da radi, jedan od sekundarnih čvorova može da postane primar u procesu koji se naziva failover. Čvorovi odlučuju koji će čvor postati primar kroz izbore. Sami čvorovi glasaju među sobom, ko treba biti primar. Čvor koji je prestao da radi, ukoliko može da sustigne informacije i podatke ostalih čvorova, se može automatski uključiti ponovo u rad sistema. Shard-ovan klaster predstavlja horizontalno skaliranje, pri čemu su podaci distribuirani kroz više servera. Glavna uloga isparčane MongoDB baze podataka jeste skaliranje operacija upisa i čitanja kroz više krhotina [4].

## Skup replika

Skup replika je skup mongod-a koje dele kopije istih informacija između sebe. Članovi skupa replika mogu imati jednu od 2 uloge. Mogu biti u ulozi primara, pri čemu su sve operacije čitanja i upisa, uslužene od strane ovog čvora. Ili mogu biti u ulozi sekundarnog čvora, pri čemu je njegova odgovornost , repliciranje svih informacija sa primara, i zatim budu visoko dostupni za slučaj prestanka rada primara. Upis podataka u skup replika, obavlja se podsredstvom primarnog čvora. Zatim su podaci replicirani na sekundarne čvorove. Mehanizam replikacije zavisi od protokola koji definiše na koji način sekundarni čvorovi čitaju podatke sa primara. Kod MongoDB baze podatka ovo je asinhroni protokol replikacije koji može imati različite verzije. Postoje verzije PV0 i PV1. Različite verzije imaju razlike u načinu postizanja trajnosti i dostupnosti kroz skup replika. Trenutno je PV1, difoltna verzija protokola, on je baziran na RAFT protokolu. Najbitniji deo ovog mehanizma predstavlja log operacija ili oplog. Oplog predstavlja log baziran na operacijama, koji za cilj ima praćenje svih operacija upisa koje su potvrđene od strane skupa replika. Svaki put kada je operacija upisa uspešno izvršena na primaru, ona se pamti u logu operacija u njenoj idempotentoj[[1]](#footnote-1) formi. Osim uloge primara i sekundarnog čvora, čvor može biti u arbitrarnoj ulozi (što se ne savetuje, jer izaziva probleme sa konzistencijom). Član arbitar ne čuva nikakve podatke pri sebi. Služi samo za odlučivanje pri izboru primara među sekundarnim čvorovima. On nikada ne može postati primar. Skup replika je otporan na otkaz, što znači da imaju failover mehanizam koji zahteva da većina čvorova u skupu replika budu slobodni da u slučaju otkaza izaberu novog primara. Bez čvora primara se ne može vršiti upis, te se od ostatka čvorova odlučuje ko će biti primaran čvor, i to se dešava kroz izbore, sam izbor zavisi od verzije protokola. Da bi set replika se smatrao dostupnim neophodno je da polovina čvorova bude dostupna. Najčešće se savetuje da je broj replika u setu neparan, jer paran broj replika, iako veći dozvoljava jednak broj otkaza kao i prvi neparan broj iza parnog, te taj jedan čvor je samo redundansa koja ne povećava dostupnost. Maksimalan broj replika u jednom skupu je 50, što je korisno za geografsku distribuciju podataka, ali samo sedam članova maksimalno može učestvovati u izborima, jer bi u suprotnom glasanje previše dugo trajalo, bez ikakvog poboljšanja u pogledu dostupnosti. Od tih 7 članova, jedan će biti primaran, dok će ostali moći biti izabrani u slučaju otkaza primara. Među sekundarnim čvorovima, mogu će definisati takozvani skriveni čvorovi, oni mogu imati posebno opterećenje koje omogućava samo čitanje podataka, ili kopije podataka koje su skrivene od ostatka aplikacija. Skriveni čvorovi takođe mogu da budu zakasneli, time one bivaju syncovani sa primarom nakon određenog vremenskog perioda, što je korisno u situacijama, kada se napravi greška na primaru, kako se ne bi koristili backup fajlovi [4].

# Podešavanje skupa replika lokalno

Ovi koraci, predstavljaju korake neophodne za podešavanje MongoDB klastera na Windows operativnom sistemu. Prvo se pronalazi konfiguracioni fajl u bin folderu foldera MongoDB instalacije, putanja bi trebalo da izgleda *<direktorijum u kome je instaliran MongoDB>/bin/mongod.cfg.* U okviru konfiguracionog fajla dodaju se opcije za replikaciju, pri ovakvoj deklaraciji je veličina oploga postavljena na difoltnu vrednost, kao i *enableMajorityReadConcern.* Ime seta replike je postavljeno na rs0. Na sledećoj slici je prikazan isečak konfiguracionog fajla vezan za replikaciju.



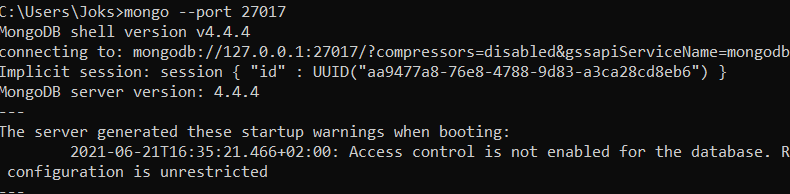
*Slika 1 – Podešavanje replikacije u konfiguracionom fajlu*

Nakon ovog podešavanja, ukoliko instanca mongod-a već radi, trebalo bi je isključiti, zatim otvoriti komandni promt u administrativnom mod-u, i restartovati mongod instancu kao član skupa replika na način na koji je prikazan na sledećoj slici. Opcija *replSet* se odnosi na ime skupa replika koji je već postavljen prethodno u *mongod.cfg* fajlu.



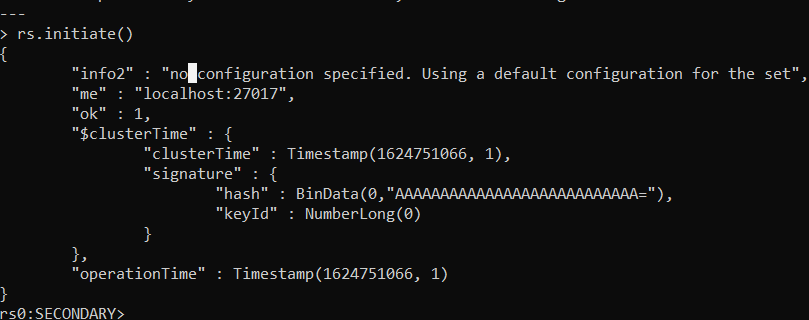
*Slika 2 – Pokretanje mongod instance*

Nakon izvršenja mongod instanca je započeta. Sledeći korak je otvaranje novog komandnog prompta i povezivanje sa instancom na difoltnom portu pomoću mongo komande, što je prikazano na slici ispod.



*Slika 3 – Početak nove instance*

Nakon povezivanja se skup replika inicira komandom *rs.replicate()* čije je dejstvo prikazano na slici 4.



*Slika 4 – Inicijacija skupa replika*

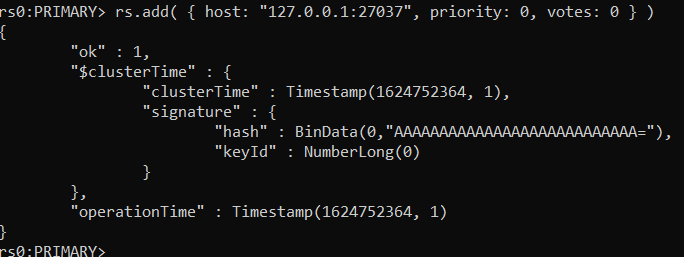
Nakon izvršenja ovog koraka skup replika radi, i mogu se dodavati novi članovi, na novim komandnim prompt-ovima su kreirane nove instance koje osluškuju portove 27027 i 27037 redom, i zatim se dodaju u skup replika pomoću komande *rs.add() .* Na sledećim slikama će biti prikazano dejstvo narednih naredbi.



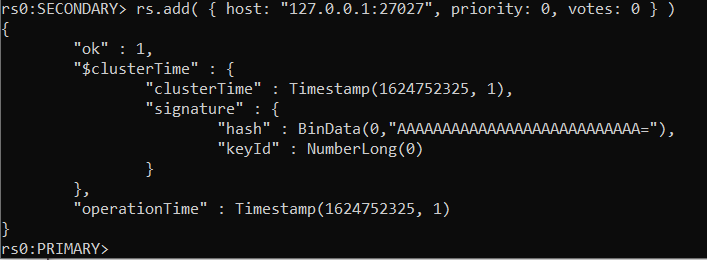
*Slika 5 – Pokretanje druge instance*



*Slika 6 – Pokretanje treće instance*

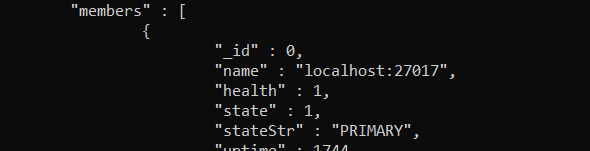


*Slika 7 – Dodavanje druge instance u skup replika*

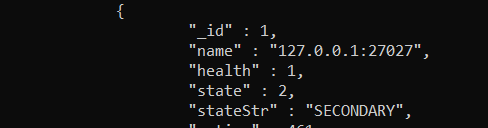


*Slika 8 – Dodavanje treće instance u skup replika*

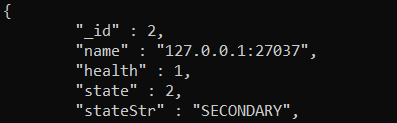
Nakon izvršenja prethodnih naredbi, u skupu replika ima jedan primarni čvor i dva sekundarna čvora u skupu replika, kako bi se to potvrdilo može se koristiti naredba *rs.status(),* čije je dejstvo prikazano na slikama 9,10 i 11 [5].



*Slika 9 – Potvrda postojanja primarnog čvora na portu 27017*



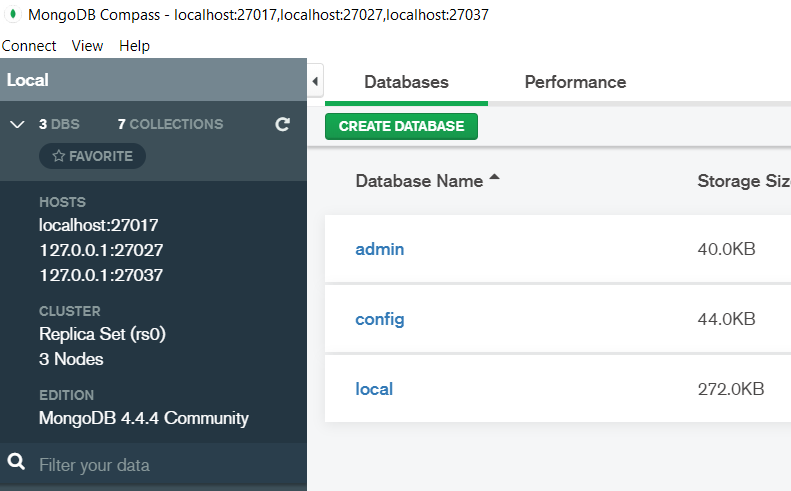
*Slika 10 – Potvrda postojanja sekundarnog čvora na portu 27027*



*Slika 11 – Potvrda postojanja sekundarnog čvora na portu 27037*

Na klaster se može povezati i koristeći MongoDB Compass alat, čiji je prikaz nakon povezivanja prikazan na slici 12, i to sa navedenom kofiguracijom, navodeći kao konekcioni string:

“mongodb://localhost:27017,localhost:27027,localhost:27037/?replicaSet=rs0”



*Slika 12 – Slika MongoDB Compass alata nakon konekcije na skup replika*

## Dokument konfiguracije replike

Dokument konfiguracije replika je BSON dokument, kojim upravljamo pomoću JSON formata, u kome se nalaze informacije vezane za skupove replika, koji su definisani, i deljeni su sa svim čvorovima u skupu. Samom izmenom ovog dokumenta se vrši izmena skupa replika u skladu sa očekivanom topologijom i opcijama za replikaciju. Osim toga konfiguracija se može izmeniti putem shell naredbi, naredbe kao što su *rs.add(), rs.initiate(), rs.remove()*. Konfiguracioni fajl se može dobiti izvršenjem naredbe *rs.config().* On je prikazan na slici ispod.



*Slika 13 – Izgled konfiguracije*

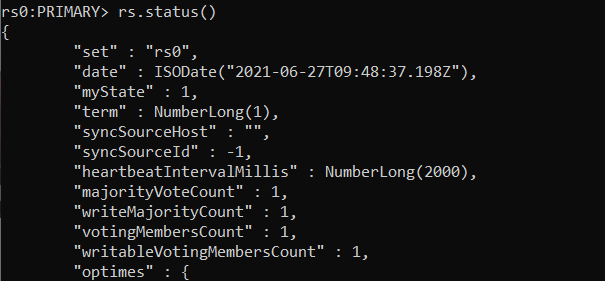
Neka od važnijih polja u konfiguraciom opcijama su:

* *\_id* – String koja predstavlja ime skupa replika, ono koje je navedeno u *.cfg* fajlu, ukoliko se oni razlikuju, doći će do poruke o greški.
* *version* – Integer vrednost, koja se inkrementira svaki put kada se konfiguracija skupa replika menja
* *members* – Definiše topologiju skupa replika, svaki element ovog niza je ugrždeni dokument, koji čuva informacije o pojedinačnom čvoru. Svaki ugnježdeni dokument sadrži sledeća polja
  + *host* – On sadrži informacije o imenu hosta, kao i o broju porta
  + *arbiterOnly, hidden* – Flegovi koji kažu da li čvor čuva u sebi same podatke ili ne, i da li je čvor skriven, odnosno da li je vidljiv aplikaciji, koristi se često kada se želi da određeni čvor radi samo određene funkcije. Ukoliko je čvor skriven, on nikada ne može postati primar.
  + *priority* – Integer vrednost, koja omogućava postavljanje hijerarhije među čvorovima, prioritei mogu biti vrednosti od 0 do 1000, vrednosti sa višim prioritetima su češće izabrani kao primari u slučaju izbora. Promena prioriteta čvora izaziva izbore, jer se na nju gleda kao na promenu topologije. Ukoliko se prioritet čvora podesi na 0, naj čvor neće moći da postane primar. U slučaju da je čvor samo arbitar ili skirven, prioritet mora biti 0.
  + *slaveDelay* – Integer vrednost koja definiče kašnjenje replikacije u sekundama. Difoltna vrednost je 0. Ukoliko se ova opcija podesi, ona dovodi do toga da je čvor sigurno skriven, kao i da mu je prioritet 0.
  + *\_id* – Jedinstveni identifikator čvora u nizu čvorova, integer vrednost. Njegova vrednost se ne može izmeniti [6].

## Komande za replikaciju

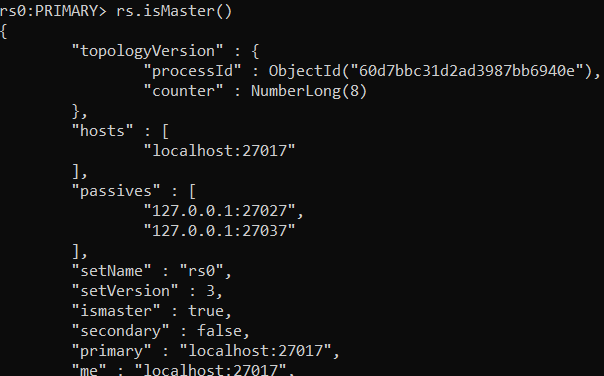
Komande za replikaciju omogućavaju dobijanje više informacija o skupu replika. Ovo su neke od naredbi koje mogu biti korišćene.

* *rs.status()* – Služi za dobijanje informacija o stanju čvorova u skupu replika, informacije koje dobija su bazirane na heartbeatovima. Za svaki čvor se dobija informacija o stanju čvora, ulozi, uptime (vremenu otkada ovaj čvor radi), optime (vreme kada je ovaj čvor poslednji put primenio operacije iz svog oplog-a). Na sledećoj slici je prikazan rezulat izvršenja ove operacije.



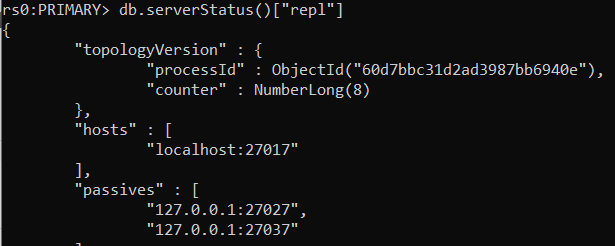
*Slika 14 – Rezultat izvršenja status komande*

* *rs.isMaster()* – Opisuje ulogu čvora na kome je izvršena ova naredba, kao i informacije o samom skupu replika. Polja *ismaster* i *secondary* definišu da li je čvor primar ili sekundarni čvor. Takođe daje ime primarnog čvora bez obzira na to da li se naredba izvršava na primarnom ili sekundarnom čvoru.



*Slika 15- Rezultat izvršenaj isMaster komande*

* *db.serverStatus()[‘repl’]* – Predstavlja deo izlaza prilikom izvršenja db.serverStatus() naredbe, jako slično izlazu iz funkcije rs.isMaster(). Najbitnija razlika između dve funkcije je postojanje polja rbid koje broji broj rollback-ova koji su se izvršili na ovom čvoru. Na sledećoj slici je prikazan izlaz prilikom izvršenja ove funkcije [7].

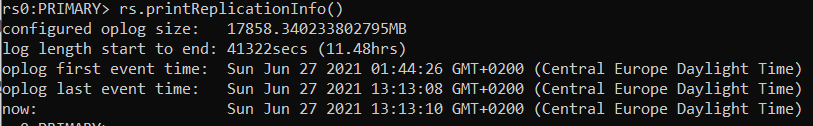




*Slika 16 – Rezultat izvršenja db.serverStatus()[‘repl’] naredbe*

## Oplog fajl

Oplog.rs fajl je centralni deo mehanizma za replikaciju. On prati sve naredbe koje su replicirane u skupu replika. Svaka informacija i operacija koja treba biti replicirana će biti smeštena u ovu kolekciju. Ona je *capped* kolekcija, što znači da ima svoj limit u pogledu veličine. Po difoltu oplog.rs kolekcija zauzima oko 5% slobodnog diska. Veličina oplog fajla definiše prozor replikacije, odnosno vreme koje je neophodno da se ispuni oplog fajl. Informacije o oplog polju se mogu dobiti izvršavanjem naredbe *printReplicationInfo.* Rezultat izvršenja ove naredbe je prikazan na slici ispod.

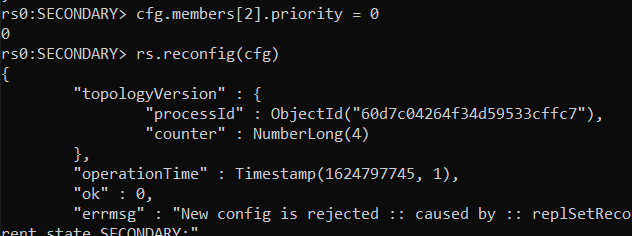


*Slika 17 – Informacije o oplog fajlu*

Može se primetiti da osim veličine oplog fajla, daje informaciju za koliko vremena će na osnovu trenutnog opterećenja, vremena biti neophodno da se overwrite-uju upisi u oplog fajl. Naravno treba imati u vidu da ovo nisu realistične predikcije jer nije previše stvari rađeno nad skupom replika. Nakon iniciranja čvora i dodavanja čvora u skup replika se ovaj fajl kreira. Takođe veličina oplog fajla se može definisati u okviru sekije za replikaciju u okviru konfiguracionog fajla. Sve izvršene operacije se upisuju u oplog fajl, dok on ne dostigne svoj limit. Nakon dostizanja limita prve izvršene operacije se prepisuju sa novim operacijama. Prozor replikacije je bitan jer definiše koliko vremena čvor u skupu replika može da ne radi, bez ljudske pomoći kako bi se sam vratio u normalan rad. Sekundarni čvorovi repliciraju podatke iz oploga primara, i primenjuju ih na svoj oplog. Ukoliko neki čvor prestane da radi, on traži poslednju zajedničku tačku sa ostalim čvorovima, i zatim uzima sve operacije iz oplog fajla od tog trenutka, i primenjuje ih kod sebe. Ukoliko je čvor dugo nije uspeo da se oporavi, i ne može da pronađe zajedniču tačku, jer je ona prepisana novim operacijama, taj čvor se stavlja u mod za oporavak [8].

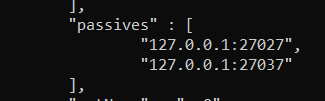
## Failover i izbori

Primarni čvor predstavlja prvu tačku komunikacije između klijenta o baze podataka. Čak i u slučaju da sekundari prestanu da rade, klijent će nastaviti da komunicira sa primarom sve dok, primar ne postane nedostupan. Čak i u idealnom sistemu, primar mora nekada biti nedostupan, jer se sa vremena na vreme mora izvršavati upgrade, samog sistema, što primar čini nedostupnim. Rolling update-ovi vrše update, prvo nad svim čvorovima pojedinačno. Izbori se izvršavaju svaki put kada dođe do promene u topologiji. Rekonfigurisanje skupa replika, uvek izaziva izbore koji mogu izabrati novi čvor kao primar. Novi primarni čvor se sigurno bira u situacijama kada je trenutni primarni čvor nedostupan, ili kada trenutni primarni čvor postane sekundarni čvor. Glasanje funkcioniše tako što određeni čvorovi predlažu sebe za primarnog čvora, samim tim glasaju za sebe, i traže podršku od ostalih čvorova. Ukoliko je skup replika definisan sa neparnim brojem članova, nema problema, jer će doći do izglasavanja. Do problema se dolazi ukoliko u skupu replika postoji paran broj članova glasača. U tom slučaju će se glasanje samo ponoviti, problem sa ponavljanjem izbora je da će aplikacija za pristup morati da čeka da se izbori završe, i da se primar izabere. Još jedan bitan aspekt prilikom izbora primara jeste prioritet, difoltni prioritet čvora je 1, i svaki čvor sa prioritetom većim od 1 može biti izabran za primara. Povećanjem vrednosti prioriteta, se povećava šansa tog čvora da postane primar. Čvor sa prioritetom 0, ne može biti primar, ali može glasati u izborima. Čvoru se prioritet može primeniti izmenom konfiguracije, što je prikazano na sledećoj slici.



*Slika 18 – Izmena prioriteta čvora*

Nakon izvršenja ove naredbe nad oba sekundarna čvora, može se videti da su ovi čvorovi postali pasivni, izvršenjem komande *isMaster()* [9]*.*



*Slika 19 – Rezultat postavljanja prioriteta na 0*

## Write concern

Kod MongoDB baze podataka, write concern je mehanizam potvrde koji programeri mogo da dodaju operacijama upisa. Viši nivoi potvrde garantuju jaču trajnost, odnosno da su upisi propagirani broju replika koji je definisan write concern-om. Što je veći broj članova skupa replika koji treba da potvrdi upis, to je veća šansa da je upis trajan u slučaju otkaza. *Majority*, odnosno većina je prosta većina, broj članova se deli sa dva i zaokružuje se rezultat deljenja. Prilikom dobjanja prednosti u trajnosti, povećava se vreme koje je neophodno da se postigne određena trajsnost, pošto odgovarajuće potvrde moraju da se sačekaju. Mogući nivoi write concern-a jesu:

* 0 – Ne čeka se ni na jednu potvrdu, upis može biti uspešan ili bezuspešan, aplikaciju ne zanima, samo se javlja greška u slučaju nemogućnosti uspešnog povezivanja sa čvorom. Jako je brza, ali nema nikakve provere.
* 1 – Ona predstavlja difoltnu vrednost, što znači da aplikacija čeka potvrdu od jednog člana replike, tačnije od primara. Ovo je osnovni parametar za uspeh operacije upisa.
* >=2 – Povećavaju broj neophodnih potvrda, u ovom slučaju se čekaju potvrde i sekundarnih članova.
* “majority” – Čekanje potvrde od većine članova skupa replika.

U slučaju korišćenja “majority” opcije, ne mora se brinuti o promeni write concern-a, u slučaju povećanja skupa replika. Bitno je napomenuti da se ovim ne podešava da li će podaci biti replikovani od strane MongoDB baze podataka, već trajnosti izmenjenih podataka. Postoje dve dodatne opcije write concern-a, prva je *wtimeout,* ona specificira maksimalno vreme koje može da protekne, dok se operacija ne proglasi neuspešnom. Druga opcija je *j* ili *journal,* koja traži da se potvrdom smatra da svaki član skupa replike koji dobije upis, i zavede u svoje polje *journal* [10] .

## Read concern

Read concern je pandan write concern-u, kod operacija čitanja, gde programeri mogu definisati, u svojoj aplikaciji, kako da izvrše operacije čitanja u skladu sa zadatom trajnošću. Read concern, omogućava trajnost podataka, u slučaju failovera. Na isti način na koji operacije upisa definišu koliko trajna operacija upisa treba da bude, read concern definiše za operacije čitanja. Operacija čitanja vraća podatke čiji je upis potvrđen od strane članova skupa replika, čiji je broj definisan read concern-om. Dokument koji nije ispunio određeni nivo read concern-a, nije nužno izgubljen, već u trenutku čitanja, nije propagiran dovoljnom broju članova skupa replika. Postoje četiri nivoua read concern-a, i oni su navedeni:

* *local* – Vraća najsvežije moguće podatke. Bilo koji sveži podaci upisani u primar, se kvalifikuju za ovaj nivo. Ne postoje nikakve garancije da će podaci biti sigurni u slučaju failover-a. On predstavlja difoltan nivo kod operacije čitanja primara.
* *available* – Predstavlja difoltan nivo kod operacije čitanja kod sekundarnih čvorova. Ona ima posebno ponašanje kod shard-ovanih klastera. Upit vraća podatke sa instance, bez garancija, da je podatak upisan na većinu replika (podaci mogu biti roll-back-ovani)
* *majority* – Vraća podatke koji su potvrđeni, kao upisani od strane većine članova skupa replika. Jedini način da su dokumenti izgubljeni prilikom čitanja, korišćenjem ove opcije, je u slučaju da je većina čvorova, palo, što je malo verovatno. Ona nudi znatno veću garanciju u odnosu na prethodno navedene. Ali razlika je da možda neće uvek biti vraćeni najsvežiji podaci iz klastera.
* *linearizable* – Upit vraća podatke koji prikazuje sve upise koji su potvrđeni od većine, a izvršeni pre početka starta operacije čitanja.

U zavisnosti od aplikacije treba se izabrati traženi nivo read concern-a. Ukoliko se žele najsigurniji podaci, onda se treba izabrati *majority* read concern, a ukoliko se žele najsveži podaci, onda je *local* bolji izbor [11].

## Read preference

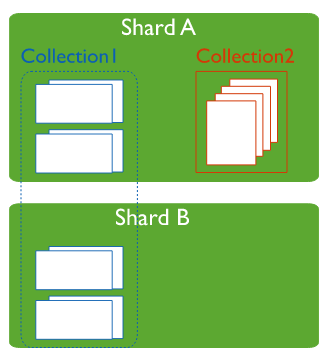
Read preference omogućava usmeravanje operacija čitanja određenim članovima skupa replika. Ovo je prevashodno podešavanje drajvera. Po difoltu, aplikacije vrše upis i čitanje podataka sa primara. Sa skupovima replika, podaci su replicirani svim članovima kojima je dozvoljeno smeštanje podataka. Sa read preference-om se može rutirati operacija čitanja, drugim članovima skupa replika, umesto primaru. Postoji pet podržanih modova read preference-a, i oni su navedeni u nastavku.

* *primary* – Difoltni mod. Sve operacije čitaju sa primara.
* *primaryPreffered* – Rutira operacije čitanja primaru, u slučaju nedostupnosti primara, ukoliko je došlo do izbora ili failover-a, aplikacija može rutirati zahteve za čitanje dostupnim sekundarnim čvorovima.
* *secondary* – Sve operacije čitaju sa sekundarnih članova.
* *secondaryPreffered* – Sve operacije čitaju se šalju sekundarnim članovima, u slučaju nedostupnosti sekundarnih članova, zahtev se prosleđuje primaru.
* *nearest* – Operacije čitanja se prosleđuju skupu replika sa najmanjim mrežnim kašnjenjem, bez obzira na to da li je primar ili sekundarni čvor.

U slučaju čitanja sa sekundara, treba imati na umu, da u zavisnosti od kašnjenja replikacije, se mogu dobiti zastareli podaci. Koliko su podaci zastareli, u potpunosti zavisi od kašnjenja između primara i sekundarnih čvorova. Zastarela čitanja se mnogo češće pojavljuju kod geografski distribuiranih skupova replika [12].

# Shard-ovani klasteri

Sa povećanjem volumena podataka, mašine ne mogu da obrade zahteve klijenata na odgovarajući način. Prvo rešenj ovog problema jeste poboljšavanje mašina, u vidu dodavanja RAM memorije, prostora na disku, ili moćniji CPU. Ove metode predstavljaju vertikalno skaliranje. Ova metoda jako brzo, shodno povećanju obima podataka može postati jako skupa. Kod MongoDB baze podataka, podacu se skaliraju horizontalno, što znači da umesto poboljšavanja pojedinačnih mašina, se dodaje više mašina, i ditribuiraju se podaci na te mašine. Shardovan klaster u MongoDB bazi podataka je kolekcija podataka koja je distribuirana na više shard-ova (servera) kako bi se postigla horizontalna skalabilnost i bolje performase pri izvršenju operacija čitanja i upisa. Shard-ovanje je jako korisno za kolekcije koje imaju veliku količinu podataka i veliki broj upita. Na slici ispod je prikazan primer shard-ovanja [13] .



*Slika 20 – Primer shard-ovanja*

Na slici se može videti da je kolekcija 1 shard-ovana, dok kolekcija 2 nije. Ukoliko se kolekcija 1 čuva na jednom serveru, može doći do povećanja korišćenja CPU-a, jer sam računar ne bi mogao da podnese opterećenje svih zahteva. Dodavanje još jednog shard-a kolekciji 1 omogućava distribuciju tereta.

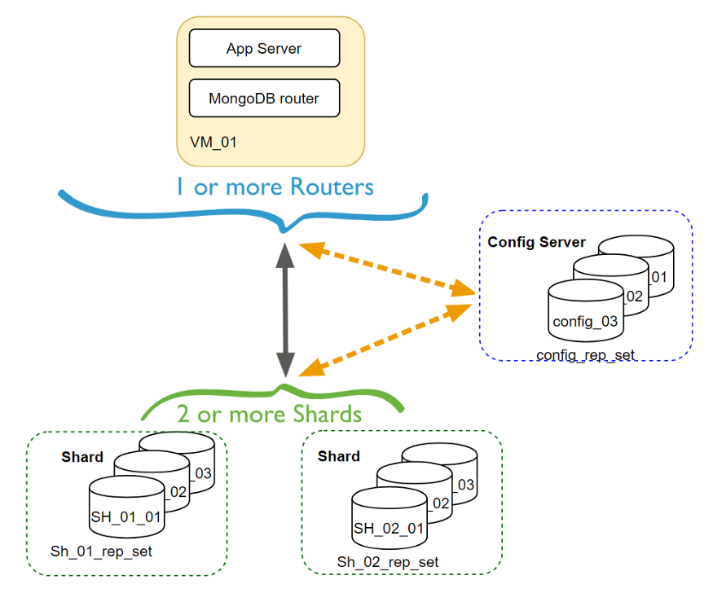
Kako bi se garantovala visoka dostupnost u shard-ovanom klasteru, svaki shard se postavlja kao skup replika. Na taj način se garantuje tolerancija na otkaz, za sve podatke, bez obzira na kom shard-u se nalaze. Sa distribuiranjem podataka na više servera, upiti postaju komplikovaniji, Baza se pretražuje, tražeći specifičan dokument. Između shard-ovanog klastera i klijenata se uspostavlja proces rutiranja, koji prihvata zahteve od strane klijenata, i odlučuje koji shard treba obraditi taj upit. Proces rutiranja se zove *mongos*. Klijenti se povezuju na mongos umesto povezivanja na svaki shard pojedinačno [14] .

## Arhitektura shard-ovanog klastera

Najbitniji aspekt shard-ovanog klastera je mogućnost dodavanja proizvoljnog broja shard-ova. Za rutiranje klijentskih zahteva se koristi proces mongos, koji rutira upite, tačnim shard-ovima. Mongos mora da zna na koji načins u podaci distribuirani, kako bi znao gde da preusmeri upit. Oni koriste metadata podatke iz kolekcija koje su shard-ovane, kako bi tačno preusmerili upite. Podaci nisu smešteni na mongos-u, već na konfiguracionim serverima koji konstantno prate gde se nalazi svaki podatak u klasteru. To je posebno bitno jer informacija koju shard sadrži može da se menja sa vremenom. Mongos vrši upit nad konfiguracionim serverima često, u slučajevima kada se podatak pomeri. Podaci se pomeraju, kako bi se osigurala podjednaka distribucija podataka na svakom serveru. Posao konfiguracionog servera je da odluči koji podaci moraju da se pomere kako bi distribucija bila ujednačenija. Kod klastera koji je shard-ovan javlja se i pojam primarnog shard-a. Svaka baza ima dodeljen primarni shard, i sve kolekcije koje nisu shard-ovane će biti dodeljene primarnom shard-u. Ne moraju sve kolekcije u okviru shard-ovanog klastera da budu distribuirane. Konfiguracioni serveri će dodeliti primarni shard svakoj bazi podataka, nakon kriranja [15].

## Podešavanje shard-ovanog klastera

U nastavku će biti prikazano podešavanje MongoDB klastera čija je struktura prikazana na slici ispod.



*Slika 21 – Arhitektura primera shard-ovanog klastera*

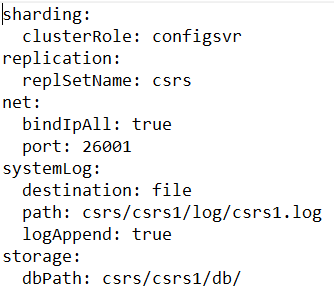
Koraci za kreiranje arhitekture koja je prikazana na slici iznad jesu:

* Podešavanje config servera skupa replika
* Shard-ovi 1 i 2 kao skup replika
* Mongos sa konfiguracijom koja pokazuje na konfiguracioni server
* Dodavanje Shard 01 u klaster kroz ruter

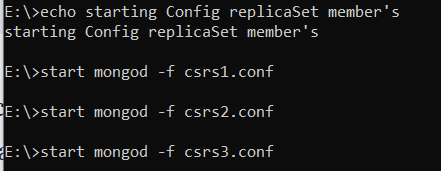
Koraci za podešavanje config servera obuhvataju, pravljenje log i db fajlova za svaki konfiguracioni server, kreiranje odgovarajućih konfiguracionih fajlova, inicijaciju svakog mongod procesa i povezivajući sa odgovarajućim konfiguracionim folderom. Zatim inicijacija klijenta skupa replika i dodavanje konfiguracionih servera u skup replika. U narednim slikama su prikazani ti koraci.



*Slika 22 – Kreiranje neophodnih direktorijuma*



*Slika 23 – Izgled konfiguracionog fajla*



*Slika 24 – Pokretanje mongod procesa*



*Slika 25 – Pokretanje klijenta*



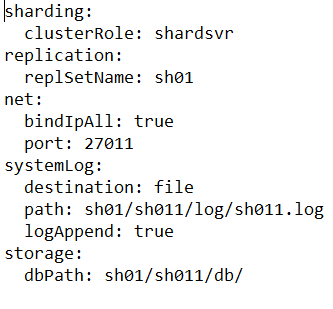


*Slika 26 – Dodavanje konfiguracionih servera*

Nakon podešavanja konfiguracionih servera, neophodno je podesiti dva shard-a, od kojih je svaki skup replika od 3 čvora. Prvo se definiše konfiguracioni fajl za svaki član klastera, zatim se konfigurišu da budu članovi istog skupa replika. Nakon toga se korak ponovi još jednom za drugi skup replika, odnosno drugi shard. Na sledećim slikama su prikazani koraci.



*Slika 27 – Kreiranje neophodnih direktorijuma*



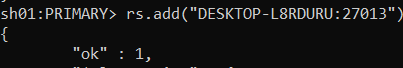
*Slika 28 – Izgled konfiguracionog fajla*

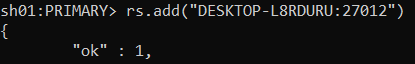


*Slika 29 – Pokretanje klijenta skupa replika*



*Slika 30 – Iniciranje skupa replika*



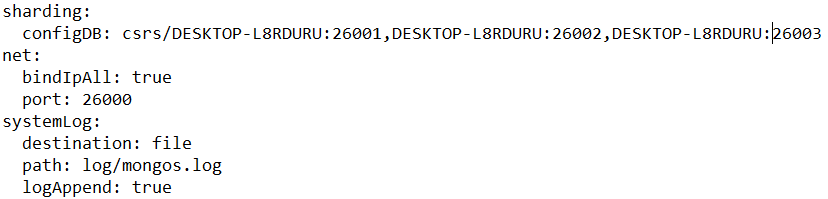


*Slika 31 – Dodavanje članova u skup replika*

Zatim sledi konfiguracija rutera, za ruter je neophodan samo log folder, pri čemu u konfiguracionom fajlu se mora ukazivati na configDB. Na sledećim slikama je prikazana konfiguracija rutera.



*Slika 32 – Kreiranje log foldera*

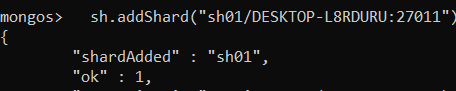


*Slika 33 – Konfiguracioni fajl za mongos*

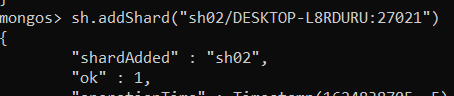
Poslednji korak predstavlja dodavanje shard-ova ruteru, odnosno mongos-u. Čije je dejstvo prikazano na sledećim slikama. Vrši se povezivanje na mongos klijenta, i zatim dodavanje shard-ova pojedinačno [16].



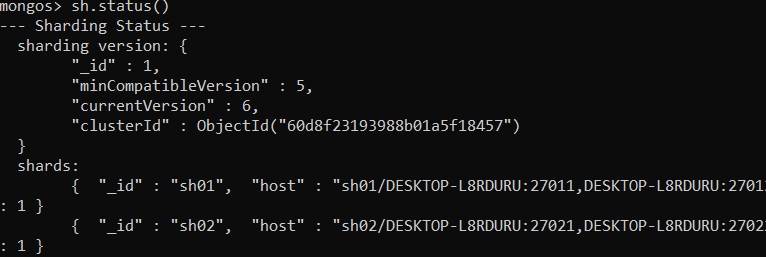
*Slika 34 – Povezivanje na mongos klijenta*



*Slika 35 – Dodavanje prvog shard-a*



*Slika 36 – Dodavanje drugog shard-a*



*Slika 37 – Provera statusa*

## Balansiranje

Ključ za shardovanje treba da obezbedi jednaku distribuciju podataka u okviru klastera. MongoDB baza podataka deli shard-ovane kolekcije u chnk-ove podataka. Prilikom dodavanja podataka u kolekciju, broj chunck-ova raste. MongoDB balanser identifikuje koji shard-ovi imaju previše chunk-ova i automatski pomeraju chunk-ove kroz shard-ove u shard-ovanom klasteru, kako bi se došlo do jednake distribucije podataka. Trenutno balanser radi nad primarnim članom konfiguracionog servera skupa podataka. Ukoliko balanser uoči nebalansiranost, započinje se runda balansiranja. Balanser može da migrira delove paralelno. Jedan shard može maksimalno da učestvuje u jednoj migraciji odjednom. Balansiranje ima uticaja i na performase, balanser ima ugrađeno ponašanje, kako bi smanjio ometanje rada baze. Balanser se može prekinuti u bilo kom trenutku. Ukoliko je balanser u tom trenutku u sred runde balansiranja, balanser prestaje tek nakon što se izvrši balansiranje [17] .

# Zaključak

Kada se govori o savremenim bazama podataka, ne može se govoriti o single server rešenjima. MongoDB je kao baza podataka, kreiran kako bi se bolje nosila baza sa velikom količinom nestruktuiranih ili polustruktuiranih podataka, koji se javljaju kao posledica pojave i upotrebe Interneta. Upravo Internet omogućava povezivanje velikog broja ljudi, i komunikaciju sa svih delova planete, što povećava obim i količinu informacija koje se čuvaju. Kao rešenje za te obime podataka javljaju se klasteri, u MongoDB bazi podataka, pojam klaster se vezuje za skupove replika, kao i za shard-ovane klastere. Jedan vrši replikaciju svih podataka na čvorove koji su u skupu replika, kako se u slučaju failover-a, podaci ne bi izgubili, ili kako bi se smanjila latencija, posebnim podešavanjima. Kod shard-ovanih klastera, neophodno je izdeliti same podatke na više grupa (chunk-ova), po nekom ključu za deljenje podataka po grupama. MongoDB baza podataka, nudi poprilično lak API za manipulaciju klasterima, odnosno nad skupovima replika i shard-ova. Takođe poprilično je fleksibilna, i pruža dosta mogućnosti i kreativnosti u upravljanju topologijom klastera. Predstavlja odličnu bazu za usvajanje osnova o klasterizaciji kod NoSQL baza podataka.

# Literatura

1. Liberios Vokorokos, Matúš Uchnár, Ľubor Leščišin Performance optimization of applications based on non-relational databases - 2016 International Conference on Emerging eLearning Technologies and Applications (ICETA), ISBN: 978-1-5090-4701-7
2. What is MongoDB? Introduction, Architecture, Features & Example - <https://www.guru99.com/what-is-mongodb.html> (Pristup 20.06.2021)
3. What is Database Clustering – Introduction and brief explanation - [https://www.ndimensionz.com/2018/01/05/what-is-database-clustering-introduction-and-brief-explanation/](https://www.ndimensionz.com/2018/01/05/what-is-database-clustering-introduction-and-brief-explanation/%20) (Pristup 20.06.2021)
4. What is Replication? - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9b3d3297182d4140a45eaa/lecture> (Pristup 20.06.2021)
5. Create a MongoDB replica set in Windows - <https://adelachao.medium.com/create-a-mongodb-replica-set-in-windows-edeab1c85894> (Pristup 26.06.2021)
6. Replication Configuration Document - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9b602997182d4140a45ec1/lecture> (Pristup 26.06.2021)
7. Replication Commands - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9d722f0680dce16b2e6fe4/lecture> (Pristup 26.06.2021)
8. Local DB: Part 1 - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9b4b8697182d4140a45eb7/lecture> (Pristup 26.06.2021)
9. Failover and Elections - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5aa1862f9c30af6b8acce589/lecture> (Pristup 26.06.2021)
10. Write Concerns: Part 1 - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9d43d41cc15446c9f380d8/lecture> (Pristup 26.06.2021)
11. Read Concerns - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9d45c81cc15446c9f380de/lecture> (Pristup 26.06.2021)
12. Read Preferences - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_2_Replication/lesson/5a9d49541cc15446c9f380e2/lecture> (Pristup 26.06.2021)
13. What is Sharding? -https://university.mongodb.com/mercury/M103/2021\_June\_15/chapter/Chapter\_3\_Sharding/lesson/5aa312e396f30f818591a594/lecture (Pristup 26.06.2021)
14. When to Shard -https://university.mongodb.com/mercury/M103/2021\_June\_15/chapter/Chapter\_3\_Sharding/lesson/5aa3184f96f30f818591a598/lecture (Pristup 26.06.2021)
15. Sharding Architecture - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_3_Sharding/lesson/5aa31bc996f30f818591a59c/lecture> (Pristup 26.06.2021)
16. Mongodb: Set-up sharded cluster for windows with just a double click. - <https://medium.com/geekculture/set-up-mongodb-sharded-cluster-for-windows-with-just-a-double-click-6eedbb7b79e> (Pristup 26.06.2021)
17. Balancing - <https://university.mongodb.com/mercury/M103/2021_June_15/chapter/Chapter_3_Sharding/lesson/5aa34efcc76aaf0ebdea9a95/lecture> (Pristup 26.06.2021)

1. Operacija koja se može primeniti više puta pri čemu će rezultat uvek biti isti. [↑](#footnote-ref-1)