Elektronski fakultet, Univerzitet u Nišu

Katedra za Računarstvo i informatiku

SEMINARSKI RAD

Replikacija kod Elasticsearch baze podataka

Predmet: Sistemi za upravljanje bazama podataka

Student: Aleksandar Mitić

Broj indeksa: 1204

Sadržaj

[Uvod 3](#_Toc72681167)

[Arhitektura Elasticsearch baze podataka 4](#_Toc72681168)

[Šarding i skalabilnost 8](#_Toc72681169)

[Replikacija 11](#_Toc72681170)

[Način rada replikacije 11](#_Toc72681171)

[Izbor broja replika 13](#_Toc72681172)

[Slikanje podataka 13](#_Toc72681173)

[Druga uloga replikacije 14](#_Toc72681174)

[Kibana - korišćenje replikacije 15](#_Toc72681175)

[Izvršavanje upita nad replikama 17](#_Toc72681176)

[Zaključak 18](#_Toc72681177)

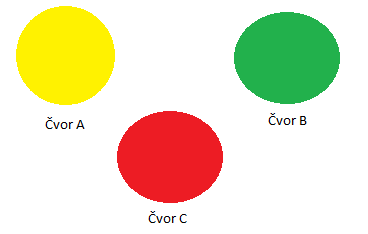
[Literatura 19](#_Toc72681178)

# Uvod

Replikacija se odnosi na skladištenje redundante kopije podataka. Počevši od verzije 7.x, Elasticsearch baza podataka kreira jednu primarnu kopiju šarda (eng. „shard“) sa faktorom replikacije postavljenim na 1. Replike se nikada ne dodeljuju istom čvoru na kojem su dodeljeni primarni shard-ovi, što znači da bi trebalo da postoje najmanje dva čvora u klasteru kako bi se kreirale replike u sistemu. Ukoliko primarni shard otkaže odnosno ode u grešku, replika automatski postaje primarni shard u sistemu. Na taj način mehanizam replikacije se bori sa otkazom hardvera ili bagovitim softverom. Pored ove funkcionalnosti replikacija ima još važnu jednu ulogu kod Elasticsearch baze podataka a to je povećanje propusnosti u sistemu. Kreiranjem novih replika u sistemu se povećava propusnost jer Elasticsearch replike tretira kao primarne šardove i nad njima se mogu pisati upiti kao i nad primarnim šardovima. Pre samog objašnjenja replikacije u radu su objašnjene osnovne komponente sistema i njihova uloga.

# Arhitektura Elasticsearch baze podataka

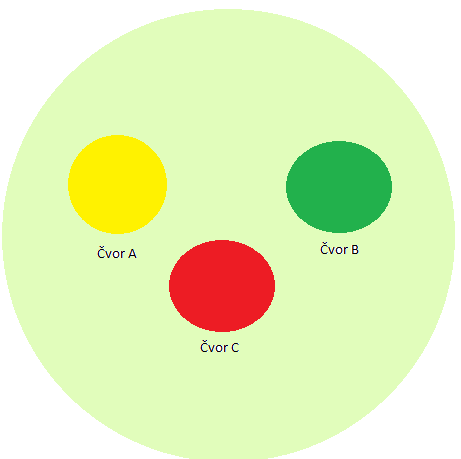
Kada se pokrene ElasticSearch baza podataka, ono što se zapravo desi jeste pokretanje čvora. Čvor je u suštini instanca Elasticsearch-a koji skladišti podatke. Ukoliko je potrebno obezbediti veliku količinu podataka, na primer za skladištenje terabajt podataka, potrebno je pokrenuti onoliko čvorova koliko je potrebno. Na slici 1, grafički su prikazani čvorovi A,B i C:



**Slika 1**: Grafički prikaz čvorova

Tada će svaki čvor predstavljati deo ukupne količine podataka. Na ovaj način, ElasticSearch omogućava skladištenje podataka na različitim virtuelnim ili fizičkim mašinama, koje omogućavaju skladištenje vise terabajt podataka, čak iako svaka mašina sadrži disk veličine nekoliko stotine gigabajta. Čvor kod Elasticsearch baze podataka se odnosi na instancu, ne mašinu, tako da je moguće pokrenuti onoliko čvorova koliko je potrebno. To znači da je moguće pokrenuti na primer pet čvorova na mašini na kojoj se inicijalno gradi sistem, bez potrebe za virtualnom mašinom ili kontejnerima. U tom slučaju potrebno je na produkcijskom sistemu, tako da se svaki čvor pokreće na namenskoj mašini, virtuelnoj mašini ili kontejneru [2].

Svaki čvor kod Elasticsearch pripada klasteru (eng. „claster”). Klaster predstavlja kolekciju povezanih čvorova koji zajedno sadrže sve podatke. Na slici 2, je prikazan klaster sa tri čvora A,B i C:



**Slika 2**: Grafički prikaz klastera za tri čvora

Moguće je kreirati onoliko klastera koliko je potrebno, ali jedan klaster je obično dovoljan. Inicijalno klasteri su međusobno nezavisni. Moguće je implementirati pretraživanje podataka preko više klastera (eng. „cross-cluster searches“) ali takvo pretraživanje obično nije pogodno. Takođe, klastere je moguće pokrenuti tako da oni nezavisno služe različitoj svrsi. Na primer, moguće je pokrenuti klaster koji vrši pretraživanje e-commerce aplikacija i drugi za analizu performansi pretraživanja. Razlog za razdvajanje različitih poslova na različite klastere, jeste jednostavno razdvajanje poslova logički i mogućnost različitog konfigurisanja klastera.

Pomenuto je kako se kreira čvor kod Elasticsearch-a, ali kako se kreira klaster? Ono šta se zapravo desi kada se pokrene čvor jeste da se automatski formira i klaster. Kada se čvor pokrene, on će se, ili pridružiti postojećem kreiranom klasteru ukoliko je konfigurisan ili će formirati klaster koji sadrži samo taj čvor. Čvor kod Elasticsearch-a mora da pripada klasteru čak iako je kreiran samo jedan čvor, i u tom slučaju, taj čvor takođe mora da pripada klasteru. Kada su u pitanju skalabilnost i dostupnost podataka može da dođe do problema ukoliko klaster sadrži samo jedan čvor, ali za razvoj sistema sasvim je dovoljno da klaster sadrži samo jedan čvor.

Svaka jedinica koja se skladišti u klasteru, kod Elasticsearch baze podataka, se naziva dokument. Dokumenti predstavljaju JSON objekte koji sadrže podatke koje je potrebno skladištiti. Za skladištenje podataka, JSON objekat kod Elasticsearch-a, pored podataka koje je potrebno sačuvati, sadrži i dodatne meta podatke (eng. „metadata“) koje Elasticsearch interno koristi. Na primer, za skladištenje osobe (eng. „person“) JSON objekat može da izgleda kao na slici 3:



**Slika 3**: Primer skladištenja osobe u JSON obliku

Kao što je moguće primetiti na slici 3, na desnoj strani, JSON objekat sadrži dva polja „name“ i „country“ koje je potrebno sačuvati. Ova dva polja se kod Elasticsearch-a skladište u okviru polja „\_source“ koji je sastavni deo dokumenta. Pored ovih polja moguće je dodati i bilo koja druga polja za skladištenje u Elasticsearch bazi podataka. Korisnik ima potpunu kontrolu nad kreiranjem polja u dokumentu Elasticsearch baze podataka.

Svaki dokument unutar Elasticsearch baze podataka se skladišti u okviru indeksa (eng. „index“). Indeksi logički grupišu dokumente i isto tako obezbeđuju konfiguracione opcije za grupisane dokumente koje su u vezi sa dostupnošću i skalabilnost dokumenata. Indeks je zapravo kolekcija dokumenata sličnih karakteristika koji su logički povezani. Na primer podatke nekih osoba je moguće skladištiti unutar indeksa „Osoba“, moguće je zatim kreirati i drugi indeks „Departman“ koji sadrži nazive departmana gde se svaki departman skladišti kao dokument. Na slici 4 dat je primer indeksa „Osoba“ i „Departman“. Indeks može da sadrži onoliko dokumentata koliko je potrebno tj. ne postoji ograničenje u skladištenju broja dokumenata unutar indeksa. Prilikom pretraživanja podataka kod Elasticsearch-a specificiraju se indeksi koje je potrebno pretraživati.



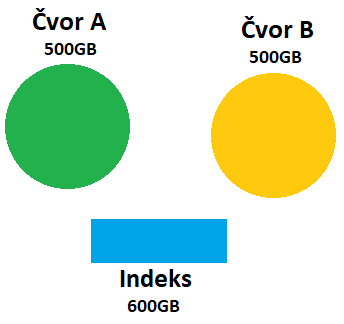
**Slika 4**: Grafički prikaz indeksa „Osoba" i „Departman"

# Šarding i skalabilnost

Da bi se razumeo način rada replikacije, potrebno je prvo shvatiti kako funkcioniše šarding (eng. „sharding“) i skalabilnost kod Elasticsearch-a. U prethodnom poglavlju, bilo je reči da čvorovi kod Elasticsearch-a igraju glavnu ulogu kada je u pitanju skalabilnost, što se tiče skladištenja podatka i prostora na disku. Nije moguće sačuvati jedan terabajt podataka ukoliko Elasticsearch sadrži samo jedan čvor od 500 gigabajta. Međutim, ukoliko se doda još jedan čvor u klasteru sa odgovarajućm količinom, tada Elasticsearch će smeštati podatke na oba čvora, što bi značilo da klaster sadrži dovoljno prostora za skladištenje podataka.

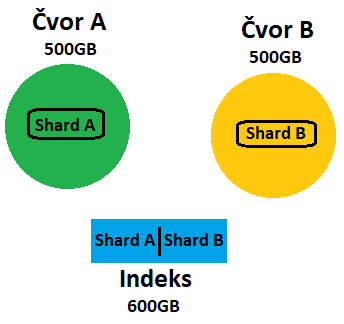
Elasticsearch distribuira podatke po čvorovima pomoću šardinga. Šarding predstavlja način deljenja Elasticsearch indeksa na manje delove, gde se svaki deo naziva šard (eng. „shard“). Šarding funkcioniše na nivou indeksa ne i na nivou klastera ili na nivou čvorova. Razlog za to jeste, što, jedan indeks kod Elasticsearch može da sadrži bilion dokumenata dok drugi indeks može da sadrži nekoliko stotina dokumenata. Naravno, ovo zavisi od načina konfigurisanja indeksa na sistemu. Glavni razlog za deljenje indeksa na šardove jeste horizontalno skaliranje podataka.

Kako šarding funkcioniše? Na primer, ukoliko postoje dva čvora istog kapaciteta recimo 500 gigabajta, i postoje ogorman indeks koji zauzima 600 gigabajta kapaciteta na disku, kao na slici 5:



**Slika 5**: Primer šardinga velikog indeksa na čvorove A i B

Na slici 5, se vidi da se indeks ne može smestiti ni na jednom čvoru kompletno jer premašuje njihove kapacitete. Dakle, smeštanje indeksa na jednom šardu nije opcija, zbog toga što se šard smešta na jednom čvoru. Umesto toga moguće je podeliti indeks na dva šarda gde će svaki čuvati po 300 gigabajta, kao na slici 6:



**Slika 6**: Deljenje indeksa na šard A i šard B

Naravno, indeks je moguće podeliti i na više šardova, na primer na četiri šarda sa po 150 gigabajta kapaciteta. U svakom čvoru će ostati po 200 gigabajta slobodnog prostora što se može iskoristiti za skladištenje drugih indeksa ukoliko je potrebno. Šard jer moguće smestiti na bilo kom čvoru. Na primer, ukoliko je indeks podeljen na pet šarda, u tom slučaju šardovi se mogu ukoliko je neophodno, ali ne moraju podeliti na pet različita čvorova. Šarding ide ruku pod ruku sa povećavanjem dostupnog prostora na disku ukoliko je neophodno, potencijalno dodavanjem novih čvorova u klasteru. Svaki šard je nezavisan i može se posmatrati kao potpuno funkcionalni indeks. Svaki šard zapravo podstavlja Apache Lucene indeks jer je Elasticsearch izgrađen na Apache Lucene platformi. To znači, da Elasticsearch indeks sa pet šarda se zapravo sastoji od pet Apace Lucene indeksa u pozadini. Šard nema predefinisanu veličinu prostora na disku izdvojenu za sladištenje podataka, ali zato ima limit u broju dokumenata koje može da skladišti a to je nešto više od dva biliona dokumenata po šardu.

Glavni razlog deljenja indeksa na šardove jeste skaliranje obima podataka. Pomoću šardinga moguće je uskladištiti bilione dokumenata u sklopu jednog indeksa, što uobičajno ne bi bilo izvodljivo bez šardinga. Još jedna uobičajna upotreba šardinga nad indeksom, jeste deljenje indeksa na manje delove koji se lakše uklapaju na čvorovima. Pomoću šardinga se znatno poboljšavaju performance sistema, jer takođe omogućava da se upiti distribuiraju i paralelno izvršavaju preko indeksa. Ovo znači da se upit može izvršiti na više šardova u istom vremenskom intervalu povećavajući na taj način performance i propusnost u sistemu. Šardove je moguće smestiti na više različitih čvorova što znači da hardver može biti iskorišćen na više različitih čvorova.

Do verzije 7 Elasticsearch je inicijalno dodeljivao pet šarda jednom indeksu. Veoma često se kreiraju puno malih indeksa zajedno sa malim klasterima, što nije pogodno jer tada dolazi do over-sharding-a odnosno do prevelikog broja šarda. Ranije nije bilo moguće da se promeni broj šardova jednom kada se indeks kreira, pa su se u tom slučaju kreirali novi indeksi kako bi se povećao broj šardova. Pored toga što je ovaj proces veoma nezgodan takođe je znao da oduzima puno vremena.

Kako bi se ovaj problem prevazišao, nakon verzije 7 Elasticsearch prilikom kreiranja indeksa inicijlano kreira jedan šard, što je za male i srednje indekse sasvim dovoljno. Ukoliko je potrebno povećati broj šardova u sistemu za to postoji Split API [6]. Ovaj API takođe uključuje kreiranje novih indeksa kako bi se povećao broj šardova, ali sam proces je dosta jednostavniji za korišćenje. Za potrebe redukovanja broja šardova u sistemu postoji Shrink API [7].

Koliki je optimalan broj šardova potreban u sistemu? Odgovor na ovo pitanje ne postoji. Optimalan broj šardova u sistemu zavisi od mnogih faktora, počev od broja čvorova u klasteru, od njihovog kapaciteta, od broja indeksa i njihovih veličina, od broja upita koji se izvršavaju nad indeksima...Dovoljno je da se zna da, ukoliko se predviđa da će indeks sadržati milion dokumenata da je potrebno kreirati bar pet šarda. U suprotnom je sasvim dovoljno uzeti inicijlane vrednsti pa tek nakon toga povećavati broj šardova ukoliko postoji potreba za tim.

Šarding dakle predstavlja pro ces podele indeksa na manje delove koji se nazivaju šardovima. Ovom tehnikom se omogućava povećanje veličine indeksa i poboljšanje propusnosti indeksa. Glavni razlog jeste skaliranje podataka pa povećanje propusnosti indeksa predstavlja bonus. Šarding tehnikom moguće je uskladištiti indeks koji zauzima čak 700 gigabajta prostora na disku, čak iako ne postoji nijedan čvor koji ne može da prihvati tu količinu podataka.

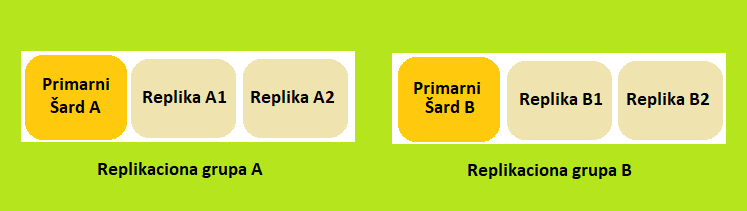
# Replikacija

Kada čvor na kome je smešten šard sa podacima otkaže dolazi do gubitka podataka jer njihova kopija ne postoji. Jedan od najvećih problema jeste taj što hardver u bilo kom trenutku može da otkaže. Ukoliko se koriste više hard diskova za rad sa klasterom veće su šanse da dođe do otkaza. Zbog problema sa hardverom Elasticsearch podržava replikaciju podataka kao otpornost na ovakve otkaze. Elasticsearch podržava replikaciju šardova i što je pri pokretanju sistema inicijalno pokrenuto bez ikakvih dodatnih konfiguracija. Mnoge baze podataka takođe obezbeđuju replikaciju podataka ali njhova konfiguracija zna da bude veoma komplikovana. Dakle, kod Elasticsearch baze podataka replikacija podataka je veoma laka i jednostavna.

## Način rada replikacije

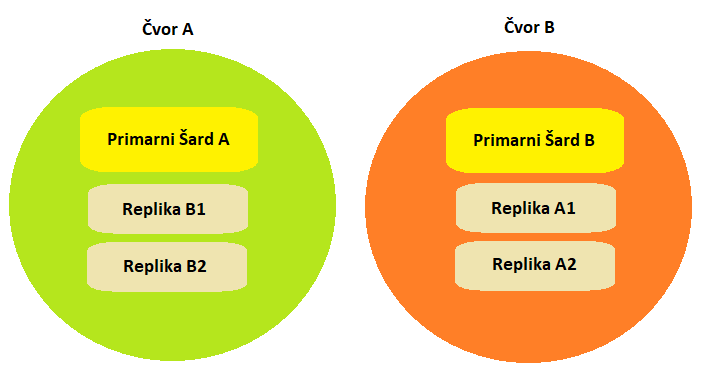
Kao što je već rečeno, indeks se kreira i konfiguriše za skladištenje podataka zajedno sa velikim brojem šardova koji se mogu čuvati na distribuiranim čvorovima. Isto tako, replikacija se konfiguriše na nivou indeksa. Replikacija funkcioniše tako što kreira kopije svakog šarda koji indeks sadrži. Ove kopije kod Elasticsearh baze podataka se nazivaju kopije šardova (eng. „replica shards“). Šard koji se replicira jednom ili više puta se naziva primarnim šardom (eng. „primary shard“). Primarni šard i prelicirani šard se kod Elasticsearch-a nazivaju i kao replikaciona grupa (eng. „replication group“).

Replicirani šardovi predstavljaju kompletnu kopiju šardova pa se nad njima mogu primenjivati razni upiti baš kao i nad primarnim šardovima. Prilikom kreiranja indeksa može se specificirati koliko replika svakog šarda će Elasticsearch kreirati. Dakle, indeks sadrži šardove koji mogu da sadrže svoje kopije. Na slici 8 su prikazane dve replikacione grupe A i B gde indeks sadrži dva primarna šarda gde svaki sadrži po dve svoje kopije.



**Slika 8**: Primer replikacionih grupa A i B

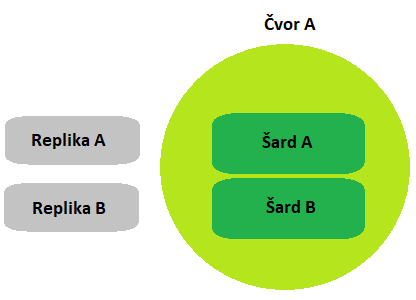
Međutim ukoliko dođe do otkaza hardvera, pored primarnih šardova izgubiće se i primarne kopije. Zbog toga Elasticsearch nikada ne skladišti primarne šardove i njihove kopije na istim čvorovima. Ukoliko jedan čvor nestane odnosno da ode u grešku, ovim obezbeđuje da uvek postoji bar jedna kopija primarnog šarda koja je dostupna na nekom drugom čvoru. Koliko kopija će ostati zavisi od same konfiguracije indeksa za kreiranje replika i koliko čvorova klaster sadrži. Na slici 9, može se videti da su kopije šardova smešteni na drugom čvoru u odnosu na primarni šard kome pripadaju:



**Slika 9**: Primer skladištenja primarnih šardova i njihovih replika

Replikacija ima smisla za klastere koji sadrže više od jednog čvora jer u suprotnom replikacijom nije moguće sprečiti gubitak podataka ukoliko čvor otkaže. Naravno, moguće je konfigurisati indekse da kreira kopije iako postoji samo jedan čvor u klasteru ali to neće imati nikakvog efekta ukoliko se ne doda još jedan čvor u klasteru.

Recimo da postoje dva indeksa u klasteru i da oba indeksa za replikaciju koriste inicijalnu konfiguraciju. Svaki indeks sadrži po jedan šard i smešteni sun a čvoru A kao na slici 10:



**Slika 10**: Čvor A sa primarnim šardovima A i B

Iako su inicijalno kreirane i replike šardova A i B, one nisu dodeljene jer postoji samo jedan čvor u klasteru. Dodavanjem još jednog čvora u klasteru Elasticsearch omogućiće replikaciju u sistemu odnosno dodeliće replike drugom čvoru jer se na prvom čvoru nalaze primarni šardovi A i B. Da je inicijalna kofiguracija promenjena na dve replike po šardu u tom slučaju bi se smestile i te dve dodatne kopije na drugom čvoru. Dodavanjem još jednog čvora u klasteru, Elasticsearch bi i na tom čvoru prosledio replike šardova kako bi dodatno obezbedio dostupnost podataka i u tom slučaju sistem bi podržao i otkaz čak dva čvora u istom vremenskom intervalu bez gubitka podataka.

## Izbor broja replika

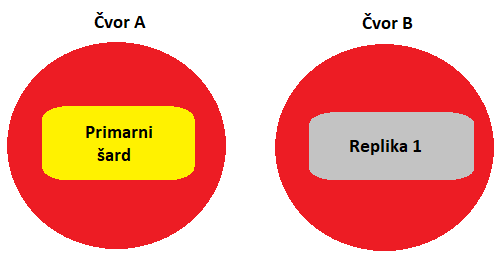
Glavno pitanje kod Elasticsearch baze podataka jeste koliki broj replika je idealan? Uglavnom jedna ili dve replike po šardu su sasvim dovoljne ali to isto zavisi i od same konfiguracije baze. Na primer ukoliko dva čvora otkažu u istom vremenskom intervalu da li je moguće tada povratiti podatke iz drugog izvora podataka na primer iz neke relacione baze podataka. Prilikom vraćanja podataka u Elasticsearch bazi podataka podaci su nedostupni za rad sve dok se u potpunosti ne vrate. Ukoliko se Elasticsearch koristi za nešto veoma bitno poput bolničkih podataka, ovaj rizik, nedostupnost podataka bi bio preveliki. S tim u vezi, ukoliko Elasticsearch koristi podatke koji nisu previse bitni, trebalo bi bar jednom replicirati šardove dok s druge strane ukoliko su podaci veoma bitni trebalo bi praviti minimalno dve replike po šardu i više. To takođe znači da su potrebni bar dva ili više čvorova u klasteru što bi trebalo uzeti u obzir prilikom same konfiguracije sistema kako bi se podaci zaštitili od gubitaka.

## Slikanje podataka

Kao i kod drugih baza podataka Elasticsearch takođe podržava tzv. slikanje podataka (eng. „snapshots“) kako bi se sprečio gubitak podataka. Slikanje podataka se koristi za vraćanje podataka u nekom vremenskom intervalu ukoliko je došlo do gubitka podataka. Slikanje podataka se može raditi kako na nivou indeksa tako i na nivou celog klastera. Pošto Elasticsearch podržava slikanje podataka zašto je onda potrebna replikacija? Replikacija je zapravo sprečava da dođe do gubitka podataka ali ona funkcioniše samo nad živim podacima (eng. „live data“). Ovo praktično znači da replikacija obezbeđuje da se podaci smešteni u indeksu u nekom vremenskom intervalu ne izgube. S druge strane, slikanjem podataka Elasticsearch obezbeđuje da izvoz podataka iz klastera (ili određenih indeksa) koje je zatečeno u nekom vremenkom intervalu u fajl. Taj fajl se kasnije može iskoristiti za vraćanje podataka odnosno za vraćanje na prethodno snimljeno stanje klastera ili indeksa u sistemu. Ovo je veoma pogodno na primer kada je potrebno vratiti milione dokumenata koji su smešeni u indeksu. Slikanje podataka se obično radi pre izvršavanja upita. Izvršavanjem upita veoma često dolazi do nekih neplaniranih akcija na primer jer se se testni podaci razlikuju od produkcijskih živih podataka. Koji god da je razlog, podaci u sistemu su narušeni i potrebno je vraćanje tih podataka na prethodno stanje. Replikacija ovde ne pomaže jer je njen zadatak da obezbedi da do gubitka podataka ne dođe. Umesto replikacije, potrebno je podatke vratiti prethodno napravljenom slikom. Slikanje podataka se uobičajno radi za dnevno vraćanje podataka i ručno slikanje podataka se može primeniti u trenutku pre promene podataka kako bi se dodatno obezbedilo čuvanje podataka u slučaju neplaniranih grešaka. S druge strane, replikacija samo obezbeđuje oporavak indeksa u slučaju otkaza čvora i preusmerava izvršenje upita na drugi čvor kao da se ništa nije desilo.

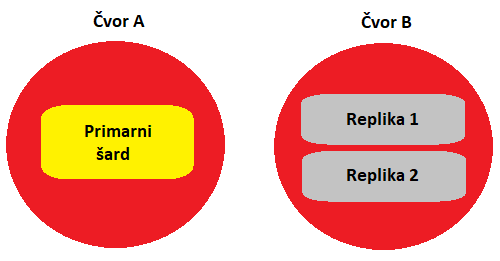
## Druga uloga replikacije

Pored toga što sprečava gubitak podataka, replikacija se koristi za povećanje propusnosti indeksa u sistemu. Na primer, za veb sajt na kojem se prodaju neki proizvodi kod Elasticsearch baze podataka oni se smeštaju u indeks „proizvodi“. Najpopularniji proizvodi se smeštaju na početnoj stani veb sajta i upiti se veoma često izvršavaju nad njima. Inicijalno, Elasticsearch baza podataka je konfigurisana tako da indeks sadrži samo jedan šard sa jednom replikom jer veb sajt ne sadrži previše proizvoda tj. dokumenata. Bez obzira na mali broj proizvoda puno upita se izvršavaju nad njima. U ovom slučaju može da dođe do zagušenja primenom upita od strane puno korisnika. Ovo zagušenje se može rešiti dodavanjem novog čvora u klasteru kao na slici 11:



**Slika 11**: Dodavanje dodatnog čvora kao pomoć prilikom zagušenja u sistemu

Imajući u obzir da je kreiran samo jedan šard i jedna njegova kopija ovo rešenje neće pomoći jer podatke nije moguće distribuirati na više od dva čvora. Kao rešenje moguće je kreirati još jedan primarni šard i dodatne replike šardova ukoliko se kreira još jedan čvor kako bi se povećala propusnost ali time bi se samo povećala cena (eng. „cost”) sistema. Umesto moguće je povećati broj replika za jedan ili za onoliko koliko je potrebno kao na slici 12:

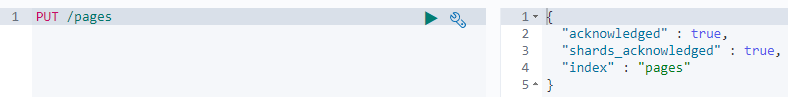


**Slika 12**: Dodavanje dodatne replike

Kako postoje samo dva čvora dodavanjem dodatne replike se ne povećava dostupnost indeksa ali se povećava propusnost indeksa. Veća propusnost se dobija jer se kod Elasticsearch baze podataka replike ponašaju kao primarni šard jer se nad njima takođe mogu pisati upiti. Ovo praktično znači da se obe replike mogu koristiti u istom vremenskom intervalu tj. da se izvršavaju upiti nad njima. Ovo je moguće jer Elasticsearch automatski koordiniše gde će se upit izvršiti i zbog paralelne obrade koju takođe obezbeđuje. Elasticsearch će automatski izabrati šard nad kojim će se izvršiti upit na osnovu mnogih faktora. To praktično znači da ukoliko postoje tri upita koje je potrebno izvršiti u istom vremenskom intervalu da će se oni izvršiti na tri različita šarda: nad primarnim šardom i nad njegovim replikama. Dakle, paralelno je moguće obrađivati upite nad istim indeksom. Dodavanjem novih replika samo dolazi do povećanja performansi ukoliko resursi hardvera nekog čvora nisu potpuno iskorišćeni. Naravno, ukoliko su čvorovi već zauzeti obradom zahteva drugih indeksa dodavanjem novih replika neće se videti poboljšanje u sistemu. Pored toga, potreban je dodatni prostor na disku za skladištenje replike šardova jer se radi o kopiji primarnog šarda nad kome se takođe mogu primenjivati upiti. Replikacija se dakle koristi za povećanje dostupnosti i propusnosti indeksa u sistemu.

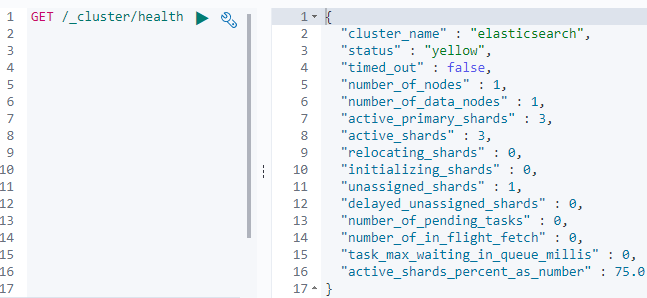
## Kibana - korišćenje replikacije

Kreiranje novog indeksa u Kibana konzoli je veoma jednostavno navođenjem ključne reči „PUT“ i specificiranjem naziva za indeks kao na slici 13:



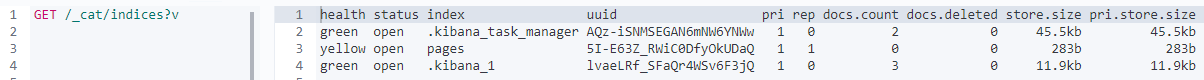
**Slika 13**: Kreiranje indeksa „pages”

S obzirom na to da ništa nije specificirano, indeks je kreiran prema inicijalnim podešavanjima kreiranjem jednog primarnog šarda i jedne replike šarda. Nakon kreiranja indeksa naredbom „GET/\_cluster/health“ se prikazuje stanje klastera koje je prikazano na slici 14:



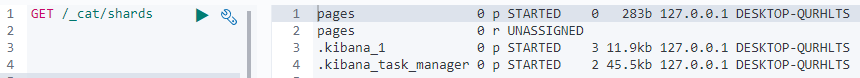
**Slika 13**: Stanje klastera nakon kreiranje indeksa sa primarnim šardom i replikom

Veoma je važno to što je status klastera „yellow“ (žut) a ne „green“ (zelen). Ukoliko se izlistaju indeksi koji postoje u klasteru videće se razlog zbog čega je status klastera „yellow“ [3]. Na slici 14 prikazana je naredba za prikaz više informacija o kreiranim indeksimana klasteru:



**Slika 14**: Prikaz više informacija o kreiranim indeksima na klasteru

Na slici 14, se vidi da je status na novokreiranom indeksu „pages“ žut. Razlog za to jeste što indeks sadrži repliku šarda ali da taj šard nije dodeljen nijednom čvoru. To praktično znači da ne postoji dodatni čvor na kome bi Elasticsearch mogao da smesti repliku šarda. Replika šarda je u stanju čekanja da bude dodeljena prvom kreiranom čvoru u klasteru. Zbog toga je status klastera žute boje. Indeks je potpuno funkcionalan ali postoje rizici od gubljenja podataka ukoliko čvor ode u grešku pa status „yellow“ predstavlja upozorenje da podaci mogu da se izgube. Kako bi se ovo dokazalo na slici 15 prikazane su dodatne informacije o šardovima:

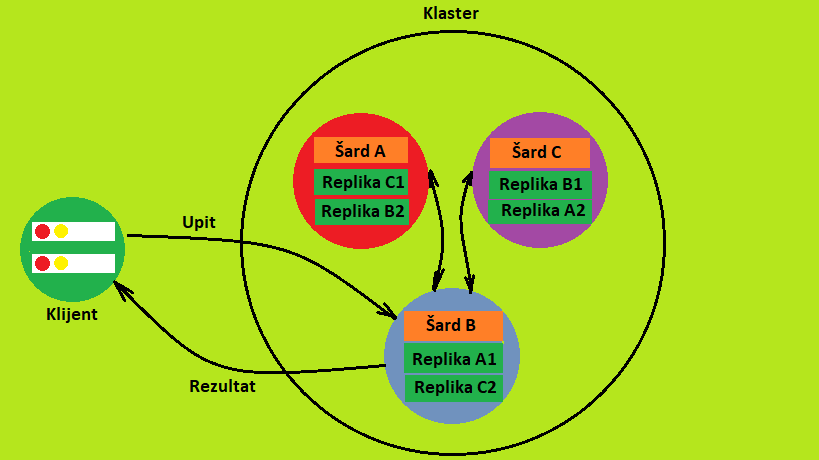


**Slika 15**: Prikaz dodatnih informacija o šardovima

Rezultati na slici 15, pokazuju listu svakog kreiranog šarda zajdeno sa dodatnim informacijama o šardovima uključujući kome indeksu pripadaju. Na samom vrhu se mogu videtu dva kreirana šarda za novokreirani indeks „pages“ od kojih je jedan primarni označen slovom „p“ u trećoj koloni i jednom replikom (označen slovom „r“ u drugom redu u trećoj koloni). Sledeća kolona predstavlja status šarda. Kao što se može videti sa slike, primarni šard ima status „STARTED“ što znači da je potpuno funkcionalan i dostupan za upite. Repliciran šard, sa druge strane ima status „UNASSIGNED“ jer je potrebno dodati još jedan čvor u klasteru kako bi replikacija imala efekta u sistemu.

## Izvršavanje upita nad replikama

Pretpostavimo da klaster sadrži tri čvora sa jednim indeksom koji je distriburiran preko tri šarda A, B i C kao na slici 16:



**Slika 16**: Primer pretraživanja podataka nad replikama

Svaki šard sadrži po dve replike, tako da se svaka replikaciona grupa sastoji od primarnog šarda i dve replike. Pretpostavimo da klijent pošalje upit klasteru koji se prosleđuje čvoru koji sadrži šard B. Ovaj čvor se naziva koordinator („coordinated node“) čime se označava da je on odgovoran za slanje upita drugim čvorovima u klasteru, da prihvati rezultat i da ga pošalje klijentu. Inicijalno, svaki čvor može da se ponaša kao koordinator što znači da svaki čvor može da prihvati HTTP zahtev. Kako koordinacioni čvor sadrži šard koji je potrebno pretražiti čvor će sam izvršiti upit. Naravno, ovo ne mora uvek da bude uvek slučaj ali kako se ovde radi sa jednim indeksom ovo će uvek biti slučaj. Koordinacioni čvor zatim šalje (eng. „broadcast”) zahtev svakom šardu sa tim indeksom nebitno da li čvorovi sadrže primarni šard ili repliku. Čvorovi zatim šalju koridinacionom čvoru rezultate koje on spaja i sortira i takav rezltat vraća nazad klijentu [4].

# Zaključak

U radu je predstavljen i bliže okarakterisan način replikacije podataka kod Elasticsearch baze podataka. Kroz poglavlja je opisana i sama arhitektura Elasticsearch baza podataka, rasčlanjene su komponente sistema i objašnjena je njihova uloga u sistemu i detaljno je objašenjana replikacija.

Hardver u bilo kom trenutku može da otkaže i softver ponekad može da sadrži bagove. Dodavanjem što više hardvera u sistemu rizik da hardver prestane sa radom iz nekog razloga je sve veći. Uzimajući ovo u obzir, najbolje je koristiti mehanizam koji je tolerantan na otkaze tj. mehanizam koji je tolerantan na greške u sistemu i koji sadrži mehanizam za otkazivanje (eng. „failover”) [5]. U tom slučaju najbolje je izabrati mehanizam replikacije u sistemu. Kod Elasticsearch-a ovaj mehanizam se dobija automatski i veoma se jednostavno koristi nad šardovima. Svakom replikacijom šarda u sistemu dodatno sistem povećava svoju propusnost jer se svaka replika tretira kao primarni šard. Replike i primarni šardovi se nikada ne skladište na istom čvoru u sistemu tako da u slučaju otkaza nekog čvora podaci i dalje postoje na nekom drugom čvoru.

# Literatura

1. Apache Lucene veb sajt <https://lucene.apache.org/>
2. Elastic veb sajt <https://www.elastic.co/>
3. Kibana veb sajt <https://www.elastic.co/kibana>
4. CodingExplained [https://codingexplained.com/understanding-replication-in-elasticsearch](https://codingexplained.com/coding/elasticsearch/understanding-replication-in-elasticsearch)
5. Opster Elasticsearch Replication [https://opster.com/elasticsearch-replication/](https://opster.com/elasticsearch-glossary/elasticsearch-replication/)
6. Split API [https://www.elastic.co/indices-split-index.html](https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-split-index.html)
7. Shrink API [https://www.elastic.co/indices-shrink-index.html](https://www.elastic.co/guide/en/elasticsearch/reference/master/indices-shrink-index.html)