UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Backup/restore u MongoDB bazi podataka

Master akademske studije
Seminarski rad iz predmeta Sistemi za upravljanje bazama podataka

Mentor:
Prof. dr Aleksandar Stanimirović

Student:
Sanja Milenković 1549

Niš, septembar 2024. godine

Sadržaj

1. Uvod....3
2. Backup strategije....4
   2.1 Full backup....4
   2.2 Inkrementalni backup....5
   2.3 Diferencijalni backup....5
   2.4 Poređenje različitih strategija backup-ovanja....6
3. Backup u kontekstu MongoDB-ja....7
   3.1 Backup servera....7
      3.1.1 Fajlsistem snapshot....7
      3.1.2 Kopiranje fajlova podataka....8
      3.1.3 Backup sa mongodump-om....8
      3.1.4 Backup sa Atlas-om....9
      3.1.5 Backup sa MongoDB menadžerom u oblaku ili Ops Menadžerom....9
   3.2 Backup seta replika....10
   3.3 Backup izlomljenog (sharded) klastera....11
      3.3.1 Backup i restore celog klastera....11
      3.3.2 Backup i restore jedinstvenog shard-a....11
4. Oporavak seta replika....12
5. Backup i restore sharded (izlomljenog) klastera....14
   5.1 Backup sa snimkom fajl sistema....14
   5.2 Backup sa dump-om baze podataka....14
   5.3 Zakazivanje backup-ova....15
6. Zaključak....17
7. Literatura....18

1. Uvod

U svetu savremenih informacionih tehnologija, baze podataka igraju ključnu ulogu u skladištenju i upravljanju velikim količinama podataka. Pouzdanost i sigurnost podataka postaju od suštinskog značaja, posebno u sistemima koji su kritični za poslovanje. Upravo zbog ovih razloga koncepti kao što su backup i restore uvedeni su kao standardne prakse u upravljanju bazama podataka. Ovi procesi omogućavaju čuvanje rezervnih kopija podataka i njihovo vraćanje u slučaju gubitka ili oštećenja, čime se obezbeđuje kontinuitet poslovanja i minimalizuje mogućnost gubitka podataka.

Kod MongoDB baza podataka, backup i restore imaju poseban značaj zbog prirode ove baze koja je dizajnirana za rad sa distribuiranim sistemima i velikim skupovima podataka. MongoDB podržava horizontalno skaliranje, što omogućava lako širenje sistema, ali i donosi nove izazove u pogledu sigurnosti i dostupnosti podataka. Backup procesi omogućavaju pravovremeno čuvanje podataka na više tačaka, dok restore procesi omogućavaju brzo oporavljanje sistema u slučaju kvarova ili grešaka.

Prednosti implementacije backup i restore procedura su višestruke. Osim očuvanja integriteta i sigurnosti podataka, ovi procesi smanjuju vreme potrebno za oporavak sistema, omogućavaju pravljenje istorijskih pregleda podataka i pomažu u smanjenju rizika od katastrofalnih gubitaka. U kontekstu MongoDB-a, fleksibilnost i skalabilnost ovih procesa omogućavaju primenu u različitim poslovnim okruženjima, od malih aplikacija do velikih enterprise sistema, čineći ih neophodnim alatom za svakog administratora baze podataka.

2. Backup strategije

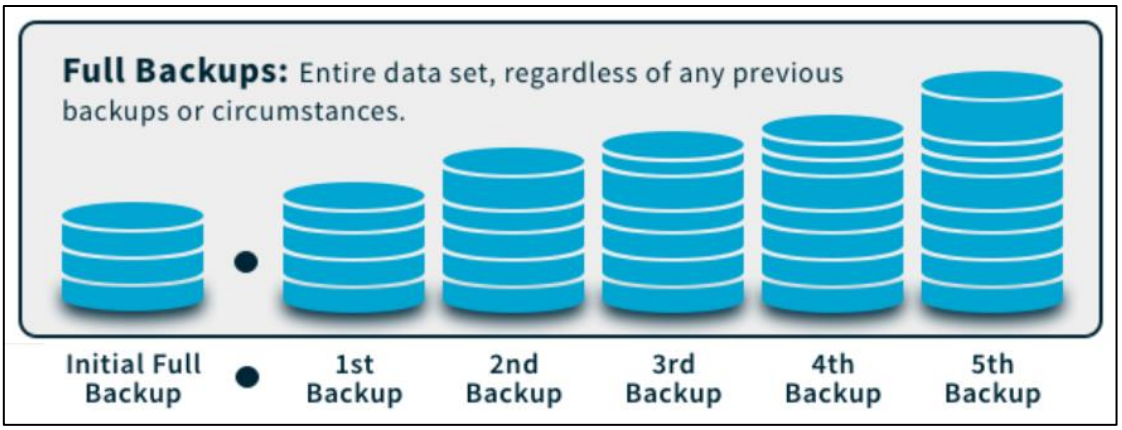
Tri su glavna tipa backup strategija koji se koriste za kreiranje rezervnih kopija digitalnih resursa:

- potpuni (full) backup
- inkrementalni backup
- diferencijalni backup

Iako je backup neophodna stavka u pružanju zaštite podataka, okolnosti se mogu razlikovati među različitim organizacijama. Stoga, odabir adekvatne backup strategije zahteva taktički pristup – potrebno je odabrati rešenje koje će u datom trenutku organizaciji pružiti najveći mogući nivo zaštićenosti podataka, a neće previše opterećivati samu mrežu. U nastavku će biti dat kratak pregled najpopularnijih backup strategija, uz osvrtanje na osnovne prednosti i mane svake.

2.1 Full backup

Full backup odnosi se na proces kreiranja jedne ili više kopija svih fajlova jedne organizacije u jednoj jedinstvenoj backup operaciji, a sa ciljem njihove zaštite. Pre početka samog procesa backupovanja, specijalista zaštite podataka, najčešće backup administrator, definiše koji će opseg fajlova učestvovati u ovom backup-u. Suštinski, kada govorimo o full backup-u, svi podaci su backup-ovani i premešteni na zasebno skladište. Iako full backup pruža dobru zaštitu od gubitka podataka, međutim vreme i troškovi potrebni da se obavi kopiranje svih podataka često ovaj tip backup-a čine nepoželjnom opcijom. Dakle, svaki put kada se odradi backup baze podataka, kreira se i čuva nova kopija cele baze podataka.



Slika 1. Full backup

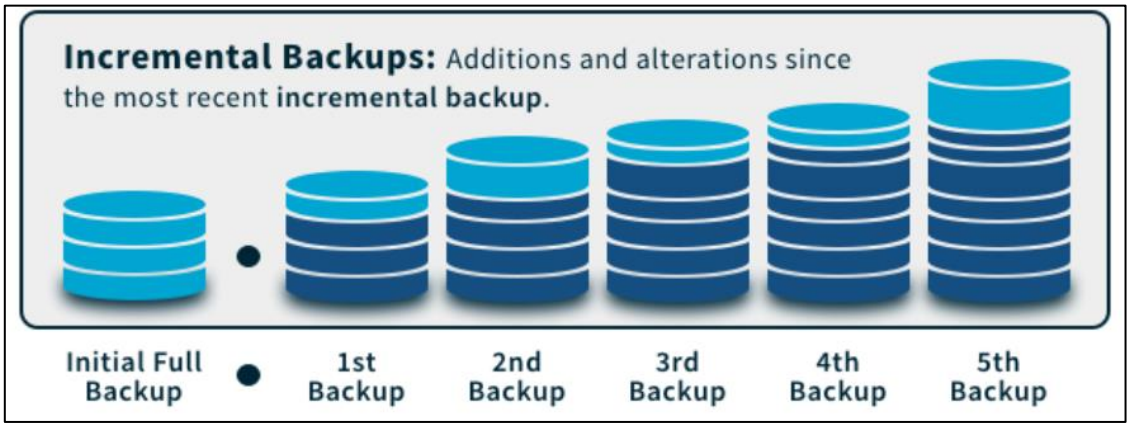
Prednosti ovakvog tipa backup-a su evidentni – celokupan sistem, svi njegovi fajlovi su u potpunosti zaštićeni i nema rizika od potencijalnog gubitka podataka. Međutim, kao što je već naglašeno, potpuni (full) backup zauzima veliki skladišni prostor i potrebno je mnogo vremena za njegovo kreiranje, budući da može da se desi da full backup potraje nekoliko puta duže od ostalih tipova backup-ovanja. Takođe, ono što može predstavljati rizik, jeste upravo činjenica da se svi podaci nalaze na jednom mestu, u okviru jednog backup-a. Bilo kakvo kompromitujuće dešavanje, gubitak, krađa ili nelegalni pristup ovim podacima, daje pristup

apsolutno svim podacima organizacije, te je potrebno voditi računa i o enkripciji podataka, čime se uvodi dodatni nivo kompleksnosti i troškova.

Preporuka je da full backup koriste male organizacije, koje generišu malu količinu podataka, budući da ovakav način kreiranja rezervnih kopija neće zauzimati veliki deo skladišnog prostora i neće oduzimati previše vremena za kreiranje rezervne kopije.

2.2 Inkrementalni backup

Inkrementalni backup podrazumeva backup-ovanje svih datoteka, foldera, podataka i hard diskova, na kojima se desila neka promena od prethodne backup aktivnosti, te se može desiti da govorimo o full backup-u ili o poslednjem inkrementalnom backup-u u nizu. Dakle, samo se nedavne izmene (tzv. inkrementi) backup-uju, čime se zauzima manji procenat skladišnog prostora i postiže veća brzina kreiranje rezervne kopije. Međutim, vreme oporavka je duže jer će biti potrebno pristupiti više backup datoteka, kako bi se postigao oporavak čitavog sistema.



Slika 2. Inkrementalni backup

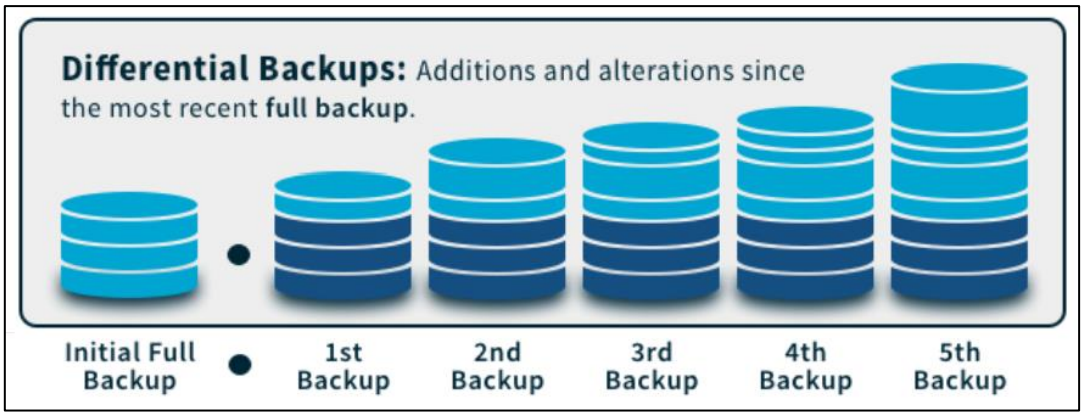
Kao osnovna prednost ovakvog pristupa navodi se efikasno zauzeće skladišnog prostora, budući da nema bespotrebnog backup-ovanja podataka koji se nisu promenili između dve sesije backup-a. Takođe, iz istog razloga, ističe se i velika brzina kojom se backup može odraditi, kao i mogućnost kreiranja backup-a u bilo kom trenutku i koliko god često da je neophodno. Sa druge strane, kao osnovni nedostatak navodi se oporavak koji traje jako dugo, budući da je finalnu verziju sistema potrebno složiti iz različitih backup datoteka. Takođe, moguće je ostvariti pun backup samo ako su svi pojedinačni backup fajlovi neoštećeni.

Ovakvu backup strategiju uglavnom biraju organizacije koje se suočavaju sa velikom količinom podataka, jer inkrementalni backup-ovi zauzimaju manje skladišnog prostora i relativno se brzo sprovode.

2.3 Diferencijalni backup

Diferencijalni backup se nalazi negde između full backup-a i inkrementalnog backup-a. Ova strategija prilikom kreiranja backup-a će uključiti samo datoteke, foldere i hard diskove koji su kreirani ili modifikovani od poslednjeg full backup-a (za razliku od promena koje su nastale od poslednjeg inkrementalnog backup-a). Samo male količine podataka su backup-

ovane između trenutnog i prethodnog backup-a, tako da se zauzima manja količina skladišnog prostora i potrebno je manje vremena. Ovaj tip backup-a često se naziva i **kumulativnim inkrementalnim backup-om**.



Slika 3. Diferencijalni backup

Neke od prednosti koje se vezuju za diferencijalni backup jesu to što je zauzeće skladišnog prostora manje nego kada govorimo o full backup-ovima, veme oporavka je brže nego kod inkrementalnih backup-ova, a sam proces kreiranja rezervne kopije je daleko brži nego kod full backup-a. Sa druge strane, postoji šansa neuspešnog oporavka ukoliko bilo koji od backup setova nije kompletan. U poređenju sa inkrementalnim backup-om, sam proces traje duže i zauzima mnogo veći prostor, dok u poređenju sa full backup-om, oporavak traje duže i znatno je kompleksniji.

2.4 Poređenje različitih strategija backup-ovanja

U narednoj tabeli, biće dato poređenje osnovnih aspekata koji mogu uticati na odabir adekvatne backup strategije, poput veličine skladišnog prostora, brzine kojom se sam backup može obaviti, brzine oporavka, podataka koji su neophodni i količine dupliranih datoteka.

	Full	**Diferencijalni**	**Inkrementalni**
Skladišni prostor	veliki	srednji do veliki	mali
Brzina backup-a	najsporiji	brz	najbrži
Brzina oporavka	najbrži	brz	najsporiji
Neophodo za oporavak	najskoriji backup	najskoriji full backup i najskoriji diferencijalni backup	najskoriji full backup i svi inkrementalni backup-ovi od full-a
Dupliranje	mnogo duplikata fajlova	ima duplikata	nema duplikata

3. Backup u kontekstu MongoDB-ja

MongoDB je dokument-orijentisana baza podataka, karakterišu je visoka fleksibilnost i skalabilnost. Veoma grubom analogijom, jedan dokument u MongoDB bazi podataka može se smatrati jednim redom u relacionoj bazi podataka, ali sa daleko fleksibilnijim modelom. Naime, ugnježdavanje dokumenata omogućava da se u bazi podataka predstave kompleksne hijerarhijske veze. Nema predefinisanih shema podataka, te ključevi i vrednosti u samim dokumentima nemaju fiksirane tipove ili veličine.

Dokumenti se dalje mogu organizovati u kolekcije, koje su ekvivalent tabelama u relacionim bazama podataka. Kolekcije odlikuju dinamičke sheme, što znači da dokumenti u okviru jedne kolekcije mogu imate apsolutnu različitu strukturu, iako se to smatra lošom praksom.

3.1 Backup servera

Postoji mnoštvo načina kojima se backup može postići i nezavisno od konkretne metode, kreiranje rezervne kopije može predstavljati izvestan napor sistema, budući da je potrebno čitanje svih podataka iz baze podataka u memoriju. Stoga, govoreći u kontekstu MongoDB baze podataka, backup-ove bi trebalo raditi nad sekundarnim serverima u setovima replika ili ako govorimo o jedinstvenom, standalone serveru, backup treba raditi kada je server neopterećen. U nastavku biće dat pregled tehnika koje mogu biti primenjene na bilo koji **mongod**, nezavisno od toga da li se radi o standalone serveru ili članu seta replika.

3.1.1 Fajlsistem snapshot

Najjednostavniji način da se kreira backup MongoDB baze podataka jeste da se kreira **fajlsistem snapshot.** (snimak sistema osnovnih datoteka) Međutim, da bi se ovaj tip backup-a postigao, potrebno je da sam fajlsistem podržava snapshotting i **mongod** mora biti pokrenut sa aktiviranom journaling opcijom. Ovaj metod, za slučaj da su prethodna dva uslova ispunjena ne zahteva dodatnu pripremu – moguće je jednostavno kreirati snimak sistema datoteka u bilo kom trenutku. Međutim, kako bismo bili sigurni da je snimak sistema validan, journaling opcija mora biti aktivirina i sam „dnevnik“ (eng. *journal*) se mora nalaziti na istom logičkom skladištu kao i drugi MongoDB podaci. Ukoliko pak journaling nije uključen, ne postoji garancija da će snimak sistema biti konzistentnan ili validan.

Da bi se obavio oporavak baze podataka, pre svega treba se postarati da mongod nije pokrenut. Konkretna komanda za oporavak iz snimka zavisi od samog fajlsistema, bitno je samo ponovo pokrenuti mongod nakon potpunog oporavka. Budući da je snimak sistema kreiran dok je sistem još uvek bio „živ“, snapshot zapravo jeste ilustracija situacije u kojoj je mongod proces ubijen nakon kreiranja snimka. Stoga, na restartu, mongod će obraditi sve journal fajlove i nastaviti sa radom normalno.

3.1.2 Kopiranje fajlova podataka

Ukoliko skladišni sistem ne podržava koncept snapshota, moguće je kopirati fajlove i direktno koristeći **cp** ili **rsync** naredbe, ili slične alate. Budući da kopiranje višestrukog broja fajlova nije atomična operacija, potrebno je stopirati sve upise u **mongod** pre samog početka kopiranja, što se može postići jednostvanom komandom **fsynclock()**. Ova komanda zaključava bazu podataka za nove upise, čime se obezbeđuje da podaci u data direktorijumu imaju poslednju konzistentnu informaciju i da se ne menjaju. Nakon izvršenja ove komande, MongoDB će čuvati u redu sve novodošle upise, ali ih neće obrađivati sve dok baza podataka ne bude ponovo otključana. U suprotnom, može doći do kopiranja fajlova dok su oni u nevalidnom stanju. Ovakav tip backup-a, koji nastaje fizičkim kopiranjem podataka koji se nalaze u osnovi, ne podržava oporavak na stanje iz određenog trenutka u prošlosti.

3.1.3 Backup sa mongodump-om

Backup sa mongodump-om se smatra sporijom metodom i za backup i za restore, ali se smatra dobrim pristupom za backup zasebnih baza podataka, kolekcija, pa čak i podsetova kolekcija.

Mongodump čita podatke iz MongoDB baze podataka i kreira visoko pouzdane BSON fajlove, koje potom **mongorestore** alat može da iskoristi za populisanje MongoDB baze podataka. Mongodump i mongorestore su jednostavni i efikasni alati za backup i restore malih MongoDB konfiguracija, ali nikako nisu idealni za kreiranje backup-ova velikih sistema.

Mongodump i mongorestore oni funkcionišu nad mongod procesom koji nije stopiran i mogu direktno da manipulišu podacima. Podrazumevano ponašanje mongodump-a je da ne hvata sadržaj lokalne baze podataka. Naime, svaka mongod instanca ima svoju sopstvenu lokalnu bazu podataka koja čuva podatke koji su korišćeni u samom procesu replikacije i druge podatke vezane za samu instancu. Ova baza podataka je nevidljiva za replikaciju, te kolekcije u ovoj bazi podataka ne učestvuju u replikaciji.

Mongodump čuva samo dokumente baze podataka. Kao rezultat, backup je efikasan u pogledu potrebnog prostora za njegovo skladištenje, ali mongorestore i mongod moraju da ponovo kreiraju indekse nakon oporavka baze podataka.

Drugi klijenti mogu da nastave da modifikuju podatke, budući da mongodump hvata i output ovih operacija. Za setove replika, mongodump pruža i --oplog opciju koja omogućava uključivanje u izlazni oplog sve unose koji su se dogodili tokom mongodump operacije. Ovo dozvoljava odgovarajućoj mongorestore operaciji da ponovi sačuvani oplog. Kako bi se oparavak baze podataka obavio iz backupa kreiranog sa --oplog opcijom, potrebno je upotrebiti mongorestore sa --oplogReplay opcijom.

Da bi se obavio backup svih baza podataka, potrebno je jednostavno izvršiti komandu **mongodump**. Ukoliko su i mongodump i mongod procesi pokrenuti na istoj mašini, moguće je jednostavno specificirati port na kojem je mongod trenutno aktivan.

```
$ mongodump -p 31000
```

Mongodump će kreirati dump direktorijum u trenutnom direktorijumu, koji će sadržati dump svih fajlova baze podataka. Konkretni podaci su čuvani u .bson fajlovima, koji zapravo sadrže sve dokumente kolekcija u BSON formatu, međusobno konkateniranih. Kako bi se lakše manipulisalo ovakvim formatom podataka, može se koristiti **bsondump** alat, koji je podrazumevano uključen u MongoDB bazu podataka.

Za korišćenje mongodump-a server uopšte ne mora da bude pokrenut, potrebno je da se samo kroz opciju --dbpath specificira lokacija na kojoj su skladišteni podaci i mongodump će jednostavno koristiti te podatke kako bi kreirao backup. Ukoliko je server pokrenut, ne treba specificirati putanju do same baze podataka.

```
$ mongodump --dbpath /data/db
```

Glavni problem kod korišćenja mondodump-a za backup-ovanje je što ovakav backup nije trenutni backup, tj. može se desiti da sistem nastavi da prihvata operacije upisa, paralelno sa kreiranjem backup-a. Primera radi, neko može pokrenuti kreiranje rezervne kopije koja uzrokuje da mongodump izbaci bazu podataka A. Dok mongodump izbacuje bazu podataka B, neko drugi izbacuje A. Međutim, pošto ga je mongodump već izbacio, na ovaj način se može dobiti snimak podataka u stanju koji nikada nije postojao na originalnom serveru.

Kako bi se ovakva situacija izbegla, ako je mongod proces pokrenut sa --replSet opcijom, može se upotrebiti i mongodump --oplog opcija. Na ovaj način će se voditi evidencija o svim operacijama koje se dešavaju na serveru dok se dump osposobljava, pa se ove operacije mogu ponovo izvršiti nakon što je backup oporavljen.

Da bi se podaci oporavili iz mongodump backup-a, potrebno je iskoristiti mongorestore alat.

```
$ mongorestore -p 31000 --oplogReplay dump/
```

3.1.4 Backup sa Atlas-om

MongoDB Atlas jeste MongoDB servis u oblaku, koji nudi 2 u potpunosti kontrolisane metode za backup:

- **Cloud backup** (rezervne kopije u oblaku) – koje koriste izvornu funkciju snimka provajdera usluga u oblazu za implementaciju, kako bi ponudile robusne opcije backup-ova. Ova vrsta backup-a pruža:
  - On-demand snapshots (snimke na zahtev) – koji omogućavaju da se u datom trenutku pokrene snimak sistema
  - Kontinuirani backup – koji omogućava da se zakažu ponavljajući backup-ovi
- **Legacy backup** – koji kreira inkrementalne backup-ove podataka, ali se smatra prevaziđenim načinom kreiranja rezervnih kopija (**deprecated**)

3.1.5 Backup sa MongoDB menadžerom u oblaku ili Ops Menadžerom

MongoDB menadžer u oblaku (eng. MongoDB Cloud Manager) je hostovani MongoDB servis za backup, monitoring i automatizaciju. MongoDB menadžer u oblaku

podržava backup i oporavak MongoDB setova replika i sharded klastera kroz grafički korisnički interfejs.

MongoDB menadžer u oblaku kontinualno kreira backup-ove MongoDB konfiguracija, tako što čita podatke iz oplog-a i kreira snimke podataka u unapred definisanim vremenskim intervalima. Takođe i nudi mogućnost da se oporavak određene konfiguracije obavi iz određenog trenutka.

3.2 Backup seta replika

Generalno govoreći, backup-ovi treba da se kreiraju na sekundarnim serverima, kako bi se na taj način izbeglo opterećenje primarnog servera i kako bi se moglo odraditi zaključavanje sekundara bez uticaja na funkcionisanje samog sistema (ukoliko se zahtevi za čitanje podataka ne šalju ovim instancama). Za kreiranje backup-a seta replika, može se koristiti bilo koja od 3 prethodno navedene metode, s naglaskom da je snimak sistema fajlova ili kreiranje kopije fajlova podataka preporučeni način. Bilo koja od ovih tehnika može biti primenjena nad sekundardnim serverima setova replika, bez modifikacija.

Mongodump, iako validna metoda backup-ovanja MongoDB baze podataka, nije najjednostavnija u kontekstu sistema sa omogućenom replikacijom. Pre svega, ukoliko se backup radi preko mongodump-a, potrebno je backup-ove kreirati koristeći --oplog opciju, kako bi se dobio snapshot određenog trenutka u vremenu, u suprotnom, stanje backup-a neće se poklapati sa stanjem svih ostalih replika u klasteru.

Kako bi se set replika oporavio iz mongodump backup-a, potrebno je pokrenuti ciljanu repliku kao standalone server sa praznim direktorijumom podataka. Najpre, na standardan

```
> use local
> db.createCollection("oplog.rs", {"capped" : true, "size" : 10000000})
```

način treba pokrenuti mongorestore sa --oplogReplay opcijom. Nakon toga, iako bi ova replika trebalo da ima validnu kopiju svih podataka, još uvek nedostaju informacije iz oplog-a. Te je potrebno kreirati oplog koristeći **createCollection** komandu.

Potrebno je potom populisati sam oplog. Najlakši način da se oplog.bson fajl oživi iz dump-a jeste u **local.oplog.rs** kolekciju. Local.oplog.rs je ograničena kolekcija koja čuva oplog. Moguće je definisati njenu veličinu u MB prilikom kreiranja.

Bitno je samo napomenuti da se ne radi u dump-u oplog-a, već o operacijama koje su se desile za vreme kreiranja backup-a. Nakon što je mongorestore operacija okončana, moguće

```
$ mongorestore -d local -c oplog.rs dump/oplog.bson
```

je restartovati server kao regularni član replika seta.

3.3 Backup izlomljenog (sharded) klastera

Izlomljene klastere nije moguće „savršeno” backup-ovati dok su aktivni: nije moguće dobiti snimak celokupnog stanja klastera u određenom trenutku. Međutim, ovaj nedostatak se uglavnom zaobilazi činjenicom da, kako vaš klaster postaje veći, postaje sve manje verovatno da će biti potrebno vratiti ceo klaster iz backup-a. Stoga, kada se radi o izlomljenom klasteru, potrebno je fokusirati se na backup pojedinačnih delova: servera sa konfiguracijama i replikacionih setova pojedinačno.

Pre nego što izvršite bilo koju od ovih operacija na izlomljenom klasteru (bilo backup ili povraćaj podataka), potrebno je isključiti balanser. Nije moguće dobiti konzistentan snimak podataka dok se šardovi kreću.

3.3.1 Backup i restore celog klastera

Kada je klaster veoma mali ili se nalazi u fazi razvoja, možda će biti potrebno da se zaista izvrši backup i obnovu celog sistema. Ovo se može postići isključivanjem balasera i pokretanjem komande **mongodump**. Na taj način se kreira bekap svih šardova na mašini na kojoj se pokreće mongodump.

Kako bi se obnovili podaci iz ovog tipa backup-a, pokrenite **mongorestore**.

Nakon isključivanja balasera, alternativno se može napraviti backup fajl sistema ili direktorijuma podataka za svaki šard i konfiguracione servere. Međutim, neizbežno je da će kopije biti napravljene u nešto drugačijim trenucima, što može, ali i ne mora predstavljati problem.

3.3.2 Backup i restore jedinstvenog shard-a

Najčešće se javlja potreba da se obnovi samo jedan šard u klasteru. Jedan od načina jeste da se oporavak obavi koristeći backup tog šarda koji je napravljen jednom od metoda za pojedinačne servere koje su prethodno opisane. Međutim, postoji jedan važan problem na koji treba obratiti pažnju: polazeći od pretpostavke da je backup klastera obavljen u ponedeljak. U četvrtak dolazi do kvara diska i potrebno je obnoviti podatke iz prethodno kreiranog backup-a. Međutim, u međuvremenu, došlo je do promene podataka i ovi podaci nisu premešteni na ovaj šard, posledično, backup koji je šarda koji je kreiran u ponedeljak neće sadržati ove nove podatke. Možda ćete moći da koristite backup konfiguracionog servera kako biste utvrdili gde su ti šardovi bili u ponedeljak, ali to je znatno složenije nego jednostavno obnavljanje šarda. U većini slučajeva, obnavljanje šarda uz gubitak podataka iz tih šardova je prihvatljivija opcija.

4. Oporavak seta replika

Da bi se uspešno obavio oporavak seta replika, potrebno je proći niz koraka koji će biti dati u nastavku, a sa ciljem oporavka podataka u novi set replika.

- *Pribaviti backup MongoDB fajlova*

Ovi fajlovi mogu biti kreirani kao snapshot fajl sistema. Potrebno je obratiti posebnu pažnju ukoliko se koristi šifrovana mašina za skladištenje koja koristi neki od poznatih mehanizama šifrovanja poput AES256-GCM šifre, te u tom kontekstu razlikovati 2 moguće situacije: oporavak iz "hot" backup-a i oporavak iz "cold" backup-a. Naime, ukoliko je backup kreiran kao "hot backup", što znači da je mongod instanca još uvek pokrenuta, MongoDB može detektovati prljave ključeve na startup-u i da automatski prebaci ključ baze podataka, kako bi se izbegla ponovna inicijalizacija. Oporavak od "cold backup-a", što znači da mongod nije pokrenut, onemogućava MongoDB da uoči prljave ključeve pri pokretanju.

- *Odbacivanje lokalne baze podataka pri backup-u*

Ukoliko se oporavak pokreće iz fajlsistem backup-a ili bilo kog backup-a koji sadrži lokalnu bazu podataka, treba je odbaciti. Najpre, treba pokrenuti standalone mongod koristeći fajlove podataka sa putanje na kojoj se nalazi sam backup. Treba i aktivirati sve opcije koje su i inicijalno bile pokretane.

```
mongod --dbpath /data/db <startup options>
```

Potom je potrebno, obrisati lokalnu bazu podataka komandama koje su prikazane.

```
use local
db.dropDatabase()
```

Po završetku posla, potrebno je ponovo ugasiti standalone server.

- *Pokrenuti replika set na jedinstvenom čvoru*

Pokrenuti mongod instancu kao replika set sa jednim čvorom. Definisati putanju do backup-a podataka uz pomoć --dbpath opcije i imena seta replike sa --replSet opcijom.

- *Konektovati mongosh i mongod instancu*

Sa iste mašine na kojoj je već pokrenut mongod, pokrenuti i mongosh. Jednostavnom komandom **mongosh**, moguće je povezati se sa mongod koji osluškuje localhost na podrazumevanom portu 27017. Ukoliko to nije slučaj, pomoću opcije --port moguće je specificirati konkretan port.

- *Inicirati novi replika set*

Koristeći komandu **rs.initiate()** nad jednim članom seta replike, MongoDB inicira set koji se sastoji iz trenutnog člana i koji koristi podrazumevanu konfiguraciju seta replika.

```
rs.initiate( {
   _id : <replName>,
   members: [ { _id : 0, host : <host:port> } ]
})
```

MongoDB pruža 2 opcije za oporavak sekundarnih članova seta replika:

- manuelno kopiranje fajlova baze podataka u svaki folder podataka
- omogućavanjem inicijalne sinhronizacije, kako bi se distribucija podataka obavila automatski

Ukoliko se radi o velikoj bazi podataka, inicijalna sinhronizacija može zahtevati dosta vremena kako bi se uspešno završila, te je za ovaj tip baza podataka poželjniji pristup kopiranje fajlova baze podataka na svakom host-u.

Sprovođenjem naredna 4 koraka moguće je kreirati dodatne članove seta replika sa oporavljenim podacima koji su zapravo nastali kopiranjem MongoDB fajlova podataka.

1. Ugasiti mongod instancu koja je oporavljena pomoću **--shutdown** ili **db.shutdownServer()**
2. Kopirati podatke sa primarnog čvora na sve sekundarne čvorove, konkretno na lokaciju **dbPath** za sve ostale članove seta replika
3. Pokrenuti mongod instancu koja je oporavljena
4. Dodati sekundarne članove u set replika. U **mongosh** sesiji koja je vezana za primarni čvor, potrebno je dodati sekundare koristeći **rs.add()** metod.

5. Backup i restore sharded (izlomljenog) klastera

U nastavku će biti prikazano kako se izlomljeni klasteri mogu oporaviti pomoću mongodump i mongorestore strategije ili snimaka sistema. Bitno je napomenuti da postoje i koordinisani backup i restore procesi pomoću plaćenih MongoDB alata poput MongoDB Atlas, MongoDB Cloud Manager-a i MongoDB Ops Manager-a.

5.1 Backup sa snimkom fajl sistema

U nastavku je opisana procedura za kreiranje backup-a svih komponenti sharded klastera u MongoDB bazi podataka, koristeći snapshot-ove fajl sistema za snimanje stanja zasebnih mongod instanci. Važno je napomenuti da je za pravljenje backup-a sharded klastera neophodno zaustaviti sve operacije upisa u klaster.

Da bi se osigurao konzistentan backup pomoću snapshot-a fajl sistema, potrebno je prvo zaustaviti balanser, prekinuti sve operacije upisa, kao i sve šematske transformacije koje se odvijaju na klasteru.

Jedan od ključnih koraka u ovoj proceduri je **zaustavljanje balansera** pre nego što se pokrene proces pravljenja snapshot-a. Ako je balanser aktivan, podaci koji se migriraju između shard-ova mogu dovesti do nekompletnih ili duplikovanih podataka u backup-u. Komanda **sh.stopBalancer()** se koristi za zaustavljanje balancera, a njegov status može se proveriti pomoću **sh.getBalancerState()**. MongoDB balanser je pozadinski proces koji nadgleda količinu podataka na svakom shard-u u kolekciji. Ukoliko količina podataka na nekom od shard-ova nadmaši definisani prag migracije, balanser će automatski pokušati da migrira podatke između šardova i da uspostavi jednakost količine podataka među shard-ovima. Podrazumevano, proces balansiranja je uvek aktivan.

Kako bi se obezbedila konzistentnost backup-a, neophodno je koristiti komandu **fsync** ili metodu **db.fsyncLock()** koja zaustavlja operacije upisa u klaster, čime se smanjuje mogućnost inkonzistentnosti podataka. Ova procedura mora biti pažljivo praćena kako bi backup bio uspešan.

Potom je potrebno obaviti najpre backup primarnog config servera, kao i backup primarnih shard-ova. Po završetku svih operacija, treba ponovo otključati klaster **db.fsyncUnlock()** metodom i startovati balanser kako bi se nastavile migracije podataka u klasteru.

5.2 Backup sa dump-om baze podataka

Počevši od verzije MongoDB 7.1, moguće je kreirati backup podataka sharded klastera koristeći mongodump.

Mongodump predstavlja alat koji kreira binarni izvoz sadržaja baze podataka. Da biste obezbedili konzistentnost backup-a, pre nego što pokrenete proces, morate zaustaviti **balanser**, obustaviti sve upise, kao i sve operacije transformacije šeme.

Nešto slično postupku koji je opisan u prethodnom poglavlju, i oporavak korišćenjem mongodump alata podrazumeva prolazak kroz par koraka.

Najpre je potrebno pronaći adekvatan period kada je moguće obaviti backup. Poterbno je nadgledati aktivnost same aplikacije i baze podataka kako bi se pronašlo vreme kada je najmanja verovatnoća da će migracije delova podataka ili bilo kakve operacije transfromacije šeme moći da se obave.

Potom, potrebno je stopirati balanser. Ukoliko je operacija balansiranja trenutno aktivna, operacija stopiranja će sačekati njen završetak. Tako da je pre nastavka drugih operacija potebno utvrditi da je balanser zapravo ugašen, pomoću metode **sh.getBalancerState()**

Kao što je i ranije navedeno, sharded klaster mora ostati zaključan sve dok je backup proces u toku, kako bi se baza podataka zaštitila od upisa, koje mogu dovesti do nekonzistentnosti u backup-u.

Pomoću komande prikazane na narednoj slici, potrebno je kreirati backup sharded klastera.

```
mongodump \
   --host mongos.example.net \
   --port 27017 \
   --username user \
   --password "passwd" \
   --out /opt/backups/example-cluster-1
```

Po završetku kreiranja backup-a, potrebno je ponovo otključati klaster i ponovo startovati balanser.

5.3 Zakazivanje backup-ova

U sharded klasteru, balanser je zadužen za raspoređivanje podataka kroz klaster, tako da svaki shard ima okvirno istu ili približnu količinu podataka. Ipak, kao što je više puta naznačeno, potreno je deaktivirati balanser u vreme kreiranja backup-ova kako bismo bili sigurni da neće doći do migracije podataka koji potencijalno mogu uticati na sadržaj backup-ova.

Ukoliko je zakazan automatski backup, moguće je takođe onemogućiti sve operacije balansiranja u tom periodu, koristeći narednu komandu:

```
use config
db.settings.updateOne(
   { _id : "balancer" },
   { $set : { activeWindow : { start : "06:00", stop : "23:00" } } },
   true
)
```

Ovom komandom je balanser konfigurisan tako da radi između 06:00 i 23:00. Potrebno je dakle backup zakazati i obaviti izvan ovog vremenskog okvira.

6. Zaključak

Backup i restore jesu procesi koji igraju ključnu ulogu u obezbeđivanju kontinuiteta rada sistema koji koristi MongoDB bazu podataka. Mogućnost pravovremenog i pouzdanog pravljenja rezervnih kopija podataka pruža neophodnu sigurnost u slučajevima neplaniranih kvarova, oštećenja podataka ili napada. Metode poput snimanja stanja baze podataka (snapshot) i inkrementalnih backup-ova omogućavaju optimizovano čuvanje podataka, uz smanjenje opterećenja na resurse sistema.

Jedna od najvećih prednosti MongoDB baze podataka, jeste njena fleksibilnost u izboru metoda za backup i restore, bilo da je reč o manuelnim tehnikama ili automatizovanim alatima. Takođe, mogućnost backupovanja podataka u realnom vremenu omogućava da aplikacije nastave s radom bez prekida, čime se minimizuju negativni uticaji na korisničko iskustvo.

Ipak, ovi procesi nisu bez izazova. Njihovo pravilno upravljanje može zahtevati dodatne resurse, naročito kod velikih, distribuiranih sistema sa velikim količinama podataka. Takođe, u slučajevima kada se koristi replikacija ili razlomljeni klasteri, procesi backup-a mogu postati kompleksniji, zahtevajući pažljivo planiranje kako bi se izbegli problemi sa konzistentnošću podataka.

Backup i restore metode jesu od suštinskog značaja za održavanje pouzdanosti i dostupnosti MongoDB baza podataka i uprkos tehničkim izazovima, pravilna implementacija ovih metoda može značajno doprineti sigurnosti podataka i dugoročnoj stabilnosti sistema.

7. Literatura

[1] “Designing Data Intensive Applications”, Martin Kleppmann, O’Reilly, 2017.

[2] “MongoDB – The Definitive Guide”, Kristina Chodorow, O’Reilly, 2013.

[3] https://www.mongodb.com/docs/manual/core/backups/

[4] https://www.mongodb.com/resources/basics/backup-and-restore

[5] “Practical MongoDB - Architecting, Developing and Administering MongoDB”, Shakuntala Gupta Edward, Navin Sabharwal

[6] https://www.unitrends.com/blog/types-of-backup-full-incremental-differential

[7] “Backup and recovery”, W. Preston, O’Reilly, 2006.

[8] https://www.percona.com/blog/mongodb-backup-best-practices/