**Univerzitet u Nišu, Elektronski fakultet**

logo_1960_4**Katedra za računarstvo**

**Replikacija podataka kod MariaDB**

Mila Rančić 1370

**Predmet: Sistemi za upravljanje bazama podataka**

Sadržaj

[Uvod 3](#_Toc106917727)

[Replikacija podataka kod MariaDB 4](#_Toc106917728)

[Tipovi replikacije 5](#_Toc106917729)

[Kako funkcioniše replikacija? 6](#_Toc106917730)

[Paralelna replikacija 7](#_Toc106917731)

[Slave logs 8](#_Toc106917732)

[Binlog kod MariaDB 8](#_Toc106917733)

[Binarno evidentiranje zasnovano na naredbama 8](#_Toc106917734)

[Binarno evidentiranje zasnovano na redovima 10](#_Toc106917735)

[Izbor formata binarnog log-a 11](#_Toc106917736)

[Global transaction id - GTID 11](#_Toc106917737)

[Dual master 12](#_Toc106917738)

[Konfigurisanje replikacije 14](#_Toc106917739)

[Instaliranje potrebnih alata i servisa 14](#_Toc106917740)

[Konfigurisanje master instance 15](#_Toc106917741)

[Konfigurisanje slave instance 17](#_Toc106917742)

[Primeri replikacije podataka 20](#_Toc106917743)

[Master – slave replikacija 20](#_Toc106917744)

[Promena master-a 22](#_Toc106917745)

[GTID replikacija 24](#_Toc106917746)

[Dual-master replikacija 30](#_Toc106917747)

[Zaključak 35](#_Toc106917748)

[Reference 36](#_Toc106917749)

# Uvod

U današnje vreme, zbog postojanja sve većeg broja aplikacija koje skladište podatke u baze podataka, replikacija je neophodna i primenjuje se svuda. Od bankovnih računa preko Facebook profila do Instagram slika, svi podaci koje ljudi smatraju važnim gotovo sigurno će se replicirati na više mašina kako bi se osigurala dostupnost i trajnost podataka.

Replikacija omogućava konfigurisanje jednog ili više servera kao replike drugog servera, čuvajući njihove podatke sinhronizovanim sa glavnom serverom. Glavni server je poznat kao takav jer obezbeđuje podatke za replikaciju. Replicirani podaci mogu biti kompletan skup baza podataka, jedna baza ili čak tabele podataka preuzete iz željene baze.

Osnovni problem koji rešava replikacija je da se podaci jednog servera sinhronizuju sa nekim drugim. Ovo ne samo da je korisno za poboljšanje performansi aplikacije, već je i osnova mnogih strategija za visoku dostupnost, skalabilnost, oporavak od grešaka, pravljenje rezervnih kopija, analiza, skladištenje podataka i mnogih drugih zadataka.

U prvom delu je objašnjeno šta je replikacija i kako ona funkcioniše kod MariaDB. Naredni deo bavi se, verovatno i najvaznijim principom kod replikacije – log fajlovima. Tu je objašnjeno koje vrste log fajlova postoje, kako izgledaju podaci u svakom od njih i po cemu se oni razlikuju. Takođe, obradjene su njihove prednosti i nedostaci i kada bi trebalo koji koristiti.

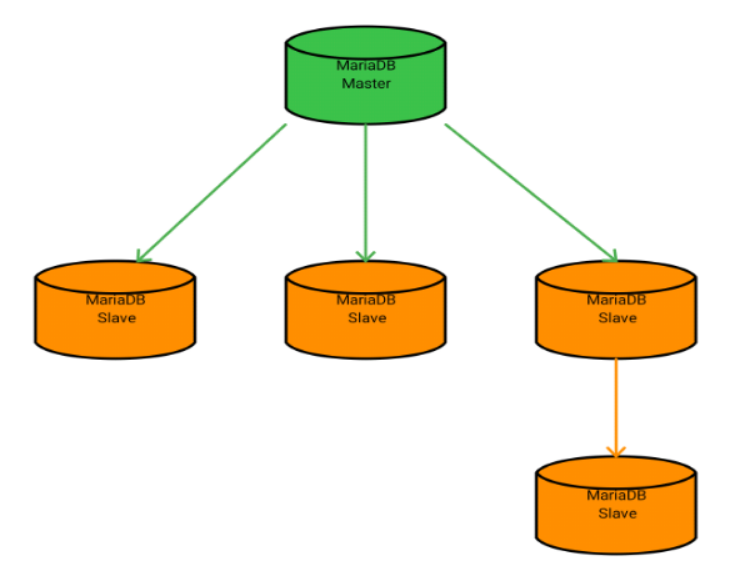
Treći deo odnosi se na demonstraciju nekih od koncepata objasnjenih u prethodna dva poglavlja. U njemu je izvršena konfiguracija master i slave servera i na konkretnim primerima demonstrirano kako replikacija funkcioniše. Pokrivena su oba osnovna tipa replikacije: standardni i GTID. Pored toga, prikazana je i dual master arhitektura, kako se konfiguriše i kako radi.

# Replikacija podataka kod MariaDB

Replikacija je proces kopiranja podataka iz različitih baza podataka, poznatih kao glavne (master) baze ili serveri na više podređenih (slave) servera. Master server naziva se glavni server, jer obezbeđuje podatke za replikaciju. Svi slave serveri koji su povezani sa istim masterom vide i rade na istim podacima. Replicirani podaci mogu biti kompletan skup baza podataka, jedna baza ili čak tabele podataka preuzete iz željene baze.

MariaDB podržava ugrađenu replikaciju, koja se može koristiti u više namena. Najčešći razlog za to je izgradnju okruženja za rešavanje niza različitih problema sa performansama, podržavajući pravljenje rezervnih kopija i ublažavanje grešaka sistema.

Master evidentira ažuriranja, koja se zatim prenose do slave servera. Slave emituje poruku u kojoj se navodi da je uspešno primio ažuriranje, što omogućava slanje narednih ažuriranja.



Slika : Standardna master-slave replikacija

Upotreba replikacije kod MariaDB utiče na poboljšanje:

* Skalabilnosti
* Sigurnosti podataka
* Analize podataka
* Distribucije podataka na daljinu

***Skalabilnost:*** Posedovanje jednog ili više slave servera omogućava čitanje podataka na njima, čime se smanjuje opterećenje glavnog servera, koji dozvoljava samo obavljanje operacije pisanja.

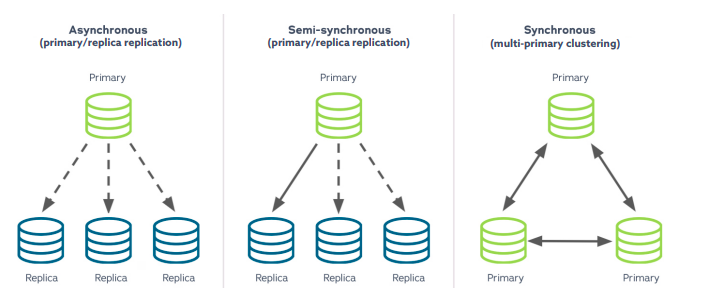
***Rezervne kopije***: Ovo uključuje repliciranje podataka na slave servere koji se mogu koristiti kao rezervne kopije podataka u slučaju pada mastera. Ova rezervna kopija može delovati kao samostalni server u stabilnom stanju. Slave se može pauzirati i isključiti bez uticaja na rad mastera, tako da se može napraviti efikasan snimak „živih“ podataka koji bi inače zahtevali da se master isključi.

***Analiza podataka***: Podaci se mogu analizirati na slave serveru bez dodavanja dodatnog opterećenja glavnom serveru.

***Distribucija podataka***: Korišćenje replikacije, omogućava kopiranje podataka i manipulaciju njima lokalno bez povezivanja na udaljeni master server. Nakon naknadnog povezivanja, promene će biti vidljive master serveru i ostalim slave serverima.

## Tipovi replikacije

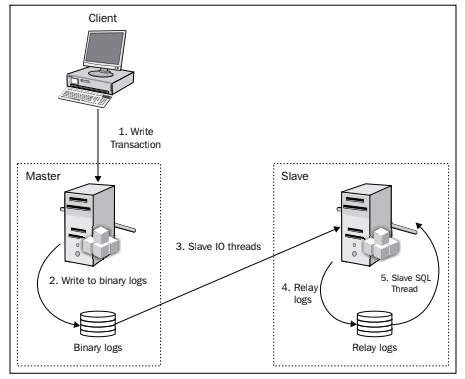
Standardna MariaDB replikacija je asinhrona, ali MariaDB takođe pruža opciju semi-sinhrone i sinhrone replikacije. Razlika je jednostavno u vremenu širenja promena. Kod sinhrone replikacije ukoliko je doslo do promene na masteru, promena će odmah biti primenjena na ostale slave servere. Od slave servera se očekuje da vrate potvrdu da su primili dogadjaj. U slučaju asinhrone, promene će biti stavljene u red čekanja i zapisane nešto kasnije. Takođe, to znači da nema potrebe za stalnom vezom između master-a i slave-a.

Kašnjenje sa asinhronom replikacijom može biti kratko ili dugo. Što više upita za pisanje postoji, to je više informacije koje će morati da se repliciraju. U najboljim slučajevima, vremenska razlika replikacije može biti u sekundama ili minutima, ali u najgorem slučaju to može biti sat ili nekoliko sati. Loša strana korišćenja ovog tipa replikacije je što, ukoliko dođe do toga da se master server „sruši“, neke od najnovijih promena mogu biti izgubljene. Semi-sinhrona replikacija čeka samo jednu repliku da potvrdi da je primila i evidentirala događaje. Ona ima negativni uticaj na performanse, ali povećava integritet podataka. Kašnjenje kod nje zasnovano na vremenu povratnog puta do replike i nazad, moguće je minimizovati ga za servere u neposrednoj blizini preko brzih mreža.

Slika 2: Tipovi replikacije

## Kako funkcioniše replikacija?

Replikacija je funkcija koja omogućava da se sadržaj jednog ili više master servera preslikava na jedan ili više slave servera (koji se nazivaju replike).

 Glavni mehanizam koji se koristi u replikaciji je binarni log. Ako je binarno evidentiranje omogućeno, sva ažuriranja baze podataka (manipulacija podacima i definicija podataka) se upisuju u binarni log kao događaji. Replike čitaju binarni log sa svakog master servera da bi pristupile podacima za repliciranje. Relay log se kreira na slave serveru. Stare evidencije relay loga se uklanjaju kada više nisu potrebne.

Slika 3: Šema standardne master-slave replikacije

Slave server vodi evidenciju dokle je replicirao podatke iz master binlog-a. Pamti poziciju poslednjeg primenjenog događaja. Ovo mu omogućava da se ponovo poveže i nastavi sa mesta gde je stao ukoliko dođe do privremenog zaustavljanja replikacije. Master i slave serveri ne moraju da budu u stalnoj međusobnoj komunikaciji. Sasvim je moguće isključiti servere van mreže ili prekinuti vezu sa mrežom, a kada se vrate, replikacija će se nastaviti tamo gde je stala. Takođe omogućava da se slave server isključi, klonira, a zatim da novi slave server nastavi replikaciju sa istog mastera.

U MariaDB replikaciji se koriste tri vrste niti, što je prikazano u sledećoj tabeli:

*Table 1: Vrste niti kod MariaDB*

|  |  |
| --- | --- |
| Gde se izvršava | Ime niti |
| Master | Binlog dump thread |
| Slave | SQL I/O thread |
| Slave | Slave SQL thread |

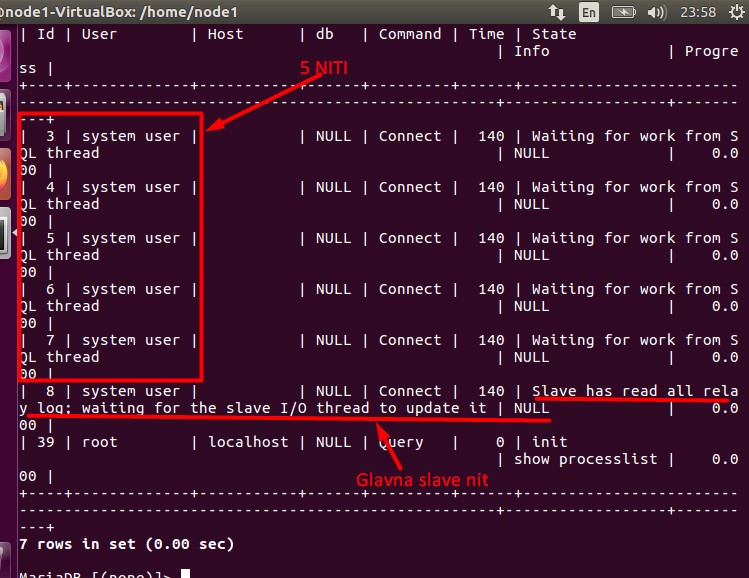
Kad god se izmeni master baza podataka, promena se upisuje u datoteku, takozvani binarni log ili binlog. Ovo radi klijentska nit koja je izvršila upit koji je modifikovao bazu podataka. Na masteru se pokreće Binlog dump nit, koja neprekidno čita masterov binlog i šalje ga slave-u. Slave kreira SQL I/O nit, koja prima taj log i upisuje ga u datoteku: relay log. Ona ne izvršava događaje direktno. Vec poseduje i drugu nit, koja se zove SQL nit, koja čita relay log i primenjuje promene na slave server. Bilo koja konekcija između mastera i slave-a je zahtevana od strane slave-a.

## Paralelna replikacija

Pre MariaDB 10.0, bilo je moguće pokrenuti samo jednu slave SQL nit za svaki slave server. Nepostojanje niti loše je za performanse, jer obično master izvršava jednu operaciju pisanja koristeći nekoliko paralelnih niti. Ponekad jedna nit na slave serveru nije dovoljana da ponovi događaje mastera sa prihvatljivim performansama.

MySQL takođe poseduje paralelnu replikaciju, ali se ona razlikuje od one kod MariaDB po implementaciji i konfiguraciji.

Najvažnija promenljiva servera za MariaDB paralelnu replikaciju je @@slave\_parallel\_threads. Paralelna replikacija se sastoji od pokretanja grupe niti koja je u stanju da primeni mnoge događaje na paralelan način. Svaka nit u ovom skupu naziva se radna nit. Ono što je važno je da se ne mogu svi događaji paralelizovati. MariaDB će ipak izvršiti neke operacije uzastopno, kada je to neophodno za ispravno repliciranje podataka.

 Ova funkcija je opciona i nije podrazumevano omogućena. Da bi se osposobila, neophodno je da se prilikom konfigurisanja mastera, u podešavanja, postavi vrednost promenljive @@slave\_parallel\_threads. Vrednost predstavlja broj radnih niti koje će biti pokrenute na svakom slave serveru. Kao posledica toga, svi slave koji repliciraju podatke sa istog mastera će imaju isti broj radnih niti. Takođe, ako slave replicira više mastera, isti broj radnih niti mora biti konfigurisan na svim masterima.

Slika 4: Niti kod paralelne replikacije

## Slave logs

Slave serveri moraju da zabeleže informacije o konfiguraciji i trenutnom stanju replikacije. Ove informacije se ne smeju izgubiti, čak ni u slučaju pada servera.

Zbog toga, svaki slave održava tri loga:

* ***Relay log*** -sadrži događaje koje je primio od binarnog log-a mastera. Ovaj log piše slave I/O nit, a čita i izvrsava slave SQL nit, ili radna nit ukoliko je paralelna replikacija uključena.
* ***Master log*** - čuva informacije koje su neophodne za povezivanje sa masterom, kao i koordinate binarnih logova mastera. Koordinate se sastoje od imena loga i pozicije poslednjeg binarnog događaja loga koji je primljen.
* ***Relay log info*** - čuva informacije o poslednjem događaju relay loga koji je primenjena od strane slave SQL niti ili radnih niti.

Log relay-a je uvek upisan u datoteke. Informacije master log-a i relay-a se mogu upisati u datoteke ili u sistemske tabele u bazi podataka.

## Binlog kod MariaDB

MariaDB replikacija je zasnovana na binarnom logu. Binarni log vodi evidenciju o događajima koji modifikuju baze podataka. Organizovan kao linearni niz događaja. SQL upit koji menja bazu podataka će generisati jedan ili više događaja i dodati ih u binlog.

Upiti se, u binlog, mogu evidentirati na tri načina:

* Statement format
* Row format
* Mixed format

Sa STATEMENT formatom, događaji su svi SQL upiti koji menjaju ili bi mogli da promene neke podatke. Upisani su u tekstualnom formatu. Sa ROW formatom, događaji su sve modifikacije koje se javljaju kao a posledica odgovarajucih naredbi. Upisani su u binarnom formatu. MIKSED format beleži naredbe kada je to moguće, a može i da snimi modifikacije.

U zavisnosti od formata, mogu se definisati dve vrste replikacije:

1. Replikacija zasnovana na naredbama (statement-based) ili
2. Replikacija zasnovana na redovima (row-based).

### Binarno evidentiranje zasnovano na naredbama

Format STATEMENT je najstariji format binarnog log-a. Ukoliko se primenjuje ovaj format, za evidentiranje naredbi, biće zabeležene sve naredbe koje sadrže: UPDATE, DELETE ili REPLACE sa WHERE klauzolom, jer se njima menjaju podaci. Ove naredbe biće upisane čak iako imamo situaciju da se njima ne menja nijedan red. Naredba kao što je SELECT se nikada ne upisuju u log, jer se njom ne menjaju podaci.

Format STATEMENT ima važno ograničenje: samo determinističke naredbe treba poslati na server. Determinističke znači da, ako se izvrše dva puta na identične baze podataka, one moraju nužno imati iste efekte. Naravno ovo ograničenje se ne primenjuje na naredbe koje ne menjaju podatke, kao što je SELECT, jer se njima ne utiče na replicirane podatke.

Ako naredba koristi trenutnu vremensku oznaku, trenutnog korisnika ili nasumične podatke, one će da proizvode različite rezultate na slave serverima. MariaDB pokušava da otkrije nesigurne naredbe i daje sledeća upozorenja:

*Note (Code 1592): Unsafe statement written to the binary log using statement format since BINLOG\_FORMAT = STATEMENT. Statement is unsafe because it uses a system function that may return a different value on the slave.*

Međutim, MariaDB ne proverava da li su vrednosti dodeljene korisničkim promenljivama determinističke ili ne. Odnosno, kada se koriste promenljive, lako je umetnuti nedeterminističke podatke bez dobijanja bilo kakvog upozorenja:

*MariaDB [test]> SET @a = RAND();*

*Query OK, 0 rows affected (0.00 sec)*

*MariaDB [test]> CREATE TABLE example ENGINE = InnoDB SELECT @a;*

*Query OK, 1 row affected (0.50 sec)*

*Records: 1 Duplicates: 0 Warnings: 0*

MariaDB poseduje mehanizme za otkrivanje nedeterminističkih naredbi. Pošto se koriste algoritmi koji ne analiziraju neke naredbe kako treba, 1592 upozorenje je osmišljeno za neke determinističke naredbe. Ovaj problem je dobro poznat i može uzrokovati da se log za evidenciju grešaka brzo prepuni. Kao jedno od resenja preporučuje se upotreba MIKSED formata.

Postoje brojni razlozi zašto neka naredba nije deterministicka. Neki od njih su sledeci:

* Postojanje nedeterminističkih funkcija, kao što su RAND() ili USER(), kao i FOUND\_ROWS(), ROW\_COUNT() i LOAD\_FILE().
* Korisnički definisane funkcije (funkcije napisane na jeziku C i instalirane na server).
* Reference na serverske promenljive. Njihova vrednost se može razlikovati na slave serverima. Postoje neki izuzeci, ali, pošto oni trenutno nisu dokumentovani, preporucljivo je da se jednostavno izbegava referenciranje serverskih promenljivih.
* Naredba UPDATE sa klauzulom LIMIT (čak i ako je determinističko ORDER BY specificirano. Ovo je poznata greška).
* AUTO\_INCREMENT vrednost je automatski generisana i nije prva kolona u primarnom ključu.
* Naredba LOAD DATA INFILE se nije smatrala bezbednom pre verzije 10.0.
* Sistemska tabela je uključena u naredbu. Tabele u bilo kojoj sistemskoj bazi podataka (mysql, information\_schema ili performance\_schema) mogu da skladište različite podatke na različitim serverima, čak i ako su serveri primili iste naredbe.

Ukoliko se evidentiranje u binlog vrši korišćenjem formata STATEMENT, mogu se koristiti samo READ COMMITTED i READ UNCOMMITTED nivoi izolacije. Razlog je taj što sa REPEATABLE READ i SERIALIZABLE redosled izvršenja naredbi može zavisiti od vremena izvršenja naredbi. Ovo se dešava zato što naredba koja koristi zaključavanje može da odloži izvršenje druge naredbe.

Neke nedeterminističke funkcije su bezbedne. To je zato sto taj binarni log sadrži informacije neophodne za njihovo tačno repliciranje. Na primer, datum ili vremenske funkcije su bezbedne jer binarni log čuva vremensku oznaku za svaku naredbu.

Sigurne nedeterminističke funkcije su:

* DATABASE(), SCHEMA(),
* CONNECTION\_ID(),
* LAST\_INSERT\_ID(),
* CURDATE(), CURRENT\_DATE(),
* CURTIME(), CURRENT\_TIME(),
* CURRENT\_TIMESTAMP(), UNIX\_TIMESTAMP(), NOW(),
* UTC\_DATE(), UTC\_TIME(), UTC\_TIMESTAMP(),
* LOCALTIME(), LOCALTIMESTAMP().

### Binarno evidentiranje zasnovano na redovima

Kao što je ranije pomenuto, ako se koristi ROW format, binarni log sadrži promene podataka. Glavna prednost ovog formata je u tome što nema ograničenja kao kod STATEMENT format. Sve naredbe su sigurne sa ROW. Odnosno, pošto se sve modifikacije podataka evidentiraju, nije bitno da li su naredbe koje su proizvele promene bile determinističke ili ne. Bilo koji nivo izolacije mogu se koristiti, uključujući i READ COMMITTED i READ UNCOMMITTED.

Važna prednost ovog formata je prilikom izvršavanja složenih naredbi. Master izvršava SQL naredbu i evidentira modifikaciju u log. Slave server nema potrebe da opet izvršava naredbu, jer to može uključiti dodatni rad kao što je grupisanje ili ređanje redova po redosledu, izvršenje funkcija i tako dalje. Dovoljno je da primeni promenu na redove, sto je dosta jednostavnija operacija.

Međutim, ovaj format poseduje i važan nedostatak. Ako naredba modifikuje mnogo redova, ROW format zahteva mnogo više podataka za upis u log. Ovo može učiniti binarni log prevelikim. Ako se koristi replikacija, master će morati da pošalje mnogo više podataka klijentima.

Takodje, mogu se javiti izuzeci. Neke operacije jesu upisane u log, ali na drugačiji način.

1. Sledece izuzeci se evidentiraju kao naredbe, kao u slučaju korišćenja formata STATEMENT:

* DDL naredbe, kao što su CREATE TABLE i DROP TABLE.
* Naredbe koje implicitno modifikuju tabele u mySQL bazu podataka. Ovo uključuje naredbe kao što su CREATE USER ili GRANT, ali ne i naredbe koje eksplicitno modifikuju sistemske tabele, kao što su INSERT ili UPDATE.
* Naredbe koje uključuju privremene tabele.

1. Ukoliko imamo klauzulu CREATE … SELECT, ona se evidentira na sledeci nacin: prvi deo CREATE TABLE komanda je napisana kao naredba i evidentirana je kao naredba, dok se SELECT evidentira kao redovi.

## Izbor formata binarnog log-a

Izbor formata binarnog log-a je veoma važan. Efikasnost replikacije može biti u velikoj meri pogođena ovim izborom. Obično se podaci koji se koriste upisuju u bazu podataka koriscenjem INSERT ili UPDATE naredbi. Ako je mnogo podataka ubačeno na ovaj način, format STATEMENT bi mogao da rezultira većim binarnim logom. Takođe, on nije efikasan za radna opterećenja koja se sastoje od mnogo naredbi koje menjaju samo jedan red. Prilikom koriscenja formata STATEMENT, programeri treba da budu svesni njegovih važnih ograničenja. S druge strane, ROW format je dosta kompaktniji za same podatke. Kako bi se napravio sto bolji izbor prilikom odabira odgovarajuceg formata, najbitnije je dobro proučiti njihove prednosti i mane, izuzetke, ograničenja.

Da bi se izvršio izbor formata binarnog log-a, potrebno je postaviti vrednost za promenljivu binlog\_format u konfiguracionoj datoteci ili dodati u okviru naredbe --binlog-format. Ako ništa od pomenutog nije navedeno, podrazumevani format je STATEMENT. Pozivanjem pormenljive servera @@binlog\_format, moguće je proveriti koji se format za evidentiranje trenutno koristi. Master i slave ne moraju da koriste isti format evidentiranja.

## Global transaction id - GTID

GTID je uveden u MariaDB 10 i dodaje novi događaj svakoj transakciju u binlogu. Novi događaj je globalni ID koji predstavlja jedinstveni identifikator u svim repliciranim serverima. To olakšava jedinstvenu identifikaciju istih binlog događaja na različitim serverima koji se međusobno repliciraju. Po tome se razlikuje od klasične replikacije.

GTID se sastoji od tri broja, povezana isprekidanom linijom, poput x-y-z. Sta svaki od njih predstavlja objašnjeno je na sledeći način:

* x: Prvi broj, odnosno ID domena
* y: Drugi broj, odnosno ID servera (kao klasična replikacija)
* z: Treći broj, odnosno redni broj (povećava se na svaki događaj)

Korišćenje globalnog ID-a transakcije pruža dve glavne prednosti:

1. ***Laka promena mastera, odnosno povezivanje slave servera i repliciranje podataka sa drugog master servera.***

Slave server pamti globalni ID transakcije poslednje grupe događaja primenjene sa prethodnog mastera. Ovo omogucava slave serveru da bez problema nastavi replikaciju na novom masteru odakle je stao, pošto su globalni ID-ovi transakcija poznati u celoj hijerarhiji replikacije. Ovo nije slučaj kada se koristi standardna replikacija, jer u tom slučaju slave zna samo specifično ime datoteke i koordinate poslednjeg primenjenog događaja prethodnog mastera. Ne postoji jednostavan način da se iz toga pogodi tačno ime datoteke i tacna pozicija na novom.

1. ***Stanje slave servera se snima na način bezbedan od sudara.***

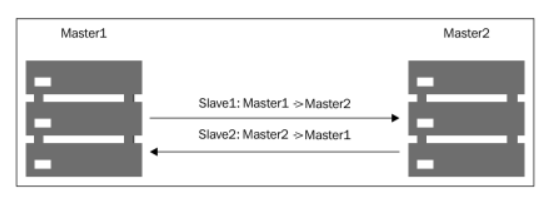
Slave prati svoju trenutnu poziciju (globalni ID transakcije poslednje primenjene transakcije) u sistemskoj tabeli gtid\_slave\_pos. Ako ova tabela koristi transakcioni mehanizam za skladištenje (kao što je InnoDB, koji je podrazumevani), onda se ažuriranja stanja vrše u istoj transakciji kao i ažuriranja podataka. Ta tabela se ažurira svaki put kada se transakcija upiše. Ako se slave sruši, lako je uhvatiti poziciju poslednjeg stanja i videti sa masterom da li se poklapa sa poslednjom transakcijom koju je izvrsio. Ovo nije slučaj za standardne replikacije, gde se stanje beleži u datoteci relay-log.info, koja se ažurira nezavisno od stvarnih promena podataka i ne može lako da se povrati podatke ako se slave server sruši.

## Dual master

Glavni problem koji se javlja kod proste replikacije, gde imamo jedan master, je njegovo otkazivanje. Rešenje tog problema je da se izbegne posedovanje jednog mastera i da se napravi dupla master replikacija.

Dual master replikacija je vrlo jednostavna za implementaciju. To je, u stvari replikacija slave servera na obe strane, pri čemu se u jednom slučaju master ponasa kao slave server za mastera 2, a u drugom master 2 kao slave za master 1.

Sledeći dijagram prikazuje dvostranu replikaciju:



Slika 5: Šema dual mastera

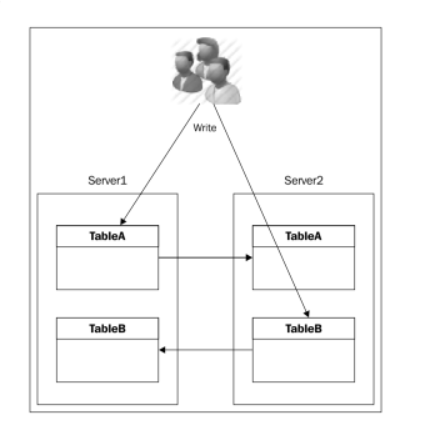
To znači je moguće pisati u obe MariaDB instance u isto vreme. Međutim, to će verovatno dovesti do nedosledosti podataka između dva hosta. Ta vrsta problema može se minimizovati dodavanjem sledećih opcija u MariaDB konfiguracionim fajlovima:

*auto-increment-increment=2*

*auto-increment-offset=1*

Ovo će izbeći kolizije primarnog ključa u istim redovima iste tabele:

* auto-increment-increment: Ovo postavlja broj mastera (ovde 2)
* auto-increment-offset: Ovo postavlja jedinstveni broj (uzimajući server-id je dobro rešenje)



Slika 6: Primer dual mastera

Na prethodnom dijagramu može se videti sledeće:

• Tabela A na serveru 1 se replicira u tabelu A na serveru 2

• Tabela B na serveru2 se replicira na tabelu B na serveru1

U ovom slučaju, ne postoji mogući problem. Međutim, generalno je lakše održavati replikacija koja u potpunosti replicira sve podatke umesto dela podataka.

# Konfigurisanje replikacije

Minimalno okruženje u kome je moguće demonstrirati rad replikacije sastoji se od 2 instance, od kojih je bar jedna master instanca. Zbog toga je najbolje početi sa postavkom master-slave replikacije. Iako je okruženje jednostavno, to neće smetati da objasnimo sve što je potrebno uraditi kao i sve principe koji će važiti i u bilo kojoj drugoj postavci. Jedina razlika je koliko puta je potrebno ponoviti konfigurisanje, kao i vrednost samih parametara.

Instance sa kojima ćemo, bar na početku, raditi su podignute na Ubuntu 16.04 i imaju 192.168.0.122. i 192.168.0.124. ip adrese, respektivno.

## Instaliranje potrebnih alata i servisa

Da bismo došli do dela vezanog za samu bazu podataka i replikaciju, potrebno je instalirati nekoliko osnovnih alata i pokrenuti servise. Zato počinjemo sa instaliranjem MariaDB servera pomoću komande:

apt-get install mariadb-server -y

Kada je to urađeno, pokrećemo MariaDB servise i omogućavamo da se oni pokreću pri pokretanju same instance:

systemctl start mysql  
systemctl enable mysql

Podrazumevano, MariaDB nije “sigurna”. Zato je treba osigurati i to pomoću:

mysql\_secure\_installation

To će pokrenuti niz pitanja no koje treba odgovoriti na sledeći način:

Set root password? [Y/n] n  
Remove anonymous users? [Y/n] y  
Disallow root login remotely? [Y/n] y  
Remove test database and access to it? [Y/n] y  
Reload privilege tables now? [Y/n] y

## Konfigurisanje master instance

Kada kreiramo okruženje za replikaciju prva stvar koju je potrebno uraditi je, naravno, kreiranje bar jedne master instance. Master je regularna MariaDB instanca koja ima jedinstveni ID i vodi računa o binarnom logu.

Pre svega, potrebno je postaviti ID servera. ID mora biti jednistven i to je celobrojna vrednost od četiri bajta sa minimalnom vrednošću 1. Ako server ID nije postavljen, ili je 0, replikacija će biti onemogućena. Takođe, binarni log mora biti uključen na master serveru. To je neophodno u cilju “snimanja” događaja koje će biti poslati slave instancama.

Master konfigurišemo tako što se u njegov ***my.cnf*** fajl dodaju, odnosno, aktiviraju neke opcije. To izgleda ovako:

[mysqld]  
bind-address = 192.168.0.101  
server\_id=1  
log-basename=master  
log-bin=/var/log/mysql/mariadb-bin

, gde je:

* server-id: Jedinstveni ID instance
* bind-address: IP adresa koju će osluškivati ova instanca. Podrazumevano je 127.0.0.1, dok 0.0.0.0 znači vezivanje svih dostupnih interfejsa
* log-bin: Mesto na kome se čuva log

Pored ovih osnvnih, tu su i sledeći parametri:

expire\_logs\_days=10

sync\_binlog = 1

slave\_compressed\_protocol

transaction-isolation = READ COMMITTED

binlog\_format = row

, koji znače:

1. ***expire\_logs\_day***: Broj dana koliko se log fajl zadržava. Služi tome da spreči da log postane toliko veliki da zauzme svu dostupnu memoriju. Nakon isteka broja dana definisanog ovim parametrom log se resetuje.
2. sync\_binlog: Podešavanje sinhronizacije log fajla nakon svakog upisa. Kada je postavljen na 1 sinhronizacija je aktivirana. To predstavlja najsigurniju, ali i najsporiju opciju.
3. ***slave\_compressed\_protocol:*** Za uključivanje kompresije, ali samo u slučaju da je aktivirana i na slave instanci. Zahteva dodatni CPU, ali i ubrzava replikaciju.
4. ***binlog\_format***: Pomoću ove opcije bira se koji binlog format se koristi: row, statement ili mixed. Za postizanje sigurne replikacije najbolji je row format.

Kada je fajl konfigurisan potrebno je restartovati MariaDB servis pomoću:

systemctl restart mysql

Sada je naša master instanca spremna za pružanje usluga tako da je naredni korak kreiranje naloga, odnosno korisnika za replikaciju. Ovaj nalog će imati samo prava za replikaciju. Naravno, moguće je dodati i ostala prava ovom korisniku, ili koristiti postojećeg, ali iz sigurnosnih razloga to nije preporučljivo.

Dakle, na master instanci kreiramo korisnika kome omogućavamo da se konektuje sa slave instanci. Prvo se logujemo na MariaDB shell:

mysql -u root -p

i radimo:

MariaDB [(none)]> create user 'replication'@'192.168.0.124' identified by 'password';

, posle čega podešavamo privilegije za replikaciju za sve baze u okviru sistema:

MariaDB [(none)]> grant replication slave on \*.\* to 'replication'@'192.168.0.124';

Na kraju, radimo restart privilegija:

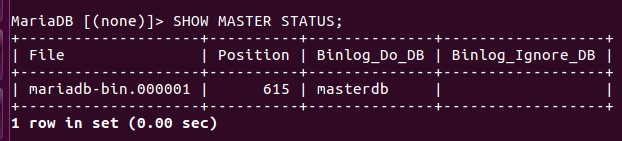
MariaDB [(none)]> FLUSH PRIVILEGES;

Ako neku od baza u sistemu ne želimo da repliciramo onda nećemo omogućiti permisije za nju.

U ovom trenutku bi trebalo da možemo da dobijemo status master instance pomoću:

MariaDB [(none)]> SHOW MASTER STATUS;

Ako je sve u redu, dobićemo izlaz poput sledećeg:



Slika 7 - Master status

Kolona *file* odgovara binarnom logu dostupnom u *logbin* direktorijumu, dok sledeća kolona pokazuje trenutnu poziciju u okviru tog fajla.

## Konfigurisanje slave instance

Slično kao i kod prethodnog primera počinjemo podešavanjem opcija u *my.cnf* fajlu:

[mysqld]  
bind-address = 192.168.0.122  
server-id = 2  
replicate-do-db=masterdb

binlog\_format = row

read\_only

Ovde je interesantno pomenuti opciju *read\_only* pomoću koje možemo biti sigurni da slave neće imati uticaj na promene na master-u.

Sada je potrebno konektovati se na master instancu kako bi sam proces replikacije mogao da počne. Da bismo to uradili potrebne su nam binlog koordinate, koje smo videli pozivanjem *SHOW MASTER STAUTUS* komande. Nju možemo pozvati opet odavde i dobiti vrednosti koordinata. Njih ćemo iskoristiti u okviru *CHANGE MASTER TO* naredbe. To izgleda ovako:

MariaDB [(none)]> CHANGE MASTER TO

-> MASTER\_HOST = '192.168.0.122,

-> MASTER\_USER = 'slave\_user',

-> MASTER\_PASSWORD=’passsword’

, -> MASTER\_LOG\_FILE = 'binlog.000001',

-> MASTER\_LOG\_POS = 615;

U ovom trenutku slave zna sve što mu je potrebno da zna o replikaciji, sve neophodne dozvole su postavljene i obe instance su u stanju rada. Da bismo započeli replikaciju dovoljno je da pokrenemo *START SLAVE* komandu:

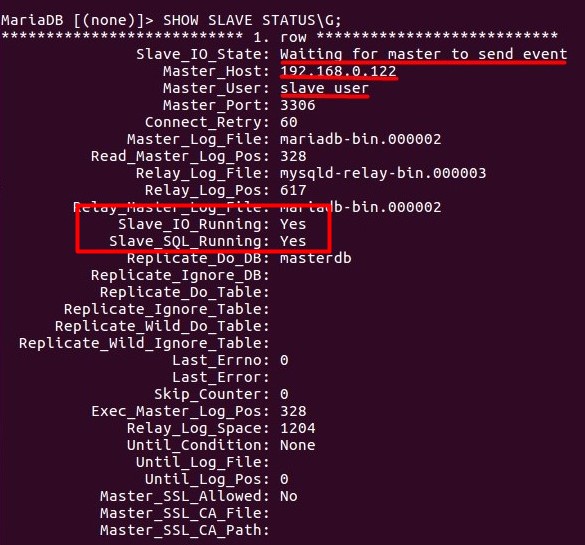
MariaDB [(none)]> START SLAVE;

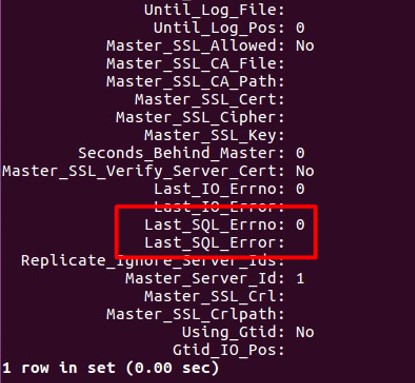
Query OK, 0 rows affected (0.02 sec)

Status slave čvora, kao i to da li ispravno radi proveramo pomoću:

MariaDB [(none)]> SHOW SLAVE STATUS;

, pri čemu dobijamo:





Slika 8 - Slave status

Ovde vidimo pregršt zanimljivih informacija, od kojih su najbitnije:

* Master\_host i master\_user
* Slave\_IO\_Running and Slave\_SQL\_Running: Ako su oba postavljena na yes onda je replikacija uspešna. Ako je bar jedno od njih no, postoji problem
* Last\_Errno and Last\_Error: Ukazuju na probleme i prikazuju upit koji nije mogao da se izvrši

# Primeri replikacije podataka

Ovde ćemo prikazati nekoliko primera kopiranja, repliciranja podataka u prethodno postavljenom klasteru. Proverom stanja određenih baza i tabela dokazaćemo da se podaci upisani na jednoj mogu videti na nekoj od instanci predviđenih za to. Kao što to i najviše ima smisla, počinjemo sa replicarenjem podataka sa master na slave repliku.

## Master – slave replikacija

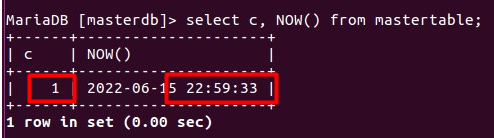
Pre svega, ulogujemo se na MariaDB master čvor i kreiramo bazu pomoću:

MariaDB [(none)]> create database masterdb;

Nakon toga, kreiramo test tabelu u koju dodajemo podatak:

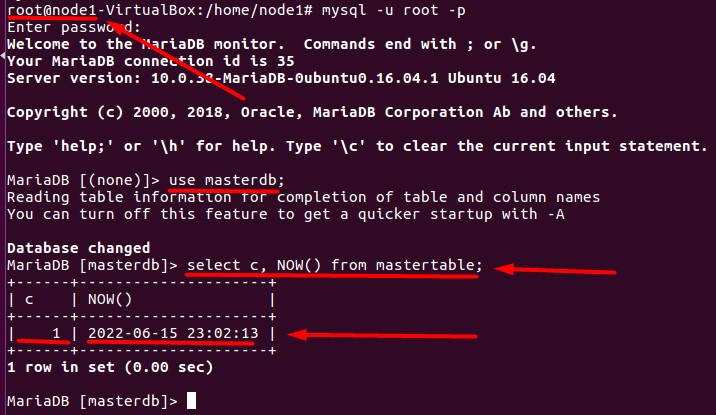
MariaDB [(none)]> use masterdb;  
MariaDB [masterdb]> create table mastertable (c int);  
MariaDB [masterdb]> insert into mastertable (c) values (1);

Kada proverimo stanje tabele, dobijamo:



Slika 9 - Stanje tabele na master instanci

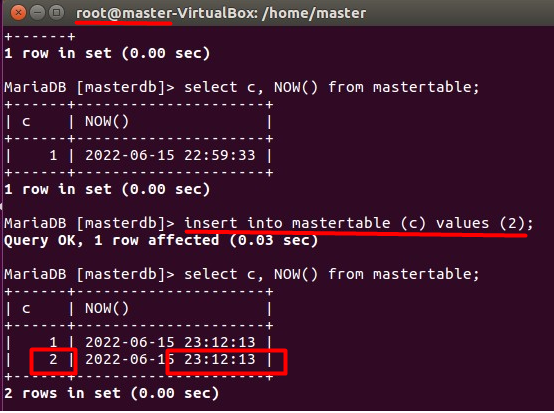
Sada se logujemo na slave instancu *Node1* i radimo čitanje iz te iste tabele. Ako je sve kako treba i replikacija uspešna, trebalo bi da vidimo *masterdb* bazu, *mastertable* tabelu i isti podatak (1) upisan u nju. Pokrećemo komandu za čitanje pri čemu dobijamo:



Slika 10 - Podaci na slave instanci Node1

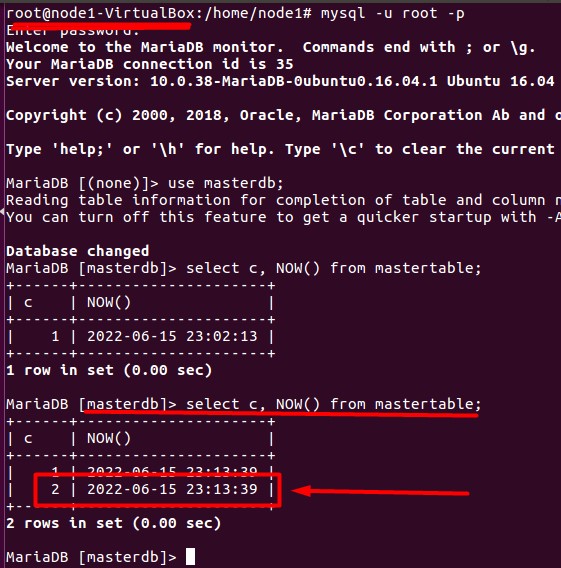
Vidimo da je upisani podatak zaista tu i vidimo da on pročitan nakon što je izvršen upis na master (vreme kao dokaz za to).

Nastavljamo sa proverom da li replikacija uspešno radi, tako što se vraćamo na master i upisujemo još jedan podatak u tabelu:



Slika 11 - Upis novog podatka na master

Kao i malopre, proveravamo da li podatak postoji na Node1 čvoru:



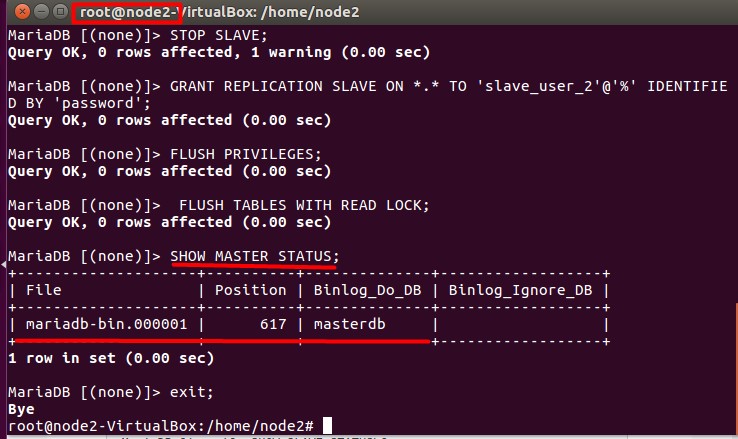
Slika 12 - Podaci na Node1 nakon drugog upisa

i potvrđujemo da je zaista došlo do replikacije.

## Promena master-a

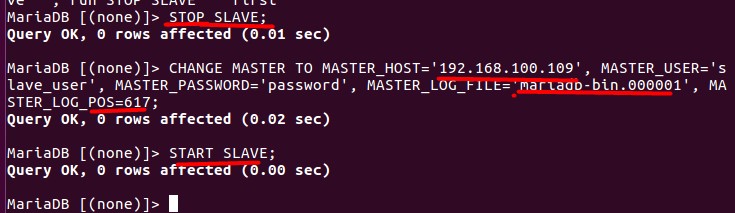
Pri konfigurisanju klastera često dolazi do situacije kada želimo da instanca koja je već konfigurisana da osluškuje jedan master prestane sa tim i počne da replicira podatke sa neke drugog master-a. Srećom, u MariaDB je to lako uraditi i proces se svodi na korišćenje već objašenjene *CHANGE MASTER TO* komande, samo sa drugačijim parametrima.

Da bismo to demostrirali, moramo pre svega da kreiramo još jednu instancu koja će biti master broj 2. koju nazivamo Node2. Proces kreiranja i konfigurasanja je isti kao i za Master čvor tako da ga nećemo opet objašnjavati. Samo je bitno napomenuti da je potrebno pokrenuti *SHOW MASTER STATUS* u cilju dobijanja koordinata log fajla.



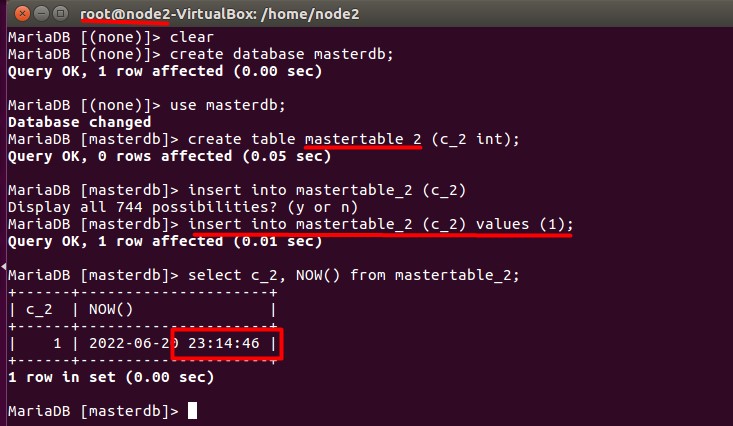
Slika 13 - Status 2. master instance Node2

Kada je to urađeno, na Node1 čvoru radimo:



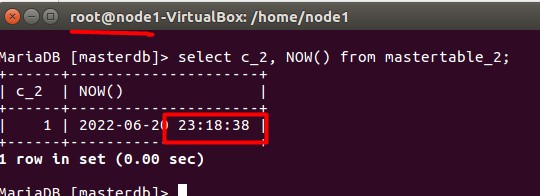
Slika 14 - Promena mastera na Node1

Za sada, sve deluje u redu, ali još uvek nemamo dokaz da Node1 sada replicira instancu Node2. Vreme je da to saznamo tako što ćemo upisati podatke na novi master:



Slika 15 - Upis podataka na novu master instancu Node2

Kao i do sada, sledi čitanje podataka na slave čvoru:

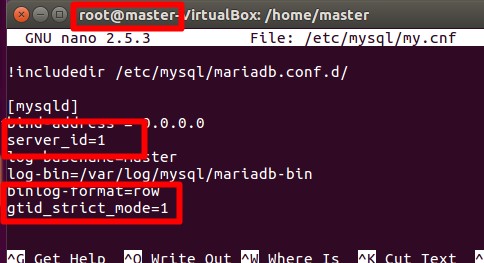


Slika 16 - Podaci na Node1 slave instanci

, čime potvrđujemo da je došlo do repliciranja podataka sa Node1 na Node2.

## GTID replikacija

Konfigurisanje klastera za izvršenje GTID replikacije podrazumeva i pripremanja svih instanci za to. To dalje znači da će biti nekih promena pre svega u konfiguracionim fajlovima, u načinu na koji dobijamo koordinate log-a itd. Počinjemo od mastera, gde konfiguracioni fajl izgleda u velikoj meri isto kao i do sada, sa jednom jako bitnom promenom:



Slika - Konfiguracioni fajl za Master sa GTID

Kao što je to naznačeno u primeru, to je opcija *gtid\_strict\_mode=1* , koja nam dozvoljava da imamo isti log za sve čvorove u klasteru koji koriste GTID.  Nakon toga, kao i kod klasične replikacije, kreiramo nalog za replikaciju i obezbeđujemo sva potrebna prava:

MariaDB [(none)]> create user 'gtid\_replication'@'192.168.0.124’

identified by 'password';

MariaDB [(none)]> grant replication slave on \*.\* to

'gtid\_replication'@'192.168.0.124;

MariaDB [(none)]> FLUSH PRIVILEGES;

, i proveravamo trenutnu GTID poziciju pomoću komande:

MariaDB [(none)]> SELECT @@GLOBAL.gtid\_current\_pos;

+-----------------------------+

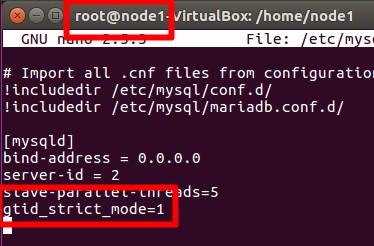
| @@GLOBAL.gtid\_current\_pos   |

+-----------------------------+

| 0-1-10         |

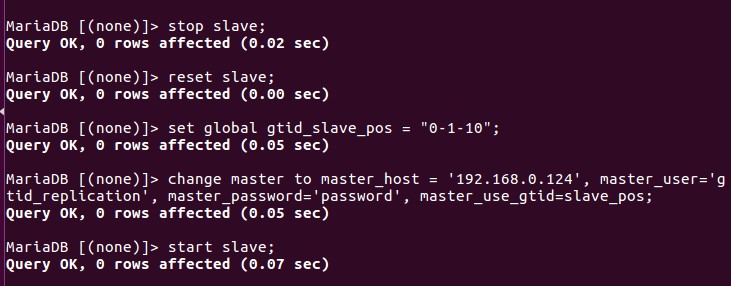
+-----------------------------+

Sledi nekoliko koraka koji se odnose na *slave* instancu. Kao i za master, prvi od njih je vezan za konfiguracioni fajl:



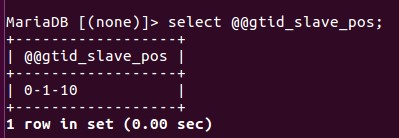
Slika - Konfiguracioni fajl za slave

Nakon toga se ulogujemo na MariaDB shell na instanci i radimo sledeći niz komandi:



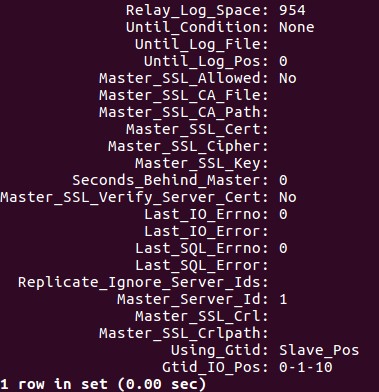
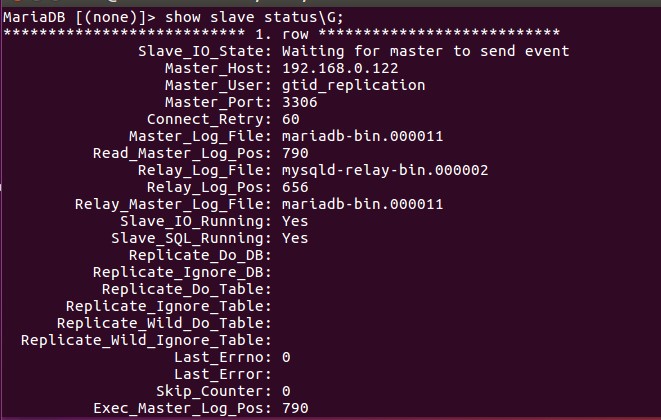
Slika - Povezivanje slave-a na master

Ako je sve u redu, možemo da proverimo trenutnu poziciju slave instance:



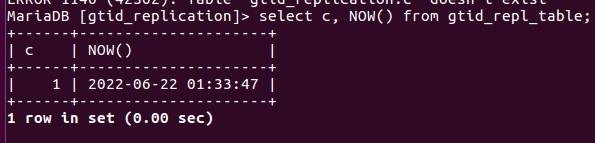
Slika - Pozicija slave instance

, kao i status:



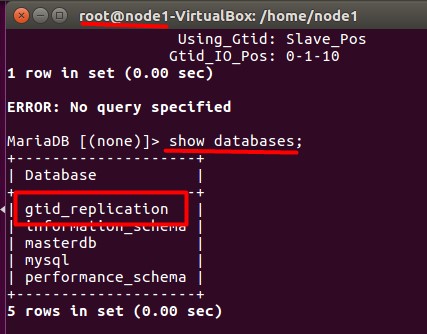
Slika - Status slave instance

Poslednji korak je potvrditi da zaista dolazi do replikacije. Zato na *master-u* radimo kreiranje baze *“gtid\_replication”,* kreiranje tabele *“gtid\_repl\_table”* i upis podatka “1” u nju:

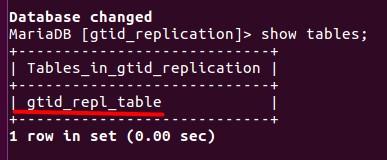


Slika - Dodavanje podataka na master

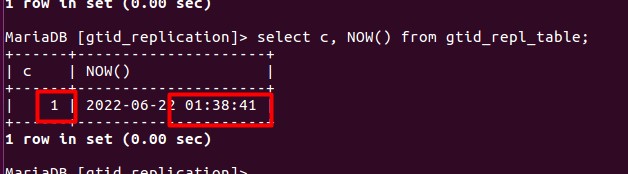
Nakon toga proveramo da li podaci postoje na *slave-*u i to u istom redosledu, prvo bazu, zatim tabelu i na kraju podatke u njoj:



Slika - Provera baze na slave instanci

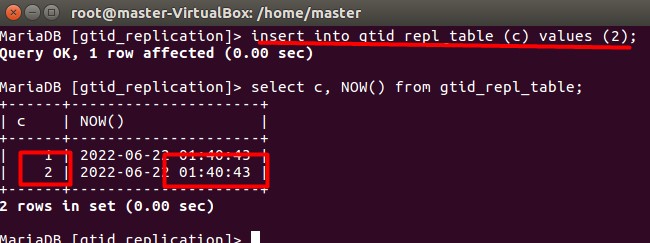


Slika - Provere tabele na slave instanci



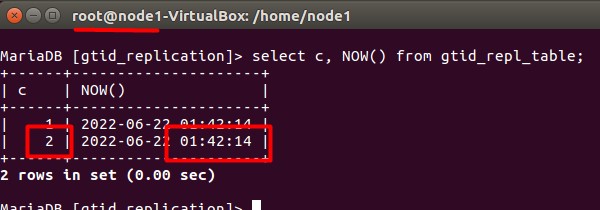
Slika - Provera podataka u tabeli na slave instanci

Sa slika vidimo da je replikacija bila uspešna i da se na *slave* instanci nalaze svi potrebni podaci. Zbog dodatne sigurnosti izvršićemo upis još jednog podataka i proveriti da li se i on replicira. Dakle, prvo upis na master:



Slika - Dodavanje novog podatka na master-u

, a onda čitanje na slave-u:



Slika - Provera postojanja "novih" podataka na slave-u

Vidimo da je i ovaj podatak uspešno repliciran, što znači da je rešenje poptuno i da funkcioniše kako treba.

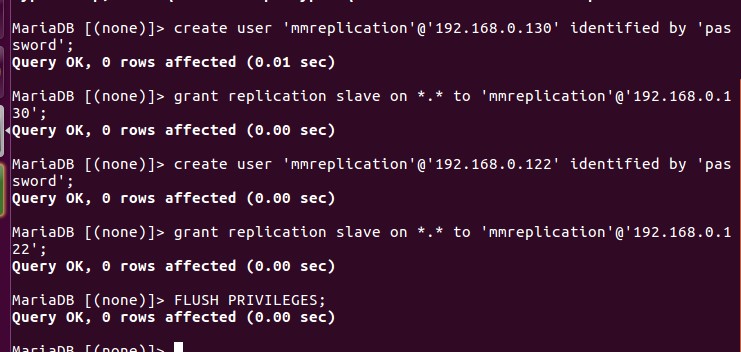
## Dual-master replikacija

U teorijskom delu objasnili smo sta podrazumeva multimaster replikacija i koje su njene prednosti. Ovde ćemo videti kako da konfigurišemo klaster tako da instancama damo mogućnost da se istovremeno ponašaju kao slave i master. Za demonstraciju biće nam potrebna jedna nova instanca. Kao i prethodne 2 i onda je podignuta na Ubuntu mašini, sa adresom 192.168.0.130. Njeno ime je *Node2* , mada ćemo je zbog lakšeg razumevanja ovde zvati Master2 instanca.

Kao što je to slučaj kod kreiranja “standardnog” okruženja, počinjemo sa konfiguracionim fajlovima, ali pošto u njima nema promene u odnosu na prethodne primere nećemo ih prikazati ovde. Nakon toga sledi kreiranje naloga za replikaciju koji će nam omogućiti:

* replikaciju sa Master2 instance na Master1 instancu
* replikaciju sa Master1 instance na Master2  instancu.

Na Master1 kreiramo naloge za replikaciju u oba smera i omogućavamo potrebne dozvole:



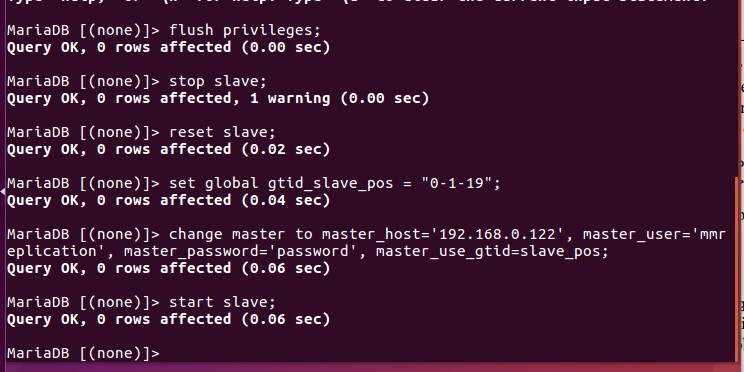
Slika - Konfigurisanje master instance - kreiranje naloga i dozvola

Takođe, potrebne su nam i vrednosti koordinata log-a koje ćemo upotrebiti posle:



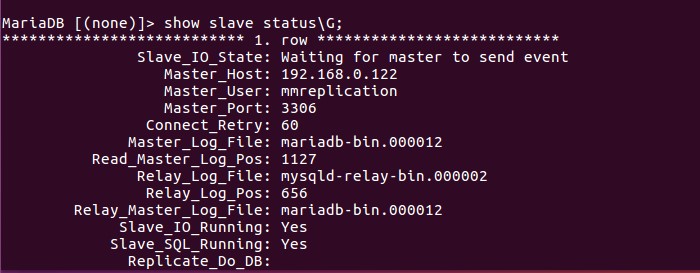
Slika - Vrednost log koordinata na masteru

Sada konfigurišemo Master2 da bude slave za instancu Master1 i to na sledeći način:



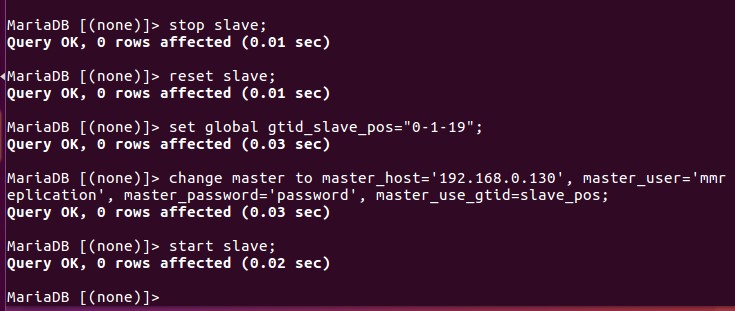
Slika - Konfigurisanje Master2 instance

Kao potvrdu da je to prošlo uspešno proveramo status slave-a na Master2 čvoru:



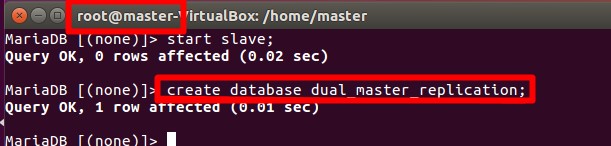
Slika - Provera statusas Master2 instance

Prelazimo na postavljanje slave repliciranja u drugom smeru. Prikupljamo vrednosti koordinate log-a, ovog puta na Master2 instanci, konfigurišemo i startujemo slave na Master1 čvoru:



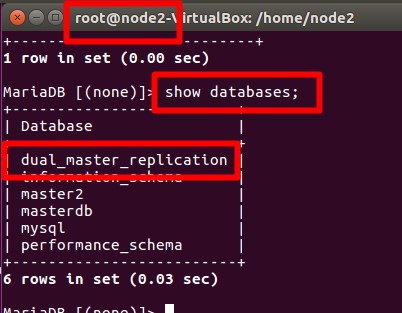
Slika - Startovanje slave-a na Master1 čvoru

Kao i kod prethodnog primera, moramo da potvrdimo da replikacija zaista radi kreiranjem baze, tabele i dodavanjem podataka. Razlika je u tome što ovog puta, radimo upis na bilo koju od 2 instance i podatke bi trebalo da vidimo na drugoj instanci iz klastera. Počinjemo kreiranjem baze na Master1:



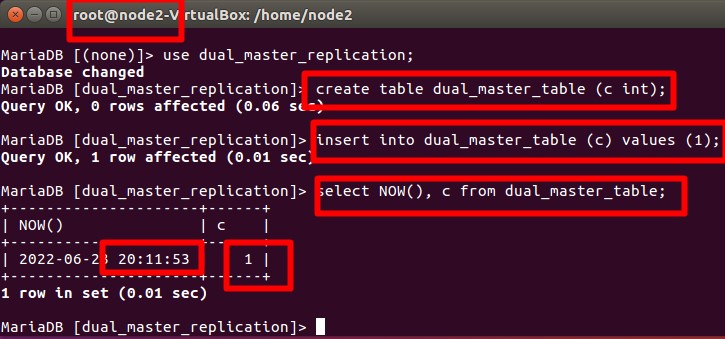
Slika - Kreiranje baze na Master1

Sada prelazimo na Master2 i pozivamo komandu za dobijanje svih baza. Ako je sve prošlo kako treba, trebalo bi da vidimo bazu koja se zove “*dual\_master\_replication”* :



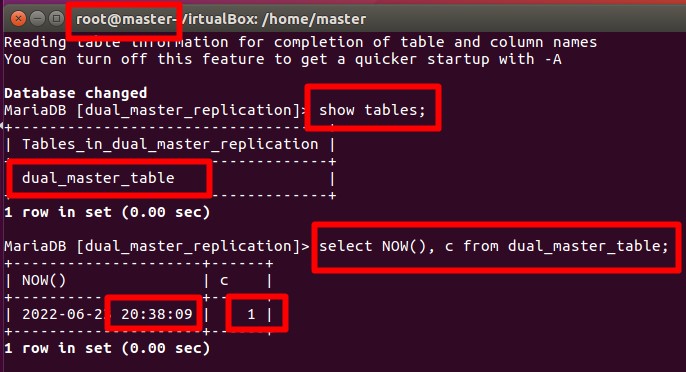
Slika - Provera baze na Master2

Vidimo da baza zaista postoji, čime smo potvrdili ispravnost našeg rešenja do pola, odnosno znamo da replikacija u jednom smeru (Master1 -> Master2) zaista radi. Sada ostaje da se proveri drugi smer. Da bismo to učinili radimo kreiranje tabele i upis podataka na Master2 instanci. Kreiraćemo tabelu koja se zove *“dual\_master\_table”* i upisujemo vrednost “1” :



Slika - Dodavanje podataka na Master2

Upis je prošao kako treba, što znači da sada proveramo da li tabela i podatak koji smo dodali postoji na Master1 čvoru. To ćemo uraditi pomoću *SHOW TABLES i SELECT* naredbi:



Slika - Čitanje podataka na Master1 instanci

Tabela *“dual\_master\_table”* je uspešno replicirana, kao i podatak koji je u nju upisan. To znači da replikacija radi i u smeru Master2 -> Master1, odnosno da je Dual Master klaster uspešno konfigurisan.

# Zaključak

Danas, sve je veći broj aplikacija koje skladište podatke u baze podataka, repliciraju svoje baze na više mašina kako bi se osigurale dostupnost i trajnost podataka. Činjenica je da replikacija nije baš jednostavan proces, ali ako se primeni u pravim okolnostima može biti izvanredno rešenje za bolji razvoj aplikacije, smanjenje opterećenja, poboljsanje efikasnost, postizanje visoke dostupnost za poboljšanje performansi i bolje iskustvo za korisnike.

Glavni mehanizam koji se koristi u replikaciji je binarni log. Replike čitaju binarni log sa svakog master servera da bi pristupile podacima za repliciranje. Izbor formata binarnog log-a je veoma važan. Efikasnost replikacije može biti u velikoj meri pogođena ovim izborom. Kako bi se napravio sto bolji izbor prilikom odabira odgovarajuceg formata, najbitnije je dobro prouciti njihove prednosti i mane, izuzetke, ograničenja.

Bilo da je konfigurisana sa asinhronom master - slave replikacijom radi jednostavnosti i performansi, sinhronom radi dostupnosti i doslednosti, ili nešto između, MariaDB ima i jedno i drugo, i fleksibilnost i sposobnost ispunjavanja složenih zahteva.

Još jedna bitna novina kod novijih verzija MariaDB je GTID. Njegovom upotrebom postiže se još bolja i lakša replikacija. Slave serveri su zaštićeni od sudara, odnosno ukoliko dodje do pada slave servera, moguce je lako nastaviti od pozicije poslednje repliciranog podatka. Takođe, moguća je i laka promena mastera u slučaju otkaza ili njegovog preopterećenja.

Prednosti replikacije mogu da primete i klijenti koji koriste replicirane podatke, jer im mogu lokalno pristupiti umesto povezivanja na udaljene baze podataka preko mreže. Ako se desi kojim slucajem da je lokalna kopija nedostupna, klijenti mogu i dalje pristupiti udaljenoj kopiji podataka. Često se dešava da postoje kompanije u kojima su korisnici isključeni tokom dana, rade koriscenjem slave servera, a zatim se na kraju radnog vremena vrši potrebno ažuriranje naloga/zaliha i drugih informacija.

Međutim, ove prednosti ne dolaze bez troškova. Replikacija zahteva više prostora za skladištenje i ažuriranje repliciranih podataka. Takođe, poseduje i neke nedostatke koji potiču od loših opštih praksi upravljanja podacima.

# Reference

[1] Federico Razzoli , “Mastering MariaDB”, Packt Publishing, 2014

[2] Pierre Mavro , “MariaDB High Performance”, Packt Publishing, 2014

[3] IRussell J.T. Dyer, “Learning MySQL and MariaDB\_ Heading in the Right Direction with MySQL and MariaDB”, O'Reilly Media, 2015

[4] MariaDB Replication Overview for SQL Server Users - <https://mariadb.com/kb/en/mariadb-replication-overview-for-sql-server-users/>

[5] Parallel Replication - <https://mariadb.com/kb/en/parallel-replication/>

[6] MariaDB Replication - <https://mariadb.com/kb/en/standard-replication/>