



UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET



MySQL InnoDB CLUSTER

Seminarski rad

Predmet: Sistemi za upravljanje bazama podataka

Student:

Marko Kocić, br. ind. 1478

Mentor:

Doc. dr Aleksandar Stanimirović

Niš, jun 2023. godina

Sadržaj

UVOD.....	3
PREGLED <i>InnoDB</i> KLASTERA.....	4
Grupna replikacija.....	6
Tehnike replikacije.....	7
Servisi grupne replikacije.....	9
<i>Single-primary</i> i <i>multi-primary</i> režim.....	10
ZAHTJEVI <i>InnoDB</i> KLASTERA.....	12
OGRANIČENJA <i>InnoDB</i> KLASTERA.....	13
RAD SA <i>InnoDB</i> KLASTEROM UZ TEORIJSKA OBJAŠNJENJA.....	14
Priprema okruženja.....	14
Konfiguracija, administracija i <i>deployment</i> <i>InnoDB</i> klastera.....	17
<i>MySQL Router</i>	28
Otkaz jednog od <i>MySQL</i> servera.....	33
ZAKLJUČAK.....	37
LITERATURA.....	38

1. UVOD

U okviru trećeg seminarskog rada iz predmeta „Sistemi za upravljanje bazama podataka“ opisaću teorijski i demonstrirati praktičnim primerima jedno od klaster rešenja *MySQL* baze podataka – *InnoDB* klaster.

Klaster u kontekstu baze podataka je skup *host*-ova koji skladišti podatke na više fizičkih lokacija i samim tim predstavlja distribuiranu bazu podataka. Klasterizacijom baze podataka dobijamo bolje performanse, skalabilnost, visoku dostupnost, otpornost na greške i efikasnije korišćenje resursa. Klaster je dobro rešenje za aplikacije visoke potražnje, sa velikim brojem korisnika i ogromnom količinom podataka, kao i kada želimo robusno i pouzdano okruženje.

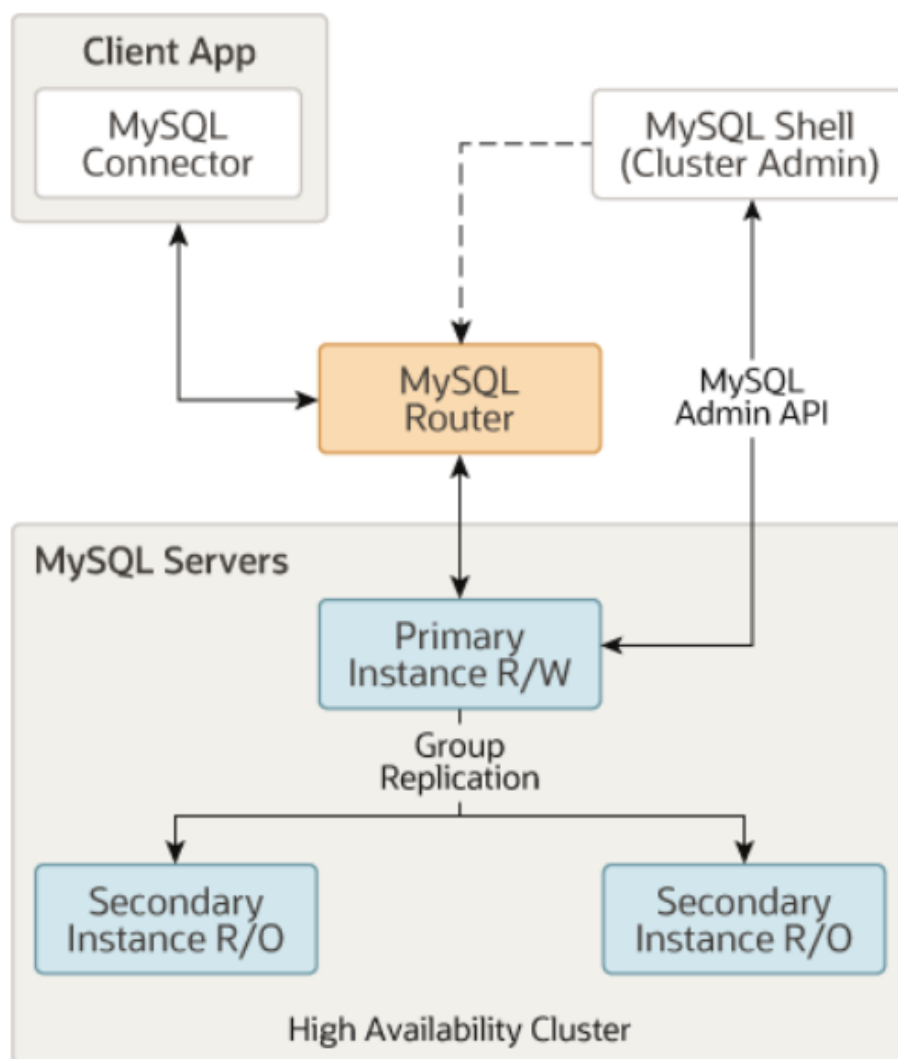
MySQL Cluster je distribuirana baza podataka koja kombinuje linearnu skalabilnost i visoku dostupnost. Pruža *in-memory* pristup u relanom vremenu sa transakcionom konzistentnošću u particionisanim i distribuiranim skupovima podataka. Dizajniran je za kritične aplikacije. Generalno gledano, to je *shared-nothing* tehnologija sa *auto-sharding DBMS*-om.

2. PREGLED *InnoDB* KLASTERA

InnoDB klaster kombinuje *MySQL* tehnologije radi *deploy*-ovanja i administracije kompletno integrisanog, visoko dostupnog, lako skalirajućeg i otpornog na greške *MySQL* rešenja. Klaster se sastoji od najmanje 3 (zbog neophodnog kvoruma – više od 50% za postizanje dogovora) i najviše 9 instanci *MySQL* servera. Tehnologije koje se koriste su sledeće:

- *MySQL Shell* – napredni klijent i kod editor za administraciju klastera;
- *MySQL Server* i grupna replikacija pomoću kojih skup instanci pruža visoku dostupnost;
- *MySQL Router* – *middleware* koji omogućava transparentno rutiranje između aplikacije i *InnoDB* klastera.

Na slici ispod je prikazana shema *InnoDB* klastera.



Slika 2.1. Shema *InnoDB* klastera u *single-primary mode*-u

Koristi se grupna replikacija, tehnologija koja pruža mehanizam za replikaciju podataka unutar klastera sa ugrađenom tolerancijom na greške. Svaka *MySQL* server instanca *InnoDB* klastera pokreće grupnu replikaciju. *InnoDB* klaster obično radi u *single-primary mode*-u u kojem postoji samo jedna primarna instanca za upis i čitanje i više sekundarnih instanci koje služe samo za čitanje. Usled neočekivane greške, klaster se automatski rekonfiguriše. Primera radi, u slučaju otkaza primarne instance, nakon izbora, jedna od sekundarnih automatski preuzima ulogu primarne (od tog trenutka se ista koristi i za čitanje i za pisanje). Režim koji takođe može biti konfigurisan je *multi-primary mode* u kojem su sve instance primarne - koriste se i za upis i za čitanje. S obzirom da se može desiti jednovremeni upis na više servera, sam sistem nakon ovih akcija mora biti u konzistentnom stanju – instance se moraju dogovoriti oko stanja. Ovaj režim obično koriste napredni korisnici.

InnoDB klaster podržava *MySQL Clone* koji omogućava jednostavno snabdevanje podacima instanci. Ovaj *plugin* (dostupan od *MySQL* verzije 8.0.17) dozvoljava kloniranje podataka kako lokalno tako i sa udaljene instance *MySQL* servera. Klonirani podaci su fizički *snapshot* skladištenih podataka u *InnoDB*-u koji uključuju sheme, tabele, metapodatke,... Sastoje se od potpuno funkcionalnog direktorijuma podataka. Za razliku od prošlosti, gde se obezbeđivanje instanci pre samog priključivanja klasteru radilo ručno (primera radi kopiranjem datoteka, prenosom transakcija), danas se proces dodavanja sinhronizovane instance u klaster radi automatski.

InnoDB klaster je čvrsto integrisan sa *MySQL Router*-om kojem se pristupa pomoću *AdminAPI*-a i vrši dodatna administracija. *MySQL Router* se automatski konfigurise na osnovu *InnoDB* klastera u procesu koji se naziva *bootstrapping* i samim tim nema potrebe za ručnom konfiguracijom. Transparentno povezuje klijentske aplikacije sa klasterom obezbeđujući rutiranje i balansiranje opterećenja. Automatski detektuje promene nastale unutar klastera i ceo sistem nastavlja sa skladnim funkcionisanjem. Primera radi, ukoliko je reč o *single-primary mode*-u i došlo je do otkaza primarne instance i zamene iste jednom od sekundarnih (sekundarna u ovom trenutku postaje primarna), naredni *write* zahtev, *MySQL Router* šalje novoj primarnoj instanci.

Klasterom se upravlja pomoću *AdminAPI* koji je sastavni deo *MySQL Shell*-a. Dostupan je u *JavaScript* i *Python* tehnologijama. Korišćenjem istog vrši se administracija i nadgledanje *deployment*-a i izbegava se ručna konfiguracija više instanci, kao i direktan rad sa grupnom replikacijom u klasteru.

Pored skinutog i instaliranog *MySQL Shell*-a, za rad sa *InnoDB* klasterom neophodno je imati i *host*-ove sa instaliranim *MySQL Server* instancama i instaliran *MySQL Router*.

2.1. Grupna replikacija

MySQL grupna replikacija (*plugin MySQL* servera) omogućava elastično i visoko-dostupno rešenje otporno na greške. Postoji ugrađeni servis koji pruža jedinstveni pogled na klaster od strane svih članova u realnom vremenu. S obzirom na činjenicu da serveri mogu napustiti klaster ili se pak pridružiti istom, pogled se ažurira u skladu sa novonastalom promenom. Primera radi, ukoliko server neočekivano napusti grupu, mehanizam za otkrivanje grešaka detektuje događaj i obaveštava sve članove grupe o promeni pogleda. Ovo se dešava automatski.

Grupna replikacija garantuje da je baza podataka stalno dostupna. Ukoliko je prilikom otkaza servera neki klijent bio direktno povezan na isti, mora se prebaciti na drugi server samostalno jer grupna replikacija nema ugrađeni metod za redirekciju (morali bismo ga sami napisati). Zbog toga se klijenti u praksi ne povezuju direktno na server, već se koristi neki tip *middleware-a* (*load balancer*) - *MySQL Router*.

Najčešći način stvaranja sistema otpornog na greške je kreiranje replika, odnosno redundantnih komponenti tako da iako dođe do greške na jednoj komponenti, sistem nastavlja normalno sa radom. S obzirom na činjenicu da je sve trgovina (*everything is trade-off*), repliciranje baze podataka pored dodatnih resursa, zahteva i dodatnu administraciju i održavanje, ali je takođe potrebno rešiti i klasične probleme koji postoje kod distribuiranih sistema.

Krajnji cilj je spojiti logiku same baze podataka i replikacije podataka sa logikom jednostavne i konzistentne koordinacije većeg broja servera. Tačnije, svi serveri (koji nisu maliciozni) se moraju složiti oko stanja sistema i podataka na istom, odnosno oko svake načinjene promene. Zapravo serveri postižu dogovor o svakoj tranziciji stanja baze podataka (svi napreduju kao jedinstvena baza podataka). Da bi transakcija bila izvršena, većina grupe se mora složiti oko redosleda iste u globalnom nizu transakcija. Odluku o izvršenju (odnosno o neizvršenju) transakcije donosi svaki server pojedinačno na osnovu ranije postignutog dogovora (sve odluke su iste).

Ako kojim slučajem dođe do particionisanja mreže (*network partitioning*) – greška na mreži prilikom koje su članovi grupe podeljeni u više grupa koje međusobno ne mogu komunicirati i ne može se postići dogovor (nema većine), sistem ne nastavlja sa radom dok problem ne bude otklonjen. Particionisanje mreže može dovesti do *split brain-a*, tačnije nekonzistentnog stanja klastera u kojem čvorovi na nejedinstveni način rukuju ulazno-izlaznim operacijama – može doći do oštećenja podataka ako ne prekinemo sa radom klastera (*downtime* - manje zlo). Inače, postoji ugrađeni automatski mehanizam zaštite od *split brain-a* (*quorum* - više od 50% servera se mora dogovoriti oko stanja) pri čemu klaster ne prestaje sa radom (sve dok imamo većinu – primera radi 2 od 3 servera je ispravno).

Iza svega ovoga stoje protokoli grupne komunikacije sistema (*GCS protocols*) koji pružaju mehanizam za otkrivanje grešaka, servis članstva u grupi, bezbednu isporuku poruka u ispravnom redosledu. U samoj srži implementacije leži *Paxos* algoritam koji omogućava grupnu komunikaciju.

Neki od slučajeva korišćenja grupne replikacije su:

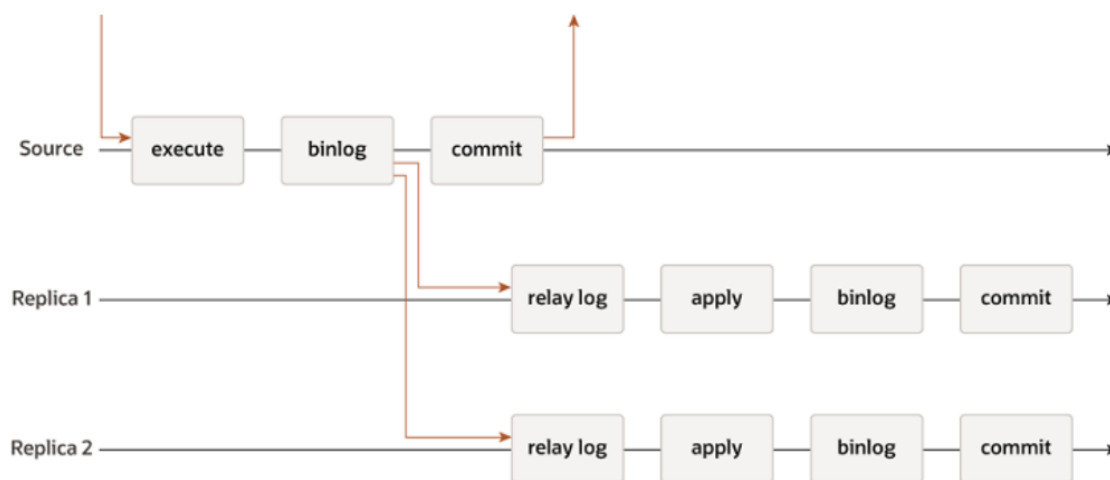
- elastična replikacija – česta dinamička promena broja servera u klasteru (*cloud* rešenja);
- visoko-dostupni *shard*-ovi – *sharding* je popularan pristup kod skaliranja operacija pisanja;
- autonomni sistemi – *deploy*-ovanje čiste *MySQL* grupne replikacije.

2.1.1. Tehnike replikacije

Dve tehnike koje ćemo proučiti u ovom delu rada su tradicionalna replikacija i grupna replikacija. U nastavku objašnjavamo kako grupna replikacija radi „pod haubom“, šta je sve neophodno i koje su razlike u odnosu na klasičnu asinhronu *MySQL* replikaciju.

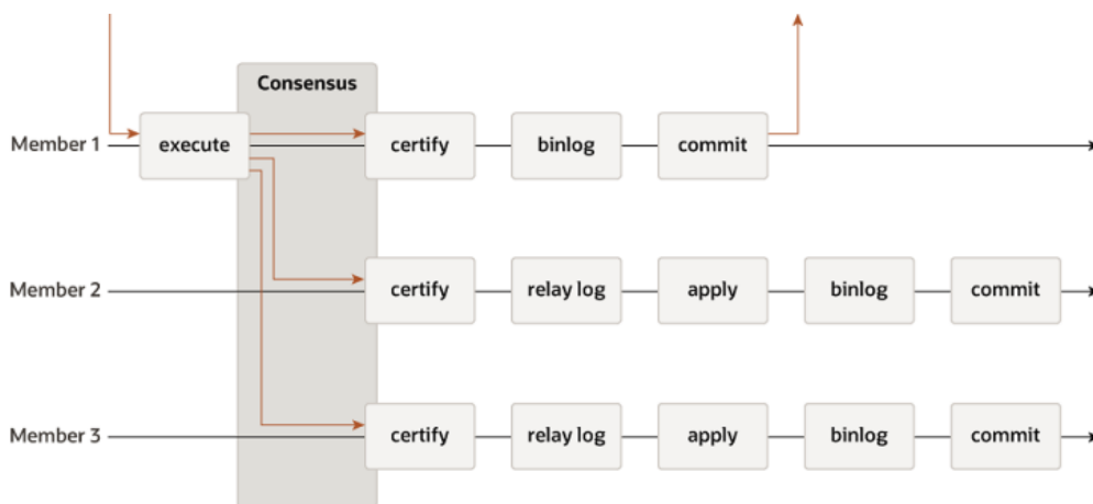
Tradicionalna *MySQL* replikacija pruža jednostavno repliciranje od izvora (*source*) do replika. Izvor je primarni, a replike (može ih biti i više) su sekundarne. Izvor izvršava transakciju, *commit*-uje je i kasnije šalje replikama (zbog toga je asinhrona) na ponovno izvršenje (*statement-based* replikacija) ili pak samo primena (*row-based* replikacija). Ovo je klasičan primer *shared-nothing* sistema jer svi serveri podrazumevano imaju sve podatke.

Takođe postoji i semisinhrona replikacija koja dodaje jedan sinhroni korak protokolu. Naime, *commit*-ovanje transakcije na primarnom serveru se vrši tek nakon pristizanja potvrde od replika o uspešnom primanju transakcije.



Slika 2.2. Asinhrona *MySQL* replikacija

odstupa od dogovorenog redosleda transakcija ukoliko se time ne narušava konzistentnost i validnost. Zapravo, grupna replikacija je *eventual consistency* sistem, što znači da se može desiti da čitanje sa neke replike ne vrati poslednju vrednost (zarad visoke dostupnosti), ali čim dođe do usporavanja ili zaustavljanja saobraćaja, svi članovi imaju identične podatke. Primera radi, može se desiti da se u *multi-primary mode*-u eksternalizuje lokalna transakcija odmah nakon sertifikacije iako postoji neka *remote* transakcija koja je u globalnom redu ispred lokalne. Naravno ovo je dozvoljeno nakon utvrđivanja procesa sertifikacije da nema konflikata. U *single-primary mode*-u veoma je mala šansa *commit*-ovanja lokalnih nekonfliktnih transakcija na primarnom serveru u redosledu koji se razlikuje od dogovorenog globalnog redosleda. Na replikama koji su *read-only* tipa, transakcije se uvek *commit*-uju prema globalnom redosledu.



Slika 2.4. *MySQL* grupna replikacija

2.1.2. Servisi grupne replikacije

Neki od servisa grupne replikacije su:

- članstvo grupe
- detekcija greške (*failure detection*)
- tolerancija greške (*fault-tolerance*)
- opažanje (*observability*).

Skup servera čini replikacionu grupu. Ima svoje ime i serveri mogu napuštati (namerno ili nenamerno) i pridruživati se grupi i to bilo kada, pri čemu se grupa prilagođava. Članovi grupe se pored dogovora o *commit*-ovima, moraju dogovoriti i o pogledu na samu grupu (klaster).

Servis detekcije greške je distribuirani servis koji identifikuje da određeni server ne komunicira sa ostatkom grupe i da je verovatno *offline*. Ukoliko se sumnja ispostavi tačnom,

vrši se izbacivanje člana iz grupe u cilju nastavka pravilnog rada klastera. U klasteru između svaka dva člana postoji *point-to-point* komunikacioni kanal koji leži na *TCP/IP socket*-ima (potpuno povezani graf). Jedan kanal se koristi za slanje, a drugi za primanje poruke. Ukoliko ne primi poruku od nekog člana klastera za x sekundi (po dokumentaciji 5s) sumnja se da je član otkazao i da mu je status *unreachable*. Ako sumnja traje duže od y sekundi (po dokumentaciji 10s), član koji sumnja propagira svoj stav i ostalim članovi. Treba doći do konsenzusa kako bi izbacili člana za kojeg se sumnja da je otkazao.

Servis tolerancije na greške pruža nastavak očekivanog rada klastera usled f grešaka. Naime, potrebna je većina aktivnih instanci radi postizanja kvoruma i donošenja odluke. Ukupan broj servera n može tolerisati f grešaka pri čemu je $n = 2*f + 1$. Primera radi, klaster od 3 servera (ili pak 4) može tolerisati najviše 1 grešku jer usled iste još uvek postoji većina ($\frac{2}{3}$, odnosno $\frac{3}{4}$ – preko 50%). Ukoliko dođe do većeg broja otkaza (od dozvoljenog), sistem se blokira.

Observability servis pruža uvid u dešavanje iza kulisa - shema performasi igra bitnu ulogu. S obzirom da je reč o distribuiranoj prirodi sistema (koja sama po sebi ima dosta problema) treba detaljno ispratiti stanje sistema nakon transakcije i sinhronizaciju ostalih instanci.

2.1.3. *Single-primary* i *multi-primary* režim

Grupna replikacija radi u jedan od sledeća dva režima:

- *single-primary*
- *multi-primary*.

Svi članovi grupe moraju raditi u istom režimu – ne može se desiti da jedni rade u jednom, a neki drugi u drugom. Sistemska promenljiva *group_replication_single_primary_mode* određuje režim rada i vrednost joj je ista kod svih članova grupe. Podrazumevana vrednost je *ON* i ne može se ručno menjati prilikom vršenja grupne replikacije.

U *single-primary* režimu samo na jednom serveru je dozvoljeno i čitanje i pisanje (*read-write* operacije), dok je na ostalim serverima moguće samo čitanje (*read-only*). Obično je primar server koji je prvi pristupio grupi. Serveri koji su kasnije dodati klasteru su *read-only* replike. U poređenju sa *multi-primary mode*-om, provera konzistentnosti nije toliko stroga jer grupna replikacija nameće upis samo na jednoj instanci. Ukoliko dođe do odsustva primara (iz bilo kojeg razloga), automatski se pokreće izbor novog primara. Ukoliko svi članovi grupe pokreću *MySQL* server verzije 8.0.13 ili veće, sam administrator sistema može imenovati novu primarnu instancu (*read-write* instancu). Može se desiti da novi primar sporije radi i da još uvek nije izvršio sve transakcije (operacije upisa) iz reda čekanja, te tako da pojedine operacije čitanja vrata neku zastarelu vrednost. Ovo se veoma retko dešava, pogotovo kada je *flow control* mehanizam grupne replikacije ispravno podešen. Ovaj

mehanizam ima ulogu u minimizaciji razlike između brzih i sporih članova grupe. Takođe u novijim verzijama *MySQL*-a (od verzije 8.0.14) može se podesti *BEFORE_ON_PRIMARY_FAILOVER* nivo konzistencije ili pak veći (sistemska promenljiva *group_replication_consistency*) koji zahteva primenu celokupnog *backlog*-a, pa tek nakon toga, moguće je izvršenje novih transakcija.

Automatski proces izbora novog primara uključuje gledanje novog pogleda od strane svakog člana grupe, kao i izbor najpogodnijeg člana. Svaki član donosi odluku lokalno na osnovu ugrađenog algoritma koji se može razlikovati među *MySQL* verzijama servera, tako da i to treba imati u vidu. S obzirom da svi članovi treba doneti istu odluku, algoritam koji se koristi mora odgovarati onom iz najstarije verzije *MySQL* servera u klasteru – bira se onaj sa najnižom verzijom. Ukoliko više članova ima istu najnižu verziju, težina člana igra sledeću bitnu ulogu. Određuje je sistemska promenljiva *group_replication_member_weight* i vrednost joj može biti od 0 do 100 (podrazumevano 50). Postavljanjem veće vrednosti, dajemo prednost instanci prilikom prvog narednog izbora (primera radi imamo bolji hardver na toj instanci). Ako *MySQL* verzija nema ovu promenljivu, ovaj faktor se ignoriše (na primer verzija 5.7). Treći i ujedno poslednji faktor koji se gleda (ukoliko se ne može odlučiti na osnovu prethodna 2) jeste leksikografski redosled jedinstvenog identifikatora servera – član sa najnižom vrednošću biće izabran. To je ujedno i garancija da će svi članovi grupe doći do iste odluke.

U *multi-primary* režimu nijedan član nema specijalnu ulogu – svi su *read-write* replike (svi su jednaki). Grupna replikacija se zasniva na optimističnoj paradigmi – naredbe se izvršavaju optimistično, a kasnije ukoliko je neophodno poništavaju (*roll back*-uju). Cilj je nemati nekonzistentno stanje, odnosno svaki novonastali konflikt se mora rešiti. Problem koji ovde postoji jeste konkurentno izvršavanje transakcija koje se odnose na iste redove jedne tabele, a rešenje ovog problema je opisano u okviru istog poglavlja u delu o grupnoj replikaciji.

3. ZAHTEVI *InnoDB* KLASTERA

Postoje određeni zahtevi koji moraju biti ispunjeni radi dobrog funkcionisanja samog klastera, a neki od njih su:

- Kako *InnoDB* koristi grupnu replikaciju, instance servera moraju poštovati zahteve koji tamo postoje (primera radi korišćenje *InnoDB storage engine*-a za skladištenje podataka, definisan primarni ključ, jedinstveni identifikator servera,...). *AdminAPI* pruža *dba.checkInstanceConfiguration()* metodu radi provere ispunjenja zahteva grupne replikacije od strane instance (ukoliko je sve u redu, vraća *status: ok*), kao i metodu *dba.configureInstance()* u cilju konfigurisanja instance tako da ispunjava sve zahteve. Korišćenjem *sandbox deployment*-a (izolovano okruženje) instance su automatski pravilno konfigurisane.
- Podaci koji se koriste u *InnoDB* klasteru i samim tim u grupnoj replikaciji moraju biti skladišteni u *InnoDB storage engine*-u. Za razliku od *MyISAM storage engine*-a koji ne podržava transakcije, *InnoDB*, koji je podrazumevani *storage engine* u trenutnoj verziji, podržava iste i generalno se preporučuje za baze podataka koje stalno vrše neku obradu (*busy databases*). Neophodna je transformacija tabele tako da za skladištenje podataka koristi *InnoDB storage engine* (ukoliko ga već ne koristi) radi izbegavanja grešaka prilikom grupne replikacije.
- *InnoDB* klaster ne podržava ručno konfigurisanje asinhronih kanala replikacije, jedino su dozvoljeni automatski kreirani kanali korišćenjem grupne replikacije – *group_replication_applier* i *group_replication_recovery*.
- *group_replication_tls_source* ne sme biti postavljen na *mysql_admin*.
- Shema performasi (*feature* za nadgledanje delanja *MySQL* servera na niskom nivou) mora biti uključena na svakoj instanci u klasteru.
- Od verzije 8.0.23. instance treba konfigurisati tako da koriste paralelnu replikaciju (upotreba više niti prilikom replikacije).
- Skripta koja *MySQL Shell* koristi u cilju konfiguracije servera u klasteru zahtevaju pristup *python*-u – sistem mora imati konfigurisan *python*.
- Od verzije 8.0.17, instance moraju imati jedinstveni identifikator u klasteru.
- Relevantne opcije konfiguracije instanci, a pogotovo opcije konfiguracije grupne replikacije moraju biti u 1 opcionoj datoteci, koja treba biti specificirana radi korišćenja od strane *AdminAPI*-a.

4. OGRANIČENJA *InnoDB* KLASTERA

Iako nudi veliki broj pogodnosti, *MySQL InnoDB* klaster poseduje određena ograničenja kojih *developer* treba biti svestan. Neka od poznatih ograničenja *InnoDB* klastera su sledeća:

- Kompleksno podešavanje i konfiguracija – zahteva pažljivo planiranje i dobro poznavanje tehnologija na kojima „leži“ *MySQL InnoDB* klaster;
- *InnoDB* klaster ne upravlja ručno konfigurisanim asinhronim kanalima replikacije – može se desiti da zbog asinhronne replikacije čvorovi nemaju iste podatke i da samim tim postoji nekonzistentnost u klasteru;
- S obzirom da se „oslanja“ na *MySQL* grupnu replikaciju (tehnologija odgovorna za replikaciju podataka na više čvorova), ograničenja koja važe za grupnu replikaciju, odnose se i na ograničenja samog klastera (na primer minimalni broj servera u replikacionoj grupi je 3, a maksimalni 9, ili pak zahtev da čvorovi budu u okviru lokalne mreže zbog niske latencije);
- *InnoDB* klaster radi samo sa *MySQL* bazom podataka – ne može se koristiti za druge baze podataka;
- Upis u *single-primary mode*-u se vrši samo na jednom čvoru radi postizanja konzistencije, ali se time ograničava skalabilnost upisa u klasteru;
- Preporuka je da se *deployment InnoDB* klastera vrši u lokalnoj mreži jer ukoliko se vrši na internetu, postojaće primetan uticaj na performanse upisa – stabilna mreža sa niskom latencijom je veoma važna za komunikaciju servera unutar klastera radi ostvarivanja grupne replikacije;
- Povezivanje *AdminAPI*-a sa instancama servera u klasteru se vrši samo *TCP/IP* konekcijom ili pak standardnim *MySQL* protokolom. Ne mogu se koristiti recimo *pipe*-ovi ili *Unix socket*-i. Ista ograničenja važe i za povezivanje između samih instanci servera. Za klijentske aplikacije ovo ograničenje ne važi;
- *AdminAPI* i *InnoDB* klaster mogu podržavati instance *MySQL* servera verzije 5.7, ali u tom slučaju postoje dodatna ograničenja i pojedini *feature*-i se ne mogu koristiti;
- Postoji zavisnost od same mreže – bilo koji problem ili pad iste rezultira upitnom dostupnošću i performansama samog klastera.

5. RAD SA *InnoDB* KLASTEROM UZ TEORIJSKA OBJAŠNJENJA

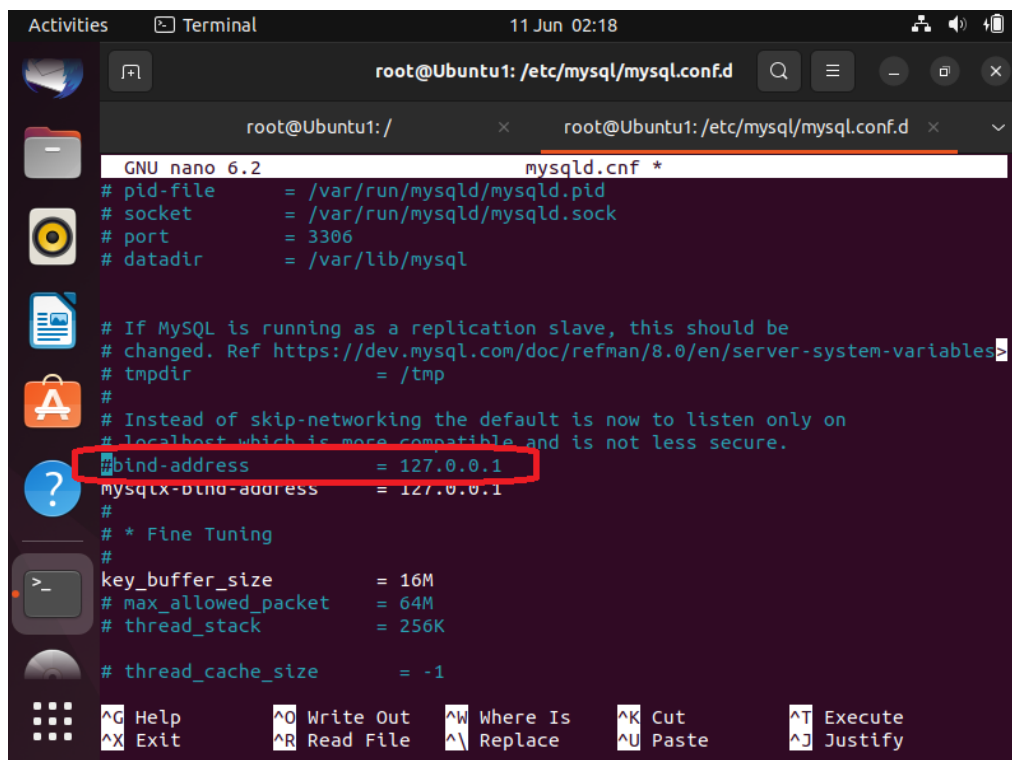
5.1. Priprema okruženja

Najpre smo instalirali *Oracle VM VirtualBox* radi omogućavanja virtuelizacije – proširivanje postojećeg računara pokretanjem više instanci istovremeno (u našem slučaju 3) sa različitim operativnim sistemom. Na svim *host*-ovima smo instalirali *Linux* operativni sistem distribucije *Ubuntu 22.04*. Korišćenjem *apt*-a (interaktivni *command-line* alat za upravljanje *deb* paketima) skidamo i instaliramo *MySQL* server na svakom *host*-u sledećom komandom:

`sudo apt install mysql-server` (nakon komandi za ažuriranje i nadogradnju *apt* paket repozitorijuma).

Isprva se nismo mogli povezati na *MySQL* server (*root@Ubuntu1*) dobijajući grešku 111 – *Can't connect to MySQL server*. Podrazumevano *MySQL* server „sluša“ samo na *localhost* interface-u, te odmah na početku moramo menjati konfiguracionu datoteku */etc/mysql/mysql.conf.d/mysqld.conf*. Inače, važno je napomenuti da su sve promene vršene od strane privilegovanog korisnika - administratora. Uklanjam sledeću liniju:

`bind-address = 127.0.0.1` (zakomentarišući je, što je prikazano na slici ispod).



Slika 5.1. Uklanjanje sporne linije u konfiguracionoj datoteci

Takođe, važno je naglasiti da nakon svake izmene konfiguracione datoteke treba *restart*-ovati server radi ponovog učitavanja i primene iste. U te svrhe je korišćena sledeća naredba:

`systemctl restart mysql.service.`

Na svim instancama instaliramo *MySQL Shell* radi potencijalne administracije (ne dolazi zajedno sa instalacijom *MySQL* servera). Na slici ispod je prikazan postupak.

```
vboxuser@Ubuntu2:~$ snap install mysql-shell
mysql-shell 8.0.23 from Canonical✓ installed
vboxuser@Ubuntu2:~$ mysqlsh
Cannot set LC_ALL to locale en_US.UTF-8: No such file or directory
MySQL Shell 8.0.23

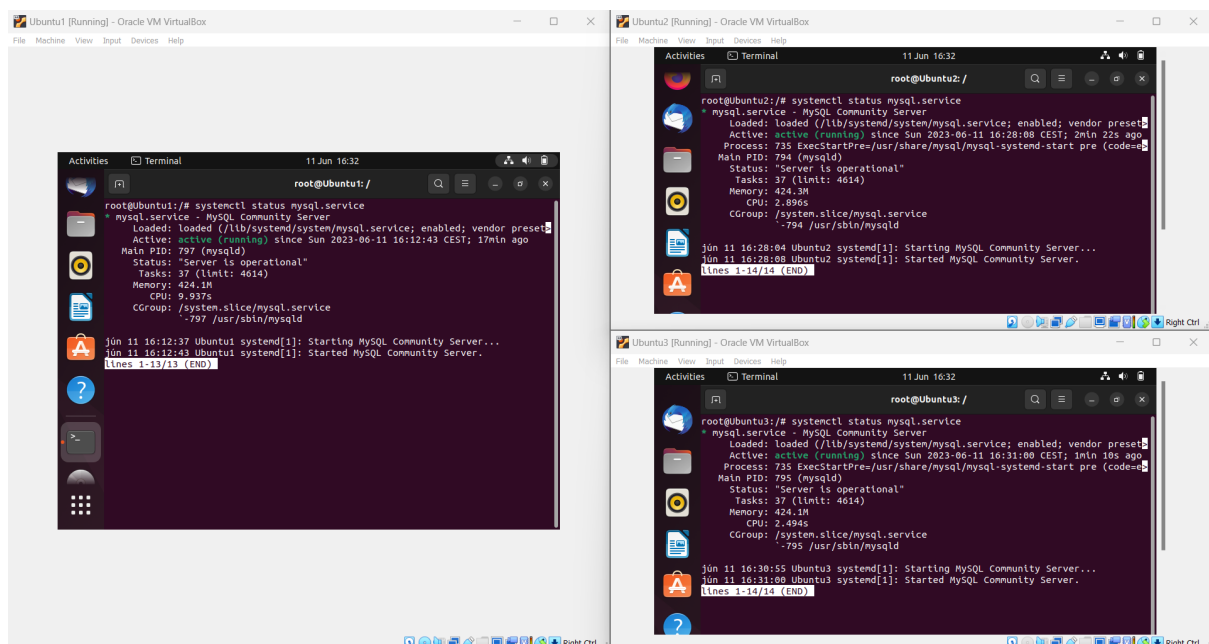
Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
mysql-py> |
```

Slika 5.2. Instalacija *MySQL Shell*-a

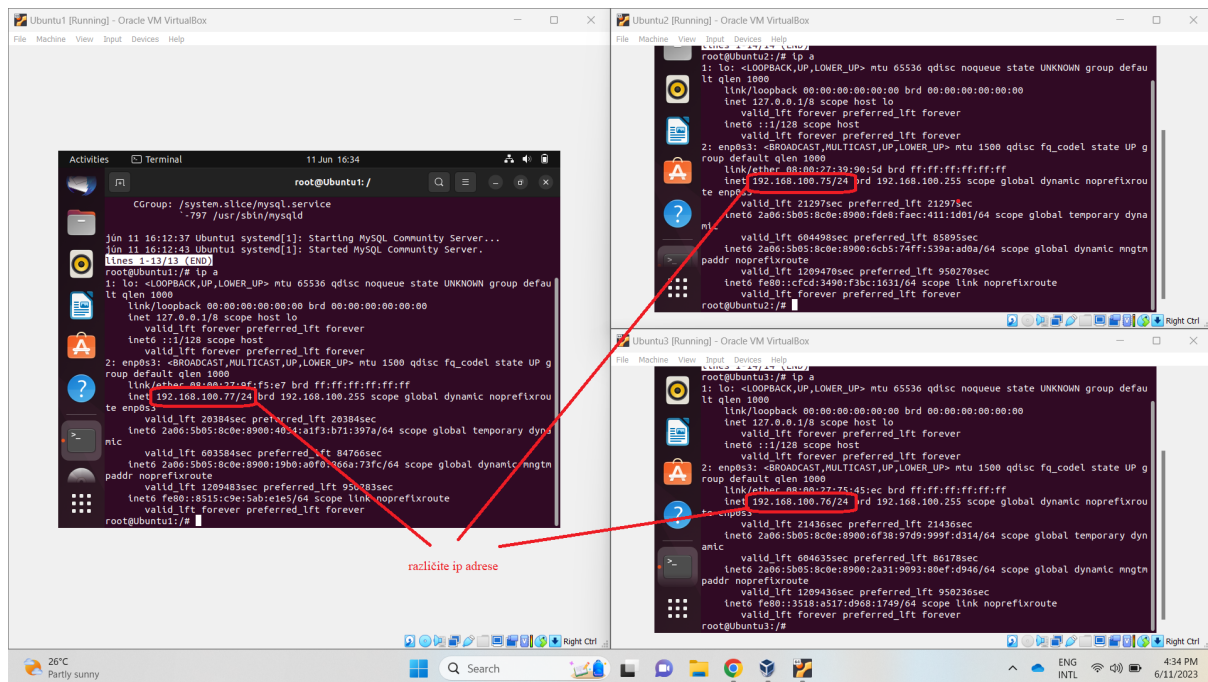
Nakon pokretanja mašina, podrazumevano *MySQL* serveri su pokrenuti i to je prikazano na slici ispod. Rade nezavisno jedan od drugog, još uvek nema klastera. Provera je izvršena sledećom komandom:

`systemctl status mysql.service.`



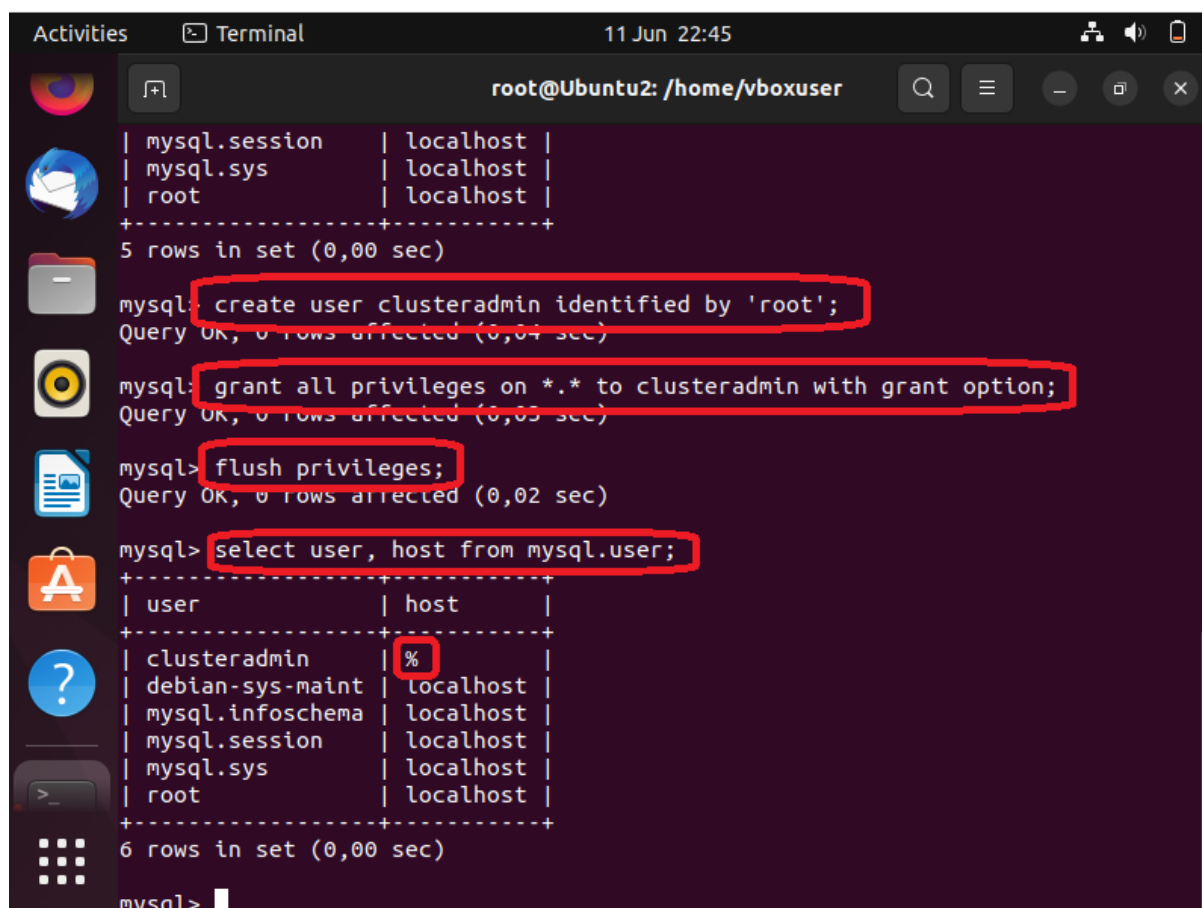
Slika 5.3. Svi *MySQL* serveri su aktivni (*running*)

Proverom *ip* adresa (komandom: *ip a*) ustanovili smo da su sve instance inicijalno imale istu lokalnu *ip* adresu. Ova činjenica predstavlja problem jer je cilj ostvariti komunikaciju između *host*-ova u lokalnoj mreži, te je neophodno imati različite adrese. Ovaj problem rešavamo postavljanjem *Network Adapter*-a na *Bridged Adapter*-a u kartici *Network* u okviru podešavanja svakog *host*-a. Konačno su dobijene različite *ip* adrese i to se da videti na slici ispod.



Slika 5.4. Svaki *host* ima različitu lokalnu *ip* adresu

Poželjno je kreirati novog korisnika za administraciju kako pojedinačnog servera, tako i klastera u celini. Na svakoj mašini pokrećemo *MySQL Shell* i kreiramo novog *clusteradmin* korisnika. Dodeljujemo mu sve privilegije sa mogućnošću dodele privilegija ostalim korisnicima na specifičnom nivou koje sam *clusteradmin* poseduje (klauzula *with grant option*). Nakon toga, za svaki slučaj, izvršavamo *flush privileges* komandu radi eventualnog čišćenja keširane memorije servera. Po dokumentaciji ova komanda nije neophodna jer smo dodelu privilegija vršili pomoću *grant* naredbe - *MySQL* odmah opaža promene i automatski ponovo učitava *grant* tabele. Ukoliko bi se pak modifikacija *grant* tabela vršila direktno korišćenjem *INSERT*, *UPDATE* ili *DELETE* naredbi, onda bi *flush privileges* naredba bila neophodna. Konačno selektujemo kolone *user* i *host* iz *mysql.user* tabele radi uveravanja uspešnog kreiranja novog korisnika kome se može pristupiti sa svih lokacija. Vrednost *host* kolone za novokreiranog korisnika (%) označava mogućnost povezivanja *TCP/IP* konekcijom sa bilo kojeg *host*-a, ali ne uključuje *localhost* kod koga se povezivanje ostvaruje korišćenjem *UNIX socket*-a.



The screenshot shows a terminal window titled 'Terminal' with the prompt 'root@Ubuntu2: /home/vboxuser'. The terminal displays the following MySQL commands and their outputs:

```
mysql> select user, host from mysql.user;
+-----+-----+
| user          | host          |
+-----+-----+
| clusteradmin  | %             |
| debian-sys-maint | localhost    |
| mysql.infoschema | localhost    |
| mysql.session | localhost    |
| mysql.sys     | localhost    |
| root          | localhost    |
+-----+-----+
6 rows in set (0,00 sec)

mysql> create user clusteradmin identified by 'root';
Query OK, 0 rows affected (0,04 sec)

mysql> grant all privileges on *.* to clusteradmin with grant option;
Query OK, 0 rows affected (0,03 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0,02 sec)

mysql> select user, host from mysql.user;
+-----+-----+
| user          | host          |
+-----+-----+
| clusteradmin  | %             |
| debian-sys-maint | localhost    |
| mysql.infoschema | localhost    |
| mysql.session | localhost    |
| mysql.sys     | localhost    |
| root          | localhost    |
+-----+-----+
6 rows in set (0,00 sec)

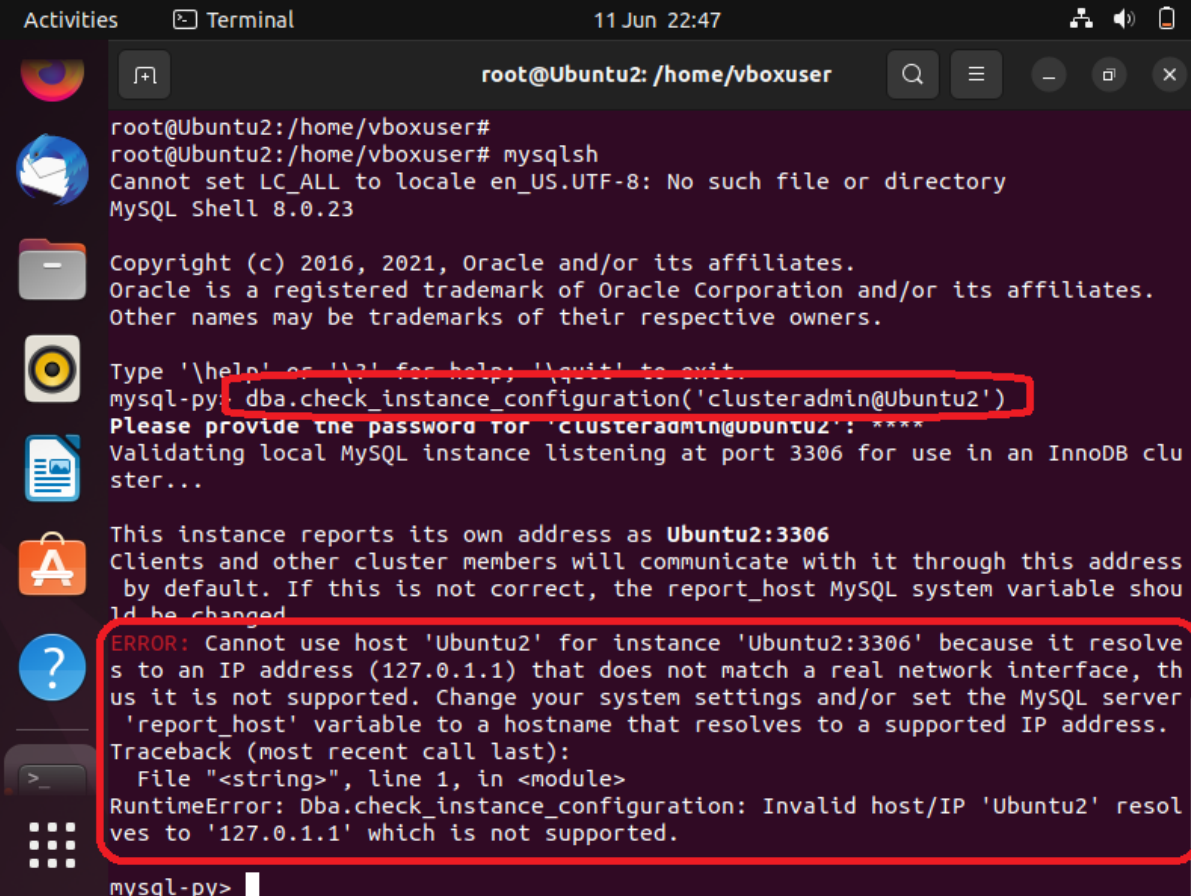
mysql>
```

Slika 5.5. Kreiranje novog korisnika sa svim privilegijama

5.2. Konfiguracija, administracija i *deployment* InnoDB klastera

Korišćenjem *AdminAPI*-a koji je sastavni deo *MySQL Shell*-a i u okviru istog globalne promenljive *dba* i njenih metoda postiže se konfiguracija, administracija i *deployment* InnoDB klastera. Pre kreiranja produkcionog *deployment*-a, neophodno je proveriti da li su sve *MySQL* instance (instance koje će činiti klaster) pravilno konfigurisane. Verifikaciju ispunjavanja liste zahteva InnoDB klastera opisanu u trećem poglavlju vršimo funkcijom *dba* promenljive *check_instance_configuration* i kao parametar eventualno prosleđujemo željenu instancu. Ukoliko nemamo parametar, podrazumeva se provera konfiguracije instance sa kojom smo se prethodno povezali.

Na slici ispod je prikazan pokušaj provere instance *clusteradmin@Ubuntu2*. Javila se greška jer prethodno nismo ispravno definisali *ip* adresu mašine, već u */etc/hosts* datoteci podrazumevano stoji 127.0.1.1 adresa za *Ubuntu2* server. Neophodno je izmeniti pomenutu datoteku postavljanjem realne lokalne *ip* adrese *host*-a (u našem slučaju 192.168.100.75). Kao što smo već naveli, *ip* adresa se može videti izvršenjem *ip a* naredbe. Na slici 5.7. vidimo postavljanje ispravne *ip* adrese. Potrebno je ovaj postupak ponoviti i na preostale dve mašine.

A terminal window titled 'Terminal' with the date '11 Jun 22:47' and the user 'root@Ubuntu2: /home/vboxuser'. The terminal shows the execution of 'mysqlsh' which prompts for a password. The user enters 'clusteradmin@Ubuntu2'. The terminal then displays an error message: 'ERROR: Cannot use host 'Ubuntu2' for instance 'Ubuntu2:3306' because it resolves to an IP address (127.0.1.1) that does not match a real network interface, thus it is not supported. Change your system settings and/or set the MySQL server 'report_host' variable to a hostname that resolves to a supported IP address.' The error message is highlighted with a red box. The terminal also shows the MySQL version '8.0.23' and the copyright notice for Oracle.

```
root@Ubuntu2: /home/vboxuser# mysqlsh
Cannot set LC_ALL to locale en_US.UTF-8: No such file or directory
MySQL Shell 8.0.23

Copyright (c) 2016, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

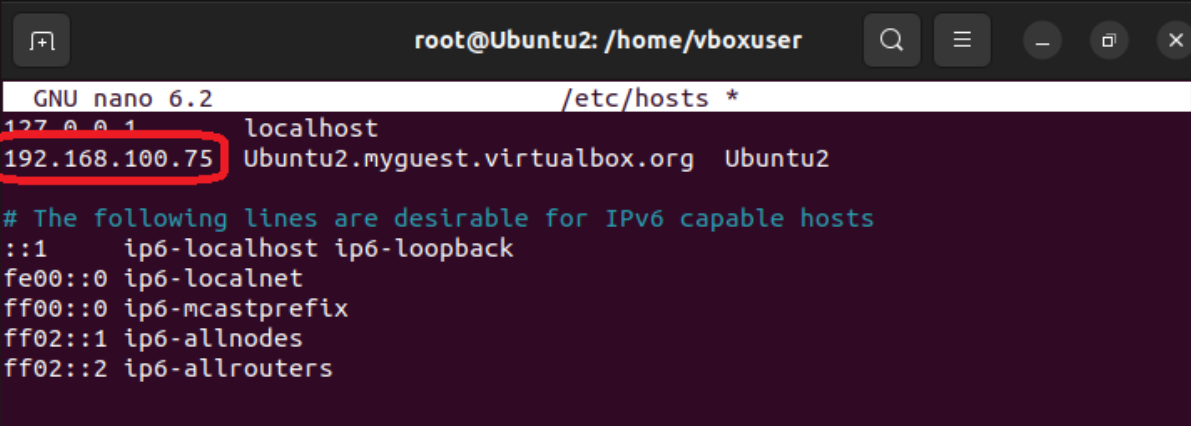
Type '\help' or '\?' for help; '\quit' to exit.
mysql-py> dba.check_instance_configuration('clusteradmin@Ubuntu2')
Please provide the password for 'clusteradmin@Ubuntu2': ****
Validating local MySQL instance listening at port 3306 for use in an InnoDB cluster...

This instance reports its own address as Ubuntu2:3306
Clients and other cluster members will communicate with it through this address
by default. If this is not correct, the report_host MySQL system variable should
be changed.

ERROR: Cannot use host 'Ubuntu2' for instance 'Ubuntu2:3306' because it resolves
to an IP address (127.0.1.1) that does not match a real network interface, thus
it is not supported. Change your system settings and/or set the MySQL server
'report_host' variable to a hostname that resolves to a supported IP address.
Traceback (most recent call last):
  File "<string>", line 1, in <module>
RuntimeError: Dbapi.check_instance_configuration: Invalid host/IP 'Ubuntu2' resolves
to '127.0.1.1' which is not supported.

mysql-py>
```

Slika 5.6. Greška prilikom pokušaja provere konfiguracije instance

A terminal window showing the 'nano' text editor editing the '/etc/hosts' file. The file content is as follows:

```
127.0.0.1 localhost
192.168.100.75 Ubuntu2.myguest.virtualbox.org Ubuntu2

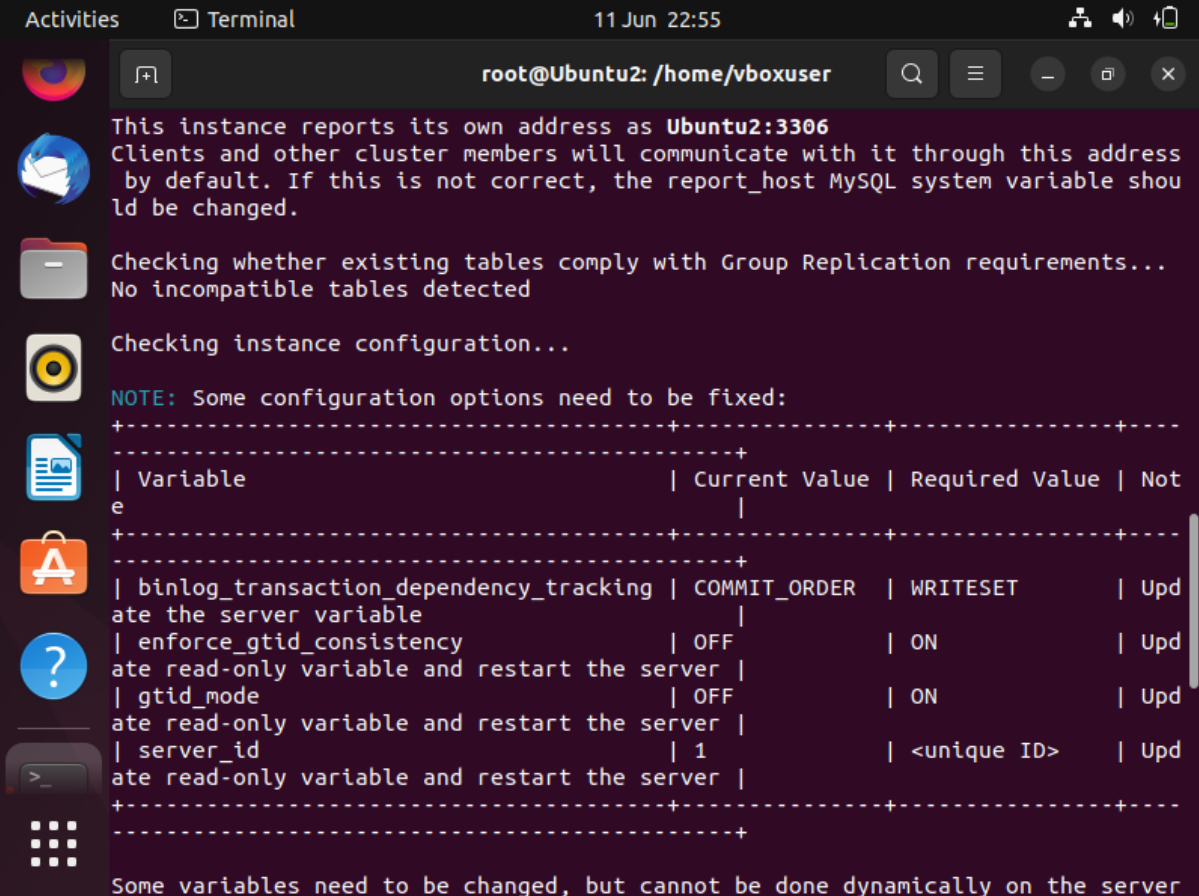
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

The IP address '192.168.100.75' is highlighted with a red box.

Slika 5.7. Postavljanje ispravne ip adrese servera

Sledeći poziv `check_instance_configuration` je uspešan i rezultat istog je prikazan na slikama 5.8. i 5.9. Najpre se vrši provera ispunjavanja zahteva grupne replikacije od strane postojećih tabela, a nakon toga sama provera konfiguracije instance. Prilikom našeg izvršenja, nema nijedne sporne tabele, ali pojedine konfiguracione opcije zahtevaju „popravku“, odnosno promenu vrednosti:

- *binlog_transaction_dependency_tracking* – umesto *COMMIT_ORDER*, vrednost treba biti *WRITESET* (ne zahteva restart-ovanje servera);
- *enforce_gtid_consistency* – umesto *OFF*, vrednost treba biti *ON* (zahteva restart-ovanje servera);
- *gtid_mode* – umesto *OFF*, vrednost treba biti *ON* (zahteva restart-ovanje servera);
- *server_id* – vrednost treba biti jedinstveni *id* instance (zahteva restart-ovanje servera).



```

root@Ubuntu2: /home/vboxuser
This instance reports its own address as Ubuntu2:3306
Clients and other cluster members will communicate with it through this address
by default. If this is not correct, the report_host MySQL system variable shou
ld be changed.

Checking whether existing tables comply with Group Replication requirements...
No incompatible tables detected

Checking instance configuration...

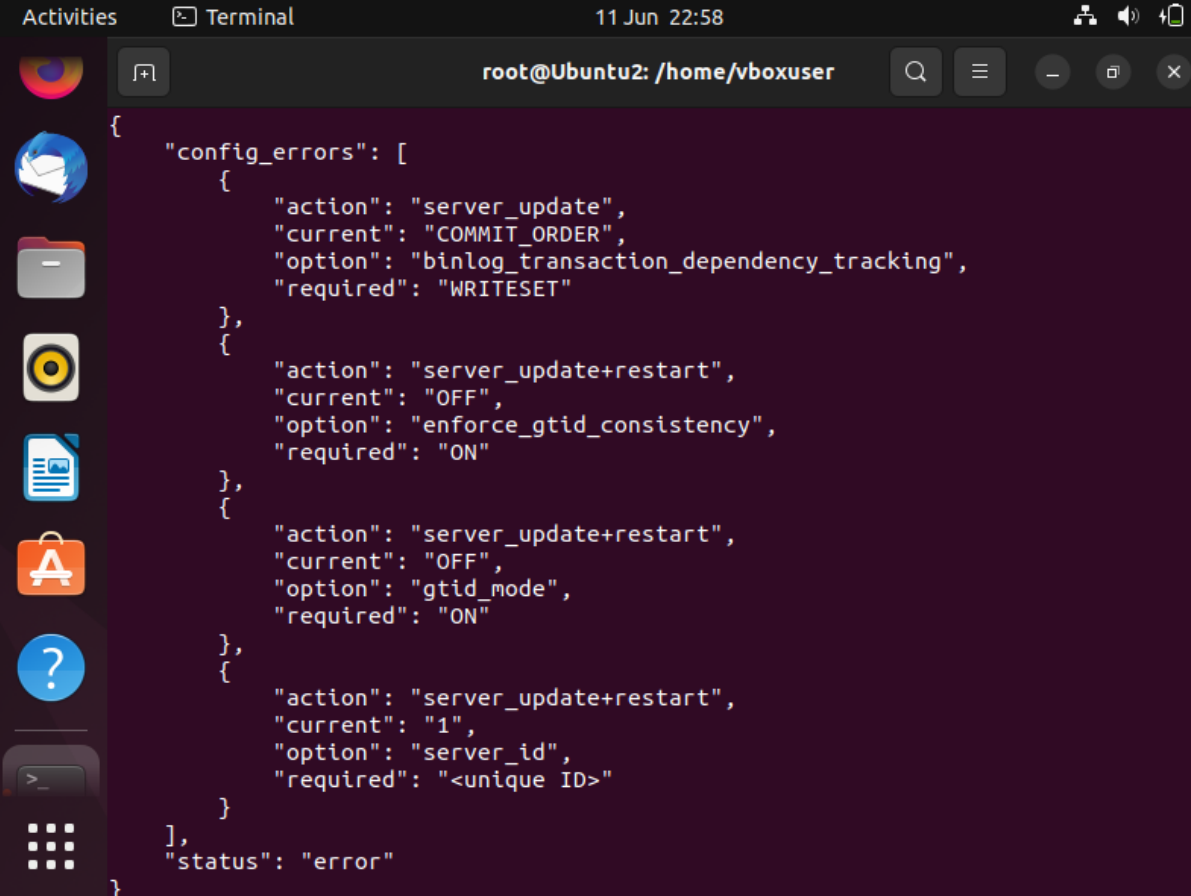
NOTE: Some configuration options need to be fixed:
+-----+-----+-----+-----+
| Variable | Current Value | Required Value | Note |
+-----+-----+-----+-----+
| binlog_transaction_dependency_tracking | COMMIT_ORDER | WRITESET | Update the server variable |
| enforce_gtid_consistency | OFF | ON | Update read-only variable and restart the server |
| gtid_mode | OFF | ON | Update read-only variable and restart the server |
| server_id | 1 | <unique ID> | Update read-only variable and restart the server |
+-----+-----+-----+-----+

Some variables need to be changed, but cannot be done dynamically on the server

```

Slika 5.8. Provera konfiguracije instance – vrednosti pojedinih opcija moraju biti promenjene

Na slici ispod je tabela sa slike iznad prikazana u čitljivijem, *json* formatu.



```
{
  "config_errors": [
    {
      "action": "server_update",
      "current": "COMMIT_ORDER",
      "option": "binlog_transaction_dependency_tracking",
      "required": "WRITESSET"
    },
    {
      "action": "server_update+restart",
      "current": "OFF",
      "option": "enforce_gtid_consistency",
      "required": "ON"
    },
    {
      "action": "server_update+restart",
      "current": "OFF",
      "option": "gtid_mode",
      "required": "ON"
    },
    {
      "action": "server_update+restart",
      "current": "1",
      "option": "server_id",
      "required": "<unique ID>"
    }
  ],
  "status": "error"
}
```

Slika 5.9. Opcije sa neispravnim (*current*) i zahtevanim (*required*) vrednostima

Neophodno je pre dodavanja instance u *InnoDB* klaster, pozvati metodu *dba* globalne promenljive *configure_instance* i kao parametar poslati željenu instancu (za početak *'clusteradmin@Ubuntu1:3306'*). Funkcija vrši proveru prikladnosti konfiguracije radi upotrebe u okviru *InnoDB* klastera i konfiguriše instancu ukoliko pronade podešavanja koja nisu kompatibilna sa zahtevima *InnoDB* klastera. Na slici ispod je prikazano izvršenje *configure_instance* funkcije nad *Ubuntu1* instancom. Kao i kod funkcije *check_instance_configuration* imamo štampanje svih konfiguracionih opcija koje zahtevaju promenu vrednosti, ali ovoga puta nas funkcija pita da li želimo da izvršimo neophodne promene konfiguracije – naš odgovor je *y* (da). S obzirom da promena određenih opcija zahteva *restart*-ovanje servera, na sledeće pitanje o istom odgovaramo takođe sa *y* (da). Na samom kraju dobijamo obaveštenje da je *Ubuntu1* instanca konfigurisana tako da se može koristiti u okviru *InnoDB* klastera, kao i da je *MySQL* server *restart*-ovan. Treba pozvati ovu funkciju i za preostale 2 instance (*dba.configure_instance('clusteradmin@192.168.100.75')* i *dba.configure_instance('clusteradmin@192.168.100.81')* - gde su 192.168.100.x ip adrese servera). Umesto ip adresa može se napisati i *Ubuntu2:3306* i *Ubuntu3:3306* respektivno (*resolver* mora raditi ispravno u cilju dobijanja željenih rezultata).

```

Activities  Terminal  11 Jun 23:09  vboxuser@Ubuntu1: ~
+-----+-----+-----+-----+
| Variable | Current Value | Required Value | Note |
+-----+-----+-----+-----+
| binlog_transaction_dependency_tracking | COMMIT_ORDER | WRITESET | Update the server variable |
| enforce_gtid_consistency | OFF | ON | Update read-only variable and restart the server |
| gtid_mode | OFF | ON | Update read-only variable and restart the server |
| server_id | 1 | <unique ID> | Update read-only variable and restart the server |
+-----+-----+-----+-----+

Some variables need to be changed, but cannot be done dynamically on the server
.
Do you want to perform the required configuration changes? [y/n]: y
Do you want to restart the instance after configuring it? [y/n]: y
Configuring instance...

WARNING: '@@slave_parallel_workers' is deprecated and will be removed in a future release. Please use replica_parallel_workers instead. (Code 1287).
The instance 'Ubuntu1:3306' was configured to be used in an InnoDB cluster.
Restarting MySQL...
NOTE: MySQL server at Ubuntu1:3306 was restarted.
mysql-py>

```

Slika 5.10. Uspješno konfiguiranje instance za *InnoDB* klaster

Dobro je još jednom proveriti validnost konfiguracija svih instanci za rad u *InnoDB* klasteru. Važno je dobiti informaciju da je *status ok*. Provera instance *Ubuntu2*, koja se vrši sa *host-a Ubuntu1*, odnosno *remote*, prikazana je na slici ispod.

```

mysql-py> dba.check_instance_configuration('clusteradmin@192.168.100.75')
Please provide the password for 'clusteradmin@192.168.100.75': ****
Validating MySQL instance at Ubuntu2:3306 for use in an InnoDB cluster...

This instance reports its own address as Ubuntu2:3306
Clients and other cluster members will communicate with it through this address
by default. If this is not correct, the report_host MySQL system variable should
be changed.

Checking whether existing tables comply with Group Replication requirements...
No incompatible tables detected

Checking instance configuration...
Instance configuration is compatible with InnoDB cluster

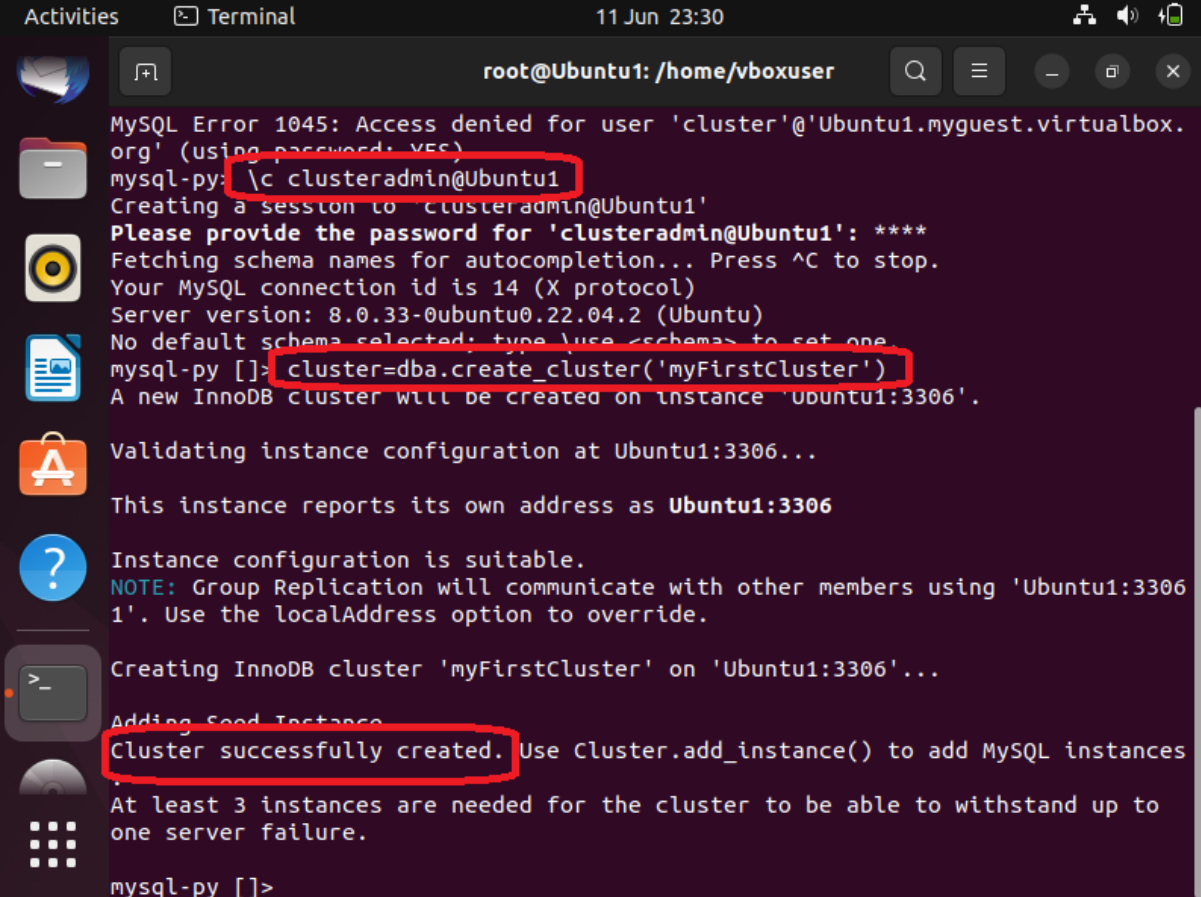
The instance 'Ubuntu2:3306' is valid to be used in an InnoDB cluster.

{
  "status": "ok"
}
mysql-py>

```

Slika 5.11. Provera konfiguracije instance *Ubuntu2*

S obzirom na činjenicu da su sve instance pripremljene za rad u okviru *InnoDB* klastera, možemo kreirati sam klaster. Koristeći *MySQL Shell* na *host-u Ubuntu1*, kreiramo konekciju ka *clusteradmin@Ubuntu1* naredbom `\c clusteradmin@Ubuntu1`. Koristi se podrazumevani *X* protokol koji nudi najbolju integraciju sa *MySQL* serverom (nismo eksplicitno zadali konekcijske parametre vezane za protokol). U cilju omogućavanja istog, *X plugin* treba biti instaliran i omogućen na samoj instanci. Za razliku od prethodne verzije *MySQL* 5.7. gde se morao ručno instalirati, od verzije *MySQL* 8.0. *default*-no dolazi sa datom instalacijom, tako da nema potrebe za dodatnim koracima. Nakon kreiranja sesije, metodom *create_cluster dba* globalne promenljive kreiramo klaster čije ime prosledujemo kao parametar. Referencu novokreiranog klastera smeštamo u lokalnu promenljivu koju smo imenovali *cluster*. Ukoliko istu izgubimo, moguće je ponovo je dobiti korišćenjem metode *get_cluster()* globalne *dba* promenljive. Dobijamo poruku da je novi *InnoDB* klaster uspešno kreiran na *host-u Ubuntu1* i da je sama instanca *Ubuntu1* dodata nakon provere u klaster. Takođe, dobijamo poruku da je potrebno imati najmanje 3 instance radi otpornosti našeg klastera na otkaz nekog od servera. Opisani proces je prikazan na slici ispod.



```
Activities Terminal 11 Jun 23:30
root@Ubuntu1: /home/vboxuser

MySQL Error 1045: Access denied for user 'cluster'@'Ubuntu1.myguest.virtualbox.
org' (using password: YES)
mysql-py: \c clusteradmin@Ubuntu1
Creating a session to 'clusteradmin@Ubuntu1'
Please provide the password for 'clusteradmin@Ubuntu1': ****
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 14 (X protocol)
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)
No default schema selected; type \use <schema> to set one
mysql-py []: cluster=dba.create_cluster('myFirstCluster')
A new InnoDB cluster will be created on instance 'Ubuntu1:3306'.

Validating instance configuration at Ubuntu1:3306...
This instance reports its own address as Ubuntu1:3306

Instance configuration is suitable.
NOTE: Group Replication will communicate with other members using 'Ubuntu1:3306
1'. Use the localAddress option to override.

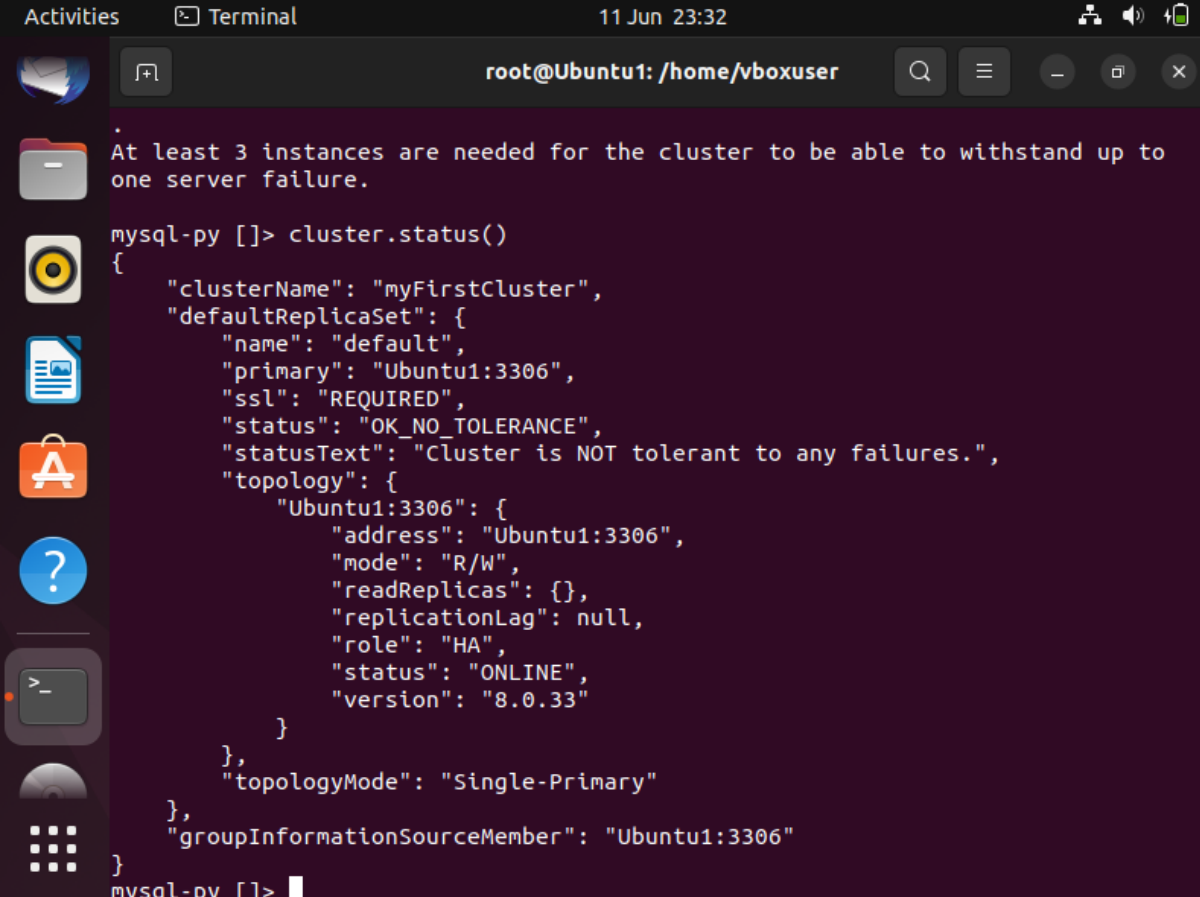
Creating InnoDB cluster 'myFirstCluster' on 'Ubuntu1:3306'...
Adding Seed Instance
Cluster successfully created. Use Cluster.add_instance() to add MySQL instances
At least 3 instances are needed for the cluster to be able to withstand up to
one server failure.

mysql-py []>
```

Slika 5.12. Kreiranje klastera

Na slici ispod je prikazano nadgledanje novokreiranog klastera metodom *klastera status()*. Dobijeni odgovor je *json* formata. Vidimo ime klastera, naziv primarne instance (*Ubuntu1:3306*) na kojoj se vrši kako čitanje, tako i upis, samu topologiju – informacije o

samoj instanci. Radi se o *single-primary mode*-u, odnosno upis se može vršiti na samo jednoj instanci. Takođe, ova funkcija nas obaveštava da klaster nije tolerantan ni na jednu grešku s obzirom da se sastoji od samo jedne instance. Zapravo, mi ovde ne dobijamo nikakve prednosti klasterizacije, zato hajdemo dodati još instanci našem klasteru.



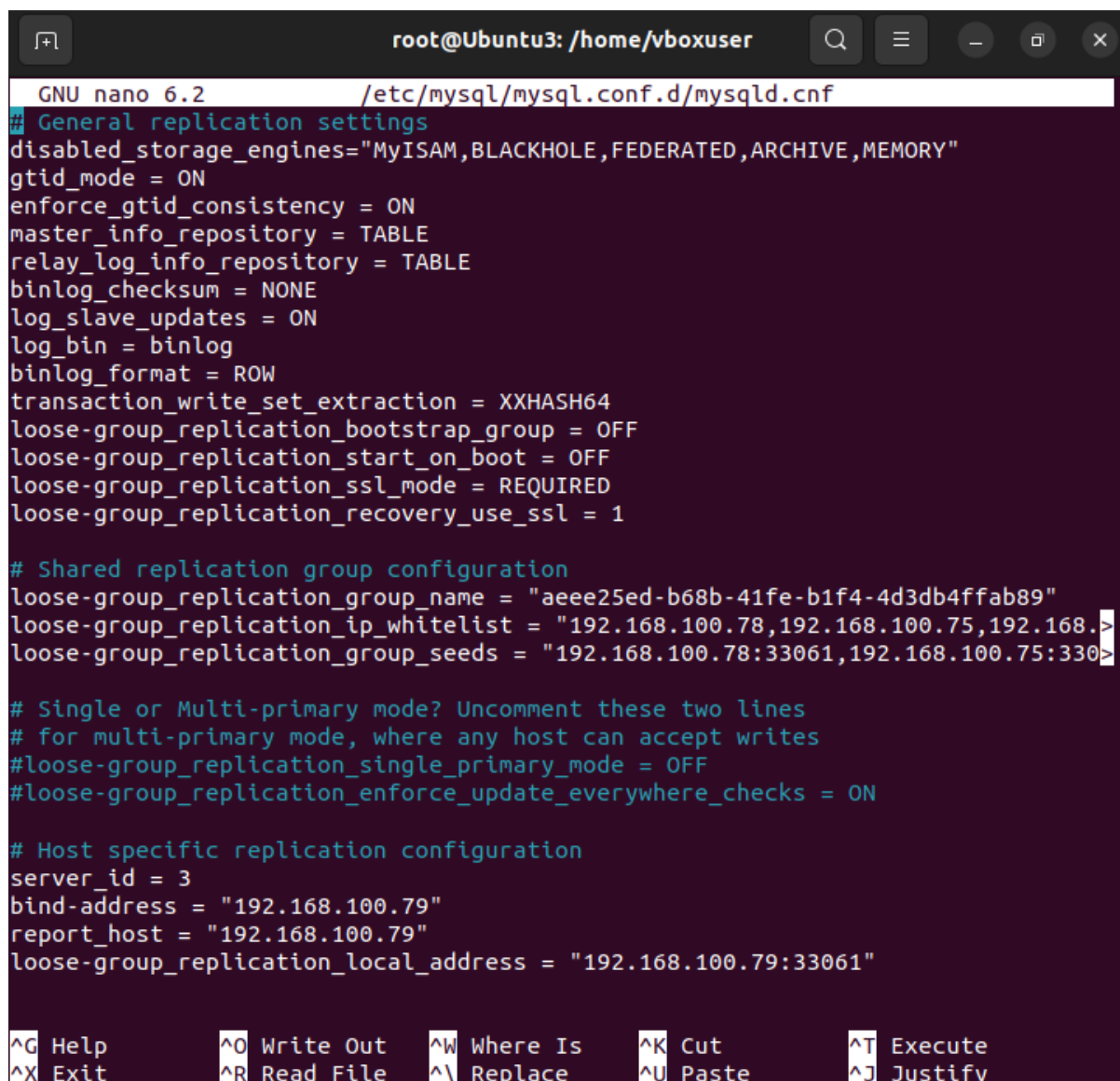
```
At least 3 instances are needed for the cluster to be able to withstand up to
one server failure.

mysql-py []> cluster.status()
{
  "clusterName": "myFirstCluster",
  "defaultReplicaSet": {
    "name": "default",
    "primary": "Ubuntu1:3306",
    "ssl": "REQUIRED",
    "status": "OK_NO_TOLERANCE",
    "statusText": "Cluster is NOT tolerant to any failures.",
    "topology": {
      "Ubuntu1:3306": {
        "address": "Ubuntu1:3306",
        "mode": "R/W",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      }
    }
  },
  "topologyMode": "Single-Primary"
},
"groupInformationSourceMember": "Ubuntu1:3306"
}
mysql-py []>
```

Slika 5.13. Status klastera sa jednom instancom

Kako bismo klasteru dodali još 2 instance (*Ubuntu2* i *Ubuntu3*) neophodno je izmeniti konfiguracione datoteke sve 3 instance tako da omogućavaju grupnu replikaciju. Ukoliko bismo pokušali sa dodavanjem instanci bez promena istih, dobili bismo grešku. Na slici ispod je prikazan deo konfiguracione datoteke *Ubuntu3* instance koji se odnosi na grupnu replikaciju – */etc/mysql/mysql.conf.d/mysql.cnf* datoteka.

U klasteru koristimo *gtid* zasnovanu replikaciju. *GTID* (*global transaction identifier*) je jedinstveni identifikator kreiran i dodeljen svakoj transakciji izvršenoj na izvornom serveru. Identifikator je jedinstven na nivou cele replikacione mreže i omogućava replikaciju baziranu na transakcijama kod *MySQL* baza podataka. Svaka transakcija se može pratiti od inicijalnog izvršenja na izvornom serveru, pa sve do primene iste na replikama i to bez oslanjanja na *log* datoteke prilikom pokretanja servera replika. Glavna prednost replikacije bazirane na transakcijama je mnogo lakši oporavak od greške u poređenju sa replikacijom zasnovanom na datotekama.



```
GNU nano 6.2 /etc/mysql/mysql.conf.d/mysqld.cnf
# General replication settings
disabled_storage_engines="MyISAM,BLACKHOLE,FEDERATED,ARCHIVE,MEMORY"
gtid_mode = ON
enforce_gtid_consistency = ON
master_info_repository = TABLE
relay_log_info_repository = TABLE
binlog_checksum = NONE
log_slave_updates = ON
log_bin = binlog
binlog_format = ROW
transaction_write_set_extraction = XXHASH64
loose-group_replication_bootstrap_group = OFF
loose-group_replication_start_on_boot = OFF
loose-group_replication_ssl_mode = REQUIRED
loose-group_replication_recovery_use_ssl = 1

# Shared replication group configuration
loose-group_replication_group_name = "aeee25ed-b68b-41fe-b1f4-4d3db4ffab89"
loose-group_replication_ip_whitelist = "192.168.100.78,192.168.100.75,192.168.100.79"
loose-group_replication_group_seeds = "192.168.100.78:33061,192.168.100.75:33061,192.168.100.79:33061"

# Single or Multi-primary mode? Uncomment these two lines
# for multi-primary mode, where any host can accept writes
#loose-group_replication_single_primary_mode = OFF
#loose-group_replication_enforce_update_everywhere_checks = ON

# Host specific replication configuration
server_id = 3
bind-address = "192.168.100.79"
report_host = "192.168.100.79"
loose-group_replication_local_address = "192.168.100.79:33061"

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

Slika 5.14. Deo konfiguracione datoteke koji se odnosi na grupnu replikaciju

Prvi sektor se odnosi na opšta podešavanja replikacije. Između ostalog, uključuje i onemogućavanje svih *storage engine*-a koji nisu *InnoDB*, postavljanje *gtid_mode*-a na *ON* kako bi svaka transakcija imala jedinstveni identifikator radi praćenja prilikom replikacije, primoravanje svih transakcija na poštovanje *gtid* konzistencije (*enforce_gtid_consistency: ON*), sirov (*row*) *binlog* format, zahtevani *ssl mode*, kao i druga podešavanja (lista svih podešavanja je data na slici iznad).

Drugi sektor se odnosi na deljenu konfiguraciju grupne replikacije – ovaj deo je potpuno isti u svim konfiguracionim datotekama sve 3 instance. Naziv grupe predstavlja jedinstveni, prethodno kreirani identifikator (pomoću komande *uuidgen*). Opcija koja se odnosi na *ip whitelist* kao vrednost ima niz svih *ip* adresa instanci koje čine klaster (uključujući i svoju). I na kraju, opcija koja se odnosi na *group seeds* ima vrednost niz *ip_address_of_instance:33061*, gde je 33061 port namenjen operacijama grupne replikacije.

Takođe, i ovde se moraju naći *ip* adrese svih instanci radi uspešnog izvođenja grupne replikacije.

Treći sektor je u celini zakomentarisani jer smo želeli da imamo pasivnu replikaciju, odnosno omogućen *single-primary mode*. Ukoliko se pak predomislimo i hoćemo imati aktivnu replikaciju, odnosno da omogućimo upis na svim instancama (*multi-primary mode*), dovoljno je samo skinuti komentar (obrisati # sa početka linije) sa sledeće dve linije u svim konfiguracionim datotekama:

- *#loose-group_replication_single_primary_mode = OFF*
- *#loose-group_replication_enforce_update_everywhere_checks = ON*

i *restart*-ovati *MySQL* servere komandom:

`systemctl restart mysql.service.`

Poslednji, ali ne i najmanje važan sektor koji se odnosi na grupnu replikaciju, specifičan je za samu instancu (ti delovi konfiguracije se razlikuju od instance do instance). Prva opcija je *server_id* i ona kod mašine *Ubuntu3* ima vrednost 3. Neophodno je da ove vrednosti na nivou jednog klatera budu različite. Mi smo konfigurali tako da instanca *Ubuntu1* ima *server_id* 1, a *Ubuntu2* 2. Opcije *bind-address* i *report_host* imaju vrednost *ip* adrese same instance, dok *loose-group_replication_local_address* ima vrednost *instance_ip_address:33061*, gde je 33061 port namenjen operacijama grupne replikacije.

Na posletku, kada imamo konfiguracione datoteke instanci koje omogućavaju grupnu replikaciju, ostaje nam samo ponovno učitavanje istih *restart*-ovanjem servera.

Konačno možemo dodati preostale 2 instance klasteru pozivanjem metode klastera *add_instance* sa parametrom *korisnik@host*. Za metod oporavka biramo *Clone*. Na sledeće 2 slike je prikazano uspešno dodavanje instance *clusteradmin@Ubuntu3* klasteru. Prethodno je to isto urađeno i za instancu *clusteradmin@Ubuntu2*.

```
Activities Terminal 12 Jun 18:47
root@Ubuntu1: /home/vboxuser

* Waiting for server restart... ready
* 192.168.100.75:3306 has restarted, waiting for clone to finish...
** Stage RESTART: Completed
* Clone process has finished: 73.24 MB transferred in about 1 second (~73.24 MB /s)

State recovery already finished for '192.168.100.75:3306'

The instance '192.168.100.75:3306' was successfully added to the cluster.

mysql-py []> cluster.add_instance('clusteradmin@192.168.100.79')

NOTE: The target instance '192.168.100.79:3306' has not been pre-provisioned (G
TID set is empty). The Shell is unable to decide whether incremental state reco
very can correctly provision it.
The safest and most convenient way to provision a new instance is through autom
atic clone provisioning, which will completely overwrite the state of '192.168.
100.79:3306' with a physical snapshot from an existing cluster member. To use t
his method by default, set the 'recoveryMethod' option to 'clone'.

The incremental state recovery may be safely used if you are sure all updates e
ver executed in the cluster were done with GTIDs enabled, there are no purged t
ransactions and the new instance contains the same GTID set as the cluster or a
subset of it. To use this method by default, set the 'recoveryMethod' option t
o 'incremental'.

Please select a recovery method [c]lone/[I]ncremental recovery/[A]bort (default
Clone):
```

Slika 5.15. Dodavanje instance klasteru

```
Activities Terminal 12 Jun 18:45
root@Ubuntu1: /home/vboxuser

while, you may need to manually start it back.

* Waiting for clone to finish...
NOTE: 192.168.100.75:3306 is being cloned from 192.168.100.78:3306
** Stage DROP DATA: Completed

** Clone Transfer      FILE COPY =====
==== 0% Not Started  PAGE COPY =====
===== 0% Not Started  REDO COPY =====
===== 0% Not Started** Clone Transfer      FILE COPY #####
##### 100% Completed  PAGE COPY #####
##### 100% Completed  REDO COPY #####
##### 100% Completed

NOTE: 192.168.100.75:3306 is shutting down...

* Waiting for server restart... ready
* 192.168.100.75:3306 has restarted, waiting for clone to finish...
** Stage RESTART: Completed
* Clone process has finished: 73.24 MB transferred in about 1 second (~73.24 MB /s)

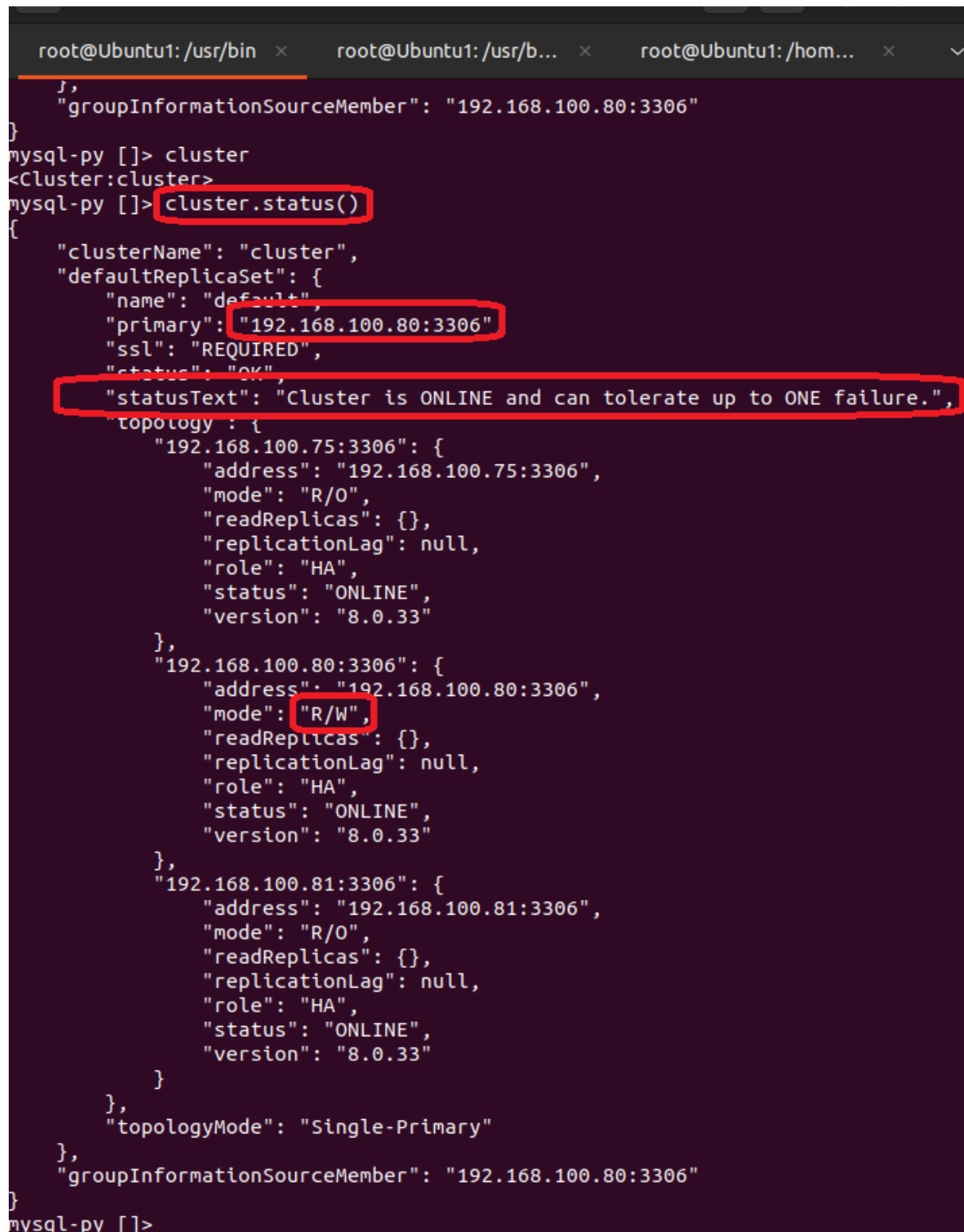
State recovery already finished for '192.168.100.75:3306'

The instance '192.168.100.75:3306' was successfully added to the cluster.

mysql-py []>
```

Slika 5.16. Instanca je uspešno dodata u klaster

U želji da proverimo status klastera nakon dodavanja instanci istom, izvršavamo metodu samog klastera *status()*. Na slici ispod je prikazan rezultat izvršenja ove funkcije.



```
root@Ubuntu1: /usr/bin x root@Ubuntu1: /usr/b... x root@Ubuntu1: /hom... x
},
"groupInformationSourceMember": "192.168.100.80:3306"
}
mysql-py []> cluster
<Cluster:cluster>
mysql-py []> cluster.status()
{
  "clusterName": "cluster",
  "defaultReplicaSet": {
    "name": "default",
    "primary": "192.168.100.80:3306",
    "ssl": "REQUIRED",
    "status": "OK",
    "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
    "topology": {
      "192.168.100.75:3306": {
        "address": "192.168.100.75:3306",
        "mode": "R/O",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      },
      "192.168.100.80:3306": {
        "address": "192.168.100.80:3306",
        "mode": "R/W",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      },
      "192.168.100.81:3306": {
        "address": "192.168.100.81:3306",
        "mode": "R/O",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      }
    }
  },
  "topologyMode": "Single-Primary"
},
"groupInformationSourceMember": "192.168.100.80:3306"
}
mysql-py []>
```

Slika 5.17. Status klastera sa 3 instance

Od bitnijih informacija koje smo dobili, vidimo da je reč o *single-primary mode*-u (*topologyMode*), tačnije da je *ip* adresa primarne instance 192.168.100.80 (*Ubuntu1*), a port je 3306 (podrazumevani za *MySQL*). U *multi-primary mode*-u ovo polje se ne prikazuje. Takođe, klaster je online (može se koristiti) i s obzirom na činjenicu da imamo 3 instance u istom, otporan je na maksimalno 1 grešku – ukoliko dođe do iste, što ćemo kasnije i demonstrirati, klaster nastavlja sa radom, ali više nije otporan ni na jednu grešku. Očito, data je i sama topologija klastera sa detaljima o svakoj instanci pojedinačno. Jedino primarna instanca (192.168.100.80) je u *R/W* režimu – omogućeno i čitanje i pisanje, dok su ostale instance u *R/O* (*read-only*) *mode*-u – moguće samo čitanje. *Role*-a je *HA* (*high-availability*) – reč je o visokoj dostupnosti što je veoma pogodno kod relanih sistema. Svi serveri su dostupni (*online*) i verzije 8.0.33 (aktuelna verzija *MySQL*-a).

5.3. *MySQL Router*

MySQL Router je jedna od tehnologija koja se koristi prilikom rada sa *InnoDB* klasterom i omogućava transparentno rutiranje između aplikacije i samog klastera. *InnoDB* klaster je čvrsto spregnut sa *MySQL Router*-om. Pristupa se pomoću *AdminAPI*-a radi vršenja dodatne administracije. *MySQL Router* se automatski konfiguriše na osnovu *InnoDB* klastera u procesu koji se naziva *bootstrapping* i samim tim nema potrebe za ručnom konfiguracijom. Transparentno povezuje klijentske aplikacije sa klasterom obezbeđujući rutiranje i balansiranje opterećenja. Automatski detektuje promene nastale unutar klastera i ceo sistem nastavlja sa skladnim funkcionisanjem.

MySQL Router najpre treba instalirati. Biramo mašinu *Ubuntu1* (na njoj smo kreirali i klaster) i izvršavamo sledeće naredbe kao privilegovani korisnik:

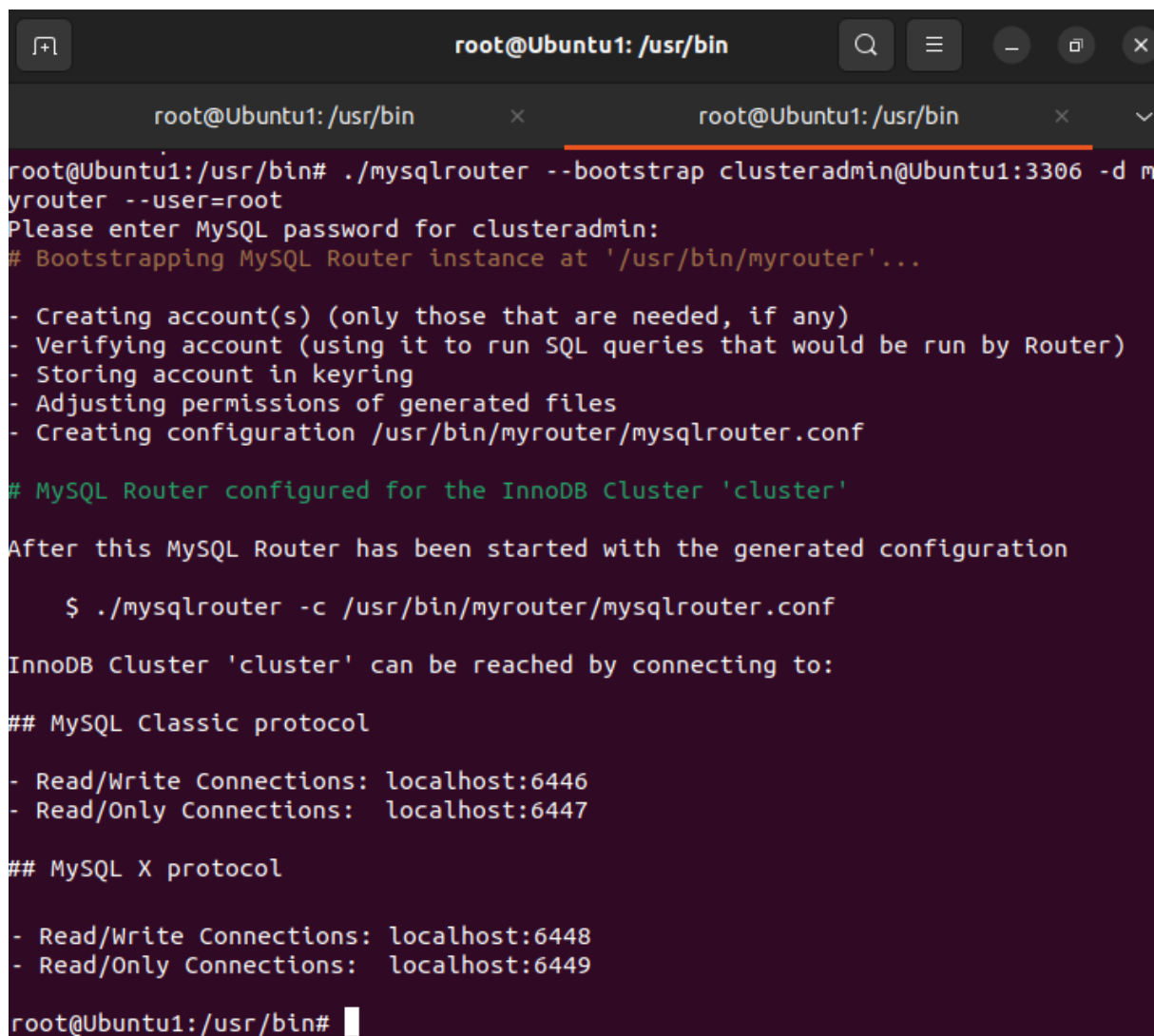
```
sudo apt-get update
sudo apt-get install mysql-router.
```

S obzirom da nisam znao gde se instalacija nalazi, morao sam izvršiti i naredbu

```
dpkg -L mysql-router
```

radi pronalaska lokacije instalacije samog paketa. Saznajem da se isti nalazi na putanji *usr/bin/mysqlrouter*.

Vršimo prethodno opisani *bootstrapping* proces (prikazano na slici ispod). Dobijamo informaciju da ukoliko želimo samo da čitamo (*read-only*), našem klasteru pristupamo korišćenjem adrese *localhost:6447* za *MySQL Classic protocol*, odnosno *localhost:6449* za *MySQL X protocol*. Upis u klaster se može vršiti korišćenjem *localhost:6446* za *MySQL Classic protocol*, odnosno *localhost:6448* za *MySQL X protocol*. Naravno na portu 6446 (odnosno 6448) se može vršiti i čitanje sa klastera.



```
root@Ubuntu1: /usr/bin
root@Ubuntu1: /usr/bin# ./mysqlrouter --bootstrap clusteradmin@Ubuntu1:3306 -d myrouter --user=root
Please enter MySQL password for clusteradmin:
# Bootstrapping MySQL Router instance at '/usr/bin/myrouter'...

- Creating account(s) (only those that are needed, if any)
- Verifying account (using it to run SQL queries that would be run by Router)
- Storing account in keyring
- Adjusting permissions of generated files
- Creating configuration /usr/bin/myrouter/mysqlrouter.conf

# MySQL Router configured for the InnoDB Cluster 'cluster'

After this MySQL Router has been started with the generated configuration

$ ./mysqlrouter -c /usr/bin/myrouter/mysqlrouter.conf

InnoDB Cluster 'cluster' can be reached by connecting to:

## MySQL Classic protocol

- Read/Write Connections: localhost:6446
- Read/Only Connections: localhost:6447

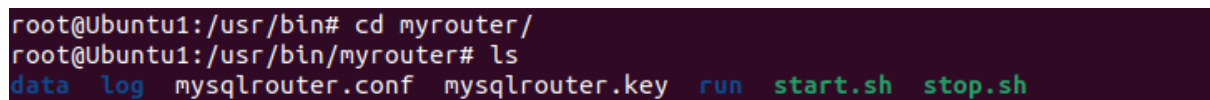
## MySQL X protocol

- Read/Write Connections: localhost:6448
- Read/Only Connections: localhost:6449

root@Ubuntu1: /usr/bin#
```

Slika 5.18. *Bootstrapping* – automatska konfiguracija *MySQL Router*-a

Sadržaj direktorijuma *mysqlrouter* je dat u nastavku:



```
root@Ubuntu1: /usr/bin# cd myrouter/
root@Ubuntu1: /usr/bin/myrouter# ls
data  log  mysqlrouter.conf  mysqlrouter.key  run  start.sh  stop.sh
```

Datoteka *mysqlrouter.conf* je automatski kreirana na osnovu konfiguracije samog *InnoDB* klastera tokom procesa *bootstrapping* i njen sadržaj je dat u nastavku:

```
# File automatically generated during MySQL Router bootstrap
[DEFAULT]
user=root
logging_folder=/usr/bin/myrouter/log
runtime_folder=/usr/bin/myrouter/run
data_folder=/usr/bin/myrouter/data
```

keyring_path=/usr/bin/myrouter/data/keyring
master_key_path=/usr/bin/myrouter/mysqlrouter.key
connect_timeout=5
read_timeout=30
dynamic_state=/usr/bin/myrouter/data/state.json
client_ssl_cert=/usr/bin/myrouter/data/router-cert.pem
client_ssl_key=/usr/bin/myrouter/data/router-key.pem
client_ssl_mode=PREFERRED
server_ssl_mode=AS_CLIENT
server_ssl_verify=DISABLED
unknown_config_option=error

[logger]
level=INFO

[metadata_cache:bootstrap]
cluster_type=gr
router_id=1
user=mysql_router1_qhycwg26hral
metadata_cluster=cluster
ttl=0.5
auth_cache_ttl=-1
auth_cache_refresh_interval=2
use_gr_notifications=0

[routing:bootstrap_rw]
bind_address=0.0.0.0
bind_port=6446
destinations=metadata-cache://cluster/?role=PRIMARY
routing_strategy=first-available
protocol=classic

[routing:bootstrap_ro]
bind_address=0.0.0.0
bind_port=6447
destinations=metadata-cache://cluster/?role=SECONDARY
routing_strategy=round-robin-with-fallback
protocol=classic

[routing:bootstrap_x_rw]
bind_address=0.0.0.0
bind_port=6448
destinations=metadata-cache://cluster/?role=PRIMARY
routing_strategy=first-available

protocol=x

[routing:bootstrap_x_ro]

bind_address=0.0.0.0

bind_port=6449

destinations=metadata-cache://cluster/?role=SECONDARY

routing_strategy=round-robin-with-fallback

protocol=x

[http_server]

port=8443

ssl=1

ssl_cert=/usr/bin/myrouter/data/router-cert.pem

ssl_key=/usr/bin/myrouter/data/router-key.pem

[http_auth_realm:default_auth_realm]

backend=default_auth_backend

method=basic

name=default_realm

[rest_router]

require_realm=default_auth_realm

[rest_api]

[http_auth_backend:default_auth_backend]

backend=metadata_cache

[rest_routing]

require_realm=default_auth_realm

[rest_metadata_cache]

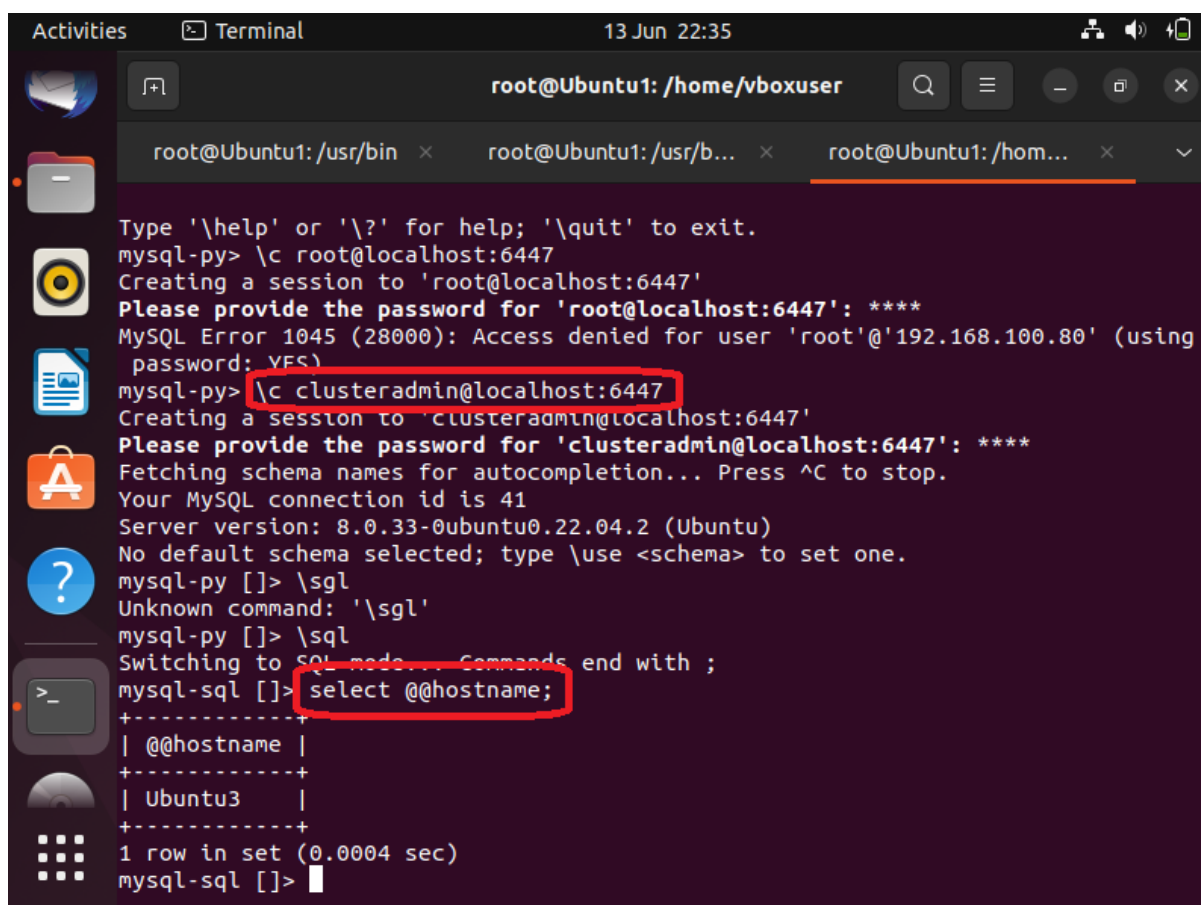
require_realm=default_auth_realm

U *mysqlrouter.conf* datoteci vidimo vrednosti pojedinih opcija za *log-ovanje*, *time-out-e*, sertifikate, metapodatke o samom *router-u*,...

Rad *MySQL Router*-a (sam proces) započinjemo izvršenjem *start.sh* skripte što je prikazano u nastavku:

```
root@Ubuntu1:/usr/bin/myrouter# ./start.sh
root@Ubuntu1:/usr/bin/myrouter# PID 4430 written to '/usr/bin/myrouter/mysqlrouter.pid'
stopping to log to the console. Continuing to log to filelog
```

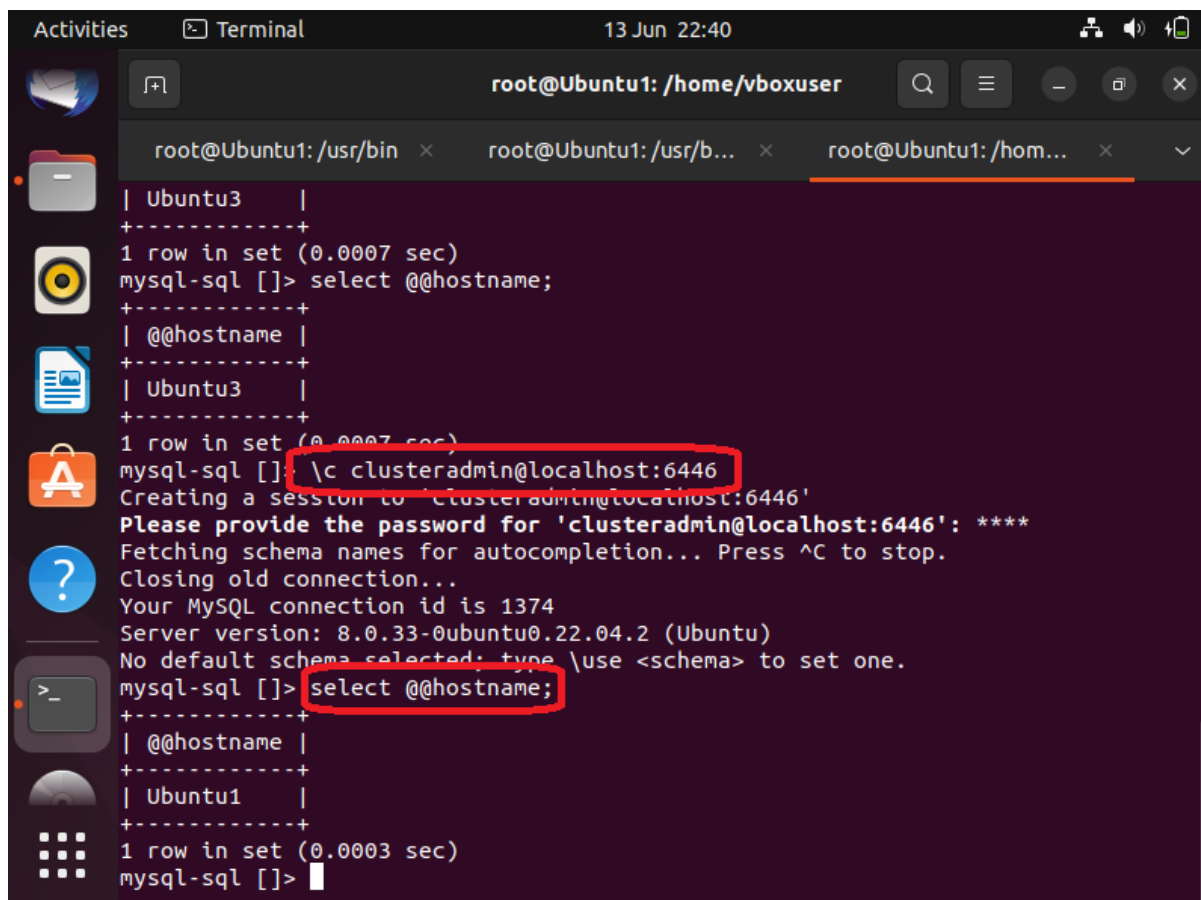
Na slici ispod je prikazano kreiranje konekcije na *localhost:6447* koristeći *MySQL Shell*. Nakon promene režima (sa *mysql-py* na *mysql-sql*) izvršavamo *select @@hostname* naredbu. Rezultat izvršenja je *Ubuntu3* jer smo povezani na port koji dozvoljava samo čitanje. Rezultat izvršenja je mogao biti i *Ubuntu2*, ali nikako *Ubuntu1* koji je primarna instanca (nije *read-only*).



```
Activities Terminal 13 Jun 22:35
root@Ubuntu1: /home/vboxuser
root@Ubuntu1:/usr/bin x root@Ubuntu1:/usr/b... x root@Ubuntu1:/hom... x
Type '\help' or '? for help; \quit' to exit.
mysql-py> \c root@localhost:6447
Creating a session to 'root@localhost:6447'
Please provide the password for 'root@localhost:6447': ****
MySQL Error 1045 (28000): Access denied for user 'root'@'192.168.100.80' (using
password: YES)
mysql-py> \c clusteradmin@localhost:6447
Creating a session to 'clusteradmin@localhost:6447'
Please provide the password for 'clusteradmin@localhost:6447': ****
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 41
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)
No default schema selected; type \use <schema> to set one.
mysql-py []> \sql
Unknown command: '\sql'
mysql-py []> \sql
Switching to SQL mode... Commands end with ;
mysql-sql []> select @@hostname;
+-----+
| @@hostname |
+-----+
| Ubuntu3    |
+-----+
1 row in set (0.0004 sec)
mysql-sql []>
```

Slika 5.19. Instanca zadužena za *read-only* operacije

Ukoliko želimo nešto da upišemo na klaster, moramo se povezati na port 6446. Izvršenjem prethodne naredbe (*select @@hostname*) dobijamo kao rezultat instancu *Ubuntu1* jer je ona jedina instanca na kojoj se može vršiti upis (primarna instanca) koji se kasnije propagira na sve ostale instance u klasteru radi sinhronizacije (kako bi sve instance bile konzistentne – imale iste vrednosti). Idealni slučaj je kada vršimo čitanje na bilo kojoj instanci u okviru klastera i to odmah nakon upisa na primarnoj instanci i kao rezultat uvek dobijamo poslednju upisanu vrednost. Naravno i ovde postoji tolerancija na kašnjenje kroz samu mrežu, koje je inače veoma malo jer se radi o lokalnoj mreži.



```
root@Ubuntu1: /home/vboxuser
root@Ubuntu1: /usr/bin x root@Ubuntu1: /usr/b... x root@Ubuntu1: /hom... x
| Ubuntu3 |
+-----+
1 row in set (0.0007 sec)
mysql-sql []> select @@hostname;
+-----+
| @@hostname |
+-----+
| Ubuntu3 |
+-----+
1 row in set (0.0007 sec)
mysql-sql []> \c clusteradmin@localhost:6446
Creating a session to 'clusteradmin@localhost:6446'
Please provide the password for 'clusteradmin@localhost:6446': ****
Fetching schema names for autocompletion... Press ^C to stop.
Closing old connection...
Your MySQL connection id is 1374
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)
No default schema selected; type \use <schema> to set one.
mysql-sql []> select @@hostname;
+-----+
| @@hostname |
+-----+
| Ubuntu1 |
+-----+
1 row in set (0.0003 sec)
mysql-sql []>
```

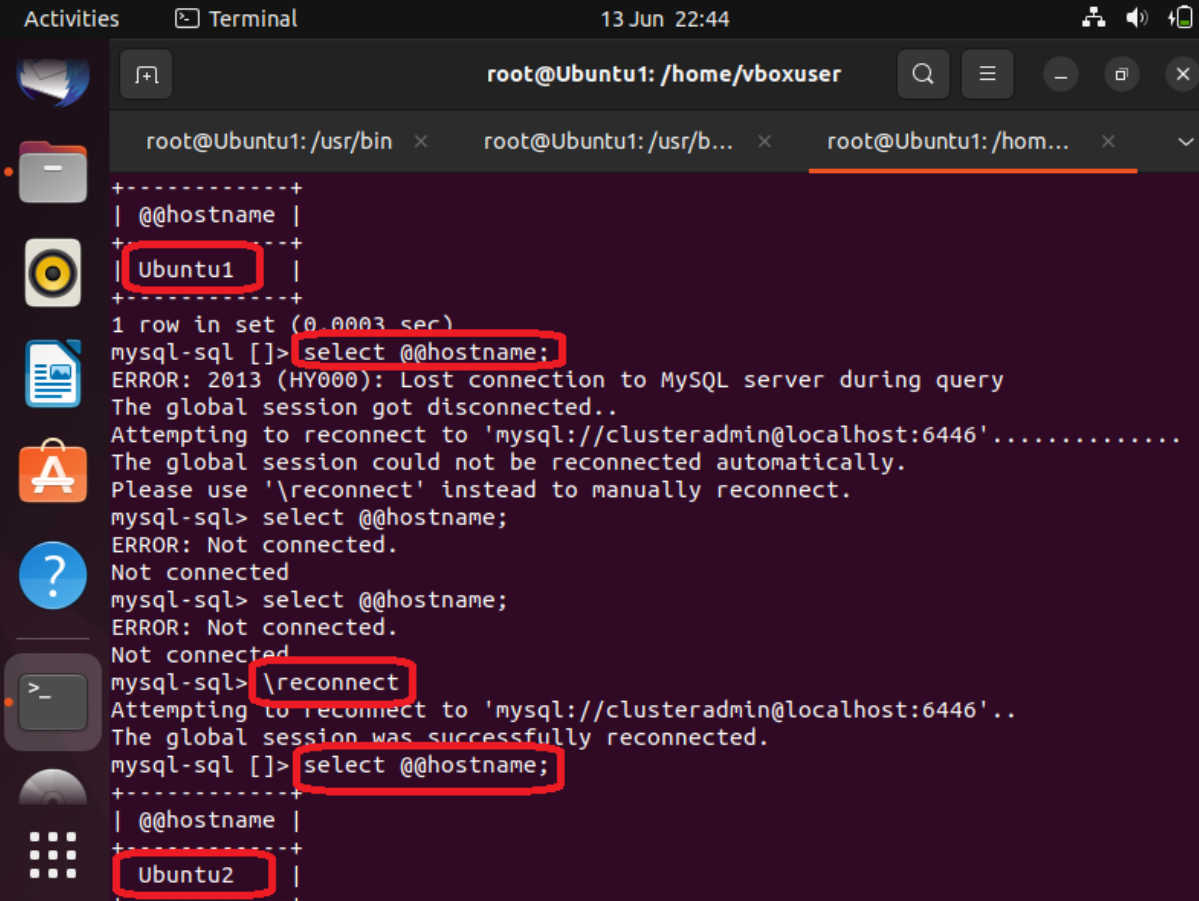
Slika 5.20. Instanca zadužena za operacije pisanja i čitanja

Mi se u nastavku obraćamo isključivo *MySQL Router*-u (aplikacija se povezuje sa istim) koji tačno zna kome prosleđuje koji tip naredbe, vodeći računa o opterećenju samih instanci, kao i tipu operacije koji instanca podržava (sa ciljem što boljeg rada čitavog sistema). U slučaju eventualnog otkaza neke baze podataka (nekog servera), sistem nastavlja sa radom uz eventualno malo veće zagušenje i samim tim i kašnjenje, ali nemamo obustavu rada – imamo visoku dostupnost i otpornost na greške. Klaster se u opisanom slučaju sam rekonfiguriše (biće prikazano u nastavku rada) i *MySQL Router* prima to k znanju radi ispravnog vršenja rutiranja. Baza podataka nam nije više *single point of failure* (*MySQL Router* nažalost postaje *SPOF*). Po potrebi možemo dodavati nove servere u klaster kako nam broj korisnika aplikacije bude rastao – imamo skalabilno rešenje.

5.4. Otkaz jednog od *MySQL* servera

Izenada, u jednom trenutku otkazuje jedna instanca (tačnije improvizujemo izvršavanjem komande na primarnoj instanci za stopiranje *MySQL* servisa). S obzirom da je reč o instanci na kojoj je jedino bio dozvoljen upis, a konekcija sa serverom se izgubila, sam klaster (s obzirom da se upis trenutno ne može vršiti) pokreće izbor za novu primarnu

instancu radi normalnog funkcionisanja klastera u nastavku. Sistem mora nastaviti sa nesmetanim radom. Na slici ispod vidimo pokušaj dobijanja instance zadužene za upis na klaster, ali bezuspešan – konekcija je izgubljena. Ponovo se povezujemo (`\reconnect`) i ovoga puta dobijamo naziv *host*-a zaduženog za *write/read* operacije. Zaključujemo da je nakon izbora, nova primarna instanca *Ubuntu2*. Mašina *Ubuntu3* ima istu ulogu kao i do sada – *read/only* replika.



The screenshot shows a terminal window titled 'Terminal' with the date '13 Jun 22:44'. The user is logged in as 'root' on 'Ubuntu1' with the path '/home/vboxuser'. The terminal displays the output of a MySQL query: 'select @@hostname;'. The result is 'Ubuntu1'. Below this, there is an error message: 'ERROR: 2013 (HY000): Lost connection to MySQL server during query'. The message continues: 'The global session got disconnected.. Attempting to reconnect to 'mysql://clusteradmin@localhost:6446'..... The global session could not be reconnected automatically. Please use '\reconnect' instead to manually reconnect.' The user then enters 'mysql-sql> select @@hostname;' and receives 'ERROR: Not connected.' and 'Not connected'. The user then enters 'mysql-sql> \reconnect' and receives 'Attempting to reconnect to 'mysql://clusteradmin@localhost:6446'.. The global session was successfully reconnected.' Finally, the user enters 'mysql-sql []> select @@hostname;' and receives the result 'Ubuntu2'.

```
root@Ubuntu1: /home/vboxuser
root@Ubuntu1: /usr/bin x root@Ubuntu1: /usr/b... x root@Ubuntu1: /hom... x
+-----+
| @@hostname |
+-----+
| Ubuntu1    |
+-----+
1 row in set (0.0003 sec)
mysql-sql []> select @@hostname;
ERROR: 2013 (HY000): Lost connection to MySQL server during query
The global session got disconnected..
Attempting to reconnect to 'mysql://clusteradmin@localhost:6446'.....
The global session could not be reconnected automatically.
Please use '\reconnect' instead to manually reconnect.
mysql-sql> select @@hostname;
ERROR: Not connected.
Not connected
mysql-sql> select @@hostname;
ERROR: Not connected.
Not connected
mysql-sql> \reconnect
Attempting to reconnect to 'mysql://clusteradmin@localhost:6446'..
The global session was successfully reconnected.
mysql-sql []> select @@hostname;
+-----+
| @@hostname |
+-----+
| Ubuntu2    |
+-----+
```

Slika 5.21. Otkaz primarne instance i izbor nove

U želji da vršimo nadgledanje klastera, povezujemo se na novu primarnu instancu. U promenljivu *cluster* učitavamo novu referencu klastera (`dba.get_cluster()`) i izvršavamo naredbu `cluster.status()`. Novi status klastera upozorava da isti više nije otporan ni na jednu grešku jer 1 od 3 instance nije aktivna. Vidimo da je *Ubuntu2* (192.168.100.75) primarna replika – režim *R/W* (mogući i upis). Takođe da se primetiti da je *Ubuntu1* instanca (192.168.100.75) u režimu *n/a*, tačnije nije *online*. Postoji greška, a status ove instance je *MISSING*. Klaster nastavlja sa radom (malo će ove dve replike imati više posla nego do sada, ali nemamo prekid u radu – velika prednost klasterizacije).

```
root@Ubuntu1: /usr/bin
mysql-py> \c clusteradmin@192.168.100.75
Creating a session to 'clusteradmin@192.168.100.75'
Please provide the password for 'clusteradmin@192.168.100.75': ****
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 535
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)
No default schema selected; type \use <schema> to set one.
mysql-py []> cluster=dba.get_cluster()
mysql-py []> cluster.status()
{
  "clusterName": "cluster",
  "defaultReplicaSet": {
    "name": "default",
    "primary": "192.168.100.75:3306",
    "ssl": "REQUIRED",
    "status": "OK_NO_TOLERANCE",
    "statusText": "Cluster is NOT tolerant to any failures. 1 member is not active.",
    "topology": {
      "192.168.100.75:3306": {
        "address": "192.168.100.75:3306",
        "mode": "R/W",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      },
      "192.168.100.80:3306": {
        "address": "192.168.100.80:3306",
        "mode": "n/a",
        "readReplicas": {},
        "role": "HA",
        "shellConnectError": "MySQL Error 2003 (HY000): Can't connect to MySQL server on '192.168.100.80' (111)",
        "status": "(MISSING)"
      },
      "192.168.100.81:3306": {
        "address": "192.168.100.81:3306",
        "mode": "R/O",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      }
    },
    "topologyMode": "Single-Primary"
  },
  "groupInformationSourceMember": "192.168.100.75:3306"
}
mysql-py []>
```

Slika 5.22. Instanca zadužena za operacije pisanja i čitanja

Nakon oporavka *MySQL* servera (ponovnim pokretanjem istog), *Ubuntu1* se „vraća“ u klaster (tačnije ponovo je *online*), nastavlja sa radom, ali je jako bitno napomenuti da više nije primarna replika (bez obzira što je ponovo aktivan), jer je tu ulogu preuzela instanca *Ubuntu2* (nakon pređašnjeg izbora). *Ubuntu1* je sada sekundarna replika – instanca u *read-only* režimu. Klaster je ponovo tolerantan na 1 grešku/otkaz.



```
mysql-py []> cluster.status()
{
  "clusterName": "cluster",
  "defaultReplicaSet": {
    "name": "default",
    "primary": "192.168.100.75:3306",
    "ssl": "REQUIRED",
    "status": "OK",
    "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
    "topology": {
      "192.168.100.75:3306": {
        "address": "192.168.100.75:3306",
        "mode": "R/W",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      },
      "192.168.100.80:3306": {
        "address": "192.168.100.80:3306",
        "mode": "R/O",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      },
      "192.168.100.81:3306": {
        "address": "192.168.100.81:3306",
        "mode": "R/O",
        "readReplicas": {},
        "replicationLag": null,
        "role": "HA",
        "status": "ONLINE",
        "version": "8.0.33"
      }
    },
    "topologyMode": "Single-Primary"
  },
  "groupInformationSourceMember": "192.168.100.75:3306"
}
mysql-py []>
```

Slika 5.23. *Ubuntu1* ponovo *online*, ali sada u *read/only mode*-u

6. ZAKLJUČAK

Na kraju rada, ostalo je izvesti zaključke iz celokupnog istraživanja date teme. Klasterizacija ima ogromnu primenu u produkciji (u realnim aplikacijama), a s obzirom da je *MySQL* jedna od najpopularnijih baza podataka, *InnoDB* klaster svakako igra bitnu ulogu. Proučavajući *InnoDB* klaster, naučio sam kako funkcionišu mnoge stvari „pod haubom“, video probleme sa kojima se sistem suočava i načine na koje ih rešava. Administracija i održavanje instanci je malo zahtevniji posao, ali kada se pogledaju dobici (proširljivost sistema, visoka dostupnost, otpornost na greške,...), očitio isplativ. Istražujući ovu temu, stekao sam veliko praktično znanje koje će mi bez sumnje značiti u budućem radu.

LITERATURA

- [1] *MySQL*, <https://dev.mysql.com/> , jun 2023.
- [2] *MySQL InnoDB Cluster*,
<https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-innodb-cluster.html> , jun 2023.
- [3] *How To Configure MySQL Group Replication on Ubuntu*
<https://www.digitalocean.com/community/tutorials/how-to-configure-mysql-group-replication-on-ubuntu-20-04> , jun 2023.