UNIVEZITET U NIŠU

ELEKTRONSKI FAKULTET

Seminarski rad

Klaster rešenja u MySQL

Predmet: Sistemi za upravljanje bazama podataka

Mentor: Aleksandar Stanimirović

Student: Marija Stojanović 1034

Niš, jun 2020. god

Sadržaj:

1. Uvod

Baza podataka prestavlja neizostavni deo svake aplikacije koji je spreman da skladišti i dostavlja informacije i podatke kojima će se kroz alplikaciju manipulisati. Server baze podataka sam za sebe može biti back-end deo aplikacije baze podataka, a može biti i računarska oprema. S vremena na vreme može biti kombinacija oba, tj kombinacija programa i računarske opreme. Bilo kako bilo, server baza podataka je vrlo važan deo koji održava funkcionalnosti aplikacija zbog čega se veliki deo današnjih firmi okreću ka grupisanju baza podataka.

Klasterovanje baza podataka je proces koji kombinuje više od jedne serverske instance koji se povezuju na istu bazu podataka. Ponekada jedan server može biti neadekvatan za upravljanje velikom količinom podataka ili velikim brojem upita koji su potrebni, upravo je to trenutak u kome klaster baze podataka postaje neophodan. Klasterovanje baza podataka, klasterovanje SQL servera kao i klasterovanje SQL-a je usko povezano sa SQL jezikom koji se koristi za manipulaciju nad podacima u bazi podataka.[5]

MySQL Klaster je tehnologija koja omogućava nedeljivo klasterovanje MySQL sistema za upravljanje bazama podataka. Dizajnirana je da omogući visoku dostupnost kao i visoku propusnost sa jako malim kašnjenjima, a takođe omogućava i skoro linearnu skalabilnost. [1]
MySQL Klaster je izgrađen na NDB mehanizmu za skladištenje koji omogućava visoko skalabilnu. real-time, ACID propustljivu, transakcionu bazu podataka koja kombinuje 99.999% dostupnosti sa niskim TCO. Dizajniran je za rad nad distribuiranom, multi-master arhitekturom bez mogućnosti da ona otkaže prilikom nepravilnosti u radu jedne od tački iz celokupne arhitekture, MySQL klaster skalira se horizontalno na robnom hardveru kako bi izdržao intenzivna radna opterećenja koja se javljaju prilikom čitanja i upisivanja podataka u baze, a koja se obavljaju putem SQL i NoSQL interfejsa. [2]

2. Klasterovanje baza podataka

Glavni razlozi klasterovanja baze podataka su pre svega prednosti koje server dobija. Ove prednosti su: redundantnost podataka, balansiranje opterećenja, visoka dostupnost i na kraju mogućnost nadgledanja i automatizacije.

Redundantnost podataka

Više računara sarađuje kako bi uzajamno skladištili podatke pomoću klasterovanja baza podataka. Ovo omogućava prednost redundantnosti podataka. Svi računari su sinhronizovani što znači da svaki nod ima iste podatke kao i svi ostali nodovi. U bazi podataka, potrebno je izbeći neke vrste ponavljanja (redundantnosti) jer one dovode do zabuna u samim podacima. U slučaju da jedan računar zakaže, imaćemo na raspolaganju sve podatke na drugim računarima.[5]

Balansiranje opterećenja

Balansiranje opterećenja ili skalabilnost ne dolaze kao podrazumevani uz bazu podataka. Ovo mora biti omogućeno upravo klasterovanjem. Takođe ovo zavisi i od samih podešavanja. U osnovi, balansiranje opterećenja postiže se raspoređivanjem opterećenja između različitih računara koji su deo klastera. Ovo ukazuje na to da je omogućena podrška za više istovremenih korisnika baze podataka, tako da iako se pojavi veći skok u sabraćaju, postoji veća sigurnost da će ovakav veći saobraćaj biti podržan. Jedna mašina tj. računar neće morati da podnese sav taj saobraćaj. Ovo može da obezbedi skaliranje bez problema. Takođe ovo se direktno povezuje sa velikom dostupnošću baze podataka. Bez balansiranja opterećenja, određena mašina bi mogla biti preopterećena i saobraćaj bi se usporio, što dovodi do smanjenja saobraćaja do nule. [5]

Visoka dostupnost

Ako smo u mogućnosti da pristupimo bazi podataka, podrazumeva se da je ona dostupna. Visoka dostupnost se odnosi na to da je većinu vremena baza podataka konstantno dostupna. Količina dostupnosti koja je potrebna u velikoj meri zavisi od broja transakcija koje se izvršavaju nad datom bazom podataka kao i koliko se često vrši bilo kakva analiza podataka. Sa klasterovanjem baza podataka, može se postići ekstremna dostupnost zahvaljujući balansiranju opterećenja i dodavanjem dodatnih mašina u klaster. U slučaju da se server isključi, baza podataka će u svakom slučaju biti dostupna. [5]

Nadgledanje i automatizacija

Uobičajna baza podataka se može koristiti za ovu vrstu pogodnosti zato što nadgledanje i automatizacija mogu biti odrađene na vrlo jednostavan način pomoću softvera. Naravno ova pogodnost je mnogo veća i izraženija ako je prisutan klaster. Tipično, pogodnost je ta da klasterovanje dopušta automatizaciju velikog broja procesa nad bazom podataka u istovremeno sa postavljanjem pravila radi upozoravanja na potencijalne probleme. Ovo omogućava korisniku da se ne mora vraćati ručno unazad kako bi proverio. Sa klasterovanom bazom podataka, automatizacija je od pomoći zato što omogućava da se dobijaju obaveštenja ukoliko korisnik previše zahteva od sistema. Međutim klaster baze podataka će imati određenu mašinu koja će se koristiti kao sistem za upravljanje bazom podataka. Kontrolna tabla za ceo klaster. Ova odabrana mašina imala bi skripte koje se automatski pokreću za čitav klaster baze podataka i rade sa svim nodovima baze podataka.[5]

3. Arhitektura MySQL klastera

MySQL klaster projektovan je nad distribuiranom, multi-master, ACID arhitekturom koja nema nijednu tačku otkaza ili greške. On koristi auto-shareding (particionisanje) kako bi skalirao operacije čitanja i upisa na tržišnom (commodity) hardveru i može mu se pristupati putem SQL i Non-SQL API-ja.

Replikacija

Interno, MySQL klaster koristi sinhronu replikaciju kroz dvofazni mehanizam za komitovanje kako bi se garantovalo da su podaci upisani na više nodova u trenutku nakon što su podaci komitovani. Ovo je svakako u suprotnosti sa uobičajenom MySQL replikacijom koja je asinhrona. Dve kopije podataka (replike) su potrebne kako bi dostupnost podataka bila zagarantovana. MySQL klaster automatski kreira grupe nodova od broja replika kao i data nodova koji su specificirani od strane korisnika. Ažuriranja se sinhrono replikuju među članovima grupe nodova kako bi zaštitili podatke od gubljenja i kako bi omogućili brži oporavak od kvarova i problema koji nastaju među nodovima.

Takođe, moguće je obaviti asinhronu replikaciju među klasterima, što je poznato pod MySQL replikacijom klastera ili geografskom replikacijom. Ovo se obično koristi kako bi se vršila replikacija klastera među data centrima zbog mogućnosti reparacije nakon neke katastrofe ili kako bi se redukovali efekti mrežnog kašnjenja lociranjem podataka fizički bliže korisnicima istih. Za razliku od standardnih MySQL replikacija, geografska replikacija MySQL klastera koristi optimističnu kontrolu konkurentnosti kao i kocept Epoha kako bi omogućio mehanizam za detekciju i rešavanje konflikata, omogućavajući aktovno/aktivno klasterovanje među data centrima.

Horizontalno particionisanje podataka (auto-shareding)

MySQL klaster je implementiran kao potpuno distribuirana multi-master baza podataka koja omogućava da sva ažuriranja odrađena od strane bilo koje aplikacije ili SQL noda momentalno budu dostupna svim nodovima koji pistupaju klasteru, i svaki data nod može prihvatiti operacije upisa podataka.

Podaci u tabelama MySQL klastera (NDB) se automatski particionišu kroz sve data nodove u sistemu. Ovo je se bazira na algoritmu za heširanje koji se oslanja na primarni ključ tabele i transparentan je kranjoj aplikaciji. Klijenti se mogu povezati na bilo koji od nodova u klasteru i njihovi upiti automatski pristupaju ispravnim delovima potrebnim da zadovolje upit ili izvrše transakciju.

Korisnici mogu definisati svoje šeme za particionisanje. Ovo omogućava developerima da daju “svest o distribuiranosti” aplikacijama tako što podele vrše na osnovu podključa koji je zajednički za sve redove kojima pristupaju transakcije. Ovo omogućava da su podaci koji su korišćeni da bi se kompletirale transakcije lokalizovani u istom delu, čime se redukuju mrežni skokovi.

Hibridno skladištenje

MySQL klaster omogućava setove podataka koji su veći od kapaciteta svake od mašina na kojima mogu biti skladišteni kao i kapaciteta za pristup istim kroz više mašina.

MySQL čuva sve indeksirane kolone u distribuiranoj memoriji ili na disku koji ima keš mememoriju. Čuvanje neindeksiranih kolona na disku dopušta MySQL klasteru da skladišti datasetove koji su veći nego memorija koja je agregirana u klasterovanim mašinama.
MySQL upisuje Redo logove na disk za sve izmene u podacima a takođe i redovno proverava podatke na disku. Ovo omogućava klasteru da se konstantno oporavlja sa diska nakon potpunog prekida rada klastera. Trenutno podrazumevano zakašnjenje asinhronih upisa je 2 sekunde, i moguće je konfigurisati ga. Normalne scenariji u kome se dešavaju tačke pucanja ne rezultiraju nikakvim gubitkom podataka zahvaljujući sinhronoj replikaciji podataka u klasteru.

Kada se tabela MySQL klastera čuva u memoriji, klaster će jedino pristupiti disku kako bi se upisali Redo rekordi i checkpointi. Kako su ovi upisi sekvencijalni i limitirani obrasci random pristupa, MySQL klaster može postići veću brzinu protoka upisa na ograničenom hardveru diska u poređenju sa tradicionalnim disk baziranim keširanjem RDBMS.

Shared nothing

MySQL klaster projektovan je tako da nema nijednu tačku otkaza. Ako se omogući pravilno postavljanje klastera, svaki pojedinačni nod, sistem i deo hardvera može otkazati bez da ceo klaster otkaže zbog toga. Deljivi disk (SAN) nije neophodan. Konekcije među nodovima može biti standardna Ethernet, Gigabit Ethernet, InfiniBand ili SCI.

SQL i NoSQL API

Kako MySQl klaster čuva tabele u data nodovima, a ne na MySQL serveru, dostupno je više interfejsa za pristup bazi podataka:

- SQL pristup preko MySQL servera
- NoSQL API-ji u kojima MySQL klaster biblioteke mogu biti ugrađene u aplikaciju kako bi se omogućio direktan pristup data nodovima bez prolaženja kroz sloj SQL-a. Oni uključuju:

    - Memcached
    - Node.js / JavaScript
    - Java and JPA
    - HTTP / REST
    - NDB API (C++)

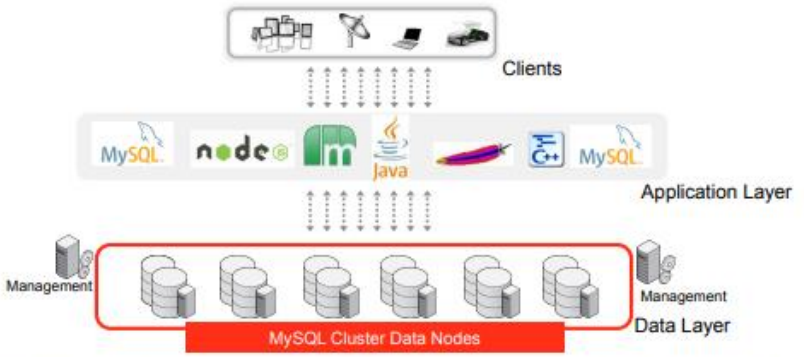


Figure 1. *The MySQL Cluster architecture is designed for high scalability and 99.999% availability with SQL and NoSQL APIs*

MySQL Cluster Manager

Deo komercijalnog MySQL Cluster CGE, MySQL Cluster Manager je alat projektovan kako bi se uprostilo kreiranje i administracija MySQL Cluster CGE baze podataka automatizovanjem svakodnevnih upravljačkih taskova, uključujući online skaliranje, ažuriranje, backup-restore i rekonfiguracije. MySQL Cluster Manager takođe nadgleda i automatski obnavlja MySQL serverske aplikacione nodove i upravljačke nodove, kao i MySQL Cluster data nodove.

4. Implementacija

- **Data node:** U ovim nodovima se čuvaju podaci. Tabele se automatski dele među data nodovima što takođe transparentno upravlja balansiranjem opterećenja, replikacijom, otkazima i samooporavakom.
- **Management node:** Koristi se za konfiguraciju i nadgledanje klastera. Oni su neophodni jedino za startovanje i restartovanje klastera. Takođe mogu biti kokfigurisani kao arbitri, ali to nije obavezno (umesto toga, MySQL serveri mogu biti konfigurisani kao arbitri)
- **SQL node:** MySQL server koji se konektuje na sve data nodove kako bi obavio skladištenje i preuzimanje podataka. Ovaj tip noda je opcionalni; moguće je vršiti upite na data nodove direktno kroz NDB API, ili jednostavno koristiti C++ API ili neki dodatni NoSQL API koji je prethodno opisan.

Generalno očekivano je da se svaki od nodova pokreće na posebnom fizičkom hostu, virtuelnoj ili cloud instanci. Najpreporučljivije je da se nodovi ne lociraju u okviru jedne iste grupe nodova na jednom fizičkom hostu jer bi tako mogli biti predstavljeni kao jedna tačka otkaza.

Zahtevi

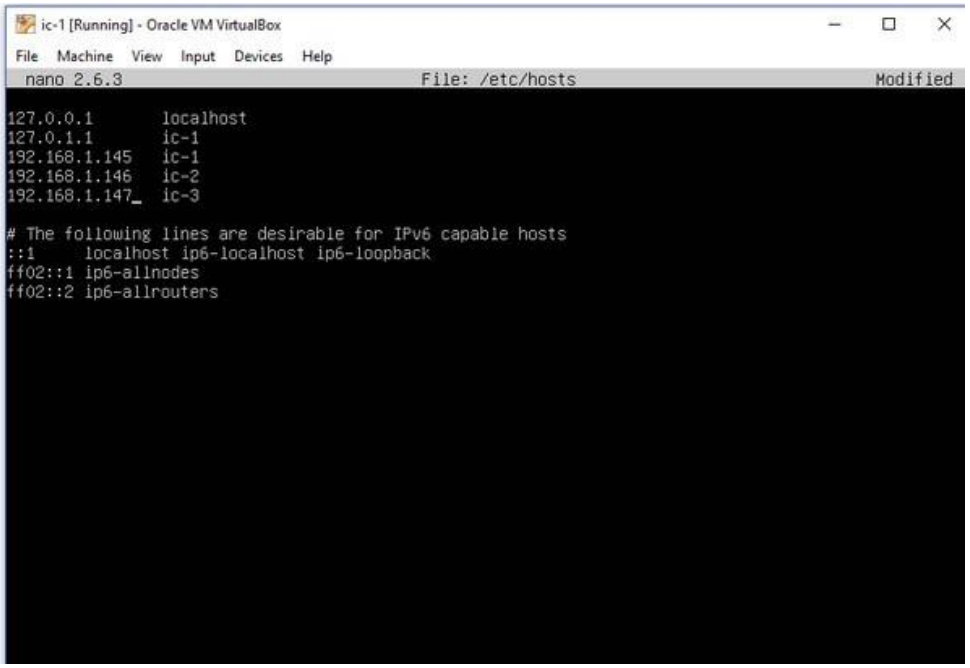
Za potrebe evaluacije i developmenta, mogu se svi nodovi pokrenuti na jednom hostu. Za potpunu redundantnost i toleranciju ng greške, potrebno je miimum 6 fizičkih hostova: [4]

- 2 x data noda
- 2 x SQL/NoSQL aplikaciona noda
- 2 x management noda

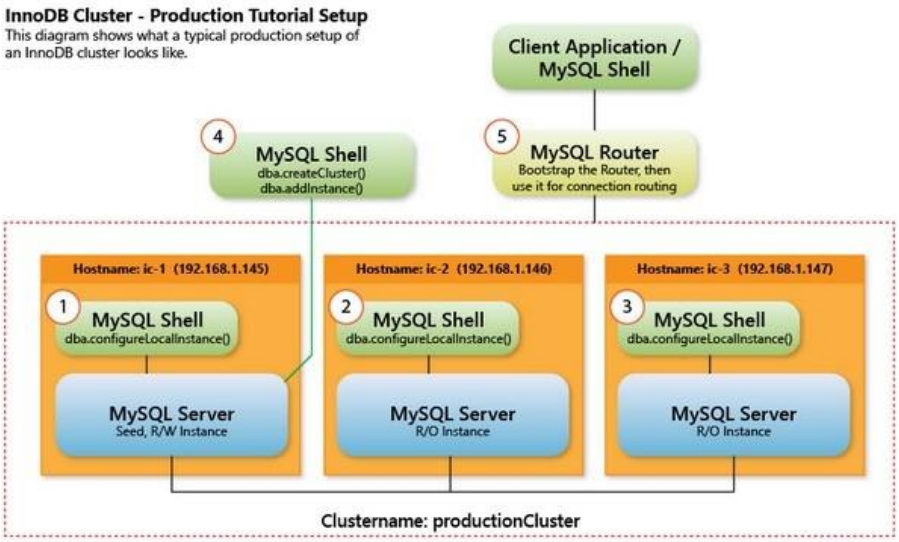
Takođe je moguće locirati nodove za upravljanje (management nodes) i aplikacijske nodove na istom mestu što smanjuje broj čvorova na 4.[4]

5. Primer formiranja MySQL InnoDB klastera

Polazimo od pretpostavke da su postavljene 3 instance Ubuntu virtuelne mašine sa instaliranim MySQL serverom kao i MySQL Shell-om. Na prvoj, drugoj i trećoj virtuelnoj masini konfigurisane su sledeće IP adrese i hostname-ovi kao na slici.



Sada imamo tri virtuelne mašine spremne da budu deo InnoDB klastera. U nastavku, pokazaćemo kako se može kreirati klaster i koristiti MySQL Router kako bi se mapirali svi upiti ka bazi podataka.
Na sledećoj slici prikazana je realna struktura InnoDB klastera. U realnom svetu postavka bi bila na konkretnim serverima dok su u našem prikazu korišćene virtuelne masine radi demonstracije realne strukture.



Pošto se u realnom svetu za servere obično koristi Linux operativni sistem ovde prikazujemo postavku u njemu. Kako bismo pravilno konfigurisali InnoDB klaster potrebno je na svakoj od virtuelnih mašina omogućiti administratorske privilegije sledećom sudo komandom:

```
$ su
$ apt-get install sudo
```

```
$ sudo nano /etc/sudoers
```

Da bismo koristili prethodno pomenuti MySQL Shell na svakoj od virtuelnih mašina potrebno je instalirati Python

```
$ sudo apt-get install python
```

Zatim je potrebno instalirati MySQL APT repository koji omogućava dev pakete za instalaciju i upravljanje MySQL servera, klijenata i ostalih komponenti na trenutnim verzijama Debian i Ubuntu distribucijama.

```
$ sudo wget http://dev.mysql.com/get/mysql-apt-config_0.8.4-1_all.deb
$ sudo dpkg -i ./mysql-apt-config_0.8.4-1_all.deb
```

```
$ sudo apt-get update
```

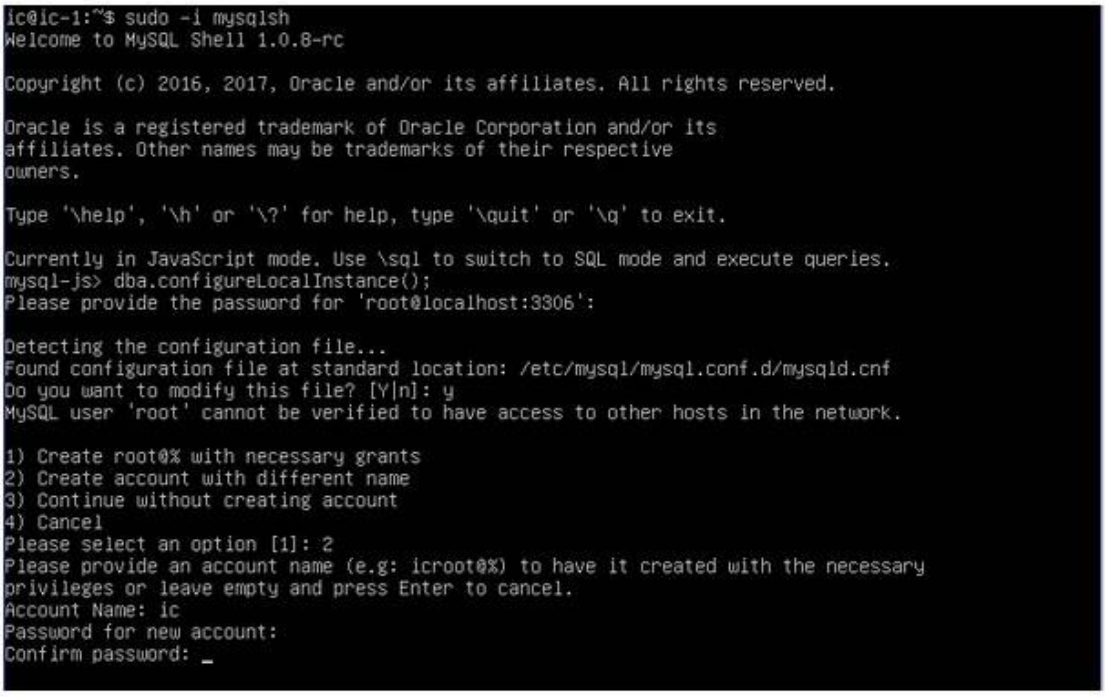
Nakon instalacije potrebno je pokrenuti MySQL Shell na svakoj od virtuelnih mašina, a zatim konfigurisati lokalne instance.

```
$ sudo -i mysqlsh
```

```
mysql-js> dba.configureLocalInstance();
```

MySQL Shell će pronaći podrazumevani konfiguracioni fajl i pitaće da li želimo da ga modifikujemo. Potrebno je odgovoriti unosom “Y”. Obzirom da za root korisnika nije omogućen remote login, imamo tri opcije za nastavak konfiguracije:

1) Omogućiti remote login za root korisnika
2) Kreirati novog korisnika
3) Ne kreirati novog korisnika i ne omoguciti remote login za root korisnika



Mi ćemo izabrati opciju 2), tj. kreiranje novog korisnika. Pojaviće se izvestaj sa promenama koje napravljene od strane MySQL Shell-a i poruka koja kaže da potrebno restartovati MySQL servis kako bi se one primenile.

Izaći iz MySQL Shell-a komandom i restartovati MySQL servis:

mysql-js> \q

```
$ sudo systemctl restart mysql.service
```

U ovom trenutku, host je spreman da bude deo InnoDB klastera.

Sledeći korak je instalacija MySQL Router-a koji nam omogućava da sakrijemo konfiguraciju mreže preko proxy servera i mapiramo upite nad bazama podataka na klaster.
Uobičajeno, MySQL Router je instaliran na klijentskoj mašini. Međutim, za potrebe ovor rada mi ćemo ga instalirati na jednoj od naših virtuelnih masina.

U terminalu izvršićemo sledeću komandu:

```
$ sudo apt-get install mysql-router
```

Kada je instalacija MySQL Router-a završena, možemo kreirati klaster.

SSL

Pre nego što nastavimo dalje, potrebno je da se osvrnemo na važne aspekte koji se odnose na sigurnost.

Ako se koristimo MySQL Server Community edition, moguće je da nemamo omogućen SSL na našem serveru. Možemo to proveriti sledećom komandom u terminalu:

```
$ mysql -u root -p --execute="show variables like '%have%ssl%'"
```

Verifikacija se vrši proverom vrednosti promenljivih “have_ openssl” i “have_ssl”. Ukoliko je vrednost “YES” imamo uključen SSL, u suprotnom ga nemamo. U slučaju da nemamo SSL uključen i želimo da ga omogućimo na serveru možemo koristiti mysql_ssl_rsa_setup alatku. Mysql_ssl_rsa_setup alatka generiše SSL sertifikat i fajlove sa ključevima, kao i RSA key-pair fajlove koji su potrebni za podršku sigurnosnih konekcija koristeći SSL. Jedini uslov za njegovo korišćenje je posedovanje openssl binary u promenljivama okruženja (path environment). To se može obaviti sledećom komandom:

```
$ mysql_ssl_rsa_setup --datadir=mydir
```

Vrednost “--datadir” opcije je mesto gde će se sertifikat kreirati, što bi trebalo da bude MySQL folder sa podacima. Ukoliko ta opcija nije data, mysql_ssl_rsa_setup, on koristi podrazumevani MySQL kompajlirani folder sa podacima kada je MySQL Server instaliran ili specificiran u odgovarajući mysqld.cnf fajl.

Ukoliko se odlučilo da se omogući SSL, pri kreiranju klastera i dodavanju instanci potrebno je omoguciti **ipWhitelist** opcioni parametar kako bi se specificirao raspon IP adresa kojima će biti

dozvoljeno konektovanje na klaster. Vrednost ipWhiteList promenljive je podrazumevano postavljen na “AUTOMATIC”, sto dozvoljava konekcije samo od privatnih podmreža na hostu gde je klaster kreiran.

Takođe možemo specificirati SSL mod koji će biti korišćen da se konfigurišu članovi klastera koristeći memberSslMode parametar. Postoje sledeće opcije:

- REQUIRED: Ukoliko se koristi, SSL (enkripcija) će biti uključena za instance da komuniciraju sa ostalim članovima klastera.
- DISABLED: Ukoliko se koristi, SSL (enkripcija) će biti isključena.
- AUTO: Ukoliko se koristi, SSL (enkripcija) će biti uključena ukoliko je podržano od strane instance, u suprotnom će biti isključena

Podrazumevano se postavlja na “AUTO” ukoliko drugačije nije specificirano.

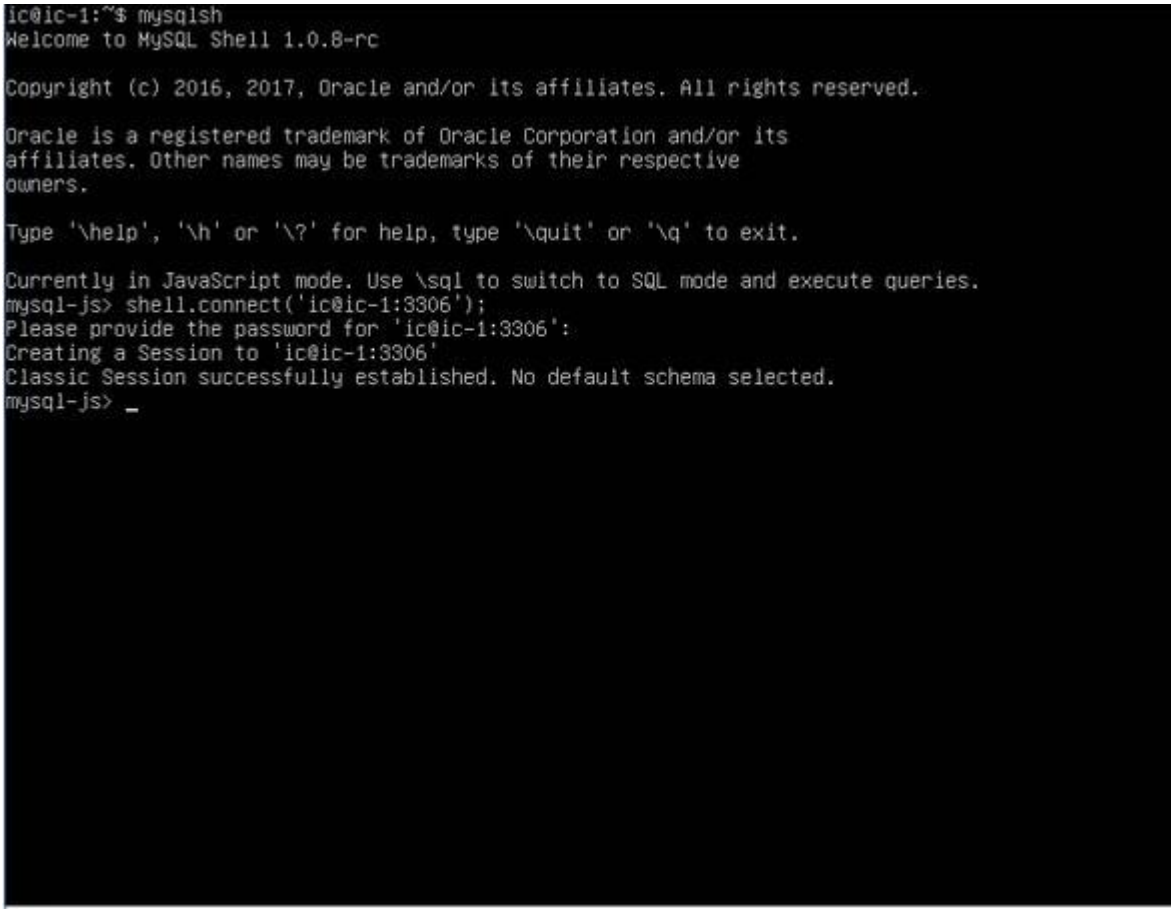
Kreiranje InnoDB klastera

Pokrenuti MySQL Shell sledecom komandom:

```
$ mysqlsh
```

Zatim kreirati classic sesiju do hosta koristeci korisnika kreiranog u konfiguracionom koraku I hostname hosta:

```
mysql-js> shell.connect('ic@ic-1:3306');
```

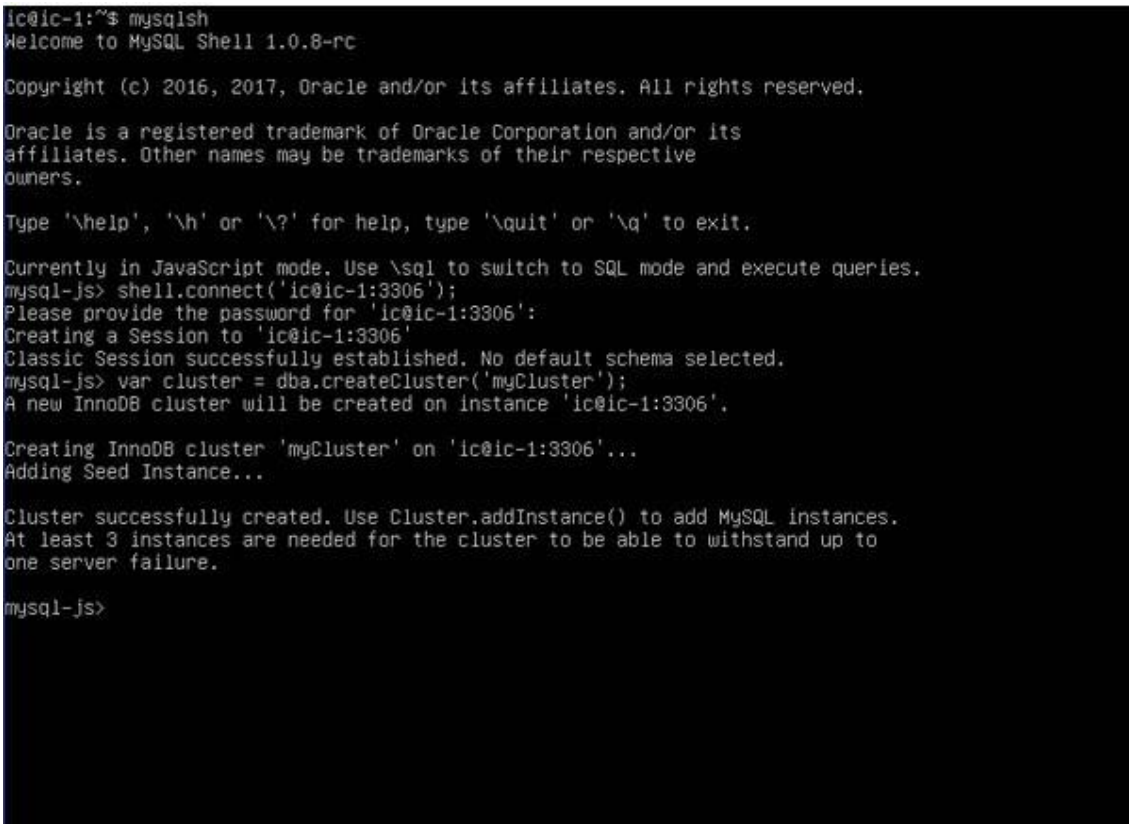


Sada kreiramo klaster dodeljivanjem povratne vrednosti promenljivoj za kasnije korišćenje:

```
mysql-js> var cluster = dba.createCluster('myCluster');
```

Ne smemo zaboraviti da koristimo ipWhitelist opcije u slučaju da je SSL uključen.

Pojaviće se nekoliko poruka sa infomacijom o kreiranju klastera i funkcijom potrebnom da se dodaju instance u klasteru:



U sledećem primeru klaster je kreiran korišćenjem ipWhiteList opcije i postavljanjem SSL kao potrebnog za sve članove klastera:

mysql-js> dba.createCluster('myCluster', { memberSslMode: 'REQUIRED',
ipWhitelist:'10.196.0.0/16,127.0.0.1/8'});

Prva adresa u opsegu IP adresa je adresa našeg hosta, druga IP adresa je localhost. Možemo dodati više IP opsega razdvajajući ih zarezom. Broj nakon / je broj mrežnih bitova kada se koristi CIDR vrednost.

Sledeća komanda je primer korišćenja ipWhiteList I memberSslMode opcija za dodavanje novih instanci klasteru:

mysql-js> cluster.addInstance('user@host:3306', { memberSslMode: 'REQUIRED',
ipWhitelist:'10.196.0.0/16,127.0.0.1/8'});

Ne smemo zaboraviti da postavimo istu memberSslMode vrednost prilikom kreiranja klastera ukoliko nije postavljen na ‘AUTO’ kada pokušavamo da dodamo novu instancu klasteru.

Jednom kada je klaster kreiran, može se videti status klastera pozivanjem sledece funkcije:

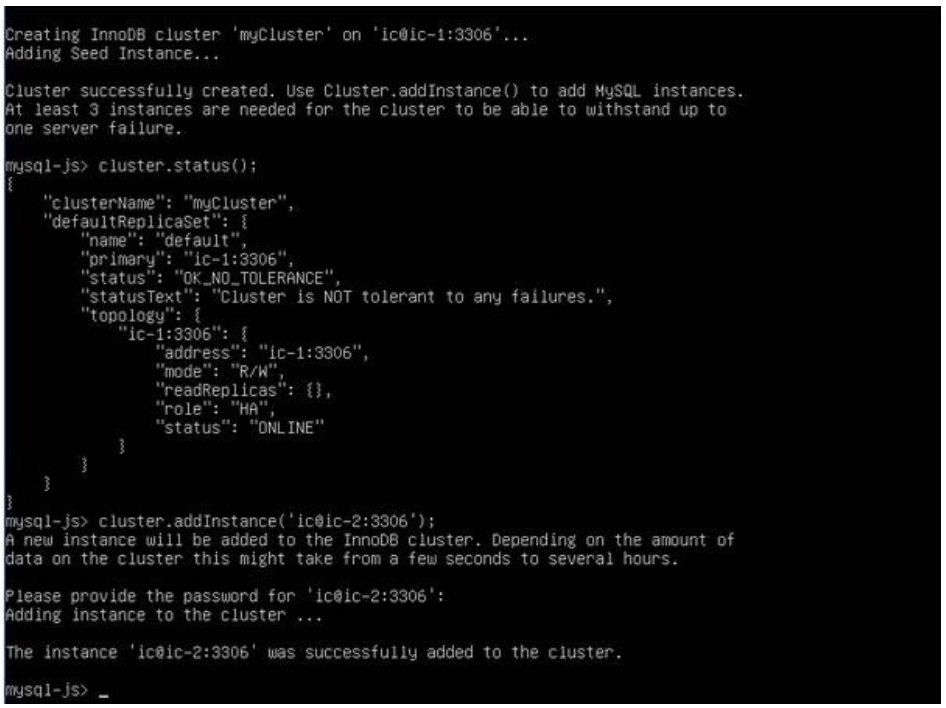
```
mysql-js> cluster.status();
```

```
Type '\help', '\h' or '\?' for help, type '\quit' or '\q' to exit.

Currently in JavaScript mode. Use \sql to switch to SQL mode and execute queries.
mysql-js> shell.connect('ic@ic-1:3306');
Please provide the password for 'ic@ic-1:3306':
Creating a Session to 'ic@ic-1:3306'
Classic Session successfully established. No default schema selected.
mysql-js> var cluster = dba.createCluster('myCluster');
A new InnoDB cluster will be created on instance 'ic@ic-1:3306'.

Creating InnoDB cluster 'myCluster' on 'ic@ic-1:3306'...
Adding Seed Instance...

Cluster successfully created. Use Cluster.addInstance() to add MySQL instances.
At least 3 instances are needed for the cluster to be able to withstand up to
one server failure.

mysql-js> cluster.status();
{
    "clusterName": "myCluster",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "ic-1:3306",
        "status": "OK_NO_TOLERANCE",
        "statusText": "Cluster is NOT tolerant to any failures.",
        "topology": {
            "ic-1:3306": {
                "address": "ic-1:3306",
                "mode": "R/W",
                "readReplicas": {},
                "role": "HA",
                "status": "ONLINE"
            }
        }
    }
}
mysql-js>
```

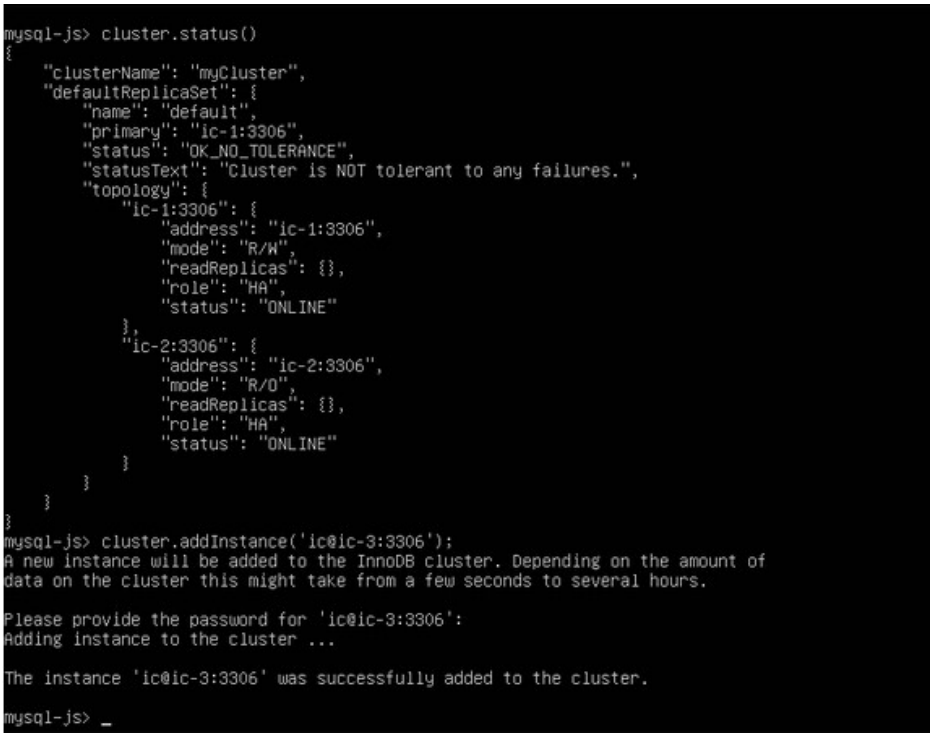
Za dodavanje novih instanci potrebno je koristiti sledeću komandu, kako bi bili sigurni da koristimo validnog korisnika i IP adresu već konfigurisanog hosta:

Ne smemo zaboraviti da koristimo ipWhiteList ukoliko je SSL uključen!

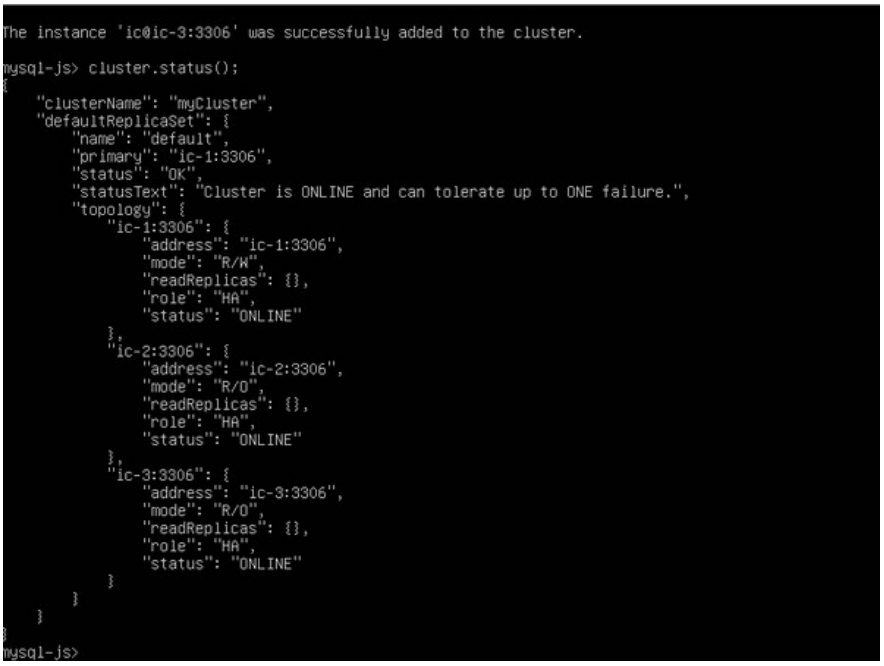
```
mysql-js> cluster.addInstance('ic@ic-2:3306');
```



Unesite šifru za korisnika kada se to traži. Moguće je dodati koliko god hosta želimo u klaster, ali imajući na umu da je potrebno najmanje tri kako bi se postigla tolerancija na otkaze.



Ovde možemo videti da su sve instance u klasteru online:



Konfiguracija perzistentnog klastera

Kako bismo omogućili perzistenciju konfigurisanog klastera za svaku instancu, tako da ukoliko se restart dogodi, instance se automatski dodaju u klaster, moramo koristiti dba.configureLocalInstance() ponovo na svakoj instanci. Komanda će ažurirati my.cnf fajlove sa parametrima potrebnim za automatsko dodavanje u klaster pri pokretanju klastera.

Unosimo sledeće komande, lokalno na svakoj od instanci:

```
mysql-js> dba.configureLocalInstance('ic@ic1-:3306');
```

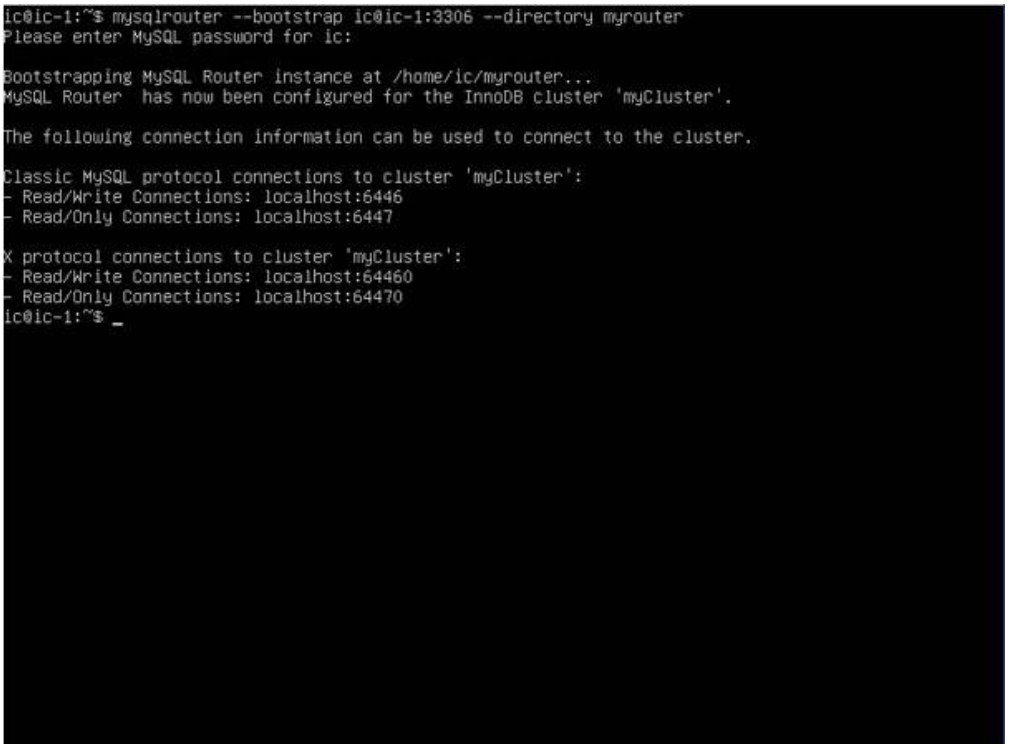
```
mysql-js> dba.configureLocalInstance('ic@ic-2:3306');
```

```
mysql-js> dba.configureLocalInstance('ic@ic-3:3306');
```

Korišćenje MySQL Router-a

Sada je vreme da odradimo bootstrap Router-a. Otvoriti novi terminal i uneti sledecu komandu i uneti šifru korisnika kada bude zahtevano:

```
$ mysqlrouter --bootstrap ic@ic-1:3306 --directory myrouter
```

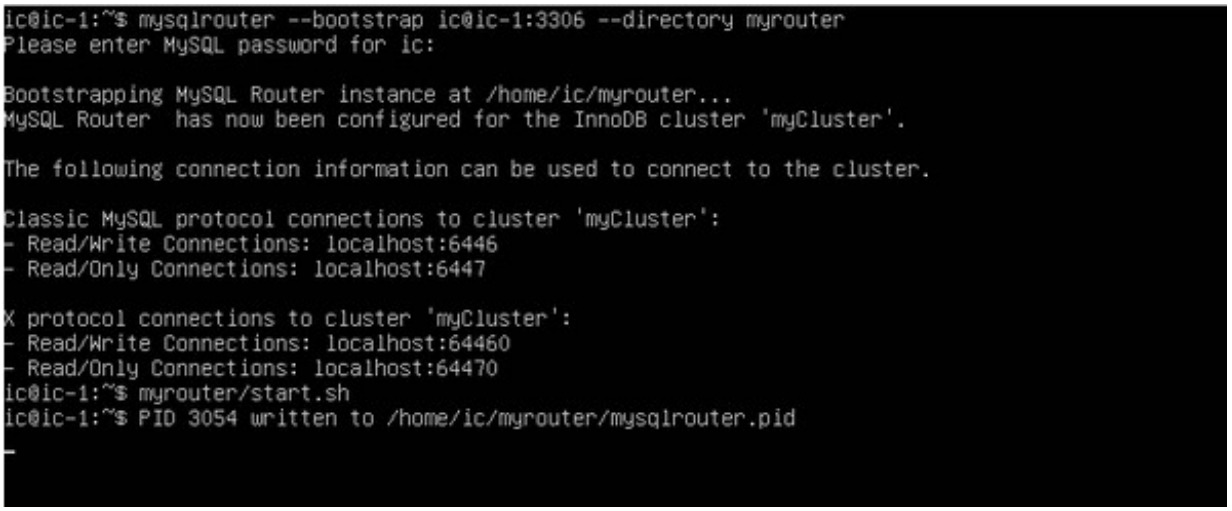


Prethodnim komandama postignuto je sledeće:

- Specifična konfiguracija za klaster “myCluster” je kreirana, MySQL Router je konektovan na klaster i ekstraktovani su metapodaci za pokretanje.
- Folder pod nazivom “myrouter” je kreiran u “home” folderu i sadrži konfiguraciju potrebnu za pokretanje MySQL Router-a.
- Četiri TCP porta su generisana za konektovanje na klaster: read only i read-write za klasičan protokol i za X protokol.

Za pokretanje MySQL Router-a unosimo sledeću komandu:

```
$ myrouter/start.sh
```



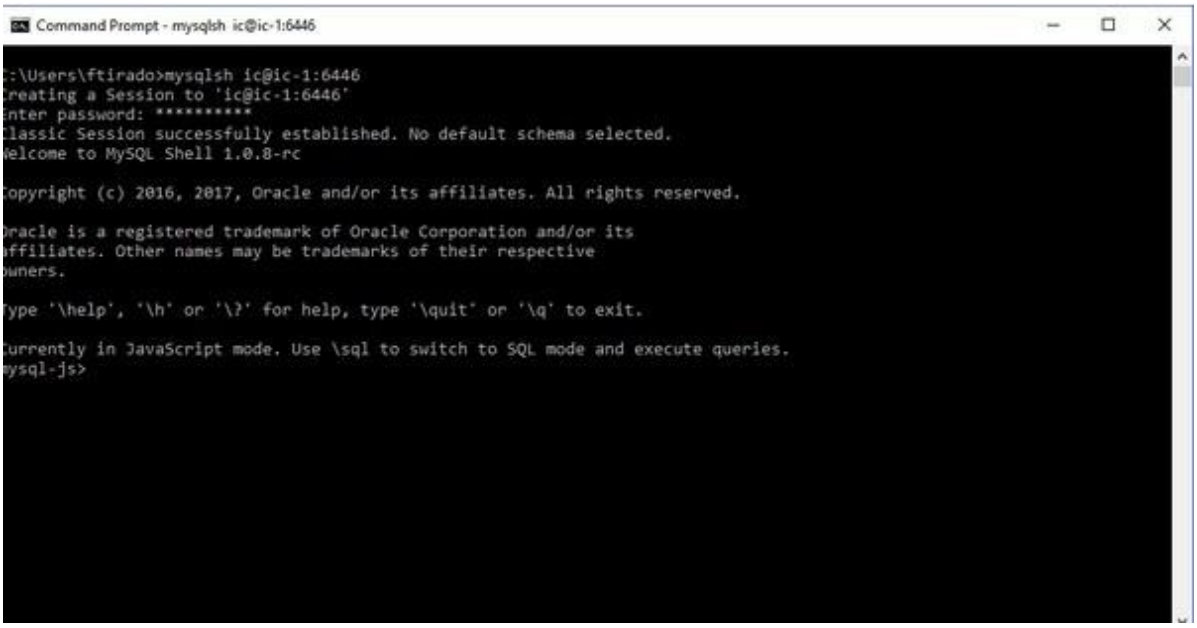
Za zaustavljanje MySQL Router-a u terminalu pokrecemo stop skriptu sledecom komandom:

```
$ myrouter/stop.sh
```

Remote Konekcija

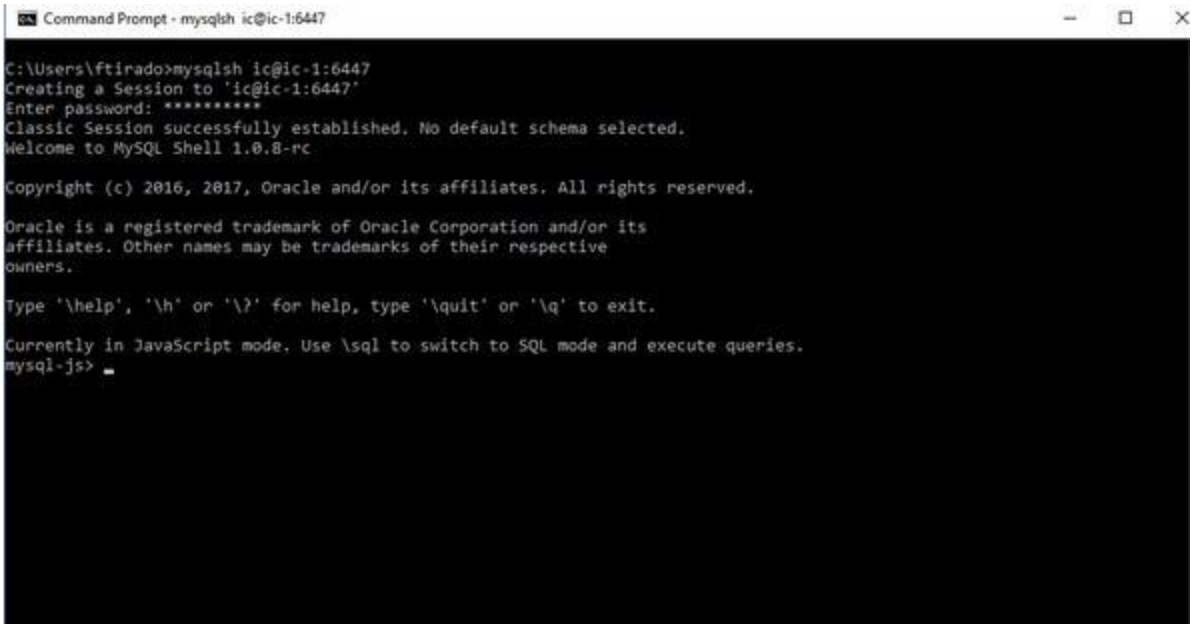
Sada se možemo konektovati na klaster korišćenjem IP adrese generisanom od strane MySQL Router-a. Sledeća slika prikazuje Windows host koji je konektovan na klaster korišćenjem read/write porta:

```
$ mysqlsh ic@ic-1:6446
```

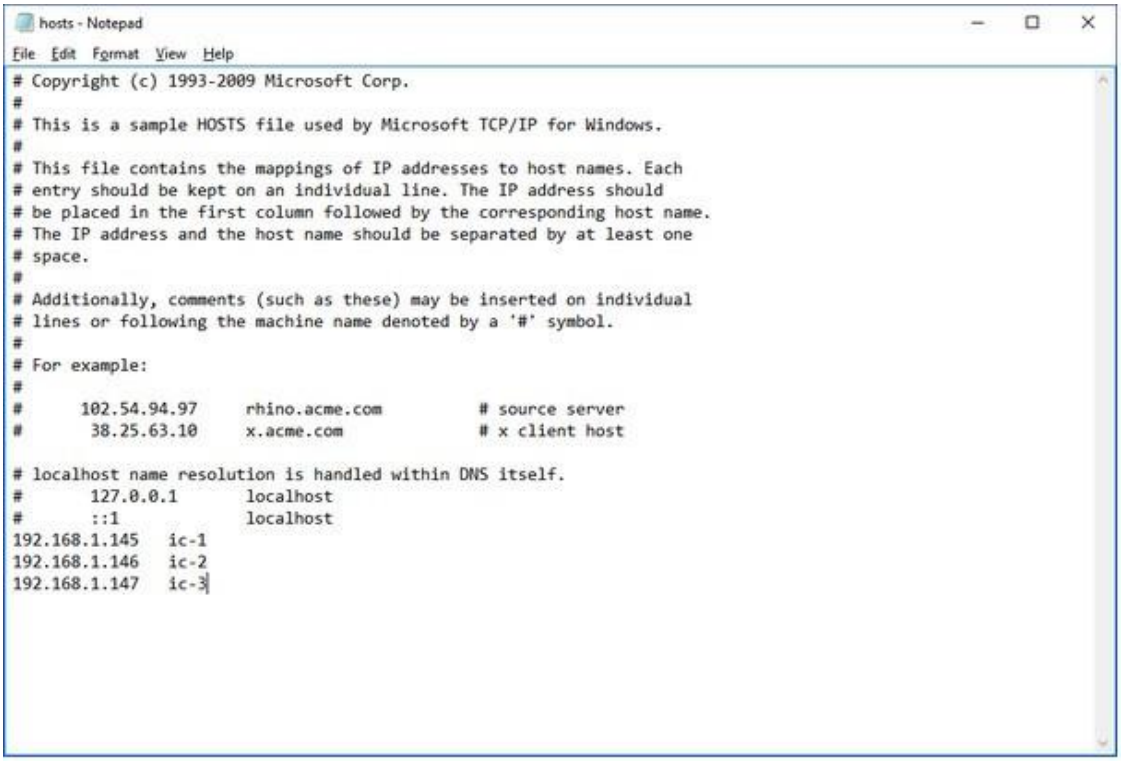


Sledeća slika prikazuje Windows host koji je konektovan na klaster korišćenjem read-only porta:

```
$ mysqlsh ic@ic-1:6447
```



Da bismo se konektovali na remote host korišćenjem njegovog imena, potrebno je takođe konfigurisati mapiranje host-a u Windows-u. Fajl koji je potrebno izmeniti nalazi se u direktorijumu “C:\ Windows\System32\drivers\etc\hosts”. Jednom kada se konfiguriše mapiranje hosta fajl bi trebalo da izgleda ovako:



6. Zaključak

Kroz prethodna poglavlja objasnili smo šta je to klasterovanje baza podataka i šta je to MySQL klaster. Naveli smo i koji su to razlozi i prednosti formiranja klastera baza podataka. Prošli smo i kroz samu arhitekturu i način funkcionisanja MySQL klastera i na kraju dali primer formiranja jednog MySQL InnoDB klastera.
Iako je prilično jasno da klasterovanje baza podataka ima velike benefite za održavanje, manipulaciju i dostupnost baza podataka aplikacijama, možemo reći da klasteri baza podataka iako su odlično rešenje za neke slučajeve mogu biti i sasvim nepotrebni, pa čak i suvišan deo. Potrebno je voditi računa o tome kada, zašto, kako i u koje svrhe koristiti ovakvu vrstu tehnologije.

7. Literatura

[1] Web sajt: https://en.wikipedia.org/wiki/MySQL_Cluster

[2] Web-sajt: https://www.mysql.com/products/cluster/faq.html

[3] Web-sajt: https://towardsdatascience.com/high-availability-mysql-cluster-with-load-balancing-using-haproxy-and-heartbeat-40a16e134691

[4] Web sajt: https://www.mysql.com/products/cluster/faq.html

Web sajt: https://ndimensionz.com/kb/what-is-database-clustering-introduction-and-brief-explanation/#:~:text=Database%20Clustering%20is%20the%20process,a%20Data%20Cluster%20is%20needed

[5] Web sajt: https://www.mysql.com/products/cluster/mysql-cluster-datasheet.pdf

[6] Web sajt: https://mysqlserverteam.com/mysql-innodb-cluster-real-world-cluster-tutorial-for-ubuntu-and-debian/?fbclid=IwAR1QIEdencoO32ZWzNHfK6LXbDbFxAHMEn8o9le2eQ_q6oi0gtaPLQWhb1s