**요구사항 정의서**

**목차**

1. 프로젝트 개요
   1. 프로젝트 개요
      1. 프로젝트 소개
      2. 프로젝트 배경
      3. 프로젝트 시나리오
2. 설계
   1. ERD
   2. 릴레이션 스키마
3. 구현
   1. 릴레이션
   2. 기능 별 쿼리
   3. 시스템 아키텍처

1. 프로젝트 개요

1.1. 프로젝트 개요

1. 1. 1. 프로젝트 소개

최근 소비자들 간 협력을 통해 다양한 제품을 합리적인 가격으로 구매할 수 있는 효과적인 방법 중 하나로 공동구매가 부상하고 있습니다. 특히, 토스와 같은 대기업이 시장에 참여함으로 공동구매 시장이 점차 성장하고 있습니다.

단체 구매력이 강화되어 저희 플랫폼의 가격 경쟁력이 향상되면 더 많은 소비자 참여를 모을 수 있습니다.

저희 공동구매 플랫폼 PamPam은 다양한 품목들을 제공하여 여러 사람들이 참여해 보다 합리적인 가격으로 원하는 상품을 구매할 수 있습니다. 소비자들은 그룹을 형성하여 상품을 함께 구매함으로 그룹 내에서 혜택을 공유할 수 있습니다.

* 1. 2. 프로젝트 배경

텍스트, 스크린샷, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

최근 국내 근원물가의 증가로 인해 소비자들에게 경제적인 부담을 가져오고 있습니다. 근원물가가 상승할수록 소비자가 필수 소비제를 구매하는 데 더 많은 돈을 지불해야 합니다.

또한, 현재 근원물가는 최근 여러 전쟁과 그로 인한 유가 상승, 에너지 값 상승하여 국내 근원물가 상승세가 지속적으로 상승하고 있습니다.

언제 끝날지 모르는 이러한 물가 상승 배경은 저희 공동구매 서비스가 소비자의 요구사항을 충족시켜줄 하나의 대안이 될 수 있습니다.

1. 1. 3. 프로젝트 시나리오

2. 설계

2. 1. 개념 설계 – ERD

도표, 그림, 스케치, 종이접기이(가) 표시된 사진

자동 생성된 설명

2. 2. 논리 설계 – 릴레이션 스키마

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

3. 구현

* 1. 릴레이션

|  |  |
| --- | --- |
| 구매자 회원가입 테이블 | CREATE TABLE Consumer (  Consumer\_ID VARCHAR(45) PRIMARY KEY NOT NULL UNIQUE,  Consumer\_PW VARCHAR(30) NOT NULL,  Consumer\_Name VARCHAR(20) NOT NULL,  Consumer\_Addr VARCHAR(50) NOT NULL,  Consumer\_pNum VARCHAR(15) NOT NULL,  SocialLogin VARCHAR(30) NULL DEFAULT '소셜 로그인 사용 안함'  ); |
| 판매자 회원가입 테이블 | CREATE TABLE Seller (  Consumer\_ID VARCHAR(45) PRIMARY KEY NOT NULL,  Consumer\_PW VARCHAR(30) NOT NULL,  Consumer\_Name VARCHAR(20) NOT NULL,  Consumer\_Addr VARCHAR(50) NOT NULL,  Consumer\_pNum VARCHAR(15) NOT NULL,  Business\_Number VARCHAR(30) NOT NULL  ); |

|  |  |
| --- | --- |
| 상품 테이블 | CREATE TABLE Product (  Product\_ID INT AUTO\_INCREMENT PRIMARY KEY NOT NULL UNIQUE,  Image VARCHAR(100),  Name VARCHAR(20) NOT NULL,  Information TEXT(2000),    Seller\_ID VARCHAR(45),  Category\_ID INT,  FOREIGN KEY (Seller\_ID) REFERENCES Seller(Seller\_ID),  FOREIGN KEY (Category\_ID) REFERENCES Category(Category\_ID)    ); |

|  |  |
| --- | --- |
| 가격 테이블 | CREATE TABLE Price (  Price\_ID INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,  Price INT NOT NULL,  Number\_of\_People INT NOT NULL,    Product\_ID INT,  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID)  ); |

|  |  |
| --- | --- |
| 카테고리 테이블 | CREATE TABLE Category (  Category\_ID INT AUTO\_INCREMENT PRIMARY KEY,  Category\_Name VARCHAR(40)  ); |

|  |  |
| --- | --- |
| 마감일 테이블 | CREATE TABLE DeadLine (  DeadLine\_ID INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,  Deadline DATE,  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID)  ); |

|  |  |
| --- | --- |
| 좋아요 테이블 | CREATE TABLE Heart (  Consumer\_ID VARCHAR(45),  Product\_ID INT,  FOREIGN KEY (Consumer\_ID) REFERENCES Consumer(Consumer\_ID),  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID)  ); |

|  |  |
| --- | --- |
| 주문 테이블 | CREATE TABLE Order (  Order\_idx INT AUTO\_INCREMENT PRIMARY KEY NOT NULL,  Order\_Date DATE,    Product\_ID INT,  Consumer\_ID VARCHAR(45),  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID),  FOREIGN KEY (Consumer\_ID) REFERENCES Consumer(Consumer\_ID)  ); |

|  |  |
| --- | --- |
| 장바구니 테이블 | CREATE TABLE Shopping\_Cart (  Product\_ID INT,  Consumer\_ID VARCHAR(45),  FOREIGN KEY (Product\_ID) REFERENCES Product(Product\_ID),  FOREIGN KEY (Consumer\_ID) REFERENCES Consumer(Consumer\_ID)  ); |

* 1. 기능 별 쿼리
     1. 회원가입

|  |  |
| --- | --- |
| 회원가입(일반회원) | INSERT INTO Consumer(`Consumer\_pNum`, `Consumer\_Addr`,`Consumer\_Name`,`Consumer\_PW`,`Consumer\_ID`)  values('전화번호', '주소, '백호신3', 'qwer1234', 'choihosin3'); |
| 회원가입(소셜 로그인) | INSERT INTO Consumer(Consumer\_pNum, Consumer\_Addr, Consumer\_Name, Consumer\_PW, Consumer\_ID, SocialLogin)  values('전화번호', '주소', '이름', '비밀번호', '아이디', '소셜로그인'); |
| 회원가입(판매자) | INSERT INTO Seller(Business\_Number, Seller\_pNum, Seller\_Addr, Seller\_Name, Seller\_PW, Seller\_ID) values('사업자번호', '전화번호', '주소', '이름', '비밀번호', '아이디'); |

* + 1. 로그인

|  |  |
| --- | --- |
| 일반 로그인 | select \* from Consumer where Consumer\_ID = '아이디' , Consumer\_PW = '비밀번호'; |
| 소셜 계정으로 로그인 | Select \* from Consumer where SocialLogin='소셜계정'; |

* + 1. 회원 정보 수정

|  |  |
| --- | --- |
| 회원정보를 수정 | UPDATE Consumer SET Consumer\_PW ='비밀번호' WHERE Consumer\_ID='아이디'; |

* + 1. 상품 등록

|  |  |
| --- | --- |
| 판매자 상품 등록 | INSERT INTO Product(Product\_ID, Image, Name, Information) values(상품\_아이디, '상품\_이미지', '상품\_이름', '상품\_정보'); |

* + 1. 상품 검색

|  |  |
| --- | --- |
| 상품 검색 | SELECT \* FROM Product  WHERE Name LIKE '상품\_이름'; |

* + 1. 카테고리 선택

|  |  |
| --- | --- |
| 카테고리 선택 | SELECT \* FROM Product  INNER JOIN Category  ON Product.Category\_ID = Category.Category\_ID  WHERE Product.Category\_ID = 카테고리\_번호; |

* + 1. 쇼핑카트에 상품 추가

|  |  |
| --- | --- |
| 쇼핑카트에 상품 추가 | INSERT INTO Shopping\_Cart(Product\_ID, Consumer\_ID) values(상품\_아이디, '구매자\_아이디'); |

* + 1. 판매자 상품 삭제

|  |  |
| --- | --- |
| 판매자 상품 삭제 | DELETE FROM Product  Where Product\_ID = 상품\_아이디; |

* + 1. 구매자 주문 내역 조회

|  |  |
| --- | --- |
| 구매자 주문 내역 조회 | SELECT \*  FROM Buy  INNER JOIN Product  ON Buy.Product\_ID = Product.Product\_ID  WHERE Consumer\_ID= "구매자\_아이디"; |

* + 1. 관심 상품 목록을 조회

|  |  |
| --- | --- |
| 관심 상품 목록을 조회 | SELECT \*  FROM Shopping\_Cart  INNER JOIN Product  ON Product.Product\_ID = Shopping\_Cart.Product\_ID  WHERE Consumer\_ID= "구매자\_아이디"; |

* + 1. 공동구매 인원수별 가격 조회

|  |  |
| --- | --- |
| 공동구매 인원수별 가격 조회 | Select pricePerPerson  from Price  where Product\_ID=1 and Number\_Of\_People = 총\_인원수; |

* 1. 시스템 아키텍처

텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

DB1 컴퓨터에 Master 설정

DB2 컴퓨터에 Slave 설정

DB3 컴퓨터에 Slave 설정

저희 공동구매 서비스는 특정 이벤트 시간에 많은 구매 행위가 발생합니다. 따라서 Multi-Primary-Mode로 구성하여 하나의 DB 서버에 구매 행위가 몰려서 발생하는 병목현상을 줄일 수 있습니다.

Replication 구성과 비교했을 때 Cluster 구성은 DB 서버 증설이 비교적 간단하고, 유연한 대처가 가능합니다. 하지만, Replication은 DB서버와 스토리지를 같이 증설해야 되기 때문에 특정 시간대에 끊김 없는 서비스를 제공하는 것은 Cluster 구성이 더 유리합니다.

MySQL Cluster를 사용하는 이유

PAMPAM의 특성상 동시간대에 대용량 트래픽이 발생할 수 있을 것이라고 예상했습니다. 따라서 대용량 트래픽이 발생할 경우 여러 개의 서버로 무리 없는 서버 동작을 위해 클러스터링을 사용하였습니다. 또한 우리 서버는 하나의 상품 DB에서 수많은 읽기 작업과 쓰기 작업이 이루어지고 있는데, 읽기 작업이 더 많이 이루어진다고 생각하여 전용 서버를 Single Primary로 구성해주었습니다.

장점

1. 노드들 간의 데이터를 동기화하여 항상 일관성있는 데이터를 얻을 수 있다.
2. 1개의 노드가 죽어도 다른 노드가 살아 있어 시스템을 계속 장애 없이 운영할 수 있다.

단점

1. 여러 노드들 간의 데이터를 동기화하는 시간이 필요하므로 레플리케이션에 비해 쓰기 성능이 떨어진다.
2. 장애가 전파된 경우 처리가 까다로우며, 데이터 동기화에 의해 스케일링에 한계가 있다.

클러스터링 작동원리

1. 1개의 노드에 쓰기 트랜잭션이 수행되고, commit을 실행한다.
2. 실제 디스크에 내용을 쓰기 전에 다른 노드로 데이터의 복제를 요청한다.
3. 다른 노드에서 복제 요청을 수락했다는 신호를 보내고, 디스크에 쓰기를 시작한다.
4. 다른 노드로부터 신호를 받으면 실제 디스크에 데이터를 저장한다.

MySQL CLUSTER 구성 과정

1)리눅스 가상 머신 IP 설정(다른 Slave 서버 2대와 mysql-router 서버도 동일하게 적용)

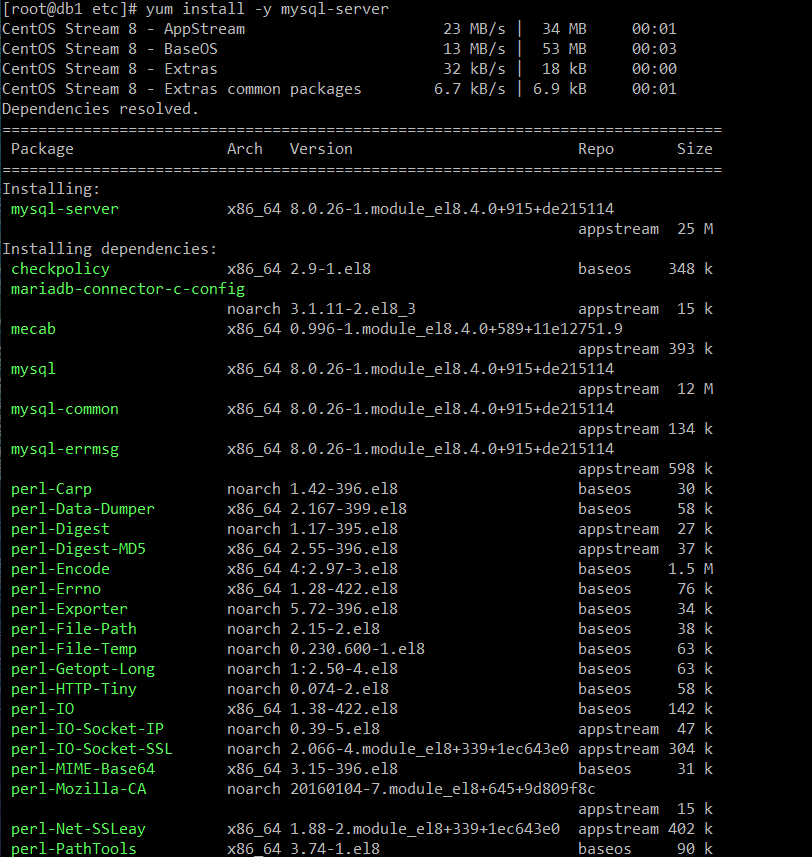
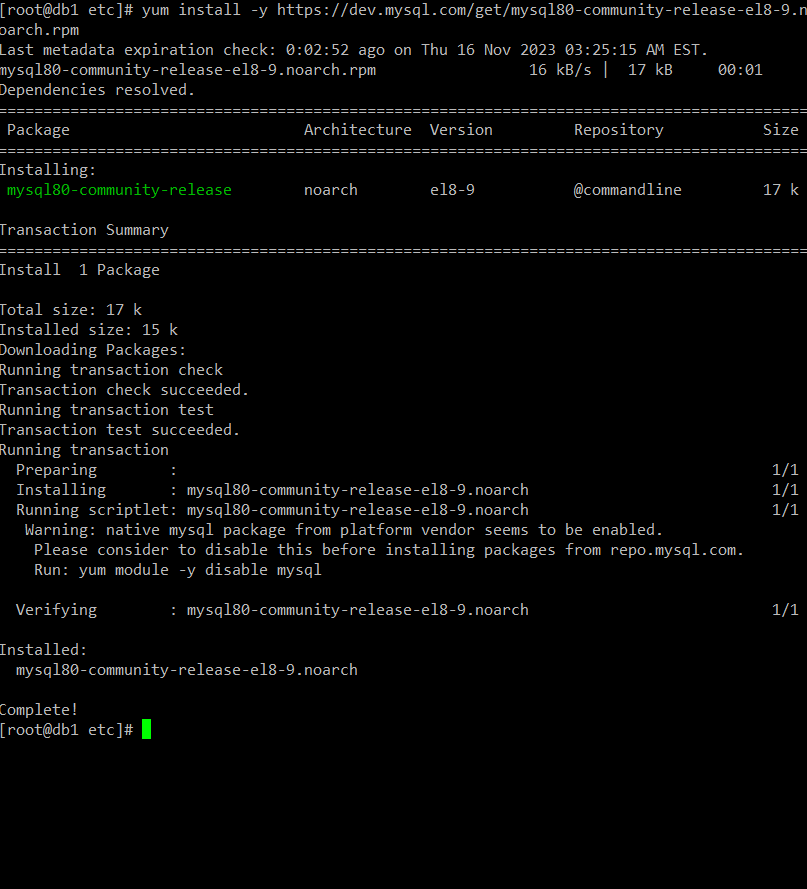
2)네트워크 재시작

3)Hostname 설정

4)리눅스 4대의 hosts 설정

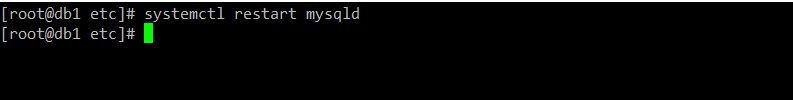
텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

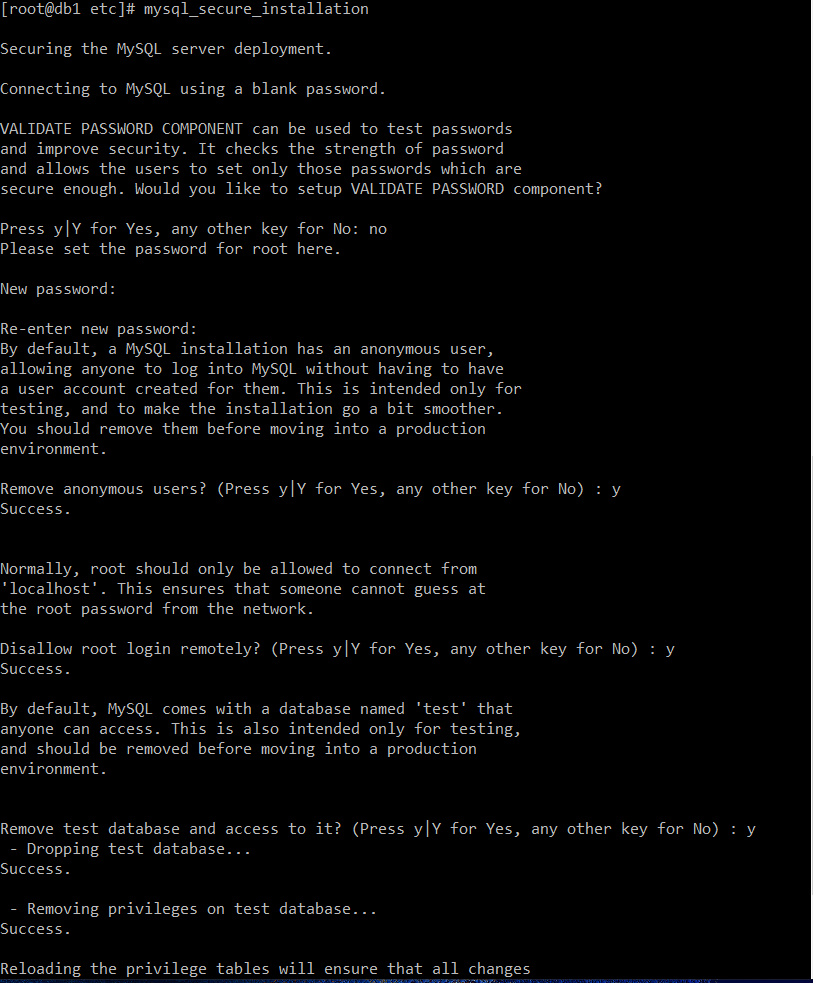


Mysql 서버를 yum 명령어를 통해 설치를 진행한다.

yum install -y mysql-server

mysql 서버 실행

systemctl start mysqld

mysql 초기설정

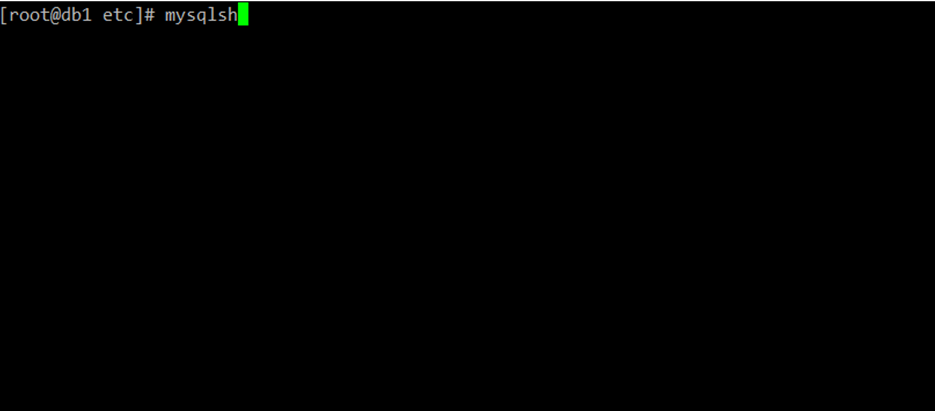
mysql\_server\_installation

no > password > password > y > y > y > y

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

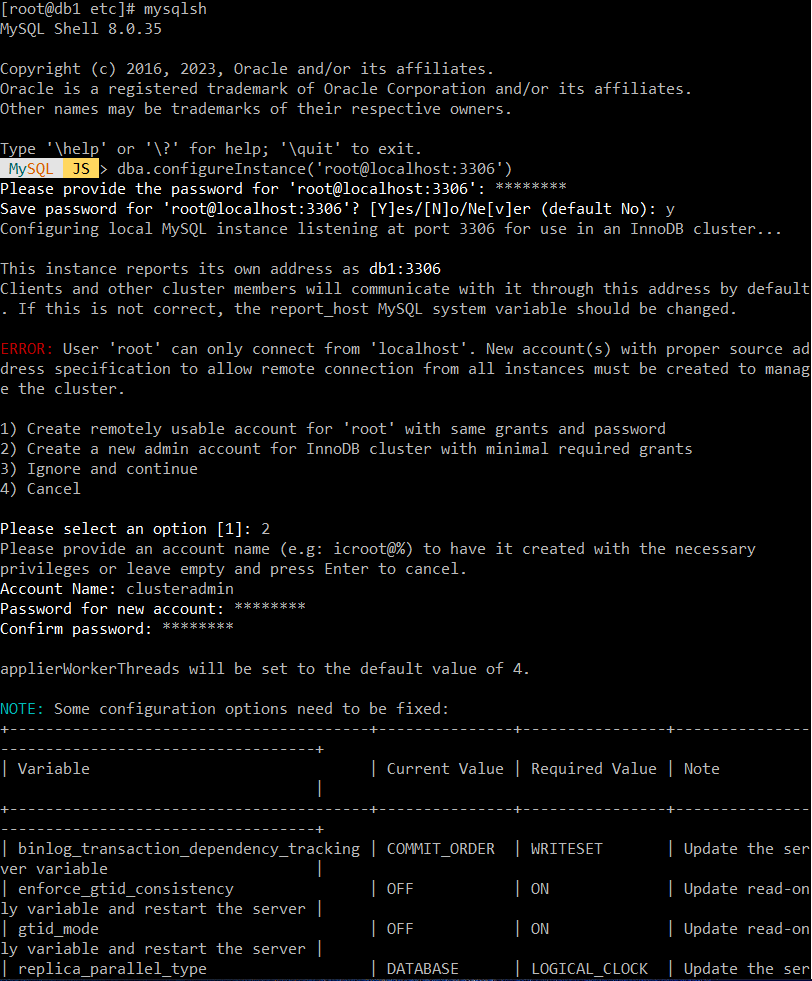
자동 생성된 설명

Mysql-shell 설치

yum install -y mysql-shell

Mysqlsh 실행

mysqlsh

mysqlh 클러스터 구성용 계정 생성

dba.configureInstance('root@localhost:3306')

패스워드 입력

패스워드 저장할거니 ? : Y

2))클러스터 구성할 수 있는 권한을 가지고 있는 새로운 계정을 만들 것이다.

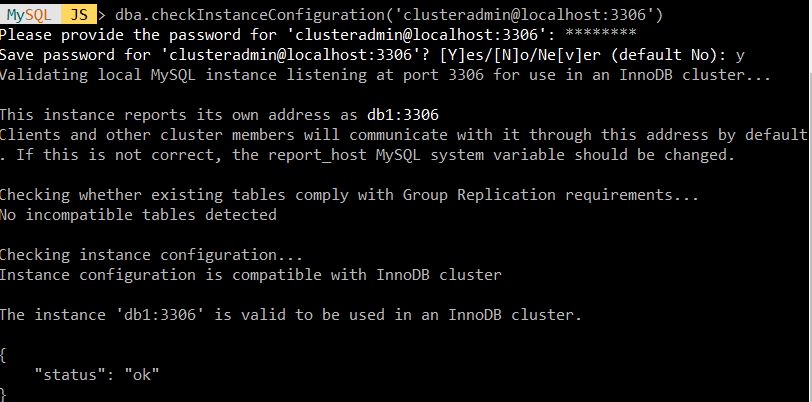
account name = clusteradmin 새로운 계정 이름

pwd = qwer1234 새로운 계정 패스워드

패스워드 확인

필수설정을 바꿀거냐? : y

설정 끝나고 재시작할거냐? : y

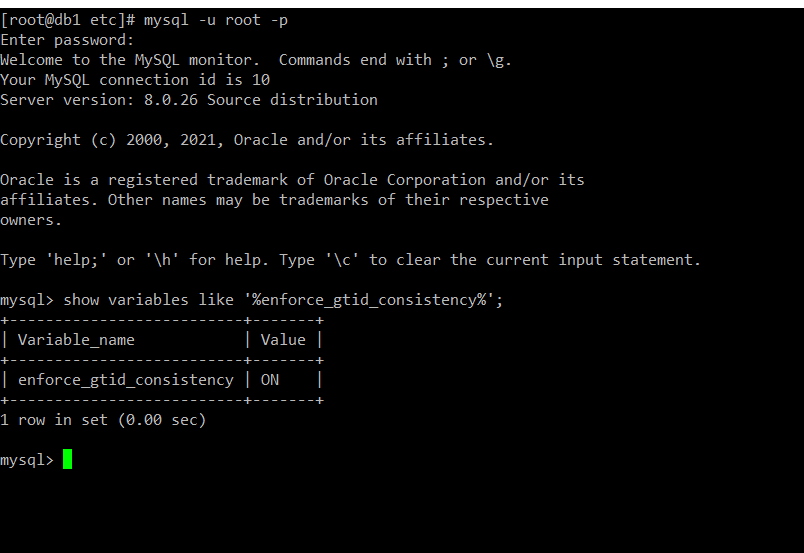


dba.checkInstanceConfiguration('clusteradmin@localhost:3306')

패스워드 입력하고 저장

y

실행후 유효한가에 대한 ok 확인

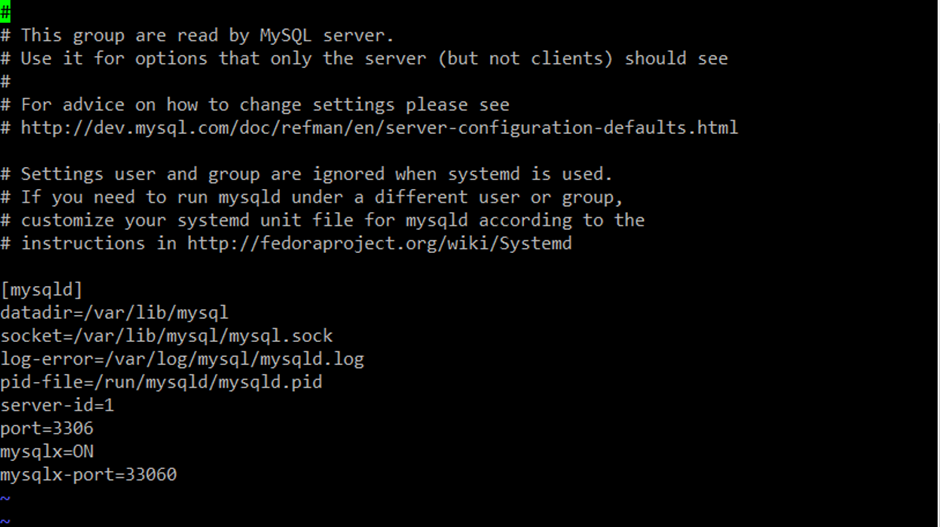


\quit

mysql -u root -p

show variables like '%enforce\_gtid\_consistency%';

실행 결과에서 value에 on으로 된 것 확인

mysql 서버 설정

1. 서버 ID 및 포트 설정

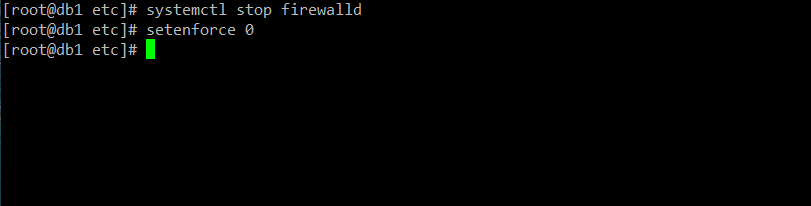
vi /etc/my.cnf.d/mysql-server.cnf

server-id=[각자 다른 숫자]

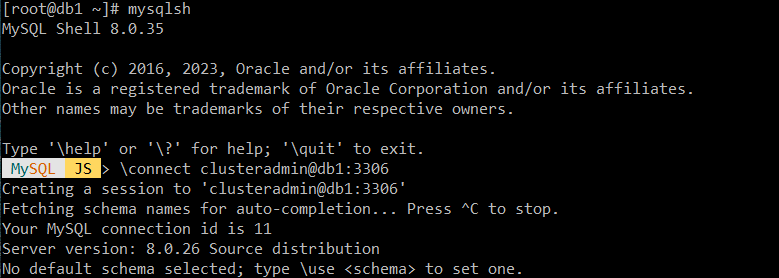
port=3306

mysqlx=ON

 mysqlx-port=33060

 설정후 재시작

방화벽 끄기

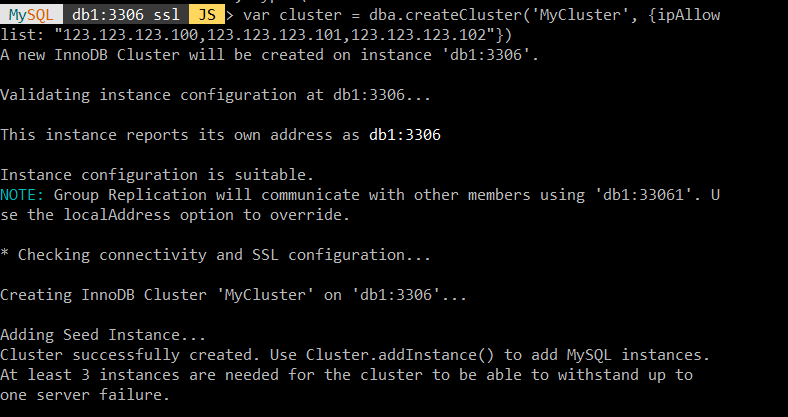


1) clusteradmin 계정으로 로그인

\connect clusteradmin@db1:3306

패스워드 입력

y



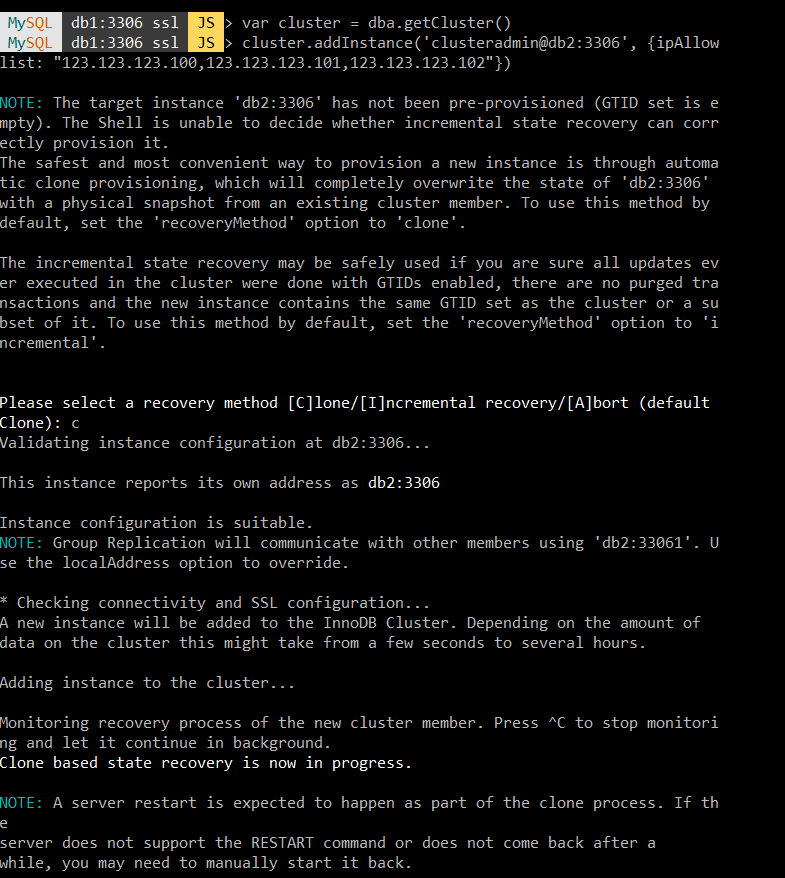
클러스터 생성 var cluster = dba.createCluster('MyCluster', {ipAllowlist: "db1, db2, db3"})

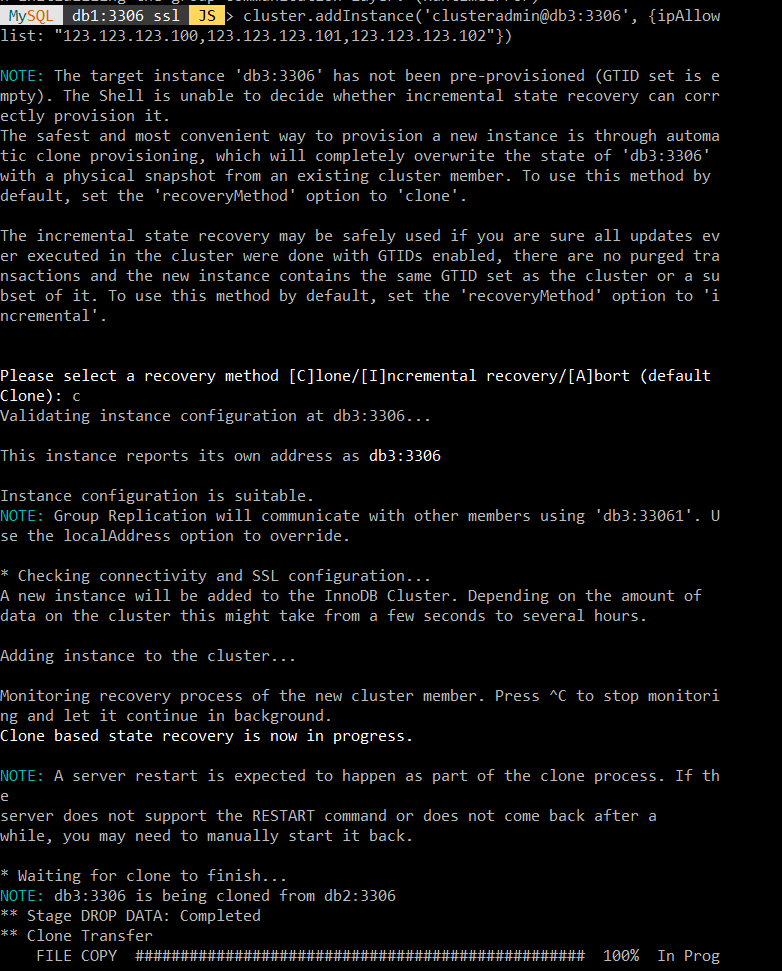
3) 클러스터에 인스턴스 추가

var cluster = dba.getCluster()

cluster.addInstance('clusteradmin@db2:3306', {ipAllowlist: "db1, db2, db3"})

C 클론 방식으로 데이터를 공유하겠다





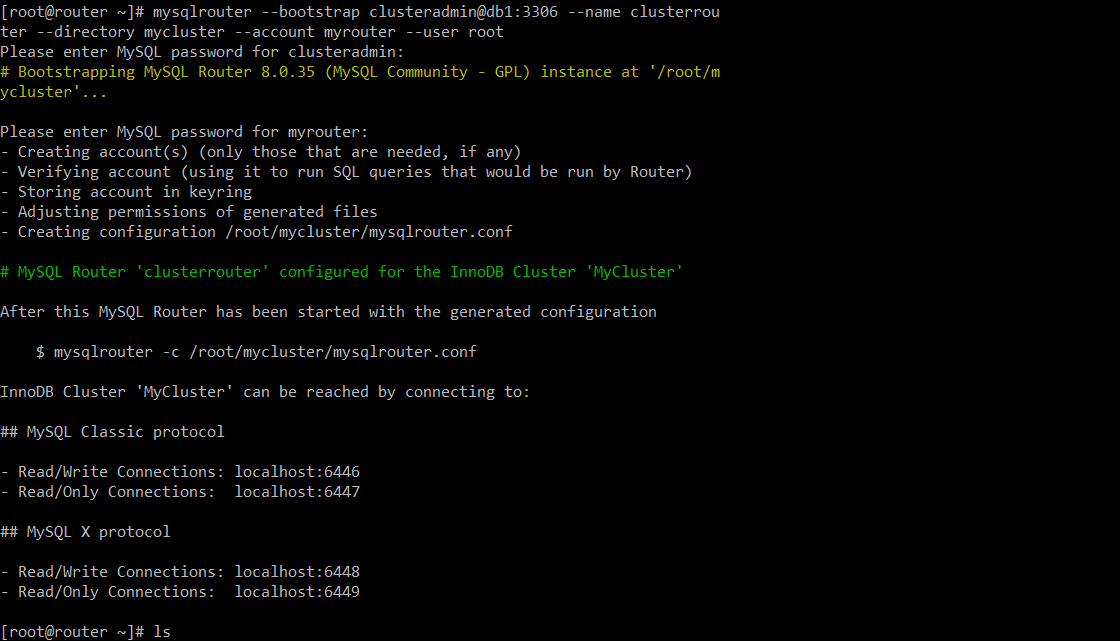
cluster.addInstance('clusteradmin@db3:3306', {ipAllowlist: "db1, db2, db3"})

C 클론 방식으로 데이터를 공유하겠다

클러스터 확인

var cluster = dba.getCluster()

 cluster.describe() 실행했을 때 "topology" 에 db1, db2, db3이 있으면 성공



라우터 구성(router)

1) 방화벽 끄기

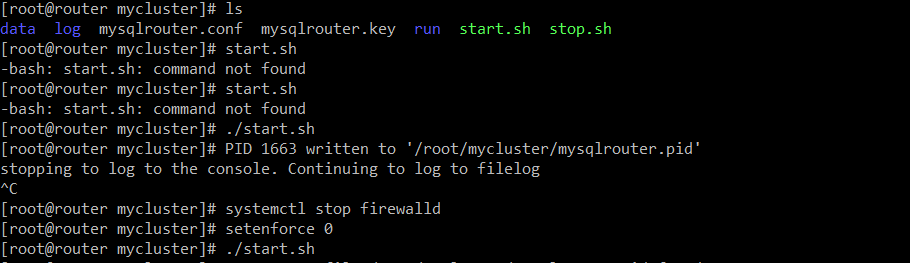
2) myslq-router 설치

yum install -y https://dev.mysql.com/get/mysql80-community-release-el8-9.noarch.rpm

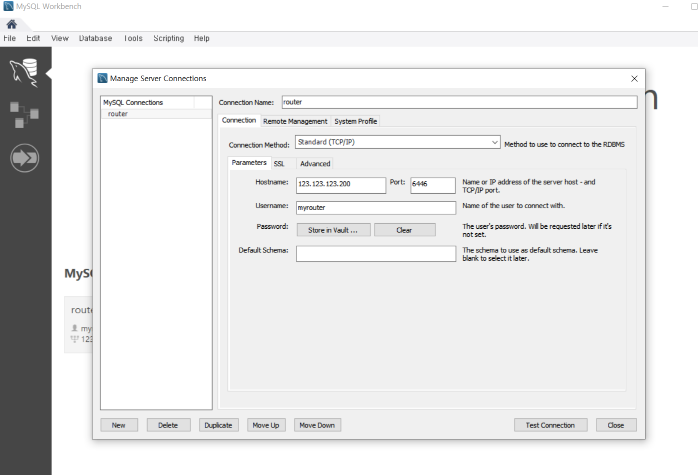
yum install -y mysql-router

3)mysql- router 구성

mysqlrouter --bootstrap clusteradmin@db1:3306 --name clusterrouter --directory mycluster --account myrouter --user root

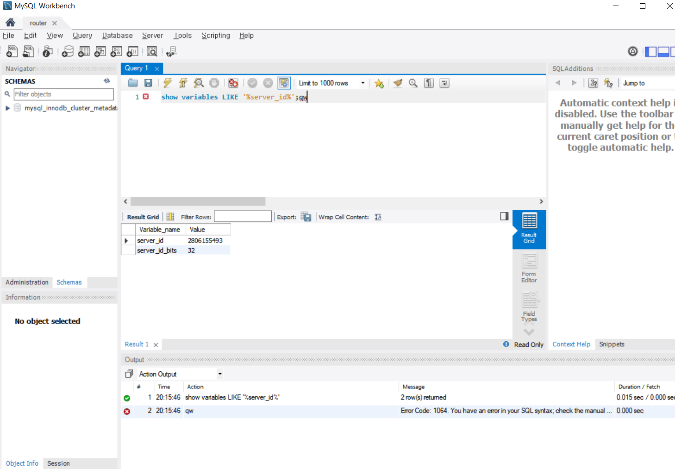
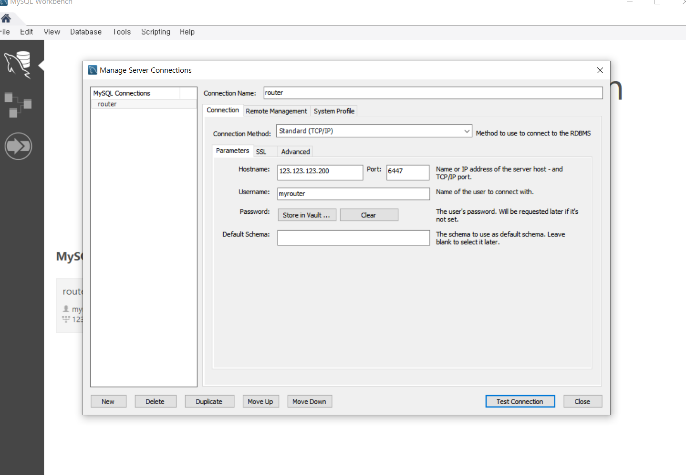


4) 라우터 실행

/root/mycluster/start.sh

텍스트, 소프트웨어, 컴퓨터 아이콘, 웹 페이지이(가) 표시된 사진

자동 생성된 설명6446 실행확인

6447 실행확인